Design and Implementation of Radix-8 Booth Multipliers for High-Performance FPGA-Based Accelerators: An Estimate (AxBMs) on FPGA

Mr.B.Ajantha Reddy [1] Mr.M.Ramana Reddy [2] Nakka Venkata Pavan Kumar [3] Sivapuram Sravan [4] Pola Sandeep Reddy [5] Boggu Ganesh Reddy [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 project, ASIC-based platforms have been the focus of previous ideas for approximate radix-8 Booth multipliers. These multipliers cannot produce equivalent performance benefits when applied to FPGA-based hardware accelerators because they are based on an approximation as described for ASIC-based systems. The inherited architectural distinctions between FPGAs and ASICs are the cause of this. By suggesting high performance approximation radix-8 Booth multipliers whose designs are aimed at FPGA-based systems, this brief fills this gap. Thus, AxBM1 and AxBM2, two approximate radix-8 Booth multipliers, are proposed. The 6-input lookup table (LUT) and related carry chains of the FPGAs are completely utilized because of the implementation of approximation. When compared to the previous best FPGA-targeted design (Booth Approx), AxBM2 shows a 49% improvement in latency. Complementing errors is one of AxBM2's advantages; when paired with truncation, this ability can result in energy savings of up to 60%. Better energy gains are also possible for a given error constraint thanks to an enhanced resolution of the prior state-of-the-art error-energy Pareto front. In an example study, the suggested multipliers are used with Sobel edge detection; AxBM2 found 98.45% of the edges with 26.41% less energy.

Keywords: Booth Multipliers (Radix-AxBs),5G Communications, FPGA, Verilog HDL, Xilinx Tool.

I.INTRODUCTION

for more than a decade approximate multiplier circuits have been studied for ASIC-based systems [1]–[3]. Recently in [4], ASIC-targeted unsigned multipliers have been investigated for FPGA-based systems. The results show the limited performance gains of ASIC-based multipliers when implemented by FPGAs. This is due to the direct translation of ASIC-based designs with no consideration for the internal structure of FPGAs. An FPGA provides design granularity at LUT-level whereas an ASIC provides design granularity at gate-level. Approximate multipliers for FPGA-based systems have not been extensively studied in the technical literature, there are only a few designs as FPGA-based approximate multipliers.

State-of-the-art FPGAs such as Xilinx Ultrascale+ and Intel Startix-10 provide hard DSP blocks that can be used for high-performance multiplication. Both fixed point and floating-point operations can be performed using these DSP blocks. However, [5] has presented that for some applications the use of DSP blocks result in a degraded performance. This is due to the fixed-bit width and fixed locations of the DSP blocks. Therefore, it is always desirable to have logic-based soft multipliers along with hard-core DSP blocks. An extensive body of research has been

reported on the design of exact soft multipliers for FPGA-based systems. Reference [6] has implemented accurate radix-4 Booth multiplier on Xilinx FPGAs. The proposed array-like architecture uses 50% less slice resources and provides a multiply-accumulate operation without additional hardware. In [7], a so-called multiplier regularization has been introduced using the adaptive logic module (ALM) of Intel FPGAs. The proposed implementation is amenable to parametrization and a 10% to 35% reduction in area has been reported compared to the Intel Megafunction IP.

Recently, some research works have been reported on the design of approximate multipliers for FPGA-based systems. Reference [8] has used elementary approximate blocks to design higher order multipliers for an FPGA-based fabric. The carry propagation path is reduced thus, achieving 43.66% less area compared to the exact multiplier. Ebrahimi et al. [9] have presented a novel architecture of approximate logarithmic multiplier (LeAp) tailored for FPGAs. LeAp outperforms the exact 32x32 accurate multiplier by achieving 42.1% improvement in power. Reference [10] has presented approximate logic compressors (3:2 and 4:2) based (8/16/32) bit multipliers. The proposed designs when implemented on FPGA report higher gains (up to 7.1x) in power-delay-area product compared to lookup table-based multiplier IPs. In [11], signed radix-4 accurate and approximate Booth multipliers have been proposed that targets FPGA-based systems. An improvement of up to 63% in the area is achieved compared to Xilinx Vivado's area-optimized multiplier IP. Recently, a machine learning based methodology has been proposed to determine the set of Pareto-optimal FPGA-based approximate circuits (ACs) from the state-of-the-art ASIC-based approximate design. This scheme reduces the design exploration time to achieve high-performance and low-area in FPGA-based implementations [12].

However, to the author's best knowledge; there has been no extensive research to date on an approximate radix-8 Booth multiplier for FPGA-based systems. The work presented in [11] is applicable to the design of radix-4 Booth multiplier for FPGA-based systems. As the radix-8 algorithm generates a small number of partial products and thus, it requires fewer adders to accumulate partial products compared to the radix-4 algorithm, we believe that FPGA-targeted approximate radix-8 multipliers need to be studied. Hence this is the target of this work. The main contributions of our work are summarized as follows:

1) Two approximate radix-8 Booth multipliers (called AxBM1 and AxBM2) based on approximate Booth encoders are proposed for FPGA-based systems.

2) A simplified partial product generator (PPG) based on the Booth-encoded signals is proposed. The proposed PPG is mapped using a single 6-input LUT, so different from the conventional PPG which requires two (6-input) LUTs for its implementation.

3) The resolution of the state-of-the-art error-energy Pareto front is improved, as the approximations are carried out by considering the underlying structure of FPGAs, i.e., lookup tables (LUTs).

4) Compared to the recently proposed FPGA-based approximate multiplier [11], AxBM2 achieves an improvement of 26% in terms of power-delay product (PDP) with comparable error characteristics.

2.LITERATURE SURVEY

A Booth multiplier consists of Booth encoding, partial product generator, partial product accumulation and final addition. Previous designs of approximate radix-8 Booth multipliers have

addressed the issue of generating odd multiplicands in the radix-8 encoder [14], [18]. However, these approximate designs when mapped to FPGAs require nearly the same number of LUTs as an exact design. Table I summarizes the LUTs utilization of the radix-8 encoder, odd multiplicand, and PPG in both the exact and existing approximate radix-8 Booth multipliers [14], [18]. The hardware utilization is calculated based on the 6-input lookup tables (referred to as LUT6_2) of Xilinx FPGAs.

A Radix-8 Booth Multiplier is a type of binary multiplier that uses radix-8 (base-8) encoding to reduce the number of partial product rows generated during the multiplication process. The Booth algorithm, originally designed for base-2 (binary) multiplication, is extended to radix-8 to further optimize the multiplication process.

II.EXISTING SYSTEM AND PROPOSED SYSTEM

1.EXISTING SYSTEM

A Booth multiplier consists of Booth encoding, partial product generator, partial product accumulation and final addition. Previous designs of approximate radix-8 Booth multipliers have addressed the issue of generating odd multiplicands in the radix-8 encoder [14], [18]. However, these approximate designs when mapped to FPGAs require nearly the same number of LUTs as an exact design. Table I summarizes the LUTs utilization of the radix-8 encoder, odd multiplicand, and PPG in both the exact and existing approximate radix-8 Booth multipliers [14], [18]. The hardware utilization is calculated based on the 6-input lookup tables (referred to as LUT6_2) of Xilinx FPGAs (Fig. 2).

In a 16x16 multiplier (Fig. 1) the total number of dots is 107, so these many PPGs are required to generate all partial products. As an exact 1-bit PPG (with 8-inputs) takes two LUTs then the generation of 107 partial product bits require 214 LUTs. The radix-8 encoder also requires two LUTs, one LUT to generate (×1 and ×2) whereas a second LUT to generate (×3 and ×4). The 16-bit multiplier uses six encoders (Fig. 1) therefore, in total 12 LUTs are utilized. The 3C multiplicand (referred to as the hard multiple) requires a carry propagation adder (C + 2C), and 16 LUTs are utilized [19]. Reference [14] has proposed an approximate recoding adder to generate the 3C multiple, whereas [18] has approximated the odd multiples to their nearest power of two; the corresponding reduction (when mapped to FPGA) has been reported in Table I. The PPG in [14] has eight inputs, while the PPG in [18] has seven inputs. However, when mapped to the FPGAs, they both use the same number of LUTs as the exact design due to use of 2 LUTs per PPG. In this brief, Booth encoding and partial product generation are investigated for FPGA-based systems. A methodology is presented to reduce the number of LUTs both for the radix-8 encoder and PPG. The proposed encoder requires only three signals (compared to five signals in the exact encoder). Moreover, the proposed PPG is implemented using a single LUT, because it has six inputs (compared to eight inputs in the exact PPG), thus, achieving a 50% reduction in the PPG LUTs.

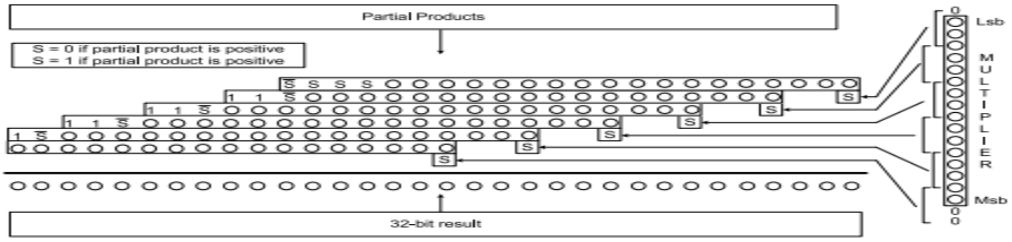

Fig. 1. Dot Diagram of a 16×16 multiplier using Radix-8 Booth Algorithm partial product bit; s/s: sign bit/inversion.

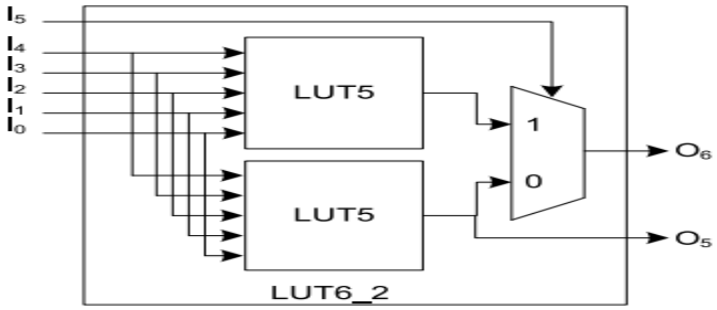

Fig. 2. Xilinx 6-input look up table.

2.PROPOSED SYSTEM

A. Approximate Radix-8 Booth Encoders

The recently proposed approximate Booth encoder [18] is based on four Booth-encoded signals (s, ×1, ×2 and ×4); however, by mapping this encoder to FPGAs, performance is not improved, because it still requires two LUTs for its implementation, same as required in the exact radix-8 encoder.

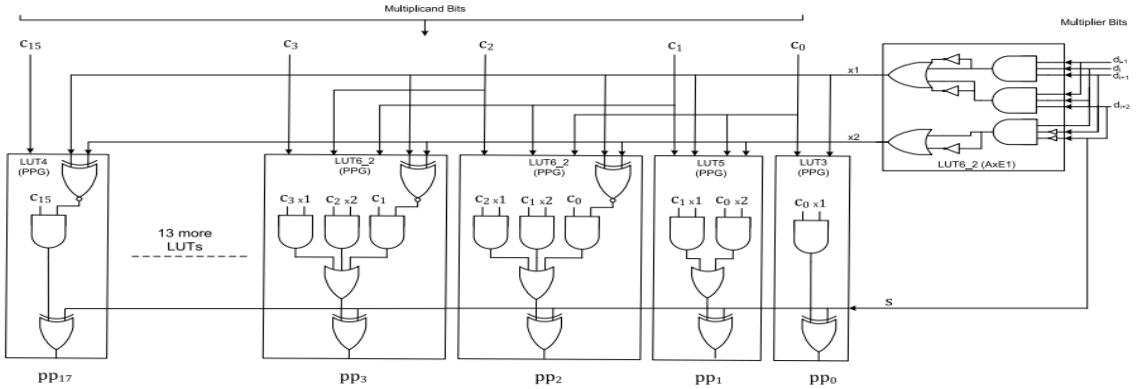

Fig. 3. Partial product generation logic of a 16x16 Booth multiplier using FPGA-targeted approximate radix-8 encoder (AxE1) and the partial product generator.

B. FPGA-targeted Radix-8 Partial Product Generator

The proposed simpler partial product generator is based on 6-input signals that can be mapped using a 6-input LUT (Fig. 3). ck, ck−1 and ck−2 are the multiplicand bits whereas s, ×1 and ×2

are the Booth encoded signals. The ×4 signal is generated by XNORing the ×1 and ×2 signals. As discussed in Section II, the radix-8 16×16 Booth multiplier requires the generation of 107 partial product bits. Therefore, using the proposed PPG all required bits are generated using 107 LUTs compared to the exact PPG which requires 214 LUTs. Hence, an improvement of 50% in terms of complexity is achieved.

C. Radix-8 Booth Multiplier Variants

The FPGA-targeted encoders and the PPG presented in the previous section are utilized to propose two variants of Booth multipliers (called AxBM1 and AxBM2). AxBM1 is based on the AxE1 encoder while AxBM2 uses the AxE2 encoder. Fig. 3 shows the partial product selection logic for a 16-bit Booth multiplier; the proposed encoder and PPG are both implemented using a single 6-input LUT. Therefore, 19 LUTs (18LUTs for PPG and 1 LUT for encoder) are used to generate one of the six partial product rows for a 16-bit radix8 Booth multiplier. It has been shown in [19] that the use of LUT primitives reduce the hardware and latency of synthesized designs. Thus, to achieve significant hardware and performance gains for the proposed approximate multiplier designs, a LUT primitive-based design is pursued. AxBM2 has the feature of complementing positive and negative errors; therefore, the error will be less than AxBM1. This feature is exploited to achieve a hardware-accuracy trade-off by applying truncation in the AxBM2 multiplier. Reference [14] has shown that for a radix-8 Booth multiplier truncation up to nine least significant bits (LSBs) provides a favourable hardware-accuracy trade-off. Therefore, in AxBM2 nine LSBs are truncated to save additional power and reduce the delay. To compensate the error generated by the truncated lower part of AxBM2, an additional ‘1’ (average error) is added to the 10th bit of the AxBM2 multiplier. A carry save adder (CSA) tree is used to reduce the partial product’s matrix to an addition of only two operands. Finally, a carry propagation adder (CPA) is utilized for the final computation of the binary result (produced by the CSA tree). To investigate an efficient implementation of a Booth multiplier for the FPGA-based systems, the error is bounded by accurately performing the partial product accumulation and the final addition.

III. RESULTS AND ANALYSIS DISCUSSION

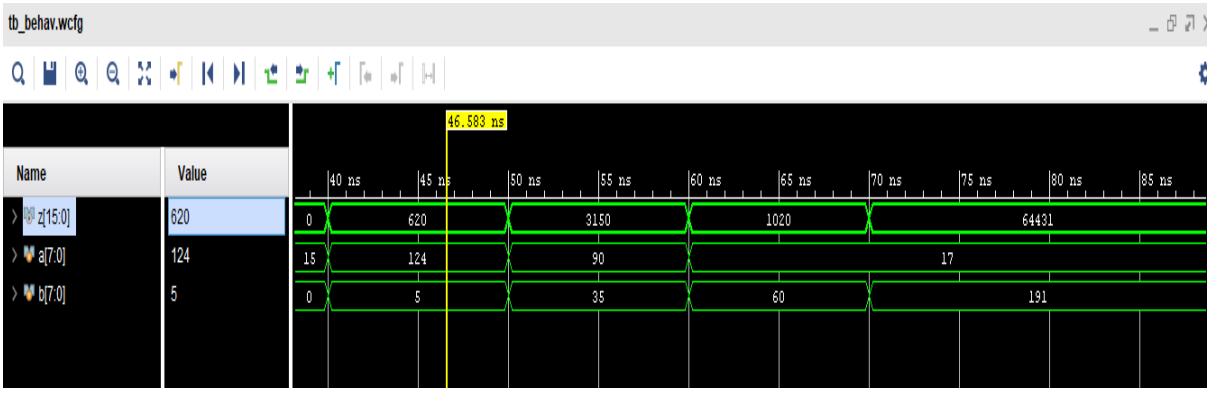

FIG1.SIMULATION OUTPUT

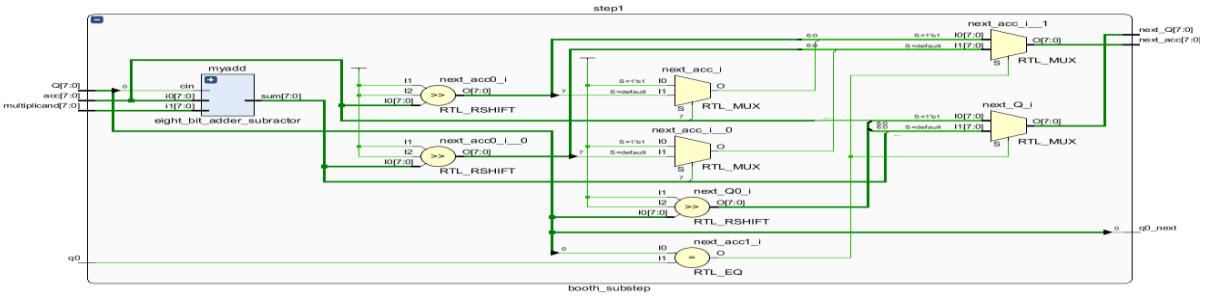

FIG2.RTL SCHEMATIC

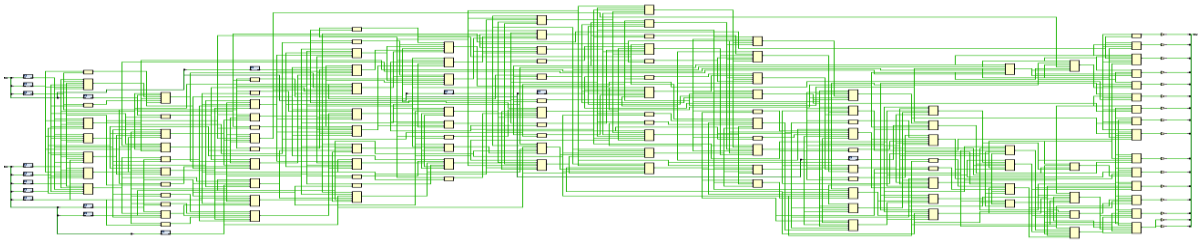

FIG3.SYNTHESIS DESIGN

Resource	Utilization	Available	Utilization %
LUT	87	117120	0.07
IO	32	204	15.69

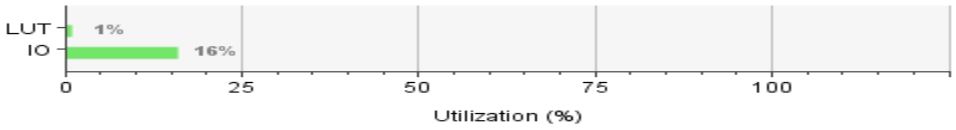

FIG.4.UTILIZATION OF LUTS AND FF AND IOBs

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**24.483 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**58.5°C**
Thermal Margin:	41.5°C (29.7 W)
Effective ϑJA:	1.4°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

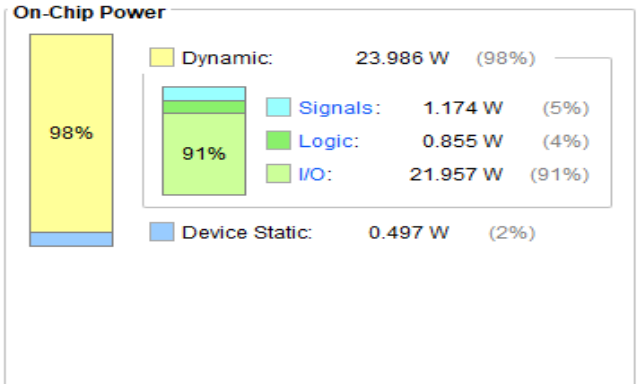

FIG5.POWER REPORT

Conclusion

In this project, the design of an approximate radix-8 Booth multiplier for FPGA-based systems has been studied. Two approximate designs (AxBM1 and AxBM2) based on approximate Booth encoders have been proposed. The proposed approximation reduces the hardware for the Booth-encoded signals and lead to a design of a simpler partial product generator which is efficiently

mapped using a single 6-input LUT. The proposed multipliers improve the resolution of the previous error-energy Pareto front, as approximations are proposed to account for the design granularity of FPGA's.

REFERENCES

[1] W. Liu, F. Lombardi, and M. Shulte, "A retrospective and prospective view of approximate computing [point of view]," Proc. IEEE, vol. 108, no. 3, pp. 394–399, Mar. 2020.
[2] H. Pettenghi, F. Pratas, and L. Sousa, "Method for designing efficient mixed radix multipliers," Circuits Syst. Signal Process., vol. 33, no. 10, pp. 3165–3193, 2014.
[3] W. Liu et al., "Design and analysis of approximate redundant binary multipliers," IEEE Trans. Comput., vol. 68, no. 6, pp. 804–819, Jun. 2019.
[4] S. Ullah, S. S. Murthy, and A. Kumar, "SMApproxLib: Library of FPGA-based approximate multipliers," in Proc. 55th Annu. Design Autom. Conf. (DAC), San Francisco, CA, USA, 2018, pp. 1–6.
[5] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 26, no. 2, pp. 203–215, Feb. 2007.
[6] M. Kumm, S. Abbas, and P. Zipf, "An efficient softcore multiplier architecture for Xilinx FPGAs," in Proc. 22nd Symp. Comput. Arithmetic (ARITH), Lyon, France, 2015, pp. 18–25. [7] M. Langhammer and G. Baeckler, "High density and performance multiplication for FPGA," in Proc. 25th Symp. Comput. Arithmetic (ARITH), Amherst, MA, USA, 2018, pp. 5–12.
[8] Y. Guo, H. Sun, and S. Kimura, "Small-area and low-power FPGA based multipliers using approximate elementary modules," in Proc. 25th Asia South Pac. Design Autom. Conf. (ASP-DAC), Beijing, China, 2020, pp. 599–604.
[9] Z. Ebrahimi, S. Ullah, and A. Kumar, "LeAp: Leading-one detection based softcore approximate multipliers with tunable accuracy," in Proc. 25th Asia South Pac. Design Autom. Conf. (ASP-DAC), Beijing, China, 2020, pp. 605–610.
[10] N. Van Toan and J. Lee, "FPGA-based multi-level approximate multipliers for high-performance error-resilient applications," IEEE Access, vol. 8, pp. 25481–25497, 2020.
[11] S. Ullah, H. Schmidl, S. S. Sahoo, S. Rehman, and A. Kumar, "Areaoptimized accurate and approximate softcore signed multiplier architectures," IEEE Trans. Comput., vol. 70, no. 3, pp. 384–392, Mar. 2021, doi: 10.1109/TC.2020.2988404.
[12] B. S. Prabakaran, V. Mrazek, Z. Vasicek, L. Sekanina, and M. Shafique, "ApproxFPGAs: Embracing ASIC-based approximate arithmetic components for FPGA-based systems," 2020. [Online]. Available: arXiv:2004.10502.
[13] 7 Series FPGAs Configurable Logic Block, User Guide, Xilinx, San Jose, CA, USA, 2016. [Online].Available:https://www.xilinx.com/support/documentation/userguides/ug4747SeriesCLB.pdf
[14] H. Jiang, J. Han, F. Qiao, and F. Lombard, "Approximate radix-8 booth multipliers for low-power and high-performance operation," IEEE Trans. Comput., vol. 65, no. 8, pp. 2638–2644, Aug. 2016.
[15] W. Liu, L. Qian, C. Wang, H. Jiang, J. Han, and F. Lombardi, "Design of approximate radix-4 Booth multipliers for error-tolerant computing," IEEE Trans. Comput., vol. 66, no. 8, pp. 1435–1441, Aug. 2017.

[16] V. Leon, G. Zervakis, D. Soudris, and K. Pekmestzi, “Approximate hybrid high radix encoding for energy-efficient inexact multipliers,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 26, no. 3, pp. 421–430, Mar. 2018.
[17] S. Venkatachalam, E. Adams, H. J. Lee, and S.-B. Ko, “Design and analysis of area and power efficient approximate booth multipliers,” IEEE Trans. Comput., vol. 68, no. 11, pp. 1697–1703, Nov. 2019.
[18] H. Waris, C. Wang, and W. Liu, “Hybrid low radix encoding-based approximate booth multipliers,” IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 67, no. 12, pp. 3367–3371, Dec. 2020.
[19] B. S. Prabakaran et al., “DeMAS: An efficient design methodology for building approximate adders for FPGA-based systems,” in Proc. Design, Autom. Test Eur. Conf. Exhibit. (DATE), Dresden, Germany, Mar. 2018, pp. 917–920.