

HARDWARE AND SOFTWARE CO-SIMULATION PLATFORM FOR  
CONVOLUTION OR CORRELATION BASED IMAGE PROCESSING  
ALGORITHMS

SAYED OMID AYAT

UNIVERSITI TEKNOLOGI MALAYSIA

HARDWARE AND SOFTWARE CO-SIMULATION PLATFORM FOR  
CONVOLUTION OR CORRELATION BASED IMAGE PROCESSING  
ALGORITHMS

SAYED OMID AYAT

A project report submitted in partial fulfillment  
Of the requirements for the award of the degree of  
Master of Engineering (Electrical - Computer and Microelectronics System)

Faculty of Electrical Engineering  
Universiti Teknologi Malaysia

JUNE 2014

My genuine dedications to,  
My beloved wife, father and mother;  
My enduring and experienced supervisor;  
Who are always there for me,  
Every step of the way,  
Thanks!

## ACKNOWLEDGEMENT

First and foremost, I would like to convey my deepest gratitude to my supervisor, Prof. Dr. Mohamed Khalil, who has been the most influential person who always guide, motivate, support, and advise me throughout my thesis completion with his inspiring and constructive comments and valuable feedbacks.

Furthermore, I would like to extend my deepest and sincere appreciation to my beloved family, especially to my wife, who always support me, encourage me, and listen to me along my research. They are my source of inspiration and my greatest refuge.

Finally, I would like to pay special tribute to my seniors, Miss Lee, Mr. Sia, Mr. Liew and my cherished friends, who are willing to guide me with patience and keenness. Thank you so much for deserving a lot of time and passion to guide, teach, motivate and share their experiences and knowledge with me.

## ABSTRACT

Software implementation of image processing algorithms in which convolution or correlation is applied is too slow to be real-time. As long as the system design gets larger, it should be partitioned into two parts: software and hardware. In order to achieve real time performance, it is essential to map the fast convolution or correlation module, which is the heaviest computation intensive part, in hardware instead of software. Our test case is “generic image pre-processing algorithm” which includes resizing, noise filtering and normalization. In noise filtering part of the preprocessing algorithm in which convolution is used should be implemented in hardware while the rest of the preprocessing algorithm stays in software. Next, to verify our hardware/design software we can deploy it on FPGA board, but it is very time consuming and involves a lot of technical complexities. In that case, this design used hardware/software co-simulation and direct programming interface (DPI-C) whereas it allows System Verilog calls C functions and vice versa. The proposed work has overcome the problems faced when running a co-simulation based on Modelsim simulated using direct programming interface (DPI) technique.

## ABSTRAK

Pelaksanaan perisian algoritma pemprosesan imej di mana kekusutan atau korelasi digunakan adalah terlalu lambat untuk dilaksanakan dalam masa nyata. Apabila reka bentuk sistem bertambah besar, proses berkenaan harus dibahagikan kepada dua bahagian: perisian dan perkakasan. Untuk mencapai prestasi masa nyata, ia adalah penting untuk memetakan modul kekusutan atau modul korelasi, yang merupakan bahagian pengiraan yang intensif, dalam perkakasan dan bukannya perisian. Kes ujian kami adalah algoritma pra-pemprosesan imej yang generik termasuk penyelerasan saiz, penapisan hingar dan normalisasi. Bahagian kekusutan dalam modul penapisan hingar telah direkakan sebagai perkakasan manakala modul-modul lain dalam algoritma pra-pemprosesan imej kekal sebagai perisian. Seterusnya, untuk mengesahkan rekabentuk perkakasan / perisian berkenaan, papan *Field-Programmable Gate Arrays* boleh digunakan tetapi proses tersebut mengambil masa yang panjang dan melibatkan banyak kerumitan teknikal. Dalam kes itu, reka bentuk ini menggunakan simulasi perkakasan / perisian dengan teknik antara muka pengaturcaraan langsung (DPI-C) yang membolehkan panggilan fungsi C dari Sistem Verilog dan sebaliknya. Kerja yang dicadangkan telah mengatasi masalah yang dihadapi apabila melaksanakan simulasi bersama berdasarkan simulator Modelsim yang menggunakan antara muka pengaturcaraan langsung teknik (DPI).

## TABLE OF CONTENTS

CHAPTER	TITLE	PAGE
	<b>DECLARATION</b>	i
	<b>DEDICATION</b>	vi
	<b>ACKNOWLEDGEMENTS</b>	vi
	<b>ABSTRACT</b>	iv
	<b>ABSTRAK</b>	v
	<b>TABLE OF CONTENTS</b>	vi
	<b>LIST OF TABLES</b>	ix
	<b>LIST OF FIGURES</b>	vi
	<b>LIST OF ABBREVIATIONS</b>	vi
	<b>LIST OF APPENDICES</b>	
<b>1</b>	<b>INTRODUCTION</b>	1
	1.1 Background of Study	1
	1.2 Problem Statements	2
	1.4 Objectives	3
	1.5 Scope of the Research	4
	1.6 Expected Contribution	5
<b>2</b>	<b>BACKGROUND AND LITERATURE REVIEW</b>	7
	2.1 Convolution and Correlation	7
	2.1.1 Introduction to Convolution and Correlation	7
	2.1.2 Convolution and Correlation Algorithmic behavior	10
	2.2 Pattern Recognition algorithm	13

2.3	Literature Review	15
2.3.1	Previous Works on Hardware Design of Convolution and AGU	15
2.3.2	Previous Works on Co-simulation using System Verilog	20
<b>3</b>	<b>DESIGN METHODOLOGY</b>	<b>22</b>
3.1	Project Overall workflow	22
3.2	Software Tools and Design Environment	23
3.2.1	MATLAB	24
3.2.2	Dev-C++	24
3.2.3	Quartus	24
3.2.4	Mentor Graphic ModelSim	25
3.3	Direct Programming Interface	25
3.4	DPI Examples	26
3.4.1	Example 1: Imported task	27
3.4.2	Example 2: Imported Methods	27
3.4.3	Example 3: Data Types and Exported methods	30
3.4.4	Example 4: Time Issue in Co-Simulation	32
3.4.5	Example 5: Pure and Context	33
3.4.6	Example 6: Open Arrays	35
3.4.7	Example 7: Packed Arrays	37
<b>4</b>	<b>DESIGN IMPLEMENTATION</b>	<b>39</b>
4.1	Software Implementation of Image Preprocessing Algorithm	39
4.1.1	Software Implementation of Resizing Algorithm	42
4.1.2	Software Implementation of Convolution Algorithm	42
4.1.3	Software Implementation of Normalization Algorithm	44
4.2	Hardware Implementation of Convolution and Correlation module	46



4.2.1	Datapath Unit	47
4.2.2	Control Unit	52
4.2.3	Address Generation Unit	53
4.3	Software and Hardware Platform	56
4.3.1	Test case 1 (ModelSim Evaluation)	57
4.3.2	Software and Hardware Platform Development for Pre-processing Algorithm	61
<b>5</b>	<b>RESULTS AND DISCUSSION</b>	<b>68</b>
5.1	Pure Software Implementation Results	68
5.2	Results for Hardware Implementation of Convolution and Correlation Module	70
5.2.1	Simulation Results for AGU and Top Module	71
5.2.2	Matlab Output	77
5.3	Hardware and Software Co-Simulation Results	79
5.3.1	Co-Simulation Results	80
<b>6</b>	<b>CONCLUSION AND RECOMMENDATION</b>	<b>88</b>
6.1	CONCLUSION	88
6.2	BENCHMARKING	89
6.3	RECOMMENDATION	91
	<b>REFERENCES</b>	<b>91</b>
	APPENDICES A-E	92-115

**LIST OF TABLES**

<b>TABLE NO.</b>	<b>TITLE</b>	<b>PAGE</b>
2.1	Data Matching between System Verilog and C	30
5.1	Difference Between Matlab and Modelsim error	79
6.1	Hardware Benchmarking	91

## LIST OF FIGURES

<b>FIGURE NO.</b>	<b>TITLE</b>	<b>PAGE</b>
1.1	Typical Pattern Recognition Algorithm	5
2.1	Example of Kernel Coefficients for Gaussian Filter	8
2.2	Example of Kernel Coefficients for Average filter	9
2.3	Example of Kernel Coefficients for Sharpening	9
2.4	2D Convolution Operation	10
2.5	Padding The Image with One Edge Pixel	12
2.6	Shifting the kernel	13
2.7	Typical Pattern Recognition Algorithm	14
2.8	Convolution Hardware by Using Shiftregistr	15
2.9	FBD of 2D Convolution Computation Unit	16
2.1	Circular Buffer to feed in the Input Data	17
2.11	Register Chain or FIFO	17
2.12	3×3 Kernel and 18×18 Image Size	18
2.13	ZiqZaq Movement of AGU for Circular Buffer	19
2.14	Simulation Of Circular Buffer Behavior	19
2.15	Processing Unit for Circular Buffer	20
2.16	Testbench matching the Real Application	20
2.17	Hardware-Software Communication Method	21
3.1	Project Overall Workflow	23

3.2	DPI Example 1	27
3.3	Simulation Output for Example 1	27
3.4	DPI Example 2: Imported Methods, part A	28
3.5	DPI Example 2: Imported part B	29
3.6	Co-Simulation Output for Example 2: Imported	29
3.7	Logic Data Type in C	31
3.8	Codes for Example 3: Data Types and Exported methods	32
3.9	Co-Simulation Output for Example 3: Data Types and Exported methods	32
3.1	Codes for Example 4: Time Issue in Co-Simulation	33
3.11	Co-Simulation Output for Example 4: Time Issue in Co-Simulation	33
3.12	Codes for Example 5: Pure and Context _1	34
3.13	Codes for Example 5: Pure and Context_2	35
3.14	Co-Simulation results of Example 5: Pure and Context	35
3.15	Codes for Example 6: Open Arrays	36
3.16	Co-Simulation Result for Example 6: Open Arrays	37
3.17	Codes for Example 7: Packed Arrays	38
3.18	Co-Simulation Result for Example 7	38
4.1	Typical Pattern recognition algorithm	40
4.2	Preprocessing algorithm	41
4.3	Sample Input Image into resizing function	41
4.4	Output Image from Resizing function	41
4.5	5×5 Input image to Resizing sub algorithm	42
4.6	10×10 output image from Resizing sub algorithm	42

4.7	Gaussian Filter Coefficients	43
4.8	12×12 Padded Image in Convolution Subroutin	43
4.9	10×10 2D Convolution Output	44
4.1	10×10 Normalization Output Image	45
4.11	Software Implementation of Preprocessing Algorithm for 5×5 Input Image	45
4.12	2D Convolution or 2D Correlation Operation.	46
4.13	Proposed DFG for Adder Tree Computation Unit	47
4.14	FBD of Top Module for 2D Convolution and Correlation Hardware	48
4.15	FBD of Data Unit For 8×8 input Image and 3×3 kernel size	49
4.16	Typical 3×3 Kernel Coefficients	50
4.17	Modelsim Transcript Output	51
4.18	Modelsim Waveform Output	51
4.19	System Verilog Code for "Adder Chain"	52
4.2	Example of Kernel Coefficients for Gaussian Filter	52
4.21	ASM chart for CU	53
4.22	Address Generation Unit	54
4.23	ASM chart of AGU	56
4.24	Pure Software Implementation of Preprocessing Algorithm	57
4.25	Software and Hardware Implementation of Preprocessing Algorithm	57
4.26	Top Module Function Block Diagram	58
4.27	Behavioral Algorithm for Test case 1 (ModelSim Evaluation)	58
4.28	Convolution Module with Avalon Interface	59

4.29	Outputs on ModelSim Transcript	60
4.3	Block Diagram of Resizing Algorithm (in software) Integrated with Convolution Module (In Hardware).	61
4.31	Integration Summary of Resizing Algorithm ( in software) with Convolution Module (in Hardware).	62
4.32	Convolution / Convolution Module with Avalon Interface	63
4.33	Interface Behavior Algorithm	64
4.34	Timing Issue of the Pre-Processing Algorithm	65
4.35	Hardware-Software Block Diagram	66
5.1	Input 5×5 Image Data	68
5.2	10×10 Resized Input Image	69
5.3	12×12 Padded Image	69
5.4	10×10 Convolution Output	70
5.5	10×10 Normalization Output	70
5.6	ModelSim Output for WIDTH=8, HEIGHT=4, HEIGHT_BITS=2, KERNEL_SIZE=3 (End of the Image)	72
5.7	ModelSim Output for WIDTH=8, HEIGHT=4,  HEIGHT_BITS=2, KERNEL_SIZE=3	72
5.8	ModelSim Output for WIDTH=8, HEIGHT=4, KERNEL_SIZE=5	73
5.9	ModelSim Output for WIDTH=8, HEIGHT=4, KERNEL_SIZE=5 (End of the Image)	73
5.1	ModelSim Output for WIDTH=8, HEIGHT=4, KERNEL_SIZE=7×7	74
5.11	ModelSim Output for WIDTH=8, HEIGHT=4, KERNEL_SIZE=7×7 (End of the Image)	74
5.12	ModelSim Output for Image size= 200x128, kernel size=3x3 (zoomed out)	76

5.13	ModelSim Output for Image size=200x128 kernel size=3x3(zoomed in)	76
5.14	Comparison Between Matlab and ModelSim Output	78
5.15	Difference between Resized Padded Co-Simulation (Modelsim) Output and Padded Software (Desired) Output	80
5.16	ModelSim Co-Simulation Output 1	82
5.17	ModelSim Co-Simulation Output 2	83
5.18	Difference between Co-Simulation (Modelsim) Output and Software (Desired) Output of Convolution Algorithm	84
5.19	Sending the Data to the Software	85
5.2	ModelSim Transcript output for Padded Resized 12x12 Image	86
5.21	ModelSim Transcript Output for Convolution 10x10 Output	86
5.22	ModelSim Transcript output for Normalization 10x10 Output Image	87
5.23	Difference Between Co-Simulation Output and Desired Software Output for Normalization Algorithm	87
6.1	Critical path delay	89
6.2	Maximum frequency	89
6.3	Spending Time for Co-Simulation	90
6.4	Using Text File as an Interface between Hardware and Software	91
6.5	Timing Performance Comparison	91

**ABBREVIATIONS**

AGU	-	Address generation unit
ALU	-	Arithmetic logic unit
ASM	-	Algorithmic state machine
C	-	C programming language
CU	-	Control Unit
DFG	-	Data flow graph
DPI	-	Direct programming interface
DU	-	Data path unit
FPGA	-	Field programmable gate array
HDL	-	Hardware description language
HDVL	-	Hardware description and verification language
IDE	-	Integrated development environment
ISS	-	Instruction set simulator
MATLAB	-	Matrix laboratory
PLI	-	Programming language interface
RAM	-	Random access memory
RGB	-	Red Green Blue
RTL	-	Register transfer level
DFG	-	data flow graph
SoC	-	System-on-chip



SV	-	SystemVerilog
VLI	-	Verilog Procedural Interface
2D	-	2-dimensional

**LIST OF APPENDICES**

<b>APPENDIX</b>	<b>TITLE</b>	<b>PAGE</b>
A	Source Codes For Co-Simulation Examples	93
B	Source Codes for Hardware Implementation of Convolution and Correlation module	98
C	Source Codes for Software Implementation of Image Preprocessing Algorithm	104
D	Source Code for Hardware and Software Cosimulation	108
E	Tutorial for Modelsim Co-Simulation	115

# CHAPTER 1

## INTRODUCTION

### 1.1 Background of Study

This thesis proposes a co-simulation platform using the state-of-art System Verilog HDVL via direct programming interface (DPI) technique applied on image processing sub algorithms. This chapter gives an overview of the proposed title and, problem statement, objectives, scope of work, expected contribution and thesis outline.

In the recent years, as the technology grows system-on-chip (SOC) designs become bigger and bigger. Having all the design in hardware makes our design bulky and expensive. On the other hand implementing into the pure software will consequence very slow design since we are not able to use parallel processing as we are in the hardware design. The hardware design is done only when performance is taken into account. This is because hardware design will speed up the whole system or an application.

There is a report shows that functional verification has become the biggest bottleneck as it consumes roughly 70% of the chip development time and efforts [1]. To verify our hardware/software design we can deploy it on FPGA board, but it is

very time consuming and involves a lot of technical complexities. In that case, this design used hardware/software co-simulation and direct programming interface (DPI-C) whereas it allows System Verilog calls C functions and vice versa. The proposed work has overcome the problems faced when running a co-simulation based on Modelsim simulator using direct programming interface (DPI) technique.

Image processing becomes very useful and important these days as the other technologies grows. It is widely used in different area such as military, astronomy, security systems, robotics and etc. The image itself can be declared and defined by its pixel values and these values can be assigned to matrix elements. In the case of color images which are consist of a combination of 3 main colors, i.e. red, green and blue. Each color can separately assign to a particular matrix. In the case of the N by M pixel gray level image it can only defined by one matrix that is N by M. Because of this definition of image, we are dealing with 2 dimensions matrixes.

There are a lot of operations that can operate on 2D matrix or in other word our input image. For instance simple operations like addition, multiplication, subtraction and so on. But when we want to process the real image using some operation such as edge detecting, blurring, sharpening and even logical operation such as dilation and erosion we have to use more advanced operation called “convolution” and “correlation”. In this work convolution and correlation module is implemented in hardware.

## **1.2 Problem Statements**

In this segment at first the problems are stated and then, according to each statement related solution is proposed.

Software implementation of convolution or correlation algorithm is too slow to be real-time. It is because in software we are only restricted to execute one operation

at a same time. Since both convolution and correlation share most part of their algorithm, it can be merged together in one module. Another difficulty is that in the real world applications input image size is not fixed. On the other hand filtering is different from one application to another and kernel size also depends on that specific application.

The whole image processing algorithms in which convolution or correlation is applied should not be implemented in hardware because it will be costly. Besides, when one algorithm is working efficiently on software it is not appropriate to implement it in hardware. The hardware implementation is advisable only when parallel processing is possible so that we will be able to speed up the design. We should find a way for co-designing the hardware/software algorithm. As long as we are using the free version of ModelSim we may face some limitations. The biggest difficulty faced is that software part of algorithm doesn't consume time simulation.

Verification via deploying into FPGA board is very time consuming and involves a lot of technical complexities.

### **1.3 Objectives**

Design and implement a 2D Convolution or 2D Correlation hardware module targeting for FPGA. And parameterizing the whole code to handle any kernel and Image size.

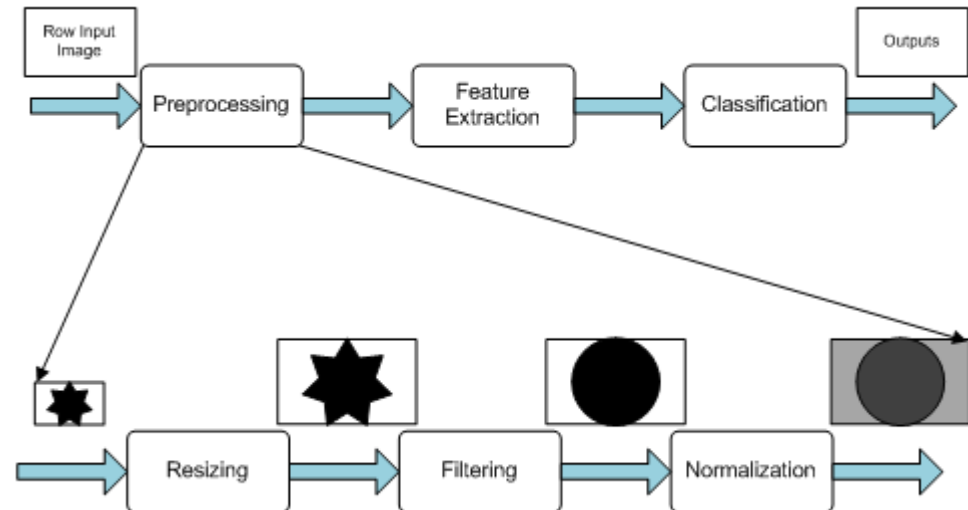
To partitioned into two parts in order to achieve real time performance: software and hardware. Software and hardware parts should be able to communicate with each other through DPI-C. We have to establish a platform to overcome Modelsim limitations in such a way that can be applied to more HW/SW co-simulation cases.

It is good to know whether the hardware/design software works fine or not. But as it is mentioned Verification via deploying into FPGA board is very time consuming and involves a lot of technical complexities. Therefore, to verify our design, we use HW/SW co-simulation using DPI-C.

## 1.4 Scope of the Research

The scopes of the work for this thesis are:

- I. Our test case is “generic image pre-processing algorithm”, which is the part of pattern recognition algorithm and includes 3 parts: resizing, filtering and normalization.



**Figure 2-1** Typical Pattern Recognition Algorithm

- II. HW/SW design is written in System Verilog and C, verification will be possible by employing MODELSIM free simulator and DPI-C.
- III. After getting results, it will be compared with pure software

implementation in MATLAB or Dev-C++.

- IV. In this paper Open source software tools such as Quartus II 13.0 and Altera-ModelSim 10.1d are used to ensure this work can be repeated and extended in the future.

### **1.5 Expected Contribution**

The expected contribution of this work is to establish a platform to be able to simulate hardware-software designs before deploying into the FPGA board. The free version of Modelsim is being used and the proposed methodology should overcome its limitations to perform co-simulation.

## REFERENCES

- [1] "Modi D., Sitapara H., Shah R., Mehul, E., (2011). Integrating MATLAB with Verification HDLs for Functional Verification of Image and VideoProcessing ASIC, 2(2), 258–265.."
- [2] E. SALIBA and A. DIPANDA, "An overview of Pattern Recognition," *wikiprogress.org*, 2013.
- [3] B. Baba, "A HIGH SPEED 2D CONVOLUTION HARDWARE MODULE FOR IMAGE PROCESSING APPLICATIONS IN HARDWARE," UNIVERSITY TECHNOLOGY MALAYSIA, 2013.
- [4] A. H. Heng, "Reconfigurable address generation unit for 2D correlation in FPGA," Master, Universiti Teknologi Malaysia, 2012.
- [5] K. K. HORNG, "FPGA IMPLEMENTATION OF A RECONFIGURABLE ADDRESS GENERATION UNIT FOR IMAGE PROCESSING APPLICATIONS," MASTER, UNIVERSITY TECHNOLOGY MALAYSIA, 2013.
- [6] A. Freitas, *Hardware-Software Co-verification Using the SystemVerilog DPI*: Techn. Univ. Chemnitz, Fakultät für Informatik, 2007.
- [7] <http://www.wikipedia.org/>.
- [8] <http://www.testbench.in>.
- [9] <http://www.doulos.com/knowhow/sysverilog/tutorial/dpi/>.
- [10] N. A. a. F. T. Yarman-Vural, "An Overview of Character Recognition Focused on Off-Line Handwriting," 2001.
- [11] R. E. W. Rafael C. Gonzalez, *Digital Image Processing*, Second Edition ed., 2002.
- [12] "Khalil-Hani, M. Digital Systems: VHDL & Verilog Design. UTMSkudai: Prentice Hall. 2012.."
- [13] N. P. Escudero, "PERFORMANCE ESTIMATION FOR THE DESIGN SPACE EXPLORATION OF SYSTEM-ON-CHIP SOLUTIONS," PHD, 2003.
- [14] J. K. J. KAN, "SOFTWARE AND HARDWARE CO-SIMULATION PLATFORM FOR IMAGE PROCESSING," Master, UTM, 2014.