2 Stage Convolutional Network

Name: Darsh Asher
Unityid: dkasher
StudentID:. 200476123

Delay 0 (ns to run provided provided example). 59677.10ns
Delay 1 (ns to run provided provided example). 22593,9ns
Clock period: 4.9ns
cycles 0": 12179
cycles 1": 4611

Logic Area: (um^2)
2463.69

Memory: N/A

1/(delay0.area) ($ns^{-1}.um^{-2}$)
6.80152e-9
1/(delay1.area) ($ns^{-1}.um^{-2}$)
17.96e-9

Delay (TA provided example. TA to complete)

1/(delay.area) (TA)

Abstract:

This piece of hardware is a 2 stage convolutional network with RELU Activation function that can accept multiple different sized input Matrix up till the size of 64x64. The Input, Kernel and Outputs are stored in 3 separate SRAMs. First stage of this Convolutional network is multiplying the part of input matrix with kernel matrix. This part is a 3x3 matrix whose each element will be multiplied with the corresponding element of the kernel matrix. Convolution is carried out by adding all the corresponding values obtained after multiplication. This will be done for N-2 times horizontally and N-2 times vertically until the end is reached. A RELU function is implemented after Convolution stage to limit the values between 0 to 127. Then 2nd stage is Max Pooling. In this stage the Output of Convolution Stage is taken. 2x2 matrices are formed and the maximum value out of these 4 values are taken and stored in the output SRAM.

2 Stage Convolutional Network

Darsh Asher

1. Introduction

1.1 Summary

The hardware designed in this module implements a kernel convolution and Max Pooling. The kernel convolution is designed to be used with neural networks in which each element of kernel is multiplied with corresponding 3x3 subset of input matrix. This is done (N-2)x(N-2) times so that all elements of input matrices are covered. Then a Max-Pooling stage is implemented in which maximum out of the 2x2 subset of convolutional matrix is stored in the final output SRAM.

1.2 Challenges

There were many challenges I faced in this project. First challenge was to write the code from hardware perspective which I am not quite used to. I struggled with this project in the beginning because I was thinking from Object- Oriented perspective. Second challenge which I faced was storing the values from Input SRAM as I didn't understand in the beginning how memory works. Third major challenge which I faced was changing the addresses as the data to be read was not linear and different addresses were to be accessed for this project. I also faced challenge while changing subset of input matrix horizontally and vertically.

1.3 Result Summary

Using this design I was able to achieve a high throughput and maximize using memory bandwidth on the input memory. The final design used 2463.69 um2 of space and worked at a clock period of 4.9 ns.

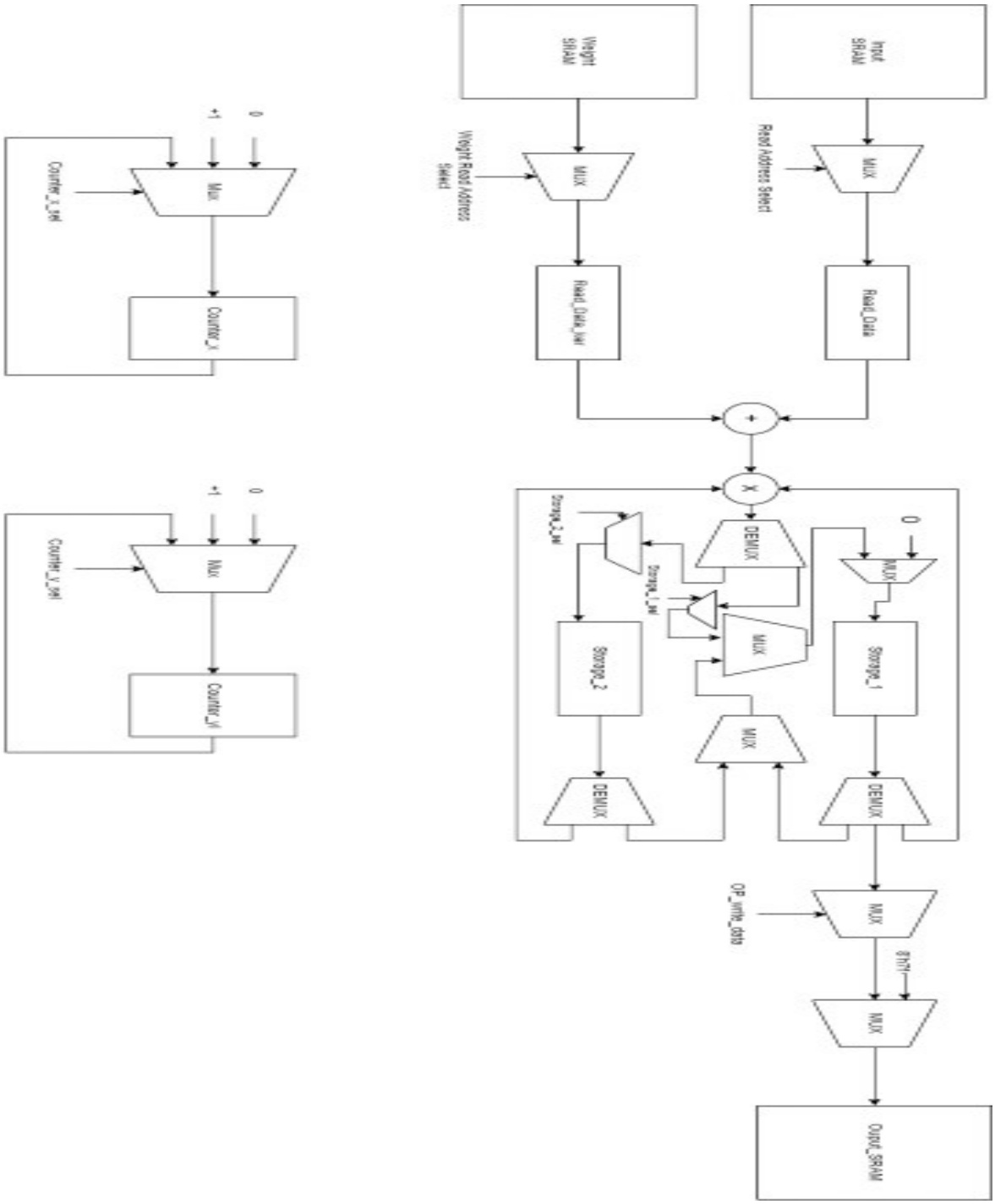
2. Micro-Architecture

This hardware stores single value taken from input and single value from kernel matrix. These values are then multiplied and sent back for addition. This process takes place 8 more times. After this it is stored in Storage_1 register. Then Read_data register stores value for 2nd matrix. This takes place 9 times and then it is stored in Storage_2 register. Then Contents of Storage_1 and Storage_2 are compared and the largest one is stored in Storage_1. Then similar process takes place for 3rd and 4th matrix as well. These values are stored in Storage_2 register. After each Convolution, Storage_2 is Storage_1 and highest value is stored in it. If any value is less than 0, then 0 is stored in its place. Finally when the value is stored in output SRAM, it is compared with 127 in order to keep values between 0 to 127. Whichever number is lower is stored in the Output SRAM. Similar process takes place (N/2-2)x(N/2-2) times and we directly obtain the output of MaxPooling.

Controlling FSM has 47 states in order to carry out both convolution and MaxPooling. FSM controls the address which are to be read from Input SRAM and address where the

data has to be written into the Output SRAM. 4 convolution multiplications and 3 comparisons take place in order to get one output element of MaxPooling matrix. This process goes on until all subsets of Input matrix are done.

2.1 Architecture Diagram:



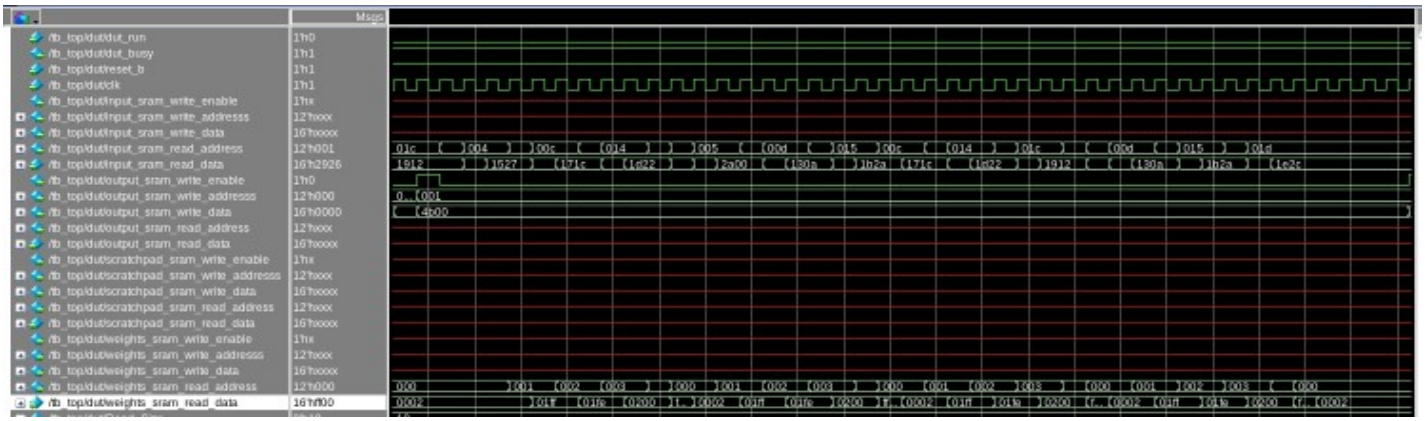
3.1 Interface

Interfacing with this hardware consists of 2 input and 1 output SRAMS that require addresses and data lines as well as a write enable line for the output SRAM. It requires a clock and for a reset to be asserted before operation begins. Afterwards the go input will trigger the module to start calculations on the matrix at address 0 while asserting a busy signal high. After it encounters a matrix terminated in 0x00FF it will resume waiting for the next assertion of go.

Signal (in module order)	Width (Bits)	Signal Description
dut_run	1	Signals module matrix is ready to run.
dut_busy	1	Signals device module is processing.
reset_b	1	Resets the module into wait state and initializes registers to 0.
clk	1	Clock input for module flip flops.
dut_sram_write_address	12	The address in the output SRAM that dut_sram_write_data is stored in.
dut_sram_write_data	16	Data send to output SRAM.
dut_sram_write_enable	1	Must be high for data to be written.
dut_sram_read_address	12	Input SRAM read address.
sram_dut_read_data	16	Input line for data from SRAM. Delayed from address by 1 clock cycle.
dut_wmem_read_address	12	Weight SRAM read address.
wmem_dut_read_data	16	Input line for weight from SRAM. Delayed from address by 1 clock cycle.

3.2 Timing Diagram:

The following image is a sample timing diagram for the interface. It shows 3 separate parts of a run with multiple matrixes. First occurs the loading where the module starts reading in data and starting the output. This occurs until the next matrix is encountered which is shown in the second part of the timing diagram and finally followed by the end of processing signaled by a matrix terminated in 0x00FF. F's in the names below are used to indicate the final data or final address that contains an input or output.



4. Technical Implementation

The whole module is contained within the top_without_mem.v

4.1 Hierarchy

1. MyDesign – Top Module
    - Controller – FSM controller
    - Size Count Register
    - Read Address
    - Read N
    - Read Data
    - Read Weight Address
    - Read Weight Data
    - Accumulator
    - Storage 1
    - Storage 2
    - Counter X
    - Counter Y
    - O/p write address
    - O/p write enable
    - O/p write data
    - Counter

This hierarchy allowed the bulk of Datapath to be within its own module while the submodules simplified the amount of HDL that had to be written

5. Verification

Verification was done through the provided test bench. The inputs and the outputs were placed in folders for each run along with the expected output vectors. The test bench

asserts the run signal and then compares the output to the expected output vectors for the given input.

6. **Results Achieved**

Metric	Results
Area	2463.69 um^2
Clock cycles 0	12179
Clock cycles 1	4611
Clock Period	4.9ns
No. of States	47 states
Delay 0	59677.1ns
Delay 1	22593.9ns
Performance 0	6.80152e-9 ($ns^{-1}um^{-2}$)
Performance 1	17.96e-9 ($ns^{-1}um^{-2}$)

7. **Conclusions**

The hardware is able to operate with a high throughput utilizing the maximum possible bandwidth of the input memory. It takes different time period depending on the size of matrix with a total cell area of 2463.69um^2