Engineering and Technology Journal
Journal homepage: https://etj.uotechnology.edu.iq

The Proposition of Three Approaching Ways to Implement Tan-sigmoid Activation Function in FPGA

Manal T. Ail*, **Bassam H. Abed**

Electrical Engineering Dept., University of Technology-Iraq, Alsina'a street, 10066 Baghdad, Iraq.
*Corresponding author Email: eee.19.36@grad.uotechnology.edu.iq

HIGHLIGHTS

- The challenge is building a tan-sigmoid function in hardware with efficient performance.
- Proposed three approaches: tan-sigmoid using the log approximation, the segmentation, and the polynomial methods.
- These approaches are efficiently implemented in the Xilinx Spartan-3A xc3s700a-4fg484 platform.
- The resource utilization is about (1%) for slices and LUT.

ARTICLE INFO

Handling editor: Ivan A. Hashim
Keywords:
Neural Network
Field programmable gate array (FPGA)
Activation function
Tan- sigmoid

ABSTRACT

Hyperbolic tangents and Sigmoid are commonly used for Artificial Neural Networks as activation functions. The complex equation of the activation function is one of the most difficult to be implemented in hardware because containing division and exponential, which gives non-linear behavior. The challenge is building a tan-sigmoid function in hardware with efficient performance. Therefore, this work will focus on implementing the activation function in FPGA. To overcome this challenge, a different approach was proposed in this paper, efficient hardware-implemented for tan-sigmoid in terms of the number of slices occupied and the resources utilization are designed. In this work, three approaches are proposed: tan-sigmoid using the log approximation method, tan-sigmoid using segmentation method, and tan-sigmoid using the polynomial method. These approaches are efficiently implemented in the Xilinx Spartan-3A xc3s700a-4fg484 platform. Hardware synthesis and FPGA implementations illustrate that the proposed tan-sigmoid only takes up to 1% of logic resources in the first and second proposed approaches. While, 4% showed in the third proposed approach, with the best efficiency and significantly confirmed the lowest implementation costs than the traditional approach.

1. Introduction

Neurons in the human brain have side connections with neighbors to know behavior patterns for the neural network [1]. Artificial Neural Networks (ANNs) are influenced by neural biological network architecture. The online training of ANN is similar to the neuroplasticity function, which is represented as the capacity of the brain to reorganize the neural network through the reform of neuronal ties. As explained, the biological neural network is educated and developed in our minds by new concepts [2]. The goal of building intelligent systems, in addition to the developments in high-speed computing, has shown that ANN is capable of mapping, modeling, and classifying non-linear systems [3,4].

In several fields, artificial neural networks were commonly used. Until now, most work performed in this field involved software simulations, the investigation of capabilities of ANN models, or new algorithms. ANNs can solve various problems in pattern recognition, signal processing, control systems, etc. But hardware implementation is also essential for applicability and neural network [5, 6].

ANNs typically use complex applications for training periods and different training passes in the training operation [7]. Due to the high latency of host processor preparation, software implementation of the ANNs for those applications is ineffective. Training operation can be accelerated by developing customized hardware, parallel, pipeline, and versatile bit-width data trajectories [8]. Analog or digital systems can be used to implement neural networks. Although the analog systems are precise, they are hard to use and have weight storage issues. Digital implementation is more common because they have greater precision, better repeatability, lower noise sensitivity, better testing, greater durability, and compatibility with other types of preprocessors [5,6]. The digital implementations hardware in NN is classification as digital signal processor (DSP)-based, Application Specific Integrated Chip (ASIC)-based, and Field programmable gate array (FPGA)-based implementations. The DSP-based

http://doi.org/10.30684/etj.v40i2.2160
Received 08 May 2021; Accepted 18 June 2021; Available online 25 February 2022
2412-0758/University of Technology-Iraq, Baghdad, Iraq
This is an open access article under the CC BY 4.0 license http://creativecommons.org/licenses/by/4.0

implementation is sequential and therefore does not maintain the layer of neurons' parallel architecture. Implementations of ASIC do not enable users to re-configure. Thus, FPGA is the appropriate hardware for implementing neural networks because it conserves parallel neuron architecture in the layer and provides flexibility for reconfiguration [9].

Since the early 1990s, the hardware implementation of an ANN with field programmable gate arrays has been an important field of applied research for various domains. At first, the only widely accepted method was the use of Hardware Description Languages for VLSI circuits, particularly very large integration circuits (VHDL) or Verilog. Today, engineers can build, simulate and validate a design using modern Electronic Design Automation software and test complex systems easily with high confidence in the first time proper operation of the final product [10]. In recent years researchers have been trying implementation of an efficient tan-sigmoid on FPGA. Biradar et al. [8] showed that the hardware implementation of a multi-layer feed-forward neural network based on creating custom IPs is designed using the system-on-chip design methodology on Virtex 5 XUPV5-LX110T. Using of System-on-Chip design methodology enables design reuse while improving the performance metrics. Results for Hyperbolic tangent activation function approximation with ε = 0.04. (As the number of slice LUTs is 2%, the number of Slice Registers is 2%, the number of Bonded IOBs is 1%, the number of Block RAM/FIFO is 1%, the number of DSP48Es is 4%, and the number of memory is 1% KB).

The Xilinx Coordinate Rotation Digital Computer (CORDIC) has been used to approximate the Log sigmoid transfer function on FPGA. The architecture can be used up to 266,429 MHz with a clock rate. FPGA chip statistics and results of the experimental study for practical use were shown as the number of Slice register = 28%, the number of Slice LUT = 57%, the number of Fully used LUT-FF pair = 63%, the number of Bonded IOB = 48%, RAM and the number of FIFO block = 0%, the number of BUFG/ BUFGctrl = 3%, and the number of DSP48E1s is 1%) [11]. Mitra and Chattopadhyay created the Inverse Definite Minimum Time function (IDMT) and tested it using XILINX Spartan-3AN FPGA with a very simple ANN- model. The Sigmoid activation has played a vital role in the design and implementation of the ANN .where the result is (the number of 4 input LUT is 1.56%, the number of Slices is 2.54%, the number of I/O Pin is 1.19%, the number of 18x18 Mult is 6.25%, and the number of 16bit RAMB is 0.00%) [12]. The idea of the proposed work is to mix up the second-order non-linear function (SONF) and the differential look-up table (d LUT) to reduce the variance between hardware-based and software-based ANN in sigmoid function outputs a maximum of ‘’0.0022’’. Such accuracy is ten times the one using SONF and two times greater than that of using traditional LUT with ‘’16kbits’’ ROM [13]. P. Ramachandran et al. suggested replacing rules with a swish, which improves classification accuracy, which is written as f (x) = x sigmoid (βx) simply replacing ReLUs with Swish units enhances Image Net’s top-1 rating accuracy by ‘’0.9% ‘’for Mobile NASNet-A and ‘’0.6%’’ for Inception-ResNet-v2. Swish simplicity and its similitude with the ReLU enable the replacement of ReLUs by Swish units in any neural network by learners. Ramachandran et al. and Sawaguchi and Nishi [14,15] proposed a deterministic dropout algorithm increase efficiency for the Neural Network Learning accelerator, a novel implementation on Xilinx Zynq-7020 (the resource utilization in LUT is 66%, in Register is 47%, in DSP is 51%). Mish has been suggested as a novel activation function [16], which can be: $f(x) = x.tanh$ (soft plus (x)), Experiments show that Mish is more efficient in many deep networks across challenging datasets than both ReLU and Swish, along with other regular activation functions. For instance, the Top 1 test accuracy of the Mish network was increased by ‘’0.494%’’ and ‘’1.671%’’, compared to the same CIFAR100 classification network in Squeeze Excite Net- 18. Sarić et al. presented worked automated system based on a high-precision machine learning algorithm on FPGA chips developing an automated system for diagnosis of epileptic seizures using MLP ANN with the best data accuracy developed in Tensor Flow framework and further tested in MATLAB that makes it compact and easily useable in daily diagnostics in healthcare facilities can successfully be implemented. The high-efficiency automated device has achieved accuracy ‘’95.14’’ percent, which uses the total hardware footprint of ANN implementation, at ‘’11’’ percent. Cyclone IV E series board tested in FPGA Altera D2-115 [17]. Based on the neural network design, Koyuncu et al. [18] presented the hydrogen energy system for running on VHDL Field programming gate chips with 32-bit floating-point IEEE-754-1985 and synthesized to use the programming tools of Xilinx ISE support the Virtx-7 FPGA chip (VC7VX485T, Package FFG 1761 2 speed) Program 14.7. (As usage is reduced number of registers for slices is 17%, the number of LUTs for Slice is 35%, the number of DSP48EL is 1%, the number for IOBs is 27%, and the high frequency of the clock 281.702 MHZ). Zhang et al. [19] proposed an efficient hardware implementation system to identify objects for optical remote sensing. First, we design the hardware CNN-based model, which establishes the basis to map the network efficiently into FPGA, implemented on a Xilinx ZYNQ xc7z035 FPGA. (The resource utilization in LUT is 48.4%, FF is 31.7%, Dsp is 21.3%, Bram is 74%). This paper aims to design and implement an efficient tan- sigmoid in a different approach with the best efficiency and significantly confirmed the lowest implementation costs than the traditional approach.

The paper is organized as follows: the second section explains the theoretical part of the non-linear activation function in the artificial neural network. The third section describes the approximation method for the log sigmoid function. In the fourth section, the proposed work is presented. The fifth section presents the results and compares the proposals and previous works. Finally, section sixth introduces the conclusion.

2. Activation Function

ANN consists of several architectural categories. Many variables must be considered to solve a specific problem in the application of the ANN since the ANN is only excellent when the selection exactly matches the target problem. Multiple feed-forward networks consist of the input layer, hidden layer, and output layer, one of the popular ANN classes. Each neuron of the previous layer is passed to the next layer. Neuron determines the amount of each one it receives. Each neuron's output value is determined by its activation function. The general structure of the artificial neural multi-layer network is shown in Figure 1 [20].

The primary neuron structure is shown in Figure 2 that $(p1, p2 \ldots . pn)$ are inputs , $(w_{11}, w_{12} \ldots . w_{1R})$ are the weight associated and the bias is b, and f is the transfer function [12].

The challenges during artificial neural network implementation include the non-linear functions used in the models of the neural networks [12]. The concept of a hardware implementation for neural networks is not as easy to perform sigmoid functions. Direct implementation is very costly for non-linear Sigmoid functions [21]. because of the unlimited exponential series and division [22]. A challenging task faced by designers is efficiently implementing the sigmoid feature in the FPGA [23] Several methods were suggested to solve these problems to simplify implementation [24]. One of these is the approximation log sigmoid method.

3. Approximation log sigmoid method

The log sigmoid curve is an S-shape curve that varies between 0 and 1. The log sigmoid is shown in Figure 3. In hardware, log sigmoid cannot simply apply since it contains an infinite sequence of exponentials. Computationally simplified sigmoid functional alternatives are, in most cases, used approximate sigmoid functions is a practical approach with simple FPGA designs called Piece-wise linear approximation (PLW). This defines a combination of $y = ax + b$ lines used for approximating log sigmoid function, a series of shifts, and operations used to perform the sigmoid functions [23]. The PWL approach approximates the sigmoid function by dividing it into five linear parts, called segments. Table 1 represents the log sigmoid function segments. The approximation precision can be achieved efficiently by increasing the number of segments. Still, the hardware implementation will be more complicated, showing the approximation function performance compared with the Log Sigmoid function [20].

4. The Proposed tan-sigmoid design

Three approaches have been proposed for achieving efficient tan-sigmoid on FPGA. These approaches are tan-sigmoid design using approximation log-sigmoid method, tan-sigmoid design using look-up table method, and tan-sigmoid design using the polynomial method.

4.1 Proposed tan-sigmoid design using Approximation log sigmoid method

This approach proposed a solution to the non-linear activation function problem. The practical tan-sigmoid function with the FPGA design approach has been considered for implementing the Artificial Neural Network model. However, it is complicated to achieve direct implementation of this function and very costly because containing exponential, which gives non-linear behavior .They are not included in FPGA library blocks and are challenging to synthesize to suit every form of parallel neural network. Moreover, this activation function contains a division operation that VHDL can apply, but still far from area efficient and speed to be successfully incorporated into this design.

Tan- Sigmoid and Log -sigmoid activation functions are widely used in artificial neural networks. That which can be represented in equations (1) and (2) [18] [25] [22].

$$logsigmoid = \frac{1}{1+e^{-x}} \tag{1}$$

$$tan - sigmoid = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{2}$$

Tan-sigmoid is the S-formed curve that ranges from -1 to 1, whereas the log -Sigmoid is different in the range of 0 to 1. Therefore, the tan-sigmoid equation in (2) can be rewritten as follows to simplify the equation and reduce the hardware cost and find a relationship between tan-sigmoid and log sigmoid:

$$tansig(x) = \frac{Sinh(x)}{cosh(x)} = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{3}$$

$$= \frac{e^{x}-e^{-x}+e^{-x}-e^{-x}}{e^{x}+e^{-x}} \tag{4}$$

$$= 1 - \frac{2e^{-x}}{e^{x}+1} \tag{5}$$

By splitting the numerator and denominator by e^{-x}, (5) becomes:

$$tansig = 1 - \frac{2}{e^{2x}+1} \tag{6}$$

$$tansig = 2logsig(2x) - 1 \tag{7}$$

Tan-sigmoid can therefore be represented using log-sigmoid. This work used an approximation log sigmoid method for tan-sigmoid where the tan curve was divided into five parts, as shown in Table 2. The proposed implementation offers high precision and effective hardware implementation simultaneously. Figure 5 shows the proposed FPGA hardware implementation using the approximation method.

The significance of this proposal is to substitute multiplication by simple shift operations such as, if X is the input, then X is shifted one time to the left become (2X). The conditions are added as in Table 2. Two multiplexers were used to choose the

output according to the conditions set. The selector of the first multiplexer (Mux) is select (00) when the input is less than or equal -4. While it is select (01) when the input is greater than- 4 and less than or equal to -0.6, select (10) when the absolute input is less than 0.6, and select (11) when the input is greater than or equal to 0.6 and less than 4. The selector for the second multiplexer (Mux) chooses (0) when the input is greater than 4. The output is shifted one time to the left, which is multiplied by the number 2 and subtracted from the one. The shift and add operations can be eliminated and replaced with a simple logic design by performing a direct transformation from input to sigmoidal output. This technique replaced the multiplication and addition operations with a simple gate design, resulting in a very small and fast digital approximation of a tan-sigmoid function. Figure 6 shows the result of the proposed tan-sigmoid design using an approximation method. The blue line represents the original tan-sigmoid function. In contrast, the red line represents the approximate solution for the digital circuit's output. The proposal has been implemented in MATLAB R2012a and was created that linked with ISE 14.7 Xilinx Spartan-3A and Spartan3AN xc3s700a-4fg484 FPGA Xilinx, and as shown, the proposed matches the original function.

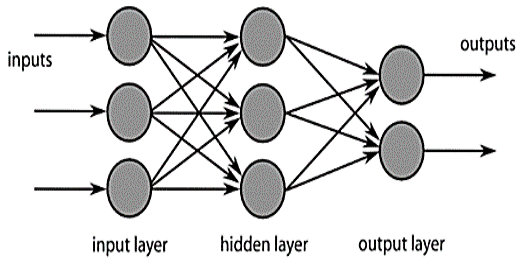

Figure 1: General artificial neural network multi-layer structure [20]

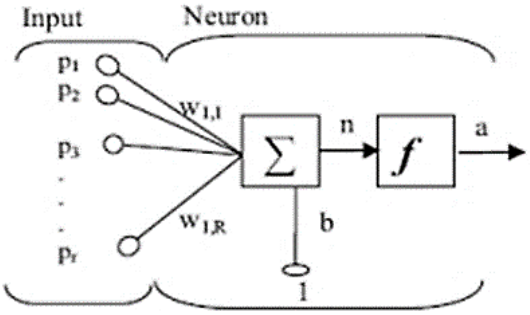

Figure 2: General artificial neuron model [12]

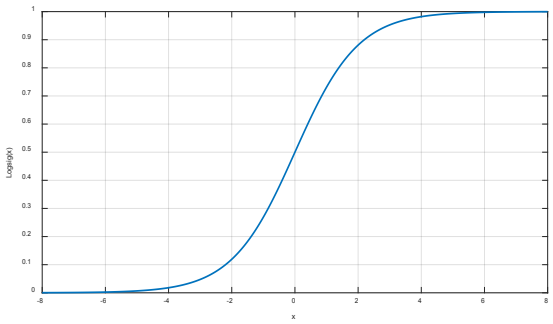

Figure 3: Log sigmoid function [20]

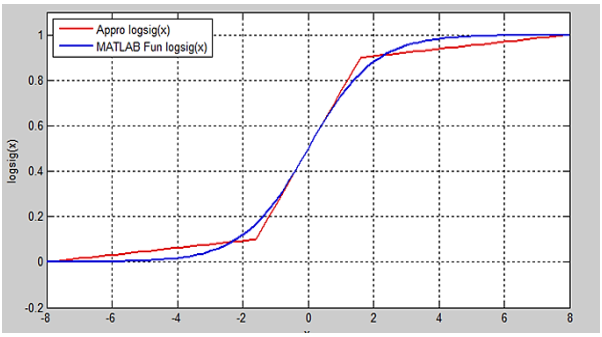

Figure 4: Output comparison between log sigmoid function and approximation

Table 1: log sigmoid Approximation

Condition	Operation
$x \leq -8$	$f(x) = 0$
$-8 < x \leq -1.6$	$f(x) = \left(8 - \frac{\lvert x \rvert}{64}\right) - 1$
$\lvert x \rvert < 1.6$	$f(x) = \left(\frac{x}{4} + 0.5\right)$
$1.6 \leq x < 8$	$f(x) = 1 - \left(8 - \frac{\lvert x \rvert}{64}\right)$
$x > 8$	$f(x) = 1$

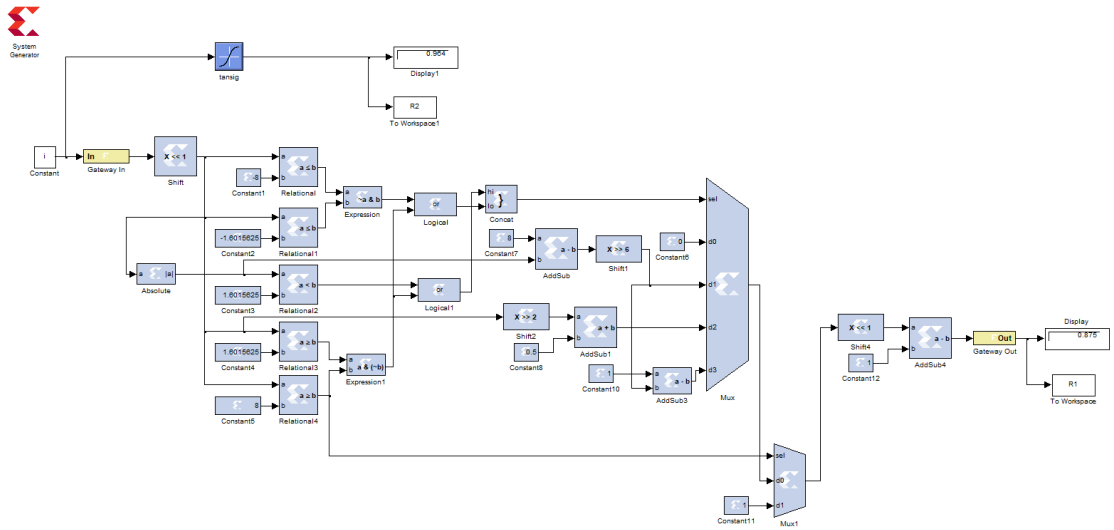

Figure 5: proposed tan-sigmoid implementation on FPGA hardware using an approximation method

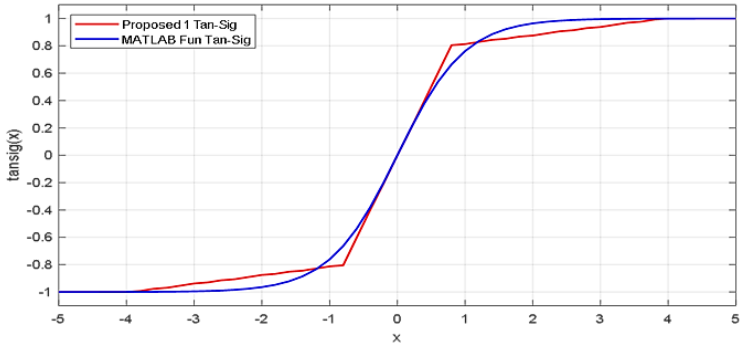

Figure 6: The approximant action result on FPGA for the proposed tan-sigmoid

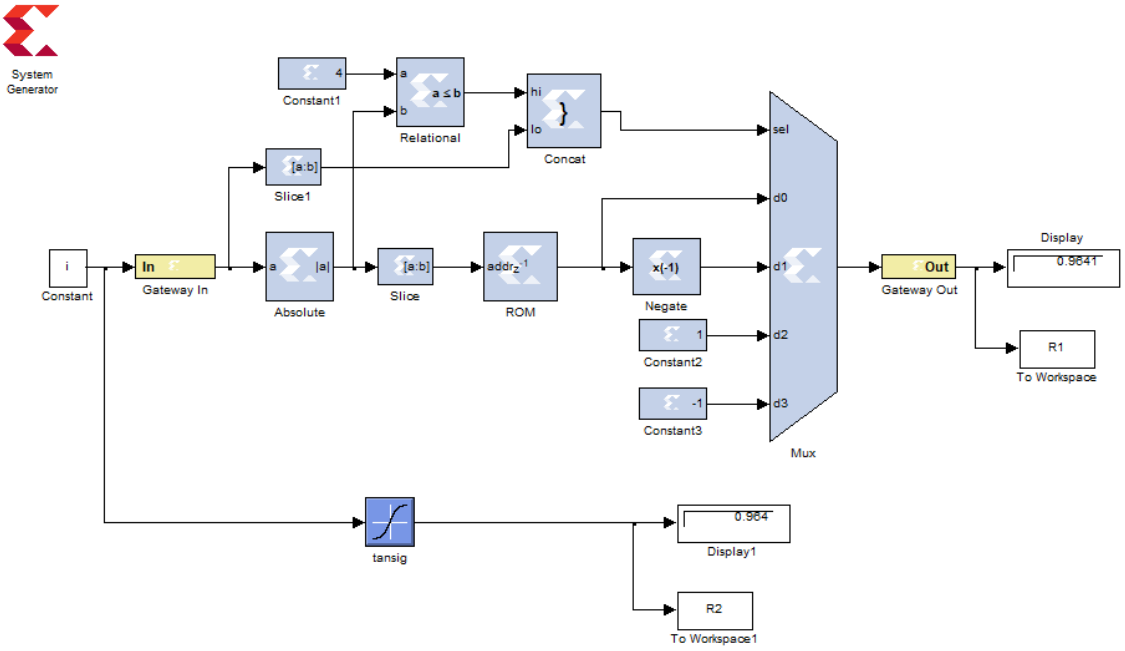

Figure 7: proposed tan-sigmoid implementation on FPGA hardware using the look-up table method

4.2 Proposed tan-sigmoid design using a look-up table method

Another proposal for tan-sigmoid is to use the look-up table method by dividing the sigmoid function into four linear segments. From the curve tan sigmoid, notice from the tan that the influencing values are from 0 to 4, after which their value is stable at 1, and from 0 to -4, they are the same influencing values. Still, they are reversed, and after -4, they are stable at -1. Therefore, suggest a memory to store the tan-sigmoid magnitude of the values from 0 to 4. Table 3 shows the calculation of the tan-sigmoid segments. By increasing the number of segments, the exact segmentation can be achieved, and the hardware area can be further enhanced. The proposed look-up table for a non-linear function is adequate.

The non-linear function outputs for different inputs from (0-4) are stored in the manner of the above table at memory location; Figure 7 shows proposed FPGA hardware implementation using the look-up table method.

If x is the input, then adding absolute and slicing it only takes 7 bits because the memory address is seven. Then, the conditions are added as in Table 3. When compared with No. 4, one multiplexer is used to choose the output according to the conditions set. The selector for multiplexer select (00) when the input is less than or equal -4, select (01) when the input greater than of 0 and less than or equal 4, select (10) when the input less than 0, and greater than -4 and select (11) when the input greater than 4. The rounding result of the proposed tan-sigmoid function in FPGA is shown in Figure 8. The proposed design corresponds to the original function. The original function is represented by the blue line, while the red line represents the look-up table solution for the digital circuit's output.

Table 2: Tan-sigmoid Approximation

Condition	Operation
$x \leq -4$	$f(x) = -1$
$-4 < x \leq -0.6$	$f(x) = \left(8 - \frac{\lvert 2 * x \rvert}{64}\right) - 1$
$\lvert x \rvert < 0.6$	$f(x) = \left(\frac{2 * x}{4} + 0.5\right)$
$0.6 \leq x < 4$	$f(x) = 1 - \left(8 - \frac{\lvert 2 * x \rvert}{64}\right)$
$x > 4$	$f(x) = 1$

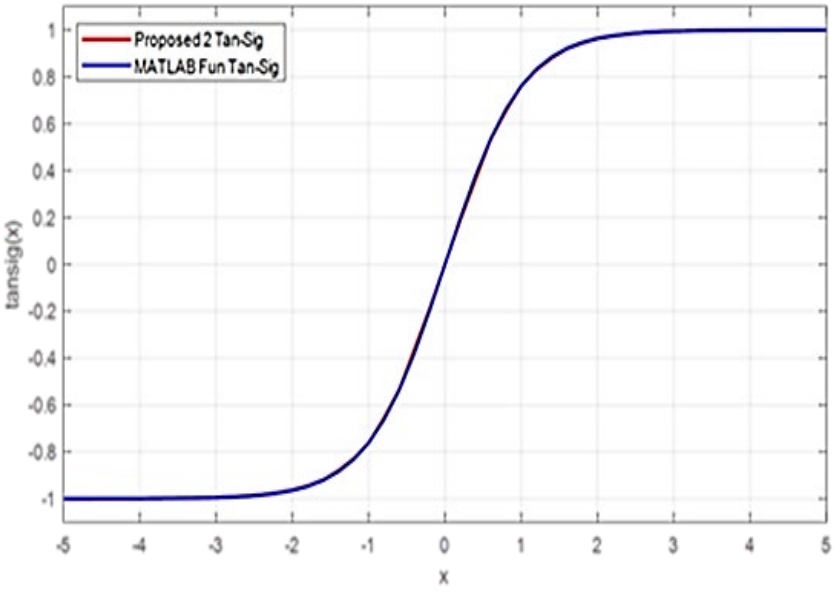

Figure 8: The look-up table result on FPGA for the proposed tan-sigmoid

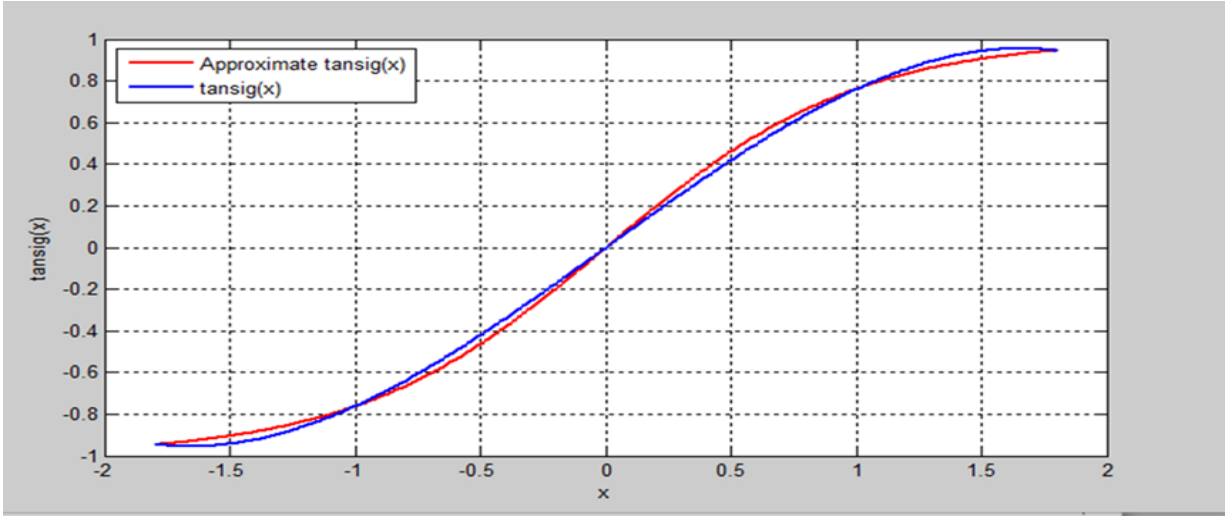

Figure 9: Curve polynomial for tan-sigmoid

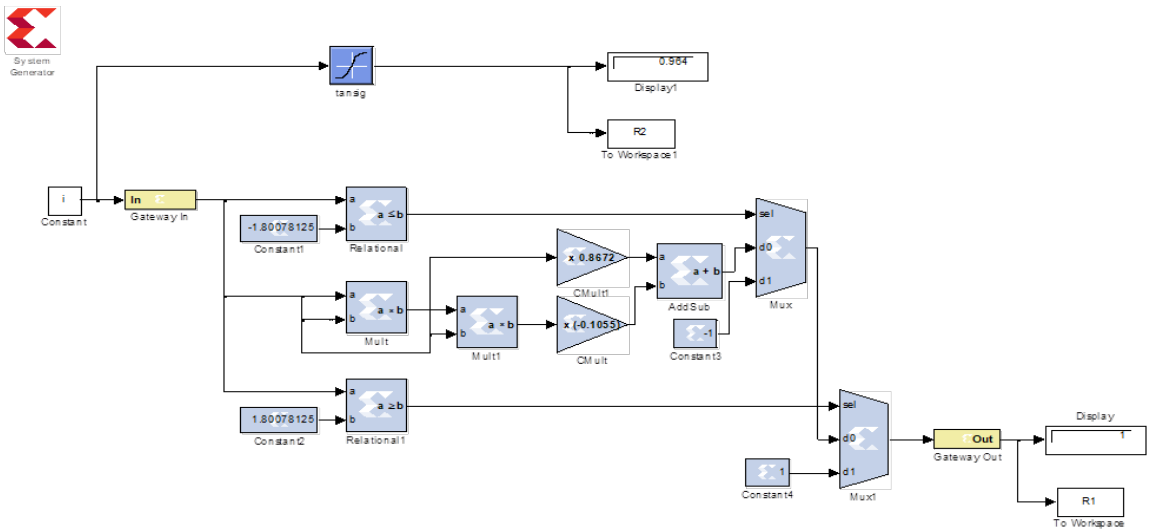

Figure 10: Curve polynomial tan-sigmoid proposed implementation on FPGA hardware

Table 3: Tan- sigmoid look-up table

Condition	Operation
$x \leq -4$	$f(x) = -1$
$0 < x \leq 4$	$f(x) = Memory$
$0 > x > -4$	$f(x) = Memory * -1$
$x > 4$	$f(x) = 1$

Table 4: Tan-Sigmoid Polynomial

Condition	***Operation***
$X <= -1.8$	$f(x) = -1$
$-1.8 < X <= 1.8$	$f(x) = polynomial\ equation$
$x > 1.8$	$f(x) = 1$

Table 5: Resources used in the proposed design

Device Utilization Summary			
Logic Utilization	Used	Available	Utilization
Number of 4 input LUTs	450	11,776	3%
Number of occupied Slices	286	5,888	4%
Number of Slices containing only related logic	286	286	100%
Number of Slices containing only unrelated logic	0	286	0%
Total Number of 4 input LUTs	475	11,776	4%
Number Used as logic	450		
Number Used as route-thru	25		
Number of bonded IOBs	81	372	21%
Number of MULT 18XS10s	3	20	15%
Average Fan out of Non-Clock Nets	2.18		

Table 6: Resources used in the proposed design

Device Utilization Summary			
Logic Utilization	Used	Available	Utilization
Number of 4 input LUTs	61	11,776	1%
Number of occupied Slices	38	5,888	1%
Number of Slices containing only related logic	38	38	100%
Number of Slices containing only unrelated logic	0	38	0%
Total Number of 4 input LUTs	63	11,776	1%
Number Used as logic	61		
Number Used as route-thru	2		
Number of bonded IOBs	40	372	10%
Number of BUFGMUXs	1	24	4%
Number of RAMB168WEs	1	20	5%
Average Fan out of Non-Clock Nets	1.80		

Table 7: Resources used in the proposed design

Device Utilization Summary			
Logic Utilization	Used	Available	Utilization
Number of 4 input LUTs	136	11,776	1%
Number of occupied Slices	99	5,888	1%
Number of Slices containing only related logic	99	99	100%
Number of Slices containing only unrelated logic	0	99	0%
Total Number of 4 input LUTs	185	11,776	1%
Number Used as logic	136		
Number Used as route-thru	49		
Number of bonded IOBs	35	372	9%
Average Fan out of Non-Clock Nets	2.17		

Table 8: Comparison between proposed approaches

Proposed	**Slice**	**LUT**	**FF**	**utilization**
Proposed tan-sigmoid design using approximation log sigmoid method	99	185	0	1%
proposed tan-sigmoid design using the look-up table method	38	63	0	1%
proposed tan-sigmoid design using the polynomial method	286	475	0	4%

Table 9: Comparison with previous work

Study		Device	Slice	LUT	FF	Utilization
Proposed tan- sigmoid design using the look-up table method		Xilinx Spartan-3A and Spartan3AN xc3s700a-4fg484 FPGA	38	63	0	1%
R. G. Biradar, et al. [8]	approximation of tangent	Virtex – 5	1761	1976	2	2%
R. Sarić, et al. [17]	MLP ANN with dataset developed in Tensor Flow	Altera D – 115 FPGA	114	12,971	0	LUT 17% Slice 12%

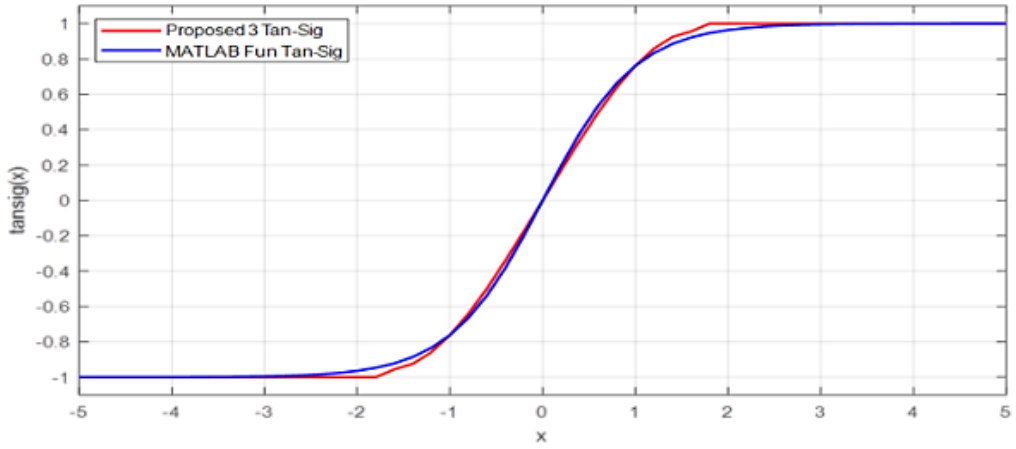

Figure 11: The Polynomial tan-sigmoid result on FPGA

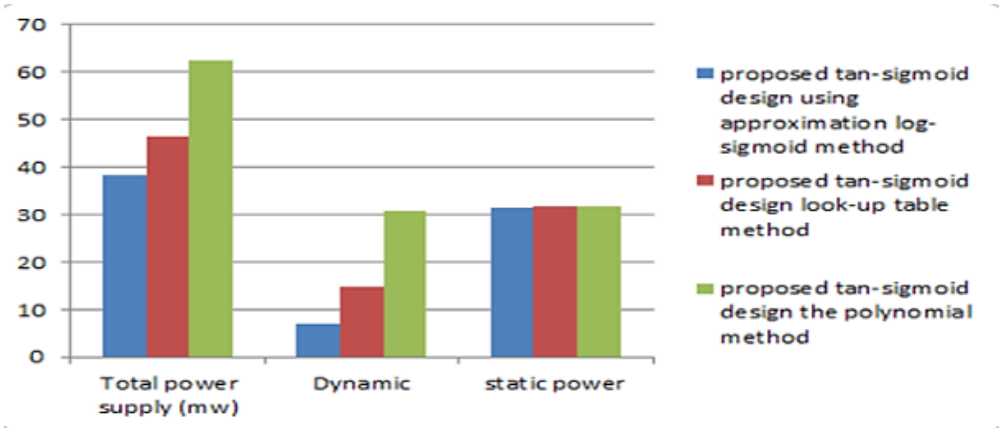

Figure 12: Power supply summary comparison between the proposed approaches

4.3 Proposed tan-sigmoid design using the polynomial method

Another proposal that can be used to implement the tan-sigmoid function using a polynomial method is by dividing the sigmoid function into three linear segments. From the tan-sigmoid curve, notice that the influencing values are from 0 to 1.8, after which their value is stable at 1 and from 0 to -1.8 they are the same influencing values, but they are reversed, and after -1.8, they are stable at -1. Therefore make work on the influencing value of the curve polynomial (-1.8 to 1.8) as in Figure 9.

Figure10 shows the proposed FPGA hardware implementation using a polynomial method.N If the x input, the conditions are added in Table 4. Two multiplexers were used to choose the output according to the conditions set. The selector for the first multiplexer (Mux) select (0) when the input is less than or equal to -1.8 and select (1). and the input greater than -1.8 and less than or equal 1.8. The selector for the second multiplexer (Mux) select (0) when the input is greater than or equal to 1.8. Figure 11 shows that the proposed design matches the original tan-sigmoid function. The result of designing the original function is represented by the blue line, while the red line represents the proposed polynomial solution for the digital circuit's output. Table 4 shows the implementation technique to curve polynomial for tan- sigmoid function.

The expression of the polynomial tan-sigmoid can be described as follows:

$$y = -0.1053x^3 + 0.8675 \quad (8)$$

5. Results and Discussion

This section shows the results of the proposed designs and implementation in ISE design suite 14.7 (Xilinx Spartan-3A xc3s700a-4fg484 FPGA) to execute efficient tan sigmoid as shown in Figure 5, Figure 7, and Figure10. The approaches proposed in MATLAB 2012a were created linked with ISE 14.7 Xilinx. Table 5 shows the resource utilization of the proposed tan-sigmoid using an approximation method, which shows the utilization for the number of occupied slices. The total number of 4 input LUTs is ‘’1%’’.

Table 6 shows the resource utilization of the proposed tan-sigmoid using the look-up table method, which shows the utilization for the number of occupied slices. The total number of 4 input LUTs is ‘’1%’’. Table 7 shows the resource utilization of the proposed tan-sigmoid using the polynomial method, which shows the utilization for the number of occupied slices. The total number of 4 input LUTs is ‘’4%’’. Table 8 shows a post-implementation summary and a comparison of the proposed designs. It can be seen that the second proposal took the least number of occupied slices and the total number of 4 input LUTs.

In comparison, the third proposal took the highest number of occupied slices and the total number of 4 input LUTs. The results show that the best proposal among them is the second proposal because it occupies the least space in FPGA. As a result, the proposed circuits can be applied with higher speed and precision in different applications of ANN-based real-time hardware.

Figure 12 shows a comparison between the proposed approaches in terms of power consumption. It shows that the proposed tan-sigmoid design using an approximation method is the lowest consumption of total power supply.

The proposed tan-sigmoid design using the look-up table method was compared to the previous studies because it is the best proposal among the proposed approaches. As illustrated in Table 9, the proposed implementation is efficient tan-sigmoid and low an area than R.G Biradar et al. [8] and [12]. The architectures proposed are successful and suitable for the large-scale implementation of the ANN networks.

6. Conclusion

In the development and implementation of the ANNs, The continuous function of activation plays a significant role since hardware application of this form in its natural shape is difficult. In this research, three proposed tan-sigmoid designs are based on a different method. These methods are tan-sigmoid based on the log approximation method, tan-sigmoid based on the look-up table method, and tan-sigmoid the polynomial method. Where these proposed are implemented by using Hard FPGA blocks. From the performance analysis in terms of resource utilization, it can be concluded that the proposed tan-sigmoid using the look-up table method that has the best proposal among them is the second proposal because it occupies the least space in FPGA. The proposed design achieves an efficient tan-sigmoid that requires less hardware and more quickly using FPGA. From this conclusion, the efficient design of tan-sigmoid in this research. The resource utilization is about (1%) for slices and LUT. While FF (0%) compared with two other proposed approaches, it is the lowest resource utilization relative to the related works.

Author contribution

All authors contributed equally to this work.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Data availability statement

The data that support the findings of this study are available on request from the corresponding author.

Conflicts of interest

The authors declare that there is no conflict of interest.

References

[1] F. Hashim, A. Ja’afar, K. A. Ahmad, A. J. S. Hi-Fi. Syam2., MLP Based Tan-Sigmoid Activation Function for Cardiac Activity Monitoring, MATEC, Web. Conf., 255 (2019) 3005. https://doi.org/10.1051/matecconf/201925503005

[2] L. Zhang, Artificial neural network model-based design and fixed-point fpga implementation of hénon map chaotic system for brain research, 2017 IEEE XXIV, Int. Conf. Electron. Electr. Eng. Comput., (2017) 1–4. https://doi.org/10.1109/INTERCON.2017.8079643

[3] M. Alas , S. I. Ali, Prediction of the high-temperature performance of a geopolymer modified asphalt binder using artificial neural networks, Int. J. Technol., 10 (2019) 417–427. https://doi.org/10.14716/ijtech.v10i2.2421

[4] E. Srinivasan , S. Himavathi, Neural network implementation using FPGA: issues and application, Int. J. Inf. Technol., 2 (2008) 86–92.

[5] S. R. Chiluveru , M. Tripathy, Non-linear activation function approximation using a REMEZ algorithm, IET. Circuits, Devices .Syst., 15 (2021) 630-640. https://doi.org/10.1049/cds2.12058

[6] A. Savran , S. Ünsal, Hardware implementation of a feed-forward neural network using fpgas, third. Int. Conf. Electr. Electron. Eng., (2003) 3–7.

[7] K. M. Hamdia, X. Zhuang, T. Rabczuk, An efficient optimization approach for designing machine learning models based on genetic algorithm, Neural. Comput. Appl., 33 (2021) 1923–1933. https://doi.org/10.1007/s00521-020-05035-x

[8] R. G. Biradar, A. Chatterjee, P. Mishra, K. George, FPGA implementation of a multi-layer Artificial Neural Network using System-on-Chip design methodology , Conf. Cogn. Comput. Inf. Process., (2015) 1–6.

[9] S. Himavathi, D. Anitha, E. Srinivasan, Feedforward neural network implementation in FPGA using layer multiplexing for effective resource utilization, IEEE, trans. Neural. netw., 183 (2007) 880–888.

[10] A. Tisan , J. Chin, An end-user platform for FPGA-based design and rapid prototyping of feed-forward artificial neural networks with on-chip backpropagation, IEEE, Trans. Learn. Technol., 123 (2016) 1124–1133. https://doi.org/10.1109/TII.2016.2555936

[11] M. Alçın, I.Pehlivan, İ. Koyuncu, Hardware design and implementation of a novel ANN-based chaotic generator in FPGA, Optik., 127 (2016) 5500–5505. https://doi.org/10.1016/j.ijleo.2016.03.042

[12] S. Mitra , P. Chattopadhyay, Challenges in implementation of ANN in embedded system, Int. Conf. Electr. Electron. Optim. Tech., (2016) 1794–1798 . https://doi.org/10.1109/ICEEOT.2016.7754996

[13] S. Ngah, R. A. Bakar, A. Embong, S. Razali, Two-steps implementation of sigmoid function for artificial neural network in Field Programmable Gate Array ARPN, J. Eng. Appl. Sci., 11 (2016) 4882–4888.

[14] P. Ramachandran, B. Zoph, Q. V Le, Searching for activation functions, arXiv Prepr. arXiv1710.05941, 1 (2017). https://doi.org/10.48550/arXiv.1710.05941

[15] S. Sawaguchi , H. Nishi, Slightly-slacked dropout for improving neural network learning on FPGA, ICT Express., 4 (2018) 75–80. https://doi.org/10.1016/j.icte.2018.04.006

[16] D. Misra, Mish: A self regularized non-monotonic neural activation function, arXiv Prepr. arXiv1908.08681, 1 (2019). https://doi.org/10.48550/arXiv.1908.08681

[17] R. Sarić, D. Jokić, N. Beganović, L. G. Pokvić, A. Badnjević, FPGA-based real-time epileptic seizure classification using Artificial Neural Network, Biomed. Signal. Process. Control., 62 (2020) 102106. https://doi.org/10.1016/j.bspc.2020.102106

[18] I. Koyuncu, M. Alcin, P. Erdogmus, M. Tuna, Artificial Neural Network-Based 4-D Hyper-Chaotic System on Field Programmable Gate Array, Int. J. Intell. Syst. Appl. Eng., 82 (2020) 102–108. https://doi.org/10.18201/ijisae.2020261591

[19] N. Zhang, X. Wei, H. Chen, W. Liu, FPGA implementation for CNN-based optical remote sensing object detection, ELECTR., 10 (2021) 282. https://doi.org/10.3390/electronics10030282

[20] S. Ngah , R. A. Bakar, Sigmoid function implementation using the unequal segmentation of differential look-up table and second order nonlinear function, J. Telecommun. Electron. Comput. Eng., 9 (2017) 103–108.

[21] I. A. D. Williamson, T. W. Hughes, M. Minkov, B. Bartlett, S. Pai, S. Fan, Reprogrammable electro-optic nonlinear activation functions for optical neural networks, IEEE, J. Sel. Top. Quantum. Electron., 26 (2019) 1–12. https://doi.org/10.1109/JSTQE.2019.2930455

[22] H. M. Al-Rikabi, M. A. Al-Ja'afari, A. H. Ali, S. H. Abdulwahed, Generic model implementation of deep neural network activation functions using GWO-optimized SCPWL model on FPGA, Microprocess. Microsyst., 77 (2020) 103141. https://doi.org/10.1016/j.micpro.2020.103141

[23] H. K. Ali , E. Z. Mohammed, Design artificial neural network using FPGA, IJCSNS, 10 (2010) 88.

[24] S. Gomar, M. Mirhassani, M. Ahmadi, Precise digital implementations of hyperbolic tanh and sigmoid function, Asilomar ,Conf. Signals. Syst. Comput., (2016) 1586–1589. http://dx.doi.org/10.1109/ACSSC.2016.7869646

[25] L. Moreira, R. Vettor, C. Guedes. Soares, Neural Network Approach for Predicting Ship Speed and Fuel Consumption, J. Mar. Sci. Eng., 9 (2021) 119. http://dx.doi.org/10.3390/jmse9020119

[26] L. Li, S. Zhang, J. Wu, An efficient hardware architecture for activation function in deep learning processor, 2018 IEEE, Int. Conf. Image. Vis. Comput., (2018) 911–918 http://dx.doi.org/10.1109/ICIVC.2018.8492754