ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24JUN1109

Comparison Study and Analysis of Implementing Activation Function of Machine Learning in MATLAB and FPGA

Mallika Roy[1]; Jishnu Nath Paul[2]; Josita Sengupta[3]; Swagata Bhattacharya[4] (Assistant Professor)
[1,2,3]Department of Electronics & Communication Engineering, Guru Nanak Institute of Technology, Kolkata, India
[4] Department of Electronics & Communication Engineering, Guru Nanak Institute of Technology, Kolkata, India

Abstract:- This study examines the implementation and comparative analysis of sigmoid, approximation sigmoid, and hard sigmoid activation functions on FPGA using Verilog HDL and Xilinx ISE simulator and investigates key performance parameters including device usage, clock load, and time characteristics among. The findings suggest that sigmoid functions provide greater accuracy at the expense of larger processors. An approximate sigmoid roughly strikes a balance between accuracy and efficiency, whereas a hard sigmoid is more efficient but imprecise. Comparison of MATLAB results showed the effect of non-stationary computation and lower number, where lower quantization level resulted in improved accuracy. This study highlights the trade-off involved in FPGA-based neural network implementations and fixed-point emphasis. It also suggests future research on reducing the representation and developing effective activation algorithms.

Keywords:- *Activation Function; FPGA; Verilog HDL; Xilinx; MATLAB; Machine Learning.*

I. INTRODUCTION

Implementation of activation functions in field programmable gate arrays (FPGAs) is essential to improve neural network performance and performance. Among many activation functions, sigmoid functions and their derivatives (approximately sigmoid and hard sigmoid) are of particular utility in various neural network applications Each of these functions has specific trade-offs between computational complexity and accuracy, and making their application in hardware an area of great interest.

Processing functions must be implemented in hardware to implement hardware-based machine learning or AI algorithms for edge computing, edge device, or IoT applications These processing functions are important components of machine learning or deep learning each way, which provides the nonlinearity necessary for neural networks to learn Similarly, activation functions representing complex patterns in data are important in machine learning because they enable networks to make complex and binding decisions subtle interactions in the data.

The study of activation functions is important for hardware applications because improving these functions can significantly improve system performance and performance. Hardware efficiency used in activation functions can reduce computing load, power consumption, and processing speed, all important considerations for edge devices with limited resources .Furthermore, understanding and developing these functions to include smart sensors, autonomous systems and real-time data analysis. Extensive research and development in this area is crucial to enhance edge computing technology capabilities though has contributed to the development of effective and responsive AI models capable of delivering timely insights and actions across a variety of applications.

In order to bring neural networks to life in hardware, it is necessary to implement activation functions in FPGAs. Activation functions are important nonlinear features that add complexity and learnability to artificial neural networks. FPGA implementation is important from its ability to bridge the gap between software simulation and real-time application. While software can easily train and simulate neural networks, FPGAs provide the hardware basis for implementing these networks in real-world situations. Engineers can create neural network accelerators that process information faster than a conventional CPU or GPU by adding activation functions to the fabric of the FPGA .This hardware acceleration is important for applications that require real-time feedback, such as graphics recognition, language processing, and autonomous systems.

The approximate sigmoid function is a feasible method that uses simple mathematical approximations to reduce the processing load. This approximation can significantly reduce the resources used on FPGAs and has acceptable accuracy for most real applications. Similarly, the complexity of the sigmoid function, characteristic of its piecewise linear design, provides additional reduction in complexity, allowing faster and less wasteful implementation This study investigates implementation strategies for such implementation these three on FPGAs. It examines design issues, resource allocations, and performance outcomes for each project. By distinguishing between classic sigmoid and approximations and tight variations, this research seeks to highlight the trade-offs and potential benefits in terms of speed, resources, and overall network performance.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24JUN1109

II. PRELIMINARIES OF ACTIVATION FUNCTION

Diagrams showing key components and connections can be used to visualize the structure of artificial tissues. The typical structure of the artificial neuron diagram is illustrated below and categorized:

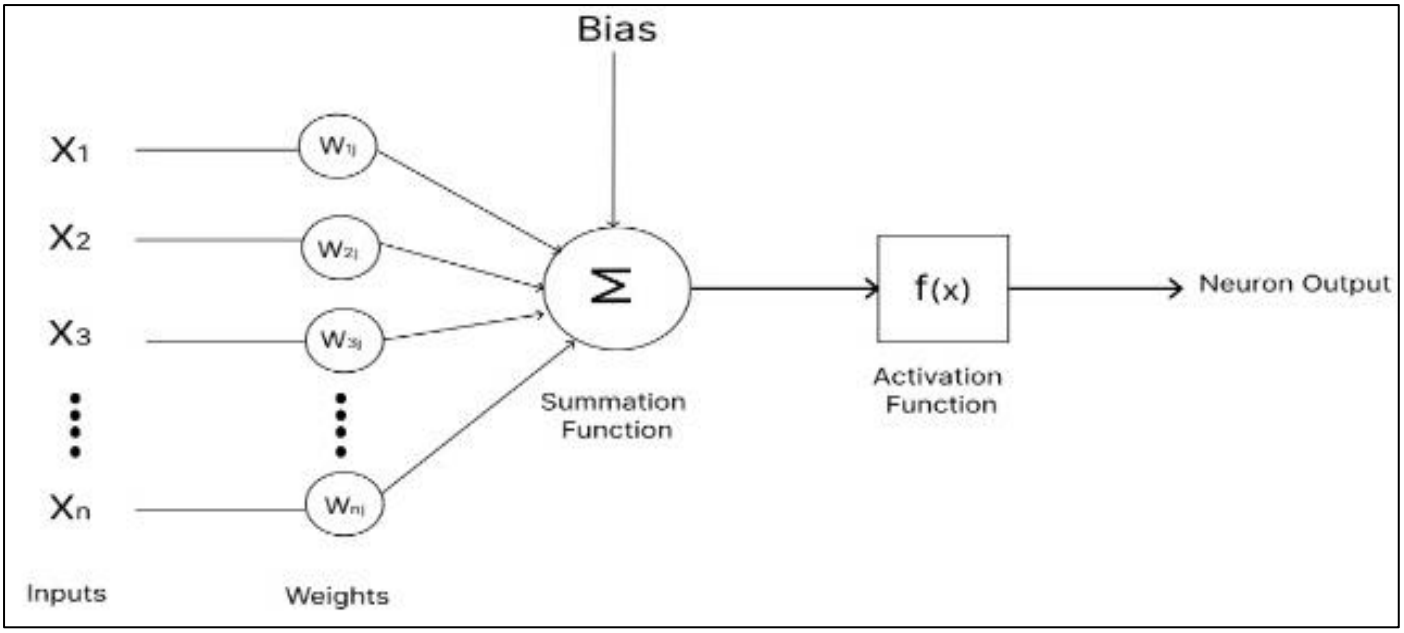

Fig 1 Artificial Neuron Diagram

The artificial neuron received several inputs (X1, X2, X3, ….Xn), each representing an attribute or property of the processed data, weights are assigned, which are shown as W1j, W2j, W3j,…Wnj, indicating the relative importance of the resulting neurons. The weights are learned in the training phase of the neural network function, often abbreviated ∑.

$$S=\sum_{i=1}^{n}(Wij.Xi) \quad (1)$$

Where S represents the combination of outcomes, Wij is the weight and Xi is the input.

The term 'bias' is added to the weighted sum, allowing the neuron to adjust its output independently of the inputs and can help in shifting the activation function.

$$Z= S + bias = \sum_{i=1}^{n}(Wij.Xi) + bias \quad (2)$$

The weighted total plus bias is then transmitted through the activation function denoted by f. Activation functions add nonlinearity to the output of neurons, allowing them to detect complex structures in the input and make nonlinear decisions.

$$Y=f(Z) \quad (3)$$

Where Y denotes the final output of the root. The output of the activation function represents the final output of the artificial neuron, which can be propagated to other neurons in the network.

This neural network approach was implemented using FPGAs. The FPGA-based neuron part is shown in the figure below.

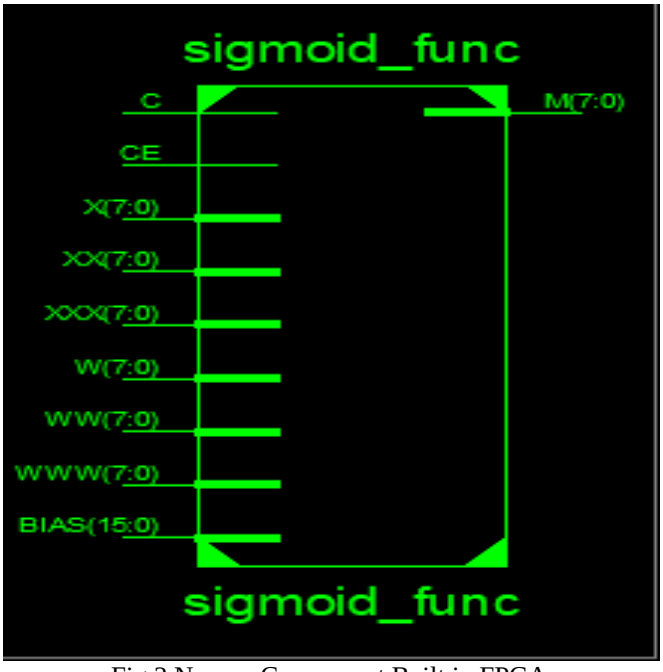

Fig 2 Neuron Component Built in FPGA

The input signals are X[7:0], XX[7:0], and XXX[7:0], while the weights of neurons are W[7:0], WW[7:0], and WWW [. 7:0], each of 8 bits. BIAS[15:0] is the 16-bit bias of the neuron, C is the input clock of the system, CE is the chip enabler of the system and M[7:0] is the 8-bit output of the neuron.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24JUN1109

This section contains three basic functions: the sigmoid function, the approximate sigmoid function, and the hard sigmoid function.

➢ *Sigmoid Function*

Sigmoid activation functions are commonly utilized in neurons, particularly for binary classification issues. It maps any real-valued number to a location (0,1). It is defined as-

$$F1(x) = \frac{1}{1+e^{-x}} \quad (4)$$

The sigmoid function is defined as a unique S-shaped curve.

➢ *Approximate Sigmoid*

An approximate sigmoid function attempts to reduce computational complexity while maintaining a close approximation to the normal sigmoid function.

$$F2(x) = \frac{1}{2}\left[\frac{x}{1+|x|} + 1\right] \quad (5)$$

➢ *Hard Sigmoid Function*

The hard sigmoid function is computationally efficient with binary-like results. It can be defined as a piecewise linear function:

$$F3(x) = \begin{cases} 0 & if\ x \leq -2.5 \\ 1 & if\ x \geq 2.5 \\ 0.2x + 0.5 & otherwise \end{cases} \quad (6)$$

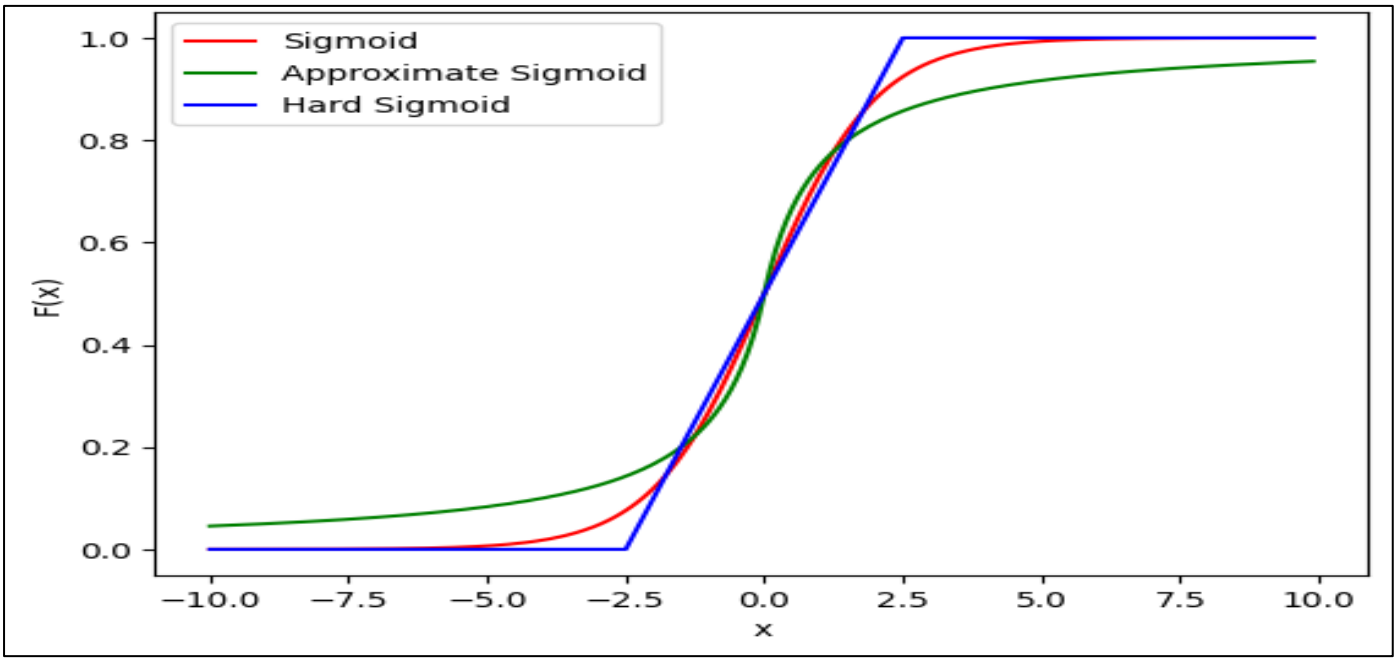

Fig 3 Comparison of Sigmoid, Approximate Sigmoid and Hard Sigmoid Functions

III. SIMULATION AND RESULTS

All of the Artificial Neuron blocks, including the Sigmoid Function, Approximate Sigmoid, and Hard Sigmoid, are designed in Verilog HDL and simulated with the Xilinx ISE simulator version 14.5. Figures 4, 5, and 6 demonstrate the device use for the sigmoid function, approximate sigmoid, and hard sigmoid, respectively.

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	22	4656	0%
Number of Slice Flip Flops	21	9312	0%
Number of 4 input LUTs	35	9312	0%
Number of bonded IOBs	123	158	77%
Number of MULT18X18SIOs	3	20	15%
Number of GCLKs	1	24	4%

Fig 4 Device Utilization for Sigmoid in FPGA

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24JUN1109

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	45	4656	0%
Number of Slice Flip Flops	15	9312	0%
Number of 4 input LUTs	88	9312	0%
Number of bonded IOBs	74	158	46%
Number of MULT18X18SIOs	3	20	15%
Number of GCLKs	1	24	4%

Fig 5 Device Utilization for Approximate Sigmoid in FPGA

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	39	4656	0%
Number of Slice Flip Flops	32	9312	0%
Number of 4 input LUTs	75	9312	0%
Number of bonded IOBs	116	158	73%
Number of MULT18X18SIOs	3	20	15%
Number of GCLKs	1	24	4%

Fig 6 Device Utilization for Hard Sigmoid in FPGA

Figures 7, 8, and 9 illustrate simulation waveforms. Figure 7 depicts the simulated waveform of the Sigmoid function, which is the original sigmoid function. Input X[7:0] is defined as 01100110, input XX[7:0] is specified as 00011001, and input XXX[7:0] is defined as 01001100, with the observed output being M[7:0] = 01011000.

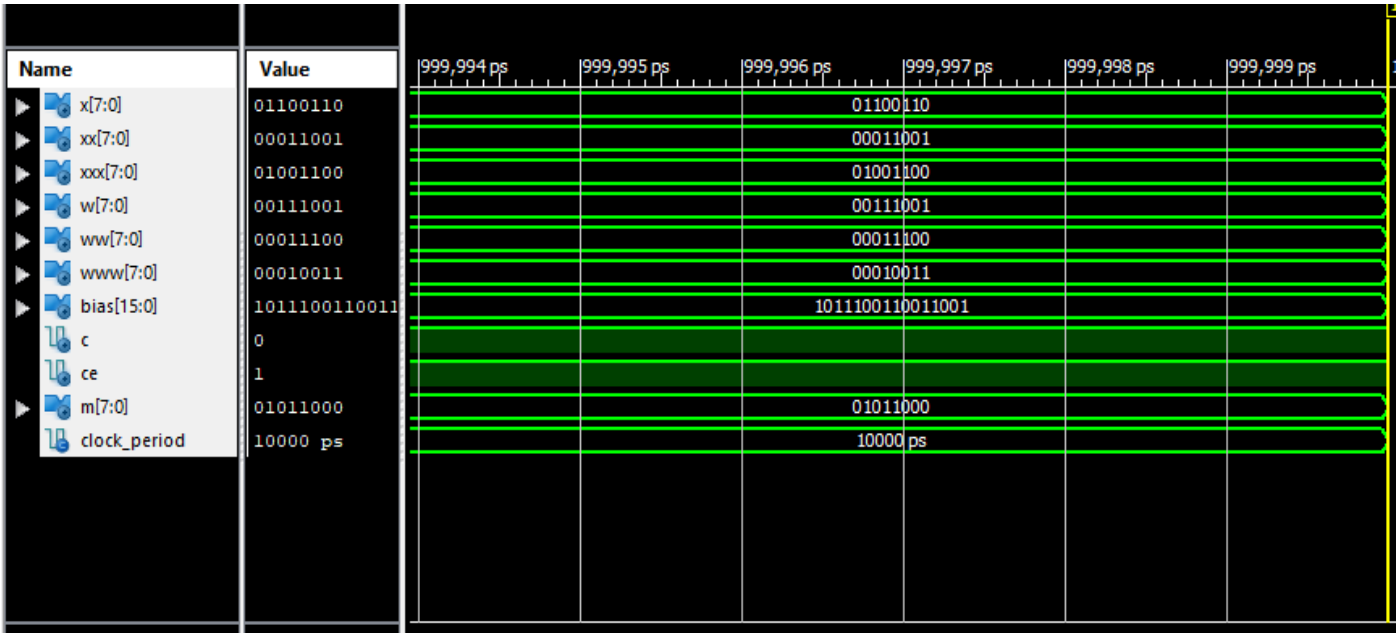

Fig 7 Simulation Waveform for Sigmoid Function in FPGA

Figure 8 depicts the simulation waveform for an approximate sigmoid function. The output M[7:0] is shown as 01011111, which looks to be significantly greater than the initial sigmoid result. While both outputs follow similar trends and respond to the same inputs, the approximate sigmoid output produces slightly greater activation levels. This disparity could be explained by the approximation utilized to calculate the sigmoid function in the approximate sigmoid implementation.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24JUN1109

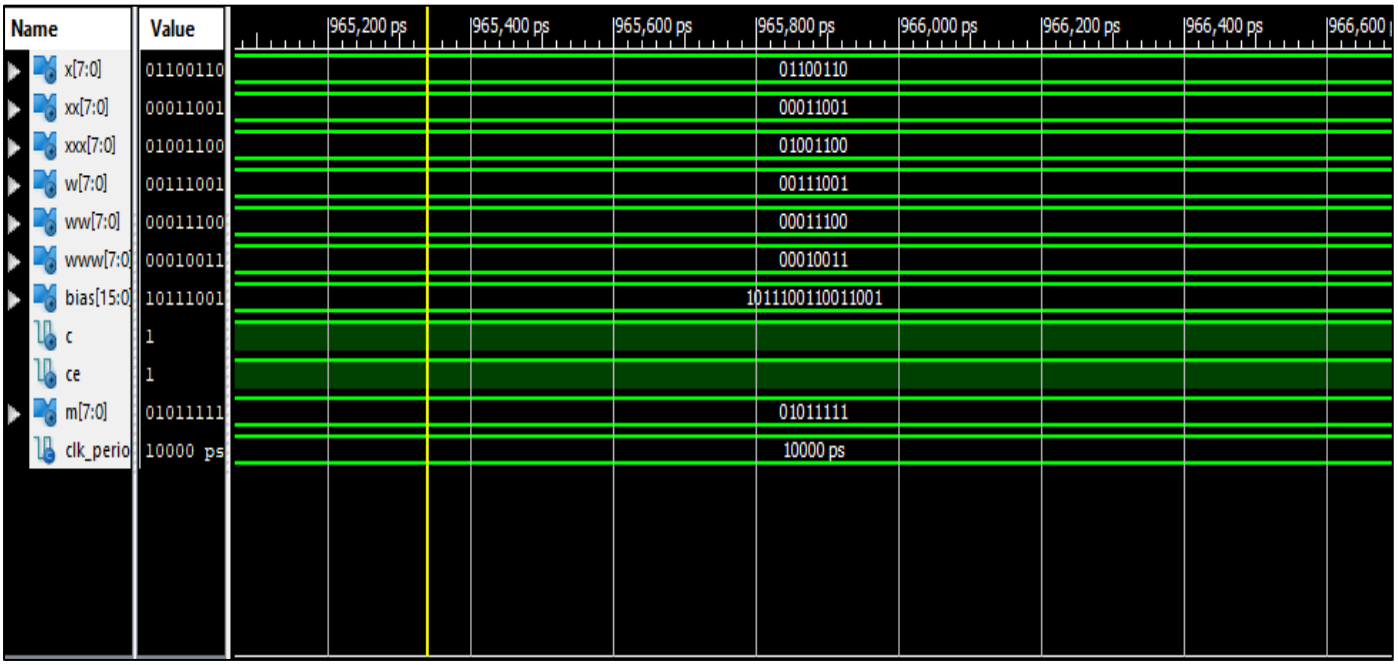

Fig 8 Simulation Waveform for Approximate Sigmoid Function in FPGA

Figure 9 depicts the simulated waveform of a hard sigmoid function, with the output as result[7:0] = 00000000. The hard sigmoid function is a computationally efficient approximation of the sigmoid function that trades off accuracy for speed. It produces binary outputs (0 and 1) which can be useful for applications where a clear decision boundary is required.

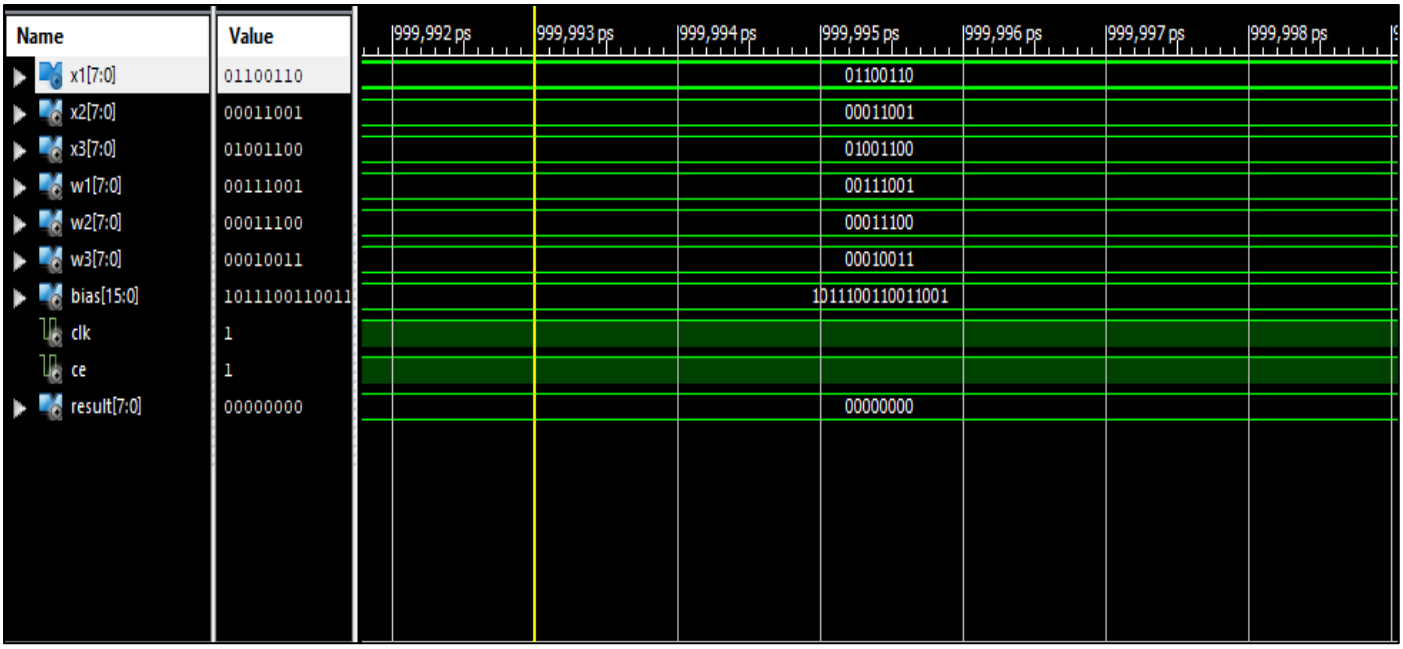

Fig 9 Simulation Waveform for Hard Sigmoid in FPGA

The following table shows the neurons' output using the sigmoid, approximation sigmoid, and hard sigmoid functions achieved with the FPGA technique.

Table 1 Results of Sigmoid, Approximation Sigmoid and Hard Sigmoid

X1	X2	X3	Sigmoid	Approximate Sigmoid	Hard Sigmoid
0.8	0.2	0.6	0.3	0.4	0
0.3	-0.7	-0.2	0.6	0.7	1
0.4	0.9	0.1	0.6	0.7	0.7
1.6	1.9	0.1	0.8	0.8	1
0.6	0.9	0.4	0.4	0.3	0.2

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24JUN1109

All of the blocks are implemented on FPGA in accordance with their respective requirements.

Family – Spartan3E

Device –XC3ES500E

Package – PQ208

Speed – -5

IV. PERFORMANCE AND TIMING ANALYSIS OF SIGMOID FUNCTION IMPLEMENTATIONS

This section compares the performance and timing properties of the sigmoid, approximation sigmoid, and hard sigmoid function implementations. The following aspects are analyzed.

➢ *Clock Load Comparison:*

This comparison compares the number of clock loads required by each function, illustrating variances in resource use.

➢ *Maximum Frequency Comparison:*

This analysis compares the maximum operating frequency achieved by each function, providing information on the speed and efficiency of the implementations.

➢ *Timing Summary Comparison:*

This comparison contains three key timing metrics:

- Minimum period required for each function.
- Minimum input arrival time before clock.
- Maximum output time after clock.

➢ *The Graphs below Show these Comparisons:*

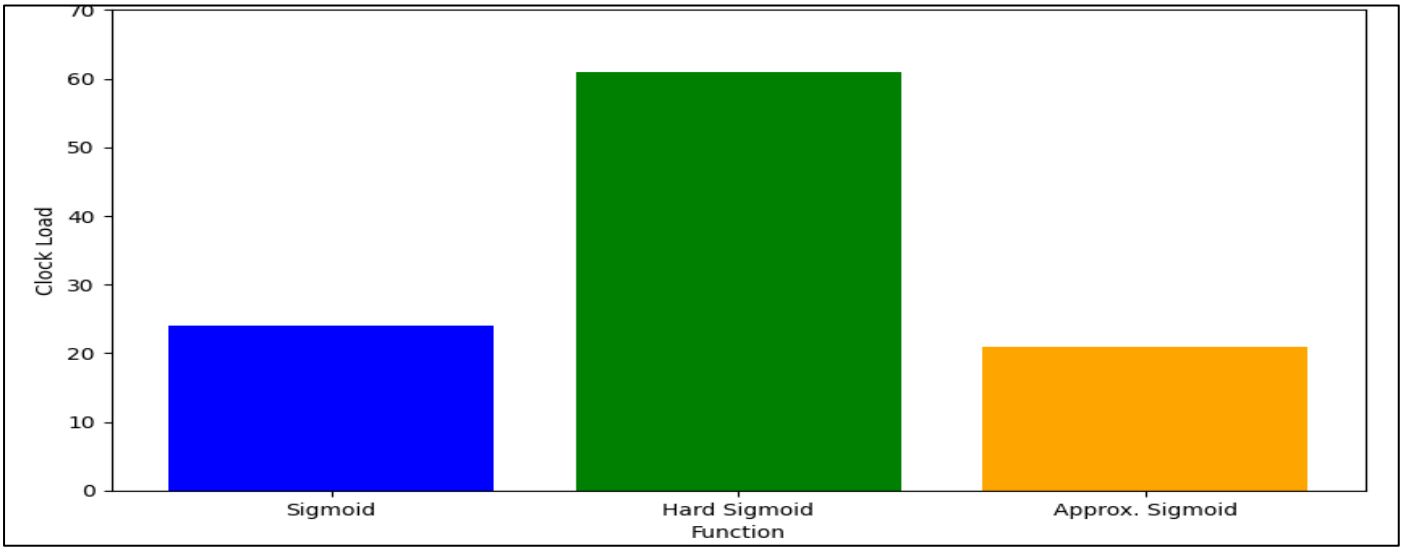

Fig 10 Clock Load Comparison

The BUFGP buffer is used to load the sigmoid function at a clock rate of 24. The hard sigmoid function uses the BUFGP buffer, which has a higher clock load of 61. The approximation sigmoid function uses the BUFGP buffer with a clock load of 21. This means that the hard sigmoid function consumes the most clock resources, while the approximate sigmoid uses the fewest.

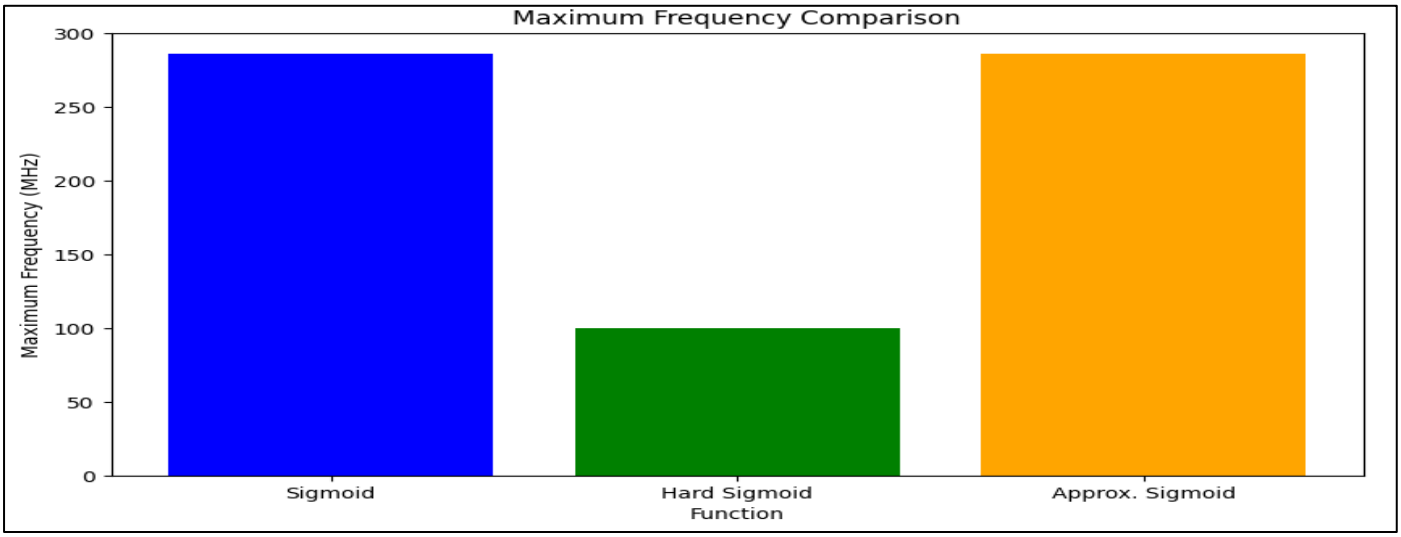

Fig 11 Maximum Frequency Comparison

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24JUN1109

The minimum period for both the sigmoid and approximation sigmoid functions is 3.492ns, which corresponds to a maximum frequency of 286.369MHz.

The hard sigmoid function has a slightly longer minimum period of 9.952ns, resulting in a maximum frequency of 100.478MHz.

This indicates that the hard sigmoid function runs at a slower rate than the sigmoid and approximate sigmoid functions.

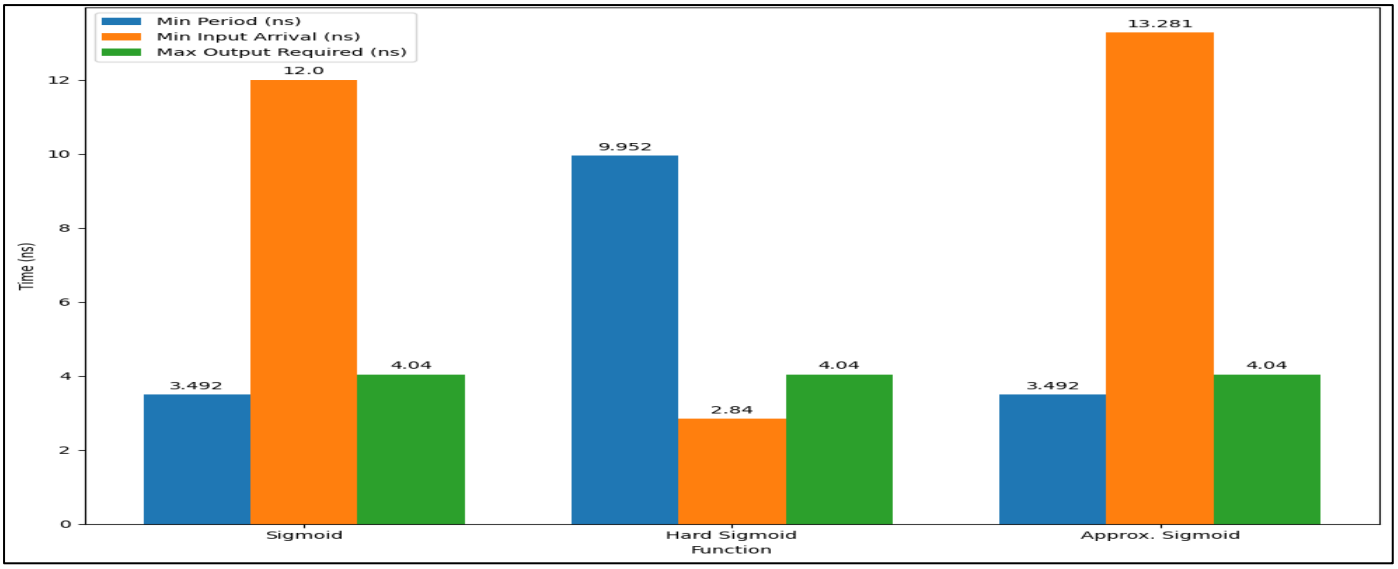

Fig 12 Timing Summary Comparison

The sigmoid function has a minimum input arrival time of 12.000ns before the clock and a maximum output time of 4.040ns after the clock.

The lowest input arrival time for the hard sigmoid function is 2.840ns, and the maximum output necessary time after the clock is also 4.040ns.

The approximate sigmoid function has a minimum input arrival time of 13.281ns before the clock and a maximum output time of 4.040ns after the clock.

The hard sigmoid function has a faster input arrival time, which means it can process inputs faster than the other two functions.

V. COMPARATIVE ANALYSIS OF SIGMOID FUNCTION IMPLEMENTATION IN MATLAB AND FPGA

Experiments were carried out utilizing the sigmoid functions on input data sets in both MATLAB and Xylinx platforms. The MATLAB implementation relied on built-in functions, whereas the Xylinx implementation used fixed-point arithmetic with quantization levels of Q8, Q14, and Q16.

The following table compares the output of neurons calculated using the MATLAB program to those generated using the FPGA technique.

Table 2 Results of MATLAB and FPGA with Different Quantization Levels

Output of neuron in MATLAB	Output of neuron in FPGA(Q8)	Output of neuron in FPGA(Q14)	Output of neuron in FPGA(Q16)
0.31	0.3	0.4	0.6
0.98	0.6	0.4	0.7
0.93	0.6	0.5	0.5
0.89	0.8	0.7	0.5
0.05	0.4	0.1	0.01

All inputs and outputs from the neuron created in FPGA were recorded in binary format. The values in the table were displayed in decimal form to make them easier to understand. The table above demonstrates that the outputs of neurons estimated in MATLAB differ from those generated by FPGA implementations with varied quantization levels (Q8, Q14, and Q16). These variations are primarily due to the style of representation. In MATLAB, the sigmoid function is calculated using built-in techniques that employ floating-point arithmetic for high precision and accuracy. On the other hand, the FPGA implementation uses fixed-point arithmetic with quantization levels, with inputs and intermediate values recorded in binary format.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24JUN1109

This fixed-point encoding contains quantization errors, particularly at lower quantization levels like Q8, which result in slight discrepancies from the MATLAB findings. Figures 13, 14 and 15 illustrate a comparison between MATLAB and FPGA at different quantization levels, as well as their errors.

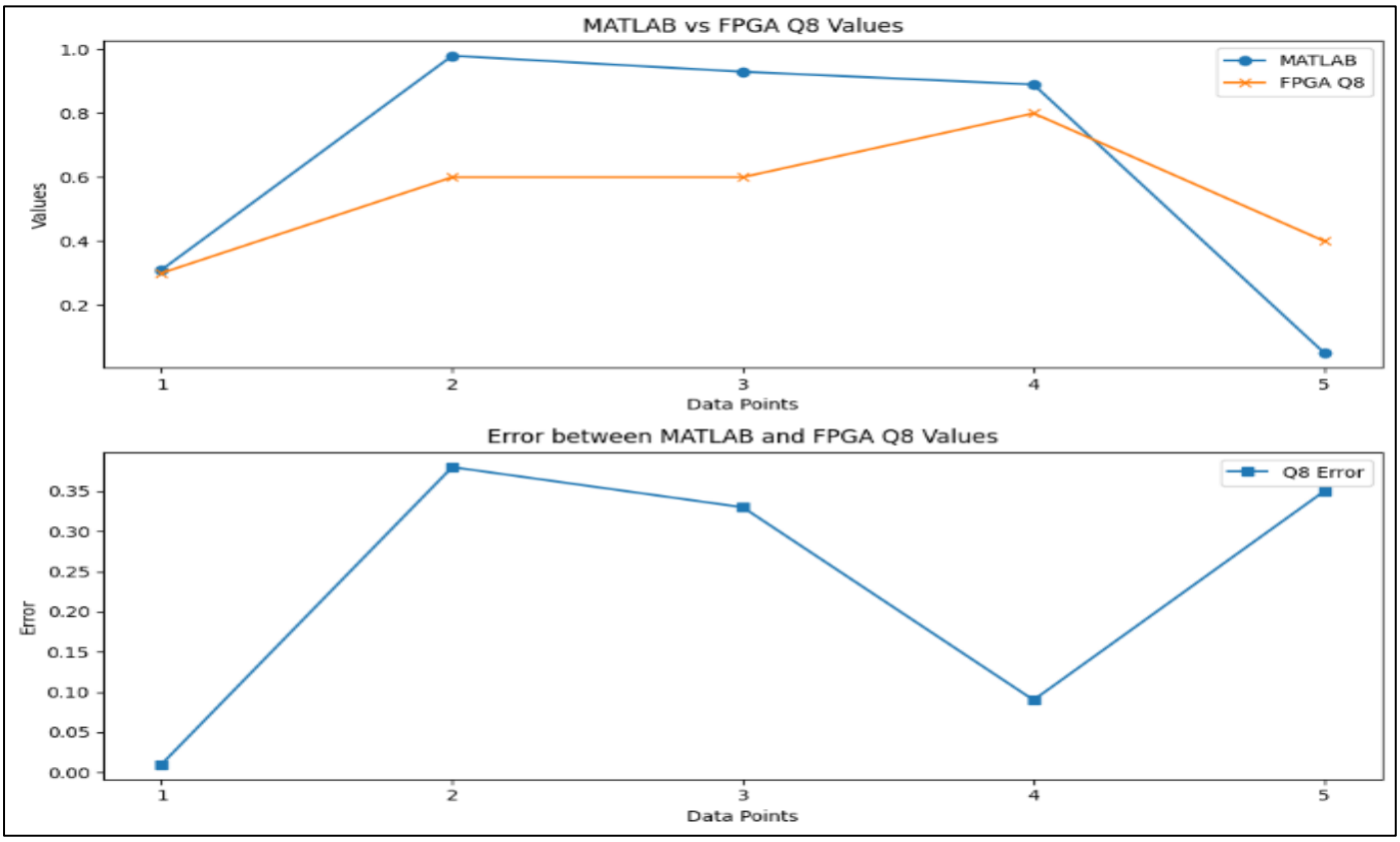

Fig 13 Graph showing Comparison of MATLAB and FPGA (Q8) Sigmoid Function Outputs for given Data Points, along with the Corresponding Error between them

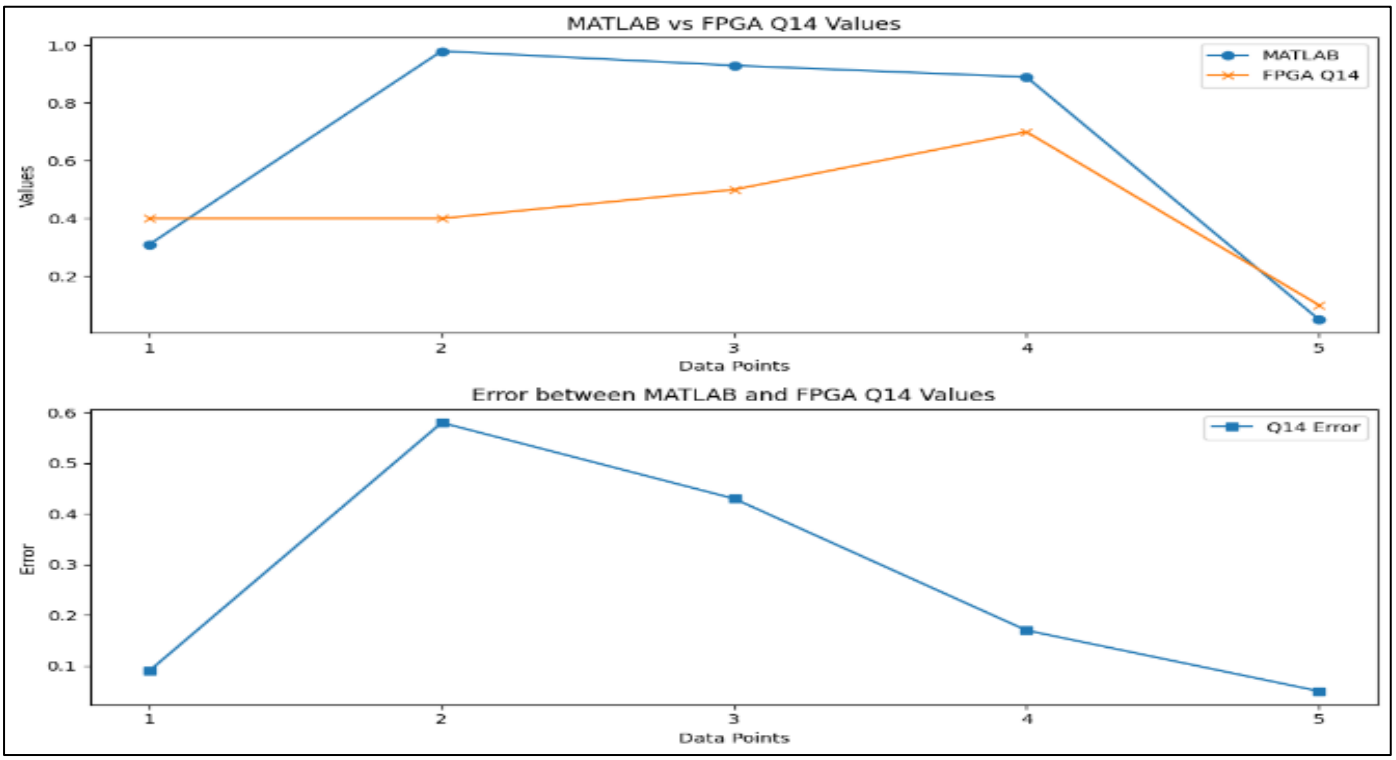

Fig 14 Graph showing Comparison of MATLAB and FPGA (Q8) Sigmoid Function Outputs for given Data Points, along with the Corresponding Error between them

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24JUN1109

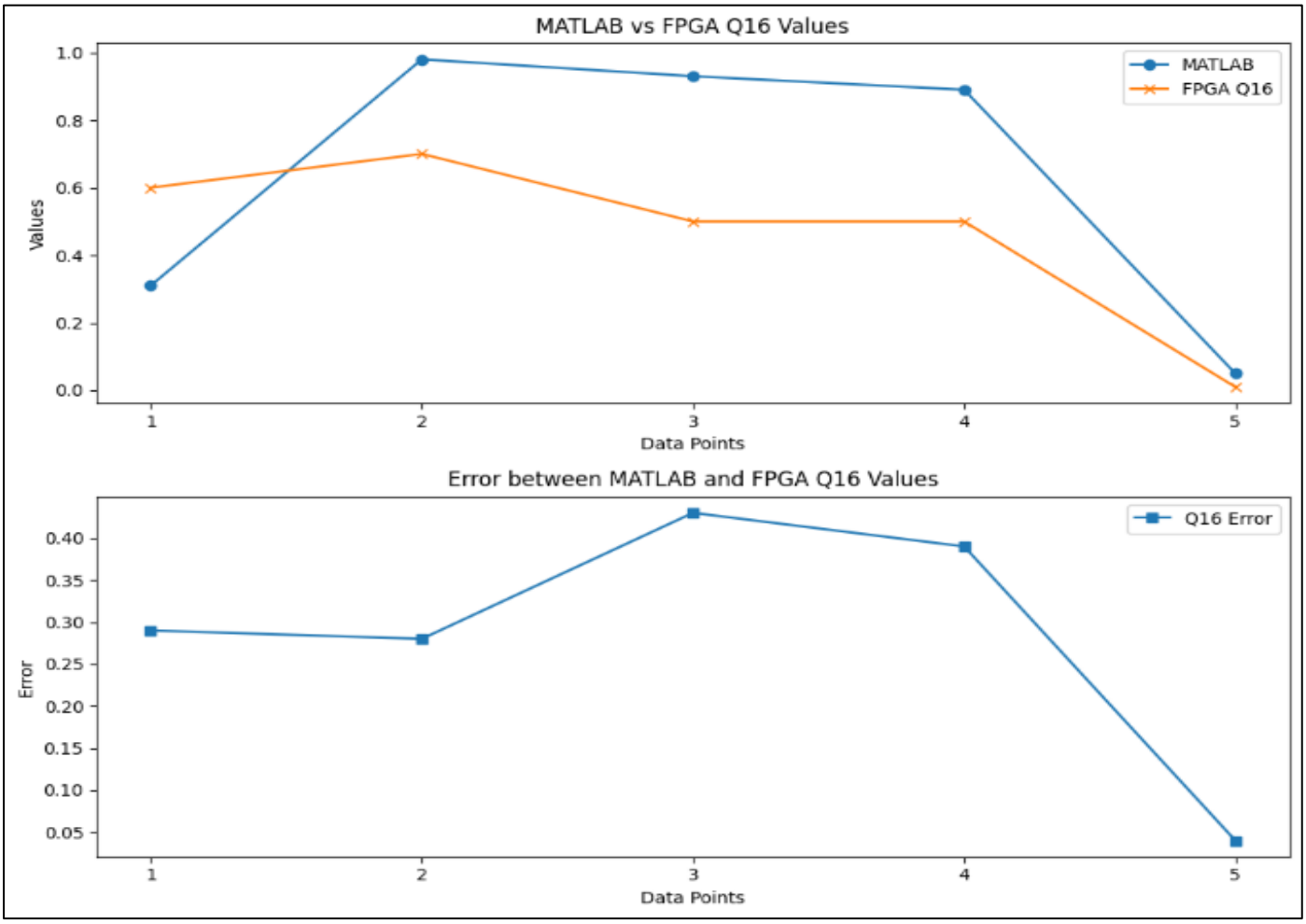

Fig 15 Graph showing Comparison of MATLAB and FPGA (Q8) Sigmoid Function Outputs for given Data Points, along with the Corresponding Error between them

The following table shows the mean square error (MSE) between the outputs of the neuron in MATLAB and the FPGA outputs with different quantization levels (Q8, Q14, and Q16) has been calculated.

Table 3 Mean Square Error between MATLAB and FPGA

Mean Square Error (Q8)	Mean Square Error (Q14)	Mean Square Error (Q16)
0.0768	0.1136	0.10022

The MSE for the Q8 quantization level is the lowest, at 0.0768, as indicated in the table, showing that the Q8 quantization level yields FPGA outputs that are most similar to MATLAB outputs. The MSE for the Q14 quantization level is the greatest, at 0.1136, indicating that it causes the most substantial mistake as compared to the MATLAB outputs. The MSE for the Q16 quantization level is 0.10022, which is greater than Q8 but lower than Q14, indicating a moderate margin of error. Lower quantization levels, such as Q8, produce more accurate outputs but may also need more FPGA resources.

VI. CONCLUSION

This report presented the implementation and comparative analysis of the sigmoid, approximate sigmoid, and hard sigmoid activation functions on an FPGA. The results demonstrated that while the original sigmoid function offers high accuracy, the approximate sigmoid provides a good balance between accuracy and computational efficiency. The hard sigmoid, on the other hand, is highly efficient in terms of computation but at the cost of reduced precision. Comparing the FPGA results with MATLAB implementations highlighted the impact of fixed-point arithmetic and quantization errors. The discrepancies observed were more pronounced at lower quantization levels. Future work could focus on optimizing the fixed-point representation and exploring other hardware-efficient activation functions.

➢ *The Key Findings from the Performance and Timing Analysis are:*

- *Clock Load:*

The hard sigmoid function utilizes the most clock resources, while the approximate sigmoid requires the fewest.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24JUN1109

- *Maximum Frequency:*

The sigmoid and approximation sigmoid functions can work at higher frequencies than the hard sigmoid function.

- *Timing Metrics:*

The hard sigmoid function processes inputs quickly but less accurately.

In addition to the performance and timing analysis, a Mean Square Error (MSE) analysis was performed between the MATLAB outputs and FPGA outputs for various quantization levels, revealing that the implementation with Q8 quantization level is the most accurate when compared to MATLAB, while Q14 introduces the most error. In conclusion, the MSE analysis highlights the trade-off between quantization levels and accuracy, which guides the selection of an optimal fixed-point representation for balancing precision and resource utilization in FPGA-based designs.

REFERENCES

[1]. Bañuelos-Saucedo, M A, et al. "Implementation of a Neuron Model Using FPGAS." *Journal of Applied Research and Technology*, vol. 1, no. 03, 1 Oct. 2003, https://doi.org/10.22201/icat.16656423.2003.1.03.611. Accessed 25 Aug. 2023.

[2]. Beiu, Valeriu. *Closse Approximations of Sigmoid Functions by Sum of Step for VLSI Implementation of Neural Networks*. 2014.

[3]. Deng, Li. "A Tutorial Survey of Architectures, Algorithms, and Applications for Deep Learning." *APSIPA Transactions on Signal and Information Processing*, vol. 3, 2014, www.cambridge.org/core/journals/apsipa-transactions-on-signal-and-informationa-processing/article/tutorial-survey-of-architectures-algorithms-and-applications-for-deep-learning/023B6ADF962FA37F8EC684B209E3DFAE, https://doi.org/10.1017/atsip.2013.9. Accessed 15 Aug. 2019.

[4]. Dubey, Shiv Ram, et al. "Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark." *Neurocomputing*, vol. 503, Sept. 2022, pp. 92–108, https://doi.org/10.1016/j.neucom.2022.06.111. Accessed 28 May 2024.

[5]. Feng, Jianli, and Shengnan Lu. "Performance Analysis of Various Activation Functions in Artificial Neural Networks." *Journal of Physics: Conference Series*, vol. 1237, June 2019, p. 022030, https://doi.org/10.1088/1742-6596/1237/2/022030.

[6]. Gustineli, Murilo. "A Survey on Recently Proposed Activation Functions for Deep Learning." *ArXiv.org*, 6 Apr. 2022, arxiv.org/abs/2204.02921. Accessed 2 July 2023.

[7]. Kwan, H.K. "Simple Sigmoid-like Activation Function Suitable for Digital Hardware Implementation." *Electronics Letters*, vol. 28, no. 15, 1992, p. 1379, https://doi.org/10.1049/el:19920877.

[8]. Muhammed, Thamer, et al. *IMPLEMENTATION of a SIGMOID ACTIVATION FUNCTION for NEURAL NETWORK USING FPGA IMPLEMENTATION of a SIGMOID ACTIVATION FUNCTION for NEURAL NETWORK USING FPGA*. 2012.

[9]. Ngah, Syahrulanuar, and Rohani Abu Bakar. *Sigmoid Function Implementation Using the Unequal Segmentation of Differential Lookup Table and Second Order Nonlinear Function*.

[10]. Reza Raeisi, and Armin Kabir. *IMPLEMENTATION of ARTIFICIAL NEURAL NETWORK on FPGA*. 1 Jan. 2006. Accessed 3 June 2024.