AxBMs: Approximate Radix-8 Booth Multipliers for High-Performance FPGA-Based Accelerators

Haroon Waris, *Graduate Student Member, IEEE*, Chenghua Wang, Weiqiang Liu, *Senior Member, IEEE*, and Fabrizio Lombardi, *Life Fellow, IEEE*

(Invited Paper)

***Abstract*—The focus of existing designs on approximate radix-8 Booth multipliers has been on ASIC-based platforms. These multipliers are based on an approximation as defined for ASIC-based systems, so they cannot achieve comparable performance gains when used for FPGA-based hardware accelerators. This is due to the inherited architectural differences between FPGAs and ASICs. This brief bridges this gap by proposing high-performance approximate radix-8 Booth multipliers whose designs target FPGA-based systems. Hence, two approximate radix-8 Booth multipliers (referred to as AxBM1 and AxBM2) are proposed. Approximation is implemented such that the 6-input lookup table (LUT) and the associated carry chains of the FPGAs are fully utilized. AxBM2 exhibits 49% improvement in delay compared to the previous best FPGA-targeted design (Booth-Approx). AxBM2 has the advantage of complementing errors; this feature has been combined with truncation to achieve up to 60% in energy savings. Moreover, the resolution of the previous state-of-the-art error-energy Pareto front is improved, such that better energy gains can be achieved for a given error constraint. As a case study, the proposed multipliers are applied to the application of Sobel edge detection, AxBM2 detected 98.45% edges with energy savings of 26.41%.**

***Index Terms*—Booth encoding, lookup table, ASIC, FPGA, Pareto front.**

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

Manuscript received January 31, 2021; accepted March 8, 2021. Date of publication March 11, 2021; date of current version April 30, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 62022041 and Grant 61871216; and in part by the Six Talent Peaks Project in Jiangsu Province under Grant 2018-XYDXX-009. This brief was recommended by Associate Editor J. R. Cavallaro. *(Corresponding author: Weiqiang Liu.)*

Haroon Waris, Chenghua Wang, and Weiqiang Liu are with the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China (e-mail: haroonwaris@nuaa.edu.cn; chwang@nuaa.edu.cn; liuweiqiang@nuaa.edu.cn).

Fabrizio Lombardi is with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115 USA (e-mail: lombardi@ece.neu.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSII.2021.3065333.

Digital Object Identifier 10.1109/TCSII.2021.3065333

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.

The rest of this brief is organized as follows. Section II articulates the problem statement. The proposed FPGA-targeted approximate Booth multipliers are described in Section III. The proposed multipliers are compared with the state-of-the-art approximate multipliers in Section IV and its applicability is demonstrated in Section V in the Sobel edge detector. Finally, Section VI concludes this brief.

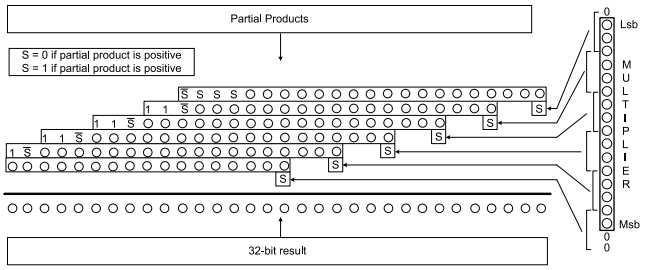


Fig. 1. Dot Diagram of a 16×16 multiplier using Radix-8 Booth Algorithm. ○: partial product bit; s/$\bar{s}$: sign bit/inversion.

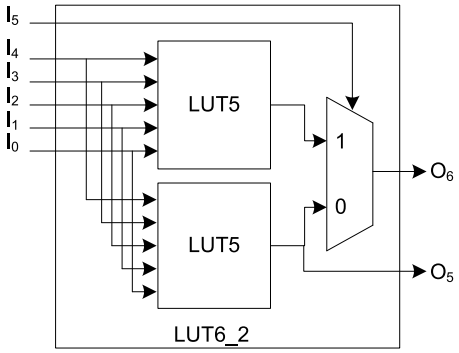


Fig. 2. Xilinx 6-input look up table.

TABLE I
LUTs UTILIZATION IN EXACT AND APPROXIMATE RADIX-8 ENCODER, ODD MULTIPLICAND (3C) AND PPG

Designs	Encoder	Odd Multiple	PPG	Total (LUTs)
Exact	12	16	214	238
R8ABM1 [14]	12	8	214	230
HLR-BM2 [18]	12	-	214	222
Proposed	6	-	107	113

II. PROBLEM FORMULATION

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 ($\times 1$ and $\times 2$) whereas a second LUT to generate ($\times 3$ and $\times 4$). 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 (Table I).

III. PROPOSED DESIGNS

A. Approximate Radix-8 Booth Encoders

The recently proposed approximate Booth encoder [18] is based on four Booth-encoded signals (s, $\times 1$, $\times 2$ and $\times 4$); 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.

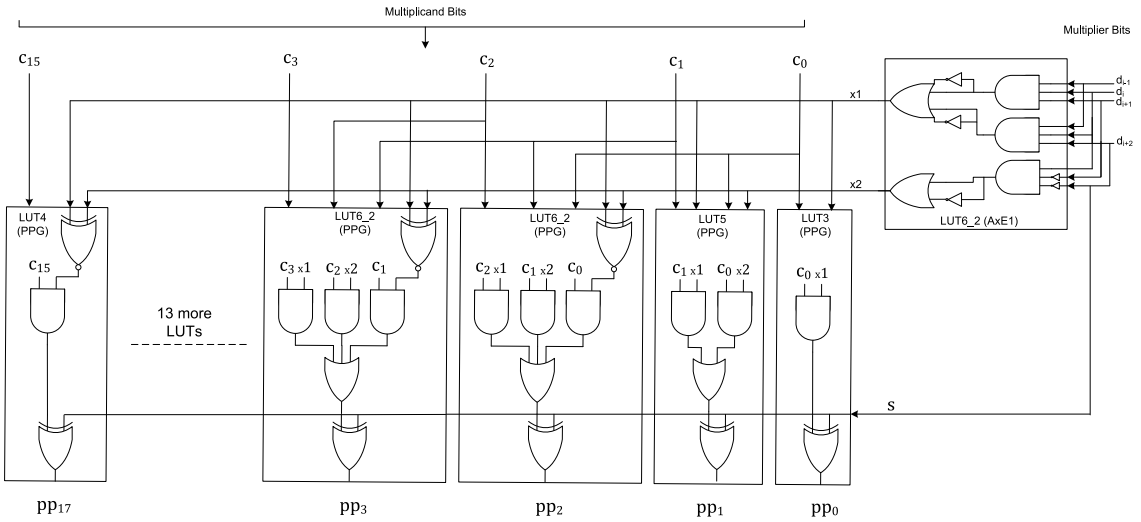


Fig. 3. Partial product generation logic of a 16x16 Booth multiplier using FPGA-targeted approximate radix-8 encoder (AxE1) and the partial product generator.

TABLE II
ERROR DISTANCE OF PROPOSED APPROXIMATE RADIX-8 BOOTH ENCODERS (AXE1 AND AXE2)

Exact	AxE1				AxE2				ED
	s	×1	×2	PP_x	s	×1	×2	PP_x	
0	0	1	0	+1C	0	1	0	+1C	1
+1C	0	1	0	+1C	0	1	0	+1C	0
+1C	0	1	0	+1C	0	1	0	+1C	0
+2C	0	0	1	+2C	0	0	1	+2C	0
+2C	0	0	1	+2C	0	0	1	+2C	0
+3C	0	0	0	+4C	0	0	0	+4C	1
+3C	0	0	0	+4C	0	0	0	+4C	1
+4C	0	0	0	+4C	0	0	0	+4C	0
-4C	1	0	0	-4C	1	0	0	-4C	0
-3C	1	0	0	-4C	1	0	1	-2C	1
-3C	1	0	0	-4C	1	0	1	-2C	1
-2C	1	0	1	-2C	1	0	1	-2C	0
-2C	1	0	1	-2C	1	0	1	-2C	0
-1C	1	1	0	-1C	1	1	0	-1C	0
-1C	1	1	0	-1C	1	1	0	-1C	0
0	1	1	0	-1C	1	1	0	-1C	1

In this brief, two new Booth encoders (called AxE1 and AxE2) are proposed for FPGA-based systems. The approximation is implemented, such that the encoder can easily be mapped using a single 6-input LUT. The generation of the $\times 4$ encoded signal in an exact radix-8 encoder is a complex operation as it requires two XOR gates (2-input), one XNOR gate (2-input) and one OR gate (3-input). This complexity can be reduced by using small approximation in the encoder. The partial products against the inputs {(0000) and (1111)} are approximated to $\pm$1C (Table II). Then, the $\pm$4C multiplicand can easily be generated by XNORing $\times 1$ and $\times 2$, so there is no need to separately generate the $\times 4$ encoded signal (Table II). In AxE1, the absolute value of the approximate product is greater than the exact counterpart, because $\pm$3C are approximated to $\pm$4C multiplicands. AxE2 is designed such that for a specific logic circuit, positive and negative errors complement each other. Thus, the positive 3C multiplicand is approximated to 4C whereas the negative 3C multiplicand is approximated to 2C. Moreover, the error difference is always kept as one (Table II) against all approximated values in both the encoders (AxE1 and AxE2). Therefore, the proposed encoders reduce the hardware, while keeping the error in bounds.

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). c_k, c_{k-1} and c_{k-2} are the multiplicand bits whereas s, $\times 1$ and $\times 2$ are the Booth encoded signals. The $\times 4$ signal is generated by XNORing the $\times 1$ and $\times 2$ 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 radix-8 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 favorable hardware-accuracy trade-off. Therefore, in AxBM2 nine LSBs are truncated to save additional power and reduce the delay. To compensate

TABLE III
COMPARATIVE PERFORMANCE ANALYSIS OF EXACT AND APPROXIMATE BOOTH MULTIPLIERS

Design	Area (LUTs)	Delay (ns)	Power (mW)	PDP (pJ)	MRED (%)	NMED (10^{-5})	PRED (%)
Vivado IP	286	4.27	8.04	34.35	-	-	-
Exact radix-8	320	4.92	9.13	44.91	-	-	-
AxBM1	194	3.68	4.90	18.03	0.05	0.86	97.24
AxBM2	161	3.45	4.12	14.21	0.03	0.71	98.63
MUL16_CP [10]	219	4.78	5.57	26.62	0.04	11.8	99.15
Booth-Approx. [11]	137	6.88	2.78	19.14	0.02	0.31	96.27
R8ABM1 [14]	312	4.78	8.99	42.97	0.04	1.93	98.95
R4ABM1 [15]	265	4.12	7.65	31.51	0.03	0.84	98.51
RAD64 [16]	251	4.01	7.41	29.71	0.08	4.79	99.58
ABM1 [17]	245	3.92	7.23	28.34	0.01	0.54	99.68
HLR-BM2 [18]	304	4.56	8.90	40.58	0.02	0.66	99.12

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.

IV. EVALUATION AND COMPARISON

The proposed multipliers (AxBM1 and AxBM2) are implemented using Verilog HDL and synthesized using Xilinx Vivado 2018.2 on 7VX330T device of the Virtex-7 family. Area, delay and power are reported using the Vivado simulator and power analyzer. The behavioral models of the multipliers are developed in MATLAB to find the error metrics (PRED, NMED and MRED) against random 16-bit signed inputs (uniformly distributed). When targeting FPGAs, most of the published works have been on the design of unsigned approximate multipliers [4], [8], [9], [12]. For a fair comparison, only existing signed approximate designs (as applicable to FPGA-based systems) are considered [10], [11]. Moreover, (AxBM1 and AxBM2) are also compared with implementations of existing Booth multipliers [14]–[18].

Compared to the Vivado's IP, the proposed multipliers show a reduction of up to 43% in area (as measured by the number of LUTs). The small increase in the LUTs utilization of AxBM2, compared to Booth-Approx, is due to the parallel computation of the encoded partial products. However, this result in achieving a better delay and PDP, improvements of 49% and 26%, respectively, compared to the previous best FPGA-targeted design. MUL16_CP (16-bit recursive multiplier) has been designed using an 8-bit multiplier. Recently, it has been reported [20] that recursive approximate multipliers exhibit a high delay overhead due to long critical path. Therefore, compared to Vivado's IP, MUL16_CP has a large delay. Radix-8 approximate multipliers [14], [18] have similar hardware metrics as the exact radix-8 Booth multiplier. This finding is consistent with the analysis presented in Section II that ASIC-targeted multipliers when mapped to FPGAs, exhibit low-performance. Radix-4 approximate multipliers [15], [16], [17] have better hardware characteristics because the partial product generator can be mapped to a 6-input LUT. Moreover, the generation

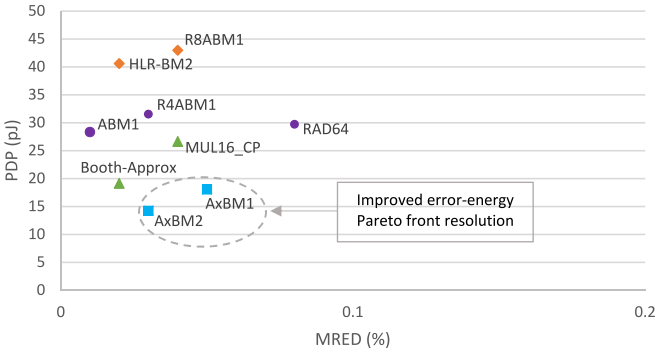


Fig. 4. Error-energy Pareto analysis of the inexact multipliers.

of odd multiplicands is not required; thus, having a 10-12% improvement in delay compared to ASIC-targeted radix-8 Booth multipliers.

Fig. 4 presents the error-energy Pareto plots for all considered approximate multipliers. ABM1 has the least MRED; however, the ASIC-targeted design when mapped to FPGAs shows a degraded performance. In AxBM1, the product is always higher than the exact value; therefore, it exhibits a higher MRED than AxBM2. Compared to MUL16_CP, both having similar MRED and designed for FPGA-based systems, AxBM1 shows an improvement of 32% in terms of energy. While, AxBM2 exhibits the least energy among all designs, an improvement of 26% is achieved compared to the previous best FPGA-targeted design (Booth-Approx). AxBM2 has the feature of complementing positive and negative errors as well as the least 9-bits are truncated. Therefore, it delivers the best error-energy trade-off and improves the resolution of the previous state-of-the-art Pareto front. Fig. 5 shows the energy gains of AxBM2 when the same mean error is introduced as in state-of-the-art designs of [10], [11], [17], [18]. Specifically, we have compared for small (i.e., up to 0.27%) and larger errors (i.e., up to 1.25%) to investigate the energy consumption at different error bounds. These results show that AxBM2 provides significant gains ranging from 21.46% to 57.93%.

V. CASE STUDY: SOBEL EDGE DETECTOR

The discrete Sobel operator is used to compute the gradient of the image intensity. It consists of two 3×3 kernels, the first kernel horizontally computes the variations in brightness while the second kernel computes in the vertical orientation. The performance of edge detection is evaluated by replacing the

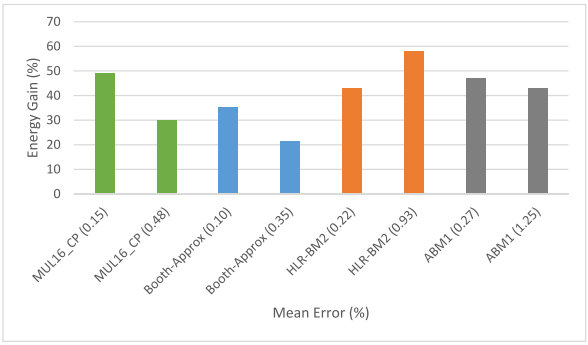
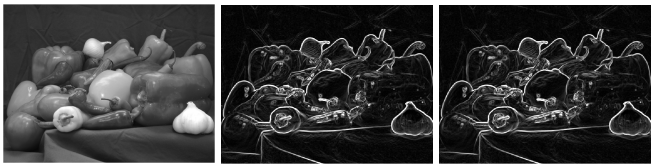


Fig. 5. AxBM2 energy gains with respect to current state-of-the-art multipliers under the same error constraint.



(a) (b) (c)

Fig. 6. (a) Input image, (b),(c) Edge detection using AxBM1/2.

TABLE IV
ENERGY GAINS AND ACCURACY RESULTS OF SOBEL EDGE DETECTION USING APPROXIMATE MULTIPLIERS

Multiplier	Energy Gain (%)	Edges Detected (%)
AxBM1	21.25	97.45
AxBM2	26.41	98.45
MUL16_CP [10]	20.31	97.81
Booth-Approx [11]	22.47	98.96
ABM1 [17]	18.56	99.23
HLR-BM2 [18]	16.95	98.70

exact multipliers with the proposed approximate multipliers. The percentage of the edges detected (this, is defined as the edges detected using an approximate multiplier over those detected using the exact radix-8 multiplier) and energy savings are used as quality metric. Fig. 6 shows the input image (16-bit pixels are used) and the output images produced by the gradient function using the proposed approximate multipliers. AxBM1 is the less accurate design, however it detects 97.45% edges with energy savings of 21.25% (Table IV). AxBM2 detects 1% more edges (98.45%) with an increase in energy savings of 26.41%.

VI. CONCLUSION

In this brief, 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, i.e., the LUTs. AxBM2 has achieved 49% and 26% improvements in delay and PDP compared to a previous best FPGA-targeted design (Booth-Approx). Moreover, the efficiency of the proposed multipliers is shown for an image processing application; energy savings of 26.41% are achieved while detecting 98.45% of the edges.

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, "Area-optimized 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.
[20] H. Waris, C. Wang, W. Liu, J. Han, and F. Lombardi, "Hybrid partial product-based high-performance approximate recursive multipliers," *IEEE Trans. Emerg. Topics Comput.*, early access, Aug. 4, 2020, doi: 10.1109/TETC.2020.3013977.