ECE 564
Binary Artificial Neural Network

Abstract

Convolutional neural networks are a powerful tool for many classification tasks but running their inherently parallel structure on a conventional serial computer can be prohibitive. To address this, FPGAs and ASICs have emerged as potential solutions, achieving better performance, power consumption, and speed through their inherent parallelism. This report summarizes the design and performance of custom hardware that computes one stage of an all-binary convolutional neural network (CNN). The design takes rows of an input matrix (maximum of 16 bits wide though 12- or 10-bit rows are also acceptable) from an SRAM and convolves it with a 3x3 kernel from another SRAM. The module computes the sign activation of the convolution with XNORs and bit-counting before writing to an output SRAM. The approach chosen uses a pipelined and parallelized datapath, enabling maximum performance given the SRAM memory bottleneck. In the final design, 46 clock cycles with a 5ns clock are needed to compute the convolution of a 16x16, 12x12, and 10x10 matrix from the input SRAM.

1. Introduction

This report details the development, design, and performance of an ASIC implementing a single stage of an all-binary convolutional neural net (CNN). Whereas typical CNNs compute full convolutions (or for a more mathematically correct term, cross-correlation), binary convolutional neural nets round each value to -1 or 1. By encoding -1 as a 0, multiplication is logically equivalent to XNOR, allowing small XNOR gates instead of large and power-hungry multipliers. This design connects to an input SRAM holding the rows of an input matrix, an input SRAM holding the weights of a 3x3 kernel, and an output SRAM holding the result. Using a pipelined functional unit and a simple state machine, the design achieves close to theoretically optimal performance given the memory bottleneck.

The key innovations of this design are in the datapath. Instead of solely parallelizing the computation (the approach of SIMD instructions), the datapath caches input rows and computes the convolution over a 3x16 window in a clock cycle. This cuts memory accesses, and therefore clock cycles, by 1/3. To address different input sizes, muxes toggle between the convolution results and a fixed logic low. This allows one datapath to work on multiple input sizes.

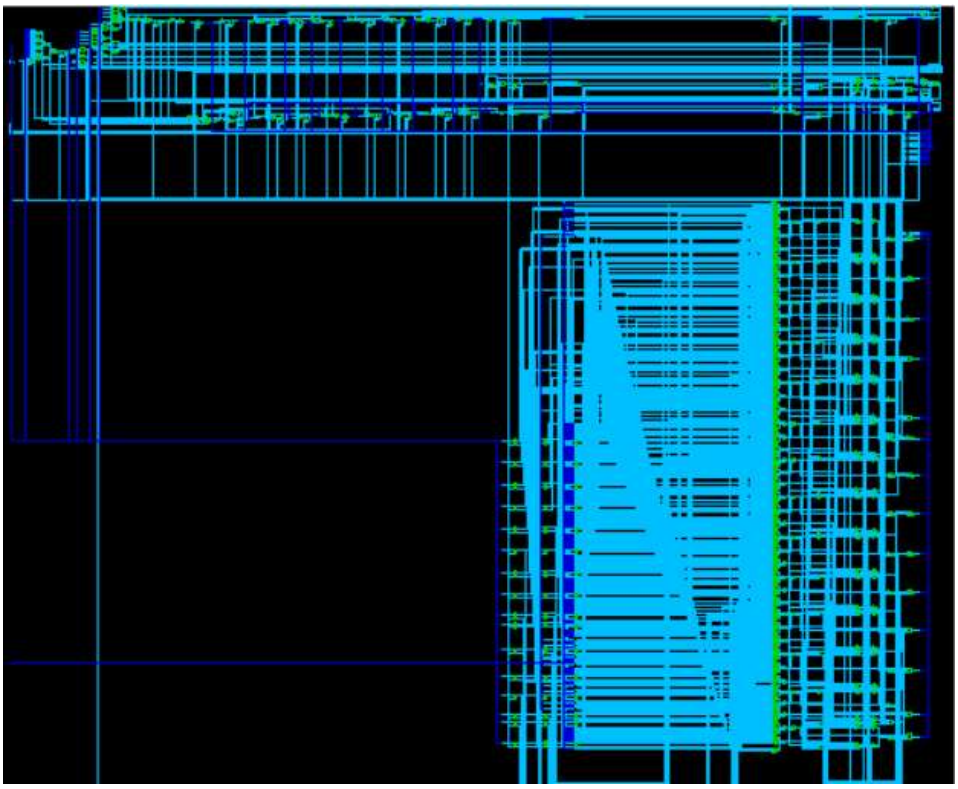

Figure 1: Final Design (datapath on the right, controller on top)

This design achieves overall good performance, convolving a 16x16, 12x12, and 10x10 input matrix with a 3x3 kernel in 46 clock cycles at a 5ns clock period. The theoretical minimum number of cycles for this would be 1+1+1+16+12+10=41, since each unique dimension and row would be read once. However, additional clock periods are used to start the pipeline and exit the convolution sequence. At 1300.474um^2 with a 5ns clock, the design meets timing, area, and performance constraints.

The rest of this report discusses the specifics of the design. Section 2 elaborates on the hardware-level algorithmic approach and datapath design. Section 3 shows the interfaces for each module and the design hierarchy. Section 4 further discusses the design and any high-level modeling used. Section 5 describes the verification approach. Section 6 includes all results achieved, and section 7 summarizes the project.

2. Micro-Architecture

The micro-architecture is split into two components: the datapath and the controller.

The controller is a state machine with an embedded counter. Since the hardware implementation depends on the state encoding, the state diagram was directly encoded in RTL and the Synopsys Design Compiler handled logic optimization. At each state control signals set the next values of registers storing SRAM memory addresses, state, and input dimension.

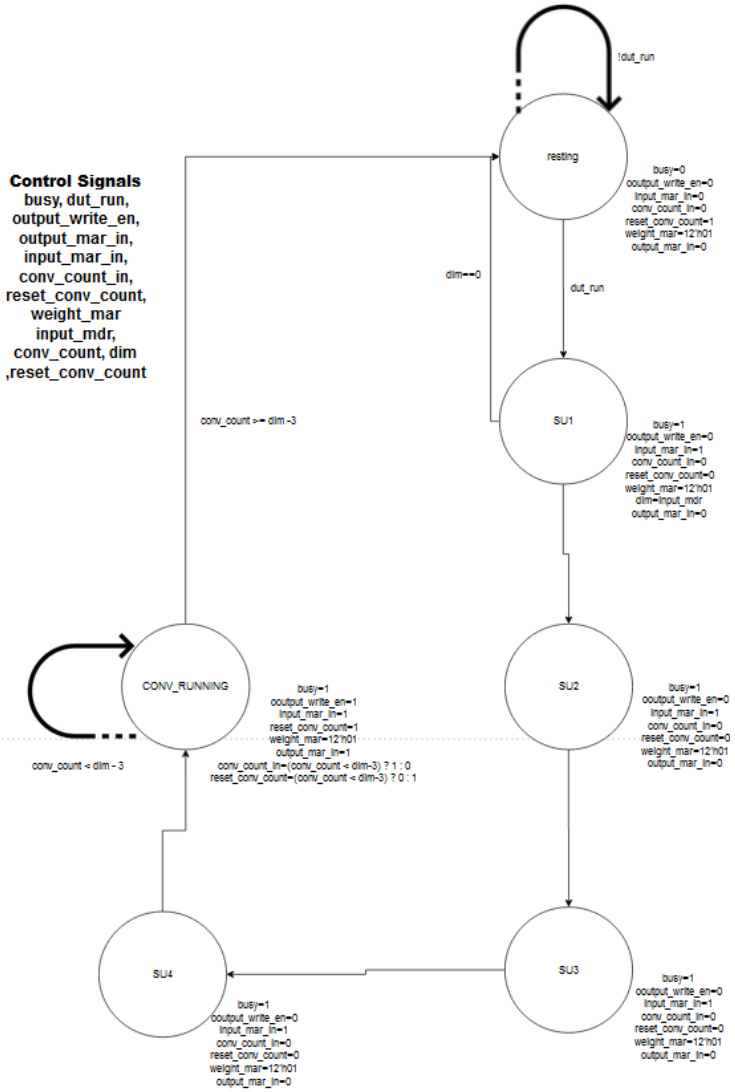

Figure 2: Controller State Machine

The datapath design is more concrete. A 3x16 array of flip flops holds 3 rows of the input matrix, and on each positive clock edge, the data in each row shifts up one row. Fourteen convolution functional units are attached to every 3x3 subregion of this cache. At the outputs of the functional units, multiplexers can either accept or reject the result depending on input dimension.

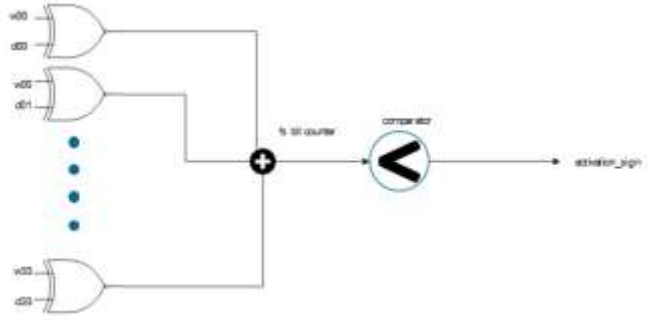

Figure 3: Single Convolution Unit

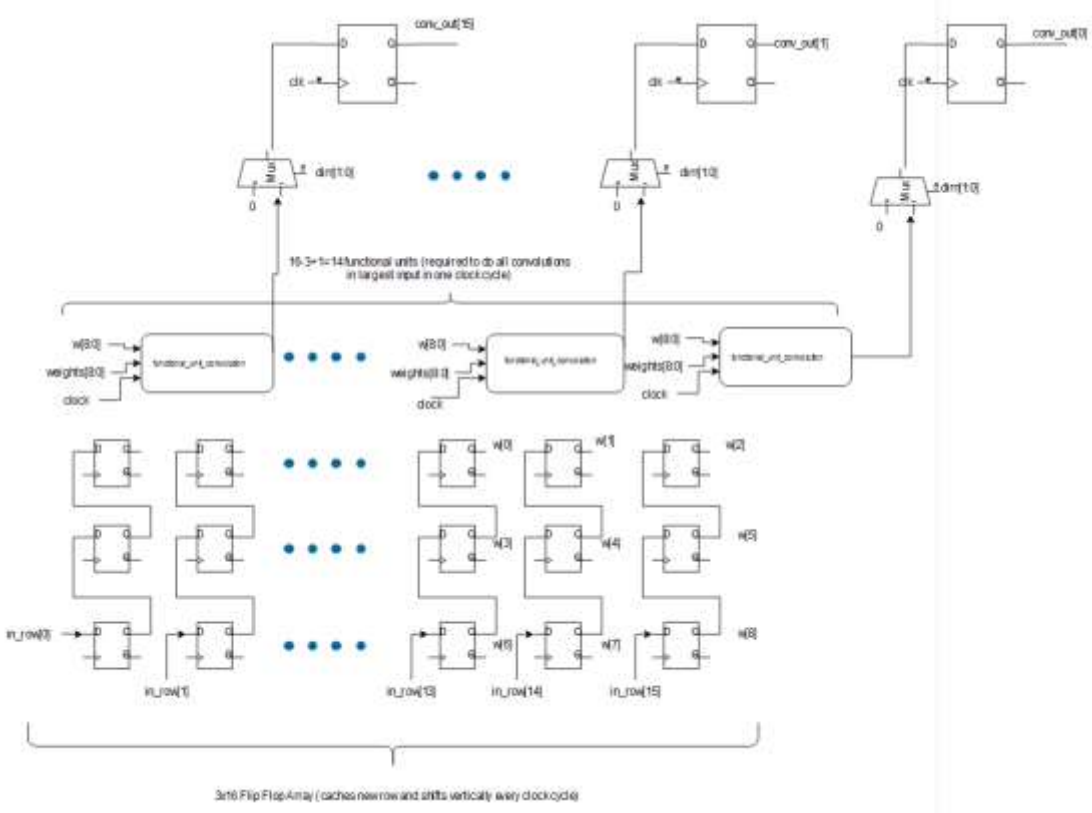

Figure 4: Pipelined Convolution Unit

The datapath is fed the input rows and dimension by the controller, which modulates the output SRAM write enable signal to store outputs to memory. Since three clock cycles must pass for all flip flops to hold valid data, the controller takes a three clock cycle delay into account.

3. Interface Specification

The top level interface is three similar groups of connections for each SRAM (using an interface object may be advantageous) and some simple control signals for resetting the state machine and indicating chip activity. All signals for the top-level module are shown in the table below.

Signal Name	**Type**	**Function**
dut_run	input	Synchronous start signal
reset_b	input	Asynchronous reset signal
clk	input	External clock
dut_busy	output	Goes high when convolution is being computed
sram_dut_read_data	input[15:0]	Input SRAM data
wmem_dut_read_data	input[15:0]	Weight SRAM data
dut_sram_write_data	output[15:0]	Data to write to output SRAM
dut_sram_write_address	output[11:0]	Address to write to in output SRAM

dut_sram_read_address	output[11:0]	Address to read from in input SRAM
dut_wmem_read_address	output[11:0]	Weight SRAM read address
dut_sram_write_enable	output	Write enable for output SRAM

4. Technical Implementation

Since the time needed to make a high level model and implement the design in Verilog is roughly equivalent, beyond a basic Python implementation of the all-binary convolution, no high level modeling was used. Multiple other approaches were considered (different datapaths for each input size, shift registers, copying multiple matrices and concurrently doing the convolutions), but clear issues in each approach removed the need for high level modeling.

As previously discussed, the design was split hierarchically into a datapath and controller. The controller is a simple state machine with an embedded counter, and the datapath is a 3x16 bank of registers connected to 14 convolution units. The top level module without SRAM is shown below in figure 5. Note: this hierarchy is only for the convenience of the designer. The design did not have enough gates to cause issues for the design compiler, but the existing partitions do not contain the critical path in one module or register all outputs. Therefore, the design was flattened in synthesis.

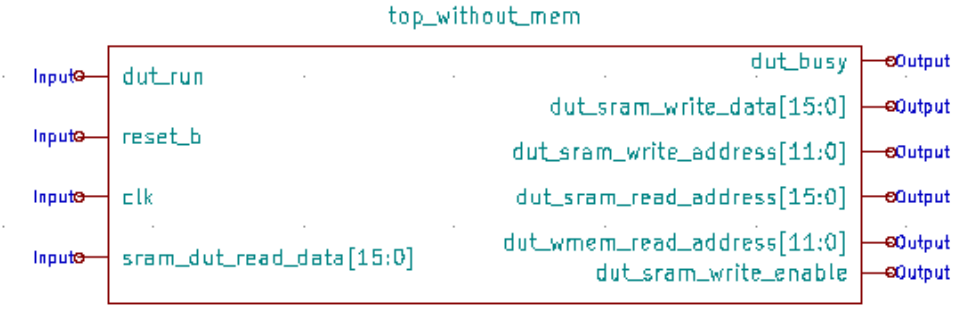

Figure 5: module top_without_mem

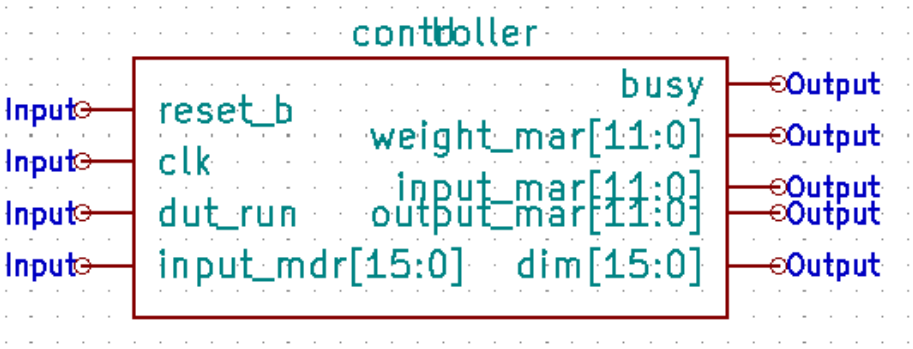

Figure 6: module controller

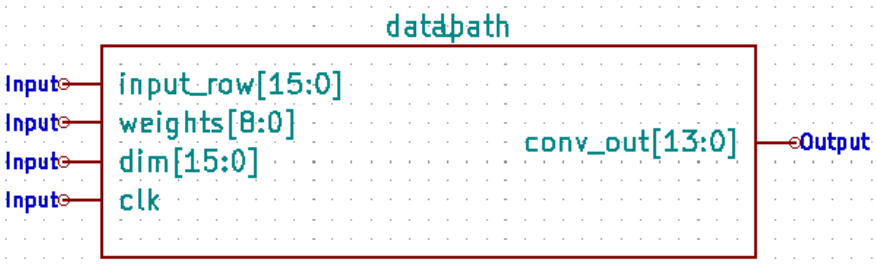

Figure 7: module datapath

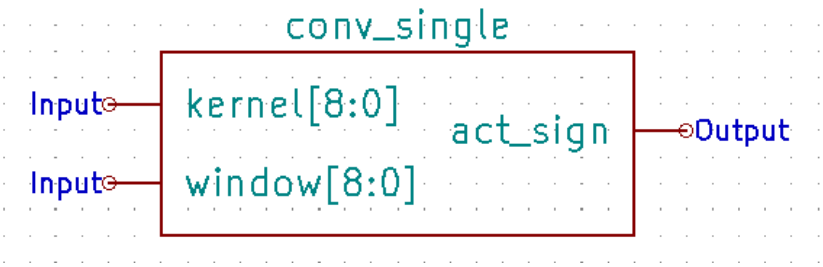

Figure 8: module convolution unit

5. Verification

Verification was done in a bottom up order starting with the individual convolution units. A .dat file of possible windows and a fixed kernel were fed to the convolution_unit.v file. The same files were put through the Python model and the results were compared.

A similar process was applicable for the datapath file. Rows of each matrix were fed to the unit, and the results were compared to the expected outputs. Note that the first two lines of an output are incorrect but are required to start the pipeline.

Finally, the top level design was verified with the testbench provided by the instructors and TAs; this waveform is shown below. At the start and end of each example, the current and next state registers were observed along with the output write location and data in the waveforms window. The testbench log (clipped below) also showed no errors. Additional verification with more example matrices, reordered matrices, or new kernels is possible, but the functional coverage was reasonably high.

Clipped Output Transcript

```
# ------------------------------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=46
# ----------------------------------------------------------------------------------
#
…
# ------------------------------load results to output_array------------------------------
#
# ------------------------------load results to golden_output_array------------------------------
#
```

```
# ------------------------------Round 1 start compare ------------------------------
#
# ------------------------------Round 1 Your report-------------------------------
#
# Check 1 : Correct g results = 32/32
# computeCycle=46
# ---------------------------------------------------------------------------------
#
```

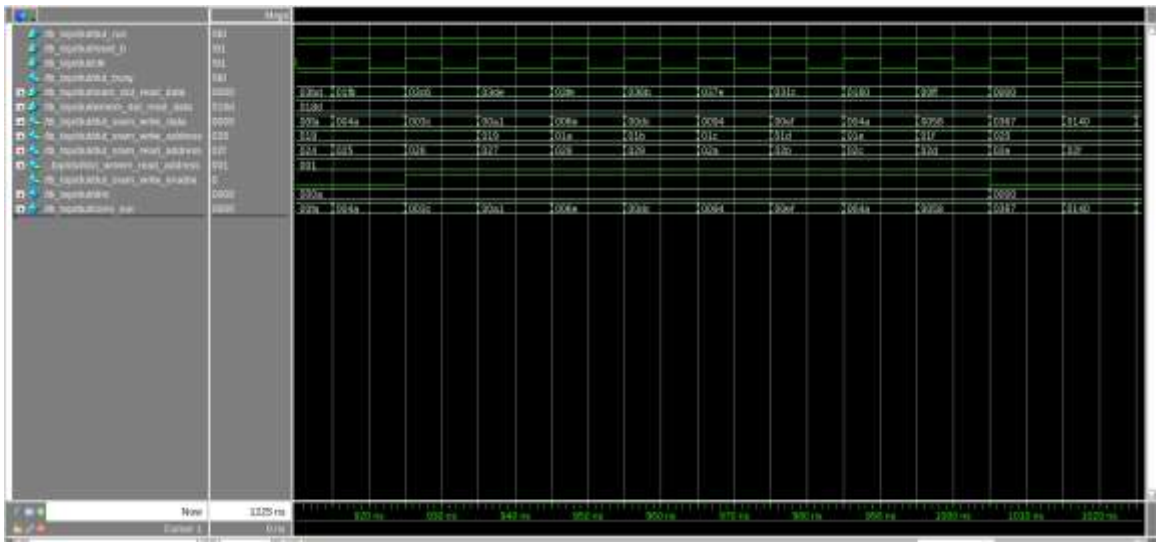

Figure 9: DUT Waveform

6. Results Achieved

The example testbench runs through a 16x16, 12x12, and 10x10 matrix convolution in 46 cycles at a 5ns clock period. All timing constraints are met even with the aggressive clock period. More modest clock speeds are possible but halving the clock period from 10ns to 5ns only incurred a roughly 85um^2 increase in area. The power, cell area, and timing information reports from Synopsys Design Compiler are given below. The final synthesis report shows 7 warnings (all related to not using the top 7 bits of an input value and safely ignorable).

Clipped Power Report

```
****************************************
Report : power
        -analysis_effort low
Design : top_without_mem
Version: P-2019.03-SP1
Date   : Sun Nov 14 13:49:19 2021
****************************************


Library(s) Used:

    NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm (File:
/afs/eos.ncsu.edu/lockers/research/ece/wdavis/tech/nangate/NangateOpenCellLibrary_PDKv1_2_v2008
_10/liberty/520/NangateOpenCellLibrary_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 = 119.4297 uW (78%)
Net Switching Power = 33.5198 uW (22%)

Total Dynamic Power = 152.9494 uW (100%)

Cell Leakage Power = 4.6971 uW

Power Group	Internal Power	Switching Power	Leakage Power	Total 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	6.3543e-02	3.8111e-03	9.7058e+05	6.8325e-02 (43.34%)
sequential	0.0000	0.0000	0.0000	0.0000 (0.00%)
combinational	5.5886e-02	2.9709e-02	3.7265e+06	8.9321e-02 (56.66%)
Total	0.1194 mW	3.3520e-02 mW	4.6971e+06 pW	0.1576 mW

Clipped Cell Report

u2/R3_reg[12]	DFF_X1	NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm	4.7880 n
u2/R3_reg[13]	DFF_X1	NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm	4.7880 n
u2/R3_reg[14]	DFF_X1	NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm	4.7880 n
u2/R3_reg[15]	DFF_X1	NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm	4.7880 n
Total 673 cells			1300.4740

Timing Reports

****

Report : timing
-path full
-delay max
-max_paths 1
Design : top_without_mem

```
Version: P-2019.03-SP1
Date   : Sun Nov 14 01:08:18 2021
****************************************

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

  Startpoint: wmem_dut_read_data[5]
              (input port clocked by clk)
  Endpoint: dut_sram_write_data[5]
            (output port 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
  input external delay            0.6580     0.6580 r
  wmem_dut_read_data[5] (in)      0.0788     0.7368 r
  U475/ZN (INV_X4)                0.0801     0.8169 f
  U418/ZN (INV_X1)                0.4371     1.2540 r
  U476/ZN (AOI22_X1)              0.2378     1.4918 f
  U482/S (FA_X1)                  0.9365     2.4283 r
  U480/ZN (NAND2_X1)              0.0988     2.5271 f
  U481/ZN (OAI21_X1)              0.2434     2.7705 r
  U487/ZN (NOR2_X1)               0.1074     2.8779 f
  U441/ZN (INV_X1)                0.0872     2.9651 r
  U378/ZN (NAND2_X1)              0.0768     3.0419 f
  U488/Z (XOR2_X1)                0.3645     3.4064 f
  U490/ZN (OAI21_X1)              0.2659     3.6723 r
  U455/ZN (INV_X1)                0.0649     3.7373 f
  U491/ZN (AOI21_X1)              0.4458     4.1831 r
  U492/ZN (INV_X4)                0.1739     4.3569 f
  dut_sram_write_data[5] (out)    0.0000     4.3569 f
  data arrival time                          4.3569

  clock clk (rise edge)           5.0000     5.0000
  clock network delay (ideal)     0.0000     5.0000
  clock uncertainty              -0.0500     4.9500
  output external delay          -0.5660     4.3840
  data required time                         4.3840
  -----------------------------------------------------------
  data required time                         4.3840
  data arrival time                         -4.3569
  -----------------------------------------------------------
  slack (MET)                                0.0271

****************************************
Report : timing
       -path full
       -delay min
       -max_paths 1
Design : top_without_mem
Version: P-2019.03-SP1
Date   : Sun Nov 14 01:08:17 2021
```

```
****************************************

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

  Startpoint: u2/R1_reg[0]
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: u2/R2_reg[0]
            (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
  u2/R1_reg[0]/CK (DFF_X1)                0.0000     0.0000 r
  u2/R1_reg[0]/Q (DFF_X1)                 0.0597     0.0597 r
  u2/R2_reg[0]/D (DFF_X1)                 0.0000     0.0597 r
  data arrival time                                  0.0597

  clock clk (rise edge)                   0.0000     0.0000
  clock network delay (ideal)             0.0000     0.0000
  clock uncertainty                       0.0500     0.0500
  u2/R2_reg[0]/CK (DFF_X1)                0.0000     0.0500 r
  library hold time                      -0.0162     0.0338
  data required time                                 0.0338
  -----------------------------------------------------------
  data required time                                 0.0338
  data arrival time                                 -0.0597
  -----------------------------------------------------------
  slack (MET)                                        0.0259
```

7. Conclusions

This report details the design and performance of custom hardware for computing one stage of an all-binary convolutional neural net. The design rapidly convolves a 3x3 input kernel with an input matrix, transferring data between two input SRAMs and one output SRAM. Key results include maximizing on the given memory bandwidth constraint, relatively small area (1300 um^2), high clock speed (0.2GHz), fully met timing constraints, and a flexible controller and datapath that can be adapted to more complex scenarios.