Department of Computer Science and Engineering
Pandit Deendayal Energy University
Digital Electronics and Computer Organization Lab – **20CP203P**

Lab Assignment 3

In this Lab, we shall learn 'Module Instantiation' as well as writing a Verilog code using a 'Structural' and 'Behavioral' method. You will write the code for one module; the same module will be used in other module of your Verilog Code.

A module provides a template from which you can create actual objects. When a module is invoked, Verilog creates a unique object from the template. Each object has its own name, variables, parameters, and I/O interface. The process of creating objects from a module template is called instantiation, and the objects are called instances.

Previously, we have used these instances in the Testbench code. Now we shall use it in Verilog Code also.

Question: 1 Implement the following expression using the Verilog Hardware Description Language (HDL) then compare with truth table whether your circuit produced same output or not? Moreover, verify your circuit against the waveform.

1. F(A, B, C)= A'BC + AB'C + ABC
2. F(A, B, C, D)=ABCD' + A'BC D+ AB'CD' + ABC
3. F(A, B, C, D, E, F) = ABC + DE + F
4. F(A, B) = (A'+B') (A+B') (A'+B) (A+B)
5. F(A, B) = ((A.B')'+((A)'(B)')')'
6. F(A, B) = (((A)'+B)'. ((A)'+(B)')')'

Question: 2 Implement the XOR gate using Behavioural and Structural code (shown in Fig: 1) of Verilog Hardware Description Language.

```
 1   module Lab4_Build_XOR(a, b, c);
 2   input a, b;
 3   output c;
 4   wire a_not, b_not;
 5   wire x, y;
 6   not(a_not, a);
 7   not(b_not, b);
 8   and(x, a_not, b);
 9   and(y, a, b_not);
10   or(c, x, y);
11   endmodule
```

Fig. 1: Example Structural code of XOR gate operation.

Question: 3: Implement the following expression using the Verilog Hardware Description Language (HDL) (Structural Coding).

1. F (A, B, C) = (A'+B'+C')(A+B'+C')(A'+B+C')(A'+B')`
2. F (A, B, C, D, E) = ((A+B).(C+D+E))'

Question: 3 Implement the following expression using universal NAND and NOR gate. Write down Verilog Structural and Behavioral code for that expression.

1. F (A, B, C) = (AB'C) + (AB'C')
2. F (A, B) = A'B' + AB' + A'B + AB

Question: 4 Write down three modules in a single Verilog code to design AND, OR and NOT gate. Now use those modules to design the XOR gate. Use AND, OR and NOT gate as the instances to implement the XOR gate. Write the corresponding Testbench code for the verification of your XOR gate.

Truth Table of XOR Gate:

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Question: 5 Consider the following expression:

$$Y = A'.B'.C' + A'.B.C' + A.B'.C' + A.B'.C$$

(i) Generate the Truth Table manually for the same.

(ii) Design one three input 'And' gate module and one four input 'OR' gate module using Verilog. Instantiate those two modules to design the above-mentioned expression. Design the corresponding TestBench code for the verification purpose.

(iii) Now Minimize the given expression.

(iv) Design again 'AND' gate and 'OR' gate module with required number of inputs and instantiate them to implement the minimized expression. Design the corresponding TestBench code for the verification purpose.

- **Sample Verilog Design Code for Instantiation of module:**

```
module not_gate (input e, output f);
  assign f = ~e;
```

```
endmodule

module and_gate (input a, b, output c);
  assign c = a & b;
endmodule

module or_gate (input p, q, output r);
  assign r = p | q;
endmodule

module build_xor (input m, n, output o);
  wire x, y, a_not, b_not;
  not_gate n1 (.e(m),.f(a_not));
  not_gate n2 (.e(n),.f(b_not));
  and_gate a1 (.a(a_not),.b(n),.c(x));
  and_gate a2 (.a(m),.b(b_not),.c(y));
  or_gate o1 (.p(x),.q(y),.r(o));

endmodule
```

- **Sample TestBench Code for Question 1:**

```
module tb_xor_str;
   reg A,B;
   wire C;
 build_xor x1 (.m(A), .n(B), .o(C));
 initial
   begin
 A = 0; B =0; #5;
 A = 0; B =1; #5;
 A = 1; B =0; #5;
 A = 1; B =1; #5;
   end
 initial
 begin
   $dumpfile("dump.vcd");
   $dumpvars(1);
 end
endmodule
```

Submission Instructions:

- Prepare the submission file according to the following process:
    1. Copy the Verilog code, the Test Bench Code in a Word File.
    2. Take the ScreenShot of Waveform and paste into the same word file.

3. Repeat Step 1 and 2 for all the programs.
4. Copy and Paste all the Verilog code, Testbench Code and Waveform into a single word file as 1_verilog, 1_TestBench, 1_Waveform, 2_verilog, 2_TestBench, 2_Waveform… etc.
5. Convert it into pdf file, Print it and prepare a file for the verification in the next lab.