Li T, Jiang HL, Mo H *et al.* Approximate processing element design and analysis for the implementation of CNN accelerators. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 38(2): 309-327 Mar. 2023. DOI: 10.1007/s11390-023-2548-8

Approximate Processing Element Design and Analysis for the Implementation of CNN Accelerators

Tong Li[1] (李　彤), Hong-Lan Jiang[1,*] (姜红兰), *Member, CCF, IEEE*, Hai Mo[2] (莫　海)
Jie Han[3] (韩　杰), *Senior Member, IEEE*, Lei-Bo Liu[2] (刘雷波), *Senior Member, IEEE*, and
Zhi-Gang Mao[1] (毛志刚)

[1] *Department of Micro-Nano Electronics, Shanghai Jiao Tong University, Shanghai 200240, China*

[2] *School of Integrated Circuits, Tsinghua University, Beijing 100084, China*

[3] *Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 1H9, Canada*

E-mail: tongli_licu@sjtu.edu.cn; honglan@sjtu.edu.cn; moh19@mails.tsinghua.edu.cn; jhan8@ualberta.ca
liulb@tsinghua.edu.cn; maozhigang@sjtu.edu.cn

Received June 3, 2022; accepted March 23, 2023.

Abstract　As a primary computation unit, a processing element (PE) is key to the energy efficiency of a convolutional neural network (CNN) accelerator. Taking advantage of the inherent error tolerance of CNNs, approximate computing with high hardware efficiency has been considered for implementing the computation units of CNN accelerators. However, individual approximate designs such as multipliers and adders can only achieve limited accuracy and hardware improvements. In this paper, an approximate PE is dedicatedly devised for CNN accelerators by synergistically considering the data representation, multiplication and accumulation. An approximate data format is defined for the weights using stochastic rounding. This data format enables a simple implementation of multiplication by using small lookup tables, an adder and a shifter. Two approximate accumulators are further proposed for the product accumulation in the PE. Compared with the exact 8-bit fixed-point design, the proposed PE saves more than 29% and 20% in power-delay product for 3×3 and 5×5 sum of products, respectively. Also, compared with the PEs consisting of state-of-the-art approximate multipliers, the proposed design shows significantly smaller error bias with lower hardware overhead. Moreover, the application of the approximate PEs in CNN accelerators is analyzed by implementing a multi-task CNN for face detection and alignment. We conclude that 1) an approximate PE is more effective for face detection than for alignment, 2) an approximate PE with high statistically-measured accuracy does not necessarily result in good quality in face detection, and 3) properly increasing the number of PEs in a CNN accelerator can improve its power and energy efficiency.

Keywords　approximate computing, convolutional neural network (CNN), sum of products (SoP), data representation, multiplier

1　Introduction

Convolutional neural networks (CNNs) have successfully been applied to many applications, such as image recognition[1, 2], face detection and alignment[3, 4], video recognition[5], natural language processing[6], among others. Inspired by the working principle of animal visual cortex, a CNN uses a small number of neurons repetitively to respond to the restricted region of visual fields that are partially overlapped[7]. Compared with a multilayer perceptron (MLP) consisting of fully-connected layers, a CNN has a significantly simpler connectivity and lower computational complexity; thus, a CNN makes the processing of more complex tasks and datasets possible using general-purpose processors with a high performance and ac-

Regular Paper

Special Section on Approximate Computing Circuits and Systems

This work was supported in part by the National Natural Science Foundation of China under Grant No. 62104127 and the National Key Research and Development Program of China under Grant No. 2022YFB4500200.

*Corresponding Author

curacy[8]. For a better trade-off between energy efficiency and performance, domain-specific hardware has been investigated, e.g., tensor processing unit (TPU)[9]. In addition, various accelerators have been devised for different CNNs for pursuing a higher energy-efficiency and performance[10–14]. In a CNN accelerator, processing elements (PEs) are the basic units for computing the sum of products (SoP) for the convolution operation, which usually consist of multiplier arrays and accumulators (or multiply-and-accumulates (MACs)). Fig.1 lists the number of required MACs in the convolutional (CONV) layers and fully-connected (FC) layers in several popular CNNs[15]. It shows that the computation for CONV layers is becoming increasingly dominating, indicating that an efficient hardware implementation of the CONV layer is crucial to a CNN accelerator.

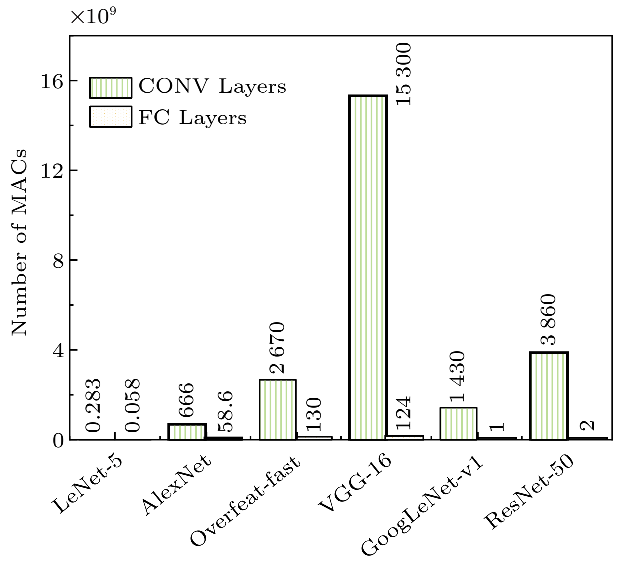


Fig.1. Required number of MACs in different CNNs[15]. The sizes of the input images for LeNet-5, AlexNet and Overfeat-fast are 28×28, 227×227 and 231×231, respectively. The input size for the other CNNs is 224×224.

Although neural networks (NNs) are commonly executed in floating-point format on general-purpose processors, a lot of work has been done on the quantization especially on CNN accelerators[16–19] to improve the computing efficiency. In these quantization-based designs, weights and activations are quantized into low precision fixed-point formats, such as 16-bit, 8-bit and 4-bit, with a comparable accuracy to their 32-bit floating-point counterparts. As fixed-point multiplication and addition consume fewer hardware resources than their corresponding floating-point designs, quantization can drastically reduce the area and power consumption of a PE. Due to the existence of power walls[20, 21], the smaller area and the less power a single PE consumes, the more computational resources can be integrated on a single NN chip. Thus, higher performance and energy efficiency can be achieved. In addition, the memory size and accesses can also be reduced with bit-wise operations after quantization.

As a hardware-efficient computing paradigm, approximate computing has widely been used in various error-tolerant applications including machine learning[22, 23]. As a CNN is often used in human perception related tasks (e.g., image recognition and face detection) with a noisy input dataset, it can tolerate a certain level of errors[24]. Moreover, many learning algorithms in CNNs are probability-based, and do not require exact computation results. Therefore, to achieve improvements in performance, power and energy efficiency, approximation techniques can be applied to the implementation of CNN accelerators at the cost of acceptable quality loss.

Compared with adders, multipliers with relatively high complexity are more frequently considered for approximation in a deep neural network (DNN). Approximate multipliers have been applied to the implementations of MLPs and CNNs[25, 26], which results in significant reductions in delay, area and power consumption without accuracy loss. However, the PEs in these designs are not optimized with respect to a specific type of applications and their exact floating-point or fixed-point multipliers are simply replaced by existing approximate multiplier designs. [27] has shown that some combinations of approximate multipliers and adders can result in a higher accuracy of DNN than accurate designs. Moreover, 16-bit fixed-point multipliers are utilized in [25], which is very conservative for the inference of an NN[28]. A 16-bit multiplier has a larger approximation space than an 8-bit multiplier that is more commonly used for inference. Although 8-bit approximate multipliers are exploited in [26], the implemented NNs (MLP and LeNet-5) and tested datasets (MNIST and SVHN) are relatively simple. Therefore, it is not yet clear about how approximate computing benefits a CNN accelerator, what kind of application scenarios approximation can be applied to, or what level of approximation a CNN accelerator can accept for a reasonable accuracy.

In this paper, an approximate PE is proposed specifically for CNN accelerators considering both multiplication and addition. To simplify the multiplication in a PE, an approximate data format is first defined for the weights, where a stochastic rounding is used to improve the accuracy of applications. A multiplication is then easily implemented by small lookup tables (LUTs), an adder and a shifter. In addition, an approximate adder tree constructed by approximate carry-propagate adders (CPAs) and an approximate Wallace tree consisting of full adders and an approximate CPA are designed for the product accumulation in the PE. The simulation results show that the proposed approximate PE achieves more than 29% and 20% reductions in power-delay product (PDP) compared with the exact 8-bit designs for 3×3 and 5×5 sum of products, respectively. Last but not least, the role of approximate PEs acting in a CNN accelerator is analyzed by using a multi-task CNN (MTCNN) for face detection and alignment, i.e., applying approximate PEs with different configurations in the implementation of the MTCNN. Three important conclusions are drawn to guide the application of approximate computing in CNN accelerators.

The paper is organized as follows. Section 2 introduces the basic architecture of a CNN and the computation of its CONV layer, and the approximation techniques for arithmetic circuits. An approximate PE is then proposed for the SoP operation in CONV layers in Section 3. Section 4 evaluates the approximate PE in terms of accuracy and circuit measurements. Also, several state-of-the-art approximate multipliers are considered for comparison. In Section 5, the approximate PEs are further assessed in a multi-task CNN. Finally, the paper is concluded in Section 6.

2 Background

2.1 CNN Accelerators

Fig.2 shows an example of CNN architecture consisting of an input layer, three CONV layers, a pooling layer, and an FC layer as the output layer. CNN is proposed based on three basic architectural ideas, which are local receptive fields, shared weights and spatial (or temporal) sub-sampling[1]. The input of a CNN is usually a set of 2-D feature maps (e.g., pixels of images), where each 2-D map is referred to as a channel. A CONV layer performs the convolution of an input channel (or an output feature map from its previous layer) and a 2-D filter, i.e., repetitively computing the sum of products (SoP) of the 2-D filter and a same size of input that is denoted as a receptive field. To introduce nonlinearity into CNN, a nonlinear activation function (e.g., sigmoid and rectified linear unit) is then applied to the outputs of a convolution result. A pooling layer following a CONV layer is used to sub-sample the output of a CONV layer by keeping the average or maximal value of its receptive filed. In addition, several FC layers are often added to the end of a CNN for generating output information.

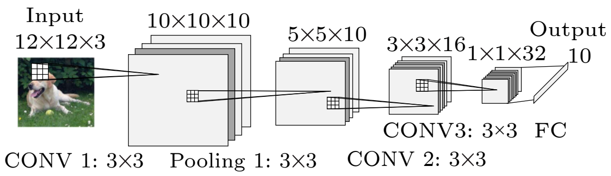


Fig.2. Example of CNN architecture.

As CONV layers dominate the computation of a CNN, they are specifically focused in this paper. The computation structure of a CONV layer is shown in Fig.3. In this computation, each input channel corresponds to a 2-D filter with a size of $W_{\rm f} \times H_{\rm f}$, while all channels share one bias. Thus, the number of 2-D filters for each output feature map is equal to the number of the input channels ($N_{\rm in}$), resulting in a 3-D filter. The number of 3-D filters is equal to the number of the output feature maps ($N_{\rm out}$), as shown in Fig.3. The output of a CONV layer is computed as

$$O(u)(x)(y) = \sum_{k=0}^{N_{\rm in}-1} \sum_{i=0}^{W_{\rm f}-1} \sum_{j=0}^{H_{\rm f}-1} W(u)(k)(i)(j) I(k)(x+i)(y+j) + B(u), \quad (1)$$

where $0 \leqslant x \leqslant W_{\rm out}-1$, $0 \leqslant y \leqslant H_{\rm out}-1$, and $0 \leqslant u \leqslant N_{\rm out}-1$. $W_{\rm out}$ and $H_{\rm out}$ are the width and the height of an output feature map $\boldsymbol{O}(u)$, respectively. $\boldsymbol{W}$ consists of $N_{\rm out}$ 3-D filters. $\boldsymbol{I}$ contains $N_{\rm in}$ input feature maps with a size of $W_{\rm in} \times H_{\rm in}$. $\boldsymbol{B}$ is the bias vector. The size of each output feature map depends on the sizes of the input feature map and the 2-D filter, i.e., $W_{\rm out} = W_{\rm in} - W_{\rm f} + 1$ and $H_{\rm out} = H_{\rm in} - H_{\rm f} + 1$. Generally, the 2-D filter is very small. The commonly-used sizes are 3×3 and 5×5.

As depicted in (1), the basic computation operation in a CONV layer is a $W_{\rm f} \times H_{\rm f}$ SoP, $P_{\rm sum} = \sum_{i=0}^{W_{\rm f}-1} \sum_{j=0}^{H_{\rm f}-1} W(u)(k)(i)(j) I(k)(x+i)(y+j)$. Thus, a PE in a CNN accelerator is usually designed to implement a fixed size of SoP, consisting of a multiplier array and an adder tree[25, 29, 30]. Consequently, $W_{\rm f} \times H_{\rm f}$ multipliers and $W_{\rm f} \times H_{\rm f} - 1$ adders are required in a conventional PE. The critical path delay and circuit area of the PE are given by

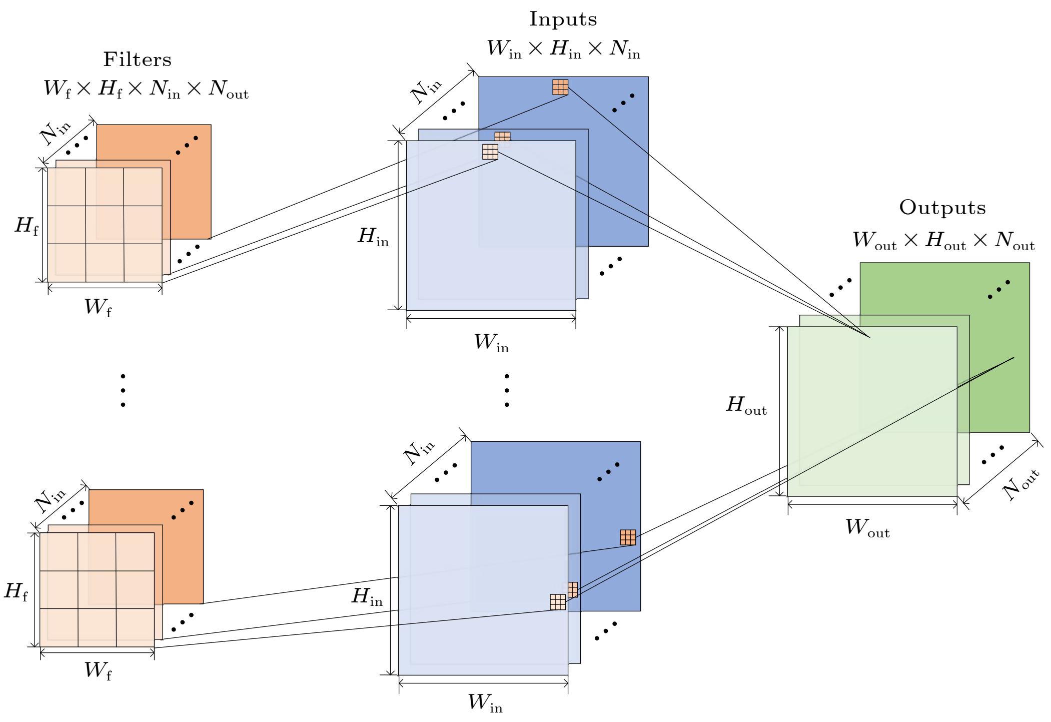

Filters
$W_f \times H_f \times N_{in} \times N_{out}$
Inputs
$W_{in} \times H_{in} \times N_{in}$
Outputs
$W_{out} \times H_{out} \times N_{out}$
N_{in}
H_f
W_f
H_{in}
W_{in}
H_{out}
W_{out}
N_{out}

Fig.3. Computation architecture of a CONV layer.

$$t_{PE} = t_{mul} + \lceil \log_2(W_f \times H_f) \rceil t_{add},$$

and

$$S_{PE} = W_f \times H_f \times S_{mul} + (W_f \times H_f - 1)S_{add},$$

where t_{mul} and t_{add} are the critical path delay of the multiplier and CPA, respectively. S_{mul} and S_{add} are the circuit area of the multiplier and CPA, respectively. Note that the CONV layers are usually implemented as matrix multiplications in general-purpose processors by reshaping the inputs.

As the floating-point representation for the data in CNN accelerators significantly limits the performance and energy efficiency, a lot of effort has been made on the simplification of data representation. Some designs directly reduce the bit width of data and are usually followed by a retraining operation. Also, some studies modify the standard floating-point representation format. [31] proposes a new data representation format denoted as BISCALED-FXP, which caters to the disparate range and resolution requirements of long-tailed data distributions. While reducing the bit width of data, the same classification accuracy can be remained. This design leads to 1.43x–3.86x and 1.4x–3.7x improvements in performance and energy efficiency, respectively. However, the SoP operations still require many hardware resources due to the structural limitations of floating-point adders. To solve this problem, quantization is commonly utilized in the inference of CNNs, which represents floating-point numbers in fixed-point format; thus, fixed-point computing elements with low hardware consumption are sufficient. Numerous quantization methodologies have been developed to ensure a high accuracy for CNNs[32].

As most quantized CNN accelerators are designed for inference, 8-bit[30] or 16-bit[25, 29, 33] fixed-point multipliers and 16-bit or larger fixed-point adders are commonly used in a PE. Compared with using floating-point arithmetic circuits, using small fixed-point designs in a PE can significantly reduce its critical path, area and power consumption. However, without sufficiently considering the characteristics of the CONV layer, a straightforward combination of basic computing elements is not very efficient. Therefore, a PE is proposed in this paper, specifically for the CONV layer, taking advantage of the CNN characteristics and approximation techniques[32].

2.2 Approximate Arithmetic Circuits

Considering that a large number of applications exhibit intrinsic error tolerance, massive research has been conducted in the field of approximate computing to achieve hardware improvements. In approximate computing, errors have been viewed as a commodity that can be traded for significant gains in hardware-efficiency. As a result, it has been comprehensively investigated in compute-intensive applications with error tolerance such as DNNs, for higher performance- and energy-efficiency[12, 13]. As MACs dominate the computing hardware of a CNN accelerator, a multiplier is usually approximated due to its relatively high complexity and error tolerance compared with an adder[27]. Thus, the commonly-used structures of approximate multipliers and several state-of-the-art designs are introduced next.

Considering an 8×8 unsigned fixed-point multiplier with the inputs $A = A_7 ... A_0$ $(A = \sum_{i=0}^{i=7} A_i 2^i)$ and $B = B_7 ... B_0$ $(B = \sum_{i=0}^{i=7} B_i 2^i)$, the product $P = A \times B$ can be given by

$$P = \sum_{i=0}^{i=7} \sum_{j=0}^{j=7} (A_i B_j) 2^{i+j}.$$

The corresponding classical unsigned fixed-point multiplier is shown in Fig.4, and it consists of a partial product generation (PPG) unit, a partial product reduction (PPR) tree, and a final CPA. The PPG unit ANDs each multiplier bit with all the multiplicand bits with a radix (power of 2). Booth encoding is also commonly used in the PPG to reduce the number of partial products, thus reducing the circuit area. For a fast accumulation, the reduction tree conducts parallel compression to collapse the partial products into two operands, which are fed to the final adder for generating the final product.

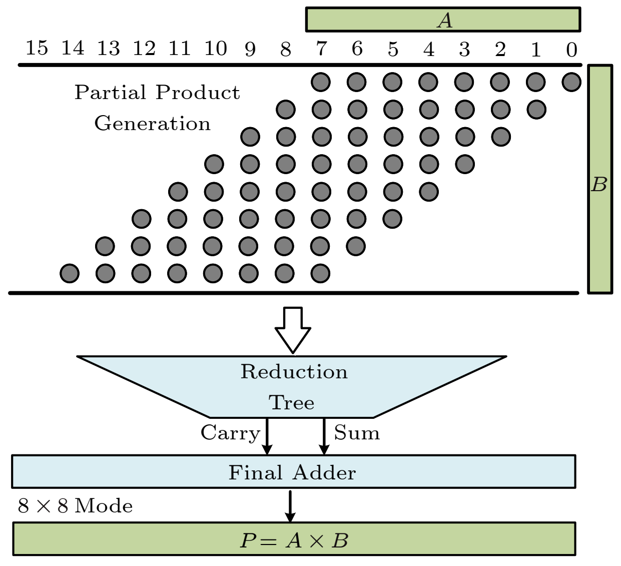


Fig.4. A classical 8×8 unsigned fixed-point multiplier.

Many schemes have been proposed to approximate a multiplier at the stages of PPG and PPR. For the approximation in PPG, some bits of the input operands or the partial product array are truncated or perforated; thus, a smaller multiplier or PPR tree is sufficient for processing the remaining bits compared with conventional designs. Example designs include the truncated multiplier, the partial product perforation-based multiplier[34], the error-tolerant multiplier[35], and the dynamic range unbiased multiplier (DRUM)[36]. The omitted and remaining bits can be either statically or dynamically selected according to the truncation scheme. The latter is more accurate at the cost of a more complex circuit for dynamically selecting the input operands and post processing the intermediate results. Moreover, dynamic selection can result in unbiased errors; thus, it is suitable for accumulative operations[27]. To achieve a better accuracy-hardware tradeoff, error compensation circuits can be formulated through probabilistic analysis of the truncated partial products. For instance, the approximate Booth multipliers in [37, 38] are designed based on the approximation in PPG and error compensation.

The approximation at the PPR stage focuses more on the circuit itself. This type of approximation approaches commonly relies on logic simplification, which simplifies the compression elements through the Karnaugh map or other circuit related methods. As an efficient compression element, 4-2 compressor is commonly approximated to substitute some less significant exact compressors. For instance, approximate multipliers in [39–41] are devised based on approximate 4-2 compressors. [40] proposes five architectures of high-accuracy approximate 4-2 compressors with shorter critical paths by decomposition and recombination methods which are then used to construct multipliers with a better tradeoff between accuracy and hardware. To lower the error probability, the partial products with different probabilities can be obtained by different encoding schemes. The compressors in the PPR can then be simplified and reorganized as per the encoding results[42].

Compared with the above approximation methods at the circuit level, approximating at the algorithmic level can achieve more benefits in hardware. As the first attempt, the Mitchell algorithm uses simple operations to approximate the logarithmic and anti-logarithmic computations of binary numbers[43]. Thus, an adder is utilized to do the multiplication. This multiplier is the basis of the logarithmic multipliers, denoted as LM-ORG in this paper. To improve the accuracy of LM-ORG, a truncated binary-logarithm converter and a set-one-adder (SOA) are used in

ALM-SOA[44]. In the SOA, some least significant bits of the addition result are set to constant "1"s and an AND gate is used to generate a carry-in for the accurate sub-adder processing the higher bits. Moreover, [45] improves the Mitchell algorithm by mitigating its single-sided errors. Two approximate multipliers are achieved by using the exact (ILM-EA) and approximate adders (ILM-AA), respectively. In general, the logarithmic approximation based multipliers (LMs) are very hardware-efficient for the applications requiring large bit width yet with large errors.

As different approximation approaches result in various error and circuit characteristics, proper methods should be selected as per the application scenarios and requirements. In the quantized CNN accelerators, since the multiplier of small bit width is generally used, the approximation multiplier with a better optimization in the computation of small inputs would result in a high accuracy. In addition, the cumulative effect of bias on errors is catastrophic for CNNs and needs to be considered with particular care[27].

3 Design of Approximate PE

To design an approximate PE with a good trade-off between accuracy and hardware overhead, data representation, multiplication, and accumulation are comprehensively studied.

3.1 Approximate Data Representation Format

In a CNN accelerator for inference, the trained weights and input data are often stored in an offchip or onchip memory. Thus, the storage precision and format are predetermined and can be adapted for an efficient computing. To benefit from this observation, we propose to define an approximate data format for the weights, aiming to facilitate the design of approximate multiplications for quantized CNNs.

Floating-point formats such as single precision and half precision are usually used for a computation that requires a wide dynamic range and/or a high precision. A floating-point format consists of a sign (1-bit), an exponent (8-bit for single precision and 5-bit for half precision) and a mantissa (23-bit for single precision and 10-bit for half precision). In a floating-point format, the width of the exponent determines the dynamic range, and the width of the mantissa influences the precision (or resolution). As discussed in Section 1, a CNN is error-tolerant due to its noisy inputs, human perception related application scenarios, and learning algorithms. Thus, a wide dynamic range is more important than a high precision for the computations in a CNN accelerator. To this end, Google has proposed a new floating-point format with 8-bit exponent and 7-bit mantissa to enable the capability of training in TPUv2[46]. Although this representation significantly improves the energy efficiency, floating-point computing elements with high complexity are necessary. To avoid using floating-point computations while maintaining a wide representation range in the quantized CNNs, an approximate format with tunable dynamic range is defined for approximately representing the weights. In this format, the bit width of weights can be decreased, and the mantissa width can be adjusted as needed for precision.

In general, the weights in CNNs exhibit a long tail distribution, i.e., a significant majority of the weights have small magnitude, whereas a small fraction of them are orders of magnitude larger. This is consistent with the characteristic of floating-point representation, where smaller numbers are more densely distributed with higher resolution and larger numbers are sparsely represented with lower resolution, as depicted in Fig.5. Thus, similar to the floating-point representation, the value of W in the newly defined approximate format is given by

$$W = (-1)^{sign} \times (1 + \frac{mantissa}{2^m}) \times 2^{exponent-bias},$$

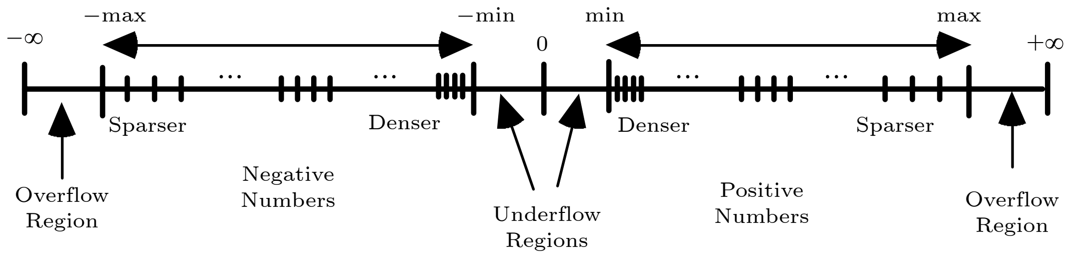


Fig.5. Floating-point number representation.

where *sign*, *exponent* and *mantissa* are 1-bit sign, e-bit exponent and m-bit mantissa, respectively. *bias* is the exponent bias that is determined by the range of the weight set. Also, the values of e and m vary with the required dynamic range and precision respectively. Zero is indicated when all bits in the exponent and mantissa are 0.

To represent an n-bit fixed-point number with 1 sign bit, n_{int} integer bits and n_{frac} fractional bits in the approximate format, e and *bias* should be determined to guarantee the representation range. As the minimum possible non-zero magnitudes that the proposed and fixed-point formats can be represented are 2^{1-bias} and $2^{-n_{\text{frac}}}$ respectively, *bias* should be equal to $n_{\text{frac}}+1$. Also, $\max(exponent) - bias + 1 = \log_2(\max(|weight|))$, where $\max(exponent)$ is the maximum value of $exponent$, i.e., $\max(exponent) = 2^e - 1$. $\max(|weight|)$ is the maximum absolute value of the number to be approximated ($weight$), which is approximately $2^{n_{\text{int}}}$ in a fixed-point format. Thus, $e = \log_2(n_{\text{int}} + bias) = \log_2(n_{\text{int}} + n_{\text{frac}} + 1) = \log_2(n)$. Consequently, e and *bias* of the approximate format are given by $e = \lceil \log_2(n) \rceil$ and $bias = n_{\text{frac}} + 1$, respectively. The NaN (not a number) and infinite number in IEEE-754 standard are discarded and replaced by more valid weight values, which increases the dynamic range and resolution that can be represented. Also, it is unnecessary to support subnormal numbers but to increase the valid values of the minimum scale, and the corresponding hardware can be simplified.

In the approximate format, the width of the mantissa is reduced to the most extent to simplify the implementation of a multiplication. Meanwhile, to ensure a high accuracy for the entire computation, a stochastic rounding is exploited for generating the mantissa. Fig.6 shows an example of transforming an 8-bit fixed-point number to the approximate format, where the 8-bit fixed-point number ($weight = (w_7 w_6.w_5 \ldots w_0)_2$) contains 1 sign bit, 1 integer bit ($n_{\text{int}} = 1$) and 6 fractional bits ($n_{\text{frac}} = 6$). Thus, $e = 3$ and $bias = 7$ for the approximate format. The width of the mantissa m is set to 2 in this example. In this case, $sign = w_7$, $exponent = \lfloor \log_2 |weight| \rfloor + bias = 6$, and $mantissa = (01)_2 + r_s$, where r_s is a binary bit for stochastic rounding. Specifically, r_s is generated stochastically by comparing the ignored bits $(w_2 w_1 w_0)_2$ with a random (or pseudorandom) number S that is uniformly distributed. To keep a high accuracy, the bit width of S should be equal to or larger than $(n-m-2)$, and "0"s should be appended to the

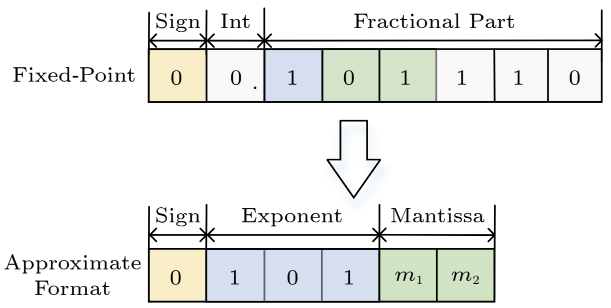


Fig.6. Transformation from an 8-bit fixed-point number to the proposed approximate format.

least significant bit positions if the number of ignored bits is smaller than $(n-m-2)$. Then, $r_s =$ "1" if $S \leqslant (w_2 w_1 w_0 0)_2$; otherwise $r_s =$ "0". This ensures that the probability of r_s being "1" is close to $(w_2 w_1 w_0 0)_2 / 2^4$[47].

3.2 Multiplier Design

Representing both the input operands of the multiplication in approximate format, the multiplication can be easily implemented by a reduced-width multiplier and an adder. However, the accumulation for the numbers in approximate format requires exponent alignment and normalization, which results in a higher complexity compared with that in fixed-point format. Moreover, the stochastic rounding process makes the situation worse. Thus, in this design, the weights of a CNN are stored in the proposed approximate format, whereas the input data and output results of each layer are represented in standard fixed-point format. The weights are converted to the approximate format offchip; hence, no hardware overhead is increased for the format conversion. In this case, the multiplication is simplified while the workload for the accumulation is not increased.

As a result, the multiplication can be completed by a reduced-width multiplier and a shifter. The carry-save array and Wallace tree are the two basic structures for implementing the PPR of a multiplier. Also, an LUT-based multiplier can be very efficient for a small size. Conventional LUT-based computation relies on memory devices to save the fixed set of multiplication results. Here, we do not need to use SRAM or register files because the LUT stores fixed values and can be shared among different units. Small LUTs can be implemented with pure combinational logic leveraging power and ground as "1"s and "0"s respectively, thus consuming less area and delay. To perform an $m \times n$ multiplication, an LUT with a size of 2^{m+n} is required, i.e., the size of LUT increases exponentially with the size of the multiplication. Thus, the efficiency of an LUT-based multiplier depends on its size. To figure out the proper sizes for a multiplier that can be efficiently implemented by using LUTs, different sizes of multipliers implemented using LUT-based and array structures are evaluated. Fig.7 shows the hardware comparison of the multipliers with different structures in critical path delay, power dissipation, area and power-delay product (PDP). As per Fig.7, an LUT-based multiplier has a smaller or similar delay compared with an array multiplier for the same size. However, the area and the power dissipation of an LUT-based multiplier increase more rapidly with the size than those of an array multiplier. Considering the PDP, an LUT-based multiplier is more efficient than an array multiplier when the size is smaller than 3×5. As the mantissa width of the weights represented in approximate format is very small, the required multiplier for the PE can be implemented by several small sub-multipliers. Therefore, LUTs are utilized in this design for implementing sub-multipliers to lower the critical path delay and power dissipation.

Fig.8 shows the implementation structure for the multiplication of weight W in the approximate format and data D in the fixed-point format, where the bit widths for both of the two operands are 6-bit. The multiplication of $W = (-1)^{S_w} 2^{E_w - bias}(1 + M_w 2^{-2})$ and $D = (-1)^{d_5} 2^1 + \sum_{i=0}^{4} d_i 2^{i-4}$ is computed as

$$P_{\text{out}} = W \times D = (-1)^{S_w} 2^{E_w - bias - 6} M_{\text{int}} \times D_{\text{int}}, \quad (2)$$

where $S_w = w_5$, $E_w = (w_4 w_3 w_2)_2$ and $M_w = (w_1 w_0)_2$ are the sign, exponent and mantissa of W, respectively. $M_{\text{int}} = (1 w_1 w_0)_2 = 2^2 + \sum_{i=0}^{1} w_i 2^i$ is an unsigned integer, and $D_{\text{int}} = (-1)^{d_5} 2^5 + \sum_{i=0}^{4} d_i 2^i$ is a signed in-

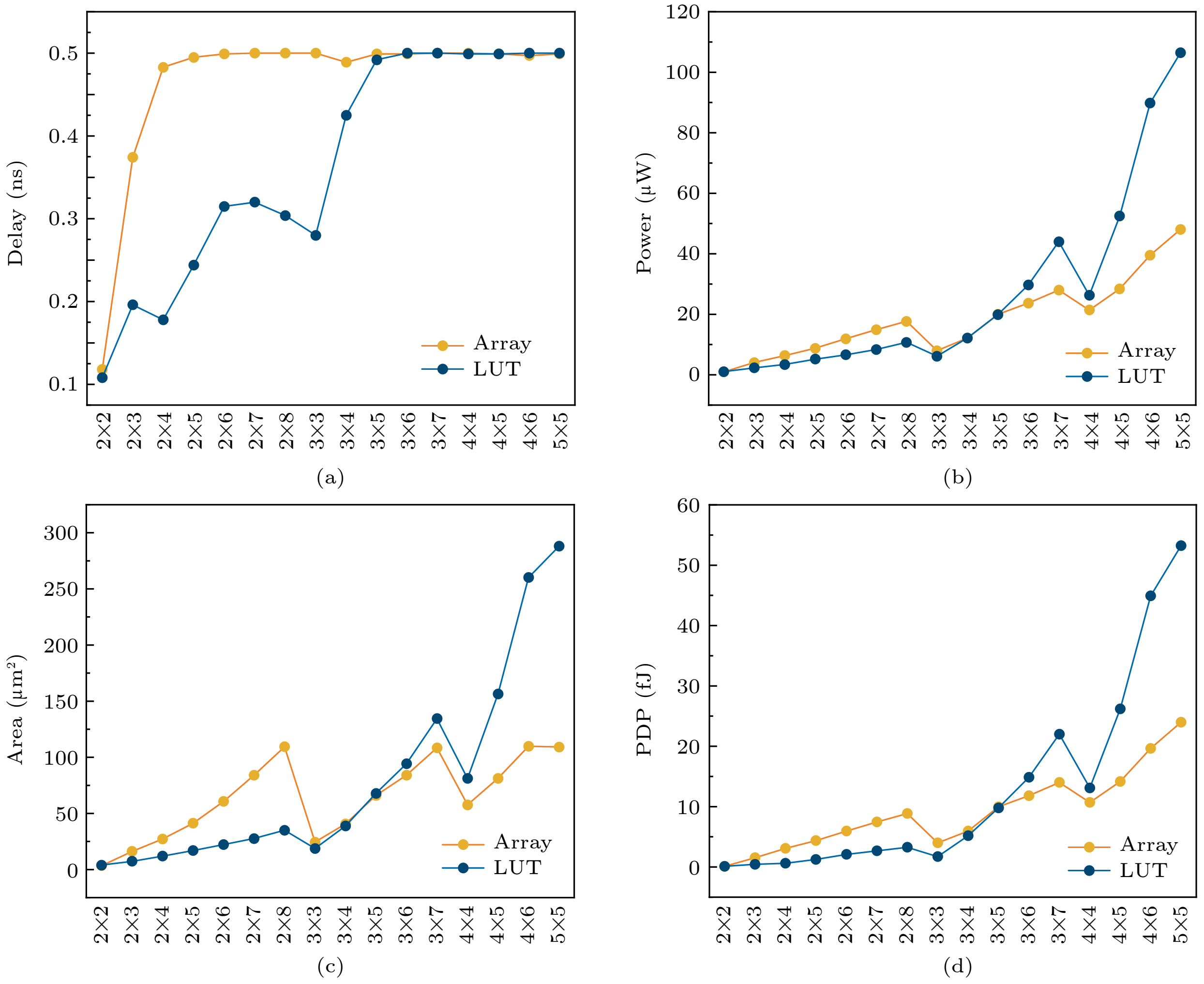


Fig.7. Circuit characteristic comparison of LUT-based and array multipliers. (The clock frequency for the syntheses is 2 GHz.) (a) Delay. (b) Power. (c) Area. (d) PDP.

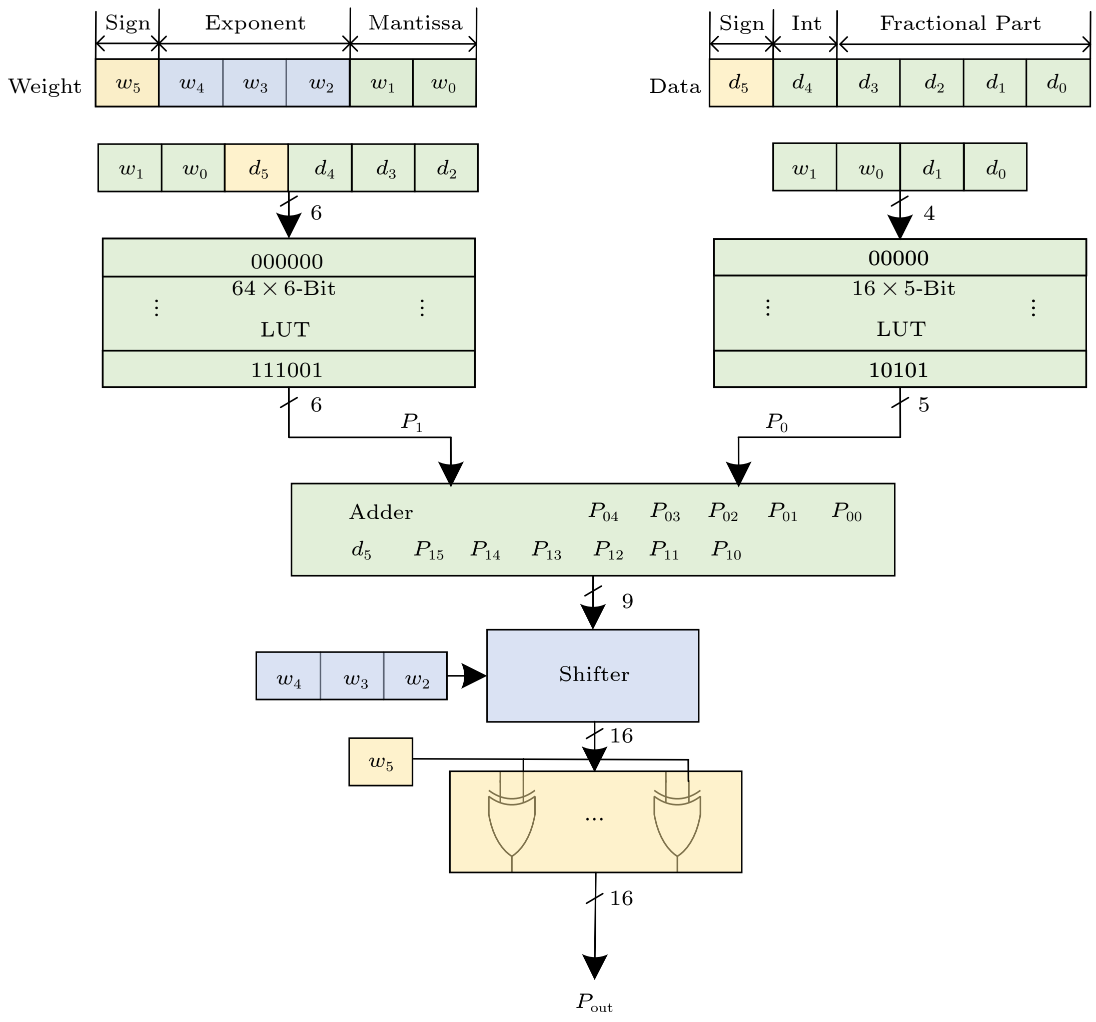

Sign
Exponent
Mantissa
Weight
w_5 w_4 w_3 w_2 w_1 w_0
Sign
Int
Fractional Part
Data
d_5 d_4 d_3 d_2 d_1 d_0
w_1 w_0 d_5 d_4 d_3 d_2
6
000000
64×6-Bit
LUT
111001
6
P_1
w_1 w_0 d_1 d_0
4
00000
16×5-Bit
LUT
10101
5
P_0
Adder
P_{04} P_{03} P_{02} P_{01} P_{00}
d_5 P_{15} P_{14} P_{13} P_{12} P_{11} P_{10}
9
w_4 w_3 w_2
Shifter
16
w_5
16
P_{out}

Fig.8. Proposed multiplication structure. P_{0i} and P_{1i} are the i-th ($i = 0, 1, \cdots$) partial product bits of the partial products P_0 and P_1, respectively.

teger in 2's complement. As per (2), this multiplication can be implemented by using a 2×6 multiplier ($M_{\text{int}} \times D_{\text{int}}$), a shifter and a sign processing. To better exploit the advantages of LUTs in small bit width multiplications, the 2×6 multiplier is further divided into two small LUT-based sub-multipliers.

To determine the sizes of the two LUTs, the circuit balance in the data path should be considered. Imbalance data paths within the circuit can cause many spurious activities and glitches[48]. Also, the imbalance can propagate to the product accumulation stage of the PE, which results in a lot of bootless power consumption. The higher bits of the adder need to wait for the carries from the lower bits. If the outputs from the left LUT in Fig.8 arrive before the carries, the flips are spurious activities and propagate downwards. Therefore, a larger LUT for generating the higher bits and a smaller LUT for the lower bits would help balance the inputs inside the adder, thus reducing the power consumption. It should be noted that the circuit balance cannot completely eliminate all spurious activities, but reducing imbalance can save power. Consequently, the two LUTs implement $(1w_1w_0)_2 \times (d_5d_4d_3d_2)_2$ and $(1w_1w_0)_2 \times (d_1d_0)_2$. As $(1w_1w_0)_2$ contains a constant "1", the sizes of the required LUTs are $2^{2+4} = 64$ and $2^{2+2} = 16$, respectively. Specifically, $P_0 = (1w_1w_0)_2 \times (d_1d_0)_2$ is an unsigned multiplication, whereas $P_1 = (1w_1w_0)_2 \times (d_5d_4d_3d_2)_2$ is a multiplication of an unsigned number and a signed number. Thus, the multiplication results stored in the two LUTs are different, as shown in Fig.8. As the multiplication result P_1 has the same sign as the input data, only 6-bit results are stored in the LUT. Then, a simple adder is used to add the two partial products P_0 and P_1, resulting in a 9-bit output. By using a shifter, the addition result is right or left shifted by $|E_w - bias - 6|$ bits. Finally, P_{out} is obtained after a sign processing by using a 1's complement operation, i.e., P_{out} is the inverted output of the shifter if $w_5 =$ "1"; otherwise it is the output of the shifter. Note that the product result is in fixed-point format.

Table 1 shows the comparison of circuit measurements for multipliers implemented by using different decomposition schemes. It shows that the multiplier based on two LUTs is superior to the one implemented by using a single LUT. Compared with a 2×6 array multiplier, the proposed two LUT-based designs are better in delay, area and PDP. For a larger multiplication, the array multiplier outperforms the LUT-based designs. Thus, the array multiplier should be utilized for the proposed multiplication when a larger size is required.

Table 1. Circuit Measurements of the Multipliers Using Different Design Schemes

Size	Design	Power (μW)	Delay (ns)	Area (μm^2)	PDP (fJ)
2×6	Array	4.15	0.36	47.12	1.494
	LUT	7.72	0.30	58.46	2.316
	2 LUTs	4.23	0.25	25.57	1.058
3×6	Array	9.81	0.41	63.25	4.022
	LUT	12.97	0.50	233.20	6.485
	2 LUTs	11.64	0.42	69.67	4.889

3.3 Accumulator Design

An adder tree (AT) constructed by $W_{\text{f}} \times H_{\text{f}} - 1$ CPAs is commonly used to accumulate the products in a $W_{\text{f}} \times H_{\text{f}}$ SoP. In a CNN, the output of one layer is the input of the next layer; the output of a PE should be rounded to a fixed width. Thus, the lower part of a PE result is less important and can be approximated. To lower the hardware overhead, an approximate adder with the approximate lower part can be used in an AT. In the proposed design, the lower-part-OR (LOA) adder with close to zero mean error is exploited to guarantee a high accuracy[49]. In an LOA, k lower bits of an adder are added by using k OR gates, and an AND gate is used to generate a carry-in for the exact adder that processes the higher bits[50]. The accuracy of an LOA varies with k, so does the AT using LOAs. The critical path delay and area of the LOA-based AT are

$$t_{\text{AT}} = \lceil \log_2(W_{\text{f}} \times H_{\text{f}}) \rceil t_{\text{LOA}},$$

and

$$S_{\text{AT}} = (W_{\text{f}} \times H_{\text{f}} - 1) S_{\text{LOA}},$$

where t_{LOA} and S_{LOA} are the critical path delay and area of the utilized LOA respectively.

In addition, a Wallace tree (WT) can be used for a fast accumulation. As shown in Fig.9, a WT consists of full adders (FAs) and a CPA, where the FAs in each stage work in parallel. The critical path delay and area of a WT with $W_{\text{f}} \times H_{\text{f}}$ inputs are given by

$$t_{\text{WT}} = \lfloor \log_{1.5}(W_{\text{f}} \times H_{\text{f}}) \rfloor t_{\text{FA}} + t_{\text{add}},$$

and

$$S_{\text{WT}} = n(W_{\text{f}} \times H_{\text{f}} - 2) S_{\text{FA}} + S_{\text{add}},$$

where t_{FA} and S_{FA} are the critical path delay and area of a FA, respectively, and n is the bit width of the input operands. Compared with the conventional AT, a WT for the same number of inputs has a shorter critical path[51]. To further reduce the hardware overhead, an LOA is utilized for the final addition of the WT as shown in Fig.9.

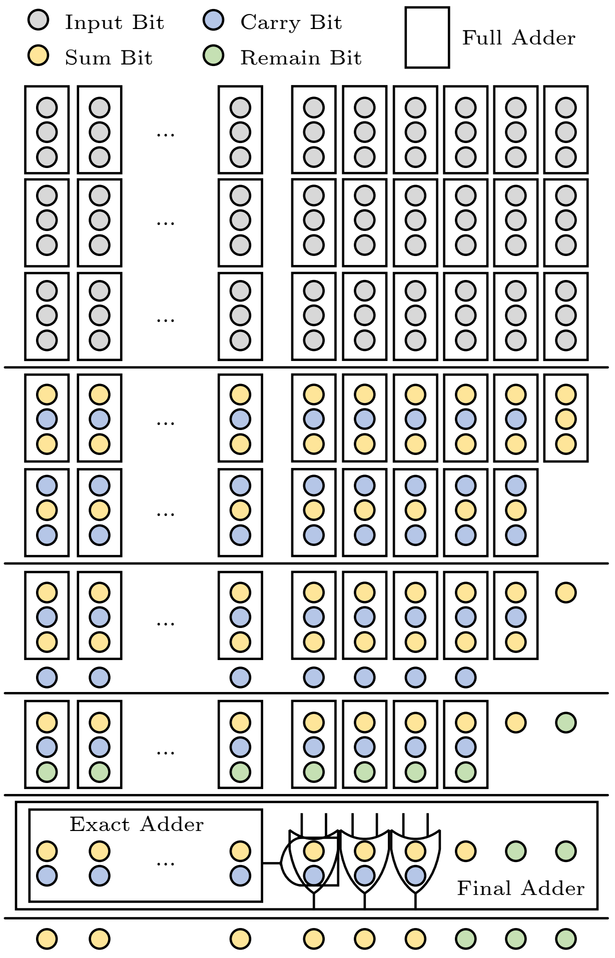


Fig.9. A 9-input Wallace tree consisting of full adders and a multi-bit adder.

4 Evaluation of PEs

This section evaluates the merits of the proposed approximate PEs. Based on the approximate data format, multiplier, and accumulator, this paper proposes two PE structures. For the purpose of comparison, one exact and four approximate PEs consisting of state-of-the-art approximate 8×8 multipliers are constructed. While the exact WT is utilized, the multiplier varies from the exact design to an existing approximate design. The considered multipliers are denoted as Exact, Prop, 4-2com1, 4-2com2, Truc-Booth, DRUM-k5, DRUM-k6, ALM-ORG, ALM-SOA3, and ILM-EA, which are specified as follows:

• Exact: the exact multiplier considered here is implemented based on the Booth encoding; the partial products are accumulated by using exact WT;

• Prop: the proposed LUT-based multiplier;

• 4-2com1: the multiplier using the approximate 4-2 compressor in [40] for the PPR;

• 4-2com2: the multiplier using the approximate 4-2 compressor in [39] for the PPR;

• Truc-Booth: the approximate truncated Booth multiplier proposed in [37];

• DRUM-k5: the DRUM[36] using a 5×5 accurate multiplier;

• DRUM-k6: the DRUM[36] using a 6×6 accurate multiplier;

• ALM-ORG: the LM using the Mitchell algorithm[43];

• ALM-SOA3: the ALM-SOA using three approximate LSBs in the SOA[44];

• ILM-EA: the LM using an improved Mitchell algorithm and an accurate adder[45].

As 3×3 and 5×5 filters are widely used in CNNs, the PEs with sizes of 3×3 and 5×5 are implemented and evaluated in terms of error and circuit characteristics. Fig.10 summarizes the delay, area and PDP comparison of different PEs with respect to MED, AVE and MRED. AVE is the average value of the errors, which indicates the error bias of an approximate arithmetic circuit. MED is the mean of the absolute errors, and MRED is the mean of the relative error distance[27]. In this simulation, n, n_{int} and m are set to 8, 1 and 2, respectively. The error metrics are achieved by Monte Carlo simulations with 10 million random input combinations. The circuits are implemented in Verilog and synthesized in TSMC 28 nm technology with a supply voltage of 0.81 V using Synopsys Design Compiler. For the syntheses, all designs are imposed on the minimum area constraints, under a clock frequency of 500 MHz.

As shown in Fig.10, the accumulation of more multiplication results generally leads to larger errors. For a specific structure of PE, the values of MED and MRED increase with their sizes except for the MRED of DRUM. In terms of AVE, all the designs are concentrated around 0 except for 4-2com2 and Truc-Booth. Although the AVE of 4-2com1 is not very large, it is still more deviated from 0. The proposed design has significantly small AVEs, which is consistent with our expectation for using stochastic rounding. DRUM-k6 and ILM-EA also have very small AVEs due to their unbiased approximation schemes. It is worth noting that the AVE of ALM-ORG with single-sided errors in the unsigned design is very close to 0. This is because the single-sided errors turn to be symmetric in the signed design. Although ILM-EA uses an error compensation scheme to ensure unbiased errors, its AVE is larger than that of ALM-ORG because the errors in the signed ILM-EA are not strictly symmetric. The error biases can be accumulated in the CNNs, leading to non-convergence[27].

Figs.10(a)–10(c) demonstrate the relationship between delay and accuracy for the considered PEs. The delay increases as MED and MRED decrease, indicating a compromise between accuracy and performance. It can also be inferred that the PEs are more evenly optimized for the speed at a larger size under the condition that the minimum area is the synthesis constraint. The 5×5 PEs have smoother delay variations compared with the 3×3 PEs, and their delays are all close to the Exact PE. The PEs with smaller values of MED and MRED generally have larger areas, as shown in Figs.10(d)–10(f). Our proposed design has the smallest delay and area compared with the other PEs except for Truc-Booth with substantial errors. Although LMs are hardware-efficient in the unsigned multiplication especially for a large bit-width, they cannot beat the other approximate multipliers due to their extra consumption in the data conversion from unsigned into 2's complement. Figs.10(g)–10(i) illustrate that approximate PEs exhibit a similar PDP trend versus both MED and MRED. Except for the Truc-Booth, our proposed approximate PE has the smallest PDP. On average, the approximate 3×3 and 5×5 PEs achieve about 29% and 20% reductions in PDP compared with the Exact design.

To assess the accumulator, Fig.11 reports the circuit and error characteristics of the proposed PEs consisting of proposed approximate multipliers and accumulators with different parameters. The WT-0 and AT-0 are the exact WT and AT accumulators, respectively; thus, they have the same error results. For a same k, the PEs using AT show larger values of

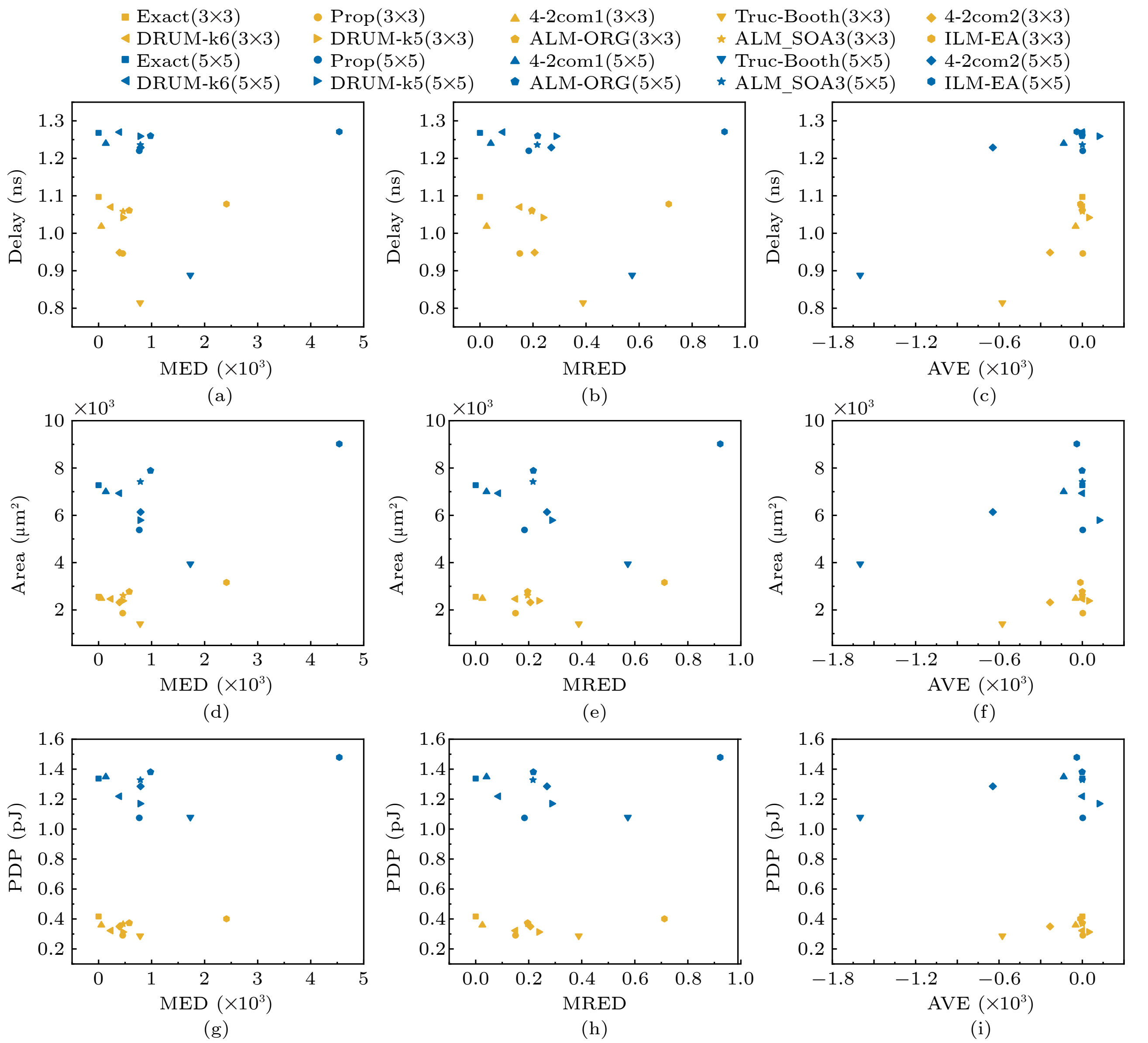


Fig.10. Error and circuit characteristics of the considered 3×3 and 5×5 PEs with different multiplication schemes. (a) Delay vs MED. (b) Delay vs MRED. (c) Delay vs AVE. (d) Area vs MED. (e) Area vs MRED. (f) Area vs AVE. (g) PDP vs MED. (h) PDP vs MRED. (i) PDP vs AVE.

MED and MRED than those using WT. The 5×5 PEs show relatively large AVEs when k equals 6 or 8, due to the error accumulation. However, benefiting from the unbiased designs for the multiplier and the accumulator, the AVEs of the proposed PEs are around 0 for all k values. This indicates that the proposed PE is unbiased. Figs.11(g)–11(i) show that the value of k does not have a significant effect on the overall PDP of the PEs. This means that the accumulator is not so important as the multiplier to the hardware overhead of the PE. As the ‘‘compile ultra’’ operation is used in the syntheses, the values of the critical path delay for the considered PEs are not strictly consistent with a theoretical analysis. Overall, the 3×3 PEs using WTs as the accumulators are faster than those using ATs, whereas the PEs using ATs are more hardware-efficient than those using WTs for 5×5 PEs.

5 Application of Approximate PEs to CNN Accelerators

This section analyzes the application of the approximate PEs in CNN accelerators. Specifically, the possible approximation level of PEs is tested for an acceptable accuracy loss in a CNN. The application

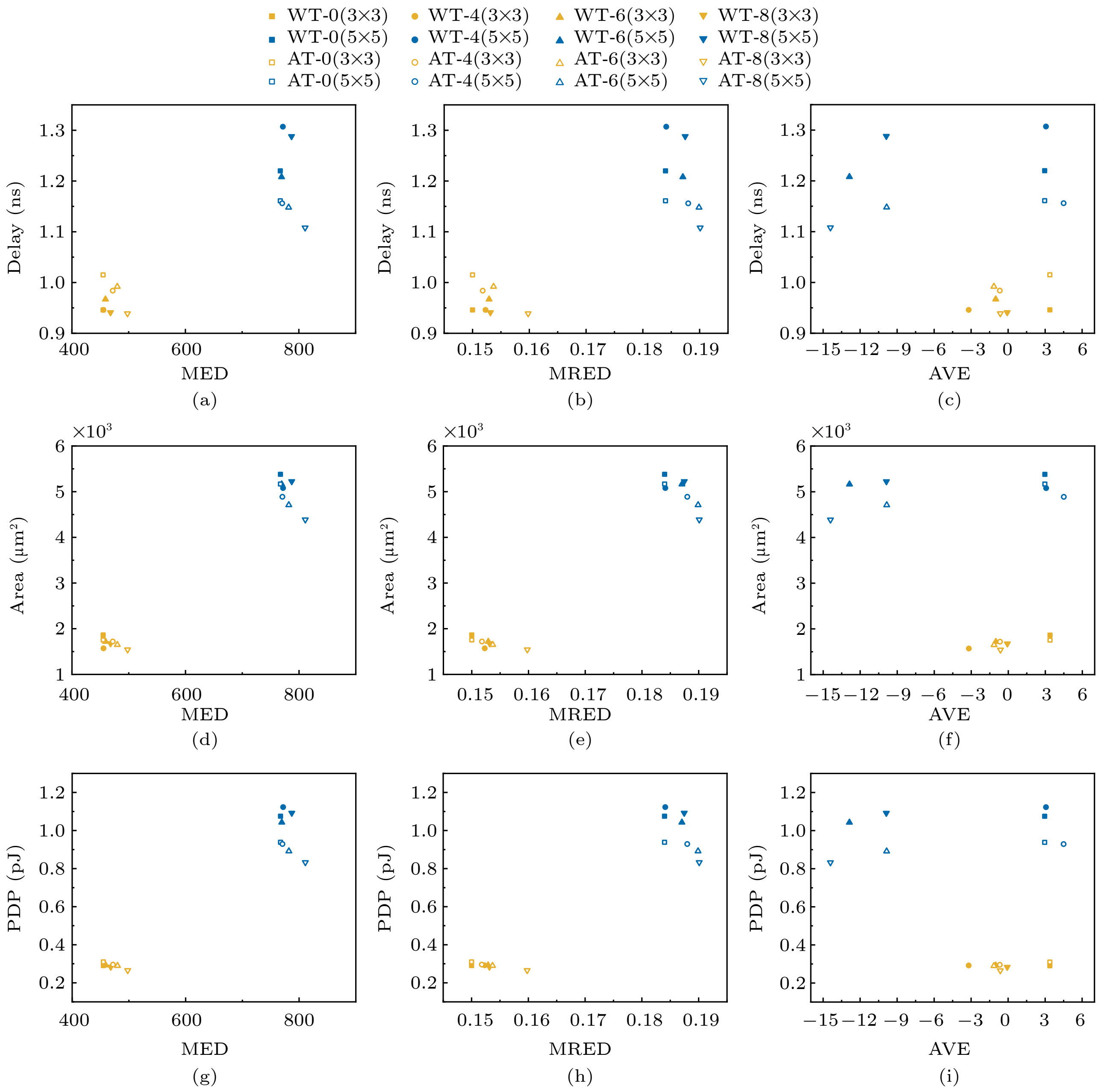


Fig.11. Error and circuit characteristics of the considered 3×3 and 5×5 PEs with different numbers of lower bits for approximate accumulation. (a) Delay vs MED. (b) Delay vs MRED. (c) Delay vs AVE. (d) Area vs MED. (e) Area vs MRED. (f) Area vs AVE. (g) PDP vs MED. (h) PDP vs MRED. (i) PDP vs AVE.

scenarios that approximate PEs can be used are analyzed. Moreover, the power efficiency and the energy efficiency of the CNN accelerators using approximate PEs are evaluated.

As two basic tasks using CNNs, face detection and alignment have widely been investigated. Using the correlation between these two tasks, a multi-task CNN (MTCNN) has been proposed for joint face detection and alignment[3, 30]. In this MTCNN, three CNNs, the proposal network (P-Net), the refine network (R-Net), and the output network (O-Net), are cascaded. The P-Net is a fully convolutional network consisting of four CONV layers and a max-pooling layer; the R-Net has three CONV layers, two max-pooling layers and two FC layers; the O-Net has four CONV layers, three max-pooling layers and two FC layers. The filter size for all the CONV layers is 3×3.

To analyze the accuracy and hardware capabilities of approximate PEs in the application of CNN accelerators, approximate PEs with various configurations are used to implement the above MTCNN. Also, the PEs consisting of state-of-the-art approximate multipliers considered in Section 4 are tested in this application. Note that the CNN accelerators are designed for the inference of the MTCNN. The approximate PEs are utilized in both CONV and FC layers. The accuracy and circuit measurements of the MTCNN accelerators using different PE structures are reported in Table 2, where the synthesis clock frequency for the power and area estimation is 700 MHz. TPR and NME are the true positive rate for the face detection on the FDDB[52] dataset and the normalized mean error for the face alignment on the AFLW dataset[53], respectively. MACs/Frame indicates the number of required MACs to detect the faces in one image averaged over the FDDB dataset.

Due to the large error biases, the approximate multipliers 4-2com1, 4-2com2, and Truc-Booth cannot achieve converged face detection results (TPR $<$ 0.1%), as shown in Table 2. Also, the TPR for DRUM-k5 is very low. Thus, in the design of CNN accelerators, AVE is the most important error metric for the approximate PE, i.e., its value should be close to 0 in order to make the system work. The proposed AT-4, DRUM-k6, and ILM-EA result in high TPRs. However, the latter two designs show poor NMEs. Thus, they are suitable for face detection rather than alignment. Table 2 also shows that the NME results on the AFLW dataset are approximately consistent with the error characteristics reported in Fig.11, i.e., a PE with a higher accuracy leads to a lower NME in the accelerator. Note that the NME for AT-8 is meaningless due to its significantly small TPR. However, the approximate PEs with small values of statistical error metrics do not necessarily result in high TPRs in CNN accelerators, e.g., the CNN accelerator using CR has a lower TPR than those using WT-0, WT-4, WT-6, AT-0, AT-4 and AT-6. The only difference between CR and WT-0 is the rounding scheme utilized in the approximate format. Rounding to the nearest number is used in CR in lieu of the stochastic rounding. The accelerator implemented by using an approximate PE can achieve a very close TPR to the Exact design, whereas its NME is relatively large. These indicate that the statistically-measured error results cannot properly guide the utilization of PEs in CNN accelerators; using stochastic rounding in a CNN accelerator improves the accuracy of face detection; the approximate PE is more effective for the face detection than for alignment.

Considering control logic and memory, the 16 PEs account for around 10.5%–14.9% of the area of the entire accelerator, as shown in Table 2. The area reduction due to the approximate PE is about 4% compared with the Exact 8-bit design. With a similar or higher accuracy, the proposed PEs also outperform DRUM in terms of power, speed and area. In addition, the power efficiency P_{eff} and the energy efficiency E_{eff} of a CNN accelerator are computed; they are given by

$$P_{\mathrm{eff}}=\frac{N_{\mathrm{PE}}\times N_{\mathrm{mul}}\times R_{\mathrm{PE}}\times f_{\mathrm{clk}}}{P}, \quad (3)$$

Table 2. Accuracy and Circuit Measurements of the MTCNN Accelerators with 16 PEs

Design	TPR (%)	NME (%)	Peak Frequency (MHz)	Power (mW)	Area (mm^2)	Peak Performance (GOPS)	MACs/ Frame (G)	P_{eff} (TOPS/W)	E_{eff} (mJ/Frame)
Exact	90.37	3.58	740	47.544	0.153 7	107	0.598	2.120	0.282
CR	83.39	3.43	783	45.984	0.147 5	113	0.590	2.192	0.269
WT-0	85.48	4.14	797	45.984	0.147 5	115	0.533	2.192	0.243
WT-4	87.68	4.30	796	45.998	0.147 6	115	0.528	2.191	0.240
WT-6	85.26	8.46	803	45.836	0.147 1	116	0.531	2.199	0.241
WT-8	73.33	19.60	804	45.772	0.146 7	116	0.539	2.202	0.244
AT-0	85.48	4.14	783	45.988	0.146 5	113	0.533	2.191	0.243
AT-4	89.19	4.57	794	45.868	0.146 2	114	0.588	2.197	0.267
AT-6	84.90	7.32	791	45.622	0.145 6	114	0.567	2.209	0.256
AT-8	40.44	N/A	792	45.342	0.144 6	114	N/A	2.223	N/A
4-2com1	N/A	N/A	787	46.396	0.153 1	114	N/A	2.172	N/A
4-2com2	N/A	N/A	781	46.208	0.151 6	113	N/A	2.181	N/A
Trunc-Booth	N/A	N/A	774	45.196	0.143 5	112	N/A	2.230	N/A
DRUM-k5	0.30	N/A	783	45.918	0.152 2	113	0.570	2.195	0.259
DRUM-k6	88.90	11.80	785	46.337	0.152 9	114	0.597	2.175	0.274
ALM-ORG	86.20	7.80	775	47.023	0.157 8	114	0.386	2.143	0.180
ALM-SOA3	83.30	7.70	781	46.884	0.156 3	114	0.378	2.149	0.176
ILM-EA	89.50	8.10	779	47.286	0.161 9	115	0.485	2.131	0.227

and

$$E_{\text{eff}} = \frac{N_{\text{mac}} \times P}{N_{\text{PE}} \times N_{\text{mul}} \times R_{\text{PE}} \times f_{\text{clk}}}. \quad (4)$$

In (3) and (4), N_{mac} is the average number of required MACs to detect the faces in an image, and N_{PE} and N_{mul} are the number of PEs in the accelerator and the number of multipliers in each PE, respectively. R_{PE} is the utilization rate of PEs, f_{clk} is the clock frequency, and P is the power dissipation. The utilization rate of PEs here is 100%. It is worth noting that the number of MACs/Frame using LMs is significantly smaller than that using the accurate and other approximate designs. Thus, LMs achieve higher improvements in energy efficiency. This indicates that some features of the approximation scheme used in LMs can make the task of image detection converge faster.

As per Figs.10 and 11 and Table 2, PEs are significantly smaller than the memory and control units in the MTCNN accelerator; thus, more PEs should be integrated within an accelerator to improve the power and energy efficiency. Table 3 reports the circuit measurements of MTCNN accelerators with 32 PEs and 64 PEs, respectively, where the utilization rates for the PEs are 100% and 99% respectively, with no extra memory required. It shows that the power dissipation and area of the accelerator are only slightly increased when the number of integrated PEs is doubled/quadrupled. The power efficiencies of the exact accelerators with 32 PEs and 64 PEs are 1.8x and 3.1x as high as that of the one with 16 PEs; the same improvements occur in energy efficiency. Also, the hardware improvements due to the approximate PEs are increased with the number of PEs integrated in the MTCNN accelerator. The approximate accelerator using 64 AT-4 achieves 18.02% reduction in area and 10.21% increase in power efficiency.

Table 3. Circuit Measurements of the MTCNN Accelerators with 32 PEs and 64 PEs

PEs	Design	Peak Frequency (MHz)	Power (mW)	Area (mm^2)	Peak Performance (GOPS)	P_{eff} (TOPS/W)	E_{eff} (mJ/Frame)
32	Exact	707	55.368	0.212 6	204	3.641	0.164
	WT-4	778	52.277	0.190 6	224	3.856	0.137
	AT-4	778	52.018	0.185 8	224	3.876	0.152
64	Exact	693	70.946	0.294 2	399	5.683	0.105
	WT-4	758	64.789	0.251 2	436	6.223	0.085
	AT-4	766	64.379	0.241 2	441	6.263	0.094

6 Conclusions

This paper proposed a low-power approximate PE and analyzed the application of the approximate PEs in CNN accelerators. In the PE design, an approximate data format is defined for the weights using stochastic rounding; hence, the multiplication is accomplished by using small LUTs, a simple adder and a shifter. Also, two approximate accumulators were proposed for the product accumulation in the PE. The evaluation results showed that the proposed approximate 3×3 PE achieves 29% reduction in PDP compared with the exact 8-bit fixed-point design. The proposed design has a higher accuracy and lower hardware consumption than the PEs using state-of-the-art approximate multipliers. The application of the approximate PEs in CNN accelerators is then analyzed by using an MTCNN, i.e., approximate PEs with various configurations are utilized to implement the MTCNN respectively. The simulation results showed that an approximate PE is more effective for the face detection than for alignment. An approximate PE with smaller statistically-measured error metrics does not necessarily result in a higher accuracy in the face detection. Using stochastic rounding in a CNN accelerator improves the face detection accuracy. The power efficiency and the energy efficiency of a CNN accelerator can be improved by properly increasing the number of integrated PEs.

References

[1] LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998, 86(11): 2278–2324. DOI: 10.1109/5.726791.

[2] Xie X Z, Niu J W, Liu X F, Li Q F, Wang Y, Han J, Tang S J. DG-CNN: Introducing margin information into convolutional neural networks for breast cancer diagnosis in ultrasound images. *Journal of Computer Science and Technology*, 2022, 37(2): 277–294. DOI: 10.1007/s11390-020-0192-0.

[3] Zhang K P, Zhang Z P, Li Z F, Qiao Y. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 2016, 23(10): 1499–1503. DOI: 10.1109/LSP.2016.2603342.

[4] Caroppo A, Leone A, Siciliano P. Comparison between

deep learning models and traditional machine learning approaches for facial expression recognition in ageing adults. *Journal of Computer Science and Technology*, 2020, 35(5): 1127–1146. DOI: 10.1007/s11390-020-9665-4.

[5] Ji S W, Xu W, Yang M, Yu K. 3D convolutional neural networks for human action recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2013, 35(1): 221–231. DOI: 10.1109/TPAMI.2012.59.

[6] Collobert R, Weston J. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. the 25th International Conference on Machine Learning*, Jul. 2008, pp.160–167. DOI: 10.1145/1390156.1390177.

[7] Matsugu M, Mori K, Mitari Y, Kaneda Y. Subject independent facial expression recognition with robust face detection using a convolutional neural network. *Neural Networks*, 2003, 16(5/6): 555–559. DOI: 10.1016/S0893-6080(03)00115-1.

[8] Szegedy C, Liu W, Jia Y Q, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A. Going deeper with convolutions. In *Proc. the 2015 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2015. DOI: 10.1109/CVPR.2015.7298594.

[9] Jouppi N P, Young C, Patil N *et al.* In-datacenter performance analysis of a tensor processing unit. In *Proc. the 44th Annual International Symposium on Computer Architecture*, Jun. 2017. DOI: 10.1145/3079856.3080246.

[10] Liu Z G, Whatmough P N, Zhu Y H, Mattina M. S2TA: Exploiting structured sparsity for energy-efficient mobile CNN acceleration. In *Proc. the 2022 IEEE Int. Symp. High-Performance Computer Architecture (HPCA)*, Apr. 2022, pp.573–586. DOI: 10.1109/HPCA53966.2022.00049.

[11] Li S Y, Hanson E, Qian X H, Li H H, Chen Y R. ESCALATE: Boosting the efficiency of sparse CNN accelerator with kernel decomposition. In *Proc. the 54th Annual IEEE/ACM Int. Symp. Microarchitecture*, Oct. 2021, pp.992–1004. DOI: 10.1145/3466752.3480043.

[12] Guesmi A, Alouani I, Khasawneh K N, Baklouti M, Frikha T, Abid M, Abu-Ghazaleh N. Defensive approximation: Securing CNNs using approximate computing. In *Proc. the 26th ACM Int. Conf. Architectural Support for Programming Languages and Operating Systems*, Apr. 2021, pp.990–1003. DOI: 10.1145/3445814.3446747.

[13] Ham T J, Jung S J, Kim S, Oh Y H, Park Y, Song Y, Park J H, Lee S, Park K, Lee J W, Jeong D K. A^3: Accelerating attention mechanisms in neural networks with approximation. In *Proc. the 2020 IEEE Int. Symp. High Performance Computer Architecture (HPCA)*, Feb. 2020, pp.328–341. DOI: 10.1109/HPCA47549.2020.00035.

[14] Mo H Y, Zhu W P, Hu W J, Wang G B, Li Q, Li A, Yin S Y, Wei S J, Liu L B. 9.2 A 28nm 12.1TOPS/W dual-mode CNN processor using effective-weight-based convolution and error-compensation-based prediction. In *Proc. the 2021 IEEE International Solid-State Circuits Conference (ISSCC)*, Feb. 2021, pp.146–148. DOI: 10.1109/ISSCC42613.2021.9365943.

[15] Sze V, Chen Y H, Yang T J, Emer J S. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 2017, 105(12): 2295–2329. DOI: 10.1109/JPROC.2017.2761740.

[16] Tu F B, Wu W W, Wang Y, Chen H J, Xiong F, Shi M, Li N, Deng J Y, Chen T B, Liu L B, Wei S J, Xie Y, Yin S Y. Evolver: A deep learning processor with on-device quantization-voltage-frequency tuning. *IEEE Journal of Solid-State Circuits*, 2021, 56(2): 658–673. DOI: 10.1109/JSSC.2020.3021661.

[17] Hubara I, Courbariaux M, Soudry D, El-Yaniv R, Bengio Y. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 2017, 18(1): 6869–6898.

[18] Gysel P M, Ristretto: Hardware-oriented approximation of convolutional neural networks [Master's Thesis]. University of California, Berkeley, 2016.

[19] Zhou S C, Wang Y Z, Wen H, He Q Y, Zou Y H. Balanced quantization: An effective and efficient approach to quantized neural networks. *Journal of Computer Science and Technology*, 2017, 32(4): 667–682. DOI: 10.1007/s11390-017-1750-y.

[20] Karpuzcu U R, Sinkar A, Kim N S, Torrellas J. EnergySmart: Toward energy-efficient manycores for near-threshold computing. In *Proc. the 19th Int. Symp. High Performance Computer Architecture (HPCA)*, Feb. 2013, pp.542–553. DOI: 10.1109/HPCA.2013.6522348.

[21] Villa O, Johnson D R, Oconnor M, Bolotin E, Nellans D, Luitjens J, Sakharnykh N, Wang P, Micikevicius P, Scudiero A, Keckler S W, Dally W J. Scaling the power wall: A path to exascale. In *Proc. the Int. Conf. High Performance Computing, Networking, Storage and Analysis*, Nov. 2014, pp.830–841. DOI: 10.1109/SC.2014.73.

[22] Han J, Orshansky M. Approximate computing: An emerging paradigm for energy-efficient design. In *Proc. the 18th IEEE European Test Symposium*, May 2013. DOI: 10.1109/ETS.2013.6569370.

[23] Yuan M K, Dai L Q, Yan D M, Zhang L Q, Xiao J, Zhang X P. Fast and error-bounded space-variant bilateral filtering. *Journal of Computer Science and Technology*, 2019, 34(3): 550–568. DOI: 10.1007/s11390-019-1926-8.

[24] Zhang Q, Wang T, Tian Y, Yuan F, Xu Q. ApproxANN: An approximate computing framework for artificial neural network. In *Proc. the 2015 Design, Automation & Test in Europe Conference & Exhibition*, Mar. 2015, pp.701–706.

[25] Chen T S, Du Z D, Sun N H, Wang J, Wu C Y, Chen Y J, Temam O. DianNao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. *ACM SIGARCH Computer Architecture News*, 2014, 42(1): 269–284. DOI: 10.1145/2654822.2541967.

[26] Ansari M S, Mrazek V, Cockburn B F, Sekanina L, Vasicek Z, Han J. Improving the accuracy and hardware efficiency of neural networks using approximate multipliers. *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, 2020, 28(2): 317–328. DOI: 10.1109/TVLSI.2019.

2940943.

[27] Jiang H L, Santiago F J H, Mo H, Liu L B, Han J. Approximate arithmetic circuits: A survey, characterization, and recent applications. *Proceedings of the IEEE*, 2020, 108(12): 2108–2135. DOI: 10.1109/JPROC.2020.3006451.

[28] Courbariaux M, Bengio Y, David J P. Training deep neural networks with low precision multiplications. arXiv: 1412.7024, 2014. https://arxiv.org/abs/1412.7024, Apr. 2023.

[29] Chen Y H, Krishna T, Emer J S, Sze V. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE Journal of Solid-State Circuits*, 2017, 52(1): 127–138. DOI: 10.1109/JSSC.2016.2616357.

[30] Mo H Y, Liu L B, Zhu W P, Li Q, Liu H, Hu W J, Wang Y, Wei S J. A 1.17 TOPS/W, 150fps accelerator for multi-face detection and alignment. In *Proc. the 56th ACM/IEEE Design Automation Conference*, Jun. 2019.

[31] Jain S, Venkataramani S, Srinivasan V, Choi J, Gopalakrishnan K, Chang L. BiScaled-DNN: Quantizing long-tailed datastructures with two scale factors for deep neural networks. In *Proc. the 56th ACM/IEEE Design Automation Conference (DAC)*, Jun. 2019.

[32] Nagel M, Fournarakis M, Amjad R A, Bondarenko Y, van Baalen M, Blankevoort T. A white paper on neural network quantization. arXiv: 2106.08295, 2021. https://arxiv.org/abs/2106.08295, Apr. 2023.

[33] Parashar A, Rhu M, Mukkara A, Puglielli A, Venkatesan R, Khailany B, Emer J, Keckler S W, Dally W J. SCNN: An accelerator for compressed-sparse convolutional neural networks. In *Proc. the 44th Annual International Symposium on Computer Architecture*, Jun. 2017, pp.27–40. DOI: 10.1145/3079856.3080254.

[34] Zervakis G, Tsoumanis K, Xydis S, Soudris D, Pekmestzi K. Design-efficient approximate multiplication circuits through partial product perforation. *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, 2016, 24(10): 3105–3117. DOI: 10.1109/TVLSI.2016.2535398.

[35] Kyaw K Y, Goh W L, Yeo K S. Low-power high-speed multiplier for error-tolerant application. In *Proc. the 2010 IEEE International Conference of Electron Devices and Solid-State Circuits (EDSSC)*, Dec. 2010. DOI: 10.1109/EDSSC.2010.5713751.

[36] Hashemi S, Bahar R I, Reda S. DRUM: A dynamic range unbiased multiplier for approximate applications. In *Proc. the 2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov. 2015, pp.418–425. DOI: 10.1109/ICCAD.2015.7372600.

[37] Chen Y H, Chang T Y. A high-accuracy adaptive conditional-probability estimator for fixed-width booth multipliers. *IEEE Trans. Circuits and Systems I: Regular Papers*, 2012, 59(3): 594–603. DOI: 10.1109/TCSI.2011.2167275.

[38] He Y J, Yi X L, Zhang Z J, Ma B, Li Q. A probabilistic prediction-based fixed-width booth multiplier for approximate computing. *IEEE Trans. Circuits and Systems I: Regular Papers*, 2020, 67(12): 4794–4803. DOI: 10.1109/TCSI.2020.3001654.

[39] Lin C H, Lin I C. High accuracy approximate multiplier with error correction. In *Proc. the 31st International Conference on Computer Design (ICCD)*, Oct. 2013, pp.33–38. DOI: 10.1109/ICCD.2013.6657022.

[40] Kong T Q, Li S G. Design and analysis of approximate 4-2 compressors for high-accuracy multipliers. *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, 2021, 29(10): 1771–1781. DOI: 10.1109/TVLSI.2021.3104145.

[41] Esposito D, Strollo A G M, Napoli E, De Caro D, Petra N. Approximate multipliers based on new approximate compressors. *IEEE Trans. Circuits and Systems I: Regular Papers*, 2018, 65(12): 4169–4182. DOI: 10.1109/TCSI.2018.2839266.

[42] Venkatachalam S, Ko S B. Design of power and area efficient approximate multipliers. *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, 2017, 25(5): 1782–1786. DOI: 10.1109/TVLSI.2016.2643639.

[43] Mitchell J N. Computer multiplication and division using binary logarithms. *IRE Trans. Electronic Computers*, 1962, EC-11(4): 512–517. DOI: 10.1109/TEC.1962.5219391.

[44] Liu W Q, Xu J H, Wang D Y, Wang C H, Montuschi P, Lombardi F. Design and evaluation of approximate logarithmic multipliers for low power error-tolerant applications. *IEEE Trans. Circuits and Systems I: Regular Papers*, 2018, 65(9): 2856–2868. DOI: 10.1109/TCSI.2018.2792902.

[45] Ansari M S, Cockburn B F, Han J. An improved logarithmic multiplier for energy-efficient neural computing. *IEEE Trans. Computers*, 2021, 70(4): 614–625. DOI: 10.1109/TC.2020.2992113.

[46] Norrie T, Patil N, Yoon D H, Kurian G, Li S, Laudon J, Young C, Jouppi N P, Patterson D. Google's training chips revealed: TPUv2 and TPUv3. In *Proc. the Hot Chips 32 Symposium*, Aug. 2020. DOI: 10.1109/HCS49909.2020.9220735.

[47] Liu S T, Han J. Hardware ODE solvers using stochastic circuits. In *Proc. the 54th ACM/EDAC/IEEE Design Automation Conference*, Jun. 2017. DOI: 10.1145/3061639.3062258.

[48] Ranasinghe A C, Gerez S H. Glitch-optimized circuit blocks for low-power high-performance booth multipliers. *IEEE Trans. Very Large Scale Integration Systems*, 2020, 28(9): 2028–2041. DOI: 10.1109/TVLSI.2020.3009239.

[49] Jiang H L, Liu C, Liu L B, Lombardi F, Han J. A review, classification, and comparative evaluation of approximate arithmetic circuits. *ACM Journal on Emerging Technologies in Computing Systems*, 2017, 13(4): Article No. 60. DOI: 10.1145/3094124.

[50] Mahdiani H R, Ahmadi A, Fakhraie S M, Lucas C. Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications. *IEEE Trans. Circuits and Systems I: Regular Papers*, 2010, 57(4): 850–862. DOI: 10.1109/TCSI.2009.2027626.

[51] Jiang H L, Liu L B, Jonker P P, Elliott D G, Lombardi

F, Han J. A high-performance and energy-efficient FIR adaptive filter using approximate distributed arithmetic circuits. *IEEE Trans. Circuits and Systems I: Regular Papers*, 2019, 66(1): 313–326. DOI: 10.1109/TCSI.2018.2856513.

[52] Jain V, Learned-Miller E G. FDDB: A benchmark for face detection in unconstrained settings. Technical Report, UM-CS-2010–009, Dept. of Computer Science, University of Massachusetts, Amherst, 2010. http://vis-www.cs.umass.edu/fddb/FDDB-folds.tgz, Mar. 2023.

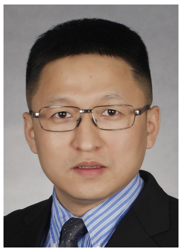
[53] Köstinger M, Wohlhart P, Roth P M, Bischof H. Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization. In *Proc. the 2011 IEEE International Conference on Computer Vision Workshops*, Nov. 2011, pp.2144–2151. DOI: 10.1109/ICCVW.2011.6130513.

Tong Li received his Bachelor's degree in electronic information engineering from College of Electronic Science and Engineering, Jilin University, Changchun, in 2016. He is currently pursuing his Ph.D. degree with the Department of Micro-Nano Electronics at Shanghai Jiao Tong University, Shanghai. His research interests include high energy efficient arithmetic circuit design and SoC design.

Hong-Lan Jiang received her B.Sc. and Master's degrees in instrument science and technology from Harbin Institute of Technology, Harbin, in 2011 and 2013, respectively. In 2018, she received her Ph.D. degree in integrated circuits and systems from University of Alberta, Edmonton. From 2018 to 2021, she worked as a postdoctoral fellow with the School of Integrated Circuits, Tsinghua University, Beijing. She is currently an associate professor with the Department of Micro-Nano Electronics, Shanghai Jiao Tong University, Shanghai. Her research interests include approximate computing, reconfigurable computing, and stochastic computing. She serves as an associate editor for Microelectronics Reliability. She is a member of CCF and IEEE.

Hai Mo received his B.S. degree in computer science from Huazhong University of Science and Technology, Wuhan, in 2019. He received his M.S. degree in integrated circuit engineering from the School of Integrated Circuits at Tsinghua University, Beijing, in 2022. His research interest spans the area of architecture design, in-memory computing, and hardware implementation for deep learning accelerators.

Jie Han received his B.Sc. degree in electronic engineering from Tsinghua University, Beijing, in 1999, and his Ph.D. degree in applied physics from the Delft University of Technology, Delft, in 2004. He is currently a professor and program director of computer engineering in the Department of Electrical and Computer Engineering, University of Alberta, Edmonton. His research interests include approximate computing, stochastic computing, reliability and fault tolerance, nanoelectronic circuits and systems, novel computational models for nanoscale and biological applications. He was a recipient of the Best Paper Award at NANOARCH 2015 and DATE 2023, and Best Paper Nominations at GLSVLSI 2015, NANOARCH 2016, ISQED 2018 and DATE 2022. He was nominated for the 2006 Christiaan Huygens Prize of Science by the Royal Dutch Academy of Science. His work was recognized by Science, for developing a theory of fault-tolerant nanocircuits in 2005. He served as a general chair for NANOARCH 2021, GLSVLSI 2017 and DFT 2013, and a technical program committee chair for NANOARCH 2022, GLSVLSI 2016, DFT 2012, and the Symposium on Stochastic and Approximate Computing for Signal Processing and Machine Learning in 2017. He serves (or has served) as an associate editor for the IEEE Transactions on Emerging Topics in Computing (TETC), the IEEE Transactions on Nanotechnology, the IEEE Circuits and Systems Magazine, the IEEE Open Journal of the Computer Society, Microelectronics Reliability and the Journal of Electronic Testing: Test and Application (JETTA, Elsevier). He is a senior member of IEEE.



Lei-Bo Liu received his B.S. degree in electronic engineering and his Ph.D. degree in electronic engineering with the Institute of Microelectronics, both from Tsinghua University, Beijing, in 1999 and 2004, respectively. He is currently a full professor with the School of Integrated Circuits, Tsinghua University, Beijing. His current research interests include reconfigurable computing, mobile computing, and very large-scale integration digital signal processing.

Zhi-Gang Mao received his Ph.D. degree in electronic engineering from Rennes University, Rennes, in 1992. He is currently a professor with the Department of Micro-Nano Electronics, Shanghai Jiao Tong University, Shanghai. His current research interests include VLSI (very large scale integration) design methodology, high-speed digital circuit design technology, signal processor architecture, and hardware security technology and reliability in semiconductor devices.