6.111 Final Project Checklist
A Hand Controlled Digital Audio Synthesizer

Behram Mistree
Alexander Sanchez

- **Deliverables**
  - Neural Network
    - *Oscillators* **Behram**
      - Creates one of three signals: sine wave, saw tooth wave, cosine wave
      - Will demonstrate using the logic analyzer to show the waveforms being generated
    - *Weight generator* **Alex**
      - Calculates the new value of the weights based on the propagated error, propagates its error, and multiplies the inputs by the current weights
      - Will demonstrate using ModelSim
    - *Input Function* **Alex**
      - Computes the sum of all the weighted inputs
      - Will demonstrate using ModelSim
    - *Output Function* **Behram**
      - Evaluates the Sigmoid function and its derivative at the value of the output of the Input Function
      - Will demonstrate using ModelSim
    - *Neuron Bus* **Alex**
      - Controls the data flow between levels of neurons so that the number of wire connections between levels can be reduced
      - Will demonstrate using ModelSim
    - *FSM for controlling memory access* **Alex**
      - Controls which part of the ADSR envelope the network is computing and tells the weight generator which weights in memory to use.
      - Will demonstrate using ModelSim
    - *Sample FSM* **Alex**
      - Controls which of the stored ADSR samples the network is currently using to learn from and when the network compares its output to the stored samples
      - Will demonstrate using ModelSim
    - *Error Calculator* **Alex**
      - Compares the output of the network to a stored audio sample and calculates the squared errors

        - Will demonstrate using ModelSim
    - The entire Neural Network will learn how to create audio signals that sound like they were produced by real instruments. It will achieve this by taking in some arbitrary signal (the output of the Oscillator), applying weights to the inputs at each level of the network, comparing the output of the network to a desired signal, and then adjusting the weights. This process will continue until a satisfactory signal can consistently be produced by the network.
        - This functionality will be demonstrated by showing the network generating signals that it previously learned how to create as well as beginning to learn to create a new signal.
- Subtractive Synthesis
    - *Oscillators* **Behram**
        - Creates one of three signals: sine wave, saw tooth wave, cosine wave
        - Will demonstrate using the logic analyzer and ModelSim to show the waveforms being generated
    - *Pulse Width Modulation* **Behram**
        - Applies Pulse Width Modulation to an incoming audio signal
        - Will demonstrate using logic analyzer and ModelSim.
    - *ADSR FSM for accessing stores envelopes in memory* **Behram**
        - Controls which of the four sets of values (Attack, Decay, Sustain, and Release) from memory that the ADSR Envelope Applier uses.
        - Will demonstrate that the FSM is working properly using ModelSim
    - *ADSR Envelope Applier* **Behram**
        - Will apply an ADSR envelope to an incoming audio signal
        - Will demonstrate using the logic analyzer and ModelSim
    - *Filter* **Behram**
        - Applies a low pass filter to an audio signal.
        - Will demonstrate using the logic analyzer and ModelSim
- Hand Detection Module
    - *NTSC Decode* **Behram**
        - Use Javier's code to get YCrCb
    - *get_hcount_vcount* **Behram**
        - Modifies Javier's code to get the current pixel displaying from camera's position on the screen.
    - *hand_detect* **Behram**
        - hand_detect collects the camera data. It scans for LED's and returns the number of LED's detected.

        - Will test hand_detect by seeing if LED's motion is correctly output to monitor through monitor module.
    - *test_acceptable* **Behram**
        - test_acceptable checks if a pixel returned from the camera is suitably colored.
        - Will demonstrate test_acceptable working in ModelSim.
    - *within_distance* **Behram**
        - within_distance checks if the x and y positions of a pixel on the computer screen are suitably different from the x and y positions of previously stored pixels.
        - Will demonstrate within_distance working in ModelSim.
    - *YCrCb2RGB* **Behram**
        - Using Xilinx created module to convert to the RGB color space from YCrCb.
- Other
    - *Volume* **Behram**
        - Takes in the x and y positions of all detected LED's and returns the volume that corresponds to the position of a hand with a volume LED.
        - Will demonstrate with ModelSim.
    - *Note* **Behram**
        - Takes in the x and y positions of all detected LED's and returns the note values that correspond to each hand with a note LED.
        - Will demonstrate with ModelSim.
    - *Keyboard* **Behram**
        - Will modify the 6.111 keyboard module online for our purposes.
    - *Audio* **Behram**
        - Will implement 6.111 audio module from Lab 4.

If Time Permits

- Synthesizer
    - Looper **Alex**
        - Will store the output signal into memory and play it back while the synthesizer is still generating signals.
        - Will demonstrate by recording the output of the system and then looping it back while creating more signals with the system
    - *Voice In* **Alex**
        - Will use voice input as another instrument.
        - Will demonstrate by speaking into the microphone and hearing the voice in the output
    - *Vibrato* **Alex**
        - Will vary the pitch of a note

        - Will demonstrate by playing the same note with and without vibrato applied.
- LED Control Tower **Behram**
    - The LED control tower will display particular sequences of multi-colored LED's. The duration and sequences of these LED's will be read by the camera and played as music on the output.