Laboratory Exercise 6 - Week 1
Latches and Flip-flops

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

Part I

Intel 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. A style of VHDL code that uses logic expressions to describe this circuit is given in Figure 2. 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. The operation of the RS latch is behaviorally described in the characteristic table shown in Table 1.

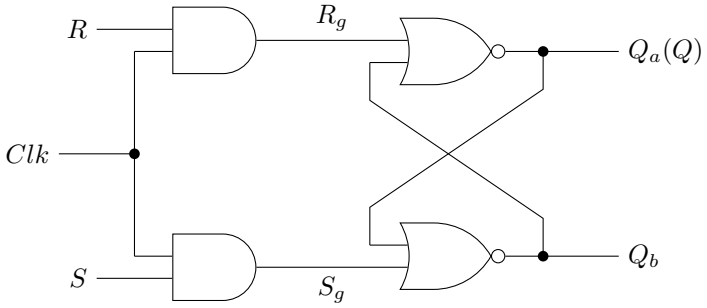

Figure 1: A gated RS latch circuit.

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3
4  ENTITY part1 IS
5      PORT ( Clk, R, S : IN STD_LOGIC;
6             Q : OUT STD_LOGIC);
7  END part1;
8
9  ARCHITECTURE Structural OF part1 IS
10     SIGNAL R_g, S_g, Qa, Qb : STD_LOGIC ;
11     ATTRIBUTE KEEP : BOOLEAN;
12     ATTRIBUTE KEEP OF R_g, S_g, Qa, Qb : SIGNAL IS TRUE;
13 BEGIN
14     R_g <= R AND Clk;
15     S_g <= S AND Clk;
16     Qa <= NOT (R_g OR Qb);
17     Qb <= NOT (S_g OR Qa);
18     Q <= Qa;
19 END Structural;
```

Figure 2: 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 KEEP is included by using a VHDL ATTRIBUTE statement 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.

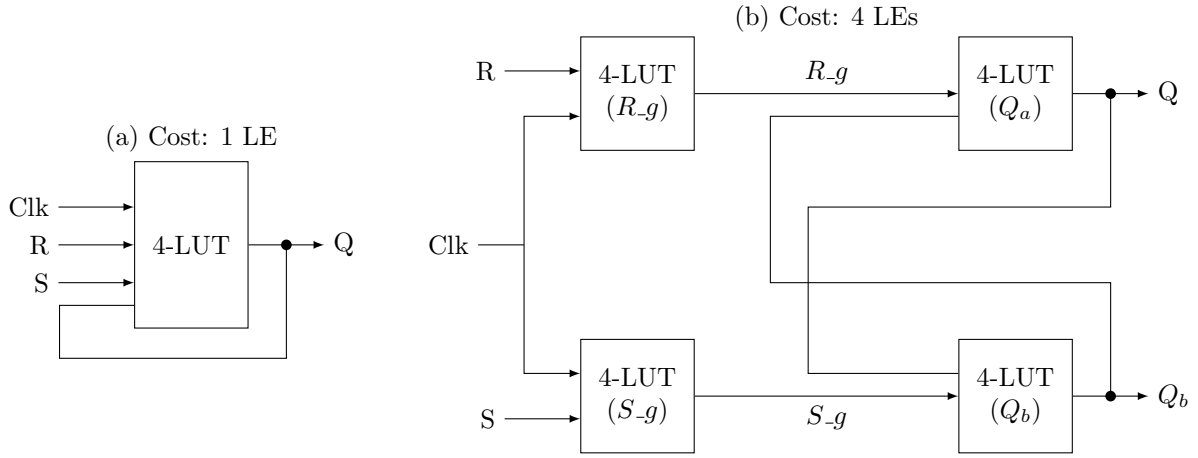

Figure 3: Implementation of the RS latch from Figure 1.

Clk	R	S	Q(t+1)	Description
0	x	x	Q(t)	Hold
1	0	0	Q(t)	Hold
1	0	1	1	Set
1	1	0	0	Reset
1	1	1	x	Indeterminate

Table 1: Characteristic table of the Gated RS Latch.

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

1. Create a new Quartus project for your DE0-CV board.
2. **First, implement the circuit using Schematic Design.** Draw the circuit using the Block Editor and appropriate logic gates.
3. Compile the schematic design and perform a functional simulation to verify its correctness. Ensure the behavior matches the expected RS latch operation.
4. **Next, implement the circuit using Structural VHDL Design.**
5. Generate a VHDL file with the code in Figure 2 and include it in the project.
6. Compile the code. Use the Quartus RTL Viewer tool to examine the gate-level circuit produced from the code, and use the Technology Map Viewer tool to verify that the latch is implemented as shown in Figure 3b.
7. Simulate the behavior of your VHDL code by using the simulation feature provided in the Quartus software. First, create a vector waveform file (*.vwf) using the Quartus software to specify the inputs and outputs of your circuit. Then, use the commands available in the Quartus Simulation Waveform Editor tool to run a simulation of the circuit. The resulting waveforms should look similar to those shown in Figure 4.

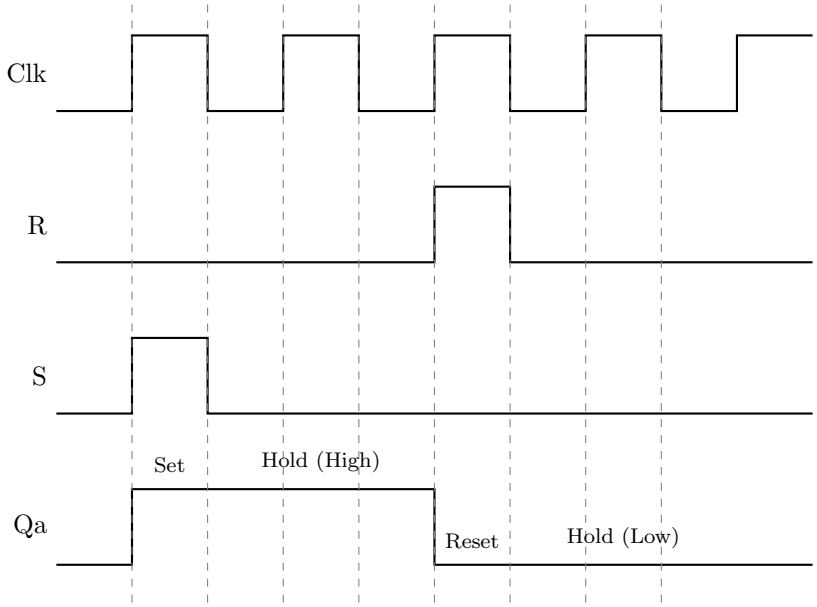

Figure 4: Expected waveforms for the RS latch.

Part II

Figure 5 shows the circuit for a gated D latch. Its corresponding characteristic table is provided in Table 2.

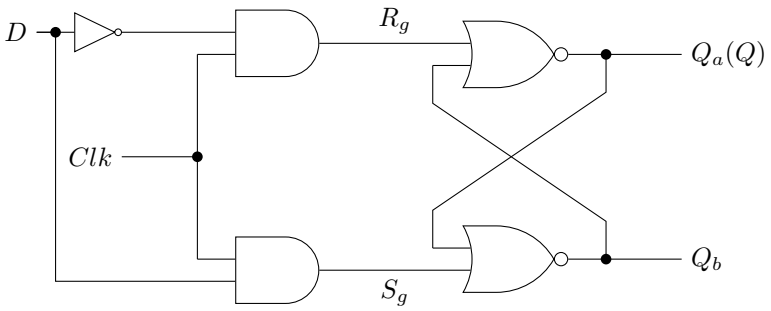

Figure 5: Circuit for a gated D latch.

Clk	D	Q(t+1)	Description
0	x	Q(t)	Hold
1	0	0	Reset
1	1	1	Set

Table 2: Characteristic table of the Gated D Latch.

Perform the following steps:

1. Create a new Quartus project.
2. **First, implement the circuit using Schematic Design.** Draw the circuit using the Block Editor.
3. Compile the schematic design and perform a functional simulation to verify its correctness.
4. **Next, implement the circuit using Structural VHDL Design.**

5. Modify the VHDL code from Part I to implement the gated D latch. Use the KEEP directive to ensure that separate logic elements are used to implement the signals R, S_g, R_g, Qa, and Qb.
6. Compile your project and then use the Technology Map Viewer tool to examine the implemented circuit.
7. 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.
8. Create a new Quartus project which will be used for implementation of the gated D latch on your DE0-CV board. This project should consist of a top-level module that contains the appropriate input and output ports (pins) for your board. Instantiate your latch in this top-level module. Use switch SW0 to drive the D input of the latch, and use SW1 as the Clk input. Connect the Q output to LEDR0.
9. Include the required pin assignments and then compile your project and download the compiled circuit onto your DE0-CV board.
10. Test the functionality of your circuit by toggling the D and Clk switches and observing the Q output.

Part III

Figure 6 shows the circuit for a master-slave D flip-flop. The characteristic table for this negative-edge triggered flip-flop is shown in Table 3.

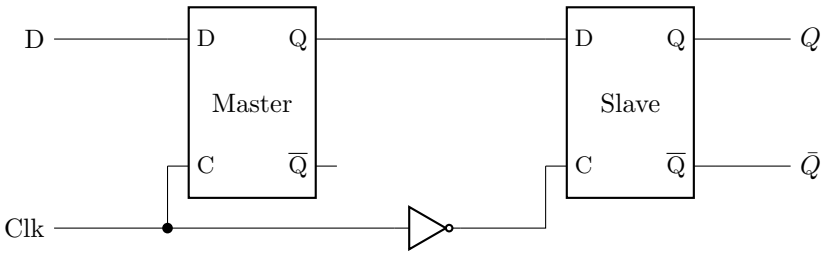

Figure 6: Circuit for a master-slave D flip-flop.

Clk	D	Q(t+1)	Description
↓	0	0	Reset
↓	1	1	Set
0	x	Q(t)	Hold
1	x	Q(t)	Hold

Table 3: Characteristic table of the Master-Slave D Flip-Flop (Negative-Edge Triggered).

Perform the following:

1. Create a new Quartus project.
2. **First, implement the circuit using Schematic Design.** Create a symbol for the Gated D latch from Part II and use it to build the circuit.
3. Compile the schematic design and perform a functional simulation to verify its correctness.
4. **Next, implement the circuit using Structural VHDL Design.**
5. Create a VHDL entity/component for the gated D latch from Part II. Then, write a top-level VHDL file that instantiates two copies of this component to implement the master-slave flip-flop.
6. Include in your project the appropriate input and output ports for your DE0-CV board. Use switch SW0 to drive the D input of the flip-flop, and use SW1 as the Clock input. Connect the Q output to LEDR0.

7. Include the required pin assignments and then compile your project.
8. Use the Technology Viewer to examine the D flip-flop circuit, and use simulation to verify its correct operation.
9. Download the circuit onto your DE0-CV board and test its functionality by toggling the D and Clock switches and observing the Q output.

Updated By: R. Sutthaweekul
Release Date: 2026-01-01