Sign Language
Recognition and Translation

ECE 532
Final Report

Emily Miao
Tomasz Piotrowski
Ellen Zhu

I. Overview

Communication is a vital part of everyday life. However, certain disabilities may prevent individuals from the most basic means of communication: listening and talking. When a person suffers from hearing loss, they may learn to communicate through other means, such as lip reading and sign language. But not everyone knows sign language, and thus creating a communication barrier between people with hearing loss, and people without this handicap.

This project aims to close this communication gap by implementing a system which recognizes the 26 letters and 10 numbers of the standard American Sign Language (ASL), and displaying the signed word or sentence on a monitor. This system allows anyone who can read to be able to communicate with persons without hearing who uses ASL.

Current implementations of sign language recognition mostly consists of detecting motions with cameras and processing the information through software, such as the Kinect Sign Language Translator [1]. Although there are hardware implementations on FPGA, such as Development of Sign Signal Translation System Based on Altera's FPGA DE2 Board, they also rely on cameras and video processing to recognize hand signals [2]. The team is to build a simple system operating on FPGA that recognizes and translates ASL to text using sensors.

5 flex sensors (one per finger) as well as an accelerometer were placed on a glove in order to measure hand positions that correlate to letters in sign language. An analog to digital converter was used to read the values from the flex sensors. The accelerometer communicates with the FPGA via SPI. The finger positions representing each letter are stored in memory and the data gathered from the peripherals is compared to the stored values to decide which letter the user is trying to communicate. Once a letter has been decided the FPGA displays the letter on a VGA display. The various blocks used can be seen in the block diagram in Figure 1 below.

The goal of this project was to develop a system that is able to recognize the hand gestures of the 26 letters and 10 numbers of the American Sign Language (ASL), and translate them into text display using an FPGA.

The following IPs were used in order to achieve our goal:

- XADC
    - Xilinx existing IP that reads an analog voltage between 0-1V and outputs a digital value of 9 bits.
- SPI
    - Xilinx existing IP that was used for communicating with the accelerometer.
    - Every SPI device has different requirements for communication so the data fed through the SPI interface was custom, but the IP block itself was not touched.
- TFT
    - Xilinx existing IP to control the VGA display.
    - Used to display the result of our project on the monitor
- Microblaze
    - This Xilinx IP is the brain of the project, drives all the connections and handles communications between IPs
- MIG
    - Xilinx existing IP as the interface of DDR memory.
    - Used to add DDR into our system as one frame of TFT IP is 2M, Bram is not big enough.
- Uartlite
    - Xilinx existing IP to aid with debugging by communicating with SDK via UART and printing statements
- GPIO
    - Xilinx existing IP used to enable the use of switches, LEDs, and the buttons that we added to our glove
    - Only modification was mapping the buttons to external pins
- Slice
    - Xilinx existing IP used to split a port connection to the bits we want
    - Used to split the output of TFT as its output for colors (RGB) are 6 bits each and the board we use is only 4 bits
- ASL
    - Custom IP block created to decode which letter the user is making based on sensor, accelerometer, and button input
    - Utilizes AXI master-slave interface to obtain sensor input data
- ACCEL
    - Modified IP block to handle accelerometer data, from Xilinx user demo
    - Makes use of SPI IP to communicate with accelerometer and receive glove orientation

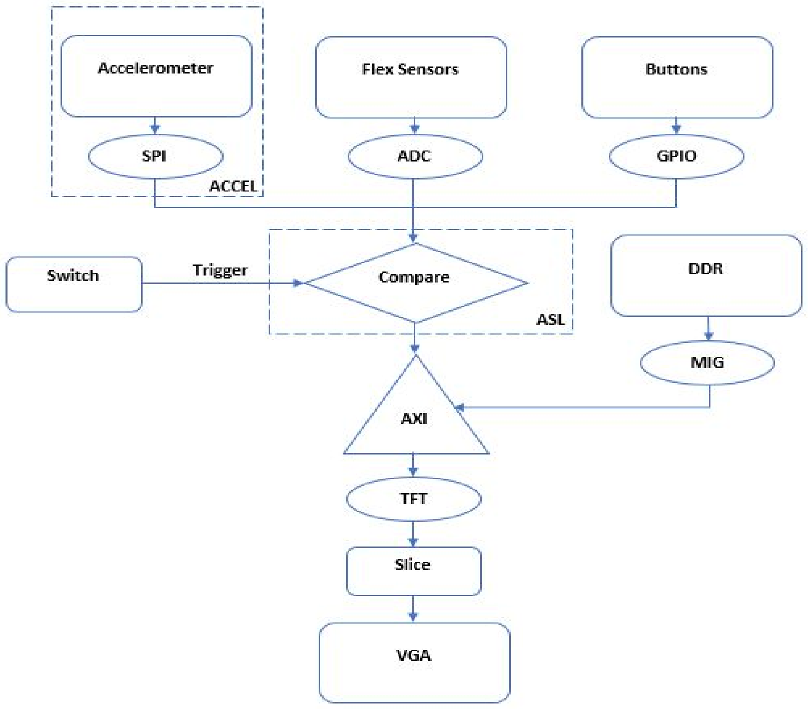

Figure 1: Block Diagram

II. Specifications

1. Functions

- The system should be able to accurately detect static hand gestures and match them to the corresponding letter or number of ASL.
- The system should be able to accurately determine when a static hand gesture is not part of the 26 letters and 10 numbers of ASL.
- The system should be able to recognize the end of a word or the end of a sentence.
- The system should be able to display the gestured word or sentence on the VGA display.

2. Peripherals

- Variable resistances will be used for each finger joint to obtain data on the curvature of each finger.
- An accelerometer will be placed on the back of the hand to detect the orientation of the hand.
- A VGA display will be used to display the letter, word or sentence that has been gestured using ASL
- A camera will be used to capture images of the hand. The data will be used to calculate hand positioning and orientation.

3. Constraints and Limitations

- The system shall operate on a Xilinx Nexys 4 DDR FPGA board.
- The system shall operate with a 100 MHz clock speed.
- The system shall use data from variable resistors, a gyroscope, and a camera to compute the gestured letter or number.
- The system shall use a VGA controller for displaying images on the VGA display.
- The system may not be able to accurately detect dynamic hand gestures, such as letters J and Z of ASL. In such case, contingency plans will be implemented.

III. Outcome

Unfortunately, the final result of our project was not a complete success. We are only able to read the letters “A” “S” and a space and display them to the VGA. The reason for this is the inaccuracy of the sensors used. Perhaps with some more tuning it would be possible to read a few more letters, since in simulation nearly every letter is recognized with consistent simulated sensor data.

To further improve the system, more flex sensors could be added, especially individual ones over each joint, so as to get better information about the position of each joint. We used buttons between each finger to determine whether the fingers were together or apart, but a better method would have been to use less pointy buttons for comfort, or better yet, IR sensors. Using IR sensors would provide a precise distance between fingers and would be far more comfortable for the wearer.

There is not much that could be done differently if we were to start over, as the main restrictions were time and budget. Each sensor was fairly expensive (~20$ x 5) and there was not enough time to properly tune the values, especially since the sensors kept giving inconsistent readings.One thing that could have been done differently would be to have started putting the various parts together sooner to allow more time for testing. However, as mentioned before, time was a limiting factor due to other courses and projects.

If someone were to take over our project, the recommended next steps would include starting by implementing the aforementioned improvements to get more accurate readings. A big next step which would be a whole project in itself would be to implement machine learning to gather the sensor data that corresponds to each letter. With this, the glove could adapt to the wearer and would be more consistent in the readings. Most of sign language uses entire words instead of letters, and these signs heavily rely on motions, so adding a gyroscope and implementing the recognition of these movements would also be a hefty undertaking that would expand our project.

IV. Project Schedule

	Milestone Description	**Expected Completion Date**	**Actual Completion Date**	**Person Responsible**
M1	**Physical Hardware Implementation**			
	1.1 - Custom build gloves with embedded sensors, and be able to receive and access sensor information on FPGA.	Feb 8	Feb 15 Sensors took longer to ship than anticipated	Tom
	1.2 - Setup camera to be able to take pictures and save data in memory.	Feb 8	Camera was removed	Tom
M2	**VGA Display**			
	2.1 - Using only black and white colors, display letters on the VGA using a VGA controller.	Feb 8	Feb 22 Took more time to figure out the TFT IP than expected	Ellen
M3	**Hardware Implementation**			
	3.1 - Process sensor information using software module. Able to calculate hand orientation and finger positions for static letters based on sensor information.	Feb 8	Feb 28 Needed to use sensor data.	Emily
	3.2 - Compute the gestured letter or number based on unique hand orientation and finger position for each letter.	Feb 8	Mar 6	Emily
M4	**Software Implementation**			
	4.1 - Able to receive sensor data and send the data to hardware module through memory mapping.	Feb 29	Mar 14 External time restrictions caused delay	Tom

	Milestone Description	**Expected Completion Date**	**Actual Completion Date**	**Person Responsible**
	4.2 - Able to receive computed data from hardware module, decode the data, and send to VGA controller for display.	Feb 29	Mar 20 External time restrictions caused delay	Tom
M5	**Communication Modules**			
	5.1 - Connect software, hardware, memory, and other controller blocks. Ensure communication between blocks works as expected.	Feb 22	Mar 14 Had issues putting the VGA together with the XADC	Ellen
	5.2 - Use processor to control sensors, hardware and software modules. ie. When to receive data, when to start processing data.	Feb 29	Mar 14 Had issues putting the VGA together with the XADC	Ellen
M6	**Additional Gestures**			
	6.1 - Add additional gestures to hardware implementation for end of letter, end of word, end of sentence, and clear VGA screen.	Mar 7	Mar 27 Delay due to issues putting system together	Emily
	6.2 - Display words and sentences correctly on VGA.	Mar 14	Mar 14	Ellen
M7	**Tracing Moving Motions (Optional)**			
	7.1 - Update hardware implementation to be able to follow and trace hand motion (for letters J and Z).	Mar 14	N/A	Emily

	Milestone Description	**Expected Completion Date**	**Actual Completion Date**	**Person Responsible**
	7.2 - If unable to implement motion tracing, replace letters J and Z with 2 unique static hand gestures instead.	Mar 14	Mar 27 Delay due to issues putting system together	Emily
M8	**Testing**			
	8.1 - Test the implementation on one person. Modify the system parameters as needed.	Mar 14	Apr 3 System was not ready for testing as early as anticipated	Tom
M9	**Calibration (Optional)**			
	9.1 - Using extra calibration parameters, allow the glove to work for virtually anyone based on calibrated values.	Mar 21	N/A	Everyone
M10	**Testing (Optional)**			
	10.1 - Test the implementation on multiple people, taking into account margin of error for people who have never signed before. Modify system parameters as needed.	Mar 21	N/A	Everyone

V. Description of the Blocks

The following IPs were used in our design:

- XADC [3]
  - Xilinx existing IP that reads an analog voltage between 0-1V and outputs a digital value of 9 bits.
  - The XADC is the basic building block that enables analog mixed signal (AMS) functionality which is new to 7 series FPGAs. By combining high quality analog blocks with the flexibility of programmable logic, it is possible to craft customized analog interfaces for a wide range of applications.
  - The XADC includes a dual 12-bit, 1 Mega sample per second (MSPS) ADC and on-chip sensors.
- AXI Quad SPI v3.2 [4]
  - Xilinx existing IP that was used for communicating with the accelerometer.
  - Every SPI device has different requirements for communication so the data fed through the SPI interface was custom, but the IP block itself was not touched.
  - The LogiCORE™ IP AXI Quad Serial Peripheral Interface (SPI) core connects the AXI4 interface to those SPI slave devices that support the Standard, Dual, or Quad SPI protocol instruction set. This core provides a serial interface to SPI slave devices. The Dual/Quad SPI is an enhancement to the standard SPI protocol and provides a simple method for data exchange between a master and a slave.
- TFT [5]
  - Xilinx existing IP to control the VGA display.
  - One frame is 2M and one pixel is 4 bytes with the format of "0bXXXX XXXX RRRR XXXX GGGG XXXX BBBB XXXX"
  - The frame data need to be stored at the frame address of this IP
  - Used to display the result of our project on the monitor
- Microblaze [6]
  - This Xilinx IP is the brain of the project, drives all the connections and handles communications between IPs.
  - Reconfigured to enable 32-bit multiplier, integer divider, floating point unit, barrel shifter, branch target cache, and AXI data interface. Local instruction and data memory is configured to 128KB.
  - Microblaze has 32 general purpose registers, and 14 special purpose registers including: program counter, machine status register, exception address register, exception status register, branch target register, floating point status register, exception data register, process identifier register, zone protection register, translation look-aside buffer registers, and process version register.
  - Data is read from flex sensors, the accelerometer, and buttons, and written to the custom IP by memory mapping the slave registers of all the IP's.
- MIG [7]
  - Xilinx existing IP as the interface of DDR memory.
  - Used to add DDR into our system as one frame of TFT IP is 2M, Bram is not big enough.

- Uartlite [8]
    - Xilinx existing IP to aid with debugging by communicating with the console via UART for printing statements
    - Performs parallel-to-serial conversion on characters received through the AXI4-Lite interface and serial-to-parallel conversion on characters received from a serial peripheral.
    - Transmits and receives up to 8-bit characters.
- GPIO [9]
    - Xilinx existing IP used to enable the use of input and output from external pins.
    - The GPIO is used to map switches and LED's, which was used for debugging.
    - The GPIO is also used for buttons that we added to the glove, which is used for recognizing letters.
- Slice [10]
    - Xilinx existing IP used to split a port connection to the bits we want
    - Used to split the output of TFT as its output for colors (RGB) are 6 bits each and the board we use is only 4 bits
- ASL
    - Custom IP block created to decode which letter the user is making based on sensor, accelerometer, and button input.
    - Utilizes AXI master-slave interface to obtain sensor, accelerometer, and buttons input data.
    - Customized each letter by setting a range of valid sensor values of each sensor for that letter.
    - Further increase accuracy by using accelerometer data to decipher the position and orientation of the hand, and button data to decipher if the fingers are touching each other.
- ACCEL[11]
    - Modified IP block to handle accelerometer data, from Xilinx user demo
    - Makes use of SPI IP to communicate with accelerometer and receive glove orientation
    - To read from the on-board data registers, the Chip Select line must first be pulled low and then send a command byte to read from the data registers (0x0B). The desired address byte must be sent next, and then the desired byte is received with the MSB first on the falling clock edge. Because the address pointer auto-increments to the next address byte, it is possible to read multiple bytes consecutively by continuing to pulse the Serial Clock line.

VI. Description of the Design Tree

The following is the directory structure of our design as posted on GitHub.

README.t
- Includes a description of the project and how to use the project.

doc
- Final Report.pdf
- Final Presentation.ppt

src
- ASL_Project
    - Includes the block diagram that connects the VGA, and all IP's to a microblaze processor.
    - Includes the software code, ASL_main.c, that controls and enables communication between all the IP's.
- ASL_IP
    - Includes the custom IP to compute the letter based on sensor data
- ADC
    - Includes the modified XADC IP to obtain the flex sensor data
- ACCEL
    - Includes the modified IP with SPI to obtain the accelerometer data
- HEX
    - Includes the modified IP to use the HEX display for debugging
- board_files
    - Includes the board files necessary to use the DDR board

hardware
- glove.jpg
    - Picture of the glove with flex sensors, accelerometer, and buttons

VII. Tips and Tricks

1. Integrate your various parts as soon as possible. There are many unsuspecting issues that will arise, so it is best to iron out the bugs of merging various parts of your project.
2. Parallelize as much as possible, and in such a way that team members are not waiting on each other to do their parts.
3. If you have any parts you need to order online, do so as soon as you know what you need. Shipping is an unfortunate way to waste valuable time.
4. Keep constant communication with your partners, and ensure any changes are made known right away.
5. Communicate what data your various parts are expecting and will be outputting so other group members can plan for it.
6. Spend some time to analyze your design for potential failures, and have a backup plan in place if it comes to it.
7. Start with small pieces and build up, slowly combining. It is easier and quicker to debug many small parts than one big one.
8. Make use of TA and Professor's knowledge. Ask questions on Piazza early on if you get stuck so have the most amount of time to try a solution.

References

[1] "Kinect Sign Language Translator expands communication possibilities", Microsoft Research. [Online] 2013, URL:
http://research.microsoft.com/en-us/collaboration/stories/kinect-sign-language-translator.aspx
Accessed: 2016 January

[2] "Development of Sign Signal Translation System Based on Altera's FPGA DE2 Board", TechRepublic. [Online] 2011, URL:
http://www.techrepublic.com/resource-library/whitepapers/development-of-sign-signal-translation-system-based-on-altera-s-fpga-de2-board/
Accessed: 2016 January

[3] Xilinx. (2015 May). "7 Series XADC" [Online] Available:
http://www.xilinx.com/support/documentation/user_guides/ug480_7Series_XADC.pdf
Accessed: 2016 January

[4] Xilinx. (2015 Nov.). "AXI Quad SPI v3.2" [Online] Available:
http://www.xilinx.com/support/documentation/ip_documentation/axi_quad_spi/v3_2/pg153-axi-quad-spi.pdf
Accessed: 2016 January
[5] Xilinx. (2015 November). "AXI Thin Film Transistor Controller v2.0" [Online] Available:
http://www.xilinx.com/support/documentation/ip_documentation/axi_tft/v2_0/pg095-axi-tft.pdf
Accessed: 2016 March

[6] Xilinx. (2015 November). "MicroBlaze Processor Reference Guide" [Online] Avalable:
http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/mb_ref_guide.pdf
Accessed: 2016 April

[7] Xilinx. (2010 September). "Memory Interface Solutions User Guide" [Online] Available:
http://www.xilinx.com/support/documentation/ip_documentation/ug086.pdf
Accessed: 2016 January

[8] Xilinx. (2015 November). "AXI UART Lite v2.0 LogiCore IP Product Guide" [Online] Avalable:
http://www.xilinx.com/support/documentation/ip_documentation/axi_uartlite/v2_0/pg142-axi-uartlite.pdf Accessed: 2016 April

[9] Xilinx. (2015 November). “AXI GPIO v2.0 LogiCore IP Product Guide” [Online] Avalable: http://www.xilinx.com/support/documentation/ip_documentation/axi_gpio/v2_0/pg144-axi-gpio.pdf Accessed: 2016 April

[10] Xilinx. (2014 November). “7 Series DSP48E1 Slice User Guide” [Online] Available: http://www.xilinx.com/support/documentation/user_guides/ug479_7Series_DSP48E1.pdf Accessed: 2016 April

[11] Xilinx. (2015 Feb.). “PmodACL2™ Reference Manual” [Online] Available: https://reference.digilentinc.com/_media/pmod:pmod:pmodACL2_rm.pdf Accessed: 2016 January