Final Project Report

StudentID. Name: Nandana Balachandran

Delay (ns to run provided example).
Clock period:
cycles": 48

Area:

1892.59(um^2)

Logic:

Memory: N/A

1/(delay.area) ($ns^{-1}.um^{-2}$)
$1.1x\ 10^{-5}$

Delay (TA provided example. TA to complete)

1/(delay.area) (TA)

Binary Convolutional Neural Network

Nandana Balachandran

Abstract

The purpose of the project is to implement a fixed size single stage binary neural network. Binary neural networks are used to save storage and computational complexity when it comes to training deep models or resource intensive trainings. It is found to have a speed up of 58 when compared to the normal neural networks. Accuracy can be controlled in this model and although the accuracy is lesser than the normal convolutional neural network, studies are being conducted to improve it.

The input value is stored in binary format in an input SRAM and the weight for the neural network in a weight SRAM and the computed output should be stored in an output SRAM. The neural network is fully connected and the computation is done similar to how a normal convolutional neural network is done except that XNOR is used instead of multiplying. The weight matrix is given a fixed size of 3 x 3. The normal convolutional neural network would have weight and input decimal values and the first element of the weight matrix is multiplied and the second with the second and so on and finally an addition is done. However, here, the values are calculated by doing XNOR of the input matrix and the weight matrix and the final value is stored in the output SRAM. The implementation is done by using multiple finite state machines as the control and data fetching which fetches the input and weight data and computation block which does the XOR calculation and the writing to output SRAM which writes the evaluated data into the SRAM.

Introduction

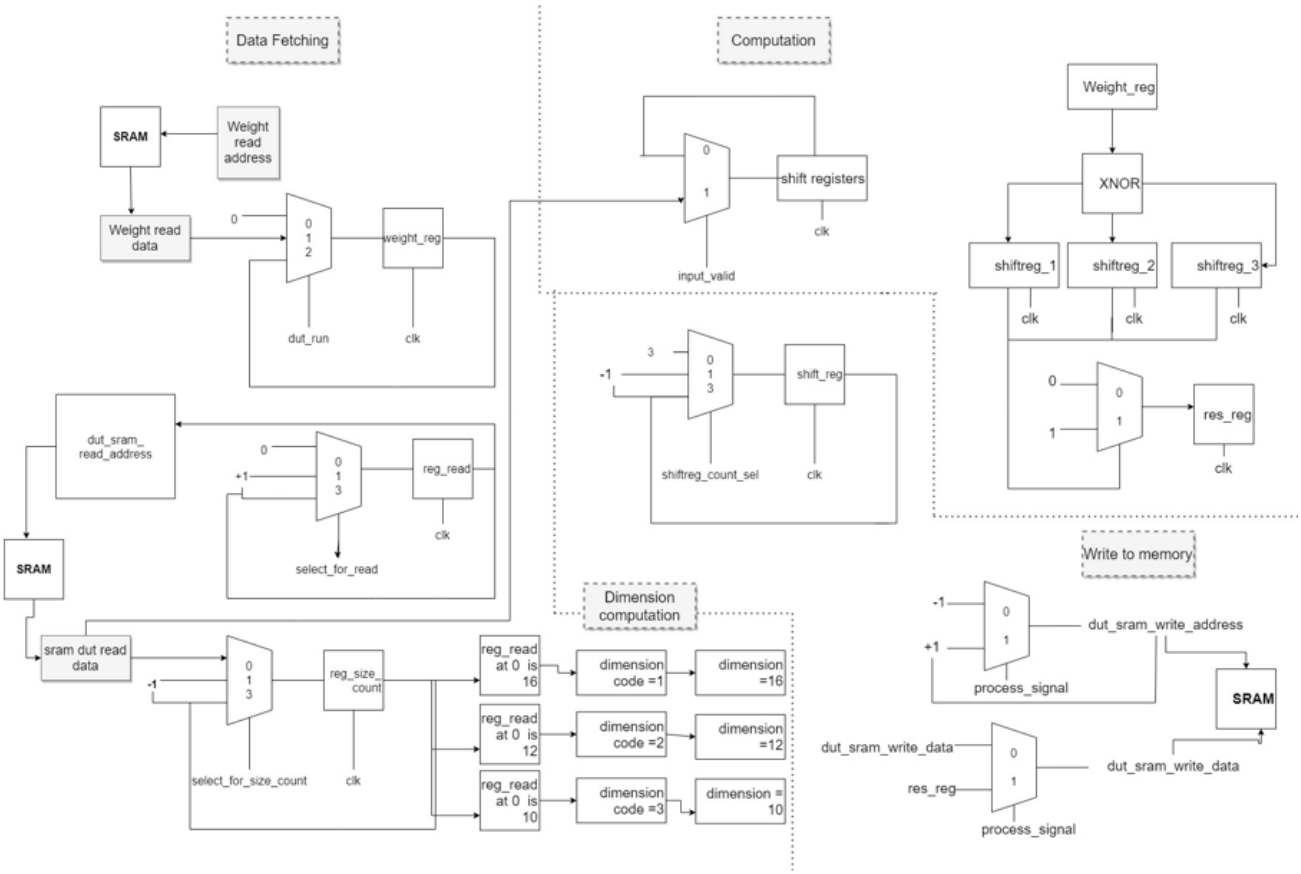
The hardware being designed here can be explained is a binary neural network that uses XNOR logic for the computation of the result values. The input, weight and output are stored in their respective SRAMs. The golden outputs

that were provided is used to verify the obtained result. Optimization of the design is done and the used clock, throughput and area is calculated.

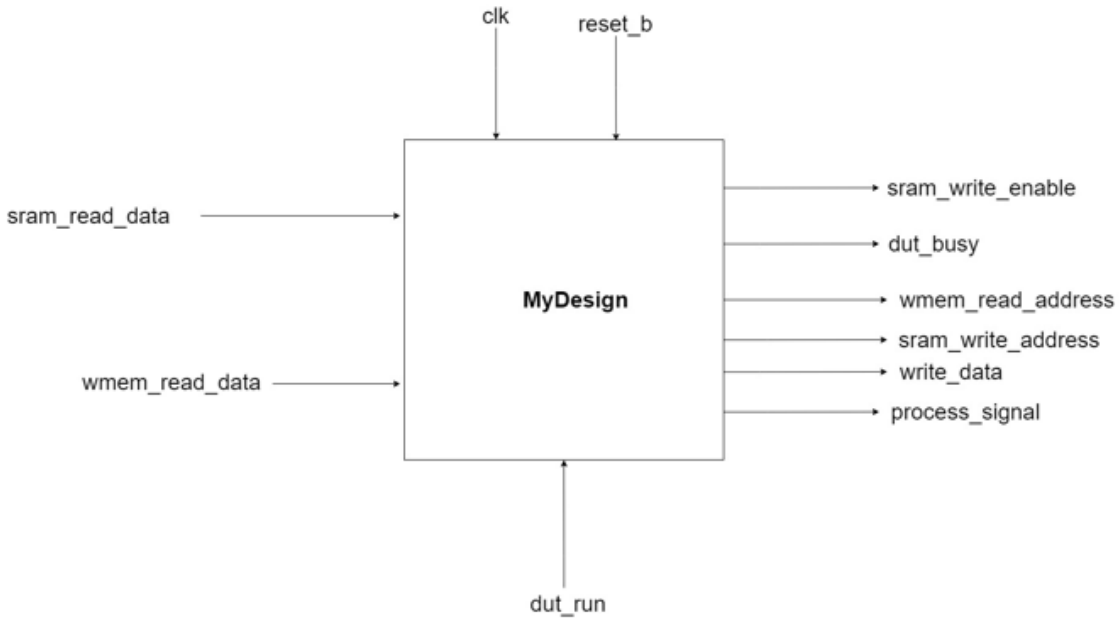
The key idea used here is using three 16-bit shift registers which stores the values and computation is done on all the data from the three 16 bit registers in parallel with the weight register and this improves the efficiency and reduces the clock cycles that are taken for the computation to be done. Writing to the output SRAM memory is also done simultaneously and this saves the computation cycles. The computation cycle that was obtained is 48.

Micro-Architecture

The data fetching part has the data being imported from the input SRAM using a select line that is controlled by an FSM. The data is getting read into the register which counts the number of data that is read into the shift registers. Dimension value is calculated from a dimension code value that is stored based on the value that is read from the SRAM.
Weight data is fetched using the run select line. The weight address does not require a select line as the address is constant. The shift reg count select line makes sure the three registers gets the first three values taking three clock cycles initially. Later, 16 registers are input into the shift registers together and the computation is done. Computation is done in parallel as the weight registers and the input data is computed using the XNOR operation and the output value is checked for 0 and the if the count is greater than 4, then it is stored in the output result register as 0 and if not it is stored as 1. The writing to memory is done using the process signal. Process signal is used to indicate when the computation starts. So, as the computation starts, the write address increments and the data starts getting written to the ouput SRAM simultaneously.



Interface Specifications



Detailed description of interface to your design, to data sheet standards.

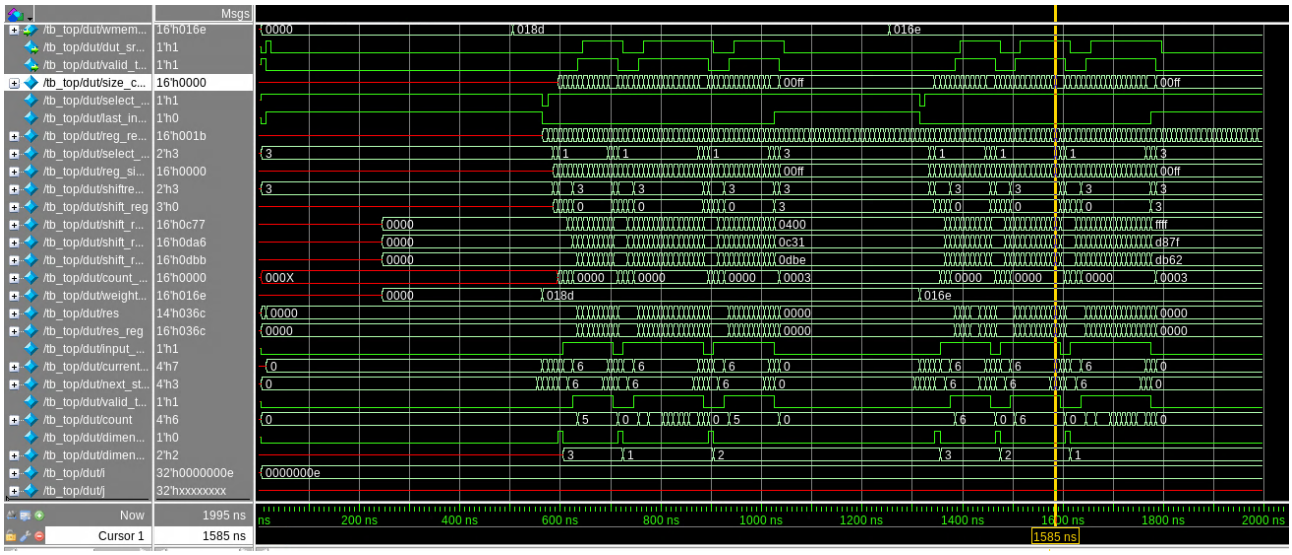
	Signal	Width	Function
1	dut_run	1 bit	Start signal
2	reset_b	1 bit	Reset signal
3	dut_busy	1 bit	Triggered when the computation starts
4	wmem_dut_read_data	16 bits	Read weight data
5	dut_sram_write_address	12 bits	Write address to write into SRAM
6	sram_dut_read_data	16 bits	Read data from the input SRAM
7	dut_wmem_read_address	12 bits	Read address to read from SRAM
8	dut_sram_write_data	16 bits	Write data to output SRAM
9	dut_sram_read_address	12 bits	Read address from input SRAM
10	dut_sram_write_enable	1 bit	Enable signal to write data to SRAM

Internal Signals

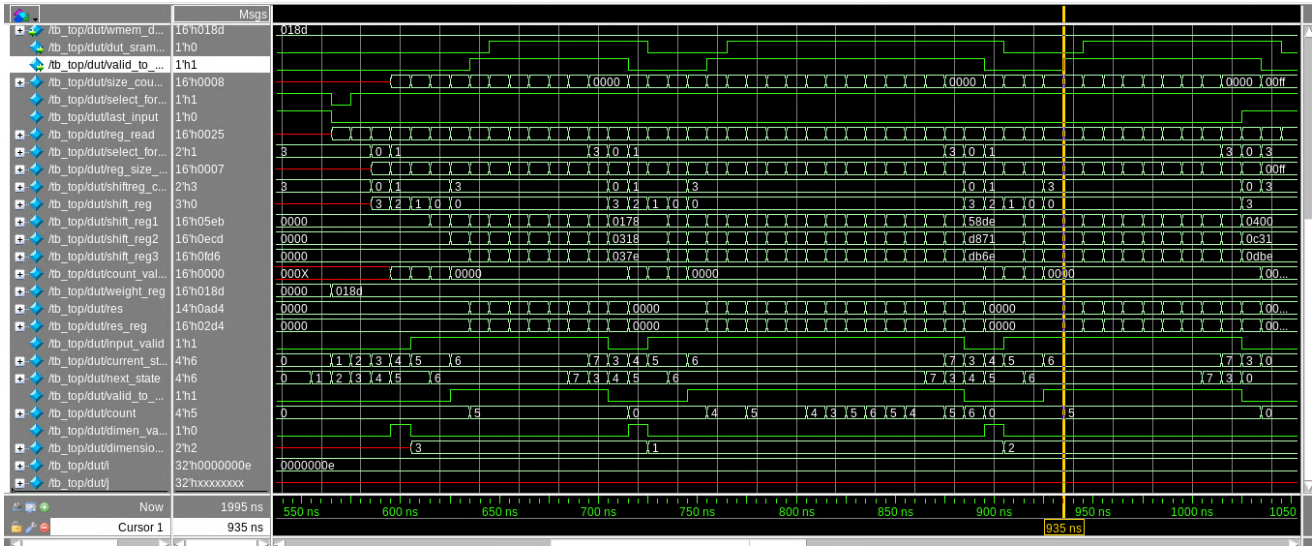
	Signal	Width	Function
1	select_for_read	1 bit	Select line for read address
2	shiftreg_count_sel	2 bits	Select line for shift registers
3	select_for_size_count	2 bits	Select line for reading dimension value
4	process_signal_sel	1 bit	Signal for computation
5	last_input	1 bit	Signal to signal the finish of one input tracefile
6	Input_valid	1 bit	Select line for input read data to process

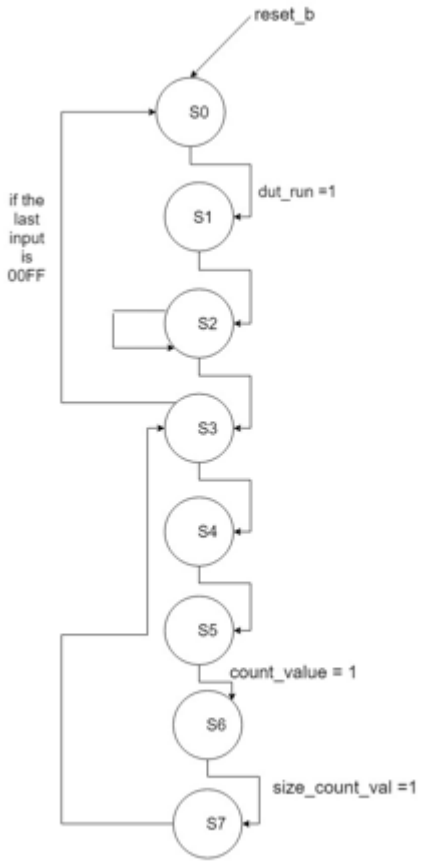
Interface timing diagram

Timing diagram for both the input files



Timing diagram for one input file



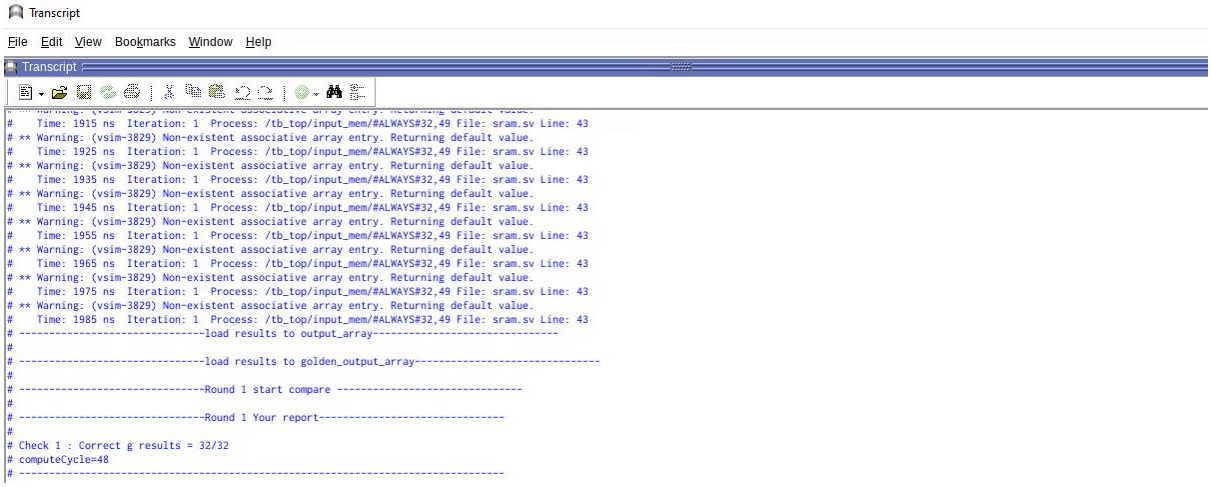


S0 is the reset state where all the registers are latched and when dut_run is 1, it goes to s1. S1 state reads the first address. S2 is another state where the values are latched. S3 reads the dimension of the input data and start reading the data and decrements the counter till all the data values are read. If the last data 00FF then go to the S0 state as it reaches the end of reading the data, if not carry on to the S4 state. S4 state carries on with adding the values to the shift register and decrement the counter. S5 state keeps adding values to the register till it gets the whole 16 bits per register. S6 manages the XNOR computation. S7 proceeds to read the address for the next set of input data and goes to the state s3.

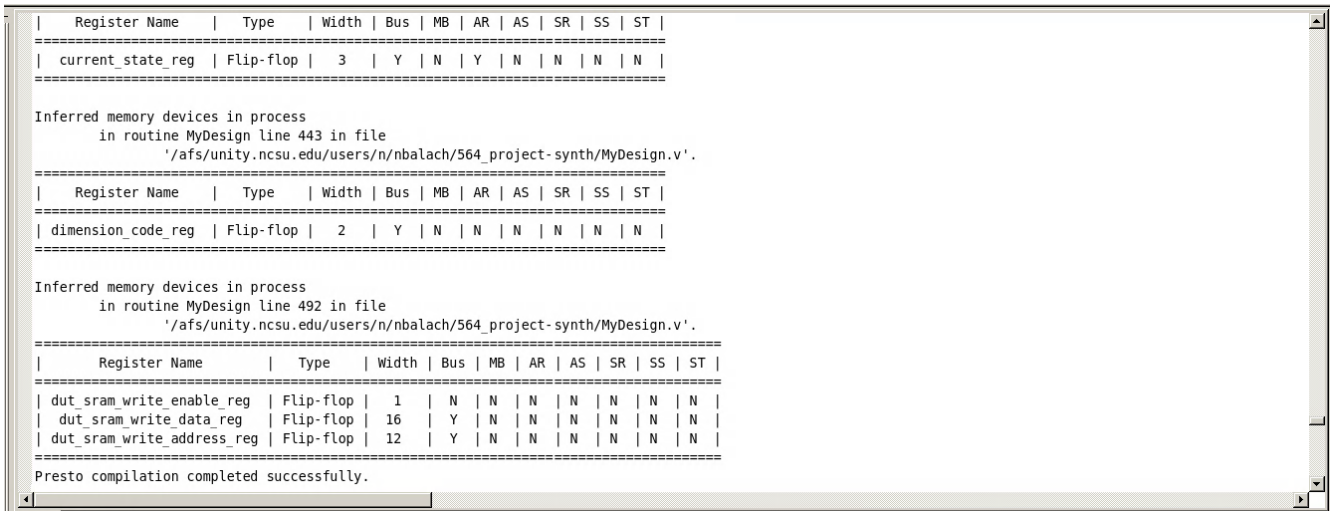
Verification

Verification is done against a test bench that does two round of computation for each input files – input_0 and input_0 files and verifies the output.

The transcript report 48 clock cycles for both the input files.



Transcript
File Edit View Bookmarks Window Help
Transcript
** Warning: (vsim-3829) Non-existent associative array entry. Returning default value.
Time: 1175 ns Iteration: 1 Process: /tb_top/input_mem/#ALWAYS#32,49 File: sram.sv Line: 43
** Warning: (vsim-3829) Non-existent associative array entry. Returning default value.
Time: 1185 ns Iteration: 1 Process: /tb_top/input_mem/#ALWAYS#32,49 File: sram.sv Line: 43
** Warning: (vsim-3829) Non-existent associative array entry. Returning default value.
Time: 1195 ns Iteration: 1 Process: /tb_top/input_mem/#ALWAYS#32,49 File: sram.sv Line: 43
** Warning: (vsim-3829) Non-existent associative array entry. Returning default value.
Time: 1205 ns Iteration: 1 Process: /tb_top/input_mem/#ALWAYS#32,49 File: sram.sv Line: 43
** Warning: (vsim-3829) Non-existent associative array entry. Returning default value.
Time: 1215 ns Iteration: 1 Process: /tb_top/input_mem/#ALWAYS#32,49 File: sram.sv Line: 43
** Warning: (vsim-3829) Non-existent associative array entry. Returning default value.
Time: 1225 ns Iteration: 1 Process: /tb_top/input_mem/#ALWAYS#32,49 File: sram.sv Line: 43
** Warning: (vsim-3829) Non-existent associative array entry. Returning default value.
Time: 1235 ns Iteration: 1 Process: /tb_top/input_mem/#ALWAYS#32,49 File: sram.sv Line: 43
------------------------------load results to output_array------------------------------
#
------------------------------load results to golden_output_array------------------------------
#
------------------------------Round 0 start compare ------------------------------
#
------------------------------Round 0 Your report------------------------------
#
Check 1 : Correct g results = 32/32
computeCycle=48
--
#



```
|    Register Name    |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
=======================================================================================
|  current_state_reg  | Flip-flop |   3   |  Y  | N  | Y  | N  | N  | N  | N  |
=======================================================================================

Inferred memory devices in process
        in routine MyDesign line 443 in file
                '/afs/unity.ncsu.edu/users/n/nbalach/564_project-synth/MyDesign.v'.
=======================================================================================
|    Register Name    |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
=======================================================================================
| dimension_code_reg  | Flip-flop |   2   |  Y  | N  | N  | N  | N  | N  | N  |
=======================================================================================

Inferred memory devices in process
        in routine MyDesign line 492 in file
                '/afs/unity.ncsu.edu/users/n/nbalach/564_project-synth/MyDesign.v'.
===========================================================================================
|       Register Name        |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
===========================================================================================
| dut_sram_write_enable_reg  | Flip-flop |   1   |  N  | N  | N  | N  | N  | N  | N  |
|  dut_sram_write_data_reg   | Flip-flop |  16   |  Y  | N  | N  | N  | N  | N  | N  |
| dut_sram_write_address_reg | Flip-flop |  12   |  Y  | N  | N  | N  | N  | N  | N  |
===========================================================================================
Presto compilation completed successfully.
```

Results Achieved

Throughput = 1.1×10^{-5} $ns^{-1}.um^{-2}$

Power = 0.2661 mW

Area = 1892.59 um^2

Clock period achieved = 2.8 ns

Conclusions

The single stage binary neural network has been implemented and the result has been verified using the test bench provided and all the values are found to be coinciding with 32/32 correct results with both the golden outputs provided respectively. The synthesis has been done and the optimisation has been done to get the most optimum values for clock period, power and area.