Spring 2021

# California State University, Northridge

Department of Electrical & Computer Engineering


# Experiment 7

# Register File Models

April 10, 2021

## ECE 526L

Written By: Jose Luis Martinez

## Introduction

In this lab we are tasked to design RAM and ROM modules using register files. Both of these modules are to be made with scalable address depths and data widths so that they can be used elsewhere.

The RAM module is to be designed as shown in **Fig. 1** with the Data port being bidirectional. The RAM block will have an ADDR input which will specify which address to read or write from. A OE input which will enable or disabled the output of the RAM. A WS input that on the rising edge will write a value to the current address if the OE is logic LOW. Finally the RAM will have a CS line that will enable the RAM module when it is logic level LOW.

The ROM module is to be designed as shown in **Fig. 2,** with this time the Data port should only be an output. The ROM module will have an ADDR input which will specify which address to read from. The OE line will enable or disable the output. The CS line will enable on LOW and disable on HIGH.


**Fig. 1** RAM Register File Model

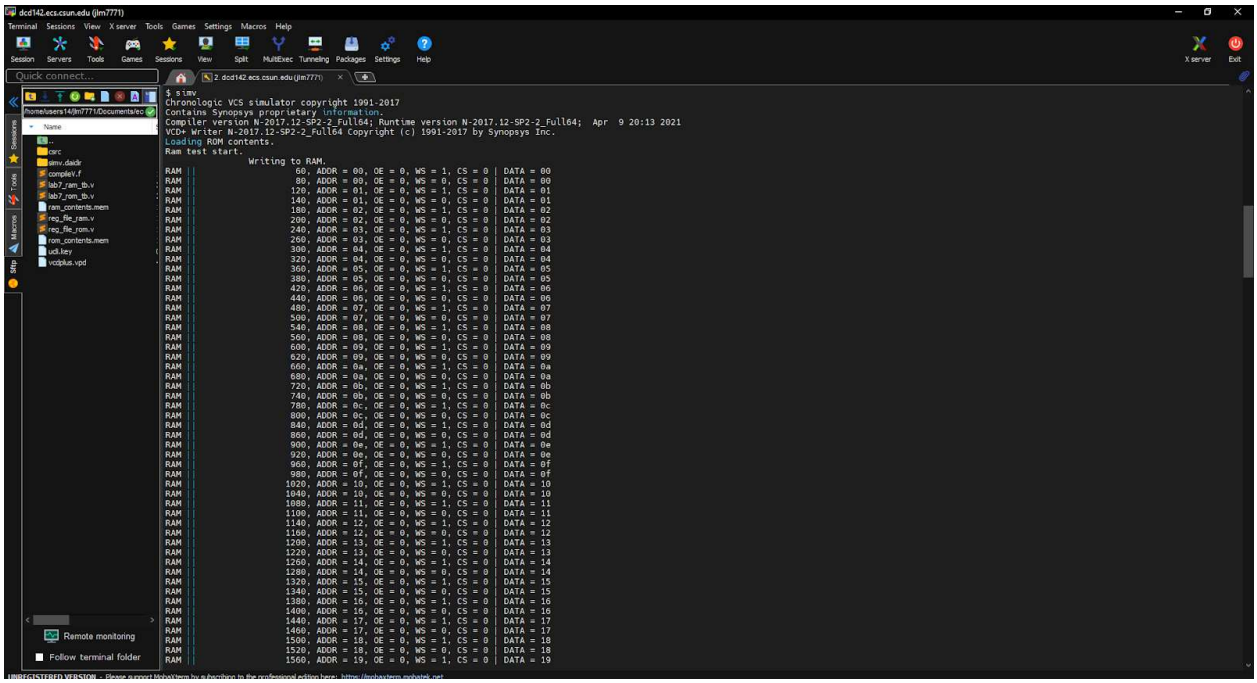

**Fig. 2** ROM Register File Model

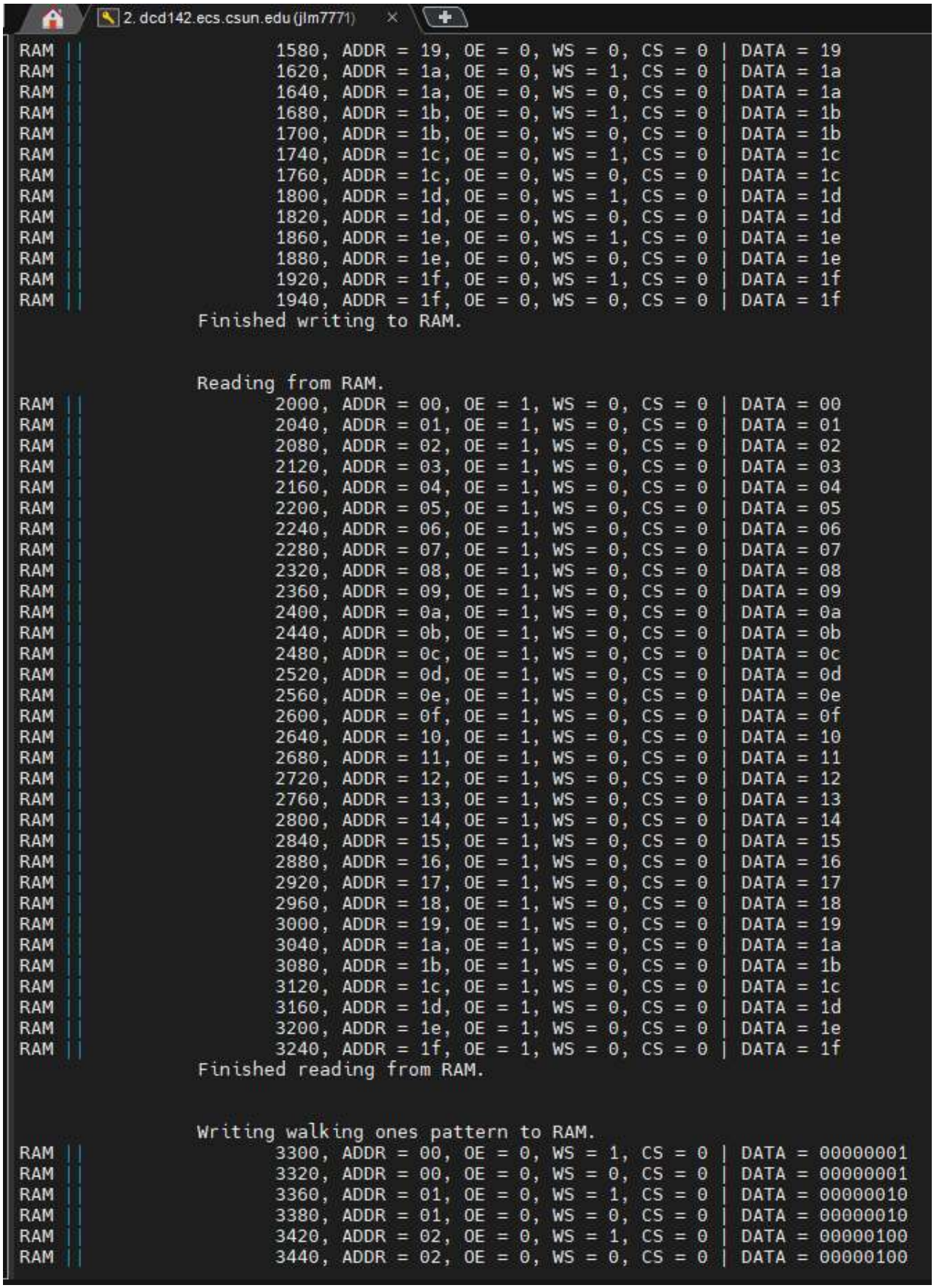
# Methodology

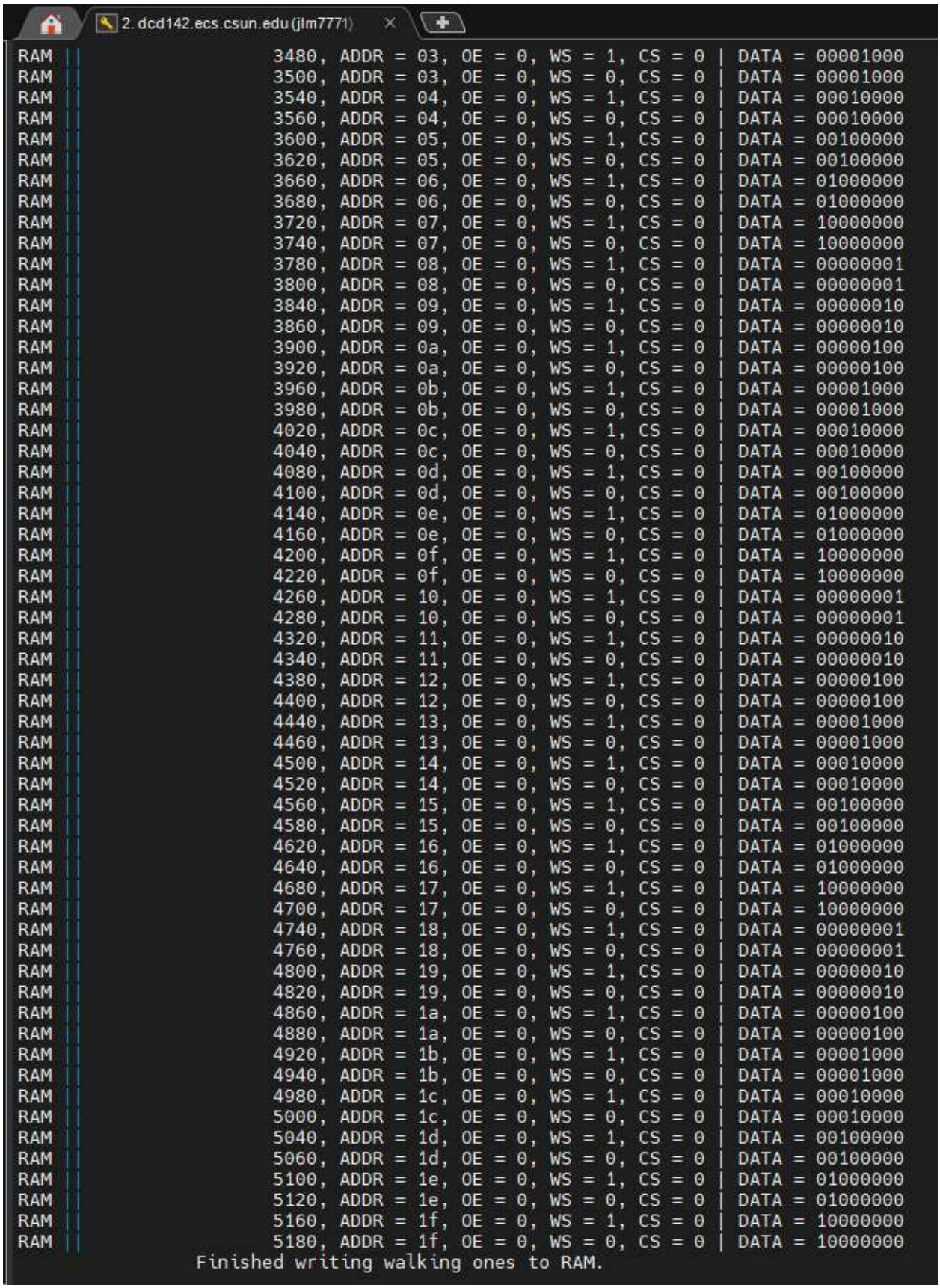
I followed these steps in order to complete the lab:

1. The RAM module was created as shown in **Fig. 7.**
    a. The RAM module has four inputs and one inout port.
        i. The ROM module has parameters for WIDTH and DEPTH so it can be modified easily for other applications.
        ii. The module has an assign statement that will set DATA to the value of the current address only when OE is HIGH and CS is LOW.
        iii. The module has an always statement that will check for the positive edge of WS and if OE and CS are both LOW the module will write whatever is in the DATA line to the current address.
2. The ROM module was created as shown in **Fig. 8.**
    a. The ROM module has three inputs and one output port.
        i. The ROM module has parameters for WIDTH and DEPTH so it can be modified easily for other applications.
        ii. The module has an assign statement that will set DATA to the value of the current address only when OE is HIGH and CS is LOW.
        iii. The module also has an initial begin block with a $readmemh to load up the ROM contents from the rom_contents.mem file.
3. After the base modules were made, I made two test benches to test out the required tasks assigned for each of the modules.
4. The first testbench was created as shown in the file named lab7_ram_tb.v in **Fig. 9.**
    a. The testbench file has an instantiation of the reg_file_ram module.
    b. The testbench first writes to each address the value of the address so address 0x00 has a value of 0x00 and 0x01 has a value of 0x01 and so on.
    c. Then the testbench will read each one of the values by enabling the OE and going from each address 0x00 to 0x1f.
    d. Then the test bench will repeat a similar process with walking ones.
    e. Finally the test bench will show the disabled state of the RAM module.
    f. Notice how this testbench does not have a $finish. This is because the second testbench will run immediately after this one so removing the $finish will stop the simulation from ending prematurely.
5. The second test bench file was created as shown in the file names lab7_rom_tb.v in **Fig. 10.**
    a. This testbench has one instantiation of the RAM module and another of the ROM module.
    b. The first part of the testbench goes through all of the addresses and reads out their DATA in order to ensure that the ROM was initialized correctly using $readmemh.
    c. The second part of the testbench reads data from the ROM and scrambles it into the RAM. Then the contents of the RAM is read in order to ensure that the data was scrambled correctly.
6. With the testbenches implemented I was able to verify that my modules are working according to the specifications.

# Results/Verilog Files




2. dcd142.ecs.csun.edu (jlm7771)
RAM || 1580, ADDR = 19, OE = 0, WS = 0, CS = 0 | DATA = 19
RAM || 1620, ADDR = 1a, OE = 0, WS = 1, CS = 0 | DATA = 1a
RAM || 1640, ADDR = 1a, OE = 0, WS = 0, CS = 0 | DATA = 1a
RAM || 1680, ADDR = 1b, OE = 0, WS = 1, CS = 0 | DATA = 1b
RAM || 1700, ADDR = 1b, OE = 0, WS = 0, CS = 0 | DATA = 1b
RAM || 1740, ADDR = 1c, OE = 0, WS = 1, CS = 0 | DATA = 1c
RAM || 1760, ADDR = 1c, OE = 0, WS = 0, CS = 0 | DATA = 1c
RAM || 1800, ADDR = 1d, OE = 0, WS = 1, CS = 0 | DATA = 1d
RAM || 1820, ADDR = 1d, OE = 0, WS = 0, CS = 0 | DATA = 1d
RAM || 1860, ADDR = 1e, OE = 0, WS = 1, CS = 0 | DATA = 1e
RAM || 1880, ADDR = 1e, OE = 0, WS = 0, CS = 0 | DATA = 1e
RAM || 1920, ADDR = 1f, OE = 0, WS = 1, CS = 0 | DATA = 1f
RAM || 1940, ADDR = 1f, OE = 0, WS = 0, CS = 0 | DATA = 1f
Finished writing to RAM.
Reading from RAM.
RAM || 2000, ADDR = 00, OE = 1, WS = 0, CS = 0 | DATA = 00
RAM || 2040, ADDR = 01, OE = 1, WS = 0, CS = 0 | DATA = 01
RAM || 2080, ADDR = 02, OE = 1, WS = 0, CS = 0 | DATA = 02
RAM || 2120, ADDR = 03, OE = 1, WS = 0, CS = 0 | DATA = 03
RAM || 2160, ADDR = 04, OE = 1, WS = 0, CS = 0 | DATA = 04
RAM || 2200, ADDR = 05, OE = 1, WS = 0, CS = 0 | DATA = 05
RAM || 2240, ADDR = 06, OE = 1, WS = 0, CS = 0 | DATA = 06
RAM || 2280, ADDR = 07, OE = 1, WS = 0, CS = 0 | DATA = 07
RAM || 2320, ADDR = 08, OE = 1, WS = 0, CS = 0 | DATA = 08
RAM || 2360, ADDR = 09, OE = 1, WS = 0, CS = 0 | DATA = 09
RAM || 2400, ADDR = 0a, OE = 1, WS = 0, CS = 0 | DATA = 0a
RAM || 2440, ADDR = 0b, OE = 1, WS = 0, CS = 0 | DATA = 0b
RAM || 2480, ADDR = 0c, OE = 1, WS = 0, CS = 0 | DATA = 0c
RAM || 2520, ADDR = 0d, OE = 1, WS = 0, CS = 0 | DATA = 0d
RAM || 2560, ADDR = 0e, OE = 1, WS = 0, CS = 0 | DATA = 0e
RAM || 2600, ADDR = 0f, OE = 1, WS = 0, CS = 0 | DATA = 0f
RAM || 2640, ADDR = 10, OE = 1, WS = 0, CS = 0 | DATA = 10
RAM || 2680, ADDR = 11, OE = 1, WS = 0, CS = 0 | DATA = 11
RAM || 2720, ADDR = 12, OE = 1, WS = 0, CS = 0 | DATA = 12
RAM || 2760, ADDR = 13, OE = 1, WS = 0, CS = 0 | DATA = 13
RAM || 2800, ADDR = 14, OE = 1, WS = 0, CS = 0 | DATA = 14
RAM || 2840, ADDR = 15, OE = 1, WS = 0, CS = 0 | DATA = 15
RAM || 2880, ADDR = 16, OE = 1, WS = 0, CS = 0 | DATA = 16
RAM || 2920, ADDR = 17, OE = 1, WS = 0, CS = 0 | DATA = 17
RAM || 2960, ADDR = 18, OE = 1, WS = 0, CS = 0 | DATA = 18
RAM || 3000, ADDR = 19, OE = 1, WS = 0, CS = 0 | DATA = 19
RAM || 3040, ADDR = 1a, OE = 1, WS = 0, CS = 0 | DATA = 1a
RAM || 3080, ADDR = 1b, OE = 1, WS = 0, CS = 0 | DATA = 1b
RAM || 3120, ADDR = 1c, OE = 1, WS = 0, CS = 0 | DATA = 1c
RAM || 3160, ADDR = 1d, OE = 1, WS = 0, CS = 0 | DATA = 1d
RAM || 3200, ADDR = 1e, OE = 1, WS = 0, CS = 0 | DATA = 1e
RAM || 3240, ADDR = 1f, OE = 1, WS = 0, CS = 0 | DATA = 1f
Finished reading from RAM.
Writing walking ones pattern to RAM.
RAM || 3300, ADDR = 00, OE = 0, WS = 1, CS = 0 | DATA = 00000001
RAM || 3320, ADDR = 00, OE = 0, WS = 0, CS = 0 | DATA = 00000001
RAM || 3360, ADDR = 01, OE = 0, WS = 1, CS = 0 | DATA = 00000010
RAM || 3380, ADDR = 01, OE = 0, WS = 0, CS = 0 | DATA = 00000010
RAM || 3420, ADDR = 02, OE = 0, WS = 1, CS = 0 | DATA = 00000100
RAM || 3440, ADDR = 02, OE = 0, WS = 0, CS = 0 | DATA = 00000100


2. dcd142.ecs.csun.edu (jim777)
RAM || 3480, ADDR = 03, OE = 0, WS = 1, CS = 0 | DATA = 00001000
RAM || 3500, ADDR = 03, OE = 0, WS = 0, CS = 0 | DATA = 00001000
RAM || 3540, ADDR = 04, OE = 0, WS = 1, CS = 0 | DATA = 00010000
RAM || 3560, ADDR = 04, OE = 0, WS = 0, CS = 0 | DATA = 00010000
RAM || 3600, ADDR = 05, OE = 0, WS = 1, CS = 0 | DATA = 00100000
RAM || 3620, ADDR = 05, OE = 0, WS = 0, CS = 0 | DATA = 00100000
RAM || 3660, ADDR = 06, OE = 0, WS = 1, CS = 0 | DATA = 01000000
RAM || 3680, ADDR = 06, OE = 0, WS = 0, CS = 0 | DATA = 01000000
RAM || 3720, ADDR = 07, OE = 0, WS = 1, CS = 0 | DATA = 10000000
RAM || 3740, ADDR = 07, OE = 0, WS = 0, CS = 0 | DATA = 10000000
RAM || 3780, ADDR = 08, OE = 0, WS = 1, CS = 0 | DATA = 00000001
RAM || 3800, ADDR = 08, OE = 0, WS = 0, CS = 0 | DATA = 00000001
RAM || 3840, ADDR = 09, OE = 0, WS = 1, CS = 0 | DATA = 00000010
RAM || 3860, ADDR = 09, OE = 0, WS = 0, CS = 0 | DATA = 00000010
RAM || 3900, ADDR = 0a, OE = 0, WS = 1, CS = 0 | DATA = 00000100
RAM || 3920, ADDR = 0a, OE = 0, WS = 0, CS = 0 | DATA = 00000100
RAM || 3960, ADDR = 0b, OE = 0, WS = 1, CS = 0 | DATA = 00001000
RAM || 3980, ADDR = 0b, OE = 0, WS = 0, CS = 0 | DATA = 00001000
RAM || 4020, ADDR = 0c, OE = 0, WS = 1, CS = 0 | DATA = 00010000
RAM || 4040, ADDR = 0c, OE = 0, WS = 0, CS = 0 | DATA = 00010000
RAM || 4080, ADDR = 0d, OE = 0, WS = 1, CS = 0 | DATA = 00100000
RAM || 4100, ADDR = 0d, OE = 0, WS = 0, CS = 0 | DATA = 00100000
RAM || 4140, ADDR = 0e, OE = 0, WS = 1, CS = 0 | DATA = 01000000
RAM || 4160, ADDR = 0e, OE = 0, WS = 0, CS = 0 | DATA = 01000000
RAM || 4200, ADDR = 0f, OE = 0, WS = 1, CS = 0 | DATA = 10000000
RAM || 4220, ADDR = 0f, OE = 0, WS = 0, CS = 0 | DATA = 10000000
RAM || 4260, ADDR = 10, OE = 0, WS = 1, CS = 0 | DATA = 00000001
RAM || 4280, ADDR = 10, OE = 0, WS = 0, CS = 0 | DATA = 00000001
RAM || 4320, ADDR = 11, OE = 0, WS = 1, CS = 0 | DATA = 00000010
RAM || 4340, ADDR = 11, OE = 0, WS = 0, CS = 0 | DATA = 00000010
RAM || 4380, ADDR = 12, OE = 0, WS = 1, CS = 0 | DATA = 00000100
RAM || 4400, ADDR = 12, OE = 0, WS = 0, CS = 0 | DATA = 00000100
RAM || 4440, ADDR = 13, OE = 0, WS = 1, CS = 0 | DATA = 00001000
RAM || 4460, ADDR = 13, OE = 0, WS = 0, CS = 0 | DATA = 00001000
RAM || 4500, ADDR = 14, OE = 0, WS = 1, CS = 0 | DATA = 00010000
RAM || 4520, ADDR = 14, OE = 0, WS = 0, CS = 0 | DATA = 00010000
RAM || 4560, ADDR = 15, OE = 0, WS = 1, CS = 0 | DATA = 00100000
RAM || 4580, ADDR = 15, OE = 0, WS = 0, CS = 0 | DATA = 00100000
RAM || 4620, ADDR = 16, OE = 0, WS = 1, CS = 0 | DATA = 01000000
RAM || 4640, ADDR = 16, OE = 0, WS = 0, CS = 0 | DATA = 01000000
RAM || 4680, ADDR = 17, OE = 0, WS = 1, CS = 0 | DATA = 10000000
RAM || 4700, ADDR = 17, OE = 0, WS = 0, CS = 0 | DATA = 10000000
RAM || 4740, ADDR = 18, OE = 0, WS = 1, CS = 0 | DATA = 00000001
RAM || 4760, ADDR = 18, OE = 0, WS = 0, CS = 0 | DATA = 00000001
RAM || 4800, ADDR = 19, OE = 0, WS = 1, CS = 0 | DATA = 00000010
RAM || 4820, ADDR = 19, OE = 0, WS = 0, CS = 0 | DATA = 00000010
RAM || 4860, ADDR = 1a, OE = 0, WS = 1, CS = 0 | DATA = 00000100
RAM || 4880, ADDR = 1a, OE = 0, WS = 0, CS = 0 | DATA = 00000100
RAM || 4920, ADDR = 1b, OE = 0, WS = 1, CS = 0 | DATA = 00001000
RAM || 4940, ADDR = 1b, OE = 0, WS = 0, CS = 0 | DATA = 00001000
RAM || 4980, ADDR = 1c, OE = 0, WS = 1, CS = 0 | DATA = 00010000
RAM || 5000, ADDR = 1c, OE = 0, WS = 0, CS = 0 | DATA = 00010000
RAM || 5040, ADDR = 1d, OE = 0, WS = 1, CS = 0 | DATA = 00100000
RAM || 5060, ADDR = 1d, OE = 0, WS = 0, CS = 0 | DATA = 00100000
RAM || 5100, ADDR = 1e, OE = 0, WS = 1, CS = 0 | DATA = 01000000
RAM || 5120, ADDR = 1e, OE = 0, WS = 0, CS = 0 | DATA = 01000000
RAM || 5160, ADDR = 1f, OE = 0, WS = 1, CS = 0 | DATA = 10000000
RAM || 5180, ADDR = 1f, OE = 0, WS = 0, CS = 0 | DATA = 10000000
Finished writing walking ones to RAM.

```
2. dcd142.ecs.csun.edu (jlm7771)
                Reading walking ones from RAM.
RAM ||                  5240, ADDR = 00, OE = 1, WS = 0, CS = 0 | DATA = 00000001
RAM ||                  5280, ADDR = 01, OE = 1, WS = 0, CS = 0 | DATA = 00000010
RAM ||                  5320, ADDR = 02, OE = 1, WS = 0, CS = 0 | DATA = 00000100
RAM ||                  5360, ADDR = 03, OE = 1, WS = 0, CS = 0 | DATA = 00001000
RAM ||                  5400, ADDR = 04, OE = 1, WS = 0, CS = 0 | DATA = 00010000
RAM ||                  5440, ADDR = 05, OE = 1, WS = 0, CS = 0 | DATA = 00100000
RAM ||                  5480, ADDR = 06, OE = 1, WS = 0, CS = 0 | DATA = 01000000
RAM ||                  5520, ADDR = 07, OE = 1, WS = 0, CS = 0 | DATA = 10000000
RAM ||                  5560, ADDR = 08, OE = 1, WS = 0, CS = 0 | DATA = 00000001
RAM ||                  5600, ADDR = 09, OE = 1, WS = 0, CS = 0 | DATA = 00000010
RAM ||                  5640, ADDR = 0a, OE = 1, WS = 0, CS = 0 | DATA = 00000100
RAM ||                  5680, ADDR = 0b, OE = 1, WS = 0, CS = 0 | DATA = 00001000
RAM ||                  5720, ADDR = 0c, OE = 1, WS = 0, CS = 0 | DATA = 00010000
RAM ||                  5760, ADDR = 0d, OE = 1, WS = 0, CS = 0 | DATA = 00100000
RAM ||                  5800, ADDR = 0e, OE = 1, WS = 0, CS = 0 | DATA = 01000000
RAM ||                  5840, ADDR = 0f, OE = 1, WS = 0, CS = 0 | DATA = 10000000
RAM ||                  5880, ADDR = 10, OE = 1, WS = 0, CS = 0 | DATA = 00000001
RAM ||                  5920, ADDR = 11, OE = 1, WS = 0, CS = 0 | DATA = 00000010
RAM ||                  5960, ADDR = 12, OE = 1, WS = 0, CS = 0 | DATA = 00000100
RAM ||                  6000, ADDR = 13, OE = 1, WS = 0, CS = 0 | DATA = 00001000
RAM ||                  6040, ADDR = 14, OE = 1, WS = 0, CS = 0 | DATA = 00010000
RAM ||                  6080, ADDR = 15, OE = 1, WS = 0, CS = 0 | DATA = 00100000
RAM ||                  6120, ADDR = 16, OE = 1, WS = 0, CS = 0 | DATA = 01000000
RAM ||                  6160, ADDR = 17, OE = 1, WS = 0, CS = 0 | DATA = 10000000
RAM ||                  6200, ADDR = 18, OE = 1, WS = 0, CS = 0 | DATA = 00000001
RAM ||                  6240, ADDR = 19, OE = 1, WS = 0, CS = 0 | DATA = 00000010
RAM ||                  6280, ADDR = 1a, OE = 1, WS = 0, CS = 0 | DATA = 00000100
RAM ||                  6320, ADDR = 1b, OE = 1, WS = 0, CS = 0 | DATA = 00001000
RAM ||                  6360, ADDR = 1c, OE = 1, WS = 0, CS = 0 | DATA = 00010000
RAM ||                  6400, ADDR = 1d, OE = 1, WS = 0, CS = 0 | DATA = 00100000
RAM ||                  6440, ADDR = 1e, OE = 1, WS = 0, CS = 0 | DATA = 01000000
RAM ||                  6480, ADDR = 1f, OE = 1, WS = 0, CS = 0 | DATA = 10000000
                Finished reading walking ones from RAM.


                Showing disabled state.
RAM ||                  6520, ADDR = 1f, OE = 1, WS = 1, CS = 1 | DATA = zzzzzzzz
RAM ||                  6540, ADDR = 1f, OE = 1, WS = 0, CS = 1 | DATA = zzzzzzzz
RAM ||                  6560, ADDR = 00, OE = 0, WS = 0, CS = 1 | DATA = zzzzzzzz
RAM ||                  6580, ADDR = 00, OE = 0, WS = 0, CS = 1 | DATA = zzzzzzzz
                RAM test finished.


ROM test start.
                Reading from ROM.
ROM ||                  7040, ADDR = 00, OE = 1, CS = 0 | DATA = xx
ROM ||                  7080, ADDR = 01, OE = 1, CS = 0 | DATA = xx
ROM ||                  7120, ADDR = 02, OE = 1, CS = 0 | DATA = xx
ROM ||                  7160, ADDR = 03, OE = 1, CS = 0 | DATA = xx
ROM ||                  7200, ADDR = 04, OE = 1, CS = 0 | DATA = 58
ROM ||                  7240, ADDR = 05, OE = 1, CS = 0 | DATA = ed
ROM ||                  7280, ADDR = 06, OE = 1, CS = 0 | DATA = b7
ROM ||                  7320, ADDR = 07, OE = 1, CS = 0 | DATA = 34
ROM ||                  7360, ADDR = 08, OE = 1, CS = 0 | DATA = c9
ROM ||                  7400, ADDR = 09, OE = 1, CS = 0 | DATA = 8f
ROM ||                  7440, ADDR = 0a, OE = 1, CS = 0 | DATA = a0
ROM ||                  7480, ADDR = 0b, OE = 1, CS = 0 | DATA = 9b
ROM ||                  7520, ADDR = 0c, OE = 1, CS = 0 | DATA = 65
```

```
2. dcd142.ecs.csun.edu (jlm7771)
ROM ||                7560, ADDR = 0d, OE = 1, CS = 0 | DATA = 11
ROM ||                7600, ADDR = 0e, OE = 1, CS = 0 | DATA = 03
ROM ||                7640, ADDR = 0f, OE = 1, CS = 0 | DATA = 4c
ROM ||                7680, ADDR = 10, OE = 1, CS = 0 | DATA = da
ROM ||                7720, ADDR = 11, OE = 1, CS = 0 | DATA = 7e
ROM ||                7760, ADDR = 12, OE = 1, CS = 0 | DATA = f2
ROM ||                7800, ADDR = 13, OE = 1, CS = 0 | DATA = 26
ROM ||                7840, ADDR = 14, OE = 1, CS = 0 | DATA = 86
ROM ||                7880, ADDR = 15, OE = 1, CS = 0 | DATA = 95
ROM ||                7920, ADDR = 16, OE = 1, CS = 0 | DATA = fd
ROM ||                7960, ADDR = 17, OE = 1, CS = 0 | DATA = b1
ROM ||                8000, ADDR = 18, OE = 1, CS = 0 | DATA = xx
ROM ||                8040, ADDR = 19, OE = 1, CS = 0 | DATA = xx
ROM ||                8080, ADDR = 1a, OE = 1, CS = 0 | DATA = xx
ROM ||                8120, ADDR = 1b, OE = 1, CS = 0 | DATA = xx
ROM ||                8160, ADDR = 1c, OE = 1, CS = 0 | DATA = 12
ROM ||                8200, ADDR = 1d, OE = 1, CS = 0 | DATA = af
ROM ||                8240, ADDR = 1e, OE = 1, CS = 0 | DATA = 33
ROM ||                8280, ADDR = 1f, OE = 1, CS = 0 | DATA = xx
            Finished reading from ROM.


            Scrambling ROM contents into RAM.
RAM ||                8360, ADDR = 00, OE = 0, WS = 1, CS = 0 | DATA = xx
RAM ||                8380, ADDR = 00, OE = 0, WS = 0, CS = 0 | DATA = xx
RAM ||                8440, ADDR = 01, OE = 0, WS = 1, CS = 0 | DATA = xx
RAM ||                8460, ADDR = 01, OE = 0, WS = 0, CS = 0 | DATA = xx
RAM ||                8520, ADDR = 02, OE = 0, WS = 1, CS = 0 | DATA = xx
RAM ||                8540, ADDR = 02, OE = 0, WS = 0, CS = 0 | DATA = xx
RAM ||                8600, ADDR = 03, OE = 0, WS = 1, CS = 0 | DATA = xx
RAM ||                8620, ADDR = 03, OE = 0, WS = 0, CS = 0 | DATA = xx
RAM ||                8680, ADDR = 04, OE = 0, WS = 1, CS = 0 | DATA = 13
RAM ||                8700, ADDR = 04, OE = 0, WS = 0, CS = 0 | DATA = 13
RAM ||                8760, ADDR = 05, OE = 0, WS = 1, CS = 0 | DATA = de
RAM ||                8780, ADDR = 05, OE = 0, WS = 0, CS = 0 | DATA = de
RAM ||                8840, ADDR = 06, OE = 0, WS = 1, CS = 0 | DATA = ed
RAM ||                8860, ADDR = 06, OE = 0, WS = 0, CS = 0 | DATA = ed
RAM ||                8920, ADDR = 07, OE = 0, WS = 1, CS = 0 | DATA = 0d
RAM ||                8940, ADDR = 07, OE = 0, WS = 0, CS = 0 | DATA = 0d
RAM ||                9000, ADDR = 08, OE = 0, WS = 1, CS = 0 | DATA = d2
RAM ||                9020, ADDR = 08, OE = 0, WS = 0, CS = 0 | DATA = d2
RAM ||                9080, ADDR = 09, OE = 0, WS = 1, CS = 0 | DATA = ea
RAM ||                9100, ADDR = 09, OE = 0, WS = 0, CS = 0 | DATA = ea
RAM ||                9160, ADDR = 0a, OE = 0, WS = 1, CS = 0 | DATA = 44
RAM ||                9180, ADDR = 0a, OE = 0, WS = 0, CS = 0 | DATA = 44
RAM ||                9240, ADDR = 0b, OE = 0, WS = 1, CS = 0 | DATA = e3
RAM ||                9260, ADDR = 0b, OE = 0, WS = 0, CS = 0 | DATA = e3
RAM ||                9320, ADDR = 0c, OE = 0, WS = 1, CS = 0 | DATA = 9c
RAM ||                9340, ADDR = 0c, OE = 0, WS = 0, CS = 0 | DATA = 9c
RAM ||                9400, ADDR = 0d, OE = 0, WS = 1, CS = 0 | DATA = 81
RAM ||                9420, ADDR = 0d, OE = 0, WS = 0, CS = 0 | DATA = 81
RAM ||                9480, ADDR = 0e, OE = 0, WS = 1, CS = 0 | DATA = a0
RAM ||                9500, ADDR = 0e, OE = 0, WS = 0, CS = 0 | DATA = a0
RAM ||                9560, ADDR = 0f, OE = 0, WS = 1, CS = 0 | DATA = 1a
RAM ||                9580, ADDR = 0f, OE = 0, WS = 0, CS = 0 | DATA = 1a
RAM ||                9640, ADDR = 10, OE = 0, WS = 1, CS = 0 | DATA = 73
RAM ||                9660, ADDR = 10, OE = 0, WS = 0, CS = 0 | DATA = 73
RAM ||                9720, ADDR = 11, OE = 0, WS = 1, CS = 0 | DATA = 3f
RAM ||                9740, ADDR = 11, OE = 0, WS = 0, CS = 0 | DATA = 3f
```

2. dcd142.ecs.csun.edu (jlm7771)

```
RAM ||                9800, ADDR = 12, OE = 0, WS = 1, CS = 0 | DATA = 75
RAM ||                9820, ADDR = 12, OE = 0, WS = 0, CS = 0 | DATA = 75
RAM ||                9880, ADDR = 13, OE = 0, WS = 1, CS = 0 | DATA = 2c
RAM ||                9900, ADDR = 13, OE = 0, WS = 0, CS = 0 | DATA = 2c
RAM ||                9960, ADDR = 14, OE = 0, WS = 1, CS = 0 | DATA = 68
RAM ||                9980, ADDR = 14, OE = 0, WS = 0, CS = 0 | DATA = 68
RAM ||               10040, ADDR = 15, OE = 0, WS = 1, CS = 0 | DATA = c9
RAM ||               10060, ADDR = 15, OE = 0, WS = 0, CS = 0 | DATA = c9
RAM ||               10120, ADDR = 16, OE = 0, WS = 1, CS = 0 | DATA = df
RAM ||               10140, ADDR = 16, OE = 0, WS = 0, CS = 0 | DATA = df
RAM ||               10200, ADDR = 17, OE = 0, WS = 1, CS = 0 | DATA = c5
RAM ||               10220, ADDR = 17, OE = 0, WS = 0, CS = 0 | DATA = c5
RAM ||               10280, ADDR = 18, OE = 0, WS = 1, CS = 0 | DATA = xx
RAM ||               10300, ADDR = 18, OE = 0, WS = 0, CS = 0 | DATA = xx
RAM ||               10360, ADDR = 19, OE = 0, WS = 1, CS = 0 | DATA = xx
RAM ||               10380, ADDR = 19, OE = 0, WS = 0, CS = 0 | DATA = xx
RAM ||               10440, ADDR = 1a, OE = 0, WS = 1, CS = 0 | DATA = xx
RAM ||               10460, ADDR = 1a, OE = 0, WS = 0, CS = 0 | DATA = xx
RAM ||               10520, ADDR = 1b, OE = 0, WS = 1, CS = 0 | DATA = xx
RAM ||               10540, ADDR = 1b, OE = 0, WS = 0, CS = 0 | DATA = xx
RAM ||               10600, ADDR = 1c, OE = 0, WS = 1, CS = 0 | DATA = 21
RAM ||               10620, ADDR = 1c, OE = 0, WS = 0, CS = 0 | DATA = 21
RAM ||               10680, ADDR = 1d, OE = 0, WS = 1, CS = 0 | DATA = ee
RAM ||               10700, ADDR = 1d, OE = 0, WS = 0, CS = 0 | DATA = ee
RAM ||               10760, ADDR = 1e, OE = 0, WS = 1, CS = 0 | DATA = a5
RAM ||               10780, ADDR = 1e, OE = 0, WS = 0, CS = 0 | DATA = a5
RAM ||               10840, ADDR = 1f, OE = 0, WS = 1, CS = 0 | DATA = xx
RAM ||               10860, ADDR = 1f, OE = 0, WS = 0, CS = 0 | DATA = xx
                Finished scrambling contents from ROM to RAM.


                Reading scrambled contents from RAM.
RAM ||               10920, ADDR = 00, OE = 1, WS = 0, CS = 0 | DATA = xx
RAM ||               10960, ADDR = 01, OE = 1, WS = 0, CS = 0 | DATA = xx
RAM ||               11000, ADDR = 02, OE = 1, WS = 0, CS = 0 | DATA = xx
RAM ||               11040, ADDR = 03, OE = 1, WS = 0, CS = 0 | DATA = xx
RAM ||               11080, ADDR = 04, OE = 1, WS = 0, CS = 0 | DATA = 13
RAM ||               11120, ADDR = 05, OE = 1, WS = 0, CS = 0 | DATA = de
RAM ||               11160, ADDR = 06, OE = 1, WS = 0, CS = 0 | DATA = ed
RAM ||               11200, ADDR = 07, OE = 1, WS = 0, CS = 0 | DATA = 0d
RAM ||               11240, ADDR = 08, OE = 1, WS = 0, CS = 0 | DATA = d2
RAM ||               11280, ADDR = 09, OE = 1, WS = 0, CS = 0 | DATA = ea
RAM ||               11320, ADDR = 0a, OE = 1, WS = 0, CS = 0 | DATA = 44
RAM ||               11360, ADDR = 0b, OE = 1, WS = 0, CS = 0 | DATA = e3
RAM ||               11400, ADDR = 0c, OE = 1, WS = 0, CS = 0 | DATA = 9c
RAM ||               11440, ADDR = 0d, OE = 1, WS = 0, CS = 0 | DATA = 81
RAM ||               11480, ADDR = 0e, OE = 1, WS = 0, CS = 0 | DATA = a0
RAM ||               11520, ADDR = 0f, OE = 1, WS = 0, CS = 0 | DATA = 1a
RAM ||               11560, ADDR = 10, OE = 1, WS = 0, CS = 0 | DATA = 73
RAM ||               11600, ADDR = 11, OE = 1, WS = 0, CS = 0 | DATA = 3f
RAM ||               11640, ADDR = 12, OE = 1, WS = 0, CS = 0 | DATA = 75
RAM ||               11680, ADDR = 13, OE = 1, WS = 0, CS = 0 | DATA = 2c
RAM ||               11720, ADDR = 14, OE = 1, WS = 0, CS = 0 | DATA = 68
RAM ||               11760, ADDR = 15, OE = 1, WS = 0, CS = 0 | DATA = c9
RAM ||               11800, ADDR = 16, OE = 1, WS = 0, CS = 0 | DATA = df
RAM ||               11840, ADDR = 17, OE = 1, WS = 0, CS = 0 | DATA = c5
RAM ||               11880, ADDR = 18, OE = 1, WS = 0, CS = 0 | DATA = xx
RAM ||               11920, ADDR = 19, OE = 1, WS = 0, CS = 0 | DATA = xx
RAM ||               11960, ADDR = 1a, OE = 1, WS = 0, CS = 0 | DATA = xx
```

```
RAM ||                12000, ADDR = 1b, OE = 1, WS = 0, CS = 0 | DATA = xx
RAM ||                12040, ADDR = 1c, OE = 1, WS = 0, CS = 0 | DATA = 21
RAM ||                12080, ADDR = 1d, OE = 1, WS = 0, CS = 0 | DATA = ee
RAM ||                12120, ADDR = 1e, OE = 1, WS = 0, CS = 0 | DATA = a5
RAM ||                12160, ADDR = 1f, OE = 1, WS = 0, CS = 0 | DATA = xx
              Finished reading scrambled contents from RAM.


End of ROM test.


$finish called from file "lab7_rom_tb.v", line 97.
$finish at simulation time                121800
           V C S   S i m u l a t i o n   R e p o r t
Time: 12180000 ps
CPU Time:      0.220 seconds;       Data structure size:   0.0Mb
Fri Apr  9 20:13:32 2021
$
```

**Fig. 3** simv output

**Fig. 4** RAM waveforms

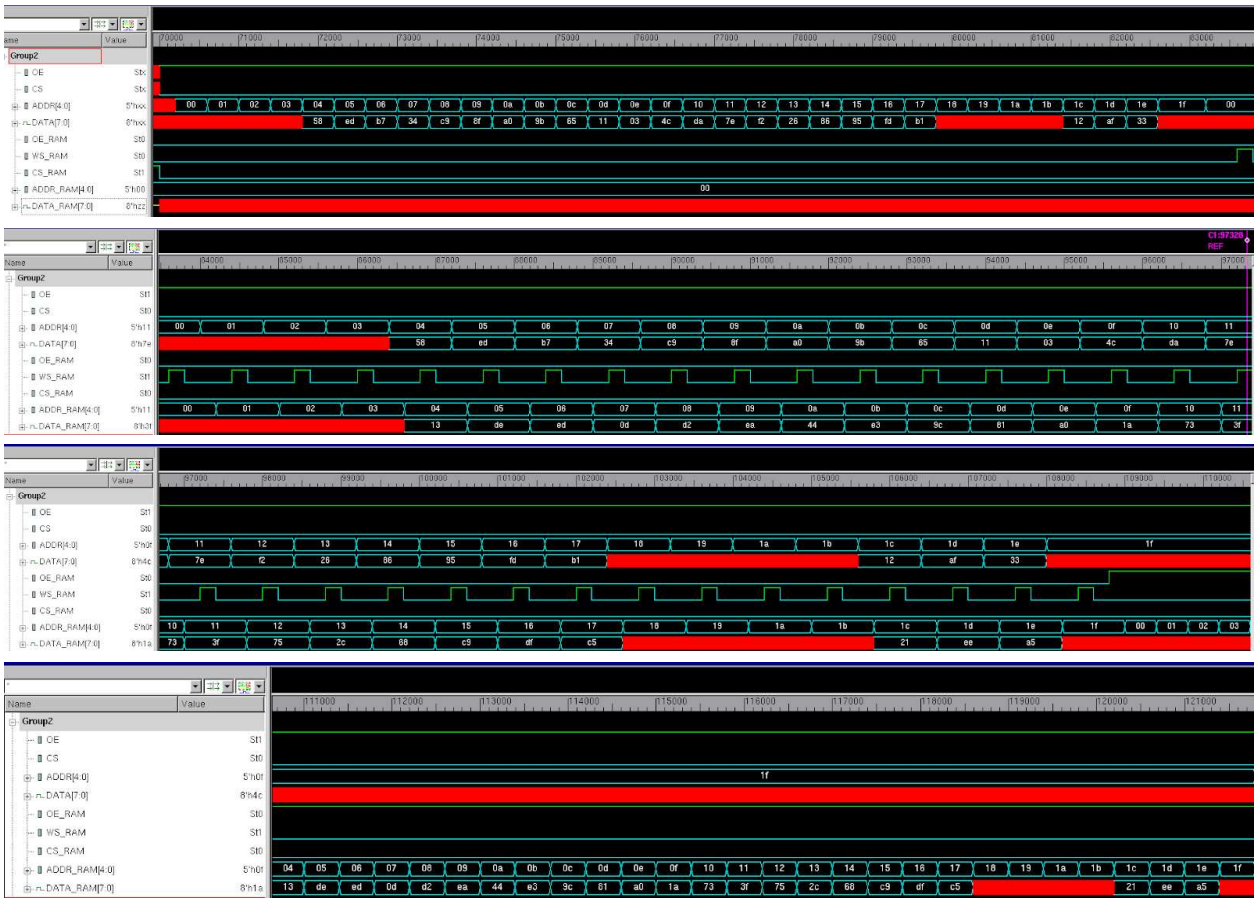


**Fig. 5** ROM waveforms

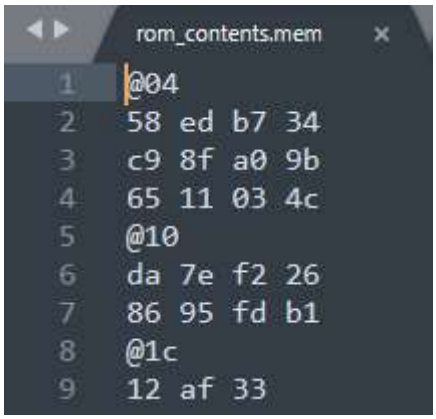



**Fig. 6** ram_contents.mem

reg_file_ram.v | rom_contents.mem | compileV.f | lab7_rom_tb.v | reg_file_r

```
/*********************************************************************************
***                                                                            ***
*** ECE 526 L Experiment #7             Jose Luis Martinez, Spring, 2021     ***
***                                                                            ***
*** Experiment 7 - Register File Models                                       ***
***                                                                            ***
*********************************************************************************
*** Filename: reg_file_ram.v   Created by Jose Luis Martinez, April 7, 2021 ***
***                                                                            ***
*********************************************************************************/

`timescale 1ns/100ps

module reg_file_ram(ADDR, OE, WS, CS, DATA);
    parameter WIDTH = 8;
    parameter DEPTH = 5;

    input OE, WS, CS;
    input [DEPTH-1:0]ADDR;
    inout [WIDTH-1:0]DATA;

    reg [WIDTH-1:0]RAM[0:(2**DEPTH)-1];

    assign DATA = (OE & !CS) ? RAM[ADDR] : {WIDTH{1'bz}};

    always@(posedge WS) begin
        if (!OE & !CS) begin
            RAM[ADDR] <= DATA;
        end
    end

endmodule
```

**Fig. 7** reg_file_ram.v

reg_file_ram.v | reg_file_rom.v | rom_contents.mem | compileV.f | lab7_ram

```
/*********************************************************************
***                                                                ***
*** ECE 526 L Experiment #7             Jose Luis Martinez, Spring, 2021    ***
***                                                                ***
*** Experiment 7 - Register File Models                            ***
***                                                                ***
*********************************************************************
*** Filename: reg_file_rom.v    Created by Jose Luis Martinez, April 7, 2021***
***                                                                ***
*********************************************************************/

`timescale 1ns/100ps

module reg_file_rom(ADDR, OE, CS, DATA);
    parameter WIDTH = 8;
    parameter DEPTH = 5;

    input OE, CS;
    input [DEPTH-1:0]ADDR;
    output [WIDTH-1:0]DATA;

    reg [WIDTH-1:0]ROM[0:(2**DEPTH)-1];

    initial begin
        $display("Loading ROM contents.");
        $readmemh("rom_contents.mem", ROM, 0, (2**DEPTH)-1);
    end

    assign DATA = OE ? ROM[ADDR] : {WIDTH{1'bz}};

endmodule
```

**Fig. 8** reg_file_rom.v

reg_file_ram.v | reg_file_rom.v | rom_contents.mem | compileV.f | lab7_ram_

```
/*****************************************************************************
***                                                                        ***
*** ECE 526 L Experiment #7             Jose Luis Martinez, Spring, 2021    ***
***                                                                        ***
*** Experiment 7 - Register File Models                                    ***
***                                                                        ***
*****************************************************************************
*** Filename: lab7_ram_tb.v    Created by Jose Luis Martinez, April 7, 2021 ***
***                                                                        ***
*****************************************************************************/

`timescale 1ns/100ps

`define WIDTH_TB 8
`define DEPTH_TB 5
`define CLK_PER 20
`define STRING "RAM || %d, ADDR = %h, OE = %b, WS = %b, CS = %b | DATA = %h"
`define STRINGB "RAM || %d, ADDR = %h, OE = %b, WS = %b, CS = %b | DATA = %b"

module lab7_ram_tb();
    reg OE, WS, CS;
    reg [`DEPTH_TB-1:0]ADDR;
    wire [`WIDTH_TB-1:0]DATA;
    reg [`WIDTH_TB-1:0]DATA_TB;

    reg [`WIDTH_TB-1:0]LOOP;

    reg_file_ram ram1(.ADDR(ADDR), .OE(OE), .WS(WS), .CS(CS), .DATA(DATA));

    assign DATA = (!OE & !CS) ? DATA_TB : 8'bz;

    initial begin
        $vcdpluson;
        OE <= 0;
        WS <= 0;
        CS <= 0;
        ADDR <= 0;
        #(`CLK_PER)
        $display("Ram test start.",);
        $display("\t\tWriting to RAM.");

        for (LOOP = 0; LOOP < 32; LOOP = LOOP+1) begin
            #(`CLK_PER)
            ADDR <= LOOP;
            DATA_TB <= LOOP;
            #(`CLK_PER)
            WS <= 1;
            $strobe(`STRING, $time, ADDR, OE, WS, CS, DATA);
            #(`CLK_PER)
```

```
50          WS<= 0;
51          $strobe(`STRING, $time, ADDR, OE, WS, CS, DATA);
52      end
53
54      #(`CLK_PER)
55      $display("\t\tFinished writing to RAM.\n\n");
56      $display("\t\tReading from RAM.");
57
58      OE <= 1;
59
60      for (LOOP = 0; LOOP < 32; LOOP = LOOP+1) begin
61          #(`CLK_PER)
62          ADDR <= LOOP;
63          #(`CLK_PER)
64          $strobe(`STRING, $time, ADDR, OE, WS, CS, DATA);
65      end
66      #(`CLK_PER)
67      $display("\t\tFinished reading from RAM.\n\n");
68      $display("\t\tWriting walking ones pattern to RAM.");
69
70      DATA_TB <= `WIDTH_TB'b1;
71      OE <= 0;
72
73      for (LOOP = 0; LOOP < 32; LOOP = LOOP+1) begin
74          #(`CLK_PER)
75
76          if (!LOOP) begin
77              ADDR <= LOOP;
78              DATA_TB <= `WIDTH_TB'b1;
79          end else begin
80              ADDR <= LOOP;
81              DATA_TB <= {DATA_TB[`WIDTH_TB-2:0], DATA_TB[`WIDTH_TB-1]};
82          end
83          #(`CLK_PER)
84          WS <= 1;
85          $strobe(`STRINGB, $time, ADDR, OE, WS, CS, DATA);
86          #(`CLK_PER)
87          WS<= 0;
88          $strobe(`STRINGB, $time, ADDR, OE, WS, CS, DATA);
89      end
90
91      #(`CLK_PER)
92      $display("\t\tFinished writing walking ones to RAM.\n\n");
93      $display("\t\tReading walking ones from RAM.");
94
95      OE <= 1;
96
97      for (LOOP = 0; LOOP < 32; LOOP = LOOP+1) begin
98          #(`CLK_PER)
```

```
 99              ADDR <= LOOP;
100              #(`CLK_PER)
101              $strobe(`STRINGB, $time, ADDR, OE, WS, CS, DATA);
102          end
103          #(`CLK_PER)
104          $display("\t\tFinished reading walking ones from RAM.\n\n");
105
106          $display("\t\tShowing disabled state.");
107          CS <= 1;
108          #(`CLK_PER)
109          WS <= 1;
110          $strobe(`STRINGB, $time, ADDR, OE, WS, CS, DATA);
111          #(`CLK_PER)
112          WS <= 0;
113          $strobe(`STRINGB, $time, ADDR, OE, WS, CS, DATA);
114          #(`CLK_PER)
115          OE <= 0;
116          ADDR <= 0;
117          $strobe(`STRINGB, $time, ADDR, OE, WS, CS, DATA);
118          #(`CLK_PER)
119          $strobe(`STRINGB, $time, ADDR, OE, WS, CS, DATA);
120          #(`CLK_PER)
121          $display("\t\tRAM test finished.\n\n");
122      end
123
124  endmodule;
```

**Fig. 9** lab7_ram_tb.v

lab7_rom_tb.v | reg_file_ram.v | reg_file_rom.v | rom_contents.mem | compileV.f | lab7_

```
/*********************************************************************
***                                                                ***
*** ECE 526 L Experiment #7            Jose Luis Martinez, Spring, 2021    ***
***                                                                ***
*** Experiment 7 - Register File Models                            ***
***                                                                ***
*********************************************************************
*** Filename: lab7_rom_tb.v    Created by Jose Luis Martinez, April 7, 2021 ***
***                                                                ***
*********************************************************************/

`timescale 1ns/100ps

`define WIDTH_TB 8
`define DEPTH_TB 5
`define CLK_PER 20
`define STRINGR "ROM || %d, ADDR = %h, OE = %b, CS = %b | DATA = %h"

module lab7_rom_tb();
    reg OE, CS;
    reg [`DEPTH_TB-1:0]ADDR;
    wire [`WIDTH_TB-1:0]DATA;

    reg OE_RAM, WS_RAM, CS_RAM;
    reg [`DEPTH_TB-1:0]ADDR_RAM;
    wire [`WIDTH_TB-1:0]DATA_RAM;
    reg [`WIDTH_TB-1:0]DATA_TB;

    reg [`WIDTH_TB-1:0]LOOP;

    reg_file_rom rom1(.ADDR(ADDR), .OE(OE), .CS(CS), .DATA(DATA));
    reg_file_ram ram2(.ADDR(ADDR_RAM), .OE(OE_RAM), .WS(WS_RAM), .CS(CS_RAM), .DATA(DATA_RAM));

    assign DATA_RAM = (!OE_RAM & !CS_RAM) ? DATA_TB : 8'bz;

    initial begin
        OE_RAM <= 0;
        WS_RAM <= 0;
        CS_RAM <= 1;
        ADDR_RAM <= 0;
        #(7000)
        $vcdpluson;

        $display("ROM test start.");
        OE <= 1;
        CS <= 0;
        CS_RAM <= 0;
        $display("\t\tReading from ROM.");

```

```
        for (LOOP = 0; LOOP < 32; LOOP = LOOP+1) begin
            #(`CLK_PER)
            ADDR <= LOOP;
            #(`CLK_PER)
            $strobe(`STRINGR, $time, ADDR, OE, CS, DATA);
        end
        #(`CLK_PER)
        $display("\t\tFinished reading from ROM.\n\n");

        $display("\t\tScrambling ROM contents into RAM.");
        for (LOOP = 0; LOOP < 32; LOOP = LOOP+1) begin
            #(`CLK_PER)
            ADDR <= LOOP;
            #(`CLK_PER)
            DATA_TB[7] <= DATA[0];
            DATA_TB[6] <= DATA[7];
            DATA_TB[5] <= DATA[1];
            DATA_TB[4] <= DATA[6];
            DATA_TB[3] <= DATA[2];
            DATA_TB[2] <= DATA[5];
            DATA_TB[1] <= DATA[3];
            DATA_TB[0] <= DATA[4];
            ADDR_RAM <= LOOP;
            #(`CLK_PER)
            WS_RAM <= 1;
            $strobe(`STRING, $time, ADDR_RAM, OE_RAM, WS_RAM, CS_RAM, DATA_RAM);
            #(`CLK_PER)
            WS_RAM <= 0;
            $strobe(`STRING, $time, ADDR_RAM, OE_RAM, WS_RAM, CS_RAM, DATA_RAM);
        end

        #(`CLK_PER)
        $display("\t\tFinished scrambling contents from ROM to RAM.\n\n");

        $display("\t\tReading scrambled contents from RAM.");

        OE_RAM <= 1;

        for (LOOP = 0; LOOP < 32; LOOP = LOOP+1) begin
            #(`CLK_PER)
            ADDR_RAM <= LOOP;
            #(`CLK_PER)
            $strobe(`STRING, $time, ADDR_RAM, OE_RAM, WS_RAM, CS_RAM, DATA_RAM);
        end
        #(`CLK_PER)
        $display("\t\tFinished reading scrambled contents from RAM.\n\n");
        $display("End of ROM test.\n\n");
        $finish;
    end


endmodule
```

**Fig. 10** lab7_rom_tb.v

## Analysis

For our RAM module we conducted several tests in order to verify its functionality. We first wrote to every single address the value of the address so 0x00 = 0x00, 0x01 = 0x01, 0x02 = 0x02, …, 0x1f = 0x1f. We then verified that we wrote those values to the RAM by reading from every address. We also needed to verify that our output bits are capable of operating independently so a Walking Ones test can help verify that. So the walking ones test was applied by setting the first address of the RAM to 0x01 and shifting left. As we can see from **Fig. 3** and **Fig. 4** we are able to verify that our RAM module output bits are able to operate independently.

For our ROM module we also conducted a test to verify if the module was functioning correctly. The ROM module should load its data from a file called rom_contents.mem and to check if it has dones so i used a for loop and went through every address and printed its values. We should see in **Fig. 3**, **Fig. 5,** and **Fig. 6** that the data was loaded correctly.

Another test was done by combining both register file modules into one test. This test was to read the contents from the ROM module, scramble the bit order of each byte by [7654 3210] to [0716 2534], and store them into the RAM module. As we can see in **Fig. 3** and **Fig. 5** the scrambling algorithm is implemented and we can look at the particular address of 0x10. Our value should be 0xda but our RAM module shows that there is a value of 0x73 meaning that our scrambler is working correctly.

## Conclusion

In conclusion, a RAM and ROM module were created using register files and used parameters so they can be scalable. I also performed various tests like writing/reading form addresses, a walking ones test, and a scrambling algorithm. I was able to perform the tasks provided by this lab and was able to verify that it is working with the aforementioned tests. This lab has taught me about how to use $readmemb & $readmemh and how to create memories using register files.

Academic Dishonesty

Submitting any report that is not entirely your own work is a form of academic dishonesty and will not be tolerated. Each and every lab report must include the following statement, signed and dated by the student. Lab reports without the statement will be summarily rejected.

I hereby attest that this lab report is entirely my own work. I have not copied either code or text from anyone, nor have I allowed or will I allow anyone to copy my work.

Name (printed) Jose L Martinez

Name (signed) Jose Martinez Date 4/11/2021