An Efficient FPGA-Based Dilated and Transposed Convolutional Neural Network Accelerator

Tsung-Hsi Wu, Chang Shu, and Tsung-Te Liu, *Member, IEEE*

***Abstract*— This work presents a Field Programmable Gate Array (FPGA)-based deep neural network (DNN) accelerator that can maintain consistently high efficiency when executing various neural network architectures, including convolutional neural network (CNN), transposed and dilated convolution (TD-convolution) operations for modern computer vision (CV) tasks. To deal with the utilization degradation issue with a large processing unit (PE) array, a 3-D mapping strategy that adaptively tailors different layer configurations is proposed to optimize the parallelism dimensions of the PE, which significantly increases the hardware utilization to enhance the accelerator efficiency. Moreover, to minimize the implementation and performance overhead resulting from the TD-convolution operations, a unified processing flow is proposed to realize an integrated operation of traditional and TD-convolution. This allows the accelerator to bypass redundant zero operations, further boosting overall efficiency. The 4096-PE accelerator implementation on Intel Stratix 10 FPGA achieves a throughput performance of 2.597–2.870 TOPS with an efficiency of 0.63-0.70 GOPS/DSP across various DNN networks. This represents 1.72× and 1.73× improvement in throughput and efficiency, respectively, compared to the state-of-the-art designs.**

***Index Terms*— Computer vision, deep neural network, field programmable gate array, hardware accelerator.**

I. Introduction

DEEP neural networks (DNNs) have been widely adopted in various computer vision (CV) tasks due to their superior accuracy over the traditional methods. Typical applications of the CV-based DNNs range from image classification [1] and object detection [2] to image segmentation [3]. To achieve higher accuracy performance, larger and deeper DNN models are proposed [1], [2], [3], and [19]. This leads to a substantial increase in computation complexity. Now Giga-level multiply-and-accumulate (MAC) operations are usually required for CV-based DNNs for high accuracy performance. Moreover, these CV-based models often employ the variants of traditional convolution, including transposed convolution (T-conv) and dilated convolution (D-conv), to enhance the model accuracy performance. While T-conv is utilized for upsampling and generating high-resolution outputs by interspersing zeros between feature maps, D-conv is employed to expand the receptive field and capture more contextual information from the input feature map by interspersing zeros between weight kernels. These abnormal convolutions could seriously degrade the computation throughput [12], [13], [14], [16], [17], [18]. Typical ASIC accelerators have limited hardware resources, which hinders the possibility of real-time inference operation. On the other hand, the FPGA-based DNN accelerator has been gaining popularity to achieve real-time DNN inference tasks due to its rich processing and memory resources. Moreover, it offers higher reconfigurability and lower development cost than the ASIC-based accelerators [4], [5], [6].

Although FPGA inherently possesses a large number of processing element (PE) units, the corresponding accelerator throughput performances in the previous works [4] and [5] are unfortunately not improved in proportion to the total number of PE units (#PE). The relationship between the throughput and #PE is given by:

$$Throughput \propto \#PE \times \ Freq \times Efficiency_{op} \qquad (1)$$

where *Freq* represents the system frequency, and *Efficiency$_{op}$* represents the operational efficiency of the accelerator, which is the product of PE utilization and runtime efficiency. Ideally, throughput improvement should be proportional to *#PE*. However, PE utilization and runtime efficiency might degrade with an increasing *#PE*, which impacts the highest throughput that an accelerator can potentially realize. A saturation of throughput improvement due to the degradation of PE utilization with increased *#PE* is observed in [4]. Reference [5] also showed a decrease in operational frequency as *#PE* is scaled from 1024 to 4096. As a result, the potential degradation in PE utilization and runtime efficiency brought by increasing the *#PE* must be carefully considered to maximize the possible performance gain.

On the other hand, in addition to the traditional convolution, T-conv and D-conv (TD-conv) could serve as major structural components in modern CV-based models. For example, 53% of the operations are T-conv for Gr-ConvNet [19], while 98% are D-conv for SegNet [3]. Therefore, it is imperative to demonstrate efficient operations in all three types of convolutions. While most DNN accelerators nowadays can perform excellently in traditional convolution tasks, they cannot perform well on TD-conv models. Recent works [12], [13], [14], [16], [17], and [18] proposed methods that skip the unnecessary zero operations in TD-conv to recover the

Manuscript received 29 April 2024; revised 18 June 2024; accepted 27 June 2024. Date of publication 27 August 2024; date of current version 25 October 2024. This work was supported in part by the National Science and Technology Council, Taiwan, and in part by the Intelligent and Sustainable Medical Electronics Research Fund in National Taiwan University. This article was recommended by Associate Editor Y. Lao. *(Tsung-Hsi Wu and Chang Shu contributed equally to this work.) (Corresponding author: Tsung-Te Liu.)*

Tsung-Hsi Wu was with the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 10617, Taiwan. He is now with Google LLC, New Taipei City 220, Taiwan.

Chang Shu and Tsung-Te Liu are with the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 10617, Taiwan (e-mail: ttliu@ntu.edu.tw).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSI.2024.3428636.

Digital Object Identifier 10.1109/TCSI.2024.3428636

corresponding performance loss when executing these models. However, these approaches need either additional independent architecture or extra processing flow to support the required computation, which causes significant implementation overhead. As a result, an efficient accelerator architecture that simultaneously optimizes both conventional and TD-conv operations is essential to enhance the overall efficiency.

To address the above issues, this paper proposes a unified FPGA accelerator architecture that supports both traditional and newly emerging convolutions with high efficiency. The key contributions of this paper are as follows.

1) A reconfigurable three-dimensional (3-D) PE mapping approach that dynamically selects the optimal parallelism degree and dimension is proposed to accommodate various DNN characteristics. The proposed approach can effectively maintain high PE utilization on different DNN models that exhibit a wide range of weight shapes and various operation types to simultaneously achieve high flexibility and efficiency, even under the use case of large *#PE*.
2) A unified and efficient TD-conv processing scheme is proposed to avoid all the unnecessary zero-padding operations without the need for additional process flow or hardware, which realizes efficient computations for both conventional and TD-conv operations.
3) A 4096-PE accelerator using the proposed 3-D PE mapping approach and unified processing flow is implemented on Intel Stratix 10 SoC FPGA. The proposed accelerator achieves 2.597–2.870 TOPS throughput performance with an efficiency of 0.63-0.70 GOPS/DSP across various DNN networks including VGG-16, ResNet-50, ENet, Gr-ConvNet and SegNet. This represents 1.72× and 1.73× performance improvement in throughput and efficiency, respectively, compared with the state-of-the-art designs.

The organization of this paper is as follows. Section II presents the proposed 3-D mapping approach. Section III discusses how the proposed accelerator efficiently supports both traditional and TD-conv operations. Section IV describes the proposed hardware architecture. Section V shows the implementation results and performance comparisons. Finally, Section VI summarizes this paper.

II. Proposed 3-D PE Mapping Approach

A. Processor Design Issues With Large-Scale PE Array

Many FPGA accelerators have been proposed in pursuit of high-performance DNN inference [4], [5], [6], [8], [9], and [11] by utilizing the maximal processing resource in FPGAs. However, when supporting various DNNs, they all suffer from diminishing performance improvement as *#PE* increases. This is mainly due to the severely decreased PE utilization when the dimensions and shapes of DNNs vary dramatically across different network layers of architectures. Reference [4] used a fixed PE mapping strategy and showed an apparent saturation in throughput improvement as *#PE* grows, where a 3.3x increase in *#PE* results in only a 1.06× to 1.66× improvement in throughput. Reference [5] also showed

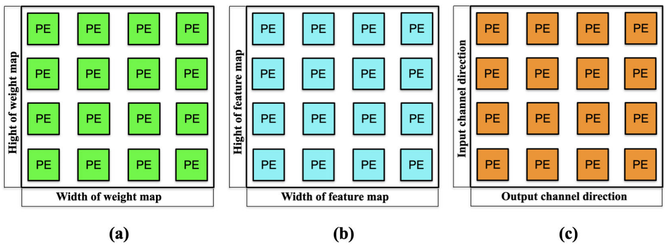

Fig. 1. Three categories of the physical dimension mapping scheme of the PE: (a) Weight-level parallelism, (b) Feature map-level parallelism, and (c) Channel-level parallelism.

a 12% $Efficiency_{op}$ degradation when the number of PE is scaled from 1024 to 4096. To address this issue, [6] selected an optimal PE mapping scheme for every layer based on the roofline model [7]. However, since its accelerator implementation still employed a fixed PE mapping scheme across different DNN layers, it achieves only sub-optimal performance. Reference [8] explored parallelism within weight maps using a 3×3 convolution engine, but its flexibility and efficiency seriously degrade when applied to different weight shapes. Reference [9] mapped all convolutional layers onto dedicated hardware with different mapping schemes for AlexNet [10]. However, its scalability is highly limited as the DNN models become deeper. Reference [11] showed that the unrolling and tiling dimensions, as well as the convolution loop order, could significantly affect the data access and reuse patterns. Based on the above discussions, a flexible PE mapping approach that adaptively caters to the corresponding shapes and sizes of different target DNN models can be a promising approach to potentially exercise the maximal computation capacity and achieve enhanced accelerator performance across various DNN models.

In this section, we will introduce the proposed reconfigurable 3-D PE mapping approach to address the PE utilization degradation issue, especially when a large number of PEs are deployed in a DNN accelerator. In Section II-B, we will discuss the attributes of several traditional two-dimensional (2-D) parallelism schemes and their suitable application scenarios. The proposed 3-D mapping approach that combines the advantages of multiple 2-D parallelism strategies will be described in detail in Section II-C. Finally, Section II-D will present the experimental results on several DNN models.

B. Characteristics of Different 2-D Parallelism Schemes

Maximizing the level of parallelism according to different DNN layers has been explored by several previous works [4], [6], [7], [8], [9], [10], [11], [12], [15], and [20]. In this section, we categorize different parallelism dimensions into the following three groups based on the reported architectures [4], [6], [7], [8], [9], [10], [11], [12], [15], and [20], as shown in Fig. 1.

1) Weight-Level Parallelism: [8], [9], [15], and [20] exploited weight-level parallelism, which was unrolled and mapped to the PE by the x-axis and y-axis of the weight maps, as shown in Fig. 1(a). This parallelism demonstrates high PE utilization, particularly with the specific weight kernel

TABLE I
ADVANTAGES AND DISADVANTAGES OF DIFFERENT PARALLELISM STRATEGIES

	Advantage	Disadvantage
Weight-level	Suitable weight kernel	Unsuitable weight kernel
Map-level	Large feature map size	Small feature map size
Channel-level	Large channel number	Small channel number

size. However, when the shape of the weight maps changes dramatically, significant degradation in PE utilization occurs.

2) Map-Level Parallelism: [4], [11], [15], and [20] exploited map-level parallelism, which was unrolled and mapped to the PE by the x-axis and y-axis of the output feature map, as shown in Fig. 1(b). This parallelism would perform well when the output feature size is large enough, typically seen in the early layers of DNN. Nevertheless, if the output features become smaller, such as in the late layer, or when a larger PE array is employed, the decrease in PE utilization would become substantial. This would cause severe performance degradation.

3) Channel-Level Parallelism: Unlike the previous two categories, channel-level parallelism elevates the level of parallelism to the third dimension, where unrolling across input feature maps or weight maps is explored. References [6], [7], [9], [10], and [12] utilized channel-level parallelism and showed great PE utilization across several layers. Since most of the input/output channel numbers have a power of two, the resulting performance can be less susceptible to the size changes of the feature map numbers across layers. However, the degradation in PE utilization would still occur if the input/output channel number is small. Fig. 1(c) illustrates this category.

In summary, as shown in Table I, each parallelism strategy has advantages and disadvantages under different application scenarios. The key idea of the proposed 3-D mapping approach is to adaptively identify and employ the most suitable mapping method for different DNN models. This way, the accelerator can maintain high PE utilization with minimum idle resources across various DNN models to realize the optimum inference performance.

C. Proposed 3-D PE Mapping Approach

To better explain the proposed 3-D mapping approach, two of the representative DNN models with substantially different characteristics for image segmentation and robotic grasping, SegNet and Gr-ConvNet, are used as application examples to demonstrate the capability of the proposed approach. Table II summarizes and compares the characteristics of the two models. Unlike Gr-ConvNet, SegNet has much more convolution layers (i.e., 37 layers) and larger channel numbers (i.e., up to 512 channels). The two models also have a wide range of weight shapes and operation types, including traditional and T-conv/D-conv operations, and have a dramatic difference in Giga MACs (GMACs) by 47x. The detailed input and output feature channel dimensions of these two models, N_{if_ch} and N_{of_ch}, are shown in Fig. 2.

TABLE II
MODEL CHARACTERISTICS OF SEGNET AND GR-CONVNET

Characteristics	SegNet [3]	Gr-ConvNet [19]
Number of layers	37	17
Weight size	1 x 1, 3 x 3, 7 x 7	2 x 2, 3 x 3, 4 x 4, 9 x 9
Channel number	3, 64, 128, 256, 512	4, 16, 32, 64
Feature map size	40 x 30, 80 x 60, 320 x 240	56 x 56, 112 x 112, 224 x 224
Number of GMAC operations	331	7.02

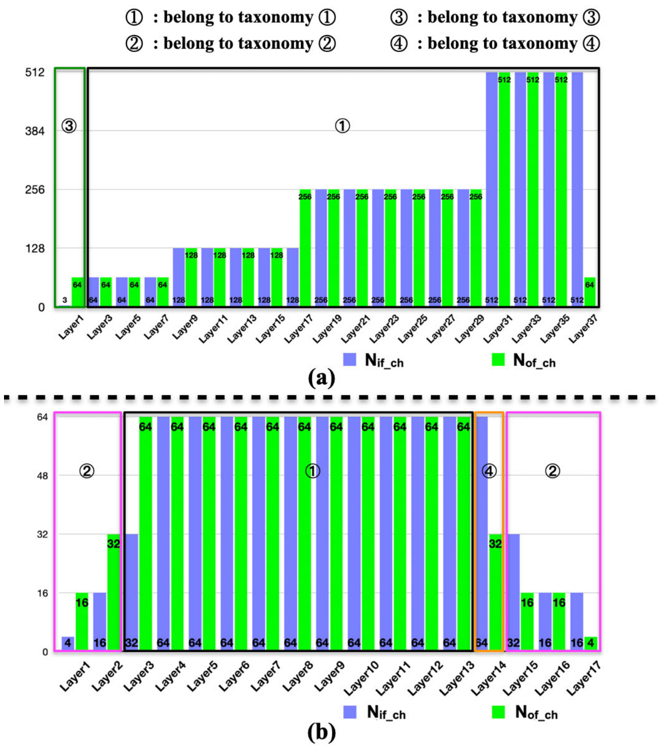

Fig. 2. Input and output feature channel dimension of each layer in (a) SegNet and (b) Gr-ConvNet.

To demonstrate the design concept and the performance advantage of the proposed approach, we assume an accelerator design example with the baseline mapping implementation shown in Fig. 3, which achieves the highest average PE utilization across different layers of the target models. The accelerator consists of 16 processing blocks, each of which comprises 128 PEs, with the selected parallelism size of the input and output feature channel dimension for each convolution layer, $P_{if\_ch} = 16$ and $P_{of\_ch} = 64$, as well as output feature x-axis dimension $P_{of\_x} = 8$. For this baseline implementation, we can observe $N_{if\_ch} \geq P_{if\_ch}$ and $N_{of\_ch} \geq P_{of\_ch}$ in the middle layers of both SegNet and Gr-ConvNet. This means all PEs can be fully utilized for convolution for this baseline implementation, which realizes the high processing efficiency of the accelerator.

On the other hand, when N_{if_ch} and N_{of_ch} are smaller than P_{if_ch} and P_{of_ch}, such as in the case of the early layers of Gr-ConvNet, a large number of PE in the baseline implementation would be idled since the feature map could not map to

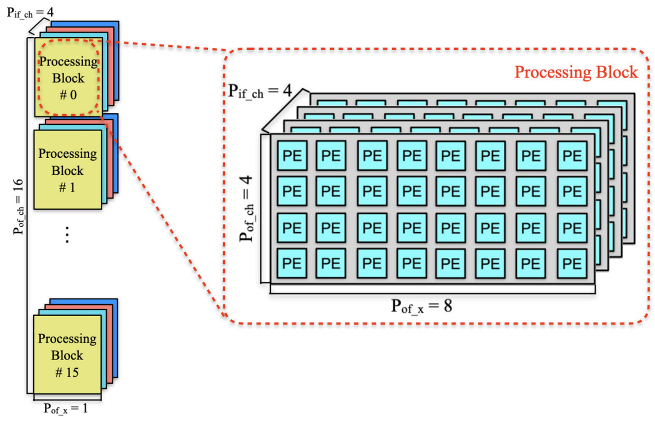

Fig. 3. Accelerator design example with the baseline mapping $P_{if\_ch} = 16$, $P_{of\_ch} = 64$, and $P_{of\_x} = 8$.

all the PE. This would cause low PE utilization and degraded performance. Since there is generally an inverse relationship between the channel numbers and the feature map sizes in DNNs, we could instead explore less parallelism in both P_{if_ch} and P_{of_ch}, while leveraging the spared parallelism in P_{of_x} to enhance PE utilization. Therefore, the main idea of the proposed 3-D mapping approach is to adaptively reconfigure the PE array with the optimum parallelism strategy based on the actual DNN characteristics to maximize PE utilization. We taxonomize all layer configurations into four groups according to the relationship between the input and output channel numbers, as illustrated in Fig. 2, and the degree of parallelism in each dimension. These four categories of layer configurations and their relationships are given by:

Taxonomy ①: $N_{if\_ch} \geq P_{if\_ch}$ and $N_{of\_ch} \geq P_{of\_ch}$

When the number of PE is small, the computation of most DNN layers belongs to this taxonomy. For the example shown in Fig. 2, layer 2 to layer 37 of SegNet and layer 3 to layer 13 of Gr-ConvNet belong to this. Therefore, the baseline implementation shown in Fig. 3 and Fig. 4(a) performs well, and no further reconfiguration is needed. In this case, all the PEs can be fully utilized to achieve high performance.

Taxonomy ②: $N_{if\_ch} \geq P_{if\_ch}$ and $N_{of\_ch} \geq P_{of\_ch}$

Taxonomy ② happens typically in early DNN layers where the numbers of input channels are small, and the sizes of feature maps are large. For example, layer 1 of Gr-ConvNet falls into this taxonomy with $N_{if\_ch} = 4$, $N_{of\_ch} = 16$, and $N_{of\_x} = 224$. Since N_{if_ch} and N_{of_ch} are far smaller than P_{if_ch} and P_{of_ch}, most PEs are idle in the baseline mapping, which results in a very low PE utilization of 0.0625.

Because the number of input channels is small and the sizes of the output feature maps are large, P_{if_ch} and P_{of_ch} can be actually folded to provide additional parallelism for P_{of_x}. Fig. 4(b) shows a reconfiguration example with a folding factor 4. The parallelism in each dimension now is $P_{if\_ch} = 4$, $P_{of\_ch} = 16$, and $P_{of\_x} = 128$, respectively. With this optimized configuration, the PE utilization can significantly increase from 0.0625 to 0.875.

Taxonomy③: $N_{if\_ch} < P_{if\_ch}$ and $N_{of\_ch} \geq P_{of\_ch}$

Similar to Taxonomy②, Taxonomy ③ usually happens in early layers, but with the output channel size larger than

TABLE III
SUMMARY OF THE RECONFIGURABLE 3-D MAPPING TAXONOMY

Taxonomy	Conditions	Folding direction
①	$N_{if\_ch} \geq P_{if\_ch}$ and $N_{of\_ch} \geq P_{of\_ch}$	-
②	$N_{if\_ch} < P_{if\_ch}$ and $N_{of\_ch} < P_{of\_ch}$	P_{if_ch} and P_{of_ch}
③	$N_{if\_ch} < P_{if\_ch}$ and $N_{of\_ch} \geq P_{of\_ch}$	P_{if_ch}
④	$N_{if\_ch} \geq P_{if\_ch}$ and $N_{of\_ch} < P_{of\_ch}$	P_{of_ch}

the unrolling factor of output channel direction. Layer 1 of SegNet falls into this taxonomy with $N_{if\_ch} = 3$, $N_{of\_ch} = 64$, and $N_{of\_x} = 160$. Since N_{of_ch} is larger than or equal to P_{of_ch}, we only need to fold P_{if_ch} by a factor of 4, and the spared parallelism can be leveraged by the dimension P_{of_x}, as shown in Fig. 4(c). The total degree of parallelism can thus be reconfigured as $P_{if\_ch} = 4$, $P_{of\_ch} = 64$, and $P_{of\_x} = 32$. Therefore, the PE utilization can rise from 0.1875 to 0.75.

Taxonomy④:$N_{if_ch} \geq P_{if_ch}$and $N_{of\_ch} < P_{of\_ch}$

Layer 14 of Gr-ConvNet belongs to this taxonomy with $N_{if\_ch} = 64$ and $N_{of\_ch} = 32$. Since N_{if_ch} is larger than or equal to P_{if_ch}, we need to fold P_{of_ch} only by a factor of 2, and the spared parallelism can be leveraged by the dimension P_{of_x}, as shown in Fig. 4(d). The total degree of parallelism can now be reconfigured as $P_{if\_ch} = 16$, $P_{of\_ch} = 32$, and $P_{of\_x} = 16$. As a result, the overall PE utilization can be substantially increased from 0.5 to 1.

D. Summary and Experimental Results

Table III summarizes the taxonomy of the proposed reconfigurable 3-D PE mapping approach. While previous works suffer from low PE utilization and severe performance degradation across different DNN architectures or layers, the proposed approach can effectively maintain PE utilization to realize high performance. Fig. 5 compares the PE utilization performances of the baseline mapping and the proposed reconfigurable 3-D mapping approach for SegNet and Gr-ConvNet. When executing inference in complex SegNet, both methods can realize high PE utilization since almost all layers belong to Taxonomy①. However, significant degradation in PE utilization can be observed in the early and late layers of Gr-ConvNet due to the unoptimized parallelism strategy explained in the previous section. As a result, the overall PE utilization drops to 0.74 with the baseline mapping. On the other hand, the proposed reconfigurable 3-D PE mapping approach can maintain a high PE utilization of 0.99 to maximize the performance for both networks. This represents a significant improvement of 34% in the PE utilization for Gr-ConvNet.

III. PROPOSED PROCESSING FLOW FOR TD-CONV LAYERS

A. Design Issues of Supporting T- and D-Convolution

T-conv and D-conv operations become popular in emerging variants of DNNs to support modern CV tasks. Unlike traditional convolution, T-conv adds zero paddings between the input pixels and up-samples the input feature map into

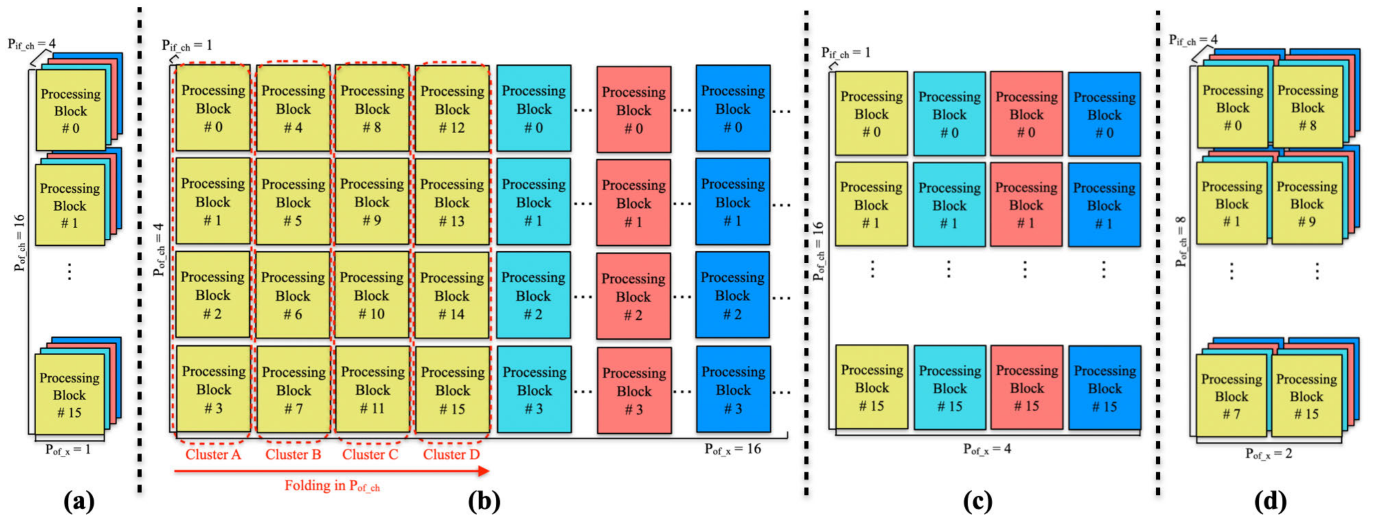

Fig. 4. PE reconfiguration examples for (a) Taxonomy① : $N_{if\_ch} \geq P_{if\_ch}$ and $N_{of\_ch} \geq P_{of\_ch}$, (b) Taxonomy② : $N_{if\_ch} < P_{if\_ch}$ and $N_{of\_ch} < P_{of\_ch}$, (c) Taxonomy③: $N_{if\_ch} < P_{if\_ch}$ and $N_{of\_ch} \geq P_{of\_ch}$, and (d) Taxonomy④: $N_{if\_ch} \geq P_{if\_ch}$ and $N_{of\_ch} < P_{of\_ch}$.

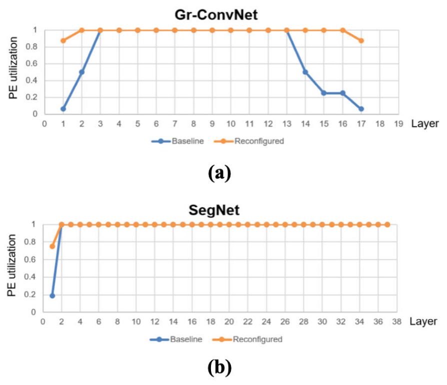

Fig. 5. PE utilization of different layers in both baseline mapping and the proposed reconfigured mapping for (a) Gr-ConvNet and (b) SegNet.

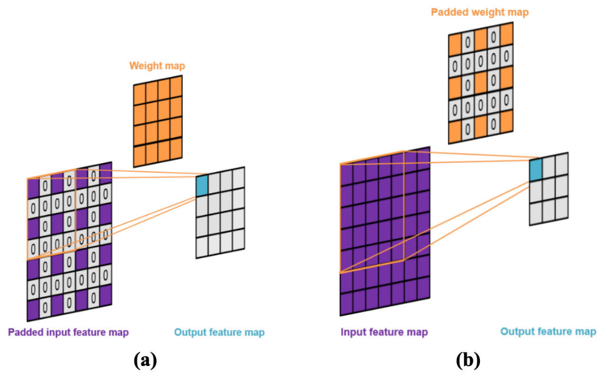

Fig. 6. Two emerging variants of convolutional operations: (a) Transposed convolution and (b) Dilated convolution.

a larger one, as shown in Fig. 6(a). Similarly, D-conv adds zero paddings between the weight pixels to enlarge the window of the weight kernel, as shown in Fig. 6(b). Conventional approaches [12], [13], [14], [16], [17], [18] that support T-conv and D-conv can be categorized into two methodologies: direct and convolution-like implementation. Direct implementations [12], [13], [16], and [17] employ an opposite computation flow to that of traditional convolution. The main design issue for this approach is to address the overlapping sum, which seriously lowers hardware utilization and efficiency. On the other hand, convolution-like implementations [14] and [18] use a similar processing flow to traditional convolution, enabling the possibility of a uniform architecture for T-conv and traditional convolution [14]. Reference [18] padded the input feature map with zeros between input pixels and implemented T-conv as traditional convolution. The main design issue for this approach is to avoid the redundant zero operations that degrade performance and efficiency. However, current approaches [14], [18] require either independent architecture or processing flow to support the computation of T-conv. Reference [18] employs separate architectures, limiting the supported weight shapes to 3×3 and 1×1. Reference [14] requires an extra convolution loop or decomposition process, which seriously lowers the computation efficiency.

In this section, a unified and efficient processing flow for both traditional and TD-convolution operations is proposed. Unlike previous works, the proposed approach efficiently bypasses all the redundant zeros operations without needing a separate architecture or processing flow. This significantly reduces the required operation number to maximize processing efficiency.

B. Proposed Unified Processing Flow

The proposed unified processing flow consists of two steps: row selection and intra-row sliding window step.

Step 1 (Row Selection Step): Both T-conv and D-conv have horizontal zero paddings either in input feature maps or weight maps when one output row is computed. Based on this observation, the main idea of the proposed design is to utilize a row selection module to choose the corresponding non-zero rows for only meaningful convolutions. The zero padding rows can thus be skipped to avoid the horizontal zero paddings.

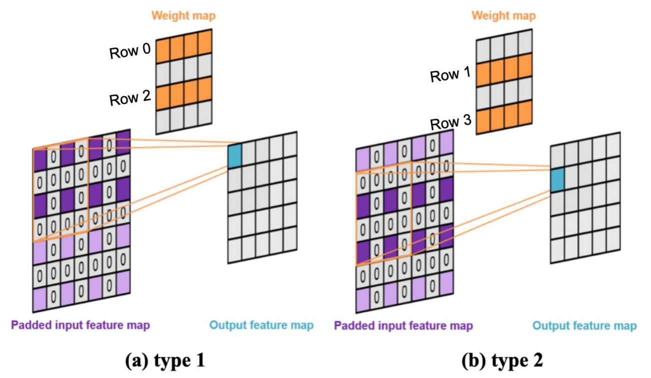

Fig. 7. Illustration of Step 1 for T-conv. Rows in dark purple represent the rows selected for computing one corresponding output row: (a) Rows 0 and 2 of the weight map are involved in non-zero operations for type 1, and (b) Rows 1 and 3 of the weight map are involved in non-zero operations for type 2.

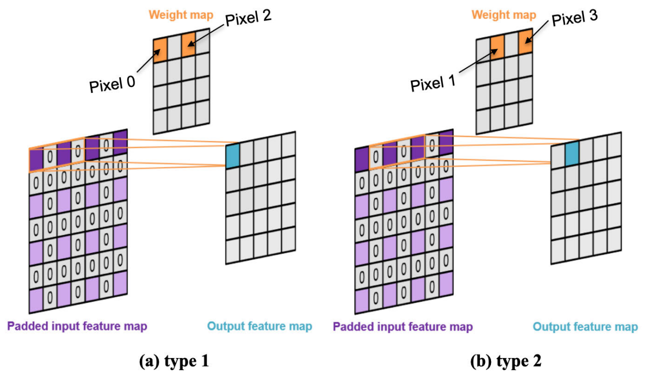

Fig. 8. Illustration of Step 2 for T-conv. Rows in dark purple represent the row selected for 1-D convolution: (a) Pixel 0 and 2 in Row 0 of the weight map are convolved with the non-zero input pixels in the first sliding window for type 1, and (b) Pixel 1 and 3 in Row 0 of the weight map are convolved with the non-zero input pixels in the second sliding window for type 2.

Fig. 7 illustrates an example of processing 4 $\times$ 4 T-conv with the proposed approach. All the row selection patterns for all output rows can be generalized into two types, either type 1 or type 2, according to the patterns that the weight kernel involves convolution with the entirely zero feature map rows (i.e., row 1 and row 3 can be skipped in type 1, while row 0 and row 2 can be skipped in type 2). Moreover, D-conv can be efficiently supported in the same manner. As a result, a unified row selection process for both T-conv and D-conv can be achieved to skip the horizontal padding rows without needing individual hardware resources.

Step 2 (Intra-Row Sliding Window Step): Once the selected rows from Step 1 are fetched, a 1-D convolution will be performed on the selected input row and weight row. Fig. 8 illustrates this step for T-conv. The same operation as that in step 1 is adopted at pixel level instead of row level to perform the 1-D sliding window operation. Again, a similar operation can be employed for D-conv processing. All vertical zero paddings are skipped, and only non-zero pixels are selected for meaningful convolution operation.

By combining Steps 1 and 2, all the zero paddings can be skipped entirely while the same processing flow as the traditional convolution can be kept. As a result, the operations

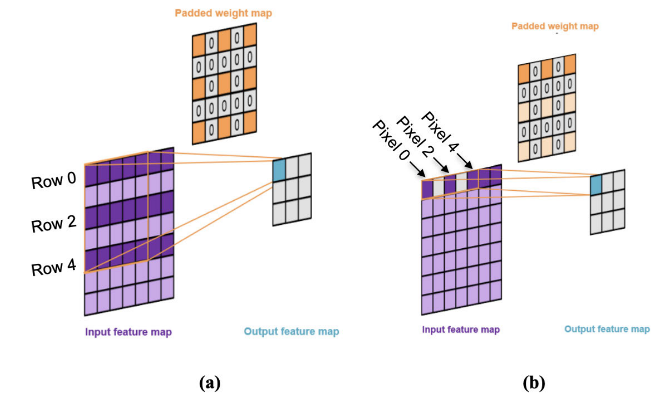

Fig. 9. Illustration of (a) Step 1 and (a) Step 2 for D-conv.

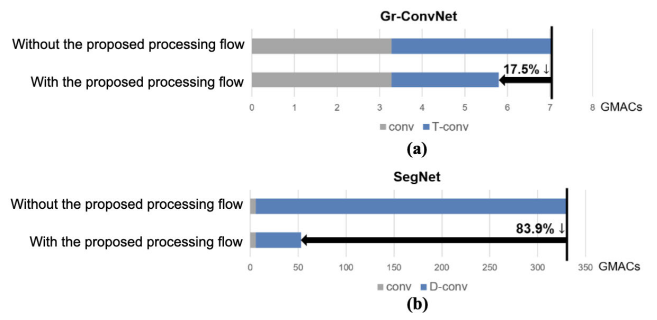

Fig. 10. Comparison of the operation number in GMACs for (a) Gr-ConvNet and (b) SegNet with and without the proposed processing flow.

of T-conv and D-conv can be efficiently supported with one unified processing engine without the need for separate hardware architecture or processing flow.

For D-conv, the flow is similar to that of T-conv. As shown in Fig. 9 (a), row 0, row 2, and row 4 of the input feature will not be convolved with the weight kernel rows that are entirely zeros, and hence are selected for step 2. In step 2, as depicted in Fig. 9 (b), pixel 0, pixel 2, and pixel 4 of the rows chosen are selected to complete the convolution operation.

C. Experimental Results

Compared to traditional convolution, both D-conv and T-conv involve more zero padding, which causes additional computation overhead and decreased efficiency. Fig. 10 shows the experimental results with and without the proposed unified processing flow for Gr-ConvNet and SegNet. For Gr-ConvNet, 53% of the operations are T-conv, and 33% of T-conv are redundant zero paddings. Regarding SegNet, 98% of the operations are D-conv, and 85% of D-conv are unnecessary zeros. By employing the proposed unified processing flow to skip the redundant zero operations, a significant reduction of 17.5% and 83.9% in operations can be achieved for Gr-ConvNet and SegNet, respectively.

IV. Proposed Accelerator Hardware Architecture

Fig. 11 shows an overview of the proposed accelerator hardware architecture. The pixel and weight buffers are constructed with two sets of 64 banks of on-chip global buffers

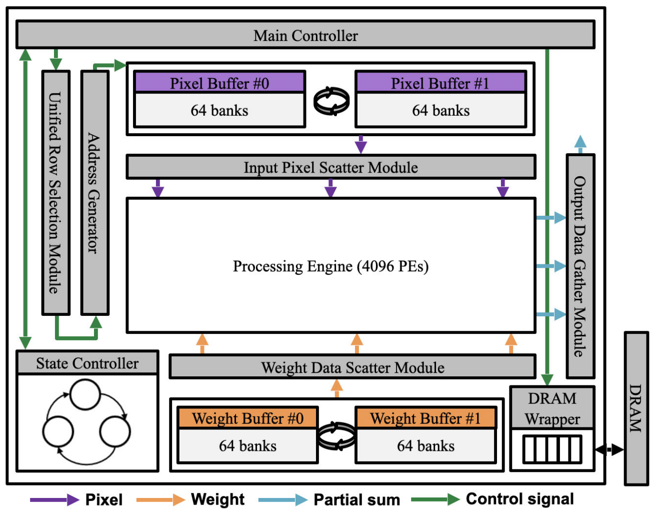

Fig. 11. Overall architecture of the proposed accelerator.

TABLE IV
SUMMARY OF DATA SCATTER AND GATHER MODE SELECTED IN DIFFERENT TAXONOMIES

Taxonomy	Data scatter mode	Data gather mode
①	Separate	Post-addition
②	Shared	Addition-free
③	Shared	Addition-free
④	Separate	Post-addition

TABLE V
RESOURCE UTILIZATION OF THE PROPOSED ACCELERATOR ON STRATIX 10 SOC FPGA

Type	Utilized	Available	Percentage
Logic utilization (in ALMs)	241972	933120	26%
Total DSP Blocks	4098	5760	71%
BRAM Blocks (M20K)	7120	11721	61%

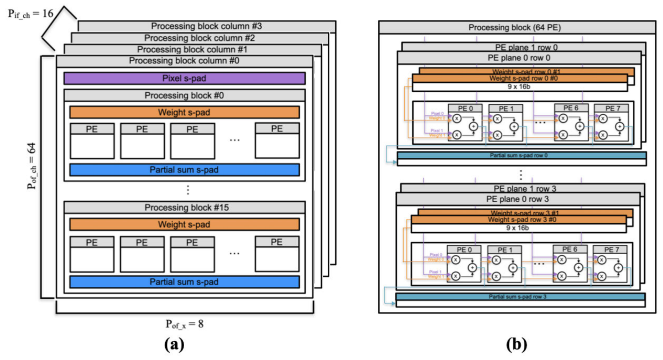

Fig. 12. The architecture of (a) processing engine and (b) processing block.

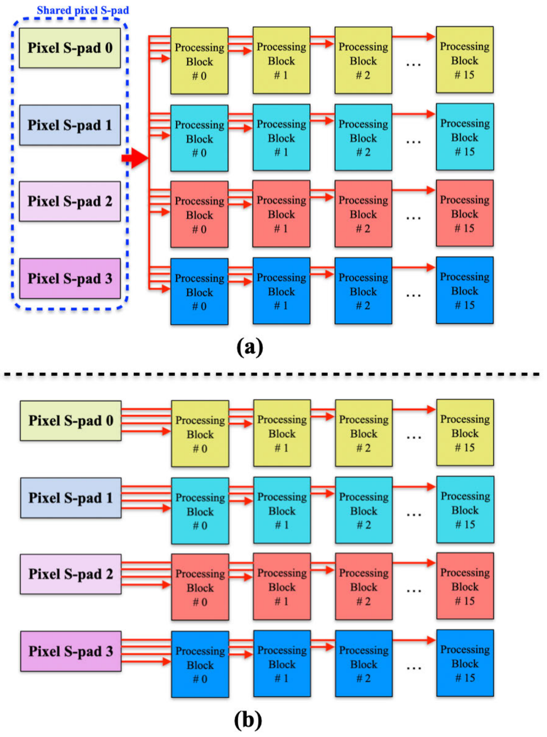

Fig. 13. Two types of modes for the data scatter module: (a) the shared mode and (b) the separate mode.

operating in a ping-pong mode. Row selection module and address generator are employed to calculate the feature map rows and pixels for the proposed unified processing flow described in Section III. This enables efficient support for both traditional and TD-conv computation while avoiding redundant zero operations.

The proposed processing engine shown in Fig. 12(a) implements the proposed 3-D PE mapping approach for efficient computation across different DNN models described in Section II. It consists of 4096 PEs and is partitioned into 64 processing blocks. Each processing block shown in Fig. 12(b) comprises 64 PEs with one set of shared local controllers. A processing block explores 2, 4, and 8 parallelisms in P_{if_ch}, P_{of_ch}, and P_{of_x} directions, respectively. Each PE implemented by the DSP block in FPGA is configured as two 18-bit $\times$ 19-bit multipliers followed by an accumulator that sums up the products generated by the two multipliers. The accelerator employs the proposed 3-D flexible PE mapping strategy to dynamically reconfigure the processing blocks at run time, enabling optimum parallelism selection and high PE utilization.

The input pixel and weight data scatter modules and the output data gather module, shown in Fig. 11, collaborate with the processing engine to support the proposed reconfiguration scheme. The data scatter module is responsible for communicating between the on-chip buffers and the local scratch pads (s-pads) in the processing engine. It realizes the desired configuration by distributing the corresponding input to the correct pixel s-pads. Taxonomy ② and ③, where N_{if_ch} is smaller than P_{if_ch}, require a data scatter module to operate in the shared mode, as shown in Fig. 13(a). In this mode, the folding is performed in the P_{if_ch} direction, so all the pixel s-pads are spliced into a large shared s-pad to store one row of the input feature map.

The other mode is the separate mode, where each pixel s-pad acts as an independent local buffer and stores its pixel row for

TABLE VI
COMPARISON WITH THE STATE-OF-THE-ART FPGA-BASED ACCELERATORS

Design	’22 TCAS-I [14]	’20 VLSI [5]	’20 TCAD [4]	Ours
FPGA Device (#available DSP block)	Arria 10 (1687)	XC7Z100 (2020)	Stratix 10 (5760)	Stratix 10 (5760)
DNN Model	VGG-16 / ENet	VGG-16	VGG-16 / ResNet-50	VGG-16 / ResNet-50 / ENet / GR-Conv Net / SegNet
Precision (bits)	8	8	16	16
Frequency (MHz)	200	200	300	190
#DSP blocks	607 (36%)	1986 (98%)	4108 (71%) / 3152 (55%)	4096 (71%)
Throughput (GOPS)	355 / 202	1218	1605 / 651	2766 / 2597 / 2648 / 2842 / 2870
GOPS/DSP	0.58 / 0.33	0.61	0.39 / 0.21	0.68 / 0.63 / 0.65 / 0.69 / 0.70
Support of TD-CONV	Y	N	N	Y
Energy efficiency (GOPS/W)	N.A.	68.8	N.A.	72.7
Operational efficiency (GOPS/MHz/#PE)	86.6% / 49.3%	74.4%	32.7% / 17.3%	88.9% / 83.4% / 85.1% / 91.3% / 92.2%

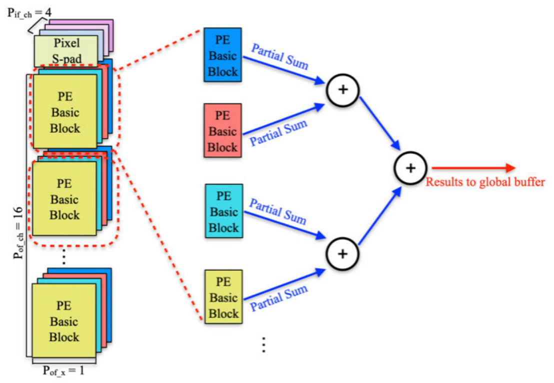

Fig. 14. The post-addition mode for the data gather module.

different input channels, as shown in Fig. 13(b). Taxonomy ① and ④ require a data scatter module to operate in this mode since no folding is performed in the P_{if_ch} direction.

The data gather module reads the partial sums from the partial sum s-pads inside the processing blocks and stores them back to global buffers. Similar to the data scatter module, two data gather modes, addition-free and post-addition modes, are employed according to the different taxonomy. For taxonomy ① and ④, the data gather module operates in the post-addition mode, as shown in Fig. 14. Since the proposed 3-D PE mapping approach includes parallelism in input feature channel dimensions, an adder tree is required to add up the partial sums generated from all the unrolled input channels before being pushed back to the global buffer.

On the other hand, since the redundant parallelism in the input feature channel dimension is allocated in the output feature dimension in taxonomy ② and ③, no further addition operation is required with $P_{if\_ch} = 1$. As a result, the data gather module operates in the addition-free mode. The partial sums are read from processing blocks and directly stored back in the global buffer without going through the adder tree. Table IV summarizes the corresponding data scatter and gather modes for different selected taxonomies. By using the proposed processing engine, data scatter and gather modules, the proposed 3-D PE mapping approach can be efficiently implemented to realize high-performance operations across different DNN models.

V. IMPLEMENTATION AND EXPERIMENTAL RESULT

A. Hardware Implementation

The proposed accelerator architecture is implemented on Intel Stratix 10 SoC FPGA. In the beginning, the host will transmit parameters defining the shape and size of the DNN model to the FPGA. After that, the host will transmit the feature maps for convolution operations to the FPGA, while the results processed by the accelerator will be transmitted back to the host after the computation is completed. The system is coded with Verilog, simulated with Cadence Xcelium Logic Simulator, and compiled by Quartus Prime Pro. The available resources and utilization data of the Stratix 10 FPGA are listed in Table V. All the PEs in the proposed design are implemented with the DSP blocks. The available DSP resource on FPGA thus determines the maximum number of PE that can be realized in this design.

B. Experimental Result and Comparison

Table VI summarizes the experimental results compared to the state-of-the-art FPGA-based accelerators. The proposed design achieves 2.597–2.870 TOPS throughput performance with an efficiency of 0.63-0.70 GOPS/DSP across various DNN networks, substantially outperforming the previous works. By employing the proposed 3-D mapping approach, the design can maintain high operational efficiency under a wide range of DNN models. Compared to [4] on the same hardware implementation platform, the proposed design achieves significant throughput improvement of 1.72× and 3.99× in VGG-16 and ResNet-50, respectively. Furthermore, the design can support efficient processing of both T-conv and D-conv with the proposed unified architecture, which ensures high throughput and energy efficiency across different convolution operations. Compared to [14] executing ENet that contains both T-conv and D-conv operations, the proposed design realizes 1.73× higher operational efficiency. Moreover, the significantly improved Throughput/DSP performance in both VGG-16 and ENet clearly demonstrates the advantages of the proposed unified processing flow.

VI. Conclusion

This work proposes a highly efficient and flexible FPGA-based accelerator for DNN inference. A reconfigurable 3-D PE mapping approach that realizes an optimum parallelism strategy based on the actual DNN characteristics is proposed to deal with the severe utilization degradation issue with a large PE array. The proposed approach can effectively maintain high PE utilization of accelerator across different DNN models. Moreover, a unified processing flow for both traditional and TD-conv operations is proposed. The accelerator with the proposed processing flow can efficiently skip all the redundant zero operations without needing separate architecture or processing flow. The proposed accelerator with 4096 PEs is implemented on Intel Stratix 10 SoC FPGA. It achieves 2.597–2.870 TOPS throughput performance with 83.4%–92.2% operational efficiency across various DNN networks, including VGG-16, ResNet-50, ENet, Gr-ConvNet, and SegNet. The accelerator with the proposed 3-D PE mapping and unified processing flow demonstrates 1.72× and 1.73× higher throughput and efficiency performance, respectively, compared to the state-of-the-art designs.

Acknowledgment

The authors would like to thank Bing-Chen Wu, Chieh-Fu Tsai, and Hongming Chen for valuable technical discussions.

References

[1] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2014, *arXiv:1409.1556*.

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.

[3] V. Badrinarayanan, A. Kendall, and R. Cipolla, “SegNet: A deep convolutional encoder-decoder architecture for image segmentation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017.

[4] Y. Ma, Y. Cao, S. Vrudhula, and J.-S. Seo, “Automatic compilation of diverse CNNs onto high-performance FPGA accelerators,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 2, pp. 424–437, Feb. 2020.

[5] Y. Yu, C. Wu, T. Zhao, K. Wang, and L. He, “OPU: An FPGA-based overlay processor for convolutional neural networks,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 1, pp. 35–47, Jan. 2020.

[6] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, “Optimizing FPGA-based accelerator design for deep convolutional neural networks,” in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, Feb. 2015, pp. 161–170.

[7] S. Williams, A. Waterman, and D. Patterson, “Roofline: An insightful visual performance model for multicore architectures,” *Commun. ACM*, vol. 52, no. 4, pp. 65–76, Apr. 2009.

[8] J. Qiu et al., “Going deeper with embedded FPGA platform for convolutional neural network,” in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, Feb. 2016, pp. 26–35.

[9] H. Li, X. Fan, L. Jiao, W. Cao, X. Zhou, and L. Wang, “A high performance FPGA-based accelerator for large-scale convolutional neural networks,” in *Proc. 26th Int. Conf. Field Program. Log. Appl. (FPL)*, Aug./Sep. 2016, pp. 1–9.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, vol. 25, no. 2, pp. 1097–1105.

[11] Y. Ma, Y. Cao, S. Vrudhula, and J.-S. Seo, “Performance modeling for CNN inference accelerators on FPGA,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 4, pp. 843–856, Apr. 2020.

[12] X. Zhang, S. Das, O. Neopane, and K. Kreutz-Delgado, “A design methodology for efficient implementation of deconvolutional neural networks on an FPGA,” 2017, *arXiv:1705.02583*.

[13] D. Wang, J. Shen, M. Wen, and C. Zhang, “Towards a uniform architecture for the efficient implementation of 2D and 3D deconvolutional neural networks on FPGAs,” 2019, *arXiv:1903.02550*.

[14] X. Wu, Y. Ma, M. Wang, and Z. Wang, “A flexible and efficient FPGA accelerator for various large-scale and lightweight CNNs,” *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 69, no. 3, pp. 1185–1198, Mar. 2022.

[15] Y.-H. Chen, T. Krishina, J.-S. Emer, and V. Sze, “Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks,” *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Nov. 2016.

[16] S. Liu and W. Luk, “Towards an efficient accelerator for DNN-based remote sensing image segmentation on FPGAs,” in *Proc. 29th Int. Conf. Field Program. Log. Appl. (FPL)*, Sep. 2019, pp. 187–193.

[17] W. Mao, J. Wang, J. Lin, and Z. Wang, “Methodology for efficient reconfigurable architecture of generative neural network,” in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2019, pp. 1–5.

[18] D. Im, D. Han, S. Choi, S. Kang, and H.-J. Yoo, “DT-CNN: An energy-efficient dilated and transposed convolutional neural network processor for region of interest based image segmentation,” *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 10, pp. 3471–3483, Oct. 2020.

[19] S. Kumra, S. Joshi, and F. Sahin, “Antipodal robotic grasping using generative residual convolutional neural network,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 9626–9633.

[20] M. Gao, X. Yang, J. Pu, M. Horowitz, and C. Kozyrakis, “TANGRAM: Optimized coarse-grained dataflow for scalable NN accelerators,” in *Proc. 24th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Apr. 2019, pp. 807–820.

Tsung-Hsi Wu received the B.S. degree in electrical engineering from National Cheng Kung University, Tainan, Taiwan, in 2020, and the M.S. degree in electronics engineering from National Taiwan University, Taipei, Taiwan, in 2022. His research interests include efficient and flexible neural network accelerator.

Chang Shu received the B.S. degree in electronics engineering from Feng Chia University, Taichung, Taiwan, in 2020, and the M.S. degree in electronics engineering from National Taiwan University, Taipei, Taiwan, in 2023. He is currently pursuing the Ph.D. degree with the Graduate Institute of Electronics Engineering, National Taiwan University. He is also an FPGA Engineer at Hikvision, Hangzhou, China. His research interests include efficient and flexible neural network accelerator.

Tsung-Te Liu (Member, IEEE) received the B.S. degree in electrical engineering and the M.S. degree in electronics engineering from National Taiwan University, Taipei, Taiwan, in 2002 and 2004, respectively, and the Ph.D. degree in electrical engineering from the University of California at Berkeley, Berkeley, CA, USA, in 2012.

From 2004 to 2005, he was with MediaTek Inc., Hsinchu, Taiwan, where he was involved in circuit and system design for wireless communications. From 2005 to 2012, he was a member of Berkeley Wireless Research Center (BWRC), University of California at Berkeley. From 2012 to 2014, he was with the Interuniversity Microelectronics Center (IMEC), Leuven, Belgium, where he conducted research on circuit development for advanced CMOS technology. In 2014, he joined National Taiwan University, where he is currently a Professor with the Graduate Institute of Electronics Engineering and the Department of Electrical Engineering. His research interests include energy-efficient circuits and system designs. He was a recipient of several design and teaching awards.