IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 6 August 2023, accepted 28 August 2023, date of publication 1 September 2023, date of current version 7 September 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3311031

RESEARCH ARTICLE

R-MaS3N: Robust Mapping of Spiking Neural Networks to 3D-NoC-Based Neuromorphic Systems for Enhanced Reliability

WILLIAMS YOHANNA YERIMA, (Student Member, IEEE), KHANH N. DANG, (Member, IEEE), AND ABDERAZEK BEN ABDALLAH, (Senior Member, IEEE)

Graduate School of Computer Science and Engineering, The University of Aizu, Aizuwakamatsu, Fukushima 965-8580, Japan

Corresponding author: Williams Yohanna Yerima (d8222115@u-aizu.ac.jp)

This work was supported in part by the University of Aizu through Competitive Research Funding (CRF), 2023; and in part the VLSI Design and Education Center, University of Tokyo, Japan, in collaboration with Synopsys, Inc., and Cadence Design Systems, Inc.

ABSTRACT Neuromorphic computing utilizes spiking neural networks (SNNs) to offer power/energy-efficient solutions for complex machine-learning problems in hardware. However, neural circuits are prone to faults caused by variability in the manufacturing flow, process variations, and manufacturing defects. This work proposes a mapping approach, R-MaS3N, that leverages the reuse of existing neurons for robust mapping of SNNs to a 3D-NoC-based neuromorphic system (NR-NASH). A heuristic-based partitioning technique is employed to partition neurons in the layers of an SNN application using neuron firing patterns. Moreover, a neuronal partitioning approach cluster mapped neurons in the layers of the neuromorphic neural circuits based on connectivity patterns and spiking activities. Evaluation results show that the proposed fault-tolerant mapping method maintains a remapping efficiency of 100% with a fault rate of 40% in the 3D NoC-based neuromorphic system. With a NoC system configuration of $4 \times 4 \times 4$ and 256 neurons per cluster, our approach has a remapping time of $71\times$ less than the previous approach with the same NoC system configuration parameters. In addition, the mean time to failure (MTTF) of the mapping method for system configuration $5 \times 5 \times 5$ NoC size at a 40% fault rate surpasses the previous method at 20% fault rate by 16% for $4 \times 4 \times 4$ NoC size.

INDEX TERMS Reliable neuromorphic, mapping, neural reuse, 3D-NoC, clustering.

I. INTRODUCTION

Spiking neural network (SNN) models mimic the brain's biological computations. SNN models communicate through synapses which are links between neurons [1], [2], [3]. The accuracy of SNNs is inferior to that of state-of-the-art artificial neural networks (ANNs) [4]. Nevertheless, their biological plausibility and their unique energy efficiency characteristics have drawn interest [4], [5].

These applications are implemented in neuromorphic computing hardware such as ODIN [2], NASH [6], [7], IBM Truenorth [2], Loihi [8], and SpiNNaker [5], [8] in order to explore their energy-efficient capabilities. Similar to the human brain, these computing hardware have multiple neural circuits interconnected by a shared interconnect in the form of tile-based architectures [4], [5], [6], [9], [10]. Each tile consists of a neuron or synapse circuit, peripheral logic, and a network interface (NI) for passing AER packets across the network [5], [11].

To execute SNNs on neuromorphic hardware, neurons and synapses must be mapped to their neuronal and synaptic circuits. However, the brain's neuronal circuit could be susceptible to faults from various sources, such as noise in synaptic transmission and fluctuations in post-synaptic

The associate editor coordinating the review of this manuscript and approving it for publication was Mohammad Hossein Moaiyeri.

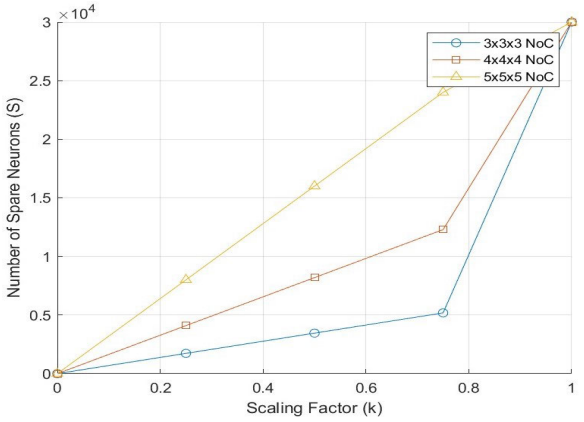


FIGURE 1. Motivation example 1: Relationship between scaling factor and number of redundancy neurons in large scale NoC-based neuromorphic systems.

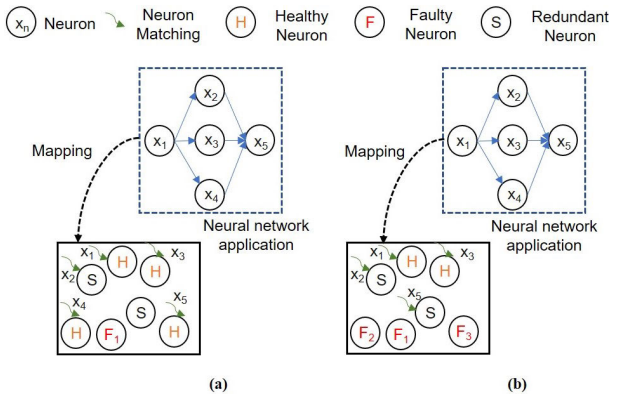


FIGURE 2. Motivation example 2: (a) A neural network application mapped to a neural circuit having redundancy. (b) A neural network application mapped to a neural circuit having redundancy depleted after several failures occurred at double rates.

potentials [12], eventually leading to errors in neuronal circuit outputs. Likewise, neural circuits in neuromorphic hardware may fail due to electrical interference and cross-talk between neurons [12], [13].

SNNs have been mapped to neuromorphic hardware using several approaches and methodologies. Existing techniques, notably in [14], [15], [16], and [17], emphasized hardware performance at the expense of resiliency and robustness, which are critical factors for maintaining reliable computation output for optimal performance. Consequently, it is imperative to find the most efficient way to map SNN applications to neuromorphic hardware which would have significant implications for the performance and reliability of the application.

A. BACKGROUND AND MOTIVATION

Recent research shows that the human brain can tolerate or recover from synaptic or neuron faults by reorganizing structurally and functionally [24], [25], [26]. Nevertheless, the time required to recover depends on the type of fault [25]. Additionally, neural networks exhibit some intrinsic fault tolerance properties. However, these properties are limited and cannot be generalized to all neural network models [24]. Our previous fault injection experiments in [7] and [26] showed that SNNs have some inherent fault tolerance properties. Furthermore, the experiments demonstrate the impact of faults on SNN accuracy in digit classification tasks. As neuromorphic hardware utilizes SNNs for efficient computations and low power consumption, resiliency and fault tolerance are key areas to consider before deployment in critical systems [27], [28].

[0]This footnote provides additional information about Fig. 1:
* For a NoC-based system with N as the number of neurons/cluster, W as the NoC size, the total neurons P in the system is $N \times W$.
* For example, let N=256 and W=5 × 5 × 5, P=256×5 × 5 × 5=32000. The required S for the NoC system for k=0.25 is 0.25 × 32000=8000
* The scaling factor (k) depends on various factors such as the system architecture and reliability requirements.

A popular fault-tolerant technique is the addition of redundancies [24]. This concept has widely been used to achieve resilience and robustness in computing systems. This technique is effective, which is one of its apparent advantages. Our work in [7] and [26] rely on augmented redundancy for neuron mapping to a 3D NoC-based neuromorphic hardware to ensure fault tolerance and reliability recovery.

However, as neuromorphic systems continue to scale up in size and complexity, and with a high degree of fault, the number of spare neurons required for repairs in the neuromorphic system also increases in size. Fig. 1 describes this scenario. In the context of a neural system, let P represent the total number of neurons, and S denote the count of spare neurons required for fault tolerance. A linear relationship between P and S can be expressed as $S = k \times P$, where k is a scaling factor determining the proportion of spare neurons needed for each 3D NoC-based neuromorphic system. Fig. 1 illustrates that as the size of a 3D NoC-based neuromorphic system scale up, there is a corresponding increase in the demand for spare neurons necessitating additional physical space to accommodate them. In addition, It may also require extensive computational efforts to design an optimal mapping solution. This poses significant cost challenges in the design of integrated circuits and other hardware implementations. Another downside of redundancy is that it is finite despite being highly effective and reliable.

In Fig. 2, we illustrate how fault tolerance mapping with redundancy can become constrained over time and ineffective for prior studies in [7] and [26]. Fig. 2 describes a neural network application with 5 tasks mapped to a neural circuit (NC) with 7 neurons. The NC is assumed to have two redundant neurons, and one randomly fails. Mapping the neural network application will not affect NC performance since it has more redundant repair resources as shown in Fig. 2a.

During run-time, in the event of an additional fault and subsequent remapping, task X_4 would remain unmapped. This outcome occurs due to the depletion of redundant

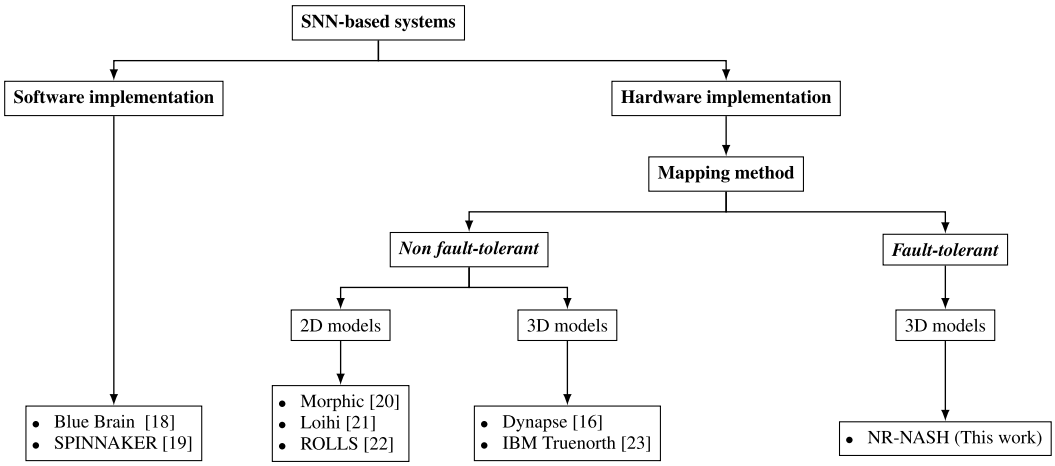


FIGURE 3. A taxonomy of spiking neural network-based systems based on implementation, models, and mapping methods.

resources, a situation depicted in Fig. 2b. As a result of this unmapping, the NC's performance could potentially degrade thereby exerting an adverse impact on the overall system performance.

Several strategies may be employed, including dynamic resource allocation [29], optimizing allocation based on fault patterns [6], and adding additional spare neurons periodically to replenish redundant memory resources [30], [31]. However, these options are costly especially in hardware. With integrated chips, it is not feasible to introduce redundancy periodically. We propose re-purposing existing neurons in the system relying on the neural reuse theory [32], [33]. This is due to the limitations of the current state of the art, our prior proposals in [7] and [26], and the disadvantages of alternative solutions. Our investigation into brain fault tolerance unveiled insights into re-purposing specific brain areas to accommodate errors caused by faulty neurons. These findings support our argument that a brain-inspired system can enhance fault tolerance by reusing neuronal populations in designated areas.

Our approach offers a clear advantage in scalability. This is because the size of a 3D NoC-based neuromorphic system can increase without increasing the size of neurons required for fault tolerance. Furthermore, by eliminating the reliance on finite fault-tolerant resources, our approach ensures a more sustainable and efficient system with low mapping costs. This work's contributions are summarised as follows:

- A robust mapping scheme (R-MaS3N) that leverages a neuron reuse strategy for mapping spiking neural networks (SNNs) to 3D-NoC-based neuromorphic systems. This mapping scheme eliminates an increase in the size of a system due to the use of redundancy-based fault-tolerant approaches.
- A heuristic-based method for partitioning neurons in the layer of an SNN application. This approach provides a striking balance between performance and reliability.
- A partitioning technique that clusters neurons within the layers of the neuromorphic system. This approach ensures that underutilized neurons are first leveraged for fault tolerance and neuron utilization recovery before most utilized.
- Design and evaluation of the proposed mapping approach. The assessment validates the effectiveness of the mapping strategy in achieving reliable and robust mapping of SNNs to 3D-NoC-based neuromorphic systems.

II. PRIOR WORKS

SNN-based systems can be implemented either in software or hardware as depicted in Fig. 3. For hardware implementation often referred to as neuromorphic systems, the implementation could be in a 2D or 3D platform. In Fig. 3, we present a taxonomy of well-known related studies.

For 3D models like the dynapse processors, to implement a fault-tolerant mapping method, its hierarchical routing grid architecture might require more complex fault detection and rerouting mechanisms to fully capitalize on the hierarchical structure. On the other hand, fault-tolerant mapping on our NR-NASH can be easily implemented since multicast routing allows for simultaneous data distribution to multiple destinations.

This section presents known works on fault tolerance and mapping techniques for reliability in neuromorphic computing systems.

A. NEURAL REUSE THEORY

Multiple cognitive functions could be performed by the brain using existing neural circuits. Authors in [34] proposed the neural reuse theory. The theory is a form of neuroplasticity whereby neural elements developed for one purpose are repurposed for another. Furthermore, the theory states that plasticity occurs at individual neuronal levels and large-scale brain regions [33]. While the theory is conceivable, several

underlying assumptions must be considered [33]. As claimed by the authors in [32], [33], and [35], some neurons in the brain may borrow sub-components to perform specific tasks in addition to contributing to non-primary tasks.

The authors in [32] hypothesized a redeployment perspective about the neural reuse theory. The authors suggested that local circuits in the brain may perform low-level computations. These computations can be repurposed for many higher-level functions. The authors concluded that neural reuse theories are not full-fledged theories of how the brain works. Instead, the theory describes how neural resources are deployed to support cognitive operations. Additionally, the theory provides a framework for understanding how the brain achieves cognitive flexibility and efficiency.

Perhaps re-purposing neural resources can inspire new computational models and algorithms. We aim to reuse some parts of brain-inspired hardware for continued information processing support from an economical and fault-tolerant perspective.

B. FAULT-TOLERANCE IN NEURAL COMPUTATION

Neural computation involves information processing and computation performed by the human brain or artificial neural networks (ANNs). However, like any computational system, neural computation can be susceptible to faults or errors. The errors could be caused by process variations, thermal issues, and leakages that result in computational errors [28]. These errors can affect the accuracy, reliability, or integrity of computation output results [28]. Authors in [36] and [37] proposed to handle these errors by utilizing explicit redundancy for both neurons and synapses of the neural network (NN).

In another approach, authors in [28] proposed to identify individual NN components with different functionality. The authors proposed to train the NN components separately with different objectives for improved performance and higher generalization. However, since redundancy increases the NN structure, our previous work in [26] aimed to identify and remove faulty components based on an idea similar to pruning [38], [39].

Our SNN fault-tolerant method in [26] provides an effective solution to neuron failures during computation than the method proposed by the authors in [39]. This is because the work by the authors in [39] targeted faulty synapse connections instead of the whole faulty neuron. Furthermore, this approach increases the computational overhead in identifying and tracking faulty synapse connections. Neural network application training and retraining with diversified parameters is another method proposed for handling errors in neural computation by the authors in [40] and [41]. This method is costly and time-consuming, especially in hardware.

C. SNN MAPPING IN NEUROMORPHIC HARDWARE

The routing architecture in a neuromorphic system makes the task of mapping applications challenging. Despite these challenges, there have been several proposals to achieve or maintain optimum system performance and/or determine how to trade off hardware performance with reliability after mapping. However, mapping applications to neuromorphic hardware still presents non-trivial solutions.

To map an SNN application to neuromorphic hardware, two procedures are typically used; dividing neurons into several clusters according to hardware constraints and placing the clusters into hardware processors. Mapping strategies like Espine [1], Neumap [14], SpineMap [16], PACMAN [16], [42], and PSOPART [43] employs these two part procedure. As part of these approaches, heuristic algorithms [14], [16], [44] are used for the space search and particle swarm optimization (PSO) [42] for the optimization process.

In PSOPART, neurons are mapped directly to cores using an instance of PSO. In PACMAN, neurons are mapped to SpiNNaker cores on a first-come, first-served basis. The PSO optimization technique is employed in Neumap, Espine, and SpineMap, but neuron partitioning occurs before mapping. Although these existing methods are effective, they have limitations especially when scaling the size of the hardware and applications. There are two main reasons for these limitations: First, neurons of an application can fail independently as a result of internal and external influences [26]. Secondly, because neurons can fail independently, scaling the application size results in more neurons that can fail. Consequently, hardware performance could be negatively affected. However, in these mapping methods, reliability and resilience are traded against performance metrics such as power consumption, spike latency, on-chip network congestion, and throughput.

D. FAULT-TOLERANT MAPPING IN NEUROMORPHIC HARDWARE

Limited work has been done on developing mapping strategies with fault tolerance for neuromorphic hardware. Redundancy, dynamic reconfiguration, and fault detection are common mechanisms used to design fault-tolerant mapping strategies [24], [45]. A combination of these fault-tolerant mechanisms with common approaches such as quadratic programming [15], [16], integer linear programming [16], genetic algorithms [7], and min-max optimization [7], [28] has been employed in developing robust mapping techniques for neuromorphic hardware.

Authors in [46] proposed the fault-tolerant mapping of a memristor-based crossbar using integer linear programming and hierarchical clustering. This approach reduces hardware costs while simultaneously improving the mapping rate. However, it requires significantly more computation and execution time and is limited to 2D neuromorphic hardware.

The authors in [47] also proposed a fault-tolerant mapping technique based on pruning. Using the approach, applications are successfully mapped to the hardware. However, the pruning process complicates the mapping strategy and may lead to incompatibility.

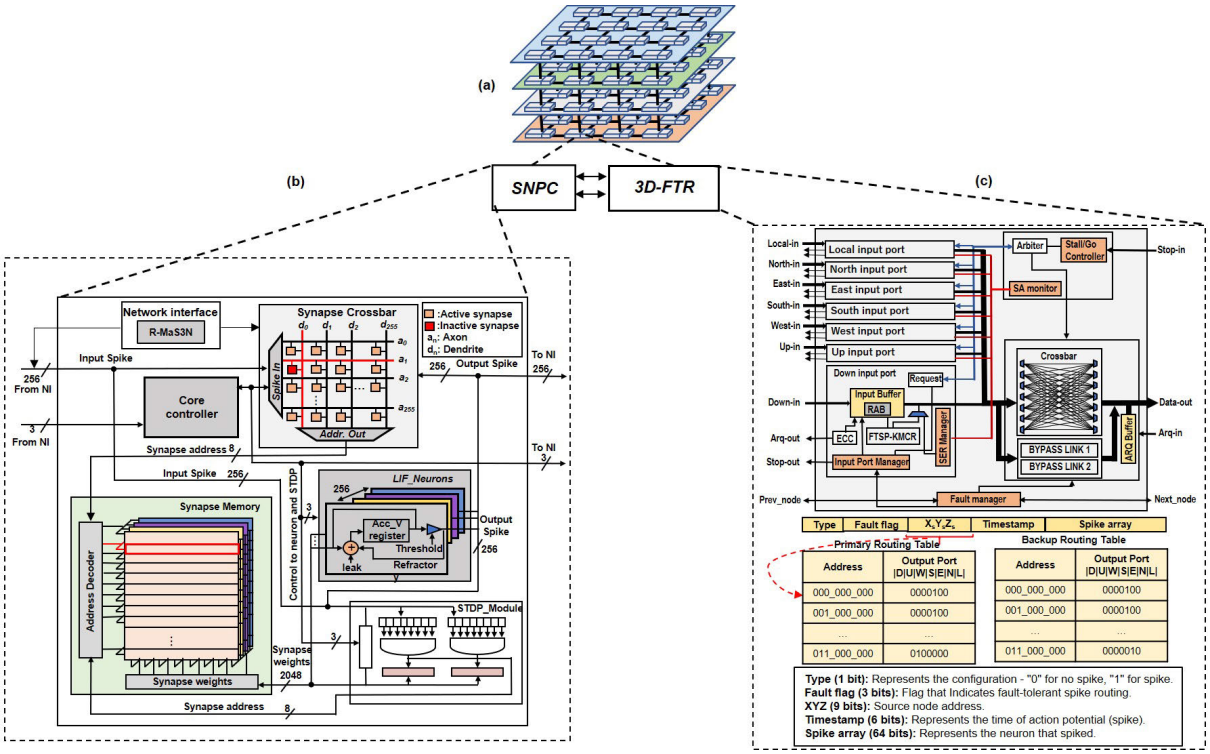


FIGURE 4. A high-level view of the robust 3D NoC-based neuromorphic system (NR-NASH): (a) 4 × 4 × 4 system configuration, (b) Spiking neuro processing core, (c) 3D Fault-tolerant router.

In our prior work in [7], we utilized min and max and a genetic algorithm with redundancy elements to ensure fault tolerance. The method achieves 100% mapping success with reduced migration costs. However, the method is time-consuming and relies on finite resources. Also, if defective components exceed the redundancy, it becomes ineffective. To address the limitations of our prior work in [7], our recent work in [26] suggests that faulty neurons should be repaired based on their contribution levels to a particular process. This method maps even if faulty ones outnumber redundancy components. However, some more contributing neurons might not be selected for repairs which could negatively impact a system's performance. Additionally, scaling up the system size means increasing the size of redundancy components for repairs which could also get exhausted over time.

Authors in [45] proposed a run-time mapping scheme under a lifetime constraint. The mapping scheme dynamically maps incoming applications on multi-core systems. The mapping method adopts a borrowing strategy to manage many-core resources at multiple scales. The technique considers only aging parts of the hardware and not the components of the neural circuit.

R-MaS3N differs from other methods that use traditional fault-tolerant mechanisms. R-MaS3N uses a novel concept based on re-purposing neurons for fault tolerance with low mapping costs for neuromorphic circuits. The mapping scheme eliminates the need for increased system size when employing redundancy-based fault-tolerant approaches. Furthermore, it mitigates the finite resource bottleneck limiting traditional fault-tolerant methods' scalability and performance.

III. NEUROMORPHIC HARDWARE OVERVIEW

We discuss the neuromorphic system in this section. According to Fig. 4, the neuromorphic system is 3D NoC-based with STDP adaptive capability. The system generally comprises a spiking neuro-processing core (SNPC) and a 3D FT router. As part of the SNPC, there is a network interface (NI). Encoders and decoders are incorporated into the NI for decoding and encoding spikes. The NI supports the mapping process as part of the neuromorphic system.

A. SPIKING NEURON PROCESSING CORE (SNPC)

The SNPC (Spiking Neuro Processing Core) shown in Fig.4 consists of several components: the LIF array, synapse memory, synaptic crossbar, network interface (NI), control unit, and the STDP learning module.

Within the LIF array, the membrane voltage of a neuron is computed by accumulating input synapse values. A leak value reduces these synapse values to simulate their decay operations, like in biological neurons. The remaining value is compared against a voltage threshold. If the accumulated value surpasses the voltage threshold, the neuron fires an output spike and enters a refractory stage. On the other

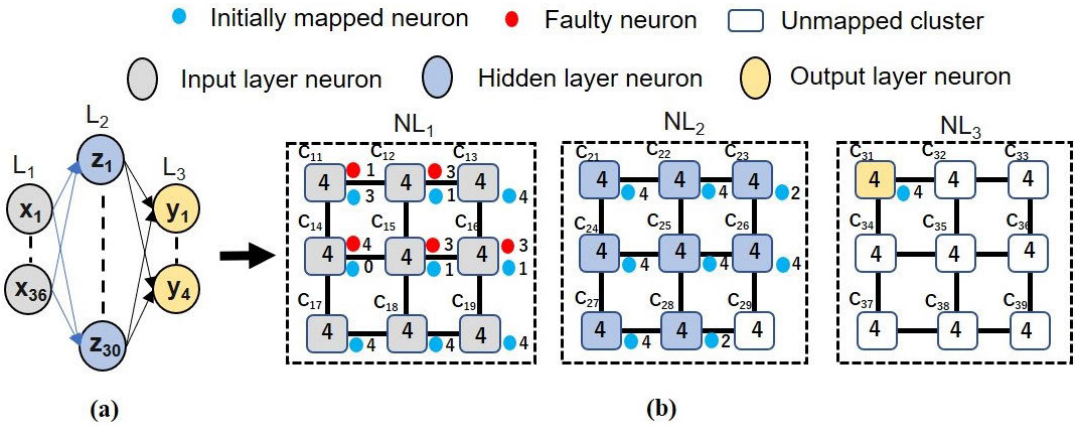


FIGURE 5. An illustration of SNN mapping example on a 3 × 3 × 3 NoC-based neuromorphic system: (a) Neural network application, (b) Neural network application mapping to neuro cores of the 3D NoC-based neuromorphic system based on our mapping method in [48]. However, it's important to note that there are some faulty neurons in clusters C_{11},C_{12},C_{14},C_{15} and C_{16} of NL_1 leading to fewer mapped neurons than the expected maximum capacity.

hand, if the accumulated value does not exceed the threshold, no output spike is generated.

The synapse crossbar implemented using a crossbar architecture represents the synaptic connections between neurons. A 1-bit value represents each synapse, and the corresponding synaptic weights are stored in the synapse memory. Presynaptic spikes sent to the SNPC are received at the synapse crossbar. Spikes from the postsynaptic neurons are identified based on their synapses [48]. The weights associated with these synapses are fetched from the synapse memory and transmitted to the LIF neuron for accumulation.

To facilitate learning, the STDP learning module updates the synaptic weights using trace-based STDP learning [48]. The learning module uses 16 presynaptic spikes grouped according to their arrival time relative to a postsynaptic spike [48]. During the learning process, spike traces arriving before the postsynaptic spike are incremented, while those coming afterward are decremented.

The operation of the SNPC is overseen by the control unit, which operates in six distinct states. The first state is ''idle,'' during which the SNPC awaits presynaptic spikes. Once presynaptic spikes arrive, the second state, ''download,'' is initiated to download the presynaptic spikes. Following the download, the third state, ''accumulation,'' commences, where the spikes are weighted and sent to the LIF neurons for accumulation. After the accumulation, the fourth state, ''leak,'' begins allowing the membrane potential of the neurons to leak. Subsequently, the membrane potential is compared against a predetermined threshold in the fifth state. If the membrane potential exceeds the threshold, an output spike is fired. The sixth state activates the STDP learning module. Learning is performed if the learning conditions are met, and a signal is sent to the control unit to reset to the idle state. If the learning conditions are unsatisfied, the SNPC returns to the ''idle'' state.

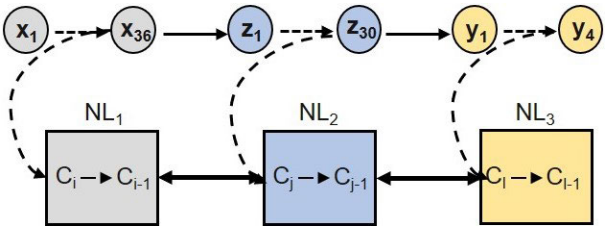


FIGURE 6. An illustration of the mapping sequence on the 3D NoC-based neuromorphic hardware described in Fig. 5. Neurons from each layer of the network application are mapped to the corresponding layer of the neuromorphic system.

The network interface (NI) is an internal component of the SNPC and incorporates the mapping method. The NI supports single and burst transaction modes for reading and writing weight memory and parameters of each neuron [6]. Additionally, it ensures communication between neurons through the on-chip network framework using an encoder and decoder. More details about the SNPC operational dynamics can be found in [6], [7], and [48].

B. 3D FT ROUTER

The 3D FT-router illustrated in Fig. 4 employs a multicast routing algorithm called K-means clustering routing (KMCR) for spike distribution [49]. To alleviate congestion in packet traffic, a variant called shortest path k-means clustering routing (SP-KMCR) was proposed and utilized. A fault-tolerant approach known as fault-tolerant shortest path k-means clustering routing (FTSP-KMCR) was introduced to address potential faults in route links, building upon the SP-KMCR concept [49]. This technique ensures that spike data reaches the destination router even in the presence of a fault along the route.

The router incorporates a random access buffer (RAB) [49] and Bypass-on demand link [48], [49] to handle faults in

the input buffer and crossbar during packet forwarding. The router features seven input and output ports with four ports designated for intra-layer connections to other routers. The routing process within the router involves four pipeline stages: buffer writing (BW), where received spike packets are stored in the input buffer; routing calculation (RC) which determines the next address in either the X, Y, or Z dimension for the source packet; switch allocator (SA) responsible for stall/go flow control, triggering the matrix-arbiter scheduler to allocate the appropriate port of the next router or local SNPC; and crossbar traversal (CT) ensuring that the packet traverses the crossbar to the allocated output port [48]. The 3D-FT router is discussed in more detail in [6], [7], [49], and [50].

IV. R-MaS3N: ROBUST MAPPING OF SNNs TO NEUROMORPHIC SYSTEMS

A. PROBLEM FORMULATION

In a system with an application model, the mapping problem can be formulated as finding a placement function $P : S \rightarrow N_c$.

S represents the application neurons, and N_c represents the set of neuron cores in the neuromorphic system. The objective of a mapping algorithm F is to determine the optimal placement of neurons S onto N_c as represented by the equation:

$$F(S, N_c) = P \tag{1}$$

In a neuron core N_{c_i}, each neuron S_i is mapped to its corresponding coordinates (x_i, y_i) by the placement function P. This can be expressed as follows:

$$P(S_i) = (x_i, y_i) \tag{2}$$

However, some S neurons of the application model may remain unmapped after mapping due to F faulty neurons in the N_c of the neuromorphic system. We introduce Equation 3 to represent these unmapped neurons.

$$U = S - M - F \tag{3}$$

M represents mapped neurons as determined by Equation 2.

Neurons unmapped to any N_c during the initial mapping process are represented by U. To address this issue, we need to find a mapping strategy that uses Q underutilized least active neurons to map the remaining U unmapped neurons. The remapping algorithm aims to efficiently map the U unmapped neurons to Q underutilized neurons for fault tolerance and enhanced reliability. We introduce the remapping procedure in Equation 4.

$$Remap(U, Q) = P' \tag{4}$$

where P' is the updated placement function resulting from the remapping function that maps U unmapped neurons to Q underutilized neurons.

B. PROBLEM DEFINITION

Fig. 5 illustrates the mapping function P presented by Equation 1 with faulty neurons in a 3D NoC-based neuromorphic system. The mapping methodology we proposed in [48] is layer-to-layer based. The mapping process involves assigning application neurons to clusters within each layer of the 3D NoC-based neuromorphic system. This assignment is performed sequentially, starting with a cluster and mapping application neurons until all available neurons within that cluster are utilized. The process then proceeds to the next cluster in that layer. This continues until all clusters with available neurons in that particular layer have been mapped with application neurons. Using the method, we map the neural network application in Fig. 5a to the 3D NoC-based neuromorphic system in Fig. 5b. Fig. 6 describes the mapping sequence.

From Fig. 5b, each cluster in the 3D NoC-based neuromorphic system has 4 neurons. NL_1 of the 3D NoC-based neuromorphic system has 36 neurons with 1, 3, 2, 4, 3, and 3 faulty neurons in C_{11}, C_{12}, C_{13}, C_{14}, C_{15}, and C_{16} clusters respectively. L_1 of the neural network application also has 36 neurons. After initial mapping, 16 application neurons are left unmapped. Therefore, all application neurons required for a certain process are not available on the hardware. This may result in reduced reliability or even failure to perform the desired tasks due to degraded or incomplete functionality.

Fig. 7 illustrates how the proposed R-MAS3N addresses this issue to enhance system reliability. Based on the initial mapping shown in Fig. 5b, there are 16 unmapped neurons of L_1 of the neural network application. The neurons are partitioned into most active and least active partitions based on their spiking behavior. After partitioning, 10 neurons belong to the most active partition while 6 are assigned to the least active partition as described in Fig. 7b. In the next step, all mapped neurons in NL_1 of the 3D NoC-based neuromorphic system are partitioned into high and less-active regions.

According to Fig. 7c, clusters C_{15}, C_{18}, and C_{19} belong to the less-active region. The mapped neurons in C_{15}, C_{18}, and C_{19} clusters are sorted to determine underutilized and most utilized neurons using rank sorting. C_{18} and C_{19} have 2 and 3 underutilized neurons respectively after rank sorting. 1, 2, and 1 neurons appear to be most utilized in C_{15}, C_{18}, and C_{19} clusters respectively. Since we are mapping for fault-tolerance and enhanced reliability, R-MAS3N first maps 10 neurons in the most active partition to underutilized neurons in C_{18} and C_{19} clusters. Similarly, 6 neurons in the least active partition are mapped to the most utilized neurons in C_{15}, C_{18}, and C_{19} clusters.

C. R-MaS3N METHODOLOGY

The authors in [51] demonstrated how existing components can be reused or repurposed for improved performance. The authors proposed a novel method for reusing convolution layers without adding new ones. While their objective was

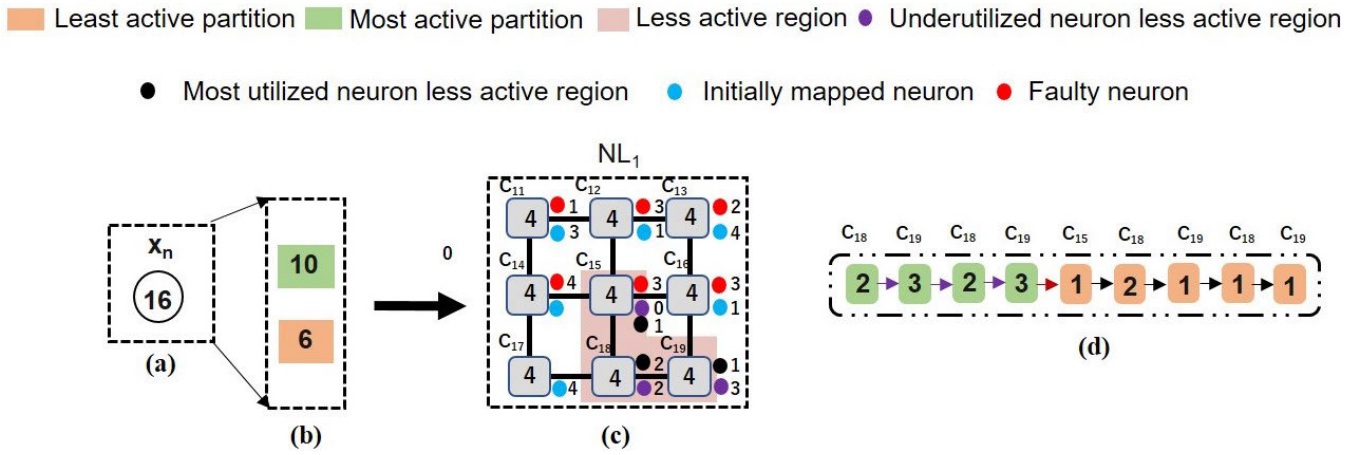


FIGURE 7. An illustration of the proposed solution on a 3 × 3 × 1 NoC-based neuromorphic system for the mapping issue described in Fig. 5: (a) Unmapped neurons of L_1 of the neural network application after initial mapping, (b) Unmapped neurons partitioned into most active and least active partitions, (c) Partitioned unmapped neurons in most active and least active partitions are remapped to neurons in the less-active regions of the layer of the 3D NoC-based neuromorphic system, (d) The remapping sequence in NL_1.

to enhance performance, our goal is to improve reliability through fault tolerance.

In our proposed method, mapping begins with the initial mapping of neurons of an SNN model to clusters of a 3D NoC-based neuromorphic system. Once the 3D NoC-based neuromorphic system is mapped with the SNN application, we randomly set some neurons of the 3D NoC-based neuromorphic system to be faulty.

To achieve fault tolerance and enhanced system reliability, R-MaS3N uses a two-step methodology. Firstly, unmapped layer neurons are partitioned into the most active and least active partitions based on their firing patterns. In the second step, neurons in the 3D NoC-based neuromorphic system are clustered into high and less-active regions. In the less active region, neurons are rank sorted based on their spiking activities. By partitioning, only neurons in the less active region of the 3D NoC-based neuromorphic system are used for mapping in order to achieve fault tolerance for enhanced reliability. In this way, high-active neurons can avoid being mapped high-spiked neurons.

A crucial component of the R-MaS3N approach is reusing existing neurons. Using the reuse strategy minimizes resource waste by utilizing existing neurons efficiently. Below, we provide a detailed design description of two significant procedures in the remapping step of the R-MaS3N. In this section, clustering and partitioning are interchangeably used.

1) SNN LAYER PARTITIONING (SLP) DETAILED DESIGN

The mapping process starts after initial mapping leaves some application neurons unmapped due to faults in the 3D NoC-based neuromorphic system. The first step in the remapping process is partitioning the application model neurons into two partitions based on their firing patterns. Algorithm 1 formalizes our idea. We introduce the denotation C_{most} for the most active neurons cluster and C_{least} for the least active neurons cluster. We determined these partitions using

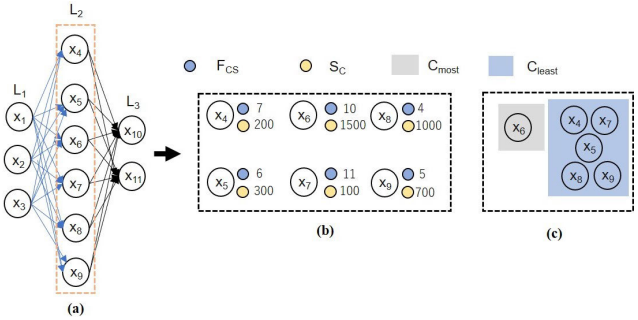


FIGURE 8. Example of an SNN layer partitioned into C_{most} and C_{least} neuron partitions : (a) SNN application, (b) Neurons characteristics layout, (c) Neurons cluster layout.

metrics such as spike counts and consecutive spike frequency. To measure these metrics, let $N = n_1, n_2, \ldots, n_m$ denote the set of all neurons in the SNN model, where m is the total number of neurons.

We denote S_c as a spike count. The S_c of a neuron n_i is the total number of spikes generated by that neuron over a specific period. It can be calculated as follows:

$$S_c(n_i) = \sum_j (t_j - t_{j-1}) \quad \forall j \tag{5}$$

where t_j denotes the time stamp of the $j-th$ spike generated by neuron n_i.

Similarly, we denote the frequency of consecutive spike counts as F_{cs}. The F_{cs} of a neuron n_i is the average number of consecutive spikes generated by that neuron. It can be calculated as follows:

$$F_{cs}(n_i) = \frac{S_c(n_i)}{N_{se} - 1} \tag{6}$$

where N_{se} is the total number of spike events generated by a neuron n_i.

To perform partitioning, we calculate the average spike count Avg_{S_c} and the average frequency of consecutive spike counts $Avg_{f_{cs}}$ across all neurons in the system using Equations 7 and 8. Neurons with spike counts and frequency of consecutive spike counts above the respective averages are clustered as the most active neurons (C_{most}) according to Equation 9. In contrast, neurons with spike counts and frequency of consecutive spike counts below or equal to the averages are clustered as the least active neurons (C_{least}) using Equation 10.

$$\text{Avg}_{S_c} = \frac{1}{m}\sum S_c(n_i) \quad \forall n_i \in N \tag{7}$$

$$\text{Avg}_{f_{cs}} = \frac{1}{m}\sum F_{cs}(n_i) \quad \forall n_i \in N \tag{8}$$

$$C_{\text{most}} = \{n_i \in N \mid S_c(n_i) > \text{Avg}_{S_c} \text{ and } F_{cs}(n_i) > \text{Avg}_{f_{cs}}\} \tag{9}$$

$$C_{\text{least}} = \{n_i \in N \mid S_c(n_i) \leq \text{Avg}_{S_c} \text{ or } F_{cs}(n_i) \leq \text{Avg}_{S_c}\} \tag{10}$$

To better understand the SNN layer neurons partitioning method with algorithm 1, consider the example illustrated in Fig. 8. Suppose we have a 3-layer MLP as described in Fig. 8a. Using algorithm 1 to cluster neurons of layer 2, let's say neurons X_4, X_5, X_6, X_7, X_8, and X_9 as described in Fig. 8b have F_{cs} of 7, 6, 10, 11, 4, and 5 and S_c of 200, 300, 1500, 100, 1000, and 700 respectively. The $Avg_{f_{cs}}$ computed with Equation 7 is 7. The Avg_{S_c} computed using Equation 8 is 633. We place these neurons in the C_{most} and C_{least} partitions according to Equations 9 and 10. Neurons X_4, X_5, X_7, X_8, and X_9 are placed in the C_{least} partition while X_6 is placed in the C_{most} partition. This is described in Fig. 8c.

When addressing the mapping problem in Fig. 5, neurons in the C_{most} partition are mapped first to the underutilized neurons, and then to the most utilized neurons from the C_{less} partition of the 3D NoC-based neuromorphic system. The same applies to unmapped applications neurons in the C_{least} partition. The SNN layer partitioning problem is distinct from the classical graph partitioning problem [15], [16], [43]. The classical graph partitioning problem divides a graph into subsets to optimize objectives like edge reduction between partitions and balanced sizes. However, similarities exist between the two, as both involve dividing elements such as neurons or graph nodes into groups based on specific criteria.

2) NEURONAL PARTITIONING (NP) DETAILED DESIGN

After partitioning neurons in the layer(s) of the SNN application, the next step is to cluster neurons on the 3D NoC-based neuromorphic hardware into high-active and less-active regions. Underutilized neurons are first prioritized for reuse before the most utilized neurons. Algorithm 2 formalizes clustering neurons in layer(s) of the 3D NoC-based neuromorphic system into high-active and less-active regions. For neurons to be classified as highly active, they must satisfy two conditions: having a high number of positive synaptic connections (PC) than negative synaptic connections (NC) and a spike count S_c above the average spike count. We assume these parameters to be derived from the configuration of the 3D NoC-based neuromorphic hardware.

Algorithm 1 SNN Layer Partitioning Algorithm

```
procedure LAYER Partitioning(N)        ▷ Input: Set of neurons N
    Calculate Avg_{S_c} and Avg_{f_cs}
    Initialize empty sets C_most and C_least
    for n_i ∈ N do
        if S_c(n_i) > Avg_{S_c} and F_cs(n_i) > Avg_{f_cs} then
            Move n_i to C_most
        else
            Move n_i to C_least
        end if
    end for
    return C_most, C_least   ▷ Output: Neurons in C_most and C_least regions
end procedure
```

Algorithm 2 Neuronal Partitioning

```
procedure NEURON Partitioning(n)   ▷ Input: Set of all mapped neurons M_n
    Calculate Avg_{S_c}   ∀n_i ∈ M_n.
    Initialize empty sets C_high and C_less.
    for each neuron n_i ∈ M_n do
        Calculate PC.
        Calculate NC.
        Calculate S_c.
        if PC > NC and S_c > Avg_{S_c} then
            Move neuron n_i to C_high.
        else
            Move neuron n_i to C_less.
        end if
    end for
    return C_high, C_less   ▷ Output: Neurons in C_high and C_less regions
end procedure
```

We denote the set of neurons in the high active region as C_{high} and the set of neurons in the less active region as C_{less}.

We also denote PC as the number of positive synaptic connections an n_i has. The PC of n_i is computed using Equation 11:

$$PC(n_i) = \sum(\text{Number of positive synapses connected to } n_i) \tag{11}$$

Similarly, we denote NC as the number of negative synaptic connections for a neuron n_i. The NC of n_i is calculated using Equation 12.

$$NC(n_i) = \sum(\text{Number of negative synapses connected to } n_i) \tag{12}$$

Algorithm 3 Rank-Based Sorting

```
procedure RankSort(n)                ▷ Input: Set of n ∈ C_less
    Initialize an empty list rank_list
    for n_i ∈ C_less do
        Calculate the rank_score(n_i), based on S_c
        Append (n_i, rank_score(n_i)) to rank_list
    end for
    Sort rank_list in ascending order of rank scores
    return rank_list                 ▷ Output: Sorted list of n and their rank scores
end procedure
```

To cluster neurons into either a C_{high} or a C_{less} region, we apply the following criteria:

A neuron belongs to the highly active region ($n_i \in C_{\text{high}}$) if Equations 13 and 14 hold true.

$$PC(n_i) > NC(n_i) \tag{13}$$

$$S_c(n_i) > \text{Avg}_{S_c} \tag{14}$$

For a neuron to be classified as less active ($n_i \in C_{\text{less}}$), it does not meet the criteria for high activity.

Once partitioning is completed, the next step is to identify underutilized and most utilized neurons within the C_{less} region by performing rank-order sorting using algorithm 3. We consider their S_c as the primary criterion. First, we calculate the S_c for each neuron $n_i \in C_{less}$ region. Next, we assign a rank score (RS) to each neuron based on its S_c. The rank score can be derived by ranking S_c in ascending order and assigning lower scores to neurons with fewer S_c. Neurons are then sorted based on their rank scores in ascending order. Neurons having the lowest rank scores indicating fewer S_c are placed at the top of the sorted list. Neurons in the first half of the sorted list are termed as underutilized while neurons in the second half are termed most utilized.

To better understand the neuron partitioning method with algorithm 2, consider the example illustrated in Fig. 9. Suppose we have a layer of a NoC-based neuromorphic system as described in Fig. 9. Using the NP algorithm to determine the C_{high} and C_{less} areas in the layer of the 3D NoC-based neuromorphic system, let's say the neurons in each cluster in the layer have their respective S_c and the number of PC and NC computed according to Equations 5, 11 and 12 respectively. The Avg_{S_c} is 369 according to equation 7. We used Equations 13 and 14 to classify neurons into the C_{high} and C_{less} partitions.

Clearly, all neurons in C_1 to C_4, C_6, and neuron X_9 in C_5 have S_c above Avg_{S_c}. In addition, they also have more PC than NC. Therefore, neurons in these clusters are assigned to the C_{high} partition. All neurons in C_7, C_8, C_9, and neuron X_{10} in C_5 have more NC than PC and have S_c below the computed Avg_{S_c}. Therefore, these neurons are assigned to the C_{less} partition.

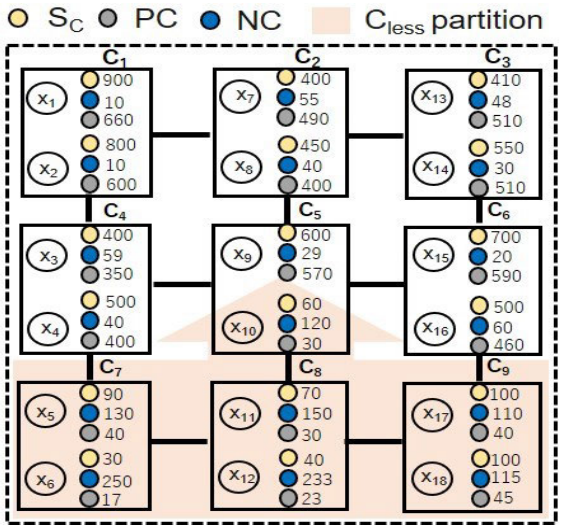


FIGURE 9. Example of neurons in clusters of a 3 × 3 × 1 NoC-based neuromorphic system partitioned into C_{high} and C_{less} partitions using algorithm 2.

Since the task of mapping SNNs to neuromorphic systems remains a challenge, our mapping method is not without constraints. The mapping constraints are the following:

From Equation 3, we earlier denoted M as the number of mapped neurons and U as the number of unmapped neurons.

1) A neuron can be mapped to only one neuron during the initial mapping.

$$M \leq U \tag{15}$$

2) A neuron can only be remapped twice. This implies that the total number of mappings (both initial and remapping) cannot exceed three.

$$M \leq 3U \tag{16}$$

3) If the number of unmapped neurons is greater than twice the number of mapped neurons M of C_{less}, reusing a neuron for mapping fails.

$$U > 2 \times M_{C_{less}} \tag{17}$$

V. EVALUATION RESULTS

In this section, we present the results of our evaluation of R-MaS3N. We provide insights into the effectiveness, efficiency, and reliability of R-MaS3N through analysis and discussion. The evaluation first involved SNN application mapping experiments to a 3D NoC-based neuromorphic system.

The second part involves evaluating the performance of our proposed mapping strategy to the 3D NoC-based neuromorphic system having some cluster neurons randomly set to be faulty. Lastly, we analyzed our mapping method's reliability in the 3D NoC-based neuromorphic system for different NoC sizes. This was to demonstrate the robustness of our mapping method in handling fault scenarios.

TABLE 1. Configuration used for evaluating R-MaS3N.

Parameter	Value
Neurons per cluster	128 and 256
NoC sizes	$3 \times 3 \times 3$, $4 \times 4 \times 4$ and $5 \times 5 \times 5$
Fault rates (%)	10, 20, 30, and 40

A. EVALUATION METHODOLOGY

For the first stage of our evaluation, we proposed several SNN applications with their detailed description presented in Table 2. These proposed applications are indicated with the '*MLP*' preceding a number (e.g., *MLP*_1794) where the number represents the total number of neurons in the application as reported in column 4 of Table 2. Columns 2 and 3 report the applications' topology and the total number of neurons for each application, respectively.

We proceed to map these SNN applications with a portion of neurons randomly set as faulty, with rates of 10%, 20%, 30%, and 40% per layer of the 3D NoC-based neuromorphic system. This mapping is performed across NoC sizes ranging from $3 \times 3 \times 3$ to $5 \times 5 \times 5$ as specified in Table 1. Unlike our previous work in [7] and [26], where a portion of neurons set as faulty pertains to the entire 3D NoC-based neuromorphic system at once, potentially resulting in an uneven distribution of faults across layers. The rationale behind this new fault insertion approach is to ensure that each layer of the 3D NoC-based neuromorphic system has an even fault rate distribution.

For the second phase of the evaluation, we assessed the proposed R-MaS3N's output neuron mapping efficiency and neuron utilization behavior across all fault rates considering NoC sizes from $3 \times 3 \times 3$ to $5 \times 5 \times 5$ of the 3D NoC-based neuromorphic system having 128 and 256 neurons per cluster as outlined in Table 1. Mapping efficiency in this context refers to the method's ability to efficiently remap neurons once a fault is detected in the system, which results in unmapped neurons of the SNN application.

We also evaluated R-MaS3N mapping cost as a function of execution time. The measurement of the mapping cost encompasses critical stages of the mapping method such as initial mapping, the setting of some neurons faulty, faulty neurons detection, and subsequent remapping of previously unmapped neurons. More importantly, we give a time complexity analysis of the neuron remapping stage in R-MaS3N and evaluate the remapping time under the fault rates described in Table 1. This enabled us to make a fair comparison with the remapping time of our previous fault-tolerant mapping schemes.

To describe the robustness of R-MaS3N when mapping with faults to a 3D NoC-based neuromorphic system, we performed a reliability analysis of mapping the proposed SNN applications described in Table 1. In particular, we evaluated Mean Time To Failure (MTTF) which represents the average time until failure occurs within the system under the fault rates outlined in Table 1.

TABLE 2. Applications used for evaluating R-MaS3N.

App.	Topology	Neurons	App. ID
MLP	784_1000_10	1,794	MLP_1794
	784_2000_10	2,794	MLP_2794
	784_2000_2000_10	4,794	MLP_4794
	784_4000_4000_10	8,794	MLP_8794
	784_3000_3000_3000_10	9,794	MLP_9794
	784_6000_6000_6000_10	18,794	MLP_18794

**All the applications are feedforward connections.*
**The total number of neurons for any topology is the sum of all neurons from the input layer to the last layer in the topology.*

The robust mapping method is developed in Python programming language and executed in a GPU-accelerated environment. Our 3D NoC-based neuromorphic system is written in Verilog HDL language but was translated to Python programming language for ease of evaluation. All our experiments were conducted on a high-performance system running Windows 10 with a Core i7 processor, 32-GB RAM, and NVIDIA GeForce-GTX GPU.

B. NEURON MAPPING EFFICIENCY

We evaluated all the SNN applications presented in Table 2 on a 3D NoC-based neuromorphic system having 128 and 256 neurons per cluster for NoC sizes $3 \times 3 \times 3$, $4 \times 4 \times 4$, and $5 \times 5 \times 5$ for all fault rates outlined in Table 1. As illustrated in Figures 10, 11, and 12, as the rate of designated faulty neurons increases, with successive initial mappings, the count of unmapped application neurons before repairs (*UA_BR*) also increases. This phenomenon persists as long as the fault rate continues to rise. However, after remapping to underutilized and most utilized neurons from a selected area of the 3D NoC-based neuromorphic system (*UN_NS*) in the fault-tolerant step of the R-MaS3N, the number of unmapped application neurons after repairs (*UA_AR*) becomes 0. This applies to all the proposed SNN applications in Table 2.

C. MAPPING COST

The mapping cost was assessed based on the mapping time of key stages in mapping the SNN applications outlined in Table 2 to a 3D NoC-based neuromorphic system for NoCs sizes from $3 \times 3 \times 3$ to $5 \times 5 \times 5$. The stages were initial mapping, setting some neurons as faulty, initial mapping again, and remapping the unmapped neurons. We quantify the computational effort involved in the robust mapping of neurons of SNN applications to clusters in the 3D NoC-based neuromorphic system. As described in Fig. 13, the mapping cost of our mapping method is less than 10 seconds when mapping a large-scale SNN (i.e., *MLP*_18794) to a $5 \times 5 \times 5$ NoC-based neuromorphic system having 40% neurons faulty.

[0]This footnote provides additional information about Fig. 10, Fig. 11, and Fig. 12:
* The data points on the left y-axis are associated with *UA_BR* and *UA_AR* while on the right y-axis with *UN_NS*.

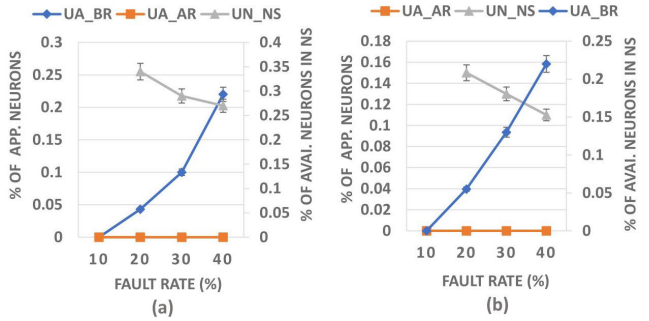


FIGURE 10. Output neuron remapping behavior over various fault rates of the mapping method on a 3 × 3 × 3 NoC-based neuromorphic system: (a) *MLP*_1794, (b) *MLP*_2794.

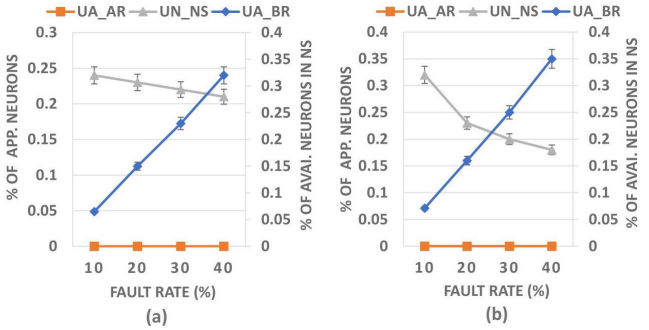


FIGURE 11. Output neuron remapping behavior over various fault rates of the mapping method on a 4 × 4 × 4 NoC-based neuromorphic system: (a) *MLP*_4794, (b) *MLP*_8794.

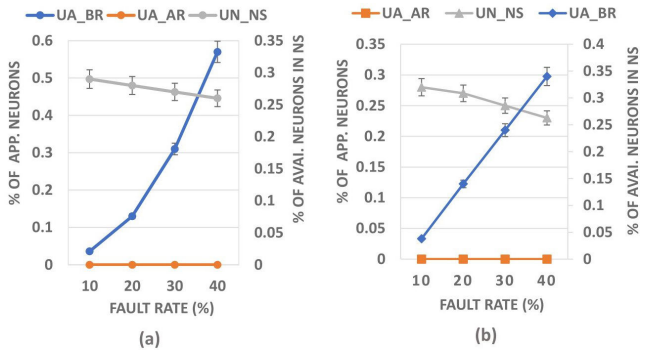


FIGURE 12. Output neuron remapping behavior over various fault rates of the mapping method on a 5 × 5 × 5 NoC-based neuromorphic system: (a) *MLP*_9794, (b) *MLP*_18794.

D. NEURON UTILIZATION BEHAVIOR

It is crucial to understand neuron utilization behavior in order to determine if a mapping strategy has restored neuron utilization during system processes. We evaluated neuron utilization patterns pre and post randomly setting of some portion of neurons as faulty for the 3D-NoC-based neuromorphic system for NoC sizes from $3\times3\times3$ to $5\times5\times5$. From Fig. 14, before some neurons are set to be faulty, the 3D NoC-based neuromorphic system achieves peak application neuron utilization. As the count of faulty neurons increase steadily, neuron utilization decreases. However, the

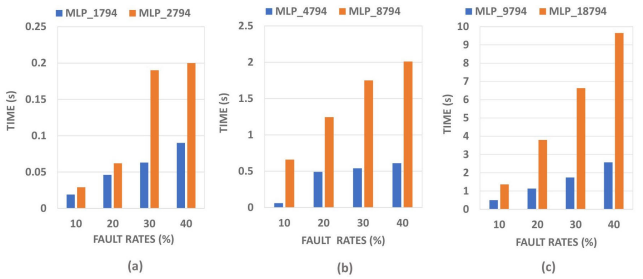


FIGURE 13. Plot of mapping cost of the mapping method for different NoC sizes of a 3D NoC-based neuromorphic system: (a) 3 × 3 × 3, (b) 4 × 4 × 4, (c) 5 × 5 × 5 under various fault rates.

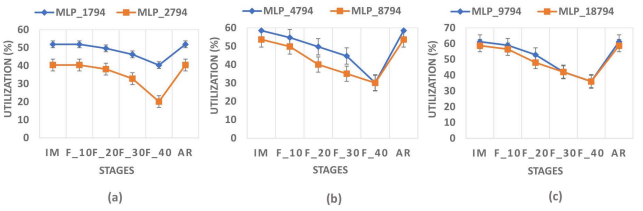


FIGURE 14. Plot of neuron utilization for all the stages of the mapping method for different NoC sizes of a 3D NoC-based neuromorphic system: (a) 3 × 3 × 3, (b) 4 × 4 × 4, (c) 5 × 5 × 5.

3D NoC-based neuromorphic system restores its total operational capacity after remapping.

E. RELIABILITY ANALYSIS

In order to develop fault-tolerant or resilient systems, reliability analysis is essential. Mean time to failure (MTTF) is at the core of this analysis. Despite its widespread use, the measure is also misused quite often. It has been misinterpreted as a guarantee of a minimum lifespan. MTTF, on the other hand, refers to the average time until the failure of a system [52]. Fig. 15 illustrates MTTFs of the R-MaS3N in a 3D NoC-based neuromorphic system with different NoC sizes.

Based on the assumption that failures occur constantly over time and are independent events, we derived the MTTF. Our mapping method has two failure rates; the initial mapping failure rate (IFR) and the remap failure rate (RFR). If the remapping fails, the failure rate is one, as given in Equation 18.

$$RFR = \begin{cases} 1, & \text{If } U > 2 \times M_{C_{less}} \\ 0 & \text{Otherwise} \end{cases} \tag{18}$$

where U represents the number of unmapped application neurons and M_C_{less} the number of mapped neurons of the 3D NoC-based neuromorphic system in the C_{less} region.

Since our mapping method is fault-tolerant, provided that the constraints mentioned earlier are observed during mapping. The MTTF for a k fault-tolerant system [7] with the mapping method is given in Equation 19:

$$MTTF_{RS} = \left(\frac{1}{\lambda \cdot \left(\frac{k}{p}\right)}\right) \cdot t \tag{19}$$

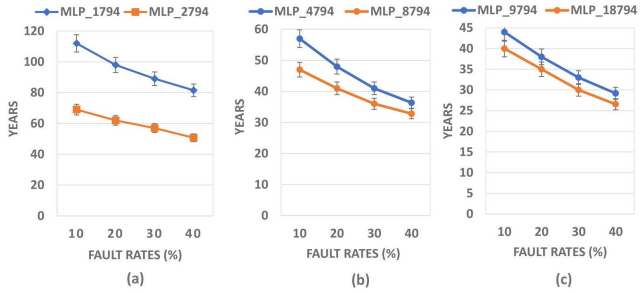


FIGURE 15. Mean time to failure of the mapping method for different NoC sizes of a 3D NoC-based neuromorphic system: (a) 3 × 3 × 3, (b) 4 × 4 × 4, (c) 5 × 5 × 5.

where λ is the initial mapping failure rate per neuron (in failures per hour), k is the number of neurons to be mapped, p is the number of available neurons to be used for the remapping, and t is the time unit. Unlike in [7] where the λ is set at a constant value, here λ is computed taking into account the specific characteristics and conditions of the system.

Fig. 15 shows the MTTF of R-MaS3N for mapping the proposed SNN applications in Table 2 to a 3D-NoC-based neuromorphic system for different NoC sizes. R-MaS3N has a minimum MTTF of 50.73 years, 35.36 years, and 26.54 years for 3 × 3 × 3, 4 × 4 × 4, and 5 × 5 × 5 NoC sizes with a fault rate of 40%.

F. TIME COMPLEXITY

The SLP in algorithm 1 takes N sets of unmapped layer neurons as input and categorizes them into C_{most} and C_{least} partitions. The algorithm operates in linear time. It iterates over $n_i \in N$ neurons and performs averaging and comparison to determine which partition to place a neuron. Due to this, the algorithm's time complexity is $O(N)$, where N is the number of neurons.

The NP in algorithm 2 partitions the M_n mapped neurons in the 3D NoC-based neuromorphic system. The NP algorithm is also linearly time-complex. A loop is iterated over $n_i \in M_n$. Each iteration involves a set of calculations to determine each n_i connectivity and activity pattern. The time complexity of this scenario is $O(M_n)$, where M_n is the number of mapped neurons. The partitioning process can efficiently handle a large neuron set since the time complexity of SLP and NP algorithms is linear.

The time complexity to remap unmapped N neurons layer to layer to existing neurons is $O(N^2)$. Overall execution time (E_t) for neuron remapping includes the time complexities of SLP and NP algorithms and layer-to-layer remapping. The mathematical expression is given in Equation 20.

$$E_{mt} = O(N) + O(M_n) + O(N^2) \tag{20}$$

Table 3 shows the remapping time evaluation of the remapping step of the R-MaS3N for the SNN applications proposed in Table 2 under the highest fault rate (i.e. 40%). For the 3D-NoC-based neuromorphic system with the largest NoC size (i.e., 5×5x5), remapping takes less than 10 seconds. As compared to the GA-based remapping method for the 3D-NoC-based neuromorphic system for the smallest NoC size (i.e., 4 × 4x4) in [7], the R-Mas3N remapping algorithm time represents a 71× significant reduction and is thus scalable.

TABLE 3. R-MaS3N remapping time in the 3D-NoC-based neuromorphic system for different SNN applications.

Network size	SNN Configuration	Time (s)
3 × 3 × 3	784_1000_10	0.06
	784_2000_10	0.18
4 × 4 × 4	784_2000_2000_10	0.58
	784_4000_4000_10	1.97
5 × 5 × 5	784_3000_3000_3000_10	2.54
	784_6000_6000_6000_10	9.60

The mapping time takes into account the time:
- To perform layer partitioning of unmapped neurons in the layer(s) of an SNN application.
- To perform neuron partitioning of neurons of the neuromorphic system.
- To perform neuron remapping for the highest fault rate.

VI. CONCLUSION AND FUTURE WORK

This work presented a mapping approach, R-MaS3N, that leverages the reuse of existing neurons for robust mapping of SNNs to a 3D-NoC-based neuromorphic system (NR-NASH). A heuristic-based partitioning technique is employed to partition neurons in the layers of an SNN application using neuron firing patterns. Moreover, a neuronal partitioning approach cluster mapped neurons in the layers of the neuromorphic neural circuits based on connectivity patterns and spiking activities.

From the evaluation results, we conclude that the proposed approach maintains a remapping efficiency of 100% in the 3D NoC-based neuromorphic system. With a NoC system configuration of 4 × 4 × 4 and 256 neurons per cluster, our approach has a remapping time of 71× less than the previous approach with the same configuration parameters. In addition, the MTTF of the mapping method of the 3D-NoC-based neuromorphic system (NR-NASH) with NoC size 5×5×5 at a 40% fault rate surpasses the previous method at 20% fault rate by 16% for 4 × 4 × 4 NoC size.

Thermal management strategies will be explored in the future, in addition to power and energy consumption evaluations. This is to enable the development of more resilient and efficient system architectures.

REFERENCES

[1] T. Titirsha, S. Song, A. Das, J. Krichmar, N. Dutt, N. Kandasamy, and F. Catthoor, "Endurance-aware mapping of spiking neural networks to neuromorphic hardware," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 2, pp. 288–301, Feb. 2022.

[2] K. Yamazaki, V.-K. Vo-Ho, D. Bulsara, and N. Le, "Spiking neural networks and their applications: A review," *Brain Sci.*, vol. 12, no. 7, p. 863, 2022. [Online]. Available: https://www.mdpi.com/2076-3425/12/7/863

[3] M. Pfeiffer and T. Pfeil, "Deep learning with spiking neurons: Opportunities and challenges," *Frontiers Neurosci.*, vol. 12, pp. 1–10, Oct. 2018. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2018.00774

[4] J. Kim, J. Koo, T. Kim, and J.-J. Kim, "Efficient synapse memory structure for reconfigurable digital neuromorphic hardware," *Frontiers Neurosci.*, vol. 12, pp. 1–6, Nov. 2018. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2018.00829

[5] H. C. V. Ngu and K. M. Lee, "Effective conversion of a convolutional neural network into a spiking neural network for image recognition tasks," *Appl. Sci.*, vol. 12, no. 11, p. 5749, Jun. 2022.

[6] A. Ben Abdallah and K. N. Dang, "Toward robust cognitive 3D brain-inspired cross-paradigm system," *Frontiers Neurosci.*, vol. 15, pp. 1–11, Jun. 2021.

[7] K. N. Dang, N. A. V. Doan, and A. B. Abdallah, "MigSpike: A migration based algorithms and architecture for scalable robust neuromorphic systems," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 2, pp. 602–617, Apr. 2022.

[8] M. Davies, A. Wild, G. Orchard, Y. Sandamirskaya, G. A. F. Guerra, P. Joshi, P. Plank, and S. R. Risbud, "Advancing neuromorphic computing with loihi: A survey of results and outlook," *Proc. IEEE*, vol. 109, no. 5, pp. 911–934, May 2021.

[9] S. Song, J. Hanamshet, A. Balaji, A. Das, J. L. Krichmar, N. D. Dutt, N. Kandasamy, and F. Catthoor, "Dynamic reliability management in neuromorphic computing," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 17, no. 4, pp. 1–27, Jul. 2021, doi: 10.1145/3462330.

[10] X. Wang, T.-M. Choi, X. Yue, M. Zhang, and W. Du, "An effective optimization algorithm for application mapping in network-on-chip designs," *IEEE Trans. Ind. Electron.*, vol. 67, no. 7, pp. 5798–5809, Jul. 2020.

[11] Y. Zhang, P. Qu, and W. Zheng, "Towards 'general purpose' brain-inspired computing system," *Tsinghua Sci. Technol.*, vol. 26, no. 5, pp. 664–673, Oct. 2021.

[12] C. Zhang, D. Zhang, and A. Stepanyants, "Noise in neurons and synapses enables reliable associative memory storage in local cortical circuits," *Eneuro*, vol. 8, no. 1, pp. 1–20, Jan. 2021.

[13] J. Timcheck, J. Kadmon, K. Boahen, and S. Ganguli, "Optimal noise level for coding with tightly balanced networks of spiking neurons in the presence of transmission delays," *PLOS Comput. Biol.*, vol. 18, no. 10, Oct. 2022, Art. no. e1010593.

[14] C. Xiao, Y. Wang, J. Chen, and L. Wang, "Topology-aware mapping of spiking neural network to neuromorphic processor," *Electronics*, vol. 11, no. 18, p. 2867, Sep. 2022. [Online]. Available: https://www.mdpi.com/2079-9292/11/18/2867

[15] A. Balaji, T. Marty, A. Das, and F. Catthoor, "Run-time mapping of spiking neural networks to neuromorphic hardware," *J. Signal Process. Syst.*, vol. 92, no. 11, pp. 1293–1302, Nov. 2020.

[16] A. Balaji, A. Das, Y. Wu, K. Huynh, F. G. Dell'Anna, G. Indiveri, J. L. Krichmar, N. D. Dutt, S. Schaafsma, and F. Catthoor, "Mapping spiking neural networks to neuromorphic hardware," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 1, pp. 76–86, Jan. 2020.

[17] C. Xiao, J. Chen, and L. Wang, "Optimal mapping of spiking neural network to neuromorphic hardware for edge-AI," *Sensors*, vol. 22, no. 19, p. 7248, Sep. 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/19/7248

[18] M. Henry, "The blue brain project," *Nature Rev. Neurosci.*, vol. 2, no. 7, pp. 153–159, 2006.

[19] S. B. Furber, D. R. Lester, L. A. Plana, J. D. Garside, E. Painkras, S. Temple, and A. D. Brown, "Overview of the SpiNNaker system architecture," *IEEE Trans. Comput.*, vol. 62, no. 12, pp. 2454–2467, Dec. 2013.

[20] C. Frenkel, J.-D. Legat, and D. Bol, "Morphic: A 65-nm 738k-synapse/mm^2 quad-core binary-weight digital neuromorphic processor with stochastic spike-driven online learning," *IEEE Trans. Biomed. Circuits Syst.*, vol. 13, no. 5, pp. 999–1010, Oct. 2019.

[21] 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*, vol. 38, no. 1, pp. 82–99, Jan. 2018.

[22] N. Qiao, H. Mostafa, F. Corradi, M. Osswald, F. Stefanini, D. Sumislawska, and G. Indiveri, "A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses," *Frontiers Neurosci.*, vol. 9, pp. 39–55, Apr. 2015.

[23] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G.-J. Nam, B. Taba, M. Beakes, B. Brezzo, J. B. Kuang, R. Manohar, W. P. Risk, B. Jackson, and D. S. Modha, "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.

[24] C. Torres-Huitzil and B. Girau, "Fault and error tolerance in neural networks: A review," *IEEE Access*, vol. 5, pp. 17322–17341, 2017.

[25] B. Yu, "Neuroscience: Fault tolerance in the brain," *Nature*, vol. 532, pp. 449–450, Apr. 2016.

[26] W. Y. Yerima, O. M. Ikechukwu, K. N. Dang, and A. B. Abdallah, "Fault-tolerant spiking neural network mapping algorithm and architecture to 3D-NoC-based neuromorphic systems," *IEEE Access*, vol. 11, pp. 52429–52443, 2023.

[27] K. Patel and C. Schuman, "Impact of noisy input on evolved spiking neural networks for neuromorphic systems," in *Proc. Annu. Neuro-Inspired Comput. Elements Conf.*, Apr. 2023, pp. 52–56.

[28] V. Duddu, D. V. Rao, and V. Balas, "Towards enhancing fault tolerance in neural networks," in *Proc. 7th EAI Int. Conf. Mobile Ubiquitous Syst., Comput., Netw. Services*, 2020, pp. 59–68.

[29] F. Zhao and Y. Zeng, "Dynamically optimizing network structure based on synaptic pruning in the brain," *Frontiers Syst. Neurosci.*, vol. 15, pp. 1–12, Jun. 2021.

[30] A. P. Johnson, J. Liu, A. G. Millard, S. Karim, A. M. Tyrrell, J. Harkin, J. Timmis, L. J. McDaid, and D. M. Halliday, "Homeostatic fault tolerance in spiking neural networks: A dynamic hardware perspective," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 2, pp. 687–699, Feb. 2018.

[31] J.-S. Seo, B. Brezzo, Y. Liu, B. D. Parker, S. K. Esser, R. K. Montoye, B. Rajendran, J. A. Tierno, L. Chang, D. S. Modha, and D. J. Friedman, "A 45 nm CMOS neuromorphic chip with a scalable architecture for learning in networks of spiking neurons," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Sep. 2011, pp. 1–4.

[32] M. Anderson, "Neural reuse: A fundamental organizational principle of the brain," *Behav. Brain Sci.*, vol. 33, pp. 245–266, Aug. 2010.

[33] R. Ptak, N. Doganci, and A. Bourgeois, "From action to cognition: Neural reuse, network theory and the emergence of higher cognitive functions," *Brain Sci.*, vol. 11, no. 12, p. 1652, Dec. 2021. [Online]. Available: https://www.mdpi.com/2076-3425/11/12/1652

[34] A. Ione, "After phrenology: Neural reuse and the interactive brain after phrenology: Neural reuse and the interactive brain by Michael L. Anderson. MIT Press, Cambridge, MA, U.S.A., 2014. 432 pp., illus. Trade. ISBN: 978-0-262-02810-3," *Leonardo*, vol. 49, pp. 89–91, Feb. 2016.

[35] M. L. Anderson, "Neural reuse in the organization and development of the brain," *Develop. Med. Child Neurol.*, vol. 58, pp. 3–6, Mar. 2016.

[36] L.-C. Chu and B. W. Wah, "Fault tolerant neural networks with hybrid redundancy," in *Proc. Int. Joint Conf. Neural Netw.*, vol. 2, Jun. 1990, pp. 639–649.

[37] T. Haruhiko, M. Masahiko, K. Hidehiko, and H. Terumine, "Enhancing both generalization and fault tolerance of multilayer neural networks," in *Proc. Int. Joint Conf. Neural Netw.*, Sep. 2007, pp. 1429–1433.

[38] W. Guo, H. E. Yantır, M. E. Fouda, A. M. Eltawil, and K. N. Salama, "Towards efficient neuromorphic hardware: Unsupervised adaptive neuron pruning," *Electronics*, vol. 9, no. 7, p. 1059, Jun. 2020. [Online]. Available: http://hdl.handle.net/10754/663934

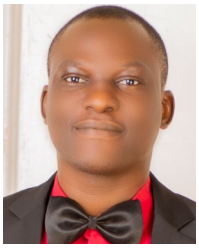
[39] G. Yuan, Z. Liao, X. Ma, Y. Cai, Z. Kong, X. Shen, J. Fu, Z. Li, C. Zhang, H. Peng, N. Liu, A. Ren, J. Wang, and Y. Wang, "Improving DNN fault tolerance using weight pruning and differential crossbar mapping for ReRAM-based edge AI," in *Proc. 22nd Int. Symp. Quality Electron. Design (ISQED)*, Apr. 2021, pp. 135–141.

[40] M. Rastogi, S. Lu, N. Islam, and A. Sengupta, "On the self-repair role of astrocytes in STDP enabled unsupervised SNNs," *Frontiers Neurosci.*, vol. 14, Jan. 2021, Art. no. 603796.

[41] K. Rhazali, B. Lussier, W. Schön, and S. Geronimi, "Fault tolerant deep neural networks for detection of unrecognizable situations," *IFAC-PapersOnLine*, vol. 51, no. 24, pp. 31–37, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S240589631832216X

[42] A. Das, Y. Wu, K. Huynh, F. Dell'Anna, F. Catthoor, and S. Schaafsma, "Mapping of local and global synapses on spiking neuromorphic hardware," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2018, pp. 1217–1222.

[43] O. Jin, Q. Xing, Y. Li, S. Deng, S. He, and G. Pan, "Mapping very large scale spiking neuron network to neuromorphic hardware," in *Proc. 28th ACM Int. Conf. Architectural Support Program. Lang. Operating Syst.*, 2023, pp. 419–432.
[44] C.-K. Lin, A. Wild, G. N. Chinya, T.-H. Lin, M. Davies, and H. Wang, "Mapping spiking neural networks onto a manycore neuromorphic architecture," in *Proc. 39th ACM SIGPLAN Conf. Program. Lang. Design Implement.*, Jun. 2018, pp. 78–89, doi: 10.1145/3192366.3192371.
[45] L. Wang, P. Lv, L. Liu, J. Han, H.-F. Leung, X. Wang, S. Yin, S. Wei, and T. Mak, "A lifetime reliability-constrained runtime mapping for throughput optimization in many-core systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 9, pp. 1771–1784, Sep. 2019.
[46] Q. Xu, S. Chen, H. Geng, B. Yuan, B. Yu, F. Wu, and Z. Huang, "Fault tolerance in memristive crossbar-based neuromorphic computing systems," *Integration*, vol. 70, pp. 70–79, Jan. 2020.
[47] A. Gebregirogis and M. Tahoori, "Approximate learning and fault-tolerant mapping for energy-efficient neuromorphic systems," *ACM Trans. Design Autom. Electron. Syst.*, vol. 26, no. 3, pp. 1–23, May 2021, doi: 10.1145/3436491.
[48] O. M. Ikechukwu, K. N. Dang, and A. B. Abdallah, "On the design of a fault-tolerant scalable three dimensional NoC-based digital neuromorphic system with on-chip learning," *IEEE Access*, vol. 9, pp. 64331–64345, 2021.
[49] T. H. Vu, O. M. Ikechukwu, and A. B. Abdallah, "Fault-tolerant spike routing algorithm and architecture for three dimensional NoC-based neuromorphic systems," *IEEE Access*, vol. 7, pp. 90436–90452, 2019.
[50] M. Ogbodo, T. Vu, K. Dang, and A. Abdallah, "Light-weight spiking neuron processing core for large-scale 3D-NoC based spiking neural network processing systems," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, Feb. 2020, pp. 133–139.
[51] O. Köpüklü, M. Babaee, S. Hörmann, and G. Rigoll, "Convolutional neural networks with layer reuse," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2019, pp. 345–349.
[52] H. Pham, *System Reliability Concepts*. London, U.K.: Springer, 2006, pp. 9–75, doi: 10.1007/1-84628-295-0_2.



WILLIAMS YOHANNA YERIMA (Student Member, IEEE) received the bachelor's degree in telecoms control and computer from Ahmadu Bello University Zaria, Nigeria, in 2016, and the master's degree in computer science from the African University of Science and Technology, Abuja, Nigeria, in 2019. He is currently pursuing the Ph.D. degree with the University of Aizu. His research reflects a diverse background with a focus on machine learning, fault-tolerant systems, and neuromorphic computing. His current research interests revolve around exploring the intersections of these fields, particularly in the domains of machine learning applied to fault-tolerant systems and neuromorphic computing for neuroprosthetics.

KHANH N. DANG (Member, IEEE) received the M.Sc. degree from Paris-Sud University, France, in 2014, and the Ph.D. degree from the University of Aizu, in 2017. He is an Associate Professor with the Division of Computer Engineering, Graduate School of Computer Science and Engineering, University of Aizu. Before joining the faculty at the University of Aizu, he was an Assistant Professor with Vietnam National University, Hanoi, Vietnam. His research interests encompass VLSI design, thermal awareness, fault tolerance, machine learning, and neuromorphic computing.

ABDERAZEK BEN ABDALLAH (Senior Member, IEEE) received the Ph.D. degree in computer engineering from the University of Electro-Communications, Tokyo, in 2002. From April 2014 to March 2022, he was the Head of the Computer Engineering Division with the University of Aizu, Japan. Since April 2022, he has been the Dean of the School of Computer Science and Engineering with the University of Aizu. He is a Full Professor with the University of Aizu. He is the author of four books, 12 patents, and more than 150 publications in peer-reviewed journal articles and conference papers. His research interests include adaptive/self-organizing systems, brain-inspired computing, interconnection networks, and AI-powered cyber-physical systems. He is a Senior Member of ACM.