A Partial Product Reduction Process Integrating Additions and Accumulations to Create a High-Performance Multiply-Accumulate Unit(MAC)

Mr.N.B.Jilani[(1)] Mrs.M.Haritha[(2)] Shaik Naseema[(3)] Katari Narmada[(4)] Palnati Neha[(5)] Nemalipuri Krishnaveni[(6)]

[1,2] *Krishna Chaithanya Institute Of Technology & Sciences, Ece Department, Markapur, Andhra Pradesh.*
[3,4,5,6,] *Krishna Chaithanya Institute Of Technology & Sciences, UG Student-ECE, Markapur, Andhra Pradesh.*

Abstract. In this paper, we provide a low-power, high-speed pipeline multiply-accumulate (MAC) architecture. Carry propagations of additions, such as additions in multiplications and additions in accumulations, frequently result in high power consumption and high route latency in a typical MAC. We incorporate a portion of additions into the partial product reduction (PPR) procedure to tackle this issue. Higher importance bits are not added or accumulated in the proposed MAC architecture until the PPR phase of the subsequent multiplication. A small-size adder is intended to gather the entire number of carries to appropriately handle the overflow in the PPR process. Experimental results demonstrate that the suggested MAC architecture can significantly lower both power consumption and circuit area under the same conditions as prior efforts. This can be achieved by functionally synthesized by using Xilinx Vivado Verilog Hdls simulator.Verilog Hdls.

Keywords: MAC(Multiply and Accumulate Unit),5G Communications, FPGA, Verilog HDL, PPrs

I.INTRODUCTION

The Multiply-Accumulate (MAC) operation lies at the heart of numerous computational tasks, including digital signal processing, neural network inference, and scientific computing. In traditional implementations, MAC units typically consist of separate multiplier and accumulator components, leading to increased hardware complexity and energy consumption. To address these challenges, there has been growing interest in optimizing MAC unit architectures to improve computational efficiency. In this paper, we present a novel approach to enhance the performance of MAC units by integrating the addition and accumulation stages, thereby streamlining the overall computation process.

The multiply-accumulate (MAC) unit is a fundamental block for digital signal processing (DSP) applications [1]. Especially, in recent years, the development of real-time edge applications has become a design trend [2,3]. Thus, there is a strong demand for high-speed low-power MAC units.

A conventional MAC unit is composed of two individual blocks: a multiplier and an accumulator (i.e., an accumulate adder). An N-bit MAC unit includes an N-bit multiplier and a (2N+α-1)-bit accumulator (adder), where α is the number of guard bits used to avoid overflow (caused by long sequences of multiply-accumulate operations). A lot of previous works [4-12] paid attention to the optimization of multiplier and the optimization of adder, respectively.

A multiplier [4-7] usually has three steps. The first step is the partial product generation (PPG) process. For example, AND gates can be used to generate a partial product matrix (PPM) for an unsigned multiplication. The second step is the partial product reduction (PPR) process. By using the Dadda tree approach [4,5,7] or the Wallace tree approach [4-6], the PPM can be reduced to become two rows. The third step is the final addition. An adder (called the final adder) is used to perform the summation of the final two rows. For an N-bit multiplier, a (2N-1)-bit adder is required for the final addition.

Various adder architectures [8-10] have been proposed for the trade-offs among delay, area, and power. Furthermore, various MAC unit models can be developed by replacing the multiplier as well as the accumulator (adder) with various architectures. Comparisons on delay, area and power among different MAC unit models are reported in [11,12].

In a conventional MAC unit, it is necessary to perform two carry propagations: additions in multiplications and additions in accumulations. Note that the carry propagation is time consuming. Therefore, in [13], the multiplier output is fed back to the input of the PPR process. Since the accumulation is handled by the final adder, only one carry propagation is required. Moreover, based on this architecture [13], the area of 16-bit MAC unit can be further reduced 3.2% by fully utilizing the compression [14].

In [13,14], they do not discuss how to accommodate guard bits in their designs. Ercegovac and Lang [15] add an extra circuit to the final adder for handling guard bits. Owing to this extra circuit, the carry propagation in the final adder becomes longer. Another drawback of this architecture [15] is that it only supports sign-magnitude numbers.

Different from those previous works [13-15] that handle the accumulation in the final adder, Hoang et al. [16] use a carry save adder to implement the final adder. In [16], a carry-save format (one sum vector and one carry vector) is sent to the accumulator without being added to only one vector. Although this architecture [16] can remove the carry propagation in the final adder, it requires a (2N+α-1)-bit accumulator. Besides, it is also noteworthy to mention that the concept of this architecture [16] has been used in modern floating-point fused multiply-add (FMA) designs [17,18].

Based on the architecture proposed by Hoang et al. [16], some attentions [19,20] have been paid to the compressor design for the PPR process. For example, by using the compressor proposed in [20], the number of LUTs for 8-bit MAC unit can be reduced 21.3% in a Virtex 7 FPGA platform.

In this project, we propose a novel MAC architecture for high performance. In order to reduce critical path delays and power dissipations (caused by carry propagations), our basic idea is to integrate a part of additions (including a part of the final addition in the multiplication and a part of the addition in the accumulation) into the PPR process. In the proposed MAC unit, the final addition of higher significance bits is not performed in the current multiplication. Instead, the final addition and accumulation of higher significance bits are performed in the PPR process of the next multiplication. As a result, the lengths of carry propagations can be greatly reduced. Moreover, to correctly deal with the overflow during the PPR process, an α-bit accumulator (adder) is designed to count the total number of carries, where α is the number of guard bits. Experimental results consistently show that the proposed approach works well in practice.

The proposed MAC unit is a two-stage (i.e., two-cycle) pipeline design. The first stage performs the PPG process, the PPR process, a part of the final addition and the α-bit addition (for handling overflow). The second stage performs an addition to produce the accumulation result. Note that, for power saving, the second stage can only be executed in the last cycle (of the entire sequence

of multiply-accumulate operations) by applying the gating technique. For an N-bit MAC unit, the main differences between the proposed architecture and the conventional architecture are below.
Final addition in the multiplication. The conventional architecture requires a (2N-1)-bit adder. On the other hand, the proposed architecture only requires a (2N-k-1)-bit adder, where k denotes the number of higher significance bits whose additions (accumulation) are not performed in the final addition.
Accumulation in the MAC unit. The conventional architecture requires a (2N+α-1)-bit adder. On the other hand, the proposed architecture only requires a (k+α)-bit adder. Moreover, by applying the gating technique, the (k+α)-bit adder can only be executed in the last cycle.
It is noteworthy to mention that the time-consuming carry propagation is also a challenging issue in the modular multiplication [21,22,23]. Thus, some high-radix, scalable, and signed digit multipliers [21,22,23] have been proposed for modular multiplication. Different from those previous works [21,22,23], the proposed MAC unit is designed for standard arithmetic (instead of modular arithmetic).
The High-Performance Multiply-Accumulate Unit (MAC) represents a critical component within digital signal processors (DSPs), application-specific integrated circuits (ASICs), and other computational platforms. Its primary function is to perform multiply-accumulate operations efficiently, which are fundamental in many computational tasks such as filtering, convolution, and matrix operations. Integrating additions and accumulations within the MAC unit is a strategy aimed at enhancing its performance and efficiency. Here's an overview of this approach:
Basic MAC Operation:
The fundamental operation of a MAC unit involves multiplying two operands (typically two input values) and adding the result to an accumulator register. This operation is expressed as $Acc=Acc+(A\times B)$, where Acc is the accumulator register and A and B are the operands.
Integrating Additions:

In traditional MAC units, additions and multiplications are separate operations. Integrating additions within the MAC unit involves optimizing the hardware to perform both operations simultaneously or in a pipelined manner. This reduces the overall latency and improves throughput.
Integrating Accumulations:
Accumulations refer to the repeated execution of the MAC operation over a sequence of operands. Integrating accumulations within the MAC unit involves optimizing the hardware to efficiently update the accumulator register without introducing additional latency between successive MAC operations.
Pipelining:
Pipelining is a technique commonly used in high-performance MAC units to overlap the execution of multiple MAC operations. By breaking down the MAC operation into multiple stages and processing multiple operations concurrently, pipelining maximizes throughput and reduces latency.
Parallelism:
Exploiting parallelism within the MAC unit involves utilizing multiple processing elements or functional units to perform MAC operations in parallel. This can significantly enhance the overall throughput of the MAC unit, especially in applications with high computational demands.
Advanced Optimization Techniques:

Advanced optimization techniques such as instruction-level parallelism, data prefetching, and register renaming can further enhance the performance of the integrated additions and accumulations within the MAC unit. These techniques aim to minimize stalls and maximize the utilization of available resources.

Application in DSPs and ASICs:

Integrated additions and accumulations in MAC units are particularly beneficial in DSPs and ASICs designed for signal processing applications. They enable these devices to efficiently execute complex algorithms such as digital filters, fast Fourier transforms (FFTs), and matrix multiplications with high throughput and low latency.

In summary, integrating additions and accumulations within the MAC unit is a key strategy for enhancing its performance and efficiency in various computational platforms, particularly in DSPs and ASICs. By optimizing hardware design, leveraging pipelining and parallelism, and employing advanced optimization techniques, MAC units can deliver high-performance computation capabilities for a wide range of applications.

II.EXISTING SYSTEM AND PROPOSED SYSTEM

EXSISTING SYSTEM

In a conventional MAC unit, carry propagations of additions (including final additions in multiplications and additions in accumulations) often result in large power consumption and large path delay. To resolve this problem, we are motived to reduce the lengths of carry propagations in the final addition and the accumulation. Our basic idea is to integrate a part of additions (including a part of the final addition and a part of the accumulation) into the PPR process. As a result, the lengths of carry propagations can be reduced.

PROPOSED METHOD

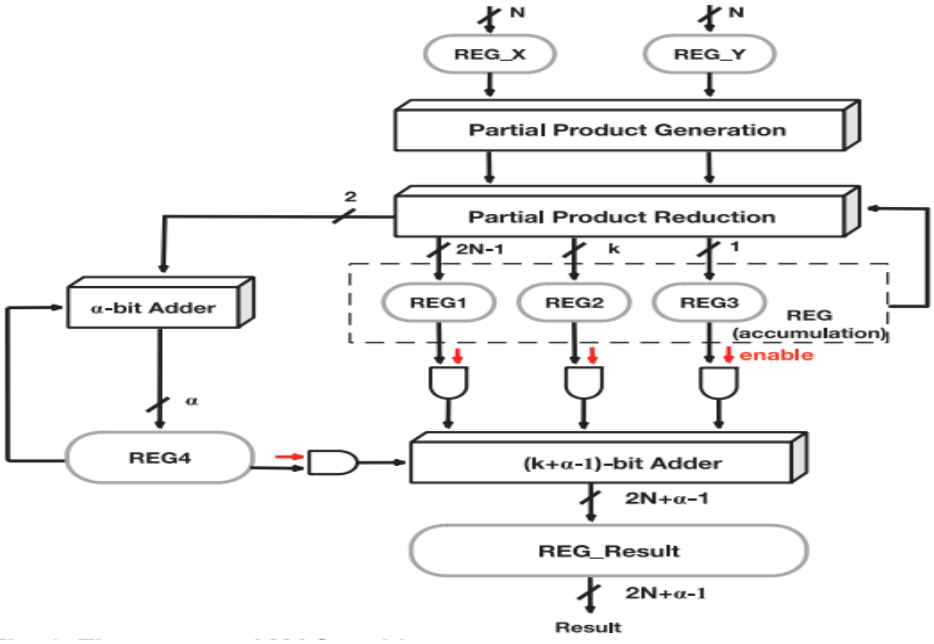

Fig 1. The proposed MAC architecture.

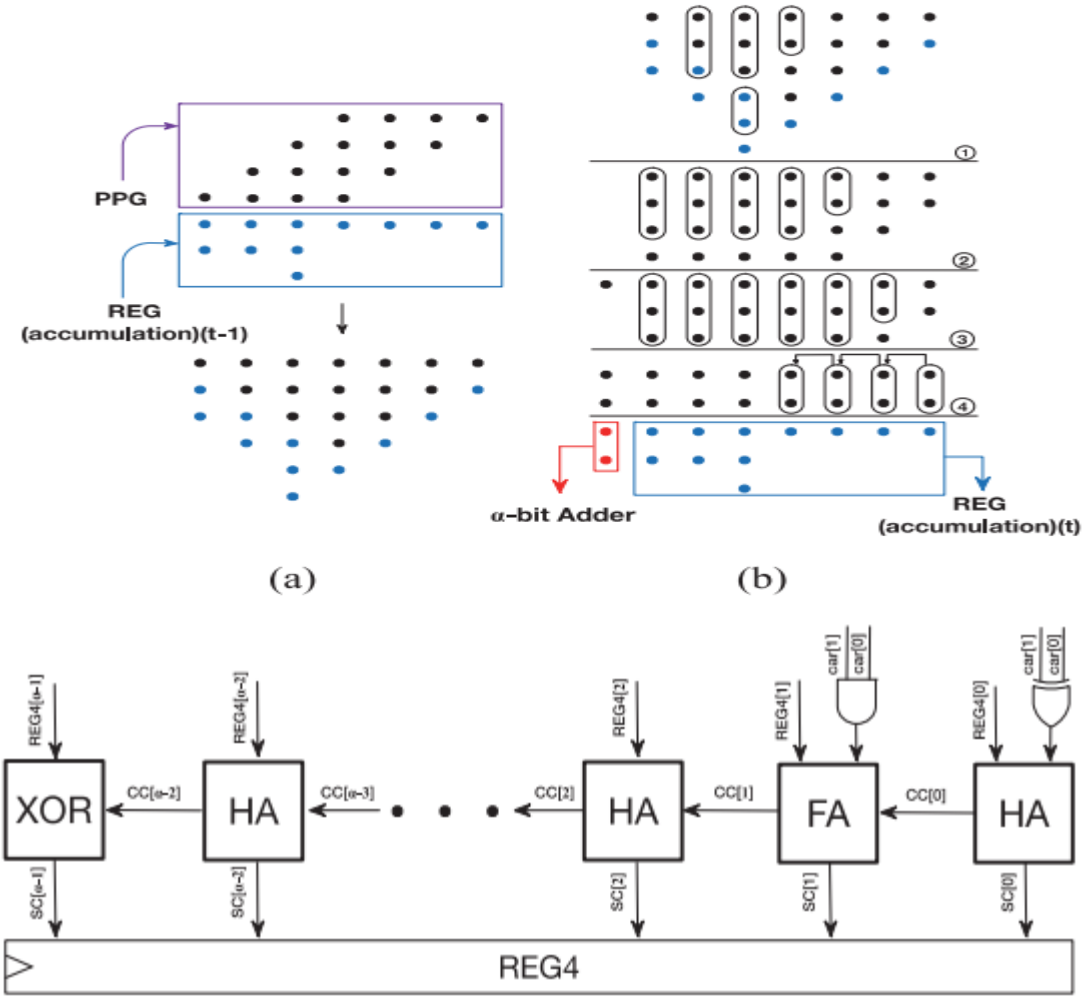

Fig2. 4-bit MAC. (a) PPM (b) PPR process.

Fig3. The mechanism of α-bit addition in the unsigned MAC unit.

Our proposed integrated MAC unit architecture consolidates the addition and accumulation stages into a unified module, eliminating the need for separate hardware components. By leveraging shared resources and optimized datapath designs, our architecture achieves significant savings in terms of area and power consumption. Furthermore, the integrated design facilitates efficient pipelining and parallelization, enabling higher throughput and reduced latency compared to traditional MAC units.

In this project, we present the proposed two-stage (i.e., two cycle) MAC architecture. The first stage performs the PPG process, the PPR process (based on the PPM that combines the PPG result and the accumulation result), the (2N-k-1)-bit addition (i.e., a part of the final addition) and the α-bit addition (for dealing with the overflow in the PPR process). Then, the second stage performs the (k+α)-bit addition to produce the accumulation result. The main features of the proposed architecture are below. To reduce the lengths of carry propagations, we integrate a part of additions into the PPR process. To handle overflow in the PPR process,

an α-bit adder is used to count the total number of carries. By applying the gating technique, the second stage can only be executed in the last cycle (of the entire sequence of multiply-accumulate operations) for power saving. The proposed two-stage pipeline MAC unit is displayed in Fig. 2. Our PPM (for the PPR process) is composed of two PPMs: one PPM is derived by the PPG and the other PPM is derived by the accumulation.

This project presents a low-power high-speed two-stage pipeline MAC architecture for real-time DSP applications. Our basic idea is to integrate a part of additions (including a part of the final addition in the multiplication and a part of the addition in the accumulation) into the PPR process. As a result, critical path delays and power dissipations caused by carry propagations can be reduced. To correctly deal with the overflow during the PPR process, an α-bit accumulator is used to count the total number of carries. Experimental results consistently show that the proposed approach works well in practice.

The proposed MAC architecture is applicable to both the design of an unsigned MAC unit and the design of a signed MAC unit. Note that the only differences between the unsigned MAC unit and the signed MAC unit are the PPM structure and the α-bit addition mechanism. Moreover, the proposed MAC architecture is also applicable to the systolic array (for performing the matrix multiplication). Implementation data show that, compared with the systolic array based on the conventional PE (i.e., the conventional MAC architecture), the systolic array based on the proposed PE (i.e., the proposed MAC architecture) can greatly reduce both circuit area and power consumption under the same timing constraint.

III. RESULTS AND ANALYSIS DISCUSSION

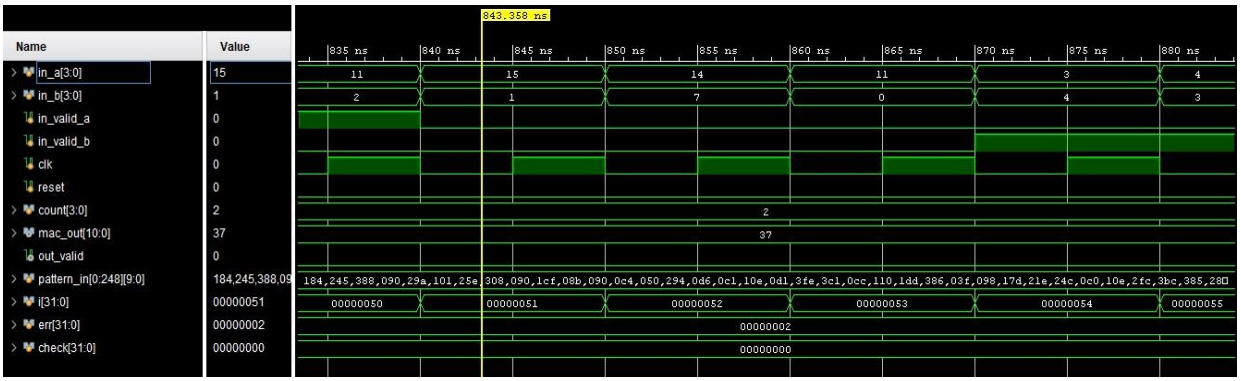

Fig1.MAC Unit output simulation.

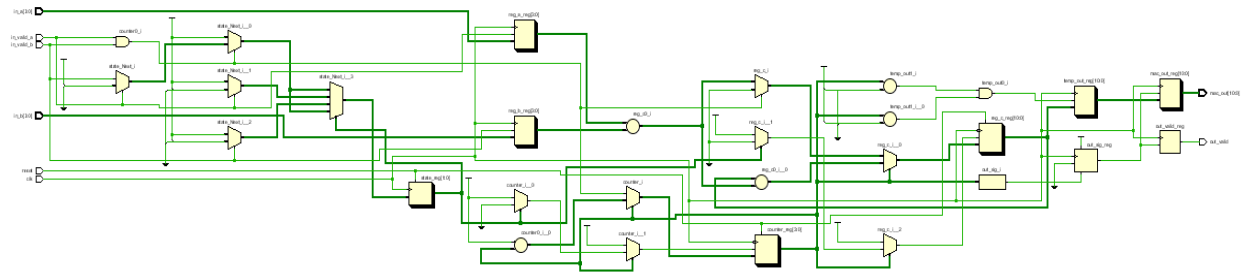

Fig2.RTL Schematic of MAC Unit

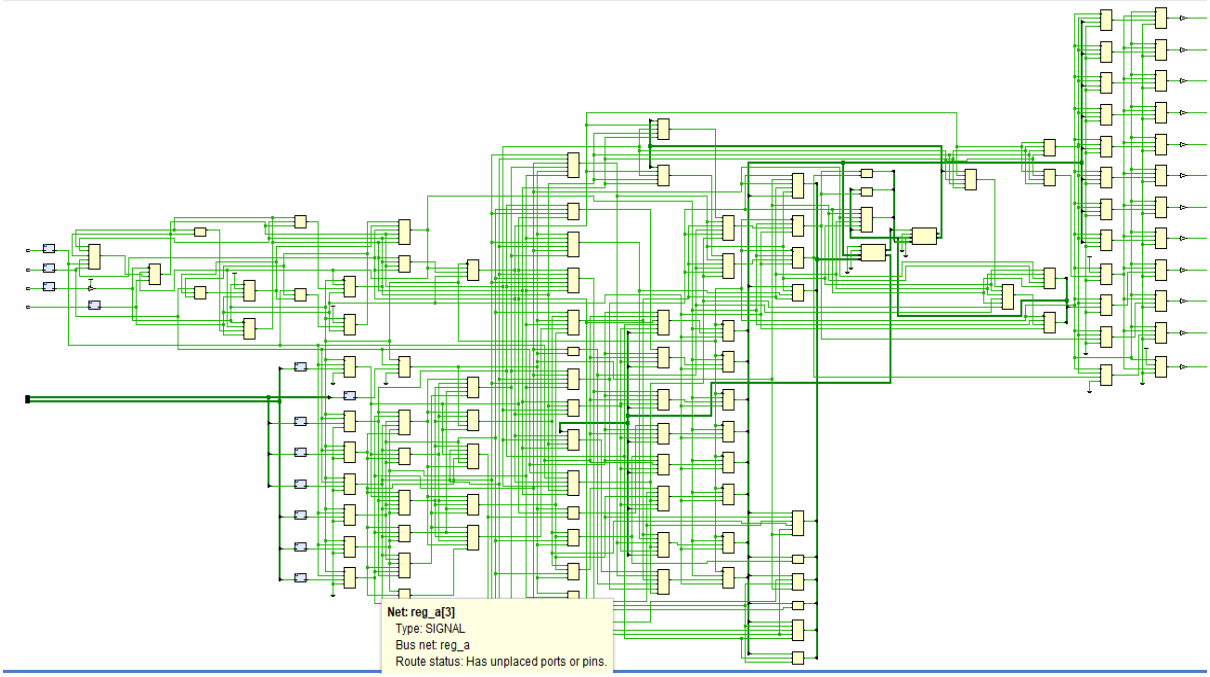

Fig3.Synthesized output of MAC Unit.

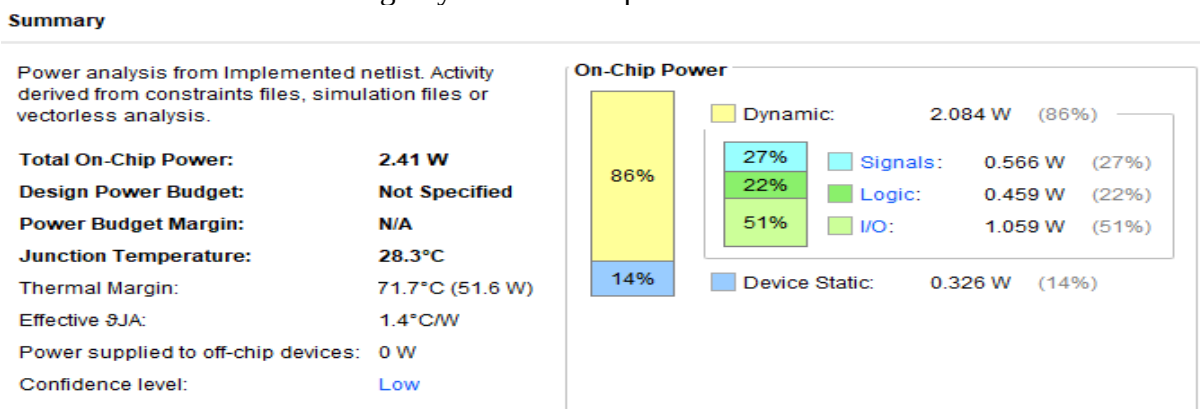

Fig4.Power Report of MAC Unit.

Resource	Utilization	Available	Utilization %
LUT	50	87840	0.06
FF	49	175680	0.03
IO	24	204	11.76
BUFG	1	352	0.28

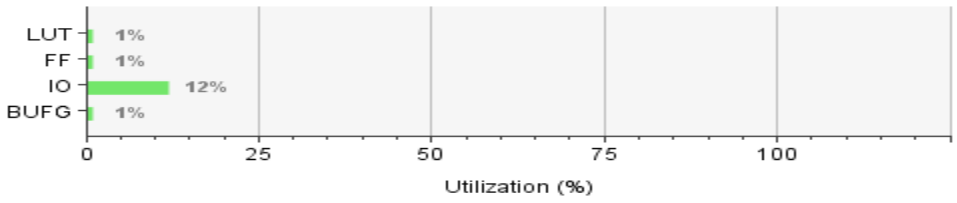

Fig5.Area LUTS and IOBs Utilization Summary

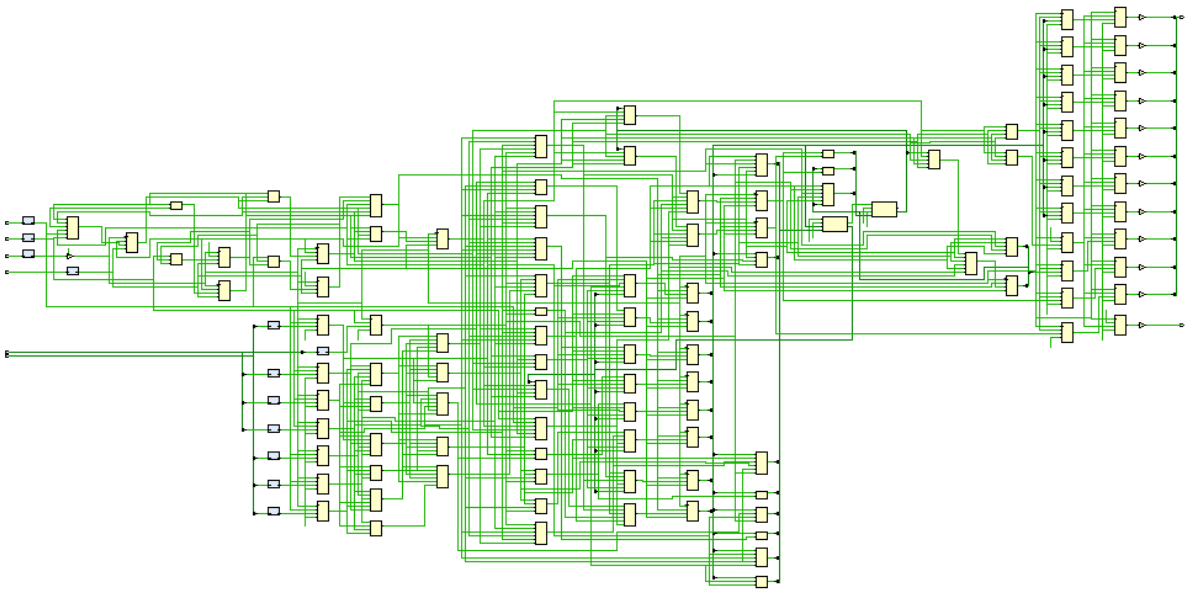

Fig6.Optimized and Implemented Design of MAC Unit.

CONCLUSION

In summary, the integration of additions and accumulations within a unified architecture represents a promising approach to enhancing the computational efficiency of MAC units. By streamlining the computation process and optimizing resource utilization, our proposed design offers significant improvements in terms of area, power, and performance. Future research directions may include exploring advanced optimization techniques and hardware-software co-design methodologies to further enhance the capabilities of integrated MAC units in high-performance computing systems.

REFERENCES

[1] A. Abdelgawad, “Low Power Multiply Accumulate Unit (MAC) for Future Wireless Sensor Networks”, in Proc. IEEE Int. Symp. Sensors Applications, Galveston, TX, 2013.
[2] H.O. Ahmed, M. Ghoneima and M. Dessouky, “Concurrent MAC Unit Design using VHDL for Deep Learning Networks on FPGA”, in Proc. IEEE Int. Conf. Computer Applications and Industrial Electronics, Penang, Malaysia, 2018.
[3] V. Camus, C. Enz and M. Verhelst, “Survey of Precision-Scalable Multiply-Accumulate Units for Neural-Network Processing”, in Proc. IEEE Int. Conf. Artificial Intelligence Circuits and Systems, Hsinchu, Taiwan, 2019.
[4] W.J. Townsend, E.E. Swartzlander, J.A. Abraham, “A Comparison of Dadda and Wallace Multiplier Delays”, in Proc. SPIE Annual Meeting Optical Science and Technology, San Diego, CA, 2003.
[5] K.L.S. Swee and L.H. Hiung, “Performance Comparison Review of 32-Bit Multiplier Designs”, in Proc. IEEE Intelligent and Advanced Systems, Kuala Lumpur, Malaysia, 2012, pp. 836-841.

[6] S. Asif and Y. Kong, "Design of an Algorithmic Wallace Multiplier using High Speed Counters", in Proc. IEEE Int. Conf. Computer Engineering & Systems, Cairo, Egypt, 2015, pp. 133-138.
[7] C.W. Tung and S.H. Huang, “Low-Power High-Accuracy Approximate Multiplier Using Approximate High-Order Compressors”, in Proc. IEEE Int. Conf. Communication Engineering and Technology, Nagoya, Japan, 2019.
[8] C. Nagendra, M.J. Irwin and R.M. Owens, “Area-Time-Power Tradeoffs in Parallel Adders”, IEEE Trans. Circuits and Systems -- II: Analog and Digital Signal Processing, vol.. 43, no. 10, pp. 689-702, Oct. 1996.
[9] J. Saini, S. Agarwal and Aditi Kansal, “Performance, Analysis and Comparison of Digital Adders”, in Proc. IEEE Int. Conf. Advances in Computer Engineering and Applications, Ghaziabad, India, 2015, pp. 81-83.
[10] L. Pilato, S. Saponara and L. Fanucci, “Performance of Digital Adder Architectures in 180nm CMOS Standard-Cell Technology”, in Proc. IEEE Int. Conf. Applied Electronics, Pilsen, Czech Republic, 2016.
[11] P. Jebashini, R. Uma, P. Dhavachelvan and H.K. Wye, “ A Survey and Comparative Analysis of Multiply-Accumulate (MAC) Block for Digital Signal Processing Application on ASIC and FPGA”, Journal of Applied Science, vol. 15, no. 7, pp. 934-946, 2015.
[12] P.A. Patil and C. Kulkarni, “ A Survey on Multiply Accumulate Unit”, in Proc. IEEE Int. Conf. Computing Communication Control and Automation, Pune, India, 2018.
[13] P.F. Stelling and V.G. Oklobdzija, “Implementing Multiply Accumulate Operation in Multiplication Time,” in Proc. IEEE Int. Symp. Computer Arithmetic, Asilomar, CA, 1997, pp. 99–106.
[14] A. Abdelgawad and M. Bayoumi, “High Speed and Area-Efficient Multiply Accumulate (MAC)