Audio and Video Gaming Control

Developed By:

Caleb Larner, Evan Madej, Robert Napier, Theo Rabban, Mike Shen, Robert Towne

EECS 452 Digital Signal Processing Lab

Project Overview

With the recent popularity in motion controlled video gaming, such as Microsoft's Kinect, Nintendo's Wii, and the PlayStation Move, we decided to implement our own controller for a personal computer.

Our project is a combination of video and audio processing used to control a video game with limited commands. For this project, we chose to demonstrate the ability of the controller with the game *Tetris*.

Camera inputs discern motion commands by detecting the red glove, while a microphone is used to recognize audio commands from a library of speech commands.

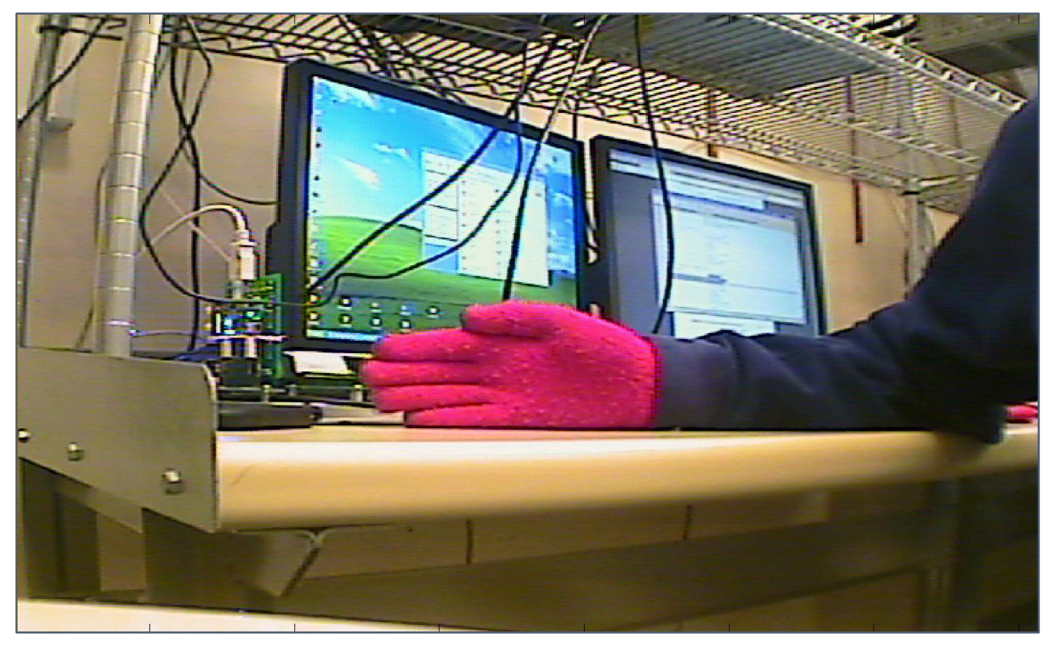
A small microprocessor, the Texas Instruments C5515, was used to process the audio signals, while an Altera DE2-70 FPGA was used to process the video inputs as well as interface with the computer over a PS/2 keyboard interface.

Video Processing

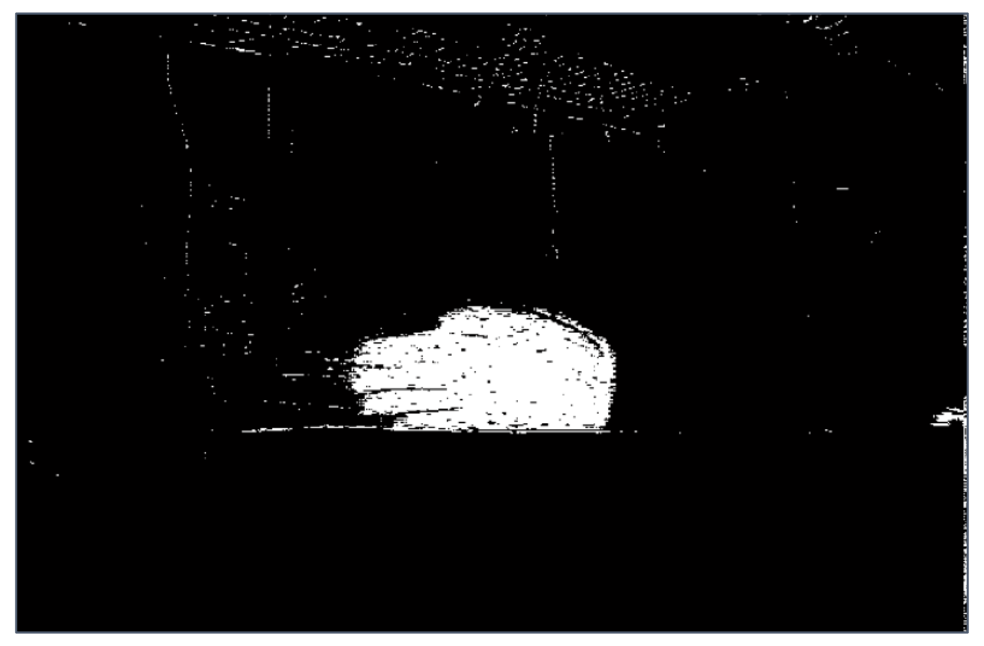
On the video end, we use a small camera to capture red glove movements on screen. The camera outputs the pixel values in a Red-Green-Blue (RGB) format, which we then convert to a Hue-Saturation-Value (HSV) format on the DE2-70 FPGA. HSV is less sensitive to lighting conditions and gives us a more accurate pixel color representation. Using these values, we set a threshold to detect red pixels and filter the HSV image into a binary image, where red is 1 (on) and everything else is 0 (off). By locating regions on the screen where there are significant portions of red (binary 1's), we can determine where the glove is. When the glove is in a particular region, we send a signal over the PS/2 keyboard interface telling the computer to take the respective action in the video game.

Camera → RGB → HSV → Thresholding → Binary Image → Region Detection → PS/2 → Computer

Original Image



Binary Image

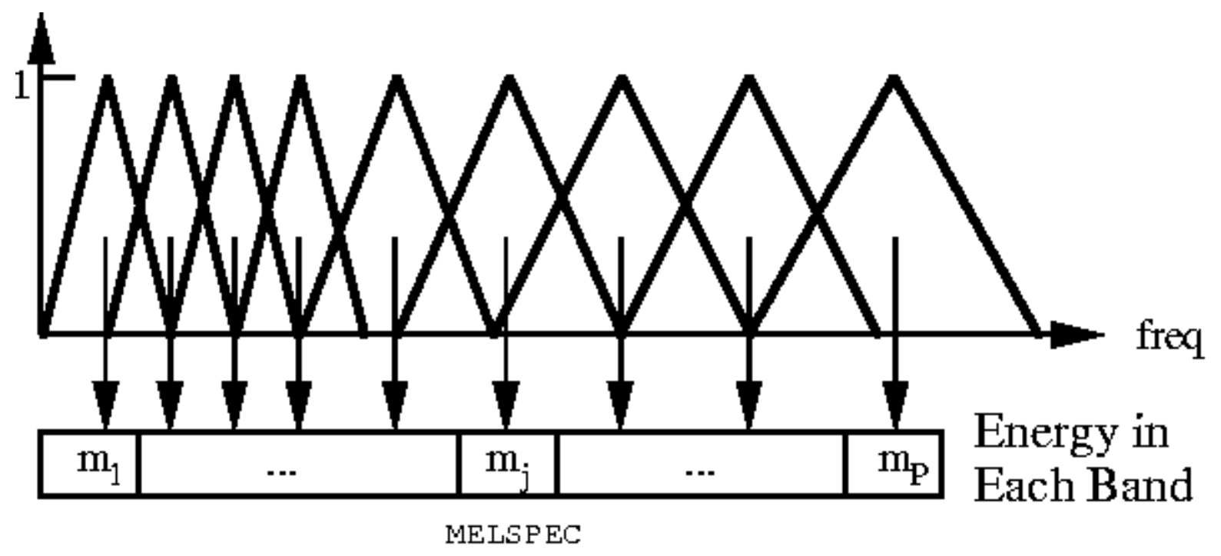


Audio Processing

On the audio end, we use a headset with a built-in microphone to capture the various commands for the video game. The audio is first sampled and then segmented into frames. Each frame is run through a Mel Filter and stored in a library on the C5515 chip. Once every command is stored, we capture audio commands from the user and compare those commands with the stored library commands using a Dynamic Time Warping Algorithm (DTW). DTW finds the smallest distance between two audio frames and chooses which word was sent by the user. Once a word is chosen, it is sent out as a command over the SPI interface to the FPGA. The FPGA then sends the corresponding command through the PS/2 keyboard interface telling the computer to take the respective action in the video game.

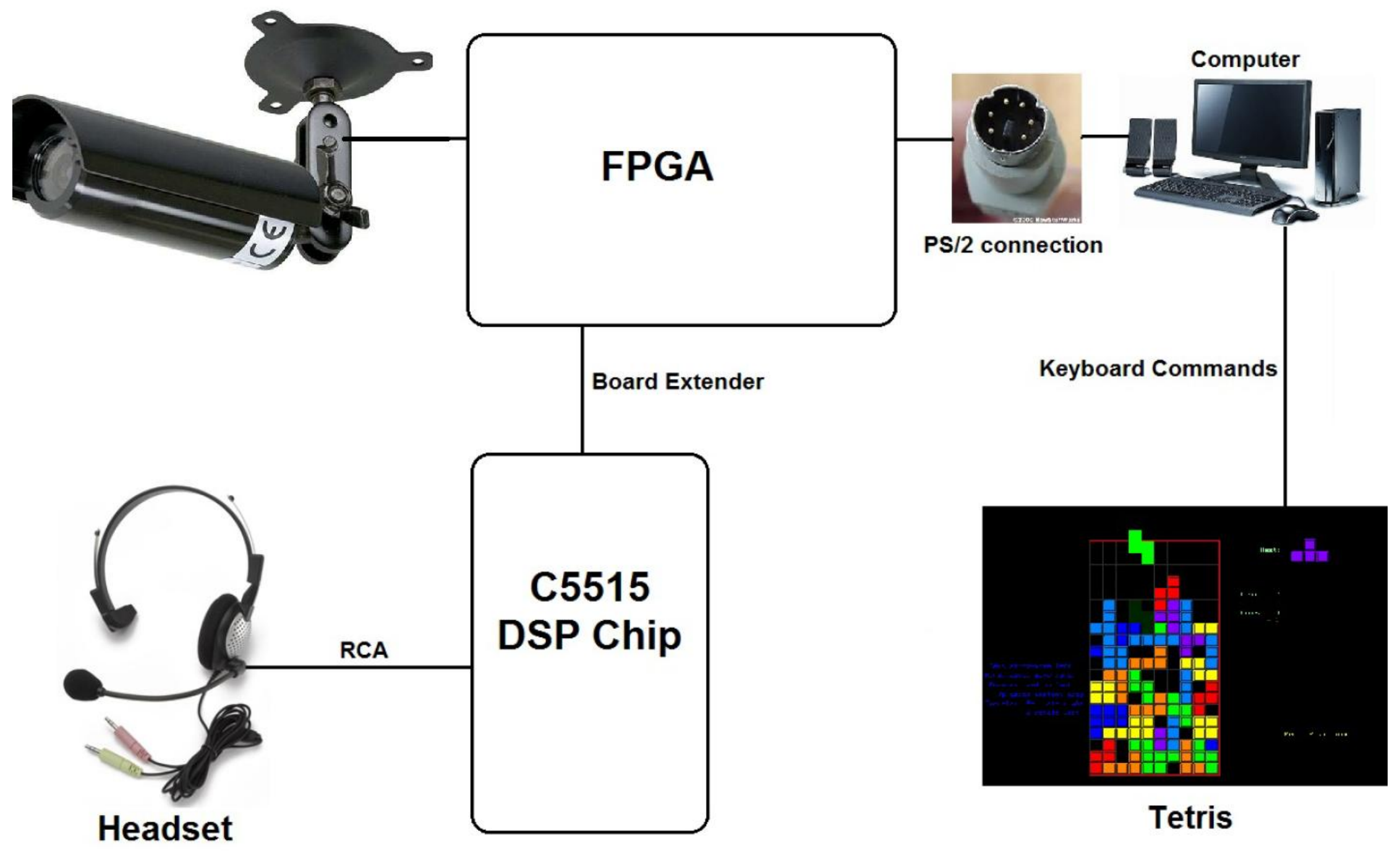
Headset →Thresholding → Sampling → Framing → FFT → Mel Filtering →DTW → SPI →FPGA

Mel Filter Bank



http://www.ee.uwa.edu.au/

System Diagram



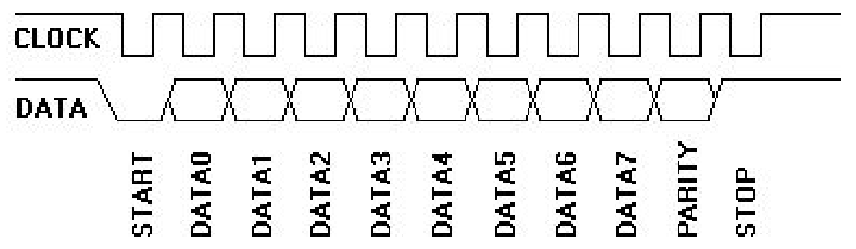
The main components are:

- NTSC Camera
- Headset with Microphone
- DE2-70 FPGA
- C5515 DSP Microprocessor
- PS/2 Connector
- Personal Computer

Communication

PS/2 Protocol

Keyboard commands were transmitted through a PS/2 interface between the DE2-70 FPGA and the PC. Below is a chart detailing the device (DE2-70) to host (PC) serial communication for an 8 bit data transmission:

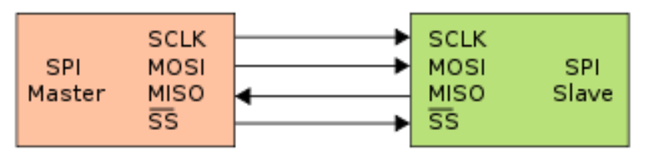


http://www.computer-engineering.org/ps2protocol

The Host (PC) reads the data signal on the falling edge of the clock. The device (DE2-70) sends the clock signal when data needs to be transferred, but ultimately the Host (PC) has control of the wires.

SPI Protocol

To transmit data between the TI C5515 Microprocessor and DE2-70 FPGA, a Serial Peripheral Interface (SPI) bus was used. SPI is a four wire bus that operates in a "full duplex" (2 way data transmission) mode. In our current setup, the C5515 is the "master" device, while the FPGA is the "slave". Below is a simplified picture of the SPI bus:



http://seemanta.net

Future Work

- Refine movement sensitivity and control
- Add more voice commands
- Reconfigure for multiple games
- Configure for mouse cursor control

Acknowledgements

A special thanks to:

Altera
Intel
Texas Instruments

Professor Kurt Metzger
Professor Mark Brehob
GSI Chao Yuan