KATHMANDU UNIVERSITY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

LAB REPORT 05

EEEG-321

Department of Electrical and Electronics Engineering

By:

Samyam Shrestha (31056)

To:

Santosh Shaha Sir

Date:

21st May, 2024

Title: Write VHDL code for 4 bit Counter

Requirements: Vivado Software, Laptop

1) 4 bit Counter

VHDL Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity fourbitcounter_56 is
   Port (
      clk : in std_logic;   -- Clock signal
      rst : in std_logic;   -- Asynchronous reset signal
      clr : in std_logic;   -- Clear signal (used to increment the counter)
      q   : out std_logic_vector(3 downto 0) -- Output of the counter
   );
end fourbitcounter_56;

architecture behavioral of fourbitcounter_56 is
   signal count: std_logic_vector(3 downto 0) := "0000"; -- Counter signal
begin
   process (clk, rst)
   begin
      if (rst = '1') then
         count <= "0000"; -- Asynchronous reset
      elsif rising_edge(clk) then
         if (clr = '1') then
            count <= count + 1; -- Increment the counter
         end if;
      end if;
   end process;
   q <= count;
end behavioral;
```

Test Bench code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity fourbitcounter_56_tb is
end fourbitcounter_56_tb;

architecture Behavioral of fourbitcounter_56_tb is
   component fourbitcounter_56
      Port(
```

```
            clk : in std_logic;
            rst : in std_logic;
            clr : in std_logic;
            q   : out std_logic_vector(3 downto 0)
        );
    end component;

    signal clk : std_logic := '0';
    signal rst : std_logic := '0';
    signal clr : std_logic := '0';
    signal q   : std_logic_vector(3 downto 0);

    constant clk_period : time := 10 ns;
    begin
    uut: fourbitcounter_56
        port map (
            clk => clk,
            rst => rst,
            clr => clr,
            q   => q
        );
    clk_process :process
    begin
        while true loop
            clk <= '0';
            wait for clk_period / 2;
            clk <= '1';
            wait for clk_period / 2;
        end loop;
    end process;
    stimuli: process begin
        rst <= '1';
        wait for 20 ns;
        rst <= '0';
        wait for 20 ns;

        clr <= '1';
        wait for 100 ns;
        clr <= '0';
        wait for 100 ns;
        wait;
    end process;
end Behavioral;
```

RTL Schematic Diagram:

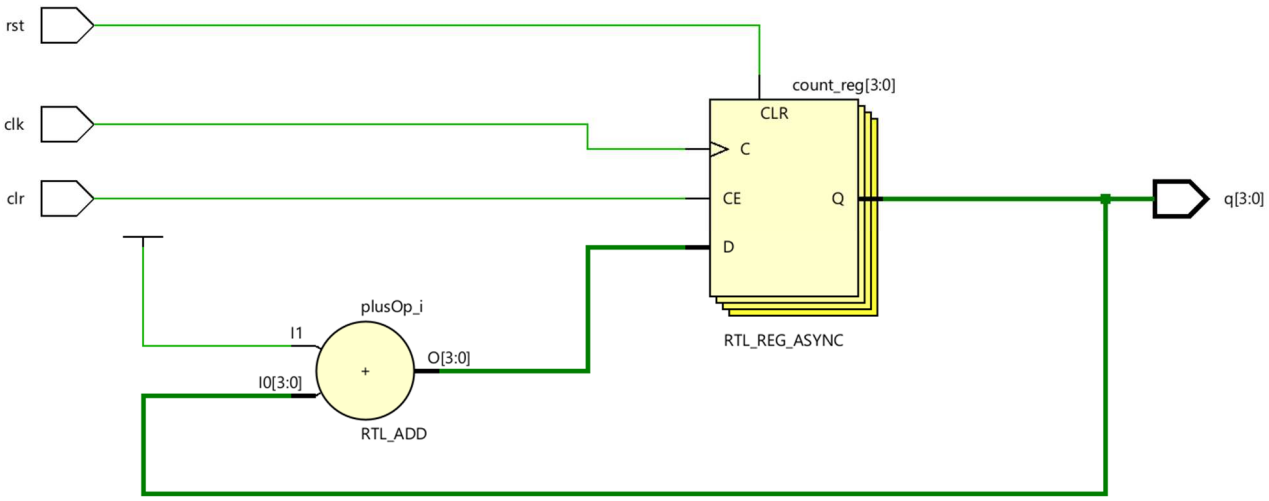
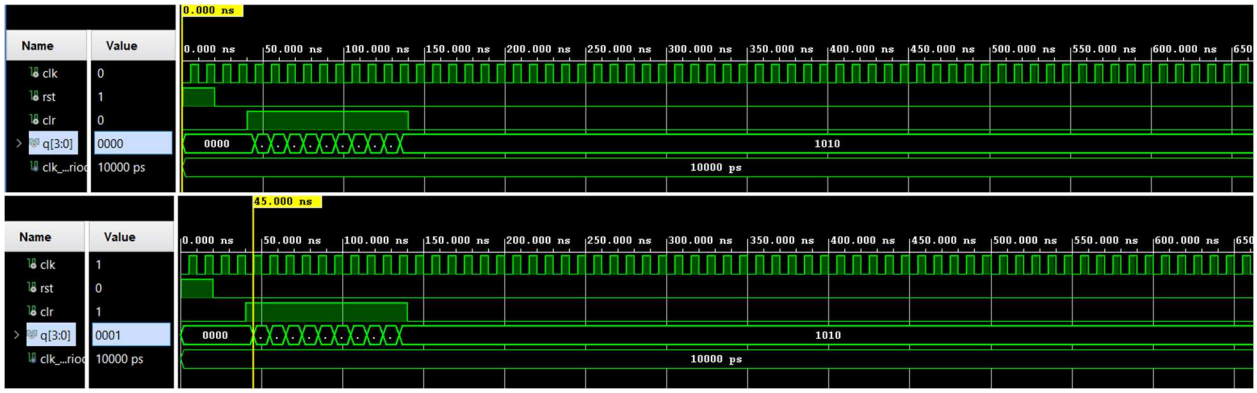


Figure 1: RTL Schematic of 4bit counter.

Simulation Model:

The simulation results are depicted in the figure below, showcasing the 4-bit counter output values ranging from 0 to 10. The output is represented by the 4-bit variable `q`, which displays the counter's value at each clock cycle.



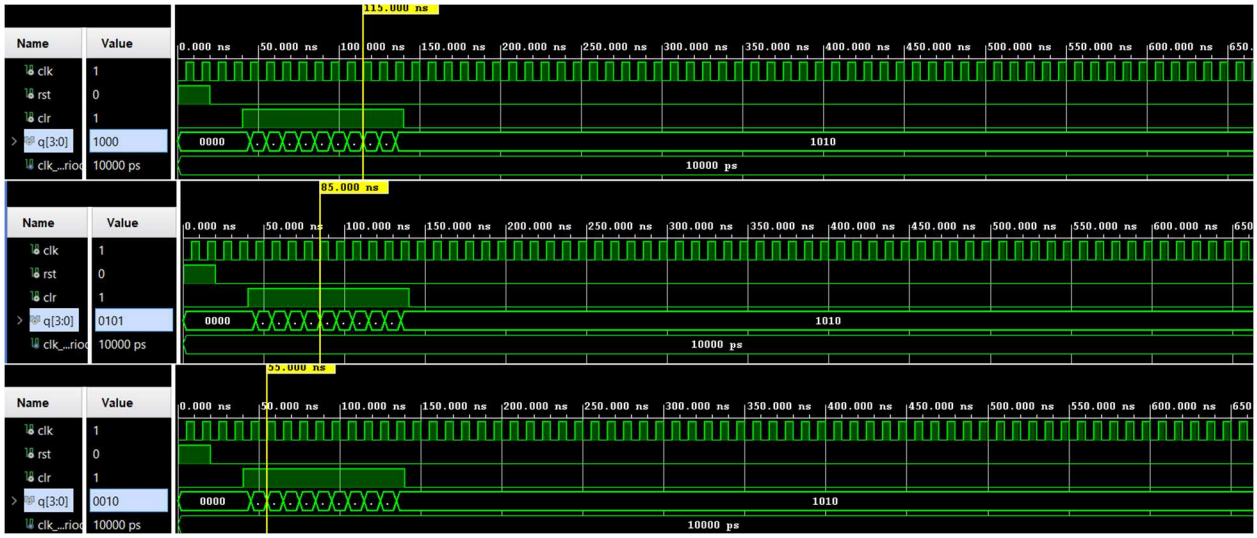


Figure 2: Simulation Model of 4bit counter

XDC File

set_property IOSTANDARD LVCMOS33 [get_ports {q[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {q[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {q[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {q[0]}]

Device Utilization Report:

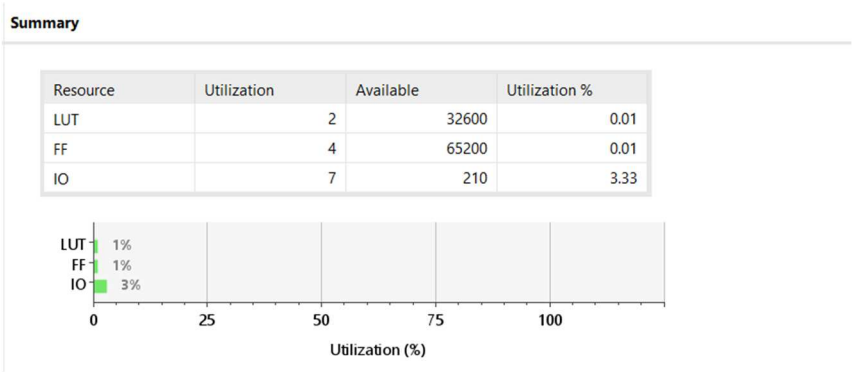


Figure 3: Device Utilization Report

Timing Analysis Report:

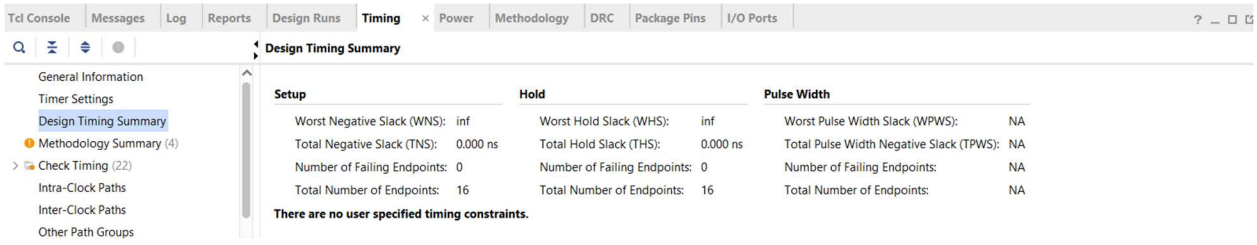


Figure 4: Timing Summary

2) 4 bit Up-down counter

VHDL Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity updowncounter_56 is
   Port (
      clk    : in std_logic;
      rst    : in std_logic;
      clr    : in std_logic;
      updown : in std_logic;
      q      : out std_logic_vector(3 downto 0));
end updowncounter_56;

architecture Behavioral of updowncounter_56 is
   signal count : std_logic_vector(3 downto 0) := "0000";
begin
   process (clk, rst)
   begin
      if (rst = '1') then
         count <= "0000";
      elsif rising_edge(clk) then
         if (clr = '1') then
            if (updown = '1') then
               count <= count + 1;
            else
               count <= count - 1;
            end if;
         end if;
      end if;
   end process;
   q <= count;
end Behavioral;
```

Test Bench Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity updowncounter_56_tb is
end updowncounter_56_tb;

architecture Behavioral of updowncounter_56_tb is
   component updowncounter_56
```

```
    Port (
        clk    : in std_logic;
        rst    : in std_logic;
        clr    : in std_logic;
        updown : in std_logic;
        q      : out std_logic_vector(3 downto 0)
    );
  end component;
  signal clk    : std_logic := '0';
  signal rst    : std_logic := '0';
  signal clr    : std_logic := '0';
  signal updown : std_logic := '0';
  signal q      : std_logic_vector(3 downto 0);
  constant clk_period : time := 10 ns;

begin
  uut: updowncounter_56
    port map (
        clk    => clk,
        rst    => rst,
        clr    => clr,
        updown => updown,
        q      => q  );
  clk_process : process
  begin
    while true loop
        clk <= '0';
        wait for clk_period / 2;
        clk <= '1';
        wait for clk_period / 2;
    end loop;
  end process;
  stimuli: process  begin
    rst <= '1';
    wait for 20 ns;
    rst <= '0';
    wait for 20 ns;

    clr <= '1';
    updown <= '1';
    wait for 100 ns;

    updown <= '0';
    wait for 100 ns;

    rst <= '1';
    wait for 20 ns;
    rst <= '0';
    wait for 20 ns;
```

```
        updown <= '1';
        wait for 50 ns;
        updown <= '0';
        wait for 50 ns;

        wait for 100 ns;
        wait;
    end process;
end Behavioral;
```

RTL Schematic

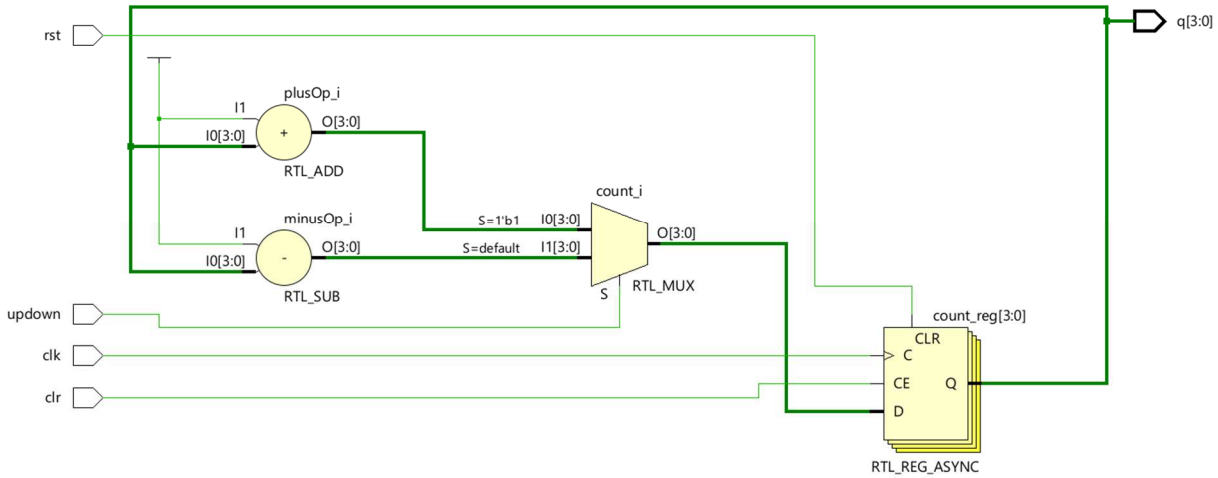


Figure 5: RTL Schematic of 4bit updown counter

XDC File

```
set_property IOSTANDARD LVCMOS33 [get_ports {q[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {q[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {q[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {q[0]}]
```

Simulation

The simulation for up-down counter is shown in the graph below. Clock Signal (**`clk`**)**:** was added to ensure the counter operates synchronously**.** Asynchronous Reset (**`rst`**)**:** resets the counter to 0. Clear Signal (**`clr`**)**:** is used to control whether the counter should increment or decrement based on the up-down signal. When up-down is '1', the counter increments. When up-down is '0', the counter decrements. The counter operates on the rising edge of the clock.


45.000 ns
Name
Value
clk
1
rst
0
clr
1
updown
1
q[3:0]
0001
clk_...rioc
10000 ps
0.000 ns
200.000 ns
400.000 ns
600.000 ns
800.000 ns
1,000.000 ns
1,200.000 ns
0000
10000 ps
55.000 ns
Name
Value
clk
1
rst
0
clr
1
updown
1
q[3:0]
0010
clk_...rioc
10000 ps
0.000 ns
200.000 ns
400.000 ns
600.000 ns
800.000 ns
1,000.000 ns
0000
10000 ps
135.000 ns
Name
Value
clk
1
rst
0
clr
1
updown
1
q[3:0]
1010
clk_...rioc
10000 ps
0.000 ns
200.000 ns
400.000 ns
600.000 ns
800.000 ns
1,000.000 ns
1,200.000 ns
0000
10000 ps
145.000 ns
Name
Value
clk
1
rst
0
clr
1
updown
0
q[3:0]
1001
clk_...rioc
10000 ps
0.000 ns
200.000 ns
400.000 ns
600.000 ns
800.000 ns
1,000.000 ns
1,200.000 ns
0000
10000 ps
165.000 ns
Name
Value
clk
1
rst
0
clr
1
updown
0
q[3:0]
0111
clk_...rioc
10000 ps
0.000 ns
200.000 ns
400.000 ns
600.000 ns
800.000 ns
1,000.000 ns
0000
10000 ps
Name
Value
clk
1
rst
0
clr
1
updown
0
q[3:0]
0100
clk_...rioc
10000 ps
0.000 ns
200.000 ns
400.000 ns
600.000 ns
800.000 ns
1,000.000 ns
0000
10000 ps

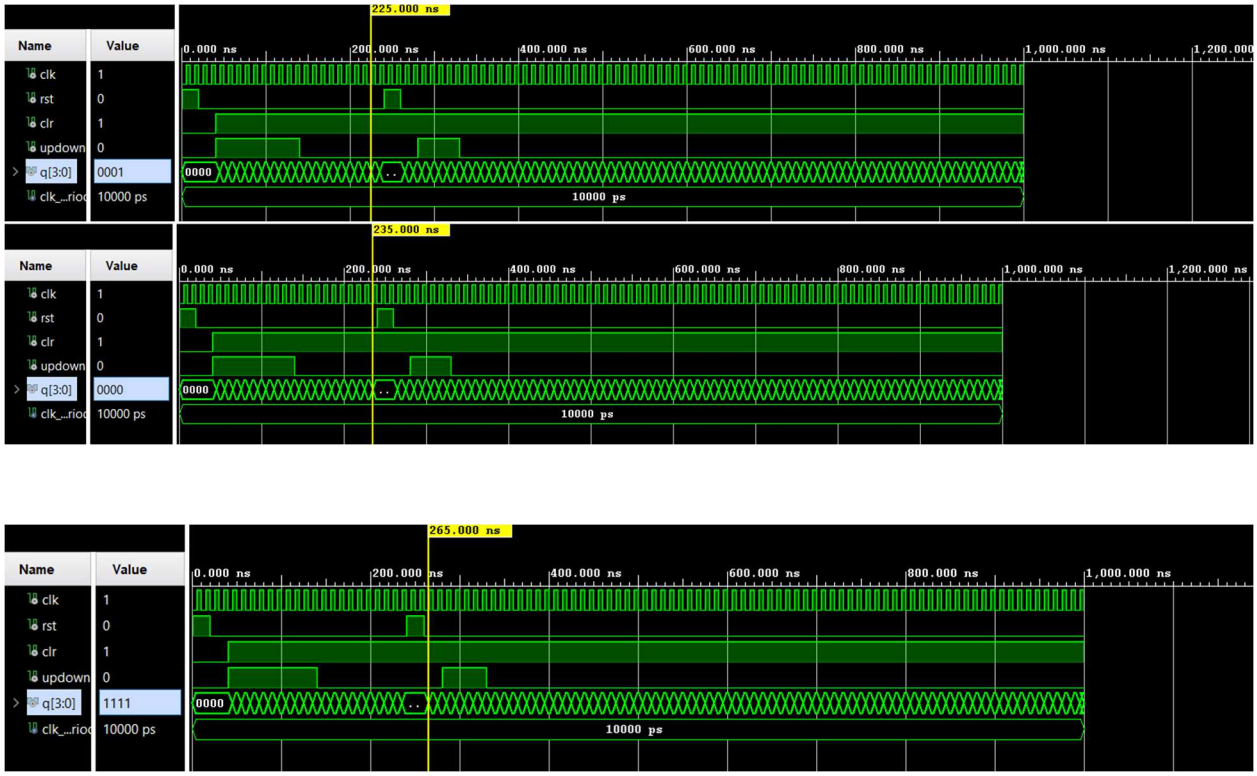


Figure 6: Timing Diagram for 4bit up-down counter

Device Utilization Report

Summary

Resource	Utilization	Available	Utilization %
LUT	2	32600	0.01
FF	4	65200	0.01
IO	8	210	3.81

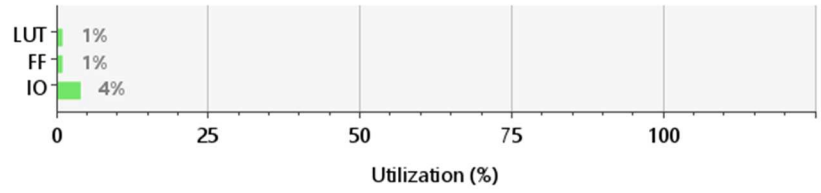


Figure 7: Device Utilization Report

Time Analysis Report

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	inf	Worst Hold Slack (WHS):	inf	Worst Pulse Width Slack (WPWS):	NA
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	NA
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	NA
Total Number of Endpoints:	16	Total Number of Endpoints:	16	Total Number of Endpoints:	NA

There are no user specified timing constraints.

Figure 8: Timing Summary for Full Adder

Conclusion

In summary, we developed and tested VHDL code for a 4-bit up/down counter using Vivado. The counter increments or decrements based on an `up-down` control signal, and it includes asynchronous reset functionality. We created the VHDL module for the up/down counter, ensuring proper synchronous operation with clock signal handling. A corresponding test-bench was designed to verify the counter's functionality, simulating various scenarios including resetting, incrementing, and decrementing. Additionally, troubleshooting steps were provided to resolve potential file access errors during simulation, such as ensuring no other processes are using the log file, manually deleting the file, and cleaning and rebuilding the project in Vivado. This comprehensive approach ensures that the counter operates correctly and that any issues encountered during simulation are addressed effectively.