Energy-Efficient and Fast Object Detection Using Reconfigurable CNN Accelerators on Mobile FPGA Systems

Dr. Venkata Shesha Giridhar Akula
Principal, Sphoorthy Engineering College Nadergul-Hyderabad

Dr. M.Surya Bhupal Rao
Associate Professor
Department of CSE (AI&ML),CVR College of Engineering
Ibrahimpatnam-Hyderabad

ABSTRACT

In resource-constrained edge computing scenario such as mobile devices, IoT devices, and electric vehicles, energy-efficient convolutional neural network (CNN) accelerators implemented on mobile Field Programmable Gate Arrays (FPGAs) are gaining significant attention due to their high accuracy and scalability. Modern mobile FPGAs, like the Xilinx PYNQ-Z1/Z2 and Ultra96, offer advantages in scalability and flexibility for deploying deep learning-based object detection applications. These FPGAs are particularly suitable for battery-powered systems, including drones and electric vehicles, as they provide energy efficiency in terms of both power consumption and compact size. However, achieving real-time processing remains a challenge due to their limited performance capabilities.

This paper introduces an optimized accelerator design flow at the register-transfer level (RTL) to enhance programming speed and energy efficiency through low-power techniques in FPGA accelerator implementation. While most optimization efforts for FPGA accelerators focus on the system level, we propose a reconfigurable CNN-based object detection accelerator design optimized at the RTL level for mobile FPGAs.

The optimization techniques are presented include various clock gating methods to minimize residual signals and deactivate unnecessary blocks, improving overall efficiency. Based on an analysis of CNN architectures, to identified and classified common computational components, such as multipliers and adders, and integrated them into a universal computing unit. This unit was modularized to suit a hierarchical RTL code structure.

The proposed design was tested using ResNet-20, a CNN with 23 layers, trained on the CIFAR-10 dataset, which consists of a test set of 10,000 images in various formats. Weight

data for the experiment was sourced from Tensil. Experimental results demonstrate that the proposed design improves power efficiency, hardware utilization, and throughput by 16%, 58%, and 15%, respectively.

Keywords:FPGA accelerator, CNN accelerator, RT level design techniques, Reconfigurable accelerator, CNN-based object detection, Mobile FPGA.

I. INTRODUCTION

Convolutional Neural Network (CNN)-based object detection applications are widely utilized across various domains, leveraging Field Programmable Gate Array (FPGA) devices for deployment in personal mobile devices, healthcare systems, smart surveillance, Advanced Driver Assistance Systems (ADAS), drones, and logistics robots [1-6]. CNNs have become indispensable for achieving high recognition accuracy in both cloud-based and edge devices. However, their implementation faces significant challenges due to high computational complexity and substantial power consumption, which are necessary to achieve fast processing speeds and high accuracy simultaneously. This complexity involves extensive computational operations, numerous memory accesses, and dynamic power consumption during data transfers and computation delays.

Real-time inference of CNN-based object detection on mobile FPGA devices is particularly challenging due to limited hardware resources, such as constrained memory size and reduced processor performance. To address these issues, researchers have developed CNN accelerators at various design levels, including system, application, architecture, and transistor levels, to enhance performance and reduce power consumption [7-9]. Recent studies have proposed flexible CNN accelerator designs for FPGA implementations, accommodating a wide range of CNN architectures from lightweight to large-scale models [11-16].

As CNN-based object detection becomes a key technology in unmanned drones, autonomous vehicles, ADAS systems, and industrial automation, research has increasingly focused on enabling real-time processing on mobile FPGA-SoC boards. This includes designing accelerators specifically for mobile FPGA-SoC systems and exploring hardware optimization techniques. Popular devices

such as the Xilinx Ultra96 and Xilinx PYNQ-Z1 are frequently used in drone and IoT applications, with numerous studies addressing their hardware limitations to achieve high performance, low power consumption, and real-time processing speeds [17-24].

The primary implementation techniques discussed in these studies include reducing CNN architecture size, pre-processing input feature maps, optimizing pipeline designs, resizing input and output feature maps, and optimizing code for efficient execution [9, 25-31]. Our previous research demonstrated that register-transfer level (RTL) optimizations effectively reduce processing time and dynamic power consumption [32].

In this work, to apply low-power techniques to the baseline RTL code of a CNN accelerator generated by Tensil and incorporate hardware-optimized methods into a reconfigurable FPGA hardware accelerator design. These optimizations are implemented through an automated RTL code optimization tool. The rest of this paper is organized as follows: Section II introduces low-power techniques at the RTL level for energy-efficient CNN acceleration and provides an overview of the basic hardware design flow using baseline CNN accelerator RTL code generated by Tensil. Section III describes.

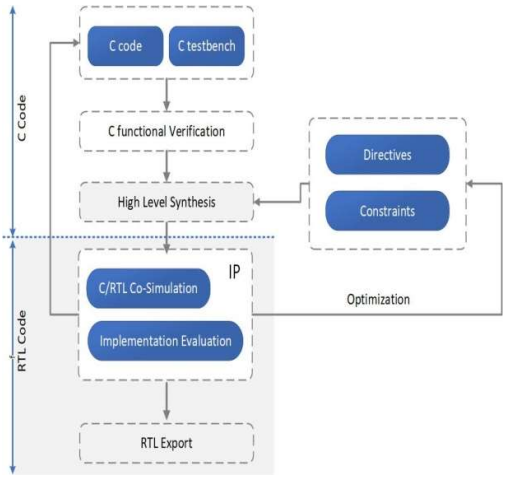

Fig 1.Vivado HLS design flow.

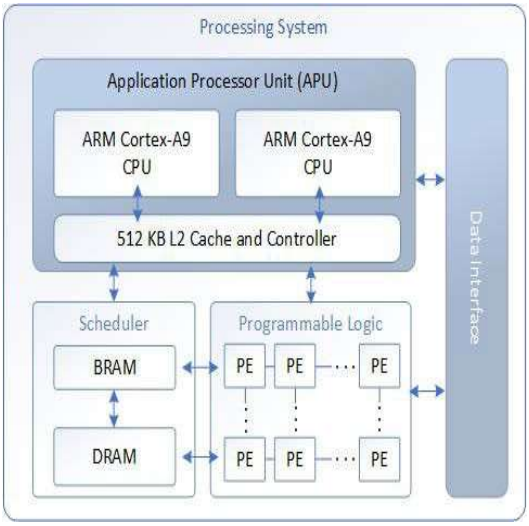

Fig.2. Proposed processing system and Programmable logic unit

II. ACCELERATOR ARCHITECTURE DESIGN

a. Architecture Overview

The block diagram in Figure 2 illustrates the data flow of the proposed processing

unit design for an FPGA-based CNN object detection accelerator. Each block in the programmable logic is specifically defined and modularized to optimize the implementation of various CNN models. The architecture is primarily divided into two main components: the computing processing logic and the memory system, as described below:

In the memory system, three key functional components handle the on-chip and off-chip data transfers required for computation. First, the buffers store data, with weights and intermediate feature maps organized in a layer-by-layer format stored in external DRAM. When loading a tile of data into the on-chip input/weight/output ping-pong buffers, the data is arranged in a specific format according to the computation mode's requirements.

Second, a dispatching module utilizes a Direct Memory Access (DMA) engine, controlled by the DMA control module, to fetch the necessary data from DRAM or save results back to DRAM. Third, the on-chip data scheduling modules, including scatter and gather units, perform serial-to-parallel or parallel-to-serial conversions, managing the data flow for subsequent computations or data transmissions.

b. Proposed Reconfigurable Accelerator Hardware Architecture

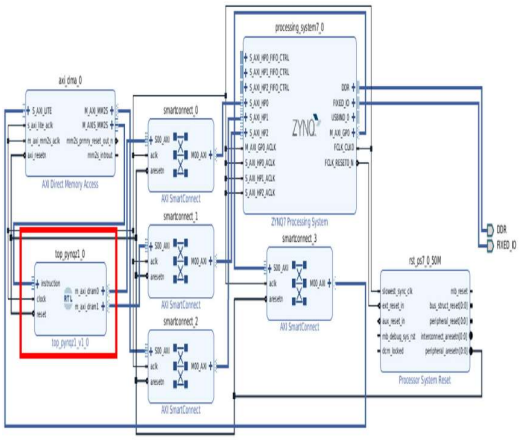

Fig. 3. IP Block design for CNN object detection.

As depicted in Figure 3, the IP block design for the CNN object detection accelerator comprises both referencing and customized IP blocks (top_pynqz1_0). Within the top_pynqz1_0 block, multiple modules are hierarchically organized, including multiply-accumulate units (MACs), pooling units (POOLs), memory bandwidth management, memory access scheduler, and convolution (CONV) computing modules. The original RTL code from Tensil lacks a hierarchical architecture; however, introducing this structure enhances resource management efficiency, although analyzing the RTL code becomes more time-consuming due to the added complexity.

The primary feature of FPGA devices is their reconfigurability. To fully leverage the flexibility of the FPGA-SoC design, the proposed RTL code for the CNN accelerator was designed with hierarchical and modular components, such as MACs, convolution (Conv), multipliers, adders, multiplexers (MUX), and arithmetic logic units (ALUs), as shown in Figure 4. This design provides scalability to support various CNN architectures, including YOLO series and ResNet20. After modularizing the MAC unit, we incorporated low-power techniques, such as clock gating and XOR/OR gates for MUXs.

This design also accommodates additional detectors, such as Single-Shot Detectors (SSD) and Multibox detectors. For memory access modules—like InnerDualPortMem1, Dual-PortMem1, MemSplitter, and MemBoundarySplitter—memory partitioning techniques are applied. To accelerate CPU computing, a memory reassignment technique is used, enabling dynamic changes to memory size and flow based on pre-assigned computations. For instance, when targeting a CNN accelerator with fixed 16-bit precision, the memory size can be pre-allocated for the input data or weights, facilitating efficient sequential operations such as convolution.

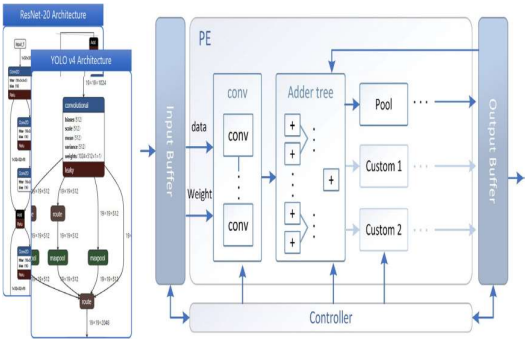

Fig. 4. Flexible accelerator design overview

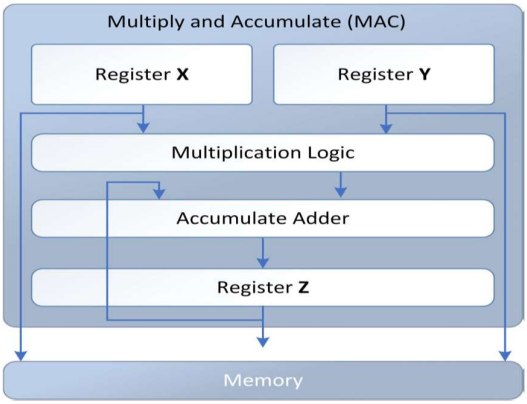

Fig. 5.Conventional MAC unit design

In contrast to the conventional design shown in Figure 5, an AND gate and a latch were added to ensure the clock is safely disabled, preventing any glitches from reaching the register clock.

V. EXPERIMENT AND RESULTS WITH DISCUSSION

a. Experiment Environment

For the basic hardware platform, we selected the PYNQ-Z1 board instead of the standard ZYNQ-7020 board. The PYNQ-Z1 is an open-source project from AMD that integrates the Xilinx ZYNQ-7020 and provides a Jupyter-based framework with Python APIs. This FPGA-SoC platform combines Programmable Logic (PL) and

Processing System (PS). The primary software development tool is Jupyter Notebook, a web-based programming platform that supports Python, C/C++, and other open-source libraries such as Open CV.

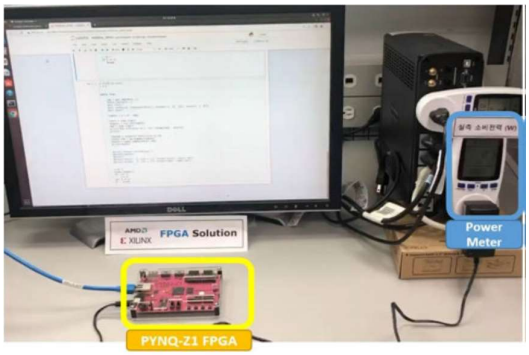

Fig. 6. FPGA testbed (Xilinx PYNQ-Z1 FPGA).

Fig.7. HW resource report comparison of Tensil sample simulation and our work tested on PYNQ-Z1.

Our experimental setup is illustrated in Figure 6. The CNN architecture used in the experiment is ResNet-20, which consists of 23 layers and was trained on the CIFAR-10 dataset, containing 10,000 images in various formats. We utilized the provided weight file and converted the ResNet model to ONNX format. ONNX, a machine learning model converter, produces the converted model code in ONNX format. The Tensil compiler generates three import artifacts: .tmodel, .tdata, and .tprog files. After the .tmodel manifest is loaded into the driver, it directs the driver to locate the binary files, program data, and weight data. These files were used without any modifications, ensuring the accuracy remained unchanged.

b. FPGA Implementation Results

Compared to Tensil's optimization results, we observed that more register buffers are activated in our proposed structure. After verifying the functionality and performance, the structure can be further refined through RTL code modifications. This enables improvements in specific hardware resources and power consumption. Analyzing the results demonstrates enhanced performance in CNN processing. Figure 16 illustrates the reduction in power consumption of the processing system unit. We achieved a power efficiency of 43.9 GOPs/W, which is 1.37 times higher than other FPGA board implementations. Additionally, hardware resource utilization in DSPs was increased by 2.2 times compared to the results from [24].

c. Power Consumption Results

Our optimization reduces dynamic power consumption by 16%. Additionally, the total on-chip power consumption is decreased by 20%. Activating the global buffer, along with the unused global clock buffer and the second global clock resource, helps improve the design's performance. This approach also addresses high fan-out signals, ensuring the device functions optimally. In the pipeline logic, inserting an intermediate flip-flop (FF) can enhance the device's speed.

Table1.Comparison result of FPGA implementation

	[17]	[18]	[19]	[20]	[21]	[22]	[23]	[24]	Ours
Year	2018	2018	2018	2018	2017	2018	2019	2022	2022
CNN Model	AlexNet	MobileNet	VGG	VGG	VGG	VGG16	AP2D-Net	ResNet-20	ResNet20
FPGA	ZYNQ-XCZ7020		ZYNQ-XCZ7020			Intel Arria10	Ultra96	PYNQ-Z1	PYNQ-Z1
Clock	200	133	214	150	150	200	300	50	50
	268	1844**	85.5	122(	919	2232**	162	198	523
DSPs	218	1278	190	216(	103(	1518	287	73	167
LUTs	49.8	-	29.9	-	-	-	54.3K	14.6K	15.2K
FFs	61.2	-	35.5	-	-	-	94.3K	9.1K	41.2K
Precision (W,A)*	(16, 16)	(16, 16)	(8, 8)	(16, 16)	(16, 16)	(16, 16)	(8-16, 16)	(16, 16)	(16, 16)
Latency(s)	0.01	0.004	0.36	0.10	0.10	0.043	0.032	0.178	0.109
Throughput (GOPs)	80.3	170.6	84.3	290	364.	715.9	130.2	55.0	63.3
Power(W)	2.21	-	-	35	25	-	5.59	1.714	1.440
Power Efficiency (GOPs/W)	36.3	-	-	8.28	14.5	-	23.3	28.2	43.9

However, excessive use of flip-flops increases computational complexity. Our low-power techniques outperform FFs in terms of performance.

VII. CONCLUSION

In this paper, the proposed highly reconfigurable FPGA hardware accelerator demonstrates improved performance in terms of processing speed and power consumption during the inference of various CNN models. The hardware optimization focuses on two main goals: enhancing throughput and reducing power consumption. To improve performance, a minimized data transfer strategy was implemented by assigning the maximum possible buffer sizes during computations and applying a controlled pipeline design to reduce data access.

To achieve energy-efficient results for CNN-based object detection, we not only controlled data access to minimize memory usage, but also introduced low-power techniques at the register-transfer level (RTL). These techniques include reconfigured Multiply-Accumulate (MAC) units, advanced clock gating applied to adders, register Z with bus-specific clocks, and OR-based MAC architecture.

The proposed hardware accelerator for ResNet-20 was implemented on the PYNQ-Z1 mobile FPGA-SoC, and power consumption was measured during

inference operations. The results showed a 15% improvement in throughput compared to the baseline RTL code, a 16% reduction in power consumption, and a 58% increase in hardware utilization. The object detection processing speed reached 9.17 FPS, demonstrating the feasibility of real-time processing on mobile FPGA devices.

REFERENCES

[1] Mei-Ling. L. Liu, Distributed [1] A. K. Jameil and H. Al-Raweshidy, ‘‘Efficient CNN architecture on FPGAusing high level module for healthcare devices,’’ IEEE Access, vol. 10, pp. 60486–60495, 2022.

[2] S. Y. Nikouei, Y. Chen, S. Song, R. Xu, B. Choi, and T. Faughnan, ‘‘Smart surveillance as an edge network service: From Harr-Cascade, SVM to a lightweight CNN,’’ in Proc. IEEE 4th Int. Conf. Collaboration Internet Comput. (CIC), Oct. 2018, pp. 256–265.

[3] K. Haeublein, W. Brueckner, S. Vaas, S. Rachuj, M. Reichenbach, and D. Fey, ‘‘Utilizing PYNQ for accelerating image processing functions in ADAS applications,’’ in Proc. 32nd Int. Conf. Archit. Comput. Syst., May 2019, pp. 1–8.

[4] Z. Zhang, M. A. P. Mahmud, and A. Z. Kouzani, ‘‘FitNN: A low-resource FPGA-based CNN accelerator for drones,’’ IEEE Internet Things J., vol. 9, no. 21, pp. 21357–21369, Nov. 2022.

[5] C. Fu and Y. Yu, ‘‘FPGA-based power efficient face detection for mobile robots,’’ in Proc. IEEE Int. Conf. Robot.Biomimetics (ROBIO), Dec. 2019, pp. 467–473.

[6] X. Li, X. Gong, D. Wang, J. Zhang, T. Baker, J. Zhou, and T. Lu, ‘‘ABMSpConv-SIMD: Accelerating convolutional neural network inference for industrial IoT applications on edge devices,’’ IEEE Trans. Netw. Sci. Eng., early access, Feb. 25, 2022, doi: 10.1109/TNSE.2022.3154412.

[7] S. Tamimi, Z. Ebrahimi, B. Khaleghi, and H. Asadi, ‘‘An efficient SRAMbased reconfigurable architecture for embedded processors,’’ IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 38, no. 3, pp. 466–479, Mar. 2019.

[8] A. J. A. El-Maksoud, M. Ebbed, A. H. Khalil, and H. Mostafa, ‘‘Power efficient design of high-performance convolutional neural networks hardware accelerator on FPGA: A case study with GoogLeNet,’’ IEEE Access, vol. 9, pp. 151897–151911, 2021.

[9] S. Lee, D. Kim, D. Nguyen, and J. Lee, ‘‘Double MAC on a DSP: Boosting the performance of convolutional neural networks on FPGAs,’’IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 38, no. 5, pp. 888–897, May 2019.

[10] S. Ullah, S. Rehman, M. Shafique, and A. Kumar, ‘‘High-performance accurate and approximate multipliers for FPGA-

based hardware accelerators,''IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 41, no. 2, pp. 211–224, Feb. 2022.

[11] 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.

[12] W. Liu, J. Lin, and Z. Wang, ''A precision-scalable energy-efficient convolutional neural network accelerator,'' IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 67, no. 10, pp. 3484–3497, Oct. 2020.

[13] H. Irmak, D. Ziener, and N. Alachiotis, ''Increasing flexibility of FPGAbasedCNN accelerators with dynamic partial reconfiguration,'' in Proc.31st Int. Conf. Field-Programmable Log. Appl. (FPL), Aug. 2021,pp. 306–311.

[14] W. Chen, D. Wang, H. Chen, S. Wei, A. He, and Z. Wang, ''An asynchronous and reconfigurable CNN accelerator,'' in Proc. IEEE Int. Conf. Electron Devices Solid State Circuits (EDSSC), Jun. 2018, pp. 1–2.

[15] C. Yang, Y. Wang, H. Zhang, X. Wang, and L. Geng, ''A reconfigurable CNN accelerator using tile-by-tile computing and dynamic adaptive data truncation,'' in Proc. IEEE Int. Conf. Integr. Circuits, Technol. Appl. (ICTA), Nov. 2019, pp. 73–74.

[16] S. Zeng, K. Guo, S. Fang, J. Kang, D. Xie, Y. Shan, Y.Wang, and H. Yang, ''An efficient reconfigurable framework for general purpose CNN-RNN models on FPGAs,'' in Proc. IEEE 23rd Int. Conf. Digit. Signal Process.(DSP), Nov. 2018, pp. 1–5.

[17] L. Gong, C. Wang, X. Li, H. Chen, and X. Zhou, ''MALOC: A fully pipelined FPGA accelerator for convolutional neural networks with all layers mapped on chip,'' IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 37, no. 11, pp. 2601–2612, Nov. 2018.

[18] L. Bai, Y. Zhao, and X. Huang, ''A CNN accelerator on FPGA using depthwise separable convolution,'' IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 65, no. 10, pp. 1415–1419, Oct. 2018.

[19] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song, Y. Wang, and H. Yang, ''Going deeper with embedded FPGA platform for convolutional neural network,'' in Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays. New York, NY, USA: Association for Computing Machinery, Feb. 2016, pp. 26–35, doi: 10.1145/2847263.2847265.

[20] T. Geng, T. Wang, A. Sanaullah, C. Yang, R. Patel, and M. Herbordt, ''Aframework for acceleration ofCNNtraining on deeply-pipelined FPGA clusters with work and weight load balancing,'' in Proc. 28th Int. Conf.

Field Program. Log. Appl. (FPL), Aug. 2018, p. 394.

[21] Y. Guan, H. Liang, N. Xu, W. Wang, S. Shi, X. Chen, G. Sun, W. Zhang, and J. Cong, ‘‘FP-DNN: An automated framework for mapping deep neural networks onto FPGAs with RTL-HLS hybrid templates,’’ in Proc. IEEE 25th Annu. Int. Symp. Field-Programmable Custom Comput. Mach. (FCCM), Apr. 2017, pp. 152–159.

[22] Y. Ma, Y. Cao, S. Vrudhula, and J. Seo, ‘‘Optimizing the convolution operation to accelerate deep neural networks on FPGA,’’ IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 26, no. 7, pp. 1354–1367, Jul. 2018.

[23] S. Li, Y. Luo, K. Sun, N. Yadav, and K. K. Choi, ‘‘A novel FPGA accelerator design for real-time and ultra-low power deep convolutional neural networks compared with Titan X GPU,’’ IEEE Access, vol. 8, pp. 105455–105471, 2020.

[24] Learn Tensil With ResNet and PYNQ Z1. Accessed: Dec. 15, 2022.[Online].Available:https://www.tensil.ai/docs/tutorials/resnet20-pynqz1/

[25] X. Zhang, Y. Ma, J. Xiong, W. W. Hwu, V. Kindratenko, and D. Chen, ‘‘ExploringHW/SWco-design for video analysis on CPU-FPGA heterogeneous systems,’’ IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 41, no. 6, pp. 1606–1619, Jun. 2022.

[26] E. Antolak and A. Pulka, ‘‘Energy-efficient task scheduling in designof multithread time predictable real-time systems,’’ IEEE Access, vol. 9, pp. 121111–121127, 2021.

[27] W. Huang, H.Wu, Q. Chen, C. Luo, S. Zeng, T. Li, and Y. Huang, ‘‘FPGAbasedhigh-throughput CNN hardware accelerator with high computing resource utilization ratio,’’ IEEE Trans. Neural Netw. Learn. Syst., vol. 33, no. 8, pp. 4069–4083, Aug. 2022.

[28] G. Lakshminarayanan and B. Venkataramani, ‘‘Optimization techniques for FPGA-based wave-pipelined DSP blocks,’’ IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 13, no. 7, pp. 783–793, Jul. 2005.

[29] D. Wang, K. Xu, J. Guo, and S. Ghiasi, ‘‘DSP-efficient hardware acceleration of convolutional neural network inference on FPGAs,’’ IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 39, no. 12, pp. 4867–4880, Dec. 2020.

[30] A. Prihozhy, E. Bezati, A. A. A. Rahman, and M. Mattavelli, ‘‘Synthesis and optimization of pipelines for HW implementations of dataflow programs,’’IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 34, no. 10, pp. 1613–1626, Oct. 2015.

[31] W. Lou, L. Gong, C. Wang, Z. Du, and X. Zhou, ‘‘OctCNN: A high throughput FPGA accelerator for CNNs using octave convolution algorithm,’’ IEEE Trans. Comput., vol. 71, no. 8, pp. 1847–1859, Aug. 2022.

[32] H. Kim and K. Choi, ‘‘Low power FPGA-SoC design techniques for CNNbasedobject detection accelerator,’’ in Proc. IEEE 10th Annu. Ubiquitous Comput., Electron. Mobile Commun. Conf. (UEMCON), Oct. 2019, pp. 1130–1134.