ECE 564. Hardware Accelerator For Convolutional Neural Network Project Report

StudentID: 200199764
Name : Ajay Rao

Delay (ns to run provided example): Clock period: 21ns # cycles: 622	Logic Area: 16892.0642 (um^2) Memory: N/A	1/(delay.area) (ns^{-1}.um^{-2}) 4.532e-9
Delay (TA provided example. TA to complete)		1/(delay.area) (TA)

Abstract

A hardware accelerator is designed to accelerate the calculation of simplified two stage version of Convolutional Neural Network. The first layer is a feature extraction layer from the input and the second layer is a fully connected layer to identify classes. The 12x12 input matrix is stored in SRAM (Input memory) along with the four 9x1 B vectors and eight 64x1 M vectors (Vector memory). The 8x1 output vector is also written back to SRAM (Output memory). The design intends to balance the tradeoffs between area of the chip and delay to complete the computation. To overcome the possible contention of vector memory bus, all the elements of B vectors are fetched from the SRAM and stored in internal registers before the starting the computations. Removal of this contention facilitated a two stage pipelined design, where the feature extraction (step 1) and class identification (step 2) computations were parallelized.

1. Introduction

- ***<u>Hardware being designed:</u>***

  Convolutional neural networks have been one of the most influential innovations in the field of computer vision. A host of companies have been using deep learning at the core of their services. Facebook uses neural nets for their automatic tagging algorithms, Google for their photo search, Amazon for their product recommendations, and Instagram for their search infrastructure. However the most popular use case of these networks is for image recognition applications because of their high accuracy, which they achieve by emulating how our own brain recognizes objects.

  By making our electronic devices recognize their surroundings, a vast number of potential useful applications have been spawned. They can be used for video surveillance, mobile robot vision, image search in data centers, and much more. This has also led to a demand for methods that can compute these computational-intensive networks in a fast and power efficient way, so that applications on mobile platforms and data centers can be supported. One such method is by using application specific hardware accelerators.

  In this project, a hardware accelerator is designed to accelerate the calculation of simplified two stage version of Convolutional Neural Network. The first layer is a feature extraction layer from the input and the second layer is a fully connected layer to identify classes.

- ***<u>Structure of the rest of this report:</u>***

  In Micro-Architecture section, high level architecture drawing with interfaces will be presented and data path will be explained. In Interface specification section, detailed description of top level interface of the design will be given. In Technical Implementation section detailed hierarchy of the design will be explained and controller functionality will be explained. After that verification approach, results achieved will be presented.

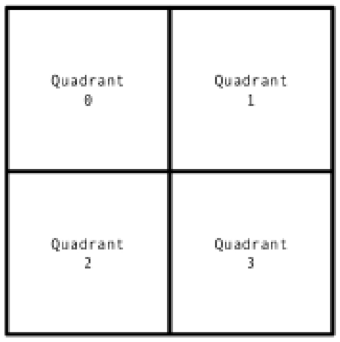
2. Micro-Architecture

- ***<u>Hardware “algorithmic” approach used:</u>***

  In this project, the computations involved in Convolutional Neural Networks are divided into two steps. These are outlined in the project specification document and a brief of the same is reproduced below:

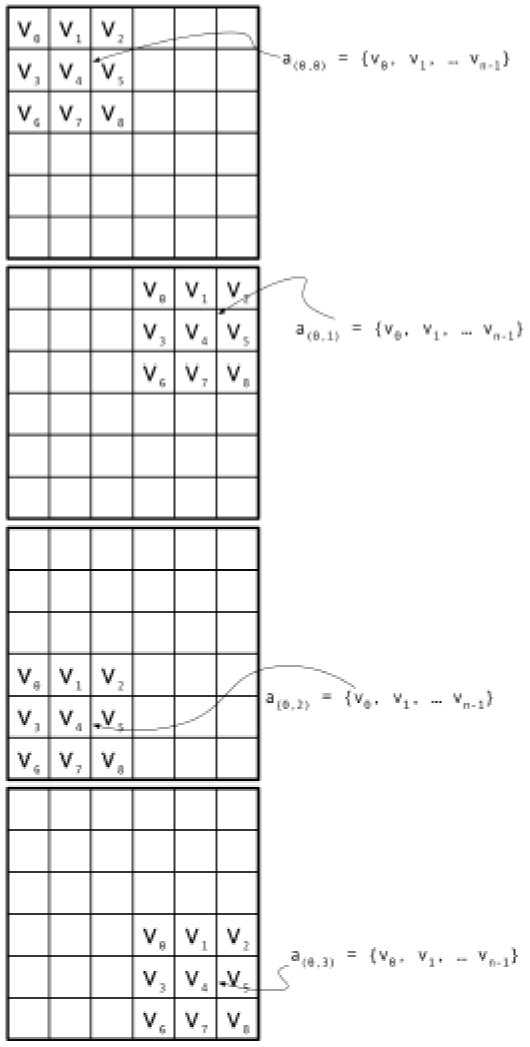
  Step 1: In this step we perform the dot product on regions of the input using four vectors (B) from the filter vector memory.

I. The 12x12 input matrix is divided as four 6x6 quadrants as shown below,



II. From each quadrant, four 1x9 vectors [a(q,i)] will be formed as shown below,

$$a_{(q,i)} \quad \begin{array}{l} q = 0, \ldots, 3 \\ i = 0, \ldots, 3 \end{array}$$


V_0 V_1 V_2
V_3 V_4 V_5
V_6 V_7 V_8
$a_{(0,0)} = \{v_0, v_1, \ldots v_{n-1}\}$
$a_{(0,1)} = \{v_0, v_1, \ldots v_{n-1}\}$
$a_{(0,2)} = \{v_0, v_1, \ldots v_{n-1}\}$
$a_{(0,3)} = \{v_0, v_1, \ldots v_{n-1}\}$

III. Next we perform a dot product of each of these [a(q,i)] vectors against each of the four $[b]_{0-3}$ vectors.

$$b = [b_0, b_1 ... b_8]$$

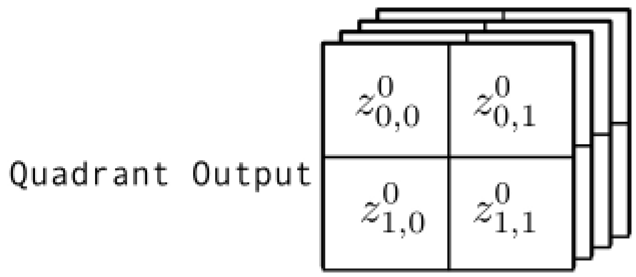
$$a = [a_0, a_1 ... a_8]$$

$$c = \sum_{n=0}^{8} a_i \cdot b_i$$

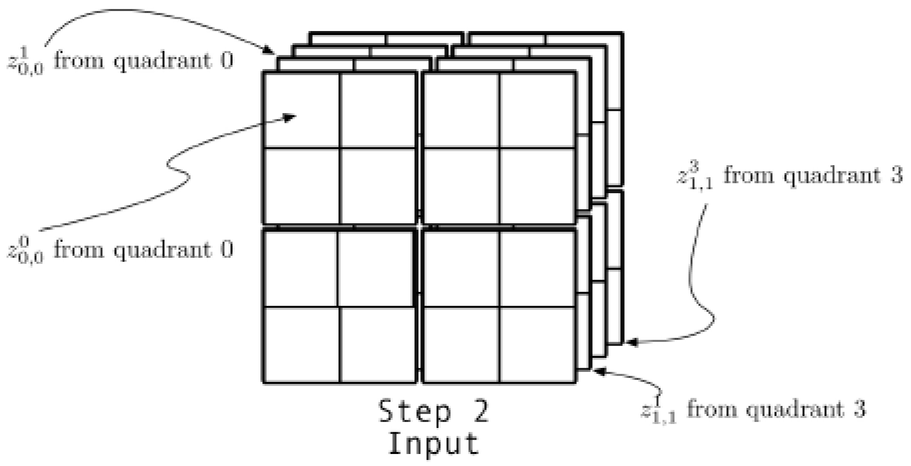
IV. The 4 int 32 c values need to be truncated to 16 bit values to form the intermediate z values that needs to be provided as an input to next stage.

$$z^b_{i,j} = f(c^b_{i,j}) = \begin{cases} c_i & , c_i \geq 0 \\ 0 & , otherwise \end{cases} \text{ where } b = 0 \ldots 3, i = 0, 1, j = 0, 1$$

V. This operation on each quadrant will form a 2x2x4 output array as shown below,



VI. This operation when performed for all B vectors on each quadrant gives us 4 2x2x4 output arrays which are arranged to for one 4x4x4 matrix as shown below,


$z^1_{0,0}$ from quadrant 0
$z^0_{0,0}$ from quadrant 0
$z^3_{1,1}$ from quadrant 3
$z^1_{1,1}$ from quadrant 3
Step 2
Input

Step 2: In this step we perform the dot products on the step one output (4x4x4 matrix) with a set of 8 64x1 M filter vectors. The output matrix from step one will be arranged in row major fashion starting from first layer for this dot product. The output of this dot product will be 8 int 32 c elements, that needs to be truncated to give the final O output.

$$m_i = [m_{i,0}, m_{i,1}...m_{i,63}]$$

$$u = [u_0, u_1...u_63]$$

$$w_i = \sum_{n=0}^{63} m_i \cdot u, i = 0, \ldots, 7$$

$$O_i = f(w_i) = \begin{cases} w_i & , w_i \geq 0 \\ 0 & , otherwise \end{cases} \text{ where } i = 0 \ldots 7$$

- ***High level architecture drawing, and description of data flow:***

Figure 1. below shows the high level architecture drawing of the design,

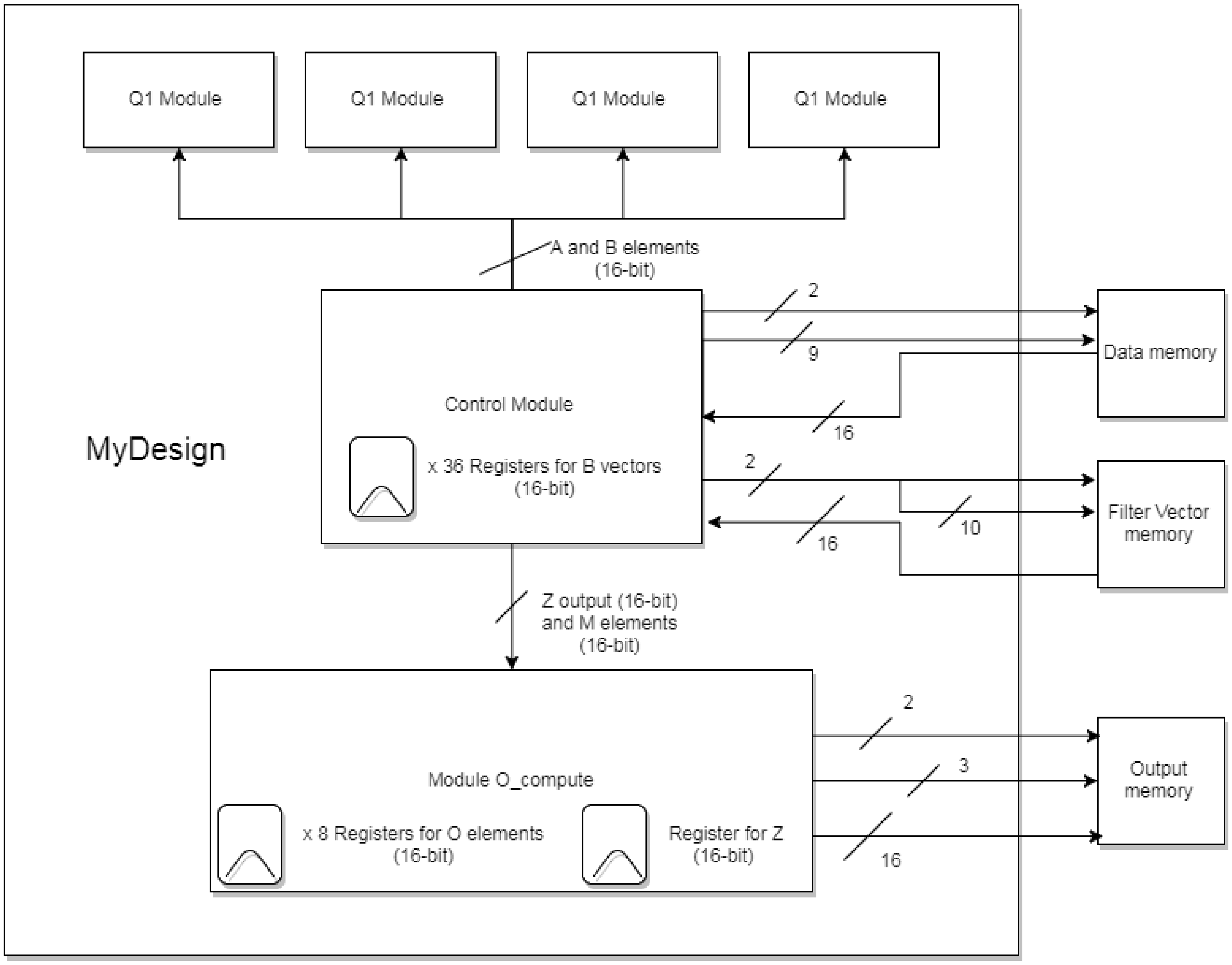


Figure 1. High level architecture

The design is divided into 4 modules: MyDesign, Control, Q and O_compute. MyDesign module is the top module that instantiates all other modules and interfaces with the test bench. Further in MyDesign there are 4 instantiations of Q module, one each for a quadrant. They work on the elements for their respective quadrants (Step 1) and produce the dot product Z. The control module is the brain of the design. Control module is responsible for generating the address of matrix A, vectors B and M in the proper sequence. Control module passes the read data from the SRAMs to other module that use this data for computation. The O_compute module computes the dot product of Z and M vectors to give the final output vector O. This vector is written back to the output SRAM by the O_compute module.
The data path for the design is as shown in Figure 2. below,

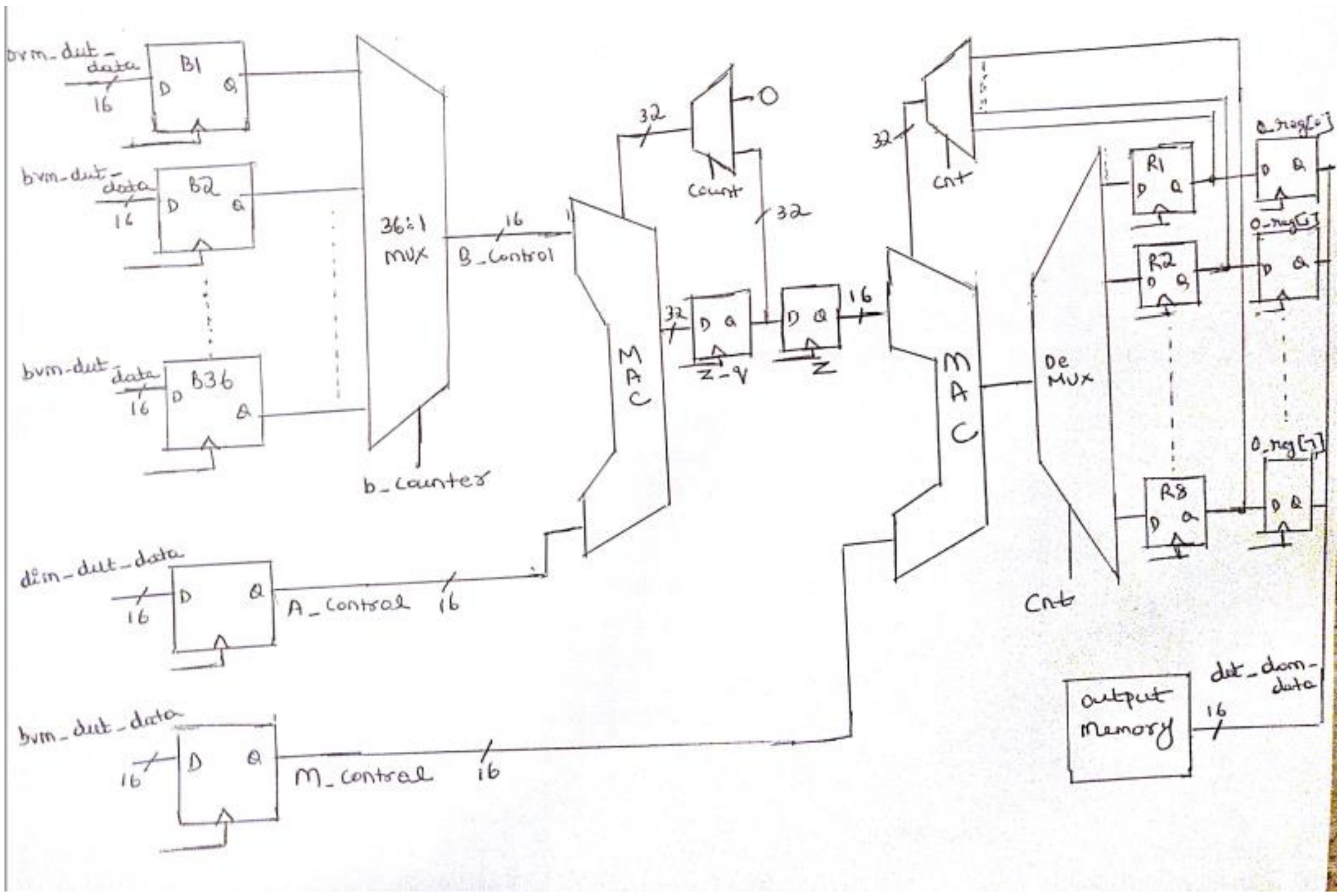


Figure 2. Data path of the design

The B data read from the filter vector memory are stored using internal registers. Then this stored B vector elements are passed on to the MAC unit to be multiplied with the A matrix elements. The sequencing of which B vector elements have to be passed when is taken care by the control module. The output of this MAC is then truncated to 16-bit value before being passed to another MAC unit, which accumulates the product of Z and M vector elements. There are 8 internal registers to store the MAC output (one each for each output vector element). A de-multiplexer is used to route the MAC output to appropriate register. After all the 64 elements of Z and M have been processed, the output of each register is truncated to 16-bit value before being written to output SRAM.

3. Interface Specification

- ***Detailed description of top level interface to the design:***

  The top module (MyDesign) interfaces with the test bench. The test bench provides the control signal 'go' for the start of computation along with the clock and global reset signals. The top module also sends out a 'finish' signal to the test bench indicating the completion of computation for the given set of data. Top module also interfaces with the three SRAMs through the test bench. The top module sends the signals enable and write enable to configure the SRAM for intended functionality. Also the top module sends the address and data to be written on their respective busses to the test bench, which in turn interfaces with the SRAM module. The read data from SRAM is passed from the test bench to the top module as its inputs.
  Figure 3. below highlights the input and output ports of the top module.

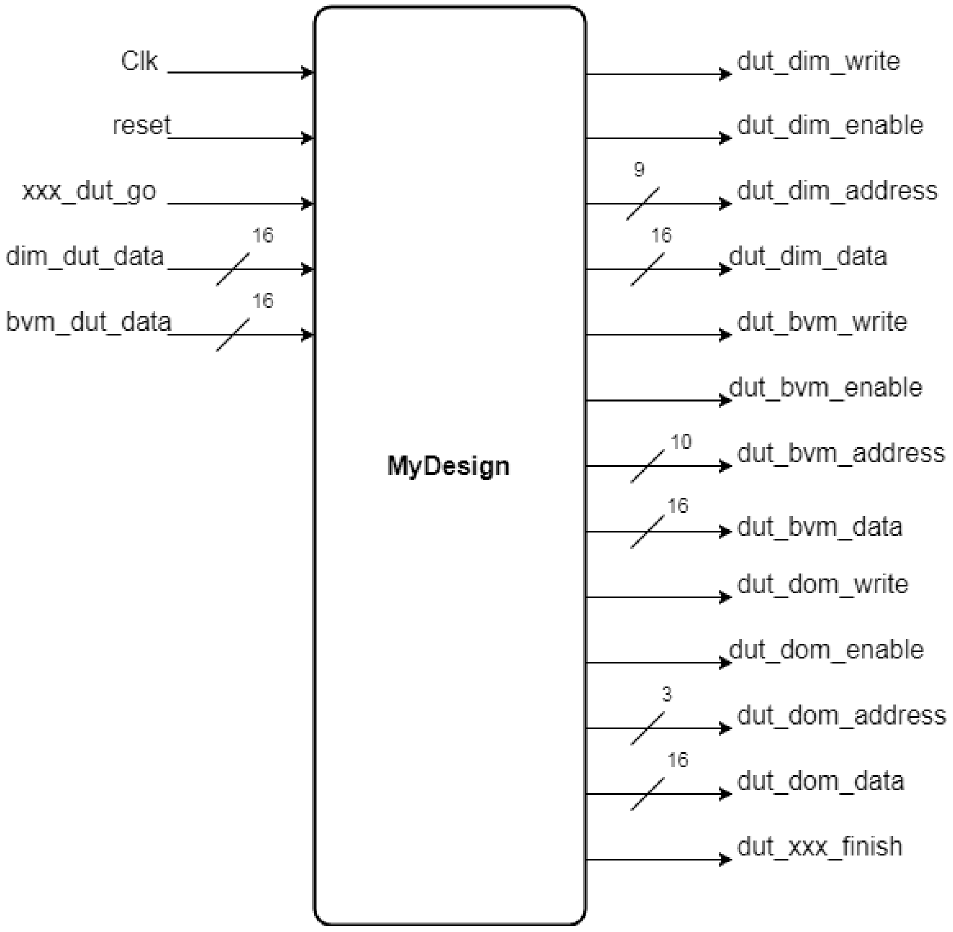


Figure 3. Interface schematic of the design.

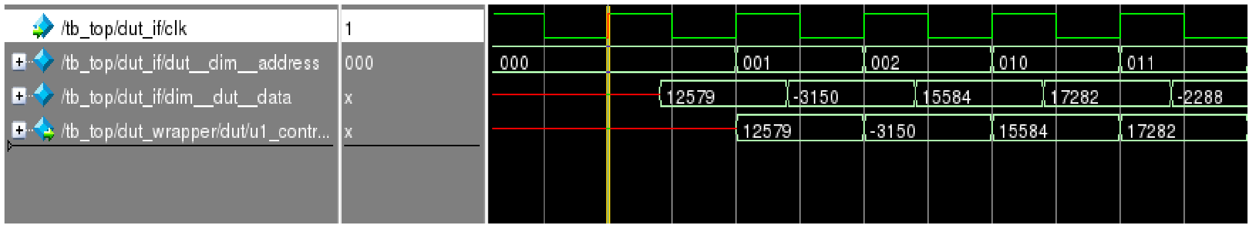
- ***Summary table listing each signal, its direction, width and function:***

Signal	Direction	Width	Function
clk	Input	1	Clock pin that receives clock pulses from the testbench.
reset	Input	1	Global reset to the design
xxx_dut_go	Input	1	Signal to start calculation on the new set of input matrix and vectors.
dim_dut_data	Input	16	16-bit bus containing read data from SRAM 0 (Input matrix A)
bvm_dut_data	Input	16	16-bit bus containing read data from SRAM 1 (Vectors B and M)
dut_dim_write	Output	1	Write enable signal for SRAM 0 (1 while writing, 0 while reading)
dut_dim_enable	Output	1	SRAM 0 enable signal (1 to enable SRAM, 0 to disable)
dut_dim_address	Output	9	9-bit address bus for SRAM 0
dut_dim_data	Output	16	16-bit write bus containing data to be written to SRAM 0
dut_bvm_write	Output	1	Write enable signal for SRAM 1 (1 while writing, 0 while reading)
dut_bvm_enable	Output	1	SRAM 1 enable signal (1 to enable SRAM, 0 to disable)
dut_bvm_address	Output	10	10-bit address bus for SRAM 1
dut_bvm_data	Output	16	16-bit write bus containing data to

			be written to SRAM 1
dut_dom_write	Output	1	Write enable signal for SRAM 2 (1 while writing, 0 while reading)
dut_dom_enable	Output	1	SRAM 0 enable signal (1 to enable SRAM, 0 to disable)
dut_dom_address	Output	3	3-bit address bus for SRAM 2
dut_dom_data	Output	16	16-bit write bus containing data to be written to SRAM 2
dut_xxx_finish	Output	1	Signal from Design to indicate completion of computation and is ready for next set of inputs.

- ***Interface timing diagram:***

The interface timing diagram with SRAM is as shown below in Figure 4.



/tb_top/dut_if/clk
1
/tb_top/dut_if/dut__dim__address
000
/tb_top/dut_if/dim__dut__data
x
/tb_top/dut_wrapper/dut/u1_contr...
x
000
001
002
010
011
12579
-3150
15584
17282
-2288
12579
-3150
15584
17282

Figure 4. Timing diagram for SRAM.

As per the project specification, the read data from SRAM is available to the design one clock cycle after the address has been sent out on the address bus. The address and read data from SRAM are registered using flops. As it can be seen from the above diagram, the address 0x00 is sent out on the clock edge that is highlighted by the yellow line. Though the data 12579 is read from SRAM during the same cycle, it is available to the design only on the next rising edge of the clock (as seen in the third row). The read data from xxx_dut_data is connected to a flop that preserves this data for a clock cycle, until the next data arrives.

4. Technical Implementation

- ***Pipelined architecture implemented:***

In order to reduce the delay in computation, the stage 1 and stage 2 of the calculation has been pipelined in this design. Figure 5. below outlines this pipeline architecture,

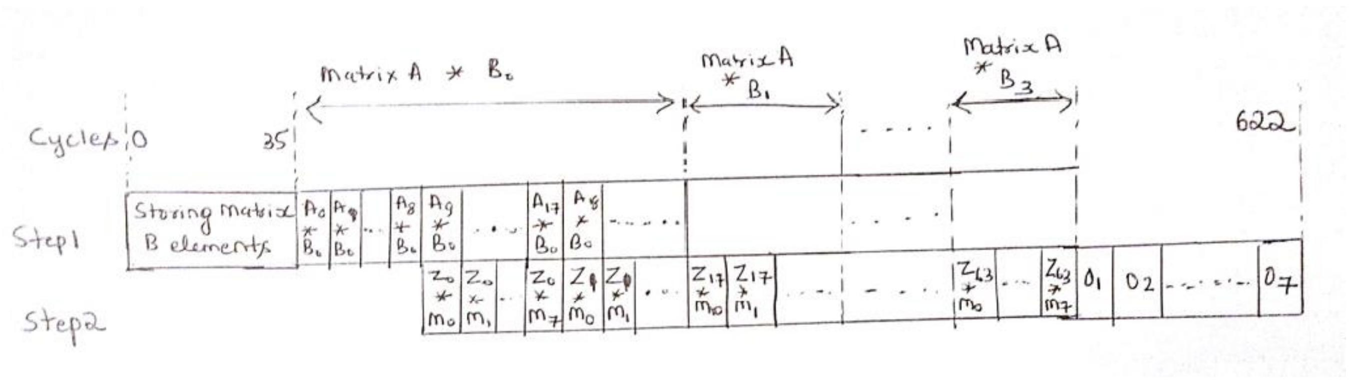


Figure 5. Overview of pipelined architecture

I. When the 'go' signal is asserted by the test bench, vector B is fetched from Filter Vector memory for the first 36 cycles and stored in internal registers.
II. From the next cycle onwards an element from input matrix A is fetched every clock cycle.
III. From the same cycle vector B_0 elements are also fetched in a cyclic order, repeating every 9 cycles.
IV. When the first 9 elements of A and B_0 vectors have been multiplied and accumulated we have our first Z element ready for being used to compute O.
V. As we have already stored B vectors, we will avoid the contention for the use of SRAM 1 readbus and hence we start fetching a new M vector element every cycle.
VI. The Z value generated at the end of every 9th cycle is retained for the next 9 cycles until the new value is generated.
VII. During this 9 cycles Z is multiplied with the M vector element that is being fetched. One element form each M vector (M_0 to M_7) are fetched. The address of the element is calculated by the control module.
VIII. After sweeping through all the elements of the matrix i.e. 144 cycles later, the process is repeated again for 3 more times. At the end of this we would have swept through matrix A 4 times. Thus having finished the multiplication of the input matrix A with all the 4 B vectors and the corresponding Z being multiplied with all the 8 M vectors.
IX. As shown in the diagram above after 44 cycles both stages of multiplication go in parallel.
X. At the end of 622 cycles we will have completed the computation of all the 8 output elements and would have written it to SRAM.

- ***Control Module implementation:***

The control module implements a two state FSM as shown in Figure 6.

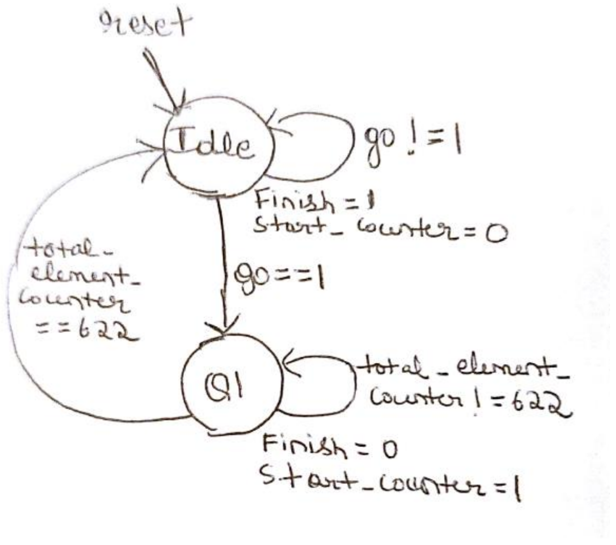


Figure 6. FSM

Idle State: The design waits for the 'go' signal to be asserted by the test bench in the Idle state. 'Finish' signal is made high in this state along with setting 'start_counter' signal to low.

Q1 State: Q1 state spawns multiple counters that aid in address calculation and synchronization of inputs and outputs. The overview of these operations are given below:

I. A master counter named 'total_element_count' counts up every clock cycle when 'start_counter' goes high. This counter is used to transition back to Idle state when all the outputs have been written to SRAM.

II. Upon setting 'start_counter', an up counter for counting 36 gets rolling. It counts up every clock cycle. This counter is used to fetch the address of B vector elements from a look up table. The counter is reset all the outputs have been written to output SRAM.

III. After fetching all the B vector elements, signal 'start_a' is made high.

IV. Upon setting 'start_a' high, two counters 'element_counter' and 'im_base_count' are initialized. Both are up counters. 'element_counter' counts till 36 and 'im_base_count' counts till 4. These counters are used to fetch the offset and base address for input matrix A from the look up table.

The Pseudo Code for these counters are as follows,

```
If element_counter == 35
 then
        element_counter = 0
        If im_base_count == 3 then im_base_count = 0
        Else im_base_count = im_base_count+1
 end
Else element_counter = element_counter +1
```

The counters are reset after all the outputs have been written to output SRAM.

V. After fetching the first 9 elements of input matrix A , the design starts fetching vector M elements in parallel with A. This ensures that Z and O calculations go in parallel. To fetch the offset and base address of M vector from the look up table, two up counters moffset_counter and mbase_counter are initialized.
The Pseudo Code for these counters are as follows,

```
If mbase_counter == 8
 then
        moffset_counter = moffset_counter + 1
        mbase_counter = 0
 end
Else mbase_counter = mbase_counter + 1
```

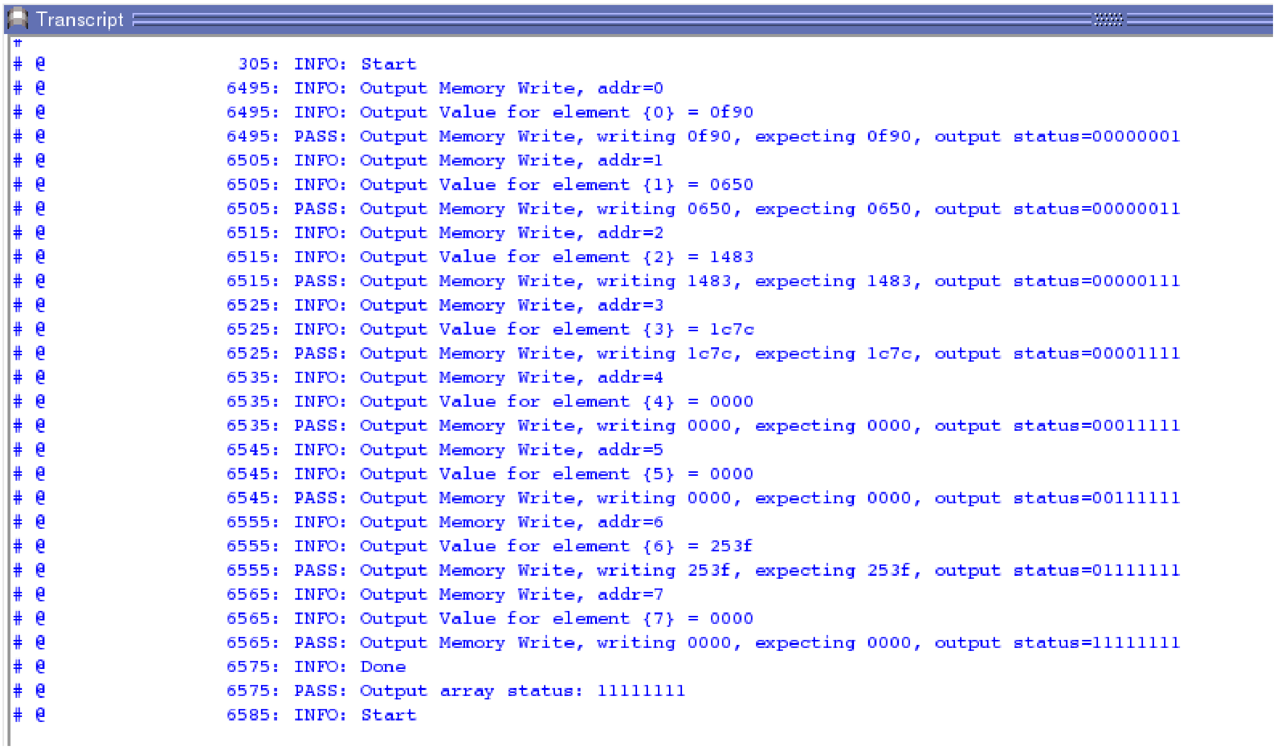
The counters are reset after all the outputs have been written to output SRAM.

VI. Control module makes use of the 'total_element_counter' to facilitate the supply of correct B vector elements after the initial 36 cycles when it is stored in the internal registers. The registers to fetch B vector elements changes every 144 cycles, i.e. after sweeping through matrix A once. Within that 144 cycle, each element of one B vector is cycled every 9 cycles.

VII. The control module also provides the start signals for each Q module and the O_compute module to start processing the data.

5. Verification

A test bench that contains three SRAMs (Input matrix A - 512*16, Filter vector memory1024*16 and Output memory 8*16), a logic to generate clock, global reset, 'go' signal upon receiving 'finish' signal is used to verify the design. The test bench is a system verilog file. The test bench uses randomization function to generate the input matrix A elements along with the filter vector B and M elements. The test bench also performs computations on these inputs and prints out the intermediate results as well as the final output vector O on the terminal.



```
#
# @                    305: INFO: Start
# @                   6495: INFO: Output Memory Write, addr=0
# @                   6495: INFO: Output Value for element {0} = 0f90
# @                   6495: PASS: Output Memory Write, writing 0f90, expecting 0f90, output status=00000001
# @                   6505: INFO: Output Memory Write, addr=1
# @                   6505: INFO: Output Value for element {1} = 0650
# @                   6505: PASS: Output Memory Write, writing 0650, expecting 0650, output status=00000011
# @                   6515: INFO: Output Memory Write, addr=2
# @                   6515: INFO: Output Value for element {2} = 1483
# @                   6515: PASS: Output Memory Write, writing 1483, expecting 1483, output status=00000111
# @                   6525: INFO: Output Memory Write, addr=3
# @                   6525: INFO: Output Value for element {3} = 1c7c
# @                   6525: PASS: Output Memory Write, writing 1c7c, expecting 1c7c, output status=00001111
# @                   6535: INFO: Output Memory Write, addr=4
# @                   6535: INFO: Output Value for element {4} = 0000
# @                   6535: PASS: Output Memory Write, writing 0000, expecting 0000, output status=00011111
# @                   6545: INFO: Output Memory Write, addr=5
# @                   6545: INFO: Output Value for element {5} = 0000
# @                   6545: PASS: Output Memory Write, writing 0000, expecting 0000, output status=00111111
# @                   6555: INFO: Output Memory Write, addr=6
# @                   6555: INFO: Output Value for element {6} = 253f
# @                   6555: PASS: Output Memory Write, writing 253f, expecting 253f, output status=01111111
# @                   6565: INFO: Output Memory Write, addr=7
# @                   6565: INFO: Output Value for element {7} = 0000
# @                   6565: PASS: Output Memory Write, writing 0000, expecting 0000, output status=11111111
# @                   6575: INFO: Done
# @                   6575: PASS: Output array status: 11111111
# @                   6585: INFO: Start
```

Figure 6. Snapshot of output verification on terminal window

The test bench verifies if all the output elements that the design writes to the output SRAM are of correct value and are written to the correct address.
As shown in the figure 6 above, the test bench prints out the status of each output memory location written. The test bench compared the value written by the DUT with the value that it has calculated while performing the same computation. The test bench prints out the PASS/FAIL status along with keeping a tab on which write(s) failed by populating the output status variable with 1 if PASS or 0 if FAIL.

6. Results Achieved

- ***Timing Reports:***

  - max_slow

    Information: Updating design information... (UID-85)
    Warning: Design 'MyDesign' contains 1 high-fanout nets. A fanout number of 1000 will be used for delay calculations involving these nets. (TIM-134)

    ```
    ****************************************
    Report : timing
    -path full
    -delay max
    -max_paths 1
    Design : MyDesign
    Version: K-2015.06-SP1
    Date   : Thu Nov  2 20:20:22 2017
    ```

```
****************************************

# A fanout number of 1000 was used for high fanout net computations.

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

Startpoint: u1_control/total_element_count_reg[3]
(rising edge-triggered flip-flop clocked by clk)
Endpoint: u1_O/R1_reg[31]
(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
u1_control/total_element_count_reg[3]/CK (DFFR_X2)    0.0000 #   0.0000 r
u1_control/total_element_count_reg[3]/Q (DFFR_X2)     0.5245     0.5245 r
u1_control/total_element_count[3] (control)           0.0000     0.5245 r
u1_O/element_cnt[3] (O_Compute)                 0.0000     0.5245 r
u1_O/U3171/ZN (INV_X4)                          0.0518     0.5763 f
u1_O/U2592/ZN (NOR3_X2)                         0.3759     0.9522 r
u1_O/U3169/ZN (INV_X4)                          0.0809     1.0331 f
u1_O/U71/ZN (OAI21_X2)                          0.1743     1.2074 r
u1_O/U2590/ZN (NAND3_X1)                        0.1289     1.3362 f
u1_O/U2633/ZN (AND2_X1)                         0.1697     1.5059 f
u1_O/U2661/ZN (OR3_X2)                          0.4151     1.9210 f
u1_O/U74/ZN (NAND4_X2)                          0.3307     2.2517 r
u1_O/U52/ZN (OR2_X4)                            0.2162     2.4679 r
u1_O/U331/ZN (NOR2_X2)                          0.0554     2.5233 f
u1_O/U404/ZN (INV_X1)                           0.1822     2.7055 r
u1_O/U368/ZN (INV_X4)                           0.0859     2.7914 f
u1_O/U2624/ZN (AOI22_X1)                        0.3505     3.1419 r
u1_O/U1918/ZN (OAI221_X2)                       0.2345     3.3764 f
u1_O/U1917/ZN (INV_X4)                          0.4209     3.7973 r
u1_O/U461/ZN (XNOR2_X1)                         0.4093     4.2066 r
u1_O/U1563/ZN (INV_X4)                          0.1733     4.3799 f
u1_O/U1562/ZN (NAND2_X2)                        0.5551     4.9350 r
u1_O/U1892/ZN (OAI22_X2)                        0.2511     5.1861 f
u1_O/U1853/Z (XOR2_X2)                          0.3240     5.5101 f
u1_O/U1852/ZN (XNOR2_X2)                        0.3323     5.8423 f
u1_O/U1797/Z (XOR2_X2)                          0.4160     6.2584 f
u1_O/U1796/ZN (XNOR2_X2)                        0.3105     6.5688 f
u1_O/U73/ZN (OAI21_X2)                          0.2073     6.7762 r
u1_O/U72/ZN (OAI21_X2)                          0.1536     6.9297 f
u1_O/U779/ZN (OAI21_X1)                         0.2884     7.2181 r
u1_O/U2720/ZN (INV_X4)                          0.0366     7.2547 f
u1_O/U778/ZN (AOI21_X1)                         0.6011     7.8558 r
u1_O/U1118/ZN (XNOR2_X2)                        0.1512     8.0069 f
u1_O/U1117/Z (XOR2_X2)                          0.3470     8.3539 f
u1_O/r662/U1_9/CO (FA_X1)                       0.5759     8.9298 f
u1_O/r662/U1_10/CO (FA_X1)                      0.5172     9.4470 f
u1_O/r662/U1_11/CO (FA_X1)                      0.5172     9.9643 f
u1_O/r662/U1_12/CO (FA_X1)                      0.5172    10.4815 f
```

```
u1_O/r662/U1_13/CO (FA_X1)                    0.5172    10.9988 f
u1_O/r662/U1_14/CO (FA_X1)                    0.5172    11.5160 f
u1_O/r662/U1_15/CO (FA_X1)                    0.5172    12.0333 f
u1_O/r662/U1_16/CO (FA_X1)                    0.5172    12.5505 f
u1_O/r662/U1_17/CO (FA_X1)                    0.5172    13.0677 f
u1_O/r662/U1_18/CO (FA_X1)                    0.5172    13.5850 f
u1_O/r662/U1_19/CO (FA_X1)                    0.5172    14.1022 f
u1_O/r662/U1_20/CO (FA_X1)                    0.5172    14.6195 f
u1_O/r662/U1_21/CO (FA_X1)                    0.5172    15.1367 f
u1_O/r662/U1_22/CO (FA_X1)                    0.5172    15.6540 f
u1_O/r662/U1_23/CO (FA_X1)                    0.5172    16.1712 f
u1_O/r662/U1_24/CO (FA_X1)                    0.5172    16.6885 f
u1_O/r662/U1_25/CO (FA_X1)                    0.5172    17.2057 f
u1_O/r662/U1_26/CO (FA_X1)                    0.5172    17.7229 f
u1_O/r662/U1_27/CO (FA_X1)                    0.5172    18.2402 f
u1_O/r662/U1_28/CO (FA_X1)                    0.5172    18.7574 f
u1_O/r662/U1_29/CO (FA_X1)                    0.5172    19.2747 f
u1_O/r662/U1_30/CO (FA_X1)                    0.5172    19.7919 f
u1_O/r662/U1_31/S (FA_X1)                    0.6334    20.4253 f
u1_O/U367/ZN (NAND2_X2)                       0.1471    20.5724 r
u1_O/U238/ZN (NAND2_X2)                       0.0680    20.6404 f
u1_O/R1_reg[31]/D (DFF_X2)                   0.0000    20.6404 f
data arrival time                          20.6404

clock clk (rise edge)                   21.0000    21.0000
clock network delay (ideal)                0.0000    21.0000
clock uncertainty                       -0.0500    20.9500
u1_O/R1_reg[31]/CK (DFF_X2)                   0.0000    20.9500 r
library setup time                      -0.2979    20.6521
data required time                          20.6521
-----------------------------------------------------------------------
data required time                          20.6521
data arrival time                          -20.6404
-----------------------------------------------------------------------
slack (MET)                                        0.0117
```

- max_slow_holdfixed

```
****************************************
Report : timing
-path full
-delay max
-max_paths 1
Design : MyDesign
Version: K-2015.06-SP1
Date   : Thu Nov  2 20:21:10 2017
****************************************

# A fanout number of 1000 was used for high fanout net computations.

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

Startpoint: u1_control/total_element_count_reg[3]
```

```
(rising edge-triggered flip-flop clocked by clk)
Endpoint: u1_O/R1_reg[31]
(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
u1_control/total_element_count_reg[3]/CK (DFFR_X2)    0.0000 #   0.0000 r
u1_control/total_element_count_reg[3]/Q (DFFR_X2)     0.5245     0.5245 r
u1_control/total_element_count[3] (control)           0.0000     0.5245 r
u1_O/element_cnt[3] (O_Compute)                0.0000     0.5245 r
u1_O/U3171/ZN (INV_X4)                         0.0518     0.5763 f
u1_O/U2592/ZN (NOR3_X2)                        0.3759     0.9522 r
u1_O/U3169/ZN (INV_X4)                         0.0809     1.0331 f
u1_O/U71/ZN (OAI21_X2)                         0.1743     1.2074 r
u1_O/U2590/ZN (NAND3_X1)                       0.1289     1.3362 f
u1_O/U2633/ZN (AND2_X1)                        0.1697     1.5059 f
u1_O/U2661/ZN (OR3_X2)                         0.4151     1.9210 f
u1_O/U74/ZN (NAND4_X2)                         0.3307     2.2517 r
u1_O/U52/ZN (OR2_X4)                           0.2162     2.4679 r
u1_O/U331/ZN (NOR2_X2)                         0.0554     2.5233 f
u1_O/U404/ZN (INV_X1)                          0.1822     2.7055 r
u1_O/U368/ZN (INV_X4)                          0.0859     2.7914 f
u1_O/U2624/ZN (AOI22_X1)                       0.3505     3.1419 r
u1_O/U1918/ZN (OAI221_X2)                      0.2345     3.3764 f
u1_O/U1917/ZN (INV_X4)                         0.4209     3.7973 r
u1_O/U461/ZN (XNOR2_X1)                        0.4093     4.2066 r
u1_O/U1563/ZN (INV_X4)                         0.1733     4.3799 f
u1_O/U1562/ZN (NAND2_X2)                       0.5551     4.9350 r
u1_O/U1892/ZN (OAI22_X2)                       0.2511     5.1861 f
u1_O/U1853/Z (XOR2_X2)                         0.3240     5.5101 f
u1_O/U1852/ZN (XNOR2_X2)                       0.3323     5.8423 f
u1_O/U1797/Z (XOR2_X2)                         0.4160     6.2584 f
u1_O/U1796/ZN (XNOR2_X2)                       0.3105     6.5688 f
u1_O/U73/ZN (OAI21_X2)                         0.2073     6.7762 r
u1_O/U72/ZN (OAI21_X2)                         0.1536     6.9297 f
u1_O/U779/ZN (OAI21_X1)                        0.2884     7.2181 r
u1_O/U2720/ZN (INV_X4)                         0.0366     7.2547 f
u1_O/U778/ZN (AOI21_X1)                        0.6011     7.8558 r
u1_O/U1118/ZN (XNOR2_X2)                       0.1512     8.0069 f
u1_O/U1117/Z (XOR2_X2)                         0.3470     8.3539 f
u1_O/r662/U1_9/CO (FA_X1)                      0.5759     8.9298 f
u1_O/r662/U1_10/CO (FA_X1)                     0.5172     9.4470 f
u1_O/r662/U1_11/CO (FA_X1)                     0.5172     9.9643 f
u1_O/r662/U1_12/CO (FA_X1)                     0.5172    10.4815 f
u1_O/r662/U1_13/CO (FA_X1)                     0.5172    10.9988 f
u1_O/r662/U1_14/CO (FA_X1)                     0.5172    11.5160 f
u1_O/r662/U1_15/CO (FA_X1)                     0.5172    12.0333 f
u1_O/r662/U1_16/CO (FA_X1)                     0.5172    12.5505 f
u1_O/r662/U1_17/CO (FA_X1)                     0.5172    13.0677 f
u1_O/r662/U1_18/CO (FA_X1)                     0.5172    13.5850 f
u1_O/r662/U1_19/CO (FA_X1)                     0.5172    14.1022 f
u1_O/r662/U1_20/CO (FA_X1)                     0.5172    14.6195 f
```

```
u1_O/r662/U1_21/CO (FA_X1)                 0.5172    15.1367 f
u1_O/r662/U1_22/CO (FA_X1)                 0.5172    15.6540 f
u1_O/r662/U1_23/CO (FA_X1)                 0.5172    16.1712 f
u1_O/r662/U1_24/CO (FA_X1)                 0.5172    16.6885 f
u1_O/r662/U1_25/CO (FA_X1)                 0.5172    17.2057 f
u1_O/r662/U1_26/CO (FA_X1)                 0.5172    17.7229 f
u1_O/r662/U1_27/CO (FA_X1)                 0.5172    18.2402 f
u1_O/r662/U1_28/CO (FA_X1)                 0.5172    18.7574 f
u1_O/r662/U1_29/CO (FA_X1)                 0.5172    19.2747 f
u1_O/r662/U1_30/CO (FA_X1)                 0.5172    19.7919 f
u1_O/r662/U1_31/S (FA_X1)                 0.6334    20.4253 f
u1_O/U367/ZN (NAND2_X2)                    0.1471    20.5724 r
u1_O/U238/ZN (NAND2_X2)                    0.0680    20.6404 f
u1_O/R1_reg[31]/D (DFF_X2)                0.0000    20.6404 f
data arrival time                        20.6404

clock clk (rise edge)              21.0000    21.0000
clock network delay (ideal)           0.0000    21.0000
clock uncertainty                 -0.0500    20.9500
u1_O/R1_reg[31]/CK (DFF_X2)                0.0000    20.9500 r
library setup time                -0.2979    20.6521
data required time                        20.6521
-----------------------------------------------------------------------
data required time                        20.6521
data arrival time                        -20.6404
-----------------------------------------------------------------------
```

slack (MET)   0.0117

- min_fast_holdcheck

```
Information: Updating design information... (UID-85)
Warning: Design 'MyDesign' contains 1 high-fanout nets. A fanout number of 1000 will be used for delay
calculations involving these nets. (TIM-134)

****************************************
Report : timing
-path full
-delay min
-max_paths 1
Design : MyDesign
Version: K-2015.06-SP1
Date   : Thu Nov  2 20:20:47 2017
****************************************

# A fanout number of 1000 was used for high fanout net computations.

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

Startpoint: u_q2/count_reg[2]
(rising edge-triggered flip-flop clocked by clk)
Endpoint: u_q2/count_reg[2]
(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
u_q2/count_reg[2]/CK (DFF_X2)      0.0000 #   0.0000 r
u_q2/count_reg[2]/Q (DFF_X2)       0.0524     0.0524 r
u_q2/U95/ZN (OAI33_X1)             0.0210     0.0734 f
u_q2/count_reg[2]/D (DFF_X2)       0.0000     0.0734 f
data arrival time                             0.0734

clock clk (rise edge)              0.0000     0.0000
clock network delay (ideal)        0.0000     0.0000
clock uncertainty                  0.0500     0.0500
u_q2/count_reg[2]/CK (DFF_X2)      0.0000     0.0500 r
library hold time                 -0.0004     0.0496
data required time                            0.0496
-----------------------------------------------------------
data required time                            0.0496
data arrival time                            -0.0734
-----------------------------------------------------------
slack (MET)                                   0.0238
```

- ***Area Report:***

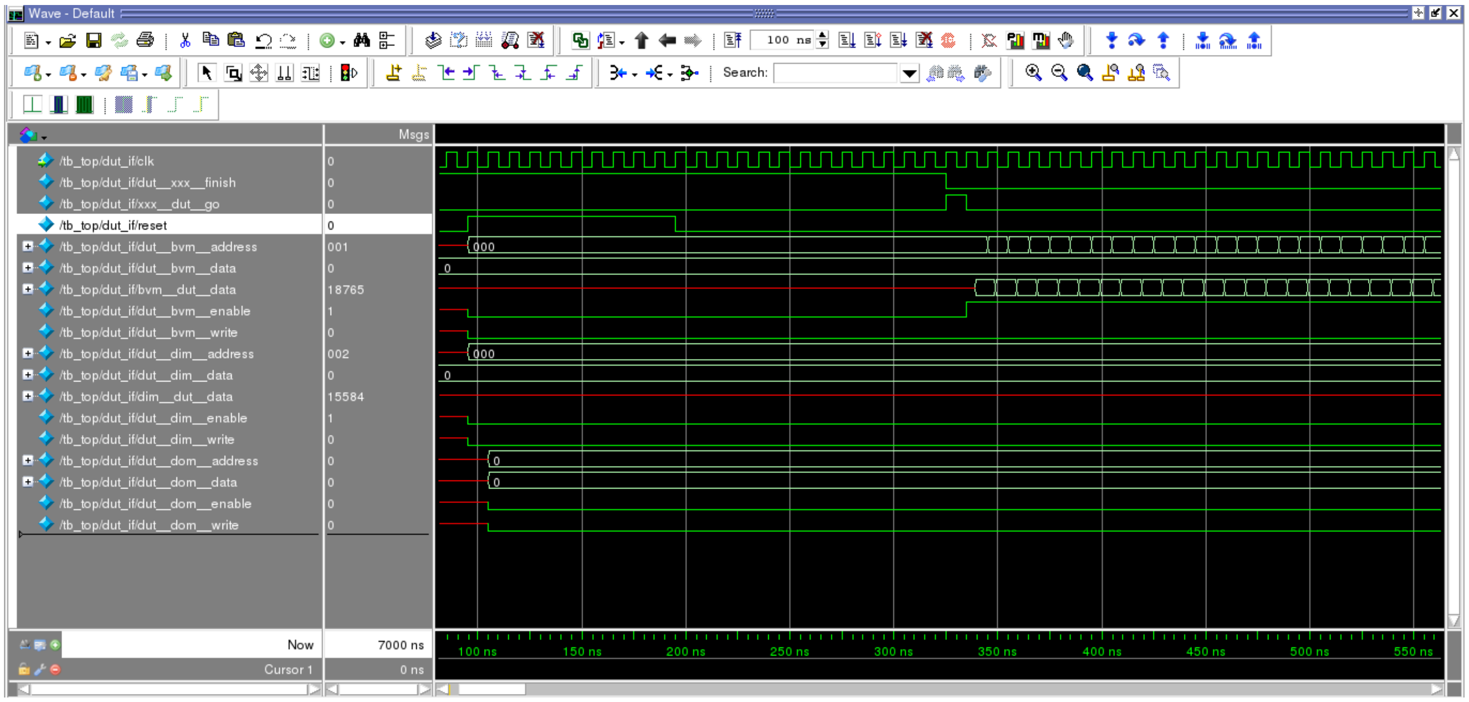
```
****************************************
Report : cell
Design : MyDesign
Version: K-2015.06-SP1
Date   : Thu Nov  2 20:21:10 2017
****************************************

Attributes:
b - black box (unknown)
h - hierarchical
n - noncombinational
r - removable
u - contains unmapped logic

Cell                 Reference       Library             Area  Attributes
--------------------------------------------------------------------------------
U1                   INV_X8          NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.7980
U71                  INV_X4          NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U72                  INV_X4          NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U73                  INV_X4          NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U74                  INV_X4          NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U75                  INV_X4          NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
```

```
0.5320
U76               INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U77               INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U78               INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U79               INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U80               INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U81               INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U82               INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U83               INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U84               INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U85               INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U86               INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U87               INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U88               INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U89               INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U90               INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U91               INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U92               INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U93               INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U94               INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U95               INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U96               INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U97               INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U98               INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U100              INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U101              INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U102              INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U103              INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U104              INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
```

```
0.5320
U105              INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U106              INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U107              INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U108              INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U109              INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U110              INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U111              INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U112              INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U113              INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U114              INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U115              INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U116              INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U117              INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
U118              INV_X4      NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
0.5320
u1_O              O_Compute            5629.0920 h, n
u1_control          control          5313.0841 h, n
u_q1             Q_1             1448.6360 h, n
u_q2             Q_3             1447.5720 h, n
u_q3             Q_2             1450.7640 h, n
u_q4             Q_0             1577.1140 h, n
-----------------------------------------------------------------------------
Total 54 cells                          16892.0642
```

- ***<u>Simulation Snapshots:</u>***

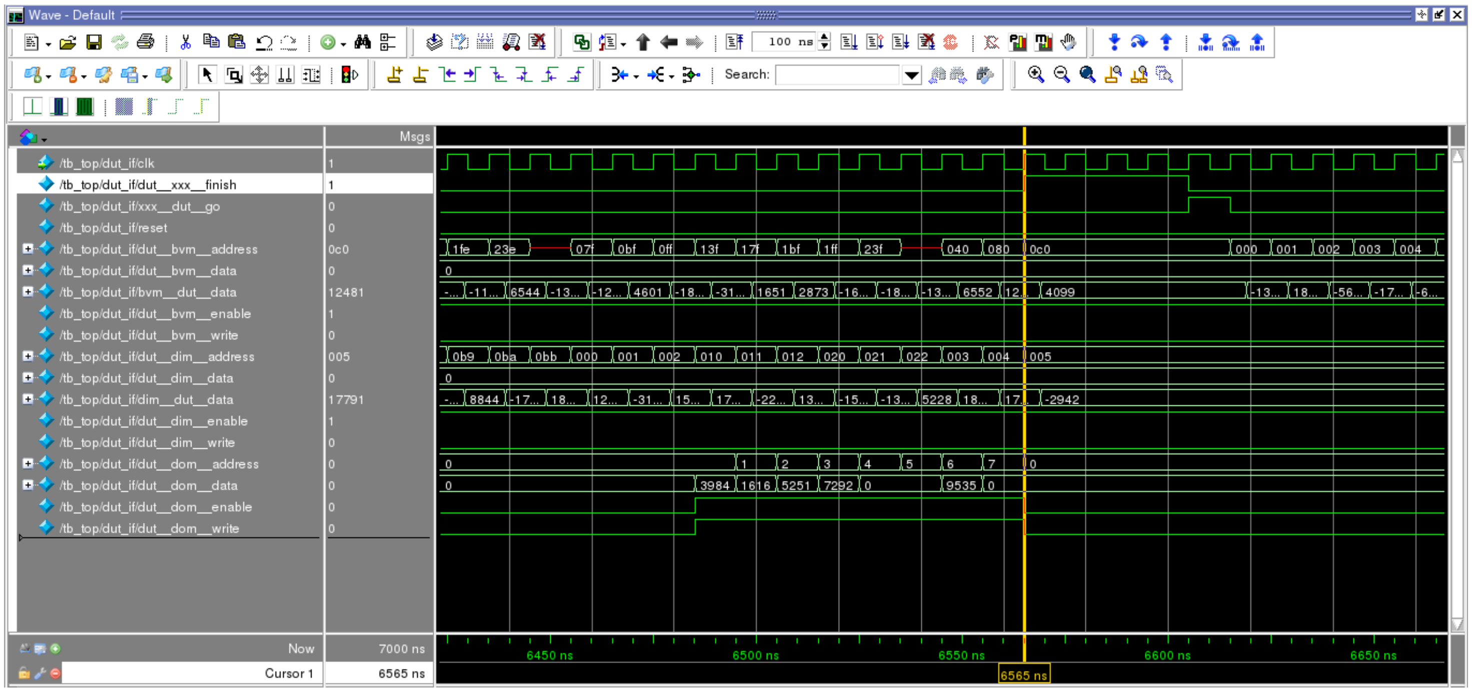
  - Figure 7. below shows the simulation when the computation is started after 'go' signal is received from the test bench:



	Msgs
/tb_top/dut_if/clk	0
/tb_top/dut_if/dut__xxx__finish	0
/tb_top/dut_if/xxx__dut__go	0
/tb_top/dut_if/reset	0
/tb_top/dut_if/dut__bvm__address	001
/tb_top/dut_if/dut__bvm__data	0
/tb_top/dut_if/bvm__dut__data	18765
/tb_top/dut_if/dut__bvm__enable	1
/tb_top/dut_if/dut__bvm__write	0
/tb_top/dut_if/dut__dim__address	002
/tb_top/dut_if/dut__dim__data	0
/tb_top/dut_if/dim__dut__data	15584
/tb_top/dut_if/dut__dim__enable	1
/tb_top/dut_if/dut__dim__write	0
/tb_top/dut_if/dut__dom__address	0
/tb_top/dut_if/dut__dom__data	0
/tb_top/dut_if/dut__dom__enable	0
/tb_top/dut_if/dut__dom__write	0
Now	7000 ns
Cursor 1	0 ns

Figure 7. Simulation snapshot showing the start of computation

➢ Figure 8. below shows the completion of computation and 'finish' signal being asserted after the 8 output elements are written to the output SRAM (dut_dom_data):



	Msgs
/tb_top/dut_if/clk	1
/tb_top/dut_if/dut__xxx__finish	1
/tb_top/dut_if/xxx__dut__go	0
/tb_top/dut_if/reset	0
/tb_top/dut_if/dut__bvm__address	0c0
/tb_top/dut_if/dut__bvm__data	0
/tb_top/dut_if/bvm__dut__data	12481
/tb_top/dut_if/dut__bvm__enable	1
/tb_top/dut_if/dut__bvm__write	0
/tb_top/dut_if/dut__dim__address	005
/tb_top/dut_if/dut__dim__data	0
/tb_top/dut_if/dim__dut__data	17791
/tb_top/dut_if/dut__dim__enable	1
/tb_top/dut_if/dut__dim__write	0
/tb_top/dut_if/dut__dom__address	0
/tb_top/dut_if/dut__dom__data	0
/tb_top/dut_if/dut__dom__enable	0
/tb_top/dut_if/dut__dom__write	0
Now	7000 ns
Cursor 1	6565 ns

Figure 8. Simulation snapshot showing the completion of computation

7. Conclusions

A hardware accelerator for two stage Convolutional Neural Network was designed. The design pipelined the two stages to achieve minimum computation time while having a check on the area of the chip. Below is the summary of the results obtained:
Clock time = 21ns
Execution time = 622 cycles
Area obtained = 16892.0642 um2