Name	SID #

EEM16/CSM51A (Fall 2017)

Logic Design of Digital Systems

Prof. Ankur Mehta : mehtank@ucla.edu

Design lab 3 | **Assigned Wednesday Nov. 22, 2017**
due 4pm Friday Dec. 8, 2017

Instructions

This lab is to be done individually. You may consult with others to share thoughts and ideas, but all of your submitted work must be yours alone. Be sure to indicate with whom you've collaborated and in what manner.

You may use any tools or refer to published papers, books, or course notes. You're allowed to make use of online tools such as Logisim, WolframAlpha, etc., provided you properly cite them in the provided file.

Submission procedure

Upload your Verilog files, as named in the skeleton code provided, to the Gradescope autograder. Be sure to include the Verilog modules, testbenches, and coversheet comments.

0 Morse decoding using a neural network

0.1 Overview

Our design labs will revolve around creating a system to decode an input Morse code bitstream into an alphabetic character display. The key element in this system will be a feedforward neural network with 4 input nodes representing the Morse pulse widths, 4 hidden nodes, and 26 output nodes representing the likelihoods of each character. A schematic of the system is shown in Fig. 1.

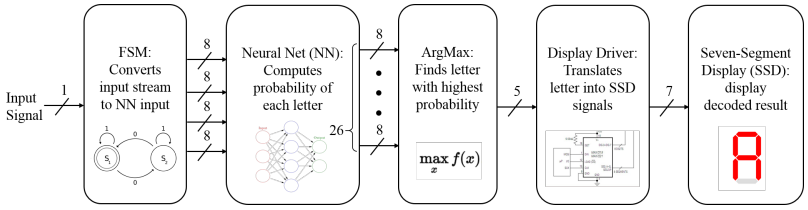

Figure 1: Block diagram of Morse decoder

0.2 Input

The input will be a 1-bit time series indicating the Morse code pulses as shown in Fig. 2. This time series will be parsed into a set of pulse widths (durations) using a finite state machine: each '1' pulse is separated by a short '0' gap, while sets of pulses defining a single character are separated by a long '0' gap.

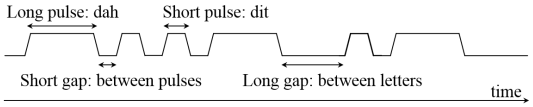

Figure 2: Input pulse timing diagram for the morse code sequence: — • • — • — = PA

0.3 Neural net

A feed-forward neural net serves as a general purpose input-output function; it contains a number of weights and biases that determine the form of this function. Training these weights is out of the scope of this project, so the pre-trained weights will be supplied in a memory. The resulting network maps the (4) input pulse widths to the (26) likelihoods that those pulses code each possible letter of the alphabet.

The neural net is built of 3 layers of neurons, each fully connected to the layers before and after as shown in Fig. 3.

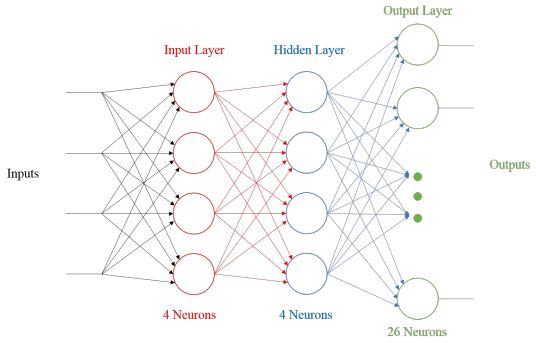

Figure 3: Connectivity diagram for the neural network

0.4 Neuron

Each neuron stores an 8-bit unsigned integer value. The input neurons will be assigned their values based on the measured pulse widths. The hidden layer and output layer neurons each compute their value as a function of the neuron values in the previous layer. This computation consists of three steps as shown in Fig. 4:

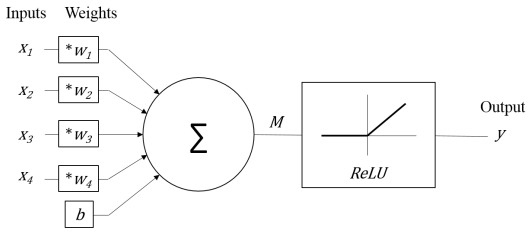

Figure 4: Mathematical operations within a neuron

- each incoming neuron value is multiplied by a pre-trained 8-bit signed integer weight to get a 16-bit signed integer weighted value,
- all weighted values are summed together with a pre-trained signed integer bias term, and
- the sum is rectified, thresholded, and scaled down to get the resulting 8-bit unsigned integer value.

0.5 Output

The values on the 26 output neurons are representative of the likelihood that the input Morse code sequence maps to each respective letter of the alphabet; the neuron with the highest value indicates the decoded letter. To get a single 5-bit value for the decoded letter (0=A, 1=B, . . . , 25=Z), the output values must be compared to determine the index of the maximum value.

0.6 Display

The decoded letter can be presented (with a few modifications) on a 7-segment LED display as shown in Fig. 5. Each segment of the display must then be driven appropriately as a function of the 5-bit letter index.

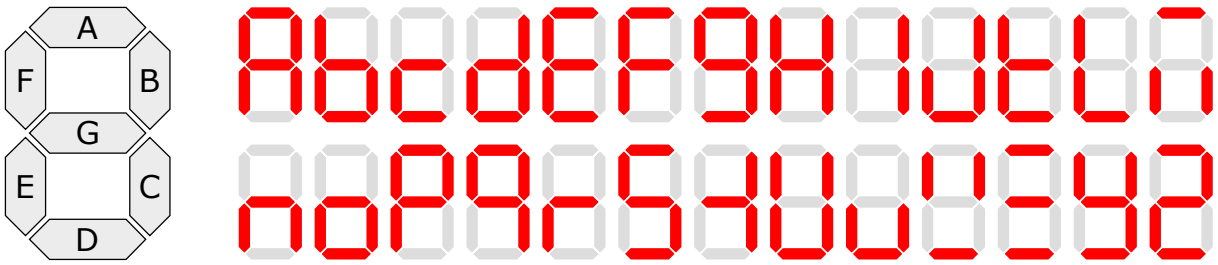

Figure 5: The alphabet as realized on a 7-segment LED display

3 Lab assignment 3

3.1 Pipelined neuron

Each neuron will take in the following inputs:

- The system clock,
- Four 8-bit signed integer weights w_1, w_2, w_3, w_4,
- One 8-bit signed integer bias b,
- Four 8-bit unsigned integer inputs x_1, x_2, x_3, x_4,
- A one bit flag (indicating `new` or `reset`) that pulses high for one clock cycle when a new set of inputs x is ready for processing;

and generate the following outputs:

- One 8-bit unsigned integer output y,
- A one bit flag (`ready`) that is held high so long as the output value has the correct value for the inputs.

The value is determined by the algebraic equation $y = \max(0, \frac{1}{4}(w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b))$.

3.1(a). Build the Rectified Linear Unit (ReLU) module using combinational logic that takes in a signed integer `in` and outputs the unsigned integer `out` $= \max(0, \texttt{in}/4) = \texttt{in}/4$ if `in` > 0, and 0 otherwise.

3.1(b). Create a five-input 16-bit pipelined ripple carry adder using the five-input 4-bit ripple carry adders from lab 1 and any necessary d-registers.

3.1(c). Using the sequential multiplier module from lab 2, the pipelined ripple-carry adder from above, and the ReLU from above, implement the neuron module according to the rules for single-clock sequential logic. Ensure that the outputs are directly generated by a d-register as specified by our pipelining rules.

3.2 Pipelined neural net

Our neural net will consist of an input layer of 4 neurons and an output layer of 26 neurons (a hidden layer won't be necessary). The framework for the neural net will be provided, instantiating each of the neurons with the appropriate values. It will take in the following inputs:

- The system clock,
- Four 8-bit unsigned integer inputs x_1, x_2, x_3, x_4,
- A one bit flag (indicating `new` or `reset`) that pulses high for one clock cycle when a new set of inputs x is ready for processing;

and generate the following outputs:

- 26 8-bit unsigned integer outputs $y_A, \ldots, y_Z$,
- A one bit flag (`ready`) that gets set for one clock cycle when those output values are appropriately generated.

3.2(a). Build an inter-layer module connecting the outputs of the input layer of neurons to the inputs of the output layer of neurons. This should hold the outputs of the input neurons until they have all been generated, then set the `new` signal for the output neurons high for one clock cycle.

3.2(b). We will set the shortest duration of our input Morse code bitstream ("dit" length) to be 2 clock cycles, so the shortest time between successive inputs to our neural net will be 8 clock cycles (i.e. a stream of E characters: a dit followed by a long gap). Pipeline the neural net to ensure a throughput of at least one per 8 clock cycles. That is, the system should still generate correct results even when a new set of inputs arrives every 8 clock cycles.

3.3 Morse decoder

We now have all the modules necessary to implement our complete Morse decoder!

3.3(a). Connect the input de-serializer from lab 2 to the input of the above neural net. Connect output of the neural net to the maximum index module of lab 1, followed by the display driver of lab 1. Add any necessary registers / pipeline stages to ensure that the throughput of the system remains at least one per 8 clock cycles.

3.3(b). Add an output flag indicating when the display driver output is valid.

3.H EE89 extra assignment

3.H(a). Assemble all the modules from this and all previous labs into an end-to-end morse decoder, taking an input morse code stream tapped out on a push-button on the board and displaying the output letters on the 7-segment displays.

3.H(b). (Optional) Add a shift register to the output to display the last 4 decoded letters.