Laboratory Exercise 4 – (Part 1. Simulation Only)

Latches, Flip-flops, and Registers

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

Part I (20%)

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 Parts IV to V 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 is given in the template **part1.vhd**.

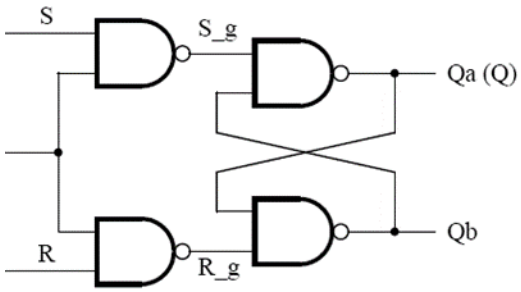


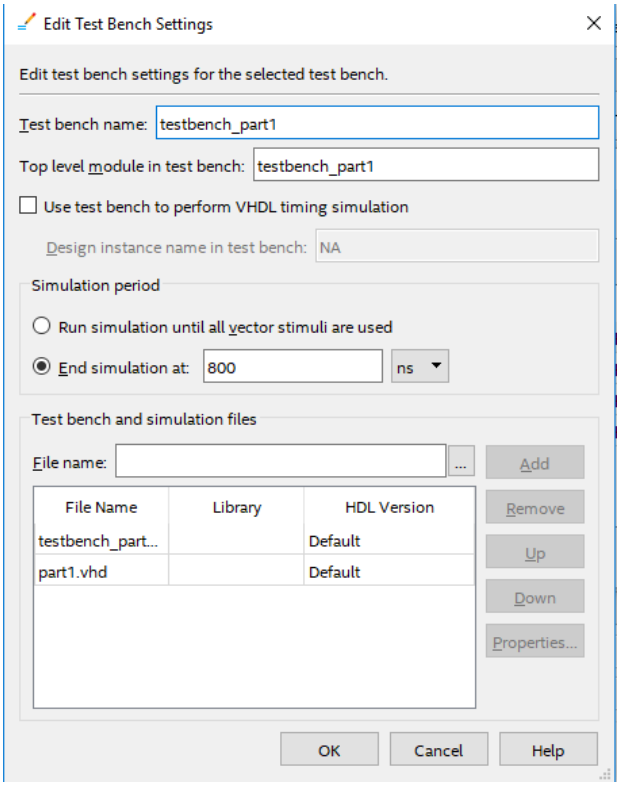
Figure 1. A gated RS latch circuit.

Quartus performs optimisations to use as little FPGA logic as possible. As such, the compiler may remove the internal signals R_g and S_g, because they are not outputs. To preserve these internal signals in the implemented circuit, it is necessary to include a *compiler directive* in the code. (This is the *keep* directive by using a VHDL ATTRIBUTE statement that is included in the template code. Do not worry about remembering this – it will only be of use in this lab). It instructs the Quartus II compiler to use separate logic elements for each of the signals *R_g, S_g,* Q*a,* and Q*b*.

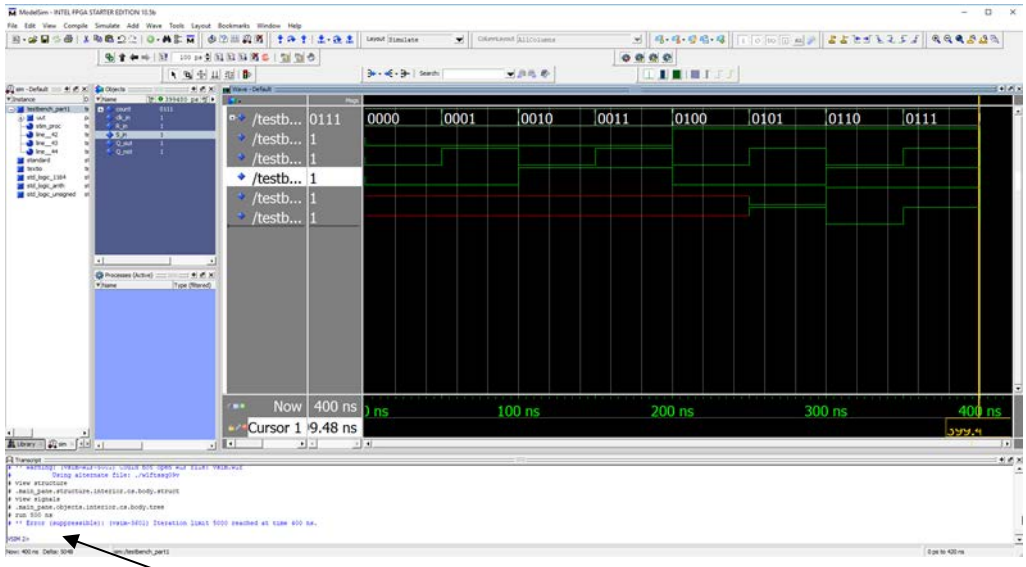
Create a Quartus II project for the RS latch circuit as follows:

1. Create a new project for the RS latch. Select as the target chip the Cyclone V 5CSEMA6F31C6, which is the FPGA chip on the Altera DE1-SOC board. Add the templates **part1.vhd** and **testbench_part1.vhd**
2. See what happens to this latch when run in simulation:
    a. Set **part1.vhd** as the top module (project navigator pane -> files -> right-click on part1.vhd -> set as top module)
    b. Next set up set up modelsim (tools -> options -> eda tool options -> modelsim altera: C:\intelFPGA\16.1\modelsim_ase\win32aloem\). Then add a testbench: **testbench_part1.vhd** (task pane -> RTL simulation. Right click on RTL simulation -> edit settings -> EDA tool settings -> Choose Modelsim-Altera, then compile testbench -> testbenches….

Now make sure to set test bench name and top module in testbench as testbench_part1, and include both testbench_part1.vhd and part1.vhd and run for 800ns. Your test bench settings should look like:

3. 

Note that the simulation does not run for 800ns. If you look at the transcript pane in ModelSim, it specifies there is an error.



Try to come up with an explanation for this.

Part II

Figure 2 shows the circuit for a gated D latch.

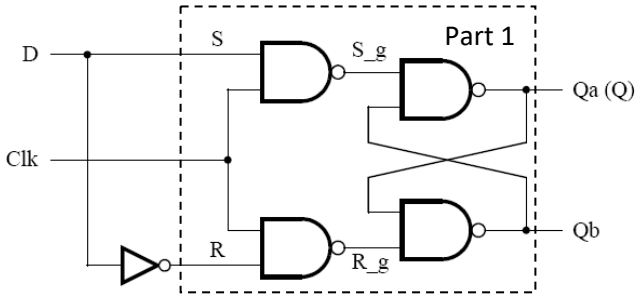
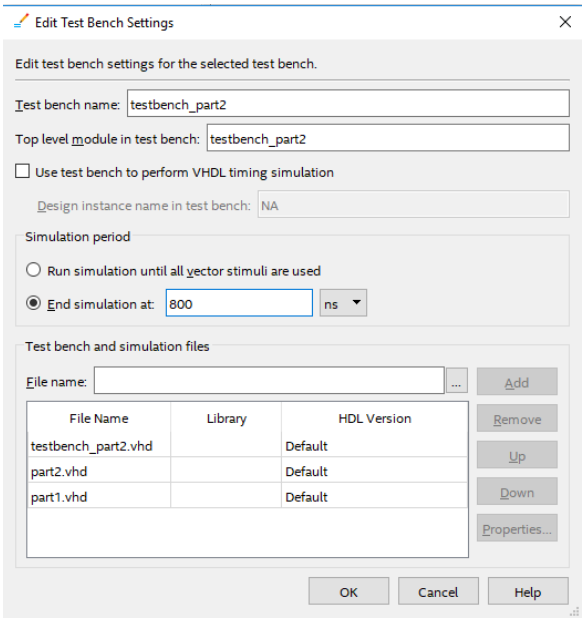


Figure 2. Circuit for a gated D latch.

Perform the following steps:

1. Create a new Quartus II project. Create a VHDL model for part2 (you can start with the template **part2.vhd**). This should instantiate **part1.vhd** as a component (*you need to include **part1.vhd** in your project!)
2. Verify that the latch works properly for all input conditions:
    a. Set **part2.vhd** as the top module (project navigator pane -> files -> right-click on part1.vhd -> set as top module), Next set up set up modelsim (tools -> options -> eda tool options -> modelsim altera: C:\intelFPGA\16.1\modelsim_ase\win32aloem\).
    b. Add a testbench: **testbench_part2.vhd** (task pane -> RTL simulation. Right click on RTL simulation -> edit settings -> EDA tool settings -> Choose Modelsim-Altera, then compile testbench -> testbenches….

       Now make sure to set test bench name and top module in testbench as testbench_part2, and include both **testbench_part2.vhd, part2.vhd and part1.vhd** and run for 800ns. Your testbench settings should look like:



Part III (20%)

Figure 3 shows the circuit for a master-slave D flip-flop.

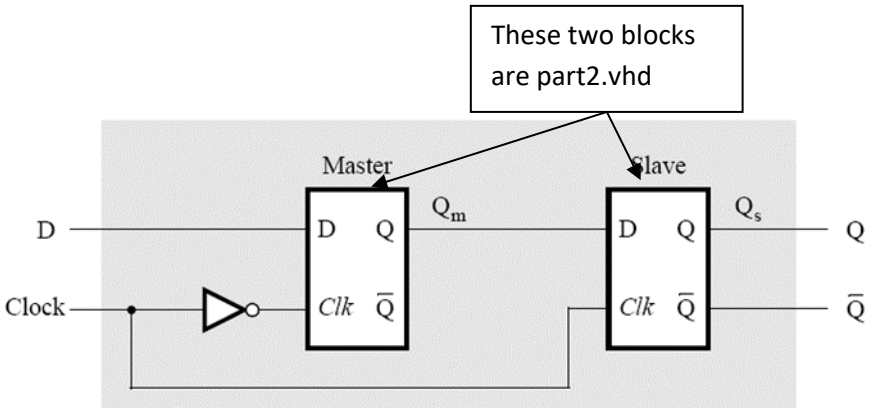
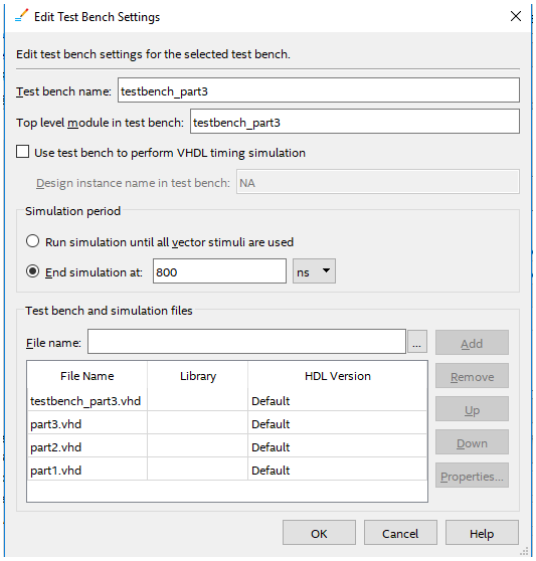


Figure 3. Circuit for a master-slave D flip-flop.

Perform the following:

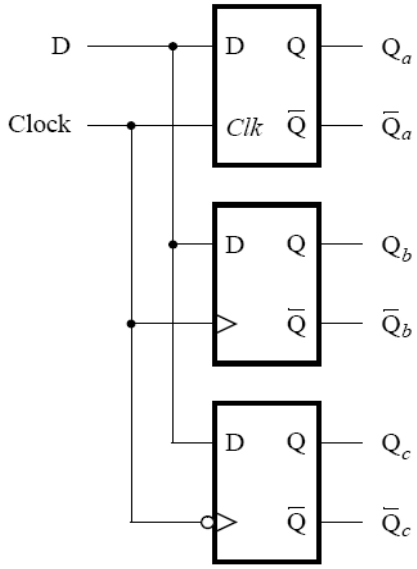
1. Create a new Quartus II project. Complete the template **part3.vhd**. This should instantiate two copies of your gated D latch (**part2.vhd**) to implement the master-slave flip-flop. (*you need to include part2.vhd in your project. Also, because this uses part1.vhd, include this as well!)
3. Verify that the flip-flop works properly for all input conditions:
    a. Set **part3.vhd** as the top module (project navigator pane -> files -> right-click on part1.vhd -> set as top module), Next set up set up modelsim (tools -> options -> eda tool options -> modelsim altera: C:\intelFPGA\16.1\modelsim_ase\win32aloem\).
    b. Add a testbench: **testbench_part3.vhd** (task pane -> RTL simulation. Right click on RTL simulation -> edit settings -> EDA tool settings -> Choose Modelsim-Altera, then compile testbench -> testbenches….

    Now make sure to set test bench name and top module in testbench as **testbench_part3**, and include both **testbench_part3.vhd, part3.vhd, part2.vhd** and **part1.vhd** and run for 800ns. Your test bench settings should look like:



Part IV (20%)

Figure 4 shows a circuit with three different storage elements: a gated D latch, a positive-edge triggered D flipflop, and a negative-edge triggered D flip-flop.



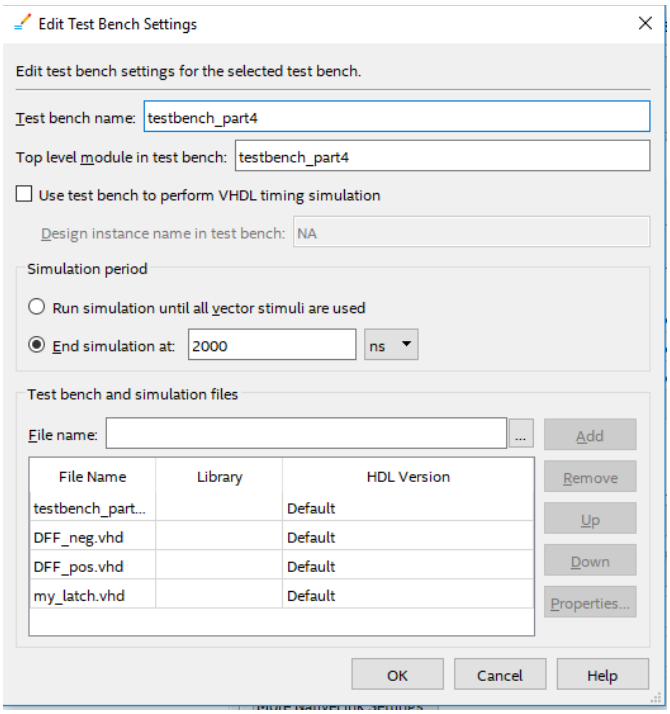
(a) Circuit

Figure 4. Circuit and waveforms for Part IV.

Implement and simulate this circuit using Quartus II software as follows:

1. Create a new project. Add the templates:
    a. **my_latch.vhd**,
    b. **dff_neg.vhd,**
    c. **dff_pos.vhd**,
    d. **testbench_part4.vhd**
2. Study **my_latch.vhd**, **dff_neg.vhd, dff_pos.vhd**. These are written in high –level VHDL and no longer use the *keep* directive (that is, the VHDL ATTRIBUTE statement) from Parts I to III. This is the common way to specify latches and flip flops. You should remember these.
4. Compare the behavior of these latches and flip-flop using simulation:
    a. Set **any module apart from the testbench** as the top module (project navigator pane -> files -> right-click on part1.vhd -> set as top module), Next set up set up modelsim (tools -> options -> eda tool options -> modelsim altera: C:\intelFPGA\16.1\modelsim_ase\win32aloem\).
    b. Add a testbench: **testbench_part4.vhd** (task pane -> RTL simulation. Right click on RTL simulation -> edit settings -> EDA tool settings -> Choose Modelsim-Altera, then compile testbench -> testbenches….

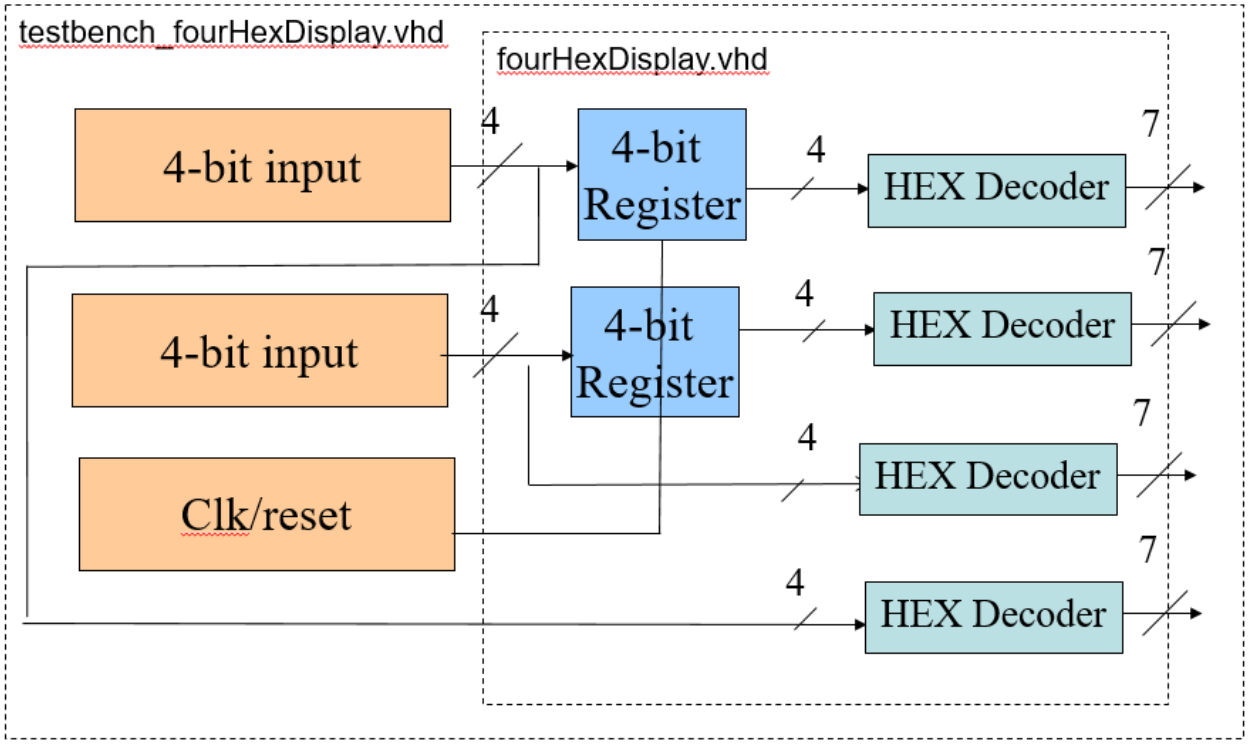
    Now make sure to set test bench name and top module in testbench as **testbench_part4**, and include both **testbench_part4.vhd, my_latch.vhd**, **dff_neg.vhd, dff_pos.vhd** and run for 2000ns. Your test bench settings should look like:



Prepare an explanation as to how these outputs compare and whether the output is as expected. You should discuss this with a demonstrator in the lab.

Part V

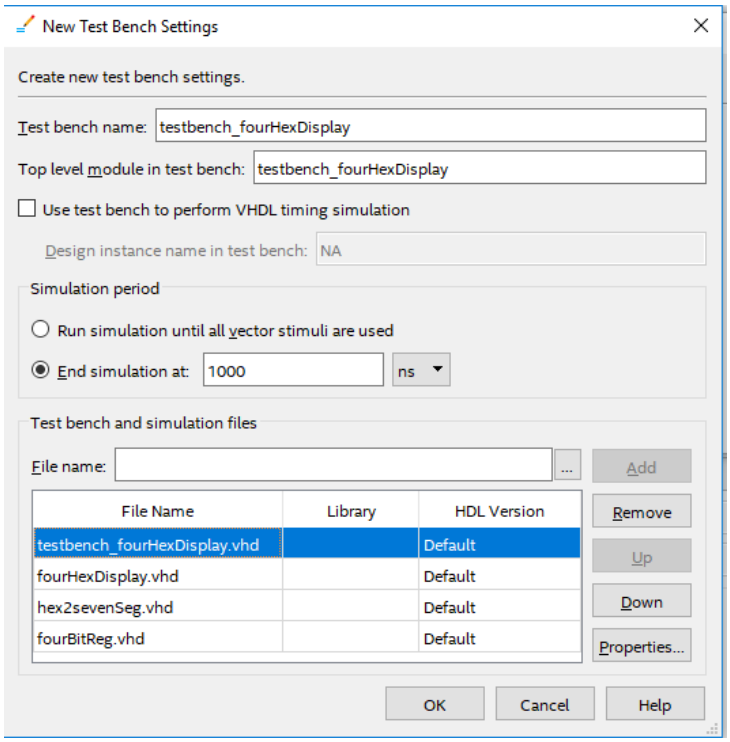
We wish to display the hexadecimal value of an 8-bit number A on the two 7-segment displays, $HEX3-2$. We also wish to display the hex value of an 8-bit number B on the four 7-segment displays, $HEX1-0$. The values of A and B are inputs to the circuit which are provided by means of switches SW_{7-0}. This is to be done by first setting the switches to the value of A and then setting the switches to the value of B; therefore, the value of A must be stored in the circuit.



1. Write VHDL files that provide the necessary functionality:
    a. A hex to seven-seg decoder (**hex2sevenSeg.vhd**). You have done something very similar to this in lab 2. You can use your old code or simply re-create this using a case statement.
    b. An entity for a 4-bit register with asynchoronous reset (**fourBitReg.vhd**)
    c. An entity to connect up the various components (**fourHexDisplay.vhd**)

2. Simulate this circuit:
    a. Set: **fourHexDisplay.vhd** as the top module (project navigator pane -> files -> right-click on part1.vhd -> set as top module), Next set up set up modelsim (tools -> options -> eda tool options -> modelsim altera: C:\intelFPGA\16.1\modelsim_ase\win32aloem\).
    b. Then add a testbench: **testbench_fourHexDisplay.vhd** (task pane -> RTL simulation. Right click on RTL simulation -> edit settings -> EDA tool settings -> Choose Modelsim-Altera, then compile testbench -> testbenches….

    Now make sure to set test bench name and top module in testbench as testbench_fourHexDisplay, and include both **hex2sevenSeg.vhd, fourBitReg.vhd, fourHexDisplay.vhd**) and run for 800ns. Your test bench settings should look like:



c. Modify the testbench to test the functionality of your design by toggling the inputs and observing the output.