PROJECT REPORT

DEEP NEURAL NETWORK

SAI CHARAN REDDY PISATI

200490526

Name: Sai Charan Reddy Pisati
Unity Id: spisati2
Student Id: 200490526

Delay:30920.4ns
Clock period:5.9ns
#Cycles:4908
#Cycles input1=3554
#Cycles input2=1354

Logic Area:
$18091.1918\mu m^2$

MEMORY=N/A

1/(delay.area)
$= 0.178767151 \times 10^{-8} \mu m^{-2} ns^{-1}$

Delay(TA provided example) TA to complete

1/delay.area(TA)

Abstract:

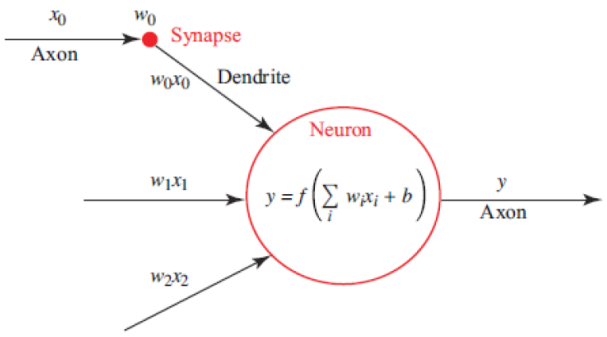
Convolutional Neural Networks are designed to detect the real time images. The aim is to design the model using Verilog language at the hardware level. The purpose of it is to improve storage and computation, and serves as a technique for deep models on resource-limited devices. The project intends to develop a multi-stage neural network which performs convolution and ReLu activation function and the max pooling operation on the inputs provided from SRAM Memory unit.The model takes the NXN input matrix as input each of size 8-bit, weight matrix kernel of 8-bit size. The operation is performed on input and kernel matrix then transferred to the ReLu function, is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It is being used because a model that uses it is easier to train and often achieves better performance. The model follows a 18 stage Finite State Machine, following the data reading from addresses and writing back the convoluted output to Output SRAM'S. The hardware is designed to meet the hold and setup violations, minimal area and slack. The outputs are verified with ModelSim and synthesis is done using Synopsys Design Vision.

Contents

1.Introduction

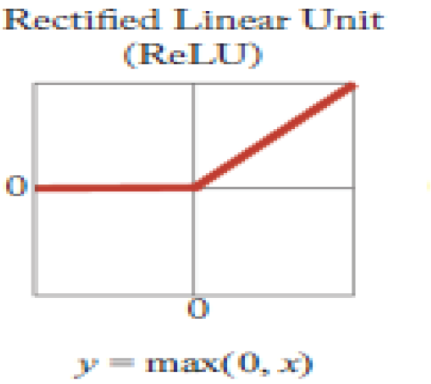
A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image, and be able to differentiate one from the other.

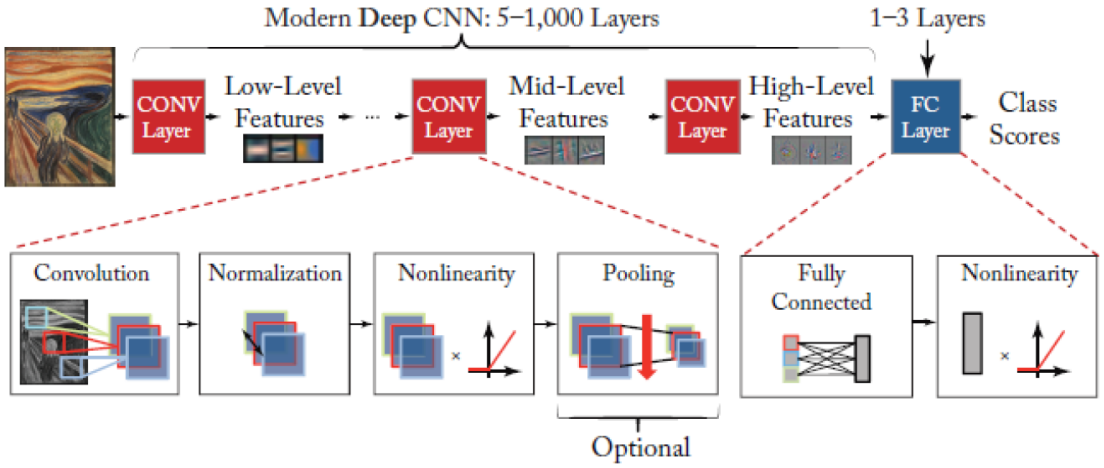
An ANN is a multi-level network that can perform inference. It's composed, in part of a set of neurons per the figure below. x,w,f() are the activation, weights and nonlinear functions.



The input to the neuron is the weighted sum of inputs Σ w_i*x_i, where x_i are the inputs and w_i are the weights. In this project, all variables are 8-bit signed integers. x_i can only take on positive values. Computations are to saturate, not overflow. With 8-bits you can represent integers from -128 to +127 digital. +127 is represented as 0111_1111 or 7F.

The output of the sum of weights is then passed to an activation function f. This is a nonlinear function and represents an important reason why ANNs are so effective. We will use the ReLu activation function, illustrated below

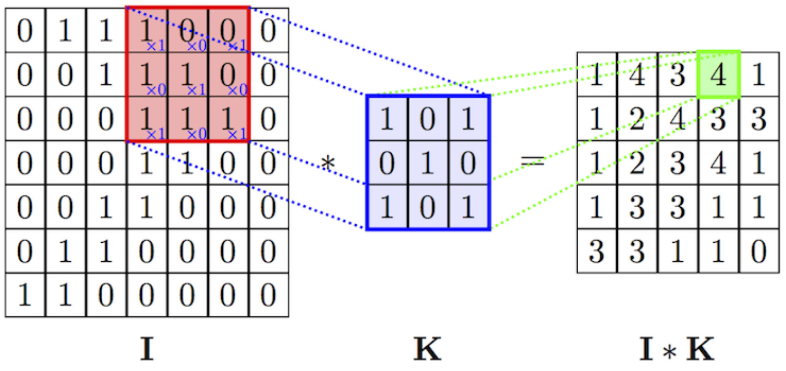




In a convolutional neural network layer, the weight matrix, called a kernel, is smaller than the input matrix, and an output matrix is produced with a size smaller than the input.

In a convolutional stage the kernel is multiplied by a subrange of the input to produce the feature map. For example, using row major ordering, element [0,1] of the feature map is produced by the following computation

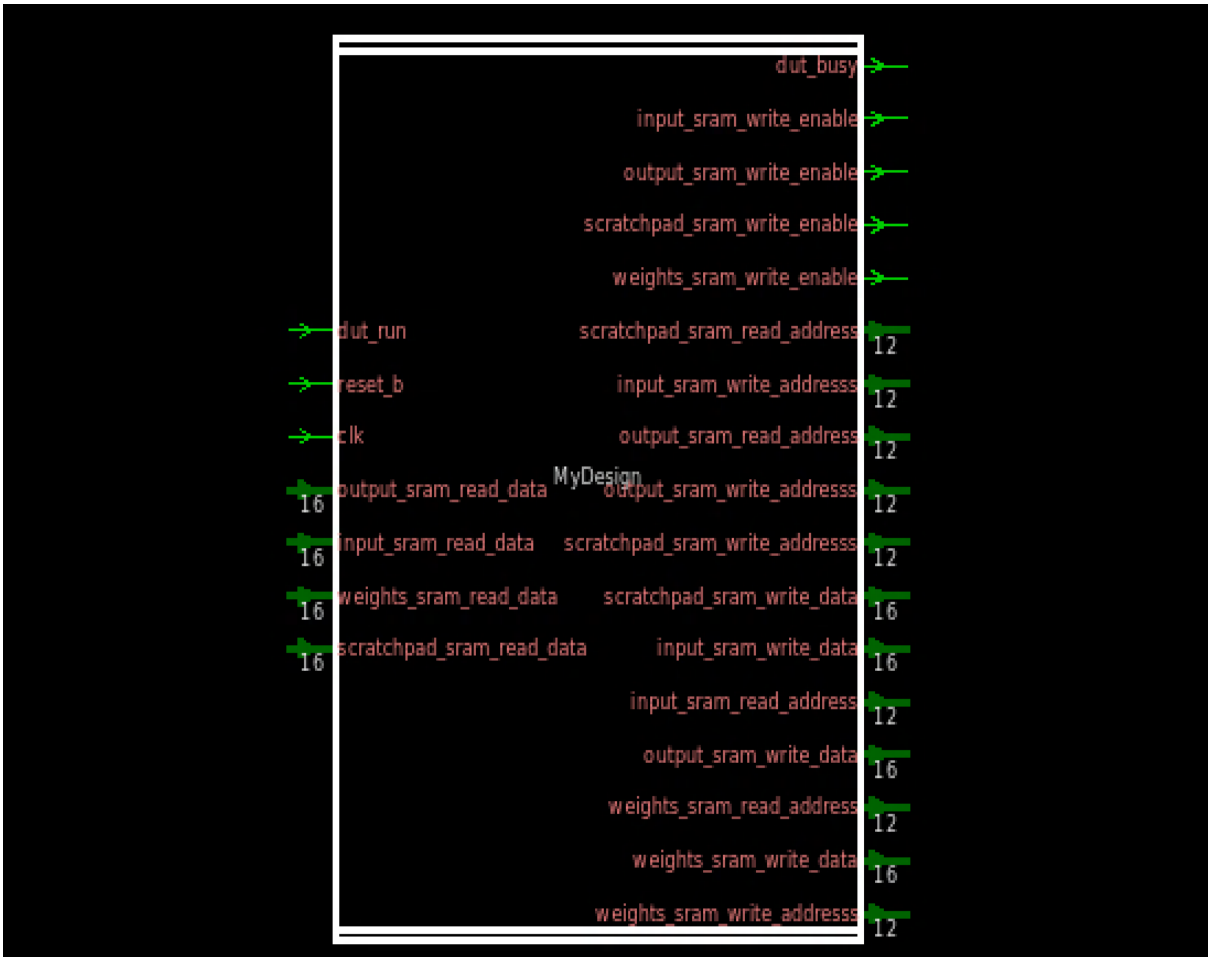
F[0,1] = f(i[0,1]*k[0,0]+ i[0,2]*k[0,1] + i[0,3]*k[0,2] + .. i[2,3]*k[2,2])



The further sections elaborate about the design, architecture, technical implementation, FSM, schematic and results achieved

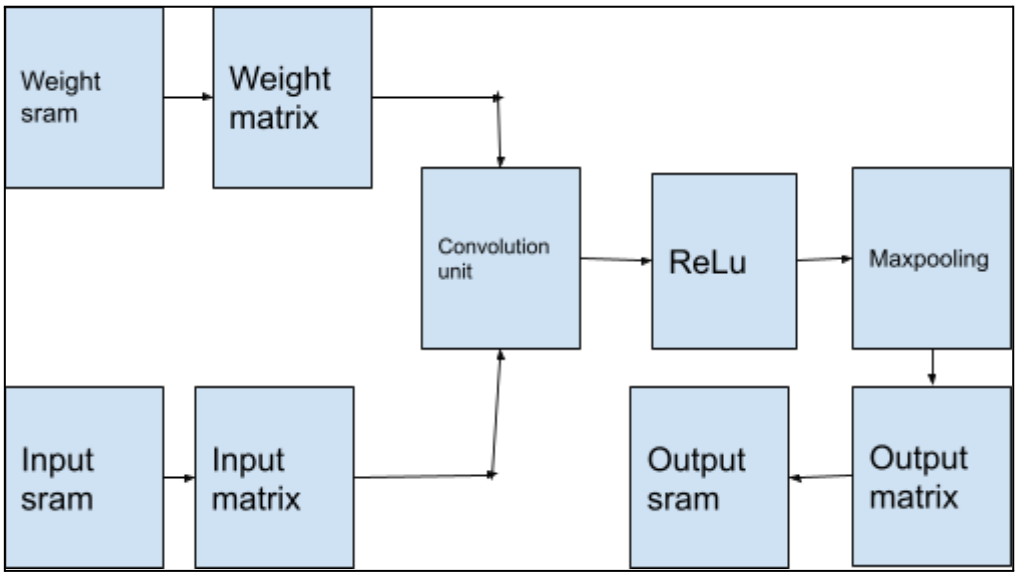
2.Micro Architecture

“Design before coding “ approach has been used while designing the model. The design consists of Input SRAM, Weight SRAM’s, which provide the required inputs to the model, higher byte and lower byte are the values stored in the matrices. The interface consists of the read_address, read_data for both the SRAM’s. The information passed to the Multiply & Adder units, stored in the accumulator then written to the output SRAM.



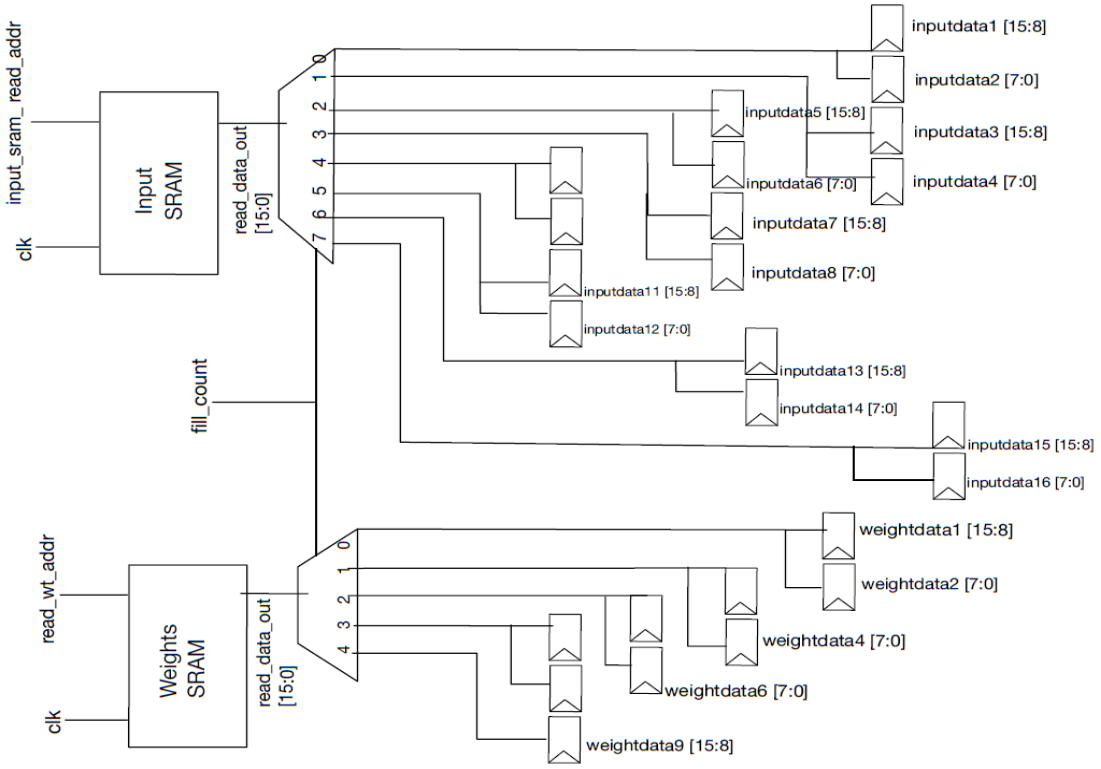
Design’s I/O ports illustration.

High Level Design

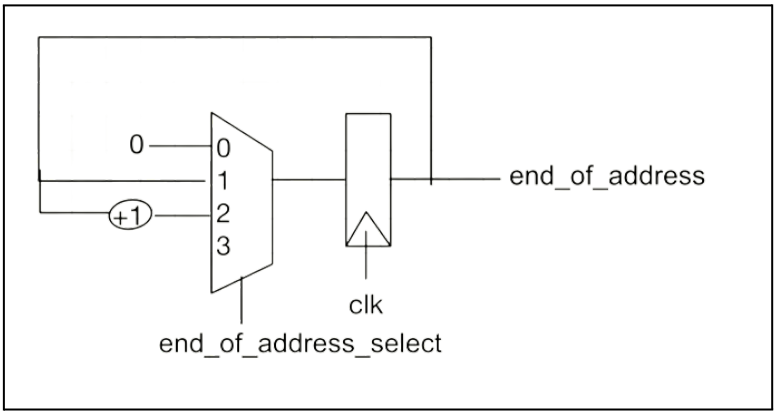


Algorithm

The algorithm implemented is, we took 4x4 input from NxN input address and 3x3 weight matrices from KxK weight addresses are fetched to the registers. Then 3x3 inputs are taken in the order then multiplied with the corresponding weights, then added to the adder unit. I used control signals like rc and cc to keep the counts of the rows and counts when I have to iterate. We need to send the outputs of multiply & add units to the ReLu unit. We use two relus for two multiply & adder units. Two max pooling units which filter out the clutter. Gives the maximum value of the window. Then, we write the data to the output SRAM. We enable the write enable signal when we need to write the data to the addresses. We store the two relu outputs on the each address. r+size/2 formula to iterated over the rows and counts to move around. The architecture shows the data flow from the input address and weight address to the required registers. The data is stored in the registers to hold the data and then transferred to the multiply & add units for convolution.



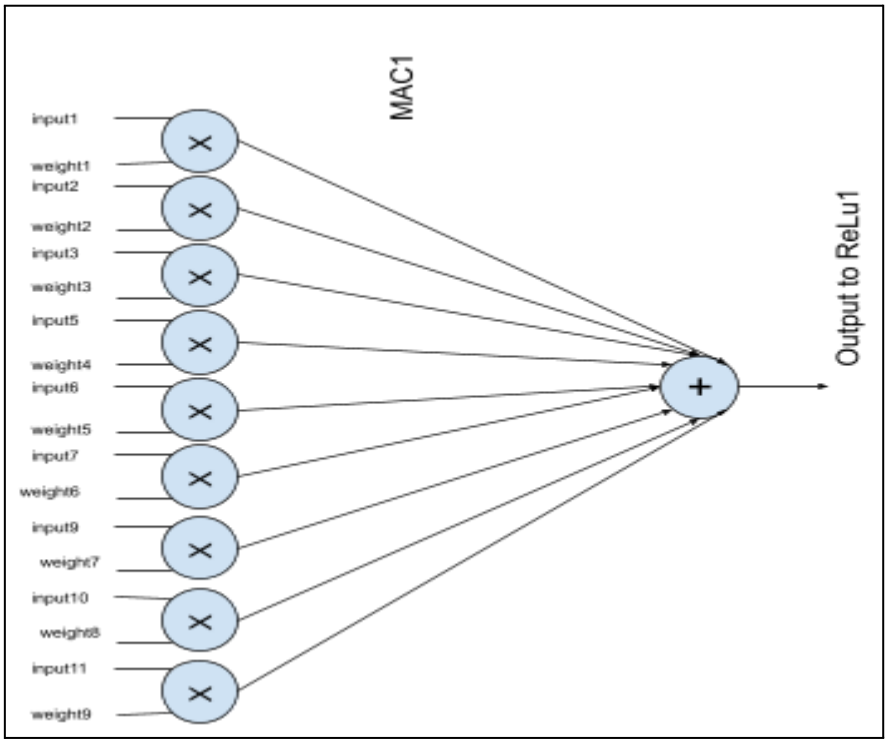
The select lines decides the state in s11 which enables the writing data to the SRAM for output

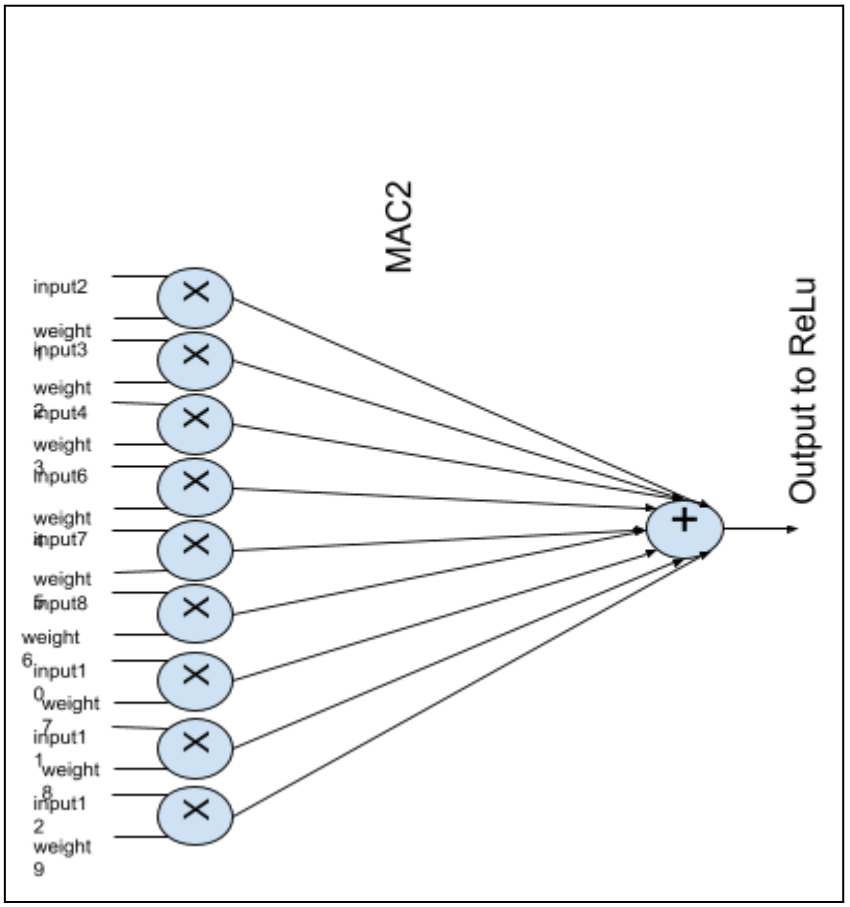


The block diagram writes the data to output SRAM address with a control signal which tells it to start writing.

Convolution Units

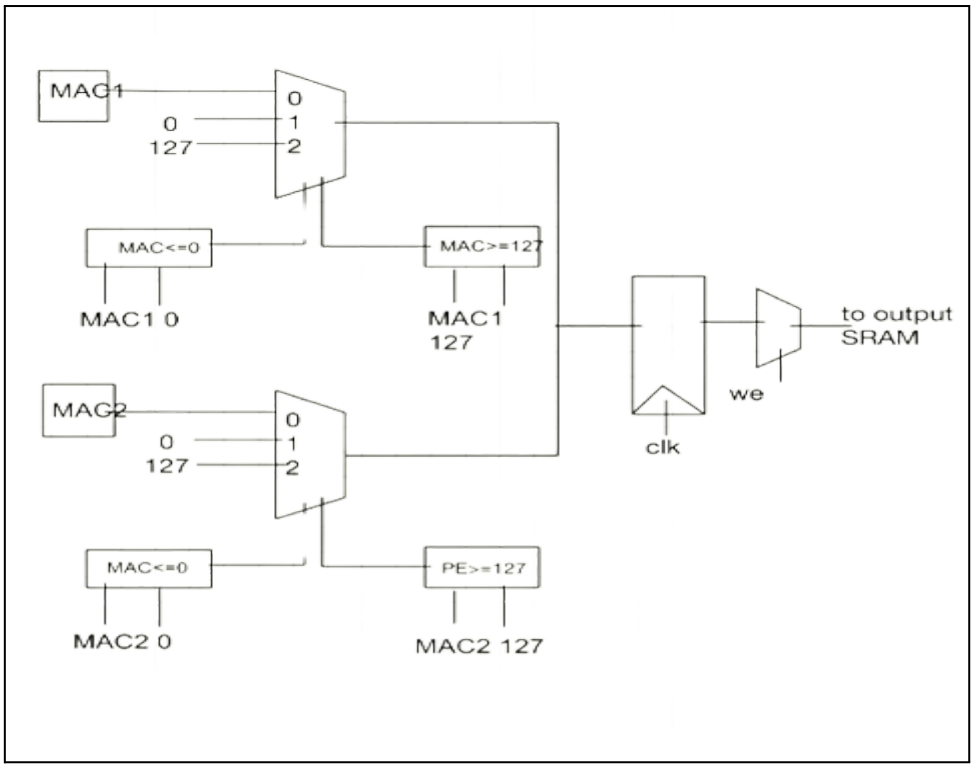
Two Multiply & Adder units are used for convolution using the algorithm.



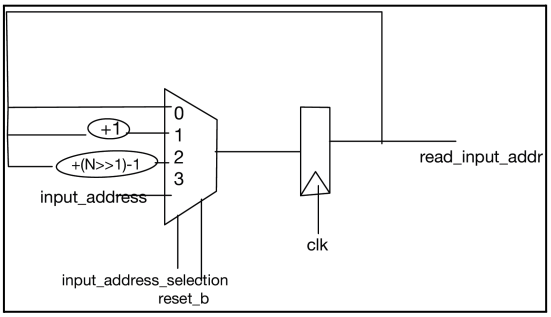


ReLu units

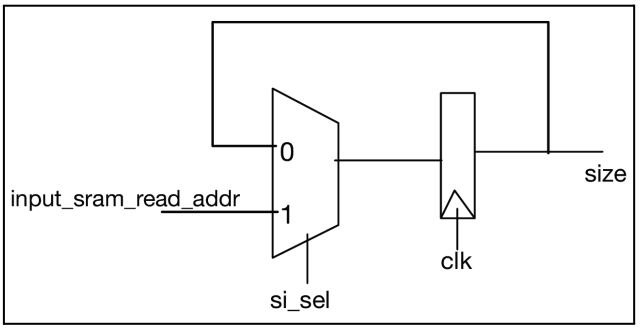
Convolved outputs are passed to the ReLu activation function, when convolved output is greater than 127,it saturates, when the values are negative then output is zero, when input is between the extremes then remains the same.



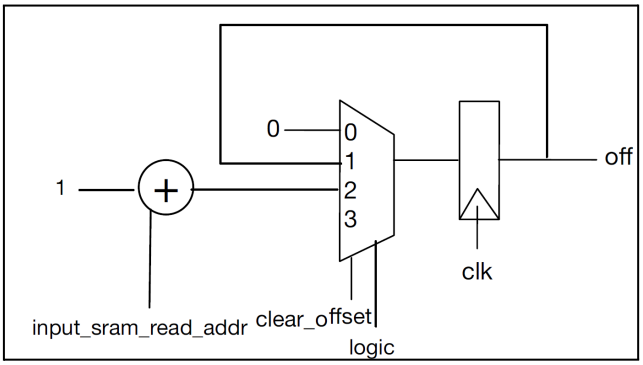
Select lines for the input address and shifting

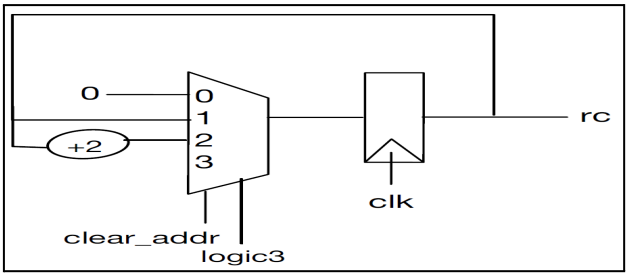


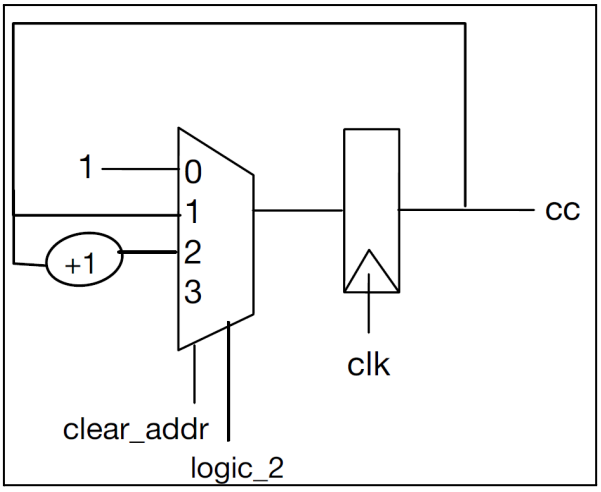
Block diagram describes the size taken from the firstaddress and has a control signal to initiate.

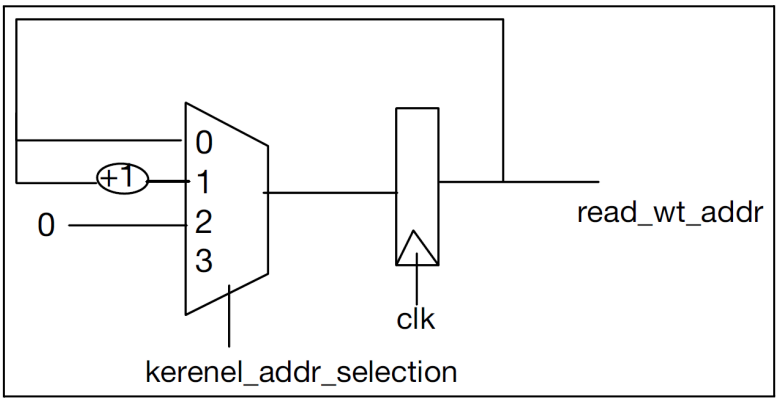


Select line clear_offset, which controls taking the input of the next image. It avoids over writing the current data, and starting from zero.

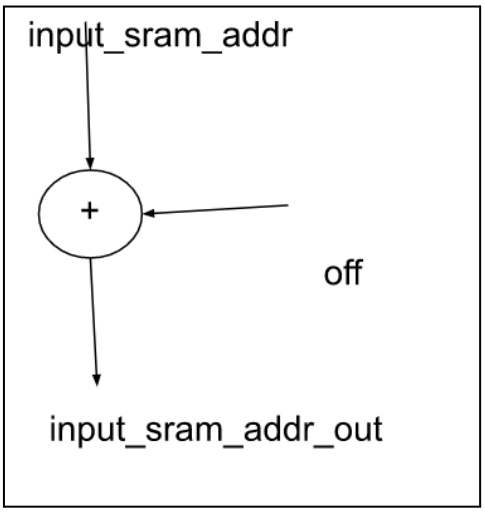




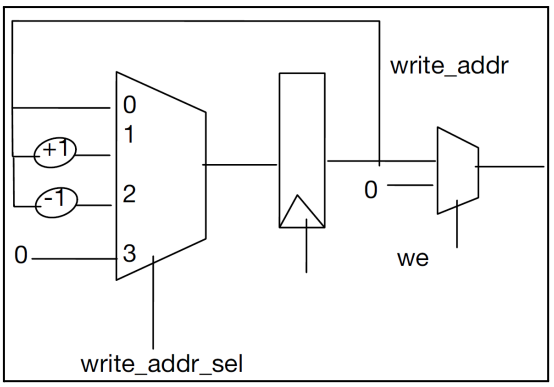


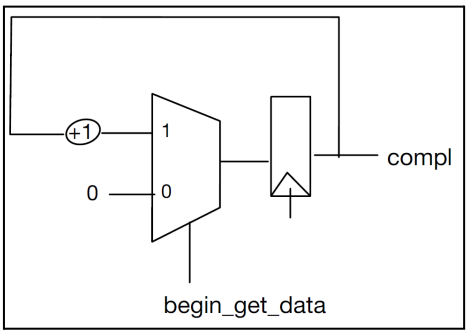


Input address reading and offset value., along with the enable signal to writing the sram

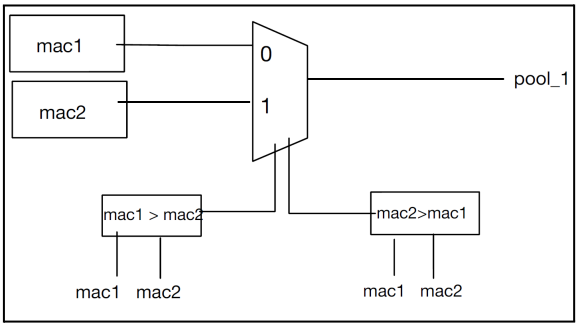


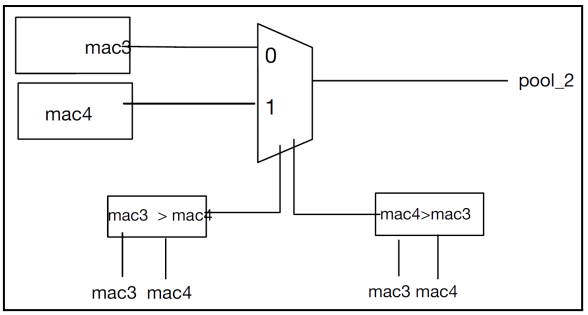
, compl control signal to set the count values of the sram, iterates to next thing when it is reached or saturated.





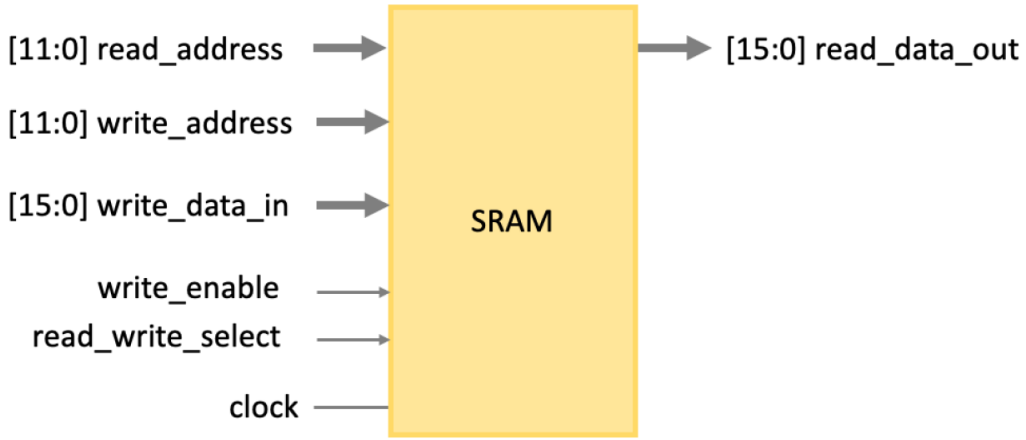
Max pooling units and their control signals





3.Interface Specifications

SRAM interface



TYPE	WIDTH	NAME	FUNCTION
wire	1	dut_run	Indicates function started
reg	1	dut_busy	Model is running
wire	1	reset_b	Reset the hardware
wire	1	clk	clock
reg	1	input_sram_write_enable	Enable for write/read
reg	12	input_sram_write_addr	Write address
reg	16	input_sram_write_data	Writes the data to SRAM
reg	12	input_sram_read_addr	Read address to input SRAM
reg	16	input_sram_read_data	Input data
reg	1	output_sram_write_enable	Enable for write/read
reg	12	output_sram_write_addr	Write address
reg	16	output_sram_write_data	Writes the data to outputSRAM
reg	12	output_sram_read_addr	Read address to output SRAM
reg	16	output_sram_read_data	output data to output SRAM
reg	1	weight_sram_write_enable	Enable for write/read
reg	12	weight_sram_write_addr	Write address for weights
reg	16	weight_sram_write_data	Writes the data to weight SRAM

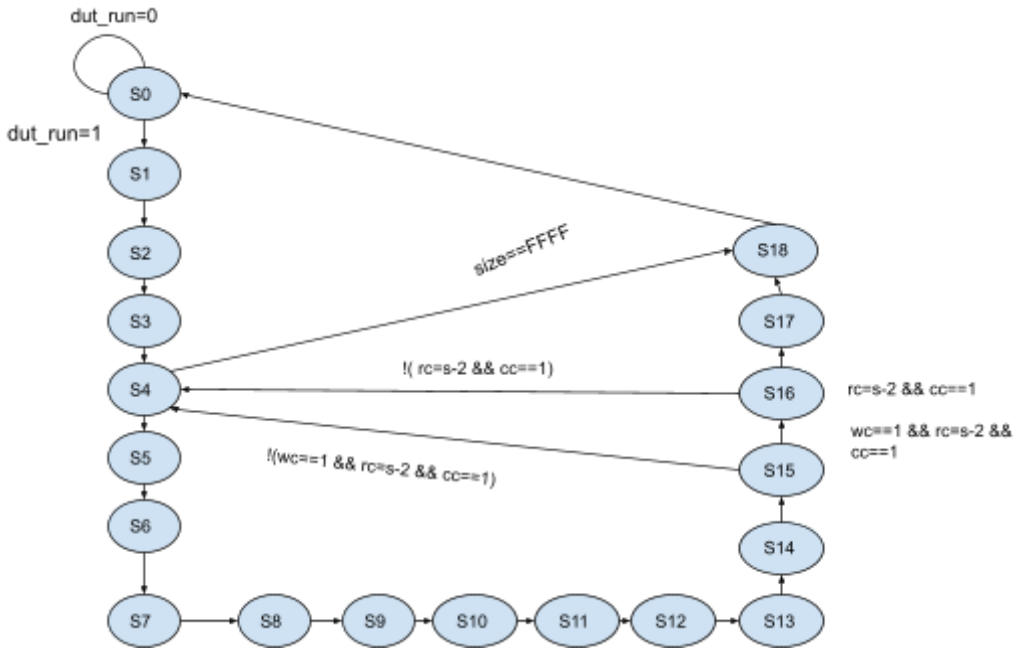
reg	12	weight_sram_read_addr	Read address to weight SRAM
reg	16	weight_sram_read_data	Weight Input data
reg signed	16	input	Data inputs for convolution
reg signed	8	w_data	weights
reg	1	get_data_begin	Enable for fetching data
reg	4	reg_flag_select	Enable for splitting the data
reg	1	higher_bits_select	Selects the data's higher byte
reg	1	lower_bits_select	Selects the data's lower byte
reg	1	valid_conv_check	Convolution enable
reg	2	kerenel_select	Flag for Weight matrix selection
reg	2	input_sel_address	Enable for input addr select
reg	1	output_write_addr	Enable to write for outputSRAM
reg signed	20	mac1,mac2	units for convolution
reg	8	relu1,relu2	Activation functions
reg	16	output_relu	Relu output
reg	7	end_of_address	Wraps the output sram address to next row
reg	2	end_of_address_sel	enable
reg	4	current_state	FSM current state
reg	4	next_state	FSM next state
reg signed	30	temp11,temp22…..temp88	Temporary storing
reg	4	compl	Control signal for address filling
reg	1	wc	Write count
reg	2	wc_sel	Selection line
reg	1	next_addr_sel	Selection for next incoming addr
reg	3	cc	Column count
reg	5	cc_sel	Column count select
reg	1	clear_new	Resets the offset

Reg signed	20	pool_1,pool_2	Max poolers
reg	20	off	Offset value
reg	16	output _dimension	Size for output
reg	8	relu_final	Relu final output
reg	16	final	Max poolin output
reg	1	clear_addr_selection	Clears the addresses for new image

4.Technical Implementation

FSM

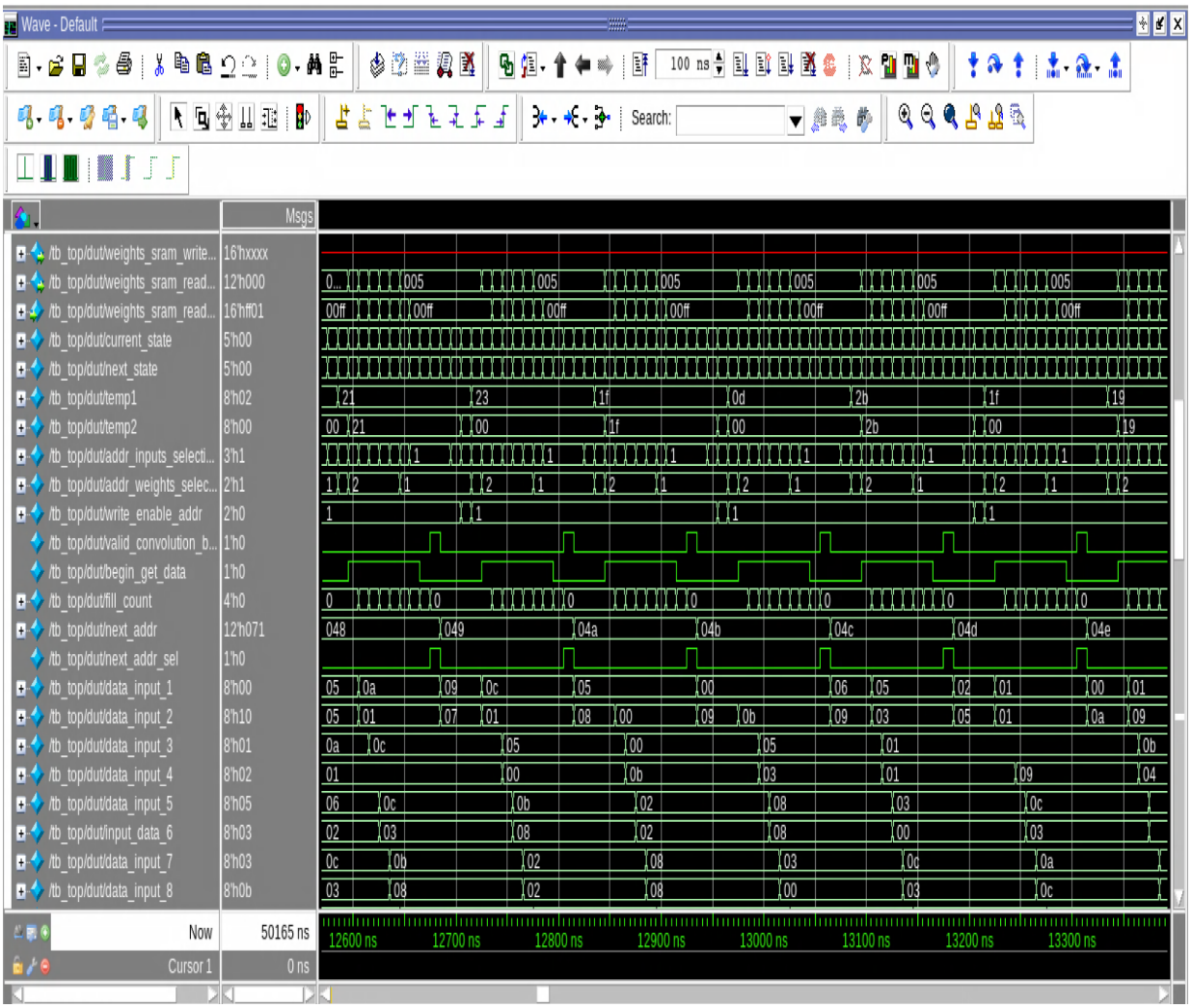
Finite state machine is necessary to design the control path, so that model knows what to do at each step. Control signals control the path the model needs to go.
Total 18 states are used for designing the model.



When the dut_run goes high the FSM starts, else it remains in the s0 state. Then FSM goes up to state 15, where it again checks for the condition(writing the data to output sram addresses). If the condition meets it goes state 16 , otherwise iterates over s4 state, reiterates over, when it meets the condition, moves to s16 state, then moves on to s17. There it again checks the condition for the row count and column count to iterate over the next row or shifting to next rows and columns. Else it goes to s2. When the size is equal to FFFF or saturated s18, it moves to s0 state.

5.Verification

- Design verification is where you test (“verify”) that your design outputs match your design inputs. Using testbench,outputs that are convolution output, Relu output are verified with the output SRAM waveforms.
- The testbench compares the computed output with the golden reference values and reports any mismatch.
- The read address increments by 1 and needs to hold when performing convolution.
- As the read address increments the data is fetched into the convolution unit.
- The convolution is performed at Multiply&Adder units and transferred to the ReLu unit, then stored in the unit.
- As soon as write enable is asserted the output is written back to the SRAM.
- Any discrepancy in the output data is caught by comparing output result with golden values.



The design is verified using the make verify-<file>-extra.

6.Results Achieved

Clock Period	6.5ns
Area	18091.1918μm2
Total Cycles	4908
Delay	30920.4ns

Transcript

```
# Loading work.MyDesign(fast)
# run -all
# +CLASS+564
# +RUN_TYPE+base
# --------------------------------start_simulation--------------------------------
#
# INFO::tb_top.input_mem.loadInitFile::readmem : ../564/input_0/input_sram_564.dat
# INFO::tb_top.weight_mem.loadInitFile::readmem : ../564/input_0/weight_sram_564.dat
# --------------------------------Round 0 start--------------------------------
#
# --------------------------------Round 0 check start--------------------------------
#
# --------------------------------store results to g_result.dat--------------------------------
#
# --------------------------------load results to output_array--------------------------------
#
# --------------------------------load results to golden_output_array--------------------------------
#
# INFO::tb_top::readmem : ../564/input_0/golden_base_outputs_564.dat
# --------------------------------Round 0 start compare --------------------------------
#
# --------------------------------Round 0 Your report--------------------------------
#
# Check 1 : Correct g results = 143/143
# computeCycle=3554
# ------------------------------------------------------------------------------------
#
# INFO::tb_top.input_mem.loadInitFile::readmem : ../564/input_1/input_sram_564.dat
# INFO::tb_top.weight_mem.loadInitFile::readmem : ../564/input_1/weight_sram_564.dat
# --------------------------------Round 1 start--------------------------------
#
# --------------------------------Round 1 check start--------------------------------
#
# --------------------------------store results to g_result.dat--------------------------------
#
# --------------------------------load results to output_array--------------------------------
#
# --------------------------------load results to golden_output_array--------------------------------
#
# INFO::tb_top::readmem : ../564/input_1/golden_base_outputs_564.dat
# --------------------------------Round 1 start compare --------------------------------
#
# --------------------------------Round 1 Your report--------------------------------
#
# Check 1 : Correct g results = 55/55
# computeCycle=1354
#
```

Timing Reports

****

Timing_max_slow:

```
Point                                    Incr       Path
-------------------------------------------------------------
clock clk (rise edge)                   0.0000     0.0000
clock network delay (ideal)             0.0000     0.0000
data_input_4_reg[4]/CK (DFF_X1)         0.0000     0.0000 r
data_input_4_reg[4]/Q (DFF_X1)          0.5833     0.5833 f
U13739/Z (XOR2_X2)                      0.3981     0.9814 f
U2504/ZN (INV_X4)                       0.1743     1.1557 r
U13738/ZN (NAND2_X4)                    0.1157     1.2715 f
U13058/ZN (OAI22_X2)                    0.2374     1.5089 r
U13115/ZN (XNOR2_X2)                    0.3687     1.8776 r
U12595/ZN (XNOR2_X2)                    0.3202     2.1978 r
U12859/ZN (XNOR2_X2)                    0.3134     2.5112 r
U12858/ZN (XNOR2_X2)                    0.3198     2.8309 r
U2081/ZN (NAND2_X1)                     0.0920     2.9230 f
U1223/ZN (NAND2_X1)                     0.1843     3.1073 r
U10165/ZN (OAI21_X1)                    0.1653     3.2726 f
U14966/ZN (NAND2_X2)                    0.1438     3.4164 r
U5250/ZN (XNOR2_X2)                     0.2911     3.7075 r
U11569/ZN (XNOR2_X2)                    0.3184     4.0259 r
U1565/ZN (NAND2_X1)                     0.1212     4.1471 f
U5237/ZN (NAND2_X2)                     0.1286     4.2757 r
U5235/ZN (NAND2_X2)                     0.0687     4.3444 f
U1864/ZN (NAND2_X2)                     0.1232     4.4676 r
U12828/ZN (XNOR2_X2)                    0.2911     4.7586 r
U12827/ZN (XNOR2_X2)                    0.3304     5.0890 r
U12865/ZN (NOR2_X2)                     0.0946     5.1836 f
U2682/ZN (INV_X4)                       0.0952     5.2788 r
U5908/ZN (NAND2_X2)                     0.0758     5.3546 f
U1792/ZN (NOR2_X2)                      0.1572     5.5117 r
U6203/ZN (NAND2_X2)                     0.0885     5.6003 f
U5909/ZN (NAND3_X2)                     0.1179     5.7182 r
U1757/ZN (INV_X1)                       0.0722     5.7903 f
U6456/ZN (NAND2_X2)                     0.0906     5.8809 r
U13584/ZN (NAND3_X2)                    0.0669     5.9478 f
mac_2_reg[18]/D (DFF_X2)                0.0000     5.9478 f
data arrival time                                  5.9478

clock clk (rise edge)                   6.3000     6.3000
clock network delay (ideal)             0.0000     6.3000
clock uncertainty                      -0.0500     6.2500
mac_2_reg[18]/CK (DFF_X2)               0.0000     6.2500 r
library setup time                     -0.3021     5.9479
data required time                                 5.9479
-------------------------------------------------------------
data required time                                 5.9479
data arrival time                                 -5.9478
-------------------------------------------------------------
slack (MET)                                        0.0000
```

Timing_max_slow_hold

```
-----------------------------------------------------------
clock clk (rise edge)                    0.0000     0.0000
clock network delay (ideal)              0.0000     0.0000
data_input_4_reg[4]/CK (DFF_X1)          0.0000     0.0000 r
data_input_4_reg[4]/Q (DFF_X1)           0.5833     0.5833 f
U13739/Z (XOR2_X2)                       0.3981     0.9814 f
U2504/ZN (INV_X4)                        0.1743     1.1557 r
U13738/ZN (NAND2_X4)                     0.1157     1.2715 f
U13058/ZN (OAI22_X2)                     0.2374     1.5089 r
U13115/ZN (XNOR2_X2)                     0.3687     1.8776 r
U12595/ZN (XNOR2_X2)                     0.3202     2.1978 r
U12859/ZN (XNOR2_X2)                     0.3134     2.5112 r
U12858/ZN (XNOR2_X2)                     0.3198     2.8309 r
U2081/ZN (NAND2_X1)                      0.0920     2.9230 f
U1223/ZN (NAND2_X1)                      0.1843     3.1073 r
U10165/ZN (OAI21_X1)                     0.1653     3.2726 f
U14966/ZN (NAND2_X2)                     0.1438     3.4164 r
U5250/ZN (XNOR2_X2)                      0.2911     3.7075 r
U11569/ZN (XNOR2_X2)                     0.3184     4.0259 r
U1565/ZN (NAND2_X1)                      0.1212     4.1471 f
U5237/ZN (NAND2_X2)                      0.1286     4.2757 r
U5235/ZN (NAND2_X2)                      0.0687     4.3444 f
U1864/ZN (NAND2_X2)                      0.1232     4.4676 r
U12828/ZN (XNOR2_X2)                     0.2911     4.7586 r
U12827/ZN (XNOR2_X2)                     0.3304     5.0890 r
U12865/ZN (NOR2_X2)                      0.0946     5.1836 f
U2682/ZN (INV_X4)                        0.0952     5.2788 r
U5908/ZN (NAND2_X2)                      0.0758     5.3546 f
U1792/ZN (NOR2_X2)                       0.1572     5.5117 r
U6203/ZN (NAND2_X2)                      0.0885     5.6003 f
U5909/ZN (NAND3_X2)                      0.1179     5.7182 r
U1757/ZN (INV_X1)                        0.0722     5.7903 f
U6456/ZN (NAND2_X2)                      0.0906     5.8809 r
U13584/ZN (NAND3_X2)                     0.0669     5.9478 f
mac_2_reg[18]/D (DFF_X2)                 0.0000     5.9478 f
data arrival time                                   5.9478

clock clk (rise edge)                    6.3000     6.3000
clock network delay (ideal)              0.0000     6.3000
clock uncertainty                       -0.0500     6.2500
mac_2_reg[18]/CK (DFF_X2)                0.0000     6.2500 r
library setup time                      -0.3021     5.9479
data required time                                  5.9479
-----------------------------------------------------------
data required time                                  5.9479
data arrival time                                  -5.9478
-----------------------------------------------------------
slack (MET)                                         0.0000
```

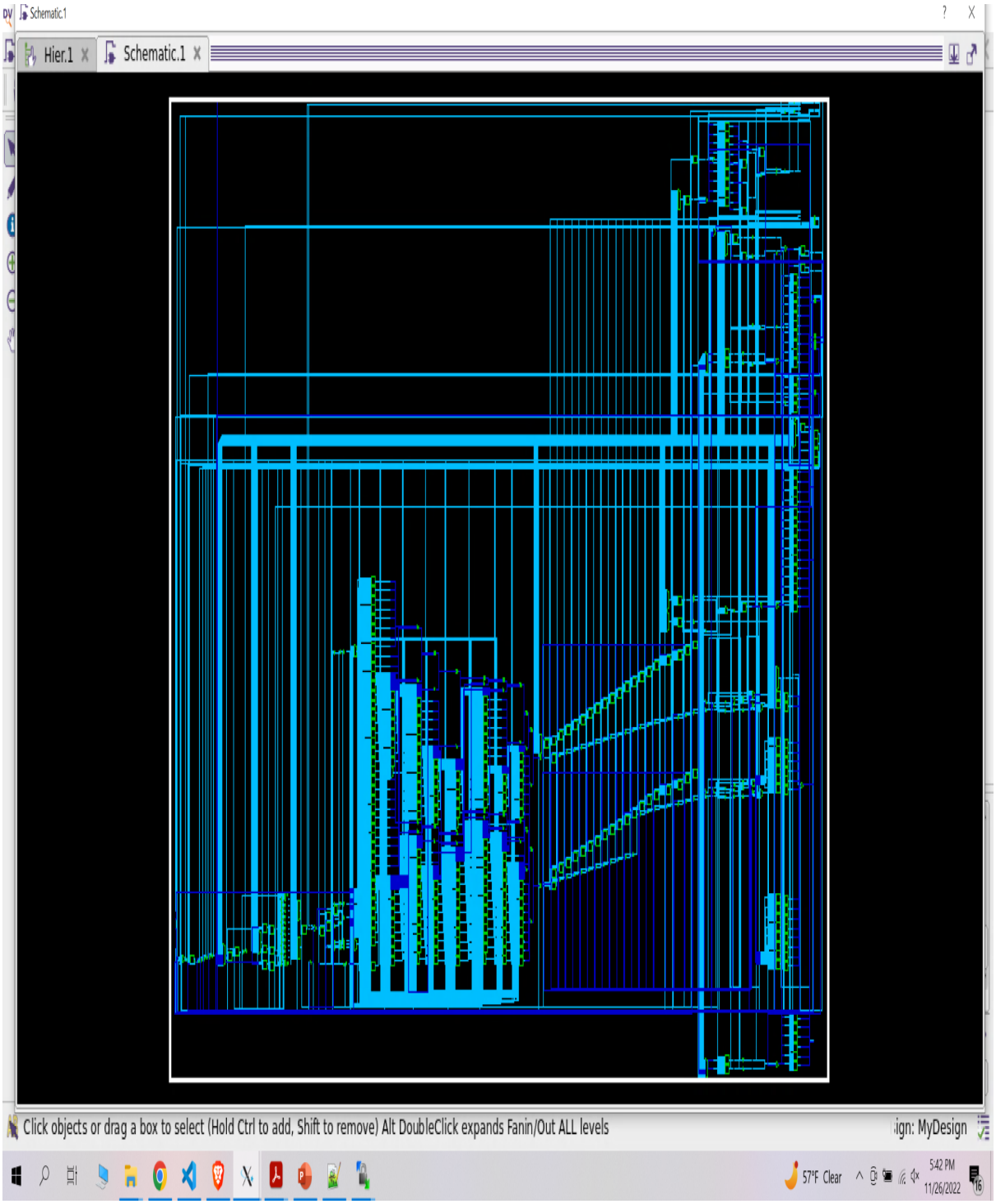
Timing_min_fast_hold

```
Information: Updating design information... (UID-85)

****************************************
Report : timing
        -path full
        -delay min
        -max_paths 1
Design : MyDesign
Version: S-2021.06-SP3
Date   : Mon Dec  5 14:40:04 2022
****************************************

Operating Conditions: fast   Library: NangateOpenCellLibrary_PDKv1_2_v2008_10_fast_nldm
Wire Load Model Mode: top

  Startpoint: compl_reg[0]
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: compl_reg[0]
            (rising edge-triggered flip-flop clocked by clk)
  Path Group: clk
  Path Type: min

  Point                                   Incr       Path
  -----------------------------------------------------------
  clock clk (rise edge)                 0.0000     0.0000
  clock network delay (ideal)           0.0000     0.0000
  compl_reg[0]/CK (DFF_X1)              0.0000     0.0000 r
  compl_reg[0]/Q (DFF_X1)               0.0624     0.0624 r
  U15082/ZN (NOR2_X1)                   0.0222     0.0846 f
  compl_reg[0]/D (DFF_X1)               0.0000     0.0846 f
  data arrival time                                0.0846

  clock clk (rise edge)                 0.0000     0.0000
  clock network delay (ideal)           0.0000     0.0000
  clock uncertainty                     0.0500     0.0500
  compl_reg[0]/CK (DFF_X1)              0.0000     0.0500 r
  library hold time                     0.0002     0.0502
  data required time                               0.0502
  -----------------------------------------------------------
  data required time                               0.0502
  data arrival time                               -0.0846
  -----------------------------------------------------------
  slack (MET)                                      0.0344
```

Total Area

```
--------------------------------------------------------------------------------
Total 15479 cells                                   18091.1918
1
```

Schematic:


Schematic.1
Hier.1
Schematic.1
Click objects or drag a box to select (Hold Ctrl to add, Shift to remove) Alt DoubleClick expands Fanin/Out ALL levels
ign: MyDesign
57°F Clear
5:42 PM
11/26/2022

7.Conclusion

The key takeaway from the project are:

- Understood the importance of Design before Coding in complex circuits.
- Control signals, control path and data path are needed before coding.
- Latches can lead to timing issues and race conditions. They may lead to combinatorial feedback - routing of the output back to the input - which can be unpredictable.
- The circuit should meet all the timing requirements.
- There should not be any Hold or Set-Up time violations.

Deep Neural Network consists of a Convolutional neural network along with ReLu activation function along with Max pooling which works well for the given inputs and weights. The design can be tweaked in order to change according to the inputs. No unintentional latches were formed during the synthesis, which is a major thing in real-time design. No timing arcs were observed, no setup and hold violations. The model has been developed to meetup all the violations. Common warnings were ignored and other warnings were removed. The following design generated the 19625.4798μm^2, with clock period 5.9ns and clock cycles are 3554 for input1 and 1354 for 2nd input, by meeting all the timing constraints.
The trade-off between Area and Clock Period:

CLOCK	AREA	VIOLATED/MET
10	14768.3201	MET
9	14954.2541	MET
8	15496.8941	MET
7	16571.8000	MET
6.5	17273.5079	MET
6	18076.5618	MET
6.3	18091.1918	MET