

Spring 2021  
California State University, Northridge  
Department of Electrical & Computer Engineering

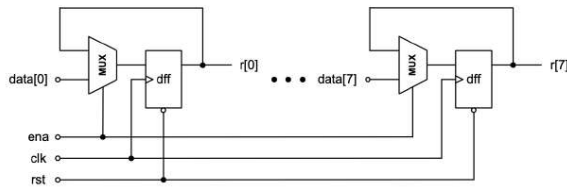


Experiment 4  
8-Bit Register  
February 18, 2021  
ECE 526L

Written By: Jose Luis Martinez

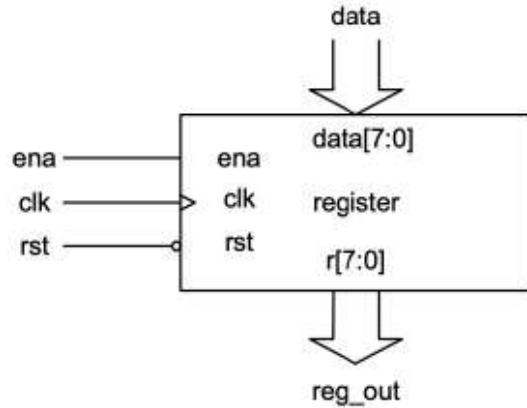
## Introduction

In this lab we are to design an 8-bit register using the D-FF from the previous lab and a 2x1 mux module. We are to set up the circuit as it is shown in **Fig. 1**.



**Fig. 1** 8-Bit Register Schematic

The circuit should take in an enable, clk, rst, and an 8-bit data signal as the inputs. The register should output a 8-bit reg\_out as its output. **Fig. 2** is the circuit symbol for the register.



**Fig. 2** Register Circuit Symbol

While doing this we must make sure to add the proper delays to the 2x1 mux and select an appropriate clk frequency.

## Methodology

For laboratory 4 these were the steps I took to complete the experiment.

1. Made a definitions.v verilog file that contains the timescale, time delays, and strings. See **Fig. 3**.
2. Looking at **Fig. 1** we need to create 2 new modules: mux2\_1 and register.
3. mux2\_1.v was created in **Fig. 6**.
4. Using mux2\_1.v created above and D\_FF.v created from the last lab I made the top level module called register.v in **Fig. 7**.
5. Finally the last verilog file name Lab4\_tb.v was made in **Fig. 8** to test out various test cases for the register module.

## Results

```
1  /*****
2  ***
3  *** ECE 526 L Experiment #4          Jose Luis Martinez, Spring, 2021 ***
4  ***
5  *** Experiment 4 - 8-bit Register ***
6  ***
7  *****/
8  *** Filename: definitions.v Created by Jose Luis Martinez, Febuary 11, 2021 ***
9  ***
10 *****/
11
12 `timescale 1ns/100ps
13
14 `define FAN_OUT_1    0.5
15 `define FAN_OUT_2    0.8
16 `define FAN_OUT_3    1
17 `define TIME_DELAY_1 3
18 `define TIME_DELAY_2 4
19 `define TIME_DELAY_3 5
20 `define PRIMARY_OUT  2
21 `define CLK_PER 100
22 `define MONITOR_STR_1 "%d: clock = %b, reset = %b, enable = %b, data = %h| register output = %h"
```

Fig. 3 definitions.v

```
1  /*****
2  ***
3  *** ECE 526 L Experiment #4          Jose Luis Martinez, Spring, 2021 ***
4  ***
5  *** Experiment 4 - 8-Bit Register ***
6  ***
7  *****/
8  *** Filename: SR_latch2.v   Created by Jose Luis Martinez, Febuary 22, 2021 ***
9  ***
10 *****/
11
12 `include "definitions.v"
13
14
15 module SR_latch2(Q, Q_bar, s0, s1, r0, r1);
16     output Q, Q_bar;
17     input s0, s1, r0, r1;
18
19     parameter DELAY_1 = 0;
20     parameter DELAY_2 = 0;
21
22     nand #(DELAY_1) NAND1(Q, s0, s1, Q_bar);
23     nand #(DELAY_2) NAND2(Q_bar, r0, r1, Q);
24
25 endmodule
```

Fig. 4 SR\_latch2.v

```

1  /*****
2  ***
3  *** ECE 526 L Experiment #4          Jose Luis Martinez, Spring, 2021 ***
4  ***
5  *** Experiment 4 - 8-Bit Register ***
6  ***
7  *****/
8  *** Filename: D_FF.v          Created by Jose Luis Martinez, February 22, 2021 ***
9  ***
10 *****/
11
12 `include "definitions.v"
13
14 module D_FF(q, q_bar, clock, data, clear);
15     output q, q_bar;
16     input clock, data, clear;
17     wire cbar, clkbar, dbar, clr, clk, d, sbar, s, r, rbar;
18
19     not #(`TIME_DELAY_1 + `FAN_OUT_1) NOT1(cbar, clear);
20     not #(`TIME_DELAY_1 + `FAN_OUT_3) NOT2(clr, cbar);
21     not #(`TIME_DELAY_1 + `FAN_OUT_1) NOT3(clkbar, clock);
22     not #(`TIME_DELAY_1 + `FAN_OUT_2) NOT4(clk, clkbar);
23     not #(`TIME_DELAY_1 + `FAN_OUT_1) NOT5(dbar, data);
24     not #(`TIME_DELAY_1 + `FAN_OUT_1) NOT6(d, dbar);
25
26     SR_latch2 sr11( .Q(sbar), .Q_bar(s), .s0(rbar), .s1(1'b1), .r0(clr), .r1(clk));
27     defparam sr11.DELAY_1 = `TIME_DELAY_3 + `FAN_OUT_1;
28     defparam sr11.DELAY_2 = `TIME_DELAY_3 + `FAN_OUT_3;
29
30     SR_latch2 sr12( .Q(r), .Q_bar(rbar), .s0(clk), .s1(s), .r0(clr), .r1(d));
31     defparam sr12.DELAY_1 = `TIME_DELAY_3 + `FAN_OUT_2;
32     defparam sr12.DELAY_2 = `TIME_DELAY_3 + `FAN_OUT_2;
33
34     SR_latch2 sr13( .Q(q), .Q_bar(q_bar), .s0(s), .s1(1'b1), .r0(clr), .r1(r));
35     defparam sr13.DELAY_1 = `TIME_DELAY_3 + `FAN_OUT_1 + `PRIMARY_OUT;
36     defparam sr13.DELAY_2 = `TIME_DELAY_3 + `FAN_OUT_1 + `PRIMARY_OUT;
37
38 endmodule

```

Fig. 5 D\_FF.v

```

1  /*******
2  ***                                     ***
3  *** ECE 526 L Experiment #4           Jose Luis Martinez, Spring, 2021 ***
4  ***                                     ***
5  *** Experiment 4 - 8-Bit Register      ***
6  ***                                     ***
7  *****/
8  *** Filename: mux2_1.v               Created by Jose Luis Martinez, Febuary 22, 2021 ***
9  ***                                     ***
10 *****/
11 `include "definitions.v"
12
13 module mux2_1 (OUT, A, B, SEL);
14     input A, B, SEL;
15     output OUT;
16     wire A1, B1, NSEL;
17
18     not #(`TIME_DELAY_1 + `FAN_OUT_1) NOT(NSEL, SEL);
19     and #(`TIME_DELAY_2 + `FAN_OUT_1) AND1(A1, A, NSEL);
20     and #(`TIME_DELAY_2 + `FAN_OUT_1) AND2(B1, B, SEL);
21     or #(`TIME_DELAY_2 + `FAN_OUT_1) ORI(OUT, A1, B1);
22
23 endmodule

```

**Fig. 6** mux2\_1.v



```

1  /*******
2  ***
3  *** ECE 526 L Experiment #4          Jose Luis Martinez, Spring, 2021 ***
4  ***
5  *** Experiment 4 - 8-Bit Register ***
6  ***
7  *****/
8  *** Filename: register.v      Created by Jose Luis Martinez, February 22, 2021 ***
9  ***
10 *****/
11
12 module register ( CLK, RST, ENA, DATA, R);
13     input CLK, RST, ENA;
14     input [7:0]DATA;
15     output [7:0]R;
16
17     wire [7:0] mux0, R_BAR;
18
19     mux2_1 mux0(.OUT(mux0[0]), .A(R[0]), .B(DATA[0]), .SEL(ENA));
20     mux2_1 mux1(.OUT(mux0[1]), .A(R[1]), .B(DATA[1]), .SEL(ENA));
21     mux2_1 mux2(.OUT(mux0[2]), .A(R[2]), .B(DATA[2]), .SEL(ENA));
22     mux2_1 mux3(.OUT(mux0[3]), .A(R[3]), .B(DATA[3]), .SEL(ENA));
23     mux2_1 mux4(.OUT(mux0[4]), .A(R[4]), .B(DATA[4]), .SEL(ENA));
24     mux2_1 mux5(.OUT(mux0[5]), .A(R[5]), .B(DATA[5]), .SEL(ENA));
25     mux2_1 mux6(.OUT(mux0[6]), .A(R[6]), .B(DATA[6]), .SEL(ENA));
26     mux2_1 mux7(.OUT(mux0[7]), .A(R[7]), .B(DATA[7]), .SEL(ENA));
27
28     D_FF dff0(.q(R[0]), .q_bar(R_BAR[0]), .clock(CLK), .data(mux0[0]), .clear(RST));
29     D_FF dff1(.q(R[1]), .q_bar(R_BAR[1]), .clock(CLK), .data(mux0[1]), .clear(RST));
30     D_FF dff2(.q(R[2]), .q_bar(R_BAR[2]), .clock(CLK), .data(mux0[2]), .clear(RST));
31     D_FF dff3(.q(R[3]), .q_bar(R_BAR[3]), .clock(CLK), .data(mux0[3]), .clear(RST));
32     D_FF dff4(.q(R[4]), .q_bar(R_BAR[4]), .clock(CLK), .data(mux0[4]), .clear(RST));
33     D_FF dff5(.q(R[5]), .q_bar(R_BAR[5]), .clock(CLK), .data(mux0[5]), .clear(RST));
34     D_FF dff6(.q(R[6]), .q_bar(R_BAR[6]), .clock(CLK), .data(mux0[6]), .clear(RST));
35     D_FF dff7(.q(R[7]), .q_bar(R_BAR[7]), .clock(CLK), .data(mux0[7]), .clear(RST));
36
37 endmodule

```

Fig. 7 register.v

```

1  /*****
2  ***
3  *** ECE 526 L Experiment #4          Jose Luis Martinez, Spring, 2021 ***
4  ***
5  *** Experiment 4 - 8-Bit Register ***
6  ***
7  *****/
8  *** Filename: LAB4_tb.v      Created by Jose Luis Martinez, February 22, 2021 ***
9  ***
10 *****/
11
12 `include "definitions.v"
13
14 module Lab3_tb();
15     reg clock, reset, enable;
16     reg [7:0]data;
17     wire [7:0]R1;
18     register reg1(.CLK(clock), .RST(reset), .ENA(enable), .DATA(data), .R(R1));
19
20     initial begin
21         $monitor('MONITOR_STR_1, $time, clock, reset, enable, data, R1);
22     end
23
24     initial begin
25         clock = 0;
26         forever begin
27             #(^CLK_PER) clock = ~clock;
28         end
29     end
30
31     initial begin
32         $vcdpluson;
33         data = 8'h00;
34         enable = 1;
35         reset = 0;
36
37         #200 data = 8'hf0;
38         enable = 1;
39         reset = 0;
40
41         #200 data = 8'hff;
42         enable = 1;
43         reset = 1;
44
45         #200 data = 8'h01;
46         enable = 1;
47         reset = 1;
48
49         #200 data = 8'heb;
50         enable = 1;
51         reset = 1;


```

```

52
53     #200 data = 8'hff;
54         enable = 0;
55         reset = 1;
56
57     #200 data = 8'h01;
58         enable = 0;
59         reset = 1;
60
61     #200 data = 8'hf0;
62         enable = 1;
63         reset = 0;
64
65     #200 data = 8'hff;
66         enable = 1;
67         reset = 0;
68
69     #200 data = 8'h01;
70         enable = 0;
71         reset = 0;
72
73     #400 $finish;
74 end
75
76
77 endmodule

```

Fig. 8 Lab4\_tb.v



```

1 | vcs -debug -full64 definitions.v SR_latch2.v D_FF.v Lab4_tb.v mux2_1.v register.v

```

Fig. 9 compileV.f



```

$ simv
Chronologic VCS simulator copyright 1991-2017
Contains Synopsys proprietary information.
Compiler version N-2017.12-SP2-2_Full64; Runtime version N-2017.12-SP2-2_Full64; Feb 27 20
VCD+ Writer N-2017.12-SP2-2_Full64 Copyright (c) 1991-2017 by Synopsys Inc.
0: clock = 0, reset = 0, enable = 1, data = 00| register output = xx
23: clock = 0, reset = 0, enable = 1, data = 00| register output = 00
100: clock = 1, reset = 0, enable = 1, data = 00| register output = 00
200: clock = 0, reset = 0, enable = 1, data = f0| register output = 00
300: clock = 1, reset = 0, enable = 1, data = f0| register output = 00
400: clock = 0, reset = 1, enable = 1, data = ff| register output = 00
500: clock = 1, reset = 1, enable = 1, data = ff| register output = 00
521: clock = 1, reset = 1, enable = 1, data = ff| register output = ff
600: clock = 0, reset = 1, enable = 1, data = 01| register output = ff
700: clock = 1, reset = 1, enable = 1, data = 01| register output = ff
728: clock = 1, reset = 1, enable = 1, data = 01| register output = 01
800: clock = 0, reset = 1, enable = 1, data = eb| register output = 01
900: clock = 1, reset = 1, enable = 1, data = eb| register output = 01
921: clock = 1, reset = 1, enable = 1, data = eb| register output = eb
1000: clock = 0, reset = 1, enable = 0, data = ff| register output = eb
1100: clock = 1, reset = 1, enable = 0, data = ff| register output = eb
1200: clock = 0, reset = 1, enable = 0, data = 01| register output = eb
1300: clock = 1, reset = 1, enable = 0, data = 01| register output = eb
1400: clock = 0, reset = 0, enable = 1, data = f0| register output = eb
1423: clock = 0, reset = 0, enable = 1, data = f0| register output = 00
1500: clock = 1, reset = 0, enable = 1, data = f0| register output = 00
1600: clock = 0, reset = 0, enable = 1, data = ff| register output = 00
1700: clock = 1, reset = 0, enable = 1, data = ff| register output = 00
1800: clock = 0, reset = 0, enable = 0, data = 01| register output = 00
1900: clock = 1, reset = 0, enable = 0, data = 01| register output = 00

$finish called from file "Lab4_tb.v", line 73.
$finish at simulation time 20000
VCS Simulation Report
Time: 2000000 ps
CPU Time: 0.210 seconds; Data structure size: 0.0Mb
Sat Feb 27 20:20:21 2021
[1] + Done dve -full64 &
$

```

Fig. 10 simv output

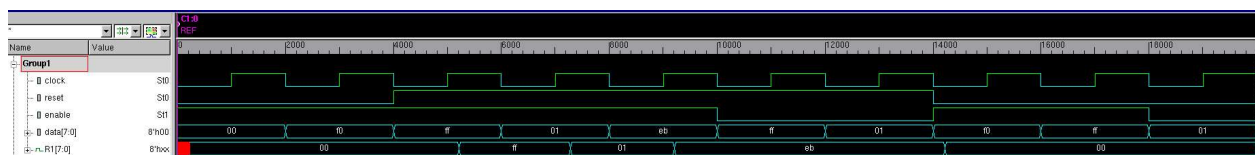


Fig. 11 waveforms

## Analysis

For our register design we had to first design a 2 by 1 multiplexor with proper delays. Because each mux was connected in the same way to each DFF, the delays for all of them are the same. So for the 2 by 1 mux the NOT has a time delay of 3.5ns, both NAND gates the time delay is 4.5ns and the OR gate also has a 4.5ns delay.

Now that we have our 2 by 1 multiplexor we can now combine it with the DFF module to make the register. So 8 sets of mux2\_1 and DFF modules were created in the register.v verilog

file. The output of the mux is connected to the data input of the respective DFF. Input A of the mux is connected to the output of the respective DFF. Input B of the mux is connected to the respective Data input line of the register. The selection bit of the Mux is connected to the enable line of the register. The reset line of the register is connected to the clear input of every DFF module. Finally the output of the register is connected to the outputs of the DFF modules.

Finally I used the Lab4\_tb.v to test out the register. I tested all possible cases to find out if the register was working properly. So I tested cases where enable = 1 & reset = 1, enable = 1 & reset = 0, enable = 0 & reset = 1, enable = 0 & reset = 0.

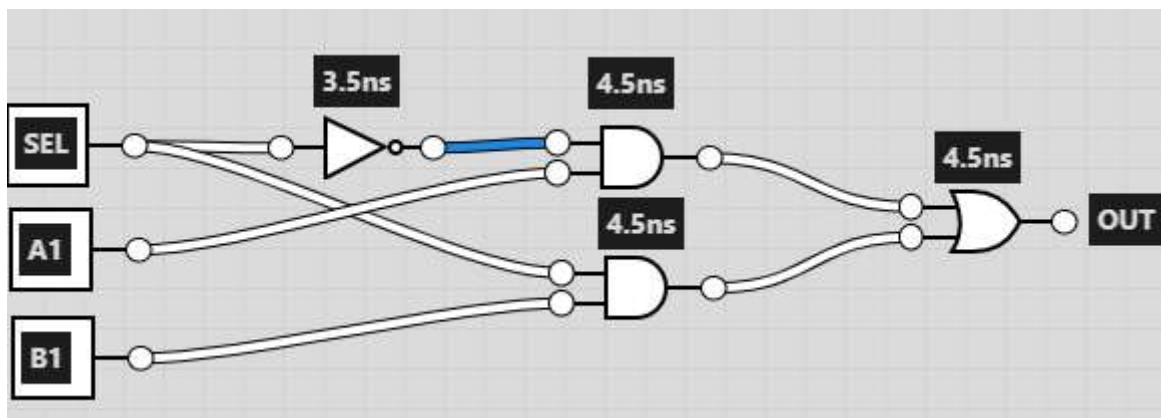
As you can see from **Fig. 10** and **Fig. 11** the register is working as intended. When the reset input is LOW the output of the register is 0. When both enable and reset is HIGH the value of the register output is equal to the data input whenever the clock rises. Whenever the enable is LOW and register is HIGH, the value of the register does change regardless of the value of data and clk inputs.

## Conclusion

In conclusion we used the DFF we designed from the last lab and created a mux module in order to combine them and create a 8-bit register. We also applied the proper delays to the muxes in order to simulate them properly. From our results we were able to see that we correctly implemented the register by testing all possible cases in our Lab4\_tb.v verilog file.

## Questions

Calculate the maximum operating frequency of the entire design. In order to do this, you should calculate the longest path's delay.

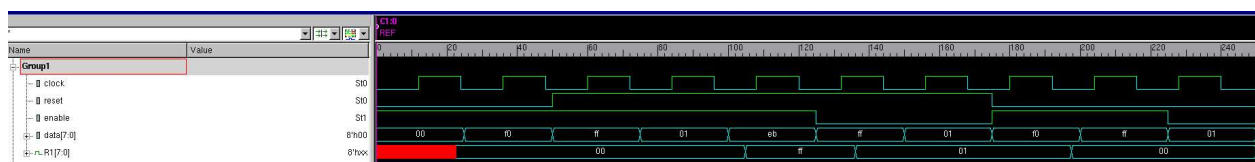


**Fig. 12** 2x1 Mux with time delays

In order to calculate the maximum operating frequency of the register we need to calculate the critical path of the register. From the last lab we found our critical path time delay to be 45.1ns. But now we must find the critical path for the mux because it is connected to the data input of the DFF. From **Fig. 12** we can see that the critical path for the mux is through NOT -> NAND1 -> OR meaning the time delay is 3.5ns + 4.5ns + 4.5ns = 12.5ns. So adding the two delays our new critical path delay is 12.5ns + 45.1ns = **57.6ns**. Using that information we can calculate that the maximum operating frequency is  $F = \frac{1}{T} = \frac{1}{57.6ns} = 17.4MHz$ .

Demonstrate in your simulation and analyze in your report what happens when you exceed that frequency. Correlate your calculated and observed maximum frequencies.

Now that we calculated our maximum operating frequency to be 17.4MHz we can try to see what would happen if we tried a frequency of 40MHz.



**Fig. 13** Waveform at 40MHz

From our graph we can see that our register output no longer matches **Fig. 11** as the hex value 0xeb no longer shows up on R.

## 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 Luis Martinez

Name (signed) Jose Martinez Date 2/27/2021