ECE532 Final Report

Fingertip Air-Writing Recognition on FPGA

Yueqi Chen

Gujiang Lin

Qingnan Yu

2018-04-09

Table of Contents

1. Overview

1.1 Motivation

Human-Computer Interaction (HCI) technologies play increasingly important roles in today's world [1]. Motion gestures provide an easier and more direct way for humans to control or interact with the computer, compared with traditional devices such as the mouse and the keyboard. In recent years, motion recognition has been widely used to control hardware components, but it is not accurate enough to generate text input to computers. This leads to the research and development of air-writing recognition, which translates handwriting in the air to linguistic characters as digital data.

Air writing has been demonstrated by using wearable devices, such as an accelerometer and gyroscope augmented mobile phone, or a 3D gesture input device [2]. These designs do not provide the most natural writing experience because they rely on extra hardware peripherals, which must be worn or held all the time. A much more natural writing method is that users can air-write with their fingers only. Therefore, we have decided to build a camera-based fingertips air-writing recognition system with fingertips recognition that does not require any wearable devices.

1.2 Goals

Our goal is to build a system that translates fingertips movements in the air into English characters without any hardware peripherals on the user’s hand. The system should track the position of the fingertip using a video camera input and translate fingertip movements into English characters. The recognized characters will then be sent to the Internet for further processing and applications. By connecting the device to the Internet, we can explore many potential applications of air-writing recognition, for example, text messaging, web browsing, home remote control and virtual reality.

1.3 Block Diagram

The final block diagram of our system is shown in Fig. 1. Detailed description of each block is illustrated in Section 4.

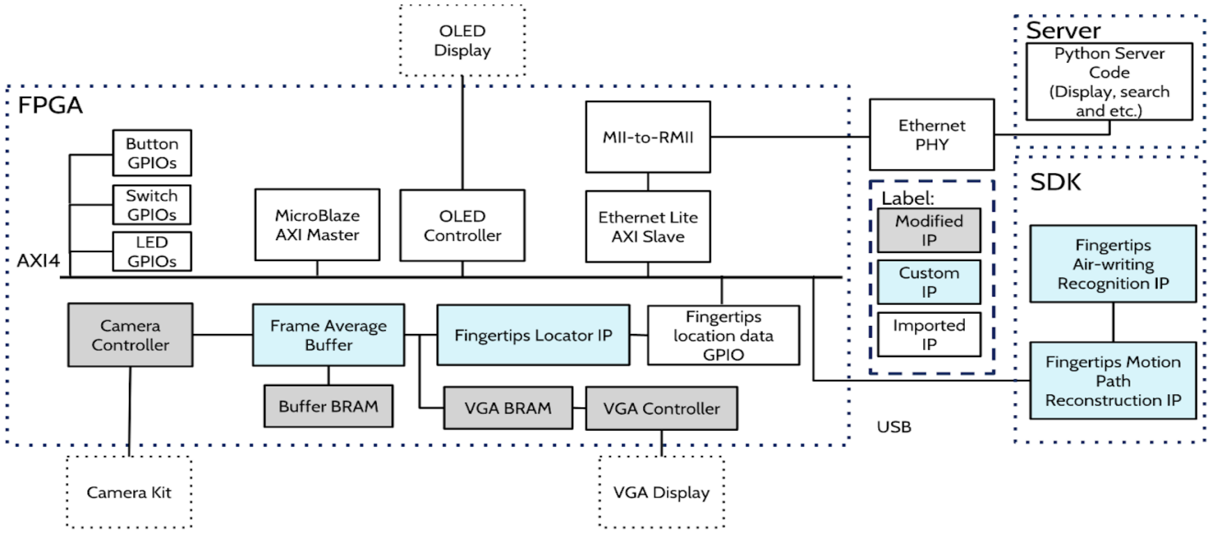


Figure 1. Block diagram of the system

1.4 Brief Description of IP

IP	**Function**	**Origin**
MicroBlaze	A 32-bits soft processor	Xilinx
BRAM	Store the buffering frames **blk_mem_buffer_vga**: store mono-color processed frame **blk_mem_gen**: store original YCbCr frame from the camera	Xilinx

IP	**Function**	**Origin**
GPIOs	General purpose input/output core **axi_gpio_LED**: used for debugging purpose **axi_gpio_SW**: control character recognition feature **axi_gpio_BTNL/BTNR**: reset and send message **axi_gpio_ftps_data**: send 32-bits data fingertip.	Xilinx
Ethernet	Send recognized characters to a computer server through the on-board Ethernet port The block design is created by following the Diligent Tutorial. Software code from the echo server example is modified so that the hardware system works as a TCP/IP client.	Xilinx Group
PmodOLEDrgb	Write recognized characters on Pmod OLEDrgb (a 96x64 pixel graphics OLED display) The IP block is imported from Digilent Vivado Library. Software code is modified to display characters based on its example demo program.	Xilinx
video_in	Format the input frame from the Pmod camera OV7670 Camera configurations are modified: 1. RGB is changed to YUV 2. HREF, HSTART, HSTOP, VSTRT and VREF values are modified to eliminate the offset	Piazza Modified
vga	Read pixels stored in block RAM and draw frames on VGA display (The size of each frame is 320x240.)	Piazza

IP	**Function**	**Origin**
frame_average_buffer	Take input data from the camera and output the average of two frames	Group
ftps_locator	Locate the fingertip position	Group
vga_data_mux (deprecated)	Switch 14 is used to select the VGA output between the original camera input and the mono-color image with fingertip location	Group

2. Outcome

2.1 Results

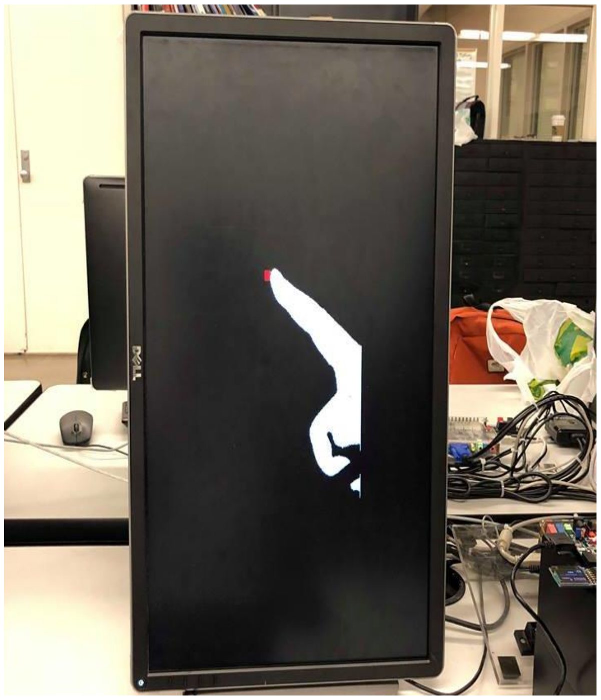
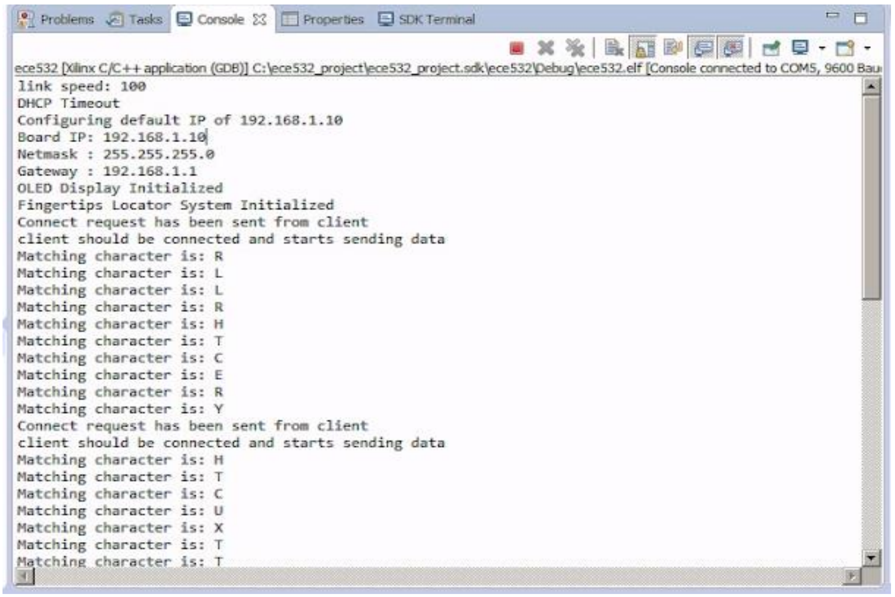


Figure 2. Location of the fingertip is displayed on a VGA monitor



```
ece532 [Xilinx C/C++ application (GDB)] C:\ece532_project\ece532_project.sdk\ece532\Debug\ece532.elf [Console connected to COM5, 9600 Bau
link speed: 100
DHCP Timeout
Configuring default IP of 192.168.1.10
Board IP: 192.168.1.10
Netmask : 255.255.255.0
Gateway : 192.168.1.1
OLED Display Initialized
Fingertips Locator System Initialized
Connect request has been sent from client
client should be connected and starts sending data
Matching character is: R
Matching character is: L
Matching character is: L
Matching character is: R
Matching character is: H
Matching character is: T
Matching character is: C
Matching character is: E
Matching character is: R
Matching character is: Y
Connect request has been sent from client
client should be connected and starts sending data
Matching character is: H
Matching character is: T
Matching character is: C
Matching character is: U
Matching character is: X
Matching character is: T
Matching character is: T
```

Figure 3. Example SDK output

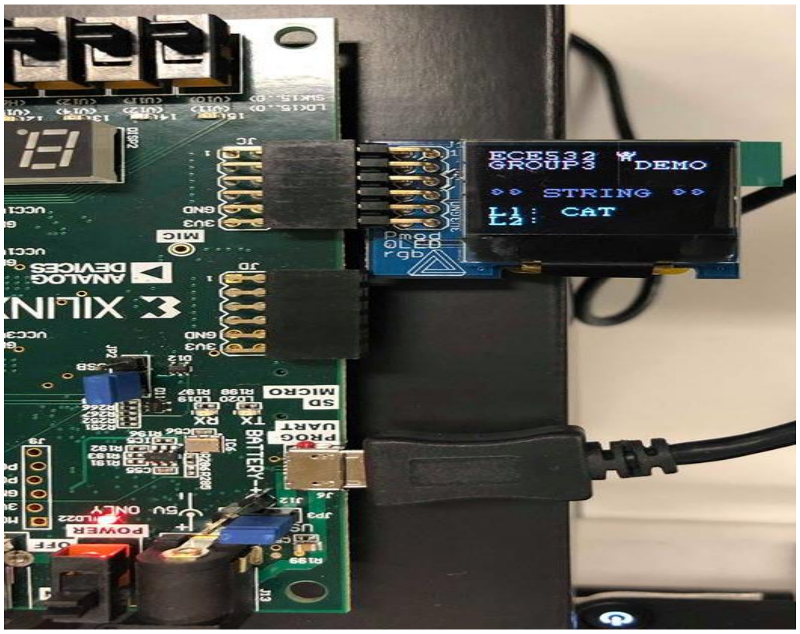
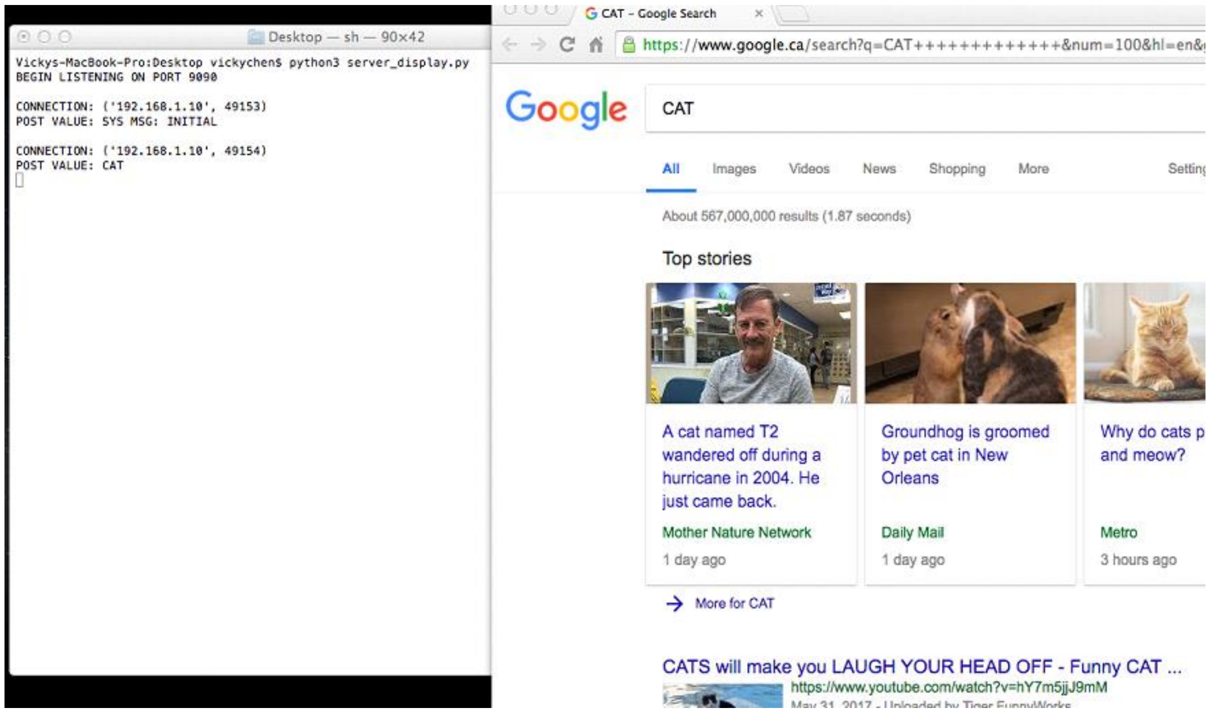


Figure 4. Displaying recognized characters "CAT" on OLEDrgb



```
Vickys-MacBook-Pro:Desktop vickychen$ python3 server_display.py
BEGIN LISTENING ON PORT 9090

CONNECTION: ('192.168.1.10', 49153)
POST VALUE: SYS MSG: INITIAL

CONNECTION: ('192.168.1.10', 49154)
POST VALUE: CAT
```

About 567,000,000 results (1.87 seconds)

A cat named T2 wandered off during a hurricane in 2004. He just came back.
Mother Nature Network
1 day ago

Groundhog is groomed by pet cat in New Orleans
Daily Mail
1 day ago

Why do cats p and meow?
Metro
3 hours ago

CATS will make you LAUGH YOUR HEAD OFF - Funny CAT ...
https://www.youtube.com/watch?v=hY7m5jjJ9mM

Figure 5. Sending recognized characters "CAT" to computer server and searching it on Google

Table 1 and Table 2 below summarize the status of the proposed functional requirements and acceptance tests.

Table 1. Functional requirements and features

Functional Requirements and Features	**Status**
Tracking the fingertip to generate the fingertip motion path	Complete, as shown in Fig. 2
Reconstructing and formatting the fingertip air-writing data	Complete
Recognizing the English characters from the fingertip air-writing data	Complete with lower accuracy than expected, as shown in Fig. 3
Displaying English characters on the OLEDrgb	Complete, as shown in Fig. 4
Searching the recognized word on Google	Complete, as shown in Fig. 5

Table 2. Acceptance tests

Acceptance Criteria	**Status**
The system shall recognize all English characters and Arabic numbers	Only English characters are recognizable. Recognition of Arabic numbers needs to be implemented.
The system shall correctly identify the user's fingertip from video input and track its movements under different lighting conditions and background environments	Background has to be black. Implementation of background subtraction algorithm is needed
The system shall generate characters from handwritings of users who have different writing styles, writing speeds, hand sizes and skin colors with at least 80% accuracy	Template-matching algorithm does not work well with different writing styles. An algorithm that can recognize different writing styles is needed
OLED screen shall display the characters no later than one second after the user has finished writing	Pass

When the user writes in the air, the system will detect the fingertip and track its movements. After one character is processed, it will be displayed on the OLED screen. Once an entire word is processed, it will be sent to the server through the Ethernet and the server will search that word on Google, as shown in Fig. 5.

The final design completes all the proposed features with some constraints:

- Background has to be black
- Speed of writing is limited by frame rate of the camera (30 frames per second)
- Some characters have low accuracy, such as K, M, N, P, Q, W, X

If we could start over, we would improve the accuracy of the character recognition algorithm. Since character recognition can be implemented and tested independent of other parts of the project, we should start researching on the algorithms that can provide more flexibility for different writing styles in the early design phase.

2.2 Possible Further Improvements

The accuracy of the character recognition algorithm could be improved. The template-matching algorithm is relatively primitive because the templates themselves can hardly match different handwriting styles. In the future, we shall research on intelligent character recognition, which can recognize different writing styles. It would be a much greater success if we can implement machine learning in this project.

If someone were to take over the project, I would recommend doing research on the intelligent character recognition algorithm and implementing it on hardware as next steps. They can also implement background subtraction so that this project can work in different environments.

3. Project Schedule

Milestone #	Original plan	Actual accomplishments
1	System interface and Ethernet setup	Created the block design and integrated TCP/IP client implementation
2	Test the camera kit and implement image processing IP	**Yueqi Chen**: integrated and tested Pmod camera kit **Gujiang Lin**: integrated and tested Frame Average Buffer IP **Qingnan Yu**: researched on character recognition algorithm

Milestone #	Original plan	Actual accomplishments
3	Devise an algorithm that generates writing path from multiple images for air-writing recognition	**Yueqi Chen**: implemented and tested correlation calculation algorithm in MATLAB **Gujiang Lin**: implemented Image Processing and Path Reconstruction IP **Qingnan Yu**: implemented character recognition algorithm using template matching in C
4	Implement character recognition	**Yueqi Chen**: integrated character recognition software code on SDK and tested with ideal data input **Gujiang Lin**: implemented Fingertips Locator IP **Qingnan Yu**: modified templates and loaded onto SDK
5	Mid-project demonstration	Tested template matching on SDK with actual fingertip movement data and Fingertips Locator IP The system can track the fingertip location and encode that information into a mono-color image. The template matching code can compare the fingertip movement and templates, then output a character with the highest correlation. The system works with a low accuracy and needs further improvement.
6	System-level debugging and optimization	**Yueqi Chen**: integrated Pmod OLEDrgb to the system **Gujiang Lin**: fixed errors in Fingertips Locating IP and merged 3 GPIO into 1 **Qingnan Yu**: hardware implementation of template matching algorithm (incomplete)
7	Final Demonstration	Improved accuracy for character recognition

We have been consistent with original milestones but with some deviations:

1. Due to time constraint, we could not finish implementing character recognition algorithm on hardware.
2. Pmod OLEDrgb was integrated to the design until Milestone 6 since we just found the latest Pmod library from the official Diligent GitHub account.

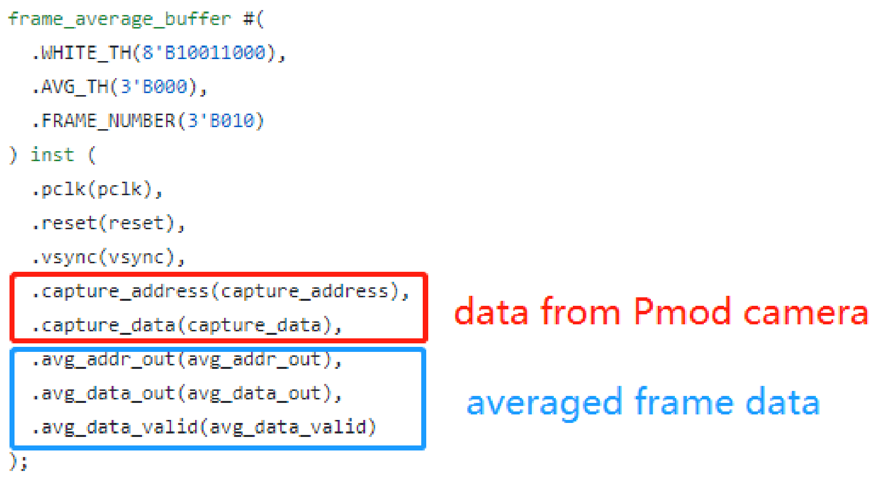
Each team member was assigned on individual tasks and followed weekly milestone deadline. Therefore, we were able to deliver our design at the final demonstration.

4. Description of the System Components

4.1 Custom IP

Modules listed in this section are implemented by the team.

Frame_average_buffer: This buffer module takes input data from the camera and outputs the average of two frames. This module will first compare the camera's input data in YUV format to a threshold value, WHITE_TH. The intermediate data will be "1" if the input is greater than the threshold or "0" if not. This buffer module contains one sub-block memory to store the frame data. Once the frame counter reaches FRAME_NUMBER, the stored frame data will be compared with another threshold value, AVG_TH, and output "1" (human hand area) if the input is greater than the threshold or "0" (background area) if not. Fig. 6 shows the module declaration.

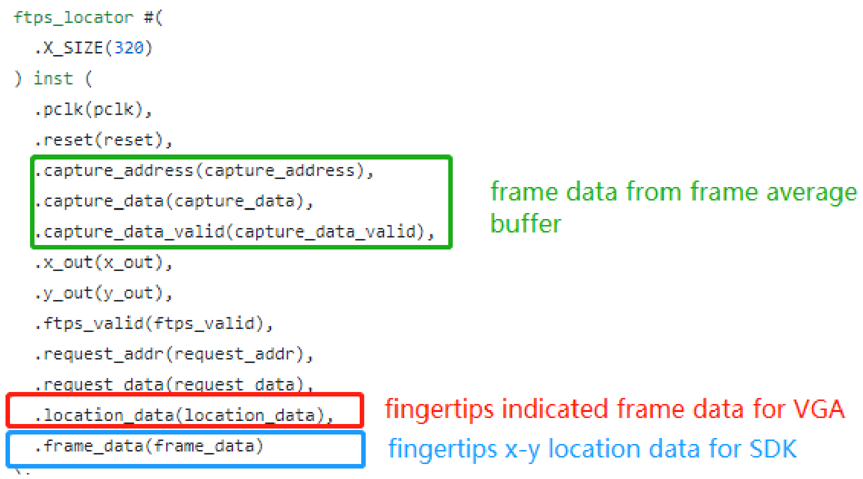


```
frame_average_buffer #(
  .WHITE_TH(8'B10011000),
  .AVG_TH(3'B000),
  .FRAME_NUMBER(3'B010)
) inst (
  .pclk(pclk),
  .reset(reset),
  .vsync(vsync),
  .capture_address(capture_address),
  .capture_data(capture_data),
  .avg_addr_out(avg_addr_out),
  .avg_data_out(avg_data_out),
  .avg_data_valid(avg_data_valid)
);
```

Figure 6. frame_average_buffer module interface

To test this module, the 1-bit output is expanded into 16 bits ("0" corresponds to "0x0000" and "1" corresponds to "0xFFFF"), and displayed on a VGA monitor. The result should be a mono-color human hand shape.

ftps_locator: This module reads the frame data from the frame average buffer, analyzes the fingertip location from each frame and indicates the location with a 10x10 square on the frame with different colors. This module will first find out which direction the user's hand is entering from (top, bottom, left and right), and indicate the color accordingly (blue for top, cyan for bottom, red for right, green for left and yellow for none). According to the entry direction, fingertip location x_out and y_out are chosen from the minimum or maximum coordinates of human hand area. Fig.7 shows the module declaration. This module has two outputs. One is connected to the VGA block memory for the VGA display, as shown in Fig. 2. The other output is connected to gpio_ftps_data to be read by the Microblaze processor that runs functions implemented on SDK.



```
ftps_locator #(
  .X_SIZE(320)
) inst (
  .pclk(pclk),
  .reset(reset),
  .capture_address(capture_address),
  .capture_data(capture_data),
  .capture_data_valid(capture_data_valid),
  .x_out(x_out),
  .y_out(y_out),
  .ftps_valid(ftps_valid),
  .request_addr(request_addr),
  .request_data(request_data),
  .location_data(location_data),
  .frame_data(frame_data)
);
```

frame data from frame average buffer
fingertips indicated frame data for VGA
fingertips x-y location data for SDK

Figure 7. frame_average_buffer module interface

This module is tested by displaying the output on the VGA monitor. The result should indicate the position of the fingertip correctly with corresponding color based on entering direction of the user's hand.

vga_data_mux: This is a 2-to-1 mux module controlled by SW14. It will select the data input between the original video frame and fingertips located mono-color frame, and display it on the VGA monitor. This module was used only for internal test and was deprecated during the final demonstration.

4.2 Modified IP

The modules listed in this section are imported either from the Xilinx official library or from project examples provided on Piazza. The parameters and some given code are modified.

axi_gpio_LED: General-purpose input/output IP connected to the LED lights. LED0 indicates whether Pmod video camera has finished its configuration. LED1 indicates whether Microblaze is processing character recognition. LED15 indicates whether the characters recognition feature has been turned on.

axi_gpio_SW: General-purpose input/output IP connected to the switches. SW14 selects the frame data being displayed on the VGA monitor (for internal testing only, deprecated). SW15 controls the character recognition feature.

axi_gpio_BTNL/BTNR: General-purpose input/output IP connected to the button keys. Pressing BTNL resets the whole system. Pressing BTNR sends the recognized characters to the computer server through the Ethernet port.

axi_gpio_ftps_data: General-purpose input/output IP connected to the fingertip locator IP. The output is 32 bits, containing the valid bit, x location and y location for fingertip data.

blk_mem_buffer_vga: The block memory IP stores the mono-color processed frame data for VGA (16 bits * 76800).

blk_mem_gen: The block memory IP stores the original YCbCr frame data from the camera for VGA (16 bits * 76800).

video_in: This module formats frames from the video camera. Example code is provided on Piazza and we changed output configuration to YCbCr instead of RGB.

4.3 Imported IP

The modules listed in this section are imported from the Xilinx official library or from project example provided on Piazza. They are not modified.

MicroBlaze: Microblaze IPs (including related IPs, such as microblaze_0_axi_intc, microblaze_0_axi_periphdesign, MicroBlaze and etc.) are imported by following the tutorial provided on Piazza.

axi_ethernetlite: Ethernet IPs (including related IPs, such as mig_7series, mii_to_rmii and etc.) are imported by following the Diligent Tutorial. This module is tested by sending strings from the Nexys4 DDR board to the computer server through the Ethernet port.

PmodOLEDrgb: The IP block is imported from Digilent Vivado Library. It displays characters and strings on the 96x64 Pixel Graphics OLED screen. This module is tested by sending different characters with various colors at different positions on the Pmod OLED display.

vga444: VGA display controller IP provided with Pmod camera example on Piazza. This module is tested by displaying the video input from the Pmod camera on a VGA monitor.

4.4 SDK

The SDK code was implemented based on the LwIP Echo Server template. We modified main.c and echo.c files.

main.c: The main function first initializes the Ethernet connection between computer server and the Nexys4 DDR board (client). After the board is connected to the server, the processor sends "SYSTEM INITIAL" to the server to test the connection. The processor then remains in an infinite while loop, and executes the following functions in order:

- Check if SW15 is turned on to enable character recognition. If character recognition is turned on, it will execute the following functions in order:

    - Read fingertip x-y location data from gpio_ftps, scale the coordinates from 320x240 to 36x26 and store the data into a 36-line integer frame array. Each bit in the frame array represents one pixel on the frame.
    - Compare the frame array to 26 predefined template arrays (A-Z). When there is no more fingertip data to be read (X=0 and Y=0 for 5 continuous frames), the algorithm starts correlation computation.
    - Store the recognized character result from the template-matching algorithm in a string.
    - Update OLED display with the recognized character.
- If BTNR is pressed, send the recognized characters to the server.
- Check if there are TCP sending requests that need to be handled.

The following tests were done to ensure the main function works correctly:

- Both raw x-y location and compressed x-y location data were printed on the Terminal to check correctness.
- Inputting the golden results to test correlation function and the accuracy is 100%.
- Sample strings were sent to OLED display to check PmodOLEDrgb module.
- Sample strings were sent to the computer server to make sure the Internet protocol works correctly.

echo.c: This source file contains all the helper functions and predefined data for main.c

- **int template[26][36]:** This is a predefined template array for 26 capital English letters (A-Z), and each bit in the array represents one pixel on the frame.
- **int start_application:** This is a helper function that sends the message to the server through Ethernet. The maximum input is 16 characters.

- **My_OLED_Initialize/ My_OLED_InitDATA/My_OLED_UPDATE_DATA**: These are helper functions that update the OLED display.
- **Char_recognition:** This is a helper function that calculates the correlation between one template array and 26 templates for character recognition.

5. Description of Design Tree

The most important files in our project directory are listed below.

ece532_project - root directory

- /ece532_project.sdk
    - /ece532/src/
        - echo.c: includes TCP/IP client code, template matching code, and PmodOLEDrgb code
        - main.c: main software code file that controls the overall flow of the system
- /ece532_project.srcs
    - /constrs_1/new/
        - eth_ref_clk.xdc: constraint file
- /ip_repository: contains the IP modules used in the project
    - /utoronto.ca_user_frame_average_buffer_1.0: contains the custom Frame Average Buffer IP
    - /utoronto.ca_user_ftps_locator_1.0: contains the custom Fingertips Locator IP
    - /utoronto.ca_user_vga444_1.0: contains the VGA module provided on Piazza
    - /utoronto.ca_user_vga_data_mux_1.0: contains the custom vga_data_mux module used for debugging purpose (deprecated)
    - /utoronto.ca_user_video_in_1.0: contains the Pmod camera module provided on Piazza
    - /vivado-library-master: contains the PmodOLEDrgb module

- /server_py
    - server_display.py: python code for TCP/IP server
- ece532_project.xpr: the main project

6. Tips and Tricks

- Learn how to use the DSP unit as soon as possible
- When designing software for Microblaze, always estimate the memory usage of the code
- Never assume the camera will give you “clean” data, pre-processing or filtering is necessary because the input is always noisy

Bibliography

[1] S. Xu and Y. Xue, "Air-writing characters modelling and recognition on modified CHMM," *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC),* pp. 001510-001513, 2016.

[2] R. Islam, H. Mahmud, M. K. Hasan and H. A. Rubaiyeat, "Alphabet Recognition in Air Writing Using Depth Information," *the Ninth International Conference on Advances in Computer-Human Interactions (ACHI)*, pp. 299-301, 2016.