Multi-Mode Borderguard Controllers for Efficient On-Chip Communication in Heterogeneous Digital/Analog Neural Processing Units

Hong Pang*◇, Carmine Cappetta†, Riccardo Massa‡, Athanasios Vasilopoulos§, Elena Ferro*§, Gamze Islamoglu*, Angelo Garofalo*¶, Francesco Conti¶, Luca Benini*¶◇, Irem Boybat§◇, Thomas Boesch‖◇
*ETH Zurich, Switzerland †STMicroelectronics, Cornaredo, Italy ‡STMicroelectronics, Agrate, Italy
§IBM Research - Zurich, Switzerland ¶University of Bologna, Italy ‖STMicroelectronics, Geneva, Switzerland
◇{hopang,lbenini}@iis.ee.ethz.ch, ibo@zurich.ibm.com, thomas.boesch@st.com

***Abstract*—Driven by the growing demand for data-intensive parallel computation, particularly for Matrix-Vector Multiplications (MVMs), and the pursuit of high energy efficiency, Analog In-Memory Computing (AIMC) has garnered significant attention. AIMC addresses the data movement bottleneck by performing MVMs directly within memory, significantly reducing latency and enhancing energy efficiency. Integrating AIMC with digital units for non-MVM operations yields heterogeneous Neural Processing Units (NPUs) that can be combined in a tiled architecture to deliver promising solutions for end-to-end AI inference. Besides powerful heterogeneous NPUs, an efficient on-chip communication infrastructure is also pivotal for inter-node data transmission and efficient AI model execution. This paper introduces the Borderguard Controller (BG-CTRL), a multi-mode, path-through routing controller designed to support three distinct operating modes—time-scheduling, data-driven, and time-sliced data-driven (TSDD)—each offering varying levels of routing flexibility and energy efficiency depending on the data flow patterns and AI model complexity. To demonstrate the design, BG-CTRLs are integrated into a 9-node system of heterogeneous NPUs, arranged in a 3x3 grid and connected using a 2D mesh topology. The system is synthesized using STM 28nm FD-SOI technology. Experimental results show that the BG-CTRL cluster achieves an aggregate throughput of 983 Gb/s, with an energy efficiency of up to 0.41 pJ/B/hop at 0.64 GHz, and a minimal area overhead of 204 kGE.**

***Index Terms*—Heterogeneous Neural Processing Unit, Data Communication, Analog In-Memory Computing, Edge AI**

I. Introduction

In the era of advanced edge AI applications, the growing demand for data-intensive parallel computation, particularly for Matrix-Vector Multiplication (MVM), alongside the need for high energy efficiency, has propelled the development of specialized neural processing accelerators. Analog In-Memory Computing (AIMC) has emerged as a key technology in this area, exploiting the row parallelism of memory arrays to perform MVMs directly in the analog domain where the data resides. This approach is well-suited for AI model inference, where MVMs typically account for >90% of the overall operations. Non-volatile memory-based AIMC is particularly attractive for high on-chip weight storage capacity [1].

Multiple AIMC tiles are often employed on AIMC-based DNN accelerators to maximize the on-chip weight capacity [1], [2]. The AIMC tiles can contain lightweight digital logic for simple operations [3]. However, to enable end-to-end AI inference, AIMC tiles are often complemented by digital accelerators, general-purpose cores, and memory units within a heterogeneous system architecture, which handles non-MVM operations [2]. Figure 1(a) illustrates such a heterogeneous system, or *Neural Processing Unit (NPU)*, comprising multiple AIMC tiles and other digital units.

To ensure high energy efficiency at the system level for AIMC tiles, an efficient communication infrastructure is crucial. A circuit-switched 2D mesh communication can cater to this need, while also supporting data operations such as multicasting [2]. Moreover, a circuit-switched approach reduces hardware overhead compared to packet-switching, making it well-suited for area-constrained edge AI systems. At the same time, ensuring routing flexibility to meet the diverse demands of AI models while maintaining low energy consumption is an important consideration for such edge systems.

To this end, we propose the Borderguard Controller (BG-CTRL), a multi-mode, path-through routing controller in a circuit-switched approach. It supports three distinct operating modes, namely *time-scheduling*, *data-driven*, and *time-sliced data-driven (TSDD)*, each catering to different balance of routing flexibility and energy efficiency. Time-scheduling mode activates data paths during specific intervals based on a specialized instruction set, handling complex and irregular inter-node communication patterns. Data-driven mode establishes fixed data transmission paths for the entire runtime, maximizing energy efficiency. TSDD mode extends data-driven approach by dividing time into multiple slices within a period, activating different routing paths in each slice. The path-through capability ensures a consistent transmission latency of just one clock cycle, regardless of the hop count (i.e., the number of intermediate nodes that data traverses).

We provide a detailed study of the proposed BG-CTRL by constructing a 9-node system, organized as a 3-by-3 grid, interconnected with a 2D mesh. Inter-node routing is facilitated by borderguards (data multiplexer), with each node equipped with BG-CTRLs and FIFOs for data storage. This system is synthesized using STM 28nm FD-SOI technology under worst-case conditions (SS/0.9 V/-40°C).

978-3-9826741-0-0/DATE25/© 2025 EDAA

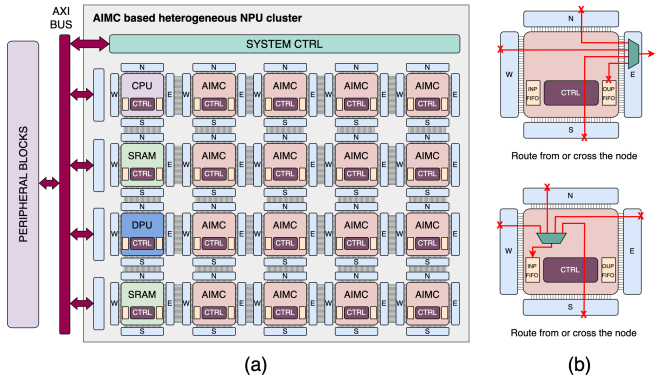

Fig. 1. (a) Heterogeneous *Neural Processing Unit* system architecture. (b) *Borderguard* connectivity with other nodes and FIFOs, with one multiplexer shown in each scenario for clarity.

The contributions of the paper are the following:

1) We developed a path-through routing controller with three operating modes: time-scheduling, data-driven, and TSDD.
2) We designed a comprehensive instruction set for time-scheduling and TSDD modes to handle complex data flow.
3) We integrated the BG-CTRL cluster into an accelerator node by designing a custom node wrapper, and provided a comprehensive explanation of the node wrapper's microarchitecture.
4) We evaluated the BG-CTRL design, showing an aggregate throughput of 983 Gb/s and an energy efficiency of up to 0.41 pJ/B/hop at 0.64 GHz, with a minimal area overhead of 204 kGE. Additionally, we provided a comparative analysis against various state-of-the-art designs.

II. Related Work

Two widely studied schemes for inter-node communication are circuit-switching and packet-switching. Packet-switching segments data into flits that may traverse different routes to the destination, offering high resource utilization and routing flexibility [4]–[7]. However, this method incurs significant hardware overhead for packet management and implementing specific routing algorithms [8].

On the other hand, circuit-switching, while less flexible than packet-switching, simplifies control logic and ensures low latency by establishing dedicated paths before transmission, and has garnered increasing research interest [2], [4], [9]–[13]. For instance, [11] leverages a centralized, configurable switch to manage data streams between sources and sinks, using a backpressure mechanism to regulate flow. While data paths can be reconfigured by updating configuration registers, the centralized switching model leads to issues like unbalanced workload distribution and high routing complexity as the network grows, limiting its scalability in larger systems. One approach to overcome the scalability is to employ multiple connected domains, each of which contains a centralized switch [12]. However, data traffic between domains can create bottlenecks, requiring careful task mapping to minimize such traffic and optimize performance. A similar communication scheme is adopted in [13], where shared memory is employed within each domain, and the program manages both computation and data-driven inter-domain communication. In [10], the compute nodes are organized in a hierarchical fashion, with a concentrated mesh-based statically-scheduled communication adopted on the inter-domain level. Nonetheless, this concentrated mesh relies on a centralized router shared across multiple tiles, which can lead to bandwidth contention and complex traffic management. To address scalability issues while leveraging circuit-switched-based communication, a fully distributed system can be adopted. In [2] computation nodes are organized in a rectangular grid and interconnected with a high-bandwidth 2D mesh. Each node contains a time-scheduling local controller with a custom instruction set. Yet, the programmable controller handles both control and routing, potentially leading to reduced flexibility or trade-offs in optimization for certain workloads.

In response to these shortcomings, the BG-CTRL proposed in this paper offers a distributed circuit-switched routing solution that supports both data-driven and time-scheduling approaches. The BG-CTRL is detached from the host core of the node, and provides greater flexibility by allowing each BG-CTRL to function as an independent block for each input/output port. Additionally, the ability to enable or disable individual BG-CTRLs improves power efficiency, making this solution particularly well-suited for resource-limited edge AI systems.

III. Multi-Mode Borderguard Controller

We focus on the heterogeneous NPU system shown in Fig. 1(a), where the nodes are organized in a rectangular grid and are interconnected via a 64-bit 2D mesh. Each of the nodes consist of an accelerator or memory unit, wrapped with a digital node wrapper for control and communication logic. The routing is done through multiplexer-based *borderguards*, as shown in Fig. 1(b). To accommodate the constraints of resource-limited edge AI systems, a small FIFO configuration is adopted, with a parametric depth and data width.

In such an NPU, the data path consists of source, intermediate, and destination nodes. The source and intermediate nodes output data, while the destination node receives it. To facilitate this data transmission, two types of BG-CTRLs are designed, as illustrated in Fig. 2(a): the *data-out BG-CTRL (DOBC)* and *data-in BG-CTRL (DIBC)*. The DOBC handles data forwarding between BGs and connects to output FIFOs (OFIFOs) for data popping (node-to-mesh), while the DIBC manages data pushing into input FIFOs (IFIFOs) (mesh-to-node). Furthermore, the BG-CTRL is designed with high versatility, supporting various network topologies such as mesh, ring, crossbar, and tree, thanks to its distributed architecture.

Figure 2(b) presents the BG-CTRL architecture, where the DOBC and DIBC are nearly identical except for their respective data popping/pushing control logic. Each BG-CTRL supports three operating modes: 1) time-scheduling, 2) data-driven, and 3) TSDD. Based on configuration registers (data-driven mode) or instructions (time-scheduling or TSDD mode), BG-CTRL generates a data selection signal, driving the borderguard output. The input data sources for the multiplexer include the four sides (West, North, East, South) and the OFIFOs.

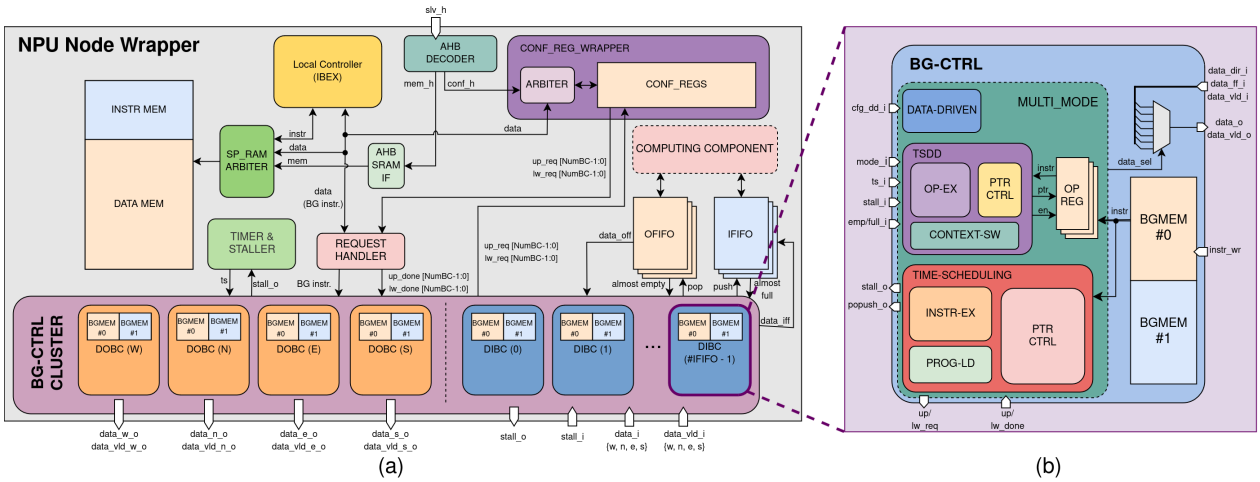

Fig. 2. (a) Integration of the BG-CTRL cluster to the Neural Processing Unit node wrapper. The connection between host core and computing component omitted for simplicity. (b) Multi-mode BG-CTRL architecture.

A. Time-Scheduling Mode

In time-scheduling mode, the data path is predefined and becomes active at a specific time, referred to as the *timestamp*. A Timer block reports the current timestamp value. To maintain synchronization across all nodes, the global stall signal, which is triggered when any source or destination is not ready, prevents the Timer from updating. The time-scheduled controller checks the global stall signal before executing any instruction.

Within the time-scheduling block in Fig. 2(b), the instruction pointer controller (PTR CTRL) prepares the address and accesses the code memory once an instruction is completed. The executor (INSTR-EX) then executes the instruction at the designated timestamp. Due to limited code memory (BGMEM) capacity, the entire data path scheduling program may not fit in memory. To address this, the program loader (PROG-LD) monitors execution progress and reloads the remaining instructions when any memory bank is fully accessed. The memory is divided into two banks, allowing one bank to be reloaded while the other is accessed for execution, effectively hiding the reloading latency. To facilitate complex data path formation while minimizing program size, a comprehensive set of instructions is provided, as detailed in Table I. Each instruction, except those that set the timestamp, is associated with an active timestamp for execution.

B. Data-Driven Mode

Instead of instructions, data-driven mode uses configuration registers (`op` for operation, `dir` for data source, and `dest` for transmission destination) to set up a fixed routing path for the entire runtime. The DOBC supports both data popping and forwarding, while the DIBC is dedicated to data pushing. For the DOBC, all three registers must be configured: `dir` specifies either the side from which data arrives (for forwarding) or the OFIFO (for popping). For the DIBC, `dest` is not needed, as the DIBC is already positioned in the destination node. In data-driven mode, the BG-CTRL transmits data whenever the source data is valid and the destination IFIFO is ready, independent of timestamps or time slices.

C. Time-Sliced Data-Driven Mode

In TSDD mode, the execution period is divided into *time slices*, each corresponding to a clock cycle. A specific instruction, such as data forwarding or popping/pushing, is assigned to each slice. The Timer broadcasts these time slices in a round-robin manner, advancing every clock cycle regardless of whether the corresponding instruction is successfully executed. Table II outlines the instructions available for TSDD mode, where DEST indicates the destination node index, and CNT specifies the number of data units to be transmitted, with 0 signifying infinite data transmission.

In Fig. 2(b), each operation register (OP REG) stores an instruction designated for a specific time slice. PTR CTRL activates the appropriate operation register based on the current time slice, after which the instruction is sent to the operation executor (OP-EX) for execution.

The context switching block (CONTEXT-SW) updates the instruction at a specific time slice once the current instruction has been fully executed. This feature enables model partitioning and application switching without requiring a system reset.

D. Selection of the Operating Mode

The three operating modes discussed illustrate a trade-off between routing flexibility and energy efficiency, each suited for different scenarios. The data-driven mode is most effective for static, well-pipelined inter-node communication and is the most energy-efficient of the three. However, it does not permit multiple data paths to share the same link, which can lead to conflicts. The TSDD (Time-Sliced Data-Driven) mode solves this by assigning different data paths to separate time slices, offering greater flexibility but potentially reducing throughput if the active time slice does not align with actual data arrival. For more complex applications with irregular and dynamic data transmission patterns, the time-scheduling mode offers superior control by managing data flow in fine-grained timestamps and

TABLE I
INSTRUCTION SET AND FORMAT FOR TIME-SCHEDULING MODE.

Operation INST[23:20]	**Field 2** INST[19:16]	**Field 1** INST[15:12]	**Field 0** INST[11:0]	**Functionality**
SET_TS	TS	TS	TS	Configures the upper 20 bits of the active timestamp (0 by default).
SET_OTS	NA	NA	Offset TS	Configures the implicit active offset timestamp value (1 by default).
INC_TS	NA	NA	TS	Increments the upper 20 bits of the active timestamp by 1.
FWIM	DIR	NA	TS	Forwards data from the specified source (DIR).
FW	DIR	NA	Offset TS	DIR: OFIFO ID (for data popping) or side (for data forwarding/pushing).
POPUSHIM	RP	RP	TS	DOBC: Pops data from an OFIFO; DIBC: Pushes data into an IFIFO.
POPUSH	RP	RP	Offset TS	
REPEATIM	NR	RP	TS	Forms a loop with NR instructions, repeating for RP iterations.
REPEAT	NR	RP	Offset TS	RP = 0 indicates infinite loop.
REPEATL	NR[9:6]	RP[9:6]	{NR[5:0], RP[5:0]}	`REPEATL` active timestamp is set using `SET_OTS`.
WAITIM	NA	NA	TS	Keeps BG-CTRL idle, with data-selecting signal unchanged.
WAIT	NA	NA	Offset TS	
RESTART	RP	RP	TS	Reruns the entire program for RP iterations. (RP = 0 indicates infinite iterations.)
DONE	NA	NA	TS	Signals the end of the program.

NA: Unused field TS: Active timestamp DIR mapping configuration: 0: West, 1: North, 2: East, 3: South, (N+4): OFIFO #N

TABLE II
INSTRUCTION SET AND FORMAT FOR TSDD MODE.

Operation INST[23:20]	**Field 2** INST[19:16]	**Field 1** INST[15:12]	**Field 0** INST[11:0]
FW	NA	DIR	CNT
POP	DEST	DIR	CNT
PUSH	NA	DIR	CNT
WAIT	NA	NA	NA
END	NA	NA	NA

supporting multi-level loop nesting. This mode can maximize performance when the NPU's computing latency is predictable; otherwise, a safety margin can be introduced for active timestamps. However, time-scheduling consumes the most power compared to the other two modes.

IV. INTEGRATION OF BG-CTRL INTO THE NPU NODE

The BG-CTRL is dedicated solely to managing data communication on the NPU. To handle the rest of the control tasks and proper execution of the computing component (accelerator), other units are employed within the node wrapper.

The host core handles the main control flow within the node. This unit is realized with an IBEX core [14], an open-source 32-bit RISC-V CPU. A compact configuration is adopted for the NPU, utilizing the 32-bit RV32IMC instruction set with a 2-stage pipeline and a 3-cycle multiplier. This configuration achieves a performance of 2.47 CoreMark/MHz with an area of 26.60 kGE, synthesized using the Yosys toolchain [14]. The host core is responsible for loading routing and computing programs onto the BG-CTRLs and computing components, respectively. Additionally, it handles program reloading requests from the BG-CTRLs when the mapped application size exceeds the code memory capacity of the BG-CTRLs.

In addition to the BG-CTRL cluster and the host core, the node wrapper includes configuration registers, scratchpad memory, a request handler, IFIFOs and OFIFOs. The configuration registers store essential runtime information, such as the BG-CTRL operating mode, data-driven configuration for each BG-CTRL and control states. The scratchpad memory consists of an instruction memory for host core programs and data memory for BG-CTRL programs and the computing component. The request handler functions as an intermediary between the host core and the BG-CTRL cluster, forwarding program loading requests from the host core to the appropriate BGMEM.

In the adopted mesh topology, four DOBCs—one for each side—and at least one DIBC for data pushing are required. Multiple IFIFOs (and corresponding DIBCs) help alleviate traffic congestion and enable data multicasting, providing data flow flexibility. In time-scheduling mode, strict global synchronization across nodes ensures that data ejected from an OFIFO can reach all nodes simultaneously. As a result, multiple IFIFOs, whether in the same node or across different nodes, can concurrently receive the same data stream, as long as their respective DIBCs execute the `POPUSH` instructions at the same timestamp as the source DOBC. In TSDD and data-driven modes, data multicasting is supported as well, but only for IFIFOs located within the same node.

V. EXPERIMENTAL METHODOLOGY

A 9-node architecture, arranged as a 3-by-3 grid and interconnected with a 2D mesh, is developed using register transfer level (RTL) models, and simulated and benchmarked in Siemens/Mentor QuestaSim 2021.3. The design has been synthesized using Synopsys Design Compiler 2021.06 for the STM FD-SOI 28nm technology under worst-case conditions (SS/0.9V/-40°C) to extract area results. Unless otherwise specified, a 640 MHz clock frequency and 156 ps I/O delays are applied. Moreover, since the data path is combinational, any data paths where the source node is traversed again later must be avoided to prevent deadlocks during static timing analysis (STA). Under typical-case conditions (TT/0.9V/25°C), the design achieves timing closure at a frequency of 820 MHz. Power measurements are obtained with Synopsys PrimeTime 2022.3, utilizing post-synthesis simulation under typical-case conditions at a frequency of 640 MHz. The design parameters for BG-CTRL are detailed in Table III, with the *Nested Loop Depth* parameter specifying the maximum nesting level of loops in time-scheduling mode.

TABLE III
BG-CTRL DESIGN PARAMETERS CONFIGURATION.

Parameter	Setting	Parameter	Setting
# Mesh Rows	3	# Mesh Columns	3
# MO-FIFOs	4	# MI-FIFOs	3
# DOBC	4	# DIBC	3
Fifo Depth	32	Link Data Width	64 bits
BGMEM Access Latency	1 cycle	BGMEM Depth	256
Instruction Width	24 bits	Nested Loop Depth	5
Program Reloading	Enabled	Context Switching	Enabled

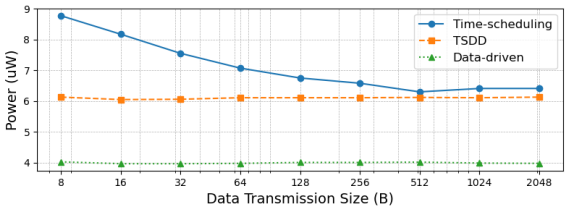

Fig. 3. Power consumption under multiple modes with data transmission of size varying from 8 B to 2 kB

VI. RESULTS

A. Latency and Bandwidth

The zero-load access latency from one node to an adjacent node is measured. In both time-scheduling and TSDD modes, the combined latency, including instruction execution and data transmission, totals 3 cycles, assuming both the data source and destination are ready with no stalls. Specifically, the first cycle is allocated to determining the data source, the second cycle assesses the FIFO's almost full/empty status, and the third cycle executes the data transmission. Despite this delay, it remains transparent to users, allowing data transmission to be scheduled as if each single-flit transmission is completed in 1 cycle due to the well-pipelined stages. Conversely, in data-driven mode, the entire transmission is completed in just 1 cycle. Additionally, thanks to the path-through nature of the router design, the transmission latency remains constant regardless of the number of hops along the data path. When connected via two sets of unidirectional 64-bit links, the interconnect achieves a peak link bandwidth of 41 Gb/s (82 Gb/s duplex) while operating at 640 MHz. Configured in a 3x3 mesh topology, the aggregate bandwidth reaches 983 Gb/s, utilizing all twelve duplex links (128-bit) in concurrent transmission.

B. Power and Energy Efficiency

Post-synthesis power simulation is conducted under typical-typical corner to obtain the power consumption of the BG-CTRLs during data transfers ranging from 8 B to 2 kB. The analysis focuses on a data path comprising three nodes: a source node, an intermediate node, and a destination node. To accurately assess the total power consumption during data transmission, the power consumed by BG-CTRL clusters in all the three active nodes is aggregated for the final result. Fig. 3 shows a clear power efficiency advantage for the Data-Driven mode, with a consistent consumption of approximately 4 µW (0.67x TSDD mode), regardless of the transmission size. Conversely, the Time-Scheduling mode exhibits a decreasing power trend, starting at approximately 9 µW for smaller data size and stabilizing around 6 µW (1.05x TSDD mode) for larger transfers. The rationale behind this trend is that, in Time-Scheduling mode, a single `POPUSH` instruction can manage up to 255 data flits (8B each). As a result, regardless of the number of flits being transferred (up to 255), the BG-CTRL only accesses the BGMEM once. Consequently, as the number of transferred data flits increases, the BGMEM access power is spread over a larger workload, effectively reducing its impact on the overall power consumption.

To further explore the decreasing power trend observed in Time-Scheduling mode, Fig. 4 provides a breakdown of the power consumption for the nodes along the data path, including both the IFIFOs/OFIFOs and the BG-CTRL clusters. Active BG-CTRL refers to the BG-CTRL managing data communication, while CTRL represents its internal control logic. As illustrated in Fig. 4(a), during 8B data transfer, active BG-CTRLs consume over half of the total cluster energy, with nearly 40% attributed to BGMEM. This high power consumption is attributed to low instruction utilization in small data transfers, leading to significant energy consumption over a short transmission period. However, as the data size increases to 1kB (Fig. 4(b)), the power impact of BGMEM access diminishes, diluted by the larger data volume. Additionally, reduced switching activity in the active BG-CTRLs due to the static data path lowers control logic power consumption. Consequently, the power contribution of active BG-CTRLs is 4.3x lower than in the 8B case, with active BG-CTRLs consuming only 3% more energy than inactive ones, demonstrating the power efficiency advantage of Time-Scheduling mode in large data transfers.

Fig. 5 illustrates the impact of loop control mechanism on power consumption of the time-scheduled controllers, where 1 kB of data is transmitted across varying loop nesting levels, with the total number of iterations held constant. The total power accounts for the combined consumption of time-scheduled controllers along the previously defined data path. As shown in the figure, data transmission with a single non-nested loop and a one-level nested loop consumes 1.15x and 1.23x more power, respectively, compared to the baseline (transmission without loops). However, the additional power consumption introduced by higher-level nested loops is minimal. Despite this power overhead, loops effectively reduce program size, thereby decreasing the frequency of program reloading events, which are highly power-intensive.

C. Area

The area distribution between DOBC (28.82 kGE) and DIBC (28.49 kGE) is quite similar, differing primarily in the control logic for data popping and pushing. Figure 6 presents the area breakdown of DOBC in terms of kilo-gate equivalent (kGE). Notably, the BGMEM occupies more than half of the total DOBC area. The data-driven controller occupies only 1% of the DOBC area, owing to its simple protocol. The time-scheduling controller is roughly twice the size of the TSDD controller. The remaining 8% of the area is used for clock gating (CG). Within the time-scheduling controller, the PTR CTRL is the

TABLE IV
COMPARISON OF THIS WORK WITH THE STATE-OF-THE-ART.

	Ref [4]	Ref [5]	Ref [6]	Ref [2]	**This Work**
Circuit/packet switched	Circuit + Packet	Packet	Packet	Circuit	Circuit
Process	32nm	32nm	12nm	14nm	28nm
Frequency (GHz)	0.5[a], 1.75[b]	0.5	1.23[t]	1	0.64[s]
Link DW (bits)	18[a], 32[b]	64	512	512	64
Link Peak BW (Gb/s)	9.1[a], 57[b]	32	629[t]	171	41[s]
Data Packeting/Conversion	Required	Required	Required	Not Required	**Not Required**
Overhead flit fraction[c]	2/(n+1)	0	N.R.	0	**0**
Packet Peak Throughput (flits)	0.33/cycle	0.33/cycle	1/cycle	0.33/cycle	**1/cycle**
Min. Latency (cycles)[c]	2h+n+1[a], N.R.[b]	h+n+t+1	4h+n+4	n	**n**
Area (kGE)	N.R.	N.R.	504	N.R.	**204**
Power (mW)	N.R.	N.R.	5.23[g]	N.R.	**0.0152**[d], **0.0134**[e], **0.0125**[f]
Energy Efficiency (pJ/B/hop)	N.R.	N.R.	0.30[g]	0.36-1.01[h]	**0.41**[d], **0.38**[e], **0.24**[f]
Deterministic Routing Algorithm	N.R.	Dimension-Ordered	XY + Table-based	Arbitrary	**Arbitrary**
Data Driven	✓	×	×	×	✓
Data Multicasting	×	×	×	✓	✓

[a] Wormhole, [b] Circuit switched, [c] h (hops), n (data flits), t (turns), [d] Time-scheduling, [e] TSDD, [f] Data-Driven, [g] Normalized value
[h] with energy consumed by routing control (instruction execution) not included, [t] Estimated under typical corner, [s] Estimated under slow corner

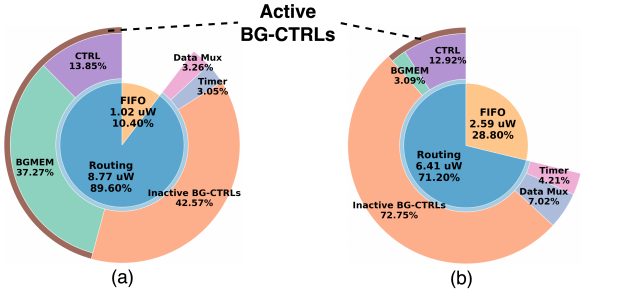

Fig. 4. Power breakdown of BG-CTRL under time-scheduling mode for data transmission of (a) 8B and (b) 1kB.

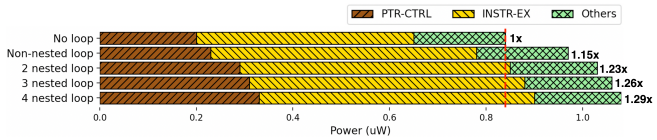

Fig. 5. Power breakdown of the time-scheduling block for data transmission of 1 kB with different loop levels.

largest block, occupying nearly 70% of the controller's area due to the complex control mechanisms required to support multi-level loops. In contrast, the PTR CTRL in the TSDD controller is simpler, as it does not manage loop control. Additionally, OP-EX in the TSDD controller is significantly smaller than in the time-scheduling controller, reflecting the simpler instruction set in TSDD mode.

D. Comparison with prior art router designs

A comparative overview of prior art is provided in Table IV, including both packet-switched and circuit-switched prior art, assuming single-direction transmission. We also consider designs implemented with similar processes. A notable feature in our approach is the integration of data source/destination within the instructions or configuration registers, eliminating the need for data packeting and conversion within the node. This feature results in zero header/tail overhead and minimal accessory hardware, with area overhead being 2.5x smaller than

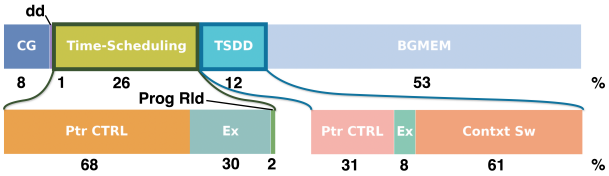

Fig. 6. Area breakdown of the BG-CTRL for a DOBC (28.82 kGE).

[6]. Additionally, the BG-CTRL is up to 2.5x more energy efficient than the circuit-switched approach presented in [2], although [2] includes more complex features such as fine-grained transmission. Furthermore, BG-CTRL is highly flexible, supporting a wide range of deterministic routing algorithms—or even hybrid combinations running concurrently—ensuring high adaptability across diverse data transmission scenarios.

VII. CONCLUSION

We proposed BG-CTRL, a multi-mode, path-through routing controller designed to efficiently manage data transmission for heterogeneous NPU systems, balancing flexibility and energy efficiency. To demonstrate our design, the BG-CTRL cluster is integrated into a 3x3 mesh structure, achieving an aggregate throughput of 983 Gb/s, with energy efficiency reaching up to 0.41 pJ/B/hop at 0.64 GHz, and a minimal area overhead of 204 kGE, all without the need for additional data-packeting or conversion circuits. Additionally, BG-CTRL can be programmed with arbitrary routing algorithms, making it adaptable to a wide range of application requirements.

ACKNOWLEDGMENT

This project is supported by European Union (Horizon Europe Grant Agreement n°101070634), Swiss State Secretariat for Education, Research and Innovation (SERI) under contracts number SBFI 22.00202 and 23.00205 and UK Research and Innovation (UKRI) under the UK government's Horizon Europe funding guarantee [grant number 10040829]. We thank K. G. Karunaratne, I. Sanli, W. Simon, E. Zelioli for technical discussions and A. Sebastian for management support.

REFERENCES

[1] M. Le Gallo et al., “A 64-core mixed-signal in-memory compute chip based on phase-change memory for deep neural network inference,” Nature Electronics, pp. 1–14, 2023.

[2] S. Jain et al., ”A Heterogeneous and Programmable Compute-In-Memory Accelerator Architecture for Analog-AI Using Dense 2-D Mesh,” in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 31, no. 1, pp. 114-127, Jan. 2023.

[3] E. Ferro et al., ”A Precision-Optimized Fixed-Point Near-Memory Digital Processing Unit for Analog In-Memory Computing,” 2024 IEEE International Symposium on Circuits and Systems (ISCAS), Singapore, Singapore, 2024, pp. 1-5.

[4] B. Bohnenstiehl et al., ”KiloCore: A 32-nm 1000-Processor Computational Array,” in IEEE Journal of Solid-State Circuits, vol. 52, no. 4, pp. 891-902, April 2017

[5] M. McKeown et al., ”Piton: A Manycore Processor for Multitenant Clouds,” in IEEE Micro, vol. 37, no. 2, pp. 70-80, Mar.-Apr. 2017

[6] T. Fischer, M. Rogenmoser, M. Cavalcante, F. K. Gürkaynak and L. Benini, ”FlooNoC: A Multi-Tb/s Wide NoC for Heterogeneous AXI4 Traffic,” in IEEE Design & Test, vol. 40, no. 6, pp. 7-17, Dec. 2023

[7] G. Paulin et al., ”Occamy: A 432-Core 28.1 DP-GFLOP/s/W 83% FPU Utilization Dual-Chiplet, Dual-HBM2E RISC-V-Based Accelerator for Stencil and Sparse Linear Algebra Computations with 8-to-64-bit Floating-Point Support in 12nm FinFET,” 2024 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits), Honolulu, HI, USA, 2024, pp. 1-2

[8] B. Mao et al., ”Routing or Computing? The Paradigm Shift Towards Intelligent Computer Network Packet Transmission Based on Deep Learning,” in IEEE Transactions on Computers, vol. 66, no. 11, pp. 1946-1960, 1 Nov. 2017

[9] Kaleem, M., Isnin, I.F.B. (2021). A Survey on Network on Chip Routing Algorithms Criteria. In: Saeed, F., Al-Hadhrami, T., Mohammed, F., Mohammed, E. (eds) Advances on Smart and Soft Computing. Advances in Intelligent Systems and Computing, vol 1188. Springer, Singapore

[10] A. Shafiee et al., ”ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars,” 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), Seoul, Korea (South), 2016, pp. 14-26

[11] G. Desoli et al., ”14.1 A 2.9TOPS/W deep convolutional neural network SoC in FD-SOI 28nm for intelligent embedded systems,” 2017 IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, CA, USA, 2017, pp. 238-239

[12] G. Desoli et al., ”16.7 A 40-310TOPS/W SRAM-Based All-Digital Up to 4b In-Memory Computing Multi-Tiled NN Accelerator in FD-SOI 18nm for Deep-Learning Edge Applications,” 2023 IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, CA, USA, 2023, pp. 260-26

[13] A. Ankit et al., “PUMA: A programmable ultra-efficient memristorbased accelerator for machine learning inference,” in Proc. 24th Int. Conf. Archit. Support Program. Lang. Oper. Syst., Apr. 2019, pp. 715–731.

[14] lowRISC, ”Ibex RISC-V Core,” GitHub. [Online]. Available: https://github.com/lowRISC/ibex. [Accessed: 01-Jun-2024].