NextGen Scientific Publication
American International Journal of Computer Science and Technology

Volume 6 Issue 1, Pg. No. 120-132, AIJCST-V6I1P112, 2024
https://doi.org/10.63282/3117-5481/AIJCST-V6I1P112

Original Article

Network-on-Chip Architecture Generation Using Arteris FlexNoC

***VenkataKrishna Brahmam Pogadadanda**

Senior RTL Design Engineer at AMD-Xilinx, USA.

Abstract:

Network-on-Chip (NoC) is now the leading communication infrastructure for the contemporary System-on-Chip (SoC) designs as complexity of multicore processors and heterogeneous computing platforms keeps increasing. Conventional bus-based interconnects have several drawbacks like the lack of scalability, high latency, limited bandwidth, and low power efficiency which make them unsuitable for handling the increasing communication needs of advanced semiconductor systems. Hence, scalable and reconfigurable NoC architectures have been widely adopted to facilitate the efficient exchange of data between processing elements, memory units, and peripherals. This paper focuses on the generation of NoC architecture using Arteris FlexNoC which is a commercial tool that is very popular for automated interconnect design and optimization in SoC development. Our proposed method details the definition of system requirements, configuration of network topology, selection of communication protocol, and the generation of optimized interconnect architectures within FlexNoC's automated design environment. In addition, the paper discusses how the choice of topology, routing schemes, and parameter settings can influence the overall performance of the system. Simulation results show that opting for Arteris FlexNoC as a design platform can significantly cut down design complexities, increase scalability, and improve communication efficiency while at the same time reducing latency and power consumption in comparison to old-style interconnect methods. Besides, the automatic generation process allows fast design space exploration and easy verification, so it is a good fit for complicated SoC applications. The results demonstrate the capability of FlexNoC in producing reliable, high-efficiency NoC architectures for embedded and computing systems of tomorrow. To summarize, the proposed method.

Keywords:

Network-on-chip (NoC), Arteris Flexnoc, System-on-chip (Soc), Interconnect Architecture, AMBA AXI, NoC Optimization, Performance Analysis, Hardware Design Automation.

Article History:

Received: 02.12.2023

Revised: 04.01.2024

Accepted: 17.01.2024

Published: 28.01.2024

1. Introduction

1.1. Background and Overview

Technological progress in semiconductors has enabled the consolidation of various processing elements, memories, and peripherals into one piece of silicon, leading to the rise of very complicated System-on-Chip (SoC) designs. Some time ago SoC designs, the main form of communication between different modules was through shared bus-based architectures. A reason that bus-based communication methods were simple and cheap for small-scale systems is that they become inefficient when the number of integrated cores and functional units increases. Limitations of traditional buses include high latency, bandwidth bottlenecks, poor scalability, and increased power usage when multiple devices try to communicate simultaneously. These challenges have inspired researchers and designers to find more efficient communication infrastructures that can handle the demands of modern computing.

Network-on-Chip (NoC) is considered as one of the best alternatives to traditional interconnection methods for eliminating the communication issues of conventional interconnect methods. NoCs use the idea of computer networks and replace shared buses by packet-based communication networks that support scalable and parallel data transfer among components. The communication paths in NoC systems are created through routers, switches, and links, which contribute to the increase of throughput and the decrease of congestion. NoC architectures support different topologies including mesh, torus, ring, and tree structures, thus providing designers with the opportunity to select the one best aligned with their performance requirements.

1.2. Challenges in NoC Design

Designing an efficient Network-on-Chip architecture is a very tough job because of the complexity of the modern SoCs. One of the biggest problems is scaling. When more cores and IP blocks are put together, a communication network must support a very large data traffic without any performance loss. Old types of interconnection architectures do not have enough scalability because communication bottlenecks increase with the system size and cause more latency and less throughput.

Latency and bandwidth constraints are indeed major NoC design concerns. For example, applications in artificial intelligence, machine learning, multimedia processing, and real-time embedded systems require extremely rapid and dependable communication among various processing units. NoC structures, if not well-designed, can cause long transmission delays and lead to the issue of insufficient bandwidth. Therefore, it becomes very important to select the correct routing algorithms and network topologies to achieve efficient data transfer.

Power consumption is another major issue that needs to be addressed in Systems on Chip (SoC) using Network on Chip (NoC). As communication networks are responsible for a large amount of chip resources, individual switching activities and excessive movements of data can be among the factors increasing dynamic and static power consumption. Besides, minimizing energy consumption while still ensuring top performance is very much at the top of design priorities, mainly for portable devices and those operating by batteries.

Coping with congestion and handling Quality of Service (QoS) demands make things even more complicated when implementing NoC. Network congestion, packet delay, and resource contention can all be formed when traffic is heavily loaded. Also, nowadays, most applications are not only satisfied with best effort communication but they also expect to have communication with guaranteed bandwidth and low latency. On top of that, critical tasks communication should be given high priority; as a result, elaborate QoS mechanisms within NoC facilities are a must.

1.3. Problem Statement

As the complexity of System-on-Chip (SoC) continues to increase, designing efficient interconnect communication systems has become a major challenge. The old bus-based interconnects are incapable of meeting the performance, scalability, and bandwidth requirements of modern multi-core and heterogeneous platforms. On the other hand, the architectures based on Network-on-Chip (NoC) are a good solution. However, manually designing and optimizing NoC topologies is a very complex and time-consuming task. The designers have to make a thorough choice of routing strategies, communication protocols, topologies, and quality-of-service parameters even as they tackle the power, latency, and congestion issues.

Manual NoC implementation poses a higher risk of design errors, challenges in verification, and possible inefficient resource use. Besides, traditional methods for interconnect design are quite rigid and not well suited to the rapidly changing SoC requirements. Thus, there is an urgent demand for automated tools that can create configurable and well-optimized NoC architectures while lowering design efforts and enhancing reliability.

Arteris FlexNoC offers a solution to these problems by giving an automated platform for NoC generation, customization, and verification. The platform not only allows the efficient exploration of different design options but also helps to easily integrate communication networks in complex SoCs. Hence, this paper aims at harnessing the power of Arteris FlexNoC for the development of scalable and efficient NoC architectures as a way to break free from the challenges posed by conventional interconnect design methods.

1.4. Motivation

The development of artificial intelligence, machine learning, autonomous systems, data analytics, and multimedia processing requires very complex System-on-Chip (SoC) architectures. Modern SoCs consist of different kinds of processing units like CPUs, GPUs, DSPs, accelerators, and memory subsystems all combined on a single chip. To reach high performance, low latency, and minimum power consumption, efficient communication among these units is very important. However, the traditional bus-based interconnects cannot meet these needs because of their limited scalability and communication bottlenecks, thus, the introduction of advanced Network-on-Chip.

Another factor that is leading us to carry out this study is the large expansion of the semiconductor industry and the consequent desire for shortening SoC development time. The high performing chips have to be designed, verified and deployed at a fast pace in order to stay competitive in these market conditions. Manual design of interconnects requires a lot of engineering time and is more susceptible to errors. So automated NoC generation tools are very helpful, as they speed up the development process and still keep the design reliable and efficient.

The demand for configurable and reusable interconnect Intellectual Property (IP) blocks has also grown significantly in recent years in the area of hardware design. The use of flexible network-on-chip (NoC) platforms gives the possibility to the designers to change a variety of aspects such as topologies, routing schemes, bandwidth, and protocols based on the needs of the application. Arteris FlexNoC is an implementation of such a platform which offers a robust set of tools for automatic NoC design, NoC optimization, and NoC verification. Through this platform, SoC designers are assisted to efficiently handle communication challenges of SoCs that utilize new technologies.

FlexNoC helps to greatly improve productivity by raising the level of automation, hitting a higher target of scalability, and making verification a lot easier. It is these benefits that inspire a thorough investigation of the generation of NoCs based on FlexNoC as a promising method for the emerging architectures of SoCs that are scalable and of high performance.

2. Literature Review

2.1. Traditional Interconnect Architectures

Back in the days before Network-on-Chip (NoC) architectures were introduced, most of the communications within System-on-Chip (SoC) designs were done through the traditional interconnect means, such as shared buses, crossbar switches, and point-to-point communication systems. These setups worked well for the first generation of integrated circuits where the number of processing elements and volume of communications were limited. However, as the semiconductor technology kept evolving and SoCs integration density became higher, these classical communication methods started showing several drawbacks.

Shared bus architecture was the earliest and the most widely used interconnect method in embedded systems and microprocessor-based designs. In this setup, many different components get the chance to communicate over a single shared communication channel. Despite the fact that bus-based systems are simple, cost-effective, and easy to implement, they have very serious scalability issues. The more the connected modules, the higher the bus contention, which eventually causes communication delays, lower bandwidth, and performance bottlenecks. On top of that, only one transaction can be done at a time, hence limiting the parallel communication.

Initially,crossbar interconnects aimed at improving communication efficiency by enabling simultaneous data transfers going to and coming from different components. A crossbar switch establishes dedicated paths between the senders and the receivers, which lessens contention while increasing throughput. However, crossbar architectures require a very large number of switches and links particularly if the system is enlarged, which leads to higher hardware complexity, more area usage, and more power consumption.

Point-to-point communication solutions come with offering direct connections between the modules involved in communication, thus delivering low latency, and high bandwidth. This way of communication results in better performance, yet in the case of large-scale systems, it turns out to be not feasible mainly due to excessive wiring complexity, routing congestion, and very limited scalability. The aforementioned issues of traditional interconnect architectures served as a catalyst for the concept of scalable and efficient communication infrastructures, namely Network-on-Chip.

2.2. Evolution of Network-on-Chip

Traditional interconnect architectures have their limitations that ultimately brought about the development of Network-on-Chip (NoC) which serves as a scalable communication solution for present-day System-on-Chip designs. Taking inspiration from computer networking, NoC brought packet-switched communication which facilitates the effective transfer of data between several processing elements that are all encapsulated in one chip. While shared bus architectures only allow one communication at a time, with NoC multiple components can communicate in parallel which not only increases bandwidth utilization and lessens congestion but also makes the system more scalable.

Packet- switched NoC systems break down data into smaller packets that get sent through routers and communication links. Each packet is equipped with routing data that will determine its path to the network independently. Such a method offers maximum flexibility and permits simultaneous communication transactions. Besides, using packets for communication generally results in a better sharing of resources and a higher degree of fault tolerance when compared with traditional interconnect ways. There are a lot of different NoC topologies proposed and implemented to focus on the optimization of communication performance according to application requirements while mesh topology being one of them. Mesh topology is easy to understand, regular, and easy to design for a larger number of components. Mesh networks consist of routers arranged in a grid pattern where every router is linked to its adjacent nodes. Torus topology is a step forward in mesh architecture, it connects edge nodes and thus it forms wrap-around links that lead to a reduction in communication distance and latency.

Tree-based topologies offer a hierarchical structure for communication that helps in lowering the routing complexity and enhancing bandwidth distribution. Ring topology connects routers in a circle thus resulting in lower hardware complexity and less area consumption for smaller systems. Routing techniques have a big share in making communication within NoC architectures efficient. Deterministic routing techniques transmit packets following a fixed route which results in predictable latency and a simpler implementation. On the other hand, deterministic routing may cause congestion if the traffic is heavy.

2.3. Existing NoC Design Methodologies

Significant changes have been made in the design methods of Network-on-Chip (NoC) architectures to accommodate the increasing complexity of modern SoC systems. At first, the development of NoCs was primarily a manual activity, in which engineers tailored routers, communication links, routing algorithms, and topologies to the application requirements. Manual production of NoCs, although giving the possibility of time large flexibility and wide customizations, was on the other hand time-consuming and prone to errors. Designers had to carefully tune parameters such as bandwidth allocation, latency, congestion control, and quality of service. Thus, the design process for large-scale systems became quite challenging.

One of the reasons automated NoC synthesis tools came to existence was as a remedy to these problems, to simply the process of architecture generation and optimization. Using these tools, designers can easily create interconnect topologies, specify routing protocols, and refine communication paths to meet performance criteria. Automated NoC design platforms not only can speed up the development cycle drastically but also result in more accurate and scalable designs. Moreover, they enable rapid scenario testing of different design options so that engineers can decide between power consumption, latency, throughput, and area utilization. Leading-edge NoC synthesis tools are wisdom-loaded with verification that help easily ensure compliance with the protocols and timing analysis.

Both FPGA and ASIC platforms are popular options to implement and evaluate NoC architectures. An FPGA implementation by nature and its properties like flexibility, reconfigurability, and ability to fast prototyping make it primarily fit for research and validation of a new design stage in an academic environment. Developers testing various network topologies, routing protocols and methods of communication may use the FPGA platform as it does not involve the huge manufacturing cost of the ASIC fabrication. Nonetheless, compared to the work on dedicated hardware, the performance of an FPGA implementation decreases and power consumption increases.

3. Proposed Methodology

3.1. Overview of Proposed Framework

This methodology aims at producing and refining Network-on-Chip (NoC) designs automatically through the use of the Arteris FlexNoC platform. It helps making the operation of creating interconnections for very complex System-on-Chip (SoC) a more straightforward task while at the same time enhancing the communication performance, scalability, and power efficiency of the resulting designs. The workflow presented here brings together requirement analysis, traffic characterization, topology preference, routing set-

up, and performance tweaking into a single automated work environment. Such a strategy reduces the manual design workload and makes it possible to quickly evaluate various NoC configurations catering to the needs of heterogeneous multi-core systems.

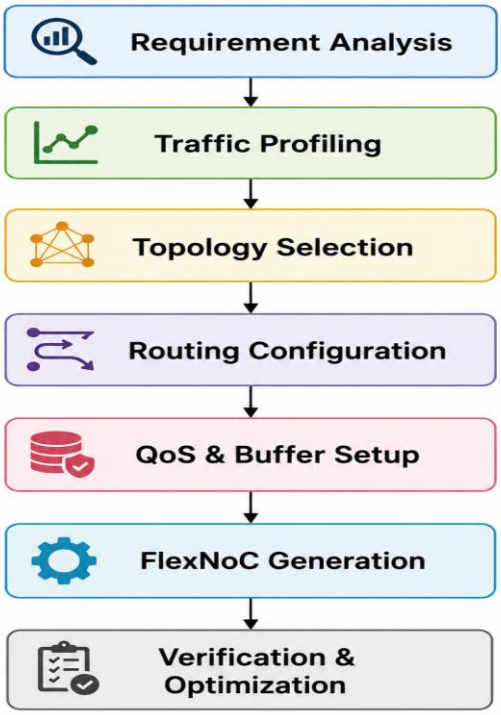

Figure 1. Proposed NoC Architecture Generation Framework

A process starts with documenting communication needs of processing cores, memory controllers, and peripheral modules in a given System-on-Chip (SoC). The network on silicon (NoC) topologies and routing algorithms which are most suitable for the design are selected with the help of the application trace and bandwidth requirements chart in the flexNoc configuration environment. In addition, the system provides Quality of Service (QoS) policies, buffer control, clock-domain crossing mechanisms, and protocol adaptation to enable communication among IP blocks that operate at different clock frequencies.

In order to guarantee seamless integration with the full SoC design flow, standard interfaces e.g., AMBA AXI and APB protocols are employed. The NoC design, in turn, is checked and optimized for aspects like latency, throughput, congestion management, and power consumption. The lengthy process is drastically reduced with the help of the system introduced, at the same time, providing scalable and reusable interconnect solutions for today's SoC applications including AI accelerators, multimedia systems, and embedded computing platforms.

3.2. Arteris FlexNoC Design Environment

Arteris FlexNoC enables the automatic creation, configuration, and fine-tuning of NoC architectures with an easy-to-use graphical user interface that allows users to quickly assemble communication networks, integrate IP cores, and define routing policies suitably. This GUI makes it easy to build scalable network-on-chip architecture by straightforward dragging and dropping of components, choosing topology, setting protocol, and adjusting performance parameters. Such a visual method of design eliminates a lot of manual coding work and greatly increases the speed of the interconnect development.

With FlexNoC, users can integrate quite different types of IP blocks like processors, memory controllers, DMA engines, accelerators, and peripheral interfaces, according to their needs. They just show the relationships between masters and slaves in communications. In addition, they are able to set the traffic priorities based on the requirements of the software. The software system takes care of interconnect designing and guarantees that the protocols and timing for each communication path are preserved.

FlexNoC has a real advantage in its compatibility with various industry-standard AMBA protocol families, including AXI, AHB, and APB. By providing support for the AXI protocol, the network opens up the possibility of implementing burst-based communication, which is the best fit for applications that handle large amounts of data. On the other hand, APB, being a low-power protocol, is great for communication between peripherals. Besides this, the platform makes it possible to convert protocols, deal with different clock domains, and manage available bandwidth among other things that are usually required in the integration of heterogeneous SoCs.

FlexNoC offers a comprehensive set of verification and analysis tools to assist designers in conceptualizing and optimizing their network in terms of latency, throughput, congestion, and power consumption. The reliability of the design is additionally enhanced with the aid of automated design rule checking as well as protocol verification. Taking all these features into account, FlexNoC is extremely capable of developing scalable, configurable, and high-performance NoC architectures in both FPGA and ASIC-based systems.

Table 1. Features and Capabilities of the Arteris FlexNoC Design Environment

Feature	Description	Benefit
GUI-Based Configuration	Visual NoC topology generation and customization	Simplifies interconnect design process
AXI/APB Protocol Support	Supports industry-standard AMBA communication protocols	Ensures compatibility with SoC components
Automated Routing	Automatic generation of optimized routing paths	Reduces manual configuration effort
QoS Management	Traffic prioritization and bandwidth allocation	Improves communication reliability
Clock Domain Crossing	Supports communication between different clock domains	Enables synchronization across modules
IP Integration	Easy integration of processors, memories, and peripherals	Enhances design flexibility
Verification Support	Built-in protocol and performance verification	Improves design accuracy and reliability
Scalability	Supports large multi-core and heterogeneous SoCs	Suitable for advanced high-performance systems

3.3. NoC Architecture Generation Process

Using Arteris FlexNoC for developing the NoC architecture is a set of several separate steps which in total lead to the production of the scalable and well-optimized interconnect structures for the modern SoC. Firstly the requirement analysis is taking place which is the phase of detailed study of the communication characteristics that will happen between the different IP blocks. The parameters such as the bandwidth of the communication, the latency limits, the typical traffic patterns, the protocol compatibility, and the clock frequencies are only some of the ones that the NoC configuration will be based on. Without a doubt, requirement analysis is the main phase for finding out the communication needs of an application.

After requirement analysis, traffic profiling is initiated to examine the SoC masters and slaves communication. In fact, traffic profiling identifies the high-bandwidth communication paths, the burst transactions, and the congestion-prone areas. Understanding the traffic characteristics and then choosing the suitable topologies, and finally optimizing the resource allocation will lead to the desired higher communication efficiency.

Topology selection is probably the most essential step for generating a NoC. Depending on the usage demands, one can pick between mesh, ring, tree, or even custom hierarchical topologies. Usually, mesh topologies are preferred for multi-core systems that can be scaled up due to their orderly structure and a routing that is quite efficient. The network topology chosen is the main factor that affects the latency, bandwidth allocation, routing complexity, and hardware overhead of the system.

Once a topology had been selected, routing configuration decides the routes that data packets will take through a networkFlexNoC offers choices in deterministic and adaptive routing methods. On the one hand, deterministic routing guarantees that communications occur in a specific, fixed way. On the other hand, adaptive routing is empowered to reroute the traffic for congestion avoidance and throughput enhancement. In fact, an optimized routing configuration is what permits packets to be delivered correctly and timely so as to minimize communication delays to the greatest extent.

Afterwards, traffic flow management within routers and communication links is done through packet buffering and bandwidth Packet buffering and bandwidth allocation are arranged after that to control traffic flow inside routers and communication links. Buffers hold packets for a short period of time to allow for congestion to clear and to minimize packet loss. High priority is given to critical communication channels by bandwidth allocation methods, and they also ensure that network resources are utilized effectively.

Configuring clock-domain crossing is a requirement in heterogeneous SoCs where different modules operate at different clock frequencies. FlexNoC will automatically add synchronization mechanisms to allow data to be transferred safely between asynchronous domains without affecting timing reliability.

The configuration of Quality of Service (QoS) is often viewed as a vital tool for managing traffic priorities and guaranteeing the performance of communication in applications sensitive to latency. By using QoS methods, the network bandwidth gets segmented based on the importance of the traffic, and this segmentation prevents the traffic of lower priority from disturbing the communication channels of higher priority.

In the end, the NoC architecture that has been produced is checked and analyzed for performance in order to determine throughput, latency, congestion, and power consumption. The design is continuously refined until the desired performance goals are met. The automated generation process that FlexNoC offers greatly decreases the complexity of designing, at the same time that it enhances the scalability and the general efficiency of the communication.

4. Case Study

4.1. Experimental Setup

Our goal was to figure out how well the Network-on-Chip that we came up with, using the Arteris FlexNoC platform, would perform. We set up both the implementation and the simulation environment using a combination of hardware and software tools that are commonly used nowadays in SoC design workflows. The experiments were conducted on a high-performance computer having an Intel Core i7 processor, 16 GB RAM, and a Linux-based operating system in order to allow efficient simulation and synthesis operations. As for the software environment, there were the Arteris FlexNoC design suite, some RTL simulation tools, and waveform analysis utilities for performance evaluation and verification.

Automated NoC generation and interconnect configuration were performed through the use of Arteris FlexNoC 5.x. With this platform, it was feasible to create different topologies, set protocols, assign bandwidth, ensure QoS, and find the best routing paths, within a single environment. For functional simulation and verification, we used simulation tools that are standards in the industry, like Synopsys VCS and ModelSim, to study packet transfer behavior, timing characteristics, and protocol compliance. Power estimation and synthesis analysis were performed using EDA tools which can measure switching activity and interconnect resource usage.

The SoC environment that was constructed contained a number of processing elements including processor cores based on ARM, memory controllers, DMA modules, and peripheral interfaces that were connected through the generated NoC structure. The communication between master and slave devices was implemented by using the standard AMBA AXI and APB protocols. differential traffic scenarios were developed to evaluate the system throughput, latency, congestion handling, and power efficiency under various workload conditions.

Table 2. Experimental Setup and Simulation Environment

Parameter	Configuration	Purpose
Processor	Intel Core i7	Supports simulation and synthesis tasks
RAM	16 GB	Handles large NoC simulations

Operating System	Linux Ubuntu	Provides stable development environment
FlexNoC Version	FlexNoC 5.x	Automated NoC generation
Simulation Tool	ModelSim / Synopsys VCS	Functional verification and timing analysis
Protocol Support	AXI/APB	Inter-IP communication
Target Platform	Multi-core SoC	Performance evaluation
Verification Method	RTL Simulation	Validation of NoC functionality

4.2. System Design Implementation

The main system design implementation aimed at creating a scalable and configurable NoC architecture, which is a network on chip, capable of supporting a multi-core System-on-Chip environment. The chosen SoC configuration included ARM-based cores, shared memory modules, DMA units and peripherals that were interconnected with each other through the Arteris FlexNoC network. The design was based on fast and efficient communication among the components with a very large bandwidth and very low latency, trying not to waste hardware resources unnecessarily. The developed interconnect could readily support the different processing elements and memory units communicating with each other simultaneously, hence accommodating various traffic scenarios.

Since a mesh network is a regular structure that is easy to scale and also has routing efficiency, it was planned as the network for the NoC. In the designed system, the arrangement of routers was made in a 2D grid layout and each node interacted with its adjacent routers via communication links that were dedicated. The mesh network ensured a fair sharing of bandwidth and also a reduction in congestion when compared to the old bus-based systems. During implementation, traffic characteristics were thoroughly analyzed to make communication as efficient as possible. Different types of traffic, such as burst and continuous, were considered to model real-world SoC workloads. Communication paths with very large bandwidth between processors and memory modules were given highest priorities using the QoS policies from FlexNoC. Adaptive routing strategies were set up so that the traffic could be automatically spread out and the network congestion minimized at times of extremely heavy usage.

In the FlexNoC environment, several configuration parameters were adjusted to achieve the maximum performance possible. Router buffer depth, link bandwidth, arbitration policies, packet size, and virtual channel allocation were among the parameters that were optimized to meet the communication requirements. In order to communicate asynchronously (without the need to synchronize clocks) between IP blocks working at different frequencies, clock-domain crossing modules were included. Protocol support for AXI enabled complex burst transactions at high speeds for data-intensive applications, while APB interfaces provided communication with peripherals which operate at low speeds.

Following RTL generation, design integration was carried out inside SoC environment, and verified through simulation tools. Based on the performance metrics like latency, throughput, congestion behavior, and power consumption, the effectiveness of newly crafted NoC design was validated. Implementation has shown that it is not only scalable, but also handles communication efficiently and reduces design complexity when compared with traditional interconnect approaches

4.3. Simulation and Validation

The simulation and validation activity was carried out to evaluate the correctness and efficiency of the generated NoC design when running different scenarios. Functional verification via RTL simulation techniques was used to identify the exact packet transfer between the connected IP blocks. ModelSim and Synopsys VCS simulation tools were employed for validating routing operations, protocol compliance, correct packet delivery, and clock domain synchronization. Different traffic patterns like random, burst, and uniform were generated for the analysis of the network's robustness and communication reliability.

Whereas, throughput demonstrated the amount of data being transferred over the net at any one time. A network on chip system based on a mesh-architecture was really good at using bandwidth and allowed different communications to go on at the same time in parallel, which was a very different result from an old shared-bus type of system. Adaptive routing strategies also helped a great deal in maintaining a constant throughput level even under heavy traffic conditions, as they controlled the network congestion by changing the routes.

Latency analysis has been done to check the communication delay from the source node to the destination node. The simulation results showed that with optimized routing configurations and efficient buffer allocation the packets transfer delay was greatly reduced. Besides, by using QoS-based traffic prioritization latencies for the time-critical communication channels were further optimized.

Power estimation was realized to look into the energy efficiency of the design interconnect architecture. Simulation-driven switching activity has been utilized to determine the dynamic power consumption inside the routers and communication links. The outcomes have proved that the implemented optimization methods have proficiently eliminated the unnecessary data movement and switching activities overhead which consequently led to the dropping of the power consumption below that of the traditional interconnect methods.

In conclusion, the simulation and validation work have corroborated that the suggested FlexNoC-based architecture is capable of offering a scalable, dependable, and high-performance communication which is well suited for advanced multi-core SoC systems.

5. Results and Discussion

5.1. Performance Metrics

The generated Network-on-Chip architecture was reviewed based on a few very important performance factors such as latency, throughput, area utilization, power consumption, and scalability. These factors were measured to find out how well the proposed FlexNoC-based interconnect architecture can meet modern multi-core SoC communication requirements.

Latency means the time it takes for the data to travel from the source node to the destination node. The simulation results indicated that the proposed NoC architecture could have much lower latency than traditional bus-based interconnects. Adaptive routing methods and the choice of optimal topology helped reduce packet traversal time and also cut down delays due to congestion. The proper allocation of buffers made the communication more responsive even under heavy load situations.

Throughput analysis is a technique to find out how many data were actually transmitted over the network in a given time interval. As the mesh-based architecture was implemented, it revealed a very high throughput level because it could support a parallel communication mode and utilize bandwidth efficiently. While with the traditional shared buses only one pair of devices could communicate at a time, the NoC structure allowed data transfers between multiple processing elements at the same time not only increasing the overall communication performance but also meeting the resource constraints.

Area usage is one of the parameters that reflect the efficiency of an architecture. We considered router count, interconnect complexity, and hardware resource consumption while estimating area utilization. It is true that the NoC architecture with its additional routing resources contributed more than bus systems at a simpler level, however, by topology generation through optimization and configurable routing structures, the designs could achieve a good hardware utilization balance. Furthermore, the FlexNoC's modular design feature made it easier to integrate IP and helped lower the amount of hardware redundancy that is not needed.

Analysis of power consumption showed that the switching activity was down while the communication paths inside the created architecture were also optimized. With the help of power-aware routing techniques, the amount of redundant data movement as well as dynamic power consumption were kept to a minimum. The design was still able to deliver reliable output even though it was under some energy limitations.

By carrying out a scalability test, it was proved that the suggested structure was capable of handling more processing elements and communication nodes thereby showing minor decline in performance. As such, this approach is well suited to the production of large-scale heterogeneous systems-on-chip.

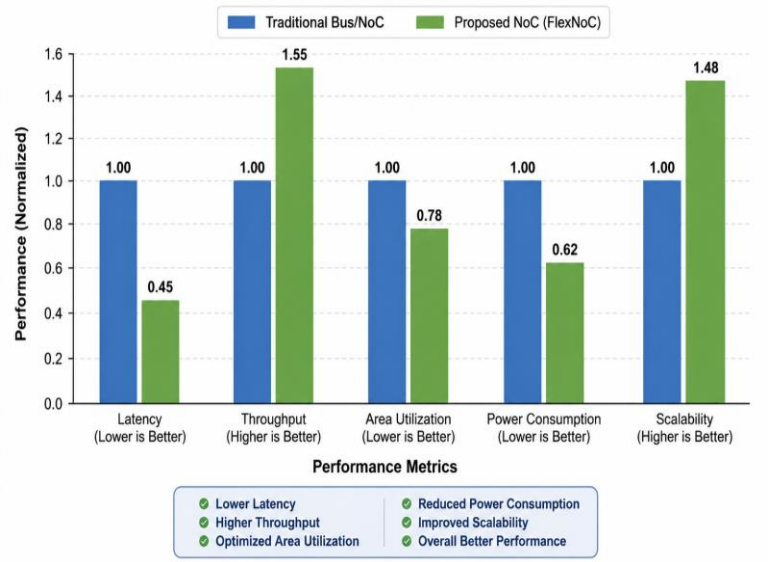

Figure 2. Performance Comparison of Proposed NoC Architecture

5.2. Comparative Analysis

The newly proposed NoC architecture developed with Arteris FlexNoC was compared with the conventional interconnect approaches and NoC design methodologies in order to assess its performance, scalability, design complexity, and communication efficiency.

In comparison to traditional bus-based NoC systems, the proposed approach showed a significant increase in throughput and a reduction in latency. Typical shared bus architectures experience communication bottlenecks because several components have to compete for a single communication channel. This leads to higher latency and limited scalability as the number of processing elements grows. On the other hand, the FlexNoC-based architecture proposed here allows parallel communication among several IP blocks, which considerably raises bandwidth usage and lowers congestion.

The little debate we have here is about the advantages of FlexNoC automated generation of NoC in comparison to manual NoC design. There were big changes in the productivity of design and optimization of the efficiency. The manual interconnect generation is a whole engineering. It involves decision on the topology, routing setup, protocol integration, and verification.

The topologies configuration, routing setup, etc., are very time-demanding and error-prone. FlexNoC automated lots of the architecture generation including routing configuration, bandwidth allocation, QoS management, and protocol adaptation. The overall designing time was reduced through this, and the accuracy and scalability were increased as well.

Aside from that, other existing NoC tools were put head-to-head with Arteris FlexNoC on configurability, protocol support, optimization capabilities, and industrial applicability. Most of the academic NoC tools mainly focus on research and experimental works and offer limited support for large scale commercial ASIC implementation. On the other hand, FlexNoC brings industry standard features like AMBA AXI/APB compatibility, support for clock-domain crossing, advanced QoS, and verification capabilities integration.

Moreover, the proposed architecture exhibited better congestion control because it employed adaptive routing mechanisms and traffic-aware optimization techniques. Also, resource utilization was significantly more efficient since the fabricated topology was closely aligned to the application-specific communication requirements. Acknowledging that FlexNoC incurs some extra hardware complexity compared to simple bus architectures, the enhancements in scalability, throughput and communication reliability brought about by FlexNoC more than compensate for the additional costs.

In brief, the comparative study verified that the proposed method offers a more scalable, configurable, and efficient solution for modern multi-core SoC communication than both the conventional interconnect methods and several existing NoC generation approaches.

5.3. Discussion of Results

The simulation and performance results have revealed that the proposed method of NoC architecture design using Arteris FlexNoC has delivered a great touch of effectiveness. Large scale improvements in communication efficiency, throughput, and latency cutting down, alongside scalability, were the areas that caught our attention when comparing to the traditional interconnect architectures. The resulted NoC architecture was allowed to support concurrent communication between multiple processing elements without compromising the network's performance stability even under diverse traffic scenarios.

Among many enhancements, the most considerable one is the latency trimming done at the communication level, thanks to the optimized routing and wisely chosen topology. Adaptive routing techniques brought about the possibility of shifting traffic on the fly which means that instances of congestion can be altogether avoided even when the load is quite heavy. QoS is a space where latencies less sensitive to disruption bring their game, however our system goes beyond by highly securing the allocation of bandwidth for the most critical operations.

Due to the implementation of effective packet-based communication and the ability to perform concurrent data transfers, throughput performance was also greatly enhanced. While in a typical bus-based system the performance gets limited due to communication contention, our architecture allowed for a number of simultaneous communication paths, thus increasing overall utilization of the network bandwidth.

Power optimization strategies led to decreased switching activity and energy consumption of communication links and routers. This enhancement is especially useful for portable devices and systems having energy limitations. Besides, the designed architecture proved to be efficient in terms of hardware resource utilization thanks to tailored generation of the topology and the allocation of buffers which were optimized.

Still, even with these new changes, some compromises were made during the project execution. More routers and sophisticated routing techniques enabled the system to scale better and allowed it to have a higher throughput, but on the other hand, it also resulted in increased hardware complexity and area overhead. Likewise, congestion control was definitely improved by adaptive routing methods but the increased complexity of routing logic and higher verification efforts also came along with it.

6. Conclusion and Future Scope

6.1. Conclusion

This research was aimed at the creation and enhancement of Network-on-Chip (NoC) designs via the Arteris FlexNoC platform for contemporary System-on-Chip (SoC) products. The main point of the study was to produce an interconnect structure that is not only scalable and configurable but is also able to relieve the restrictions of old communication systems based on buses.

The article traced the evolution of interconnecting component architectures, the challenges of Network on Chip (NoC) design, and the need for automation in managing the complexities of multi-core and heterogeneous SoCs. The authors presented a methodology that included the identification of requirements, analysis of running traffic, choice of topology, setting routing, Quality of Service (QoS) arrangement, and the final tuning with the FlexNoC tool. The final product was measured against a set of main KPIs like delay, bandwidth, area usage, energy requirement, and ability to scale.

6.2. Future Scope

While the FlexNoC-based NoC architecture presented in this paper has achieved a great deal of communication performance and scalability, new ideas and research work can be found by the authors. The focus of the future work should be in using advanced optimization and traffic managing systems that can respond to the adaptability and efficiency requirements of NoC in increasingly complex SoC environments.

There is a potential for using AI and machine learning techniques in NoC automated optimization. For one thing, AI-driven algorithms can monitor communication patterns, forecast congestion, and re-route on the fly so as to boost throughput and cut down on latency. Traffic prediction models based on machine learning can also be useful in locating choke points well ahead of time so as to facilitate proper traffic management and resource allocation planning.

Another important aim of research studies is the developing of dynamic and adaptive routing strategies capable of efficiently dealing with real-time traffic changes. Unlike fixed routing schemes, adaptive communication networks have the capacity to constantly identify the most efficient paths for packet transmissions according to the fluctuations in the network conditions, therefore they can improve communication reliability and maximize the use of network resources.

Future Network-on-Chip (NoC) systems are likely to feature chiplet-based architectures, in which several smaller chips are linked together to build large heterogeneous computing systems. To enhance scalability and modularity in advanced semiconductor designs, FlexNoC can be combined with chiplet communication frameworks.

References

[1] Roy, Ankita, et al. "A Novel Network on Chip Architecture for FPGA Smart NIC." *2023 IEEE Women in Technology Conference (WINTECHCON)*. IEEE, 2023.

[2] Nightingale, Andy. "Optimize SoC Design with a Network-on-Chip Strategy: Utilizing physically aware interconnect IP from trusted third-party vendors can reduce design time and increase productivity." *Electronic Design* 71.4 (2023): 28-30.

[3] Suryadevara, S. S. K., & Shaik, K. (2023). Real-Time Anomaly Detection and Attack Mitigation for Cloud-Based Content Delivery Paths Using AI. *International Journal of Emerging Research in Engineering and Technology*, *4*(1), 175-185. https://doi.org/10.63282/3050-922X.IJERET-V4I1P119

[4] Schirrmeister, Frank, Rocco Jonack, and Michael Frank. "Addressing DRAM Performance Analysis Challenges for Network-on-Chip (NoC) Design." *Proceedings of the International Symposium on Memory Systems*. 2023.

[5] Gaddam, R. R. (2021). Hermetic ML Environments using Conda-Lock and Docker. American International Journal of Computer Science and Technology, 3(4), 22-34. https://doi.org/10.63282/3117-5481/AIJCST-V3I4P103

[6] Shiramalla, Rupesh. "Design of a Unified API Interface Using Workato for Cross-Platform Data Orchestration Between Salesforce and Oracle ERP." *International Journal of Emerging Trends in Computer Science and Information Technology* 3.1 (2022): 157-168.

[7] Singla, Garbí, Félix Tobajas, and Valentín de Armas. "Network-on-chip emulation framework for multimedia SoC development." *VLSI Circuits and Systems VI*. Vol. 8764. SPIE, 2013.

[8] Katangoori, Sivadeep, and Sushil Deore. "Edge-Cloud Hybrid Data Pipelines: Architectures for Federated Analytics and Learning". *American International Journal of Computer Science and Technology*, vol. 4, no. 3, May 2022, pp. 20-34, https://doi.org/10.63282/3117-5481/AIJCST-V4I3P103.

[9] Muppaneni, R. K. (2023). AI-Driven Forecasting in Dynamics 365 Sales: What Businesses Need to Know. International Journal of AI, BigData, Computational and Management Studies, 4(1), 168-176. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V4I1P117

[10] Janac, K. Charles. "Network-on-Chip (NoC): The Technology that Enabled Multi-processor Systems-on-Chip (MPSoCs)." *Multi-Processor System-on-Chip 1: Architectures* (2021): 195-225.

[11] Allenki, S. S. (2023). Applying Cloud Security Best Practices in Regulated Environments. *American International Journal of Computer Science and Technology*, *5*(3), 48-60. https://doi.org/10.63282/3117-5481/AIJCST-V5I3P105

[12] Takkalapally, D. (2023). HoloSearchAI: AI-Driven Latency Optimization Framework for Distributed Search Systems. *International Journal of Emerging Trends in Computer Science and Information Technology*, *4*(3), 217-227. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I3P122

[13] Abdelfattah, Mohamed S., and Vaughn Betz. "Embedded Networks-on-Chip for FPGAs." *Reconfigurable Logic: Architecture, Tools and Applications* (2018): 149-184.

[14] Parakala, A. (2023). Vendor Highlights – IoT, AI, and Process Mining. *International Journal of Emerging Trends in Computer Science and Information Technology*, *4*(4), 135-146. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I4P115

[15] Kommuru, Madhurima. "Concurrency is hard: Java vs Python pitfalls you can't ignore." *International Journal of Modern Research in Science & Engineering* 5.2 (2022): 01-18.

[16] Kumar Doodala, A. N. (2023). Offline-First Android Architecture for waste management in low connectivity zones. *International Journal of Emerging Trends in Computer Science and Information Technology*, *4*(1), 201-209. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I1P121

[17] COURTOIS, M. Bernard, et al. *ASIC Design Methodology for 3D NOC Based Heterogeneous Multi Processor on Chip*. Diss. Université Henri Poincaré, 2013.

[18] Muppaneni, R. K. (2023). Low-Code Revolution: How Power Platform Extends Dynamics 365 Capabilities. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *4*(3), 162-171. https://doi.org/10.63282/3050-9262.IJAIDSML-V4I3P119

[19] Vanapalli, Satyendra Kumar. "Data-Driven Decision Making: Integrating AI Tools with CRM Platforms." *International Journal of Applied Data Science & Modern Computing* 6.1 (2023): 01-21.

[20] Katangoori, S., & Katangoori, A. (2023). Intelligent ETL Orchestration with Reinforcement Learning and Bayesian Optimization. *International Journal of Emerging Research in Engineering and Technology*, *4*(4), 208-219. https://doi.org/10.63282/3050-922X.IJERET-V4I4P123

[21] Balamurugan, K., et al. "Roadmap for machine learning based network-on-chip (M/L NoC) technology and its analysis for researchers." *Journal of Physics Communications* 6.2 (2022): 022001.

[22] Srigadde, B. R. (2023). Creating Object Quick Actions with Lightning Web Components. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *4*(2), 167-180. https://doi.org/10.63282/3050-9262.IJAIDSML-V4I2P118

[23] Shiramalla, R. (2022). Predictive Record Assignment Engine in Salesforce using LWC and Einstein AI. *International Journal of AI, BigData, Computational and Management Studies*, *3*(3), 147-159. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V3I3P117

[24] Fatollahi-Fard, Farzad, et al. "OpenSoC Fabric: On-chip network generator." *2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 2016.

[25] Vppalapati, M. (2023). When Identity Decisions Throttle Data Movement. *International Journal of Emerging Research in Engineering and Technology*, *4*(3), 160-170. https://doi.org/10.63282/3050-922X.IJERET-V4I3P117

[26] Muppaneni , K. (2023). Virtual DOM vs Real DOM: Performance Benchmarks. *International Journal of AI, BigData, Computational and Management Studies*, *4*(4), 180-189. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V4I4P118

[27] Lin, Dee, and Kurt Shuler. *Optimizing Enterprise-Class SSD Host Controller Design with Arteris FlexNoC Network-On-Chip Interconnect IP*. Technical report, Arteris, Inc, 2016.

[28] Suryadevara, S. S. K., & Nakirikanti, S. (2023). Privacy-Preserving Personalization Using Federated Learning in AEM . *International Journal of AI, BigData, Computational and Management Studies*, *4*(4), 190-199. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V4I4P119

[29] Takkalapally, D., & Takkellapally, M. R. (2023). GC-TuneHFT: AI-Based Garbage Collection Optimization in High-Frequency Trading Environments. *American International Journal of Computer Science and Technology*, *5*(6), 25-37. https://doi.org/10.63282/3117-5481/AIJCST-V5I6P103

[30] Kommuru, Madhurima, and Appala Nooka Kumar Doodala. "Performance Bottlenecks Necks in Data Heavy Python Applications." *International Journal of Applied Data Science & Modern Computing* 6.2 (2023): 01-19.

[31] Kang, Jintaek, et al. "Fast simulation of a many-NPU network-on-chip for microarchitectural design space exploration." *2021 24th Euromicro Conference on Digital System Design (DSD)*. IEEE, 2021.

[32] Gaddam, R. R. (2021). Vertex AI as a Unified Control Plane for MLOps. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *2*(2), 92-102. https://doi.org/10.63282/3050-9262.IJAIDSML-V2I2P110

[33] Kumar Doodala, A. N., Thatraju, S., & Kankanala, V. (2023). Post- Pandemic QA evolution in Healthcare IT. *International Journal of Emerging Trends in Computer Science and Information Technology*, *4*(2), 223-232. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I2P122

[34] Sarihi, Amin, et al. "A survey on the security of wired, wireless, and 3D network-on-chips." *IEEE Access* 9 (2021): 107625-107656.

[35] Vppalapati, M., & Talasila, P. K. . (2023). Unobservable Performance: Storage Failures That Leave No Metrics Behind. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *4*(4), 177-188. https://doi.org/10.63282/3050-9262.IJAIDSML-V4I4P120

[36] Allenki, S. S. (2023). Reducing Security Vulnerabilities with Encryption, IAM, and Regular Audits. *International Journal of Emerging Trends in Computer Science and Information Technology*, *4*(1), 265-275. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I1P127

[37] Catania, Vincenzo, et al. "Cycle-accurate network on chip simulation with noxim." *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 27.1 (2016): 1-25.

[38] Parakala, A. (2023). Citizen-Facing Automation: Chatbots and Self-Service in Public Services. *International Journal of AI, BigData, Computational and Management Studies*, *4*(4), 108-118. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V4I4P112

[39] Muppaneni, K., & Vejella, M. (2023). Security and Data Privacy in Redux Stores. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *4*(4), 153-162. https://doi.org/10.63282/3050-9262.IJAIDSML-V4I4P117

[40] Vanapalli, Satyendra Kumar. "Reducing Financial Fraud Using Machine Learning and CRM Data Models." *International Journal of Machine Learning and Predictive Analytics* 6.2 (2023): 01-20.

[41] Srigadde, B. R. (2023). The Hidden Gem: Lightning Headless Component. *International Journal of Emerging Trends in Computer Science and Information Technology*, *4*(1), 244-254. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I1P125

[42] Kumar, Madhur, et al. "NoxyGen: A Network-On-Chip RTL Generator and Validation Tool." *Proceedings of the 16th International Workshop on Network on Chip Architectures*. 2023.