Accelerating Frequency-domain Convolutional Neural Networks Inference using FPGAs

Yi Chen[1], Bosheng Liu[1], Yongqi Xu[1], Jigang Wu[1*] , Xiaoming Chen[2], Peng Liu[1], Qingguo Zhou[3], Yinhe Han[2]
[1] School of Computer Science and Technology, Guangdong University of Technology, Guangzhou, China
[2] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[3] School of Information Science and Engineering, Lanzhou University, Lanzhou, China

***Abstract*—Low-end field programmable gate arrays (FPGAs) are difficult to deploy typical convolutional neural networks (CNNs) owing to the limited hardware resources and the increasing model computational complexity. Fast Fourier transform (FFT) is a promising solution for saving both computation and memory footprint by convolving in the frequency domain. However, few FPGA accelerators can take full advantage at the computation level, because of the distinct element-wise complex calculation in the frequency domain. In this work, we present an FPGA-based 8-bit inference accelerator (called FAF) that packs frequency-domain calculations into digital signal processing (DSP) blocks to fully utilize DSPs for performance boost. We then provide a mapping dataflow to maximize the reduction of redundant packing operations by frequency-domain data reuse. Evaluations based on representative CNN benchmarks show that our work can achieve 1.5-6.9× better power efficiency compared with representative FPGA baselines.**

***Index Terms*—Convolutional neural network, frequency-domain, FPGA, hardware accelerator**

I. Introduction

Convolutional neural networks (CNNs) have made great achievements in many computer vision tasks [1]–[3]. Recently, 8-bit CNN inference becomes a popular approach for embedded mobile devices owing to its superior capability to reduce both model sizes and memory requirements. For example, 8-bit CNN inferences have been widely supported by deep learning frameworks (e.g., TensorFlow, PyTorch, NVIDIA TensorRT, and Xilinx DNNDK) [4]–[6]. However, state-of-the-art CNNs have significantly deeper layers and higher computation complexity, which make them hard to be deployed onto resource-limited embedded mobile devices [7]. Fast Fourier transform (FFT) [8], [9] is one of the promising solutions that can significantly save computation by replacing original spatial convolutions with simpler frequency-domain element-wise complex multiplications. It is observed that modern CNNs tend to be over-parameterized, and a large portion of redundant both computation and memory access can be replaced by 8-bit element-wise complex calculations, therefore reducing the computational complexity and heavy cost of CNN inference on embedded devices [10], [11].

FPGA-based accelerators have attracted significant attention for CNN inference owing to their great advantages of reducing both time-to-market and design costs [12], [13]. Prior work [14] provides efficient 8-bit FPGA-based CNN accelerations in spatial convolution. Nevertheless, they cannot fully exploit the benefit of digital signal processing (DSP) [15] resources and the frequency-domain reduced computation complexity. To utilize the reduced computation complexity, some FPGA accelerators focus on frequency-domain CNN accelerations by flexible data parallelism [8], data locality [9], and block convolutions (such as overlap-save [10]). Nevertheless, they suffer from significant high memory footprints and bandwidth bottlenecks because of the high-precision calculations [16]. Coupling 8-bit calculation into DSPs can improve throughput but mainly face two challenges in frequency domain: (1) It is difficult to deploy more than one 8-bit frequency-domain complex input into the same DSP block because of the complicated calculation pattern of frequency-domain convolution (i.e., each complex input has real and imaginary parts, and each complex multiplication consists of three real number multiplications); And (2), directly mapping coupled complex inputs to DSPs will incur significant redundant packing operations, because of the massive amounts of complex multiplications in frequency-domain CNNs. For example, it requires 1.1 Giga of packing operations for Resnet18 in frequency domain, which is a significantly high cost in hardware resources.

To address such challenges, we provide an FPGA-based accelerator (called FAF), which efficiently processes CNNs in the 8-bit frequency domain for the inference phase. Specifically, we build a DSP-based frequency-domain complex multiplication unit that can pack a couple of 8-bit complex multiplications into one DSP for high throughput. We then provide a dataflow to fully exploit the reusable coupled inputs to reduce the redundant packing operations for energy saving. In summary, this paper makes the following contributions,

- We propose an FPGA-based hardware architecture for accelerating CNN inference in frequency domain.
- We develop the DSP-based complex multiplication unit and mapping dataflow to couple complex multiplication into DSP blocks and to exploit the data reuse for high throughput and data movement reduction.
- Evaluation results based on Xilinx Ultra96-V2 FPGA and two representative benchmarks (Resnet18 and VGG16) show that compared with a state-of-the-art FPGA spatial accelerator baseline [14], FAF achieves 2.4× higher power efficiency. Compared with frequency-domain representative FPGA accelerator baselines (i.e., [8]–[10]), FAF gains 1.5–6.9× higher power efficiency under less than 1.6% accuracy loss.

II. Proposed FAF Accelerator

A. FAF Architecture

Fig. 1 outlines the proposed FAF architecture. FAF mainly consists of four components: on-chip buffers (ABin, ABout, LB, and WB), FFT/iFFT, Ctl, and FPE. The on-chip buffers consist of block random access memory (BRAM). ABin and

*Corresponding author(asjgwucn@outlook.com)

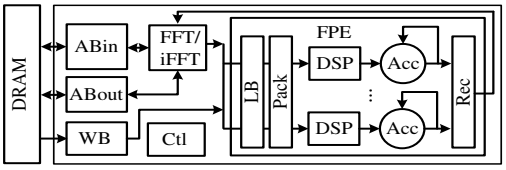


Fig. 1. Architecture of FAF.

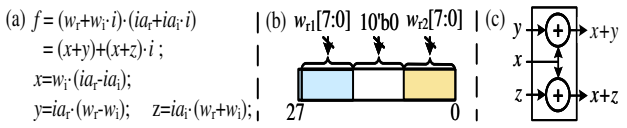


Fig. 2. Pack and Rec components in FPE for packing inputs and recovering outputs based on the element-wise multiplication in subfigure (a), respectively. (b) Pack. And (c) Rec components.

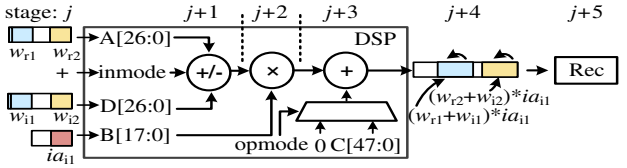


Fig. 3. FPE datapath illustration through an example of calculating $(w_{\mathrm{r}1}+w_{\mathrm{i}1})\cdot ia_{\mathrm{i}1}$ and $(w_{\mathrm{r}2}+w_{\mathrm{i}2})\cdot ia_{\mathrm{i}1}$ based on Fig. 2(a).

ABout alternately accommodate input/output activations. LB is a local buffer for the generated complex input activations. WB accommodates frequency-domain complex weights. FFT/iFFT performs data transformation between spatial and frequency domains. FPE performs element-wise complex multiplication based on coupled inputs. FPE includes Pack, DSP, Acc, and Rec components. The Pack component couples complex inputs into groups for frequency-domain DSP calculation. Acc accumulates the outputs. Rec recovers the real and imaginary parts of complex output activations. Ctl is a central controller that triggers all components to work correctly.

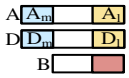
Fig. 2 depicts the Pack and Rec components of FPE based on the basic element-wise complex multiplication of subfigure (a). Each complex input has real and imaginary parts. For example, complex input activations ($(ia_{\mathrm{r}}+ia_{\mathrm{i}}\cdot i)$) and complex weights ($(w_{\mathrm{r}}+w_{\mathrm{i}}\cdot i)$) respectively have two items (ia_r and ia_{i}, w_r and w_{i}), referring to the real and imaginary parts. i denotes an imaginary unit. Fig. 2(b) depicts the Pack component, which receives two 8-bit data (i.e., $w_{\mathrm{r}1}$ and $w_{\mathrm{r}2}$) as inputs, and then integrates them into a long bit-width register (e.g., 27-bit registers). Fig. 2(c) depicts the Rec component, which recovers the real and imaginary parts of a complex output activation based on Fig. 2(a).

Frequency-domain Multiplication Unit. Fig. 3 depicts the datapath of FPE. Each FPE completes the multiplication in six stages (i.e., from stage j to $j+5$) based on an example of Fig. 2(a) (i.e., $(w_{\mathrm{r}1}+w_{\mathrm{i}1})\cdot ia_{\mathrm{i}1}$ and $(w_{\mathrm{r}2}+w_{\mathrm{i}2})\cdot ia_{\mathrm{i}1}$, where $<w_{\mathrm{r}1}, w_{\mathrm{i}1}>$ and $<w_{\mathrm{r}2}, w_{\mathrm{i}2}>$ are complex weights for different output channels. $ia_{\mathrm{i}1}$ is the imaginary part of a complex input activation). Specifically, at stage j, two groups of 8-bit weights ($(w_{\mathrm{r}1}, w_{\mathrm{r}2})$ and $(w_{\mathrm{i}1}, w_{\mathrm{i}2})$) are respectively packed into A and D registers for calculation. At stages $j+1$ and $j+2$, DSP respectively performs addition and multiplication operations for the coupled inputs. Since $w_{\mathrm{r}2}$ and $w_{\mathrm{i}2}$ are considered as

Algorithm 1: Generation of C for DSP

Input: A_{l}, D_{l}, B, *inmode* // A, D, B, *inmode* are inputs of DSP.
//A_{l} and D_{l} are the least significant 8 bits of A and D.
Output: C

1 **if** *inmode* is "+" **then**
2 **if** (A_{l}<0 and D_{l}<0) **then**
3 C=-(B<<9); //B is shifted left by 9 bits, and then reversed
4 **else if** (A_{l}<0 and D_{l}>=0) or (A_{l}>=0 and D_{l}<0) **then**
5 C=-(B<<8); //B is shifted left by 8 bits, and then reversed
6 **else** //A_{l}>=0 and D_{l}>=0
7 C=0;
8 **else** //inmode is "-"
9 **if** (A_{l}<0 and D_{l}<0) **then** C=0;
10 **else if** (A_{l}<0 and D_{l}>=0) or (A_{l}>=0 and D_{l}<0) **then**
11 **if** (A_{l}<0 and D_{l}>=0) **then** C=-(B<<8);
12 **else** C=B<<8;
13 **else** //(A_{l}>=0 and D_{l}>=0)
14 C=0;



unsigned inputs after being coupled into the least significant 8 bits of registers, it requires conversion from unsigned to signed bit calculation for the final output ($(w_{\mathrm{r}2}+w_{\mathrm{i}2})\cdot ia_{\mathrm{i}1}$). We add a bias (identified as C) at stage $j+3$ for the conversion, which comes from the C register of DSP blocks. C is generated based on Equation (1) and Algorithm 1, which is exhibited in the following two paragraphs. At stages $j+4$ and $j+5$, FPE accumulates the temporal results and recovers the final complex outputs.

Equation (1) depicts the generation of C for correcting the product result for the least significant 8-bit coupled inputs.

$$\begin{aligned} C &= (A_l \ inmode \ D_l)\cdot B-(A_{ul} \ inmode \ D_{ul})\cdot B \\ &= -(a_{n-1} \ inmode \ d_{n-1})\cdot B\cdot 2^n \end{aligned} \quad (1)$$

where A_l and D_l are respectively the least significant 8 bits of A and D (A_l and D_l are signed numbers). A_{ul} and D_{ul} are respectively the corresponding unsigned inputs. a_{n-1} and d_{n-1} are the signed bits of A_l and D_l, respectively. B refers to the signed multiplicand input.

Algorithm 1 exhibits the operations for generating C, according to the parameters of $inmode$ and the signed bit of both A_l and D_l. Specifically, when *inmode* is "+", referring to the addition between A_l and D_l, C is generated in three situations (lines 2-7 of Algorithm 1): First, if both A_l and D_l are negative ($a_{n-1}=1$ and $d_{n-1}=1$, because "1" refers to a negative data and "0" identifies a positive one), $C=-(B<<9)$ based on Equation (1); Second, if either A_l or D_l is negative, $C=-(B<<8)$ based on Equation (1); Finally, if both are positive, C equals zero without modification. Similarly, when *inmode* is "-", referring to the subtraction between A_l and D_l, C is generated based on lines 9-14 according to Equation (1).

B. Mapping Dataflow

Fig. 4 describes the dataflow based on a complete element-wise complex multiplication on FPE. Figs. 4 (a) and (b) illustrate the utilized overlap-save based frequency-domain convolution. ia_{fft}, oa_{fft}, and w_{fft} are frequency-domain complex input activations, output activations, and weights, respectively. The edge size of ia_{fft} is 3. ia_{fft} has four tiles, and each tile has a size of 2. The input channel is 2. oa_{fft} has two output channels. Each complex number of ia_{fft}, w_{fft}, and oa_{fft} has two items, referring to the real and imaginary parts. Fig. 4(b) exhibits the required calculation corresponding to the final outputs for complex activations. To clearly exhibit the calculation

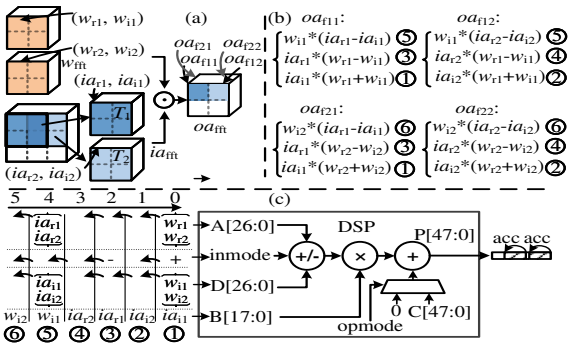


Fig. 4. Mapping dataflow. (a) Illustration of frequency-domain convolutions. (b) Basic operations for calculations. And (c) computing process.

TABLE I
STATISTIC NUMBER OF MULTIPLICATIONS IN BENCHMARKS.

CNNs	Resnet18		VGG16	
Convolutions	Spatial	Freq.-dom.	Spatial	Freq.-dom.
# of Mult. (GOP)	1.79	1.24	15.35	5.99

TABLE II
CONFIGURATIONS OF FAF.

Frequency (MHz)	200	# of FPEs	96
FFT size	8	FFT Parallel	16
WB Buffer (MB)	0.13	ABin/ABout/LB Buffer (MB)	0.1

sequence of the dataflow, we mark the calculation phases from ① to ⑥. Fig. 4(c) depicts the mapping dataflow. To maximize data reuse, we first pack the complex weights because they occupy the maximum ratio (66.7%) of the addition/subtraction operations (Fig. 4(b)). For example, the packed weight vectors $\{w_{\mathrm{r}1}, w_{\mathrm{r}2}\}$ and $\{w_{\mathrm{i}1}, w_{\mathrm{i}2}\}$ (corresponding to sequence ①) at execution cycle 0 can be reused from cycles 1 to 3 under the configuration of "*inmode*". Furthermore, since the addition/subtraction engine generates the same output at both cycles 0 and 1, and both cycles 2 and 3, we can avoid utilizing the addition/subtraction engine at execution cycles 1 and 3 by reusing the generated results at prior execution cycles. After we have reused the coupled complex weights, we couple the complex activations for calculations, which are conducted at execution cycles 4 and 5. Specifically, the packed complex activations vectors $\{ia_{\mathrm{r}1}, ia_{\mathrm{r}2}\}$ and $\{ia_{\mathrm{i}1}, ia_{\mathrm{i}2}\}$ at execution cycle 4 can be reused at execution cycle 5 because they utilize the same complex input activations. Similarly, at execution cycle 5, we can reuse the addition results from the addition/subtraction engine at the prior execution cycle.

Data Reuse. The coupled inputs are reused in three folds. First, the coupled complex weights from different output channels can be reused by ia_{fft} from both the real/imaginary parts and different input tiles. For example, the packed weight vectors $\{w_{\mathrm{r}1}, w_{\mathrm{r}2}\}$ and $\{w_{\mathrm{i}1}, w_{\mathrm{i}2}\}$ for $ia_{\mathrm{i}1}$ at execution cycle 0 can be reused by $ia_{\mathrm{i}2}$ at execution cycle 1, where $ia_{\mathrm{i}1}$ and $ia_{\mathrm{i}2}$ belong to different tiles of input activations. The same coupled weights at execution cycle 0 for $ia_{\mathrm{i}1}$ can also be reused by $ia_{\mathrm{r}1}$ under the configuration of *inmode*, where $ia_{\mathrm{r}1}$ and $ia_{\mathrm{i}1}$ are the real and imaginary parts of an ia_{fft}. Second, the coupled activations from different tiles can be reused by complex weights from different output channels. For example, the packed complex activations vectors $\{ia_{\mathrm{r}1}, ia_{\mathrm{r}2}\}$ and $\{ia_{\mathrm{i}1}, ia_{\mathrm{i}2}\}$ at execution cycle 4 for $w_{\mathrm{i}1}$ can be reused at execution cycle 5 for $w_{\mathrm{i}2}$, where $w_{\mathrm{i}1}$ and $w_{\mathrm{i}2}$ are the imaginary parts of complex weights for different output channels. In addition, the generated temporal results inside FPE can also be reused for energy saving. For example, the results of the addition/subtraction engine at execution cycles 0 and 2 can be reused at execution cycles 1 and 3, respectively. In this case, we can save the calculation by data reuse.

III. EVALUATION

Baselines. We evaluate the proposed accelerator in two steps. First, we compare our work with the state-of-the-art FPGA-based 8-bit spatial convolution accelerator baseline [14]. Ref. [14] is an efficient FPGA-based CNN accelerator with fully exploiting data reuse and resource utilization for 8-bit spatial convolutions. Second, since there is no prior FPGA-based 8-bit frequency-domain accelerator, we evaluate our work by comparing it with high-precision frequency-domain representative baselines [8]–[10]. Specifically, Ref. [8] exploits different data parallelism to improve the throughput. Ref. [9] exploits data locality to reduce communication overhead. Ref. [10] provides an overlap-save based frequency-domain convolution accelerating solution to reduce data movements.

For a comprehensive evaluation, we exploit three major accelerator configurations to exhibit the effect of using LUTs and DSPs for FPEs: (a) The FPEs of our work are implemented with LUTs without packing inputs, denoted by **FAF-lut**; (b) The FPEs of our work are implemented with DSP blocks without coupling 8-bit inputs for calculation, denoted by **FAF-dsp-basis**; (c) The FPEs of our work are implemented with DSP blocks, which provide the capability to couple two groups of 8-bit complex multiplications into one DSP for throughput boost, denoted by **FAF**.

Benchmarks and Configurations. We evaluate the proposed FPGA-based accelerator based on two representative CNNs (Resnet18 [2] and VGG16 [3]) with the ImageNet dataset [17]. Both Resnet18 and VGG16 are two representative CNNs. Table I statistics the number of multiplication for the benchmarks in both spatial and frequency domains. The accuracy loss of the benchmarks under 8-bit frequency-domain convolutions is evaluated by comparing them with the benchmarks in spatial domain. We conduct our evaluations based on Xilinx Ultra96-V2 FPGA [18], which includes DSP48E2 DSP blocks. We implement the behavior of our design by Verilog and synthesize the hardware implementation based on Xilinx Vivado Design Suite. Table II shows the configuration of FAF.

Comparison with State-of-the-art Accelerators. Table III compares FAF with representative FPGA-based both spatial and frequency-domain CNN solutions. Compared with the FPGA-based 8-bit spatial CNN accelerator baseline [14], FAF has 2.4× higher throughput under with 1.01× scale of multipliers. The higher throughput comes from the efficient utilization of DSP and the reduced computation complexity in the frequency domain. Compared with frequency-domain representative baselines [8]–[10], FAF achieves 1.5–6.9× higher power efficiency. The higher power efficiency comes from

TABLE IV
COMPARISON UNDER DIFFERENT CONFIGURATIONS.

	FAF-lut		FAF-dsp-basis		FAF	
CNNs	Resnet	VGG	Resnet	VGG	Resnet	VGG
FF Usage (%)	26.08	27.13	25.07	26.0	26.31	27.30
LUT Usage (%)	49.39	51.04	38.53	39.48	40.59	42.17
DSP Usage (%)	21.11	21.94	47.78	48.61	47.78	48.61
BRAM	419	419	419	419	419	419
Performance (GOP/s)	56.94	99.24	56.94	99.24	113.68	198.14
Power (W)	3.143	3.1	2.895	2.944	2.979	3.054
Power efficiency (GOP/s/W)	18.12	32.01	19.66	33.71	**38.16**	**64.87**

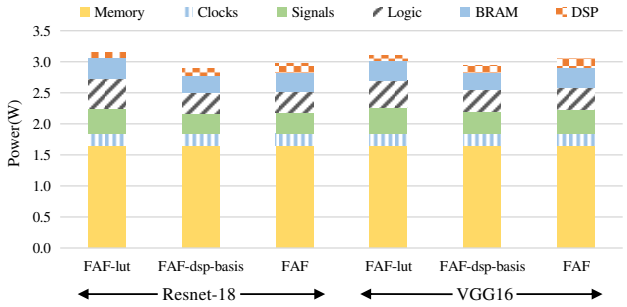


Fig. 5. Power breakdown under three configurations and two benchmarks.

three folds: first, packing frequency-domain 8-bit inputs for DSP blocks can boost the throughput and save both memory access and computation overhead; second, FAF can exploit the reusable coupled frequency-domain inputs for energy saving; third, FAF can save power from the reduced addition/subtraction calculation in the addition/subtraction engine of DSPs by reusing prior calculation results.

Performance and Power Efficiency Breakdown. Table IV shows the performance and power efficiency breakdown under three accelerator configurations. We can observe that: (1) Packing 8-bit complex multiplications into DSPs can boost power efficiency. The higher power efficiency comes from the boosted throughput and lower power consumption. For example, compared with FAF-lut, FAF achieves 2.1× and 2.0× better power efficiency on Resnet18 and VGG16 benchmarks, respectively; (2) The overhead for packing groups of inputs for DSP blocks is small. For example, compared with FAF-dsp-basis, FAF consumes a bit more LUT resources (2.1% and 2.8% for Resnet18 and VGG16, respectively). Nevertheless, compared with FAF-dsp-basis, FAF achieves 2.0× higher performance by fully utilizing DSP blocks; (3) Compared with LUT resources, the utilization of DSP blocks consumes a bit less power for calculation. For example, FAF-dsp-basis has a bit less power consumption than FAF-lut on Resnet18 and VGG16 benchmarks (0.92× and 0.95×, respectively), owing to the advantage of customized hardware components.

TABLE V
INFERENCE ACCURACY (TOP-5) COMPARISON FOR BENCHMARKS.

	Spatial (32-bit float)	Freq.-dom. (32-bit float)	Freq.-dom. (8-bit fixed)
Resnet-18	98.88%	98.02%	96.84%
VGG16	94.48%	93.87%	93.39%
Average	96.66%	95.92%	95.10%

Fig. 5 shows the power breakdown under three accelerator configurations. Specifically, memory access dominates the total power, occupying 53.0%, 56.7%, and 54.8% of the total power on FAF-lut, FAF-dsp-basis, and FAF, respectively. Therefore, it is critical to utilize 8-bit frequency-domain complex multiplications for calculating CNNs in embedded mobile devices to save the memory footprint and energy consumption.

Tradeoff Analysis. Table V shows the accuracy performance for both spatial and frequency-domain models. Compared with original spatial benchmarks (32-bit float), frequency-domain 8-bit complex multiplication can achieve convincing accuracy performance with on average 1.6% accuracy loss, because of the reduced data width representation. Nevertheless, the reduced calculation precision contributes to significant memory footprint and computation saving. In addition, when the data width representation is reduced from 32-bit to 8-bit in frequency domain, it incurs 0.8% accuracy loss because of the powerful capability of CNNs.

IV. CONCLUSION

This work introduces a frequency-domain 8-bit FPGA CNN inference accelerator, which couples 8-bit frequency-domain multiplications into DSP blocks for high throughput. We exploit frequency-domain data reuse to maximize the reduction of redundant coupling operations to save both data movements and computation overhead. Comprehensive evaluations show that our work achieves 1.5-6.9× better power efficiency compared with FPGA-based state-of-the-art solutions.

TABLE III
CHARACTERISTICS OF RECENT SOLUTIONS.

	[14]	[8]	[10]	[9]	ours	
Device	ZYNQ7020	ZYNQ7020	ZC706	Intel QuickAssist QPI FPGA	Ultra96-v2 board	
Spatial/Freq.-dom.	Spatial	Freq.-dom.	Freq.-dom.	Freq.-dom.	Freq.-dom.	
Frequency (MHz)	214	154	166	200	200	
Precision	8-bit fixed	32-bit float	16-bit fixed	32-bit float	8-bit fixed	
Multiplier	190	-	480	192	192	
CNNs	VGG16	Resnet18	VGG16	VGG16	Resnet18	VGG16
DSP Utilization	190(86.4%)	-	900(100%)	224(87.5%)	172(47.8%)	175(48.6%)
Logic Utilization	29.867K(56%)	-	350K(46%)	200.5K(85.5%)	28.640K(41%)	29.541K(41.87%)
BRAM	171(61%)	-	1090(73%)	910(64%)	419(96.9%)	419(96.9%)
Performance(GOP/s)	84.3	80.95	277.8	123.48	113.68	198.14
Power(W)	3.5	3.26	9.4	13.18	2.979	3.054
Power efficiency(GOP/s/W)	24.1	24.83	29.4	9.37	**38.16**	**64.87**
Resource Efficiency (GOP/s/Multiplier)	0.44	-	0.58	0.64	**0.59**	**1.03**

REFERENCES

[1] R. Andri, B. Bussolino, A. Cipolletta, L. Cavigelli, and Z. Wang, "Going further with winograd convolutions: Tap-wise quantization for efficient inference on 4x4 tiles," in *Proceeding of the IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2022, pp. 582–598.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2016, pp. 770–778.

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

[4] R. T. Syed, M. Andjelkovic, M. Ulbricht, and M. Krstic, "Towards reconfigurable cnn accelerator for fpga implementation," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 70, no. 3, pp. 1249–1253, 2023.

[5] M. Verucchi, G. Brilli, D. Sapienza, M. Verasani, M. Arena, F. Gatti, A. Capotondi, R. Cavicchioli, M. Bertogna, and M. Solieri, "A systematic assessment of embedded neural networks for object detection," in *Proceedings of IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2020, pp. 937–944.

[6] A. Rodriguez *et al.*, "Lower numerical precision deep learning inference and training," *Intel White Paper*, vol. 3, pp. 1–19, 2018.

[7] Z.-G. Liu, P. N. Whatmough, Y. Zhu, and M. Mattina, "S2ta: Exploiting structured sparsity for energy-efficient mobile cnn acceleration," in *Proceedings of IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2022, pp. 573–586.

[8] T. Abtahi, C. Shea, A. Kulkarni, and T. Mohsenin, "Accelerating convolutional neural network with fft on embedded hardware," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 9, pp. 1737–1749, 2018.

[9] C. Zhang and V. Prasanna, "Frequency domain acceleration of convolutional neural networks on cpu-fpga shared memory system," in *Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2017, pp. 35–44.

[10] Y. Liang, L. Lu, Q. Xiao, and S. Yan, "Evaluating fast algorithms for convolutional neural networks on fpgas," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 4, pp. 857–870, 2019.

[11] W. Sun, H. Zeng, Y.-h. E. Yang, and V. Prasanna, "Throughput-optimized frequency domain cnn with fixed-point quantization on fpga," in *Proceedings of the International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, 2018, pp. 1–8.

[12] L. Xuan, K.-F. Un, C.-S. Lam, and R. P. Martins, "An fpga-based energy-efficient reconfigurable depthwise separable convolution accelerator for image recognition," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 10, pp. 4003–4007, 2022.

[13] A. Boutros, S. Yazdanshenas, and V. Betz, "Embracing diversity: Enhanced dsp blocks for low-precision deep learning on fpgas," in *Proceeding of the International Conference on Field Programmable Logic and Applications (FPL)*, 2018, pp. 35–357.

[14] K. Guo, L. Sui, J. Qiu, J. Yu, J. Wang, S. Yao, S. Han, Y. Wang, and H. Yang, "Angel-eye: A complete design flow for mapping cnn onto embedded fpga," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 37, no. 1, pp. 35–47, 2017.

[15] B. Ronak and S. A. Fahmy, "Multipumping flexible dsp blocks for resource reduction on xilinx fpgas," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 9, pp. 1471–1482, 2016.

[16] S.-E. Chang, Y. Li, M. Sun, R. Shi, H. K.-H. So, X. Qian, Y. Wang, and X. Lin, "Mix and match: A novel fpga-centric deep neural network quantization framework," in *Proceeding of the IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2021, pp. 208–220.

[17] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proceeding of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2009, pp. 248–255.

[18] R. Miyagi, N. Takagi, S. Kinoshista, M. Oda, and H. Takase, "Zytlebot: Fpga integrated ros-based autonomous mobile robot," in *Proceeding of the International Conference on Field-Programmable Technology (ICFPT)*, 2021, pp. 1–4.