

**HARDWARE SOFTWARE PARTITIONING USING
PARTICLE SWARM OPTIMIZATION (PSO) IN IMAGE
PROCESSING APPLICATION**

TAN JIA ZHENG

UNIVERSITI SAINS MALAYSAI

2018

**HARDWARE SOFTWARE PARTITIONING USING
PARTICLE SWARM OPTIMIZATION IN IMAGE
PROCESSING APPLICATION**

By

TAN JIA ZHENG

Thesis submitted in fulfilment of the requirements

for the degree of

Bachelor of Engineering (Electronic Engineering)

JUNE 2018

ACKNOWLEDGEMENT

Firstly, I want to thank my final year project and research supervisor Assoc. Prof. Dr. Zaini Abdul Halim for her patient guidance and advices throughout the time of doing my final year project. Her continuous support and precious advices are one of the factors in successful completion of my project and are highly appreciated. I feel proud to be one of the students of my supervisor.

Besides, I would also like to thank the PhD student, Tan Earn Tzeh for his precious advices and support for my final year project. He is always willing to share his experiences and knowledges to me and giving me advices about the final year project and research.

In addition, I want to thank my precious course mates that study together for four years in my university life. They are always willing to spend their time with me, sitting together exchanging knowledges, discussing about the problems faced during the time of doing final year project.

Lastly, I would like to thank my family for giving me the love and support. Without their support, I would not have successfully completed my final year project and thesis.

TABLE OF CONTENTS

Table of Contents

ACKNOWLEDGEMENT	i
TABLE OF CONTENTS	ii
LIST OF TABLES	v
LIST OF FIGURES	vi
LIST OF ABBREVIATION	viii
ABSTRAK	ix
ABSTRACT	x
CHAPTER ONE - INTRODUCTION	1
1.1 Research Background	1
1.2 Problem Statement	4
1.3 Objectives of Research	4
1.4 Scope of Research	5
1.5 Thesis Organization	6
CHAPTER TWO – LITERATURE REVIEW	7
2.1 Overview	7
2.2 Particle Swarm Optimization	7
2.3 Processor (CPU)	8
2.4 Field Programmable Gate Array (FPGA)	8
2.5 Hardware-Software Partitioning Algorithm	9
2.6 Recognition of vehicle’s plate number	11
2.7 Chapter Summary	13
CHAPTER THREE - METHODOLOGY	14
3.1 Overview	14
3.2 Determination of the nodes	15
3.3 Obtaining Data from Altera board	21

3.3.1	Execution time in HPS	21
3.3.2	Measurement of Resources in FPGA	25
3.3.3	Execution Time in FPGA	26
3.4	Formulation of Hardware-Software Partitioning	29
3.5	Partitioning approach using PSO algorithm	30
3.6	Summary	32
CHAPTER FOUR – RESULT AND DISCUSSION		33
4.1	Overview	33
4.2	Accuracy of image processing algorithm developed	33
4.3	Execution time and resources utilization in FPGA	36
4.4	Execution time in HPS	38
4.5	Comparison of execution time between HPS and FPGA	39
4.6	Hardware-Software partitioning using PSO algorithm	40
4.6.4	Test Case 1	40
4.6.4	Test Case 2	43
4.6.3	Test Case 3	44
4.6.4	Test Case 4	46
4.6.5	Test Case 5	47
4.6.6	Test Case 6	48
4.6.7	Summary from all test case	49
4.7	Performance Comparison	50
4.8	Summary	52
CHAPTER FIVE - CONCLUSION		53
5.1	Conclusion	53
5.2	Future Work	54
REFERENCES		55
APPENDICES		57
Appendix A: Results in Visual Studio		57

Appendix A.1: Result for test case 1	57
Appendix A.2: Result for test case 2	57
Appendix A.3: Result for test case 3	58
Appendix A.4: Result for test case 4	59
Appendix A.5: Result for test case 5	59
Appendix B: Coding in MATLAB for PSO algorithm	60
Appendix B.1: Main	60
Appendix B.2: Cost Function	60
Appendix B.3: BPSO algorithm	61
Appendix B.4: DrawConvergenceCurves	62
Appendix C: Coding for FPGA simulation in ModelSim	63
Appendix D: Coding for HPS	90
Appendix E: Coding in Visual Studio 2017	96

LIST OF TABLES

	Page
Table 3.1: Nodes assignment	15
Table 3.2: Conceptual illustration on horizontal profiling	19
Table 3.3: Conceptual illustration on vertical profiling	20
Table 4.1: Image before and after image processing	33
Table 4.2: Resources used in each node in FPGA	36
Table 4.3: Execution time for each node	36
Table 4.4: Time and resources used in FPGA	37
Table 4.5: Execution time for HPS	38
Table 4.6: Average Execution time for each node.....	38
Table 4.7: Execution time for each node in HPS and FPGA.....	39
Table 4.8: Result for test case one	40
Table 4.9: Result of the total resources used	41
Table 4.10: Result for test case two	43
Table 4.11: Result for test case three	44
Table 4.12: Result for test case four	46
Table 4.13: Result for test case five.....	47
Table 4.14: Result for test case six	48
Table 4.15: Result of the partitioning	50
Table 4.16: Comparison between HW/SW partitioning and pure hardware	51
Table 4.17: Comparison between HW/SW partitioning and pure software	51

LIST OF FIGURES

	Page
Figure 2.1: Modification of a searching points by PSO	8
Figure 2.2: Flow chart to extract car plate region.....	12
Figure 3.1: Project Implementation Flow	14
Figure 3.2: Image of vehicle	15
Figure 3.3: Image after crop	16
Figure 3.4: Grayscale image	16
Figure 3.5: Binary image	17
Figure 3.6: Image of removed car plate's characters	17
Figure 3.7: Image after subtraction.....	18
Figure 3.8: Image after removed noise vertically	18
Figure 3.9: Image of removed noise horizontally.....	19
Figure 3.10: Image after cropped in row	20
Figure 3.11: Image of vehicle's plate number	20
Figure 3.12: Flow Chart for Software.....	22
Figure 3.13: Internet IP address of the board.....	22
Figure 3.14: SoC EDS Command Shell.....	23
Figure 3.15: Files in the directory.....	23
Figure 3.16: Command to copy file into SD card.....	24
Figure 3.17: Execute the C file	24
Figure 3.18: Step to compile Verilog code Quartus Software	25
Figure 3.19: Compilation Report	26
Figure 3.20: Simulation in ModelSim	27

Figure 3.21: Add the project file to generate wave form.....	27
Figure 3.22: Run the project	28
Figure 3.23: Number of clock in node 1, C1	28
Figure 3.24: Number of clock in node 2, C2	29
Figure 3.25: PSO algorithm framework	30
Figure 4.1: Graph of execution time against node.....	39
Figure 4.2: Convergence graph for (a) C=1022, (b) C=681, (c) C=341.....	42
Figure 4.3: Convergence graph for C=1022	45
Figure 4.4: Convergence graph for C=681	45
Figure 4.5: Convergence graph for C=341	46
Figure A.1: Result for each node for test case 1	57
Figure A.2: Result for each node for test case 2.....	57
Figure A.3: Result for each node for test case 3.....	58
Figure A.4: Result for each node for test case 4.....	59
Figure A.5: Result for each node for test case 5.....	59

LIST OF ABBREVIATION

ALU	Arithmetic Logic Unit
ARM	Advanced RISC Machines
CPU	Central Processing Unit
FPGA	Field Programmable Gate Array
GA	Genetic Algorithm
GUI	Graphical User Interface
HDL	Hardware Description Language
HGAPSO	Hybrid GA and PSO
HPS	Hard Processor System
HW	Hardware
ILP	Integer Linear Programming
IPSO	Improved Particle Swarm Optimization
PSO	Particle Swarm Optimization
SoC	System on Chip
SW	Software

**PEMBAHAGIAN PERISIAN PERKAKASAN MENGGUNAKAN OPTIMIZASI
PERTIMBANGAN SWARM (PSO) DALAM PEMROSESAN PENGECUAN IMAGE
ABSTRAK**

Dalam projek ini, pengenalan plat kereta akan dilaksanakan dalam pembahagian perisian perkakasan dengan menggunakan algoritma PSO. Rangka kerja untuk pembahagian perisian perkakasan menggunakan algoritma PSO dalam MATLAB dibangunkan. Prestasi antara penyelesaian dalam pembahagian perisian perkakasan dan penyelesaian tanpa pembahagian disiasat. Formula pemprosesan imej disahkan di Visual Studio. Kemudian pengkodan ditulis dalam Verilog untuk perkakasan dan bahasa C untuk perisian untuk mendapatkan masa pelaksanaan dan penggunaan sumber. Kemudian data akan diproses di MATLAB menggunakan algoritma PSO untuk menentukan hasil optimum dalam pembahagian. Parameter algoritma PSO seperti bilangan lelaran dan bilangan zarah diubah untuk mendapatkan nilai optimum bagi parameter. Tiga kekangan yang berbeza, $C=1022$, $C=681$ dan $C=341$ akan diambil kira untuk menghasilkan penyelesaian yang optimum. Penyelesaian untuk $C=1022$ menggunakan 55% daripada jumlah sumber perkakasan (1362) dalam perkakasan tulen. Ia adalah 1.11 kali lebih cepat daripada perkakasan tulen dan 1.45 kali lebih cepat daripada perisian tulen. Penyelesaian untuk $C=681$ menggunakan 49.7% daripada jumlah sumber perkakasan dalam perkakasan tulen dan ia adalah 1.09 kali lebih cepat daripada perkakasan tulen dan 1.42 kali lebih cepat daripada perisian tulen. Penyelesaian untuk $C=341$ menggunakan 22.03% daripada jumlah sumber perkakasan dalam perkakasan murni dan ia adalah 1.05 kali lebih cepat daripada perkakasan murni dan 1.36 kali lebih cepat daripada perisian tulen. Prestasi dalam pembahagian perisian perkakasan adalah lebih tinggi atau lebih baik berbanding perkakasan tulen dan perisian tulen. Pembahagian perisian perkakasan mempunyai kelajuan pemprosesan yang cepat dan kurang menggunakan sumber perkakasan. Kerja masa depan projek ini adalah untuk melaksanakan penyelesaian partition perisian perkakasan di Altera DE1-SoC.

HARDWARE SOFTWARE PARTITIONING USING PARTICLE SWARM OPTIMIZATION (PSO) IN IMAGE PROCESSING

APPLICATION

ABSTRACT

In this project, car plate identification will be implemented in hardware-software partitioning by using PSO algorithm. The framework for hardware-software partitioning using PSO algorithm in MATLAB is developed. The performance between the solution in hardware-software partitioning and the solution without partitioning is investigated. The image processing's formulas are verified in Visual Studio. Then the coding is written in Verilog for hardware and C language for software to obtain the execution time and resources consumption. Then the data will be processed in MATLAB using PSO algorithm to determine the optimal result in partitioning. The PSO algorithm parameters such as the number of iteration and number of particles are varied to obtain the optimum value for the parameters. Three different constraints value, $C=1022$, $C=681$ and $C=341$ are take into consideration to generate an optimum solution. The solution for $C=1022$ use 55% of the total hardware resources (1362) in pure hardware. It is 1.11 times faster than pure hardware and 1.45 times faster than pure software. The solution for $C=681$ use 49.7% of the total hardware resources in pure hardware and it is 1.09 times faster than pure hardware and 1.42 times faster than pure software. The solution for $C=341$ use 22.03% of the total hardware resources in pure hardware and it is 1.05 times faster than pure hardware and 1.36 times faster than pure software. Performance in hardware-software partitioning is higher or better compare to pure hardware and pure software. Hardware-software partitioning has fast in processing speed and use less in hardware resources. Future work of this project is to implement the hardware-software partitioning solution in Altera DE1-SoC.

CHAPTER ONE

INTRODUCTION

1.1 Research Background

Image processing is a method that analyse and processes the digital data inside the digital image. For mathematical analysis, a digital image may be defined as a two dimensional function $f(x,y)$ where x and y are the coordinates in the image and the function $f(x,y)$ is the intensity of the image at the x and y coordinates. In digital image, it has a finite number of elements which has its own particular coordinate and intensity. These elements are called pixels. Pixel is the most widely used term to denote the elements of a digital image. There are various image processing techniques which are image preprocessing, image enhancement, image segmentation, feature extraction and image classification [1].

Image processing can be time and resources consuming task. Image with high resolution will consume a lot of time and resources when the image goes through processing steps. Therefore, there are many methods are proposed in order to reduce time and resource consumption. For example, color processing and contrast enhancement will make the image clearer and can be processed easily [2] [3].

Vehicle's plate recognition is one of the application in image processing [4]. Vehicle's plate recognition system allow user read the plate number faster. By using this system, police are able to check whether vehicle plate is registered or not. In this project, image of the vehicle is captured and will go through image processing steps until it extracts the plate's number [5]. From the original vehicle's image until it extracts the plate's number, it goes through many steps and these steps are the nodes that will be implemented in hardware software partitioning. There are various type of method to extract the plate number [6] such as finding the plate's colour, finding the shape of plate

and finding the area with white characters. These methods are able to extract the car's plate but the processing steps are different to process until extract the car plate.

There are two type of processor which are hard processor and soft processor. For hard processor, it is Hard Processor System (HPS) and the processor is fabricated on silicon and it is fix such as Advanced RISC Machines (ARM) processor. For soft processor, the processor is designed using register through Field Programmable Gate Array (FPGA), and it is flexible such as Nios. In embedded System-On-Chip (SoC) that platform that is incorporating FPGA and HPS are used. Normally the HPS is used for Graphical User Interface (GUI) and FGA is used for processing for calculation such as algorithm. In HPS, application is done through software using C language. In FPGA, application is done through hardware. Normally, Hardware Description Language (HDL) is used such as Verilog or VHDL.

Altera DE1 SoC is one of the embedded system platforms which has HPS (software) and FPGA (hardware). This is beneficial since application specific hardware is usually much faster in processing speed than software, but it is also expensive. On the other hand, software is cheaper and easily to maintain but its performance is slow [7]. It shows a good trade-off between cost and performance can be achieved if both software and hardware are used for embedded system application. In order to obtain effective solution, hardware software partitioning should be used in the application.

Nowadays embedded systems becomes more complicated with more tasks should be executed by the system. Hence hardware-software partitioning is proposed to distribute the task either to implement in hardware or software. Some tasks will be implemented in hardware and some tasks will be implemented in software [8]. There are many algorithms that can be implemented in hardware software partitioning such as Integer Linear

Programming (ILP), Simulated Annealing (SA), Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) [9].

PSO is a numerical search algorithm which used to find optimal solution from constrained requirement. The PSO algorithm was introduced by Kennedy and Eberhart [10]. PSO algorithm was inspired by the social behavior of bird flocking or fish schooling. In the original concept of PSO, particles fly through the search space influenced by two factors [11]: one is the individual's best position ever found; the other is the group's best position. PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithm (GA). The system is initialized with a population of random solutions and searches for optima by updating generations. In PSO, the potential solutions called particles, it hold the best solution and will repeatedly update it to an optimal solution. Compared to GA, the advantages of PSO are easy to implement and there are few parameters to adjust. [12].

In this project, Altera DE1 SoC platform is used. This platform consists of FPGA and ARM Cortex A9 [13]. Vehicle's plate recognition is the image processing application that will be studied using this platform. In image processing, the task is split into smaller tasks that is known as nodes. Each node will be simulated in the platform to perform hardware-software partitioning. Each node with have its own execution time and resources utilization. By using PSO algorithm, near-optimal result will be determined and the result is analyzed [14] .

1.2 Problem Statement

Image processing mostly done in CPU (software). When image processing fully implemented in CPU, the execution time to do the image processing will be longer [15]. However when it is implemented in hardware or FPGA, more resources will be consumed. Comparing between CPU (software) and FPGA (hardware) for image processing application, software used more time compared to hardware and hardware used more resources than software [16] [17]. In order to trade-off between the execution time and resources, therefore hardware software partitioning is the best solution [18].

1.3 Objectives of Research

The main goal of the project is to perform hardware-software partitioning using PSO algorithm for image processing application. To achieve it, several specific objectives have to be fulfilled:

1. To develop a framework for hardware-software partitioning using PSO algorithm for car plate identification using image processing method.
2. To investigate the performance of the hardware-software partitioning in term of processing speed and resources consumption for car plate identification.

1.4 Scope of Research

In this project, PSO algorithm will be implemented in HPS and FPGA of Altera DE1-SoC board. The function of each nodes will be identified in Visual Studio. These functions will be implemented in processor by using C programming language and will be implemented in FPGA by using Verilog Hardware Description Language. Data such as execution time and resources utilization for each task in FPGA and data such as execution time for each task in HPS are collected and tabulated.

With the help of MATLAB, the framework for hardware-software partitioning is developed. There are two parameters are varied which are number of iteration and number of particles. PSO algorithm is used to generate the best solution in hardware-software partitioning. Three different constraint are applied to PSO algorithm which are $C=1022$, $C=681$ and $C=341$ where the C is the maximum hardware resources constraint. Solutions are that generated from PSO algorithm cannot exceed the limit, C and with minimum execution time.

1.5 Thesis Organization

This thesis consists of 5 chapters. Chapter 1 is introduction that discusses about research background, main motivation of doing this research, problem statement, objective of the research and scope of the project.

Chapter 2 is literature review that reviews about the previous related work done on implementation of Particle Swarm Optimization in hardware software partitioning. Some basic knowledge related to the project is also being discussed.

Chapter 3 is methodology. It discusses about the project implementation flow, the function of each nodes in image processing, design flow for hardware and software and implementation of PSO algorithm in hardware and software. Besides that, the performance of each nodes in terms of execution time and resources utilization is also being discussed.

Chapter 4 presents the result and analysis of the data that obtained in Chapter 3. The performance of each nodes in hardware and software in terms of execution time and resources utilization will be presented. Analysis of the results is discussed in this chapter.

Chapter 5 is conclusion that presents the summary of the work and results of the project. Future works of this project is also included.

CHAPTER TWO

LITERATURE REVIEW

2.1 Overview

Recently, there are many research on implementation of PSO algorithm for hardware software partitioning. Various type of method in implementation have been studied in this past few years. In this chapter, some fundamental knowledge that related to the project has been discussed. Besides that, previous work done on implementation of PSO algorithm in hardware software partitioning on embedded system also has been reviewed in this chapter.

2.2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is evolutionary algorithms that based on swarm intelligence. In PSO, each particle is assumed to memorize the best position. The best position that has ever been found by the whole swarm termed global best and by each particles know as personal best. In order to find the global optimum of the optimization problem, particles need to learn from the global best (gbest) and personal best (pbest) [19].

Figure 2.1 shows that the concept of searching the best point by the PSO algorithm. The current search point and the current velocity vectors are called x^k and v^k respectively. The modified values for the particles position and velocity vectors are estimated based on the interaction with other particles and the last best position of the particles itself [20]. In PSO algorithm, particles are update its position based on its current velocity to obtain best position. This process will continues until the optimum result is found.

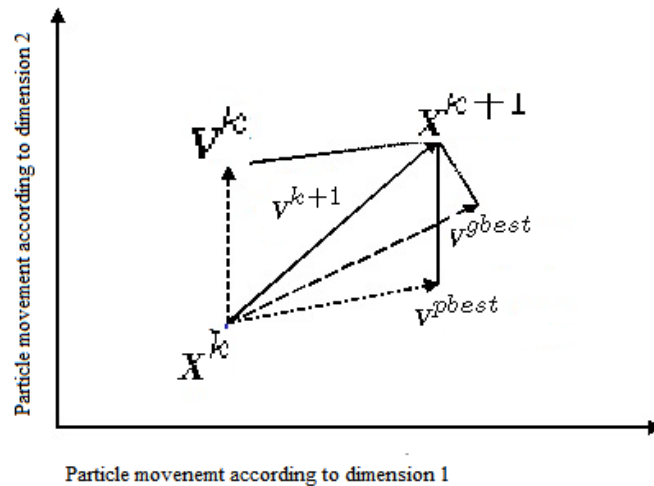


Figure 2.1: Modification of a searching points by PSO

2.3 Processor (CPU)

Processor or Central Processing Unit (CPU) is a component of a computer system. It carries out the instructions of a computer program by performing the logical control, input/output (I/O) operations and basic arithmetic specific instructions. CPU contains arithmetic logic unit (ALU) that can performs arithmetic and logic operations, register that store the result of ALU and control unit that control the fetching and execution of instruction [21].

2.4 Field Programmable Gate Array (FPGA)

FPGA is an integrated circuit designed to be configured by customer or a designer after manufacturing. The FPGA is generally specify using a hardware description language (HDL). FPGA contains an array of programmable logic blocks and allow the block to be configured [22]. One of the important features of the FPGAs is the ability of parallel processing which is different to processor that runs program in sequential manner. Flexibility is another feature of FPGA since the functionality of the FPGA can be changed every time the device is powered up. Change can be made by downloading new configuration file into the device [23].

2.5 Hardware-Software Partitioning Algorithm

M.B.Abdelhalim, A. E. Salama and S. E.-D. Habib [24] had presented the hardware software partitioning using PSO technique. PSO offers a reasonable coverage of the design when it is compared to Genetic Algorithm (GA), which is another evolutionary technique. In the comparison to solve the partitioning problem, the results show that the 8% improvement in the result quality in favour of PSO compared to GA. Another advantage of PSO is its performance in term of speed. They also found out that the PSO is outperforms than GA in term of cost function and the execution times. Besides that, they also tested several type of combination of PSO and GA. The best result obtained from the combination of PSO followed by another PSO. They propose to name this successive PSO algorithm as the Re-excited PSO algorithm. In this research, PSO is a good algorithm in solving hardware software partitioning.

Alakananda Bhattacharya, Amit Konar, et al. [25] presented hardware software partitioning problem in embedded system design using PSO algorithm. They provided an alternative approach to solve this problem using PSO algorithm. They analyzed the proposed scheme with Integer Linear Programming (ILP). Optimal solutions is obtained from the Integer Linear Programming. Computer system and embedded system include both hardware and software components. Usually, application specific hardware is much faster than software and also better in power consumption, but expensive at the same time. For software, it is cheaper but slow and consumes much power when implemented.

Traditionally, hardware software partitioning was accomplished manually. However, when it comes to high complexity in embedded system, researchers prefer automatic approach to solve the problem. They compared the PSO with the GA, ACO and ILP. From the result, ILP works most efficiently with many thousand nodes and yields optimal solution whereas the PSO give nearest optimal solution on an average. In

further observation, PSO based algorithm outperforms GA, ACO and ILP in terms of run time for the given partitioning problem.

Peter Arato and Sandor Juhasz [26] presented about hardware software partitioning in embedded system design. They mentioned that the most important steps in design of embedded system is hardware software partitioning which deciding which components of the system should be implemented in hardware and which ones in software. They have made it possible to investigate the complexity of the problem. They presented two algorithm which are ILP and GA. ILP can handle up to thousands of nodes and yield optimal result. For GA, it gives near optimal solutions on average. Therefore the hardware software partitioning is important in embedded system. It can help to save a lot of execution time and resources utilization used for complex system.

Jia Wei Tang, Yuan Wen Hau and MN Marsono [27] had done a research on hardware software partitioning in embedded system on chip application. For chip application, partitioning is very important in order to get faster execution time and save resources. From the result they obtained, they found that to have an efficient HW/SW partitioning, it has trade off in the throughput and area.

James Kennedy and Russell Eberhart [28] presented in Particle Swarm Optimization. They presented a concept for the optimization of nonlinear functions using PSO method. PSO developed by the authors comprises a very simple concept and can be implemented in a few lines of computer code. It requires mathematical operators and trade off in terms of memory requirements and speed. From the research, PSO is an extremely simple algorithm that able for optimizing a wide range of function.

Nie Ru and Yue Jianhua [11] proposed a hybrid method algorithm combining GA and PSO based hybrid algorithm. It can solve local minimum problem in PSO and has higher efficiency of search. Researchers also did analysis on GA, PSO, improved PSO

(IPSO), hybrid GA and PSO (HGAPSO) in their research. From the simulation results, it is found that the IPSO is more efficient than standard PSO and HGAPSO. It is a simple and effective model to handle different kinds of nonlinear optimization problems.

Soren Ebbesen, Pascal Kiwitz and Lino GuzzeHa [10] presented about Generic Particle Swarm Optimization Matlab function. They mentioned that the PSO is a numerical search algorithm which is used to find parameter that minimize a given objective or fitness function. Matlab is widely used software for numerical computing. PSO is inspired by the coordinated motion of animal living in a groups. The common goal of all groups' members is to find the most favourable location within a specified search space. In this research, researcher proposed this method to solve optimization problems based on the fitness function of the system. Therefore, PSO algorithm is one of the methods to solve hardware software partitioning in MATLAB.

2.6 Recognition of vehicle's plate number

Ragini Bhat and Bijender Mehandia [29] proposed recognition of vehicle's number plate using MATLAB. They clearly mentioned the steps to recognise the plate number from an image of vehicle. The steps are input image, Grayscale image, and binary gradient image with sobel edge detector, dilate image, binary image with filled holes, remove connected object, extraction of number plate area and finally is the image of the number plate of vehicle. This is one of the methods to perform image processing to recognize the vehicle's plate number.

Rajesh Kannan Megalingam, Prasanth krishana, et al [5] proposed a method to extract the car plate region. The steps are shown in Figure 2.2. Researchers have successfully tested the method to extract the plate region for many types of vehicle. Results show that the successful rate for plate extraction is 91%.

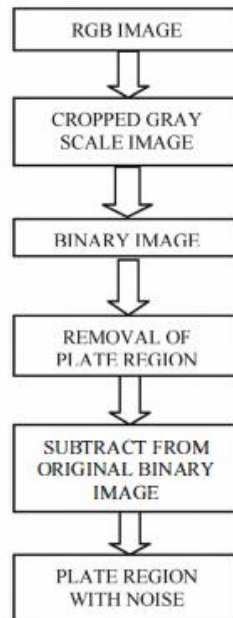


Figure 2.2: Flow chart to extract car plate region

From Figure 2.2, it shows the six steps to extract the car plate region. The RGB image is the image of vehicle. Then the RGB image is cropped by scale image. After that, image is converted to binary image which is in 2 dimensional (D). Forth step is the remove the plate region then subtract the image from the original binary image. Final step is remove the noise in the plate region. Therefore, the extraction of car plate region is successfully complete. This research was done the steps using MATLAB. Image of vehicles with different plate dimension were capture and tested.

2.7 Chapter Summary

In overall, there were many researchers did research on the implementation of algorithm in hardware software partitioning. They made comparison in algorithm for hardware-software partitioning. Besides that, researchers also made enhancement on the algorithm to improve the performance. However, there is still lack of implementation of PSO algorithm in Hardware software partitioning on hybrid platform like Altera De1-SoC that consist of hard processor and FPGA. It would be better to compare the hardware and software performance in hybrid platform like Altera DE1 SoC and perform the combination of the Hardware and software in one system.

CHAPTER THREE

METHODOLOGY

3.1 Overview

In this project, all nodes' function and formulas will be implemented in Visual Studio. The nodes are the steps that must be carried out in image processing, starts from image of a vehicle until the extraction of the vehicle's plate number. After that, the nodes' function is run on FPGA using Verilog and on HPS using C language. Platform that is being used is Altera DE1-SoC which consist both FPGA and HPS. The execution time and resources utilization used of each node is determined. Then, these FPGA's data and HPS's data are analyzed in MATLAB by using PSO algorithm. Finally, all results are compared to each other in term of execution time and resources used. Figure 3.1 shows the overall project implementation flow.

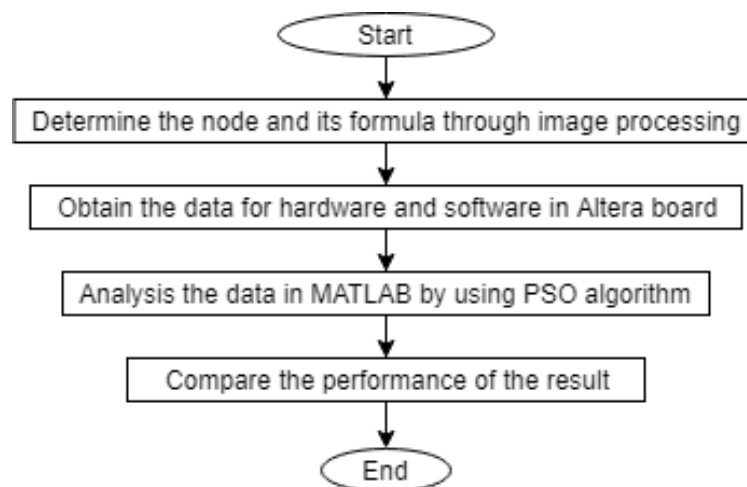


Figure 3.1: Project Implementation Flow

3.2 Determination of the nodes

In this steps, the image of the vehicle will be processed until the image of vehicle's plate number is shown. This process is carried out in the Visual Studio. The nodes are assigned according to the flow of the image processing steps in identifying the car plate. The nodes and its task are tabulated in Table 3.1.

Table 3.1: Nodes assignment

Node	Task
1	Load image
2	Crop image
3	Convert image to Grayscale image
4	Convert Grayscale image to binary image
5	Remove car plate character
6	Subtract between images
7	Remove noise vertically
8	Remove noise horizontally
9	Horizontal profiling and Y-coordinates extraction
10	Crop the image with Y-coordinates
11	Vertical Profiling and X-coordinate extraction
12	Extract the plate

Table 3.1 shows that the twelve steps to process the image of the vehicle. In node 1, the 640*480 RGB image is loaded into Visual Studio (OpenCV). 3-Dimensional array of 640 columns and 480 rows with depth of 3 is being created and loaded, where the first layer is defined as Red Channel, second layer is defined as Green Channel and the third layer of the array is defined as Blue Channel. The Image is shown in figure 3.2.



Figure 3.2: Image of vehicle

In node 2, the image is cropped from the size of 640*480 pixels to 280*360 pixels. It removes unwanted boundary regions by removing first and last 60 rows from the image and first and last 180 columns from the image. The cropped image is shown in figure 3.3



Figure 3.3: Image after crop

Figure 3.3 shows that the image after crop. Any picture will go through this steps in order to fix a size to reduce processing time.

In node 3, the conversion of RGB (Red, Green, Blue) to Grayscale image is processed. In every pixel, 3 channels of colour RGB are converted into single channel of colour is ranged from 0 to 255. The formula for conversion of the RGB to Grayscale is expressed as Equation (3.1).

$$\text{Gray} = (\text{R} + \text{G} + \text{B}) / 3 \quad (3.1)$$

Equation (3.1) is applied for every pixel. A grayscale image can be imagined as 2-Dimensional array. The grayscale image is shown in Figure 3.4.



Figure 3.4: Grayscale image

In node 4, grayscale image is converted to binary image. Every pixel in grayscale image is compared with a threshold value of 150. If pixel value in grayscale image is larger than the threshold value, then the pixel is changed to 255 which represented bit '1' and also represented white colour pixel. Otherwise, the pixel value is changed to 0 which represented it '0' and also represented block colour pixel. The binary image is shown in Figure 3.5.



Figure 3.5: Binary image

In node 5, the car plate's characters is removed. The binary image is processed row by row to find continuous connecting white pixels. If the continuous connecting white pixel is less than 30 pixels, then the connecting white pixels turn into black pixels. Otherwise, the connecting white pixels remain unchanged. The image of removed car plate's characters is shown in Figure 3.6.

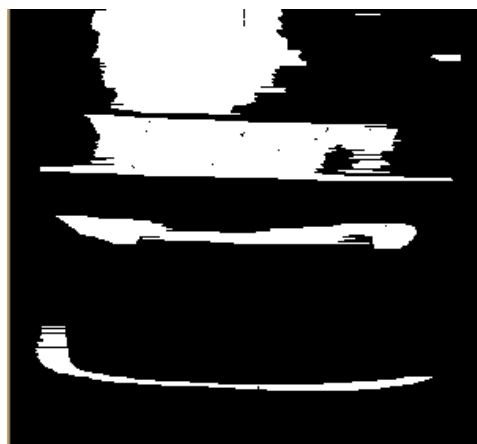


Figure 3.6: Image of removed car plate's characters

In node 6, Subtraction between images is carried out. Subtract the image of removed plate's character (Figure 3.6) from the binary image (Figure 3.5) will give a result as shown in Figure 3.7



Figure 3.7: Image after subtraction

In node 7, noise in the image is removed vertically. The image from figure 3.7 is analyzed column by column to find continuous connecting white pixels. If the connecting white pixels is less than 3 pixels, then it is converted to black pixels. Otherwise, it remain unchanged. After the noise in vertical is removed, the result is shown in Figure 3.8.



Figure 3.8: Image after removed noise vertically

In node 8, noise in the image is removed horizontally. The image from figure 3.78 is analyzed row by row to find continuous connecting white pixels. If the connecting white pixels is less than 3 pixels, then it is converted to black pixels. Otherwise, it remain unchanged. After the noise in horizontal is removed, the result is shown in Figure 3.9.



Figure 3.9: Image of removed noise horizontally

In node 9, image of removed noise is then processed through horizontal profiling and Y-coordinate extraction. Total number of white pixel in a row is calculated. The step is repeated from the top row until end of the row. If the total number of white pixel is larger than 50, the row is labelled as logic '1' else logic '0'. Then, the image is checked from top row to end of the row. When the row logic is changed from 0 to 1, the row number with logic '1' is stored in a variable 'Y1'. When the row logic is changed from 0 to 1, the row number with logic '0' is stored in a variable 'Y2'. The 'Y1' and 'Y2' is the row number for the plate. The concept is illustrated as shown in Table 3.2.

Table 3.2: Conceptual illustration on horizontal profiling

Row number	Number of white pixels	Logic		
1	10	0		
2	8	0		
3	9	0		
4	3	0		
5	65	1	Plate Location	Y1
6	80	1		
7	75	1		
8	55	1		
9	4	0		Y2
10	12	0		

In node 10, the image is cropped according to the Y-coordinate. Image from Figure 3.9 is crop into smaller size from Y1 to Y2. The result is shown in Figure 3.10.



Figure 3.10: Image after cropped in row

In node 11, image in Figure 3.10 is further process in vertical profiling and X-coordinate extraction. Total number of white pixels is calculated in each column from left to right. If the total number of white pixel is larger than 20, the row is labelled as logic '1' else logic '0'. Then, the image is checked from top row to end of the row. When the row logic is changed from 0 to 1, the row number with logic '1' is stored in a variable 'X1'. When the row logic is changed from 0 to 1, the row number with logic '0' is stored in a variable 'X2'. The 'X1' and 'X2' is the row number for the plate. The concept is illustrated as shown in Table 3.3.

Table 3.3: Conceptual illustration on vertical profiling

Column number	1	2	3	4	5	6	7	8	9	10
Number of white pixel	2	7	8	21	30	25	27	40	5	2
Logic	0	0	0	1	1	1	1	1	0	0
Plate Region										
				X1						X2

In node 12, extraction of car plate is carried out. The car plate is extracted from the binary image as shown in Figure 3.5. The image is crop based on the X-coordinate and Y-coordinate which are X1 to X2 and Y1 and Y2. The extracted car plate is shown in Figure 3.11.



Figure 3.11: Image of vehicle's plate number

3.3 Obtaining Data from Altera board

From the previous step, all the steps formulas are verified through image processing in Visual Studio. After that, each node is implemented in both HPS and FPGA. C programming language is used to implement the functions into HPS. For implementation in FPGA, Verilog Hardware Description Language is used. The data such as execution time and resources utilization are collected from both HPS and FPGA.

3.3.1 Execution time in HPS

After the node and its formula are determined through image processing in Visual Studio, the formula is written in C language. To measure the time taken for each node, a library called 'time.h' is utilized. Therefore the time taken for each node can be calculated. In C programming file, the number of clock cycle is taken before and after each node by using `C=clock()` function.

After clock is initialized, timer start counting every 1 micro second. After the program has been complete, the timer will stop counting. The total time required for one node's program is calculated by dividing total clock with clock cycle per second (1000000 clocks per second). The steps is repeated for every node. After the C programming file is written, then the code file is then to HPS. The process can be summarized as shown in figure 3.12.

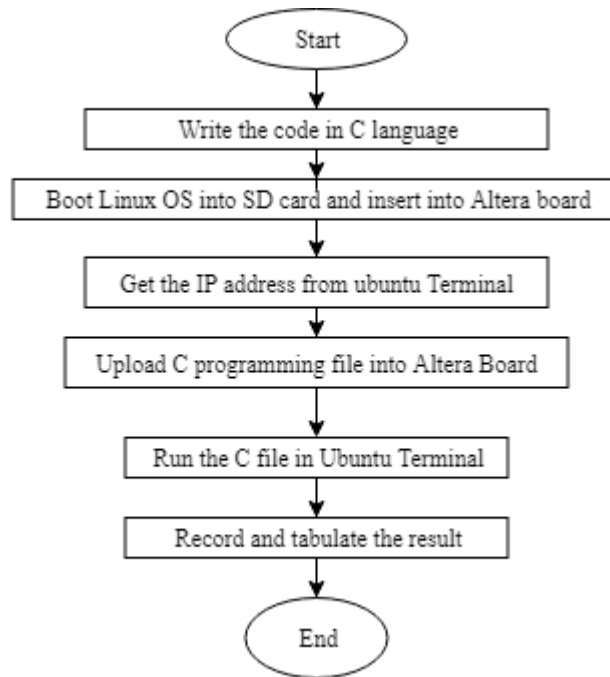


Figure 3.12: Flow Chart for Software

Figure 3.12 shows the flow chart for software part from beginning until data collection. After the code is written in C language, Linux OS is booted into SD card and SD card is inserted into Altera Board.

After that, the IP address of the Altera DE1-Soc is identified by using command ‘ifconfig’ in Ubuntu terminal as show in Figure 3.13. It shows that the internet address is 10.122.23.54.

```

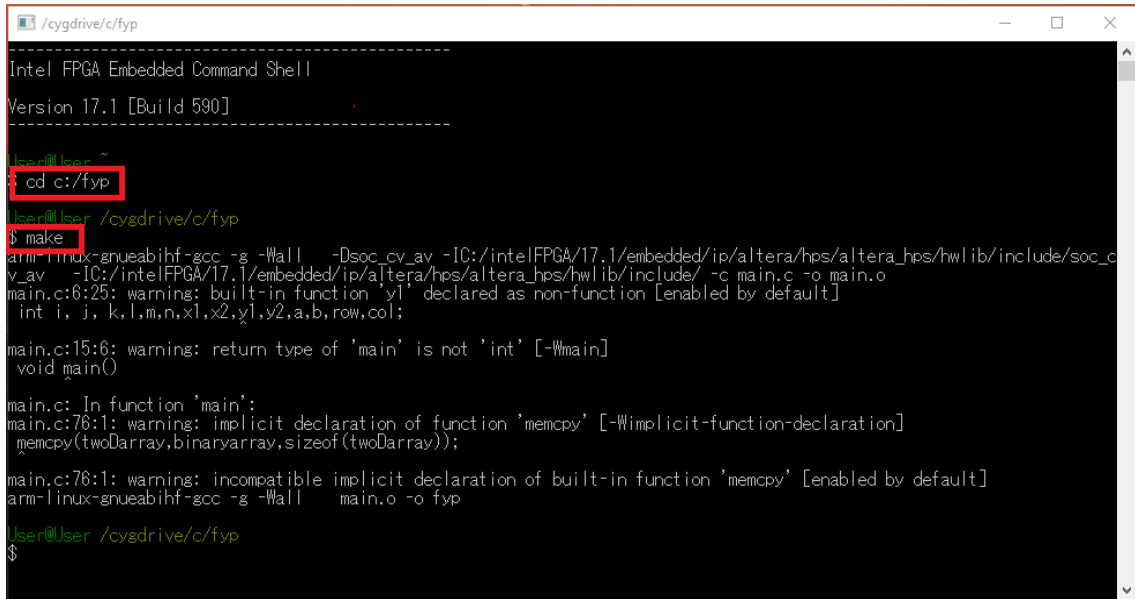
ubuntu@DE1_SoC: ~
File Edit Tabs Help
ubuntu@DE1_SoC:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 3a:43:87:d0:fe:3a
          inet addr:10.122.23.54  Bcast:10.122.23.255  Mask:255.255.252.0
          inet6 addr: fe80::5b0be:ca95:8d75:b509/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:364 errors:0 dropped:0 overruns:0 frame:0
          TX packets:200 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:57530 (57.5 KB)  TX bytes:23201 (23.2 KB)
          Interrupt:27 Base address:0xc000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:73 errors:0 dropped:0 overruns:0 frame:0
          TX packets:73 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:4908 (4.9 KB)  TX bytes:4908 (4.9 KB)

ubuntu@DE1_SoC:~$
  
```

Figure 3.13: Internet IP address of the board

After that, Altera Embedded Command Shell is used to upload the C programming file into the board through some command as shown in Figure 3.14. It shows the command used in the command shell. Firstly, Linux “cd” command is used to change current directory to the project folder name “fyp”. After that, “make” command is used to start building (compiling and linking) process.



```
Intel FPGA Embedded Command Shell
Version 17.1 [Build 590]

User@User ~
$ cd c:/fyp
User@User /cygdrive/c/fyp
$ make
arm-linux-gnueabi-gcc -g -Wall -Dsoc_cv_av -IC:/intelFPGA/17.1/embedded/ip/altera/hps/altera_hps/hwlib/include/soc_c
v_av -IC:/intelFPGA/17.1/embedded/ip/altera/hps/altera_hps/hwlib/include/ -c main.c -o main.o
main.c:6:25: warning: built-in function 'yl' declared as non-function [enabled by default]
    int i, j, k, l, m, n, x1, x2, y1, y2, a, b, row, col;
main.c:15:6: warning: return type of 'main' is not 'int' [-Wmain]
    void main()
main.c: In function 'main':
main.c:76:1: warning: implicit declaration of function 'memcpy' [-Wimplicit-function-declaration]
    memcpy(twoDarray, binaryarray, sizeof(twoDarray));
main.c:76:1: warning: incompatible implicit declaration of built-in function 'memcpy' [enabled by default]
arm-linux-gnueabi-gcc -g -Wall    main.o -o fyp
User@User /cygdrive/c/fyp
$
```

Figure 3.14: SoC EDS Command Shell

After the building process is finished, “ls” command is used to list all the files in the current directory. It shows that the executable file “fyp” is generated successfully as shown in Figure 3.15.

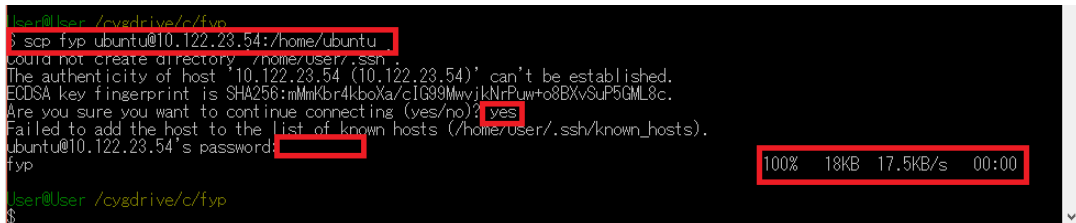


```
User@User /cygdrive/c/fyp
$ ls
fyp main.c .main.o Makefile
```

Figure 3.15: Files in the directory

After the file is successfully generated, the file is then copied into the SD card in Altera board. In Altera Embedded Command Shell, command “scp fyp ubuntu@10.122.23.54:/home/ubuntu” is used to copy the file into the folder “home/Ubuntu”. When prompt message “Are you sure you want to continue connecting (yes/no)”, “yes” is typed and ENTER button is pressed. Next, the password is required.

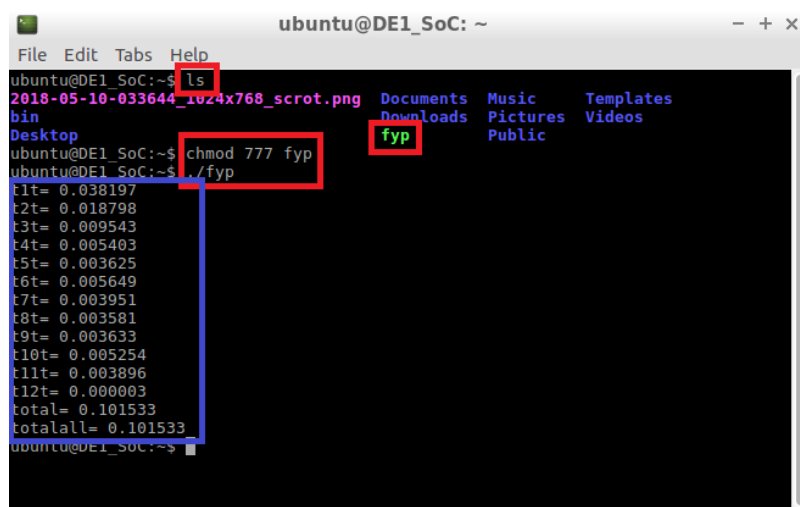
The password “tempPWD” is key in and ENTER button is pressed. After the file successfully copied into the SD card, it shows 100% at the bottom right as shown in Figure 3.16.



```
user@user /cygdrive/c/fyp
> scp fyp ubuntu@10.122.23.54:/home/ubuntu
could not create directory /home/user/.ssh :
The authenticity of host '10.122.23.54 (10.122.23.54)' can't be established.
ECDSA key fingerprint is SHA256:mMnkbr4kboXa/cIG99MvYjkNfPuw+o8BXvSUp5GML8c.
Are you sure you want to continue connecting (yes/no): yes
Failed to add the host to the list of known hosts (/home/user/.ssh/known_hosts).
ubuntu@10.122.23.54's password:
fyp
100% 18KB 17.5KB/s 00:00
user@User /cygdrive/c/fyp
$
```

Figure 3.16: Command to copy file into SD card

After the file is successfully copied into SC card, “ls” command is used in the ubuntu terminal to check whether the file is in the folder as shown in Figure 3.17. After that, command “chmod 777 fyp” is run in order to change the file permission. Finally, the fyp file can be executed by running the command “./fyp”. After the file is executed, the execution time of each node is shown in blue rectangular.



```
ubuntu@DE1_SoC: ~
File Edit Tabs Help
ubuntu@DE1_SoC:~$ ls
2018-05-10-033644_1024x768_scr0t.png  Documents  Music  Templates
bin  Downloads  Pictures  Videos
Desktop  fyp  Public
ubuntu@DE1_SoC:~$ chmod 777 fyp
ubuntu@DE1_SoC:~$ ./fyp
t1t= 0.038197
t2t= 0.018798
t3t= 0.009543
t4t= 0.005403
t5t= 0.003625
t6t= 0.005649
t7t= 0.003951
t8t= 0.003581
t9t= 0.003633
t10t= 0.005254
t11t= 0.003896
t12t= 0.000003
total= 0.101533
totalall= 0.101533
ubuntu@DE1_SoC:~$
```

Figure 3.17: Execute the C file

The result of the execution time in software is recorded and tabulated.