

Laboratory 5: Sequential Circuits

Learning Objectives

- Design and simulate latches and flip-flops
- Analyze state machines

Sequential circuits are digital circuits in which the output depends on the present and past sequence of the inputs. Latches and flip-flops are commonly used as memory units in sequential circuits. In lab, you will investigate the operation of various types of latches and flip-flops, and analyze a simple state machine.

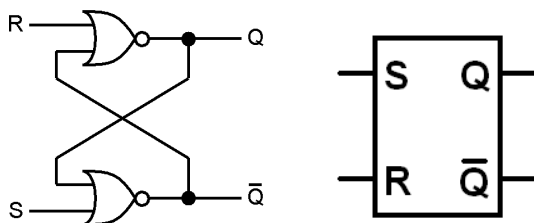
The theoretical background required for this lab are

- latches
- flip-flops
- analysis of state machines

1. Background

S-R Latch

The SR latch (Set-Reset) can be built from two 2-input NOR gates as shown in figure below.



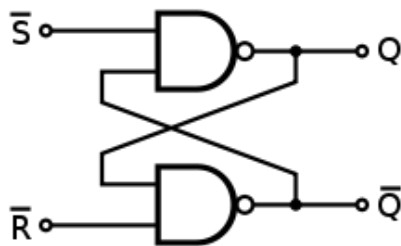
| S | R | Q | \bar{Q} |
|---|---|-----------|-----------|
| 0 | 0 | No change | No Change |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

The circuit has two inputs (S and R), and two outputs (Q and \bar{Q}).

$\bar{S} - \bar{R}$ Latch

An $\bar{S} - \bar{R}$ latch, can be built from NAND gates as shown in figure below.

The Operation of a $\bar{S} - \bar{R}$ latch is similar to a S-R latch with two difference: $\bar{S} - \bar{R}$ latch is active low, and when both \bar{S} and \bar{R} inputs are asserted simultaneously, both outputs go to 1.



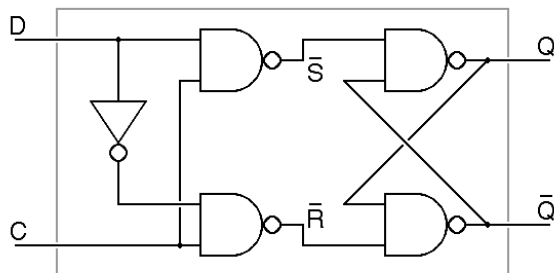
| \bar{S} | \bar{R} | Q | \bar{Q} |
|-----------|-----------|-----------|-----------|
| 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | No change | No change |

D Latch

A D Latch (D for data) can be used to store one bit of information. The D latch can be seen as an extension of the S-R latch that removes the possibility of invalid input states.

As can be seen in the figure below, the righthand side of the logic diagram of a D latch is just an $\bar{S} - \bar{R}$ latch.

Whenever C is 1, whatever that appears on the input D, appears on the output Q (Q follows D), and when C goes to 0, the output holds its last value (hold).

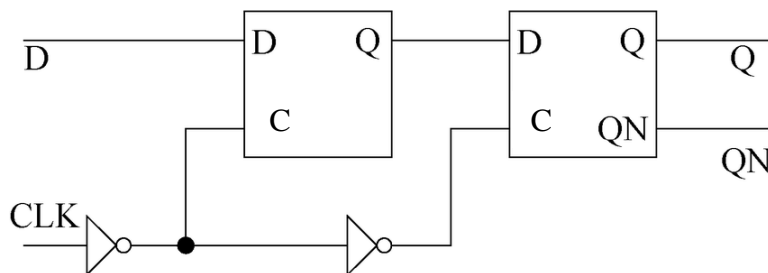


| C | D | Q | \bar{Q} |
|---|---|-----------|-----------|
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |
| 0 | x | No change | No change |

D Flip-Flop

A D flip-flop is distinguished from the D latch by its edge triggered behavior. A positive-edge triggered D flip-flop samples its D input and changes its Q and QN outputs only at the rising edge of a clock (CLK) signal.

Figure below shows a positive-edge-triggered D flip-flop built from a pair of D latches. The first latch is called the master latch. The second latch is called the slave latch.



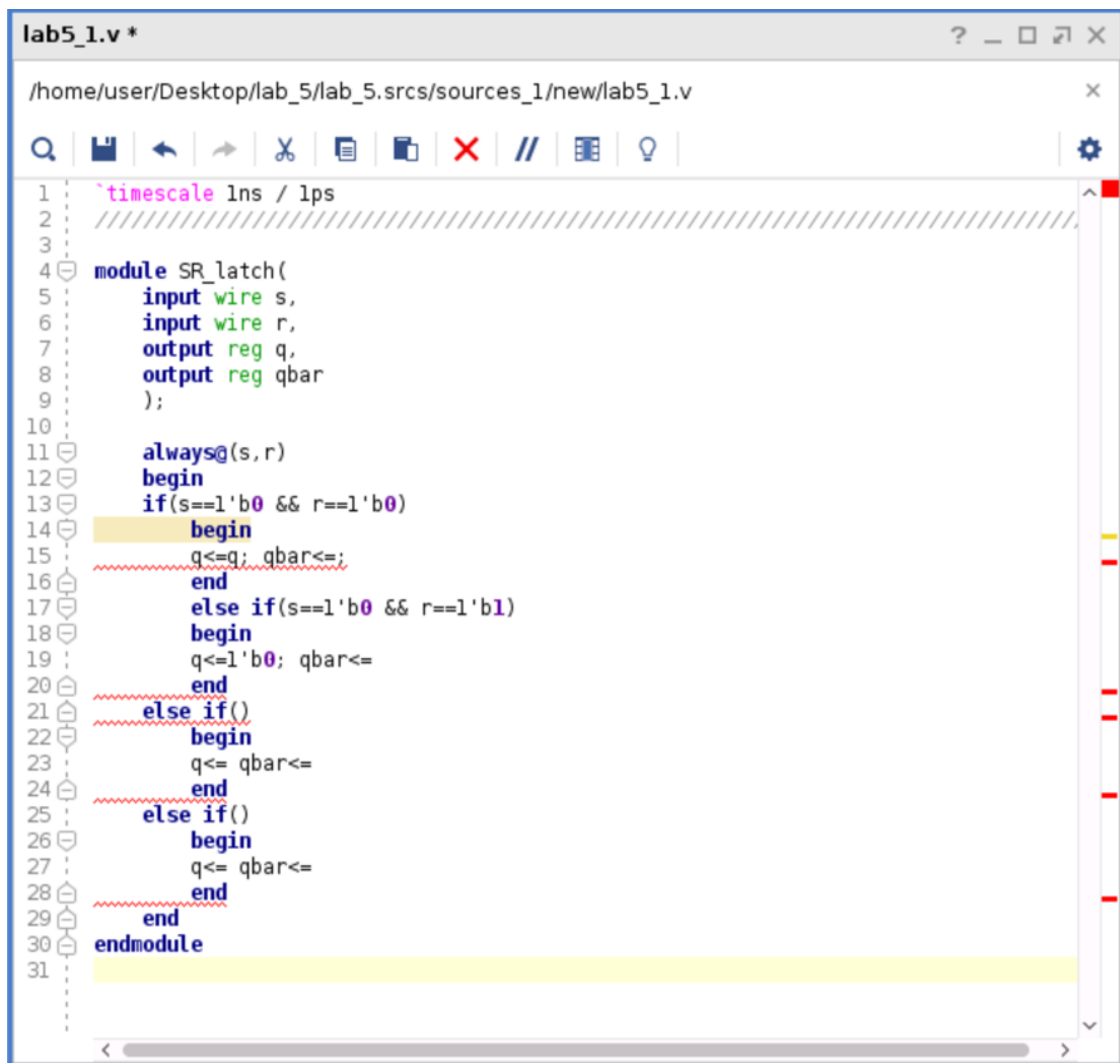
| D | CLK | Q | QN |
|---|-----|--------|---------|
| 0 | | 0 | 1 |
| 1 | | 1 | 0 |
| x | 0 | last Q | last QN |
| x | 1 | last Q | last QN |

2. Pre-Lab

Note that you need to submit individualized pre-lab report for this lab.

2.1 Design and simulate a S-R latch using Vivado

- Create a project file for lab5.
- Here we design a S-R latch using the transition table.
- Create a Verilog source file and implement the SR latch by completing the code below.



```
lab5_1.v *
/home/user/Desktop/lab_5/lab_5.srscs/sources_1/new/lab5_1.v

1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////
3
4  module SR_latch(
5      input wire s,
6      input wire r,
7      output reg q,
8      output reg qbar
9  );
10
11  always@(s,r)
12  begin
13      if(s==1'b0 && r==1'b0)
14          begin
15              q<=q; qbar<=;
16          end
17      else if(s==1'b0 && r==1'b1)
18          begin
19              q<=1'b0; qbar<=
20          end
21      else if()
22          begin
23              q<= qbar<=
24          end
25      else if()
26          begin
27              q<= qbar<=
28          end
29      end
30  endmodule
31
```

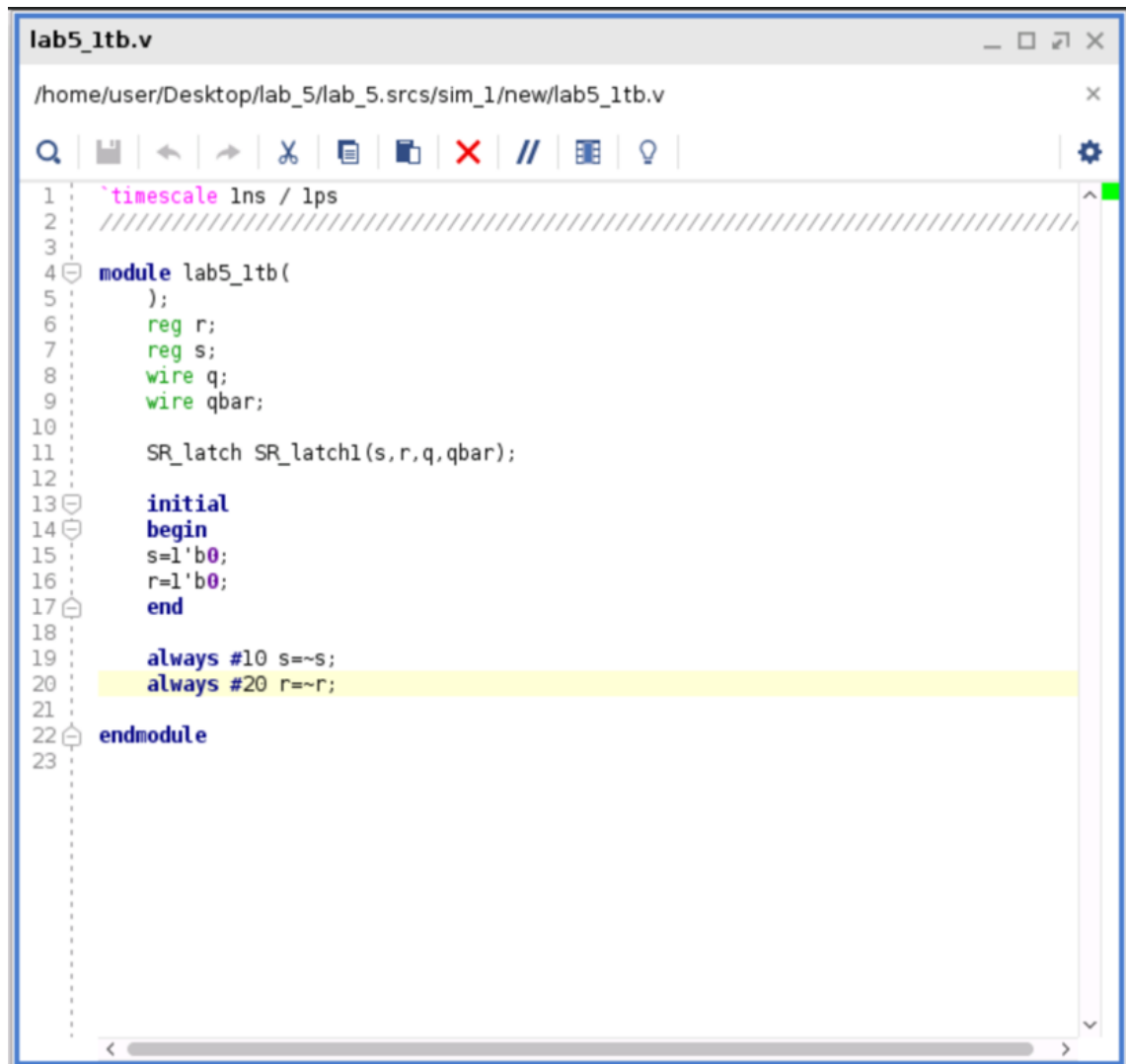
Note: here we are using **non-blocking assignment operator (<=)** instead of blocking assignment (=) that we used previously.

In a blocking assignment (=), execution flow within the procedure is blocked until the assignment is completed. Therefore, the result of the next assignment may depend on the first one being completed. A non-blocking assignment does not block other Verilog

statements from being evaluated during the current time step. It is as statements are executed at the same time. Non-blocking assignments are only made to register data types and therefore only permitted inside of procedural blocks such as always blocks.

Note: *always@(*)* blocks are used to describe events that should happen under certain conditions. *always@* is followed by a set of parentheses in which

- On the Flow Navigator pane, under the RTL Analysis, click on the Open Elaborated Design. Click on the schematic to create the RTL schematic of the synthesized design.
- Test your code by developing a testbench file. Test all the rows of the S-R latch transition table in your code. Don't forget to set the finish time for your test bench same as last week's simulation. Simulate the design.
- View the wave forms and check it with the RS latch transition table.
- Submit the implementation and testbench codes, schematic and the simulation results.

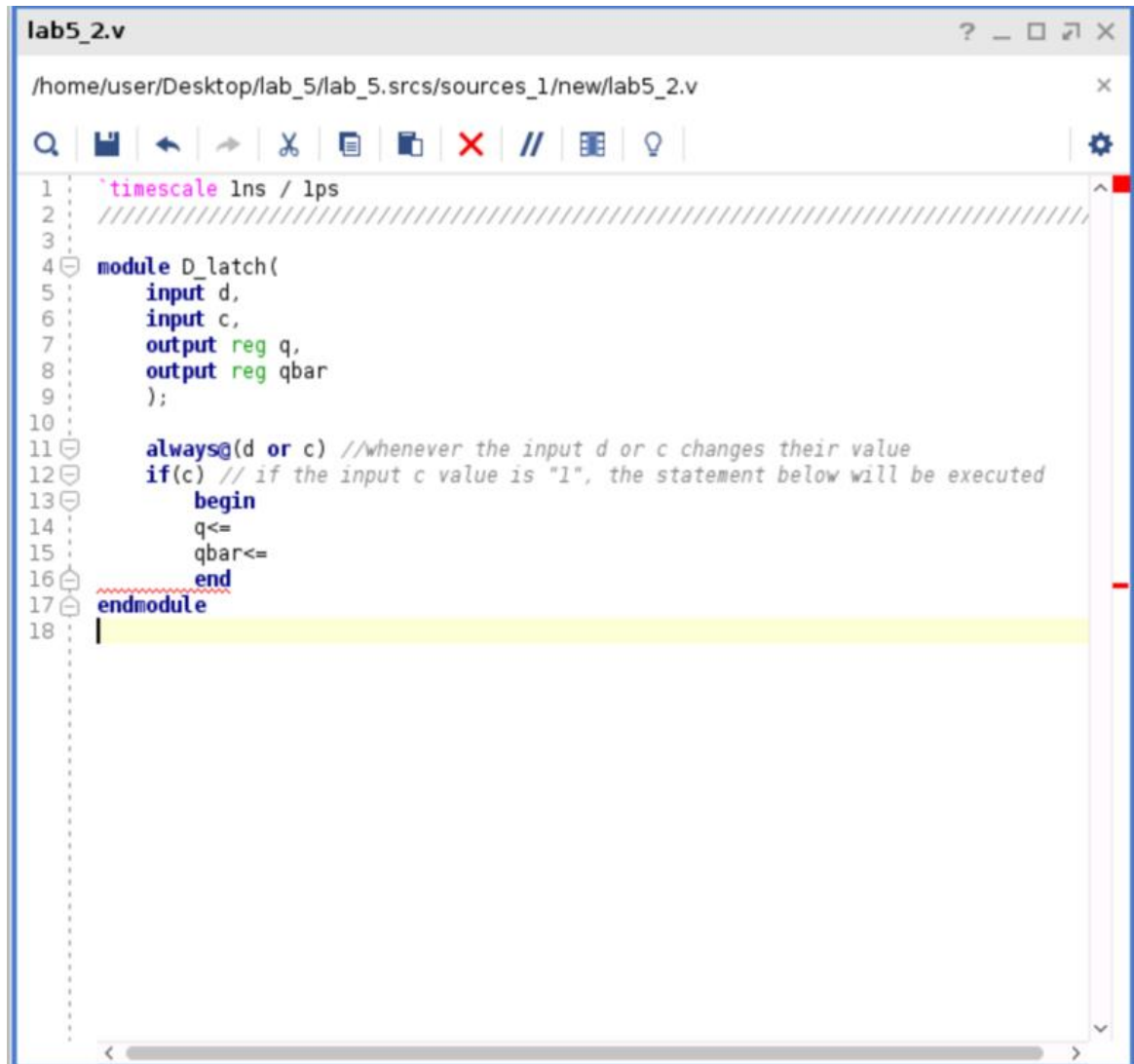


```
lab5_1tb.v
/home/user/Desktop/lab_5/lab_5.srscs/sim_1/new/lab5_1tb.v

1  `timescale 1ns / 1ps
2  //////////////////////////////////////
3
4  module lab5_1tb(
5      );
6      reg r;
7      reg s;
8      wire q;
9      wire qbar;
10
11      SR_latch SR_latch1(s,r,q,qbar);
12
13      initial
14      begin
15          s=1'b0;
16          r=1'b0;
17      end
18
19      always #10 s=~s;
20      always #20 r=~r;
21
22  endmodule
23
```

2.2 Design and simulate a D latch using Vivado

- Here we design a D latch using the transition table
- Create a Verilog source file and implement the D latch by completing the code below.



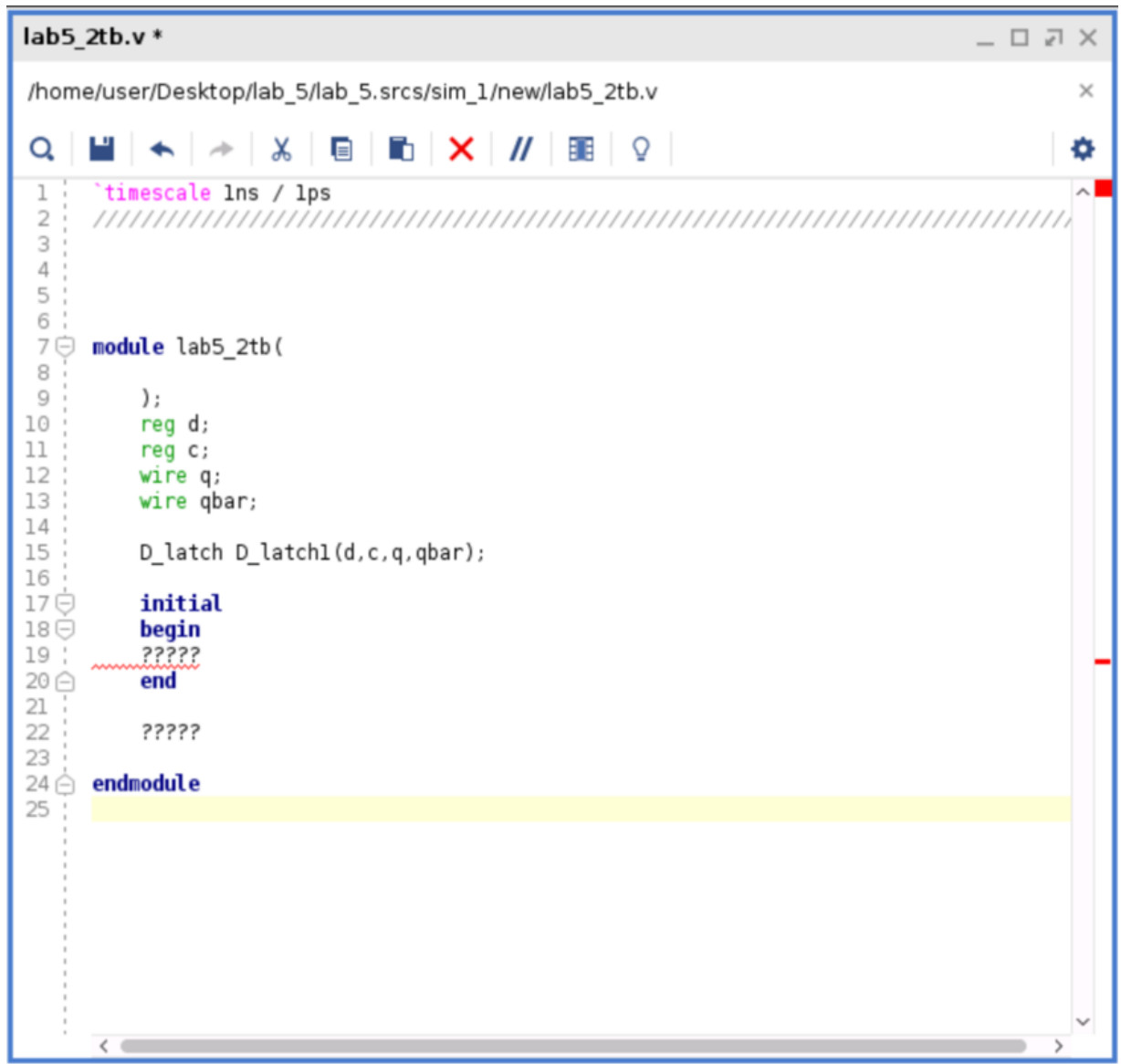
```
lab5_2.v
/home/user/Desktop/lab_5/lab_5.srcs/sources_1/new/lab5_2.v

1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////
3
4  module D_latch(
5      input d,
6      input c,
7      output reg q,
8      output reg qbar
9  );
10
11  always@(d or c) //whenever the input d or c changes their value
12  if(c) // if the input c value is "1", the statement below will be executed
13  begin
14      q<=
15      qbar<=
16  end
17  endmodule
18
```

Note: we just specify what to do when c is high or “1” and we do not specify what to do when c is low or “0”. In this case, when c is low the circuit will remember or keep the previous state of the outputs.

Also, the always block is sensitive to d and c, therefore, whenever d changes the output will be updated.

- On the Flow Navigator panel, under the RTL Analysis, click on the Open Elaborated Design. Click on the schematic to create the schematic of the circuit.
- Develop a testbench to validate the design behavior. It should generate all the transitions for the inputs in the transition table (complete the code below).



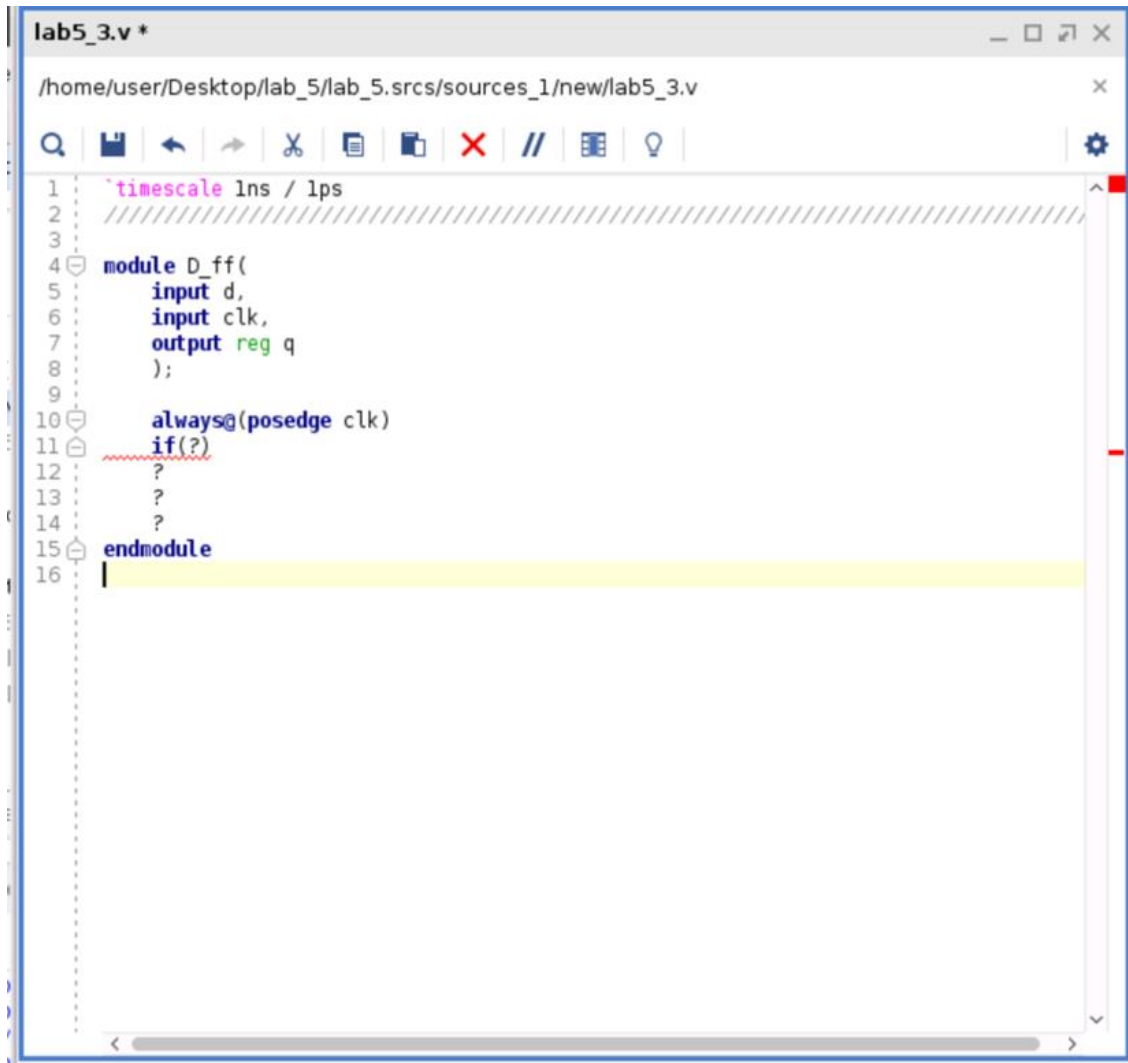
```
lab5_2tb.v *
/home/user/Desktop/lab_5/lab_5.srcs/sim_1/new/lab5_2tb.v

1  `timescale 1ns / 1ps
2  //////////////////////////////////////
3
4
5
6
7  module lab5_2tb(
8
9  );
10     reg d;
11     reg c;
12     wire q;
13     wire qbar;
14
15     D_latch D_latch1(d,c,q,qbar);
16
17     initial
18     begin
19         ?????
20     end
21
22     ?????
23
24 endmodule
25
```

- Submit the implementation and testbench codes, schematic and the simulation results.

2.3 Design and simulate the D flip-flop circuit using Vivado

- Create Verilog source file
- Implement a D flip-flop by completing the Verilog code below.



The screenshot shows a Verilog code editor window titled 'lab5_3.v *'. The file path is '/home/user/Desktop/lab_5/lab_5.srcs/sources_1/new/lab5_3.v'. The code is as follows:

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////
3
4  module D_ff(
5      input d,
6      input clk,
7      output reg q
8  );
9
10     always@(posedge clk)
11     if(?)
12         ?
13         ?
14         ?
15 endmodule
16
```

Note: the always block is sensitive to the rising edge (**posedge**) on the clk signal.

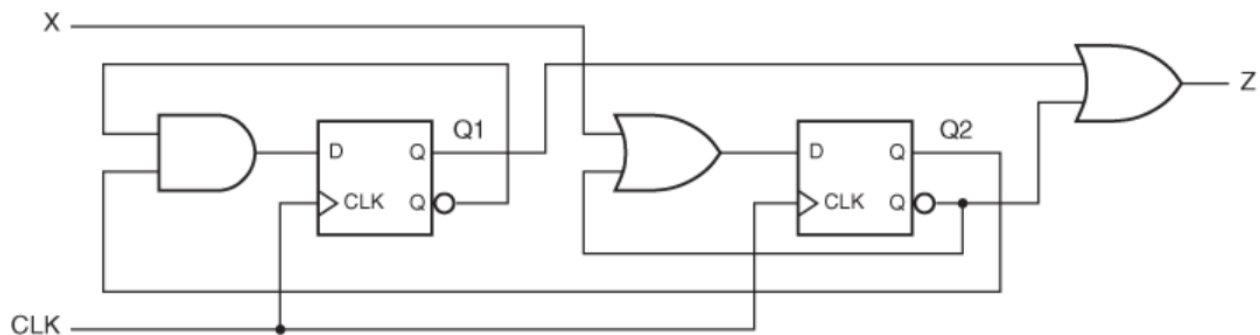
Hint: you should check the clk signal value using if(?) and then write the statement for the if block that should be executed.

- On the Flow Navigator panel, Under the RTL Analysis, Click on the Open Elaborated Design. Click on the schematic to create the RTL schematic of the circuit.
- Develop a test bench code to validate the D flip-flop and test all the transitions based on the D flip-flop transition table.
- Submit the implementation and testbench codes, schematic and the simulation results.

2.4 Analyze a simple state machine

For the circuit shown below:

- Write down the excitation equations of the two D flip-flops, carrying the state variables Q1 and Q2. Derive the transition equations of the state machine. Include this in your pre-lab report.
- Using the transition equations and the state names, derive the state transition table. Include this in your pre-lab report.
- Using the state transition table, draw the state diagram. Include this in your pre-lab report.
- Implement and simulate the state machine by instantiating the D flip-flop module that you wrote in 2.3. View the output waveforms by developing a test bench. Submit your implementation codes, testbench, RTL schematic and waveforms.



3. Experiments

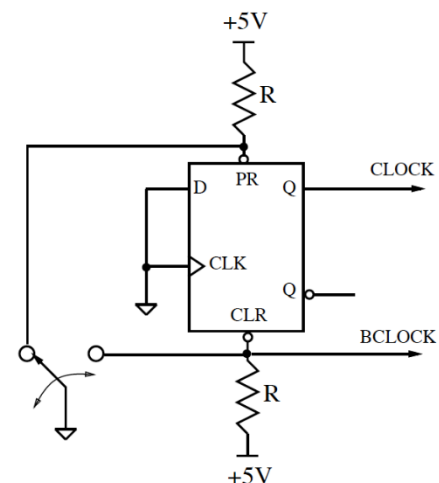
3.1 Implement a clock generator circuit

Before implementing the circuit in 2.4, you will need to build a circuit generating the clock signal. At the end of this laboratory, do not remove the clock generation circuit from the protoboard.

The circuit generating the clock signal is shown below.

To be able to use a mechanical switch as a clock signal, the switch must be debounced first. The shown circuit here achieves this with the help of the asynchronous inputs of a D flip-flop. Be sure that you understand how it functions.

- Implement the debouncing circuit. Use the single-pole double-throw switch provided. The resistors are 620 Ω . Attach a long wire to the CLOCK output. For the moment connect the wire to an LED for the ICOD unit.



- Generate several clock cycles. How many switches are needed for one cycle? Draw the waveform of CLOCK for two cycles and indicate the position of the switch for each signal level.

3.2 Implement a simple state machine

- Implement the circuit you analyzed in 2.4. Use SN74LS4A for D flip-flops, SN74LS08 for the AND gate and SN74LS32 for the OR gates. The input X will be connected to a switch in the ICOD unit.
- Connect the \overline{PR} of the two SN74LS4A D flip-flops together and to a switch of the ICOD unit.
- Connect the \overline{CLR} of the two SN74LS4A D flip-flops together and to a switch.
- Connect the CLK signals of the two SN74LS4A D flip-flops together, and to the debounced clock signal.
- Connect the two state variables Q1 and Q2 to LEDs.
- Use the LED's and/or the logic probe to check if the inputs into the flip-flops follow the values from the excitation equations.
- Verify that the inputs follow the excitation equations in the presence of your TA

3.3 Testing your state machine – Test 1

- Use the circuit you implemented in the previous section i.e. Section 3.2 to test your state machine. You can do this by following the steps outlined in this section.
- Set the input X to 1 i.e. connect it to 5V by using the switch connected to the input.
- Connect the output Z to the LED.
- Toggle the CLK signal and note down the values of Q1, Q2, and Z using the LED's connected to these signals.
- Keep toggling the CLK signal to fill the following table for the CLK transition:

| <u>CLK Transition #</u> | <u>Q1</u> | <u>Q2</u> | <u>Z</u> |
|-------------------------|-----------|-----------|----------|
| 1 | | | |
| 2 | | | |
| 3 | | | |
| 4 | | | |

| | | | |
|---|--|--|--|
| 5 | | | |
| 6 | | | |
| 7 | | | |

3.4 Testing your state machine – Test 2

- Use the circuit you implemented in the previous section i.e. Section 3.2 to test your state machine. You can do this by following the steps outlined in this section.
- Set the input X to 0 i.e. connect it to GND by using the switch connected to the input and note down the output of the state variables Q1 and Q2.
- Connect the output Z to the LED.
- Toggle the CLK signal and note down the values of Q1, Q2, and Z using the LED's connected to these signals.
- Keep toggling the CLK signal to fill the following table for the CLK transition:

| <u>CLK Transition #</u> | <u>Q1</u> | <u>Q2</u> | <u>Z</u> |
|-------------------------|-----------|-----------|----------|
| 1 | | | |
| 2 | | | |
| 3 | | | |
| 4 | | | |
| 5 | | | |
| 6 | | | |
| 7 | | | |

4. Lab 5 Report

Your final report for Lab 5 should include the following:

- your individual prelab reports, with all the materials requested
- images of the circuits that you implemented and discussion of the experimental results