Friedrich-Alexander-Universität
Technische Fakultät

Lehrstuhl für Informatik 3

Rechnerarchitektur

Dustin Heither

Flexible and Efficient QoS Provisioning in AXI4-Based Network-on-Chip Architecture - A brief comprehension

Paper in the subject ’Neuartige Rechnerarchitekturen’

11. Oktober 2025

Please cite as:
Dustin Heither, “Flexible and Efficient QoS Provisioning in AXI4-Based Network-on-Chip Architecture - A brief comprehension,” Seminar Paper, University of Erlangen, Dept. of Computer Science, October 2025.

Friedrich-Alexander-Universität
Erlangen-Nürnberg

Friedrich-Alexander-Universität Erlangen-Nürnberg
Department Informatik
Rechnerarchitektur

Martensstr. 3 · 91058 Erlangen · Germany

www3.cs.fau.de

Flexible and Efficient QoS Provisioning in AXI4-Based Network-on-Chip Architecture - A brief comprehension

Paper in the subject 'Neuartige Rechnerarchitekturen'

vorgelegt von

Dustin Heither

geb. am 10. May 1992
in Oberhausen

angefertigt am

Lehrstuhl für Informatik 3
Rechnerarchitektur

Department Informatik
Friedrich-Alexander-Universität Erlangen-Nürnberg

Betreuer: **M.Sc. Philipp Holzinger**
Betreuender Hochschullehrer: **Prof. Dr.-Ing. Dietmar Fey**

Beginn der Arbeit: **1. April 2025**
Abgabe der Arbeit: **11. Oktober 2025**

Erklärung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde.
Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Declaration

I declare that the work is entirely my own and was produced with no assistance from third parties.
I certify that the work has not been submitted in the same or any similar form for assessment to any other examining body and all references, direct and indirect, are indicated as such and have been cited accordingly.

(Dustin Heither)
Erlangen, 11. Oktober 2025

Contents

List of Acronyms

ACE	AXI Coherency Extensions
AHB	Advanced High-performance Bus
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
ASB	Advanced System Bus
AXI3	Advanced eXtensible Interface 3
AXI4	Advanced eXtensible Interface 4
BiNoC	Bidirectional Network-on-Chip
CHI	Coherent Hub Interface
DiffServ	Differentiated Services
DNN	Deep Neural Network
FinFET	Fin Field-Effect Transistor
GRS	Guaranteed-Rate Service
GS	Guaranteed Service
IntServ	Integrated Services
IPTV	Internet Protocol Television
LCS	Latency-Critical Service
MMP	Markov Modulated Process
NI	Network Interface
NoC	Network-on-Chip
QoS	Quality of Service
SoC	System-on-Chip
TDM	Time Divison Multiplexing
TMU	Transaction Monitoring Unit

URS	Unspecified-Rate Service
VC	Virtual Channel
VoD	Video on Demand
VoIP	Voice over IP

Chapter 1

Introduction

As part of the seminar “Neuartige Rechnerarchitekturen”, the paper “Flexible and Efficient QoS Provisioning in AXI4-Based Network-on-Chip Architecture” by Wang and Lu (2022) is analyzed in detail [1]. The goal of this analysis is to engage deeply with the presented architecture, identify key concepts and findings, and prepare a seminar presentation that clearly conveys these to fellow students. The seminar is part of the Bachelor’s program in Computer Science at Friedrich-Alexander-Universität Erlangen-Nürnberg and was attended during the summer semester of 2025.

The title of the paper already points to three essential core aspects: Quality of Service (QoS), Advanced eXtensible Interface 4 (AXI4), and Network-on-Chip (NoC). These terms will be explained in more detail, placed in their technical context, and examined in relation to each other. The objective is to provide a comprehensive overview and analysis of the architecture developed by Wang and Lu.

With the increasing number of processor cores and functional units in modern System-on-Chip (SoC) designs, the demands on efficient and reliable communication between these components also grow. For this reason, NoC architectures are becoming increasingly important. At the same time, many applications require low latencies and guaranteed bandwidth, making flexible QoS support essential. The AXI4 standard has become a widely adopted protocol in the industry, highlighting the relevance and timeliness of the approach presented by Wang and Lu [2–4].

To provide technical context, the key concepts (NoC, QoS, and AXI4) are first compared with the current state of the art and relevant research literature. This is followed by a detailed description of the architecture introduced by Wang and Lu, including discussion of the role of the subnetworks (Virtual Channel (VC) and Time Divison Multiplexing (TDM)) and the Network Interface (NI). Building on this, the experimental results are presented and critically evaluated.

Chapter 2

Background and State of the Art

2.1 Network-On-Chip

EVOLUTION OF BUSES

A network-on-chip is a communication system used in modern SoC architectures to efficiently connect various components (e.g., processors, memory, and specialized units). Instead of using classic buses or point-to-point connections, NoC relies on a network-like communication principle inspired by computer networks or high-performance computers [5].

Figure 2.1 shows the development of bus technologies over the past few years.

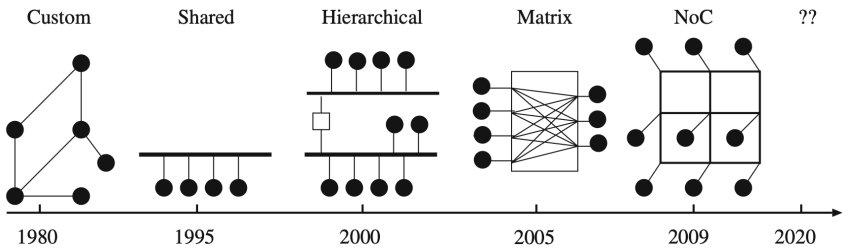


Figure 2.1 – Evolution of Interconnections [6]

Before 1980, custom solutions were typically used for on-chip communication. Starting around 1995, so-called shared-bus architectures such as ARM's AMBA bus [7] and IBM's CoreConnect [8] were introduced. These approaches enabled a more modular design with standardized interfaces and supported the reuse of IP components (IP reuse) [9].

However, as bandwidth demands increased, shared-bus systems became a bottleneck. To address this issue, hierarchical bus architectures were introduced. These utilize multiple buses or bus segments to reduce the load on the main bus. Local communication between modules on the same bus segment is possible without burdening the entire bus. Nevertheless, such architectures are only limitedly scalable, inflexible, and lead to increased design complexity. The more cores are connected, the harder it becomes to meet timing constraints (time closure[1]) and ensure QoS [9].

Another alternative was the bus matrix — a full crossbar system that allows parallel connections between components. However, as system size grows, the complexity of wiring also increases and can eventually improve the effort required for the logic itself. Moreover, such systems do not clearly separate transport, transaction, and physical layers. Therefore, when a system upgrade is needed, it often affects the entire interface design and all connected blocks [9].

[1]Time Closure: Successfully ensuring all timing constraints are satisfied in the design so that the chip can operate reliably at its target clock frequency.

Against this background, some researchers in the early 2000s proposed implementing communication between different processing units on a chip via a predefined platform—an integrated switching network known as Network-on-Chip. NoCs fulfill key requirements of modern SoCs: Reusability, scalable bandwidth, and energy efficiency. NoCs have replaced wired connections and instead utilize intelligent network infrastructures. They draw on models, techniques, and tools from network communication, thereby replacing fixed wiring with packet-based communication [9].

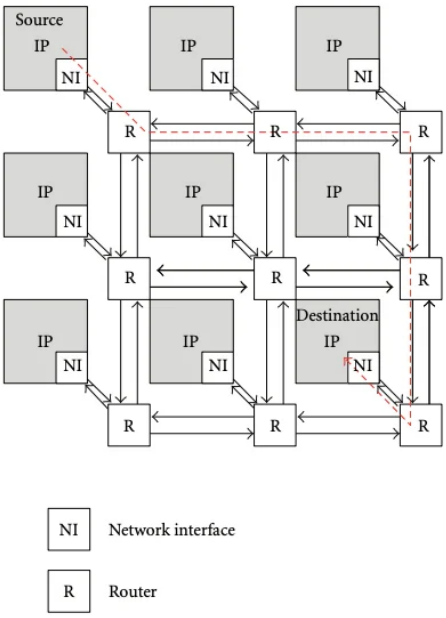


Figure 2.2 – NoC Components
[10]

NOC COMPONENTS
A NoC consists - as you can see in figure 2.2 - of the following three essential components [9, 11]:

1. **Links** physically connect routers and enable data transfer between them. A link may contain multiple logical channels, each implemented over a set of wires with synchronization logic.

2. **Routers** establish the communication paths through switching matrices connecting multiple input and output ports. They implement routing and flow control policies that define the forwarding strategy.

3. **NIs** bridge the IP cores and the NoC. Since IPs often communicate via bus protocols like AMBA AXI, NIs translate transactions into packets suitable for the NoC.

ADVANTAGES AND DISADVANTAGES

A NoC offers several advantages over traditional bus-based communication structures: It enables scalable and parallel data transfer between many processor cores or components, significantly improving performance and efficiency in complex systems. Moreover, the structured interconnection enhances fault tolerance and bandwidth while reducing latency [12].

However, NoCs also bring disadvantages, such as increased design and implementation effort, as well as additional chip area and power consumption. In particular, the complexity of routing and control can complicate development and validation. Overall, however, in modern multicore systems the advantages of NoCs usually outweigh the disadvantages, as they provide a flexible and high-performance communication infrastructure [12].

Note: Detailed discussions of NoC topology, routing, flow control, and protocol variations are omitted due to space limitations. For a detailed information about these subtopics, see [9, 12–17].

2.2 AMBA and AXI-Protocol

EVOLUTION OF AMBA

Advanced Microcontroller Bus Architecture is an open-standard, on-chip interconnect specification for the connection and management of functional blocks in SoC-designs.

The Advanced Microcontroller Bus Architecture (AMBA) standard was first introduced by ARM in 1996 to provide a scalable and reusable on-chip communication interface for SoC-designs. Over time, AMBA evolved through several generations to meet increasing system complexity and performance requirements [18].

- **AMBA 1 (1996)**: Introduced basic buses like Advanced Peripheral Bus (APB) and Advanced System Bus (ASB) for simple communication.
- **AMBA 2 (1999)**: Added the Advanced High-performance Bus (AHB), offering higher performance for pipelined systems.
- **AMBA 3 (2003)**: Introduced Advanced eXtensible Interface 3 (AXI3), supporting out-of-order transactions and multiple outstanding operations.
- **AMBA 4 (2010)**: Introduced AXI4, which includes. *Note: Due to space limitations, detailed explanations of the simplified AXI4-Lite and streaming-oriented AXI4-Stream variants are omitted.*
  - AXI4: Full-featured interface with support for burst transactions.
  - AXI4-Lite: Simplified, single transaction interface used for register-mapped control [19].
  - AXI4-Stream: Designed for high-speed data streaming without address lines [20].
- **AMBA 5 (2013 +)**: Introduced AXI Coherency Extensions (ACE) and Coherent Hub Interface (CHI) for multicore and cache-coherent systems.

Among the various interfaces introduced in the AMBA family, the AXI4 protocol has become the de facto standard for high-performance interconnects in modern SoC designs. AXI4 provides a flexible, high-bandwidth, and scalable communication mechanism suitable for a wide range of applications, from simple register access to complex burst-based memory transfers [21].

AXI4 PROTOCOL

The AXI4 protocol is a key part of the AMBA specification, designed for high-performance, high-frequency system designs. It provides a point-to-point interface between master and slave components, supporting efficient burst-based data

transfers with minimal latency and flexible timing. AXI4 separates read and write operations into independent channels, allowing simultaneous or decoupled transactions in both directions. This decoupling enables high throughput and efficient use of bus bandwidth, which is critical in complex SoC architectures [22].

AXI CHANNELS

The AMBA AXI4 protocol defines five independent channels for read and write operations. Write transactions use the **AW** (address/control), **W** (data), and **B** (response) channels, while read transactions use the **AR** (address/control) and **R** (data/response) channels [22].

By decoupling address and data transfers, AXI4 enables out-of-order and pipelined communication between masters and slaves, improving throughput and flexibility. All channels employ a simple `VALID/READY` handshake: data is transferred only when both signals are asserted, ensuring reliable and flow-controlled transactions [22].

TRANSACTIONS & BURST MODES

In the AXI protocol, a **transfer** refers to a single data exchange synchronized by a `VALID/READY` handshake, while a **transaction** consists of one or more such transfers forming a complete read or write operation [22].

Each transaction is characterized by control attributes that define its structure and behavior, including the number of data beats, the transfer size, and the burst type. These parameters allow AXI to efficiently support both single accesses and high-throughput burst transactions [22].

AXI supports several **burst modes** that control how addresses change across consecutive data transfers. This allows efficient handling of both fixed-address operations (e.g., register access) and sequential block transfers in memory [22].

ADVANTAGES

To summarize, AXI4 incorporates several key features that make it well-suited for high-performance systems. One of its core advantages is the use of independent read and write channels, which allows these operations to be carried out concurrently, thereby increasing overall throughput.

Additionally, AXI4 supports multiple outstanding transactions, enabling masters to issue several requests without needing to wait for prior transactions to complete. The protocol also permits out-of-order completion of responses, granting flexibility in how slaves manage and return data.

Efficient data transfer is further achieved through the use of burst transactions, which allow multiple data beats to be sent in a single transfer. Moreover, AXI4 does not impose a strict timing relationship between the address and data phases, offering greater design flexibility. Finally, all communication channels rely on a

straightforward two-way handshake protocol, utilizing the `VALID` and `READY` signals for effective flow control [23, 24].

2.3 Quality of Service

WHAT IS QUALITY OF SERVICE IN NETWORKING?

QoS refers to a set of technologies and techniques used in networking to manage traffic and ensure the efficient and predictable performance of critical applications. It allows organizations to prioritize certain types of traffic over others, ensuring that resource-intensive or time-sensitive services maintain high performance even under constrained bandwidth conditions.

QoS is particularly important in networks that handle real-time data, such as Internet Protocol Television (IPTV), online gaming, media streaming, video conferencing, Video on Demand (VoD), and Voice over IP (VoIP). By applying QoS policies, organizations can optimize the behavior of multiple applications, gaining visibility and control over network characteristics such as bit rate, jitter, packet loss, and latency [25–28].

TYPES OF TRAFFIC

Different types of traffic are affected by various network parameters that influence performance. *Bandwidth* refers to the maximum rate at which data can be transferred across the network, while throughput represents the actual rate achieved. *Latency* is the delay experienced in transmitting data from source to destination. *Jitter*, on the other hand, refers to the variation in packet arrival times, often caused by network congestion. This variation can lead to packets arriving late or out of sequence, affecting the quality of real-time applications such as voice and video [27, 28].

HOW DOES QOS WORK?

As businesses increasingly rely on networks to transmit information between endpoints, data is divided into packets for transmission. These packets, much like letters in envelopes, are routed through the network. Since bandwidth is limited, QoS is responsible for determining which packets receive priority to ensure that critical traffic is delivered reliably and efficiently. QoS achieves this by classifying traffic based on predefined policies and assigning priorities to different classes of traffic. High-priority packets, such as those carrying voice or video, are given preferential treatment over less critical traffic like file downloads or email.

The implementation of QoS involves several key mechanisms. Traffic classification and marking are used to identify and label packets according to their type or importance. Queuing mechanisms then determine the order in which packets are transmitted, using techniques such as priority queuing or weighted fair queuing. Bandwidth management ensures that different traffic types receive appropriate resource allocation, while policing and shaping tools are used to enforce traffic limits or smooth traffic flows. Congestion management techniques help to prevent buffer

overflow and packet loss during periods of high network usage. These combined techniques ensure that network resources are allocated efficiently and in accordance with organizational priorities [25–27].

QOS MODELS

QoS can be implemented using several different models. The best-effort model provides no guarantees and treats all traffic equally, making it suitable only for non-critical applications [29].

The Integrated Services (IntServ) model offers strict QoS guarantees by reserving network resources for specific traffic flows using signaling protocols such as the Resource Reservation Protocol (RSVP) [30].

In contrast, the Differentiated Services (DiffServ) model is more scalable and widely used in modern enterprise networks. DiffServ classifies and manages traffic into different service levels without requiring end-to-end signaling, allowing more flexible and efficient QoS implementation [29].

2.4 Related Work

RECENT PUBLICATIONS

Since the authors provided an overview of related work within their paper (see Wang & Lu [p. 1524–1526, Sec. II]), therefore this chapter focuses on several new contributions that are closely related to AXI4-based NoCs and QoS-aware architectures.

PATRONoC introduces an open-source, fully AXI4-compliant NoC fabric specifically designed for multi-accelerator Deep Neural Network (DNN) platforms. It demonstrates up to 34% improved area efficiency and achieves between two- and eight-fold throughput improvements compared to state-of-the-art designs [31].

Another line of research is represented by FlooNoC, which has appeared in two iterations. The 2023 version presents a low-latency, wide-channel AXI4-compatible NoC, achieving 629 Gbps per link at 1.23 GHz in 12 nm Fin Field-Effect Transistor (FinFET)[2] technology with only 10% area overhead [32].

An extended 2024 version further enhances performance by reaching 645 Gbps per link, 103 Tbps aggregate bandwidth, and energy efficiency of 0.15 pJ/B per hop, while still maintaining very low hardware overhead [33].

AXI-REALM (2023–2024) provides a real-time extension for AXI4 interconnects, introducing credit-based traffic regulation and observability of per-manager traffic. When implemented in a Linux-capable RISC-V SoC, it reduces worst-case memory latency from over 264 cycles to fewer than 8 cycles, with an area overhead as low as 2.45% [34, 35].

From the perspective of QoS-aware router design, AQ-BiNoC introduces an anticipative mechanism that improves the latency of high-priority Guaranteed Service (GS) packets by roughly 14–35% compared to traditional NoC and Bidirectional Network-on-Chip (BiNoC)[3] routers under various traffic scenarios [36].

Finally, reliability aspects of AXI4 have been addressed in 2025 with the proposal of a Transaction Monitoring Unit (TMU) for AXI4 interconnects, which enables real-time detection of protocol violations and timeout faults. The TMU supports monitoring of up to 32 outstanding transactions, offering different levels of granularity to balance coverage and cost [37].

[2] FinFET is a 3D transistor structure where the conducting channel is formed in a thin vertical “fin” of silicon. Unlike traditional planar MOSFETs, its gate wraps around multiple sides of the fin, providing better electrostatic control, reducing leakage current, and enabling higher switching speeds and energy efficiency at advanced technology nodes (e.g., 12 nm, 7 nm).

[3] A BiNoC is a NoC architecture in which communication channels between routers can dynamically switch direction. Instead of having two fixed unidirectional links per connection, a single physical link can be time-multiplexed to carry data either way, depending on traffic demand.

Chapter 3

Structure of the proposed Architecture

OVERALL STRUCTURE

The following chapter summarizes the architecture proposed by Wang and Lu [1].

The proposed NoC architecture in Figure 3.1 is designed to support three distinct QoS) schemes in order to satisfy the heterogeneous communication requirements of different AXI4 masters and slaves:

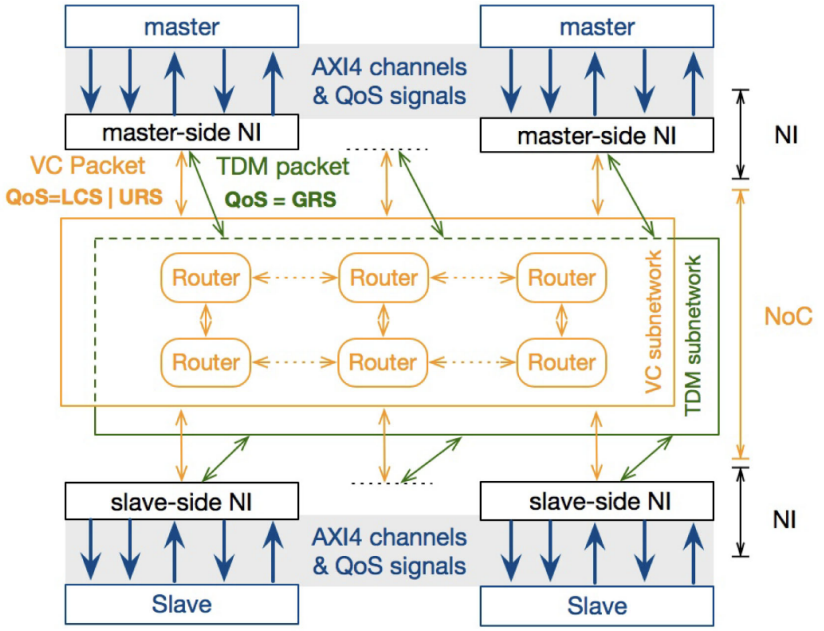


Figure 3.1 – System architecture

- Latency-Critical Service (LCS) A low-latency forwarding service designed for bursty but non-streaming message transmissions. It provides fast delivery but does not guarantee bandwidth. Typical use cases include CPU-like masters that require short response times.
- Guaranteed-Rate Service (GRS) A streaming service that ensures guaranteed bandwidth for large-volume data flows. It tolerates moderate latency but requires sustained throughput, as in GPU-like masters or other bandwidth-demanding accelerators.
- Unspecified-Rate Service (URS) A best-effort service that relies on currently available resources. It provides neither guaranteed bandwidth nor low latency, but aims at fairness among flows. URS is suitable for I/O interfaces such as SATA or USB.

To accommodate these QoS requirements, the system architecture is divided into three main components: (i) AXI4-based master/slave nodes, (ii) NIs that perform protocol message conversion and QoS mapping, and (iii) the NoC fabric itself, which consists of two subnetworks: a VC-based subnetwork for LCS and URS traffic, and a TDM-based subnetwork for GRS traffic.

This separation enables the NoC to meet diverse application needs without compromising performance or protocol compliance.

MESSAGE FORMAT CONVERSION

Two approaches can be used for message format conversion in AXI4-based NoCs. The first is a direct mapping, where each of the five AXI4 channels is converted into a separate packet format. In this case, the NoC must provide five dedicated paths in both subnetworks, one for each AXI4 channel. Although this preserves the semantics of AXI4, it couples the NoC tightly to the protocol, reduces design flexibility, and leads to poor resource utilization due to missing resource sharing.

The second approach consolidates AXI4 transactions into four unified packet types: *read request*, *read response*, *write request*, and *write response*. These packets share the same NoC resources and are annotated with a QoS identifier (LCS, GRS, or URS), which determines whether they are routed through the VC or TDM subnetwork. This design choice decouples the NoC from the specifics of the AXI4 protocol, improves resource utilization, and enables compatibility with a wide range of interconnect architectures (e.g., buses or NoCs).

For these reasons, the authors adopt the second approach as the foundation of their high-performance and flexible AXI4-based NoC system. Based on this design, the network interface fulfills three core functionalities:

1. **Dispatching:** The NI receives signals from the AXI4 channels and NoC subnetworks.Transactions are dispatched either to the appropriate AXI4 channel (read/write) or to the correct NoC subnetwork (VC or TDM) according to their QoS identifier (LCS, GRS, URS).

2. **Message format conversion:** The NI translates AXI4 transactions into NoC packets and vice versa, thereby decoupling the NoC fabric from protocol-specific details.

3. **QoS inheritance:** On the slave side, response packets inherit the QoS class of their corresponding requests. This ensures that QoS policies are consistently maintained across both request and response paths.

Figure 3.2 illustrates this process. AXI4 transactions from the five original channels (read address, read data, write address, write data, write response) are mapped into four packet types, labeled with QoS information, and forwarded to

the appropriate subnetwork. On the return path, the QoS inheritance mechanism guarantees end-to-end service differentiation across the NoC.

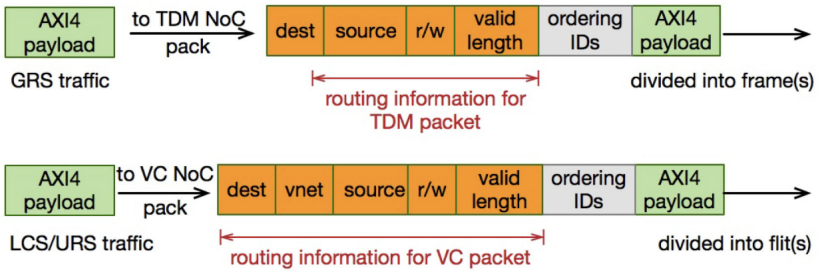


Figure 3.2 – Message format conversion process in the NI

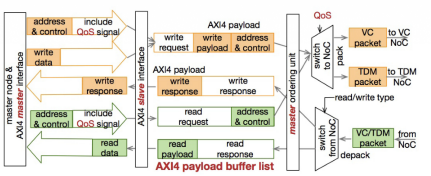
MASTER-SIDE AND SLAVE-SIDE NI ARCHITECTURE

The master-side NI (see Figure 3.3a) converts AXI4 requests into NoC packets and assigns them to the appropriate subnetwork (VC or TDM) based on their QoS identifier (LCS, GRS, URS). It ensures that transactions are correctly encapsulated and routed to meet the QoS requirements of the originating master (e.g., CPU, GPU, I/O).

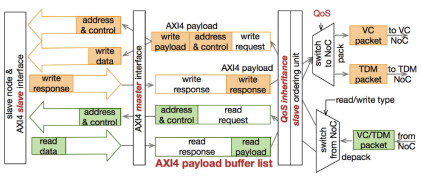
The slave-side NI (see Figure 3.3b), on the other hand, unpacks NoC packets into AXI4 responses and delivers them to the slave devices. Since AXI4 response signals do not carry QoS identifiers, the slave-side NI applies a QoS inheritance mechanism, in which the response packet inherits the QoS class of its corresponding request. This mechanism guarantees that QoS policies are consistently enforced across both request and response paths, thereby maintaining end-to-end service differentiation in the NoC.

QOS INHERITANCE

Because AXI4 response signals do not include QoS identifiers, the NI implements a QoS inheritance mechanism: The response packets automatically inherit the QoS



(a) Master-Side NI architecture



(b) Slave-Side NI architecture

Figure 3.3 – Slave- and Master-Side NI Architectures

information of their corresponding request packets. This ensures consistent QoS handling across request/response transactions.

ROLE OF TWO SUBNETWORKS (VC & TDM)
To efficiently support heterogeneous QoS requirements, the NoC employs a dual-subnetwork structure:

- The VC is used for LCS and URS packets. It provides low-latency transmission for critical traffic and fair, best-effort service for background traffic. Several flow control schemes are supported, ranging from strictly separated VCs for different QoS classes to shared VC approaches with priority arbitration.
- The TDM is dedicated to GRS packets. By pre-allocating both paths and time slots, it guarantees bandwidth and ensures predictable latency for streaming traffic.

The separation into VC and TDM subnetworks prevents interference between different QoS services and allows resources to be allocated more effectively.

The VC-based subnetwork employs several flow control policies that differ in how latency-critical and best-effort packets share virtual channels. Evaluations show that a hybrid scheme, where latency-critical traffic can access all VCs while best-effort traffic is limited to one, achieves the best latency–throughput balance.

In the TDM subnetwork, static routing tables define contention-free paths based on precomputed schedules. Each router forwards packets synchronously according to assigned time slots, providing deterministic latency and guaranteed bandwidth for real-time flows.

TRAFFIC CONVERTER
A key element of the NI is the Traffic Converter (see figure 3.4), which dynamically balances the load between the two subnetworks: the VC subnetwork and the TDM subnetwork.

- **VC to TDM Conversion:** When the VC subnetwork experiences congestion, selected latency-critical (LCS) packets are redirected to the TDM subnetwork. These packets are stored in a dedicated FIFO (GRS_LCS FIFO) and scheduled with lower priority than native guaranteed-rate (GRS) traffic to preserve bandwidth guarantees.
- **TDM to VC Conversion:** If the TDM subnetwork is congested and VC is underutilized, GRS packets may be offloaded to the VC subnetwork. A controller estimates the queuing delay and determines whether rerouting will reduce latency. Converted packets are stored in the LCS_GRS FIFO and fairly arbitrated with native LCS traffic.

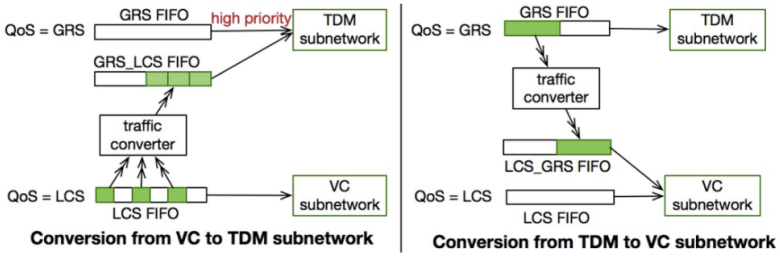


Figure 3.4 – Traffic Conversion Unit

This bidirectional conversion mechanism is implemented after the switch-to-NoC unit and before packetization, allowing real-time traffic adaptation. It improves overall utilization, reduces packet latency, and enhances throughput while maintaining QoS guarantees across traffic classes.

Chapter 4

Results of proposed architecture

4.1 Custom Simulator

The referred authors (Wang & Lu) developed a custom simulator, shown in Figure 4.1 that supports AXI4, a dual-subnetwork NoC architecture, and three QoS schemes simultaneously.

The simulator is built upon the well-known *BookSim2* [38] and *Gem5* [39] frameworks. The implementation, written in C++, models a system consisting of 168 nodes, two subnetworks, and eight off-chip memory controllers. The design is divided into four identical subareas, each organized as a 7×6 mesh structure, where every node is connected to a router.

Each node comprises a processor, a private L1 cache, a shared L2 cache, and both a master-side and slave-side network interface. Consequently, every node is capable of functioning simultaneously as a master and a slave. Additionally, each subarea includes two off-chip memory controllers, which are connected to the central routers within the subarea.

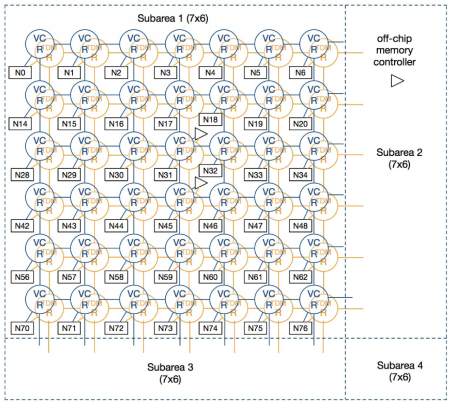


Figure 4.1 – Simulator Architecture [6]

The VC network is based on Gem5, but for ease of implementation it adopts a cycle-triggered model similar to BookSim2, rather than the event-triggered approach used in Gem5.

The TDM network, by contrast, has a simpler architecture, which facilitates straightforward comparison.

The NI is customized to support the AXI4 protocol. Therefore, the results obtained from the proposed simulator can be directly compared with those generated by Gem5 and BookSim2, provided that their NoCs are instantiated with the same configuration.

To regulate the overall injection rate, including LCS, URS, and GRS, the simulator employs a two-level Markov Modulated Process (MMP) model in each traffic gener-

ator. The external MMP models the state of a process or thread, whose execution interval varies from nanoseconds to milliseconds.

The internal MMP, in turn, models the injection state of request messages during the execution of a process or thread. The parameters α and β of the MMP are set according to real-world thread behavior, reflecting both the processing interval and the message injection rate during execution.

The QoS tag for each request is assigned randomly, based on predefined rates. As a result, a processor is restricted to generating one or two types of QoS schemes, similar to processors in real-world systems. The address of each request is also randomly generated from the processor's communication pairs, which record possible interactions between master and slave nodes. Since the influence of the AXI4 ordering requirement is not discussed in this work, the request ID is also selected randomly. This means that ordering constraints may exist for some requests; however, these constraints do not affect the performance of the NoC interconnect, as ordering units are excluded from consideration.

The proposed simulator models a 14×12 mesh NoC topology, where the average hop count across all requests is four. The TDM period consists of 64 slots (i.e., 64 simulation cycles, one cycle per slot). The VC subnetwork includes two virtual networks, each employing an individual flow control mechanism.

Each virtual network contains one virtual channel for LCS packets and one VC for URS packets, with a buffer depth of four flits[4]. The VC router is implemented as a two-stage pipeline, with each stage and each link transfer incurring one cycle of latency.

The NI can operate at up to 600 MHz in 40-nm technology, but is modeled at 500 MHz in the simulator. To align with the 2 GHz global clock, the NI pipeline latency is scaled from one to four NoC cycles, without impacting subnetwork latency.

Furthermore, the NI channel width is doubled from 128 to 256 bits, enabling 128 Gb/s throughput, which exceeds the 117 Gb/s maximum throughput requirement of the 2 GHz subnetworks.

[4]A flit (flow control unit or flow control digit) is a link-level atomic piece that forms a network packet or stream.

4.2 Experimental Results

THROUGHPUT

The proposed NoC architecture achieves high throughput compared with baseline designs. The upper bound analysis shows that the VC subnetwork reaches up to 12.288 Gb/s and the TDM subnetwork up to 7.364 Gb/s, giving a combined throughput of nearly 19.652 Gb/s (around 117 Gb/s per node). The dual-subnetwork design with QoS-aware mechanisms enables efficient resource usage and sustains performance under uniform and non-uniform traffic patterns.

TRAFFIC INJECTION

A two-level MMP-based traffic generator was employed to simulate realistic process/thread execution intervals and injection behavior. Experimental results show that traffic injection exhibits burstiness and suspension periods, closely resembling real workloads. Compared with traditional MMP models, the proposed generator more accurately reflects realistic traffic characteristics, showing intervals of both zero injection and high bursts.

RESOURCE UTILIZATION

The average utilization of VC router ports increases from 4.4% to 16.5% as traffic grows, reaching saturation at around 60 Gb/s per node. The slot utilization of the TDM subnetwork is about 17.12%, constrained by path establishment and deterministic routing. These utilization levels are considered efficient given hardware and contention limitations.

LATENCY

Latency results demonstrate that QoS-aware flow control significantly reduces transfer delay. For latency-critical service (LCS) packets, the *Individual_shared* mechanism yields the lowest latencies, increasing only slightly from 21.2 to 25.3 cycles as injection rates grow from 12 Gb/s to 108 Gb/s.

Best-effort (URS) packets show higher delays, but the trade-off favors LCS performance. Increasing VC numbers can further reduce LCS latency (up to 23.5% improvement), although excessive VCs may cause arbitration overhead.

PERFORMANCE OF TRAFFIC CONVERSION

The traffic converter balances load between the VC and TDM subnetworks. Two scenarios were evaluated:

- Converting LCS packets to GRS traffic reduces average LCS latency in the VC subnetwork from over 1000 cycles to below 65 cycles, while also improving URS latency.

- Converting GRS packets to LCS-oriented VC paths reduces queuing delays in the TDM subnetwork, lowering average latency from 180 cycles to 35 cycles without significantly affecting LCS/URS packets.

This mechanism improves both throughput and latency, achieving up to 93.85% performance improvement compared with static QoS approaches.

Chapter 5

Conclusion and critical reflexion

CONCLUSION

In this paper, a flexible and efficient QoS provisioning scheme was presented for AXI4-based NoC architectures. The proposed design introduces a network interface (NI) capable of AXI4-to-packet conversion, a QoS inheritance mechanism for round-trip support, and a dual-subnetwork structure consisting of a VC-based wormhole network and a TDM-based virtual-circuit network.

Experimental results demonstrate that the architecture achieves high throughput (up to 19,652 Gb/s), low latency for latency-critical flows, and effective traffic balancing through a traffic converter. The system therefore satisfies heterogeneous QoS requirements for CPU-like, GPU-like, and I/O devices within a single unified architecture.

CRITICAL REFLECTION

Although the proposed approach successfully supports three distinct QoS schemes and offers clear performance benefits, several limitations remain:

- **Complexity:** The introduction of dual subnetworks and traffic converters increases hardware and design complexity, potentially impacting scalability and implementation cost.
- **Simulation scope:** Experiments relied on synthetic traffic generators; while the two-level MMP model is more realistic than conventional models, validation under full application-driven benchmarks (e.g., real workloads) remains necessary.
- **Dynamic adaptability:** The traffic converter relies on predefined thresholds and rules for switching packets between subnetworks. More advanced, runtime-adaptive methods (e.g., machine learning-based controllers) could further optimize latency and throughput.

These points highlight promising directions for extending the system towards industrial deployment and real-world applications.

EVALUATION

Overall, the paper makes a strong contribution by bridging the gap between the AXI4 protocol's QoS requirements and efficient NoC design. Compared to existing works, it uniquely integrates three QoS services, a robust NI design, and an effective load-balancing mechanism. The experimental evaluation shows tangible performance improvements with reasonable hardware overhead.

However, the work could be strengthened by including comparisons against more application-specific benchmarks, a deeper analysis of area/power trade-offs, and implementation studies in advanced process technologies. Despite these limitations, the architecture establishes a flexible framework for next-generation SoCs where heterogeneous cores and devices demand simultaneous support for low latency, guaranteed bandwidth, and best-effort services. It therefore represents a valuable step towards scalable and QoS-aware NoC-based communication infrastructures.

List of Figures

Bibliography

[1] B. Wang and Z. Lu, "Flexible and Efficient QoS Provisioning in AXI4-Based Network-on-Chip Archtecture," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 5, pp. 1523–1536, May 2022 (cit. on pp. 1, 13).

[2] S. L. Jake Ke, T. Nowatzki, and J. Cong, "Demystifying FPGA Hard NoC Performance," 2025 (cit. on p. 1).

[3] J. R. Gomez-Rodriguez, R. Sandoval-Arechiga, S. Ibarra-Delgado, V. I. Rodriguez-Abdala, J. L. Vazquez-Avila, and R. Parra-Michel, "A Survey of Software-Defined Networks-on-Chip: Motivations, Challenges and Opportunities," *Micromachines*, 2021 (cit. on p. 1).

[4] B. Talwar and B. Amrutur, "Traffic engineered NoC for streaming applications," *Microprocessors and Microsystems*, 2013 (cit. on p. 1).

[5] D. Serpanos and T. Wolf, "Architecture of Network Systems," in *Architecture of Network Systems*, 2011, pp. 239–248 (cit. on p. 3).

[6] B. A. Abderazek, *Multicore Systems On-Chip: Practical Software/Hardware Design*, 2nd ed. 2013, vol. 7 (cit. on pp. 3, 19).

[7] ARM, *AMBA*. [Online]. Available: `https://www.arm.com/architecture/system-architectures/amba` (cit. on p. 3).

[8] I. B. M. Corporation, *The CoreConnect™ Bus Architecture*, 1999. [Online]. Available: `https://www.scarpaz.com/2100-papers/SystemOnChip/ibm_core_connect_whitepaper.pdf` (visited on 09/01/2025) (cit. on p. 3).

[9] S. Unnikrishnan, *Network on Chip – an Overview*, Nov. 2021. [Online]. Available: `https://ignitarium.com/network-on-chip-an-overview/` (visited on 09/01/2025) (cit. on pp. 3–5).

[10] J. Hertz, *Why SoCs Need NoCs: Network on Chip and the Future of Computing*, Jul. 2025. [Online]. Available: https://www.allaboutcircuits.com/news/why-socs-need-nocs-network-on-chip-and-future-%20computing/ (visited on 09/28/2025) (cit. on p. 4).

[11] Q. Yu and P. Ampadu, "A Flexible and Parallel Simulator for networks-on-Chip with Error Control," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–13, 2010 (cit. on p. 4).

[12] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. 2004 (cit. on p. 5).

[13] K. Lee, S.-J. Lee, and H.-J. Yoo, "Low-power network-on-chip for high performance SoC design," *IEEE Transactions on Very Large Scle Integration (VLSI) Systems*, vol. March, 2006 (cit. on p. 5).

[14] X. Ma, "A summary of the routing algorithm and their optimization,performance," *Personal essay for the principles and practices of interconnection network*, pp. 1–9, 2024 (cit. on p. 5).

[15] J. Hu and R. Marculescu, "DyAD: Smart routing for networks-on-chip," *DAC*, pp. 260–263, 2004 (cit. on p. 5).

[16] J. Fang, D. Zhang, and L. Xiaqing, "ParRouting: An Efficient Area Partition-Based Congestion-Aware Routing Algorithm for NoCs," *Micromechanics*, vol. 11, pp. 1–17, 2020 (cit. on p. 5).

[17] S. J. Luneque, N. Nedjah, and L. de Macedo Mourelle, "Routing for applications in NoC using ACO-based algorithms," *Applied Soft Computing*, vol. 13, pp. 2224–2231, 2013 (cit. on p. 5).

[18] B. Walshe, *What is AMBA?* Dec. 2014. [Online]. Available: https://community.arm.com/arm-community-blogs/b/architectures-and-processors-blog/posts/what-is-amba?utm_source=chatgpt.com (visited on 09/02/2025) (cit. on p. 6).

[19] ARM, *AMBA® AXI-Stream*, 2021. (visited on 09/02/2025) (cit. on p. 6).

[20] ARM, *AMBA® AXI™ and ACE™ Protocol specification*, 2021. (visited on 09/02/2025) (cit. on p. 6).

[21] A. Inc., *Xilinx AXI-Based IP Overview*, 2025. [Online]. Available: https://www.aldec.com/en/support/resources/documentation/articles/1585/print_page?utm_source=chatgpt.com (visited on 09/02/2025) (cit. on p. 6).

[22] A. Ltd., *Introduction to AMBA AXI4*, 2020. [Online]. Available: https://developer.arm.com/-/media/Arm%20Developer%20Community/PDF/Learn%20the%20Architecture/102202_0100_01_Introduction_to_AMBA_AXI.pdf?revision=369ad681-f926-47b0-81be-42813d39e132 (visited on 09/02/2025) (cit. on p. 7).

[23] S. St. Micheal, *Introduction to the Advanced Extensible Interface (AXI)*, Oct. 2019. [Online]. Available: https://www.allaboutcircuits.com/technical-articles/introduction-to-the-advanced-extensible-interface-axi/?utm_source=chatgpt.com (visited on 09/02/2025) (cit. on p. 8).

[24] T. A. of Verification, *Understanding with AXI Protocol and Cache Coherency*, Jun. 2021. [Online]. Available: https://theartofverification.com/understanding-with-axi-protocol-and-cache-coherency/?utm_source=chatgpt.com (visited on 09/02/2025) (cit. on p. 8).

[25] H. Rhim and M. Simic, *What Is Quality of Service in Networking?* Mar. 2024. [Online]. Available: https://www.baeldung.com/cs/quality-of-service? (visited on 09/03/2025) (cit. on pp. 9, 10).

[26] HPE Juniper Networking, *What is quality of service?* [Online]. Available: https://www.juniper.net/us/en/research-topics/what-is-qos.html? (visited on 09/03/2025) (cit. on pp. 9, 10).

[27] Paloalto Networks, *What is Quality of Service?* [Online]. Available: https://www.paloaltonetworks.com/cyberpedia/what-is-quality-of-service-qos (visited on 09/03/2025) (cit. on pp. 9, 10).

[28] Fortinet, *What Is Quality Of Service (QoS) In Networking?* [Online]. Available: https://www.fortinet.com/resources/cyberglossary/qos-quality-of-service?utm_source=chatgpt.com (visited on 09/03/2025) (cit. on p. 9).

[29] A. Bruno and S. Jordan, *WAN Availability and QoS*, Apr. 2024. [Online]. Available: https://www.ciscopress.com/articles/article.asp?p=3192413&seqNum=7&utm_source=chatgpt.com (visited on 09/03/2025) (cit. on p. 10).

[30] NetworkLessons, *Introduction to RSVP*. [Online]. Available: https://networklessons.com/quality-of-service/introduction-to-rsvp?utm_source=chatgpt.com (visited on 09/03/2025) (cit. on p. 10).

[31] V. Jain *et al.*, "PATRONoC: Parallel AXI Transport Reducing Overhead for Networks-on-Chip targeting Multi-Accelerator DNN Platforms at the Edge," Jul. 2023 (cit. on p. 11).

[32] T. Fischer, M. Rogenmoser, M. Cavalcente, G. Frank K., and B. Luca, "FlooNoC: A Multi-Tbps Wide NoC for Heterogeneous AXI4 Traffic," Jun. 2023 (cit. on p. 11).

[33] T. Fischer, T. Benz, G. Frank K., and L. Benini, "FlooNoC: A 645 Gbps/link 0.15 pJ/B/hop Open-Source NoC with Wide Physical Links and End-to-End AXI4 Parallel Multi-Stream Support," Mar. 2025 (cit. on p. 11).

[34] T. Benz *et al.*, "AXI-REALM: A Lightweight and Modular Interconnect Extension for Traffic Regulation and Monitoring of Heterogeneous Real-Time SoCs," Nov. 2023 (cit. on p. 11).

[35] T. Benz *et al.*, "AXI-REALM: Safe, Modular and Lightweight Traffic Monitoring and Regulation for Heterogeneous Mixed-Criticality Systems," Jul. 2025 (cit. on p. 11).

[36] W.-C. Tsai, H.-E. Lin, Y.-C. Lan, and S.-J. Chen, "Anticipative QoS Control: A Self-Reconfigurable On-Chip Communication," *Micromachines*, vol. 13, no. 10, p. 1669, 2022 (cit. on p. 11).

[37] C. Liang, T. Benz, A. Ottaviano, A. Garofalo, L. Benini, and D. Rossi, "Towards Reliable Systems: A Scalable Approach to AXI4 Transaction Monitoring," Jan. 2025 (cit. on p. 11).

[38] N. Jiang *et al.*, "A Detailed and Flexible Cycle-Accurate Network-on-Chip Simulator," *Proceedings of the 2013 IEEE International Symposium on Performance Analysis of Systems and Software*, 2013 (cit. on p. 19).

[39] gem5, *Gem5*. [Online]. Available: `https://www.gem5.org` (visited on 09/05/2025) (cit. on p. 19).