ECE 464 Project: Binary Artificial Neural Network

Name: Alan Zheng
Unityid: 200265055
StudentID: azheng3

Delay (ns to run provided example)
Clock period: **2.06 ns**
cycles": **4**
Delay = 8.24

Logic Area: (um^2)
603.82

Memory: N/A

1/(delay.area) ($ns^{-1}.um^{-2}$)
1/(2.06*4*603.82)
= 1/4975.48 = 2.01e-4

Delay (TA provided example. TA to complete)

1/(delay.area) (TA)

Abstract
The main purpose of this project is to design a fixed size single stage binary convolutional artificial neural network. The fixed size regards to the 3x3 weight matrix, and a 4x4 input matrix giving us an output matrix of 2x2. The design uses three simulated SRAM units used to read and write values to and from .dat files alongside with my design that allows the values to perform binary convolutions. My initial approach was using a counter as the control strategy that does a set of multiplication for all four convolutions for each weight index. Afterwards, the design sums up the number of 1's and set the output SRAM data. However, I was able to optimize this further making a much more efficient design by reducing the number of states and removing unnecessary registers. The result that the design was able to achieve was an area of 603.82 um^2 with a clock period of 2.06 nanoseconds with a performance of 4975.48.

ECE 464 Project: Binary Artificial Neural Network

Alan Zheng

Note. This outline is not intended as a rigid structure but as guidance.

1. Introduction

The design created here is a fixed Binary Convolution Artificial Neural Network with a fixed input size of 4x4, a weight size of 3x3 which results to an output matrix of 2x2. The control strategy used was a counter that performs a linear sequence of states. This resulted in a unit that can handle the fixed inputs and weight with a total area of 603.82 um^2 with a clock period of 2.06 nanoseconds.

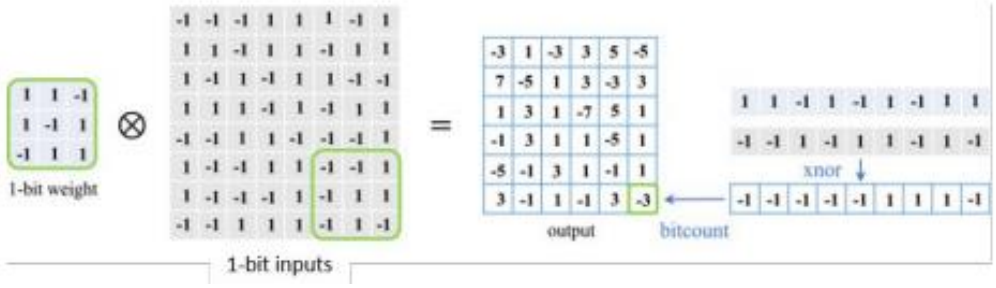

Figure 1: Binary Convolution Example

A. Convolution and File format

The main mathematical formula performed here is a binary convolution. An example of a binary convolution can be seen in Figure 1, where the weight matrix is multiplied with the values in the input matrix before it is summed up and calculated to an output of 1 for positive or 0 for negative. The multiplication regarding the binary output is exactly like a 2 input XNOR gate show in Figure 2. There are four outputs because the 3x3 weight matrix can "fit" in the input matrix 4 different ways see Figure 3, with 9 total multiplications where each weight gets one input value for each output value.

Weight Bit	Input Bit	Multiplication
-1	-1	1
-1	1	-1
1	-1	-1
1	1	1

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	1

Figure 2: Convolution Multiplication and XNOR Truth Table

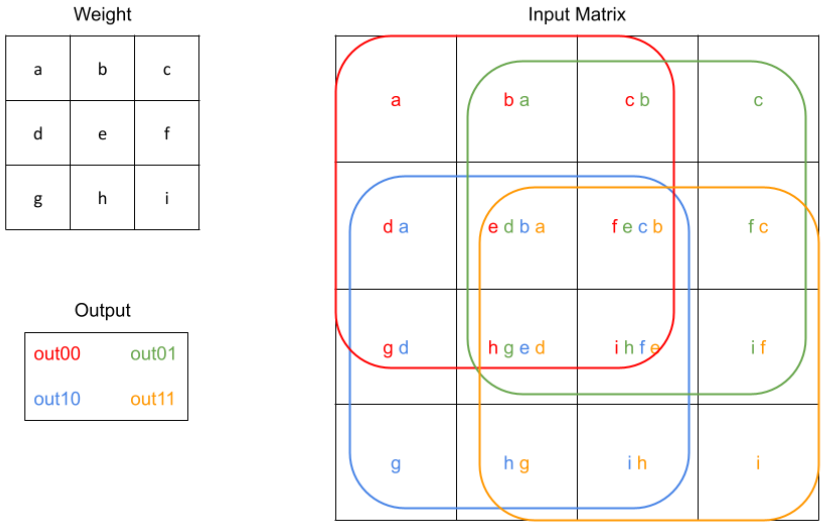

Figure 3: Multiplication

It is also important to clarify the format of the 3 .dat files that values are extracted and imported from. In Figure 4, we can observe how the .dat file is structured with the left-hand number being the address and the right hand being the values in the matrix. In Figure 5, we can see how that number is converted into a matrix that is used for convolution.

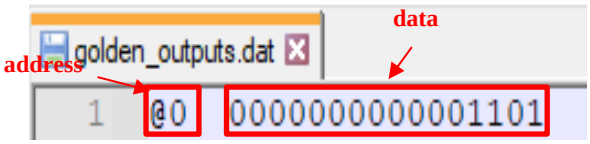

Figure 4: Example .dat file structure

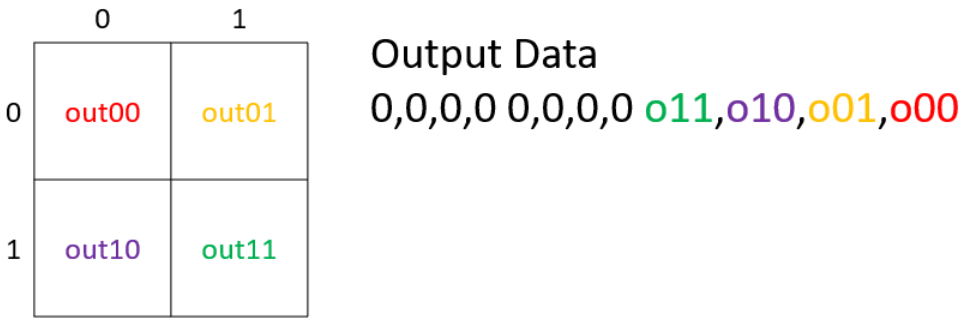

Figure 5: Matrix to Data

B. Report Overview

The rest of this report will outline 3 major sections one explains some design implementations and algorithms, the second describes the timing diagram and design interface, and third, the verification techniques used to ensure correct functionality.

2. Micro-Architecture

This design uses a counter that has a linear sequence of states with a few if statements that decides that progress. In Figure 6, we can observe the counter going through the linear path along with that is done along the way.

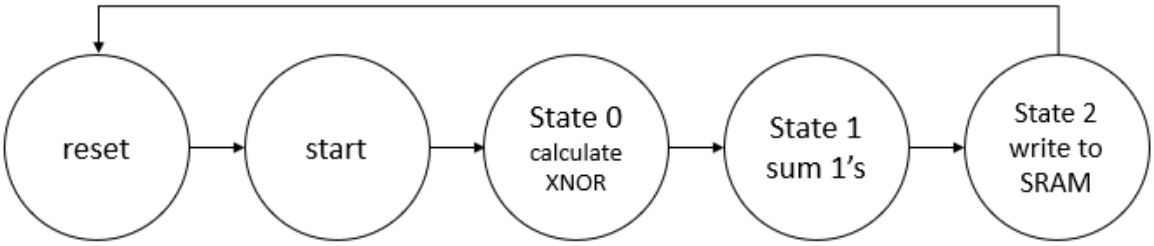

Figure 6: Design General Counter Diagram

Although the counter is controlled by if statements, there is also a case state where main convolution takes place without much logic blocking it. There are 3 case states; state 0, performs the convolution multiplication by using XNOR with the input and weights, state 1 sets a flag high to perform the summation of the 1's in each convolution multiplication, state 2, checks the summed value if they are above the threshold, positive (1) vs negative (0), which the output sets and writes to the SRAM. State 3 also sets write address and turns write enable high.

3. Interface Specification

There are two types of interfaces that can be discussed here one being the SRAM and second being my design.

A. SRAM Interface

In Figure 7, we can see that input, output, and control signals of the SRAM.

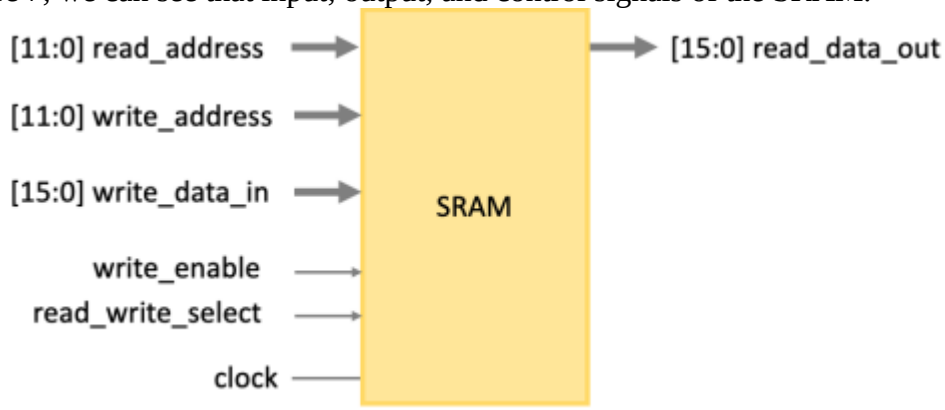

Figure 7: SRAM Interface

The three SRAMs are called in the testbench in not in my design. The SRAM interface signal read_write_select and write_enable are used to determine if the SRAM is reading an address or writing to one. The read_address and write_address are just like what it means, the address that it is reading/writing to. Similarly write_data_in and read_data_out indicates what is being read or written.

B. Design Module Interface

All three SRAM has the same interface however, not all the input/outputs are used and solely depend on the interface of my design. In Figure 8, we can observe that the output

of the SRAM on the input and weight SRAM is the input to the interface and the output of the interface is the input of the write SRAM.

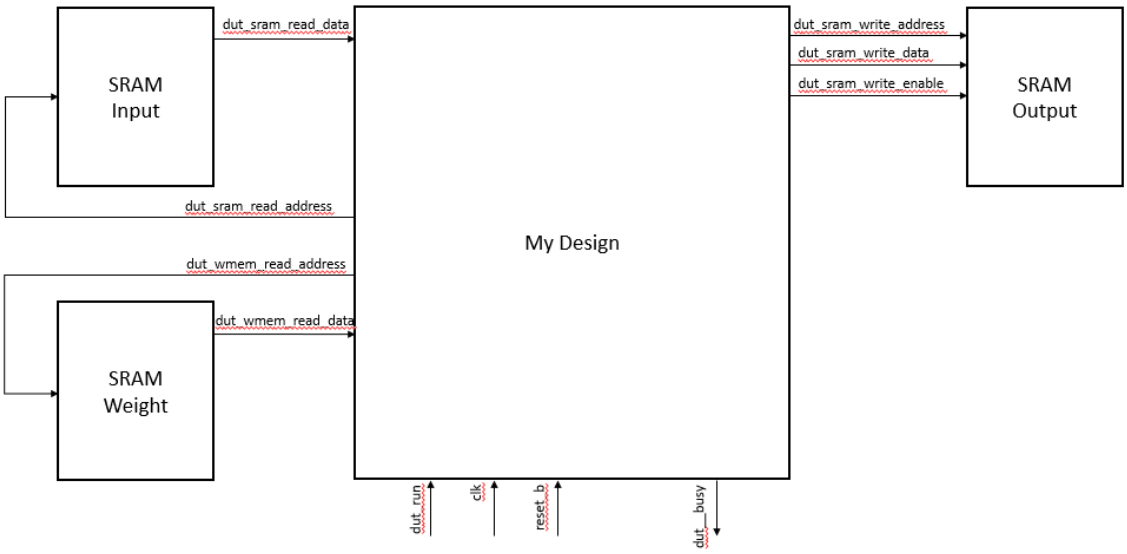

Figure 8: Design Interface

In Table 1, we can see which signals are included in the interfaced described in Figure 8 as well as what their functions are.

Design Interface Signals	
Input Signals	
dut_run	Starts the calculation
clk	Master clock
reset_b	Set module to waiting state and reset registers
dut_sram_read_data	Result value is passed to module from Input SRAM
dut_wmem_read_data	Result value is passed to module from Weight SRAM
Output Signals	
dut_busy	Set to high when module is working
dut_sram_read_address	Sets address and sent to Input SRAM
dut_sram_write_address	Sets address and sent to Output SRAM
dut_sram_write_data	Sets data and sent to Output SRAM
dut_sram_write_enable	Set to high and let Output SRAM to write dut_sram_write_data to dut_sram_write_address
dut_wmem_read_address	Sets address and sent to Input SRAM

Table 1: Interface Signal

C. ModelSim Waveforms

There are two major area of interest when looking at the timing diagram. There is an initial setup where the SRAM fetches the input from input_sram.dat and weight_sram.dat based on the address set by the reset controller in the top design. The second stage computes the binary convolution based on the input and weight values and writes the

output to an SRAM once both dut_busy is low and write_enable is high. We can see both stages in Figure 9.

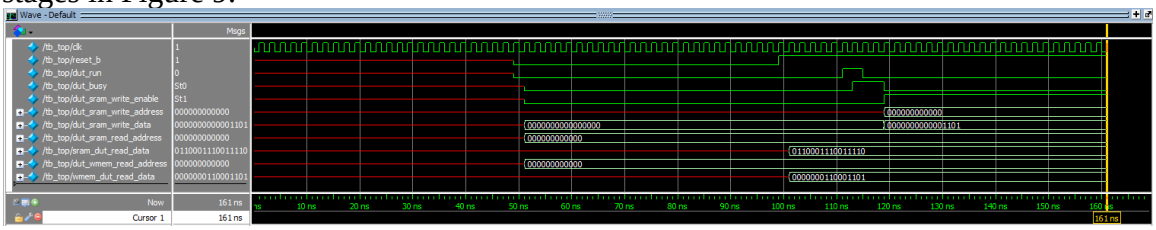

Figure 9: ModelSim Waveform at Fast Clock Period

In Figure 9, the first column indicates the signal name on each row. The column next to that indicates the final values, in binary, of the signals. Finally, the third section shows the behavior of each signal as well as indicating the values of that signal in binary at a given point in time.

5. Verification

There were many hours spent in debugging as well as verifying that the convolution given has the proper values at each stage of the counter. By using $display commands in the top module, the values can be displayed in ModelSims transcript for debugging purposes. The command was used to ensure that the convolution multiplication results in the correct array of bits as well as the proper number of 1's and 0's.
This test requires the use of a .dat file with expected outputs and compares with our result. To verify the correctness of my design, I have tested the design against additional inputs, weight matrix, and intended output alongside with the one provided. The input and weight matrix I have made are randomly generated using RNG.

Additionally, the design can also be verified by using Synopsys to ensure that there are no unintended latches as well as optimizing the clock period to area ratio which can be seen in the next section.

6. Results Achieved

To check the physical characteristics of the design, the module was synthesized using Synopsys2019 over NC States remote Grendel server. The module was synthesized with pre-configured set of constraints defined in script1Project.tcl and script2Project.tcl which are included by the instructor. Proceeding the synthesis, a report of the design's timing and area can be used. The results can be observed in Figure 10 and 11 respectfully.

```
****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : MyDesign
Version: P-2019.03-SP1
Date   : Mon Nov 15 20:22:46 2021
****************************************

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

  Startpoint: convolution3_reg_4_
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: dut_sram_write_data_reg_3_
            (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
  convolution3_reg_4_/CK (DFF_X2)                         0.0000     0.0000 r
  convolution3_reg_4_/QN (DFF_X2)                         0.3320     0.3320 f
  U332/ZN (INV_X4)                                        0.0911     0.4231 r
  U645/ZN (INV_X2)                                        0.0405     0.4637 f
  U335/ZN (NAND2_X2)                                      0.0764     0.5400 r
  U333/ZN (NAND2_X2)                                      0.0592     0.5993 f
  U331/ZN (NAND2_X2)                                      0.1055     0.7048 r
  U181/ZN (INV_X1)                                        0.0527     0.7575 f
  U397/ZN (NAND2_X1)                                      0.1667     0.9242 r
  U601/ZN (NAND3_X2)                                      0.0925     1.0166 f
  U338/ZN (NAND2_X2)                                      0.1065     1.1231 r
  U173/ZN (NAND2_X1)                                      0.0800     1.2031 f
  U437/ZN (INV_X1)                                        0.0792     1.2823 r
  U436/ZN (NAND2_X1)                                      0.0563     1.3386 f
  U435/ZN (NAND2_X1)                                      0.1168     1.4554 r
  U592/ZN (NOR2_X2)                                       0.0782     1.5336 f
  U591/ZN (NAND2_X2)                                      0.1191     1.6527 r
  U648/ZN (NAND2_X2)                                      0.0586     1.7113 f
  dut_sram_write_data_reg_3_/D (DFF_X2)                   0.0000     1.7113 f
  data arrival time                                                  1.7113

  clock clk (rise edge)                                   2.0600     2.0600
  clock network delay (ideal)                             0.0000     2.0600
  clock uncertainty                                      -0.0500     2.0100
  dut_sram_write_data_reg_3_/CK (DFF_X2)                  0.0000     2.0100 r
  library setup time                                     -0.2977     1.7123
  data required time                                                 1.7123
  -----------------------------------------------------------------------
  data required time                                                 1.7123
  data arrival time                                                 -1.7113
  -----------------------------------------------------------------------
  slack (MET)                                                        0.0010
```

Figure 10: Timing Report

```
****************************************
Report : area
Design : MyDesign
Version: P-2019.03-SP1
Date   : Mon Nov 15 20:22:54 2021
****************************************

Library(s) Used:

    NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm (File: /afs/eos.ncsu.edu/lockers/researc
h/ece/wdavis/tech/nangate/NangateOpenCellLibrary_PDKv1_2_v2008_10/liberty/520/NangateOpenCellL
ibrary_PDKv1_2_v2008_10_slow_nldm.db)

Number of ports:                           89
Number of nets:                           629
Number of cells:                          571
Number of combinational cells:            479
Number of sequential cells:                44
Number of macros/black boxes:               0
Number of buf/inv:                        143
Number of references:                      23

Combinational area:               393.147996
Buf/Inv area:                      76.608001
Noncombinational area:            210.672005
Macro/Black Box area:               0.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:                  603.820000
Total area:                 undefined
1
```

Figure 11: Area Report

```
****************************************
Report : power
        -analysis_effort low
Design : MyDesign
Version: P-2019.03-SP1
Date   : Mon Nov 15 20:22:58 2021
****************************************


Library(s) Used:

    NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm (File: /afs/eos.ncsu.edu/lockers/researc
h/ece/wdavis/tech/nangate/NangateOpenCellLibrary_PDKv1_2_v2008_10/liberty/520/NangateOpenCellL
ibrary_PDKv1_2_v2008_10_slow_nldm.db)


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  =  59.1135 uW   (83%)
  Net Switching Power  =  11.8655 uW   (17%)
                         ---------
Total Dynamic Power    =  70.9789 uW  (100%)

Cell Leakage Power     =   2.5543 uW


                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  A
ttrs
--------------------------------------------------------------------------------------------
----
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       4.3479e-02       2.5823e-03        3.4836e+05        4.6410e-02  (  63.11%)
sequential         0.0000           0.0000            0.0000            0.0000  (   0.00%)
combinational  1.5635e-02       9.2832e-03        2.2059e+06        2.7124e-02  (  36.89%)
--------------------------------------------------------------------------------------------
----
Total          5.9113e-02 mW     1.1865e-02 mW     2.5543e+06 pW     7.3533e-02 mW
1
```

Figure 12: Power Report

The important aspect in timing is having a tight, nonnegative slack value, which was calculated as 0.001. A loose slack means there are additional breathing room for the design. A negative slack means that the timings are not met, thus are violated. It is important to note that adjusting the clock period can affect the slack, area, and power consumption. As clock periods decrease, area and power consumption increase. The power consumption information can be observed in Figure 12.
In this module, the total area is 603.82, a slack of 0.001, a clock period of 2.06 ns, and the total dynamic power of 70.9789 uW.

7. Conclusions

The Fixed Binary Convolution Artificial Neural Network is designed using a counter and 3 SRAMs. This designed was verified using $display at the critical variables in the stages where their values must be right to have proper output. The functionality is confirmed by the testbench which compares the intended output with the user output. Once verification is completed, synthesis can be performed to give stats on the physical design of the module.
As for the design, I believe I made my code efficient in terms of clock period and power consumption. The main issue I had when designing was trying to understand binary convolution and converting that into design. Even though I spent a few hours understanding the project, ultimately it was crucial as I was able to get a working prototype within a few hours and a synthesizable prototype afterwards in another few hours. Despite multiple hours spent in optimizing the final prototype, I am very pleased with my final design.