Laboratory Exercise 3

Latches, Flip-flops, and Registers

The purpose of this exercise is to investigate sequential circuits.

Preparation

In advance of your lab period write the Verilog code for Parts I to IV. Also, make sure that you have successfully simulated Parts I to IV with ModelSim. You will need to demonstrate your simulations for some of these parts to your teaching assistant (TA).

In-lab

You are required to complete all parts of the exercise. Your TA will ask you to demonstrate some of these parts.

Part I

Figure 1 depicts a gated RS latch circuit. Verilog code that describes this latch using Boolean logic expressions is given in Figure 2.

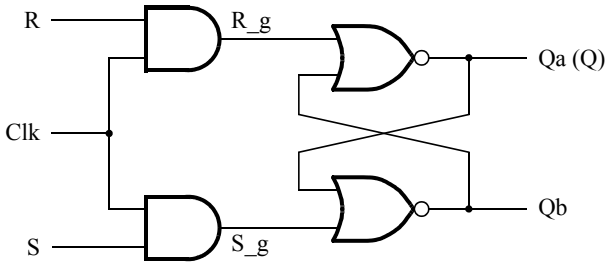


Figure 1: A gated RS latch circuit.

```
module part1 (Clk, R, S, Qa, Qb);
    input Clk, R, S;
    output Qa, Qb;

    wire R_g, S_g;
    assign R_g = R & Clk;
    assign S_g = S & Clk;
    assign Qa = ~(R_g | Qb);
    assign Qb = ~(S_g | Qa);
endmodule
```

Figure 2: Verilog code for a gated RS latch.

Perform the following:

1. Create a Verilog source-code file for the gated RS latch, using the code in Figure 2.

2. You are to simulate this Verilog code using the input waveforms for *Clk*, *R*, and *S* that are illustrated in Figure 3. You can create these waveforms by making a new ModelSim project and drawing the waves, or by making a Quartus project and then creating a new *University Program VWF* file to simulate within Quartus, or by writing a ModelSim testbench. Sample ModelSim setup files for using a testbench are provided as part of the *design files* for this exercise. Make sure that your simulation results for outputs Qa and Qb match those depicted in Figure 3, and that you fully understand how the latch circuit works.

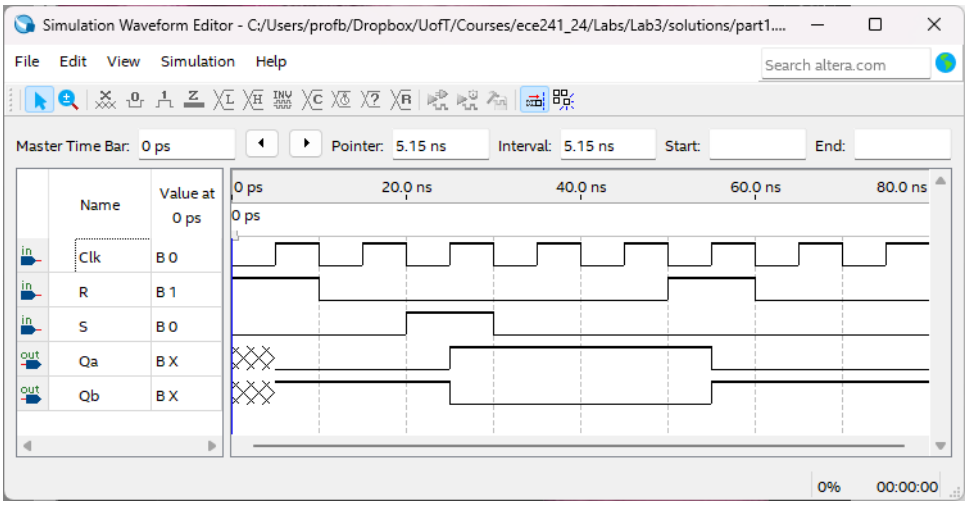


Figure 3: Simulation waveforms for the gated RS latch.

Part II

Figure 4 shows the circuit for a gated D latch.

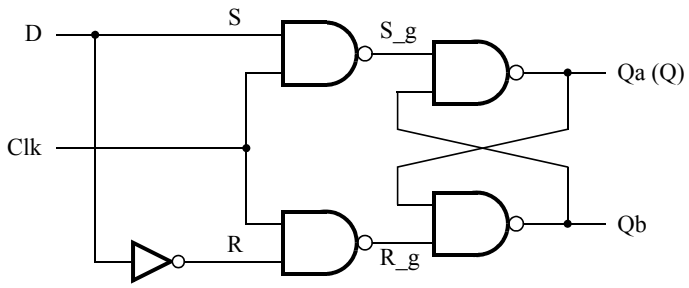


Figure 4: Circuit for a gated D latch.

Perform the following:

1. Write a Verilog module, named *D_latch*, using the style of code from Figure 2 for the gated D latch.

2. Simulate your gated D latch using the same method that you chose for Part I. Verify that the latch works properly for all input conditions.

3. After successfully simulating your gated D latch, use Quartus to implement the latch on the DE1-SoC board. For your Quartus project, you should create a top-level Verilog module, named *part2*, that contains the appropriate input and output ports (pins) for the board. Instantiate your latch in this top-level module. Use switch SW_0 to drive the *D* input of the latch, and use SW_1 as the *Clk* input. Connect the Q output to $LEDR_0$.

4. Since your *part2.v* file uses the SW switches and LEDR lights on the DE1-SoC board, you may want to simulate your design with the DESim tool. A file *top.v* for use with *DESim* is included with the design files for this exercise. To directly use this *top.v* file, define your *part2.v* module as:

```
module part2 (SW, LEDR);
    input [1:0] SW;
    output [9:0] LEDR;
    ...
endmodule
```

5. After completing simulations of your design, make sure that your Quartus project includes the required pin assignments, then compile the project and download the resulting circuit into the DE1-SoC board.

6. Test the functionality of your circuit.

Part III

Figure 5 shows the circuit for a master-slave D flip-flop.

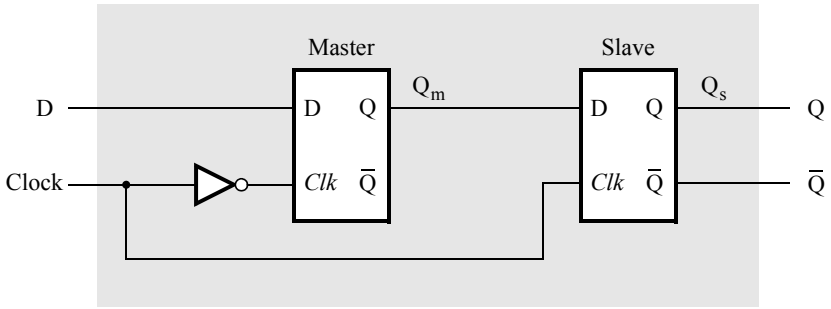


Figure 5: Circuit for a master-slave D flip-flop.

Perform the following:

1. Write a Verilog file, named *part3.v*, that instantiates two copies of your gated D latch module from Part II to implement the master-slave flip-flop. Use switch SW_0 to drive the *D* input of your flip-flop, and use SW_1 as the *Clock* input. Connect the Q output to $LEDR_0$.

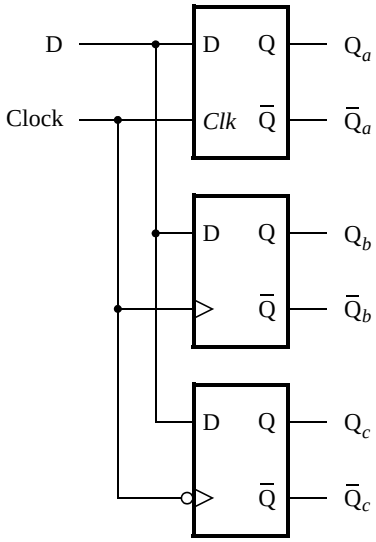
2. Simulate your Verilog code using ModelSim. You may want to also simulate your Verilog code with DESim. ModelSim setup files and a *top.v* file for DESim are included with the design files for this exercise.

3. After completing simulations of your design, make sure that your Quartus project includes the required pin assignments, then compile the project and download the resulting circuit into the DE1-SoC board.

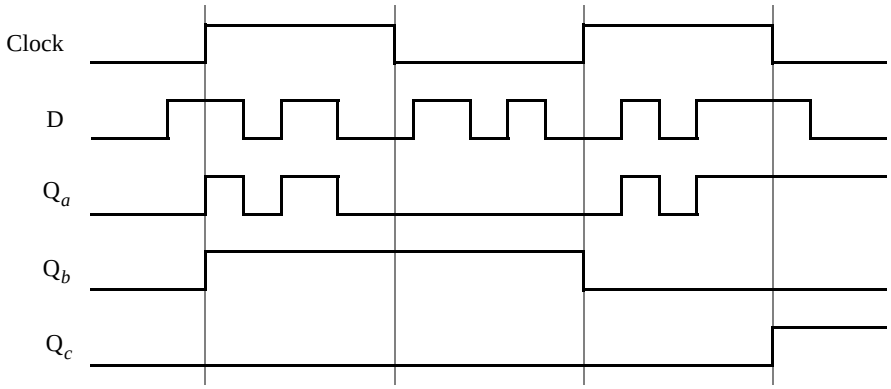
4. Test the functionality of your circuit.

Part IV

Figure 6 illustrates a circuit that includes three different storage elements: a gated D latch, a positive-edge triggered D flip-flop, and a negative-edge triggered D flip-flop.



(a) Circuit



(b) Timing diagram

Figure 6: Circuit and waveforms for Part IV.

Perform the following:

1. Write Verilog code that instantiates the three storage elements.
2. Simulate your Verilog code by using ModelSim. You can choose to use ModelSim either by drawing waveforms or using a testbench. For the latter approach, a ModelSim testbench is provided as part of the *design files* for this exercise.

   Regardless of your ModelSim methodology, simulate using at least the input waveforms shown for signals *D* and *Clock* in Figure 6. Make sure that you understand the different behaviors of the storage elements.

Part V

We wish to display the hexadecimal value of an 8-bit number A on the two 7-segment displays *HEX*$3-2$. We also wish to display the hex value of an 8-bit number B on the two 7-segment displays *HEX*$1-0$. The values of A and B are inputs to the circuit which are provided by means of switches SW_{7-0}. To input the values of A and B, first set the switches to the desired value of A, store these switch values in a register, and then change the switches to the desired value of B. Finally, use an adder to generate the arithmetic sum $S = A + B$ (the + symbol means *addition* in this expression; it does not mean OR), and display this sum on the 7-segment displays *HEX*$5-4$. Show the carry-out produced by the adder on LEDR[0].

Your Verilog module should be defined as:

```
module part5 (SW, KEY, HEX5, HEX4, HEX3, HEX2, HEX1, HEX0, LEDR);
    input [7:0] SW;
    input [1:0] KEY;
    output [6:0] HEX5, HEX4, HEX3, HEX2, HEX1, HEX0;
    output [9:0] LEDR;
    ...
    ...
endmodule
```

1. A ModelSim testbench is provided as part of the *design files* for this exercise, as well as a top-level file for use with DESim. Simulation is optional, at your discretion, for this part of the exercise.
2. Create a new Quartus project which will be used to implement the desired circuit on the DE1-SoC board.
3. Write a Verilog file that provides the necessary functionality. Use KEY_0 as an active-low asynchronous reset, and use KEY_1 as a manual clock input.
4. Include the necessary pin assignments for the switches and 7-segment displays, and then compile the circuit.
5. Download your circuit into the DE1-SoC board and test its functionality by operating the switches and observing the displays.