Final Project Report
Single Stage Convolution of Binary Artificial Neural Network

Name: Melvin Orichi Socana
Unityid: msorichi
StudentID:. 200245135

Delay (ns to run provided provided example).
Clock period: 7
cycles": 4

Logic Area: (um^2)
680.162006 um^2

1/(delay.area) ($ns^{-1}.um^{-2}$)

Delay (TA provided example. TA to complete)

1/(delay.area) (TA)

Single Stage Convolution of Binary Artificial Neural Network

Melvin Orichi Socana

Abstract

The purpose of this design is to act as a single stage convolution for use in a binary convolutional neural network. The implementation has a fixed size with a 4x4 input matrix and 3x3 kernel matrix. The design is interfaced by two SRAMs that contain information about the input and weight matrix as well as act as a destination for the resultant matrix. The algorithm uses a finite state machine that waits for a signal, "run/go," and reads in the matrices. The matrices are sliced up to create the illusion of a sliding window and features are subsequently extracted through a crudely developed function. Once the final 2x2 result matrix is extracted, it is zero-padded with 12 bits to correctly format into the output SRAM. The total area used was approximately 680 um^2 with a clock period of 7 ps. Although results were satisfactory, it is slightly irritating to not have an easily scalable solution.

1. Introduction

The design being implemented is a single convolutional stage of a neural network. This design is created via a finite state machine with 4 states. All weights and inputs are taken in as binary. The overall idea of convolution is relatively simple. The kernel matrix is overlayed over the top-left corner of the input matrix to start and the element-wise product or Hadamard product is calculated. After multiplying all values the values are summed together and represent one feature of the output. The kernel window then slides to the next set of inputs and this process repeats until all features have been extracted. Values of -1 are routed to 0 bit while values of 1 are routed 1 bit. These inputs arrive from an input SRAM while the final output is written to an output SRAM. No pooling or fully-connected layers are included in the design.

The resulting microarchitecture, variable functions, high-level implementations, and results are referenced in the following sections.

2. Micro-Architecture

There are 7 inputs that go into the design. Three control signals including a reset signal, the clock signal, and the go signal to tell the design to start computing the feature map are used. The last four input signals are the read addresses and data from the SRAMs. There are four outputs that the design creates. A busy signal indicates if the design is still attempting to compute the first input and kernel matrix is has recently received. The last three outputs are used to interface with the output SRAM and write to memory. These outputs include a write enable, a write address, and the data to write into the SRAM.

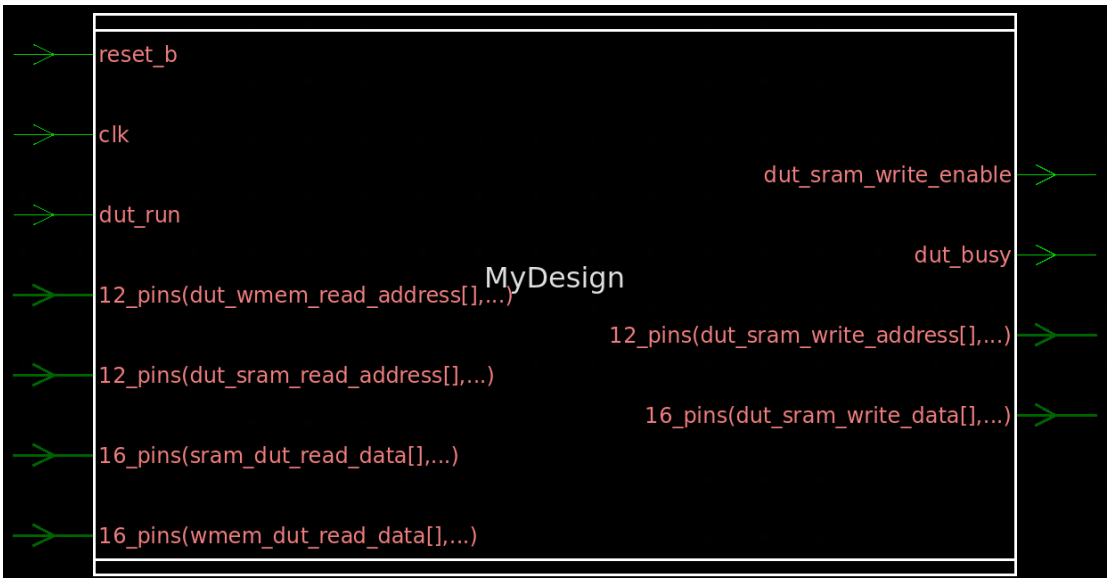

Figure 1 – High Level Design

3. Interface Specification

The top-level module used was a provided testbench specifically to ensure the design meets project specifications. Three SRAMS representing the input SRAM, kernel SRAM, and output SRAM are created and subsequently interfaced with the convolution design. After a hard reset, the run signal is set high, and the resulting outputs begin to be computed. Although this project did not require it, ideally the address of the input matrices would be incremented, the data those addresses pointed to would update, and an output would be recomputed for each round of values. The figure below shows a basic timing diagram for a single computation that takes 4 computation cycles and correctly creates the desired output.

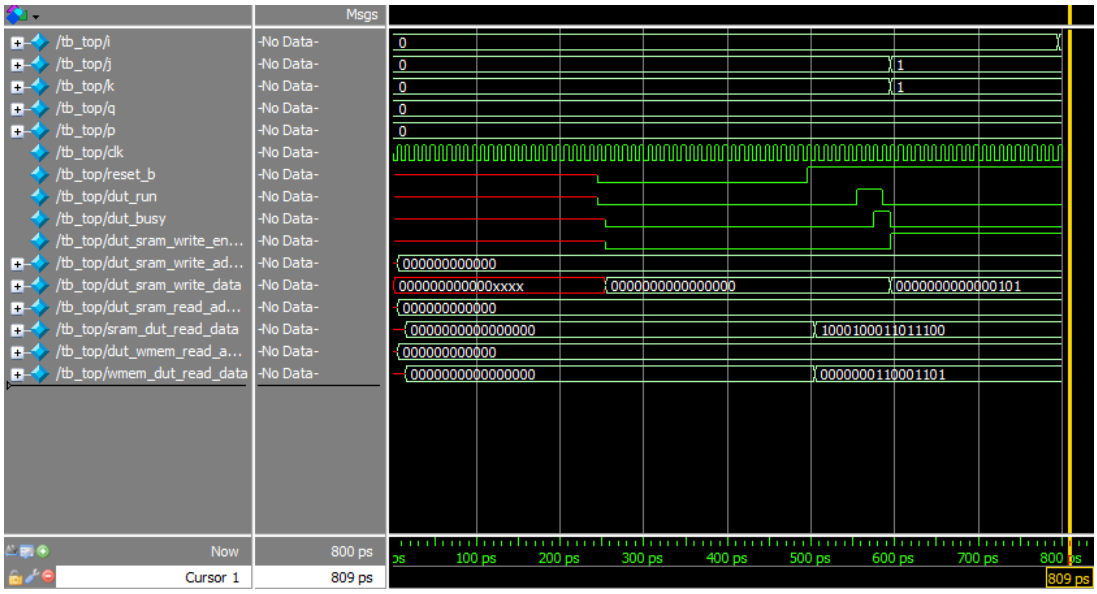

Figure 2 -Timing Diagram

4. Verification

Various approaches were used as verification steps throughout the process. Firstly, the convolution algorithm was made and tested via a simple test bench created that send in two 16-bit values for the input and kernel matrix and observes the output. After the convolution algorithm was verified to work correctly, interfacing of the SRAMs was accomplished. Although there was no specific code utilized to test if the SRAMs were functioning properly and interfacing with the design, ModelSim simulations and a model testbench provided enough responses to determine if there was a proper interface. Simply setting the address of the input SRAM to 0 and observing the reaction was enough to observe the change in data values. A robust testbench was then observed at the end to ensure that all inputs and outputs were correctly flowing through the design.

5. Results Achieved

The design successfully calculates the convolution of the 4x4 input matrix and 3x3 weight matrix in 4 computation cycles. The total area was 680.162006 um^2. With a clock period set to 7, the observed lowest slack was approximately ~0.504. There were no unconstrainted paths and computing performance via the equation:

$$performance = area * cycles * clkperiod$$

a performance of 19044.536168 was obtained.

6. Conclusions

Overall, this was a project that was enjoyable to complete and helpful in better understanding complex hardware designs. Although the design may have been created in a crude manner, it was very helpful to see how complex logic flows through a system. In the future if a return to this project was made, there is room to make the various algorithms and functions used in the hardware more easily scalable while also maintaining minimum timing and area.