Modeling and Simulation of Asynchrony in Neuromorphic Computing

Zareen Sheikh, Vivek Khetade

Abstract: ***Neuromorphic computing is a non-von Neumann architecture which is also referred to as artificial neural network and that allows electronic system to function in the same manner as that of the human brain. In this paper we have developed neural core architecture analogous to that of the human brain. Each neural core has its own computational element neuron, memory to store information and local clock generator for synchronous functioning of neuron along with asynchronous input-output port and its port controller. The neuron model used here is a tailor-made of IBM TrueNorth's neuron block. Our design methodology includes both synchronous and asynchronous circuit in order to build an event-driven neural network core. We have first simulated our design using Neuroph studio in order to calculate the weights and bias value and then used these weights for hardware implementation. With that we have successfully demonstrated the working of neural core using XOR application. It was designed in VHDL language and simulated in Xilinx ISE software.***

Index Terms**: *IBM TrueNorth, Neuromorphic Computing, Port Controller, Synchronous-Asynchronous Design.***

I. INTRODUCTION

Neuromorphic computing is brain inspired computation. Human brain has lakes of neurons with synapse and dendrites that has the ability to store information, ability to retain fast response and capability to understand and respond. Computer systems design is different from the organization of human brain. Clock is used to synchronize activities while the brain runs in an asynchronous manner. There is a clear partition between memory and computation circuits [1], whereas in brain the memory and computation appear to be tightly integrated. Neuromorphic computing is a computing system which follows the structure and principle of biological computation in artificial substrate such as silicon, in order to achieve similar computational performance [2]. Digital implementation of spiking neurons is more efficient and enables one to one correspondence between hardware and software [3].

The neuron present in the human brain performs three basic functions i.e. it receives information from other neurons, integrates it to determine whether the information should be allowed to pass and transfers the information to the other neurons or muscles or glands. Fig. 1 shows the biological structure of neuron which consists of cell body called soma, many short branching processes called dendrites and other processes longer than dendrites known as axon. The detailed representation of the biological structure is explained in table I below.

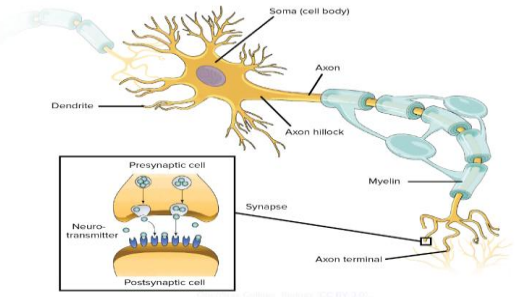

Fig. 1: Biological Structure of Neuron

The brain is what it is because of the structural and functional properties of its interconnected neurons. The basic function of neurons is to transfer information to other parts of the body [4]. A neuron communicates with each other with the help of synapse. The neuron send message by releasing a chemical messenger called neurotransmitter into the synaptic cleft. The neurotransmitter cross the synapse and gets attached to the key side of the receptor of the next neuron. When the neurotransmitter gets attached to the receptors they cause a change inside the receiver neuron and the message is delivered.Similarly neuromorphic chips are basically an electronic system which performs the mimicry of human brain. Neuromorphic hardware is a link between an electronic system and human brain. The component used for implementing neuromorphic hardware is micro and nano scale transistor. The electrochemical function of human neurons is replicated with the help of non-linearity of silicon transistor. Transistors in neuromorphic chips are used as switches and they also provide neuroplasticity which is a characteristic of human brain that enables it to modify synapse while processing the data. There is a major requirement of neuromorphic computing because it provides flexibility, strong memory, asynchronous spiking signal and timing.

.Components in Biological Neuron	Functions Performed
Dendrites	Fine hair like extension helps neurons to receive incoming stimuli.
Soma (cell body)	Helps in directing incoming impulses from the dendrites to the axon.

Revised Manuscript Received on July 05, 2019.

Zareen Sheikh, Department of Electronics Engineering, Shri Ramdeobaba College of Engineering and Management, Nagpur, India.

Vivek Khetade, Assistant Professor, Department of Electronics Design Technology, Shri Ramdeobaba College of Engineering and Management , Nagpur, India.

Axon Hillock	- Special part present in the soma. -Connects to the axon. -Trigger zone because the action potential starts here.
Axon	Long, cylindrical shaped which transmits the electrical signal from the soma (cell body) to the axon terminal.
Myelin	Increases the speed of electrical signals.
Synapse	-A junction between two neurons. -Used to determine the direction to the nerve signal that travels through the nervous system.
Presynaptic cell	-When an action potential enters into the axon terminal the Presynaptic neuron transmit/fires the neurotransmitter. -In mammals, the presynaptic terminal operates exactly in the same way in both the central and peripheral nervous system.
Neurotransmitter	Chemical messenger that transmit signal across neuromuscular junction or chemical synapse.
Postsynaptic cell	Receives neurotransmitter and fires an action potential if the received neurotransmitter strong.
Axon terminal	-Enlarge and club or button shaped present at the axon ending. -Helps in connecting to the other neuron or nerve cells. -Contains neurotransmitters which release them at the synapse. -The neurotransmitter contains electrical impulse and they transmit these impulses to the target cell.

I: Various Components in Biological Neuron.

Our main contribution in this paper is a synchronous-asynchronous neural core architecture which consists of neuron block, a random number generator, memory, local clock generator, plus 4 phase bundled data protocol for asynchronous interconnection. The design of neuron block is taken from IBM's TrueNorth neuron design [5]. It consists of synapse unit, leak and leak reversal unit, integrator unit, threshold and reset unit. In order to implement event-driven communication circuit we used asynchronous technique and for computation we used synchronous technique.In this paper, we present the design of the neural core and its asynchronous-synchronous design flow. In section II, we review work related to neural core and their architecture. In section III, we described our neural network core architecture. In section IV, we have described different blocks present in our neural core. In section V, we have modeled XOR with the neural core. In section VI, we demonstrated all the results obtained. In section VII, we presented our conclusion

II. LITERATURE REVIEW

There are various designs that have demonstrated neuromorphic computation, some of the recent neuromorphic projects are shown in Table II.

III. PROPOSED ARCHITECTURE

The proposed Neuromorphic Computing is based on biological neural network. Where the neurons receive signals through its dendrites process it and then transfer the signal to next successive neuron through its axon terminal. The behavior of biological neuron is asynchronous based on the stimulus of previous neurons. The proposed architecture consists of honey comb structure of neural core. It has asynchronous connectivity with six neighboring neural core through its six ports as shown in figure 2(a). Neural core is the basic building block of the architecture. It receives and transmits the information through their ports. It processes the signals from other neural core and transfer the signal to another connected neural core.

Figure 2(b) shows the single neural core with six ports. Each port has separate lines for input port and output port along with the handshake signal i.e. request and acknowledgement. This single neural core communicate with six neighboring core through these ports asynchronously using 4 phase bundled data protocol.

The six input port of the neural core are labeled as INE, IE, ISE, ISW, IW, and INW. Whereas the six output ports of the neural core are labeled as ONE, OE, OSE, OSW, OW, ONW. The request and acknowledgement of each input port are labeled as ReqINE, ReqIE, ReqISE, ReqISW, ReqIW, ReqINW, AckINE, AckIE, AckISE, AckISW, AckIW, and AckINW. The request and acknowledgement of each output port are labeled as ReqONE, ReqOE, ReqOSE, ReqOSW, ReqOW, ReqONW, AckONE, AckOE, AckOSE, AckOSW, AckOW, and AckONW. These ports are

Parameter	IBM TrueNorth	SpiNNaker	Neurogrid	BrainScaleS
No. Of Neurons	1 million neurons	1 billion neurons	1 million neurons	512 neurons
Synapse	256 million synapse	1 trillion synapses	6 billion synapses	128K Synapse
Dendrites	256 input axons	Each neuron has 1000 inputs.	-	-
Network	It is a many core processor network on chip with 4096 cores each with 256 programmable simulated neurons.	Real-time spiking neural network	Mixed analog-digital neuromorphic system organized in a tree routing network is a sixteen chip system	Mixed analog-digital design which accelerates the search for intersecting networks
No. Of operation per watt	58 Giga-Synaptic Operations per Second per Watt (GSOPS/W) of peak computational performance and 400 Gega-Synaptic Operation Per Second per Watt (GDOPS/W) of computational energy efficiency	Consumes about 25-36w running in real-time	In real time a total of one million neurons are simulated by Neurogrid for 3.1W approximately	Consumes approximately 1kW at wafer-scale
Year	2014	2011	2009	2015
University	DARPA SyNAPSE	University of Manchester	Stanford university	Heidelberg University
Methodology	The TrueNorth uses Corelet Programming Environment (CPE) and composed algorithm library in order to develop their applications.	SpiNNaker is a microprocessor based system for improving the performance of software simulations. SpiNNaker has fully digital architecture and uses asynchronous message passing network. Network with hundreds of thousands of neuronsand tens of millions synapse has been demonstrated by SpiNNaker using a 48-chip board.	Neurogrid follows continuous time neural dynamics through communication among biophysical component. Neurons from any chip uses multicast communication to send message asynchronously to synapse from any other chip. Similar parameter is been shared by the neurons on the same chip.	BrainScaleS has 200 thousand analog neurons and 400 million addressable synapse and consumes approximately 1kW at wafer-scale. Network for communication at wafer-scale is asynchronous and uses a hierarchical packet based approach in order to connect to a high performance FPGA system.
Chip area	Occupies 4.3 cm^2 in a 28 nm low-power CMOS process.	Integrates 18 microprocessors in 102mm^2 using a 130 nm process.	Chip is of 168mm^2 in a 180 nm process	chip is 5 x 10 mm^2
References	[5], [6], [7], [8]	[9], [10], [11]	[12], [13]	[14]

II: Various Neuromorphic Computing Systems Developed.

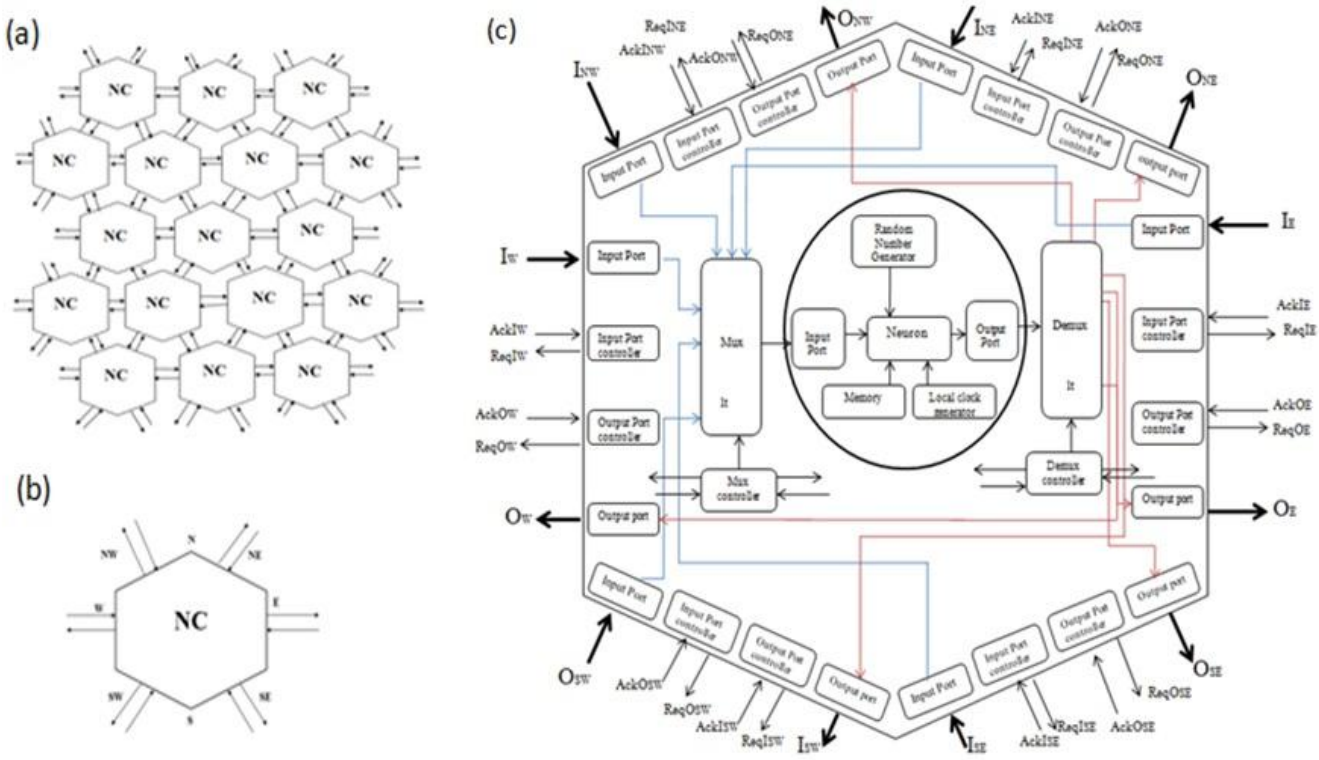

Fig. 2(a): Generic structure of Neural core, Figure 2(b): Single Neural Core, Figure 2(c): Neural core and its internal components

similar to dendrites in biological brain.

Figure 2(c) shows the internal structure core. It consists of synchronous neuron block taken from the TrueNorth architecture [5] and asynchronous port and port controller. The block diagram of synchronous neural core consists of Neurons, Random Number Generator, Memory, Local Clock generator, Input port and Output Port. Based on the incoming spike signals and trained weights the function of neuron block is to generate spike signals for other neurons connected to it. The random number generator is used to support stochastic leak, synapse and threshold function. RAM memory is used to store synaptic connectivity, weight value, leak value, threshold value and membrane potential value. Local clock generator is used to supply the clock signals to the synchronous neuron core. Input ports are used to interface the asynchronous signal with the synchronous signals and output ports are used to interface the synchronous signals with the asynchronous signals. Input port along with port controller is used to receive the asynchronous signal from the neighbouring neural core. There are six input port and six input port controller for six port of neural core. Output port along with the output port controller is used to transmit the asynchronous signal to the neighbouring neural core. There are six output port and six output controller for six port of neural core.

A. Working of Proposed Neuromorphic Computing

We have assumed that the communication between two neurons take place asynchronously with 4-phase bundled data protocol. The computations inside the neural core take place synchronously. The channel used for communication is push type where sender initiates the signal and the receiver is supposed to respond to it. When an incoming spike arrives at the input of the neural core then the input port controller respond to the spike through its acknowledgement signals. The signal from the input port is send to the asynchronous multiplexer which transmit this spike signal to the input of neuron block. The neuron process the signal and update its membrane potential and an output spike generated by the neuron depending upon its input signal and then the output spike is send to the asynchronous demultiplexer through the neuron output port. The asynchronous demultiplexer transmits this synchronous spike signal to the output port for further transfer of signal to other neural core.

IV. DESIGN OF BLOCKS OF PROPOSED ARCHITECTURE

A. Design of Input Port and Input Port Controller

New input may be latched after latch has indicates that it has latched the current data and the handshaking, on the input channel may be complete without any interaction with the output channel. If the latch controller is “normally transparent” then this may lead to excessive power consumption because inputs that make multiple transitions before settling will propagate through successive synchronizer. By using “normally opaque” latch controller, latch will act as a barrier. If handshake latch that is holding a bubble is exposed to a data on its input, the latch controller switches the latch into the transparent mode and when input data have propagated safely into the latch, it will switch the latch back into the opaque mode in which it will hold data. Fig 3(b) shows the PN specification and Fig 3(c) shows the circuit implementation of the normally opaque latch controller. It may be necessary to add some delay into Lt+ to Ain- in order to ensure that input signal has propagated through the latch before Ain-. Furthermore, the duration of

the Lt = 0 pulse that causes the latch to be transparent is determined by the gate delays in the latch controller itself and the pulse must be long enough to ensure safe latching of the data. The latch controller assumes a broad data-validity on its input channel [15].

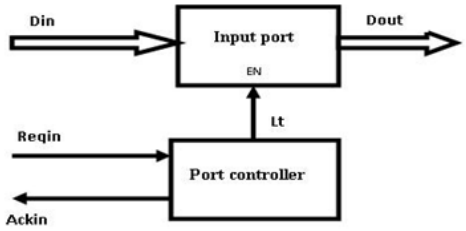

Fig. 3(a)

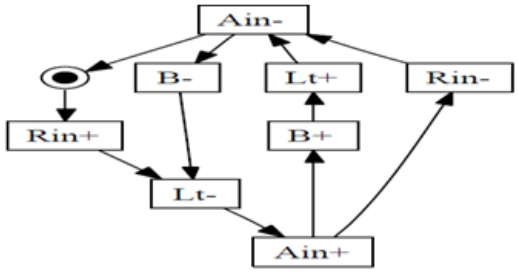

Fig. 3(b)

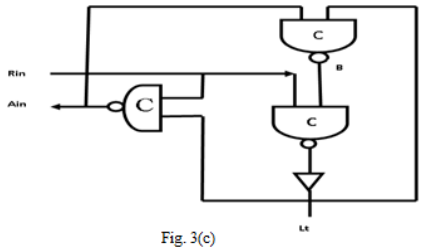

Fig. 3(c)

Fig. 3: (a) Input Port, (b) STG of Input Port, (c) Circuit Diagram of Input Port.

B. Design of Output Port and Output Port Controller

Representation of the decoupled output port is shown in fig. 4(a). It consists of an output port and a port controller. Whenever packet data is ready, it is placed on the data line of decoupled output port along with request line "ri" and port controller will give the acknowledge "ai". "ro is the output request from the output port controller and "ao" is the acknowledgement from the successive micro pipeline . "en" is decoupled output port enable signal which enable the output port. It is designed to comply with the four phase handshake protocol. Fig. 4(b) shows the STG specification of the port controller. Fig. 4(c) shows the standard-c circuit implementation of port controller. This implementation is speed independent i.e. its correctness is independent of gate delay. It is synthesized by Petrify using gC with monotonous cover option [15].

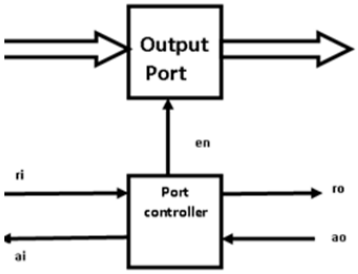

Fig. 4(a)

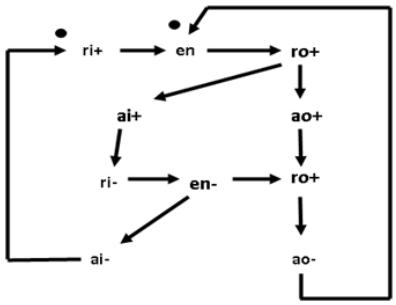

Fig. 4(b)

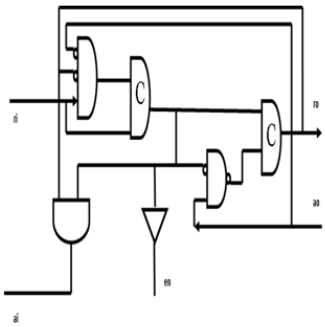

Fig. 4(c)

Fig. 4: (a) Output Port, (b) STG of Output Port, (c) Circuit Diagram of Output Port

C. Design of Asynchronous Mux

A MUX will coordinate the control channel and convey the information and the handshaking of the particular input channel to the output data channel. The other input channel is mistreated [15].

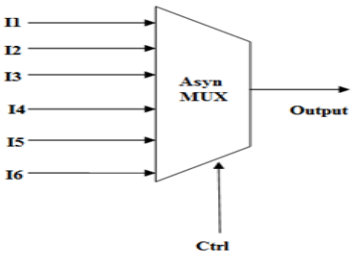

Fig. 5: Asynchronous Multiplexer

D. Design of Asynchronous Demux

A DEMUX will coordinate the control and the data input channels and guide the input to the particular output channel. The other output channel is inactive and in the rest state [15].

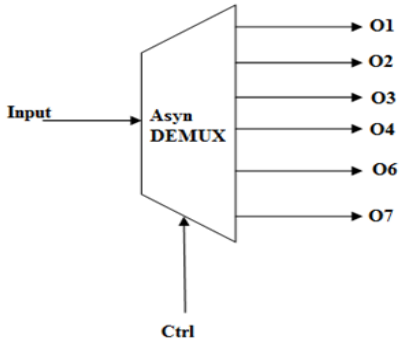

Fig. 6: Asynchronous Demultiplexer

E. Design of Neurons

The structure of neuron block is taken from IBM TrueNorth's Neurosynaptic core [5]. The neuron block used here is dual stochastic and deterministic neuron based on leaky integrates and fire neuron with a stable leak [16]. The block diagram of neuron is shown in figure 7, this leaky integrate and fire neuron consist of synapse integration, leak integration, leak mode, threshold and reset mode. Each neuron receives incoming spikes and integrates it to update its membrane potential. The synaptic input unit actualizes stochastic and deterministic for distinctive weights types with positive and negative values. The leak and leak inversion unit give a stable bias. The function of integrator unit is to add the membrane potential from the past tick with synapse and leak input. The threshold and reset unit is used to compare the membrane potential value with the threshold value. When the incoming spike is integrated a leak value is subtracted from the membrane potential and then the updated membrane potential value is compared with the threshold value and if the membrane potential value exceeds the threshold value then a spike is generated.

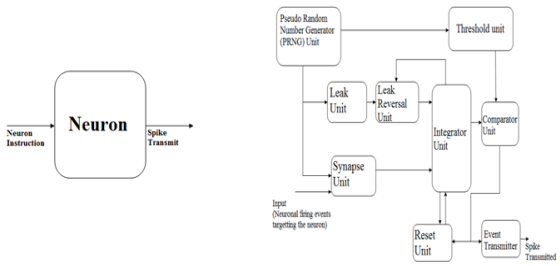

Fig. 7: Neuron Block Diagram

V. MODELING OF XOR WITH NEURAL CORE

The application for the neural core is developed using neuroph studio and simulated with Xilinx ISE software tool. Figure 8(a) shows 2 inputs XOR problem developed using neuroph studio which consist of five neurons. The number of neurons is chosen depending upon the application selected. Here the first two neural cores receive incoming spikes (information) and then further transfer the information to other neurons. The final result is obtained at the output of neuron five. The neural network is trained using traditional backpropogation algorithm. Inputs are provided along with the target value for training. Figure 8(b) shows the training graph for 2 inputs XOR. The training of neural network is done to reduce the error rate.

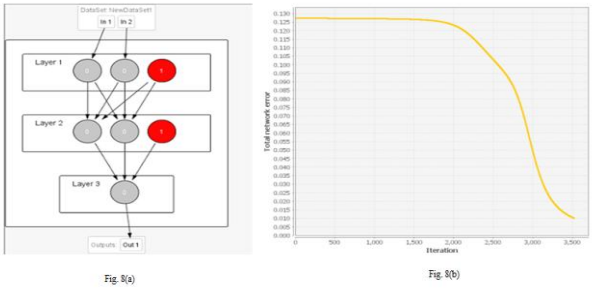

Fig. 8: (a) Network Structure for 2 input XOR, (b) Training Graph

Sr. no.	Parameter	Value
1.	Number of Layers	3
2.	Number of Neurons per Layer	min: 1 and max: 2
3.	Number of Training Iterations	3500
4.	Number of Neurons in the Hidden Layer	2
5.	Learning rate	0.2
6.	Training Type	Training by minimum error
7.	Minimum error	0.01
8.	Momemtum	0.7

III. Parameters used for Training in Neuroph Studio

A. Algorithm for XOR Implementation

Algorithmically, the approach used for implementing XOR is as follows:

a. Wait for the input signal.
b. Read neuron data from the memory.
c. Apply leak value.
d. Check threshold value.
e. Compare threshold value with the membrane potential value.
f. If membrane potential value is greater than threshold value then the spike is generated.

By using backpropogation algorithm the synapse weights are calculated. The XOR is implementated in hardware using VHDL language. The value of input is taken as it is whereas the synapse weight values are calculated.

B. Mapping of Application on the Proposed Architecture

Figure 8(c) shows the 2 input neural networks with four different outputs. There are 5 neurons present in our design with 2 inputs and 1 output. Each neurons produce different outputs for 4 different pairs of inputs. Here the value in dashed box shows the output produce through neuroph studio which is in floating point and the value in the other box shows its fixed point representation which is obtained through Xilinx simulation. The red box indicated the outputs for 00 inputs, the green box indicates the outputs for 10 inputs, the purple box indicates the outputs for 11 inputs and the blue box indicates the outputs for 01 input.

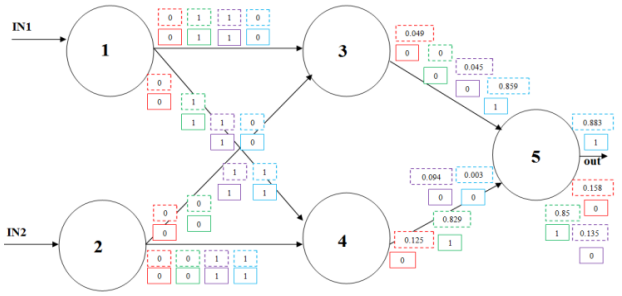

Figure 8(c): Core outputs for software and hardware simulation

Figure 8(d) shows the mapping of XOR application on neural core. There are 5 neural cores connected together in a network. The input layer consists of 2 neural cores which receive input, the hidden layer consists of 2 neural cores and the output layer consists of 1 neural core from where the final output will be generated.

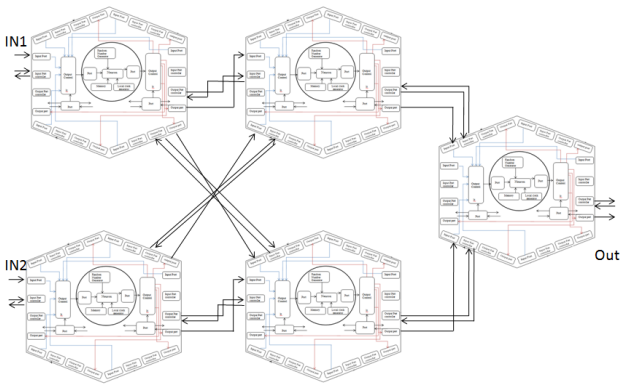

Figure 8 (d): XOR implementation using neural core.

Figure 8(e) shows the synchronous simulation wave obtained through XOR application. It consists of 2 inputs indicated as i1 and i2 which represents four different values i.e., 00, 01, 10 and 11. n1a represents the signal between input neural core 1 and the hidden neural core 3. n1b represents the signal between input neural core 1 and hidden neural core 4. n2a represents the signal between input neural core 2 and hidden neural core 3. n2b represents the signal between input neural core 2 and hidden neural core 4. n3 represents the signal between hidden layer neural core 3 and output layer neural core 5. n4 represents the signal between hidden layer neural core 4 and output layer neural core 5. Here y represents the output values for four different inputs.

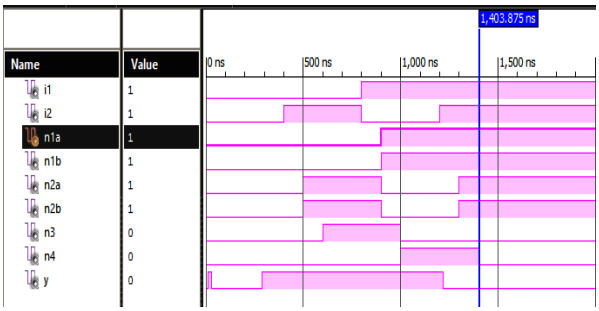

Figure 8(e): Synchronous Simulation Wave for XOR Application

Similarly, figure 8(f) represents the asynchronous simulation waveform for XOR implementation. Here the request and acknowledge signal represents the asynchronous connectivity between neural cores. We have used 4-phase bundled data protocol for asynchronous connectivity. As the sender affirm the data set the request to '1' and the receiver set the acknowledge to '1' after absorbing the data. The sender respond to the acknowledge signal by setting request to '0' and the receiver acknowledge by setting acknowledge '0' and at this stage the sender is ready to initiate the next communication cycle.

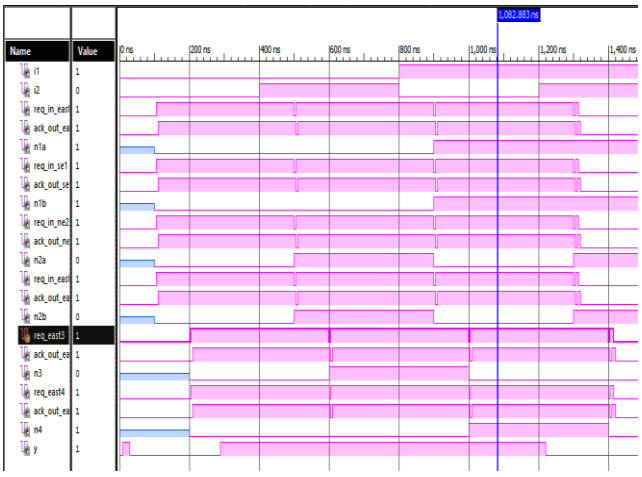

Figure 8(f): Asynchronous Simulation Wave for XOR Application.

VI. RESULT AND ITS ANALYSIS

A. Simulation of Synapse Unit

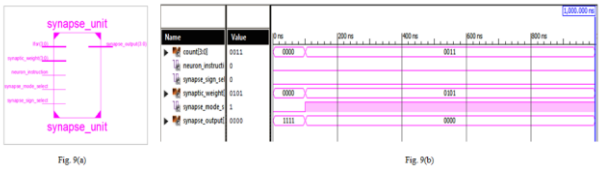

Fig. 9(a) Fig. 9(b)

Figure 9: (a) RTL view of synapse, (b) Synapse waveform

The RTL for synapse unit is shown in figure 9(a). When the neuron instruction becomes zero then 0000 is send to the deterministic path and stochastic path. Then this 0000 value is compared with lfsr value which is 0011. Here the comparator output becomes 1 as the lfsr value is greater than 0000 value. Hence a 0 value appear on the stochastic pathway. Now as the synapse mode select becomes 1 the synapse output obtain is 0000. The waveform for synapse unit is shown in figure 9(b).

B. Simulation of Leak Unit

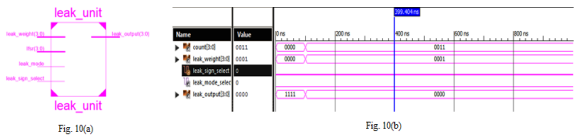

Fig. 10(a) Fig. 10(b)

Figure 10: (a) RTL view of leak unit, (b) leak unit waveform.

Figure 10(a) shows the RTL for leak unit. In leak unit the stored leak weight is transferred to both the deterministic pathway and stochastic pathway. The stored leak weight value i.e. 0001 is compared with lfsr value which is 0011. As the lfsr value is greater than the stored leak weight value so the comparator output becomes 1. A 0 value is applied to the stochastic pathway. Now as the leak mode select becomes 0 the leak output obtain is 0000. The waveform for leak unit is shown in figure 10(b).

C. Simulation of Leak Reversal Unit

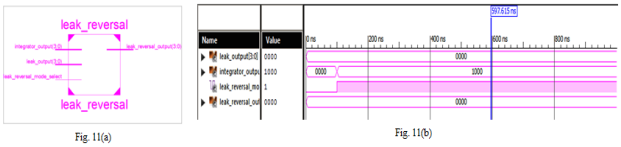

Fig. 11(a) Fig. 11(b)

Figure 11: (a) RTL view of leak reversal unit, (b) leak reversal unit waveform

The RTL for leak reversal output is shown in figure 11(a). The leak output is applied as an input to the leak reversal unit. The leak output obtained is 0000. The integrator output which is a membrane potential value from the integrator unit is 1000. The leak reversal mode select determines whether the neuron will operate in leak reversal mode or monotonic leak mode. Here the leak reversal mode select is set to 1 so the neuron will operate in monotonic leak mode and the leak reversal output becomes 0000. The waveform for leak reversal unit is shown in figure 11(b).

D. Simulation of Integrator Unit

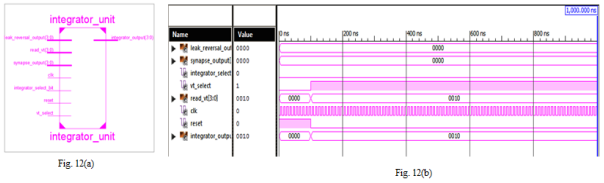

Fig. 12(a) Fig. 12(b)

Fig. 12: (a) RTL View of Integrator Unit, (b) Integrator Unit Waveform

The RTL view of integrator unit is shown in figure 12(a). The input to the integrator unit is output of the leak reversal unit and synapse unit. The integrator select bit selects one of the inputs. The vt_select bit selects either the read_vt value or the integrator output value. The read_vt value is nothing but the membrane potential value. The integrato unit consist of adder which adds the leak reversal unit or synapse units output to the membrane potential value. The output obtained through adder is the updated membrane potential value which is nothing but the integrator output. The waveform obtained for reset unit is shown in figure 12(b).

E. Simulation of Threshold Unit

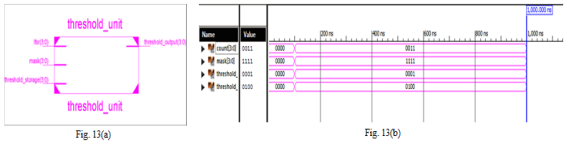

Fig. 13(a) Fig. 13(b)

Figure 13: (a) RTL view of Threshold unit, (b) Threshold Waveform

The Threshold unit was designed using VHDL and its RTL obtained is shown in figure 13(a). The input provided to Threshold unit is from lfsr, mask and threshold storage. First the lfsr and mask unit is multiplied in order to obtain intermediate pseudo random value. Here the lfsr value is 0011 and mask value is 1111 so the intermediate pseudo random value obtain is 0011. Now the obtained intermediate pseudo random value is added with the stored threshold value which is 0001. So the threshold value obtain is 0100. This simulation is carried for typical cases. The waveform for Threshold unit is shown in figure 13(b).

F. Simulation of Reset Unit

The RTL view of reset unit is shown in figure 14(a). There are three inputs to the reset unit i.e. vreset which stores a constant 0 value, integrator output which is a membrane potential value from the integrator unit and threshold output obtained which is 0100. Depending upon the configuration value selected one of the input appear as an output to the reset unit. Here configuration value selected is 01 so the reset output becomes 0010. The waveform obtained for reset unit is shown in figure 14(b).

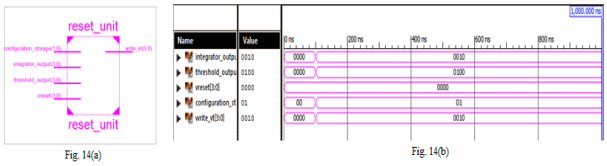

Fig. 14(a) Fig. 14(b)

Figure 14: (a) RTL view of reset unit, (b) reset unit waveform

G. Simulation of Neuron

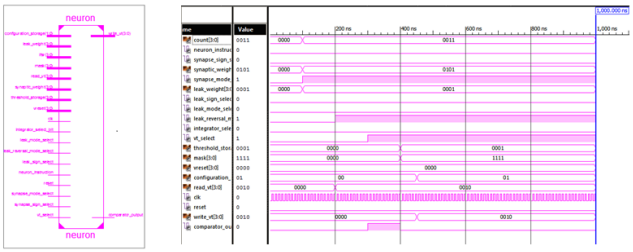

Fig. 15(a) Fig. 15(b)

Figure 15: (a) RTL view of Neuron, (b) Neuron waveform

Parameter	Value
Latency	2.007ns
Response Time	2.406ns
Bandwidth	1Ghz

IV. Design Parameter

VII. CONCLUSION

In this paper we have presented the implementation of neural core. Our proposed architecture is modular and it is possible to increase or decrease the number of neural core and layers depending upon the application. Our design involves mixed asynchronous and synchronous approach. We have demonstrated example application such as XOR which requires five neural cores fully connected to each other in an asynchronous manner. Here first our network is trained using Neuroph Studio and the obtained weights are written in VHDL package. We have synthesized our network in Xilinx and obtained the RTL schematic and its corresponding clock waveform.

REFERENCES

1. Hsin-Pai Cheng, Wei Wen, Chang Song, Beiye Liu, Hai (Helen) Li and Yiran Chen, "Exploring the Optimal Learning Technique for IBM TrueNorth Platform to Overcome Quantization Loss," Proceedings of IEEE/ACM International Symposium on Nanoscale Architectures, 2016.
2. Andrew S. Cassidy, Jun Sawada, Paul A. Merolla, John V. Arthur, Rodrigo Alvarez-Icaza, Filipp Akopyan, Bryan L. Jackson and Dharmendra S. Modha, "TrueNorth: a High-Performance, Low-Power Neurosynaptic Processor for Multi-Sensory Perception, Action, and Cognition," 2016.
3. Paul A. Merolla, John V. Arthur, Rodrigo Alvarez-Icaza, Andrew S. Cassidy, Jun sawada, Filipp Akopyan *et al.*, "A Million Spiking-Neuron Integrated Circuit with a Scalable Communication Network and Interface," *science*, vol. 345, no.6197, pp. 668-673, 2014.
4. Mohammed F. Tolba, Abdulaziz H. Elsafty, Mina Armanyos, Lobna A. Said, Ahmed H. Madian, Ahmed G. Radwan, "Synchronization and FPGA realization of fractional-order Izhikevich neuron model," Microelectronics Journal, May 8, 2019.
5. Filipp Akopyan, Jun Sawada, Andrew Cassidy, Rodrigo Alvarez-Icaza, John Arthur, Paul Merolla *et al.*, "TrueNorth: Design and Tool Flow of a 65mW 1 Million NeuronProgrammable Neurosynaptic Chip," IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems, vol. 34, pp. 1537-1557, 2015.
6. Andrew S. Cassidy, Rodrigo Alvarez-Icaza, Filipp Akopyan, Jun Sawada, John V. Arthur, Paul A. Merolla, Pallab Datta, Marc Gonzalez Tallada, Brian Taba, Alexander Andreopoulos, Arnon Amir, Steven K. Esser, Jeff Kusnitz, Rathinakumar Appuswamy, Chuck Haymes, Bernard Brezzo, Roger Moussalli, Ralph Bellofatto, Christian Baks, Michael Mastro, Kai Schleupen, Charles E. Cox, Ken Inoue, Steve Millman, Nabil Imam, Emmett McQuinn, Yutaka Y. Nakamura, Ivan Vo, Chen Guo, Don Nguyen, Scott Lekuch, Sameh Asaad, Daniel Friedman, Bryan L. Jackson, Myron D. Flickner, William P. Risk, Rajit Manohar and Dharmendra S. Modha, "Real-time Scalable Cortical Computing at 46 Giga-Synaptic OPS/Watt with ~100 x Speedup in Time-to-Solution and ~100,000 x Reduction in Energy-to-Solution," in International Conference for High Performance Computing, Networking, Storage and Analysis (SC), 2014.
7. Arnon Amir, Pallab Datta, William P. Risk, Andrew S. Cassidy, Jeffrey A. Kusnitz, Steve K. Esser, Alexander Andreopoulos, Theodore M. Wong, Myron Flickner, Rodrigo Alvarez-Icaza, Emmett McQuinn, Ben Shaw, Norm Pass and Dharmendra S. Modha, "Cognitive Computing Programming Paradigm: A Corelet Language for Composing Networks of Neurosynaptic Cores," in International Joint Conference on Neural Networks (IJCNN), IEEE, 2013.
8. Steve K. Esser, Alexander Andreopoulous, Rathinakumar Appuswamy, Pallab Datta, Davis Barch, Arnon Amir, John Arthur, Andrew Cassidy, Myron Flickner, Paul Merolla, Shyamal Chandra, Nicola Basilico, Stefano Carpin, Tom Zimmerman, Frank Zee, Rodrigo Alvarez-Icaza, Jeffrey A. Kusnitz, Theodore M. Wong, William P. Risk, Emmett McQuinn, Tapan K. Nayak, Raghavendra Singh and Dharmendra S. Modha, "Cognitive Computing Systems: Algorithms and Applications for Networks of Neurosynaptic Cores," in International Joint Conference on Neural Networks (IJCNN), IEEE, 2013.
9. Eustace Painkras, Luis A. Plana, Jim Garside, Steve Temple, Francesco Galluppi, Cameron Patterson, David R. Lester, Andrew D. Brown and Steve B. Furber, "SpiNNaker: A 1-W 18-Core System-on-chip for Massively-Parallel Neural Network Simulation," IEEE Journal of Solid-State Circuits, vol.48, no. 8, August 2013.
10. Evangelos Stromatias, Francesco Galluppi, Cameron Patterson and Steve Furber, "Power analysis of large-scale, real-time neural networks on SpiNNaker," in International Joint Conference on Neural Networks (IJCNN), pp. 1-8, IEEE, 2013.
11. Steve B. Furber, David R. Lester, Luis A. Plana, Jim D. Garside, Eustace Painkras, Steve Temple and Andrew D. Brown, "Overview of the SpiNNaker System Architecture," IEEE Transactions on Computers, vol. 62, no. 12, December 2013.
12. Ben Varkey Benjamin, Peiran Gao, Emmett McQuinn, Swadesh Choudhary, Anand R. Chandrasekaran, Jean-Marie Bussat, Rodrigo Alvarez-Icaza *et al.* "Neurogrid: A Mixed-Analog-Digital Multichip System for Large-scale Neural simulations," Proceedings of the IEEE, vol. 102, no. 5, pp. 699-716, 2014.
13. Paul Merolla, John Arthur, Rodrigo Alvarez, Jean-Marie Bussat, Kwabena Boahen, "A Multicast Tree Router for Multichip Neuromorphic Systems," IEEE Transaction on Circuits and Systems, vol. 6, no. 1, pp. 820-833, March 2014.
14. Johannes Schemmel, Andreas Grubl, Stephan Hartmann, Alexander Kononov, Christian Mayr, Karlheinz Meier, Sebastian Millner, Johanes Partzsch, Stefan Schiefer, Stefan Scholze, Rene Schuffny and Marc-Olivier Schwartz, "Live Demonstration: A Scaled-Down Version of the BrainScaleS Wafer-Scale Neuromorphic System," in International Symposium on Circuits and Systems (ISCAS), pp. 702-702, IEEE, 2012.
15. Jens Sparso, Steve Furber, "Principles of Asynchronous Circuit Design," Kluwer Academic Publishers.
16. A. S. Cassidy et al., "Cognitive computing building block: a versatile and efficient digital neuron model for neurosynaptic cores," in International Joint Conference on Neural Networks (IJCNN), IEEE, 2013.

AUTHORS PROFILE

Zareen Sheikh received the B. E degree in Electronics and Telecommunication Engineering in the year 2017 and currently pursuing M. Tech in VLSI Design from Shri Ramdeobaba College of Engineering and Management.

Vivek Khetade received the B. E degree in Electronics Engineering from RTMNU Nagpur in 1997 and M. Tech in Electronics Design Technology from Center of Electronic Design Technology of India, CEDTI Aurangabad in 2004. He received P HD Degree in Electronics Engineering from RTMNU Nagpur with Department of Electronics, Shri Ramdeobaba College of Engineering and Management Nagpur as a research center. He is Assistant Professor in Electronics Design Technology, Shri Ramdeobaba College of Engineering and Management Nagpur. His research interest include neuromorphic computing for small application, asynchronous circuit and system design, reliable design of synchronizer for GALS.