Multimedia Tools and Applications (2023) 82:28213–28234
https://doi.org/10.1007/s11042-023-14627-3

Performance analysis of multiple input single layer neural network hardware chip

Akash Goel[1] · Amit Kumar Goel[1] · Adesh Kumar[2]

Received: 12 November 2021 / Revised: 24 February 2022 / Accepted: 3 February 2023 /
Published online: 20 February 2023

Abstract
An artificial neural network (ANN) is a computational system that is designed to replicate and process the behavior of the human brain using neuron nodes. ANNs are made up of thousands of processing neurons with input and output modules that self-learn and compute data to offer the best results. The hardware realization of the massive neuron system is a difficult task. The research article emphasizes the design and realization of multiple input perceptron chips in Xilinx integrated system environment (ISE) 14.7 software. The proposed single-layer ANN architecture is scalable and accepts variable 64 inputs. The design is distributed in eight parallel blocks of ANN in which one block consists of eight neurons. The performance of the chip is analyzed based on the hardware utilization, memory, combinational delay, and different processing elements with targeted hardware Virtex-5 field-programmable gate array (FPGA). The chip simulation is performed in Modelsim 10.0 software. Artificial intelligence has a wide range of applications, and cutting-edge computing technology has a vast market. Hardware processors that are fast, affordable, and suited for ANN applications and accelerators are being developed by the industries. The novelty of the work is that it provides a parallel and scalable design platform on FPGA for fast switching, which is the current need in the forthcoming neuromorphic hardware.

Keywords ANN architecture · Single layer ANN · Virtex-5 FPGA

1 Introduction

Neurons are the cells in the nervous system that carry information to the other cells in the nerve and communicate with each other in distinctive ways. The neurons [10] are the elementary

✉ Akash Goel
a.goel54@yahoo.com

1 Department of Computer Science & Engineering, Galgotia's University, Greater Noida, NCR, India

2 Department of Electrical & Electronics Engineering, University of Petroleum and Energy Studies, Dehradun, India

functioning unit in the brain. The nerve cell or the neurons communicate [47] with each other using a dedicated connection called synapses. The neurons are categorized into three types based on their functionality, which are sensory neurons, motor neurons, and interneurons. The sensory neurons [1] send the signals to the brain or the spinal cord. The sensory neurons are responsible for the response of the different stimuli of the human such as sound, light, or touch which is affected by the sensory organs by the cells. The motor neurons get the signals to form the brain and spinal cord [2] to derive the output based on muscle contractions to glandular output. The interneurons are connecting multiple neurons with the brain region or the spinal cord. The connections of these neurons form a circuit called a neural circuit. The neurons comprise a cell body called soma, dendrites, and an axon [21]. The soma is typically compact. The dendrites and axons are the filaments that extrude from them. The dendrites can extend freely from the soma maybe a hundred micrometers. The axon hillock is the swelling point at which the axon leaves soma, which can go for 1 m in human beings or larger in other species. The axon terminals pass [57] the signals to synapses and the other cells in the body. It may be that the neurons do not have axons or dendrites in the case of the undifferentiated cells. Typically, neurons are having a cell body, dendrites, and an axon. The cell body comprises the cytoplasm and the nucleus. The axon prolongs from the cell body and regularly provides growth to minor outlets or branches before termination at nerve points. Dendrites cover the neuron cell body and accept the signals from other neurons. The main contact points are synapses responsible for the communication among neurons, which may connect one dendrite to another dendrite, one axon to another axon. The dendrites [50] are covered with synapses formed by the ends of axons from other neurons. In general nature, the neurons are electrically excitable and maintain the voltage gradients within their membranes. Therefore, the signaling mechanism is electrical and partly chemical.

The general-purpose hardware is based on the arithmetic blocks for simple in-memory calculations. Serial processing does not provide fast and sufficient performance for deep learning applications. The ANN architectures are based on parallel computation and operations. Ordinary chips cannot support a large number of highly and simultaneous operations for neuron processing. The AI-based hardware chip includes different chips that enable parallel processing. The main motivation for using the ANN and AI-based hardware accelerators is to get higher bandwidth memory chips and faster computation in comparison to general-purpose hardware.

Digital tools and simulators are appropriate and applied for discovering the measurable behavior of neural networks. Silicon neuron systems [7] are a mix of analog and digital signals that may be used to analyze behavior using VLSI integrated circuits, and simulate electrophysiological behavior for actual neuron processing at various levels of abstraction. The most recent FPGAs can handle a huge number of physical memory and logic gates [22], allowing large-scale neural networks to be implemented on hardware and at a reasonable cost. The current level of simulation and synthesis technology is that research laboratories can easily afford FPGAs. The hardware synthesis method allows researchers to work on parallel brain cell structures. Digital models will be used for cell-based controls, and digital stem coding techniques will be used to facilitate communication across the medium across vast distances. Subsequently, it is well known that the neurons can be used to module ANN of the earlier generations by equating mean firing rates of processing neurons and hardware for proficient, scalable, and low-power implementations [6] of single-layer feed-forward networks.

Human brain activity can be observed in both the local and delocal domains. The activities are linked to several functions such as vision and hearing, which are linked to specific brain regions. When a brain injury or accident occurs, the behavior of the brain neurons changes. The brain is a miniature network environment in which each portion has its own set of neural connections that are segregated from one another and confections. The local response is merged into a global understanding that causes the entire brain activity to become distressed. Machine learning and ANN-based intelligent methods have been proposed in the medical and health care industry to enhance security and train the models to improve patient treatment, diagnostics, rights, prevention, autonomy, and equality [55]. The research was offered based on deep learning-based Mobile Net V2 and long short-term memory (LSTM) to automate the process of identifying and classification [27] skin diseases. Oversampling techniques [32] can be used to determine cervical cancer based on feature extraction and spatial clustering. The synthetic minority over-sampling is used for hypertension, disease identification [31], and predictions based on the random forest machine learning method. The Wrapper filter [41] was used for disease classification and features selection. Neural networks have been applied for the CT images of the human liver for accurate diagnosis [56] of the disease related to the liver.

ANNs have several advantages that make them ideal for solving specific scenarios and difficulties. ANN systems can learn and model non-linear functions as well as construct complicated associations, which is critical for real-world solutions and associates between non-linear and complex function inputs and outputs. The sense inputs and outputs cause the neural networks to alter or learn. ANN is a term that refers to several deep learning technologies that fall under the umbrella of artificial intelligence [18]. These technologies are mostly used in commercial applications to handle pattern recognition and sophisticated signal processing difficulties. For addressing nonlinear excitation functions, the development and realization of a single neural network require computing logic such as adders, multipliers, and a complex function evaluator [40]. The precision of the computational blocks is the most significant quality in the digital implementation of a single neural network [45]. It is acquired by determining their word length, which aids in the selection of a higher resolution. The fulfillment of the function necessitates appropriate mathematical matching, as the better resolution may result in higher system costs. As a result, implementing a single neural network in hardware will necessitate the multiplier, addition, and excitation function realization blocks [49]. The testing of the advanced neural networks and machine learning algorithms will require an advanced level of FPGA and simulation tools. The FPGA provides the platform in which high performance can be achieved using data processing blocks. The most powerful and mature neuro-chips are digital neural ASICs. High computational precision, great dependability, and high programmability are all advantages of digital technology. Furthermore, advanced design tools for digital full and semi-custom design are accessible. The weights of synaptic connections can be stored on or off the chip. The trade-off between speed and size determines this decision.

The organization of the article is as follows: section 2 presents the related work, section 3 presents the structure of the single-layer neural network, and section 4 presents the design of the logarithmic multi-neuron system. The results & discussions are presented in section 5, followed by conclusions in section 5.

2 Related work

Neural networks (ANN) have been used widely for developments in a broad spectrum of perception, classification, association, control, and biomedical applications. The ANN hardware implementation was done on FPGA in the digital domain to miniaturization of component manufacturing technology. A high-speed ANN architecture was implemented on the Xilinx FPGA chip for random number generators and further used for data encryption over the network. The perceptron model of the multi-bit [4] input neuron was implemented in 130 nm technology. The model was proposed for the low power consumption baes on 4 neurons per layer. The multilayer perceptron architectures are for complex decision regions [33] and activation functions play an important role. The three-layer model first is the input layer, the second layer is the perceptron or hidden layer, and the third output layer. A feedforward network was used for the selection of concrete beams [30]. The metaheuristics approach [44] was used to realize the feed-forward ANN. The ANN engine was implemented on FPGA based on the parallel processing [14] the blocks and hardware parameters were analyzed. The backpropagation multilayer perceptron (MLP) was proposed to design based on a very large scale of integration (VLSI) parameters and FPGA [13] using a very high speed integrated circuit (VHSIC) - hardware description language (VHDL) to amylase the chip performance. The Spiking neural network (SNN) [23] was designed for targeting 64 K neurons on FPGA for hardware accelerator. The performance of the neural network was enhanced using the concept of parallelization [15] applied in both the time and space domains. The design was having 3/2, 7/3, 15/4, and 31/5 inputs/outputs. The design was implemented Altera EP3C16F484- Cyclon III FPGA on Quartus II software using VHDL.

In general, hardware systems for deep neural network (DNN) inference [52] suffer from a lack of on-chip memory, compelling access to additional memory-only processors. It was recommended to employ nonvolatile scalable memory that can scale up to a 64-chip illusion system. The hardware neural network models have been used for dataflow [60] and weight access patterns of neurons, in which recurrent neural networks (RNNs) and probabilistic graphical models are used for compute-in-memory (CIM) designs that can be implemented using CMOS technology. The neural network design was implemented on FPGA [58], and the performance may be measured using hardware metrics like memory, chip area, and size. Such hardware can be utilized to create hardware embedded chips and internet of things (IoT) applications. Deep learning approaches [11] have been successfully employed to handle a variety of artificial intelligence challenges. The FPGA has been utilized to optimize various reconfigurable computer hardware and software for AI designs. The topologic and hardware designs are based on multiple neuron processing and scalable computation. The neural network architecture can be implemented using a processing engine layout [34] for the hardware performance analysis framework for recognizing bottlenecks in the initial stages of a convolutional neural network (CNN). This methodology is useful for evaluating various architectures for embedded chips and associated applications like hardware accelerators. The ANN was modeled for various logic functions and logic gates [26]. One of the gates utilized for serval applications and quick modeling is the XOR gate. The 3-input XOR gate hardware was modeled using ANN to anticipate intelligent learning and numerical methods to improve forecast accuracy. A novel way was presented for accelerating fully linked feed-forward neural networks [48] using an FPGA-based accelerator. The program was created to make diverse implementation activities easier by dividing the architecture into elementary layers, estimating the available computational hardware resources, and generating high-level C++ descriptions using high-level synthesis (HLS) tools. The decision tree classifier and neural networks [38] have been used for the hardware in loop testing in the power window. The machine models were used to estimate the 93% accuracy of the system in automotive power window hardware. The neural networks have been used for the diagnosis of different diseases and their realization in hardware are helpful for the implementation of optimal hardware. The diagnosis of epilepsy neurological disorder [54] has been done using the analysis of the electroencephalography (EEG) signals by embedding the feed-forward multi-layer neural network architecture (MLP ANN) and FPGA using VHDL in the time-frequency domain. The multiple input neural networks [29] have been used for forecasting death cases in China due to COVID-19. The models were applicable to do the study and estimation of COVID-19 cases across the globe. The neural network inference [61] is limited because of encounters between the high computation and storage complexity and resource-restricted hardware requirements in different applications. The current study trends are developing in the direction of neural network research that is complicated in the acceleration of FPGA-based stages. The architecture of neural networks can be designed and synthesized on FPGA to estimate the hardware chip applications, and optical solutions for computing parallelism, data reuse, computing complexity, pruning, and quantization.

The reconfigurable computing architectures [59] play a very important role in real-time applications. The neural network was implemented in FPGA based on reconfigurable computing. The FPGA implementation has many challenges such as less hardware, memory utilization, minimum delay and timing parameters, and low power consumption. The VHDL programming was used to design the hardware chip of the design and on the Xilinx XC V50hq240 FPGA chip, Zynq FPGA [46] was used to test the behavior of the neural network chip and throughput optimization. The neural single-input single-output [51] and Multiple-input multiple-output neural networks were used for forecasting the total number of tourist arrivals in Spain. ANN acquires many inputs from the unique data set or output of erstwhile correlated neurons. Each input approaches through a connection, which is called synapses and which has weight [16]. The scalable ANN chip can be designed that can provide fast response, low price, less power consumption, and switch to operate with embedded chips and integration on FPGA.

The ANNs are used in a variety of applications including brain activity, modeling, and artificial intelligence. When employing HDL language and FPGA-based system retaliation [3], the number of neurons in an ANN design is limited. The ANN obtains a large number of inputs from a single data set or the output of previously connected neurons. The inputs are advanced through a link called synapses, which has a weight attached to it. The realization of the system may be using multilevel communication networks, convolution neural functions, single layer architecture, and other neural networks. The scalable ANN chip can be used to give fast response, cheap cost, and low power consumption, as well as the ability to work with embedded circuits [9] and FPGA integration. The neural systems and switching operations are followed by cluster-based models, in which a large number of units are deployed throughout a specific network to provide original and supplemental services, which can improve communication. The specialized processors use standardized software, response behavior, essential data control, and service module for coordination [28]. For logarithmic inputs, ANN modeling can be done in terms of the power of two. For large-scale network structures in which multiuser support the network's functionality, such as 2-input, 4-input, 8-input, 16-input, 32-input, 64-input, etc. The chip design and FPGA-based system integration and implementation will offer scalable computing hardware [39] and the platform in which we can extend the user and computational hardware as the communication system is needed. There is a research gap in the design and development of the chip that supports the multi-neuron-clustering environment in which multiple users are communicating in intra-exchange and interexchange environments [35].

The motivation for ANN hardware chip design is the current need for neuromorphic chips in real-world applications that need optimal hardware and memory. The deep learning-based ANN architecture is designed to provide optimal performance [17]. The hierarchical astrocyte network (HANA) design [12] is based on the hierarchical networks-on-chip (NoC) structure by providing a unique of neurons and astrocytes cells that support information exchanges between astrocyte cells and addresses the connectivity difficulty. The design was based on scalable computing. By establishing a modular array of clusters of neurons employing a hierarchical structure of low and high-level routers using 65 nm CMOS technology, the unique hierarchical NoC architecture was employed to overcome the scalability issue [63]. An embedded system-based chip [62] was designed using a cross paradigm neuromorphic chip, to simplify the structure of different neural networks spike or non-spike forms. Neuro-inspired computing chips [43] are a promising approach to the development of intelligent computing because they mimic the structure and operating principles of the biological brain. These neuro-inspired computing chips are superior to traditional systems. It is predicted to provide benefits in terms of hardware memory, energy efficiency, and computational power. The objective of the research work is to design and model of single-layer neural network chip for multiple scalable neuron inputs and estimate the hardware chip performance in terms of memory, delay, and FPGA resources.

3 Structure of single-layer neural network

In a general way, the model of the neural network [5] is depicted in Fig. 1. The model accepts the ‘n’ number of neuron inputs. Let us consider that the inputs are X_1, X_2, X_3……X_n. These inputs are processed with their corresponding weights as W_1, W_2, W_3…...W_n and ‘b’ is the bias input. The nonlinear execution function is f(x). The neuron processing is expressed with the help of the Eq. (1).

$$y = f(x) \tag{1}$$

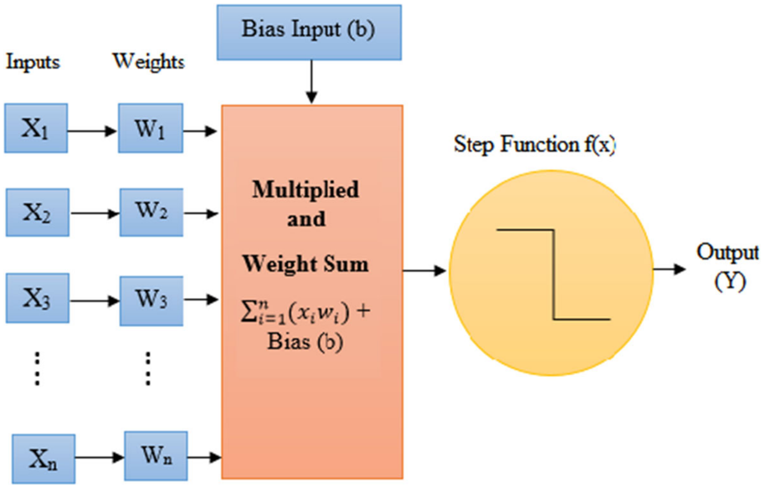


Fig. 1 ANN Structure [25]

$$x = \sum_{i=1}^{n} x_i w_i + Bias\ (b) \tag{2}$$

The W_i is mentioned as the weights for the i^th connections and b is the bias inputs. The behavior of the function f(x) is a nonlinear excitation function. The most popular excitation function used is expressed as.

For linear function,

$$f(x) = x \tag{3}$$

For log sigmoid function,

$$f(x) = \frac{1}{1 + e^{-x}} \tag{4}$$

For tan sigmoid function,

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{5}$$

Figure 2 presents the ANN structure of 8 neuron inputs with their weight coefficients, the Eq. (1) is expressed as

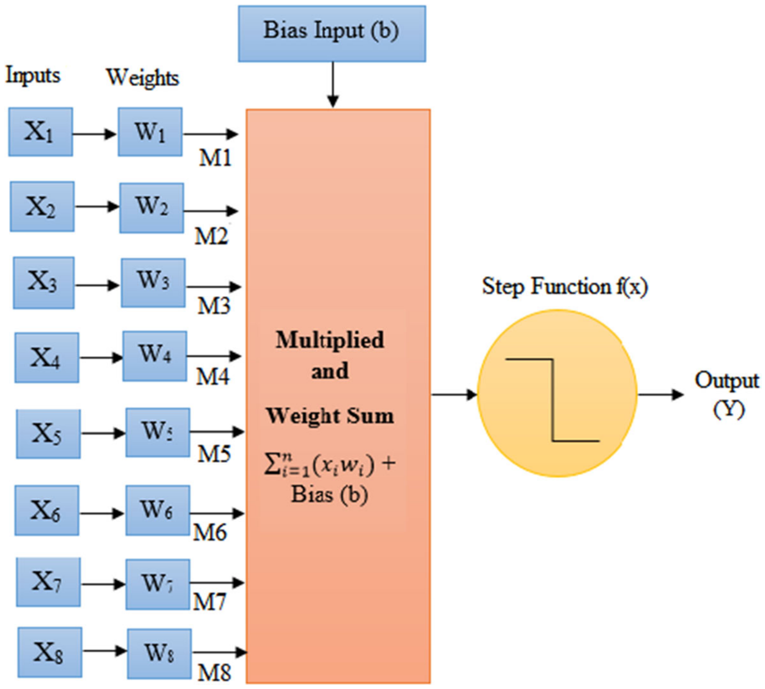


Fig. 2 ANN with 8 inputs and weights

$$y = \sum_{i=1}^{8} X_i W_i + (b) \tag{6}$$

The output is expressed as

$$y = X_1W_1 + X_2W_2 + X_3W_3 + X_4W_4 + X_5W_5 + X_6W_6 + X_7W_7 + X_8W_8 + (b) \tag{7}$$

The hardware realization of the network needs 8 multipliers and 8 adders as shown in Table 1. The multipliers are presented as M_1, $M_2 \ldots M_8$.

$$y = M_1 + M_2 + M_3 + M_4 + M_5 + M_6 + M_7 + M_8 + (b) \tag{8}$$

4 Design of Logarithmic Multi Neuron System

The scalable design of the logarithmic single layer multi-neuron system [8, 53] is shown in Fig. 3. The design has 64 neurons inputs X_1, X_2, X_3, X_4 …….X_{64} with corresponding input weights as W_1, W_2, W_3, $W_{4............}$ W_{64}. The functionality of the 64 inputs ANN can be understood with the help of parallel working of 8 blocks of 8-point ANN. The individual block accepts the 8 neuron points with their weights. The parallel execution of all the modules provides faster operation. The suggested operation is expended in terms of logarithmic execution in terms of the power of 2. It is a scalable architecture that can be progressed in the power of 2. The operation of the 64 inputs ANN can be understood with the help of Table 2. The weighted sum is obtained with the processing of 64-point ANN with a bias to provide final outputs. The scalable architecture is assigned with the module address "000", "001", "010", "011", "100", "101", "110", and "111" against the sequential processing of 8-point ANN architecture. The design is scalable can be extended to a larger extent and solve the ANN problems at a large scale.

The finite state machine (FSM) concept is used to create the ANN architecture. The state memory is used to save the current state of the machine, which requires 'N' flip-flops. A single clock signal is used to synchronize all of the flip-flops. The state vector is used to hold the state memory in the state machine as depicted in Fig. 4. The state machine processes state-0, state-1, state-2, state-3, state-4, state-5, state-6, and state-7 using the address inputs "000", "001", "010", "011", "100", "101", "110", and "111". One hot encoding approach is one in which one state is realized depending on its selection input and one output is derived all at once. The neurons X_1 to X_8 are multiplied with weights W_1 to W_8 and added with bias input-1 to produce

Table 1 Multipliers and adders

Multiplier-1 (X_1 x W_1)=M_1 Multiplier-2 (X_2 x W_2)=M_2 Multiplier-3 (X_3 x W_3)=M_3 Multiplier-4 (X_4 x W_4)=M_4 Multiplier-5 (X_5 x W_5)=M_5 Multiplier-6 (X_6 x W_6)=M_6 Multiplier-7 (X_7 x W_7)=M_7 Multiplier-8 (X_8 x W_8)=M_8	Adder (M_1+M_2+M_3+M_4+M_5+M_6 +M_7+M_8)+Bias	Step Function f(x)	Output (Y)

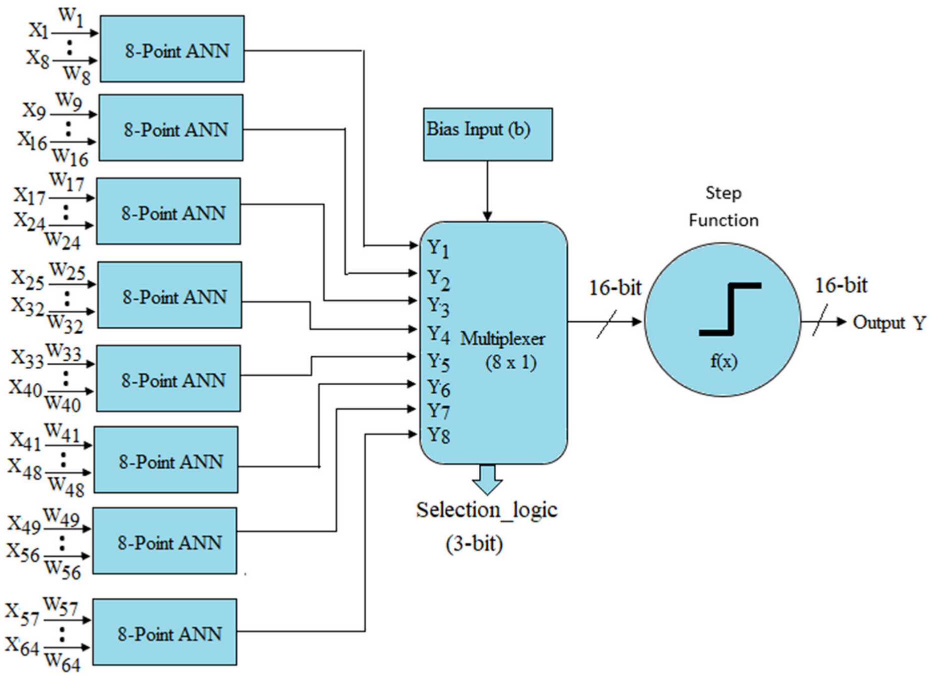


Fig. 3 Multiple input ANN (64-point) architecture

the neuron output Y_1 in state-0 (000). The neurons X_9 to X_{16} are multiplied with weights W_9 to W_{16} and coupled with bias input-2 to produce the neuron output Y_2 in state-1 (001). The neurons X_{17} to X_{24} are multiplied with weights W_{17} to W_{24} and added with bias input-3 in state-2 (010) to produce the neuron output Y_3. The neurons X_{25} to X_{32} are multiplied with weights W_{25} to W_{32} and added with bias input-4 in state-3(011) to produce the neuron output Y_4. The neurons X_{33} to X_{40} are multiplied with weights W_{33} to W_{40} and added with bias input-5 in state-4 (100), yielding the neuron output Y_5. The neurons X_{41} to X_{48} are multiplied with weights W_{41} to W_{48} and added with bias input-6 to produce the neuron output Y_6 in state-5 (101). The neurons X_{49} to X_{56} are multiplied with weights W_{49} to W_{56} and added with bias input-7 to produce the neuron output Y_7 in state-6 (110). The neurons X_{57} to X_{64} are multiplied with weights W_{57} to W_{64} and added with bias input-8 to produce the neuron output Y_8 in state-7 (111).

Table 2 Realization of 64-point ANN

Selection_logic	Execution
000	8-Point ANN $(X_1W_1+X_2W_2+X_3W_3+X_4W_4+X_5W_5+X_6W_6+X_7W_7+S_8W_8)$+Bias
001	8-Point ANN $(X_9W_9+X_{10}W_{10}+X_{11}W_{11}+X_{12}W_{12}+X_{13}W_{13}+X_{14}W_{14}+X_{15}W_{15}+X_{16}W_{16})$+Bias
010	8-Point ANN $(X_{17}W_{17}+X_{18}W_{18}+X_{19}W_{19}+X_{20}W_{20}+X_{21}W_{21}+X_{22}W_{22}+X_{23}W_{23}+X_{24}W_{24})$+Bias
011	8-Point ANN $(X_{25}W_{25}+X_{26}W_{26}+X_{27}W_{27}+X_{28}W_{28}+X_{29}W_{29}+X_{30}W_{30}+X_{31}W_{31}+X_{32}W_{32})$+Bias
100	8-Point ANN $(X_{33}W_{33}+X_{34}W_{34}+X_{35}W_{35}+X_{36}W_{36}+X_{37}W_{37}+X_{38}W_{38}+X_{39}W_{39}+X_{40}W_{40})$+Bias
101	8-Point ANN $(X_{41}W_{41}+X_{42}W_{42}+X_{43}W_{43}+X_{44}W_{44}+X_{45}W_{45}+X_{46}W_{46}+X_{47}W_{47}+X_{48}W_{48})$+Bias
110	8-Point ANN $(X_{49}W_{49}+X_{50}W_{50}+X_{51}W_{51}+X_{52}W_{52}+X_{53}W_{53}+X_{54}W_{54}+X_{55}W_{55}+X_{56}W_{56})$+Bias
111	8-Point ANN $(X_{57}W_{57}+X_{58}W_{58}+X_{59}W_{59}+X_{60}W_{60}+X_{61}W_{61}+X_{62}W_{62}+X_{63}W_{63}+X_{64}W_{64})$+Bias

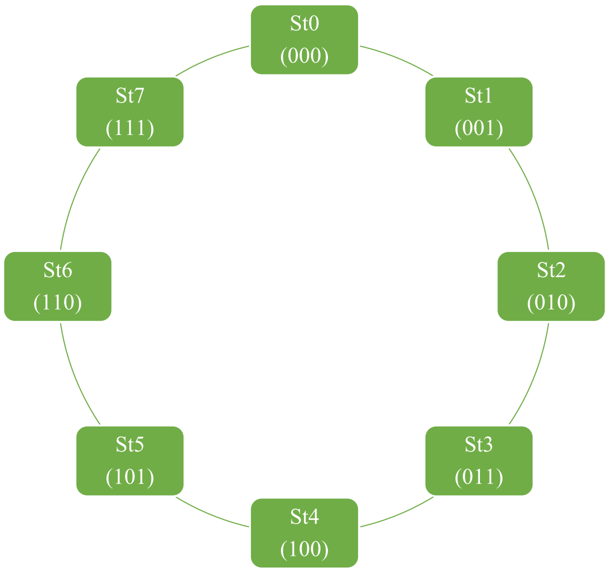


Fig. 4 FSM for 64 input ANN Processing

5 Results and discussions

The hardware chip of the 8-point ANN and 64-point ANN is designed using VHDL coding in Xilinx ISE 14.7. Figure 5 presents the register transfer level (RTL) block diagram for the 8-point to 64-point ANN chip. The RTL depicts all inputs and outputs of the designed chip.

$X_1 < 7:0 >$ to $X_{64} < 7:0 >$ presents the inputs (8-bit) of 64 neuron inputs ANN architecture with std_logic_vector data type. $W_1 < 7:0 >$ to $W_{64} < 7:0 >$ presents the weight inputs (8-bit) corresponding to neuron inputs X_1 to X_{64} of std_logic_vector data type. B_i < 15:0 > is the bias input treated as the perceptron of the ANN architecture of 16-bit with std_logic_vector data type. X_A < 15:0 > It is the activation function output ANN architecture with the 16-bit size of std_logic_vector data type. Y < 15:0 > It is the actual output with weighted sum and bias input, processed with an activation function of 16-bit of std_logic_vector data type.

Modelsim simulation of 8 input ANN in binary and integer is shown in Figs. 6 and 7 respectively. Table 3 lists the test cases used for the functional simulation of the designed ANN chip. Modelsim simulation of 64 input ANN in binary and integer is shown in Figs. 8 and 9. Table 4 lists the test cases used for the functional simulation of the designed ANN-64 with test case-1 to test case-8.

The percentage of hardware that is used by the device is given by the device utilization report [37] for the implementation of the chip. The report is taken directly from the Xilinx software as the device utilization report. The report presents the number of adders, multipliers, slices, 4 input lookup tables (LUT) [36], input/output blocks (IOB), total memory usage (kB), combinational delay (ns) that includes path delay and routing delay. The Xilinx device summary for ANN-8, ANN-16, ANN-24, ANN-32, ANN-40 ANN-48, ANN-66, and ANN-64 is given in Table 5. The target device is Virtex-5 FPGA with device xc5vlx20t-2-ff323 used for simulation and synthesis [24]. Figure 10 presents the hardware utilization curve for ANN-8 to ANN-64 hardware chips.

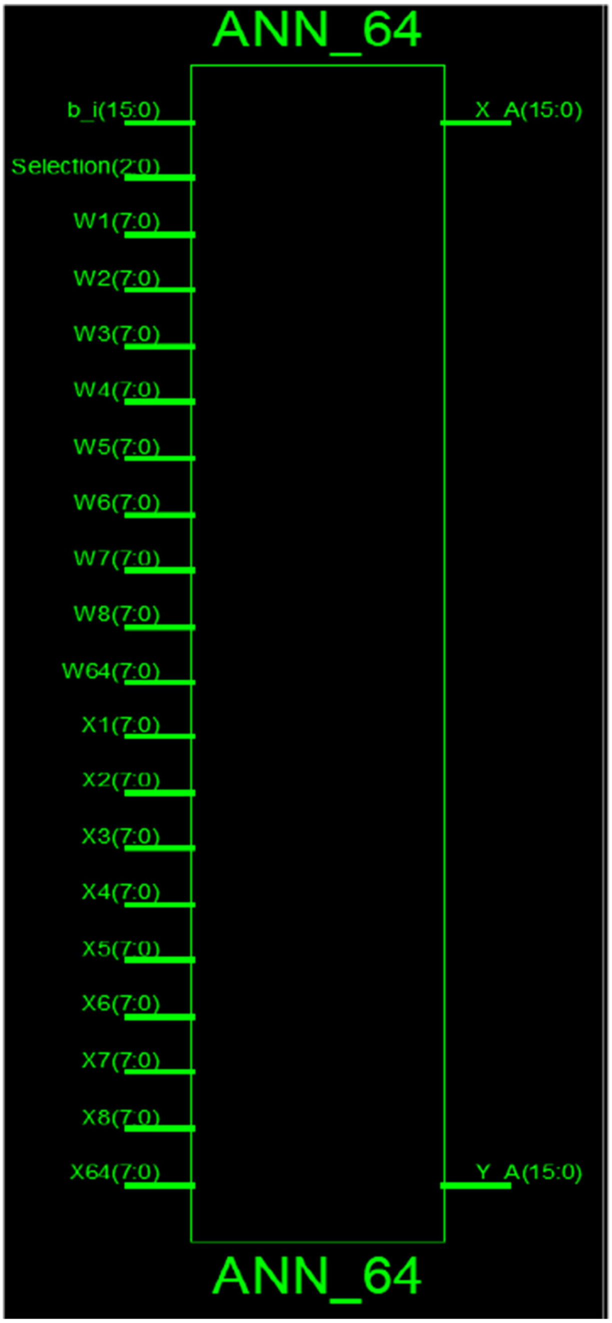
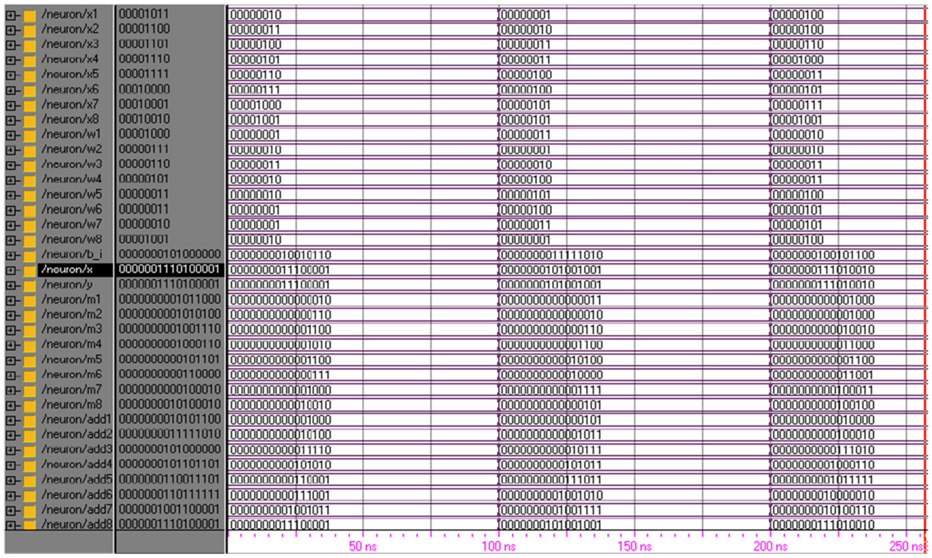


Fig. 5 RTL of ANN

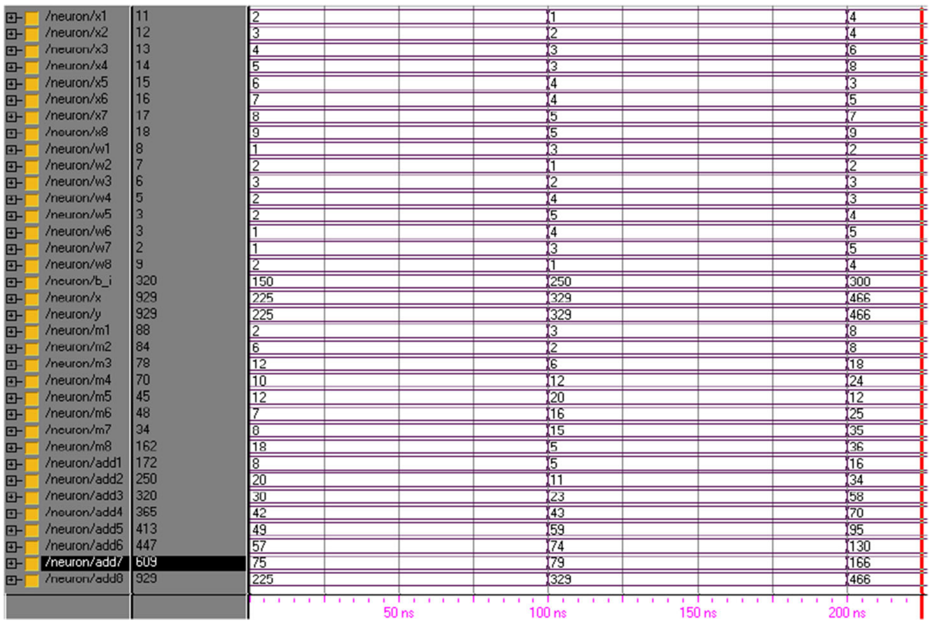
In the simulation of ANN-64, the hardware and memory usage depends on the utilizations of multipliers and adders. The detail of these units is reported directly by the software and change with the number of neurons and weight inputs. The hardware utilization will increase



```
/neuron/x1    00001011          00000010          00000001          00000100
/neuron/x2    00001100          00000011          00000010          00000100
/neuron/x3    00001101          00000100          00000011          00000110
/neuron/x4    00001110          00000101          00000011          00001000
/neuron/x5    00001111          00000110          00000100          00000011
/neuron/x6    00010000          00000111          00000100          00000101
/neuron/x7    00010001          00001000          00000101          00000111
/neuron/x8    00010010          00001001          00000101          00001001
/neuron/w1    00001000          00000001          00000011          00000010
/neuron/w2    00000111          00000010          00000001          00000010
/neuron/w3    00000110          00000011          00000010          00000011
/neuron/w4    00000101          00000010          00000100          00000011
/neuron/w5    00000011          00000010          00000101          00000100
/neuron/w6    00000011          00000001          00000100          00000101
/neuron/w7    00000010          00000001          00000011          00000101
/neuron/w8    00001001          00000010          00000001          00000100
/neuron/b_i   0000000101000000  0000000010010110  0000000011111010  0000000100101100
/neuron/x     0000001110100001  0000000011100001  0000000101001001  0000000111010010
/neuron/y     0000001110100001  0000000011100001  0000000101001001  0000000111010010
/neuron/m1    0000000001011000  0000000000000010  0000000000000011  0000000000001000
/neuron/m2    0000000001010100  0000000000000110  0000000000000010  0000000000001000
/neuron/m3    0000000001001110  0000000000001100  0000000000000110  0000000000010010
/neuron/m4    0000000001000110  0000000000001010  0000000000001100  0000000000011000
/neuron/m5    0000000000101101  0000000000001100  0000000000010100  0000000000001100
/neuron/m6    0000000000110000  0000000000000111  0000000000010000  0000000000011001
/neuron/m7    0000000000100010  0000000000001000  0000000000001111  0000000000100011
/neuron/m8    0000000010100010  0000000000010010  0000000000000101  0000000000100100
/neuron/add1  0000000010101100  0000000000001000  0000000000000101  0000000000010000
/neuron/add2  0000000011111010  0000000000010100  0000000000001011  0000000000100010
/neuron/add3  0000000101000000  0000000000011110  0000000000010111  0000000000111010
/neuron/add4  0000000101101101  0000000000101010  0000000000101011  0000000001000110
/neuron/add5  0000000110011101  0000000000110001  0000000000111011  0000000001011111
/neuron/add6  0000000110111111  0000000000111001  0000000001001010  0000000010000010
/neuron/add7  0000001001100001  0000000001001011  0000000001001111  0000000010100110
/neuron/add8  0000001110100001  0000000011100001  0000000101001001  0000000111010010
                                        50 ns        100 ns        150 ns        200 ns        250 ns
```

Fig. 6 Modelsim simulation of 8 input ANN in binary

with the increase in cluster inputs of the ANN chip. The simulation results show that the number of multipliers, adders, slices, LUTs, memory is increasing as the number of neurons are increasing in the multi-input ANN design. The reason for this is that the adders and multipliers blocks increase the number of gates and concurrent logic modules, which takes up more memory and resources on the FPGA.



```
/neuron/x1    11    2     1     4
/neuron/x2    12    3     2     4
/neuron/x3    13    4     3     6
/neuron/x4    14    5     3     8
/neuron/x5    15    6     4     3
/neuron/x6    16    7     4     5
/neuron/x7    17    8     5     7
/neuron/x8    18    9     5     9
/neuron/w1    8     1     3     2
/neuron/w2    7     2     1     2
/neuron/w3    6     3     2     3
/neuron/w4    5     2     4     3
/neuron/w5    3     2     5     4
/neuron/w6    3     1     4     5
/neuron/w7    2     1     3     5
/neuron/w8    9     2     1     4
/neuron/b_i   320   150   250   300
/neuron/x     929   225   329   466
/neuron/y     929   225   329   466
/neuron/m1    88    2     3     8
/neuron/m2    84    6     2     8
/neuron/m3    78    12    6     18
/neuron/m4    70    10    12    24
/neuron/m5    45    12    20    12
/neuron/m6    48    7     16    25
/neuron/m7    34    8     15    35
/neuron/m8    162   18    5     36
/neuron/add1  172   8     5     16
/neuron/add2  250   20    11    34
/neuron/add3  320   30    23    58
/neuron/add4  365   42    43    70
/neuron/add5  413   49    59    95
/neuron/add6  447   57    74    130
/neuron/add7  609   75    79    166
/neuron/add8  929   225   329   466
                          50 ns        100 ns        150 ns        200 ns
```

Fig. 7 Modelsim simulation of 8 input ANN in integer

Table 3 Test cases for the simulation waveform

Pins	Detail	Test case-1		Test case-2		Test case-3	
		Integer	Binary	Integer	Binary	Integer	Binary
X1<7:0>	Input	2	00000010	1	00000001	4	00000100
X2<7:0>	Input	3	00000011	2	00000010	4	00000100
X3<7:0>	Input	4	00000100	3	00000011	6	00000110
X4<7:0>	Input	5	00000101	3	00000011	8	00001000
X5<7:0>	Input	6	00000110	4	00000100	3	00000011
X6<7:0>	Input	7	00000111	4	00000100	5	00000100
X7<7:0>	Input	8	00001000	5	00000101	7	00000111
X8<7:0>	Input	9	00001001	5	00000101	9	00001001
W1<7:0>	Input	1	00000001	3	00000011	2	00000010
W2<7:0>	Input	2	00000010	1	00000001	2	00000010
W3<7:0>	Input	3	00000011	2	00000010	3	00000011
W4<7:0>	Input	2	00000010	4	00000100	3	00000011
W5<7:0>	Input	2	00000010	5	00000101	4	00000100
W6<7:0>	Input	1	00000001	4	00000100	5	00000101
W7<7:0>	Input	1	00000001	3	00000011	5	00000101
W8<7:0>	Input	2	00000010	1	00000001	4	00000100
B_i<15:0>	Input	150	0000000010010110	250	0000000011111010	300	0000000100101100
Y<15:0>	output	225	0000000011100001	329	0000000101001001	466	0000000111010010

The report predicts that the number of multipliers and 16-bit adders are increasing with the number of neurons inputs. The predicting of mean squared error (MSE), mean absolute percentage error (MAPE), root mean squared error (RMSE) is done for the FPGA hardware resources [25, 42] based on the training and validation sample neurons with different cluster inputs of ANN design. In the training (X_1 to X_{40}) are considered and (X_{41} to X_{64}) for validation. The values are determined using the equations [19, 20].

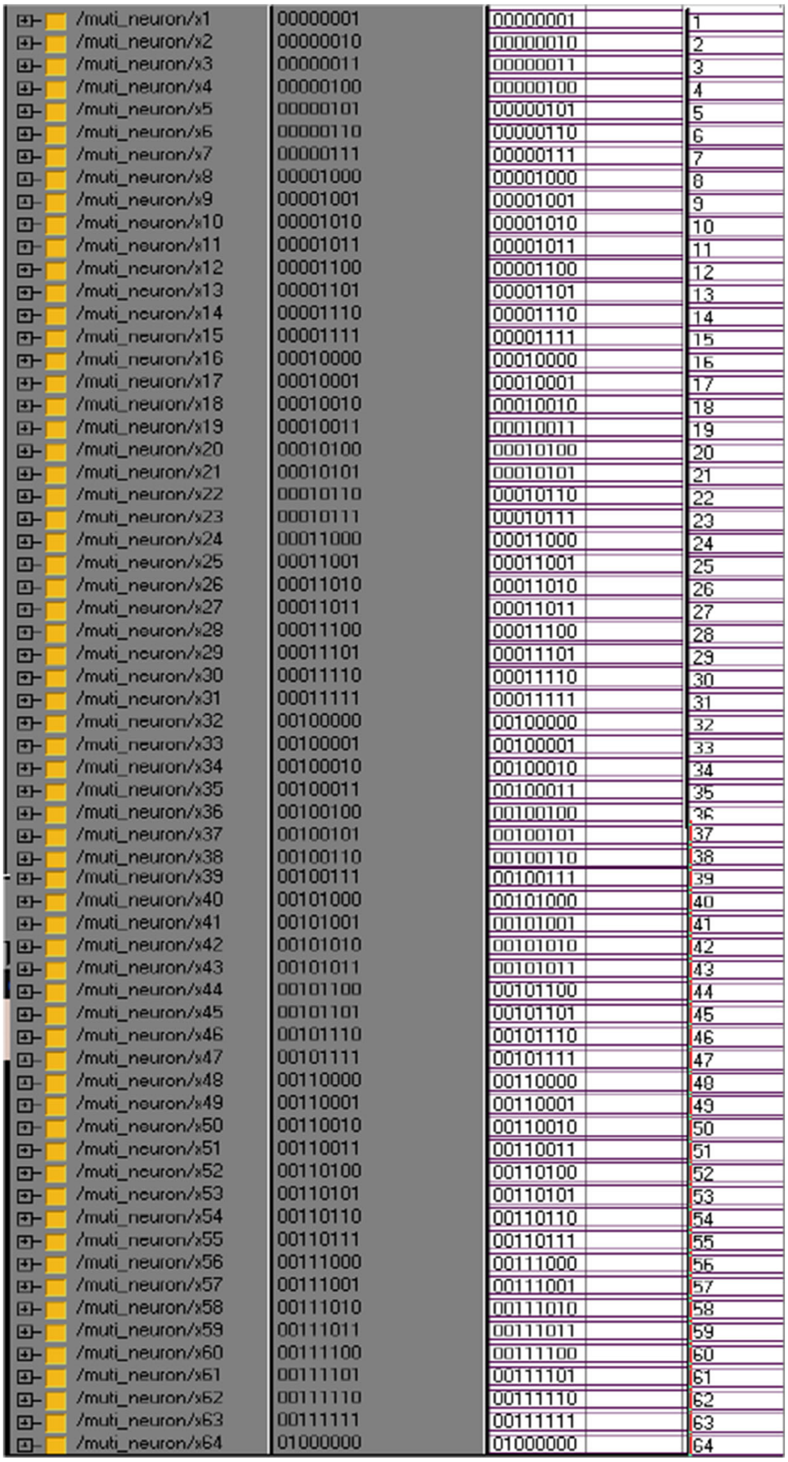
$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \widehat{y_i}\right|^2 \tag{9}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left|y_i - \widehat{y_i}\right|^2} \tag{10}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{\left|y_i - \widehat{y_i}\right|}{y_i}.100\% \tag{11}$$

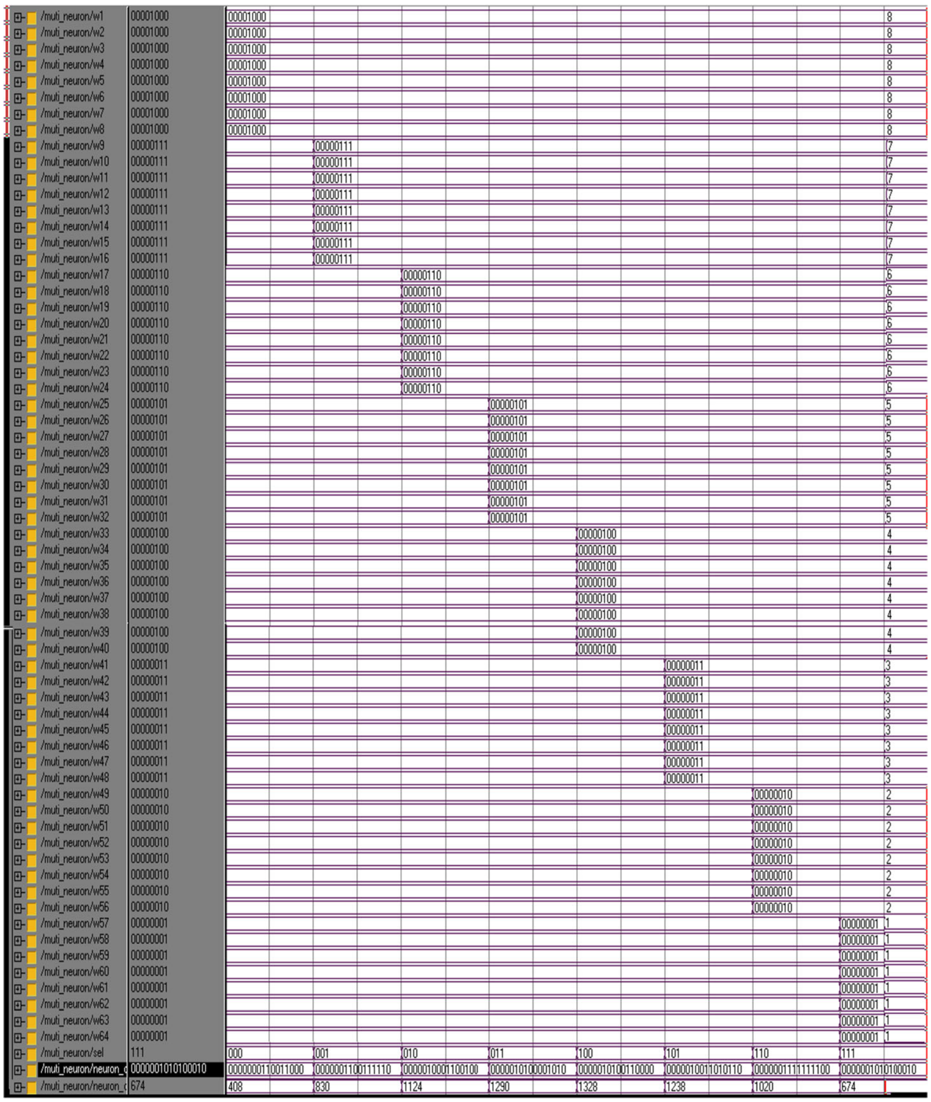
y_i is the actual value and $\widehat{y_i}$ is the predicted value for 'n' number of predications. Based on linear regression model and 200 estimations for 64 number of neurons the value of MSE = 0.00500, RMSE = 0.07071, and MAPE = − 0.003906%.

The efficiency of the hardware simulation depends on the resources utilization such as logic gates, input/output block, combinational logic, memory, and delay. For the complex nonlinear application, multilayer perceptron architecture is beneficial in comparison to single-layer multiple input ANN. On the other hand, it is simple to build up and train a single layer



/muti_neuron/x1	00000001	00000001	1
/muti_neuron/x2	00000010	00000010	2
/muti_neuron/x3	00000011	00000011	3
/muti_neuron/x4	00000100	00000100	4
/muti_neuron/x5	00000101	00000101	5
/muti_neuron/x6	00000110	00000110	6
/muti_neuron/x7	00000111	00000111	7
/muti_neuron/x8	00001000	00001000	8
/muti_neuron/x9	00001001	00001001	9
/muti_neuron/x10	00001010	00001010	10
/muti_neuron/x11	00001011	00001011	11
/muti_neuron/x12	00001100	00001100	12
/muti_neuron/x13	00001101	00001101	13
/muti_neuron/x14	00001110	00001110	14
/muti_neuron/x15	00001111	00001111	15
/muti_neuron/x16	00010000	00010000	16
/muti_neuron/x17	00010001	00010001	17
/muti_neuron/x18	00010010	00010010	18
/muti_neuron/x19	00010011	00010011	19
/muti_neuron/x20	00010100	00010100	20
/muti_neuron/x21	00010101	00010101	21
/muti_neuron/x22	00010110	00010110	22
/muti_neuron/x23	00010111	00010111	23
/muti_neuron/x24	00011000	00011000	24
/muti_neuron/x25	00011001	00011001	25
/muti_neuron/x26	00011010	00011010	26
/muti_neuron/x27	00011011	00011011	27
/muti_neuron/x28	00011100	00011100	28
/muti_neuron/x29	00011101	00011101	29
/muti_neuron/x30	00011110	00011110	30
/muti_neuron/x31	00011111	00011111	31
/muti_neuron/x32	00100000	00100000	32
/muti_neuron/x33	00100001	00100001	33
/muti_neuron/x34	00100010	00100010	34
/muti_neuron/x35	00100011	00100011	35
/muti_neuron/x36	00100100	00100100	36
/muti_neuron/x37	00100101	00100101	37
/muti_neuron/x38	00100110	00100110	38
/muti_neuron/x39	00100111	00100111	39
/muti_neuron/x40	00101000	00101000	40
/muti_neuron/x41	00101001	00101001	41
/muti_neuron/x42	00101010	00101010	42
/muti_neuron/x43	00101011	00101011	43
/muti_neuron/x44	00101100	00101100	44
/muti_neuron/x45	00101101	00101101	45
/muti_neuron/x46	00101110	00101110	46
/muti_neuron/x47	00101111	00101111	47
/muti_neuron/x48	00110000	00110000	48
/muti_neuron/x49	00110001	00110001	49
/muti_neuron/x50	00110010	00110010	50
/muti_neuron/x51	00110011	00110011	51
/muti_neuron/x52	00110100	00110100	52
/muti_neuron/x53	00110101	00110101	53
/muti_neuron/x54	00110110	00110110	54
/muti_neuron/x55	00110111	00110111	55
/muti_neuron/x56	00111000	00111000	56
/muti_neuron/x57	00111001	00111001	57
/muti_neuron/x58	00111010	00111010	58
/muti_neuron/x59	00111011	00111011	59
/muti_neuron/x60	00111100	00111100	60
/muti_neuron/x61	00111101	00111101	61
/muti_neuron/x62	00111110	00111110	62
/muti_neuron/x63	00111111	00111111	63
/muti_neuron/x64	01000000	01000000	64

Fig. 8 Modelsim simulation of 64 input ANN in binary and integer (inputs)



```
/muti_neuron/w1    00001000   00001000   8
/muti_neuron/w2    00001000   00001000   8
/muti_neuron/w3    00001000   00001000   8
/muti_neuron/w4    00001000   00001000   8
/muti_neuron/w5    00001000   00001000   8
/muti_neuron/w6    00001000   00001000   8
/muti_neuron/w7    00001000   00001000   8
/muti_neuron/w8    00001000   00001000   8
/muti_neuron/w9    00000111   00000111   7
/muti_neuron/w10   00000111   00000111   7
/muti_neuron/w11   00000111   00000111   7
/muti_neuron/w12   00000111   00000111   7
/muti_neuron/w13   00000111   00000111   7
/muti_neuron/w14   00000111   00000111   7
/muti_neuron/w15   00000111   00000111   7
/muti_neuron/w16   00000111   00000111   7
/muti_neuron/w17   00000110   00000110   6
/muti_neuron/w18   00000110   00000110   6
/muti_neuron/w19   00000110   00000110   6
/muti_neuron/w20   00000110   00000110   6
/muti_neuron/w21   00000110   00000110   6
/muti_neuron/w22   00000110   00000110   6
/muti_neuron/w23   00000110   00000110   6
/muti_neuron/w24   00000110   00000110   6
/muti_neuron/w25   00000101   00000101   5
/muti_neuron/w26   00000101   00000101   5
/muti_neuron/w27   00000101   00000101   5
/muti_neuron/w28   00000101   00000101   5
/muti_neuron/w29   00000101   00000101   5
/muti_neuron/w30   00000101   00000101   5
/muti_neuron/w31   00000101   00000101   5
/muti_neuron/w32   00000101   00000101   5
/muti_neuron/w33   00000100   00000100   4
/muti_neuron/w34   00000100   00000100   4
/muti_neuron/w35   00000100   00000100   4
/muti_neuron/w36   00000100   00000100   4
/muti_neuron/w37   00000100   00000100   4
/muti_neuron/w38   00000100   00000100   4
/muti_neuron/w39   00000100   00000100   4
/muti_neuron/w40   00000100   00000100   4
/muti_neuron/w41   00000011   00000011   3
/muti_neuron/w42   00000011   00000011   3
/muti_neuron/w43   00000011   00000011   3
/muti_neuron/w44   00000011   00000011   3
/muti_neuron/w45   00000011   00000011   3
/muti_neuron/w46   00000011   00000011   3
/muti_neuron/w47   00000011   00000011   3
/muti_neuron/w48   00000011   00000011   3
/muti_neuron/w49   00000010   00000010   2
/muti_neuron/w50   00000010   00000010   2
/muti_neuron/w51   00000010   00000010   2
/muti_neuron/w52   00000010   00000010   2
/muti_neuron/w53   00000010   00000010   2
/muti_neuron/w54   00000010   00000010   2
/muti_neuron/w55   00000010   00000010   2
/muti_neuron/w56   00000010   00000010   2
/muti_neuron/w57   00000001   00000001   1
/muti_neuron/w58   00000001   00000001   1
/muti_neuron/w59   00000001   00000001   1
/muti_neuron/w60   00000001   00000001   1
/muti_neuron/w61   00000001   00000001   1
/muti_neuron/w62   00000001   00000001   1
/muti_neuron/w63   00000001   00000001   1
/muti_neuron/w64   00000001   00000001   1
/muti_neuron/sel   111   000  001  010  011  100  101  110  111
/muti_neuron/neuron_c 0000001010100010   0000000110011000  0000001100111110  0000010001100100  0000010100001010  0000010100110000  0000010011010110  0000001111111100  0000001010100010
/muti_neuron/neuron_c 674   408  830  1124  1290  1328  1238  1020  674
```

Fig. 9 Modelsim simulation of 64 input ANN in binary (weights and outputs)

perceptron. The neural network model can be explicitly linked to statistical models, allowing it to share the covariance Gaussian density function. The realization of the MLP will provide more delay in comparison to single-layer multiple input ANN. Figure 11 shows the hardware efficiency with targeted FPGA- Virtex-5 for simulation and synthesis of the binary data. The efficiency variations are noticed with the different test cases in which 8 neurons are processed at a time and parallel processing modular design-based approach is followed to realize the 64 input ANN. The single-layer ANN hardware is used to solve simple problems and parallel processing provides fast computation time. In terms of hardware efficiency, the single-layer will provide faster response and computation time in comparison to MLP. The MLP requires more delay to compute the logic as it is processed by different hidden layers. The output

Table 4 Test cases for the simulation waveform ANN-64 point

Pin	Direction	Binary	Integer	Pin	Direction	Binary	Integer
Test Case-1							
X_1<7:0>	Input	00000001	1	W_1<7:0>	Input	00001000	8
X_2<7:0>	Input	00000010	2	W_2<7:0>	Input	00001000	8
X_3<7:0>	Input	00000011	3	W_3<7:0>	Input	00001000	8
X_4<7:0>	Input	00000100	4	W_4<7:0>	Input	00001000	8
X_5<7:0>	Input	00000101	5	W_5<7:0>	Input	00001000	8
X_6<7:0>	Input	00000110	6	W_6<7:0>	Input	00001000	8
X_7<7:0>	Input	0000011	7	W_7<7:0>	Input	00001000	8
X_8<7:0>	Input	00001000	8	W_8<7:0>	Input	00001000	8
Sel<2:0>	Input	000					
b_i_1<15:0>	Input	0000000001111000	120				
Y_1<15:0>	output	0000001100110000	408				
Test Case-2							
X_9<7:0>	Input	00001001	9	W_9<7:0>	Input	00000111	7
X_{10}<7:0>	Input	00001010	10	W_{10}<7:0>	Input	00000111	7
X_{11}<7:0>	Input	00001011	11	W_{11}<7:0>	Input	00000111	7
X_{12}<7:0>	Input	00001100	12	W_{12}<7:0>	Input	00000111	7
X_{13}<7:0>	Input	00001101	13	W_{13}<7:0>	Input	00000111	7
X_{14}<7:0>	Input	00001110	14	W_{14}<7:0>	Input	00000111	7
X_{15}<7:0>	Input	00001111	15	W_{15}<7:0>	Input	00000111	7
X_{16}<7:0>	Input	00010000	16	W_{16}<7:0>	Input	00000111	7
Sel<2:0>	Input	001					
b_i_2<15:0>	Input	0000000000100010	130				
Y_2<15:0>	output	0000001100111110	830				
Test Case-3							
X_{17}<7:0>	Input	00010001	17	W_{17}<7:0>	Input	00000110	6
X_{18}<7:0>	Input	00010010	18	W_{18}<7:0>	Input	00000110	6
X_{19}<7:0>	Input	00010011	19	W_{19}<7:0>	Input	00000110	6
X_{20}<7:0>	Input	00010100	20	W_{20}<7:0>	Input	00000110	6
X_{21}<7:0>	Input	00010101	21	W_{21}<7:0>	Input	00000110	6
X_{22}<7:0>	Input	00010110	22	W_{22}<7:0>	Input	00000110	6
X_{23}<7:0>	Input	00010111	23	W_{23}<7:0>	Input	00000110	6
X_{24}<7:0>	Input	00011000	24	W_{24}<7:0>	Input	00000110	6
Sel<2:0>	Input	010					
b_i_3<15:0>	Input	0000000010001100	140				
Y_3<15:0>	output	0000010001100100	1124				
Test Case-4							
X_{25}<7:0>	Input	00011001	25	W_{25}<7:0>	Input	00000101	5
X_{26}<7:0>	Input	00011010	26	W_{26}<7:0>	Input	00000101	5
X_{27}<7:0>	Input	00011011	27	W_{27}<7:0>	Input	00000101	5
X_{28}<7:0>	Input	00011100	28	W_{28}<7:0>	Input	00000101	5
X_{29}<7:0>	Input	00011101	29	W_{29}<7:0>	Input	00000101	5
X_{30}<7:0>	Input	00011110	30	W_{30}<7:0>	Input	00000101	5
X_{31}<7:0>	Input	00011111	31	W_{31}<7:0>	Input	00000101	5
X_{32}<7:0>	Input	00100000	32	W_{32}<7:0>	Input	00000101	5
Sel <2:0>	Input	011					
b_i_4<15:0>	Input	0000000010010110	150				
Y_4<15:0>	output	0000010100001010	1290				
Test Case-5							
X_{33}<7:0>	Input	00100001	33	W_{33}<7:0>	Input	00000100	4
X_{34}<7:0>	Input	00100010	34	W_{34}<7:0>	Input	00000100	4
X_{35}<7:0>	Input	00100011	35	W_{35}<7:0>	Input	00000100	4
X_{36}<7:0>	Input	00100100	36	W_{36}<7:0>	Input	00000100	4
X_{37}<7:0>	Input	00100101	37	W_{37}<7:0>	Input	00000100	4
X_{38}<7:0>	Input	00100110	38	W_{38}<7:0>	Input	00000100	4
X_{39}<7:0>	Input	00100111	39	W_{39}<7:0>	Input	00000100	4

Table 4 (continued)

Pin	Direction	Binary	Integer	Pin	Direction	Binary	Integer
X_{40}<7:0>	Input	00101000	40	W_{40}<7:0>	Input	00000100	4
Sel <2:0>	Input	100					
b_i_5<15:0>	Input	0000000010100000	160				
Y_5<15:0>	output	0000010100110000	1328				
Test Case-6							
X_{41}<7:0>	Input	00101001	41	W_{41}<7:0>	Input	00000011	3
X_{42}<7:0>	Input	00101010	42	W_{42}<7:0>	Input	00000011	3
X_{43}<7:0>	Input	00101011	43	W_{43}<7:0>	Input	00000011	3
X_{44}<7:0>	Input	00101100	44	W_{44}<7:0>	Input	00000011	3
X_{45}<7:0>	Input	00101101	45	W_{45}<7:0>	Input	00000011	3
X_{46}<7:0>	Input	00101110	46	W_{46}<7:0>	Input	00000011	3
X_{47}<7:0>	Input	00101111	47	W_{47}<7:0>	Input	00000011	3
X_{48}<7:0>	Input	00110000	48	W_{48}<7:0>	Input	00000011	3
Sel <2:0>	Input	101					
b_i_6<15:0>	Input	0000000010101010	170				
Y_6<15:0>	output	0000010011010110	1238				
Test Case-7							
X_{49}<7:0>	Input	00110001		W_{49}<7:0>	Input	00000010	2
X_{50}<7:0>	Input	00110010		W_{50}<7:0>	Input	00000010	2
X_{51}<7:0>	Input	00110011		W_{51}<7:0>	Input	00000010	2
X_{52}<7:0>	Input	00110100		W_{52}<7:0>	Input	00000010	2
X_{53}<7:0>	Input	00110101		W_{53}<7:0>	Input	00000010	2
X_{54}<7:0>	Input	00110110		W_{54}<7:0>	Input	00000010	2
X_{55}<7:0>	Input	00110111		W_{55}<7:0>	Input	00000010	2
X_{56}<7:0>	Input	00111000		W_{56}<7:0>	Input	00000010	2
Sel <2:0>	Input	110					
b_i_7<15:0>	Input	0000000010110100	180				
Y_6<15:0>	output	0000001111111100	1020				
Test Case-8							
X_{57}<7:0>	Input	00111001		W_{57}<7:0>	Input	00000001	1
X_{58}<7:0>	Input	00111010		W_{58}<7:0>	Input	00000001	1
X_{59}<7:0>	Input	00111011		W_{59}<7:0>	Input	00000001	1
X_{60}<7:0>	Input	00111100		W_{60}<7:0>	Input	00000001	1
X_{61}<7:0>	Input	00111101		W_{61}<7:0>	Input	00000001	1
X_{62}<7:0>	Input	00111110		W_{62}<7:0>	Input	00000001	1
X_{63}<7:0>	Input	00111111		W_{63}<7:0>	Input	00000001	1
X_{64}<7:0>	Input	01000000		W_{64}<7:0>	Input	00000001	1
Sel <2:0>	Input	111					
b_i_8<15:0>	Input	0000000010111110	190				
Y_7<15:0>	output	0000001010100010	674				

Table 5 Xilinx software parameters for ANN-8 point to ANN-64 point

Size/Parameters	Multipliers	16- bit Adders	Slices	LUTs	IOB	Delay(ns)	Memory (kB)
ANN-8	8	8	379	648	147	37.091	116,736
ANN-16	16	16	726	1280	275	37.091	124,480
ANN-24	24	24	1073	1928	403	37.091	130,048
ANN-32	32	32	1420	2560	531	37.091	137,280
ANN-40	40	40	1766	3208	659	37.091	143,424
ANN-48	48	48	2113	3840	787	37.091	151,556
ANN-56	56	56	2460	4488	915	37.091	158,724
ANN-64	64	64	2807	5120	1043	37.091	165,892

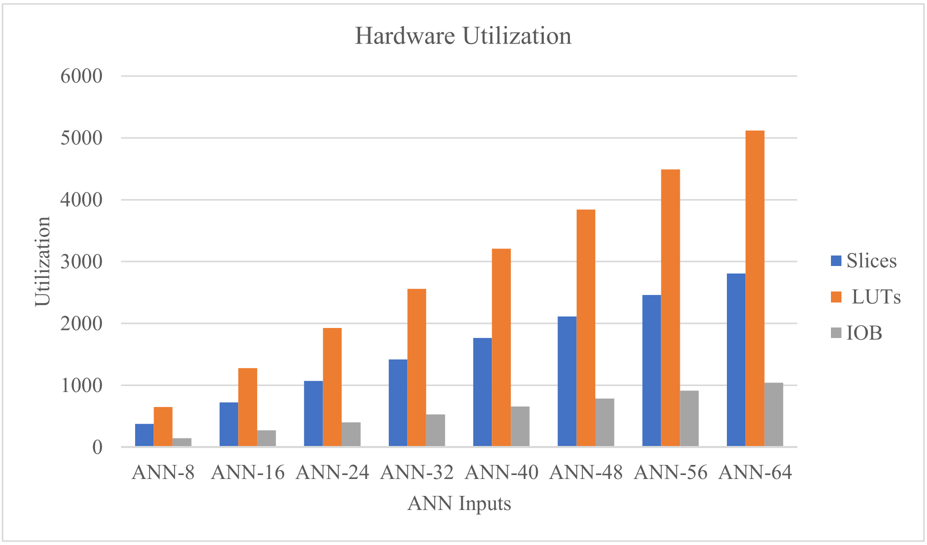


Fig. 10 Hardware utilization for ANN-8 to ANN-64 hardware chip

function of the ANN hardware chip is the throughput that depends on the length of binary input, weights, bias input, and hardware and timing parameters. The output layer receives the inputs from the layers above it, executes the calculations using its neurons, and then computes the output.

The hardware delay depends on two components of propagation delay are logic delay and routing delay. The logic delay is a function of the number and kind of logic gates the signal passes through. Because the FPGA compiler tries to cluster the components of a combinatorial path as tightly as possible on the FPGA. The routing delay is a function of the length of the wire path the signal travels, which is often modest. In the simulation, the total path delay is

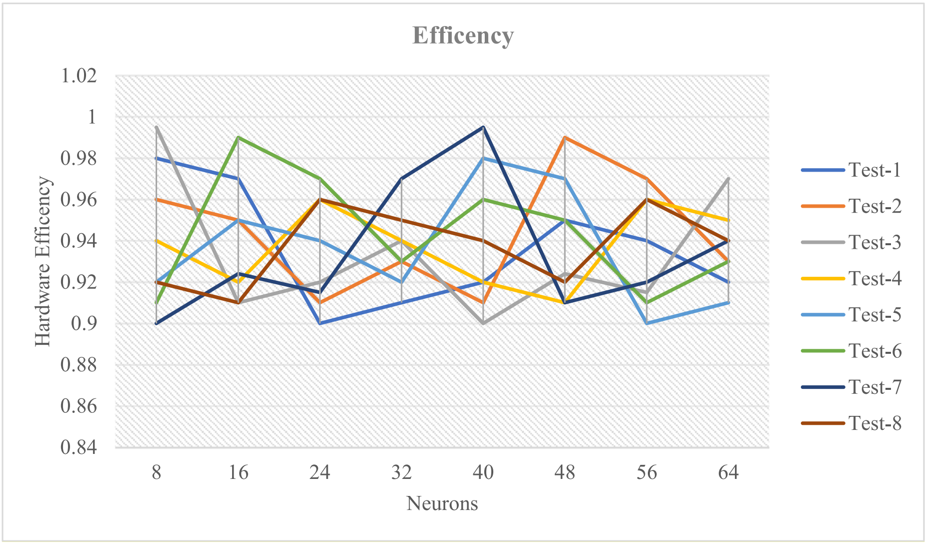


Fig. 11 Hardware efficiency with targeted FPGA

37.091 ns, in which 89.00% delay is from logic and 11.00% from routing that help to maintain the FPGA efficiency greater than 90.00% in most cases.

6 Conclusions

ANNs are known for their high degree of connectedness and massive data volumes. For the realization of the single-layer networks, neuron-level parallelism is more effective. The intrinsic distributed component of ANNs is in both memory and computational logic, suggesting that the implementation will be done directly in hardware, allowing for significant benefits as network sizes grow. The hardware chip The scalable chip design of 8 input ANN and 64 input ANN is performed successfully in Xilinx ISE 14.7. The Modelsim simulation is verified under different test cases and hardware parameters are extracted from the targeted device of Virtex-5 FPGA. The number of multipliers/adders for ANN-8, ANN-16, ANN-24, ANN-32, ANN-40, ANN-48, ANN-56 and ANN-64 are 8, 16, 24, 32, 40, 48, 56, and 64 respectively. The number of slices for ANN-8, ANN-16, ANN-24, ANN-32, ANN-40, ANN-48, ANN-56, and ANN-64 are 379, 726, 1073, 1420, 1766, 2113, 2460, and 2807 respectively. The number of LUTs for ANN-8, ANN-16, ANN-24, ANN-32, ANN-40, ANN-48, ANN-56, and ANN-64 are 648, 1280, 1928, 2560, 3208, 3840, 4488, and 5120 respectively. In the same way, the reported number of IOBs are 147, 275, 403, 531, 657, 787, 915, and 1043 for ANN-8 to ANN-64 respectively. The combinational path delay is 37.091 ns, common to all scalable modules. The hardware efficiency of the design is greater than 90.00% with the MSE = 0.00500 for ANN-64. The hardware usage summary concludes that the ANN chip hardware utilization is increasing with the ANN cluster size. The memory is also increasing from 116,736 kB to 165,892 kB. The chip hardware requirements will increase definitely with the number of neuron inputs. The biggest challenge for the hardware is to develop an embedded chip that can be compatible to support the specific hardware. The limitation of the work is that the chip design supports the 64 neurons processing ANN hardware and the chip functionality is verified in Virtex-5 FPGA. Therefore, the device resources utilization and timing parameters will change on another series of FPGA. The design can be extended further for large-scale ANN using pipelined and parallel processing that supports maximum hardware resources count and combinational blocks on the targeted FPGA. In the research work, we have followed the concept of scalable computing and modular design that can be used to support the design and development of the large-scale neuromorphic embedded chip. In the future, the research can be focused on the hardware chip design and synthesis for multilayer neural network architecture.

Declarations

Conflict of interest The authors declare that there is no conflict of interest regarding the publication of this paper.

References

1. Abiodun OI, Jantan A, Omolara AE, Dada KV, Mohamed NA, Arshad H (2018) State-of-the-art in artificial neural network applications: a survey. Heliyon 4(11):e00938. https://doi.org/10.1016/j.heliyon.2018.e00938
2. Adolphs R (2003) Cognitive neuroscience of human social behavior. Nat Rev Neurosci 4(3):165–178. https://doi.org/10.1038/nrn1056

3. Alçın M, Pehlivan İ, Koyuncu İ (2016) Hardware design and implementation of a novel ANN-based chaotic generator in FPGA. Optik 127(13):5500–5505. https://doi.org/10.1016/j.ijleo.2016.03.042
4. Alcin M, Koyuncu I, Tuna M, Varan M, Pehlivan I (2019) A novel high speed artificial neural network–based chaotic true random number generator on field programmable gate Array. Int J Circ Theory Appl 47(3):365–378. https://doi.org/10.1002/cta.2581
5. Ali HH, Haweel MT (2012) Legendre neural networks with multi input multi output system equations. 2012 Seventh International Conference on Computer Engineering & Systems (ICCES). IEEE. https://doi.org/10.1109/ICCES.2012.6408490
6. Ali HK, Mohammed EZ (2010) Design artificial neural network using FPGA. IJCSNS 10(8):88
7. Amir R, Devor M (2003) Electrical excitability of the soma of sensory neurons is required for spike invasion of the soma, but not for through-conduction. Biophys J 84(4):2181–2191. https://doi.org/10.1016/S0006-3495(03)75024-3
8. Amudha V, Venkataramani B (2009) System on programmable chip implementation of neural network-based isolated digit recognition system. Int J Electron 96(2):153–163. https://doi.org/10.1080/00207210802526828
9. Awotunde JB, Folorunso SO, Bhoi AK, Adebayo PO, Ijaz MF (2021) Disease diagnosis system for IoT-based wearable body sensors with machine learning algorithm. In: Hybrid Artificial Intelligence and IoT in Healthcare (pp. 201–222). Springer, Singapore. https://doi.org/10.1007/978-981-16-2972-3_10
10. Baliyan A, Gaurav K, Mishra SK (2015) A review of short-term load forecasting using artificial neural network models. Procedia Comput Sci 48:121–125. https://doi.org/10.1016/j.procs.2015.04.160
11. Belabed T, Coutinho MGF, Fernandes MA, Sakuyama CV, Souani C (2021) User-driven FPGA-based design automated framework of deep neural networks for low-power low-cost edge computing. IEEE Access 9:89162–89180. https://doi.org/10.1109/ACCESS.2021.3090196
12. Carrillo S, Harkin J, McDaid LJ, Morgan F, Pande S, Cawley S, McGinley B (2012) Scalable hierarchical network-on-chip architecture for spiking neural network hardware implementations. IEEE Trans Parallel Distrib Syst 24(12):2451–2461. https://doi.org/10.1109/TPDS.2012.289
13. Carvalho MB, Amaral AM, da Silva Ramos LE, da Silva Martins CAP, Ekel P (2005) Artificial neural network engine: parallel and parameterized architecture implemented in FPGA. In: International Conference on Pattern Recognition and Machine Intelligence (pp. 294-299). Springer, Berlin, Heidelberg. https://doi.org/10.1007/11590316_42
14. Carvalho AR, Ramos FM, Chaves AA (2011) Metaheuristics for the feedforward artificial neural network (ANN) architecture optimization problem. Neural Comput & Applic 20(8):1273–1284. https://doi.org/10.1007/s00521-010-0504-3
15. Cheung K, Schultz SR, Luk W (2012) A large-scale spiking neural network accelerator for FPGA systems. In: International Conference on Artificial Neural Networks (pp. 113-120). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-33269-2_15
16. Claveria O, Monte E, Torra S (2015) Multiple-input multiple-output vs. single-input single-output neural network forecasting. In: Research Institute of Applied Economics (pp. 2015-02). Barcelona University
17. Dhaka VS, Meena SV, Rani G, Sinwar D, Ijaz MF, Woźniak M (2021) A survey of deep convolutional neural networks applied for prediction of plant leaf diseases. Sensors 21(14):4749. https://doi.org/10.3390/s21144749
18. El-Madany HT, Fahmy FH, El-Rahman NM, Dorrah HT (2012) Design of FPGA based neural network controller for earth station power system. TELKOMNIKA Indones J Electr Eng 10(2):281–290
19. Fan GF, Wei X, Li YT, Hong WC (2020) Forecasting electricity consumption using a novel hybrid model. Sustain Cities Soc 61:102320. https://doi.org/10.1016/j.scs.2020.102320
20. Fan GF, Yu M, Dong SQ, Yeh YH, Hong WC (2021) Forecasting short-term electricity load using hybrid support vector regression with grey catastrophe and random forest modeling. Util Policy 73:101294. https://doi.org/10.1016/j.jup.2021.101294
21. Gevrey M, Dimopoulos I, Lek S (2003) Review and comparison of methods to study the contribution of variables in artificial neural network models. Ecol Model 160(3):249–264. https://doi.org/10.1016/S0304-3800(02)00257-0
22. Goel A, Chikara D, Srivastava AK, Kumar A (2016) Medical imaging with brain tumor detection and analysis. Int J Comput Sci Inf Secur 14(9):228. https://sites.google.com/site/ijcsis/. Accessed Sept 2016
23. Gomperts A, Ukil A, Zurfluh F (2010) Development and implementation of parameterized FPGA-based general-purpose neural networks for online applications. IEEE Trans Ind Inform 7(1):78–89. https://doi.org/10.1109/TII.2010.2085006
24. Gupta N, Jain A, Vaisla KS, Kumar A, Kumar R (2021) Performance analysis of DSDV and OLSR wireless sensor network routing protocols using FPGA hardware and machine learning. Multimed Tools Appl 80:1–19. https://doi.org/10.1007/s11042-021-10820-4

25. Gupta N, Vaisla KS, Jain A, Kumar A, Kumar R (2021) Performance analysis of AODV routing for wireless sensor network in FPGA hardware. Comput Syst Sci Eng 39(2):1–12. https://doi.org/10.32604/csse.2022.019911
26. Hamedi S, Jahromi HD (2021) Performance analysis of all-optical logical gate using artificial neural network. Expert Syst Appl 178:115029. https://doi.org/10.1016/j.eswa.2021.115029
27. Hassan M, Paulavicius R, Filatovas E, Iftekhar A (2021) A Blockchain-based intelligent machine learning system for smart health care. Preprints, 2021110034. https://doi.org/10.1016/j.inffus.2020.06.008.
28. Himavathi S, Anitha D, Muthuramalingam A (2007) Feedforward neural network implementation in FPGA using layer multiplexing for effective resource utilization. IEEE Trans Neural Netw 18(3):880–888. https://doi.org/10.1109/TNN.2007.891626
29. Huang CJ, Kuo PH (2019) Multiple-input deep convolutional neural network model for short-term photovoltaic power forecasting. IEEE Access 7:74822–74834. https://doi.org/10.1109/ACCESS.2019.2921238
30. Huang GB, Chen YQ, Babri HA (2000) Classification ability of single hidden layer feedforward neural networks. IEEE Trans Neural Netw 11(3):799–801. https://doi.org/10.1109/72.846750
31. Ijaz MF, Alfian G, Syafrudin M, Rhee J (2018) Hybrid prediction model for type 2 diabetes and hypertension using DBSCAN-based outlier detection, synthetic minority over-sampling technique (SMOTE), and random forest. Appl Sci 8(8):1325. https://doi.org/10.3390/app8081325
32. Ijaz MF, Attique M, Son Y (2020) Data-driven cervical cancer prediction model with outlier detection and over-sampling methods. Sensors 20(10):2809. https://doi.org/10.3390/s20102809
33. Joseph C, Gupta A (2010) A novel hardware efficient Digital Neural Network architecture implemented in 130nm technology. In: Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on (Vol. 3, pp. 82–87). IEEE. https://doi.org/10.1109/ICCAE.2010.5452015
34. Karbachevsky A, Baskin C, Zheltonozhskii E, Yermolin Y, Gabbay F, Bronstein AM, Mendelson A (2021) Early-stage neural network hardware performance analysis. Sustainability 13(2):717. https://doi.org/10.3390/su13020717
35. Kumar A, Baruah L, Sabu A (2015) Rotator on-chip (RoC) design based on ring topological NoC. Procedia Comput Sci 45:540–548. https://doi.org/10.1016/j.procs.2015.03.099
36. Kumar A, Kuchhal P, Singhal S (2015) Design and FPGA synthesis of three stage telecommunication switching in HDL environment. Procedia Comput Sci 48:454–460. https://doi.org/10.1016/j.procs.2015.04.119
37. Kumar A, Sharma P, Gupta MK, Kumar R (2018) Machine learning based resource utilization and pre-estimation for network on chip (NoC) communication. Wirel Pers Commun 102(3):2211–2231. https://doi.org/10.1007/s11277-018-5376-3
38. Kumar R, Ahuja NJ, Saxena M, Kumar A (2020) Automotive power window communication with DTC algorithm and hardware-in-the-loop testing. Wirel Pers Commun 114:3351–3366. https://doi.org/10.1007/s11277-020-07535-4
39. Liu J, Harkin J, Maguire LP, McDaid LJ, Wade JJ, Martin G (2016) Scalable networks-on-chip interconnected architecture for astrocyte-neuron networks. IEEE Trans Circ Syst I: Regular Pap 63(12):2290–2303. https://doi.org/10.1109/TCSI.2016.2615051
40. Maeda Y, Wakamura M (2005) Simultaneous perturbation learning rule for recurrent neural networks and its FPGA implementation. IEEE Trans Neural Netw 16(6):1664–1672. https://doi.org/10.1109/TNN.2005.852237
41. Mandal M, Singh PK, Ijaz MF, Shafi J, Sarkar R (2021) A tri-stage wrapper-filter feature selection framework for disease classification. Sensors 21(16):5571. https://doi.org/10.3390/s21165571
42. Mishra VM, Kumar A (2021) Zigbee internode communication and FPGA synthesis using mesh, star and cluster tree topological chip. Wirel Pers Commun 119(2):1321–1339. https://doi.org/10.1007/s11277-021-08282-w
43. Misra J, Saha I (2010) Artificial neural networks in hardware: a survey of two decades of progress. Neurocomputing 74(1–3):239–255. https://doi.org/10.1016/j.neucom.2010.03.021
44. Mohammadhassani M, Nezamabadi-Pour H, Suhatril M, Shariati M (2013) Identification of a suitable ANN architecture in predicting strain in tie section of concrete deep beams. Struct Eng Mech Int J 46(6):853–868
45. Moore SW, Fox PJ, Marsh SJ, Markettos AT, Mujumdar A (2012, April) Bluehive-a field-programable custom computing machine for extreme-scale real-time neural network simulation. In: 2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines (pp. 133-140). IEEE. https://doi.org/10.1109/FCCM.2012.32
46. Muthuramalingam A, Himavathi S, Srinivasan E (2008) Neural network implementation using FPGA: issues and application. World Acad Sci Eng Technol 24:2008
47. Nayak R, Jain LC, Ting BKH (2001) Artificial neural networks in biomedical engineering: a review. Comput Mech New Front New Millennium:887–892. https://doi.org/10.1016/B978-0-08-043981-5.50132-2

48. Novickis R, Justs DJ, Ozols K, Greitāns M (2020) An approach of feed-forward neural network throughput-optimized implementation in FPGA. Electronics 9(12):2193. https://doi.org/10.3390/electronics9122193
49. Ovtcharov K, Ruwase O, Kim JY, Fowers J, Strauss K, Chung ES (2015) Accelerating deep convolutional neural networks using specialized hardware. Microsoft Res Whitepaper 2(11):1–4
50. Pardo CA, Xu Z, Borchelt DR, Price DL, Sisodia SS, Cleveland DW (1995) Superoxide dismutase is an abundant component in cell bodies, dendrites, and axons of motor neurons and in a subset of other neurons. Proc Natl Acad Sci 92(4):954–958. https://doi.org/10.1073/pnas.85.21.8335
51. Posewsky T, Ziener D (2018) Throughput optimizations for FPGA-based deep neural network inference. Microprocess Microsyst 60:151–161. https://doi.org/10.1016/j.micpro.2018.04.004
52. Radway RM, Bartolo A, Jolly PC, Khan ZF, Le BQ, Tandon P, Mitra S (2021) Illusion of large on-chip memory by networked computing chips for neural network inference. Nat Electron 4(1):71–80. https://doi.org/10.1038/s41928-020-00515-3
53. Rawat AS, Rana A, Kumar A, Bagwari A (2018) Application of multi-layer artificial neural network in the diagnosis system: a systematic review. IAES Int J Artif Intell 7(3):138. https://doi.org/10.11591/ijai.v7.i3.pp138-142
54. Sarić R, Jokić D, Beganović N, Pokvić LG, Badnjević A (2020) FPGA-based real-time epileptic seizure classification using artificial neural network. Biomed Signal Process Control 62:102106. https://doi.org/10.1016/j.bspc.2020.102106
55. Srinivasu PN, SivaSai JG, Ijaz MF, Bhoi AK, Kim W, Kang JJ (2021) Classification of skin disease using deep learning neural networks with MobileNet V2 and LSTM. Sensors 21(8):2852. https://doi.org/10.3390/s21082852
56. Srinivasu PN, Ahmed S, Alhumam A, Kumar AB, Ijaz MF (2021) An AW-HARIS based automated segmentation of human liver using CT images. Comput Mater Contin 69(3):3303–3319. https://doi.org/10.32604/cmc.2021.018472
57. Stilling RM, Dinan TG, Cryan JF (2014) Microbial genes, brain & behaviour–epigenetic regulation of the gut–brain axis. Genes Brain Behav 13(1):69–86. https://doi.org/10.1111/gbb.12109
58. Teodoro AA, Gomes OS, Saadi M, Silva BA, Rosa RL, Rodríguez DZ (2021) An FPGA-based performance evaluation of artificial neural network architecture algorithm for IoT. Wirel Pers Commun:1–32. https://doi.org/10.1007/s11277-021-08566-1
59. Tsmots I, Skorokhoda O, Rabyk V (2016) Structure and software model of a parallel-vertical multi-input adder for FPGA implementation. In: 2016 XIth International Scientific and Technical Conference Computer Sciences and Information Technologies (CSIT) (pp. 158-160). IEEE. https://doi.org/10.1109/STC-CSIT.2016.7589894
60. Wan W, Kubendran R, Eryilmaz SB, Zhang W, Liao Y, Wu D, & Wong HSP (2020) 33.1 a 74 tmacs/w CMOS-RRAM neurosynaptic core with dynamically reconfigurable dataflow and in-situ transposable weights for probabilistic graphical models. In: 2020 IEEE International Solid-State Circuits Conference-(ISSCC) (pp. 498–500). IEEE. https://doi.org/10.1109/ISSCC19947.2020.9062979
61. Wu R, Guo X, Du J, Li J (2021) Accelerating neural network inference on FPGA-based platforms-a survey. Electronics 10(9):1025. https://doi.org/10.3390/electronics10091025
62. Zhang W, Gao B, Tang J, Yao P, Yu S, Chang MF, Yoo HJ, Qian H, Wu H (2020) Neuro-inspired computing chips. Nat Electron 3(7):371–382. https://doi.org/10.1038/s41928-020-0435-7
63. Zou Z, Zhao R, Wu Y, Yang Z, Tian L, Wu S, Wang G, Yu Y, Zhao Q, Chen M, Pei J, Chen F, Zhang Y, Song S, Zhao M, Shi L (2020) A hybrid and scalable brain-inspired robotic platform. Sci Rep 10(1):1–13. https://doi.org/10.1038/s41598-020-73366-9