Real-time Hand Tracking with Neural Nets on an FPGA

Brian Kubisiak <bkubisia@caltech.edu>
Quinn Osha <qosha@caltech.edu>

1 Functional Specification

For our project, we will be designing and implementing a system for tracking a hand in real time from a camera input. The system will be implemented using a Digilent Nexys 4 FPGA development board. It will take an input from a camera connected to the USB port, analyze the data on the Artix 7 FPGA, and output each from over the VGA port with a cursor overlay. The cursor overlay will indicate where in the frame the hand is located. While the neural net is computing the hand position, the frame will temporarily be stored in the board's memory—the FPGA is too small to process an entire frame at once.

1.1 Operation

The FPGA design will consist of three distinct layers:

Input Layer For communicating with the USB camera, writing the frame to memory, and converting the frame into a format that the neural net can use.

Neural Network For analyzing the camera data to determine where the hand is in the frame.

Output Layer For reading the frame back from memory, overlaying a cursor on the hand position, and generating control signals to output the frame through a VGA port.

A top-level block diagram showing the layers and their connections is shown in figure (1). The Rx and Tx inputs come from the USB-UART bridge. The Data, Address, Wr, and Rd signals go to the memory on the Digilent board. Note that the Data and Address buses must be muxed before connecting to the pins. The HSYNC, VSYNC, and Data outputs will go to the VGA port on the board.

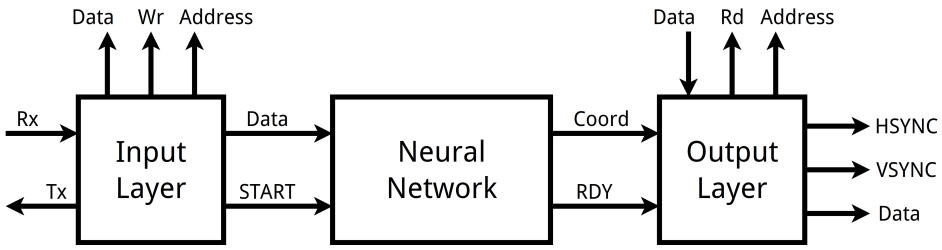

Figure 1: Top-level block diagram for the system.

1.1.1 Input Layer

The input layer is responsible for controlling the USB port, subsampling the frame data, shifting the data into the neural net, and writing each frame into memory. This process is shown in figure (2).

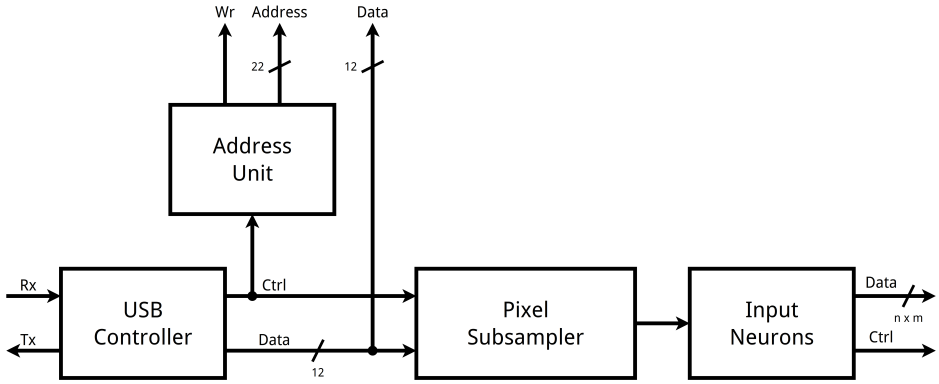

Figure 2: Block diagram for the input layer.

The USB controller interfaces with the FT2232 USB-UART bridge on the Nexys 4 board through the Tx and Rx signals. The Tx signal is needed in order to give commands to the camera over the USB port. The Rx line is responsible for reading in the data.

Once the data is read in through the UART, it is parsed by a finite state machine. This FSM will read the USB packets from the camera to extract the pixel and frame data. It will condense each 24-bit pixel into 12 bits (the size needed for the VGA) and output this data along with control signals for determining the frame timing.

The address unit will take in the control signals from the USB controller and use them to generate the proper addresses for the data. In order to improve performance, the data will be written to memory in a format suitable for VGA controller in the output layer. This address unit will take in the frame and line data to ensure that the pixels are written to the proper addresses.

Figure out the memory layout that we need.

Due to size limitations in the FPGA, the neural net cannot process the entire frame. Pixels will be averaged togther in order to compress the data without losing too much important information.

The input neurons will take the subsampled data on a parallel bus and shift the data through the input layer until the entire subsampled frame is loaded into the neurons. These neurons will then output all $n \times m$ bits of frame data to the neural net in parallel for processing. Control signals will tell the neural net to start processing the frame.

Figure out how we want to subsample.

1.1.2 Neural Network

The neural network will take in the frame data and process it to determine the coordinates of the hand in the frame.

Do more research on this part.

1.1.3 Output Layer

Read frame from memory, overlay the cursor.

Draw block diagram, write description.

1.2 Algorithms

We will be using a convolutional neural network to identify where the hand is located in each frame. The parameters for the neural network will be learned in advance on the computer, then hard-coded into the FPGA neural net. Time-permitting, we will also implement unsupervised learning on the FPGA in order to allow the system to learn without hard-coded parameters.

1.3 Inputs

The input to this system will be video input through a USB controller. Each frame that is input to the USB controller will be stored in an internal frame buffer before being processed by the neural net.

1.4 Outputs

The output from this system will be through a VGA controller. The VGA output will be taken from an output frame buffer. The output frame will be generated from the input frame with a cursor overlaid on the position of the hand (identified by the neural net).

2 Schedule

Date	Tasks
4/7–4/13	Research current implementations of neural nets on FPGAs. Decide on video input and output devices (size and pixel depth). Write code for USB controller.
4/14–4/20	Write code for input data buffer and input data handling. Design—in a high-level-language—the desired neural net.
4/21–4/27	Test and configure the neural net on the computer. Write code for output data buffer and handling.
4/28–5/4	Begin coding the neural net on the FPGA. Write code for output display handling using VGA.
5/5–5/11	Finish coding the neural net.
5/12–5/18	Write code for the cursor video output. That is, given x and y coordinates, print the cursor at that point.
5/19–5/25	Write the top-level entity for the system, combining all the blocks. Train (unsupervised) the neural net.
5/26–6/1	Finish training the neural net. Fix any other bugs/issues.
6/2–6/5	Finish documentation and final touches.

3 Demonstration

This system will be demonstrated by attaching a video camera to the USB input and a monitor to the VGA output. Then, the system should detect a hand in the input frame and output the frame to the monitor with a cursor overlaid on the position of the hand.