NC State University
Department of Electrical and Computer Engineering
ECE 564 Project: Fall 2022

Deep Neural Networks

Name: Raghul Srinivasan
Unity ID: rsriniv7
Student ID: 200483357

Delay (ns to run provided example): 33705.6 ns
Clock period: 4.8 ns
Cycles: 7022

Logic Area:
23043.5799 (um^2)

Memory: N/A

1/(delay.area):
1.2875×10^{-9} ($ns^{-1}.um^{-2}$)

Delay (TA provided example, TA to complete):

1/(delay.area) (TA):

Deep Neural Network

Raghul Srinivasan

ABSTRACT:

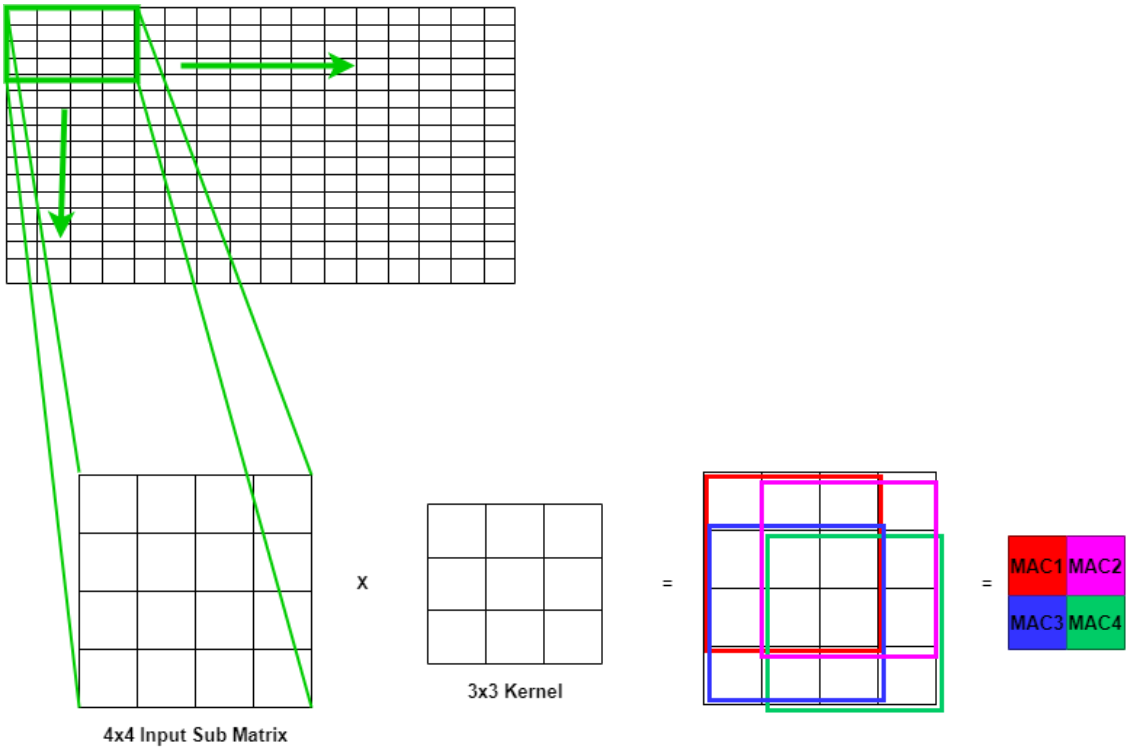
This project is aimed to design, simulate and synthesize hardware to perform a multi-stage neural network, including a convolutional layer, and a max pooling layer. Designed a complex FSM to implement pipelining and reduce latency. Performed synthesis of the design using Synopsis Design Compiler to evaluate the clock period, timing constraints, and logic area of the design. After the synthesis, the final design hardware took 7022 cycles to compute the result for the given data set with an area to be 23043.5799 μm^2 and a clock period of 4.8 ns. The overall performance per unit time of the circuit was 1.2875 x10^{-9} ns^{-1} μm^{-2}.

1. INTRODUCTION:

- The hardware implemented is a multi-stage neural network that includes convolution and max-pooing of stride 2.
- The designed hardware can produce Max Poling results for input matrices of size 4x4, 8x8, 16x16, 32x32, and 64x64 with a 3x3 Kernel matrix and ReLu function.
- Inputs and Kernels are read from input SRAM and Weight SRAM respectively.
- Max pooling output is stored in Output SRAM.
- The approach while designing the hardware was to optimize the clock period and #clock cycles by keeping a check on the logic area as well.
- Implemented hardware uses 4 MACs to perform convolution on the Input matrix and Kernel Matrix.
- The final design can generate the Max Pooling output in Output SRAM in 7022 clock cycles with a clock period of 4.8ns

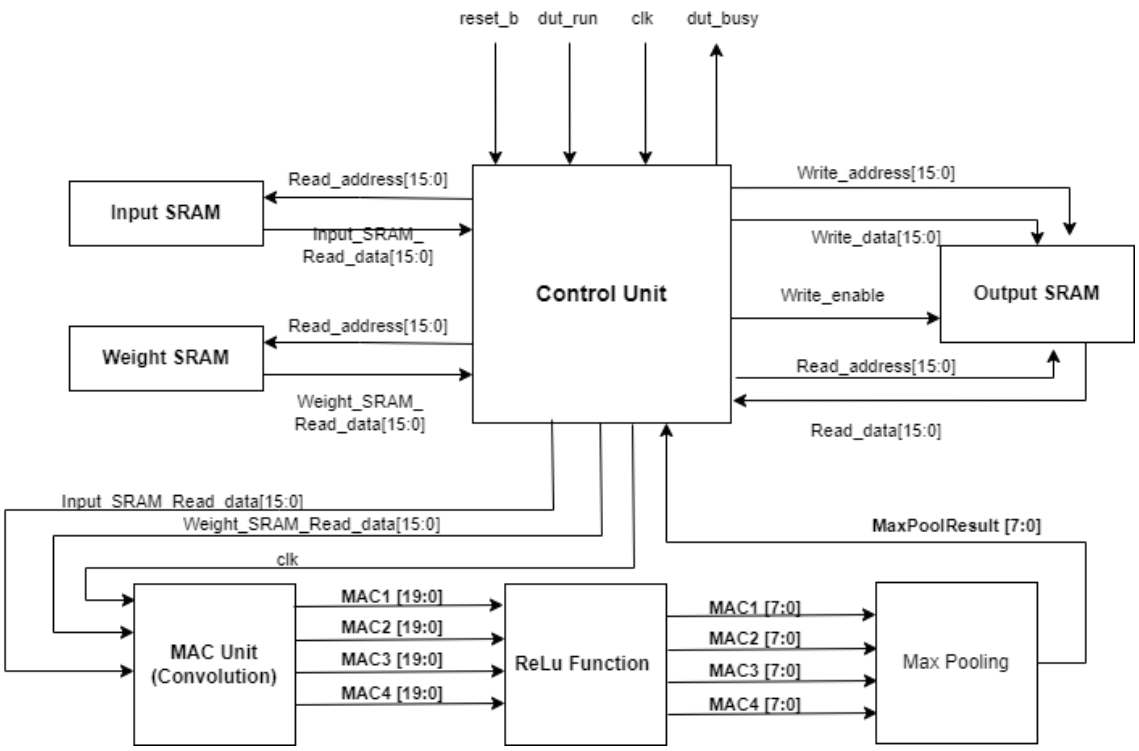
2. MICRO-ARCHITECTURE:

- The algorithmic approach used is to design the data path first and then add the required control signals. Using this as the base, FSM was designed.



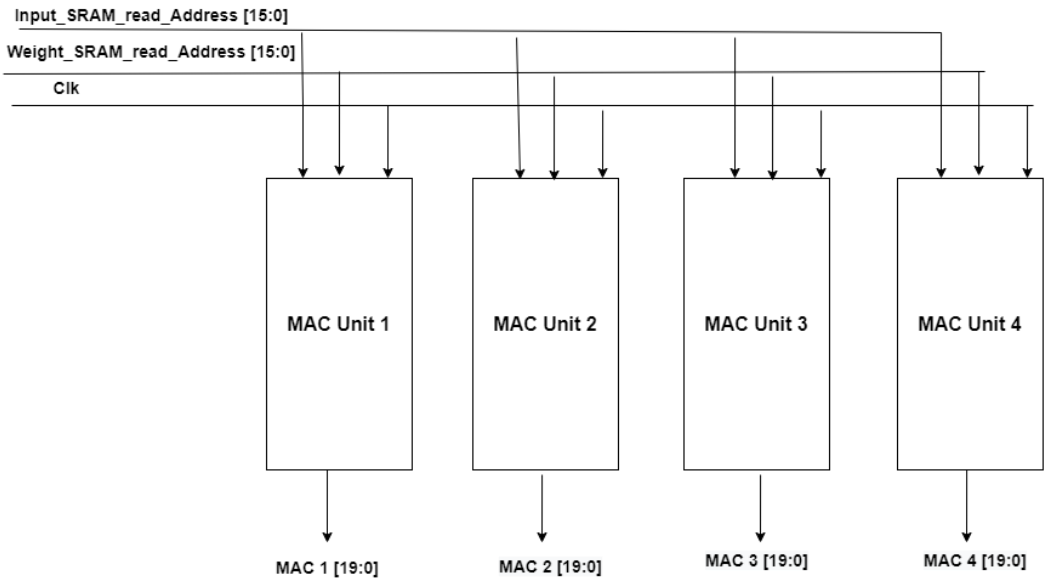
- My design consists of a convolution unit that initially takes a 4x4 Input sub matrix and does 4 Kernel Multiplication to get 4 outputs.
- Output from my convolution unit is passed to an activation function (ReLU (y = max(0,x)) and then to a Max-Pooling Layer.
- Max Pooling output is uploaded to the Output SRAM and the convolution is continued for the next set of 4x4 Input Sub Matrix.

High Level Design

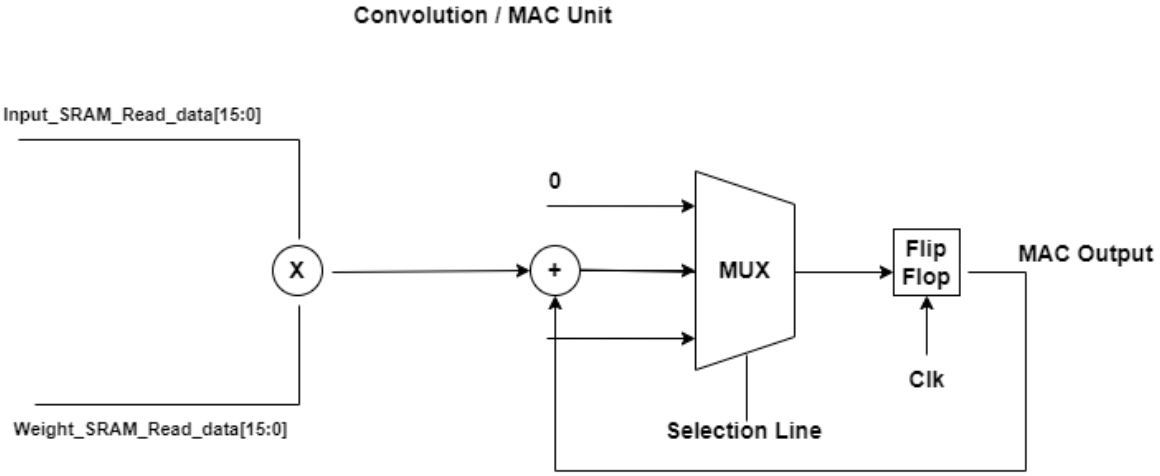


- The Control Unit interfaces with three different SRAMs to fetch Inputs and weights and to store the outputs in appropriate SRAMs. Note that write_enable is by default 0'b0 for Input SRAM and weight SRAM as we would only be reading data and won't be writing to it.
- In the case of Output SRAM, write_enable is set when we are writing the Max pooling Output to the SRAM.
- The dut_run signal is the input signal to the Hardware from the testbench to start fetching the inputs and start convolution.
- Once the convolution starts, dut_busy is set high which means that the hardware is busy computing the convolution/Max Pooling for the given inputs.
- Once the convolution is done, the hardware sends an acknowledgment signal by resetting the dut_busy signal.

All 4 MAC/ Convolution Units



- The above block diagram shows the data flow of Inputs/weights to the MACs. With the aid of control signals, we would calculate MAC1, MAC2, MAC3, and MAC4 using unique MAC units.
- Inside each of the MAC units using SelectionLines, Accumulation output is calculated for each of the 4x4 Matrix.
- Inputs and weights are signed inputs. So, the multiplier in the MAC unit is signed.
- Inputs to the multiplier are 8-bits each and the output is 16 bits and the accumulator is 20 bits



- The output of the MAC unit is passed the activation ReLu function.
- 4 ReLu outputs from the 4x4 input submatrix are compared for the largest one to be stored in output SRAM.

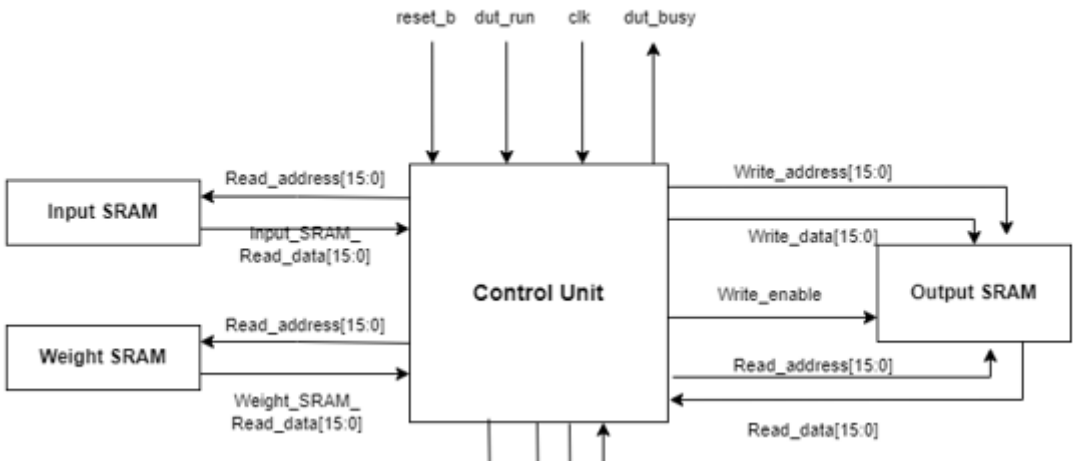
FSM

State	State - Events
S0 – Default State	This is the default state. Here the hardware waits for Dut_run to be set by the testbench to start the computation. At this state, 1. We read the 'N' value from address 0000 from input SRAM. 2. Initialize MAC1, AMC2, MAC3 and MAC4 = 0
S1	In this state, we set the dut_busy register to let know the testbench that the computation is going on. We also set the output_sram_write_enable to 0'b0
S2	We get the first 2 values of kernel Matrix and is stored in W1 and W2
S3	1. Get the next 2 values of the kernel matrix and is stored them in W3 and W4 2. Get the first 2 values of the input matrix and is stored them in R1a and R1b
S4	1. Get the next 2 values of the kernel matrix and is stored them in W5 and W6 2. Get the next 2 values of the input matrix and is stored them in R2a and R2b 3. Calculate R1a*W1 and accumulate it to MAC1 4. Calculate R1b*W1 and accumulate it to MAC2
S5	1. Get the next 2 values of the kernel matrix and is stored them in W7 and W8 2. Get the next 2 values of the input matrix and is stored in R3a and R3b 3. Calculate R1b*W2 and accumulate it to MAC1 4. Calculate R2a*W2 and accumulate it to MAC2
S6	1. Get the last value of the kernel matrix and is stored in W9 2. Get the next 2 values of the input matrix and is stored in R4a and R4b

	3. Calculate R2a*W3 and accumulate it to MAC1 4. Calculate R2b*W3 and accumulate it to MAC2 5. Calculate R3a*W1 and accumulate it to MAC3 6. Calculate R3b*W1 and accumulate it to MAC4
S7	1. Get the next 2 values of the input matrix and is stored them in R5a and R5b 2. Calculate R3a*W4 and accumulate it to MAC1 3. Calculate R3b*W4 and accumulate it to MAC2 4. Calculate R3b*W2 and accumulate it to MAC3 5. Calculate R4a*W2 and accumulate it to MAC4
S8	1. Get the next 2 values of the input matrix and is stored in R6a and R6b 2. Calculate R3b*W5 and accumulate it to MAC1 3. Calculate R4a*W5 and accumulate it to MAC2 4. Calculate R4a*W3 and accumulate it to MAC3 5. Calculate R4b*W3 and accumulate it to MAC4
S9	1. Get the next 2 values of the input matrix and is stored in R7a and R7b 2. Calculate R4a*W6 and accumulate it to MAC1 3. Calculate R4b*W6 and accumulate it to MAC2 4. Calculate R5a*W4 and accumulate it to MAC3 5. Calculate R5b*W4 and accumulate it to MAC4
S10	1. Get the next 2 values of the input matrix and is stored in R8a and R8b 2. Calculate R5a*W7 and accumulate it to MAC1 3. Calculate R5b*W7 and accumulate it to MAC2 4. Calculate R5b*W5 and accumulate it to MAC3 5. Calculate R6a*W5 and accumulate it to MAC4
S11	1. Get the next input data from input_sram_read_data 2. Calculate R5a*W7 and accumulate it to MAC1 3. Calculate R5b*W7 and accumulate it to MAC2 4. Calculate R5b*W5 and accumulate it to MAC3 5. Calculate R6a*W5 and accumulate it to MAC4
S12	1. Calculate R6a*W9 and accumulate it to MAC1 2. Calculate R6b*W9 and accumulate it to MAC2 3. Calculate R7a*W7 and accumulate it to MAC3 4. Calculate R7b*W7 and accumulate it to MAC4
S13	1. MAC1 and MAC2 are passed to the ReLu function 2. Calculate R7b*W8 and accumulate it to MAC3 3. Calculate R8a*W8 and accumulate it to MAC4
S14	1. Calculate R8a*W9 and accumulate it to MAC3 2. Calculate R8b*W9 and accumulate it to MAC4
S15	1. MAC3 and MAC4 are passed to the ReLu function 2. CurrentColumnCounter is incremented 3. OutputSramSlotFlag is toggled
S16	1. Max pooling is done among MAC1, MAC2, MAC3, MAC4 2. CurrentRowCount is incremented, if CurrentColumnCount > (N-2)/2
S17	1. Max pooling output is written to Output SRAM and Write_enable is made high for output_SRAM 2. If input data from Input SRAM is read and if it is 0xFFFF, moved to default state else moved to S18
S18	1. If it is the start of new Input, the N value is updated

	2. The same sequence is continued from S1 again

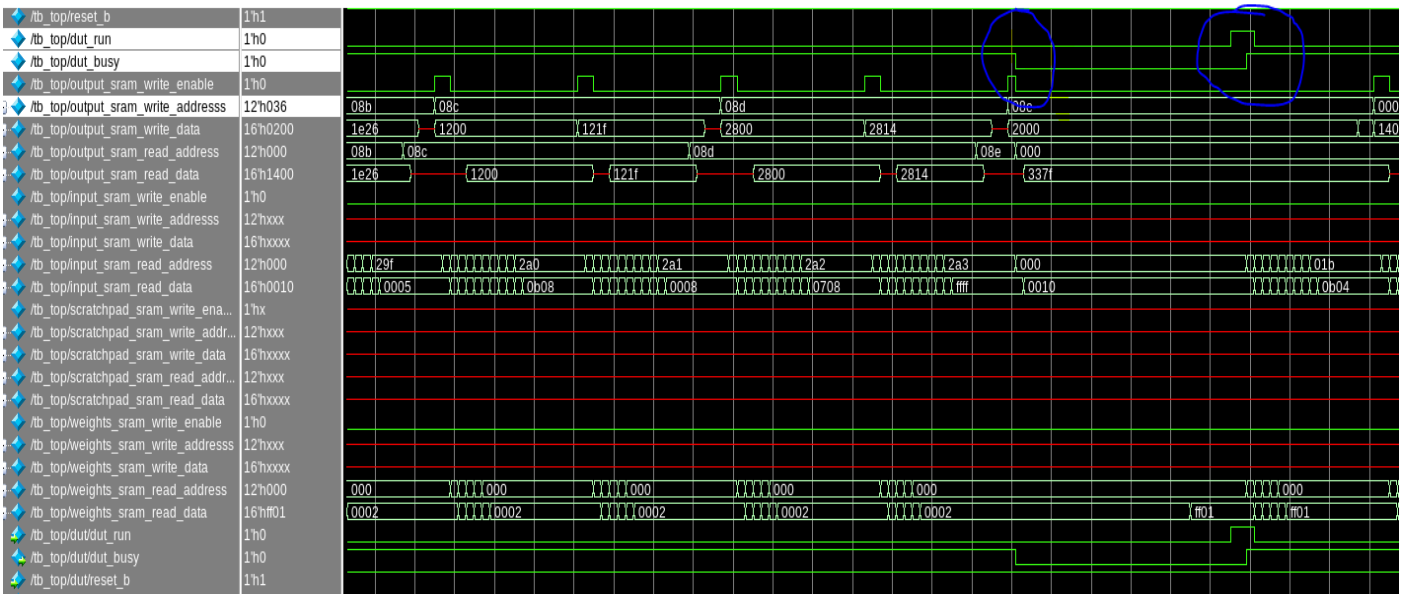
3. INTERFACE SPECIFICATION:



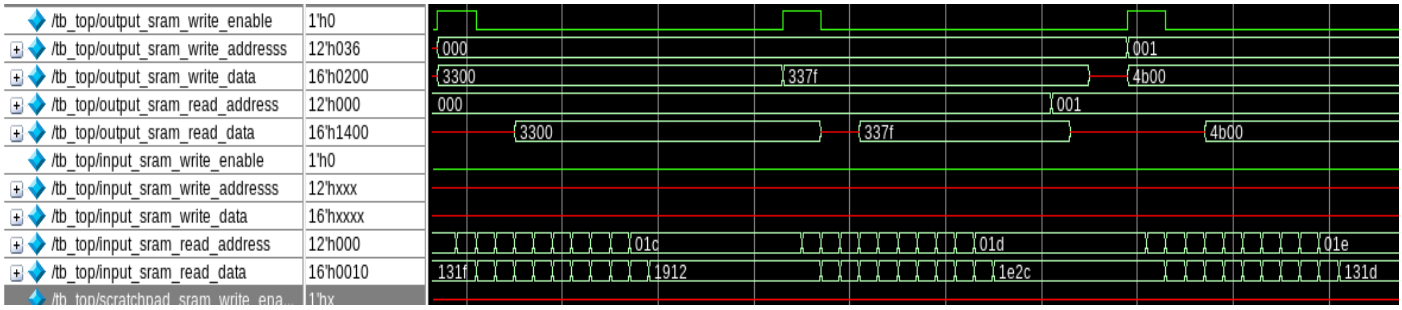
Port Name	Width	Function
Clk	1	Input Port: For feeding clock signal to the design
Reset_b	1	Input Port: Global active low signal for getting the design into a known state
Dut_run	1	Input Port: Signal to begin computation
Dut_busy	1	Output Port: Indicates if computation is in progress or not. It is low when the HW is idle.
Input_sram_write_enable	1	Output Port: Write Enable signal to Input SRAM
Output_sram_write_enable	1	Output Port: Write Enable signal to Output SRAM
Weight_sram_writ_enable	1	Output Port: Write Enable signal to Weight SRAM
Input_sram_write_address	12	Output Port: Write Address to Input SRAM
Output_sram_write_address	12	Output Port: Write Address to Output SRAM
Weight_sram_write_address	12	Output Port: Write Address to Weight SRAM
Input_sram_write_data	16	Output Port: Write Data to Input SRAM
Output_sram_write_data	16	Output Port: Write Data to Output SRAM
Weight_sram_write_data	16	Output Port: Write Data to Weight SRAM
Input_sram_read_address	12	Output Port: Read Address to Input SRAM
Output_sram_read_address	12	Output Port: Read Address to Output SRAM
Weight_sram_read_address	12	Output Port: Read Address to Weight SRAM
Input_sram_read_data	16	Output Port: Read Data to Input SRAM
Output_sram_read_data	16	Output Port: Read Data to Output SRAM
Weight_sram_read_data	16	Output Port: Read Data to Weight SRAM
Current_state	5	Register used to store the current state of FSM
Next_State	5	Register used to store the next state of FSM
Nvalue	7	Register to store the N value of the current input matrix
NvalueCounter	7	Register to store (N -2)/2 value of the current input matrix
NvalueAddress	16	Register to store the Input SRAM address where ‘N’ is stored for the current Input Matrix
CurrentRowCounter	6	Register: Row counter of Current Input Matrix
CurrentColumnCounter	6	Register: RowColumn counter of Current Input Matrix

OutputSramSlotFlag	1	Register used to set the position (upper or lower 8 bits) where the Max pool result will be stored in Output SRAM
OutputSramSlotControl	1	The control signal of OutputSramSlotFlag
InputReadAddressSel	3	Mux Selection line to select Read Address of Input SRAM
WeightReadAddressSel	3	Mux Selection line to select Read Address of Weight SRAM
OutputReadAddressSel	2	Mux Selection line to select Read Address of Output SRAM
NextInputData	16	Register to check if it is the end of File (0xFFFF) after each Input
W1,W2,W3,W4,W5, W6,W7,W8	8	Registers used to store 3*3 kernel matrix
R1a, R1b, R2a,R2b,R3a, R3b,R4a,R4b,R5a,R5b,R6a, R6b,R7a,R7b,R8a,R8b	20	Registers used to store 4*4 Input Sub Matrix
MaxPoolResult	8	Register to store the Max pooling result of 4*4 Input Sub Matrix which is to be written to output SRAM

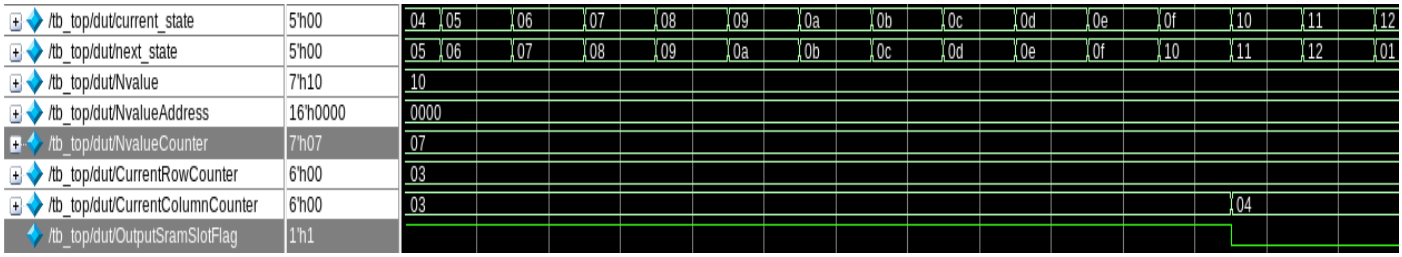
- In the waveform below, we can see the function of dut_busy and dut_run.
- When we read 0xFFFF from Input SRAM t the of the input matrix, it signifies the end of computation for that data set.
- This triggers Dut_busy to be set low by the hardware.
- Which is detected by the testbench to feed the next set of inputs by making dut_run high for one clock edge.



- In the waveform, we can see that we set the appropriate address and data which is to be written to the output SRAM.
- Following that, when we make write_enable high for one positive edge, the data in write_data is written to the SRAM in the following cycle which is evident from the read_data port.
- We can also infer that Max pooling output data is written to Output SRAM alternatively between bits [15:8] and [7:0] which is possible and tracked with the help of the Flag register.

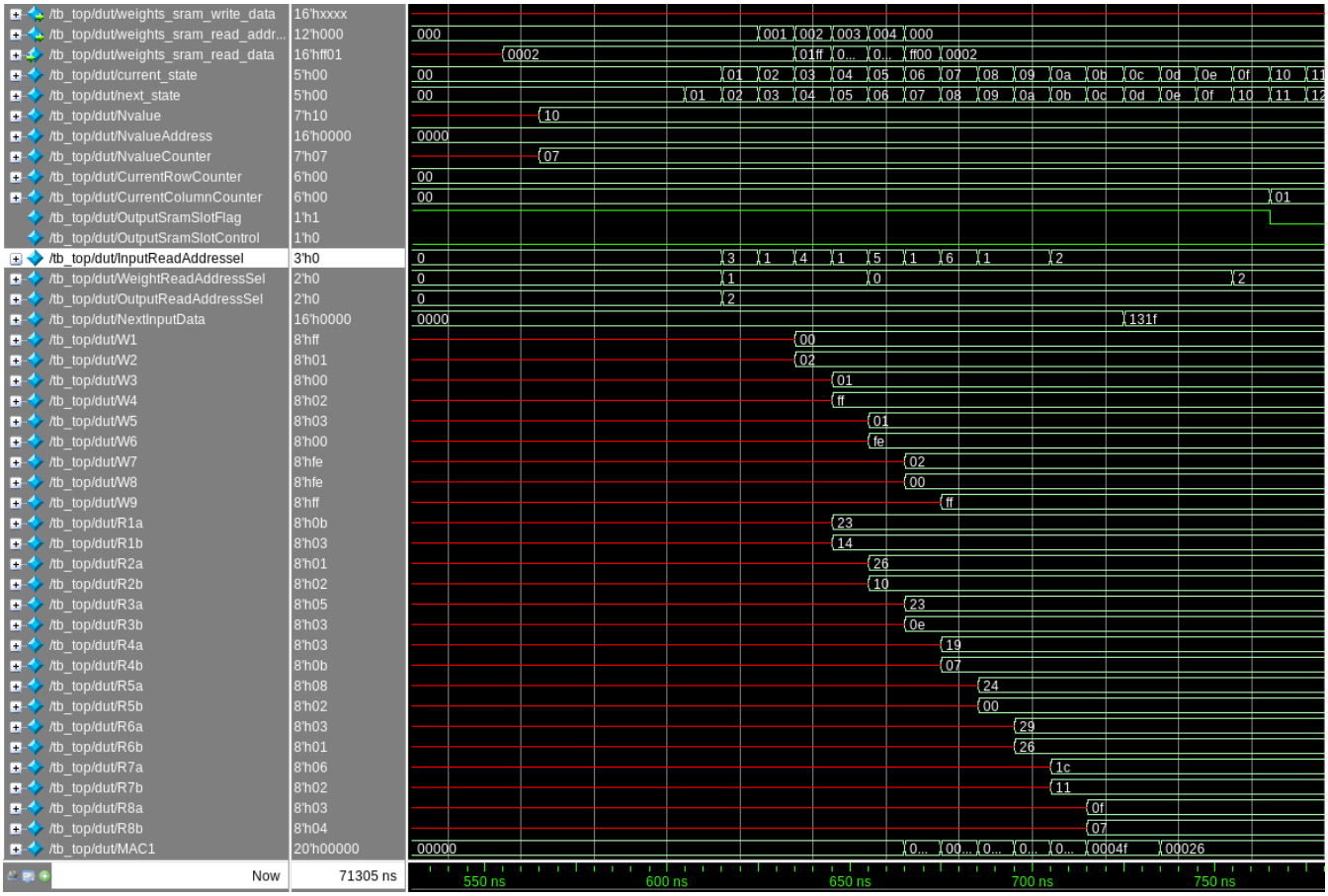


- In the below graph, the current_state and next_state control signal is used to track the current state and next stage of the FSM.
- We can also see that column counter and row counter registers of the current input matrix are tracked each time at state 0x11 of FSM.

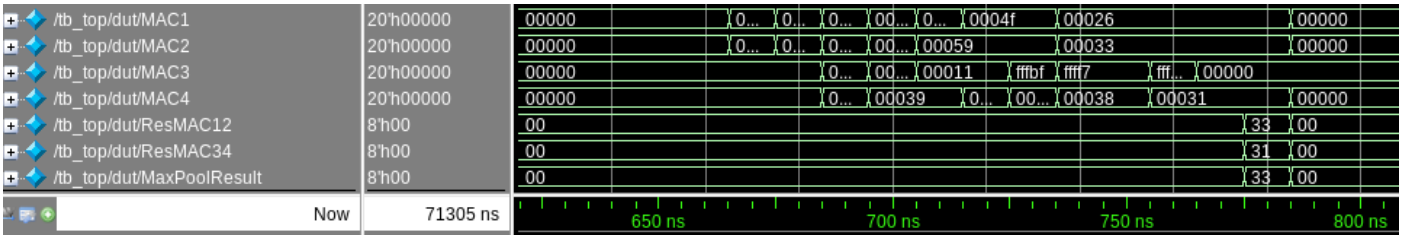


4. VERIFICATION:

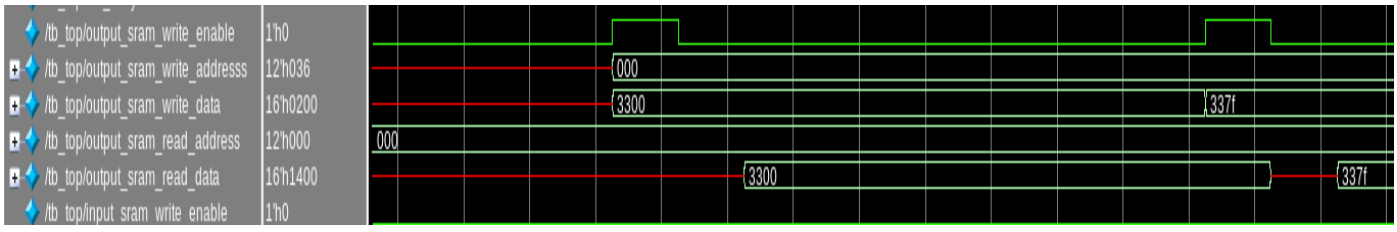
- The functionality of the design was verified using the given testbench for multiple rounds of inputs.
- Reading from the Input SRAM and Weight SRAM is initiated as soon as the dut_run input port is set by the testbench.
- ‘N’ value and ‘N’address registers are updated for each input at the right time.
- Proper latency handling of all 3 SRAMs read and write is evident.



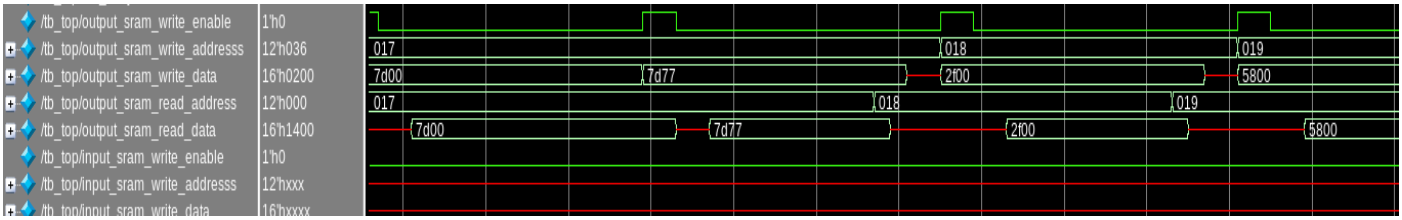
- As soon as the dut_run signal goes high, FSM moves on to S! state from S0. From the S1 state, the Input matrix from Input SRAM and kernel matrix from Weight SRAM are stored in registers W1-W8, R1a, R1b-R8a, and R8b.
- N value of the input matrix and its corresponding address is stored in NValue register and Nvalueaddress register.



- From the S4 state, MAC units start to multiply and accumulate the result.
- After calculating all the MACs and after passing through ReLu and Max pooling is done in stage S16.



- From the above graph, after each computation of the max pooling result, data is stored in Output SRAM. Each address in output SRAM contains two max pooling results which are correctly verified by the toggling function of the OutputSramSlotFlag register.



- From the above waveform, at write_address 0x18, we reached the end of the computation of the first input set.
- So MAX pooling output stream of the next input stream should start from the next subsequent address and is also verified with the function of the OutputSramSlotFlag register.

Transcripts (Model Sim):

```
# ------------------------------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=5095
# ----------------------------------------------------------------------------------
#
# 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=1927
# ----------------------------------------------------------------------------------
#
# ** Note: $finish    : ../testbench/testbench.sv(349)
#    Time: 71305 ns  Iteration: 1  Instance: /tb_top
# End time: 16:14:42 on Nov 27,2022, Elapsed time: 0:00:02
# Errors: 0, Warnings: 0
```

5. RESULTS ACHIEVED:

- Clock Period: 4.8ns
- Compute Cycle: 7022
- Area: 23043.5799 (um^2)
- Performance per unit time per unit area: 1.2875×10^{-9} ($ns^{-1}.um^{-2}$)

Check Design Command

read.log - Notepad

File Edit Format View Help

```
                '/afs/unity.ncsu.edu/users/r/rsriniv7/564/Project/projectFall2022/rtl/dut.v'.
================================================================================================
|         Register Name          |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
================================================================================================
| output_sram_read_address_reg  | Flip-flop |  12   |  Y  | N  | N  | N  | N  | N  | N  |
| weights_sram_read_address_reg | Flip-flop |  12   |  Y  | N  | N  | N  | N  | N  | N  |
|  input_sram_read_address_reg  | Flip-flop |  12   |  Y  | N  | N  | N  | N  | N  | N  |
================================================================================================

Inferred memory devices in process
        in routine MyDesign line 131 in file
                '/afs/unity.ncsu.edu/users/r/rsriniv7/564/Project/projectFall2022/rtl/dut.v'.
===============================================================================
|    Register Name    |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
===============================================================================
|  current_state_reg  | Flip-flop |   5   |  Y  | N  | N  | N  | N  | N  | N  |
===============================================================================

Inferred memory devices in process
        in routine MyDesign line 354 in file
                '/afs/unity.ncsu.edu/users/r/rsriniv7/564/Project/projectFall2022/rtl/dut.v'.
================================================================================================
|         Register Name          |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
================================================================================================
|     CurrentRowCounter_reg      | Flip-flop |   6   |  Y  | N  | N  | N  | N  | N  | N  |
|            R2b_reg             | Flip-flop |   8   |  Y  | N  | N  | N  | N  | N  | N  |
|       NextInputData_reg        | Flip-flop |  16   |  Y  | N  | N  | N  | N  | N  | N  |
|            MAC2_reg            | Flip-flop |  20   |  Y  | N  | N  | N  | N  | N  | N  |
|            MAC3_reg            | Flip-flop |  20   |  Y  | N  | N  | N  | N  | N  | N  |
|            R8a_reg             | Flip-flop |   8   |  Y  | N  | N  | N  | N  | N  | N  |
|   output_sram_write_data_reg   | Flip-flop |  16   |  Y  | N  | N  | N  | N  | N  | N  |
|            R3b_reg             | Flip-flop |   8   |  Y  | N  | N  | N  | N  | N  | N  |
|            R3a_reg             | Flip-flop |   8   |  Y  | N  | N  | N  | N  | N  | N  |
|            R7a_reg             | Flip-flop |   8   |  Y  | N  | N  | N  | N  | N  | N  |
|       NvalueCounter_reg        | Flip-flop |   7   |  Y  | N  | N  | N  | N  | N  | N  |
|            R8b_reg             | Flip-flop |   8   |  Y  | N  | N  | N  | N  | N  | N  |
|           Nvalue_reg           | Flip-flop |   6   |  Y  | N  | N  | N  | N  | N  | N  |
|            R6a_reg             | Flip-flop |   8   |  Y  | N  | N  | N  | N  | N  | N  |
|            R7b_reg             | Flip-flop |   8   |  Y  | N  | N  | N  | N  | N  | N  |
|            R5a_reg             | Flip-flop |   8   |  Y  | N  | N  | N  | N  | N  | N  |
|            R6b_reg             | Flip-flop |   8   |  Y  | N  | N  | N  | N  | N  | N  |
|       NvalueAddress_reg        | Flip-flop |  16   |  Y  | N  | N  | N  | N  | N  | N  |
|            R5b_reg             | Flip-flop |   8   |  Y  | N  | N  | N  | N  | N  | N  |
|     OutputSramSlotFlag_reg     | Flip-flop |   1   |  N  | N  | N  | N  | N  | N  | N  |
|             W9_reg             | Flip-flop |   8   |  Y  | N  | N  | N  | N  | N  | N  |
|            R4b_reg             | Flip-flop |   8   |  Y  | N  | N  | N  | N  | N  | N  |
|             W7_reg             | Flip-flop |   8   |  Y  | N  | N  | N  | N  | N  | N  |
|            R4a_reg             | Flip-flop |   8   |  Y  | N  | N  | N  | N  | N  | N  |
|             W6_reg             | Flip-flop |   8   |  Y  | N  | N  | N  | N  | N  | N  |
|             W8_reg             | Flip-flop |   8   |  Y  | N  | N  | N  | N  | N  | N  |
|            R2a_reg             | Flip-flop |   8   |  Y  | N  | N  | N  | N  | N  | N  |
|            R1a_reg             | Flip-flop |   8   |  Y  | N  | N  | N  | N  | N  | N  |
|             W5_reg             | Flip-flop |   8   |  Y  | N  | N  | N  | N  | N  | N  |
|             W3_reg             | Flip-flop |   8   |  Y  | N  | N  | N  | N  | N  | N  |
|            R1b_reg             | Flip-flop |   8   |  Y  | N  | N  | N  | N  | N  | N  |
|             W1_reg             | Flip-flop |   8   |  Y  | N  | N  | N  | N  | N  | N  |
|             W4_reg             | Flip-flop |   8   |  Y  | N  | N  | N  | N  | N  | N  |
|             W2_reg             | Flip-flop |   8   |  Y  | N  | N  | N  | N  | N  | N  |
| weights_sram_write_enable_reg  | Flip-flop |   1   |  N  | N  | N  | N  | N  | N  | N  |
|    CurrentColumnCounter_reg    | Flip-flop |   6   |  Y  | N  | N  | N  | N  | N  | N  |
|  output_sram_write_enable_reg  | Flip-flop |   1   |  N  | N  | N  | N  | N  | N  | N  |
|            MAC4_reg            | Flip-flop |  20   |  Y  | N  | N  | N  | N  | N  | N  |
|       MaxPoolResult_reg        | Flip-flop |   8   |  Y  | N  | N  | N  | N  | N  | N  |
|  input_sram_write_enable_reg   | Flip-flop |   1   |  N  | N  | N  | N  | N  | N  | N  |
| output_sram_write_addresss_reg | Flip-flop |  12   |  Y  | N  | N  | N  | N  | N  | N  |
|          dut_busy_reg          | Flip-flop |   1   |  N  | N  | N  | N  | N  | N  | N  |
|            MAC1_reg            | Flip-flop |  20   |  Y  | N  | N  | N  | N  | N  | N  |
================================================================================================
```

Total Area

cell_report_final.rpt - Notepad

File Edit Format View Help

```
                                                    4.7880  n
output_sram_write_data_reg[13]
                          DFF_X2          NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
                                                    4.7880  n
output_sram_write_data_reg[14]
                          DFF_X2          NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
                                                    4.7880  n
output_sram_write_data_reg[15]
                          DFF_X2          NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
                                                    4.7880  n
output_sram_write_enable_reg
                          DFF_X2          NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
                                                    4.7880  n
weights_sram_read_address_reg[0]
                          DFF_X2          NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
                                                    4.7880  n
weights_sram_read_address_reg[1]
                          DFF_X2          NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
                                                    4.7880  n
weights_sram_read_address_reg[2]
                          DFF_X2          NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
                                                    4.7880  n
weights_sram_read_address_reg[3]
                          DFF_X2          NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
                                                    4.7880  n
weights_sram_read_address_reg[4]
                          DFF_X2          NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
                                                    4.7880  n
weights_sram_read_address_reg[5]
                          DFF_X2          NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
                                                    4.7880  n
weights_sram_read_address_reg[6]
                          DFF_X2          NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
                                                    4.7880  n
weights_sram_read_address_reg[7]
                          DFF_X2          NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
                                                    4.7880  n
weights_sram_read_address_reg[8]
                          DFF_X2          NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
                                                    4.7880  n
weights_sram_read_address_reg[9]
                          DFF_X2          NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
                                                    4.7880  n
weights_sram_read_address_reg[10]
                          DFF_X2          NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
                                                    4.7880  n
weights_sram_read_address_reg[11]
                          DFF_X2          NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
                                                    4.7880  n
--------------------------------------------------------------------------------
Total 20044 cells                                   23043.5799
1
```

Timing_min_fast_holdcheck_tut1

timing_min_fast_holdcheck_tut1.rpt - Notepad

File Edit Format View Help

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

****************************************
Report : timing
        -path full
        -delay min
        -max_paths 1
Design : MyDesign
Version: S-2021.06-SP3
Date   : Sun Nov 27 16:19:45 2022
****************************************

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

  Startpoint: output_sram_read_address_reg[11]
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: output_sram_read_address_reg[11]
            (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
  output_sram_read_address_reg[11]/CK (DFF_X2)          0.0000     0.0000 r
  output_sram_read_address_reg[11]/QN (DFF_X2)          0.0503     0.0503 f
  U19498/ZN (NAND2_X1)                                  0.0169     0.0672 r
  U19499/ZN (OAI22_X1)                                  0.0232     0.0904 f
  output_sram_read_address_reg[11]/D (DFF_X2)           0.0000     0.0904 f
  data arrival time                                                0.0904

  clock clk (rise edge)                                 0.0000     0.0000
  clock network delay (ideal)                           0.0000     0.0000
  clock uncertainty                                     0.0500     0.0500
  output_sram_read_address_reg[11]/CK (DFF_X2)          0.0000     0.0500 r
  library hold time                                    -0.0005     0.0495
  data required time                                               0.0495
  -----------------------------------------------------------------------
  data required time                                               0.0495
  data arrival time                                               -0.0904
  -----------------------------------------------------------------------
  slack (MET)                                                      0.0409
```

Timing_max_slow_holdfixed_tut1

timing_max_slow_holdfixed_tut1.rpt - Notepad

File Edit Format View Help

```
****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : MyDesign
Version: S-2021.06-SP3
Date   : Sun Nov 27 16:19:54 2022
****************************************

Operating Conditions: slow   Library: NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
Wire Load Model Mode: top

  Startpoint: R7b_reg[6] (rising edge-triggered flip-flop clocked by clk)
  Endpoint: MAC4_reg[15]
            (rising edge-triggered flip-flop clocked by clk)
  Path Group: clk
  Path Type: max

  Point                                     Incr       Path
  -----------------------------------------------------------
  clock clk (rise edge)                   0.0000     0.0000
  clock network delay (ideal)             0.0000     0.0000
  R7b_reg[6]/CK (DFF_X2)                  0.0000     0.0000 r
  R7b_reg[6]/Q (DFF_X2)                   0.6292     0.6292 f
  U12780/ZN (XNOR2_X2)                    0.2669     0.8961 r
  U6554/ZN (NAND2_X2)                     0.1618     1.0579 f
  U3978/ZN (INV_X1)                       0.1190     1.1769 r
  U8170/ZN (NAND2_X1)                     0.0754     1.2523 f
  U8169/ZN (NAND2_X1)                     0.2101     1.4624 r
  U13126/ZN (OR2_X1)                      0.2481     1.7105 r
  U8842/ZN (XNOR2_X1)                     0.3616     2.0721 r
  U12811/ZN (XNOR2_X2)                    0.3647     2.4368 r
  U14831/S (FA_X1)                        0.7057     3.1425 f
  U3086/ZN (INV_X1)                       0.1361     3.2786 r
  U5766/ZN (NAND2_X2)                     0.0679     3.3466 f
  U5765/ZN (NAND2_X2)                     0.1314     3.4780 r
  U5261/ZN (INV_X4)                       0.0661     3.5441 f
  U1820/ZN (AND2_X1)                      0.1639     3.7080 f
  U4571/ZN (NAND2_X2)                     0.0861     3.7941 r
  U4569/ZN (NAND2_X2)                     0.0650     3.8591 f
  U4568/ZN (XNOR2_X2)                     0.2410     4.1001 f
  U4567/ZN (NAND2_X2)                     0.0884     4.1885 r
  U1541/ZN (NAND3_X1)                     0.0910     4.2794 f
  U1558/ZN (INV_X1)                       0.0962     4.3757 r
  U5002/ZN (NAND2_X2)                     0.0651     4.4408 f
  MAC4_reg[15]/D (DFF_X1)                 0.0000     4.4408 f
  data arrival time                                  4.4408

  clock clk (rise edge)                   4.8000     4.8000
  clock network delay (ideal)             0.0000     4.8000
  clock uncertainty                      -0.0500     4.7500
  MAC4_reg[15]/CK (DFF_X1)                0.0000     4.7500 r
  library setup time                     -0.3091     4.4409
  data required time                                 4.4409
  -----------------------------------------------------------
  data required time                                 4.4409
  data arrival time                                 -4.4408
  -----------------------------------------------------------
  slack (MET)                                        0.0001
```

Timing_max_slow

timing_max_slow.rpt - Notepad

File Edit Format View Help

```
****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : MyDesign
Version: S-2021.06-SP3
Date   : Sun Nov 27 16:19:34 2022
****************************************

Operating Conditions: slow   Library: NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
Wire Load Model Mode: top

  Startpoint: R7b_reg[6] (rising edge-triggered flip-flop clocked by clk)
  Endpoint: MAC4_reg[15]
            (rising edge-triggered flip-flop clocked by clk)
  Path Group: clk
  Path Type: max

  Point                                    Incr       Path
  -----------------------------------------------------------
  clock clk (rise edge)                  0.0000     0.0000
  clock network delay (ideal)            0.0000     0.0000
  R7b_reg[6]/CK (DFF_X2)                 0.0000     0.0000 r
  R7b_reg[6]/Q (DFF_X2)                  0.6292     0.6292 f
  U12780/ZN (XNOR2_X2)                   0.2669     0.8961 r
  U6554/ZN (NAND2_X2)                    0.1618     1.0579 f
  U3978/ZN (INV_X1)                      0.1190     1.1769 r
  U8170/ZN (NAND2_X1)                    0.0754     1.2523 f
  U8169/ZN (NAND2_X1)                    0.2101     1.4624 r
  U13126/ZN (OR2_X1)                     0.2481     1.7105 r
  U8842/ZN (XNOR2_X1)                    0.3616     2.0721 r
  U12811/ZN (XNOR2_X2)                   0.3647     2.4368 r
  U14831/S (FA_X1)                       0.7057     3.1425 f
  U3086/ZN (INV_X1)                      0.1361     3.2786 r
  U5766/ZN (NAND2_X2)                    0.0679     3.3466 f
  U5765/ZN (NAND2_X2)                    0.1314     3.4780 r
  U5261/ZN (INV_X4)                      0.0661     3.5441 f
  U1820/ZN (AND2_X1)                     0.1639     3.7080 f
  U4571/ZN (NAND2_X2)                    0.0861     3.7941 r
  U4569/ZN (NAND2_X2)                    0.0650     3.8591 f
  U4568/ZN (XNOR2_X2)                    0.2410     4.1001 f
  U4567/ZN (NAND2_X2)                    0.0884     4.1885 r
  U1541/ZN (NAND3_X1)                    0.0910     4.2794 f
  U1558/ZN (INV_X1)                      0.0962     4.3757 r
  U5002/ZN (NAND2_X2)                    0.0651     4.4408 f
  MAC4_reg[15]/D (DFF_X1)                0.0000     4.4408 f
  data arrival time                                 4.4408

  clock clk (rise edge)                  4.8000     4.8000
  clock network delay (ideal)            0.0000     4.8000
  clock uncertainty                     -0.0500     4.7500
  MAC4_reg[15]/CK (DFF_X1)               0.0000     4.7500 r
  library setup time                    -0.3091     4.4409
  data required time                                4.4409
  -----------------------------------------------------------
  data required time                                4.4409
  data arrival time                                -4.4408
  -----------------------------------------------------------
  slack (MET)                                       0.0001
```

6. CONCLUSION:

- A deep neural network with Convolution and max poling was designed and verified.
- The trade-off between area and clock period is observed.
- The importance of designing the data path first and then going for the control path was realized.