IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 30, 2022, accepted May 16, 2022, date of publication May 30, 2022, date of current version June 13, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3179112

A Low-Power and High-Accuracy Approximate Multiplier With Reconfigurable Truncation

FANG-YI GU, (Graduate Student Member, IEEE), ING-CHAO LIN, (Senior Member, IEEE), AND JIA-WEI LIN, (Graduate Student Member, IEEE)

Department of Computer Science and Information, National Cheng Kung University, Tainan 701, Taiwan

Corresponding author: Ing-Chao Lin (iclin@mail.ncku.edu.tw)

This work was supported in part by the Ministry of Science and Technology under Grant 110-2221-E-006-084-MY3 and Grant 109-2628-E-006-012-MY3; and in part by the Intelligent Manufacturing Research Center from the Featured Areas Research Center Program by the Ministry of Education, Taiwan.

ABSTRACT Multipliers are among the most critical arithmetic functional units in many applications, and those applications commonly require many multiplications which result in significant power consumption. For applications that have error tolerance, employing an approximate multiplier is an emerging method to reduce critical path delay and power consumption. An approximate multiplier can trade off accuracy for lower energy and higher performance. In this paper, we not only propose an approximate 4-2 compressor with high accuracy, but also an adjustable approximate multiplier that can dynamically truncate partial products to achieve variable accuracy requirements. In addition, we also propose a simple error compensation circuit to reduce error distance. The proposed approximate multiplier can adjust the accuracy and power required for multiplications at run-time based on the users' requirement. Experimental results show that the delay and the average power consumption of the proposed adjustable approximate multiplier can be reduced by 27% and 40.33% (up to 72%) when compared to the Wallace tree multiplier. Moreover, we demonstrate the suitability and reconfigurability of our proposed multiplier in convolutional neural networks (CNNs) to meet different requirements at each layer.

INDEX TERMS Approximate computing, approximate multiplier, CNN accelerator, deep learning, high precision, reconfigurable approximate design.

I. INTRODUCTION

Multipliers are among the most critical arithmetic functional units in many applications, such as digital signal processing (DSP), computer vision, multimedia processing, image recognition, and artificial intelligence. Those applications commonly need numerous multiplications that result in huge power consumption. The high-power consumption is a challenge for implementing those applications, especially on mobile devices. Therefore, many studies have proposed techniques to reduce the power consumption of multiplier circuits. One solution to reduce the power consumption of a multiplier is to approximate multiplication if the targeted applications allow error tolerance, or in other words, if they are related to human senses. Due to the human sensory limitations, such as limited viewing spectrum and hearing range, the accurate computing results are not necessary. The approximate multipliers sacrifice accuracy in exchange for the reduction of cell area, timing delay, and power consumption.

Approximate multipliers can be categorized into two types. The first type is to control the timing path of the multiplier, which can be achieved by using the dynamic voltage scaling. If a lower voltage is applied to a multiplier, the delay of the critical path will increase. Therefore, when the violation of the timing path happens, the errors occur, generating approximate results. The second type is to modify the functional behaviors of multipliers, which is to redesign the accurate multiplier circuits e.g., Wallace Tree Multiplier and Dadda Tree multiplier. Among the redesigning multipliers, most of the previous works proposed inaccurate m-n compressors that have m inputs and generate n outputs. These inaccurate compressors were used to compress the partial products within multiplication since the procedure of compressing partial products consumed most of the multiplier energy and caused long path delay.

The associate editor coordinating the review of this manuscript and approving it for publication was Mario Donato Marino.

Most of these previous approximate multipliers only provided fixed output accuracy and fixed required power. However, the ability to dynamically adjust accuracy and power consumption is useful for some applications, such as artificial intelligence whose requirement is changing over time. Note that in order to achieve an adjustable multiplier design, additional hardware cost is unavoidable.

In this work, we propose a high accuracy 4-2 compressor, based on which, we further design a high accuracy approximate multiplier. In addition, we propose a dynamic input truncation technique to adjust the accuracy and required power. The contributions of the paper are summarized as follows:

- We propose a high accuracy approximate 4-2 compressor that can be used to construct the proposed approximate multiplier.
- We design a simple error compensation circuit to further reduce the error distance.
- We propose a dynamic input truncation technique that can be used to adjust accuracy and power required for a multiplication. The proposed technique is suitable for CNNs as power consumption can be easily adjusted depending on the different requirements for each layer.
- Based on the proposed 4-2 compressor, error compensation circuit and the dynamic input truncation technique, we propose a high-accuracy and reconfigurable approximate multiplier.

Experimental results show that the proposed adjustable approximate multiplier with dynamic input truncation can reduce the delay by 26.85% and can reduce the average power consumption by 40.33% (up to 72%) compared to Wallace Tree Multiplier. Moreover, the error rate of the proposed multiplier is 11.57% and it has the smallest mean error distance compared to other approximate multipliers. The proposed adjustable approximate multiplier with dynamic truncation also has less area overhead compared to other programmable multipliers.

The rest of this paper is organized as follows: Section II introduces previous approximate multipliers and the metric for comparing different approximate multipliers. Section III introduces our proposed approximate multiplier. Section IV compares accuracy, delay, area, and power. Section V compares the results when the proposed multiplier and the previous works are applied to CNNs. Section VI concludes the paper.

II. PRELIMINARIES

A. APPROXIMATE MULTIPLIER DESIGNS

Approximate multipliers can be constructed with different approaches, including scaling the supply voltage, truncating some partial product rows, designing the simplification multiplier equations, and using the approximate compressors to reduce the number of partial products rows.

In [1], [2], voltage scaling technique was used to control the supply voltage of the logic gates, which helped reduce the power consumption. If the supply voltage was lower than the required nominal voltage, timing violation would occur, resulting in approximate results. However, if timing violation happened in critical paths, the erroneous value might be very large.

In [3], [4], approximate multipliers were constructed by truncating the partial product columns which were close to the least significant bit (LSB) column to reduce the carry propagation length. The closer the partial product was to the LSB column, the smaller the weight of the partial product was. Since the weights of truncated partial products were small, it would not cause a large error distance.

In [5], Zendegani *et al.* proposed an approximate multiplier by modifying the accurate multiplication equation, as shown in (1), where the rounded numbers of the input A and B were denoted by A_r and B_r. To reduce hardware cost, they used the nearest values of 2^n to approximate A_r and B_r, and therefore, $A_r \times B, B_r \times A$, and $A_r \times B_r$ could be computed by the shift operation. $(A_r - A) \times (B_r - B)$ was small and could be ignored. The modified multiplication equation is shown in (2), where $(A_r - A) \times (B_r - B)$ was abandoned.

$$A \times B = (A_r - A) \times (B_r - B) + A_r \times B + B_r \times A - A_r \times B_r \quad (1)$$

$$A \times B \cong A_r \times B + B_r \times A - A_r \times B_r \quad (2)$$

Wallace [6] proposed a fast multiplier architecture called Wallace tree multiplier. In a Wallace tree multiplier, accurate 2-2 compressors (half-adder) and 3-2 compressors (full adder) were used to reduce the number of rows of partial products. Accurate 4-2 compressors can also be used in a Wallace Tree Multiplier to produce a more regular layout [7]. The reduced partial products are summed by a carry propagating adder to obtain the final product. Because of its popularity, most previous work designed approximate compressors modified from the accurate 4-2 compressor. The details of the accurate and approximate compressors are described in the next subsection.

B. ACCURATE AND APPROXIMATE 4-2 COMPRESSOR

An accurate 4-2 compressor can be implemented with two full adders as shown in FIGURE 1. In a multiplication, $X_{-1} \sim X_4$ are the partial products in the same column and C_{i-1} is the carry-in from the previous column compressor. The accurate 4-2 compressor will produce three outputs: C_i, carry, and sum. FIGURE 2 shows the block diagram of most of the approximate 4-2 compressors. Compared to the accurate 4-2 compressor in FIGURE 1, the approximate 4-2 compressor does not use Ci-1 and does not generate C_i. Thus, the number of inputs becomes four, and the number of outputs becomes two. By doing this, the complexity of compressing the partial product can be reduced greatly. However, errors are unavoidable when the four inputs are all 1, which leads to '100' as the binary output result that requires at least three output ports.

Momeni *et al.* [8] proposed two approximate 4-2 compressors. The first approximate 4-2 compressor Momeni_acc had five inputs and three outputs like an accurate 4-2 compressor

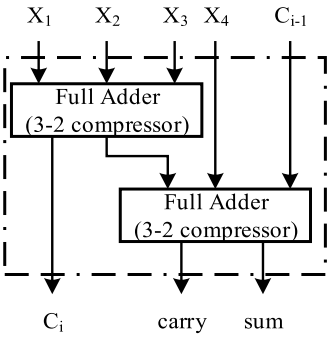

FIGURE 1. Accurate 4-2 compressor.

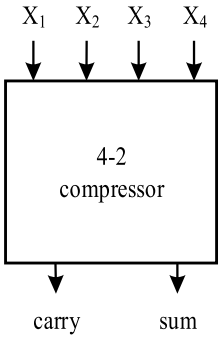

FIGURE 2. Approximate 4-2 compressor.

while the second approximate 4-2 compressor Momeni_fast had four inputs and two outputs like other approximate 4-2 compressors. Yang *et al.* [9] modified the accurate 4-2 compressor and proposed three approximate 4-2 compressors (ACCI1, ACCI2, and ACCI3). The difference between ACCI1, ACCI2, and ACCI3 were the equations to generate the sum bit. Ha *et al.* [3] augmented the approximate 4-2 compressor of Yang ACCI3 and designed a simple error recovery circuit for the modified compressor. ACCI3 generated errors when both X3 and X4 were 1. However, Ha [3] modified the Boolean function of the generated carry so the error value was always -1, which allowed the error recovery circuit to be simplified. Lin et al [10] replaced XOR gate with MUX to reduce the delay of the proposed approximate 4-2 compressor. Similar to Yang ACCI1, the error happened when the X1, X2, X3, and X4 were 1. However, in Lin's compressor, the value of error distance was two, not one. Edavoor *et al.* [11] proposed a dual-stage 4-2 compressor, whose error distance was especially designed to either be positive or negative. They split the partial product reduction into multi-stage and cascaded the dual-stage 4-2 compressor in different stages, where the error wase in the process. Sabetzadeh et al [12] presented a majority-based 4-2 compressor. The carry bit was produced by a 3-input major gate, and the sum bit was always 1. Strollo [13] used a stacking circuit technique, which counted the number of 1's in the inputs to design the approximate 4-2 compressor. The stacking circuit converts four inputs to three inputs and a full adder was used to produce the sum and carry bits. Xiao *et al.* [14] took the ununiform data distribution of CNNs' activations and weights that followed Gaussian-like distributions into consideration to achieve a balanced tradeoff among latency, power, and accuracy. These above-mentioned approximate 4-2 compressors could rapidly reduce the partial products and had lower timing delay, cell area, and power consumption when compared to the accurate 4-2 compressor.

C. ADJUSTABLE APPROXIMATE COMPUTING

Akbari *et al.* [15] proposed four adjustable approximate 4-2 compressors, which are all composed of a supplementary part and an approximate part with the difference in the structure of the approximate part. There are two operation modes: accurate mode and approximate mode. In the accurate mode, the supplementary part is active and fine-tunes the results output from the approximate part to produce accurate results. While in the approximate mode, the supplementary part is powered down with power gating so that the approximate result is directly obtained as the final result. Yang et al [16] modified the accurate 2-2 compressor (half-adder) and the 3-2 compressor (full adder). The modified compressors have a mask signal to control whether the result is approximate or accurate. Guo *et al.* [18] divided a multiplier into two sub-multipliers to design an adjustable approximate multiplier that supports multiplications with various input bit widths. De la Guia Solaz *et al.* [4] proposed a programmable truncated multiplier. They replaced the 2-input AND gates with the 3-input AND gates for generating the partial product elements with truncation ability. When high accuracy is not required, some partial product columns are truncated at runtime. Hammad *et al.* [17] analyzed the influence of bit-width precision on accuracy and proposed the concept of predicting and dynamically configuring the precision of approximate multipliers for CNN inference with a precision preprocessor.

D. EVALUATION METRICS

To compare the approximate multipliers, there are some metrics for analyzing the approximate multipliers. For any $N \times N$ multiplier, error rate (ER) is the ratio of number of the erroneous results with respect to the total test case. ER can be expressed by (3). Error distance (ED) is the difference between the accurate result (S) and erroneous result (S^*), expressed by (4). Relative error distance (RED) is defined as the ratio of ED to accurate output, expressed by (5). Mean error distance (MED) is the average of the ED values and is defined as (6). Mean relative error distance (MRED) is defined as the average value of RED, expressed by (7). Worst error distance (WED) is the max value of the total ED, expressed by (8). Normalized mean error distance (NMED) is defined as (9), where Smax is the max value of the accurate results.

$$ER = no.of \quad erroneous\ results/22N \tag{3}$$

$$ED = \left|S - S^*\right| \tag{4}$$

$$RED = \left|S - S^*\right| /S \tag{5}$$

$$MED = \frac{\Sigma_1^{2^{2N}} ED}{2^{2N}} \tag{6}$$

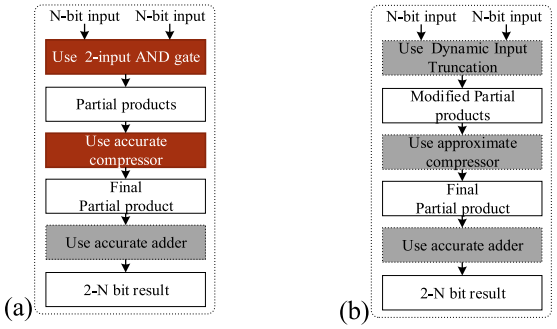

FIGURE 3. (a) Traditional and (b) our proposed multiplication flow.

$$MRED = \frac{\Sigma_1^{2^{2N}} RED}{2^{2N}} \tag{7}$$

$$WED = ED_{max} \tag{8}$$

$$NMED = MED/S_{max} \tag{9}$$

III. PROPOSED APPROXIMATE MULTIPLIER

This section first introduces the difference between the traditional and our proposed multiplication flow. Then, our proposed high-accuracy 4-2 compressor and a proposed error compensation circuit are introduced. Afterwards, it proceeds to dynamic input truncation that is used to construct the adjustable multiplier. Finally, it introduces the overall architecture of the proposed approximate multiplier and explains why our proposed multiplier is suitable for CNNs.

A. PROPOSED FLOW AND APPROXIMATE MULTIPLIER

FIGURE 3(a) shows the overall flow of the traditional flow for multiplication that generates accurate results. First, accurate partial products are produced using 2-input AND gates, and later compressed by the accurate compressors. Finally, accurate adders sum the compressed partial products to generate the result. FIGURE 3(b) shows our proposed flow for the proposed approximate multipliers. The differences between traditional multiplication and the proposed multiplication are the steps of generating partial products and compressing the partial products, which will be introduced in Section III.B. In the step of generating partial products, we use the dynamic input truncation that will be introduced in Section III.C to generate the modified partial products.

B. PROPOSED HIGH-ACCURACY 4-2 COMPRESSOR

In this paper, a high-accuracy and low-power approximate 4-2 compressor is proposed. The proposed 4-2 approximate compressor is shown in FIGURE 4. The design of the proposed 4-2 approximate compressor is described as follows. Four inputs $X_1 \sim X_4$ are used to generate $W_1 \sim W_4$ using Eqs. (10)-(13). Because an incorrectly computed carry bit has a higher error distance than the sum bit, i.e., an incorrect carry bit produces two times ED of that produced by an incorrect sum bit, the carry bit in the proposed compressor is designed always to be correctly generated. The equations of generating carry bit are shown in (14)-(16). The carry bit will become 1 under three circumstances. One is X_1 and X_2 are both 1.

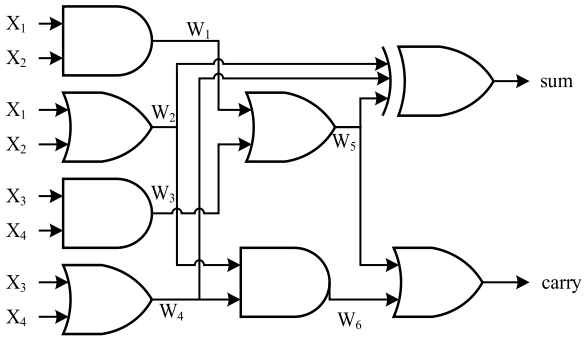

FIGURE 4. Gate-level implementation of proposed 4-2 compressor.

Another is X_3 and X_4 are both 1. The third is either of X_1 or X_2 is 1 and either of X_3 or X_4 is 1. (14) checks the first two situations, and (15) checks third situation. (16) produces the final carry bit.

$$W_1 = X_1 \; AND \; X_2 \tag{10}$$

$$W_2 = X_1 \; OR \; X_2 \tag{11}$$

$$W_3 = X_3 \; AND \; X_4 \tag{12}$$

$$W_4 = X_3 \; OR \; X_4 \tag{13}$$

$$W_5 = W_1 \; OR \; W_3 \tag{14}$$

$$W_6 = W_2 \; AND \; W_4 \tag{15}$$

$$Carry \; = W_5 \; OR \; W_6 \tag{16}$$

The proposed equation to generate the sum bit is shown in (17). In an accurate 4-2 compressor, the sum bit is generated with four XOR gates built within the two full adders. Whereas in our proposed compressor, we generate the sum bit by inputting W_2 and W4 into a 2-input XOR gate to utilize the signals that are used to generate the carry bit. By sharing the common signals, we can reduce the circuit area and static power consumption. However, we found that the error distance is large if only W_2 and W4 are fed into a 2-input XOR gate. Because W_2 and W_4 are generated with OR gates, the error occurs either when both X_1 and X_2 are 1 or both X_3 and X_4 are 1, which lead the sum bit to the result of 1 when it is supposedly 0. To achieve high accuracy, we add W5, the signal used to detect these two cases, into the XOR gate. For example, if both X_1 and X_2 are 1, both W_2 and W_5 will be 1, and the sum bit will turn out to be '0 XOR W_4', resulting in W_4 as the sum bit. In this case, the number of bits that need to be considered are only X_3 and X_4. However, when all four inputs are 1, the sum bit turns out to be 1, resulting in the error distance of 1.

$$Sum = W_5 \; XOR \; W_2 \; XOR \; W_4 \tag{17}$$

TABLE 1 is the truth table of our proposed approximate 4-2 compressor. An error will only occur when all four inputs equal to 1. The probability of a partial product being 1 is 1/4 considering the probability of a bit in the multiplicand and a bit in the multiplier both being 1 is $(1/2)^2$, so the probability of four inputs being all 1 is only $(1/4)^4$. And even if the error happens, the difference between accurate output and our

TABLE 1. Truth table of our proposed approximate 4-2 compressor design.

X_3	X_3	X_2	X_1	carry	sum	diff.
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	1	0	0
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	1	0	0
0	1	1	1	1	1	0
1	0	0	0	0	1	0
1	0	0	1	1	0	0
1	0	1	0	1	0	0
1	0	1	1	1	1	0
1	1	0	0	1	0	0
1	1	0	1	1	1	0
1	1	1	0	1	1	0
1	1	1	1	1	1	-1

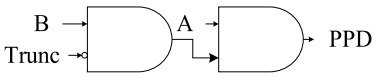

FIGURE 5. Modified partial product.

output is only 1 which is negligible.

$$Error = W_1 \; AND \; W_3 \tag{18}$$

For the error detection purpose, we only need an extra AND gate to detect whether both W_1 and W_3 are 1, because W_1 uses an AND gate to detect whether both X_1 and X_2 are 1, and W_3 uses an AND gate to detect whether both X_3 and X_4 are 1. The equation of the error detection circuit (EDC) is shown in (18). Therefore, the error compensation circuit of the proposed 4-2 compressor can be easily constructed by adding an extra AND gate.

C. DYNAMIC INPUT TRUNCATION

To achieve an adjustable approximate multiplier at runtime, we propose a dynamic input truncation technique, which uses two 2-input AND gates, as shown in FIGURE 5, to produce a partial product, whose equation is shown in (19), where A is the multiplicand and B is the multiplier. The Trunc signal is used to determine whether the partial product PPD should be truncated. If the Trunc is 1, the partial product is truncated to 0. To be more precise, the Trunc signals save the power by truncating the PPDs in the multiplications to zeros. In other words, we can think of the Trunc signals serving the role as disabling the hardware units in the corresponding columns.

$$PPD_{ij} = (\sim Trunc \; AND \; B_i) \; AND \; A_j \tag{19}$$

For an 8 × 8 multiplier, each bit of the multiplier is corresponded to 8 bits of the multiplicand; therefore, we propose to reduce hardware costs by sharing gates with an extra AND gate. For example, PPD_{00} equals $\sim Trunc_0 \cdot B_0 \cdot A_0$ and PPD_{01} equals $\sim Trunc_0 \cdot B_0 \cdot A_1$. In this case, $\sim Trunc_0 \cdot B_0$ can be computed in advance to generate a mask, and three 2-input AND gates are required, as shown in FIGURE 6. The control of the Trunc signals in the proposed approximate multiplier will be discussed in more details in the next subsection.

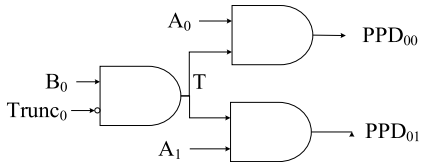

FIGURE 6. A gate sharing example to reduce the number of gates.

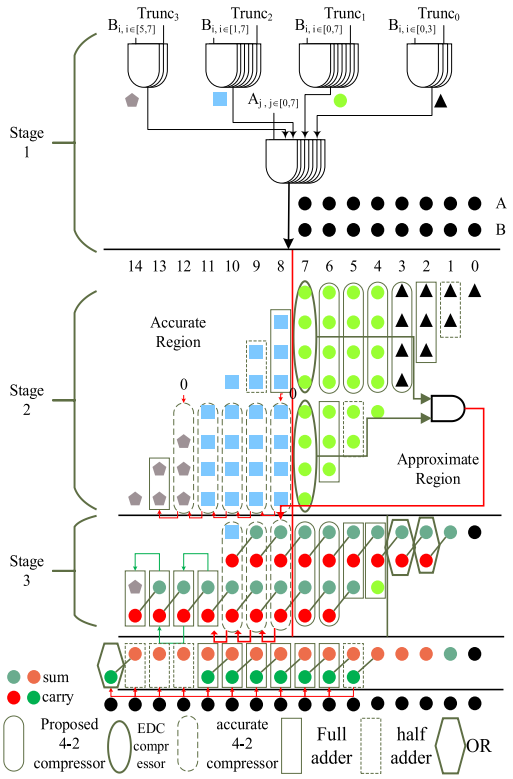

FIGURE 7. Proposed approximate multiplier.

D. THE PROPOSED APPROXIMATE MULTIPLIER

FIGURE 7 shows an approximate multiplier with the proposed technique. Even though the input width of the multiplier is designed to be 8-bit, the proposed technique can still be extended to larger multipliers. The proposed approximate multiplier contains three stages. In the first stage, each partial product is generated by two 2-input AND gates as shown previously in FIGURE 5 with the gate sharing technique applied in FIGURE 6 to further reduce hardware costs. Depending on the requirements, the accuracy of the generated partial product can be determined based on the Trunc signal. In our proposed approximate multiplier, to make the control more efficient as well as to reduce the hardware costs, we design a 4-bit Trunc signal with each bit to control more than one partial product column, which we call "3-4-4-4 partition", specifically, each bit from MSB to LSB to control column 14th∼12th, 11th ∼8th, 7th∼4th and 3rd ∼0th respectively, corresponding to the color of khaki, sky blue, green and black in Stage 2 in FIGURE 7. For example, if the $Trunc_{(3-0)}$ is 0101_2, column 14th ∼12th and 7th ∼4th are accurate, and column 11th ∼8th and 3rd ∼0th are truncated.

Different options of controlling Trunc signals lie in the way the columns are partitioned, which allows the users to flexibly modify the proposed multiplier based on their requirements. We have conducted several experiments to try out different partitions, and the results show that 3-4-4-4 partition as well as 3-3-3-3-3 partition both keep the balance between power saving, accuracy, and area overhead. The finer the partition is, the more flexible it is for controlling the amount of power to be saved and accuracy losses. However, it will in exchange suffer from high area overhead. As a result, 3-4-4-4 partition is used throughout our experiments, and in the case study in Section V we will further compare the CNN results produced by 3-4-4-4 partition with 3-3-3-3-3 partition.

The second stage shows the steps of compressing the partial products. After the partial products are generated, they are divided into two regions: column 14th∼8th being the accurate region, and 7th∼0th being the approximate region. The split of accurate and approximate region is decided from the most intuitive half-half separation. If we do 30-70 split, for example, with too many computations done by the approximate multiplier, the accuracy loss will be significant. On the other hand, if we do 70-30 split, the effect of the approximate computing for power reduction will be little. Because the weight of the partial products in the accurate region is higher and more important, we compress the partial products in that region with accurate 4-2 compressors. On the other hand, we use our proposed approximate 4-2 compressors and error compensation circuit to compress the partial products in the approximate region.

In the third stage, we use OR gates in columns 3rd ∼0th to generate results and ignore carry propagation considering them close to LSB, whose errors have less effect on the final results. We detect errors in the second stage with the EDC, i.e., a single AND gate, to determine whether the compensation bit should be produced. We use the proposed approximate 4-2 compressors, accurate 4-2 compressors, full adders, and half adders to compress the partial products in the remaining columns. After finishing the third stage, we get the final two partial product rows, which is summed up by using accurate adders to produce the final results.

IV. EXPERIMENTAL SETUP AND RESULTS

This section first introduces the experimental setup that is used to evaluate the approximate multipliers. Then, it compares the approximate multipliers under different evaluation metrics. Finally, it compares the critical path delay, cell area, power consumption, and power-delay product (PDP).

A. EXPERIMENTAL SETUP

The RTL codes of the approximate multipliers are implemented in Verilog HDL, and NCSim is used to simulate the approximate multipliers and generate waveforms where the switching activities of logic gates are recorded. Design Compiler is used to synthesize the RTL codes to generate the gate-level netlists with standard UMC 0.18m CMOS cell-library. PrimeTime PX is used to estimate the power consumption using waveforms. All approximate multipliers are synthesized and optimized with the same options.

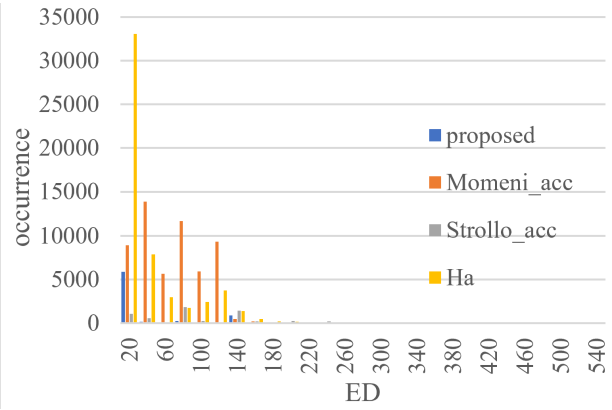

FIGURE 8. Error distribution.

We compare the proposed approximate multipliers to the accurate Wallace Tree multiplier [6] and previous approximate multipliers [3], [8], [9], [13], [16]. Two approximate multiplier designs are selected from Momeni *et al.* [8]. The first one, called Momeni_fast, has a higher speed but lower accuracy due to the fact that the 4-2 compressors used in the multiplier are all approximate. Another one, called Momeni_acc, has higher accuracy because exact 4-2 compressors are used in MSB columns and approximate 4-2 compressors are used in LSB columns. Three different approximate 4-2 compressors are proposed by Yang *et al.* [9] to construct three different Dadda multipliers. These multipliers are referred to Yang_high, Yang_medium, and Yang_low, respectively, straightforwardly meaning that the accuracy of Yang_high is higher than Yang_medium, and the accuracy of Yang_medium is higher than Yang_low. Ha *et al.* [3] proposed an approximate 4-2 compressor to build an approximate multiplier, which is referred to as Ha with error-correction circuits implemented. An accuracy-adjustable approximate multiplier design Yang_adj proposed in [16] is also listed in our comparison. Strollo *et al.* [13] proposed two different approximate multipliers. The first one uses only approximate 4-2 compressors, and the second one with both approximate and accurate compressors included called Strollo_acc, has higher accuracy and is used for our comparison.

The proposed approximate multiplier, as shown in FIGURE 7, contains high-accuracy 4-2 compressors, a simple error compensation circuit, and dynamic input truncation. Since the bit width of the Trunc signal is 4-bit, there are sixteen different configurations in our proposed approximate multiplier, e.g., the proposed_0000 indicates Trunc = 0000 (no truncation), and the proposed_0011 indicates Trunc = 0011. However, only three configurations, proposed_0000, proposed_0001, and proposed_0011, are adopted in our comparison for their more accurate results resulting from the untruncated eight leftmost partial product columns.

TABLE 2. Comparison on different accuracy metrics.

Design	ER (%)	MED	RED	MRED	WED	NMED
Momeni_fast [8]	99.30	3518	278284	4.246	8640	$5.4*10^{-2}$
Momeni_acc [8]	85.77	51.43	2918	0.044	200	$7.9*10^{-4}$
Yang_high[9]	81.29	15.33	382.1486	0.005	561	$2.3*10^{-4}$
Yang_medium[9]	83.47	25.73	503.823	0.007	561	$3.9*10^{-4}$
Yang_low[9]	83.74	35.09	556.469	0.008	577	$5.4*10^{-4}$
Ha[3]	82.91	28.05	514.25	0.007	385	$4.3*10^{-4}$
Yang_adj[16]	36.16	164.47	555.23	0.008	7716	$5.2*10^{-2}$
Strollo_acc[13]	9.29	8.46	50.122	0.0007	536	$1.3*10^{-4}$
Proposed_0000	11.57	3.957	33.59	0.0005	452	$6.1*10^{-5}$
Proposed_0001	81.29	15.39	381.00	0.0058	481	$2.3*10^{-4}$
Proposed_0011	98.04	448	6359	0.097	1793	$6.9*10^{-3}$

TABLE 3. Latency, area and power evaluations.

	Latency	Area	Power	PDP
Exact	100%	100%	100%	100%
Yang_adj [12]	81%	90%	84%	68%
Yang_high [15]	76%	86%	86%	66%
Strollo_acc [7]	70%	90%	92%	64%
Proposed_0000	73%	93%	82%	**60%**
Proposed_0001	73%	93%	69%	**51%**
Proposed_0011	73%	93%	28%	**21%**

TABLE 4. Area comparison for adjustable multiplier.

Original Area of Approximate Multiplier	Area Overhead for Programmable Truncation in [4]	Area Overhead for the Proposed Truncation
4587	487	206 (-42%)

B. ACCURACY COMPARISON

TABLE 2 compares the accuracy metrics of different approximate multipliers from different works [3], [8], [9], [13], [16]. The experimental data are identical to the data provided from each work to guarantee the reliability of the results. We use blue to mark the best result, green for the second and yellow for the third. In terms of error rate (ER), the proposed multiplier Proposed_0000 has the second-lowest error rate 11.57% following 9.29% ER of Strollo_acc. The low ER of our proposed approximate multiplier results from the use of high accurate 4-2 approximate compressors in LSB, accurate 4-2 compressors in MSB, and the error compensation circuit to reduce errors. As mentioned in Section III.D, with columns 3rd ∼0th in the partial products generating the results with OR gates, it sacrifices little accuracy for efficiency, which is a possible reason in narrow defeat of 2.3% by Strollo_acc. Regarding MED, NMED, and MRED, Proposed_0000 outstands among the others. As for WED, Proposed_0000 has the smallest value except for Momeni_acc and Ha. However, the fact that Proposed_0000 performs better in MED and MRED compared to Momeni_acc and Ha indicates that the worst error distance in our proposed multiplier occurs less frequently, and therefore the MED and MRED are lower than those of Ha and Momeni_acc.

FIGURE 8 further compares the error distance distribution of the proposed multiplier Proposed_0000, Momeni_acc, Ha, and Strollo_acc. The x-axis is the error distance, and the y-axis is the number of occurrences. Most of the errors generated by the proposed multiplier have smaller error distances and lower occurrences compared to Momeni_acc and Ha, which confirms again that although our WED is larger than Ha and Momeni_acc, the MED and MRED of our proposed multiplier are better than Ha, Momeni_acc, and Strollo_acc. In addition, most errors occurring in our proposed multiplier have a smaller ED than those in Strollo_acc, which explains the reason that although the ER of the proposed multiplier is higher than the ER of Strollo_acc, the MED in our proposed multiplier is smaller than the MED of Strollo_acc.

The performance of Proposed_0001 in terms of the overall comparison metrics in accuracy is comparable to Yang_high; therefore there is little need to compare with Yang_medium and Yang_low, and it shows better overall results than Ha and Momeni_acc. As for Proposed_0011, since more than half of the bits are truncated, low accuracy is expected. However, even if the accuracy performance is not ideal, Proposed_0011 shows enormous reduction in power consumption (TABLE 3) that allows the proposed multiplier to find a suitable combination of different configurations to achieve low power consumption in CNN applications, which will be detailed in Section V.

C. LATENCY, AREA AND POWER COMPARISON

TABLE 3 presents the latency, area, and power evaluations of accurate and approximate multipliers, where the accurate multiplier is built using 8 × 8 Wallace Tree Multiplier with accurate compressors. The proposed multipliers are compared with the multipliers with comparable results in TABLE 2.

Our proposed multiplier reduces 27% of latency and 7% of area overhead over accurate multiplier. Moreover, it shows good power reduction. The *Proposed_0011* has the maximum power reduction ratio, followed by *Proposed_0001*, *Proposed_0000, Yang_adj, Yang_high* and *Strollo_acc*. The reduction in power consumption in *Proposed_0011* is up to 72%, which is an incredible amount of power saving, and there is 31% and 18% of power reduction in *Proposed_0001* and *Proposed_0000* respectively. Overall, the proposed multiplier has the average power consumption reduction of 40.33%. Based on the requirement, users can choose different configurations of our design to achieve better accuracy or energy efficiency.

Power delay product (PDP) is the product of power and timing delay, which reflects both power dissipation and the propagation delay. It provides us a more comprehensive view on the whole system that balances the performance and the energy consumption. The *Proposed_0011* shows the lowest value in PDP, which implies that in terms of the balance in power consumption and timing delay, *Proposed_0011* has the best performance. At the same time, with low delay and energy consumption, we desire good performance in accuracy. As we mentioned previously, accuracy/energy consumption is a trade-off problem; there is no doubt that the multipliers with lower accuracy are more probable to have lower energy consumption. Taking accuracy into account, *Proposed_0000* has undeniably good accuracy compared to the others, and it still has 8% and 4% PDP reduction

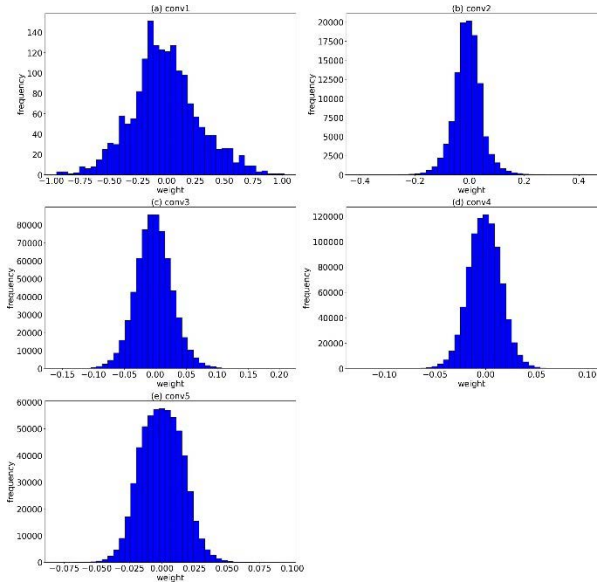

FIGURE 9. Distribution of CNNs weights in AlexNet. (a) ~ (e) represents conv1~conv5.

compared to *Yang_adj* and *Strollo_acc*. *Proposed_0001* performs comparably in accuracy compared to *Yang_high*, while it exceeds up to 15% in the PDP reduction.

D. AREA OVERHEAD COMPARISON

TABLE 4 shows the area comparison of the multiplier with programmable truncation proposed in [4] and our proposed multiplier with the dynamic input truncation technique. Table 3 shows the area overhead of the elements featuring reconfigurability over our proposed approximate multiplier without reconfigurability, which has the original area of 4587. To be more specific, if we apply components for reconfigurability such as Trunc, gate sharing circuit, etc. on our proposed approximate multiplier, there will be 206 extra area. On the other hand, if the Programmable truncation circuit proposed in [12] is applied on our proposed multiplier, there will instead be 487 extra area. The area overhead of the elements for reconfigurability of the proposed multiplier is greatly reduced by 42% under the same accuracy. [4] controls one partial product column at a time, while in our proposed approximate multiplier, one Trunc bit controls more than one partial product column with dynamic input truncation. Therefore, dynamic input truncation can significantly decrease the extra hardware cost for implementing the adjustable design.

V. CASE STUDY

CNNs have developed rapidly in the past few years and become more and more popular. It can be used in many fields, such as image recognition, self-driving cars, cancer detection, complex game playing, and natural language processing. The following is the demonstration of the effectiveness of our proposed design in CNNs. In the case study of CNNs, we utilize the reconfigurability of the proposed multiplier by changing the *Trunc* signal to switch it from *Proposed_0000* to

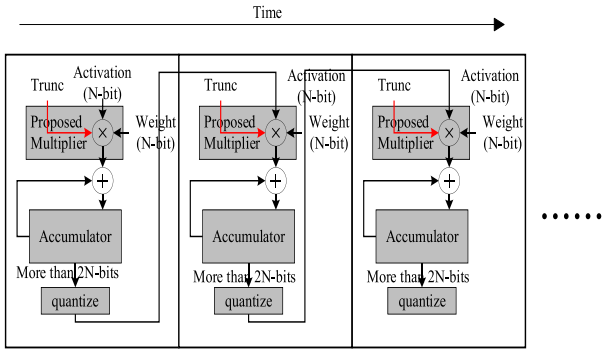

FIGURE 10. Flowchart for applying the proposed multiplier in quantized CNNs.

Proposed_0001 or *Proposed_0011* in each layer, so that we can obtain the high-accuracy results with low power consumed.

A. CNN APPLICATION

As mentioned in the Section II, CNN needs a large number of multiplication and accumulation (MAC) operations that result in large power consumption. Because CNNs can tolerate some accuracy losses, the approximate multiplier is suitable for CNNs. By replacing accurate multipliers with approximate multipliers, CNN can achieve better energy efficiency with little or no accuracy loss.

The original weights of a CNN are represented in floating-point numbers, and they are normally quantized into fixed-point numbers to be implemented in CNNs on hardware. FIGURE 9 shows the weight distribution of AlexNet at different layers. It can be seen that the weight distributions of the CNN are different in each layer; therefore, after quantization, the fraction point positions of the quantized weights vary at different layers. FIGURE 10 is the flowchart of applying the proposed multiplier in CNNs with quantized weights. The layers are computed sequentially with time-complexed hardware. The convolutional computations are done layer by layer. For simplification, the flowchart only shows the first three convolution layers, in which the results of one layer are forwarded to the next layer. In each convolution layer, the bit width of weights and activations are both N-bit, whose multiplication produce results of 2N-bit width. Because a neural network layer usually has multiple channels, e.g., AlexNet has 3 channels in the first layer, the accumulators from the previous channel must be added; therefore, the bit width of the result in the final accumulator will exceed 2N-bit. The result must be quantized into N-bit before it is passed into the next convolution layer, which means that some bits in the result will be discarded, and consequently, not all the bits of the partial products in the multiplication need to be accurately computed.

Our proposed approximate multiplier is suitable for quantized CNNs because we can adjust our proposed approximate multiplier to determine the bits that are allowed to not be accurately computed at run-time. In each convolution layer,

we use a 4-bit truncated configuration parameter *Trunc* to indicate whether the columns of the partial product are discarded, as shown previously in FIGURE 7. By using the truncation parameter, we can dynamically adjust our proposed multiplier based on the decimal point positions of the quantized weights to decrease the power consumption from numerous MACs.

B. QUANTIZATION

The weights of CNN are normally in the format of floating-point numbers, which results in high hardware cost for the implementation of CNNs. Therefore, in this case study, the floating-point weights are first converted into 8-bit integer fixed-point weights before being implemented in the CNN with the quantization method proposed in [19]. Seeing that the weight distribution of each layer varies as shown in FIGURE 9, we generate a scale in the floating-point format that represents the distribution of each layer. Before the results from the previous layer are passed into the next layer, they should be normalized with the scale, where the right shift operation is adopted with the scale approximated in the power of 2 to replace division for the purpose of reducing the computation complexity. Then, the approximated scale in the power of 2 allows us to determine the number of bits to be shifted, which is referred to as the *shift number* in the following text.

Because the results should be shifted before they are passed into the next layer, some bits of the results will be discarded; as the result, the accuracy of the result will not be greatly affected even if those bits that are supposed to be removed are truncated. Therefore, we utilize the *shift number* for each layer to decide the *Trunc* signal. The greater the *shift number* is, the more bits we can truncate without influencing too much the results. With *shift number* obtained from each layer, we try out several *Trunc* signals and select the one that gives good balance in accuracy and power consumption. In other words, we apply different configurations of the proposed approximate multiplier based on the weight distributions in each layer, which will be discussed in the next subsection.

C. RESULT DISCUSSION

Both VGG11 and AlexNet network are used in our case study with Cifar10 as the dataset. VGG11 contributes 11 weighted layers, including 8 convolutional layers and 3 fully connected layers. AlexNet consists of 8 weighted layers with 5 convolutional layers and 3 fully connected layers.

FIGURE 11 shows the flow chart of the experimental procedure. In our experiment, we first obtain the weights trained and quantized from model implemented with Python, and we proceed to the inference phase. In the inference phase, the convolutional layers are simulated with C++, in which the multiplications of the convolutional layers are done by Verilog HDL and designed to be time multiplexed. In the inference phase shown in FIGURE 11, a test image is first fed to the CNN layers with the trained integer weights input from the training phase, and the convolutional computation for the first layer starts. The computations are done layer by

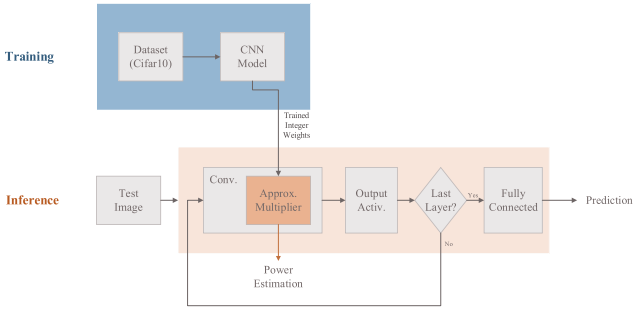

FIGURE 11. Experimental Flow Chart for CNNs.

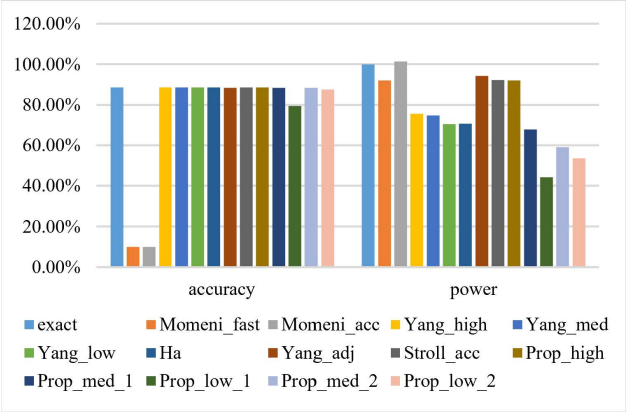

FIGURE 12. Accuracy and Power comparison in VGG11.

layer, and the power consumption, area and timing delay are estimated. When the computations of the last convolutional layer are finished, the output activation is fed to the fully connected layer, and the prediction of the test image is made. The fully connected layers that predict the categorization of the image are implemented with software.

The accurate multipliers in the network are replaced with the above-mentioned approximate multipliers and our proposed multiplier for the comparison. Based on the design described in Section III.D, three approximate multipliers are used: *Proposed_High_acc, Proposed_Medium_acc* and *Proposed_Low_acc*. *Proposed_High_acc* is designed to have high accuracy without any bit truncated in the partial products, i.e., with *Trunc* = 4'b0000, so it has the power mainly saved from the simpler design of approximate 4-2 compressors compared to accurate 4-2 compressors as well as the OR gates used in replacement of adders in LSB. *Proposed_Medium_acc* and *Proposed_Low_acc* apply not only the proposed approximate compressors, but also the dynamic input truncation, making use of the reconfigurability of our proposed multiplier to truncate different numbers of bits in different layers in CNNs.

1) VGG11

TABLE 5 shows the best Trunc signal chosen from the experimental results of the proposed multiplier with 3-4-4-4 partition and 3-3-3-3-3 partition in VGG11. Because the scales of each layer are different, Trunc signal are adjusted

TABLE 5. He trunc parameter of proposed multiplier in VGG11.

Partition	Name	Convolutional Layers (Conv1~Conv8)
3-4-4-4	Proposed_ High_acc	$Trunc = 4'b0000$
	Proposed_ Medium_acc	$Trunc = \begin{cases} 4'b0000, if\ shift < 5 \\ 4'b0001, else\ if\ shift < 11 \\ 4'b0011, else\ if\ shift < 13 \\ 4'b0111, else \end{cases}$
	Proposed_ Low_acc	$Trunc = \begin{cases} 4'b0000, if\ shift < 5 \\ 4'b0001, else\ if\ shift < 10 \\ 4'b0011, else\ if\ shift < 13 \\ 4'b0111, else \end{cases}$
3-3-3-3-3	Proposed_ High_acc	$Trunc = 5'b00000$
	Proposed_ Medium_acc	$Trunc = \begin{cases} 5'b00000, if\ shift < 5 \\ 5'b00001, else\ if\ shift < 10 \\ 5'b00011, else\ if\ shift < 12 \\ 5'b00111, else \end{cases}$
	Proposed_ Low_acc	$Trunc = \begin{cases} 5'b00000, if\ shift < 5 \\ 5'b00001, else\ if\ shift < 9 \\ 5'b00011, else\ if\ shift < 12 \\ 5'b00111, else \end{cases}$

TABLE 6. Accuracy and power consumption comparison in VGG11 of other approximate multipliers.

Design	Accuracy	Power
Exact	88.69%	100%
Momeni_fast [8]	10% (-78.69%)	92% (-8%)
Momeni_acc [8]	10% (-78.69%)	101% (+1%)
Yang_high[9]	88.63% (-0.06%)	76% (-24%)
Yang_medium[9]	88.63% (-0.06%)	75% (-25%)
Yang_low [9]	88.68% (-0.01%)	71% (-29%)
Ha[3]	88.64% (-0.05%)	71% (-29%)
Yang_adj[16]	88.42% (-0.27%)	94% (-6%)
Strollo_acc[13]	88.61% (-0.08%)	92% (-8%)

TABLE 7. Accuracy and power consumption comparison in VGG11 of the proposed multiplier.

Partition	Design	Accuracy	Power
-	Exact	88.69%	100%
3-4-4-4	Proposed_High_acc_1	88.60% (-0.09%)	92% (-8%)
	Proposed_Medium_acc_1	88.45% (-0.24%)	**68% (-32%)**
	Proposed_Low_acc_1	79.40% (-9.29%)	**44% (-56%)**
3-3-3-3-3	Proposed_High_acc_2	88.60% (-0.09%)	92% (-8%)
	Proposed_Medium_acc_2	88.39% (-0.30%)	**59% (-41%)**
	Proposed_Low_acc_2	87.56% (-1.13%)	**54% (-46%)**

based on the shift number determined by the scale at runtime. For example, with 3-4-4-4 partition, if the shift number is 10 in the first convolutional layer and Proposed_Medium_acc is chosen, the Trunc signal will be 0001 in Conv1, whereas under if Proposed_Low_acc is chosen, the Trunc signal will be 0011 in Conv1; if the shift number is 4 in the second convolutional layer with Proposed_Medium_acc chosen, Trunc signal will be 0000 in Conv2.

TABLE 6 and TABLE 7 show the accuracy and the power consumption of different approximate multipliers and the proposed multipliers applied in the VGG network, respectively. With only accurate multipliers used, VGG11 network provides the accuracy of 88.69%, and the power consumption is set as 100% as a reference.

The accuracy and the power consumption of the proposed multiplier are plotted with the bar graph shown in FIGURE 12 for giving a clearer overview of the comparison. Most of the approximate multipliers give relatively accurate results close to the one given by the accurate multipliers except for *Momeni_fast* and *Momeni_acc*. *Momeni* approximate multipliers produce errors when the input bits are all zeros, i.e., they produce non-zero values, which clearly shows their unsuitableness in CNNs. It is because most of the CNNs including VGG11 and AlexNet, apply ReLU activation function, through which the negative results are rectified to zeros. These zero outputs normally do not to contribute to the power consumption; however, *Momeni* multipliers do not produce zeros, so they continuously consume power when it's not supposed to, which consequently results in the abnormal 101% power consumption that is higher than the reference.

As shown in TABLE 7, in 3-4-4-4 partition, *Proposed_Medium_acc* and *Proposed_Low_acc* have the top two performances in terms of power consumption in comparison with the power of other approximate multipliers presented in TABLE 6. *Proposed_Medium_acc* saves 32% of power in compromise of only 0.24% loss of accuracy, which shows great balance in accuracy and power consumption compared to other approximate multipliers. As for *Proposed_Low_acc*, it loses 9.29% of accuracy in exchange for 56% less power consumption. If the application does not require high precision, *Proposed_Low_acc* can be a suitable option if the power saving is the priority

Except for 4-bit *Trunc* signals, we also try out 5-bit *Trunc* signals, i.e., 3-3-3-3-3 partition, with each bit controlling column 15^{th} ~13^{th}, 12^{th} ~10^{th}, 9^{th} ~7^{th}, 6^{th} ~4^{th}, and 3^{rd} ~1^{st} respectively. In 3-3-3-3-3 partition, one bit of *Trunc* signal controls only 3 partial columns in contrast with that in 3-4-4-4 partition controlling 4 partial columns (except for the MSB); by partitioning the *Trunc* signal into a more fine-grained manner, it allows us to further fine-tune the results. Even though one extra bit of *Trunc* signal causes slightly higher area overhead, it can lead to substantial increase in the potential of accuracy or power saving. Without any bit truncated, the *Proposed_High_acc* in 3-4-4-4 and 3-3-3-3-3 partition are basically the same. As for the other two proposed approximate multipliers, *Proposed_Medium_acc* has slight decrease in 0.06% of accuracy and 9% more of power reduction, while *Proposed_Low_acc* improves 8.16% in accuracy with a loss of 10% power reduction. *Proposed_Low_acc* only has 1.13% difference of accuracy with the exact value, but still shows the lowest power consumption among all the approximate multipliers in TABLE 6. Note that the *Trunc* parameters are chosen by ourselves from multiple experimental results, to show that 3-3-3-3-3 partition is endowed with higher potential in improving accuracy and power consumption, we present the one that performs much better in power saving in sacrifice of small accuracy loss, and the one that performs better in accuracy in exchange for power consumption.

2) ALEXNET

TABLE 8, in which the *shift* represents the *shift number*, lists the *Trunc* signals we choose according to different *shift*

TABLE 8. The trunc parameter of proposed multiplier in AlexNet.

Partition	Name	Convolutional Layers (Conv1~Conv8)
3-4-4-4	Proposed_ High_acc	$Trunc = 4'b0000$
	Proposed_ Medium_acc	$Trunc = \begin{cases} 4'b0000, if\ shift < 5 \\ 4'b0001, else\ if\ shift < 13 \\ 4'b0011, else\ if\ shift < 21 \\ 4'b0111, else \end{cases}$
	Proposed_ Low_acc	$Trunc = \begin{cases} 4'b0000, if\ shift < 5 \\ 4'b0001, else\ if\ shift < 12 \\ 4'b0011, else\ if\ shift < 21 \\ 4'b0111, else \end{cases}$
3-3-3-3-3	Proposed_ High_acc	$Trunc = 5'b00000$
	Proposed_ Medium_acc	$Trunc = \begin{cases} 5'b00000, if\ shift < 5 \\ 5'b00001, else\ if\ shift < 11 \\ 5'b00011, else\ if\ shift < 12 \\ 5'b00111, else \end{cases}$
	Proposed_ Low_acc	$Trunc = \begin{cases} 5'b00000, if\ shift < 5 \\ 5'b00001, else\ if\ shift < 10 \\ 5'b00011, else\ if\ shift < 12 \\ 5'b00111, else \end{cases}$

TABLE 9. Accuracy and power consumption comparison in AlexNet of other approximate multipliers.

Design	Accuracy	Power
Exact	81.92%	100%
Momeni_fast [8]	10% (-71.92%)	99% (-1%)
Momeni_acc [8]	10% (-71.92%)	121% (+21%)
Yang_high[9]	80.75% (-1.17%)	68% (-32%)
Yang_medium[9]	81.23% (-0.69%)	68% (-32%)
Yang_low [9]	81.34% (-0.58%)	68% (-32%)
Ha[3]	80.58% (-0.44%)	64% (-36%)
Yang_adj[16]	81.40% (-0.52%)	97% (-3%)
Strollo_acc[13]	81.64% (-0.28%)	93% (-7%)

TABLE 10. Accuracy and power consumption comparison in AlexNet of the proposed multiplier.

Partition	Design	Accuracy	Power
-	Exact	81.92%	100%
3-4-4-4	Proposed_High_acc_1	81.63% (-0.29%)	91% (-9%)
	Proposed_Medium_acc_1	80.88% (-1.04%)	**58% (-42%)**
	Proposed_Low_acc_1	80.82% (-1.10%)	**57% (-43%)**
3-3-3-3-3	Proposed_High_acc_2	81.63% (-0.29%)	91% (-9%)
	Proposed_Medium_acc_2	81.30% (-0.62%)	**63% (-39%)**
	Proposed_Low_acc_2	78.71% (-3.21%)	**51% (-49%)**

numbers in AlexNet. The accuracy and power consumption under AlexNet architecture with the compared approximate multipliers and the proposed multiplier are shown in TABLE 9 and TABLE 10, respectively. Similar to the results in VGG11, *Momeni* is not suitable in AlexNet, while other approximate multipliers give relatively accurate results with certain decrease in power consumption. We can see that *Proposed_Medium_acc* and *Proposed_Low_acc* are the two that have the most power reduction, reducing 42% and 43% of power consumption with only up to 1.1% loss in accuracy with 3-4-4-4 partition. Similarly, we try out 5-bit *Trunc* signals (3-3-3-3-3 partition) under AlexNet architecture. The details of *Trunc* and the corresponding accuracy and power consumption are provided in the lower half of TABLE 8 and TABLE 10. We can see that *Proposed_Medium_acc* improves 0.42% in accuracy but decreases 5% in power reduction and *Proposed_Low_acc* increases 6% in power reduction but losses 2.11% in accuracy.

This section has described the methods used to demonstrate the effectiveness of the proposed multiplier in CNNs. It begins by describing the quantization method for reducing the computational complexity, shift number used for normalizing the varying weight distributions in each CNN layer, and the experimental environment, tools, and procedure. It goes on to present the results of the accuracy and power consumption of the proposed multipliers with two partition methods, 3-4-4-4 and 3-3-3-3-3 partition, in comparison with the approximate multipliers from other papers under two commonly used neural networks, VGG11 and AlexNet. Our proposed multiplier shows the flexibility in adjusting Trunc signals according to the needs; moreover, it presents the capability in achieving good accuracy with great reduction in power consumption.

VI. CONCLUSION AND FUTURE WORKS

In this paper, a high accuracy approximate 4-2 compressor that can be used to construct an approximate multiplier is proposed. The proposed approximate multiplier dynamically truncates partial products to adjust the accuracy and a simple error compensation circuit is used to reduce the error distance. The delay and the average power consumption of the proposed adjustable approximate multiplier is reduced by 27% and 40.33% (up to 72%), respectively, compared to the Wallace tree multiplier. Compared to other approximate multipliers, our proposed multiplier has the lowest mean error distance and lowest average power consumption. To demonstrate the effectiveness, the proposed multiplier is used in the inference of CNNs, with VGG11 and AlexNet as the models, and it shows great reduction in power consumption with little loss in accuracy. Our proposed multiplier has high reconfigurability with *Trunc* signals easily adjusted at runtime, which are chosen empirically from the *shift number* used in the quantization phase when the results are passed to the next layer, and it also has flexibility of modifying the *Trunc* signals with different partitions. Depending on the applications, high accuracy, low power consumption or balance between the two can be achieved.

However, we have yet found the optimal formulas to choose the best way of partitioning the *Trunc* signals. Future work concerns deeper analysis of different partition methods so that the relationship between hardware costs, accuracy, and power consumption can be derived in more concrete or mathematical expressions. For the current study, we can notice that the proposed adjustable approximate multiplier requires different *Trunc* signals for different networks, or more specifically, different convolutional layers to attain satisfying results. Future work shall put importance in this aspect. A feasible solution is to examine the features or properties of different convolutional layers to find out a most suitable set of parameters depending on the types of layers. By doing so, with those pre-examined parameters discovered, the users can apply the proposed multipliers in any CNN.

REFERENCES

[1] B. Moons and M. Verhelst, "DVAS: Dynamic voltage accuracy scaling for increased energy-efficiency in approximate computing," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, Jul. 2015, pp. 237–242.
[2] D. Mohapatra, V. K. Chippa, A. Raghunathan, and K. Roy, "Design of voltage-scalable meta-functions for approximate computing," in *Proc. Design, Autom. Test Eur.*, Mar. 2011, pp. 1–6.
[3] K. Yin Kyaw, W. Ling Goh, and K. Seng Yeo, "Low-power high-speed multiplier for error-tolerant application," in *Proc. IEEE Int. Conf. Electron Devices Solid-State Circuits (EDSSC)*, Dec. 2010, pp. 1–4.
[4] M. de la Guia Solaz, W. Han, and R. Conway, "A flexible low power DSP with a programmable truncated multiplier," *IEEE Trans. Circuits Syst.*, vol. 59, no. 11, pp. 2555–2568, Nov. 2012.
[5] R. Zendegani, M. Kamal, M. Bahadori, A. Afzali-Kusha, and M. Pedram, "RoBa multiplier: A rounding-based approximate multiplier for high-speed yet energy-efficient digital signal processing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 2, pp. 393–401, Feb. 2017.
[6] C. S. Wallace, "A suggestion for a fast multiplier," *IEEE Trans. Electron. Comput.*, vol. EC-13, no. 1, pp. 14–17, Feb. 1964.
[7] A. Weinberger, "4:2 carry-save adder module," *IBM Tech. Discl. Bull.*, vol. 23, no. 8, pp. 3811–3814, 1981.
[8] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, "Design and analysis of approximate compressors for multiplication," *IEEE Trans. Comput.*, vol. 64, no. 4, pp. 984–994, Apr. 2015.
[9] Z. Yang, J. Han, and F. Lombardi, "Approximate compressors for error-resilient multiplier design," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFTS)*, Oct. 2015, pp. 183–186.
[10] C.-H. Lin and I.-C. Lin, "High accuracy approximate multiplier with error correction," in *Proc. IEEE 31st Int. Conf. Comput. Design (ICCD)*, Oct. 2013, pp. 33–38.
[11] P. J. Edavoor, S. Raveendran, and A. D. Rahulkar, "Approximate multiplier design using novel dual-stage 4:2 compressors," *IEEE Access*, vol. 8, pp. 48337–48351, 2020.
[12] F. Sabetzadeh, M. H. Moaiyeri, and M. Ahmadinejad, "A majority-based imprecise multiplier for ultra-efficient approximate image multiplication," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 11, pp. 4200–4208, Nov. 2019.
[13] A. G. M. Strollo, E. Napoli, D. De Caro, N. Petra, and G. D. Meo, "Comparison and extension of approximate 4–2 compressors for low-power approximate multipliers," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 9, pp. 3021–3034, Sep. 2020.
[14] H. Xiao, H. Xu, X. Chen, Y. Wang, and Y. Han, "Fast and high-accuracy approximate MAC unit design for CNN computing," *IEEE Embedded Syst. Lett.*, early access, Dec. 21, 2021, doi: 10.1109/LES.2021.3137335.
[15] O. Akbari, M. Kamal, A. Afzali-Kusha, and M. Pedram, "Dual-quality 4:2 compressors for utilizing in dynamic accuracy configurable multipliers," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 4, pp. 1352–1361, Apr. 2017.
[16] T. Yang, T. Ukezono, and T. Sato, "A low-power high-speed accuracy-controllable approximate multiplier design," in *Proc. 23rd Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2018, pp. 605–610.
[17] I. Hammad, L. Li, K. El-Sankary, and W. M. Snelgrove, "CNN inference using a preprocessing precision controller and approximate multipliers with various precisions," *IEEE Access*, vol. 9, pp. 7220–7232, 2021.
[18] C. Guo, L. Zhang, X. Zhou, W. Qian, and C. Zhuo, "A reconfigurable approximate multiplier for quantized CNN applications," in *Proc. 25th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2020, pp. 235–240.
[19] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2704–2713.
[20] P.-Y. Chen, F.-Y. Gu, Y.-H. Huang, and I.-C. Lin, "WRAP: Weight RemApping and processing in RRAM-based neural network accelerators considering thermal effect," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2022, pp. 1245–1250.

FANG-YI GU (Graduate Student Member, IEEE) received the B.E. degrees in energy engineering and computer science from the National Cheng Kung University, Tainan, Taiwan, in 2021, where she is currently pursuing the Ph.D. degree with the Department of Computer Science and Information Engineering. Her research interests include very large-scale integration design and deep neural network accelerator.

ING-CHAO LIN (Senior Member, IEEE) received the M.S. degree in computer science from the National Taiwan University, Taipei, Taiwan, and the Ph.D. degree from the Department of Computer Science and Engineering, The Pennsylvania State University, State College, PA, USA, in 2007. His current research interests include very large-scale integration design and computer-aided design for nanoscale silicon, energy-efficient reliable system design, and computer architecture. He has been a Senior Member of ACM, since May 2016. He has served on the technical program committee of several conferences, such as ASP-DAC, ICCAD, ICCD, and GLSVLSI. He was awarded the Excellent Young Researcher Award by the Chinese Institute of Electrical Engineering, in 2015, and the Best Young Professionals (Formerly GOLD) Award by the IEEE Tainan Section, in 2016, and the Humboldt Fellowship for Experienced Researcher, in 2019.

JIA-WEI LIN (Graduate Student Member, IEEE) received the B.E. degree in management information system from the National Chung Cheng University, Chiayi, Taiwan. He is currently pursuing the master's degree in computer science and information engineering with the National Cheng Kung University, Tainan, Taiwan. His research interests include very large-scale integration design and deep neural network accelerator.