Design and Development of Novel AXI Interconnect based NoC Architecture for SoC with Reduced Latency and Improved Throughput

[a]Nagarjuna Malladhi, [b]Girish. V. Attimarad
[a]Research Scholor, K S School of Engineering and Management, Bangalore, India
[a]Vardhaman College of Engineering, Hyderabad, India
[b]K S School of Engineering and Management, Bangalore, India
[a]malldhinagarjuna@gmail.com

Abstract

A novel AXI interconnect-based Network-on-Chip (NoC) architecture is presented in this research. The purpose of the architecture is to make System-on-Chip (SoC) designs more efficient by reducing latency and improving throughput. Because of its high performance and bandwidth capabilities, the Advanced eXtensible Interface (AXI), which is a component of the Advanced Microcontroller Bus Architecture (AMBA) of the ARM architecture, is used. This configuration makes it possible to communicate effectively inside the chip. The proposed architecture overcomes the scalability limits that are inherent in conventional bus systems. This is accomplished by integrating AXI with NoC principles, which enables more efficient data transmission over a greater number of linked modules. By introducing an effective routing system and a network interface that has been improved, This research work enables packet transfer to occur without interruption. A 2x2 mesh topology is used to simulate the proposed architecture, and an XY routing algorithm is included into the simulation in order to guarantee that deadlock and livelock-free operations are carried out. This highlights the potential of the proposed architecture in high-performance computing applications that require rapid data exchange and minimal response times. The simulation results demonstrate significant improvements over traditional interconnect approaches, yielding a lower latency of 0.99 microseconds and a higher throughput of 4.363 flits per cycle which demonstrates the potential of the proposed architecture.

Keywords: AXI Interconnect, NoC, SoC, Router, Mesh Topology.

1. Introduction

The AXI interface is essential in SoC architectures for enabling communication among different system components, like processors, memory modules, peripherals, and other IP blocks [1]. It is based on the ARM Advanced Microcontroller Bus Architecture (AMBA) protocol, ensuring compatibility and interoperability across various System on Chip (SoC) setups [2]. The AXI connection is known for its scalability, being able to adapt to a variety of system configurations to fulfill the specific requirements of each SoC design [3]. It improves resource consumption and promotes system efficiency, whether handling simple single-master setups or overseeing intricate multi-layered systems. This versatility also applies to its capacity to handle many sorts of transactions, including read, write, and atomic operations, to facilitate the smooth integration of IP blocks with different data processing needs. The AXI connection has strong arbitration features to efficiently handle simultaneous access requests from several masters in the SoC context. The connection guarantees fair resource allocation by using round-robin scheduling and priority-based arbitration procedures, which help prevent bottlenecks and contention concerns. This sturdy

construction improves system performance and creates a favorable environment for developing intricate embedded systems for many applications.

Interconnects play a crucial function as the foundation of effective communication among nodes in distributed systems inside NoC designs [4]. They easily connect processor parts, memory units, and other IP blocks spread across the network, serving as the basis for integrated NoC systems. Also, follow defined protocols and topologies based on network-on-chip design concepts, promoting interoperability and scalability in various NoC setups. In NoC designs, interconnects are notable for their capacity to adapt to various system needs, supporting both homogeneous and heterogeneous node topologies. NoC interconnects boost routing efficiency and enhance overall network performance by coordinating processing units or IP blocks with various capabilities [5]. They are versatile in supporting many communication paradigms, such as packet-switched and circuit-switched architectures, to seamlessly integrate multiple processing units in the NoC ecosystem.

NoC systems include sophisticated routing algorithms and flow management technologies to efficiently handle data traffic and reduce congestion. These interconnects use virtual channel allocation and quality-of-service (QoS) priority mechanisms to provide fair resource distribution and reduce performance issues. This sturdy construction improves network speed and creates a favorable environment for developing advanced, high-performance computing systems for many applications. Numerous obstacles have surfaced in the current interconnects of NoC designs, limiting the best possible system performance and scalability. An emerging problem is the growing intricacy and diversity of system designs, which put pressure on traditional connection methods. As Network-on-Chip (NoC) designs advance to include various processor elements, memory units, and specialized IP blocks, current interconnects often face challenges in establishing effective communication channels among these different components [6]. This complexity worsens routing inefficiencies, latency bottlenecks, and contention difficulties, which impede overall network speed and responsiveness.

Another obstacle is the increasing need for bandwidth and energy efficiency in Network-on-Chip (NoC) systems. System-on-chip designs include an increasing number of cores and functional units, which require interconnects to handle expanding data traffic volumes while adhering to strict power limits. Current interconnections may have challenges in meeting these requirements, resulting in higher power use, thermal concerns, and reduced performance. Furthermore, the changing workload allocation and communication patterns in NoC designs make it more difficult, requiring flexible and scalable interconnect solutions that can allocate resources dynamically and enhance energy efficiency. Additionally, the absence of defined interfaces and protocols presents a major challenge to the interoperability and scalability of Network-on-Chip (NoC) designs. Integrating various IP blocks and system components may be challenging without consistent communication protocols, leading to compatibility problems and design inefficiencies. The lack of standardized interconnect topologies hinders the ability to easily transfer and reuse across various NoC designs, resulting in vendor lock-in and impeding innovation. To tackle these problems, industry collaboration is needed to create shared interconnect standards and frameworks, promoting a more open and compatible environment for NoC development and implementation.

Creating an AXI Interconnect-based NoC architecture for SoCs to decrease latency and improve throughput requires careful strategic planning. The design should prioritize using the scalability

and flexibility of the AXI interface to provide effective communication channels among the system components [7]. The design may efficiently distribute traffic by using several AXI interconnects arranged in a hierarchical or mesh topology, reducing contention and latency bottlenecks and maximizing resource efficiency. The design should focus on implementing advanced features including out-of-order transaction scheduling, Quality of Service(QoS) methods, and adaptive routing algorithms. The characteristics allow for dynamic resource allocation and prioritizing depending on the criticality and urgency of data transfers, improving system responsiveness and throughput. Furthermore, including specialized buffers and flow control methods into the AXI interconnects helps reduce congestion and minimize latency spikes when there is a significant volume of data traffic [8].

Optimizing the AXI Interconnect-based NoC architecture for power efficiency is crucial for maintaining sustained performance improvements. The design may reduce energy consumption without sacrificing performance by using clock gating, voltage scaling, and dynamic reconfiguration of interconnect resources. Furthermore, integrating low-power modes and intelligent power management techniques allows for precise control over power consumption, hence improving the energy efficiency of the SoC architecture. The AXI Interconnect-based NoC design may efficiently fulfill current SoC applications' strict performance requirements by balancing latency reduction, throughput enhancement, and power optimization. The organization of the paper as follows: Section-II describes the Literature survey and proposed AXI interconnect based NoC is explained in Section-III. The simulation results are discussed in Section-IV.

2. Literature

Kun-Chih (Jimmy) Chen et al [9] introduced an approach for flattening Deep Neural Networks (DNN) is to transform different DNN operations into Multiply-Accumulate (MAC)-like operations, enabling the implementation of operations like convolution and pooling in contemporary DNN architectures. A DNN slicing approach is proposed to evaluate large-scale DNN models in a resource-limited NoC environment. The assessment results show a significant reduction in off-chip memory accesses when compared to current DNN models. Performance analysis is carried out when discussing the trade-offs between various design factors.

Phan-Duy Bui et al [10] proposed the Unified System Network Architecture (USNA), a highly flexible and space-efficient NoC design that can be tailored to various topologies. The USNA provides a high degree of flexibility in port configurations to accommodate different quantities of local cores and router linkers. It also facilitates quality of service operations for routers and linkers. This research examines the network performance of the USNA, focusing on measures like average latency and saturated throughput, along with the installation cost. Multiple network architectures are studied with an equal number of local cores under uniform random traffic circumstances. The simulation findings show that the USNA performs better or equals other NoCs in terms of performance, with a smaller footprint and lower power consumption.

Gokul Krishnan et al [11] performed an experimental evaluation on several Deep Neural Networks to quantitatively analyze the performance of the IMC architecture using both NoC-tree and NoC-mesh configurations. NoC-tree is recommended for compact DNNs at the edge, whereas NoC-

mesh is necessary for accelerating DNNs with high connection density. A method is suggested to identify the best connection option for a certain DNN. This method uses analytical models of Network-on-Chip (NoC) to assess the total communication delay of the Deep Neural Network (DNN) being studied. Optimizing the connectivity in the IMC design might potentially increase the energy-delay-area product for VGG-19 inference by up to 6 times compared to current ReRAM-based IMC systems.

Aravindhan_Alagarsamy et al [12] An innovative hybrid topology called Four Regular Dense Spidergon (FRDS) is presented as a deterministic Network-on-Chip (NoC) design with 64 cores. A mix of cluster and general heuristic-based techniques is suggested for mapping applications into the FRDS topology. The cores are mapped into the topology using a genetic algorithm (GA) and simulated annealing (SA) inside a general heuristic method to guarantee effective mapping and fair performance comparison of the proposed FRDS. Experimental findings show that the suggested FRDS achieves a faster execution time compared to Mesh design when utilizing genuine benchmark traces under normalized settings.

C. De Sio et al [13] evaluated the dependability of the connecting module in programmable hardware in relation to radiation-induced failures in the configuration layer. An intentional fault injection effort is under underway to simulate the effects of radiation on the configuration memory of the AP-SoC Zynq 7000. The main focus is on the specific portion of the configuration memory responsible for programming the connection module in the programmable logic. The connection module plays a critical role in several applications and mitigation measures, such as hardware-accelerated concepts, Dynamic Partial Reconfiguration, or Triple Modular Redundancy. It is particularly important when aiming for high performance, bandwidth, and reliability. The fault injection results are analyzed and categorized based on their impact on the availability of the processor-system side along with the effect of the fault model on data computed by cores on the programmable logic side.

Demyana Emil et al [14] analyzed a low-power, straightforward design multi-core RISC-V processor that is derived from the open-source single RISC-V core processor, Taiga. The device combines two Taiga cores, solving issues related to cache coherence, connectivity, and memory architecture. Data consistency between caches and main memory is ensured by the use of the snoopy protocol. A specialized peripheral unit tailored for hardware coordinates duties across the operating cores. The primary memory unit is structured for consistency and control, using a dual-port arrangement following a specified protocol in the interface, consisting of 8192 lines and word addressable units. A UART peripheral device has been included for communication with other devices due to its widespread use in many devices and CPUs. The processor is developed using System Verilog HDL and thoroughly tested on several testbenches to guarantee correct operation.

Santhi Chebiyyam et al [15] suggested incorporating a memory controller into a multi-core System-on-Chip (SoC) using the Advanced Extensible Interface (AXI4Lite) protocol. This approach improves the performance of the multi-core SoC by using the burst mode capabilities of the AXI protocol. The architecture suggested is implemented in System Verilog HDL using the Vivado tool. The experimental results show that the suggested strategy surpasses traditional approaches documented in current literature in terms of power consumption and space utilization.

The suggested methodology results in a significant 90% decrease in power usage, as shown by numerical measurements.

Biruk Seyoum et al [16] presented DART, a tool created to automate the whole design process in a real-time Dynamic Partial Reconfiguration (DPR)-based system, including software and hardware components. DART is designed to minimize the manual work usually needed by existing tools for Xilinx Zynq 7-series and Ultrascale+ FPGA-based SoCs, without assuming extensive knowledge in programmable logic design under DPR. The tool automatically manages partitioning, floorplanning, and implementation stages, which include routing and bitstream creation. It produces a sequence of bitstreams according to tasks marked with high-level timing specifications. Mathematical optimization methods are used to solve partitioning and floorplanning issues, and a series of automatically created scripts connect with vendor products to aid in synthesis and implementation processes. DART's performance is evaluated via experimental assessment utilizing a case study application from an accelerated image processing system.

Jayshree et al [17] presented three on-chip connecting strategies designed to reduce performance decline caused by the ongoing reduction in global interconnect parameters. The suggested approaches include 2-D network-on-chip based interconnection (NoC-BI), point-to-point based interconnection (PTP-BI), and AXI4 streaming and AXI4 light bus based interconnection (AXI4-BBI). Analyzing various connections strategies via experiments to compare resource use, latency, throughput, and energy consumption. The NoC-BI approach aims to mitigate denial of service (DoS) threats, including deadlock and livelock concerns, in order to improve security in multimedia systems-on-chips. A dynamic adaptive (DyAD) routing method is suggested to alter routing according to congestion data on the route. The results show that NoC-BI has great scalability and performs better across several measures compared to PTP-BI and AXI4-BBI.

Ian Swarbrick et al [18] described the Network-on-Chip (NoC) in Xilinx's planned Versal architecture, which includes a robust NoC incorporated into Xilinx's future 7nm design devices. The devices have a range of new robust features that make up the Adaptable Computing Acceleration Platform (ACAP) devices. There is an increasing tendency in FPGA devices to strengthen frequently used components such as processors, memory controllers, and IO controllers. The next Xilinx devices have a device-wide memory-mapped NoC that connects components and the fabric in a unified way. This Network-on-Chip (NoC) enables smooth communication across the CPU system, FPGA fabric, memory subsystem, and other specialized accelerator functions. The article outlines the Versal architecture NoC and explains several unique features of the design. It shows that strengthening the Network on Chip (NoC) allows users to quickly create high-performance system-level connections.

Joshua Lant et al [19] put forward a network interface architecture and networking infrastructure designed to be included into the FPGA fabric of a sophisticated MPSoC device. This configuration enables communication across networks of devices in distributed and shared memory environments, with the goal of reducing the need for expensive software networking system calls. They discuss their implementation and prototype system, focusing on important design choices for using the Xilinx Zynq Ultrascale+, an advanced MPSoC, and overcoming problems presented by the device's restrictions and limits. The authors demonstrate a working prototype system that links

two MPSoCs, facilitating communication between the processor and a distant memory location, as well as an accelerator. They then assess the present implementation's constraints and pinpoint opportunities for enhancement to increase its preparedness for production deployment.

Debasis Behera et al [20] proposed to improve the efficiency of Embedding-Memory-Management-Units in a Network-on-Chip (NoC) system. The study investigates the use of a 3D Network-on-Chip (NoC) to enhance NoC performance, resulting in significant progress. The research also uses first-in-first-out (FIFO) buffers in NoC routers to temporarily hold data packets. A suggestion suggests using RAM as an intermediary between the crossbar switch and input ports. The simulation findings show that the study achieves memory usage levels between 0 and 16 out of 64 in a data storage stack, with a constantly strong "almost empty" signal.

3. Proposed AXI interconnect based NoC Architecture

An AXI interconnect-based NoC architecture is a highly advanced and innovative approach to designing and implementing communication subsystems in complex integrated circuits, particularly in System-on-Chip (SoC) topologies. The Advanced eXtensible Interface (AXI) is a component of the ARM Advanced Microcontroller Bus Architecture (AMBA) standard. It functions as a high-performance and high-bandwidth bus interface that facilitates the connection of components inside a microcontroller system. The integration of AXI with NoC principles enables the development of a communication infrastructure that is scalable, efficient, and adaptable. This infrastructure is capable of meeting the high data transmission demands of current computer applications.

The proposed architecture utilizes the AXI protocol's capabilities in facilitating rapid data transmission, handling many data streams, and effectively managing concurrency. By using a Network-on-Chip (NoC) approach, the design may overcome the constraints in scalability that are present in conventional bus systems. This enables more efficient transfer of data across a greater number of linked modules or processing units. The NoC design functions as a network fabric that links different components, including processors, memory blocks, and I/O devices. This allows them to interact with each other via a common network infrastructure.

Within this architectural framework, the nodes of the NoC are interconnected by routers, which control the routing of data packets depending on network circumstances and destination addresses. These routers are specifically intended to accommodate the characteristics of the AXI protocol, including out-of-order transaction completion, burst transfers, and split transactions. This ensures that they are compatible with AXI-compliant modules and maximizes the efficiency of data transmission and overall system performance. The block diagram of the proposed AXI interconnect based NoC architecture is depicted in Figure 1.

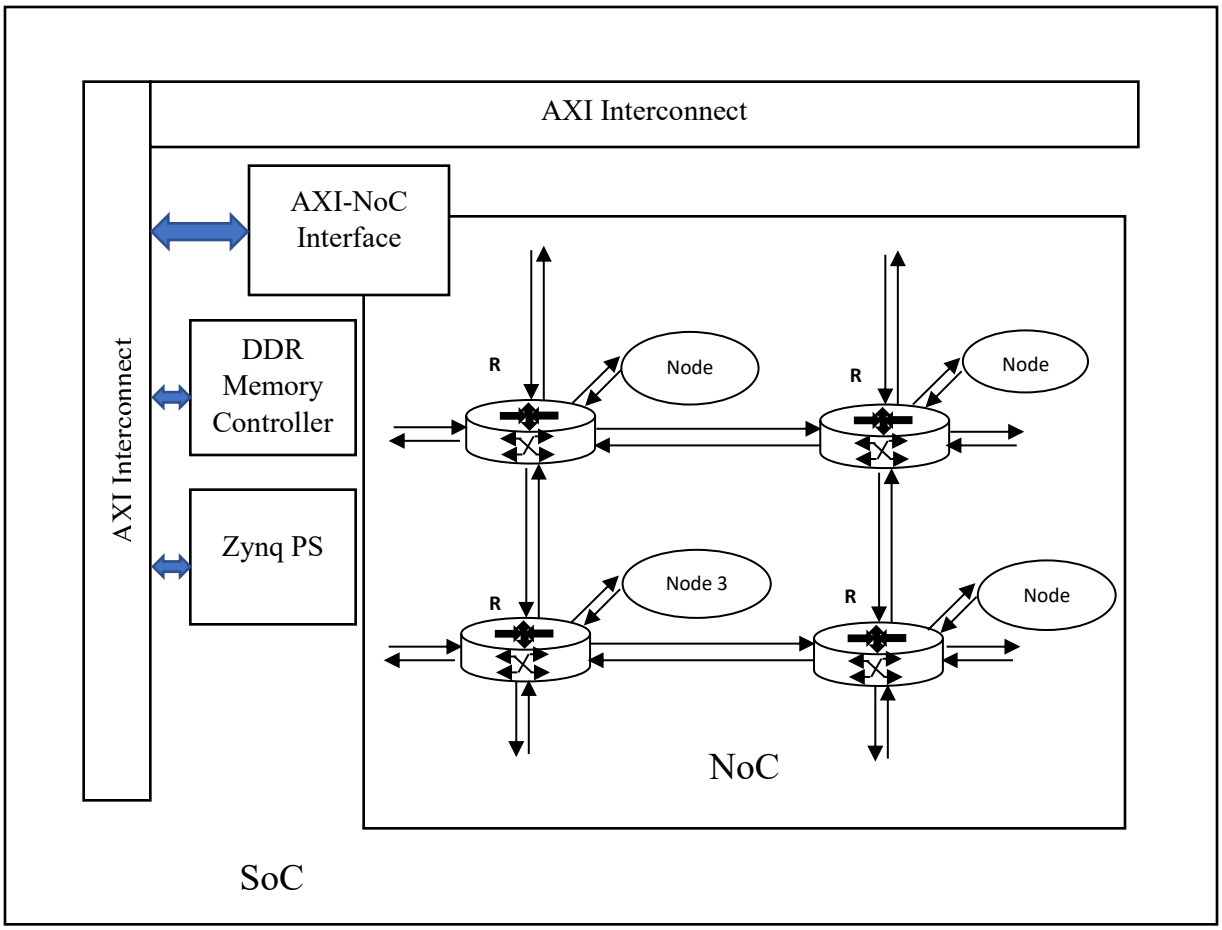


Figure 1: Proposed AXI interconnect based NoC Architecture

The block diagram of a proposed NoC architecture based on AXI interconnect would consist of many essential components:

AXI-Master and Slave modules: AXI-compliant Master and Slave Modules refer to the components such as processor units, memory controllers, and peripheral devices that establish communication using the AXI interface. Masters are responsible for initiating transactions, whilst slaves are responsible for responding to them.
Routers: Routers are essential components of the NoC, responsible for overseeing the routing of data packets between nodes. They do this by using network topology and routing algorithms. They have been designed specifically to effectively manage the needs of the AXI protocol.
Network Interfaces (NIs): NIs, positioned between the AXI modules and the routers, convert AXI transactions into NoC packets and vice versa, guaranteeing smooth integration between the AXI interface and the NoC architecture.
Interconnect links: Interconnect links refer to the physical or logical connections that exist between routers, allowing for the transport of data packets across the network. They may be engineered to accommodate different bandwidths and latency demands.

3.1 NoC Architecture

The NoC architecture comprises Routers, network interfaces, IPs, along with connections. Figure 2 displays the components of a (2×2) Mesh topology. The network topologies are determined by the connectivity of the Router, NI, IP, and link. The essential elements of NoC architecture consist of the routing algorithm, network structure, including switching mechanisms. The router is an essential element of a SoC that is constructed using NoC architecture, similar to other types of networks. There are communication lines that connect the whole chip, and the NoC Router is responsible for efficiently directing incoming packets to either the core that they are meant for or the router that comes after it along the routing path that extends from the originating point to the destination.

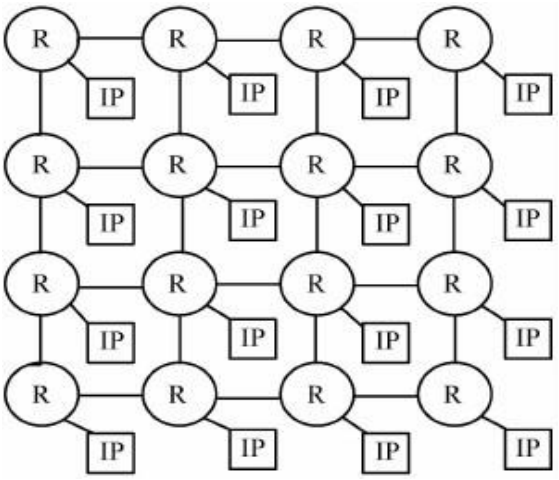


Figure 2: NoC Architecture

Network interfaces provide the link between the IP cores as well as the on-chip router network. A Network Interface in a NoC serves as an intermediary between the computing unit and the communication system. Network interfaces facilitate the transfer of data produced by IP blocks into data packets and also provide additional routing information dependent on the underlying NoC network. NoC routers serve as the primary means of steering packets in a communication network. Routers enable the transfer of packets to the chosen connection in order to reach their intended destination.

3.2 Router

A router is a crucial element in the communication infrastructure of a NOC system. The router enables the efficient transmission of network communication from its origin to its intended endpoint. It ensures the synchronization of data transmission, which is a crucial component of communication networks. The router's design has five buffers: north, south, east, west, and a local buffer, as seen in Figure 3.

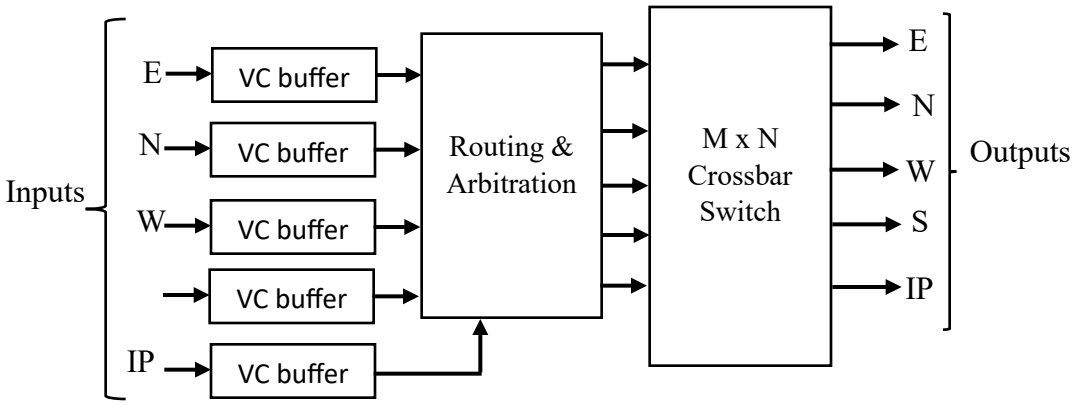


Figure 3: Router (2D mesh topology)

The local buffer serves as a means to connect the IP core, while the first four ports are used to establish connections with other routers inside the network. Routers, which are intelligent devices, accept incoming packets, analyze their destination, and determine the optimal path for transmitting packets from the source node to the destination node. Using the routing function, a router decodes the data from the incoming message and determines the packet's destination. The OSI model is followed in the construction of the NoC router. Every layer in the model created by OSI has certain tasks to complete.

3.3 AXI Network Interface

The network interface establishes the logical link between the IP core and the network. The network interface serves as an intermediary between the router and the IP core. NI keeps an eye on packets being sent and received inside the IP core. Simultaneous bidirectional communication is made possible via the network interface. It starts by gathering IP core data. After that, it divides the data into packets, gives each packet a destination address, and transmits the packets to the router. The packets are then sent to their intended destination once it has removed the packetization and received them from the associated routers.

3.3.1 AXI interconnect

AXI interconnects have been designed to meet the demands of on-chip communication that requires both high bandwidth and low latency. These interconnects are meant to be compatible and adaptable to different design requirements. The AXI connection plays a vital role in the advancement of complex digital systems that need effective data transfer across many processor units, memory blocks, and peripherals. The multi master and multi slave with AXI interconnect is depicted in Figure 4.

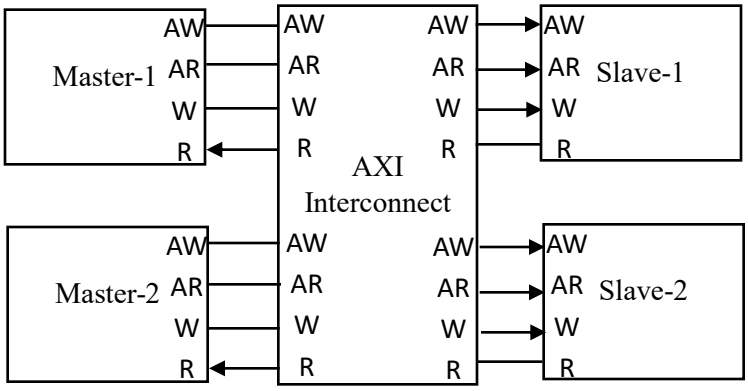


Figure 4: AXI interconnect with multi master and slave

The fundamental purpose of the AXI protocol is to enable direct communication between master and slave devices in a system. It facilitates efficient data transfers by allowing for concurrent addresses, transfers of data that are not aligned, and transactions that occur in bursts. These properties play a crucial role in achieving the efficiency necessary in contemporary high-speed computing systems. The protocol specifies many channels (read address, write address, read data, write data, and write response) that function autonomously, enabling concurrent data transactions, hence enhancing throughput and system performance to a large extent. The channels in the AXI protocol is shown in Figure 5.

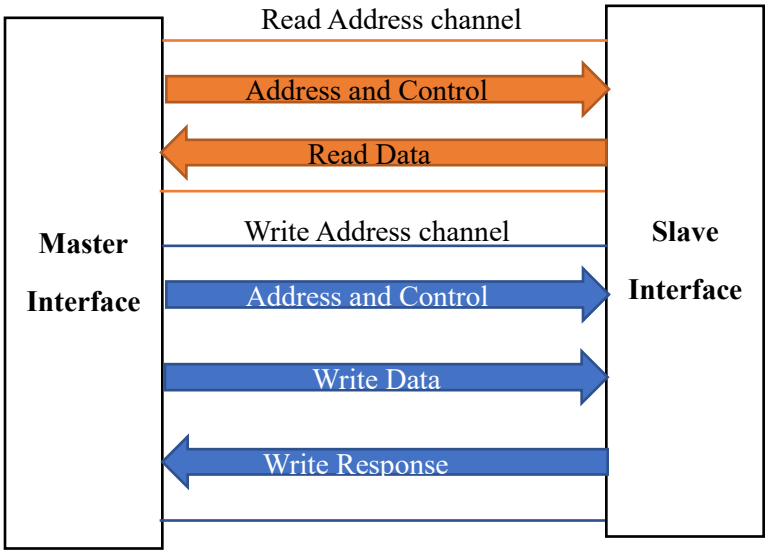


Figure 5: AXI channels

The AXI protocol defines five distinct types of channels:
Read Address Channel: The master device utilizes this channel to transmit read requests to the slave device. The read address channel conveys details on the data's source address, as well as transaction parameters such as data amount, burst type, and transaction ID. This allows the slave to comprehend the specific facts that the master is soliciting.

Read Data Channel: The slave uses the read data channel to provide the data requested by the master via the read address channel. In addition to the primary data, this channel also transmits the response status (showing whether the read operation was successful or not) and the transaction ID, which assists the master in linking the incoming data with the appropriate request.
Write Address Channel: The write address channel, like the read address channel, is used by the master to commence a write operation. The data packet contains the destination address, transaction characteristics, and a distinct transaction identifier. This channel efficiently conveys information to the subordinate on the master's desire to record data and the precise details of the activity.
Write Data Channel: The write data channel is tasked with transmitting the factual data from the master to the slave for the purpose of writing it to the designated location. This channel transmits the data along with the write strobes (which signify the valid data bits in the transfer) and the transaction ID. The write strobes are essential for performing partial writes or for writing data that is less than the bus width.
Write Response Channel: Once a write operation is started using the write address and data channels, the slave utilizes the write response channel to confirm the successful completion of the write operation. The system returns a status that indicates whether the write transaction was successful or not, together with the transaction ID. This allows the master to verify that the write operation was done accurately.
Each of these channels functions autonomously, enabling concurrent processing of numerous transactions. This greatly improves the data throughput and overall efficiency of the system. The division of channels for addressing, data transmission, and control signals in SoC designs that use the AXI protocol reduces congestion and optimizes performance. The AXI connection offers significant benefits due to its capability to accommodate various degrees of concurrency and effectively manage transactions of varying sizes. The flexibility of AXI-based systems allows them to be customized for many purposes, ranging from basic control duties to intricate data-intensive activities. The AXI connection has a split-transaction mechanism that separates the request and response portions of a transaction. This design reduces latency and enhances data throughput inside the system.

4. Simulation Results

The proposed AXI interconnect is integrated with NoC architecture, rather than the traditional bus-based NoC architecture. The 2x2 mesh topology is considered for the simulation and *xy* routing algorithm is incorporated in the router. The mesh topology is characterized by a grid-like structure with *n* rows and *m* columns. In a mesh architecture, each router is linked to the neighboring router using cables. The network's *(x, y)* coordinates are used to specify the address of the router and IP cores. In a Mesh Topology, the detection of defects and the avoidance of problematic nodes during packet routing in the network are straightforward and efficient. This topology is the most straightforward to implement compared to other topologies. In this architecture, packets traverse a dedicated connection and are delivered only to their intended destinations.

4.1 Routing Algorithm

Routing algorithms play a crucial role in optimizing communication inside a NoC. These methods ascertain the precise path that a packet should follow in order to reach its intended destination node. Several routing algorithms have been suggested for implementation in NoC systems, and

they may be categorized based on their distinct characteristics and needs. The routing algorithm may be classed as source, distributed, or centralized based on where the routing option is made. In a centralized algorithm, the route is selected by the central controller. Source routing involves the selection of a path by the source router before transmitting a packet, while distributed routing involves the selection of the routing path by intermediate routers. The *xy* routing algorithm is chosen in this work due to adoptability in nature and suitable for 2D mesh topologies.

The *xy* routing approach belongs to the category of distributed deterministic routing algorithms. *xy* routing is free from both deadlocks and livelocks. The *xy* routing algorithm typically selects the shortest and predetermined route for packet transmission. This approach is applicable to both regular and irregular network topologies. Each node in the mesh network is identified by its coordinates, represented as *(x, y),* where *x* represents its horizontal location and *y* represents its vertical position.

The path from the source node to the destination node is pre-established and stays constant regardless of the network's condition. Under conditions of non-congestion, the NoC network exhibits a significant level of reliability and encounters little latency. This strategy establishes a sequential movement of packets, first in the X-axis and then in the Y-axis. It blocks packets from using other paths to circumvent blocked pathways. The present position of the router, indicated by its *(x, y)* coordinates, is compared to the coordinates of the destination router to establish the route. The data packet is first routed down the X-axis and then along the Y-axis until it reaches its designated destination IP core. The XY routing technique in Mesh topology allows for just half of the available turns by restricting the other half of turns. XY routing involves the initial movement of a packet in the x-direction. Once the packet reaches the desired column, it is then transported in the y-direction, either upwards or downwards. The xy algorithm is reported in Table 1.

Table1: XY Routing for 2D Mesh Network-on-Chip (NoC)

Algorithm: XY Routing for 2D Mesh Network-on-Chip (NoC)

1. Inputs:
   - Source Node Coordinates: (X_source, Y_source)
   - Destination Node Coordinates: (X_dest, Y_dest)
2. Output:
   - Selected Output Channel
3. Procedure:
   - Calculate the differences between the destination and source coordinates:
     - X_offset = X_dest - X_source
     - Y_offset = Y_dest - Y_source
   - If the offsets are both zero (meaning source and destination are the same), the algorithm terminates as no routing is needed.
   - If the Y_offset is positive, the selected output channel is North (Y+), indicating movement towards a higher Y coordinate.
   - If the Y_offset is negative, the selected output channel is South (Y-), indicating movement towards a lower Y coordinate.
   - If the X_offset is positive, after any north or south movement, the selected output channel is East (X+), indicating movement towards a higher X coordinate.

- If the X_offset is negative, after any north or south movement, the selected output channel is West (X-), indicating movement towards a lower X coordinate.

If the Y_{offset} is positive, the xy routing strategy routes packets to the west buffer. When the value is negative, the packet is routed to the left, namely towards the east buffer. If the X_{offset} is not equal to zero, the packet is sent either upwards or downwards along the y-axis. If both the Y_{offset} and X_{offset} values are equal to zero, it signifies that the packet has successfully arrived at its intended destination. The route from the starting node to the target node is consistently the most direct and stays constant. This technique demonstrates reduced latency in situations of low network traffic due to its static nature. Nevertheless, its efficiency declines considerably when there is congestion and a restricted selection of alternative routes. When faced with a consistent traffic pattern, this NoC routing method outperforms other algorithms. The xy routing algorithm network experiences a much higher load in its central region compared to the average load over the whole network. This results in a concentration of traffic in the center, which is often referred to as a hotspot. If there is a faulty node along the route, the packet will get trapped in one of the switches.

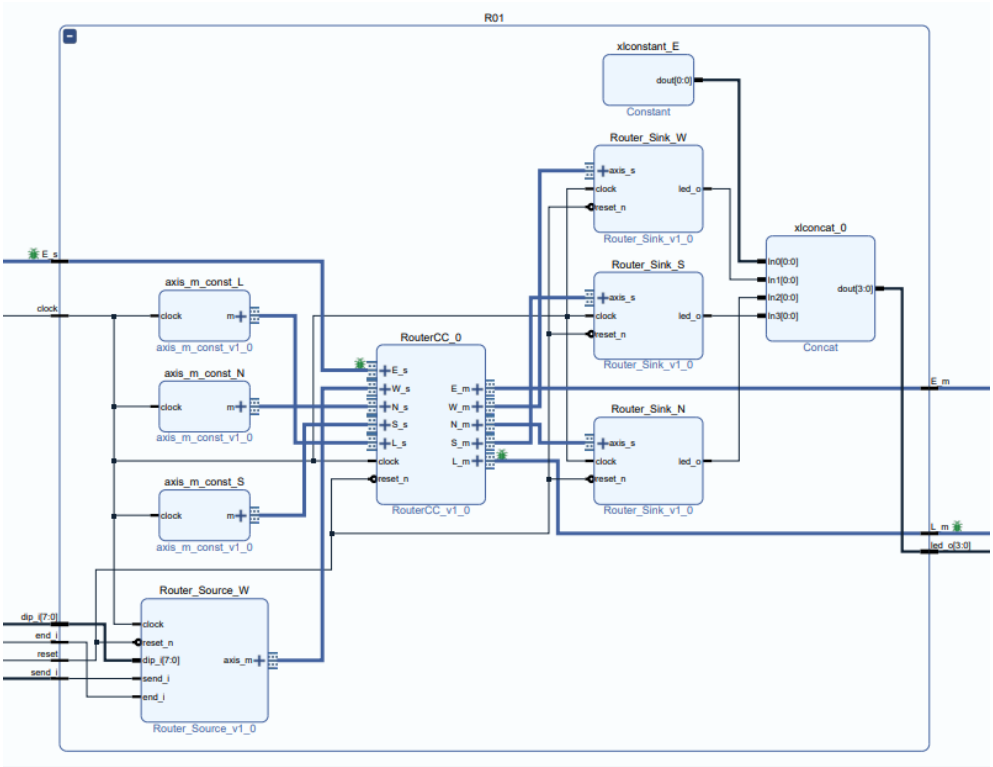


Figure 6: Schematic of NoC Architecture

The Figure 6 is a schematic design of a NoC router. Below is a concise description of the various blocks shown in the schematic:

- **RouterCC (Router Crossbar Connect):** This is the core component of the router, which is responsible for directing data packets from the source to the destination. The system may include a crossbar switch that links several input and output channels, which may originate from the East (E), West (W), North (N), South (S), and local (L) directions.

- **Router_Sink_W, Router_Sink_S, Router_Sink_N:** These blocks are sink modules, which serve as endpoints in the NoC where data packets are received from the router. The letters "*W*", "*S*", and "*N*" represent the west, south, and north directions, respectively. Buffers may be included to store incoming data.
- **Router_Source_W:** This module serves as a source, primarily responsible for transmitting data packets into the network. The connection will be oriented towards the West, in accordance with the naming tradition.
- **axis_m_const_L, axis_m_const_VI_0, axis_m_const_N, and axis_m_const_S:** These blocks might potentially function as data generators or placeholders for data streams, serving as the sources of traffic for the router. The RouterCC may receive either constant or variable data for the purpose of routing it to various sinks.
- **dip_[7:0], send_i, end_i:** These signals are inputs and controls for the *Router_Source_W* module. "*dip*" likely represents the data input, "*send_i*" might be a signal to initiate data transmission, and "*end_i*" may indicate the completion of data transmission.
- **xconstant_E:** This component serves as a constant generator that produces a consistent value. It is used for controlling or monitoring purposes inside the NoC. The constant value is oriented eastward.
- **xlconcat_0:** This block is a concatenation module that merges numerous input signals into a single broader output signal.

The signals *axis_s, clock, led_o,* and *reset_n* are often used as interface signals. "*axis_s*" refers to the AXIS (AXI Stream) interface. "*clock*" represents the system clock signal. "*led_o*" is an output signal used to drive an LED for debugging or status signaling. Lastly, "*reset_n*" is an active-low reset signal. The figure 6 also illustrates the interconnections among these blocks, which symbolize the transmission of data and control signals inside the router. The connections to the *East (E_m), West (W_m), North (N_m), South (S_m),* and *Local (L_m)* with the *RouterCC* block describe the possible routes for data transmission inside the network. The nomenclature used for these blocks and signals implies the presence of a standardized interface, most likely AXIS, which is widely used in FPGA and ASIC architecture for the purpose of streaming data.

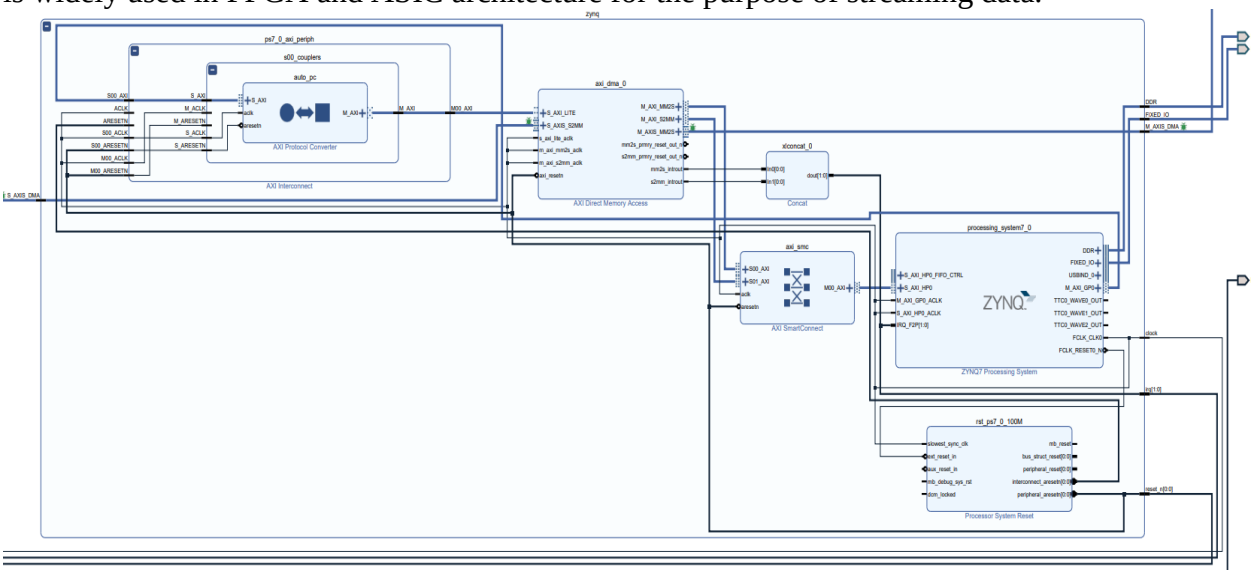


Figure 7: AXI interconnect with Zynq SoC

The Figure 7 depicts a schematic representation of an AXI interconnect topology that connects with a NoC. Below is a breakdown of the many elements that are depicted:

AXIS DMA: The AXIS DMA block is a controller that represents an AXI Stream Direct Memory Access (DMA). DMA controllers facilitate the autonomous movement of data between memory and peripherals, eliminating the need for CPU involvement.

ps7_0_axi_periph: The block labeled "*ps7_0_axi_periph*" indicates an AXI peripheral that is linked to the processing system. The term "*ps7_0*" suggests that it is a component of a Xilinx Zynq-7000 series SoC, where "*ps7*" refers to the seventh version of the Processing System.

s00_couplers: The s00_couplers module enables the linkage between the AXI peripheral and the AXI interconnect. It functions as an intermediary, ensuring proper communication between the peripheral and the AXI Interconnect.

auto_pc (AXI Protocol Converter): The *auto_pc* is a component that handles the conversion of protocols, potentially across multiple AXI interfaces (such as from AXI3 to AXI4). This guarantees interoperability across IP blocks that may be using disparate versions of the AXI protocol.

M_AXI, S_AXI: The M_AXI and S_AXI interfaces refer to the master and slave AXI interfaces, respectively. The "*M_AXI*" interface serves as the primary interface responsible for initiating read and write transactions. On the other hand, the "*S_AXI*" interface functions as the secondary interface that replies to these transactions launched by the master.

AxCxK, AxRESxT: The control signals *AxCxK* and *AxRESxT* are part of the AXI protocol.

- "*AxCxK*" is an abbreviation for AXI Clock, which refers to the clock signal used for the AXI interface.
- "*AxRESxT*" is most likely an abbreviation for AXI Reset, which serves as the reset signal for the AXI interface.

ack, aresetn: The control signals often seen in digital circuits are known as "*ack*" and "*aresetn*".

- "*ack*" is often used as an acknowledgement signal during the process of handshaking between different components.
- The "*aresetn*" signal is a kind of reset signal that is active-low and asynchronous. It is used to reset the interface or component.

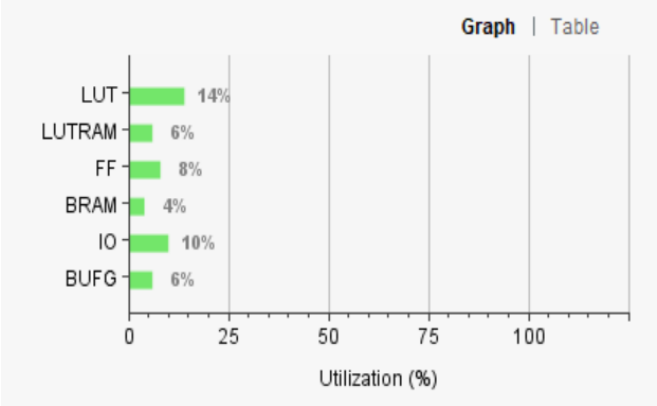
AXI Interconnect: The AXI Interconnect is a pivotal component that facilitates communication between various masters and slaves inside the system. It manages the process of directing transactions from masters to the right slaves.

Figure 8: Simulation results of Proposed AXI interconnect based NoC Architecture

The Figure 8 is a waveform derived from a simulation of a NoC architecture using AXI interconnect. An analysis of the fundamental components included in the waveform described as below:

- **dma_dout:** This indicates the digital output signal originating from the DMA block. The signal seems to be a binary signal with two bits, where each bit alternates at distinct time intervals throughout the simulation.
- **slot_0: dma_s Interface:** This interface facilitates the DMA transaction specifically for slot 0. The signal comprises:
  - **TVALID:** Indicates that the master is performing a legitimate transfer.
  - **TREADY:** Indicates the slave's readiness to accept a transfer.
  - **TLAST:** Indicates the last transfer inside a transaction.
  - **TDATA:** The data that is being sent.
- **slot_2: Conn1 Interface and slot_1: S_AXI Interface:** The interface for *slot_2* is Conn1, whereas the interface for slot_1 is S_AXI. These interfaces are supplementary to the AXI connection. The presence of the same signals (*TVALID*, *TREADY*, *TLAST*, *TDATA*) indicates the occurrence of many simultaneous contacts with the interconnect.

In Figure 8, data sent when both *TVALID* and *TREADY* signals are in a high state. *TDATA* displays the specific data that is being sent throughout these transactions. The hexadecimal value denotes the information payload. A high value of *TLAST* signals the completion of a series of data transfers. The simulation results indicate that the DMA is effectively starting transactions with the interface, and data is being transferred and received as expected without any apparent faults or conflicts seen in the waveform.



(a) Area utilization

Total On-Chip Power:	1.71 W
Junction Temperature:	44.7 °C
Thermal Margin:	40.3 °C (3.4 W)
Effective ϑJA:	11.5 °C/W

(b) Power utilization

Figure 9: Area and power utilization summary

Table 2: Resource Utilization comparison results

Method	**Resource Utilization (Slices) in %**
PTP-BI [17]	50.58
AXI4-BBI [17]	34.48
Proposed AXI interconnect	14

The Table 2 presents a summary of the resource use of various connectivity technologies, with a special emphasis on the proportion of slices used in Zynq 7000 SoC architecture. The first technique mentioned, PTP-BI [17], refers to Point-to-Point Bidirectional Interconnect, as cited in source [17]. It represents 50.58% of the slices. In this particular architecture, direct connections are made between two ends, enabling data flow in both directions. The significant portion of the slice consumption suggests a possibly extended connection configuration that necessitates a huge amount of logic resources. PTP-BI is followed by AXI4-BBI [17], which refers to an AXI version 4 Bus-Based Interconnect, also mentioned in the same source [17]. This connection utilizes a smaller percentage of slices, 34.48%. The AXI4 standard is renowned for its exceptional performance and is often used in system-on-chip architectures that operate at high frequencies. Based on the statistics, it can be inferred that a bus-based connection is more efficient in terms of resource use when compared to the point-to-point bidirectional architecture. The Proposed AXI interconnect based design demonstrates resource efficiency by employing 14% of the available slices. This signifies a substantial decrease in the use of resources as compared to the PTP-BI and AXI4-BBI approaches. Latency and throughput are crucial performance measures for assessing NoC designs. Comprehending and computing these metrics is essential for designers to guarantee that the NoC fulfills the required criteria for effective data transmission inside a SoC. Given that the simulation ended after 3497 cycles and 3000 cycles measured, the approximate latency would be 497 cycles. Each clock cycle required 2ns time period, so latency would be 994ns or 0.99 µs.

Latency in a Network-on-Chip (NoC) refers to the time it takes for a data packet to get from the starting node to the ending node. It encompasses the duration required for routing, the processing that occurs at intermediate switches, and any potential delays in queuing. The latency may be determined by measuring the number of cycles (clock cycles), which is dependent on the clock frequency of the system. Throughput in a NoC refers to the rate at which data may be sent between nodes within a certain time frame. The typical unit of measurement is bits per second (bps) or transactions per cycle. In this simulation, the simulation ended after 3497 cycles. 15257 flits sent, and also 15257 flits received by the end of the simulation. The formula for throughput (*Th*) would then be:

$$Th = \frac{number\ of\ flits\ sent\ or\ received}{Total\ Cycles} \quad (2)$$

The proposed NoC has a throughput of roughly 4.363 flits per cycle, as shown by the given statistics. On average, about 4.363 flits are successfully conveyed via the network each cycle.

Table 3: Latency and Throughput comparison results

Method	Latency (µs)	Throughput (flits per cycle)
PTP-BI [17]	33.9	-
AXI4-BBI [17]	56.29	-
NoC-BI [17]	5.87	0.38
Proposed AXI interconnect	0.99	4.363

The Table 3 provides a comparative study of several connectivity techniques in network architectures, evaluated based on their latency and throughput measurements. Latency is defined in microseconds (µs), whereas throughput is evaluated by the number of flits. The first approach is the PTP-BI [17], with a recorded latency of 33.9 µs. The AXI4-BBI [17] approach has a much greater delay of 56.29 µs. The NoC-BI (Network-on-Chip Bidirectional Interconnect) developed in [17] demonstrates a significant increase in performance, achieving a latency of just 5.87 µs and a throughput of 0.38 flits per cycle. The Proposed AXI connection, which demonstrates outstanding performance metrics: a minimal latency of just 0.99 µs, coupled with a high throughput of 4.363 flits per cycle. These numerical metrics demonstrate a connection that has been specifically designed to maximize both speed and efficiency. This architecture facilitates efficient data transfer with low latency and allows a huge volume of data to be processed, which is very beneficial in high-performance computing applications that need fast data interchange and decreased response times.

Conclusions

The findings of the research that was carried out in this article provide evidence that the AXI interconnect-based NoC architecture that was proposed is beneficial in improving the performance of SoCs. The findings that were acquired from the simulation tests show proof of the architecture's improved performance. In comparison to typical NoC designs, the architecture achieved much reduced latency and greater throughput. The NoC architecture that has been proposed provides a

solution that is scalable, efficient, and adaptive for the increasingly data-intensive applications that are being used today. It stands out as a viable approach for the development of future SoC implementations. An infrastructure for communication that is both resilient and high-speed is produced as a result of the combination of AXI protocols and NoC principles. This infrastructure is suited for complicated integrated circuits and the demands of current computing workloads. A second factor that contributes to the attractiveness of the proposed design is the significant decrease in resource use, which is down to 14% of slices. This provides an alternative to current interconnect techniques that is more efficient with resources. The findings of this study provide a significant contribution to the field of on-chip network design and establish a standard for future research in the creation of SoC architectures that are high-performance, low-power, and efficient in terms of area use.

References

1. Chakravarthi, Veena S., and Shivananda R. Koteshwar. "System on Chips (SOC)." In *System on Chip (SOC) Architecture: A Practical Approach*, pp. 17-35. Cham: Springer Nature Switzerland, 2023.
2. Harish, T. L., and M. C. Chandrashekhar. "Review on Design and Verification of an Advanced Extensible Interface-4 Slave Devices." *ACS Journal for Science and Engineering* 3, no. 2 (2023): 15-20.
3. Schmid, Robert, Max Plauth, Lukas Wenzel, Felix Eberhardt, and Andreas Polze. "Accessible near-storage computing with fpgas." In *Proceedings of the Fifteenth European Conference on Computer Systems*, pp. 1-12. 2020.
4. Manzoor, Misbah, and Roohie Naaz Mir. "A Review of Design Approaches for Enhancing the Performance of NoCs at Communication Centric Level." *Scalable Computing: Practice and Experience* 22, no. 3 (2021): 347-364.
5. Poovendran, R., and S. Sumathi. "An area-efficient low-power SCM topology for high performance network-on Chip (NoC) architecture using an optimized routing design." *Concurrency and computation: practice and experience* 31, no. 14 (2019): e4760.
6. Venkataraman, N. L., and Rajagopal Kumar. "An efficient NoC router design by using an enhanced AES with retiming and clock gating techniques." *Transactions on Emerging Telecommunications Technologies* 31, no. 12 (2020): e3839.
7. Uma, V., and Ramalatha Marimuthu. "D-wash–A dynamic workload aware adaptive cache coherance protocol for multi-core processor system." *Microelectronics Journal* 132 (2023): 105675.
8. Javed, Aqib, Jim Harkin, Liam McDaid, and Junxiu Liu. "Predicting Networks-on-Chip traffic congestion with Spiking Neural Networks." *Journal of Parallel and Distributed Computing* 154 (2021): 82-93.
9. Chen, Kun-Chih Jimmy, Masoumeh Ebrahimi, Ting-Yi Wang, Yuch-Chi Yang, and Yuan-Hao Liao. "A NoC-based simulator for design and evaluation of deep neural networks." *Microprocessors and Microsystems* 77 (2020): 103145.
10. Bui, Phan-Duy, and Chanho Lee. "Unified system network architecture: flexible and area-efficient NoC architecture with multiple ports and cores." *Electronics* 9, no. 8 (2020): 1316.
11. Krishnan, Gokul, Sumit K. Mandal, Chaitali Chakrabarti, Jae-Sun Seo, Umit Y. Ogras, and Yu Cao. "Impact of on-chip interconnect on in-memory acceleration of deep neural networks." *ACM Journal on Emerging Technologies in Computing Systems (JETC)* 18, no. 2 (2021): 1-22.

12. Alagarsamy, Aravindhan, Sundarakannan Mahilmaran, Lakshminarayanan Gopalakrishnan, and Seok-Bum Ko. "FRDS: An efficient unique on-Chip interconnection network architecture." *Integration* 87 (2022): 90-103.
13. De Sio, C., S. Azimi, and L. Sterpone. "On the analysis of radiation-induced failures in the AXI interconnect module." *Microelectronics Reliability* 114 (2020): 113733.
14. Emil, Demyana, Mohammed Hamdy, and Gihan Nagib. "Development an efficient AXI-interconnect unit between set of customized peripheral devices and an implemented dual-core RISC-V processor." *The Journal of Supercomputing* (2023): 1-20.
15. Chebiyyam, Santhi, M. Gurunadha Babu, M. Ajay Kumar, and L. Radhika Rani. "Development of Low Power and Area Efficient Multi-core Memory Controller Using AXI4-Lite Interface Protocol." In *International Conference on Data Science and Communication*, pp. 691-703. Singapore: Springer Nature Singapore, 2023.
16. Seyoum, Biruk, Marco Pagani, Alessandro Biondi, and Giorgio Buttazzo. "Automating the design flow under dynamic partial reconfiguration for hardware-software co-design in FPGA SoC." In *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, pp. 481-490. 2021.
17. Jayshree, Gopalakrishnan Seetharaman, and Debadatta Pati. "Design and Area Performance Energy Consumption Comparison of Secured Network-on-Chip with PTP and Bus Interconnections." *Journal of The Institution of Engineers (India): Series B* 103, no. 5 (2022): 1479-1491.
18. Swarbrick, Ian, Dinesh Gaitonde, Sagheer Ahmad, Brian Gaide, and Ygal Arbel. "Network-on-chip programmable platform in VersalTM ACAP architecture." In *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 212-221. 2019.
19. Lant, Joshua, Caroline Concatto, Andrew Attwood, Jose A. Pascual, Mike Ashworth, Javier Navaridas, Mikel Lujan, and John Goodacre. "Enabling shared memory communication in networks of mpsocs." *Concurrency and Computation: Practice and Experience* 31, no. 21 (2019): e4774.
20. Behera, Debasis, Suvendu Narayan Mishra, Prabodh Kumar Sahoo, and Heli Amit Shah. "An enhanced approach towards improving the performance of embedding memory management units into Network-on-Chip." *e-Prime-Advances in Electrical Engineering, Electronics and Energy* 6 (2023): 100332.