IEEE Asian Solid-State Circuits Conference
November 4 – 6, 2019
The Parisian Macao, Macao SAR, China

An Asynchronous Reconfigurable SNN Accelerator With Event-Driven Time Step Update

Jilin Zhang*, Hui Wu*, Jinsong Wei†, Shaojun Wei*‡, Hong Chen*‡
*Institute of Microelectronics, Tsinghua University,
†School of microelectronics, University of Science and Technology of China, Hefei, China
‡Beijing National Research Center for Information Science and Technology, Beijing, China
Email: hongchen@tsinghua.edu.cn

***Abstract*—In this paper, we put forward an asynchronous spiking neural network (SNN) accelerator with 1024 neurons and 1 million synapses, which is reconfigurable in terms of network connection and neuron parameters. Bundled data asynchronous circuits are adopted to design the neuromorphic computation core and mesh network. Multicast communication is used to transmit packet among and within each core for less packet transmission and better energy efficiency. A novel time step update mechanism, which updates neurons in an event-driven manner without considering the chip-wide activity of other unrelated neurons, is proposed to improve the performance of speed. The SNN accelerator is verified by classifying MNIST handwritten digit with Xilinx VC707 FPGA. The results show that the accelerator achieves 98% accuracy with MINST database, and more than 1 GIPS/W energy efficiency which is 32 times better than previous work.**

Keywords—SNN; asynchronous circuit; FPGA; neuromorphic computation

I. INTRODUCTION

Spiking neural network (SNN), getting inspiration from brain, approaches the energy efficiency of biology by using temporally sparse connection, reduced precision, and approximate computation. However, SNN is inherently parallel in architecture whereas today's von Neumann CPU architectures and GPU variants are serial processing architecture. As a result, it is difficult for CPU and GPU to realize SNN to solve complex problem effectively. When implemented on parallel hardware, like FPGA, SNN can take full advantage of its inherent parallelism and run orders of magnitude faster than software simulations; thus, becoming appropriate for real-time applications [1].

Several SNN hardware accelerators on FPGA have been proposed in the scientific literature. NeuroFlow [2] represented the state of art in the field of FPGA architectures for SNN. It can simulate up to 600,000 neurons by six FPGA boards with 30-40W power consumption per FPGA board. Authors in [3] implemented a 1,440 neurons network with power consumption up to 8.5W when executing emulation with a 100 MHz clock on the FPGA. Researcher in [4] proposed Minitaur, which is an event-based implementation of SNN with 65,000 neurons with the power consumption of 1.5W.

All these accelerators aim to realize large-scale neural network. However, they consume too much power to be used in embedded applications. As we know, asynchronous circuits have the advantages of power efficiency because of its event-driven nature. That is, asynchronous circuits only work when data needs to be processed, otherwise it will remain idle and consume little power. This property makes asynchronous circuits suitable for the implementation of SNN because the spikes in SNN are sparse in terms of space and time. In this paper, we propose an asynchronous reconfigurable SNN accelerator with event-driven time step update mechanism to achieve flexibility, high performance and energy efficiency.

II. LINK-JOINT CIRCUITS

We adopt link-joint-based asynchronous bundled data circuits to design our SNN accelerator. The link-joint is a model proposed by asynchronous research center [5] which concludes all the self-timed asynchronous controller with a concise and comprehensive structure. The asynchronous controller with Click element [6] using link-joint model is shown in Fig. 1, in which every link has four handshake signals: *empty*, *full*, *fill*, and *drain*. A *full* signal indicates that the link's data signals are stable and valid. The link presents *full* signal at its output end to let a receiver know whether or not it is full. An *empty* signal indicates that the link's data have been released and may be replaced by fresh data. The link presents an *empty* signal at its input end to let a sender know whether or not it is empty. A joint may act when some or all of its input links are full and some or all of its output links are empty. When it acts, it computes results from data it gets from its full input links, and passes these results to some or all of its empty output links. It then fills selected empty output links, and drains selected full input links, destroying the conditions that enabled its action. Links that it fills or drains begin their data transport actions to carry results away or fetch fresh input data [5].

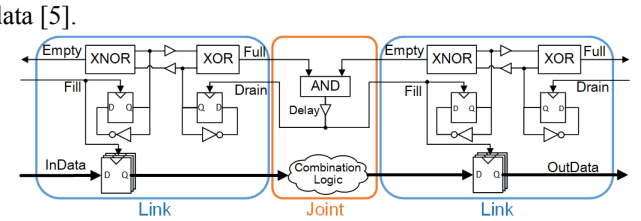

Fig. 1. Click based link-joint circuit.

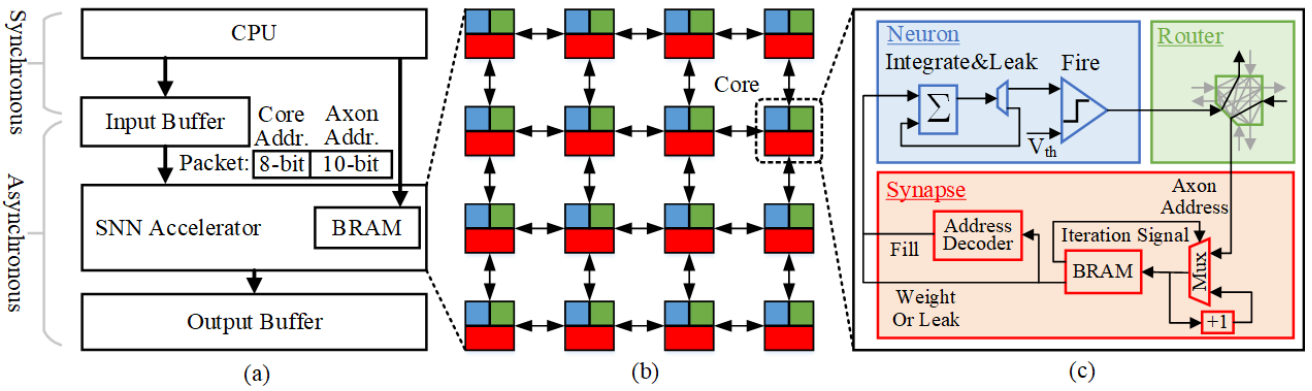

Fig. 2. (a) The System architecture of our SNN accelerator, (b) mesh network (16 cores, 1024 neurons, 1 million synapses), (c) neuromorphic core.

III. Design of the SNN Accelerator

A. System Overview

The system architecture of the SNN accelerator is illustrated in Fig. 2 (a). The address information of a spike is stored in a spike packet, including 8-bit-dual-rail address of its target core and 10-bit address of its axon, the input spike packets of the mesh network are first stored in the input buffer. The configuration information from CPU will be stored in the block RAM (BRAM) in each core. The SNN accelerator is comprised of 1024 neurons and 1 million synapses, which are divided into 16 neuromorphic cores, and each core is connected by a mesh network (shown in Fig. 2(b)). The spike packet from input buffer is multicast throughout the mesh network, and each core is responsible for receiving and sending the spike packet to neuron for the integration operation, as shown in Fig. 2(c). The final results will be stored in the output buffer.

The operation of our accelerator can be divided into three phases. The first one is configuration phase, in which the network connection, synapses weight, neuron leak and threshold voltage will be determined according to the configuration information stored in BRAM. The numbers of layer, core, synapse, and the topology of the SNN are reconfigurable for different applications. The second phase is integration phase. The spike packets from input buffer are sent to the BRAM of its target core, and weight and the address of its target neuron will be read from BRAM based on the axon address in the spike packets. The target neuron will receive the weight and perform integration operation by adding the weight into its membrane potential. The third phase is called time step update (TSU) phase, in which a TSU packet is used to update each core's time step in an event-driven manner. In this phase, the leak values and addresses of the neuron whose time step needs to be updated will be read form BRAM. The value of the leak will be subtracted from the neuron's membrane potential, and the new membrane potential will be compared to the threshold voltage. If it exceeds threshold voltage, a spike will be generated and sent to the next neurons using address event representation (AER) circuit [7].

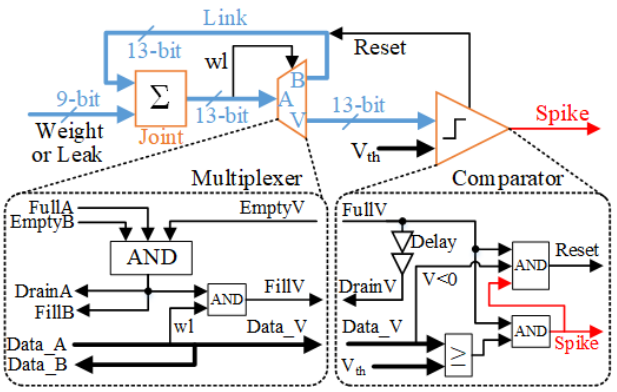

Fig. 3. Asynchronous neuron deisgn.

B. Design of Neurons

The Leak-Integrate-and-Fire (LIF) model is adopted to design the neuron circuit. The operation of LIF neuron is described in equation (1):

$$u(t+1) = u(t) + \sum sw - L \quad (1)$$

where $u(t)$ denotes the membrane potential at time step t, s is an input spike, w is the weight of that spike, L is the leak. LIF neuron integrates the weight of all the spike into its membrane potential across time step along with leak. When its membrane potential exceeds the threshold voltage, the neuron will generate spike and pass it to the next neurons.

The structure of the LIF neuron is depicted in Fig. 3, in which the blue line and orange polygon represent link and joint respectively. Here we regard the leak of neuron as an inhibitory synapse (i.e. synapse with negative weight). In integration phase, 8-bit weight and 1-bit *wl* signal used to distinguish weight (when *wl* is logic '0') and leak (when *wl* is logic '1'), will be the input of target neuron. Here we choose 8-bit weight to allow a relatively high accuracy and low memory footprint. The weight value will be integrated into neuron's membrane potential and the updated membrane potential will not be sent into the comparator in Fig. 3 in this phase because *wl* is now logic '0'. 13-bit is chose to represent membrane potential so that the sum of spike's weight in one time step will not overflow. In TSU phase, the leak will be

sent and integrated to membrane potential, and the updated membrane potential will be sent into the comparator in Fig. 3. If it exceeds the threshold voltage, this neuron will generate a spike and reset its membrane potential to 0. If the membrane potential is small than 0, which can be caused by inhibitory synapse or leak, neuron will reset its membrane potential into 0 without producing any spike.

C. Design of Routers

The schematic of the router in each core is shown in Fig. 4, in which five input and output channels are local, north, south, east, and west channels respectively. The local channel is connected to the core, and other four channels are connected to other adjacent routers in the cores on its north, south, east, and west sides respectively. As discussed above, the spike packet contains 8-bit destination core address and 10-bit axon address. We adopt dual rail encoding to encode the core address in order to multicast packet among cores. Each bit in the address for a set of destination cores is represented by a 2-bit symbol. Each symbol can be 0, 1, or '*'. With '*' indicating that cores with either 0 or 1 at that bit location in their addresses are destinations. When processing the '*' symbol in a router, packets branch to multiple output ports and continue to different destination cores [8].

When the router receives a packet from any input channel, the Routing Logic selects its output channel according to the destination core address and then generates a request signal along with the output channel information. Allocator will grant access for its input channel based on the status of requested output channels, which are represented by the *Empty* signals. When one input channel has been granted, the packet will be sent to the corresponding output channel after Mux.

D. Design of Synapses

As shown in Fig. 2 (c), the synapse circuit contains BRAM, which stores the neuron address, synapse weight, and 1-bit iteration signal to implement multicast operation within a core. An address decoder is used to change 6-bit neuron address from BRAM into 64-bit *fill* signals. To multicast packets within core, BRAM iterates over a list of target neurons for an axon's fan-out distribution. For example, if an axon is connected with multiply neurons in the core, after the first neuron's address and synapse weight is read from BRAM, an iteration signal will make BRAM continue to read the next address of neuron and its synapse weight.

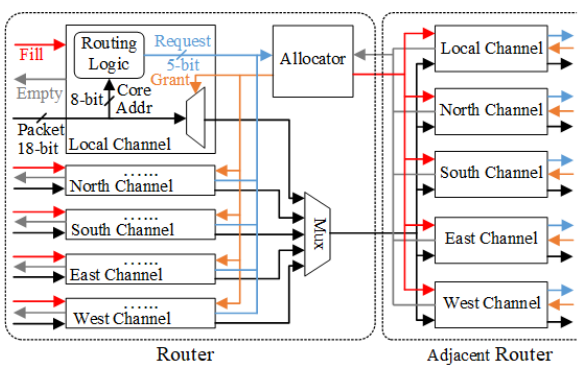

Fig. 4. The schematic of the router in mesh network.

IV. Event-driven Time Step Update Mechnism

There are two families of neuron update mechanisms for the implementation of spiking neural networks. One is time step (or clock-driven) update, in which the operation of neurons has been divided into several discrete time step and updated simultaneously at every tick of a clock. The other one is asynchronous (or event-driven) update, in which the neurons are updated only when they receive or emit a spike [9]. The time step update has the drawback of system performance proportional to network size, and the event-driven update is not as widely used as clock-driven for its complicated computation process. In order to take the advantage of both event-driven and clock-driven update, we put forward a new event-driven time step update mechanism, which improves the system performance and enables the system support universal discrete time step update.

As mentioned in section III, the TSU packet is used to update the time step of each core separately as shown in Fig. 5. Once all the spike packets are read out from the input buffer within a time step, a TSU packet from input buffer will be sent into mesh network. The TSU packet will appear in the first layer of neural network, which indicates the spike packets in current time step are all integrated. Then a TSU packet with new target core address appears in the second layer core. Similarly, when core in the second layer receives all the TSU packet from first layer core, it will send a new TSU packet target in the third layer cores, and go into the next time step. Finally, every neuron's time step can be updated asynchronously in an event-driven manner, which avoids the worst-case chip-wide activity of other unrelated neurons.

However, in order to ensure all the neurons are updated at an appropriate time, the TSU packet needs to stay between the adjacent spike packets with time step t and $t+1$ respectively. This requires the routing method of mesh network to be deterministic, which means the routing path between any source and destination core needs to be fixed and not affected by the network congest. Indeterministic routing will adjust the routing path based on the congest level of network. It may result in fewer hops than time step t packet to reach target neuron and early updates for TSU packet.

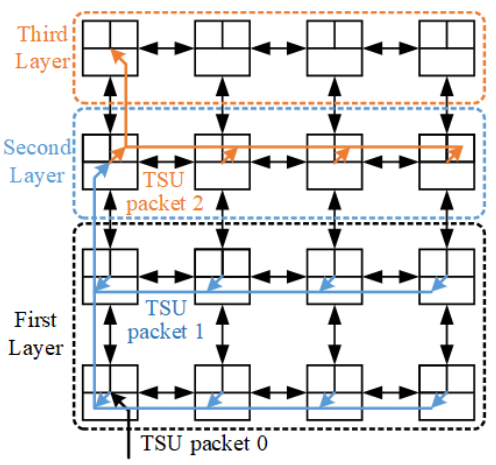

Fig. 5. Time step update example.

TABLE I. EXPERIMENT RESULTS WITH COMPARISON OF PREVIOU WORK

	TVLSI 2014[4]	ISCAS 2016[10]	IJCNN 2015[11]	This work
Platform	Spartan-6 LX150	Spartan-6 LX150	SpiNNaker (ASIC)	XC7VX6 90T
Neuron number	65K	65K	2400	1024
Synapse number (million)	16	33	48	1
Weight precision	16-bit fix point	16-bit fix point	22-bit fix point	8-bit fix point
Clock rate (MHz)	75	132	150	-
MINST accuracy	92%	94%	95%	98%
MINST classification latencies (ms)	9.2	2	20	1.1
Peak Performance (MIPS)	19	53.5	16	726
Power(idle/active, W)	1.1/1.5	-	-/0.50	0.36/0.70
Energy efficiency (GIPS/W)	0.013	-	0.032	1.037

V. EXPERIMENTAL RESULTS

We implemented our design with Xilinx FPGA VC707 board, and a four-layer feedforward neural network is adopted to perform handwritten digital recognition from MINST database. This 784-512-384-10 network has two hidden layer with neurons fully connected between layers. This network is trained with a feedforward backpropagation method on PC with the method discussed in [12]. The experimental platform is shown in Fig. 6, and the results show that accelerator has the final error rate of 2%, achieving 98% accuracy. The comparison results between our accelerator and previous works are illustrated in table I, from which we find that our accelerator is 2 times faster than that in [10], and the energy efficiency is 32 times better than that in [11].

The Demo link is https://vimeo.com/342519869 (password: asscc2019zjl).

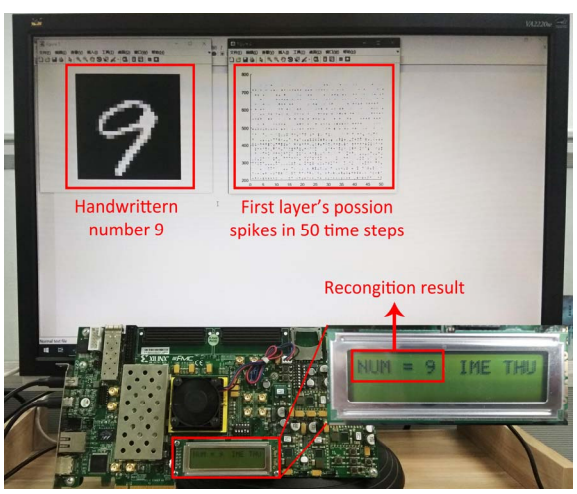

Fig. 6. The experimental platform.

VI. CONSULSION

In this paper, we implemented an asynchronous reconfigurable SNN accelerator, and a novel time step update mechanism is proposed to improve the speed. Packet is multicast among and within cores to improve energy-efficiency. The SNN accelerator has been verified on Xilinx VC707 board, and the results show that the accelerator achieves 98% accuracy at more than 1 GIPS/W of energy-efficiency, which is 32 times better than other works. The performance will be further improved in the future.

ACKNOWLEDGMENT

This work is supported by National Natural Science Foundation of China (No. 61674090).

REFERENCES

[1] L. P. Maguire et al. "Challenges for large-scale implementations of spiking neural networks on FPGAs," in Neurocomput. 71, 1-3 (December 2007), 13-29.

[2] Cheung Kit, Schultz Simon R., Luk Wayne, "NeuroFlow: A General Purpose Spiking Neural Network Simulation Platform using Customizable Processors," Frontiers in Neuroscience, vol. 9, p.516, 2016.

[3] Pani Danilo, Meloni Paolo, Tuveri Giuseppe, et al., "An FPGA Platform for Real-Time Simulation of Spiking Neuronal Networks," Frontiers in Neuroscience, vol. 11, p. 90, 2017.

[4] D. Neil and S. Liu, "Minitaur, an Event-Driven FPGA-Based Spiking Network Accelerator," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 22, no. 12, pp. 2621-2628, Dec. 2014.

[5] M. Roncken, I. Sutherland, C. Chen, Y. Hei, et al., "How to think about self-timed systems," 2017 51st Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, 2017, pp. 1597-1604.

[6] A. Peeters, F. t. Beest, M. d. Wit and W. Mallon, "Click Elements: An Implementation Style for Data-Driven Compilation," 2010 IEEE Symposium on Asynchronous Circuits and Systems, Grenoble, 2010, pp. 3-14.

[7] K. A. Boahen, "Point-to-point connectivity between neuromorphic chips using address events," in IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. 47, no. 5, pp. 416-434, May 2000.

[8] G. K. Chen, R. Kumar, H. E. Sumbul, P. C. Knag and R. K. Krishnamurthy, "A 4096-Neuron 1M-Synapse 3.8-pJ/SOP Spiking Neural Network With On-Chip STDP Learning and Sparse Weights in 10-nm FinFET CMOS," in IEEE Journal of Solid-State Circuits, vol. 54, no. 4, pp. 992-1002, April 2019.

[9] R. Brette, M. Rudolph, T. Carnevale, M. Hines, D. Beeman, J. M. Bower, et al., "Simulation of networks of spiking neurons: A review of tools and strategies," J. Comput. Neurosci., vol. 23, no. 3, pp. 349–398, 2007.

[10] I. Kiselev, D. Neil and S. Liu, "Event-driven deep neural network hardware system for sensor fusion," 2016 IEEE International Symposium on Circuits and Systems (ISCAS), Montreal, QC, 2016, pp. 2495-2498.

[11] E. Stromatias, D. Neil, F. Galluppi, M. Pfeiffer, S. Liu and S. Furber, "Scalable energy-efficient, low-latency implementations of trained spiking Deep Belief Networks on SpiNNaker," 2015 International Joint Conference on Neural Networks (IJCNN), Killarney, 2015, pp. 1-8.

[12] Diehl, Peter & Neil, Dan & Binas, Jonathan & Cook, Matthew & Liu, Shih-Chii & Pfeiffer, Michael. (2015). Fast-Classifying, High-Accuracy Spiking Deep Networks Through Weight and Threshold Balancing. International Joint Conference on Neural Networks, IJCNN. 10.1109/IJCNN.2015.7280696.