ECE564
ASIC & FPGA Design with Verilog
Final Project

Name: Rhushikesh Anand Prabhune
Unity ID: rprabhu
Student ID: 200321627

Delay (ns to run provided provided example). Clock period: 10.5 ns # cycles: 4105	Logic Area: (um^2) 9330.48205 Memory: N/A	1/(delay.area) (ns^{-1}.um^{-2}) 9.590480E-6
Delay (TA provided example. TA to complete)		1/(delay.area) (TA)

Abstract: A fixed point multiply and accumulate was designed which was then given as an input to a tanh interpolation unit to get the output. An area of 9330.48205 um^2 was obtained with a clock period of 10.5 ns. Input was taken from SRAM for X values. The Wg values were taken from a ROM. Matrix multiplication was then carried out 16 times by repeating the values of X and Wg accordingly. The tanh values were stored in ROM which were used for interpolation. If the values were above a particular limit or below a particular negative limit, a fixed value is used and there is no need for interpolation in such a case. This can be done because the tanh function becomes flat after a particular limit.

Fixed Point Matrix Multiplication and Accumulation and Tanh Interpolation

Rhushikesh Anand Prabhune

Introduction

Hardware designed:

- Hardware for a fixed-point multiplication which takes in 16-bit values of the variables was designed. Multiplication and accumulation is carried out which then proceeds to calculate the tanh values of the matrix multiplication outputs.
- The tanh of the matrix multiplication outputs is carried out through interpolation of tanh data stored in a ROM. If the output is greater than or less than a particular threshold, a constant value is considered since the tanh function saturates after a particular value. The graph of tanh is given below for reference.
- The design reads the matrix values from an SRAM and a ROM. The calculation starts when the busy flag is set high.
- The output values are stored in a SRAM after which the busy flag is set to low.

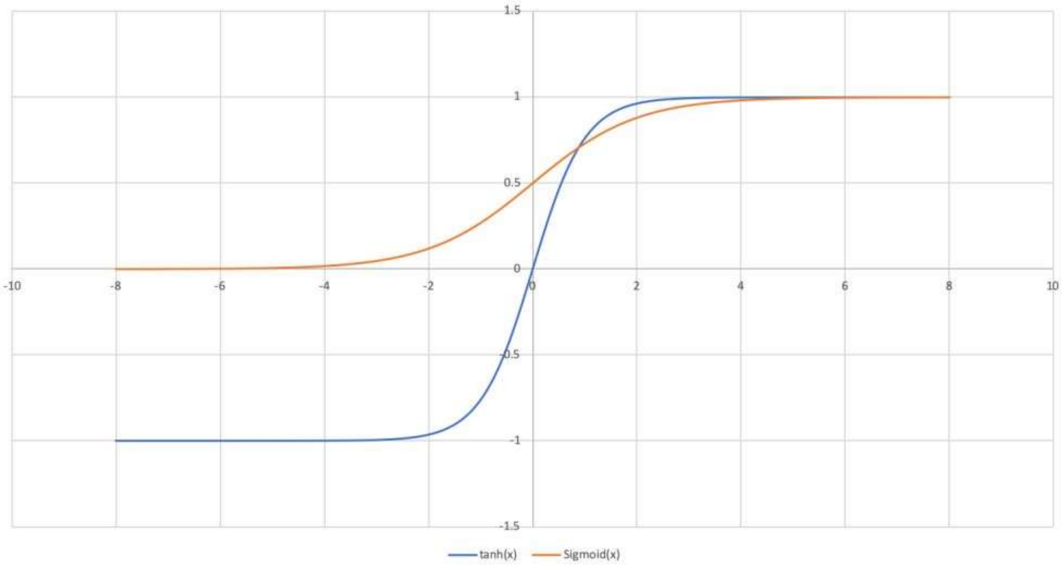


Figure 1: tanh graph

Summary of Key Innovations:

- While doing fixed-point multiplication the operands are sign extended to meet the number of digits the multiplication will yield. In this project, a multiplication of two 16-bit operands will yield a 32-bit result. Therefore, the operands are sign extended with 16-bits to get two 32-bit operands.
- The products will be accumulated together 16 times. Hence, taking the upper limit (the products are less than one, upper limit 0.99), the accumulator can reach just below 16. Hence, four extra bits are allocated to the accumulator to accommodate the upper limit. The product is sign extended with 4-bits.

Summary of results achieved:

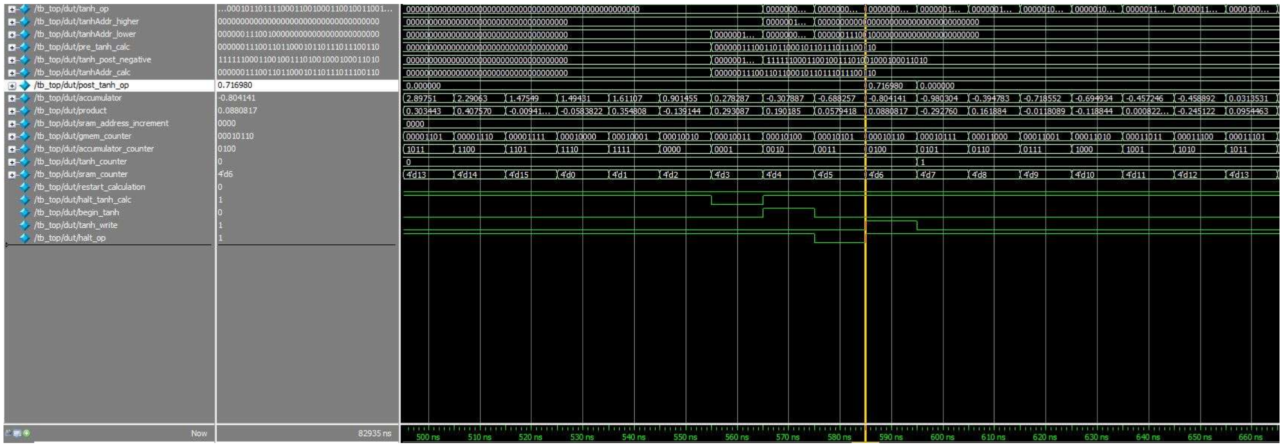
- Area: 9330.48205 um^2
- Number of Cycles: 4105
- Clock Period: 10.5 ns
- Performance (Area * #Cycles * Clock Period): 402167102.56 (um^2)(ns)

Micro-Architecture

- The SRAM address which is used to fetch the X values is controlled by a 4-bit counter ‘sram_counter’ which resets after every 16 cycles. At the same time, there is a ‘sram_address_increment’ which increments after 256 cycles. The SRAM address is controlled by ‘sram_counter’.
- The address obtained when this counter becomes zero is dependent on the ‘sram_address_increment’ counter. The ‘sram_address_increment’ counter is controlled by the ‘gmem_counter’ which resets itself after 256 cycles since it is an 8-bit counter.
- The useful value of the accumulator occurs after every 16 cycles. For this purpose a 4-bit counter ‘accumulator_counter’ is used. When the ‘accumulator_counter’ becomes zero, the value of accumulator is registered in a flop which can then proceed to the tanh calculation. Also for all the other times, the products are accumulated in the accumulator.
- The tanh calculation first checks if the output from accumulator is positive or negative. If negative, it takes the 2’s complement of the output for the interpolation process. If positive, no change is made to the output.
- If the value after the above step is above 12’h1ff, the value taken for tanh address is 12’h1ff.
- If the above condition is not met, an interpolation is done using the tanh data in SRAM to get the final output which is then written into the SRAM
- The write_enable flag is set high while writing to SRAM and low while reading from SRAM.

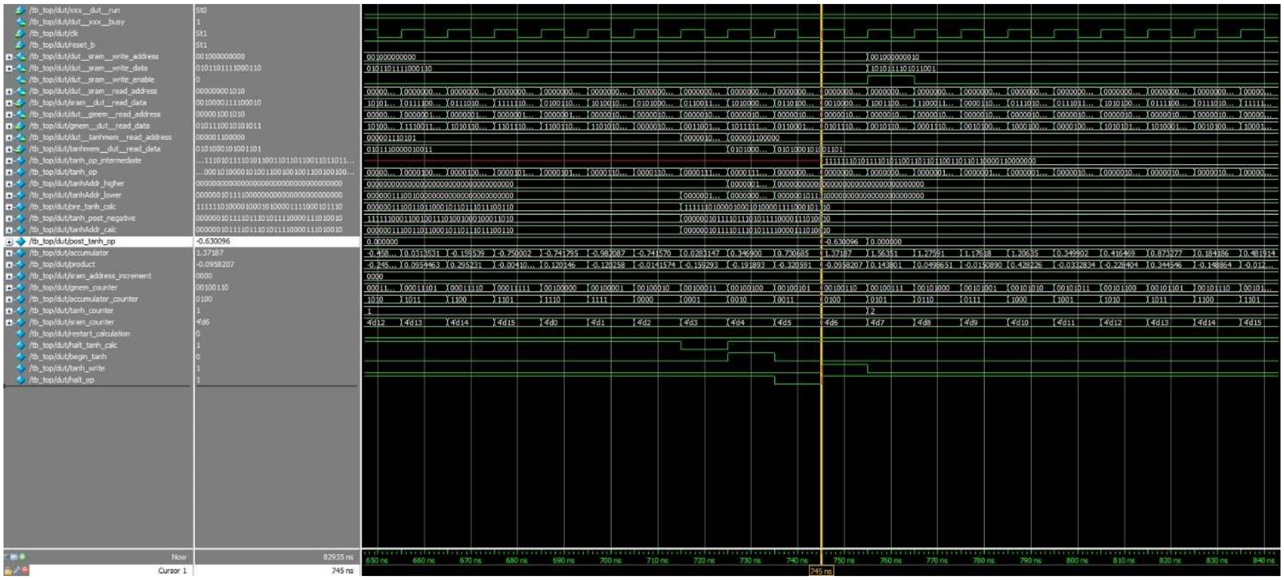
Verification

- ➢ The final output was compared with the MATLAB tanh outputs to guarantee the correct results using the given test bench.
- ➢ Intermediate product and accumulate results were also verified by manually checking if the outputs were correct.
- ➢ The waveforms were observed carefully to check the output after a certain number of cycles.
- ➢ For example, the accumulator result was checked after every 16 cycles and cross-checked with the correctly calculated results.



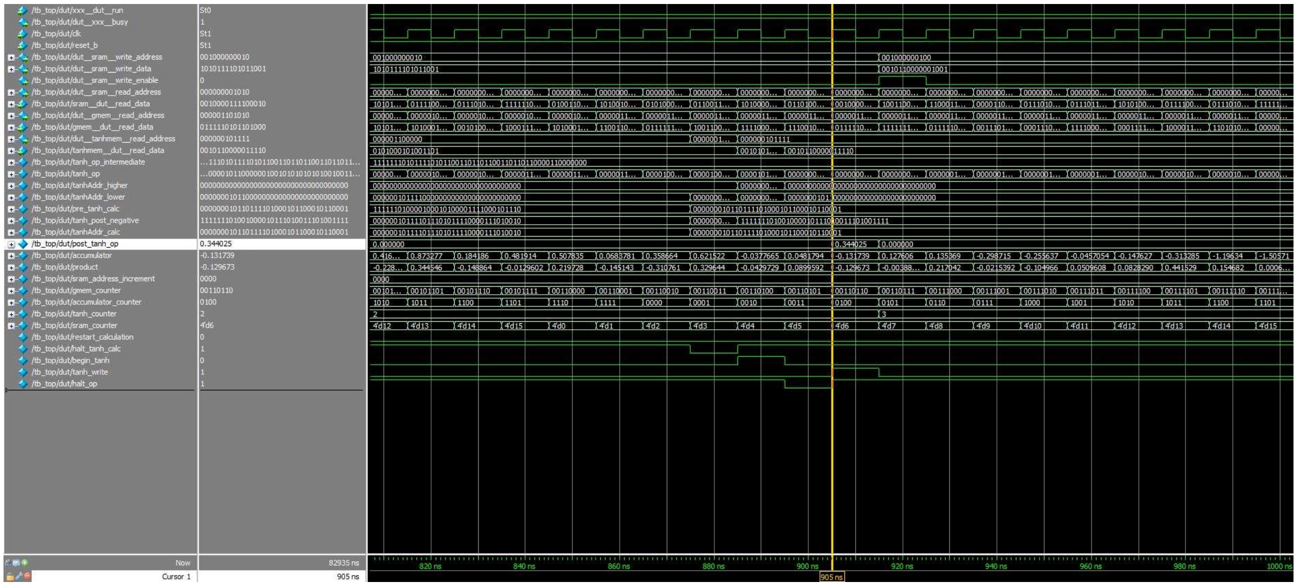
```
/tb_top/dut/tanh_op                    ...000101101111000110010001100100110 01...
/tb_top/dut/tanhAddr_higher            00000000000000000000000000000000000
/tb_top/dut/tanhAddr_lower             00000011100100000000000000000000000
/tb_top/dut/pre_tanh_calc              00000011100110110001011011101110 0110
/tb_top/dut/tanh_post_negative         11111100011001001110100100010001 1010
/tb_top/dut/tanhAddr_calc              00000011100110110001011011101110 0110
/tb_top/dut/post_tanh_op               0.716980
/tb_top/dut/accumulator                -0.804141
/tb_top/dut/product                    0.0880817
/tb_top/dut/sram_address_increment     0000
/tb_top/dut/gmem_counter               00010110
/tb_top/dut/accumulator_counter        0100
/tb_top/dut/tanh_counter               0
/tb_top/dut/sram_counter               4'd6
/tb_top/dut/restart_calculation        0
/tb_top/dut/halt_tanh_calc             1
/tb_top/dut/begin_tanh                 0
/tb_top/dut/tanh_write                 1
/tb_top/dut/halt_op                    1
Now                                    82935 ns
```

Figure 2: First output value from tanh



```
/tb_top/dut/xxx__dut__run              St0
/tb_top/dut/dut__xxx__busy             1
/tb_top/dut/clk                        St1
/tb_top/dut/reset_b                    St1
/tb_top/dut/dut__sram__write_address   001000000000
/tb_top/dut/dut__sram__write_data      0101101111000110
/tb_top/dut/dut__sram__write_enable    0
/tb_top/dut/dut__sram__read_address    000000001010
/tb_top/dut/sram__dut__read_data       0010000111100010
/tb_top/dut/dut__gmem__read_address    000001001010
/tb_top/dut/gmem__dut__read_data       0101110010101011
/tb_top/dut/dut__tanhmem__read_address 000001100000
/tb_top/dut/tanhmem__dut__read_data    0101000101001101
/tb_top/dut/tanh_op_intermediate       ...1110101111010110011011011001101 1011...
/tb_top/dut/tanh_op                    ...0001010000101001100100100110010 0100...
/tb_top/dut/tanhAddr_higher            00000000000000000000000000000000000
/tb_top/dut/tanhAddr_lower             00000010111100000000000000000000000
/tb_top/dut/pre_tanh_calc              11111101000010001010000111100010 1110
/tb_top/dut/tanh_post_negative         00000010111101110101111000011101 0010
/tb_top/dut/tanhAddr_calc              00000010111101110101111000011101 0010
/tb_top/dut/post_tanh_op               -0.630096
/tb_top/dut/accumulator                1.37187
/tb_top/dut/product                    -0.0958207
/tb_top/dut/sram_address_increment     0000
/tb_top/dut/gmem_counter               00100110
/tb_top/dut/accumulator_counter        0100
/tb_top/dut/tanh_counter               1
/tb_top/dut/sram_counter               4'd6
/tb_top/dut/restart_calculation        0
/tb_top/dut/halt_tanh_calc             1
/tb_top/dut/begin_tanh                 0
/tb_top/dut/tanh_write                 1
/tb_top/dut/halt_op                    1
Now                                    82935 ns
Cursor 1                               745 ns
```

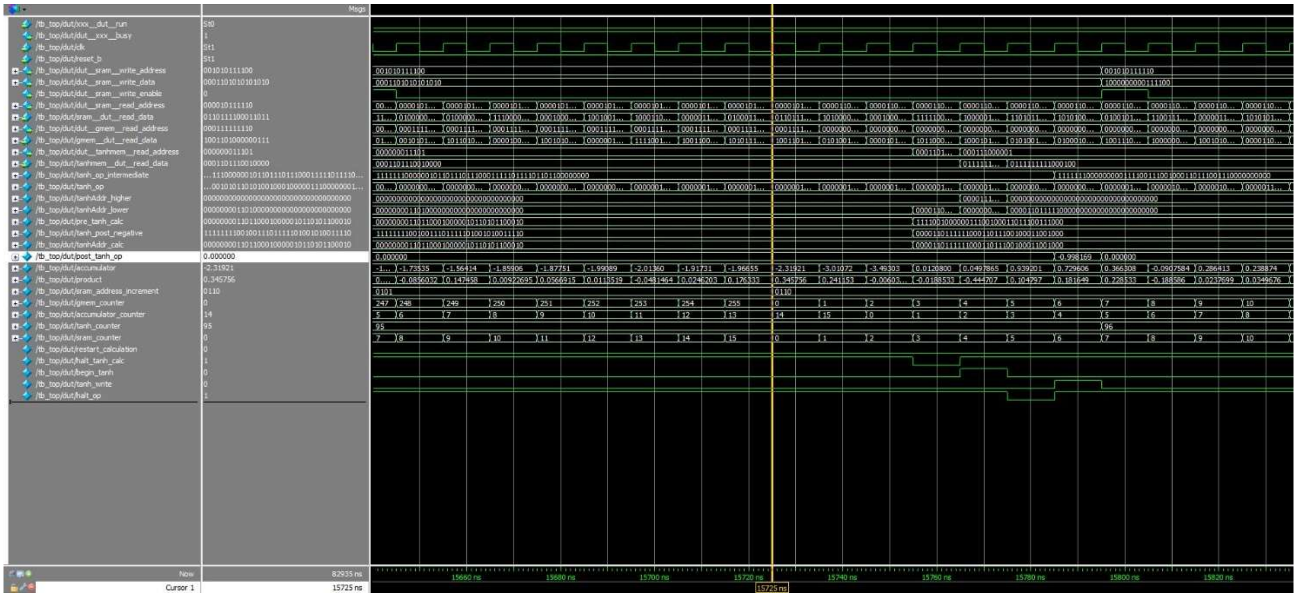
Figure 3: Second output value from tanh



```
/tb_top/dut/xxx__dut__run                St0
/tb_top/dut/dut__xxx__busy               1
/tb_top/dut/clk                          St1
/tb_top/dut/reset_b                      St1
/tb_top/dut/dut__sram__write_address     001000000010
/tb_top/dut/dut__sram__write_data        1010111101011001
/tb_top/dut/dut__sram__write_enable      0
/tb_top/dut/dut__sram__read_address      000000001010
/tb_top/dut/sram__dut__read_data         0010000111100010
/tb_top/dut/dut__gmem__read_address      000001101010
/tb_top/dut/gmem__dut__read_data         0111110101101000
/tb_top/dut/dut__tanhmem__read_address   000000101111
/tb_top/dut/tanhmem__dut__read_data      0010110000011110
/tb_top/dut/tanh_op_intermediate         ...1110101111010110011011011001101101 1...
/tb_top/dut/tanh_op                      ...0000101100000010010101010101001001 1...
/tb_top/dut/tanhAddr_higher              00000000000000000000000000000000
/tb_top/dut/tanhAddr_lower               00000001011000000000000000000000
/tb_top/dut/pre_tanh_calc                00000001011011110100010110001011 0001
/tb_top/dut/tanh_post_negative           11111110100100001011101001110100 1111
/tb_top/dut/tanhAddr_calc                00000001011011110100010110001011 0001
/tb_top/dut/post_tanh_op                 0.344025
/tb_top/dut/accumulator                  -0.131739
/tb_top/dut/product                      -0.129673
/tb_top/dut/sram_address_increment       0000
/tb_top/dut/gmem_counter                 00110110
/tb_top/dut/accumulator_counter          0100
/tb_top/dut/tanh_counter                 2
/tb_top/dut/sram_counter                 4'd6
/tb_top/dut/restart_calculation          0
/tb_top/dut/halt_tanh_calc               1
/tb_top/dut/begin_tanh                   0
/tb_top/dut/tanh_write                   1
/tb_top/dut/halt_op                      1

Now        82935 ns
Cursor 1   905 ns
```

Figure 4: Third output value from tanh



```
/tb_top/dut/xxx__dut__run                St0
/tb_top/dut/dut__xxx__busy               1
/tb_top/dut/clk                          St1
/tb_top/dut/reset_b                      St1
/tb_top/dut/dut__sram__write_address     001010111100
/tb_top/dut/dut__sram__write_data        0001101010101010
/tb_top/dut/dut__sram__write_enable      0
/tb_top/dut/dut__sram__read_address      000010111110
/tb_top/dut/sram__dut__read_data         0110111100011011
/tb_top/dut/dut__gmem__read_address      000111111110
/tb_top/dut/gmem__dut__read_data         1001101000000111
/tb_top/dut/dut__tanhmem__read_address   000000011101
/tb_top/dut/tanhmem__dut__read_data      0001101110010000
/tb_top/dut/tanh_op_intermediate         ...1110000001011011101110001111011110...
/tb_top/dut/tanh_op                      ...0010101101010010001000001110000000 1...
/tb_top/dut/tanhAddr_higher              00000000000000000000000000000000
/tb_top/dut/tanhAddr_lower               00000000110100000000000000000000
/tb_top/dut/pre_tanh_calc                00000000110110001000001011010110 0010
/tb_top/dut/tanh_post_negative           11111111001001110111110100101001 1110
/tb_top/dut/tanhAddr_calc                00000000110110001000001011010110 0010
/tb_top/dut/post_tanh_op                 0.000000
/tb_top/dut/accumulator                  -2.31921
/tb_top/dut/product                      0.345756
/tb_top/dut/sram_address_increment       0110
/tb_top/dut/gmem_counter                 0
/tb_top/dut/accumulator_counter          14
/tb_top/dut/tanh_counter                 95
/tb_top/dut/sram_counter                 0
/tb_top/dut/restart_calculation          0
/tb_top/dut/halt_tanh_calc               1
/tb_top/dut/begin_tanh                   0
/tb_top/dut/tanh_write                   0
/tb_top/dut/halt_op                      1

Now        82935 ns
Cursor 1   15725 ns
```

Figure 5: gmem_counter and sram_counter reset

Results Achieved

Synopsys 2017 Design Compiler was used to synthesize the Verilog RTL and the 45nm Nandgate OpenCell Library was used for this purpose.
Final Area: 9330.48205 um^2
Clock Period: 10.5 ns

- Medium Compile effort
- Clock Skew: 0.05 ns
- Setup Violation Check: 0.0043 (Slack MET)
- Hold Violation Check: 0.0217 (Slack MET)
- Hold fixed Setup Violation Check: (Slack MET)

Conclusions

The above project is a gate of a Long Short Term Memory (LSTM) cell. LSTM is an artificial recurrent neural network (RNN) architecture. There are many applications of such an architecture in the present world which covers robotics, text generation, speech generation etc. RNN’s not only takes the input but also allows some information to be passed on so that along with the input, it can be used to generate the output. This has many uses as described above. Thus, with the increasing importance of neural networks in the present world, designing architectures which support such technologies is becoming increasingly important.