A Configurable Approximate Multiplier for CNNs using Partial Product Speculation

Xiaolu Hu*, Ao Liu*, Xinkuang Geng*, Zizhong Wei†, Kai Jiang†, Honglan Jiang*
* Department of Micro-Nano Electronics, Shanghai Jiao Tong University, Shanghai, China
† Inspur Academy of Science and Technology, Jinan, Shandong, China
{huxiaolu, 2017la, xinkuang}@sjtu.edu.cn, {wzz, jiangkai}@inspur.com, honglan@sjtu.edu.cn

***Abstract*—To improve the performance and energy efficiency of the compute-intensive convolutional neural networks (CNNs), approximate multipliers have widely been investigated, taking advantage of the inherent error tolerance in CNNs. However, as per their divergencies in the capability of error tolerance, different CNN models and datasets may require various accuracy in multiplication. Thus, in this paper, we propose an energy-efficient approximate multiplier with configurable accuracy to satisfy the continuously evolving requirements of CNNs. In this design, the approximation level is configured by changing the processing scheme for inputs due to their significance to accuracy. The correlations between partial products (PPs) are utilized to eliminate the generation and accumulation of some less significant PPs that are speculated by their adjacent more significant ones. Consequently, four approximate multiplier configurations are devised for 8×8 unsigned multiplication, denoted as AMPPS_S2, AMPPS_S3, AMPPS_S4, and AMPPS_S6. Compared with existing approximate multipliers, the proposed designs show significantly higher accuracy. Compared with an exact carry-save array multiplier, AMPPS_S2 can reduce the power dissipation, delay, and area by 21.7%, 35.1%, and 24.4%, respectively. Moreover, to enhance the efficiency of the proposed approximate multiplier in systolic array-based hardware architectures for CNNs, a novel encoding strategy is proposed for storing the pre-trained weights. Obtaining a similar classification accuracy to the accurate implementation (tested in ResNet18 and ResNet50 on ImageNet), the 32×32 systolic array using AMPPS_S2 shows a 13.5% reduction in power consumption and a 19.2% reduction in area. Overall, the experimental results demonstrate that the proposed approximate multiplier results in higher accuracy in CNN-based image classification, with lower hardware overhead, compared with state-of-the-art approximate multipliers.**

***Index Terms*—approximate multiplier, CNN, partial product speculation, accuracy-configurable**

I. Introduction

As the key to the platforms of big data analytics and artificial intelligence, computing hardware has recently attracted unprecedented attention. A lot of efforts have been made to improve the performance and energy efficiency of computing hardware. However, following Moore's law [1], the size of transistors is approaching its physical limit, leading to many critical problems such as "dark silicon" [2]. Therefore, it is of great challenge to satisfy the ever-increasing demands for computing hardware by using conventional design techniques. To break these technical barriers, accuracy has been considered as a potential trade for lower power consumption and shorter delay, which stimulates the evolution of approximate computing. Approximate computing has widely been utilized in compute-intensive applications where complete accurate computations are not necessary due to their characters of error tolerance or resilience [3], [4]. However, as the capabilities of error resilience in various applications may be different, accuracy-configurable approximate strategies are highly demanded.

Multiplication, as one of the essential arithmetic operations, significantly influences the overall hardware overhead of a computing system. For instance, billions of multiplications can be required for executing a convolutional neural network (CNN) when processing an image [5], [6]. Thus, numerous approximate multipliers have been devised to improve the energy efficiency as well as the performance of accelerators for CNNs, with acceptable loss in accuracy [7], [8]. In these accelerators, the multipliers constituting the majority of the computing cores are dedicatedly implemented by using various approximation techniques. In general, existing approximate multipliers can be classified into three typical types as per their approximation stages and approaches [9]–[17].

As a straightforward methodology, reducing the number of partial products or simplifying the partial product tree of an accurate multiplier can directly save some hardware resources for partial product generation and accumulation [9]–[11]. Among this type of approximate multipliers, the dynamic range unbiased multiplier (DRUM) [9] is a representative design with a good tradeoff between accuracy and hardware consumption. In DRUM, each input is truncated as per its leading-one position, and a '1' is appended to the least significant bit for compensating the truncation error; a reduced-width accurate multiplier and a barrel shifter are then used to achieve the final approximate product. DRUM has been utilized in the design of an approximate tensor processing unit [7]; however, it does not perform well in complex CNNs, like Resnet34. Similarly, TOSAM in [10] reduces the number of partial products by identifying the leading-one position of each input operand, but it still costs a lot to do the summation. In [11], the least significant half of the partial products is regarded as a constant. Although this design achieves very high hardware efficiency, it leads to significantly high errors, especially for small inputs.

Using approximate compressors in the partial product accumulation is another common method of designing approximate multipliers [12]–[14]. In [12], two 2×2 approximate multipliers

This work was supported in part by the National Key Research and Development Science and Technology under grant 2022YFB4500200; and in part by the National Natural Science Foundation of China under grant numbers 62374108 and 62104127.

979-8-3503-4859-0/DATE24/© 2024 EDAA

are designed by using approximate 4:2 compressors, which are utilized to construct approximate recursive multipliers (AxRMs). A six-gate and two gate-free approximate compressors is introduced in [13] to build approximate multipliers. The approximate adder in [14] is able to simplify the partial product accumulation by limiting the carry propagation. These designs may get great efficiency; however, their structures need to be changed a lot if various accuracy is required.

Another type of approximate multipliers simplifies the circuit at the algorithm level. Mitchell's algorithm [18] based on logarithmic approximation has been applied to many approximate logarithmic multipliers (ALMs) [15]–[17]. In [15], [16], different compensation strategies are utilized to alleviate the large single-sided errors due to Mitchell's algorithm. In the power density aware ALM (PAALM), the high power density issue of existing ALM designs is mitigated, but a huge error distance may happen due to overcompensation [17]. In general, ALMs can be very hardware-efficient yet with a low accuracy.

In this work, an accuracy-configurable approximate multiplier that utilizes partial product speculation (PPS) is initially proposed. By analyzing the potential correlations between adjacent partial products, the values of the partial products in a column can be speculated by the features of the adjacent ones. Hence, in this design, we propose to use the speculated values to substitute some less significant partial products to reduce their cost pertaining to generation and accumulation. However, this scheme would lead to a large relative error if partial products of '1's are concentratedly located in the least significant columns. To take this case into consideration, we introduced a pre-processing block that enables an accurate accumulation for the less significant partial products when the more significant partial products are '0's. In addition, by adjusting the pre-processing block, the accuracy of the approximate multiplier can be configured as per the requirements.

The rest of the paper is organized as follows. Section II presents the basic theory and scheme for the proposed approximate multiplier. Section III shows the error characteristics and circuit measurements of the proposed design. Also, state-of-the-art approximate multipliers are compared in this section. Section IV introduces the design of systolic array for CNNs based on the proposed approximate multiplier. In addition, the accuracy of the considered multipliers is further compared by testing them in several CNN applications. Section V concludes this paper.

II. Theory and Design

Fig. 1 shows the partial product (PP) array of 8×8 unsigned multiplication, where each PP is generated by using an AND gate. Each column in the array corresponds to a weight factor, with the more significant bits on the left and the less significant bits on the right. To obtain the final product of the multiplication, a bitwise compression is performed on the PP array.

A. Partial Product Speculation

The right part of Fig. 1 shows the correlations among the PPs in columns 5 and 6. The blue arrow in the figure indicates

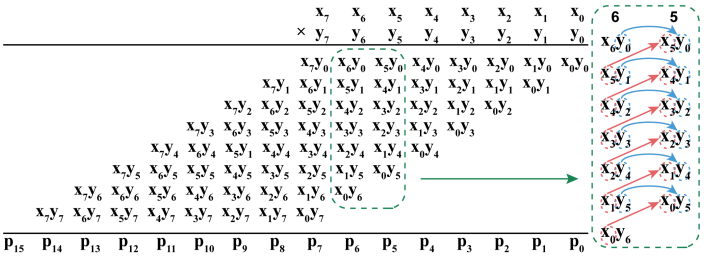

Fig. 1. The partial product array of 8×8 unsigned multiplication and the correlation among adjacent partial products.

that the PP involves the same multiplier bit as the PP on its left side, while the red arrow indicates that the PP involves the same multiplicand bit as the PP on its lower left side. It can be seen that the PP can be directly determined to be '1' when its left and lower-left PPs are both '1.' For instance, when $x_3y_3 =$ '1' and $x_2y_4 =$ '1', as per the operation of the AND gate, it is easy to obtain $x_2y_3 = x_2 \& y_3 =$ '1.'

Based on the above observation, we propose to use the PPs of a column to speculate the sum of the PPs in their right adjacent column, enabling an approximation without PP generation and accumulation. A signal, *spec_flag*, is generated to speculate the sum result; the *spec_flag* is given by

$$spec\_flag = (pp_0 \& pp_1) | \cdots | (pp_{n-2} \& pp_{n-1}), \qquad (1)$$

where n represents the number of PPs in the given column, and pp_i is the PP in the ith row of the given column. When *spec_flag* = '1,' there must exist at least one PP of '1' in the speculated column. On the other side, *spec_flag* = '0' indicates that there is a high possibility that the sum of the speculated column is equal to '0.'

B. The Proposed PPS-Based Multiplication Scheme

Fig. 2 illustrates the proposed 8×8 unsigned multiplication based on PPS. The PP tree is divided into three parts, accurate part, approximate part, and truncation part. The PPs in the least significant five columns and the first row are truncated and approximated by using two constants '1's in the 4th and 5th columns when the accurate result is not zero. Otherwise, these two constants are forced to be '0'. In the approximate part, an approximate compressor is proposed to speculate the sum of column 5 and provide an approximate compression for column 6, as shown in Fig. 3. The speculation value for the sum of column 5 and the compression strategy for column 6 are obtained by an exhaustive simulation under the condition that the final product achieves the minimum mean error distance. When the *spec_flag* is '1,' the speculation value for the sum of column 5 is 2, which is implemented by adding a carry bit to column 6. When the *spec_flag* is '0,' the speculation value is '0.' The approximate compression for column 6 is performed by

$$\begin{cases} comp\_s1 = pp_0 \\ comp\_s2 = pp_3 \\ comp\_c1 = pp_1 \& pp_2 \\ comp\_c2 = pp_5 | pp_4 \end{cases}. \qquad (2)$$

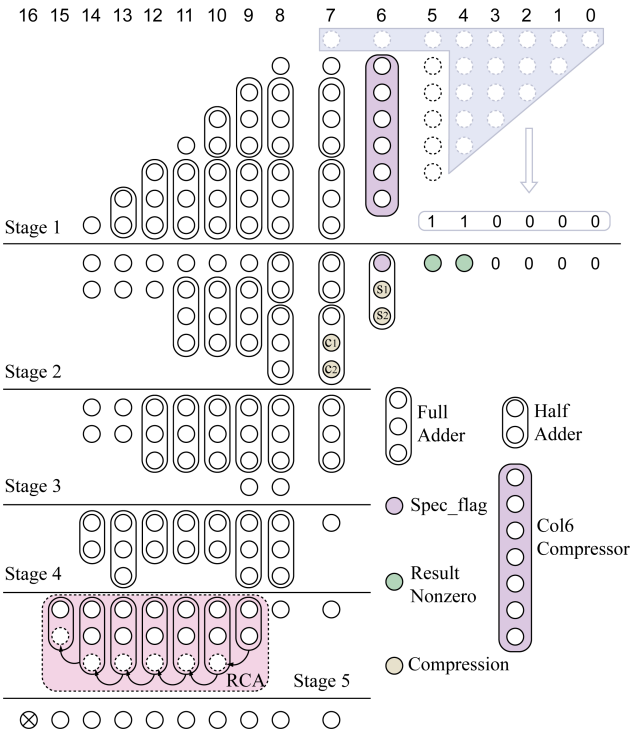

Fig. 2. The proposed 8×8 unsigned PPS-based multiplication scheme.

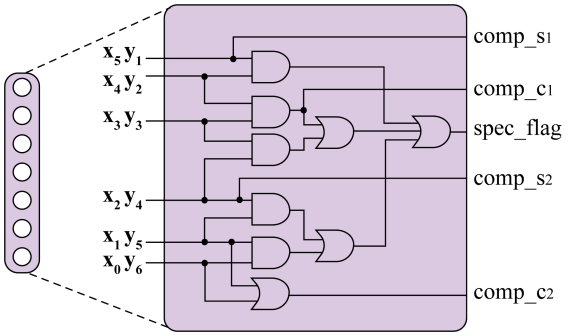

Fig. 3. The compressor module of column 6.

The rest of the PPs are exactly compressed due to their high significance. Ultimately, the product is obtained by using a Wallace tree and a ripple carry adder (RCA).

C. Pre-Processing Block and Encoding Strategy of Weights

In the PPS-based multiplication scheme, almost half of the partial products are truncated, leading to large relative errors for small inputs. To improve the accuracy for small inputs, a pre-processing block is designed for the approximate multiplier in a configuration-friendly manner. Specifically, a leading-one detector (LOD) and a shifter are incorporated to move the non-zero bits of small inputs to the left more accurate part of the PPS-based multiplication. As the approximate multiplier is intended for CNNs, its inputs are generally the input and weight of a neuron. The weights are usually pre-trained and stored in a memory. Thus, a novel encoding strategy is proposed for the weights to simplify the pre-processing block. Finally, the weights stored in the memory are uniquely encoded following the training process.

The configuration of accuracy is achieved by setting the maximum numbers of shift bits for the input and weight pre-processing, denoted as BS_{imax} and BS_{wmax}, respectively. As BS_{wmax} increases, the hardware cost increases, but the error decreases. To begin, Fig. 4(a) illustrates the encoding strategy when BS_{wmax} is 1, where an 8-bit *w* is encoded as an 8-bit *w'*. The most significant bit (MSB) of *w'* denoted as *shift_w* preserves the information regarding the shift width. *shift_w* = '0' indicates that the leading-one of *w* locates at the MSB position. In this case, an OR gate is used on the least significant two bits of *w*, resulting in a 7-bit *mult_w* for *w'*. Otherwise, *mult_w* is directly obtained from *w* when *shift_w* is '1.' In Fig. 4(b) and (c), BS_{wmax}s are 2 and 3, respectively, requiring two bits to store *shift_w*. This means that only six bits remain to store the weight bits. In Fig. 4(c), for the sake of storing more valid bits, the '1' is hidden when the leading-one of *w* is in the most significant three bits, the six bits following the leading-one are stored as *mult_w*. Otherwise, the shift width is 3, encoded as *shift_w* ='11'. In this case, the most significant bit of the reserved bits (corresponding to the hidden '1' for the other three cases) is positioned at the end for convenient extraction. Fig. 4(b) illustrates the strategy for the scenario that BS_{wmax} is 2, which is a simplification of Fig. 4(c). Note that, by using this encoding strategy, the inputs can be encoded offline and stored in the memory, without additional hardware cost in the multiplication. For example, when *w* is 8'b00101111, it is encoded as *w'* = 8'b10101111, 8'b10011111, and 8'b10011110 for BS_{wmax} = 1, 2, and 3, respectively. To decode *w'*, when *shift_w*<BS_{wmax}, a hidden 1 is concatenated to the left of *mult_w*; otherwise, the least significant bit of *mult_w* is concatenated to the left of the other *mult_w* bits. The obtained bits are then left shifted by *shift_w* bits to get the decoded weight for the PPS-based multiplicaiton block.

D. The Proposed Approximate Multiplier

Fig. 5 shows the architecture of the proposed approximate multiplier, consisting of a pre-processing block for inputs, a weight pre-processor, a PPS-based multiplication block, and a right shifter. The pre-processing block acquires the leading-one position (the number of bits for left shift) and the 8-bit input for the multiplication block. The weight pre-processor is utilized to convert original weight data into the encoded weight *w'* following the proposed strategy. For the constant weights in CNNs, the weight pre-processor is avoided by encoding the weights offline, and *w'* is stored in the memory. The final product is obtained by using a right shifter, where the number of bits for right shift is the sum of the numbers of bits for left shift for the input and weight when encoding.

Out of the many possible configurations, three configurations of the proposed approximate multiplier using PPS (AMPPS) are considered, denoted as AMPPS_S2, AMPPS_S4, and AMPPS_S6, whose BS_{imax} and BS_{wmax} are 1, 2, and 3, respectively. The maximum shift bits for the right shifter of these multipliers are 2, 4, and 6, respectively. In addition,

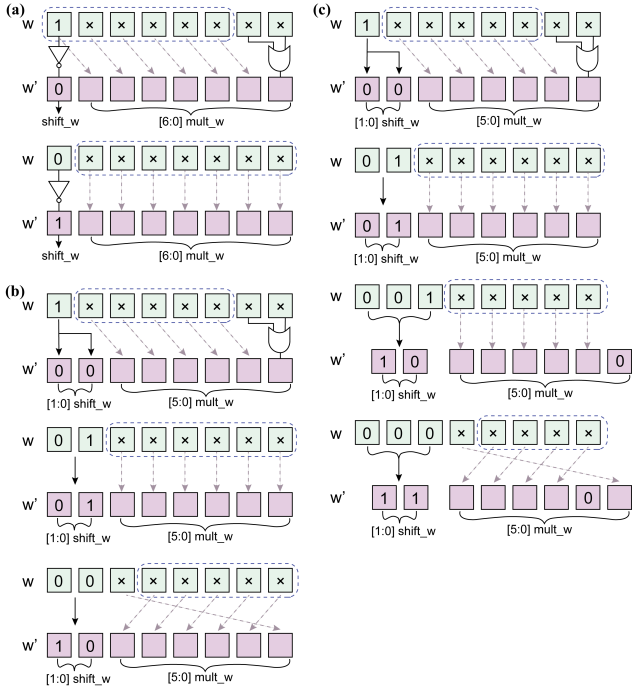

Fig. 4. The diagram of the encoding strategy of weights shift at most (a) one bit, (b) two bits, and (c) three bits.

AMPPS_S3 with 2 as BS_{wmax} and 1 as BS_{imax} (the maximum shift bits for the right shifter is 3) is evaluated.

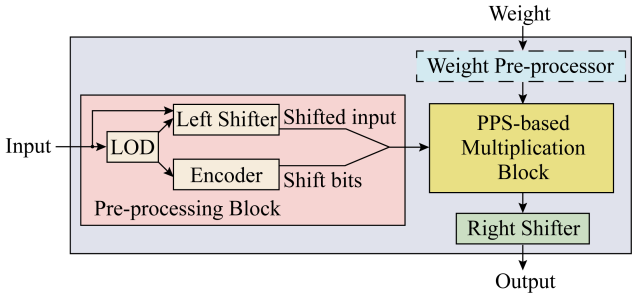

Fig. 5. The proposed approximate multiplier using PPS.

III. EVALUATION AND COMPARISON

In this section, the proposed approximate multiplier and other state-of-the-art designs with different approximation techniques are evaluated in both circuit and error characteristics.

A. Error Analysis

The normalized mean error distance (NMED), mean relative error distance (MRED), error rate (ER), and maximum error distance (MaxED) are computed to evaluate the errors of approximate multipliers. The simulation results for the considered approximate multipliers are tabulated in Table I.

As the maximum shift bits for the right shifter increase, the error metrics of the proposed multiplier significantly decrease, especially for the MRED. However, the maximum error distance remains the same, and it happens when the two input operands are 239 and 255, respectively. This is because the accuracy configuration does not work if the leading-ones of both the input operands occur in the MSB position. According to Table I, AxRM1 [12] performs well in NMED and MaxED, but its MRED is relatively high, which means that the approximate results for small inputs show large deviations from the accurate ones. Compared with other state-of-the-art approximate multipliers, the proposed design exhibits remarkably high accuracy in NMED, MRED, and MaxED. The ERs of the proposed multiplier configurations are also lower than the other designs except for DRUM7 [9]. The MaxED of PAALM [17] is the largest, which occurs when the inputs are 129 and 255 with the approximate result of 65024 due to overcompensation.

TABLE I
ERROR RESULTS OF 8×8 APPROXIMATE MULTIPLIERS

Multiplier	***NMED*** $(\times 10^{-4})$	***MRED*** $(\times 10^{-2})$	***ER*** $(\%)$	***MaxED***
AxRM1	2.58	0.76	75.00	50
DRUM7	16.15	0.48	60.64	1012
PAALM	357.41	12.16	93.09	32129
Mul_2 [13]	178.02	14.77	98.86	9811
Mul_ECM	60.97	9.74	99.97	2277
TOSAM(6,4)	14.77	0.62	94.03	1009
TOSAM(6,2)	51.74	2.10	98.16	3841
TOSAM(5,2)	55.13	2.16	97.77	4353
TOSAM(5,1)	102.28	4.06	98.85	7681
AMPPS_S2	7.63	0.80	96.27	417
AMPPS_S3	7.30	0.58	95.88	417
AMPPS_S4	7.03	0.39	95.68	417
AMPPS_S6	6.92	0.28	94.41	417

B. Circuit Analysis

To measure the circuit characteristics, in this work, all multipliers are synthesized by Synopsys Design Compiler (DC) using 28nm HLMC technology in the typical condition with 25°C and 0.9V supply voltage. Taking into account the factor of switching activity, the waveform file of a million random input combinations with 1GHz frequency is created by Verilog Compile Simulator (VCS) and used as the stimuli of the multipliers to estimate the power dissipation.

TABLE II
HARDWARE COMPARISON OF 8×8 APPROXIMATE MULTIPLIERS

Multiplier	***Delay*** (ns)	***Power*** (mW)	***PDP*** (fJ)	***Area*** (μm^2)	***ADP*** $(ns \cdot \mu m^2)$
Exact_CSA	0.862	0.1990	171.6	254.9	219.8
AxRM1	0.723	0.1724	124.7	234.3	169.5
DRUM7	0.842	0.1974	166.2	227.9	191.9
PAALM	1.216	0.1865	226.8	227.0	276.1
Mul_2 [13]	0.414	0.0484	20.0	69.4	28.7
Mul_ECM	0.535	0.0949	50.7	123.4	66.0
TOSAM(6,4)	1.128	0.2490	280.9	277.2	312.8
TOSAM(6,2)	0.783	0.1633	127.8	185.3	145.0
TOSAM(5,2)	0.713	0.1573	112.1	180.2	128.4
TOSAM(5,1)	0.671	0.1353	90.7	159.9	107.2
AMPPS_S2	0.560	0.1558	87.2	192.8	108.0
AMPPS_S3	0.626	0.1628	102.0	204.4	128.0
AMPPS_S4	0.718	0.1733	124.5	217.5	156.2
AMPPS_S6	0.718	0.1879	134.9	232.8	167.0

Table II shows the results for critical path delay, power consumption, power-delay product (PDP), area and area-delay product (ADP) for the considered approximate multipliers. Except for Mul_2 [13] and Mul_ECM [11] with very low accuracy, the proposed AMPPS_S2 presents the lowest PDP; it reduces the power dissipation, delay, PDP, area, and ADP by 21.7%, 35.1%, 49.2%, 24.4%, and 50.9%, respectively, compared to the exact carry-save array multiplier (Exact_CSA). Among the four considered configurations, AMPPS_S3 and AMPPS_S4 exhibit a medium level of performance. AMPPS_S3 outperforms TOSAM(5,1) and TOSAM(5,2) [10] in terms of delay; however, its power dissipation and area are higher. Despite consuming slightly more power than AxRM1, AMPPS-S4 surpasses it in other aspects of hardware measurements. For AMPPS_S6, although it utilizes more hardware resources for higher accuracy, it still boasts superior delay and power compared to DRUM7, PAALM, and TOSAM(6,4). Furthermore, it achieves reductions of 5.6%, 16.8%, 21.4%, 8.7%, and 24.0% in power dissipation, delay, PDP, area, and ADP, respectively.

IV. SYSTOLIC ARRAY AND CNN APPLICATIONS

A. The Design of Systolic Array

Systolic array (SA) comprises a considerable number of parallel processing elements (PEs), serving as the compute core of many CNN accelerators. This work constructs a weight-stationary systolic array, the same as the one in APTPU [7] that aims at simplifying the circuit structure for improved performance and energy efficiency. Fig. 6 illustrates the architecture of the 4×4 approximate systolic array, consisting of the weight buffer, pre-processing units, and approximate processing elements (APEs). The weight buffer stores the encoded weights as per the proposed encoding strategy, and the pre-processing unit serves the same function as the pre-processing block in Fig. 5. The multiplication in the APE is implemented by the proposed PPS-based multiplication block and right shifter.

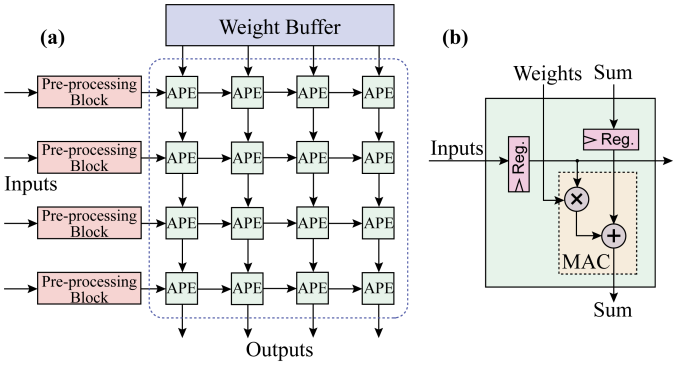

Fig. 6. (a) The approximate design of the systolic array, and (b) the internal architecture of a processing element.

Table III reports the power consumption and area of the approximate SAs, including the pre-processing blocks and APEs for the proposed approximate multiplication scheme. The clock frequency for the circuit syntheses is 500MHz. The parameter *n* denotes the size of a SA, i.e., *n*×*n* PEs in total. Table III shows that the SA using AMPPS_S2 exhibits very high power efficiency, i.e., its power consumption is reduced by 15.8%, 14.2%, and 13.5% for the sizes of 8×8, 16×16, and 32×32, respectively, compared with the exact SA. Additionally, this SA experiences an area reduction of 18.7%, 18.7%, and 19.2% in the 8×8, 16×16, and 32×32 sizes, respectively. TOSAM(5,2) can achieve similar power and area reductions to AMPPS_S2; however, its accuracy is much lower (see Table I). The SA equipped with AMPPS_S3 consumes slightly more power compared to that using AMPPS_S2; however, it is more efficient than those using AxRM1 and DRUM7. The SAs utilizing AMPPS_S4 and AMPPS_S6 result in relatively small improvements in power consumption and area.

TABLE III
POWER AND AREA COMPARISON OF SYSTOLIC ARRAYS

Multiplier	**Power**(mW)			**Area**(μm^2)		
	n=8	***n=16***	***n=32***	***n=8***	***n=16***	***n=32***
Exact	14.62	61.85	261.44	28396	122266	523305
AxRM1	13.79	58.92	248.38	26670	114076	482643
DRUM7	13.85	59.24	254.73	25417	109895	466934
TOSAM(6,4)	15.44	65.70	284.67	28818	123689	532282
TOSAM(5,2)	11.96	51.96	225.24	22497	98145	421134
AMPPS_S2	12.32	53.10	226.27	23091	99421	423001
AMPPS_S3	13.27	56.10	236.97	24526	105522	445451
AMPPS_S4	14.00	59.12	250.08	26011	110806	471418
AMPPS_S6	14.08	59.98	253.30	26376	113179	479415

B. CNN Applications

To further assess the accuracy of the considered approximate multipliers, they are utilized in VGG and Resnet on two typical image classification datasets, Cifar-10 and ImageNet. In these applications, the multiplications in the convolutional layers of the pre-trained CNN models are implemented by using the approximate multipliers. Note that the inputs and weights of the CNN models are quantized into 8-bit unsigned format; thus, 8x8 unsigned multipliers are sufficient. The other operations such as addition in these simulations remain accurate.

The classification accuracy for Cifar-10 is shown in Fig.7, where the simulations are performed by using TFApprox [19]. For all the three models, the four configurations of the proposed approximate multiplier generally result in higher classification accuracy than the other designs. PAALM, Mul_2 [13] and Mul_ECM produce serious errors that lead to significantly low classification accuracy. It is noteworthy that AMPPS_S6 achieves very close classification accuracy to the exact multiplier even for the very complex Resnet34. AMPPS_S4 and AMPPS_S3 show the second and third highest accuracy of the approximate multipliers in Resnet models. Although AMPPS_S2 exhibits a slightly lower accuracy than AxRM1 in Resnet18, it outperforms AxRM1 in the other two models. In summary, a great advantage of the proposed multipliers is shown in the CNN applications.

For ImageNet, to ensure a high accuracy, the pre-trained CNN models are quantized into 9-bit signed format; thus, besides an 8x8 unsigned multiplication, additional sign bits

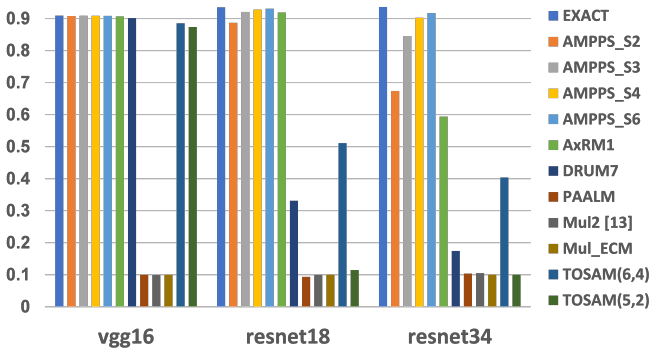

Fig. 7. The classification accuracy for Cifar-10 by using different approximate multipliers in VGG-16, Resnet-18, and Resnet-34, respectively.

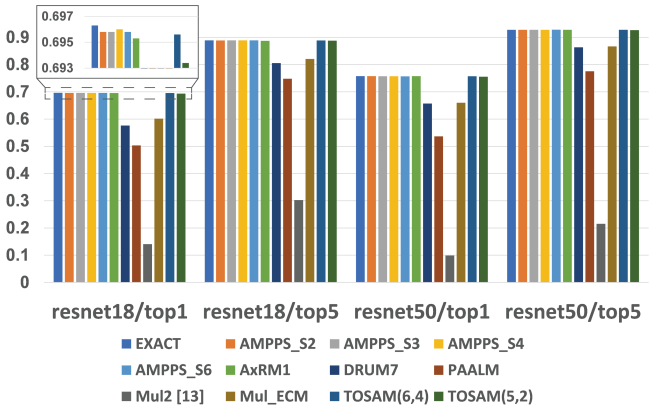

Fig. 8. The classification accuracy of ImageNet by using different approximate multipliers in Resnet-18 and Resnet-50, respectively.

processing is required. Fig. 8 shows the classification accuracy of ImageNet using the considered approximate multipliers for the unsigned multiplications. In this application, the proposed multiplier, AxRM1, and TOSAMs exhibit excellent accuracy that is on par with the exact multiplier. Among these multipliers, AMPPS_S2 shows the lowest hardware overhead.

V. CONCLUSION

This paper proposed an accuracy-configurable approximate 8×8 unsigned multiplier by using partial product speculation. Considering the correlations between adjacent PPs, the less significant PPs are speculated by the more significant PPs, eliminating the PP generation and accumulation. A pre-processing block is further utilized to ensure a high accuracy for small inputs. Moreover, to cooperate with the proposed multiplication scheme, a novel encoding strategy is devised for the stored weights of CNNs. Compared with an exact carry-save array multiplier, AMPPS_S2 reduces the PDP by 49.2%; it shows reductions of 13.5% and 19.2% in power dissipation and area for implementing a 32×32 approximate SA, respectively. Finally, the proposed multiplier is utilized in several CNN applications for image classification on Cifar-10 and ImageNet. The simulation results show that the proposed multiplier results in very close classification accuracy to the exact implementation. Compared with state-of-the-art approximate multipliers, the proposed design shows significantly higher accuracy considering similar hardware overhead.

REFERENCES

[1] M. Lundstrom, "Moore's law forever?," *Science*, vol. 299, no. 5604, pp. 210–211, 2003.

[2] 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.

[3] 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.

[4] G. Rodrigues, F. Lima Kastensmidt, and A. Bosio, "Survey on approximate computing and its intrinsic fault tolerance," *Electronics*, vol. 9, no. 4, p. 557, 2020.

[5] M. Cho and Y. Kim, "FPGA-based convolutional neural network accelerator with resource-optimized approximate multiply-accumulate unit," *Electronics*, vol. 10, no. 22, p. 2859, 2021.

[6] M. Huang, X. Zhang, X. Lan, H. Wang, and Y. Tang, "Convolution by multiplication: Accelerated two-stream fourier domain convolutional neural network for facial expression recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 3, pp. 1431–1442, 2021.

[7] M. E. Elbtity, P. S. Chandarana, B. Reidy, J. K. Eshraghian, and R. Zand, "APTPU: Approximate computing based tensor processing unit," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 12, pp. 5135–5146, 2022.

[8] M. A. Hanif, F. Khalid, and M. Shafique, "CANN: Curable approximations for high-performance deep neural network accelerators," in *Proceedings of the 56th Annual Design Automation Conference*, pp. 1–6, 2019.

[9] S. Hashemi, R. I. Bahar, and S. Reda, "DRUM: A dynamic range unbiased multiplier for approximate applications," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 418–425, 2015.

[10] S. Vahdat, M. Kamal, A. Afzali-Kusha, and M. Pedram, "TOSAM: An energy-efficient truncation- and rounding-based scalable approximate multiplier," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 5, pp. 1161–1173, 2019.

[11] F. Sabetzadeh, M. H. Moaiyeri, and M. Ahmadinejad, "An ultra-efficient approximate multiplier with error compensation for error-resilient applications," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 70, no. 2, pp. 776–780, 2023.

[12] H. Waris, C. Wang, C. Xu, and W. Liu, "AxRMs: Approximate recursive multipliers using high-performance building blocks," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 2, pp. 1229–1235, 2022.

[13] L. Sayadi, S. Timarchi, and A. Sheikh-Akbari, "Two efficient approximate unsigned multipliers by developing new configuration for approximate 4:2 compressors," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 70, no. 4, pp. 1649–1659, 2023.

[14] H. Jiang, C. Liu, F. Lombardi, and J. Han, "Low-power approximate unsigned multipliers with configurable error recovery," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 1, pp. 189–202, 2019.

[15] P. Yin, C. Wang, H. Waris, W. Liu, Y. Han, and F. Lombardi, "Design and analysis of energy-efficient dynamic range approximate logarithmic multipliers for machine learning," *IEEE Transactions on Sustainable Computing*, vol. 6, no. 4, pp. 612–625, 2020.

[16] W. Liu, J. Xu, D. Wang, C. Wang, P. Montuschi, and F. Lombardi, "Design and evaluation of approximate logarithmic multipliers for low power error-tolerant applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 9, pp. 2856–2868, 2018.

[17] S. Yu and S. X.-D. Tan, "PAALM: Power density aware approximate logarithmic multiplier design," in *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, pp. 128–133, 2023.

[18] J. N. Mitchell, "Computer multiplication and division using binary logarithms," *IRE Transactions on Electronic Computers*, vol. EC-11, no. 4, pp. 512–517, 1962.

[19] F. Vaverka, V. Mrazek, Z. Vasicek, and L. Sekanina, "Tfapprox: Towards a fast emulation of dnn approximate hardware accelerators on gpu," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 294–297, 2020.