IMPLEMENTATION AND COMPARISION OF ACTIVATION FUNCTIONS WITH DIFFERENT METHODS

Ragavi Pobbathi Ashok
Department of Electrical and Electronics Engineering
The University of Auckland
Auckland, New Zealand.
rpob456@aucklanduni.ac.nz

Adedomola Wuraola and Nitish Patel
Department of Electrical and Computer Engineering
The University of Auckland
Auckland, New Zealand
awur978@aucklanduni.ac.nz
&
nd.patel@aucklandac.nz

***Abstract*— this paper describes the development of the activation function of neural network in hardware. In this work, two different hardware implementations are used for both hyperbolic tangent sigmoid function and SQNL function. The hyperbolic tangent function is commonly used as the activation function in artificial neural networks. The first method uses a lookup table to approximate the function, while the second method reduces the size of the table by using range addressable decoding as opposed to the classic decoding scheme using VHDL language in QUATRUS software and simulation in Modelsim. In this method, the function values are pre-calculated in MATLAB and stored in a table. Comparison of tansig and SQNL activation functions with the above methods are done to find the best method with less resources and less error by altering the parameters of the segments in inputs. Hardware synthesis results show that proposed methods perform significantly faster and use less area compared to other similar methods with the same amount of error.**

Keywords*—*Quartus, Modelsim, MATLAB.

I. INTRODUCTION

The Artificial neural network (ANN) has become the main focus over the last few decades. The use of artificial neural network has become popular in many fields such as speed estimation, image processing, and pattern recognition.

Almost since the beginning of Artificial neural networks are used for modelling and control. ANN have been implemented mostly in software, but need for hardware implementation has arisen. One of the first commercial implementations was developed by Nestor and Intel in 1993 for an Optical Character Recognition (OCR), which uses Radial Basis Functions [1].

The need for physical implementations also arises in medical applications of ANN which work inside or in close connection with the body. The work referred in [2] contains many examples of neural vision systems that cannot be used attached to computers.

A large share of the difficulty of implementing an ANN in hardware comes from the non-linear activation function, most of the times a hyperbolic tangent. Since this function cannot be easily calculated by elementary functions, it has to be replaced by an approximation that is both less resource consuming and does not delay the calculation too much.

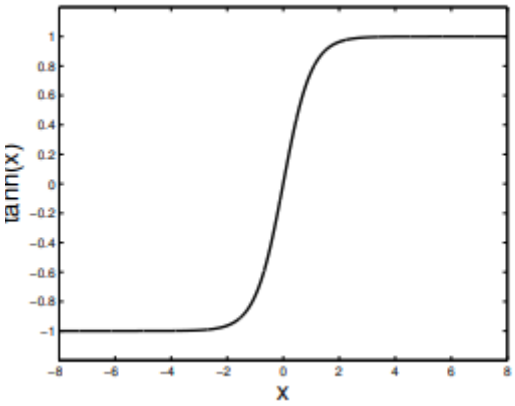

Figure 1. The Hyperbolic Tangent Activation Function

The hyperbolic tangent function is among the most widely used activation functions in ANNs. As it is shown in Fig. 1, this function produces a sigmoid curve, which a curve is having an "S" shape currently, there are several different approaches for the hardware implementation of the activation function. Piecewise linear approximation (PWL), lookup tables (LUTs), and hybrid methods have been widely used for this purpose. With the use of current hardware synthesizers, LUTs are not only faster, but also occupy less area than piecewise linear approximation methods. In this work, LUT implementation, Piecewise linear approximation method, Piece wise nonlinear approximation method, Implementation of full polynomial, Taylor series are proposed. Range addressable lookup tables are proposed as a solution that offers advantages compared to simple LUT implementation in terms of speed and area utilization. We initially present all the

above methods for the implementation for the hyperbolic tangent function and SQNL function. Comparison of the above activation functions are done by varying the segments of the input. The results show that a LUT implementation was significantly faster and smaller than a PWL approximation method, while possessing the same level of accuracy. RALUT implementations were presented afterwards, which yielded even further improvements over LUTs in area and speed. This paper presents an automated solution that makes use of Quartus, Mat lab and Model sim to obtain an FPGA implementation.

II. LITERATURE REVIEW

[3] In this paper the author focuses on implementation of activations functions (SELU and tanh) on hardware using the combinational circuit-based implementation method using the SELU introduced functions to converge the activation Functions. The proposed solution can be still improved by the efficiency in terms of area and timing delay.

Hardware implementation of sigmoid, hyperbolic, tangent, Electronic Automation

[4] In this paper the author used different methods for the approximation of activation function in hardware

*Piecewise Linear Approximations

Uses a series of linear segments to approximate uses a series of linear segments to approximate the activation function. The number and locations of these segments are chosen such that error, processing time, and area utilization are minimized.

*Lookup table Approximation

Uses a series of linear segments to approximate the activation function. The number and locations of these segments are chosen such that error, processing time, and area utilization are minimized.

*Hybrid Approximation Methods

Hybrid methods usually combine two or more of the previously mentioned methods to achieve better performance.

In all, LUT based methods need storage/memory and an extra pipeline stage for the memory access. All these methods, except the one that stores function values in LUTs, either require relatively complex calculations/logic or several clock cycles to minimize the approximation error

[5] This paper implements the implementation of hyperbolic tangent function in hardware using lookup table to approximate the function and the second one reduces the size using range addressable decoding

They use more area compared to the other methods.

[6] This paper presents an automated solution that makes use of System Generator of Xilinx to obtain an FPGA implementation of a neuron using Chebyshev interpolation. Two solutions were tested: Booth's algorithm and 2's complement, with improved solutions for the activation function.

It requires more resources to implement.

[7] The sigmoid activation function is implemented using approximation Function. The circuits are extracted, which will implement the neural network algorithm. The circuits are extracted by creating data flow graph from the approximated equation (1). The graph is then used to configure in hardware

$$f(net) = \frac{1}{2}\left[\frac{net}{1+|net|}+1\right]$$

Eq 1

III. METHODOLOGY

A. *Tansig activation function*

The transfer function is used to convert the input signals to output signals. In this work, Tansig function is used to active the function of the networks. Here 'tansig' transfer function is defined as

$$\text{tansig}(x) = 2/\left(1+\exp(-2x)\right)-1, \qquad \text{Eq 2}$$

Where x represents the corresponding input in Eq 2

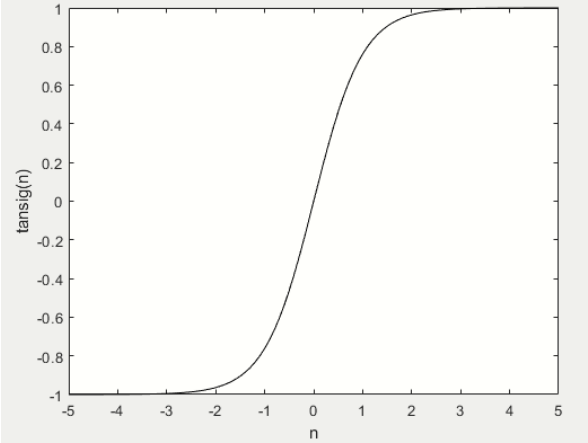

Figure 2. Represent the tansig activation function for the input which ranges from(-5,5).

B. *SQNL activation function*

It is a nonlinear activation function designed to implement on ASIC or FPGA hardware. This function mainly uses hard non-linearity and a time average to produce a precise square law non-linearity. The function is the represented as Eq 3

$$f_A(x) = \frac{2}{1+e^{ax+bx^3}} - 1 \qquad \text{Eq 3}$$

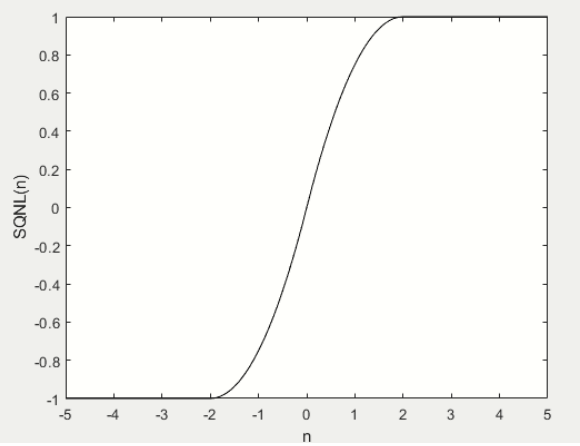

Figure 3. Represent the SQNL activation function for the input which ranges from(-5,5).

IV. NUMERICAL ANALYSIS

A. Piecewise Linear Approximation Method

The piecewise approximation method approximates by dividing the tansig function into linear segments called pieces. The accuracy of the approximation can be achieved by increasing the number of segments, also by subsequently increasing the area utilization of hardware.

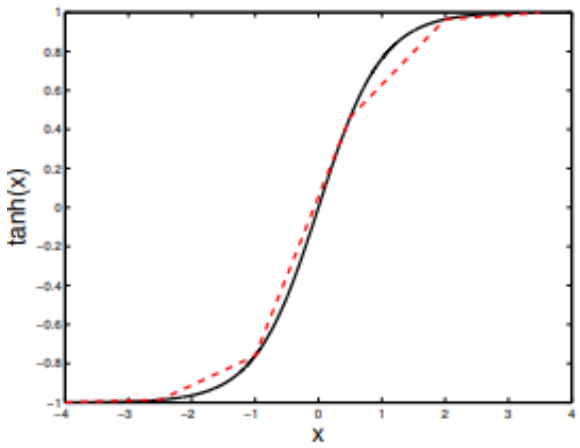

Figure 4. Piecewise Linear Approximation Method of tanh(x) with Five Segments with degree one.

B. Piecewise Nonlinear Approximation Method

This method can be used to implement the tansig function with piecewise second order approximation. Generally, this method approximated the tan sigmoid function by Eq 4.

$$f(x) = p1*x^2 + p2*x + p3 \quad \text{Eq 4}$$

The main drawback of this method is the need of two multiplications and two additions, which results in a low conversion speed.

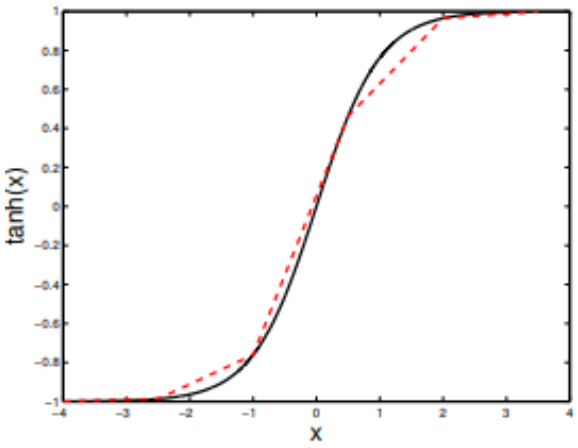

Figure 5. Piecewise Nonlinear Approximation Method of tanh(x) with Five Segments with degree two.

C. Full Polynomial Approximation Method

The full polynomial approximation method represents the tansig activation with third order.

This method gives the good result compared to the last two methods, but it results in a very low conversion speed because of more number of multiplications and additions as Eq 5.

$$f(x) = p1*x^3 + p2*x^2 + p3*x + p4 \quad \text{Eq 5}$$

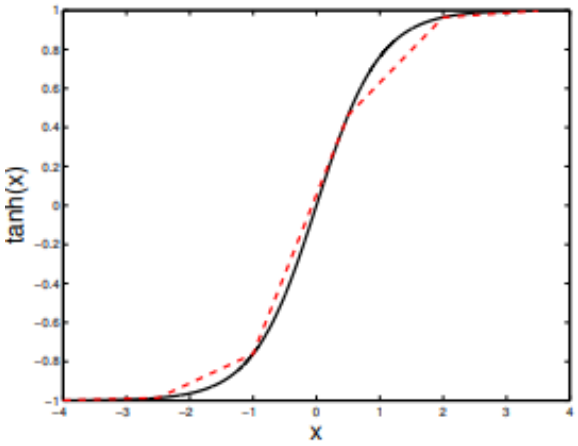

Figure 6. Full polynomial Approximation Method of tanh(x) with Five Segments with degree 3.

D. Range Addressable look up table

A range addressable lookup table, originally proposed to accurately approximate non-linear, discontinuous functions, shares many aspects with the classic LUT with a few notable differences. In LUTs, every data point stored by the table corresponds to a unique address. In RALUTs, every data point corresponds to a range of addresses. This alternate addressing approach allows for a large reduction in data points, particularly in situations where the output remains constant over a range. An example of this is the hyperbolic tangent function, where the output changes only slightly outside the range of (−2, 2). Rather than storing every individual point, a single point is used to represent an entire range. To implement the hyperbolic tangent function, we first wrote MATLAB code to select the minimum number of data points while keeping the maximum error beneath a specified threshold.

This large reduction in stored values is what drives the RALUT approach to achieve better results than a LUT implementation of the same function.

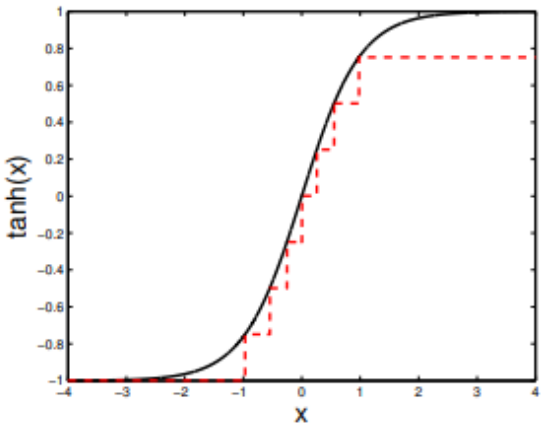

Figure 7. Range Addressable Lookup Table Approximation of tanh(x) with Eight Points

E. Look up Table

The major advantages of using a LUT is its high operating speed, particularly when compared to PWL approximation which uses multipliers in its design. There are two different ways to implement a lookup table in hardware. The first is to use a ROM. The main drawback of this method is that no further optimization can be done after the exact input/output bit patterns are known. The second method is to use a logic synthesizer to implement the table as a purely combinational circuit. This works well because the synthesizer performs very well in optimizing away large amounts of logic. In our implementation, we generated MATLAB code to determine the number of input and output bits as well as the output bit patterns themselves for a table with a specified maximum error.

Once the input/output characteristics of the table were determined, HDL code employing them was written.

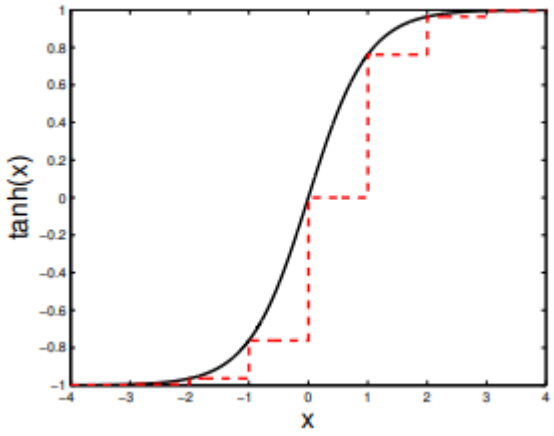

Figure 8. Lookup Table Approximation of tanh(x) with Eight Points

F. Taylor Series Expansion

e^u value calculation approach is the Taylor Series expansion. With being a real or complex number, the Taylor Series of an f(u) function is defined as shown in Eq.6

$$f(u)=f(a)+\frac{f'(a)}{1!}(u-a)+\frac{f''(a)}{2!}(u-a)^2+\cdots+\frac{f^n(a)}{n!}(u-a)^n \quad \text{Eq.6}$$

If a=0, the series is identified as the Maclaurin series. The Taylor Series for e u exponential function for 5th order is as shown in Eq.7

$$e^u = 1+u+\frac{u^2}{2!}+\frac{u^3}{3!}+\frac{u^4}{4!}+\frac{u^5}{5!}+\ldots\ldots+\frac{u^n}{n!} \quad \text{Eq.7}$$

After obtaining the e u exponential function, the TSTF is calculated by applying it to the expression given in Eq.8

$$TanSig(u)=\left(e^{2u}-1\right)/\left(e^{2u}+1\right) \quad \text{Eq.8}$$

The convergence of the function generally increases as the expansion order of the Taylor Series increases. However, the increase in the number of processes in hardware implementation also leads to an increase in hardware rate. Thus, a third order TS expansion (TS-3) was used in order to achieve convergence of the TSTF in this study. A design was made using VHDL in order to implement the TSTF on FPGA via the Taylor series

V. IMPLEMENTATION OF EXPONENTIAL FUNCTION

The implementation is divided into two parts

A. *Use of Table*

- Look up table Algorithm

STEP1: Represent a range of values (inputs).
STEP2: Perform the necessary activation function (ex-tansig) for the range of inputs.
STEP3: Obtain the output and tabulate the corresponding input and output.
STEP4: Tabulate the output and input.

- RALUT Algorithm

STEP1: Represent a range of values (input).
STEP2: Perform the necessary activation function (ex-tansig) for the range of inputs
STEP3: Obtain the output and collate a range of addresses (Output) to a single input value.
STEP4: Tabulate the output and input.

B. *Approximation methods*

- PWL Algorithm

STEP 1: Represent a range of values (inputs).
STEP2: Perform the necessary activation function (ex-tansig) for the range of inputs.
STEP3: Approximate the original function (original values) to the necessary polynomial values to get the same original function (Approximated values) of first order.
STEP4: Store the polynomial values

- PWNL Algorithm

STEP 1: Represent a range of values (inputs).
STEP2: Perform the necessary activation function (ex-tansig) for the range of inputs.
STEP3: Approximate the original function (original values) to the necessary polynomial values with a order 2 to get the same original function (Approximated values) of second order.
STEP4: Store the polynomial values

- FULL polynomial Algorithm

STEP 1: Represent a range of values (inputs).
STEP2: Perform the necessary activation function (ex-tansig) for the range of inputs.
STEP3: Approximate the original function (original values) to the necessary polynomial values with a order 3 to get the same original function (Approximated values) of last order.
STEP4: Store the polynomial values

- Taylor series expansion Algorithm

STEP1: Represent a range of values (inputs).
STEP2: Perform the necessary activation function (ex-tansig) for the range of inputs.
STEP3: Apply Taylor series for 5^{th} order and get the results of exponential values and find the appropriate output values.
STEP4: Store the output values.

VI. ERROR ANALYSIS FOR TAN SIGMOID

A. *LUT/ROM*

The analysis is performed in mat lab with different range of inputs for Tang sig activation function. The correspondence input and output are noted and stored. Quartus is used to store each output and input values in a different ROM addresses and for the Hard ware implementation.

B. *Range addressable Look up table*

The analysis is performed in the Mat lab considering different segments and results are shown below choosing the least error between the segments.
The input values are divided in to 8,12,24,32 segments and RALUT function is performed to calculate the output.

The results are plotted in the table 1 calculating the mean square error and root mean square error.

STRUCTURE-RALUT

SEGMENTS	Mean square error	Root mean square error
8	0.0030	0.0548
12	0.0012	0.0345
24	5.4633e-04	0.0234
32	2.2023e-04	0.0148

Table1. Results of RALUT with different input segments

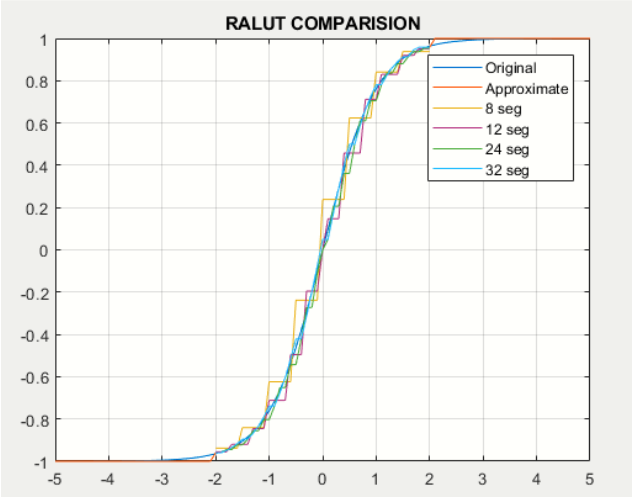

Figure 9. Comparison of different RALUT segments with original tansig function.

From the analysis it can be noted that error is minimum with large segments.

C. *Piecewise linear approximation method*

The analysis is performed in the Mat lab considering different segments and results are shown below choosing the least error between the segments

The input values are divided in to 4,8,12,16 segments and PWL function is performed to calculate the output.

The results are plotted in the table 2 calculating the mean square error and root mean square error.

STRUCTURE –PWL

SEGMENTS	Mean square error	Root mean square error
4	2.1080e-04	0.0145
8	6.5964e-05	0.0081
12	6.0476e-05	0.0078
16	5.6578e-05	0.0075

Table2. Results of PWL with different input segments.

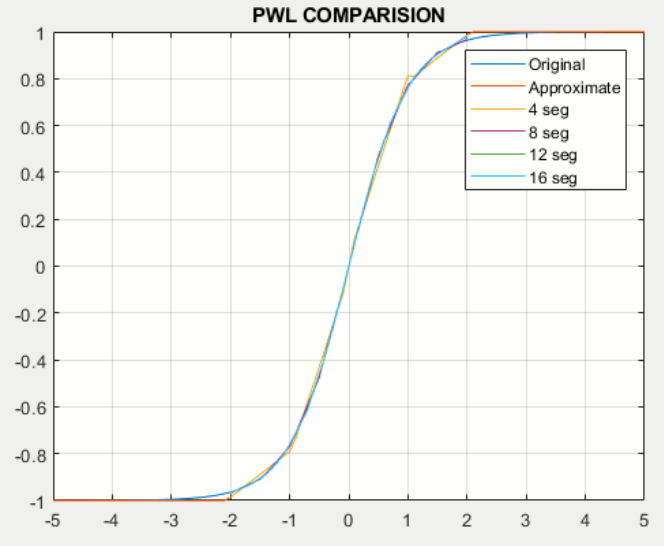

Figure 10. Comparison of different PWL segments with original tansig function.

From the analysis it can be noted that error is minimum with large segments.

D. *Piecewise Nonlinear approximation method*

The analysis is performed in the Mat lab considering different segments and results are shown below choosing the least error between the segments

The input values are divided in to 4,8,12,16 segments and PWNL function is performed to calculate the output.

The results are plotted in the table 3 calculating the mean square error and root mean square error.

STRUCTURE –PWNL

SEGMENTS	Mean square error	Root mean square error
4	5.4604e-05	0.0074
8	5.2948e-05	0.0073
12	5.2876e-05	0.0073
16	5.26789-05	0.0070

Table3. Results of PWNL with different input segments
STRUCTURE –PWNL

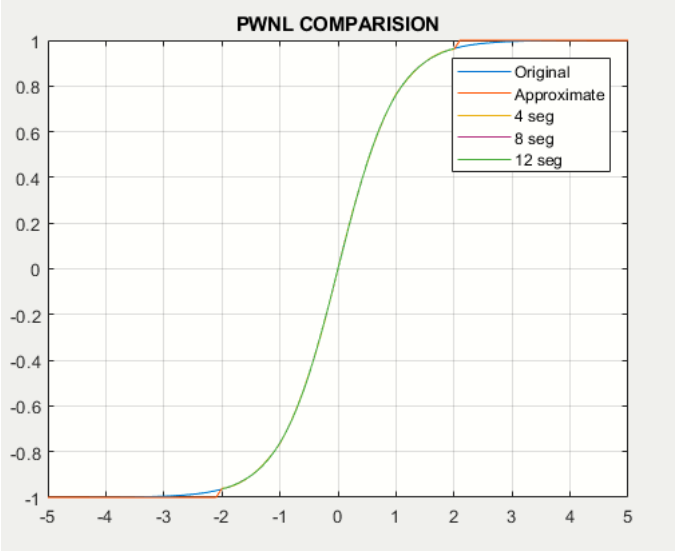

Figure 11. Comparison of different PWNL segments with the original tansig function.

From the analysis it can be noted that error is minimum with large segments.

E. *Full polynomial approximation method*

The analysis is performed in the Mat lab with different ranges and results are shown below choosing the least error between the different inputs.

The Different input values are considered, and full polynomial function is performed to calculate the output.

The results are plotted in the table 4 calculating the mean square error and root mean square error.

STRUCTURE-FULL POLYNOMIAL

Ranges	Mean square error	Root mean square error
-2:0.1:2	0.0010	0.0322
-5:0.1:5	4.7281e-04	0.0217
2:0.1:2	2.6383e-04	0.0162

Table4. Results of Full polynomial with different inputs

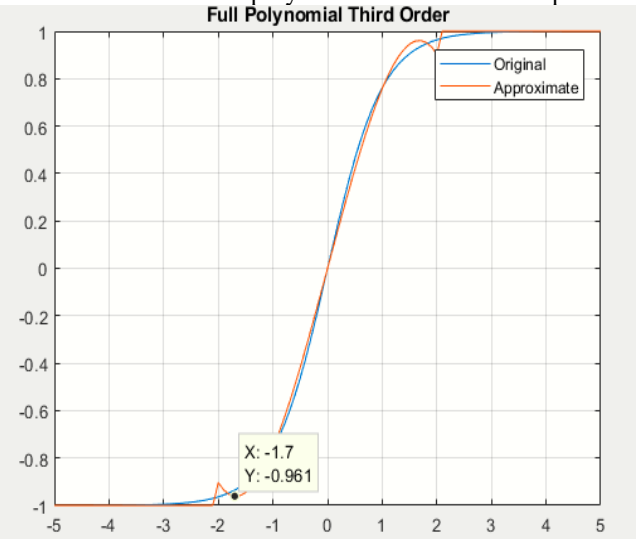

Figure 12. Representation of Full Polynomial with the original tansig function

From the analysis it can be noted that error is minimum with large segments.

VII. ERROR ANALYSIS FOR SQNL

A. LUT/ROM

The analysis is performed in mat lab with different range of inputs for SQNL activation function. The correspondence input and output are noted and stored. Quartus is used to store each output and input values in a different ROM addresses and for the Hard ware implementation.

B. Range addressable Look up table

The analysis is performed in the Mat lab considering different segments and results are shown below choosing the least error between the segments

The analysis is performed in the Mat lab considering different segments and results are shown below choosing the least error between the segments

The input values are divided in to 8,12,24,32 segments and RALUT function is performed to calculate the output.

The results are plotted in the table 5 calculating the mean square error and root mean square error.

STUCTURE-RALUT

SEGMENTS	Mean square error	Root mean square error
8	0.0029	0.0542
12	0.0012	0.0343
24	5.0214e-04	0.0224
32	1.6936e-04	0.0130

Table5. Results of RALUT with different input segments

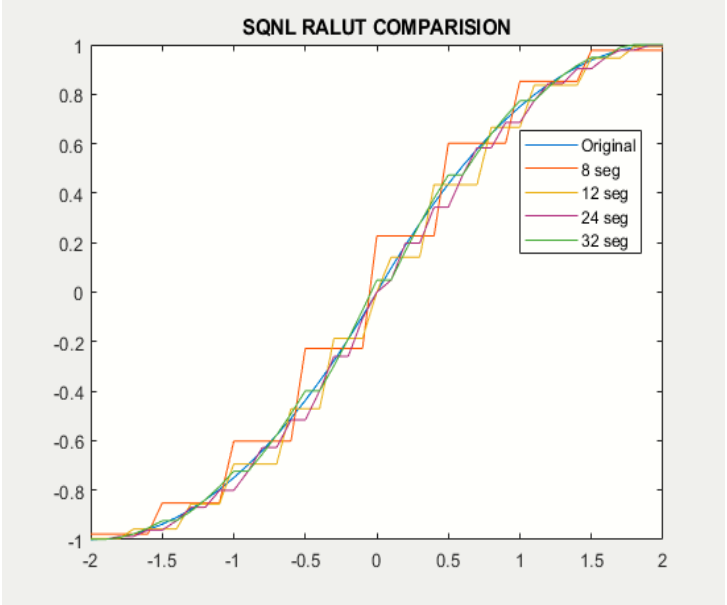

Figure 13. Comparison of different RALUT segments with original SQNL function.

From the analysis it can be noted that error is minimum with large segments.

C. Piecewise linear approximation method

The analysis is performed in the Mat lab considering different segments and results are shown below choosing the least error between the segments

The input values are divided in to 4,8,12,16 segments and PWL function is performed to calculate the output.

The results are plotted in the table 2 calculating the mean square error and root mean square error.

The results are plotted in the table 6 calculating the mean square error and root mean square error.

STRUCTURE –PWL

SEGMENTS	Mean square error	Root mean square error
4	3.8598e-04	0.0196
8	3.2047e-05	0.0057
12	8.7347e-06	0.0030
16	5.0020e-06	0.0022

Table6. Results of PWL with different input segments

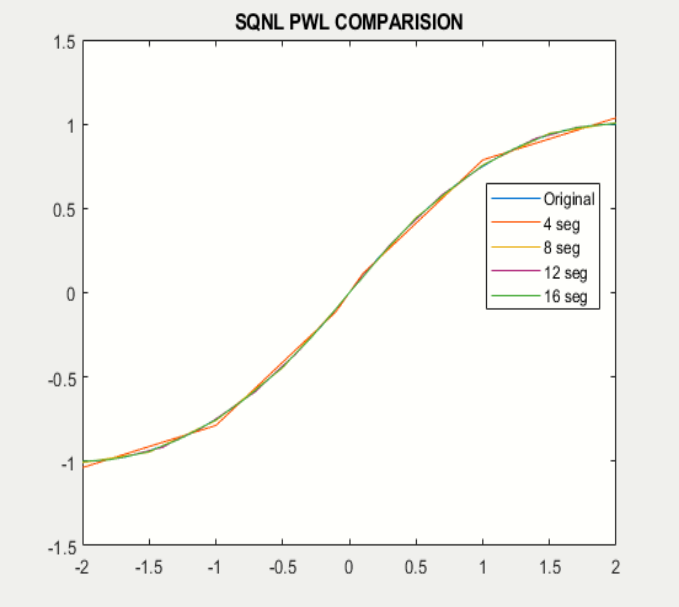

Figure 14. Comparison of different PWL segments with original SQNL function.

From the analysis it can be noted that error is minimum with large segments

D. Piecewise Nonlinear approximation method

The analysis is performed in the Mat lab considering different segments and results are shown below choosing the least error between the segments.
The input values are divided in to 4,8,12,16 segments and PWNL function is performed to calculate the output.

The results are plotted in the table 7 calculating the mean square error and root mean square error.

STRUCTURE –PWNL

SEGMENTS	Mean square error	Root mean square error
4	1.6035e-30	1.2663e-15
8	3.6006e-29	6.0005e-15
12	6.6983e-28	2.5881e-14
16	2.2857e-28	1.5119e-14

Table7. Results of PWNL with different input segments

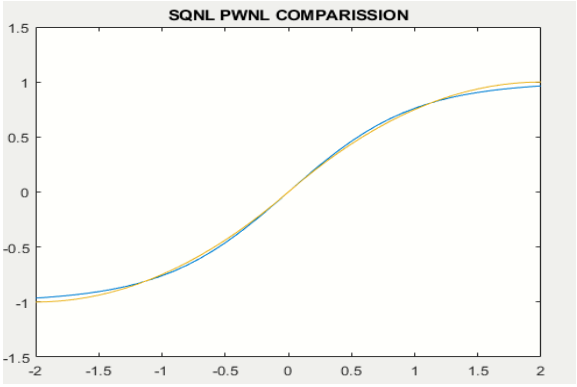

Figure 15. Comparison of different PWNL segments with original SQNL function

E. Full polynomial approximation method

The analysis is performed in the Mat lab with different ranges and results are shown below choosing the least error between the different inputs.
The Different input values are considered, and full polynomial function is performed to calculate the output.
The results are plotted in the table 8 calculating the mean square error and root mean square error.

Structure-FULL POLYNOMIAL

Ranges	Mean square error	Root mean square error
-2:0.1:2	3.7868e-04	0.0195
-5:0.1:5	1.5372e-04	0.0124
-9:0.1:9	8.5779e-05	0.0093

Table8. Results of Full polynomial with different inputs

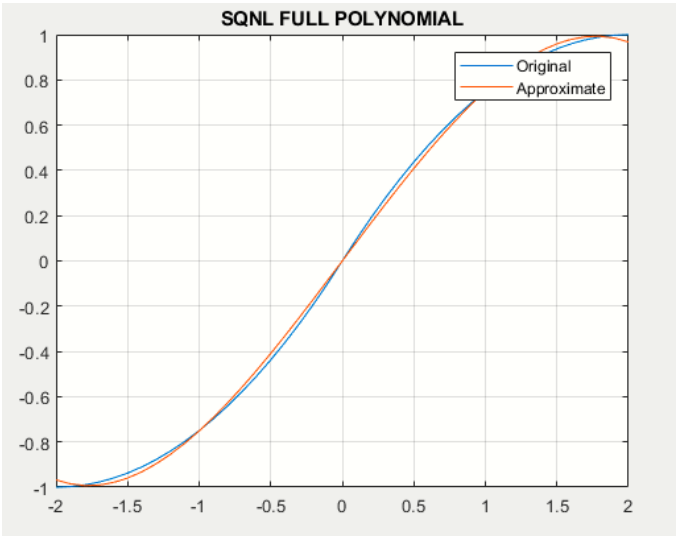

Figure 16. Comparison of Full polynomial with original SQNL function

From the error analysis of both tansig and SQNL activation functions it can be noted that SQNL activation function analysis have less error compared to tansig activation function.

When compared with all the above methods Full Polynomial approximation method have less error.But,as mentioned earlier it takes more time and require more resources because of the addition and multiplication factors in it and same implies for PWL and PWNL,Taylor series

Hard ware implementation is done for LUT, RALUT in Model sim.

VIII. HARD WARE IMPLENTATION

Hard Ware implementation is done in MODELSIM before testing on FPGA**.**

LUT/ROM Implementation

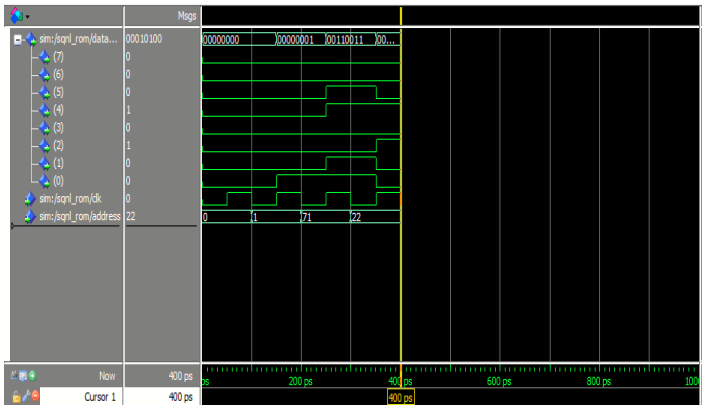

Figure 17. Implementation of LUT in Modelsim

RALUT IMPLEMENATION

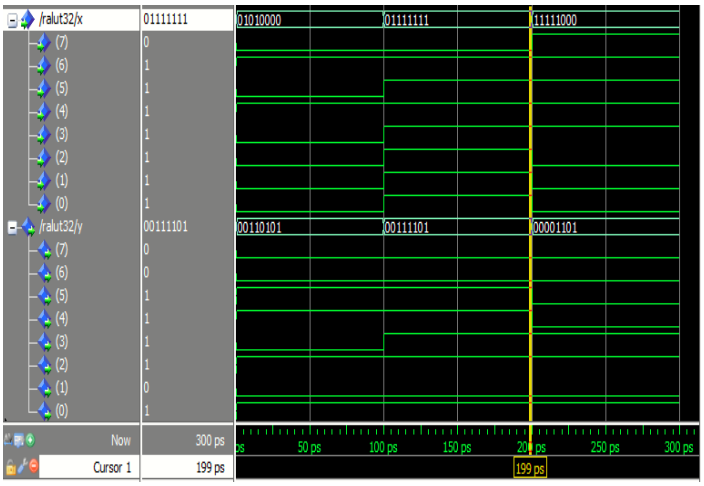

Figure 18. Implementation of RALUT in Modelsim

The resources required to implement on FPGA is shown

Table 9 gives the number of resources required for inputs with 8 segments

S/N	METHOD	ALM	REGISTETRS	DSP
1	LUT(case statement)	19	-	-
2	LUT(ROM)	28	8	-
3	RALUT	7	-	-
4	Piecewise linear (ax+b)	18	-	1
5	Piecewise Nonlinear	20	-	2
6	Full polynomial	28		3
7	Taylor series (3rd order)	2,380	-	2

Table 9. The number of resources required to implement 8 bits on FPGA

Table 10 gives the number of resources required for inputs with 16-bits

S/N	METHOD	ALM	REGISTETRS	DSP
1	LUT (case statement)	1762	-	-
2	LUT(ROM)	265	15	-
3	RALUT	21	-	-
4	Piecewise linear (ax+b)	36	-	1
5	Piecewise Nonlinear	36	-	2

Table 10. The number of resources required to implement 16 bits on FPGA

Table 11 gives the resources which have the least error, which is obtained from the last segment.

S/N	METHOD	ALM	REGISTETRS	DSP
1	RALUT-32 SEG	8	-	-
2	PWL-16 SEG	28	15	1
3	PWNL-16 SEG	28	-	2
4	Full polynomial	28	-	3

Table 11. The number of resources required to implement 8 bits with different segments on FPGA

IX. CONCLUSION

From the Section VI and VII it can be concluded that SQNL activation function has less error when compared to tangsig activation function by varying the input with different segments. RALUT would be considered to implement on FPGA as its architecture is simple and do not require many resources compared to other methods.This method would be cost effective as it doesn't contain any DSP block to implement. This paper gives the information of the methods that can be used to implement activation function on hardware and comparison between the 2 activation functions i.e., Tansig and SQNL. The results are tested for different inputs with different segments to get the best results.
The next step would be implementing it on DE1-SOC kit to verify the results obtained.

X. REFERENCES

[1] Morgado Dias, F. (2010). ***Sistemas Digitais – Princípios e Prática*****. Primeira Edição. FCA.**

[2] Stieglitz, T. and Meyer, J. (2006). Biomedical Microdevices for Neural Implants, BIOMEMS Microsystems. Vol. 16, pp. 71-137

[3] Tao Yang, Yadong Wei, Zhijun Tu, Haolun Zeng, Michel A. Kinsy, Nanning Zheng, Pengju Ren, Design Space Exploration of Neural Network Activation Function Circuits.

[4] Ashkan Hosseinzadeh Namin, Karl Leboeuf, Huapeng Wu, and Majid Ahmadi Department of Electrical and Computer Engineering, University of Windsor Windsor, Ontario N9B 3P4 Canada Email," Artificial Neural Networks Activation Function HDL coder".

[5] Karl Leboeuf, Ashkan Hosseinzadeh Namin, Roberto Muscedere, Huapeng Wu, and Majid Ahmadi Department of Electrical and Computer Engineering, University of Windsor 401 Sunset Avenue, Windsor, Ontario N9B 3P4 Canada,
High Speed VLSI Implementation of the Hyperbolic Tangent Sigmoid Function

[6] Darío Baptista and Fernando Morgado-Dias,
On the implementation of different hyberpoblic solutions in FPGA

[7] Thamer M.Jamel ,Ban M. Khammas,Implementation of a sigmoid activation function for nueral network using fpga