PATRONoC: Parallel AXI Transport Reducing Overhead for Networks-on-Chip targeting Multi-Accelerator DNN Platforms at the Edge

Author(s):
Jain, Vikram; Cavalcante, Matheus; Bruschi, Nazareno; Rogenmoser, Michael; Benz, Thomas; Kurth, Andreas; Rossi, Davide; Benini, Luca; Verhelst, Marian

Publication date:
2023

Permanent link:
https://doi.org/10.3929/ethz-b-000639837

Originally published in:
https://doi.org/10.1109/DAC56929.2023.10247800

PATRONoC: Parallel AXI Transport Reducing Overhead for Networks-on-Chip targeting Multi-Accelerator DNN Platforms at the Edge

Vikram Jain[†], Matheus Cavalcante[⋆], Nazareno Bruschi[⊕], Michael Rogenmoser[⋆], Thomas Benz[⋆], Andreas Kurth[⋆], Davide Rossi[⊕], Luca Benini[⋆⊕] and Marian Verhelst[†]
[†]ESAT-MICAS, KU Leuven, Belgium, [⋆]IIS, ETH Zurich, Switzerland, [⊕]University of Bologna, Italy
Email: vikram.jain@kuleuven.be

***Abstract*—Emerging deep neural network (DNN) applications require high-performance multi-core hardware acceleration with large data bursts. Classical network-on-chips (NoCs) use serial packet-based protocols suffering from significant protocol translation overheads towards the endpoints. This paper proposes PATRONoC, an open-source fully AXI-compliant NoC fabric to better address the specific needs of multi-core DNN computing platforms. Evaluation of PATRONoC in a 2D-mesh topology shows 34 % higher area efficiency compared to a state-of-the-art classical NoC at 1 GHz. PATRONoC's throughput outperforms a baseline NoC by 2-8× on uniform random traffic and provides a high aggregated throughput of up to 350 GiB/s on synthetic and DNN workload traffic.**

***Index Terms*—Networks-on-chip, multi-core DNN platforms, AXI, high-performance systems**

I. Introduction

Deep neural networks (DNNs) have become one of the primary workloads in computing platforms of data centers and edge devices in the internet of things (IoT). Given the high proliferation of DNN workloads, research into designing and developing high-performance specialized hardware accelerators for DNN has gained much interest, as evidenced by the several DNN accelerators presented in the past decade [1]. In the quest to support the ever-growing requirements of DNN workloads, hardware architectures have evolved from small single-core implementations to homogeneous [2] and heterogeneous [3]–[5] multi-core hardware implementations[1]. The trend of going multi-core can bring performance gains. However, it also brings new challenges, such as resource partitioning, workload mapping, complex hardware implementations, memory hierarchy design, and data communication bottlenecks between cores.

Multi-CPU-based general-purpose computing traditionally uses networks-on-chip (NoCs) and their various optimizations for inter-CPU data communication. Many topologies exist to balance the scalability of CPU cores, throughput, latency, and area impact of the NoC. Moreover, NoC protocols are designed for packetization and serialization over fairly narrow channels between cores (e.g., 32 bits), which reduces the number of routing resources needed. However, this implies additional hardware at the network's edges for protocol translation and serialization/deserialization (SERDES) from standard channel-oriented protocols at the endpoints (e.g., AXI4 or AXI5) to the NoC protocol. Moreover, due to their serialized nature, these NoCs need a high clock frequency to meet the bandwidth requirements, thus needing clock domain crossing hardware.

Such traditional narrow-channel NoCs work well for inter-CPU cache traffic. However, the traffic of DNN workloads is mostly deterministic, with large bursts of non-coherent data movements requiring high bandwidth interconnection to achieve high performance and low latency. To achieve high bandwidth, typical solutions either 1) use a narrow NoC and operate it at 2-8× the core frequency [6] or 2) build a wide NoC with multiple channels [7]. The latter solution gains traction as advanced technology scaling enables the area-efficient integration of more and more on-chip interconnect resources [7], [8]. However, modern NoCs need more than just wide links to answer the needs of DNN workloads, as packet-based serial NoC protocols are inadequate for workloads that rely on burst-based traffic.

This paper proposes a template for burst-based homogeneous AXI-compliant NoCs to address the requirements of emerging multi-core DNN platforms and to tackle the challenges of packet-based serial NoCs. This work builds upon the open-source elementary AXI building blocks of [9], which focuses on crossbar-based topologies, towards a fully-configurable open source AXI-based NoC framework, PATRONoC. PATRONoC is subsequently used in a mesh topology and extensively benchmarked to demonstrate the benefits of having AXI-based NoCs. As such, this work makes the following contributions:

- We present an open-source parameterizable AXI-compliant NoC designed for providing high bandwidth links for multi-core DNN computing platforms. The NoC is available at https://github.com/pulp-platform/axi.
- We demonstrate that using an AXI protocol for the NoC creates a fully homogeneous network interface to avoid high cost of protocol translation and provides a standard plug-and-play support for ease of integration.
- We show that using the AXI protocol end-to-end, a multi-channel, wide NoC with burst support and high bandwidth between cores as well as to-and-from memory can be supported, thereby improving performance of DNN applications on multi-core platforms.

The rest of the paper is organized as follows. Section II discusses the architectural overview of the proposed NoC,

[1]In this paper, the terms core and accelerator are used interchangeably.

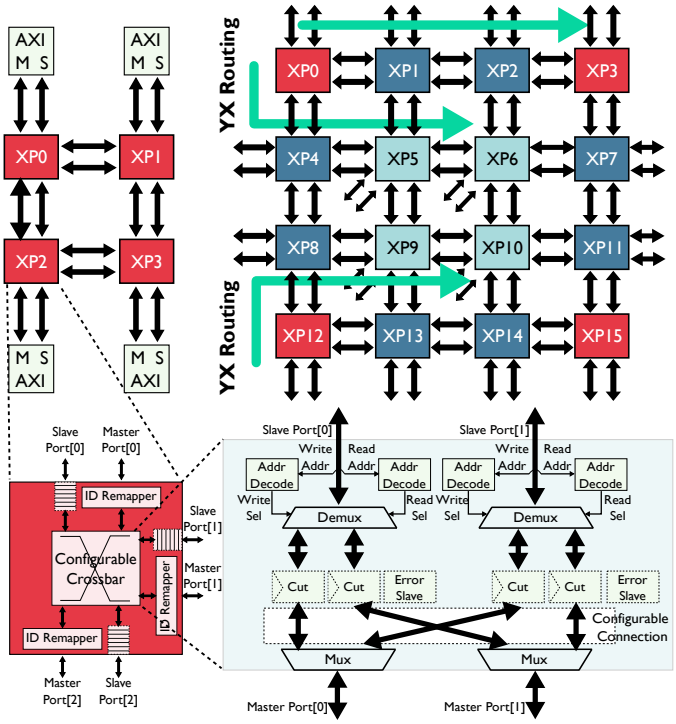

Fig. 1. PATRONoC instances as a 2×2 mesh (left) and a 4×4 mesh (right). The AXI masters and slaves are not shown in the 4×4 mesh for ease of readability. Elementary blocks used for the NoC are also shown: XP (bottom-left) and XBAR (bottom-right). Red XP is 3-master and 3-slave, light blue XP is 4-master and 4-slave, and, dark blue XP is 5-master and 5-slave.

followed by details of the NoC's physical implementation with GlobalFoundries' modern 22FDX technology in Section III. We evaluate our NoC with synthetic and real traffic patterns extracted from DNN workloads in Section IV. Subsequently, we compare our work against other modern NoC solutions in Section V before presenting our conclusions in Section VI.

II. Interconnect Architecture of PATRONoC

This section provides architectural and physical implementation details of PATRONoC for a mesh topology. NoCs are built with many elementary routing elements, each forwarding data from the ingress ports to the egress ports according to their topology-specific routing table. In this work, we extend the AXI crosspoint (XP) from [9], shown in Fig. 1 (bottom), allowing it to be used as PATRONoC's routing element. The XP consists of a configurable crossbar (XBAR) switch and ID remappers to ensure isomorphic XP ports. It is fully AXI-compliant and supports bursts, multiple outstanding transactions, and transaction ordering. We used the XP as the building block for a homogeneous, 2D mesh topology NoC with widely configurable dimensions, as shown in Fig. 1. Although this work uses the 2D mesh as a proof-of-concept, any regular topology, such as a torus, butterfly, or ring, can also be modularly built using our building blocks. We focused on the mesh due to its popularity in research and its remarkable simplicity, scalability, and efficiency [7]. Fig. 1 shows the two mesh topologies, 2×2 (top-left) and 4×4 (top-right), used to evaluate PATRONoC.

The meshes are built by instantiating the XPs in a 2D grid and connecting the NESW-bound links to neighboring

TABLE I
Main parameters of the PATRONoC 2D mesh.

Parameter	Values
Mesh Dimension	N×M
Number of AXI Masters	1 to N×M (default)
Number of AXI Slaves	1 to N×M (default)
Data Width	8 bits to 1024 bits
Address Width	Arch. dependent (32 or 64 bits)
ID Width	1 bit to 16 bits
Max #Outstanding Trans.	1 to 128
XBAR Connectivity	Partial (default) or Fully connected
Register Slice	Single channel or all channels (default)

XPs. AXI masters and slaves can be connected as NoC endpoints at each XP. A common AXI master is a core or a DNN accelerator, and AXI slaves can be memory or I/O tiles. Each XBAR is configured with a static routing table used for deterministic dimension-ordered routing in the mesh. Specifically, PATRONoC uses a source-based YX routing scheme, as shown with the green arrows in Fig. 1, to reduce the complexity of the route calculation step of the crosspoints. In this algorithm, a transaction is first passed forward in the same column until it reaches the same row as the destination XP and then passed forward in the same row until it reaches the destination XP. An automated script generates the address-based routing table for each XP which is used for routing the AXI transactions based on their destination address.

PATRONoC is highly parameterizable, taking advantage of the flexibility of the AXI protocol. The parameters that can be tuned at design time are shown in Table I. The number of AXI masters and slaves indicate the number of connected cores and memory/IO tiles in the design. Both ranges for possible number of AXI masters and slaves are valid for the N×M 2D mesh and are topology-dependent. For example, in a concentrated mesh, multiple masters and slaves can connect to the same XP. Furthermore, the data width (DW) can be tuned to meet the system's bandwidth requirements, while the address width (AW) can be tuned to support a larger global address space.

The AXI protocol identifies transactions with IDs used by the master endpoints to distinguish independent transactions. The number of unique IDs can be configured using the ID width (IW) and increases with the number of masters. All transactions from the same master with the same ID must remain ordered, but there is no ordering requirement between transactions with different IDs. Multiple outstanding transactions enable the master to hide the memory latency. A higher max. number of outstanding transactions (MOT) improves performance, as all AXI building blocks can support multiple in-flight transactions, preventing bandwidth degradation when the NoC is saturated.

The XBAR connectivity parameter configures the XP to either connect all slave ports to all master ports in the case of a fully-connected network or partially connect slaves and masters in the case of a mesh or other non-point-to-point topologies. The last parameter is the register slice (cut), shown in Fig. 1, that can be optionally inserted at design time on some or all AXI channels, improving the timing of the design at the cost of increased latency. The rest of the paper evaluates PATRONoC

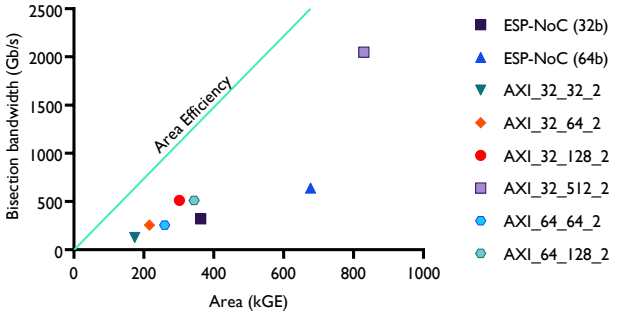

Fig. 2. Implementation results showing area versus bisection bandwidth of PATRONoC and ESP-NoC [10] in 2×2 mesh configurations. PATRONoC's configurations are represented as AXI_AW_DW_IW.

in 2×2 and 4×4 mesh topologies with multiple configurations based on the DW, AW, IW, and MOT parameters.

III. Implementation Results

This section provides implementation results in terms of complexity and scalability of the NoC and its parameters. The implementation is done in GlobalFoundries' 22FDX technology node using a ten-layer metal stack. We used eight-track standard cells of SLVT/LVT type, characterized at worst-case scenario (SS/0.72 V/125 °C) for timing analysis. The designs from Section II are synthesized using Synopsys' Design Compiler 2022.03 in topographical mode, taking physical endpoint placement constraints into account. All designs achieve a clock frequency of 1 GHz at the worst-case condition corner with a register slice on every AXI channel.

The 2×2 PATRONoC mesh, shown in Fig. 1 (top-left), is first synthesized with different AW and DW parameters, keeping IW $= 2$ bits, MOT $= 1$, and other parameters at default values. Fig. 2 shows the area versus bisection bandwidth (DW-dependent) of the mesh for the different configurations. As expected, the design area scales up with increasing AW and DW, taking up mere 174 kGE for the smallest configuration of AW $= 32$ bits and DW $= 32$ bits. The biggest design shown in Fig. 2, with DW $= 512$ bits, takes an on-chip area of 830 kGE.

The benefit of having a homogeneous NoC is evident when the design is compared to classic NoC solutions. This work uses ESP-NoC [10] as our baseline NoC. ESP-NoC is a state-of-the-art open-source packet-based NoC including six planes for coherent and non-coherent traffic for multi-core heterogeneous systems. Synthesis results showing the area of the 2×2 ESP-NoC mesh in its 32-bit- and 64-bit-flit configurations are presented in Fig. 2. Compared to PATRONoC's configuration with AW $= 32$ bits and DW $= 64$ bits, ESP-NoC takes up 68 % more area to provide only 25 % more throughput (five 32-bit wide planes providing 160 Gbit/s). The area overhead can be attributed to ESP-NoC's multiple planes with large protocol translation interfaces at each endpoint. The advantage of PATRONoC is much more evident in Fig. 2 when comparing its area efficiency (slope) with ESP-NoC. We define area efficiency as the bisection bandwidth normalized to the standard cell area, providing a measure of NoC performance at a given complexity. Fig. 2 shows that PATRONoC is closer to the

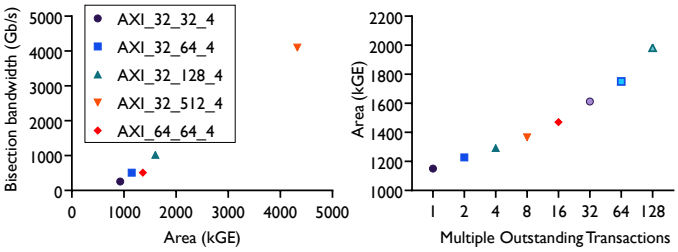

Fig. 3. Implementation results showing area vs. bisection bandwidth of PATRONoC in 4×4 mesh configurations (left). Configurations are represented as AXI_AW_DW_IW. Area vs. MOT tradeoff for DW $= 64$ bits (right).

Pareto front providing better area efficiency compared to the ESP-NoC in 32-bit and 64-bit configurations.

We implement the 4×4 mesh shown in Fig. 1 (top-right) to show the scalability of PATRONoC. For building the 4×4 mesh, the IW of the AXI blocks is increased to 4 to support 16 unique IDs required for 16 masters. The results of the area and bisection bandwidth of this mesh are summarized in Fig. 3 (left). As the mesh dimensions change, the area overhead of the NoC becomes approximately 32 % compared to the 2×2 mesh in similar AW and DW configurations, leading to a drop in area efficiency by 25 %. Increasing the MOT improves the performance of the NoC at the cost of larger complexity in terms of area. Fig. 3 (right) shows the tradeoff between MOT and the area of the 4×4 PATRONoC with DW $= 64$ bits. While this work focuses more on performance and area aspects of the NoC, the power consumption at 1 GHz for the 4×4 PATRONoC is 45 mW (for DW $= 32$ bits) and 171 mW (for DW $= 512$ bits) on uniform random traffic. This accounts for less than 10 % of the projected power consumption of a complete platform, assuming that a typical DNN accelerator connected to one NoC node uses 100 mW to 200 mW.

IV. Performance Evaluation

PATRONoC's performance is characterized in terms of throughput versus injected load through a cycle-accurate register-transfer level (RTL) simulation. This section evaluates the performance of the 4×4 PATRONoC mesh in two configurations: 1) as a slim NoC with DW $= 32$ bits and 2) as a wide NoC with DW $= 512$ bits, both with AW $= 32$ bits, IW $= 4$ bits, and MOT $= 8$. Each master is a DMA engine, and the slaves are AXI-capable memories that cater to the DMA requests. The configurable and workload-specific maximum burst length is used by the RTL model of the DMA engine to create AXI-compliant bursts (adhering to address boundaries and max number of beats) for the NoC. In our evaluation framework, the workload-specific burst length is randomized within a user-defined range to emulate a random burst length with a random source and destination address, while the bursts in the NoC are subject to AXI compliance. All analyses assume a clock frequency of 1 GHz for the endpoints and the NoCs.

A. Uniform Random Traffic

The Noxim simulator [11] is used to set the baseline NoC performance, taking a 4×4 mesh with the default XY routing, 32-bit flits, and eight flits per packet to closely match the slim

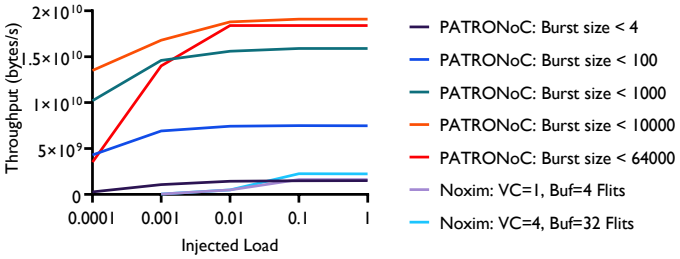

Fig. 4. Uniform Random Traffic with Poisson distribution using Noxim simulator for a 4×4 2D mesh and uniform random traffic on the slim PATRONoC with increasing DMA burst length.

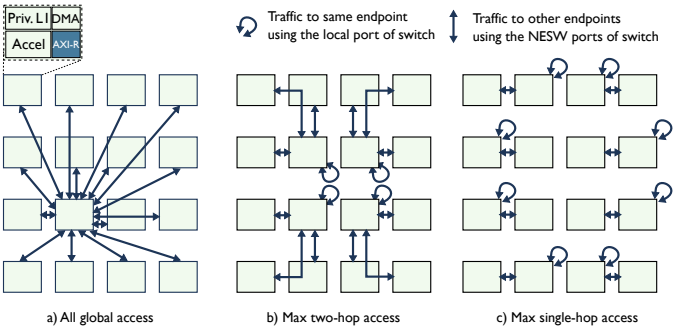

Fig. 5. Synthetic traffic patterns for the performance evaluation.

PATRONoC configurations. Fig. 4 shows the non-exhaustive characterization of the Noxim NoC on this traffic in two configurations: 1) a standard single virtual channel with 4 flits per router buffer for a compact implementation, and 2) four virtual channels with 32 flits per router buffer for high performance. The saturation throughput of the Noxim NoCs are 1.6 GiB/s and 2.25 GiB/s, respectively. Increasing the number of virtual channels (VCs) [12] and flits per buffer improves the NoC's performance, but also increases router complexity.

Fig. 4 also shows the NoC throughput for the uniform random traffic running on the 4×4 slim PATRONoC mesh. It is clear that PATRONoC is beneficial for burst traffic. At small transfer lengths of less than 4 B, similar to normal CPU traffic, PATRONoC performs equivalently to the Noxim NoC with 1.5 GiB/s throughput. However, when using longer bursts, PATRONoC's performance improves and reaches up to 19 GiB/s aggregated throughput at DMA burst lengths up to 10 KiB and 64 KiB. This provides an improvement of 8.4× over the saturation throughput achieved by the best Noxim NoC configuration (4 VCs, buffers 32-flit deep), showing that PATRONoC largely outperforms it by using bursts.

B. Synthetic Traffic

Fig. 5 shows the three synthetic traffic patterns considered: 1) all global access, 2) max two-hop access, and, 3) max single-hop access. We characterize the 4×4 PATRONoC mesh in both slim and wide configurations with the synthetic patterns.

a.) All global access: In this traffic pattern, all the AXI master and DMA endpoints communicate with a single slave endpoint leading to predominately global accesses. Fig. 5a) shows this traffic pattern on the 4×4 mesh, where the endpoint $(2, 1)$ acts as the AXI slave. *b.) Max two-hop access*: In this use case, the AXI slave accesses are distributed to four endpoints $(1, 1)$, $(1, 2)$, $(2, 1)$, and $(2, 2)$. This considers architectures that have a distributed shared L2/L1 memory, either uniform or non-uniform. The 16 AXI masters can communicate to any of the four endpoints, but in this case, the masters are restricted to only communicate to slaves which are a maximum of two hops away. *c.) Max single-hop access*: In this traffic pattern, the AXI slaves are further distributed across eight endpoints along the edges except for the corners. The 16 masters are restricted to access only slaves which are at most one hop away. The last two cases are considered because in traffic from many DNN workloads, data scheduling can be done on nearby cores to prevent long latency and low-performance data communication.

The slim NoC can be used in architectures that are area-constrained but require more throughput than what most traditional NoCs can provide. Fig. 6 (left) shows the NoC utilization, with respect to bisection bandwidth, of the slim NoC on the three synthetic patterns at different burst sizes. Starting with traffic pattern a.), the slim NoC provides a minimum of 1.5 GiB/s of throughput with short bursts. This is approximately 4.7 % NoC utilization considering the slim NoC has a 32 GiB/s bisection bandwidth. The access pattern limits the traffic to a few links of the NoC and, thus, a low utilization is expected. The throughput improves considerably with increasing burst length and reaches a maximum of 6 GiB/s for burst lengths up to 64 KiB, providing a NoC utilization of around 18.75 %. For pattern b.), the NoC performs similarly to pattern a.) for short burst lengths. However, the aggregated throughput improves considerably with larger bursts and saturates at 17.2 GiB/s for burst lengths up to 10 KiB and 64 KiB. This leads to a higher NoC utilization of about 53.75 %, showing that all mesh links can be utilized more efficiently. Similar to pattern b.), the pattern c.) under-performs at small bursts but the aggregated saturation throughput at larger bursts improves to 22.5 GiB/s for bursts up to 64 KiB with a NoC utilization of 70.3 %.

The wide NoC is geared towards high-bandwidth large-burst multi-core DNN-workload traffic. A significant performance gain can be achieved with such wide NoC, but being parameterizable means that also alternative DWs between 32 bits and 512 bits can be considered by designers to find an optimal size for given system requirements. Fig. 6 (right) shows the NoC utilization characteristic of the wide NoC on the synthetic access patterns with different burst sizes. For the traffic pattern a.), the wide NoC can only achieve a utilization of 0.29 % at small bursts up to 4 B large, providing a maximum throughput of 1.5 GiB/s (bisection bandwidth of 512 GiB/s). As seen with the slim NoC, this is an expected performance degradation with this access pattern. The degradation in NoC utilization is further exacerbated by the wide DWs but short burst lengths. The throughput improves, however, with larger burst sizes of up to 64 KiB and reaches saturation at 95 GiB/s with 18.55 % NoC utilization. Both patterns b.) and c.) result in low throughput and utilization with small burst sizes. The aggregated throughput improves at larger bursts with length up to 10 KiB and 64 KiB

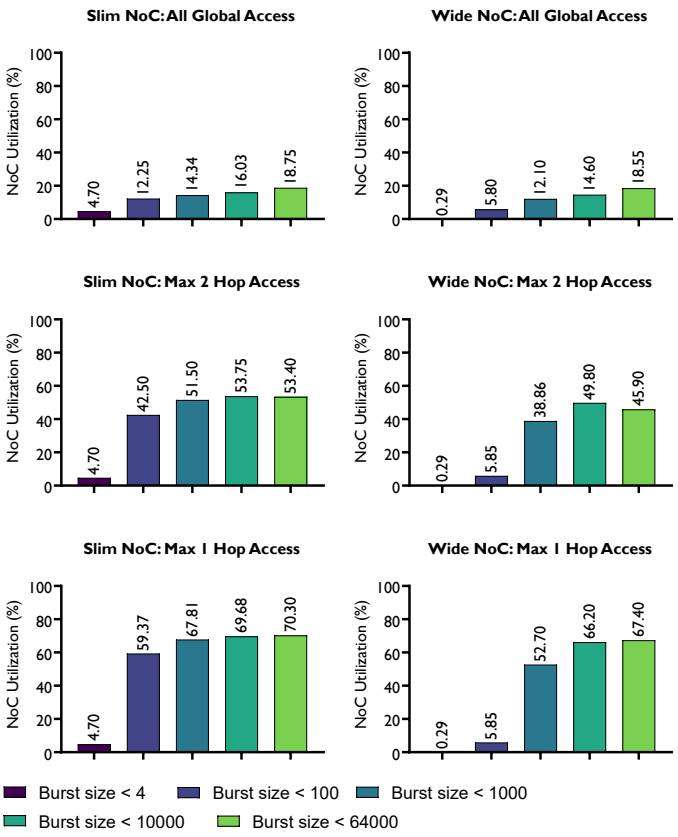

Fig. 6. NoC utilization at maximum injected load for the synthetic random traffic running on the slim and wide PATRONoC using all global access, max 2 hop, and max 1 hop traffic patterns with different DMA burst sizes.

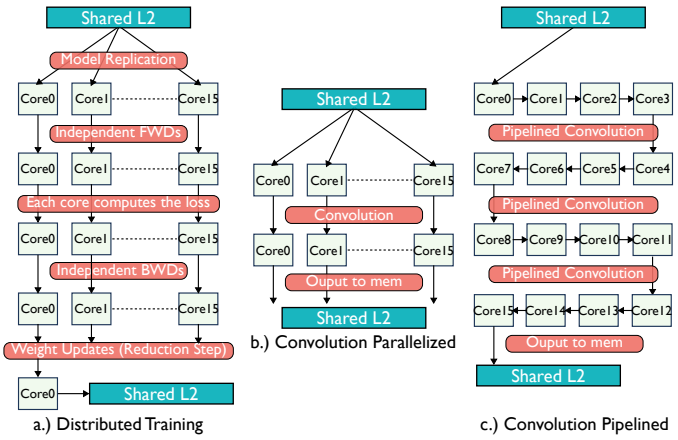

Fig. 7. Overview of the DNN workloads used for PATRONoC evaluation. FWD and BWD in (a) represents the forward and backward propagation workloads, respectively, used in DNN training.

reaching 255 GiB/s (49.8 % utilization) and 345 GiB/s (67.4 % utilization) for the patterns b.) and c.), respectively.

C. DNN Workload Traffic

Synthetic traffic does not capture the full scope of access patterns in real multi-core hardware architectures running DNN workloads. In order to characterize the NoC in more realistic use cases, this section evaluates three emulated CNN-based

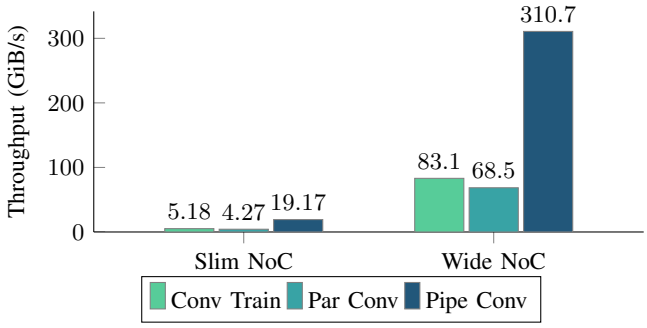

Fig. 8. Throughput analysis for DNN workload traffic on the PATRONoC.

workloads: a.) distributed training, b.) parallelized convolutions, and c.) pipelined convolutions, shown in Fig. 7. We use GVSoC [13] to generate real traffic patterns for the RTL simulation. GVSoC is an open-source, highly configurable, and event-driven simulator for heterogeneous RISC-V-based SoC platforms used for full-system software development and performance evaluation.

a.) Distributed training: For this workload, we replicate and deploy a ResNet-34 (90 % channel shrink factor) distributed training model for the ImageNet dataset on 16 cores. In terms of data communication, a mix of L2 to L1 (core), L1 (core) to L2, and L1 (core) to L1 (core) transfers are needed. *b.) Parallelized convolution* [14]: This is a CNN-based inference workload in which the layers of the network and inputs are tiled and deployed on separate cores. This is a pure L2 to L1 (core) and L1 (core) to L2 memory traffic pattern and has no inter-core communication. *c.) Pipelined convolution* [14]: Depth-first or pipeline dataflow is used in many new DNN platforms to efficiently run CNN-based inference. In this scheme, layers are executed in parallel, in a pipelined way across the different cores to reduce the data traffic to higher memory levels [15]. This workload has mostly L1 (core) to L1 (core) traffic and only cores 0 and 15 do L1 (core) to/from L2 transfers.

Fig. 8 shows the evaluation results of the 4×4 slim and wide NoCs running the three DNN workloads. For the slim NoC, the parallelized convolution—which consists of mostly core to/from shared memory transfers—reaches a throughput of 4.27 GiB/s. For the training workload, the throughput is better than the parallelized convolution workload as it involves a mix of core to/from shared memory and core-to-core transfers. On the pipelined convolution, which consists of predominantly core-to-core traffic, the NoC achieves a high 19.17 GiB/s throughput. Similar trends are reported for the 4×4 PATRONoC wide NoC shown in Fig. 8 (right), but at much higher throughput, with pipeline convolution reaching a peak throughput of 310 GiB/s.

V. Related Work

NoCs are an active area of research, and much effort has gone into optimizing topologies, routing algorithms, flow control schemes, and the microarchitecture of routers [12], [26], [27]. Multi-core (CPU) architectures have been exploiting these optimizations of NoCs for many decades. However, NoCs for multi-accelerator DNN platforms are still in nascent stage.

TABLE II
COMPARISON OF PATRONOC WITH STATE-OF-THE-ART NOCS IN SOCS

Work	Metric				
	Open Source	Full AXI	Burst-support	Config-urable	NoC-BW (Gbps)*
SpiNNaker [16]	×	×	×	×	5 (async)
Reza et al [17]	×	×	×	×	4000
MCM [18]	×	×	×	×	35
MC-NoC [6]	×	×	×	×	2368
NeuNoC [19]	×	×	×	×	-
TETRIS [20]	×	×	×	×	-
PUMA [21]	×	×	×	×	-
OpenSoC [22]	✓	×	×	✓	-
ESP-SoC [4]	✓	×	×	Limited	351
Celerity [23]	✓	×	×	Limited	80
FlexNoC [24]	×	×	×	-	-
Constellation [25]	✓	×	×	✓	-
Andreas et al. [9]	✓	✓	✓	✓	2146
PATRONoC	✓	✓	✓	✓	2700

*Normalized to 1 GHz for fair comparison.

Table II provides a brief overview of state-of-the-art NoCs used in multi-core DNN platforms compared to PATRONoC. PATRONoC is the only design that provides open-source AXI-compliant homogeneous burst-based configurable NoC for multi-core DNN platforms. Moreover, PATRONoC outperforms most of the designs in terms of throughput, with the exception of [17], which uses a bigger 8×8 concentrated mesh (CMesh) topology with primarily local access patterns. Moreover, its results are taken from the gem5 simulator [28], and the RTL of the design is not openly available. Using a CMesh topology for PATRONoC would similarly improve its performance.

OpenSoC Fabric [22] is among the few open-source NoCs with a custom non-coherent NoC protocol. It provides a socket to plug AXI-Lite-based endpoints. Unfortunately, AXI-Lite does not support bursts needed by high-performance systems. The ESP framework [4], [10] also provides an open-source implementation of its multi-plane NoC, supporting coherent and non-coherent endpoints. The NoC is a 2D mesh topology and uses a custom packet-based protocol. We used ESP-NoC as a baseline for comparison with PATRONoC. Section III shows that PATRONoC is more area efficient and provides higher bandwidth owing to its homogeneous network. BaseJump Manycore is an open-source non-coherent NoC based on a 2D mesh used in the Celerity chip [23]. Those NoCs are generally limited to meshes and use classical packet-based NoC protocols, which lead to high area overhead and low bandwidth. In comparison, PATRONoC can be used to design any topology, while providing a highly parameterizable NoC.

VI. CONCLUSION

This work presented the first homogeneous AXI-compliant network-on-chip architecture, building a complete open-source infrastructure for generating various NoC topologies. Using the benefits of a burst-based AXI protocol, PATRONoC targets the emerging field of multi-core DNN platforms requiring high-bandwidth burst-based traffic. The NoC provides high-performance gain compared to state-of-the-art NoCs by using its burst capability and achieves up to a maximum of 310 GiB/s aggregated throughput on DNN workloads. The work provides insight into the exploration of different design parameters which affect the performance and complexity of the NoC. It also enables future work to explore different NoC topologies which might be suited for emerging DNN platforms.

REFERENCES

[1] A. Reuther *et al.*, "AI Accelerator Survey and Trends," in *HPEC*, 2021.
[2] V. Jain *et al.*, "TinyVers: A 0.8-17 TOPS/W, 1.7 μW-20 mW, Tiny Versatile System-on-chip with State-Retentive eMRAM for Machine Learning Inference at the Extreme Edge," in *VLSI*, 2022.
[3] K. Ueyoshi *et al.*, "DIANA: An End-to-End Energy-Efficient Digital and ANAlog Hybrid Neural Network SoC," in *ISSCC*, 2022.
[4] T. Jia *et al.*, "A 12nm Agile-Designed SoC for Swarm-Based Perception with Heterogeneous IP Blocks, a Reconfigurable Memory Hierarchy, and an 800MHz Multi-Plane NoC," in *ESSCIRC*, 2022.
[5] A. Di Mauro *et al.*, "Kraken: A Direct Event/Frame-Based Multi-sensor Fusion SoC for Ultra-Efficient Visual Processing in Nano-UAVs," in *HCS*, 2022.
[6] J.-Y. Kim *et al.*, "A 118.4 GB/s Multi-Casting Network-on-Chip With Hierarchical Star-Ring Combined Topology for Real-Time Object Recognition," *JSSC*, 2010.
[7] J. Balfour *et al.*, "Design Tradeoffs for Tiled CMP On-Chip Networks," in *ICS*, 2006.
[8] D. Petrisko *et al.*, "NoC Symbiosis (Special Session Paper)," in *NOCS*, 2020.
[9] A. Kurth *et al.*, "An Open-Source Platform for High-Performance Non-Coherent On-Chip Communication," *IEEE TComp*, 2022.
[10] D. Giri *et al.*, "NoC-Based Support of Heterogeneous Cache-Coherence Models for Accelerators," in *NOCS*, 2018.
[11] V. Catania *et al.*, "Noxim: An open, extensible and cycle-accurate network on chip simulator," in *ASAP*, 2015.
[12] W. J. Dally *et al.*, *Principles and practices of interconnection networks*. Elsevier, 2004.
[13] N. Bruschi *et al.*, "GVSoC: A Highly Configurable, Fast and Accurate Full-Platform Simulator for RISC-V based IoT Processors," in *ICCD*, 2021.
[14] N. Bruschi *et al.*, "Scale up your In-Memory Accelerator: Leveraging Wireless-on-Chip Communication for AIMC-based CNN Inference," in *AICAS*, 2022.
[15] K. Goetschalckx *et al.*, "DepFiN: A 12nm, 3.8TOPs depth-first CNN processor for high res. image processing," in *VLSI*, 2021.
[16] E. Painkras *et al.*, "SpiNNaker: A 1-W 18-Core System-on-Chip for Massively-Parallel Neural Network Simulation," *IEEE JSSC*, 2013.
[17] M. F. Reza *et al.*, "Energy-Efficient and High-Performance NoC Architecture and Mapping Solution for Deep Neural Networks," in *NOCS*, 2019.
[18] B. Zimmer *et al.*, "A 0.32–128 TOPS, Scalable Multi-Chip-Module-Based Deep Neural Network Inference Accelerator With Ground-Referenced Signaling in 16 nm," *JSSC*, 2020.
[19] X. Liu *et al.*, "Neu-NoC: A high-efficient interconnection network for accelerated neuromorphic systems," in *ASP-DAC*, 2018.
[20] M. Gao *et al.*, "TETRIS: Scalable and Efficient Neural Network Acceleration with 3D Memory," *SIGARCH Comput. Archit. News*, 2017.
[21] A. Ankit *et al.*, "PUMA: A programmable ultra-efficient memristor-based accelerator for machine learning inference," in *ASPLOS*, 2019.
[22] F. Fatollahi-Fard *et al.*, "OpenSoC Fabric: On-chip network generator," in *ISPASS*, 2016.
[23] S. Davidson *et al.*, "The Celerity Open-Source 511-Core RISC-V Tiered Accelerator Fabric: Fast Architectures and Design Methodologies for Fast Chips," *MICRO*, 2018.
[24] J.-J. Lecler *et al.*, "Application driven network-on-chip architecture exploration & refinement for a complex SoC," *DAES*, 2011.
[25] J. Zhao *et al.*, "Constellation: An Open-Source SoC-Capable NoC Generator," in *NoCArc*, 2022.
[26] N. E. Jerger *et al.*, "On-chip networks," *Synthesis Lectures on Computer Architecture*, 2017.
[27] G. De Micheli *et al.*, "Networks on chips: From research to products," in *Proceedings of the 47th Design Automation Conference*, 2010.
[28] J. Lowe-Power *et al.*, "The gem5 Simulator: Version 20.0+," *CoRR*, 2020. [Online]. Available: https://arxiv.org/abs/2007.03152