Fall 2019

California State University, Northridge

Department of Electrical and Computer Engineering

ECE 526L

Experiment # 7
Register File Models

October 31, 2019
SUBMITTED: November 19, 2019

Authors: Ridge Tejuco
Professor: Ronald Mehler Ph.D

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) _____

Name (signed) _____

Date _____

Analysis of RAM register file

Starting at 2 ns, the Write Strobe is continuously changed to start sequentially writing values in the data line to the registers. Looking closely at the registers at the bottom of Figure 1. The registers are addresses are filled and changed at every Write strobe. Figure 2 shows the completely filled register file at 64 ns.

Figure 1. Write strobe

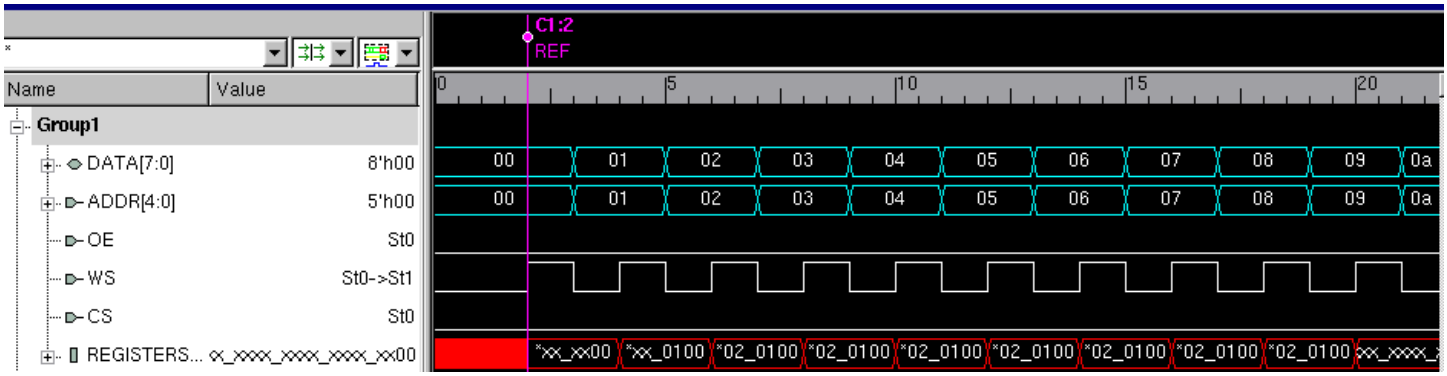
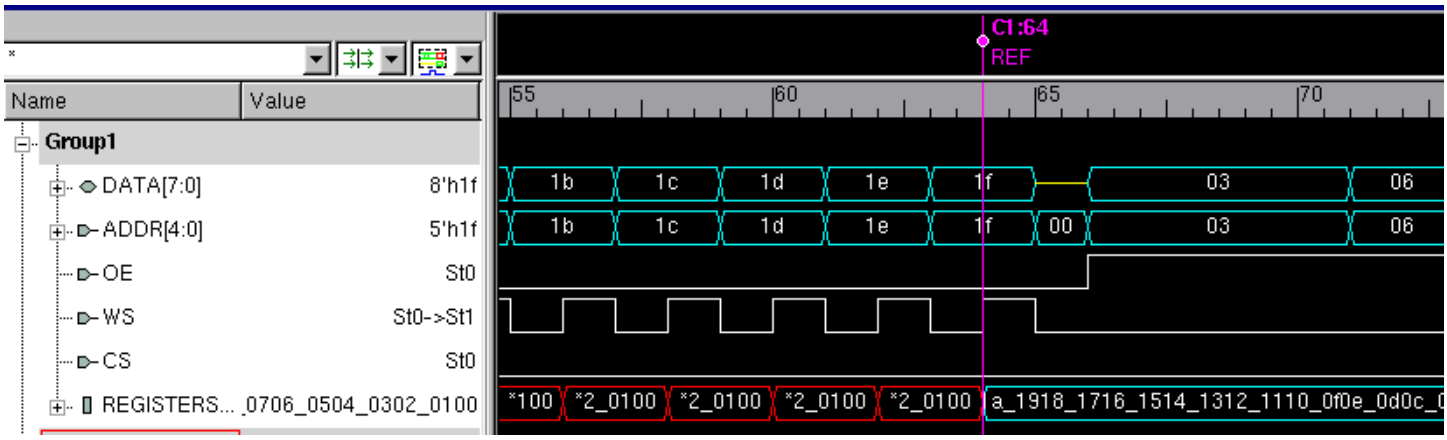
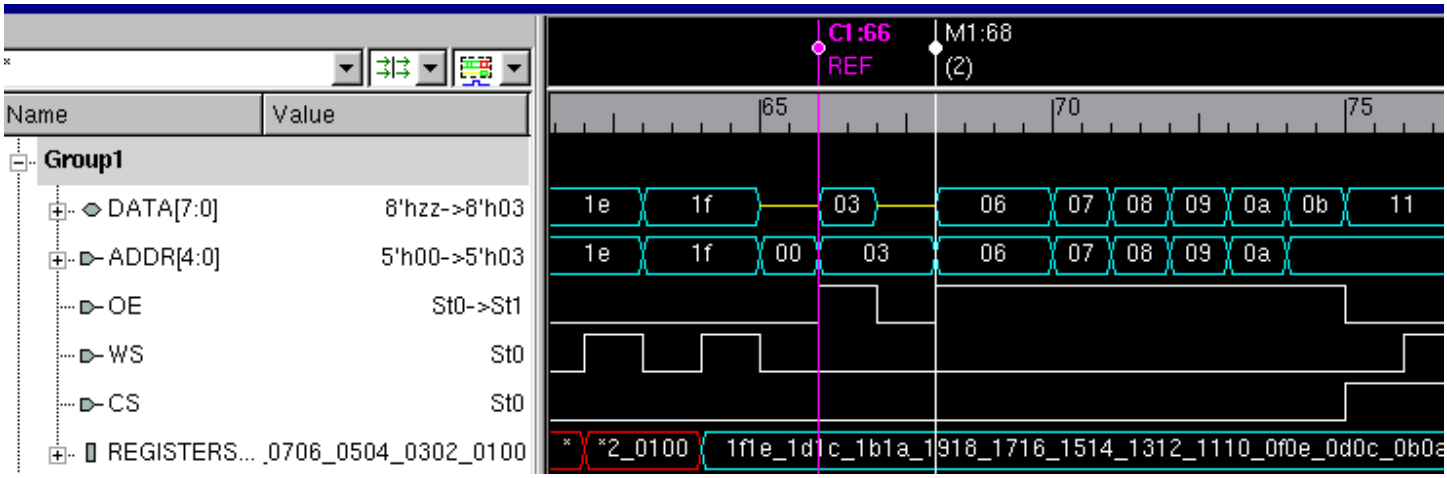


Figure 2. Fully loaded register file



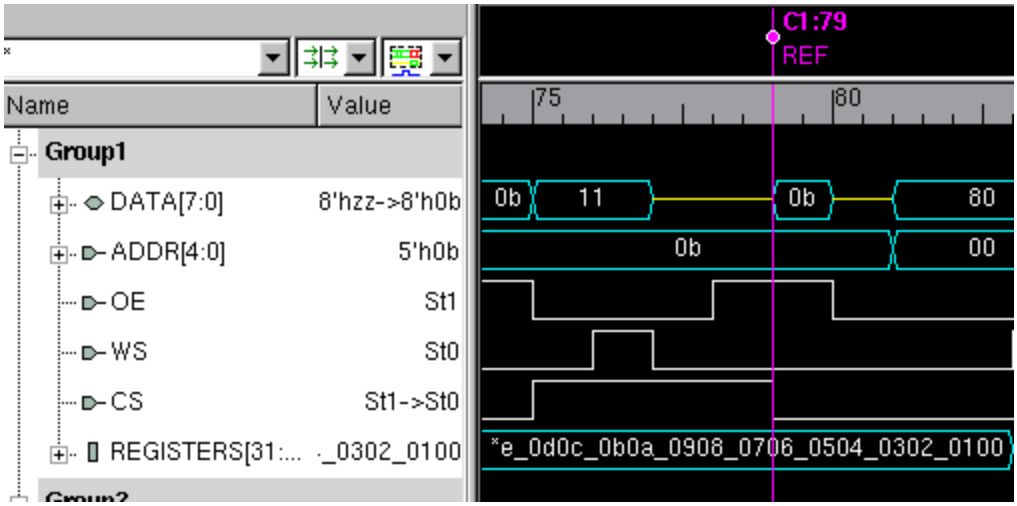
After loading is complete, the input to the data line is set to high z. The Output Enabled is set at 66 ns. Figure 3 shows an individual read at 66 ns and a block read at 68 ns.

Figure 3. Output enable



In figure 4, the Chip Select was tested by setting CS high then trying to write the value 0x11 to the address 0x0b. When the address 0xb was read while CS was still set high, a value of high impedance was returned. When read again while CS was set low, the returned value was 0x0b at address 0x0b. This was the initial value set during the sequential load, which verifies that WS or OE did not work while CS was high.

Figure 4. Chip select



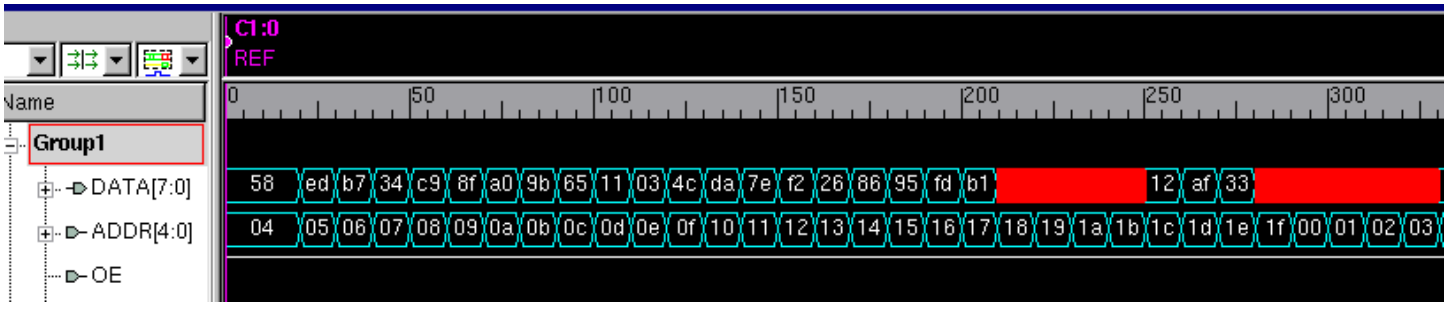
A walking 1 was written to addresses 0x00 to 0x07 at 83 ns. These values were subsequently read at 99 ns.

Figure 5. Walking ones

Analysis of ROM

In Figure 6, the memory of the rom was checked starting at 0 ns. The address started at 4 and was incremented every nano second. Looking closely, the output at each address matches the specification in the lab.

Figure 6. Reading the ROM

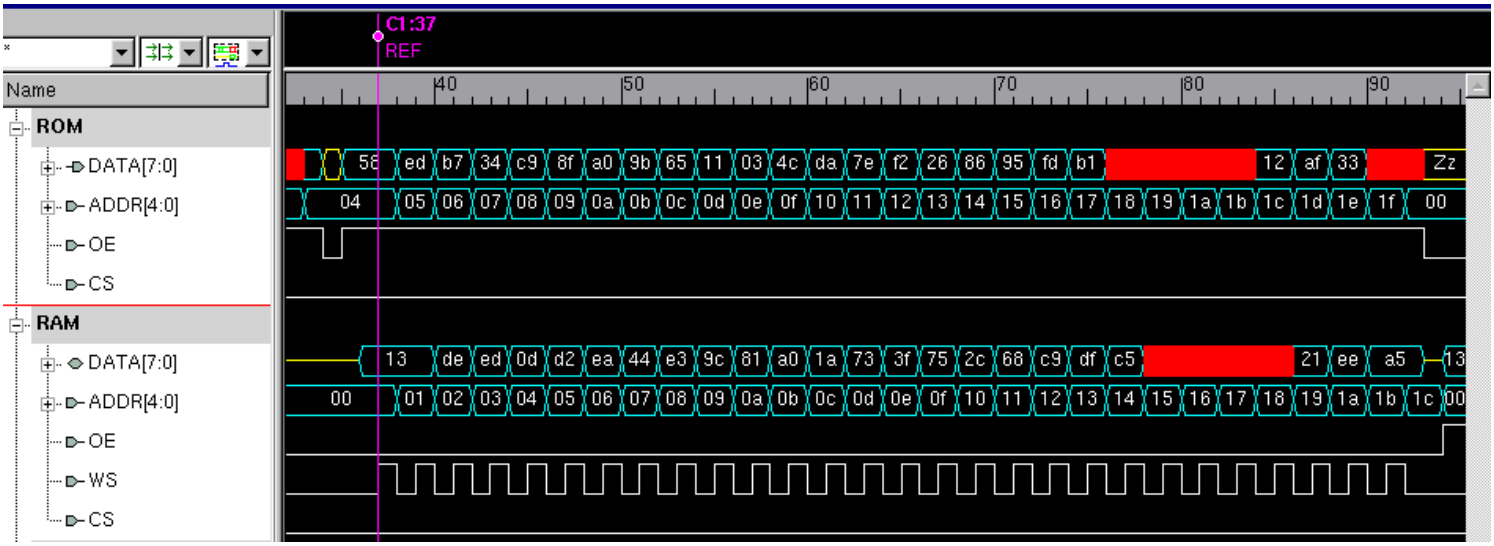


In Figure 7, the values read from the ROM ,starting at address 5'h04, are scrambled then stored in RAM starting at address 5'h00 of the RAM. The scrambling algorithm for the first few memory values were checked by the following.

	7654 3210		0716 2534	
0x58.)	0101 1000	----->	0001 0011	which equals 0x13.
0xED.)	1110 1101	----->	1101 1110	which equals 0xDE.
0xB7.)	1011 0111	----->	1110 1101	which equals 0xED.

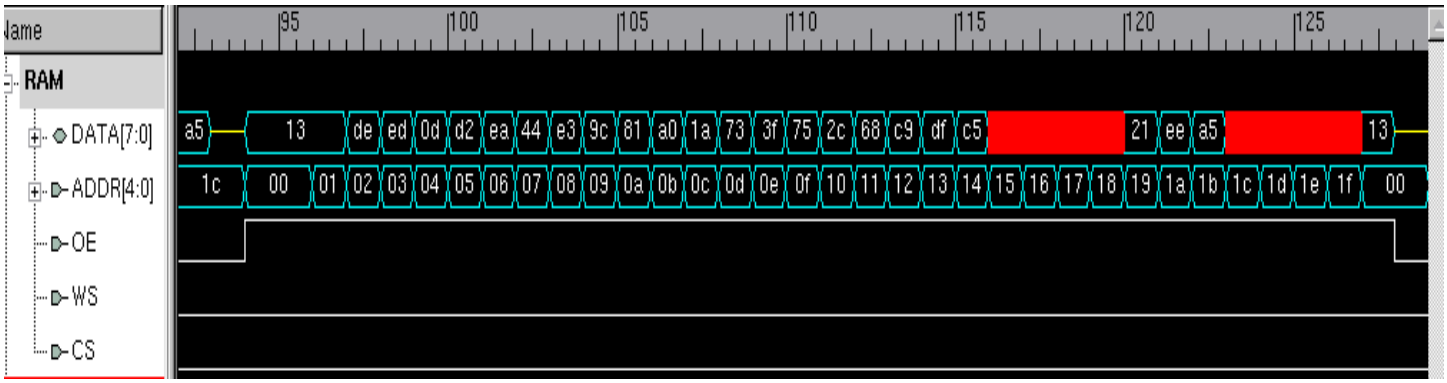
A quick but not infallible way to check these values is to check the number of 1's in each value. Both the original and the scrambled values should have the same number of 1's.
By comparing these values to the input to the RAM in Figure 7, It shows that the values were correctly scrambled.

Figure 7. Reading the ROM and writing the scrambled value to RAM



Finally, the RAM was read by setting RAM_OE to 1'b1 to ensure the values were correctly written to RAM which is shown in Figure 8.

Figure 8. Reading the RAM.



Question

The only edge sensitive signal is the Write Strobe of the RAM. This is to make sure that the Data and Address values are stable when changing the value of the memory. If a Write Strobe was instead Write Enable, a clock signal would still be needed.

Writing with level sensitivity would not work because the data and the address would have to change at exactly the same time which is not guaranteed. This would result in writing the wrong values to a memory location.

```
//TB_ROM.sv
/*=========================================
CSUN - ECE 526 - Lab7
Ridge Tejuco
10/31/2019
=========================================
TB_ROM.sv
- initialize partial memory
- read and scramble data out
- store into RAM
=========================================*/
`timescale 1ns/100ps;
module TB_ROM();
    parameter DEPTH = 32, SIZE = 8, WIDTH = $clog2(DEPTH);
    reg [SIZE - 1:0] ROM_DATA, DATA_IN;
  wire [SIZE - 1: 0] RAM_DATA;
    reg [WIDTH - 1: 0] ROM_ADDR,RAM_ADDR;
  reg ROM_OE,ROM_CS;
  reg RAM_OE,RAM_WS,RAM_CS;
  //ROM(OE,CS,ADDR,DATA);
  ROM UUT(ROM_OE,ROM_CS,ROM_ADDR,ROM_DATA);
  //REGISTER_FILE(OE,WS,CS,ADDR,DATA);
  REGISTER_FILE RAM(RAM_OE,RAM_WS,RAM_CS,RAM_ADDR,RAM_DATA);
  assign RAM_DATA = DATA_IN;
  initial begin
     $vcdpluson;
     $readmemh("data.mem",UUT.MEM,4);
     // init starting signals
     ROM_ADDR = 5'b00100;ROM_CS = 1'b0;ROM_OE = 1'b1;
     RAM_WS = 1'b0;RAM_OE = 1'b0;RAM_CS = 1'b0;
     RAM_ADDR = 5'b00000;
     DATA_IN = 8'bz;

     //test 5'h04 - 5'h1E, then check 5'h1F - 5'h03 are empty
     // block read
     #1 repeat(32) #1 ROM_ADDR = ROM_ADDR + 1;
     #1 ROM_OE = 1'b0;

     //write scrambled ROM to RAM
     #1 ROM_ADDR = 5'b00100; ROM_OE = 1'b1;
```

```
    RAM_ADDR = 5'b00000; //start writing at address 5'h00

    #1 repeat(28) begin
    DATA_IN = {ROM_DATA[0],ROM_DATA[7],ROM_DATA[1],
              ROM_DATA[6],ROM_DATA[2],ROM_DATA[5],
              ROM_DATA[3],ROM_DATA[4]};
    #1 RAM_WS = ~RAM_WS;
    #1 RAM_WS = ~RAM_WS;
    ROM_ADDR = ROM_ADDR + 1;RAM_ADDR = RAM_ADDR + 1;
    end
    #1 ROM_OE = 1'b0; DATA_IN = 8'bz;
    RAM_WS = 1'b0;

    // Check the contents of RAM through block read
    #1 RAM_ADDR = 5'b00000; RAM_OE = 1'b1;
    #1 repeat(32) #1 RAM_ADDR = RAM_ADDR + 1;
    #1 RAM_OE = 1'b0;
    #1 $finish;
  end
endmodule

//end of TB_ROM.sv
```

```
//ROM.sv
/*=========================================
CSUN - ECE 526 - Lab7
Ridge Tejuco
10/29/2019
=========================================
-READ ONLY, Chip select, ADDR, DATA OUT
=========================================*/
module ROM(OE,CS,ADDR,DATA);
    parameter DEPTH = 32, SIZE = 8, WIDTH = $clog2(DEPTH);
    output reg [SIZE - 1:0] DATA;
    input wire [WIDTH - 1: 0] ADDR;
    input wire OE,CS;
    reg [SIZE - 1:0] MEM [DEPTH-1:0];
    always@(CS,OE,ADDR) begin
        if(!CS && OE) DATA <=  MEM[ADDR];
     else DATA <= {(SIZE-1){1'bZ}};
    end
Endmodule
```

```
//TB_REGISTER_FILE.sv
/*==================================
CSUN - ECE 526 - Lab 7
Ridge Tejuco
October 24, 2019
==================================
- write/read to all memory location
- individual and block read
- verify enable/disable
- demonstrate high impedance and Walking ones
==================================*/
`timescale 1ns / 100ps
module TB_REGISTER_FILE();
  parameter SIZE = 8, DEPTH = 32, WIDTH = $clog2(DEPTH);
  reg OE,WS,CS;
  reg [WIDTH-1:0] ADDR;
  wire [SIZE - 1:0] DATA;
  reg [SIZE-1:0] DATA_IN;

  //REGISTER_FILE(OE,WS,CS,ADDR,DATA);
  REGISTER_FILE UUT(OE,WS,CS,ADDR,DATA);
  assign DATA = DATA_IN;

  initial begin
     $vcdpluson;
     OE = 1'b0; WS = 1'b0;CS = 1'b0;ADDR = 5'b00000;
     DATA_IN = 8'h00;
     //write to each address
     #1 repeat(32) begin
     #1 WS = ~WS;
     #1 WS = ~WS;
     DATA_IN = DATA_IN+1;
     ADDR = ADDR+1;
     end

     //Individual read
     DATA_IN = 8'bz;
     #1 ADDR = 5'b00011; WS = 1'b0;
     OE = 1'b1;
     #1 OE = 1'b0;
```

```
    //block read
    #1 ADDR = 5'b00110;
    OE = 1'b1;
    #1 repeat(5) begin
    #1 ADDR = ADDR + 1;
    end

    //test CS
    #1 OE = 1'b0;
    CS = 1'b1;
    DATA_IN = 8'h11;
    #1 WS = ~WS;
    #1 WS = ~WS;
    DATA_IN = 8'bz;
    #1 OE = 1'b1;


    //Walking ones

    #1 CS = 1'b0;
    #1 OE = 1'b0;
    #1 ADDR = 5'b00000;
    DATA_IN = 8'h80;
    //fill 0 - 7
    #1 repeat(8) begin
    #1 WS = ~WS;
    #1 WS = ~WS;
    ADDR = ADDR + 1;
    DATA_IN = DATA_IN >> 1;
    end
    DATA_IN = 8'hz;
    //read 0 - 7
    #1 ADDR  = 5'b00000;
    OE = 1'b1;
    #1 repeat (7) #1 ADDR = ADDR + 1;

    #5 $finish;
  end
endmodule // END of TB_REGISTER_FILE.sv
```

```
//REGISTER_FILE.sv
/*=========================================
CSUN - ECE 526 - Lab7
Ridge Tejuco
10/29/2019
=========================================
-READ, WRITE, Control, ADDR, DATA OUT
=========================================*/
module REGISTER_FILE(OE,WS,CS,ADDR,DATA);
    parameter DEPTH = 32, SIZE = 8, WIDTH = $clog2(DEPTH);
    inout wire [SIZE - 1:0] DATA;
    input wire [WIDTH - 1: 0] ADDR;
  reg [SIZE - 1:0] OUT;
    input wire OE,WS,CS;
    reg [DEPTH-1:0][SIZE - 1:0] REGISTERS ;
  always@(posedge WS) begin
     if(!CS) REGISTERS[ADDR] <= DATA;
  end
  always@(OE,CS,ADDR) begin
     if(!CS && OE) OUT <= REGISTERS[ADDR];
     else OUT <= {(SIZE){1'bZ}};
  end
  assign DATA = OUT;
Endmodule
```

```
// CSUN ECE 526
// Ridge Tejuco
// data.mem
58 ED B7 34
C9 8F A0 9B
65 11 03 4C
DA 7E F2 26
86 95 FD B1
xx xx xx xx
12 AF 33
```