Name: Ramachandran Sekanipuram Srikanthan
Unityid: rsekani
StudentID: 200367158

Delay (ns to run provided provided example).
Clock period: 2.5 ns
# cycles”: 59

Area: (um^2)
2067.35

Memory: N/A

1/(delay.area) (ns^{-1}.um^{-2})
3.2793 * 10^{-6}

Delay (TA provided example. TA to complete)

1/(delay.area) (TA)

Abstract

The project focusses developing a module for performing binary convolution of the input image. A weight of size 3x3 is being used to perform the binary convolution. The input matrix can be of 16x16, 12x12 or 10x10 in size. The binary convolution algorithm differs from the normal convolution in the usage of XNOR operation rather than convolution multiplication as in the case of basic convolution procedure. This drastically reduces the hardware involved in performing the operation. The input matrix and the weight data are being stored in SRAM. The module takes data and weights from these SRAM, performs computation and stores in output SRAM. Thus, the module can be used as a standalone chip which can be plugged into the system directly and connected to corresponding memory address to perform the binary convolution. The RTL level logic description is being synthesized using the Synopsys design compiler using Synopsys PDK which uses 45nm nodes. The module can work up to a clock period of 2.5ns while having positive slack and occupies an area of 2067.35um^2.

Binary Artificial Neural Network

Ramachandran Sekanipuram Srikanthan

Abstract

The project focusses developing a module for performing binary convolution of the input image. A weight of size 3x3 is being used to perform the binary convolution. The input matrix can be of 16x16, 12x12 or 10x10 in size. The binary convolution algorithm differs from the normal convolution in the usage of XNOR operation rather than convolution multiplication as in the case of basic convolution procedure. This drastically reduces the hardware involved in performing the operation. The input matrix and the weight data are being stored in SRAM. The module takes data and weights from these SRAM, performs computation and stores in output SRAM. Thus, the module can be used as a standalone chip which can be plugged into the system directly and connected to corresponding memory address to perform the binary convolution. The RTL level logic description is being synthesized using the Synopsys design compiler using Synopsys PDK which uses 45nm nodes. The module can work up to a clock period of 2.5ns while having positive slack and occupies an area of 2066.55um^2.

Introduction

The project focused on designing a standalone hardware for performing binary convolution operation which can be used directly used by providing input matrices and weight matrices. A finite state machine controller is built into the module to read the matrices, perform binary convolution operation and store the output.

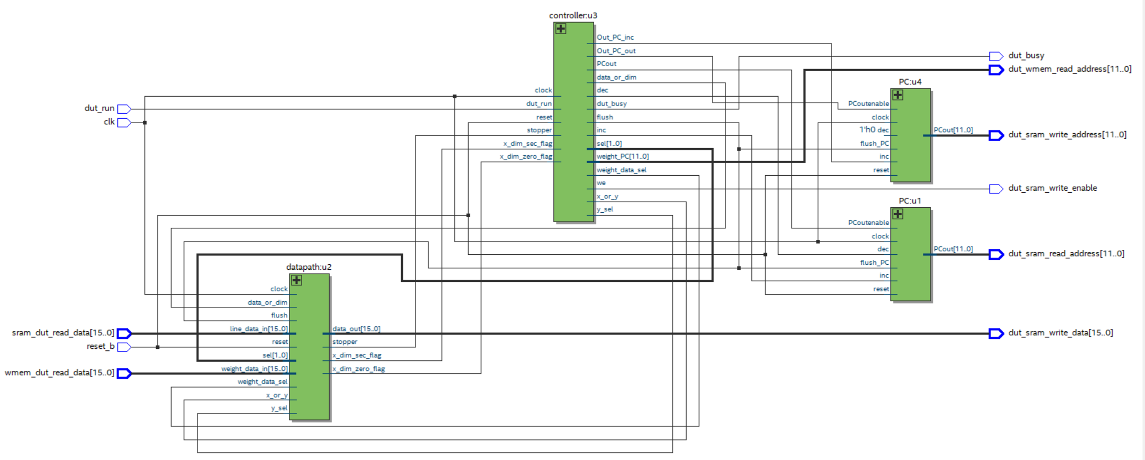
The Binary convolution algorithm, in compared to convolution reduces the overhead on complex multiplication and addition by reducing the convolution problem to bit wise computation. The input matrix contains only binary values of +1 and -1 (which is represented as 0 in hardware). This data is convoluted by using basic XNOR gates instead of multipliers. Even though the computation looks smaller, doing this in on chip core causes overhead in performing each operation. Thus, this project focused on developing a separate standalone accelerator to perform the operation.

The module gets input and weight matrix from the respective SRAM, then performs computation and stores in a separate output SRAM. The design is divided into two parts as data path which takes care of computation of binary convolution and a controller orchestrates each micro-operation to get the result. A finite state machine controller with 10 states is being used to perform the operation. The design also contains a separate Counter register for each input and output SRAM to store the current address which is being accesses. The module can perform binary convolution for input matrix sizes of 16x16, 12x12 and 10x10 with a weight matrix of size 3x3 thereby producing a result of 14x14,10x10 and 8x8 respectively. The module also supports internal zero padding while storing the output but needs external zero padding of inputs and weights.

The report describes the micro architecture, interface specifications, timing specifications and final metrics of the design.

Micro-Architecture

Overall Design

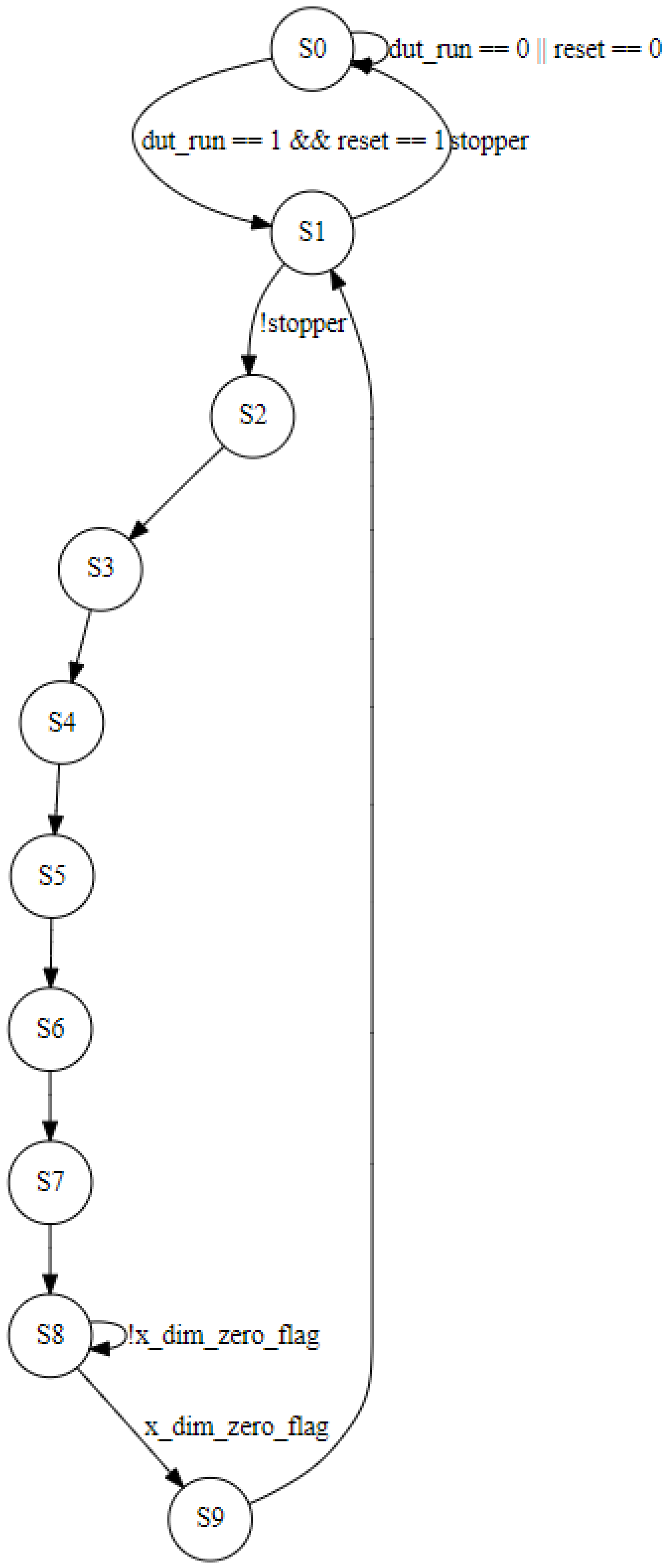


The design is divided into 4 modules which is being wired together in the top-level design entity. The modules are listed below.

- Controller
- PC_in
- Datapath
- PC_out

The control signals from the output of controller are being wired to the control signal inputs of the PC_in, PC_out and data path. The PC_in and PC_out modules are being used to keep track of the address from which the input SRAM and output SRAM must be accessed. The Datapath takes the control signals and performs the binary convolution and signals the controller whenever the current set of values from the input matrix is completed or the test bench ended.

Controller:


S0
dut_run == 0 || reset == 0
dut_run == 1 && reset == 1
stopper
S1
!stopper
S2
S3
S4
S5
S6
S7
S8
!x_dim_zero_flag
x_dim_zero_flag
S9

State	inc	PCout	dec	data_or_dim	x_or_y	sel	y_sel	weight_data_sel	we	Out_PC_inc	Out_PC_out	weight_PC	dut_busy
S0	0	0	0	0	0	0	0	0	0	0	0	0	0
S1	1	1	0	0	0	0	0	0	0	0	0	1	1
S2	1	1	0	0	0	0	0	1	0	0	0	1	1
S3	1	1	0	0	1	2	1	1	0	0	0	1	1
S4	1	1	0	1	1	1	0	0	0	0	0	1	1
S5	1	1	0	1	1	1	0	0	0	0	0	1	1
S6	1	1	0	1	1	1	0	0	0	0	0	1	1
S7	1	1	0	1	1	1	0	0	0	0	0	1	1
S8	depends	1	depends	1	1	1	0	0	1	1	1	1	1
S9	0	1	1	1	1	0	0	0	0	0	0	1	1

Whenever there is an active low signal in the reset input, the controller goes into the S0 and stays there until reset becomes high. The reset signal is wired to all the sequential components of the design which brings the flip flops from the unknown state to 0. Now when the go signals which is given as input to module becomes high and reset signal is high, it goes to S1 state and starts computation from there. Here the busy signal is set. The corresponding control signals are listed in the above table.

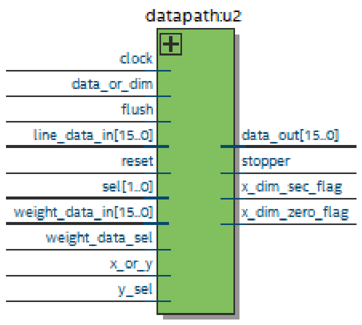
At State S1, the address from the PC is passed into the input data and weight SRAM. Since there is a delay of one cycle between the rising edge where we send the input to the rising edge where the value comes out of SRAM, there is a transition to temporary state S2. Here we just set the increment flag of input data PC, to facilitate the arrival of data in each cycle onwards.

Now the input data arrives into the data path at the state S3. Since the first two data are x dimension and y dimension, it is being directed to the corresponding registers. Similarly, the weight data is being directed into the weight register.

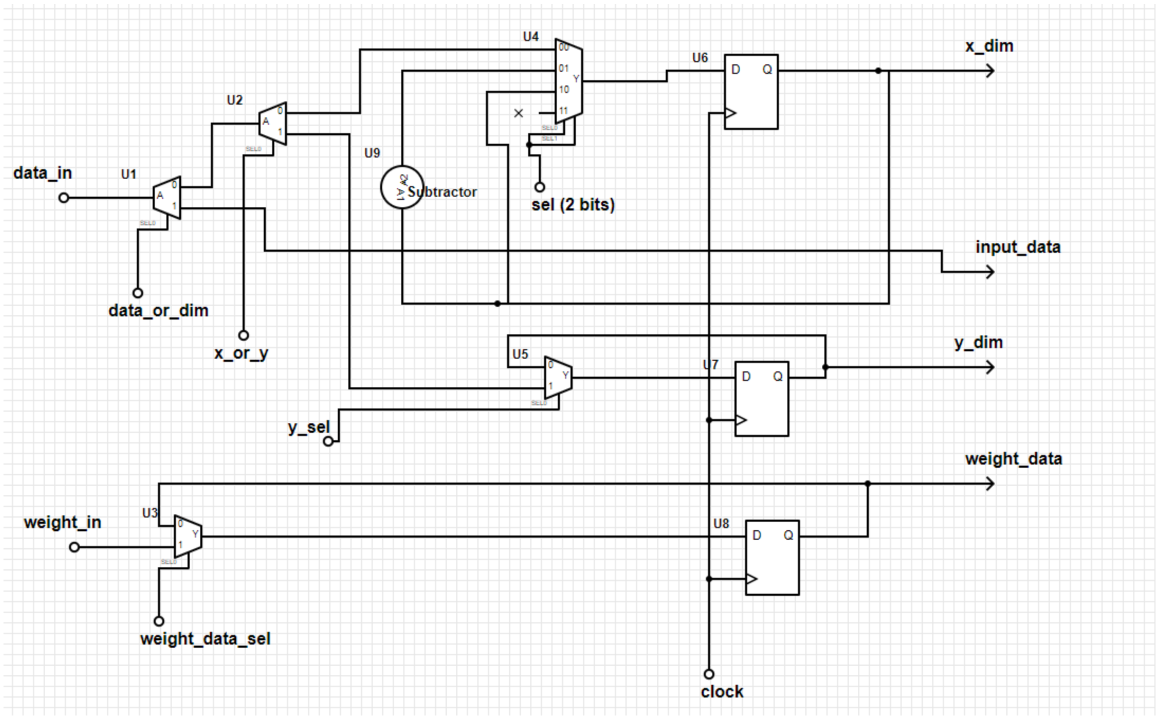
From the State S4, input matrix data arrives into the data path. The first XNOR block computes the binary convolution with the weight W2 (W [8:6]). The next two XNOR array computes the binary convolution with the weight W1 (W [5:3]) and W0 (W [2:0]). At the state S7 the binary convolution of the first 3 rows of the input ends and the final 16 bits value with zero padding is being stored in the data register. Now at State S8 write enable flag is being set and along with the write address and data is being passed to output SRAM. From State S8 onwards the same steps are being performed and in each rising edge of clock a new output is being generated until the x dimension which is being decremented on reading each line becomes 0. In that case, since we have two newer values from the next set of inputs into the pipe, we flush those and decrement the PC by 2 (at S8 and S9) and go back to state S1 and start the same process for the next set of inputs.

Finally, when the input data becomes “0000000011111111”, the PCs’ data and registers data are being cleared out and the state transitions to S0 and busy signal becomes 0 indicating the end of computation. Now the results are stored in the corresponding locations in the output SRAM.

Datapath

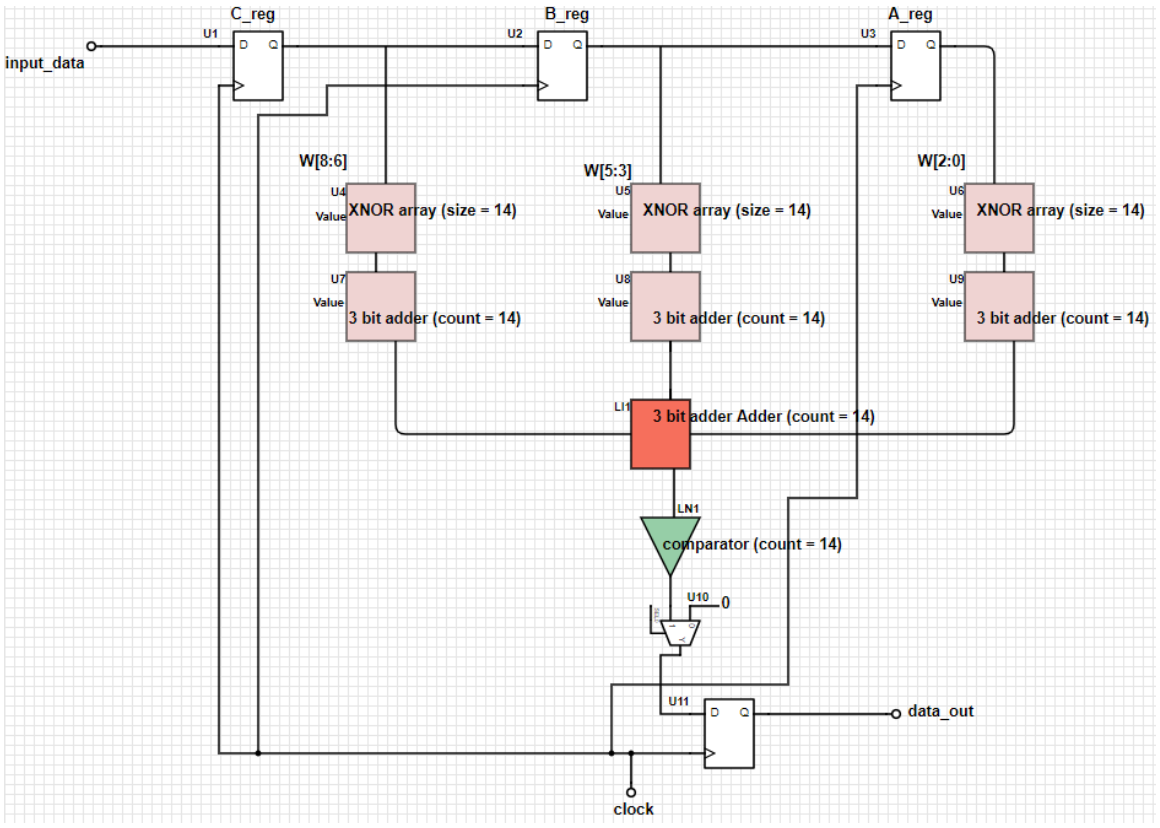


The Datapath contains two functional units one for fetching the data and directing it into appropriate registers and other for performing the binary computation. Additionally, the data path has separate external module for storing and sending the address for fetching the data (module PC).



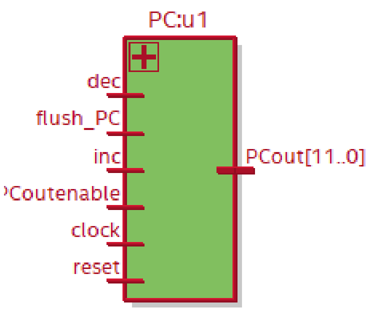
The above diagram denotes the circuit for the fetching unit. The data and weight matrices are obtained as input to the circuit from the corresponding SRAM. Initially for the first two lines in the input data line, we have x and y dimension which is being diverted to corresponding registers using a demux. Similarly, the weight data value is being directed to weight registers.

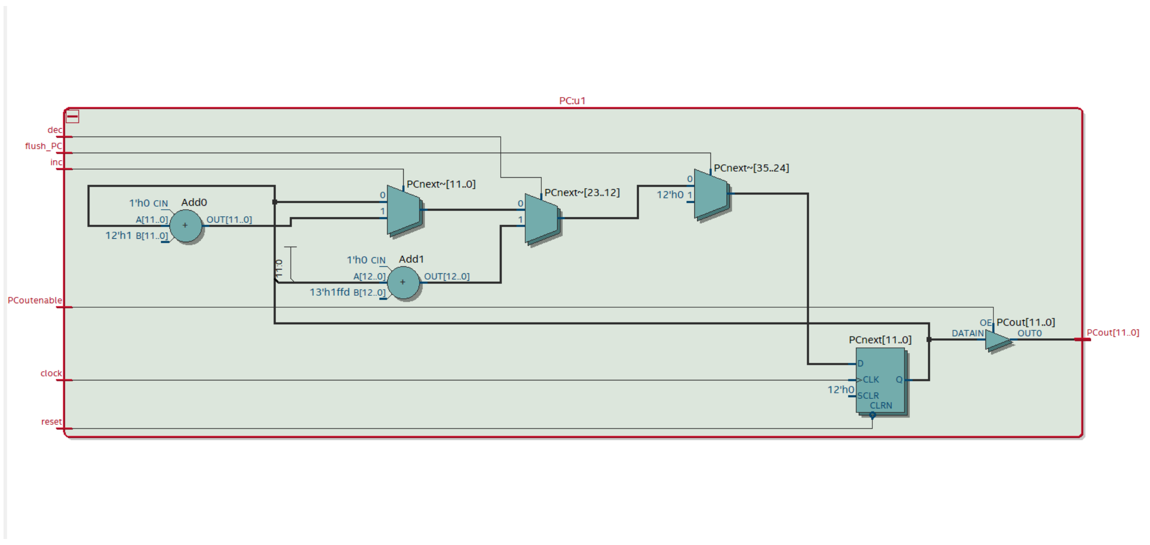
Now to detect the end of the input matrix for a particular x, y dimension, we use a decrementor which decrements the x dimension value by 1 whenever a new line of input matrix is being read. When the x dimension register becomes 0, a flag is being set which triggers the control to start with the next test bench.



The above is the logic diagram for performing the computation of binary convolution. The weight matrix data from the register is being split into 3 lines W0, W1 and W2 which contains W [2:0], W [5:3] and W [8:6] respectively. The input matrix data which is being directed from the fetch stage is taken as input. Initially line 0 of input matrix is convoluted with W2 value and number of ones in the 3-bit value is being computed. Since this value is not being used and the output of other two registers are stale, we ignore the result for the first three cycles. Now in the third cycle we may have line 0 convoluted with W0, line 1 with W1 and line 2 with W2 which forms the convolution value for the first 3 rows. From the number of ones from the individual registers, we perform a row wise sum for each column and then we compare if the value denoting the number of ones produced by the weight matrix acting on 3x3 element array from the input matrix is greater than 4. If so, we output a value of 1 or else it is 0. Now we use a multiplexer to test if the value needs to be included in the final output or it is a part of zero padded bits. Based on the y dimension register's value we choose the 0's for the necessary bits. Finally, we store the value in the data_out register. Now in the next cycle, we pass the write data, write address and write enable to output SRAM so that it writes the final convolution value.

Program Counter

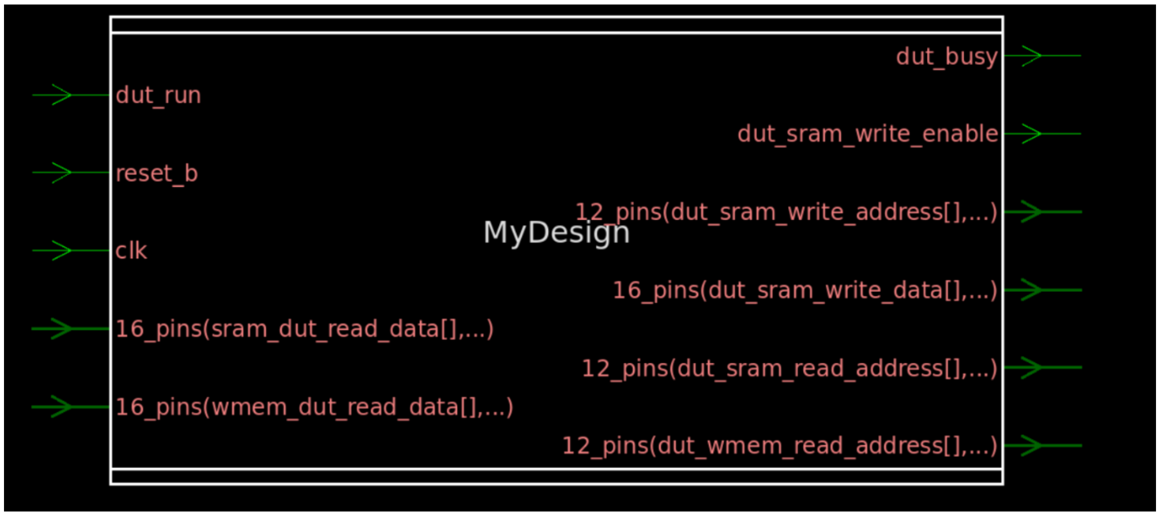




A separate count register is being used to store the address from which the input matrix needs to accessed and the output value must be written to. The PC's output contains a tri state buffer which prevents the unnecessary memory access by setting the address to high impedance value. The PC module has two signals to for performing increment and decrement. The value is being stored in the register and sent out to SRAM address line when the PCout signal is being set.

Interface Specification

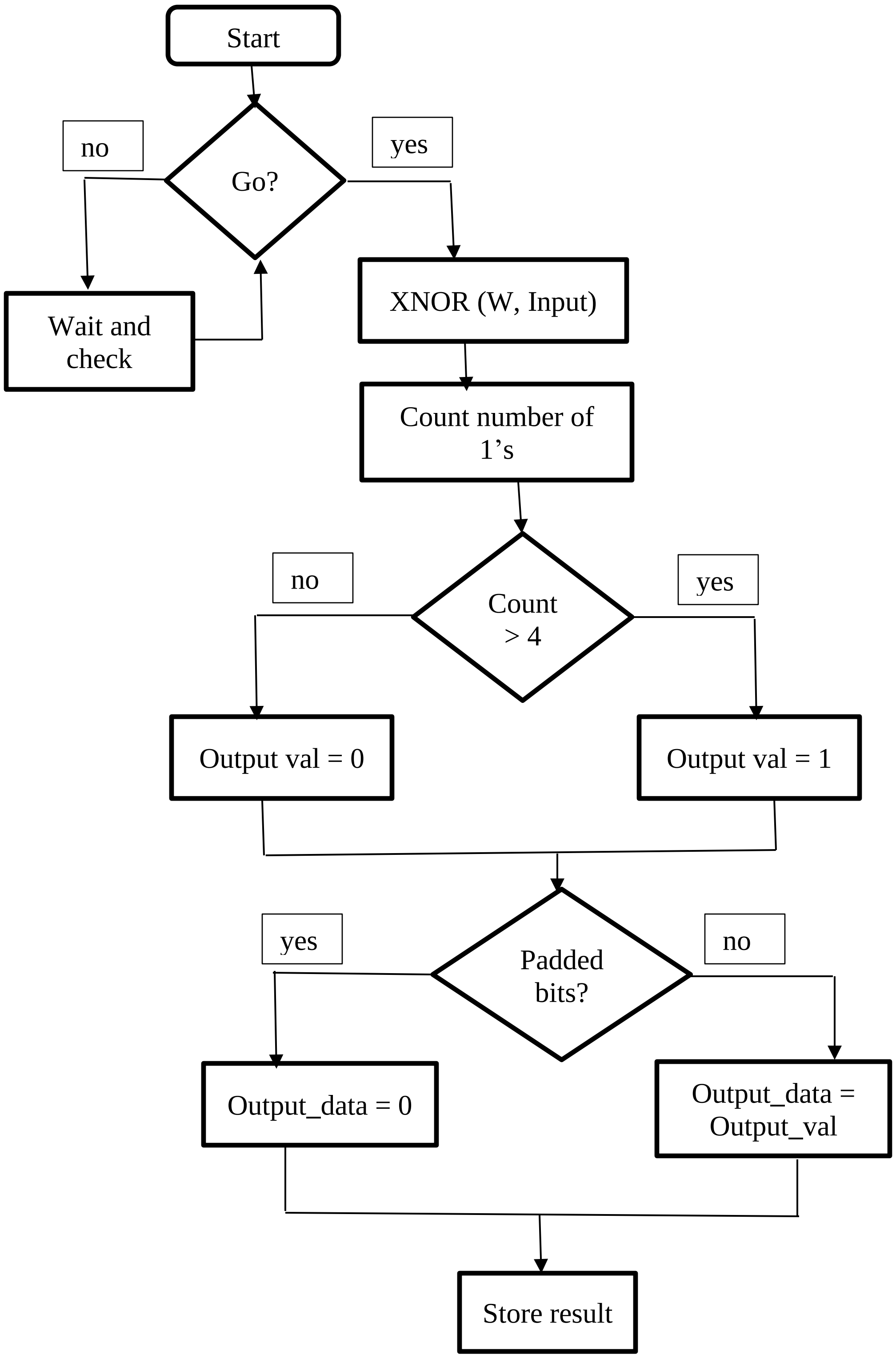
Port Name	Width	Function
dut_run	1	Go signal which when signalled computation starts
reset_b	1	Asynchronous reset
clk	1	clock
sram_dut_read_data	16	The input matrix and dimension value from input SRAM
wmem_dut_read_data	16	The weight data from weight SRAM
dut_busy	1	Busy signal, high when computation is being performed
dut_sram_write_enable	1	High for writing value in output SRAM
dut_sram_write_address	12	Address to write the result
dut_sram_write_data	16	Output of binary convolution
dut_sram_read_address	12	The address from which the input matrix and dimension are read
dut_wmem_read_address	12	The address from which the weight is being read

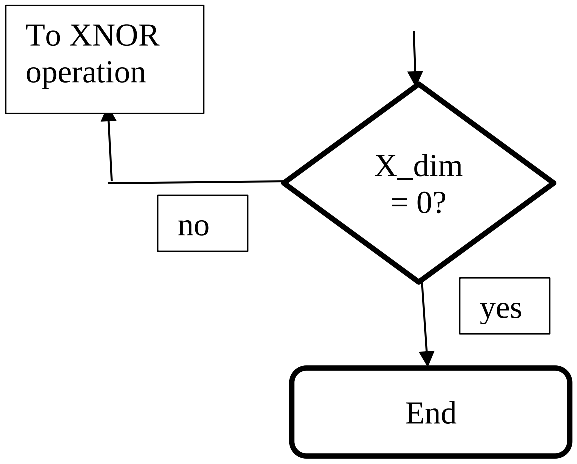

MyDesign
dut_run
reset_b
clk
16_pins(sram_dut_read_data[],...)
16_pins(wmem_dut_read_data[],...)
dut_busy
dut_sram_write_enable
12_pins(dut_sram_write_address[],...)
16_pins(dut_sram_write_data[],...)
12_pins(dut_sram_read_address[],...)
12_pins(dut_wmem_read_address[],...)

The standalone module contains 11 pins which can be connected directly to memory for performing binary convolution. The accelerator does not require a separate core to orchestrate its actions. The module contains its own FSM controller. The input signals to the module are clock, reset, dut_run, sram_dut_read_data and wmem_dut_read_data. The module signals the dut_busy signal while performing the convolution operation. The dut_sram_write_enable, dut_sram_write_address and dut_sram_write_data are used to store the convolution result into the output SRAM. The dut_sram_read_address and dut_sram_wmem_read_address is being connected to PC module to send the next address to read from input and weight SRAM.

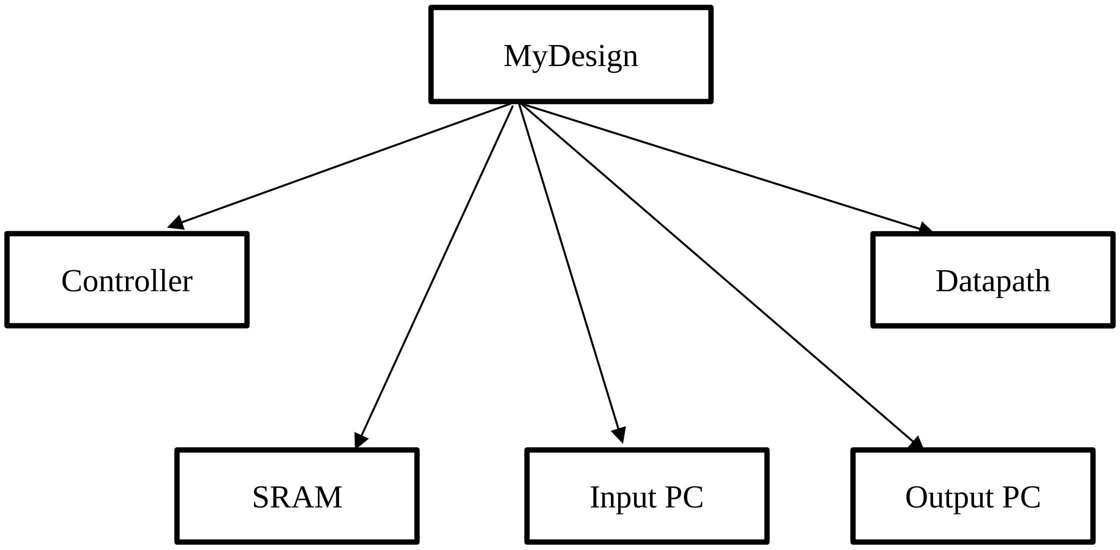
Technical Implementation

Algorithm





Hierarchy

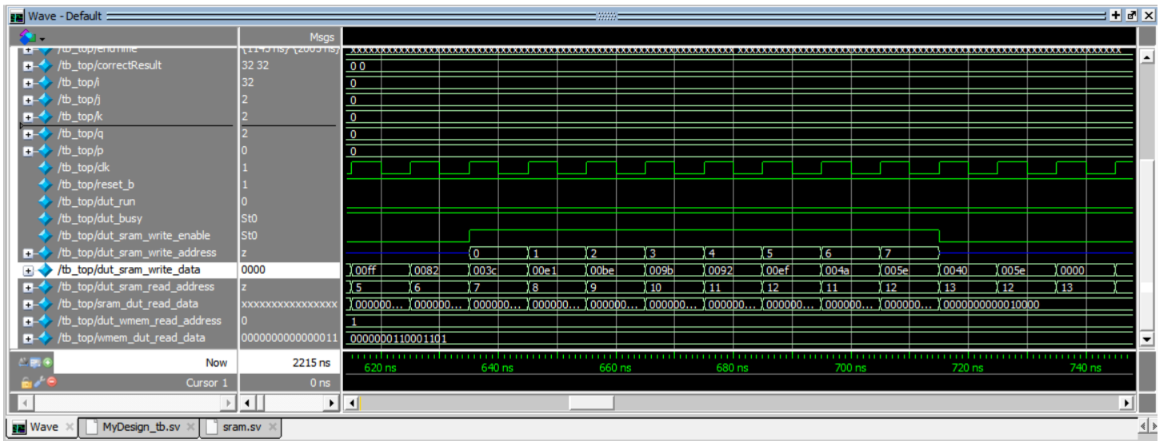


The MyDesign module acts as the top-level module which connects various signals from the modules at the hierarchy next to it. The MyDesign module can be directly used by the test bench to test or by the synthesizer to directly transform into a netlist.

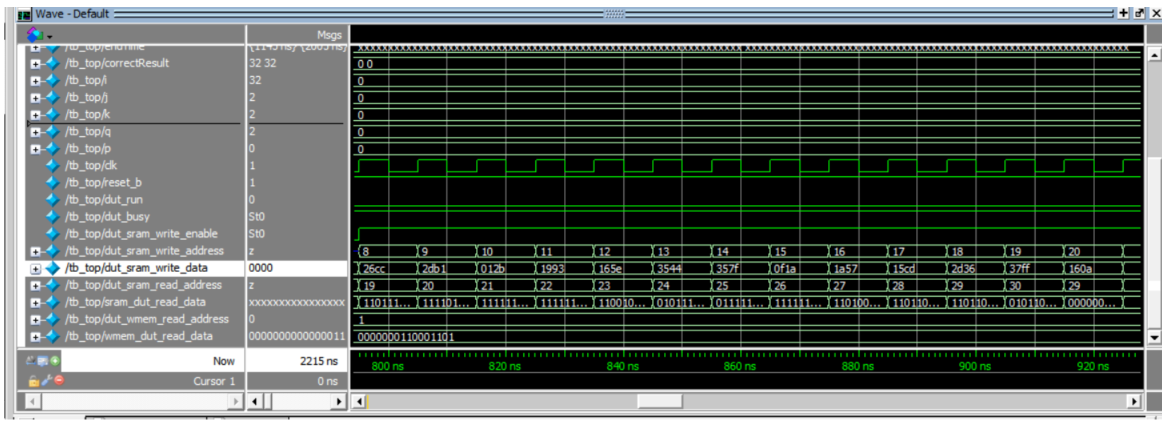
Verification

The verification is being performed by using the results obtained by the emulation of the same accelerator specifications using a high-level programming language. These results are being compared with the output of the module using a testbench written in system Verilog.

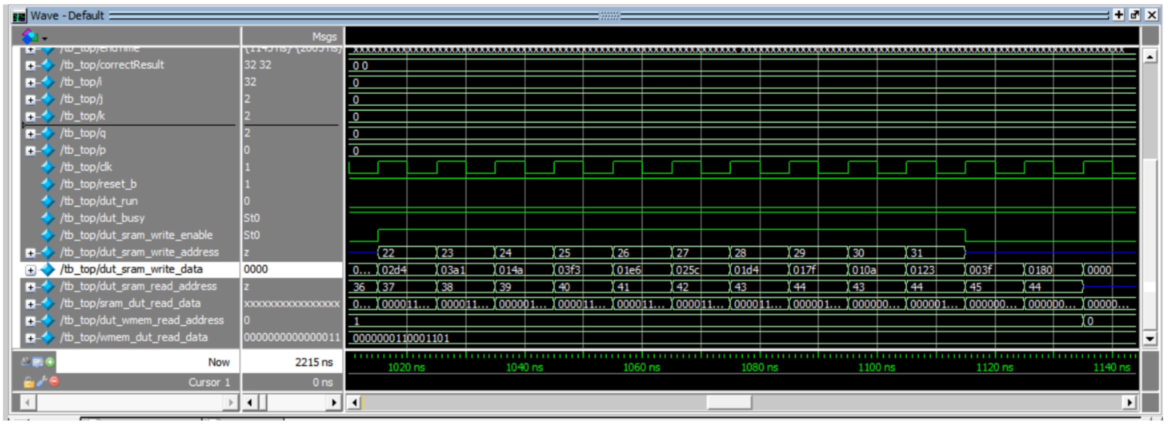
After synthesis using Synopsys design compiler, the netlist may contain some additional elements which may be tested for functional verification post synthesis. To facilitate the functional verification, the netlist and the corresponding digital logic library is being taken from the PDK and tested with the same test bench to match the test results from the emulator. The timing diagram for a test bench for 3 sets of computation is shown below.



```
Wave - Default
                                      Msgs
/tb_top/endTime                       {1145 ns} {2665 ns}
/tb_top/correctResult                 32 32
/tb_top/i                             32
/tb_top/j                             2
/tb_top/k                             2
/tb_top/q                             2
/tb_top/p                             0
/tb_top/clk                           1
/tb_top/reset_b                       1
/tb_top/dut_run                       0
/tb_top/dut_busy                      St0
/tb_top/dut_sram_write_enable         St0
/tb_top/dut_sram_write_address        z
/tb_top/dut_sram_write_data           0000
/tb_top/dut_sram_read_address         z
/tb_top/sram_dut_read_data            xxxxxxxxxxxxxxxx
/tb_top/dut_wmem_read_address         0
/tb_top/wmem_dut_read_data            0000000000000011
Now                                   2215 ns
Cursor 1                              0 ns
```



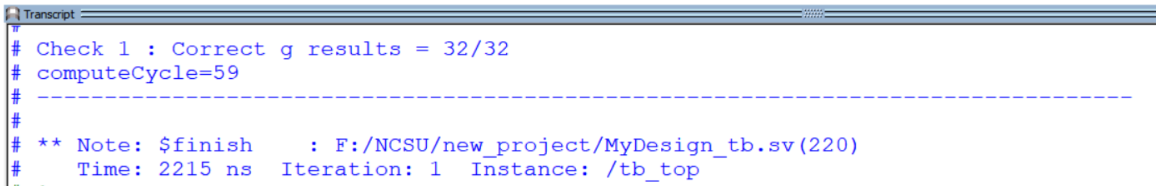
```
Wave - Default
                                      Msgs
/tb_top/endTime                       {1145 ns} {2665 ns}
/tb_top/correctResult                 32 32
/tb_top/i                             32
/tb_top/j                             2
/tb_top/k                             2
/tb_top/q                             2
/tb_top/p                             0
/tb_top/clk                           1
/tb_top/reset_b                       1
/tb_top/dut_run                       0
/tb_top/dut_busy                      St0
/tb_top/dut_sram_write_enable         St0
/tb_top/dut_sram_write_address        z
/tb_top/dut_sram_write_data           0000
/tb_top/dut_sram_read_address         z
/tb_top/sram_dut_read_data            xxxxxxxxxxxxxxxx
/tb_top/dut_wmem_read_address         0
/tb_top/wmem_dut_read_data            0000000000000011
Now                                   2215 ns
Cursor 1                              0 ns
```



```
Wave - Default
                                      Msgs
/tb_top/endTime                       {1145 ns} {2665 ns}
/tb_top/correctResult                 32 32
/tb_top/i                             32
/tb_top/j                             2
/tb_top/k                             2
/tb_top/q                             2
/tb_top/p                             0
/tb_top/clk                           1
/tb_top/reset_b                       1
/tb_top/dut_run                       0
/tb_top/dut_busy                      St0
/tb_top/dut_sram_write_enable         St0
/tb_top/dut_sram_write_address        z
/tb_top/dut_sram_write_data           0000
/tb_top/dut_sram_read_address         z
/tb_top/sram_dut_read_data            xxxxxxxxxxxxxxxx
/tb_top/dut_wmem_read_address         0
/tb_top/wmem_dut_read_data            0000000000000011
Now                                   2215 ns
Cursor 1                              0 ns
```

Results Achieved

Simulation output



```
Transcript
#
# Check 1 : Correct g results = 32/32
# computeCycle=59
# ------------------------------------------------------------------------------------
#
# ** Note: $finish    : F:/NCSU/new_project/MyDesign_tb.sv(220)
#    Time: 2215 ns  Iteration: 1  Instance: /tb_top
```

Setup time report

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

  Startpoint: clk_r_REG702_S3
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: clk_r_REG620_S3
            (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
  clk_r_REG702_S3/CK (DFFR_X1)           0.0000     0.0000 r
  clk_r_REG702_S3/Q (DFFR_X1)            0.7200     0.7200 r
  U3237/ZN (XNOR2_X1)                    0.3781     1.0981 r
  U3236/ZN (NAND2_X1)                    0.1179     1.2160 f
  U3206/ZN (INV_X1)                      0.0927     1.3088 r
  U4076/ZN (OAI21_X1)                    0.0872     1.3960 f
  U3306/ZN (AOI21_X2)                    0.3498     1.7458 r
  U4114/ZN (INV_X2)                      0.0566     1.8024 f
  U4113/ZN (NAND2_X2)                    0.1200     1.9223 r
  U4112/ZN (NAND2_X2)                    0.0663     1.9887 f
  clk_r_REG620_S3/D (DFFS_X2)            0.0000     1.9887 f
  data arrival time                                 1.9887

  clock clk (rise edge)                  2.5000     2.5000
  clock network delay (ideal)            0.0000     2.5000
  clock uncertainty                     -0.0500     2.4500
  clk_r_REG620_S3/CK (DFFS_X2)           0.0000     2.4500 r
  library setup time                    -0.4610     1.9890
  data required time                                1.9890
  -----------------------------------------------------------
  data required time                                1.9890
  data arrival time                                -1.9887
  -----------------------------------------------------------
  slack (MET)                                       0.0003
```

Hold time report

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

  Startpoint: clk_r_REG594_S1
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: clk_r_REG646_S3
            (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
  clk_r_REG594_S1/CK (DFFS_X2)           0.0000     0.0000 r
  clk_r_REG594_S1/Q (DFFS_X2)            0.0644     0.0644 r
  U4071/ZN (NOR2_X1)                     0.0124     0.0768 f
  clk_r_REG646_S3/D (DFFR_X1)            0.0000     0.0768 f
  data arrival time                                 0.0768

  clock clk (rise edge)                  0.0000     0.0000
  clock network delay (ideal)            0.0000     0.0000
  clock uncertainty                      0.0500     0.0500
  clk_r_REG646_S3/CK (DFFR_X1)           0.0000     0.0500 r
  library hold time                      0.0024     0.0524
  data required time                                0.0524
  -----------------------------------------------------------
  data required time                                0.0524
  data arrival time                                -0.0768
  -----------------------------------------------------------
  slack (MET)                                       0.0244
```

Area Report

```
Library(s) Used:

    NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm (File:
/afs/eos.ncsu.edu/lockers/research/ece/wdavis/tech/nangate/NangateOpenC
ellLibrary_PDKv1_2_v2008_10/liberty/520/NangateOpenCellLibrary_PDKv1_2_
v2008_10_slow_nldm.db)

Number of ports:                           89
Number of nets:                          1459
Number of cells:                         1340
Number of combinational cells:           1137
Number of sequential cells:               190
Number of macros/black boxes:               0
Number of buf/inv:                        261
Number of references:                          48

Combinational area:                  1117.731983
Buf/Inv area:                         144.704001
Noncombinational area:                949.619996
Macro/Black Box area:                   0.000000
Net Interconnect area:       undefined  (No wire load specified)

Total cell area:                     2067.351979
Total area:                  undefined
```

Power Report

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


Global Operating Voltage = 0.95
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1pW


  Cell Internal Power  = 388.3688 uW   (92%)
  Net Switching Power  =  34.1491 uW    (8%)
                         ---------
Total Dynamic Power    = 422.5179 uW  (100%)

Cell Leakage Power     =   9.5607 uW


                 Internal         Switching           Leakage
Total
Power Group      Power            Power               Power
Power   (   %    )  Attrs
------------------------------------------------------------------------
---------------------------
io_pad             0.0000            0.0000            0.0000
0.0000  (   0.00%)
memory             0.0000            0.0000            0.0000
0.0000  (   0.00%)
black_box          0.0000            0.0000            0.0000
0.0000  (   0.00%)
clock_network      0.0000            0.0000            0.0000
0.0000  (   0.00%)
register           0.3483        5.9468e-03        2.3917e+06
0.3567  (  82.55%)
```

```
sequential         0.0000            0.0000            0.0000
0.0000  (   0.00%)
combinational  4.0036e-02        2.8202e-02        7.1689e+06
7.5407e-02  (  17.45%)
------------------------------------------------------------------------
---------------------------
Total          0.3884 mW   3.4149e-02 mW    9.5607e+06 pW    0.4321 mW
```

Conclusions

Thus, the binary convolution accelerator is being implemented in Verilog in RTL level and tested. The module is also being synthesized using Synopsys design complier using 45nm technology PDK. The accelerator can function up to a clock period of 2.5ns and occupy an area of 2067.35 um^2 consuming a total power of 0.4321 mW. The accelerator can perform binary convolution operation for input size of 16x16, 12x12 and 10x10 with a weight size of 3x3. The module can work as a standalone circuit and can work when directly connected to the corresponding memory lines.