Small Scale Parameterized Inference Engine

Vishnu Bharadwaj
Shruti Narake
Saurabh Patil

accellera SYSTEMS INITIATIVE
intel
2019 DESIGN AND VERIFICATION DVCON CONFERENCE AND EXHIBITION INDIA

Contents

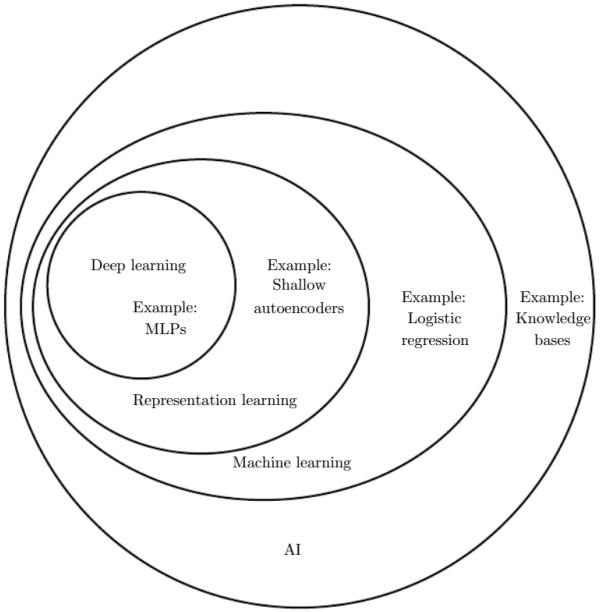
- AI and applications
- Neural Networks
- Software Modelling
- Hardware Implementation
- Conclusion
- Future Scope

- WHAT?
  - Building Basic block of Neural network.
- WHY?
  - To support building of different Neural network topologies.
  - To accelerate applications in mobile devices, OCR etc
- HOW?
  - By using Deep Neural Networks

- Problem Definition: To hardware accelerate AI based computationally intensive tasks.

Basic Terminologies

- AI is a field of computer science that gives computers the ability to learn without being explicitly programmed
- Key terms in AI:
  - Learning
  - Supervised & Unsupervised.
  - Cost function & Gradient descent

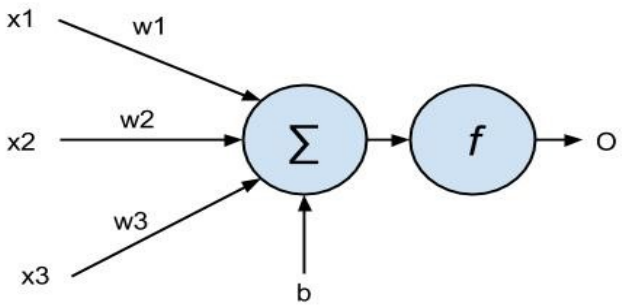


$$H_w = w_0 + x.w_1 + x^2.w_2 + ..$$

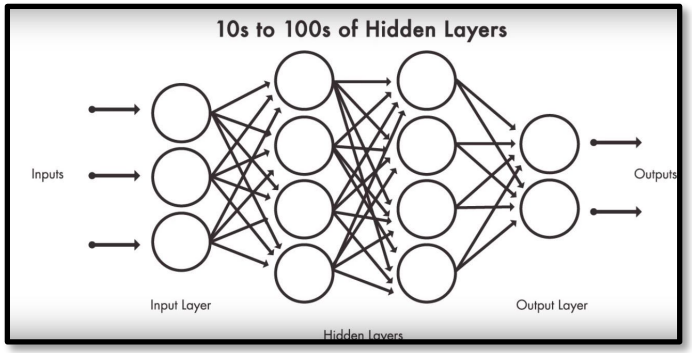
$$C_{MST}(W, B, S^r, E^r) = \frac{0.5}{m}\sum_j \left(H_j^L - E_j^r\right)^2$$

Key Terms in Neural Networks

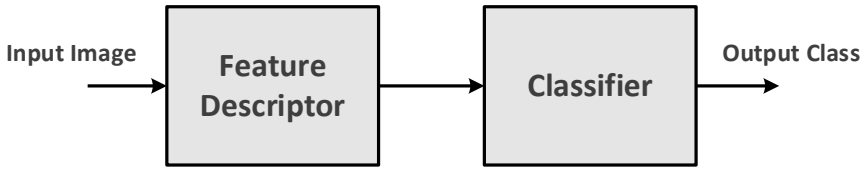
- Feed forward NN
- Backpropagation
- Activation function
- Perceptron
- Deep neural nets
- Convolution NN



$$f(w^t.x) = f\left(\sum_{i=1}^{n} w_i.x_i + b\right)$$

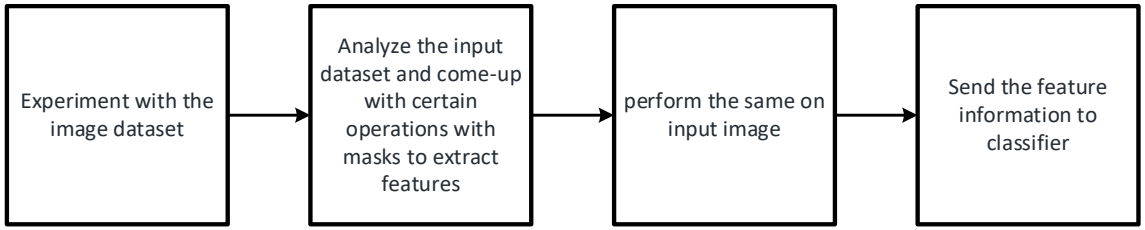


Computer Vision Algorithms

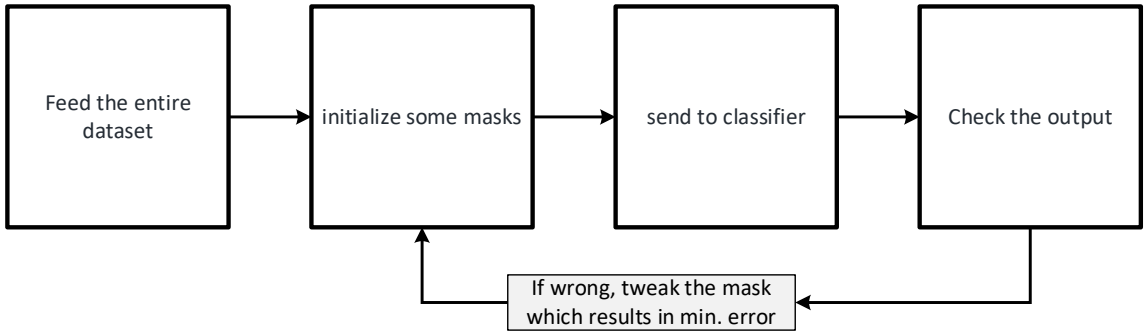


Computer Vision Recognition problem blocks

Traditional algorithm

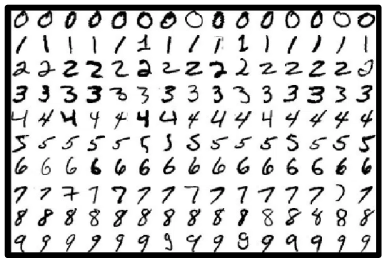


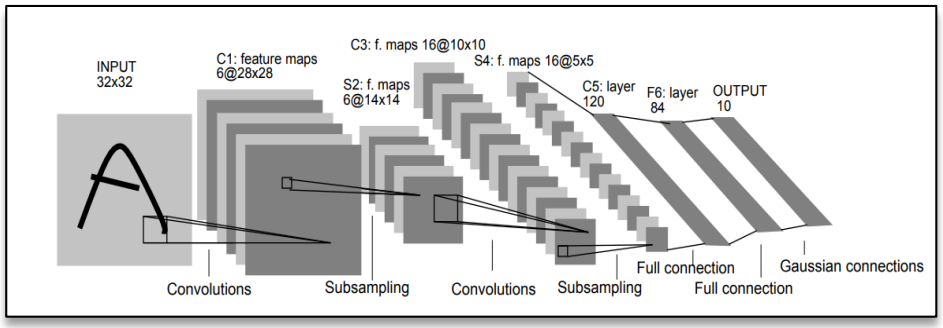
NN



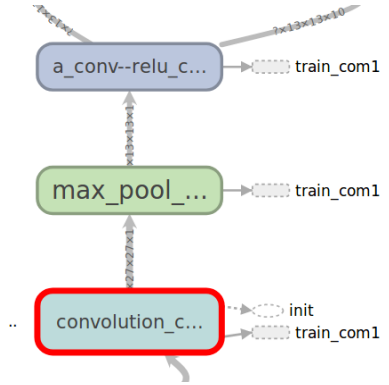
Topology and Datasets

- MNIST Database
- LeNet Topology






```
38 # CNN layers
39     for iter_i in range(1,num_cnn_layers+1,1):
40         "('string_%d'%iter_1) is the syntax to make it dynamic"
41         # convolution
42         with tf.name_scope('convolution_com1'):
43             com1_W_params["w{0}".format(iter_i)] = tf.Variable( tf.truncated_normal([
   {0}".format(iter_i)], com1_Fnum_params["fnum{0}".format(iter_i)] ], stddev=0.1) )
44
45             com1_B_params["b{0}".format(iter_i)] = tf.Variable( tf.constant(0.1, shape
46
47             com1_Z_params["z{0}".format(iter_i)] = tf.nn.conv2d( com1_A_params["a{0}".
   ID" ) + com1_B_params["b{0}".format(iter_i)]
48
49         # maxpool
50         if (com1_Maxpool_ops["pool{0}".format(iter_i)]) == True:
51             with tf.name_scope('max_pool_com1'):
52                 com1_Maxpool_params["max_pool{0}".format(iter_i)] = tf.nn.max_pool( co
   de{0}".format(iter_i)], padding='VALID' )
53
54
55         # relu
56         with tf.name_scope('a_conv--relu_com1'):
57             if (com1_Maxpool_ops["pool{0}".format(iter_i)]) == True:
58                 com1_A_params["a{0}".format(iter_i)] = tf.nn.relu( com1_Maxpool_params
59             else:
60                 com1_A_params["a{0}".format(iter_i)] = tf.nn.relu( com1_Z_params["z{0}
61
```

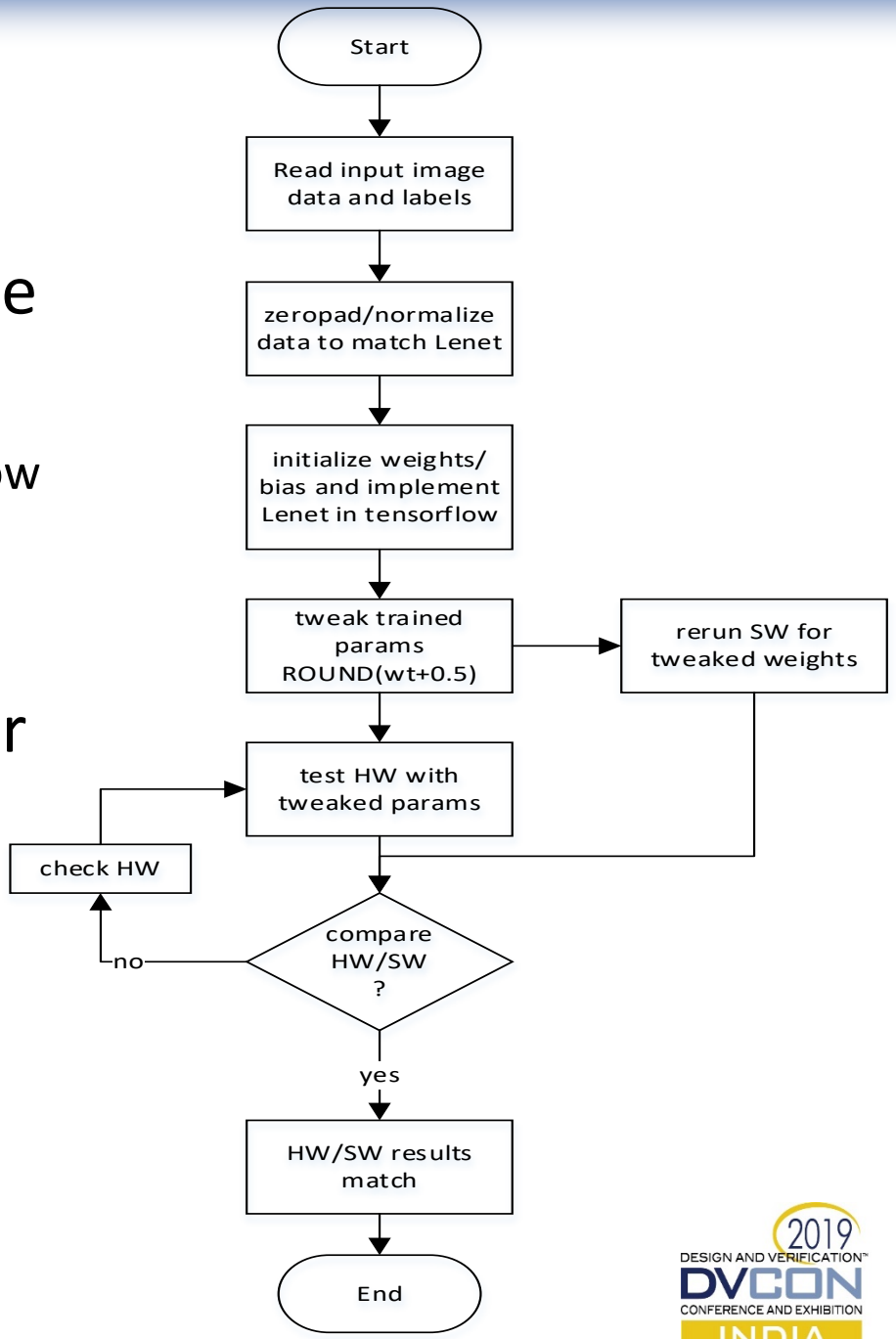


Implementation

- Training is **iterative**, inference is one-shot

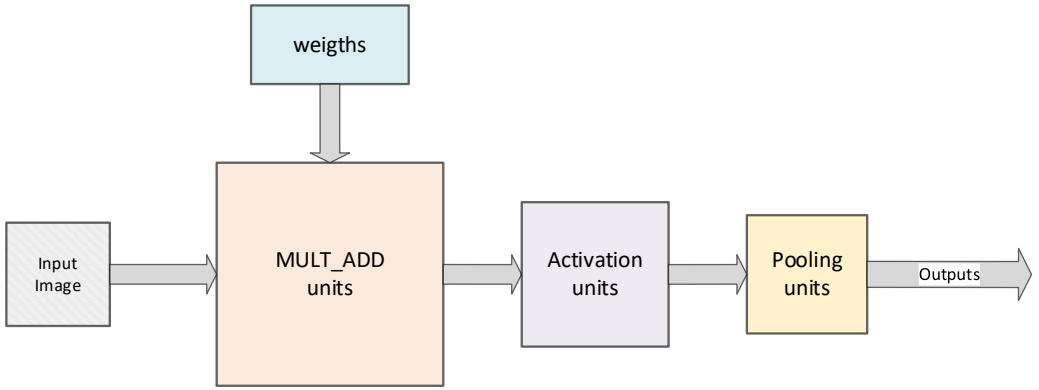
  Matlab -> Python -> Numpy -> Tensorflow

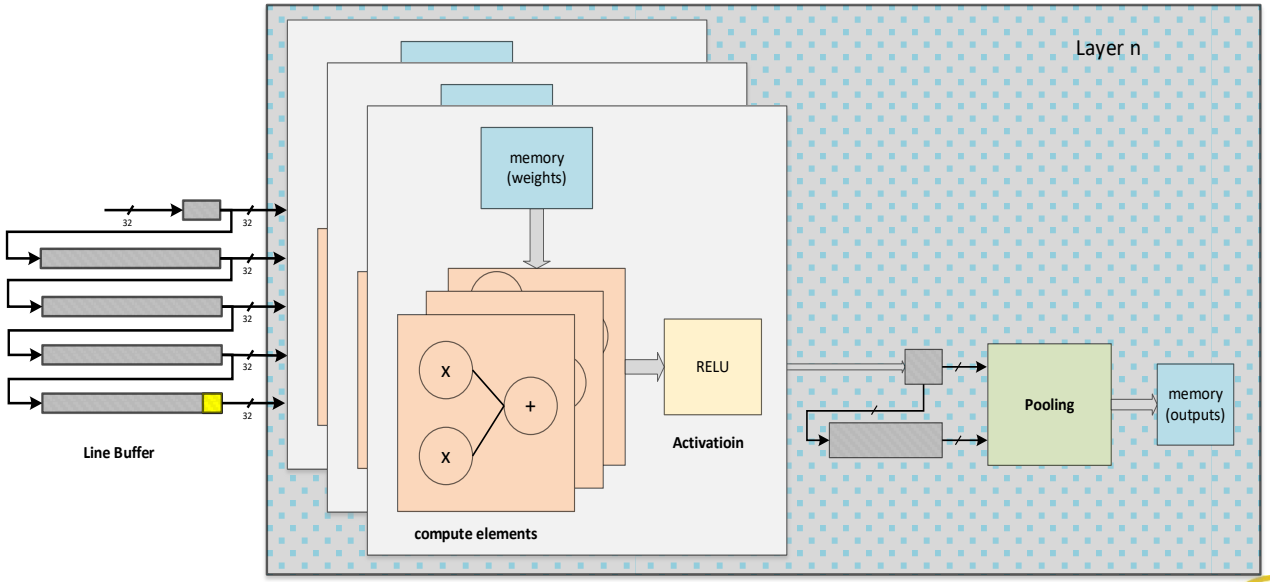
- Others: neon, Keras, torch
- Output Visualization – Tensor board



Hardware Architecture

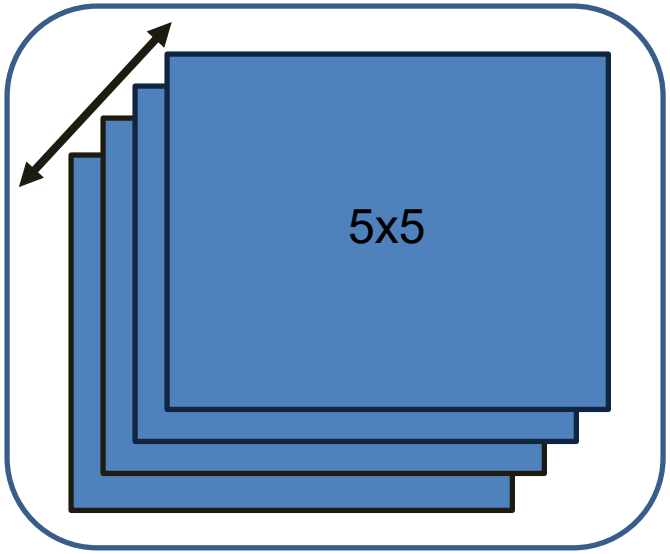
- Line Buffers
- Routing Matrix
- Mult Add
- Parallel Adder
- ReLU
- Pooling





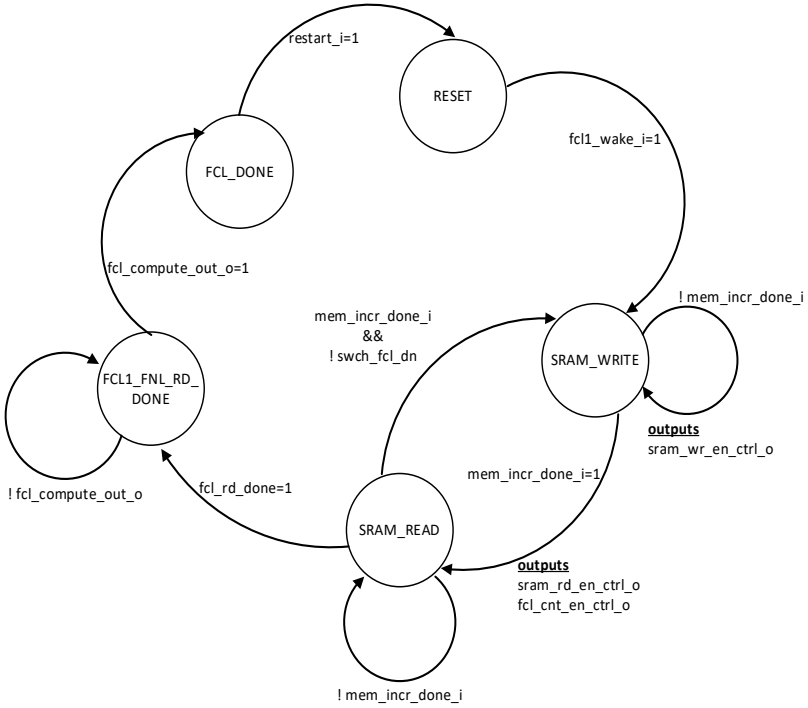
FCL Mapped as Convolution

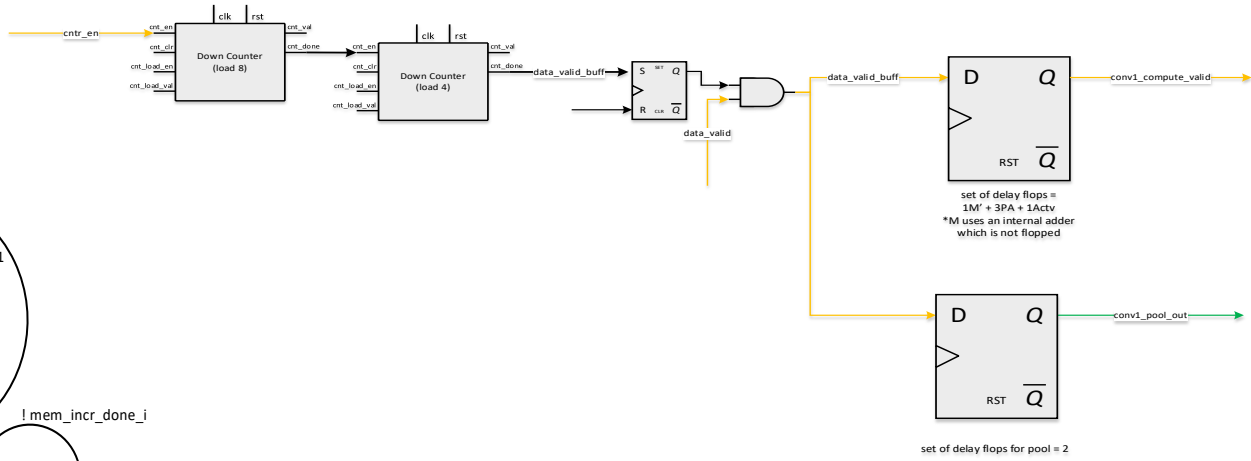
- Flattened 400 units -> CNN
- Keep the filter size = 5x5
- Number of filters = 16
- Output volume formed is [1x120]

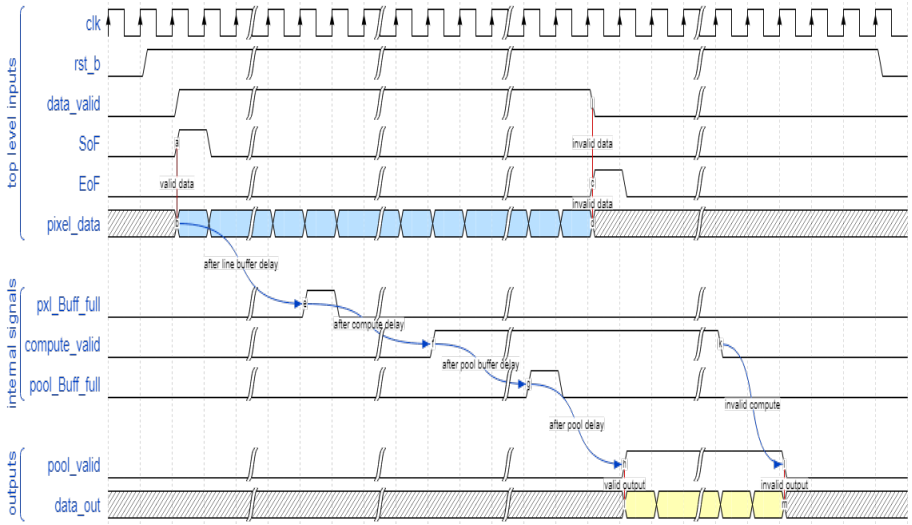


120 sets of 16 such filters

Control flow





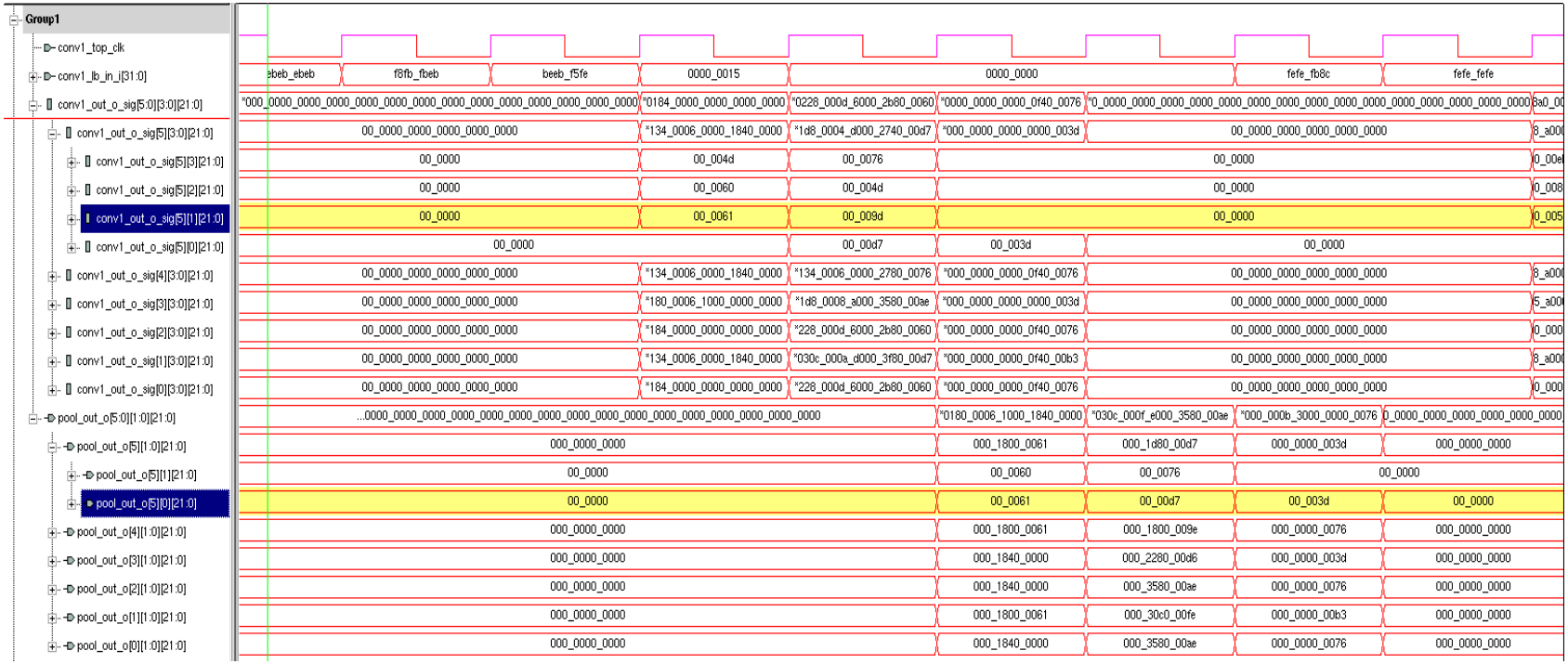


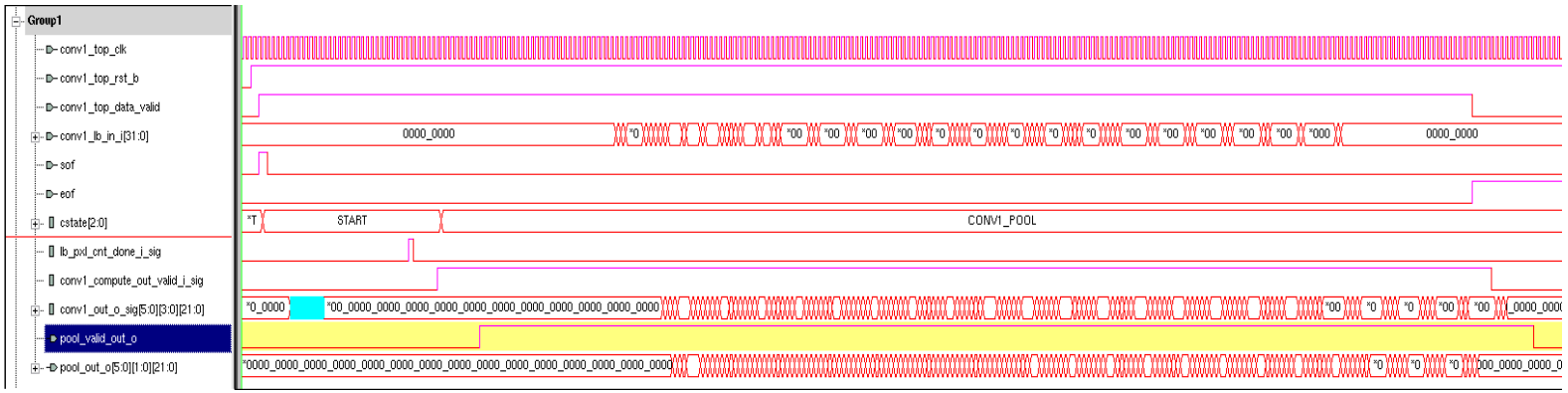
Results

RTL Parameter	Description
NUM_STRIDE_LEN	No. of parallel convolution operation
NUM_FILTER	No. of filters in the layer
LB_NUM_SHIFT_CONV	Length of line buffer for convolution
LB_NUM_TAPS_CONV	No. of outputs from buffer for convolution
LB_NUM_SHIFT_POOL	Length of line buffer for pooling
LB_NUM_TAPS_POOL	No. of outputs from line buffer for pooling
PIXEL_WIDTH	Width of a pixel before/after scaling

Parameter	Value
No. of Combinational cells	79827
No. of sequential cells	49183
Combinational area	21877 um^3
Non-combination area	31120 um^3
Total Dynamic power	19.527 mW
Frequency	400 MHz
Minimum slack	**+ 556.87 ps**

Contd…


Group1
conv1_top_clk
conv1_lb_in_i[31:0]
conv1_out_o_sig[5:0][3:0][21:0]
conv1_out_o_sig[5][3:0][21:0]
conv1_out_o_sig[5][3][21:0]
conv1_out_o_sig[5][2][21:0]
conv1_out_o_sig[5][1][21:0]
conv1_out_o_sig[5][0][21:0]
conv1_out_o_sig[4][3:0][21:0]
conv1_out_o_sig[3][3:0][21:0]
conv1_out_o_sig[2][3:0][21:0]
conv1_out_o_sig[1][3:0][21:0]
conv1_out_o_sig[0][3:0][21:0]
pool_out_o[5:0][1:0][21:0]
pool_out_o[5][1:0][21:0]
pool_out_o[5][1][21:0]
pool_out_o[5][0][21:0]
pool_out_o[4][1:0][21:0]
pool_out_o[3][1:0][21:0]
pool_out_o[2][1:0][21:0]
pool_out_o[1][1:0][21:0]
pool_out_o[0][1:0][21:0]


Group1
conv1_top_clk
conv1_top_rst_b
conv1_top_data_valid
conv1_lb_in_i[31:0]
sof
eof
cstate[2:0]
START
CONV1_POOL
lb_pxl_cnt_done_i_sig
conv1_compute_out_valid_i_sig
conv1_out_o_sig[5:0][3:0][21:0]
pool_valid_out_o
pool_out_o[5:0][1:0][21:0]

Conclusion and Future Scope

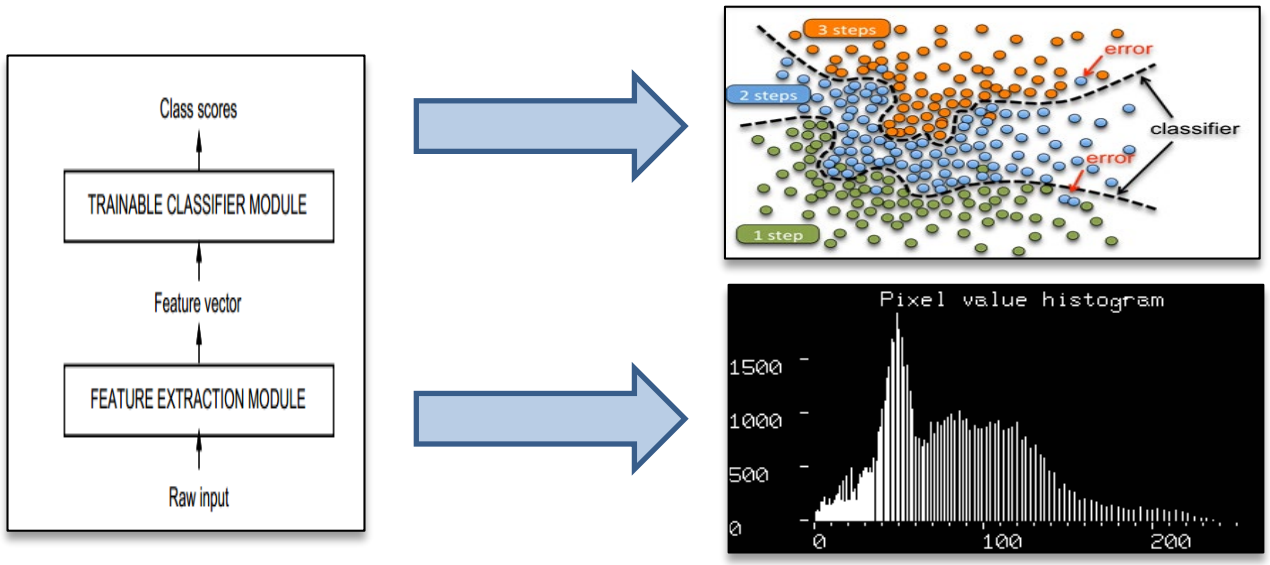
- CNN & FCL model for hardware
- This design gives flexibility to implement any topology because of the scalability feature being added.
- The novel approach of implementing vectorization, parallel and pipelined design adds to performance in terms of speed

- Fixed point implementation for higher accuracy
- Gate count optimizations

Questions?

Backup slides

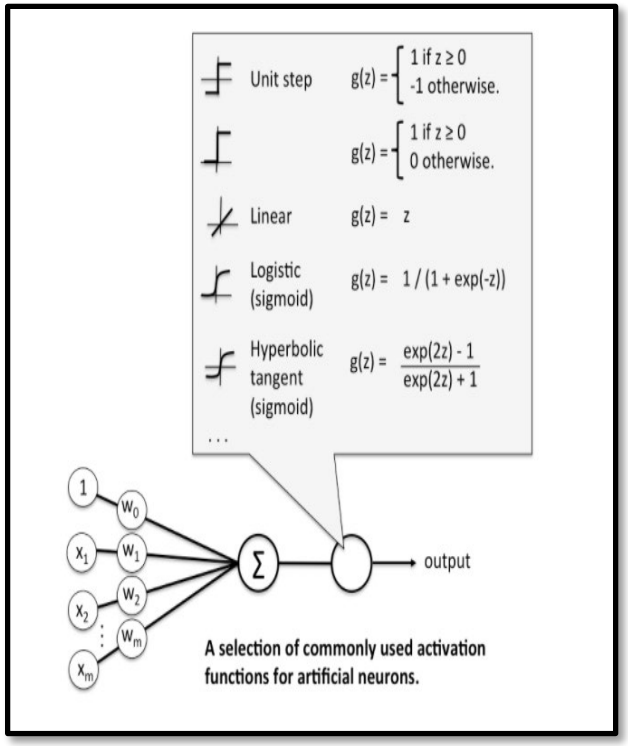
Traditional Algorithm

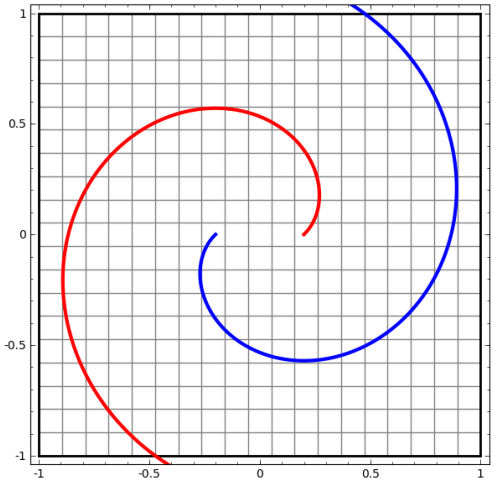


- Handcrafting the features is time consuming.

Neural Network -Perceptron

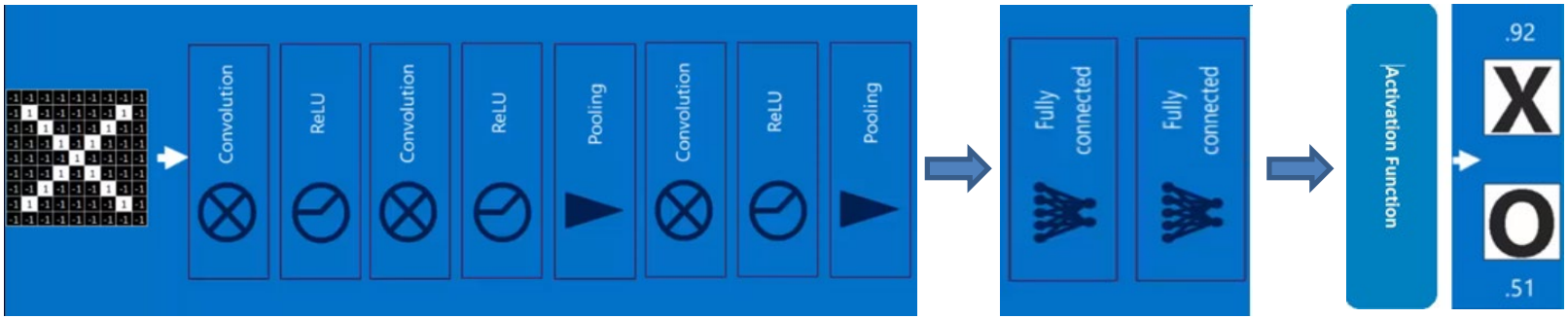
Perceptron- a basic neural network building block





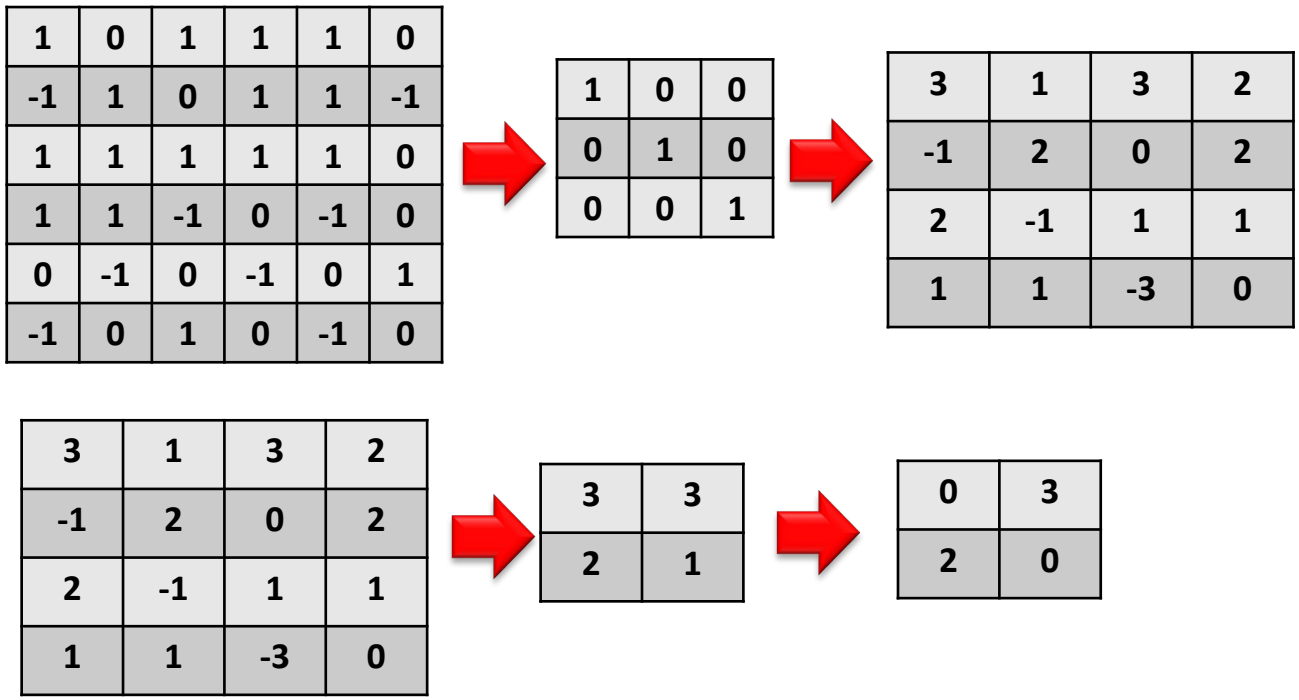
MLP can classify Non linearly separable functions.

Topology

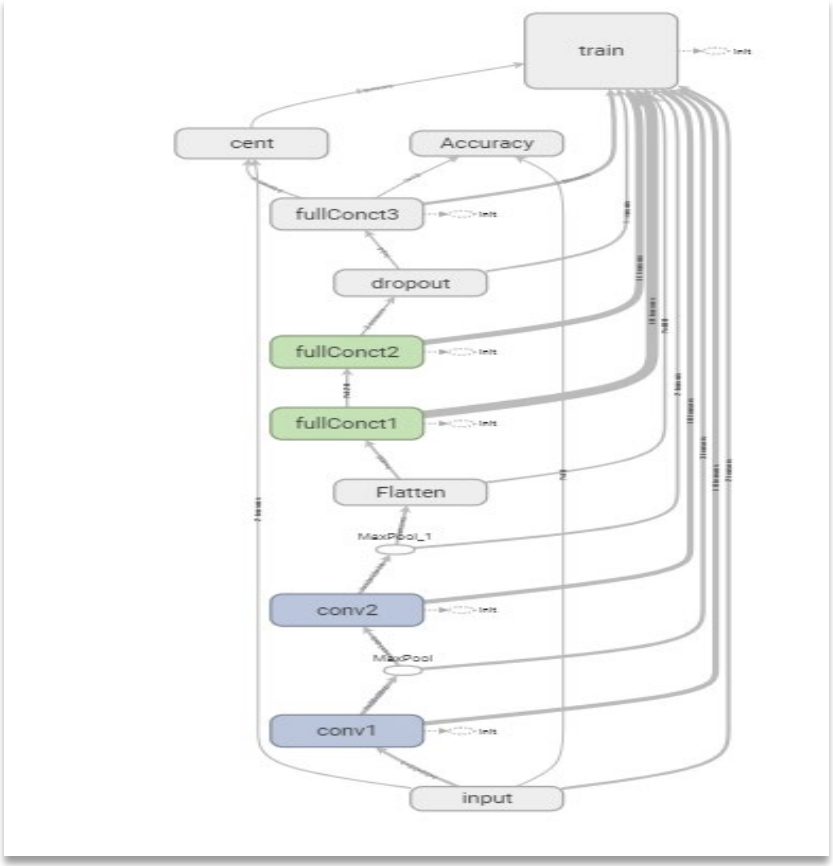


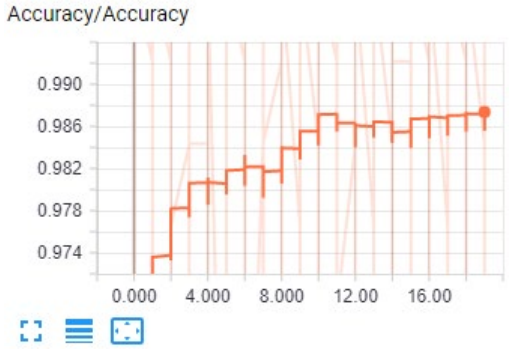
- Neural Network used in Feature extraction are called Convolutional Neural Network(CNN)
- MLP's used in Classification is Fully Connected Neural Networks(FCNN) .

Convolution, Pooling and Activation



Tensor Board Visualisation





Memory Requirement

LeNet			Weight size	Bias size (No. of Filters)
CNN 1	input data size	32*32*1	5*5*1*6 =150	6
	kernal (filter) size	5*5		
	kernal (filter) stride	1		
	No. of Kernals (filters)	6		
	dimension before pool	28*28*6		
	Max pool	yes		
	pool dimension	2x2		
	Pool stride	2		
	Activation	RELU		
	out dimension	14*14*6		
CNN 2	input data size	14*14*6	5*5*6*16=2400	16
	kernal (filter) size	5*5		
	kernal (filter) stride	1		
	No. of Kernals (filters)	16		
	dimension before pool	10*10*16		
	Max pool	yes		
	pool dimension	2*2		
	Pool stride	2		
	Activation	RELU		
	out dimension	5*5*16		

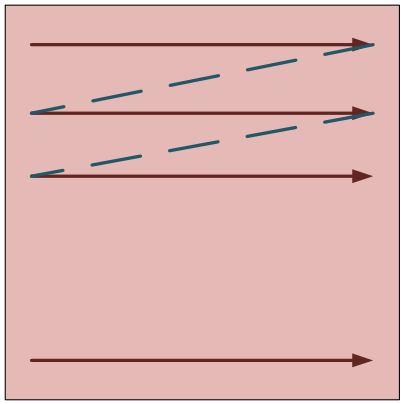
FC 1`	input data size	5*5*16	120*400=48000	120
	fc neurons	120		
	Activation	RELU		
FC 2	input data size	120	84*120=10080	84
	fc neurons	84		
	Activation	RELU		
Soft max	input data size	84	10*84=840	10
	fc neurons	10		
			Total = 61470	Total = 246
			61470 values ~= 120.058KB if 16 bits	

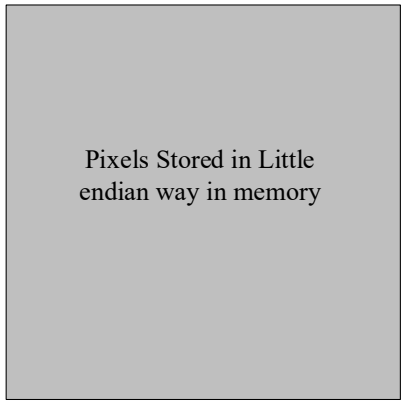
Software Results

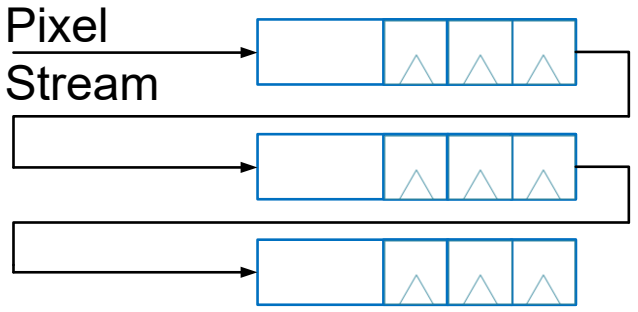
HyperParameters	Values
Epoch	20
Batchsize	128
Dropout	0.8
Learning rate	0.001

TOPOLOGY	VALIDATION SET ACCURACY	TESTING SET ACCURACY
LeNet	99.2	98.6
DreamNet	99	97.3
LeNet for Universal OCR	98.1	97.6

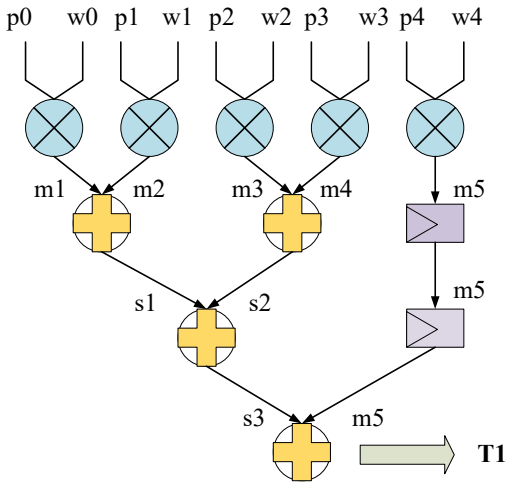
Hardware Components

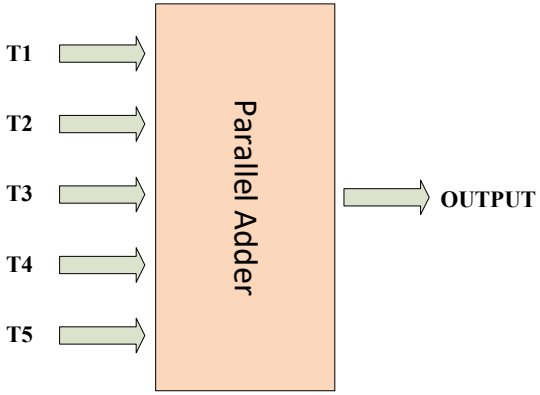






=Pixel
Register





Latency of mult add
- 3 cycles

Latency of parallel add
- 3 cycles

Hardware Results