Received: 22 March 2024 | Revised: 26 July 2024 | Accepted: 14 August 2024

DOI: 10.4218/etrij.2024-0135

SPECIAL ISSUE

ETRI Journal WILEY

Asynchronous interface circuit for nonlinear connectivity in multicore spiking neural networks

Sung-Eun Kim | **Kwang-Il Oh** | **Taewook Kang** | **Sukho Lee** | **Hyuk Kim** | **Mi-Jeong Park** | **Jae-Jin Lee**

Artificial Intelligence SoC Research Division, Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea

Correspondence
Sung-Eun Kim, Artificial Intelligence SoC Research Division, Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea.
Email: sekim@etri.re.kr

Funding information
This work was supported by the Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korea government (24ZS1230, memory-computation convergence neuromorphic computing technology).

Abstract
To expand the scale of spiking neural networks (SNNs), an interface circuit that supports multiple SNN cores is essential. This circuit should be designed using an asynchronous approach to leverage characteristics of SNNs similar to those of the human brain. However, the absence of a global clock presents timing issues during implementation. Hence, we propose an intermediate latching template to establish asynchronous nonlinear connectivity with multipipeline processing between multiple SNN cores. We design arbitration and distribution blocks in the interface circuit based on the proposed template and fabricate an interface circuit that supports four SNN cores using a full-custom approach in a 28-nm CMOS (complementary metal–oxide–semiconductor) FDSOI (fully depleted silicon on insulator) process. The proposed template can enhance throughput in the interface circuit by up to 53% compared with the conventional asynchronous template. The interface circuit transmits spikes while consuming 1.7 and 3.7 pJ of power, supporting 606 and 59 Mevent/s in intrachip and interchip communications, respectively.

KEYWORDS
asynchronous, connectivity, interchip communication, interface circuit, intrachip communication, nonlinear connectivity, spiking neural network

1 | INTRODUCTION

Spiking neural networks (SNNs), inspired by the human brain, use spikes for information transmission [1, 2]. Because of the spike-based operation, SNNs feature a simple architecture and afford high energy efficiency [3, 4]. However, implementing SNNs remains challenging [5], and their scalability is inevitably limited. To achieve scalability, an interface circuit that integrates multiple SNNs must be developed. While conventional synchronous approaches can be used to design such an interface, they suffer from high power consumption of the global clock, large scale clock trees, spike delays, and clock skew. To overcome these issues and enhance scalability, an asynchronous approach for the interface circuit is necessary [6, 7].

The properties of asynchronous circuits differ inherently from those of synchronous circuits [8]. Synchronous circuits rely on a global clock to provide reference time information throughout the system. To implement these circuits, clock buffers are repeatedly inserted to secure a time window around the clock edge and prevent clock skew issues in diverse and scattered functional blocks. These clock buffers consume additional power and increase the silicon area. In contrast, asynchronous circuits do not use discrete time. Instead, they employ asynchronous backbone logic based on handshaking protocols to address the challenges associated with global clock architectures [9, 10].

Recent studies on interface circuits for neural networks focus on implementing nonlinear connectivity in multiple neural cores with reduced spike latency while minimizing hardware resources. A hybrid routing scheme that supports both multicast and unicast reduces peak spike traffic and achieves low spike latency [11]. The merge-and-link scheme improves connectivity by decreasing spike transmission latency from a topological perspective [12]. However, most existing studies rely on synchronous approaches due to implementation convenience, which introduces synchronization issues between different timing points [13, 14]. In contrast, the dual-level mask scheme, as an asynchronous approach, significantly reduces area and spike latency for sparse events but comes with high power consumption and low throughput [15]. In this work, we propose an asynchronous interface circuit that supports nonlinear connectivity, particularly suited for applications with intense event competition.

The architecture of the proposed multicore SNN integrated circuit (IC), as shown in Figure 1, comprises four SNN core blocks and one asynchronous interface block. Each SNN core block comprises spike buffers, synapses, and neural units for neural computations. These four SNN core blocks are designed using analog techniques,

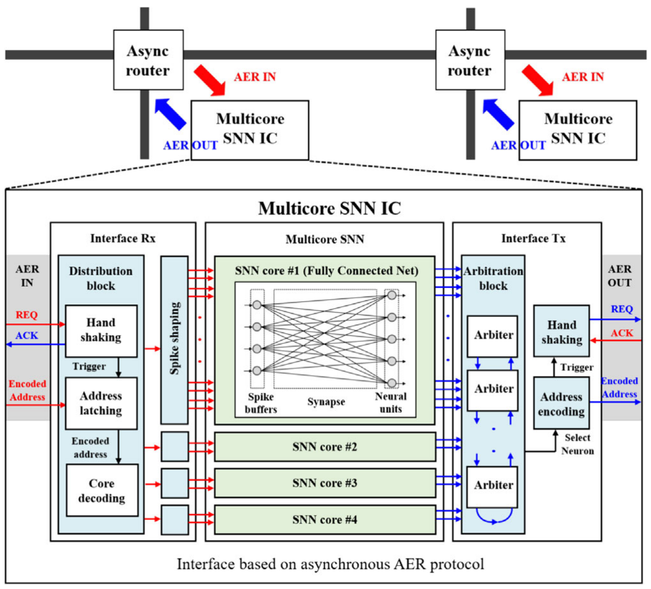


FIGURE 1 Architecture of multicore spiking neural network (SNN) integrated circuit (IC).

and the network is fully interconnected. The asynchronous interface block comprises arbitration and distribution blocks based on a four-phase address-event-representation (AER) handshaking protocol, enabling asynchronous communication channels between the SNN cores. The arbitration and distribution blocks are constructed to support asynchronous nonlinear pipelines. The interface circuit receives an encoded address from an externally transmitting SNN IC, distributes spikes to the spike buffers in the SNN core blocks, accumulates spikes from the neural units in the SNN core blocks, and transmits the neural processing results as an encoded address to an external receiving SNN IC through an interchip communication channel.

The remainder of this paper is organized as follows: Section 2 analyzes the behaviors of conventional templates for asynchronous backbone logic. Section 3 introduces the intermediate latching template and discusses the structure of the arbitration and distribution blocks based on this proposed template for asynchronous communication. Section 4 presents the implementation of the multicore SNN IC to validate the proposed interface circuit, including an evaluation and analysis of the interface block's performance. Finally, Section 5 provides the conclusions.

2 | CONVENTIONAL ASYNCHRONOUS TEMPLATE

The asynchronous approach is a compelling alternative to the synchronous approach for SNNs owing to its high speed and low power consumption. Unlike synchronous methods, the asynchronous approach does not rely on a global clock for timing; instead, the validity of data must be verified in advance. Two primary asynchronous design methods exist, bundled-data (BD) and quasi-delay-insensitive (QDI) designs, both of which provide reference time information to each functional block. In the BD design method, data are represented with binary codes similar to those in synchronous designs. A single wire carries one bit of information, and a control signal is used to validate the data. In contrast, the QDI design method incorporates a control signal within the data itself. The dual-rail encoding is widely used method in QDI systems. In dual-rail encoding, each bit of data is represented by two wires, A and B. A high logic value is represented by setting wire A high and wire B low, while a low logic value is represented by setting wire A low and wire B high. Data validity is verified when the two wires are set to opposite values simultaneously.

In SNNs, data are represented by spikes without values, and the presence of spikes indicates the validity of

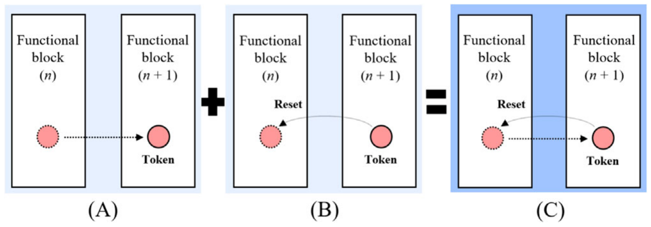


FIGURE 2 Evaluation parameters of asynchronous systems: (A) Forward latency, (B) backward latency, and (C) cycle time = (A) + (B).

the data. Because no additional control signal for data validity is required in SNNs, the Muller pipeline based on the QDI template can be employed [10]. The Muller pipeline controls the overall signal flow over the entire asynchronous system and conveys accurate time information to each functional block based on the handshaking protocol. This renders the circuit insensitive to variations and ensures a reliable operation in asynchronous systems. In this section, we analyze the conventional asynchronous QDI templates for the Muller pipeline proposed in previous studies [16, 17].

The evaluation parameters for asynchronous systems should be defined before analyzing the behaviors of conventional templates. Unlike synchronous systems, the performance of asynchronous systems is determined by the time between the generations of successive valid tokens. Three parameters are defined for asynchronous systems, as shown in Figure 2: forward latency, backward latency, and cycle time. Forward latency is the time required to move a token between two adjacent stages. Backward latency is the time required to reset the stage to receive a new token after shifting it to the next stage. The local cycle time is the sum of forward and backward latencies. It represents the shortest time required to complete handshaking between adjacent stages.

2.1 | Weak-conditioned half buffer (WCHB) template

The WCHB is widely used as an asynchronous template for the Muller pipelines [18]. The structure of the WCHB and its signal flow in the pipeline are illustrated in Figure 3. The WCHB comprises a functional block, a completion detector (CD), two data signals (input and output), and two acknowledgement signals (Left-ack and Right-ack), where t_{func} is the time delay of a functional logic block, t_{CD} is the time delay of the CD, and t_{rst} is the time delay resulting from resetting the functional logic block. The CD indicates whether the data is valid. When a CD is placed on the input side of a

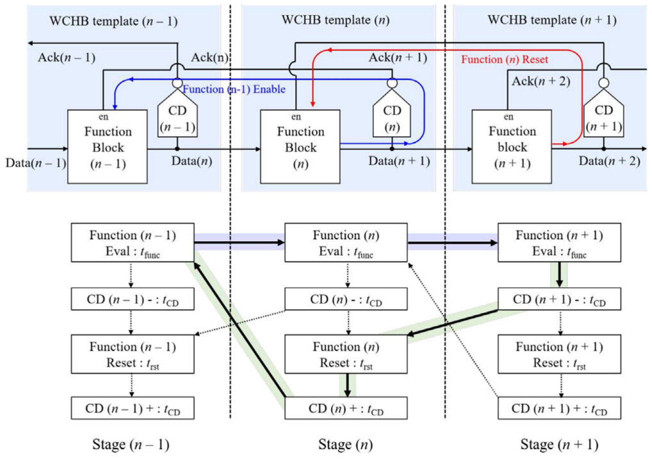


FIGURE 3 Structure and data flow of weak-conditioned half buffer (WCHB) pipeline.

functional block, it can be called as an LCD indicating the validity of the input data. Conversely, when a CD is on the output side, it can be called as an RCD indicating the validity of the output data. The initial state of a WCHB is specified as no input data and no active-low acknowledgement signals. The signal flow of a single WCHB template is described as follows: First, valid input data arrives at the functional block. Next, the output data are propagated to the next functional block after time t_{func}. The output data then activate the Left-ack signal, which resets the input data through the CD block at time t_{CD}. Finally, the activated Left-ack signal releases the input data to its initial state at time t_{rst}. The validity and neutrality of the output data indicate those of the input data [16, 19]. This feature, known as weak-conditioned logic, is fundamental for the WCHB. Additionally, the WCHB cannot hold two distinct tokens for the input and output data, nor can it permit concurrent read/write functions, unlike a full buffer. In fact, a half buffer must be read before the WCHB can be rewritten.

$$\begin{aligned}\text{FL}_{\text{WCHB}} &= 3\times t_{\text{func}},\\ \text{BL}_{\text{WCHB}} &= 2\times t_{\text{CD}} + t_{\text{rst}},\\ \text{CT}_{\text{WCHB}} &= 3\times t_{\text{func}} + 2\times t_{\text{CD}} + t_{\text{rst}}.\end{aligned} \tag{1}$$

To analyze one cycle time for the WCHB, three stages should be built in the pipeline, as shown in Figure 3. To complete one cycle of handshaking in function $(n-1)$ of the WCHB in the pipeline, data must propagate to function $(n+1)$ through function (n), and the acknowledgement signal must return from function $(n+1)$ to function $(n-1)$, as highlighted by the thick lines. The evaluation parameters for the WCHB, forward latency, backward latency, and cycle time, are given in (1). Here, FL_{WCHB} and BL_{WCHB} represent the forward and backward latencies of the WCHB, respectively, and CT_{WCHB} denotes the cycle time, which is the sum of FL_{WCHB} and BL_{WCHB}. FL_{WCHB} employs three t_{func} elements, whereas BL_{WCHB} employs two t_{CD} and one t_{rst}.

2.2 | Precharged half buffer (PCHB) template

The PCHB is an alternative to the WCHB [20]. The structure of the PCHB and its signal flow in the pipeline are shown in Figure 4. The PCHB is composed of a dynamic functional block with a CD at the input and output, also known as the LCD and RCD, respectively, where t_{func} is the time delay of the functional logic block, t_{CD} is the time delay of the CD, t_{pre} is the precharge time of the functional logic block, and t_C is the time for synchronizing the validity and neutrality of the input and output through the Muller-C element (MCE). To confirm that the validity and neutrality of the input and output are secured simultaneously, the MCE is employed, which is a basic circuit in the backbone pipeline of the asynchronous system. Owing to the MCE, the output is transformed to the logic value of the inputs when all inputs are equal. The most commonly used two-input MCE holds the output value until the values of the two inputs match. Notably, the functional block in the PCHB is composed of dynamic logic, which enables faster operation. The functional block in the PCHB operates in two steps: precharge and evaluation. Precharge step is for setting up the initial conditions, and evaluation step is for determining the final output state. Although the PCHB aligns with the fundamental concepts of the WCHB, it separately

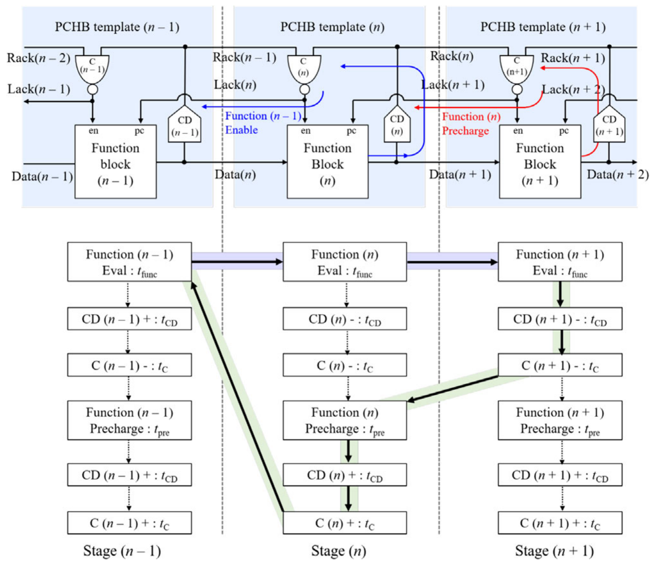


FIGURE 4 Structure and data flow of precharged half buffer (PCHB) pipeline.

examines the validity and neutrality of the input and output using the LCD and RCD, respectively. After examining the validity and neutrality of the input and output, the functional block is enabled, and the functional block in the previous stage is charged to its initial state. The initial state of a single PCHB block is specified as no input data and no active-low acknowledgement signals. Valid input data arrive at the functional block, and the result is propagated to the next functional block after time t_{func}. The validity of both the input and output data activates the Left-ack signal, which resets the input data through the MCE at time $t_{CD} + t_C$. The activated Left-ack signal then releases the input data to its initial state at time t_{pre}.

$$\begin{aligned} FL_{PCHB} &= 3 \times t_{func}, \\ BL_{PCHB} &= 2 \times (t_{CD} + t_C) + t_{pre}, \\ CT_{PCHB} &= 3 \times t_{func} + 2 \times (t_{CD} + t_C) + t_{pre}. \end{aligned} \tag{2}$$

The behavior of a three-stage linear PCHB pipeline is similar to that of a WCHB pipeline except for the additional terms of t_C and t_{pre} for the dynamic logic. To complete one cycle of handshaking in function $(n - 1)$ of the PCHB pipeline, data must propagate to function $(n + -1)$, and the acknowledgement signal must return from function $(n+1)$ to function $(n-1)$, as highlighted path with thick lines, similar to the WCHB pipeline. The evaluation parameters for the PCHB, forward latency, backward latency, and cycle time are given in (2). Here, FL_{PCHB} and BL_{PCHB} represent the forward and backward latencies of the PCHB, respectively, CT_{PCHB} denotes the cycle time, which is the sum of FL_{PCHB} and BL_{PCHB}. The transitions in forward latency for both the PCHB and WHCB are aligned. However, due to the faster dynamic logic used in the PCHB compared to complementary logic in the WCHB, FL_{PCHB} is lower than FL_{WCHB}. In terms of backward latency, the PCHB has more transitions owing to the MCE for controlling the enabled signal, which adds a typically insignificant delay overhead to its cycle time.

3 | DESIGN OF ASYNCHRONOUS INTERFACE CIRCUIT FOR SNN

The conventional asynchronous backbone logic is optimal to provide reference time information for linear asynchronous applications with single input and output channels. However, the connectivity of SNNs is more complex than that of linear asynchronous applications. To scale up a single SNN core to multiple cores, nonlinear connectivity such as join and fork functions should be implemented in the interface circuit. The join function has multiple inputs for a single output channel, whereas the fork function has a single input for multiple output channels. To obtain nonlinear connectivity with multiple pipelines in SNNs, the template for the asynchronous backbone logic must be modified to process these pipelines. Without modifications to the templates, processing nonlinear connectivity inevitably results in delays when the signals interconnect. This section presents an intermediate latching template for asynchronous nonlinear connectivity in SNNs, the structures of the arbitration block for the join function, and the distribution block for the fork function based on the proposed template.

3.1 | Intermediate latching template

To implement nonlinear connectivity in SNNs and configure multiple pipelines for delivering congestion-free spikes in parallel, a template is designed to decouple individual functional blocks with a short cycle time. We present an intermediate latching template designed to reduce the cycle time of each functional block. By decoupling individual functional blocks, we improve throughput through parallel processing and prevent delays in one stage from propagating to neighboring stages.

The structure of the proposed intermediate latching template and its signal flow are shown in Figure 5. The proposed template comprises a functional block, a storage unit, a CD, two data signals (input and output), and two acknowledgement signals (Left-ack and Right-ack), where t_{func} is the time delay of the functional logic block, t_{latch} is the time delay of latching data in the storage, t_{CD} is the time delay of the CD, and t_{rst} is the time delay of resetting the functional logic block. The initial state of the proposed template is specified as no input data and no active-low acknowledgement signals. The signal flow operates as follows: In the write path, valid data arrive at the functional block, and the result is propagated to the

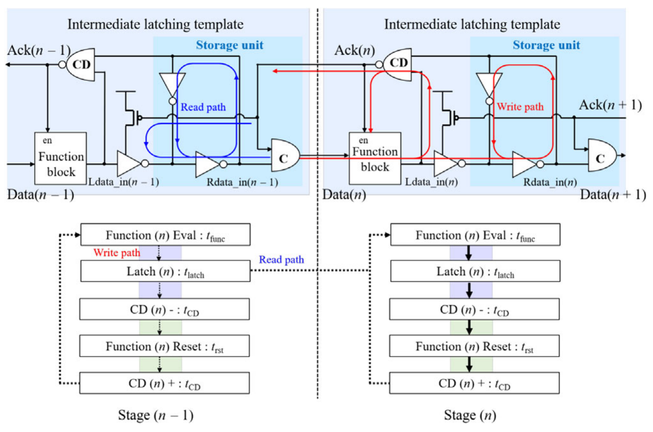


FIGURE 5 Structure and data flow of proposed intermediate latching pipeline.

output of the functional block at time t_{func}. Subsequently, the result is latched during storage at time t_{latch}. The latched data activate the Left-ack signal to reset the storage in the previous stage through the CD block at time t_{CD} and releases the functional block in the current stage to its initial state at time t_{rst}. This establishes latched data, in which the functional block in the current stage can receive data for the next task without waiting for a response from the next stage. The results are latched in the storage unit, waiting for the Right-ack signal from the functional block in the next stage. In the read path, when the next stage is ready to read the results, the Right-ack is activated from the next stage, and the data in the storage unit are released to the functional block in the next stage. The released data then return to its initial state. Because each functional block has its own storage for temporary data regardless of the status of the neighboring functional blocks, modularization with a short cycle time can be achieved in an asynchronous system.

$$\begin{aligned} \text{FL}_{\text{InterLatch}} &= t_{\text{func}} + t_{\text{latch}}, \\ \text{BL}_{\text{InterLatch}} &= 2 \times t_{\text{CD}} + t_{\text{rst}}, \\ \text{CT}_{\text{InterLatch}} &= t_{\text{func}} + t_{\text{latch}} + 2 \times t_{\text{CD}} + t_{\text{rst}}. \end{aligned} \tag{3}$$

The forward latency, backward latency, and cycle time, are expressed in (3), where $\text{FL}_{\text{InterLatch}}$ and $\text{BL}_{\text{InterLatch}}$ represent the forward and backward latencies of the intermediate latching template, respectively, $\text{CT}_{\text{InterLatch}}$ is the cycle time of the intermediate latching template, which is the sum of the forward and backward latencies of the intermediate latching template. The forward latency depends on t_{func} and t_{latch}, whereas the backward latency depends on t_{CD} and t_{rst}. When the proposed template is compared with conventional asynchronous half buffer templates, such as the WCHB and PCHB, cycle time for each template should be analyzed. Conventional templates require three stages to be interconnected for one-cycle operation, a characteristic feature of half buffers used to stabilize signal flow in asynchronous systems. In these templates, each stage is strongly correlated with its neighboring functional stages. The advantages of conventional templates are that they have relatively simple structure and stable data flow in the pipeline based on strong correlation with adjacent stages. However, the strong correlation with neighboring stages results in a long cycle time and prompts a delay in one stage, thus affecting the operating time in neighboring stages. As a result, a one-stage delay can extend the entire operation time.

To resolve the issues of conventional templates, the proposed template suggests to use a single stage for one-cycle operation in the pipeline. To decouple individual functional stages in a pipeline with intermediate latches, it requires additional cost for forward latency, area, and power. The forward latency has an overhead of t_{latch} when compared with other conventional asynchronous templates, such as the half buffer, and additional area and power are required because intermediate latches must be designed inside the template. However, in terms of the signal path, the correlation among the stages is dissolved so that a delay in one stage does not affect the operation time in neighboring stages. This improvement enhances the system's throughput and allows multiple stages to process data simultaneously in parallel. For each stage, this template with its reduced cycle time eliminates the need to wait for its turn to process tasks due to shared hardware resources. When the input data are sparse, conventional templates may be sufficient, as they offer stability with low cost of area and power consumption. However, in applications such as interface circuits with intense data competition, the proposed template enhances throughput by reducing cycle time in the signal path.

3.2 | Arbitration block for join function

The arbitration block implements an asynchronous join function in the interface circuit. It receives parallel spikes from the neural units in multiple SNN cores and provides a token to a selected neural unit for permission to use the communication channel without congestion. To prevent congestion, the arbitration block generates an acknowledgement signal for a selected neural unit using a handshaking protocol. The arbitration block based on the proposed intermediate latching template decouples individual function blocks. This decoupling of neural units from communication reduces response time. Moreover, this functional separation enables parallel neural units to resume their tasks without waiting for the communication channel to be ready.

The structure of an arbitration block with the proposed template is shown in Figure 6. The arbitration block with the intermediate latching template is based on a hierarchical tree structure [21, 22]. It comprises several levels of unit arbitration circuits, each with two input channels. The number of input channels in the arbitration block can be increased by stacking additional unit arbitration circuits. Each unit arbitration circuit forwards the arbitration input to the next level. When a neural unit in the SNN core is selected in an arbitration block, it receives an acknowledgement signal from the top of the tree. Each unit arbitration circuit employs a mutually exclusive function to manage the access to the output channel shared among several independent parties [10]. This function ensures that only one output is activated

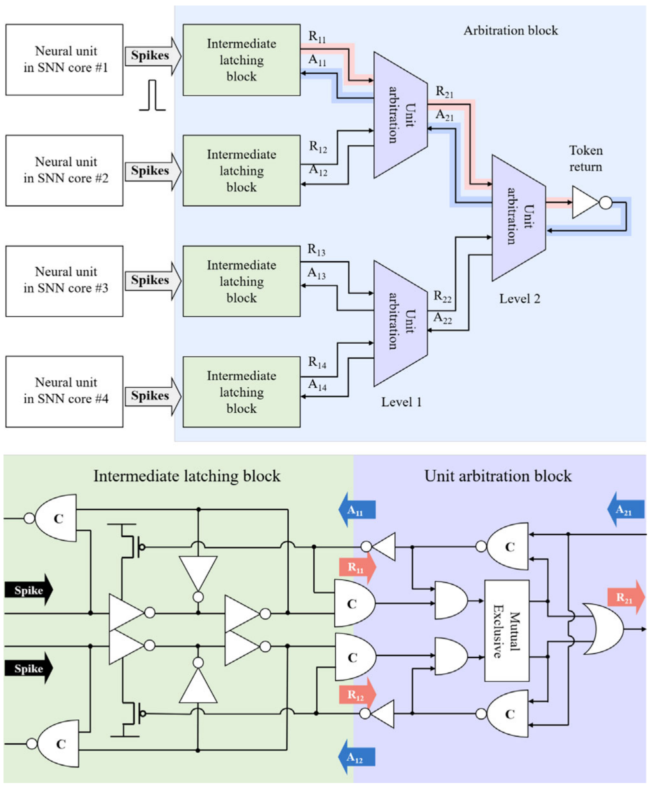


FIGURE 6 Arbitration block with proposed template.

from the two input request signals by blocking the other. If one input request signal arrives before another signal, the subsequent input request signal is blocked while the first request is processed.

Neural units in multiple SNN cores fire spikes with a request signal to obtain a token in the arbitration block for communication. Moreover, interchip communication is a time-consuming process owing to hardware restrictions, such as those from interchip input/output (I/O) pads and wire capacitance. Suppose that a neural unit generates a spike every 10 ns and an interchip communication requires 40 ns to transmit a spike to another SNN core. In conventional asynchronous approaches, a neural unit must stop generating spikes while waiting for interchip communication to complete. As a result, the neural unit loses the opportunity to generate up to four-spikes in 40 ns. In SNN systems with intense competition among neural units, the next firing opportunity might be given to other units. The arbitration block with the proposed intermediate latching template addresses this by resetting the neural units after latching the spike. This allows the intermediate latches to wait for interchip communication instead of the neural units themselves. As a result, neural units can immediately receive the next input spike without waiting for the completion of interchip communication, thus mitigating potential errors from nonlinear parallel spike processing.

3.3 | Distribution block for fork function

The distribution block implements an asynchronous fork function in the interface circuit. Furthermore, it receives the encoded addresses through the communication channel and generates an acknowledgement signal after delivering a spike to the target spike buffer in multiple SNN cores. Before passing the spikes to the target spike buffer in the SNN cores, the spike width must be modified to ensure an appropriate synaptic operation. During communication, the cycle time is determined by the time required to activate an acknowledgement signal on the receiver side. As the number of SNN cores that require spike time information increases, communication delays can introduce errors into system calculations. However, the distribution block with the proposed intermediate latching template reduces the cycle time by generating the acknowledgement signal after latching the encoded address on the receiver side.

The structure of a distribution block with the proposed intermediate latching template is shown in Figure 7. The distribution block comprises three main components: the handshaking block, the intermediate latching block, and the address decoding block. The handshaking block generates an acknowledgement signal in response to a request signal. The intermediate latching block temporarily stores the encoded address, and the address decoding block generates a spike in the target spike buffer. The distribution block employs a hierarchical tree structure to deliver spikes, preventing excessive path searches. This structure is crucial for efficiently delivering spikes to the target neuron and supporting multiple SNN cores.

When a transmitter attempts to send a spike through a communication channel, the request signal from the transmitter is activated with the encoded address of the firing neuron. The handshaking block in the distribution block receives the request signal and triggers the intermediate latches to store the encoded addresses. After the encoded address is latched, an acknowledgement signal is generated from the handshaking block. The latched encoded address is then delivered to the address decoding block when the SNN core block is ready to process the spikes. The decoding block generates a spike in the target spike buffer. The intermediate latching template in the distribution block generates a fast response to the request signal, reducing the cycle time. By generating an acknowledgment signal independently of the next functional block, the proposed template minimizes cycle time and enhances the overall computational speed of the system.

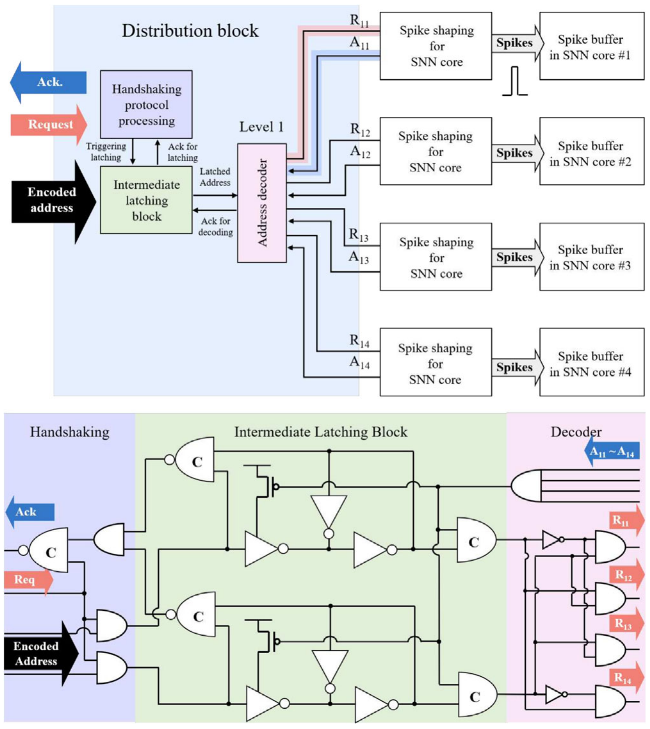


FIGURE 7 Distribution block with proposed template.

3.4 | SNN core block

The multicore SNN IC comprises four SNN cores, each containing spike buffers, synapses, and neural units. Each SNN core receives spikes from the previous layer, processes spikes in parallel, and generates spikes for the subsequent layer. The scale of the SNN core can be expanded using an interface circuit. The SNN cores can be implemented with digital or analog methods [23]. Digital cores utilize a synchronous global clock to implement spike buffers, synapses, and neural units. In a synchronous digital SNN, all components are synchronized with the clock during operation, but this approach fails to adequately mimic the human brain, which operates asynchronously. To overcome the limitations of digital synchronous architectures and better leverage the characteristics of the human brain, analog asynchronous architectures should be adopted to implement SNNs. In an analog implementation, input spikes with charge accumulation are processed based on weights stored in the synapses.

Our implemented SNN core adopts a fully connected crossbar architecture that connects all input spike buffers to all output neural units. Each SNN core consists of 100 input spike buffers, 20 output neural units, and 2000 synapses. The size of each SNN core is adaptable based on the application. Synapses are generated using pulsed currents in a simple basic structure that enhances the flow of synaptic current to accumulate charges in membrane capacitors. The synapses are controlled using an 8-bit weight, and input spikes accumulate charges in the membrane potential according to the frequency and amplitude of the synaptic weights. The output neural units employ a simplest spiking neural model with a leaky integrated-and-fire structure. In this model, input spikes from input spike buffers accumulate over time in the membrane capacitance. When the accumulated charge in the membrane capacitor exceeds a certain threshold, an electrical spike is fired by the output neural units. The encoded address of the fired neural unit is transferred to the target spike buffer in the next layer of the SNN core through the interface circuit.

4 | IMPLEMENTATION AND ANALYSIS

The proposed template is primarily designed to reduce cycle time during asynchronous signal flow, facilitating nonlinear connectivity in multiple pipelines. To validate the improvements of the proposed template, an interface circuit including arbitration and distribution blocks based on the proposed design was fabricated using a 28-nm complementary metal–oxide–semiconductor (CMOS) fully depleted silicon on insulator (FDSOI) process. The layout and die photograph of the fabricated multicore SNN IC are shown in Figure 8. The circuit was designed with analog design environments and implemented using a full-custom approach to optimize signal routing paths. As shown in Figure 8, the interface circuit for core selection is located in the center of the layout, with four SNN cores symmetrically distributed around it. This interface circuit delivers spike signals to the target SNN core, while the interface circuit for neurons attached to each SNN core block transmits spike signals to the target spike buffer in the target SNN core. All signal routing paths from the core to the neurons were symmetrically designed to account for wire delays caused by the length and width of the signal paths. The interface circuit used a 0.9 V nominal supply voltage. In addition, to enhance the compatibility with external digital ICs, standard digital I/O pads of 3.3 V were employed. An evaluation board for the multicore SNN IC was developed to validate the performance of the interface circuit, with a digital synchronous field-programmable gate array (FPGA) installed to test the IC on the evaluation board.

The evaluation board including two multicore SNN ICs expanded through the interface circuit is shown in Figure 9. The multicore SNN ICs on the left and right sides correspond to the first and second layers, respectively. Input spikes for the multicore SNN IC were generated using a digital FPGA IC. The multicore SNN IC for the first layer processes the input spikes from the FPGA and delivers the results to the next multicore SNN IC for the second layer through the interface circuit. The interface circuit merges parallel spikes from the

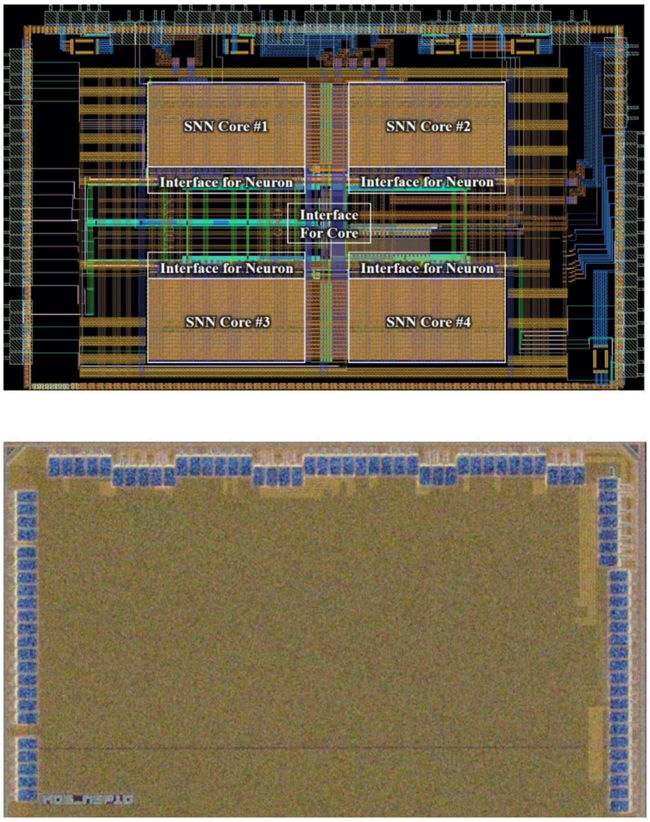


FIGURE 8 Layout and die photograph of multicore spiking neural network (SNN) integrated circuit (IC).

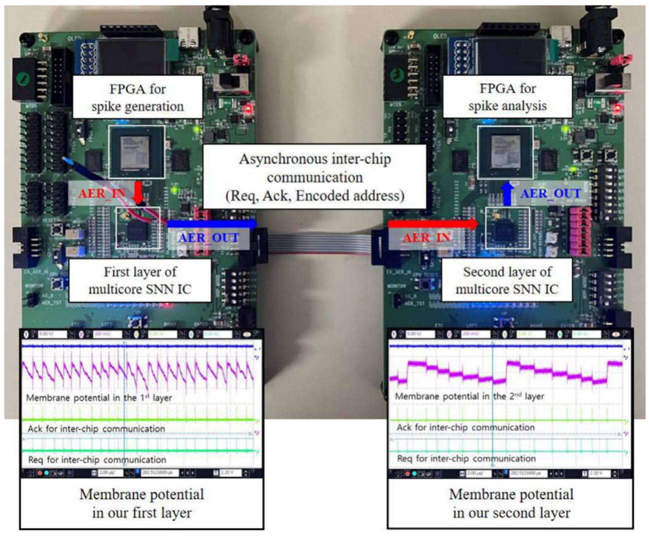

FPGA for spike generation
FPGA for spike analysis
Asynchronous inter-chip communication (Req, Ack, Encoded address)
AER_IN
AER_OUT
AER_IN
AER_OUT
First layer of multicore SNN IC
Second layer of multicore SNN IC
Membrane potential in the 1st layer
Ack for inter-chip communication
Req for inter-chip communication
Membrane potential in the 2nd layer
Ack for inter-chip communication
Req for inter-chip communication
Membrane potential in our first layer
Membrane potential in our second layer

FIGURE 9 Evaluation of expansion using two multicore spiking neural network (SNN) integrated circuits (ICs).

multicore SNN IC in the first layer and distributes them to multicore SNN IC in the second layer. To analyze the operational characteristics of the two layers, the SNN cores were operated in lateral inhibition mode, reducing the number of spikes as the layers deepened. Additionally, the membrane potentials of both layers were monitored. The periods of firing spikes in the first layer was shorter than those in the second layer, confirming that the communication channel was established appropriately using handshaking AER protocols.

The power consumption for the spike transmission was measured to evaluate the energy efficiency of the asynchronous communication channel. Continuous spikes were generated, and the power consumption in the interface circuit was measured. The energy required for spike transmission was calculated by dividing the measured power by the spike count. Power measurements were taken for spike periods of 50 ns, 37.5 ns, 25 ns, and 12.5 ns, including all power consumed under a 0.9 V supply voltage. The energy consumption per spike transmission for interchip communication is shown in Figure 10. The energy consumption per spike decreased as the spike period and supply voltage were reduced. The interface circuit achieved a low energy consumption of 1.7 pJ and 3.7 pJ for spike transmission in intrachip and interchip communications, respectively, at a nominal supply voltage of 0.9 V. Faster spikes resulted in lower energy consumption per spike transmission. When the supply voltage was reduced from the nominal voltage of 0.9 V to 0.6 V for low-energy operation, the energy per spike transmission was the least. However, the lower supply voltage resulted in an additional time latency in the buffer stages, thereby decreasing the throughput of the interface circuit. Throughput measurements for the interface circuit were performed at the nominal supply voltage of 0.9 V.

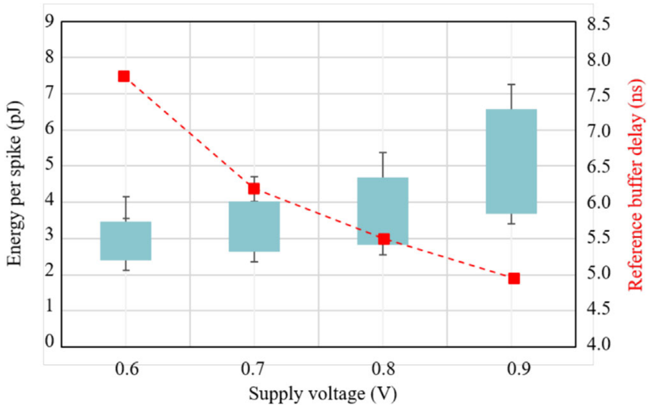


FIGURE 10 Energy per spike transmission (in picojoules) and reference buffer delay (in nanoseconds) for interchip communication.

The delay in spikes refers to the time required to transfer a spike from one SNN core to other SNN core. This can be evaluated by measuring the time delay between a spike fired in the first layer and the spike detected in the second layer. The spike delay encompasses three steps, encoding the spike, latency in the wiring buffers, and decoding the spike. This delay is related to the forward latency of the interface circuit. In intrachip communication, the delay in the wiring buffers is insignificant due to the short signal path and absence of I/O pad capacitance. For intrachip communication, the delay in the wiring buffers was 0.7 ns. When a spike was fired in the SNN core of the first layer, it took 2.9 ns for a spike to be detected in the target spike buffer in the SNN core of the second layer on the receiver side. In interchip communication, owing to long wiring path and high capacitance of I/O pads, the delay in wiring buffers for the exchanging handshake protocol was 5 ns for both the rising and falling edges. The delay for spikes between the SNN cores in interchip communication was approximately 10.6 ns.

Throughput refers to the time interval required to transfer a spike from one SNN core to other SNN core and indicates how quickly the interface circuit can transmit spikes periodically between SNN cores. It is closely related to the cycle time of the interface circuit. As similar with the delay in spikes, the environment for intrachip communication is better than that for interchip communication. The delay on wiring buffers in interchip communication can be extended up to 10 ns due to the handshaking protocol. For intrachip communication, the cycle time for transmitting a spike between SNN cores was 1.7 ns, supporting a throughput of 606 Mevent/s (millions of events per second). In contrast, the interchip communication required 16.9 ns, supporting a throughput of 59.2 Mevent/s through the I/O pads. Compared to the conventional WCHB template, the proposed intermediate latching template reduced cycle times from 3.6 to 1.7 ns in intrachip communication and from 19 to 16.9 ns in interchip communication. The throughput of the proposed template was improved by up to 53%.

A comparison of the performance of previous studies for neuromorphic systems is presented in Table 1. IBM Truenorth, designed with digital neurons and hybrid asynchronous-synchronous networks, uses different supply voltages for intrachip and interchip communications to improve compatibility with external components. It transfers spikes in packet form with a minimum time step of 1 ms, requiring 2.3 pJ with a 0.77 supply voltage for intrachip and 26 pJ with a 1.8 V supply voltage for interchip communication. Dynap exhibits a mixed-signal architecture with analog spiking neurons and analog synapses as the computing units. It comprises a bidirectional

TABLE 1 Performance comparison of previous studies for multicore neuromorphic system. (on: intrachip / off: interchip).

	IBM Truenorth [24]	Dynaps [25]	Intel Loihi [26]	This study
Technology	28 nm	FDSOI 28 nm	FinFET 14 nm	FDSOI 28 nm
Neuron Type	Digital	Analog	Digital	Analog
Routing	Hybrid (Async, Sync)	Async (AER)	Async (Packet)	Async (AER)
Latency (in wiring buffers)	1 to 15 time step (1 time: 1ms)	5 ns (off)	2.1 ns (on) 4.1 ns/6.5 ns (off)	0.7 ns (on) 5 ns (off)
Throughput (Mevent/s)	-	32.3 (off)	-	606 (on)/59 (off)
Energy for spike transmission (pJ)	2.3 pJ@0.77 V (on) 26 pJ@1.8 V (off)	11 pJ@1 V (off)	1.7 pJ@0.75 V (on) 3.5 pJ@0.75 V (off)	1.7 pJ@0.9 V (on) 3.7 pJ@0.9 V (off)

interface block with power and I/O efficiency in interchip communication, utilizing standard digital I/Os with address-event transceiver block that can change the direction of communication. It was composed of FIFO stage and transceiver buffers. Dynap supports 32.3 for unidirection and 28.6 Mevents/s for bidirection communication with 5 ns latency. Intel Loihi, using the Proteus design method for high-performance asynchronous design, supports intrachip and interchip communications extending hierarchically in four planar directions to other chips [27]. It has a latency of 2.1 ns for intrachip communication, and 4.1 ns in direction of east to west and 6.5 ns in direction of north to south for interchip communication. Intel Loihi comprises 128 neural cores, and employs mesh-type network structure with the packet type with energy of 1.7 pJ for intrachip and 3.5 pJ for interchip communication with a 0.75 V supply voltage. This study uses a hierarchical tree-type network structure supporting four SNN cores with AER protocol, achieving the shortest intrachip latency of 0.7 ns, attributed to the scale of the networks and optimized routing paths between SNN cores.

5 | CONCLUSIONS

We proposed an asynchronous interface circuit for nonlinear connectivity in multiple SNN cores. To achieve asynchronous nonlinear connectivity, an intermediate latching template is introduced, which reduces cycle time compared to conventional asynchronous templates. The arbitration and distribution blocks in the interface circuit were designed using this template. An interface circuit that supports multiple SNN cores was designed with full-custom methods for optimizing all the routing paths and fabricated in a 28-nm CMOS FDSOI process. The proposed template enhanced the throughput for communication by up to 53% compared with the conventional asynchronous template. The interface circuit achieved a throughput of 606 and 59.2 Mevent/s in intrachip and interchip communication respectively and low-energy consumption of 1.7 and 3.7 pJ for intrachip and interchip communication at a nominal supply voltage of 0.9 V for one spike transmission. The proposed interface circuit may enable SNNs to be scaled up to large networks with high speed and low energy consumption, likely enhancing artificial intelligence applications.

ACKNOWLEDGMENTS

This work was supported by the Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korea government (24ZS1230, memory-computation convergence neuromorphic computing technology).

CONFLICT OF INTEREST STATEMENT

The authors declare that there are no conflicts of interest.

ORCID

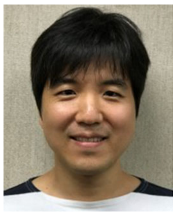
Sung-Eun Kim https://orcid.org/0000-0001-9039-8902
Kwang-Il Oh https://orcid.org/0000-0002-8715-7929
Taewook Kang https://orcid.org/0000-0001-9147-3898

REFERENCES

1. A. Basu, L. Deng, C. Frenkel, and X. Zhang, *Spiking neural network integrated circuits: a review of trends and future directions* (IEEE Custom Integr. Circuits Conf., Newport Beach, CA, USA), 2022, pp. 1–8.
2. Y. Kuang, X. Cui, Y. Zhong, K. Liu, C. Zou, Z. Dai, Y. Wang, D. Yu, and R. Huang, *A 64k-neuron 64m-1b-synapse 2.64 pj/-sop neuromorphic chip with all memory on chip for spike-based models in 65nm CMOS*, IEEE Trans. Circuits Syst. II: Express Briefs **68** (2021), no. 7, 2655–2659.
3. D. V. Christensen, R. Dittmann, B. Linares-Barranco, A. Sebastian, M. Le Gallo, A. Redaelli, S. Slesazeck, T. Mikolajick, S. Spiga, S. Menzel, and I. Valov, *2022 roadmap on neuromorphic computing and engineering*, Neuromorphic Comput. Eng. **2** (2022), no. 2, 022501.
4. A. R. Young, M. E. Dean, J. S. Plank, and G. S. Rose, *A review of spiking neuromorphic hardware communication systems*, IEEE Access **7** (2019), 135606–135620.

5. S. M. Nowick and M. Singh, *Asynchronous design part 1: overview and recent advances*, IEEE Des. Test **32** (2015), no. 3, 5–18.
6. H. K. O. Berge and P. Hafliger, *High-speed serial AER on FPGA* (IEEE Int. Symp. Circuits Syst., New Orleans, LA, USA), 2007, pp. 857–860.
7. A. Lines, P. Joshi, R. Liu, S. McCoy, J. Tse, Y.-H. Weng, and M. Davies, *Loihi asynchronous neuromorphic research chip*, Energy **10** (2018), 15.
8. M. Mahowald, *Vlsi analogs of neuronal visual processing: a synthesis of form and function*, Ph.D. Thesis, California Institute of Technology, 1992.
9. X.-G. Guan, X.-Y. Tong, and Y.-T. Yang, *Quasi delay-insensitive high speed two-phase protocol asynchronous wrapper for network on chips*, J. Comput. Sci. Technol. **25** (2010), no. 5, 1092–1100.
10. J. Spars and S. Furber, *Principles asynchronous circuit design*, Springer, 2002.
11. C. Ding, Y. Huan, H. Jia, Y. Yan, F. Yang, L. Liu, M. Shen, Z. Zou, and L. Zheng, *A hybrid-mode on-chip router for the large-scale FPGA-based neuromorphic platform*, IEEE Trans. Circuits Syst. I: Regular Papers **69** (2022), no. 5, 1990–2001.
12. P. J. Zhou, Q. Yu, M. Chen, G. C. Qiao, Y. Zuo, Z. Zhang, Y. Liu, and S. G. Hu, *Fullerene-inspired efficient neuromorphic network-on-chip scheme*, IEEE Trans. Circuits Syst. II: Express Briefs **2023** (2023), 1376–1380.
13. B. Lin, L. Wang, Z. Yang, J. Tie, G. Zhou, and X. Yu, *A configurable inter-chip connection architecture for multicore neuromorphic chip* (4th Int. Conf. Frontiers Technol. Inform. Comput., Qingdao, China), 2022, pp. 928–931.
14. M. Sadeghi, Y. Rezaeiyan, D. F. Khatiboun, and F. Moradi, *Hardware implementation of a resource-efficient router for multi-core spiking neural networks* (IEEE Int. Symp. Circuits Syst., Monterey, CA, USA), 2023, pp. 1–5.
15. S. Ouyang, K. Zhou, H. Jiang, C. Li, J. Liang, F. Zhu, X. Zhang, and Q. Liu, *A scalable area-efficient low-delay asynchronous AER circuits design for neuromorphic chips*, IEEE Trans. Circuits Syst. II: Express Briefs **2024** (2024), 2804–2808.
16. A. M. Lines *Pipelined asynchronous circuits*, Master's Thesis, California institute of Technology, 1998.
17. A. J. Martin, *The limitations to delay-insensitivity in asynchronous circuits, Beauty is our business: a birthday salute to Edsger W. Dijkstra*, Springer, 1990, pp. 302–311.
18. Y. Thonnart, E. Beigné, and P. Vivet, *A pseudo-synchronous implementation flow for WCHB QDI asynchronous circuits* (IEEE 18th Int. Symp. Asynchronous Circuits Syst., Kgs. Lyngby, Denmark), 2012, pp. 73–80.
19. C. L. Seitz, *System timing*, Introduct. VLSI Syst. **1980** (1980), 218–262.
20. P. A. Beerel, R. O. Ozdag, and M. Ferretti, *A designer's guide to asynchronous vlsi*, Cambridge University Press, 2010.
21. A. M. T. Linn, C. Shoushun, and Y. K. Seng, *Adaptive priority toggle asynchronous tree arbiter for AER-based image sensor* (IEEE/IFIP 19th Int. Conf. VLSI System-on-Chip, Hong Kong, China), 2011, pp. 66–71.
22. S.-C. Liu, T. Delbruck, G. Indiveri, A. Whatley, and R. Douglas, *Event-based neuromorphic systems*, John Wiley & Sons, 2014.
23. F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G.-J. Nam, and B. Taba, *Truenorth: Design and tool flow of a 65 Mw 1 million neuron programmable neurosynaptic chip*, IEEE Trans. Comput.-Aided Design Integr. Circ. Syst. **34** (2015), no. 10, 1537–1557.
24. P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, and B. Brezzo, *A million spiking-neuron integrated circuit with a scalable communication network and interface*, Science **345** (2014), no. 6197, 668–673.
25. N. Qiao and G. Indiveri, *A bi-directional address-event transceiver block for low-latency inter-chip communication in neuromorphic systems* (IEEE Int. Symp. Circuits Syst., Florence, Italy), 2018, pp. 1–5.
26. M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, and Y. Liao, *LOIHI: a neuromorphic manycore processor with on-chip learning*, IEEE Micro **38** (2018), no. 1, 82–99.
27. P. A. Beerel, G. D. Dimou, and A. M. Lines, *Proteus: an ASIC flow for Ghz asynchronous designs*, IEEE Des. Test Comput. **28** (2011), no. 5, 36–51.

AUTHOR BIOGRAPHIES



Sung-Eun Kim received his B.S. degree in Electrical & Computer Engineering from Hanyang University, Seoul, and his M.S. degree in Electrical Engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Republic of Korea, in 2002 and 2004, respectively. Since March 2004, he has been with the Electronics and Telecommunications Research Institute (ETRI), Daejeon, Republic of Korea, where he is currently a Principal Researcher. He has been primarily involved in researching analog circuit design for human body communications, power management of energy harvesting systems, and spiking neural networks.

Kwang-Il Oh received his B.S. degree in Electrical Engineering from Kyungpook National University, Daegu, Republic of Korea, in 2002, and his M.S. and Ph.D. degrees in Electrical Engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Republic of Korea, in 2004 and 2009, respectively. From March 2009 to February 2015, he was a senior engineer at LX Semicon Co., Ltd., Daejeon, Republic of Korea, where he was involved in the high-bandwidth interface circuits design for LCD modules. Since March 2015, he has been with the Electronics and Telecommunications Research Institute (ETRI), Daejeon, Republic of Korea, where he is involved in neuromorphic chip design. His interests include low power analog circuits, signal integrity, and neuromorphic circuits.

Taewook Kang received his B.S. and M.S. degrees in Electronics and Electrical Engineering from Pohang University of Science and Technology, Pohang, Republic of Korea, in 2005 and 2007, respectively, and his Ph.D. degree from Chungnam National University, Daejeon, Republic of Korea in 2023. Since February 2007, he has been with the Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea, where he is currently a Principal Researcher. His research interests include human body communications, radio channel modeling, power management of energy harvesting systems, betavoltaic battery, statistical and adaptive signal processing, and spiking neural networks.

Sukho Lee received his Ph.D. degree in Information Communications Engineering from Chungnam National University, Daejeon, Republic of Korea, in 2010. He is currently a principal researcher with the AI Edge SoC Research Section, Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea. His current research interests include ultralow power system-on-chip design, embedded system design, video codec design, and video image processing.

Hyuk Kim received his B.S. and M.S. degrees in Electronic Engineering from Chonnam National University, Gwangju, Republic of Korea, in 1997 and 1999, respectively. Since August 1999, he has been with the Electronics and Telecommunications Research Institute (ETRI), Daejeon, Republic of Korea, where he is currently a Principal Researcher. He has been primarily involved in the development of error-correcting codes for mobile communication modem. His research interests include error correction codes, wireless communications, human body communications, wearable computing systems, and system on chip (SoC) design.

Mi-Jeong Park received her B.E., M.E., and Ph.D. degrees from Hanbat National University, Daejeon, Republic of Korea, in 2005, 2007, and 2012, respectively. Since 2012, she has been with the Electronics and Telecommunications Research Institute (ETRI), Daejeon, Republic of Korea, where she is currently a Senior Engineer. She has been working as a digital circuit designer. Her research interests include the embedded software design for hardware control as well as the digital frontend design for wireless and human body communication.

Jae-Jin Lee received his B.S., M.S., and Ph.D. degrees in Computer Engineering from Chungbuk National University, Cheongju, Republic of Korea, in 2000, 2003, and 2007, respectively. He is currently a Project Leader with the Low-Power AI System-on-Chip Design Research Division, Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea. His research interests include ultralow-power embedded RISC-V processor designs and event driven neuromorphic computing architectures for brain-inspired spiking deep neural networks.

How to cite this article: S.-E. Kim, K.-I. Oh, T. Kang, S. Lee, H. Kim, M.-J. Park, and J.-J. Lee, *Asynchronous interface circuit for nonlinear connectivity in multicore spiking neural networks*, ETRI Journal **46** (2024), 878–889, DOI 10.4218/etrij.2024-0135