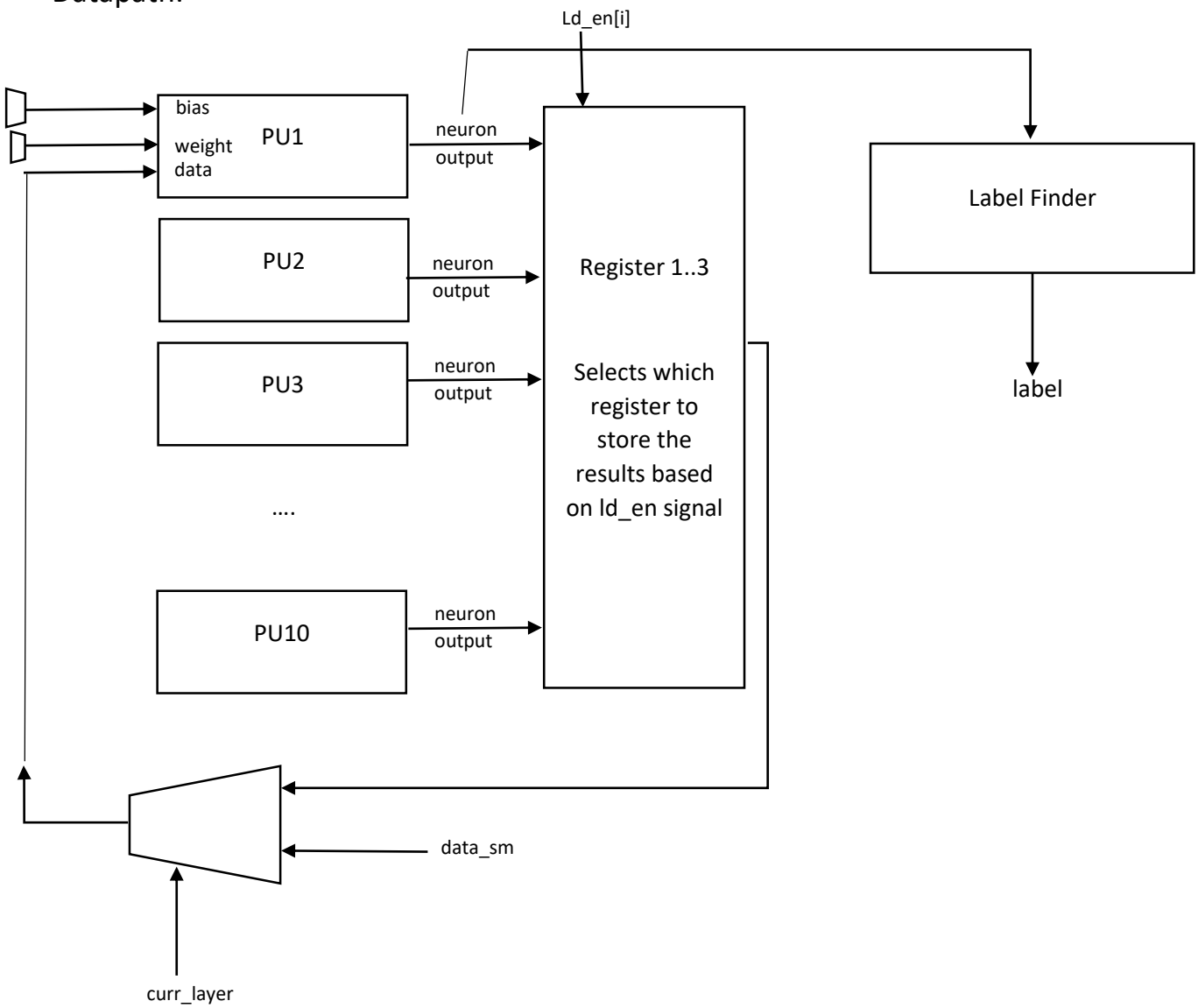
Hardware Implementation of a Multi-Layer Perceptron Neural Network

Kianoush Arshi-810198438

Emad Emami-810198???

Datapath:



LabelFinder:

Selects the label with the highest probability.

Mux4to1s:

- Weight_mux: Multiplexer for selecting the appropriate weights based on current layer(curr_layer).
- Bias_mux: Multiplexer for selecting the appropriate biases based on current layer(curr_layer).

PU:

Calculates the probability of each label based on given data.

Includes:

- MAC: Multiplies input data by respective weight and sums up the results.
- Bias Sum: Adds biases to the mac results.
- Shift Sum: Shifts the bias sum results for the ReLU function.
- ReLU Function: Activation function applied to shifted sums. Sets negative values to zero and keeps the positive values.
- Saturation: Saturates the relu result so that the values don’t exceed given range.(-127,+127)

Registers:

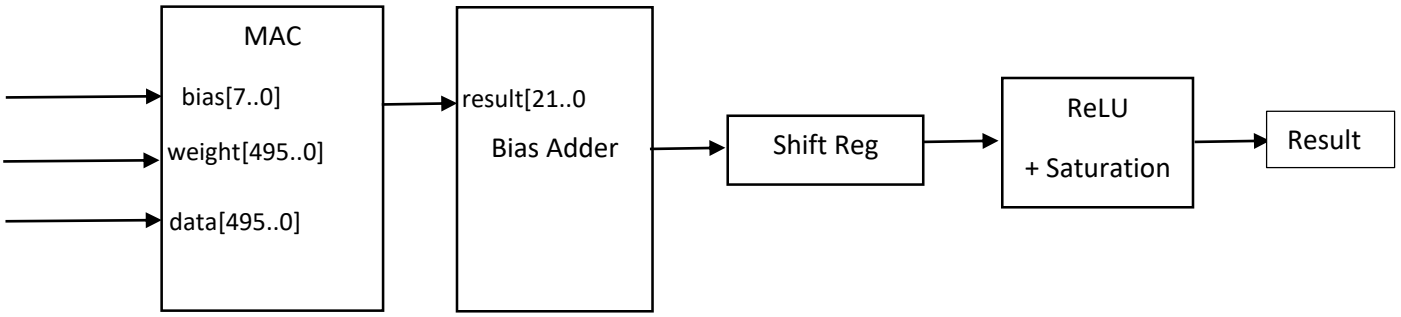
Store the results of the previous layer.

PU:

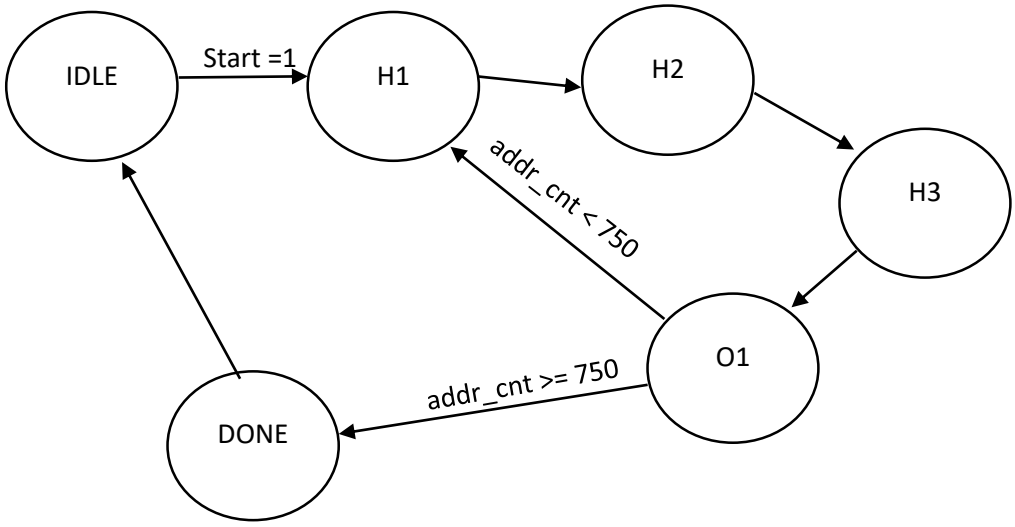
The PU module consists of a MAC unit, a bias adder, a shift register and ReLU function with saturation.

Initially, the input is multiplied by weight input using the MAC unit. Afterwards, the results are added with the input bias values. Note that the bias values are multiplied by 127 since we will be shifting the resulting sum in the next stage.

Finally, after the shift, the results are passed through a ReLU function and the negative values are set to zero and are saturated to have the maximum value of 127.



Controller:



In states H1 to H3, neurons will calculate in groups of ten. The results will be stored in a register.

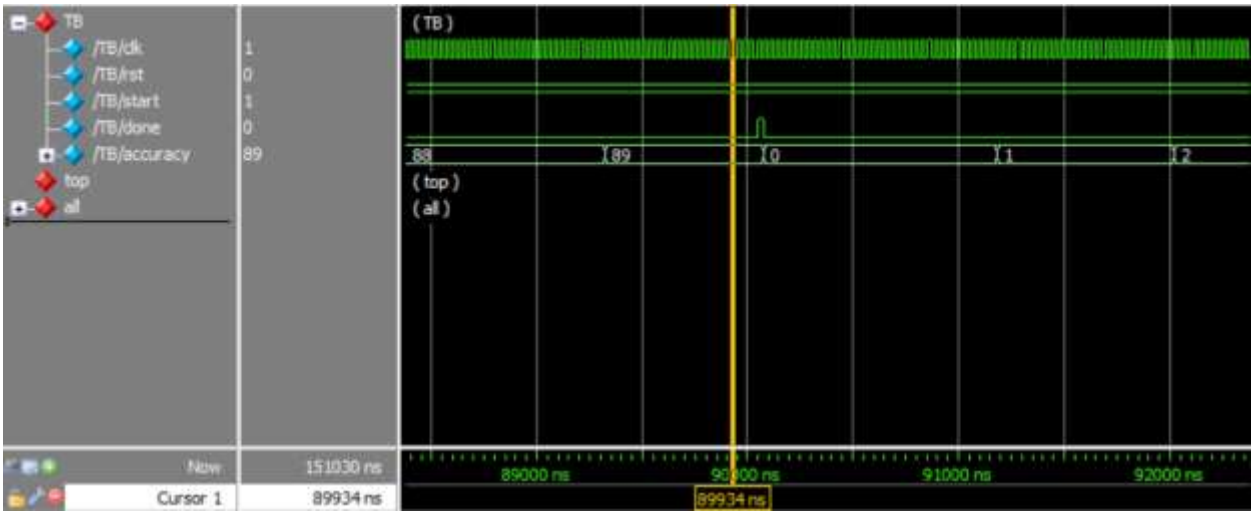
In state O1, the hidden layer results are passed to the ten neurons of the output and the probabilities of labels are calculated. Furthermore, the predicted label will be compared with the actual label and if they were identical, the clk_en signal of the correct count counter will be set to 1 for 1 clock cycle.

In state DONE, when all 750 test cases are handled, the done signal will be switched and the final accuracy will be reported.

Signals:

IDLE: Init = 1	H1: curr_layer = 1 ld_en = 20'd0, 10'b1111111111	H2: curr_layer = 2 ld_en = 10'd0, 10'b1111111111, 10'd0
H3: curr_layer = 3 ld_en = 10'b1111111111, 20'd0	O1: curr_layer = 4 inc_addr=1	Done: done=1

Simulation Results:



Based on the above wave form, the approximate accuracy of the model is 89%(the floating point values are ignored).

Notes:

- A python script (fix_data) was written in order to fix the format of the input data for the readmemh functions.
- A Keras model was implemented using Tensorflow with estimated accuracy of 93%. The generated weights and biases are stored and files startin with “custom”. Using these values the accuracy of the model will be 23% which is a significant decrease. This is probably because the Keras model uses a Softmax function as the activation function of the output layer rather than ReLU.