Predicting Networks-on-Chip Traffic Congestion with Spiking Neural Networks

Aqib Javed, Jim Harkin, Liam McDaid, Junxiu Liu
School of Computing, Engineering and Intelligent Systems,
Ulster University, Magee Campus, Derry, Northern Ireland, United Kingdom

Abstract— **Network congestion is one of the critical reasons for degradation of data throughput performance in Networks-on-Chip (NoCs), with delays caused by data-buffer queuing in routers. Local buffer or router congestion impacts on network performance as it gradually spreads to neighbouring routers and beyond. In this paper, we propose a novel approach to NoC traffic prediction using Spiking Neural Networks (SNNs) and focus on predicting local router congestion so as to minimise its impact on the overall NoCs throughput. The key novelty is utilising SNNs to recognise temporal patterns from NoC router buffers and predicting traffic hotspots. We investigate two neural models, Leaky Integrate and Fire (LIF) and Spike Response Model (SRM) to check performance in term of prediction coverage. Results on prediction accuracy and precision are reported using a synthetic and real-time multimedia applications with simulation results of the LIF based predictor providing an average accuracy of 88.28%-96.25% and precision of 82.09%-96.73% as compared to 85.25%-95.69% accuracy and 73% and 98.48% precision performance of SRM based model when looking at congestion formations 30 clock cycles in advance of the actual hotspot occurrence.**

Keywords— **Networks-on-Chip; congestion prediction; network traffic; Spiking Neural Networks**

I. Introduction

The downsizing of the transistor has enabled many processing cores to be integrated on a single chip to form a scalable and parallel multi-processor System-on-Chip [1]. These devices are highly dependent on resource utilization and system reliability as they exploit parallel computation for performance gains and require scalable, high interconnect bandwidth between processor cores [2]. In particular, for multi-processor systems the interconnect challenges includes communication topology, routing schemes, arbitration, switching, and flow control [3]. Networks-on-Chip (NoC) has been proposed [4] to address these communication challenges where information is communicated as small packets of data. The NoC provides high bandwidth and concurrent communication channels for transferring of data between multiple cores [5]. NoCs face congestion problems due to uneven workload distribution across many routers [3] leading to hotspots of congestion. The Quality of Service (QoS) of a NoC is a measurement of run time performance. One promising solution to maximise the QoS is adaptive routing algorithms which aim to balance the traffic congestion across many routers in the NoC [6].

To date neural networks have shown success in the domains of pattern recognition and prediction [7] and in particular, Spiking Neural Networks (SNNs) can now be implemented in hardware [8-9] with low area overheads. Therefore, neural networks can be explored as a mechanism for prediction of NoC traffic congestions, i.e., identify hotspots before they occur. In this paper, the temporal nature of SNNs is harnessed for hotspot prediction in NoCs. The SNN training algorithm of SpikeProp with Leaky Integrate and Fire (LIF) and also Spike Response Model (SRM) neuron models are explored in the prediction of router congestion. The ultimate goal of the proposed work is to use a hardware scalable SNNs to predict network congestion and support a congestion handling scheme to minimise router traffic unbalance.

Major contributions of this paper include:

- Novel use of SNNs in the prediction of congestion in NoC architectures.
- Simulation and evaluation of two SNN models to evaluate prediction coverage on NoC architecture.
- Validation of results and performance of SNN models under different traffic applications generated by Noxim simulator.

Section II provides background on SNN networks and existing approaches. Section III reports on the SNN-based NoC hotspot prediction methodology, and the experimental setup is outlined in section IV. Section V present simulation results and section VI provides a conclusion and outlines future work.

II. Preliminaries

This section gives a brief introduction to the cause and effects of congestion on NoC architectures and also presents an overview of existing research on congestion prediction.

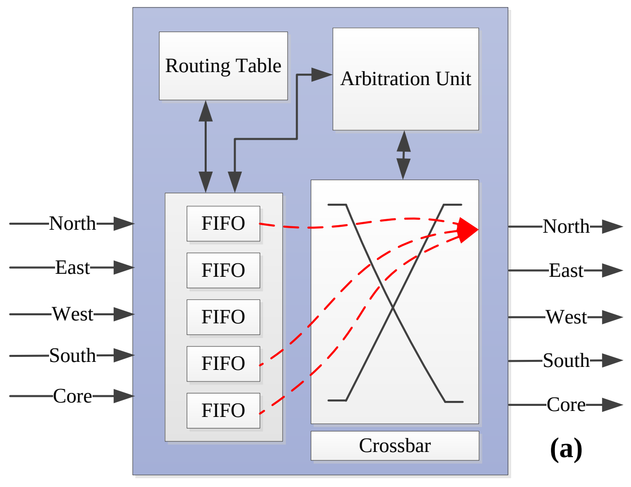
A. Congestion and Effects on NoC Performance

In NoC architectures, routing switches and network channels are susceptible to congestion due to the unbalanced load distribution caused by executing applications[6]. Different routing algorithms i.e., XY, Even-Odd etc. are employed to overcome traffic unbalancing problem in on-chip interconnects. Goal of routing algorithms are to distribute load across all available routing paths. Routing algorithms can be classified as Deterministic and Adaptive where Deterministic algorithms follow a fixed path from source to destination node. Adaptive algorithms can change routing path depending on current network conditions. These algorithms mostly use buffer information to identify on-path congestion where routers have limited numbers of available buffers to compensate unbalanced traffic flows. One promising solution is to add additional buffers on input and output ports of routers. These buffers can avoid the backpressure effect by queuing more input packets and stopping the spread of congestion towards neighbouring nodes however queuing will cost further delays in packet transmission that will also affect overall system performance (latency and throughput). The congestion problem will still persist if these buffers are filled [10], [11]."

Consider that a 4-input buffer NoC router (router-X) encounters with burst traffic. The neighbouring/on-path routers are equipped with an adaptive routing algorithm which can react to congested nodes and re-route data through alternative path. Hence the data packet is routed towards the destination node thus keeping the flow away from the congested node. The data packets queued at input buffers adds to the network delay as these packets are soaked out of network thus affecting network throughput.

Now, consider a router-X with 6-input buffers to provide more space to incoming data packets. The additional buffers will compensate the formation and spread of congestion towards neighbouring node. As the adaptive routing algorithm will have available spaces in router-X therefore data packets will be forwarded towards router-X. As the router is encountered with burst traffic it may soon run-out of buffers. The data packets queued in the additional buffers have to wait more than usual before getting forwarded by the router-X. These additional packet delays will add further to the network delay. As more packets are soaked out of the network this will further affect network throughout. .

Path congestion occurs in three phases [12] including 1). *Switch contention inside router*. If a number of packets from different input channels compete for the same output channel, switch contention will occur (Shown in Fig. 1(a)); 2). *Router congestion*. Several inputs from multiple channels are competing to connect with same output channel, only one



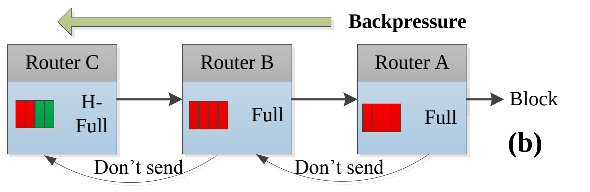


Fig. 1 (a). Illustration of switch contention (dashed lines show multiple switch request to the North output). (b). Effect of congestion and Backpressure.

channel will get access to output channel and rest of packets are queued at input buffers of their respective channels: and 3). *Channel congestion*. As input buffers have limited data storage capacity and may run out of space on arriving of new data. Queue of packets causes back pressure to neighbour routers and data starts queuing in buffers of neighbour nodes. This back pressure untimely leading to cluster or network congestion as shown in Fig 1(b).

Network topology, arbitration, switching and data flow mechanisms must be considered to avoid NoC congestion[3]. Buffer, links and status register are on-chip resources of NoC architecture and flow control mechanisms use these resources to minimize transmission delay[13]. Nodes with higher routing loads are more susceptible to congestion[3], [10]. Mostly, NoC architecture are employed with Warm-hole Flow Control (WFC) mechanism to packetize data into small chunks (called flits: header, body and tail) and transmit them in a pipelined form. Routing functions establish set of routing channels as soon as it receives header flit. Body flits followed by tail flit routed on same channels. As tail flit leave the router, routing function cancel the reservation and reallocate it to next on queue header flit. Unfortunately, blockage of header flit along established path cause body and tail flits to wait on host router [14]. Queuing of packetized data leads router into congestion that will ultimately cause back pressure to neighbour routers. Virtual Channels (VC) can be used with buffers to minimize effects of flow control mechanisms.

Recent studies [3], [12], [15] showed that local congestion has a significant impact on network performance. Data transmission between source to destination nodes can suffer with high latency if congestion is encountered. The proposed work focuses on the router congestion to avoid local congestion and restrict the spreading of congestion to neighbouring routers.

B. Related work

Adaptive routing algorithms with local or global information require congestion information across all nodes to make decisions. Depending on path diversity, adaptive algorithms re-route data packets around congestion nodes or flatten the distribution of traffic among the links. The key inhibitor in an adaptive routing algorithm is its need to have local and global network information, which is difficult to obtain without increased area and delays in decision making.

Regular mesh NoCs use a static two-dimensional routing algorithm called XY routing [16]. In XY, data travels in the X-dimension before moving in the Y-dimension while trans-versing from source to destination nodes. A congestion aware adaptive XY routing (DyXY) [17] approach was proposed to improve XY latency problems. Depending on the neighbour congestion status, DyXY transmits data in either X or Y directions. The Odd-Even (OE) routing algorithm [18] is proposed to avoid dead-lock caused in XY routing. DyAD [19] is a hybrid routing algorithm which gives both static and adaptive routing conditions. Depending on path congestion, DyAD dynamically switch between routing algorithms. If path is clear then the OE (static) routing algorithm will be deployed to transmit data, if the path is congested then the odd-even turn (adaptive) algorithm will be used to transmit data towards destination node. A routing controller in adaptive routing algorithms determine the minimal path to optimize NoC performance. Therefore overall performance of routing algorithms are dependent on the routing controller [15]. Most routing algorithms monitor local traffic to identify on path congestion and re-route data through less congested node. This approach uses congestion information of the neighbouring routers to identify and forward data packets through least congested on-path nodes. These routing algorithms have misjudgement problems [20] as they forward data packets towards congested nodes. These queued data packets add into the spread of congestion across non-congested neighbouring nodes, thus affecting network performance by increasing the packet latency and decreasing network throughput [3], [20].

Congestion-Aware Adaptive Routing (CAAR) methods [15] are key research areas in tackling NoC congestion. CAAR use selection functions to react to congestion. Output buffer length (OBL) selection functions [14] use buffer occupancy information of downstream routers to make decisions. Selection strategies based on Neighbours-on-Path concepts use free slot information of all neighbours to select next hop. Some selection functions use switch contention information as an additional metric for decision purpose. A Path-Congestion Aware Adaptive Routing (PCAR) algorithm use free buffer slots and crossbar demand as metrics to a selection function. An Odd-Even turn model for mesh platform with build-in guaranteeing QoS arbitration use buffer information and switch values to identify congestion-free minimal path[15].

All these works are reactive to congestion and use real-time network values i.e. available buffer slots, crossbar demand etc. to calculate on path congestion. These congestion-aware techniques are helpful in reducing the effect of an already occurred congestion, via bypassing the flow of data through other available, less congested channels. Hence, minimizing the workload of overburdened routers and therefore enhancing overall NoC efficiency. These CAAR algorithms have no prior information about congestion until they reach close to the congested node. NoC efficiency can be increased further if congestion in a network or node is predicted before arrival of data packets close to congested nodes. It provides enough time for a routing node to react on potential congestion and re-route data through alternative paths. This advanced prediction and data re-routing technique reduces the probability of predicted nodes entering into a congestion state, thus improving overall network throughput [13].

Work presented in [21] used a low power application-driven traffic pattern table for predicting end to end traffic to dynamically adjust the working frequencies/voltages of the routers in NoC. It achieves 86% dynamic power savings in NoC links with 21% packet latency overheads. Other solutions include the execution of task mapping at run time, to meet real-time traffic constraints [22]. However due to time variation in application behaviour, task mapping has limited effort on improving the latency performance [23].

i. Neural Network and Prediction:

Neural Networks (NNs) comprise of neurons (computational nodes) and synapses (communication links) to classify nonlinear and dynamic behaviours of systems. Depending on type of neural architecture and mathematical modelling, NNs can be classified into different categories. Neurons in Artificial Neural Network (ANN) are organized in different topologies, usually in the form of layers, where information processed in each neuron layer is transferred to subsequent layers and collated an output layer. Neurons are modelled by an activation function that is applied when a change in neuron input occurs [24]. Comparably to synapses in biological neurons, ANN neurons are connected through synaptic weights and ANNs use learning algorithms i.e., backpropagation to update synaptic weights to establish a relationship between connected (input and output) neurons [25]. To date neural networks have shown improvement in the domains of pattern recognition and prediction [26][27][28][29]. Therefore, neural networks will be explored as a mechanism for detection and prediction of network traffic congestion patterns in real-time applications.

An Evolving Fuzzy Neural Network (EFuNN) based low latency path prediction algorithm is proposed in [30] to minimize latency issues caused by congestion-aware routing algorithms. The proposed work showed slight improvement in routing latency by incorporating existing congestion information however, the model failed to predict on path congestion.

An ANN based hotspot prediction mechanism was previously presented in [10] that monitored online statistical data to predict hotspot interconnect with an average efficiency of between 62-92%. However, it is not scalable with high latency and hardware overheads. In the approach of [31], a two layered neural network technique is demonstrated which is able to detect and re-route data under congestion. It uses the hamming network to compute the link buffer utilization and then uses the competitive network to find the worst congestion node. Throughput was increased by 15%, however it does not explore the prediction of congestion.

These approaches used ANNs to predict congestion and to the best of the authors' knowledge none have considered SNNs in classifying temporal NoC traffic patterns for prediction of congestion.

SNNs are third generation NNs and claimed as more suitable learning tools for processing spatiotemporal information [32]. SNNs take rate encoded inputs as spikes and process encoded information between the input and output layers of the neural network in a temporal format. In ANNs, neural layers are updated with inputs in a single time step, whereas in SNNs inputs are applied in temporal form over a time series of steps. The key advantage is that SNNs exhibit low hardware area and power overheads compared to ANNs as the core neural computations are different [33].

NoC architectures process digital data and the traffic patterns that flow between NoC are temporal/discrete in nature. Therefore, this work proposed SNN based prediction model that process temporal NoC traffic patterns to predict local congestion.

ii. SNN Model

SNNs closely mimic the operations of the biological nervous systems by incorporating different neural information dimensions such as time, frequency, and phase [34][35]. Depending on neural complexity, different SNN models are proposed to present more biologically realistic neurons [9]. This work explores Leaky Integrate and Fire (LIF) neurons & Spike Response Model (SRM) neurons to model neural networks.

LIF Neural Model: The LIF neuron model is effective in modelling biological neurons for simulating spiking networks

[36] because it provides good trade-off between computational complexity and hardware area cost. LIF neurons are driven by exponentially decaying synaptic currents generated by its synaptic afferents[37]. It integrates all synaptic input currents to produce a spike after it reaches a certain threshold [38]. The neuron membrane potential $u_i(t)$ of ith neuron at time of $'t'$ leaks out to a resting potential in the absence of an input current. It is described by

$$\tau_m \frac{d(u_i)}{dt} = -u_i(t) + RI_i^{syn}(t), \tag{1}$$

where τ_m describes the membrane time constant of the neuron, R is the membrane resistance and $I_i^{syn}(t)$ is synaptic current of ith neuron. On arrival of each spike, membrane potential changes. When it crosses certain threshold, neuron fires a spike and goes to a refractory period.

SRM Neural Model: Spike response model is generalized variation of LIF neuron model. Neuron membrane potential excitation and fire are explicitly relay on time of the last spike.[39] The membrane potential 'u' of cell 'i' at time 't', $u_i(t)$ is defined as:

$$u_i(t) = \eta(t - \dot{t}) + \int_0^{\infty} \varepsilon(t - \dot{t}, s) I(t - s) ds, \tag{2}$$

where ε (also called linear filter of membrane) is linear response to input current $I(t - s)$. '$\dot{t}$' is the time of last spike of neuron i. The time dependency in membrane potential enables the refractoriness in neuron. Kernel 'η' is response of neuron to its own spike.

Spikeprop-Learning Algorithm: In past few decades different learning algorithms for SNNs have been proposed to impose a precise input-output mapping of spike trains to SNNs. Bothe's SpikeProp [40] is based on arithmetic calculations and the concept is very similar to the back propagation algorithm for ANNs. SpikeProp learns and updates the cost function in terms of the difference between actual firing times t_j^a at output neuron j and the desired firing time t_j^d at output neuron j by

$$E = \frac{1}{2} \sum_{j \in J} (t_j^a - t_j^d)^2. \tag{3}$$

To minimize the squared difference E, the weight w_{ij}^k of each separate connection k of the synaptic terminal between pre-synaptic input of neuron i to neuron j should be calculated by

$$\Delta w_{ij}^k = -\eta \frac{\partial E}{\partial w_{ij}^k} \tag{4}$$

where η is learning rate of the network.

This work focuses on identifying patterns that leads to congestion and evaluates prediction capability of the proposed SNN for NoC architecture on synthetic and multimedia applications.

III. SNN-BASED NOC TRAFFIC HOTSPOT PREDICTION

This paper proposes the novel use of an SNN in predicting NoC traffic congestion. This work proposed two congestion prediction models based on LIF and SRM neurons. Both models are integrated with SpikeProp learning algorithm to update synaptic weights between spiking neurons. This section discusses proposed prediction methodology and congestion criteria to identify and predict local congestion in NoC architecture.

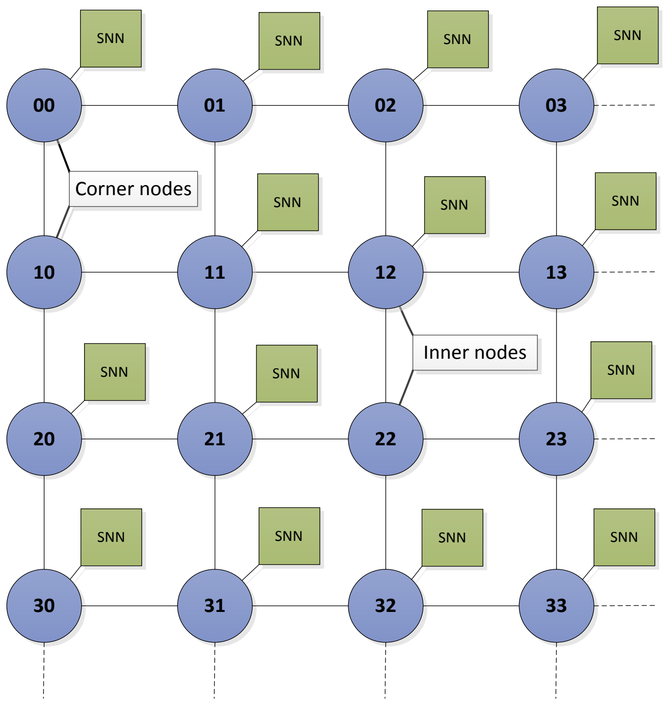


Fig. 2. 4x4 NoC integrated with proposed SNN based congestion prediction

A. Proposed Prediction Methodology

To design an SNN based prediction methodology for NoC architectures, we considered all factors that leads towards the formation of congestion. Local congestion does not happen immediately but sources from an overburden node and accumulates over time. As more data is pushed towards an overloaded node, it starts to queue data in the input buffers and starts to effect network throughput. When this router enters congestion there are often many routers which are not congested. Due to the process of backpressure, these uncongested nodes begin to enter into a congestion state. NoC traffic is highly dependent on following three factors: (i) location of the router, (ii) application running on the system and (iii) Packet Injection Rate (PIR). The approach of [10] investigated the effeteness of buffer utilization data in determining NoC behaviour especially for congestion prediction.

This work considered router input buffer utilization as a congestion prediction parameter pattern. Every router has a different buffer utilization pattern depending on the mapped application therefore, every router will be treated independently to identify congestion. We propose a layer of SNN over the routing nodes where each router is connected with its own SNN. This model will help to identity the location of a router and its congestion status. SNNs takes buffer utilization as an input and gives predictive congestion status as an output. The proposed prediction model is scalable with network sizes, traffic scenarios, topologies and NoC simulator as each node requires its own SNN that is responsible for prediction of local congestion.

Routing algorithms are responsible for data transmission from source to destination nodes. Adaptive routing algorithms and CAAR algorithms are only able to see congestion in their neighbouring nodes (discussed in Section II) and often suffered with misjudgement issues[20]. These routing algorithms gets

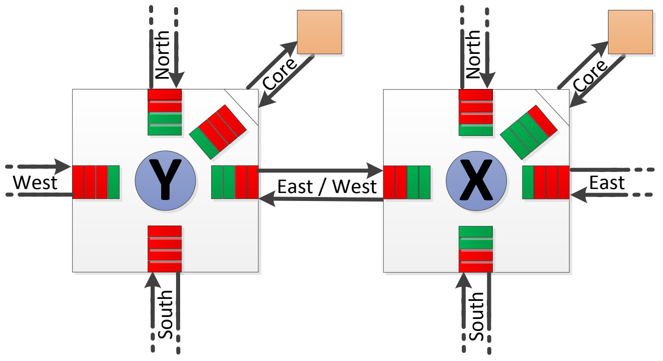


Fig. 3. Buffer Utilization model with 4-buffer slots for each input (Green are free slots; Red are occupied slots)

the opportunity to use prediction information to re-route the data through alternative nodes and hence preventing the node to enter into congestion state without effecting network throughput. Using the hotspot status of all routers and traffic patterns we may predict the congestion of whole network however this is not the scope of this initial work.

Routers can be classified based on spatial location such as 1). Inner router and 2). Corner router. Due to uneven traffic distribution, Inner routers are more prone to network congestion as compared to corner routers [3]. Fig. 2 depicts an example 4x4 NoC architecture, where Router ID# [11, 12, 21, 22 etc.] are inner routers and Router ID# [00, 10, 20, 30 etc.] are corner routers. Routers are interconnected with bi-directional channels through buffers. Buffer utilization data will be extracted from these buffers of each router for training and validation of SNN to predict NoC congestion.

B. *Congestion Criteria.*

Network congestion is one of the critical reasons for degradation of data throughput performance in NoCs. Congestion occurs when a network fails to deliver packets from source to destination within a predefined time (time required for data from the source node towards destination node i.e., latency). Data queues in input buffers are the foremost cause of delays that effects network latency and throughput performance. Congestion adds delays in packet re-transmission which impacts predefined transmission time/network latency.Buffer Utilization is an important NoC performance parameter [41], [10] and have been demonstrated as a most effective parameter in identifying congestion. The proposed congestion prediction criteria is based on overall input buffer utilization. For example, the router shown in Fig. 3 has five inputs (North, East, South, West and Local Interface). Each input buffer can store up to 4 data packets, i.e. for five inputs there are a total of 20 data slots.

For any given router, if the buffer slots of all input channels are 50% or more occupied by queued data, with at least one fully occupied channel (full channel=4), then the router will be considered as congested. Therefore, for any given traffic scenario and selection strategy, utilization patterns that contains one fully occupied input channel with overall occupancy level of 50% or more, are defined as *congestion patterns*, and those patterns that produce occupancy level less than 50% are considered as *non-congestion patterns*. Compared to previous congestion criteria in [10], [42], the proposed congestion criteria labels each congestion inducing pattern irrespective of traffic distribution, injection rate, topology and network size.

The hardware implementation of the SNN prediction model manages NoC congestion and that requires time to process utilization information in order to make a prediction on local congestion and to re-route data through alternative paths to avoid the potential congestion hotspot. The SNN requires 5-7 clock cycles to predict a router status and also requires additional clock cycles to forward information towards the congestion handling mechanisms. The congestion handling mechanism consumes 3-4 clock cycles to process congestion information and make an alternative routing path decision.

Defining prediction time (in clock cycles) is a critical element in the design of the prediction algorithm as it should not be significant in size whereby an incorrect alternative routing decision is made that causes delays in packet latency. At the same time, prediction time should not be too short such that it does not provide sufficient time for the congestion handing mechanism to avoid the formation of potential hazard [10].

Therefore, from early experimentation we considered an optimised 30 clocks look-ahead benchmark to predict local congestion and forward prediction information towards the congestion handling mechanism for suppression of any negative impact. In this work it is assumed that the proposed SNN predictor will predict congestion 30 clock cycles in advance to provide enough time for the congestion handling algorithm to react on it.

C. *Prediction Model:*

NoC router throughput is measured in term of the number of flits per clock cycle. Local congestion drastically affect network throughput by forcing routers to queue data at input buffers[3]. Using both synthetic and multi-media application traces, we have collected buffer utilization levels of all network routers. Data extracted from each router is used for training and validation of its connected SNN predictor (as shown in Fig.4). To evaluate our approach, synthetic traffic patterns and a multimedia application were run independently on the NoC simulator [42] to generate utilization data for each router. Every node has its own SNN to predict local congestion (discussed in section III.A). Therefore, for each traffic trace, buffer utilization patterns of each node will be fed into its own SNN for training and validation. The PIR was increased to generate congestion in NoC architecture.

As explained in section II, a single congested router may affect performance of the whole network by forcing its neighbouring routers to enter into a congestion state. By monitoring the router input buffer occupancy levels, the spiking neural network will be able to identify and predict potential hotpots caused by either back pressure of neighbour routers or by high traffic flow/loads.

The scope of this work is limited to evaluating the prediction coverage of the proposed (LIF and SRM based) SNN models under various traffic conditions, and to identify an efficient solution based on prediction performance. The SNN model with the higher congestion prediction coverage will be integrated inside each NoC router to minimize the impact of potential congestion and thus improves network throughput and latency.

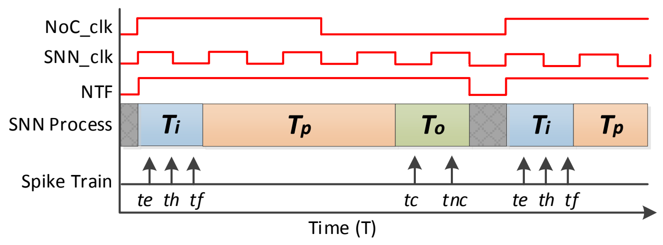


Fig. 5. Timing diagram of SNN for encoding and transmission of temporal information.

D. *SNN Encoding*

SNN encoding depends on the temporal pattern at which neurons generate spikes. Spiking neural models work on spike arrival time and encode information in the spatial domain. SNN neurons read spikes over specific periods of time before reacting on it. This work used SpikeProp as a learning algorithm that works on spike propagation from the input layer to the output layer. So, neurons in the SNN layer will read the time of the input spike, process it through a hidden layer and then produce a spike to the output layer. An example timing diagram of the proposed prediction method is shown in Fig. 5. Neural Time Frame (NTF) is defined as time required by SNN to process information from input layer neuron till generation of spike at output layer neurons. NTF is used for monitoring spike patterns and at the end of the fixed time, the NTF window provides time to spiking neurons to undergo through refractory period. We assumed that SNN runs at higher frequency (SNN_clk) to generate output spike with in one NoC clock cycle (NoC_clk). As spikes are based on current buffer levels, each neuron will fire a spike based on its current occupancy level. In this approach, it is assumed that each input buffer has four available slots to pipeline incoming data and spike time associated to them are t_0,t_1,t_2,t_3 and t_4 respectively, where t_0 depicts all slots are empty and t_4 reflects all slots are occupied. Also T_i is the maximum time at which a neuron fires a spike; after that time, it is reset to zero. The SNN processes the data in time T_p and gives an output in time T_o (T_p+T_i <T_o<t_m), where t_m is the maximum output spike time. The output of the SNN indicates a router status as either Congested (t_c) or un-congested (t_{nc}). To minimize latency and sufficiently predict ahead of the time of congestion occurring, T_p must be small in duration. During simulations, SNN models with different parameters were explored to minimize overall processing time; e.g. the number of hidden layers and number of neurons in each layer that not only decreases training time but also improves the prediction capability of proposed model.

IV. Experimental Setup

This section presents the experimental setup established to generate utilization data from NoC architectures and the use of data in SNN prediction of congestion hotspots. Synthetic and multimedia application traffic patterns are used as target applications. These applications are mapped to the network simulator to generate buffer utilization data for each router. PIR is increased to generate router congestion. Datasets with predefined congestion values are used for training of proposed prediction models. NoC architectures use a random selection strategy with an XY routing algorithm which enables each router to send data randomly hence creating the probability of congestion at different NoC nodes.

A. Simulation Setup

Experiments were carried out using synthetic and multimedia application data traces for checking the effectiveness of the proposed SNN predictor. These traces contained mapping information for each NoC node (e.g., source node, destination node and PIR) to generate network traffic. Inner nodes are more prone to congestion therefore this work utilised a 4x4 2D mesh topology with random selection strategy that provides enough inner and corner hotspot nodes (shown in Fig.4) to analyse prediction coverage of the proposed prediction model under different traffic patterns. Depending on the location of the router, each router is connected between 2-4 neighbouring routers through network channels. We used the Noxim simulator [43] which is an open source cycle accurate simulator to extract buffer utilization data for the SNN training and validation. All routers have 2 VCs per input port, and each input has 4 buffering slots. Parameters used for Noxim are defined in table 1. Simulations were run for 1,000 clocks to generate buffer utilization data. Input buffer patterns that lead to congestion after 30 clocks cycles were defined as congestion occurring patterns, and patterns which didn't transform into congestion after 30 cycles are noted as un-congested patterns. Collected data was divided (60:40) for training and validation purposes, respectively.

TABLE 1 NOXIM CONFIGURATION PARAMETERS

Simulator	Noxim
Topology	Mesh
Network Size	4x4
Arbitration	Round Robin
Selection Strategy	Random
Packet Size	Two Flits
Traffic Distribution	Synthetic, Multimedia
Routing Algorithm	XY routing
Buffer Depth	8 Flits / 4 Packet Slots
Flit Size	32-Bit

It is assumed that buffer information for each input will arrive as an individual packet and will be updated after a fixed time. As the SNN predictor receives inputs as temporal patterns, buffer levels are encoded as a spike time of input to a neuron. The time required for the SNN to process input patterns and

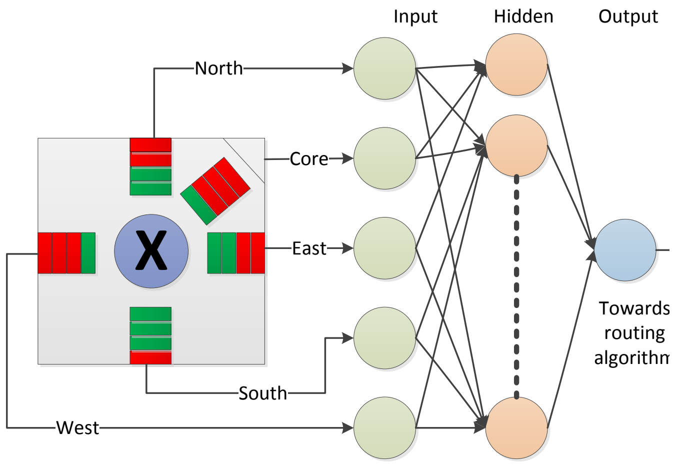


Fig. 4. Connections between router input channels and proposed SNN based congestion prediction model.

generate an output is termed as a processing time, as noted in Fig. 5.

i. Traffic Scenarios

To check the effectiveness of the proposed SNN predictors we considered two traffic scenarios: a). Synthetic traffic scenario and b) real-time application scenario.

a. Synthetic Traffic Scenarios

The Noxim simulator provides built-in synthesis traces to evaluate network performance under varied traffic loads. These traces include transpose1, transpose 2, butterfly and shuffle. A 4x4 2D mesh topology (shown in Fig.2) is used as the structure for performance evaluation of the synthetic traffic.

Transpose traffic: In transpose1, router (x, y)sends packets to $(N-1-x, M-1-y)$, where N and M represents size of mesh along horizontal and vertical axis respectively.

Transpose2 traffic: allows a router (x, y) to communicate with router (y, x). Transpose1 and transpose2 are symmetric and transpose to each other.

Butterfly traffic: Packets are sent using $(k-ary\ n-fly\colon kn)$ configuration, where n is used to direct output stages and k is the radix of each switch.

Shuffle traffic: this traffic send packets from (x, y) to $((N+x-1) mod\ N,\ (N+y-1) mod\ N\ )$.

b. Real-Time Multi-Media Traffic Scenarios

We considered a Multimedia System (MMS) and Moving Picture Experts Group-4 (MPEG-4) application traces to analyse performance on real-time multi-media traffic scenarios. MMS and MPEG-4 application traces (source node, destination node and PIR) are mapped into the Noxim simulator. More detail on extraction of real-time multi-media traffic traces to map on 2D NoC are discussed in [44]. Once the applications are mapped, data packets are traversed through 2D mesh using routing algorithms and a selection strategy (Table 1) to generate utilization datasets.

MultiMedia System (MMS): A generic *MMS* benchmark consisting of the H.263 video encoder-decoder and mp3 audio encoder-decoder were designed in [10] to evaluate performance of heterogeneous architectures. The MMS application includes 40 tasks that are scheduled across NoC nodes using mapping information (traces) generated using [45], [46]. The MMS traces are mapped on 4x4 NoC using Noxim.

Moving Picture Experts Group-4 (MPEG-4): MPEG-4 is a widely used standard video decoder application to benchmark system performance [47]–[49]. MPEG4 application traces mimic the communication of interactive audio-visual scenes between nodes and they are generated using Mesh based On-Chip interconnection Architectures (MOCA) in [44]. Using the Noxim simulator, these traces are mapped across NoC nodes to generate buffer utilization patterns.

ii. Hotspot Generation

Routers are more likely to get congested under high traffic loads. A small increment in PIR increases the amount of data in the NoC channels hence increases network latency. Fig. 6 shows the effect of PIR on network latency using synthetic and multi-

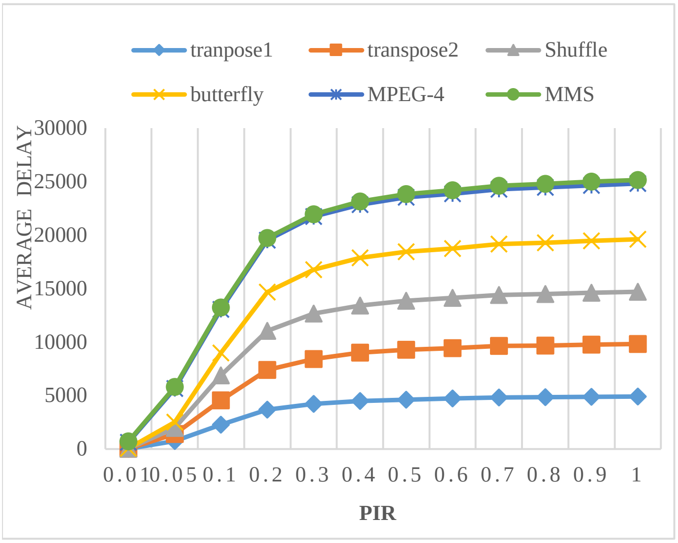


Fig. 6. Average delays under different PIRs and traffic traces

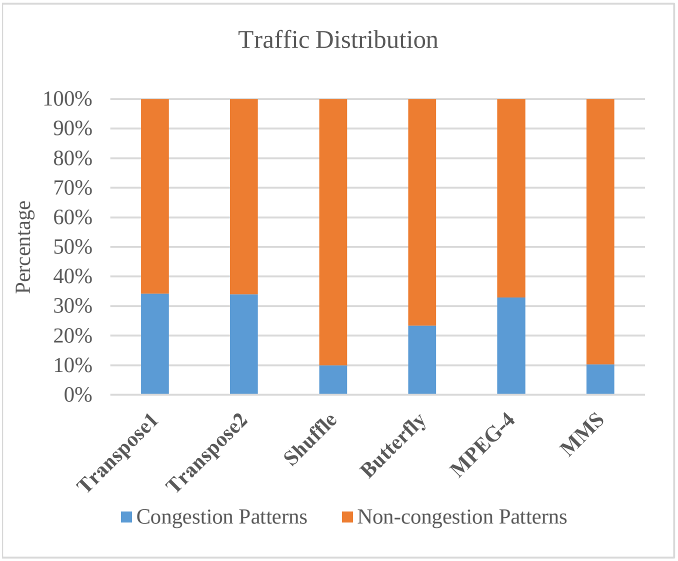


Fig. 7. Congestion distribution across different traffic scenarios

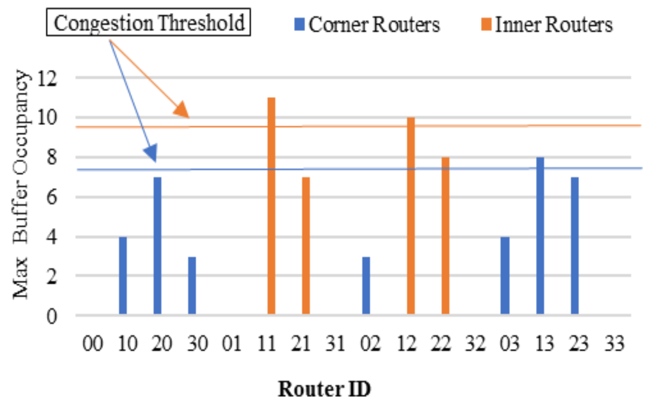


Fig. 8. Maximum buffer utilization on each router for MMS

media applications. The figure shows that a high PIR leads the network into congestion. After a certain increment level in PIR, the network reached saturation. With further increases in PIR the network throughput plateaus out as the routers become congested. To generate congestion conditions in the NoC architecture, PIR must be high enough to maintain the network in saturation condition. Depending on the routing algorithm used, saturation points fall at different PIR values for distinct traffic scenario. As illustrated from Fig. 6, every traffic pattern has a different saturation point, but all traffic patterns enter into congestion after PIR=~0.5. In this work, PIR is increased up to 0.5 to force routers to enter into congestion. This enables the SNN predictor to be evaluated. Fig.7 illustrates the overall traffic distribution to determine the percentage of traffic labelled as congested and non-congested. It is depicted that Tranpose-1 and transpose-2 traffic scenarios have generated the highest density (>30%) of congestion patterns whereas the shuffle traffic scenario generated (~11%) the least congestion causing patterns.

Fig. 8 shows the number of flits routed by each router during 1,000 clocks cycles: routers 0,0 through to 3,3 as per Fig.2. For the MMS application, it is depicted from generated patterns that inner routers have more traffic loads compared to corner routers. Because of application mapping and routing algorithm, these routers have to process more data packets and are more prone to congestion. It is also clear from utilization datasets that some router doesn't even enter into congestion because of traffic distribution. As the MMS application has a static traffic distribution, patterns are likely to repeat randomly. According to the proposed congestion criteria in Section III-B, it is evident from figure 8, that only three routers ID#[11, 12 and 13] entered into congestion state. If congestion for these routers is predicted in advance, then the network can prevent the impact from back pressure via re-routing packets on alternate paths. The same selection criteria will be used for MPEG-4, transpose1, transpose2, Butterfly and Shuffle traffic.

iii. SNN Architecture

The SNN predictor is modelled and simulated using MATLAB and uses a fully connected 3-layer model (shown in Fig. 4). In the proposed prediction model, the number of connected input channels defines the number of input layer neurons, and the level of connected buffers reflects the time of first spike to input neuron. The number of input channels depends on the location of the node in the NoC. For the 4x4 NoC, inner routers have more input channels as compared to corner router (as shown in Fig.2). The output layer of the SNN has one neuron and its spike time will determine the predicted router status (i.e., congested, non-congested) after 30 clocks. The number of hidden layers and number of neurons in each hidden layer depends on the complexity of the dataset (utilization patterns). In order to minimize network size without compromising network performance, we analysed different numbers of hidden layers with different number of neurons in each layer, i.e., to find the optimal hidden layer size. The SNN is optimized to a size where further reduction in hidden layer neurons will start reducing the prediction performance. During the SNN training, the neuron membrane threshold 'θ' and learning rate 'τ' help to regulate synaptic weights to generate spikes within the predefined time (NTF). These parameters are tuned to improve the learning rate of synapse weights and firing times of the SNN with in one NTF. Simulations were run on training datasets to achieve less than 5% Mean Square Error (MSE). The SNN predictor was developed in MATLAB for training and validation of NoC hotspot data. The parameters used for the configuration of the proposed SNN model are defined in Table 2.

TABLE 2 SNN CONFIGURATION PARAMETERS

Parameter	Value
Neurons in input layer	3-4 for corner nodes and 5 for inner nodes
Neurons in hidden layer	15
Neurons in output layer	1 neuron (congested/non-congested)
Threshold (θ)	40
Learning rate (τ)	3
Mean Square Error (MSE)	≤ 5%

The proposed model is highly scalable and can be implemented with any NoC size, mapping information, selection strategy and architecture. For any NoC configuration, the SNN is required to train on initial traffic patterns generated using a defined configuration to predict local congestion. The experimental setup has considered synthetic and multimedia applications to analyse prediction coverage under varying traffic scenarios. Each application has different traces for mapping on 4x4 NoC (discussed in Section IV.A.i. Traffic Scenarios). As a change in the mapping information changes source/destination nodes thus altering traffic flow which ultimately changes location of congestion. Section-IV-A(ii) discussed experimentation conditions that leads to hotspot formation and in Fig.8 and example of hotspot formation in an 4x4 NoC using MMS application traces is depicted.

Neural Computation begins with the start of a neural time frame (Fig. 5) and buffer utilization data arrives at the input neurons of the connected SNN predictor. A control signal (NTF) is part of the SNN that will allow input neurons to read utilization values from the Noxim simulator. Neurons in the SNNs are trained to produce spikes at each layer within predefined time (T_i, T_p and T_o (section III.D)). After a predefined time, a control signal disables the SNN from reading inputs from the buffers. The SNN will take time to process the information and a compute result. Data is processed through the network layers in the form of spikes and the final result is the aggregated in the output network layer. The spike time of the output neuron determines the status of the router as congested or non-congested after 30 clock cycles. For a given input, if the SNN predictor spikes within a defined time 't_c' (i.e. time is used to classify patterns as congested) then the network is more prone to getting congested; if the SNN predictor spikes after that time 't_{nc}', then the NoC router will not get congested for the next 30 clock cycles. In MATLAB, SNNs are trained to generated output spikes within the output spike time T_o. Once an output neuron spikes and information is forwarded to the congestion handling mechanism, the control signal (NTF) resets and enables the input layer of neurons to read buffer utilization values from Noxim and to then propagate spikes through the hidden layer towards output layer to predict potential hotspots.

iv. Performance Analysis Parameters

The proposed prediction models are analyse using two confusion matrix-based performance parameters: prediction accuracy (P_a) and prediction precision (P_p).

$$P_a = \frac{(\sum TP + \sum TN}{\sum(P + N)} \qquad (5)$$

$$P_p = \frac{\Sigma\, TP}{\Sigma P} \qquad (6)$$

where positive (P) associates with congestion patterns and negative (N) is associated with non-congestion patterns. TP and TN define correct predictions of the congestion patterns (P) and non-congestion patterns (N), respectively. Simulation results include average prediction accuracy and prediction precision of the whole mesh network as well as inner and corner nodes to further analyse prediction performance at each node.

V. RESULTS AND DISCUSSION

This section provides simulation results and outlines hardware utilization of proposed SNN based NoC traffic predictor.

A. *Simulation Analysis:*

To evaluate prediction performance of the proposed congestion method, the dataset is divided into two parts for training and validation. Experimental datasets extracted from noxim simulator are divided 60% for training and 40% for validation. Neural algorithms (SNN models) are trained for each traffic scenario to compare prediction capabilities of SRM-based SNN predictor with SpikeProp used as the learning algorithm (SRM-Spikeprop), and the LIF neuron based SNN predictor with SpikeProp to update synaptic weights during training (LIF-Spikeprop). For simulation analysis, we use congestion prediction accuracy and prediction precision as parameter to evaluate performance. To evaluate performance of both neuron models, results are analysed as an average prediction accuracy of i). the whole NoC and ii). Inner and corner routers only.

i. Performance Evaluation Under Synthetic Traffic Application

Built- Noxim synthetic traces were used to generate router buffer utilization patterns in the NoC. To determine the prediction accuracy and prediction coverage of synthetic traffic, buffer utilization patterns from each router were fed to the SNN for the particular router to predict its congestion 30 clocks in advance.

The average prediction accuracy of the proposed SNN network models for the four synthetic traffic patterns are provided in Table 3. The LIF model shows an accuracy between 88.28% to 94.84% as compared to SRM model that predicts congestion between 85.28% to 92.91% accuracy.

TABLE 3 AVERAGE PREDICTION ACCURACY UNDER SYNTHETIC TRAFFIC TRACES

	LIF-Spikeprop (%)	SRM-Spikeprop (%)
Transpose1	88.28	85.28
Transpose2	90.63	92.72
Butterfly	90.23	88.28
Shuffle	94.84	92.91

TABLE 4 AVERAGE PREDICTION ACCURACY OF INNER AND CORNER NODES SNN UNDER SYNTHETIC TRAFFIC TRACES

	LIF-Spikeprop (%)		SRM-Spikeprop (%)	
	Corner	Inner	Corner	Inner
Transpose1	89.98	83.19	86.19	82.56
Transpose2	88.88	95.88	91.67	95.88
Butterfly	86.98	100.00	84.38	100.00
Shuffle	97.19	87.81	98.46	76.25

Table 4 provides the performance of LIF-SpikeProp and SRM-Spikeprop SNN models on synthetic application traces for Corner and Inner router. For inner routers, the SRM based model gives an overall prediction accuracy of between 76.25% and 100% as compared to LIF-based SNN predictor with between 83.19% and 100%. For the corner router, the LIF model predictor exhibits between 86.98% and 97.19%, with the SRM model between 86.19% and 98.46% accuracy.

TABLE 5 AVERAGE PREDICTION PRECISION UNDER SYNTHETIC TRAFFIC TRACES

	LIF-Spikeprop (%)	SRM-Spikeprop (%)
Transpose1	91.97	91.97
Transpose2	82.09	73.00
Butterfly	88.66	88.66
Shuffle	85.42	85.23

Table 5 shows prediction precision performance of the proposed LIF-SpikeProp and SRM-Spikeprop based SNN models on synthetic application traces. The LIF-based prediction model has shown 82.09%~91.79% prediction precision as compared to 73.00%-91.97% prediction precision for the SRM-based SNN model.

ii. Performance Evaluation Under Real-Time Multi-Media Application

To determine the prediction accuracy and prediction coverage on real-time applications, the MPEG-4 [44] and MMS [2] applications traces were mapped on the Noxim simulator to generate buffer utilization patterns.

TABLE 6 AVERAGE PREDICTION ACCURACY UNDER REAL-TIME MULTIMEDIA TRAFFIC TRACES

	LIF-Spikeprop (%)	SRM-Spikeprop (%)
MPEG-4	95.73	95.45
MMS	96.25	95.69

Table 6 depicts the average congestion prediction performance of LIF and SRM models on real-time multimedia applications. SRM model shows 95.45% to 95.69% accuracy as compared with LIF model 95.73% to 96.25% average prediction accuracy.

TABLE 7 AVERAGE PREDICTION ACCURACY OF INNER AND CORNER NODES SNN UNDER REAL-TIME MULTIMEDIA TRAFFIC TRACES

	LIF-Spikeprop (%)		SRM-Spikeprop (%)	
	corner	inner	corner	inner
MPEG-4	98.21	88.31	98.21	87.19
MMS	96.48	99.56	95.73	99.56

Table 7 shows accuracy results for LIF-Spikeprop and SRM-SpikeProp. For corner routers, the LIF-SpikeProp is able to predict congestion between 96.48% to 98.21% accurately compared with SRM-SpikeProp where 95.73% to 98.21% accuracy is achieved. However, in the inner routers the LIF model shows an accuracy of 88.31% to 99.56% as compared with SRM based model with accuracy of 87.19% to 99.56%.

TABLE 8 AVERAGE PREDICTION PRECISION UNDER MULTIMEDIA TRAFFIC TRACES

	LIF-Spikeprop (%)	SRM-Spikeprop (%)
MPEG-4	96.73	97.47
MMS	93.92	98.48

The prediction precision performance of proposed LIF-based and SRM-based SNN models on multi-media application traces are provided in table 8. SRM based prediction model predicted congestion with more precision (97.47%-98.48%) as compared to LIF based SNN model (93.92%-96.73%).

B. *Discussion:*

In the synthetic traffic application, patterns are more likely to repeat after certain time intervals. In a regular traffic flow, these patterns are easy to predict. As depicted in Table 3, Butterfly traffic shows 100% prediction accuracy in corner routers. That happens because of the nature of regular application, were an increase in PIR increases network delay but doesn't affect traffic patterns in the network. Problems arise when nodes start entering into the congestion state. Congestion in a node causes delays in transmission of the data flowing through the congested node. Congestion changes the behaviour of data flow patterns and hence makes difficulties for prediction algorithm. Neural networks are exceptionally good at learning under changing behaviour, specifically for temporal patterns in digital NoC, SNNs are an ideal candidate to predict congestion. Congestion prediction is difficult for those routers which show irregularities in utilization patterns under high PIR.

Simulation results shows that LIF-SpikeProp performed well in predicting congestion under different traffic conditions as compared to the SRM-SpikeProp model. In the synthetic traffic application, SRM-Spikeprop model predict congestion in with 92.91% maximum average accuracy as compared to LIF-SpikeProp model 94.84%. In the real-time multimedia application, LIF-SpikeProp model predicts congestion up to 95.69% accurately as compared to SRM based model that shows 96.25% peak average prediction accuracy. Moreover, LIF based prediction model gives an overall prediction precision between 82.09% and 96.73% as compared to SRM-based SNN predictor with between 73% and 98.48%.

As explained in Section II-A, Inner routers are more susceptible to congestion. Tables 4 and 7 show the prediction accuracy of both models in inner and corner nodes. Results for inner routers shows that both neural models gives an overall maximum prediction accuracy of 100% (butterfly traffic traces). Whereas in corner nodes, LIF-SpikeProp and SRM-SpikeProp are capable to predict congestion with 98-21% and 98.46% peak average accuracy respectively. Overall results shows that LIF model predict congestion with more accurately and precisely as compared to SRM mode.

The scope of this work is limited to check congestion prediction coverage of spiking neurons in NoC architecture. The overall efficiency of network is dependent on congestion prediction and congestion handling mechanism. In this work, the SNN congestion predictor showed good prediction accuracy and in term of prediction coverage, the LIF-Spikeprop predictor was more accurate as compared to SRM-Spikeprop. The output of the SNN predictor is ideally forwarded to a routing algorithm or congestion handling mechanism to enable a new routing decision to be made in advance of the congestion being created. This aspect of the work is not addressed in this paper and focuses only on the prediction capability.

C. *Hardware Area Overhead*:

The work from [50] provide area estimates in evaluating the compactness of the proposed SNN predictor. By using the same approximation for neural model (shown in Fig. 4) we can calculate hardware overhead with respect to EMBRACE static and adaptive router [51], [52]. In NoC architecture, one static router, one adaptive router, one neurons, and one synapse will utilize $201 \times 10^{-3}\, mm^2$, $56 \times 10^{-3}\, mm^2$, $9 \times 10^{-6}\, mm^2$ and $24 \times 10^{-8}\, mm^2$ area respectively using 90 nm CMOS technology. In this work we proposed a layer of SNN predictors over the NoC where each router is connected to its own SNN. Therefore, we will evaluate hardware overhead in term of SNN to router overhead percentage. Hardware cost for one fully connected 3-layer SNN model [5:10:1] is $288 \times 10^{-6}\, mm^2$. Therefore, hardware costs for SNN is ~0.14% of static router whereas in adaptive router SNN increases the hardware by ~0.51%. SRM neuron consumes 2.5 times more hardware area as compared to LIF model[53], [54]. Comparing with EMBRACE static and adaptive router, SRM costs ~0.35% and ~1.26% area overhead. Analysis shows that overall hardware cost for SNN predictors is very low which makes them suitable for implementation on NoC for congestion prediction.

D. *Comparison against Existing Neural Prediction Techniques:*

In the NoC, ANNs are used in congestion aware routing algorithms. The only work of congestion prediction in NoCs is presented in [10], where an ANN-based predictor forecasts congestion based on synthetic and real-time multimedia applications. It uses a multi-layer ANN as an independent processing element (PE) in the NoC architecture that is capable of reading utilization data of neighbouring routers and predicting congestion with an accuracy ranging from 65% to 92%.

Due to the temporal nature of buffer utilization patterns, the SNN based prediction model showed an improved accuracy of 88.28%-95.69% compared to the ANN model [10]. Our proposed SNN predictor is computationally fast and requires less clock cycles to process data for the prediction task. The ANN model used one neural network for a 4x4 cluster which increases workload in determining exact congestion location to routing algorithm. However, our work proposed a layer of SNNs where every router has its own SNN predictor. In term of hardware overhead, the proposed models utilized 0.14%-0.51% and 0.35%-1.26% additional hardware as compared to 5.6% for the ANN based predictor. Thus, the two proposed configuration costs only exhibit minimal hardware overhead and provide improved prediction accuracy.

VI. CONCLUSION

In this work we proposed a novel SNN based congestion predictor that uses buffer utilization to predict congestion 30 clocks in advance of it occurring. Every router is associated with one SNN to identify hotspot locations. The proposed SNN prediction strategy is based on the integration of the backpropagation SpikeProp training algorithm with two spiking LIF and SRM neurons to encode synaptic weights. The two LIF and SRM neuron models were explored to assess which is most effective for prediction. Both SNN Predictors were trained and validated on synthetic and multimedia application traffic traces to evaluate prediction performances. This work explored and analysed the prediction performance of proposed SNN predictors on i) overall NoC network array and ii) on Inner and corner routers only. Simulation results showed that LIF-SpikeProp provided better prediction performance with 88.28%-96.25% accuracy and 82.09%-96.73% precision as compared to SRM-SpikeProp with 85.25%-95.69% accuracy and 73%-98.48% prediction precision. Both models performed exceptionally well in predicting NoC congestion. Results suggests that the LIF-Spikeprop model has a slight performance advantage on SRM-Spikeprop model.

Future work includes exploration of other parameters i.e. channel latency, traveling time and link utilization to enhance prediction performance. In next step we will introduce and assess congestion handling/routing mechanisms that will receive the prediction outcome from the SNN and be able to avoid the creation of the congested path by re-routing the data in advance.

References

[1] M. Amin, M. Shakir, A. Javed, M. Hassan, and S. A. Raza, "Low-cost fault tolerant methodology for real time MPSoC based embedded system," *Int. J. Reconfigurable Comput.*, vol. 2014, 2014.

[2] J. Liu, J. Harkin, Y. Li, and L. Maguire, "Online traffic-aware fault detection for networks-on-chip," *J. Parallel Distrib. Comput.*, vol. 74, no. 1, pp. 1984–1993, 2014.

[3] M. Tang, "Analysis on Local Congestion of Network-on-Chip," in *Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013)*, 2013, no. Iccsee, pp. 2863–2866.

[4] A. Benmessaoud Gabis and M. Koudil, "NoC routing protocols – objective-based classification," *J. Syst. Archit.*, vol. 66–67, pp. 14–32, 2016.

[5] U. Y. Ogras and R. Marculescu, "Prediction-based flow control for network-on-chip traffic," *Proc. DAC-44*, pp. 839–844, 2006.

[6] J. Liu, S. Member, J. Harkin, Y. Li, S. Member, and L. P. Maguire, "Fault-Tolerant Networks-on-Chip Routing With Coarse and Fine-Grained Look-Ahead," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 35, no. 2, pp. 260–273, 2016.

[7] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436 – 444, 2015.

[8] J. Liu, J. Harkin, L. P. Maguire, L. J. McDaid, J. J. Wade, and G. Martin, "Scalable Networks-on-Chip Interconnected Architecture for Astrocyte-Neuron Networks," *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 63, no. 12, pp. 2290–2303, 2016.

[9] 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 Networks Learn. Syst.*, vol. 29, no. 4, pp. 1287–1300, 2018.

[10] E. Kakoulli, V. Soteriou, and T. Theocharides, "Intelligent hotspot prediction for network-on-chip-based multicore systems," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 31, no. 3, pp. 418–431, 2012.

[11] U. Y. Ogras and R. Marculescu, "Prediction-based Flow Control for Network-on-Chip Traffic," in *Proceedings - Design Automation Conference, July 24-28, 2006, San Francisco, California, USA*, 2006, pp. 839–844.

[12] E. J. Chang, H. K. Hsin, S. Y. Lin, and A. Y. Wu, "Path-congestion-aware adaptive routing with a contention prediction scheme for network-on-chip systems," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 33, no. 1, pp. 113–126, 2014.

[13] H. Cai, Y. Yang, F. Qu, J. Wu, and B. Wang, "Congestion Prediction Algorithm for Network on Chip," vol. 11, no. 12, pp. 7392–7398, 2013.

[14] G. Ascia, V. Catania, M. Palesi, I. C. Society, D. Patti, and I. C. Society, "Implementation and Analysis of a New Selection Strategy for Adaptive Routing in Networks-on-Chip," *IEEE Trans. Comput.*, vol. 57, no. 6, pp. 809–820, 2008.

[15] P. Huang and W. Hwang, "An Adaptive Congestion-Aware Routing Algorithm for Mesh Network- on-Chip Platform."

[16] J. Latif, S. Azam, H. N. Chaudhry, and T. Muhammad, "Performance Evaluation of Modern Network-on-Chip Router Architectures," vol. 2, no. 2, 2016.

[17] Ming Li, Qing-An Zeng, and Wen-Ben Jone, "DyXY - a proximity congestion-aware deadlock-free dynamic routing method for network on chip," *2006 43rd ACM/IEEE Des. Autom. Conf.*, pp. 849–852, 2006.

[18] S. K. Subramaniam and R. Nilavalan, "Network Performance Optimization Using Odd and Even Routing Algorithm for pipeline network," in *2016 8th Computer Science and Electronic Engineering (CEEC)*, 2016, pp. 118–123.

[19] S. T. Atik, M. M. Imran, J. N. Mahi, J. A. Jeba, Z. I. Chowdhury, and M. S. Kaiser, "An Adaptive Routing Algorithm for on-chip 2D Mesh Network with an Efficient Buffer Allocation Scheme," in *2018 International Conference on Computer, Communication, Chemical, Material and Electronic Engineering (IC4ME2)*, 2018, pp. 1–4.

[20] H. Tseng, R. Wu, W. Chang, Y. Lin, and D. Duh, "An Efficient Traffic-Based Routing Algorithm for 3D Networks-on-Chip," in *Int'l Conf. Embedded Systems, Cyber-physical Systems, & Applications (ESCS'16)*, 2016, pp. 73–79.

[21] Y. S. C. Huang, K. C. K. Chou, and C. T. King, "Application-driven end-to-end traffic predictions for low power NoC design," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 21, no. 2, pp. 229–238, 2013.

[22] E. Carvalho, N. Calazans, and F. Moraes, "Congestion-aware task mapping in NoC-based MPSoCs with dynamic workload," *Proc. - IEEE Comput. Soc. Annu. Symp. VLSI Emerg. VLSI Technol. Archit.*, no. April, pp. 459–460, 2007.

[23] T. Chen, W. Fu, B. Xie, and C. Wang, "Packet triggered prediction based task migration for network-on-chip," in *20th Euromicro International Conference on Parallel, Distributed and Network-based Processing Packet*, 2012, vol. 38, no. 4, pp. 316–324.

[24] L. Ali, A. Rahman, A. Khan, M. Zhou, A. Javeed, and J. A. L. I. Khan, "An Automated Diagnostic System for Heart Disease Prediction Based on χ 2 Statistical Model and Optimally Configured Deep Neural Network," vol. 7, 2019.

[25] I. Khan, H. Zhu, D. Khan, and M. K. Panjwani,

"Photovoltaic Power prediction by Cascade forward artificial neural network," pp. 145–149, 2017.

[26] S. H. Adil, M. Ebrahim, and K. Raza, "Prediction of Eye State Using KNN Algorithm," *2018 Int. Conf. Intell. Adv. Syst.*, pp. 1–5, 2018.

[27] M. Saghir, Z. Bibi, S. Bashir, and F. H. Khan, "Churn Prediction using Neural Network based Individual and Ensemble Models," *2019 16th Int. Bhurban Conf. Appl. Sci. Technol.*, pp. 634–639, 2019.

[28] J. Latif, C. Xiao, A. Imran, and S. Tu, "Medical Imaging using Machine Learning and Deep Learning Algorithms: A Review," *2019 2nd Int. Conf. Comput. Math. Eng. Technol.*, pp. 1–5, 2019.

[29] M. Jawad, A. Rafique, I. Khosa, I. Ghous, J. Akhtar, and S. M. Ali, "Improving Disturbance Storm Time Index Prediction Using Linear and Nonlinear Parametric Models : A Comprehensive Analysis," *IEEE Trans. Plasma Sci.*, vol. 47, no. 2, pp. 1429–1444, 2019.

[30] M. Rezaei-ravari, "Low Latency Path Prediction Mechanism in 2D - NoC," *Electr. Eng. (ICEE), Iran. Conf.*, pp. 1565–1570, 2018.

[31] H. Cai, Y. Yang, F. Qu, J. Wu, and B. Wang, "Congestion Prediction Algorithm for Network on Chip," *TELKOMNIKA Indones. J. Electr. Eng.*, vol. 11, no. 12, pp. 7392–7398, 2013.

[32] X. Wang, X. Lin, and X. Dang, "Supervised learning in spiking neural networks : A review of algorithms and evaluations," *Neural Networks*, vol. 125, no. 2020 Special Issue, pp. 258–280, 2020.

[33] B. Han, A. Sengupta, and K. Roy, "On the Energy Benefits of Spiking Deep Neural Networks : A Case Study," in *International Joint Conference on Neural Networks (IJCNN)*, 2016, no. 1, pp. 971–976.

[34] A. Taherkhani, A. Belatreche, Y. Li, and L. P. Maguire, "A Supervised Learning Algorithm for Learning Precise Timing of Multiple Spikes in Multilayer Spiking Neural Networks," *IEEE Trans. Neural Networks Learn. Syst.*, pp. 1–14, 2018.

[35] W. Maass, "On the relevance of time in neural computation and learning," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1997.

[36] A. Mohemmed, S. Schliebs, S. Matsuda, and N. Kasabov, "Method for training a spiking neuron to associate input-output spike trains," in *IFIP Advances in Information and Communication Technology*, 2011.

[37] Q. Yu, H. Tang, K. C. Tan, and H. Yu, "A brain-inspired spiking neural network model with temporal encoding and learning," *Neurocomputing*, 2014.

[38] W. Gerstner and M. W. Kistler, *Spiking Neuron Models*. Cambridge University Press, 2002, 2002.

[39] R. Jolivet, T. J. Lewis, W. Gerstner, R. Jolivet, T. J. Lewis, and W. Gerstner, "Generalized Integrate-and-Fire Models of Neuronal Activity Approximate Spike Trains of a Detailed Model to a High Degree of Accuracy," *J. Neurophysiol.*, no. 92, pp. 959–976, 2004.

[40] S. M. Bohte, J. N. Kok, and H. La Poutre, "SpikeProp : Backpropagation for Networks of Spiking Neurons Error-Backpropagation in a Network of Spik- ing Neurons," *Esann*, no. May, pp. 419–424, 2000.

[41] N. Mastronarde and C. Ababei, "Benefits and costs of prediction based DVFS for NoCs at router level," in *27th IEEE International System-on-Chip Conference (SOCC)*, 2014, pp. 255–260.

[42] N. Alfaraj, J. Zhang, Y. Xu, and H. J. Chao, "HOPE : Hotspot Congestion Control for Clos Network On Chip," in *Proceedings of the Fifth ACM/IEEE International Symposium, Pittsburgh, PA*, 2011, no. c, pp. 17–24.

[43] V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti, "Noxim : An Open , Extensible and Cycle-accurate Network on Chip Simulator," *2015 IEEE 26th Int. Conf. Appl. Syst. Archit. Process.*, pp. 162–163, 2015.

[44] V. D. B, M. Ho, V. Nguyen, and T. Ngo, "The Analyzes of Network-on-Chip Architectures Based on NOXIM Simulator," in *Advances in Information and Communication Technology. ICTA 2016. Advances in Intelligent Systems and Computing, vol 538. Springer, Cham*, 2017, vol. 1, pp. 603–611.

[45] Z. Fang, W. U. Ning, Z. Xiaoqiang, Z. Lei, and Y. E. Yunfei, "An Energy-Aware Voltage-Frequency Island Partition Method for NoC," *Chinese J. Electron.*, vol. 25, no. 3, 2016.

[46] J. Liu *et al.*, "Self-repairing learning rule for spiking astrocyte-neuron networks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2017.

[47] T. Bjerregaard and S. Mahadevan, "A Survey of Research and Practices of Network-on-Chip," *ACM Comput. Surv.*, vol. 38, no. 1, pp. 1–51, 2006.

[48] Z. Qian *et al.*, "A Support Vector Regression (SVR) - Based Latency Model for Network-on-Chip (NoC) Architectures," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 35, no. 3, pp. 471–484, 2016.

[49] N. Architectures, K. Srinivasan, and K. S. Chatha, "A Technique for Low Energy Mapping and Routing in Network-on-Chip Architectures," in *ISLPED '05. Proceedings of the 2005 International Symposium on Low Power Electronics and Design, August 8-10,2005, San Diego, California, USA*, 2005, pp. 387–392.

[50] J. Liu, J. Harkin, M. Mcelholm, and L. Mcdaid, "Case Study : Bio-inspired Self-adaptive Strategy for Spike-based PID Controller," *2015 IEEE Int. Symp. Circuits Syst.*, pp. 2700–2703, 2015.

[51] S. Carrillo *et al.*, "Advancing interconnect density for spiking neural network hardware implementations using traffic-aware adaptive network-on-chip routers," *Neural Networks*, vol. 33, pp. 42–57, 2012.

[52] J. Harkin, F. Morgan, L. Mcdaid, S. Hall, B. Mcginley,

and S. Cawley, “A Reconfigurable and Biologically Inspired Paradigm for Computation Using Network-On-Chip and Spiking Neural Networks,” *Int. J. Reconfigurable Comput.*, vol. 2009, 2009.

[53] A. Rosado, M. Bataller, and J. Guerrero, “FPGA implementation of Spiking Neural Networks,” in *Proceedings of the 1st IFAC Conference on Embedded Systems, Computational Intelligence and Telematics in Control - CESCIT , Germany*, 2012, vol. 45, no. 4, pp. 139–144.

[54] A. Pulikkakudi *et al.*, “Fault-tolerant Learning in Spiking Astrocyte-Neural Networks on FPGAs,” in *31st International Conference on VLSI Design (VLSID 2018) & 17th International Conference on Embedded Systems (ES 2018)*, 2018.