ISSN 2277-2685
IJESR/Jan-Mar. 2025/ Vol-15/Issue-1s/68-75
Tanneru Sharanya *et. al.*, / International Journal of Engineering & Science Research

Design Of Power Efficient POSIT Multiplier

[1]Mrs. Kazi Nikhat Pravin, [2]Tanneru Sharanya, [3]Shikilambatla Sravika, [4]Chintalphani Trisha Reddy
[1] Associate Professor, Electronics and Communication Engineering, BRECW
[2,3,4]B.Tech Students, Department of Electronics and Communication Engineering, BRECW

ABSTRACT

Design Of Power Efficient POSIT Multiplier Abstract Posit number system has been used as an alternative to the IEEE floating-point number system in many applications, especially the recent popular deep learning. Its non-uniformed number distribution fits well with the data distribution of deep learning and thus can speed up the deep learning training process. Among all the related arithmetic operations, multiplication is one of the most frequent operations used in applications. However, due to the bit-width flexibility nature of posit numbers, the hardware multiplier is usually designed with the maximum possible mantissa bit-width. As the mantissa bit-width is not always the maximum value, such multiplier design leads to a high power consumption especially when the mantissa bit-width is small. In this brief, a power-efficient posit multiplier architecture is proposed. The mantissa multiplier is still designed for the maximum possible bit-width however, the whole multiplier is divided into multiple smaller multipliers. Only the required small multipliers are enabled at run-time. Those smaller multipliers are controlled by the regime bit-width which can be used to determine the mantissa bit-width. This design technique is applied to 8-bit, 16-bit, and 32-bit posit formats in this brief, and an average of 16% power reduction can be achieved with negligible area and timing overhead. Power consumption distribution of a posit multiplier Posit component extraction in hardware arithmetic unit Datapath of the proposed posit multiplier.

1-INTRODUCTION

A power-efficient Posit multiplier is a key component in modern computing systems that leverages the advantages of the Posit number system, an emerging alternative to traditional floating-point representation. The Posit system provides higher accuracy, dynamic range, and efficiency for various computational tasks, making it particularly attractive for machine learning, scientific computing, and embedded systems. However, one of the main challenges in implementing Posit arithmetic, especially multiplication, is achieving high performance while minimizing power consumption.

A power-efficient design of the Posit multiplier focuses on reducing energy usage without sacrificing the precision and speed that Posits offer. This involves optimizing the hardware architecture, utilizing techniques such as operand truncation, approximate computing, and advanced low-power design methods. By developing such a multiplier, designers aim to improve both computational efficiency and energy consumption, which is crucial for battery- powered and resource-constrained devices, as well as large-scale data centers where power is a critical concern.

2-LITERATURE SURVEY

In this section, some of the previous works in the field of approximate multipliers are briefly reviewed. In, an approximate multiplier and an approximate adder based on a technique named broken-array multiplier (BAM) were proposed. By applying the BAM approximation method of to the

conventional modified Booth multiplier, an approximate signed Booth multiplier was presented in. The approximate multiplier provided power consumption savings from 28% to 58.6% and area reductions from 19.7% to 41.8% for different word lengths in comparison with a regular Booth that saved the power by 31.8%–45.4% over an accurate multiplier. An approximate multiplier. Kulkarni et al. suggested an approximate multiplier consisting of a number of 2 × 2 inaccurate building blocks signed 32-bit multiplier for speculation purposes in pipelined processors was designed in. It was 20% faster than a full-adder-based tree multiplier while having a probability of error of around 14%.

In, an error-tolerant multiplier, which computed the approximate result by dividing the multiplication into one accurate and one approximate part, was introduced, in which the accuracies for different bit widths were reported. In the case of a 12-bit multiplier, a power saving of more than 50% was reported. The use of approximate multipliers in image processing

applications, which leads to reductions in power consumption, delay, and transistor count compared with those of an exact multiplier design, has been discussed in the literature. An accuracy-configurable multiplier architecture (ACMA) was suggested for error-resilient systems. To increase its throughput, the ACMA made use of a technique called carry-in prediction that worked based on a pre-computation logic. When compared with the exact one, the proposed approximate multiplication resulted in nearly 50% reduction in the latency by reducing the critical path. Also, Bhardwaj et al. presented an approximate Wallace tree multiplier (AWTM). Again, it invoked the carry-in prediction to reduce the critical path. In this work, AWTM was used in a real-time benchmark image application showing about 40% and 30% reductions in the power and area, respectively, without any image quality loss compared with the case of using an accurate Wallace tree multiplier (WTM) structure. Approximate unsigned multiplication and division based on an approximate logarithm of the operands have been proposed. In the proposed multiplication, the approximate logarithms' summation determines the operation's result. Hence, the multiplications are simplified to some shift and add operations. A method for increasing the accuracy of the multiplication approach was proposed. It was based on the decomposition of the input operands. This method considerably improved the average error at the price of increasing the hardware of the approximate multiplier by about two times. A dynamic segment method (DSM) is presented, which performs the multiplication operation on an m-bit segment starting from the leading one bit of the input operands. A dynamic range unbiased multiplier (DRUM) multiplier, which selects an m-bit segment starting from the leading one bit of the input operands and sets the least significant bit of the truncated values to one, has been proposed.

In this structure, the truncated values are multiplied and shifted to left to generate the final output. An approximate 4 ×4 WTM has been proposed that uses an inaccurate 4:2 counters. In addition, an error correction unit for correcting the outputs has been suggested. To construct larger multipliers, this 4 × 4 inaccurate Wallace multiplier can be used in an array structure. Most of the previously proposed approximate multipliers are based on either modifying the structure or complexity reduction of a specific accurate multiplier. In this paper, we propose performing the approximate multiplication through simplifying the operation. The difference between our work and the previous is that, although the principles in both works are almost similar for unsigned numbers, the mean error of our proposed approach is smaller. In addition, we suggest some approximation techniques when the multiplication

is performed for signed number.

3-Mathematical Foundation and Architecture

The mathematical foundation and architecture of posit multipliers are integral to understanding the advantages and operational efficiency of the posit number system in modern computing. This foundation's core lies in the mathematical principles governing posit numbers, specifically the regime, exponent, and fraction fields. These components allow posits to represent a wide range of values with dynamic precision, thus addressing many limitations inherent in traditional floating-point systems. In this section, we will break down the multiplication process within posit arithmetic, providing a step-by-step analysis that showcases how combining these fields enables efficient and accurate calculations. Furthermore, error analysis and accuracy considerations will be discussed, emphasizing the significance of rounding modes and exception handling in ensuring reliable computational outcomes. As we transition from the mathematical principles to the architecture of the posit multiplier, we will explore the specific structures and components that define its operation. This includes input handling, data flow, and output generation, along with the distinct roles played by various modules in the multiplication process. A comparison with floating-point multiplier architecture will illustrate how the posit multiplier achieves greater efficiency and precision, positioning it as a superior alternative in the context of energy-sensitive and high- performance computing applications.

Software Requirement

The software must facilitate efficient processing of posit numbers, incorporating error handling and optimization techniques to enhance computational speed and accuracy. Additionally, requirements for performance monitoring and power management features are essential to align with the project's objectives of minimizing energy consumption. Clear and well-defined software requirements not only guide developers in creating robust applications but also help in managing expectations among stakeholders, ultimately leading to the successful delivery of a product that meets its intended purpose efficiently and effectively.

XILINX Software

Xilinx Tools is a suite of software tools used for the design of digital circuits

Implemented using Xilinx Field Programmable Gate Array (FPGA) or Complex Programmable Logic Device (CPLD). The design procedure consists of (a) design entry, (b) synthesis and implementation of the design, (c) functional simulation and (d) testing and verification. Digital designs can be entered in various ways using the above CAD tools: using a schematic entry tool, using a hardware description language (HDL) – VHDL or VHDL or a combination of both. In this lab we will only use the design flow that involves the use of VHDL HDL.

The CAD tools enable you to design combinational and sequential circuits starting with VHDL HDL design specifications. The steps of this design procedure are listed below:

- Create VHDL design input file(s) using template driven editor.
- Compile and implement the VHDL design file(s).
- Create the test-vectors and simulate the design (functional simulation) without using a PLD (FPGA or CPLD).

- Assign input/output pins to implement the design on a target device.
- Download bit stream to an FPGA or CPLD device.
- Test design on FPGA/CPLD device
- A VHDL input file in the Xilinx software environment consists of the following segments:
- Header: module name, list of input and output ports.
- Declarations: input and output ports, registers and wires.
- Logic Descriptions: equations, state machines and logic functions.
- End: end module

4- RESULT

The results and simulation phase is a critical component in the development of the power-efficient posit multiplier, as it validates the design's functionality, performance, and efficiency. This phase involves the application of various simulation tools and methodologies to analyze the multiplier's behavior under different conditions and inputs, ensuring it meets the specified requirements. By leveraging simulation environments, developers can effectively assess key performance indicators such as speed, accuracy, and power consumption before physical implementation. The evaluation of the posit multiplier encompasses a range of scenarios, including varying operand sizes and operational contexts, to demonstrate its versatility and robustness. Through comprehensive testing and analysis, the simulation results provide valuable insights into the multiplier's operational efficiency, revealing its potential advantages over traditional floating-point systems. Ultimately, this phase not only confirms the effectiveness of the design but also highlights areas for further optimization, paving the way for its application in real-world computational tasks.

RTLAnd Technology Schematics

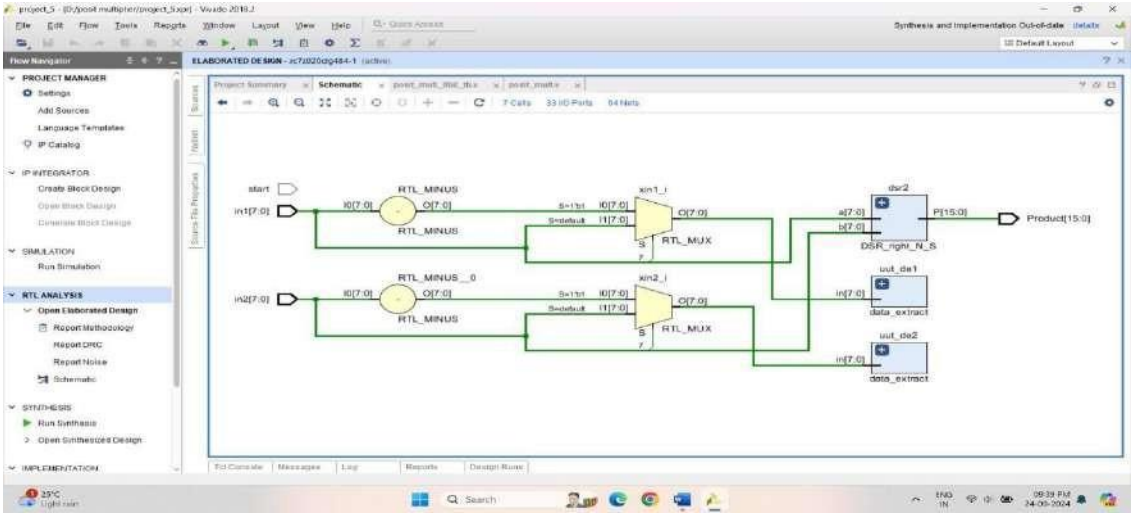


Fig 5.2.1: 8-bit RTL view of posit multiplier (A Snapshot from Vivado Software)

In the context of a power-efficient posit multiplier, the RTL diagram illustrates the key components, such as the registers for input and output, the arithmetic logic units (ALUs)
responsible for multiplication, and any control logic needed to manage data flow and timing. The diagram typically includes the following elements: Registers: Represent storage locations for operands and results.
Arithmetic Logic Units (ALUs): Perform multiplication and other arithmetic operations. Control Signals: Manage the sequencing of operations and data transfers.
Data Paths: Indicate the routes through which data travels between components.
The RTL diagram is instrumental for developers and engineers in understanding the architecture of the circuit, facilitating easier debugging, optimization, and synthesis into physical hardware.

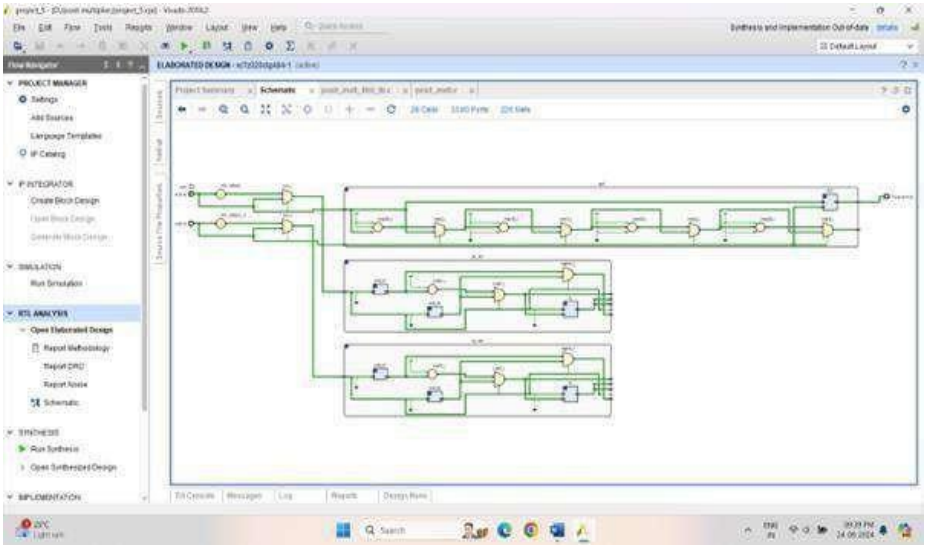
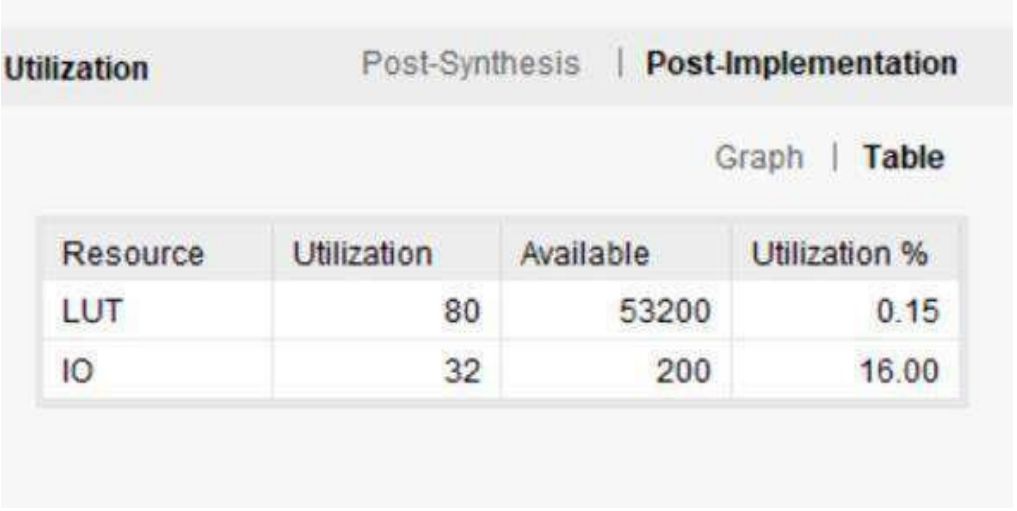


Fig 5.2.2: 8-bit Detailed view of RTL posit multiplier (A Snapshot from Vivado Software)



Resource	Utilization	Available	Utilization %
LUT	80	53200	0.15
IO	32	200	16.00

Fig 5.3.1: Synthesis result of Area

5.1.1 Power

Fig 5.3.3: Synthesis result of power.

5.1.2 8 Bit Un-Signed POSIT Multiplier:

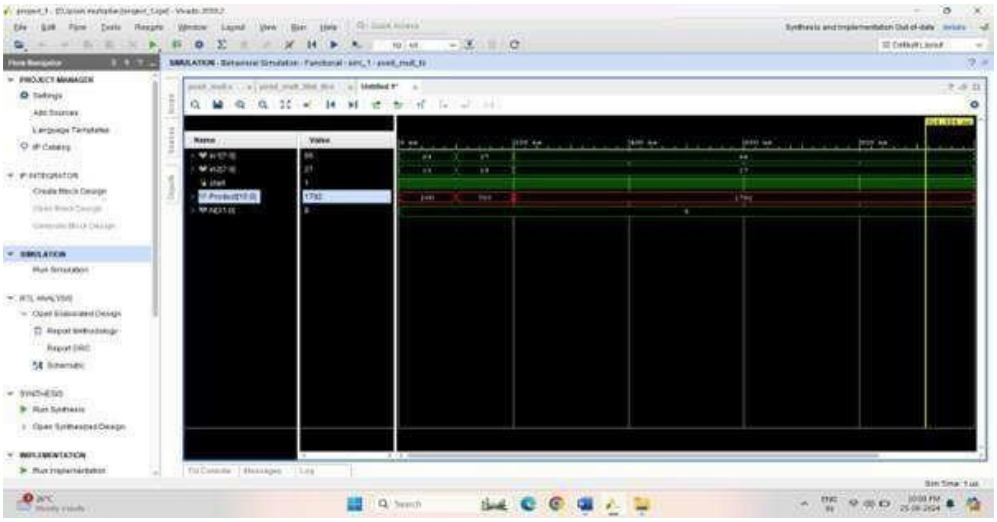


Fig 5.3.4: Simulation result of 8-bit un-signed posit multiplier (A Snapshot from Vivado Software)

5.1.3 8-Bit Signed POSIT Multiplier:

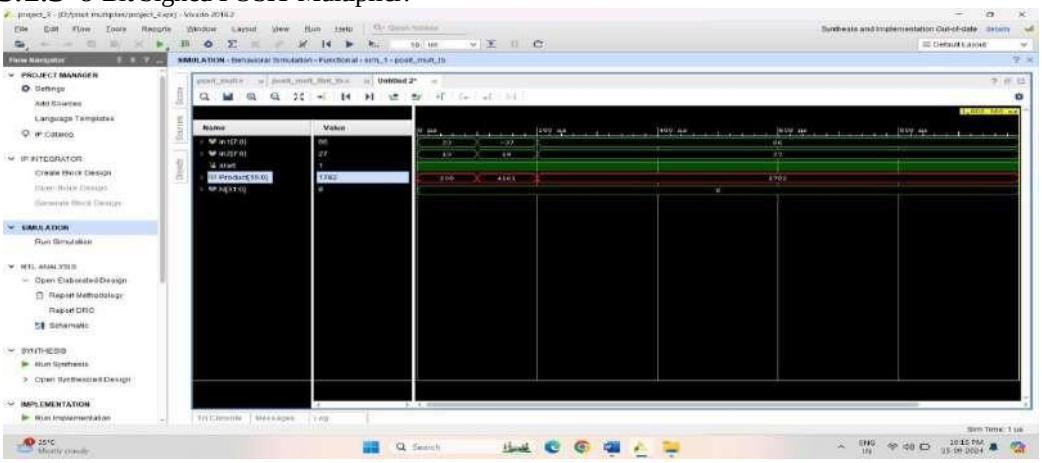


Fig 5.3.5: Simulation result of 8-bit signed posit multiplier (A Snapshot from Vivado Software)

Table 1: COMPARISON TABLE

PARAMETERS	EXISTING	PROPOSED
AREA	144 LUTS	80 LUTS
POWER(W)	0.143	0.065
DELAY	8.806ns	5.232ns

It can be seen that the comparison of the 8-bit POSIT multiplier in terms of operating speed indicates that the Posit multiplier and the Rounding Algorithm multiplier have similar characteristics of power. But in terms of hardware complexity, the 8-bit modified Rounding Algorithm multiplier gives a reduction in the area and delay by almost compared to the 8-bit Posit multiplier. Moreover, with the schematic implementation of the 8-bit Posit multiplier, the modified ROBA multiplier reduces hardware complexity compared to the Posit multiplier.

5- Conclusion

In conclusion, this chapter has highlighted the design and implementation of the power- efficient posit multiplier, showcasing its advantages over traditional floating-point systems in terms of performance and energy efficiency. The results affirm its viability for modern computational applications. Looking ahead, there are several promising avenues for further development, including integration with advanced computing architectures, optimization techniques, and specialized hardware implementations. Expanding its functionality and pursuing standardization will also enhance its adoption. Additionally, exploring its application in emerging fields such as quantum computing and IoT can unlock new opportunities. Overall, the posit multiplier is well-positioned to contribute significantly to the future of efficient computing solutions.

REFERENCES

[1] Efficient Approximate Posit Multipliers for Deep Learning Computation – This paper addresses the cost challenges of posit arithmetic in hardware and proposes efficient designs for deep learning applications (2023).

[2] Design of Power Efficient Posit Multiplier using Compressor-Based Adder – This paper focuses on breaking the mantissa multiplier into smaller units for power optimization in deep learning applications (2023)

[3] Dynamic-Precision Efficient Posit Multiplier for Neural Networks– This work explores a posit multiplier optimized for dynamic precision control in neural network computations, offering better power and area efficiency (2023)

[5] Sitar, D., and Gustafson, J. L., "A Novel Power-Efficient Posit Multiplier for IoT Applications" (2022)

[6] Muruganatham Ganesan, "Power-Efficient Posit Multiplier Design with Approximate Computing," 2022

[7] "High-Performance Arithmetic Circuits" – Pramod Kumar Meher (2022)

[8] "Efficient Floating-Point and Posit Arithmetic Hardware Implementations" – F. Hossain, A. George (2021)

[9] Hardware Design of Posit Arithmetic Units" by Vincent Granados, et al., (2021)

[10] Rajesh Kumar, Manoj Kr. Gupta, "FPGA Implementation of Posit Multiplier for Energy-
Efficient Computation," 2020

[11] Johnson, J - "Rethinking floating point for deep learning," CoRR, 2018.

[12] Chaurasiya, R. et al- "Parameterized posit arithmetic hardware generator," IEEE 36th Int. Conf. Comput. Design (ICCD), 2018.

[13] Jaiswal, M. K., & So, H.-K - "Architecture generator for type-3 unum posit adder/subtractor," IEEE Int. Symp. Circuits Syst. (ISCAS), 2018.

[14] Podobas, S., & Matsuoka, S - "Hardware implementation of POSITs and their application in FPGAs," IEEE Int. Parallel Distrib. Process. Symp. Workshops (IPDPSW), 2018.

[15] Carmichael, Z. et al - "Deep positron: A deep neural network using the posit number system," 2018.

[16] S. van der Linde, “Posits als vervanging van floating-points: Een vergelijking van Unum Type III Posits met IEEE 754 Floating Points met Mathematica en Python”, Bachelor’s Thesis, Delft University of Technology, Sep. 26, 2018.

[17] H. F. Langroudi, Z. Carmichael, J. L. Gustafson, and D. Kudithipudi, “PositNN: Tapered Precision Deep Learning Inference for the Edge”, 2018.