IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2023.0322000

A Reconfigurable CNN-based Accelerator Design for Fast and Energy-Efficient Object Detection System on Mobile FPGA

HEEKYUNG KIM[1], (Student, IEEE), and KYUWON KEN CHOI[1], (Senior Member, IEEE)
[1]Department of Electrical and Computer Engineering, Illinois Institute of Technology, IL 60616 USA (hkim104@hawk.iit.edu)

Corresponding author: Heekyung Kim (e-mail: hkim104@hawk.iit.edu).

This work was supported by the Technology Innovation Program of the Ministry of Trade, Industry & Energy (MOTIE, Korea) Korea Electronics Technology Institute (KETI), South Korea (Software and Hardware Development of cooperative autonomous driving control platform for commercial special and work-assist vehicles), under Grant 1415181272.

ABSTRACT In limited-resource edge computing circumstances such as on mobile devices, IoT devices, and electric vehicles, the energy-efficient optimized convolutional neural network (CNN) accelerator implemented on mobile Field Programmable Gate Array (FPGA) is becoming more attractive due to its high accuracy and scalability. In recent days, mobile FPGAs such as the Xilinx PYNQ-Z1/Z2 and Ultra96, definitely have the advantage of scalability and flexibility for the implementation of deep learning algorithm-based object detection applications. It is also suitable for battery-powered systems, especially for drones and electric vehicles, to achieve energy efficiency in terms of power consumption and size aspect. However, it has a low and limited performance to achieve real-time processing. In this article, optimizing the accelerator design flow in the register-transfer level (RTL) will be introduced to achieve fast programming speed by applying the low-power techniques on FPGA accelerator implementation. In general, most accelerator optimization techniques are conducted on the system level on the FPGA. In this article, we propose the reconfigurable accelerator design for a CNN-based object detection system on mobile FPGA. Furthermore, we present RTL optimization design techniques that will be applied such as various types of clock gating techniques to eliminate residual signals and to deactivate the unnecessarily active block. Based on the analysis of the CNN-based object detection architecture, we analyze and classify the common computing operation components from the Convolutional Neuron Network, such as multipliers and adders. We implement a multiplier/adder unit to a universal computing unit and modularize it to be suitable for a hierarchical structure of RTL code. Experimental results show that the proposed design process improves the power efficient consumption, hardware utilization, and throughput by 16%, up to 58%, and 15%, respectively.

INDEX TERMS FPGA Accelerator; CNN Accelerator; RT Level Design Techniques; Low Power Techniques; Reconfigurable Accelerator; CNN-based Object Detection; Low Power Consumption; High Performance; Mobile FPGA

I. INTRODUCTION

CONVOLUTIONAL Neural Network(CNN)-based object detection application has been applied in various systems including Field Programmable Gate Array (FPGA) devices from personal mobile devices to industrial machines such as healthcare devices, smart surveillance systems, Advanced Driver Assistance Systems (ADAS), drones, and logistics robots [1]–[6]. To achieve high accuracy of recognition, CNNs have become an essential feature of many diverse object detection-adopted devices, whether it is cloud-based or edge devices. The primary implementation issue of the CNN application is that computing complexity is above average and the power consumption amount is huge to achieve fast processing speed and high accuracy at the same time. High computing complexity also involves a large number of operation units, and massive memory accesses as well. The dynamic power consumption occurs over the data transfer process and in the time-delay process of computing operation. It seems impossible to make the real-time inference of CNN-based object detection on mobile FPGA devices which

have limited hardware resources such as memory size and lower processor performance. In these power and hardware resource-limited circumstances, to improve performance and reduce power consumption, many researchers have proposed CNN accelerators at various design levels including system level, application level, architecture level, and transistor level [7]–[10]. Recent studies have proposed a flexible CNN accelerator design for FPGA implementation at the transistor level and a flexible FPGA accelerator for various CNN architectures from lightweight CNN to large-scale CNN [11]–[16].

Since CNN-based object detection applications become more common technology for unmanned drones, autonomous vehicles, ADAS systems on the vehicle, and industrial automation systems, researchers have been conducting CNN object detection-related research in terms of the following topics; implementation on the mobile FPGA-SoC board for real-time processing, accelerator design for mobile FPGA-System-On-Chip (SoC), and hardware optimization techniques, are becoming popular. To overcome the lack of hardware resources on the mobile FPGAs such as Xilinx Ultra 96 and Xilinx PYNQ-Z1 which are popular FPGA-SoC devices implemented on drone and IoT devices, many papers have been published to achieve high performance, low power consumption, and real-time processing speed [17]–[24]. The main focus of their proposed implementation techniques in those papers is reducing the size of the CNN architecture, pre-processing the input feature map, tightening pipe-lining design, size adjustment of the input and output feature maps, and code optimization [9], [25]–[32]. Moreover, in previous our research, we verified that RT-level optimization is able to not only reduce the processing time but also, save dynamic power [33]–[35].

Therefore, in this work, we applied the low-power techniques to the baseline RTL code of the CNN accelerator generated from the Tensil and applied the hardware-optimized techniques to the proposed reconfigurable FPGA hardware accelerator design through the proposed automated optimization tool for RTL code. The rest of this paper is organized as follows: Section II introduces the low-power techniques in RT-level for energy-efficiently accelerating the CNN computing operation and overviews the basic RT-level-based optimization hardware design flow based on generated a baseline CNN accelerator RTL code by Tensil. Section III describes the architecture of the proposed accelerator and the design details of data flow and processing modules in two parts, optimization & modularization and low power techniques. Section V discusses the implementation and simulation results with previous works. Finally, the conclusions are given in Section VI.

II. BACKGROUND

To design the CNN accelerator on an FPGA-SoC board, the use of CAD tools and platforms is required. Each manufacturer provides the CAD tools and development platforms for the implementation process and reconfigurable components and parts on the FPGA (e.g., Vivado from Xilinx, Quartus Prime from Intel, and PYNQ). However, due to the closed platform feature of the Xilinx FPGA products, in the High Level of Synthesis (HLS) design flow as shown in Figure 1, the Vivado HLS system can verify the functionality of the C/C++/System C code and convert the code to the register-transfer level (RTL) code for the FPGA hardware operation and optimization [7], [23], [36], however, once the RTL code is generated by Vivado HLS Tool, the code is no longer readable or modifiable. On the other hand, the platform-based design flow can import the VHDL/Verilog code to set as customized IP blocks so that we can easily modify the hardware design at the RT level or gate level and intuitively configure the data flow for Processing System (PS) and Programmable Logic (PL) through the Vivado IP Integrator.

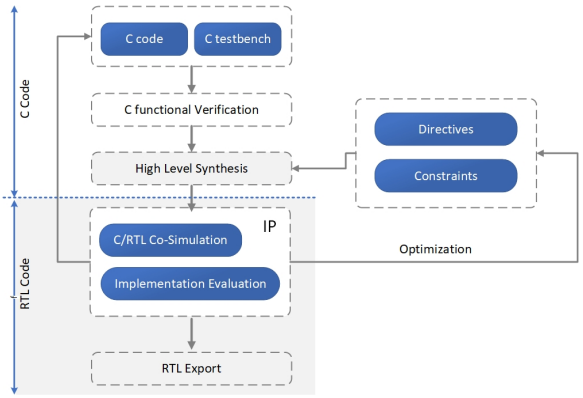


FIGURE 1. Vivado HLS Design Flow

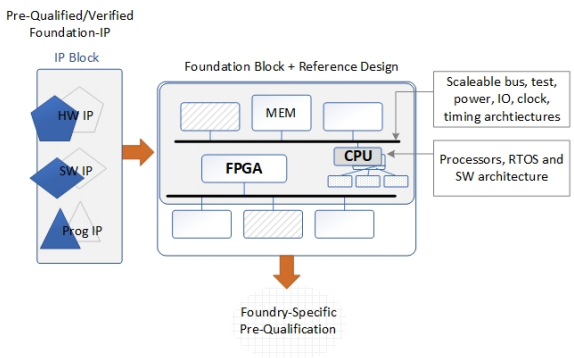


FIGURE 2. FPGA SoC Platform Design Architecture

A. PLATFORM-BASED DESIGN FLOW WITH RTL CODE

The platform-based design flow was introduced by Xilinx Vivado which is an integrated design environment program tool as shown in Figure 1. The RTL code can be imported into the IP block, and it can be assembled with other peripheral IP blocks and PS IP blocks to generate the hardware design for the bitstream. Jupyter Notebook is a web-based primary computing environment of the PYNQ which is linked to Xilinx platforms [37]. PYNQ is running based on Python on the Jupyter Notebook with Linux kernel on the FPGA. However,

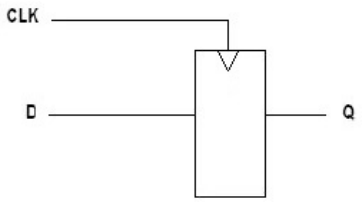


FIGURE 3. Conventional Register

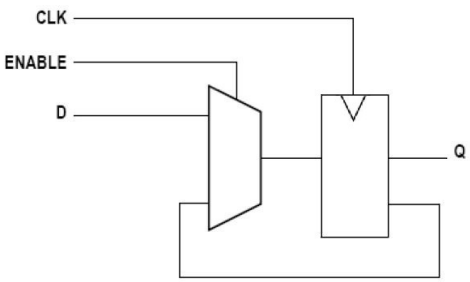


FIGURE 4. Local Explicit Clock Enable(LECE)

the Python library is not fully supported in the PYNQ.

B. BASELINE RTL CODE GENERATION FOR CNN ACCELERATOR: TENSIL

Tensil is a set of tools for designing the accelerator including an RTL generator, a model compiler, and a set of drivers [24]. The basic processing flow is that using the selected machine learning accelerator architectures on the limited FPGA-SoC devices, it generates the RTL code by using a model compiler. The primary advantage of Tensil is that it is able to create an accelerator without quantization or other degradation. Tensil applies a few of the optimization techniques for the selected FPGAs, thus the optimization performance is not effective enough. Previously we applied our low-power techniques to CNN accelerator RTL code and verified the performance [33]. In section III, we apply the techniques to the Tensil RTL code and evaluate the effectiveness of the techniques on the FPGA-SoC boards, PYNQ-Z1.

C. LOW POWER TECHNIQUES AT RT-LEVEL

1) Low Power Clock Techniques

Clock Gating(CG) is a basic low-power technique to enhance performance and efficiency by disabling unnecessary clock cycles as shown in Figure 3. Standby states are included in many parts of the CNN computing process. This leads to a significant amount of power consumption. CG eliminates unnecessary clock cycle occurrences. Local Explicit Clock Enable (LECE) [38]–[40] is a method using *ENABLE* signal for 2:1 multiplexer or multiplexed D flip-flop to update the output on the rising edge of the clock only when the *ENABLE* signal is high as shown in Figure 4. The more bits are used as an input, the more *ENABLE* signals occur. The Local Explicit Clock Gating (LECG) [38]–[40] has the equivalent fundamental of the LECE as shown in Figure 5, however,

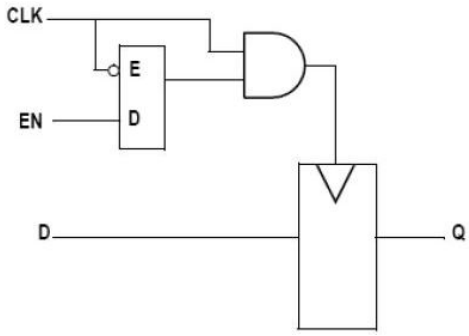


FIGURE 5. Local Explicit Clock Gating (LECG)

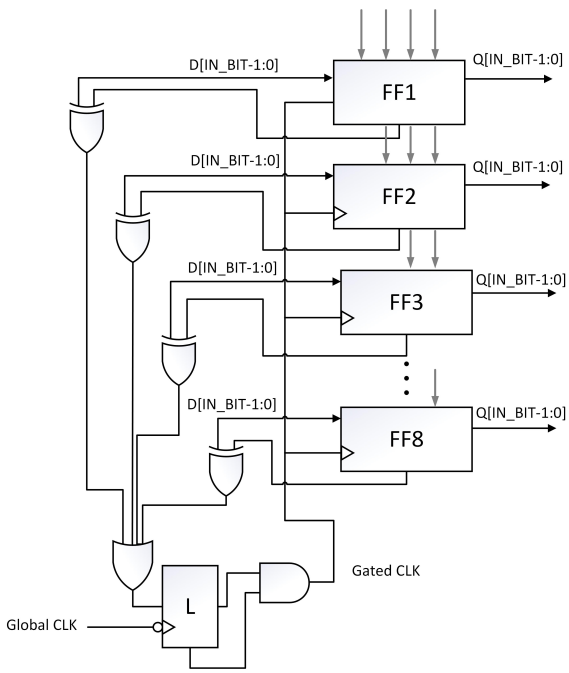


FIGURE 6. Bus-Specific Clock Gating (BSCG)

LECG has the advantage of reducing power consumption in the case of the multi-bit of output, by updating output at once when completing the output update.

Bus-Specific Clock Gating (BSCG) [38]–[40] utilizes the clock gating technique and adjusts the *EN* signal based on the comparison of I/O signals as shown in Figure 6. In terms of power consumption, XOR gates are significantly lower power-consuming logic gates for the gate-level power analysis compared to AND/OR gates. Enhanced Clock Gating (ECG) [38], [40] consists of XOR gates to control the input clock signals and enable signals when considering multi-bit I/O data as shown in Figure 7. The efficiency of the power reduction would be maximized when there are larger pipelines and IO bit sizes.

III. ARCHITECTURE DESIGN OF ACCELERATOR

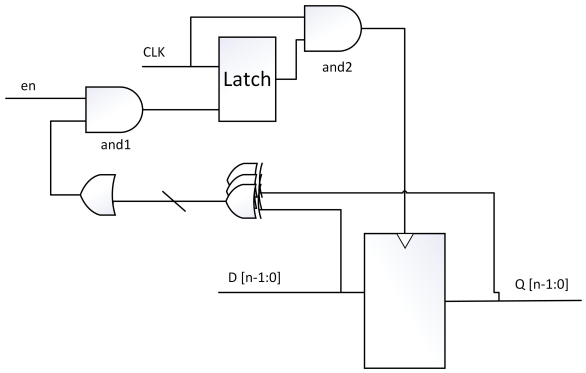


FIGURE 7. Enhanced Clock Gating (ECG)

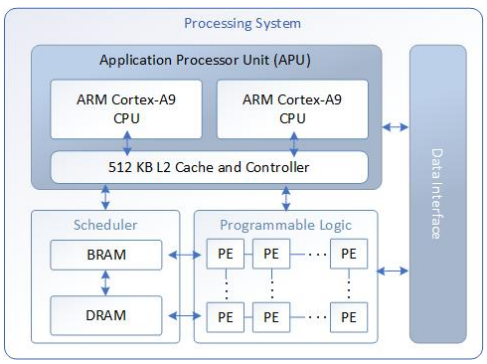


FIGURE 8. Proposed Processing System and Programmable Logic Unit

A. ARCHITECTURE OVERVIEW

The block diagram in Figure 8 shows a data flow of the proposed processing unit design for an FPGA-based CNN object detection accelerator. For the programmable logic, each type of block is defined specifically and is modularized to enhance the implementation efficiency of various CNN models. The proposed architecture can be mainly divided into computing processing logic and memory system as detailed as follows: In the memory system, there are three main functional components for the on-chip and off-chip data transfer to prepare data for computation. First, the buffers are responsible for storing data. All the weights and intermediate feature maps are arranged in a layer-by-layer format which is stored in external DRAM. When loading a tile of data to the on-chip input/weight/output ping-pong buffers, they are arranged in a unique format according to the requirement of computation mode. Second, a dispatching module employs Direct Memory Access (DMA) engine through DMA descriptors generated by the DMA control module to fetch required data from DRAM or save the results back to DRAM. Third, the on-chip data scheduling modules, consisting of scatter and gather modules, realize the serial-to-parallel or parallel-to-serial conversions, which manipulate the data flow for the following computation or transmission.

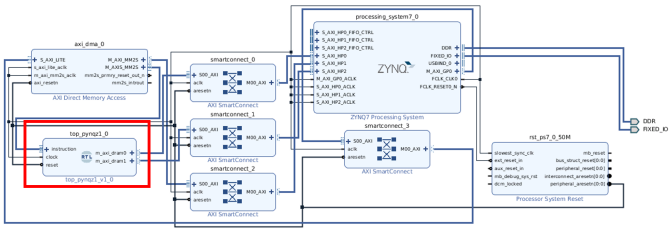


FIGURE 9. IP Block Design for CNN Object Detection

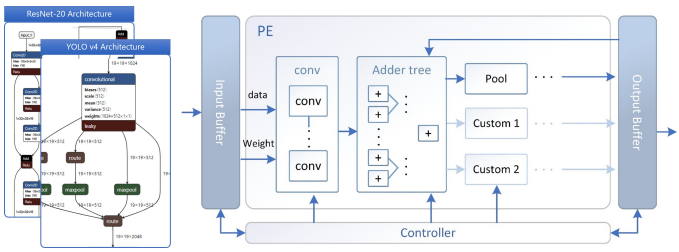


FIGURE 10. Flexible Accelerator Design Overview

B. THE PROPOSED RECONFIGURABLE ACCELERATOR HARDWARE ARCHITECTURE

As shown in Figure 9, the IP block design for the CNN object detection accelerator consists of referencing IP blocks and customized IP blocks (top_pynqz1_0). In the top_pynqz1_0 block, there are hierarchically defined multiply-accumulate units (MACs), POOLs, memory bandwidth, memory access scheduler, and CONV computing modules. The original Tensil's RTL codes do not have hierarchical architecture, However, in this case, analysis of the RTL code would take a long time.

The primary feature of FPGA devices is in reconfigurability. Therefore, to maximize the flexibility of the FPGA-SoC design, the proposed RTL code of the CNN accelerator was designed with hierarchical and modularized main modules including MACs, Conv, Multiplier, Adder, MUX, and ALU as shown in Figure 10. This figure shows that the proposed flexible accelerator design has the scalability to support the different CNN architectures such as the YOLO series and ResNet20. After the modularization of the MAC unit, we applied our low-power techniques such as clock gating, XOR gate, and OR gate for MUXs. This design is able to accommodate add-on detectors, such as Single-Shot Detectors (SSD) and Multibox detectors. For the memory access modules such as *InnerDualPortMem1*, *DualPortMem1*, *MemSplitter*, and *MemBoundarySplitter*, memory partitioning techniques are applied. To accelerate the CPU computing operation, the memory reassignment technique has been applied so that the memory size and flow would be changed once it detected the pre-assigned computation. For example, our target CNN accelerator architectures should be using fixed 16-bit, then we can pre-assign the memory size prior to the input data or the weight. This would be helpful to compute the sequential computation operation such as convolution operation.

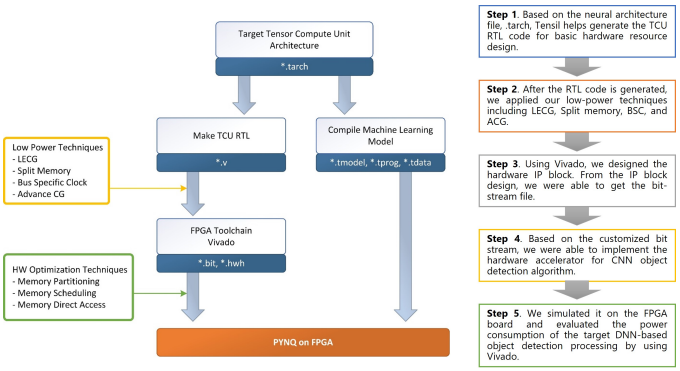


FIGURE 11. Low Power Design Flow

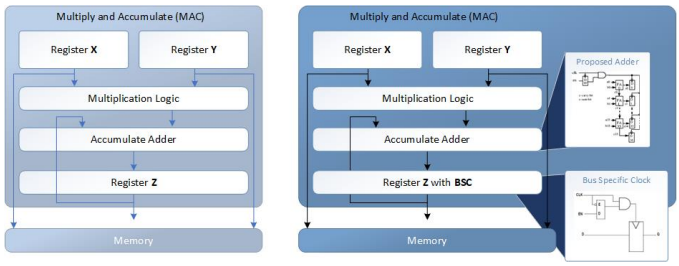


FIGURE 12. Conventional MAC (Left) Proposed MAC (Right)

```
always@ clock, mul_input, passthrough
      if(!clock)
          latch_q <= |(passthrough ^ mul_input);

          assign gclk = latch_q && clock;

always @(posedge gclk) begin
      if (reset) begin
           weight <= 0;
      end else if (load) begin
           weight <= add_input;
      end
      if (reset) begin
           passthrough <= 0 for fixed-data input size (16'b);
      end else begin
           passthrough <=mul_input;
      end
      if (reset) begin
          ⋮
      end else if () begin
          ⋮
      end
    end
```

FIGURE 13. Psuedocode for Proposed MAC Operation

IV. PROPOSED HARDWARE IMPLEMENTATION

A. LOW POWER HW DESIGN TECHNIQUES AT RTL

As shown in Figure 9, in this experiment, our optimization for low power is targeted to the Tensil's processing flow. Step 1. Based on the neural architecture file, .tarch, Tensil helps generate the TCU RTL code for basic hardware resource design as shown in Figure 11. Step 2. After the RTL code is generated, we applied our low-power techniques including LECG, Split memory, BSC, and ECG. Step 3. Using Vivado, we designed the hardware IP block. From the IP block design, we were able to get the bit-stream file. Step 4. Based on the customized bit stream, we were able to implement the hardware accelerator for the CNN object detection algorithm. Step 5. We simulated it on the FPGA board and evaluated the power consumption of the target DNN-based object detection processing by using Vivado.

The optimized multiplier design using a power-efficient adder block based on power analysis was implemented at the transistor level as shown in Figure 12 (right). In convolution computation, the computation complexity of the multipliers can cause dynamic power consumption and delays. To reduce the complexity of the adder and multiplier, first, we tested the full adder designs through the transistor-level design process. For the MAC module, Bus Specific Clock (BSC) is applied. In a conventional register, the data input is active and lasts until the end of the period. In this case, the power could be wasted. When BSC is applied to register Z, the XOR can control enabling the clock so that the clock toggles are not wasted. The AND gate and Latch were added to safely disable the clock without allowing any glitches to reach the register clock.

B. PROPOSED MAC HARDWARE DESIGN

The detailed technique approaches are as follows: 1. MAC unit is the major power consumption unit of the convolution operation in which the data transmissions occur frequently. This technique is applied to remove the wasted clock toggles during the data input is deactivated. 2. The proposed adder group with BSC can reduce the wasted clock toggles so that it can reduce the power consumption of the adder unit. 3. In stochastic multiplication, two unary bit-streams can be operated using AND gate and the OR gate can be applied instead of the MUX operation. Not only as the same as MUX, OR gate can support parallel MAC operation, but also, it consequences a reduced dynamic power consumption result. Eventually, we utilized this parallel MAC structure using OR gate as shown in Figure 12 (b). Figure 13 shows the proposed MAC pseudocode which has been applied to BSC and OR-based MAC computing operations.

C. FLEXIBLE ACCELERATOR DESIGN FOR MULTI-ARCHITECTURE AND OPTIMIZATION TECHNIQUES

Based on the analysis result of the target CNN architecture, we customize the pipe-lining of the data flow and assign maximized buffer capacity in the BRAM and external memory. The fixed-point numbers are able to reduce the computation resource consumption and it also is able to reduce the bandwidth requirements, however, for getting high performance, the optimized size of bandwidth should be defined by the

FIGURE 14. GPU Testbed (GPU TITAN X)

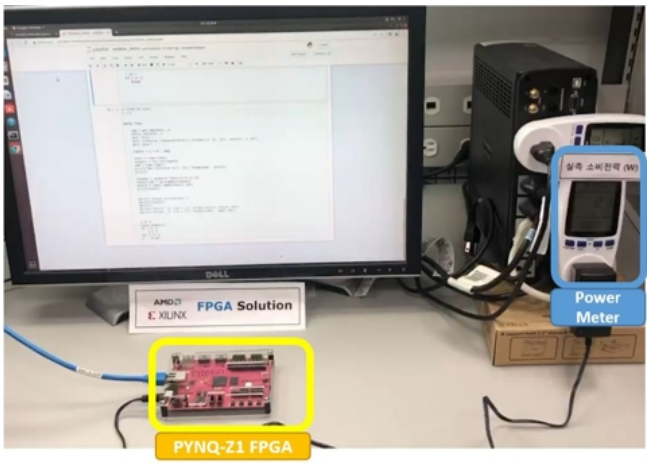


FIGURE 15. FPGA Testbed (Xilinx PYNQ-Z1 FPGA)

analysis of the network architecture. Once the data transmission size is fixed, memory splitting and merging should be applied. Our CNN accelerator is based on the 16-bit fixed point bandwidth which is given by the reference [24]. We modularize the RTL code based on thorough analysis, which helps easy modification for implementing the accelerator design.

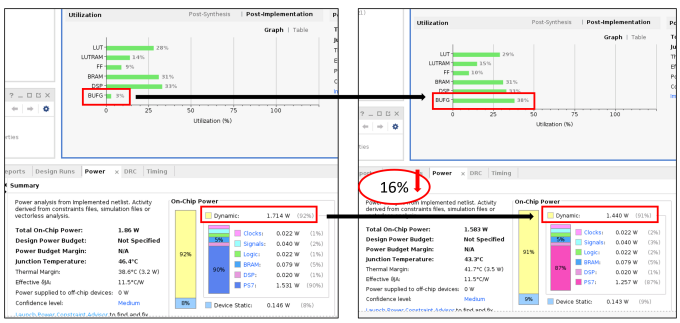
V. EXPERIMENT AND RESULTS WITH DISCUSSION

A. EXPERIMENT ENVIRONMENT

For the basic hardware platform, we chose the PYNQ-Z1 board instead of the regular ZYNQ-7020 board, where the PYNQ is an open-source project from AMD [37]. It embeds Xilinx ZYNQ-7020, and also provides a Jupyter-based framework with Python APIs. The PYNQ-Z1 board has the FPGA-SoC platform which is composed of PL and PS. The basic software development tool is Jupyter Notebook, a web-based software programming platform. It is also supporting Python, C/C++ programming languages, and other open-source libraries such as OpenCV. Our experiment environment is as follows in Figure 14 and Figure 15. The imported CNN architecture is the Resnet-20. It has 23 layers and it was trained with CIFAR-10 which provides a test set of 10,000 images in several formats. We used the provided weight file and converted the ONNX format of the ResNet Model [41]. ONNX, a machine learning (ML) model converter, provides the converted ML model code in ONNX format. Tensil compiler generates three import artifacts, a .tmodel, .tdata, and .tprog files. Once the .tmodel manifest for the model into the driver is loaded, it tells the driver where to locate the binary files, program data, and weights data. They were not open data and, we are using them without any modification, so that means the accuracy was not changed.

B. FPGA IMPLEMENTATION RESULTS

Compared with Tensil's optimization result, we verified more register buffers are activated for our proposed structure. Once we check the functionality and performance result, then you



HW Resource Report Comparison of Tensil Sample Simulation and Our Work tested on PYNQ-Z1

FIGURE 16. HW Resource Report Comparison of Tensil Sample Simulation and Our Work tested on PYNQ-Z1

can modify the structure by RTL code modification. Then we can improve the specific hardware resources and power consumption of the design. Analyzing the result leads to improved performance of CNN processing. Figure 16 shows the power consumption reduction of the processing system unit. We were able to archive the 43.9 (GOPs/W) as a power efficiency result, compared to other FPGA board implementations, it increased 1.37 times. the hardware resource utilization in DSPs is increased 2.2 times from the result of [24].

C. POWER CONSUMPTION RESULTS

Our optimization will decrease 16% of the dynamic power consumption. Also, the total On-Chip power will deduct 20% of the total power consumption. Once the global buffer is activated, the unused global clock buffer and the second global clock resource will help to improve the performance of the design. Moreover, this can be the solution to some high fan-out signals to make the device fully functional. In the pipeline logic, inserting an intermediate flip-flop(FF) can improve the working speed of the device, however, too many flip-flops make computational complexity. Our low-power techniques show better performance than the performance of FFs.

TABLE 1. Long Table

	[17]	[18]	[19]	[20]	[21]	[22]	[23]	[24]	Ours
Year	2018	2018	2018	2018	2017	2018	2019	2022	2022
CNN Model	AlexNet	MobileNetV2	VGG16	VGG16	VGG19	VGG16	AP2D-Net	ResNet20	ResNet20
FPGA	ZYNQ-XCZ7020	Intel Arria10-SoC	ZYNQ-XCZ7020	Virtex-7 VX690t	Stratix V GSMDS	Intel Arria 10	Ultra96	PYNQ-Z1	PYNQ-Z1
Clock (MHz)	200	133	214	150	150	200	300	50	50
BRAMs	268	1844**	85.5	1220	919**	2232**	162	198	523
DSPs	218	1278	190	2160	1036	1518	287	73	167
LUTs	49.8K	-	29.9K	-	-	-	54.3K	14.6K	15.2K
FFs	61.2K	-	35.5K	-	-	-	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)	0.016	0.004	0.364	0.106	0.107	0.043	0.032	0.178	0.109
Throughput (GOPs)	80.35	170.6	84.3	290	364.4	715.9	130.2	55.0	63.3
Power(W)	2.21	-	-	35	25	-	5.59	1.714	1.440
Power Efficiency (GOPs/W)	36.36	-	-	8.28	14.57	-	23.3	28.2	43.9

VI. CONCLUSION

In this article, the proposed highly reconfigurable FPGA hardware accelerator showed improved performance in terms of the processing speed and power consumption result during inference of various CNNs. The hardware optimization is conducted mainly for two purposes: to improve the throughput and to reduce power consumption. For improving performance, the minimized data transferring strategy was applied by assigning the maximum amount of buffers during the computations and by applying a controlled pipeline design for minimized data access. For achieving energy efficient results of CNN object detection operation, not only the data access controlling for minimized memory access, but also we proposed the RT level low power techniques-applied reconfigured MAC units such as advanced clock gating-applied adder, register Z with bus specific clock, and OR-based MAC architecture to RTL code of the proposed accelerator. The proposed hardware accelerator for ResNet-20 was implemented on mobile FPGA-SoC, PYNQ-Z1, and the power consumption was measured during inference operation. As a result, the throughput result showed a 15% improvement compared with the baseline RTL code of the accelerator, also power consumption was reduced by 16%, and hardware utilization was increased by 58%. The object detection processing speed was 9.17FPS, which shows that real-time processing is feasible in mobile FPGA.

ACKNOWLEDGEMENT

We thank our colleagues from KETI and KEIT who provided insight and expertise that greatly assisted the research and greatly improved the manuscript.

REFERENCES

[1] A. K. Jameil and H. Al-Raweshidy, "Efficient cnn architecture on fpga using high level module for healthcare devices," *IEEE Access*, vol. 10, pp. 60 486–60 495, 2022.

[2] S. Y. Nikouei, Y. Chen, S. Song, R. Xu, B.-Y. Choi, and T. Faughnan, "Smart surveillance as an edge network service: From harr-cascade, svm to a lightweight cnn," in *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*, 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 *ARCS Workshop 2019; 32nd International Conference on Architecture of Computing Systems*, 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 of Things Journal*, vol. 9, no. 21, pp. 21 357–21 369, 2022.

[5] C. Fu and Y. Yu, "Fpga-based power efficient face detection for mobile robots," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2019, pp. 467–473.

[6] X. Li, X. Gong, D. Wang, J. Zhang, T. Baker, J. Zhou, and T. Lu, "Abm-spconv-simd: Accelerating convolutional neural network inference for industrial iot applications on edge devices," *IEEE Transactions on Network Science and Engineering*, pp. 1–1, 2022.

[7] S. Tamimi, Z. Ebrahimi, B. Khaleghi, and H. Asadi, "An efficient sram-based reconfigurable architecture for embedded processors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 3, pp. 466–479, 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. 151 897–151 911, 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 Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 5, pp. 888–897, 2019.

[10] S. Ullah, S. Rehman, M. Shafique, and A. Kumar, "High-performance accurate and approximate multipliers for fpga-based hardware accelerators," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 2, pp. 211–224, 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 Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 3, pp. 1185–1198, 2022.

[12] W. Liu, J. Lin, and Z. Wang, "A precision-scalable energy-efficient convolutional neural network accelerator," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 10, pp. 3484–3497, 2020.

[13] H. Irmak, D. Ziener, and N. Alachiotis, "Increasing flexibility of fpga-based cnn accelerators with dynamic partial reconfiguration," in *2021 31st International Conference on Field-Programmable Logic and Applications (FPL)*, 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 *2018 IEEE International Conference on Electron Devices and Solid State Circuits (EDSSC)*, 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 *2019 IEEE International Conference on Integrated Circuits, Technologies and Applications (ICTA)*, 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 *2018 IEEE 23rd International Conference on Digital Signal Processing (DSP)*, 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 Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2601–2612, 2018.

[18] L. Bai, Y. Zhao, and X. Huang, "A cnn accelerator on fpga using depthwise separable convolution," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 10, pp. 1415–1419, 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 *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 26–35. [Online]. Available: https://doi.org/10.1145/2847263.2847265

[20] T. Geng, T. Wang, A. Sanaullah, C. Yang, R. Patel, and M. Herbordt, "A framework for acceleration of cnn training on deeply-pipelined fpga clusters with work and weight load balancing," in *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*, 2018, pp. 394–3944.

[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 *2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2017, pp. 152–159.

[22] Y. Ma, Y. Cao, S. Vrudhula, and J.-s. Seo, "Optimizing the convolution operation to accelerate deep neural networks on fpga," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 7, pp. 1354–1367, 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. 105 455–105 471, 2020.

[24] "Learn tensil with resnet and pynq z1," https://www.tensil.ai/docs/tutorials/resnet20-pynqz1/, accessed: 2022-12-15.

[25] X. Zhang, Y. Ma, J. Xiong, W.-M. W. Hwu, V. Kindratenko, and D. Chen, "Exploring hw/sw co-design for video analysis on cpu-fpga heterogeneous systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 6, pp. 1606–1619, 2022.

[26] E. Antolak and A. Pułka, "Energy-efficient task scheduling in design of multithread time predictable real-time systems," *IEEE Access*, vol. 9, pp. 121 111–121 127, 2021.

[27] W. Huang, H. Wu, Q. Chen, C. Luo, S. Zeng, T. Li, and Y. Huang, "Fpga-based high-throughput cnn hardware accelerator with high computing resource utilization ratio," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 8, pp. 4069–4083, 2022.

[28] Y. Ma, Y. Cao, S. Vrudhula, and J.-s. Seo, "Optimizing the convolution operation to accelerate deep neural networks on fpga," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 7, pp. 1354–1367, 2018.

[29] G. Lakshminarayanan and B. Venkataramani, "Optimization techniques for fpga-based wave-pipelined dsp blocks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 7, pp. 783–793, 2005.

[30] D. Wang, K. Xu, J. Guo, and S. Ghiasi, "Dsp-efficient hardware acceleration of convolutional neural network inference on fpgas," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 12, pp. 4867–4880, 2020.

[31] A. Prihozhy, E. Bezati, A. A.-H. Ab Rahman, and M. Mattavelli, "Synthesis and optimization of pipelines for hw implementations of dataflow programs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1613–1626, 2015.

[32] W. Lou, L. Gong, C. Wang, Z. Du, and X. Zhou, "Octcnn: A high throughput fpga accelerator for cnns using octave convolution algorithm," *IEEE Transactions on Computers*, vol. 71, no. 8, pp. 1847–1859, 2022.

[33] Y. Kim, H. Kim, N. Yadav, S. Li, and K. K. Choi, "Low-power rtl code generation for advanced cnn algorithms toward object detection in autonomous vehicles," *Electronics*, vol. 9, no. 3, 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/3/478

[34] H. Kim and K. Choi, "Low power fpga-soc design techniques for cnn-based object detection accelerator," in *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, 2019, pp. 1130–1134.

[35] ——, "The implementation of a power efficient bcnn-based object detection acceleration on a xilinx fpga-soc," in *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2019, pp. 240–243.

[36] Y. Kim, Q. Tong, K. Choi, E. Lee, S.-J. Jang, and B.-H. Choi, "System level power reduction for yolo2 sub-modules for object detection of future autonomous vehicles," in *2018 International SoC Design Conference (ISOCC)*, 2018, pp. 151–155.

[37] "Pynq: Python productivity," http://www.pynq.io/, accessed: 2023-2-15.

[38] L. Li, K. Choi, S. Park, and M. Chung, "Selective clock gating by using wasting toggle rate," in *2009 IEEE International Conference on Electro/Information Technology*, 2009, pp. 399–404.

[39] W. Wang, Y.-C. Tsao, K. Choi, S. Park, and M.-K. Chung, "Pipeline power reduction through single comparator-based clock gating," in *2009 International SoC Design Conference (ISOCC)*, 2009, pp. 480–483.

[40] Y. Zhang, Q. Tong, L. Li, W. Wang, K. Choi, J. Jang, H. Jung, and S.-Y. Ahn, "Automatic register transfer level cad tool design for advanced clock gating and low power schemes," in *2012 International SoC Design Conference (ISOCC)*, 2012, pp. 21–24.

[41] "Compile an ml model," https://www.tensil.ai/docs/howto/compile/, accessed: 2023-2-15.

HEEKYUNG KIM received the B.S. in electronic and electrical engineering from Hongik University, Seoul, Korea, in 2012 and M.S. degrees in electrical and computer engineering from the Illinois Institute of Technology, Chicago, in 2015. She is currently pursuing a Ph.D. degree in computer engineering from the Illinois Institute of Technology, Chicago.

Her current research interests include low-power and high-performance HW/SW optimization for CNN accelerator design for FPGA and ASIC, wireless sensor networks design and sensor data analysis and embedded HW/SW system design for robotics, drones, IoTs.

KYUWON KEN CHOI (Senior Member, IEEE) received the Ph.D. degree in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, in 2002. During the Ph.D. degree, he proposed and conducted several projects supported by NASA, DARPA, NSF, and SRC regarding power-aware computing and communication. Since 2004, he had been with the Takayasu Sakurai Laboratory, The University of Tokyo, Japan, as a Postdoctoral Research Associate, working on leakage power reduction circuit techniques. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, Illinois Institute of Technology. He was a Senior CAD Engineer and a Technical Consultant for low-power system-on-chip (SoC) design in Samsung Semiconductor, Broadcom, and Sequence Design, prior to joining IIT. In the past, he had eight-year industry experience in the area of VLSI chip design from compiler level to circuit level. Last few years, by using his novel low-power techniques, everal processor, and control VLSI chips were successfully fabricated in deep-submicrometer CMOS technology and more than 80 peer-reviewed journals and conference papers have been published. He is currently the Director of the VLSI Design and Automation Laboratory (DA-Lab), IIT.

His research interests include ultra-low power circuit and IC design for multimedia, and mobile or energy harvesting applications. He is also a TPC member for several IEEE circuit design conferences. For last seven years, DA-Lab has been awarded several projects about hardware design for modern applications, including IoT, AI, deep learning, unmanned vehicles, and energy harvesting, about 1.6 million U.S. dollars from U.S. federal agencies and Korea Government. He has served as the President in the Korean-American Scientists and Engineers Association (KSEA)-Chicagoland Chapter. He is currently the Technical Group Director and an Advisory Election Committee in the KSEA Head Quarter. He is also the Editor-in-Chief of the Journal of Pervasive Technologies and a Guest Editor of Springer and Wiley Journals.