Final Year Project (2023 – 2024)

ELEC/CPEG Proposal Report

Hardware Acceleration of Data Processing

Project ID: **ZW01a-23**

Supervisor: **Professor Wei Zhang**

Author (Student ID): **CHIO Yat Hei (20765204), LUK Pak Him (20766571), SHUM Kwan Ho (20767898)**

Date: **11 September 2023**

Main Objective

This project aims to design and implement an image processing workflow with color to grayscale, edge detection, noise reduction and black-and-white masking algorithms, which is usually applied for pre-processing of image processing algorithms on an FPGA board to enable hardware acceleration of the workflow.

Objective Statements

1. To design and implement an optimized data flow and pipeline stages for median filter, verify the correctness of it.

2. To implement a data cache system, and consolidate it with the implemented sobel filter system.

3. To design and implement an optimized sobel filter for the edge detection part for the whole system, and verify the correctness of it, and integrate into the system.

4. To design and implement Otsu's method that can be updated concurrently and compute final weighting by combining concurrent parts, and integrate into the system.

5. To implement color to grayscale and integrate into the system.

6. To design and implement interface for image input and output, and integrate into the system.

Table of contents

SECTION 1—INTRODUCTION

1.1 Background and Engineering Problem

Nowadays, High performance computing has become an important fast growing branch in the computation industries, as more and more demand of computation came from the widespread growth of Artificial Intelligence (AI), data analytics and mining, computational simulations such as computational fluid dynamics (CFD), etc. While Central Processing Units (CPUs) have been the backbone of computation over the past decades, the upsurge of demand in computational power meant that these general-purpose processors could not simply keep up with the required computational power. Numerous processor architectures are being developed, where various ASICs are being developed targeting specific applications, such as GPUs used to accelerate 3D-graphics rendering, as well as dedicated proprietary circuits for modems. However, ASICs are expensive to develop, which could only be justified by ordering and producing a large amount of chips.[1]

In order to provide a middle point between proprietary ASIC and general-purpose CPUs, Field Programmable Gate Array(FPGA) is being developed by Xilinx to provide programmable IC and circuits to suit the need for low-volume customized ICs. FPGA chips are based on matrices of programmable computational-logic-blocks (CLBs) placed across the chip, which are being connected by customizable interconnects. Currently, FPGA is being used in various applications, such as automobile ECUs, Data Centre switches, and even plane and rocket guidance and control systems. With the increasing demand in high-performance computing and time-critical high-speed computation, FPGA has seen a rise in popularity. [2]

In this project, we are going to use FPGA to its advantage, and apply it for the use of acceleration in image processing algorithms. FPGA provides a platform for dedicated circuit designs and allows concurrent computation and operation with its various CLBs, which also enables a flexible placement of logic blocks and circuits, making it suitable for a wide variety of applications. Image processing methods such as edge detection and noise reduction are highly parallelizable, and being able to be split into pipelining stages, which is why we chose to implement these methods with optimized and efficient algorithms on FPGA boards. Our project is aiming to implement an image processing workflow from a colored image to a noise-reduced black-and-white image on an FPGA board, which is quite a common pre-processing technique used for machine learning and image recognition applications.

We believe that using FPGA for image processing algorithms definitely hints new possibilities for future computational applications, on server-grade processing, personal computation and embedded systems. Currently, programmable logic device manufacturers such as Altera and Xilinx, are producing SOCs that combine CPU and other functional blocks with programmable circuits, which could possibly enhance performance of heterogeneous computing by configuring the logic and circuit according to the computational need. It also provides a low-power solution for parallel processing in embedded applications, such as IOT devices that require a high degree of image processing.

1.2 Objectives

This project aims to design and implement an image processing workflow with color to grayscale, edge detection, noise reduction and black-and-white masking algorithms, which is usually applied for pre-processing of image processing algorithms on an FPGA board to enable hardware acceleration of the workflow.

1.2.1 Objective Statements

Objective 1: To design and implement an optimized data flow and pipeline stages for median filter, verify the correctness of it.

Objective 2: To implement a data cache system, and consolidate it with the implemented sobel filter system.

Objective 3: To design and implement an optimized sobel filter for the edge detection part for the whole system, and verify the correctness of it, and integrate into the system.

Objective 4: To design and implement Otsu's method that can be updated concurrently and compute final weighting by combining concurrent parts, and integrate into the system.

Objective 5: To implement color to grayscale and integrate into the system.

Objective 6: To design and implement interface for image input and output, and integrate into the system.

1.3 Literature Review of Existing Solutions

Compute Unified Device Architecture (CUDA)

In 2006, Nvidia created CUDA, a parallel computing platform and application programming interface (API) specifically designed for utilizing graphical processing units (GPUs) for general computing purposes. CUDA functions as a software layer that provides direct access to the virtual instruction set and parallel computational elements of the GPU, enabling the execution of compute kernels efficiently with direct control over the GPU's capabilities. To maintain a low learning curve for programmers who are familiar with programming languages like C, the CUDA parallel programming model consists of three key components — a hierarchy of thread groups, shared memories, and barrier synchronization, which are designed as minimal language extensions.

However, there are certain restrictions associated with CUDA. Firstly, As CUDA is developed by Nvidia, only Nvidia GPUs are CUDA-enabled.[3] Even though there are some projects attempting to implement CUDA on non-Nvidia GPUs by converting it to OpenCL, it remains unclear whether these implementations can deliver optimal performance on non-Nvidia GPUs since those projects often focus on specific versions of CUDA.

Secondly, all CUDA source code is now processed according to C++ syntax rules, which narrows the choice of programming languages and potentially limits the performance of hardware acceleration on CUDA-enabled GPUs.

Advanced Vector Extensions (AVX)

Nowadays, most CPUs in mainstream personal computers are using the x86 architecture and its derivatives, where the architecture also gains significant market share in the server and HPC market. While GPUs and other ASIC processors, which are purposely designed for concurrent and vector operations, have been increasingly popular, many computational systems are still solely based on CPUs. With the increasing popularity of vector calculations, those that commonly appear in applications such as machine learning and image processing, Intel developed the Advanced Vector Extensions instruction set, so that x86-based CPUs could also handle large vector operations of up to 512-bits, using a single instruction (SIMD).[5] This not only reduces the amount of instructions, thus saving time for instruction fetching, but also boosts performance due to the presence of dedicated hardware. Users could now achieve vector operations on a CPU without a need of a dedicated GPU or ASIC to handle these operations, reducing the complexity of such computation systems.

While AVX provides a great solution by allowing users to run vector operations on a CPU, there are also some limitations and drawbacks. One disadvantage is that AVX instructions, as most of the vector processors, are only able to accelerate vector operations, which form a part of the HPC applications. Applications that are less vectorized and more time-critical such as communications and system control could not utilize the parallelization of AVX architecture, where more complex operations are undergoing concurrently. It is also less flexible, compared to proprietary hardware on ASIC chips or customizable FPGA chips.

SECTION 2—METHODOLOGY

2.1 Overview

2.1.1 System Description

The proposed system aims to provide an accelerated method on doing image processing by using FPGA designs. The image processing methods that are going to be implemented include color to grayscale, edge detection, noise reduction and grayscale thresholding. These image processing methods are quite important in current real life examples, including the preprocessing of images which would be used for image recognition and machine learning.

In a basic image preprocessing workflow, a colored image is first being turned into grayscale, as in many cases the color of the object being recognized does not matter. The resulting image would be transformed from a 3D matrix to a 2D matrix, allowing easier operations afterwards. The color to grayscale is usually done by taking the average of the three colors (RGB) with different weights. Then the image is being passed to an edge detection filter, by detecting the discontinuities of an image and identifying the edges, which help extract the overall shape of an object. The processing is usually done by convoluting a filter onto the image, where there are various filters, including Canny filter, Sobel filter, Prewitt filter etc. As the image after edge detection might contain unwanted noise signals due to the nature of the filters, a noise-reduction step is to be carried out, by methods of spatial filtering such as a median filter. A grayscale threshold is applied so that the image contains only the black and white outlines of the image. The threshold can be determined by a fixed threshold, or by global methods such as the Otsu's method.

The whole system is being divided into a few functional blocks: the input memory block, the logic blocks for processing, as well as the output memory block. For some of the operations, there is also intermediate memory for temporary storing. The Input memory block stores the original image for processing. The processing procedure would possibly be in a pipelined style, where the image is being processed from one step to another. Figure 1 shows the data flow diagram of the completed system.

For color to grayscale functional block, as it is a point operation, each pixel is calculated separately and stored in the intermediate cache memory. To speed up the processing time, the processing is also being split up into multiple stages, where the weighted value of the RGB colors are calculated separately, then being added up. If possible, multiple pixels can be processed concurrently, subjected to the number of functional blocks available.

After passing the grayscale functional block, the processed pixel is being stored to an intermediate memory block, waiting to be fetched to the edge detection block. For the edge detection block, we opt for using the sobel filter, as it is easier to be implemented on an FPGA board. Currently, we are still deciding the optimization methods being used on our system, primarily due to the fact that each FPGA board has different characteristics, where the speed of processing in logic cells and the memory fetching speed varies. There are some optimization methods, such as saving intermediate calculation results back to the memory and reducing the number of arithmetics, suitable only for boards with fast memory access speed. Pipelining could also be implemented, as the final pixel of a convolution is calculated by summing up all the weighted neighboring pixels, benefiting from the characteristic of FPGA. The resulting image is being stored to an intermediate cache, ready for noise reduction.

For the noise reduction block, we opt to use the median filter. As it is a local operation by convolution, a similar approach as the sobel filter can be taken. Traditional single-threaded processor is most efficient by doing a merge sort, where the median is taken easily from the sorted sequence. However, with the case of implementing on an FPGA board, a better approach can be taken, by splitting the 9-pixel local area into 3 groups, computing their own medians respectively and taking the median of those three local medians to find the desired value. This approach not only speeds up the process by utilizing the concurrent processing ability of FPGA circuits, but also by possibly reducing the number of operations if the previous result is being stored and reused for the next pixel. Figure 2 shows the algorithm of the concurrent median filter processing algorithm as suggested in the paper by Bevara and Sanki.[5]

After the whole image is being noise reduced and stored in the memory block, a global operation of Otsu's method is being implemented to determine the threshold of the image, in order to turn it from a grayscale image to black and white only, thus revealing the edges. The resultant image is stored and outputted via an I/O module provided on the FPGA board.

The project will be designed through Vivado, an IDE specifically designed for writing Verilog code for Xilinx FPGA product, where software simulation is also done through the IDE. The verification of functionality is being done with the Simulation Behavioral Model, where test data could be coded to run through the simulated circuit and see if the results correspond to the target output. The Post-implementation Simulation tool also allows resource usage and timing measurements being made, thus allowing the evaluation of different design approaches.

During the project, different design approaches such as those with different optimization and pipelining methods, as well as different emphasis on resource allocation for memory block or processing unit, are being evaluated. They will be implemented on the same FPGA board to provide a level playing ground for resource limitation, while the timing simulation measured would reflect the performance of each design, with the least time spent the better.

Real-life tests of the performance and efficiency will also be made after all the design is being completed and verified, which focuses on measuring both the time of computation of a batch of test data to determine the performance of the implemented design, as well as measuring the power drawn to determine the efficiency of the design. If possible, the performance is being compared against normal CPU operations with both single-thread/multi-thread performance, as well as against other vector-machine architectures.

2.1.2 System Block Diagram

Figure 1: System Diagram of the completed image processing system

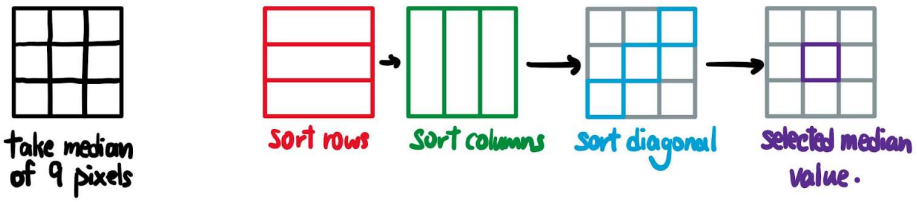

Figure 2: Median Filter Algorithm with concurrency suggested by Bevara and Sanki

2.1.3 Components List

Table 1. List of Specification

Item*	Specifications/Model
Artrex ultrascale	

2.1.4 ECE Knowledge

Knowledge from various ECE courses could be applied to the project, especially those about circuit and logic designs.

ELEC2350, as an introductory course for computer organization and architecture, outlines how modern processors are being designed, in order to maximize performance and efficiency. Nowadays, most processors are being designed with pipelined stages, where a process is being broken down into smaller stages and functional blocks so that the hardware can be used more efficiently with less idle time, and thus improving performance and efficiency. In this project, each functional block or process is also being broken up into pipeline stages, which improves processing performance for the high performance computing applications.

In courses ELEC3310 and ELEC4320, the knowledge of FPGA and Verilog is being introduced in the course, which are the tools being used to implement the design of our project on. ELEC3310 also focuses on digital-circuit and logics, including combinational and sequential logic gates, as well as finite state machines, counters and registers. While the course ELEC4320 focuses on the design principles and considerations of an FPGA system, including hardware placement, resources available and memory management, which aids on the design and consideration of the implementation of our system.

The course ELEC4130 introduces the methods of digital image processing, how digital images are being formed by pixels, and the view of them in both spatial and frequency domain. The course introduces how and why filters are being designed in the way they are, and how to perform convolution on images with kernels. The sobel and median filter introduced in the course serve as the algorithm applied in the project for the edge detection and noise reduction parts of the whole system.

2.2. Objective Statement Execution

2.2.1 Median filter for noise reduction

To design and implement an optimized data flow and pipeline stages for median filter, verify the correctness of it.

Tasks:

Design of median filter

Issac

Implement the computational block of the median filter algorithm

Haven

Implement the median filter algorithm by using the computational blocks

Issac

Verify the correctness of the median filter

Issac

Measure the performance of the implemented median filter

Haven

2.2.2 Data cache for fetching and storing

To implement a data cache system, and consolidate it with the implemented sobel filter system.

Tasks:

Implement the data cache

Gordon

Verify the correctness of data cache

Gordon

Combine data cache with the sobel filter

Issac

2.2.3 Sobel filter for edge detection

To design and implement an optimized sobel filter for the edge detection part for the whole system, and verify the correctness of it, and integrate into the system.

Tasks:

Design of optimized sobel filter

Issac

Implementation of sobel filter

Issac

Verify the correctness of sobel filter

Gordon

Measure the performance of the implemented sobel filter
Gordon
Combine the sobel filter with the rest of the system
Issac

2.2.4 Otsu's method for grayscale to black-and-white

To design and implement Otsu's method that can be updated concurrently and compute final weighting by combining concurrent parts, and integrate into the system.
Tasks:

Design the algorithm for concurrent pipelining of Otsu's algorithm
Haven
Implement the parallelized Otsu's algorithm
Haven
Verify the correctness of the implemented Otsu's algorithm
Gordon
Measure the performance of the implemented Otsu's algorithm
Gordon
Combine the Otsu's algorithm with the rest of the system
Issac

2.2.5 Color to grayscale

To implement color-to-grayscale and integrate into the system.
Tasks:

Implement color to grayscale filter
Issac
Measure the performance of the implemented color-to-grayscale filter
Gordon
Combine the color to grayscale filter with the rest of the system
Haven

2.2.6 Input and Output

To design and implement interface for image input and output, and integrate into the system.
Tasks:

Implement I/O for the system
Haven
Verify correctness of the whole system
Issac
Measure the performance of the whole system
Gordon

SECTION 3— Project Planning

3.1 Project Schedule

The project Schedule is included in the next pages.

Table 2. Project Schedule

Objective Statements	Task	Group Member in charge	wk1-3	wk4-6	wk7-9	wk10-12	wk13-15	wk16-18	wk19-21	wk22-24	wk25-27	wk28-30
Median filter for noise reduction												
	Design of median filter	**Issac**										
	Implement the computational block of the median filter algorithm	**Haven**										
	Implement the median filter algorithm by using the computational blocks	**Issac**										
	Verify the correctness of the median filter	**Issac**										
	Measure the performance of the implemented median filter	**Haven**										

Objective Statements	Task	Group Member in charge	wk1-3	wk4-6	wk7-9	wk10-12	wk13-15	wk16-18	wk19-21	wk22-24	wk25-27	wk28-30
Data cache for fetching and storing												
	Implement the data cache	**Gordon**										
	Verify the correctness of data cache	**Gordon**										
	Combine data cache with the sobel filter	**Issac**										
Sobel filter for edge detection												
	Design of optimized sobel filter	**Issac**										
	Implementation of sobel filter	**Issac**										
	Verify the correctness of sobel filter	**Gordon**										

Objective Statements	Task	Group Member in charge	wk1-3	wk4-6	wk7-9	wk10-12	wk13-15	wk16-18	wk19-21	wk22-24	wk25-27	wk28-30
	Measure the performance of the implemented sobel filter	Gordon										
	Combine the sobel filter with the rest of the system	Issac										
Otsu's method for grayscale to black-and-white												
	Design of the algorithm for concurrent pipelining of Otsu's algorithm	Haven										
	Implement the parallelized Otsu's algorithm	Haven										
	Verify the correctness of the implemented Otsu's algorithm	Gordon										
	Measure the performance of the implemented Otsu's algorithm	Gordon										

Objective Statements	Task	Group Member in charge	wk1-3	wk4-6	wk7-9	wk10-12	wk13-15	wk16-18	wk19-21	wk22-24	wk25-27	wk28-30
	Combine the Otsu's algorithm with the rest of the system	Issac										
Color to grayscale												
	Implement color to grayscale filter	Issac										
	Measure the performance of the implemented color-to-grayscale filter	Gordon										
	Combine the color to grayscale filter with the rest of the system	Haven										
Input and Output												
	Implement I/O for the system	Haven										
	Verify correctness of the whole system	Gordon										

Objective Statements	Task	Group Member in charge	wk1-3	wk4-6	wk7-9	wk10-12	wk13-15	wk16-18	wk19-21	wk22-24	wk25-27	wk28-30
	Measure the performance of the whole system	**Issac**										

3.2 Budget

Items*	Cost
Total	$0

REFERENCES

[1] M. Rouse, Ed. (8 Dec. 2016), “Application-specific integrated circuit,” Techopedia, https://www.techopedia.com/definition/2357/application-specific-integrated-circuit-asic (accessed Aug. 31, 2023).

[2] “What is an FPGA? Field Programmable Gate Array,” Xilinx, https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html (accessed Aug. 31, 2023).

[3] “CUDA GPUs - compute capability,” NVIDIA Developer, https://developer.nvidia.com/cuda-gpus (accessed Aug. 31, 2023).

[4] “High-Performance Computing(HPC) Intel Advanced Vector Extensions 512 (Intel AVX-512) – Technology Brief,” Intel, https://www.intel.com/content/www/us/en/architecture-and-technology/avx-512-solution-brief.html (accessed Aug. 31, 2023).

[5] V. Bevara and P. K. Sanki (4 Jan. 2020), “A new fast and efficient 2-D median filter architecture,” Department of ECE, SRM University-Andhra Pradesh, Guntur, India [Online]. Available: https://www.ias.ac.in/article/fulltext/sadh/045/0192.

APPENDICES

Appendix A — Meeting Minutes

Meeting 1:

Date: 22/07/2023

Time: 3pm

Location: Mong Kok

Attendees: Haven, Issac, Gordon

Minutes taken by: Gordon

- All members decided the topic of this final year project to be design and implement a system for image processing on a FPGA board
- All members discussed the details of the proposal and decided to finish the introduction and methodology by September 1st
- Haven have sent a email to the supervisor, Professor Zhang, to have a meeting discussing about the technical detail of the system and the proposal

Table 1. Action Items for Next Meeting

Action Item to be completed	By when	By whom
Proposal 1.1	Aug 15	Haven
Proposal 1.3.1	Aug 15	Gordon
Proposal 1.3.2	Aug 15	Issac
Proposal 2.1	Aug 15	Issac

Next Meeting: 23/08/2023/3pm/HKUST

Meeting 2:

Date: 23/08/2023

Time: 3pm

Location: HKUST, Room 2449

Attendees: Prof. Zhang, Haven, Issac, Gordon

Minutes taken by: Gordon

- Section 1 of the proposal has been mostly finished.
- Section 2 of the proposal is undergoing a good progress
- Prof. Zhang has suggested that the proposal should mainly focus on the methodology and it should be written with details and diagrams to explain the system clearly.
- Issac has explained the theory of the system to Prof. Zhang and she gave some ideas for the system, including what hardwares or algorithm could be implemented.
- Haven has some enquiries regarding the proposal and suggested arranging a regular meeting with Prof. Zhang, in which a bi-weekly regular meeting is agreed among the group.

Table 1. Action Items from Previous Meeting

Action Item to be completed	**By when**	**By whom**	**Status**
Proposal 1.1	Aug 15	Haven	80%
Proposal 1.3.1	Aug 15	Gordon	Completed
Proposal 1.3.2	Aug 15	Issac	Completed
Proposal 2.1	Aug 15	Issac	50%

Table 2. Action Items for Next Meeting

Action Item to be completed	**By when**	**By whom**
Proposal 1.2	Sept 1	Issac
Proposal 2.1.3	Sept 1	Issac
Proposal 2.1.4.1	Sept 1	Issac
Proposal 2.1.4.2	Sept 1	Haven
Proposal 2.2	Sept 1	Issac
Proposal 3.1	Sept 1	Gordon

Next meeting: 04/09/2023/Time/HKUST

Meeting 3:

Date: 04/09/2023

Time: 11am

Location: HKUST, LSK room1032

Attendees: Prof. Zhang, Haven, Issac, Gordon

Minutes taken by: Gordon

- First Draft of proposal being reviewed by Professor Zhang, where she is satisfied with the most part of the proposal.
- Prof. Zhang suggests minor changes to the proposal, particularly the background of introduction and the project schedule.
- All group members agrees to amend the proposal ASAP and submit it to FYPMS a few days before the official deadline.

Action Item to be completed	By when	By whom	Status
Proposal 1.2	Sept 1	Issac	Completed
Proposal 2.1.3	Sept 1	Issac	Completed
Proposal 2.1.4.1	Sept 1	Issac	Completed
Proposal 2.1.4.2	Sept 1	Haven	Completed
Proposal 2.2	Sept 1	Issac	Completed
Proposal 3.1	Sept 1	Gordon	Completed

Action Item to be completed	By when	By whom
Proposal 1.1	Sept 9	Issac
Proposal 3.1	Sept 9	Gordon