The Edward S. Rogers Sr. Department of
Electrical and Computer Engineering
University of Toronto

ECE532 Digital Systems Design

Final Design Report

Air Draw

	Name	Email
Team 1	Charles Huang	zichun.huang@mail.utoronto.ca
	Lei Liu	liulei.liu@mail.utoronto.ca
	Ourong Lin	hughor.lin@mail.utoronto.ca
	Haley Han	haley.han@mail.utoronto.ca

Submission Date: April 14, 2023

Acknowledgements

We would like to extend our sincere thanks to Professor Jason Anderson and Daniel for their unwavering support and invaluable feedback during our design project. Their guidance was crucial in helping us navigate the challenges we faced, particularly in the initial stages when we struggled to clarify our objectives. Without their assistance, it would have been significantly more challenging to overcome the obstacles we encountered, and we greatly appreciated their constructive comments to help alleviate our confusion.

Furthermore, we would like to acknowledge the commitment and perseverance of each team member who worked tirelessly to overcome numerous obstacles and achieve success in this endeavor. We firmly believe that every member of the team, as well as those who provided their support along the way, deserves credit for our accomplishment.

— Team 1

Table of Contents

1. Overview

1.1 Background & Motivation

Our team intends to develop a drawing system utilizing the Xilinx Nexys 4 DDR FPGA and its peripherals. This system will enable users to draw any shape in the air, with the captured drawing traces saved as a computer image. The inspiration for this project was derived from the "sparkler drawing" technique, a time-lapse photography approach used with sparkler fireworks. Our project, called AirDraw, aims to enhance the drawing experience by offering an "air canvas" in place of a conventional screen. Basic drawing tools, such as the pen, pen colors, and eraser, will be remotely accessed through a capacitive button controller. AirDraw will be capable of detecting user motions via camera based on the positions of a bright flashlight. The system will process the drawing patterns and display the trace simultaneously on a VGA screen.

1.2 Block Diagram

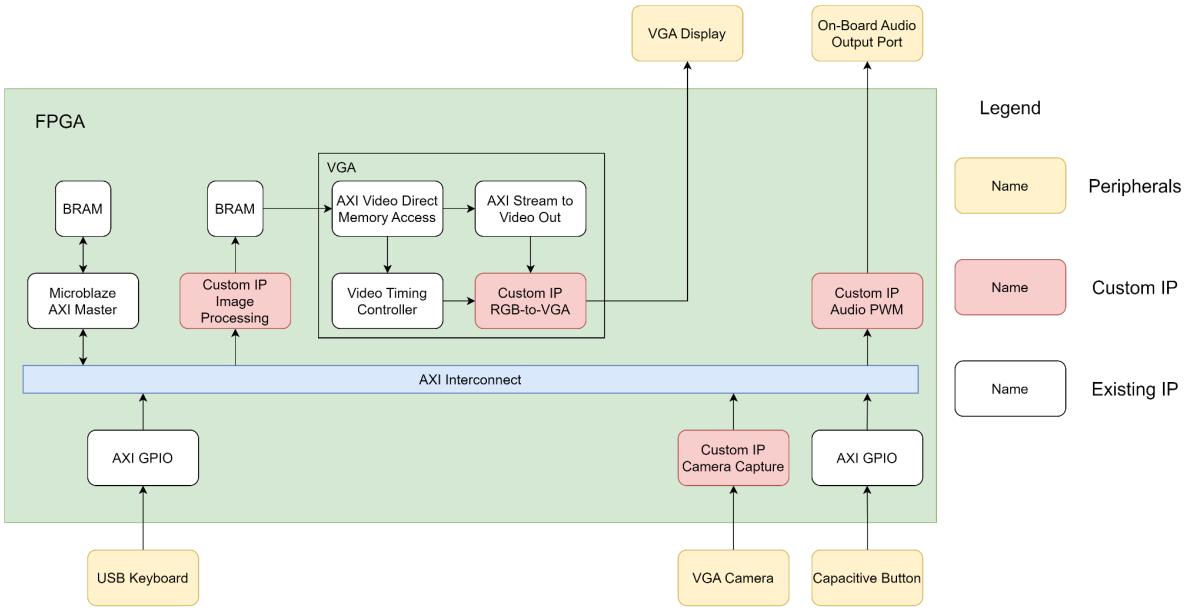

Figure 1: Final high-level block diagram of AirDraw

1.3 Brief Description of IP used

From the block diagram above, all the red blocks represent custom IPs that were either implemented or modified from an existing online source by the team. The white blocks such as the IPs within the VGA, BRAMs, GPIOs, and the MicroBlaze are all existing IPs from the Vivado IP library.

1.3.1 MicroBlaze Processor Code

The process simply runs one c file that contains all VGA drawing functionalities along with a big polling loop that iterates to check data being sent from all other peripherals.

- It receives pixel locations from the image processing unit and sets the corresponding pixel in the memory for VGA to be displayed.
- It polls the scancode from the keyboard in a certain mode and draws corresponding letters on VGA.
- It polls the capacitive button input and either switches the drawing mode or enables drawing on the VGA based on the input. It also sends the corresponding sound effect code to the audio module.
- A menu allows the user to choose different drawing modes (i.e. change colors, line width, erase, etc.)

1.3.2 VGA

The VGA unit consists of all IPs from the existing Vivado IP library except for the RGB-to-VGA block which was taken from the Digilent IP library (more detail will be discussed in section 4.1).

1.3.3 Camera Capture + Image Processing

The custom IP of camera capturing and image processing was combined into a large Verilog module along with custom divider and communication IPs.

- It reads data from the video camera (i.e. RGB-888 code).
- After receiving the whole frame, it applies a grayscale filter.
- It looks for all pixels that have luminescence above a threshold and computes a rolling sum of these pixel coordinates.
- The coordinates are sent to the divider IP and compute an average.
- The averaged output is sent to the communication IP and will be stored into a BRAM.
- Two modules were taken from an online source that basically loads a register file for the camera (more detail will be discussed in section 4.2).

1.3.4 Keyboard & Capacitive Button & Audio

Each peripheral has their own custom IP along with GPIOs connecting to them. GPIOs are connected to the MicroBlaze through an AXI interconnect to either send output or receive input of the IPs.

- The keyboard IP takes keyboard input and transfers the correct scancode to the GPIOs.
- The capacitive button IP takes button input and transfers the on/off bits to the GPIOs.
- The Audio IP takes input from GPIO (that receives data from the MicroBlaze) and generates a specific pattern of pulse width modulation to play a sound. The audio module was taken from an online source but was modified significantly (more detail will be discussed in section 4.4).

1.3.5 Other IPs

BRAMs: block RAM

AXI interconnect: Interconnect for all modules to communicate with the MicroBlaze

Processor_System_Reset: allows external resets

UART: Allows USB interaction with the processor

Clock Generator: Provides clocks

2. Outcome results

2.1 Project Results

The final project accomplished proposed features with the exception of TCP transmission and meets all system design constraints and requirements as shown in the table below. Furthermore, along the path of our development, the team was able to abound system functionalities and optimize module implementations to achieve higher performances. A simple output of our design is shown in Figure 2 and the detailed Demo video can be found at: https://youtu.be/JYAf4uGFNko

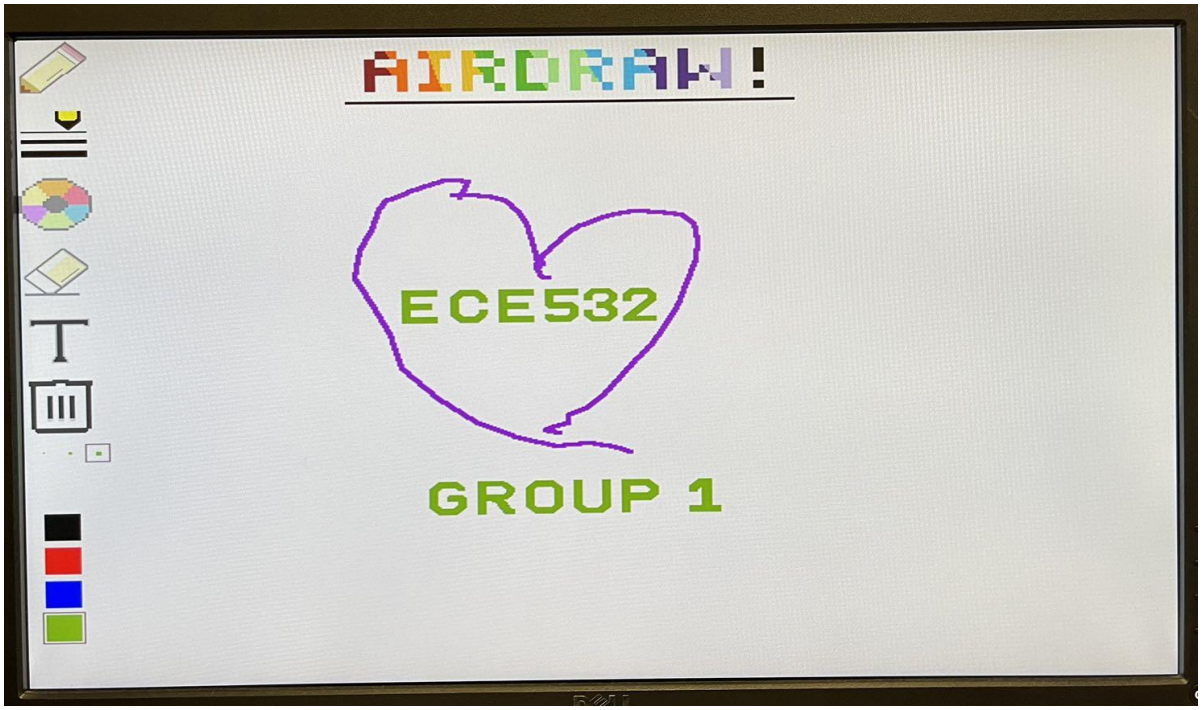

Figure 2: Design Output Sample

2.2 What If ?

Initially, the project made progress as expected, but some functional blocks faced obstacles that were initially deemed impossible. However, after seeking consultation from our TA, it was discovered that these blocks were indeed feasible. Additionally, the project was based in Vivado EDA and required cosmic supporting files, which made version control challenging as more custom modules were added. As a result, the team had to resort to using portable storage devices for version updates and system level integration.

If the team had the opportunity to start over, we would engage in discussions with their TA and intra-team members not only during tutorial sessions but also through communication channels such as Piazza and email when facing bottlenecks. We would also spend time seeking a version control machine or service that alleviates the frustration involved in integration and merging. The team excelled in other aspects of their development, including research, communication, and collaboration, which we intend to continue with their current approach.

2.3 Future Improvement & Next Steps

In scrutiny, it became evident that the primary bottleneck in the design lies in the image processing of the camera input, specifically in the detection of pixels with higher luminance levels. Ideally, the image processing method should be both rapid and precise, capable of eliminating any outliers and accurately reporting the position to the MicroBlaze interface. While the project achieved satisfactory results in detecting luminescent pixels, the team only utilized basic grey scale processing without incorporating more advanced image processing techniques such as a Gaussian filter or Sobel operator which requires the former. Therefore, future development will involve implementing a Gaussian filter at the RTL level first to eliminate noise and potential outliers before approaching more advanced computer vision algorithms.

3. Project Schedule

<table>
<tr><th>MS#</th><th>Original Plan</th><th>Actual Accomplishments</th></tr>
<tr><td rowspan="2">1</td><td>☐ Create a draft block diagram in Vivado
☑ Be familiar with required peripherals, protocol, and necessary IP blocks.</td><td>- Research on VGA and USB keyboard peripherals</td></tr>
<tr><td colspan="2">The group divided the work and mainly focused on research, and realized that without further research, it was hard to create a block diagram from scratch.</td></tr>
<tr><td rowspan="2">2</td><td>Create testbenches on:
☑ Typing/clicking on keyboard/mouse
☐ Audio output
☐ Capture frame from the camera
☑ Plotting pixels on a VGA display with a given position</td><td>- Create and validate the block diagram design connecting MicroBlaze and VGA
- Create packaged keyboard IP with MicroBlaze through GPIO</td></tr>
<tr><td colspan="2">The mouse was removed from the project plan due to the device resource limitation. The group members faced challenges using Vivado to create block designs, which caused them to spend more time than anticipated on the task. Therefore, the research on the camera and audio was moved to future milestones.</td></tr>
<tr><td rowspan="2">3</td><td>☐ Implement the image processing algorithm in hardware
☑ Implement ping-pong buffering for display
☐ Ensure the functionality of the capacitive button</td><td>- Improve VGA frame speed
- Implement the functions of drawing lines and shapes in SDK
- Completed the Camera module, and display onto the VGA
- Research on capacitive button</td></tr>
<tr><td colspan="2">The group just started to research the camera, therefore the image processing algorithm was moved to the next milestone. A functional camera demo was completed at the end of this milestone. The group decided to use a single buffer by examining the speed of playing animations. Aside, the group started to plan on the user interface design and implement some basic functions. The group tried to verify the button signals through I2C protocol but failed.</td></tr>
</table>

<table>
<tr><td rowspan="2">4</td><td>☐ Finalize the image processing algorithm
☑ Able to display output from the algorithm onto VGA display
☑ Integrate the camera onto the system</td><td>- Implement the first version of image processing algorithm;
- Integrate the keyboard, the capacitive button, and image processing algorithm into the main project
- Implement the ‘Draw’, ‘Line Weight’, ‘Colour’, and ‘Erase’ Mode using SDK.</td></tr>
<tr><td colspan="2">The image processing algorithm still had space for improvement such as compute the division in hardware to improve speed and use filters to improve the accuracy of capturing position. The keyboard integration was done a week prior to the scheduled milestone 5.</td></tr>
<tr><td rowspan="2">5</td><td>☐ Integrate keyboard/mouse, capacitive button, and audio onto the system
☐ Establish network modules within the system</td><td>- Research on audio.
- Improve on the image processing algorithm by detecting luminescence rather than RGB.
- Implement the ‘Text’ mode and add a new function in ‘Colour’ mode (software).</td></tr>
<tr><td colspan="2">The project plan had significant changes compared to the original plan. The function of exporting data from FPGA through the network was removed from the project plan. The audio was not verified and the group continued debugging on it.</td></tr>
<tr><td rowspan="2">6</td><td>☑ Complete all remaining work and practice design optimization
☑ Finalize and conduct final testing on the design</td><td>- Integrate the audio into the main project.
- Implement the ‘Clear’ Mode and add audio effects using SDK.
- Finalize design and conduct final testings on corner cases.</td></tr>
<tr><td colspan="2">The group completed all the tasks on time.</td></tr>
</table>

Overall, the group has an effective management on the tasks and each team member has been actively and significantly contributed to the project. Some of the plans were delayed due to the high amount of time debugging on the hardware block designs since every single change on the block design requires a recompilation on the synthesis, implementation, and bitstream generation. In addition, the group should have allocated more time for the development of applications on software logics such as the user interface design and image processing algorithm.

4. Description of Blocks

The whole system we designed can be broken down into six parts including five peripheral subsystems and one MicroBlaze processor. The session will be discussing all the IP blocks used within each subsystem.

4.1 VGA

The VGA subsystem contains these blocks: Dynamic Clock Generator, Video Timing Controller, AXI Video Direct Memory Access, AXI Interconnect, AXI4-Stream to Video Out, AXI Interconnect, Memory Interface Generator, and RGB to VGA Output. All these blocks are pre-existing IP blocks from the Vivado IP library, except for RGB to VGA Output which is taken from the Digilent Github site: https://github.com/Digilent/vivado-library.

The VDMA receives data from the MicroBlaze processor and pushes the processed RGB data into an AXI-Stream to Video Out buffer. The output of this buffer will be converted into actual pixels through a RGB to VGA block in the right top corner, and will eventually be sent to the VGA output ports. The dynamic clock generator and the video timing controller are used to produce a clock signal to the video buffer.

4.2 Camera

The camera subsystem contains the following blocks along with their origin listed:

Name	Origin	Link
AXI BRAM Controller	Vivado IP Library	N/A
Block Memory Generator	Vivado IP Library	N/A
camera capture	Online Resource, Modified	https://github.com/okchan08/OV7670
average	Originally Created	N/A

comm	Originally Created	N/A
camera button	Online Resource, Unmodified	https://github.com/okchan08/OV7670
camera controller	Online Resource, Modified	https://github.com/okchan08/OV7670

The "camera button" and "camera controller" are from an online GitHub source. They simply perform the functionality of receiving on-board button signals from the FPGA ("camera button") and starting to load a register file ("camera controller"). The only changes we made to the "camera controller" were to write our own register file that includes the modes that we desired. For example, we changed the RGB format from RGB-888 to RGB-565 for debugging on VGA since the Nexys Board VGA port only supports RGB-565 format.

The "camera capture" is also from the same source, but we made many modifications to it. The original "camera capture" simply takes an 24-bit RGB code input from the camera input ports and directly passes it to the VGA units. We added part of our image processing algorithm into the block so that it now takes the RGB input, converts it to grayscale, detects a certain threshold, and computes a rolling sum of all the above-threshold pixel coordinates.The “average” modules were used to compute the average by dividing the calculated rolling sum from the “camera capture” output by the number of detected pixels. The “comm” module then stores these averages to the BRAM in different addresses so that the MicroBlaze can read the averages directly through an AXI BRAM Controller.

4.3 Capacitive Button & Keyboard

The capacitive button and keyboard subsystems both contain two blocks, including one AXI GPIO block and another custom IP block we created ourselves. Within the capacitive button subsystem, there is a "button control" block, which the team wrote, that simply takes the input of the button peripheral and sends the output to an AXI GPIO, which is a pre-existing Vivado IP block. Similarly, within the keyboard subsystem, we created a "PS2Receiver" block that takes the keyboard input and sends a corresponding scancode to an AXI GPIO.

4.4 Audio

The audio subsystem contains one modified IP from an online Github resource and three pre-existing Vivado IPs. The modified IP is called "PWMAudio" (https://github.com/sburg3/pwm-audio), which originally just produces a single tone by changing the on-board switches. We modified it so that it now takes input from the MicroBlaze to select which audio tone it is going to play and outputs the correct PWM signal to produce the multiple correct audio tones. The input of the previous IP comes from an AXI GPIO attached to two Slices that separate the most significant bit from the total input. These AXI GPIO and Slices are pre-existing Vivado IP blocks.

4.5 MicroBlaze

The rest of the system is entirely a MicroBlaze processor and its auxiliary IP blocks. These blocks contain: Clocking Wizard, MicroBlaze itself, MicroBlaze Debug Module, MicroBlaze Local Memory, two Processor System Resets, AXI Interconnect, AXI Uartlite, AXI Interrupt Controller, and Concat. All of them are pre-existing Vivado IP blocks. These blocks are all preliminary modules needed for the MicroBlaze processor to behave correctly.

5. Our Design Tree

The Github link is available at: https://github.com/Leiliu99/ece532_airdraw

5.1 Project Overview

AirDraw is a design developed using Nexys 4 DDR FPGA board. Users can move the flashlight in front of the camera, VGA will output the user's movement on the screen. Users can further add text onto VGA through the keyboard. Capacitive button is also used as an enable signal and switching between menus.

Please check our report for a detailed description.

You can also watch our demo here: https://youtu.be/JYAf4uGFNko

5.2 Hardware requirement

Vivado Design Suite(version 2018.3)

Nexys 4 DDR FPGA board

OV7670 camera (connected to JA, JB ports)

Pmod CDC1: Capacitive Input Button (connected to JD port)

keyboard (connected to USB port)

speaker/earphones with jack plug (connected to audio port)

5.3 File Description

We only summarized the key files below.

vga_microblaze.xpr: this is the main project files.

vga_microblaze.srcs: this contains block diagram, custom ips and custom RTL module.

- /sources_1/new/PWMAudio.vhd: toplevel custom RTL module for audio
- /sources_1/new/button_control.v: custom RTL module for capacitive button
- /sources_1/new/camera_capture.v: custom RTL module for camera
- /sources_1/new/average.v, comm.v: custom RTL modules for image process
- /sources_1/new/PS2Receiver.v: custom RTL module for keyboard

vga_microblaze.sdk: Vivado software development kit that contains c code for GUI operation of the design.(Please check vga_microblaze.sdk/vga_cam/src/).

- /zybo_vga: sw setup for vdma
- /letter_array: const c array stored for letters and numbers' pixel
- PS2Decoder.h: keyboard decoding
- image_arrays.h: const c array stored for menu logos and title
- btn_crtl/draw: helper functions
- main.c: main software program

vga_microblaze.ip_user_files: this contains the pre-existing IP files

Other sub-folders contain compiler information which are needed if you download the project and run it locally.

5.4 Running AirDraw

Open vga_microblaze.xpr with Vivado. You may need to upgrade ip's or re-compile the design to regenerate the bitstream.

Export hardware with bitstream, launch sdk with existing project(vga_cam).

Run the design under the vga_cam project inside the sdk window.

5.5 Documentation

- ece532_group1_report.pdf
- Final Demo.pptx

6. Tips and tricks

On the technical side, we encountered difficulties with the PMOD CDC1 capacitive button. After reviewing the pinout description table, we initially assumed that the button communicated with the board via the I2C protocol. However, our attempts to recover the signal using the I2C protocol were unsuccessful. We later discovered that the button only requires power to transmit a valid signal to the board when pressed.
Besides, we recommend using “add module” instead of “add IP” when adding your custom RTL into block diagram. We discovered that the added block may not update if you use “add IP” for integration and modify your RTL code later. In contrast, “add module” always updates the block diagram accordingly.

On the non-technical side, we want to emphasize that version control is not easy in such a hardware-based group project. To simplify merging changes in a hardware-based group project, we suggest splitting the group into two teams focused on hardware and software respectively. Members can switch between teams as needed. The hardware team will be responsible for modifying custom IPs or block diagrams, while the software team can concentrate on developing sdk to design GUI and examine instructions. After the hardware team has finished their changes, they can generate a new bitstream and export the hardware for use by the sdk team.

7. Video

Our group demo is uploaded in this link: https://youtu.be/JYAf4uGFNko