**Mini Project**

### 3 : 8 Decoder Using Basic Logic Gate

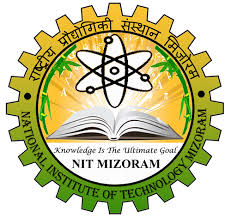
**Submitted by**

Amanjeet Kumar ( BT18EE011 )

Harshit Singh Tomar ( BT18EC009 )

Pawan Kumar Chaudhary ( BT18EE018 )

Monu Sharma ( BT18EE022 )



**Department of Electronics And Communication Engineering**

National institute of Technology , Mizoram

December 2020

Software Requirements :

To do this mini project, we must install the following software: Xilinx ISE 10.1

Hardware Requirements :

To do this mini project, we must have the following hardware: •Spartan-3 Start up Kit, containing the Spartan-3 Start up Kit Demo Board.

Starting the ISE Software :

To start ISE, double-click the desktop icon,



or start ISE from the Start menu by selecting:

Start → All Programs → Xilinx ISE 10.1 → Project Navigator

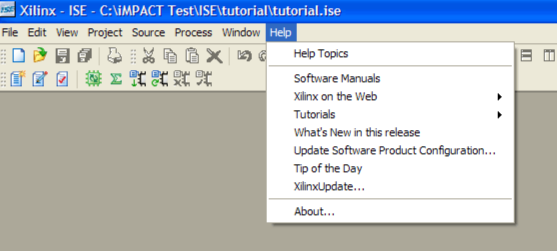
Note: Your start-up path is set during the installation process and may differ from the one above

Accessing Help :

At any time during the tutorial, you can access online help for additional information about the ISE software and related tools.

To open Help, do either of the following:

* + Press F1 to view Help for the specific tool or function that you have selected or highlighted.
  + Launch the ISE Help Contents from the Help menu. It contains information about creating and maintaining your complete design flow in ISE.

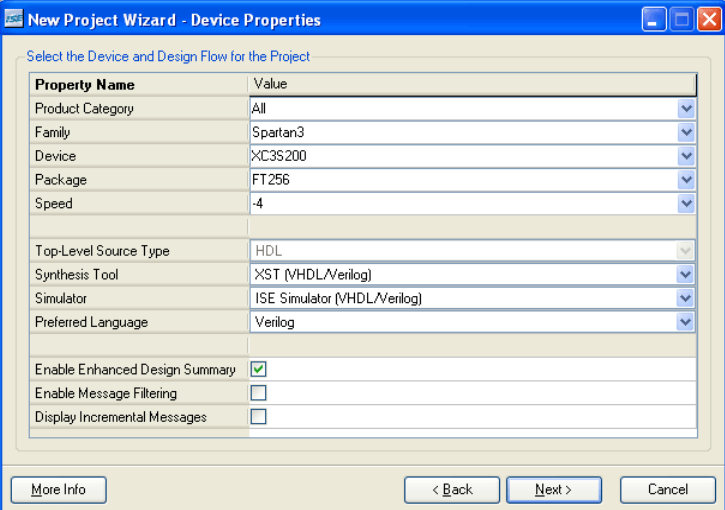


Create a New Project :

To create a new project:

1. Select File>New Project... The New Project Wizard appears.
2. Type tutorial in the Project Name field.
3. Enter or browse to a location (directory path) for the new project. A tutorial subdirectory is created automatically.
4. Verify that HDL is selected from the Top-Level Source Type list.
5. Click Next to move to the device properties page.
6. Fill in the properties in the table as shown below:

* Product Category: All
* Family: Spartan3
* Device: XC3S200
* Package: FT256
* Speed Grade: -4
* To p - L e v e l S o u rc e Ty p e : HDL
* Synthesis Tool: XST (VHDL/Verilog)
* Simulator: ISE Simulator (VHDL/Verilog)
* Preferred Language: Verilog (or VHDL)
* Verify that Enable Enhanced Design Summary is selected. Leave the default values in the remaining fields. When the table is complete, your project properties will look like the following:



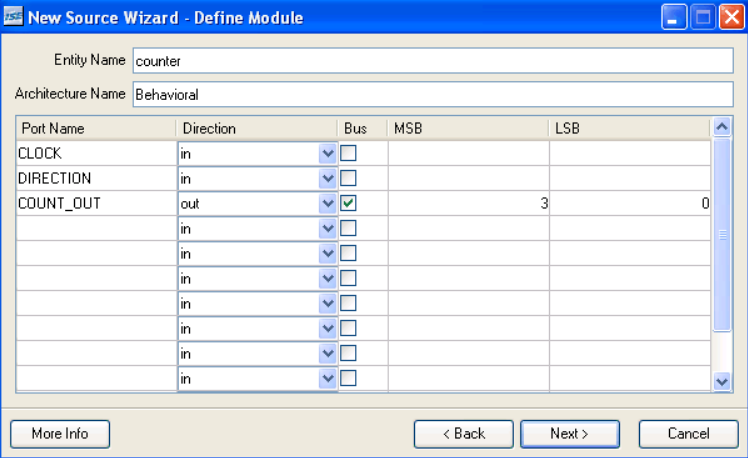
7.Click Next to proceed to the Create New Source window in the New Project Wizard. At the end of the next section, your new project will be complete.

Create an HDL Source :

In this section, you will create the top-level HDL file for your design. Determine the language that you wish to use for the tutorial. Then, continue either to the “Creating a VHDL Source” section below, or skip to the “Creating a Verilog Source” section.

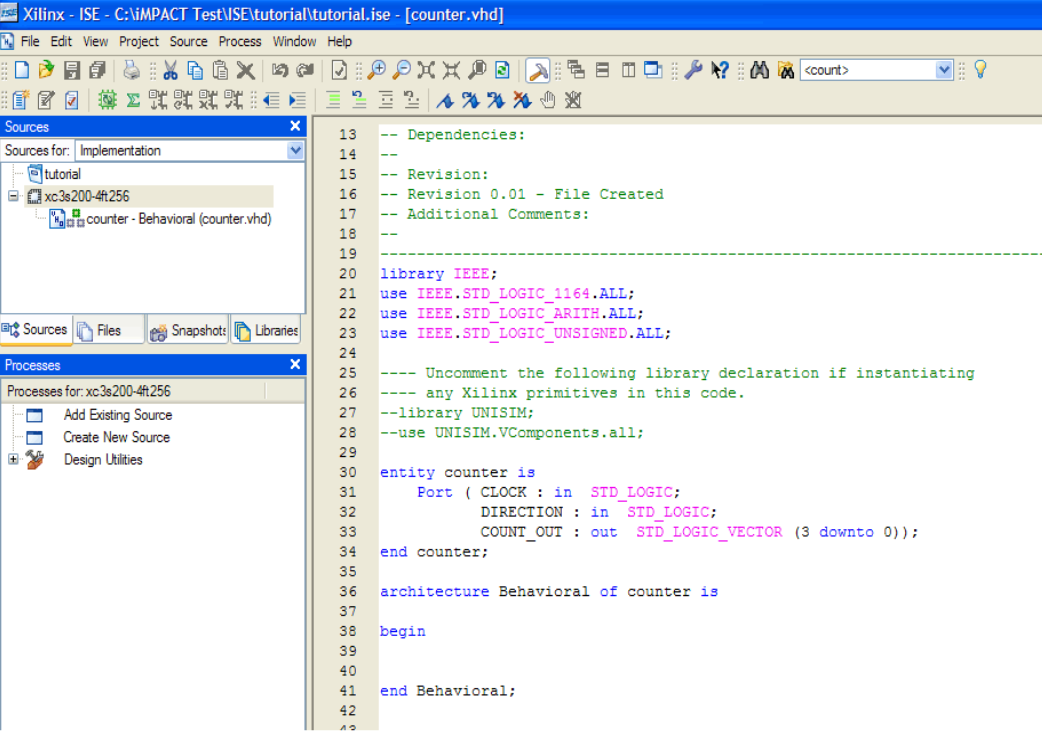
Creating a VHDL Source :

* + 1. Create a VHDL source file for the project as follows:
    2. 1.Click the New Source button in the New Project Wizard.
    3. 2.Select VHDL Modules the source type.
    4. 3.Type in the file name counter.
    5. 4.Verify that the Add to project checkbox is selected.
    6. 5.Click Next.
    7. 6.Declare the ports for the counter design by filling in the port information as shown below:



* + - 1. Click Next, then Finishin the New Source Wizard - Summary dialog box to complete the new source file template.
      2. Click Next, then Next, then Finish.

The source file containing the entity/architecture pair displays in the Workspace, and the counter displays in the Source tab, as shown below:



Using Language Templates (VHDL) :

The next step in creating the new source is to add the behavioural description for the counter. To do this you will use a simple counter code example from the ISE Language Templates and customize it for the counter design

1. Place the cursor just below the begin statement within the counter architecture.
2. Open the Language Templates by selecting Edit→ Language Templates...

Note: You can tile the Language Templates and the counter file by selecting Window→ Tile Vertically to make them both visible.

1. Using the “+” symbol, browse to the following code example: VHDL → Synthesis Constructs → Coding Examples → Counters → Binary →Up/Down Counters → Simple Counter
2. With Simple Counter selected, select Edit→ Use in File, or select the Use Template in File toolbar button. This step copies the template into the counter source file.

Close the Language Templates.

Final Editing of the VHDL Source :

1. 1.Add the following signal declaration to handle the feedback of the counter output below the architecture declaration and above the first begin statement:signal count\_int : std\_logic\_vector(3 downto 0) := "0000";
2. 2.Customize the source file for the counter design by replacing the port and signal name placeholders with the actual ones as follows:

* replace all occurrences of <clock> with CLOCK
* replace all occurrences of <count\_direction> with DIRECTION
* replace all occurrences of <count> with count\_int

1. 3.Add the following line below the end process;
2. Statement :COUNT\_OUT <= count\_int;
3. 4.Save the file by selecting File → Save.

When you are finished, the counter source file will look like the following:

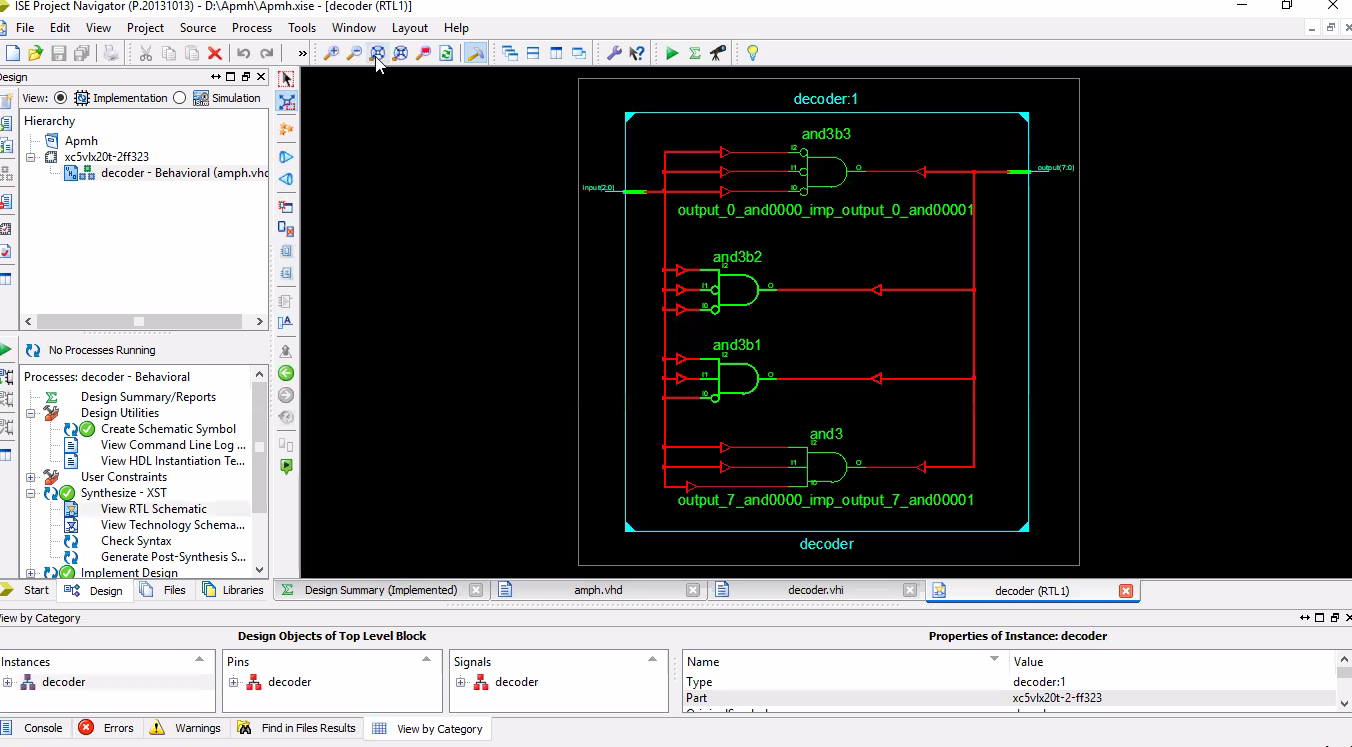
Note : You have now created the VHDL source for the tutorial project.

3 : 8 Decoder using basic logic gates :

Here is the code for 3 : 8 **Decoder** using basic logic gates such as AND,NOT,OR etc .The module has one 3-bit input which is decoded as a 8-bit output.

library IEEE;  
use IEEE.STD\_LOGIC\_1164.ALL;  
  
--entity declaration with port definitions  
entity decoder is  
port(    input :        in std\_logic\_vector(2 downto 0);  --3 bit input  
            output : out std\_logic\_vector(7 downto 0)  -- 8 bit ouput);  
end decoder;  
--architecture of entity  
architecture Behavioural of decoder is  
  
begin   
output(0) <= (not input(2)) and (not input(1)) and (not input(0));  
output(1) <= (not input(2)) and (not input(1)) and input(0);  
output(2) <= (not input(2)) and input(1) and (not input(0));  
output(3) <= (not input(2)) and input(1) and input(0);  
output(4) <= input(2) and (not input(1)) and (not input(0));  
output(5) <= input(2) and (not input(1)) and input(0);

output(6) <= input(2) and input(1) and (not input(0));  
output(7) <= input(2) and input(1) and input(0);  
  
end Behavioural ;



Checking the Syntax of the New Counter Module :

When the source files are complete, check the syntax of the design to find errors and t typos.

1. 1.Verify that Implementation is selected from the drop-down list in the Sources window.
2. 2.Select the counter design source in the Sources window to display the related processes in the Processes window.
3. 3.Click the “+” next to the Synthesize-XST process to expand the process group.
4. 4.Double-click the Check Syntax process.

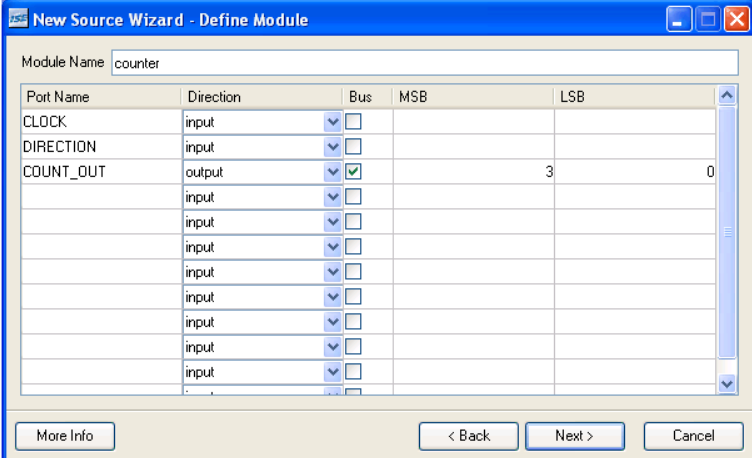
Note: You must correct any errors found in your source files. You can check for errors in the Console tab of the Transcript window. If you continue without valid syntax, you will not be able to simulate or synthesize your design.

1. 5.Close the HDL file.

Creating a Verilog Source :

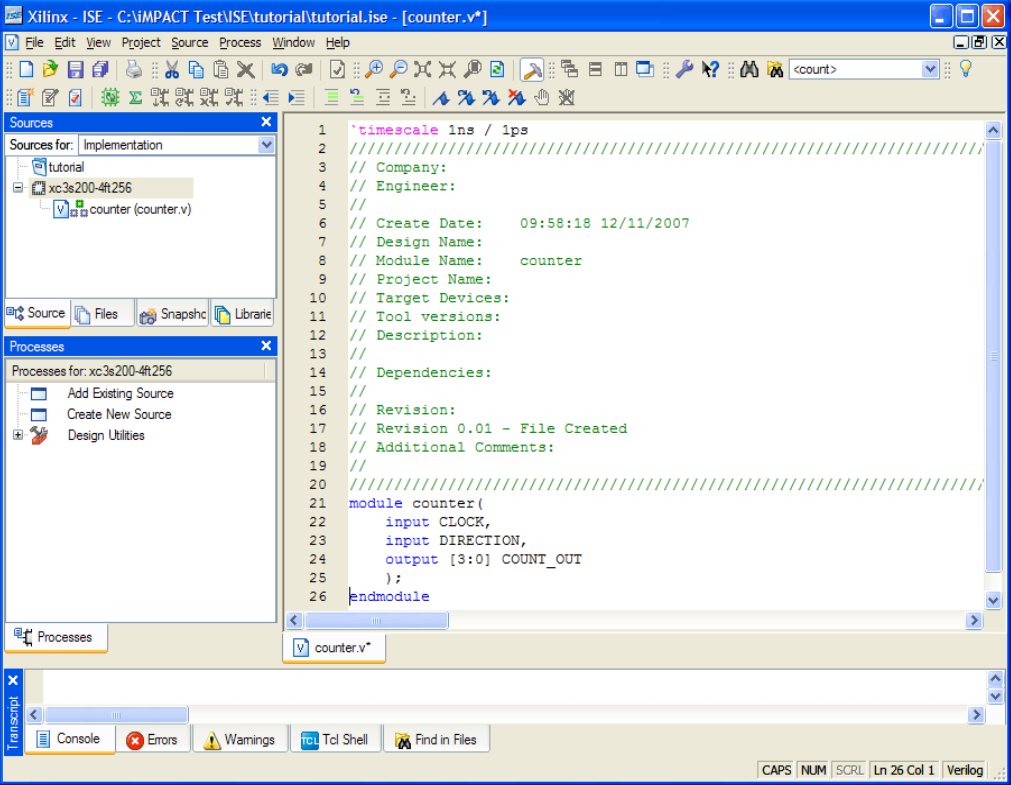
Create the top-level Verilog source file for the project as follows:

1. 1.Click New Source in the New Project dialog box.
2. 2.Select Verilog Modules the source type in the New Source dialog box.
3. 3.Type in the file name counter.
4. 4.Verify that the Add to Project checkbox is selected.
5. 5.Click Next.
6. 6.Declare the ports for the counter design by filling in the port information as shown below:



1. 7.Click Next, then Finish in the New Source Information dialog box to complete the new source file template.
2. 8.Click Next, then Next, then Finish.

The source file containing the counter module displays in the Workspace, and the counter displays in the Sources tab, as shown below:



Using Language Templates (Verilog) :

The next step in creating the new source is to add the behavioural description for counter. Use a simple counter code example from the ISE Language Templates and customize it for the counter design.

1. 1.Place the cursor on the line below the output [3:0] COUNT\_OUT; statement.
2. 2.Open the Language Templates by selecting

Edit→ Language Templates...

Note: You can tile the Language Templates and the counter file by selecting Window→Tile Vertically to make them both visible.

3. Using the “+” symbol, browse to the following code example:

Verilog → Synthesis Constructs → Coding Examples → Counters → Binary →Up/Down Counters → Simple Counter

4. With Simple Counter selected, select Edit →Use in File, or select the Use Template in File toolbar button. This step copies the template into the counter source file.

5. Close the Language Templates.

Design Simulation :

Verifying Functionality using Behavioural Simulation :

Create a test bench waveform containing input stimulus you can use to verify the functionality of the counter module. The test bench waveform is a graphical view of a test bench.

Create the test bench waveform as follows:

1. Select the counter HDL file in the Sources window.
2. Create a new test bench source by selecting Project → New Source.
3. In the New Source Wizard, select Test Bench Waveform as the source type, and type counter\_tbw in the File Name field.
4. Click Next.
5. The Associated Source page shows that you are associating the test bench waveform with the source file counter. Click Next.
6. The Summary page shows that the source will be added to the project, and it displays the source directory, type, and name. Click Finish.
7. You need to set the clock frequency, setup time and output delay times in the Initialize Timing dialog box before the test bench waveform editing window opens. The requirements for this design are the following

♦ The counter must operate correctly with an input clock frequency = 25 MHz .

♦ The DIRECTION input will be valid 10 ns before the rising edge of CLOCK.

♦ The output (COUNT\_OUT) must be valid 10 ns after the rising edge of CLOCK.

The design requirements correspond with the values below.

Fill in the fields in the Initialize Timing dialog box with the following information:

♦ Clock High Time: 20 ns.

♦ Clock Low Time: 20 ns.

♦ Input Setup Time: 10 ns.

♦ Output Valid Delay: 10 ns.

♦ Offset: 0 ns.

♦ Global Signals: GSR(FPGA)

Note: When GSR(FPGA) is enabled, 100 ns. is added to the Offset value automatically.

♦ Initial Length of Test Bench: 1500 ns.

The test bench program used for testing the design is given below:

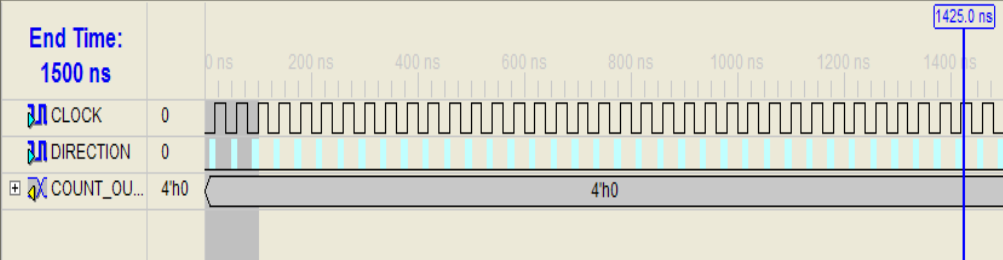
library IEEE;  
use IEEE.STD\_LOGIC\_1164.ALL;  
--this is how entity for your test bench code has to be declared.  
entity testbench is  
end testbench;  
  
architecture behaviour of testbench is  
--signal declarations.  
signal input : std\_logic\_vector(2 downto 0) :=(others => '0');  
signal output :  std\_logic\_vector(7 downto 0) :=(others => '0');  
  
begin  
--entity instantiation  
UUT : entity work.decoder port map(input,output);  
  
--definition of simulation process  
tb : process   
begin  
input<="000";  --input = 0.  
wait for 2 ns;  
input<="001";   --input = 1.  
wait for 2 ns;  
input<="010";   --input = 2.  
wait for 2 ns;  
input<="011";   --input = 3.  
wait for 2 ns;  
input<="100";   --input = 4.  
wait for 2 ns;  
input<="101";   --input = 5.  
wait for 2 ns;  
input<="110";   --input = 6.  
wait for 2 ns;  
input<="111";   --input = 7.  
wait;  
end process tb;  
  
end;

8.Click Finish to complete the timing initialization.

9.The blue shaded areas that precede the rising edge of the CLOCK correspond to the Input Setup Time in the Initialize Timing dialog box. Toggle the DIRECTION port to define the input stimulus for the counter design as follows:

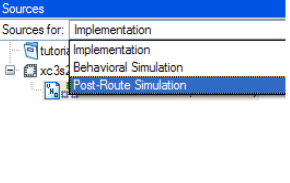
* Click on the blue cell at approximately the 300 ns to assert DIRECTION high so that the counter will count up.
* Click on the blue cell at approximately the 900 ns to assert DIRECTION low so that the counter will count down.

Note: For more accurate alignment, you can use the Zoom In and Zoom Out toolbar buttons



10. Save the waveform.

11. In the Sources window, select the Behavioral Simulation view to see that the test bench waveform file is automatically added to your project.



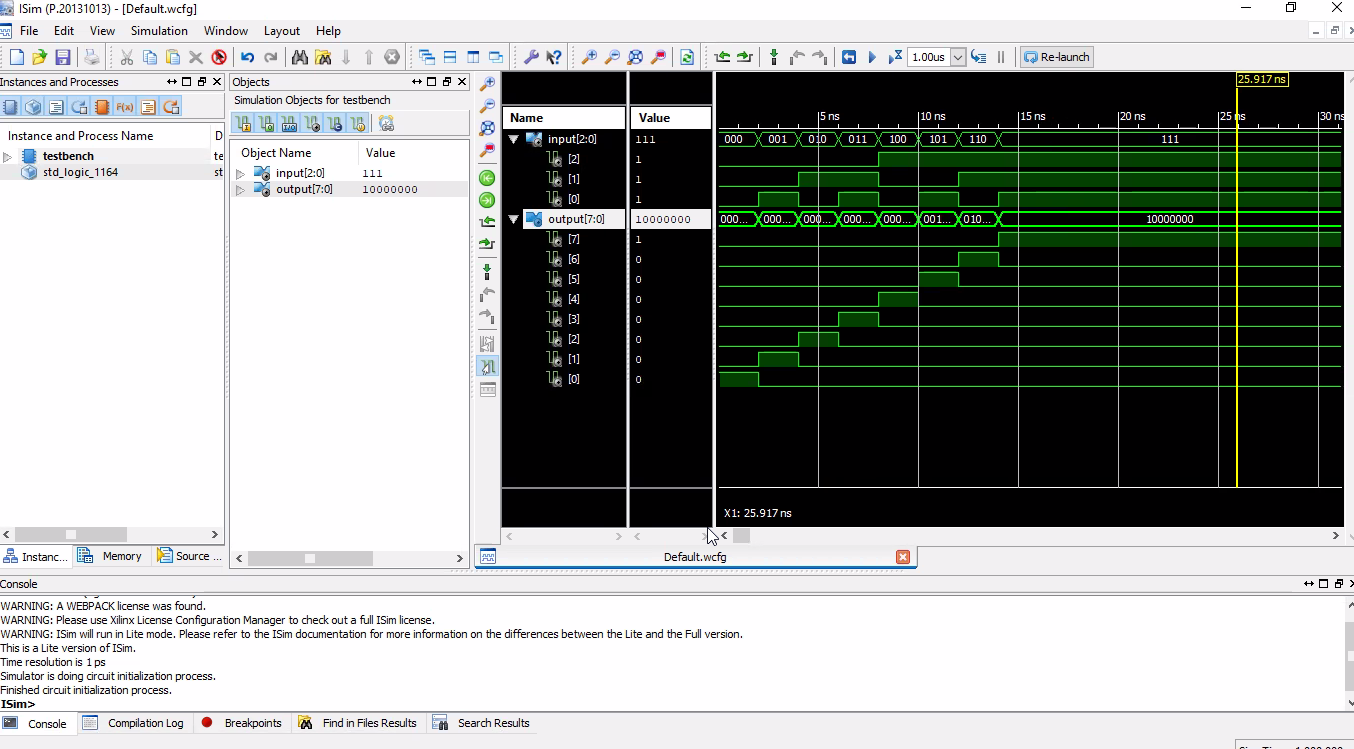
12. Close the test bench waveform.

Simulating Design Functionality :

Verify that the counter design functions as you expect by performing behavior simulation as follows:

1. Verify that Behavioral Simulation and counter\_tbw are selected in the Sources window.
2. In the Processes tab, click the “+” to expand the Xilinx ISE Simulator process and double-click the Simulate Behavioral Model process.The ISE Simulator opens and runs the simulation to the end of the test bench.
3. To view your simulation results, select the Simulation tab and zoom in on the transitions.

The simulation waveform results will look like the following:



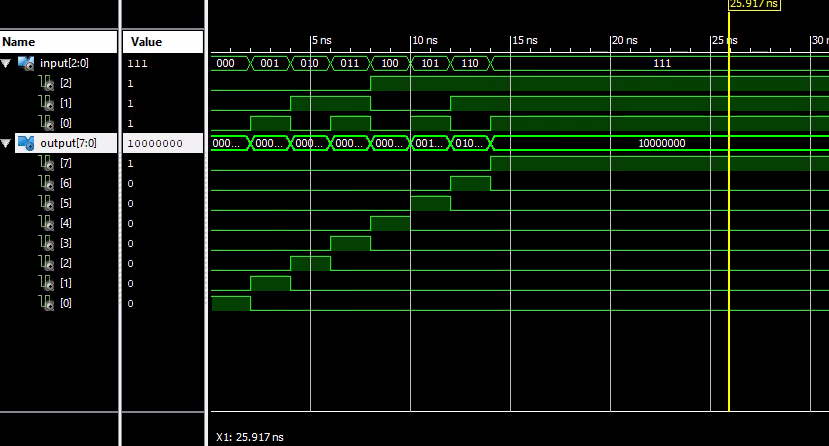
Note:You can ignore any rows that start with TX.

4.Verify that the counter is counting up and down as expected.

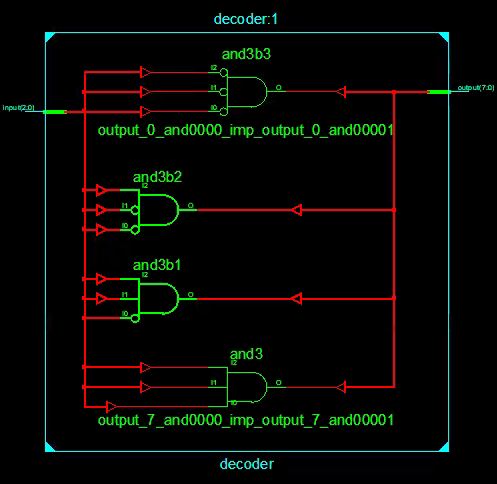
5.Close the simulation view. If you are prompted with the following message, “You have an active simulation open. Are you sure you want to close it?“, click Ye s to continue

You have now completed simulation of your design using the ISE Simulator.

Final result : The simulated waveform is shown below:

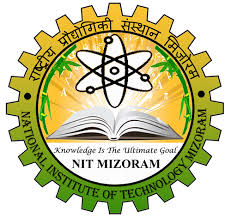


The code was synthesized using **XILINX ISE XST** . The RTL schematic of the design is shown below :



**THANK**

**YOU…**

****

**SUBMITTED TO :**

**Mr. Pragati Singh**

**Assistant professor ( Adhoc )**

**Electronics And Communication department**