



بسم الله الرحمن الرحيم

∞∞∞∞

تم رفع هذه الرسالة بواسطة / سامية زكى يوسف

بقسم التوثيق الإلكتروني بمركز الشبكات وتكنولوجيا المعلومات دون أدنى

مسئولية عن محتوى هذه الرسالة.

ملاحظات: لا يوجد





Cairo University

EFFICIENT HIGH LEVEL SYNTHESIS IMPLEMENTATION FOR DEEP CONVOLUTIONAL NEURAL NETWORKS ON FPGA

By

Muhammad Nabil Muhammad Sarg

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Electronics and Communications Engineering

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2022

**EFFICIENT HIGH LEVEL SYNTHESIS
IMPLEMENTATION FOR DEEP CONVOLUTIONAL
NEURAL NETWORKS ON FPGA**

By
Muhammad Nabil Muhammad Sarg

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Electronics and Communications Engineering

Under the Supervision of

Prof. Ahmed Hussein Mohamed

Dr. Hassan Mostafa Hassan

.....

.....

Professor
Electronics and Communications
Department
Faculty of Engineering, Cairo University

Assistant Professor
Electronics and Communications
Department
Faculty of Engineering, Cairo University

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2022

**EFFICIENT HIGH LEVEL SYNTHESIS
IMPLEMENTATION FOR DEEP CONVOLUTIONAL
NEURAL NETWORKS ON FPGA**

By
Muhammad Nabil Muhammad Sarg

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Electronics and Communications Engineering

Approved by the
Examining Committee

Prof. Ahmed Hussein Mohamed Khalil, Thesis Main Advisor

Dr. Hassan Mostafa Hassan Mostafa, Advisor

Prof. Ahmed Nader Mohiieldin, Internal Examiner

Prof. Ahmed Hassan Kamel Madian, External Examiner
- Atomic Energy Authority and Nile University

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2022

Engineer's Name: Muhammad Nabil Muhammad Sarg
Date of Birth: 1/1/1990
Nationality: Egyptian
E-mail: Muhammad.sarg@gmail.com
Phone: 01024307849
Address: Electronics and Communications
Engineering Department,
Cairo University, Giza 12613, Egypt
Registration Date: 1/10/2016
Awarding Date:/....../2022
Degree: Master of Science
Department: Electronics and Communications Engineering



Supervisors:
Prof. Ahmed Hussein Mohamed Khalil
Dr. Hassan Mostafa Hassan Mostafa

Examiners:
Prof. Ahmed Hussein Mohamed Khalil (Thesis main advisor)
Dr. Hassan Mostafa Hassan Mostafa (advisor)
Prof. Ahmed Nader Mohieldin (Internal examiner)
Prof. Ahmed Hassan Kamel Madian (External examiner)
(Atomic Energy Authority and Nile University)

Title of Thesis:
Efficient High Level Synthesis Implementation for Deep Convolutional Neural
Networks on FPGA

Key Words:
Convolutional Neural Network; Field Programmable Gate Array; Hardware
Accelerator; High-Level Synthesis

Summary:
Deep Convolutional Neural Networks (CNNs) play an important role in computer vision applications. However, they are computational-intensive and resource-consuming. Thus, it is hard to integrate them into embedded platforms. The FPGA is one of the most promising platforms for accelerating Deep CNNs due to its configurability, high-performance, low power, and shorter development cycles. In this research, we introduce a diverse High Level Synthesis (HLS) C++ Deep CNNs compiler that generates highly efficient implementation for accelerating the inference task of Deep CNNs on FPGAs. We developed HLS C++ implementations for the Deep CNNs blocks that efficiently utilizes the FPGA available resources to achieve the maximum inference performance. The developed compiler automates the customization of the accelerator HLS implementation to best fit the selected Deep CNN model. The proposed work is demonstrated with an implementation of two different Deep CNN models, Resnet-50 and VGG16, on Xilinx Zynq MPSoC using SDSoC development environment.

Disclaimer

I hereby declare that this thesis is my own original work and that no part of it has been submitted for a degree qualification at any other university or institute.

I further declare that I have appropriately acknowledged all sources used and have cited them in the references section.

Name: Muhammad Nabil Muhammad Sarg Date: ../ ../2022

Signature: Muhammad Nabil Muhammad Sarg

Dedication

To my parents, my wife, and my little child.

Acknowledgments

I would like to offer my sincere thanks to my supervisors Prof. Ahmed Hussien and Dr. Hassan Mostafa. I'd want to convey my heartfelt thanks to Dr. Hassan for his invaluable guidance, encouragement, and active assistance during this project.

My heartfelt thanks and love to my family for their support and encouragement.

Table of Contents

DISCLAIMER.....	I
DEDICATION.....	II
ACKNOWLEDGMENTS	III
TABLE OF CONTENTS.....	IV
LIST OF TABLES	VII
LIST OF FIGURES	VIII
NOMENCLATURE	X
ABSTRACT	XI
CHAPTER 1 : INTRODUCTION	12
1.1. MOTIVATION.....	12
1.2. CONTRIBUTION	13
1.3. ORGANIZATION OF THE THESIS	13
CHAPTER 2 : CONVOLUTIONAL NEURAL NETWORKS	14
2.1. INTRODUCTION	14
2.2. ARTIFICIAL NEURAL NETWORK.....	14
2.3. INFERENCE VS. TRAINING	16
2.4. CONVOLUTIONAL NEURAL NETWORKS.....	17
2.5. CNN BUILDING BLOCKS.....	18
2.5.1. Convolution Layer	18
2.5.2. Batch Normalization and Scaling Layer	19
2.5.3. Activation Functions	19
2.5.3.1. The Rectified Linear Unit (ReLU)	19
2.5.3.2. Softmax.....	19
2.5.4. Pooling Layer.....	20
2.5.5. Fully Connected Layer.....	20
2.5.6. Element-Wise Layer	20
2.6. IMAGENET CHALLENGE	21
2.7. POPULAR CNNs.....	21
2.7.1. AlexNet	21
2.7.2. VGG-16.....	22
2.7.3. GoogLeNet.....	22
2.7.4. ResNet-50	22
CHAPTER 3 : FIELD PROGRAMMABLE GATE ARRAY AND SYSTEM ON CHIP	28
3.1. INTRODUCTION	28
3.2. FPGAs vs. GENERAL PURPOSE PROCESSORS	28

3.3.	FPGAs vs. ASICs	28
3.4.	FPGA STRUCTURE	29
3.4.1.	Configurable Logic Block.....	29
3.4.2.	Block RAM.....	29
3.4.3.	UltraRAM	29
3.4.4.	Digital Signal Processor.....	30
3.5.	SYSTEM ON CHIP	31
3.5.1.	SoC Features	31
3.5.2.	SoC Applications	31
3.5.3.	Xilinx Zynq UltraScale+ MPSoC	31
CHAPTER 4 : DEVELOPMENT LANGUAGE AND TOOLS		34
4.1.	INTRODUCTION	34
4.2.	HIGH-LEVEL SYNTHESIS	34
4.2.1.	HLS Features	34
4.2.2.	HLS Challenges	35
4.2.3.	HLS Tools.....	35
4.2.3.1.	Vivado HLS	35
4.2.4.	HLS Optimization Directives.....	36
4.2.4.1.	Pipelining.....	36
4.2.4.2.	Unrolling.....	37
4.2.4.3.	Array Partitioning	37
4.2.4.4.	Resources Specification	38
4.2.4.5.	Interfaces.....	38
4.2.4.6.	Specifying Memory Blocks.....	38
4.2.5.	HLS Data Types.....	38
4.3.	SOFTWARE DEFINED SYSTEM-ON-CHIP	39
4.3.1.	SDSoC Features	39
4.3.2.	Design Flow	40
4.3.2.1.	SDSoC Optimization Directives	41
4.4.	ML DEVELOPMENT FRAMEWORKS	41
4.4.1.	Keras	41
CHAPTER 5 : IMPLEMENTATION AND EVALUATION		42
5.1.	INTRODUCTION	42
5.2.	COMPILER	42
5.2.1.	Compiler Components	42
5.2.1.1.	Module.....	43
5.2.1.2.	Scheduler	43
5.2.1.3.	Parameters.....	43
5.2.1.4.	Writer.....	43
5.2.1.5.	Builder	43
5.2.1.6.	Accelerator.....	43
5.3.	THE PROPOSED CNN ACCELERATOR.....	46
5.3.1.	Accelerator Architecture	46
5.3.2.	Memory	48
5.3.2.1.	URAM Utilization	48
5.3.3.	Convolution Loops Optimization.....	49
5.3.3.1.	Loop Tiling	49
5.3.3.1.1.	Stride > 1 and Kernel = 1	50

5.3.3.1.2.	Stride > 1 and kernel > 1	50
5.3.3.2.	Loop Transformation	53
5.3.3.3.	Loop Pipelining and Unrolling.....	53
5.3.4.	Tensors Tiling	53
5.3.4.1.	Weights Reordring	54
5.3.4.2.	Using Multiple Weights Ports.....	55
5.3.4.3.	Weights Packetization.....	56
5.3.5.	The Pooling Layer.....	57
5.3.6.	The Element-Wise Layer	57
5.3.7.	The Rectified Linear Unit (ReLU).....	58
5.3.8.	The Fully Connected Layer.....	58
5.3.9.	The Softmax Function.....	59
5.4.	EVALUATION	60
5.4.1.	Design Flow	60
5.4.2.	Experimental Results	61
CHAPTER 6 : DISCUSSION AND CONCLUSION.....		65
6.1.	CONCLUSION.....	65
6.2.	FUTURE WORK	65
LIST OF PUBLICATIONS.....		67
REFERENCES.....		68
APPENDIX A: EVALUATION BOARD		72
APPENDIX B: ACCELERATOR HLS C++ CODE		74
APPENDIX C: COMPILER CODE		86

List of Tables

Table 2.1: Shape Parameters of A CNN Layer	18
Table 2.2: Summary of Popular CNNs.....	21
Table 2.3: ResNet-50 vs. ResNet-152	23
Table 3.1: Block RAM and UltraRAM comparison [31].....	30
Table 5.1: An example for the obtained values of the ofmaps tile height in ResNet; $P_{oy} = 7$, $B_{iy} = 7$	52
Table 5.2: An example for the obtained values of the ifmaps tile height in ResNet; $P_{oy} = 7$, $T_{oy} = 7$	52
Table 5.3: Tracing of the tiling pseudo code presented in figure 5.11	56
Table 5.4: The relation between the bus width and the achievable burst size [43].....	56
Table 5.5: Resources Consumption of Xilinx Floating Point IP Cores.....	62
Table 5.6: Inference Timing on PS.....	63
Table 5.7: Acceleration Results.....	63
Table 5.8: Comparison With Existing HLS FPGA-Based Accelerators.....	64
Table A.1: ZCU104 Key Features.....	73

List of Figures

Figure 2.1: An example of a four-layer feed-forward neural network with one input layer, three hidden layers, and an output layer of five neurons [5]	14
Figure 2.2a: A Biological Neuron [23], [5]	15
.....	15
Figure 2.2b: Artificial neuron or hidden unity where x_i , w_i , $f(.)$, and b are the activations, weights, nonlinear function, and bias, respectively [3]	15
Figure 2.3a: Simple neural network example and terminology, Neurons and synapses	16
.....	16
Figure 2.3b: Simple neural network example and terminology, Compute weighted sum for each layer [3]	16
Figure 2.4: The LeNet-5 network's architecture, which performs well on calssifying digits [46]	17
Figure 2.5: The learned features for the digit '7' [46]	17
Figure 2.6: An example for the convolution operation with an ifmaps tensor and weights tensor to produce the ofmaps tensor [42]	18
Figure 2.7: Pseudo code for the convolution operation (assuming the output buffer is initialized with the bias). 'S' is the convolution stride	19
Figure 2.8: An example for max pooling with a window of 2x2 and a stride of 2	20
Figure 2.9: AlexNet CNN Architecture [28]	22
Figure 2.10: VGG-16 CNN Architecture	24
Figure 2.11: GoogLeNet CNN Architecture [29]	25
Figure 2.12: The first layers of ResNet-50 model that contains multiple parallel branches involving different types of layers constructing a DAG	26
Figure 2.13: Different ResNet Architectures [2]	27
Figure 3.1: Basic DSP48E1 Slice Functionality [32]	30
Figure 3.2: Xilinx Zynq UltraScale MPSoCs Block Diagram (EV variant) [34]	32
Figure 3.3: Top Level Block Diagram of Xilinx Zynq UltraScale+ MPSoC [3333]	33
Figure 4.1: An example for using arbitrary precision data types [43]	39
Figure 4.2: SDSoC Design Flow [3636]	40
Figure 4.3: An example that shows the use of SDS data zero_copy pragma with the HLS interface pragma	41
Figure 5.1: The developed compiler view	44
Figure 5.2: The developed compiler directory tree.	44
Figure 5.3: An example that shows the assignment of some layers into modules by the developed compiler. (a) Shows the layers of the model. (b) Shows the assignment of these layers into modules	45
Figure 5.4: Architecture of the developed accelerator	47
Figure 5.5: Pseudo code for performing ReLU and copying the ofmaps tensor tile to the DRAM	47
Figure 5.6: Pseudo code for the convolution operation (assuming the output buffer is initialized with the bias). 'S' is the convolution stride	48
Figure 5.7: Code for creating and reading a URAM word for floating point data.	49
Figure 5.8a: An example for reshaping a single 5x5 ifmap to 4 ifmaps of 3x3 ('x' denotes a 'don't-care')	51
Figure 5.8b: An example for padding and reshaping a single weights window of 5x5 to 4 weights windows of 3x3	51

Figure 5.9: Pseudo code for the tiled convolution operation.....	53
Figure 5.10: An example for tiling tensors.....	54
Figure 5.11: Pseudo code for looping over tiles for convolution.	54
Figure 5.12: Illustrating the reordering of the weights elements for a weights tile of 2 1x1 kernels.....	55
Figure 5.13: An example for splitting weights on two streams.....	55
Figure 5.14: An example for distributing weights elements in packets where each packet contains two elements.	57
Figure 5.15: An example for reshaping the ifmaps and kernels tensor of a FC layer to execute it as a Conv. layer	59
Figure 5.16: C++ code for the softmax function.	60
Figure A.1: Xilinx Zynq ZCU104	72

Nomenclature

AI	Artificial Intelligence
ANN	Artificial Neural Networks
ASIC	Application-Specific Integrated Circuit
BN	Batch Normalization
BRAM	Block Random Access Memory
CNN	Convolutional Neural Network
CPU	Central Procession Unit
DL	Deep Learning
DRAM	Double Random Access Memory
DSP	Digital Signal Processing
FC	Fully Connected
FPGA	Field-Programmable Gate Array
GPU	Graphical Procession Unit
HLS	High-Level Synthesis
IFMAP	Input Feature Map
LUT	Look Up Table
MAC	Multiply and Accumulate
MPSoC	Multi Programmable System on Chip
OFMAP	Output Feature Map
RTL	Register Transfer Logic
URAM	Ultra Random Access Memory