Received May 13, 2019, accepted June 13, 2019, date of publication June 26, 2019, date of current version July 24, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2925085

Fault-Tolerant Spike Routing Algorithm and Architecture for Three Dimensional NoC-Based Neuromorphic Systems

THE H. VU, (Student Member, IEEE), OGBODO MARK IKECHUKWU, (Student Member, IEEE), AND ABDERAZEK BEN ABDALLAH, (Senior Member, IEEE)
Adaptive Systems Laboratory, Graduate School of Computer Science and Engineering, The University of Aizu, Aizuwakamatsu 965-8580, Japan

Corresponding authors: The H. Vu (d8182106@u-aizu.ac.jp) and Abderazek Ben Abdallah (benab@u-aizu.ac.jp)

This work was supported in part by the Competitive Research Funding (CRF), The University of Aizu, reference P-2-2018, and in part by the VLSI Design and Education Center (VDEC), The University of Tokyo, Japan, in Collaboration with Synopsys, Inc., and Cadence Design Systems, Inc.

ABSTRACT Neuromorphic computing systems are an emerging field that takes its inspiration from the biological neural architectures and computations inside the mammalian nervous system. The spiking neural networks (SNNs) mimic real biological neural networks by conveying information through the communication of short pulses between neurons. Since each neuron in these networks is connected to thousands of others, high bandwidth is required. Moreover, since the spike times are used to encode information in SNN, very low communication latency is also necessary. On the other hand, the combination of Two-dimensional Networks-on-Chip (2D-NoC) and Three-dimensional Integrated Circuits (3D-ICs) can provide a scalable interconnection fabric in large-scale parallel SNN systems. Although the SNNs have some intrinsic fault-tolerance properties, they are still susceptible to a significant amount of faults; especially, when we talk about integrating the large-scale SNN models in hardware. Consequently, the need for efficient solutions capable of avoiding any malfunctions or inaccuracies, as well as early fault-tolerance assessment, is becoming increasingly necessary for the design of future large-scale reliable neuromorphic systems. This paper first presents an analytical model to assess the effect of faulty connections on the performance of a 3D-NoC-based spiking neural network under different neural network topologies. Second, we present a fault-tolerant shortest-path k-means-based multicast routing algorithm (FTSP-KMCR) and architecture for spike routing in 3D-NoC of spiking neurons (3DFT-SNN). Evaluation results show that the proposed SP-KMCR algorithm reduces the average latency by 12.2% when compared to the previously proposed algorithm. In addition, the proposed fault-tolerant methodology enables the system to sustain correct traffic communication with a fault rate up to 20%, while only suffering 16.23% longer latency and 5.49% extra area cost when compared to the baseline architecture.

INDEX TERMS Spiking neural networks, performance assessment, fault-tolerant, k-means based multicast routing, scalable architecture.

I. INTRODUCTION

Brain-inspired computing or neuromorphic computing, is a biologically inspired approach, created from highly connected neurons to not only model neuroscience theories but also solve machine learning problems. The term neuromorphic was first introduced by Carver Mead in 1990 [1], where it referred to very large scale integration (VLSI) with analog components to mimic biological neural systems. Such systems can be categorized as non-spiking and spiking approaches.

Spiking neural networks (SNNs) attempt to mimic the information processing in the mammalian brain based on parallel arrays of neurons which communicate via spike events. Unlike the typical multi-layer perceptron networks where neurons fire at each propagation cycle, the neurons in the

The associate editor coordinating the review of this manuscript and approving it for publication was Mostafa Rahimi Azghadi.

SNN model fire only when a membrane potential reaches a specific value. In SNNs, information is encoded using various encoding schemes, such as coincidence coding, rate coding or temporal coding [2]. SNNs typically employ integrate-and-fire neurons model [3], [4] in which a neuron generates voltage spikes (roughly 1ms in duration per spike) that can travel down nerve fibers if it receives enough stimuli from other neurons with the presence of external stimuli. These pulses may vary in amplitude, shape, and duration; but, they are generally treated as identical events. The Hodgkin-Huxley [5] conductance-based neuron is often used to efficiently model the non-linear and stochastic dynamics of the ion channel in a biological neuron. However, the Hodgkin-Huxley model is too complicated to be used for a large-scale simulation or hardware implementation.

Recently, a number of deep SNNs have been proposed [6]. These networks show success in different pattern recognition tasks [7], [8]. However, although these models are known as multi-layer, they do not have many trainable layers when compared to traditional deep neural networks. This is due to the lack of an efficient learning rule to directly train deep spiking network. On the other hand, large-scale SNNs are used to simulate the complex activity of the brain. For example, a 2.5-million-neuron model, named Spaun, is presented in [9] (see Fig. 1 (a, b)). Spaun captured many aspects of neuroanatomy, neurophysiology, psychological behavior, and also performed well at digit recognition task. In a deep SNN, the communication between neurons plays an integral part in their implementation. By mapping a number of neurons into a planar structure and stacking the resulting planar die on top of one another with Through-Silicon Vias (TSVs), as Fig. 1 (c), communication latency can be greatly reduced.

The software simulation of SNNs is a flexible method for investigating the behavior of neuronal systems. However, the simulation of a large (deep) SNN system in software is slow. An alternative approach is a hardware implementation which provides the possibility to generate independent spikes accurately and simultaneously output spikes in real time. Hardware implementations also have the advantage of computational speed-up over software simulations and can take full advantage of their inherent parallelism. Specialized hardware architectures with multiple neuro-cores could exploit the parallelism inherent within neural networks to provide high processing speeds with low-power, which make SNNs suitable for embedded neuromorphic devices and control applications.

A. BACKGROUND AND MOTIVATION

In an efficient SNN with a proper neuron and network models, the arrival time of a synaptic input, such as a Dirac delta function or a shaped post-synaptic potential (EPSP/IPSP), to a neuron has a significant effect on the time of the output (i.e., a spike) of the given neuron (referred as the output time). As a result, any timing violations affect the proper functioning (firing) of the spiking neurons, as shown in Fig. 2, and also the on-chip learning capability of the whole system.

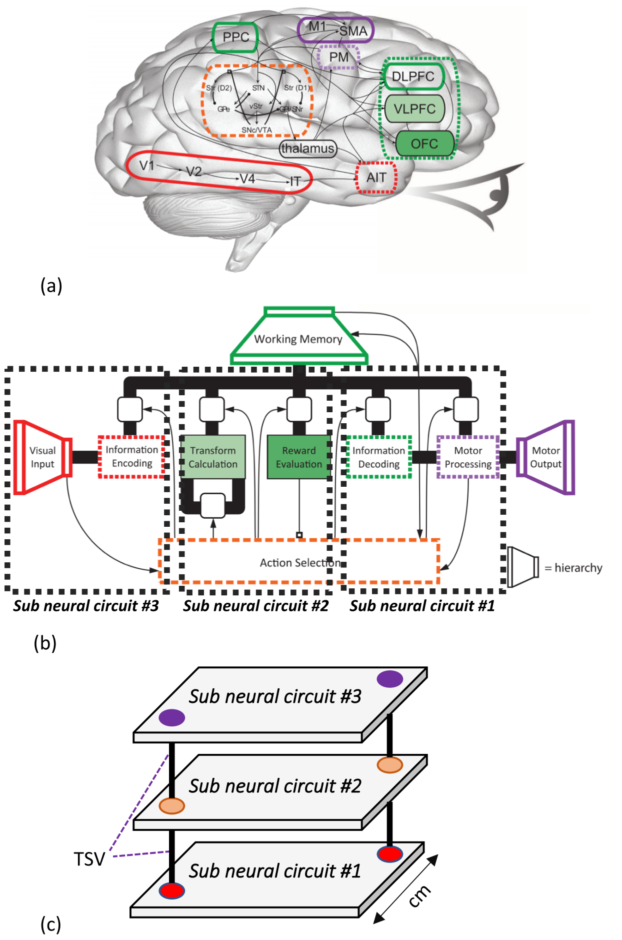


FIGURE 1. Large-scale 3D neuromorphic architecture example. This figure is inspired by the anatomical and functional architecture of Spaun [9]. (a) The anatomical architecture of Spaun indicates major brain structures and their connectivity. (b) The architecture of Spaun, where thick black lines illustrate communication between elements of the cortex while thin lines show connectivity between Basal Ganglia and the cortex. (c) 3D neuromorphic architecture using through silicon vias (TSVs) as vertical connections between layers that can deal with the high connectivity challenge of Spaun architecture when it is partitioned (black dot boxes) and mapped the 3D structure.

A shared bus as a communication medium is a poor choice for implementing a large-scale complex SNN chip/system with multicast routing because adding neurons decreases the communication capacity of the chip and may affect the neurons' firing rate due to the increasing length of the shared bus. Moreover, the non-linear increase in neural connectivity is too significant to be directly implemented using a dedicated point-to-point communication scheme.

Two-dimensional packet-switched Network-on-Chip (2D-NoC) [10] has been considered as a potential solution to deal with the interconnection problems found in previously proposed shared communication medium based SNNs [11], [12]. However, such interconnect strategies make it difficult to achieve high scalability with low power consumption, especially in large-scale SNN chips. Apart from the packet-switching scheme, which we considered in this work, the circuit-switching approach which has, compared to packet-switching, guaranteed throughput, lower hardware

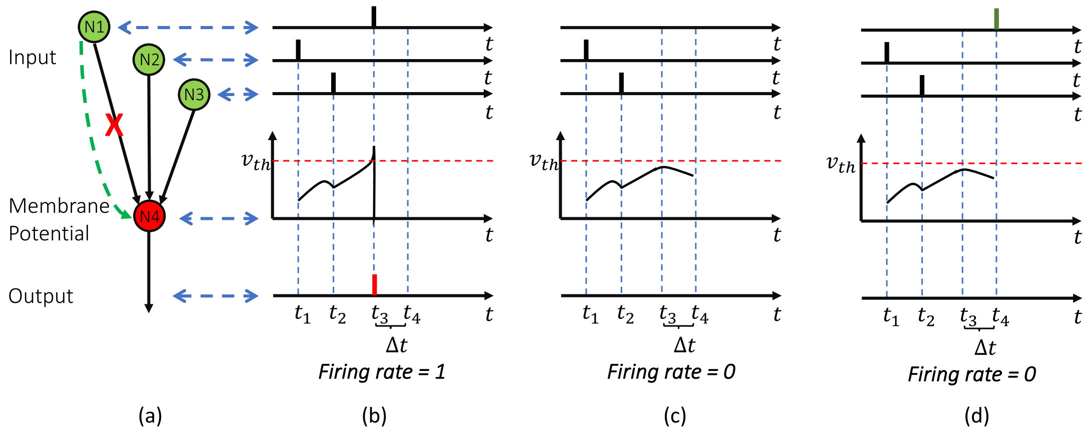


FIGURE 2. example of the connection-fault effect on the firing rate: (a) a postsynaptic neuron (N4) receiving incoming spikes from three pre-synaptic neurons, (b) with no connection fault, the firing rate = 1, (c) with the N1-to-N4 connection fault, the firing rate = 0, (d) long latency of a connection with an inefficient fault-tolerant routing algorithm resulting in the firing rate = 0.

complexity, and higher energy efficiency may also be used to explore the performance of different NoCs. However, the circuit-switching approach has a longer setup time.

In the past few years, the benefits of 3D-ICs and mesh-based NoCs have been fused into a promising architecture opening a new horizon for IC design, especially in AI-powered chips. The parallelism of NoCs can be enhanced in the third dimension thanks to the short wire length, and low power consumption of the interconnects of 3D-ICs. As a result, the 3D-NoC paradigm is considered to be one of the most advanced and auspicious architectures for the future of IC design. 3D-NoCs are capable of providing extremely high bandwidth, and low power interconnects [13] to satisfy the high requirements of emerging artificial intelligence (AI) applications. When combining 3D-NoC and SNN, a spiking neuron can be considered as a PE (neuro-core). The inter-neuron connectivities are implemented in the form of transmitting spike packets/flits via the scalable interconnection network. In this context, the PEs refer to the Spiking Neuron Processing Cores (SNPCs) attached to the 3D-NoC routers, the NoC channels are analogous to the synapses of the neurons, and the NoC topology refers to the way the neurons are interconnected within the network.

One of the main problems of hardware implementations for SNNs is their reliability potential. Although it has been claimed that SNNs have some intrinsic fault-tolerance properties thanks to their massive and parallel structures inspired by the biological neural models, it is not always the case when it comes to practical cases [14]. In fact, with the challenges inherited from the continuing shrinkage of semiconductor components, the implementation of SNNs in hardware exposes them to a variety of faults [14]. The fault risk becomes even more important as we move towards integrating large-scale SNNs for embedded systems when the yield becomes a major problem [15]. When considering the inter-neuron communication reliability, faults may affect the system performance, especially when they occur in critical applications (e.g., aerospace, autonomous car, biomedical, etc.). Such failures can result in undesirable inaccuracies or even irreversible severe consequences. In SNNs, when faults occur in the inter-neuron connections, the postsynaptic neurons become silent or near silent (low firing activity). As shown in Fig. 2 (c), at the presence of a broken link in the N1-to-N4 connection, the membrane potential of N4 fails to reach the threshold that would allow it to fire an output spike, as it is the case in Fig. 2 (b). This leads to a reduction in the firing rate of the post-synaptic neuron. Consequently, it may have impact on the overall performance of SNN models based on the rate coding method [16]. Neurons with low firing rates become more susceptible to noisy firing rates and temporal jitter of spikes resulting in an increase of the variance [17]. As a result, it demands efficient fault-tolerant techniques. In such mechanisms, the recovery time is one of the important requirements. The long latency of a fault-tolerant routing method may influence the firing rate, as shown in Fig. 2 (d). It may impact especially, SNN models using a temporal coding method that is based on the relative timing between spikes. Therefore, the challenge to find efficient fault-tolerant solutions is becoming more important with the integration of large SNNs onto silicon.

Routing algorithms are considered as one of the most efficient recovery mechanisms in SNNs as they play a vital role in neuron communication performance. A routing algorithm can influence the load balance across the network and the overall latency of a system [10] in fault-free scenarios. Since the traffic pattern in a given SNN is in a one-to-many fashion, where a pre-synaptic neuron sends spikes to a subset of post-synaptic ones, the use of conventional unicast-based routing in large-scale SNNs is inefficient [18]. In addition, when considering fault-tolerance requirements, the routing algorithm should be carefully chosen to minimize the inter-neuron communication latency; otherwise, the postsynaptic node accuracy can be compromised despite the fact that failure has been worked around. Fig. 2 (d) illustrates a clear example of such a case. In this figure, we can see that the long latency due to an inappropriate routing can prevent the postsynaptic neuron from timely firing output spike.

B. CONTRIBUTIONS

The main contributions of this work are summarized as follows:

- A comprehensive study providing a mathematical model to analyze the effects of link faults on the performance of 3D-NoC of spiking neurons architecture with different neural network topologies and communication methods. The goal is to provide a performance assessment that helps designers easily understand and evaluate the effect of faults on the system performance before the actual hardware development of the SNN system.
- A low-latency multicast routing algorithm for spike routing in 3D-NoC of spiking neurons architecture. The proposed algorithm, named SP-KMCR, is an improvement over our previously proposed KMCR algorithm [19]. It can eliminate a potential traffic congestion in the network due to the spike concentration in the centroid. Instead, SP-KMCR alleviates this congestion by allowing source nodes to send their spike packets through the shortest path.
- A fault-tolerant routing mechanism to deal with link faults in the 3DFT-SNN system. It composes of pre-defined primary and backup branches to route spike packets in both cases of with and without link faults. Furthermore, a fault management algorithm is also proposed to deal with the presence of faults.
- Architecture, hardware design and evaluation of the proposed 3DFT-SNN system. The proposed 3DFT-SNN performance and reliability are validated based on an RTL-level implementation, while the area/power analysis is performed using 45-nm CMOS technology.

The remainder of this paper is organized as follows. In Section II, we present some of the prior works related to the proposed research. Section III presents the analytic model for the 3D-NoC of spiking neurons architecture. Section IV describes the main hardware components of the proposed architecture. We dedicate Section V for the proposed routing algorithm. Experimental results are shown in Section VI. We discuss the challenges that should be addressed in the proposed architecture in Section VII before we end this paper with concluding remarks in Section VIII.

II. PRIOR WORKS

This section surveys related works on SNN implementations, focusing mainly on those that are based on packet-switched neurons communication (synaptic connections) and those related to fault tolerance in neural networks. These surveyed works have shown a functional expression of scalability for large-scale SNN hardware development. Research in this field follows two approaches; software simulation approach and full-custom hardware design approach. An example of the software-based simulation approach is the *Blue Brain* project [20]. The Blue Brain system can simulate up to 108 simple neurons or up to 104 very complex neurons as well as local and global synaptic plasticity rules defined for each neuron. The simulation environment is supported on the IBM Blue Gene/Q, a supercomputer with 65,536 cores. The power consumption of this approach is in the order of hundreds of kilowatts, which is costly. the system also is somewhat slow (i.e., low level of parallelism), and this hinders it from achieving biologic real-time execution on large-scale networks. As an alternative, full-custom hardware implementations were proposed to best the challenges of the software simulation stated above. The full-custom hardware implementations are based on combined hierarchical buses, point-to-point, or NoC interconnects, to support the routing of spikes in SNN systems. The rest of this section surveys the principal proposed works. In Fig. 3, we present a summary of techniques of SNN routing on different interconnect platforms.

A. SPIKE ROUTING HIERARCHICAL BUS

The approaches proposed in [22] and [23] are low-cost shared-bus based SNN architectures that support multicast and broadcast routing, but they have a drawback in scalability. The architectures of the works proposed in [29] and [30], boosted throughput; but, were constrained to small-size neural networks.

B. SPIKE ROUTING 2D NETWORK-ON-CHIP

Several 2D-NoC interconnect SNN research projects are currently ongoing [11], [12], [18], [26], [31]. Hereafter, we focus mainly on surveying some familiar projects. The Neurogrid project [32] uses analog computation to emulate ion-channel activity and a digital communication scheme to support synaptic connections. The Neurogrid has 16 neuro-cores and each can hold a total of 256×256 quadratic integrate-and-fire neuron models. For communication between close neuro-cores, it uses an external FPGA and bank of SRAMs. The Neurogrid has a constraint on the number of neurons it can accommodate on a layer (up to 2,175 neurons), and this prevents it from offering biological real-time performance [32].

The H-NoC (Hierarchical NoC) [12] is based on the EMBRACE neuro cell which provides the leaky integrate & fire neuron model with dynamic synapses, a packet decoder/encoder, and a network interface to communicate with digital NoC routers. It is organized into three layers: module, tile, and cluster. At the base, each module router can connect ten neural cells, each as a main neural computation element can support multiple neurons. In the same manner, ten module routers are connected to a tile router. An attractive work in [33] proposed a combination of hierarchical architecture and mesh routing strategies. The architecture consists of multiple levels of routers. Besides, there are many works [34], [35] that adopt the address event representation (AER) communication protocol. This protocol is suitable

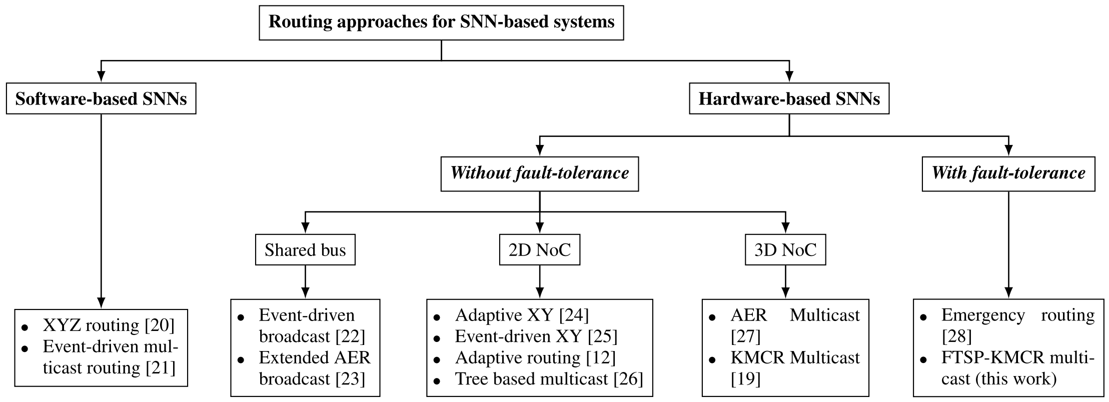


FIGURE 3. Summary of spiking neural network architectural routing methods on various interconnect platforms.

for implementing event-based SNNs. While [34] proposed a hierarchical address event architecture, [35] presented modularized AER architecture with source and destination routing approaches.

FACETS [36] uses a mixed-signal and high-density hardware neural network architecture based on a combination of analog neurons and a digital multilayer bus communication scheme To support the neurons interconnection, this architecture uses a combination of hierarchical buses for handling neuron communication inside the wafer, and off-wafer routers implemented on an FPGA based on a 2D-torus topology. *FACETS* can offer hardware acceleration with up to 10 μs inter-spike interval per wafer. However, the architecture consumes a large amount of power estimated to 1kW per wafer [36].

SpiNNaker [21] is another project that proposes a fully digital multiprocessor architecture for executing spiking neural network applications. The interconnection between each node is handled by a NoC using six links, which is wrapped into a triangular lattice; this lattice is then folded onto a surface of a toroid. A node incorporates 18 ARM968 processor cores (each of them emulating up to 1,000 neurons) and two NoC routers, one for handling the communication between the microprocessors and the peripherals, and the second for controlling the communications between processors and neighbor nodes. Each node can offer up to 16,000 Izhikevich neuron models, having a power consumption budget of 1W.

Another work, named *ClosNN*, is presented in [11]. The *ClosNN* system uses a customized NoC architecture based on Clos topology for the neural network. It is designed to overcome with a high diameter of mesh and low bisection bandwidth of a hierarchical tree. The architecture suffers from wire/router physical limitations. On the other hand, a ring topology interconnect architecture was proposed in [37] to achieve fixed latency.

C. SPIKE ROUTING 3D NETWORK-ON-CHIP

The work in [38] investigated the architecture and design of a 3D stacked neuromorphic accelerator. The architecture targeted processing applications on a CMOS vision sensor next to the first neural network layer. The authors claimed that only modest adaptations would be required to use the system for other applications. The 3D stacking architecture used face-to-face bonding of two 20cm wafers using microbumps.

A recent work was presented in [27] about a real-time digital neuromorphic system for the simulation of large-scale conductance-based SNNs. The architecture was implemented in six Altera Stratix III FPGA boards to simulate one million neurons [27]. An *AER* multicast routing mechanism was used for inter-neuron communications. Although the NoC architecture meets the requirements of the system, it is hardly deployed in embedded neuromorphic systems [39].

Apart from the works mentioned above, routing methods for NoC-based SNNs need to be taken into consideration. This is because the spike routing method affects the load balance across the network and also the spike latency, as previously stated in Section I. In general, these works can be classified as unicast-based [40], path-based [41], and tree-based [42]. A comparison between these methods is presented in [41].

D. FAULT-TOLERANCE FOR NEURAL NETWORK ARCHITECTURES

There have been many works proposed to solve the faults occurrence in hardware implementations of neural networks. In [14], authors surveyed fault tolerance in neural networks. Naeem *et al.* [43] presented a new learning rule mimicking self-repair capability of the brain, in which the learning rule could re-establish the firing rate of neurons when synaptic faults occur. Another self-repairing hardware architecture was proposed in [44]. This architecture features self-detect and self-repair synaptic faults, and maintains the system performance with a fault rate of 40%. However, the experiment was taken with only two neurons, and the architecture may suffer a scalability limitation due to its area overhead. In SpiNNaker [28], an emergency routing was proposed to deal with congested or broken links in a 2D-NoC torus topology. The algorithm is based on redundancy in the NoC architecture to automatically redirect a blocked packet through adjacent links to its destination. This enables the system to avoid the timing violations of SNNs when congestion or faults occur.

III. COMPREHENSIVE ANALYTIC PERFORMANCE ASSESSMENT

This section analyzes the effect of connection faults on the performance of 3D mesh interconnect architecture over different spiking neural network topologies and various spike routing methods. Our assessment was inspired by [18], [45], in which various interconnect architectures were analyzed under different SNN traffics. Regarding SNN topologies, our assessment is also taken under the Hopfield neural network (HF) and Randomly Connected neural network (RNDC). In HF [46], each neuron sends its spikes to all the others, presenting upper bound of the neural network connectivity. However, RNDC [47] simply presents neural network models which its connection probability exponentially decreases with the distance between neurons. We adopt the property of exponentially decaying connection probability from the neuro-biological data behaviour reported in [48], which suggests that the probability of the connection between two neurons decreases exponentially with the distance between them. Furthermore, in each neural network topology, our assessment is performed under different spike routing methods including unicast (UC), multicast (MC), and broadcast (BC). The goal of this analytical model is to help designers early understand the effect of faulty links on the performance of their architecture over neural network topologies and spike routing schemes.

A. ASSUMPTION AND NETWORK MODEL

We analyze the performance of a $\sqrt[3]{n} \times \sqrt[3]{n} \times \sqrt[3]{n}$ 3D mesh architecture composing of n spiking neural processing cores (SNPCs) and n spike routers. We assume that each SNPC has s spiking neurons, and the link fault rate α has a uniform distribution. Furthermore, the "O" notation is used to compare the analysis results between the different spike routing methods. With the link fault rate α, the total number of functional links in the architecture is given by:

$$TL = 3(1-\alpha)\sqrt[3]{n^2}(\sqrt[3]{n}-1). \tag{1}$$

As in [49], the mean distance between two nodes in 3D-mesh can be determined by:

$$\overline{Dist} = \frac{\sqrt[3]{n^2}-1}{\sqrt[3]{n}}. \tag{2}$$

B. PERFORMANCE ANALYSIS OF HOPFIELD NEURAL NETWORK

1) UNICAST-BASED SPIKE ROUTING

In unicast-based spike routing, in order to send a spike to all the others in HF, source neuron needs to send *n-1* packets. Hence, the total number of hops for a spike is given by (3):

$$\begin{aligned} TotalDist_{UC}^{HF} &= (n-1).\overline{Dist} \\ &= \frac{(n-1)(\sqrt[3]{n^2}-1)}{\sqrt[3]{n}} \\ &\approx n\sqrt[3]{n}. \end{aligned} \tag{3}$$

From (3), the effective bandwidth of the system is determined by (4):

$$\begin{aligned} BW_{eff,UC}^{HF} &= \frac{\overline{w}.TL}{TotalDist_{UC}^{HF}}.f_{NoC}.U_{NoC} \\ &= \frac{3\overline{w}(1-\alpha)(\sqrt[3]{n}-1)}{\sqrt[3]{n^2}}.f_{NoC}.U_{NoC} \\ &= O\left(\frac{1}{\sqrt[3]{n}}\right), \end{aligned} \tag{4}$$

where $\overline{w}$ is the wire number per connection, f_{NoC} is the connection frequency, U_{NoC} is the connection utilization factor for a 3D-mesh architecture. With n SNPCs, the average spiking rate of a single SNPC is presented as (5):

$$\begin{aligned} f_{p,out,UC}^{HF} &= \frac{BW_{eff,UC}^{HF}}{n} \\ &= \frac{3\overline{w}(1-\alpha)(\sqrt[3]{n}-1)}{n\sqrt[3]{n^2}}.f_{NoC}.U_{NoC} \end{aligned} \tag{5}$$

Besides, the maximal firing frequency for the architecture under unicast-based spike routing:

$$f_{spike,max,UC}^{HF} = \frac{s}{T_{refractory}} \cong \frac{s}{10n.T_{cycle}} = \frac{s.f_{NoC}}{10n}, \tag{6}$$

where s is the number of neurons in a SNPC, $T_{refractory}$ is the period the neuron cannot fire again. T_{cycle} is the link delay ($T_{cycle} = 1/f_{NoC}$). As mentioned above, in order to send a spike, the source node needs to send $n-1$ packets; thus, it takes $n.T_{cycle}$. Here, we are not including the router delay as it is a constant, independent of the network size. In our analysis, we assume $T_{refractory} \cong 10n.T_{cycle}$, similarly to [45].

By dividing (5) by (6), we can determine whether s neurons can fire at the maximal rate or not. This is represented by K, as given in (7):

$$\begin{aligned} K_{UC}^{HF} &= \frac{f_{p,out,UC}^{HF}}{f_{spike,max,UC}^{HF}} = \frac{30\overline{w}(1-\alpha)(\sqrt[3]{n}-1)}{s\sqrt[3]{n^2}}.U_{NoC} \\ &= O\left(\frac{1}{\sqrt[3]{n}}\right). \end{aligned} \tag{7}$$

In equation (7), $K > 1$ means that the architecture can deliver all spikes injected by s given neurons in each SNPC.

2) MULTICAST AND BROADCAST BASED ROUTING SCHEMES

In both MC and BC, the source node needs to send only one packet for each spike. Therefore, the hop count is determined by (8):

$$TotalDist_{MC/BC}^{HF} \cong n. \tag{8}$$

From (8), the efficient bandwidth, the average spiking rate, the maximal firing frequency, and the K metric for MC and BC are expressed by the equations (9), (10), (11), (12), respectively:

$$BW^{HF}_{eff,MC/BC} = \frac{3\overline{w}(1-\alpha)(\sqrt[3]{n}-1)}{\sqrt[3]{n}}.f_{NoC}.U_{NoC} = O(1) \quad (9)$$

$$f^{HF}_{p,out,MC/BC} = \frac{BW^{HF}{eff,MC/BC}}{n} = \frac{3\overline{w}(1-\alpha)(\sqrt[3]{n}-1)}{n\sqrt[3]{n}}.f_{NoC}.U_{NoC} \quad (10)$$

$$f^{HF}_{spike,max,MC/BC} = \frac{s}{T_{refractory}} \cong \frac{s}{T_{cycle}\overline{Dist}} = \frac{s\sqrt[3]{n}}{\sqrt[3]{n^2}-1}.f_{NoC} \quad (11)$$

$$K^{HF}_{MC/BC} = \frac{f^{HF}_{p,out,MC/BC}}{f^{HF}_{spike,max,MC/BC}} = O\left(\frac{1}{\sqrt[3]{n^2}}\right). \quad (12)$$

In summary, equations (4), (7), (9), and (12) demonstrate the effect of fault rate on the architecture performance (i.e., in terms of efficient bandwidth and spike rate which a given architecture can maintain) when run on Hopfield neural network. Compared to UC, MC and BC offer higher bandwidth and the number of neurons can fire at the maximum rate.

C. PERFORMANCE ANALYSIS OF RNDC NEURAL NETWORK

As stated in [18], we also define the connection probability ($p(a, b)$) of two neurons a and b in the 3D-mesh architecture, as (13):

$$p(a,b) = \frac{C}{8\pi\lambda^3}e^{-D(a,b)/\lambda}, \quad (13)$$

where $C = N_{links} = ||p(.)||$ is the mean connection count per neuron, λ is a constant presenting spatial connectivity, and $D(a, b)$ is Euclidean distance between a and b.

From (13), the mean distance ($\overline{Dist}^{RNDC}$) between the connected neurons is calculated as (14):

$$\overline{Dist}^{RNDC} = \frac{1}{8\pi\lambda^3}\iiint_{x,y,z}\sqrt{x^2+y^2+z^2}e^{-\sqrt{x^2+y^2+z^2}/\lambda}\mathrm{d}x\mathrm{d}y\mathrm{d}z = \frac{24\pi\lambda^4}{8\pi\lambda^3} = 3\lambda \quad (14)$$

1) UNICAST BASED ROUTING

We first consider the case of the UC where a neuron needs to send C packets to C post-synaptic neurons. Thus, the total distance for a single spike is determined by (15):

$$TotalDist^{RNDC}_{UC} = \overline{Dist}^{RNDC}.N_{links} = 3\lambda C. \quad (15)$$

Since C does not depend on the NoC dimension, C is kept the same as [45] ($C \cong \sqrt{n}$) for fair comparison with 2D-mesh. λ presents locality measurement, a small λ means that the neurons are connected more locally, and vice versa. For a 2D-mesh NoC, $\lambda \cong \sqrt[3]{n}$ leading to optimal performance. Thus, for a 3D-mesh NoC we can determine λ by:

$$\lambda \cong \sqrt[4]{n}. \quad (16)$$

As a result, the efficient bandwidth can be represented as:

$$BW^{RNDC}_{eff,UC} = \frac{3\overline{w}(1-\alpha)\sqrt[3]{n^2}(\sqrt[3]{n}-1)}{3\sqrt[4]{n}\sqrt{n}}.f_{NoC}.U_{NoC} = \frac{\overline{w}(1-\alpha)\sqrt[3]{n^2}(\sqrt[3]{n}-1)}{\sqrt[4]{n^3}}.f_{NoC}.U_{NoC} = O(\sqrt[4]{n}). \quad (17)$$

Furthermore, the average spiking rate of a single SNPC($f^{UC}_{p,out}$), the maximal spiking rate ($f^{UC}_{spike,max}$), and the K ratio for UC can be depicted as:

$$f^{UC}_{p,out} = \frac{BW^{RNDC}_{eff,UC}}{n} = \frac{\overline{w}(1-\alpha)\sqrt[3]{n^2}(\sqrt[3]{n}-1)}{n\sqrt[4]{n^3}}.f_{NoC}.U_{NoC} \quad (18)$$

$$f^{UC}_{spike,max} = \frac{s}{10.C.T_{cycle}} = \frac{s.f_{NoC}}{10\sqrt{n}} = O\left(\frac{1}{\sqrt{n}}\right) \quad (19)$$

$$K^{RNDC}_{UC} = \frac{10\overline{w}(1-\alpha)\sqrt{n}\sqrt[3]{n^2}(\sqrt[3]{n}-1).f_{NoC}}{s.n\sqrt[4]{n^3}.f_{NoC}}.U_{NoC} = O\left(\frac{1}{\sqrt[4]{n}}\right). \quad (20)$$

2) MULTICAST AND BROADCAST BASED ROUTING SCHEMES

For MC, a packet needs to travel along a 3λ path to reach the first destination, and then add one hop for each of the remaining. With a total of C destination nodes, the hop count for each packet is therefore determined by:

$$TotalDist^{RNDC}_{3DMesn,MC} = C + \overline{Dist}^{RNDC} = C + 3\lambda \quad (21)$$

Therefore, the efficient bandwidth can be formulated as:

$$BW^{RNDC}_{eff,MC} \cong \frac{3\overline{w}(1-\alpha)\sqrt[3]{n^2}(\sqrt[3]{n}-1)}{\sqrt{n}+3\sqrt[4]{n}}.f_{NoC}.U_{NoC} = O(\sqrt{n}). \quad (22)$$

The average spike rate for each SNPC is given by:

$$f^{MC}_{p,out} = \frac{BW^{RNDC}_{eff,MC}}{n} = \frac{3\overline{w}(1-\alpha)(\sqrt[3]{n}-1)}{\sqrt[3]{n}(\sqrt{n}+3\sqrt[4]{n})}.f_{NoC}.U_{NoC}. \quad (23)$$

With a link delay T_{cycle} and average distance $\overline{Dist}^{RNDC}$, the maximal spiking frequency is determined as (24):

$$f^{MC}_{spike,max} = \frac{s}{T_{cycle}.\overline{Dist}^{RNDC}} = \frac{s.f_{NoC}}{3\lambda} = \frac{s.f_{NoC}}{3\sqrt[4]{n}} = O\left(\frac{1}{\sqrt[4]{n}}\right). \quad (24)$$

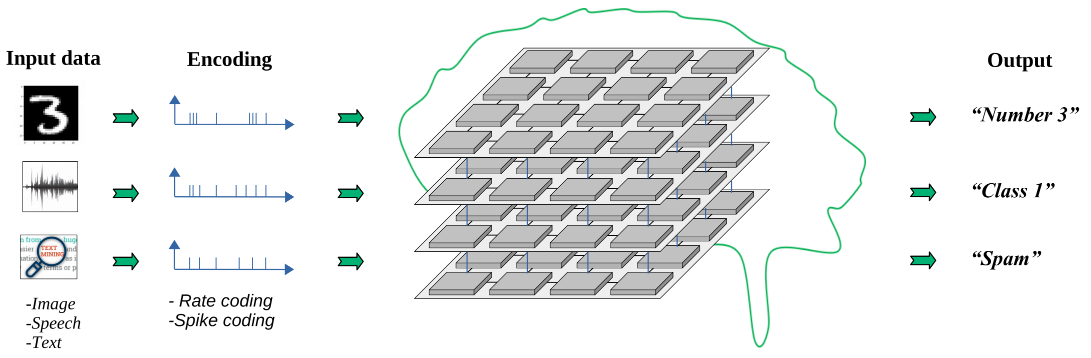


FIGURE 4. 3DFT-SNN system architecture high-level view.

From (23) and (24), the K ratio is given by the following equation:

$$K = \frac{f_{p,out}^{MC}}{f_{spike,max}^{MC}} = \frac{3\overline{w}(1-\alpha)(\sqrt[3]{n}-1)3\sqrt[4]{n}}{s.\sqrt[3]{n}(\sqrt{n}+3\sqrt[4]{n})}.U_{NoC} = O\left(\frac{1}{\sqrt[4]{n}}\right) \quad (25)$$

For the case of BC, the RNDC architecture is similar to the case of Hopfield. The only difference is in the network size (i.e., C for RNDC and n for Hopfield) As a result, the performance metrics are kept similar to the HF neural network.

In summary, for running RNDC on the architecture with a link fault rate α, MC offers higher spiking frequency compared to UC and BC. From the assessment analysis for both Hopfield and RNDC neural network topologies, we can see that the link failure causes performance degradation in the communication architecture. This may lead to timing violations of spikes. Therefore, a low-latency fault-mechanism routing method is imperative to deal with this issue.

IV. 3DFT-SNN ARCHITECTURE OVERVIEW

A. OVERALL SYSTEM ARCHITECTURE

This section presents the proposed 3DFT-SNN system, as shown in Fig. 4. The system comprises several stacked 2D layers (4×4 2D layers of spiking neural tiles stacked together are shown as an example) of spiking neural tiles connected by a 3D-mesh architecture. A spiking neural tile consists of a spiking neural processing core (SNPC) and a fault-tolerant multicast router (FTMC-3DR). SNPCs accommodate spiking neurons as main computation units in the system. FTMC-3DR routers are responsible for delivering spike packets generated by neurons in SNPCs to their post-synaptic neurons.

B. SPIKING NEURAL TILE

1) SPIKING NEURON PROCESSING CORE (SNPC)

Fig. 5 shows the high-level architecture of the proposed spiking neural processing core (SNPC) composing of several main components. The decoder is responsible for determining post-synaptic neurons for each incoming spike (packet). After arriving the destination neural tile, the incoming spike is forwarded to local SNPC by the local router. Based on "neuron ID" extracted from the spike packet, the decoder looks up in a LUT to determine the post-synaptic neurons. This information is sent to the control unit for neural computation. The SNPC is based on a crossbar approach similar to other systems, such as TrueNorth [26] and ODIN [50]. Here, we use on-chip SRAMs to implement an N $\times$ N crossbar (N is the number of neurons). Each synapse is represented by 5 bits, 1 bit for synaptic types (i.e., excitatory and inhibitory) and 4 bits for weight. The array of leaky-integrate and fire (LIF) neurons is the main computation unit of the neuron core where neural calculations are performed. Here, we adopt the LIF model because of its trade-off between biological computation ability and hardware complexity. The proposed neuron core is inherited from our recent work [4].

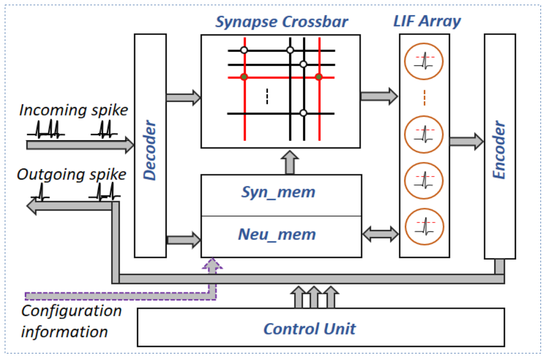
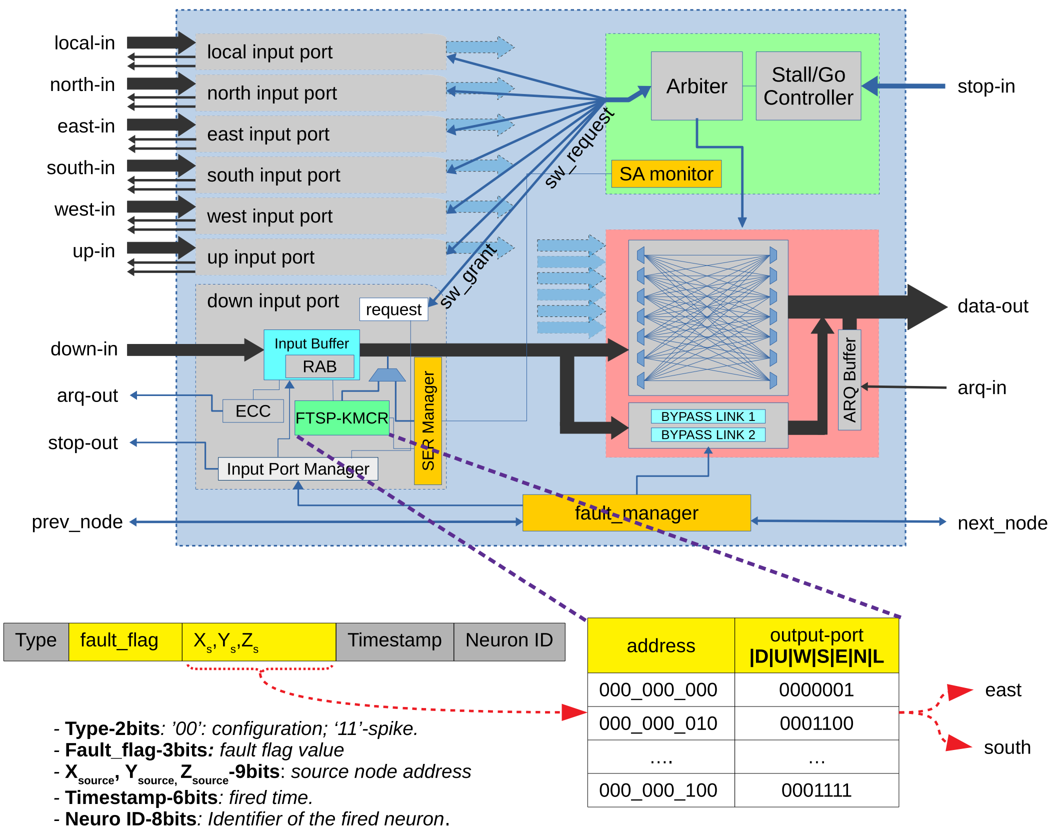


FIGURE 5. Spiking Neuron Processing Core (SNPC) architecture.

We have to note here that there is a wide diversity of spiking neuron models that have been proposed to mimic spiking response, from the integrate-and-fire (IF) [3] model and its quadratic and exponential variants to multiple-variable models such as the Hodgkin-Huxley (HH) [51] and Izhikevich (IZ) [52] models. The leaky integrate-and-fire model adopted in this work is a simplified version and neglects many aspects of neuronal dynamics. In particular, the input, which may arise from pre-synaptic neurons or the current injection, is integrated linearly. However, our proposed system can still support other complex neuron models without major



Type	fault_flag	X_s,Y_s,Z_s	Timestamp	Neuron ID

- **Type-2bits**: *'00': configuration; '11'-spike.*
- **Fault_flag-3bits**: *fault flag value*
- **X_{source}, Y_{source}, Z_{source}-9bits**: *source node address*
- **Timestamp-6bits**: *fired time.*
- **Neuro ID-8bits**: *Identifier of the fired neuron*.

address	output-port |D|U|W|S|E|N|L
000_000_000	0000001
000_000_010	0001100
….	…
000_000_100	0001111

FIGURE 6. Fault-tolerant Multicast Spike 3D Router architecture (FTMC-3DR).

modification to the proposed algorithms and architecture. Particularly, the memories (i.e., *Syn_mem* and *Neu_mem*) need to be extended to accommodate more required parameters of the more complex neuron model. The spike packet format also needs to be adjusted to convey other information required for complex neuron models.

2) FAULT-TOLERANT MULTICAST 3D ROUTER ARCHITECTURE (FTMC-3DR)

The fault-tolerant multicast 3D router (FTMC-3DR) architecture is represented in Fig. 6, where the proposed routing methods are integrated. The router is designed with four pipeline stages: buffer writing (BW), routing calculation (RC), switch arbitration (SA), and crossbar traversal (CT). At the first stage, an incoming spike (packet) is stored in the *Input Buffer* before being processed. Next, the source address of the packet (X_s, Y_s, Z_s) is extracted and computed to determine which is the output port. This routing calculation will be presented in Section V. After routing computation, a request (*sw_request* signal) is sent to *Switch-Allocator* in order to use the selected output port. The *Switch-Allocator* consists of two main components: *Stall/Go* flow control (the most common use in systems [10]) and *Matrix-arbiter* scheduler. Here, the *Matrix-arbiter* with least recently served priority is employed since it provides fast computation, inexpensive implementation, and strong fairness [10]. Finally, after granted (via *sw_grant* signal), the packet is sent to desired output port through the crossbar.

The proposed router relies on sophisticated recovery techniques based on system reconfiguration with redundant structural resources to handle hard faults in the input-buffers, crossbar, and links [13], [53], in addition to soft errors in the routing pipeline stages [54]. These mechanisms aim to alleviate faults occurring in the system.

V. K-MEANS CLUSTERING BASED FAULT-TOLERANT MULTICAST SPIKE ROUTING ALGORITHM

A. OVERVIEW OF THE K-MEANS BASED MULTICAST ROUTING ALGORITHM (KMCR)

Before presenting the proposed fault-tolerant multicast routing algorithm, we first survey our previous work, named KMCR [19]. The KMCR is a multicast routing scheme for SNN-based 3D-mesh NoCs. It is based on a tree-based routing mechanism combined with the k-means clustering algorithm. The routing tree from a given source node to its destinations is formed by: (1) adopting the k-means aim to divide the destination set into balanced partitions. This mitigates the high congestion usually found in wormhole tree-based architectures. After applying the k-means, the centroid(s) and its(their) labelled destinations are determined, (2) the first part of the tree is formed from the source node to centroid(s), and (3) the other part of the tree is a spanning sub-tree from

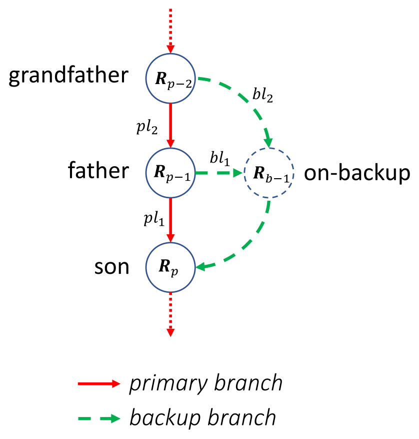


FIGURE 7. **Primary and backup branches.**

the centroid(s) to its(their) destinations. In KMCR, k needs to be determined before performing KMCR. Therefore, in [19], we also proposed a method to choose the optimal value of k.

B. SHORTEST PATH K-MEANS BASED MULTICAST ROUTING ALGORITHM (SP-KMCR)

In the KMCR, the source node sends spike packets to centroids which then delivers the spikes to destinations. The use of centroids is to guarantee that the overall distance from them to the destinations is minimum. However, this may cause traffic congestion on the link to the centroids since the traffic from different sources is concentrated there.

To deal with this issue, we propose a new routing method. After destination subsets are determined by adopting k-means, from a given source, we first calculate the numbers of hops from the source to all the nodes in the subsets. For each subset, we then select a node which has the shortest path to the source. Contrary to the KMCR, the source sends its spike packets to the shortest path node of each subset instead of the centroid node, and we named our new method SP-KMCR. This may eliminate the potential problem of traffic congestion and reduce the average latency as well. Furthermore, It is worth mentioning that our new method requires more computations for finding the shortest path compared to the KMCR. However, the computations in both our new method and the previous KMCR are executed off-line, making the runtime overhead the same for both algorithms.

C. FAULT-TOLERANT SHORTEST PATH K-MEANS BASED MULTICAST ROUTING ALGORITHM (FTSP-KMCR)

The shortest path fault-tolerant multicast routing algorithm is based on the SP-KMCR. The basic idea of the FTSP-KMCR is as follows: (1) off-line computations of a primary routing tree from a given source node to its destinations and backup routing branches are performed. (2) After the off-line calculation, the routing tables are configured.

The illustration of the primary and backup routing branches is shown in Fig. 7. When a faulty primary branch is detected, some pre-planned backup branch(es) is (are) used to bypass the faulty links. The SP-KMCR mechanism is used to calculate the branches (red) in the primary tree. The backup branches are alternative routes of the primary ones. For a considered router (i.e., "son"), the backup branches (green) are computed for the cases of faults occurring in primary connections. For example, when the father-to-son primary connection is faulty (i.e., pb_1), bb_1 and bb_2 are the backup branches used for maintaining the traffic between the "father" and "son". This is the same for the case where both pb_2 and pb_1 are faulty.

In our proposed algorithm, the computations of primary and backup routes are critical computational tasks. These calculations are performed off-line, and this reduces runtime overhead of the proposed routing algorithm; hence avoiding any possible timing violations in SNNs. As presented in algorithm 1, the source and destination addresses (S, T) and the number of subsets (clusters) (k) are pre-defined as inputs, while output parts are a primary tree (P_{pr}) from each source to destinations and backup branches (P_{bk}). After that, the routing computation is done according to the following steps:

- **Step 1:** from destination addresses, destination subsets are determined by adopting k-means, as shown in lines *6-19*.
- **Step 2:** finding the shortest path from each source to a node (named $sp_i \in SP$) in each subset (with k subsets T^k, a given source node has k SP nodes), as depicted in lines *20-25*.
- **Step 3:** the first part of the primary tree is formed from the source node to SP ones. This is done by adopting dimension order routing (DOR) algorithm [55] from the source to each SP node, then merge with the same route. Alternative variations of the DOR are then adopted to calculate backup branches in order to guarantee that backup branches are separated from the primary routes. For example, if the formation of the primary tree uses DOR of ZYX, for backup branches, we use other variations of the DOR such as YZX or XZY.
- **Step 4:** following the same computation in step 2, the second part of the primary tree from SP nodes to their destinations in the same group and backup branches are calculated.

It should be noted that only primary and backup routing paths are off-line computations. These computation results are then used to configure routing tables in routers. Since the configuration process is performed during the application mapping before running time, it does not affect the category of online learning processes where only synaptic strengthens (weights) are updated. This guarantees that the computation overhead of backup branches does not affect the recovery time of the proposed routing algorithm, and also reduces the required hardware cost of the system.

D. FAULT MANAGEMENT ALGORITHM

After the routing information is configured, the fault-management algorithm is implemented to handle incoming

Algorithm 1 Off-Line Calculations of the Primary and Backup Branches

```
/* Input and output */
Input: // Source node address ($S$), destination node addresses ($T$), and the number of subsets ($k$)
1     $S = \{s_1(x_1, y_1, z_1), s_2(x_2, y_2, z_2), \ldots, s_n(x_n, y_n, z_n)\}$
2     $T = \{t_1(x_1, y_1, z_1), t_2(x_2, y_2, z_2), \ldots, t_m(x_m, y_m, z_m)\}$
3     $k$
Output: //Primary (pr) and backup (bk) branches from S to T
4     $P_{pr} = \{p_{pr,1}(s_1 \to T), p_{pr,2}(s_2 \to T), \ldots p_{pr,n}(s_n \to T)\}$
5     $P_{bk} = \{p_{bk,1}(s_1 \to T), p_{bk,2}(s_2 \to T), \ldots p_{bk,n}(s_n \to T)\}$
/* Centroid node assignment */
// Initial centroid nodes by randomly select from $T$
6  foreach $c_i \in C$ do
7      $c_i \leftarrow t_j \in T$
8  end
// Evaluate centroid nodes
9  do
       // Calculate the distance between $t_i \in T$ to $c_j \in C$
10     foreach $t_i \in T$ do
11         $d(t_i, c_j) = |x_i - x_j| + |y_i - y_j| + |z_i - z_j|$
12     end
       // Assign each destination to its centroid by minimum distance
13     foreach $t_i \in T$ do
14         $l(t_i) \leftarrow argmind(c_i, t_j)$
15     end
       // Update centroid
16     foreach $c_i \in C$ do
17         $c_i \leftarrow update(mean(t_{ij}))$
18     end
19 while $C$ != $const$;
/* Finding the shortest paths */
20 foreach $s_i \in S$ do
21     foreach $t_i \in T^k$ do
22         $d(s_i, t_j) = |x_i - x_j| + |y_i - y_j| + |z_i - z_j|$
23     end
24     $sp_i \leftarrow min(d(s_i, t_j))$
25 end
/* Creating primary and backup branches */
// from each source to SP node
26 foreach $s_i \in S$ do
27     $p_{pr}(s_i, sp_j) \leftarrow DOR^{v.1}\_based\_tree(s_i, sp_j)$
28     $p_{bk}(s_i, sp_j) \leftarrow DOR^{v.\neq 1}\_based\_tree(s_i, sp_j)$
29 end
// from each SP node to its destinations
30 foreach $sp_i \in SP$ do
31     $p_{pr}(sp_i, t_j) \leftarrow DOR^{v.1}\_based\_tree(sp_i, t_j)$
32     $p_{bk}(sp_i, t_j) \leftarrow DOR^{v.\neq 1}\_based\_tree(sp_i, t_j)$
33 end
```

packets, as shown in Fig. 8. For a given incoming packet, *fault_flag_val* is extracted to indicate whether the packet is in the primary or backup branch. At the same time, the source address is also used to compute its expected primary output port. In the case where *fault_flag_val* $= 0$ (i.e., the router plays the role of "father" or "grandfather"), the calculated *output_port* is then determined to be faulty or not. If it is not faulty, the packet is forwarded to the calculated output port in the primary branch. Otherwise, *output_port* is switched to use a backup_branch, and the *fault_flag_val* is also initiated to inform the next on-backup routers that this packet is on the backup branch. In the case where *fault_flag_val* $\neq 0$ (i.e., the router role is as a on-backup or "son" router), the *output_port* is routed through the backup route, and *fault_flag_val* is also decreased by one.

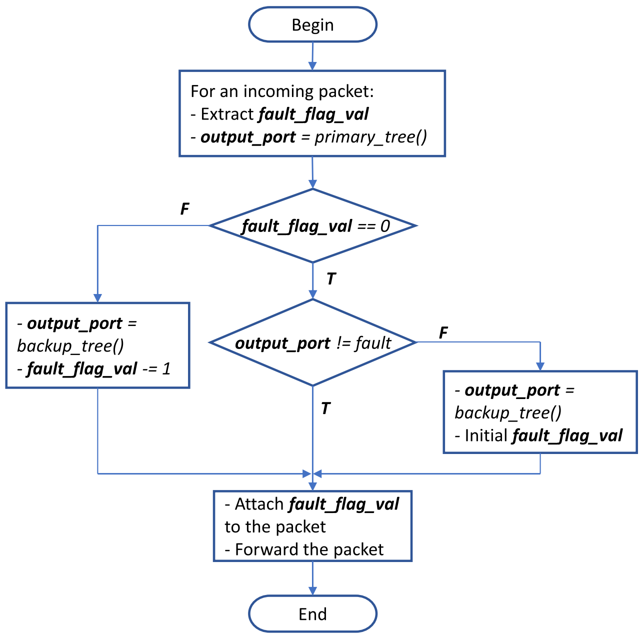


FIGURE 8. Fault-management algorithm applied for "son", on-backup, "father" and "grandfather" routers.

VI. PERFORMANCE EVALUATION

A. EVALUATION METHODOLOGY

The proposed 3DFT-SNN is described in Verilog HDL and evaluated in terms of average spike latency and throughput. The system implementation is synthesized with Synopsys Design Compiler and compiled using NANGATE 45nm CMOS technology. Also, experiments are conducted with investigated applications, both with and without fault injection.

In the experiments, we selected two well-known applications for hardware-based SNNs [56], [57], and they are Inverted Pendulum and Wisconsin Data-set. Application mapping method and experiment setup are similar to [19], [58]. In this work, we mapped SNNs onto the proposed system in a layer-to-layer fashion to take full advantage of the proposed routing algorithm and the 3D mesh NoC topology. It should be mentioned that the mapping strategy of SNNs onto NoC based system plays an important role in deploying SNN applications. It affects not only the overall performance but also the power consumption of the whole system. In [59], the authors proposed two mapping

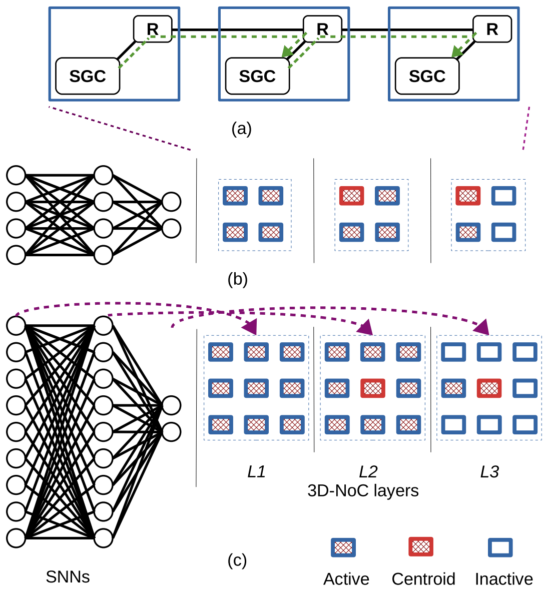


FIGURE 9. Layer-to-layer application mapping: (a) block diagram of a node in each layer (b) Inverted pendulum (c) Wisconsin Data-set, as presented in [58].

TABLE 1. Simulation configuration.

System	XYZ-UB	KMCR	SP-KMCR (this work)	FTSP-KMCR (this work)
NoC size	2×2×3			
	3×3×3			
Buffer depth	4			
Switching	Wormhole			
Flow control	Stall-go			
Scheduling	Matrix-Arbiter			
Routing	Unicast-based multicast	Multicast	Multicast	Fault-tolerant multicast

methods: (1) a relatively conventional approach that puts highly communicating tasks together, and (2) an approach based on active degrees of neurons. In our experiments, the applications were mapped onto the 3DFT-SNN system in a layer-to-layer manner as represented in Fig. 9. In this mapping method, the neurons in the same network layer are placed in the same 3DFT-SNN layer, and neurons only send their spike to the ones in the next layer. This approach offers multiple parallel connections between layers (vertical connections), less congestion, and low spike latency when compared to the 2D integration method [13]. As shown in Fig. 9, it is easy to find that $k = 1$. By experiment, we found that an increase of k results in raising average latency. This is because each source node needs to send multiple copies of spike packets to SP nodes, as found in [19]. In evaluations, the selection of k is therefore equal to *1*.

To make a performance comparison with the proposed algorithm, we also implemented a unicast-based multicast (Section III), named XYZ-UB. XYZ is one of the variations of dimension order routing (DOR). It is a simple algorithm, easy to implement, and free of deadlock and lifelock [55]. The evaluation configuration is shown in Table 1.

As mentioned before, the FTMC-3DR is based on our previous 3D router architectures (SHER-3DR) [13], [53], [54] which relies on sophisticated recovery techniques to handle hard faults in the input-buffers, crossbar, and links as well as soft errors in the routing pipeline stages. However, during the evaluation of the 3DFT-SNN system, all the fault-tolerance techniques which are found in the previous SHER-3DR are disabled for fair comparison.

B. PERFORMANCE EVALUATION WITHOUT FAULT INJECTION

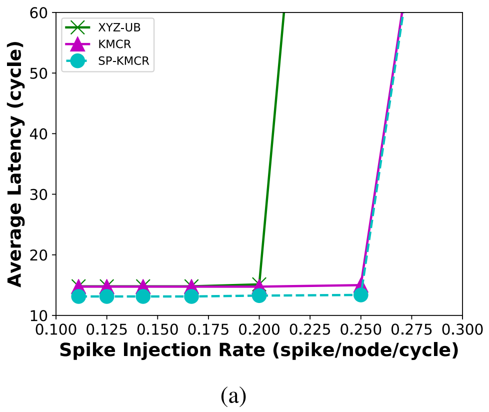
We first compare the SP-KMCR with the previous KMCR and the XYZ-UB in terms of average latency and throughput to explore its performance potential. The average communication latency as a function of Spike Injection Rate (SIR) is shown in Fig. 10.

For the Inverted Pendulum application, the SP-KMCR and KMCR show almost the same average latency as XYZ-UB before this latter reaches its saturation point at SIR $= 0.2$. The SP-KMCR and KMCR sustain almost the same latency while providing 25% higher SIR, when compared to XYZ-UB. This can be explained by the fact that XYZ-UB needs to send multiple copies of a given spike, resulting in high traffic contention. These results mean that the proposed system can maintain high SNN traffic such as fast and bursting operation modes of spiking neurons [60]. Furthermore, this is also consistent with the analysis in Section III-C, where multicast has higher maximum frequency compared to unicast-based multicast, as in (19), (24) with fault rate $\alpha = 0$.

On the other hand, SP-KMCR reduces the latency by 12.2% when compared to KMCR before it reaches the saturation point, even with a small network size such as the one used for the Inverted Pendulum application. This is due to the fact that source nodes in the SP-KMCR send spikes to their shortest path node instead of centroid node in KMCR; resulting in alleviating the congestion in the intermediate node like the centroid. The evaluation results also demonstrate that the SP-KMCR enables systems running SNN applications with a smaller time-step, to improve their acceleration potential.

For Wisconsin Data-set benchmark, the increase of the network size causes a higher latency when compared to the Inverted Pendulum application. The latency also increases with the increase of SIR. The unicast-based system suffers higher latency compared to the KMCR by about 14.43%, at SIR $= 11\%$. Furthermore, KMCR can support a higher SIR reaching up to 22.22% when compared to the unicast-based. Compared to the KMCR, the SP-KMCR reduces the average latency by 9.5%, at the highest injection rate.

Fig. 11 shows the evaluation results of the average throughput. As shown in the Fig. 11, SP-KMCR and KMCR achieve 24.5% and 22% higher throughput compared to XYZ-UB while executing the Inverted Pendulum and Wisconsin Data-set benchmarks, respectively. This comes from the fact that the SP-KMCR and KMCR help the system servicing SNN traffic at higher injection rate.



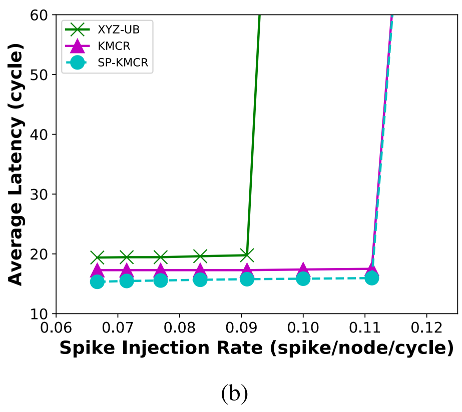
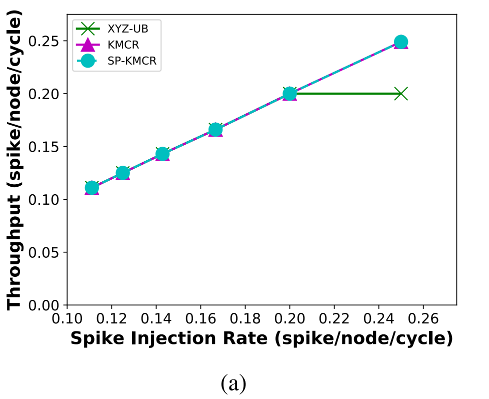


FIGURE 10. Average latency over various SIRs: (a) Inverted Pendulum (b) Wisconsin Data-set.



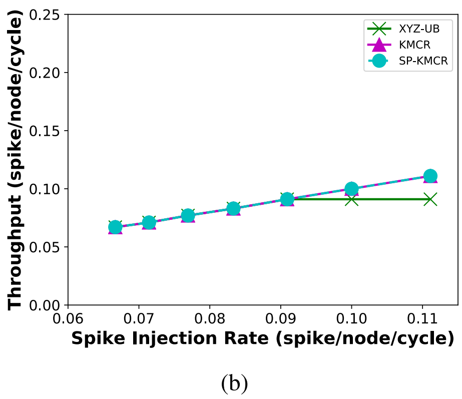


FIGURE 11. Average throughput over various SIRs: (a) Inverted Pendulum (b) Wisconsin Data-set.

The evaluation results also show the proposed multicast routing benefits in terms of efficient bandwidth when running SNN applications. These results validate the analysis in Section III-C, with fault rate $\alpha = 0$.

In [19], we evaluated and compared the KMCR with three other existing works: Dragonfly [61], H-NoC [12], and Cmesh [62]. The evaluation results showed that KMCR can maintain a higher SIR before saturation point, by about 8.7% when compared to the best algorithm (i.e., Dragonfly) among the three considered works. This allows us to further believe that the proposed SP-KMCR in this research shows better performance when compared to Dragonfly, H-NoC, and Cmesh.

To further explore the improvement of SP-KMCR, we evaluate both KMCR and SP-KMCR with larger network sizes ($3 \times 3 \times 2$, $4 \times 4 \times 2$, and $5 \times 5 \times 2$). Here, all nodes in the first layer send packets to all the other nodes in the second layer. Fig. 12 compares the performance of KMCR and SP-KMCR in terms of average latency. The evaluation result shows that the SP-KMCR achieves lower average latency compared to KMCR, at about 10.29%, 16.86%, and 23.57% with 3DNoC sizes of $3 \times 3 \times 2$, $4 \times 4 \times 2$, and $5 \times 5 \times 2$, respectively. It means that the proposed SP-KMCR improves the average latency, especially for large network sizes.

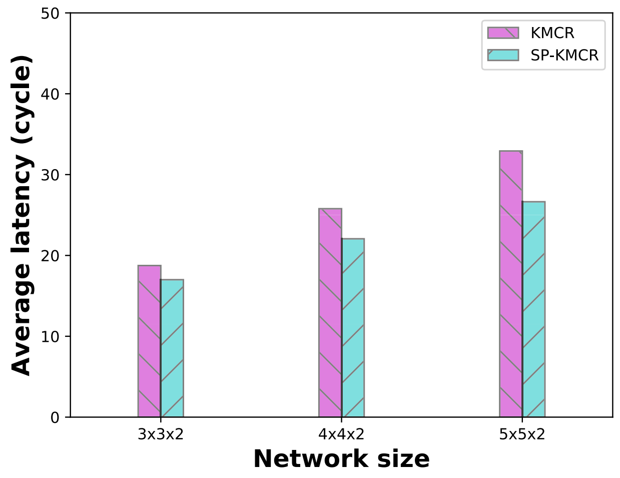
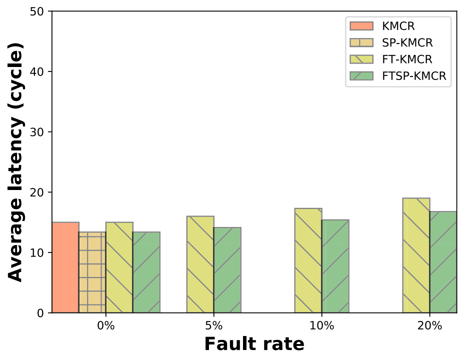


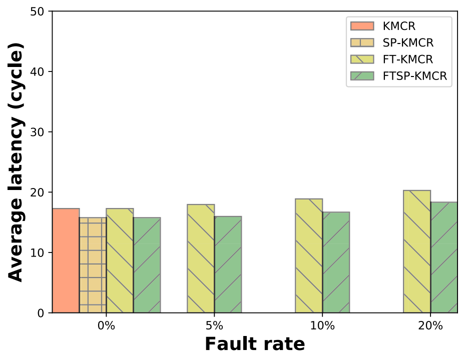
FIGURE 12. Average latency comparison of KMCR and SP-KMCR over different network sizes.

C. PERFORMANCE EVALUATION WITH FAULT INJECTION

In this evaluation, we explore the fault-tolerance potential of the proposed algorithm. In this experiment, the fault-tolerant mechanism was added to both KMCR and SP-KMCR baselines, denoted as FT-KMCR and FTSP-KMCR, respectively.

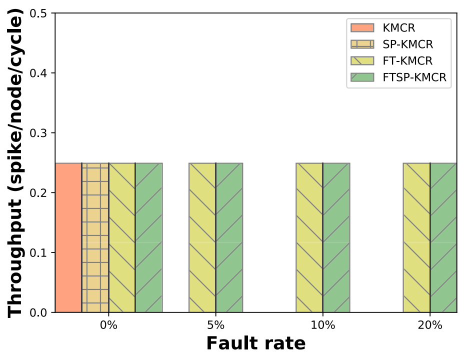


(a)

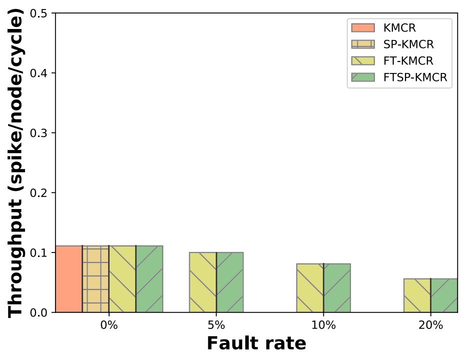


(b)

FIGURE 13. **Average latency over various fault rates: (a) Inverted Pendulum (b) Wisconsin Data-set.**



(a)



(b)

FIGURE 14. **Average throughput over various fault rates: (a) Inverted Pendulum (b) Wisconsin Data-set.**

The average latency as a function of fault rate is shown in Fig. 13. As represented in this figure, the FT-KMCR and FTSP-KMCR keep the same latency compared to their baseline systems when no fault is injected. This is because both fault-tolerant systems and their baseline use the same routing tree. On the other hand, there is a slight increase in the average latency of the FT-KMCR and FTSP-KMCR when increasing the fault rate. This comes from the fact that the fault-tolerant architectures use backup branches resulting in high traffic at the remaining healthy links. For Inverted Pendulum, the average latency of the FT-KMCR increases by about 6.67%, 15.33% and 26.67% at 5%, 10%, and 20% fault rates, respectively, compared to the KMCR. Here, the maximum spike injection rate is 0.25 spike/node/cycle. The figures for FTSP-KMCR are lower: about 5.61%, 15.10%, and 25.34%. The increase of average latency results in the system running in a longer timestep; however, the system can correctly run SNN applications at a higher fault rate.

For Wisconsin Data-set, the average latency of FTSP-KMCR increases by 1.27%, 5.77%, and 16.23% when compared to its SP-KMCR baseline system. These evaluation results show that the proposed FTSP-KMCR has lower average latency compared to the FT-KMCR in both applications. The latency reduction is due to two main reasons: first, as explained above, the source nodes in FTSP-KMCR send packets to their shortest path node instead of centroid. Second, since backup routing computations are performed off-line, the runtime overhead of the proposed fault-tolerant technique is the same as its baseline. This helps the system to better reduce the effect of timing violations in SNNs caused by the long latency of recovery mechanisms.

Fig. 14 compares the throughput of the proposed and the baseline systems. For the Inverted Pendulum, the architectures show similar average throughput results when increasing the fault rate, thanks to the redundancy of the architecture used for backup routing paths. For example, as shown in the right side of Fig. 9 (b), when the ZYX version of the DOR is used for determining the primary tree, all intra-layer links in the first layer ($L1$) are not used. These links can be used as potential backup branches. This allows the proposed fault-tolerant architecture to maintain the communication traffic at the highest spike injection rate (i.e., the rate before the saturation point at 0% fault rate). This is the reason why the throughput is unchanged, while the average latency increases when raising the fault rate due to the larger hop count of the backup branches. On the other hand, there is a

TABLE 2. Power consumption of the KMCR and FTSP-KMCR under the benchmarks.

System	KMCR		FTSP-KMCR	
	Inv. Pen.	Wis.	Inv. Pen.	Wis.
Area (mm^2)	0.102	0.346	0.108	0.365
Power (mW)	10.13	34.20	10.64	35.92

decrease in the average throughput of the proposed system under Wisconsin Data-set benchmark when increasing the fault rate. This is caused by the larger number of neurons used in this application resulting in higher contention in primary and backup branches. Therefore, the proposed architecture is not able to keep the same spike injection rate when increasing the fault rate, as it was the case for the Inverted Pendulum application. At a fault rate of 20%, the throughput of FT-KMCR and FTSP-KMCR decreases by 49.5% (i.e., the spike injection rate of 0.056 spike/node/cycle) compared to the system without fault. Nevertheless, the proposed algorithm was capable of correctly delivering all the spikes to their destinations despite this high fault rate. The evaluation results also show that a higher fault rate leads to reduce the spiking frequency, as analyzed in Section III-C.

D. HARDWARE COMPLEXITY EVALUATION

We also evaluated the hardware complexity in terms of area cost and total power consumption of the KMCR and FTSP-KMCR architectures (i.e., the entire network) to observe the extra hardware resources necessary for the proposed fault-tolerant method, as shown in Table 2. Regarding the area cost, the FTSP-KMCR uses more area than the KMCR system (about 5.88% and 5.49% for the Inverted Pendulum and the Wisconsin dataset, respectively). This is also consistent with the power consumption results, in which the FTSP-KMCR consumes a higher amount of power (about 5.03% and 4.97% for the Inverted Pendulum and the Wisconsin dataset, respectively). This hardware overhead is mainly due to the extra hardware needed for the backup branches.

VII. DISCUSSION

In the previous section, we demonstrated the capabilities of the proposed SP-KMCR algorithm to improve the average latency when compared to its KMCR predecessor. In addition, we validated the reliability of the proposed architecture to sustain correct inter-neural communication even at a 20% fault rate, while ensuring small hardware complexity and graceful performance degradation. Nevertheless, a couple of points need to be discussed in order to exploit the full potential of the proposed architecture. Hereafter, we address these challenges and highlight the possible solutions.

The first point to be addressed is the improvement of the proposed fault-tolerant multicast routing algorithm to deal with multiple faults. In particular, when successive multiple faults occur in the primary branches. In the current implementation, the proposed algorithm should forward spike packets through backup branches for every single faulty link. In Fig. 7, for example, if both pb_1 and pb_2 are faulty, incoming spikes from "grandfather" are firstly forwarded to "father" through backup branches, then from "father" to "son". However, the latency increases due to the additional unnecessary hop travel. This issue can be tackled by using direct backup paths from "grandfather" (or another ancestor) to "son", and this requires that "grandfather" (or another ancestor) has to know the status of the primary branches to "son". To do so, "son" monitors the status of the upstream link. When a fault is detected, the "son" should notify all the routers on its backup path. In this fashion, whenever and wherever any fault occurs, our algorithm will be able to reconfigure the network to deliver multicast packets as long as the routing table size allows it.

The second challenge is the implementation capability of large neural networks. Although the number of neurons per core (SNPC) is small in the current experiments, the proposed SP-KMCR approach improves the average latency, especially for large network sizes. In our final system, each SNPCs will support up to 256 neurons. As a result, the system can accommodate about 7,000 neurons with a network size of $3 \times 3 \times 3$, as used in the current evaluation. This is a plausible scenario because, setting the operation frequency of our 3DFT-SNN to 100MHz, takes an average of 2.02ns to deliver a spike at a fault rate of 20%. With 256 neurons in an SNPC, it takes 517.12ns. This means that the network architecture can still perform $1,933.8\times$ faster than the real-time requirement of SNNs; that is 1ms (spike rate up to 1KHz) [60]. It should be mentioned that these calculations are taken under the time-multiplexing manner handled by the input scheduler in SNPC. This requires handling of the SNN state where time constants should be maintained, especially in the bursting operation modes of spiking neurons [60]. To deal with this issue, we can adopt the idea of compression that was first introduced in [12], in which spikes generated by neurons in the same layer are compressed into a single packet before injecting into the network. This technique allows our system to handle large network traffic.

VIII. CONCLUSION

In this paper, we presented a comprehensive analytic performance assessment and low-latency fault-tolerant algorithm for spike traffic routing in 3D packet-switching network of spiking neurons (3DFT-SNN). The detailed analysis is proposed to analyze the performance of different neural network topologies. The goal of this analytical model is to help designers early understand the effect of faulty links on the performance of their architecture over neural network topologies and spike routing schemes. The proposed fault-tolerant multicast routing algorithm is based on a so-called Tree-based-routing (TBR) combined with the K-means clustering. The proposed 3DFT-SNN has been evaluated with real benchmarks in terms of latency and throughput, as well as the hardware complexity (area cost and power consumption). From the evaluation results, we conclude that the proposed SP-KMCR algorithm reduces the average latency by 12.2% when compared to its KMCR predecessor.

Furthermore, the proposed fault-tolerant variation (i.e, FTSP-KMCR) enables the system to maintain a correct traffic communication with a fault rate of 20% while suffering only 16.23% longer latency and 5.49% extra area cost compared to the SP-KMCR without fault-tolerance capabilities.

REFERENCES

[1] C. Mead, "Neuromorphic electronic systems," *Proc. IEEE*, vol. 78, no. 10, pp. 1629–1636, Oct. 1990.

[2] J. A. Levin, V. Rangan, and E. C. Malone, "Efficient hardware implementation of spiking networks," U.S. Patent WO 2014/189 970 A3, Nov. 27, 2014.

[3] N. Burkitt, "A review of the integrate-and-fire neuron model: I. Homogeneous synaptic input," *Biol. Cybern.*, vol. 95, no. 1, pp. 1–19, 2006. doi: 10.1007/s00422-006-0068-6.

[4] K. Suzuki, Y. Okuyama, and A. B. Abdallah, "Hardware design of a leaky integrate and fire neuron core towards the design of a low-power neuro-inspired spike-based multicore soc," in *Proc. Inf. Process. Soc. Tohoku Branch Conf.*, Feb. 2018. [Online]. Available: http://www.topic.ad.jp/ipsj-tohoku/doku.php?id=report:seminar_20180210

[5] J. H. Goldwyn, N. S. Imennov, M. Famulare, and E. Shea-Brown, "Stochastic differential equation models for ion channel noise in Hodgkin-Huxley neurons," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 83, no. 1, 2011, Art. no. 041908.

[6] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, "Deep learning in spiking neural networks," *Neural Netw.*, vol. 111, pp. 47–63, Mar. 2019.

[7] Y. Cao, Y. Chen, and D. Khosla, "Spiking deep convolutional neural networks for energy-efficient object recognition," *Int. J. Comput. Vis.*, vol. 113, no. 1, pp. 54–66, 2015.

[8] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers Comput. Neurosci.*, vol. 9, p. 99, Aug. 2015.

[9] C. Eliasmith, T. C. Stewart, X. Choo, T. Bekolay, T. DeWolf, Y. Tang, and D. Rasmussen, "A large-scale model of the functioning brain," *Science*, vol. 338, no. 6111, pp. 1202–1205, 2012.

[10] A. Ben Abdallah, *Advanced Multicore Systems-on-Chip: Architecture, On-Chip Network, Design*. Singapore: Springer, 2017.

[11] R. Hojabr, M. Modarressi, M. Daneshtalab, A. Yasoubi, and A. Khonsari, "Customizing clos network-on-chip for neural networks," *IEEE Trans. Comput.*, vol. 66, no. 11, pp. 1865–1877, Nov. 2017.

[12] S. Carrillo, J. Harkin, L. J. McDaid, F. Morgan, S. Pande, S. Cawley, and B. McGinley, "Scalable hierarchical Network-on-Chip architecture for spiking neural network hardware implementations," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 12, pp. 2451–2461, Dec. 2013.

[13] K. N. Dang, A. B. Ahmed, Y. Okuyama, and A. B. Abdallah, "Scalable design methodology and online algorithm for TSV-cluster defects recovery in highly reliable 3D-NoC systems," *IEEE Trans. Emerg. Topics Comput.*, to be published.

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

[15] P. M. Furth and A. G. Andreou, "On fault probabilities and yield models for VLSI neural networks," *IEEE J. Solid-State Circuits*, vol. 32, no. 8, pp. 1284–1287, Aug. 1997.

[16] P. U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu, and M. Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2015, pp. 1–8.

[17] M. Pfeiffer and T. Pfeil, "Deep learning with spiking neurons: Opportunities and challenges," *Frontiers Neurosci.*, vol. 12, p. 774, Oct. 2018. [Online]. Available: https://www.frontiersin.org/article /10.3389/fnins.2018.00774

[18] D. Vainbrand and R. Ginosar, "Scalable network-on-chip architecture for configurable neural networks," *Microprocessors Microsyst.*, vol. 35, no. 2, pp. 152–166, Mar. 2011. doi: 10.1016/j.micpro.2010.08.005.

[19] T. H. Vu and A. B. Abdallah, "Low-latency K-means based multicast routing algorithm and architecture for three dimensional spiking neuromorphic chips," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, Kyoto, Japan, Feb. 2019, pp. 1–8.

[20] H. Markram, "The blue brain project," *Nature Rev. Neurosci.*, vol. 7, pp. 153–160, Feb. 2006.

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

[22] A. Mortara, E. A. Vittoz, and P. Venier, "A communication scheme for analog VLSI perceptive systems," *IEEE J. Solid-State Circuits*, vol. 30, no. 6, pp. 660–669, Jun. 1995.

[23] J. Lazzaro and J. Wawrzynek, "A multi-sender asynchronous extension to the AER protocol," in *Proc. 16th Conf. Adv. Res. VLSI*, Mar. 1995, pp. 158–169.

[24] S. Carrillo, J. Harkin, L. McDaid, S. Pande, S. Cawley, B. McGinley, and F. Morgan, "Advancing interconnect density for spiking neural network hardware implementations using traffic-aware adaptive network-on-chip routers," *Neural Netw.*, vol. 33, no. 9, pp. 42–57, 2012.

[25] 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 Des. Integr. Circuits Syst.*, vol. 34, no. 10, pp. 1537–1557, Oct. 2015.

[26] 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, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, Aug. 2014.

[27] S. Yang, J. Wang, B. Deng, C. Liu, H. Li, C. Fietkiewicz, and K. A. Loparo, "Real-time neuromorphic system for large-scale conductance-based spiking neural networks," *IEEE Trans. Cybern.*, vol. 49, no. 7, pp. 2490–2503, Jul. 2019.

[28] J. Wu and S. Furber, "A multicast routing scheme for a universal spiking neural network architecture," *Comput. J.*, vol. 53, no. 3, pp. 280–288, Apr. 2009.

[29] K. A. Boahen, "Point-to-point connectivity between neuromorphic chips using address events," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 47, no. 5, pp. 416–434, May 2000.

[30] S.-C. Liu, J. Kramer, G. Indiveri, T. Delbrück, T. Burg, and R. Douglas, "Orientation-selective aVLSI spiking neurons," *Neural Netw.*, vol. 14, nos. 6–7, pp. 629–643, 2001.

[31] S. Furber, "Large-scale neuromorphic computing systems," *J. Neural Eng.*, vol. 13, no. 5, 2016, Art. no. 051001.

[32] B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J.-M. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. A. Merolla, and K. Boahen, "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations," *Proc. IEEE*, vol. 102, no. 5, pp. 699–716, May 2014.

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

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

[35] C. Zamarreno-Ramos, A. Linares-Barranco, T. Serrano-Gotarredona, and B. Linares-Barranco, "Multicasting mesh AER: A scalable assembly approach for reconfigurable neuromorphic structured AER systems. Application to convNets," *IEEE Trans. Biomed. Circuits Syst.*, vol. 7, no. 1, pp. 82–102, Feb. 2013.

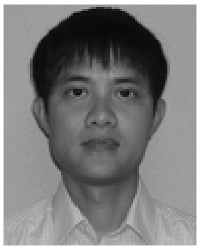
[36] J. Schemmel, J. Fieres, and K. Meier, "Wafer-scale integration of analog neural networks," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN), IEEE World Congr. Comput. Intell.*, Jun. 2008, pp. 431–438. doi: 10.1109/IJCNN.2008.4633828..

[37] S. Pande, F. Morgan, G. Smit, T. Bruintjes, J. Rutgers, B. McGinley, S. Cawley, J. Harkin, and L. McDaid, "Fixed latency on-chip interconnect for hardware spiking neural network architectures," *Parallel Comput.*, vol. 39, no. 9, pp. 357–371, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167819113000549

[38] B. Belhadj, A. Valentian, P. Vivet, M. Duranton, L. He, and O. Temam, "The improbable but highly appropriate marriage of 3D stacking and neuromorphic accelerators," in *Proc. Int. Conf. Compil. Archit. Synth. Embedded Syst. (CASES)*, Oct. 2014, pp. 1–9.

[39] M. A. Ehsan, Z. Zhou, and Y. Yi, "Modeling and analysis of neuronal membrane electrical activities in 3D neuromorphic computing system," in *Proc. IEEE Int. Symp. Electromagn. Compat. Signal/Power Integrity (EMCSI)*, Aug. 2017, pp. 745–750.

[40] D. Xiang and K. Shen, "A new unicast-based multicast scheme for network-on-chip router and interconnect testing," *ACM Trans. Des. Automat. Electron. Syst.*, vol. 21, no. 2, p. 24, Jan. 2016.
[41] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, J. Flich, and H. Tenhunen, "Path-based partitioning methods for 3D networks-on-chip with minimal adaptive routing," *IEEE Trans. Comput.*, vol. 63, no. 3, pp. 718–733, Mar. 2014.
[42] F. A. Samman, T. Hollstein, and M. Glesner, "Adaptive and deadlock-free tree-based multicast routing for networks-on-chip," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 7, pp. 1067–1080, Jul. 2010.
[43] M. Naeem, L. J. McDaid, J. Harkin, J. J. Wade, and J. Marsland, "On the role of astroglial syncytia in self-repairing spiking neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 10, pp. 2370–2380, Oct. 2015.
[44] J. Liu, J. Harkin, L. P. Maguire, L. J. Mcdaid, and J. J. Wade, "SPANNER: A self-repairing spiking neural network hardware architecture," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 4, pp. 1287–1300, Apr. 2018.
[45] D. Vainbrand and R. Ginosar, "Network-on-chip architectures for neural networks," in *Proc. NOCS*, May 2010, pp. 135–144.
[46] R. Rojas, *Neural Networks: A Systematic Introduction*. Berlin, Germany: Springer-Verlag, 1996.
[47] R. Legenstein and W. Maass, "Edge of chaos and prediction of computational performance for neural circuit models," *Neural Netw.*, vol. 20, no. 3, pp. 323–334, 2007.
[48] J. Milton, *Dynamics of Small Neural Populations*. Providence, RI, USA: AMS, 1996, ch. 8.
[49] B. Parhami, "Exact formulas for the average internode distance in mesh and binary tree networks," *Comput. Sci. Inf. Technol.*, vol. 1, no. 2, pp. 165–168, 2013.
[50] C. Frenkel, M. Lefebvre, J.-D. Legat, and D. Bol, "A 0.086-mm^2 12.7-pJ/SOP 64k-synapse 256-neuron online-learning digital spiking neuromorphic processor in 28-nm CMOS," *IEEE Trans. Biomed. Circuits Syst.*, vol. 13, no. 1, pp. 145–158, Feb. 2019.
[51] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *J. Physiol.*, vol. 117, no. 4, pp. 500–544, Aug. 1952. [Online]. Available: http://jp.physoc.org/content/117/4/500.abstract
[52] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Trans. Neural Netw.*, vol. 14, no. 6, pp. 1569–1572, Nov. 2003.
[53] A. B. Ahmed and A. B. Abdallah, "Adaptive fault-tolerant architecture and routing algorithm for reliable many-core 3D-NoC systems," *J. Parallel Distrib. Comput.*, vols. 93–94, pp. 30–43, Jul. 2016.
[54] K. N. Dang, M. Meyer, Y. Okuyama, and A. B. Abdallah, "Reliability assessment and quantitative evaluation of soft-error resilient 3D network-on-chip systems," in *Proc. IEEE 25th Asian Test Symp. (ATS)*, Nov. 2016, pp. 161–166.
[55] C.-H. Chao, K.-Y. Jheng, H.-Y. Wang, J.-C. Wu, and A.-Y. Wu, "Traffic- and thermal-aware run-time thermal management scheme for 3D NoC systems," in *Proc. 4th ACM/IEEE Int. Symp. Netw. Chip*, Washington, DC, USA, May 2010, pp. 223–230. doi: 10.1109/NOCS.2010.32.
[56] S. Cawley, F. Morgan, B. Mcginley, S. Pande, L. Mcdaid, S. Carrillo, and J. Harkin, "Hardware spiking neural network prototyping and application," *Genetic Program. Evolvable Mach.*, vol. 12, no. 3, pp. 257–280, Sep. 2011.
[57] E. Pasero and M. Perri, "Hw-Sw codesign of a flexible neural controller through a FPGA-based neural network programmed in VHDL," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Jul. 2004, vol. 4, pp. 3161–3165.
[58] H.-T. Vu, Y. Okuyama, and A. B. Abdallah, "Analytical performance assessment and high-throughput low-latency spike routing algorithm for spiking neural network systems," *J. Supercomput.*, to be published. doi: 10.1007/s11227-019-02792-y.
[59] Y. Ji, Y. Zhang, H. Liu, and W. Zheng, "Optimized mapping spiking neural networks onto network-on-chip," in *Algorithms and Architectures for Parallel Processing*. Springer, 2016, pp. 38–52.
[60] W. Gerstner and W. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge, U.K.: Cambridge Univ. Press, 2002.
[61] N. Akbari and M. Modarressi, "A high-performance network-on-chip topology for neuromorphic architectures," in *Proc. IEEE Int. Conf. Comput. Sci. Eng. (CSE) IEEE Int. Conf. Embedded Ubiquitous Comput. (EUC)*, vol. 2, Jul. 2017, pp. 9–16.
[62] Y. Dong, C. Li, Z. Lin, and T. Watanabe, "Multiple network-on-chip model for high performance neural network," *J. Semicond. Technol. Sci.*, vol. 10, no. 1, pp. 28–36, 2010.



THE H. VU received the B.S. degree in electronics engineering from the Hung Yen University of Technology and Education, in 2009, and the M.S. degree in electronics engineering from the Hanoi University of Science and Technology, in 2013. He is currently pursuing the Ph.D. degree with the Adaptive Systems Laboratory, The University of Aizu, Aizuwakamatsu, Japan. His current research interests include adaptive deep neuro-inspired computing systems for pattern recognition and network-on-chip.

OGBODO MARK IKECHUKWU received the master's degree in computer science from the African University of Science and Technology (AUST). He is currently pursuing the Ph.D. degree with the Adaptive Systems Laboratory (ASL), The University of Aizu, where he is also a member. His current research interest includes neuro-inspired architectures in hardware.

ABDERAZEK BEN ABDALLAH received the Ph.D. degree in computer engineering from The University of Electro-Communications, Tokyo, in 2002. He was a Research Associate with the Graduate School of Informatics and Engineering, The University of Electro-Communications, from 2002 to 2007. He held several visiting professorships at several universities, including The Hong Kong University of Science and Technology, Hong Kong, from 2010 to 2013, and the Huazhong University of Science and Technology, Wuhan, from 2010 to 2016. He has been a Full Professor of computer science and engineering and the Head of the Division of Computer Engineering, The University of Aizu, Aizuwakamatsu, Japan, since 2014. He has also been a Faculty Member with The University of Aizu, since 2007. His current research interests include computer architecture, adaptive/self-organizing systems, power and reliability-aware architectures, network on chips (photonic, hybrid), fault-tolerance, and neuro-inspired/neuromorphic computing. Most recently, his research group has been building scalable, large-scale, and energy-efficient neuromorphic processing systems using application-specific hardware and embedded reconfigurable systems. He has authored three books, published more than 150 journal articles and conference papers, and holds several patents in these areas. He is a Senior Member of ACM and IEEE.