

Analysis of single layer artificial neural network neuromorphic hardware chip

Aruna Pant, Adesh Kumar, Piyush Kuchhal

Department of Electrical and Electronics Engineering, School of Advanced Engineering, UPES, Dehradun, India

Article Info

Article history:

Received Jun 15, 2024

Revised Jul 30, 2024

Accepted Aug 12, 2024

Keywords:

Control logic

Field programmable gate array

Single layer artificial neural network

Neuromorphic hardware chip

System performance

ABSTRACT

The neuromorphic architectures are hardware network systems designed with neural functions. Neural networks seen in biology serve as an inspiration for network systems. A synapse connects every node or neuron in an artificial neural network (ANN) to every other node. As in biological brains, the amplitude of the linking between nodes referred to as synaptic weights will regulate the connection. In contrast to conventional design, ANN uses many highly organized dealing pieces that work together to solve real-world issues. The design of the neuromorphic hardware chip is discussed in the paper. The target device used is a Virtex-5 Field Programmable Gate Array (FPGA) and the simulation is taken on Xilinx ModelSim software. This chip is designed for 20 neuron inputs, each of the neuron inputs is 8-bit. Each 20-neuron input is multiplied by 20 input weights and each weight is 8-bit so when these 20 input weights are multiplied by 20 neuron inputs in the multiplier it gives 16-bit output. A control logic is used in this neuromorphic hardware chip design which is used to feed multiplier output to each input of the hidden layer. The system-level outcome of the hidden layer is then given to the multiplexer which has 20 inputs and one single output. The multiplexer is used to select any of the 20 outputs of the control logic. Finally, to gain an understanding of the performance of this neuromorphic hardware chip, we have computed the hardware utilization parameters. These parameters include slices, input/output blocks (IOBs), registers used, memory, and the overall propagation delay used by the hardware chip.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Adesh Kumar

Department of Electrical and Electronics Engineering, School of Advanced Engineering, UPES

Dehradun India

Email: adeshmanav@gmail.com

1. INTRODUCTION

Neuromorphic computing seeks to mimic the functionality of the human brain. It is predicated on the notion that the brain is a supercomputer with great efficiency and power that can process information in a non-linear fashion. To accomplish this, neuromorphic computing makes use of a network of interconnected neurons, which are basic processing units that process data. The electrical and chemical characteristics of biological neurons are mimicked in the design of the artificial neurons. Because of its ability to handle information in a manner close to that of the human brain, neuromorphic computing is a perfect fit for the improvement of brain-computer interfaces (BCIs). The capacity of neuromorphic computing to process information in a manner like the human brain is one of its primary advantages. This implies that it can be applied to develop BCIs that make sense to users more naturally. For instance, a neuromorphic BCI might make use of the same signals such as electroencephalography (EEG) signals that the brain uses to regulate

movement. The capacity of neuromorphic computing to process massive amounts of data quickly is another advantage. Additionally, by lowering the amount of power needed to operate BCIs, neuromorphic computing may enhance their overall performance. This may result in the creation of BCIs that are longer-lasting, lighter, and more portable. A device known as brain-computer interaction enables direct connection between the brain and external devices without relying on peripheral nerves. Nowadays, BCI is more of a science-invented topic than it was in the precedent because of significant breakthroughs in neurology, signal processing, computational neuroscience, sensors, and other related fields. BCI is an expanding field of study. Through the use of the EEG or other brain signals, people with severe mobility disabilities can communicate with external assistive devices through the use of BCI technology [1]. The interconnected microprocessing units are called neurons [2]. Four general functions of neurons are input, trigger, conduction, and output [3]. The electrical activity produced by brain structures and recorded from the surface of the scalp using electrodes is known as an electroencephalogram, or EEG. EEG is the primary method used by researchers to define brain activity since it can be recorded non-invasively with the use of disposable equipment. Through a full BCI system, the EEG or brain activity can be applied in real-time processing to control external equipment. A data collecting system, pre-processing of the obtained signals, feature extraction, feature classification, post-processing of the classifier output, and lastly the control user interface and device controller are the standard components of a BCI scheme.

In recent years, the field of BCI has witnessed significant breakthroughs in both the medical and information technology fields. The non-invasive version of BCIs, which is based on EEG, is particularly popular [4]. BCI is based on an EEG and presents new avenues of communication connecting the human brain and a computer [5]. Among the numerous different methods that are accessible, artificial neural networks (ANNs) are a well-established method that has several successful applications in the field of BCI research [6]. It is possible to apply neural networks using either analog or digital systems. Because of its benefits precision, more repeatability, reduced sensitivity to noise, improved testability, increased flexibility, and compatibility with various preprocessor types-the digital approach is more often used. The hardware implementations of digital neural networks are further divided into three categories: field programmable gate array (FPGA) based, digital signal processing (DSP) based, and application-specific integrated circuits (ASIC) based [7]. Since DSP implementation is sequential, the parallel architecture of the neurons in a layer is lost [8]. The ASIC applications do not allow for user reconfiguration. Because it allows for flexible reconfiguration and maintains the parallel architecture of the nerve cell neurons in a layer, FPGA is an appropriate piece of hardware for neural network implementation.

2. RELATED WORK

Three computational traits commonly linked with ANNs include parallelism, modularity, and dynamic adaptation. Because FPGA-based reconfigurable computing architectures allow for rapid reconfiguration and concurrency to adjust an ANN's weights and topology, they are an excellent choice for implementing ANNs. Large-scale neural network FPGA realization remains a difficult task due to the 'multiplication-rich' nature of ANN algorithms and their comparatively high implementation costs. Numerous studies have been published in this field, including multi-chip realization [9], NNs with certain limitations to obtain faster speed of operation at a cheaper cost [10], and new NN multiplication algorithms [11]. The assessment of EEG signals by incorporating the feed-forward multilayer ANN and FPGA utilizing VHDL language in the time-frequency representations for the domain analysis has been used to diagnose epilepsy, a neurological condition [12]. The current requirement for neuromorphic chips in real-world applications that require the best hardware and memory is what drove the development of ANN hardware chips. The architecture of the deep learning-based ANN is intended to deliver the best performance [13]. Architecture for using ANNs on FPGAs. The framework frees the user from handling low-level hardware details, allowing for the quick deployment of already trained ANNs on FPGA systems [14]. There is only one activation operational block incorporated in the new hardware architecture for feed-forward neural network (FFNN), that is external to the whole network. To minimize complexity and area usage. Intermediate results are easily accessible because the single activation block is not part of the network [15]. Additionally, only one hidden layer is used in this instance; all other layers ought to make use of the same hardware. The layer that is put into practice will be the biggest layer with the most neurons [16]. The architecture is often a systolic array. It is a uniform network of linked data processing units (DPU) known as cells or nodes [17]. The idea behind ANNs originated with brain concepts that were modified for use with digital computers. ANNs were first developed as mathematically modeled brain neurons [18]. These studies demonstrate that every neuron in ANNs receives information as input from either external or other neurons. This data is distributed as an output that is applied as a non-linear function and calculated as the weighted total of the inputs [19]. The best option for FFNN realization is hardware for FPGA implementations, but there are

numerous obstacles to overcome [20]. Activation function implementation, matrix multiplication implementation, weight matrix storage, partial results, coefficients, etc. are important for selecting an appropriate number format to optimize area, power, and speed [21]. Neural networks have many applications in very large-scale integration (VLSI) design, one of which is learning how to categorize brain behavior patterns. Furthermore, it is in situations where there is a strong correlation [22]. It is illustrated with VLSI networks of spike neurons and a multidimensional pattern of varying sacking rates [23]. The robust spike-driven flexibility approach is used to implement dynamic synapses [24]. Moreover, the input pattern to the output neuron, the input signal will offer an extra contribution spike train during training. It is employed to respond to the input prototype at a modest rate or with a high charge [25]. Silicon neurons were implemented based on the application needs of a collection of solutions. Different computational models can be implemented using the wide range of neuromorphic silicon circuits available [26]. The prediction of the overall number of tourists arriving in Spain was made using neural single-input single-output neural networks [27] and multiple-input multiple-output neural networks. An ANN receives a large number of signal inputs from the distinct data collection or output of previously associated neurons. Every input enters through a link known as a synapse, which has a mass [28]. A scalable ANN chip can be created with features like quick response times, low costs, low power consumption, and the ability to integrate with FPGA and work with embedded circuits. To implement the ANN, it is necessary to ensure that there are enough FPGA logic elements (LEs) or lookup tables (LUTs) available. This is crucial for handling the required number of neurons and their connections. Efficient utilization of registers or flip-flops is crucial for the storage of weights and intermediate values. It is required to calculate the necessary throughput to determine the clock frequency for your design. For your whole design, make sure that the setup and hold times are within the allowed range.

3. METHODS

Neuromorphic systems necessitate unique network topologies. The most common kind of network model implementation is the feed-forward neural network, which is employed in multilayer sensing. In addition to these, there are many other types of neural network topologies, such as artificial neural networks, cellular neural networks, recurrent neural networks, cellular automata, spiking neural networks, artificial neural networks, and neural networks that are pulse-coupled. In keeping with the current trend in neural network research, the hop-field network, a recurrent neural network (RNN) network design, was particularly popular in early neural morphology implementations. More modern implementations of this architecture exist. For example, data categorization, fault detection, graph partition, etc. work together to solve real-world issues. Artificial neural networks, or ANNs, are computer programs that draw stimulation from the human brain [29]. They can resolve extremely complicated issues that the traditional processing system is unable to handle. Because of its quick processing speed and accurate output, ANN is used in many different fields. There are already several varieties of ANNs. The most popular and straightforward one is feed-forward neural networks (FFNN). Neural networks and neuromorphic networks are the two ways to do ANN.

Neural networks are simply a software approach that runs on PC-based platforms. Due to its inability to manage the required throughput, a PC-based system is unfit for high-speed processing, prediction, decision-making, or categorization. Therefore, neuromorphic design or another hardware approach will be the way to go for ANN implementation [30]. Central processing units (CPUs) and graphics processing units (GPUs) are ideal for this task due to their lightning-fast data processing capabilities; nevertheless, they are relatively powerful when compared to FPGAs and other hardware platforms. Figure 1 presents the architecture of the single-layer feed-forward neural network. Figure 2 presents the functional blocks of the single-stage neuron network. We designed a system using VHDL modeling, which includes a single-stage neuron chip. We designed three primary units: the multiplication unit, the control unit, and a multiplexer [31]. The neuromorphic chip which had been designed can be divided mainly into three parts.

- Multiplier unit: The multiplier that was used to configure multiplies 20 neuron inputs with 20 neuron weights each having 8-bit information and 8-bit weights and gives one single output which is 16 bits. So, after the multiplication of neuron weight, we get the values, $M_0, M_1, M_2, M_3, M_4, M_5, M_6, M_7, \dots, M_{19}$.
- Control unit: In the control logic we have the inputs from $M_0, M_1, M_2, M_3, M_4, M_5, M_6, M_7, \dots, M_{19}$, one source address of 5 bits, and the outputs are $N_0, N_1, N_2, N_3, N_4, N_5, N_6, N_7, \dots, N_{19}$. The destination address is also 5 bits. Therefore, based on the source address and destination address the output of the multiplier is fed to each node of the hidden layer.
- Multiplexer unit: From the control unit we get 20 outputs which are from $N_0, N_1, N_2, N_3, N_4, \dots, N_{19}$. In the multiplexer unit, we have 20 inputs. Each input is 16 bits and one selection line which is 5 bits and based upon the selection line the output of the multiplexer unit we get.

To achieve parallelism, it is necessary to execute the computations of several neurons simultaneously, if the available FPGA resources allow for it. Pipelining improves the network's throughput by breaking down the processing into smaller parts. To provide proper management and synchronization of the input/output (I/O) pins, particularly when dealing with multiple clock domains, it is essential to ensure that all components of the design are synchronized with each other. To minimize power consumption, it is necessary to optimize the design by reducing switching activity and implementing low-power design solutions. Based on the simulation, the design is confirmed to have data rates for the ANN inputs and outputs that are consistent with the input/output bandwidth of the FPGA. The constraints of your FPGA board will determine the allocation of pins for data inputs, outputs, and control signals. The design is verified on the FPGA, utilizing in-circuit testing and debugging techniques, such as test benches and internal signal monitoring. The ANN is structured in a modular fashion, enabling easy expansion and modification to accommodate more intricate networks or diverse uses in the future. This functionality allows you to customize the ANN to suit different datasets and scenarios by modifying parameters such as the number of neurons or layers.

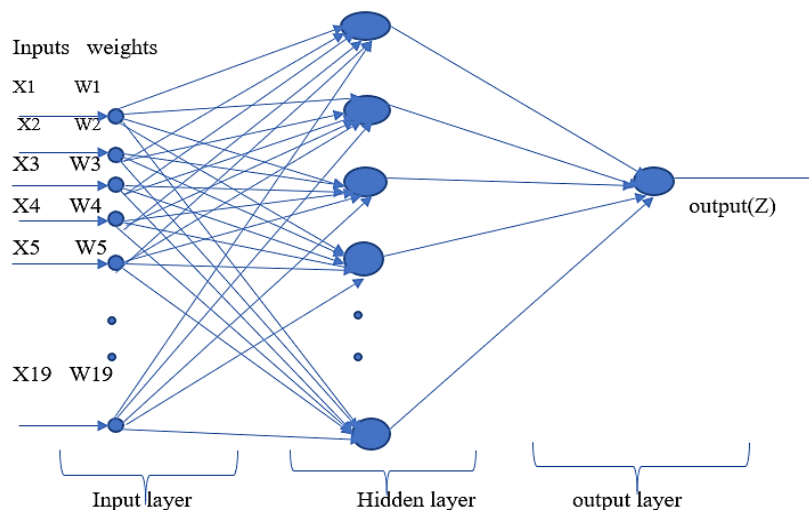


Figure 1. Architecture of single-layer feed-forward neural network

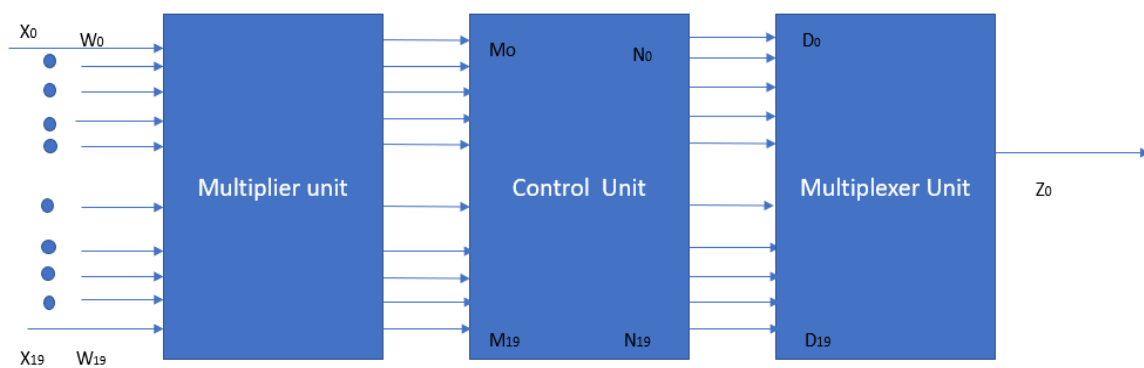


Figure 2. Functional blocks of single-stage neuron network

4. RESULTS AND DISCUSSION

The hardware neuromorphic chip for 20 neurons input is designed and each neuron is 8-bit input which is given first to the multiplier. The neuron input which we have given is from X_0 , X_1 , X_2 to X_{19} and the weights which we have used here are W_0 , W_1 , W_2 , W_3 to W_{19} . The weights with the input are first multiplied by the multiplier. It is important to optimize timing and data pathways, provide adequate synchronization, manage FPGA resources efficiently, consider power consumption and I/O limits, and design a single-layer ANN in HDL for FPGA.

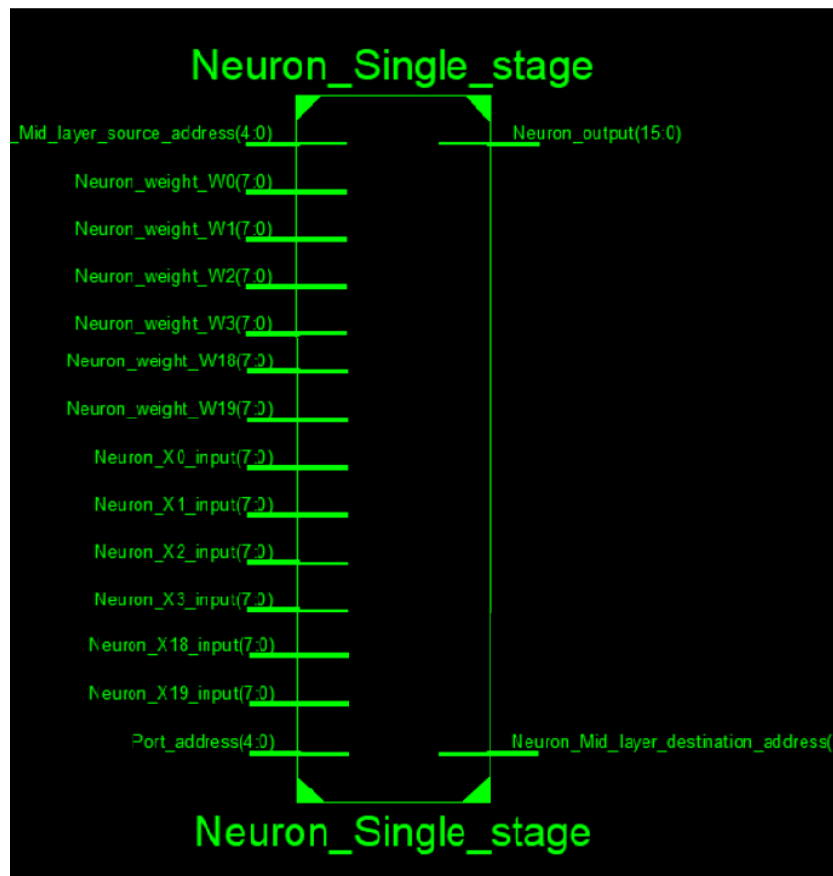


Figure 3. Single stage neuromorphic hardware chip register transistor logic (RTL)

The products we get are M_0, M_1, M_2 to M_{19} , which are the inputs of the control logic, and the outputs of the control logic are N_0, N_1, N_2 to N_{19} . The control logic is used to feed one neuron input to each of the inputs in the hidden layer. The output of the control logic is given to M_3 the multiplexer which mux all the inputs and gives a single output Z_0 . The multiplexer inputs are from D_0 to D_{19} , and it has one selection line which is of 5 bits, and one single output Z_0 which is of 16 bits. The inputs are processed as $\text{Neuron_X0_input}(7:0)$ to $\text{Neuron_X19_input}(7:0)$ presenting the 20 neurons inputs, the corresponding weights are $\text{Neuron_weight_w0}(7:0)$ to $\text{Neuron_weight_w19}(7:0)$. The port address (4:0) denotes layer address which is limited to single layer here. The $\text{neuron_output}(15:0)$ is the 16-bit output of the neural network. Figure 3 presents the RTL of a stage neuromorphic hardware chip. Figure 4 represents the schematic of single-stage neuromorphic hardware chips, respectively. The use of RTL in a VHDL programming language is crucial to developing a single-layer ANN that optimizes speed, resource consumption, and power efficiency, while also ensuring deterministic performance, scalability, and flexibility. To establish a scalable ANN, it is imperative to construct an architecture that can effectively handle high-dimensional, noisy, and temporal data. To create a versatile network for EEG applications, focus on incorporating modularity, optimizing memory use, implementing effective preprocessing techniques, and employing robust training methods. To enhance the system's performance and user-friendliness, it is advisable to consider the utilization of hardware acceleration, as well as the aspects of interpretability and real-world validation. ANNs in EEG applications can receive a diverse range of inputs, which can be complex and varied. These inputs include raw time-series signals, pre-processed and filtered data, and extracted features. To properly utilize EEG data in ANN, it is crucial to perform appropriate preprocessing, feature extraction, and data structure. To achieve the best performance and precise outcomes, it is advisable to adjust the input representation based on the specific problem and neural network structure.

Figures 5 and 6 represent the simulation waveform for single stage 20 inputs and outputs of ANN in binary using ModelSim. Figures 7 and 8 represent the simulation waveform for single stage 20 inputs and outputs of ANN in an integer using ModelSim. Tables 1 and 2 show the amount of hardware parameter used and its percentage used. The device utilization report for this neuromorphic hardware chip is taken from Xilinx software. The number of slices used is 320 in the neuromorphic hardware chip which is out of a total

of 12480. The number of LUTs used is 232 out of a total of 12480. Input-output blocks (IOB) used is 351 out of a total of 172. The total delay used in this neuromorphic hardware chip is 6.359 ns. The memory utilized for this neuromorphic hardware chip is 4665304 kilobytes. The target device used is Virtex-5 FPGA with the device (xc5v1x20t-2-ff323) for simulation and synthesis. The design utilized a power of 12.90 milliwatts and experienced a delay of 9.20 nanoseconds. Figure 9 depicts the neuromorphic hardware chip utilization chart.

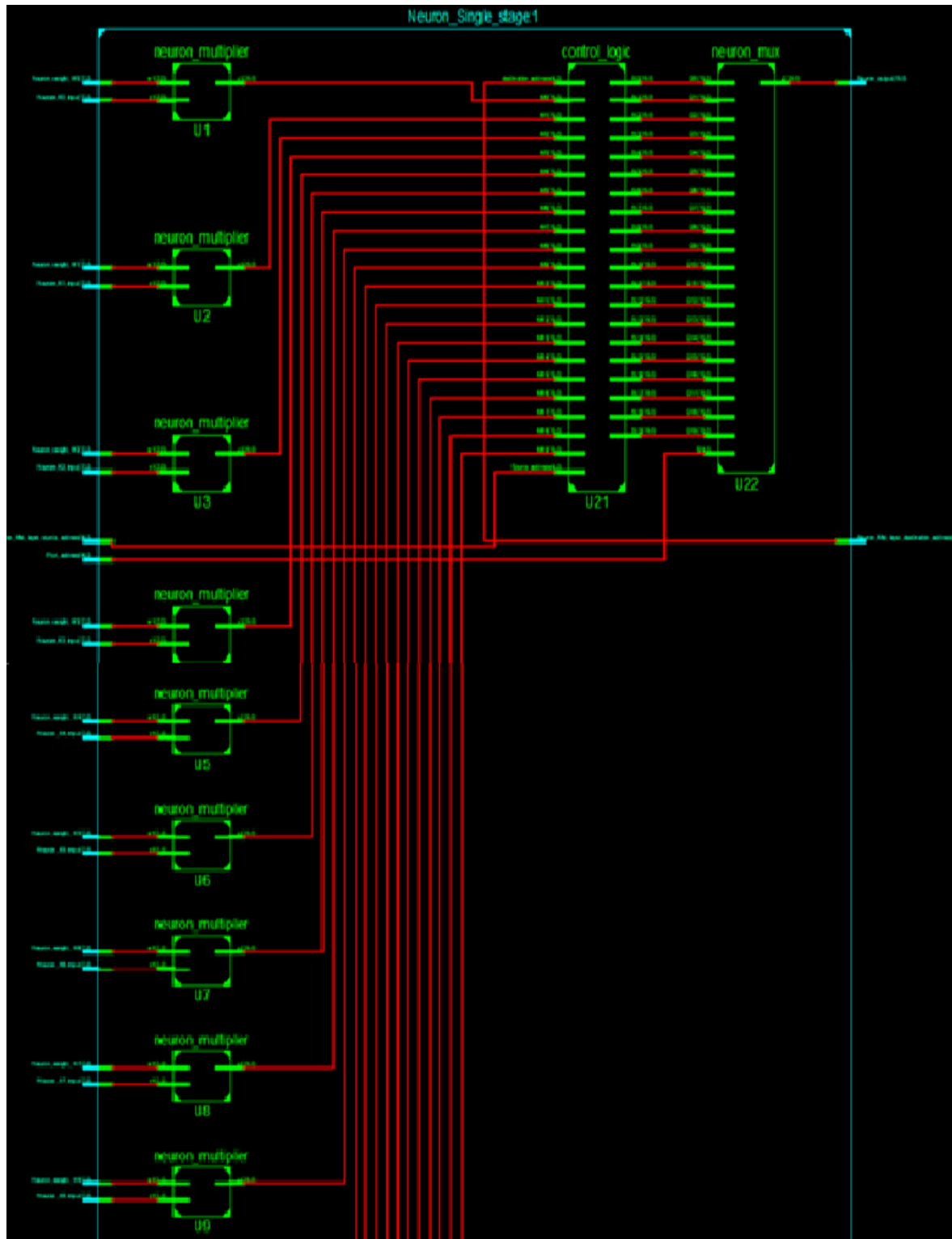


Figure 4. Schematic of single-stage neuromorphic hardware chip

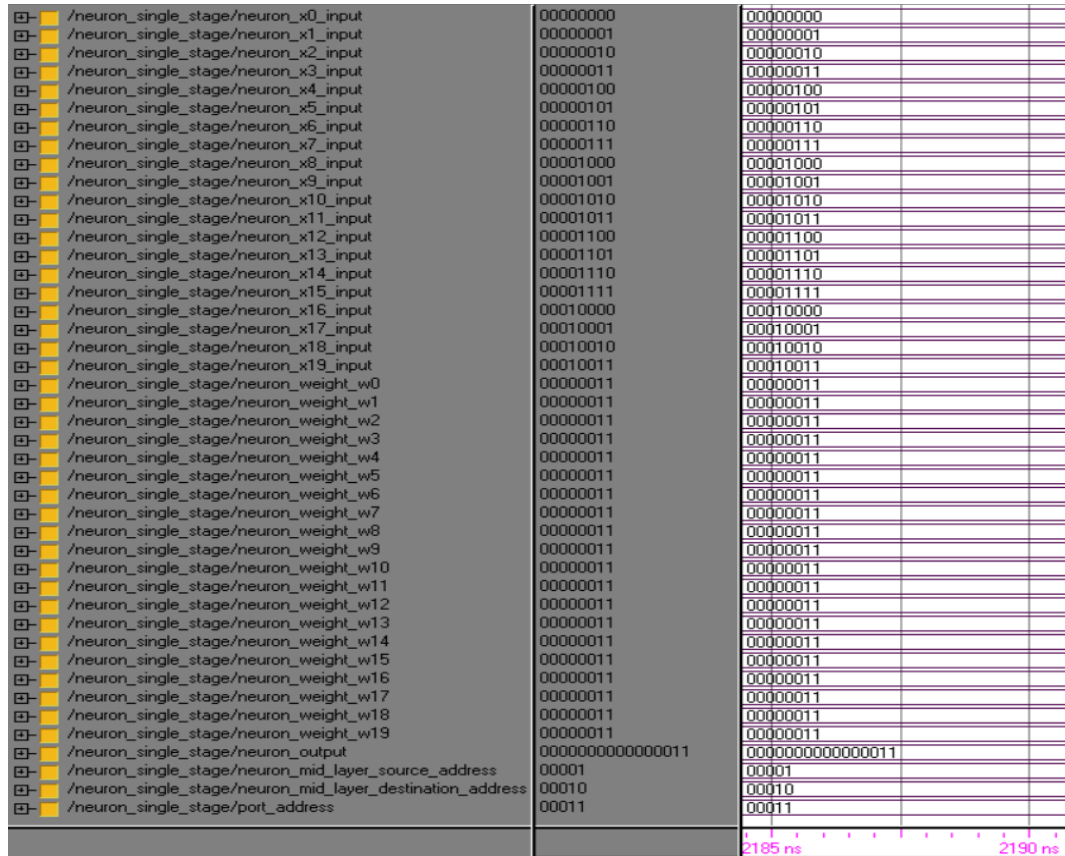


Figure 5. ModelSim simulation of single stage 20 inputs ANN in binary

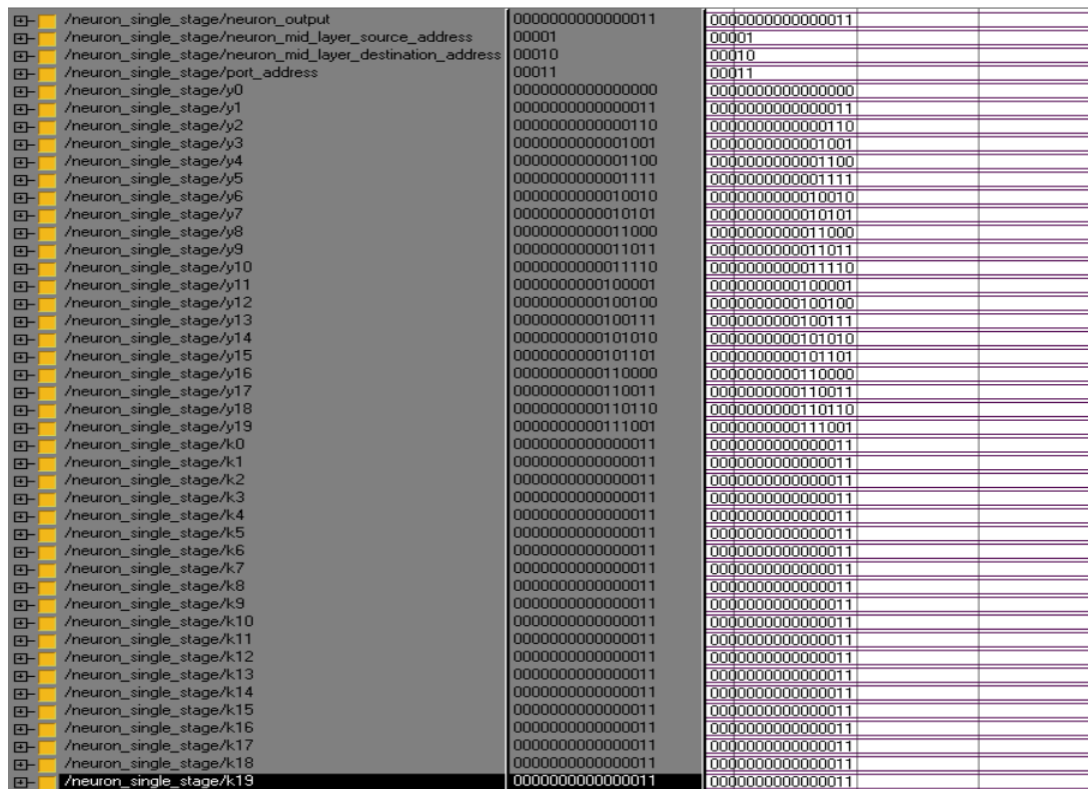


Figure 6. ModelSim simulation of the single-stage outputs of ANN in binary

| | | | | | |
|----|---|--------------------|--------------------|--|--|
| 01 | /neuron_single_stage/neuron_output | 0000000000000011 | 0000000000000011 | | |
| 02 | /neuron_single_stage/neuron_mid_layer_source_address | 00001 | 00001 | | |
| 03 | /neuron_single_stage/neuron_mid_layer_destination_address | 00010 | 00010 | | |
| 04 | /neuron_single_stage/port_address | 00011 | 00011 | | |
| 05 | /neuron_single_stage/y0 | 0000000000000000 | 0000000000000000 | | |
| 06 | /neuron_single_stage/y1 | 0000000000000011 | 0000000000000011 | | |
| 07 | /neuron_single_stage/y2 | 0000000000000110 | 0000000000000110 | | |
| 08 | /neuron_single_stage/y3 | 00000000000001001 | 00000000000001001 | | |
| 09 | /neuron_single_stage/y4 | 00000000000001100 | 00000000000001100 | | |
| 10 | /neuron_single_stage/y5 | 00000000000001111 | 00000000000001111 | | |
| 11 | /neuron_single_stage/y6 | 000000000000010010 | 000000000000010010 | | |
| 12 | /neuron_single_stage/y7 | 000000000000010101 | 000000000000010101 | | |
| 13 | /neuron_single_stage/y8 | 00000000000001000 | 00000000000001000 | | |
| 14 | /neuron_single_stage/y9 | 00000000000001011 | 00000000000001011 | | |
| 15 | /neuron_single_stage/y10 | 000000000000011110 | 000000000000011110 | | |
| 16 | /neuron_single_stage/y11 | 00000000000000001 | 00000000000000001 | | |
| 17 | /neuron_single_stage/y12 | 00000000000001000 | 00000000000001000 | | |
| 18 | /neuron_single_stage/y13 | 000000000000010111 | 000000000000010111 | | |
| 19 | /neuron_single_stage/y14 | 00000000000001010 | 00000000000001010 | | |
| 20 | /neuron_single_stage/y15 | 000000000000010101 | 000000000000010101 | | |
| 21 | /neuron_single_stage/y16 | 000000000000010000 | 000000000000010000 | | |
| 22 | /neuron_single_stage/y17 | 000000000000010011 | 000000000000010011 | | |
| 23 | /neuron_single_stage/y18 | 000000000000010110 | 000000000000010110 | | |
| 24 | /neuron_single_stage/y19 | 000000000000011001 | 000000000000011001 | | |
| 25 | /neuron_single_stage/k0 | 00000000000000011 | 00000000000000011 | | |
| 26 | /neuron_single_stage/k1 | 00000000000000011 | 00000000000000011 | | |
| 27 | /neuron_single_stage/k2 | 00000000000000011 | 00000000000000011 | | |
| 28 | /neuron_single_stage/k3 | 00000000000000011 | 00000000000000011 | | |
| 29 | /neuron_single_stage/k4 | 00000000000000011 | 00000000000000011 | | |
| 30 | /neuron_single_stage/k5 | 00000000000000011 | 00000000000000011 | | |
| 31 | /neuron_single_stage/k6 | 00000000000000011 | 00000000000000011 | | |
| 32 | /neuron_single_stage/k7 | 00000000000000011 | 00000000000000011 | | |
| 33 | /neuron_single_stage/k8 | 00000000000000011 | 00000000000000011 | | |
| 34 | /neuron_single_stage/k9 | 00000000000000011 | 00000000000000011 | | |
| 35 | /neuron_single_stage/k10 | 00000000000000011 | 00000000000000011 | | |
| 36 | /neuron_single_stage/k11 | 00000000000000011 | 00000000000000011 | | |
| 37 | /neuron_single_stage/k12 | 00000000000000011 | 00000000000000011 | | |
| 38 | /neuron_single_stage/k13 | 00000000000000011 | 00000000000000011 | | |
| 39 | /neuron_single_stage/k14 | 00000000000000011 | 00000000000000011 | | |
| 40 | /neuron_single_stage/k15 | 00000000000000011 | 00000000000000011 | | |
| 41 | /neuron_single_stage/k16 | 00000000000000011 | 00000000000000011 | | |
| 42 | /neuron_single_stage/k17 | 00000000000000011 | 00000000000000011 | | |
| 43 | /neuron_single_stage/k18 | 00000000000000011 | 00000000000000011 | | |
| 44 | /neuron_single_stage/k19 | 00000000000000011 | 00000000000000011 | | |

Figure 7. ModelSim simulation of single stage 20 input ANN in integer

| | | | | | |
|-------|---|-------------------|-------------------|--|--|
| 00-00 | /neuron_single_stage/neuron_output | 0000000000000011 | 0000000000000011 | | |
| 00-01 | /neuron_single_stage/neuron_mid_layer_source_address | 00001 | 00001 | | |
| 00-02 | /neuron_single_stage/neuron_mid_layer_destination_address | 00010 | 00010 | | |
| 00-03 | /neuron_single_stage/port_address | 00011 | 00011 | | |
| 00-04 | /neuron_single_stage/y0 | 0000000000000000 | 0000000000000000 | | |
| 00-05 | /neuron_single_stage/y1 | 0000000000000011 | 0000000000000011 | | |
| 00-06 | /neuron_single_stage/y2 | 0000000000000110 | 0000000000000110 | | |
| 00-07 | /neuron_single_stage/y3 | 00000000000001001 | 00000000000001001 | | |
| 00-08 | /neuron_single_stage/y4 | 00000000000001100 | 00000000000001100 | | |
| 00-09 | /neuron_single_stage/y5 | 0000000000000111 | 0000000000000111 | | |
| 00-0A | /neuron_single_stage/y6 | 00000000000010010 | 00000000000010010 | | |
| 00-0B | /neuron_single_stage/y7 | 00000000000010101 | 00000000000010101 | | |
| 00-0C | /neuron_single_stage/y8 | 00000000000011000 | 00000000000011000 | | |
| 00-0D | /neuron_single_stage/y9 | 00000000000011011 | 00000000000011011 | | |
| 00-0E | /neuron_single_stage/y10 | 00000000000011110 | 00000000000011110 | | |
| 00-0F | /neuron_single_stage/y11 | 00000000000100001 | 00000000000100001 | | |
| 00-10 | /neuron_single_stage/y12 | 00000000000100100 | 00000000000100100 | | |
| 00-11 | /neuron_single_stage/y13 | 00000000000100111 | 00000000000100111 | | |
| 00-12 | /neuron_single_stage/y14 | 00000000000101010 | 00000000000101010 | | |
| 00-13 | /neuron_single_stage/y15 | 00000000000101101 | 00000000000101101 | | |
| 00-14 | /neuron_single_stage/y16 | 00000000000110000 | 00000000000110000 | | |
| 00-15 | /neuron_single_stage/y17 | 00000000000110011 | 00000000000110011 | | |
| 00-16 | /neuron_single_stage/y18 | 00000000000110110 | 00000000000110110 | | |
| 00-17 | /neuron_single_stage/y19 | 00000000000111001 | 00000000000111001 | | |
| 00-18 | /neuron_single_stage/k0 | 00000000000000011 | 00000000000000011 | | |
| 00-19 | /neuron_single_stage/k1 | 00000000000000011 | 00000000000000011 | | |
| 00-1A | /neuron_single_stage/k2 | 00000000000000011 | 00000000000000011 | | |
| 00-1B | /neuron_single_stage/k3 | 00000000000000011 | 00000000000000011 | | |
| 00-1C | /neuron_single_stage/k4 | 00000000000000011 | 00000000000000011 | | |
| 00-1D | /neuron_single_stage/k5 | 00000000000000011 | 00000000000000011 | | |
| 00-1E | /neuron_single_stage/k6 | 00000000000000011 | 00000000000000011 | | |
| 00-1F | /neuron_single_stage/k7 | 00000000000000011 | 00000000000000011 | | |
| 00-20 | /neuron_single_stage/k8 | 00000000000000011 | 00000000000000011 | | |
| 00-21 | /neuron_single_stage/k9 | 00000000000000011 | 00000000000000011 | | |
| 00-22 | /neuron_single_stage/k10 | 00000000000000011 | 00000000000000011 | | |
| 00-23 | /neuron_single_stage/k11 | 00000000000000011 | 00000000000000011 | | |
| 00-24 | /neuron_single_stage/k12 | 00000000000000011 | 00000000000000011 | | |
| 00-25 | /neuron_single_stage/k13 | 00000000000000011 | 00000000000000011 | | |
| 00-26 | /neuron_single_stage/k14 | 00000000000000011 | 00000000000000011 | | |
| 00-27 | /neuron_single_stage/k15 | 00000000000000011 | 00000000000000011 | | |
| 00-28 | /neuron_single_stage/k16 | 00000000000000011 | 00000000000000011 | | |
| 00-29 | /neuron_single_stage/k17 | 00000000000000011 | 00000000000000011 | | |
| 00-2A | /neuron_single_stage/k18 | 00000000000000011 | 00000000000000011 | | |
| 00-2B | /neuron_single_stage/k19 | 00000000000000011 | 00000000000000011 | | |

Figure 8. ModelSim simulation of single-stage outputs of ANN in integer

Table 1. Hardware parameters utilized by neuromorphic hardware chip

| Neuromorphic Chip size | Multiplier used | Target device (FPGA) | LUTs | Slices | Delay (ns) | IOBs | Memory (KB) |
|------------------------|-----------------|------------------------------|------|--------|------------|------|-------------|
| ANN-20 | 20 | Virtex-5 (xc5v1x20t-2-ff323) | 232 | 320 | 6.359 | 351 | 4665304 |

Table 2. Hardware parameters utilized by neuromorphic hardware chip

| Hardware parameters | Parameter used | Available | Percentage (%) of parameters used |
|---------------------|----------------|-----------|-----------------------------------|
| Slices | 320 | 12480 | 2 |
| Registers | 232 | 12480 | 1 |
| IOBs | 351 | 172 | 204 |

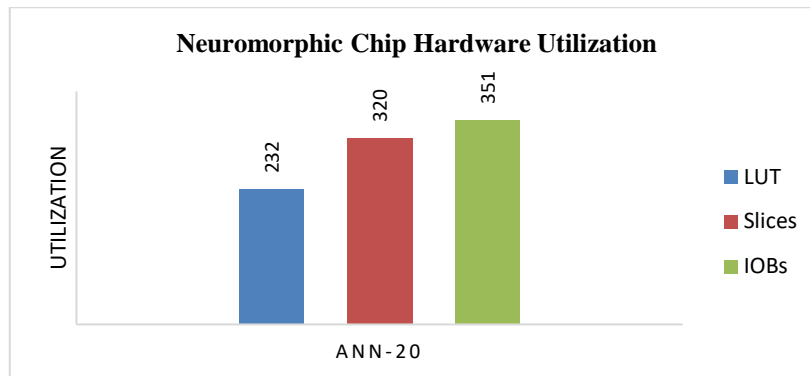


Figure 9. Neuromorphic hardware chip utilization chart

5. CONCLUSION

The architecture and operation of the human brain serve as the inspiration for the design of a chip known as a neuromorphic hardware chip. The neuromorphic hardware chip for 20-neuron input was designed successfully in Virtex-5 FPGA Xilinx ISE 14.7. The schematic of the single-stage neuromorphic chip is extracted successfully and the ModelSim simulation for different test inputs, respectively. The RTL of the single-stage neuromorphic chip is done using the detailed hardware parameters utilized by the neuromorphic hardware chip. The hardware utilization chart for the neuromorphic hardware chip includes the number of LUTs utilized in the neuromorphic hardware chip, which is 232, the Slices utilized are 320, and the IOBs used are 351. The total path delay is 6.359 ns, and the memory utilized in the design of the neuromorphic hardware chip is 46,65304 KB. Hardware parameters percentage utilized in the neuromorphic hardware chip design, number of LUT utilized in the neuromorphic hardware chip which is 232 and available are 12480 so percentage of parameter utilized is only 1%, and the Slices utilized are 320 and available are 12480, so percentage of parameter utilized is only 2%, and the input-output or IOBs used are 351 and available are 172 which is 204% of the whole.

ACKNOWLEDGEMENTS




Thanks to the UPES VLSI design and DSP Lab to provide the platform for carrying out the work.

REFERENCES




- [1] A. Goel, A. K. Goel, and A. Kumar, "Performance analysis of multiple input single layer neural network hardware chip," *Multimedia Tools and Applications*, vol. 82, no. 18, pp. 28213–28234, Jul. 2023, doi: 10.1007/s11042-023-14627-3.
- [2] J. Misra and I. Saha, "Artificial neural networks in hardware: A survey of two decades of progress," *Neurocomputing*, vol. 74, no. 1–3, pp. 239–255, Dec. 2010, doi: 10.1016/j.neucom.2010.03.021.
- [3] M. Marchesi, G. Orlandi, F. Piazza, and A. Uncini, "Fast Neural Networks Without Multipliers," *IEEE Transactions on Neural Networks*, vol. 4, no. 1, pp. 53–62, 1993, doi: 10.1109/72.182695.
- [4] A. Goel, A. K. Goel, and A. Kumar, "The role of artificial neural network and machine learning in utilizing spatial information," *Spatial Information Research*, vol. 31, no. 3, pp. 275–285, 2023, doi: 10.1007/s41324-022-00494-x.
- [5] A. S. Rawat, A. Rana, A. Kumar, and A. Bagwari, "Application of multi layer artificial neural network in the diagnosis system: a systematic review," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 7, no. 3, pp. 138–142, Aug. 2018, doi: 10.11591/ijai.v7.i3.pp138-142.
- [6] U. Pandey, A. Pathak, A. Kumar, and S. Mondal, "Applications of artificial intelligence in power system operation, control and planning: a review," *Clean Energy*, vol. 7, no. 6, pp. 1199–1218, Nov. 2023, doi: 10.1093/ce/zkad061.
- [7] A. Kumar, P. Chauda, and A. Devrari, "Machine Learning Approach for Brain Tumor Detection and Segmentation," *International*

- Journal of Organizational and Collective Intelligence*, vol. 11, no. 3, pp. 68–84, Jul. 2021, doi: 10.4018/ijoci.2021070105.
- [8] S. Dhyani, A. Kumar, and S. Choudhury, "Analysis of ECG-based arrhythmia detection system using machine learning," *MethodsX*, vol. 10, p. 102195, 2023, doi: 10.1016/j.mex.2023.102195.
 - [9] S. Dhyani, A. Kumar, and S. Choudhury, "Review of analysis of ECG based arrhythmia detection system using machine learning," 2023, p. 20029, doi: 10.1063/5.0178062.
 - [10] R. H. Turner and R. F. Woods, "Highly efficient, limited range multipliers for LUT-based FPGA architectures," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 10, pp. 1113–1117, Oct. 2004, doi: 10.1109/TVLSI.2004.833399.
 - [11] R. Sarić, D. Jokić, N. Beganović, L. G. Pokvić, and A. Badnjević, "FPGA-based real-time epileptic seizure classification using Artificial Neural Network," *Biomedical Signal Processing and Control*, vol. 62, p. 102106, Sep. 2020, doi: 10.1016/j.bspc.2020.102106.
 - [12] V. S. Dhaka *et al.*, "A survey of deep convolutional neural networks applied for prediction of plant leaf diseases," *Sensors*, vol. 21, no. 14, p. 4749, Jul. 2021, doi: 10.3390/s21144749.
 - [13] P. Škoda, T. Lipić, A. Srp, B. M. Rogina, K. Skala, and F. Vajda, "Implementation framework for Artificial Neural Networks on FPGA," in *MIPRO 2011 - 34th International Convention on Information and Communication Technology, Electronics and Microelectronics - Proceedings*, 2011, pp. 274–278.
 - [14] J. Schmidhuber, "Deep Learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, Jan. 2015, doi: 10.1016/j.neunet.2014.09.003.
 - [15] J. Kim, K. Hwang, and W. Sung, "X1000 real-time phoneme recognition VLSI using feed-forward deep neural networks," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 7510–7514, doi: 10.1109/ICASSP.2014.6855060.
 - [16] L. D. Medus, T. Iakymchuk, J. V. Frances-Villora, M. Bataller-Mompean, and A. Rosado-Munoz, "A Novel Systolic Parallel Hardware Architecture for the FPGA Acceleration of Feedforward Neural Networks," *IEEE Access*, vol. 7, pp. 76084–76103, 2019, doi: 10.1109/ACCESS.2019.2920885.
 - [17] T. V. Huynh, "Deep neural network accelerator based on FPGA," in *2017 4th NAFOSTED Conference on Information and Computer Science, NICS 2017 - Proceedings*, Nov. 2017, vol. 2017-January, pp. 254–257, doi: 10.1109/NAFOSTED.2017.8108073.
 - [18] J. Park and W. Sung, "FPGA based implementation of deep neural networks using on-chip memory only," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2016, pp. 1011–1015, doi: 10.1109/ICASSP.2016.7471828.
 - [19] K. Guo, S. Zeng, J. Yu, Y. Wang, and H. Yang, "[DL] A survey of FPGA-based neural network inference accelerators," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 12, no. 1, pp. 1–26, Mar. 2019, doi: 10.1145/3289185.
 - [20] T. Choudhary, V. Mishra, A. Goswami, and J. Sarangapani, "A comprehensive survey on model compression and acceleration," *Artificial Intelligence Review*, vol. 53, no. 7, pp. 5113–5155, Feb. 2020, doi: 10.1007/s10462-020-09816-7.
 - [21] A. P. D. A. Ferreira and E. N. D. S. Barros, "A high performance full pipelined architecture of MLP neural networks in FPGA," in *2010 IEEE International Conference on Electronics, Circuits, and Systems, ICECS 2010 - Proceedings*, Dec. 2010, pp. 742–745, doi: 10.1109/ICECS.2010.5724619.
 - [22] I. Giacomo *et al.*, "Neuromorphic silicon neuron circuits," *Frontiers in Neuroscience*, vol. 5, 2011, [Online]. Available: http://www.frontiersin.org/Journal/Abstract.aspx?s=755&name=neuromorphic_engineering&ART_DOI=10.3389/fnins.2011.00073.
 - [23] M. K. Singh, N. Singh, and A. K. Singh, "Speaker's Voice Characteristics and Similarity Measurement using Euclidean Distances," in *2019 International Conference on Signal Processing and Communication, ICSC 2019*, Mar. 2019, pp. 317–322, doi: 10.1109/ICSC45622.2019.8938366.
 - [24] F. Corradi, D. Bontrager, and G. Indiveri, "Toward neuromorphic intelligent brain-machine interfaces: An event-based neural recording and processing system," in *IEEE 2014 Biomedical Circuits and Systems Conference, BioCAS 2014 - Proceedings*, Oct. 2014, pp. 584–587, doi: 10.1109/BioCAS.2014.6981793.
 - [25] T. Posewsky and D. Ziener, "Throughput optimizations for FPGA-based deep neural network inference," *Microprocessors and Microsystems*, vol. 60, pp. 151–161, Jul. 2018, doi: 10.1016/j.micpro.2018.04.004.
 - [26] A. Kumar, G. Verma, M. K. Gupta, M. Salauddin, B. K. Rehman, and D. Kumar, "3D multilayer mesh NoC communication and FPGA synthesis," *Wireless Personal Communications*, vol. 106, no. 4, pp. 1855–1873, Jun. 2019, doi: 10.1007/s11277-018-5724-3.
 - [27] S. Anwar, K. Hwang, and W. Sung, "Fixed point optimization of deep convolutional neural networks for object recognition," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2015, pp. 1131–1135, doi: 10.1109/ICASSP.2015.7178146.
 - [28] P. D. Deotale and L. Dole, "Design of FPGA based general purpose neural network," Feb. 2015, doi: 10.1109/ICICES.2014.7033843.
 - [29] S. Himavathi, D. Anitha, and A. Muthuramalingam, "Feedforward neural network implementation in FPGA using layer multiplexing for effective resource utilization," *IEEE Transactions on Neural Networks*, vol. 18, no. 3, pp. 880–888, May 2007, doi: 10.1109/TNN.2007.891626.
 - [30] Y. Zhang and S. Zhang, "Optimized deep learning architectures with fast matrix operation kernels on parallel platform," in *2013 IEEE 25th International Conference on Tools with Artificial Intelligence*, Nov. 2013, pp. 71–78, doi: 10.1109/ICTAI.2013.21.
 - [31] Z. Yu, A. M. Abdulghani, A. Zahid, H. Heidari, M. A. Imran, and Q. H. Abbasi, "An overview of neuromorphic computing for artificial intelligence enabled hardware-based hopfield neural network," *IEEE Access*, vol. 8, pp. 67085–67099, 2020, doi: 10.1109/ACCESS.2020.2985839.
 - [29] S. Himavathi, D. Anitha and A. Muthuramalingam, "Feedforward neural network implementation in FPGA using layer multiplexing for effective resource utilization," in *IEEE Transactions on Neural Networks*, vol. 18, no. 3, pp. 880–888, May 2007, doi: 10.1109/TNN.2007.891626.
 - [30] Y. Zhang and S. Zhang, "Optimized deep learning architectures with fast matrix operation kernels on parallel platform," in *2013 IEEE 25th International Conference on Tools with Artificial Intelligence*, Herndon, VA, USA, 2013, pp. 71–78, doi: 10.1109/ICTAI.2013.21.
 - [31] Z. Yu, A. M. Abdulghani, A. Zahid, H. Heidari, M. A. Imran and Q. H. Abbasi, "An overview of neuromorphic computing for artificial intelligence enabled hardware-based hopfield neural network," in *IEEE Access*, vol. 8, pp. 67085–67099, 2020, doi: 10.1109/ACCESS.2020.2985839.




BIOGRAPHIES OF AUTHORS

Aruna Pant    is working as a junior research fellow (JRF) in the Department of Electrical and Electronics Engineering, UPES, Dehradun, India. She has an M.Tech. in Electronics and Communication Engineering from Amity University, Lucknow, India. Her area of interest is biomedical signal processing, EEG signals, and digital signal processing. She has 12 years experience of teaching. She can be contacted at arunapant41@gmail.com.



Adesh Kumar    has been working as a professor in the Department of Electrical and Electronics, Engineering, in the Department of Electrical and Electronics Engineering, School of Advanced Engineering, The University of Petroleum and Energy Studies (UPES), Dehradun, India since 2010. He has a B.Tech. in Electronics and Communication Engineering from UPTU, Lucknow India in 2006. M.Tech. (Hons) in Embedded Systems Technology, from SRM University, Chennai in 2008. Ph.D. (Electronics Engineering) from UPES, Dehradun India in 2014. He has also worked as a senior engineer in TATA ELXSI LIMITED Bangalore and as a faculty member in ICFAI University, Dehradun. His areas of interest are VLSI design, embedded systems design, telecommunications, and signal processing. He has supervised 8 Ph.D. scholars and 5 candidates are doing research under his supervision. He has worked on more than 10 editorial assignments for edited books, conference proceedings, and journals, having citations of more than 2000, h-index of 25 and i10-index of 45. He has 16 years of experience in the teaching+ research industry, published more than 120 research papers in international peer-reviewed journals (SCI/Scopus) and conferences. He can be contacted at adeshmanav@gmail.com or adeshkumar@ddn.upes.ac.in.



Piyush Kuchhal    has extensive experience in education, research, and administration. With more than 22 years of experience, he has flourished in a wide range of positions and demonstrated exceptional expertise in his field. He holds a Ph.D. in Physics from the IIT Roorkee and an M.Sc. in Physics with a concentration in Engineering Physics from the same institution. His exceptional academic achievements throughout his education prove his commitment to excellence. Prof. Piyush has held significant positions of leadership within academic institutions. He oversaw operations, curriculum design, and academic quality as Cluster-Head of Electrical Engineering at UPES. In addition, he has held the positions of Academic Coordinator, Associate Dean of Applied Sciences, and Department Head of Physics. Moreover, he is a committed and accomplished professional with expertise in teaching, research, and administrative administration. His extensive knowledge, academic accomplishments, and substantial contributions make him an asset in the academic and research communities. He can be contacted at pkuchhal@ddn.upes.ac.in.