S. B. JAIN INSTITUTE OF TECHNOLOGY,MANAGEMENT & RESEARCH,NAGPUR.

Practical No. 07

Aim: Design and Simulation of up-down counter and verify it using test bench.

Name of Student	**: Aadesh R. Motghare**
Roll No.	**: 41(ET20065)**
Semester/Year	**: 6th Sem/3rd Year**
Academic Session	**: 2022-23**
Date of Performance	**:**
Date of Submission	**:**

AIM: Design and Simulation of up-down counter and verify itusing test bench.

OBJECTIVE:

- To verify the functionality of up-down counter.
- To develop the logic for designing of digital System like traffic signals counter, watches using up-down counter.

SOFTWARE: - Xilinx ISE9.1.

THEORY:-

In digital logic and computing, a counter is a device which stores (and sometimes displays) the number of times a particular event or process has occurred, often in relationship to a clock signal.

In practice, there are two types of counters:

• Up counters, which increase (increment) in value

• Down counters, which decrease (decrement) in value

A counter that can change state in either direction, under the control of an up–down selector input, is known as an up–down counter. When the selector is in the up state, the counter increments its count and when the selector is in the down state, the counter decrements the count.

Both Synchronous and Asynchronous counters are capable of counting "Up" or counting "Down", but there is another more "Universal" type of counter that can count in both directions either Up or Down depending on the state of their input control pin and these are known as Bidirectional Counters. Bidirectional counters, also known as Up/Down counters, are capable of counting in either direction through any given count sequence and they can be reversed at any point within their count sequence by using an additional control input.

LOGICAL DIAGRAM:-

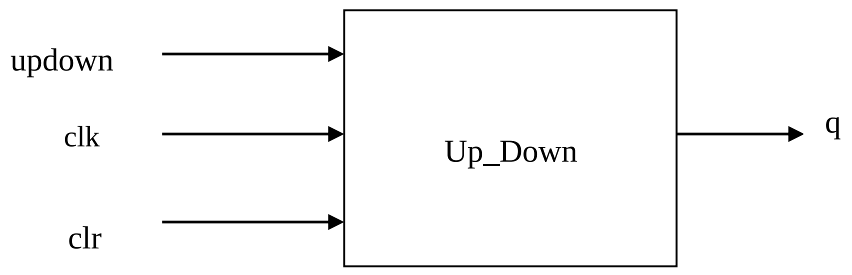


Fig 1. Logic diagram of up-down counter

TRUTH TABLE:-

Q3	**Q2**	**Q1**	**Q0**	**Up counter**				**Down counter**			
0	0	0	0	0	0	0	1	1	1	1	1
0	0	0	1	0	0	1	0	1	1	1	0
0	0	1	0	0	0	1	1	1	1	0	1
0	0	1	1	0	1	0	0	1	1	0	0
0	1	0	0	0	1	0	1	1	0	1	1
0	1	0	1	0	1	1	0	1	0	1	0
0	1	1	0	0	1	1	1	1	0	0	1
0	1	1	1	1	0	0	0	1	0	0	0
1	0	0	0	1	0	0	1	0	1	1	1
1	0	0	1	1	0	1	0	0	1	1	0
1	0	1	0	1	0	1	1	0	1	0	1
1	0	1	1	1	1	0	0	0	1	0	0
1	1	0	0	1	1	0	1	0	0	1	1
1	1	0	1	1	1	1	0	0	0	1	0
1	1	1	0	1	1	1	1	0	0	0	1
1	1	1	1	0	0	0	0	0	0	0	0

Table 1. Truth table of 4 bit up-down counter

VHDL CODE

STEPS FOR PROGRAM:-

Step 1.Library /Package Declaration : Involves declaration of all libraries and respective packages used in the design.

```
LIBRARY library_name;

USE library_name.package_name.all;
```

Step 2.Entity:

```
    ENTITY entity_name is
            PORT(signal_name(s): mode signal_type;
                  signal_name(s): mode signal_type;
                  …);
    end ENTITY entity_name;
```

Signals of the same mode and signal_type can be grouped on 1 line

MODE describes the direction data is transferred through port

- in – data flows into the port
- out – data flows out of port *only*
- buffer – data flows out of port *as well as read* internally.
- inout – bi-directional data flow into and out of port

SIGNAL_TYPE defines the data type for the signal(s)

- bit – single signals that can have logic values 0 and 1.
- bit_vector – bus signals(vector form of bit) that can have logic values 0 and 1.
- std_logic – part of std_logic_1164 package of IEEE library. Used to

represent 2 value logical values i.e 0 and 1 as well as other values such as high impedance ,don't care and others as described below.

- std_logic_vector – bus signals (vector form of std_logic) but IEEE standard for simulation and synthesis note that all vectors must have a range specified example for a 4 bit bus: bit_vector (3 downto 0) or std_logic_vector (3 downto 0).

In order to use std_logic and std_logic_vector we must include the library and package usage declarations in the VHDL model before the entity statement as follows:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
```

Values for std-logic:

U	un-initialized (undefined logic value)
X	forced unknown logic value
0	Logic low
1	Logic High
Z	high impedance (tri-state)
W	weak unknown
L	weak 0
H	weak 1
-	don't care value (for synthesis minimization)

Step 3: Architecture Declaration

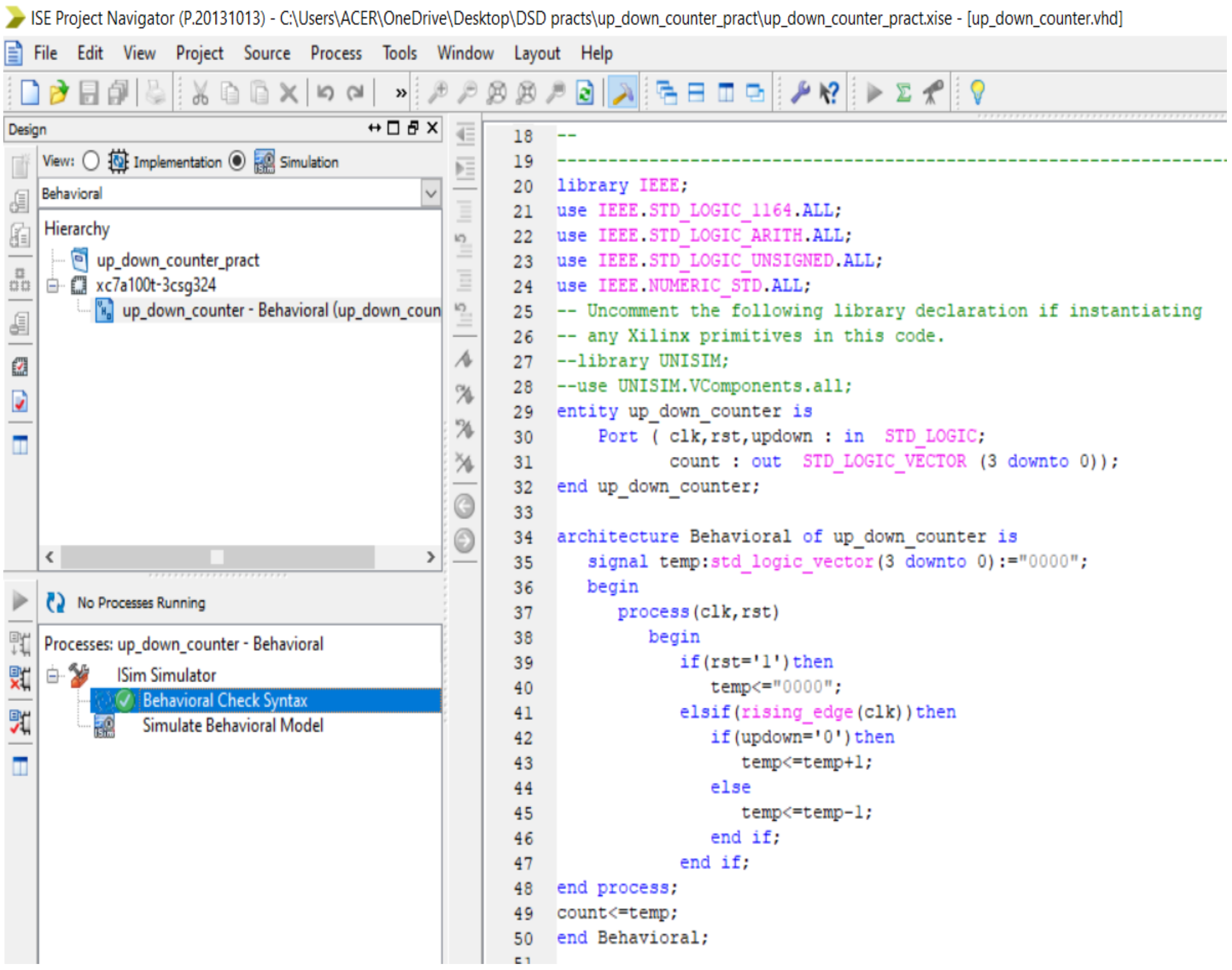
```
architecture architecture name of entity_name
  architecture_declarative_part;
begin
Statements;
 end architecture_name;
```

Here we should specify the entity name for which we are writing the architecture body. The architecture statements should be inside the begin and end keyword. Architecture declarative part may contain variables, constants, or component declaration.

Step 4: Simulate the VHDL code and remove the syntax errors if any.

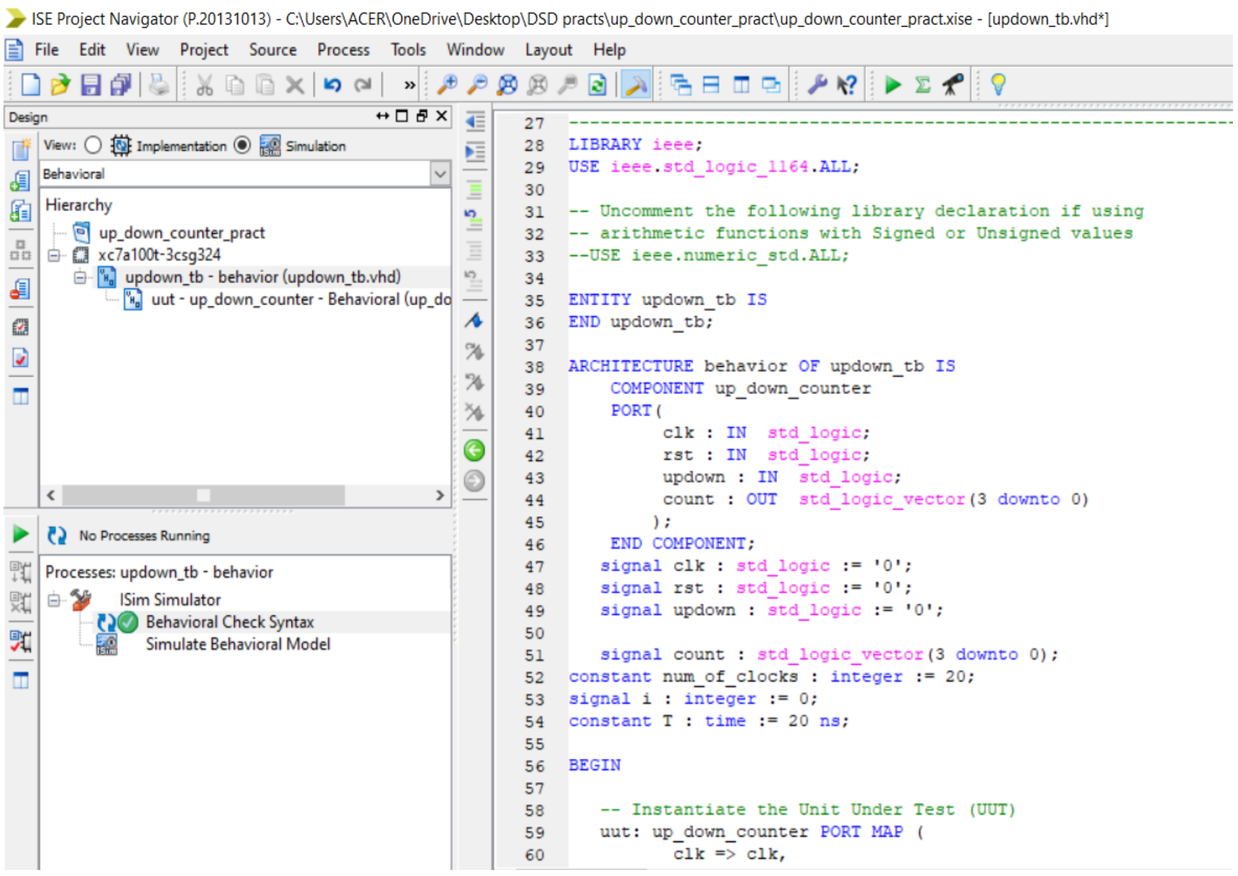
Step 5: Write the testbench and verify the design

CODE:

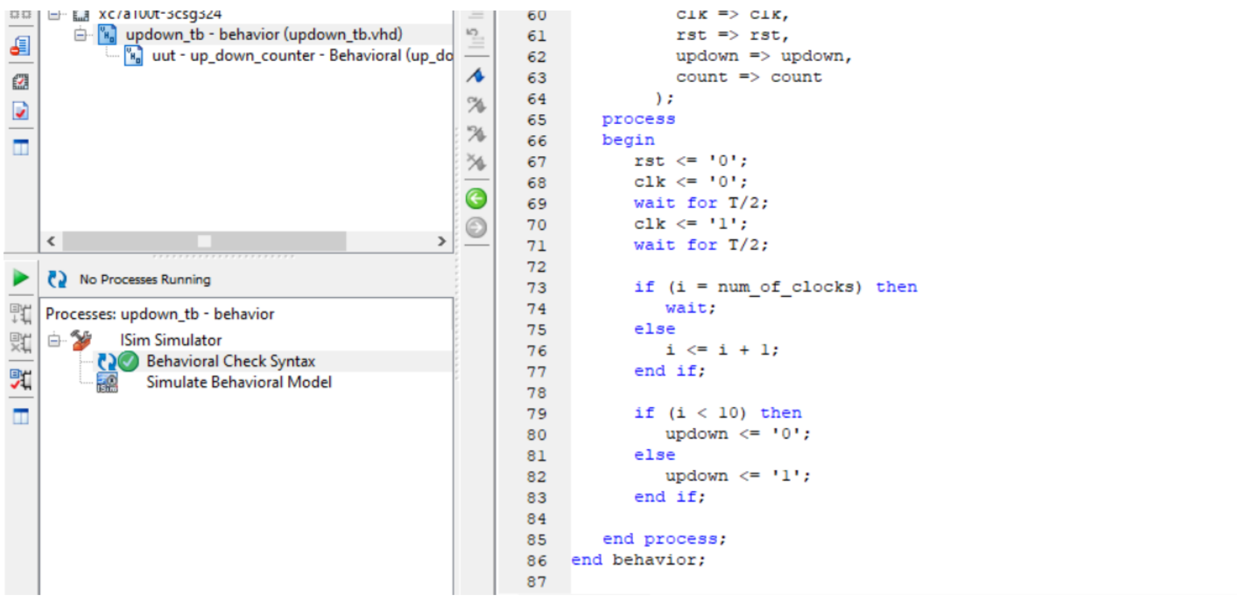

ISE Project Navigator (P.20131013) - C:\Users\ACER\OneDrive\Desktop\DSD practs\up_down_counter_pract\up_down_counter_pract.xise - [up_down_counter.vhd]
File
Edit
View
Project
Source
Process
Tools
Window
Layout
Help
Design
View:
Implementation
Simulation
Behavioral
Hierarchy
up_down_counter_pract
xc7a100t-3csg324
up_down_counter - Behavioral (up_down_coun
No Processes Running
Processes: up_down_counter - Behavioral
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model

```
--
-----------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;
-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
entity up_down_counter is
    Port ( clk,rst,updown : in  STD_LOGIC;
           count : out  STD_LOGIC_VECTOR (3 downto 0));
end up_down_counter;

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

TESTBENCH:

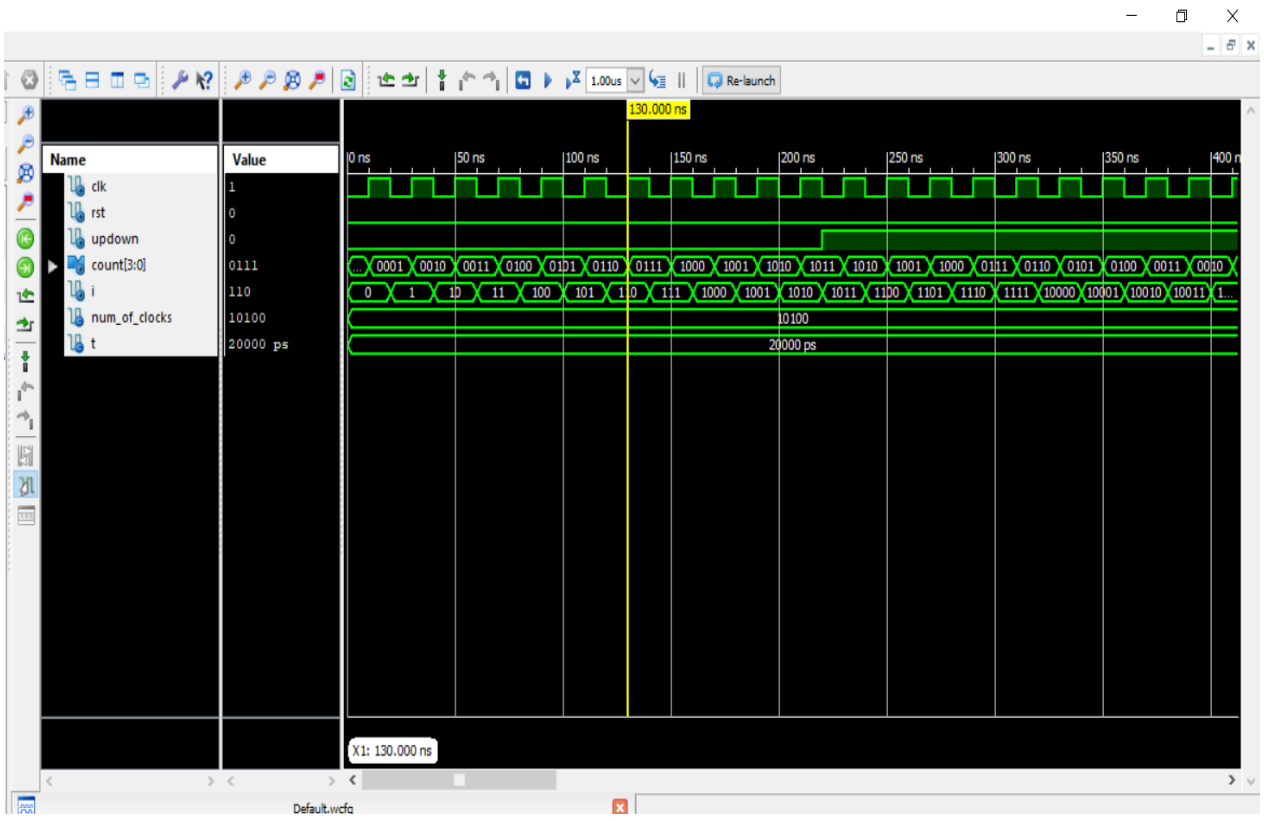

ISE Project Navigator (P.20131013) - C:\Users\ACER\OneDrive\Desktop\DSD practs\up_down_counter_pract\up_down_counter_pract.xise - [updown_tb.vhd*]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Behavioral
Hierarchy
up_down_counter_pract
xc7a100t-3csg324
updown_tb - behavior (updown_tb.vhd)
uut - up_down_counter - Behavioral (up_do
No Processes Running
Processes: updown_tb - behavior
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model

```
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

ENTITY updown_tb IS
END updown_tb;

ARCHITECTURE behavior OF updown_tb IS
    COMPONENT up_down_counter
    PORT(
         clk : IN  std_logic;
         rst : IN  std_logic;
         updown : IN  std_logic;
         count : OUT  std_logic_vector(3 downto 0)
        );
    END COMPONENT;
   signal clk : std_logic := '0';
   signal rst : std_logic := '0';
   signal updown : std_logic := '0';

   signal count : std_logic_vector(3 downto 0);
constant num_of_clocks : integer := 20;
signal i : integer := 0;
constant T : time := 20 ns;

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: up_down_counter PORT MAP (
          clk => clk,
```


xc7a100t-3csg324
updown_tb - behavior (updown_tb.vhd)
uut - up_down_counter - Behavioral (up_do
No Processes Running
Processes: updown_tb - behavior
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model

```
          rst => rst,
          updown => updown,
          count => count
        );
   process
   begin
      rst <= '0';
      clk <= '0';
      wait for T/2;
      clk <= '1';
      wait for T/2;

      if (i = num_of_clocks) then
         wait;
      else
         i <= i + 1;
      end if;

      if (i < 10) then
         updown <= '0';
      else
         updown <= '1';
      end if;

   end process;
end behavior;
```

WAVEFORM:-



RESULT:-

The VHDL code for up down counter is executed and desired output is obtained.

CONCLUSION:

Here, I successfully design and test the Up-Down Counter Using VHDL code.

DISCUSSION & VIVA VOCE

1. Explain the working of up down counter.
2. Explain the modeling style used to design up down counter.
3. What are the applications of up down counter?

REFERENCE:

VHDL Primer–J Bhasker –Pearson Education

NPTEL Video Lecture link- https://www.youtube.com/watch?v=PnwYW3RWARw.