CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

Department of Electrical and Computer Engineering

ECE 526L

LAB – 5: Modelling of a Schematic 8-bit Register

Written By: Raj Kumar

Introduction:

In this lab, we will know about 8-Bit Register using **Synopsys VCS**in **Linux OS** environment and different terminal commands

Methodology:

The first thing is being familiar with the Linux and Synopsys VCS that will navigate your system using terminal.

Use Linux based text editors with command line using,
"gedit" and the file should have **".v"**extension

Create a Verilog model of 8- bit register as given;

Open ▾ | dff.sv ~/526LSP24_raj_kumar/Lab5 | Save

dff.sv | mux2_1.sv | reg.sv | reg_tb.sv | sr_latch2.sv

```
`timescale 1ns / 100ps

`define PRIMARY_OUT     2
`define FAN_OUT_1       0.5
`define FAN_OUT_2       0.8
`define FAN_OUT_3       1
`define TIME_DELAY_1    3
`define TIME_DELAY_2    4
`define TIME_DELAY_3    5

module dff(q, qbar,clock, data, clear);
        input clock, data, clear;
        output q, qbar;

        wire s, sbar, r, rbar, clkbar, cbar, dbar, clock, clear, data, clr, clk, d;

        not #(`TIME_DELAY_1 + `FAN_OUT_1) N1(cbar, clear);
        not #(`TIME_DELAY_1 + `FAN_OUT_1) N2(clkbar, clock);
        not #(`TIME_DELAY_1 + `FAN_OUT_1) N3(dbar, data);
        not #(`TIME_DELAY_1 + `FAN_OUT_1) N4(clr, cbar);
        not #(`TIME_DELAY_1 + `FAN_OUT_1) N5(clk, clkbar);
        not #(`TIME_DELAY_1 + `FAN_OUT_1) N6(d, dbar);

        SR_Latch2 s1(s, sbar, clr, clk, 1'b1, rbar);
        SR_Latch2 s2(r, rbar, s, clk, clr, d);
        SR_Latch2 s3(q, qbar, s, 1'b1, clr, r);
endmodule
```

SystemVerilog ▾ Tab Width: 8 ▾ Ln 21, Col 59 ▾ INS

This is the fig of **"dff.sv"**, store it in new file, also in folder **"Lab5"**.

Now, the **sr_latch2**:

sr_latch2.sv
~/526LSP24_raj_kumar/Lab5

```
`timescale 1ns / 100ps

`define PRIMARY_OUT     2
`define FAN_OUT_1       0.5
`define FAN_OUT_2       0.8
`define FAN_OUT_3       1
`define TIME_DELAY_1    3
`define TIME_DELAY_2    4
`define TIME_DELAY_3    5

module SR_Latch2(Q, Qnot, s0, s1, r0, r1);
        input s0, s1, r0, r1;
        output Q, Qnot;

        nand #(`TIME_DELAY_3 + `FAN_OUT_2) NAND1(Q, s0, s1, Qnot);
        nand #(`TIME_DELAY_3 + `FAN_OUT_2) NAND2(Qnot, r0, r1, Q);

endmodule
```

This is the fig of **"sr_latch2.sv"** and save it and Lab5.

Now use the multiplexor model provided in the assignment and add the same gate delays for the multiplexor cells as used elsewhere from Lab4.

mux2_1.sv
~/526LSP24_raj_kumar/Lab5

```
`timescale 1ns / 1ns

`define PRIMARY_OUT     2
`define FAN_OUT_1       0.5
`define FAN_OUT_2       0.8
`define FAN_OUT_3       1
`define TIME_DELAY_1    3
`define TIME_DELAY_2    4
`define TIME_DELAY_3    5

module MUX2_1(OUT, A, B, SEL);
        output OUT;
        input A, B, SEL;

        wire A1, B1, SEL_N;

        not #(`TIME_DELAY_1 + `FAN_OUT_1) NOT1(SEL_N, SEL);
        not #(`TIME_DELAY_2 + `FAN_OUT_1) AND1(A1, A, SEL_N);
        not #(`TIME_DELAY_2 + `FAN_OUT_1) AND2(B1, B, SEL);
        or #(`TIME_DELAY_2 + `FAN_OUT_1) OR(OUT, A1, B1);
endmodule
```

This is the fig of **"mux2_1.v"** and save it in Lab5.

Now from the given assignment the register has the logic symbol, from its symbol the functions of the **registers "rst"** and **enable "ena"**can be inferred. use the following module header as given in the assignment and save it as **"reg.v"** and save it in **Lab5**.

The below figure is the design file of register;

Open ▾ reg.sv ~/526LSP24_raj_kumar/Lab5 Save

dff.sv | mux2_1.sv | reg.sv | reg_tb.sv | sr_latch2.sv

```
`timescale 1ns / 100ps

module register(clk, rst, ena, data, r);

output  [7:0] r;
input clk, rst, ena;
input   [7:0] data;
wire    [7:0] i;
wire    [7:0] r_bar;

        dff d0(r[0], r_bar[0], clk, i[0], rst);
        dff d1(r[1], r_bar[1], clk, i[1], rst);
        dff d2(r[2], r_bar[2], clk, i[2], rst);
        dff d3(r[3], r_bar[3], clk, i[3], rst);
        dff d4(r[4], r_bar[4], clk, i[4], rst);
        dff d5(r[5], r_bar[5], clk, i[5], rst);
        dff d6(r[6], r_bar[6], clk, i[6], rst);
        dff d7(r[7], r_bar[7], clk, i[7], rst);
                //OUT, A, B, SEL
        MUX2_1 M0(i[0], r[0], data[0], ena);
        MUX2_1 M1(i[1], r[1], data[1], ena);
        MUX2_1 M2(i[2], r[2], data[2], ena);
        MUX2_1 M3(i[3], r[3], data[3], ena);
        MUX2_1 M4(i[4], r[4], data[4], ena);
        MUX2_1 M5(i[5], r[5], data[5], ena);
        MUX2_1 M6(i[6], r[6], data[6], ena);
        MUX2_1 M7(i[7], r[7], data[7], ena);
endmodule
```

SystemVerilog ▾ Tab Width: 8 ▾ Ln 29, Col 1 ▾ INS

Now, write a testbench module to verify the functionality of ourdesign and save it as **"reg_tb.v"**

Open ▾ reg_tb.sv ~/526LSP24_raj_kumar/Lab5 Save

dff.sv | mux2_1.sv | reg.sv | reg_tb.sv | sr_latch2.sv

```
`timescale 1ns / 1ns
`define PULSE #50

module reg_tb();

        reg clk, ena, rst;
        reg [7:0] data;
        wire [7:0] r;
                //clk, rst, ena, data, r
        register UUT(clk, rst, ena, data, r);

        initial begin
                clk = 1'b0;
                rst = 1'b0;
                ena = 1'b0;

                #50 rst = 1'b1;
                #100 ena = 1'b1;

                #300 $finish;
        end

always `PULSE clk = ~clk;

always @(posedge clk)
        $monitor("%d r = %b, clk = %b", $time, r, clk);
        initial begin
                $vcdpluson;
                        data[0] = 1'b1;
                        data[1] = 1'b1;
                        data[2] = 1'b0;
                        data[3] = 1'b0;
                        data[4] = 1'b1;
                        data[5] = 1'b0;
```

SystemVerilog ▾ Tab Width: 8 ▾ Ln 41, Col 1 ▾ INS

reg_tb.sv
~/526LSP24_raj_kumar/Lab5

dff.sv | mux2_1.sv | reg.sv | reg_tb.sv | sr_latch2.sv

```
        wire [7:0] r;
                //clk, rst, ena, data, r
        register UUT(clk, rst, ena, data, r);

        initial begin
                clk = 1'b0;
                rst = 1'b0;
                ena = 1'b0;

                #50 rst = 1'b1;
                #100 ena = 1'b1;

                #300 $finish;
        end

always `PULSE clk = ~clk;

always @(posedge clk)
        $monitor("%d r = %b, clk = %b", $time, r, clk);
        initial begin
                $vcdpluson;
                        data[0] = 1'b1;
                        data[1] = 1'b1;
                        data[2] = 1'b0;
                        data[3] = 1'b0;
                        data[4] = 1'b1;
                        data[5] = 1'b0;
                        data[6] = 1'b1;
                        data[7] = 1'b0;
                $display("%d data = %b", $time, data);
                $display("%d r = %b", $time, r);
        end
endmodule
```

SystemVerilog | Tab Width: 8 | Ln 41, Col 1 | INS

Now for compiling we should create a directory, write a command line code and name it as **lab5.f**

Use the command **"ls"** and then next use the command **"chmod +x lab5.f ".**

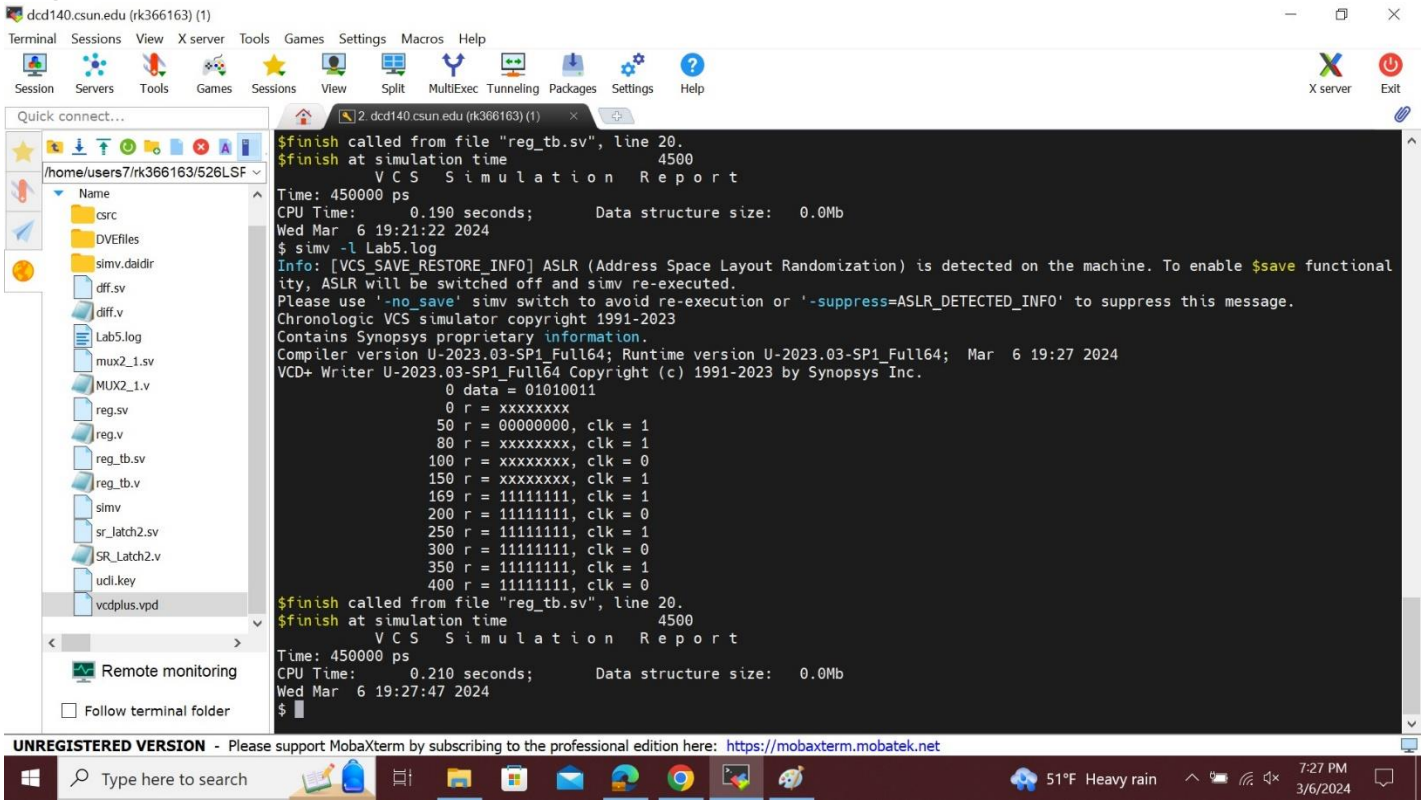
Then use command,

"vcs -debug_access+all -sverilog dff.sv mux2_1.sv sr_latch2.sv reg.sv reg_tb.sv"

simv:

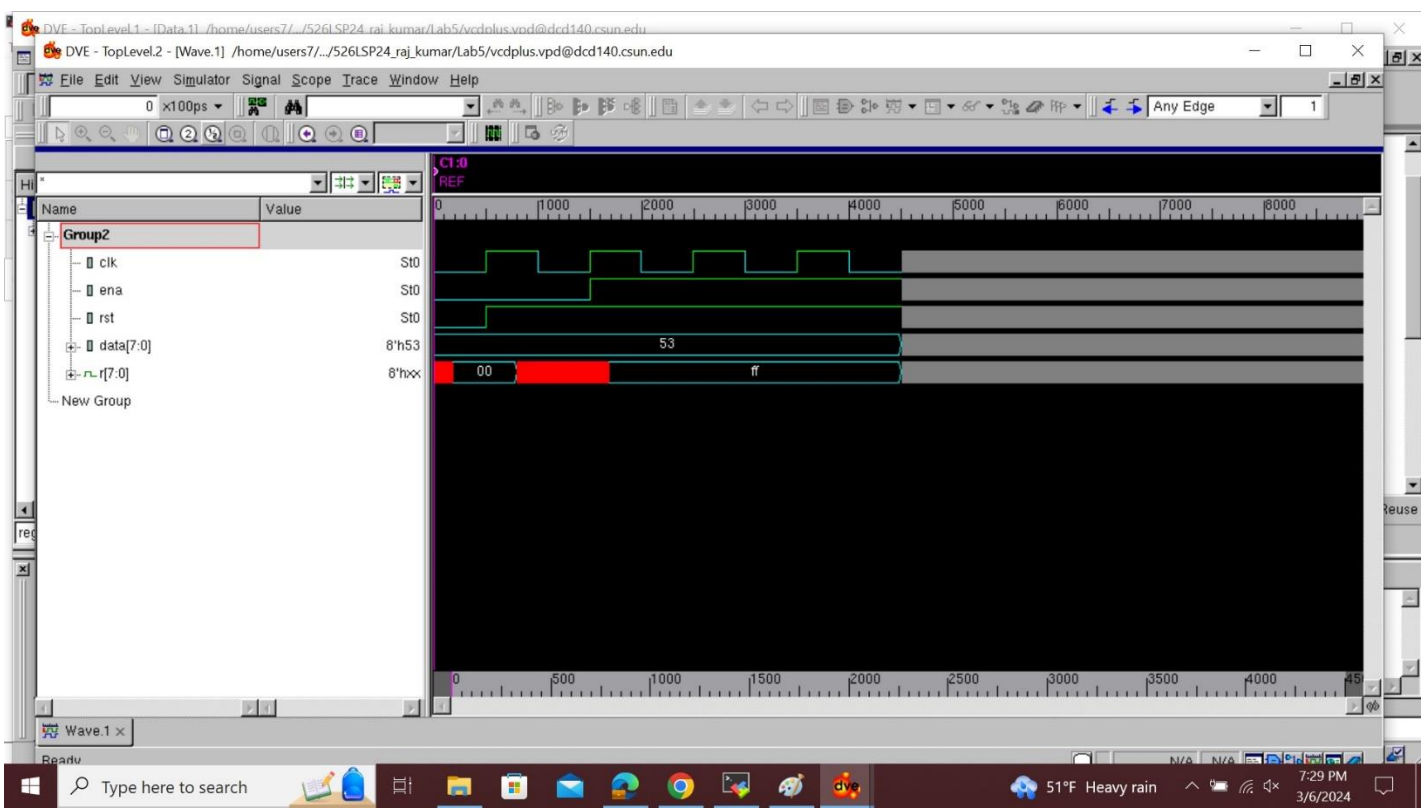
```
$ chmod + x Lab5.f
chmod: cannot access 'x': No such file or directory
chmod: cannot access 'Lab5.f': No such file or directory
$ ls
csrc    diff.v    Lab5.log   MUX2_1.v  reg_tb.sv  reg.v  simv.daidir   SR_Latch2.v  vcdplus.vpd
dff.sv  DVEfiles  mux2_1.sv  reg.sv    reg_tb.v   simv   sr_latch2.sv  ucli.key
$ simv
Info: [VCS_SAVE_RESTORE_INFO] ASLR (Address Space Layout Randomization) is detected on the machine. To enable $save functional
ity, ASLR will be switched off and simv re-executed.
Please use '-no_save' simv switch to avoid re-execution or '-suppress=ASLR_DETECTED_INFO' to suppress this message.
Chronologic VCS simulator copyright 1991-2023
Contains Synopsys proprietary information.
Compiler version U-2023.03-SP1_Full64; Runtime version U-2023.03-SP1_Full64;  Mar  6 19:21 2024
VCD+ Writer U-2023.03-SP1_Full64 Copyright (c) 1991-2023 by Synopsys Inc.
                   0 data = 01010011
                   0 r = xxxxxxxx
                  50 r = 00000000, clk = 1
                  80 r = xxxxxxxx, clk = 1
                 100 r = xxxxxxxx, clk = 0
                 150 r = xxxxxxxx, clk = 1
                 169 r = 11111111, clk = 1
                 200 r = 11111111, clk = 0
                 250 r = 11111111, clk = 1
                 300 r = 11111111, clk = 0
                 350 r = 11111111, clk = 1
                 400 r = 11111111, clk = 0
$finish called from file "reg_tb.sv", line 20.
$finish at simulation time                4500
           V C S   S i m u l a t i o n   R e p o r t
Time: 450000 ps
CPU Time:      0.190 seconds;       Data structure size:   0.0Mb
Wed Mar  6 19:21:22 2024
$
```

Log file:



To see the wave form first type **"dve"** command line then wewill get the blank simulation window then go to file and opendatabase then choose **vcdplus.vpd** and open the file.

Then finally select all the signals with the mouse ,then rightclick and select to **"add to wave->new wave view".** Then we will get the waveform as shown in below figure,



This is the wave form of Schematic 8-bit register.

Analysis of result:

Data is stored using an 8-bit register with an 8-bit mux. When weprovide the data from 0 to 7, which is represented as 8 bits, it is for the register that the multiplexer allotted to us. which functionsas an on/off switch thanks to its one output, eight inputs, and three select pins. According to the provided data, the simv outputcontains the storage data from the multiplexer. The design code sets the cutoff frequency at 269 seconds. That was made very evident in the waveform "Dve" above. The enable input, reset, reg, and input data are used to generate the final output, which is shown as a clock with a 50% duty cycle.

Conclusion:

In this lab we completed schematic for 8 -bit register.

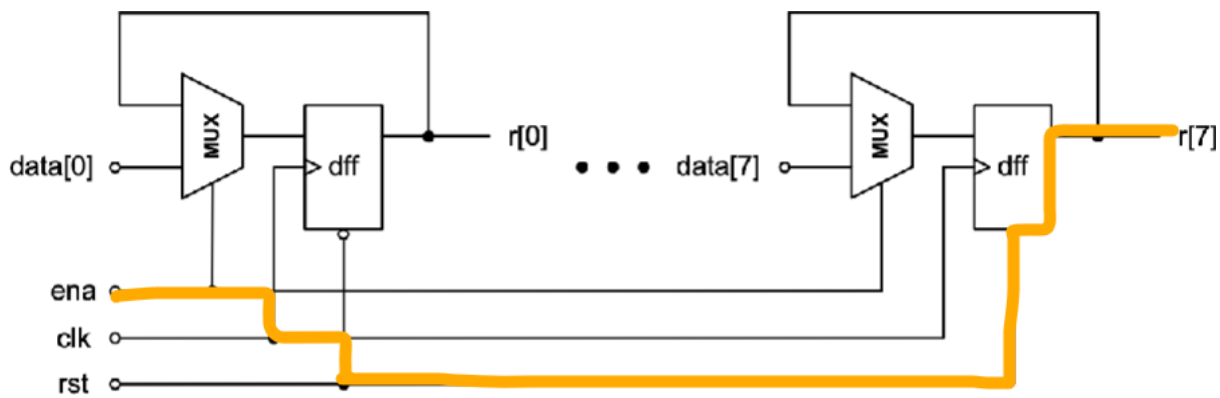
I hereby attest that this lab report is entirely my own work. I have not copied either code or text from anyone, neither have I allowed nor I will let anyone to copy my work.

Name(printed) Raj Kumar

Name(signed) Raj Kumar Date 03/07/2024

LAB REPORT QUESTIONS

1. The longest path,



The calculation of longest path is;
= (3+0.5) +(3+o.8) + (3+0.8) + (3+2)
= 16.1 ns

2. Maximum operating frequency range;

The formula for maximum operating frequency range is 1/f,

i.e ~(1 /16.1) = 0.062MHz.