Project Plan Report /10

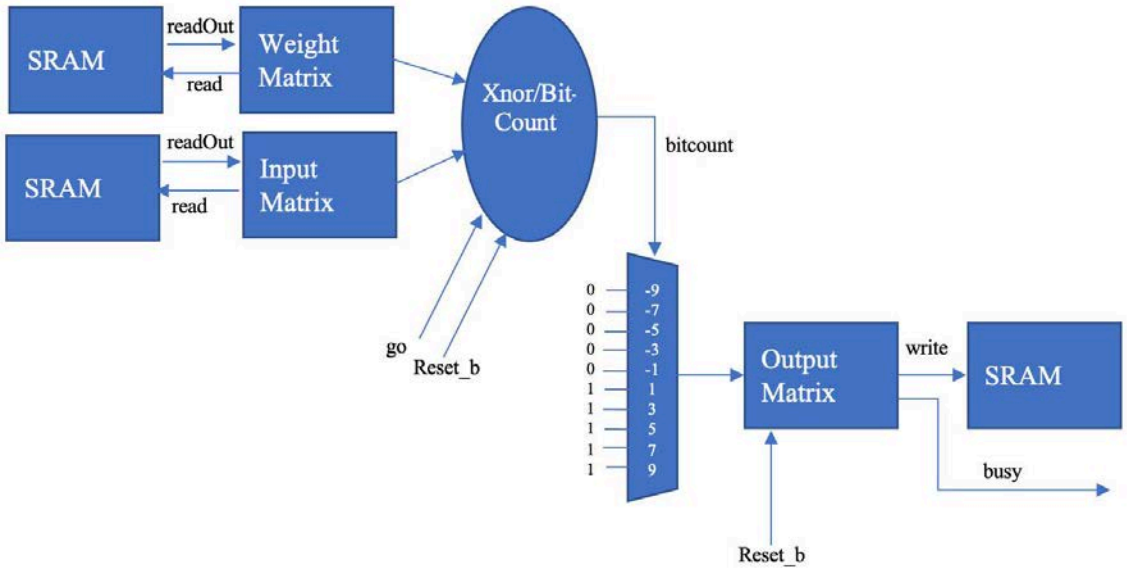
Name: Jon Mark Long **StudentID**: 200256447 **UnityID**: jmlong4	10/24 - datapath design 10/29 - controller design 11/1 - code datapath	11/4 - code controller 11/6 - synthesis 11/10 - timing and adjustments	**Schedule**

Summary Risk Plan: The biggest risk is my inexperience which makes project planning and estimation of timeline difficult. It's hard to know how difficult something will be when you haven't done it often. Another risk is hitting a roadblock that I am not technically proficient enough to solve with out help. I plan to work early so that I am not running into deadlines and have plenty of time to seek assistance if I need it

Verification Plan: After a hand design of the project, I will first create a hand drawn timing diagram. Then after I have designed and coded my project I will create a test bench that check and outputs errors when the output does not match the expected output I created in my hand drawn timing diagram. Lastly I'll double check that my design meets all the specifications of the project.

Brief Description of Mode of operation, including selected algorithms: First read in the weights, then the inputs, then do the first xnors, add 1 to bitcount when the xnor = 1 and minus 1 when xnor = 0, then check bitcout for plus or minus 9,7,5,3,1, if it's the plus version, that output bit gets 1, if minus, 0 (I think this will be less logic than a comparison statement). Once the output matrix is complete, write it back to memory making sure to pause reading. I also plan on trying to find patterns to reuse inside the matrix since the internal values are reused 9 times, but the outside border only once. Those patterns and pipelining to read or write to memory every clock cycle will enable an efficient and compact hardware design of this project.

High level sketch.



Weight and Input Matrices hold the info to send to the computation block which executes then outputs a count of the 1's relevant to the current Output Matrix cell. This count then decides whether to send a 0 or 1 to the Output Matrix. When the Matrix is complete it is written to the SRAM.

Final Project Report

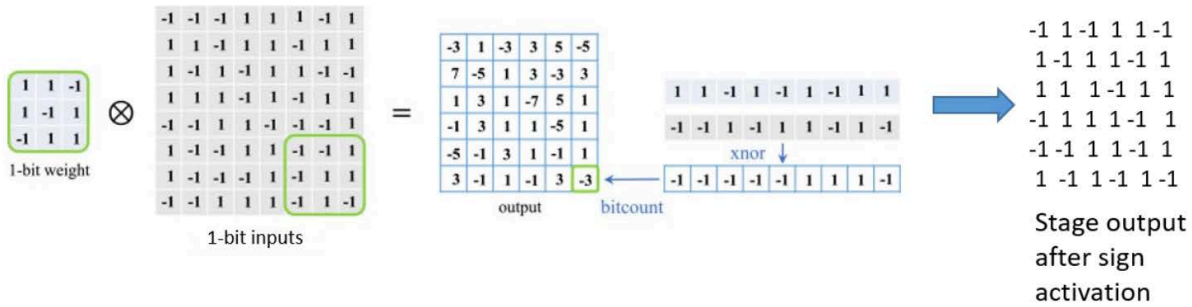
Name: Jon Mark Long
Unityid: jmlong4@ncsu.edu
StudentID: 200256447

Delay (ns to run provided provided example). Clock period: 3 # cycles": 2	Logic Area: (um^2) 296.323997 Memory: N/A	1/(delay.area) ($ns^{-1}.um^{-2}$) delay = period.cycles delay = 6 1/ (6*296.323997) = 0.00056244
Delay (TA provided example. TA to complete)		1/(delay.area) (TA)

Abstract:

Researchers discovered that neural networks can be trained so that weights and inputs can be rounded to just their sign, -1 and 1. That allows multiply and add to be replaced by XNORs and bit counting during the convolution stage. The project's goal is to implement a single stage of an all binary convolution neural network. All the weight and input data is binary. They take on values of -1 and 1, which are represented by 0 and 1 respectively. The inputs come from an input SRAM, the weights from a weight SRAM and the outputs are written to an output SRAM memory. This project only accepts a 3x3 weight matrix, a 4x4 input matrix, and a 2x2 output matrix.

Through a multi-stage development and verification process, this project's solution was to continuously compute the convolution since power would only be expended when new inputs were presented which would cause new computations and new values stored in flip-flops. This project achieved a total power of 6.2646e-02 mW, a total cell area of 296.32um^2, a clock period of 3, and it presents the output 2 cycles from when the signal is sent to compute the solution.



Binary Convolution

Jon Mark Long

Abstract:

Researchers discovered that neural networks can be trained so that weights and inputs can be rounded to just their sign, -1 and 1. That allows multiply and add to be replaced by XNORs and bit counting during the convolution stage. The project's goal is to implement a single stage of an all binary convolution neural network. All the weight and input data is binary. They take on values of -1 and 1, which are represented by 0 and 1 respectively. The inputs come from an input SRAM, the weights from a weight SRAM and the outputs are written to an output SRAM memory. This project only accepts a 3x3 weight matrix, a 4x4 input matrix, and a 2x2 output matrix.

Through a multi-stage development and verification process, this project's solution was to continuously compute the convolution since power would only be expended when new inputs were presented which would cause new computations and new values stored in flip-flops. This project achieved a total power of $6.2646e^{-2}$ mW, a total cell area of $296.32um^2$, a clock period of 3, and it presents the output 2 cycles from when the signal is sent to compute the solution.

1. Introduction

a. In order to fulfill the requirments of a single stage binary convolution for a neural network, this project consists of the following hardware:
   i. Input, Weight, and Output SRAMs
   ii. XNORs that compute from Input and Weight data
   iii. Adders to count the number of set bits in previous computation
   iv. Muxes where if the addition equals 5,6,7,8, or 9 then a 1 will be written to the Output SRAM
   v. FlipFlops to hold SRAM addresses, data being written, write enable, and the busy control signal.

b. Summary of results achieved.
   i. Total power: $6.2646e^{-2}$ mW
   ii. Total cell area: $296.32um^2$
   iii. Clock period: 3
   iv. Cycles to compute: 2

c. Table of Contents

2. Micro-Architecture

a. High level architecture drawing, and description of data flow

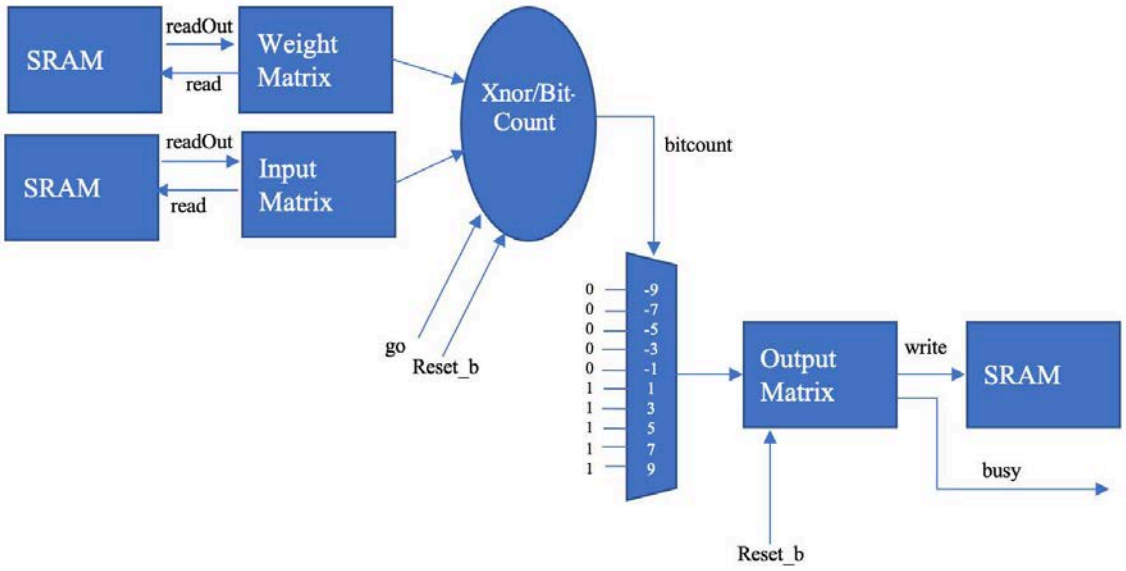
i. Datapath:

1. if reset is presented, the addresses and sram write data are reset, otherwise the addresses loop through a flip flop, and the write data goes through a flip flop and always gets the output matrix data.
2. The following description happens continuously: the weight matrix (3x3) and appropriate regions (top left, top right, lower left, & lower right quadrants) of the input matrix (4x4) are XNORed together.
3. Each of the four outputs of this computation goes through an adder that adds all the set bits from the binary values
4. (e.g. if the XNOR outputs 1100100 then the adder would output 3 because there are 3 set bits).
5. If that addition equals 5,6,7,8 or 9, then a 1 is placed into the appropriate location of the output matrix
6. (e.g. if the last example was for the top left quadrant of the input matrix, then the output of 3, which is not a 5,6,7,8, or 9, would mean a zero is placed in the top left of the output matrix (2x2)).

ii. Controller

1. Since the solution is computed as soon as inputs are presented, the Controller states only monitor the go signal.
2. On the posedge of go, busy and the sram write enable go high.
3. After one clock cycle, busy and sram write enable go back low and wait for a new go signal.

iii. block diagram



3. Interface Specification

a. Top level interface signals and descriptions:

i. Descrition:

1. For this project only one convolution is computed so the sram addresses are fixed at 0. The top module is provided with a clock and reset signal. Once reset goes high to indicate the system is ready for the computation, the input and weight matrices' data is provided. The module precomputes the result as soon as data is provided so as soon as the go signal is presented, busy goes high, the module writes the solution to the output matrix, and busy goes low 2 cycles after go went high to tell the system that the computation is finished and the output matrix now holds the correct value.

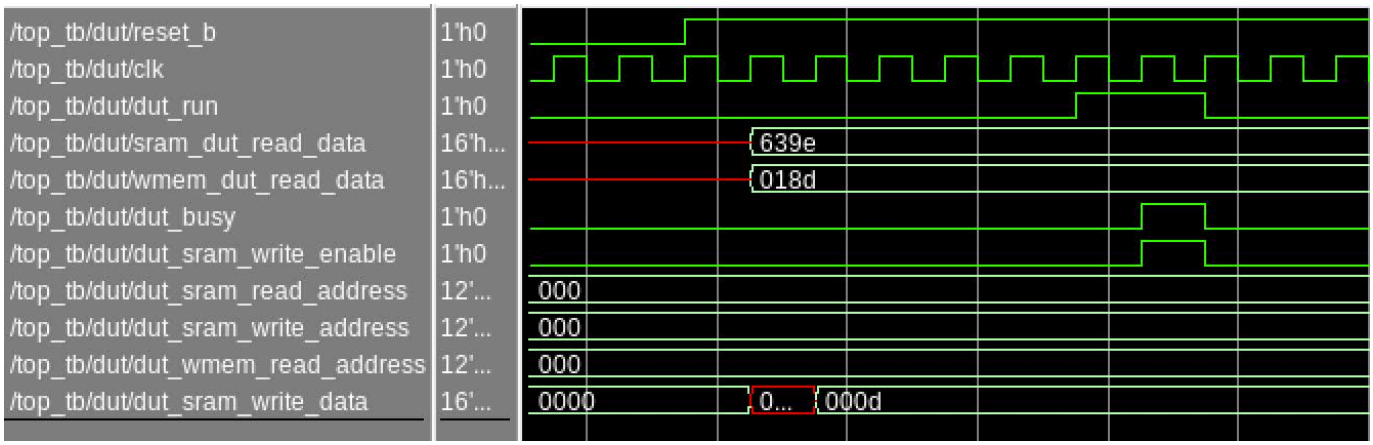
ii. Control signals

1. Inputs
   a. reset_b: Active low synchronous reset signal, clear machine state
   b. clk: System clock
   c. go: Start processing if high, wait if low
2. Output
   a. busy : high while computation occurs, low after computation and when waiting for next go signal

iii. Memory interface

1. Inputs
   a. sram_dut_read_data: 16 bit data input for input matrix
   b. wmem_dut_read_data: 16 bit data input for weight matrix
2. Outputs
   a. dut_sram_write_enable: high if writing, low if reading
   b. dut_sram_read_address: 12 bit address for input matrix
   c. dut_sram_write_address: 12 bit address for output matrix
   d. dut_wmem_read_address: 12 bit address for weight matrix
   e. dut_sram_write_data: 16 bit data to be written to output sram address

b. Interface timing diagram



4. Technical Implementation

a. Hierarchy
    i. Top: this project was implemented using a Top (without memory, the synthesizable portion) module which instantiated a single Datapath and Controller module. The SRAM was instantiated separately.

    ii. Controller: monitors what state the system is in, when control signals reset_b and go are presented, and when to write the output to the output sram.

    iii. Datapath: continuously computes the single stage of the binary convolution neural network and writes the appropriate output matrix to the output sram when the controller sends the sram write enable signal.

5. Verification

a. The provided testbench was used at every step of development to verify that each building block was working as expected before adding a following building block. First, a simple sram read was implemented to make sure that the memory interface was being used correctly, and input data was arriving at the expected time. Next, several iterations of the datapath were tested to make sure that the computation was occuring correctly. Lastly, the timing of go arriving, write enable and busy going high, and the process comletely was all fine tuned so that everything happened at the right time.

6. Results

a. cell report
    i. 305 cells
    ii. 296.3240 area
b. timing_max_slow
    i. 0.0033 slack (MET)
c. timing_max_slow_holdfixed
    i. 0.0033 slack (MET)
d. timing_min_fast_holdcheck
    i. 0.0869 slack (MET)
e. Power
    i. 60.7949 uW Total Dynamic Power
    ii. 1.8508 uW Cell Leakage Power
f. Computation
    i. 2 cycles to complete
g. Clock
    i. period of 3

7. Conclusion
   a. This project accurately implemented a single stage of a binary convolution neural network according to the project specifications. Through a multi-stage development and verification process, this project's solution was to continuously compute the convolution and write the correct output to the output SRAM at the appropriate time. Some adjustment would be needed for this project to compute consecutively and move to the next addresses after each computation. I actually implemented that originally and then removed it to optimized speed, area, and power. Further work would also be needed to expand this project to accept other input and weight matrix sizes.