

**CALIFORNIA STATE UNIVERSITY, NORTHRIDGE**

**Department of Electrical and Computer Engineering**

**ECE 526L**

**LAB – 5**

**SCHEMATIC 8 -BIT REGISTER**

**Professor : Orod Haghighara**

**Written By : Gunupudi. Tarun Kumar**

## INTRODUCTION

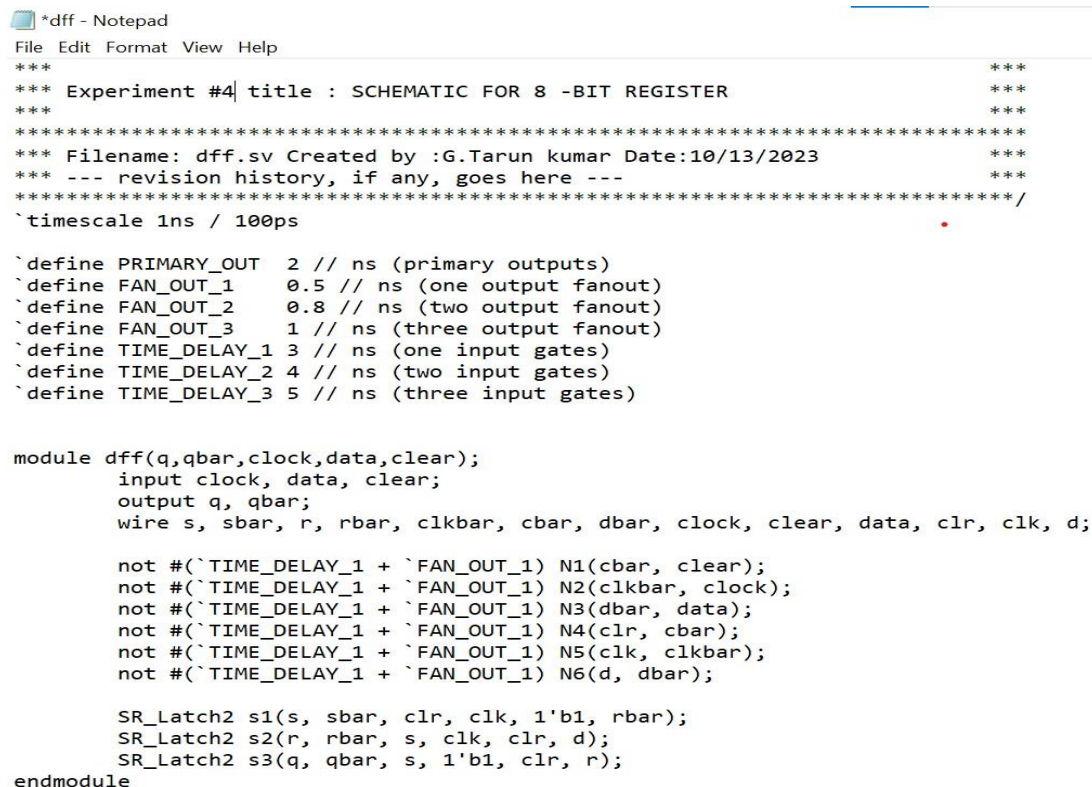
In this lab we will know about 8-Bit Register using Synopsys VCS in Linux OS environment and different terminal commands

## METHODOLOGY

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

Now to write a Verilog code for the module using a text editor. always use linux based text editors and the file should have “.v” extension.

This lab is continuation of lab4, in this lab we should reuse the DFF that is developed in lab3 and should create a Verilog model of 8-bit register as given in fig 3



```
*dff - Notepad
File Edit Format View Help
***
*** Experiment #4 | title : SCHEMATIC FOR 8 -BIT REGISTER ***
***
*****
*** Filename: dff.sv Created by :G.Tarun kumar Date:10/13/2023 ***
*** --- revision history, if any, goes here --- ***
*****/
`timescale 1ns / 100ps

`define PRIMARY_OUT 2 // ns (primary outputs)
`define FAN_OUT_1 0.5 // ns (one output fanout)
`define FAN_OUT_2 0.8 // ns (two output fanout)
`define FAN_OUT_3 1 // ns (three output fanout)
`define TIME_DELAY_1 3 // ns (one input gates)
`define TIME_DELAY_2 4 // ns (two input gates)
`define TIME_DELAY_3 5 // ns (three input gates)

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
```

This is the fig of “dff.v ”, store it in new file, also in folder “lab5”

Now we should also attach the SR\_latch file which we have done in lab4 because DFF module depends on it.

SR\_Latch2 - Notepad

File Edit Format View Help

```

/*****
*** EE 526 L Experiment #4 Student_Name:Gunupudi.Tarun kumar, Fall, 2023 ***
***
*** Experiment #4 title : EDGE TRIGGERED D- FLIPFLOP ***
***
****
*** Filename: SR_Latch2.v Created by :G.Tarun kumar Date:10/06/2023 ***
*** --- revision history, if any, goes here --- ***
*****/
`timescale 1 ns / 100 ps

`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.v” 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 else where from lab4.



mux2\_1 - Notepad

File Edit Format View Help

```
`timescale 1 ns / 1 ns

`define PRIMARY_OUT 2 // ns (primary outputs)
`define FAN_OUT_1 0.5 // ns (one output fanout)
`define FAN_OUT_2 0.8 // ns (two output fanout)
`define FAN_OUT_3 1 // ns (three output fanout)
`define TIME_DELAY_1 3 // ns (one input gates)
`define TIME_DELAY_2 4 // ns (two input gates)
`define TIME_DELAY_3 5 // ns (three input gates)

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);
    and #(`TIME_DELAY_2 + `FAN_OUT_1) AND1(A1, A, SEL_N);
    and #(`TIME_DELAY_2 + `FAN_OUT_1) AND2(B1, B, SEL_N);
    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 figure4 given in the 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.

```

reg - Notepad
File Edit Format View Help
`timescale 1 ns / 100 ps

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

```

Now we should write a testbench module to verify the functionality of our design and save it as “reg\_tb.v” as shown in below.

```

reg_tb - Notepad
File Edit Format View Help
`timescale 1 ns / 1 ns
`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;
        data[6] = 1'b1;
        data[7] = 1'b0;

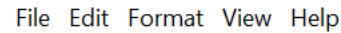
        $display("%d data= %b", $time, data);
        $display("%d r= %b", $time, r);

    end

endmodule

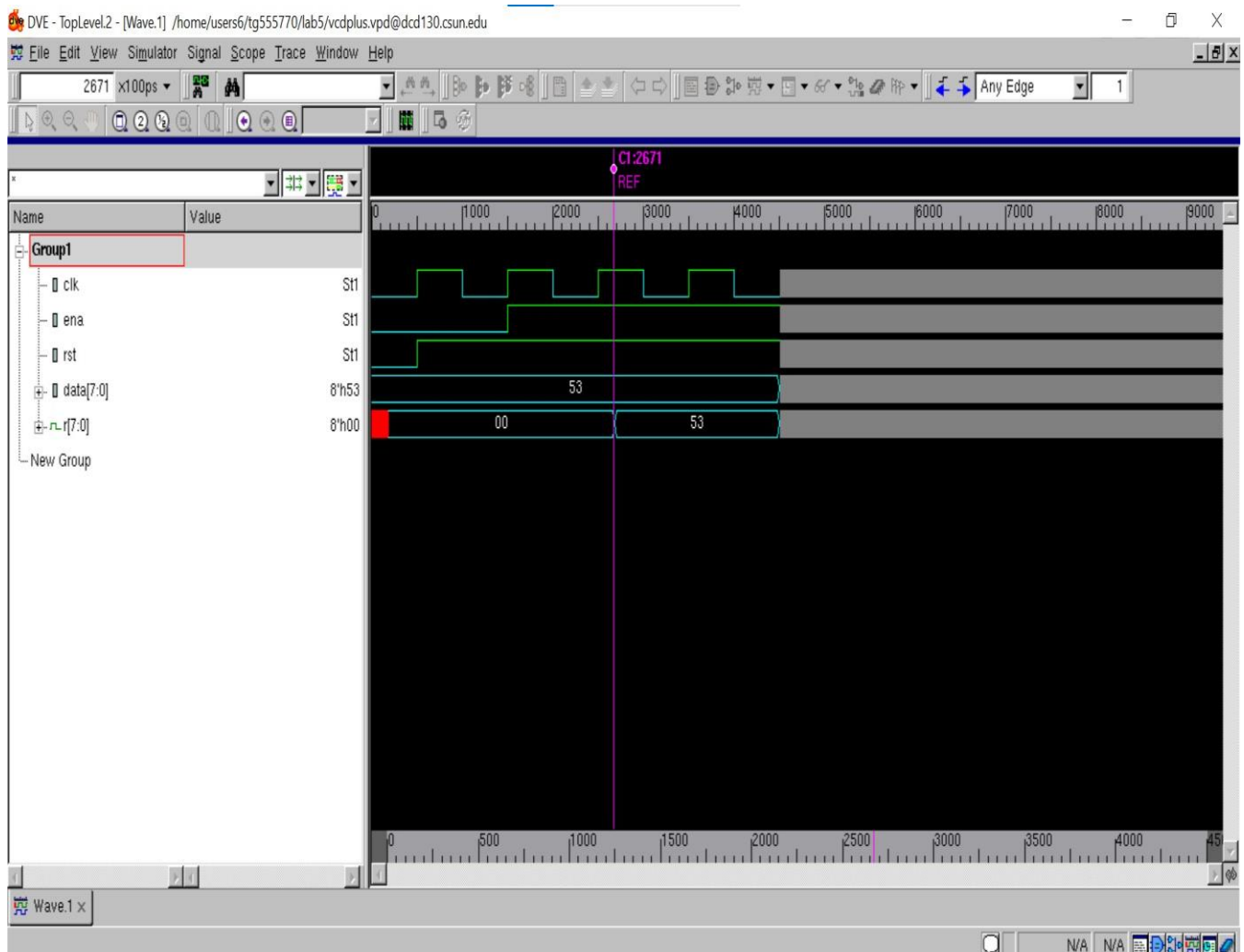
```





To see the wave form first type “dve” command line then we will get the blank simulation window then go to file and open database then choose vcdplus.vpd and open the file.

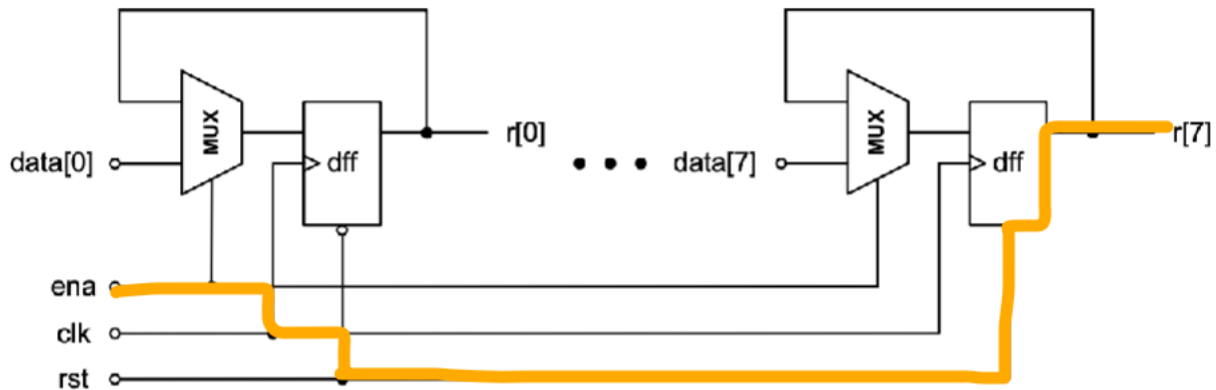
Then finally select all the signals with the mouse ,then right click and select to “ add to wave->new wave view”. Then we will get the waveform as shown in below figure.



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

# LAB REPORT QUESTIONS

## 1.longest path.



A. The calculation of longest path is

$$= (3+0.5) + (3+0.8) + (3+0.8) + (3+2)$$

$$= 16.1 \text{ ns}$$

## 2.Maximum operating frequency range

Ans: The formula for maximum operating frequency range is "1/f "

i.e  $\sim (1 / 16.1) = 0.062 \text{ MHz}$



## Analysis of result :

Data is stored using an 8-bit register with an 8-bit mux. When we provide the data from 0 to 7, which is represented as 8 bits, it is for the register that the multiplexer allotted to us. which functions as an on/off switch thanks to its one output, eight inputs, and three select pins. According to the provided data, the simv output contains 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 done schematic for 8 -bit register by using some commands and some tools by the linux synopsis VCS.

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 I will anyone to copy my work.

Name(printed) Gunupudi Tarun Kumar

Name(signed) G.Tarun

Date 10/13/2023