Final Project Report

Name:	Sidharth Mehta
Unityid:	smehta22
StudentID:.	200318890

Delay (ns to run provided provided example).	17582.24
Clock period:	4.28
# cycles":	4108

Logic Area: (um^2)

3159.0160

Memory: N/A

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

1.8004205880344346894400415986613e-8

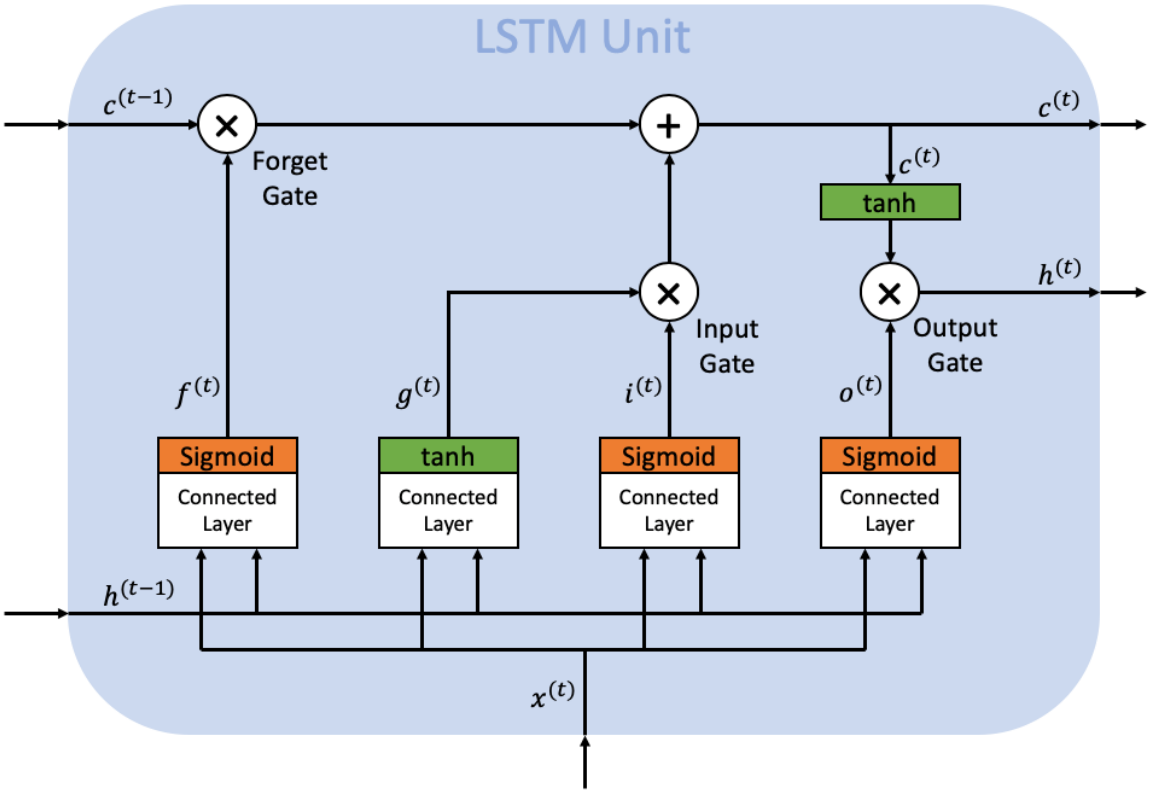
Delay (TA provided example. TA to complete)

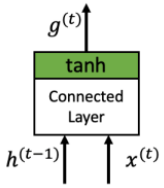
1/(delay.area) (TA)

g(t) gate within LSTM cell

Abstract

The project implements a g(t) gate within an LSTM cell. LSTM stands for Long Short Term Memory and is a type of neural network.





$$g^{(t)} = tanh(W_g x^{(t)} + U_g h^{(t-1)} + b_g)$$

where W_g and U_g are the weight matrix for $x^{(t)}$ and $h^{(t-1)}$, b_g is the bias vector of the connected layer

g(t) gate is a tanh function performed on the weighted sum of inputs. Weighted sum is calculated using matrix multiplication of weight W_g with input X_t.

- W_g is a matrix of size 16*16. Each weight is 16-bit fixed point.
- X_t is a vector of size 16 with each element being 16-bit fixed point.

g(t) gate finds its output using interpolation on weighted sum with help of a lookup table.

The aim of this project was to design g(t) gate while achieving minimal area-delay product.

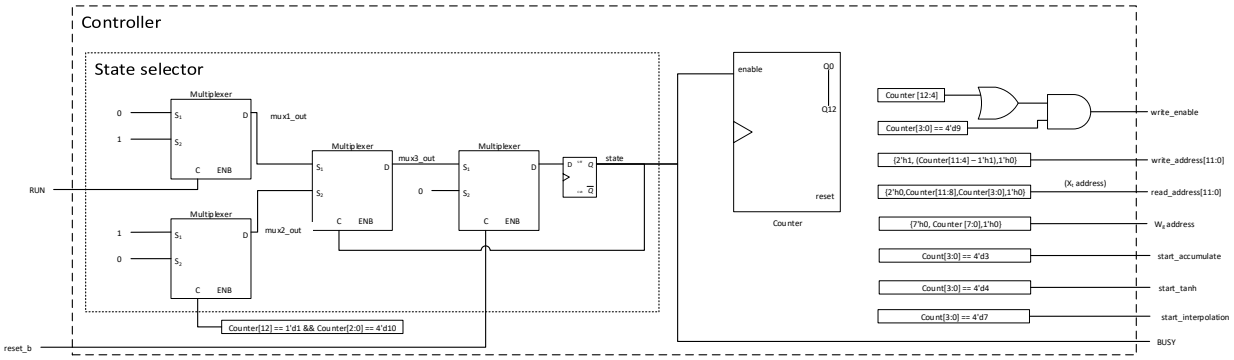
Implementation

1. ECE564MyDesign

ECE564MyDesign has 3 submodules:

a. **Controller:** Used to generate the control signals for modules.
b. **Matrix:** Used to produce matrix multiplication on W_g and X_t.
c. **functionG:** Used to calculate g(t) on output of matrix multiplication.

2. Controller



Controller has 3 submodules:

a. **State selector:** The state machine generates a state signal which is used to control the counter.

1. If reset signal is LOW then the value of state signal becomes LOW (disabled) else its dependent on what the current value of state is, RUN signal and whether the g(t) operation is complete.
2. Initially the state signal is LOW (disabled) and it stays LOW till RUN is asserted.
3. When RUN is set HIGH the state signal becomes HIGH and is now dependent on g(t) operation completion.
4. The state remains HIGH till g(t) operation is complete at which point it becomes LOW and is now dependent on RUN.

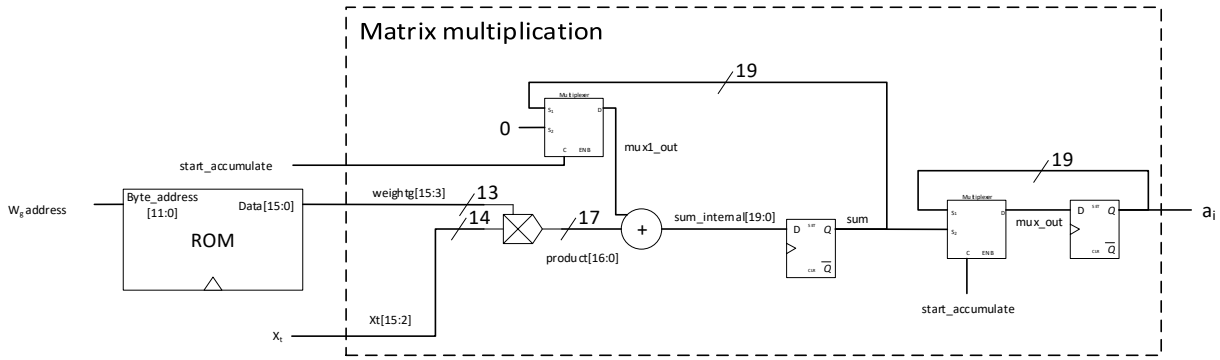
b. **Counter:** The module implements a 13 bit counter which is used to generate all the control signals which are time dependent. Its value depends on state signal. If state signal is HIGH, it increments the count by 1 and if state signal is LOW count is set to 0.

c. **Signal:** Used to generate control signals using value of counter. It is also responsible to update state machine about the status of g(t) conversion.

It generates the following control signals:

a) write_enable
b) write_address
c) read_address
d) Wg_address
e) BUSY
f) start_accumulate
g) start_tanh
h) start_interpolation
i) state_mux – Conversion complete signal

3. Matrix Multiplication



Matrix multiplication has 3 submodules:

a. multiplier: The multiplier takes in Weightg and Xt as inputs and returns their product.

Innovation:

1. In order to reduce the time and size of the multiplier the no of bits fed into the multiplier is reduced as the size of product needed is small.
2. A 3 stage multiplier is used to reduce critical path.

b. adder: The adder is fed product from the multiplier. When start_accumulate is asserted it resets the sum to 0 and starts summing up inputs to it from multiplier at every clock cycle. It also transfers sum to result module at each clock cycle.

Innovation:

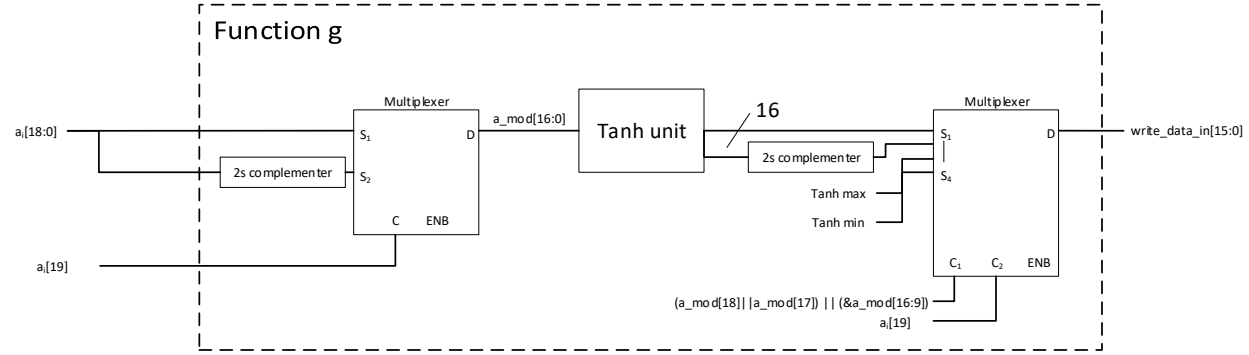
1. There was no need to saturate the sum.

Problems:

1. Need to sign extend the product due to difference in size of sum and product.

c. result: It feeds the value of matrix product to function. When start_accumulate is asserted it propagates the new value of matrix multiplication and maintains its value till start_accumulate is asserted again.

4. Function g



Function g has 6 submodules:

a. **Complement_19bit:** Used to find 2s complement of matrix multiplication outcome. As tanhmem only poses values for positive number therefore for negative numbers we need its magnitude.

   Innovation: Subtraction of number from 2^{19} gives smaller area than performing not operation followed by addition of 1.

b. **mux2_1:** Used to feed magnitude to Tanh function. The sign bit of matrix multiplication decides whether to send number itself or its 2s complement.

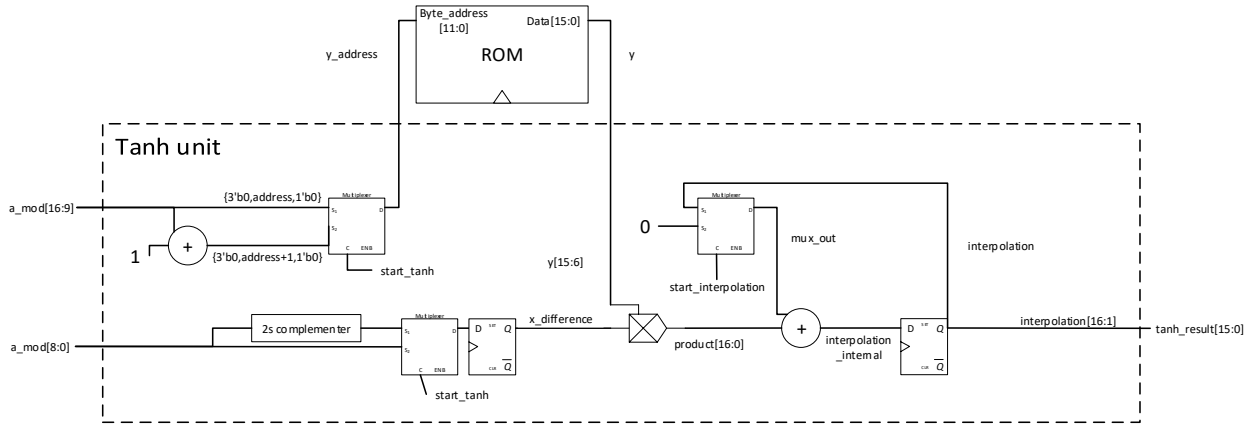
c. **tanh:** Used to calculate tanh.

d. **complement_16 bit**: Used to 2s complement output of Tanh function. It helps to get correct output if input to tanh was a negative number.

e. **c1_logic**: Used to detect if the output of function G is to be saturated. If input of Tanh is greater than 3.984 or less than -3.984 then it needs to be saturated.

f. **mux4_1**: Used to give correct output from function G. Its value depends on the sign on input and whether there is saturation or not.

5. Tanh unit



Tanh has 5 submodules:

a. **Yunit:** Used to generate address to read from in tanhmem

b. **complement_9bit:** Used to calculate x1 – x.
   Innovation:
   1. Instead of calculating x1 – x we just need to take 2s complement of last 10 bits of x.
   2. Instead of calculating x – x0 we just need last 9 bits of x.

c. **Xunit:** Used to select among x – x0 and x1 – x. start_tanh signal is used to select output from this mux.

d. **Tan_multiplier:** gives product of x_difference and y.

   Innovation:
   1. Reduction of bits of input to improve area.
   2. Multiplier is reused for multiplication.

e. **tanh_result:** Performs interpolation on data from multiplier when start_interpolation is asserted.
   Innovation:
   1. Divide logic(shift) is performed by rearranging the wires.

Port List

1. ECE564MyDesign

Signal Name	Type	Size	Function
xxx__dut__run	input	1 bit	RUN signal
dut__xxx__busy	output	1 bit	BUSY signal
clk	input	1 bit	Clock signal
reset_b	input	1 bit	reset signal
dut__sram__write_address	output	12 bit	Address to write at in SRAM
dut__sram__write_data	output	16 bit	Data to write in SRAM
dut__sram__write_enable	output	16 bit	SRAM enable signal
dut__sram__read_address	output	12 bit	Address to read data from in SRAM
sram__dut__read_data	input	16 bit	Data to read from SRAM
dut__gmem__read_address	output	12 bit	Address to read from gmem
gmem__dut__read_data	input	16 bit	Data to read from gmem
dut__tanhmem__read_address	output	12 bit	Address to read data from tanhmem
tanhmem__dut__read_data	input	16 bit	Data to read from tanhmem

2. Controller

Signal Name	Type	Size	Function
clock	input	1 bit	Clock signal
RUN	input	1 bit	RUN signal
reset_b	input	1 bit	Reset signal
write_enable	output	1 bit	Used to enable write to SRAM
write_address	output	12 bit	Address to write at in SRAM
read_address	output	12 bit	Address to read Xt from in SRAM
Wg_address	output	12 bit	Address to read Wg from in ROM
BUSY	output	1 bit	BUSY signal
start_accumulate	output	1 bit	Indicate Matrix module to start sum
start_tanh	output	1 bit	Activate Tanh operation
start_interpolation	output	1 bit	Activate interpolation

3. Matrix Multiplication

Signal Name	Type	Size	Function
clock	input	1 bit	Clock signal
start_accumulate	input	1 bit	Indicate Matrix module to start sum
Weightg	input	16 bit	Value of Wg(i,j)
Xt	input	16 bit	Value of Xt(i)
ai	output	20 bit	Output of matrix multiplication

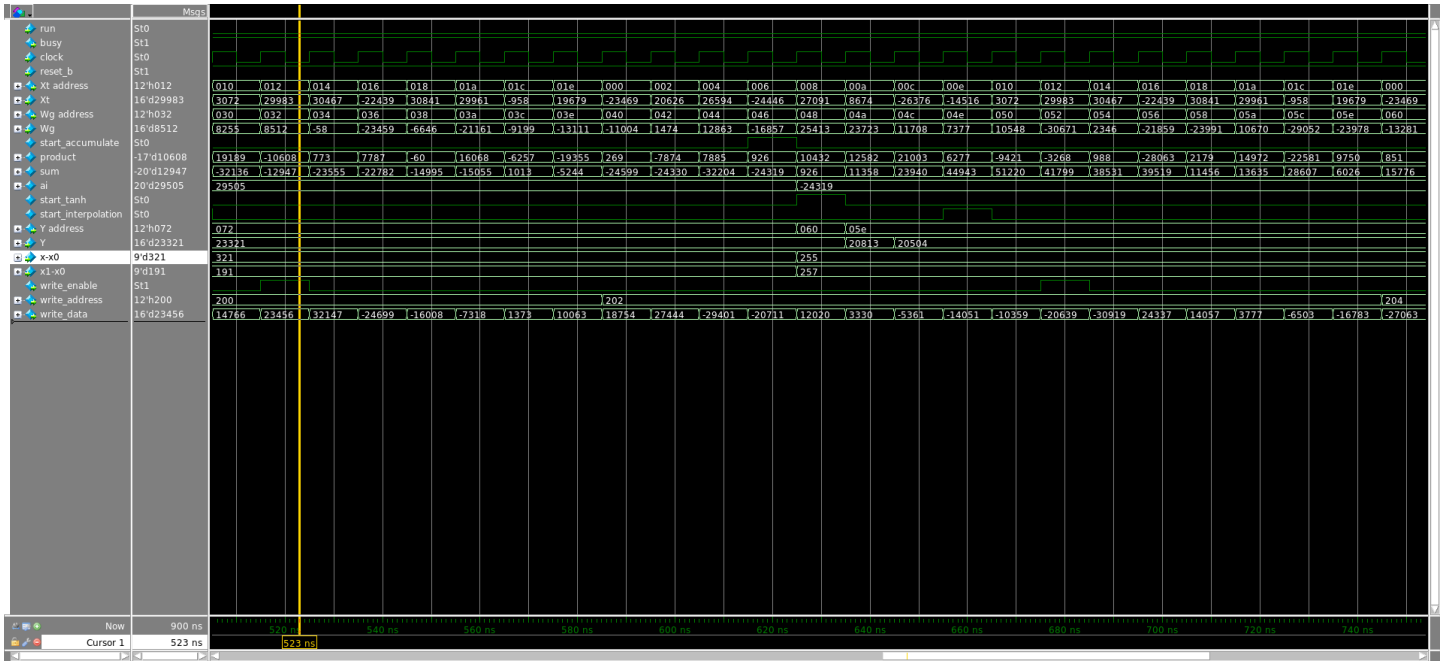
4. Function G

Signal Name	Type	Size	Function
clock	input	1 bit	Clock signal
ai	input	20 bit	Result of matrix multiplication
y	input	16 bit	Data from tanhmem
start_tanh	input	1 bit	Signal to start Tanh module
start_interpolation	input	1 bit	Signal to start Interpolation
y_address	output	12 bit	Address to read data from Tanh lookup table
write_data_in	output	16 bit	Output of g(t)

5. Tanh

Signal Name	Type	Size	Function
a_mod	input	17 bit	Magnitude of matrix multiplication
y	input	16 bit	Data from tanhmem
y_address	output	12 bit	Address to read data from in tanhmem
start_tanh	input	1 bit	Used to start Tanh multiplier
start_interpolation	input	1 bit	Used to start Interpolation on result of multiplier
tanh_result	output	16 bit	result of interpolation

Waveform



Verification

1. Checked if the matrix multiplier worked by manually calculating results for few sets and verifying them with simulation results.
2. Used ece564_project_tb_top.sv file to verify correctness of outputs for g(t).
3. Fixed issues in timing by comparing manual and results in simulation.
4. Tested some of the modules where test fixture was not helpful

Results Achieved

Area:	3159.0160
Clock Period:	4.28
Cycles:	4108
Delay:	17582.24

FILE Listing

Code files

S. No.	File Name	Module Implemented
1.	ECE564MyDesign.v	ECE564MyDesign
2.	controller.v	controller
3.	controller_state_selector.v	state_selector
4.	controller_counter.v	counter
5.	controller_signal.v	signal
6.	Matrix_top.v	matrix
7.	Matrix_multiplier.v	multiplier
8.	Matrix_adder.v	adder
9.	Matrix_result.v	result
10.	function_g_top.v	functionG
11.	twoscomplement_19bit.v	complement_19bit
12.	function_g_mux1.v	mux2_1
13.	Tanh_top.v	tanh
14.	twoscomplement_16bit.v	complement_16bit
15.	function_g_mux2_c1_logic.v	c1_logic
16.	function_g_mux2.v	mux4_1
17.	Tanh_y.v	Yunit
18.	twoscomplement_9bit.v	complement_9bit
19.	Tanh_x.v	Xunit
20.	Tanh_multiplier.v	Tan_multiplier
21.	Tanh_result.v	tanh_result
22.	controller_test.v	controller_test

Synthesis scripts

S. No.	File Name
1.	setup.tcl
2.	read.tcl
3.	Constraints.tcl
4.	CompileAnalyze.tcl
5.	.synopsys_dc.setup