IMPLEMENTATION OF HANDWRITTEN DIGIT RECOGNITION USING NEURAL NETWORKS WITH MNIST DATASET

COURSE PROJECT: HAOML 2020

SUBMITTED BY,

V.NAVEEN CHANDER

SR :17550

MTECH, 1ST YEAR,

MICROELECTRONICS AND VLSI DESIGN

INDIAN INSTITUTE OF SCIENCE,

BENGALURU 560012

JUNE 6, 2020

CONTENTS

ABSTRACT

Neural Networks are extensively used as classifiers for various datasets. However, the enormous complexity in terms of computation and storage prohibits neural networks from being used on edge-AI class of applications. This project uses a Multilayer Perceptron to efficiently classify handwritten digits trained from MNIST dataset and implement the classifier on Zynq7000 FPGA SoC hardware accelerator. With just 25,840 quantized network parameters, the network is able to produce inference with a latency of **520 µs** and maintain an accuracy of **96%** on hardware. Thus, an optimal hardware-software co-design for MLP neural network has been carried out successfully.

INTRODUCTION

Neural Networks implemented on hardware accelerators are highly efficient in terms of power and computational speed. Reducing the complexity of the hardware decreases power consumption and increases the speed of the inference engine. Fortunately, neural networks are largely over parameterized []. Various techniques exist for optimizing neural networks. They can be broadly classified as:

1. Pruning
2. Quantization
3. Multiplier-less optimization

Pruning removes the redundant parameters in a neural network by identifying those parameters which play a very little role in inference. A pruned network has weights and biases which are generally represented in floating point. Fixed point computations consume about 1/6th power as that of a floating point [Xilinx Application Note] and are faster and occupy lesser area. It has been observed from various experiments that accuracy of a neural network is not considerably affected by conversion of the network parameters to fixed point values. [Quant paper]. In applications such as IoT and edge-AI which feature battery operated devices, the challenge is to minimize the power consumption. This would call for a trade off between the accuracy of the neural network and its power consumption. Techniques like Multiplier-less optimization enable reduction in power consumption to a large extent.

The aim of this project is to build an optimized neural network using the following design flow:

1. Arrive at an optimize neural network by pruning an over-parameterized network
2. Analyse the network parameters and apply statistical methods to arrive at an optimal quantization strategy
3. Constrain the network parameters to powers of two so that multiplication/division operations can be implemented as left shifts/ right shifts.
4. Compare the accuracy with that of an unoptimized neural network.

LITERATURE SURVEY

Neural Network optimization techniques encompass pruning, quantization, low rank optimization, Huffman encoding, Weight Sharing and Network Binarization. Of these, a large literature is available on pruning and quantization techniques. We shall briefly review literatures under these two headings.

Pruning can be broadly classified into two categories – Pruning of individual weights (un-structured pruning) and pruning of filter networks (structured pruning).

Unstructured methods started in 1990s with methods like Optimal Brain Surgeon (Hassibi & Stork, 1993) and Optimal Brain Damage (LeCun et al., 1990) which pruned weights based on second derivative of a Cost function for shallow CNNs. This method ceased to be effective after the inception of DNNs in 2012. (Han. et. al 2015) proposed a pruning pipeline based on magnitude pruning of weights to demonstrate a model compression of up to 9x on VGG-16 new without appreciable loss in accuracy. While unstructured methods are highly effective in compressing models, they are empirical methods without a proper mathematical framework and do not demonstrate the same level of effectiveness in large CNNs. (Liu et al., 2019). Moreover, they involve a lot of hyper-parameters such as layer pruning threshold etc., which will vary with neural network and the dataset, thus affecting the reproducibility of technique.(Dong et al., 2017) Another major drawback of unstructured pruning technique is disruption of regularity of a network, especially in CNNs, by partially pruning filters etc., Thus, modern hardware can no longer exploit regularities in CNNs, thus imposing requirements for special hardware or sparse BLAS libraries. (Han et al., 2016)

Structured Methods propose target pruning convolutional filters or whole layers as against individual weights (Dong and Yang, 2019). A saliency score based on $\ell_1 \, or \, \ell_2$ norm of the Kernel Matrix can be set as pruning criterion (Li et al., 2017) and a fixed number of channels in each convolutional layer can be pruned depending on these scores. As the scores are adaptively computed with respect to the model as well as the task, these methods can be easily applied to different scenarios (Chen et.al.,2019). However, the drawback is that this method too tends to associate the pruning criteria mainly on the magnitude of the weight, which is an empirical criterion and can have problems as mentioned before. Molchanov et.al (2019) define a framework to set a pruning criterion based on the importance of a channel to overcome the problems associated with using weights alone in the formation of pruning criteria.

Weight sharing techniques (Babu et al., 2015), device technique to remove the neurons rather than weights. Also, to compensate for the information embedded int the lost connections, pruned weights are added to the existing neural connections. This technique is effective for the FC layers of CNNs. Another structured pruning technique called, Self-adaptive pruning technique (Chen et al., 2019) considers the fact that the inputs together with the weights determine the extent of neural activation. Thus, the weights are pruned selectively based on the input values in real time. However, it comes at an expense of building a saliency estimator function for each layer to take decisions in real time.

Quantization of neural networks has been proposed by (Courbariaus et al., 2016), (Ghasemzadeh ret al., 2018) etc., involves quantizing all the weights to binary values, $W \epsilon \{-1, 1\}$. However, does not apply well to already pruned networks since, a high weight count is required to achieve a certain accuracy from a Binary Neural Network. Tann, et al. (2017) propose a multiplier-less neural network architecture wherein all the network weights are rounded off to their nearest powers of 2, which worked well on our pruned neural network.

In this work, the pruning framework by Molchanov et.al (2019) and the quantization and multiplier less optimization framework by Tann, et al. (2017) is used to optimize the Multi-layer Perceptron

CHAPTER 1: ARRIVING AT THE OPTIMIAL MLP CONFIGURATION

Convolutional Neural Networks are ideal for image classification. However, given that the current Neural Network is built for classification of simple Black and white images from MNIST handwritten digital dataset, it is decided to implement the classifier using Multi-layer Perceptron (MLP).

The image of 28 x 28 is first, flattened to 784 x 1 and fed to the MLP.

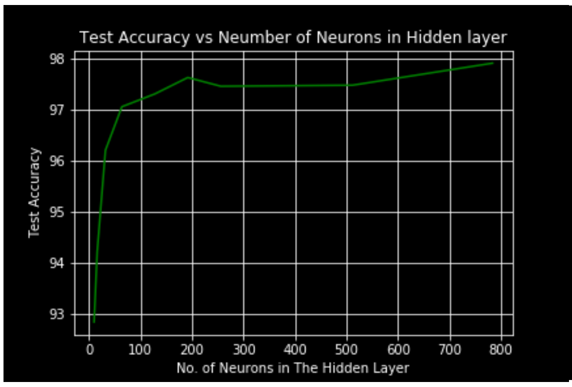
In order to zero-in on the optimal Neural Network, in terms of (a) Accuracy, (b) No. of model parameter and (c) computational latency, a series of experiments was carried out:

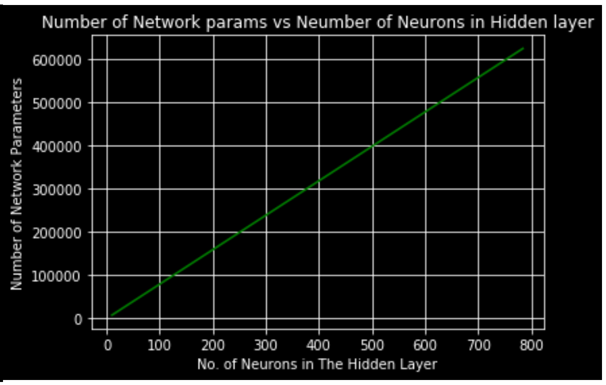
EXPERIMENT 1

Trials were carried by changing the number of neurons in the hidden layer (layer_2) and the following results were observed:

Sl.no	No. of neurons in the hidden layer	No. of parameters	Train Accuracy (%)	Test Accuracy (%)
1.	784	623290	98.89	97.90
1.	512	407050	98.83	97.47
2.	256	203530	98.67	97.45
3.	192	152650	98.47	97.62

Sl.no	No. of neurons in the hidden layer	No. of parameters	Train Accuracy (%)	Test Accuracy (%)
4.	128	101770	98.21	97.30
5.	64	50890	97.53	97.05
6.	32	25450	96.29	96.20
7.	16	12730	94.13	94.22
8.	10	7960	92.80	92.85





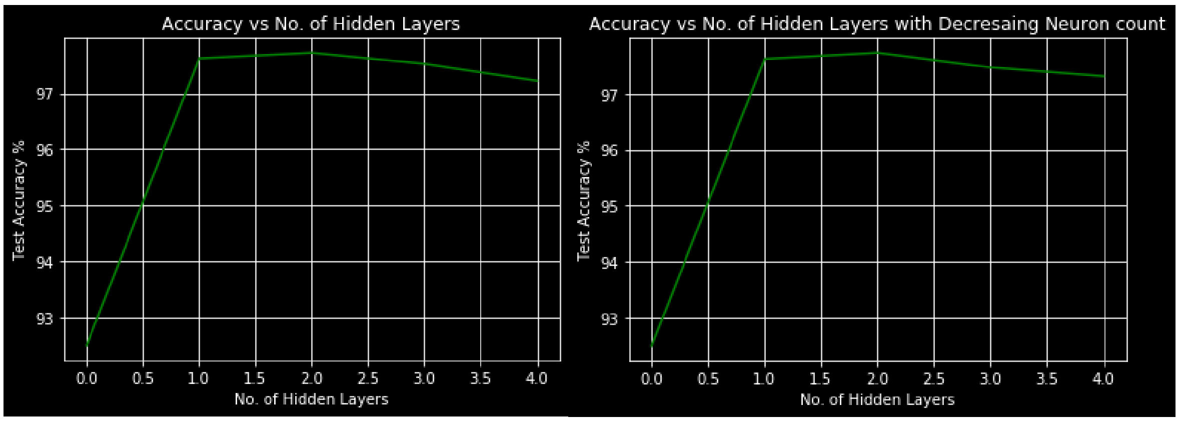
We observe that as no. of Neurons in the hidden layer starts to increase from 10, there is a sharp rise in the accuracy, which peaks at 192 neurons and then decrease slightly, followed by a slow increase. However, this increase in accuracy comes with the increased computational costs arising dur to increase in the number of parameters.

EXPERIMENT 2: DEPENDENCE ON THE NUMBER OF LAYERS IN THE MLP

Sl.no	No. of Hidden Layers	No. of parameters	Train Accuracy (%)	Test Accuracy (%)
1.	0	7850	92.36	92.50
1.	1(784→192→10)	152650	98.47	97.62
2.	2(784→192→192→10)	163722	98.59	97.72
3.	3(784→192→192→192→10)	261888	98.26	97.52
4.	4(784→192→192→192→192→10)	298944	97.74	97.22

Modification: Decrease the number of neurons in higher layers

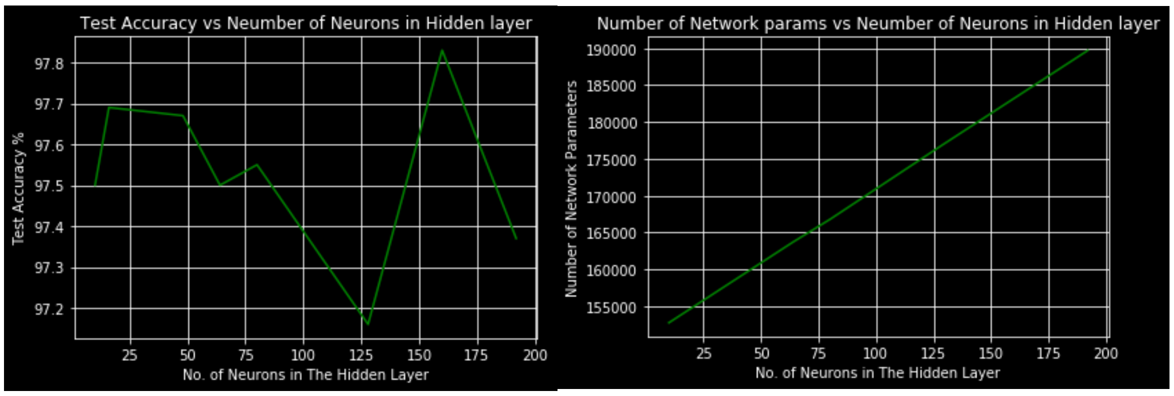
Sl.no	No. of Hidden Layers	No. of parameters	Train Accuracy (%)	Test Accuracy (%)
1.	0	7850	92.36	92.50
1.	1(784→192→10)	152650	98.47	97.62
2.	2(784→192→64→10)	163722	98.57	97.73
3.	3(784→192→64→32→10)	184330	98.37	97.47
4.	4(784→192→64→32→16→10)	186090	98.29	97.31



Observation: We see that if the number of layers increases with decreasing no. of neurons in each layer, the accuracy doesn't alter much. **Thus, maximum accuracy is obtained with two hidden layers.**

EXPERIMENT 3: VARY THE NUMBER OF NEURONS IN THE SECOND HIDDEN LAYER

Sl.no	No. of neurons in the hidden layer	No. of parameters	Test Accuracy (%)
1.	192	189706	97.500002
1.	160	183210	97.689998
2.	128	176714	97.680002
3.	80	166790	97.670001
4.	64	163722	97.500002
5.	48	160474	97.549999
6.	32	157226	97.160000
7.	16	153978	97.829998
8.	10	152760	97.369999



Observation: Accuracy doesn't change much. Therefore, select **16 neurons for the second hidden layer.**

Thus, the best neural network can be selected as:

<u>**784→192→16→10**</u>

CHAPTER 2: PRUNING (MOLCHANOV ET AL, 2019)

2.1 DEFINITIONS

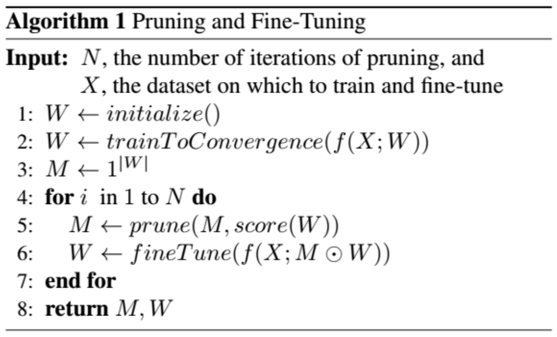
- **Network Architecture** is a family of functions $f(x;\ \cdot)$ that consists of configuration of network's parameters, sets of operations to produce outputs from inputs such as convolutions, activations, pooling etc.,
    - Ex: AlexNet, VGG Net
- **Neural Network Model** is a parameterization of the architecture, i.e., $f(x; W)$ for specific parameters W.
- **Neural Network Pruning** entails taking an input model as $f(x; W)$ and producing a new model $f(x; M \odot W')$.
    - Here, W' is a set of parameters that may be different from W,
    - $M \epsilon \{0,1\}^{|W'|}$ is a binary mask that fixes certain parameters to 0
    - $\odot$ is element-wise product operator

2.2 PRUNING ALGORITHM USED

Importance Estimation based Pruning Algorithm [Molchanov et.al, 2019] has been used to prune the neural network. Here, the salience feature for each weight is its importance which is defined by:

$$\mathcal{I}_m^{(1)}(W) = (g_m w_m)^2$$

Where $\mathcal{I}_m^{(1)}(W)$ is the importance of w_m, $g_m = \frac{\delta E}{\delta w_m}$ are elements of the gradient $\boldsymbol{g}$. Importance can easily be computed as it is available in backpropagation.



```
Algorithm 1 Pruning and Fine-Tuning
Input: $N$, the number of iterations of pruning, and
       $X$, the dataset on which to train and fine-tune
1: $W \leftarrow initialize()$
2: $W \leftarrow trainToConvergence(f(X; W))$
3: $M \leftarrow 1^{|W|}$
4: for $i$ in 1 to $N$ do
5:     $M \leftarrow prune(M, score(W))$
6:     $W \leftarrow fineTune(f(X; M \odot W))$
7: end for
8: return $M, W$
```

Figure 1 Pruning Algorithm

2.3 PRUNING RESULTS

The following model was pruned according to this algorithm on tensorflow.

```
Layer (type)                 Output Shape              Param #   
=================================================================
flatten (Flatten)            (None, 784)               0         
_________________________________________________________________
dense (Dense)                (None, 256)               200960    
_________________________________________________________________
dense_1 (Dense)              (None, 32)                8224      
_________________________________________________________________
dense_2 (Dense)              (None, 10)                330       
=================================================================
Total params: 209,514
Trainable params: 209,514
Non-trainable params: 0
```

IMPORTANCE PLOTS

The following are plots of importance of weights

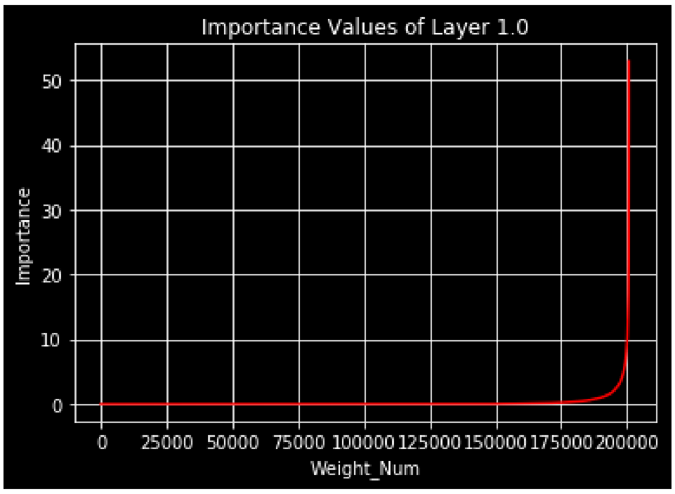


Figure 2 Importance of Layer 1 weights

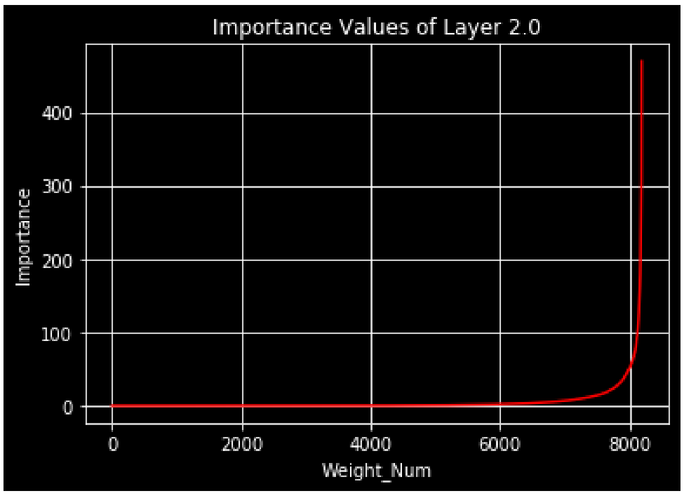


Figure 3 Importance of layer 2 Weights

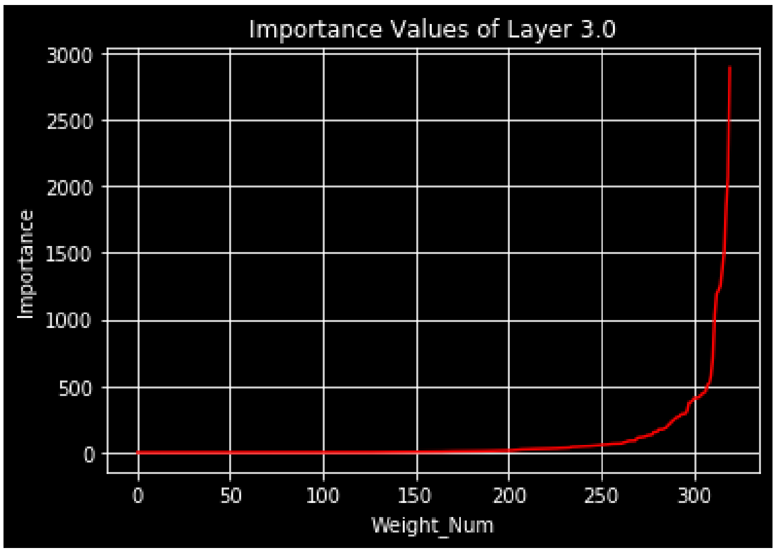


Figure 4 Importance of layer 3 Weights

OBSERVATIONS

1. Importance values decrease from last layer to first layer. The relatively low importance of the first hidden layer neurons and weights when compared to those in the last layer allow us to prune a larger percentage of weights from the first hidden layer.
2. In the fits hidden layer, most of the weights have relative importance close to zero and a clear set of weights have high importance values.

PRUNING CRITERIA

- Thus, the pruning criterion for the first layer can be set to prune all the weights that have importance **< 0.5%** of the maximum importance in first layer.
- Here, 0.5% is the Hyper-parameter.
- Second and Third layers shall not be pruned.

PRUNING RESULTS

- On pruning, the number of parameters in the first layer comes out to be = 16485. Thus, number of neurons required in the first hidden layer to achieve this neuron count is close to 32.
- Thus, the pruned network is

784→32→16→10

CHAPTER 3: QUANTIZATION FOR MULTIPLIERLESS OPTIMIZATION (TANN ET AL, 2017)

DNNs with low precision data formats have enormous potential for reducing hardware complexity (Tann et.al, 2017). Quantization for deep learning is the process of approximating a neural network that uses floating-point numbers by a neural network of low bit width numbers. This dramatically reduces both the memory requirement and computational cost of using neural networks.

The neural network can be quantized after training is finished. However, by far the most effective method for retaining high accuracy is to quantize during training. However, in this work, quantization is done after training and pruning.

Pruned Model Summary

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)             (None, 784)               0

_________________________________________________________________
dense (Dense)                 (None, 32)                25120

_________________________________________________________________
dense_1 (Dense)               (None, 16)                528

_________________________________________________________________
dense_2 (Dense)               (None, 10)                170
=================================================================
Total params: 25,818
Trainable params: 25,818
Non-trainable params: 0

_________________________________________________________________
```

3.1 STATISTICAL ANALYSIS OF MODEL PARAMETERS AND RESULTS

In order to arrive at the optimal value of Integer bits and decimal bits, it is essential to examine the spread of all model parameters and intermediate signals in each layer. In order to achieve this, the following is done:

For all training inputs,

- *Evaluate the model*
- *Store all the intermediate outputs*
- *Store the output of all intermediate layers in separate arrays*
- *Plot a histogram of the absolute values of layer outputs and weights*

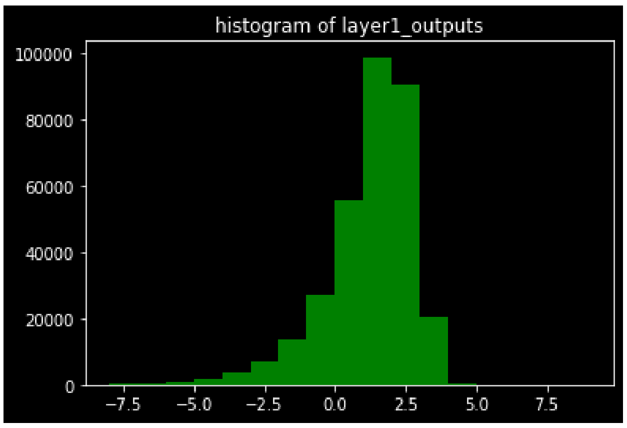
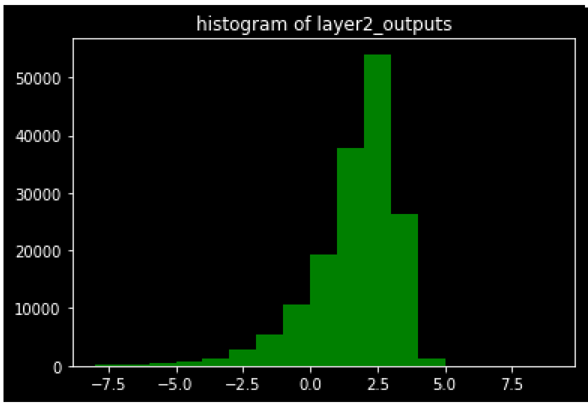
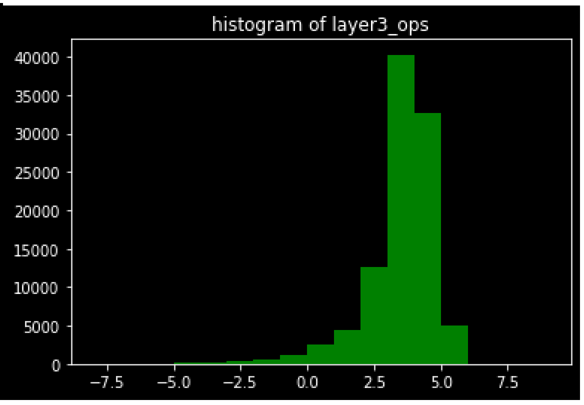
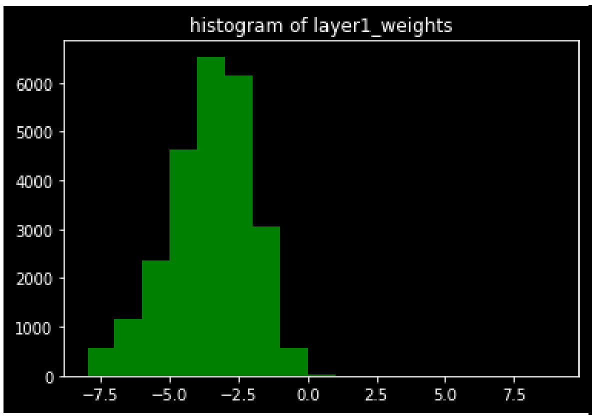
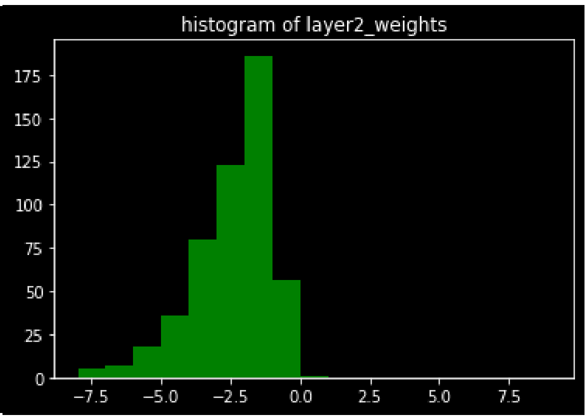


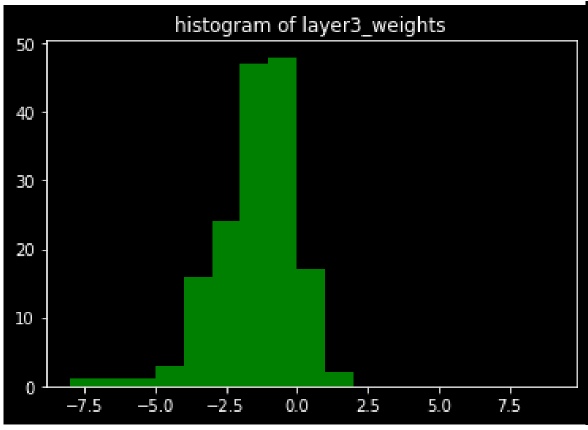
Figure 5 Histogram of Abs. value Layer Outputs. X-axis is in log2 scale.

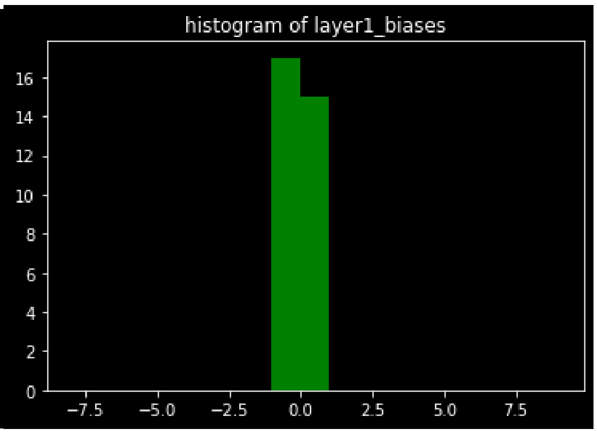

histogram of layer2_outputs

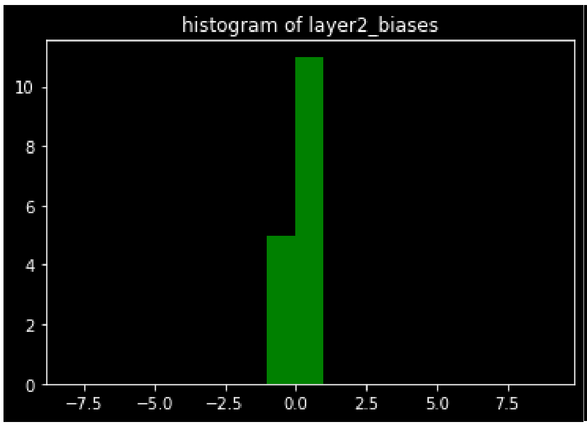

histogram of layer3_ops

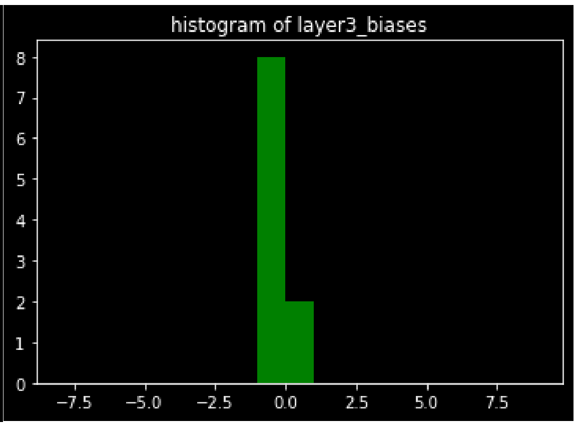

histogram of layer1_weights


histogram of layer2_weights


histogram of layer3_weights


histogram of layer1_biases


histogram of layer2_biases


histogram of layer3_biases

INFERENCE

- The range of all parameters is between -8 and +6 i.e., between 2^-8 and 2^6.
- However, most of the parameters lie between 2^-3 and 2^5.
- Range of weights is lesser. All weights AND biases lie between **-1.78 and +1.5**

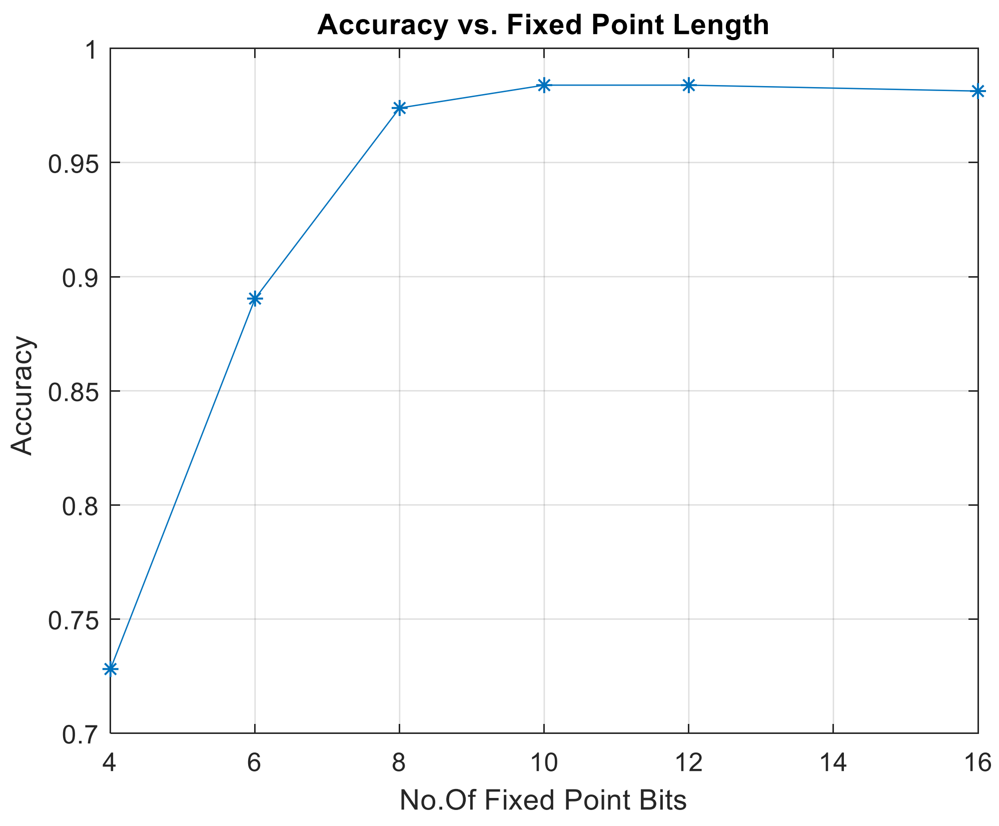
3.3 QUANTIZATION FOR MULTIPLIERLESS OPTIMIZATION

- In order to convert all weights to fixed-point values, <8,5> is used.
- <8,5> refers to 5 bits for Integer and 8 bits in total. Therefore, 3 bits for decimal
- For multiplier-less optimization, we round off all weights to nearest power of 2 between 2^-3 and 2^5
- Thus, all weights and biases are rounded off to powers of 2.
- This will convert all multiplications to shift operations.

3.4 RESULTS FOR QUANTIZATION WITH MULTIPLIERLESS OPTIMIZATION

The model parameters were exported from Python to MATLAB, so that MATLAB's fixed point tool can be employed to do calculations in the fixed point domain. Neural Network model was coded in MATLAB with power-of-2 quantized weights and inputs. The accuracy was monitored for various combinations of integer bits and decimal bits. The following are the results for accuracy:

Sl.no	Fixed Point Total Size	Fraction	Accuracy
1.	16	6	98.13
2.	12	4	98.39
3.	10	4	98.39
4.	8	3	97.39
5.	6	3	89.04
6.	4	2	72.81



- Thus, we see that the model with 8 bits is the ideal choice.

CHAPTER 4: IMPLEMENTATION USING HLS

The Neural network was coded in HLS C++ to communicate to the external world using AXI-M bus of width 16 bits.

NOTE: AXI-bus supports I/O bit widths that are multiples of 8. Therefore, only 8-bit and 16-bit architectures could be tried out in HLS.

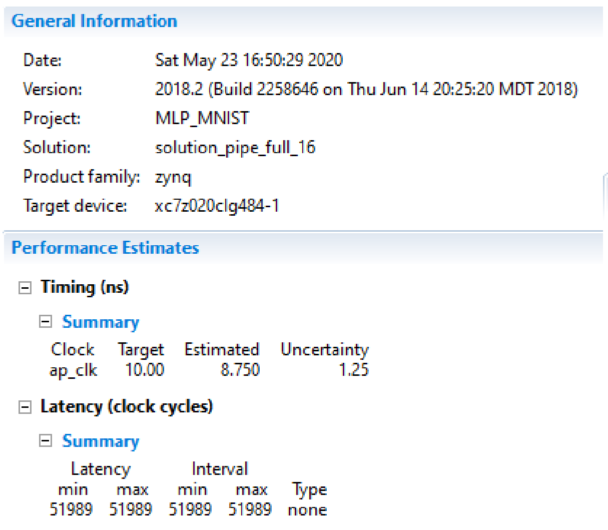
The following directives were used: PIPELINE, LOOP UNROLL, ARRAY RESHAPING while synthesizing the Neural Network. Following table shows the performance on various architectures:

Sl.no	Quantization	Loop Unrolling	Pipelining	Latency (us)	%Utilization	%Accuracy
1.	8	No	No	777.32	2.5%	20%
2.	8	No	Yes	519.90	2.5%	20%
3.	8	Yes	Yes	1.31	>> 100%	20%
4.	16	No	No	777.32	3.25%	91%
5.	16	No	Yes	519.90	3.25%	91%
6.	16	Yes	Yes	1.31	>> 100%	91%

Thus, we see that implementation Sl.no 5 is the optimal implementation in terms of Power, Performance and Area.

The results for quantized bits in MATLAB do not agree with the Hardware Results. This might be attributed to some unknown behaviour of Fixed-Point Library **ap_int.**

SYNTHESIS RESULTS

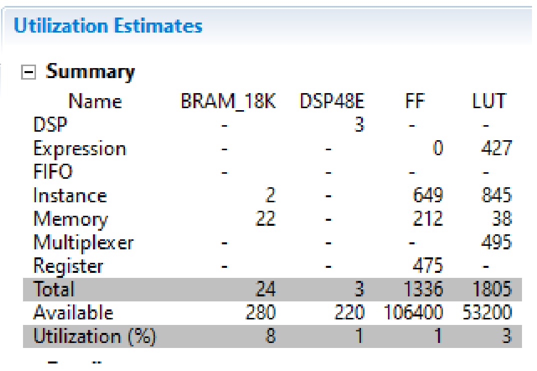


General Information

Date:	Sat May 23 16:50:29 2020
Version:	2018.2 (Build 2258646 on Thu Jun 14 20:25:20 MDT 2018)
Project:	MLP_MNIST
Solution:	solution_pipe_full_16
Product family:	zynq
Target device:	xc7z020clg484-1

Performance Estimates

```
⊟ Timing (ns)
  ⊟ Summary
     Clock   Target  Estimated  Uncertainty
     ap_clk  10.00   8.750      1.25
⊟ Latency (clock cycles)
  ⊟ Summary
       Latency       Interval
     min    max    min    max    Type
     51989  51989  51989  51989  none
```



Utilization Estimates

⊟ Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	3	-	-
Expression	-	-	0	427
FIFO	-	-	-	-
Instance	2	-	649	845
Memory	22	-	212	38
Multiplexer	-	-	-	495
Register	-	-	475	-
Total	24	3	1336	1805
Available	280	220	106400	53200
Utilization (%)	8	1	1	3

Hardware Co-simulation Results

RTL CO SIMULATION SUMMARY

- Tested Images: 50
- Pass:48
- Accuracy: 96%

DETAILED RTL CO- SIMULATION REPORT

```
////////////////////////////////////////////////////////////////////////////////////
//
// Inter-Transaction Progress: Completed Transaction / Total Transaction
// Intra-Transaction Progress: Measured Latency / Latency Estimation * 100%
//
// RTL Simulation : "Inter-Transaction Progress" ["Intra-Transaction Progress"] @
"Simulation Time"
////////////////////////////////////////////////////////////////////////////////////
//
// RTL Simulation : 0 / 10 [0.00%] @ "125000"
// RTL Simulation : 1 / 10 [0.00%] @ "779465000"
// RTL Simulation : 2 / 10 [0.00%] @ "1558815000"
// RTL Simulation : 3 / 10 [0.00%] @ "2338165000"
// RTL Simulation : 4 / 10 [0.00%] @ "3117515000"
// RTL Simulation : 5 / 10 [0.00%] @ "3896865000"
// RTL Simulation : 6 / 10 [0.00%] @ "4676215000"
// RTL Simulation : 7 / 10 [0.00%] @ "5455565000"
// RTL Simulation : 8 / 10 [0.00%] @ "6234915000"
// RTL Simulation : 9 / 10 [0.00%] @ "7014265000"
// RTL Simulation : 10 / 10 [100.00%] @ "7793615000"
////////////////////////////////////////////////////////////////////////////////////
//
$finish called at time : 7793655 ns : File
"C:/Users/mghnv/Documents/HAoML/Project/MLP_MNIST/MLP_MNIST/solution_pipe_full_16/
sim/verilog/MLP_MNIST.autotb.v" Line 437
run: Time (s): cpu = 00:00:00 ; elapsed = 00:00:32 . Memory (MB): peak = 212.523 ;
gain = 0.000
## quit
INFO: [Common 17-206] Exiting xsim at Sat Jun  6 20:40:40 2020...
INFO: [COSIM 212-316] Starting C post checking ...
started
outer loop Started
hw_result 0 :=-9.71484::sw_result 0:=-14.7148::
hw_result 1 :=-11.2109::sw_result 1:=-15.8359::
hw_result 2 :=-3.59766::sw_result 2:=-9.55859::
hw_result 3 :=-0.628906::sw_result 3:=-6.10547::
hw_result 4 :=-14.1445::sw_result 4:=-13.8398::
hw_result 5 :=-4.04297::sw_result 5:=-15.3867::
hw_result 6 :=-14.9531::sw_result 6:=-22.8398::
hw_result 7 :=7.55859::sw_result 7:=2.98047::
hw_result 8 :=-7.00391::sw_result 8:=-7.23828::
hw_result 9 :=-8.87891::sw_result 9:=-5.53125::
//--------------------------------------------------
Prediction :7
//--------------------------------------------------
outer loop Started
hw_result 0 :=-21.6836::sw_result 0:=-37.9375::
hw_result 1 :=-3.07031::sw_result 1:=-5.85938::
hw_result 2 :=6.57031::sw_result 2:=8.63672::
```

```
hw_result 3 :=-10.1016::sw_result 3:=-5.69922::
hw_result 4 :=-30.3047::sw_result 4:=-35.3086::
hw_result 5 :=-12.8477::sw_result 5:=-17.9297::
hw_result 6 :=-11.4023::sw_result 6:=-21.5898::
hw_result 7 :=-17.8789::sw_result 7:=-20.6406::
hw_result 8 :=-22.0938::sw_result 8:=-14.207::
hw_result 9 :=-40.9883::sw_result 9:=-31.7656::
//--------------------------------------------------
Prediction :2
//--------------------------------------------------
outer loop Started
hw_result 0 :=-9.49609::sw_result 0:=-12.457::
hw_result 1 :=4.59375::sw_result 1:=2.70703::
hw_result 2 :=-2.83594::sw_result 2:=-4.10547::
hw_result 3 :=-7.99219::sw_result 3:=-8.35938::
hw_result 4 :=-4.84766::sw_result 4:=-6.58984::
hw_result 5 :=-8.17969::sw_result 5:=-10.0078::
hw_result 6 :=-6.08594::sw_result 6:=-7.35547::
hw_result 7 :=-2.16016::sw_result 7:=-4.94531::
hw_result 8 :=-8.69531::sw_result 8:=-4.75781::
hw_result 9 :=-13.1094::sw_result 9:=-8.71875::
//--------------------------------------------------
Prediction :1
//--------------------------------------------------
outer loop Started
hw_result 0 :=0.429688::sw_result 0:=-1.58203::
hw_result 1 :=-15.2656::sw_result 1:=-28.875::
hw_result 2 :=-8.11328::sw_result 2:=-12.0078::
hw_result 3 :=-15.3438::sw_result 3:=-11.5391::
hw_result 4 :=-14.4688::sw_result 4:=-18.375::
hw_result 5 :=-7.14453::sw_result 5:=-13.0664::
hw_result 6 :=-8.90625::sw_result 6:=-8.14844::
hw_result 7 :=-8.08203::sw_result 7:=-14.5039::
hw_result 8 :=-26.5234::sw_result 8:=-13.2695::
hw_result 9 :=-14.0742::sw_result 9:=-17.8828::
//--------------------------------------------------
Prediction :0
//--------------------------------------------------
outer loop Started
hw_result 0 :=-9.79688::sw_result 0:=-13.5938::
hw_result 1 :=-5.40234::sw_result 1:=-25.4414::
hw_result 2 :=-5.61719::sw_result 2:=-11.5117::
hw_result 3 :=-9.91406::sw_result 3:=-11.9336::
hw_result 4 :=1.49609::sw_result 4:=-0.777344::
hw_result 5 :=-7.03516::sw_result 5:=-9.87891::
hw_result 6 :=-5.71094::sw_result 6:=-11.25::
hw_result 7 :=-2.97656::sw_result 7:=-9.55859::
hw_result 8 :=-12.3945::sw_result 8:=-11.7969::
hw_result 9 :=-3.01172::sw_result 9:=-6.99219::
//--------------------------------------------------
Prediction :4
//--------------------------------------------------
outer loop Started
hw_result 0 :=-11.2109::sw_result 0:=-15.2617::
hw_result 1 :=3.64063::sw_result 1:=1.67969::
hw_result 2 :=-4.33594::sw_result 2:=-6.45703::
hw_result 3 :=-8.83984::sw_result 3:=-9.92969::
hw_result 4 :=-3.60156::sw_result 4:=-7.71484::
hw_result 5 :=-10.2539::sw_result 5:=-11.9727::
```

```
hw_result 6 :=-8.90234::sw_result 6:=-11.4961::
hw_result 7 :=-1.64453::sw_result 7:=-3.94531::
hw_result 8 :=-8.08594::sw_result 8:=-7.15234::
hw_result 9 :=-11.9844::sw_result 9:=-9.10156::
//--------------------------------------------------
Prediction :1
//--------------------------------------------------
outer loop Started
hw_result 0 :=-19.3047::sw_result 0:=-29.418::
hw_result 1 :=-9.08594::sw_result 1:=-13.3984::
hw_result 2 :=-13.4297::sw_result 2:=-15.0625::
hw_result 3 :=-12.9648::sw_result 3:=-17.0508::
hw_result 4 :=2.01953::sw_result 4:=1.72656::
hw_result 5 :=-6.29297::sw_result 5:=-13.6797::
hw_result 6 :=-7.89453::sw_result 6:=-18.8438::
hw_result 7 :=-5.46875::sw_result 7:=-8.30469::
hw_result 8 :=-8.80859::sw_result 8:=-8.27344::
hw_result 9 :=-5.39844::sw_result 9:=-12.6563::
//--------------------------------------------------
Prediction :4
//--------------------------------------------------
outer loop Started
hw_result 0 :=-14.3359::sw_result 0:=-25.6016::
hw_result 1 :=-7.53906::sw_result 1:=-10.6875::
hw_result 2 :=-9.48047::sw_result 2:=-9.10547::
hw_result 3 :=0.359375::sw_result 3:=-6.41797::
hw_result 4 :=-7.26953::sw_result 4:=-6.74609::
hw_result 5 :=-1.32813::sw_result 5:=-7::
hw_result 6 :=-9.875::sw_result 6:=-17.1758::
hw_result 7 :=-6.24609::sw_result 7:=-6.76172::
hw_result 8 :=-6.53516::sw_result 8:=-11.8555::
hw_result 9 :=1.25391::sw_result 9:=-2.05078::
//--------------------------------------------------
Prediction :9
//--------------------------------------------------
outer loop Started
hw_result 0 :=-21.8867::sw_result 0:=-29.5156::
hw_result 1 :=-30.4492::sw_result 1:=-49.8984::
hw_result 2 :=-26.2773::sw_result 2:=-26.082::
hw_result 3 :=-22.1602::sw_result 3:=-30.8008::
hw_result 4 :=-24.7813::sw_result 4:=-18.1875::
hw_result 5 :=-0.117188::sw_result 5:=-10.7422::
hw_result 6 :=2.24609::sw_result 6:=-17.1914::
hw_result 7 :=-26.2422::sw_result 7:=-33.4961::
hw_result 8 :=-19.9766::sw_result 8:=-26.8242::
hw_result 9 :=-23.0352::sw_result 9:=-20.8008::
//--------------------------------------------------
Prediction :6
//--------------------------------------------------
outer loop Started
hw_result 0 :=-18.3633::sw_result 0:=-22.0352::
hw_result 1 :=-12.5781::sw_result 1:=-18.3242::
hw_result 2 :=-16.6289::sw_result 2:=-16.4297::
hw_result 3 :=-8.23047::sw_result 3:=-17.7695::
hw_result 4 :=-0.0273438::sw_result 4:=-5.82422::
hw_result 5 :=-8.32813::sw_result 5:=-17.0234::
hw_result 6 :=-18.2109::sw_result 6:=-25.1992::
hw_result 7 :=-0.167969::sw_result 7:=-5.76953::
hw_result 8 :=-10.7656::sw_result 8:=-12.3906::
```

```
hw_result 9 :=4.82422::sw_result 9:=-0.210938::
//-----------------------------------------------
Prediction :9
//-----------------------------------------------
```

Correctly Identified :9/10

```
////////////////////////////////////////////////////////////////////////////////
//
// Inter-Transaction Progress: Completed Transaction / Total Transaction
// Intra-Transaction Progress: Measured Latency / Latency Estimation * 100%
//
// RTL Simulation : "Inter-Transaction Progress" ["Intra-Transaction Progress"] @
"Simulation Time"
////////////////////////////////////////////////////////////////////////////////
//
// RTL Simulation : 0 / 10 [0.00%] @ "125000"
// RTL Simulation : 1 / 10 [0.00%] @ "779465000"
// RTL Simulation : 2 / 10 [0.00%] @ "1558815000"
// RTL Simulation : 3 / 10 [0.00%] @ "2338165000"
// RTL Simulation : 4 / 10 [0.00%] @ "3117515000"
// RTL Simulation : 5 / 10 [0.00%] @ "3896865000"
// RTL Simulation : 6 / 10 [0.00%] @ "4676215000"
// RTL Simulation : 7 / 10 [0.00%] @ "5455565000"
// RTL Simulation : 8 / 10 [0.00%] @ "6234915000"
// RTL Simulation : 9 / 10 [0.00%] @ "7014265000"
// RTL Simulation : 10 / 10 [100.00%] @ "7793615000"
////////////////////////////////////////////////////////////////////////////////
//
$finish called at time : 7793655 ns : File
"C:/Users/mghnv/Documents/HAoML/Project/MLP_MNIST/MLP_MNIST/solution_pipe_full_16/
sim/verilog/MLP_MNIST.autotb.v" Line 437
run: Time (s): cpu = 00:00:00 ; elapsed = 00:00:31 . Memory (MB): peak = 211.875 ;
gain = 0.000
## quit
INFO: [Common 17-206] Exiting xsim at Sat Jun  6 20:26:08 2020...
INFO: [COSIM 212-316] Starting C post checking ...
started
outer loop Started
hw_result 0 :=1.4375::sw_result 0:=-1.58203::
hw_result 1 :=-17::sw_result 1:=-46.4922::
hw_result 2 :=-4.71094::sw_result 2:=-14.418::
hw_result 3 :=-16.2617::sw_result 3:=-21.418::
hw_result 4 :=-21.8789::sw_result 4:=-24.2695::
hw_result 5 :=-8.03906::sw_result 5:=-12.4375::
hw_result 6 :=-8.21094::sw_result 6:=-11.0273::
hw_result 7 :=-17.0234::sw_result 7:=-14.0781::
hw_result 8 :=-19.3945::sw_result 8:=-19.8086::
hw_result 9 :=-20.1875::sw_result 9:=-17.8594::
//-----------------------------------------------
Prediction :0
//-----------------------------------------------
outer loop Started
hw_result 0 :=-9.72656::sw_result 0:=-12.6953::
hw_result 1 :=-17.0859::sw_result 1:=-34.8672::
hw_result 2 :=-9.05859::sw_result 2:=-13.4258::
hw_result 3 :=-9.25::sw_result 3:=-15.957::
hw_result 4 :=-14.543::sw_result 4:=-17.418::
hw_result 5 :=-6.08203::sw_result 5:=-11.6094::
hw_result 6 :=1.375::sw_result 6:=-0.882813::
```

```
hw_result 7 :=-20.8438::sw_result 7:=-22.9375::
hw_result 8 :=-5.91406::sw_result 8:=-8.10938::
hw_result 9 :=-15.0586::sw_result 9:=-21.5898::
//-------------------------------------------------
Prediction :6
//-------------------------------------------------
outer loop Started
hw_result 0 :=-18.1133::sw_result 0:=-14.7891::
hw_result 1 :=-16.2969::sw_result 1:=-17.0547::
hw_result 2 :=-13.3242::sw_result 2:=-12.582::
hw_result 3 :=-4.72656::sw_result 3:=-11.6875::
hw_result 4 :=-5.96094::sw_result 4:=-5.91797::
hw_result 5 :=-1.30859::sw_result 5:=-11.0742::
hw_result 6 :=-12.7188::sw_result 6:=-20.1016::
hw_result 7 :=-5.48438::sw_result 7:=-6.86719::
hw_result 8 :=-9.33984::sw_result 8:=-11.8359::
hw_result 9 :=1.17188::sw_result 9:=1.38281::
//-------------------------------------------------
Prediction :9
//-------------------------------------------------
outer loop Started
hw_result 0 :=0.171875::sw_result 0:=1.32031::
hw_result 1 :=-15.5859::sw_result 1:=-35.9219::
hw_result 2 :=-8.34766::sw_result 2:=-10.2813::
hw_result 3 :=-16.4609::sw_result 3:=-14.6445::
hw_result 4 :=-17.2188::sw_result 4:=-17.5508::
hw_result 5 :=-7.78906::sw_result 5:=-9.82031::
hw_result 6 :=-10.4453::sw_result 6:=-7.62109::
hw_result 7 :=-11.5586::sw_result 7:=-12.082::
hw_result 8 :=-18.8906::sw_result 8:=-11.3555::
hw_result 9 :=-19.543::sw_result 9:=-12.293::
//-------------------------------------------------
Prediction :0
//-------------------------------------------------
outer loop Started
hw_result 0 :=-14.207::sw_result 0:=-26.6016::
hw_result 1 :=4.96484::sw_result 1:=1.44531::
hw_result 2 :=-10.207::sw_result 2:=-11.707::
hw_result 3 :=-6.88281::sw_result 3:=-10.8633::
hw_result 4 :=-10.0039::sw_result 4:=-12.6602::
hw_result 5 :=-9.00391::sw_result 5:=-12.3398::
hw_result 6 :=-12.2422::sw_result 6:=-11.0469::
hw_result 7 :=-4.96094::sw_result 7:=-13.1328::
hw_result 8 :=-8.68359::sw_result 8:=-7.84766::
hw_result 9 :=-13.4023::sw_result 9:=-10.5039::
//-------------------------------------------------
Prediction :1
//-------------------------------------------------
outer loop Started
hw_result 0 :=-13.0664::sw_result 0:=-17.7656::
hw_result 1 :=-10.1211::sw_result 1:=-21.7227::
hw_result 2 :=-11.9609::sw_result 2:=-14.6836::
hw_result 3 :=-1.50391::sw_result 3:=-4.51953::
hw_result 4 :=-22.1758::sw_result 4:=-16.5859::
hw_result 5 :=1.60938::sw_result 5:=0.917969::
hw_result 6 :=-12.0898::sw_result 6:=-9.80859::
hw_result 7 :=-15.8516::sw_result 7:=-20.1914::
hw_result 8 :=-7.55469::sw_result 8:=-7.81641::
hw_result 9 :=-12.7734::sw_result 9:=-11.1367::
```

```
//--------------------------------------------------
Prediction :5
//--------------------------------------------------
outer loop Started
hw_result 0 :=-12.8867::sw_result 0:=-8.85938::
hw_result 1 :=-14.4961::sw_result 1:=-25.7852::
hw_result 2 :=-8.40625::sw_result 2:=-11.7969::
hw_result 3 :=-5.94922::sw_result 3:=-10.2695::
hw_result 4 :=-5.25781::sw_result 4:=-6.47266::
hw_result 5 :=-3.19141::sw_result 5:=-12.6953::
hw_result 6 :=-10.1758::sw_result 6:=-14.3203::
hw_result 7 :=-4.23828::sw_result 7:=-5.50391::
hw_result 8 :=-8.21875::sw_result 8:=-8.89453::
hw_result 9 :=0.242188::sw_result 9:=0.171875::
//--------------------------------------------------
Prediction :9
//--------------------------------------------------
outer loop Started
hw_result 0 :=-14.5938::sw_result 0:=-19.3594::
hw_result 1 :=-11.0781::sw_result 1:=-15.5625::
hw_result 2 :=-0.550781::sw_result 2:=-4.50781::
hw_result 3 :=-2.76172::sw_result 3:=-4.38281::
hw_result 4 :=-17.3516::sw_result 4:=-15.3086::
hw_result 5 :=-6.83203::sw_result 5:=-15.7109::
hw_result 6 :=-15.5195::sw_result 6:=-26.3125::
hw_result 7 :=9.55078::sw_result 7:=5.74219::
hw_result 8 :=-14.6953::sw_result 8:=-11.4961::
hw_result 9 :=-16.6289::sw_result 9:=-8.97266::
//--------------------------------------------------
Prediction :7
//--------------------------------------------------
outer loop Started
hw_result 0 :=-16.2188::sw_result 0:=-16.9961::
hw_result 1 :=-13.1563::sw_result 1:=-15.9336::
hw_result 2 :=-5.16406::sw_result 2:=-9.85938::
hw_result 3 :=0.789063::sw_result 3:=-2.75391::
hw_result 4 :=-15.1641::sw_result 4:=-16.8555::
hw_result 5 :=-0.890625::sw_result 5:=-9.26563::
hw_result 6 :=-6.03516::sw_result 6:=-10.1094::
hw_result 7 :=-16.9102::sw_result 7:=-18.332::
hw_result 8 :=-1.03516::sw_result 8:=-2.47656::
hw_result 9 :=-14.7148::sw_result 9:=-10.6758::
//--------------------------------------------------
Prediction :3
//--------------------------------------------------
outer loop Started
hw_result 0 :=-14.375::sw_result 0:=-18.7148::
hw_result 1 :=-4.34766::sw_result 1:=-15.7227::
hw_result 2 :=-7.72656::sw_result 2:=-12.3359::
hw_result 3 :=-6.10938::sw_result 3:=-11.7969::
hw_result 4 :=0.09375::sw_result 4:=0.0273438::
hw_result 5 :=-6.21484::sw_result 5:=-9.18359::
hw_result 6 :=-8.98828::sw_result 6:=-14.3086::
hw_result 7 :=-1.89063::sw_result 7:=-9.24609::
hw_result 8 :=-12.7031::sw_result 8:=-11.8359::
hw_result 9 :=-4.57031::sw_result 9:=-8.74219::
//--------------------------------------------------
Prediction :4
//--------------------------------------------------
```

```
Correctly Identified :10/10

////////////////////////////////////////////////////////////////////////////////
//
// Inter-Transaction Progress: Completed Transaction / Total Transaction
// Intra-Transaction Progress: Measured Latency / Latency Estimation * 100%
//
// RTL Simulation : "Inter-Transaction Progress" ["Intra-Transaction Progress"] @
"Simulation Time"
////////////////////////////////////////////////////////////////////////////////
//
// RTL Simulation : 0 / 10 [0.00%] @ "125000"
// RTL Simulation : 1 / 10 [0.00%] @ "779465000"
// RTL Simulation : 2 / 10 [0.00%] @ "1558815000"
// RTL Simulation : 3 / 10 [0.00%] @ "2338165000"
// RTL Simulation : 4 / 10 [0.00%] @ "3117515000"
// RTL Simulation : 5 / 10 [0.00%] @ "3896865000"
// RTL Simulation : 6 / 10 [0.00%] @ "4676215000"
// RTL Simulation : 7 / 10 [0.00%] @ "5455565000"
// RTL Simulation : 8 / 10 [0.00%] @ "6234915000"
// RTL Simulation : 9 / 10 [0.00%] @ "7014265000"
// RTL Simulation : 10 / 10 [100.00%] @ "7793615000"
////////////////////////////////////////////////////////////////////////////////
//
$finish called at time : 7793655 ns : File
"C:/Users/mghnv/Documents/HAoML/Project/MLP_MNIST/MLP_MNIST/solution_pipe_full_16/
sim/verilog/MLP_MNIST.autotb.v" Line 437
run: Time (s): cpu = 00:00:00 ; elapsed = 00:00:32 . Memory (MB): peak = 211.984 ;
gain = 0.000
## quit
INFO: [Common 17-206] Exiting xsim at Sat Jun  6 20:47:56 2020...
INFO: [COSIM 212-316] Starting C post checking ...
started
outer loop Started
hw_result 0 :=-10.1836::sw_result 0:=-15.4258::
hw_result 1 :=-8.62109::sw_result 1:=-16.1914::
hw_result 2 :=-16.4688::sw_result 2:=-15.7773::
hw_result 3 :=-7.60547::sw_result 3:=-11.5234::
hw_result 4 :=-0.167969::sw_result 4:=-7.89063::
hw_result 5 :=-6.26563::sw_result 5:=-13.3398::
hw_result 6 :=-15.9883::sw_result 6:=-22.1523::
hw_result 7 :=-0.265625::sw_result 7:=-5.05078::
hw_result 8 :=-10.2031::sw_result 8:=-12.2852::
hw_result 9 :=0.53125::sw_result 9:=-2.17578::
//------------------------------------------------
Prediction :9
//------------------------------------------------
outer loop Started
hw_result 0 :=-12.3789::sw_result 0:=-7.68359::
hw_result 1 :=-14.8672::sw_result 1:=-28.7891::
hw_result 2 :=-12.4219::sw_result 2:=-10.2578::
hw_result 3 :=-9.77344::sw_result 3:=-12.6602::
hw_result 4 :=-17.0742::sw_result 4:=-11.3711::
hw_result 5 :=-2::sw_result 5:=-5.42578::
hw_result 6 :=-0.109375::sw_result 6:=3.00391::
hw_result 7 :=-17.8242::sw_result 7:=-21.4219::
hw_result 8 :=-7.48047::sw_result 8:=-13.25::
hw_result 9 :=-18.1797::sw_result 9:=-18.7539::
//------------------------------------------------
```

```
Prediction :6
//-----------------------------------------------
outer loop Started
hw_result 0 :=-10.3008::sw_result 0:=-15.0234::
hw_result 1 :=-11.7266::sw_result 1:=-34.6055::
hw_result 2 :=-7.67188::sw_result 2:=-14.2344::
hw_result 3 :=-10.0391::sw_result 3:=-16.1641::
hw_result 4 :=-7.81641::sw_result 4:=-12.582::
hw_result 5 :=-8.11328::sw_result 5:=-10.8711::
hw_result 6 :=1.85938::sw_result 6:=-3.04688::
hw_result 7 :=-8.92578::sw_result 7:=-17.957::
hw_result 8 :=-10.9102::sw_result 8:=-12.2656::
hw_result 9 :=-17.3008::sw_result 9:=-24.7773::
//-----------------------------------------------
Prediction :6
//-----------------------------------------------
outer loop Started
hw_result 0 :=-23.4492::sw_result 0:=-19.8359::
hw_result 1 :=-18.875::sw_result 1:=-23.832::
hw_result 2 :=-27.4531::sw_result 2:=-18.8359::
hw_result 3 :=-10.2617::sw_result 3:=-10.5664::
hw_result 4 :=-25.8633::sw_result 4:=-17.7109::
hw_result 5 :=3.07031::sw_result 5:=4.19922::
hw_result 6 :=-12.5273::sw_result 6:=-11.5859::
hw_result 7 :=-21.1055::sw_result 7:=-28.0664::
hw_result 8 :=-14.4063::sw_result 8:=-17.3984::
hw_result 9 :=-20.3125::sw_result 9:=-16.1563::
//-----------------------------------------------
Prediction :5
//-----------------------------------------------
outer loop Started
hw_result 0 :=-10.2813::sw_result 0:=-12.0938::
hw_result 1 :=-5.89063::sw_result 1:=-14.6523::
hw_result 2 :=-7.51563::sw_result 2:=-9.23047::
hw_result 3 :=-6.16797::sw_result 3:=-11.1055::
hw_result 4 :=0.636719::sw_result 4:=0.671875::
hw_result 5 :=-4.47266::sw_result 5:=-7.84766::
hw_result 6 :=-7.03516::sw_result 6:=-9.15234::
hw_result 7 :=-1.25391::sw_result 7:=-4.88672::
hw_result 8 :=-9.50781::sw_result 8:=-9.42188::
hw_result 9 :=-1.5625::sw_result 9:=-5.05078::
//-----------------------------------------------
Prediction :4
//-----------------------------------------------
outer loop Started
hw_result 0 :=-1.41797::sw_result 0:=-0.535156::
hw_result 1 :=-34.0273::sw_result 1:=-55.6289::
hw_result 2 :=-15.7383::sw_result 2:=-21.1094::
hw_result 3 :=-24.6836::sw_result 3:=-27.7578::
hw_result 4 :=-25.9219::sw_result 4:=-19.6133::
hw_result 5 :=-10.7422::sw_result 5:=-26.0156::
hw_result 6 :=-11.4414::sw_result 6:=-12.4375::
hw_result 7 :=-23.4492::sw_result 7:=-16.9961::
hw_result 8 :=-35.1953::sw_result 8:=-20.3008::
hw_result 9 :=-25.582::sw_result 9:=-19.8672::
//-----------------------------------------------
Prediction :0
//-----------------------------------------------
outer loop Started
```

```
hw_result 0 :=-9.9375::sw_result 0:=-7.22656::
hw_result 1 :=-9.17969::sw_result 1:=-11.6719::
hw_result 2 :=-3.06641::sw_result 2:=-4.17969::
hw_result 3 :=-3.54688::sw_result 3:=-6.91797::
hw_result 4 :=-10.4375::sw_result 4:=-12.1523::
hw_result 5 :=-6.39453::sw_result 5:=-9.49219::
hw_result 6 :=-13.332::sw_result 6:=-17.4766::
hw_result 7 :=7.01563::sw_result 7:=4.63281::
hw_result 8 :=-12.1367::sw_result 8:=-9.41406::
hw_result 9 :=-5.78516::sw_result 9:=-5.80078::
//---------------------------------------------------
Prediction :7
//---------------------------------------------------
outer loop Started
hw_result 0 :=-18.7891::sw_result 0:=-20.9922::
hw_result 1 :=-12.8828::sw_result 1:=-26.5313::
hw_result 2 :=-11.2695::sw_result 2:=-15.707::
hw_result 3 :=-8.44141::sw_result 3:=-15.8867::
hw_result 4 :=-0.941406::sw_result 4:=1::
hw_result 5 :=-5.52734::sw_result 5:=-9.44141::
hw_result 6 :=-10.3984::sw_result 6:=-16.668::
hw_result 7 :=-4.95313::sw_result 7:=-13.5117::
hw_result 8 :=-13.4805::sw_result 8:=-16.4141::
hw_result 9 :=-6.01953::sw_result 9:=-8.71484::
//---------------------------------------------------
Prediction :4
//---------------------------------------------------
outer loop Started
hw_result 0 :=1.32031::sw_result 0:=2.54688::
hw_result 1 :=-18.5703::sw_result 1:=-38.3594::
hw_result 2 :=-6.10156::sw_result 2:=-9.96484::
hw_result 3 :=-15.6055::sw_result 3:=-14.2344::
hw_result 4 :=-24.1367::sw_result 4:=-27.1914::
hw_result 5 :=-8.66016::sw_result 5:=-13.418::
hw_result 6 :=-11.3516::sw_result 6:=-14.4766::
hw_result 7 :=-16.6992::sw_result 7:=-12.6602::
hw_result 8 :=-21.3047::sw_result 8:=-13.0078::
hw_result 9 :=-19.5664::sw_result 9:=-18.6758::
//---------------------------------------------------
Prediction :0
//---------------------------------------------------
outer loop Started
hw_result 0 :=-13.1445::sw_result 0:=-27.6953::
hw_result 1 :=1.84766::sw_result 1:=1.13281::
hw_result 2 :=-8.375::sw_result 2:=-11.1914::
hw_result 3 :=-4.74609::sw_result 3:=-9.50781::
hw_result 4 :=-10.168::sw_result 4:=-9.23438::
hw_result 5 :=-5.75::sw_result 5:=-7.90234::
hw_result 6 :=-10.6797::sw_result 6:=-9.86328::
hw_result 7 :=-6.99219::sw_result 7:=-14.6094::
hw_result 8 :=-6.72656::sw_result 8:=-7.53125::
hw_result 9 :=-15.0195::sw_result 9:=-13.3008::
//---------------------------------------------------
Prediction :1
//---------------------------------------------------
```

Correctly Identified :10/10

```
////////////////////////////////////////////////////////////////////////////////
//
```

```
// Inter-Transaction Progress: Completed Transaction / Total Transaction
// Intra-Transaction Progress: Measured Latency / Latency Estimation * 100%
//
// RTL Simulation : "Inter-Transaction Progress" ["Intra-Transaction Progress"] @
"Simulation Time"
////////////////////////////////////////////////////////////////////////////////////
//
// RTL Simulation : 0 / 10 [0.00%] @ "125000"
// RTL Simulation : 1 / 10 [0.00%] @ "779465000"
// RTL Simulation : 2 / 10 [0.00%] @ "1558815000"
// RTL Simulation : 3 / 10 [0.00%] @ "2338165000"
// RTL Simulation : 4 / 10 [0.00%] @ "3117515000"
// RTL Simulation : 5 / 10 [0.00%] @ "3896865000"
// RTL Simulation : 6 / 10 [0.00%] @ "4676215000"
// RTL Simulation : 7 / 10 [0.00%] @ "5455565000"
// RTL Simulation : 8 / 10 [0.00%] @ "6234915000"
// RTL Simulation : 9 / 10 [0.00%] @ "7014265000"
// RTL Simulation : 10 / 10 [100.00%] @ "7793615000"
////////////////////////////////////////////////////////////////////////////////////
//
$finish called at time : 7793655 ns : File
"C:/Users/mghnv/Documents/HAoML/Project/MLP_MNIST/MLP_MNIST/solution_pipe_full_16/
sim/verilog/MLP_MNIST.autotb.v" Line 437
run: Time (s): cpu = 00:00:00 ; elapsed = 00:00:33 . Memory (MB): peak = 211.887 ;
gain = 0.000
## quit
INFO: [Common 17-206] Exiting xsim at Sat Jun  6 21:39:54 2020...
INFO: [COSIM 212-316] Starting C post checking ...
started
outer loop Started
hw_result 0 :=-20.7813::sw_result 0:=-26.4141::
hw_result 1 :=-12.2656::sw_result 1:=-11.3281::
hw_result 2 :=-17.4609::sw_result 2:=-15.5078::
hw_result 3 :=4.80859::sw_result 3:=1.29688::
hw_result 4 :=-25.4219::sw_result 4:=-18.3477::
hw_result 5 :=-1.50391::sw_result 5:=-8.25391::
hw_result 6 :=-20.5273::sw_result 6:=-24.0313::
hw_result 7 :=-13.1445::sw_result 7:=-11.2031::
hw_result 8 :=-13.3086::sw_result 8:=-14.1406::
hw_result 9 :=-9.30078::sw_result 9:=-12.2383::
//--------------------------------------------------
Prediction :3
//--------------------------------------------------
outer loop Started
hw_result 0 :=-11.25::sw_result 0:=-21.0078::
hw_result 1 :=0.933594::sw_result 1:=0.203125::
hw_result 2 :=-8.51172::sw_result 2:=-9.95313::
hw_result 3 :=-3.37109::sw_result 3:=-9::
hw_result 4 :=-7.57422::sw_result 4:=-5.75781::
hw_result 5 :=-5.42188::sw_result 5:=-9.11719::
hw_result 6 :=-10.4727::sw_result 6:=-8.75391::
hw_result 7 :=-2.46875::sw_result 7:=-7.1875::
hw_result 8 :=-6.93359::sw_result 8:=-5.89063::
hw_result 9 :=-6.42188::sw_result 9:=-7.09766::
//--------------------------------------------------
Prediction :1
//--------------------------------------------------
outer loop Started
hw_result 0 :=-27.1094::sw_result 0:=-26.0977::
```

```
hw_result 1 :=-7.84766::sw_result 1:=-9.14063::
hw_result 2 :=-8.71484::sw_result 2:=-9.96094::
hw_result 3 :=6.47656::sw_result 3:=5.73438::
hw_result 4 :=-24.2891::sw_result 4:=-18.293::
hw_result 5 :=-4.48047::sw_result 5:=-4.51953::
hw_result 6 :=-14.8359::sw_result 6:=-21.0078::
hw_result 7 :=-14.3203::sw_result 7:=-16.6563::
hw_result 8 :=-5.66406::sw_result 8:=-8.66016::
hw_result 9 :=-19.8125::sw_result 9:=-16.4219::
//-----------------------------------------------
Prediction :3
//-----------------------------------------------
outer loop Started
hw_result 0 :=-6.39453::sw_result 0:=-15.7266::
hw_result 1 :=-21.5234::sw_result 1:=-48.707::
hw_result 2 :=-12.5547::sw_result 2:=-13.207::
hw_result 3 :=-16.332::sw_result 3:=-24.0977::
hw_result 4 :=-6.92578::sw_result 4:=-4.14844::
hw_result 5 :=-5.25781::sw_result 5:=-14.875::
hw_result 6 :=-5.79688::sw_result 6:=-12.7266::
hw_result 7 :=-9.57031::sw_result 7:=-20.2852::
hw_result 8 :=-21.5703::sw_result 8:=-21.832::
hw_result 9 :=-17.0977::sw_result 9:=-22.3945::
//-----------------------------------------------
Prediction :5
//-----------------------------------------------
outer loop Started
hw_result 0 :=-14.6797::sw_result 0:=-20.7031::
hw_result 1 :=-7.61328::sw_result 1:=-17.5039::
hw_result 2 :=-2.74219::sw_result 2:=-7.67969::
hw_result 3 :=-2.84766::sw_result 3:=-4.63672::
hw_result 4 :=-12.9219::sw_result 4:=-16.8867::
hw_result 5 :=-9.23438::sw_result 5:=-20.4219::
hw_result 6 :=-15.8516::sw_result 6:=-27.4531::
hw_result 7 :=7.30859::sw_result 7:=2.27344::
hw_result 8 :=-8.53906::sw_result 8:=-6.91797::
hw_result 9 :=-13.1406::sw_result 9:=-7.51172::
//-----------------------------------------------
Prediction :7
//-----------------------------------------------
outer loop Started
hw_result 0 :=-14.75::sw_result 0:=-32.918::
hw_result 1 :=-8.14063::sw_result 1:=-9.24609::
hw_result 2 :=5.14844::sw_result 2:=3.85547::
hw_result 3 :=-2.19922::sw_result 3:=-9.71875::
hw_result 4 :=-23.3281::sw_result 4:=-26.3633::
hw_result 5 :=-4.92969::sw_result 5:=-15.1484::
hw_result 6 :=-11.1328::sw_result 6:=-21.7422::
hw_result 7 :=-12.8984::sw_result 7:=-14.0586::
hw_result 8 :=-13.1953::sw_result 8:=-8.40234::
hw_result 9 :=-22.7031::sw_result 9:=-17.3359::
//-----------------------------------------------
Prediction :2
//-----------------------------------------------
outer loop Started
hw_result 0 :=-7.63672::sw_result 0:=-19.2891::
hw_result 1 :=-6.85938::sw_result 1:=-11.7891::
hw_result 2 :=-1.86719::sw_result 2:=-2.81641::
hw_result 3 :=-5.71875::sw_result 3:=-5.34766::
```

```
hw_result 4 :=-12.3633::sw_result 4:=-19.9063::
hw_result 5 :=-7.24219::sw_result 5:=-14.6836::
hw_result 6 :=-14.4492::sw_result 6:=-22.0469::
hw_result 7 :=4.60547::sw_result 7:=0.707031::
hw_result 8 :=-10.332::sw_result 8:=-9.38281::
hw_result 9 :=-8.66016::sw_result 9:=-10.1133::
//-----------------------------------------------
Prediction :7
//-----------------------------------------------
outer loop Started
hw_result 0 :=-13.4492::sw_result 0:=-26.8789::
hw_result 1 :=3.27344::sw_result 1:=0.972656::
hw_result 2 :=-9.59766::sw_result 2:=-13.6328::
hw_result 3 :=-5.53125::sw_result 3:=-12.5781::
hw_result 4 :=-8.12891::sw_result 4:=-9.12109::
hw_result 5 :=-7.01953::sw_result 5:=-11.6875::
hw_result 6 :=-11.7695::sw_result 6:=-9.32422::
hw_result 7 :=-2.07422::sw_result 7:=-11.6563::
hw_result 8 :=-8.09766::sw_result 8:=-6.28516::
hw_result 9 :=-12.2148::sw_result 9:=-10.9297::
//-----------------------------------------------
Prediction :1
//-----------------------------------------------
outer loop Started
hw_result 0 :=-14.7656::sw_result 0:=-17.0469::
hw_result 1 :=-1.84375::sw_result 1:=-6.49219::
hw_result 2 :=1.89063::sw_result 2:=1.44922::
hw_result 3 :=0.179688::sw_result 3:=-0.566406::
hw_result 4 :=-18.8086::sw_result 4:=-20.5078::
hw_result 5 :=-4.42578::sw_result 5:=-8.5625::
hw_result 6 :=-8.32813::sw_result 6:=-11.6406::
hw_result 7 :=-7.59766::sw_result 7:=-8.48828::
hw_result 8 :=-10.0781::sw_result 8:=-6.41797::
hw_result 9 :=-16.5039::sw_result 9:=-17.0547::
//-----------------------------------------------
Prediction :2
//-----------------------------------------------
outer loop Started
hw_result 0 :=-12.0508::sw_result 0:=-19.8203::
hw_result 1 :=2.57422::sw_result 1:=-0.519531::
hw_result 2 :=-11.5::sw_result 2:=-11.1953::
hw_result 3 :=-9.56641::sw_result 3:=-13.7891::
hw_result 4 :=-10.0703::sw_result 4:=-12.3281::
hw_result 5 :=-8.35938::sw_result 5:=-14.7734::
hw_result 6 :=-9.97656::sw_result 6:=-9.83984::
hw_result 7 :=-7.71484::sw_result 7:=-12.4375::
hw_result 8 :=-9.125::sw_result 8:=-7.81641::
hw_result 9 :=-14.5508::sw_result 9:=-13.8438::
//-----------------------------------------------
Prediction :1
//-----------------------------------------------
```

Correctly Identified :9/10

RESULTS

Here, we discuss about the performance at the hardware level. The pruned MLP was quantized to 16-bits and multiplier less optimization was applied to eliminate the need for hardware multipliers. The MLP was exported to (Zynq-7000) FPGA based hardware accelerator and the hardware performance of the optimized and unoptimized networks were evaluated:

Table 1 Hardware Results

Parameter	**Original MLP**	**Optimized MLP**
Latency (ms)	23.06	0.52
Mem Used (KB)	9600	432

Thus, we witness a significant computational advantage with the optimized MLP.

CONCLUSION

A unified framework for hardware -software codesign of MLP Neural network has been carried out successfully.