Noname manuscript No.
(will be inserted by the editor)

An efficient, low-cost routing architecture for spiking neural network hardware implementations

Yuling Luo · Lei Wan · Junxiu Liu* · Jim Harkin · Yi Cao

Received: date / Accepted: date

Abstract The basic processing units in brain are neurons and synapses that are interconnected in a complex pattern and show many surprised information processing capabilities. The researchers attempt to mimic this efficiency and build artificial neural systems in hardware device to emulate the key information processing principles of the brain. However, the neural network hardware system has a challenge of interconnecting neurons and synapses efficiently. An efficient, low-cost routing architecture (ELRA) is proposed in this paper to provide a communication infrastructure for the hardware spiking neuron networks (SNN). A dynamic traffic arbitration strategy is employed in ELRA, where the traffic status weights of input ports are calculated in real-time according to the channel traffic statuses and the port with the largest traffic status weight is given a high priority to forward packets. This strategy enables the router to serve congested ports preferentially, which can balance the overall network traffic loads. Experimental results show the feasibility of ELRA under various traffic scenarios, and the hardware synthesis result using SAED 90nm technology demonstrates it has a low hardware area overhead which maintains scalability for large-scale SNN hardware implementations.

Keywords Spiking neural networks · Networks-on-Chip · Routing arbitration

Yuling Luo, Lei Wan, Junxiu Liu
Guangxi Key Lab of Multi-Source Information Mining & Security,
Faculty of Electronic Engineering, Guangxi Normal University,
Guilin, China, 541004
E-mail: *liujunxiu@mailbox.gxnu.edu.cn of Junxiu Liu

Jim Harkin
School of Computing and Intelligent Systems,
University of Ulster, Londonderry, UK, BT48 7JL

Yi Cao
Department of Business Transformation and Sustainable Enterprise,
Surrey Business School,
University of Surrey, Surrey, UK, GU2 7XH

1 Introduction

The mammalian brain functionality is based on the specialized signal processing capabilities of massive neurons [1]. One key outcome from neuroscience research is a computing neural model of spiking neural networks (SNN) [1–3], which has the capability to emulate information processing of massive neurons in the brain. The neurons in the SNN exchange information via transmitting the spikes through the synapses [4]. The brain contains approximate 10^{10} neuron cells and 10^{15} synapses in a parallel manner [1]. The SNN has the potential to emulate the information processing capability of the brain. A large-scale SNN normally includes a significant number of neurons and synapses. In order to implement and simulate the large-scale SNNs, current software simulations are too slow and consume lots of power, thus cannot maintain the system scalability [4]. Therefore many researchers attempted to use custom hardware devices for the SNN implementations in order to address the challenge of scalability. However, the complexity of inter-neuron connectivity in SNN limits the network implementation in hardware, as the number of connections increases exponentially with the number of neurons and synapses. The traditional hardware interconnection strategy, e.g. bus, point-to-point, cannot overcome the SNN connection problems and is lack of scalability [5]. It is necessary to develop an efficient interconnect architecture for the connections of neurons and synapses in the SNN.

Recently, researchers proposed the networks-on-chip (NoC) interconnect paradigm [1, 4–6] as a promising solution to solve the inter-neuron connectivity problems and achieved a satisfactory performance [1, 4]. The NoC is similar to the computer network where the processing elements (e.g. neurons in the SNN) are connected by the routers and channels [6]. For the SNN, the spikes are packetized and can be forwarded from any source node to any destination node; thus the information exchange between the neurons is established. In general, a group of neurons (e.g.∼ten in the approach of Carrillo *et al.* [4]) are connected to one router, but the required routers and channels increase if the number of neurons increases, which leads to the hardware area and power dissipation consumption. Thus the NoC architecture (i.e. routers and channels) constraints the system scalability [1], and it should achieve a trade-off between performance (e.g. spike throughput and communication delay etc.) and resource consumption (e.g. hardware area and power consumption). *Performance:* the NoC routers are responsible for transmitting highly irregular spike events. An effective router should forward as many spike events as possible in a short time period for various traffic scenarios, i.e. to provide high throughput for the communications. In addition, the irregular spike patterns occasionally introduce traffic congestions [4], which require the router have the ability to monitor the different traffic status (e.g. busy or congested etc.) and deal with various traffic patterns [5]. *Resource consumption:* the number of required routers increases proportionally with the number of neurons and synapses [1]. A low-cost routing architecture is very crucial for large-scale SNN hardware systems. Therefore, for the SNN hardware interconnection, the NoC router architecture should provide an efficient communication mechanism for the neurons, and only need a low hardware cost for implementation which maintains the SNN system scalability.

In this paper, a compact NoC router for the SNNs is proposed which can provide an efficient routing mechanism for the neuron communications with a low hardware cost. The rest of this paper is organized as follows: Section II describes the background and summarizes related works. Section III discusses the proposed router architecture. Section IV provides performance analysis and hardware evaluation results, and Section V concludes the paper.

2 Background and related work

Various approaches have been explored for SNN implementation, including software, application-specific integrated circuit (ASIC) and field programmable gate array (FPGA) [7]. The software approaches are too slow for the large-scale SNNs simulation and have limited scalability due to the huge power consumption [8]. For example, a computer reconstruction of a piece of the neocortex (∼31K neurons and ∼37K million synapses) is completed by using the high-performance computer (HPC) in the Blue Brain project [9], where the electrical behavior of the virtual brain tissue was simulated on the IBM supercomputers. The main constraint of using the HPC is a low level of parallelism and the extremely high power consumption (in the order of hundreds of kilowatts generally). Thus researchers have looked into using the hardware devices (e.g. ASICs, FPGAs) to accelerate the computing speed and reduce power consumption and make the large-scale SNN implementation possible. The main drawback of using ASIC devices is the high cost for the development and chip manufacturing. In addition, it only has a limited level of flexibility as a tiny change would lead to a new development cycle. For large-scale SNNs, FPGA device offers a high speed of execution and good scalability, which also achieves a lower cost than ASICs.

In the hardware SNNs, a point-to-point connectivity method between neuromorphic chips using address events was proposed in the approach of Boahen [10]. Another FPGA design framework for large-scale SNN using all-to-all connection strategy was implemented in the approached of Wang *et al.* [11]. Recently, the interconnection strategy of NoC as an efficient SNN interconnect strategy has been mostly used [12–14], where the topology (i.e. the structure regarding the neuron connections) is an important factor for the system performance. It has been demonstrated that the topology of bus is not scalable for the SNN as the number of required buses is proportional to the number of neurons [4]. The mesh and torus are common used topologies, e.g. a NoC-based routing architecture based on two-dimensional mesh was proposed in the approach of Carrillo *et al.* [1], the FACETS in the approach of Schemmel *et al.* [15] was based on a 2D torus which provided the connections of several FACETS wafers, and the SpiNNaker [16] used a triangular torus topology to connect the neuron computing cores. Additionally, a hierarchical NoC architecture for SNN was proposed in the approach of Carrillo *et al.* [4], which combined the mesh and star topologies for different layers of the SNNs. Similar to the aforementioned approaches, most of current SNN hardware systems use the topologies with limited degree (e.g. mesh, torus etc.) to connect the neurons together [4, 17], however these topologies do not meet the fan-in/out requirement of layer-based SNN structures [18]. Therefore, it is necessary to look to a new hardware interconnection architecture to address this challenge [18].

This paper presents a compact and efficient NoC router architecture for the SNN implementations. It employs a traffic status weight-based arbitration and a traffic congestion-avoidance mechanism to provide traffic balance for the information transmission of multiple neurons. The main contributions of this work include:

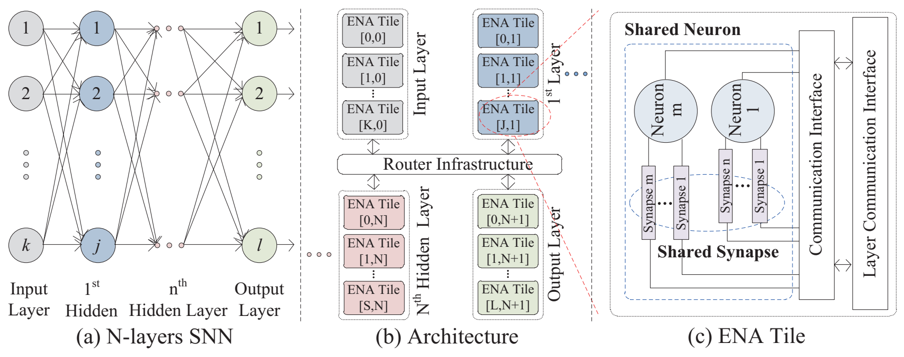


Fig. 1 The ENA architecture overview [19].

(a) a low-cost router architecture is proposed for the SNN hardware implementations; (b) a dynamic traffic status weight strategy is employed in the proposed router to sustain the quality of service under different spike traffic loads; (c) the proposed router architectures also combines with a traffic congestion mechanism to maintain the traffic load for the neurons of one input layer.

3 The ELRA router

This section presents the proposed ELRA and its application in the SNN hardware applications. In general, the SNN is a layer-based network which includes an input/output layer, and one or several hidden layers. Fig. 1(a) illustrates a typical SNN with N-layers. Each neuron in one layer communicates to all the neurons in next layer via the synapses. The interconnected architecture provides the communication mechanism for all neurons. The proposed ELRA is such a communication mechanism for the layer-based hardware SNNs. In our previous work [19], a hardware architecture, namely Efficient Neuron Architecture (ENA) is proposed, which implemented a group of neurons in one layer in hardware. Fig. 1(b) is an example to use ENAs to implement a typical SNN structure (i.e. Fig. 1(a)), where one ENA corresponds to the neurons of a layer in the SNN. A single ENA has the capability to implement up to ∼18,181 neurons and synapses. Each ENA utilizes a computing resource sharing mechanism at two levels (i.e. synapse and neuron) to reduce the occupied resources, see Fig. 1(c). If the number of neurons in one layer is more than that, multiple ENAs can be employed for accommodation. Fig. 1(b) shows that the ENAs are connected by a global communication infrastructure in order to realize a larger-scale SNN system. The aim of this paper is to propose the interconnected architecture for the communication of the layer-based SNNs and the ENAs [19] are used as the neuron node example for the hardware SNNs. Note that the ELRA is not constrained to the ENAs, however it can be applied to any other layer-based SNN hardware systems.

In this work, an efficient low-cost router architecture (ELRA) is proposed, which can forward the spike events efficiently for the communications of ENAs. The interconnections between the ELRAs and ENA tiles are illustrated in Fig. 2(a) where each ENA connects to the local port of an ELRA, and the ELRA has a

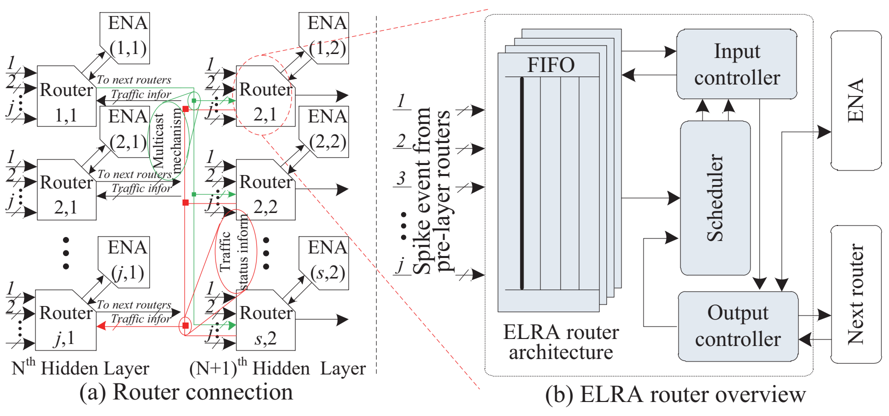
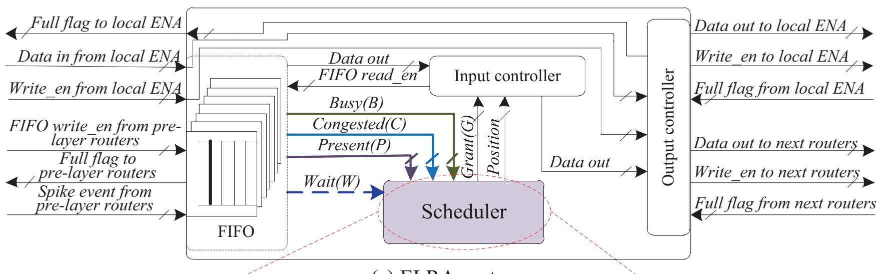


Fig. 2 Spiking neural network interconnection and ELRA structure overview.

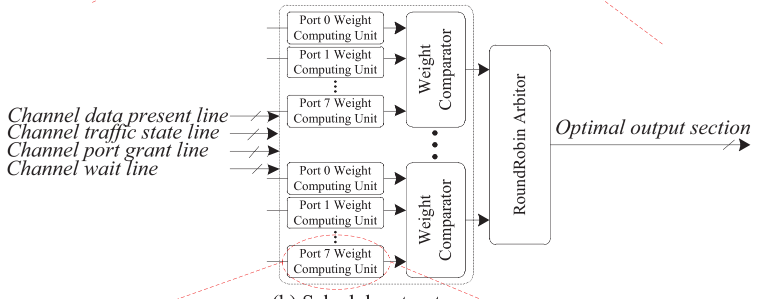
one-to-all connection to the ones in the next layer. When a spike event arrives at the ELRA, it forwards the event to the local ENA tile. After the ENA generates the spike events, the ELRA forwards these events to the nodes in next layer. The proposed ELRA can also receive the spike events from multiple sources (e.g. all the nodes in previous layer), and this requires the ELRA have the ability to arbitrate the various traffic loads. Fig. 2(b) is the overview of the proposed ELRA. It comprises of four main modules, i.e. the input buffers, the input controller, the scheduler module, and the output controller. The structure and functionality of the ELRA are presented in detail in the following sub-sections.

3.1 The ELRA structure

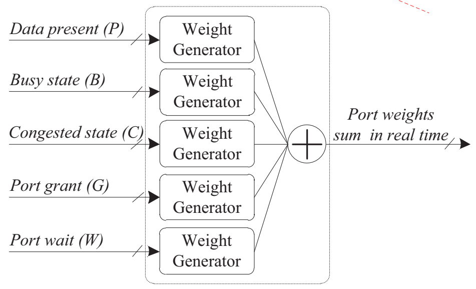
The ELRA router diagram is shown in Fig. 3(a). It includes four modules - FIFO, scheduler, input and output controller modules. When multiple spikes arrive at input ports, the FIFO that corresponds to the individual input port firstly stores the spike packets; then the scheduler provides the arbitration for the input ports where the input port with the highest traffic status weight can achieve service promptly; meanwhile, the input and output controllers control the packet forwarding processes. The hardware structure of scheduler is depicted in Fig. 3(b). It consists of three components, including traffic status weight computing unit, traffic status weight comparator and round-robin arbiter. When multiple input ports receive the packets, a traffic status weight comparison mechanism is employed to arbitrate the access of the output channel. The traffic status weights of input ports are calculated in real-time according to the input channel status, including channel data present (labeled by 'Present' in Fig. 3(a)), traffic status (Busy/Congested) and the channel wait (Wait) and grant (Grant) information. The arbitration of input ports are based on three steps: (1) to separate all the input ports into N groups and each group contains M ports. The values of N and M depends on the applications. They are generally set to the power of 2 as the number of comparator input ports is even; (2) to calculate the traffic status weights for every input port in each group according to the traffic status; then to select the input port with



(a) ELRA router


Channel data present line
Channel traffic state line
Channel port grant line
Channel wait line
Port 0 Weight Computing Unit
Port 1 Weight Computing Unit
Port 7 Weight Computing Unit
Weight Comparator
RoundRobin Arbitor
Optimal output section

(b) Scheduler structure



(c) Weight computing unit

Fig. 3 The ELRA structure: (a) an overview of the ELRA router; (b) the scheduler module; (c) the traffic status weight computing unit.

largest traffic status weight via the traffic status weight comparator; and (3) to grant the input port with largest traffic status weight in each group to access the output channel using a round-robin arbiter. Fig. 3(c) illustrates the structure of one traffic status weight computing unit. Five traffic status generators calculate the corresponding traffic status weights in real-time according to the input channel traffic status, which can be achieved by reading the statuses of corresponding FIFOs and grant information. For example, if the ‘present’ signal of FIFO is high, the FIFO has received valid packet(s). If the number of free slots (F_s) is equal to zero, the buffer is full and the channel is said to be ‘Congested’. If $F_s \leq Threshold\_v$, the channel is said to be ‘Busy’, where $Threshold\_v$ denotes a threshold value (normally half the size of the buffer). In addition, if a port has achieved a service grant, the port status is ‘Grant’ and for other ports which are not granted in this process, they are in the ‘Wait’ status if they have packet waiting for transmission (i.e. the 'Present' signal is high). The five traffic status weight factors are defined as Present/Busy/Congested/Grant/Wait. The five status signals are connected to the scheduler to aid making effective arbitration decisions. The traffic status weight computing unit is explained in detail in the following sub-sections.

3.2 Dynamic traffic status weight strategy

In order to overcome the traffic imbalance caused by the irregular spike patterns [1], a dynamic traffic status weight strategy is employed in ELRA to maintain the performance and communication service quality under various traffic load scenarios.

The traffic status weight values to be calculated include (1) the data *Present* traffic status weight w_p, (2) the channel *Busy* status traffic status weight w_b, (3) the channel *Congested status* traffic status weight w_c, (4) the port *Grant* traffic status weight w_g, (5) the port *Wait* status traffic status weight w_w. They are determined by the corresponding signals in Fig. 3(a) and are calculated based on the following dynamic traffic status weight mechanism: Firstly if the status, s, of a channel has a data present, then $s_p = 1$; if the channel is busy, status $s_b = 1$; if the channel is congested, status $s_c = 1$; if the channel is granted, status $s_g = 1$ and if the channel is waiting the grant, status $s_w = 1$. Using this status information, the values of $w_p/w_b/w_c/w_g/w_w$ are calculated using Equation (1). The data present traffic status weight w_p has the most significant impact on the channel to distinguish the state that the channel has data or not, therefore w_p is 3 if there is a data present (i.e. $s_p = 1$). Similarly, Congested has more impact than Busy, so w_c is 2, w_b is 1 if the corresponding statuses are satisfied. If the channel is granted ($s_g = 1$), w_g is -1. This is because this channel has been granted in this round, it should decrease the probability to get grant again in next round, therefore the traffic status weight is decreased by 1. However, if the channel does not get the grant and is waiting ($s_w = 1$), the probability to get the grant should be increased, i.e. $w_w = 1$. After all the traffic status weight values are generated, the total traffic status weight, w_{sum}, for each port i in each group, is calculated by Equation (2) in real-time. The port with the largest traffic status weight in each group is selected to be granted by the round-robin arbiter in the next step.

$$\begin{cases} w_p = \begin{cases} 0, & s_p = 0 \\ 3, & s_p = 1 \end{cases} \\ w_b = \begin{cases} 0, & s_b = 0 \\ 1, & s_b = 1 \end{cases} \\ w_c = \begin{cases} 0, & s_c = 0 \\ 2, & s_c = 1 \end{cases} \\ w_g = \begin{cases} 0, & s_g = 0 \\ -1, & s_g = 1 \end{cases} \\ w_w = \begin{cases} 0, & s_w = 0 \\ 1, & s_w = 1 \end{cases} \end{cases} \tag{1}$$

$$w_{sum}[i] = w_p[i] + w_b[i] + w_c[i] + w_g[i] + w_w[i] \tag{2}$$

Secondly, a round-robin policy is used to arbiter the selected ports in each group, which allows the fairness of all the groups to access to the output channel.

Based on this dynamic traffic status weight mechanism, it can be noticed that the traffic status weight of each port is calculated in real-time. When the traffic status weight sum $w_{sum}[i]$ is big enough, the port has a high probability to obtain the grant, and vice versa. This allows the priorities of all ports are updated in real-time to access to the output channel, and avoids the problems of the fixed priority scheduler (e.g. one port is occupied for a long time) and round-robin arbitrator (e.g. no priority for the busy traffic path) [1].

4 Performance analysis

This section presents performance results on the packet delay and throughput for various SNN traffic scenarios. The hardware area overhead and power consumption are also highlighted by comparing with the existing approaches. The hardware implementation is based on a Xilinx XC7Z020-CLG484 device. The system frequency is 100 MHz and the packet data width is 62-bit. The area overhead and power consumption have been evaluated based on a Synopsys Armenia Educational Department (SAED) 90 nm CMOS technology.

4.1 Packet throughput and delay

The neuron outputs spikes under various patterns [1] e.g. regular, fast, bursting and rebound as shown in Fig. 4. The rebound and bursting spiking events are irregular, which have a major impact on the packet delivery latency and may lead to traffic congestion. This requires the routers have the ability to balance the traffic load across the neural network.

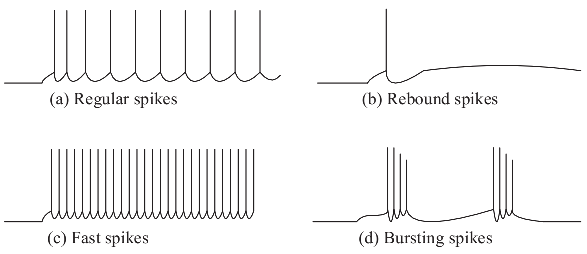


Fig. 4 The typical spike patterns.

A key aspect for the performance verification of the proposed ELRA is to analyse how the ELRA guarantees efficient routing ability (e.g. throughput) under various spiking patterns. The neuron node ENA [19] connects to the ELRA. All the ENAs use the packets to transmit the information [19]. The packet layout is illustrated by Fig. 5 which includes five parts: ENA address, neuron address, synapse address, packet type and maximum 32-bit width payload. The rate encoding scheme is used in this paper, and the SNN can be trained using the learning algorithm in our previous works [20, 21]. In addition, the proposed routing architecture can be used for other applications, e.g. traditional neural networks. The packet payload can accommodate the data that is required for transmission.

Header				
ENA address (8 bits)	Neuron address (8 bits)	Synapse address (10 bits)	Packet type* (4 bits)	Payload (32 bits)

Packet type	Value
Spike Packet	0000
Configuration Packet	1***

Fig. 5 The packet layout for the ENA [19].

Fig. 6 illustrates the average throughput under the different number of enabled ENAs with various SIRs. In this experiment the total number of ENAs is 16. The router with round-robin scheme is used as a bench mark. In order to simulate various spike traffic patterns, e.g. regular, fast, burst or rebound patterns, the experiments are carried out with different SIRs and the number of enabled ENAs. For example, in the regular or rebound spike patterns the SIR is relatively low; thus it is set to be 0.03125 (i.e. one spike packet is generated every 32 clock cycles). Fig. 6 shows that for this scenario, the ELRA achieves a slightly higher throughput than the round-robin scheme. However, if the spike traffic pattern is bursting or fast, the ELRA achieves much higher throughput compared to round-robin scheme. For these patterns, not all the neurons generate spike packets, i.e. the number of enabled ENAs only occupies a small percentage of total ENAs, but SIR is higher than regular traffic pattern (e.g. SIR is 0.5 which causes bursting activity). Fig.6 shows that when the number of enabled ENAs decreases, the throughput difference between ELRA and round-robin scheme becomes larger and ELRA has a much higher throughput. For instance, if only 2 ENAs are enabled to generate the spike packets which are a typical bursting traffic pattern, the ELRA has a 140% throughput improvement than round-robin scheme. Therefore, the ELRA has the capability to achieve a good throughput performance under various traffic patterns. Especially for the irregular traffic pattern of bursting and fast etc., it has a much higher throughput. This is because the ELRA can efficiently arbitrate the data request without wasting time on the channels with no data present. Thus the experimental result in Fig.6 demonstrates that ELRA has the capability to balance the traffic loads across the hardware interconnected SNN.

In order to illustrate the packet delay of the proposed ELRA, hardware simulation result of a single router is shown in Fig. 7. The signal of *Data_in_fifo* is the input data channel for the FIFOs. It is used to temporarily store the generated spike events from the ENAs. In this experiment the number of input groups is 3 (high dimension is [2:0], i.e. $N = 3$), and the number of data channels in each group is 8 ([7:0], i.e. $M = 8$), i.e. a total of 24 data input channels are used to receive spike events. However N and M are not limited to these values which are only used as an example for this experiment. It can be seen that for the first group only one channel has packet for transmission (i.e. X“131311C44443”), see in

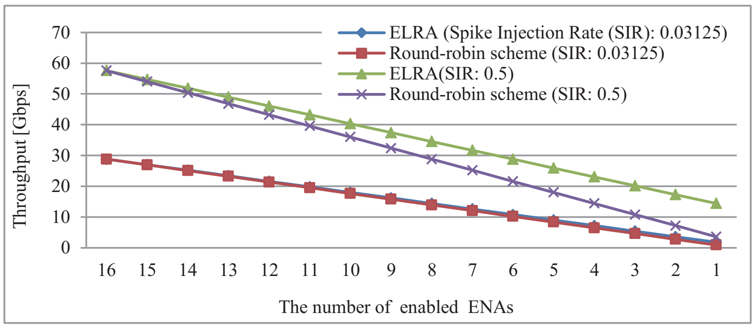


Fig. 6 Relationship between the different number of enabled ENAs and throughput per router under different SIRs.

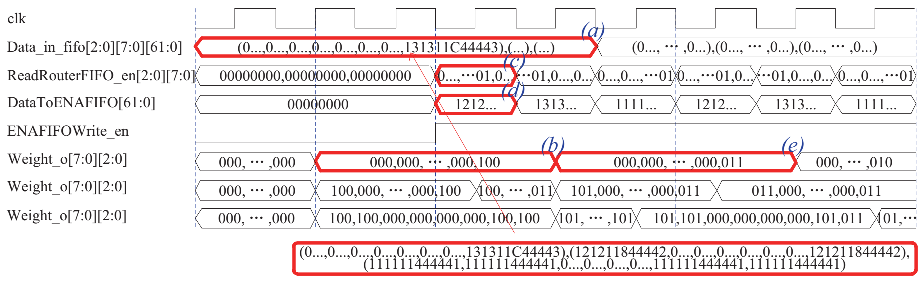


Fig. 7 Simulation result of ELRA router.

Fig. 7(a). The second and third groups have 2 and 4 channels receiving packets and they are X“121211844442” and X“111111444441”, respectively. This is an irregular spike patterns as the third group has more requests than others. Fig. 7(a) shows that the packet X“131311C44443” is injected into the first group of *Data_in_fifo* for transmission. Then the corresponding traffic status weight increases from “000” to “100”, see Fig .7(b). The grant decision is made by the signal of *ReadRouterFIFO_en* from the input controller module according to the maximum traffic status weight, see Fig. 7(c). The granted packet is transmitted through the signal of *DataToENAFIFO* in the output controller module, as shown by Fig. 7(d). Next Fig. 7(e) shows that the corresponding traffic status weight decreases from “100” to “011” via the dynamic traffic status weight mechanism. It should be noted that only 3 clock cycles are needed from spike event arriving to grant decision being made in the output controller module. The other two groups adopt a similar arbitration process and a round-robin policy is employed for selecting the granted spike packet between the different groups. Thus the proposed ELRA senses transmission requests of these packets, provides arbitration for all the input channels via the dynamic traffic status weight mechanism, and forwards the packet promptly to the output ports.

4.2 Analysis and comparison of area utilization, power and throughput

Fig. 8 summarizes the area utilization in hardware for a single ELRA router. Results show that the ELRA router is 61,472 μm^2 in total, where the FIFOs for the input ports occupy the largest area (87.7%), scheduler, input and output controllers occupy 4.99%, 6.78% and 0.53% of the entire router area. Table 1 compares the power, throughput and area utilization of the proposed ELRA router with other state of the art approaches [4–6, 22].

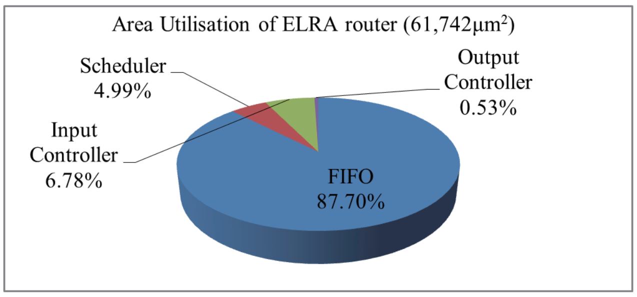


Fig. 8 Synthesis summary regarding the area utilization distributions of the proposed ELRA router.

The approaches of Liu and Harkin *et al.* [5, 6, 22]are based on 2D mesh topology. Each router contains five input FIFOs for the North/E/S/W and local ports. For the fairness of comparison, five input ports are set in proposed ELRA. The approach of Harkin *et al.* [22] provides an architecture for the hardware SNNs, incorporating analogue spiking neurons, and interconnected architecture using a 2D mesh-based NoC. However it does not have the traffic balance capability, therefore the area overhead and power consumption are relative low (i.e. 0.201^2 and 1.72 mW). Other NoC routers [4–6] and the proposed ELRA are all equipped with a traffic congestion avoidance mechanism. A hierarchical NoC architecture for hardware SNN was proposed in the approach of Carrillo *et al.* [4], which combined the mesh and star topologies for different layers of the SNNs. It can be seen that the routers have a higher power and area overhead due to the complex routing algorithms. Similarly, the approach of Liu *et al.* [6] provides adaptive fault-tolerant routing algorithm for the 2D mesh/torus-based NoCs. It can be used for the interconnection of SNN. But it also suffers from the high power and area consumption. Compared with the approach of Liu *et al.* [6], the EDAR [5] is based on a simpler routing algorithm and has a slightly lower power consumption. The proposed ELRA provides the interconnected architecture for the layer-based SNNs. The area overhead and power consumption of the ELRA are 0.062 mm^2 and 3.161 mW. Compared with the EDAR [5], the ELRA has a slightly higher power consumption, but the hardware area utilization is much less than it.

For the large-scale SNNs, the required router increase with the number of neurons in each layer. Fig. 9 shows the required router areas of two approaches, ELRA in this work and the approach of Carrillo *et al.* [1]. In the approach of Carrillo *et al.* [1], one neuron connects to a router, therefore the hardware area of

Table 1 Performance comparison with other approaches

Project Reference	Congestion Mechanism	Area Utilization	Throughput (Gbps)	Power(mW)
EMBRACE [22]	No	0.201 mm^2 (90nm CMOS)	16.0	1.72
H-NoC [4]	Yes	0.587 mm^2 (TSMC 65nm)	3.33	13.16
EDAR [5]	Yes	0.241 mm^2 (SAED 90nm)	18.0	2.291
FG [6]	Yes	0.268 mm^2 (SAED 90nm)	NA	72.31
This work	Yes	0.062 mm^2 (SAED 90nm)	18.0	3.161

routers increase linearly with the number of neurons. In this work, the hardware area of a single router is less than the router in the approach of Carrillo *et al.* [1]. In addition, the ELRA is for the layer-based SNNs. It achieves much less area overhead for a large network, e.g. for one layer with 1,440 neurons only 8 ELRAs are required; the approach of Carrillo *et al.* [1] needs one router per neuron which requires more area than the ELRA.

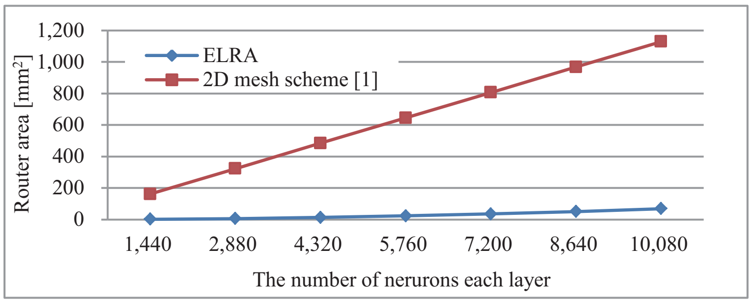


Fig. 9 The comparison of router area overhead.

5 Conclusion

This paper proposed an efficient, low-cost routing architecture (ELRA) as a fabric for interconnections of neurons and synapses in the SNN hardware implementations. A novel dynamic traffic status weight calculation and selection strategy was employed in ELRA to provide an efficient routing policy under various spike traffic patterns of SNN. The hardware area of ELRA is only 0.062 mm^2 based on a 90nm CMOS technology which demonstrated the scalability for the large SNN system.

Acknowledgments

This research was supported by the National Natural Science Foundation of China under grants 61603104 and 61661008, the Guangxi Natural Science Foundation under grants 2015GXNSFBA139256 and 2016GXNSFCA380017, the funding of Overseas 100 Talents Programme of Guangxi Higher Education, the Research Project of Guangxi University of China under grant KY2016YB059, Guangxi Key Lab of Multi-source Information Mining & Security under grant MIMS15-07, the Doctoral Research Foundation of Guangxi Normal University, the Research Project of Guangxi Centre of Humanities & Social Sciences - Ecological Environment Forecast and Harnessing in Ecologically Vulnerable Region of Pearl River and Xijiang Economic Zone (ZX2016030), and the Innovation Project of Guangxi Graduate Education (YCSZ2016034).

References

1. S. Carrillo, J. Harkin, L. Mcdaid, S. Pande, S. Cawley, B. Mcginley, and F. Morgan (2012) Advancing interconnect density for spiking neural network hardware implementations using traffic-aware adaptive Network-on-Chip routers. Neural Networks 33:42-57
2. W. Yang, J. Yang, and W. Wu (2012) A modified spiking neuron that involves derivative of the state function at firing time. Neural Process. Lett. 36:135-144
3. B. Meftah, O. Lezoray, and A. Benyettou (2010) Segmentation and edge detection based on spiking neural network model. Neural Process. Lett. 31:131-146
4. S. Carrillo, J. Harkin, L. J. McDaid, F. Morgan, S. Pande, S. Cawley, and B. McGinley (2013) Scalable hierarchical Network-on-Chip architecture for spiking neural network hardware implementations. IEEE Trans. Parallel Distrib. Syst. 24:2451-2461
5. J. Liu, J. Harkin, Y. Li, and L. Maguire (2015) Low cost fault-tolerant routing algorithm for Networks-on-Chip. Microprocess. Microsyst. 39:358-372
6. J. Liu, J. Harkin, Y. Li, and L. P. Maguire (2016) Fault-tolerant Networks-on-Chip routing with coarse and fine-grained look-ahead. IEEE Trans. Comput. Des. Integr. Circuits Syst. 35:260-273
7. S. Cawley, F. Morgan, B. McGinley, S. Pande, L. McDaid, S. Carrillo, and J. Harkin (2011) Hardware spiking neural network prototyping and application. Genet. Program. Evolvable Mach. 12:257-280
8. H. de Garis, C. Shuo, B. Goertzel, and L. Ruiting (2010) A world survey of artificial brain projects, Part I: large-scale brain simulations. Neurocomputing 74:3-29
9. H. Markram (2006) The blue brain project. Nat. Rev. Neurosci. 7:153-160
10. K. A. Boahen (2000) Point-to-point connectivity between neuromorphic chips using address events. IEEE Trans. Circuits Syst. II Analog Digit. Signal Process. 47:416-434
11. R. Wang, T. J. Hamilton, J. Tapson, A. van Schaik, A. Topology, and G. Dp (2014) An FPGA design framework for large-scale spiking neural networks. In: IEEE International Symposium on Circuits and Systems, 457-460
12. S. Jovanovic, C. Tanougast, C. Bobda, and S. Weber (2009) CuNoC: A dynamic scalable communication structure for dynamically reconfigurable FPGAs. Microprocess. Microsyst. 33:24-36
13. W. J. Dally and B. Towles (2001) Route packets, not wires: on-chip interconnection networks. In: Proceedings of the 38th Design Automation Conference, 684-689
14. L. Benini and G. De Micheli (2002) Networks on chips: a new SoC paradigm. IEEE Comput. 35:70-78
15. J. Schemmel, J. Fieres, and K. Meier (2008) Wafer-scale integration of analog neural networks. In: Proceedings of the International Joint Conference on Neural Networks, 431-438
16. X. Jin, M. Lujn, L. A. Plana, S. Davies, S. Temple, and S. B. Furber (2010) Modeling spiking neural networks on SpiNNaker. Comput. Sci. Eng. 12:91-97
17. S. Pande, F. Morgan, S. Cawley, T. Bruintjes, G. Smit, B. McGinley, S. Carrillo, J. Harkin, and L. McDaid (2013) Modular neural tile architecture for compact embedded hardware spiking neural network. Neural Process. Lett. 38:131-153

18. L. P. Maguire, T. M. McGinnity, B. Glackin, A. Ghani, A. Belatreche, and J. Harkin (2007) Challenges for large-scale implementations of spiking neural networks on FPGAs. Neurocomputing 71:13-29
19. L. Wan, Y. Luo, S. Song, J. Harkin, and J. Liu (2016) Efficient neuron architecture for FPGA-based spiking neural networks. In: Proceedings of the 27th Irish Signals and Systems Conference, 1-6
20. Y. Luo, Q. Fu, J. Liu, J. Harkin, L. McDaid, and Y. Cao (2017) An extended algorithm using adaptation of momentum and learning rate for spiking neurons emitting multiple spikes. In: Proceedings of the International Work Conference on Artificial Neural Networks, 1-11
21. J. J. Wade, L. J. McDaid, J. A. Santos, and H. M. Sayers (2010) SWAT: A spiking neural network training algorithm for classification problems. IEEE Trans. Neural Networks 21:1817-1830.
22. J. Harkin, F. Morgan, L. McDaid, S. Hall, B. McGinley, and S. Cawley (2009) A reconfigurable and biologically inspired paradigm for computation using Network-on-Chip and spiking neural networks. Int. J. Reconfigurable Comput. 2009:1-13