

Laboratory Exercise 5

Latches, Flip-flops, and Registers

The purpose of this exercise is to investigate latches, flip-flops, and registers.

Part I R-S latch

Altera FPGAs include flip-flops that are available for implementing a user's circuit. We will show how to make use of these flip-flops in Part IV of this exercise. But first we will show how storage elements can be created in an FPGA without using its dedicated flip-flops.

Figure 1 depicts a gated RS latch circuit. Two styles of Verilog code that can be used to describe this circuit are given in Figure 2. Part *a* of the figure specifies the latch by instantiating logic gates, and part *b* uses logic expressions to create the same circuit. If this latch is implemented in an FPGA that has 4-input lookup tables (LUTs), then only one lookup table is needed, as shown in Figure 3a.

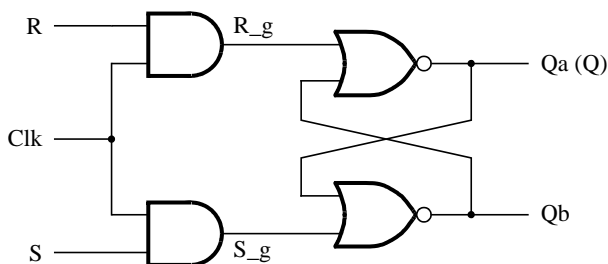


Figure 1. A gated RS latch circuit.

```
// A gated RS latch
module part1 (Clk, R, S, Q);
    input Clk, R, S;
    output Q;

    wire R_g, S_g, Qa, Qb /* synthesis keep */;

    and (R_g, R, Clk);
    and (S_g, S, Clk);
    nor (Qa, R_g, Qb);
    nor (Qb, S_g, Qa);

    assign Q = Qa;

endmodule
```

Figure 2a. Instantiating logic gates for the RS latch.

```
// A gated RS latch
module part1 (Clk, R, S, Q);
  input Clk, R, S;
  output Q;

  wire R_g, S_g, Qa, Qb /* synthesis keep */;

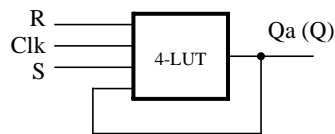
  assign R_g = R & Clk;
  assign S_g = S & Clk;
  assign Qa = ~(R_g | Qb);
  assign Qb = ~(S_g | Qa);

  assign Q = Qa;

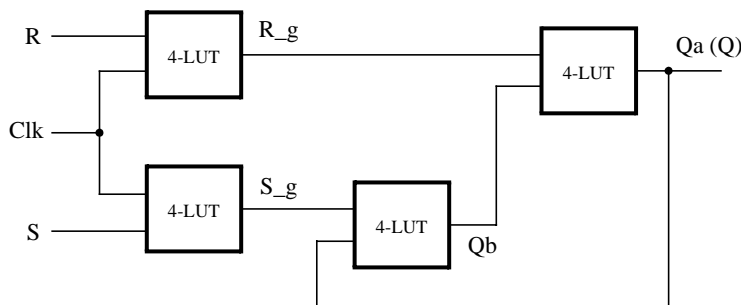
endmodule
```

Figure 2b. Specifying the RS latch by using logic expressions.

Although the latch can be correctly realized in one 4-input LUT, this implementation does not allow its internal signals, such as R_g and S_g , to be observed, because they are not provided as outputs from the LUT. To preserve these internal signals in the implemented circuit, it is necessary to include a *compiler directive* in the code. In Figure 2 the directive `/* synthesis keep */` is included to instruct the Quartus compiler to use separate logic elements for each of the signals R_g , S_g , Qa , and Qb . Compiling the code produces the circuit with four 4-LUTs depicted in Figure 3b.



(a) Using one 4-input lookup table for the RS latch.



(b) Using four 4-input lookup tables for the RS latch.

Figure 3. Implementation of the RS latch from Figure 1.

Create a Quartus project for the RS latch circuit as follows:

1. Create a new project for the RS latch. Select as the target chip the Cyclone IV EP4CE115F29C7, which is the FPGA chip on the Altera DE2 board.

2. Generate a Verilog file with the code in either part *a* or *b* of Figure 2 (both versions of the code should produce the same circuit) and include it in the project.
3. Compile the code. Use the Quartus RTL Viewer tool to examine the gate-level circuit produced from the code, and use the Technology Viewer tool to verify that the latch is implemented as shown in Figure 3b.
4. Create a Vector Waveform File (.vwf) which specifies the inputs and outputs of the circuit. Draw waveforms for the *R* and *S* inputs and use the Quartus Simulator to produce the corresponding waveforms for *R_g*, *S_g*, *Qa*, and *Qb*. Verify that the latch works as expected using both functional and timing simulation.

Part II D-latch

Figure 4 shows the circuit for a gated D latch.

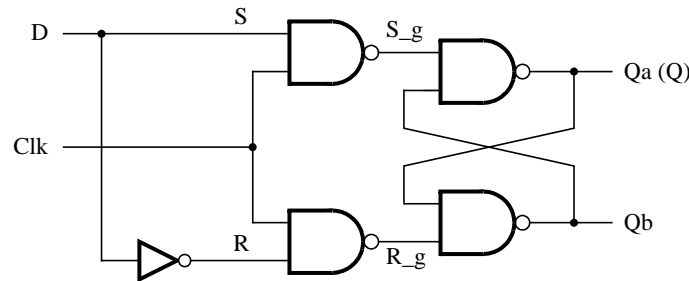


Figure 4. Circuit for a gated D latch.

Perform the following steps:

1. Create a new Quartus project. Generate a Verilog file using the style of code in Figure 2b for the gated D latch. Use the `/* synthesis keep */` directive to ensure that separate logic elements are used to implement the signals *R*, *S_g*, *R_g*, *Qa*, and *Qb*.
2. Select as the target chip the Cyclone VI EP4CE115F29C7 and compile the code. Use the Technology Viewer tool to examine the implemented circuit.
3. Verify that the latch works properly for all input conditions by using functional simulation. Examine the timing characteristics of the circuit by using timing simulation.
4. Create a new Quartus project which will be used for implementation of the gated D latch on the DE2-115 board. This project should consist of a top-level module that contains the appropriate input and output ports (pins) for the DE2-115 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.
5. Recompile your project and download the compiled circuit onto the DE2-115 board.
6. Test the functionality of your circuit by toggling the *D* and *Clk* switches and observing the *Q* output.

Part III Master-Slave D Flip-Flop

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

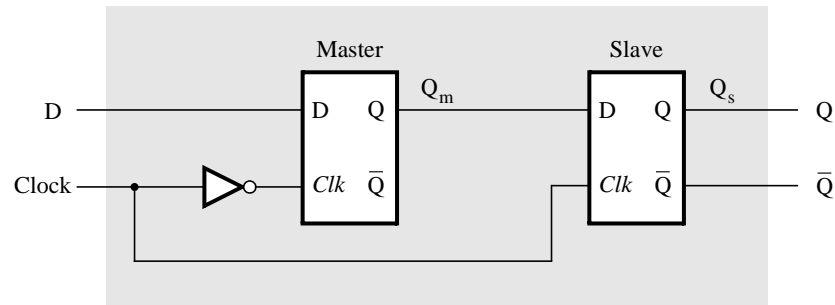


Figure 5. Circuit for a master-slave D flip-flop.

Perform the following:

1. Create a new Quartus project. Generate a Verilog file that instantiates two copies of your gated D latch module from Part II to implement the master-slave flip-flop.
2. Include in your project the appropriate input and output ports for the Altera DE2 board. Use switch SW_0 to drive the D input of the flip-flop, and use SW_1 as the Clock input. Connect the Q output to $LEDR_0$.
3. Compile your project.
4. Use the Technology Viewer to examine the D flip-flop circuit, and use simulation to verify its correct operation.
5. Download the circuit onto the DE2 board and test its functionality by toggling the D and Clock switches and observing the Q output.

Part IV Gated D-Latch, edge triggered D Flip-Flop

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

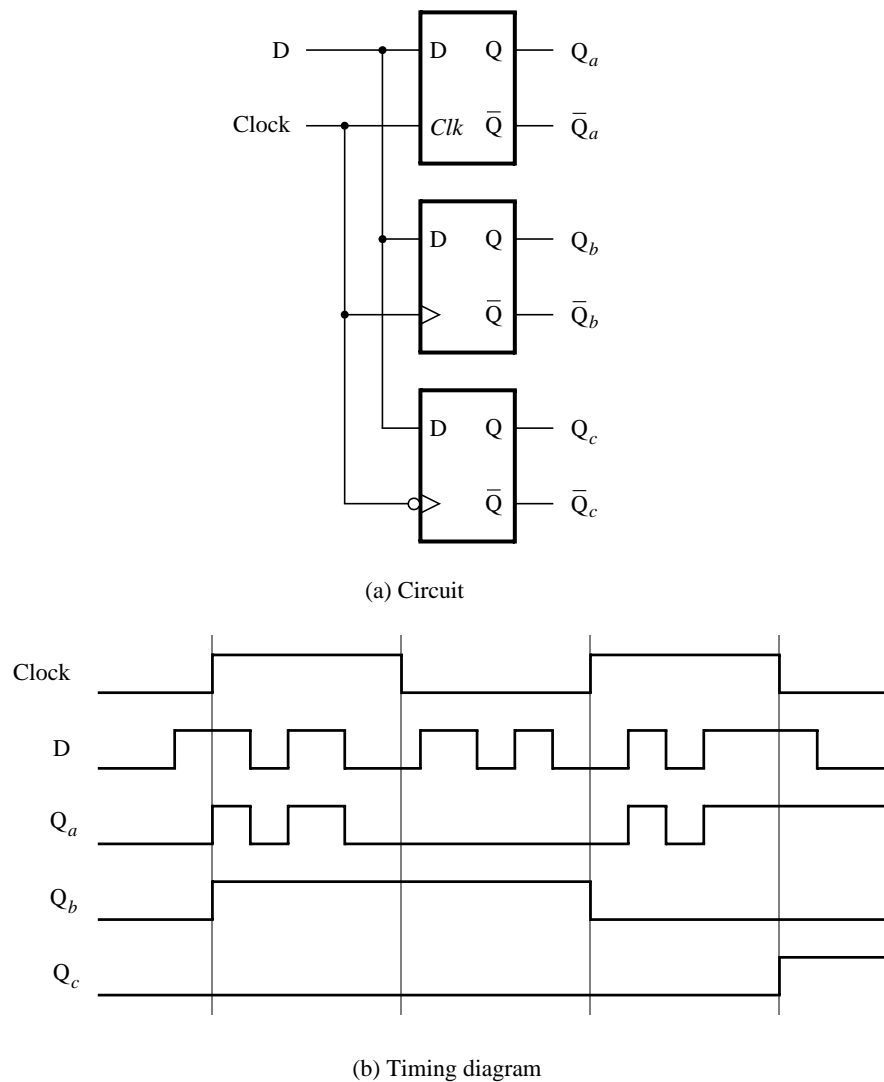


Figure 6. Circuit and waveforms for Part IV.

Implement and simulate this circuit using Quartus software as follows:

1. Create a new Quartus project.
2. Write a Verilog file that instantiates the three storage elements. For this part you should no longer use the `/* synthesis keep */` directive from Parts I to III. Figure 7 gives a behavioral style of Verilog code that specifies the gated D latch in Figure 4. This latch can be implemented in one 4-input lookup table. Use a similar style of code to specify the flip-flops in Figure 6.
3. Compile your code and use the Technology Viewer to examine the implemented circuit. Verify that the latch uses one lookup table and that the flip-flops are implemented using the flip-flops provided in the target FPGA.
4. Create a Vector Waveform File (.vwf) which specifies the inputs and outputs of the circuit. Draw the inputs *D* and *Clock* as indicated in Figure 6. Use functional simulation to obtain the three output signals. Observe the different behavior of the three storage elements.

```

module D_latch (D, Clk, Q);
    input D, Clk;
    output reg Q;

    always @ (D, Clk)
        if (Clk)
            Q = D;
endmodule

```

Figure 7. A behavioral style of Verilog code that specifies a gated D latch.

Part V Hexadecimal value Loader

We wish to display the hexadecimal value of a 8-bit number A on the two 7-segment displays, $HEX8 - 7$. We also wish to display the hex value of a 8-bit number B , C and D on the two 7-segment displays, $HEX6 - 5$, $HEX4 - 3$, and $HEX2 - 1$, respectively. The values of A , B , C and D are inputs to the circuit which are provided by means of switches SW_{7-0} . This is to be done by first setting the switches to the value of A , then setting the switches to the value of B , C and D in succession; therefore, the value of A , B and C must be stored in the circuit.

1. Create a new Quartus project which will be used to implement the desired circuit on the Altera DE2 board.
2. Write a Verilog file that provides the necessary functionality. Use KEY_0 as an active-low asynchronous reset, and use KEY_1 as a clock input.
3. Include the Verilog file in your project and compile the circuit.
4. Assign the pins on the FPGA to connect to the switches and 7-segment displays, as indicated in the User Manual for the DE2-115 board.
5. Recompile the circuit and download it into the FPGA chip.
6. Test the functionality of your design by toggling the switches and observing the output displays.