Adaptive Time-Triggered Network-on-Chip-based Multi-Core Architecture: Enhancing Safety and Energy Efficiency

Lehrstuhl für Embedded Systems
Siegen, Deutschland

DISSERTATION
zur Erlangung des Grades
eines Doktors der Ingenieurwissenschaften (Dr.-Ing.)

vorgelegt von
Andrianoelisoa Nambinina, Rakotojaona

eingereicht bei der Naturwissenschaftlich-Technischen Fakultat
der Universität Siegen
Siegen, Januar 2024

Adaptive Time-Triggered Network-on-Chip-based Multi-Core Architecture: Enhancing Safety and Energy Efficiency

Chair for Embedded Systems
Siegen, Germany

DISSERTATION
for the Degree of
Doctor of Engineering

Submitted by

Andrianoelisoa Nambinina, Rakotojaona

Submitted to the Faculty of Natural Sciences and Technology
University of Siegen
Siegen, January 2024

Gutachter

Betreuer und erster Gutachter
Prof. Dr. Roman Obermaisser
Universität Siegen

Zweiter Gutachter
Prof. Dr. Carlos Valderrama
University of Mons

Vorsitzender der Promotionskommission
Prof. Dr. Roland Wismüller
Universität Siegen

Tag der mündlichen Prüfung
26. August 2024

Gedruckt auf alterungsbeständigem holz- und säurefreiem Papier.

Acknowledgements

I would like to express my sincere gratitude to Professor Dr. Roman Obermaisser for his exceptional mentorship. His guidance has been invaluable, fostering my growth as a research scientist. I am particularly grateful for his encouragement of my research and his priceless advice, both in the field and beyond.

I extend my heartfelt thanks to my colleagues for fostering a pleasant atmosphere and a harmonious working environment throughout my tenure. Their unwavering support and collaboration have been instrumental in my professional growth.

I am deeply grateful to my mother, Véronique de Jesus, whose unwavering support during challenging times has been a constant source of strength. I also want to express my gratitude to my beloved wife, Manda. Her enduring patience and overwhelming emotional support have been invaluable throughout my professional career, playing a crucial role in helping me achieve my goals.

My heartfelt thanks go to my sister, Lalaina, and my brothers, Naina, Dapa, and Dora, for their unwavering support. Additionally, I extend my appreciation to my in-laws for their presence and support during critical moments when I was immersed in my research.

Special gratitude is reserved for my friend, Taurai George Rebanowako, for reading my chapters and providing helpful feedback. His insightful comments have significantly improved the quality of this work.

Last but not least, I want to extend my heartfelt appreciation to my family and friends for their unwavering support and encouragement throughout my journey.

Rakotojaona Andrianoelisoa Nambinina

Declaration of Authorship

I hereby declare that I am the sole author and composer of this thesis entitled "Adaptive Time-Triggered Network-on-Chip-based Multi-Core Architecture: Enhancing Safety and Energy Efficiency," and that the work contained herein is presented entirely on my own. Where I have consulted the work of others, this is always clearly stated. All statements taken literally from other writings or referred to by analogy are marked, and their sources are always given. It has been clearly stated where any part of the thesis has previously been submitted for a degree or any other qualification at this University or any other institution. I further declare that I have not submitted this thesis at any other institution to obtain a degree.

List of Publications Related to this Dissertation

1. Rakotojaona Nambinina, Daniel Onwuchekwa, Hamidreza Ahmadian, Dinesh Goyal, and Roman Obermaisser. "Time-Triggered Frequency Scaling in Network-on-Chip for Safety-Relevant Embedded Systems". In: *IEEE 2021 International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON)*. 2021.

2. Rakotojaona Nambinina, Daniel Onwuchekwa, Sabikun Nahar, Darshak Sheladiya, and Roman Obermaisser. "Extension of the LISNoC (Network-on-chip) with an AXI-based Network Interface". In: *IEEE 2022 6th International Conference on Computing Methodologies and Communication (ICCMC)*. 2022.

3. Daniel Onwuchekwa, Josepaul Paulachan, Rakotojaona Nambinina, and Roman Obermaisser. "Time-triggered Network Interface Extension for the Versal Network-on-Chip". In: *IEEE 2023 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*. 2023.

4. Rakotojaona Nambinina, Daniel Onwuchekwa, and Roman Obermaisser. "Latency-Aware Frequency Scaling in Time-Triggered Network-on-Chip Architecture". In: *IEEE 2023 7th International Conference on Computing Methodologies and Communication (ICCMC)*. 2023.

5. Rakotojaona Nambinina, Veit Wiese, Pascal Muoka, Daniel Onwuchekwa, and Roman Obermaisser. "Adaptive Time-Triggered Network-on-Chip Architecture: Enhancing Safety". In: *IEEE 3rd Smart GenCon 2023*. 2023.

Abstract

Real-time computing systems are designed to meet strict timing constraints and respond to events or inputs within specified 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 resource allocation in time-triggered systems. Static resource allocation offers valuable advantages in terms of system dependability by minimizing message congestion and contention, enabling efficient resource usage in network-on-chip (NoC) architectures. This is achieved through the pre-allocation of resources and scheduling of tasks, resulting in improved system throughput and reduced jitter. The time-triggered concept in NoC architectures provides precise knowledge about the permitted points in time for message exchanges between cores, serving as a fundamental building block for fault containment, real-time support, and enhanced system performance.

While static resource allocation excels in minimizing congestion and contention and contributes to system dependability, it may pose challenges in accommodating dynamic workloads and evolving requirements. Additionally, it can limit the achievement of fault tolerance, a crucial aspect of ensuring safety in safety-critical systems. To address these limitations, this thesis focuses on developing fault tolerance and energy-saving techniques tailored explicitly for NoC-based multi-core architectures to enhance their safety and energy efficiency.

The main goal is to incorporate fault tolerance mechanisms, such as adaptation and redundancy, into time-triggered systems without compromising the benefits of static resource allocation. The adaptation technique within the NoC is designed to support multiple schedules, allowing the NoC to switch schedules during run-time in response to context events, such as permanent faults in NoC resources (e.g., routers, links, network interfaces, and cores). By dynamically reconfiguring the schedule upon the occurrence of a permanent fault, the faulty component is effectively isolated, and tasks or messages are redistributed to other available resources. This ensures the system's operational continuity despite faults that could lead to message corruption, delays, or losses within NoC resources. This adaptation technique improves the system's safety by providing flexibility in resource allocation without sacrificing the benefits of static resource allocation.

Furthermore, this thesis incorporates seamless redundancy techniques to enhance the system's safety, especially in scenarios involving transient and permanent faults. This technique selectively applies message replication and fusion to safety-critical messages at the network interface, minimizing overhead in non-critical parts of the system. It safeguards critical data from potential failures caused by message corruption, delays, and losses in routers or links during message exchanges.

The thesis also focuses on improving energy efficiency in multi-core chips by providing low-power services. By incorporating time-triggered communication into NoC-based multi-core architectures, deterministic communication is achieved by scheduling the message's injection time and specifying the frequency to be used by each router at different points in time. This predetermined frequency in the schedule allows routers to adjust their frequencies accordingly during their active time and to clock gate the idle routers, enhancing energy efficiency and preserving the deterministic behaviour of the NoC communication.

Moreover, the adaptation techniques in the NoC are used to reconfigure the operating frequency of the NoC based on workload or power requirement variations by switching between schedules, further optimizing energy consumption. Integrating features such as time-triggered capability, adaptation, time-triggered frequency scaling, and seamless redundancy mechanisms into NoC-based multi-core architectures represents a significant advancement over the current state of the art. The results of this work have significant implications for applications relying on high-performance, safe, and energy-efficient multi-core systems in various domains, such as healthcare and transportation.

Kurzfassung

Echtzeit Computersysteme sind darauf ausgelegt, strenge Zeitvorgaben zu erfüllen und auf Ereignisse oder Eingaben innerhalb festgelegter Fristen zu reagieren. Diese Systeme werden häufig in sicherheitskritischen Anwendungen wie Raumfahrt, medizinischen Geräten, industrieller Steuerung und Fahrzeugsystemen eingesetzt. Ingenieure verlassen sich auf verschiedene Terminplanungstechniken, um sicherzustellen, dass zeitliche Vorgaben eingehalten werden. Eine solche Technik ist die statische Ressourcenzuweisung in zeitgesteuerten Systemen. Die statische Ressourcenzuweisung bietet wertvolle Vorteile in Bezug auf die Systemzuverlässigkeit, indem sie die Nachrichtenüberlastung und kontention minimiert und eine effiziente Ressourcennutzung in Network-on-Chip (NoC) Architekturen ermöglicht. Dies wird durch die vorzeitige Zuweisung von Ressourcen und die Terminplanung von Aufgaben erreicht, was zu einer verbesserten Systemdurchsatzrate und verringerter Jitter führt. Das zeitgesteuerte Konzept in NoC-Architekturen liefert präzises Wissen über die erlaubten Zeitpunkte für den Austausch von Nachrichten zwischen Kernen und dient als grundlegender Baustein für Fehleindämmung, Echtzeitunterstützung und verbesserte Systemleistung.

Obwohl die statische Ressourcenzuweisung hervorragend darin ist, Überlastung und Kontention zu minimieren und zur Systemzuverlässigkeit beizutragen, kann sie Herausforderungen bei der Anpassung an dynamische Arbeitslasten und sich entwickelnde Anforderungen darstellen. Darüber hinaus kann sie die Erreichung von Fehlertoleranz beeinträchtigen, einem entscheidenden Aspekt für die Sicherheit in sicherheitskritischen Systemen. Um diesen Einschränkungen zu begegnen, konzentriert sich diese Arbeit darauf, Fehlertoleranz und energiesparende Techniken speziell für NoC-basierte Mehrkernarchitekturen zu entwickeln, um deren Sicherheit und Energieeffizienz zu verbessern.

Das Hauptziel besteht darin, Fehlertoleranzmechanismen wie Anpassung und Redundanz in zeitgesteuerte Systeme zu integrieren, ohne die Effizienzvorteile der statischen Ressourcenzuweisung zu beeinträchtigen. Die Anpassungstechnik innerhalb des NoC ist darauf ausgelegt, mehrere Zeitpläne zu unterstützen, wodurch das NoC während der Laufzeit in Reaktion auf Kontextereignisse wie permanente Fehler in NoC-Ressourcen (z. B. Router, Links, Netzwerkschnittstellen und Kerne) Zeitpläne wechseln kann. Durch die dynamische Rekonfiguration des Zeitplans bei Auftreten eines permanenten Fehlers wird die fehlerhafte Komponente effektiv isoliert, und Aufgaben oder Nachrichten werden auf andere verfügbare Ressourcen umverteilt. Dies gewährleistet die Betriebskontinuität des Systems trotz Fehler, die zu Nachrichtenkorruption, Verzögerungen oder Verlusten in NoC-Ressourcen führen könnten. Diese Anpassungstechnik verbessert die Sicherheit des Systems, indem sie Flexibilität bei der Ressourcenzuweisung bietet, ohne die Effizienz der statischen Ressourcenzuweisung zu beeinträchtigen.

Darüber hinaus integriert diese Arbeit nahtlose Redundanztechniken, um die Sicherheit des Systems zu verbessern, insbesondere in Szenarien mit vorübergehenden und permanenten Fehlern. Diese Technik wendet selektiv Nachrichtenreplikation und fusion auf sicherheitskritische Nachrichten an der Netzwerkschnittstelle an, um Überkopf in nicht-kritischen Teilen des Systems zu minimieren. Dies schützt kritische Daten vor potenziellen Ausfällen aufgrund von Nachrichtenkorruption, Verzögerungen und Verlusten in Routern oder Links während des Nachrichtenaustauschs.

Die Arbeit konzentriert sich auch darauf, die Energieeffizienz von Mehrkernchips durch Bereitstellung von energiearmen Diensten zu verbessern. Durch die Integration der zeitgesteuerten Kommunikation in NoC-basierte Mehrkernarchitekturen wird deterministische Kommunikation erreicht, indem die Einspritzzeit der Nachricht und die Frequenz, die von jedem Router zu verschiedenen Zeitpunkten verwendet werden soll, geplant werden. Diese vorbestimmte Frequenz im Zeitplan ermöglicht es den Routern, ihre Frequenzen während ihrer aktiven Zeit entsprechend anzupassen und die inaktiven Router zu takten, um die Energieeffizienz zu steigern und das deterministische Verhalten der NoC-Kommunikation zu bewahren.

Darüber hinaus werden die Anpassungstechniken im NoC verwendet, um die Betriebsfrequenz des NoC basierend auf Arbeitslast oder Leistungsanforderungsschwankungen durch Wechsel zwischen Zeitplänen neu zu konfigurieren und damit den Energieverbrauch weiter zu optimieren. Die Integration von Funktionen wie zeitgesteuerter Fähigkeit, Anpassung, zeitgesteuerter Frequenzskalierung und nahtlosen Redundanzmechanismen in NoC-basierte Mehrkernarchitekturen stellt eine bedeutende Weiterentwicklung gegenüber dem aktuellen Stand der Technik dar. Die Ergebnisse dieser Arbeit haben bedeutende Auswirkungen auf Anwendungen, die auf leistungsstarken, sicheren und energieeffizienten Mehrkernsystemen in verschiedenen Bereichen wie Gesundheitswesen und Verkehr angewiesen sind.

Contents

List of Abbreviations

ANoC Asynchronous Network-on-Chip

APB Advanced Peripheral Bus

ASIC Application-Specific Integrated Circuit

ATTNoC Adaptive Time-Triggered Network-on-Chip

AU Agreement Unit

AXI Advanced eXtensible Interface

ATMA Adaptive Time-triggered Multi-core Architecture

BE Best Effort

BIST Built-In Self Test

CA Context Agreement

CM Context Monitor

CMOS Complementary Metal-Oxide-Semiconductor

CPU Central Processing Unit

CRC Cyclic Redundancy Check

DDR Distributed Dynamic Routing Algorithm

DTL Diode-Transistor Logic

DVFS Dynamic Voltage and Frequency Scaling

EDC Error Detection Codes

ECC Error Correction Codes

FCR Fault Containment Region

FIFO First In First Out

FPGA Field Programmable Gate Array

GALS Globally Asynchronous Locally Synchronous

GTB Global Time Base

HPB High-Performance Bus

IC Integrated Circuit

I/O Input/Output

IP Intellectual Property

LISNoC LIS Network-on-Chip

LUT Lookup Table

MCS Mixed Criticality Systems

ML-AHB Multi-Layered Advanced High-performance Bus

MPSoC Multi-Processor System-on-Chip

NI Network Interface

NoC Network-on-Chip

NSCNI Non Safety Critical Network Interface

OCP Open Core Protocol

PE Processing Elements

PL Programmable Logic

PS Processing System

QoS Quality-of-Service

RC Rate Constraint

SA Switch Allocation

SCNI Safety Critical Network Interface

SoC System-on-Chip

SPIN Scalable Programmable Interconnection Network

TLM Transaction-Level Modeling

TT Time-Triggered

TTFS Time-Triggered Frequency Scaling

TTNI Time-Triggered Network Interface

TTNoC Time-Triggered Network-on-Chip

TMR Triple Modular Redundancy

VA Virtual channel Arbitration

VC Virtual Channel

VCI Virtual Component Interface

XBAR Crossbar

List of Figures

List of Tables

Chapter 1

Introduction

Over the past few decades, the development of integrated circuits has played a crucial role in driving exponential growth in computing power. This growth can be attributed to Moore's Law, a prediction made by Gordon Moore, co-founder of Intel, in 1965. Moore's Law predicts that the number of transistors on an integrated circuit doubles approximately every 18 months. Remarkably, the semiconductor industry has managed to keep up with this prediction, integrating billions of transistors on a single chip. This increase in transistor count has facilitated the development of more complex devices, including multi-core processors. As the number of cores on a single chip increased, the limitations of shared bus architectures became more apparent due to limited bandwidth, scalability challenges, lack of flexibility, non-deterministic communication, and vulnerability to single points of failure. These limitations have led to the emergence of Network-on-Chips (NoCs) as a new communication paradigm [WCK08]. 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, such as automotive and avionics systems, deterministic communication is crucial to ensure predictable and timely message transmission and reception. 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, static resource 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, a critical aspect for ensuring safety in safety-critical systems. To address these challenges, this thesis introduces the Adaptive Time-Triggered Network-on-Chip (ATTNoC) architecture designed for multi-core systems to balance energy efficiency and safety. This architecture extends the functionality of an event-triggered-based NoC called LISNoC [TUM] 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 it to switch between them in response to context events, such as permanent faults in routers, links, Network Interfaces (NIs), or cores [Obe+19], [MAO18]. This capability enhances safety by isolating faulty sub-components and redistributing tasks or messages to other avail-

able resources. Consequently, communication within the system can continue to operate even when a permanent fault occurs in the NoC resources. Moreover, seamless redundancy is employed in the ATTNoC to enhance its safety, where critical messages can be transmitted via dual channels to tolerate permanent and transient faults in the routers and links. This approach eliminates the need for fully duplicating NoC resources, reducing overhead [Nam+23].

ATTNoC achieves energy efficiency at the router level through Time-Triggered Frequency Scaling (TTFS) techniques [NOO23]. In ATTNoC, the injection times of messages and the frequency with which they are used in each router are scheduled in advance. This allows routers to scale and clock gate the router frequency according to a predefined schedule, improving energy efficiency while preserving the deterministic behaviour of communications in ATTNoC. Furthermore, adaptive techniques in the NoC allows for dynamic frequency adjustments based on events, such as slack and workload variation, by switching between schedules. For example, a dynamic slack event in the cores or NoC may require adjusting the operating frequency and the injection time of messages in the NoC to reduce energy consumption. Combining adaptation, seamless redundancy, and time-triggered frequency scaling techniques in a time-triggered NoC-based architecture can significantly enhance these systems' safety and energy efficiency.

The following section will delve into this work's underlying motivation and contributions.

1.1 Motivation

The following technical developments in embedded systems can be identified as the driving forces behind the architecture used in this thesis.

1.1.1 Trend Towards Mixed Criticality

Mixed-criticality systems (MCSs) have gained attention in various embedded systems domains due to their ability to integrate multiple functions onto a single computing platform. This integration offers benefits such as reduced hardware costs, weight, and energy consumption [BD07], [Col+22]. However, when subsystems with varying levels of importance coexist on a shared platform, a fault-tolerant infrastructure becomes necessary to prevent interference between low-critical and safety-critical subsystems. Designing such systems is challenging, as it requires balancing safety and segregation while maintaining performance for low-critical subsystems. To fulfil these requirements, the platform must incorporate safety features that ensure system-wide segregation and spatial and temporal partitioning. This infrastructure should be capable of managing the complexities associated with both ensuring safety and delivering optimal performance for the various subsystems.

1.1.2 Trend Towards System-on-Chip

The trend towards system-on-chip (SoC) design is driven by the need to integrate complex functionalities onto a single chip. SoCs bring about improved performance, cost efficiency, and lower power consumption by assembling prefabricated components known as intellectual property (IP) blocks on a chip [Lee+05]. These IP

blocks include processors, memory, interfaces, and specialized units. The NoC infrastructure enables efficient communication between these components, providing several advantages such as energy efficiency, reliable data transmission, scalability, and parallel communication. Researchers are constantly improving NoC architectures, power management, and safety enhancements that will significantly impact the future of embedded systems.

1.1.3 Trend Towards Time-Triggered System-on-Chip Architecture

The demand for deterministic behaviour and real-time responsiveness in systems drives the trend toward time-triggered system-on-chip architectures [Obe+08]. These architectures are increasingly favoured in real-time applications that require high reliability and timing predictability. As technology advances, the adoption of time-triggered architectures in safety-critical applications is predicted to grow, significantly shaping the future of system-on-chip design and enabling advanced functionalities in various domains such as transportation and automotive. To address this need, the ATTNoC architecture incorporates time-triggered capability in the network interface of an event-triggered NoC architecture, known as LISNoC [TUM], to support deterministic communication. It allows the underlying NoC to support mixed-criticality applications and communication types, including time-triggered, rate-constrained, and best-effort.

1.1.4 Trend Towards Adaptability in System-on-Chip

Adaptation in time-triggered systems is driven by the objectives of achieving higher energy efficiency, fault recovery, and the ability to respond to changing environmental conditions [Obe+19]. Adaptive SoCs offer the flexibility and dynamic behaviour necessary to address these challenges effectively. Integrating time-triggered NoC architectures with adaptability features plays a crucial role in overcoming the limitations of static resource allocation in time-triggered systems. By incorporating adaptability, the NoC can support multiple schedules and allow the NoC to reconfigure its schedule when a context event occurs, such as a permanent fault in NoC resources. Furthermore, adaptability enables power-saving by adjusting system components' frequencies and voltage based on their activity level.

1.1.5 Trend Towards Redundancy in Multi-core Architecture

The increasing need for enhanced reliability, fault tolerance, and system resilience drives the trend toward redundancy in multi-core architecture [CS18]. Redundancy mechanisms have gained significant interest in NoC architectures. By incorporating these mechanisms in the NoC, permanent and transient faults within the network can be tolerated. However, fully duplicating the NoC resources can lead to increased overheads. Hence, there is a rising demand for low-cost, low-power, and resource-efficient redundancy designs in NoC-based multi-core platforms.

1.1.6 Trend Towards Energy Efficiency in Multi-core Architecture

The demand for longer battery life in embedded devices drives the growing trend toward energy efficiency in multi-core architecture. In this sense, using a configurable platform that can configure the frequency and voltage within the device during run-time is becoming more attractive as it can reduce the required power of the platform [Obe+19]. In NoC-based multi-core architectures, various techniques are introduced to improve power usage within the chip, such as DVFS, power gating, and clock gating. However, the use of such techniques may affect the performance as well as the safety of the systems. To strike a balance between power efficiency and performance, it is essential to consider the impact on deadlines when employing these power-saving techniques. The chosen approach should aim to minimize the occurrence of missed deadlines resulting from frequency and voltage scaling. This consideration ensures that power-saving techniques do not compromise the system's ability to meet critical timing requirements and maintain the safety guarantees for real-time application.

1.2 Contribution

This work focuses on developing an adaptive time-triggered network-on-chip (ATTNoC) for multi-core architectures. The aim is to enhance the system's fault tolerance and energy efficiency. To accomplish these objectives, we have integrated fault tolerance and power-saving techniques into the ATTNoC architecture. By incorporating fault tolerance mechanisms, ATTNoC ensures reliable network functionality in the presence of faults. This is particularly crucial in safety-critical systems, where failures can lead to severe consequences. Furthermore, the power-saving techniques integrated into ATTNoC address the demand for energy-efficient designs for computing platforms with limited power resources. Overall, this research contributes to the state of the art in the following ways:

- This research extends the functionalities of event-triggered-based NoC architectures, such as LISNoC [TUM], by incorporating source-based routing and time-triggered communication. These features enable deterministic communication in NoC, crucial for preventing message conflicts and meeting deadlines in real-time applications. Additionally, the adaptation techniques in ATTNoC provide fault tolerance through dynamic reconfiguration of network schedules. This allows the system to respond to contextual events, such as permanent faults in NoC resources like NIs, routers, cores, and links [Obe+19], [MAO18], [Nam+23]. The research's contribution is particularly valuable for applications that require high fault tolerance and real-time responsiveness, such as automotive, aerospace, medical devices, and industrial control systems.

- Seamless redundancy in ATTNoC: The ATTNoC architecture is extended to support seamless redundancy in the Time-Triggered Network Interface (TTNI). This mechanism enables message duplication and data transmission over a dual channel, ensuring continuous data transmission even during network failures due to transient or permanent faults. The architecture includes two types of network interfaces: Safety-critical NI (SCNI) and non-safety-critical NI

(NSCNI), making it suitable for mixed-critical systems. The SCNI supports redundancy, guaranteeing uninterrupted data flow even in network failures or faults in routers or links. In contrast, the NSCNI, designed for non-safety-critical applications, does not require redundancy. By incorporating seamless redundancy, the ATTNoC architecture provides a more reliable and fault-tolerant solution for applications requiring high safety [Nam+23].

- Router-level frequency scaling: This work introduces a technique called time-triggered frequency scaling (TTFS) [Nam+21], [NOO23]. In the ATTNoC architecture, messages' injection time and the routers' operating frequency are scheduled in advance. This allows routers to adjust their frequency according to a predefined schedule rather than maintaining a static frequency for idle and active routers. In addition to TTFS, clock gating is employed to turn off the clock frequency of idle routers according to a schedule. This further conserves energy. The adaptive approaches implemented in ATTNoC allow dynamic switching between schedules in response to context events. For example, if the system temperature exceeds a certain threshold, the schedule can be adapted to adjust the operating frequency of routers to prevent overheating. This flexibility enables ATTNoC to adapt to changing conditions and optimize energy consumption.

Overall, this research represents an advancement in NoC-based multi-core architecture. The innovative features and capabilities of the ATTNoC architecture can facilitate the development of more reliable and efficient systems for multi-core platforms, which are becoming increasingly crucial in today's technological landscape.

1.3 Structure of Thesis

The thesis is structured as follows: The first chapter introduces the topic, motivation, and contribution of the thesis. Chapter 2 presents the concepts that serve as the background for the thesis and provides a comprehensive discussion of the fundamental concepts and information necessary to understand it. Chapter 3 introduces NoC technology and offers a detailed explanation of an NoC, including its various components. Chapter 4 presents related work that compares several closely related results to the approach used in this thesis. Chapter 5 describes the system model used in this thesis, including the integrated safety and power-saving techniques in the ATTNoC architecture. Chapter 6 presents the energy efficiency and fault tolerance techniques used in the ATTNoC. In Chapter 7, a series of experiments and evaluations are performed to demonstrate how the system meets the requirements defined in this thesis. Finally, the conclusion of this thesis is presented in Chapter 8.

Chapter 2

Basic Concepts

This chapter introduces the basic concepts needed to understand the subsequent sections of the thesis. It covers several topics relevant to the thesis, including multi-core architecture, on-chip communications, real-time systems, time-triggered and event-triggered systems, fault tolerance, and power-saving techniques. Section 2.1 provides an overview of multi-core architectures used as the primary target systems in this thesis. It discusses the basic concepts and principles of multi-core systems, highlighting their benefits and challenges. Section 2.2 focuses on explaining on-chip communication, highlighting the differences between traditional bus-based communication and NoC communication. Section 2.3 gives an overview of real-time systems, introducing basic concepts and different types of real-time systems. Section 2.4 discusses the concepts of time-triggered and event-triggered systems, explaining the differences between these two paradigms of system behaviour and highlighting their respective advantages. The topic of fault tolerance techniques is discussed in Section 2.5, where various methods and strategies are explored to ensure the correctness of the system in case of faults. The final Section 2.6 discusses power-saving techniques, including approaches and methods for optimizing power consumption in computing systems.

2.1 Multi-core Architectures

Multi-core architectures are a design concept that integrates multiple processors or cores on a single chip in computing systems, as depicted in Figure 2.1. This concept aims to improve overall processing performance by processing tasks simultaneously on different cores. It offers several benefits, including improved performance through parallel processing, enhanced scalability to handle increasing workloads, efficient resource usage, and cost efficiency due to better performance per watt [Nae+10], [Fre+10]. In addition, a multi-core architecture provides improved responsiveness, fault tolerance and compatibility with parallel programming models. To benefit from the advantages of a multi-core architecture, efficient programming techniques and algorithms that effectively leverage parallelism are essential. However, the effectiveness of multi-core systems for individual applications depends on the degree of parallelism in the workload. Moving to a multi-core architecture may require software optimizations and redesign to fully use the available cores, which can be complex and time-consuming. On-chip communication is essential for designing a multi-core architecture as it enables efficient data sharing between cores. It also

provides effective inter-core collaboration and synchronization, improving the performance and scalability of multi-core systems.

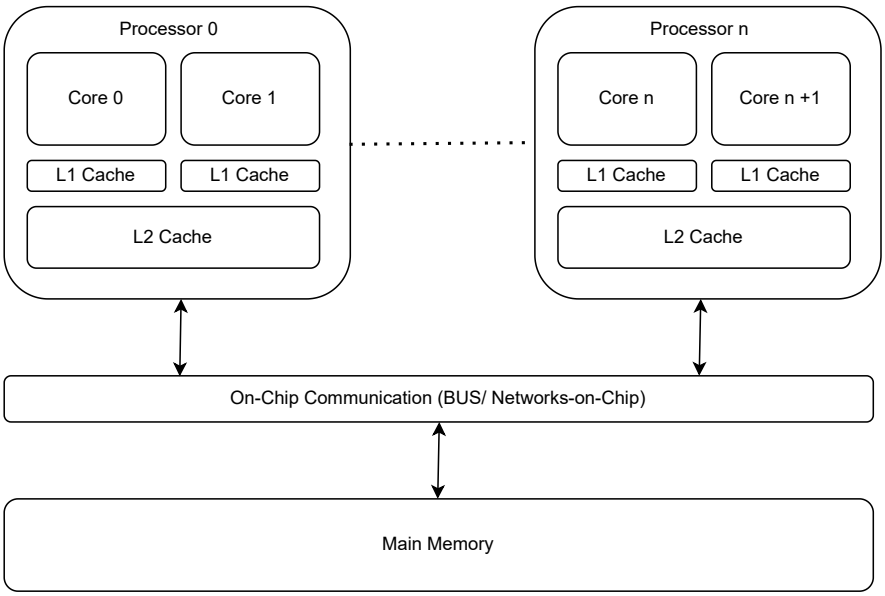


Figure 2.1: Example of multi-core architecture

2.2 On-chip Communication

On-chip communication in multi-core architectures is responsible for data exchange and coordination among multiple processing elements within a System-on-Chip (SoC). Traditionally, on-chip communication was implemented using buses, with all cores connected to a common communication bus. However, as the number of cores increased, bus-based communication became a performance bottleneck due to limited bandwidth and increased contention for accessing the bus. To overcome these limitations, a more efficient approach called Network-on-Chip (NoC) was introduced [SCB09], [Ava+10]. The following sub-sections explain common on-chip communication used in multi-core architectures.

2.2.1 Bus

In the context of on-chip communication, a bus refers to a shared communication path connecting multiple cores within a SoC. It consists of wires or conductors that carry data, control signals, and address information between the cores. The bus architecture typically comprises three main components:

- Data bus: The data bus transfers the actual data between the cores. It facilitates parallel transmission, transmitting multiple bits or bytes simultaneously. The width of the data bus determines the number of bits that can be transferred in parallel during each cycle. For instance, a 64-bit data bus can transmit 64 bits in a single cycle.

- Address bus: The address bus specifies the memory locations or I/O devices the cores intend to access for reading or writing data. It carries the necessary address information required to identify the source or destination of the data.
- Control bus: The control bus plays a crucial role in coordinating the activities of the cores and the bus itself. It is responsible for transmitting control signals that control various operations. These signals include read/write, memory select, interrupt, clock, and various synchronization signals. The control bus ensures proper synchronization and coordination between the cores and facilitates the orderly execution of operations within the system.

The Advanced Microcontroller Bus Architecture (AMBA), developed by ARM, is a widely adopted standard bus in multi-core architectures [STS11]. It is a standardized interface between SoC design components, including multiple cores, memory, and peripherals. The AMBA bus architecture encompasses several specifications, with one of the most commonly used being the Advanced High-Performance Bus (AHB). AHB is designed to connect high-performance cores, memories, and other peripherals, providing a high-bandwidth bus capable of supporting burst transfers, pipelining, and multiple bus masters. On the other hand, the Advanced Peripheral Bus (APB) is a low-power, low-bandwidth bus primarily used for connecting slower peripherals and control functions. It offers a simplified interface suitable for devices with lower data transfer requirements. The Advanced eXtensible Interface (AXI) is available for more demanding requirements. AXI is a high-performance, high-bandwidth bus with a more advanced and flexible interface. It supports burst transfers, out-of-order transactions, multiple outstanding transactions, and separate read and write channels. These AMBA bus specifications, such as AHB, APB, and AXI, provide standardized and scalable solutions for on-chip communications in multi-core architectures. They facilitate efficient data transfer and connectivity between cores and peripherals, enhancing overall system performance and functionality.

2.2.2 Network-on-Chip

Network-on-Chips (NoCs) are communication infrastructures used in multi-core architectures to facilitate efficient data exchange and coordination among multiple processor cores. Unlike traditional bus-based communication, NoCs replace the shared bus with a network composed of interconnected routers. This approach offers improved scalability, higher bandwidth, lower latency, and enhanced overall system performance. The primary components of an NoC architecture include routers, links, network interfaces, and cores, as depicted in Figure 2.2.

- Processing elements: Refers to the computing units or cores within the multi-core architecture responsible for executing the actual processing tasks. Depending on the design, these processing elements can be general-purpose processor cores, specialized accelerators, or a combination of both.
- Routers: Routers are integral components of an NoC architecture and are crucial in forwarding data packets between cores. Each core in the multi-core architecture is connected to a router. Routers have multiple input and output ports to receive and forward data packets. They use routing algorithms

to determine the optimal path for data transmission, considering congestion, latency, and available bandwidth.

- Links represent the physical or logical connections between two routers or between a router and a network interface within the NoC architecture. These links define the communication paths to transfer data between the components. A link typically consists of one or more wires or physical channels that carry data signals, control signals, and synchronization signals. Depending on the specific NoC architecture, links can be point-to-point or shared among multiple components.

- Network interfaces: Network interfaces serve as the interfaces between the processing cores and the NoC architecture. Each processing core in the multi-core architecture is connected to a network interface, which acts as the entry and exit point for data packets entering and leaving the NoC. The network interface handles the conversion of data packets between the format used by the core and the format required by the NoC, ensuring compatibility. It encapsulates data from the core into packets and transmits them to the NoC, and vice versa, when receiving packets from the NoC, it extracts the data and delivers it to the corresponding core for processing.

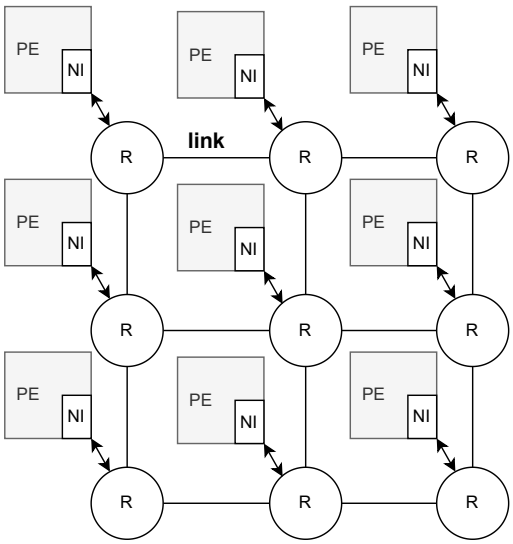


Figure 2.2: 3x3 mesh NoC architecture

NoC architectures can take various forms, such as mesh, torus, and tree-based, tailored to specific design requirements and constraints. Unlike traditional bus-based communication, NoCs provide several advantages, including scalability, higher bandwidth, reduced latency, fault tolerance, and power efficiency [AS14]. These benefits improve system performance, responsiveness, reliability, and energy optimization. Scalability is a crucial benefit of NoCs as they employ a network-based approach that easily accommodates numerous cores. The parallel and concurrent communication in NoCs enables efficient system scaling as the number of cores increases. NoCs offer higher bandwidth than buses, allowing faster data transmission. This increased bandwidth enhances system performance by facilitating efficient and timely data exchange between cores. Addressing latency in communication is another strength of

NoCs. By using multiple parallel paths and efficient routing algorithms, NoCs significantly reduce communication latency. This results in faster data transmission and improved system responsiveness. Fault tolerance is an additional advantage of NoCs. By incorporating fault tolerance mechanisms, NoCs ensure reliable communication even in the presence of faulty connections or routers. This enhances system reliability and enables uninterrupted operation despite failures. Power efficiency is a notable aspect of NoCs in multi-core architectures. By selectively activating routers and links based on communication demands, NoCs reduce overall power consumption. This optimization of energy usage contributes to extended device battery life and improved energy efficiency. In summary, NoCs provide a versatile and efficient communication infrastructure for multi-core architectures, offering scalability, higher bandwidth, reduced latency, fault tolerance, and power efficiency [AS14]. These benefits collectively enhance system performance, responsiveness, reliability, and energy optimization.

2.3 Real-time Systems

Real-time systems are computer systems designed to operate within specific time constraints, processing and responding to events as they occur. These systems are crucial for time-critical applications that require accurate and timely data processing and event handling [TH99]. In a real-time system, proper functioning relies not only on the correctness of the computation but also on the timeliness of its execution. To ensure timely execution, the system must process data, respond to events, and deliver results within predefined deadlines. Failure to meet these deadlines can result in severe consequences, such as equipment failure, loss of life, or financial loss. Three types of real-time systems are commonly recognized: Hard real-time systems, soft real-time systems, and firm real-time systems [AA14], [AGMK94], [LC06].

- Hard real-time systems have strict deadlines that must be guaranteed, and the system must ensure that tasks are completed within the specified time frame. An example of a hard real-time system is the autonomous vehicle control system used in self-driving vehicles. These systems rely on timely and accurate vehicle control, where failure to meet strict deadlines can lead to accidents or loss of vehicle control.

- Soft real-time systems are flexible when meeting deadlines and can tolerate some processing delays. However, maintaining overall system performance is still crucial. An example of a soft real-time system is a multimedia streaming application, like a video streaming service. While ensuring uninterrupted delivery of data packets and maintaining a smooth data stream is essential, occasional delays or buffering can be tolerated without significantly impacting the user experience. The system can recover from such delays and continue streaming seamlessly.

- Firm real-time systems are designed to handle tasks with specific deadlines and ensure they are completed on time. These systems aim to balance the need for strict timelines with some flexibility. An example of a firm real-time system is an online reservation platform, like an airline ticket booking system.

While immediate transaction processing is essential, occasional delays can be tolerated as long as they are infrequent and not excessively prolonged.

2.4 Event-triggered and Time-triggered Systems

In embedded real-time systems, processing and communication activities can be divided into two categories based on the triggering of tasks and messages. These categories are often referred to as event-triggered and time-triggered systems.

- Event-triggered systems: Tasks and messages are activated in response to specific events or occurrences. The events can originate from activities within the computer system (e.g., task termination) or state changes in the natural environment (e.g., alarm conditions indicated by a sensor element). Event-triggered systems are often used in interactive applications like web applications, mobile applications, and desktop user interfaces. Examples of event-triggered systems include web-based chat applications, gaming systems, and intelligent home automation systems [Mur+15].

- Time-triggered systems schedule tasks and messages according to a predetermined schedule. This ensures that tasks and messages are executed and sent at specific times, which provides predictable behaviour. Time-triggered systems are commonly used in safety-critical applications such as aircraft controls, medical equipment, and industrial automation systems [Mur+15].

Both event-triggered and time-triggered systems have advantages and are suitable for different applications. Event-triggered systems offer flexibility and adaptability to changing conditions, while time-triggered systems offer predictability and determinism. The choice between these two categories depends on the embedded system's requirements and the application type.

2.5 Faults and Fault Tolerance in Multi-Core Architectures

Multi-core architectures are increasingly used in embedded devices across healthcare, transportation, and telecommunications sectors, showcasing their advanced capabilities. It is of utmost importance to ensure the reliable functioning of these architectures due to the potential consequences of failures. While some may have catastrophic outcomes, not all failures result in such severe consequences. The impact of a failure depends on the application, task criticality, and preventive measures. In critical systems, like healthcare or transportation, failures can have serious implications, including safety risks and financial losses. In other scenarios, consequences such as system slowdowns, temporary disruptions, or data loss without significant harm may be less severe. Integrating fault tolerance techniques into the architecture is essential to manage and mitigate potential faults and system failures [Saf+22]. The following sub-sections will discuss crucial definitions and concepts in this area.

2.5.1 Faults

Faults in a multi-core architecture include unexpected deviations that may result from hardware malfunctions, software bugs, or external disturbances. These faults can be classified into different categories depending on their nature and duration. Transient faults are temporary faults or deviations that occur in the system but resolve themselves over time. They are often caused by external factors such as fluctuations in the power supply or electromagnetic interference. These transient faults may affect the system's normal behaviour but are not permanent or of long duration. In contrast, permanent faults are persistent problems until they are corrected. Hardware malfunctions, such as defective components or faulty circuitry, usually cause them. Unlike transient faults, permanent faults do not correct themselves and require intervention or repair to restore proper system function. There is also a category of faults called intermittent faults. Intermittent faults occur sporadically and unpredictably and present challenges in diagnosis and troubleshooting. They can occur as transient faults that resolve themselves over time or as permanent faults that require corrective action. The unpredictable nature of intermittent faults makes them challenging to identify and troubleshoot effectively. It is important to note that faults, regardless of their nature, can cause errors within the system. These errors can propagate through the system and affect various components or processes. Over time, the propagation of errors can lead to system failures where the system can no longer reliably perform its intended functions [Jab+17].

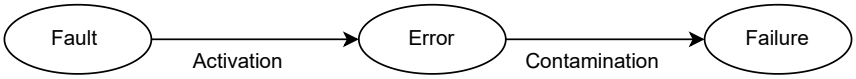


Figure 2.3: Fault-error-failure mechanism

2.5.2 Dependability

Dependability refers to the reliability and trustworthiness of a computing system in consistently providing the intended functionality, even in the presence of faults [ALR01]. It encompasses several important attributes:

- Availability: Refers to the ability of the system to remain accessible and operational when needed. It ensures the system is consistently available to perform its intended tasks without excessive downtime or interruption. System failures must be minimized to achieve high availability, and measures must be taken to ensure rapid recovery from failures.

- Reliability: Refers to the consistent performance of the system in performing its intended function over a period of time without failures or errors. It ensures that the system operates correctly and consistently and produces accurate results as expected. Reliability measures focus on reducing the likelihood of failures and addressing potential problems affecting system performance.

- Safety and security: Safety and security are critical reliability aspects. Safety is about protecting the system and its users from hazards or risks that could cause physical harm. On the other hand, security focuses on protecting the

system from unauthorized access, malicious attacks, and data breaches. Safety and security measures are essential for maintaining the system's integrity and trustworthiness and protecting users' privacy and sensitive data.

Fault tolerance techniques are used to achieve reliability. These techniques aim to mitigate the effects of faults, such as hardware failures or software errors, and ensure system stability and resilience. By incorporating fault tolerance mechanisms, the multi-core architecture can operate reliably and safely, minimizing the impact of faults and maintaining the intended functionality.

2.5.3 Fault Tolerance Techniques

Fault tolerance techniques refer to the strategies and mechanisms used to ensure the continued operation and reliability of a system or application when faults or failures occur. These techniques aim to minimize the impact of faults and prevent them from causing system-wide failures or data loss. They can be divided into hardware-based and software-based approaches.

Hardware-based fault tolerance techniques

Hardware-based fault tolerance techniques involve using redundant components and mechanisms to detect and correct faults. Examples include:

- Redundant processors: Multiple processors are used, and tasks are distributed among them. The workload can be seamlessly transferred to the remaining processors if one processor fails.
- Redundant memory modules: Duplicate memory modules store the same data, enabling error detection and recovery if one module fails.
- Redundant communication channels: Multiple communication channels are established, and data is transmitted over different paths. If one channel has a fault, the system can switch to another one.

Software-based fault tolerance techniques

Software-based fault tolerance techniques focus on fault detection, handling, and recovery mechanisms implemented at the software level. Examples include:

- Error checking and correction codes: Additional data (such as parity bits) are added to transmitted data to detect and correct errors caused by bit flips or noise.
- Error detection algorithms: Various algorithms identify errors, such as checksums or cyclic redundancy checks (CRC), enabling error detection and subsequent recovery actions.
- Error recovery protocols: Protocols are designed to handle error recovery, including retransmission of lost or corrupted data, checkpointing mechanisms, or redundancy management.

Software-based fault tolerance techniques often work with hardware-based techniques to provide comprehensive fault tolerance. By combining the two approaches, systems can detect and recover from faults at multiple levels, increasing reliability and resilience.

2.6 Power Saving Techniques in Multi-core Architectures

Power-saving techniques refer to strategies and methods used in multi-core architectures to reduce power consumption while maintaining or optimizing system performance [Jin+15]. In multi-core architectures, power-saving techniques are critical for improving energy efficiency and extending battery life in embedded systems. Here are some commonly used power-saving techniques:

- Dynamic Voltage and Frequency Scaling (DVFS): DVFS adjusts each component's operating voltage and frequency based on the workload. By dynamically scaling these parameters, the respective component can be operated at lower power during periods of low activity, which saves energy [CKP18].
- Clock gating: With clock gating, clock signals for unused or underutilized components within the multi-core architecture are selectively switched off. This technique prevents unnecessary power consumption by eliminating clock cycles in non-active parts of the system [KKD18].
- Core Shutdown: In scenarios where fewer cores can handle the workload, unused cores can be temporarily shut down or put into a low-power state. This technique reduces power consumption by disabling unused cores until their processing power is needed [Sei+09].
- Task Migration: Active tasks are dynamically shifted between cores to achieve load balancing and energy efficiency in task migration. Distributing the workload evenly across cores can save power by minimizing the use of cores with higher energy consumption.
- Power Gating: Power gating completely shuts down power to unused components or entire cores. This technique reduces static power consumption by cutting power to components not actively used [KKD18].
- Dynamic Power Management: Dynamic Power Management techniques intelligently adapt power consumption based on real-time workload demands. These techniques ensure efficient power usage without sacrificing performance by dynamically adjusting operating parameters and optimizing power allocation [Tan+14].

Chapter 3

Network-on-Chip

This chapter provides a comprehensive overview of Network-on-Chip (NoC) architectures. It covers the crucial elements of NoC design, including network topologies, on-chip switch architecture, switching methods, routing algorithms, and the critical issues related to deadlock and livelock. The chapter begins with an overview of NoC topologies and discusses their benefits and challenges. It then explains the network interface in NoCs and the generic on-chip switch architecture. It also provides insight into the internal workings of NoC switches. Various switching methods, such as store-and-forward, wormhole, and circuit switching, are discussed. The concept of virtual channels, which improves the performance and flexibility of packet switching, is also discussed. In addition, the chapter provides an overview of the routing algorithms used in NoCs. It covers deterministic, adaptive, and stochastic routing algorithms and explains their underlying principles, advantages, and limitations. Moreover, the chapter highlights the critical problems of deadlock and livelock in NoC architectures. It explains these phenomena' concepts, causes, and effects and presents strategies and techniques for avoiding and limiting deadlocks and livelocks.

3.1 Network-on-Chip Basics

The NoC is a communication infrastructure integrated into the System-on-Chip (SoC). Its purpose is to establish connections between components, whether they are heterogeneous or homogeneous. The NoC consists of several key features, including processing elements (PEs), routers, links, and network interfaces (NIs), as shown in Figure 3.1. PEs are functional units within the NoC, including cores, gateways, storage, or I/O units. Routers are key intermediaries responsible for routing and directing the flow of packets within the NoC, enabling communication between PEs. Links establish physical connections between routers. NIs serve as interfaces between the PEs and the NoC. They act as a bridge between the internal communications infrastructure of the PEs and the NoC, enabling the PEs to send and receive packets within the NoC. NIs perform tasks such as packet creation, packet injection from the sender PE into the NoC, and reception of packets from the NoC for delivery to the appropriate destination PE.

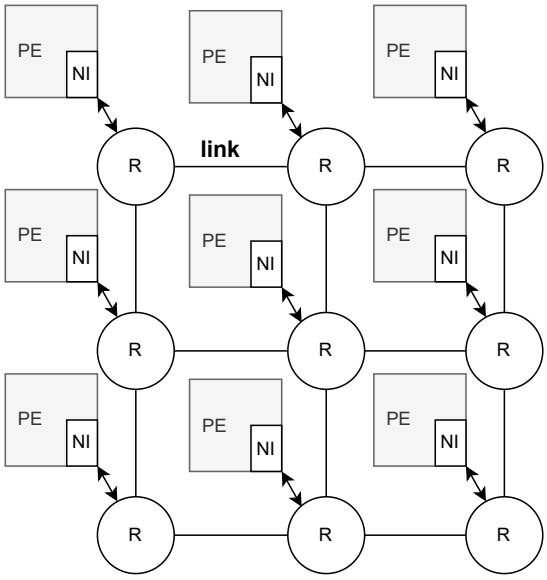


Figure 3.1: 3x3 mesh NoC architecture. PE: Processing elements, NI: Network interface and R: Router

3.1.1 OSI Layers in a NoC

In an NoC architecture, the protocol stack plays a crucial role in ensuring efficient and reliable communication between the various components of the system. It is a hierarchical structure consisting of multiple layers with defined functions and responsibilities, as depicted in Figure 3.2. These layers work together to ensure seamless data transmission and coordination within the NoC.

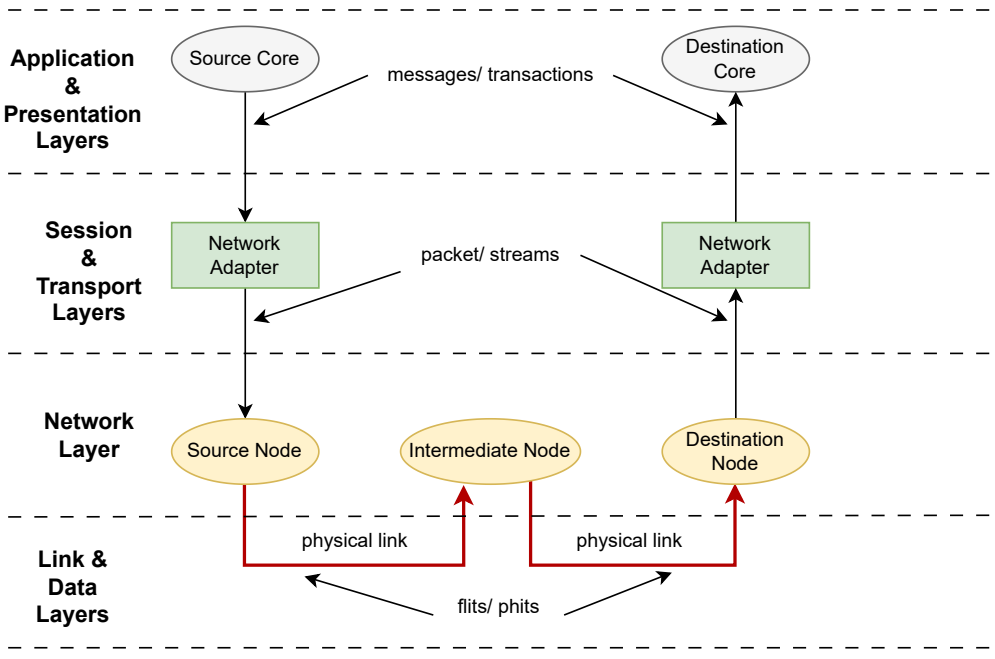


Figure 3.2: ISO/OSI reference model for NoC [Tat+14]

- Application & presentation layers: The application layer provides high-level services and functionalities for the applications running on the NoC. It includes

protocols and mechanisms specific to the application requirements. The presentation layer takes care of data representation and formatting and ensures compatibility and interoperability between the different components in the NoC [Tat+14].

- Session & transport layers: The session layer establishes, maintains, and terminates communication sessions between processing elements (PEs) within the NoC. It manages synchronization, coordination, and session-level protocols to support reliable communication. The transport layer ensures reliable and orderly data transmission between sender and receiver PEs in the NoC. It establishes end-to-end connections, controls data flow, and manages error recovery mechanisms [Tat+14].

- Network layer: The network layer is responsible for routing and addressing within the NoC. It determines the optimal paths and routes for delivering packets over the network based on routing algorithms. In addition, this layer manages the logical organization and configuration of the NoC, including the assignment of addresses to PEs and routing tables in routers [Tat+14].

- Link & data layers: The link layer ensures reliable and error-free data frame transmission between neighbouring NoC routers. It deals with framing, error detection and correction, and flow control. The data layer focuses on the physical transmission of data over the NoC communication, including tasks such as signal coding, modulation, and physical connectivity between routers [Tat+14].

3.2 Network-on-Chip Topology Architecture

The network topology is the arrangement and interconnection of processing elements and routers within a NoC. This topology plays a critical role in defining the structure and organization of the communication network within a SoC. Its purpose is to enable efficient data transmission and communication between the different components within the SoC. The chosen network topology significantly impacts communication efficiency, scalability, fault tolerance, and overall system performance. This section provides an overview of various NoC topology architectures that are commonly used. These architectures include mesh, tree-based, and custom topologies [PT16].

3.2.1 Mesh Network Topology in Network-on-Chip

A mesh topology is commonly used to connect processing elements (PEs) or IP cores on a chip. The chip is divided into a grid-like arrangement of PEs, and each PE is connected to its neighbouring PEs to form a mesh network. The mesh topology allows multiple direct paths between PEs, which makes data communication efficient and reduces message transmission time. There are two types of mesh network topologies: 2D and 3D mesh. In a 2D mesh, the PEs are arranged in a two-dimensional grid, and each PE has four neighbours (north, south, east, and west). In a 3D mesh, on the other hand, the PEs are arranged in a three-dimensional grid, and each PE has six neighbours (north, south, east, west, top, and bottom). The 3D mesh

topology provides many communication paths that support efficient data routing in vertically stacked chip layers. Mesh topologies in NoCs offer several advantages. First, they provide scalable interconnectivity that enables easy expansion of the mesh size by adding more PEs to the grid. The direct point-to-point connections minimize the distance and time required for PEs to communicate. In addition, the mesh structure enables fault tolerance, as data can be rerouted over alternate paths in case of a link failure.

3.2.2 Torus Network Topology in Network-on-Chip

A torus topology is a cyclic, multidimensional structure with a torus shape. It has similarities to a mesh but contains wraparound connections, i.e., the PEs are connected in a loop. This wraparound connectivity allows seamless communication between PEs on opposite torus edges. Torus network topologies can be implemented in 2D or 3D configurations, depending on the desired dimensionality of the connection. The torus structure provides multiple paths for efficient data communication and low-latency connections. Each processing element (PE) is directly connected to its neighbouring PEs, allowing data to traverse the torus in a minimal number of hops. This direct connection enables efficient nearest-neighbour communication, meaning PEs can easily exchange data with their immediate neighbours. In addition, torus topologies exhibit fault tolerance. In case of a link or PE failure, data can be rerouted through alternate paths due to wraparound connections. This redundancy ensures that communication can continue even if certain connections or PEs are unavailable.

3.2.3 Ring Network Topology in Network-on-Chip

A ring network topology is a circular arrangement in which each processing element (PE) or IP core is connected to its neighbouring PEs in a closed loop. Data is transmitted in a unidirectional manner and passes through each PE until it reaches its destination. Each PE is connected to its immediate neighbours to form a loop. The ring structure provides a regular connection pattern, enabling efficient data transmission and low-latency communication. Ring network topologies offer advantages such as balanced load distribution between PEs and simplified routing algorithms. They also have higher fault tolerance than linear topologies because data can be rerouted in the opposite direction if a link or PE fails. However, there are some limitations. Communication between non-adjacent PEs in the ring can result in longer paths and higher latency. In addition, the closed structure can make it difficult to dynamically insert or remove PEs without disrupting the entire network. Despite these challenges, ring network topologies are suitable for specific NoC applications, especially those with regular traffic patterns or those where simplicity is essential to the design considerations.

3.2.4 Star Network Topology in Network-on-Chip

A star network topology is a centralized structure in which all processing elements (PEs) or IP cores are connected to a central hub or switch. Each PE has its link or channel connecting it to the central hub. The hub manages the routing and distribution of data and acts as a central point of control and coordination. The star

structure simplifies routing because all communications pass through the hub, allowing for easy management and control of the network. The hub efficiently aggregates and distributes data to PEs, allowing for flexible routing and data manipulation. It also provides fault isolation, as the failure of one PE or link does not directly affect others. However, star network topologies can have limitations. The central hub can become a single point of failure or a performance bottleneck if communication demands exceed capacity. PEs further away from the hub may experience higher latency. Scalability can also be a challenge. Despite these challenges, star network topologies in NoC designs require centralized control, efficient data distribution, and fault isolation. They find application in systems with high communication coordination or when the central hub provides additional functions such as buffering or protocol conversion.

3.2.5 Tree-based Network Topology in Network-on-Chip

In NoCs, a tree-based network topology is commonly used to connect processing elements (PEs) or IP cores. This hierarchical structure has a central node or root connected to multiple child nodes, which may have child nodes. This arrangement facilitates efficient data communication and resource sharing among PEs. The tree's root node serves as the central hub or controller responsible for managing and coordinating communications within the NoC. The lower-level nodes represent individual PEs or subnets, and the hierarchical structure enables organized data routing and management. Tree-based network topologies offer advantages such as efficient data distribution and centralized control. The hierarchical structure allows the root node to distribute data or control signals to the appropriate child nodes, reducing communication overhead. Centralized control also simplifies the management and coordination of the NoC. However, tree-based topologies can have scalability and fault tolerance limitations. Due to the hierarchical structure, longer communication paths and higher latency may occur for certain PEs. In addition, if the root node fails, the entire NoC may become non-functional.

3.2.6 Irregular or Custom Network Topology in Network-on-Chip

Irregular or custom network topologies are unique and user-defined interconnection schemes to meet specific chip design or application requirements. Unlike predefined topological structures such as mesh or tree, irregular topologies do not follow a fixed pattern or arrangement. Instead, irregular or custom network topologies are designed in NoCs based on several factors. These factors include the communication patterns of the IP cores, desired performance metrics (e.g., latency or bandwidth), power consumption constraints, and available chip area. The goal of using irregular or custom topologies is to tailor the network connections to the specific requirements of the chip design or application. This customization allows for optimized communication and resource allocation while considering the system's unique characteristics and requirements. By deviating from predefined structures, irregular topologies provide flexibility and the ability to optimize communication within the chip. However, designing and implementing custom topologies can be more complex and difficult than predefined topologies. Overall, irregular or custom network

topologies in NoCs enable customized interconnection schemes that better meet the specific requirements and constraints of a particular chip design or application.

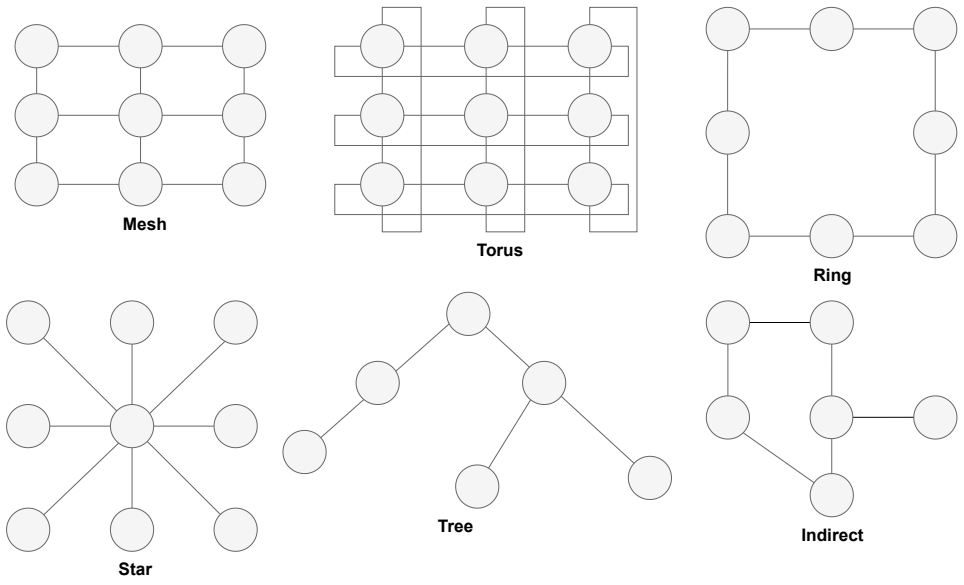


Figure 3.3: Six different common network topologies

3.3 Network Interface

The network interface (NI) module serves as the crucial link between the IP cores and the NoC by accommodating the diverse interface protocols of each IP core. Its primary purpose is to facilitate the separation of computation and communication, enabling independent reuse of the core and communication infrastructure. The NI module, also known as the network adapter, can be further divided into two components: the front-end and the back-end, as illustrated in Figure 3.4:

- The front-end component manages core requests and operates without direct knowledge of the underlying NoC. Typically, this portion is implemented as a socket using widely adopted protocols such as OCP (*OCPIP 2011*), VCI (*VSI Alliance 2011*), AXI (*ARM 2011*), DTL (*Philips Semiconductors 2002*), and others. The front end efficiently handles core communication with the NoC by employing these protocols, allowing seamless integration of different IP cores into the overall system.

- The back-end component is responsible for handling the network protocol within the NI. It performs essential functions such as packet assembly and decomposition, buffer reordering management, synchronization protocol implementation, and providing support to the router for storage, among other responsibilities. By carrying out these tasks, the back end ensures efficient and reliable data transmission within the NoC, optimizing performance and ensuring the integrity of communication.

The combined operation of front-end and back-end components within the NI module is critical to the overall system architecture, enabling effective communica-

tion between IP cores and the NoC while maintaining modularity and reusability of system components.

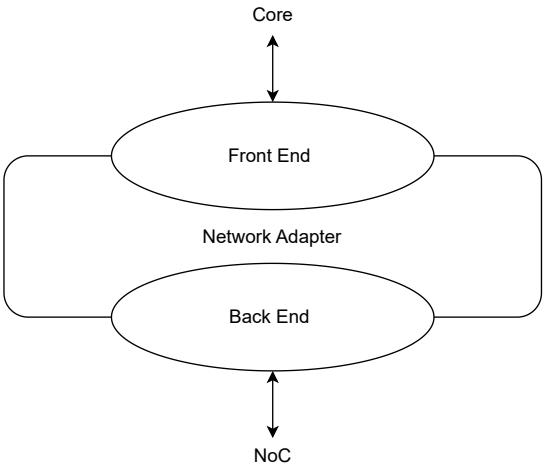


Figure 3.4: Network adapter or network interface

3.4 Generic On-Chip Switch Architecture

A router plays a crucial role in the NoC architecture by forwarding data packets between devices or networks. It usually has P input and P output ports forming a PxP configuration. In the context of 2D mesh and torus topologies, P is set to 5, consisting of four inputs from the cardinal directions (north, east, south, and west) and one input from the local processing element (PE) connected to the NoC router. Depending on the design, these input and output channels can be unidirectional or bidirectional. Buffering is essential in a network router to address congestion, output link contention, and delays caused by router-internal processing tasks such as routing computation. These factors can hinder the smooth flow of data. In NoC routers that use virtual channels, each input port consists of multiple first-in-first-out (FIFO) buffers representing different virtual channels. Figure 3.5 shows these virtual channels, where each input port is equipped with v virtual channels. The control logic serves as the central component of a NoC router. It includes four main elements: Routing computation (RC), the virtual channel arbitration logic (VA), the switch allocation (SA), and the crossbar (XBAR). These elements work harmoniously to ensure efficient operation and data forwarding within the router, as depicted in Figure 3.5.

- The routing computation (RC) unit is an essential component of an NoC router. Its main task is determining the optimal routing path for incoming packets based on the destination address in the packet header. The RC uses several techniques, including deterministic and adaptive approaches, to perform these routing computations. Deterministic routing relies on predefined algorithms such as XY routing, where the routing path is predetermined based on the coordinates of the destination address. On the other hand, adaptive routing dynamically selects the routing path based on the current network conditions, e.g., load balancing or congestion avoidance. The RC operates

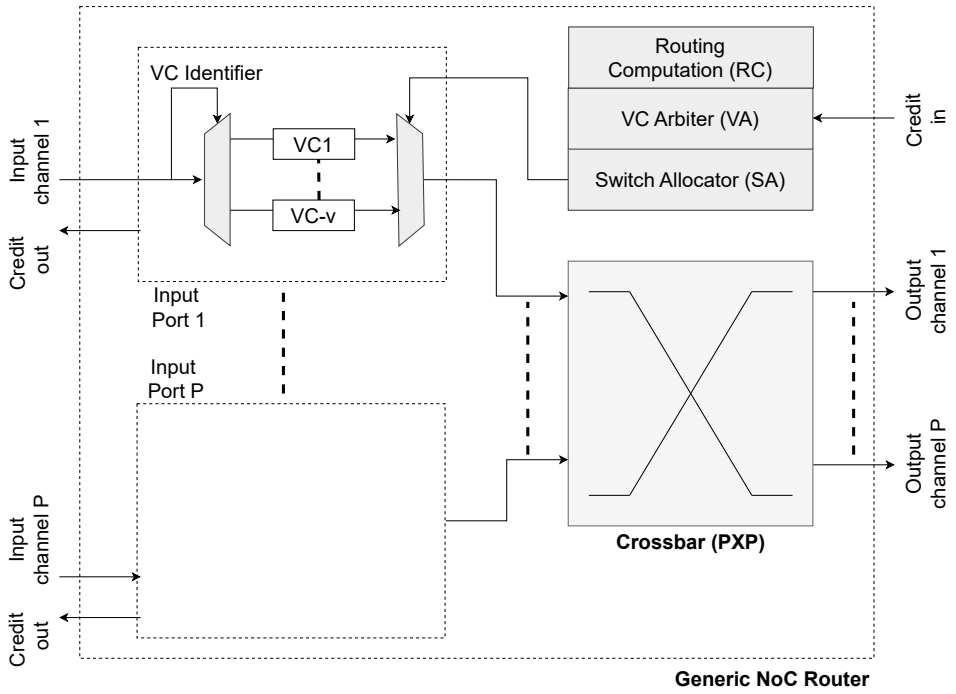


Figure 3.5: Generic one router architecture [Agr21]

packet-by-packet, performing the routing computation once for each packet that passes through the router. In particular, it focuses on the header flit of the packet, which contains essential information about the destination address. By accurately determining the appropriate outbound and virtual channels for the header flit, RC ensures that the packet is efficiently routed to its intended destination within the network.

- Virtual channel arbitration (VA) is a crucial process in NoC routers that determines which packet is allowed to use the physical channel, considering factors such as packet priority and fairness. The primary purpose of VA is to ensure controlled and orderly packet transmission and minimize conflicts and congestion within the router. The process of VA involves several steps. First, requests are generated from the input ports of the router. These requests are then queued in the virtual channel buffers. In the next step, arbitration decisions are made based on the queued requests, considering packet priority and fairness factors. The arbitration decisions determine which packets are approved for transmission. The approved packets are then transmitted over the physical channel. Finally, the VA process includes buffer management to ensure efficient use of resources. By effectively managing the VA process, routers optimize resource usage and provide reliable communication while maintaining high-quality service.

- Switch allocation (SA) is a critical aspect of a generic router architecture as it efficiently allocates available switching resources within the router. Switch allocation includes both input port allocation and output port allocation. Input port allocation determines the output port or ports to which incoming packets should be forwarded based on routing decisions. On the other hand, output port allocation establishes connections in the switching fabric to for-

ward packets to the selected output port. Various mechanisms can be used for switch allocation, including shared memory, crossbar switching, and output queuing. The choice of mechanism depends on factors such as performance requirements and available hardware resources. To optimize switch allocation, advanced scheduling and arbitration algorithms can be used. These algorithms help improve router performance by efficiently managing switching resource allocation.

- Crossbar: In a router architecture, the crossbar plays a central role as a switching fabric that connects multiple input and output ports. Regardless of the number of virtual channels or flits that can be sent from different virtual channel buffers, each input port in the router shares a single crossbar port. This decision allows the size of the crossbar to be kept small and independent of the number of virtual channels. The router architecture efficiently uses switching resources by sharing the crossbar port among the input ports.

The virtual channel router functions through a pipeline, as seen in Figure 3.6. Each packet passes through four pipeline phases: RC, VA, SA, and switch traversal (ST). To facilitate the processing of the packets, they are divided into smaller units called flits, which include a header, a body, and a tail flit. The RC and VA steps are performed exclusively on the head flit since it contains the necessary routing information. Once the RC and VA phases are completed, a dedicated data path is reserved for the packet within the virtual channel router. Subsequently, the body and tail flits follow the head flit through the reserved path. The SA and ST phases are the same for all flits and involve each flit sending a request to SA. If the request is granted, the flit can traverse the crossbar in the next cycle. After the last flit is ejected, the reserved path is released [LGY10].

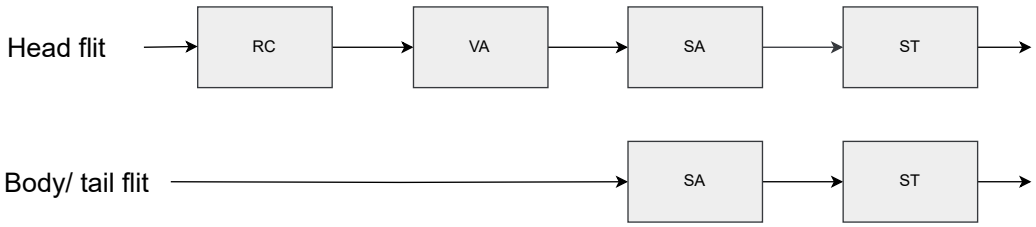


Figure 3.6: Data flow in router

3.4.1 Effects of Fault on Router Pipeline

The router pipeline provides efficient forwarding and processing of packets within a network infrastructure. However, failures within the router pipeline can significantly impact its functionality and performance. This subsection examines the effects of errors in the router pipeline to show the possible consequences and the resulting impact on network operations.

Routing Computation Fault scenario

Faults in the RC unit of an input port can lead to inaccuracies in output port computations. However, the lookahead routing protocol minimizes immediate misrouting

at the current router by correctly directing packets toward the intended output port. This reduces the chances of misrouting at the current router. It is important to note that misrouting can still occur in downstream routers if the incorrect computation remains a valid output port direction. In such cases, the packet may experience additional latency as it is redirected toward the correct destination by the next downstream router. In the case of source-based routing, which relies solely on the source address for routing decisions, faults in the RC unit cannot be rectified. If the incorrect computation results in an invalid output port direction, the packet may become trapped in the downstream router, potentially leading to a deadlock. In such instances, the downstream router may drop the packet, necessitating retransmission later to ensure successful delivery [Sha+20].

Virtual Channel Arbitration Stage Fault Scenario

If a fault occurs in the arbiter of the VA stage, it can lead to two possible scenarios: Miss-allocation or no-allocation of the downstream VC. In the case of miss-allocation, when the downstream VC is already occupied, the subsequent packet overwrites the previous one, causing data corruption. On the other hand, in the case of no allocation, the packet remains in the buffer without progressing further. This situation results in significant performance degradation, potentially leading to a deadlock [Sha+20].

Switch Allocation Stage Fault Scenario

If a fault occurs in the arbiter of the first stage of SA, it blocks packets at the associated input port. As a result, the packets within that input port cannot progress through the subsequent stages of SA. Similarly, a fault in the arbiter of the second stage of SA renders the output port inaccessible, preventing packets from being transmitted to their intended destinations. In both scenarios, the fault causes the entire input port to become permanently isolated, disrupting the flow of packets and hindering their progress within the network. This isolation of the input port results in significant performance degradation, as the affected packets cannot proceed beyond the SA stages. Moreover, the situation poses a potential risk of deadlock, where the network reaches a state where no further progress can be made.

Crossbar Fault Scenario

A fault occurring in a multiplexer within the crossbar can render an output port inaccessible, severely disrupting the overall operation of the router. However, employing adaptive routing or bypassing faulty multiplexers can mitigate the impact of the fault on the router's function. By dynamically adjusting the routing paths or bypassing the faulty components, the router can continue operating, albeit with reduced performance or capacity. Additionally, this adaptive approach enables the router to maintain functionality despite the presence of faulty multiplexers [Sha+20].

3.5 Switching Methodology

In NoC architectures, the switching methodology refers to routers' specific techniques to facilitate forwarding data packets between processing elements (PEs) or

IP cores within the chip. Various switching methodologies are available, each offering different trade-offs in terms of factors such as latency, throughput, complexity, and resource usage. This section overviews several commonly used switching methodologies in NoC designs.

3.5.1 Store and Forward

Store and forward is a widely employed switching methodology in NoC systems. In this approach, the entire data packet is received and stored in the switch's input buffer before being forwarded to the output port. The switch examines the packet's header to determine the appropriate output port. Store and forward switching offers advantages such as error detection and correction capabilities, as the entire packet is received and verified. However, it introduces additional latency due to the need to buffer the entire packet before forwarding.

3.5.2 Wormhole Switching

Wormhole switching is a low-latency switching methodology commonly used in NoC designs. This approach divides data packets into smaller flits (flow control digits), typically consisting of a header, body, and tail. Each flit is transmitted through the switch progressively, starting from the input port towards the output port. The switch examines the flit's header to determine the appropriate routing path and forwards it when the resources are available. Wormhole switching reduces latency by enabling pipelined transmission of flits, but it requires careful flow control to avoid deadlock situations.

3.5.3 Circuit Switching

Circuit switching is a switching methodology commonly used in traditional telecommunication networks. In NoC, circuit switching establishes a dedicated path or circuit between the source and destination PEs for the duration of a communication session. Once the circuit is established, data packets are transmitted along the pre-allocated path without requiring header processing at each intermediate switch. Circuit switching offers low latency since there is no packet contention or routing decisions to be made during the transmission. However, circuit switching may result in inefficient resource usage when dealing with sporadic or bursty traffic patterns.

3.5.4 Virtual Channels

Virtual Channels (VCs) are used with various switching methodologies to improve the performance and flexibility of NoC designs. Virtual channels partition the physical links between switches into multiple logical channels. Each virtual channel operates independently, allowing for concurrent transmission of packets or flits and reducing the effects of contention and blocking. As a result, virtual channels enable better usage of the network resources and enhance the overall throughput and efficiency of the NoC. By understanding and selecting the appropriate switching methodology, designers can optimize the performance, latency, and resource usage of NoC systems according to specific application requirements and traffic patterns. The choice of switching methodology depends on factors such as desired latency,

bandwidth, reliability, and the trade-offs between complexity and performance. As a result, NoC architectures often combine multiple switching methodologies and use virtual channels to balance performance, scalability, and flexibility.

3.6 Routing Algorithms

Routing algorithms play a crucial role in NoC architectures by determining the paths data packets take from source processing elements (PEs) to their destinations. In addition, these algorithms define the route selection process and impact factors such as latency, throughput, energy consumption, and fault tolerance. This section provides an overview of three commonly used routing algorithms in NoC designs: Deterministic, adaptive, and stochastic routing, along with examples of each algorithm.

3.6.1 Deterministic Routing

Deterministic routing algorithms follow predetermined paths or routes based on the packet's source and destination addresses. These algorithms rely on pre-computed routing tables or fixed mappings to determine the next hop or output port for packet forwarding. One example of a deterministic routing algorithm is XY routing. The NoC is organized as a 2D grid in XY routing, where each processing element (PE) is assigned an XY coordinate. The routing decision is made by comparing the X and Y coordinates of the source and destination PEs. The packet is then forwarded in either the X or Y direction, depending on the difference in coordinates. Another example of a deterministic routing algorithm is source-based routing. This algorithm determines the routing path at the source network interface (NI). When messages flow through the routers, each router examines the routing information in the packets' header flits to determine the next hops. This deterministic algorithm guarantees deadlock-free routing and simple implementation, but it may suffer from congestion if multiple packets contend for the same paths.

3.6.2 Adaptive Routing

Adaptive routing algorithms adjust the routing paths based on real-time network conditions, such as traffic load, congestion, or link availability. These algorithms make routing decisions at run-time, considering the network's current state and alternative paths' availability. One example of an adaptive routing algorithm is the Distributed Dynamic Routing (DDR) algorithm. DDR algorithm operates by periodically exchanging network information among neighbouring switches. Each switch maintains a local network view, including traffic loads and link status. Based on this information, switches collaboratively make routing decisions by selecting paths with lower congestion or fewer faults. The DDR algorithm allows the network to adapt to changing conditions and balance the load across the network. However, it requires additional signalling and communication overhead [NLB00].

3.6.3 Stochastic Routing

The Stochastic Routing algorithm is a probabilistic approach used in NoC architectures to optimize communication by adapting routing decisions based on statistical information [DS10]. In an NoC with a mesh or torus topology, each router monitors the traffic on its input and output ports, keeping track of statistics such as packet arrival rates and congestion levels. When a packet arrives at a router, the Stochastic Routing algorithm probabilistically selects an output port for forwarding. The selection is based on statistical information, including traffic patterns, congestion levels, and packet arrival rates. The algorithm dynamically adjusts the probabilities of selecting each output port based on the current network conditions. For example, if a port is congested, its probability of selection may be reduced to alleviate congestion. Load balancing is achieved by distributing traffic evenly across the network, considering the current traffic distribution. This helps minimize overall congestion and latency. The algorithm also incorporates fault tolerance by avoiding faulty or congested links. It reroutes packets through alternative paths whenever possible to ensure reliable communication. While Stochastic Routing has advantages in optimizing communication in NoC architectures, it is essential to consider the increased complexity, non-deterministic behaviour, challenges in achieving deterministic QoS, potential performance variability and added overhead as potential disadvantages of this approach.

3.7 Deadlock and Livelock in Network-on-Chip Systems

In the NoC architecture, the issues of deadlock and livelock are of utmost importance as they can significantly impact the efficient functioning of the network and hinder the progress of processing elements (PEs) and routers. In the context of NoC architecture, deadlock arises when there is a circular dependence in the allocation of resources, leading to multiple PEs or routers being unable to make any progress. This situation typically occurs when a PE waits for a message to be forwarded by a router while the router waits for an available output port. Consequently, communication between PEs and the routing of messages come to a halt, resulting in a system deadlock [JB17]. In contrast, livelock refers to a scenario where PEs or routers continuously engage in unproductive actions that fail to advance the overall system. Unlike deadlock, where processes are blocked, livelock involves endless and futile activities that do not contribute to meaningful progress. This can lead to a decrease in system efficiency and performance [JB17]. Therefore, effectively addressing deadlock and livelock issues in NoC systems necessitates carefully considering and implementing suitable strategies. Various techniques can be employed, including efficient resource allocation algorithms, deadlock detection mechanisms, and intelligent routing protocols. These strategies aim to prevent circular dependencies, break deadlock situations, and mitigate livelock by incorporating intelligent decision-making and adaptive behaviours. Moreover, system designers must consider factors such as message priorities, network congestion, and load balancing to minimize the occurrence of deadlock and livelock. Proper synchronization and communication protocols, combined with effective deadlock detection and recovery mechanisms, can help ensure the smooth operation of NoC systems and maximize

their performance. By comprehending the challenges posed by deadlock and livelock in NoC systems, researchers and designers can develop robust and reliable architectures that optimize resource usage, enhance system scalability, and maintain efficient communication among PEs and routers.

Chapter 4

Related Work and Research Gaps

This chapter highlights the different techniques used in Network-on-Chip (NoC) architectures to enhance the safety and energy efficiency of NoC-based multicore architectures. It begins by outlining the requirements of the Adaptive Time-Triggered Networks-on-Chip (ATTNoC) architecture. Then, it presents an overview of existing NoC architectures, highlighting the advantages and limitations of each approach. Furthermore, the chapter explores how NoCs use fault tolerance techniques to improve reliability. This includes an in-depth analysis of redundancy-based approaches and adaptation techniques, providing insight into their advantages and limitations. Additionally, the chapter discusses the latest technologies implemented in NoCs to optimize power consumption. This involves examining the details of voltage and frequency scaling techniques and clock and power gating. The chapter emphasizes how various NoC architectures can effectively use these techniques to reduce power consumption while maintaining performance. In addition, the chapter highlights the current research gap and identifies areas where researchers can improve further. This chapter offers valuable information to researchers and engineers working on NoC architectures. It provides a comprehensive understanding of the various techniques used to enhance the safety and power efficiency of NoCs, enabling readers to make informed decisions when designing and implementing NoC architectures.

4.1 Requirements

The ATTNoC-based multi-core architecture is designed to meet the high-performance requirements of embedded systems by providing capabilities such as real-time operation, adaptability, fault tolerance, energy efficiency, and an open-source NoC-based design. The architecture must fulfil critical requirements for the system's success to achieve these capabilities. Firstly, the architecture addresses the fundamental requirement of real-time capability (RE1) for embedded systems. This involves the need for the system to respond promptly and predictably to events. To ensure real-time capability, the ATTNoC architecture uses time-triggered communication in the network, minimizing delays and jitter to achieve predictable response times. Adaptability (RE2) is identified as crucial for systems operating in dynamic environments where workload and system requirements frequently change. The ATTNoC architecture achieves adaptability through multiple schedules at the network interface, allowing the platform to reconfigure its schedule in response to contextual events, thus enabling the system to meet evolving requirements. Fault tolerance (RE3) is

essential for safety-critical systems in harsh environments where transient or permanent faults can occur. The ATTNoC architecture addresses fault tolerance at the network interface level through seamless redundancy mechanisms. These mechanisms duplicate critical data at the network interface and transfer it using a dual channel, enhancing the system's fault tolerance in the event of faults in NoC routers or links. Energy efficiency (RE4) is identified as crucial for systems operating with limited energy sources. The ATTNoC architecture incorporates frequency scaling at the router level to achieve energy efficiency. This involves integrating frequency scaling into the schedule, where ATTNoC schedules the injection time of messages and predefined specific frequencies to be used in different time slots within the period. This approach enables the routers' frequencies to be adjusted according to the predefined schedule, resulting in optimized power consumption without compromising system performance. An open-source NoC-based architecture (RE5) is considered essential for the ATTNoC architecture to gain widespread adoption and support. The architecture is based on an open-source NoC architecture, allowing developers to customize and modify the system according to their requirements, making it flexible and adaptable. Lastly, the ATTNoC architecture ensures deadlock-free operation at the router level (RE6). Deadlocks can occur when multiple routers wait for each other to release resources, leading to system failure. The architecture uses source-based routing and time-triggered communication to predefine message routes and schedule message injection times to avoid cyclic paths and resource contention within NoC routers. By incorporating these requirements and techniques, the ATTNoC architecture aims to deliver reliable, efficient, high-performance systems operations, meeting the demands of modern computing applications.

4.2 Network-on-Chip

Using multi-core processors in embedded systems has created new high-performance applications, such as real-time data processing and complex control algorithms. Additionally, these processors have generated interest in Network-on-Chip (NoC) networks for inter-core data communication. NoCs offer scalability, parallelism, system modularity, high-frequency operation, and power efficiency [TUM], [Nam+22], [Bei+05], [Fer+04], [Che+13], [Bar+06], [Adr+03], [Sch07]. However, shared resources like caches, buses, and inputs/outputs in multi-core architectures introduce non-determinism in execution time analysis. To meet Quality of Service (QoS) requirements, researchers have suggested employing resource reservation and priority-based mechanisms to ensure guaranteed levels of latency and bandwidth. Various NoC architectures have been proposed in the literature, each with strengths and weaknesses. This section reviews existing NoC architectures and compares them to our adaptive time-triggered NoC (ATTNoC) architecture. The LISNoC architecture [TUM] is an open-source NoC architecture that uses wormhole packet switching for flit transmission within the NoC. It employs three flow control signals (flit, valid, and ready) to ensure reliable packet delivery. The architecture supports virtual channels that enable concurrent communication and improve the performance of NoC routers. However, the LISNoC only supports event-triggered communication and does not support source-based routing and time-triggered communication. To address these limitations, we extended the LISNoC to support source-based routing and time-triggered communication [Nam+22]. This extension allows us to schedule

the communication of the NoC and route the data based on the route defined by the schedule, which is stored in the network interface of the NoC. Furthermore, the time-triggered communication at the network interface allows the NI to inject messages based on a predefined time retrieved from the schedule. By injecting messages at predefined times, communication between processing elements (PEs) can be more deterministic. This is especially important in real-time systems or applications where precise timing is crucial. Moreover, the adaptation feature incorporated in the NoC enables the NoC to re-configure its communication schedule, such as rerouting a message due to a permanent fault that blocks communication within the NoC or re-configuring the frequency used in the NoC based on the current workload. We also added a redundancy mechanism to some critical NIs to transfer messages using dual channels to increase the reliability of the systems. The asynchronous NoC (ANoC) architecture proposed in [Bei+05] is an asynchronous NoC architecture that incorporates both Quality of Service (QoS) and Transaction-Level Modeling (TLM). It enhances efficiency and reduces latency for prioritised packets using wormhole packet switching and virtual channels. The architecture is designed to work with a 2D mesh network, where nodes are arranged in a grid-like structure. Each node is connected to its neighbouring nodes, allowing for efficient communication. Each node connects to a network interface featuring a Globally Asynchronous Locally Synchronous (GALS) interface. This interface facilitates synchronisation between the synchronous and asynchronous domains. In contrast, the ATTNoC architecture, as the name suggests, employs time-triggered communication. It incorporates a time-triggered mechanism to control and schedule communication within the NoC. This approach ensures deterministic behaviour and enables the provision of real-time guarantees. Moreover, the time-triggered architectures use a global clock synchronising NoC communication. This differs from the asynchronous nature of the ANoC architecture, where communication occurs without a global time-based clock. The Hermes Network on Chip is introduced by Moraes et al. [Fer+04]. Moraes et al. reviewed the state of the art for NoC in their work. An infrastructure called Hermes is described, which implements packet-switching techniques, mesh topology, and related interconnection architectures. The primary element of Hermes is a switch with five bidirectional ports connected to four other switches and a local IP core. The switch applies an XY routing algorithm and uses input queueing. However, our design supports a dynamic routing algorithm, which allows the NoC to re-configure its route when a context event occurs. It also supports virtual channels that can minimise the NoC's latency. The GALS NoC architecture [Che+13] solves the clock distribution problem by linking asynchronous routers to synchronous blocks. It uses wormhole packet switching with an XY routing algorithm and supports mesh topologies. In contrast, our design uses the global time to share the timing view between distributed components that may run with different clocks. Moreover, an adaptation routing algorithm is supported, allowing the NoC to re-configure the path in case of a fault. AEthereal and Nostrum [Mik+04], [Bar+06] use a resource reservation mechanism to provide guaranteed services for both throughput and latency while accommodating best-effort communication. These architectures employ a mesh topology and implement a multi-layered advanced high-performance bus (ML-AHB) interconnection architecture in the case of AEthereal. The ML-AHB architecture allows parallel access between multiple masters and slaves, enabling efficient communication within the system. Our design introduces an interconnect

structure using the NI (Network Interface) based on the AXI (Advanced eXtensible Interface) protocol. This design choice provides a scalable solution for interconnecting various components within the system-on-chip (SoC). This scalable interconnecting structure offers flexibility and facilitates efficient communication between different subsystems within the SoC. The SPIN (Scalable Programmable Interconnection Network) NoC architecture [Adr+03] introduces two on-chip communication templates based on bus and NoC approaches. In SPIN, packets are forwarded when a router receives their headers, a wormhole switching technique. SPIN has adaptive and distributed routing and supports the fat-tree topology. The RSPIN router serves as a fundamental component in a SPIN network, featuring eight ports, with each port having multiple input and output channels to facilitate efficient communication within the network. On the contrary, the ATTNoC supports the mesh topology, which offers better scalability. In a mesh topology, each router within the network is directly connected to every other router in the system. This means a direct communication path between routers, allowing for efficient and direct data exchange. In a fat tree, some routers are connected to more routers than others. This can create bottlenecks and limit the scalability of the system. The NoC in the Xilinx Versal architecture is described in [Swa+19]. This 7nm SoC from Xilinx contains a hardened NoC, which unifies communication between hardened accelerator functions, FPGA fabric, memory subsystem, and processor system. The Versal NoC uses an irregular topology built from reusable building components that can link in various ways for various devices. The Versal NoC distinguishes itself from competitors by allowing users to select the QoS at the design stage. The Versal NoC communicates with devices via the AXI standard and supports AXI streams. The router architecture is simplified, offering design flexibility and improved bandwidth usage. However, different features are still missing in the Versal NoC, such as adaptability, seamless redundancy mechanisms, and time-triggered features, which are the main features of this thesis. Moreover, VERSAL NoC is only supported in the Versal ACAP platform and cannot be used on other platforms.

The Time-Triggered NoC (TTNoC) [Sch07] introduces an on-chip time-triggered interconnection to facilitate predictable communication in real-time systems. Using simple routers and a pseudo-static communication schedule allows for efficient data transfer and fault isolation, which is particularly useful when dealing with diverse critical components. In our design, we extend the open-source LISNoC to incorporate time-triggered capability, enabling the Network Interface (NI) to inject messages at predetermined intervals. Moreover, the NI supports multiple schedules to facilitate adaptive communication and allows the NoC to re-configure its schedule in response to events. The Adaptive Time-Triggered Multi-Core Architecture (ATMA) [Obe+19], proposed by Obermaisser et al., extends the TTNoC architecture [Sch07] by incorporating adaptability features. These features enable the NoC to reconfigure its schedule based on contextual events, such as faults in NoC resources, thereby enhancing the fault tolerance capability of the architecture. However, the ATMA architecture lacks mechanisms to handle rapidly occurring transient faults and does not support frequency scaling at the router level, limiting its energy efficiency. We propose the Time-Triggered Frequency Scaling (TTFS) technique to address these limitations. This technique enhances energy efficiency at the router level by enabling the router to adjust its frequency according to a predetermined schedule. It aims to enhance energy efficiency while preserving the deterministic behaviour of the NoC

communication. Additionally, we extend the TTNI architecture of ATMA to support a seamless redundancy mechanism. These redundancy features incorporated in the TTNI allow for the exchange of critical data through a dual channel while using normal operation for low-criticality data without redundancy. This design avoids the need for fully duplicated NoC components, resulting in low overhead. These crucial features improve performance efficiency and reduce overhead in multi-core architectures [TUM], [Nam+21], [NOO23], [Obe+19], [Nam+23].

4.3 Fault Tolerance Techniques for NoC

Fault tolerance in NoC architectures has made significant progress in detecting, correcting, and masking faults. One commonly used approach is spatial redundancy, where protected NoC components are replicated to achieve adequate fault tolerance. Spatial redundancy can be classified into static, dynamic, and hybrid redundancy [Dub08]. Static redundancy passively masks faults, while dynamic redundancy detects faults and takes corrective action. Hybrid redundancy combines both static and dynamic methods [Rad+13]. For example, triple modular redundancy (TMR) is a well-known static redundancy technique that can mask a single fault but requires significant additional hardware [Dub08]. Redundancy can be applied at different layers of an NoC, e.g., 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 [Mur+06], [LLP10], and logic elements like the crossbar multiplexer, ensuring correct switching even in the presence of faults [Egh+10]. To enhance NoC connectivity and ensure uninterrupted communication even in the presence of faulty links, Kakoee et al. [KBB11] 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 adequate 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 [Con+06a]. The second trade-off is combining spatial redundancy with other techniques to reduce overall costs. The system can improve fault tolerance by integrating spatial redundancy with complementary fault tolerance techniques while considering 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 [KK19]. 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 by bit flips which oc-

cur due to low noise margins, electromagnetic coupling effects, or crosstalk [AA05], [Kim+14], [Ouy+21], [Wan+15]. Temporal redundancy, such as retransmission at the link-to-link or end-to-end level at different time intervals, is one approach for tolerating transient faults by retransmitting data during a transient fault [Ouy+21]. This method can help improve fault tolerance by introducing redundancy in the 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 acknowledgements 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 suggestions aim to avoid the costly reset [Wan+15], [DeO+12], 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 [Con+06b]. One such proposal, Bulletproof [Mot+13], 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 method that satisfies various trade-offs is needed. 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 [Obe+19]. 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 essential. By enhancing the reliability of message delivery, the system optimizes performance, leading to better overall system efficiency. By in-

corporating 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.

4.4 Low Power Techniques for NoC

In recent years, a significant amount of research has been conducted on low-power architectures for multi-core systems using Network-on-Chip (NoC). These techniques are designed to reduce power consumption by adjusting the supply voltage, frequency scaling, clock gating, and adaptive routing algorithms [Man+21]. It is essential to manage the power of NoCs efficiently in order to achieve energy efficiency in multi-core platforms, and various state-of-the-art approaches provide a wide range of low-power techniques. One of the most important aspects of reducing power consumption is manipulating the supply voltage, which has a direct effect on dynamic and static power [Man+21]. Various techniques have been developed to control the supply voltage. One of these techniques involves using different supply voltages (Multi-Voltage) for different components in combination with level shifters. However, it is essential to note that these techniques may introduce delays and additional power consumption, potentially impacting the system's performance. Another effective technique is frequency and voltage scaling, which involves the power manager controlling different power modes based on the required temporal behaviour. This approach allows for adjusting the supply voltage and clock frequency according to the specific demands of the system. Additionally, instead of scaling down the supply voltage when certain system parts are idle, components can be completely switched off for an extended period, saving power. Other low-power techniques encompass multiple power modes (e.g., idle, sleep), effective cache usage, selectively clock-gated caches, small architectures with reduced static power, Dynamic Power Management (DPM), and application-driven and operating-system-driven solutions. Integrating these techniques into the scheduling and design of software tasks is vital for maximizing power savings. However, when applying these low-power techniques to mixed-critical systems, certain limitations emerge due to their impact on timing [Mot+13]. For instance, voltage/frequency scaling can introduce varying execution times, while switching off components can result in different response times as they need to be powered on first. These complexities in timing behaviour and the potential impact on critical components pose challenges in fully using these techniques in safety-critical systems [VHL14]. Consequently, safety-critical systems make less extensive use of these power-saving methods.

Several studies have explored the application of low-power techniques in various computing architectures or platforms. For example, in a study cited as [Ila+20], researchers proposed a data-driven frequency scaling technique that resulted in a 15% reduction in the energy consumption of GPUs compared to baseline policies. Another study [Che+20] investigated the impact of dynamic voltage and frequency scaling (DVFS) techniques on power consumption in low-power microcontrollers within wireless sensor networks, highlighting the significance of DVFS on power consumption and processor energy. Researchers in [PCG19] proposed a method for scaling the voltage and frequency of multi-core CPUs, achieving average energy

savings of 25% and 22% for different microprocessors by dynamically adjusting voltage and frequency settings based on optimal policies. It is important to note that while these studies are related to power management, they differ from the present research, which explicitly focuses on frequency scaling in NoC routers. Additionally, in [Vai+19], the authors emphasised the importance of considering the granularity of DVFS (dynamic voltage and frequency scaling) in design to improve energy efficiency. Disregarding DVFS granularity could potentially increase power consumption by 19%, compromising the system's overall energy efficiency. However, this work primarily focused on the core level rather than the NoC routers, differentiating it from the research presented here. Furthermore, a study proposed a Learning-enabled Energy-Aware Dynamic Voltage/Frequency Scaling method in [M+18] to improve power consumption in NoCs. This technique employed machine learning to compensate for the performance-energy trade-off and enabled proactive energy management through an offline-trained regression model. The simulation results demonstrated an average dynamic energy saving of 17% with minimal throughput loss and no increase in latency. In contrast, this thesis focuses on enhancing the energy efficiency of NoCs by scaling their frequency using a time-triggered concept. The goal is to preserve the deterministic communication behaviour while achieving improved energy efficiency in the systems. In [Nam+21], a technique called Time-Triggered Frequency Scaling (TTFS) is introduced to enhance the energy efficiency of NoC-based multicore architectures. TTFS enables frequency scaling in NoC routers according to a predefined schedule while preserving NoC performance. However, the architecture introduced in [Nam+21] uses a centralized controller to scale the frequency of NoC routers, which can be susceptible to failure. The current research extends the TTFS technique by introducing a distributed controller to address this limitation. This distributed controller is responsible for scaling the frequency of each router based on a schedule, thereby improving the system's reliability. Existing state-of-the-art research reveals that implementing low-power techniques in processors, NoCs, or microcontrollers can significantly enhance the power consumption of these systems. However, most of these low-power techniques primarily focus on non-real-time applications. In contrast, this thesis proposes a distributed Time-Triggered Frequency Scaling (TTFS) technique as a power-saving approach to improve the energy efficiency of NoCs while maintaining their deterministic behaviour [Nam+21], [NOO23]. The TTFS technique employs a distributed controller that scales the frequency of NoC routers according to a predefined schedule using the time-triggered concept, aiming to enhance the energy efficiency of the NoC. Three different approaches are used to evaluate power consumption within the TTFS framework. The first approach, known as the global approach, synchronizes the frequency of routers across all routers. The schedule determines the operating frequency of the active routers while clock-gating the routers during the idle time of all routers according to a schedule. The second approach, called the cluster-based approach, divides the NoC into multiple clusters, each operating at a frequency determined by the schedule. Lastly, the router-based approach assigns individual schedules to each router, allowing them independent operating frequencies, regardless of the overall NoC or cluster frequencies. By incorporating techniques such as global, cluster, and router-based approaches into the TTFS framework, the thesis aims to address power consumption challenges in NoCs and provide effective power-saving solutions while considering the system's real-time requirements. In summary, while existing research has made sig-

nificant progress in low-power techniques for processors, NoCs, and microcontrollers, most of these approaches have primarily focused on non-real-time applications and the core level. On the contrary, the present research proposes the TTFS technique as a distributed power-saving approach for NoCs, with the aim of improving energy efficiency while meeting real-time demands.

4.5 Research Gaps in the State of the Art

The research on NoC-based multi-core architectures has identified three critical gaps that need to be addressed. First, addressing permanent faults in NoC platforms while considering the overheads is crucial. This is because permanent faults can lead to system failures if mishandled. However, addressing permanent faults can also introduce overheads, such as replicating data or using more complex routing algorithms. Therefore, it is essential to balance fault tolerance and overheads. Second, optimizing energy efficiency for real-time applications requires additional support for frequency scaling at the router level. This involves the development of mechanisms that enable routers to adjust their operating frequencies while meeting real-time requirements. Lastly, ensuring the reliability of the entire NoC platform requires comprehensive solutions. The Adaptive Time-Triggered Multi-Core Architecture (ATMA), introduced in [Obe+19], offers adaptability and fault tolerance but needs to overcome certain limitations. Specifically, it lacks mechanisms to effectively handle rapidly occurring transient faults, which is crucial for real-time systems with stringent safety requirements. Addressing this deficiency requires comprehensive fault tolerance techniques considering trade-offs between area, power consumption, latency, and reliability. Seamless redundancy mechanisms can enhance system reliability without incurring excessive overhead. The seamless redundancy mechanisms enable the NI to transfer critical messages through a dual channel while transferring a non-critical message to a single channel. This improves reliability without the need for full duplication of NoC resources. Adaptation techniques play a vital role in fault recovery and energy efficiency by enabling the NoC to reconfigure its schedule, facilitating message rerouting, or adjusting injection times or the operating frequency of NoC based on workload. This flexibility optimizes reliability and energy efficiency within the NoC. Existing low-power techniques for NoC-based multi-core systems primarily focus on non-real-time applications, overlooking the need for comprehensive power-saving approaches that address real-time requirements. Power consumption optimization mainly targets the processor or core level, with limited attention given to the NoC router level. This highlights the lack of a holistic approach that integrates power-saving techniques at both the core and NoC levels while effectively addressing real-time requirements. To address the identified gaps, a comprehensive solution is required to reduce power consumption while meeting real-time demands. Time-Triggered Frequency Scaling (TTFS) emerges as a promising technique that scales the frequency of NoCs based on a predefined schedule. By scheduling the frequency used in the routers in advance, the operating frequency can be clock-gated or scaled up or down according to the schedule, enhancing energy efficiency while preserving deterministic communication behaviour. Time-triggered mechanisms ensure deterministic communication by injecting each message into the NoC according to a predefined schedule. This aims to avoid message collisions in the NoC and preserve the deterministic behaviour of the NoC communication. A comprehensive

solution can be achieved by integrating TTFS, seamless redundancy mechanisms, adaptation techniques, and time-triggered mechanisms. This integrated approach effectively addresses the critical research gaps, leading to improved fault tolerance and energy efficiency in NoC-based multi-core architectures [Obe+19], [Nam+21], [NOO23], [TUM].

Chapter 5

System Model

This chapter presents the system model of an adaptive time-triggered multi-core architecture based on Networks-on-Chip (NoC). It provides a comprehensive overview of the architecture's components and functionality and the building blocks, interfaces, and services employed in the system.

5.1 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 and power-saving techniques to provide safety and energy efficiency. The ATTNoC integrates several services to fulfill the requirements outlined in section 4.1. These services include time-triggered communications, adaptability, fault tolerance, and energy efficiency while upholding system safety. These services are described in the following items.

- Time-triggered communication: This service facilitates the exchange of messages within the NoC based on a predefined 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 network interface (NI) in ATTNoC 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[Obe+19].

- Seamless redundancy mechanisms: The ATTNoC architecture introduces two types of time-triggered network interfaces (TTNIs) 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.

- Time-triggered frequency scaling (TTFS) is a technique in which the frequency of each router is predetermined and scheduled in advance. This approach enables NoC routers to adjust their operating frequency based on a predefined schedule during their active time. By clock-gating the idle routers and adjusting the frequency of active routers based on the predefined schedule, energy consumption is optimized while maintaining deterministic behaviour in the system. This ensures that messages are delivered within specified time constraints while enhancing energy efficiency within the NoC.

The ATTNoC architecture builds upon the open-source event-triggered LISNoC presented in Section 5.1.1. It supports topologies like meshes and connects resources like hard and soft processors, memory subsystems, etc. Figure 5.1 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 5.1) 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 5.1. These services encompass global time-based functionality, serving as a time reference within the ATTNoC. Time-triggered frequency scaling is used as a power-saving technique at the router level. 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.

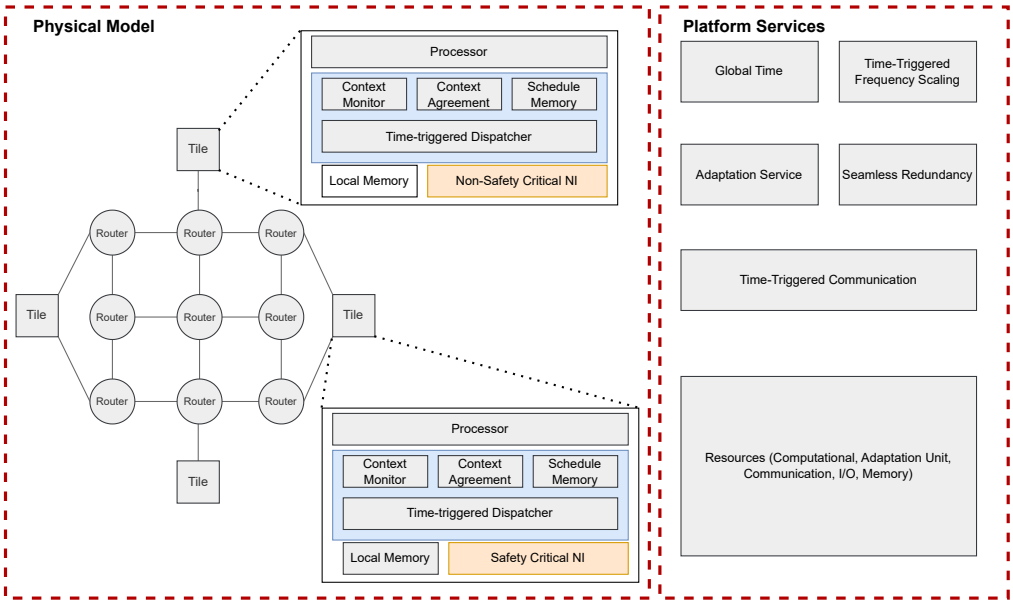


Figure 5.1: Physical and logical system model of ATTNoC

5.1.1 LIS Network-on-Chip (LISNoC)

LISNoC is an event-triggered-based Network-on-Chip (NoC). It is an open-source NoC implemented in Verilog, primarily used for academic purposes [TUM]. It serves as the foundation for implementing ATTNoC, which incorporates fault tolerance and power-saving techniques. Some of the main features of LISNoC include virtual channel support, flexible configuration, wormhole routing, and round-robin arbitration [TUM]. 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, time-triggered communication, adaptability, and redundancy mechanisms to support predictable communication and real-time applications. Furthermore, the LISNoC has been extended to support power-saving techniques. These techniques involve clock-gating the idle routers and scaling the frequency of active routers based on a predefined schedule to optimize energy consumption and preserve deterministic communication within the NoC routers. Additionally, fault tolerance techniques have been integrated into the router. In the event of a permanent fault, such as an open link or a stuck-at '0' or '1', the router is designed to bypass local input/output ports and the router crossbar by switching between schedules. This bypass mode allows the data to be sent through alternative paths, enabling the system to tolerate faults and maintain reliable communication within the NoC.

5.1.2 Time-Triggered Control in ATTNoC

ATTNoC is an enhanced version of LISNoC that incorporates time-triggered communication, enabling message exchanges based on a predefined schedule. It consists of two distinct time-triggered controllers serving different functions. The first controller in the TTNI is responsible for scheduling communication, computation, and frequency scaling within ATTNoC routers. Its primary role is to ensure that all tasks and messages are transmitted within specific time constraints, promoting reliable and predictable communication. The second controller is situated in the adaptation unit and is responsible for scheduling the components of the adaptation unit, including the context monitor (CM), context agreement (CA), and schedule switching. This controller also ensures that the distributed TTNI performs schedule switching simultaneously with any context event, e.g., slack and permanent faults in NoC resources like NIs, cores, routers, and links.

Time-Triggered Controller in Time-triggered Network Interface

The time-triggered controller in the NI is responsible for scheduling communication, computation, and the frequency to be used in the routers. The TTNIs rely on global time-based synchronization to maintain a consistent timing view in the ATTNoC. Since the frequency for each router is predefined in the schedule, the time-triggered dispatcher, which triggers message injection and frequency scaling, can adjust the operating frequency of active routers and clock-gate idle routers according to the schedule to save energy. The dispatcher also ensures that the router frequency is clock-gated when no messages pass through, preserving NoC performance and meeting message deadlines.

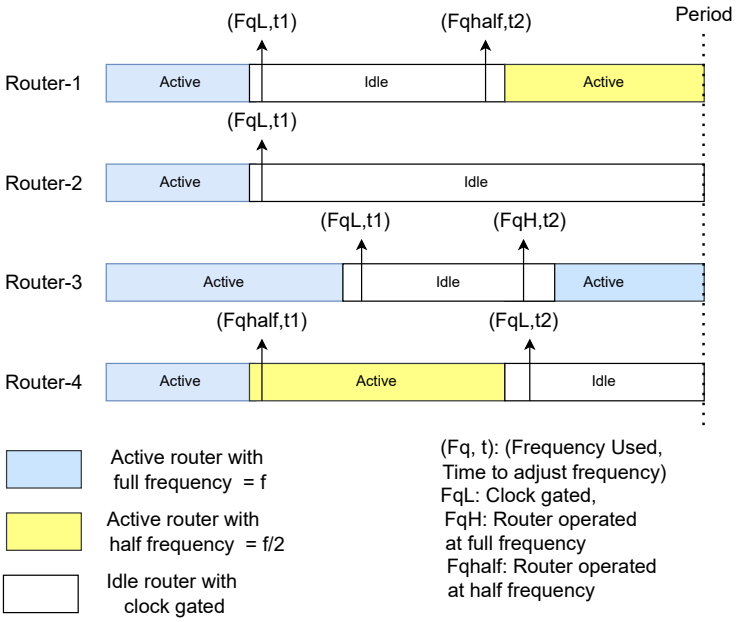


Figure 5.2: Example of frequency scaling with four routers in ATTNoC

As depicted in Figure 5.2, routers in the system are designed to accommodate multiple clock domains, including full frequency, half frequency, and clock gating. The time-triggered dispatcher triggers the frequency scaling on the routers and defines the operating frequency of each router according to the schedule. During idle periods, the router's frequency is set to clock gating mode, which helps conserve power. However, during active periods, the router's frequency is adjusted according to the schedule.

Figure 5.3 illustrates the structure of the schedule entries within the ATTNoC architecture. These schedule entries are organized as a linked list and include the InstantMsg, InstantFreq, PortID, RouterID, FreqMode, and Next values.

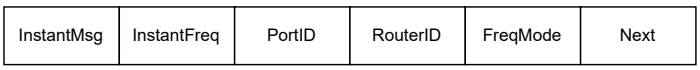


Figure 5.3: Schedule entries of a time-triggered dispatcher in TTNI.

- InstantMsg represents the specific time at which a message is injected.
- InstantFreq indicates the time the router's frequency is adjusted.
- PortID indicates the ID of the port from which the message is injected.
- RouterID refers to the unique identification number assigned to a router. For example, when the NI is connected to two routers, the RouterID specifies which of the two router frequencies connected to the NI should be adjusted.
- FreqMode defines the operating frequency used by the router.
- Next is a pointer indicating the schedule's next entry.

Time-Triggered Controller in Adaptation Unit

The time-triggered controller within the adaptation unit is crucial for scheduling various resources, such as the context monitor, context agreement, and schedule switching, as described in section 5.1.6. This scheduling process relies on a predetermined schedule to ensure simultaneous switching between schedules across distributed network interfaces (NIs) when a context event occurs. Synchronization is essential when the ATTNoC switches between schedules to prevent one NI from operating with a new schedule (S1) while another continues operating with the initial schedule (S0). Failure to properly synchronize the schedules in the distributed NIs can result in delays within the NoC due to message collisions between the original and new schedules configured in different NIs. Figure 5.4 presents the structure of the schedule entries for controlling the adaptation in the ATTNoC. The schedule for adaptation entries consists of Instant, AdaptationLogicID, and Next values. These entries are essential for effectively managing and coordinating the adaptation processes throughout the NoC.

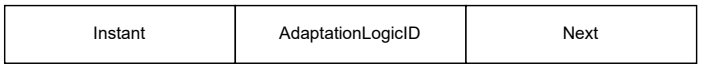


Figure 5.4: Schedule entries of a time-triggered dispatcher in the adaptation unit.

- Instant indicates when the context monitor, context agreement, or switching schedule is triggered in the ATTNoC.
- AdaptationLogicID: The ID of the schedule entries determines which subcomponents of the adaptation unit, such as the context monitor, context agreement, and schedule switching, should be triggered.
- The next is a pointer that indicates the subsequent entry of the schedule.

5.1.3 Adaptation in ATTNoC

In the ATTNoC architecture, the network interface is designed to support multiple schedules computed offline and stored in a specific memory. This allows schedule switching in response to context events occurring in the NoC, such as permanent faults. Figure 5.5 illustrates an example of multiple schedules, with each schedule connected through a global context event. This global context event enables the ATTNoC to transition from one schedule to another. Incorporating adaptation features into ATTNoC primarily aims to enhance energy efficiency and facilitate fault recovery [Obe+19]. The subsequent sections will explore how adaptation is employed to achieve these goals within ATTNoC.

Adaptation for Energy Efficiency

Dynamic slack in the system provides an opportunity to preserve system performance while reducing energy consumption in the ATTNoC. This dynamic slack can be effectively used through two strategies: local adaptation and global adaptation [Obe+19]. Local adaptation is employed when a core experiences slack, allowing a task to be completed ahead of its scheduled time. By locally reducing the processor

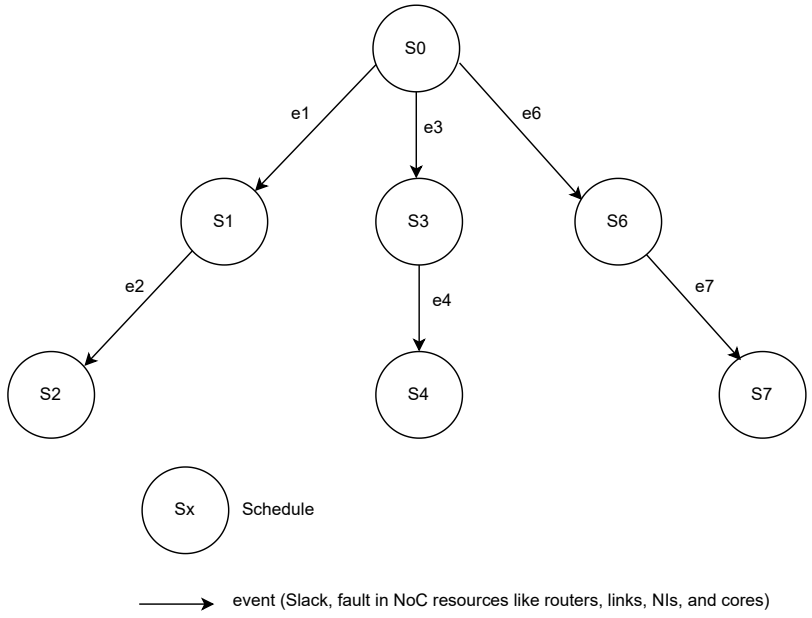


Figure 5.5: Example of multiple schedules linked to each other by event

frequency during the slack period, power consumption in the core can be effectively reduced [Obe+19]. One key advantage of local adaptation is that it optimizes the schedule of a specific core without impacting the rest of the system. The interface between the core's local schedule and the NoC's communication schedule remains unchanged, ensuring that message injection remains unaffected. Local adaptation is particularly beneficial when the slack time is less than twice the duration required for ATTNoC to transition from one schedule to another. It allows efficient power savings within individual cores while maintaining overall performance. However, global adaptation focuses on changes within a subsystem, such as a core, resulting in temporal or spatial modifications in the usage of shared resources. Figure 5.6 illustrates an example of global adaptation, where dynamic slack occurs within the sender core that needs to transmit data to another core. Adjustments to the ATTNoC schedule are made to accommodate this slack by modifying the injection time of messages. As a result, the transmitter core can transmit the data earlier during the slack period. This adjustment enables the receiving cores to begin their computations earlier, giving them more time until their deadlines. Furthermore, the surplus slack time budget can be used to reduce the clock frequency of the receiver cores, resulting in more significant energy savings by lowering the frequency of multiple receivers, as opposed to focusing solely on a single sender core, as observed in local adaptation [Obe+19].

Adaptation for Fault Recovery

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 5.7 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

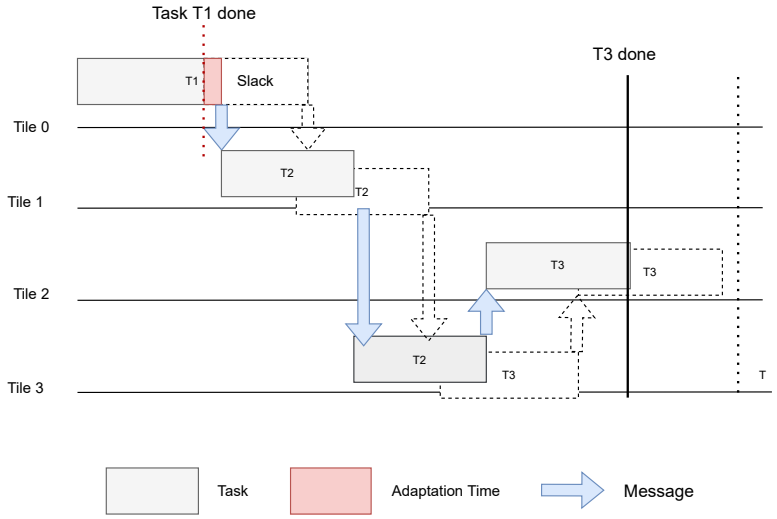


Figure 5.6: Global adaptation by changing the injection time of ATTNoC when slack occurs

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 5.7. 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.

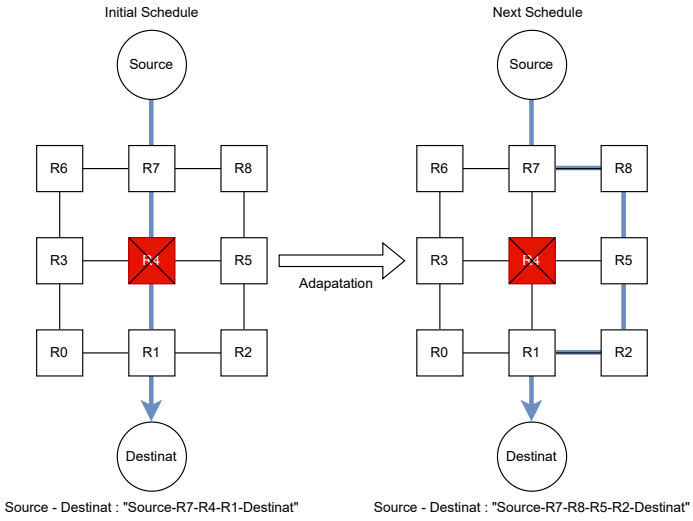


Figure 5.7: Global adaptation by isolating a faulty router in ATTNoC

5.1.4 Fault Model

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.

Failure Mode

We simulate permanent faults that may occur 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.

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 5.8. 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 its functionality. Table 5.1 below shows how each permanent

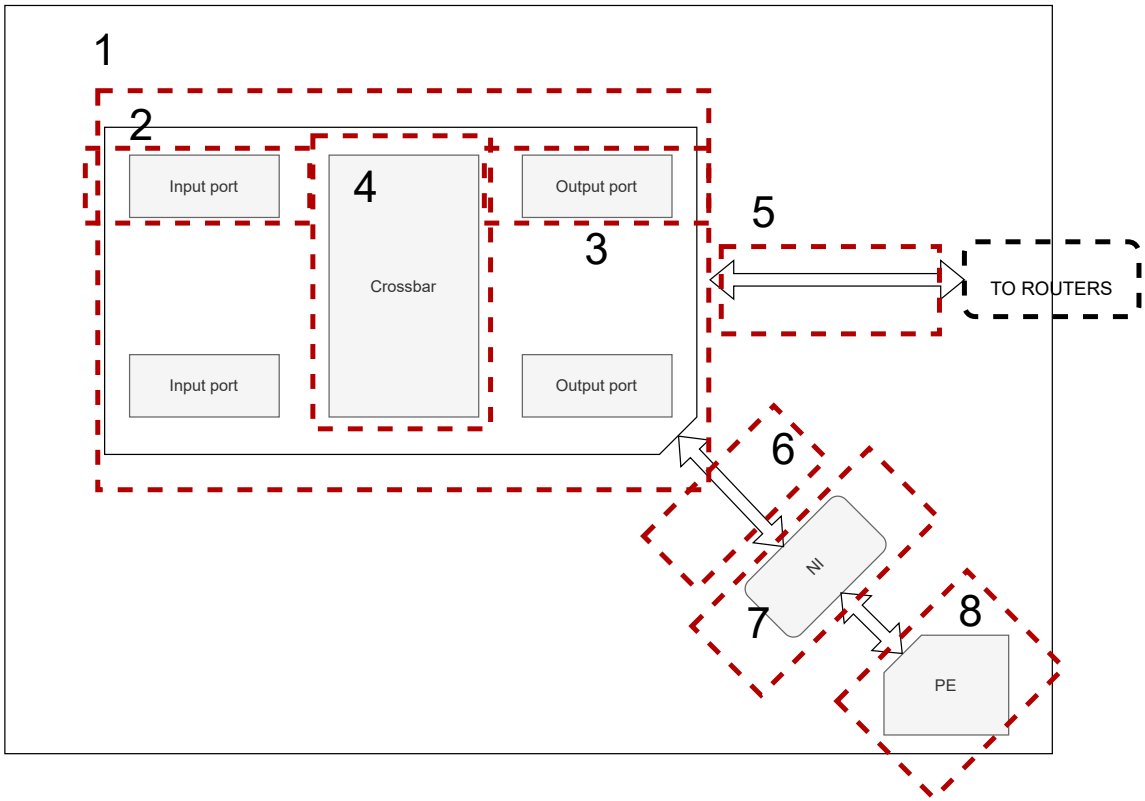


Figure 5.8: FCR in ATTNoC architecture

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

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

Table 5.1: Permanent fault in ATTNoC

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 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 and permanent faults occurring 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, as described in Section 5.1.7.

5.1.5 Power Model

In digital circuits, power consumption is typically observed in two phases: The active phase and the standby phase. During the active phase, the circuit inputs are active, producing the output, leading to dynamic and static power consumption. Dynamic power is dissipated when a transistor switches from high to low, resulting in an output. Short-circuit power consumption and power loss due to leakage also contribute to power dissipation during the active phase. However, due to the complexity of modern multiprocessor system-on-a-chip (MPSoCs), accurately modelling power consumption can be challenging. A viable approach for modelling the power consumption of Network-on-Chip (NoC) is to model the building blocks' compo-

nents, primarily composed of logic gates [Nam+21]. The total power in the digital complementary metal-oxide-semiconductor (CMOS) circuit can be expressed as:

$$P_{total} = P_{Dyn} + P_{Stat} \tag{5.1}$$

where P_{Dyn} represents dynamic power, and P_{Stat} represents static power.

The dynamic power can be further broken down into switching power and short-circuit power:

$$P_{Dyn} = P_{SW} + P_{SC} \tag{5.2}$$

where P_{SW} is the switching power, and P_{SC} is the short-circuit power.

The switching power (P_{SW}) is given by:

$$P_{SW} = \alpha \cdot F \cdot C_L \cdot (V_{dd})^2 \tag{5.3}$$

where α is the activity factor, F is the clock frequency, C_L is the load capacitance, and V_{dd} is the power voltage.

The short-circuit power (P_{SC}) is approximately 10% of the dynamic power:

$$P_{SC} = 10\% \cdot P_{Dyn} \tag{5.4}$$

The static power (P_{Stat}) is due to the leakage current in each logic block and is proportional to the supply voltage (V_{dd}). It can be approximated using:

$$P_{Stat} \approx \beta \cdot V_{dd} \cdot e^{-V_{th}/\gamma \cdot V_T} \tag{5.5}$$

Where β and γ are experimentally derived constants, V_{th} is the threshold voltage, and V_T is the Boltzmann thermal voltage that is linearly proportional to the temperature [Nam+21].

Overall, Equations 5.1 to 5.5 provide a framework for modelling power consumption in digital circuits, considering both dynamic and static power components. These equations can be useful for optimizing power consumption in digital circuits, particularly in complex MPSoCs and NoCs.

5.1.6 Tile

A tile is a building block of an ATTNoC that comprises a processing element, an adaptation unit (cf. blue in Figure 5.1), and a NI, as depicted in Figure 5.1. The processing element in the ATTNoC can be a hard processor or a soft processor, such as MicroBlaze, and it can also be an AI engine or network gateways that serve as an interface between the on-chip and off-chip domains. The processing elements are typically used for computations. The NI interface interconnects the tile to the router, as described in section 5.1.7. Finally, the adaptation unit manages and coordinates adaptation services within the NoC. It consists of a context monitor, a context agreement unit, a schedule memory, and a TT schedule, as described in the following items.

- The context monitor (CM) is responsible for reading the state of the local context from the adjacent core. The local context can be an application running on a core (e.g., slack) or a context event resulting from a fault within NoC resources. In ATTNoC, each tile has a distributed CM. A schedule is followed

to enable simultaneous monitoring of neighbouring cores by each CM. The time-triggered concept facilitates synchronization in the adaptation unit, ensuring that each CM in ATTNoC receives its local context event at a specific time defined by the schedule.

- Context agreement (CA): Through the Interactive Consistency Protocol (ICP) [Len20], the CAs establish a globally consistent context vector (global context) by collecting and agreeing on the local contexts reported by all CMs on the respective tile. The CAs initiate a synchronized context distribution phase based on a predefined schedule. Via the ICP, the local context is distributed in a triple-ring structure across the adaptation unit. The local context is sent to and collected from the nearest neighbour in the network and concatenated into a global context vector. All resources have consistent system information at the end of the distribution phase. At this point, the ICP converges, and the CAs agree on the system state. The agreed context vector is the globally consistent context vector representing the system state at all resources. The schedule memory uses the globally consistent context vector to determine the next schedule to be adapted based on the global context event.

- The Schedule memory stores multiple schedules generated offline, which are used in the ATTNoC for schedule switching when a context event occurs. Each schedule in the memory is mapped with a context event. The schedule memory is accompanied by a schedule memory controller that receives the global context from the CA block. A lookup table maps the context events to their corresponding addresses in the schedule memory, where the schedules are stored. The schedule memory controller fetches the corresponding schedule using the provided addresses to adapt and retrieve the desired schedule. Once retrieved, the schedule is written to a designated port on the core interface of the NI to facilitate schedule switching. The time-triggered dispatcher within the adaptation unit initiates the actual schedule switching. By scheduling the switching of schedules based on predefined schedules, collisions between schedules in the ATTNoC can be avoided. This ensures that each NI switches schedules simultaneously, maintaining a coherent operation of schedules within the ATTNoC.

- Time-triggered dispatcher: This component is responsible for managing the execution of the adaptation unit by following predetermined time-triggered schedules. Its primary role is to schedule and coordinate the CM and CA operations, triggering the schedule switching in the TTNI. The dispatcher ensures that all distributed TTNIs switch schedules simultaneously. This synchronization is crucial to ensure that all TTNIs operate with the same schedule. By doing so, the dispatcher prevents situations where certain TTNIs may still be using a previous schedule (S0) while others have already transitioned to a new schedule (S1). Maintaining consistency across all TTNIs is paramount, as inconsistencies could lead to system failures or malfunctions. Different TTNIs operating with different schedules can disrupt the system's intended behaviour and result in unpredictable and undesirable outcomes. Therefore, the time-triggered dispatcher ensures that all TTNIs switch schedules in unison to maintain system-wide synchronization and prevent inconsistencies.

5.1.7 Network Interface

The NI serves as the interface connecting the processing element, such as a core or a processor, with the routers in an NoC. The TTNI is an enhanced version of the NI that incorporates time-triggered and adaptive features, allowing the NoC to inject messages into the network based on a predefined schedule. Additionally, it enables the NI to switch schedules in response to context events. This enhanced functionality increases flexibility for scheduling and ensures that messages are delivered within the deadlines, which is crucial in real-time applications.

Non-Safety Critical Network Interface

The NSCNI is a temporal and spatial partitioning layer built on top of an NoC. The NSCNI is designed to support mixed-criticality systems, where different system components have different levels of criticality. To achieve this, the NSCNI provides different communication channels with different levels of criticality. These channels can be configured as time-triggered, rate-constrained, or best-effort. Time-triggered channels are deterministic, meaning that messages are transmitted at predetermined times. This is useful for critical components that require guaranteed timing for their messages to be delivered. Rate-constrained channels limit the rate of messages that can be transmitted. This is useful for components that require a certain level of bandwidth but do not have strict timing requirements. Best-effort channels have no guarantees on message delivery but can be used for non-critical components where occasional message loss is acceptable [AO15]. Figure 5.9 shows the building blocks of NSCNI. It consists of five building blocks: Core interface, adaptive time-triggered dispatcher, packetization, depacketization, and memory that stores paths. In addition, there is an AXI wrapper, an interface that connects the tile to the NI, and the router interface that connects NI to the router. The components of the NSCNI are described below:

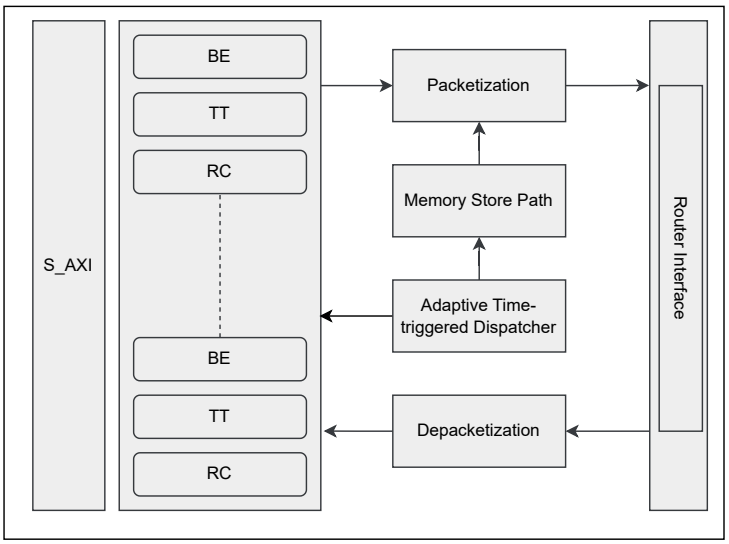


Figure 5.9: Non-safety critical NI

- The core interface is the communication link between the processing element and the NI. It facilitates the exchange of messages between cores and NIs.

The core interface is equipped with a FIFO port that can be configured as an input and output port according to the schedule. The output port of the core interface is responsible for queuing messages from the processing element. It stores these messages until they are ready to be transmitted. On the other hand, the input port of the core interface receives messages from the NoC side and forwards them to the processing element for further processing. The core interface incorporates an AXI (Advanced eXtensible Interface) wrapper to ensure efficient communication with the processing element. This wrapper enables optimized data transfer between the core interface and the processing element. When the processing element intends to send a message to the NoC, it initiates an AXI write transaction. This transaction allows the processing element to queue the new message to the output port of the core interface. Before transferring the data to the core interface, the AXI protocol ensures a proper handshaking protocol between the processing element and the NI. This ensures that the data is reliably and accurately transmitted. Each port within the core interface can be configured by the schedule as BE (best-effort), TT (time-triggered), or RC (rate-constrained). These designations determine the specific characteristics and behaviour of the messages transmitted through that port.

- The adaptive time-triggered dispatcher is responsible for scheduling communication and reconfiguring the schedule when a context event occurs. The module consists of the schedule loader and the time-triggered dispatcher, as shown in Figure 5.10. The schedule loader is responsible for reconfiguring the schedule within the TTNI whenever a context event occurs. It receives signals from the adaptation unit indicating a context event and initiates the necessary schedule reconfiguration process. This ensures that the TTNI can adapt to changing circumstances and requirements. On the other hand, the time-triggered dispatcher ensures the injection of messages into the NoC based on a predefined schedule. The dispatcher logic within the time-triggered dispatcher component retrieves schedule information from the schedule memory. It compares this schedule information with the current time of the global time base. When a match is found, the dispatcher logic triggers the injection of messages located in the core interface, enabling them to be transmitted through the NoC. The adaptive control unit within the schedule loader functions as a state machine that controls the schedule switching within the TTNI when a global context event occurs. It manages the transition between different schedules to ensure the TTNI operates with the appropriate schedule based on the global context event. The schedule memory is a dual memory that stores the current and next schedules to be adopted. The dispatcher reads the current schedule from this memory and uses it to determine the timing and sequence of message injection. The progression of global time guides the dispatcher's actions as it aligns message transmission with the predefined schedule.

- Packetization and depacketization are crucial processes in the communication flow between the core interface and the router within the TTNI. 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

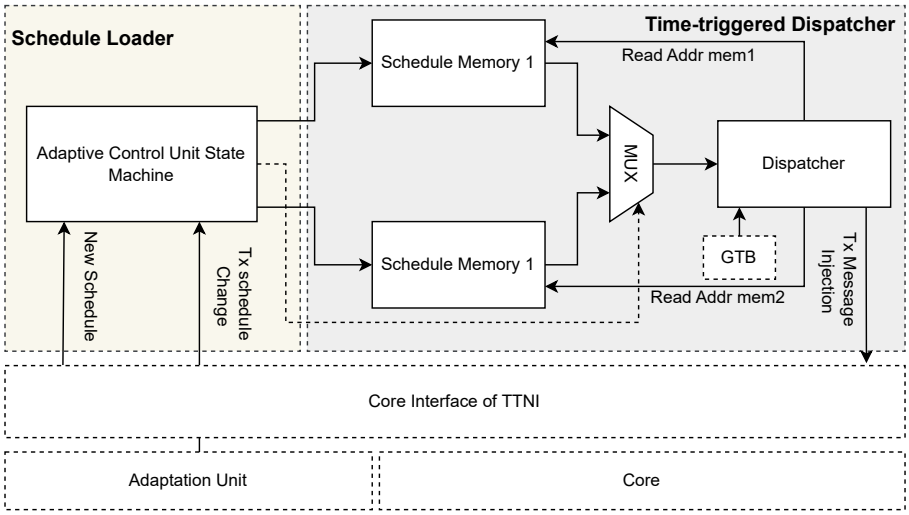


Figure 5.10: Adaptive time-triggered dispatcher in NI

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 TTNI. When packetization forms a packet, the message's path is retrieved from the memory and added to the head flit of each packet.

Safety Critical Network Interface

The SCNI extends the functionality of the NSCNI by incorporating a seamless redundancy mechanism. It introduces additional components, such as the redundancy sender and receiver controller, to enable efficient redundant transmission within the NI, as shown in Figure 5.11. 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).

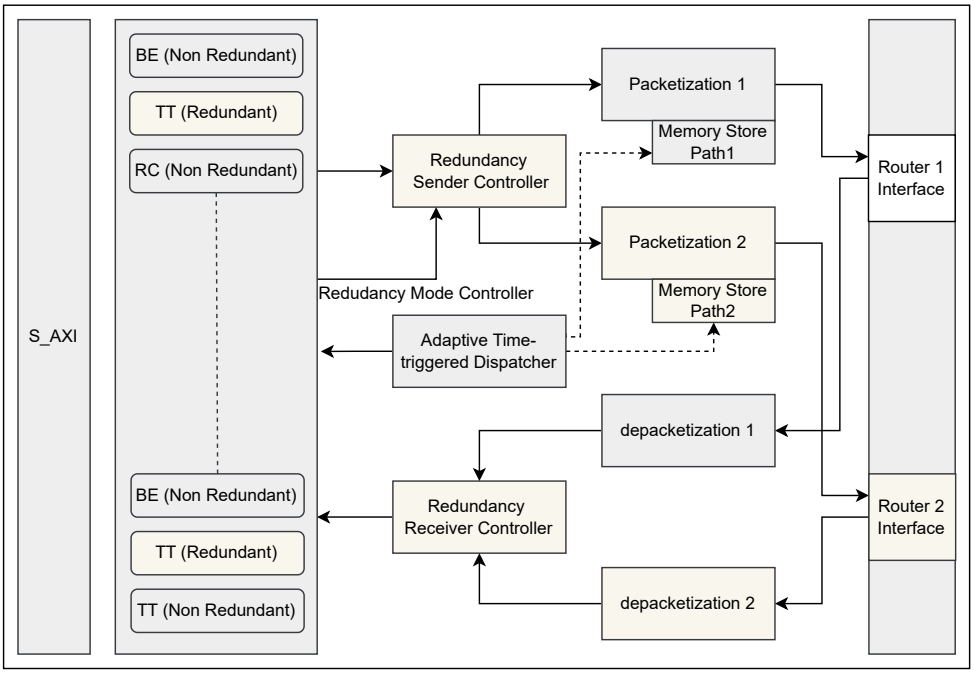


Figure 5.11: Safety critical NI

- 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 follows a predefined schedule set by the Time-Triggered (TT) 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 behaviour of the redundancy sender controller is summarized in Table 5.2. 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 packetization block receives data, and the second packetization block remains inactive. Consequently, transmission is handled exclusively by the first packetization block. Similarly, when the control bits are "001," the second packetization block receives data, while the first packetization block remains inactive. In this scenario, the second packetization block handles the transfer of flits (flow control units).

- 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

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

Table 5.2: Redundancy sender controller operation

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 5.3. 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.

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

Table 5.3: Redundancy Receiver Data Selection

5.1.8 Router

The router architecture used in ATTNoC is described in this section. The router consists of input buffers, a connection matrix, a set of output buffers, and a control unit (VC allocator, SW allocator). The buffers at the input and output ports queue the data transmitted over the channels. These buffers enable local storage of data that cannot be immediately forwarded. Each router has five channels corresponding to South, East, North, West, and Local. The South, East, North, and West channels connect neighbouring routers, while the Local channel communicates with the core. The router uses a source routing algorithm to route data from the input port to the output port of the router. This means that the routing decision for the output port is made within the router, depending on the routing opcode. The destination

of each packet is defined in the first flit (header) at the source NI. Each opcode consists of five bits that configure the direction in the router. The routing opcodes for the corresponding directions are provided in Table 5.4. Wormhole flow control is employed in ATTNoC because it supports low latency and reduced buffer size compared to other flow control methods like store and forward, making it suitable for real-time communication [Bet97].

Direction	Routing opcode
"NORTH"	"5'b00001"
"EAST"	"5'b00010"
"SOUTH"	"5'b00100"
"WEST"	"5'b01000"
"LOCAL"	"5'b10000"

Table 5.4: Routing opcode

Figure 5.12 shows the internal structure of the router. The router consists of six main components, which are explained below:

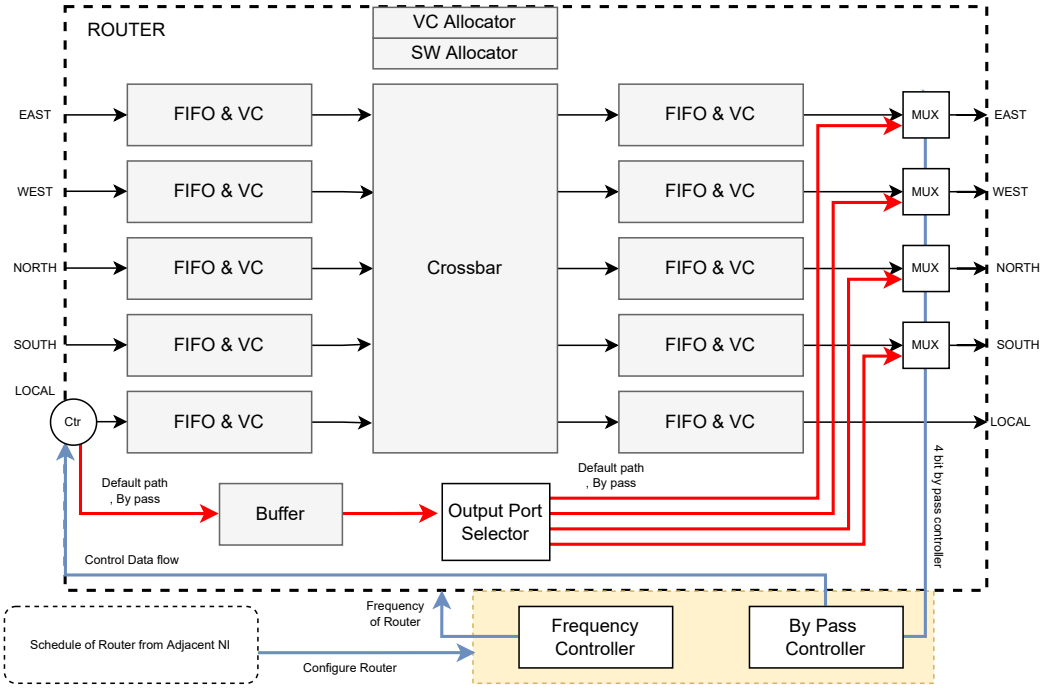


Figure 5.12: Router architecture of the ATTNoC

- First-In-First-Out Buffer: This memory is used to buffer incoming and outgoing data in the router. The buffers of FIFO are replicated in the input and output ports of the switch for virtual channel communication.
- Crossbar: This component is used to connect the input and output ports of the router. All possible input data lines are connected to the input ports of the crossbar multiplexers. The output data of the input data lines is then

controlled by the arbiter using high-priority-based messages. Time-triggered messages have the highest priority, followed by rate-constrained and best-effort messages.

- Router configuration (cf. yellow area in figure 5.12): This module is responsible for configuring the frequency of routers according to a predefined schedule from the NI. It also controls the data flow of the router. Flits can either flow through the crossbar, which is the normal operating mode of the router, or bypass the crossbar, as shown in Figure 5.12, to tolerate a permanent fault occurring in the input buffer, crossbar, or output buffer.

- Frequency Controller: This block is responsible for setting the frequency of each router based on the configuration received from the schedule provided by NI.

- Bypass controller: This component allows the router to transmit data through the router or bypass path. The bypass path is enabled only when data from the NI interface is blocked due to the router's local input port, router crossbar, or output port failure.

- Output port selector: This component determines the appropriate output port for a flit within a router when the bypass path transmits data. First, the flit is buffered until the neighbouring router is ready to receive it. Then, the output port selector uses the opcodes in the header flit to decide which output port to forward the data to the next hop.

- VC Allocator and SW Allocator: These components are part of the control unit and are responsible for allocating virtual channels (VCs) and switch (SW) resources within the router. The VC allocator assigns VCs to different traffic flows, while the SW allocator determines the appropriate paths for the flits to traverse through the switch matrix.

5.1.9 Global Time Base (GTB)

The ATTNoC operates with multiple clock domains, including those used in the heterogeneous core, NoC, and GTB. Power management requires different clock frequencies for individual IP blocks, with low-frequency clocks used for some blocks and high-frequency clocks for specialized hardware. To achieve system-wide time synchronization, the GTB is used in the ATTNoC. The GTB employs the IEEE I588 time format and uses a 64-bit counter vector driven by a macro-tick clock with a frequency denoted as $f_{macrotick}$ [Kop92]. The switching rate of a particular bit, called the period bit, determines the period. The counter vector of the time format implements the phase as a slice whose width is called the *phase slice width* and is configurable. A period starts when all bits in the phase slice are set to "0". A bit to the right of the period bit of that period is the location of the phase slice. *Dead bits* are not as crucial as a macro-tick bit. The implemented counter vector is shown in Figure 5.13, where the period bit and phase slice are designated as follows.

A 64-bit binary counter vector based on the physical second and nanoseconds are used to construct the unified time format. This time format defines nanoseconds as the largest granularity of the global time base. So the largest granularity is 2^{-31}

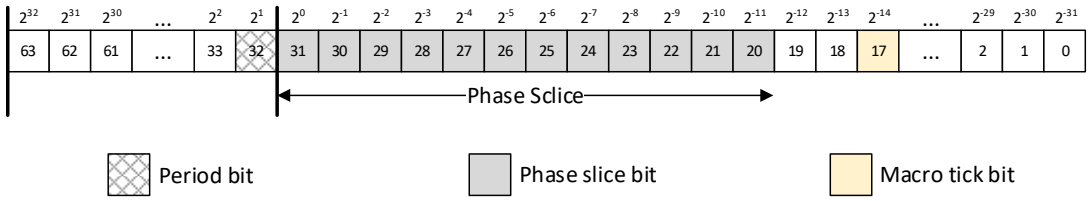


Figure 5.13: GTB Based on IEEE 1588 time format

sec ≈ 465 ps, while the global time base has a horizon of about 2^{32} sec ≈ 136 years. Figure 5.14 shows the 64-bit GTB [Abu17].

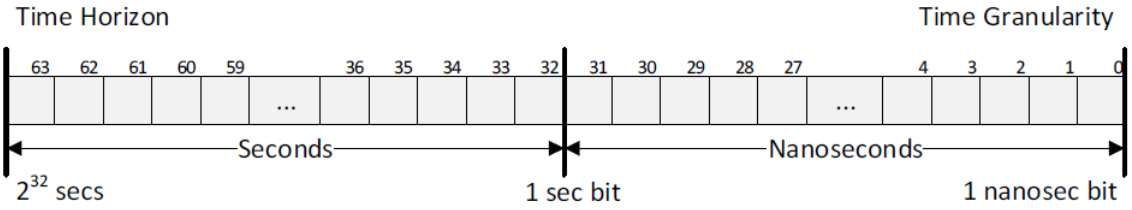


Figure 5.14: 64 Bit GTB

The main goal of the GTB, as stated in its introduction, is to provide a global time base for all tiles, even if the ATTNoC components operate with multiple clock domains. The time-triggered dispatcher in the TTNI and the adaptation unit can use the global time base in conjunction with a precomputed schedule loaded into each schedule memory of the TTNI or the adaptation unit to determine the appropriate time to start an action or to inject messages into the NoC.

Chapter 6

Energy Efficiency and Fault Tolerance for ATTNoC

The development of Network-on-Chips (NoCs) is driven by the need for efficient and reliable communication in System-on-Chip (SoC) designs. Fault tolerance and low-power techniques are critical factors in NoC design, as they directly impact the system's reliability and energy efficiency. Preventing failures and ensuring system reliability is paramount in NoC design, while low-power techniques are vital for extending the battery life of mobile devices and other low-power systems. This section addresses the fault tolerance and power-saving techniques used in the Adaptive Time-Triggered NoC (ATTNoC). These techniques include adaptation, redundancy mechanisms, and time-triggered frequency scaling. By combining these techniques, ATTNoC provides a reliable and energy-efficient communication solution that is essential for the success of modern SoCs.

6.1 Adaptation in Time-Triggered Network-on-Chip Architecture

The ATTNoC architecture supports dynamic schedule adjustments during run-time based on specific events such as permanent faults in NoC resources, including NIs, routers, cores, and links, and slack occurring in cores and NoCs. These adjustments are made while preserving important time-triggered properties, such as synchronization, temporal predictability, and avoidance of resource contention [Obe+19]. This approach enhances the system's reliability during fault events and optimizes energy efficiency by effectively using slack in cores and NoCs without compromising system performance [Obe+19]. Through adaptation, ATTNoC can tolerate permanent faults in NoC resources by reconfiguring the network with new schedules to isolate faulty components and migrate tasks or messages to other resources. Additionally, the ATTNoC architecture can adjust its operating frequency dynamically to accommodate workload fluctuations, improving energy efficiency. The following section describes the motivation for adaptation in the ATTNoC architecture and explains how the main blocks of the adaptation unit achieve the adaptation techniques.

6.1.1 Energy Efficiency for ATTNoC

The importance of energy efficiency in safety-critical systems, such as avionics, cannot be overstated. Inefficient energy usage can cause critical issues, such as component failure, fire hazards, and reduced battery life. To ensure safe operation during flight, avionics systems must be designed with stringent energy efficiency requirements [AMP19]. Traditional energy management techniques, such as Dynamic Voltage and Frequency Scaling (DVFS) and clock gating, have limited applicability in these systems due to the challenges associated with certifying multi-core processors and determining Worst-case Execution Times (WCETs) [Pro16]. Safety-critical systems demand predictable timing behaviour due to their critical applications. Here, the ATTNoC-based multi-core architecture emerges as a promising solution for power management in safety-critical systems without compromising timing predictability. The architecture is built on an adaptable time-triggered system, offering inherent determinism vital for safety-critical applications. One of its main advantages is time-triggered frequency scaling, which allows routers to adjust their operating frequencies according to a predefined schedule within the system. This feature of frequency scaling guarantees that the power consumed is optimized while the system's timing predictability remains unaffected. Moreover, the adaptive nature of the ATTNoC architecture enables dynamic switching between schedules in response to specific events, such as slack or changes in system demands. The flexibility of ATTNoC allows it to adapt to changing conditions efficiently, ensuring optimal system performance and meeting the demands of real-time applications [Obe+19].

6.1.2 Fault Recovery for ATTNoC

In safety-critical embedded systems, dependability and fault tolerance are of utmost importance. N-modular redundancy is a widely used technique for ensuring fault tolerance, but it is costly in terms of chip resources and energy consumption. In particular, emerging safety-critical application areas like autonomous vehicles demand more cost-effective solutions. Fault recovery provides a promising alternative that involves re-configuring the system to avoid failed resources. For example, if a sensor fails in an autonomous vehicle, the system can reconfigure to rely on other available sensors to maintain safe operation. This can be achieved by using adaptation techniques in the ATTNoC architecture, which allow the ATTNoC to reconfigure its schedules when a context event occurs in the NoC resources, such as faults in links, routers, NIs, and cores, and isolating the faulty components to maintain the functionality of the systems. ATTNoC uses precomputed schedules to reconfigure the system during run-time. By switching between schedules, the ATTNoC can recover from faults [Obe+19], [MAO18].

6.1.3 Architecture of Adaptation Unit in ATTNoC

The Adaptation Unit (AU) is a crucial component of the ATTNoC architecture. Its primary function is to enable schedule switching within the network when specific context events, such as slack or faults, occur in the NoC resources. This capability allows the NoC operation to continue even in the presence of permanent faults by re-configuring the ATTNoC with a new schedule, isolating faulty components, and redistributing tasks/messages to other resources. In ATTNoC, each Network

Interface (NI) can dynamically adjust its message injection time or the operating frequency of the NoC during run-time based on context events occurring in the NoC resources. To reconfigure the schedule in the NoC, each adaptation unit exchanges a local context to ensure that they all have the same information, known as the global context. Based on this global context, the new schedule is determined, which is used to reconfigure the ATTNoC. The adaptation units are interconnected in a ring structure as depicted in Figure 6.1, facilitating the efficient distribution of their local contexts.

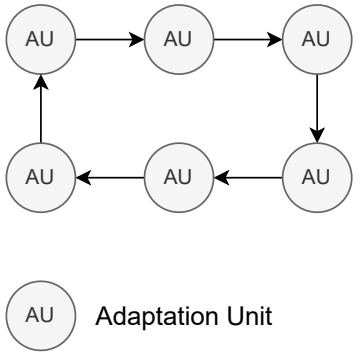


Figure 6.1: Ring topology of adaption units (6 AUs)

In Figure 6.1, six adaptation units are connected in a ring structure. The context agreement, a sub-component of each adaptation unit, is responsible for exchanging local context information between adjacent context agreements. However, permanent and transient faults in the context agreement or a link connecting two context agreements can result in inconsistencies in the network. This may lead to different global contexts being maintained by different units. To address this issue, a fault-tolerant technique suggests adding a triple-ring to the existing structure [Len20]. The triple ring enables the exchange of triple redundant messages and facilitates communication among non-direct neighbouring context agreements, as illustrated in Figure 6.2. These redundant messages can serve as message recovery through a voting mechanism.

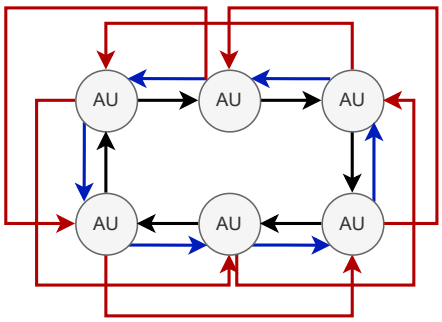


Figure 6.2: Triple-ring topology of adaption units (6 AUs)

The architecture of the adaptation unit in ATTNoC is depicted in Figure 6.3. It consists of several components, each serving a specific purpose to ensure the smooth operation of the unit. These components include the context monitor (CM), context agreement (CA), time-triggered dispatcher (TT dispatcher), and schedule memory.

Each component operates according to a schedule managed by the time-triggered dispatcher. The CM reads the local context from the adjacent core and reports it to the CA. The CA ensures that all operational CAs have consistent context information, ensuring consistent system behaviour. The schedule memory contains the precomputed multiple schedules, and the relevant schedule is chosen based on the agreed context provided by the CA. Finally, the TT dispatcher triggers the CM, CA and schedule switching in the TTNI based on the schedule retrieved from the schedule memory of the time-triggered dispatcher of the adaptation unit to synchronize the operation of the distributed adaptation unit in the ATTNoC.

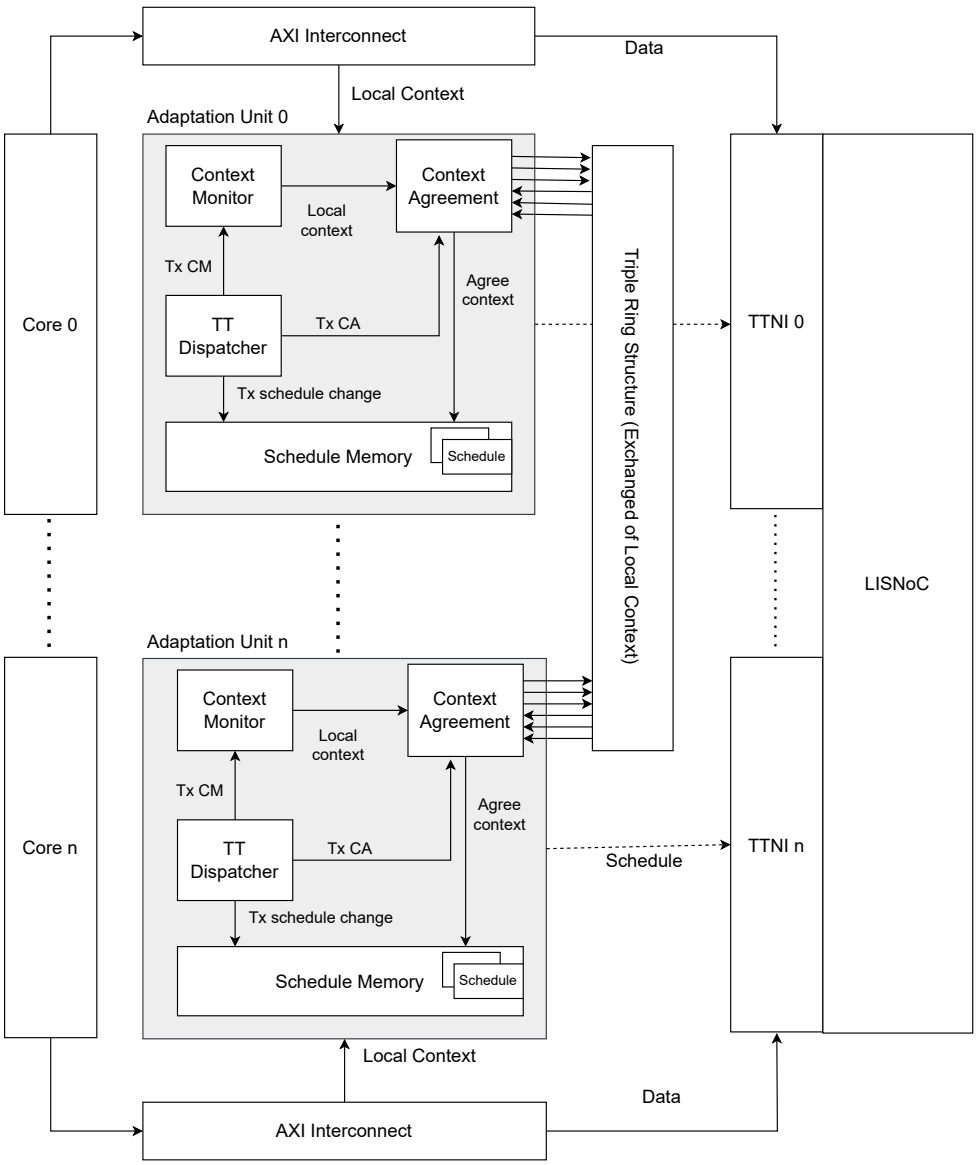


Figure 6.3: Architecture of adaptation unit in ATTNoC

The following section will discuss the potential failure modes, fault containment regions (FCR), and fault assumptions related to the adaptation unit architecture. Afterwards, we will delve into each component's architecture and state machine within the adaptation unit. The state machine outlines how each component behaves when it changes states, responds to inputs, and performs operations.

6.1.4 Fault Model in the Adaptation Unit

The fault tolerance techniques used in the adaptation unit (AU) ensure that the exchange of local context between context agreements (CA) within the ATTNoC is not hindered by faults, preserving system functionality. However, fault tolerance mechanisms used in the AU do not address faults originating from the context monitor (CM). Corrupted information introduced by CMs is not within the scope of the current fault tolerance approach.

Failure Mode

The fault model considers both permanent and transient faults. Therefore, the CAs block must be able to handle any incorrect behaviour displayed by CAs connected through a triple-ring structure and any potential link failures.

Fault Containment Region (FCR)

The fault containment region (FCR) is a designated area where faults can occur during run-time and be tolerated to limit immediate impact. In the adaptation unit architecture, each CA block and the link that connects CAs are included in the FCR. This guarantees that any faults within the area will not have a negative impact on other FCRs. Even if permanent faults occur and affect the links between CAs, the exchange of local context should still proceed correctly within each FCR.

Fault Assumptions

The fault model defines three fault types related to message correctness within the adaptation unit:

- Corrupted message: A message is considered corrupted if received but differs in context from the original message. Corruptions can occur in the CA component during context exchange or due to transmission failures on the link.
- Lost message: A message is lost when it fails to reach its intended recipient due to a fault with CAs or the links connecting them.
- Delayed message: A message is delayed if it arrives at the destination in CA significantly later than expected due to transmission failures or delays in the links or CAs.

6.1.5 Adaptation Unit Architecture

The adaptation unit plays a crucial role in reconfiguring schedules within the ATTNoC. Herein, we provided a detailed description of its constituent components.

Context Monitor (CM)

The context monitor (CM) is responsible for collects and reports adaptation-relevant information to the adaptation unit by checking the current context values from the adjacent core. To observe the slack that occurred in the core, the execution of each task in the core was recorded, including the actual execution time compared

to the WCET. This involves modifying the code of each task to track the start and end times by using instrumentation techniques [Kas+15]. The CM then uses this information to form the local context. Figure 6.4 illustrates that the CM comprises three key elements: CM interface, controller, and memory. These components work together to collect and analyze data on task execution and slack, enabling the system to optimize resource usage while meeting real-time requirements.

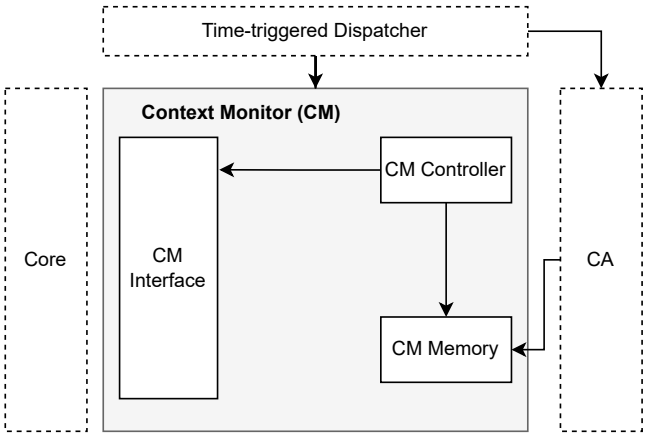


Figure 6.4: Architecture of context monitor

The CM interface is an AXI interface that facilitates communication between the core and the CM, receiving the local context from the adjacent core. On the other hand, the CM controller functions as a state machine that encodes context information from the CM interface. The CM memory is a dedicated memory unit that stores the encoded local context information from the CM controller. The CM controller operates through a state machine, as depicted in Figure 6.6. It starts in a "Wait for Triggered Context" state, where the reported context is set to null, and no context is detected. Upon being triggered by the time-triggered dispatcher, the CM moves to the "Read Context" state, which reads the local context from the CM interface and checks its validity. If the context is deemed valid, indicating the detection of slack or a permanent fault, the CM controller transitions to the "Write Context" state. Here, it encodes the received context into a 32-bit bit string with context information, as depicted in Figure 6.5, and stores the 32-bit string in the CM memory. If no context event is detected, the state machine will move to the "Context Null" state and convert the null context into a 32-bit string. In both the "Write Context" and "Context Null" states, the state machine will then return to the "Wait for Triggered Context" state and wait for the next trigger from the time-triggered dispatcher of the adaptation unit.

Reserve (5-bit)	Context Validity (1-bit)	Context type (2-bit)	Task ID (4-bit)	Context Content (15-bit)	Device ID (5-bit)

Figure 6.5: 32-bit-string local context

Within a system, a 32-bit string's local context manages data related to events, as shown in Figure 6.5. It consists of six fields, with the first reserved for future expansion. The second field, "Context Validity," indicates whether the context is valid. This is useful when the core detects a slack or fault in the network. The

third field, "Context Type," specifies whether the event is a slack or a fault. The fourth field is the "Task ID," which contains the ID of the scheduled task where the slack event originated. The fifth field is the "Context Content," which encodes the fault in the adjacent NI, link, router, or core connected to the adaptation unit. Depending on the context, it can also indicate the percentage of slack that occurs in the cores. Finally, the sixth field is the "Device ID," which contains the ID of the adaptation unit.

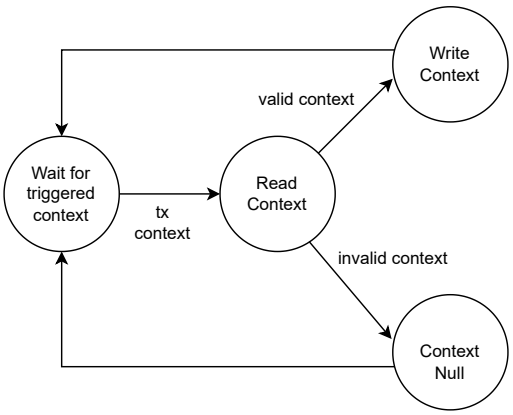


Figure 6.6: Context monitor state machine

Context Agreement (CA)

The Context Agreement (CA) allows the exchange of local context information between multiple distributed CAs, ensuring that all CAs are aware of local context events occurring in other CAs. The CA employs a triple-ring structure, as shown in Figure 6.2, as a network topology to facilitate communication between CAs, thus improving reliability and providing redundancy during the transfer of local context. This structure enables local context exchange in three ways: clockwise, counterclockwise, and through a non-adjacent CAs connector. The local context received from the clockwise channel is stored in the left memory of each CA, while the counterclockwise channel provides local context stored in the right memory. Additionally, the redundant local context received through the third channel is stored in the central memory, as shown in Figure 6.7. Once the local context has been distributed to all CAs, the CA compares the information stored in the three memories to detect any corrupted context or discrepancies. The redundant local context information received over the third channel can be used for voting to determine the correct context information. The triple-ring structure is designed to withstand various faults that could affect the exchange of local context, such as corruption, delays, or loss of local context in the CAs or the links connecting them. The CA architecture comprises four fundamental components: the CA controller, left memory, right memory, and central memory, as depicted in Figure 6.7. These components facilitate the exchange of local context between the distributed CAs in the ATTNoC architecture. By using redundancy mechanisms within the CAs, this architecture ensures a consistent and reliable exchange of local context.

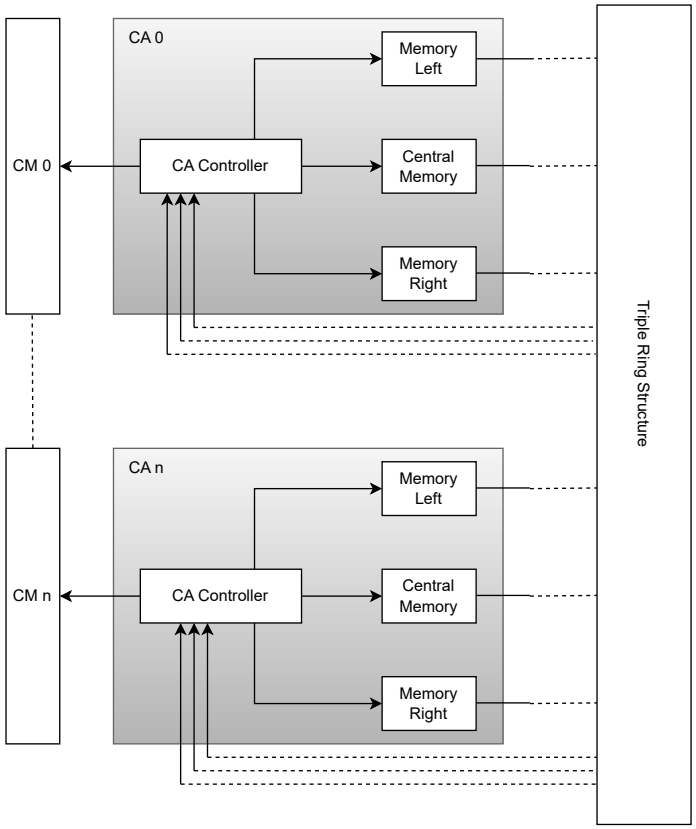


Figure 6.7: Context agreement architecture

The CA controller manages three memories in the CA block: left, right, and central. These memories store the local context before exchanging it with other CAs. Memory left receives the local context from the clockwise direction, memory right receives it from the counterclockwise direction, and the central memory receives the local context from the two CAs that are not direct neighbours, as shown in Figure 6.2. The CA operates on a predefined schedule set by the time-triggered dispatcher. As shown in Figure 6.8, the state machine of the CA controller responds to a triggered signal from the time-triggered dispatcher. It begins in the "Wait for Triggered" state, where its output memory is initialized and disabled. When triggered, the state machine transitions to the "Read context from CM" state. In this state, the CA reads the local context from the CM and saves it in a local register. After collecting the context events from the CM, the state machine transitions to the "Write" state, where it writes the input context to the three available memories (left, right, and central). Then, it moves to the "Read" state, where it synchronously reads input context from neighbouring units. These contexts are saved in the output memories as input contexts, along with the CA ID of the context origin. The state machine then moves to the "Check all context" state, where it compares the received CA IDs with the local ID. If any received IDs match the local ID, indicating that the local context is distributed to all CAs, the state machine then transitions to the "Compare received context" state. In this state, the three input contexts from the three memories (left, right, and central) are compared. If the three copies of input contexts from the clockwise, counterclockwise, and third ring are equal, the state machine transitions to the "Agree context from left Memory" state. However, if one

of the received contexts differs from the other two, the state machine transitions to the "Agree on one of two contexts" state to tolerate the faulty context. Suppose all the received contexts differ from clockwise, counterclockwise, and the third ring connection. In that case, the state machine transitions to the "report redundancy failure," reporting that the redundant paths in the CA are corrupted. For both the "Agree on context from left Memory" and "Agree on one of two contexts" states, the state machine returns to the initial state, where it waits for the next trigger from the time-triggered dispatcher.

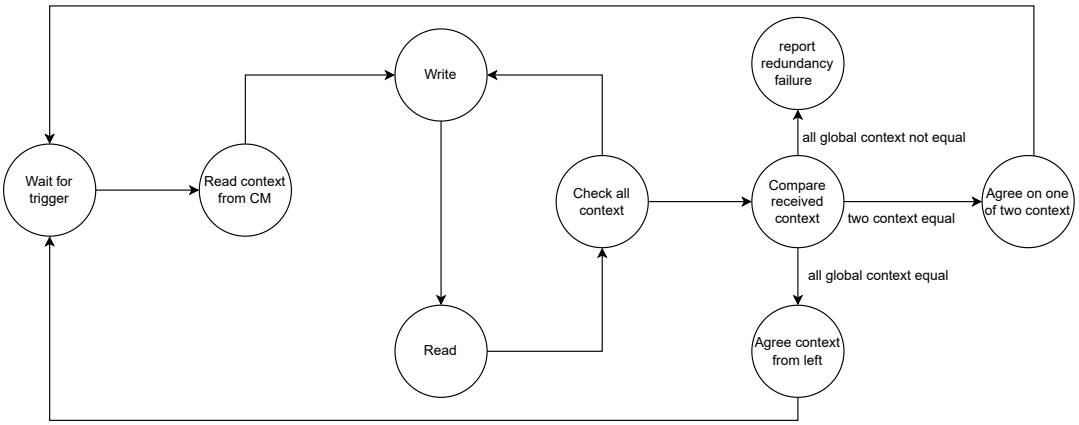


Figure 6.8: Context agreement state machine

Time-triggered Dispatcher in Adaptation Unit

The time-triggered dispatcher is a crucial part of the adaptation unit, responsible for triggering the operation of the CM, CA, and schedule switching in the ATTNoC according to a predefined time. Figure 6.9 shows that it comprises two essential components: the dispatching logic and schedule memory.

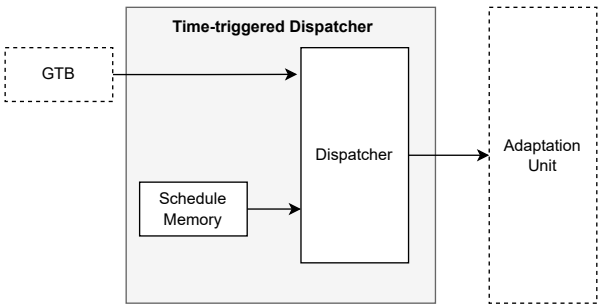


Figure 6.9: Time-triggered dispatcher

The schedule memory stores the precomputed schedule for the adaptation unit. The dispatcher logic triggers the CM, CA, and schedule switching within the ATTNoC according to this precomputed schedule to synchronize all distributed adaptation units within the ATTNoC. The predetermined schedules are computed offline and loaded into the schedule memory during system start-up. To ensure the execution of scheduled actions, the dispatcher retrieves the schedule from the schedule memory and compares it with the current time received from the global time base

(GTB). When the dispatcher detects a match between the schedule and the current time, it triggers the corresponding adaptation unit to execute the scheduled action [AO15], [PNO23].

Timeline for the Agreement

The agreement protocol in the ATTNoC follows a specific timeline to ensure synchronized activities among the adaptation units and TTNI modules, enabling cohesive system operation. This timeline includes several scheduled activities that occur in a specific order. These activities include synchronized schedule switching, triggering the CA for local context exchange, and activating the CM to read the reported local context from the adjacent core. The timeline begins with the adaptation unit collecting essential information about the system's current state by gathering local context data. This data plays a crucial role in making adaptation decisions. Once collected, the local context data is transmitted for exchange between the adaptation units. During this exchange, the assumed context state remains unchanged to maintain consistency. This is achieved by temporarily freezing the context state, ensuring that the exchanged information reflects a consistent context without concurrent updates. While the context state is frozen, any new context events that occur are collected but not immediately integrated into the ongoing local context exchange. Instead, these events are stored for future exchanges between the adaption unit and the CM. This storage allows for an up-to-date and comprehensive understanding of the system state, as the stored events are incorporated into subsequent exchanges of local context. The timeline, illustrated in Figure 6.10, shows the overlap between the propagation phase of a protocol execution starting at time t and the context collection phase of the subsequent execution of the protocol starting at t + 1 [Len20].

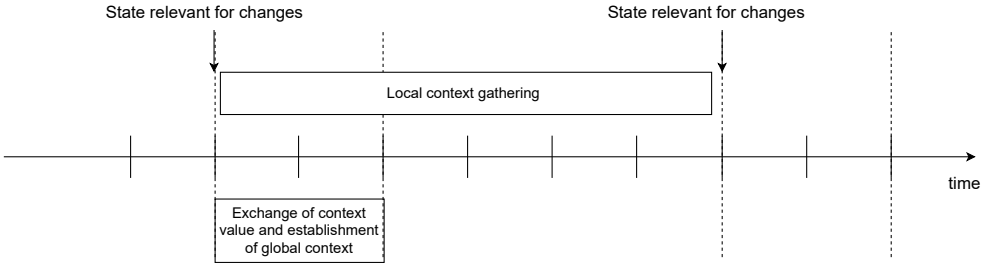


Figure 6.10: Protocol timeline: The protocol phases overlap, as the collection phase of the next execution starts as soon as the propagation of the current execution begins [Len20], [Obe+19]

6.2 Time-Triggered Frequency Scaling for ATTNoC

Time-triggered frequency scaling (TTFS) is a low-power technique used in the NoC to adjust the clock frequency of individual routers according to a predefined schedule [Nam+21] [NOO23]. In an ATTNoC-based system, the processing elements communicate and perform computational tasks according to a predetermined schedule, enabling deterministic communication among the networks. TTFS leverages this determinism to scale the frequency of a NoC router according to a predefined schedule,

thus minimizing power consumption. This technique offers several advantages over other power management methods. One key advantage is the ability to adjust the frequency of the routers or clock gate the routers according to a predefined schedule, resulting in improved power efficiency for the NoC. Additionally, the time-triggered nature of TTFS guarantees system predictability and determinism, even when frequency changes occur. In the following sections, we will examine the structure of TTFS and investigate the different methods offered by TTFS for enhancing energy efficiency.

6.2.1 Architecture of TTFS

TTFS plays a vital role in the ATTNoC architecture by enabling frequency scaling at the router level based on the schedule received from the TTNI. This capability is crucial for efficient power management in the ATTNoC. As shown in Figure 6.11, the example illustrates an ATTNoC architecture equipped with router configuration responsible for configuring the operating frequency used by the routers and selecting the router communication mode, whether it is a normal operation where messages pass through the crossbar, or a bypass mode to bypass the crossbar when permanent faults occur in the routers, as described in Section 5.12. The TTFS functionality is integrated into the router configuration (highlighted in yellow in Figure 6.11). It facilitates the configuration of the frequency of each router, which the frequency controller manages.

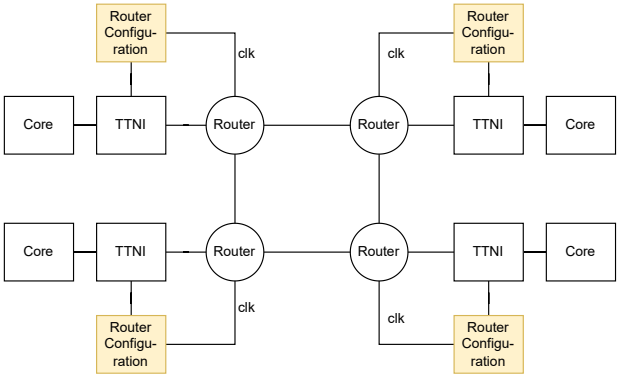


Figure 6.11: TTFS in ATTNoC : Example 2x2 mesh topology

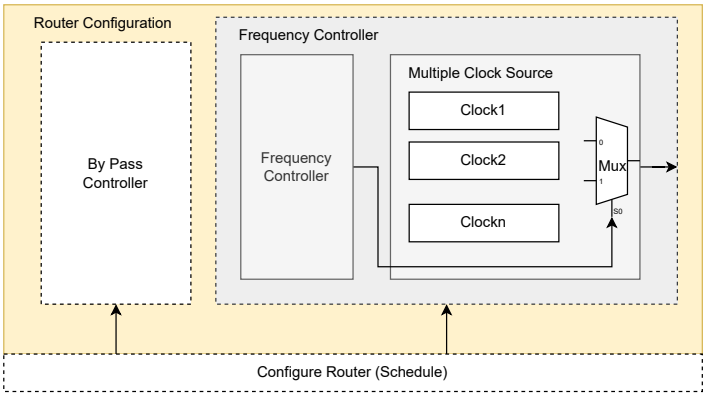


Figure 6.12: Block diagram of TTFS

The TTFS comprises two major blocks: Frequency controller and multiple clock source, as shown in Figure 6.12. The frequency controller is responsible for selecting the frequency to be used by the router according to the schedule. The multiple-clock source includes multiple clock domains, and the output of this multiple-clock source is the frequency that operates the router's clock. Therefore, the frequency controller determines the output frequency of the multiple clock source. This frequency controller is a state machine (depicted in Figure 6.13) that regulates the clock frequency of each router in the ATTNoC. The state machine of the frequency controller transitions between the states "WaitTrigger", "CheckFreqUsed", "FullFrequency", "HalfFrequency", and "clock gated". The state machine begins in the "WaitTrigger" state, where it waits for a trigger signal from the dispatcher of the TTNI. When a trigger signal is received, the state machine transitions to the "CheckFreqUsed" state. In this state, the state machine checks the FreqUsed value and transitions to the appropriate state depending on the frequency. If FreqUsed set by the schedule is full, the state machine transitions to the "FullFrequency" state. In this state, the frequency controller configures the router's clock to operate in full frequency mode and then transitions back to the "WaitTrigger" state. If FreqUsed is half, the state machine transitions to the "HalfFrequency" state. In this state, the frequency controller configures the router's clock to operate at half frequency before returning to the "WaitTrigger" state. Finally, if FreqUsed is clock gated, the frequency controller operates the router's clock with a clock gate, then goes back to the "WaitTrigger" state and waits for another trigger signal from the dispatcher of the TTNI. This approach ensures that the frequency controller adjusts the frequency of routers according to the predefined schedule, enhancing energy efficiency while preserving predictability in NoC communication.

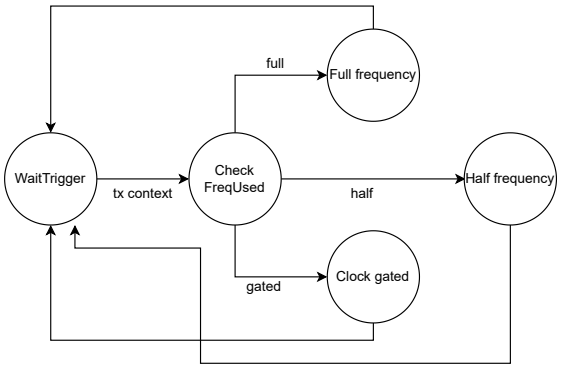


Figure 6.13: State machine of frequency controller

6.2.2 Different TTFS Techniques in ATTNoC

Reducing power consumption is a major concern in NoC architectures. To address this challenge, TTFS has introduced three techniques using global, cluster, and router-based approaches. These techniques are designed to effectively decrease power consumption within the ATTNoC.

- Global Level: This approach is employed in the NoC architecture to efficiently manage power consumption by allowing the NoC to adjust its frequency during

run-time according to a predefined schedule. At the global level, the frequency of NoC routers is uniform. This means that when communication is scheduled, all NoC routers operate with the same frequency determined by the schedule. Moreover, when any communication is scheduled in the NoC, the frequency of all routers is clock-gated according to the same schedule. To achieve this, a schedule dictates the frequency used in the NoC routers. During the shared idle times of all routers, clock gating is applied to reduce power consumption effectively. However, all routers must have a common idle time for optimal energy efficiency. If there are variations in idle times between routers, some routers may remain active while others are idle, ultimately reducing the overall effectiveness of the approach [Nam+21].

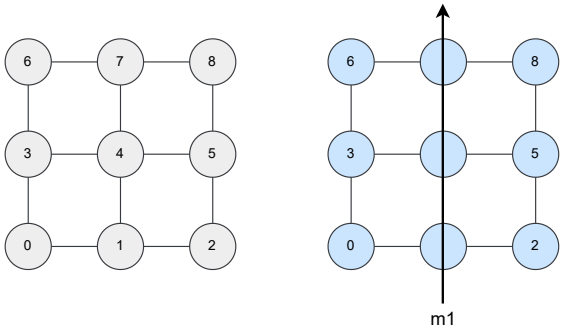


Figure 6.14: Global techniques, all routers are operated at the same frequency

Figure 6.14 illustrates the global technique employed in TTFS. On the left side of the figure, all routers are depicted in an idle state, with their frequencies being clock-gated during a shared idle time as per the schedule. This approach conserves energy and minimizes unnecessary resource usage during periods of inactivity. However, when communication is scheduled in the NoC, all routers become active, and the schedule assigns a predetermined frequency value to all routers. This is depicted on the right side of the figure, where the frequency of all routers is operated with a frequency defined by the schedule to promote efficient network operation during active communication periods.

- The cluster-level approach is used in NoC design to manage power consumption efficiently. It involves dividing the network into multiple regions or clusters, each with its own frequency control mechanism. In regions where data transmission is scheduled to occur, all the routers located in that region are set to operational frequencies defined by the schedule. However, in regions without data transmission, all routers in that region are either clock-gated or adjusted to a lower value according to the schedule to minimize power consumption. Clock-gating is a technique that turns off the clock signal to a specific section of the circuit, resulting in reduced power consumption. This approach enables the NoC system to effectively reduce power consumption by adjusting the frequency of routers in specific regions according to a predefined schedule [Nam+21].

  Figure 6.15 illustrates the cluster technique used in TTFS. The ATTNoC is divided into two regions: Region one and region two. To conserve energy, the frequency of all routers in a region is clock-gated when all routers share

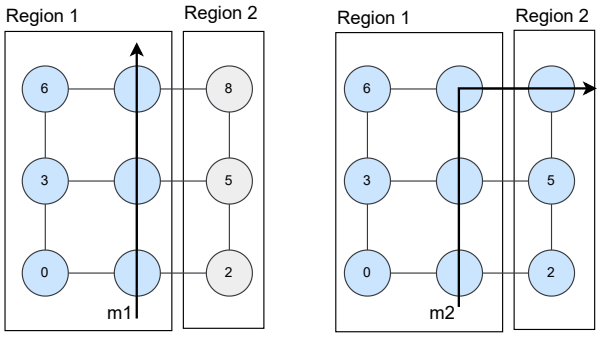


Figure 6.15: Cluster techniques, all frequencies of routers located in one region are operated simultaneously.

a common idle time. This approach minimizes power consumption during periods of inactivity. However, when communication is scheduled in a specific region, all routers in that region become active, and the schedule determines the frequency at which all routers in that region operate. This frequency assignment ensures efficient regional operation during active communication periods. The combined use of TTFS and the cluster technique contributes to reduced power consumption, effectively enhancing the energy efficiency of the NoC architecture.

- The router-based approach efficiently manages power consumption within the NoC by controlling the frequency of individual routers based on their predefined schedules. Active routers operate at frequencies determined by their schedules, enabling frequency scaling at the router level. On the other hand, idle routers are clock-gated and scaled down according to their specific schedules. Unlike cluster and global levels, where router frequencies are synchronized within the same region or the entire NoC, the router-based approach allows each router to have a different frequency. This feature provides precise control over the frequency of each router, leading to optimal power consumption aligned with the specific data transmission scheduled in advance [Nam+21].

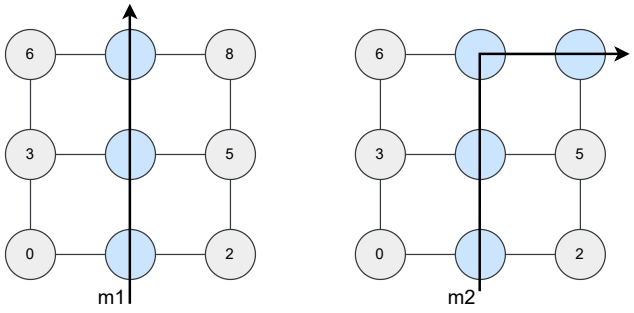


Figure 6.16: router-based techniques, the frequency of one router is scheduled individually

Figure 6.16 illustrates how TTFS employs a distributed frequency controller to manage the frequency of each router within the NoC based on a predefined schedule.

Active routers' frequencies are adjusted according to the schedule, whereas idle routers undergo clock-gating to conserve power. This approach effectively reduces power consumption and extends the battery life of battery-operated devices that use routers, ultimately enhancing energy efficiency.

6.2.3 Summary of TTFS Techniques in ATTNoC

The TTFS technique in ATTNoC introduces three distinct approaches to optimize power consumption: the global, cluster, and router-based approaches.

- Global Approach: The global approach aims to synchronize the frequency of all routers in the NoC. The frequency of active routers may vary according to a predetermined schedule, and routers are clock-gated when they have a common idle time, resulting in lower energy consumption in the NoC architecture.
- Cluster Approach: In the cluster approach, the NoC is partitioned into regions or clusters, each with its frequency control mechanism. The operating frequency within each cluster is synchronized and dictated by the schedule when data is scheduled for transmission. However, in instances where no data transmissions are scheduled within a particular cluster, that cluster's operating frequency is clock-gated per the schedule. This regional frequency control optimizes power consumption for specific network sections, enhancing energy efficiency.
- Router-based Approach: The router-based approach focuses on controlling the frequency of individual routers based on their specific schedules. Each router's specific schedule determines its operating frequency during its active time. The frequency of routers is scaled down and clock-gated when the routers are scheduled to be idle, contributing to power optimization and preserving deterministic communication behaviour within the NoC.

Overall, the TTFS technique compares these three approaches to find a suitable way to reduce power consumption within the NoC. By employing global frequency synchronization, regional frequency synchronization, and individual router frequency management, the TTFS technique enhances energy efficiency and optimizes power consumption in ATTNoC-based systems. Each approach provides unique advantages in managing power consumption in the NoC, contributing to the overall goal of energy-efficient operation in ATTNoC-based devices.

6.3 Seamless Redundancy in ATTNoC

The main requirement in mixed-criticality systems is to achieve fault containment, which aims to prevent faults in a non-safety-critical application from affecting safety-critical applications. This section introduces the seamless redundancy mechanism for on-chip networks based on ATTNoC, as presented in the system model in Chapter 5. The architecture is designed to support time-triggered messages with deterministic behaviour and minimal jitters, which helps minimize message transmission delays and ensures timely delivery within the ATTNoC architecture. The Safety-Critical NI (SCNI) plays a crucial role in supporting the redundancy mechanism by duplicating

critical messages at the NI and transferring them via different redundant paths in the NoC according to a predefined schedule. This redundancy mechanism enhances reliability and fault tolerance within the ATTNoC architecture [Nam+23].

6.3.1 Mixed-Criticality Architecture based on Mesh Topology

The architecture depicted in Figure 6.17 is an example of a mixed-criticality architecture that addresses the challenges of developing safety-critical systems requiring high reliability and fault tolerance. This design comprises four interconnected cores through an NoC, which provides a communication infrastructure for the system. The NoC contains two distinct types of NIs: Non-Safety Critical NI (NSCNI) and Safety Critical NI (SCNI). The NSCNI is a non-redundant NI connected to a single router in the NoC. It is suitable for use in non-safety-critical systems where high reliability and fault tolerance are not major concerns. Additionally, the NSCNI offers lower resource usage than the SCNI, making it an economical option in these systems. In contrast, the SCNI is a redundant NI connected to two routers in the NoC. The SCNI uses dual channels for time-triggered communication and transmits messages redundantly using two distinct paths to ensure reliable communication. This redundancy improves the reliability and fault tolerance of the system, making it a suitable choice for use in safety-critical systems. By incorporating both NSCNI and SCNI in the same architecture, the ATTNoC design enables efficient communication, improving the safety-critical system's reliability while optimizing resource usage in non-safety-critical systems. Additionally, this approach ensures that the system is reliable and cost-effective.

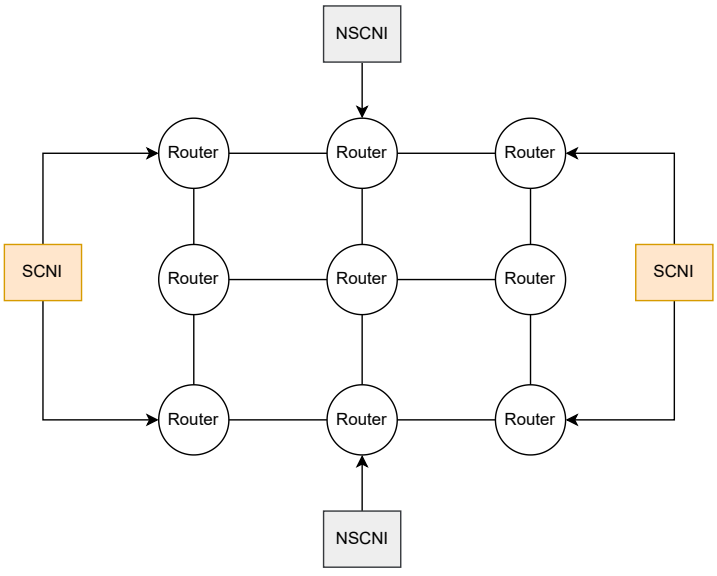


Figure 6.17: Mixed-criticality system with mesh topology

6.3.2 Fault Model for SCNI

The design of fault tolerance in the SCNI focusses on ensuring uninterrupted exchange of critical messages between SCNIs, thus preserving system functionality. It

is important to note that the fault tolerance design for SCNI does not explicitly address faults originating from the cores connected to SCNI.

Failure Mode

The fault model acknowledges the possibility of permanent and transient faults. SCNIs handle erroneous behaviour displayed by routers or link failures within the NoC while transmitting critical messages between SCNIs.

Fault Containment Region

The Fault Containment Region (FCR) encompasses the links and routers in the NoC. Its main objective is to prevent faults from propagating to other FCRs. Even if faults affect links and routers, the exchange of critical messages between SCNIs should continue without interruption.

Fault Assumptions

The fault model defines three types of fault related to message correctness within the FCR that the system can tolerate:

- Lost message: A message is classified as lost if it fails to reach the intended receiver due to a link or router failure during transmission.
- Delayed message: A message is classified as delayed if it reaches the destination NI with a significant delay due to transmission failures or delays in the link or router. Delays are critical factors in real-time applications, as they can cause the system to miss its deadline.
- Corrupted message: A message is considered corrupted if received, but its content has been altered or corrupted during transmission.

SCNI can mitigate errors by identifying and categorizing these faults, thereby ensuring reliable communication within the FCRs.

6.3.3 Conceptual Model of Extended TTNI

The SCNI is a network interface in the NoC designed to ensure reliable communication between various components in safety-critical systems, including those used in aviation, automotive, and industrial control applications. As depicted in Figure 6.18, the SCNI incorporates two ports: redundant and non-redundant. These ports can be configured during the design phase to facilitate the transmission of critical messages through a redundant path, which involves duplicating the message in the NI. On the other hand, non-critical messages are transmitted through a non-redundant path to conserve resources. The SCNI uses time-triggered communication to provide deterministic behaviour, which is crucial in real-time systems. To maintain communication reliability, the SCNI employs an error detection mechanism that uses a Cyclic Redundancy Check (CRC) to verify the integrity of messages. This method involves adding a checksum to the data block to detect transmission errors. The error detection mechanisms can recognize transmission errors, and the redundancy controller can then choose the correct data at the receiver NI without needing

three copies of the data, as described in the redundancy controller at SCNI in the following section.

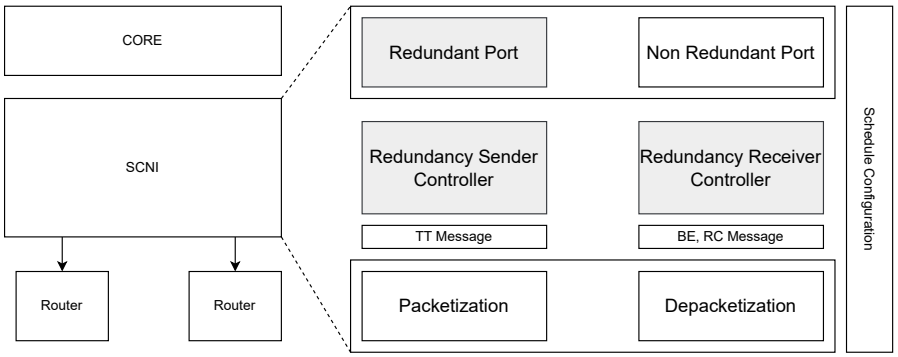


Figure 6.18: Extended TTNI with redundancy controller

Redundancy Controller in SCNI

The redundancy controller is responsible for managing the transmission of both redundant and non-redundant messages in the SCNI. Each output port within the core interface is configured to handle redundant or non-redundant messages. When a redundant port receives an incoming message, the redundancy controller duplicates it and sends it through multiple redundant paths. This ensures that the network remains highly available and resilient despite potential failures. This redundancy also prevents data loss or disruption in case one of the transmission paths fails. On the other hand, when an incoming message is received from a non-redundant port, the redundancy controller uses a precomputed schedule to determine the appropriate path for routing the message. The SCNI provides two paths, and the redundancy controller decides which path the non-redundant message should take. To minimize the risk of message collisions within the NoC, redundant and non-redundant messages from the output port of the core interface are injected at predefined times. By adhering to these predefined injection times, the NoC can effectively reduce the risk of message collisions during message exchange.

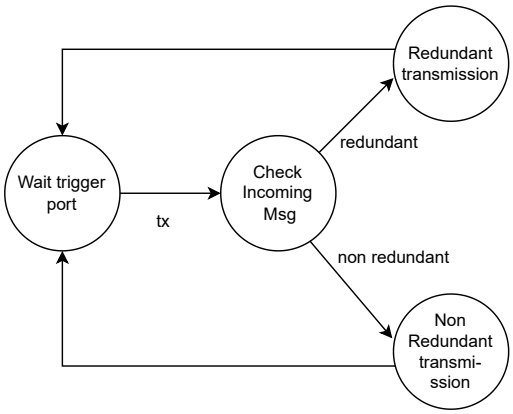


Figure 6.19: State machine of redundancy controller at the sender in SCNI

The redundancy controller is positioned at both the source and the sink of the SCNI. The redundancy controller in the sender is responsible for sending the mes-

sages from the source, and the redundancy controller in the receiver is responsible for receiving the messages from the source. The state machine of the redundancy controller at the sender in the SCNI is illustrated in Figure 6.19. The state machine begins in the "Wait trigger port" state, waiting for a trigger signal from the time-triggered dispatcher to inject the messages. Upon receiving the trigger signal, the state machine transitions to the "Check Incoming Message" state, examining the message type received from the core interface. Suppose the message is from the redundant port. Next, the state machine transitions to the "Redundant Transmission" state, where the redundancy controller duplicates the message and sends it over the redundant path. If the message is from the non-redundant port, the state machine transitions to the "Non-Redundant Transmission" state, where the message is sent using a non-redundant transmission. In both the "Redundant Transmission" and "Non-Redundant Transmission" states, the state machine transitions to the "Wait trigger port" after the message has been successfully transmitted. On the other hand, the redundancy controller at the receiver receives duplicate messages and selects the original message based on the correctness of the data from the two depacketizations in the SCNI, as illustrated in the state machine shown in Figure 6.20.

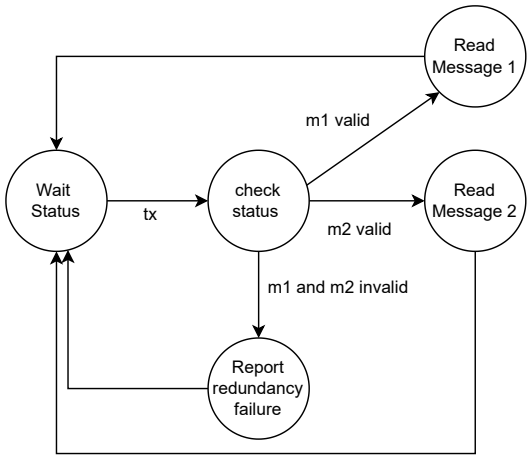


Figure 6.20: State machine of redundancy controller at the receiver in SCNI

The state machine depicted in Figure 6.20 begins in the "Wait Status" state. It remains in that state until it receives a trigger signal from depacketization that restores the message by removing the head and tail flits. Upon receiving the trigger signal, the state machine transitions to the "Check Status" state, examining the validity of messages from two different paths. In this context, a valid message means that the message has reached the destination within a specific time and that no corruption has occurred during transmission. If both messages from the first and second paths are valid, the state machine transitions to the "Read Message 1" state, where it reads the message from the first depacketization and returns to the "Wait Status" state to wait for the next trigger signal. If the message from the first path is valid while the second path is invalid (corrupted or did not reach the destination within the specified time), the state machine still transitions to "Read Message 1" and returns to the "Wait Status" state. However, suppose the message from the first path is invalid while the second message is valid. In that case, the state machine transitions to the "Read Message 2" state and receives the second message while

discarding the corrupted message from the first path. Suppose both messages from the first and second paths are invalid. In that case, the state machine transitions to the "Report Redundancy Failure" state. In this state, the state machine reports a fault in both redundant paths and returns to the "Wait Status" state to wait for upcoming messages.

Chapter 7

Results and Discussion

This chapter presents a comprehensive overview of the experimental setup used to assess the performance, energy efficiency, and fault tolerance techniques of the ATTNoC architecture. To evaluate the effectiveness of the ATTNoC architecture, multiple factors were considered. These factors included end-to-end latency, jitter, the number of uncorrupted messages under fault injection, and energy consumption. A comparison was conducted between the energy consumption of the ATTNoC architecture and the baseline LISNoC architecture without energy-saving techniques. Additionally, faults causing delays, corruption, and message dropping in the NoC components, including NIs, links, and routers, were deliberately introduced into the system to evaluate the architecture's ability to handle faults during runtime. The analysis of the ATTNoC architecture offers valuable insights into its potential to improve the reliability and performance of NoCs. The results obtained from these experiments serve as guidance for the future development and optimization of NoC designs.

7.1 Experiment Goal

The experiment aims to evaluate the performance of the ATTNoC architecture in improving the reliability and energy efficiency of NoCs, which play a crucial role in modern computing systems. The experiments assess the impact of time-triggered frequency scaling (TTFS) in ATTNoC on energy consumption. Moreover, system reliability is evaluated by introducing transient and permanent faults during runtime, considering fault tolerance techniques such as redundancy mechanisms and adaptability. To achieve these goals, tests and measurements are conducted on the ATTNoC, focusing on parameters such as latency, jitter, packet size variation, and energy consumption. These parameters provide insights into the system's performance. Finally, the experimental results are analyzed to assess performance, energy efficiency, and the ability to tolerate faults. These findings have significant implications for designing more efficient and reliable NoCs.

7.2 Field Programmable Gate Array (FPGA)-based Prototypes

The ATTNoC architecture is expected to use application-specific integrated circuit (ASIC) technology in the future, allowing it to be implemented on silicon dies using lithographic masks. ASICs are typically much more power-efficient than FPGAs, and they can also be much faster. However, developing ASICs is expensive and risky, as design errors cannot be easily corrected once silicon production has started. Therefore, the current focus is on developing and testing the ATTNoC architecture using FPGAs, which are cost-effective and do not require a specialized infrastructure. FPGAs are based on look-up tables (LUTs), which provide flexibility and make modifying designs and test series easier. FPGA vendors offer complete development kits, including professional vendor support and access to a user community. For instance, a given circuit implemented in LUTs of an FPGA may require up to 35 times more area and can be between 3-5 times slower than an ASIC implementation. Furthermore, FPGAs consume about 14 times more dynamic power than an equivalent ASIC on average [KR07]. The ATTNoC architecture presented in the system model was implemented on a Xilinx Zynq-MPSoC ZCU102 evaluation board to evaluate the functional behaviour of the ATTNoC architecture and the resource usage on the FPGA.

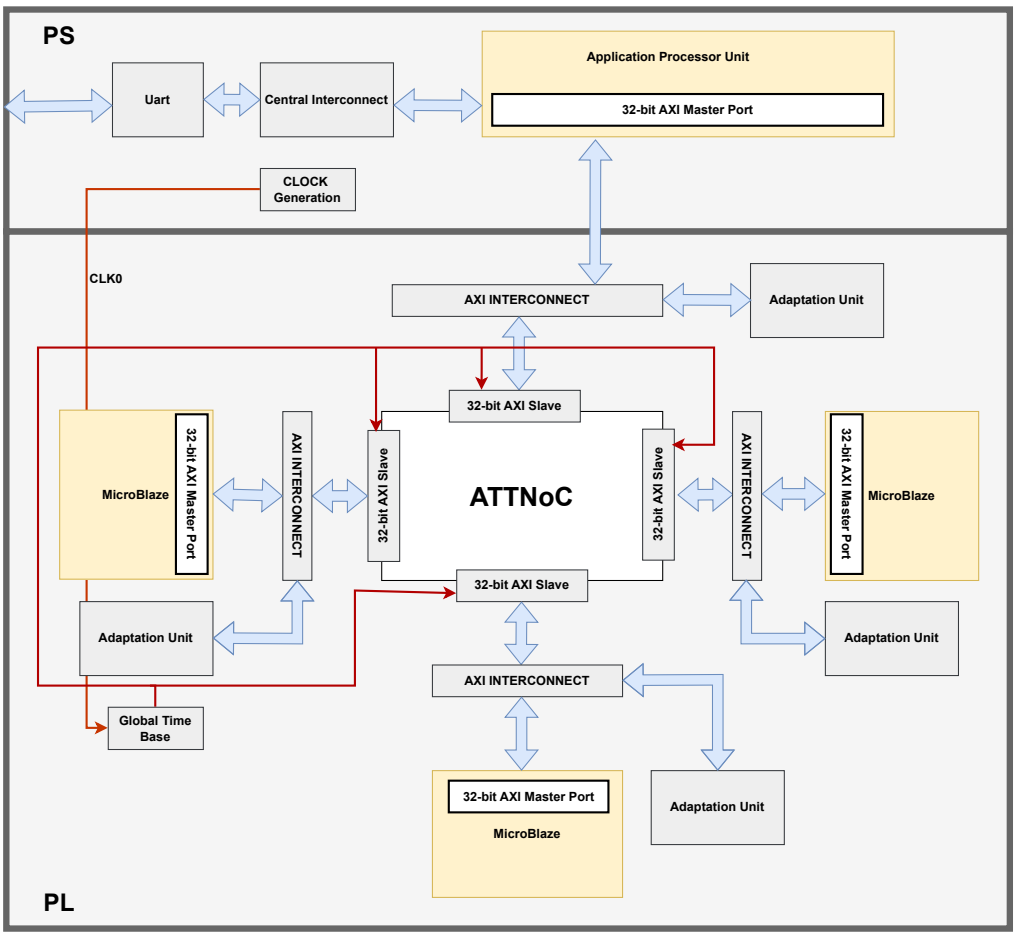


Figure 7.1: Overall system. A simple view of the ATTNoC architecture on the Xilinx Zynq-MPSoC ZCU102 (FPGA)

The ZCU102 is a Zynq UltraScale+ MPSoC Evaluation Kit that integrates a robust processing system (PS) and user-programmable logic (PL), commonly referred to as FPGA, within a single device. The PL contains the ATTNoC with 3x3 routers, adaptation units, GTB, and soft processors. Figure 7.1 illustrates that one processing element was located in the PS, which contained a quad-core Arm®Cortex®-A53 processor. On the other hand, the other three processing elements were deployed in the programmable logic, and each processing element had a single processing core that was realized as a soft-core MicroBlaze processor. The LISNoC was also used as a basis for implementing the ATTNoC.

Hardware	LUT	% LUT used	Slice Registers	% Slice Registers used
Softcore (MicroBlaze x3)	12036	4.39	9849	1.80
LISNoC router (3x3)	49320	17.99	32625	5.95
TTNI (Without Redundancy) x2	25670	9.37	21258	3.87
TTNI (With Redundancy) x2	25914	9.45	21584	3.93
Adaptation Units x4	1812	0.66	3608	0.66
GTB	2	0.00	57	0.01

Table 7.1: Resource usage of the ATTNoC

Table 7.1 outlines the resource usage for various hardware components within the ATTNoC architecture. The resource usage for each component is measured in terms of look-up tables (LUTs) and slice registers. The soft cores (MicroBlaze x3) use fewer resources, specifically 12036 LUTs and 9849 slice registers. This is because they are programmable processors implemented in the FPGA, with limited hardware resources compared to dedicated components like the ATTNoC. However, the LISNoC router (3x3) demands more resources, requiring 49320 LUTs and 32625 slice registers. Routers handle routing and data packet switching, necessitating a larger resource allocation to manage the significant data traffic and buffering requirements. The specific router configuration impacts resource usage, including the number and size of input/output ports and the number of virtual channels supported. The TTNI, serving as the communication interface between routers and other processing elements, exhibits higher resource usage than the routers. The non-redundant version of the two TTNIs uses 25670 LUTs and 21258 slice registers. In contrast, the redundant version requires slightly more resources, with 25914 LUTs and 21584 slice registers for two TTNIs with redundancy mechanisms. Redundancy in the TTNI enables message transfer through two channels, providing fault tolerance in case one channel fails. The four adaptation units, responsible for schedule switching, demonstrate relatively low resource usage compared to other components, using 1092 LUTs and 3280 slice registers for all four units combined. Lastly, the GTB (global time base), responsible for global time synchronization, exhibits the

lowest resource consumption, using only 2 LUTs and 57 slice registers.

7.3 Performance Analysis of ATTNoC

A performance analysis was conducted to assess the effectiveness of the time-triggered features in ATTNoC. The study involved measuring the latency of the baseline event-triggered LISNoC and comparing it to the extended version of LISNoC with time-triggered features, referred to as ATTNoC. Through this analysis, the impact of time-triggered features on the latency performance of the system was determined. The following section details the experimental setup used to evaluate the performance of LISNoC and ATTNoC, and presents the results obtained [NOO23].

7.3.1 Experimental Setup

The experimental setup used in this study involved adding a time-stamp module to each TTNI of the two NoCs, namely LISNoC and ATTNoC. The time-stamp module was added to both the source and sink of the NI to record the injection and arrival time of messages at the NI sender and receiver. The TTNI is a hardware interface between the processing element and the NoC. The added time stamp records the value of the global time base (GTB), a time reference used to synchronize the distributed TTNIs in the ATTNoC. To measure the latency of messages in the ATTNoC and compare it to that of the LISNoC, multiple messages with different packet sizes were injected into the NoC via the source NI, and its injection time was recorded. The time stamp was then added to the message packet. Upon reaching its destination NI, the arrival time was recorded, allowing for latency calculation by subtracting the injection time from the arrival time. This approach effectively measured and compared the latency and jitters of the LISNoC and ATTNoC. In the experiment, messages were sent through the NoCs to assess the baseline LISNoC and ATTNoC performance, as depicted in Figure 7.2. Two messages were intentionally designed to share a resource to simulate congestion and evaluate the NoCs' performance under such conditions. This allowed observation of how baseline LISNoC and ATTNoC handled congestion scenarios and assessed their effectiveness in managing resource sharing and maintaining performance. The experiment involved transmitting messages with different packet sizes from 2 to 16 between processing elements within the ATTNoC and LISNoC. Both NoCs were operated at a frequency of 200 MHz, and the experiment was repeated multiple times to ensure statistical significance. The recorded latencies were stored in a log file to determine the average, worst-case latency, and jitters of the two NoCs. It is important to note that the time-stamp is added in the TTNI module and records the time between transmitting messages between two TTNIs. Thus, any delay caused by the processing elements and the time in which messages are stored in the core interface of the TTNI before their injection into the NoC is not considered in the latency and jitter. This limitation should be taken into account when interpreting the experiment's results.

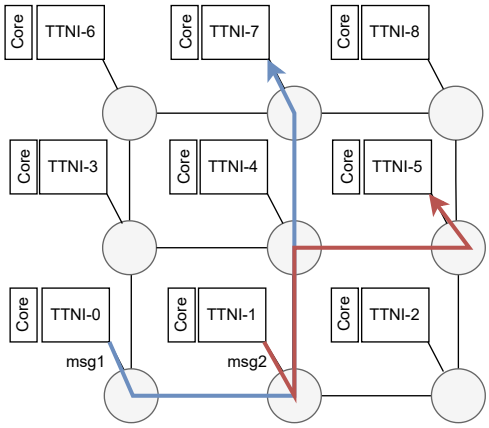


Figure 7.2: 3x3 LISNoC and ATTNoC architecture

7.3.2 Results and Discussion

To evaluate the performance of the LISNoC and the ATTNoC, two essential performance metrics are considered: latency and jitter. The latency of each message was measured and recorded in Tables 7.2 and 7.3. The recorded latency values were then used to calculate the jitter for both the LISNoC and the ATTNoC using equation 7.1.

$$Jitter = Latency_{\max} - Latency_{\min} \tag{7.1}$$

Latency refers to the delay between transmitting a message from a source NI to a destination NI within the NoC. It represents the time difference between when the message is injected into the NoC and when it is received at its intended destination. On the other hand, jitter measures the variation or fluctuation in the latency experienced by messages travelling through the NoC. It is calculated as the difference between the maximum and minimum latency values for a given communication path.

Tables 7.2 and 7.3 display the average and worst-case latency, along with the minimum latency for messages "msg1" and "msg2", in both the baseline LISNoC and the ATTNoC. These tables compare the performance variations between the two networks, highlighting their distinctive characteristics.

Packet Size	LISNoC Avg Latency (ns)	LISNoC Max Latency (ns)	LISNoC Min Latency (ns)	ATTNoC Avg Latency (ns)	ATTNoC Max Latency (ns)	ATTNoC Min Latency (ns)
2	785	2367	317	550	705	317
4	1850	2589	446	1100	1461	446
8	3600	3982	546	1870	2179	546
16	10605	13958	1912	4210	4602	1912

Table 7.2: Packet size vs latency of message-1

Packet Size	LISNoC Avg Latency (ns)	LISNoC Max Latency	LISNoC Min Latency	ATTNoC Avg Latency (ns)	ATTNoC Max Latency (ns)	ATTNoC Min Latency
2	180	285	115	125	262	115
4	340	395	198	249	370	198
8	840	1280	380	450	560	380
16	2100	3252	760	875	1035	760

Table 7.3: Packet size vs latency of message-2

Figures 7.3 and 7.4 present a comparison of average and worst-case latency versus packet size between two different NoC architectures: ATTNoC and baseline LISNoC. The analysis focuses on two messages with different packet sizes exchanged between four cores: Core 0, connected to NI 0, which sends the message "msg1" to core 7, connected to NI 7, and core 2, connected to NI 2, which sends the message "msg2" to core 5, connected to NI 5. In this context, both messages traverse the same link that connects two routers, specifically R2 and R4. This scenario leads to the potential for congestion within the network as multiple messages contend for the same network resources. The x-axis of the figures represents the burst length, indicating the number of flits transmitted, while the y-axis represents latency measured in nanoseconds. The blue and red lines in the figures illustrate the worst-case and average latency of the LISNoC, while the green and violet lines represent the average and worst-case latency of the ATTNoC.

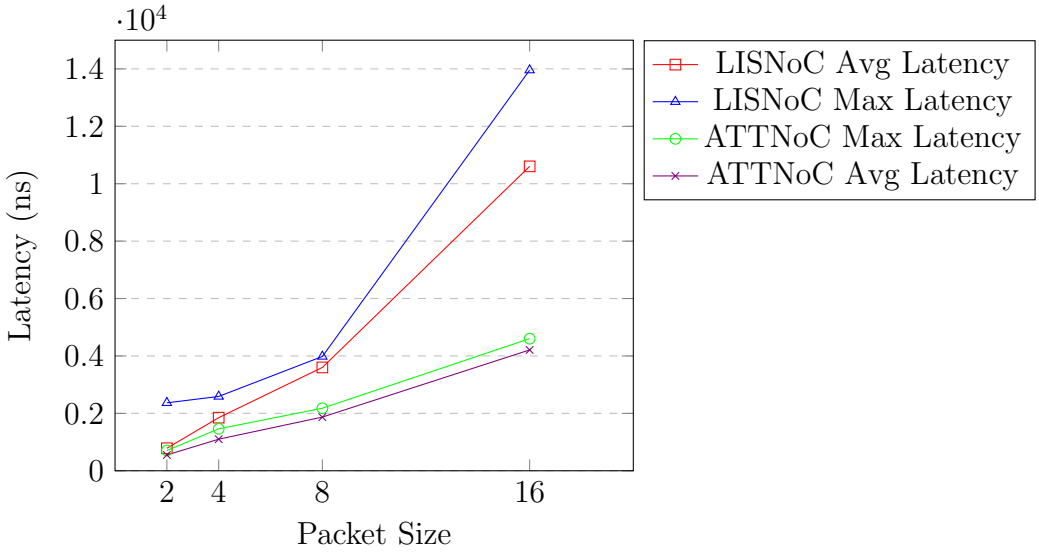


Figure 7.3: Packet size vs latency of message-1

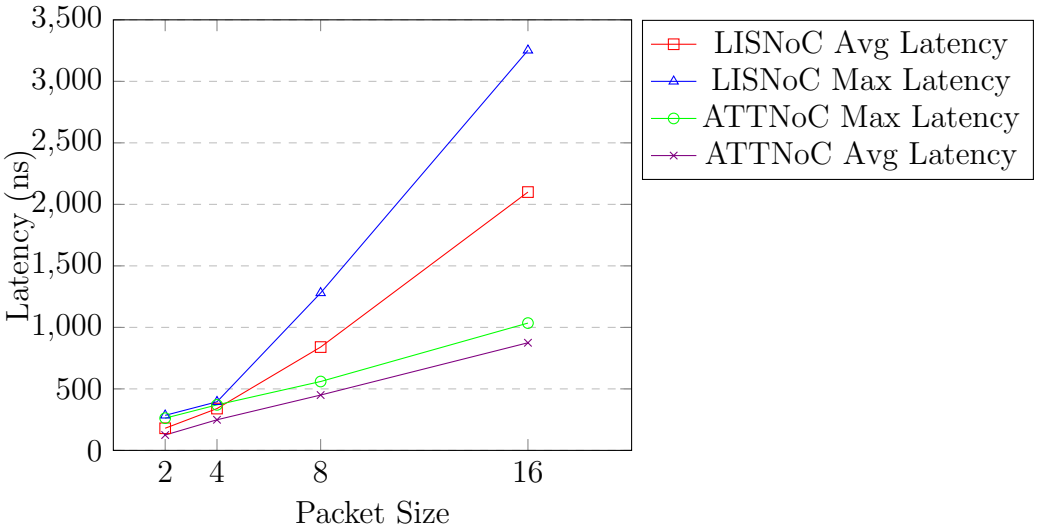


Figure 7.4: Packet size vs latency of message-2

Figure 7.3 and 7.4 illustrate the latency versus packet size for LISNoC and ATTNoC with different packet sizes. From these figures, the ATTNoC demonstrates lower latency than the LISNoC when operating at a frequency of 200 MHz and using a flit size of 32 bits for both NoC architectures. The average latency graphs, plotted for each message and various packet sizes, clearly illustrate significant differences between the two NoCs. The latency curve for the ATTNoC exhibits nearly linear behaviour for both messages ("msg1" and "msg2"). In contrast, the latency for the LISNoC shows a sudden increase as the packet size escalates from 2 to 16. These findings indicate that the ATTNoC design is more efficient and handles inter-core communication within FPGA designs, providing more consistent and predictable latency behaviour and improved overall system performance. On the other hand, the LISNoC introduces significant delay due to the shared link used by both messages during transmission from two different cores. This sharing of resources leads to congestion and negatively impacts the latency performance of the LISNoC. However, for the best-case scenario (minimum latency) for both the LISNoC and the ATTNoC, they have almost the same latency as depicted in Table 7.2 and 7.3. It is important to note that the recorded latency measurements do not account for the delay introduced during the exchange of messages between the core and NI, as well as the delay experienced when messages wait for injection in the core interface in the case of the ATTNoC. These additional delays are not taken into consideration in the latency measurements.

Figures 7.5 and 7.6 present a comparative analysis of the jitters observed in the ATTNoC and the baseline LISNoC for messages "msg1" and "msg2". These messages are injected into the NoCs and share the same link, resulting in network congestion, as depicted in Figure 7.2. The x-axis of the figures represents the packet size, determined by the number of flits transmitted, while the y-axis represents the measured jitters in nanoseconds (ns). The blue line in the plots represents the jitter observed in the ATTNoC, and the red line corresponds to the jitter in the LISNoC. By examining these figures, we gain insights into the comparative jitter performance of the two NoC architectures. The plotted data illustrates the variations in message arrival times and their impact on system timing. Additionally,

the congestion caused by the shared link between the two messages further influences the jitter characteristics.

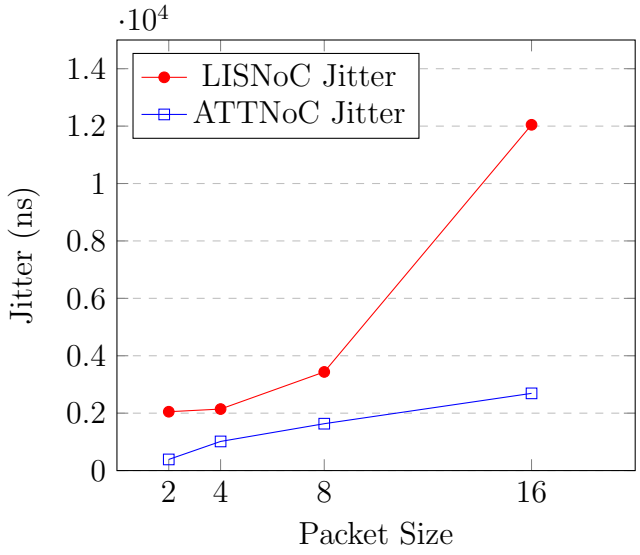


Figure 7.5: Packet size vs Jitter of message 1

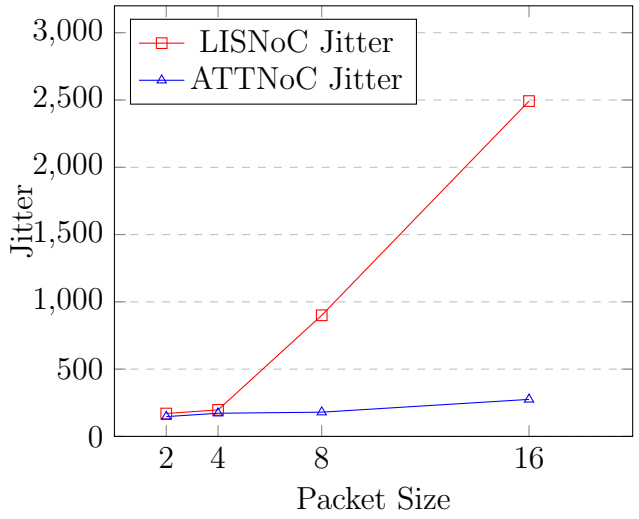


Figure 7.6: Packet size vs jitter of message 2

Figure 7.5 and 7.6 compare the average jitter for different packet sizes between the ATTNoC and the baseline LISNoC architectures [TUM]. Specifically, Figure 7.6 compares the jitter for "msg2," revealing a consistent gradient relationship between jitter and packet size in the ATTNoC. However, in the case of the LISNoC without time-triggered communication, the jitter experiences a rapid increase as the burst length expands from 2 to 16 in both messages. Similarly, Figure 7.5 compares the jitters versus the packet size for "msg1," demonstrating a low jitter of the ATTNoC over the LISNoC for packet sizes 2, 4, 8, and 16. The results show that the ATTNoC offers improved jitter compared to the LISNoC and is better suited for applications where consistent and minimal variation in message arrival times is crucial, such as real-time applications. On the contrary, the LISNoC exhibits a significant increase

in jitter with larger packet sizes, making it less desirable in situations that require precise timing and low-jitter communication.

7.4 TTFS Energy Efficiency Scenarios in ATTNoC

An energy efficiency estimation analysis was conducted to evaluate the effectiveness of using time-triggered frequency scaling (TTFS), which is a power-saving technique used in the ATTNoC architecture. The analysis involved estimating the power consumption of the routers in the network using the open-source tool ORION 3.0 [KLN15]. This tool can estimate power consumption at various levels of abstraction, including microarchitecture, implementation, and operating parameters, as well as multiple router register transfer levels [KLN15], [Nam+21]. By estimating the power consumption of the routers, it was possible to evaluate the energy efficiency of the ATTNoC architecture. The power estimation analysis enabled a comparison of the energy efficiency of the ATTNoC architecture with and without TTFS. The analysis results can be used to optimize the frequency scaling of the routers to achieve better energy efficiency.

7.4.1 Experimental Setup

The ATTNoC architecture incorporates a power-saving technique known as TTFS, allowing NoC routers to adjust their operating frequency based on a predefined schedule. The primary objective of this technique is to reduce power consumption in multi-core chips. TTFS involves three methods: global, cluster, and router-based approaches, all of which are detailed in Section 6.2.2. To assess the power-saving methods employed by the ATTNoC at the router level, these techniques are applied in three distinct cases:

- Case 1: ATTNoC without power-saving techniques. In this case (see Figure 7.7), the ATTNoC operates without any frequency scaling, meaning all routers consistently maintain a fixed frequency. The power consumption of the network routers is evaluated when power-saving techniques are not used. Since the frequency of all routers remains fixed, the three approaches (global, cluster, and router-based) consume the same power.

- Case 2: ATTNoC with clock-gating. In this case (see Figure 7.8), the frequency used in the routers is predefined in advance. When the routers are scheduled to be active, the time-triggered dispatcher sets a specific frequency for them to operate. However, when the routers are scheduled to be idle, the time-triggered dispatcher triggers the clock-gating operation, allowing the routers to be clock-gated and conserve energy.

- Case 3: ATTNoC incorporates TTFS and clock-gating. In this case (see Figure 7.9), the frequencies of the routers are adjusted over time according to a predefined schedule. Multiple clock domains are used to achieve this frequency scaling at the router level. Conversely, the clock of idle routers is gated according to the schedule to save energy.

By examining these three cases, we can evaluate the effectiveness of different power-saving methods used by the ATTNoC at the router level. This analysis provides valuable insights into the energy efficiency and power optimization capabilities of the ATTNoC architecture.

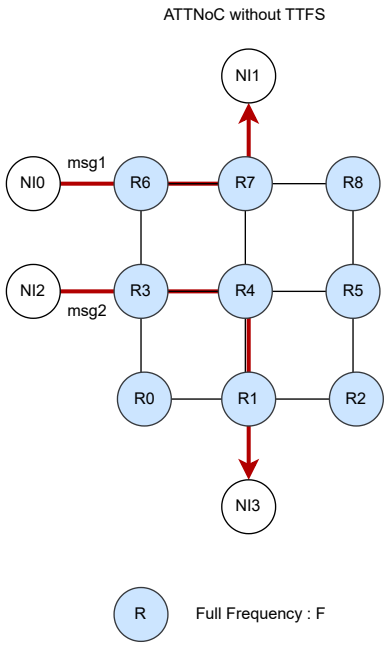


Figure 7.7: Example of ATTNoC without frequency scaling in the router

Table 7.4 presents the experimental setup used to evaluate the energy efficiency of the ATTNoC when no power-saving techniques (case-1) are employed. In this case, various messages were exchanged between cores, each message with different deadlines.

Msgs ID	Message Path	Active Routers	Idle Routers	Operating Frequency of Active Routers	Deadlines (μs)	Energy Active (μJ)	Energy Idle (μJ)
1	R6, R7	9	0	full	100	22.32	0
2	R3, R4, R1	9	0	full	120	26.78	0
3	R3, R4, R5, R2	9	0	full	520	116.71	0
4	R7, R4, R1	9	0	full	120	26.78	0
5	R8, R5, R2	9	0	full	120	26.78	0
No Msgs	-	9	0	full	560	125.00	0

Table 7.4: Case-1: ATTNoC without frequency scaling

As seen in Table 7.4, five messages are transmitted within a fixed period of 1.54 ms in the described setup. Each message follows a specific path from the source to the destination core, passing through different routers. The active routers operate at a frequency defined by the schedule to facilitate message forwarding between the source and destination cores. However, in case 1, no power-saving techniques are applied, resulting in all routers operating at full frequency even when no messages are being transmitted. This absence of power-saving measures leads to increased power consumption within the ATTNoC.

The second case involves an ATTNoC with clock-gating, where active routers operate at full frequency, and idle routers are clock-gated according to a predefined schedule (see Figure 7.8). The figure below illustrates the global, cluster, and router-based approaches to case 2.

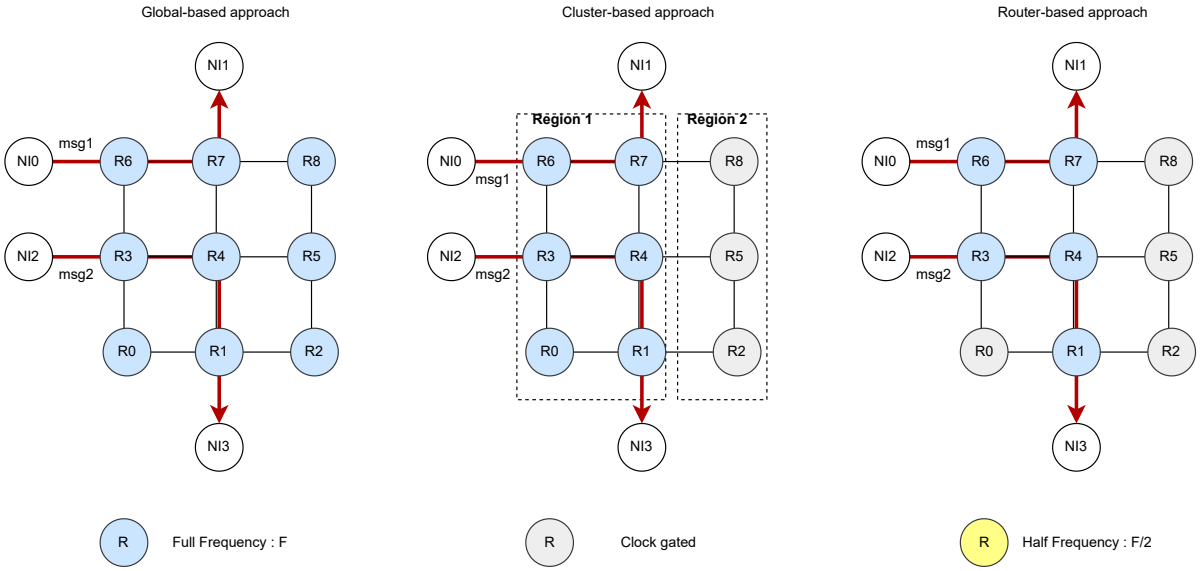


Figure 7.8: Example of ATTNoC with clock-gating

In the second case, the power consumption of the ATTNoC routers is evaluated using global, cluster, and router-based approaches, as shown in Figure 7.8. These techniques optimize the system's energy efficiency by clock-gating the idle routers. The global approach is used to enhance the power consumption of the ATTNoC architecture, and the experimental setup for this approach is presented in Table 7.5. In the global approach, all routers in the network operate at the same clock frequency. When specific routers are scheduled to transmit data, all routers in the NoC are active and operating with a full frequency mode. However, the frequency of all routers is clock-gated when all routers in the network are idle. This approach aims to reduce the power consumption of NoC routers by clock-gating the idle routers during the common idle time of all routers. By doing so, the total power consumption of the network can be reduced.

Msgs ID	Message Path	Active Routers	Idle Routers	Operating Frequency of Active Routers	Deadlines (μs)	Energy Active (μJ)	Energy Idle (μJ)
1	R6, R7	9	0	full	100	22.32	0
2	R3, R4, R1	9	0	full	120	26.78	0
3	R3, R4, R5, R2	9	0	full	520	116.07	0
4	R7, R4, R1	9	0	full	120	26.78	0
5	R8, R5, R2	9	0	full	120	26.78	0
No Msgs	-	0	9	full	560	0	2.97

Table 7.5: Case-2: ATTNoC with clock-gating using global approach

Table 7.6 shows the setup used to evaluate the power consumption of the ATTNoC using the cluster approach. This approach divides the network into several clusters or regions. The frequency of all routers in the cluster is the same during active times and idle times. When a specific router in the region is scheduled to transmit data, all routers in that region are synchronized to operate at the same frequency according to a schedule. It is assumed that the operating frequency is in full frequency mode, and the frequency of all routers in the cluster is clock-gated only during the common idle time of the routers in the cluster.

Msgs ID	Message Path	Active Routers	Idle Routers	Operating Frequency of Active Routers	Deadlines (μs)	Energy Active (μJ)	Energy Idle (μJ)
1	R6, R7	6	3	full	100	14.88	0.17
2	R3, R4, R1	6	3	full	120	17.85	0.21
3	R3, R4, R5, R2	9	0	full	520	116.07	0
4	R7, R4, R1	6	3	full	120	17.85	0.21
5	R8, R5, R2	3	6	full	120	8.92	0.42
No Msgs	-	0	9	full	560	0	2.97

Table 7.6: Case-2: ATTNoC with clock-gating using cluster approach

The configuration used to evaluate the power consumption of ATTNoC using the router-based approach is presented in Table 7.7. In this approach, the operating frequency of each router is determined by its schedule and synchronized with the scheduled messages for transmission. Unlike the global and cluster approaches, the frequency of each router is independent of the frequency of the region or the entire NoC. With this approach, the operating frequency of active routers is specifically set when scheduled to transmit messages. This ensures that routers activate only during data transmission, while inactive routers are clock-gated according to their schedules, effectively reducing power consumption.

Msgs ID	Message Path	Active Routers	Idle Routers	Operating Frequency of Active Routers	Deadlines (μs)	Energy Active (μJ)	Energy Idle (μJ)
1	R6, R7	2	7	full	100	4.96	0.41
2	R3, R4, R1	3	6	full	120	8.92	0.42
3	R3, R4, R5, R2	4	5	full	520	51.58	1.53
4	R7, R4, R1	3	6	full	120	8.92	0.42
5	R8, R5, R2	3	6	full	120	8.92	0.42
No Msgs	-	0	9	full	560	0	2.973

Table 7.7: Case-2: ATTNoC with clock-gating using router-based approach

The third case (see Figure 7.9) presents an example of an ATTNoC with multiple clock domains, including full-frequency and half-frequency modes. The dispatcher within the TTNI plays a crucial role in triggering frequency scaling and clock-gating operations within the ATTNoC. It efficiently schedules the timing for adjusting the frequency of routers and assigns specific frequencies to be used. As a result, the active router can adjust its frequency mode according to the predefined schedule. When messages requiring high bandwidth are scheduled, the NoC routers operate in high-frequency mode. Conversely, the half-frequency mode is used for scheduled communications with lower bandwidth requirements. The frequency controller in the TTFS module configures the router frequency based on the configuration information received from the time-triggered dispatcher of the TTNI. This approach allows for the optimization of power consumption in routers transmitting low-criticality messages by incorporating the half-frequency mode. Additionally, inactive routers are often subjected to clock-gating to conserve power. Notably, this frequency adjustment in the routers is solely based on a time-triggered schedule, essential for preserving the deterministic communication within the ATTNoC.

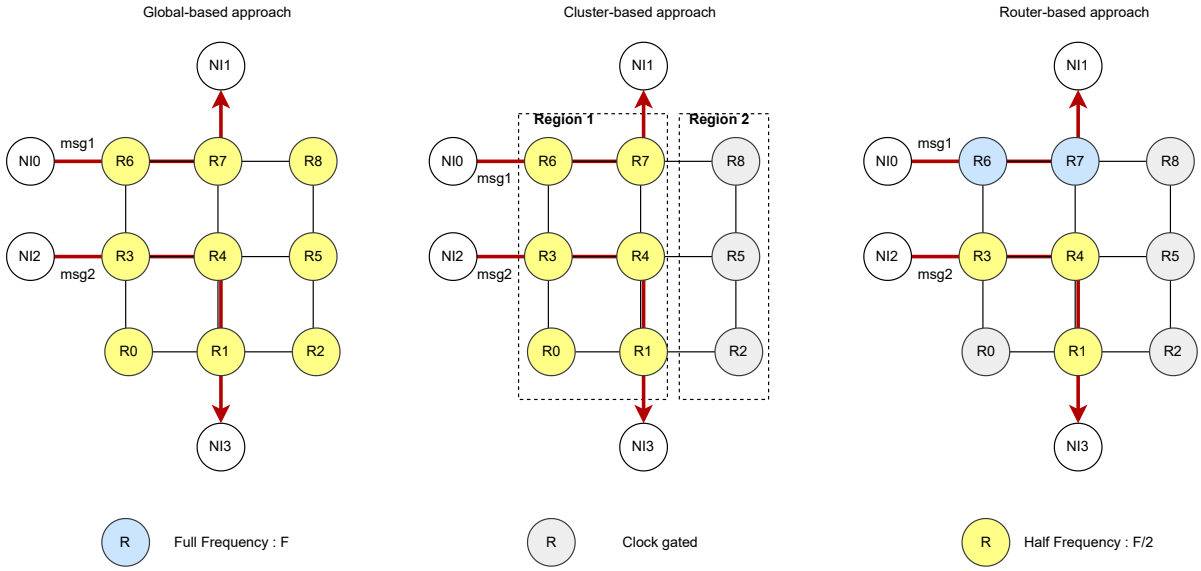


Figure 7.9: Example of ATTNoC with frequency scaling with multiple clock domain

In the third case, the active routers have two clock ranges: full-frequency and half-frequency modes. In full-frequency mode, they operate at 200 MHz; in half-frequency mode, they operate at 100 MHz. The experiment is similar to "Case 2," but with the addition of using multiple clock ranges to adjust the frequency of the routers based on a predetermined schedule. The main goal of this experiment is to assess the power consumption of the ATTNoC while using TTFS with different clock domains, specifically focusing on the global, cluster, and router-based approaches. Tables 7.8, 7.9, and 7.10 present the configurations used for the power evaluation of the ATTNoC under consideration when TTFS is applied.

Msgs ID	Message Path	Active Routers	Idle Routers	Operating Frequency of Active Routers	Deadlines (μs)	Energy Active (μJ)	Energy Idle (μJ)
1	R6, R7	9	0	half	100	22.04	0
2	R3, R4, R1	9	0	half	120	26.45	0
3	R3, R4, R5, R2	9	0	half	520	114.62	0
4	R7, R4, R1	9	0	full	120	26.78	0
5	R8, R5, R2	9	0	full	120	26.78	0
No Msgs	-	0	9	full	560	0	2.97

Table 7.8: Case-3: ATTNoC with frequency scaling using global approach

Msgs ID	Message Path	Active Routers	Idle Routers	Operating Frequency of Active Routers	Deadlines (μs)	Energy Active (μJ)	Energy Idle (μJ)
1	R6, R7	6	3	half	100	14.69	0.17
2	R3, R4, R1	6	3	half	120	17.63	0.21
3	R3, R4, R5, R2	9	0	half	520	114.62	0
4	R7, R4, R1	6	3	full	120	17.85	0.21
5	R8, R5, R2	3	6	full	120	8.92	0.42
No Msgs	-	0	9	full	560	0	2.97

Table 7.9: Case-3: ATTNoC with frequency scaling using cluster approach

Msgs ID	Message Path	Active Routers	Idle Routers	Operating Frequency of Active Routers	Deadlines (μs)	Energy Active (μJ)	Energy Idle (μJ)
1	R6, R7	2	7	half	100	4.89	0.41
2	R3, R4, R1	3	6	half	120	8.81	0.42
3	R3, R4, R5, R2	4	5	half	520	50.94	1.53
4	R7, R4, R1	3	6	full	120	8.92	0.42
5	R8, R5, R2	3	6	full	120	8.92	0.42
No Msgs	-	0	9	full	560	0	2.97

Table 7.10: Case-3: ATTNoC with frequency scaling using router-based approach

7.4.2 Results and Discussion

To establish a baseline measurement, the energy consumption of the ATTNoC is recorded without using power-saving techniques. Table 7.4 shows the baseline energy consumption, which is 343.75 μJ for ATTNoC without power-saving techniques. This measurement serves as a reference for evaluating energy efficiency in subsequent cases where power-saving techniques are considered.

Figure 7.10 illustrates the energy consumption of a 3x3 ATTNoC in case 2, considering the global, cluster, and router-based approaches. Among these approaches, the global approach exhibits the highest energy consumption, measuring 221.72 μJ. The cluster approach follows with an energy consumption of 179.59 μJ, while the router-based approach demonstrates the lowest energy consumption at 89.52 μJ. These results indicate that the router-based approach is the most energy-efficient, as it consumes the least energy. Conversely, the global approach consumes the highest energy, making it the least energy-efficient option.

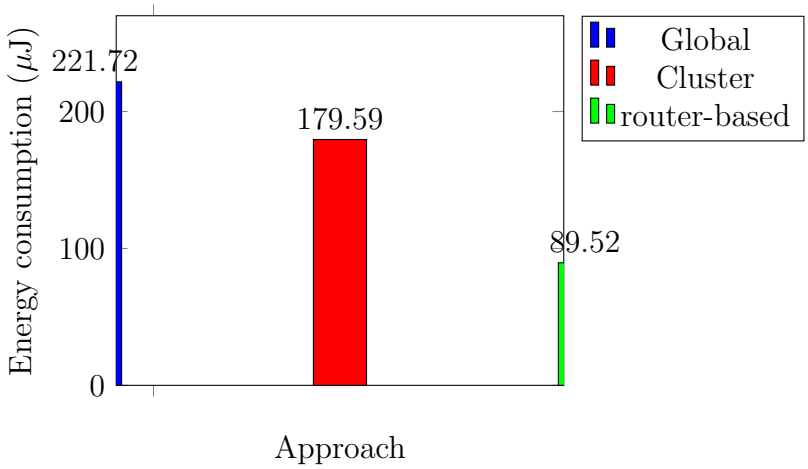


Figure 7.10: Case2: Power consumption of TTFS using different approaches.

The figure provided below compares the power consumption of the 3x3 ATTNoC in case 3, considering three different techniques: Global, cluster-based, and router-based. Figure 7.11 illustrates the energy consumption for each approach. The global approach produced the highest energy consumption among the three techniques, measuring 219.65 μJ. The cluster-based approach exhibited lower energy consumption at 177.73 μJ, while the router-based approach demonstrated the lowest energy consumption at 88.70 μJ.

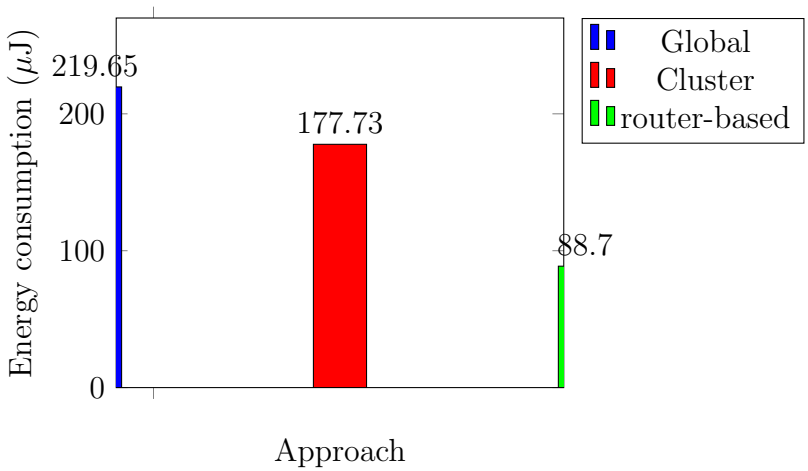


Figure 7.11: Case-3: Power consumption of TTFS using different approaches.

Figure 7.12 summarizes the comparison of power consumption for the ATTNoC architecture using three different power-saving techniques: global, cluster, and router-based approaches. The figure presents three cases: Case 1: Represents the use of ATTNoC without any power-saving techniques. Case 2: Represents the use of ATTNoC with clock-gating techniques. Case 3: Represents the use of ATTNoC with clock-gating and frequency scaling, allowing the NoC routers to adjust their operating frequency according to a predefined schedule.

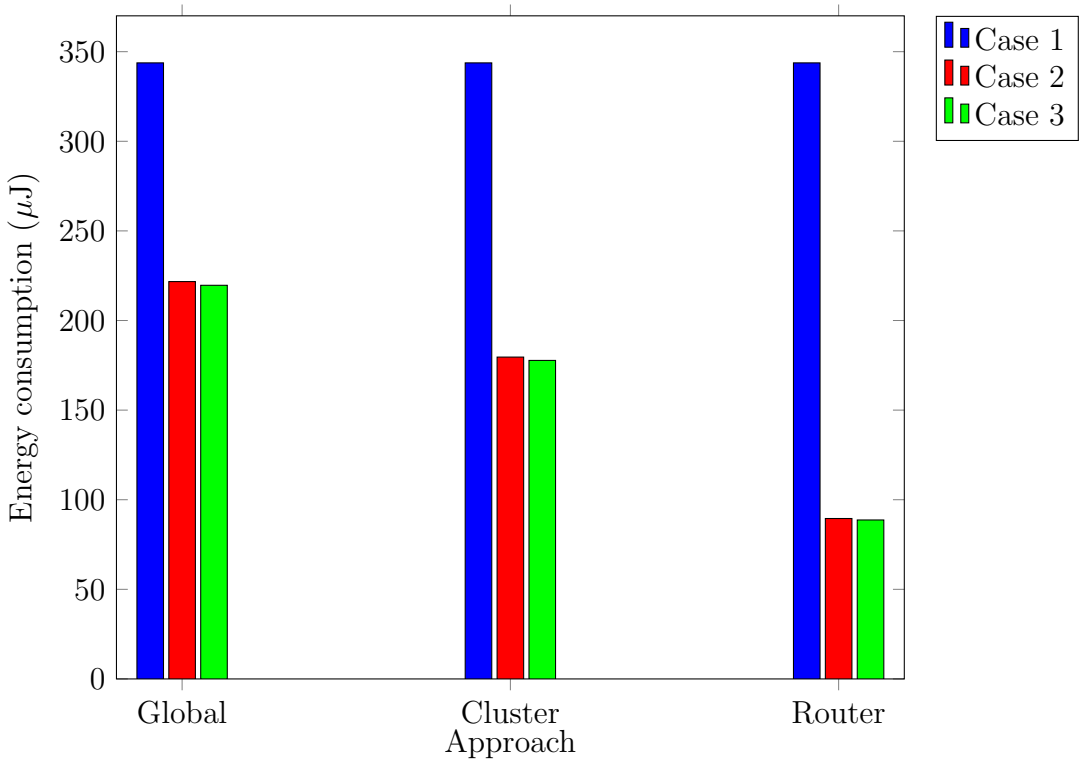


Figure 7.12: Comparison of power consumption of TTFS using different approaches for three cases (1, 2, 3).

Figure 7.12 presents a comprehensive comparison of power consumption in the ATTNoC architecture using different approaches (global, cluster, and router-based) for three cases (1, 2, and 3). In case 1, used as a baseline with no power-saving techniques employed, the energy consumption for all three approaches is 343.75 μJ. In case 2, clock-gating techniques are employed, resulting in optimized energy consumption in the ATTNoC router. Case 2 shows a significant reduction in energy consumption compared to case 1: 221.72 μJ for the global approach, 179.59 μJ for the cluster approach, and the router-based approach consumes the least energy at 89.52 μJ. Case 3 implements frequency scaling with multiple clock domains, enabling routers in the ATTNoC to operate at different frequencies, leading to further optimization of energy consumption. This approach achieves even lower energy consumption levels: 219.65 μJ for the global approach, 177.73 μJ for the cluster approach, with the router-based approach exhibiting the lowest energy consumption at 88.70 μJ. Comparing the energy consumption values across the three cases, case 3, with frequency scaling using multiple clock domains, demonstrates the most efficient energy consumption in the ATTNoC architecture. The figure also illustrates the variations in energy consumption among the global, cluster, and router-based

approaches across the three cases (1, 2, and 3). The router-based approach consistently exhibits the lowest energy consumption among the three when using power-saving techniques, especially in cases 2 and 3. However, it is essential to consider the trade-offs between energy consumption and memory requirements for each approach in the context of the specific application and the available hardware resources. The router-based approach necessitates maintaining individual schedules for each router, while the global and cluster approaches only require schedules for the entire NoC or individual clusters, respectively. While this difference in memory requirements and energy consumption influences the suitability of each approach for different applications, the choice between the three approaches should be made based on a comprehensive evaluation of both energy consumption and memory requirements in the context of the specific application and the available hardware resources. If energy efficiency is the primary concern and memory resources are not a limiting factor, then the router-based approach may be a viable option. However, the global or cluster approach may be more suitable if memory resources are limited.

7.5 Fault Tolerance Techniques in ATTNoC

This study evaluates the fault tolerance techniques used in the ATTNoC architecture, which include seamless redundancy and adaptation features. The adaptation features in the ATTNoC architecture was designed to tolerate permanent faults in the NoC resources, such as links, cores, NIs, and routers. Additionally, seamless redundancy mechanisms were designed to tolerate transient and permanent faults within the NoC components, such as links and routers, during the exchange of critical messages between safety-critical NIs (SCNIs). Two distinct tests were conducted to assess the effectiveness of the fault tolerance techniques used in ATTNoC. The first test adopted a predefined test case approach, intentionally introducing specific faults that caused message delays, corruptions, and losses within the NoC. A designed test bench was used to observe the system's response to these faults, including permanent and transient ones. The primary goal of this test was to quantify the number of uncorrupted packets received by the destination core, thereby indicating the system's reliability under known and controlled fault conditions. 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. This test aimed to assess the capability of the redundancy techniques used in the ATTNoC for tolerating transient faults. The analysis mainly focused on assessing the system's response to known permanent and transient faults in the first test scenario using the test case approach. The second test scenario also examined the system's adaptability to unpredictable fault events in the NoC resources, such as message corruption. By combining the results of these distinct test scenarios, the study aimed to comprehensively understand the inherent fault tolerance capabilities of the ATTNoC architecture.

7.5.1 Experiment Setup Based on Predefined Test Case

This experiment aimed to comprehensively assess the effectiveness of the fault tolerance techniques used in the ATTNoC, focusing on adaptation and seamless re-

dundancy 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. In the subsequent section, we will dive into the functional behaviour of the ATTNoC in the presence of a permanent fault that causes messages to drop. Subsequently, we will provide a detailed description of the experimental setup used to assess the reliability of the ATTNoC using the predefined test case, as presented in Section 7.5.1.

Functional Behaviour of the ATTNoC in the Event of a Permanent Fault

This study illustrated the functional behaviour of the ATTNoC architecture when a permanent fault occurs in the NoC resources, leading to dropped messages. The experiment uses the Vivado Xilinx tool to simulate communication between four cores under various configurations, as illustrated in Figure 7.13. These configurations include scenarios with and without redundancy mechanisms and with and without adaptation features in the ATTNoC. Specifically, a permanent fault was introduced into the link connecting R6 and NI3, resulting in dropped flits passing through the link. The simulation results, presented in Figures 7.14 to 7.17, illustrate the communication between the four NIs as depicted in Figure 7.13.

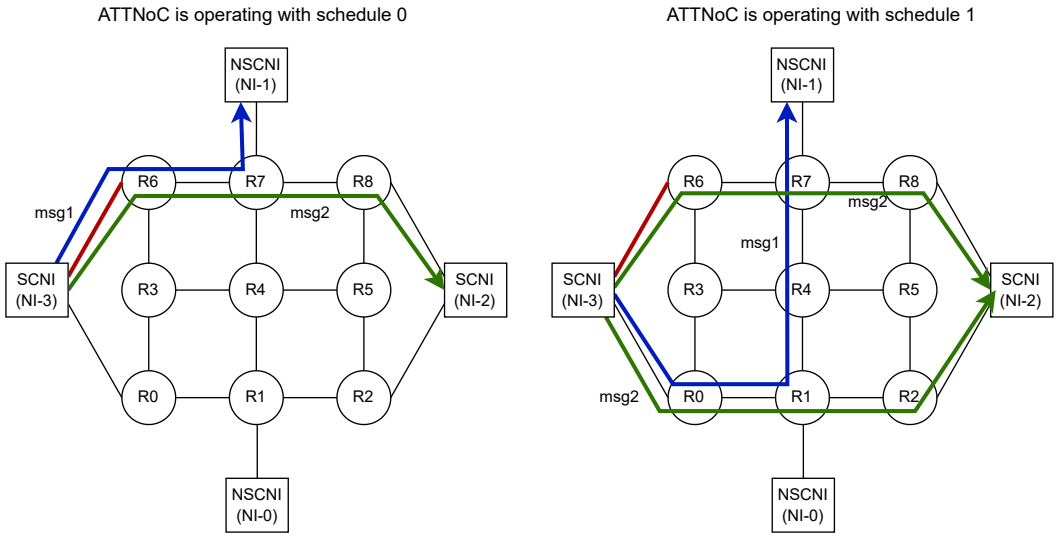


Figure 7.13: Example of an ATTNoC set up with a permanent fault in the link connecting NI3 and R6, which is indicated in red, and which uses multiple schedules

Figure 7.13 shows the experimental setup used to evaluate the communication behaviour of the ATTNoC architecture when a permanent fault is introduced. In this setup, four network interfaces (NIs) exchange two messages, namely message 1 (msg1) and message 2 (msg2). NI3 sends messages to NI1 and NI2 at different injection times. Schedule 0, shown on the left side of the figure, illustrates that NI3 sends the message msg1 to NI1 through the R6-R7 path and the message msg2 to

NI2 through the R6-R7-R8 path. However, if a permanent fault occurs in the link that connects NI3 and router R6, the adaptation features of the ATTNoC architecture enable the NoC to switch schedules depending on the context event. In this case, the ATTNoC switches to schedule 1, as shown on the right side of Figure 7.13, when the link connecting NI3 and router R6 fails. This schedule switching maintains communication within the network by activating the redundancy mechanism in NI3, which allows NI3 to transfer duplicated messages through dual channels. Moreover, by rerouting the message msg1 through a different path (R0-R1-R4-R7), the ATTNoC can isolate the faulty link.

Evaluation of ATTNoC Functional Behaviour Under Permanent Fault Scenarios

The ATTNoC communication was simulated using the Vivado Xilinx tools to evaluate the functional behaviour of the ATTNoC when a permanent fault is introduced in the NoC resources, such as routers, network interfaces (NI), and links. Figure 7.14 illustrates the simulation result of the communication between NI3 and NI2 when the redundancy mechanism is disabled in the ATTNoC. In this scenario, the messages are transmitted through the R6-R7-R8 path to reach NI2.

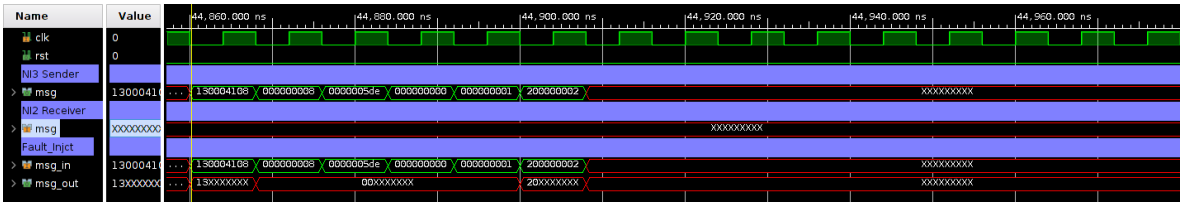


Figure 7.14: Illustration of messages exchanged between NI3 and NI2, without redundancy mechanism, fault is activated

The simulation results depicted in Figure 7.14 demonstrate that NI3 successfully transmits a packet of six flits to NI2. However, despite the successful transmission at NI3, the flits are not received at the intended destination, NI2. This communication failure is attributed to a faulty connection between router R6 and NI3, through which the messages pass. This simulation result highlights the impact of permanent faults in NoC resources such as routers, NIs, links, and cores, which can disrupt communication in the NoC. It emphasises the importance of implementing fault tolerance techniques, such as redundancy mechanisms, to ensure reliable and efficient system operation.

Including redundancy mechanisms in the ATTNoC architecture, as depicted in Figure 7.15, mitigates the impact of permanent and transient faults occurring within the network. In this scenario, the redundancy mechanisms in the TTNI are enabled, which allows the safety-critical network interface (SCNI) to duplicate messages at the sender NI and transmit them through a dual channel. More specifically, the NI3 sender transmits the messages to the destination NI2 through a dual channel using the R6-R7-R8 and R0-R1-R2 paths.

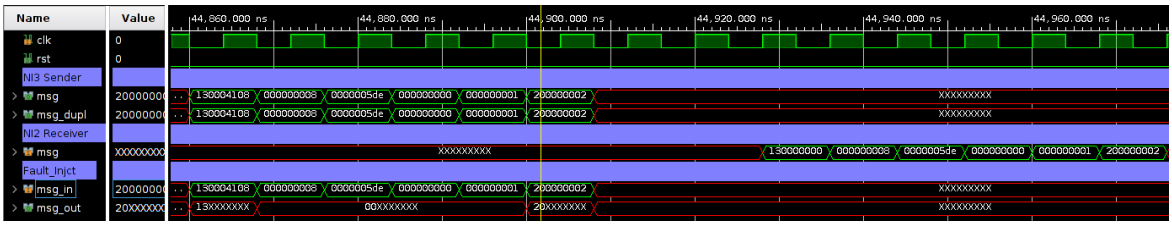


Figure 7.15: Illustration of messages exchanged between NI3 and NI2, with redundancy mechanism, fault is activated

This simulation result demonstrates the effectiveness of redundancy mechanisms in the ATTNoC architecture. The safety-critical network interface (SCNI) duplicates messages at the source NI3 and transmits them to the destination NI2 through two separate paths: R6-R7-R8 and R0-R1-R2. In Figure 7.15, it can be observed that despite the corruption of the first packet caused by the faulty connection between NI3 and router R6, the redundant packet successfully reaches the destination NI2, ensuring the reliable and efficient operation of the system. This outcome emphasizes the importance of incorporating redundancy mechanisms in developing NoCs to enhance their reliability.

This setup focuses on the communication between NI3 and NI1, as depicted in Figure 7.13. The messages are transmitted from NI3 to NI1 through the R6-R7 path. It is important to note that in this specific scenario, the adaptation functions are disabled. The network interface cannot dynamically adapt to changing communication conditions.

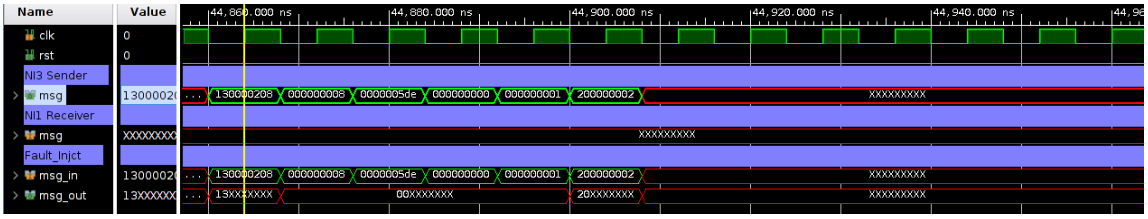


Figure 7.16: Illustration of messages exchanged between NI3 and NI1, without adaptation, fault is activated

As shown in Figure 7.16, the sender NI3 successfully transmits a packet of six flits to destination NI1. However, due to the permanent fault introduced in the link between NI3 and router R6, causing the dropped flit to traverse that link, the messages do not reach destination NI1. This illustrates the potential impact of a NoC component fault on communications within the NoC and underscores the importance of implementing fault tolerance techniques, such as adaptation functions. These techniques enable the network to reroute messages when a persistent fault occurs. By rerouting messages, the network can isolate the faulty resources in the NoC and distribute tasks or messages to other available resources, ensuring the system's continued operation in the presence of a permanent fault.

Figure 7.17 depicts the communication between NI3 and NI1, highlighting the incorporation of adaptation techniques into the ATTNoC architecture. Specifically, the links connecting router R6 with NI3 intentionally introduce a persistent fault. Adaptation techniques in the ATTNoC empower the network to switch schedules and isolate the faulty NoC resources, ensuring the continuous operation of the ATTNoC even in the presence of faults. Figure 7.17 illustrates the outcome of communication that demonstrates the successful support of adaptation techniques within the ATTNoC.

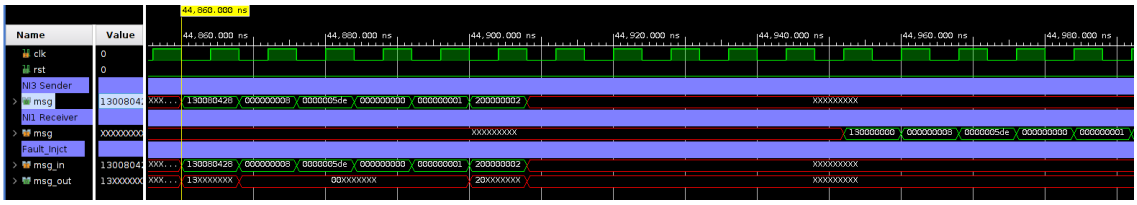


Figure 7.17: Illustration of messages exchanged between NI3 and NI1, with adaptation (flits are rerouted), the fault is activated

The simulation results demonstrated that incorporating adaptation techniques in the NoC's communication enhanced the system's ability to adapt to faulty communication components. In Figure 7.17, when the link between NI3 and router R6 failed to transmit messages, the adaptation techniques played a crucial role by redirecting the data through an alternate path, specifically R0-R1-R4-R7, by switching between schedules. This outcome emphasizes the effectiveness of adaptation techniques in ensuring the reliable operation of the NoC. By dynamically rerouting the data when a permanent fault occurs in the NoC resources, the NoC can overcome disruptions and successfully deliver messages to their intended destinations.

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 7.19. 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 depicted in Figure 7.18 and 7.19 was used to assess the ATTNoC system's functional behaviour under different fault conditions and three fault types (F1, F2, and F3). Each fault scenario can contain multiple faults S = {F1,..., Fn}. The following points describe the fault scenarios used in the experiments.

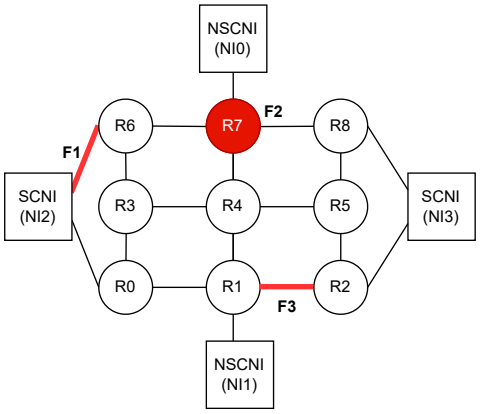
- The fault denoted as F1 represents a permanent fault that occurs at the network link that connects R6 to NI2. This fault leads to message corruption. The fault is activated when the time reaches 600 times the period, equal to 600 * T with T = 488.28 μs. Here, the period refers to the interval used for time-triggered communication, allowing eight packets to be exchanged between the NoC every period, as depicted in Figure 7.19. 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.

Figure 7.18: Fault scenario S= {F1,F2,F3} and communication schedule in the ATTNoC.

Figure 7.19 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

Figure 7.19: Fault scenario S= {F1,F2,F3}. Red in the architecture highlights an error.

To evaluate the effectiveness of fault tolerance mechanisms in 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 network interfaces (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}.

Results and Discussion

Figure 7.20 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 reliability of the system, resulting in a considerable loss of packets. In Case 3, the ATTNoC has redundancy features but no adaptation features. It received a total of 7203 packets, which is higher than the ATTNoC without any 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 help maintain the system's reliability, even in the presence of faults, increasing its robustness and highlighting the importance of including them in ATTNoC design, specifically when designing 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 overheads (as depicted in Table 7.1).

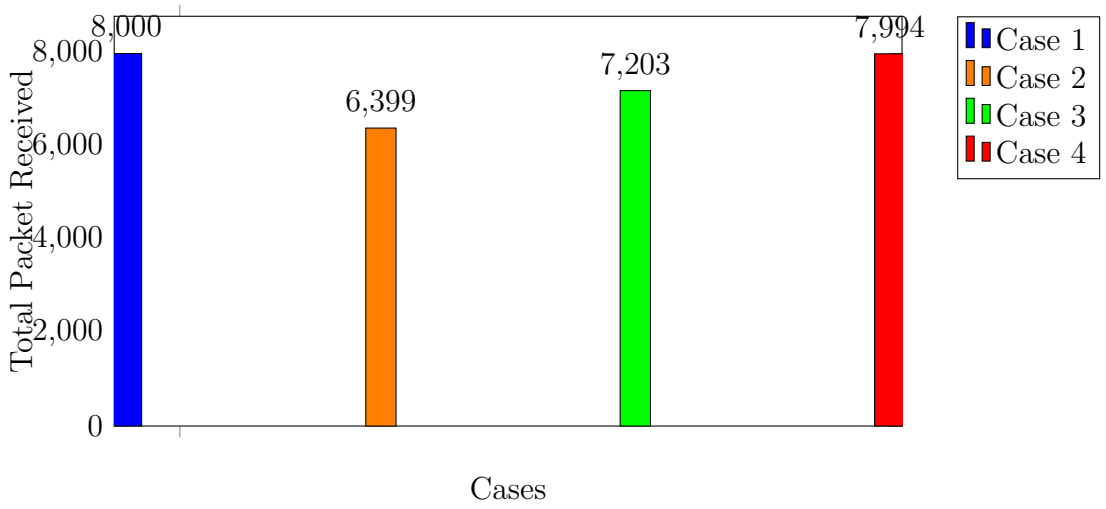


Figure 7.20: Comparison of the total packet received in four cases in ATTNoC communication.

7.5.2 Experiment Setup Based on Randomized Test Case

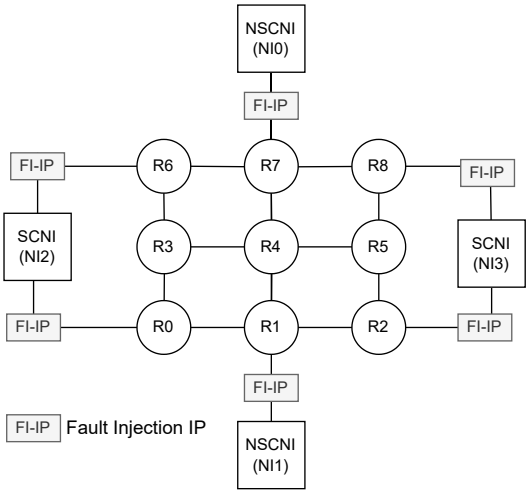
This experiment assessed the seamless redundancy mechanisms employed in ATTNoC when encountering transient faults in the fault containment region, as outlined in section 6.3.2. These mechanisms are specifically designed to tolerate transient and permanent faults that may arise in the routers and links of ATTNoC during critical message exchanges between safety-critical Network Interfaces (SC-NIs). One of the targeted faults was message corruption that can occur in the NoC resources, such as links and routers. This leads to the corruption of information exchanged between cores or potentially misrouting data to the wrong destination when the head flit containing the data route is corrupted. To effectively simulate message corruption within the ATTNoC, a technique known as "flipping a bit" was employed. This involved intentionally introducing bit flips to corrupt messages and impact the data within NoC resources.

The redundancy mechanisms integrated into network interfaces (NIs) played a crucial role in the experiment, explicitly designed to minimize the impact of transient faults and improve overall system reliability. While ATTNoC's adaptation techniques primarily focused on addressing permanent faults, the evaluation also extended to understanding how these redundancy mechanisms effectively mitigated the effects of transient faults that could corrupt messages within NoC resources such as links and routers.

The following section will describe the experimental setup, present the results, and provide detailed evaluations of the findings.

Experiment Setup

An intellectual property (IP) for injecting faults, proposed by [Lal12], was integrated into the experimental setup. It aimed to introduce transient faults into the fault containment regions defined for seamless redundancy mechanisms, as described in 6.3.2. 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 network interface, 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 7.22. 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 established: The baseline case without injected faults, the case with disabled redundancy in the safety-critical NI while faults continued to be injected, and the case activating the redundancy mechanism in the safety-critical NI and injecting faults using the 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.



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

Figure 7.21: Block diagram of ATTNoC with Fault Injection IP (FI-IP).

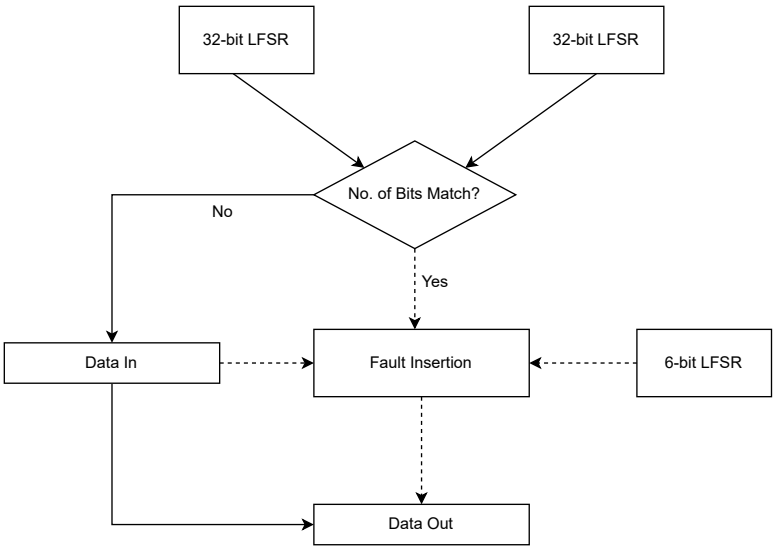


Figure 7.22: Flow of transient fault injection [Lal12].

Experiment 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 7.23. This figure compares the number of packets received without corruption and the error rate for each case. The observations from the analysis are as follows.

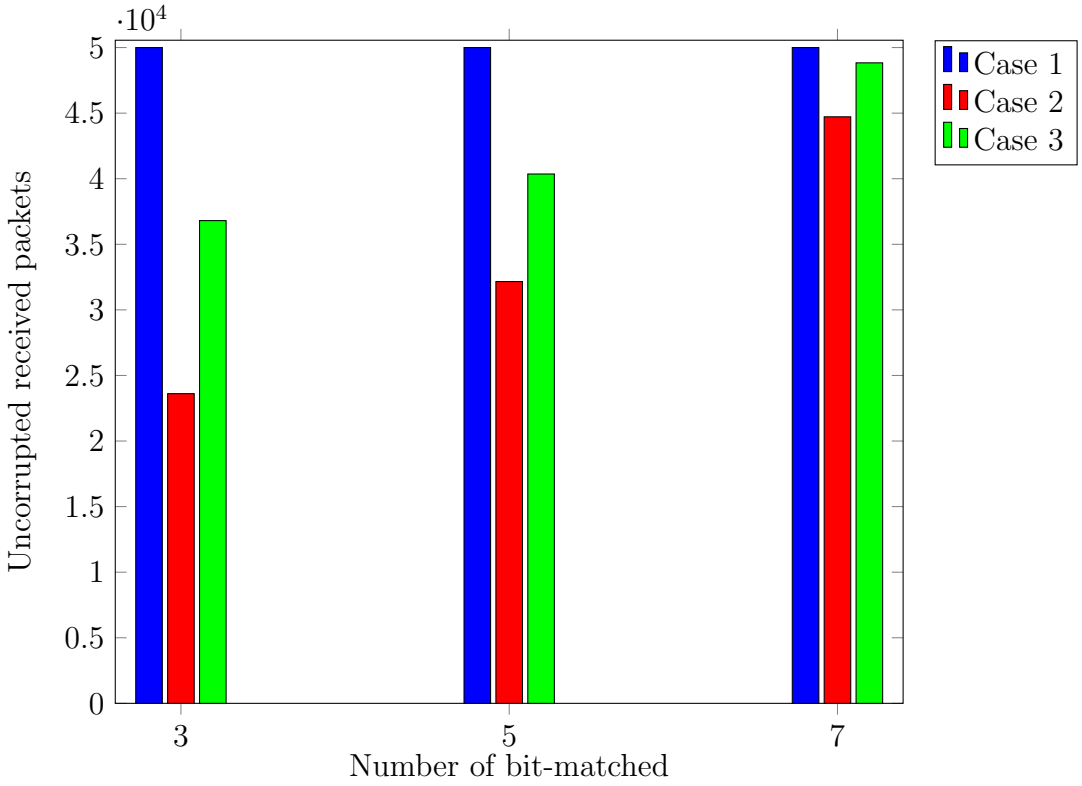


Figure 7.23: Comparison of uncorrupted packets received vs error rate for the three cases

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 overheads, such as increased resource usage. For example, incorporating redundancy

mechanisms into time-triggered network interfaces (TTNI) resulted in an increase in the number of registers and LUTs, as illustrated in Table 7.1.

Based on the results in Figure 7.23, the importance of using redundancy mechanisms in safety-critical applications, such as avionics or defence systems, becomes evident [HEAc]. 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.

Chapter 8

Conclusion and Future Work

In conclusion, the NoC-based multi-core architecture presented in this thesis offers a versatile solution to enhance energy efficiency and reliability in embedded systems, particularly in real-time applications. The ATTNoC architecture achieves a balance between energy efficiency, reliability, and performance through the integration of time-triggered communication, seamless redundancy, adaptation, and frequency scaling. The need for energy-efficient and safe applications in embedded systems has been the driving force behind this research. The ATTNoC architecture 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 efficient use of resources. 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 fault tolerance mechanism efficiently handles router faults by rerouting messages, thereby maintaining uninterrupted communication. Additionally, adopting time-triggered frequency scaling within the ATTNoC architecture enables predictable communication and facilitates frequency scaling of routers based on a predefined schedule. This approach reduces power consumption while ensuring predictable communication within the system. The research extends the functionality of event-triggered NoC architectures, specifically LISNoC, by introducing source-based routing, adaptability, seamless redundancy, and time-triggered frequency scaling. These extensions enhance its capabilities, making it suitable for real-time systems, adaptable to changing requirements, energy-efficient, and deadlock-free. In future work, the architecture's capabilities could be further expanded to support more complex multi-core systems and optimize static power usage to improve energy efficiency further. 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 pre-computed 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. Overall, the ATTNoC-based multi-core architecture provides a comprehensive and practical solution for enhancing energy efficiency and reliability in embedded systems, particularly in high-performance and

safety-critical applications. Its potential impact is significant, as it has the potential to revolutionize NoC-based multi-core platforms, enabling the development of more efficient and dependable embedded systems. The architecture effectively addresses the challenges of energy efficiency, reliability and performance, offering a pathway to achieve higher performance, energy efficiency, and reliability in embedded systems commonly used in real-time applications.

Bibliography

[Abu17] Mohammed Abuteir. "Architecture design for distributed mixed-criticality systems based on multi-core chips". PhD thesis. Universität Siegen, 2017. URL: https://dspace.ub.uni-siegen.de/handle/ubsi/1131.

[AGMK94] Adelberg, Garcia-Molina, and Kao. "Emulating soft real-time scheduling using traditional operating system schedulers". In: *1994 Proceedings Real-Time Systems Symposium*. 1994, pp. 292–298. DOI: 10.1109/REAL.1994.342704.

[Adr+03] A. Adriahantenaina et al. "SPIN: a scalable, packet switched, on-chip micro-network". In: *2003 Design, Automation and Test in Europe Conference and Exhibition*. 2003, 70–73 suppl. DOI: 10.1109/DATE.2003.1253808.

[Agr21] Pankaj Agrawal. "An Efficient Virtual Channel Router for NoC's". In: May 2021.

[AO15] Hamidreza Ahmadian and Roman Obermaisser. "Time-Triggered Extension Layer for On-Chip Network Interfaces in Mixed-Criticality Systems". In: *2015 Euromicro Conference on Digital System Design*. 2015, pp. 693–699. DOI: 10.1109/DSD.2015.33.

[AA14] Hussain Al-Asaad. "Real time scheduling of multiple executions of tasks to achieve fault tolerance in multiprocessor systems". In: *2014 IEEE AUTOTEST*. 2014, pp. 323–328. DOI: 10.1109/AUTEST.2014.6935165.

[AS14] Kaushal Amandeep and Singh Sarbdeep. "Network on Chip Architecture and Routing Techniques: A survey". In: *International Journal of Research in Engineering and Science (IJRES)* 2 (2014).

[AA05] 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. DOI: 10.1109/ICVD.2005.125.

[AMP19] Jyotika Athavale, Riccardo Mariani, and Michael Paulitsch. "Flight Safety Certification Implications for Complex Multi-Core Processor Based Avionics Systems". In: *2019 IEEE International Reliability Physics Symposium (IRPS)*. 2019, pp. 1–6. DOI: 10.1109/IRPS.2019.8720422.

[Ava+10] Annie Avakian et al. “A reconfigurable architecture for multicore systems”. In: *2010 IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW)*. 2010, pp. 1–8. DOI: 10.1109/IPDPSW.2010.5470753.

[ALR01] Algirdas Avizienis, Jean-claude Laprie, and Brian Randell. “Fundamental Concepts of Dependability”. In: (Sept. 2001).

[Bar+06] Chris Bartels et al. “Comparison of An Æthereal Network on Chip and A Traditional Interconnect for A Multi-Processor DVB-T System on Chip”. In: *2006 IFIP International Conference on Very Large Scale Integration*. 2006, pp. 80–85. DOI: 10.1109/VLSISOC.2006.313208.

[Bei+05] E. Beigne et al. “An asynchronous NOC architecture providing low latency service and its multi-level design framework”. In: *11th IEEE International Symposium on Asynchronous Circuits and Systems*. 2005, pp. 54–63. DOI: 10.1109/ASYNC.2005.10.

[Bet97] Sharad Sundaresan; Riccardo Bettati. “Distributed Connection Management for Real-Time Communication over Wormhole-Routed Networks”. In: *ICDCS ’97: Proceedings of the 17th International Conference on Distributed Computing Systems (ICDCS ’97)*. 1997.

[BD07] Alan Burns and Robert I Davis. “A survey of research into mixed criticality systems”. In: *ACM Computing Surveys (CSUR)*. Vol. 50. 6. 2007.

[CKP18] Jehee Cha, Jiho Kim, and Yongjun Park. “Core-level DVFS for Spatial Multitasking GPUs”. In: *TENCON 2018 - 2018 IEEE Region 10 Conference*. 2018, pp. 1525–1528. DOI: 10.1109/TENCON.2018.8650072.

[Che+20] R. Cheour et al. “Accurate Dynamic Voltage and Frequency Scaling Measurement for Low-Power Microcontrollors in Wireless Sensor Networks”. In: *2020 Microelectronics Journal*. 2020. DOI: DOI:10.1016/j.mejo.2020.104874.

[Che+13] Kazem Cheshmi et al. “Quota setting router architecture for quality of service in GALS NoC”. In: *2013 International Symposium on Rapid System Prototyping (RSP)*. 2013, pp. 44–50. DOI: 10.1109/RSP.2013.6683957.

[CS18] Avishek Choudhury and Biplab K. Sikdar. “Modeling Analysis of Redundancy Based Fault Tolerance for Permanent Faults in Chip Multiprocessor Cache”. In: *2018 31st International Conference on VLSI Design and 2018 17th International Conference on Embedded Systems (VLSID)*. 2018, pp. 115–120. DOI: 10.1109/VLSID.2018.47.

[Col+22] Louella Colaco et al. “ARMS: An Analysis Framework for Mixed Criticality Systems”. In: *2022 IEEE 1st International Conference on Data, Decision and Systems (ICDDS)*. 2022, pp. 1–6. DOI: 10.1109/ICDDS56399.2022.10037556.

[Con+06a] K. Constantinides et al. “BulletProof: a defect-tolerant CMP switch architecture”. In: *The Twelfth International Symposium on High-Performance Computer Architecture, 2006.* 2006, pp. 5–16. DOI: 10.1109/HPCA.2006.1598108.

[Con+06b] K Constantinides et al. "BulletProof: a defect-tolerant CMP switch architecture". In: *The Twelfth International Symposium on High-Performance Computer Architecture, 2006.* 2006, pp. 5–16. DOI: 10.1109/HPCA.2006.1598108.

[DeO+12] Andrew DeOrio et al. "A Reliable Routing Architecture and Algorithm for NoCs". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 31.5 (2012), pp. 726–739. DOI: 10.1109/TCAD.2011.2181509.

[DS10] Ulhas Deshmukh and Vineet Sahula. "Stochastic Automata Network Based Approach for Performance Evaluation of Network-on-Chip Communication Architecture". In: *2010 IEEE Computer Society Annual Symposium on VLSI*. 2010, pp. 351–356. DOI: 10.1109/ISVLSI.2010.97.

[Dub08] Elena Dubrova. *Fault-Tolerant Design: An Introduction*. Kluwer Academic Publishers, 2008.

[Egh+10] Ashkan Eghbal et al. "Designing fault-tolerant network-on-chip router architecture". In: *International Journal of Electronics* 97.10 (2010), pp. 1181–1192. DOI: 10.1080/00207217.2010.512016. eprint: https://doi.org/10.1080/00207217.2010.512016. URL: https://doi.org/10.1080/00207217.2010.512016.

[Fer+04] M. Fernando et al. "Hermes: an infrastructure for low area overhead packet- switching networks on chip". In: *Integration, vol. 38, no. 1, pp. 69-93, 2004*. 2004, pp. 69–93.

[Fre+10] Henrique Cota de Freitas et al. "Impact of Parallel Workloads on NoC Architecture Design". In: *2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing*. 2010, pp. 551–555. DOI: 10.1109/PDP.2010.53.

[HEAc] RICK HEARN. "Military Embedded Systems". In: *Optimizing avionics reliability with dissimilar redundant architectures* (Dec 06, 2018). URL: https://militaryembedded.com/avionics/computers/optimizing-reliability-dissimilar-redundant-architectures.

[Ila+20] S. Ilager et al. "A Data-Driven Frequency Scaling Approach for Deadline-aware Energy Efficient Scheduling on Graphics Processing Units (GPUs)". In: *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*. Los Alamitos, CA, USA: IEEE Computer Society, 2020, pp. 579–588. DOI: 10.1109/CCGrid49817.2020.00-35. URL: https://doi.ieeecomputersociety.org/10.1109/CCGrid49817.2020.00-35.

[Jab+17] Gul Jabeen et al. "Hybrid software reliability prediction model based on residual errors". In: *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*. 2017, pp. 479–482. DOI: 10.1109/ICSESS.2017.8342959.

[JB17] Vahid Janfaza and Elaheh Baharlouei. "A new fault-tolerant deadlock-free fully adaptive routing in NOC". In: *2017 IEEE East-West Design Test Symposium (EWDTS)*. 2017, pp. 1–6. DOI: 10.1109/EWDTS.2017.8110139.

[Jin+15] Jie Jin et al. "Low power design for on-chip networking processing system". In: *2015 28th IEEE International System-on-Chip Conference (SOCC)*. 2015, pp. 154–159. DOI: 10.1109/SOCC.2015.7406931.

[KK19] Nassima Kadri and Mouloud Koudil. "A survey on fault-tolerant application mapping techniques for Network-on-Chip". In: *Journal of Systems Architecture* 92 (2019), pp. 39–52. ISSN: 1383-7621. DOI: https://doi.org/10.1016/j.sysarc.2018.10.001. URL: https://www.sciencedirect.com/science/article/pii/S1383762118301498.

[KLN15] Andrew B. Kahng, Bill Lin, and Siddhartha Nath. "ORION3.0: A Comprehensive NoC Router Estimation Tool". In: *IEEE Embedded Systems Letters* 7.2 (2015), pp. 41–45. DOI: 10.1109/LES.2015.2402197.

[KBB11] Mohammad Reza Kakoee, Valeria Bertacco, and Luca Benini. "ReliNoC: A reliable network for priority-based on-chip communication". In: *2011 Design, Automation Test in Europe*. 2011, pp. 1–6. DOI: 10.1109/DATE.2011.5763112.

[Kas+15] Hany Kashif et al. "Static slack-based instrumentation of programs". In: Sept. 2015, pp. 1–8. DOI: 10.1109/ETFA.2015.7301505.

[Kim+14] Yoongu Kim et al. "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. DOI: 10.1109/ISCA.2014.6853210.

[Kop92] Hermann Kopetz. "Sparse time versus dense time in distributed real-time systems". In: *[1992] Proceedings of the 12th International Conference on Distributed Computing Systems* (1992), pp. 460–467. URL: https://api.semanticscholar.org/CorpusID:45095041.

[KKD18] Harekrishna Kumar, Anjan Kumar, and Vinay Kumar Deolia. "Enabling Concurrent Clock and Power Gating in 32 Bit ROM". In: *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. 2018, pp. 1–6. DOI: 10.1109/ICCCNT.2018.8493779.

[KR07] Ian Kuon and Jonathan Rose. "Measuring the Gap Between FPGAs and ASICs". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 26.2 (2007), pp. 203–215. DOI: 10.1109/TCAD.2006.884574.

[Lal12] P. Lala. "Transient and Permanent Fault Injection in VHDL Description of Digital Circuits". In: *Circuits and Systems, 2012.* Vol. 3. 2. 2012, pp. 192–199. DOI: 10.4236/cs.2012.32026.

[Lee+05] Se-Joong Lee et al. “Packet-switched on-chip interconnection network for system-on-chip applications”. In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 52.6 (2005), pp. 308–312. DOI: 10.1109/TCSII.2005.848972.

[LLP10] Teijo Lehtonen, Pasi Liljeberg, and Juha Plosila. “Analysis of Forward Error Correction Methods for Nanoscale Networks-On-Chip”. In: May 2010. DOI: 10.4108/ICST.NANONET2007.2035.

[Len20] Alina Lenz. “System-wide, fault-tolerant state agreement protocol for time-triggered MPSoC”. PhD thesis. Universität Siegen, 2020. DOI: http://dx.doi.org/10.25819/ubsi/5958. URL: https://dspace.ub.uni-siegen.de/handle/ubsi/1734.

[LC06] Jian Lin and A.M.K. Cheng. “Maximizing Guaranteed QoS in (m, k)-firm Real-time Systems”. In: *12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06)*. 2006, pp. 402–410. DOI: 10.1109/RTCSA.2006.43.

[LGY10] Feiyang Liu, Huaxi Gu, and Yintang Yang. “Performance study of virtual-channel router for Network-on-Chip”. In: *2010 International Conference On Computer Design and Applications*. Vol. 5. 2010, pp. V5–255–V5–259. DOI: 10.1109/ICCDA.2010.5541185.

[M+18] Clark M et al. “Lead: Learning-enabled energy-aware dynamic voltage/frequency scaling in nocs”. In: *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC), pp. 1-6, 2018.* 2018, pp. 1–6.

[MAO18] Adele Maleki, Hamidreza Ahmadian, and Roman Obermaisser. “Fault-Tolerant and Energy-Efficient Communication in Mixed-Criticality Networks-on-Chips”. In: *2018 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC)*. 2018, pp. 1–7. DOI: 10.1109/NORCHIP.2018.8573469.

[Man+21] Riman Mandal et al. “11 - A survey and critical analysis on energy generation from datacenter”. In: *Data Deduplication Approaches*. Ed. by Tin Thein Thwel and G.R. Sinha. Academic Press, 2021, pp. 203–230. ISBN: 978-0-12-823395-5. DOI: https://doi.org/10.1016/B978-0-12-823395-5.00005-7. URL: https://www.sciencedirect.com/science/article/pii/B9780128233955000057.

[Mik+04] Millberg Mikael et al. “Guaranteed Bandwidth using Looped Containers in Temporally Disjoint Networks within the Nostrum Network on Chip”. In: *IEEE, 2004*. 2004, pp. 5–16.

[Mot+13] Boris Motruk et al. “Power Monitoring for Mixed-Criticality on a Many-Core Platform”. In: *Architecture of Computing Systems – ARCS 2013*. Ed. by Hana Kubátová et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 13–24. ISBN: 978-3-642-36424-2.

[Mur+06] S. Murali et al. “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. DOI: 10.1145/1146909.1147124.

[Mur+15] Ayman Murshed et al. “Scheduling and allocation of time-triggered and event-triggered services for multi-core processors with networks-on-a-chip”. In: *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*. 2015, pp. 1424–1431. DOI: 10.1109/INDIN.2015.7281942.

[Nae+10] Abdul Naeem et al. “Scalability of weak consistency in NoC based multicore architectures”. In: *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. 2010, pp. 3497–3500. DOI: 10.1109/ISCAS.2010.5537833.

[NOO23] Rakotojaona Nambinina, Daniel Onwuchekwa, and Roman Obermaisser. “Latency-Aware Frequency Scaling in Time-Triggered Network-on-Chip Architecture”. In: *2023 7th International Conference on Computing Methodologies and Communication (ICCMC)*. 2023, pp. 1480–1488. DOI: 10.1109/ICCMC56507.2023.10084313.

[Nam+21] Rakotojaona Nambinina et al. “Time-Triggered Frequency Scaling in Network-on-Chip for Safety-Relevant Embedded Systems”. In: *2021 International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON)*. 2021, pp. 1–7. DOI: 10.1109/SMARTGENCON51891.2021.9645782.

[Nam+22] Rakotojaona Nambinina et al. “Extension of the LISNoC (Network -on-chip) with an AXI-based Network Interface”. In: *2022 6th International Conference on Computing Methodologies and Communication (ICCMC)*. 2022, pp. 682–686. DOI: 10.1109/ICCMC53470.2022.9753813.

[Nam+23] Rakotojaona Nambinina et al. “Adaptive Time-Triggered Network-on-Chip Architecture: Enhancing Safety”. In: *IEEE 3rd Smart GenCon 2023*. Dec. 2023.

[NLB00] N. Nikaein, H. Labiod, and C. Bonnet. “DDR-distributed dynamic routing algorithm for mobile ad hoc networks”. In: *2000 First Annual Workshop on Mobile and Ad Hoc Networking and Computing. MobiHOC (Cat. No.00EX444)*. 2000, pp. 19–27. DOI: 10.1109/MOBHOC.2000.869209.

[Obe+08] Roman Obermaisser et al. “The time-triggered System-on-a-Chip architecture”. In: *2008 IEEE International Symposium on Industrial Electronics*. 2008, pp. 1941–1947. DOI: 10.1109/ISIE.2008.4677135.

[Obe+19] Roman Obermaisser et al. “Adaptive Time-Triggered Multi-Core Architecture”. In: *Designs* 3.1 (2019). ISSN: 2411-9660. DOI: 10.3390/designs3010007. URL: https://www.mdpi.com/2411-9660/3/1/7.

[Ouy+21] Yiming Ouyang et al. “Fault-tolerant design for data efficient retransmission in WiNoC”. In: *Tsinghua Science and Technology* 26.1 (2021), pp. 85–94. DOI: 10.26599/TST.2019.9010039.

[PCG19] George Papadimitriou, Athanasios Chatzidimitriou, and Dimitris Gizopoulos. "Adaptive Voltage/Frequency Scaling and Core Allocation for Balanced Energy and Performance on Multicore CPUs". In: *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 2019, pp. 133–146. DOI: 10.1109/HPCA.2019.00033.

[PT16] Shuchi A. Parkhani and Vaishali A. Tehre. "A Survey of Different Topologies for Network-on-Chip Architecture". In: 2016.

[PNO23] Daniel Onwuchekwa Josepaul Paulachan, Rakotojaona Nambinina, and Roman Obermaisser. "Time-triggered Network Interface Extension for the Versal Network-on-Chip". In: *2023 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)* (2023).

[Pro16] Position Paper–Multi core Processors. "Certification Authorities Software Team (CAST) Federal Aviation Administration." In: *CAST-32A; Technical report*. 2016.

[Rad+13] Martin Radetzki et al. "Methods for Fault Tolerance in Networks-on-Chip". In: *ACM Computing Surveys (CSUR)* 46 (Oct. 2013). DOI: 10.1145/2522968.2522976.

[Saf+22] Sepideh Safari et al. "A Survey of Fault-Tolerance Techniques for Embedded Systems From the Perspective of Power, Energy, and Thermal Issues". In: *IEEE Access* 10 (2022), pp. 12229–12251. DOI: 10.1109/ACCESS.2022.3144217.

[SCB09] V Sanju, Niranjan N Chiplunkar, and Bini Y Baby. "Design of a generic network on chip frame work for store forward routing for 2D mesh topology". In: *2009 International Conference on Emerging Trends in Electronic and Photonic Devices Systems*. 2009, pp. 104–107. DOI: 10.1109/ELECTRO.2009.5441163.

[Sch07] Martin Schoeberl. "A Time-Triggered Network-on-Chip". In: *2007 International Conference on Field Programmable Logic and Applications*. 2007, pp. 377–382. DOI: 10.1109/FPL.2007.4380675.

[Sei+09] Ciprian Seiculescu et al. "NoC topology synthesis for supporting shutdown of voltage islands in SoCs". In: *2009 46th ACM/IEEE Design Automation Conference*. 2009, pp. 822–825. DOI: 10.1145/1629911.1630121.

[Sha+20] Muhammad Akmal Shafique et al. "NoCGuard: A Reliable Network-on-Chip Router Architecture". In: *Electronics* 9.2 (2020). ISSN: 2079-9292. DOI: 10.3390/electronics9020342. URL: https://www.mdpi.com/2079-9292/9/2/342.

[STS11] Anurag Shrivastav, G.S. Tomar, and Ashutosh Kumar Singh. "Performance Comparison of AMBA Bus-Based System-On-Chip Communication Protocol". In: *2011 International Conference on Communication Systems and Network Technologies*. 2011, pp. 449–454. DOI: 10.1109/CSNT.2011.98.

[Swa+19] Ian Swarbrick et al. “Versal Network-on-Chip (NoC)”. In: *2019 IEEE Symposium on High-Performance Interconnects (HOTI)*. 2019, pp. 13–17. DOI: 10.1109/HOTI.2019.00016.

[Tan+14] Li Tan et al. “A survey of power and energy efficient techniques for high performance numerical linear algebra operations”. In: *Parallel Computing* 40.10 (2014), pp. 559–573. ISSN: 0167-8191. DOI: https://doi.org/10.1016/j.parco.2014.09.001. URL: https://www.sciencedirect.com/science/article/pii/S0167819114001112.

[Tat+14] Konstantinos Tatas et al. *Designing 2D and 3D network-on-chip architectures*. Springer, 2014.

[TH99] H. Thane and H. Hansson. “Towards systematic testing of distributed real-time systems”. In: *Proceedings 20th IEEE Real-Time Systems Symposium (Cat. No.99CB37054)*. 1999, pp. 360–369. DOI: 10.1109/REAL.1999.818863.

[TUM] TUM. *LISNoC*. https://www.ce.cit.tum.de/lis/forschung/aktuelle-projekte/optimsoc/lis-noc/. Accessed 2020-12-25.

[Vai+19] Sundriyal Vaibhav et al. “Effect of frequency scaling granularity on energy-saving strategies”. In: *2019 International Journal of High performance computing applications*. 2019.

[VHL14] Marcus Völp, Marcus Hähnel, and Adam Lackorzynski. “Has energy surpassed timeliness? Scheduling energy-constrained mixed-criticality systems”. In: *2014 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS)*. 2014, pp. 275–284. DOI: 10.1109/RTAS.2014.6926009.

[WCK08] Isask’har Walter, Israel Cidon, and Avinoam Kolodny. “BENoC: A Bus-Enhanced Network on-Chip for a Power Efficient CMP”. In: *IEEE Computer Architecture Letters* 7.2 (2008), pp. 61–64. DOI: 10.1109/L-CA.2008.11.

[Wan+15] Junshi Wang et al. “Design of Fault-Tolerant and Reliable Networks-on-Chip”. In: *2015 IEEE Computer Society Annual Symposium on VLSI*. 2015, pp. 545–550. DOI: 10.1109/ISVLSI.2015.33.