Lab 5 – Counter with External Input

CS 2052 Computer Architecture

Dept. of Computer Science and Engineering, University of Moratuwa

Learning Outcomes

In this lab, we will design a 3-bit counter with an external input. After completing the lab, you will be able to:

- design and develop a 3-bit counter
- count in clockwise and anticlockwise directions based on an external input
- verify its functionality via simulation and on the development board

Introduction

A register that goes through a predetermined sequence of states is called a *counter*. In this lab, we will design a 3-bit counter that can show the sequence of LEDs (dark circles indicate LEDs that are lit) defined in Fig. 1. We will control the direction of counting (clockwise or anticlockwise) based on an external input. When the input button is switched off, we will count in the clockwise direction. When it is switched on, we will count in the anticlockwise direction.

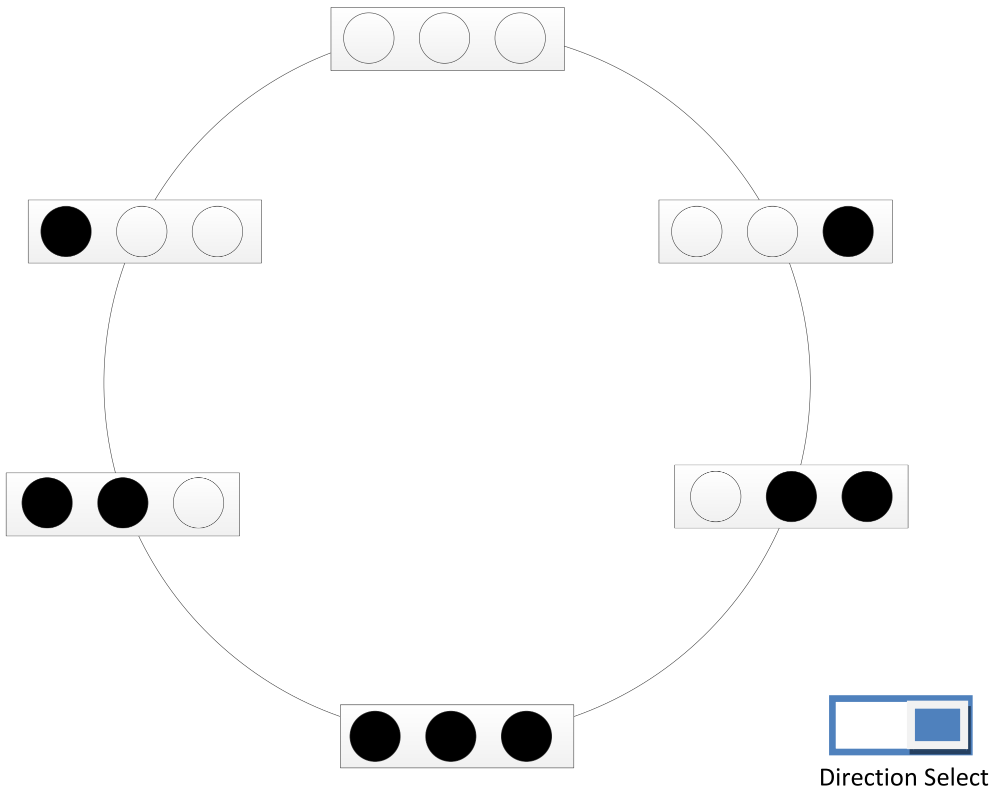


Figure 1 – Counter pattern.

Building the Circuits

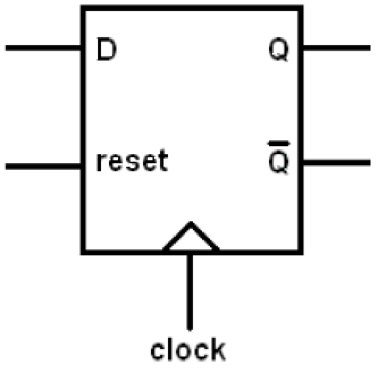
Step 1: Using the Excitation Table of a D Flip-Flop complete the following table.

Q_t			Button	Q_{t+1}			D_2	D_1	D_0
Q_2	Q_1	Q_0		Q_2	Q_1	Q_0			
0	0	0	0	0	0	1			
0	0	0	1	1	0	0			
0	0	1	0	0	1	1			
0	0	1	1	0	0	0			
…	…	…	…	…	…	…			
…	…	…	…	…	…	…			

Identify the inputs to D Flip Flops D_0, D_1, and D_2. Use Karnaugh Maps to simplify the expressions. Interpret the combinations that will not occur as don't' care.

Step 2: Building D Flip Flop.

In VHDL we can build a D Flip-Flop (FF) by defining its behavior. Let us build a D FF like the one below:

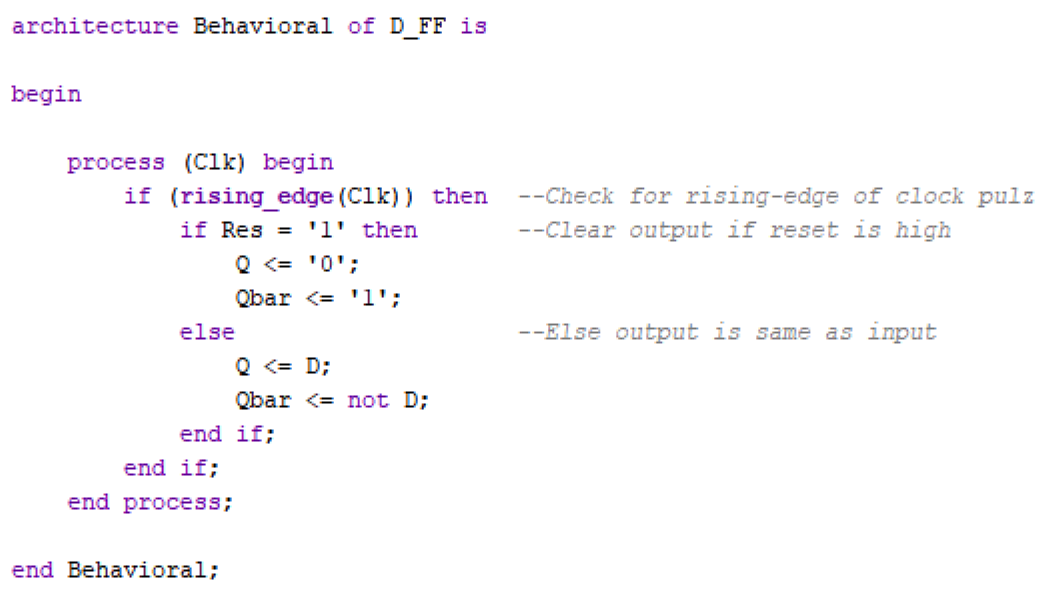


where *D* is the data input, *reset* clears FF to 0 (i.e., resets the value of FF), and FF is driven by a *clock*. *Q* and *Q*bar are the 2 outputs.

So far we have used VHDL to define the behavior of our circuits as a sequence of logic operations (this is referred to as *Structural Modelling*). VHDL also allows us to define how a circuit should behave more abstractly. This style of circuit definition is referred to as *VHDL Behavioral Modelling*. Behavioral modeling describes how the circuit should behave using high-level programming constructs such as variables, conditions, and procedures. Given a circuit defined using behavioral modeling, the VHDL synthesizer tool will decide the actual circuit implementation.

Crete a new VHDL file and name it as **D_FF**. Set the inputs as **D**, **Res**, and **Clk** and outputs as **Q** and **Qbar**.

Then add the code given in Fig. 2 to define the behavior of the D FF. The `process` is the key structure that defines the behavior of the VHDL model. It defines the functionality of an entity. Here we name the process as `Clk`. Then we detect the rising edge of the clock using `rising_edge()`function. When the clock is high, we either set the output based on the `D` input or reset it to `0`, if reset input is high (`if Ref = '1'`).



```vhdl
architecture Behavioral of D_FF is

begin

    process (Clk) begin
        if (rising_edge(Clk)) then  --Check for rising-edge of clock pulz
            if Res = '1' then       --Clear output if reset is high
                Q <= '0';
                Qbar <= '1';
            else                    --Else output is same as input
                Q <= D;
                Qbar <= not D;
            end if;
        end if;
    end process;

end Behavioral;
```

Figure 2 – Behavioral model of D flip-flop.

Simulate the D_FF using XSim and make sure it functions correctly. Name the simulation files as **D_FF_Sim**.

Step 3: Building the slow down clock.

As we want our FF to respond based on the clock signal (i.e., at rising edge) we need to use the internal clock of BASYS 3. However, the internal clock on BASYS 3 runs at 100 MHz. At this rate, we will not be able to see the changes in LEDs. Therefore, we need to slow down our clock input to at least a few hundred milliseconds. While the clock on BASYS 3 can be slow down to some extent – by changing the parameters in Design Constraints File – we can slow it down to a few Hz from the initial 100 MHz.

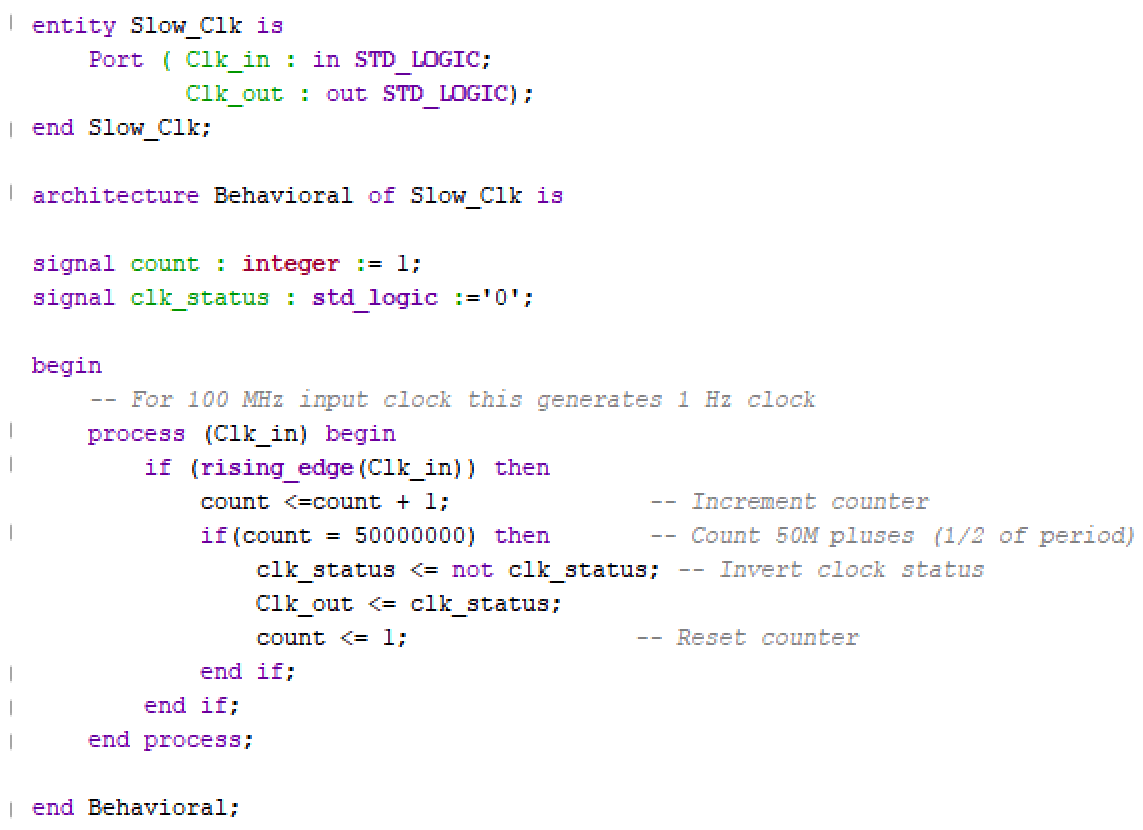
Therefore, we use a slowdown counter, which can emit an output only when a certain number of changes in the input is detected. For example, we can build an entity that emits 1 Hz clock pulses for every 100 million clock pulses generated by the 100 MHz clock. Fig. 3 shows the VHDL behavioral model for a 100 MHz to 1 Hz slow-down counter. In this example, we change the output clock plus every 0.5 seconds with a period of 1 second. `count` and `clk_status` are 2 variables where `count` is an integer set to 1 and `clk_status` is a logic value set to 0.

Create a new VHDL file and name it as **Slow_Clk**. Defines the input and output as **Clk_in** and **Clk_out**, respectively.

Simulate the Slow_Clk and make sure it functions correctly. Name the simulation files as **Slow_Clk_Sim**. For simulation purposes, you may want to reduce 50 million to a smaller value like 5 or 10.

Step 4: Building the counter.

Create a VHDL file and name it as **Counter**. Label in the inputs as **Dir** (i.e., Direction Select), **Res** (reset), and **Clk** (clock). Let the output **Q** be a 3-bit bus.



```
entity Slow_Clk is
    Port ( Clk_in : in STD_LOGIC;
           Clk_out : out STD_LOGIC);
end Slow_Clk;

architecture Behavioral of Slow_Clk is

signal count : integer := 1;
signal clk_status : std_logic :='0';

begin
    -- For 100 MHz input clock this generates 1 Hz clock
    process (Clk_in) begin
        if (rising_edge(Clk_in)) then
            count <=count + 1;              -- Increment counter
            if(count = 50000000) then       -- Count 50M pluses (1/2 of period)
                clk_status <= not clk_status; -- Invert clock status
                Clk_out <= clk_status;
                count <= 1;                 -- Reset counter
            end if;
        end if;
    end process;

end Behavioral;
```

Figure 3 – Behavioral model of slowdown counter.

Connect input **Clk** signal to **Clk_in** in **Slow_Clk** and internal output **Clk_slow** to **Clk_out**. Clocks connected to all 3 FFs should be based on **Clk_out**.

Build the counter based on the Boolean expressions you derived in Step 1.

Following is part of the VHDL code to build the counter. You need to fill the missing pieces. This is given as a reference only so you can understand the structure of the solution.

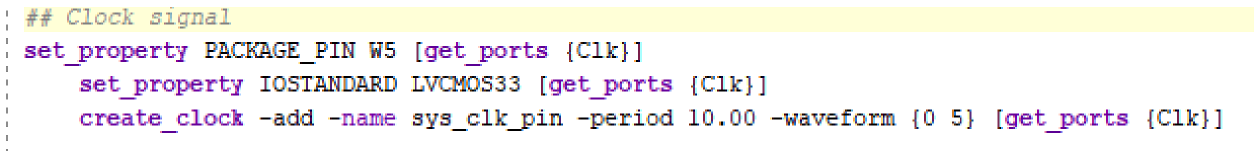
```
architecture Behavioral of Counter is

    component D_FF
    port (
        D : in STD_LOGIC;
        Res: in STD_LOGIC;
        Clk : in STD_LOGIC;
        Q : out STD_LOGIC;
        Qbar : out STD_LOGIC);
    end component;

    component Slow_Clk
    port (
        Clk_in : in STD_LOGIC;
        Clk_out: out STD_LOGIC);
    end component;


    signal D0, D1, D2 : std_logic;  -- Internal signals
    signal Q0, Q1, Q2 : std_logic;  -- Internal signals
    signal Clk_slow : std_logic;    -- Internal clock

begin

    Slow_Clk0 : Slow_Clk
        port map (
        Clk_in => Clk,
        Clk_out => Clk_slow);

   D0 <= ((not Q2) and (not Dir)) or (Q1 and Dir);
   D1 <= ...      --Fill missing details
   D2 <= ...      --Fill missing details

   D_FF0 : D_FF
        port map (
        D => D0,
        Res => Res,
        Clk => Clk_slow,
        Q => Q0);

   D_FF1 : D_FF
        port map (
        ...);           --Fill missing details

   D_FF2 : D_FF
        port map (
        ...);           --Fill missing details

    Q(0) <= Q0;
    ...                 --Fill missing details

end Behavioral;
```

Verify the functionality of your counter using the simulator. Name the simulation file as **Counter_sim**. For simulation purposes, you may want to reduce counter value in slow-down counter, but make sure to reset it to 50 million before generating the bitstream.

Step 5: Test on BASYS 3.

Connect switch **SW0** as the external input **Dir** and **BTND** as the **Res**. Connect outputs **Q0 - Q2** to LEDs **LD0 - LD2**.

To enable the internal clock, you need to uncomment the relevant lines on Design Constraints File as in Fig. 4.



```
## Clock signal
set_property PACKAGE_PIN W5 [get_ports {Clk}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Clk}]
    create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports {Clk}]
```

Figure 4 – Using the clock on BASYS 3.

Make sure to use **Clk** by replacing the default values in all 3 lines. Here `10.00 -waveform {0 5}` indicates a 100 MHz clock (i.e., delay of 10 ps) and signal high to low ratio is 0.5.

Generate the programming file (i.e., bitstream) and load it to the BASYS 3 board.

Change the switches on the board and verify the functionality of your circuit.

Demonstrate the circuit to the instructor and get the Lab Completion Log singed.

Step 6: Lab Report

You need to submit a report for this lab. Your report should include the following:

- Student name, index number, and group. Do not attach a separate front page
- State the assigned lab task in a few sentences
- Completed table in Step 1. Simplified expressions for D_0, D_1, and D_2 using Karnaugh Maps
- All VHDL codes
- All timing diagrams
- Conclusions from the lab.

Submit the lab report at the beginning of the next lab.

Bibliography

- Modeling Latches and Flip-flops, Xilinx, "Vivado Tutorial – Lab Workbook," 2015.
- Modeling Registers and Counters, Xilinx, "Vivado Tutorial – Lab Workbook," 2015.

Prepared By

- Dilum Bandara, PhD – Mar 26, 2014.
- Updated on Oct 11, 2018