Lund university

Exercise 2 in VHDL

EITF65

1 Combinational logic with VHDL

Creating a flip-flop

During this exercise a flip-flop will be constructed by using VHDL instead of discrete components. A flip-flop has three main building blocks, two latches and one control block. The exercise is therefore divided into three parts. In the first part a latch will be created, in the second part the control block will be created and in the last part they will be connected to form a flip-flop.

Setting up Vivado 2016.1

Before starting Vivado 2016.1 three files should be downloaded (`TB_flipflop.vhd`, `TB_latch.vhd` and `TB_control.vhd`). These files will be used when simulating the behaviour of your design. They can be found on the course website under exercises. See below for a direct link. `https://www.eit.lth.se/fileadmin/eit/courses/eitf65/exercises/ex2_files.zip`

The files should be placed in a folder under the path shown below:
C:\users\<login-id>\Program

Open Vivado 2016.1 and create a new project. The project needs to placed under the following path:
C:\users\<login-id>\Program\"Your folder"

If the project is not placed at the specified location the simulations will not work.

Add the downloaded files and select the correct FPGA (xc7a100tcsg324-1). When the project is created add a new design source and name it `latch` (it has to be this name otherwise the simulation will not work). If you do not know how to do this watch the following video:
`https://youtu.be/MJ8dAxzsH04`.

Task 1: The latch

A state diagram of the latch is depicted in Figure 1.

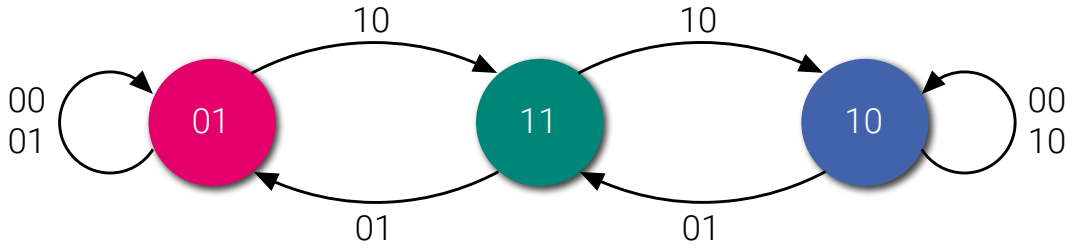

Figure 1 – State diagram for the latch.

As in the laboratory exercise 3, an input signal, ϕ (phi), should be used to enable the inputs `x1` and `x2`. The equations that can be derived from the state diagram is shown below.

$$f_1 = ((x_1 \cdot \phi)' \cdot f_2)' \tag{1}$$

$$f_2 = ((x_2 \cdot \phi)' \cdot f_1)' \tag{2}$$

The latch's ports should be named as in Table 1. If the ports do not have the correct names the provided simulation sources will not work.

Table 1 – The ports of the latch.

Port name	**Direction**	**Size**
`phi`	`input`	1
`x1`	`input`	1
`x2`	`input`	1
`f1`	`output`	1
`f2`	`output`	1

When the design is done set the file `TB_latch.vhd` as top. This is done by right clicking on the file (in the *Source* window) and select *Set as Top*. Click the following link to see a video of how this is done: `https://youtu.be/Suidvhl_sGM`

Vivado has many useful tools to help the design flow. One of these tools is the schematic view. This lets the designer view the code as a schematic. To access

this feature go to the *Flow Navigator*, which is found to the left. Click on *Open Elaborate Design*. When utilizing this feature it is important to select the design source you want to view as top. This is done in same way as when setting a simulation source as top.

When you are sure that your design is behaving as intended you can move on to the next task.

Task 2: The control circuit

To prevent race conditions while using the flip-flop a control circuit is required. A state diagram of this circuit can be seen in Figure 2.

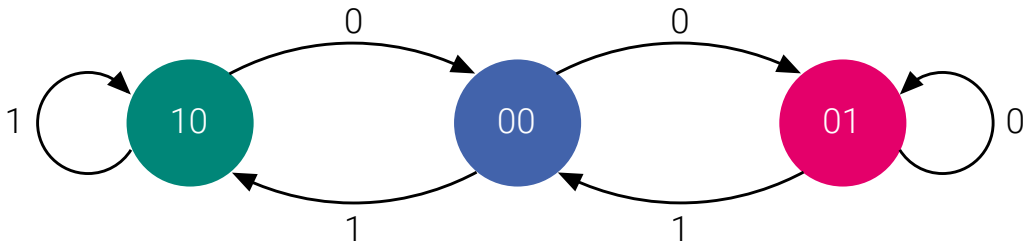

Figure 2 – State diagram for control circuit.

The equations derived from the state diagram can be seen below.

$$\phi_1 = clk \cdot \phi_2' \tag{3}$$

$$\phi_2 = clk' \cdot \phi_1' \tag{4}$$

The control block should have three ports, see Table 2. For the same reasons as before it is important that the ports have the correct names.

Table 2 – The ports of the control block.

Port name	Direction	Size
`clk`	`input`	1
`phi1`	`output`	1
`phi2`	`output`	1

Add a new design source, name it `control` and implement the control circuit. And yet again, the name of the design source is important.

If you want to view your design as a schematic do not forget to set your new source as top.

Simulate your design to verify that the control circuit is working as desired. Do not forget to set `TB_control.vhd` as the top of the simulation sources. Move on to task 3 when you have convinced yourself that the design is working.

Task 3: The flip-flop

In the last part the previously created blocks will be connected. If this is done correctly it will create a flip-flop that is triggered on the negative edge of the `clk`-signal. See Figure 3 for a schematic of how the blocks should be connected.

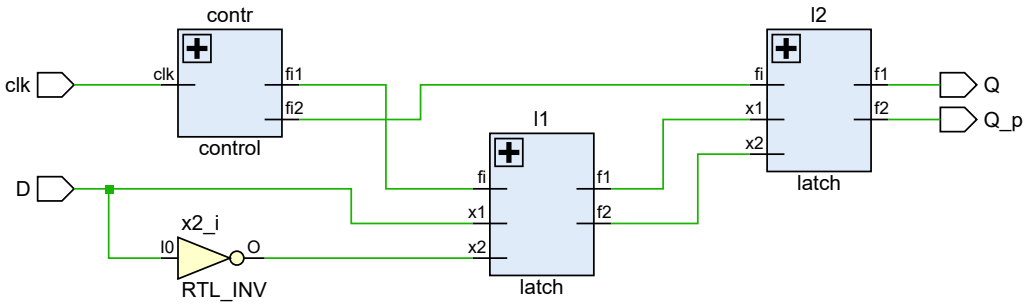

Figure 3 – Schematic view of the flip-flop.

Table 3 – The ports of the flip-flop.

Port name	Direction	Size
`clk`	`input`	1
`D`	`input`	1
`Q`	`output`	1
`Q_p`	`output`	1

Add a new design source to the project. The name should be `flipflop`. The ports are listed in Table 3. By now you should know that it is vital that the names are correct, hence there is no need to point that out again.

When the blocks are connected, it is a good idea to open the schematic view of the design (remeber to set `flipflop.vhd` as top). If the schematic is correct set `TB_flipflop.vhd` as top and start the simulation.

2 Solutions

Task 1: The latch

```
library ieee;
use ieee.std_logic_1164.all;

entity latch is
    port(
        phi : in  std_logic;
        x1  : in  std_logic;
        x2  : in  std_logic;
        f1  : out std_logic;
        f2  : out std_logic
    );
end latch;

architecture Behavioral of latch is

        signal s_f1 : std_logic;
        signal s_f2 : std_logic;

begin
    -- Outputs
    f1 <= s_f1;
    f2 <= s_f2;

    s_f1 <= not (not (phi and x1) and s_f2);

    s_f2 <= not (not (phi and x2) and s_f1);

end Behavioral;
```

Testbench output

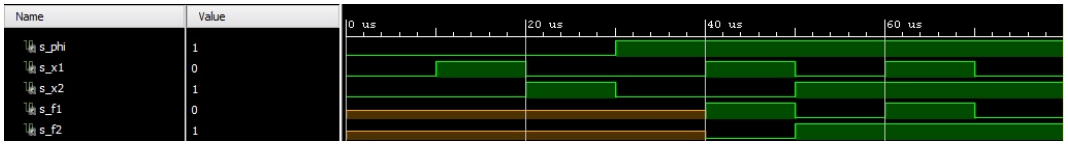

Figure 4 – Testbench output for the latch.

Task 2: The control block

```
library ieee;
use ieee.std_logic_1164.all;

entity control is
        port(
                clk  : in  std_logic;
                phi1 : out std_logic;
                phi2 : out std_logic
        );
end control;

architecture Behavioral of control is

        signal s_phi1 : std_logic;
        signal s_phi2 : std_logic;

begin

        -- Outputs
        phi1   <= s_phi1;
        phi2   <= s_phi2;

        s_phi1 <= clk and not s_phi2;
        s_phi2 <= not clk and not s_phi1;

end Behavioral;
```

Testbench output

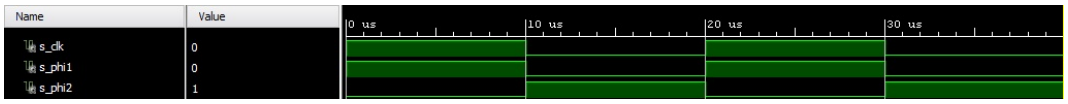

Figure 5 – Testbench output for the control block.

Task 3: The flip-flop

```
library ieee;
use ieee.std_logic_1164.all;

entity flipflop is
        port(
                D   : in  std_logic;
                clk : in  std_logic;
                Q   : out std_logic;
                Q_p : out std_logic
        );
end flipflop;

architecture Behavioral of flipflop is


        component control is
                port(
                        clk  : in  std_logic;
                        phi1 : out std_logic;
                        phi2 : out std_logic
                );
        end component;

        component latch is
                port(
                        phi : in  std_logic;
                        x1  : in  std_logic;
                        x2  : in  std_logic;
                        f1  : out std_logic;
                        f2  : out std_logic
                );
        end component;


        signal s_Q      : std_logic;
        signal s_Q_p    : std_logic;
        signal s_y1     : std_logic;
        signal s_y2     : std_logic;
        signal s_phi1   : std_logic;
        signal s_phi2   : std_logic;
        signal s_not_d  : std_logic;


begin

        -- Outputs
        Q       <= s_Q;
        Q_p     <= s_Q_p;
        s_not_d <= not D;

        contr: control
        port map(
                clk  => clk,
                phi1 => s_phi1,
                phi2 => s_phi2
        );

        l1: latch
        port map(
                phi => s_phi1,
                x1  => D,
                x2  => s_not_d,
                f1  => s_y1,
                f2  => s_y2
        );

        l2: latch
        port map(
                phi => s_phi2,
                x1  => s_y1,
                x2  => s_y2,
                f1  => s_Q,
                f2  => s_Q_p
        );

end Behavioral;
```

Testbench output

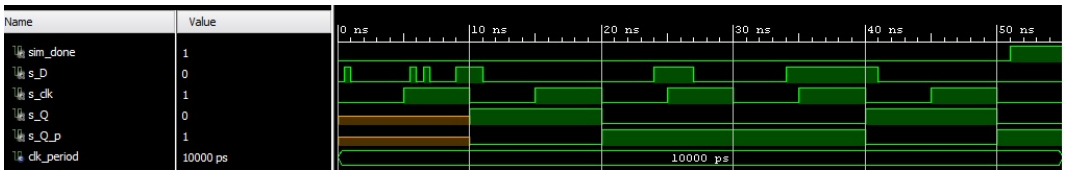

Figure 6 – Testbench output for the flip-flop.