IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

AXBMS: Approximate Radix-8 Booth Multipliers For High-Performance FPGA-Based Accelerators

[1]Mahesh Bolishetti, [2]Appidi Deepika, *[3]Madipalli Sumalatha

[1,2,3]Department of ECE,

[1,2,3]Siddhartha Institute of Technology & Sciences, Narapally, Ghatkesar, Medchal-Malkajgiri, Telangana, India

Abstract: The majority of the design efforts for near radix-8 Booth multipliers have so far focused on systems that rely on ASICs. The performance improvement that would be achieved with ASIC-based systems would be far greater than with FPGA-based hardware accelerators when using these multipliers, since they are defined as approximations. Reason being, FPGAs and ASICs are fundamentally different in their architectural concepts. Using high-performance approximation radix-8 Booth multipliers, this short proposes FPGA-based solutions to address this gap. Hence, two approximations of radix-8 Booth multipliers, AxBM1 and AxBM2, are suggested. Utilizing the 6-input lookup tables (LUTs) and supporting carry chains on FPGAs to their maximum potential requires the use of approximation. The delay performance of AxBM2 is 49% better than that of the previous top FPGA-targeted design. By combining truncation with AxBM2's complementary errors function; we were able to obtain energy savings of up to 60%. We can more precisely resolve the previous state-of-the-art error-energy Pareto front and get higher energy gains for any given error constraint. In an example study with the suggested multipliers and Sobel edge detection, AxBM2 was able to detect 98.45 percent of edges while reducing energy consumption by 26.41 percent.

Index Terms **- Radix-8, Booth Multiplier, Accelerator.**

I. INTRODUCTION

The use of approximation multiplier circuits in ASIC-based systems has been the subject of research for over ten years [1]. The use of ASIC-targeted unsigned multipliers in FPGA-based systems has recently been explored in [2]. The results demonstrate that when implemented using FPGAs, ASIC-based multipliers do not significantly improve performance. This is because the internal structure of FPGAs was not taken into account during the straight translation of designs based on ASICs. The level of design granularity provided by an ASIC is at the gate level, whereas an FPGA gives it at the LUT level. Research on approximate multipliers using field-programmable gate arrays (FPGAs) is rare and under-researched. Modern field-programmable gate arrays (FPGAs) allow for high-performance multiplication, and two examples are the Xilinx Ultrascale+ and the Intel Startix-10. In terms of floating-point and fixed-point operations, these DSP blocks can handle it all. As shown in [3], using DSP blocks can reduce performance for specific applications. This is because the DSP blocks have a fixed placement and a defined bit width, which causes this. As a result, having both hard-core DSP blocks and logic-based soft multipliers is ideal. There is a wealth of literature detailing methods for creating precise soft multipliers for use in FPGA-based systems. Xilinx FPGAs have a radix-4 Booth multiplier that is correct, as implemented in [4]. The array-like design that has been suggested can do a multiply-accumulate operation with no extra hardware and utilises half as much slice resources. An approach to field-programmable gate arrays (FPGAs) developed by Intel, known as multiplier regularisation, was presented in reference [5]. An area reduction of 10% to 35% relative to the Intel Megafunction IP is achieved with the proposed approach, which also makes parameterization possible.

A lot of new work has been focused on approximate multipliers for systems that use FPGAs. Using basic approximation blocks, [8] has developed fabric-based FPGA higher-order multipliers. Because the carry propagation channel is smaller, the area is reduced by 43.66 percent compared to the exact multiplier. In comparison to lookup table based multiplier IPs, the power-delay-area product is increased by as much as 7.1 times when using the proposed designs on FPGA

II. Literature Review

The convolution method, which requires a great deal of multiply-accumulate calculations, has recently grown substantially due to advancements in neural networks and image processing. Hardware acceleration of convolutional neural networks (CNNs) is best accomplished with field-programmable gate arrays (FPGAs) due to their versatility and low power consumption [6]. When convolutional process units are directly implemented on FPGA, the programmable logic blocks, rather than logic gates, carry out the necessary addition and multiplication operations. These operations have been thoroughly investigated for ASIC-based systems. The fundamental design differences between FPGA and ASIC mean that it could lead to the wasteful waste of hardware resources. Accelerators based on field-programmable gate arrays (FPGAs) are primarily concerned with improving and speeding up convolutional process units.

While digital signal processors (DSPs) from major tech companies like Intel and AMD can boost multipliers' performance, their limited quantity and fixed locations can lead to increased latency and unnecessary routing delays when used directly on field-programmable gate arrays (FPGAs) [7]. Manually rearranging the necessary FPGA resources to increase performance gains may be feasible for low-complexity applications. Although there may be an abundance of DSP blocks, increasing the performance of multipliers could further improve implementation efficiency when working with large-scale neural networks like VGG-16 [8]. Applications that depend on DSPs, such as Nova and Viterbi decoder, may suffer performance degradation when utilised frequently because of their fixed placements, according to previous study [9]. In addition, convolutional process units frequently employ 8 × 8 or 16 × 16 multipliers, and the fixed bit-width multiplier in digital signal processing, usually 25 × 18, is certain to cause latency. To implement CPUs on FPGA, new approaches are required.

There is a lot of literature on FPGA-based convolutional processing units, although most of it focuses on optimising adders or multipliers. To further reduce hardware cost and improve energy efficiency, approximate methods have been employed, disregarding the precise intermediate computations. For example, optimal approximation logarithmic multipliers were used in their study on convolutional process units [10,11]. After transforming the equivalent weights into a canonical signed-digit code, utilized a multiplier for every code [12]. Using the Karatsuba technique and random calculation, in [13] and [14] were able to obtain approximate multipliers. Furthermore, it was suggested in [15] to use approximate adders for optimising convolutional process units. Despite the fact that hardware resources are saved, the performance of a CNN may be negatively affected by the inevitable loss of accuracy as compared to accurate multipliers.

III. Proposed Design

3.1. Radix-4 Booth Multiplier and Its Sign Bit Extension

Using a Wallace tree for partial product compression requires expanding the symbol bits since the bit-width of multiplication is equal to the sum of the bit-widths of the multipliers. Following this, the extension of the sign bit of the radix-4 Booth multiplier will become clear.

Step 1. It is necessary to invert the sign bit of each partial product.

Step 2. Take the lowest sign bit and add 1.

Step 3. To the preceding bit of the sign bit, add 1 for each partial product.

Figure 1 shows the result of using the sign bit expansion strategy to produce the 8 × 8 radix-4 Booth multiplier. However, owing to intrinsic architectural peculiarities, implementing this multiplier directly on FPGA will result in needless resource loss.

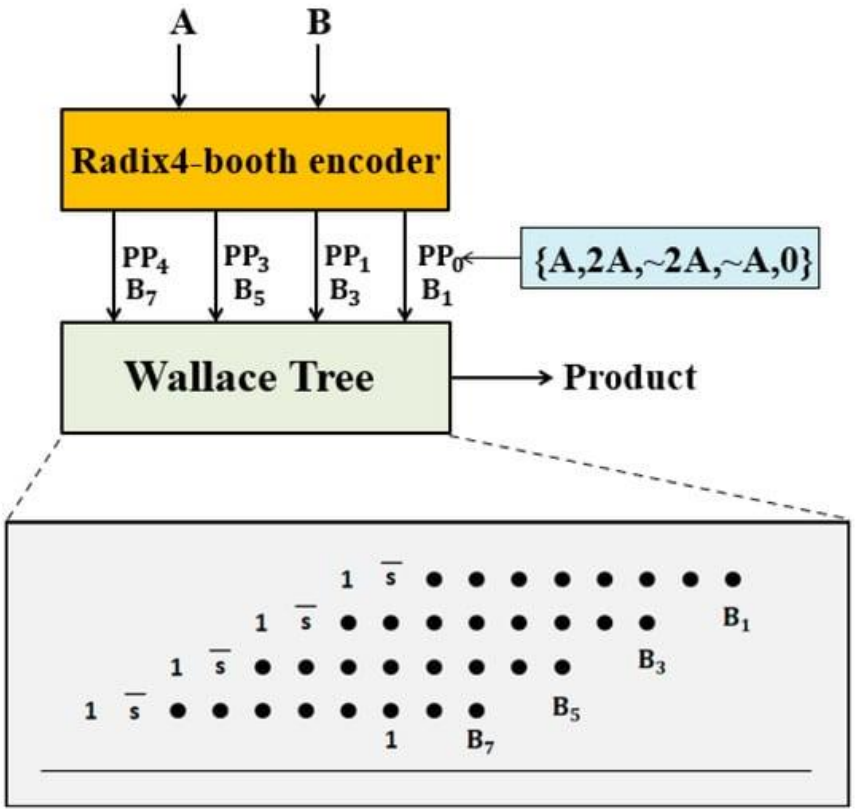

Fig 1. Booth multiplier with 8 × 8 radix-4, where S is the sign bit of the partial products.

3.2. LUT-Based Optimization of Radix-4 Booth Multiplier

Compared with the logic gates that make up the fundamental units of the radix-4 Booth multiplier, computing blocks based on field-programmable gate arrays (FPGAs) are adjustable logic blocks, mostly composed of lookup tables (LUTs) and carry chains. Carry chains are used for addition, and a truth table held by LUT maps inputs and outputs. Hence, two optimisation strategies for the convolutional process units based on FPGAs can be suggested.

Even though a fully utilized LUT usually has five inputs and two or six outputs, several three-input and four-input one-output LUTs will be formed when the radix-4 Booth multipliers are directly implemented on FPGA. As a result, many ports will remain idle when the typical Wallace tree's 3-2 compression is used. For radix-4 Booth multipliers, the multi-input layout of a LUT always requires more output ports than input ones. It is crucial to prioritise the complete utilisation of the output ports throughout implementation. One example is the 8 × 8 radix-4 Booth multiplier, where the intermediate carry is considered both an input and an output, and all partial products are inputs. The results of multiplication are perceived as outputs. There are 126 inputs and 57 outputs, which makes it more like a 5:2 ratio instead of a 6:1 ratio. The demand for the output ports of a LUT with many inputs is always greater than the demand for the input ports in this scenario.

IV. Results and Analysis

The error-energy Pareto charts for each of the approximate multipliers that were investigated are shown in Figure 2. The ASIC-targeted design exhibits degradation in performance when mapped to FPGAs, but ABM1 has the lowest MRED.

Figure 3 shows that for the same mean error, AxBM2 obtains the same energy gains as state-of-the-art designs.

Table 1 compares the proposed approximation multipliers to previous efforts in terms of power savings and quality (PSNR and SSIM).

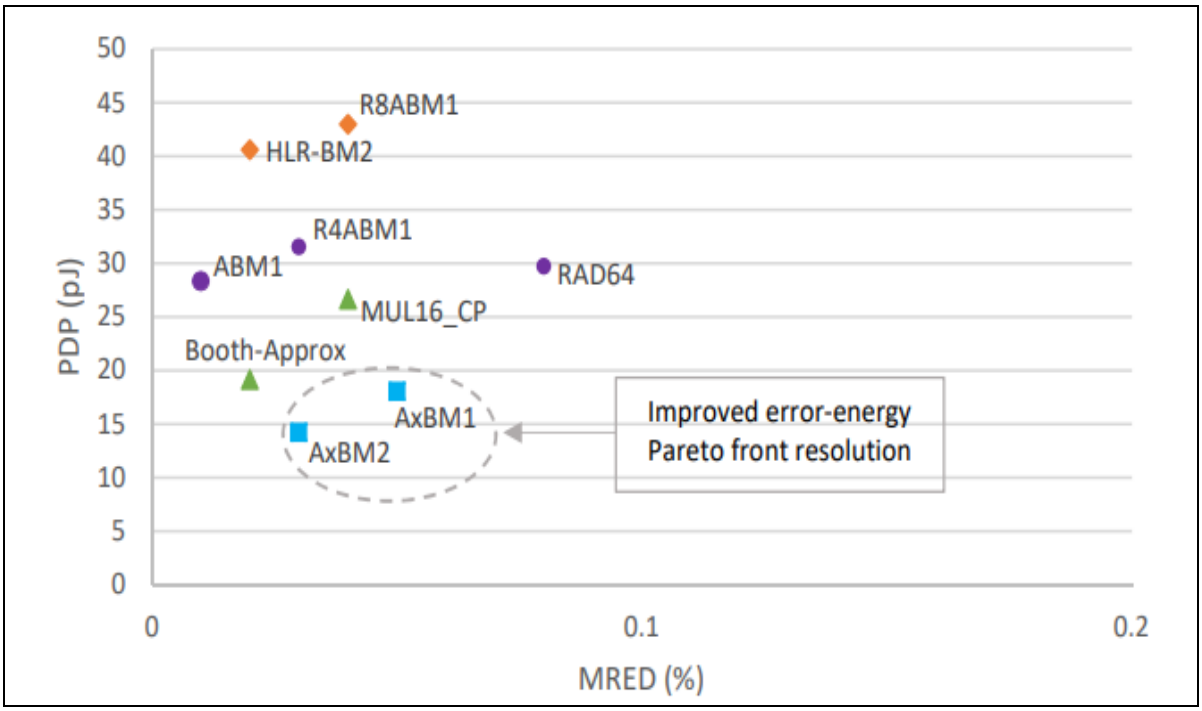

Fig. 2: Pareto evaluation of the inexact multipliers based on error-energy.

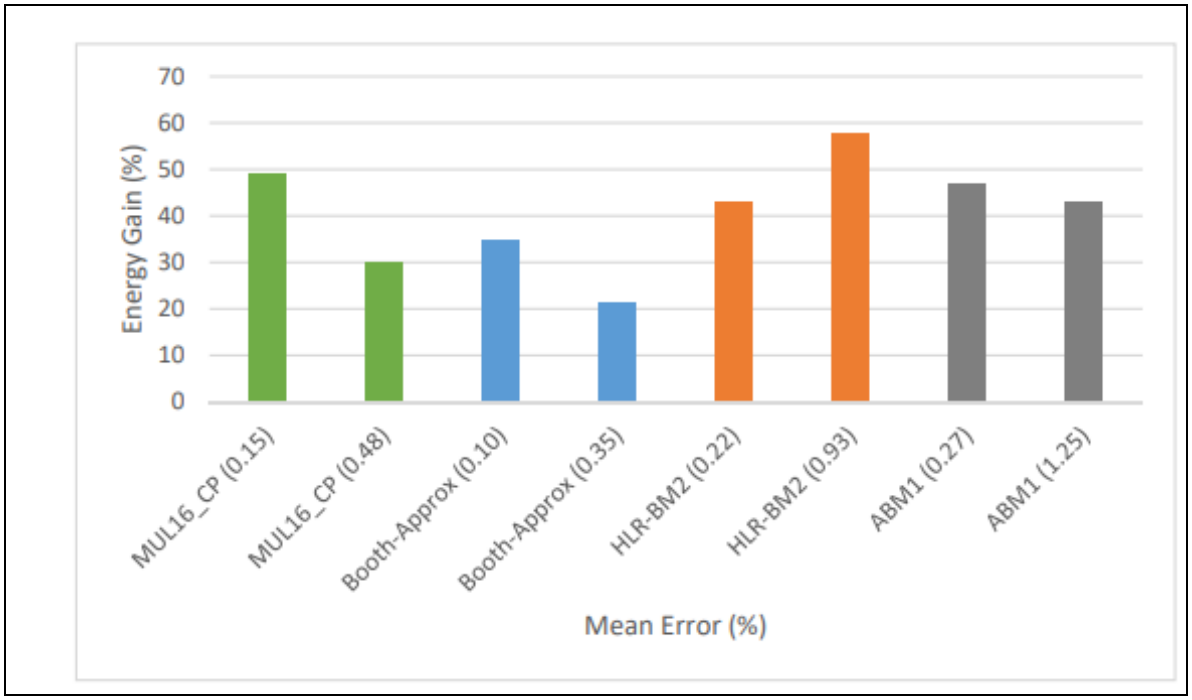

Fig. 3: Energy gains are achieved by AxBM2 when compared to state-of-the-art multipliers with the same error constraint.

Table 1. Comparison of 8 × 8 multipliers' power savings and quality at 100MHz and 1.2V.

Works	PSNR (dB)	SSIM	Power (mW)	Power Savings (%)
Exact Mult.	∞	1.0000	1.440	0.0
DSP Mult.	∞	1.0000	1.116	22.5
M8 [19]	35.14	0.9959	1.872	-30.0
M8 [14]	15.98	0.7779	1.644	-14.2
M8 [30]	28.88	0.963	1.404	2.5
M8 [29]	50.51	0.9996	2.136	-48.3
M8 [17]	44.98	0.9992	1.380	4.2
M8-Ca [23]	14.63	0.8111	1.920	-33.3
M8-Cc [23]	14.61	0.8098	1.404	2.5
M8_CP14_2	40.40	0.9983	1.128	21.7
M8_CP14_3	40.40	0.9983	1.224	15.0
M8_CP14_4	40.41	0.9983	1.248	13.3
M8_CP14_5	43.01	0.9987	1.320	8.3
M8_CP14_6	46.81	0.9989	1.368	5.0
M8_CP13_2	39.10	0.9978	0.912	36.7
M8_CP13_3	39.52	0.9980	0.936	35.0
M8_CP13_4	39.87	0.9981	0.996	30.8
M8_CP13_5	42.31	0.9985	1.032	28.3
M8_CP13_6	45.87	0.9987	1.176	18.3

V. Conclusion

This paper laid out the steps to take when designing a convolution accelerator. At first, two approaches were presented for optimising the radix-4 Booth multiplier that were efficient with resources; these approaches used LUTs and carry chains on FPGA. Then, an alternative to rounding off partial products was suggested: a generic multiply-accumulate structure. The given techniques can be implemented using the Xilinx FPGA Virtex-7 xczu3cg-sfvc784-1-e. The optimized multipliers outperformed the Vivado area-optimized multiplier IP, which had a maximum area reduction (LUT) of 51%. The proposed multiply-accumulate structure cut LUT by up to 22.7% compared to the existing methods.

References

[1] W. Liu, F. Lombardi and M. Shulte, ”A Retrospective and Prospective View of Approximate Computing,” Proceedings of the IEEE, vol. 108, no. 3, pp. 394-399, 2020.

[2] H. Pettenghi, F. Pratas and L. Sousa, ”Method for Designing Efficient Mixed Radix Multipliers,” Circuits, Systems and Signal Processing, vol. 33, no. 10, pp. 3165-3193, 2014.

[3] W. Liu, T. Cao, P. Yin, Y. Zhu, C. Wang, E. E. Swartzlander, and F. Lombardi, ”Design and Analysis of Approximate Redundant Binary Multipliers,” IEEE Transactions on Computers, vol. 68, no. 6, 2019.

[4] S. Ullah, S. S. Murthy and A. Kumar, ”SMApproxLib: Library of FPGA-based Approximate Multipliers,” in Proc. 55th Annual Design Automation Conference (DAC), 2018, pp. 1-6.

[5] I. Kuon and J. Rose, ”Measuring the Gap Between FPGAs and ASICs,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 26, no. 2, pp. 203-215, 2007.

[6] Mittal, S. A survey of FPGA-based accelerators for convolutional neural networks. Neural Comput. Appl. 2020, 32, 1109–1139.

[7] Wang, D.; Xu, K.; Guo, J.; Ghiasi, S. DSP-efficient hardware acceleration of convolutional neural network inference on FPGAs. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 2020, 39, 4867–4880.

[8] Ullah, S.; Sripadra, S.; Murthy, J.; Kumar, A. SMApproxLib: Library of FPGA-based approximate multipliers. In Proceedings of the IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 24–28 June 2018; pp. 1–6. [Google Scholar]

[9] Xilinx LogiCORE IP v12.0. Available online: https://www.xilinx.com/support/documentation/ip_documentation /mult_gen/ v12_0/pg108-mult-gen.pdf (accessed on 21 July 2023).

[10] Lentaris, G. Combining arithmetic approximation techniques for improved CNN circuit design. In Proceedings of the IEEE International Conference on Electronics, Circuits and Systems (ICECS), Glasgow, UK, 23–25 November 2020; p. 9294869.

[11] Ebrahimi, Z.; Ullah, S.; Kumar, A. LeAp: Leading-one detection-based softcore approximate multipliers with tunable accuracy. In Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC), Beijing, China, 13–16 January 2020; pp. 605–610.

[12] Csordás, G.; Fehér, B.; Kovácsházy, T. Application of bit-serial arithmetic units for FPGA implementation of convolutional neural networks. In Proceedings of the International Carpathian Control Conference (ICCC), Szilvasvarad, Hungary, 28–31 May 2018; pp. 322–327.

[13] Zhang, H.; Xiao, H.; Qu, H.; Ko, S. FPGA-based approximate multiplier for efficient neural computation. In Proceedings of the IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), Gangwon, Republic of Korea, 1–3 November 2021; pp. 1–4.

[14] Lammie, C.; Azghadi, M. Stochastic computing for low-power and high-speed deep learning on FPGA. In Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), Sapporo, Japan, 26–29 May 2019; pp. 1–5.

[15] Thamizharasan, V.; Kasthuri, N. High-Speed Hybrid Multiplier Design Using a Hybrid Adder with FPGA Implementation. IETE J. Res. 2021, 69, 2301–2309.