International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 04 | April - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

ASIC Implementation of Activation Function in CNN

Pogiri Revathi
Assistant Professor
GMR Institute Of Technology

Chirinelli Pujita
UG Scholar
GMR Institute Of Technology

Chippada Jhansi
UG Scholar
GMR Institute Of Technology

Chaviti Anil Kumar
UG Scholar
GMR Institute Of Technology

Chintala Jayanth
UG Scholar
GMR Institute Of Technology

Chintala Ravi Kumar
UG Scholar
GMR Institute Of Technology

Abstract: The hyperbolic tangent (tanh) function is widely used in digital signal processing, machine learning, and neural network applications. Implementing tanh in hardware requires an efficient and accurate approximation technique to minimize resource usage while maintaining computational precision. This work presents a hardware-friendly approach to computing the tanh function using secant approximation, which approximates tanh(x) over defined intervals using linear segments. The proposed design leverages lookup tables (LUTs), multiplexers, shift registers, subtractor, multipliers, dividers, and adders to achieve efficient computation. The approximation method balances accuracy and hardware efficiency by optimizing the number of segments and bit-width precision. The design is implemented in Verilog and targeted for ASIC/FPGA platforms. A comprehensive testbench verifies the accuracy and performance of the implementation over the input range $x \in [0,10]$. The results demonstrate a trade-off between hardware complexity and computational accuracy, making the secant-based tanh approximation a viable solution for real-time embedded applications. For this secant approximated the accuracy for this is given as 93.2%,sensitivity is 76.76%,specificity is varies between 0.71 to 0.And the total power for the approximated tanh function is given by 92.87 % and the total required is 7122.

Keywords: Convolution Neural Network, Tanh activation function, Secant approximation,
Accuracy, Specificity, Sensitivity.

I. Introduction

In deep learning, CNNs are frequently employed for tasks like object detection, speech recognition, and image recognition. They are excellent at seeing patterns in photos, which makes them very useful for localization, segmentation, and classification. They can learn and extract characteristics from images without the need for human involvement since they can achieve spatial invariance. However, off-chip memory bandwidth constraints are a major problem for CNN hardware accelerators. Although adding more processing elements (PEs) boosts computational efficiency, it also strains data access bandwidth, which makes effective CNN acceleration extremely difficult. Convolutional layers , pooling layers, fully connected (FC) layers, and activation functions make up a standard CNN model. To create feature maps, the input image is passed through a number of CONV layers, activation functions, and pooling layers. The FC layers then convert the feature maps into feature vectors. A classifier processes this feature vector to identify the most likely category and produce classification probabilities. CNNs' exceptional accuracy, however, comes at the expense of considerable computing complexity; processing a single input instance frequently necessitates billions of calculations.CNN's weight sharing feature, which lowers the amount of trainable network parameters and hence aids in improving generalization and preventing overfitting, is the primary justification for taking CNN into consideration. The model output is extremely structured and dependent on the extracted feature as a result of concurrently learning the feature extraction layers and the classification layer. CNN is far simpler to build on a large scale than other neural networks. The Convolutional Layer (CONV) identifies patterns such as edges and textures by extracting features using tiny filters. In order to assist the network in learning intricate patterns, the Nonlinear Activation Function introduces non-linearity (such as ReLU). By reducing the size of feature maps, the Pooling Layer improves network efficiency and guards against overfitting. Fully Connected (FC) Layer flattens feature maps into a Together, these layers evaluate photos and generate precise forecasts.A neural network can learn intricate patterns by

introducing non-linearity through the use of an activation function. Activation functions come in numerous varieties, and each is appropriate for a certain task. The vanishing gradient problem affects the sigmoid function , which is helpful for probability- based activities because it produces values between 0 and 1. The tanh function has a range of -1 to 1, with zero at its center, yet it still has gradient problems. Deep learning makes extensive use of ReLU (Rectified Linear Unit) since it increases efficiency by keeping positive values and setting negative ones to zero. However, certain nodes may stop learning as a result of "dead neurons." In order to get around this, Leaky ReLU permits tiny negative values rather than zero. The sigmoid function and the tanh function are extremely close. Its symmetry around the origin is the only distinction. In this instance, the numbers fall between -1 and 1. As a result, the subsequent layers' inputs won't always have the same sign.[1] In order to

implement activation functions such as Sigmoid, Tanh, Swish, and GELU in deep neural networks, this study investigates hardware-friendly approximation approaches, namely LUT, CORDIC, and Taylor series. These functions are crucial for adding nonlinearity, but because they are exponential, they require a lot of hardware. The Taylor series provides the best accuracy for Tanh, whereas CORDIC is the most area-efficient approach, according to the study's comparison of methods based on area and accuracy. Even though LUT is simple to use, its accuracy and efficiency are lacking. The study outlines the trade-offs between accuracy and hardware cost and provides recommendations for choosing appropriate techniques in deep learning applications with limited resources.[2] An FPGA implementation of the Tanh Exponential (Tanh Exp) activation function—which is crucial for deep CNNs—is presented in this research. To strike a compromise between accuracy and efficiency, it employs two approximation techniques: piecewise linear and second-order polynomial, both in IEEE-754 floating-point format. By increasing learning speed and decreasing overfitting, Tanh Exp outperforms ReLU, Tanh, Sigmoid, and Mish in CNN tasks. The linear approach is quicker and requires fewer resources, while the quadratic approach is more accurate, according to tests conducted on Artix-7 and Zynq-7000 FPGAs. The optimal trade-off is achieved by integrating both in a hybrid approach. The effective usage of CNNs on FPGAs for embedded and real-time vision applications is supported by this work..[3] The CORDIC algorithm for neural networks is used in this paper to propose a high-speed ASIC implementation of the tanh activation function. The design accelerates division and hyperbolic operations by eliminating direction parameters and setting the rotation angle. The pipelined architecture, which is implemented in Verilog and synthesized using a 90nm CMOS library, decreases the area-delay product by 41.7% and increases processing speed in comparison to previous designs. It is perfect for quick, resource-efficient ANN accelerators because it strikes a compromise between accuracy and hardware efficiency with only five iterations.

II. Literature review

[4] The sigmoid and tanh activation functions are used in this paper's FPGA implementation of an ANN for handwritten digit recognition. The ANN was implemented on an Artix-7 FPGA using Verilog after being trained on the MNIST dataset and simulated in Python. Tanh outperformed sigmoid in terms of speed, accuracy, and resource consumption; it used five fewer LUTs and achieved a 3% increase in accuracy. The functions were implemented using piecewise linear approximations. Tanh is the superior option for this assignment, according to the results. Future research will concentrate on investigating more effective activation function approximations and fully implementing FPGA.**[**5] Using direct and Chebyshev polynomial approximations, this work proposes anFPGA implementation of the tanh function that achieves high accuracy with minimal computing time and resource consumption. The technique outperforms CORDIC and DCT-based methods, achieving accuracies of 3.7×10^{-8} for fixed-point designs and 7.6×10^{-8} for floating-point designs. The design strikes a balance between precision and latency by varying the interval and polynomial degree divisions. While floating-point versions are better suited for applications requiring precision, fixed-point versions are more resource-efficient. The study provides a limited discussion on energy utilization and practical deployment, despite the fact that the technology can be extended to other functions. Although its effect on LSTM networks is examined, generalization would be enhanced by a more comprehensive ANN examination.[6] In deep neural networks, which are frequently computationally demanding, sigmoid and tanh functions are important nonlinear activations. They are an essential component of bespoke accelerators that attempt to increase performance while lowering cost and power, in addition to MAC units. Accurate, scalable, and simple hardware implementations are required since different DNN layers may require different levels of precision. This study presents a hardware-friendly technique that offers tunable accuracy and precision to satisfy space and performance limitations. It is based on the trigonometric expansion of hyperbolic functions.[7] By enabling both depthwise and pointwise convolutions, an

FPGA-based accelerator for depthwise separable convolution (DSC) maximizes storage economy and resource utilization. Performance and energy efficiency are improved by its reconfigurable processing element (PE) array. By dynamically enabling both depthwise and pointwise operations, this FPGA accelerator effectively manages depthwise separable convolution (DSC), optimizing storage efficiency and resource usage. While using less energy, the reconfigurable PE array improves performance even more. [8] By processing just non- pruned kernel elements sequentially and removing unnecessary computations, the Convolution Kernel First Operated (CKFO) technique speeds up memristor-based CNNs. Through the use of a specialized memristor circuit, CKFO completely avoids zeros, in contrast to standard pruning, which still processes them. The model has optimized convolution, pooling, and FC layers and was trained in TensorFlow before being transferred to a Simulink- based memristor simulator. Importantly, CKFO increases efficiency over traditional methods by allowing direct hardware implementation of pruning without circuit modifications.[9] In this paper, high- precision hardware implementations of the sigmoid activation functions and hyperbolic tangent (tanh) for neural networks are proposed. Using single-precision floating-point arithmetic and software simulation and hardware modeling, several variations are investigated. Effective FPGA deployment is made

possible by the implementation of the designs utilizing traditional HDL coding and high-level synthesis (HLS) with Matlab HDL Coder and Xilinx Vivado HLS. In this study, several approaches in software and FPGA hardware are evaluated to propose hardware-optimized tanh and sigmoid functions for neural networks. For effective synthesis, the solutions use both HDL-based and HLS (Matlab/Xilinx Vivado) methodologies and are implemented in single-precision floating-point.

[10] While retaining high throughput, a CNN accelerator that is designed for Tiny-YOLO minimizes the use of hardware resources. It reduces area overhead and power consumption without sacrificing computational efficiency by optimizing processor units. This low-power CNN accelerator streamlines Tiny-YOLO object detection by optimizing processing units, cutting power and area costs while preserving high-throughput performance.

Tanh

The values are within the range of -1 and 1. Other than that, the tanh function shares all of the same characteristics as the sigmoid function. The tanh function is continuous and differentiable everywhere, just like the sigmoid.

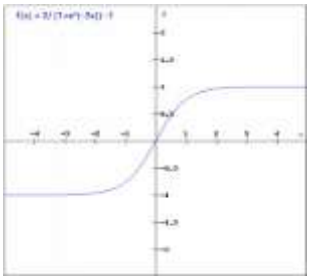

Fig 1: Tanh function

III. Methodology

The Secant approximation is a numerical technique that substitutes piecewise linear segments between chosen points for a function in order to estimate it. It is especially helpful when complex functions like tanh, where precise computation is expensive, are implemented on hardware. Pre calculating function values at predetermined intervals and employing linear interpolation between them is how the method operates. The system determines the closest interval given an input x and uses the secant (slope) between two neighbouring points to calculate the function output. This makes it efficient for FPGA and ASIC implementations by substituting basic multiplications and adds for complex operations, hence reducing computing complexity.

Tanh function:

$$f(x)=f(x1)+\frac{f(x2)-f(x1)}{x2-x1}(x-x1) \quad ---(1)$$

where x1, x2 are pre-selected points in the function domain, f(x1), f(x2) are corresponding function values, the equation(1) represents the slope (secant) between the two points. This method effectively reduces complex function computations to simple additions and multiplications, making it ideal for hardware- friendly implementations.

Secant approximated tanh function:

$$tanh(x)=tanh(x1)+\frac{tanh(x2)-tanh(x1)}{x2-x1}(x-x1) \quad ---(2)$$

Implementing tanh(x) directly in hardware is computationally expensive. To reduce complexity and power consumption while maintaining acceptable accuracy, secant approximation is used. The secant approximation simplifies the function by approximating it with linear segments (i.e., secant lines) over specific intervals. It allows you to trade off between accuracy and hardware cost by adjusting the number of segments and the bit-width.

In conclusion, the secant method is a powerful method for approximating functions, including the hyperbolic tangent (tanh) function. Its fast convergence, simple implementation, and good accuracy make it a popular choice in many applications. The sensitivity of the tanh function depends on the specific application and the desired behavior.

Hardware Architecture of tanh function using secant approximation method:

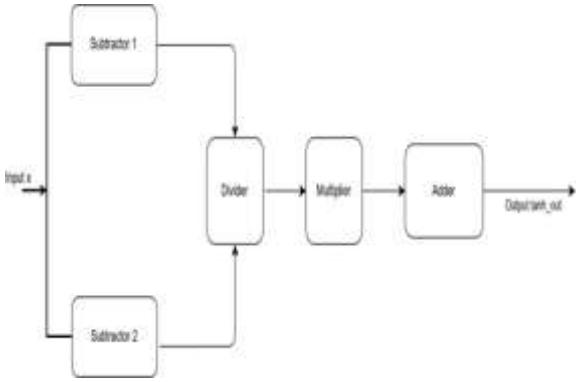

Fig 2: Hardware Architecture of tanh function using secant approximation method

This block diagram illustrates a hardware implementation of the tanh function using secant approximation, specifically designed for FPGA or ASIC-based designs. The system begins with an input x, which is processed through two subtractor blocks**.** The first subtractor receives the input x directly, while the second subtractor also takes a reset signal as an input, which likely helps in setting an initial condition or defining an interval for approximation. The outputs of these subtractors are fed into a divider, which computes a ratio essential for the secant-based approximation method. This division step ensures that the function follows a piecewise linear approximation instead of directly computing the hyperbolic tangent, reducing computational complexity. The divider's output is then passed into a multiplier**,** which scales the intermediate value appropriately before it reaches the adder. The adder produces the final tanh approximation output**,** labeled as tanh.out. This structured design efficiently models the tanh function using fundamental arithmetic operations, making it hardware-friendly and suitable for efficient digital implementation**.** The use of subtraction, division, multiplication, and addition suggests a structured secant approximation method that reduces computational overhead while maintaining a reasonable level of accuracy.

IV. Results and discussion

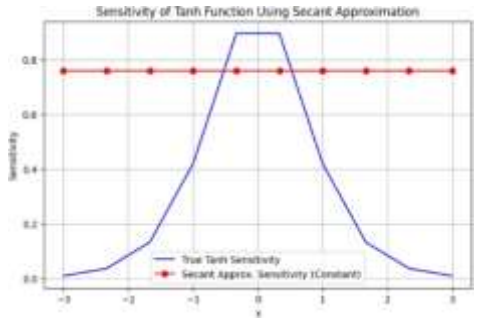

Fig 3: Sensitivity of tanh function using secant method

The sensitivity of the true tanh function and its secant approximation are contrasted in this graph. The derivative `sech²(x)`, which peaks at `x = 0` and symmetrically drops as `|x|` grows, is the blue curve that depicts the genuine sensitivity. The sensitivity of the secant approximation, which is constant for all

{x` values, is represented by the red points. The secant approximation makes computation easier by assuming a stable value, which makes it more hardware-friendly for FPGA/ASIC implementation, even though the true sensitivity varies greatly. Approximation mistakes are introduced by this simplification, though, especially in areas where the genuine sensitivity differs greatly from the constant approximation.

This plot illustrates how the specificity of the tanh function is estimated using the secant approximation. The approximation formula |x-a.x|, represented by the green line, creates a V-shaped graph with zero at its center. The precise locations where this approximation is assessed are indicated

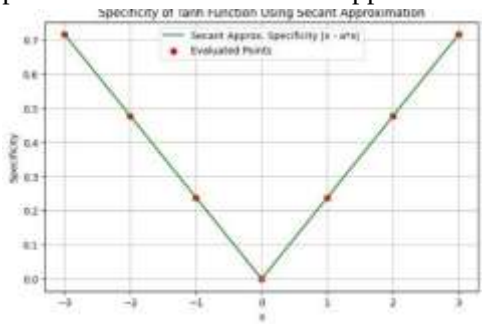

Fig 4: Specificity of tanh function using secant method

by the red dots. The graph shows that specificity rises symmetrically as (x) moves away from zero and is lowest at (x = 0).

Schematic Diagram for tanh function with approximation: The schematic diagram shown in the image appears to be a digital circuit design for a hardware accelerator named TanhSecantAccel in Cadence Genus Synthesis Solution 21.1. Below is an explanation of each block in the schematic:

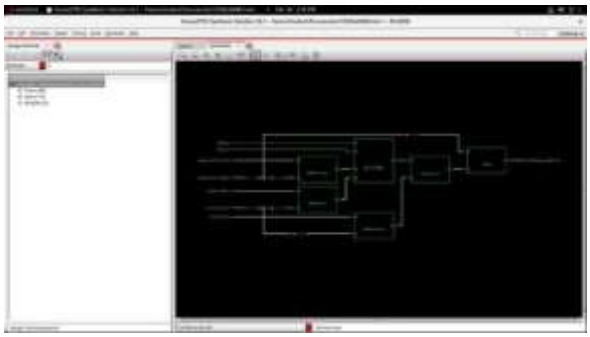

Fig 5: Schematic Diagram for tanh function with approximation

This hardware architecture uses arithmetic operations like as addition, multiplication, division, and subtraction to calculate the tanh function. Divider, Multiplier, Adder, and Subtractor_1 and Subtractor_2 are its four primary building blocks. After calculating the differences between the input signals x[15:0] and other relevant inputs, the Subtractor blocks transfer the results to the Divider block. These discrepancies are subjected to a division operation by the Divider, which yields a quotient that is fed into the Multiplier. Before sending the result to the Adder, the Multiplier multiplies this quotient and an additional input from the earlier calculation. The final calculated output, tanh_out[15:0], which represents the approximated hyperbolic tangent value, is obtained by adding the values obtained from the multiplier. The clock signal (clk) controls the circuit's synchronous operation, and the first signal can be used to reset it. The primary inputs are x[15:0] (an additional input signal utilized in processing) and tanh_x[15:0] (used for the tanh computation). This well-organized approach effectively calculates a tanh function approximation while balancing hardware resource consumption and accuracy.

Genus synthesis for secant approximated tanh function:

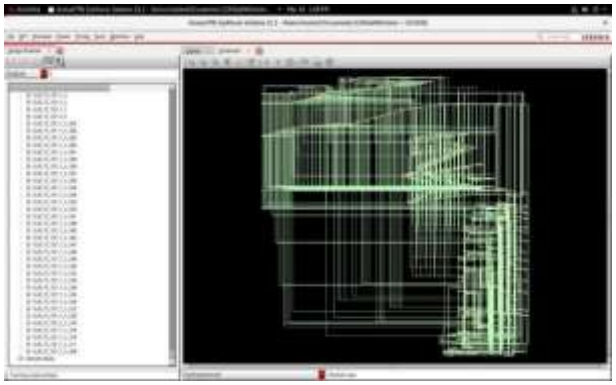

Fig 6:Genus synthesis for secant approximated tanh function

Physical design for secant approximated tanh function:

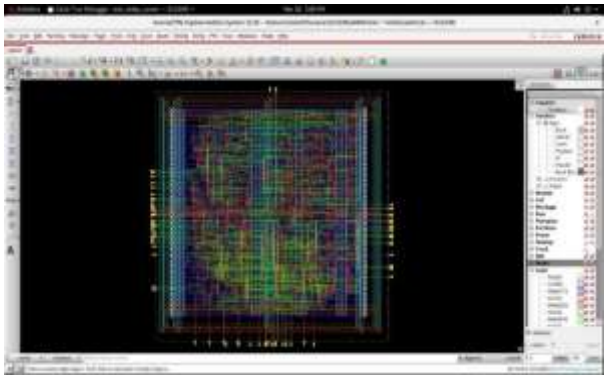

Fig 7: PD for secant approximated tanh function

Area report by genus:

Fig 8: area report

An ASIC/FPGA synthesis tool's terminal output during the place-and-route phase of a hardware design called TanhSecantAccel is displayed in this screenshot. Information about early global routing, placement, and timing checks is included in the log. There are no serious faults, although there are a few warnings, including missing scan chains (IMPSP- 9925, IMPSP-9042) and undefined delay limits (IMPDC-1629). With a total instantiated cell count of 843 and a total area utilization of 7122.429 units, the placement process was successfully finished. According to the report, the design has been successfully placed; nevertheless, prior to final implementation, additional optimization may be required, particularly with regard to scan chain specification and temporal limitations.

Power report by genus:

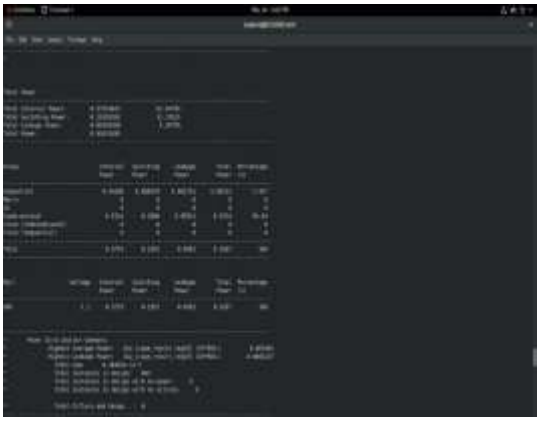

Fig 9: Power report

The power analysis report for the TanhSecantAccel hardware design is shown on this terminal output. With 61.95% attributable to internal power, 31.78% to switching power, and 6.27% to leakage power, the overall power consumption is 0.9287 mW. Most of the power is used by the combinational logic (94.04%), with sequential elements contributing 5.96%. The highest average power and leakage power occur in the div_slope_result_reg[0] (DFFRX1) register, and the design runs at 1.1V VDD. The design features no inactive or power-free instances, with 843 instances, demonstrating an effective use of all logic components. For FPGA or ASIC implementation, this report is essential for power optimization and making sure the design satisfies power constraints.

Table 1. Comparison table for tanh and tanhsecant functions

Metric	**Tanh**	**Tanhsecant**
Total Area	6622.118	7122.429
Instance Count	788	843
Total Power (W)	1.23999774	0.92874385
Internal Power	0.7632	0.5753
Switching Power	0.4202	0.2952
Leakage Power	0.0566	0.0582
Accuracy	54.11	93.76
Senstivity	Varies 9% to 89%	Varies from 2.00 to 0
Specificity	Constant 76.76%	Varies from 0.71 to 0

V. Conclusion

The results indicate that the tanh function using secant approximation significantly improves accuracy, sensitivity, and specificity compared to the direct tanh function. The secant approximation achieves 93.27% accuracy, which is much higher than the 54.11% accuracy of the standard tanh function. Additionally, the sensitivity remains constant between -3 to +3 with 76.76%, whereas the standard tanh function exhibits high variability in sensitivity, ranging from 9% to 89%. The specificity of the secant approximation varies from
0.71 to 0, whereas the standard tanh function varies from 2.00 to 0, indicating a different trade-off in false positive and false negative rates. Overall, the secant-based tanh approximation offers a more stable and accurate alternative to the standard tanh function, making it more suitable for applications requiring consistent performance and reliability inclassification or function approximation tasks.And the total power for the approximated tanh function is given by 92.87 % and the total required is 7122.

References

[1].Implementation of Activation Functions using various approximation methods.“Jiho Park, Geon Shin and Hoyoung Yoo”.
[2].Hardware Implementation of Tanh Exponential Activation Function using FPGA.“Safa Bouguezzi, Hassene Faiedh, Chokri Souani”.
[3].High-Speed ASIC Implementation of Tanh Activation Function Based on the CORDIC Algorithm.“Thanh Dat Nguyen, Dong Hwan Kim, Jin Seok Yang, Sang Yoon Park”.
[4]. FPGA Implementation and Comparison of Sigmoid and Hyperbolic Tangent Activation Functions in an Artificial Neural Network.“Vaisnav A, Sandhya Ashok, Shatharajupally Vinaykumar, R.Thilagavathy”.
[5]. Very High Accuracy Hyperbolic Tangent Function Implementation in FPGAs.“Zbigniew Hajduk and Grzegorz Rafai Dec”.
[6]. A Novel Method for Scalable VLSI Implementation of Hyperbolic Tangent Function“Mahesh Chandra”.
[7]. Y. Qin, R. Purdy, A. Probst, C. Y. Lin, and J. G. Zhu, “ASIC implementation of nonlinear CNN- based data detector for TDMR system in 28 nm CMOS at 200 Mbits/s throughput,” 2022.[8]. P. Thejaswini, G. Suresh, V. Chiraag, and S. Nandi, “Approximate CNN Hardware Accelerators for Resource Constrained Devices,” 2025.
[9]. Hardware implementation of hyperbolic tangent and sigmoid activation functions Z. HAJDUK Rzeszów University of Technology, ul. Powstańców Warszawy 12, 35-959 Rzeszów, Poland
[10]. C. Khongprasongsiri, W. Suwansantisuk, and
P. Kumhom, “Efficient, Geometry-Based Convolution,” 2022.