IMAGE CAPTURING AND ANALYSING SYSTEM USING FPGA

EKANAYAKE E.M.M.U.B. (E/17/083)

WEERASINGHE W.A.C.J. (E/19/423)

GROUP 23

INTRODUCTION

Image Processing is a multidisciplinary field that involves the analyzing and manipulation of digital images using computer algorithms. Image processing is used in many fields like medicine, agriculture, archeology, motor traffic control, aerospace investigations, scientific researches etc. Hardware devices like CPU (Central Processing Unit), GPU (Graphical Processing Unit), FPGA (Field Programmable Gate Array), ASIC (Application Specific Integrated Circuit) are vastly used for Image Processing. Among them, FPGA is very popular in the industries since it comes with features like customizability, real time processing, parallel processing, energy efficiency, real time processing etc.
In this project we are applying 2 simple image processing algorithms, Averaging filter and Median Filter.

TECHNOLOGY STACK

We use following hardware components in our project. Budget is also included here.

Component	Specification	Unit Price (Rs.)	Quantity	Total Price (Rs.)
FPGA Board	Altera Terasic DE2-115	252,520.00	1	252,520.00
TTL Converter	CH340/USB to RS232	250.00	1	250.00
Jumper Wires	Generic	20.00	2	20.00
		Grand Total		252,770.00

Table 1

We use following languages and software tools in our project

Language / Software	Description	Version / Standard
SystemVerilog	Hardware Description Language	IEEE 1800-2017
Verilog	Hardware Description Language	IEEE 1364-2005
Python	Scripting Language	3.12.0
Open CV	Python library for image processing	4.8.0
Numpy	Python library for numerical calculations	1.21.0
PySerial	Python library for UART Serial communication	3.5
Math	Python library for mathematical calculation	3.5
Python IDLE	Programming and execution tool for python	3.12.0
Quartus Prime Lite	Synthesizing RTL and programing FPGA	20.1.1
ModelSim	Verification of RTL	20.1.1

Table 2

Specific reasons behind the selection of technology stack

- We use **Altera Terasic DE2-115** since it was the only FPGA board type we can find in our faculty. It comes with a Cyclone IV FPGA module. It meets more than our requirements like

- 3,888 kb embedded memory
- 4 general purpose PLLs
- 40 GPIO pins (3.3V)
- 50 kHz oscillation clock

- We use a **TTL converter** to convert USB data to UART serial data. **Jumper wires** are used to make connection between TTL converter and FPGA (RX and TX)
- **SystemVerilog** is used since it supports packed arrays in RTL, Classes in verification purposes. Therefore, it is a good supportive HDL.
- **Verilog** is used when wrapping SystemVerilog modules to avoid multidimensional torques.
- **Python** is simple and easy for scripting purposes and supports libraries such as **OpenCV, Numpy, PySerial, Math** which are used in our project.

PROCEDURE

1. First we design an initial block diagram for the circuit. This includes filter module, UART to AXIS receiver and AXIS to UART transmitter modules. An Skid buffer is also added in case of a potential backpressure of data from UART to AXIS transmitter module towards filter module. (AXIS = AXI Stream Protocol)

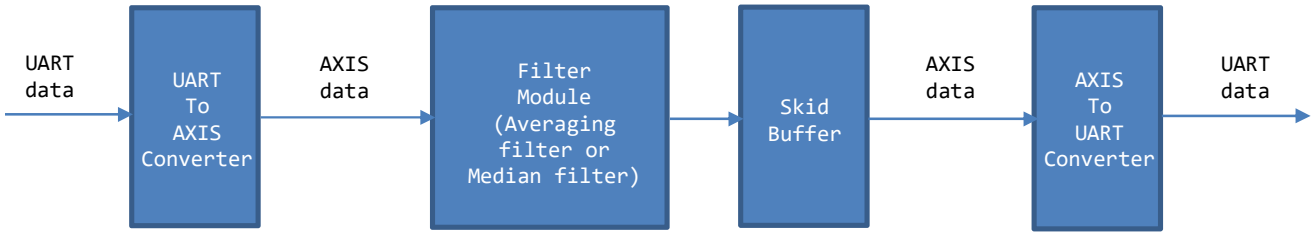

Figure 1

2. Then we design algorithms for the averaging and median filters. For averaging filter, we have to add numbers in the filter kernel. For that we choose *addition tree* method. For median filter, we have to sort elements in the filter kernel, for that we use *bitonic sort* method. Both methods highly support parallel processing, which is a key element in image processing.

3. Then we design state machine for skid buffer, UART to AXIS module and AXIS to UART module.

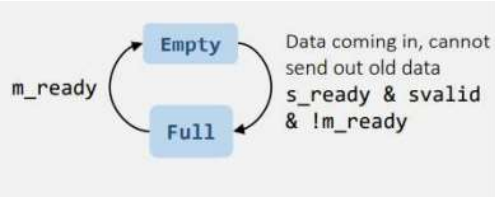

Figure 2: State Machine of Skid buffer

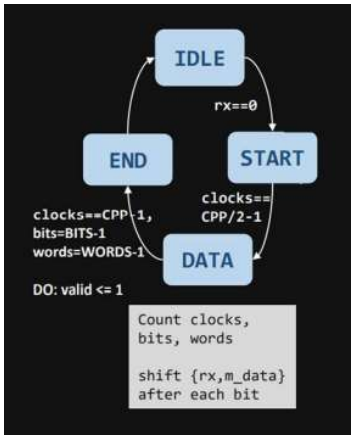

Figure 3: State Machine of UART to AXIS Converter

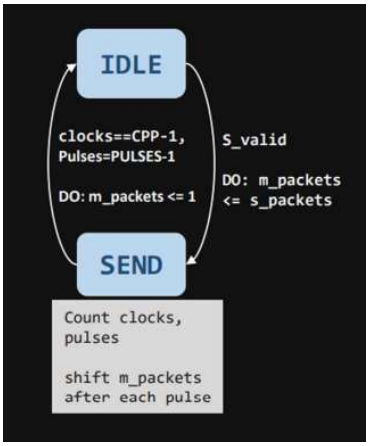

Figure 4: State Machine of AXIS to UART Converter

4. Then we write RTL using SystemVerilog for individual modules with testbenches. We simulate the RTL using test benches.

5. Finally we wrap up the all modules using RTLs written in Verilog and simulate them using testbenches. Following are the wave graphs obtained from the 2 filters.

Figure 5: Wave form of the Median filter. Here img_data is the randomly generated input data and final_img_exp is the calculated output data which is expected. rx_data is the output data from filter and as seen in 2 figures, rx_data and final_img_exp are equal, showing the success of the implementation of the filter.

Figure 6: Wave form of Averaging filter gives successful results

6. Open Quartus Prime Lite, Create a new project using project wizard, and import created RTLs and compile them. Here family as Cyclone IV E and device as EP4CE115F29C7N

7. Search for PLL(Phase Back Loop) in IP catalog and and create a pll with input of 50MHz and output of 1000MHz and generate the PLL Verilog file. Once generated, import the generated PLL module to fpga_module.sv module and input the clock signal generated from PLL to the designed filter module system

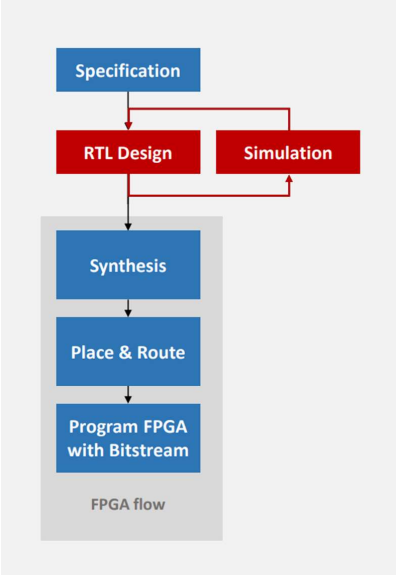

Figure 7: Flow Chart for Programing an FPGA

8. Assign the 2 GPIO pins to rx and tx ports of the generated filter using “Pin Planner” and upload the firmware to the FPGA board.

9. Connect the TTL converter to the FPGA board using 2 jumper wires as following diagram.

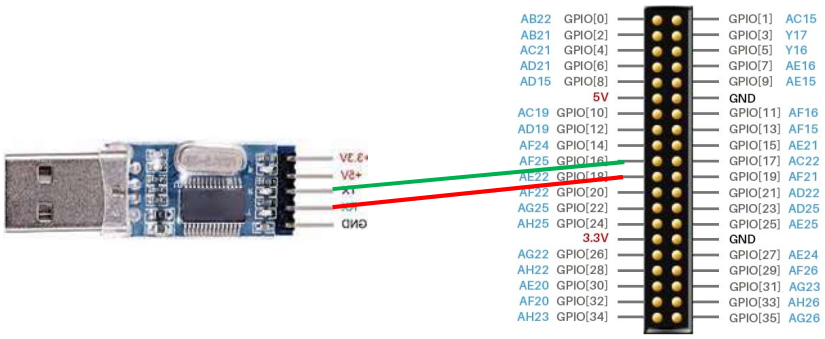

Figure 8

10. Design an algorithm to read image data, send it to the FPGA, receive back the output from FPGA and convert it back to an image (Figure 7)

11. Write python script to the algorithm and implement it.

12. Connect the TTL Converter to USB, power up the FPGA and run the python program and observe the output image

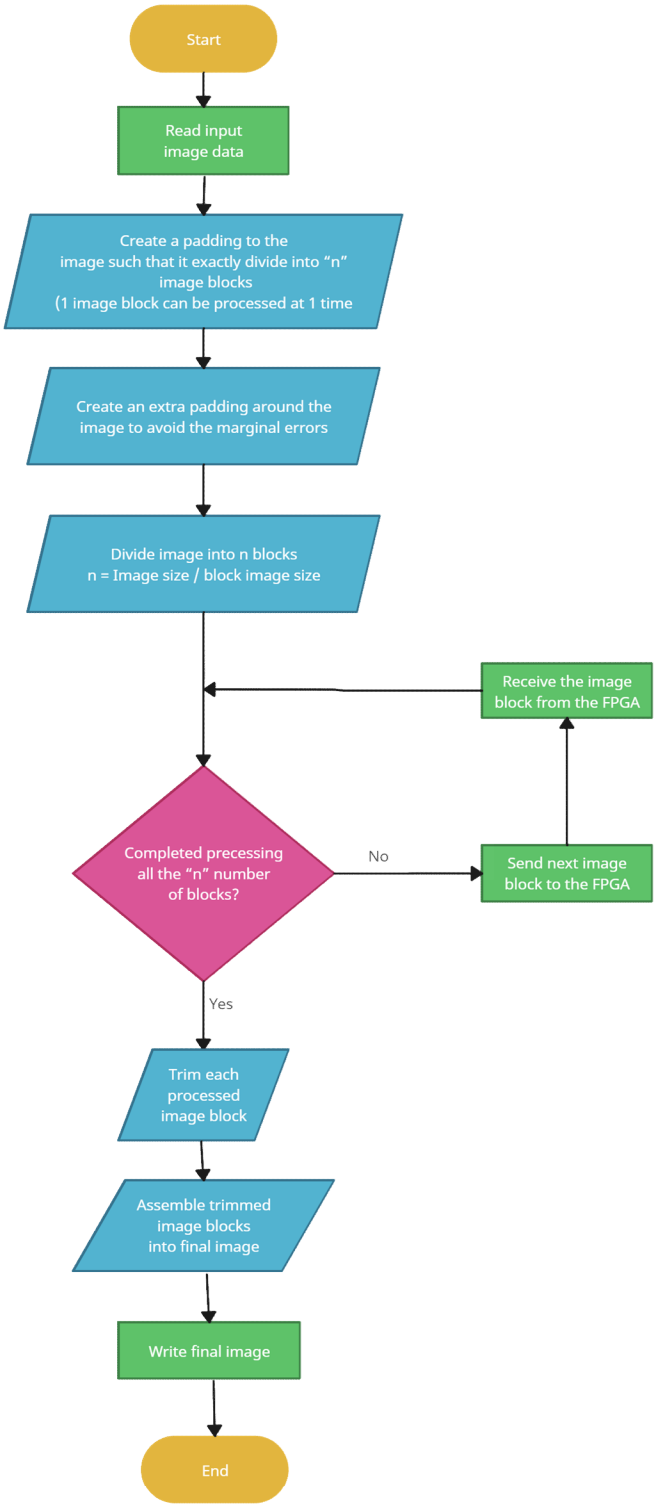

Figure 9

RESULTS

Here is the image we input. It consists with salt and pepper noise.

Figure 10

Here is the output from median filter and averaging filter

Figure 11: Output from Median Filter

Figure 12 : Output from Averaging Filter

OBSERVATIONS AND CALCULATIONS

We calculate time taken for the processing of 430x320 px image (Figure 10)

Size of a pixel	= 8b
No of pixels transmitting per 1 image block	= 7 x 7 = 49
No of bits transmitting per 1 image block	= 49 x 8b = 392b
No of bits receiving after processing 1 image block	= 392b
Baud rate of transmission	= 115,200b/s
Time taken to transmit and receive 1 image block	= 392b x 2 / 115,200b/s = 0.0068s
Pixels per image block	= 5 x 5 px = 25 px
Total number of pixels in the image	= 430 x 320 px = 137,600 px
No of image blocks in the image	= 137,600 px / 25px

	= 5504
Total time taken to transmit and receive the image	= 5504 x 0.0068s
	= 37.4272s

(Processing time and latencies are considered as negligible since they take time in nano seconds)

Observed time taken for the whole process = 44. 23s

DISCUSSION

Observed time is higher than the calculated time due to reasons like delays in the transmission, delays in the processing (errors in FPGA), Backpressure from the filter module, delays in the python code running.

UART is not suitable as the data transmission protocol since it take much time. Protocols like Ethernet or PCI should be used to experience for a real time processing.