An Ultra-Low Cost and Multicast-Enabled Asynchronous NoC for Neuromorphic Edge Computing

Zhe Su, Simone Ramini, Demetra Coffen Marcolin, Alessandro Veronesi, Milos Krstic, Giacomo Indiveri, *Senior Member, IEEE*, Davide Bertozzi, *Member, IEEE*, and Steven M. Nowick

***Abstract*— Biological brains are increasingly taken as a guide toward more efficient forms of computing. The latest frontier considers the use of spiking neural-network-based neuromorphic processors for near-sensor data processing, in order to fit the tight power and resource budgets of edge computing devices. However, a prevailing focus on brain-inspired computing and storage primitives in the design of neuromorphic systems is currently bringing a fundamental bottleneck to the forefront: chip-scale communications. While communication architectures (typically, a network-on-chip) are generally inspired by, or even borrowed from, general purpose computing, neuromorphic communications exhibit unique characteristics: they consist of the event-driven routing of small amounts of information to a large number of destinations within tight area and power budgets. This article aims at an inflection point in network-on-chip design for brain-inspired communications, revolving around the combination of cost-effective and robust asynchronous design, architecture specialization for short messaging and lightweight hardware support for tree-based multicast. When validated with functional spiking neural network traffic, the proposed NoC delivers energy savings ranging from 42% to 71% over a state-of-the-art NoC used in a real multi-core neuromorphic processor for edge computing applications.**

***Index Terms*— NoC, neuromorphic, asynchronous.**

Manuscript received 25 March 2024; revised 30 May 2024 and 16 July 2024; accepted 16 July 2024. Date of publication 25 July 2024; date of current version 13 September 2024. This work was supported in part by the European Union under Grant GA 101147319 (EBRAINS 2.0), Grant GA 101160314 (TWIN-RELECT), Grant GA 101160182 (TAICHIP), and Grant GA 101160293 (AIDA4Edge); in part by the European Research Council (ERC) under the European Union's Horizon 2020 Research and Innovation Program under Grant 724295 (NeuroAgents); in part by the Electronic Component Systems for European Leadership (ECSEL) Joint Undertaking under Agreement 876925 (ANDANTE); and in part by NSF under Grant CCF-1527796. This article was recommended by Guest Editor A. Das. *(Corresponding author: Zhe Su.)*

Zhe Su and Giacomo Indiveri are with the Institute of Neuroinformatics, University of Zurich, 8006 Zürich, Switzerland, and also with ETH Zürich, 8057 Zürich, Switzerland (e-mail: zhesu@ini.ethz.ch).

Simone Ramini is with the University of Ferrara, 44121 Ferrara, Italy.

Demetra Coffen Marcolin and Davide Bertozzi are with The University of Manchester, M13 9PL Manchester, U.K.

Alessandro Veronesi is with the IHP–Leibniz Institute for High Performance Microelectronics, 15236 Frankfurt, Germany.

Milos Krstic is with the IHP–Leibniz Institute for High Performance Microelectronics, 15236 Frankfurt, Germany, and also with the University of Potsdam, 14469 Potsdam, Germany.

Steven M. Nowick is with Columbia University, New York, NY 10027 USA.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JETCAS.2024.3433427.

Digital Object Identifier 10.1109/JETCAS.2024.3433427

I. Introduction

THERE is a growing interest in neurobiologically inspired computing methods for their potential efficiency gains. Spiking neural networks (SNNs) are particularly notable for their ability to emulate the behavior of biological neurons, where neuron activity is triggered by stimuli, resulting in the firing of spikes when a threshold is reached [1]. This event-based computational model is highly energy-efficient due to the infrequent occurrence (i.e., sparsity) of spikes in both time and space.

Neuromorphic engineering aims at efficiently implementing SNNs in hardware, leveraging properties such as processing and memory co-location, event-driven processing, and low-precision computation with binary spike encoding. The steady growth in this field, following the pioneering work of C.Mead in the late 1980s [2], has led to the development of large-scale neuromorphic computing systems [3], [4], [5], [6], which are currently pursuing further breakthroughs in scalability and system integration [7], [8], [9].

More recently, there has been a growing momentum in developing neuromorphic processors to endow decentralized edge computing devices with intelligence [10], [11], [12]. This surge is justified by the excessive power consumption associated with data communication in centralized cloud-based AI methods, propelled by the considerable potential of neuromorphic computing in applications like smart sensors, bio-signal processing, and neuromorphic robots. Nonetheless, hardware platforms for biologically-inspired edge computing are not as consolidated as their large-scale counterparts, but are still in the early stage of design space exploration. This is primarily due to the stringent power and resource constraints that must be met when designing low-to-medium scale systems with hundreds or thousands of neurons. In particular, while it is relatively easy to place tens of thousands of neuron-equivalent circuits on a chip, the millions of interconnections between these neurons are impractical to fabricate in silicon. As a workaround, neuromorphic platforms use a virtual wiring scheme called Address Event Representation (AER), whereby computational units (i.e., neurons) are given an address, encoded as a digital word, which is transmitted to destination synapse circuits as soon as an event occurs (i.e., when a neuron spikes).

Not surprisingly, a major source of overhead in neuromorphic edge computing platforms is represented by the on-chip communication infrastructure used for AER routing, which is always digital regardless the neuron and synapse circuit implementations (analog, mixed-signal or digital) [13]. Because of their highly distributed and concurrent computations, cutting-edge neuromorphic platforms are interconnected via conventional network-on-chip (NoC) architectures, which have been often force-fitted into the new environment with poor specialization for its specific messaging requirements. As a result, the energy cost of the digital AER routing infrastructure turns out to be a major hurdle in neuromorphic platforms, exacerbated either by the limited computational requirements for digital neuron simulation or by the low-power sub-threshold analog circuits used for neuron emulation [10].

There are two fundamental limitations in current neuromorphic NoCs. On the one hand, they often draw inspiration from their counterparts in traditional computing domains, where the focus is on transferring large amounts of data with high bandwidth across the memory hierarchy. In contrast, a brain-inspired communication fabric should be optimized for sending a large number of very small payloads to many destinations following statically-configured multicast paths [14]. On the other hand, SNN-based neuromorphic systems use biological inspiration and obtain energy efficiency by exploiting asynchronous event-driven computation and communication. As a result, asynchronous NoCs are more biologically-plausible than their synchronous counterparts, although more difficult to design efficiently. The asynchronous NoCs used in current neuromorphic systems can hardly unfold their energy-saving potential due to either a design style oriented to extreme robustness and ease-of-design at the cost of implementation efficiency [15], or to the use of behavioural synthesis methodologies that end up in sub-optimal circuit realizations [16], [17].

Paper Contribution This paper targets a major innovation in interconnection networks for on-chip communication in neuromorphic edge computing platforms, which relies on two fundamental pillars:

- Motivated by the short payloads of spiking messages (essentially, only neuron addressing information, in the order of only a few tens of bits [18]) and by the wide physical links increasingly falling within reach of scaled process nodes, this paper advocates *compact single-flit (i.e., one-word) packets as the most efficient network encoding format for spike messages*. They mix header information and payload into a single word for low-latency transmission, and hold promise of radical switch architecture streamlining, especially when it comes to hardware support for multicast.
- The conservative asynchronous design style used in most state-of-the-art neuromorphic NoCs, emphasizing robustness over lightweight implementation, can result in overdesign for many practical systems [15]. Following this intuition, the proposed design instead targets cost-effective realizations, delivering major overall cost benefits, at the price of handling moderate timing constraints through an automated synthesis toolflow.

By coherently and systematically developing the implications of these principles, the paper comes up with an ultra-low cost NoC architecture for the efficient delivery of spike messages, with built-in and lightweight support for the parallel tree-based multicast of single-flit packets. The switch is made available through efficient and robust asynchronous design, thus enabling spikes to be generated, routed, and consumed in an event-driven manner with maximal activity gating during idle periods, and with no overhead for global clock distribution, tuning and management.

The proposed NoC can be used either standalone, in those platforms where short messaging is prevalent, or as a dedicated network plane oriented to short messages in the presence of high traffic diversity.

It should be observed that developing an asynchronous NoC architecture around single-flit packets is far from a simple streamlining exercise of conventional architectures, but comes with *daunting challenges that have no straightforward equivalence with synchronous design:*

Challenge A: robust concurrency of internal micro-operations. The use of a performance-driven asynchronous design style pushing for rapid processing of successive packets, combined with the short transmission window of compact single-flit packets, leads to timing robustness issues that must be safely handled. In particular, with multi-flit packets the entire packet transmission time is available to safely reset switch components (e.g., the routing logic) before the new packet arrives. In contrast, with compact single-flit packets the internal reset operations of the switch from a previous packet might end up partially overlapping with the admission of a new one, potentially leading to data overrun conditions. Our solution (Section VI-B) consists of a *conservative prevention of new re-activations of critical switch subblocks too early (i.e., before they have safely reset), while only marginally affecting performance.*

Challenge B: lightweight completion detection of multicast transactions. An asynchronous multicast switch must be able to correctly detect completion of operations across a wide range of multicast transmission patterns, in the presence of arbitrary downstream traffic delays, without the use of a fixed-rate strobing clock. *This challenge is tackled by this paper with trivial circuit overhead by taking advantage of the inherent properties of high-speed asynchronous signalling protocols (Section VI-D).*

Challenge C: robustness issues with asynchronous arbiters. In asynchronous NoCs, implementing fair, high-performance and robust arbitration mechanisms is challenging, since inputs may compete and request at arbitrary points in continuous time, unaligned to clock cycles. Two well-known challenges deserve special attention in light of the high-frequency activations generated by single-flit packets:

Challenge C1: asynchronous arbiters come with variable resolution times, which leads to operational robustness issues.

Challenge C2: under high contention, during transitory operations, **real asynchronous arbiters may exhibit two grants concurrently high for a limited time** (a deasserting and a newly-asserting one), potentially leading to switch malfunctions.

For the first time, this paper addresses the above arbiter issues *at the level of the switch architecture rather than of the arbiter circuit,* where such problems can be typically alleviated but not entirely removed. Our approach to variable resolution times consists of *conditioning the activation of new input requests from the same input port to the successful deassertion of the associated grant* (Section VI-C). In a similar vein, we avoid grant overlapping by *preliminarily masking the deasserting grant before removing the corresponding input request, which causes a competing request to safely take over the arbiter (Sections VI-C and VI-E).*

The efficiency and practical relevance of the proposed NoC are validated by comparing it against a state-of-the-art asynchronous one used for spike routing in a real multi-core neuromorphic processor for edge computing applications. When put at work with a Keyword Spotting task, the proposed NoC outperforms the state-of-the-art one with energy savings ranging from 42% to 71%, respectively.

II. Related Work

A. The Communication Bottleneck in Neuromorphic Computing

A comprehensive overview comparing most of the approaches that have been proposed for AER routing in literature so far is provided in [13]. As a matter of fact, the efficiency and scalability of neuromorphic computing platforms, regardless their scale or implementation style, are strongly limited by their on-chip interconnection networks. Whenever ultra-low power subthreshold analog design is used for the neurosynaptic framework, the energy per synaptic operation can be improved by at least one order of magnitude by simply porting the digital communication fabric to a more scaled process node [10]. In a similar vein, the substantial data transfers required by solver-based digital implementations pose challenges in aligning them with a low-power event-driven neuromorphic approach [19]. Last but not least, some of the main bottlenecks in the construction of large-scale reconfigurable neuromorphic computing platforms are the bandwidth, latency and memory requirements for routing address-events among neurons [3], [4]. Several large-scale neuromorphic computing approaches have therefore restricted the space of possible network connectivity schemes to optimize bandwidth usage while minimizing power and latency [3], [6]. In all cases, the capability of many neuromorphic platforms to target acceleration factors of neural process simulations/emulations compared to biological time may quickly lead to saturating the network bandwidth.

B. Hardware Support for Parallel Tree-Based Multicast

Unlike classical digital logic circuits, SNNs are typically characterized by very large fan-in and fan-out numbers. For example, in cortical networks neurons project on average to about 10000 destinations. Hardware support for parallel tree-based multicast typically comes with significant cost [20], [21], [22], [23]. A major source of overhead consists of deadlock avoidance techniques. It is well known that even when a deadlock-free multicast routing algorithm is used (e.g., XY), deadlock may still arise as a result of the interdependences among the various multicast branches (i.e., at the multicast-replication level). There are several known solutions to this problem, which all come with relevant overhead: buffer-hungry virtual cut-through (VCT) switching [24], [25], low-performance circuit-switching approaches [26], or costly deadlock recovery schemes [27].

In the synchronous domain, it has been observed that single-flit packets make the routing of each multicast branch independent, and their switching is, by construction, deadlock-free. This attribute has been exploited by many synchronous switches [28], [29], [30] to build deadlock-free and/or simplified multicast-enabled NoC architectures.

In the asynchronous domain, no specialized architecture has been developed so far to take advantage of single-flit packets. As a result, it comes as no surprise that many neuromorphic computing platforms still take a naive approach to multicast where multiple unicast copies of a multicast packet are serially injected and independently routed to their destinations [3], [5], [13].

A few neuromorphic NoCs provide hardware support for multicast, but stay away from the high complexity intrinsic to crossbar-based switch designs by trading flexibility and/or performance for lightweight implementation. The router in [31] takes advantage of tree topologies to simplify multicast addressing, and implements deadlock-free multicast as regional broadcast during the downward routing phase. This approach is topology-specific and limits mapping flexibility. The solution in [11] advocates a hierarchical topology, where multicast is enabled only in the tree subnetworks while packet replication is used in the top-level 2D mesh topology. The approach taken by [32] exhibits limited internal concurrency, both because the target output ports of a multicast packet cannot read it concurrently and independently and because all input ports are merged into a unique processing pipeline. Deadlock freedom is guaranteed at the large overhead of capacity detection mechanisms, emergency routing and even packet dropping.

In this paper, we leverage the properties of single-flit packets to deliver cost-effective and deadlock-free packet replication for efficient multicast in asynchronous NoCs.

C. Asynchronous Design Challenges

In the vast majority of neuromorphic NoCs (and in all multicast-enabled ones), the choice of communication protocols and data encoding schemes aims to simplify hardware design (e.g., using four-phase, or "return-to-zero," protocols) and to enforce extreme timing robustness (e.g., using "delay-insensitive" data encoding), at the cost of low throughput, high area occupancy, poor coding efficiency, and high energy-per-bit [15], [33]. This approach is at odds with the ultra-low cost computing targets of neuromorphic platforms.

Moreover, asynchronous circuit design faces challenges due to the absence of native support in standard industrial CAD tools. Consequently, all neuromorphic systems incorporating asynchronous logic depend on some extent on custom tool flows, which prolong the design process and necessitate assistance from a team proficient in asynchronous logic design.

A common approach consists of using the CHP hardware description language [16] and handshaking expansions [17] for high-level description. Then, asynchronous circuits toolkits evolved from [34] are used to aid in transforming the high-level logic into silicon. The issue when these methods are applied to NoCs is that there is typically a large abstraction gap between design entry and low-level circuit implementation, with a refinement process mainly driven by the functional specification and constrained by the semantics of the modelling tools. The frequent outcome is a sub-optimal circuit implementation of NoC switches that appears unnatural to expert NoC designers.

This paper targets a major improvement in the area and energy efficiency of multicast-enabled neuromorphic NoCs by means of an RTL-equivalent design entry, enabling microarchitecture optimization, and of an efficient data encoding scheme using standard synchronous-style single-rail data channels with binary data encoding.

D. General-Purpose Asynchronous NoCs

To address the inefficiencies outlined above, one potential solution would be to adopt the latest generation of asynchronous NoC switches designed for general-purpose computing, especially [24], [35]. These switches offer promising improvements in overall cost metrics, including coding efficiency, area, power, and performance, in addition to a growing ecosystem of design and synthesis tools [15].

However, to fully realize the benefits of these solutions in the neuromorphic computing domain, thorough customization is required. Existing general-purpose architectures prioritize the efficient data transfer of large volumes of data with high bandwidth, typically utilizing long (i.e., multi-flit) packet formats which split a packet into serialized head, payload and tail flits. This approach is less suited for brain-inspired communication fabrics [14], [36]. At the same time, such general-purpose switches geared towards long packets incur a large cost to deliver hardware support for deadlock-free multicast [24], [32], [37].

Last but not least, traditional one-size-fits-all approaches to NoC design hinder efficiency in spike routing when additional types of traffic, such as configuration, debug and monitoring, need to be conveyed. Instead, a more promising alternative involves multi-plane NoC architectures with specialized sets of routers and links, each optimized for the efficient delivery of a specific message type [38].

This paper documents substantial efficiency improvements when specializing NoCs for the messaging requirements of AER routing, rather than force-fitting general-purpose architectures into the new environment.

III. Background: Edge Neuromorphic Processors

The specific focus of this paper is on addressing the communication bottleneck in neuromorphic processors for decentralized near-sensor data processing (i.e., edge computing).

With the progression of technology, there's been a notable rise in the need to process sensory data directly where it's collected, at the edge, rather than relying on distant servers.

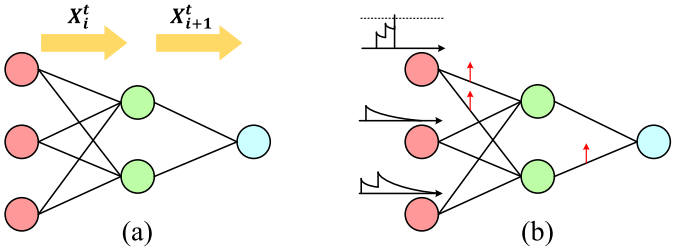


Fig. 1. The primary distinction between a) ANN and b) SNN.

This shift calls for applications that are not only close to the source of data but are also efficient in terms of energy use and quick in processing, all the while being robust and flexible enough to adapt to any changes. The key challenge lies in having processors that can handle continuous, real-time sensor data processing, extracting the necessary information while keeping energy use to a minimum.

A promising approach to meet these demands involves the use of event-based spiking neural networks (SNNs). Neuromorphic processors utilize event-based architectures that blend in-memory computing with the computational concepts inspired by the human brain, specifically for the deployment of SNNs. These processors are distinctive for their asynchronous, spike-based processing technique, promising exceptional efficiency and minimal power use, making them particularly adept at handling analog signals over noisy channels, enhancing noise resistance, and reducing both bandwidth and energy needs.

Fig. 1 illustrates the fundamental distinction in computing operations between Artificial Neural Networks (ANNs) and SNNs. In ANNs, as depicted in Fig. 1(a), computation and the propagation of neural activities across each layer occur synchronously. Conversely, SNNs operate on an event-triggered basis, as shown in Fig. 1(b). This event-driven approach to computation offers the benefit of energy efficiency, owing to the typically sparse occurrence of spikes (illustrated by red arrows) within the network, both temporally and spatially.

Considering that moving data around is a major energy drain in electronic systems, the best strategy for reducing power usage in event-based SNNs is to implement them within highly parallel, in(near)-memory computing frameworks. Owing to memory's inherent physical constraints, endlessly enlarging the core size of processors is not feasible. An effective strategy for developing extensive SNN systems is to utilize a multi-core architectural design. This approach streamlines both the planning and operational phases, making it more manageable to construct and deploy large-scale SNN systems within the tight resource budgets of edge devices. Each core within such a system is equipped with a neuro-synaptic array that includes synapse and neuron circuits.

Regardless of whether the circuits within these cores, designed to mimic the behavior of neurons and synapses, are realized through time-division multiplexed digital circuits [3], [4], [5] (offering the benefit of smaller area requirements at the expense of higher power usage) or through fine-grained fully parallel analog circuits that emulate real-time physical phenomena [6], [9], [11] (achieving lower energy consumption

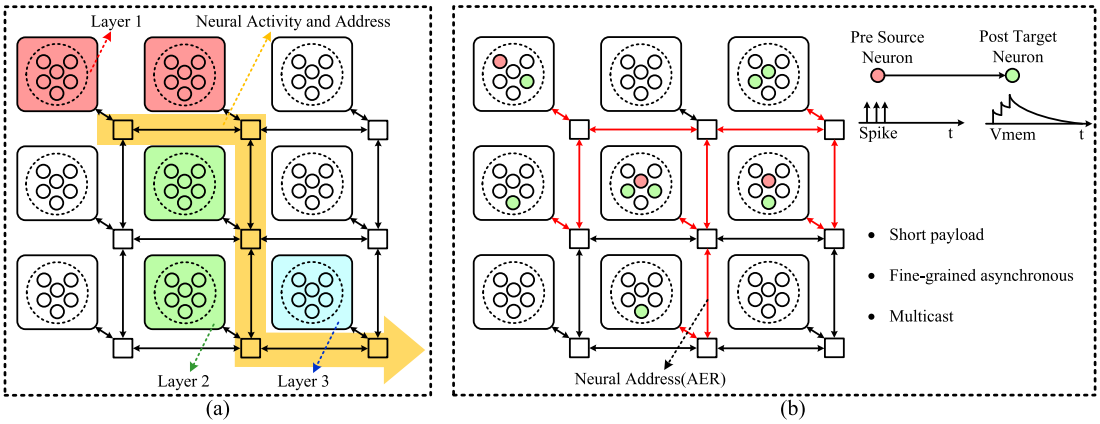


Fig. 2. The contrasting characteristics of NoCs within (a) ANN accelerators and (b) Neuromorphic processors.

but at the cost of larger area requirements), the communication strategies employed to facilitate interaction between different cores coincidentally utilize digital NoC techniques. These NoCs implement a communication protocol known as Address-Event Representation (AER). With this protocol, computational units, such as neurons, are allocated addresses encoded as digital words. These addresses are transmitted asynchronously using digital circuits as soon as neuron firing events occur. Consequently, information is encoded based on the timing of these address-events. In event-based neural networks, the analog data is represented by inter-spike intervals, which refer to the duration between successive address events from the same neuron. Neural computation is then achieved by forming connections between multiple neurons through a variety of connectivity schemes.

The architecture of the NoC used for AER communication in neuromorphic processors is significantly distinct from those NoC designs that are conventionally employed in multi-core processors, especially in comparison to those utilized in ANN accelerators. Illustrated in Fig. 2(a), the process of ANN inference (and training) demands that the NoC synchronously relays the neuron activities (or gradients) from the upstream (downstream) layer to the downstream (upstream) layer for computation. To accommodate the escalating data flow as the neural network enlarges, the NoC's bandwidth must be sufficiently large. Conversely, as depicted in Fig. 2(b), within multi-core neuromorphic processors, the data packets introduced into the NoC are finely asynchronous with dynamically varying injection rates. These packets often require multicasting or even broadcasting to numerous target cores, reflecting the extensive fan-out connectivity seen in the human brain. Additionally, these packets primarily include neuron address information, leading to compact data packet sizes in the order of a few tens of bits. Importantly, SNN utilize spikes to convey temporal data, an essential element in neural computation. This feature is particularly crucial for continuous analog neuron computing, emphasizing the NoC's latency requirements in neuromorphic systems.

Above all, the NoC is a critical factor in minimizing energy consumption in neuromorphic processors. For instance, in [11], the NoC's power consumption accounts for over 80% of the total energy usage [39].

IV. Low-Cost Design Goals of the Proposed NoC

The main goal of this paper is to come up with an *ultra-low cost* switch for AER routing in edge neuromorphic processors, with lightweight hardware support for packet replication, through which packets can be routed through the distinct outputs of a switch according to each output's own rate, in parallel, and in continuous time (i.e., not aligned to clock cycles). The ultra low-cost design goal is pursued through the following course of action:

- A less conservative asynchronous design style than the one used in most neuromorphic NoCs, aiming at high coding efficiency, low area and energy-per-bit, at the cost of exposing moderate one-sided relative timing constraints. The timing closure for the latter is delivered in a fully automated way by using scripting on top of standard industrial CAD tools [15], [40].
- Specializing the architecture for single-flit packets and exploiting the resulting architecture streamlining opportunities, that range from flattening the control hierarchy to the deadlock-free routing of multicast branches by construction.
- Removing the overhead that state-of-the-art switches in similar asynchronous design style incur for the low-latency transmission of body flits (e.g., speculation techniques [35]). In the proposed architecture there are no body flits.
- Tailoring the multicast protocol to single-flit packets, so that packets can be replicated in parallel to all required output ports of a switch by simply leveraging the native parallel replication support that is readily available through the crossbar of each NoC switch. Instead, state-of-the-art switches incur the significant control overhead and management of multiple independent read pointers from input buffers, thus leading to large buffering requirements and serious wiring congestion issues during physical design [24].

The challenge will be not to trade robust operation for architectural complexity. In fact, state-of-the-art asynchronous switches can in principle carry single-flit packets as a special case of multi-flit ones, however they overlook the decreased operation robustness incurred when using this packet format.

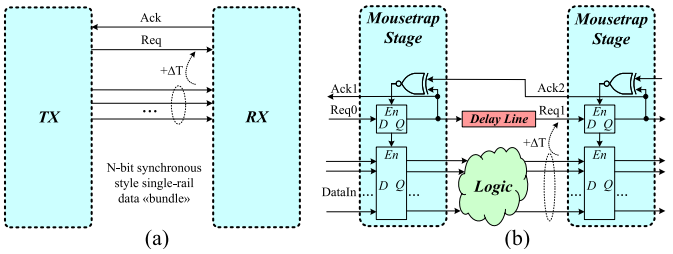


Fig. 3. Basic design choices. (a) *Bundled-data* asynchronous design. (b) *Mousetrap* pipeline.

In particular, in high-performance switch designs the concurrency of internal micro-operations may give rise to an inherent *window of vulnerability,* between the end of a packet and the arrival of the next packet at the same input port, which may lead to *data overrun conditions.* This window gets more frequently activated by single-flit packets and is thus a fundamental concern of this work.

V. Basic Design Choices

The proposed switch architecture has been designed leveraging a *bundled-data* asynchronous design style (Fig. 3(a)). Whenever data are sent, a Req signal "bundled" with the data serves as a local strobe for the receiver, which then completes the handshaking by toggling Ack [41]. In our design, these control signals toggle only once per data transfer *(i.e., a 2-phase non-return-to-zero communication protocol)*, thus enabling high throughput but potentially making switch design more complex [41].

For correct operation, a *single one-sided relative timing constraint (RTC)* must be satisfied by the synthesis tool, that the Req delay is always longer than worst case data transmission (Fig. 3(a)). This *bundling constraint* is typically met by inserting a small matched delay on the control line, when needed.

Another foundational choice for the switch concerns the asynchronous pipeline on the datapath. The switch revolves around *Mousetrap* [42], a high-performance asynchronous pipeline using 2-phase protocol and bundled-data encoding. Its stages use very simple control circuits (a single exclusive-NOR gate) and data registers (a single bank of level-sensitive D-latches), with low area and delay overheads (Fig. 3(b)). Unlike in synchronous design, single-latch registers are normally transparent, closing only transiently to protect data immediately after it enters a stage. Once data enters the next cascaded stage, the current register is reopened.

VI. New Multicast-Enabled Switch

A. High-Level Structure and Operation

The proposed design is modular and connects an arbitrary number of input port modules (IPMs) with output port modules (OPMs). The high-level structure of a 4×4 switch is illustrated in Fig. 4, including an expanded view of an IPM (left) and of an OPM (right).

Two modified versions of the Mousetrap pipeline are placed, as buffers, at the input of each IPM and at the output of each OPM, respectively. A *Packet Route Selector* reads in header information from the buffered input packet and selects the productive output port(s) toward the destination(s) by driving signals *RSi* accordingly. The activated *Request Generator* blocks generate a dedicated request signal *Reqi* for the associated OPMs with the correct polarity.

OPMs arbitrate between multiple incoming requests trying to access the associated output channel. A key OPM component is the *four-way arbiter*, mediating between competing input requests in continuous time, without any reference clock. Grant overlapping (GO), i.e., the simultaneous assertion of two grants for some time, affects all asynchronous arbiters to an extent which depends on circuit topology and number of clients. Thus, a *Masking Stage* is used to constantly enforce the one-hot encoding of arbiter decisions.

The filtered grant signals *MGs* concurrently select the correct data input of the *Crossbar Multiplexer* and enable only the winning request to propagate through the *Request Selection* block. As a result, a new valid packet is presented to the OPM buffer. The latter is an *extended Mousetrap stage* that differs from the baseline version in Fig. 3(b) mainly for the capability to handle the handshaking protocol with all possible transmitting IPMs through dedicated couples of *Req,Ack* signals.

In multicast transactions, each addressed OPM eventually *grants* the requesting IPM, and consumes the associated data concurrently and independently of the other OPMs, acknowledging successful data capture into their output Mousetrap registers. An *Ack Generator* block in each source IPM is in charge of signaling completion of packet processing through signal *AckX*, regardless of the number of activated output channels and their delays. Finally, the *Packet Route Selector* is safely reset before allowing a new packet to enter through the *PRSReady* signal.

The proposed architecture can easily implement alternative state-of-the-art encoding formats for multicast destinations. In this paper, the popular bit string addressing for multicast over generic 2D mesh topologies is used, wherein the address field has a single bit for each node in the network, which is set to 1 if the node is a destination, 0 otherwise. XY routing is then used by the routing logic to forward the multicast flit on the appropriate output ports.

B. Input Buffering and Routing Stage

The IPM admission stage stores incoming packets and routes them to the intended OPMs (Fig. 5(a)). A fundamental innovation is the enforced robust operation during the transition between any two consecutive packets, since with single-flit packets the switch control path cannot rely on a relatively long multi-flit transmission time to reset for a new packet. The vulnerability of operating with single-flit packets is largely overlooked in previous work. In this paper, the robustness guarantee is achieved through a modified input Mousetrap stage that incurs the trivial overhead of only one additional AND2 gate.

The basic operation of the IPM is as follows. A new incoming packet, signalled by *ReqIn* soon after *DataIn* arrives, makes the Mousetrap register temporarily opaque, thus safely storing data. Concurrently, the *Packet Route*

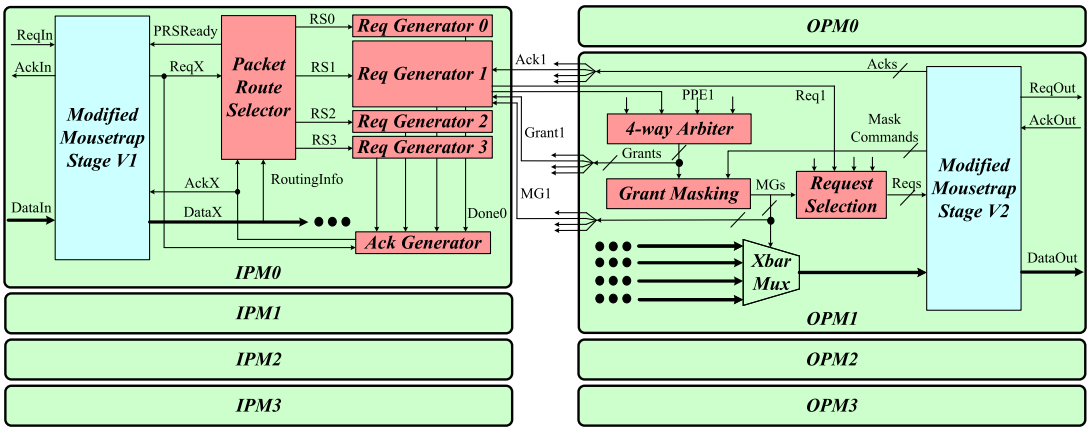


Fig. 4. Top level view. The design is modular with an arbitrary number of input port modules (IPMs) and output port modules (OPMs). A 4×4 switch is illustrated here.

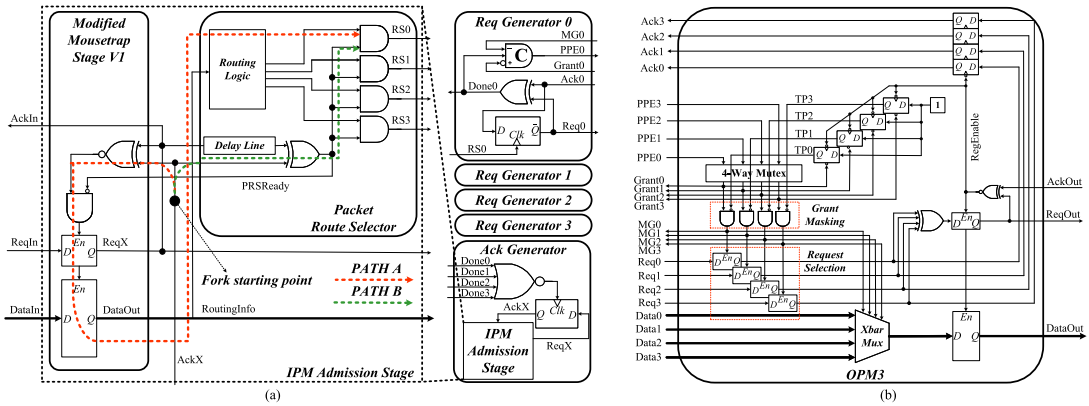


Fig. 5. New components: (a) The input buffering, the routing stage, the req generator and the ack generator in IPM (b) The grant masking and the req selection in OPM.

Selector processes the packet header and activates the *Request Generators* of the productive output ports. The routing logic is a synchronous-style, i.e. hazardous, computation block. For this reason, during the routing computation delay the propagation of glitches downstream is prevented by a barrier of AND2 gates. The enabling input of the gates is generated by a matched delay line cascaded with a XOR2 gate converting the 2-phase signals *ReqX* and *AckX* to a level signal.

Safe reopening of the input Mousetrap needs to deal with a potential race condition between i) *AckX* (indicating transaction completion at target OPMs) driving the Mousetrap control circuit to reopen the input register (path A in Fig. 5(a)), and ii) *AckX* masking routing selection signals *RSi* for a new hazard-free routing computation phase (path B). The newly added AND2 control gate in the Mousetrap allows the safe reopening of the input register only when both transients have completed, hence no fast data bit can propagate through the routing logic and tunnel the AND2 barrier before it gets shut off.

C. Request Generator

The Request Generator (see Fig. 5(a)) generates the correct polarity of the request signal *Reqi* toward the associated output port, and sends switch allocation requests to OPM asynchronous arbiters. Thanks to its focus on single-flit packets, the proposed architecture fulfils these requirements nonspeculatively (unlike the general-purpose architecture in [35]) and with marginal area overhead (just 3 logic gates), while at the same time protecting the control path from the race conditions originating from its frequent flit-, instead of packet-, level activation.

From an operational viewpoint, when the route selection signal *RSi* from the *Packet Route Selector* is asserted, the request signal *ReqI* toggles, indicating data availability to an OPM. When an *AckI* transition will eventually arrive from the associated OPM, transaction completion on that output port will be notified to the *Ack Generator* via the level-signal *Done*.

Finally, a twofold course of action is taken to deliver robust interaction with the OPM and guarantee predictable arbiter operation. First, a robust 4-phase handshaking is implemented between the *Request Generator* and the OPM at each packet through the signals *PPE* and *Grant*. The latter is a monitoring signal provided by the new OPM (Fig. 5(b)), and carries the state of the arbiter output associated with the *Request Generator* at hand. The handshaking ensures safe arbiter reset (detected by a low value on the associated arbiter *Grant*) before asserting the next request (*PPE*) from the same IPM. For this purpose, the inverted *Grant* signal is connected to the asymmetric plus input of an asymmetric C-element.

Second, before deasserting a previously-granted arbiter request, the new design makes sure that the associated arbiter output has been masked in the OPM to protect against *GO*. This is indicated by the monitoring signal *MaskedGrant(MG)* from the OPM. This way, should the deasserting grant be slow and transiently overlap with the assertion of another grant from another pending arbiter request, only the latter would be allowed to take control of the OPM. For this purpose, another asymmetric C-element driven by *PPE* and *MG* (the asymmetric minus input) is used. From an implementation standpoint, the two C-elements can be merged into a unified 3-way one, as illustrated in Fig. 5(a). To our knowledge, this is the first time that the variable arbiter resolution times and the GO concern are both addressed at the switch architecture level, thus enabling the safe integration of any arbiter circuit into the switch.

D. Ack Generator

A key challenge in designing a multicast-enabled switch is to provide an appropriate *Ack Generator*, signaling completion of multicast packet processing in continuous time, regardless of the number of activated output channels and their delays, yet retaining simple hardware.

A radical circuit simplification is achieved in this paper by leveraging the original intuition that in a two-phase communication protocol *Ack* is always a delayed version of *Req*. Thus, in a specific IPM that has started a multicast transaction toward multiple OPMs, upon successful routing of all multicast branches (indicated by all *Done* signals coming from the *Request Generators* going to zero), signal *ReqX* is simply sampled into *AckX*. The resulting circuit is as lightweight as the cascade of a NOR gate with an edge-triggered D flip-flop (see Fig. 5(a)).

E. Output Port Module

OPMs arbitrate between multiple incoming requests trying to access the associated output channel (Fig. 5(b)). Basic OPM structure and operation, as well as the arbiter implementation, are pretty aligned with state-of-the-art [24], [35]. However, there are fundamental innovations. First, the proposed switch avoids packet completion signaling to the connected IPM, since with single-flit packets it would be redundant with flit completion signaling and become the source of subtle race conditions.

Second, it delivers guarantees in operation robustness by collaborating with the *Request Generator* to fix the *GO* issue with minor circuit overhead. For this purpose, the new design uses an intermediate level of control against *GO* between arbiter outputs (*Grant*) and input signals of the *Request Selector* (*MG*) through a barrier of AND2 gates (Fig. 5(b)). Such gates are by default transparent for the grant signals. However, when a packet is captured into the output Mousetrap, a flip flop is toggled by *RegEnable*, which masks the deasserting grant. Only when the corresponding *MG* masked grant goes to zero, the *Request Generator* will release the associated arbiter input, so that an eager grant assertion by a competing arbiter input request would be harmless, since the latter would be the only one entitled to take control of the OPM.

VII. Design Entry and Timing Optimization Flow

This section describes the hardware model of the switch and the timing optimization flow to gain tight control over absolute and relative timing paths through mainstream commercial clocked CAD tools.

Our technology library lacks asynchronous special cells such as Muller C-elements and mutexes. Consequently, we have utilized standard-cell equivalents for these components [43], [44]. Additionally, other asynchronous circuits within the design necessitate a detailed specification to ensure predictable logic behavior and hazard-free operation. To achieve this level of specification while maintaining technology independence, we employ the generic GTECH Synopsys library. While certain parts of the design, such as routing logic, can be specified using conventional hardware description language (HDL) behavioral models, we integrate these with the GTECH library to create a hybrid HDL-GTECH RTL-equivalent specification. This specification remains fully synthesizable and portable to any technology library. After reading in the entry-level netlist for technology-independent mapping, logic manipulations are prevented by setting appropriate compile directives: only gate sizing and buffer insertion options are enabled.

The design is then fed to the timing optimization flow (Fig. 6), which handles the convergence of all the relative timing constraints (RTCs) in parallel. The design is first constrained for max. performance, by enforcing the *set_max_delay* command to all the timing paths in the design. In the initial incremental synthesis runs, the difference between the request delay and the datapath delay to be matched is compared with a threshold called relative timing margin (RTM), set e.g. to 10%. If the constraint is violated (e.g., request slower than the data delay), or the margin is not enough, an incremental synthesis is performed. It is important to notice that an additional *set_max_delay* constraint is imposed on the request signal (margin *max_d* in Fig. 6) to avoid largely exceeding the bundling constraint, which would lead to a waste of performance.

When several bundling constraints are handled in parallel, the tool may struggle to converge on them all. In this case, convergence is facilitated by leaving more freedom to the tool, for instance by progressively increasing the margin *max_d* by small *extra_slack* increments or, at some point, or even by changing cost priorities. Whenever the bundling constraint is overdesigned, an attempt is made to make it tighter (right branch in Fig. 6) by largely relaxing the *max_d* constraint. Empirically, we found that this degree of freedom is exploited by the incremental synthesis to come up with less conservative relative timing margins.

This flow was sufficient for the effective convergence of our designs. In other cases, the synthesis process becomes more challenging, for instance when all the point-to-point links of an entire topology have to be synthesized, placed and routed concurrently. In order to handle the scale and optimization challenge of complex switches and network topologies,

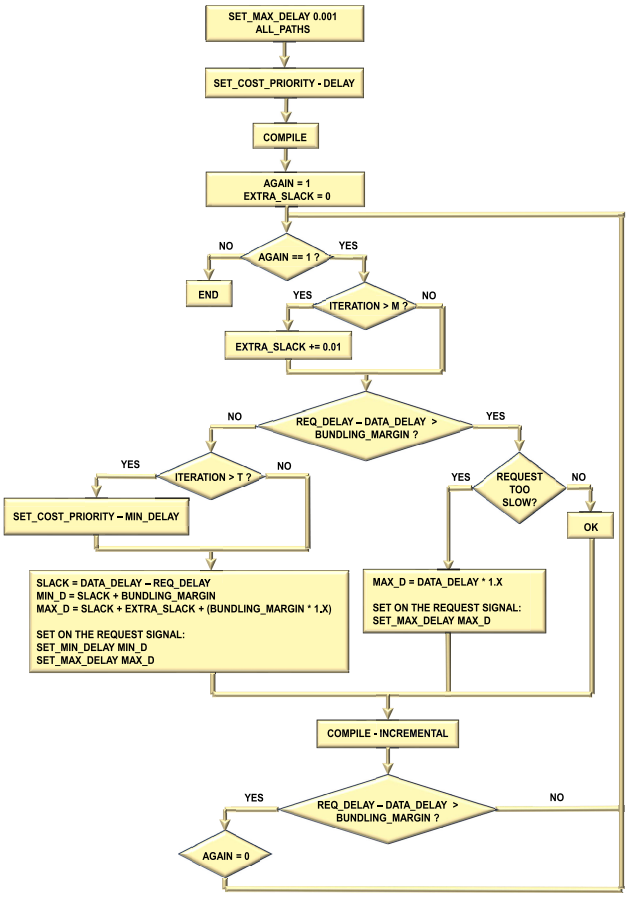


Fig. 6. Timing optimization flow. Values M, T and X are user-defined parameters.

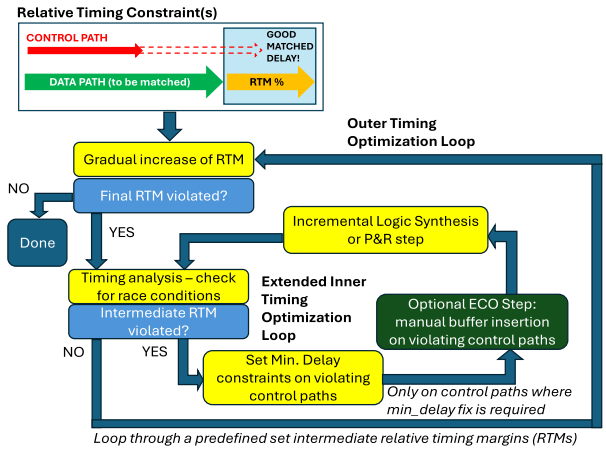


Fig. 7. Nested timing optimization loop.

the illustrated methodology is extended with an engineering change order (ECO) step, and becomes the inner loop of a nested timing optimization flow (Fig. 7).

The outer nested loop hits RTMs gradually: the RTM is not immediately set to its final target, but to more relaxed intermediate values (e.g., no margin, then 5%, 7%, and finally 10% of the datapath delay to be matched), which the synthesis and the P&R tool can more easily fulfill. Finally, the simultaneous convergence of numerous control signals on the intermediate or target RTM can be optionally expedited by selectively introducing small delay elements on control paths that are violating timing constraints through an ECO step.

VIII. Validating the Benefits of Specialization

In this section, the quality metrics of the proposed NoC architecture are assessed in comparison with state-of-the-art general-purpose asynchronous switches handling single-flit packets as a special case.

The following NoC switches are compared in terms of communication efficiency when carrying short payloads:

Baseline Unicast. This is a leading general-purpose asynchronous switch from [35] for unicast traffic. Short payloads are carried through a conventional 2-flit packet: one head flit and one body flit.

Optimized Baseline. This is again the Baseline Unicast switch with minor modifications to support single-flit packets (header and payload mixed in the same flit).

Proposed Unicast. The proposed specialized architecture for single-flit packets, restricted to unicast communication. For this, packet headers were modified to carry unicast addresses only, and algorithmic xy routing was implemented in the routing logic.

Baseline Multicast. A leading general-purpose asynchronous switch with hardware support for parallel multicast from [24].

Proposed Multicast. The proposed specialized architecture for single-flit packets with the full support for parallel multicast.

For *Baseline Multicast*, quality metrics were taken or extrapolated from the original paper, since they are referred to the same technology feature size considered in this paper. All other designs have been modelled in synthesizable RTL, then synthesized, placed and routed on a 40nm industrial technology library with commercial tools (Synopsys Design Compiler and IC Compiler). All bundling constraints have been fulfilled with at least a 10% timing margin. To assess post-layout performance, we evaluated the quality metrics under typical corner conditions by Cadence Innovus.

A. Unicast Switches

At first, all designs capable *only of unicast communications* are compared (Tab. Ia). For fair analysis, all architectures have been tuned for carrying the same compact payload, while keeping all other parameters the same (5×5 switches, 1 input and output Mousetrap, 4-bit unicast addresses for a 4×4 2-D mesh, xy routing). We have chosen a constant payload size of 28 bits, which is in the ballpark of the typical requirements for edge neuromorphic computing [11], [18]. As a result, the bitwidth slightly changes across configurations depending on the specific packet format requirements (see notes in Tab. Ia).

Proposed Unicast consistently outperforms *Baseline Unicast*, especially in energy efficiency (67% less energy-per-bit) and performance (52% lower packet latency and 24% higher throughput). This is fundamentally due to the fact that with short payloads the overhead of a 2-flit packet format becomes sensitive. An optimized alternative consists of fitting the

TABLE I
COMPARISON WITH STATE-OF-THE-ART GENERAL-PURPOSE ASYNCHRONOUS SWITCHES

5x5 Unicast Switch Designs for same 28-bit payload	Area (um2)	%over Baseline	Packet Latency (psec)	%	Packet Rate (MPkt/s)	%	Unicast Energy (pJ/bit)	%
30-bit Baseline Unicast * (2 flits: 1 head only + 1 body flit)	4024		2393		450		0.24	
32-bit Optimized Baseline ** (1 single-flit packet)	4194	+4.2%	1113	-53.5%	621	+38%	0.13	-45.8%
32-bit Proposed Unicast ** (1 single-flit packet)	3835	-4.7%	1149	-52%	560	+24.4%	0.08	-66.8%

** The body flit determines the bitwidth: 28-bit payload + 2-bit flit type. ** Bitwidth obtained by summing 28-bit payload + 4-bit unicast address (no flit type overhead for single-flit packets).*

(a) Post-layout results of unicast switches under test.

128-bit 5x5 Multicast Switch Designs	Area (um2)	%over Baseline	Header Latency (psec)	%	Packet Rate (MPkt/s)	%	Unicast Energy (pJ/bit)	%	Energy increment per additional multicast branch (pJ/bit/branch)
Baseline Multicast (multi-flit & single-flit packets)	54083		1629*		404		0.23**		[0.03-0.04]
Proposed Multicast (single-flit packets)	33650	-37.8%	1934	+18.7%	534	+32.2	0.17	-26%	[0.03-0.04]

** Performs input buffering in parallel with routing computation. ** Measured from 5-flit packets, where energy overhead for routing and switch allocation is amortized over a wider set of payload bits. Optimistic for single-flit packets.*

(b) Post-synthesis comparison with a multicast asynchronous switch.

payload into the head flit (*Optimized Baseline*). With respect to this solution, *Proposed Unicast* still saves 8.6% area and 38.5% energy-per-bit due to architecture specialization. From the performance viewpoint, an expected small penalty is measured (+3.2% latency, -9.8% throughput), which is the price to pay for the delivered safe reset guarantees of control path components between any two consecutive packets. We experimentally verified via post-layout simulation that *Optimized Baseline* relies on worst-case timing margins of 205 and 118 psec for the safe reset of its routing logic and arbiters, respectively. The work in [45] shows that for a similar 40nm process, path delay variations from hundreds of psec to several nsec can be expected, depending on supply voltage, logic depth, slew and load capacitance. This justifies the small performance overhead incurred by *Proposed Unicast* to deliver robust operation guarantees.

B. Multicast Switches

Proposed Multicast is compared with *Baseline Multicast* in Tab. Ib for a matched architecture configuration from the original paper in [24]: 128 bits, 5 I/O ports and 5-slot input buffers (we instantiated the same circular FIFOs for fair analysis). Even in this buffer-dominated configuration, *Proposed Multicast* exhibits significant area and energy-per-bit savings (37.8% and 26%, respectively). *Proposed Multicast* preserves its superior energy efficiency across N-way multicast transactions too (see last column in Tab. Ib). Latency is only apparently higher because *Baseline Multicast* performs input buffering concurrently with routing computation, a latency optimization technique that could be equally implemented in *Proposed Multicast* as well, which would likely lead to latency equalization.

Proposed Multicast exhibits a packet rate improvement over *Baseline Multicast* by 32.2%. There are two fundamental reasons. On the one hand, differently from latency, *Baseline Multicast*'s cycle time does not benefit from route computation in parallel with input buffering. On the other hand, the more complex architecture of *Baseline Multicast* determines a longer time both to receive the acknowledgement from the output ports and to reopen the input circular FIFO.

IX. VALIDATING THE SCALABILITY OF SINGLE-FLIT SWITCH

To further validate the scalability of the NoC using single-flit packets, we gradually increased the bit-width of the payload to be delivered, resulting in five groups: 28 bits, 84 bits, 120 bits, 252 bits, and 504 bits. These groups cover use cases ranging from extreme-edge neuromorphic computing devices to ultra-large neuromorphic clusters. Each group included six design points. In addition to the *Proposed Unicast* and *Optimized Baseline* design points analyzed in Section VIII, we configured the *Baseline Unicast* switch to support multi-flit packets ranging from one body flit to four body flits, supporting the same payload through reduced bitwidth/bandwidth and longer packets (Fig. 8). The experimental setup was the same as described in Section VIII. Random and uniform unicast data packets were injected, and the energy per bit was measured under post-layout simulation.

As shown in Fig. 8, during the stage of short payloads, the energy consumption clearly increases with the number of flits. This energy difference is attributed to the 4-phase control signal in both the input and output *Mousetrap* buffers. Energy consumption is more sensitive to the number of toggles of this control signal rather than to the slight increase in buffer width in minimum-buffered realizations.

As the payload size increases, the advantage of the single-flit switch over the multi-flit variants diminishes but remains significant. This is because the complexity of the "virtual

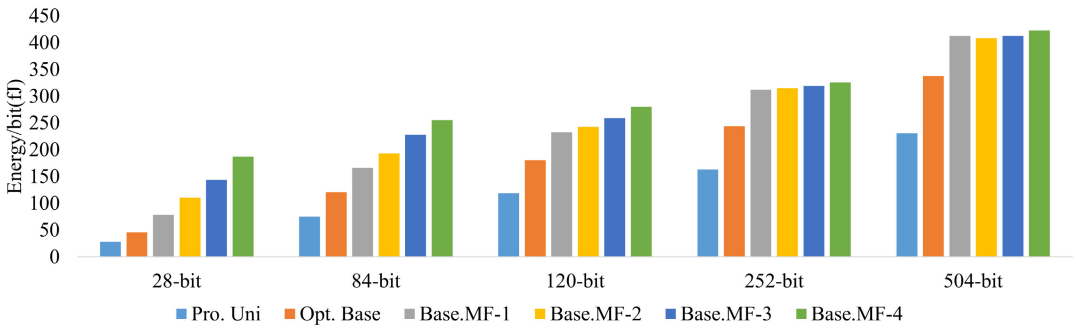


Fig. 8. Post-layout simulations measured the energy per bit for different design points across various payload sizes. The design points labeled Base.MF-*n* refer to the *Baseline Unicast* multi-flit switch with *n* body flits.

clock tree" in the control signal gradually increases, making the energy corresponding to the buffer width more apparent. A turning point occurs at around 504-bit payloads, where the design with two body flits achieves better energy efficiency than the one with one body flit.

Thanks to a carefully specialized architectural design, the *Proposed Unicast* always achieves better energy efficiency not only of all multi-flit variants, but also of the *Optimized Baseline*, and remains the most energy-efficient option across all benchmarks. The *Proposed Unicast* achieves from 2.7x to 1.5x higher packet rate than the *Baseline Unicast* with four body flits as we increase the payload size, even though the saturation point of the single-flit switch was never reached in all simulation groups. These results point to single-flit packets as the most energy efficient payload encoding format even when channel widths exceed 500 bits.

X. Validating With Neuromorphic Computing Traffic

To validate the potential of the proposed NoC in the target edge neuromorphic domain, this section encompasses three experiments that utilize SNNs to assess and compare the energy efficiency and latency of the proposed design with the state-of-the-art asynchronous NoC within the mixed-signal neuromorphic processor referred to as *DYNAP-SE* as detailed in reference [11]. The first experiment involves the SNN running in continuous time through direct emulation on *DYNAP-SE*. In contrast, the last two experiments utilize a discrete time approach through simulation, showcasing the inclusion of various types of neuromorphic hardware in our investigations.

***DYNAP-SE* R2 switch.** This switch enables the multicast of single-flit packets among multiple neurocores through a bit string-based tree hierarchy.

***DYNAP-SE* R3 switch.** This switch is tailored for unicast single-flit packets among multiple neurocores, using XY routing within a 2D-mesh network.

Optimized *DYNAP-SE* R3 switch. An optimized version of the R3 switch that we extended for multicast transactions using bit string addressing within a 2D-mesh network.

Proposed Unicast and **Proposed Multicast**. Our architecture with matched configurations to the *DYNAP-SE* NoCs.

The proposed design was constructed following the same design methodology as previously outlined but using GlobalFoundries 22FDX FD-SOI technology. However, the *DYNAP-SE* switch was realized as a hard macro, employing the original 180nm manufacturing process. This implementation involved a full custom design process within Cadence Virtuoso. In the case of both switches, the NoC was created within Cadence Innovus. To assess post-layout performance, we evaluated the quality metrics of the trace run under typical corner conditions. We used Cadence Innovus for the proposed design and Cadence AMS Simulator for the *DYNAP-SE* switch. All experimental results have already been normalized in 22nm technology. The energy report below includes dynamic and static power consumption. Three different SNNs were used for evaluation, as hereafter explained.

A. sWTA Experiment

In the first experiment, we incorporated soft Winner-Take-All (sWTA) networks in our SNN model. These sWTA networks exhibit dynamic attractor characteristics, crucial in various cognitive processes such as decision-making. We used real spike data recorded from *DYNAP-SE*, as detailed in [46], to replicate authentic population-coded neuronal activities change within the network from the removal of external input. While various NoC configurations lead to distinct spike transmission latencies, consequently influencing the generation of spike data trains (data packet injection into the NoC) due to the network operating in continuous time, our primary emphasis is on assessing the NoC's performance using actual spike traces. The impact of NoC latency on neural dynamics falls outside the scope of our research paper.

The sWTA network and NoC mapping are as depicted in Fig. 9(a) and Fig. 9(b), respectively. The sWTA model consists of 16 clusters. Each cluster contains 8 excitatory neurons, along with a global inhibitory cluster comprising 20 neurons. Input nodes evenly stimulate all neurons within a given cluster, following the standard sWTA connectivity described in [46]. Given the sparsity of neuron spikes, a single 5×5 data switch suffices in this context. We established five virtual cores of equal size, with four of them evenly distributing the 16 excitatory neuron clusters, and one core dedicated to the inhibitory neuron cluster. During synthesis, the virtual core lacked functional circuitry, generating only the previously mentioned spike trace during simulation. External stimuli were injected from the core that mapped to inhibitory

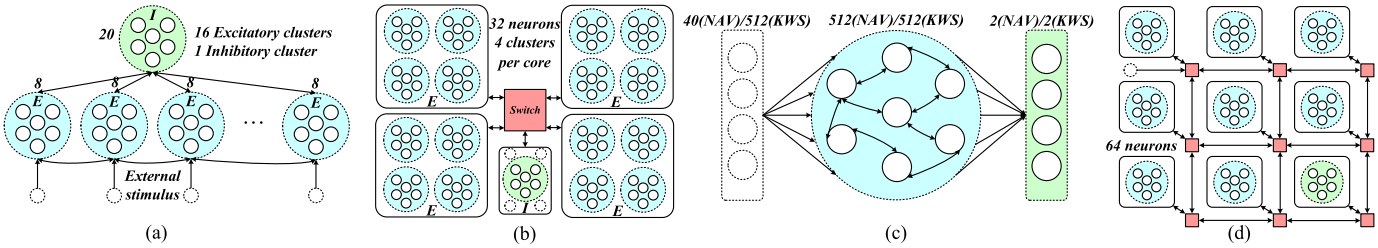


Fig. 9. Network mapping. (a) sWTA. (b) sWTA mapping. (c) RSNN. (d) RSNN mapping.

TABLE II
POST-LAYOUT COMPARISON OF SWTA EXPERIMENT

	Area	*Latency*	*Energy*
DYNAP-SE R2	$6832\mu m^2$	$1.42\,ns$	$0.81\mu J$
Pro.Unicast	$1451\mu m^2$	$0.83\,ns$	$0.69\mu J$
Pro.Multicast	$1783\mu m^2$	$0.77\,ns$	$0.31\mu J$

neurons. In this experiment, we instantiated the *DYNAP-SE R2 multicast switch*, *Proposed Unicast*, and *Proposed Multicast*. The multicast switch employs a bit string routing method with a 13-bit width, while the unicast switch utilizes the XY routing method with an 11-bit width. It's worth noting that all architectures were optimized for the same number of I/O ports (5) and featured a consistent buffering strategy with 6 *Mousetrap* stages in each output port.

1) Latency Analysis of sWTA Experiment: The observed latency represents the average time it takes for a spike data packet to travel from the moment it's introduced into the NoC to its arrival at the designated core. As illustrated in Table II, the *Proposed Unicast* reduces latency by 45.9% when compared to the *DYNAP-SE R2 switch*. This improvement can be attributed to its streamlined architecture, even though it involves the extra step of replicating the data packet for multiple destinations. However, it does result in a slight increase in latency by 7.2% when compared to the *Proposed Multicast*. This latency increase can be attributed to the specific conditions of this experiment, where the traffic pattern is multicast but the absence of data packet conflicts due to sparse neural activity leads to every packet experiencing zero-load one-hop latency. This reduces the impact of multicast traffic on latency.

2) Energy and Area Analysis of sWTA Experiment: This section provides an overview of the energy consumption for a 7-second spike trace and the area cost of the switch. Table II illustrates that the *Proposed Multicast* exhibits the highest level of energy efficiency, with a 61.7% reduction compared to the *DYNAP-SE R2 switch*. This improvement is attributed to the use of a quasi-delay insensitive (QDI) design style, in contrast to the bundled-data style employed in the proposed design, which results in substantial cost savings. As a result, the *Proposed Multicast* also achieves an area efficiency that is 3.8x greater compared to the *DYNAP-SE R2 switch*. Additionally, the *Proposed Multicast* reduces energy by 54.8% when compared to the *Proposed Unicast*, which incurs overhead due to replicating data packets for multiple destinations. A more comprehensive examination of the impact of multicast end nodes on energy is presented in the subsequent experiments.

B. NAV and KWS Experiment

In the remaining experiments, we employed a three-layer Recurrent Spiking Neural Network (RSNN), as depicted in Fig. 9(c). The first experiment involved a binary-decision navigation task(NAV) as referenced in [47]. In the second experiment, we illustrated 1-word Key Word Spotting (KWS) as mentioned in [47]. RSNN was trained using Backpropagation Through Time (BPTT), followed by the extraction of spike traces during the whole inference phase of simulation. The network's performance remains unaffected by NoC latency only if all spikes can reach the target core within the current time step.

In the first experiment, the RSNN had 40 external neurons in the input layer, while the second experiment had 512 neurons in its input layer. Both experiments employed 512 Adaptive Leaky Integrated and Fire (ALIF) neurons in the recurrent layer to capture temporal dependencies. In both cases, a 3×3 2-D mesh NoC was used, as shown in Fig. 9(d). Input layer spikes were injected from the top-left switch, and the hidden layer neurons were evenly distributed across eight neural cores. Output neurons were assigned to the bottom-right core for obtaining classification results.

We instantiated four types of switches in these two experiments: *DYNAP-SE R3 switch*, *Optimized DYNAP-SE R3 switch*, *Proposed Unicast*, and *Proposed Multicast*, each with different configurations. *Optimized DYNAP-SE R3 switch* and *Proposed Multicast* utilized bit string addressing for routing logic, resulting in a single-flit data packet with 18 bits (comprising 9 bits for routing address and 9 bits for neuron address). In contrast, *DYNAP-SE R3 switch* and *Proposed Unicast* employed algorithmic XY routing, generating a single-flit data packet with 13 bits (comprising 4 bits for routing address and 9 bits for neuron address) to manage unicast-based multicast data traffic.

(a) Varying network connections. To investigate the trade-offs between two alternative methods of supporting multicast transactions, namely packet replication (i.e., unicast-based multicast) and hardware support for parallel tree-based multicast, a cost function was introduced into network connectivity during training. This resulted in a weight matrix with small-world connectivity [48]. The approach involved retaining all-to-all connectivity within the local neural core while gradually pruning long-distance connections to achieve

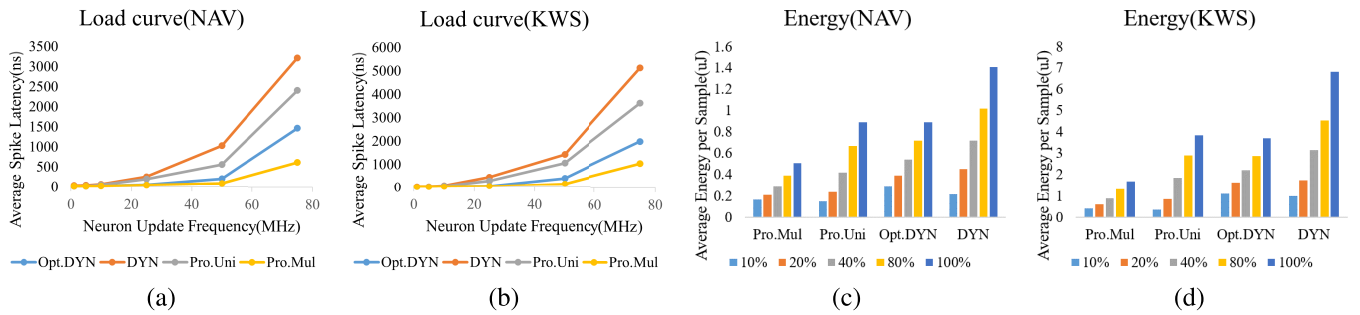


Fig. 10. Simulation results at the network level for NAV and KWS tasks. The average spike latency varies with an increase in neuron update frequency, and energy consumption changes with varying network sparsity.

a specified sparsity level. Five target sparsity levels were set at 10%, 20%, 40%, 80%, and 100%, representing the remaining long-range connectivity after pruning.

(b) Varying time steps. To evaluate the NoC's bandwidth, spike traces were extracted under different accelerated times, corresponding to varying neuron update frequencies (time steps). Six different update frequencies were configured for NAV and KWS tasks, affecting the spike packet injection rate into the NoC.

1) Latency Analysis of NAV and KWS Experiment: This section examines the average spike latency in both experiments. The target sparsity level was consistently set at an intermediate value of 40% for all cases. As depicted in Fig. 10a and Fig. 10b, the *Proposed Multicast* clearly outperforms in terms of saturating injection rate. It exhibits a 27.4% (42.7%) higher rate compared to the *Proposed Unicast* and a 31.4% (63.3%) higher rate compared to the *Optimized DYNAP-SE R3 switch* in the NAV(KWS) task. This performance advantage becomes even more evident due to the increased spike data in the KWS task. The *DYNAP-SE R3 unicast switch* fails to compete with multicast traffic due to the overhead of replicating packets to multiple target end nodes and its QDI design style. When we combine the within-core latency results from [5] with the findings from our experiments about NoC performance, after technology equalization we observe that *Proposed Multicast* could potentially achieve a 20% shorter time step for SNN simulation compared to *DYNAP-SE R3 unicast switch*. This reduction indicates a decrease in SNN latency.

2) Energy Analysis of NAV and KWS Experiment: This section delves into the topic of average energy usage per inference sample when dealing with varying connectivity constraints, as visualized in Fig. 10c and Fig. 10d. *Proposed Unicast* is the best candidate when the SNN strictly follow a small world network with sparse (10%) long-distance connectivity because a substantial energy reduction of 41.2% compared to *DYNAP-SE R3 unicast switch*. The energy-saving threshold at which *Proposed Multicast* becomes more energy-efficient than *Proposed Unicast* is observed to be at a 20% sparsity level, which translates to 20% remaining connectivity, in both experiments. To elaborate, the energy savings range from 31.3% to 68.7%, corresponding to long-distance connectivity sparsity levels ranging from 20% to 100%. The *Proposed Multicast* solution also demonstrates significantly improved energy efficiency when compared to the *Optimized DYNAP-SE R3 switch*, with energy savings ranging from 41.1% to 59.2%. To a first approximation, this means that the overall energy saving of the whole neuromorphic system will also be around 60% as an effect of the more efficient NoC. Interestingly, the *Optimized DYNAP-SE R3 switch* exhibits lower energy efficiency compared to *Proposed Unicast* at a 10% sparsity level, while achieving a 6.3% reduction in energy consumption at full connectivity. This underscores that hardware support for parallel tree-based multicast proves to be more energy-efficient than unicast-based multicast when there are multiple multicast end nodes. This trend is further validated by the fact that the increase in energy consumption for the unicast design surpasses that of the multicast design, primarily due to the added workload of replicating data packets. This contrast becomes even more pronounced in the context of KWS tasks, where the average neuron firing rate is higher. Overall, the choice between the two solutions under test depends on the actual traffic patterns of the network at hand, with hardware multicast paying off already with a relatively large number of multicast end nodes.

XI. Comparison With Synchronous Switch

We conducted a comparison between synchronous and asynchronous NoC architectures using a highly optimized single-cycle synchronous baseline switch, referred to as "xpipes". This switch was modified to implement the same design techniques for optimized single-flit multicast packet transmission as the proposed asynchronous switch. Specifically, the arbiter FSM was simplified to avoid differentiating between the transmission states of head, body, and tail flits. Additionally, each input port was enhanced with a simple "request generator" for each possible output port, functionally equivalent to the asynchronous request generators. This allows the output ports involved in a multicast transaction to read out a packet from the input buffer independently and concurrently at their own rate, synchronized to clock cycles.

Both the synchronous and asynchronous switches were instantiated with minimal buffering while still maintaining full throughput operation. The asynchronous switch includes one input and one output *Mousetrap* stage for path retiming and flow control, which is integrated into the req-ack handshaking mechanism of *Mousetrap*. In contrast, the synchronous baseline switch implements a simple "stall/go" flow control

TABLE III
POST-LAYOUT COMPARISON WITH SYNCHRONOUS SWITCH

	Area	***Latency***	***Energy/bit***
Async. Pro. Multi.	$1789\mu m^2$	$0.67\,ns$	$0.13pJ$
Sync. xpipes	$2693\mu m^2$	$0.64\,ns$	$0.21pJ$
Sync. Overhead	52%	-4%	69%

protocol. To ensure the most lightweight implementation, the synchronous baseline was configured with 2-slot input and output buffers, which is the minimum required to cover the round-trip latency in clock cycles and guarantee full throughput operation [49]. The comparison was conducted in terms of area, latency, and energy per bit following placement and routing in 22nm industrial technology.

The asynchronous switch is 52% more area efficient, mainly due to the more lightweight *Mousetrap* registers compared with synchronous ones, and to the higher number of buffer slots needed to deliver full throughput operation (i.e., flow control is built-in in asynchronous design).

The energy-per-bit of the synchronous switch is 69% higher. Most of the asynchronous savings are due to the use of single-latch-based *Mousetrap* registers (with small area footprint, and low energy and critical-path latency), lack of global clock distribution, and on-demand activation.

Finally, the asynchronous switch pays a small 5% latency overhead due to RTM, the phase conversion circuits and the gates for glitch-free operation.

XII. CONCLUSION

AER routing messages in neuromorphic NoCs can often fit easily within a single-flit packet. Building on this evidence, this paper reports on the design of a lightweight NoC switch that uses an ultra-low power asynchronous design style (2-phase bundled-data) to model a streamlined architecture specialized for short messaging. Benefits have been validated through the model of a real neuromorphic processor for edge computing, and include not only a superior energy efficiency in unicast communications, but also a radical simplification of the multicast routing protocol and of its packet replication mechanism in hardware. Future work will target applicability of this NoC to large-scale neuromorphic computing platforms as well, by focusing on a scalable multicast addressing scheme.

REFERENCES

[1] J. D. Nunes, M. Carvalho, D. Carneiro, and J. S. Cardoso, "Spiking neural networks: A survey," *IEEE Access*, vol. 10, pp. 60738–60764, 2022.

[2] C. Mead, *Analog VLSI and Neural Systems*. Reading, MA, USA: Addison-Wesley, 1989.

[3] F. Akopyan et al., "TrueNorth: Design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 10, pp. 1537–1557, Oct. 2015.

[4] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The SpiNNaker project," *Proc. IEEE*, vol. 102, no. 5, pp. 652–665, May 2014.

[5] M. Davies et al., "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan. 2018.

[6] B. V. Benjamin et al., "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations," *Proc. IEEE*, vol. 102, no. 5, pp. 699–716, May 2014.

[7] C. Mayr, S. Hoeppner, and S. Furber, "SpiNNaker 2: A 10 million core processor system for brain simulation and machine learning," 2019, *arXiv:1911.02385*.

[8] G. Orchard et al., "Efficient neuromorphic signal processing with Loihi 2," in *Proc. IEEE Workshop Signal Process. Syst. (SiPS)*, Oct. 2021, pp. 254–259.

[9] A. Neckar et al., "Braindrop: A mixed-signal neuromorphic architecture with a dynamical systems-based programming model," *Proc. IEEE*, vol. 107, no. 1, pp. 144–164, Jan. 2019.

[10] C. Frenkel, D. Bol, and G. Indiveri, "Bottom-up and top-down approaches for the design of neuromorphic processing systems: Tradeoffs and synergies between natural and artificial intelligence," *Proc. IEEE*, vol. 111, no. 6, pp. 623–652, Jun. 2023.

[11] S. Moradi, N. Qiao, F. Stefanini, and G. Indiveri, "A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs)," *IEEE Trans. Biomed. Circuits Syst.*, vol. 12, no. 1, pp. 106–122, Feb. 2018.

[12] C. Mayr et al., "A biological-realtime neuromorphic system in 28 nm CMOS using low-leakage switched capacitor circuits," *IEEE Trans. Biomed. Circuits Syst.*, vol. 10, no. 1, pp. 243–254, Feb. 2016.

[13] J. Park, T. Yu, S. Joshi, C. Maier, and G. Cauwenberghs, "Hierarchical address event routing for reconfigurable large-scale neuromorphic systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2408–2422, Oct. 2017.

[14] S. Furber, "Large-scale neuromorphic computing systems," *J. Neural Eng.*, vol. 13, no. 5, Oct. 2016, Art. no. 051001. [Online]. Available: https://api.semanticscholar.org/CorpusID:45136509

[15] D. Bertozzi et al., "Cost-effective and flexible asynchronous interconnect technology for GALS systems," *IEEE Micro*, vol. 41, no. 1, pp. 69–81, Jan. 2021.

[16] C. A. R. Hoare, *Communicating Sequential Processes*. Upper Saddle River, NJ, USA: Prentice-Hall, 1985.

[17] A. J. Martin and M. Nystrom, "Asynchronous techniques for system-on-chip design," *Proc. IEEE*, vol. 94, no. 6, pp. 1089–1120, Jun. 2006.

[18] A. R. Young, M. E. Dean, J. S. Plank, and G. S. Rose, "A review of spiking neuromorphic hardware communication systems," *IEEE Access*, vol. 7, pp. 135606–135620, 2019.

[19] A. S. Cassidy, J. Georgiou, and A. G. Andreou, "Design of silicon brains in the nano-CMOS era: Spiking neurons, learning synapses and neural architecture optimization," *Neural Netw.*, vol. 45, pp. 4–26, Sep. 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608013001597

[20] N. E. Jerger, L.-S. Peh, and M. Lipasti, "Virtual circuit tree multicasting: A case for on-chip hardware multicast support," in *Proc. Int. Symp. Comput. Archit.*, Jun. 2008, pp. 229–240.

[21] L. Wang, Y. Jin, H. Kim, and E. J. Kim, "Recursive partitioning multicast: A bandwidth-efficient routing for Networks-on-Chip," in *Proc. 3rd ACM/IEEE Int. Symp. Netw.-on-Chip*, May 2009, pp. 64–73.

[22] S. Park, T. Krishna, C.-H. Chen, B. Daya, A. Chandrakasan, and L.-S. Peh, "Approaching the theoretical limits of a mesh NoC with a 16-node chip prototype in 45 nm SOI," in *Proc. Design Autom. Conf.*, Jun. 2012, pp. 398–405.

[23] T. Krishna and L.-S. Peh, "Single-cycle collective communication over a shared network fabric," in *Proc. 8th IEEE/ACM Int. Symp. Netw.-on-Chip (NoCS)*, Sep. 2014, pp. 1–8.

[24] K. Bhardwaj and S. M. Nowick, "A continuous-time replication strategy for efficient multicast in asynchronous NoCs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 2, pp. 350–363, Feb. 2019.

[25] W. Hu, Z. Lu, A. Jantsch, and H. Liu, "Power-efficient tree-based multicast support for networks-on-chip," in *Proc. 16th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2011, pp. 363–368.

[26] R. A. Stefan, A. Molnos, and K. Goossens, "DAElite: A TDM NoC supporting QoS, multicast, and fast connection set-up," *IEEE Trans. Comput.*, vol. 63, no. 3, pp. 583–594, Mar. 2014.

[27] M. P. Malumbres, J. Duato, and J. Torrellas, "An efficient implementation of tree-based multicast routing for distributed shared-memory multiprocessors," in *Proc. 8th IEEE Symp. Parallel Distrib. Process. (SPDP)*, Oct. 1996, pp. 186–189.

[28] T. Krishna, L.-S. Peh, B. M. Beckmann, and S. K. Reinhardt, "Towards the ideal on-chip fabric for 1-to-many and many-to-1 communication," in *Proc. 44th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Dec. 2011, pp. 71–82.

[29] S. Ma, N. E. Jerger, and Z. Wang, "Supporting efficient collective communication in NoCs," in *Proc. IEEE Int. Symp. High-Perform. Comp Archit.*, Feb. 2012, pp. 1–12.

[30] F. A. Samman, T. Hollstein, and M. Glesner, "Adaptive and deadlock-free tree-based multicast routing for networks-on-chip," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 7, pp. 1067–1080, Jul. 2010.

[31] P. Merolla, J. Arthur, R. Alvarez, J.-M. Bussat, and K. Boahen, "A multicast tree router for multichip neuromorphic systems," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 3, pp. 820–833, Mar. 2014.

[32] J. Navaridas, M. Luján, L. A. Plana, S. Temple, and S. B. Furber, "SpiNNaker: Enhanced multicast routing," *Parallel Comput.*, vol. 45, pp. 49–66, Jun. 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167819115000095

[33] M. T. Moreira, W. Koven, T. F. Wu, H. E. Sumbul, and E. Beigne, "A QDI interconnect for 3D systems using industry standard EDA and cell libraries," in *Proc. 28th IEEE Int. Symp. Asynchronous Circuits Syst. (ASYNC)*, Jul. 2023, pp. 58–59.

[34] A. J. Martin and M. Nystrom, "CAST: Caltech asynchronous synthesis tools," in *Proc. 4th Asynchronous Circuit Design Work. Group Workshop*, USA, 2004.

[35] A. Ghiribaldi, D. Bertozzi, and S. M. Nowick, "A transition-signaling bundled data NoC switch architecture for cost-effective GALS multicore systems," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2013, pp. 332–337.

[36] S. Moradi and R. Manohar, "The impact of on-chip communication on memory technologies for neuromorphic systems," *J. Phys. D, Appl. Phys.*, vol. 52, no. 1, Jan. 2019, Art. no. 014003.

[37] V. Nori, B. Chauviere, M. J. Wibbels, and K. S. Stevens, "A novel asynchronous network-on-chip based on source asynchronous signaling," in *Proc. 28th IEEE Int. Symp. Asynchronous Circuits Syst. (ASYNC)*. Los Alamitos, CA, USA: IEEE Computer Society, Jul. 2023, pp. 71–77, doi: 10.1109/ASYNC58294.2023.10239625.

[38] T. Fischer, M. Rogenmoser, M. Cavalcante, F. K. Gürkaynak, and L. Benini, "FlooNoC: A multi-Tb/s wide NoC for heterogeneous AXI4 traffic," *IEEE Design Test*, vol. 40, no. 6, pp. 7–17, Dec. 2023, doi: 10.1109/MDAT.2023.3306720.

[39] Z. Su, H. Hwang, T. Torchet, and G. Indiveri, "Core interface optimization for multi-core neuromorphic processors," in *Proc. 28th IEEE Int. Symp. Asynchronous Circuits Syst. (ASYNC)*, Jul. 2023, pp. 89–98.

[40] H. Wu, Z. Su, J. Zhang, S. Wei, Z. Wang, and H. Chen, "A design flow for click-based asynchronous circuits design with conventional EDA tools," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 11, pp. 2421–2425, Nov. 2021.

[41] S. M. Nowick and M. Singh, "Asynchronous design—Part 1: Overview and recent advances," *IEEE Design Test*, vol. 32, no. 3, pp. 5–18, Mar. 2015.

[42] M. Singh and S. M. Nowick, "MOUSETRAP: High-speed transition-signaling asynchronous pipelines," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 6, pp. 684–698, Jun. 2007.

[43] K. van Berkel, F. Huberts, and A. Peeters, "Stretching quasi delay insensitivity by means of extended isochronic forks," in *Proc. 2nd Work. Conf. Asynchronous Design Methodol.*, 1995, pp. 99–106.

[44] C. Yan and M. R. Greenstreet, "Verifying an arbiter circuit," in *Proc. Formal Methods Comput.-Aided Design*, Nov. 2008, pp. 1–9.

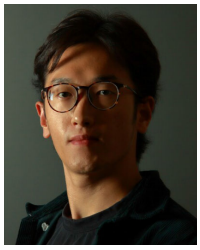
[45] Y. Cui, W. Shan, W. Dai, X. Liu, J. Guo, and P. Cao, "An improved path delay variability model via multi-level fan-out-of-4 metric for wide-voltage-range digital CMOS circuits," *Chin. J. Electron.*, vol. 32, no. 2, pp. 375–388, Mar. 2023.

[46] D. Zendrikov, S. Solinas, and G. Indiveri, "Brain-inspired methods for achieving robust computation in heterogeneous mixed-signal neuromorphic processing systems," *Neuromorphic Comput. Eng.*, vol. 3, no. 3, Jul. 2023, Art. no. 034002.

[47] C. Frenkel and G. Indiveri, "ReckOn: A 28 nm sub-mm^2 task-agnostic spiking recurrent neural network processor enabling on-chip learning over second-long timescales," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, vol. 65, Feb. 2022, pp. 1–3.

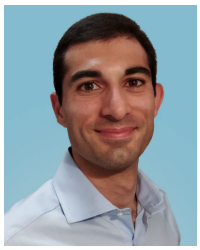
[48] V. R. C. Leite, Z. Su, A. M. Whatley, and G. Indiveri, "Cortical-inspired placement and routing: Minimizing the memory resources in multi-core neuromorphic processors," in *Proc. IEEE Biomed. Circuits Syst. Conf. (BioCAS)*, Oct. 2022, pp. 364–368.

[49] A. Pullini, F. Angiolini, D. Bertozzi, and L. Benini, "Fault tolerance overhead in network-on-chip flow control schemes," in *Proc. 18th Annu. Symp. Integr. Circuits Syst. Design*. New York, NY, USA: Association for Computing Machinery, 1145, pp. 224–229, doi: 10.1145/1081081.1081138.



Zhe Su received the M.Sc. degree in integrated circuits engineering from Tsinghua University, China, in 2020. He is currently pursuing the Ph.D. degree with the Institute of Neuroinformatics, University of Zurich, and ETH Zürich, Zürich.

His research interests is the algorithm-hardware co-design for ultra-low power neuromorphic processors.



Simone Ramini received the M.Sc. degree in electrical engineering from the University of Ferrara, Ferrara, Italy, in 2022.

He is currently a Physical Design Engineer with Infineon Technologies, Graz, Austria. His work is mainly focused on power integrity in industrial microcontrollers, ensuring robustness, and reliability in power distribution networks.

Demetra Coffen Marcolin received the master's degree in electrical engineering from the University of Ferrara in 2023. She is currently pursuing the Ph.D. degree in computer science with The University of Manchester.

The main focus of her research is on asynchronous interconnection networks for ultra-low power communication in neuromorphic computing systems.

Alessandro Veronesi received the master's degree in electrical engineering from the University of Ferrara in March 2020. He is currently pursuing the Ph.D. degree in computer science with the University of Potsdam in collaboration with IHP Microelectronics.

His main research focuses on embedded DNN architectures and dependable DNN systems for mission-critical applications.

Milos Krstic received the Dr.-Ing. degree in electronics from Brandenburg University of Technology, Cottbus, Germany, in 2006.

Since 2001, he has been with IHP, Frankfurt (Oder), Germany, where he leads the Department of System Architectures. Since 2016, he has been a Professor of design and test methodology with the University of Potsdam. For the last few years, his work was mainly focused on fault tolerant architectures and design methodologies for digital systems integration. He has been managing many international and national research and development projects at IHP (GALAXY, EMPHASE, IC-NAO, ENROL, RTU-ASIC, SEPHY, DIFFERENT, VHiSSi, RESCUE, MORAL, and BB-KI Chips). He has published more than 300 journals and conference papers and registered 12 patents.

Giacomo Indiveri (Senior Member, IEEE) is a leading Researcher in the field of neuromorphic engineering, bridging the gap between natural and artificial intelligence. As the Director of the Institute of Neuroinformatics, University of Zurich (UZH) and ETH Zürich; and a dual Professor of neuromorphic cognitive systems with both institutions, he leverages his expertise in electrical engineering, computer science, neuroscience, and machine learning to understand the brain's principles of computation. His research focuses on building low-power mixed-signal circuits, that emulate the dynamics of biological neurons and synapses. He explores how these circuits can perform efficient neural computation, shedding light on the fundamental principles governing spike-based information processing in the brain. By validating brain-inspired computational paradigms in real-world scenarios, his research holds the potential to unlock the tremendous learning capabilities of simple animals, surpassing even the most advanced deep learning solutions while overcoming their limitations in energy requirements.

Davide Bertozzi (Member, IEEE) is a Reader in advanced processing technologies with The University of Manchester, U.K. He has been a Visiting Researcher with Stanford University and several semiconductor industries (STMicroelectronics, NEC America Labs, NXP, and Samsung). The mission of his research is to stay at the forefront of system innovation by leveraging the enabling properties of interconnection architectures and emerging technologies. He contributes to the activities for neuroscience simulation acceleration within the flagship EBRAINS 2.0 project, and takes part to a collaborative effort to bring the celebrated SpiNNaker large-scale neuromorphic computing platform to the next generations of technology, implementation, and architecture. Finally, he is deeply involved in EU-funded projects (TAICHIP, TWIN-RELECT, and AIDA4Edge) to enhance networking between research institutions of the widening countries and top-class leading counterparts, encompassing joint research, knowledge transfer, and exchange of best practices in the fields of AI chip design, EDA tool flows for reliable electronics, and hybrid SNN-ANN models for adaptive edge AI, respectively. His scientific activity has led to roughly 200 publications, with five best paper awards, three best paper award nominations, and one high-impact paper award for one of the five most cited papers in the first 30 years of the International Conference on Computer Design. Dr. Bertozzi received the Wolfgang Mehr Award from IHP Microelectronics, a Leibniz Institute for Innovative Microelectronics in Germany, in 2018, for his interdisciplinary research on photonically-integrated systems.

Steven M. Nowick received the B.A. degree from Yale University and the Ph.D. degree in computer science from Stanford University.

He is a Professor Emeritus of computer science with Columbia University and the former Chair and a Co-Founder of the Computer Engineering Program. He was the Founding Chair with the Center for Computing Systems for Data-Driven Science, Columbia's Data Science Institute. He holds 13 issued U.S. patents. His main research is on design methodologies and CAD tools for synthesis and optimization of asynchronous and GALS digital systems. His particular interests include scalable and low-latency on-chip interconnection networks for shared-memory parallel processors and embedded systems, extreme low-energy digital systems, neuromorphic computing, hardware accelerators, and variation-tolerant global communication. He received the Columbia Engineering School Alumni Distinguished Faculty Teaching Award. He was a recipient of the Alfred P. Sloan Research Fellowship and the NSF CAREER and RIA Awards. He received two Best Paper Awards at the IEEE International Conference on Computer Design and one at the IEEE Async Symposium. He has co-founded the IEEE Async Symposia series, and served as the Program Co-Chair and the General Co-Chair. He was the Program Chair of the IEEE/ACM International Workshop on Logic and Synthesis; and the Program Track/Subcommittee Chair at DAC, DATE, and ICCD conferences. He was the Selection Committee Chair of the ACM/SIGDA Outstanding Dissertation in Electronic Design Automation Award and a member of the Best Paper Award Committees of DAC and ICCAD conferences. He served on the editorial boards of *IEEE Design and Test*, *ACM Journal on Emerging Technologies in Computer Systems*, IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, and IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN.