High-Efficiency FPGA-Based Approximate Multipliers with LUT Sharing and Carry Switching

Yi Guo[1,2], Qilin Zhou[1], Xiu Chen[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*

***Abstract*—Approximate multiplier saves energy and improves hardware performance for error-tolerant computation-intensive applications. This work proposes hardware-efficient FPGA-based approximate multipliers with look-up table (LUT) sharing and carry switching. Sharing two LUTs with the same inputs enables to fully utilize the available LUT resources. To mitigate the accuracy loss incurred from this approach, the truncated carry is partially reserved by switching it to the adjacent calculation. In addition, we create a library of 8×8 approximate multipliers to provide various multiplication choices. The proposed design can provide enhancements of up to 38.75% in power, 17.29% in latency, and 28.17% in area compared to the Xilinx exact multiplier. Our proposed designs are open-source at https://github.com/YnuGuoLab/DATE_FPGA_Approx_Mul and assist in further reproducing and development.**

***Keywords*—approximate computing, multiplier, FPGA**

I. Introduction

Approximate multiplier sacrifices computation precision in exchange for energy savings in error-tolerant applications. However, most state-of-the-art approximate multipliers have been on ASIC-based circuits [1-4]. They might not achieve comparable performance gains when used for FPGA-based accelerators. Therefore, the following studies on the architecture of FPGAs have been carried out to emphasize the requirement for FPGA-based system design.

For FPGA-based approximate multipliers, [5] exploits the FPGA fabrics to define a methodology for designing approximate multipliers. However, the critical path of an accurate multiplier [5] is occupied by three LUTs and three carry chains with strong correlations, undermining the efficiency of approximate computing on the multiplier. [6-9] present the low-complexity approximation methods by deleting LUTs or carry chains and indeed achieve hardware savings. Due to the lack of carry propagation, most of these approximate multipliers are facing one common problem of large accuracy loss. In [7], a library of 8×8 approximate multipliers is constructed by combining four 4×4 approximate multipliers and one adder. Nevertheless, only 35 types of configurations with similar accuracy are provided, which fail to meet the diverse demands of different error-tolerant applications.

To address the aforementioned issues, we propose an FPGA-based multiplier design with novel approximation methodologies. The main contributions are as follows:

1) A LUT-oriented 4×4 accurate multiplier is proposed, which has a structure with fewer LUT layers.

2) Based on the 4×4 accurate multiplier, we propose the approximate multipliers with methodologies of LUT sharing and carry switching.

3) A library of 8×8 approximate multipliers is created by combining 4×4 approximate multipliers and adders, generating 105 different configurations.

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

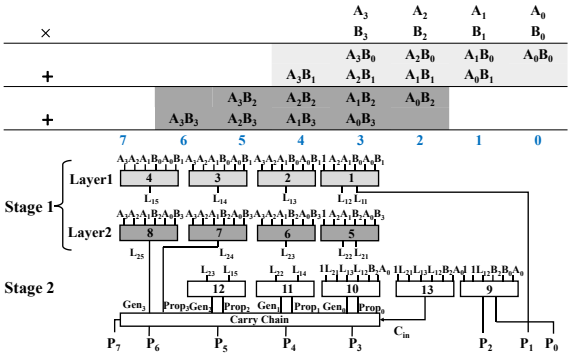
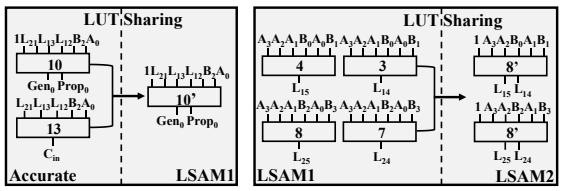


Fig. 1. The structure of the proposed accurate 4×4 multiplier.



(a) (b)

Fig. 2. Structure difference (a) Accurate multiplier and LSAM1. (b) LSAM1 and LSAM2.

II. Proposed Methodology

A. LUT-Oriented Accurate 4×4 Multiplier

Fig. 1 shows the structure of the proposed accurate 4×4 multiplier, which includes two stages. In Stage 1, two layers of LUTs are used to compute the intermediate results L_{1i} and L_{2i} in parallel which is more efficient than the traditional method of two LUTs in serial connection. In Stage 2, L_{1i} and L_{2i} generated in Stage 1 are used to compute the carry-propagate ($Prop_i$) and carry-generate (Gen_i) signals for the carry chain. Then, the final product (P_i) is computed by the carry chain. The critical path of the proposed multiplier is occupied by two LUTs and one carry chain.

B. LUT Shared 4×4 Approximate Multiplier (LSAM) with Carry Chain

1) LUT shared approximate multiplier 1 (LSAM1): As shown in Fig. 2(a), in LSAM1, LUT_{10} and LUT_{13} are combined by deleting LUT_{13} and retaining its logic functions in LUT_{10}. The probability of an error occurring caused by deleting LUT_{13} is only 0.024% (= 6/256).

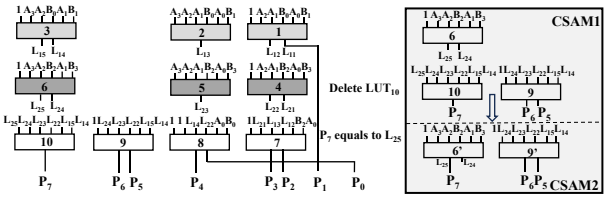


Fig. 3. (a) The structure of CSAM1. (b) The difference between CSAM1 and CSAM2.

2) LUT shared approximate multiplier 2 (LSAM2): As shown in Fig. 2(b), in LSAM2, the results of $L_{14/15}$ and $L_{24/25}$ are computed by $LUT_{4'}$ and $LUT_{8'}$ respectively, instead of four LUTs (LUT_3, LUT_4 and LUT_7, LUT_8) in LSAM1. Compared with LSAM1, the error occurrence of LSAM2 increases from 6 to 13.

C. Carry Switched 4×4 Approximate Multiplier (CSAM) with No Carry Chain

1) Carry switched approximate multiplier 1 (CSAM1): As shown in Fig. 3(a), CSAM1 deletes the carry chain based on LSAM2. The number of elements for calculating P_4 is 10 which exceeds the maximum number of inputs in LUT6. In order to calculate P_4 by one LUT6 while reducing the accuracy loss, we switch the carry from lower-order in P_4 to P_3 which is different from traditional complete truncation.

2) Carry switched approximate multiplier 2 (CSAM2): As shown in Fig. 3(b), based on CSAM1, CSAM2 is proposed by deleting LUT_{10}. The result of P_7 is set to equal to L_{25}. To lower the error incurred by elimination of LUT_{10}, the carry from lower-order in P_7 is switched to P_6. Compared with CSAM1, the error occurrence of CSAM2 increases from 55 to 60.

D. 8×8 Approximate Multipliers Built from 4×4 Multipliers

An 8×8 multiplier can be constructed by combining four 4×4 multipliers and one adder. We present three methods to design an adder. *Accurate adder (ACCA)*: P_{15}~P_4 is produced by the exact addition, which costs nine LUTs and three carry chains [6]. *Moderate approximate adder (MODA)*: *MODA* is proposed by deleting the carry chain on P_7~P_4, costing nine LUTs and two carry chains. *No carry chain adder (NCCA)*: A LUT-based adder is proposed to further reduce the hardware consumption, which costs only eight LUTs.

By combining four types of the proposed 4×4 approximate multipliers in order of decreasing accuracy and three types of adders, a total of 105 different 8×8 approximate multipliers are generated. The name of the proposed design means the types of the adder and 4×4 multipliers. For example, MODA_1234 means that the adder type is MODA while LSAM1, LSAM2, CSAM1, and CSAM2 are used for 4×4 multipliers from high-order to low-order, respectively.

III. Experiment Results and Discussions

To clarify the contributions of this work, the proposed design was coded with Verilog and synthesized for the xc7vx330tffg1157 device with Vivado 2019.1. Regarding the comparison of approximate multipliers, we implemented AMs SMA [5], Ca [6], and LM [7] using the open-source codes provided by [5]-[7], respectively.

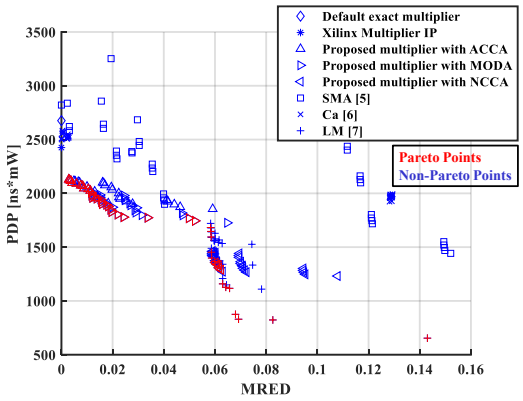


Fig. 4. Pareto optimal analysis of the proposed 8×8 multipliers with state-of-the-art approximate multipliers.

TABLE I. The Comprehensive Comparison of 8×8 Multipliers

Designs	MRED (%)	Power (mW)	Latency (ns)	Area (LUTs)
Default exact	0	504.756	5.304	71
Proposed Acc		424.639	5.219	61
Ca [6]	≈ 0.3	481.594	5.263	58
ACCA_1111		408.208	5.215	57
SMA [5]	≈ 2.0	399.464	5.990	49
MODA_1334		363.987	5.019	50
LM [7]	≈ 6.0	351.865	4.453	46
NCCA_1134		309.179	4.387	51

Fig. 4 shows the Pareto optimal analysis of the proposed 8×8 multipliers with state-of-the-art approximate multipliers. Pareto points mean that the design have better trade-off between accuracy loss and hardware savings. As shown in Fig. 4, most of the Pareto points are offered by our design, which means the proposed design is generally advantageous in most cases. Table I shows the specific performance of the proposed multipliers with other designs under similar accuracy loss. With a similar MRED, our 8×8 designs achieve lower hardware consumption than other multipliers.

Acknowledgement

This work was supported in part by the NSFC under Grant 62362065 and Program of Yunnan Key Laboratory of Intelligent Systems and Computing (No. 202205AG070003).

References

[1] H. Saadat *et al.*, "REALM: Reduced-Error Approximate Log-based Integer Multiplier," in *Proc. DATE*, pp. 1366-1371, 2020.

[2] W. Liu *et al.*, "Design and Analysis of Majority Logic-Based Approximate Adders and Multipliers," *IEEE Trans. Emerg. Topics Comput.*, vol. 9, no. 3, pp. 1609-1624, 2021.

[3] M. S. Ansari *et al.*, "A Hardware-Efficient Logarithmic Multiplier with Improved Accuracy," in *Proc. DATE*, pp. 928-931, 2019.

[4] V. Mrazek *et al.*, "EvoApprox8b: Library of Approximate Adders and Multipliers for Circuit Design and Benchmarking of Approximation Methods," in *Proc. DATE*, pp. 258-261, 2017.

[5] S. Ullah *et al.*, "SMApproxLib: Library of FPGA-based Approximate Multipliers," in *Proc. DAC*, pp. 1-6, 2018.

[6] S. Ullah *et al.*, "Area-Optimized Low-Latency Approximate Multipliers for FPGA-based Hardware Accelerators," in *Proc. DAC*, pp. 1-6, 2018.

[7] S. Yao *et al.*, "Hardware-Efficient FPGA-Based Approximate Multipliers for Error-Tolerant Computing," in *Proc. ICFPT*, pp. 1-8, 2022.

[8] Y. Guo *et al.*, "Small-Area and Low-Power FPGA-Based Multipliers using Approximate Elementary Modules," in *Proc. ASP-DAC*, pp. 599-604, 2020.

[9] Y. Guo *et al.*, "Approximate FPGA-Based Multipliers Using Carry-Inexact Elementary Modules," *IEICE Trans. Fundam. Electron. Commun. Comput.*, 103-A(9), pp. 599-604, 2020.