

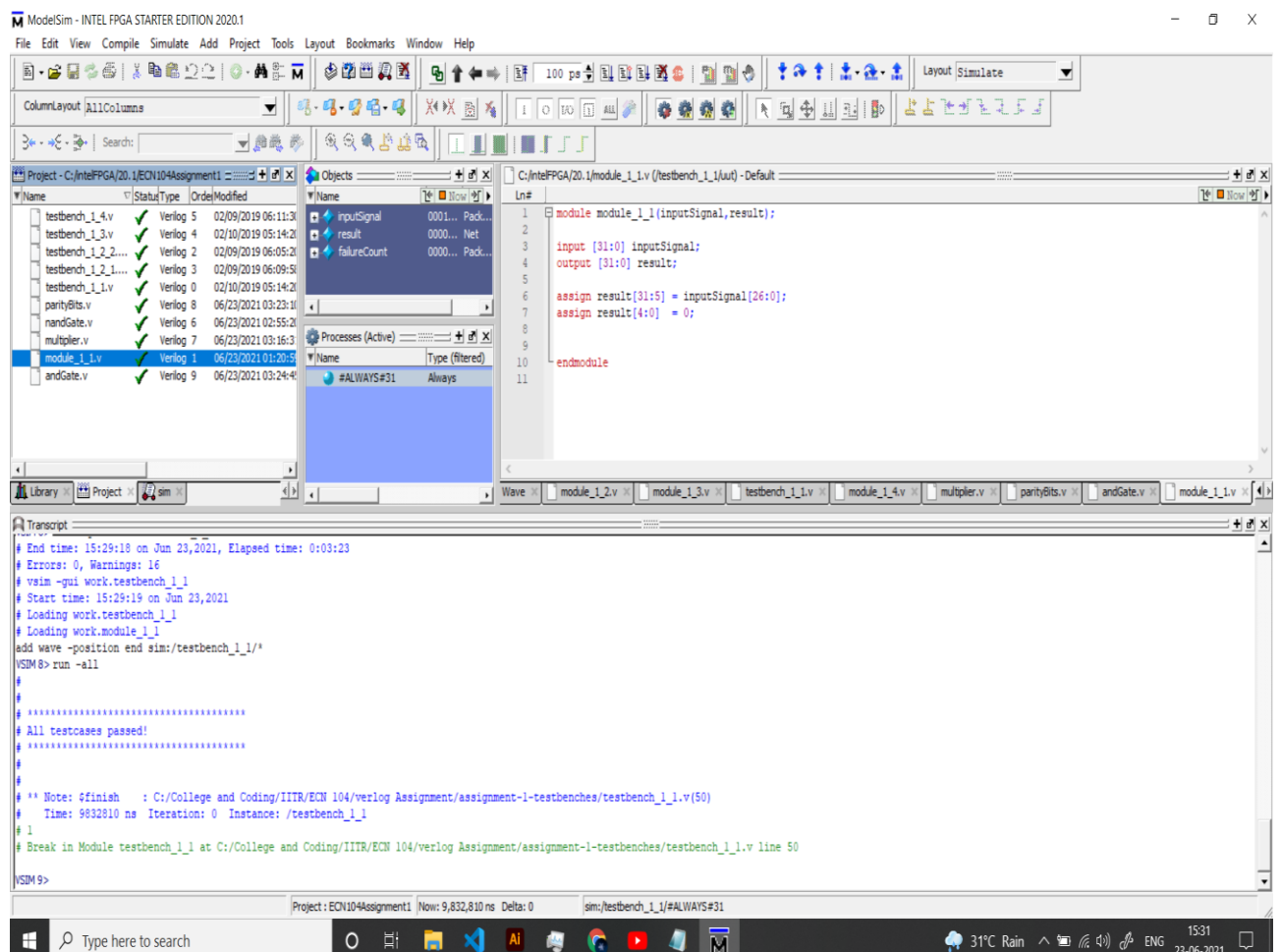
# VERILOG ASSIGNMENT -1

RAHUL KURKURE  
20114079 CSE 03

Q1. Write a Verilog module named module\_1\_1 which uses part select, takes a 32bit vector as input named inputSignal and outputs a 32bit vector result which is simply inputSignal shifted by 5 towards left. Use testbench\_1\_1.v to verify the output. (Hint: Similar to part select, Verilog supports part assign which allows assigning values to a part of a vector).

Explanation : we are going to use part select method in this question , we will set indices number 26 to 0 from index of input vector as the bits of indices 31 to 5 of result vector and for the bits number 4 to 0 we will set them as 0.

Source code :

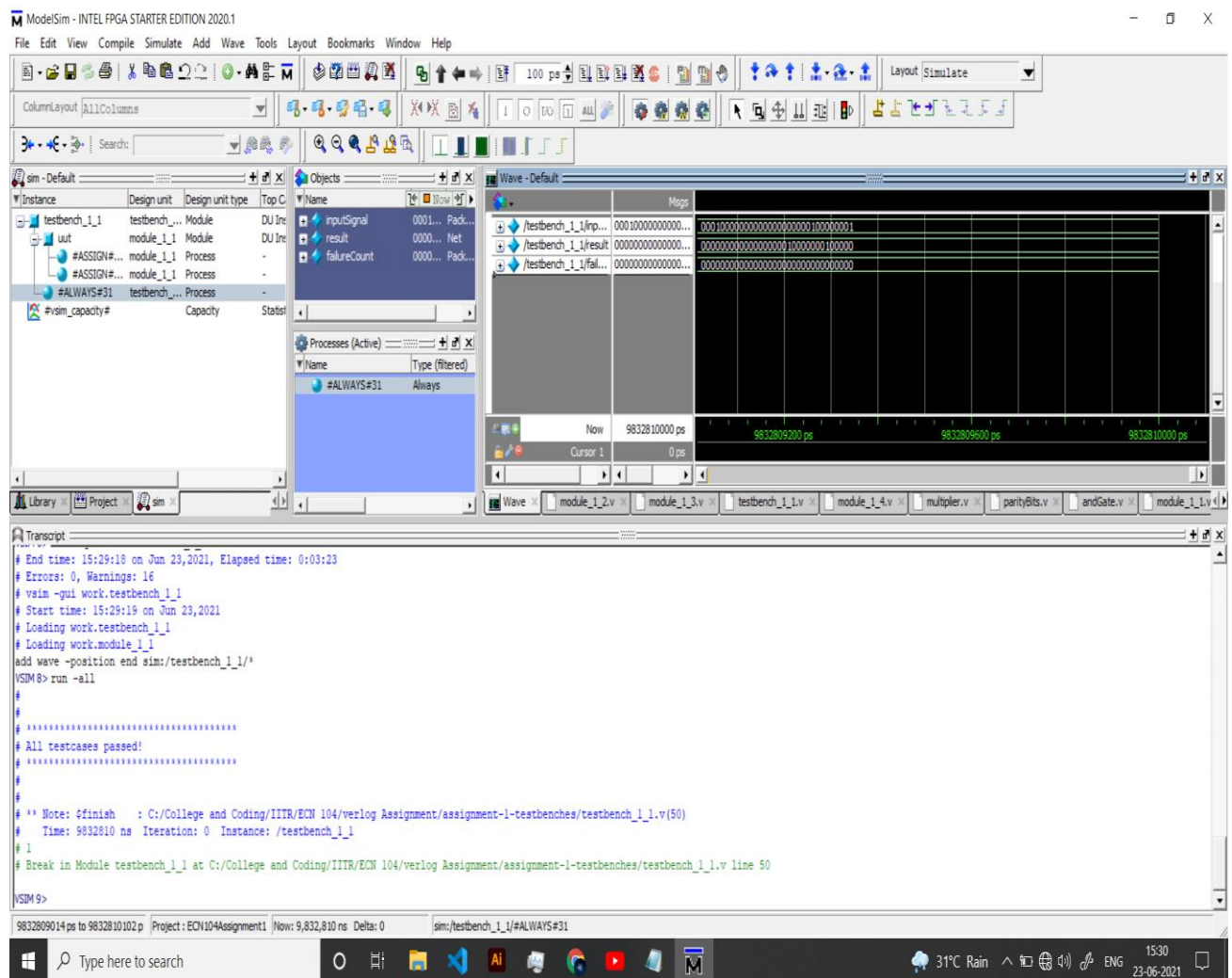


The screenshot displays the ModelSim - Intel FPGA Starter Edition 2020.1 interface. The main window shows the Verilog code for the module\_1\_1, which is a 32-bit shift register. The code is as follows:

```
1 module module_1_1(inputSignal,result);
2
3 input [31:0] inputSignal;
4 output [31:0] result;
5
6 assign result[31:5] = inputSignal[26:0];
7 assign result[4:0] = 0;
8
9
10
11 endmodule
```

The left pane shows the project structure, including the testbench\_1\_1.v file. The bottom pane shows the simulation results, indicating that all testcases passed. The simulation was run on June 23, 2021, at 15:29:18, and the elapsed time was 0:03:23. The simulation was performed using the VLSI tool, and the results were saved in the file testbench\_1\_1.v.

## Simulation :



Q2. Hierarchical design is often helpful in verifying large project easily by verifying each individual module separately. Make a module for 2 input NAND gate named nandGate. Now, make a 2 input AND gate named andGate using multiple of these NAND gates. Use testbench\_1\_2\_1.v for verifying the output of NAND gate and testbench\_1\_2\_2.v for verifying the output of the AND gate.

### Explanation :

Here we are first going to make a nand gate by calculating the and of A and B using & operator ,then inverting the result will give us NAND of A and B .

NAND :  $\sim (A \& B)$

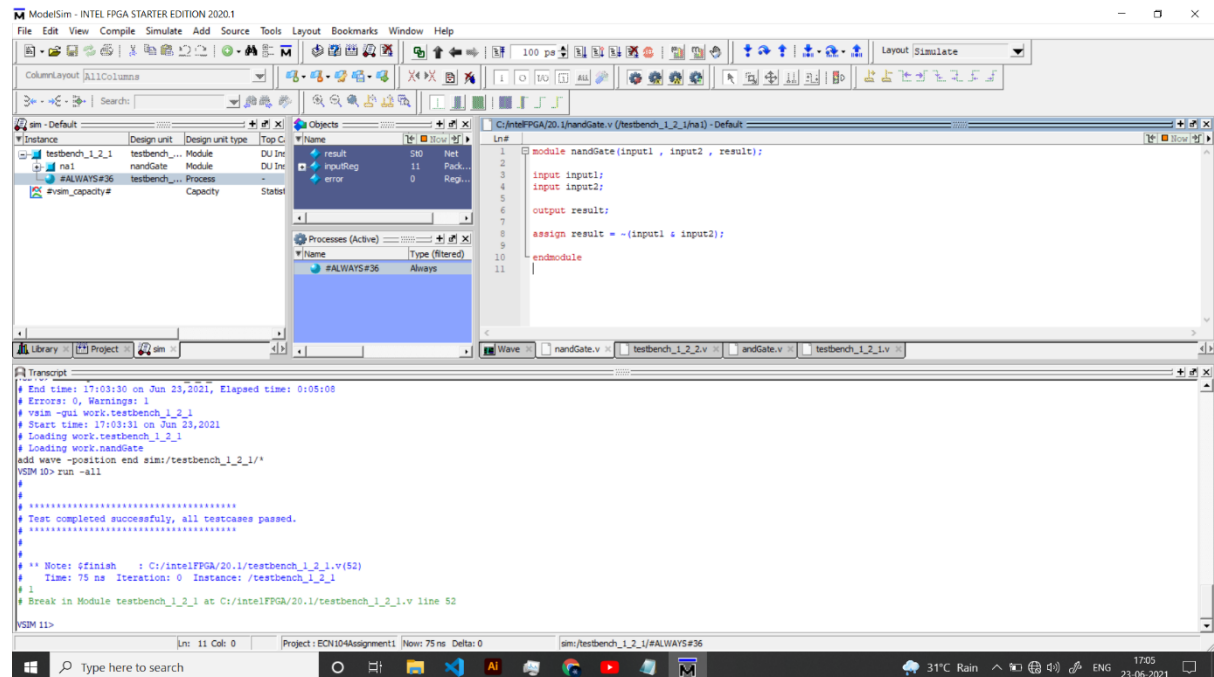
According to question we have to use NAND gates to make AND gates so first we will take nand of both inputs then we will take another NAND to invert our result .

Step 1 :  $A \text{ NAND } B : (AB)' \rightarrow A' + B'$

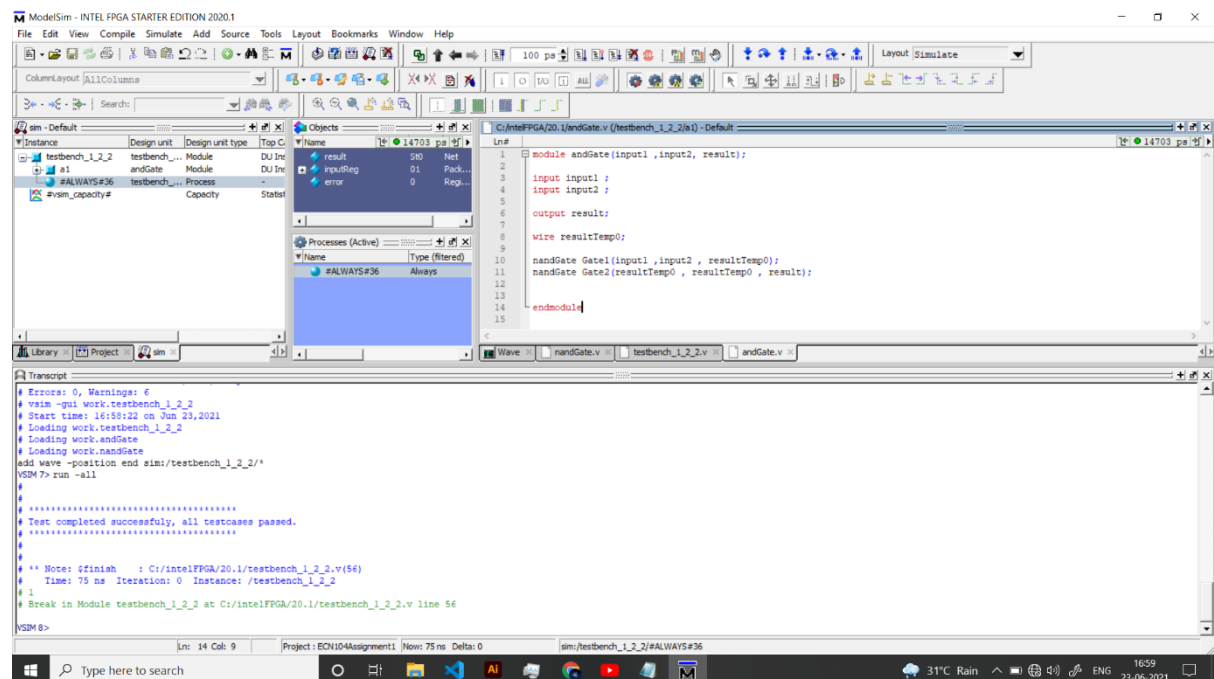
Step 2 :  $(AB)' \text{ NAND } (AB)'$

Result :  $((AB)' (AB)')' \rightarrow AB$

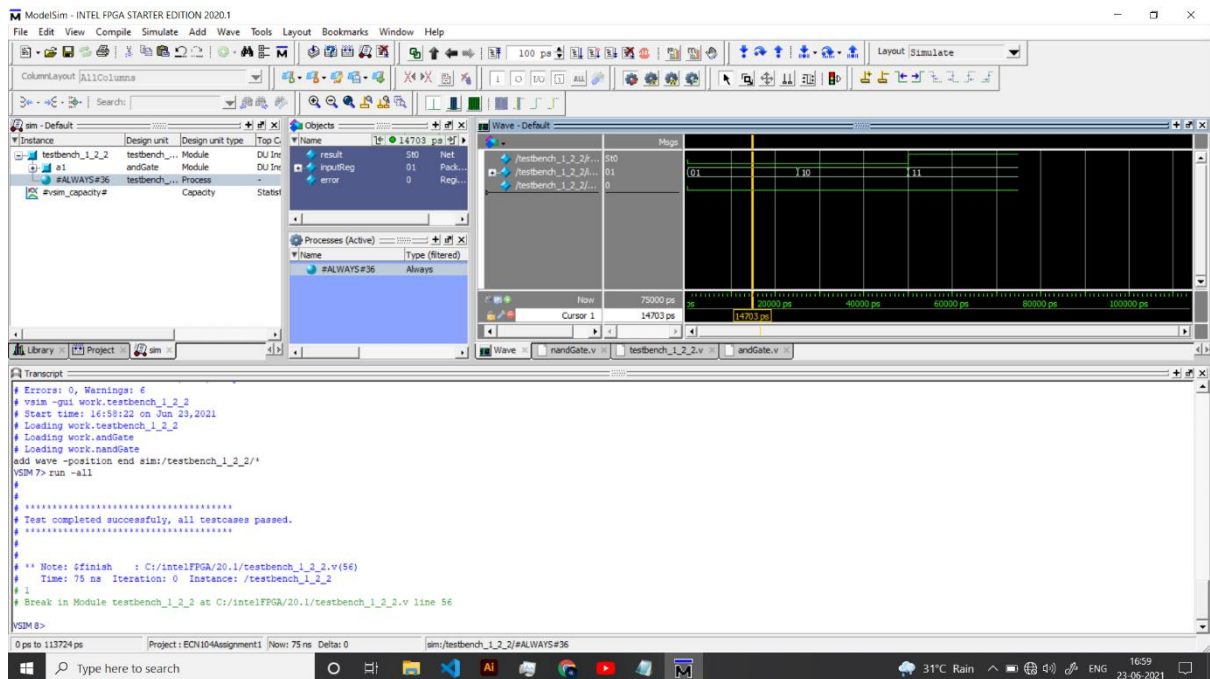
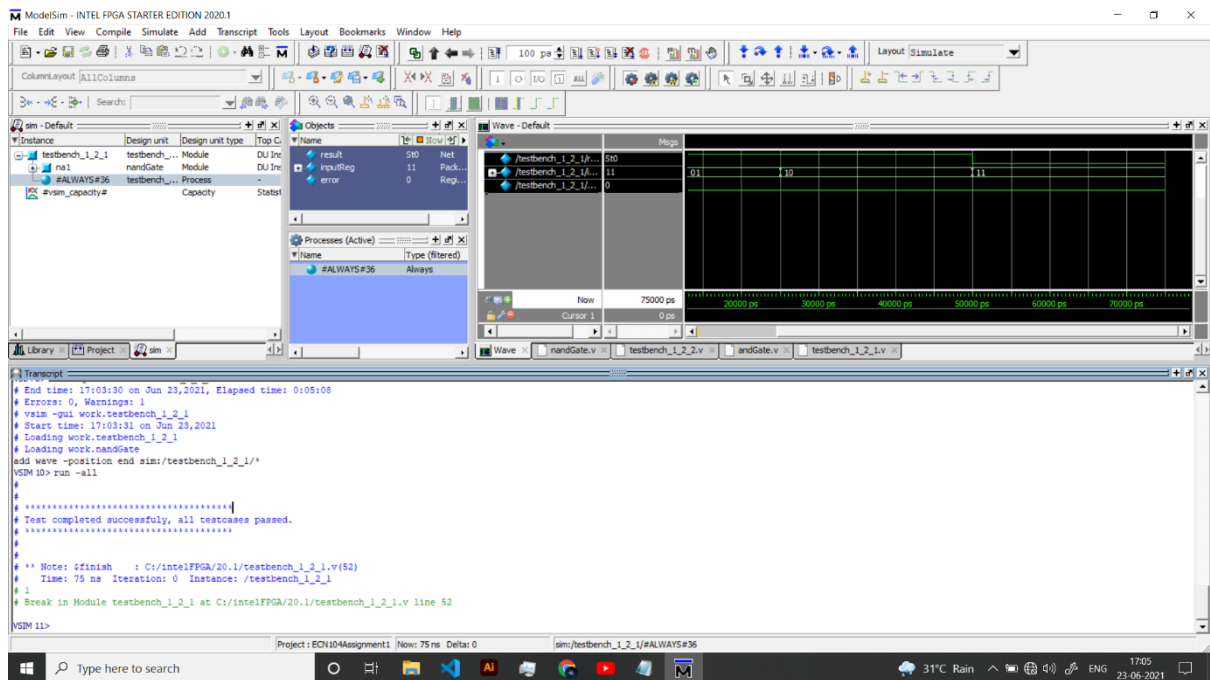
Source Code : NAND Gate :



AND Gate :



## Simulations :

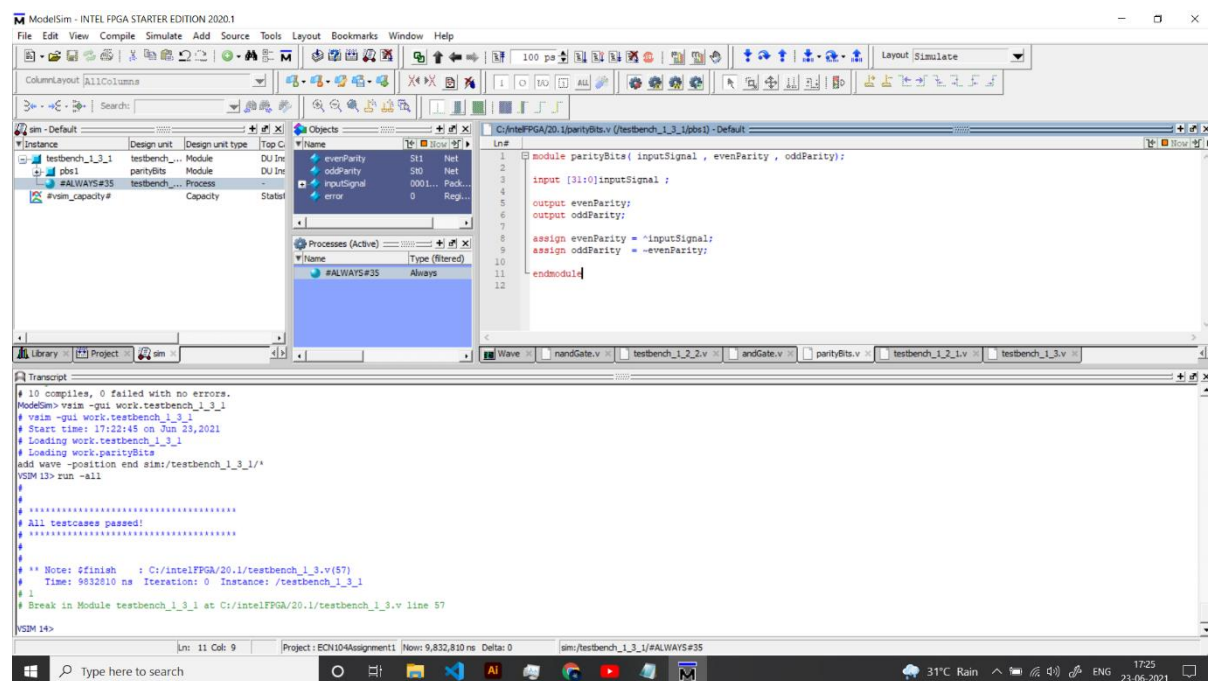


Q3. Write a Verilog module named parityBits which uses reduction operator, takes a 32bit vector as input named signal, two single bit output named parityEven and parityOdd. Use testbench\_1\_3.v to verify output. (Hint: Calculating XOR of all bits of a signal gives even parity of the signal.)

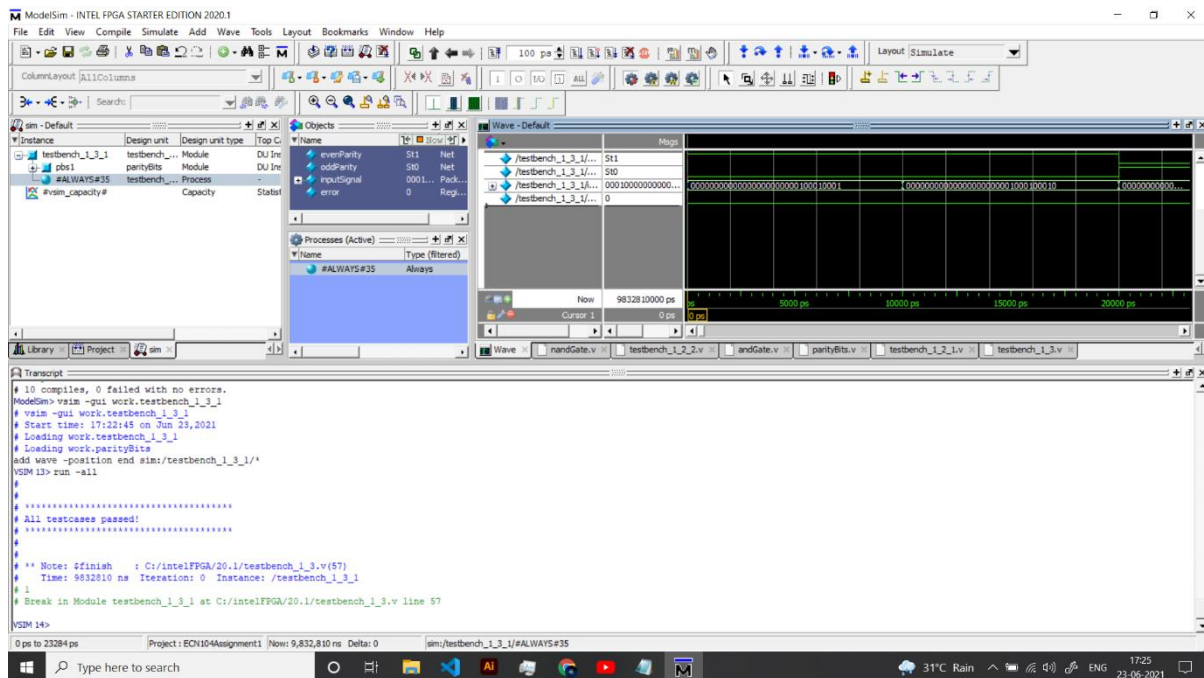
Explanation :

As hinted in the question we will use an ^ xor operator to get the even parity signal , if we just invert the even parity signal we will get an Odd parity signal .

Source Code :



## Simulation :



Q4. Write a module named multipiler for multiplying a 32-bit binary number by 10(4'b1010 in binary) without using the multiply operator. Assume the input is such that output doesn't overflow. Use testbench\_1\_4.v to verify output. RTL schematic of your module should resemble Fig. 4. The input to the module should be named inputSignal and the output should be named result to allow the testcases to run.

Explanation : Adding zeroes to the binary numbers wont make any difference so we will completely ignore the multiplication by zero , for first one we will shift the bits to the left by 1 and for next time multiplication by 1 we will shift the bits by 3 units to the left and then finnaly add those two numbers.



The screenshot displays the Intel FPGA Starter Edition 2020.1 IDE. The main window is divided into several panes:

- Menu Bar:** File, Edit, View, Compile, Simulate, Add, Source, Tools, Layout, Bookmarks, Window, Help.
- Toolbar:** Contains icons for various functions like opening files, saving, and running simulations.
- Project Browser (Left):** Shows the project structure with folders like 'sim', 'testbench\_1\_4', and 'multiplier'.
- Object Browser (Center):** Displays the hierarchy of objects in the design, including 'result', 'inputSignal', and 'error'.
- Code Editor (Right):** Shows the Verilog code for the 'multiplier' module. The code defines a module with an input 'inputSignal' and an output 'result', and implements a multiplier using two temporary variables 'temp1' and 'temp3'.
- Transcript Window (Bottom):** Displays the simulation results, showing that all testcases passed and the simulation completed successfully.

The Verilog code in the code editor is as follows:

```

module multiplier (inputSignal, result);
    input [31:0] inputSignal;
    output [31:0] result;

    wire [31:0] temp1;
    wire [31:0] temp3;

    assign temp1 = inputSignal << 1;
    assign temp3 = inputSignal << 3;
    assign result = temp1 + temp3;

endmodule

```

The transcript window shows the following output:

```

# 10 compilations failed with no errors.
ModelSim> vsim -gui work.testbench_1_4
# vsim -gui work.testbench_1_4
# Start time: 17:33:01 on Jun 23, 2021
# Loading work.testbench_1_4
# Loading work.multiplier
add wave -position end sim:/testbench_1_4/
VSM 16> run -all

.....
All testcases passed!
.....

** Note: dfinish : C:/intelFPGA/20.1/testbench_1_4.v[55]
Time: 9632010 ns Iteration: 1 Instance: /testbench_1_4
# 1
# Break in Module testbench_1_4 at C:/intelFPGA/20.1/testbench_1_4.v line 55
VSM 17>

```

ModelSim - Intel FPGA Starter Edition 2020.1

File Edit View Compile Simulate Add Transport Tools Layout Bookmarks Window Help

ColumnLayout [AllColumns]

om - Default

Design unit Design unit type Top C

testbench\_1\_4 Module DU Int

mul1 Module DU Int

#ALWAYS#45 testbench\_1\_4 Process -

#ALWAYS#45 testbench\_1\_4 Process -

#vsim\_capacity# Capacity Status

Objects

result 1010... Net

inpSignal 0001... Pack

error 0 Reg

Processes (Active)

#ALWAYS#45 Always

Wave - Default

Now 9832810000 ps

Cursor 1 0 ps

9832809200 ps 9832809600 ps 98328100 ps 98328100 ps

Transcript

```
# 10 compiles, 0 failed with no errors.
ModelSim> vsim -gui work.testbench_1_4
# vsim -gui work.testbench_1_4
# Start time: 17:33:01 on Jun 23, 2021
# Loading work.testbench_1_4
# Loading work.multiplexer
add wave -position end sim:/testbench_1_4/*
VSM 16> run -all

*****
All testcases passed!
*****

** Note: $finish : C:/IntelFPGA/20.1/testbench_1_4.v(55)
Time: 9032010 ns Iteration: 1 Instance: /testbench_1_4
# 1
# Break in Module testbench_1_4 at C:/IntelFPGA/20.1/testbench_1_4.v line 55

VSM 17>
```

Project: ECH104Assignment1 Now: 9,832,810 ns Delta: 1 sim:/testbench\_1\_4/#ALWAYS#45