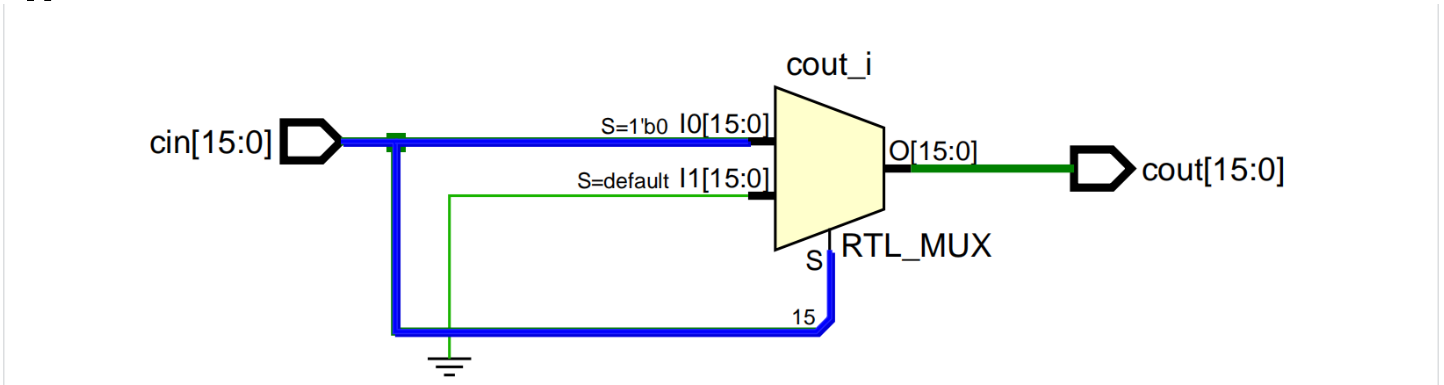
Srijan Gupta - 2020CS50444, Nischay Diwan - 2020CS50433
COL215P: Digital Logic & System Design
15 September 2022

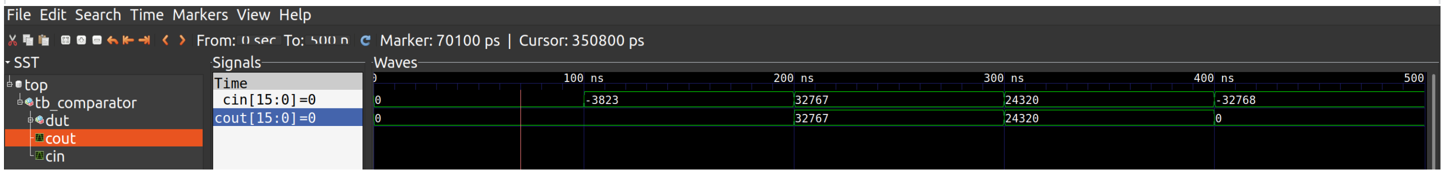
3 Layer MLP

Components

Comparator

Applies the ReLU activation function.

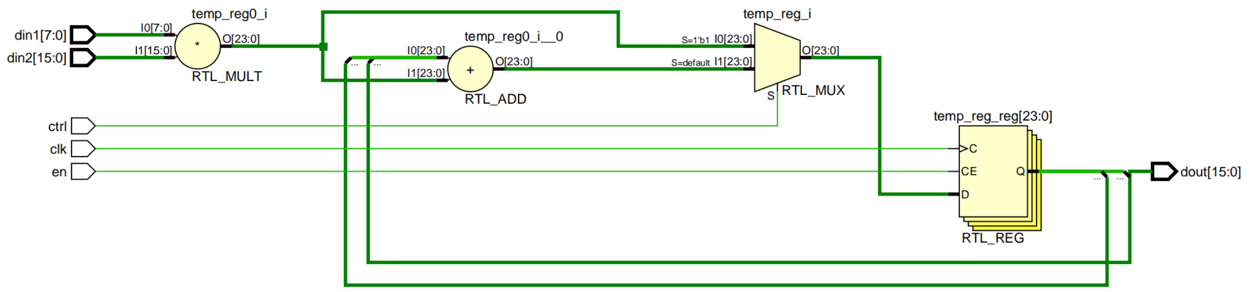


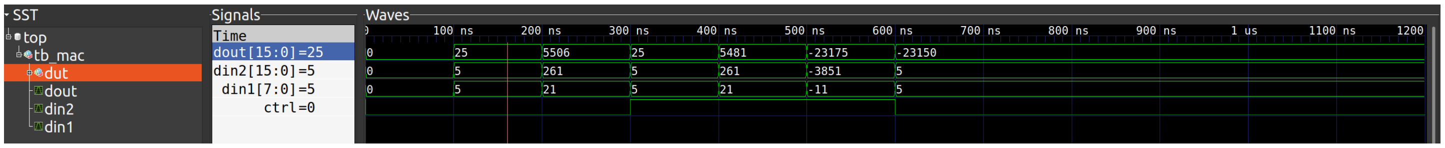

File Edit Search Time Markers View Help
From: 0 sec To: 500 n
Marker: 70100 ps | Cursor: 350800 ps
SST
top
tb_comparator
dut
cout
cin
Signals
Waves

```
Time          100 ns      200 ns      300 ns      400 ns      500
cin[15:0]=0   0   -3823   32767       24320       -32768
cout[15:0]=0  0           32767       24320       0
```

MAC (Multiply-Accumulate Control)

Performs both multiplication and addition simultaneously.
Note: Enable signal and clk signal added to this component (changes from previous stage)

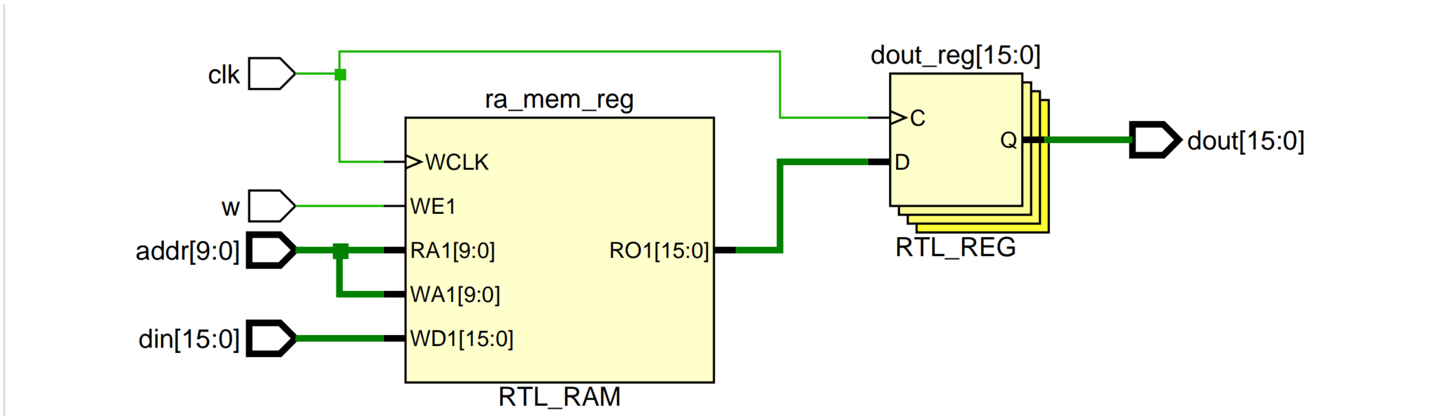


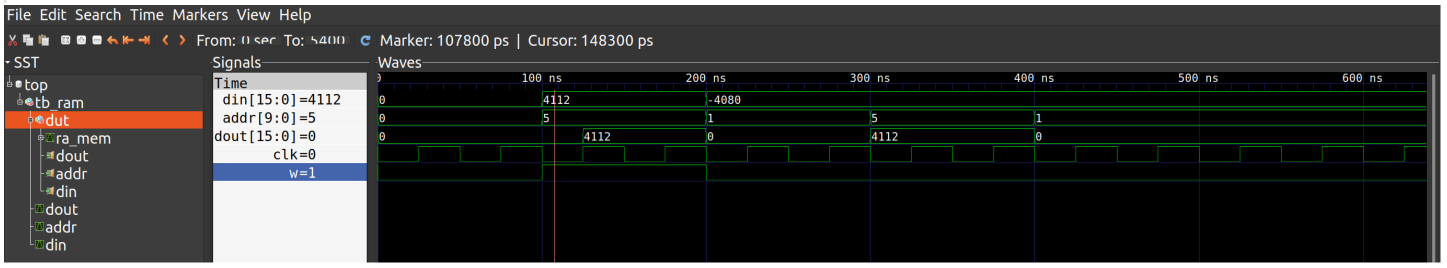


```
Time            100 ns  200 ns  300 ns  400 ns  500 ns  600 ns  700 ns  800 ns  900 ns  1 us  1100 ns  1200
dout[15:0]=25   0   25   5506   25   5481   -23175   -23150
din2[15:0]=5    0   5    261    5    261    -3851    5
din1[7:0]=5     0   5    21     5    21     -11      5
ctrl=0
```

RAM (Random Access Memory)

Memory of size 2kB with 16-bit words with both synchronous reads and synchronous writes. (Clocked read in this module - change from previous stage)

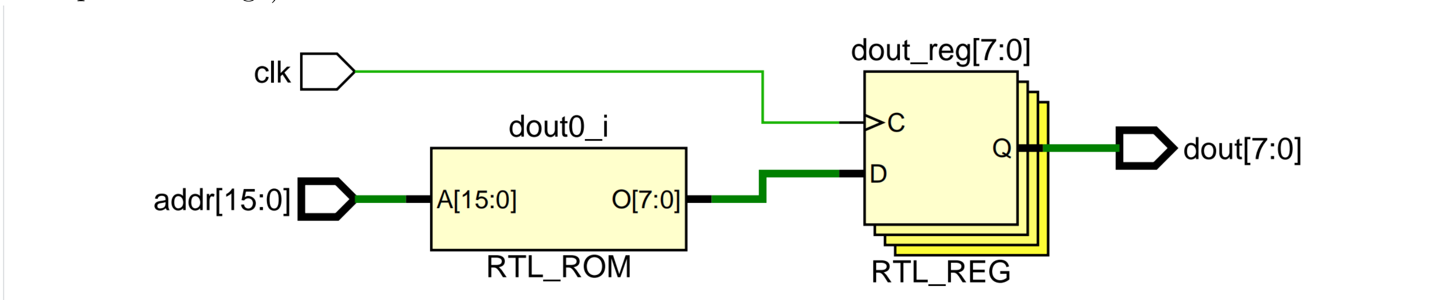


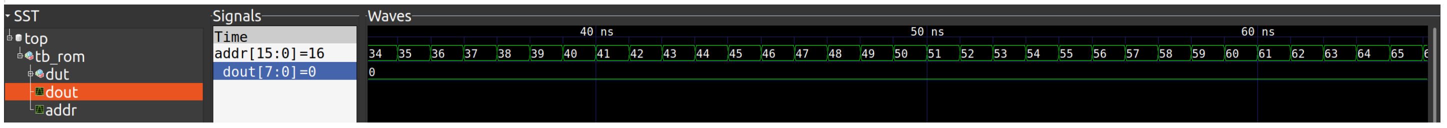

File Edit Search Time Markers View Help
From: 0 sec To: 5400
Marker: 107800 ps | Cursor: 148300 ps
SST
top
tb_ram
dut
ra_mem
dout
addr
din
dout
addr
din
Signals
Waves

```
Time
 din[15:0]=4112
 addr[9:0]=5
dout[15:0]=0
       clk=0
         w=1
```

ROM (Read-Only Memory)

Memory of size 64kB with 8-bit words with synchronous reads. (Clocked read in this module - change from previous stage)

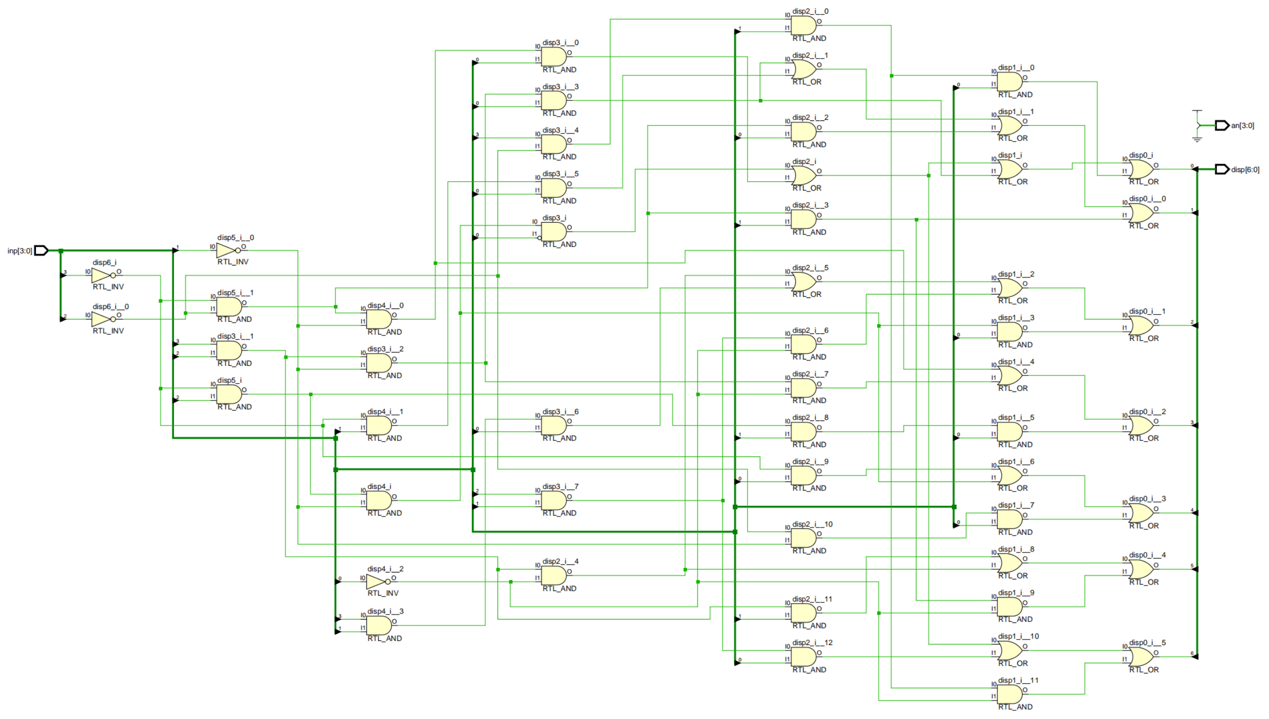


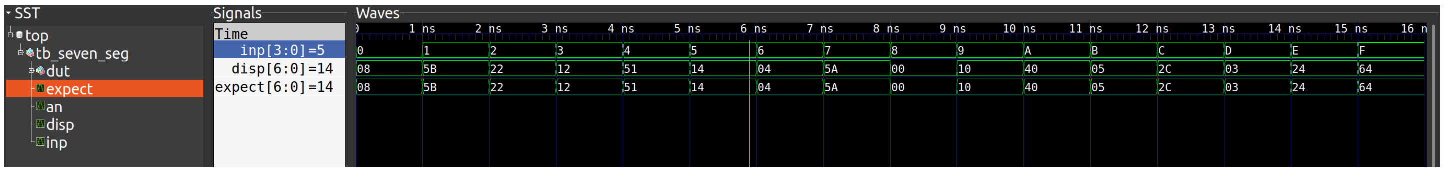


```
Time
addr[15:0]=16
 dout[7:0]=0
```

Seven Segment Display

Takes a hexadecimal input and outputs the state of the segments (on/off) to display the corresponding digit.

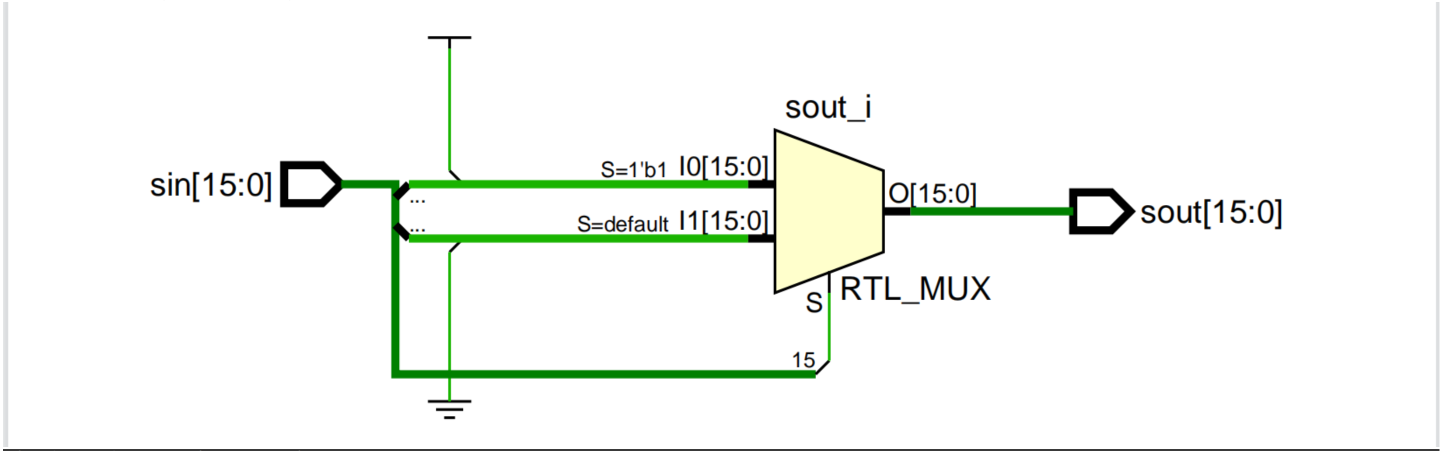


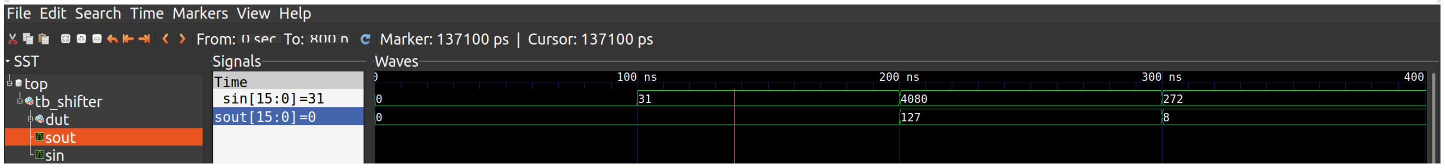


```
SST                Signals           Waves
top                Time              0     1 ns  2 ns  3 ns  4 ns  5 ns  6 ns  7 ns  8 ns  9 ns  10 ns 11 ns 12 ns 13 ns 14 ns 15 ns 16 n
 tb_seven_seg        inp[3:0]=5      0     1     2     3     4     5     6     7     8     9     A     B     C     D     E     F
  dut              disp[6:0]=14      08    5B    22    12    51    14    04    5A    00    10    40    05    2C    03    24    64
   expect        expect[6:0]=14      08    5B    22    12    51    14    04    5A    00    10    40    05    2C    03    24    64
   an
   disp
   inp
```

Shifter

Performs signed integer division by 32.

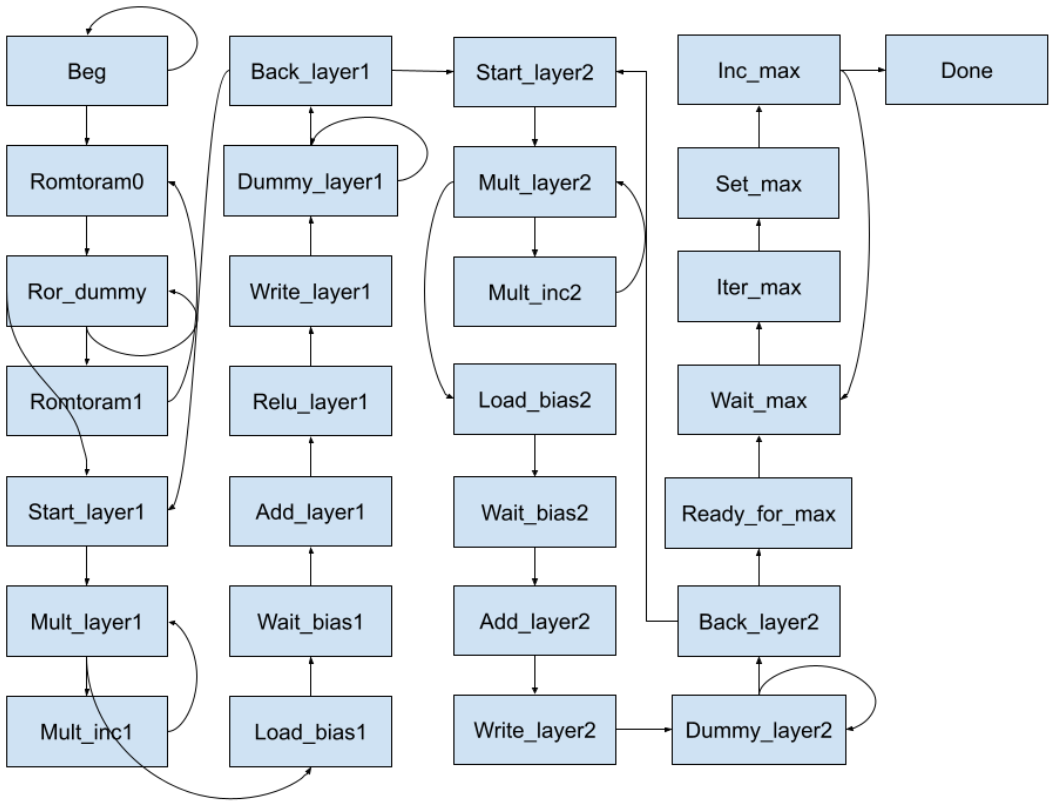




```
File  Edit  Search  Time  Markers  View  Help
From: 0 sec  To: 800 n   Marker: 137100 ps | Cursor: 137100 ps
SST               Signals          Waves
top               Time             0              100 ns              200 ns              300 ns              400
 tb_shifter        sin[15:0]=31    0              31                  4080                272
  dut             sout[15:0]=0     0                                  127                 8
   sout
   sin
```

Working of FSM

States - beg, romtoram0, ror_dummy, romtoram1, start_layer1, mult_layer1, mult_inc1, load_bias1, wait_bias1, add_layer1, relu_layer1, write_layer1, dummy_layer1, back_layer1, start_layer2, mult_layer2, mult_inc2, load_bias2, wait_bias2, add_layer2, write_layer2, dummy_layer2, back_layer2, ready_for_max, wait_max, iter_max, set_max, inc_max, done

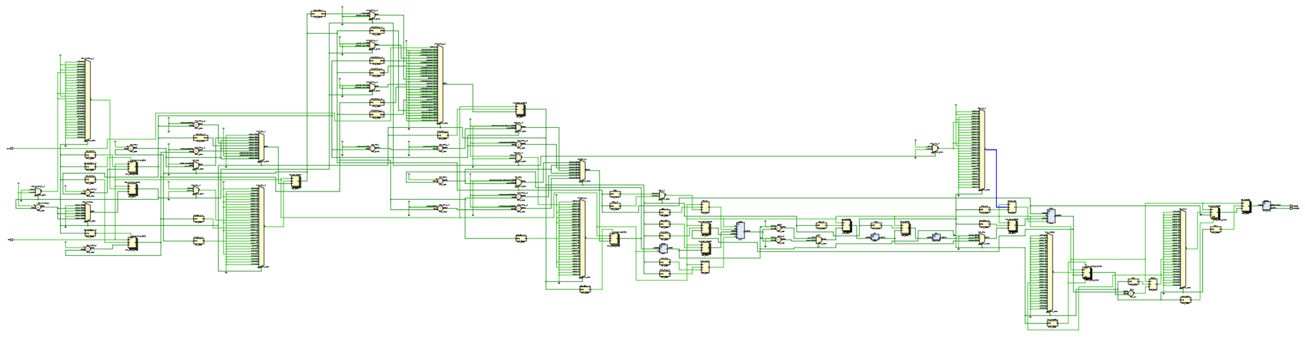


beg: Initialization of addresses, and enable signals take place in this state, and waits for start signal that is controlled by the start button, if start signal is '1', then go to `romtoram0`, else stay in `beg`.
romtoram0: RAM write is enabled and unsigned ROM output is sent to RAM. Go to `ror_dummy`
ror_dummy: Spend 4 cycles to write into the RAM. If the RAM address is 783, go to `start_layer1` else go to `romtoram1`.
romtoram1: Dummy state to allow synchronous reading, go to `romtoram0`.
start_layer1: Increment iter_layer1, disable MAC and go to `mult_layer1`.
mult_layer1: Load MAC inputs and control. If this column's multiplication is done, go to `load_bias1`, else go to `mult_inc1`.
mult_inc1: Dummy state to allow for synchronous read and multiplication. Go to `mult_layer1`.
load_bias1: Load the address of the bias. Go to `wait_bias1`.
wait_bias1: Dummy state to allow for synchronous read. Go to `add_layer1`.
add_layer1: Add the bias to the MAC output and send to the shifter. Go to `relu_layer1`.
relu_layer1: Send the shifted output to the comparator. Go to `write_layer1`.
write_layer1: Enable RAM write and send the activated output to the RAM. Go to `dummy_layer1`.
dummy_layer1: Spend 4 cycles to write into the RAM. Go to `back_layer1`.
back_layer1: If input layer computation is finished, i. e. go to `start_layer2`, else increment iter_layer1 and go to `start_layer1`. Initialize addresses for corresponding next state as well.
start_layer2: Increment iter_layer2, disable MAC and go to `mult_layer2`.
mult_layer2: Load MAC inputs and control. If this column's multiplication is done, go to `load_bias2`, else go to `mult_inc2`.
mult_inc2: Dummy state to allow for synchronous read and multiplication. Go to `mult_layer2`.
load_bias2: Load the address of the bias. Go to `wait_bias2`.
wait_bias2: Dummy state to allow for synchronous read. Go to `add_layer2`.
add_layer2: Add the bias to the MAC output and send to the shifter. Go to `write_layer2`.
write_layer2: Enable RAM write and send the shifted output to the RAM. Go to `dummy_layer2`.
dummy_layer1: Spend 4 cycles to write into the RAM. Go to `back_layer2`.
back_layer2: If hidden layer computation is finished, i. e. iter_layer2 is 10 go to `ready_for_max`, else

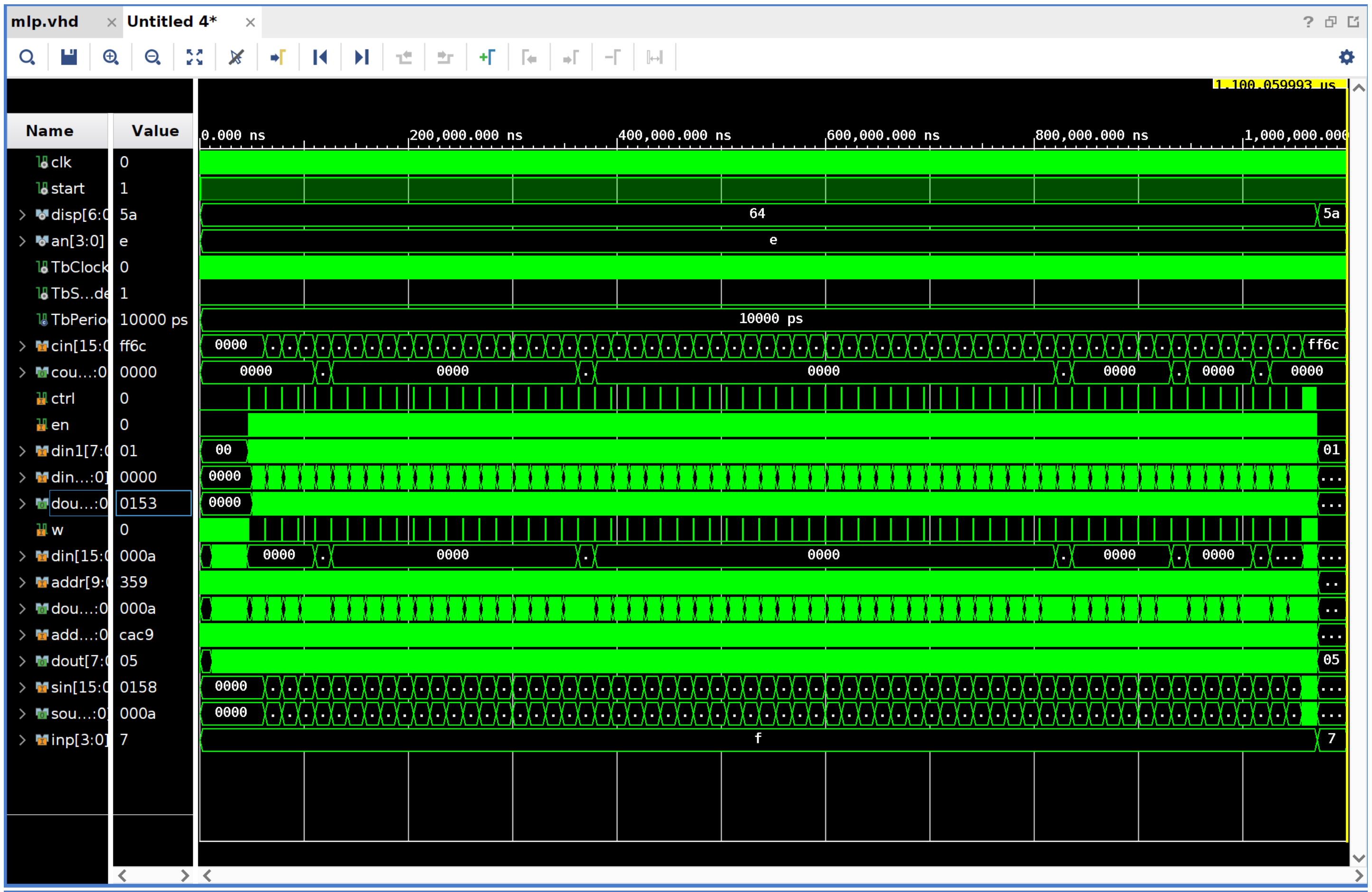
increment iter_layer2 and go to start_layer2. Initialize addresses for corresponding next state as well.
ready_for_max: Set RAM address to start of the output layer. Go to `wait_max`.
wait_max: Dummy state to allow for synchronous read. Go to `iter_max`.
iter_max: If the value of the current output is greater than the current maximum weight set the signal gt to 1 else set gt to 0. Go to `set_max`.
set_max: If the signal gt is 1, update max index and weight else do nothing. Go to `inc_max`.
inc_max: If the last element of the output layer is reached go to `done` else increment the ram address and go to `wait_max`.
done: The final state, seven segment input is set, the circuit then stays in this state infinitely.

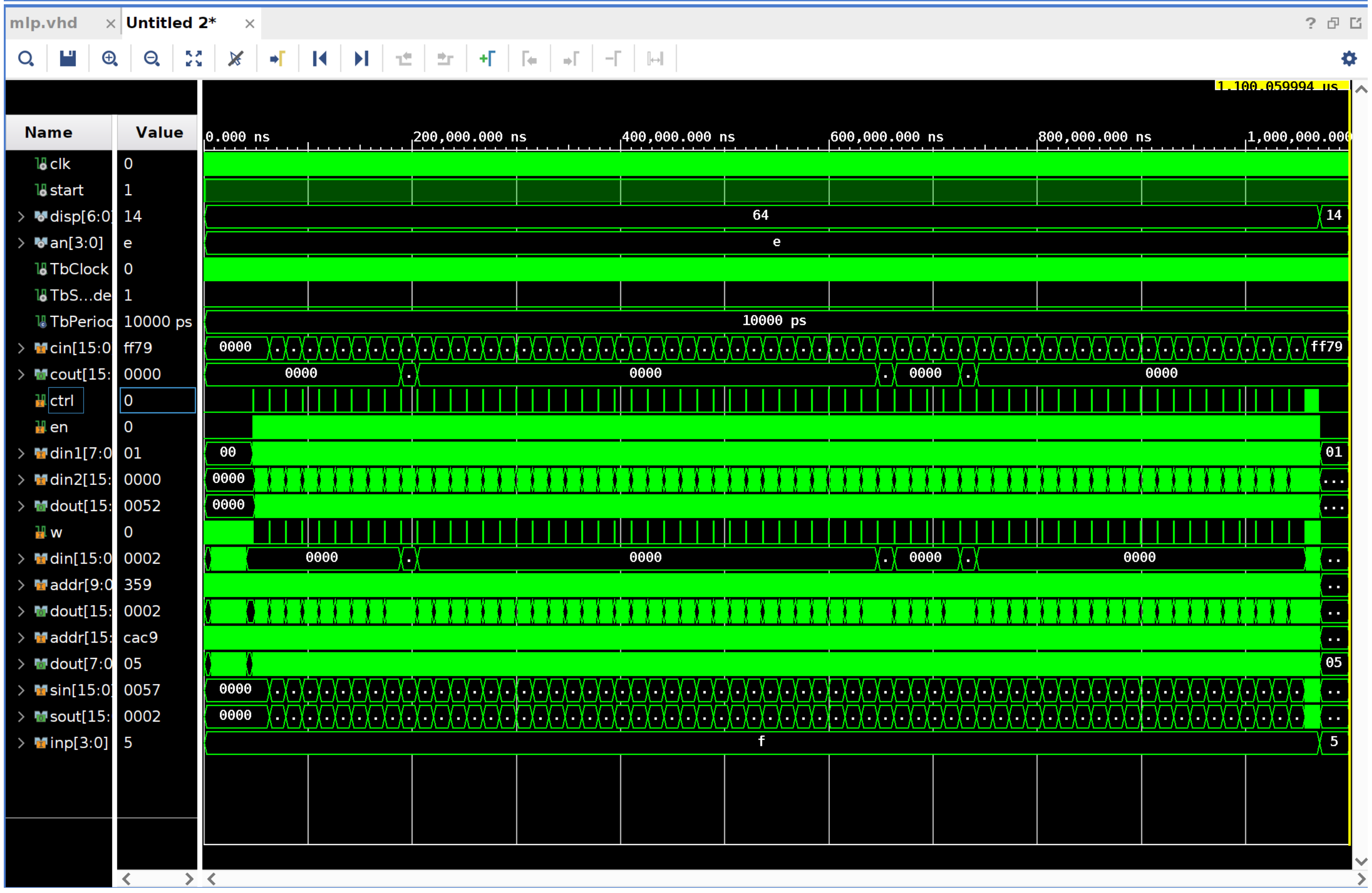
Final Design

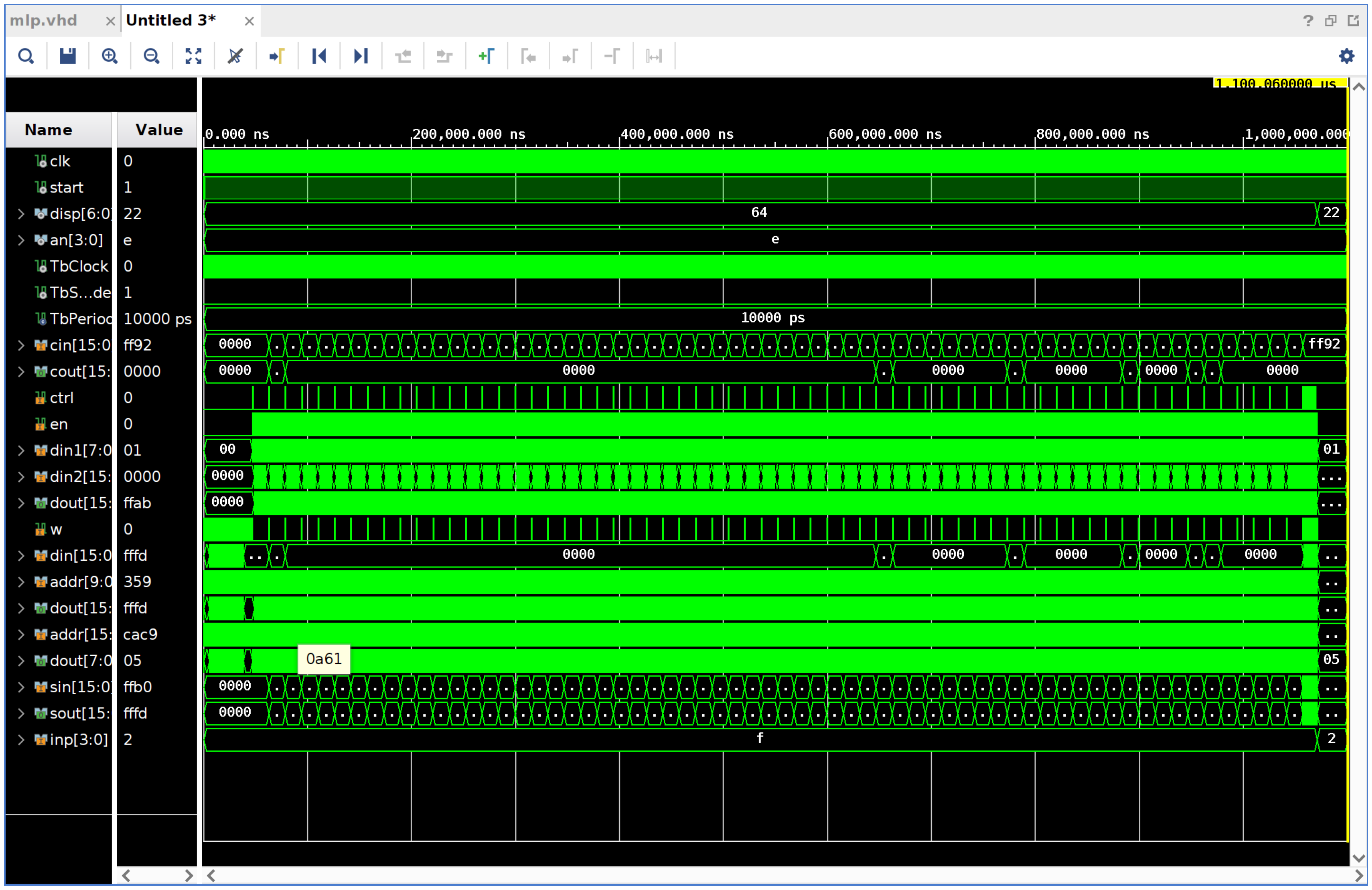
Block Diagram



Waveforms


mlp.vhd
Untitled 4*
Name
Value
clk 0
start 1
disp[6:0] 5a
an[3:0] e
TbClock 0
TbS...de 1
TbPerio 10000 ps
cin[15:0] ff6c
cou...:0 0000
ctrl 0
en 0
din1[7:0] 01
din...:0 0000
dou...:0 0153
w 0
din[15:0] 000a
addr[9:0] 359
dou...:0 000a
add...:0 cac9
dout[7:0] 05
sin[15:0] 0158
sou...:0 000a
inp[3:0] 7
1,100.059993 us
0.000 ns
200,000.000 ns
400,000.000 ns
600,000.000 ns
800,000.000 ns
1,000,000.000
64
5a
e
10000 ps
ff6c
f
7
05


mlp.vhd
Untitled 2*
Name
Value
clk 0
start 1
disp[6:0] 14
an[3:0] e
TbClock 0
TbS...de 1
TbPeriod 10000 ps
cin[15:0] ff79
cout[15:0] 0000
ctrl 0
en 0
din1[7:0] 01
din2[15:0] 0000
dout[15:0] 0052
w 0
din[15:0] 0002
addr[9:0] 359
dout[15:0] 0002
addr[15:0] cac9
dout[7:0] 05
sin[15:0] 0057
sout[15:0] 0002
inp[3:0] 5
1,100.059994 us
0.000 ns
200,000.000 ns
400,000.000 ns
600,000.000 ns
800,000.000 ns
1,000,000.000
64
14
e
10000 ps
ff79
f
5
05



Digits 7,5,2 respectively