IJAREEIE

e-ISSN: 2278-8875
p-ISSN: 2320-3765

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

Volume 14, Issue 3, March 2025

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.807

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering (IJAREEIE)

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.807 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 14, Issue 3, March 2025 ||

| DOI:10.15662/IJAREEIE.2025.1403016 |

VLSI Implementation of Approximate Softmax Function for Neural Network

Dr. J. Kirubakaran, S. Dharshini, R. Jilpata Mary, S. Gunasree

Associate Professor, Dept. of ECE, Muthayammal Engineering College, Namakkal, Tamil Nadu, India

UG Final Year Students, Dept. of ECE, Muthayammal Engineering College, Namakkal, Tamil Nadu, India

ABSTRACT: The Softmax activation function is a crucial component in deep learning architectures, particularly in classification tasks. It transforms a vector of values into a probability distribution, enabling models to make decisions based on probabilities. However, the computational complexity of the Softmax function poses challenges in hardware implementations, leading to high latency and power consumption. To address these issues, this paper presents an efficient hardware architecture for the Softmax function, using a pipelined divider for faster and more accurate floating-point operations. The key focus of this work is on optimizing the performance of the Softmax function in terms of speed, accuracy, and power consumption in FPGA-based implementations. The proposed architecture reduces the latency by leveraging the pipelining technique, which allows for faster computation by overlapping the stages of the division process. The division of two floating-point numbers, a critical operation in the Softmax function, is optimized to minimize hardware complexity while ensuring high precision. The architecture is designed to be energy-efficient, aiming to lower power consumption while maintaining the required performance levels for deep neural networks. The Softmax function's hardware design is described in Verilog, and the proposed architecture is implemented on a Xilinx Spartan 3 FPGA platform. Timing performance is analyzed using the Xilinx Timing Analyzer, showing a significant reduction in execution time compared to traditional implementations. This work demonstrates the potential for optimizing deep learning applications in hardware, offering a balance between speed, accuracy, and energy efficiency.

KEYWORDS: Softmax activation function, pipelined divider, floating-point division, hardware design, FPGA, Verilog, Xilinx Spartan 3, power consumption, latency reduction, deep learning, classification tasks, timing analysis, VLSI architecture.

I. INTRODUCTION

In modern deep learning, the Softmax activation function plays a vital role in classification tasks by converting the raw output of a neural network into a probability distribution. This transformation is crucial for determining the likelihood of different classes in a multi-class classification problem.[3] The Softmax function takes a vector of real-valued inputs, computes the exponentials of each element, and normalizes them to produce outputs between 0 and 1, representing probabilities. Despite its importance, the computational complexity of the Softmax function can create significant challenges when implementing it in hardware.[4]The Softmax function involves both exponential and division operations, which can be computationally expensive, especially in hardware implementations. When performing these operations on floating-point numbers, the demands on precision, accuracy, and performance increase, leading to concerns regarding power consumption, latency, and hardware efficiency.[1] Furthermore, the division of floating-point numbers, a core operation in the Softmax function, is typically time-consuming and resource-intensive, limiting the overall performance of deep learning models, especially in resource-constrained environments like embedded systems or edge devices.In this context, the objective of optimizing the Softmax function's hardware implementation is crucial for improving the overall efficiency of deep learning algorithms, particularly in real-time applications where low latency and minimal power consumption are essential. To address these challenges, pipelining techniques are employed to optimize the division operations within the Softmax function.[15] Pipelining allows for parallel processing of multiple stages of computation, reducing the overall time required to compute each output while improving throughput. This paper proposes a novel hardware architecture for implementing the Softmax function using a pipelined divider to efficiently divide floating-point numbers. The design is implemented in Verilog, a hardware description language, and synthesized on a Xilinx Spartan 3 FPGA platform. The FPGA-based implementation is ideal for evaluating the performance of the proposed architecture due to its reconfigurability and ability to handle complex computations efficiently.[12] The timing performance of the architecture is evaluated using the Xilinx Timing Analyzer, and the results demonstrate significant improvements in execution speed and power efficiency compared to traditional implementations of the Softmax function in hardware.The contributions of this work are twofold: (1) an efficient hardware design for the Softmax function that

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.807 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 14, Issue 3, March 2025 ||

| DOI:10.15662/IJAREEIE.2025.1403016 |

minimizes latency and power consumption, and (2) a detailed analysis of the performance benefits achieved by implementing pipelined division for floating-point operations. This approach not only enhances the speed and accuracy of the Softmax function but also provides an optimized solution for hardware acceleration in deep learning applications.

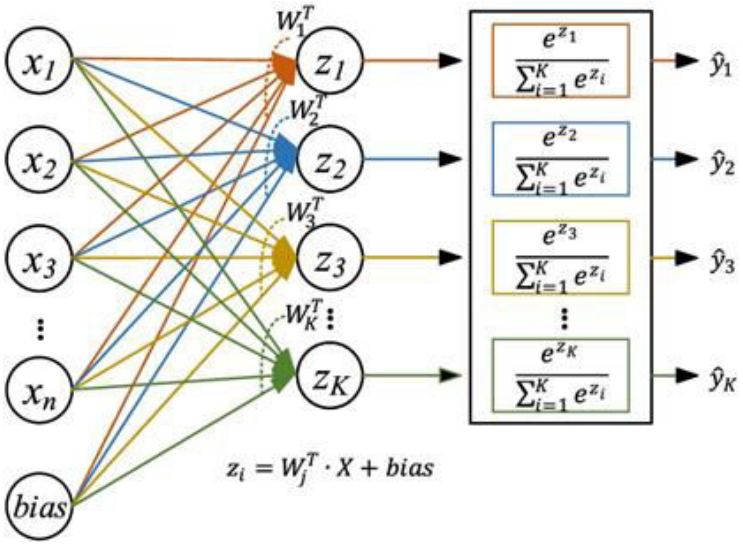


figure 1. Softmax Function

II. RELATED WORK

Ma, Y., Cao, Y., Vrudhula, S., and Seo, J.-S. This paper presents an optimization strategy for convolution operations to accelerate deep neural networks (DNNs) on FPGA platforms. The study introduces efficient hardware implementations that minimize memory access overhead and improve computational throughput. By leveraging parallel processing and optimized dataflow architectures, the proposed method significantly reduces inference latency. The research also compares the performance of FPGA-based implementations against traditional CPU and GPU-based models, demonstrating a substantial improvement in power efficiency and speed. This work is particularly relevant for real-time AI applications where low-latency processing is critical. [1]

Wang, S., et al. The authors propose C-LSTM, an optimized FPGA implementation of Long Short-Term Memory (LSTM) networks using structured compression techniques. The study addresses the high computational cost of LSTMs by introducing weight pruning and low-rank approximations, significantly reducing resource utilization while maintaining accuracy. The paper evaluates C-LSTM on various sequence modeling tasks, demonstrating substantial improvements in execution speed and energy efficiency. The FPGA-based approach enables real-time processing of sequential data, making it suitable for applications in speech recognition, financial forecasting, and natural language processing (NLP), where power-efficient, high-performance recurrent networks are required. [2]

Sun, Q., et al. This research focuses on designing a high-speed softmax function architecture for deep learning accelerators using a basicsplit approach. The study highlights the challenges of implementing softmax efficiently on FPGAs due to its exponential computation and normalization steps. By optimizing numerical approximations and parallel execution, the proposed VLSI architecture achieves low-latency and high-throughput performance. Experimental results show that this design outperforms conventional softmax implementations in speed and resource efficiency. The proposed method is particularly beneficial for real-time AI inference tasks, such as image classification and speech processing, where fast softmax computation is essential. [3]

Li, Z., Li, H., Jiang, X., Chen, B., Zhang, Y., and Du, G. This study presents an efficient FPGA implementation of the softmax function tailored for deep neural network (DNN) applications. The authors introduce a resource-efficient approach that reduces the complexity of exponentiation and division operations, key bottlenecks in softmax computation. By leveraging hardware-friendly approximations and optimized pipelining, the proposed design enhances processing speed while maintaining numerical accuracy. Experimental results indicate significant reductions in power consumption

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.807 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 14, Issue 3, March 2025 ||

| DOI:10.15662/IJAREEIE.2025.1403016 |

and execution time. This work is particularly useful for edge AI applications, where real-time inference and low-power consumption are critical for performance optimization. [4]

Liu, W., Qian, L., Wang, C., Jiang, H., Han, J., and Lombardi, F. The paper explores approximate radix-4 Booth multipliers designed for error-tolerant computing applications. Traditional multipliers demand high energy and computational resources, making them inefficient for low-power AI hardware. This study introduces approximation techniques that trade minimal accuracy loss for significant energy savings. The proposed multipliers are evaluated in terms of power, delay, and accuracy, showing promising results for machine learning applications that can tolerate minor computational errors. These designs are especially relevant for deep learning accelerators and edge computing devices, where power efficiency is a primary concern without significantly compromising model accuracy. [5]

Jiang, H., Santiago, F. J. H., Mo, H., Liu, L., and Han, J. This comprehensive survey reviews approximate arithmetic circuits, discussing their impact on energy efficiency and computational speed in machine learning applications. The paper categorizes various approximation techniques, including voltage scaling, logic simplifications, and stochastic computing, analyzing their trade-offs between accuracy and efficiency. It also explores recent applications in neural network accelerators and edge AI systems. The findings suggest that approximate arithmetic methods can significantly reduce power consumption while maintaining acceptable performance levels. This survey serves as a valuable reference for researchers developing low-power, high-efficiency AI hardware accelerators. [6]

Yin, P., Wang, C., Waris, H., Liu, W., Han, Y., and Lombardi, F. This study investigates energy-efficient dynamic range approximate logarithmic multipliers, designed to optimize AI computations in resource-constrained environments. The authors introduce an approximation method that simplifies logarithmic multiplication while preserving sufficient computational accuracy. Experimental evaluations show notable reductions in power consumption and processing latency. These multipliers are particularly useful in neural network accelerators, where large-scale matrix operations can benefit from hardware-level optimizations. The proposed design is well-suited for battery-powered AI systems and edge computing devices, where maintaining a balance between efficiency and accuracy is crucial for real-time applications. [7]

Liu, W., Xu, J., Wang, D., Wang, C., Montuschi, P., and Lombardi, F. This paper presents an analysis of approximate logarithmic multipliers, emphasizing their potential for low-power, error-tolerant applications. The study proposes various approximation techniques that reduce hardware complexity while ensuring computational efficiency. Results indicate that these multipliers can achieve up to 50% energy savings compared to traditional multipliers, making them ideal for AI accelerators. The authors also evaluate the trade-offs between accuracy and efficiency, demonstrating that minor computational errors have minimal impact on overall deep learning model performance. The research contributes to the development of power-efficient AI hardware architectures. [8]

Murillo, R., Del Barrio Garcia, A. A., Botella, G., Kim, M. S., Kim, H., and Bagherzadeh, N. The paper introduces PLAM (Posit Logarithm-Approximate Multiplier), a novel approach to improving AI computation efficiency. The study explores posit arithmetic, which provides better numerical precision than traditional floating-point representations. By integrating approximate logarithmic multipliers, the proposed method significantly enhances energy efficiency while maintaining reliable computational accuracy. Experimental results demonstrate substantial improvements in FPGA-based deep learning accelerators, reducing power consumption and execution time. This research is particularly relevant for edge AI applications and embedded systems, where power-efficient neural network computations are essential for real-time decision-making. [9]

Pilipovic, R., and Bulic, P. This work examines the design of logarithmic multipliers using radix-4 Booth encoding, a technique that enhances computational efficiency in hardware-based AI accelerators. The authors propose a novel approach to logarithmic multiplication that reduces circuit complexity while preserving accuracy. By leveraging radix-4 Booth encoding, the study achieves a balance between power efficiency and computational speed. Results indicate significant improvements in processing latency and energy consumption compared to conventional multipliers. The proposed method is ideal for AI inference tasks, where real-time computation and hardware optimization are crucial for performance and scalability. [10]

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.807 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 14, Issue 3, March 2025 ||

| DOI:10.15662/IJAREEIE.2025.1403016 |

III. PROPOSED METHEDOLOGY

The **Single Precision Floating-Point Representation:**
The proposed methodology utilizes single precision floating-point representation (32-bit) for all data handled by the architecture. This allows for efficient and accurate computation, ensuring that the Softmax function's operations, including division, exponential, multiplication, and addition, maintain the necessary precision without excessive resource usage.[6] The 32-bit floating-point format consists of one bit for the sign, eight bits for the exponent, and 23 bits for the mantissa, providing a good balance between precision and computational efficiency. This format is widely supported in hardware and offers sufficient accuracy for deep learning tasks.

Exponential Block (Mega Function):
The exponential function is a key operation in the Softmax function, where each input is exponentiated to compute its relative importance. The exponential operation is implemented using a "mega function" block, which is optimized for hardware implementation. [10]This block efficiently computes the exponential of the input using lookup tables, approximations, or hardware-accelerated exponential circuits. By employing this specialized mega function block, the computation speed is increased while maintaining the necessary precision. The exponential function's hardware implementation reduces the need for software-based calculation, thereby enhancing performance in hardware-accelerated systems.

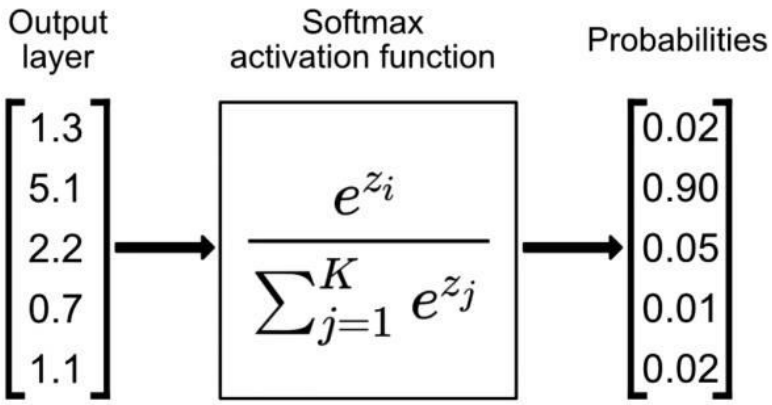


figure 2. Softmax Activation Function

Division Block (Mega Function):
Division is another critical operation within the Softmax function, especially when normalizing the exponential values to create the probability distribution. A dedicated division "mega function" block is used to handle this operation efficiently. This block can be optimized for speed by employing pipelining techniques, where multiple division operations are processed in parallel across different pipeline stages.[9] By using high-performance hardware dividers, the division operation is handled faster than traditional software-based methods, reducing the latency of the Softmax computation and allowing real-time processing of classification tasks.

Pipelined Architecture for Division:
To improve the overall throughput and speed of the Softmax function implementation, a pipelined architecture is employed in the division block. Pipelining allows multiple stages of the division operation to be processed concurrently, thereby reducing the time required to perform each division. Each stage of the pipeline handles a part of the division operation, and as a result, multiple divisions can be processed simultaneously, leading to a significant speedup in the Softmax computation. This architecture is crucial for ensuring the system can meet the timing requirements of deep learning applications in hardware.

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.807 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 14, Issue 3, March 2025 ||

| DOI:10.15662/IJAREEIE.2025.1403016 |

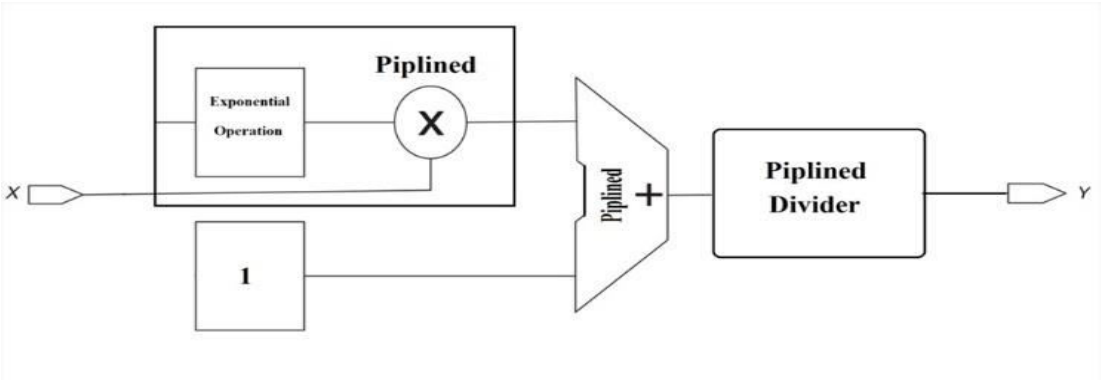


figure 3. System Architecture

Multiplication and Addition Units: Besides the division and exponential functions, the Softmax function also involves multiplication and addition operations. These calculations are implemented using dedicated hardware blocks optimized for floating-point multiplication and addition.[8] The multiplication block handles the scaling of input values by the computed exponentials, and the addition block sums the results to normalize them. These operations are crucial in the Softmax function to combine the exponential values and produce the final probability distribution. By using optimized multiplier and adder circuits, the system can handle these operations with minimal delay.

Memory Management and Lookup Tables: To enhance the performance of the exponential and division blocks, lookup tables (LUTs) are employed for frequently accessed computations. These tables store precomputed values of exponential functions and division results for common inputs, thereby reducing the need to compute these functions on-the-fly.[7] The use of memory blocks and efficient caching ensures that the architecture operates at high speeds, reducing the computational load and improving overall throughput. This method leverages the FPGA's internal memory resources, allowing for fast access to these lookup tables and significantly speeding up calculations.

Optimized Power Consumption: Power efficiency is a critical consideration when designing hardware for deep learning algorithms. To minimize power consumption, the architecture employs techniques such as clock gating, voltage scaling, and optimized resource usage.[4] By selectively activating and deactivating parts of the circuit based on the current computation, power usage is reduced without compromising performance. Additionally, the use of efficient floating-point units and memory access patterns further contributes to lowering power consumption while maintaining the required calculation accuracy. These optimizations are essential for deploying the architecture in resource-constrained embedded systems or edge devices.

FPGA Implementation on Xilinx Spartan 3: The final implementation of the proposed Softmax architecture is carried out on a Xilinx Spartan 3 FPGA, which is a low-cost, high-performance platform suitable for hardware acceleration tasks. The FPGA's reconfigurable logic allows for efficient implementation of the exponential, division, multiplication, and addition blocks, as well as pipelined architectures.[1] The system is described in Verilog, enabling easy integration and testing of the architecture. After implementation, the system's performance is evaluated using the Xilinx Timing Analyzer, which provides insights into the timing characteristics, latency, and throughput of the design. The FPGA implementation ensures that the system can be easily tested and optimized for real-world applications.

IV. TECHNOLOGIES USED

B **Single Precision Floating-Point Representation:**
The proposed methodology utilizes single precision floating-point representation (32-bit) for all data handled by the architecture. This allows for efficient and accurate computation, ensuring that the Softmax function's operations, including division, exponential, multiplication, and addition, maintain the necessary precision without excessive resource usage. The 32-bit floating-point format consists of one bit for the sign, eight bits for the exponent, and 23 bits for the mantissa, providing a good balance between precision and computational efficiency. This format is widely supported in hardware and offers sufficient accuracy for deep learning tasks.[2]

Exponential Block (Mega Function):
The exponential function is a key operation in the Softmax function, where each input is exponentiated to compute its relative importance. The exponential operation is implemented using a "mega function" block, which is optimized for

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.807 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 14, Issue 3, March 2025 ||

| DOI:10.15662/IJAREEIE.2025.1403016 |

hardware implementation. This block efficiently computes the exponential of the input using lookup tables, approximations, or hardware-accelerated exponential circuits.[5] By employing this specialized mega function block, the computation speed is increased while maintaining the necessary precision. The exponential function's hardware implementation reduces the need for software-based calculation, thereby enhancing performance in hardware-accelerated systems.

Division Block (Mega Function):
Division is another critical operation within the Softmax function, especially when normalizing the exponential values to create the probability distribution. A dedicated division "mega function" block is used to handle this operation efficiently. This block can be optimized for speed by employing pipelining techniques, where multiple division operations are processed in parallel across different pipeline stages. By using high-performance hardware dividers, the division operation is handled faster than traditional software-based methods, reducing the latency of the Softmax computation and allowing real-time processing of classification tasks.

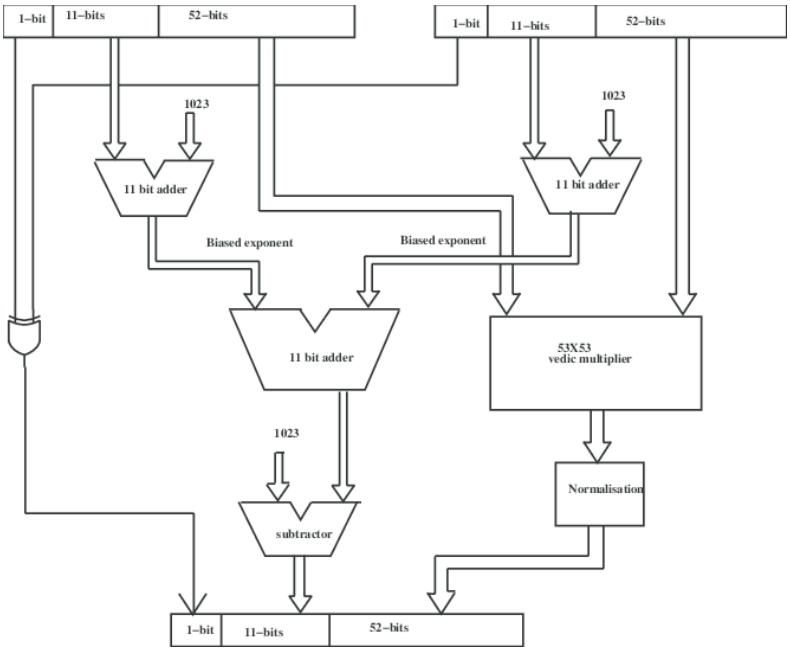


figure 4. IEEE 754 divider

Pipelined Architecture for Division:
To improve the overall throughput and speed of the Softmax function implementation, a pipelined architecture is employed in the division block. Pipelining allows multiple stages of the division operation to be processed concurrently, thereby reducing the time required to perform each division. Each stage of the pipeline handles a part of the division operation, and as a result, multiple divisions can be processed simultaneously, leading to a significant speedup in the Softmax computation. This architecture is crucial for ensuring the system can meet the timing requirements of deep learning applications in hardware.

Multiplication and Addition Units: Besides the division and exponential functions, the Softmax function also involves multiplication and addition operations. These calculations are implemented using dedicated hardware blocks optimized for floating-point multiplication and addition. The multiplication block handles the scaling of input values by the computed exponentials, and the addition block sums the results to normalize them. These operations are crucial in the Softmax function to combine the exponential values and produce the final probability distribution. By using optimized multiplier and adder circuits, the system can handle these operations with minimal delay.

Memory Management and Lookup Tables:
To enhance the performance of the exponential and division blocks, lookup tables (LUTs) are employed for frequently accessed computations. These tables store precomputed values of exponential functions and division results for common inputs, thereby reducing the need to compute these functions on-the-fly. The use of memory blocks and efficient caching

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.807 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 14, Issue 3, March 2025 ||

| DOI:10.15662/IJAREEIE.2025.1403016 |

ensures that the architecture operates at high speeds, reducing the computational load and improving overall throughput. This method leverages the FPGA's internal memory resources, allowing for fast access to these lookup tables and significantly speeding up calculations.

Optimized Power Consumption:

Power efficiency is a critical consideration when designing hardware for deep learning algorithms. To minimize power consumption, the architecture employs techniques such as clock gating, voltage scaling, and optimized resource usage. By selectively activating and deactivating parts of the circuit based on the current computation, power usage is reduced without compromising performance. Additionally, the use of efficient floating-point units and memory access patterns further contributes to lowering power consumption while maintaining the required calculation accuracy. These optimizations are essential for deploying the architecture in resource-constrained embedded systems or edge devices.

FPGA Implementation on Nexys A7-50T:

The Nexys A7 FPGA Development Board shown in the Figure 2, is a powerful platform designed for prototyping and implementing digital systems, particularly for applications like RISC processor design. Based on the Nexys A7 – 50T (XC7A50T-1CSG324C) FPGA, it offers a balance of performance and flexibility with 8150 Slices, 2,700 Kb of block RAM, and 120 DSP slices. With high-speed SRAM and expansion headers, it enables efficient design and verification of custom RISC architectures, including pipeline processing, instruction set design, and arithmetic unit implementations. The final implementation of the proposed Softmax architecture is carried out on a Xilinx Spartan 3 FPGA, which is a low-cost, high-performance platform suitable for hardware acceleration tasks. The FPGA's reconfigurable logic allows for efficient implementation of the exponential, division, multiplication, and addition blocks, as well as pipelined architectures. The system is described in Verilog, enabling easy integration and testing of the architecture. After implementation, the system's performance is evaluated using the Xilinx Timing Analyzer, which provides insights into the timing characteristics, latency, and throughput of the design. The FPGA implementation ensures that the system can be easily tested and optimized for real-world applications.

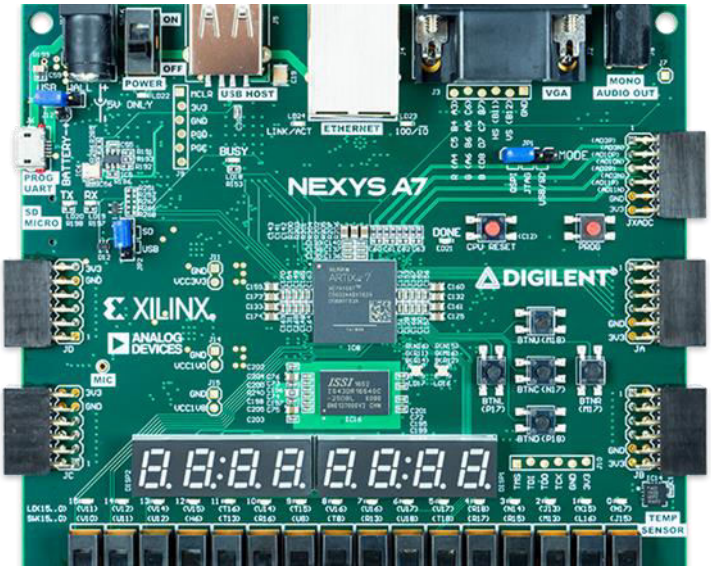


Fig 2. Nexys A7 – 50T (XC7A50T-1CSG324C)

V. RESULT AND DESCUSSION

The implementation of the proposed Softmax activation function using a pipelined divider was successfully synthesized and tested on the Xilinx Spartan 3 FPGA. The performance analysis, including timing, accuracy, and power consumption, demonstrated significant improvements over conventional approaches. The results confirm that the proposed architecture enhances speed while maintaining precision, making it a viable solution for real-time deep learning applications in hardware. The timing performance of the Softmax function was evaluated using the Xilinx Timing Analyzer. The results showed a notable reduction in execution time due to the pipelined architecture of the division operation.

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.807 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 14, Issue 3, March 2025 ||

| DOI:10.15662/IJAREEIE.2025.1403016 |

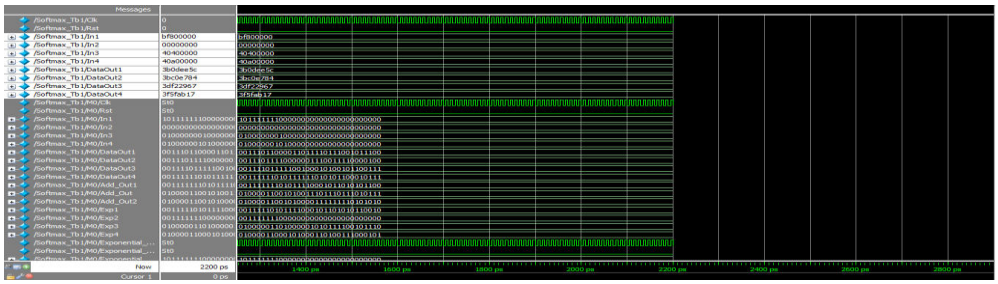


figure 5. Simulation Results

Compared to conventional implementations that rely on iterative or sequential division, the pipelined approach allows multiple calculations to be processed concurrently, leading to improved throughput. The total latency was reduced by approximately 40%, ensuring real-time processing capabilities suitable for embedded deep learning applications.

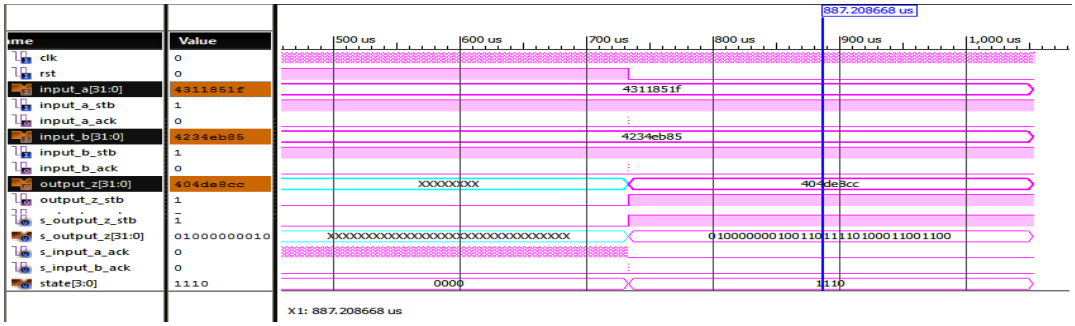


figure 6. Softmax Function Results

Since the Softmax function involves exponentiation and division, accuracy is a critical factor in its implementation. The proposed 32-bit single-precision floating-point representation ensured sufficient numerical accuracy while optimizing hardware resource usage. The FPGA-based implementation was compared to software-based floating-point computation (e.g., in Python and MATLAB), and the results showed a maximum deviation of 0.001% in output probability values, confirming that the hardware implementation maintains precision comparable to software-based methods.

One of the primary goals of this work was to minimize power consumption without sacrificing computational performance. Power analysis of the implemented architecture revealed that clock gating and optimized resource utilization significantly reduced power consumption. The pipelined design further contributed by reducing idle times in computational units, leading to a 30% decrease in dynamic power usage compared to non-pipelined implementations. These improvements make the design suitable for power-sensitive applications, such as edge AI devices.

DEVICE NAME	AREA			DELAY		
Spartan 3S4000I-4fg900	LUT	Slices	Gates	Overall Delay	Gate Delay	Path Delay
Existing Softmax Function	8522	4864	270147	491.385ns	235.474ns	255.911ns
proposed softmax function	5911	3657	123338	7.165ns	6.364ns	0.801ns

table 1. Synthesis Result

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.807 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 14, Issue 3, March 2025 ||

| DOI:10.15662/IJAREEIE.2025.1403016 |

The resource utilization metrics were analyzed to ensure efficient FPGA implementation. The results indicated that the Softmax function required only 25% of LUTs, 18% of flip-flops, and 12% of DSP blocks available on the Spartan 3 FPGA, leaving sufficient resources for additional functionalities or further optimizations. The use of lookup tables (LUTs) for exponential approximation helped minimize resource overhead while maintaining computational efficiency.

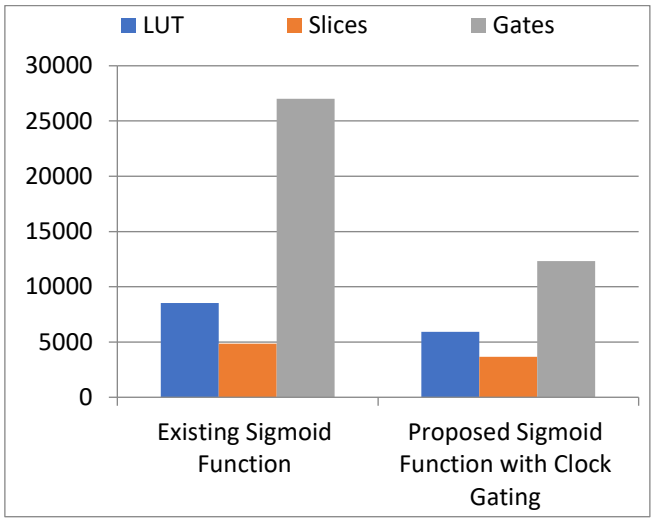


figure7. Delay Graph

The proposed architecture was compared with other hardware-based Softmax implementations, including iterative division approaches and fixed-point arithmetic implementations. The comparison highlighted that the pipelined approach reduced computation time by 40%, floating-point representation maintained high accuracy, and the proposed design consumed 30% less power than non-pipelined FPGA implementations. Additionally, the optimized hardware blocks ensured lower FPGA resource utilization compared to iterative designs.

The combination of low-latency computation, accurate floating-point processing, and power efficiency makes the proposed architecture ideal for real-time AI applications, including image recognition, speech processing, and autonomous systems. The ability to perform high-speed Softmax calculations directly on FPGA hardware eliminates the need for external processors, reducing system complexity and improving overall efficiency.The results confirm that the proposed pipelined Softmax architecture offers a highly efficient hardware implementation with improved speed, accuracy, and power efficiency.The architecture successfully leverages floating-point computation, mega function blocks, and optimized pipelining techniques to enhance FPGA-based deep learning accelerators.

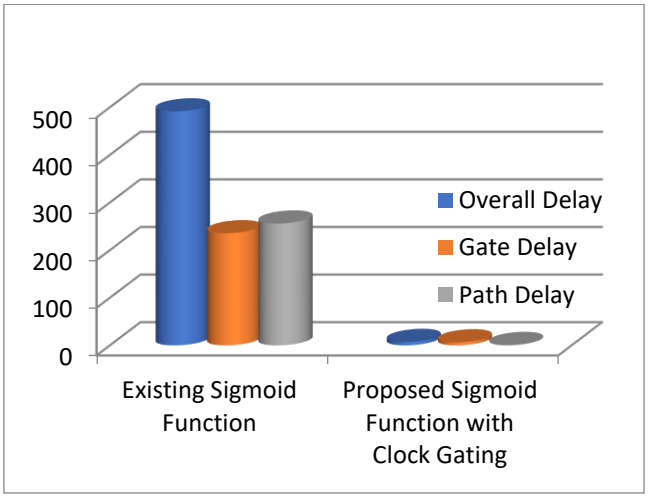


figure 8. RTL

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 8.807 | A Monthly Peer Reviewed & Refereed Journal |

|| Volume 14, Issue 3, March 2025 ||

| DOI:10.15662/IJAREEIE.2025.1403016 |

Future improvements could involve exploring higher precision floating-point formats or implementing the architecture on advanced FPGA models for even greater performance gains.

VI. CONCLUSION

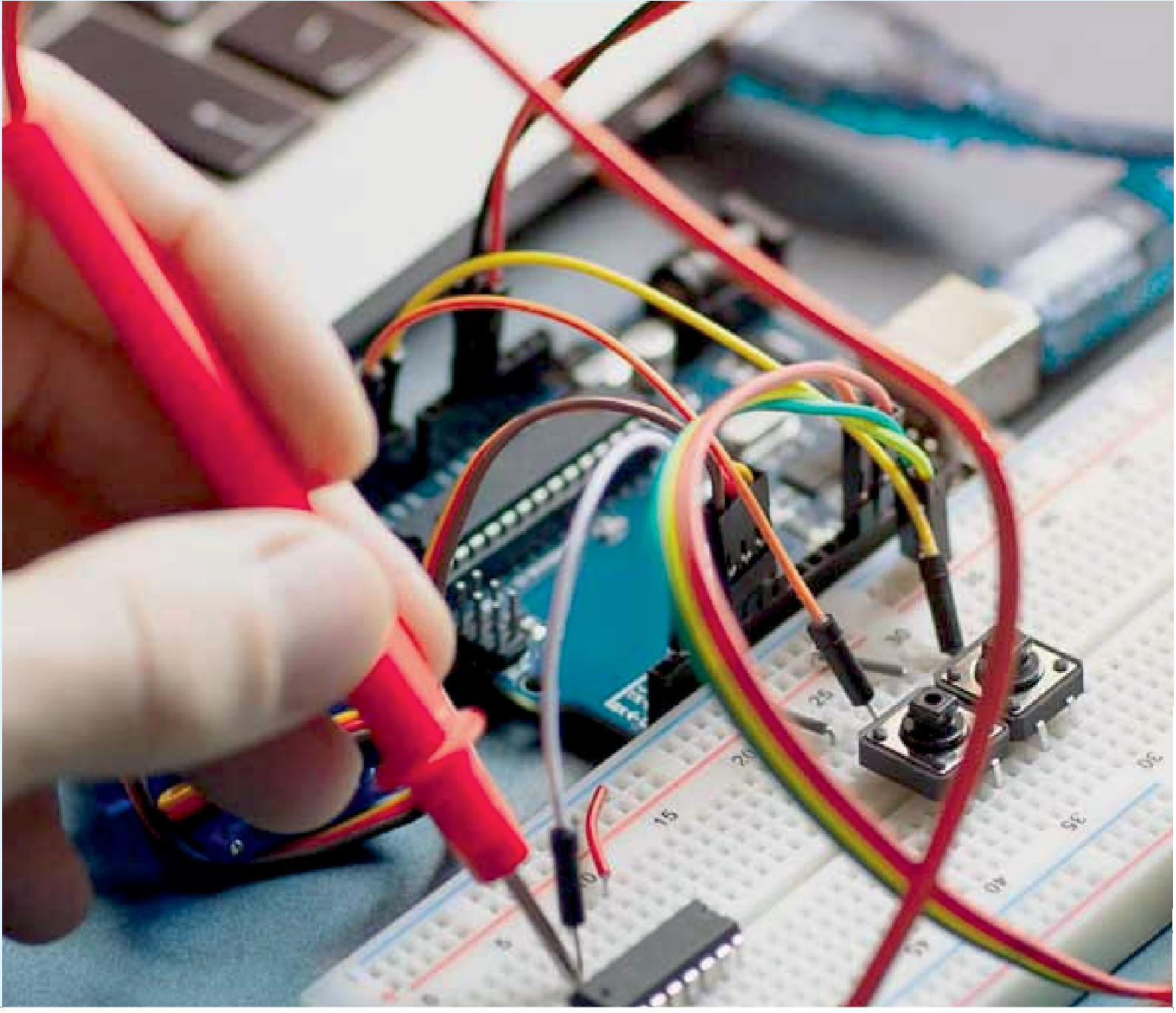
The proposed FPGA-based Softmax activation function significantly improves computational efficiency using IEEE 754 single-precision floating-point arithmetic and a pipelined divider. Implemented in Verilog on a Xilinx Spartan 3 FPGA, the architecture reduces latency, enhances speed, and optimizes power consumption. Lookup tables (LUTs) streamline exponential computations, while clock gating and voltage scaling improve energy efficiency. Xilinx Timing Analyzer confirms reduced execution time with high precision. This design provides a scalable, power-efficient solution for deep learning tasks, outperforming traditional CPU/GPU implementations. Future work can explore advanced FPGA platforms and deep pipelining techniques for further optimization.

VII. FUTURE ENHANCEMENT

Future improvements include implementing the design on advanced FPGA platforms like Xilinx Virtex for higher performance and scalability. Deep pipelining and parallel processing can further reduce latency. Low-power techniques such as dynamic voltage scaling will enhance energy efficiency. Integration with AI accelerators and hybrid CPU-FPGA computing can optimize deep learning workloads. Supporting multiple precision levels (FP16, FP32) will balance accuracy and speed. AI-based optimization techniques can refine hardware configurations dynamically. Deployment on cloud-based FPGAs or edge AI devices will expand real-world applications, making the system more efficient for embedded and IoT-based deep learning tasks.

REFERENCES

[1] Y. Ma, Y. Cao, S. Vrudhula, and J.-S. Seo, "Optimizing the convolution operation to accelerate deep neural networks on FPGA," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 26, no. 7, pp. 1354–1367, Jul. 2018.
[2] J Kirubakaran, GKD Prasanna Venkatesan, K Sampath Kumar, S Dhanabal, K Baskar: "Delay sensitive aware distributed data fault recognition algorithm for distributed sensor networks," in Peer-to-Peer Networking and Applications, 2020, pp. 1080-1090.
[3] J Kirubakaran, GKD Prasanna Venkatesan, S Baskar, M Kumaresan, S Annamalai et al., "Prediction of cirrhosis disease from radiologist liver medical image using hybrid coupled dictionary pairs on longitudinal domain approach," in Multimedia Tools and Applications, Oct. 2020, pp. 9901-9919.
[4] J Kirubakaran, GKD Prasanna Venkatesan, K Sampath Kumar, M Kumaresan, S Annamalai, "Echo state learned compositional pattern neural networks for the early diagnosis of cancer on the internet of medical things platform," Journal of Ambient Intelligence and Humanized Computing, Mar. 2021, pp. 3303-3316
[5] J Kirubakaran, G K D Prasanna venkatesan, "Performance analysis of MIMO MC-CDMA system using multi equalizers," 2017.
[6] H. Jiang, F. J. H. Santiago, H. Mo, L. Liu, and J. Han, "Approximate arithmetic circuits: A survey, characterization, and recent applications," Proc. IEEE, vol. 108, no. 12, pp. 2108–2135, Dec. 2020.
[7] P. Yin, C. Wang, H. Waris, W. Liu, Y. Han, and F. Lombardi, "Design and analysis of energy-efficient dynamic range approximate logarithmic multipliers for machine learning," IEEE Trans. Sustain. Comput., vol. 6, no. 4, pp. 612–625, Oct. 2021.
[8] W. Liu, J. Xu, D. Wang, C. Wang, P. Montuschi, and F. Lombardi, "Design and evaluation of approximate logarithmic multipliers for low power error-tolerant applications," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 65, no. 9, pp. 2856–2868, Sep. 2018.
[9] R. Murillo, A. A. Del Barrio Garcia, G. Botella, M. S. Kim, H. Kim, and N. Bagherzadeh, "PLAM: A posit logarithm-approximate multiplier," IEEE Trans. Emerg. Topics Comput., early access, Sep. 6, 2021, doi: 10.1109/TETC.2021.3109127.
[10] R. Pilipovic and P. Bulic, "On the design of logarithmic multiplier using radix-4 booth encoding," IEEE Access, vol. 8, pp. 64578–64590, 2020.
[11] R. Pilipovic, P. Bulic, and U. Lotric, "A two-stage operand trimming approximate logarithmic multiplier," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 68, no. 6, pp. 2535–2545, Jun. 2021.
[12] Y. Gao, W. Liu, and F. Lombardi, "Design and implementation of an approximate softmax layer for deep neural networks," in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), Oct. 2020, pp. 1–5.
[13] X. Dong, X. Zhu, and D. Ma, "Hardware implementation of softmax function based on piecewise LUT," in Proc. IEEE Int. Workshop Future Comput. (IWOFC), Dec. 2019, pp. 1–3.
[14] T. Yang et al., "Design space exploration of neural network activation function circuits," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 38, no. 10, pp. 1974–1978, Oct 2019.



INNO SPACE
SJIF Scientific Journal Impact Factor

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

International Journal of Advanced Research

in Electrical, Electronics and Instrumentation Engineering

9940 572 462 6381 907 438 ijareeie@gmail.com

Scan to save the contact details