Test Cases

Team: Neural Network – Theta  
Members: Thibaut Lanois, Wen Wen, Jieming Pan

Mar 24, 2017

SW Test cases:

Implementing Convolution layer functions in C

Test Case 1: Simply compare function output with MATLAB one, which considered as golden results.  
Implementing Pooling layer functions in C

Test Case 1: Simply compare function output with MATLAB one, which considered as golden results.  
Implementing Fully connected layer functions in C

Test Case 1: Test fully connected layer with Sigmoid, Relu, and Softmax function separately and compare input and output values.

Test Case 2: Integrate the functions and compare output with MATLAB output to see if they are consistent.  
  
Obtain LeNet-5 model in C

Test Case 1: Inference C model is expected to achieve more than 80% accuracy on MNIST dataset.

Test Case 2: Inference C model is expected to achieve more than 90% accuracy on MNIST dataset.

Test Case 2: Inference C model is expected to achieve more than 99.0% accuracy, which is achieved by Caffe Model with same weights and biases, on MNIST dataset.

SW Test cases:

FSM:

Test Case 1: The input of FSM are the information about the layer given by the CPU. In this benchmark we looking into the waveform when the CPU send only 1 informations that means all the input of the FSM don’t move, except the ready if we have to stop the pipe.

Test Case 2: Same as previous TB but this time the input can move, Output still on the waveform

Router-controller:

Test Case 1: The input are weight (16 bit divide into two different 8 bit, first bit are the column address and the second is line address) and date (16 bit divide into two different 8 bit, first bit are the column address and the second is line address). Then by checking to the waveform we check if the hexadecimal combination is good. The input are fixed size and with only output and input feature map. The router can fill 2 PEs

Test Case 2: The input are weight (16 bit divide into two different 8 bit, first bit are the column address and the second is line address) and date (16 bit divide into two different 8 bit, first bit are the column address and the second is line address). Then by checking to the waveform we check if the hexadecimal combination is good. The input are fixed size and with only 6 outputs and 1 input features map. The router can fill 2 PEs

Test Case 3: The input are weight (16 bit divide into two different 8 bit, first bit are the column address and the second is line address) and date (16 bit divide into two different 8 bit, first bit are the column address and the second is line address). Then by checking to the waveform we check if the hexadecimal combination is good. The input are fixed size and with only 10 outputs and 6 input features map. The router can fill 2 PEs

Test Case 4: Same as previously, except that the router can deal with “n” PE

Router-controller-Convolution:

Test Case 1: Same as Router and controller, except that we had convolution and add real data for weight and data and bias. We save the 28\*28 pixel at the end of the convolution and compare it with c.

Test Case 2: Same as previously but using features map.

Router-controller-convolution-pooling:

Test Case 1: Testing 1 layer (1 conv + 1 pool) as previously but adding pooling

Test Case 2: testing 2 layer

Test Case 3: Every pool and convolution layer (3 and 3)

Test Case 4: Make it work but with more parallelism

Implementing Pooling layer functions:

Test Case 1: The input are fabricated 16 bit data and write/read enable signals. And the output are supposed to be full/empty and data in FIFO fashion when read is enabled. This test case is to check whether two 2\* 16 bit FIFO work correctly with according signals.

Test Case 2: The input are reset (clear everything), fabricated 16 bit data (randomly generated) and write enable signals. All the data in two separate FIFOs are pooled and the maximum one outputs with a valid signal. In the same time, the reset signal is tested with clearing all internal signals and output bits. This test case is to check whether the pooling function works correctly.

Test Case 3 (Testing with Router-controller-convolution): The differences from test case 2 is that the input data are from feature map of previous convolutional layer, and output are pooled data and valid bit.

Implementing Convolution layer functions:

Test Case 1: Functionality test: Verify individual multiplication and addition in the pipeline. Check if the weights and biases are loaded correctly, and valid bits are propagated correctly.

Test Case 2: Design test: With fixed inputs from controller, in every 9 cycles of kernel multiply-add calculation, the data order and number of multiply-add calculation is as supposed. Input: fabricated data. Output: correctly calculated results.

Softmax function in LUT:

Test Case 1: Verify exponential calculation lookup table is loaded in BRAM correctly and ReLu logic is implemented correctly.

Test Case 2: Using input values from pooling to multiply-add with weight and bias. Then, converting value to exponential using LUT. Passing through ReLu logic and add results from inputs of the same feature map together into a register. Each result is also stored in a dedicated BRAM. Check input and output of each section is correct.

Test Case 3: Extracting values from BRAM in sequence and perform Softmax division and verify results.