Final Project Report

Name: Viswanatha Kasyap Pasumarthy
Unityid: vpasuma
StudentID:. 200310870

Delay (ns to run provided provided example).
Clock period:23.1
cycles":4108

Logic Area: (um^2)

78498.729

Memory: N/A

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

$1.342*10^{-10}$

Delay (TA provided example. TA to complete)

1/(delay.area) (TA)

Abstract

The goal of this project was to design a module that acts as the g(t) gate for an LSTM module. The module takes two inputs in the form of a 16x16 W_g matrix and 16 16x1 X matrices. It then calculates 16 16x1 matrices by multiplying W_g * X and performs a Tanh interpolation using a given lookup table. This module was designed to saturate at values of x less than -3.984375 and greater than 3.984375.

0. **Abstract**

The goal of this project was to design a module that acts as the g(t) gate for an LSTM module. The module takes two inputs in the form of a 16x16 W_g matrix and 16 16x1 X matrices. It then calculates 16 16x1 matrices by multiplying W_g * X and performs a Tanh interpolation using a given lookup table.

1. **Introduction**

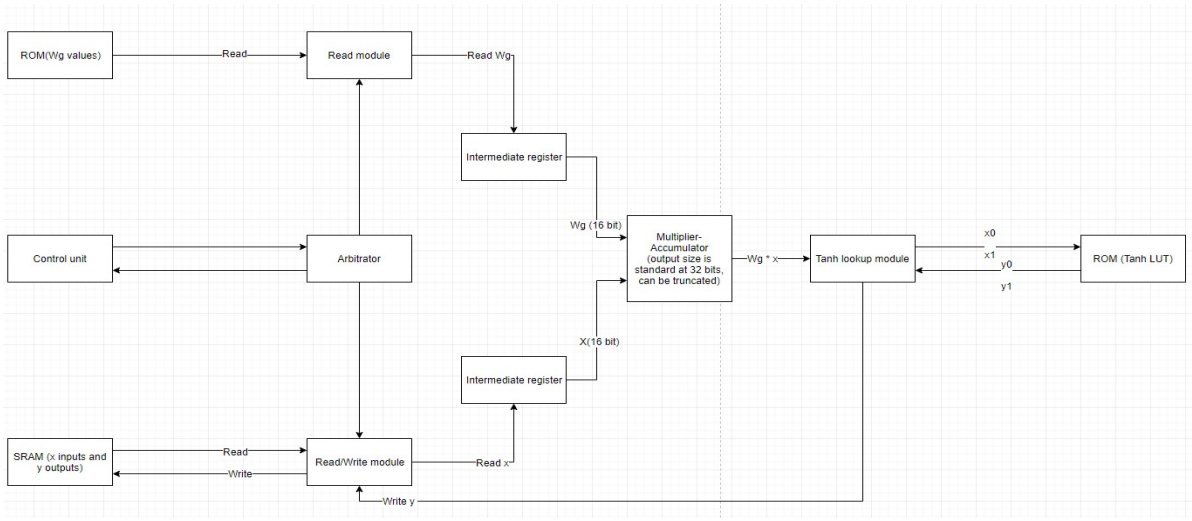
The hardware designed in this project is a combination of a matrix multiplier and a lookup interpolator. Since the hardware was designed with a broader purpose in mind, it can be easily modified to work for other similar purposes. This module was designed to saturate at values of x less than -3.984375 and greater than 3.984375, and with only minor modifications to the RTL code, can be made to work for a broader range of value. The precision of the hardware depends on the precision of the lookup table provided.

For this project, I aimed to achieve consistent results and for three independent datasets was able to achieve an accuracy of close to 96% with an error tolerance of 0.0078. The rest of this report will highlight the development process and the verification process.

2. **Development approach**

I started out with a basic design that could increment the addresses for the various SRAMs correctly. Once I was sure that the addresses were being manipulated correctly to achieve the correct matrix multiplication logic, I set to work on the accumulator logic where it has to accumulate values from 16 cycles and then move onto another set of 16 values. Once the accumulator was done, I moved to the lookup and interpolation logic which is where I spent the most amount of time and effort to synchronize my registers.

A major hurdle in the development process was the decision of choosing the bus width for each register and in the end I reached a balance between accuracy and efficiency. A high level diagram can be found below.



3. **Interfacing with the design**

The interface has four main control signals – 3 inputs and 1 output. Along with usual clock and active low reset inputs, it also accepts a third input named ‘xxx__dut__run’ which is sent by the test bench when it is has loaded all the SRAMs and is ready for the design to start running. As an acknowledgement of the run signal, the design has one output signal named ‘dut__xxx__busy’ which is monitored by the testbench to know if the design is running or idle.

Along with these control signals, there are multiple data register interfaces with the design sending out 12 bit wide addresses to lookup in the multiple SRAMs and the test bench responding with 16 bit wide data. In addition to these read address and data registers, we also have output memory which takes an active high write enable along with the write address and the data to be written, which are 12 and 16 bits wide respectively.

Attached below is a complete list of all the external interface and internal signals for the design along with a brief description

Signal Name	Width	Internal/External
xxx__dut__run	1	External
dut__xxx__busy	1	External
Clk	1	External
Reset_b	1	External
Dut__sram__write_address	12	External
Dut__sram__write_data	16	External
Dut__sram__write_enable	1	External
Dut__sram__read_address	12	External
Sram__dut__read_data	16	External
Dut__gmem__read_address	12	External
Gmem__dut__read_data	16	External
Dut__tanhmem__read_address	12	External
Tanhmem__dut__read_data	16	External
W_inp	16	Internal

X_inp	16	Internal
W_extended	17	Internal
X_extended	17	Internal
Product_store	35	Internal
Interim_product	34	Internal
counter	8	Internal
Accumulator [255:0]	36	Internal
Tanh_1	16	Internal
Tanh_2	16	Internal
Counter1	13	Internal
Counter2	13	Internal
xcount	16	Internal
Counter_tan	5	Internal
Temp_accum	36	Internal
Temp_accum_compl	36	Internal
Temp_accum_val	19	Internal
Tanh_add_1	12	Internal
Tanh_add_2	12	Internal
y	34	Internal
X0	17	Internal
X1	17	Internal
Y0	16	Internal
Y1	16	Internal
x	17	Internal
status	1	Internal
A_width	N/A	Internal
B_width	N/A	Internal
Mult_TC	1	Internal
i	N/A	Internal
X_counter	N/A	Internal

4. **Verification**

The final design was verified with a test bench for accuracy and precision while intermediary designs were tested by observing timing diagrams.

5. **Results**

Accuracy achieved - ~96%
Final cell area – 78498.73 um^2

6. **Conclusions**

This project has helped me learn many of the key aspects of RTL design while also teaching me many of the pitfalls. I believe that given more time I could have optimized the design to achieve both better accuracy and a better design area.