

# Design and implementation of deep neural network hardware chip and its performance analysis

**Aruna Pant, Adesh Kumar**

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

## Article Info

### Article history:

Received Mar 13, 2024

Revised Jun 9, 2024

Accepted Jul 17, 2024

### Keywords:

Control logic

Deep learning neural network

Hidden layers

Performance parameters

Spartan-6 FPGA

## ABSTRACT

The artificial neural network (ANN) with a single layer has a limited capacity to process data. Multiple neurons are connected in the human brain, and the actual capacity of the brain lies in the interconnectedness of multiple neurons. As a specified generalization of ANN deep learning makes use of two or more hidden layers, which implies that a greater number of neurons are required to construct the model. A network that has more than one hidden layer, also known as two or more hidden layers, is referred to as a deep neural network, and the process of training such networks is referred to as deep learning. The research article focuses on the design of a multilayer or deep neural network presented for the target field programmable gate array (FPGA) device spartan-6 (xc6stx4-2t9g144) FPGA. The simulation is carried out using Xilinx ISE and ModelSim software. There are two hidden layers in which (2×1) multiplexer blocks are utilized for processing twenty neurons into the output of ten neurons in the first hidden layer and demultiplexers (1×2) and vice versa. The hardware utilization is estimated on FPGA to compute the performance of the deep neural hardware chip based on memory, flip flops, delay, and frequency parameters. The design is scalable and applicable to various FPGA devices, which makes the work novel. FPGA-based neuromorphic hardware acceleration platform with high speed and low power for discrete spike processing on hardware with great real-time performance.

*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, University of Petroleum and Energy Studies (UPES)

Dehradun, India

Email: [adeshmanav@gmail.com](mailto:adeshmanav@gmail.com)

## 1. INTRODUCTION

Deep neural network (DNN) has a lot of popularity due to which is used for most of the applications. Like detection applications, specific hardware design of multilayer neural networks is required when a general computer is not sufficient as it has limitations in speed, cost of the chip, and consumption of energy. A deep learning model is used here which has two hidden layers. The multilayer neural network is made up of numerous layers between the input and output layers, with all hidden levels being Fully connected.

Deep learning neural network implementation on field programmable gate arrays (FPGAs) presents various challenges, including storage, resource requirements, and memory utilization [1]. Deep neural networks are made up of numerous “deep” and hidden layers, and each one of them does a particular task. Deep learning is learning about different levels of data representations and their underlying distribution of the data [2]. DNNs are most used in video editing applications, such as object detection and real-time speech

translation. DNNs will become increasingly significant in medicine, robotics, and finance for weather forecasting and detection [3]. DNNs are now the basis for many modern artificial intelligence applications [4]. Since the groundbreaking use of DNNs in speech recognition and image recognition, the number of places where DNNs have skyrocketed. These deep neural networks are used in a wide range of applications, including self-driving cars, cancer detection, and complicated gameplay [5]. In most fields, multilayer neural networks can be done much more accurately than humans. Overall, multilayer artificial neural networks (ANNs), particularly deep networks, proved themselves better at handling complicated tasks such as image identification, speech recognition, language translation, and others, making them an essential component of modern artificial intelligence systems [6]. DNNs are used in a wide variety of artificial intelligence applications, including computer vision, speech recognition, robotics, and many more. DNNs can achieve cutting-edge accuracy on a wide variety of artificial intelligence tasks [7]. The computational complexity of these networks is rather high. Better performance, higher bandwidth, and deterministic low latency are achieved with the usage of FPGA and HDL in the design of programmable processors. Parallel processing and operations [8] are the foundation of the architectures of ANN to process neuronal information, regular chips are unable to sustain a huge number of highly simultaneous processes [9]. Several distinct processors that enable parallel processing are included in the artificial intelligence-based hardware chip.

The problem statement of the work is to design the deep neural network hardware chip and estimate its performance on FPGA. The chip is designed for 20 neuron input and each neuron input is of 8 bits. 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 a 16-bit output. The multiple demultiplexer ( $1 \times 2$ ) is used which again converts 10-neuron output into 20-neuron output which is given to the second hidden layer. 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 hidden layer output is then given to the multiplexer which has 20 inputs and one single output. The mux has a 5-bit selection line which can be used to select any of the 20 outputs of the control logic. Hardware utilization characteristics such as slices used, look-up tables (LUTs), flip-flops (FFs), input/output blocks (IOBs), memory, and total propagation latency are used to calculate the deep neural hardware chip's performance.

## 2. RELATED WORK

Although software can easily create multilayer neural network models on computers, there are numerous challenges when designing deep neural networks on silicon. These challenges include implementing the multiplication of input with weights at each layer, designing the storing network weights, and activation function, and selecting an appropriate number format for calculations. To achieve performance efficiency, each layer in a deep ANN architecture, which will be the focus of this work, necessitates its own independent space for memory and its weight. In a deep network that has two or more hidden layers, the memory requirement for the weight matrices of the entire network increases at the fastest as its network's size and memory accesses grow. Furthermore, computations within each layer should be executed in parallel for applications requiring high speed. These characteristics have a substantial influence on the efficiency and performance of DNNs. In this context, the FPGA is viable for implementing a DNN accelerator. The designing of deep ANN and the use of FPGA is very prevalent in research areas [10], [11]. convolution neural networks (CNNs) have been used for real-time embedded systems, exploring the impartial classification of cutting-edge system-on-chips [12]. Multi-dimensional trade-offs are investigated to configure programmable accelerators for neural interpretation using the finest software libraries. The framework is open source to promote fair benchmarking assessments. It has become increasingly common to design multilayer perceptron neural networks on reconfigurable hardware to carry out a wide range of applications, including pattern recognition and classification. The purpose of the research by Huynh [13] was to investigate the combined impact of neural network size and decreased precision number formats, which are utilized for the representation of the ideal parameters, on the recognition rate of a handwritten digit recognition system that is based on neural networks. ANN architecture [14] was evaluated on Virtex-5 FPGA for pattern recognition and tested for scalability and hardware resource use. Deep learning is now the most advanced solution for almost all significant machine-learning tasks in a wide range of domains [15], [16]. Deep learning techniques outperform typical machine learning algorithms that require human feature extraction (handcrafted features) [17]. Deep learning models extract hierarchical features and perform better as the amount of data increases [18]. There are various deep learning methods and architectures applied in the design, such as multi-layer perceptron (MLP), deep belief network (DBN), autoencoder (AE), CNNs, recurrent neural network (RNN) with long short-term memory (LSTM) and gated recurrent units (GRU), generative adversarial network (GAN), deep reinforcement learning (DRL) [19]. These models have spanned a wide range of fields and applications. Deep learning architectures have enabled autonomous driving (AD)

systems [20] to acquire a high level of precision in their performance; these systems are comprised of numerous perception-level tasks. Traditional supervised learning methods are no longer applicable to the many tasks that make up autonomous driving systems, except for perception. In computer vision applications, CNN models have demonstrated encouraging outcomes in areas including robotics, autonomous vehicle vision systems, and drone navigation [21]. In healthcare, CNNs have also outperformed other methods, especially for image identification.

It has been used to detect tumors or other types of lesions by more experienced radiologists [22]. Brain images have been used for humans for magnetic resonance imaging (MRI) that detects disease Alzheimer's with the help of CNN [23]. Deep learning methodologies [24] have proved effective in addressing artificial intelligence difficulties, while FPGAs have been used to improve reconfigurable computer hardware and software for artificial intelligence design. Multilayer Networks have been designed using FPGA [25]. Graphic processing units (GPUs) are unsuitable for embedded applications due to their high-power consumption. Many academics have explored techniques to create deep neural networks on low-power hardware [26]. The most expensive operations in deep convolutional neural networks (DCNNs) are generated by deep CNNs, which are made up of numerous layers of three-dimensional convolutions, each of which has between tens and hundreds of filters. The restriction of flowing data in and out of the hardware chip accelerator, which is synthesized on Xilinx Zynq-7000, is a challenge for systems that run DCNNs. These systems are required to pass 3D input maps to the hardware accelerators to perform convolutions [27]. An implementation of High-Level Synthesis was employed on a Virtex-6 FPGA board, and CNN was utilized [28] to analyze the design flow. The FPGA resources can be decreased by up to 13 times when compared to accelerators that use traditional scratchpad memories, while yet maintaining the same level of performance. A fixed point DNN is designed on FPGA and uses any external memory. Its power consumption and time of execution are compared with GPU-based implementation [29]. To construct feed-forward deep neural networks in real-time, GPUs are commonly used due to the high amount of arithmetic and memory operations required [30]. The implementation of an optimized deep learning architecture was carried out on a parallel computing platform. This architecture featured flexible layer topologies and fast matrix operation kernels. This is the design to support minimum hardware utilization [31]. The integration of several types of deep architectures into a unified neural training-testing system is accomplished by the implementation of layer-wise procedures that have been meticulously designed. A neural network hardware model can be designed on FPGA. This neural model contains more types of layers, connections, and data [32]. Retraining via backpropagation was created as an alternative to direct weight quantization [33]. Feedforward DNNs with several hidden layers function well in many requests, but they require complex hardware. The system hardware complexity can be minimized by reducing the weights and signal word length. Quantized weights and fixed-point signals are used in backpropagation for retraining and output computation. However, precision numbers are used to change networks [34]. The developed networks typically used 2 to 8 bits for weights and more than 7 bits to represent signals in analog or high-precision fixed points [35]. Limited work has been done in the domain of hardware chip design and synthesis on the specific hardware specifically targeting any FPGA hardware and deep neural network processing. The problem statement of the presented work is to design the scalable hardware chip of a deep neural network and evaluate the system performance in terms of delay, frequency, LUTs, and memory.

### 3. METHODS

The system design includes an input layer, two hidden layers, and only one output layer. The design of the deep neural hardware chip consists of hidden layers which are more than one. The two hidden layers are used, and multipliers are used for multiplying the neuron inputs with the neuron's weight, apart from the (2×1) multiplexer component, the demultiplexer (1×2) component and one (8×1) multiplexer component are used. Figure 1 presents the basic building blocks for the multilayer Feed-forward artificial neural network (MFFANN). Figure 2 represents the design of a multilayer neural network. It consists of one input, two hidden layers, and one output layer. The design of this multilayer ANN is done in VHSIC Hardware Description Language (VHDL) using an FPGA Spartran-6 device. The main parts of the system design are as follows.

- Multiplier: The input layer is implemented using a multiplier that multiplies (0-19) neuron input with 20 neuron weights and each input consists of 8-bit neuron input and gives a single 16-bit output after multiplication.  $X_0, X_1, \dots, X_{19}$  as the inputs of neurons, and  $W_0, W_1, \dots, W_{19}$  as the weights by multiplying this in the multiplier we get the output  $Y_0, Y_1, \dots, Y_{19}$ , which is 16-bit.
- Control Logic: The control logic is used for the hidden layer. The output of the multiplier, which is  $Y_0, Y_1, Y_2, \dots, Y_{19}$  is given to the control logic the input of the control logic is taken as  $M_0, M_1,$

$M_2, \dots, M_{19}$  each is of 16-bit input and  $N_0, N_1, N_2, \dots, N_{19}$  is the output of the control logic which is also 16-bit. After this, the  $2 \times 1$  multiplexer is used.

- Multiplexer ( $2 \times 1$ ): There are 20 outputs of the control logic each pair of two outputs is given to a ( $2 \times 1$ ) multiplexer, so the 20 outputs of the control logic now become 10 outputs.
- Demultiplexer ( $1 \times 2$ ): The 10 outputs of the ( $2 \times 1$ ) multiplexer are again given to the ( $1 \times 2$ ) demultiplexer which now becomes 20 outputs, and the output of the ( $1 \times 2$ ) demultiplexer is again given to control logic which is the second hidden layer. The 20 outputs of the control logic are given to the 20 to 1 demultiplexer so we get the single output of 16 bits from this.

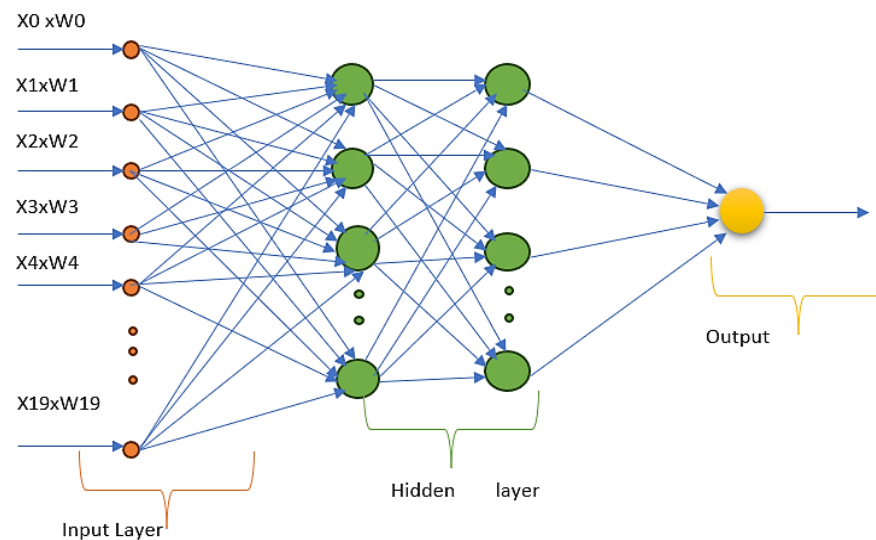


Figure 1. Multilayer feed-forward artificial neural network (MFFANN)

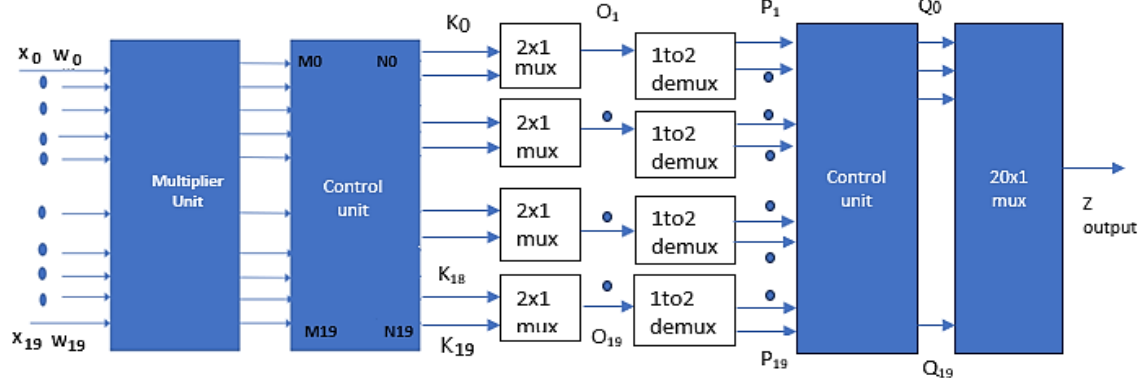


Figure 2. Block diagram representation of MFFANN

Adders and multipliers are important components that are utilized in ANNs to carry out the crucial mathematical operations that are required for processing and learning. This is a comprehensive look at the roles that they play [35]. In an ANN each neuron gets several inputs, each of which has a weight associated with it. The net input to the neuron is formed by adding up all these contributions to the neuron. Each input that is received by a neuron is multiplied by the weight that corresponds to that input. To alter the influence that each input has on the output of the neuron is an extremely important step. This multiplication takes place for every input that the neuron receives. Multipliers are utilized in the process of calculating gradients during the training of ANNs that are trained using backpropagation. The process of multiplying derivatives is performed to calculate the error gradient regarding the weights and inputs. It is essential for hardware

accelerators such as FPGAs and application-specific integrated circuits (ASIC) that are used for neural networks to have adders and multipliers that are implemented properly and efficiently. ANNs can have their speed greatly increased and their power consumption dramatically decreased if they are designed with optimized adders and multipliers, such as carry-lookahead adders and booth multipliers, respectively. The concept of parallelism is utilized in hardware implementations. The inference and training processes can be sped up by designing multipliers and adders to run in parallel [36]. This allows for the simultaneous computation of many neuron outputs, which speeds up the process. The characteristics of a neural network refer to the variables and qualities of the dataset [37]. To enhance the accuracy of your model, it is customary to use a subset of variables. Therefore, the attributes of a neural network are in the input layer, rather than the nodes in the hidden layer. In ANNs, the basic activities of summing weighted inputs and computing gradients are performed by adders and multipliers, which are essential components for conducting these operations [38]. The performance and energy efficiency of neural network hardware accelerators are directly correlated to the efficiency with which they are implemented.

#### 4. RESULTS AND DISCUSSION

The design structure of a multilayer feed-forward neural network hardware chip consists of the multiplier, control logic, multiplexers (2×1), demultiplexers (1×2), and another multiplexer (20×1) components. Figure 3 shows the RTL representation of a multilayer neural network.

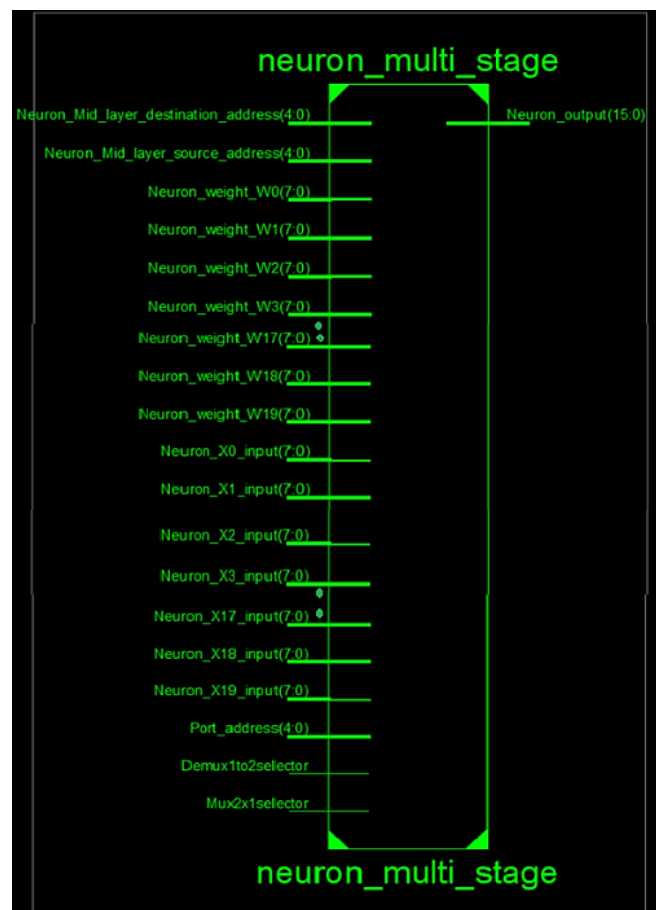


Figure 3. RTL of multilayer ANN

Figure 4 presents the simulation inputs verified in the simulation and Figure 5 presents the simulation waveform outputs of a multilayer neural network simulated in the design. Figure 6 presents the multilayer hardware chip utilization chart, and Figure 7 presents the multilayer hardware chip utilization pie chart. The usage of RTL is to present the complete details of the input and output pins in the design. The simulation input and output pins with stimuli inputs are shown in the waveforms in the form of binary data.

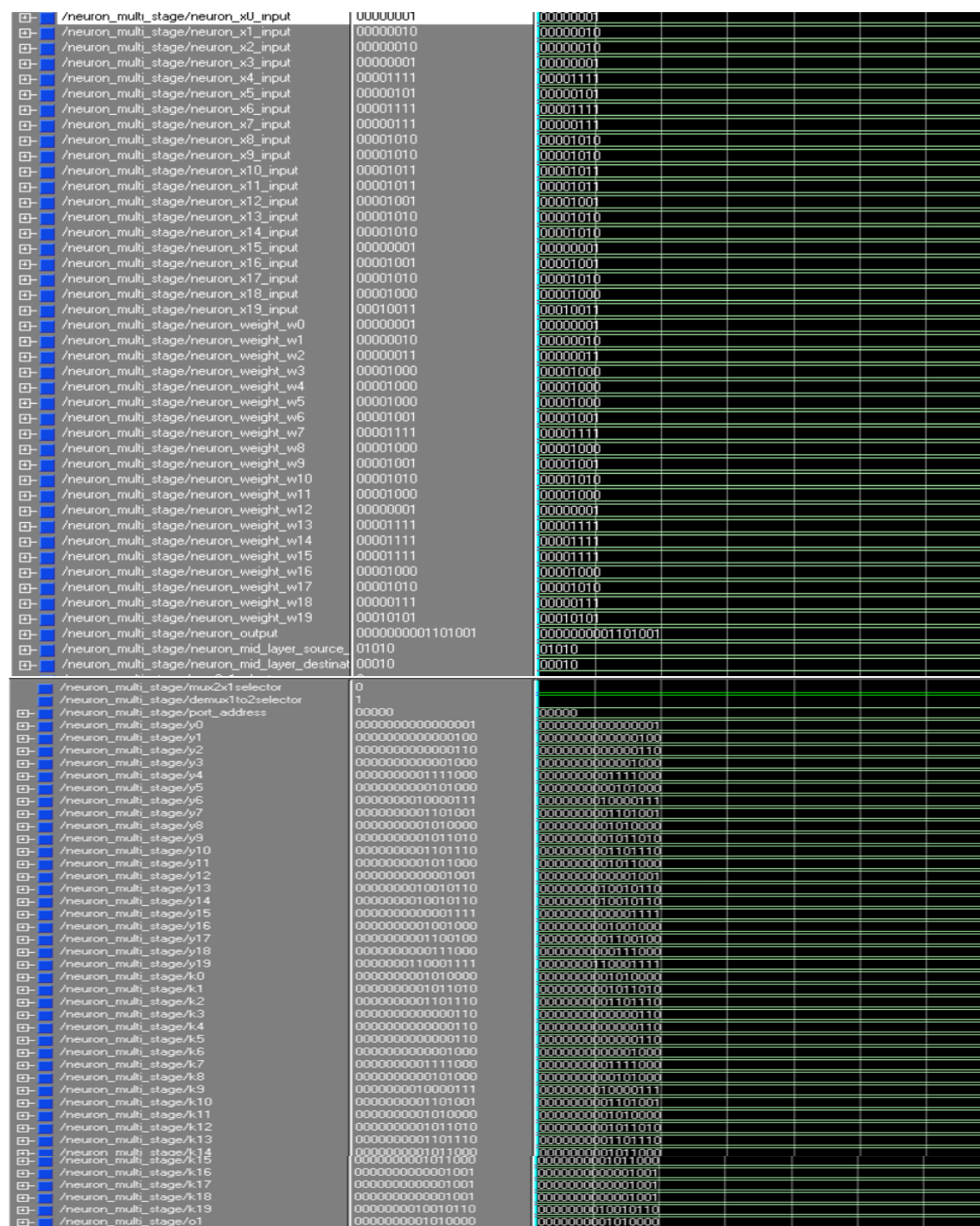


Figure 4. Simulation waveform of inputs for multilayer ANN

Neuron\_X<sub>0</sub>\_input (7:0) presents the 8-bit input of the neuron X<sub>0</sub> and Neuron\_Weight\_W<sub>0</sub>\_input (7:0) presents the corresponding weight input. Neuron\_X<sub>1</sub>\_input (7:0) presents the 8-bit input of the neuron X<sub>1</sub> and Neuron\_Weight\_W<sub>1</sub>\_input (7:0) presents the corresponding weight input. Neuron\_X<sub>2</sub>\_input (7:0) presents the 8-bit input of the neuron X<sub>2</sub> and Neuron\_Weight\_W<sub>2</sub>\_input (7:0) presents the corresponding weight input. Neuron\_X<sub>3</sub>\_input (7:0) presents the 8-bit input of the neuron X<sub>3</sub> and Neuron\_Weight\_W<sub>3</sub>\_input (7:0) presents the corresponding weight input. Neuron\_X<sub>4</sub>\_input (7:0) presents the 8-bit input of the neuron X<sub>4</sub> and Neuron\_Weight\_W<sub>4</sub>\_input (7:0) presents the corresponding weight input. Neuron\_X<sub>5</sub>\_input (7:0) presents the 8-bit input of the neuron X<sub>5</sub> and Neuron\_Weight\_W<sub>5</sub>\_input (7:0) presents the corresponding



weight input. Neuron\_X<sub>6</sub>\_input (7:0) presents the 8-bit input of the neuron X<sub>6</sub> and Neuron\_Weight\_W<sub>6</sub>\_input (7:0) presents the corresponding weight input. Neuron\_X<sub>7</sub>\_input (7:0) presents the 8-bit input of the neuron X<sub>7</sub> and Neuron\_Weight\_W<sub>7</sub>\_input (7:0) presents the corresponding weight input. Neuron\_X<sub>8</sub>\_input (7:0) presents the 8-bit input of the neuron X<sub>8</sub> and Neuron\_Weight\_W<sub>8</sub>\_input (7:0) presents the corresponding weight input. Neuron\_X<sub>9</sub>\_input (7:0) presents the 8-bit input of the neuron X<sub>9</sub> and Neuron\_Weight\_W<sub>9</sub>\_input (7:0) presents the corresponding weight input. Neuron\_X<sub>10</sub>\_input (7:0) presents the 8-bit input of the neuron X<sub>10</sub> and Neuron\_Weight\_W<sub>10</sub>\_input (7:0) presents the corresponding weight input. Neuron\_X<sub>11</sub>\_input (7:0) presents the 8-bit input of the neuron X<sub>11</sub> and Neuron\_Weight\_W<sub>11</sub>\_input (7:0) presents the corresponding weight input. Neuron\_X<sub>12</sub>\_input (7:0) presents the 8-bit input of the neuron X<sub>12</sub> and Neuron\_Weight\_W<sub>12</sub>\_input (7:0) presents the corresponding weight input. Neuron\_X<sub>13</sub>\_input (7:0) presents the 8-bit input of the neuron X<sub>13</sub> and Neuron\_Weight\_W<sub>13</sub>\_input (7:0) presents the corresponding weight input. Neuron\_X<sub>14</sub>\_input (7:0) presents the 8-bit input of the neuron X<sub>14</sub> and Neuron\_Weight\_W<sub>14</sub>\_input (7:0) presents the corresponding weight input. Neuron\_X<sub>15</sub>\_input (7:0) presents the 8-bit input of the neuron X<sub>15</sub> and Neuron\_Weight\_W<sub>15</sub>\_input (7:0) presents the corresponding weight input. Neuron\_X<sub>16</sub>\_input (7:0) presents the 8-bit input of the neuron X<sub>16</sub> and Neuron\_Weight\_W<sub>16</sub>\_input (7:0) presents the corresponding weight input. Neuron\_X<sub>17</sub>\_input (7:0) presents the 8-bit input of the neuron X<sub>17</sub> and Neuron\_Weight\_W<sub>17</sub>\_input (7:0) presents the corresponding weight input. Neuron\_X<sub>18</sub>\_input (7:0) presents the 8-bit input of the neuron X<sub>18</sub> and Neuron\_Weight\_W<sub>18</sub>\_input (7:0) presents the corresponding weight input. Neuron\_X<sub>19</sub>\_input (7:0) presents the 8-bit input of the neuron X<sub>19</sub> and Neuron\_Weight\_W<sub>19</sub>\_input (7:0) presents the corresponding weight input. Neuron\_mid\_layer\_source\_address (4:0) presents the address of the mid-layer blocks to support the design. Neuron\_mid\_layer\_destination\_address (4:0) presents the destination address of the mid layers to support the design. Mux21\_selector is used to select the specific multiplexer of the block in input. Demultiplexer12\_selector is used to select the specific demultiplexer of the block in output. Port\_address (4:0) is the address to support the memory blocks. The complete output of the entire module is taken by neuron\_output (15:0).

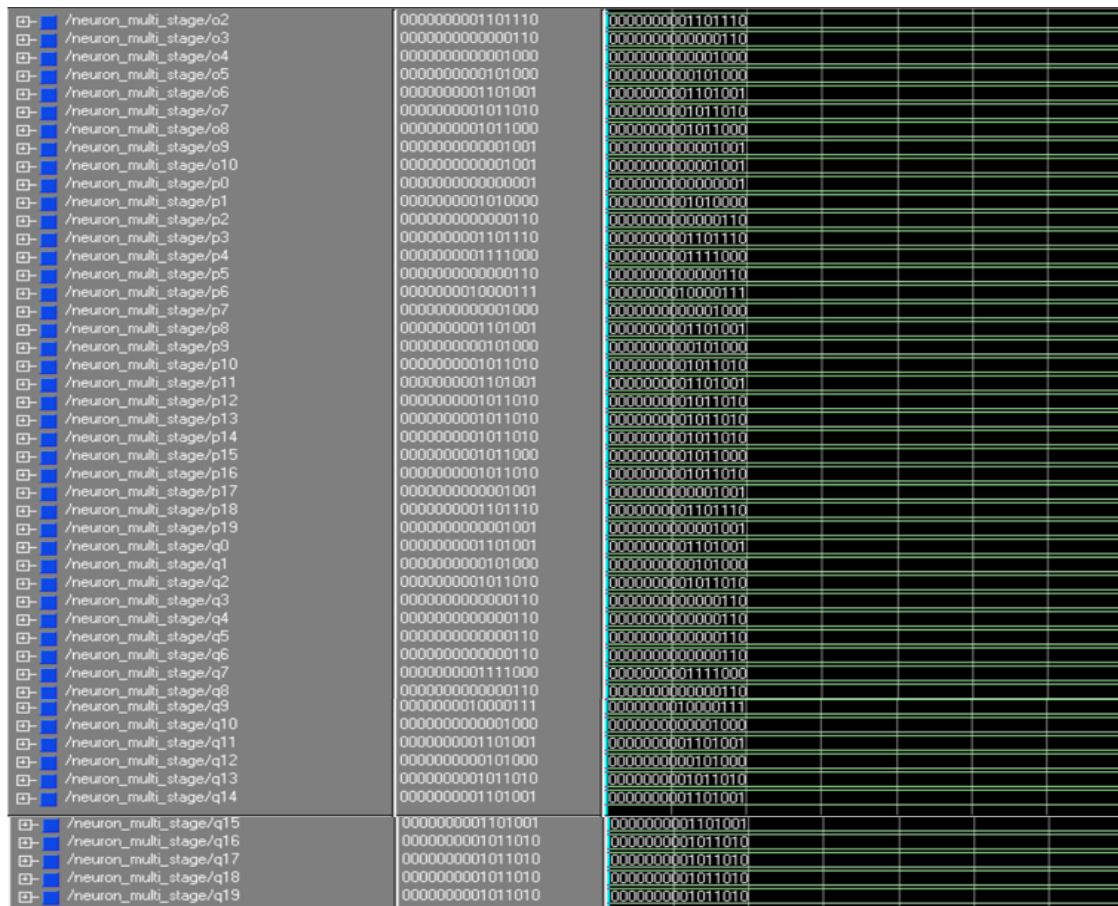


Figure 5. Simulation waveform outputs of multilayer ANN

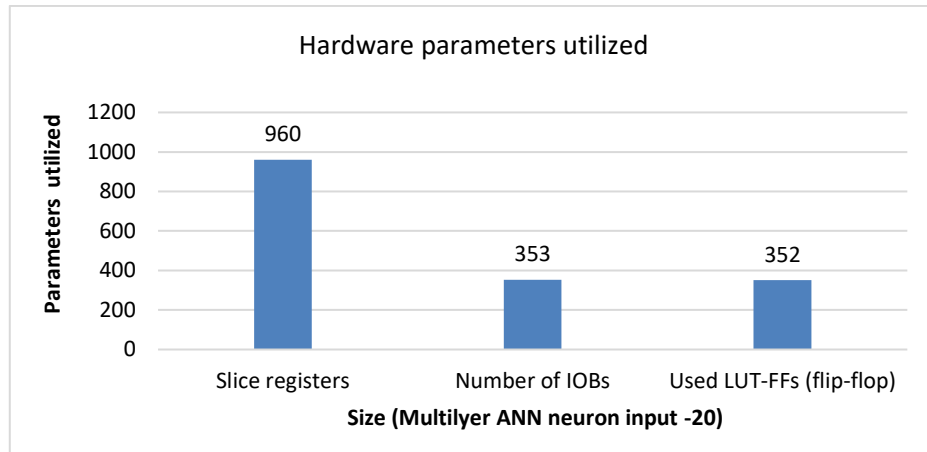


Figure 6. Multilayer hardware chip utilization chart

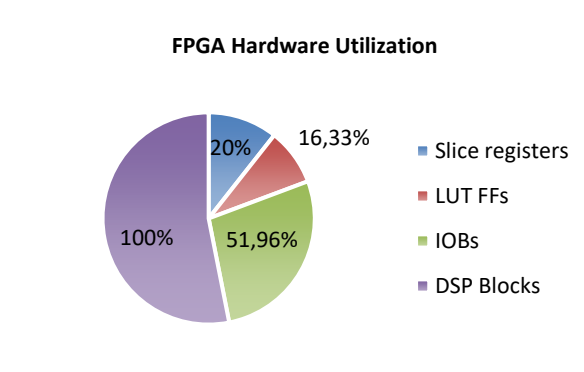


Figure 7. Multilayer hardware chip utilization pie chart

Table 1 lists the hardware parameters utilized by the multilayer hardware chip and the device utilization report for this neuromorphic hardware chip is taken from Xilinx software. The number of slices used is 960 in the multilayer hardware chip, which is out of a total of 4800. Moreover, the number of slices LUTs used is 1547 out of a total of 2400. Input-outputs used are 53 out of a total of 102. The number of DSP elements used is 8 out of a total of 8. The total delay used in multilayer hardware chips is 11.688ns. The memory utilized for the multilayer hardware chip is 4667944 kilobytes. The target device used is spartan-6 FPGA with the device (XC6TS4-2TQG144) for simulation and synthesis. The Spartan-6 family offers the most advanced capabilities for system integration at the lowest possible overall cost for applications that require high volume capacity. It is developed using an established low-power copper process technique that operates at 45 nanometers. DNNs require high memory-to-processing unit data transmission rates. Memory bandwidth can limit large-scale models with many layers and parameters. When trained and run on non-optimal hardware, DNNs can use a lot of power. This affects data mobile, centers, and embedded devices since energy efficiency is crucial. When working with larger models or more complex DNN tasks, scalability requires efficient parallelization across numerous processing units. Due to deep neural networks' complexity and resource requirements, hardware implementation has become harder.

Table 1. Hardware parameters utilized by multilayer hardware chip

Parameter	Value
ANN Chip size	ANN-20
Multiplier used	20
Target device (FPGA)	Spartan-6(xc6stx4-2t9g144)
Used LUTs FFPs	352/2155
Slices	960/4800
Delay (ns)	6.359
IOBs	53/102
Memory (KB)	4665304



## 5. CONCLUSION

The model of neural networks is called a deep neural network when it uses two or more hidden layers. DNN chips based on FPGAs can be optimized for particular neural network models or tasks, which could lead to better performance than generic alternatives. It is distinguished by the utilization of artificial neural networks with various other layers of processing units, sometimes known as neurons or nodes, to learn representations of input. These neural networks may automatically identify patterns or features in data without the need for explicit programming. The multilayer hardware chip for 20-neuron input is designed in Spartan-6 (XC6SLX4-2TQG144) FPGA Xilinx ISE 14.7 successfully. The RTL, internal schematic, and simulation of the multilayer hardware are verified. The hardware parameter utilization reports show that LUT's flip-flop used is 352 out of 2155 which is 16.33 %. The number of bonded IOBs used is 53 out of the available total 102 IOBs, which is 51.96 %. The number of DSP elements used is 8 total out of 8 which is 100.00 %, Number of slice registers used is 960 of the total available 4800 which shows that it is 20.00% utilized. The multilayer ANN design supported 313 MHz frequency, delay 6.359 ns, memory 4665304 KB, and optimal power estimation of 11.45 nW. A new, more efficient dual-register 6-input LUTs logic and a large range of built-in system-level blocks are both included in the Spartan-6 family of integrated circuits, which offers the best mix of cost, power, and performance.

## ACKNOWLEDGEMENTS

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




## REFERENCES

- [1] M. Wielgosz and M. Karwatowski, "Mapping neural networks to FPGA-based IoT devices for ultra-low latency processing," *Sensors*, vol. 19, no. 13, Jul. 2019, doi: 10.3390/s19132981.
- [2] A. Kumar, P. Sharma, M. K. Gupta, and R. Kumar, "Machine learning based resource utilization and pre-estimation for network on chip (NoC) communication," *Wireless Personal Communications*, vol. 102, no. 3, pp. 2211–2231, Oct. 2018, doi: 10.1007/s11277-018-5376-3.
- [3] W. Haensch, T. Gokmen, and R. Puri, "The next generation of deep learning hardware: analog computing," in *Proceedings of the IEEE*, Jan. 2019, vol. 107, no. 1, pp. 108–122, doi: 10.1109/JPROC.2018.2871057.
- [4] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.
- [5] L. Deng *et al.*, "Recent advances in deep learning for speech research at Microsoft," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 8604–8608, doi: 10.1109/ICASSP.2013.6639345.
- [6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.
- [7] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016, doi: 10.1038/nature16961.
- [8] D. Wang, L. Deng, P. Tang, C. Ma, and J. Pei, "FPGA-based neuromorphic computing system with a scalable routing network," in *2015 15th Non-Volatile Memory Technology Symposium (NVMTS)*, Oct. 2015, pp. 1–4, doi: 10.1109/NVMTS.2015.7457432.
- [9] M. A. Dias and D. A. P. Ferreira, "Deep learning in reconfigurable hardware: a survey," in *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, May 2019, pp. 95–98, doi: 10.1109/IPDPSW.2019.00026.
- [10] N. Nedjah, R. M. da Silva, and L. de Macedo Mourelle, "Compact yet efficient hardware implementation of artificial neural networks with customized topology," *Expert Systems with Applications*, vol. 39, no. 10, pp. 9191–9206, Aug. 2012, doi: 10.1016/j.eswa.2012.02.085.
- [11] 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.
- [12] M. Verucchi *et al.*, "A systematic assessment of embedded neural networks for object detection," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2020, pp. 937–944, doi: 10.1109/ETFA46521.2020.9212130.
- [13] T. V. Huynh, "Design space exploration for a single-FPGA handwritten digit recognition system," in *2014 IEEE Fifth International Conference on Communications and Electronics (ICCE)*, Jul. 2014, pp. 291–296, doi: 10.1109/CCE.2014.6916717.
- [14] T. V. Huynh, "Evaluation of artificial neural network architectures for pattern recognition on FPGA," *International Journal of Computing and Digital Systems*, vol. 6, no. 3, pp. 133–138, 2017, doi: 10.12785/IJCD/060305.
- [15] 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.
- [16] A. Owens, P. Isola, J. McDermott, A. Torralba, E. H. Adelson, and W. T. Freeman, "Visually indicated sounds," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 2405–2413, doi: 10.1109/CVPR.2016.264.
- [17] J. Wang, Y. Ma, L. Zhang, R. X. Gao, and D. Wu, "Deep learning for smart manufacturing: methods and applications," *Journal of Manufacturing Systems*, vol. 48, pp. 144–156, Jul. 2018, doi: 10.1016/j.jmsy.2018.01.003.
- [18] 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.
- [19] P. P. Groumpos, "Deep learning vs. wise learning: A critical and challenging overview," *IFAC-PapersOnLine*, vol. 49, no. 29, pp. 180–189, 2016, doi: 10.1016/j.ifacol.2016.11.099.
- [20] B. R. Kiran *et al.*, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, Jun. 2022, doi: 10.1109/TITS.2021.3054625.
- [21] A. Giusti *et al.*, "A machine learning approach to visual perception of forest trails for mobile robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 661–667, Jul. 2016, doi: 10.1109/LRA.2015.2509024.
- [22] R. J. Chen *et al.*, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 10, no. 3, pp. 1–8, 2018.




- [23] B. Khagi, C. G. Lee, and G.-R. Kwon, "Alzheimer's disease classification from brain MRI based on transfer learning from CNN," in *2018 11th Biomedical Engineering International Conference (BMEiCON)*, Nov. 2018, pp. 1–4, doi: 10.1109/BMEiCON.2018.8609974.
- [24] T. Belabed, M. G. F. Coutinho, M. A. C. Fernandes, C. V. Sakuyama, and C. Souani, "User driven FPGA-based design automated framework of deep neural networks for low-power low-cost edge computing," *IEEE Access*, vol. 9, pp. 89162–89180, 2021, doi: 10.1109/ACCESS.2021.3090196.
- [25] S. Hariprasath and T. N. Prabakar, "FPGA implementation of multilayer feed forward neural network architecture using VHDL," *2012 International Conference on Computing, Communication and Applications, ICCCA 2012*, 2012, doi: 10.1109/ICCCA.2012.6179225.
- [26] Y. Zhou, S. Redkar, and X. Huang, "Deep learning binary neural network on an FPGA," in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, Aug. 2017, pp. 281–284, doi: 10.1109/MWSCAS.2017.8052915.
- [27] A. Dundar, J. Jin, V. Gokhale, B. Martini, and E. Culurciello, "Memory access optimized routing scheme for deep networks on a mobile coprocessor," in *2014 IEEE High Performance Extreme Computing Conference (HPEC)*, Sep. 2014, pp. 1–6, doi: 10.1109/HPEC.2014.7040963.
- [28] M. Peemen, A. A. A. Setio, B. Mesman, and H. Corporaal, "Memory-centric accelerator design for Convolutional Neural Networks," in *2013 IEEE 31st International Conference on Computer Design (ICCD)*, Oct. 2013, pp. 13–19, doi: 10.1109/ICCD.2013.6657019.
- [29] Y. Y. Ghadi *et al.*, "Machine learning solutions for the security of wireless sensor networks: a review," *IEEE Access*, vol. 12, pp. 12699–12719, 2024, doi: 10.1109/ACCESS.2024.3355312.
- [30] Z. Chen, J. Wang, H. He, and X. Huang, "A fast deep learning system using GPU," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, Jun. 2014, pp. 1552–1555, doi: 10.1109/ISCAS.2014.6865444.
- [31] 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.
- [32] R. Zhao *et al.*, "Hardware compilation of deep neural networks: an overview," in *2018 IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, Jul. 2018, pp. 1–8, doi: 10.1109/ASAP.2018.8445088.
- [33] 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.
- [34] K. Hwang and W. Sung, "Fixed-point feedforward deep neural network design using weights +1, 0, and -1," in *2014 IEEE Workshop on Signal Processing Systems (SiPS)*, Oct. 2014, pp. 1–6, doi: 10.1109/SiPS.2014.6986082.
- [35] 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.
- [36] 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.
- [37] A. Devrari and A. Kumar, "Turbo encoder and decoder chip design and FPGA device analysis for communication system," *International Journal of Reconfigurable and Embedded Systems (IJRES)*, vol. 12, no. 2, pp. 174–185, Jul. 2023, doi: 10.11591/ijres.v12.i2.pp174-185.
- [38] 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.

## 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 is having an experience of 12 years 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, 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-25 i10-index-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.