Power-Efficient and Small-Area Approximate Multiplier Design with FPGA-Based Compressors

Yi Guo[1,2], Xiu Chen[1], Qilin Zhou[1], Heming Sun[3]
[1] *Graduate School of Information, Science and Engineering, Yunnan University, Kunming, China*
[2] *Yunnan Key Laboratory of Intelligent Systems and Computing, Yunnan University, Kunming, China*
[3] *Faculty of Engineering, Yokohama National University, Kanagawa, Japan*

2024 IEEE International Symposium on Circuits and Systems (ISCAS) | 979-8-3503-3099-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/ISCAS58744.2024.10558590

Abstract—**Approximate computing has become an emerging technique to reduce power consumption. In numerous applications, multiplication is a crucial operation, designing it for approximation is effective in optimizing system performance. In this paper, we propose low-power FPGA-based multipliers by employing the novel compressor designs. Given that the compressor is the primary unit in the multiplier, we introduce novel exact and approximate compressors with low-complexity circuits to parallelly accumulate the elements. To flexibly configure the proposed compressors, a compressor-based once-through structure is proposed to 8×8 multipliers. Two variants of the approximate multipliers are provided with different accuracy-hardware trade-offs. Compared with the exact multiplier, the proposed approximate multiplier reduces power by 57.90%, area by 33.80%, and delay by 24.78%. With a similar accuracy loss, the proposed designs save more hardware resources than others. In addition, the effectiveness of approximate multipliers is assessed in image sharpening.**

Keywords—Approximate computing, FPGA-based compressor, Low-power circuit.

I. Introduction

Approximate computing, which trades computation precision for power optimization, is an appealing choice for error-tolerant applications like multimedia processing and machine learning. These applications heavily depend on multiplication operations, thus designing the approximate multiplier is an effective approach to reduce the power consumption of applications and systems.

Generally, multiplication involves three phases: partial product generation, partial product accumulation, and final addition. The partial product accumulation phase typically leads to primary hardware consumption caused by the significant utilization of compressors. Therefore, an approximate compressor design is important to optimize the whole multiplier performance. The state-of-the-art approximate multipliers mainly focus on ASIC-based implementations and are designed by approximate compressors with fewer logic gates [1-4].

Nowadays, with the growing popularity of FPGAs, research on FPGA-targeted approximate multipliers is more and more attractive. There are two main reasons. Firstly, because of the architectural difference between ASICs and FPGAs, the abundant researches in ASIC-based approximate multipliers usually offer asymmetrical gains in FPGA-based accelerators [5]. As reported in [6], ASIC-based designs with gains of energy-delay-product (EDP) more than 30% over the exact multiplier, while only about 10% EDP saving after FPGA-based implementation. Secondly, FPGAs have the superiorities of reconfiguration capability, high energy efficiency, and fast development cycle. More and more multimedia applications tend to use FPGAs to process data. Despite the high performance of multiplication operations provided by DSP blocks in FPGAs, the designs using logic-based soft multipliers are indispensable. Thus, Xilinx and Intel also provide logic-based soft multipliers [7][8]. Based on these factors, designing approximate multipliers with FPGA fabrics is desirable.

In [6], an approximate 4×2 multiplier is introduced by utilizing four LUTs and used to construct 4×4 and 8×8 multipliers. In [9], three approximate units based on LUTs are proposed and applied in the partial product matrix (PPM), generating three approximate multipliers. In [10], three types of 4×4 approximate multipliers are designed, and a wide range of approximate 8×8 multipliers are proposed using 4×4 multipliers. In [11], an exact 4×4 multiplier is proposed, and three approximate multipliers are designed by reorganizing LUTs.

Most of the existing works on FPGA-based approximate multipliers save hardware resources. However, there still are two challenges for low-power designs. One issue is **the absence of approximate compressor design in FPGA-based multipliers**. Considering that the compressor is a power-intensive unit, its approximate design can effectively reduce power. However, because of the port limitations of FPGA fabrics, there are few studies on FPGA-based multipliers with approximate compressors. Another concern is **the decreasing effectiveness of power savings resulting from the recursive construction method**. Building large-size multipliers from small multipliers is a usual method in previous works [6, 9-11]. Nevertheless, as the input width increases, there is a corresponding increase in circuit complexity. These above issues restrict the effectiveness of approximate computing for low-power multiplier designs.

In this paper, we propose low-power FPGA-based multipliers by focusing on the compressor designs. To effectively configure compressors, a compressor-based once-through structure is introduced, which differs from the traditional recursive construction method in previous works. Our contributions are as follows:

- A novel exact compressor is proposed by lagging calculation of the carry result of partial products. The input ports of the LUT can be fully utilized by this carry-lagged compressor.
- An approximate m:2 compressor is introduced to aggregate several elements into two equal-weight vectors. Additionally, a uniform expression for the proposed m:2 compressor is provided to facilitate the expanded usage of the approximate compressor.
- A once-through methodology of the approximate 8×8 multiplier is presented by utilizing the proposed compressors on the partial product accumulation. Two variants of approximate 8×8 multipliers are introduced to meet varying performance requirements.

The rest of the paper is organized as follows: Section II provides the preliminaries. Proposed compressors are introduced in Section III. Section IV shows the proposed approximate 8×8 multipliers. The experimental results are discussed in Section V, and Section VI concludes this paper.

Corresponding authors: Yi Guo, *guoyi@ynu.edu.cn*, Heming Sun, *sun-heming-vg@ynu.ac.jp*.

Authorized licensed use limited to: Sheffield University. Downloaded on November 20,2024 at 09:30:49 UTC from IEEE Xplore. Restrictions apply.

II. Preliminaries

In this paper, we focus on Xilinx 7-series FPGA family devices, which employ a hierarchical architecture with configurable logic blocks (CLBs) as the fundamental units. One CLB consists of two slices, each slice contains four 6-input look-up tables (LUTs), eight storage units for LUT outputs, wide-function multiplexers, and a 4-bit carry chain [12].

6-input LUT is a fundamental component for implementing logic functions. Its implementation includes two types: the first type is named LUT6, as shown in Fig. 1 (a), and Fig. 1 (b) illustrates another type which is LUT6_2. For LUT6, the six inputs of the LUT are programmed to define different logic functions. This means that one LUT6 can represent different logical operations (INIT values). For example, the INIT value '8000000000000000' (hex) means that O is '1' for the input combination of '111111'. For LUT6_2, 5 inputs can be configured, and the outputs O_5 and O_6 are independent.

The structure of the carry chain is shown in Fig. 1 (c). It consists of LUTs, XOR gates, and multiplexers (with bypass signals AX/BX/CX/DX). The carry chain is constructed with four cascading full-adder units. Each unit is driven by the carry-generate signal O_5 and the carry-propagate signal O_6 which are the outputs from LUTs. Hence, the carry chain performs the same functions as the carry-look-ahead adder.

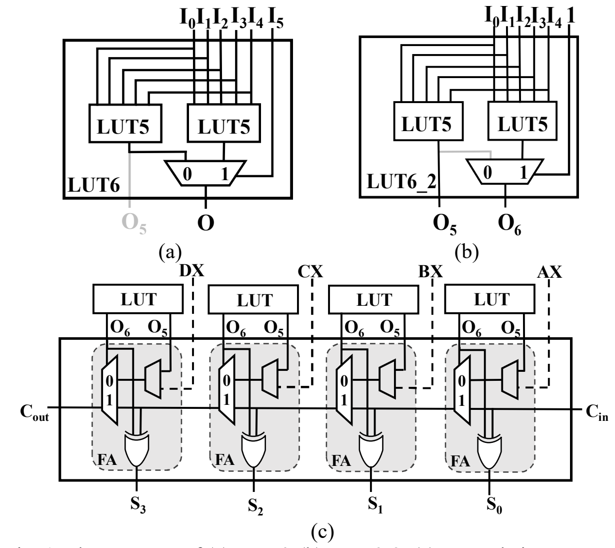

Fig. 1. The structure of (a) LUT6. (b) LUT6_2. (c) carry chain.

III. Proposed Exact and Approximate Compressors

In the multiplier, compressor is a circuit to accumulate and reduce several rows of partial products into two rows for the final adder. To efficiently accumulate partial products, we propose novel exact and inexact compressors by using LUTs.

A. Exact carry-lagged compressor

The processing of carry from previous bit is a major challenge in designing compressors. Especially in FPGA-based designs, using traditional exact compressors not only results in large cascaded circuits, but also greatly increases the usage of LUTs. To address these issues, we propose an exact carry-lagged compressor. As shown in Fig. 2, this compressor consists of two LUT6s, which generate one sum result S, and one carry result C, respectively. In the proposed compressor, two partial products ($I_2 \cdot I_3$ and $I_4 \cdot I_5$) are packed first, and then their carry result is lagged by calculating them in the next adjacent compressor. Instead, the lagged carry (C_{lag}) from the

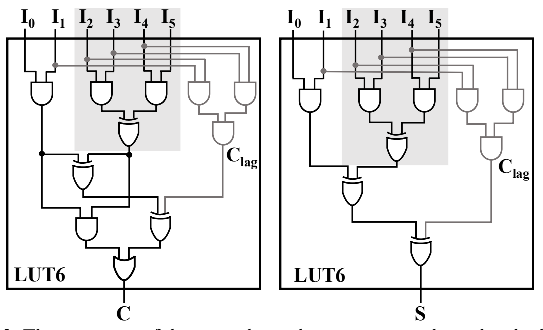

Fig. 2. The structure of the carry-lagged compressor, where the shaded area is the carry-lagged part.

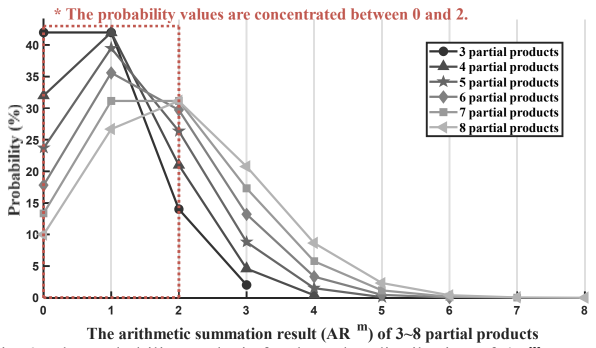

Fig. 3. The probability analysis for the value distribution of AR^m.

previous compressor is calculated in the current compressor by using two partial products ($I_1 \cdot I_2$ and $I_3 \cdot I_4$). Since the input operands are the same, this carry-lagged compressor performs more operations without incurring hardware overhead.

The outputs of the proposed compressor are calculated as:

$$S = (I_0 \cdot I_1) \oplus ((I_2 \cdot I_3) \oplus (I_4 \cdot I_5)) \oplus ((I_1 \cdot I_2) \cdot (I_3 \cdot I_4)), \tag{1}$$

$$C = (I_0 \cdot I_1) \oplus ((I_2 \cdot I_3) \oplus (I_4 \cdot I_5)) \cdot \big((I_1 \cdot I_2) \cdot (I_3 \cdot I_4)\big) + (I_0 \cdot I_1) \cdot ((I_2 \cdot I_3) \oplus (I_4 \cdot I_5)). \tag{2}$$

where notations '$\oplus$', '$\cdot$', and '+' represent the XOR, AND, and OR operations, respectively. Although the partial carry is lagged to compute, the calculation and weight of the carry result are the same as those in the exact compressor.

B. Approximate m:2 compressor

To compress several elements within one stage, we propose an approximate m:2 compressor to sum several elements into two equal-weight outputs.

Assume that the inputs of an n×n multiplier are A_{n-1}~A_0 and B_{n-1}~B_0, which are uniformly and independently distributed. A PPM is generated by applying AND operations on each bit of A with n-bit B. We define the number of elements belonging to the same column as m, and the arithmetic summation result (AR^m) of m elements is described as:

$$AR^m = \sum(t_1, t_2, \ldots, t_m). \tag{3}$$

where notation '$\sum$' means the arithmetic summation and '$t_1, t_2, \ldots, t_m$' are m elements in the same column.

The probability distribution of AR^m for different numbers of partial products is shown in Fig. 3. It can be seen that the probability of AR^m is concentrated on the range of 0~2. This implies that two numbers are sufficient to represent the cumulative result of m elements in the approximate design. Hence, the m:2 compressor is designed to sum m elements into two equal-weight outputs (Y_1, Y_2). The uniform logical expression of Y_1, Y_2 are designed and summarized as:

$$Y_1 = t_1 + t_2 + \cdots + t_m, \tag{4}$$

$$Y_2 = t_1 \cdot (t_2 + \cdots + t_m) + t_2 \cdot (t_3 + \cdots + t_m) + \cdots + t_{m-1} \cdot t_m. \tag{5}$$

Authorized licensed use limited to: Sheffield University. Downloaded on November 20,2024 at 09:30:49 UTC from IEEE Xplore. Restrictions apply.

where notations '+' and '·' represent the OR and AND operations, respectively. The m:2 compressor connects to the carry chains by calculating $Y_1 \cdot Y_2$ as the carry-generate signal (Gen) and $Y_1 \oplus Y_2$ as the carry-propagate signal ($Prop$). These calculations are operated in the same LUT with the m:2 compressor. During partial product accumulation, if the element number is less than 6, the expressions (4) and (5) can be configured in one single LUT6_2.

IV. Proposed Approximate 8×8 Multipliers

In this section, the 8×8 multipliers are introduced with the proposed compressors. For different performance results, two variants of approximate 8×8 multipliers are presented.

A. The compressor-based approximate multiplier 1 (CAM1)

The overall structure of compressor-based approximate multiplier 1 (CAM1) is shown in Fig. 4. The design of CAM1 is implemented in three stages with a once-through structure.

In stage 1, the carry-lagged compressors are used to compress the PPM from 8 rows to 6 rows accurately. The PPM is divided into two layers each with the same carry-lagged compressor operation. One compressor is implemented by two LUT6s and takes three partial products as the inputs. As illustrated in Fig. 4, the shaded area in the proposed compressor indicates the carry lagged part, and their carry calculation is performed in the next column. An example of carry-lagged compressor is shown in column 12 in Fig. 4. This compressor consists of two LUT6s and the inputs are A_7, B_5, A_6, B_6, A_5, and B_7. The outputs of this compressor are calculated as:

$$S_{16} = A_7B_5 \oplus (A_6B_6 \oplus A_5B_7) \oplus (A_6B_5 \cdot A_5B_6), \tag{6}$$

$$C_{16} = A_7B_5 \oplus (A_6B_6 \oplus A_5B_7) \cdot (A_6B_5 \cdot A_5B_6) + A_7B_5 \cdot (A_6B_6 \oplus A_5B_7). \tag{7}$$

where '$A_6B_6 \oplus A_5B_7$' is the carry-lagged part in column 12, and '$A_6B_5 \cdot A_5B_6$' is the lagged carry from column 11. For the lowest column of each layer, one LUT6_2 is used to generate the sum result and the carry calculation is lagged to the next compressor. The second highest column in each layer is accumulated with one LUT6_2. The sum result of this column is calculated by combining it with the lagged carry from the previous compressor.

In stage 2, m:2 compressors are used to accumulate 6 rows into 2 rows and connect to the final adder. Two 3:2 compressors are used in columns 3 and 11, along with three 4:2 compressors on columns 4~6. For columns 7~10, the number of items exceeds the input limitation of LUT6_2, thus requiring extra LUTs to generate the entire element (A_iB_j) in advance. Hence, A_7B_3, A_3B_7, A_2B_7, A_1B_7, A_0B_7, and A_4B_3 are generated first by using three LUT6_2s. Then, one 6:2 compressor is used in column 7, along with three 5:2 compressors applied in columns 8~10. The example of utilizing the m:2 compressor is illustrated in column 7 with the operations as:

$$pp_0 = A_0B_7, pp_1 = A_4B_3, \tag{8}$$

$$Y_{1,7} = S_5 + C_4 + S_{11} + C_{10} + pp_0 + pp_1, \tag{9}$$

$$Y_{2,7} = S_5 \cdot (C_4 + S_{11} + C_{10} + pp_0 + pp_1) + C_4 \cdot (S_{11}+C_{10} + pp_0 + pp_1) + S_{11} \cdot (C_{10} + pp_0 + pp_1) + C_{10} \cdot (pp_0 + pp_1) + pp_0 \cdot pp_1. \tag{10}$$

Note that, except for column 7, each m:2 compressor in columns 8~11 is implemented by only one LUT6_2.

In stage 3, the final adder consists of three carry chains, and the carry-in signal (C_{in1}) is equal to C_0.

B. The compressor-based approximate multiplier 2 (CAM2)

To obtain different performance results, we replaced some compressors with OR operations and proposed a

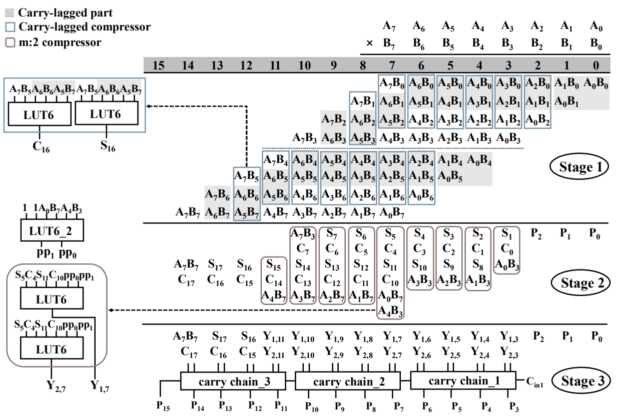

Fig. 4. The structure of the compressor-based approximate multiplier 1 (CAM1).

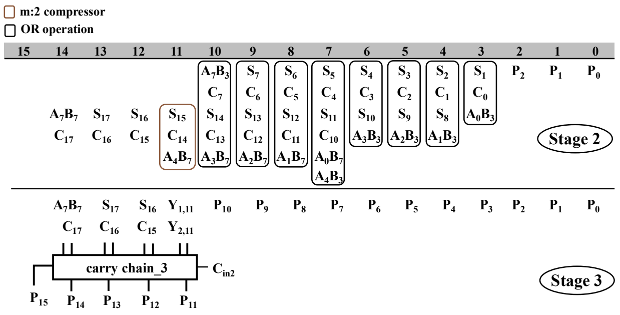

Fig. 5. Stage 2 and stage 3 of the compressor-based approximate multiplier 2 (CAM2).

compressor based approximate multiplier 2 (CAM2). The difference of CAM1 and CAM2 is on stage 2 and stage 3, as shown in Fig. 5. In CAM2, compressors in columns 3~10 are replaced by OR operations, with one LUT6_2 used for each column. The carry from columns 3~10 is partially reserved as the carry-in signal (C_{in2}) and calculated as $C_{in2} = A_7B_3 \cdot (C_7 + S_{14} + C_{13} + A_3B_7) + C_7 \cdot (S_{14} + C_{13} + A_3B_7) + S_{14} \cdot (C_{13} + A_3B_7) + C_{13} \cdot A_3B_7$. This calculation will not increase the LUT resource because it is implemented by the same LUT6_2 used for column 10.

V. Experiment Results And Discussions

The proposed multipliers were implemented using Verilog HDL and synthesized using Xilinx Vivado 2018.3 for the XC7VX485T device of the Virtex-7 family. To demonstrate the contributions of the proposed multipliers, the Vivado default exact multiplier and Xilinx multiplier IP [8] were implemented as exact cases. To show the advantages of the proposed compressor-based multipliers, we evaluated FPGA-based approximate 8×8 multipliers Cc [6], SMA1, SMA2, SMA3 [9], and LM1, LM2 [11] using their open-source codes. The ASIC-based compressor-designed multipliers AK [2], AM [3], and AP [4] were also evaluated. The hardware performance was evaluated in terms of power, area, and delay consumptions. All multipliers were synthesized and optimized within the same environment using default options. The accuracy metrics such as mean relative error distance (MRED), normalized mean error distance (NMED), and error rate (ER) were employed, as defined in [13]. The accuracy performance of the multipliers was assessed through comprehensive simulations conducted in MATLAB.

A. Evaluation of approximate 8×8 multipliers

Table I shows the accuracy results of the approximate 8×8

Authorized licensed use limited to: Sheffield University. Downloaded on November 20,2024 at 09:30:49 UTC from IEEE Xplore. Restrictions apply.

TABLE I. ACCURACY RESULTS OF APPROXIMATE 8×8 MULTIPLIERS

Designs		MRED (%)	NMED (%)	ER (%)
Proposed	CAM1	1.89	0.54	**48.68**
	CAM2	5.86	0.92	84.50
FPGA-based	Cc [6]	12.82	2.43	80.01
	SMA1 [9]	2.76	0.33	56.59
	SMA2 [9]	4.03	0.55	65.91
	SMA3 [9]	6.84	0.90	79.34
	LM1 [11]	5.83	1.15	79.76
	LM2 [11]	6.28	1.24	85.49
ASIC-based	AK [2]	**0.61**	**0.04**	97.17
	AM [3]	2.59	0.11	99.10
	AP [4]	2.42	0.11	90.94

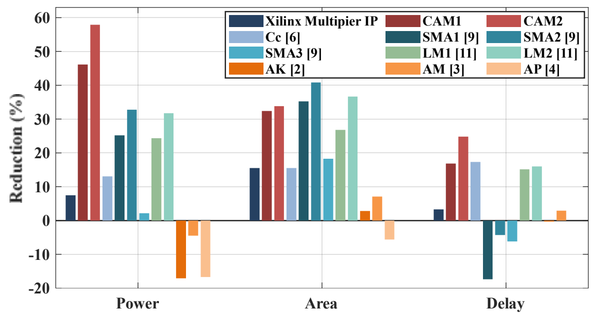

Fig. 6. The hardware performance of approximate 8×8 multipliers.

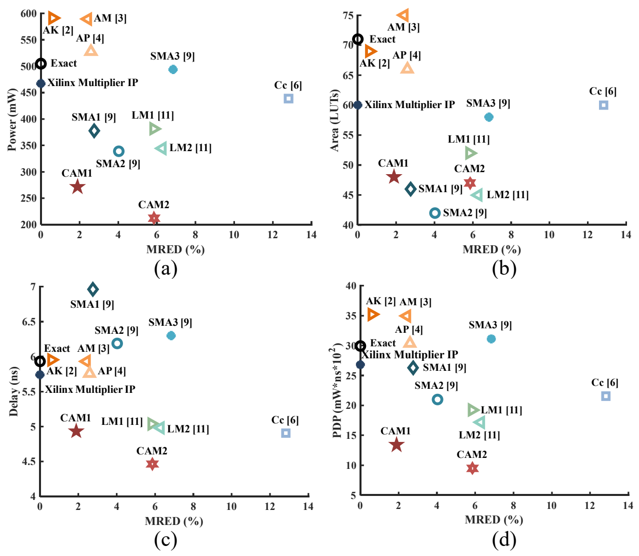

Fig. 7. The result of (a) MRED-power, (b) MRED-area, (c) MRED-delay, and (d) MRED-PDP for 8×8 multipliers.

multipliers. CAM1 has the second lowest MRED among FPGA-based designs and the lowest ER among all multipliers. This is mainly due to the compressor-based once-through structure, where the approximation only occurs in stage 2.

Compared to the Vivado default exact multiplier, the hardware reductions of 8×8 multipliers are represented in Fig. 6. The proposed multipliers exhibit reductions up to 57.90% in power, along with 33.80% in area and 24.78% in delay. Compared with other designs, including both FPGA-based and ASIC-based, the proposed multipliers demonstrate superior power-efficient performance. Here are two reasons that might cause this performance optimization. Firstly, the utilization of the proposed compressors contributes to independent and parallel computations of multiple LUTs, decreasing the circuit correlation between units and improving the utilization of input ports. Secondly, the proposed once-through structure requires fewer carry chains than other multiplication

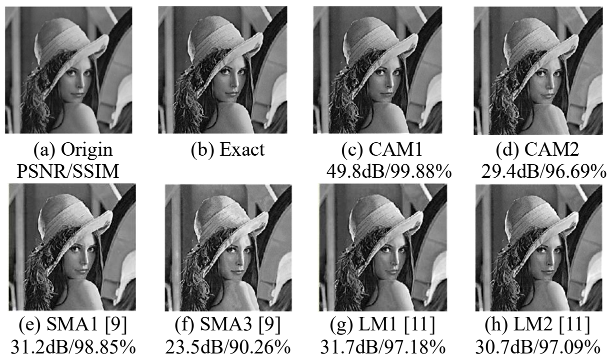

Fig. 8. Processed images by exact and approximate multipliers.

structures. This avoids the extra delay and power consumption caused by the recursive construction.

For an overall comparison of accuracy loss and hardware performance, Fig. 7 illustrates the results of MRED-power, MRED-area, MRED-delay, and MRED-Power delay product (PDP) for 8×8 multipliers. It is noticeable that the proposed multipliers are located closer to the origin, indicating a better accuracy-hardware trade-off. In comparison to the previous multipliers [6][9][11], which have at least 4 carry chains, the proposed approximate multipliers have a maximum of 3 carry chains and require fewer LUTs. This simplification in the circuit structure leads to performance optimization.

B. Evaluation on image sharpening

To validate the practical applicability of approximate multipliers, image sharpening is commonly chosen as the application [13][14]. For assessment metrics, the peak signal-to-noise ratio (PSNR) and the structural similarity index (SSIM) are usually used to measure the difference between the images processed by the exact multiplier and the approximate multiplier. Fig. 8 shows the images and their quality processed by an exact multiplier and approximate multipliers with comparable accuracy. CAM1 achieves the highest processing results with a PSNR of 49.8dB and an SSIM of 99.88%. With a PSNR value of around 30dB, the proposed design CAM2 has more power reduction than LM2 [11] (that is 57.90% vs. 31.75%).

VI. CONCLUSION

In this paper, the power-efficient FPGA-based multipliers are proposed by using approximate computing concept. Firstly, a carry-lagged compressor is proposed to compress the partial products exactly. To compress the remaining elements into 2 rows, an approximate m:2 compressor is designed. Secondly, two variants of approximate 8×8 multipliers with different accuracy are provided by using both types of compressors in a once-through structure. Finally, the experimental results indicate that the proposed multipliers achieve more hardware savings compared with other designs with similar accuracy.

ACKNOWLEDGEMENT

This work was supported in part by the Program of Yunnan Key Laboratory of Intelligent Systems and Computing (No. 202205AG070003); in part by Yunnan Fundamental Research Projects (NO. 202301AU070011); in part by JSPS KAKENHI Grant Number JP21K17770; in part by JSPS KAKENHI Grant Number JP23K16861; in part by JSPS KAKENHI Grant Number JP23H03366.

Authorized licensed use limited to: Sheffield University. Downloaded on November 20,2024 at 09:30:49 UTC from IEEE Xplore. Restrictions apply.

References

[1] W. Liu, T. Cao, P. Yin, Y. Zhu, C. Wang, E. E. Swartzlander, and F. Lo mbardi, "Design and analysis of approximate redundant binary multipl iers", *IEEE Transactions on computers*, vol. 68, no. 6, pp. 804-819, 20 19.

[2] U. A. Kumar, S. K. Chatterjee, and S. E. Ahmed, "Low-Power Compressor-Based Approximate Multipliers with Error Correcting Module," *IEEE Embedded Systems Letters*, vol. 14, no. 2, pp. 59-62, 2022.

[3] M. Zhang, S. Nishizawa and S. Kimura, "Area Efficient Approximate 4-2 Compressor and Probability-Based Error Adjustment for Approximate Multiplier," *IEEE Transactions on Circuits and Systems II: Express Brie*fs, vol. 70, no. 5, pp. 1714-1718, 2023.

[4] H. Pei, X. Yi, H. Zhou, and Y. He, "Design of Ultra-Low Power Consumption Approximate 4-2 Compressors Based on the Compensation Characteristic," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 1, pp. 461-465, 2020.

[5] B. S. Prabakaran, V. Mrazek, Z. Vasicek, L. Sekanina, and M. Shafiqu e, "ApproxFPGAs: Embracing ASIC-Based Approximate Arithmetic Components for FPGA-Based Systems," in *Design Automation Confe rence (DAC)*, pp. 1-6, 2020.

[6] S. Ullah, S. Rehman and et al., "Area-Optimized Low-Latency Approximate Multipliers for FPGA-based Hardware Accelerators," in *Design Automation Conference (DAC)*, pp. 1-6, 2018.

[7] Intel, Integer Arithmetic IP Cores User Guide, https://www.altera.com/en_US/pdfs/literature/ug/ug_lpm_alt_mfug.pdf, 2017.

[8] Xilinx, LogiCORE IP Multiplier v11.2., https://www.xilinx.com/supp ort/documentation/ip_documentation/mult_gen_ds255.pdf, 2017.

[9] S. Ullah, S. S. Murthy, and A. Kumar, "SMApproxLib: Library of FPGA-based Approximate Multipliers," in *Design Automation Conference (DAC)*, pp. 1-6, 2018.

[10] Y. Guo, H. Sun, and S. Kimura, "Small-Area and Low-Power FPGA-Based Multipliers using Approximate Elementary Modules," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 599-604, 2020.

[11] S. Yao and L. Zhang, "Hardware-Efficient FPGA-Based Approximate Multipliers for Error-Tolerant Computing," in *International Conference on Field-Programmable Technology (ICFPT)*, pp. 1-8, 2022.

[12] Xilinx, 7 Series FPGAs Configurable Logic Block User Guide, https:// www.xilinx.com/support/documention/user_guides/ug474_7Series_C LB.pdf, 2016.

[13] H. Jiang, F. J. H. Santiago, H. Mo, L. Liu, and J. Han, "Approximate Arithmetic Circuits: A Survey, Characterization, and Recent Applications," *Proceedings of the IEEE*, vol. 108, no. 12, pp. 2108-2135, 2020.

[14] Y. Guo, H. Sun, P. Lei and S. Kimura, "Design of Low-Cost Approximate Multipliers Based on Probability-Driven Inexact Compressors," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 102, no. 12, pp. 1781-1791, 2019.

Authorized licensed use limited to: Sheffield University. Downloaded on November 20,2024 at 09:30:49 UTC from IEEE Xplore. Restrictions apply.