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

Practical No. 05

Aim: Design and Implementation of BCD to 7-Segment decoder on Altera Terasic DE2 development board.

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

AIM: Design and Implementation of BCD to 7-Segment decoder on Altera Terasic DE2 development board.

OBJECTIVE:

- To verify the functionality of BCD to 7-Segment converter.
- To model the digital System like LCD display.

SOFTWARE: - Quartus-II.

THEORY:-

Here is a program for BCD to 7-segment display decoder. The module takes 4 bitBCD as input and outputs 7 bit decoded output for driving the display unit. A seven segment display can be used to display decimal digits. They have LED or LCD elements which becomes active when the input is zero. The figure shows how different digits are displayed:

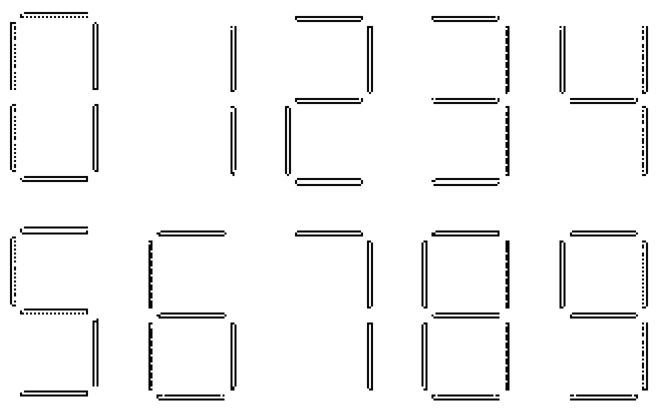


Figure. 1. Convention for Displaying Decimal Digits

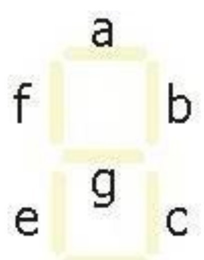


Figure 2. Seven-Segment Numerical Display

If you want a decimal number to be displayed using this code then convert the corresponding code into BCD and then instantiate this module for each digit of the BCD code.

TRUTH TABLE:

BCD				7-SEGMENT DISPLAY						
B3	**B2**	**B1**	**B0**	**A**	**B**	**C**	**D**	**E**	**F**	**G**
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	1	1	1	1	1	1	0
0	0	1	1	1	1	0	1	1	0	1
0	0	1	0	1	1	1	1	0	0	1
0	1	1	0	0	0	1	0	0	1	1
0	1	1	1	1	0	1	1	0	1	1
0	1	0	1	0	0	1	1	1	1	1
1	1	0	0	1	1	1	0	0	0	0
1	1	0	0	1	1	1	1	1	1	1
1	1	0	1	1	1	1	0	0	1	1

Table 1. Truth table of BCD to 7-Segment decoder(Common Cathode)

VHDL CODE:-

<u>STEPS FOR PROGRAM:-</u>

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 **l**ine.

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

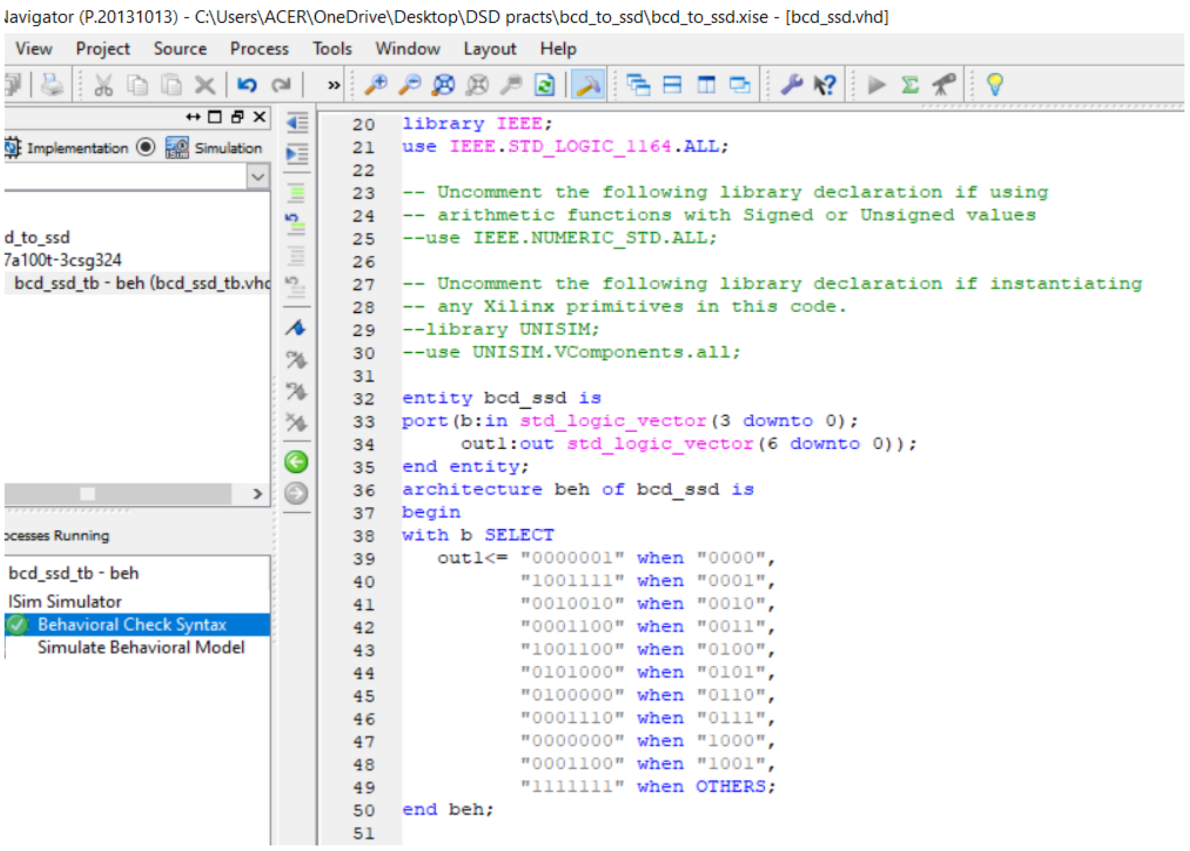
```
architecture architecture name of entity_name
  architecture_declarative_part;
begin
Statements;
 end architecture_name;
```

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.

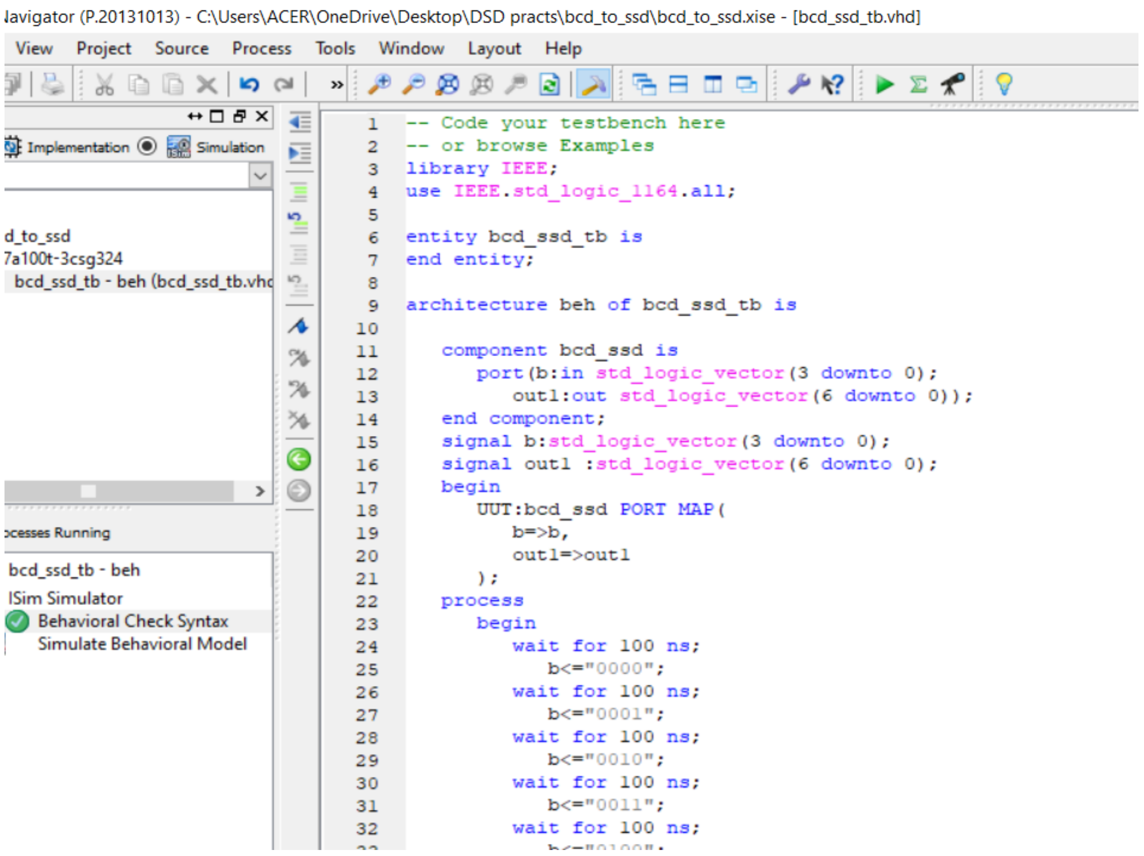
Step 5: Write the testbench and verify the design .

CODE:

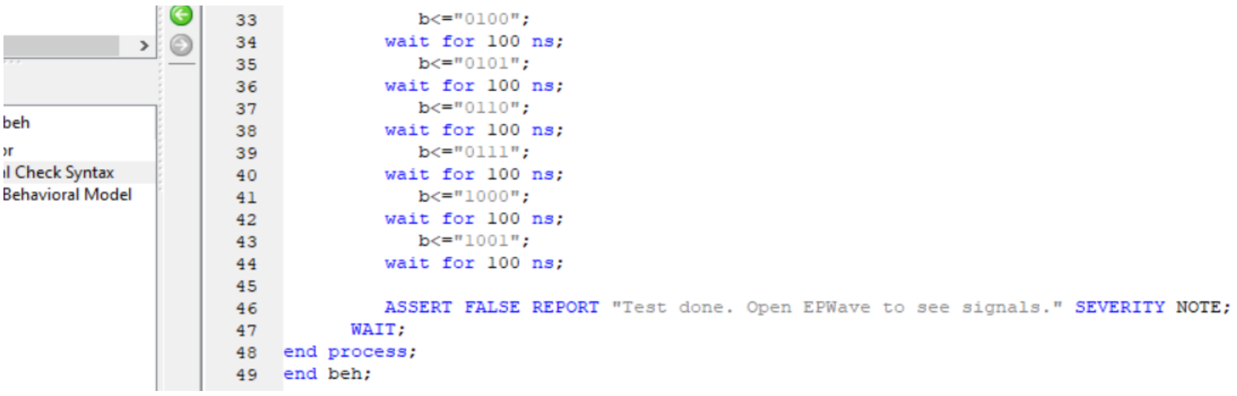

lavigator (P.20131013) - C:\Users\ACER\OneDrive\Desktop\DSD practs\bcd_to_ssd\bcd_to_ssd.xise - [bcd_ssd.vhd]
View
Project
Source
Process
Tools
Window
Layout
Help
Implementation
Simulation
d_to_ssd
7a100t-3csg324
bcd_ssd_tb - beh (bcd_ssd_tb.vhd
ocesses Running
bcd_ssd_tb - beh
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model

```vhdl
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;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity bcd_ssd is
port(b:in std_logic_vector(3 downto 0);
     out1:out std_logic_vector(6 downto 0));
end entity;
architecture beh of bcd_ssd is
begin
with b SELECT
   out1<= "0000001" when "0000",
          "1001111" when "0001",
          "0010010" when "0010",
          "0001100" when "0011",
          "1001100" when "0100",
          "0101000" when "0101",
          "0100000" when "0110",
          "0001110" when "0111",
          "0000000" when "1000",
          "0001100" when "1001",
          "1111111" when OTHERS;
end beh;
```

TESTBENCH:

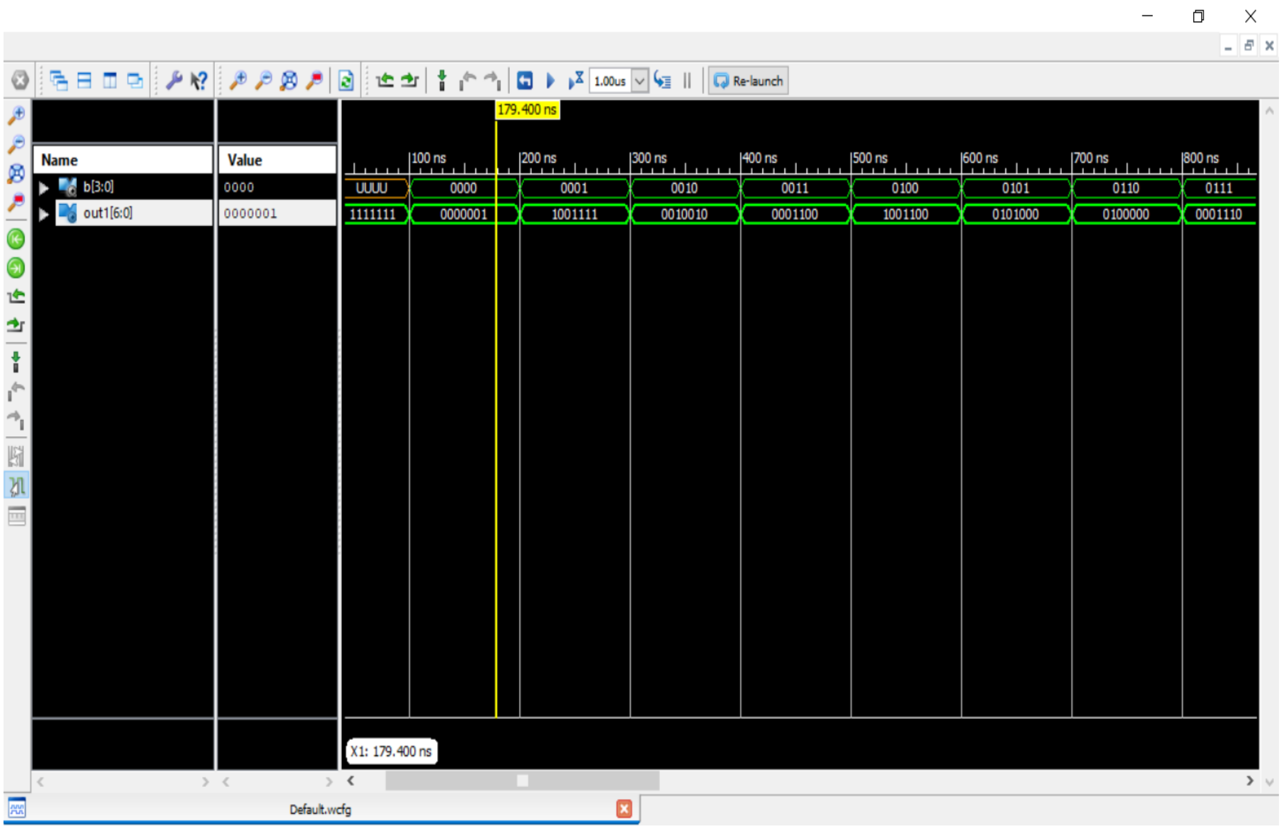

lavigator (P.20131013) - C:\Users\ACER\OneDrive\Desktop\DSD practs\bcd_to_ssd\bcd_to_ssd.xise - [bcd_ssd_tb.vhd]
View
Project
Source
Process
Tools
Window
Layout
Help
Implementation
Simulation
d_to_ssd
7a100t-3csg324
bcd_ssd_tb - beh (bcd_ssd_tb.vhd
ocesses Running
bcd_ssd_tb - beh
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model

```vhdl
-- Code your testbench here
-- or browse Examples
library IEEE;
use IEEE.std_logic_1164.all;

entity bcd_ssd_tb is
end entity;

architecture beh of bcd_ssd_tb is

   component bcd_ssd is
      port(b:in std_logic_vector(3 downto 0);
         out1:out std_logic_vector(6 downto 0));
   end component;
   signal b:std_logic_vector(3 downto 0);
   signal out1 :std_logic_vector(6 downto 0);
   begin
      UUT:bcd_ssd PORT MAP(
         b=>b,
         out1=>out1
      );
   process
      begin
         wait for 100 ns;
            b<="0000";
         wait for 100 ns;
            b<="0001";
         wait for 100 ns;
            b<="0010";
         wait for 100 ns;
            b<="0011";
         wait for 100 ns;
```



```
            b<="0100";
        wait for 100 ns;
            b<="0101";
        wait for 100 ns;
            b<="0110";
        wait for 100 ns;
            b<="0111";
        wait for 100 ns;
            b<="1000";
        wait for 100 ns;
            b<="1001";
        wait for 100 ns;

        ASSERT FALSE REPORT "Test done. Open EPWave to see signals." SEVERITY NOTE;
    WAIT;
end process;
end beh;
```

WAVEFORM:



```
Name        Value
b[3:0]      0000
out1[6:0]   0000001

b[3:0]     UUUU     0000     0001     0010     0011     0100     0101     0110     0111
out1[6:0]  1111111  0000001  1001111  0010010  0001100  1001100  0101000  0100000  0001110
```

RESULT:-

The VHDL code for BCD to 7 segment decoder is executed and desired output is obtained.

CONCLUSION:

Here, I successfully design the BCD to 7 segment decoder and also give the stimulus by writing test bench for it.

DISCUSSION & VIVA VOCE

1) Explain the working of BCD to 7-Segment converter.
2) How to design BCD to 7-Segment converter in VHDL
3) Explain the modeling style used to design BCD to 7-Segment converter.
4) What are the applications of BCD to 7-Segment converter.

REFERENCE:

- VHDL Primer–J Bhasker –Pearson Education
- NPTEL Video Lecture link https://www.youtube.com/watch?v=vNSIdKqSjO4.