LAB 7:

Register File Models

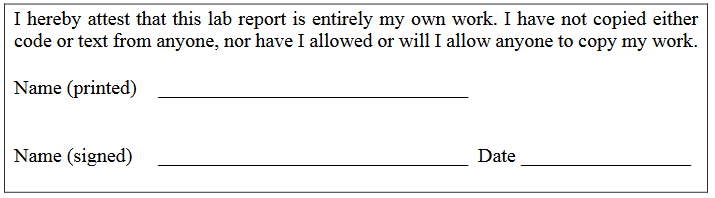
Professor Ronald Mehler

ECE 526L

Spring 2018

Garen Nikoyan

3/29/2018



**Objective:**

The objective of this lab is to create two files which are similar to RAM and ROM. The difference between these modules and actual RAM and ROM is that these modules do not use DRAM or SRAM cells, but instead use flip flops. This is bad in practice because it is the flip-flop based memory will consume more power and would take more area, but it would work much faster.

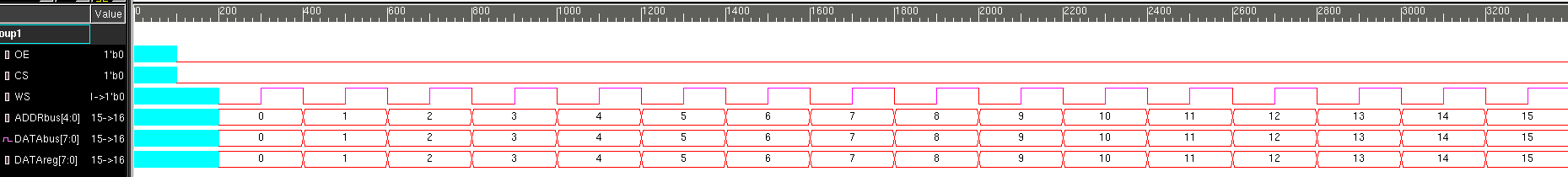
**Methodology:**

For this lab, the first step was to create a module for RAM, and then another for ROM. The main difference between the two is that the RAM could be written to, while the ROM was read-only. The ROM would be loaded with data using $readmemh and a text file. The test benches were then created. For this experiment, two test benches were created, one exclusively for the RAM module, and one for the ROM. The RAM test bench first writes to each memory location. It writes in a sequential order, matching the address number to the data being written. It then reads specifically address block 12, and then does a block read, which reads all of the addresses. It will then set CS high to check the high impedance state. Then, “walking ones” will be written, eg. 0001, 0010, 0100. Lastly, the addresses will be read to confirm the walking ones were written properly. For the second test bench, the ROM is tested by first loading the given data into it. This is done by using $readmemh and then reading each address. Each byte is then scrambled, and written to RAM, and then read from RAM to demonstrate the memory bytes have been successfully scrambled and written. As per Lecture 9 slides, tasks were not used.

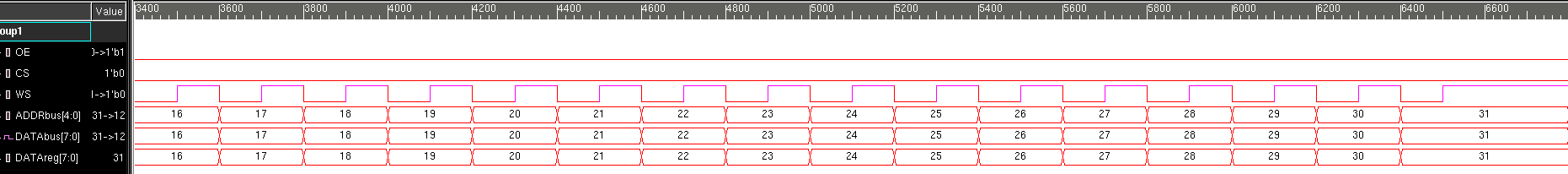
**Analysis:**

Looking at waveforms and the log, it can be seen that the modules worked correctly, as the outputs were as expected.

In Figures 1 and 2, the waveforms of the waveforms during the sequential write can be found. As expected, the RAM takes on the value of DATAreg at the positive edge of the write strobe.

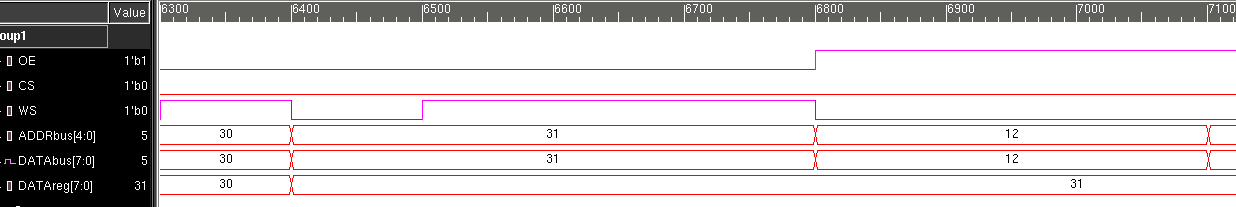


*Figure 1: writing 0-15 to the RAM*

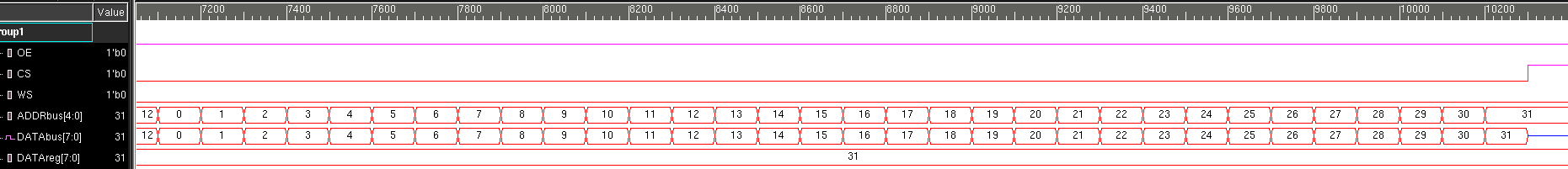
**

*Figure 2: writing 16-31 to the RAM*

In Figure 3, the waveforms during the individual read of address 12 can be found. As expected the value during the read is 12. In Figure 4, the block read can be seen, and each memory value matches the address value.

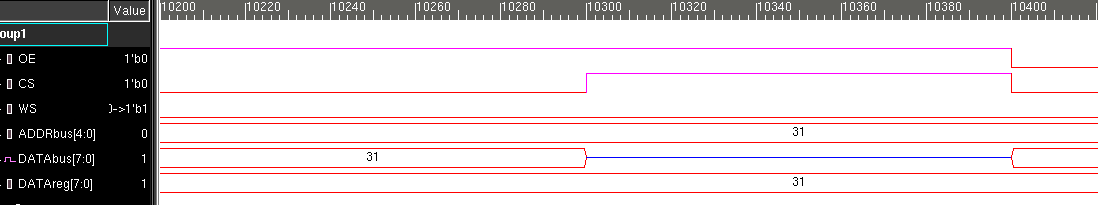


*Figure 3: Reading address 12 of RAM*

**

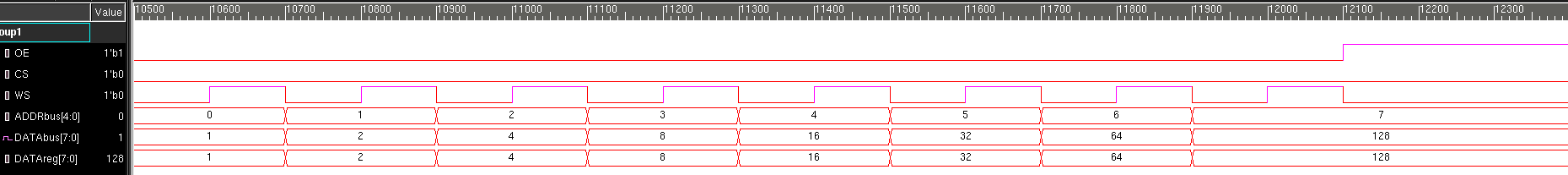
*Figure 4: Block read of RAM*

In Figure 5 CS is set high, and the databus goes to Z.

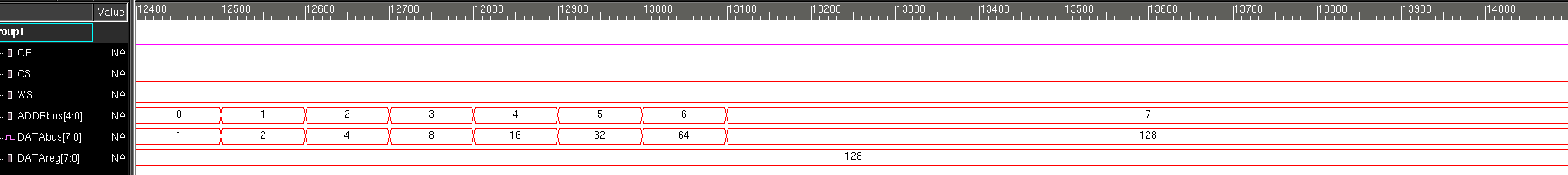


*Figure 5: Setting CS high, DATAbus goes to Z*

In Figures 6 and 7, the walking ones write and read can be found. To read the walking ones, the same methodology used in the block read was used.

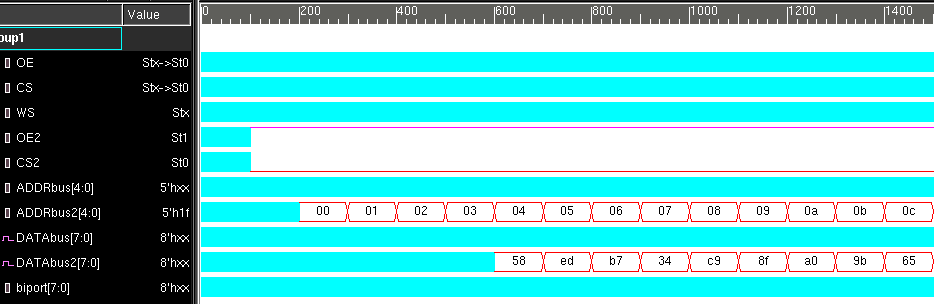
**

*Figure 6: Writing walking ones to RAM*

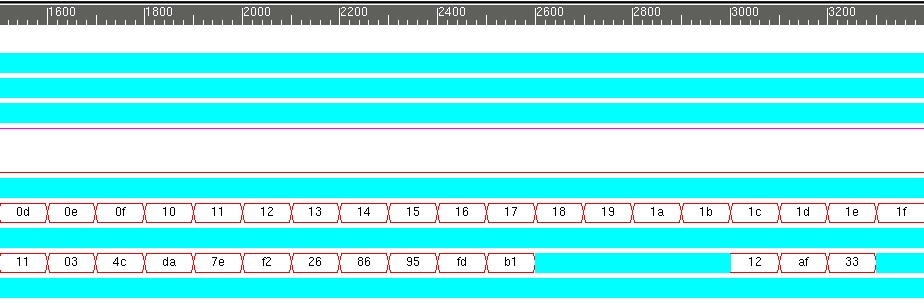
**

*Figure 7: Reading walking ones from RAM*

Now the ROM test bench was used to test the ROM. First the given data was loaded into a text file, and this data was loaded into the ROM by using $readmemh. This data was read from the ROM, and the output waveforms can be found in Figure 8 and 9.

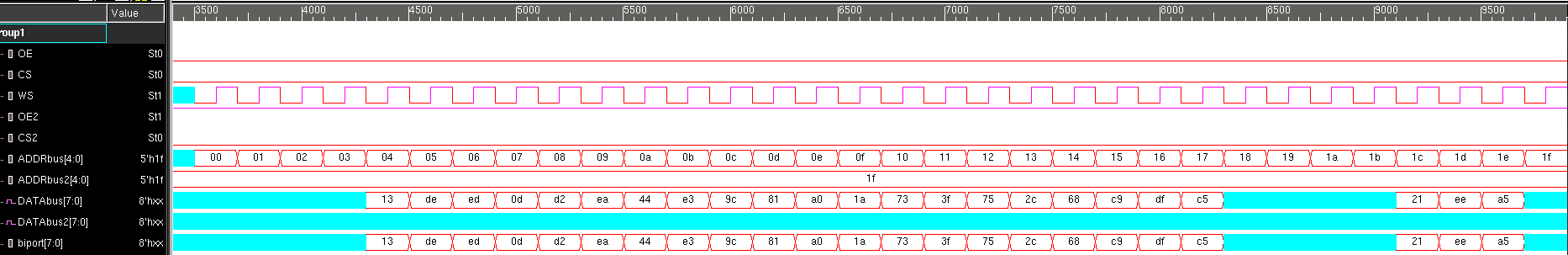
**

*Figure 8: Reading data from ROM (continued in Figure 9)*

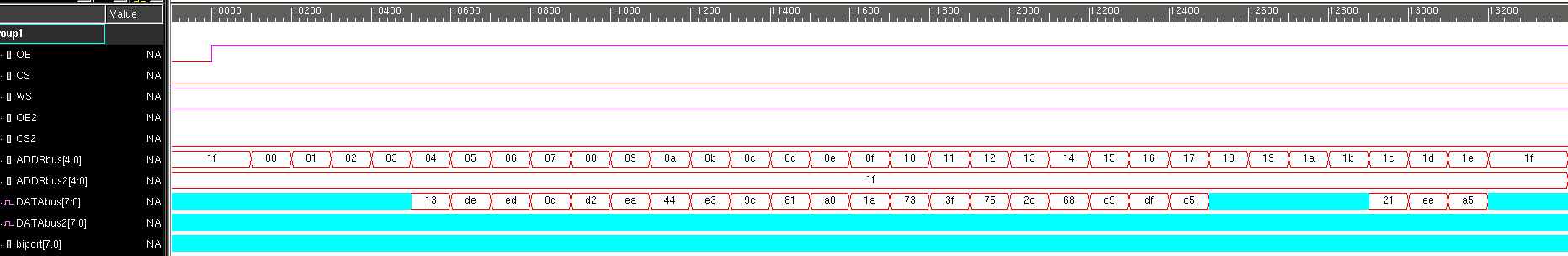
**

*Figure 9: Reading data from ROM (continuation of Figure 8)*

After the read, the data from the ROM is loaded into a buffer and scrambled, and then it is loaded into RAM. The scramble was done without time delays so it cannot be seen in the waveforms. In figure 10, the waveforms from writing from the buffer into RAM can be found. This data is then block read from the ram, and the waveforms can be found in figure 11.



*Figure 10: Writing from buffer into RAM*



*Figure 11: Block read of scrambled data in RAM*

**Modules:**

**RAM**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* ECE526L Experiment #7 Garen Nikoyan, Spring 2018

\*\*\* Register File Models

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* Filename: RAM.v Created by: Garen Nikoyan, 3/29/2018 \*\*\*

\*\*\* -Revision History

\*\*\* 3/29/2018: First draft

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* This module models memory with an 8-bit data bus and 5-bit address bus

\*\*\* This module uses an active high output enable signal, active low chip

\*\*\* select signal, and a write strobe signal. If chip select is high,

\*\*\* or output enable is low, data bus goes to Z

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

`timescale 1ns/100ps

module RAM(DATA, ADDR, OE, CS, WS);

parameter WIDTH = 8;

parameter DEPTH = 5;

inout [WIDTH-1 : 0] DATA;

input [DEPTH-1 : 0] ADDR;

input OE, CS, WS;

reg [WIDTH-1 : 0] memory [0: 2\*\*DEPTH-1];

// if OE=1 and CS=0, DATA=memory at ADDR; if not, DATA=z

// assign is used because of inout type

assign DATA = (OE && !CS) ? memory[ADDR] : 8'bz;

always@(posedge WS) begin

if (!OE && !CS)

memory[ADDR] <= DATA;

else;

end

endmodule

**ROM**

`timescale 1ns/100ps

module ROM(DATA, ADDR, OE, CS);

parameter WIDTH=8;

parameter DEPTH=5;

output reg [WIDTH-1 : 0] DATA;

input wire [DEPTH-1 : 0] ADDR;

input OE, CS;

reg [WIDTH-1 : 0] memory [0 : 2\*\*DEPTH-1];

// if OE=1 and CS=0, DATA=memory at ADDR; if not, DATA=z

always@\* DATA = (OE && !CS) ? memory[ADDR] : 8'bz;

endmodule

**Testbench:**

**RAM Test Bench**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* ECE526L Experiment #7 Garen Nikoyan, Spring 2018

\*\*\* Register File Models

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* Filename: RAM\_TB.v Created by: Garen Nikoyan, 3/29/2018 \*\*\*

\*\*\* -Revision History

\*\*\* 3/29/2018: First draft

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* This module tests a RAM module \*\*\*

\*\*\* It will write and read from every memory location using sequential

\*\*\* numbering that matches the address number. Then tests a block read

\*\*\* Verifies both enabled and disabled states, and tests high impedance

\*\*\* state. It will lastly write a walking ones pattern and then read it

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

`timescale 1ns/100ps

module RAM\_TB();

parameter WIDTH=8;

parameter DEPTH=5;

reg OE, CS, WS;

reg [DEPTH-1 : 0] ADDRbus;

wire [WIDTH-1 : 0] DATAbus;

reg [WIDTH-1 : 0] DATAreg;

integer i,j,k;

// needs to be assign because it is an inout port in the module

assign DATAbus = (!OE && !CS) ? DATAreg:8'bz;

RAM RAM1(DATAbus, ADDRbus, OE, CS, WS);

initial begin

$vcdpluson;

//$dumpfile("dump.vcd");

//$dumpvars;

$monitor(" ADDRbus =%d CS =%b OE =%b WS =%b DATAbus =%d DATAreg =%d", ADDRbus, CS, OE, WS, DATAbus, DATAreg);

end

// \*\*\* Test Write

initial begin

$monitoroff;

#10 CS=0; OE=0;

$display("\n\t\t TEST WRITE");

$monitoron;

for(i=0;i<32;i=i+1) begin

#10 $monitoroff; WS=0; DATAreg=i; ADDRbus=i;

#10 $monitoron; WS=1;

end

// \*\*\* Individual Read

#20 $display("\n\t\t INDIVIDUAL READ OF ADDR 12");

#10 CS=0; OE=1; WS=0; ADDRbus=12;

// \*\*\* Block Read

#10 $monitoroff;

#10 CS=0; OE=1;

$display("\n\t\t BLOCK READ");

for(j=0;j<32;j=j+1) begin

$strobe("\t\t RAM[%2d] = %d",j, RAM.memory[j]);

#10 ADDRbus=j;

end

// \*\*\* High Impedance State

#10 CS=1; $monitoron;

$display("\n\t\t HIGH IMPEDANCE TEST");

// \*\*\* Walking Ones

$monitorb("ADDRbus =%d CS =%b OE =%b WS =%b DATAbus =%b DATAreg =%b", ADDRbus, CS, OE, WS, DATAbus, DATAreg);

#10 k=1; CS=0; OE=0;

$display("\n\t\t WRITING WALKING ONES");

for(i=0;i<8;i=i+1) begin

#10 $monitoroff; WS=0; ADDRbus=i; DATAreg=k;

#10 $monitoron; WS=1; k=k\*2;

if(k==256) k=1;

else;

end

#10 $monitoroff; OE=1; WS=0; CS=0;

#10 $display("\n\t\t READING WALKING ONES");

#10 $monitoron;

for(i=0; i<8; i=i+1) #10 ADDRbus=i;

#100 $finish;

end

endmodule

**ROM Test Bench**

`timescale 1ns/100ps

module ROM\_TB();

parameter WIDTH=8;

parameter DEPTH=5;

reg OE,CS,WS,OE2,CS2;

reg [DEPTH-1 : 0] ADDRbus, ADDRbus2;

wire [WIDTH-1 : 0] DATAbus, DATAbus2;

reg [WIDTH-1 : 0] biport;

reg [WIDTH-1 : 0] BuffRAM [0: 2\*\*DEPTH-1];

integer i,j,k;

RAM RAM\_UUT(DATAbus, ADDRbus, OE, CS, WS);

ROM ROM\_UUT(DATAbus2, ADDRbus2, OE2, CS2);

assign DATAbus = (!OE && !CS) ? biport:8'bz;

initial $monitor("ADDR2 =%h, DATA2 =%h, OE2 =%b, CS2 =%b", ADDRbus2, DATAbus2, OE2, CS2);

initial $readmemh("ROM\_data.txt", ROM\_UUT.memory);

initial begin

$vcdpluson;

// \*\*\* Read

$display("\t\t READ");

#10 CS2=0; OE2=1;

for(i=0; i<32; i=i+1) #10 ADDRbus2 = i;

// \*\*\* Scramble

$display("\nSCRAMBLING DATA OF ROM INTO BUFFER RAM");

for (i=0; i<32; i=i+1) begin

k=0;

for(j=7; j>0; j=j-2) begin

if( ROM\_UUT.memory[i] != 0) begin

BuffRAM[i][j] = ROM\_UUT.memory[i][k];

k = k+1;

end

else;

end

for(j=0; j<7; j=j+2) begin

if(ROM\_UUT.memory[i] != 0) begin

BuffRAM[i][j] = ROM\_UUT.memory[i][k];

k = k+1;

end

else;

end

end

// \*\*\* Write to RAM

$monitor("ADDR =%h, DATA =%h, OE =%b, CS =%b, WS =%b", ADDRbus, DATAbus, OE, CS, WS);

$monitoroff;

#10 CS=0; OE=0;

$display("\n\t\t WRITE SCRAMBLED DATA TO RAM");

$monitoron;

for(i=0;i<32;i=i+1) begin

#10 $monitoroff; WS=0; biport=BuffRAM[i]; ADDRbus=i;

#10 $monitoron; WS=1;

end

// \*\*\* Block Read

#10 $monitoroff;

#10 CS=0; OE=1;

$display("\n\t\t BLOCK READ OF RAM");

for(j=0;j<32;j=j+1) begin

$strobe("\t\t RAM[%2d] = %d",j, RAM\_UUT.memory[j]);

#10 ADDRbus=j;

end

#20 $finish;

end

endmodule

**Log:**

**RAM Log**

Chronologic VCS simulator copyright 1991-2017

Contains Synopsys proprietary information.

Compiler version M-2017.03-SP1\_Full64; Runtime version M-2017.03-SP1\_Full64; Apr 5 12:37 2018

VCD+ Writer M-2017.03-SP1\_Full64 Copyright (c) 1991-2017 by Synopsys Inc.

TEST WRITE

ADDRbus = x CS =0 OE =0 WS =x DATAbus = x DATAreg = x

ADDRbus = 0 CS =0 OE =0 WS =1 DATAbus = 0 DATAreg = 0

ADDRbus = 1 CS =0 OE =0 WS =1 DATAbus = 1 DATAreg = 1

ADDRbus = 2 CS =0 OE =0 WS =1 DATAbus = 2 DATAreg = 2

ADDRbus = 3 CS =0 OE =0 WS =1 DATAbus = 3 DATAreg = 3

ADDRbus = 4 CS =0 OE =0 WS =1 DATAbus = 4 DATAreg = 4

ADDRbus = 5 CS =0 OE =0 WS =1 DATAbus = 5 DATAreg = 5

ADDRbus = 6 CS =0 OE =0 WS =1 DATAbus = 6 DATAreg = 6

ADDRbus = 7 CS =0 OE =0 WS =1 DATAbus = 7 DATAreg = 7

ADDRbus = 8 CS =0 OE =0 WS =1 DATAbus = 8 DATAreg = 8

ADDRbus = 9 CS =0 OE =0 WS =1 DATAbus = 9 DATAreg = 9

ADDRbus =10 CS =0 OE =0 WS =1 DATAbus = 10 DATAreg = 10

ADDRbus =11 CS =0 OE =0 WS =1 DATAbus = 11 DATAreg = 11

ADDRbus =12 CS =0 OE =0 WS =1 DATAbus = 12 DATAreg = 12

ADDRbus =13 CS =0 OE =0 WS =1 DATAbus = 13 DATAreg = 13

ADDRbus =14 CS =0 OE =0 WS =1 DATAbus = 14 DATAreg = 14

ADDRbus =15 CS =0 OE =0 WS =1 DATAbus = 15 DATAreg = 15

ADDRbus =16 CS =0 OE =0 WS =1 DATAbus = 16 DATAreg = 16

ADDRbus =17 CS =0 OE =0 WS =1 DATAbus = 17 DATAreg = 17

ADDRbus =18 CS =0 OE =0 WS =1 DATAbus = 18 DATAreg = 18

ADDRbus =19 CS =0 OE =0 WS =1 DATAbus = 19 DATAreg = 19

ADDRbus =20 CS =0 OE =0 WS =1 DATAbus = 20 DATAreg = 20

ADDRbus =21 CS =0 OE =0 WS =1 DATAbus = 21 DATAreg = 21

ADDRbus =22 CS =0 OE =0 WS =1 DATAbus = 22 DATAreg = 22

ADDRbus =23 CS =0 OE =0 WS =1 DATAbus = 23 DATAreg = 23

ADDRbus =24 CS =0 OE =0 WS =1 DATAbus = 24 DATAreg = 24

ADDRbus =25 CS =0 OE =0 WS =1 DATAbus = 25 DATAreg = 25

ADDRbus =26 CS =0 OE =0 WS =1 DATAbus = 26 DATAreg = 26

ADDRbus =27 CS =0 OE =0 WS =1 DATAbus = 27 DATAreg = 27

ADDRbus =28 CS =0 OE =0 WS =1 DATAbus = 28 DATAreg = 28

ADDRbus =29 CS =0 OE =0 WS =1 DATAbus = 29 DATAreg = 29

ADDRbus =30 CS =0 OE =0 WS =1 DATAbus = 30 DATAreg = 30

ADDRbus =31 CS =0 OE =0 WS =1 DATAbus = 31 DATAreg = 31

INDIVIDUAL READ OF ADDR 12

ADDRbus =12 CS =0 OE =1 WS =0 DATAbus = 12 DATAreg = 31

BLOCK READ

RAM[ 0] = 0

RAM[ 1] = 1

RAM[ 2] = 2

RAM[ 3] = 3

RAM[ 4] = 4

RAM[ 5] = 5

RAM[ 6] = 6

RAM[ 7] = 7

RAM[ 8] = 8

RAM[ 9] = 9

RAM[10] = 10

RAM[11] = 11

RAM[12] = 12

RAM[13] = 13

RAM[14] = 14

RAM[15] = 15

RAM[16] = 16

RAM[17] = 17

RAM[18] = 18

RAM[19] = 19

RAM[20] = 20

RAM[21] = 21

RAM[22] = 22

RAM[23] = 23

RAM[24] = 24

RAM[25] = 25

RAM[26] = 26

RAM[27] = 27

RAM[28] = 28

RAM[29] = 29

RAM[30] = 30

RAM[31] = 31

HIGH IMPEDANCE TEST

ADDRbus =31 CS =1 OE =1 WS =0 DATAbus =zzzzzzzz DATAreg =00011111

WRITING WALKING ONES

ADDRbus = 0 CS =0 OE =0 WS =1 DATAbus =00000001 DATAreg =00000001

ADDRbus = 1 CS =0 OE =0 WS =1 DATAbus =00000010 DATAreg =00000010

ADDRbus = 2 CS =0 OE =0 WS =1 DATAbus =00000100 DATAreg =00000100

ADDRbus = 3 CS =0 OE =0 WS =1 DATAbus =00001000 DATAreg =00001000

ADDRbus = 4 CS =0 OE =0 WS =1 DATAbus =00010000 DATAreg =00010000

ADDRbus = 5 CS =0 OE =0 WS =1 DATAbus =00100000 DATAreg =00100000

ADDRbus = 6 CS =0 OE =0 WS =1 DATAbus =01000000 DATAreg =01000000

ADDRbus = 7 CS =0 OE =0 WS =1 DATAbus =10000000 DATAreg =10000000

READING WALKING ONES

ADDRbus = 0 CS =0 OE =1 WS =0 DATAbus =00000001 DATAreg =10000000

ADDRbus = 1 CS =0 OE =1 WS =0 DATAbus =00000010 DATAreg =10000000

ADDRbus = 2 CS =0 OE =1 WS =0 DATAbus =00000100 DATAreg =10000000

ADDRbus = 3 CS =0 OE =1 WS =0 DATAbus =00001000 DATAreg =10000000

ADDRbus = 4 CS =0 OE =1 WS =0 DATAbus =00010000 DATAreg =10000000

ADDRbus = 5 CS =0 OE =1 WS =0 DATAbus =00100000 DATAreg =10000000

ADDRbus = 6 CS =0 OE =1 WS =0 DATAbus =01000000 DATAreg =10000000

ADDRbus = 7 CS =0 OE =1 WS =0 DATAbus =10000000 DATAreg =10000000

$finish called from file "RAM\_TB.v", line 66.

$finish at simulation time 14100

V C S S i m u l a t i o n R e p o r t

Time: 1410000 ps

CPU Time: 0.220 seconds; Data structure size: 0.0Mb

Thu Apr 5 12:37:01 2018

**ROM Log**

Chronologic VCS simulator copyright 1991-2017

Contains Synopsys proprietary information.

Compiler version M-2017.03-SP1\_Full64; Runtime version M-2017.03-SP1\_Full64; Apr 5 13:15 2018

VCD+ Writer M-2017.03-SP1\_Full64 Copyright (c) 1991-2017 by Synopsys Inc.

READ

ADDR2 =xx, DATA2 =xx, OE2 =x, CS2 =x

ADDR2 =xx, DATA2 =xx, OE2 =1, CS2 =0

ADDR2 =00, DATA2 =xx, OE2 =1, CS2 =0

ADDR2 =01, DATA2 =xx, OE2 =1, CS2 =0

ADDR2 =02, DATA2 =xx, OE2 =1, CS2 =0

ADDR2 =03, DATA2 =xx, OE2 =1, CS2 =0

ADDR2 =04, DATA2 =58, OE2 =1, CS2 =0

ADDR2 =05, DATA2 =ed, OE2 =1, CS2 =0

ADDR2 =06, DATA2 =b7, OE2 =1, CS2 =0

ADDR2 =07, DATA2 =34, OE2 =1, CS2 =0

ADDR2 =08, DATA2 =c9, OE2 =1, CS2 =0

ADDR2 =09, DATA2 =8f, OE2 =1, CS2 =0

ADDR2 =0a, DATA2 =a0, OE2 =1, CS2 =0

ADDR2 =0b, DATA2 =9b, OE2 =1, CS2 =0

ADDR2 =0c, DATA2 =65, OE2 =1, CS2 =0

ADDR2 =0d, DATA2 =11, OE2 =1, CS2 =0

ADDR2 =0e, DATA2 =03, OE2 =1, CS2 =0

ADDR2 =0f, DATA2 =4c, OE2 =1, CS2 =0

ADDR2 =10, DATA2 =da, OE2 =1, CS2 =0

ADDR2 =11, DATA2 =7e, OE2 =1, CS2 =0

ADDR2 =12, DATA2 =f2, OE2 =1, CS2 =0

ADDR2 =13, DATA2 =26, OE2 =1, CS2 =0

ADDR2 =14, DATA2 =86, OE2 =1, CS2 =0

ADDR2 =15, DATA2 =95, OE2 =1, CS2 =0

ADDR2 =16, DATA2 =fd, OE2 =1, CS2 =0

ADDR2 =17, DATA2 =b1, OE2 =1, CS2 =0

ADDR2 =18, DATA2 =xx, OE2 =1, CS2 =0

ADDR2 =19, DATA2 =xx, OE2 =1, CS2 =0

ADDR2 =1a, DATA2 =xx, OE2 =1, CS2 =0

ADDR2 =1b, DATA2 =xx, OE2 =1, CS2 =0

ADDR2 =1c, DATA2 =12, OE2 =1, CS2 =0

ADDR2 =1d, DATA2 =af, OE2 =1, CS2 =0

ADDR2 =1e, DATA2 =33, OE2 =1, CS2 =0

SCRAMBLING DATA OF ROM INTO BUFFER RAM

WRITE SCRAMBLED DATA TO RAM

ADDR =xx, DATA =xx, OE =0, CS =0, WS =x

ADDR =00, DATA =xx, OE =0, CS =0, WS =1

ADDR =01, DATA =xx, OE =0, CS =0, WS =1

ADDR =02, DATA =xx, OE =0, CS =0, WS =1

ADDR =03, DATA =xx, OE =0, CS =0, WS =1

ADDR =04, DATA =13, OE =0, CS =0, WS =1

ADDR =05, DATA =de, OE =0, CS =0, WS =1

ADDR =06, DATA =ed, OE =0, CS =0, WS =1

ADDR =07, DATA =0d, OE =0, CS =0, WS =1

ADDR =08, DATA =d2, OE =0, CS =0, WS =1

ADDR =09, DATA =ea, OE =0, CS =0, WS =1

ADDR =0a, DATA =44, OE =0, CS =0, WS =1

ADDR =0b, DATA =e3, OE =0, CS =0, WS =1

ADDR =0c, DATA =9c, OE =0, CS =0, WS =1

ADDR =0d, DATA =81, OE =0, CS =0, WS =1

ADDR =0e, DATA =a0, OE =0, CS =0, WS =1

ADDR =0f, DATA =1a, OE =0, CS =0, WS =1

ADDR =10, DATA =73, OE =0, CS =0, WS =1

ADDR =11, DATA =3f, OE =0, CS =0, WS =1

ADDR =12, DATA =75, OE =0, CS =0, WS =1

ADDR =13, DATA =2c, OE =0, CS =0, WS =1

ADDR =14, DATA =68, OE =0, CS =0, WS =1

ADDR =15, DATA =c9, OE =0, CS =0, WS =1

ADDR =16, DATA =df, OE =0, CS =0, WS =1

ADDR =17, DATA =c5, OE =0, CS =0, WS =1

ADDR =18, DATA =xx, OE =0, CS =0, WS =1

ADDR =19, DATA =xx, OE =0, CS =0, WS =1

ADDR =1a, DATA =xx, OE =0, CS =0, WS =1

ADDR =1b, DATA =xx, OE =0, CS =0, WS =1

ADDR =1c, DATA =21, OE =0, CS =0, WS =1

ADDR =1d, DATA =ee, OE =0, CS =0, WS =1

ADDR =1e, DATA =a5, OE =0, CS =0, WS =1

ADDR =1f, DATA =xx, OE =0, CS =0, WS =1

BLOCK READ OF RAM

RAM[ 0] = x

RAM[ 1] = x

RAM[ 2] = x

RAM[ 3] = x

RAM[ 4] = 19

RAM[ 5] = 222

RAM[ 6] = 237

RAM[ 7] = 13

RAM[ 8] = 210

RAM[ 9] = 234

RAM[10] = 68

RAM[11] = 227

RAM[12] = 156

RAM[13] = 129

RAM[14] = 160

RAM[15] = 26

RAM[16] = 115

RAM[17] = 63

RAM[18] = 117

RAM[19] = 44

RAM[20] = 104

RAM[21] = 201

RAM[22] = 223

RAM[23] = 197

RAM[24] = x

RAM[25] = x

RAM[26] = x

RAM[27] = x

RAM[28] = 33

RAM[29] = 238

RAM[30] = 165

RAM[31] = x

$finish called from file "ROM\_TB.v", line 66.

$finish at simulation time 13400

V C S S i m u l a t i o n R e p o r t

Time: 1340000 ps

CPU Time: 0.210 seconds; Data structure size: 0.0Mb

Thu Apr 5 13:15:31 2018

**Conclusion:**

Everything in this lab worked as expected. It was seen that both RAM and ROM can be created with just flip-flops, although it comes with the size and power consumption caveats. Along with this, creating memory blocks was also done by using two dimensional arrays.

With regards to the lab question, only one edge construct was used, which was the write strobe positive edge construct, used for the RAM. It seems that this is the best design as only dealing with one edge construct can be enough for completing the requirements set by the lab manual.