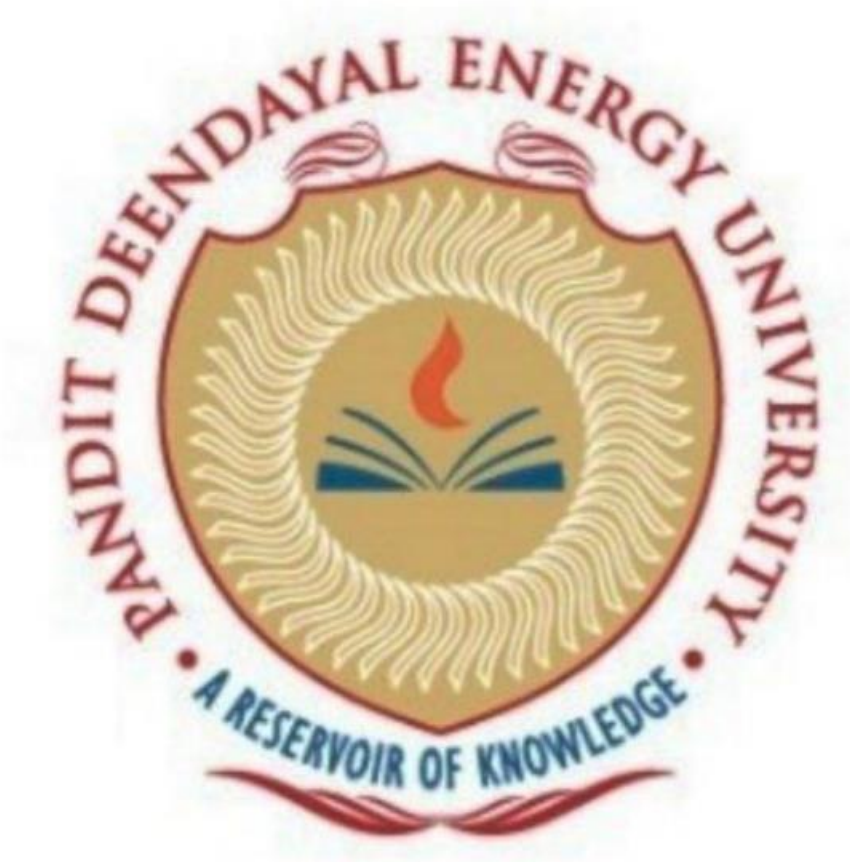
Name - Akshat Shah

Roll No - 21BCP322

Sem – III

Digital Electronics and Computer Organization Lab

Course Code – 20CP203P



Department of Computer Science Engineering
School of Technology,
Pandit Deendayal Energy University

Date: 24-08-2022

Assignment-3

Aim: To learn module instantiation using structural and behavioral methods.

Table of Contents

List of Questions:

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.

1.1 F(A, B, C)= A'BC + AB'C + ABC

Verilog Code:

testbench.sv

```
// Akshat Shah
// 21BCP322

// test bench code

module lab3_que1_tb;    // creating a module
  reg A,B,C;            //declaring reg to store input values to be given
  wire F;               // connecting to get output values
  lab3_que1 a1(A,B,C,F); // Instantiating by order as declared in module


  initial
    begin               // beginning execution

      // assigning values as input
      A=0;B=0;C=0;#1;
      $display("A=%b and B=%b and C=%b",A,B,C);
      $display("F=%b",F);

      A=0;B=0;C=1;#1;
      $display("A=%b and B=%b and C=%b",A,B,C);
      $display("F=%b",F);

      A=0;B=1;C=0;#1;
      $display("A=%b and B=%b and C=%b",A,B,C);
      $display("F=%b",F);

      A=0;B=1;C=1;#1;
      $display("A=%b and B=%b and C=%b",A,B,C);
      $display("F=%b",F);

      A=1;B=0;C=0;#1;
      $display("A=%b and B=%b and C=%b",A,B,C);
      $display("F=%b",F);

    end     // ending execution
  initial
    begin
      $dumpfile("dump.vcd");        // for storing information of waveform
      $dumpvars(1);         // specifying part of design to be stored as waveform

    end
endmodule       // enclosing the module
```

design.sv

```
// F(A, B, C)= A'BC + AB'C + ABC
module lab3_que1(a,b,c,f);
  input a,b,c;
  output f;
  assign f = ((~a)&b&c)|(a&(~b)&c)|(a&b&(~c));
endmodule
```

Fig 1.1.1- Testbench and design code

Output:

```
A=0 and B=0 and C=0
F=0
A=0 and B=0 and C=1
F=0
A=0 and B=1 and C=0
F=0
A=0 and B=1 and C=1
F=1
A=1 and B=0 and C=0
F=0
```

Fig 1.1.2 - Output as Truth table

Waveform:

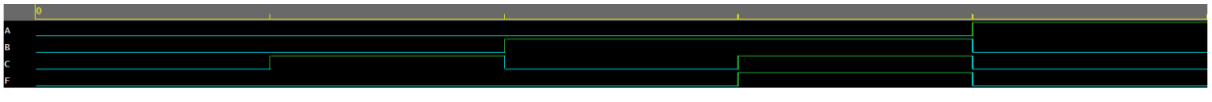


Fig 1.1.3- Waveform for output

1.2 F(A, B, C, D) = ABCD’ + A’BC D+ AB’CD’ + ABC

Verilog Code:

testbench.sv

```
// Akshat Shah
// 21BCP322

module lab3_que1_2_tb;  // creating a module
  reg A,B,C,D;
  wire F;
  lab3_que1_2 a1(A,B,C,D,F);    // Instantiating by order as declared in module


  initial
    begin                  // beginning execution

      // assigning values as input
      A=0;B=0;C=0;D=0;#1;
      $display("A=%b and B=%b and C=%b and D=%b",A,B,C,D);
      $display("F=%b",F);

      A=0;B=0;C=1;D=1;#1;
      $display("A=%b and B=%b and C=%b and D=%b",A,B,C,D);
      $display("F=%b",F);

      A=0;B=1;C=0;D=0;#1;
      $display("A=%b and B=%b and C=%b and D=%b",A,B,C,D);
      $display("F=%b",F);

      A=0;B=1;C=1;D=1;#1;
      $display("A=%b and B=%b and C=%b and D=%b",A,B,C,D);
      $display("F=%b",F);

      A=1;B=0;C=0;D=0;#1;
      $display("A=%b and B=%b and C=%b and D=%b",A,B,C,D);
      $display("F=%b",F);

    end     // ending execution
  initial
    begin
      $dumpfile("dump.vcd");        // for storing information of waveform
      $dumpvars(1);         // specifying part of design to be stored as waveform
    end
endmodule       // enclosing the module
```

design.sv

```
// F(A, B, C, D) = ABCD’ + A’BC D+ AB’CD’ + ABC

module lab3_que1_2(a,b,c,d,f);
  input a,b,c,d;
  output f;
  assign f = (a&b&c&(~d))|((~a)&b&c&d)|(a&(~b)&c&(~d))| (a&b&c);
endmodule
```

Fig 1.2.1- Testbench and design code

Output:

```
A=0 and B=0 and C=0 and D=0
F=0
A=0 and B=0 and C=1 and D=1
F=0
A=0 and B=1 and C=0 and D=0
F=0
A=0 and B=1 and C=1 and D=1
F=1
A=1 and B=0 and C=0 and D=0
F=0
```

Fig 1.2.2 - Output as truth table

Waveform:

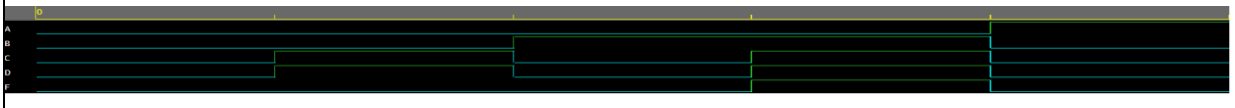


Fig 1.2.3- Waveform of output

1.3 F(A, B, C, D, E, F) = ABC + DE + F

Verilog Code:

testbench.sv

```
// Akshat Shah
// 21BCP322

module lab3_que1_3_tb;  // creating a module
  reg A,B,C,D,E,F;
  wire W;
  lab3_que1_3 a1(A,B,C,D,E,F,W);    // Instantiating by order as declared in module


  initial
    begin                        // beginning execution

      // assigning values as input
      A=0;B=0;C=0;D=0;E=0;F=0;#1;
      $display("A=%b and B=%b and C=%b and D=%b and E=%b and F=%b",A,B,C,D,E,F);
      $display("W=%b",W);

      A=0;B=0;C=0;D=0;E=0;F=1;#1;
      $display("A=%b and B=%b and C=%b and D=%b and E=%b and F=%b",A,B,C,D,E,F);
      $display("W=%b",W);

     A=0;B=0;C=0;D=0;E=1;F=0;#1;
      $display("A=%b and B=%b and C=%b and D=%b and E=%b and F=%b",A,B,C,D,E,F);
      $display("W=%b",W);

     A=0;B=0;C=0;D=0;E=1;F=1;#1;
      $display("A=%b and B=%b and C=%b and D=%b and E=%b and F=%b",A,B,C,D,E,F);
      $display("W=%b",W);


      end       // ending execution
  initial
    begin
      $dumpfile("dump.vcd");        // for storing information of waveform
      $dumpvars(1);         // specifying part of design to be stored as waveform

    end
endmodule   // enclosing the module
```

design.sv

```
// F(A, B, C, D, E, F) = ABC + DE + F

module lab3_que1_3(a,b,c,d,e,f,w);
  input a,b,c,d,e,f;
  output w;
  assign w = (a&b&c)|(d&e)|(f);
endmodule
```

Fig 1.3.1- Testbench and design code

Output:

```
A=0 and B=0 and C=0 and D=0 and E=0 and F=0
W=0
A=0 and B=0 and C=0 and D=0 and E=0 and F=1
W=1
A=0 and B=0 and C=0 and D=0 and E=1 and F=0
W=0
A=0 and B=0 and C=0 and D=0 and E=1 and F=1
W=1
```

Fig 1.3.2 - Output as truth table

Waveform:

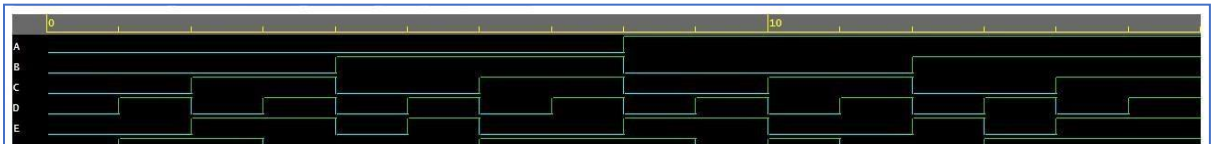


Fig 1.3.3- Waveform of output

1.4 F(A, B)= (A'+B') (A+B') (A'+B) (A+B)

Verilog Code:

testbench.sv

```
// Akshat Shah
// 21BCP322

module lab3_que1_tb;    // creating a module
  reg A,B;
  wire F;
  lab3_que1 a1(A,B,F);  // Instantiating by order as declared in module


  initial
    begin               // beginning execution

      // assigning values as input
      A=0;B=0;#1;
      $display("A=%b and B=%b ",A,B);
      $display("F=%b",F);

      A=0;B=1;#1;
      $display("A=%b and B=%b ",A,B);       // for display of values entered
      $display("F=%b",F);

      A=1;B=0;#1;
      $display("A=%b and B=%b",A,B);
      $display("F=%b",F);

      A=1;B=1;#1;
      $display("A=%b and B=%b ",A,B);
      $display("F=%b",F);

      A=1;B=0;#1;
      $display("A=%b and B=%b",A,B);
      $display("F=%b",F);

    end     // ending execution
  initial
    begin
      $dumpfile("dump.vcd");    // for storing information of waveform
      $dumpvars(1);         // specifying part of design to be stored as waveform
    end
endmodule       // enclosing the module
```

design.sv

```
// F(A, B)= (A'+B')(A+B')(A'+B)(A+B)
module lab3_que1(a,b,f);
  input a,b;
  output f;
  assign f = ((~a)|(~b))& (a|(~b)) & ((~a)|b) & (a|b);
endmodule
```

Fig 1.4.1- Testbench and design code

Output:

```
A=0 and B=0
F=0
A=0 and B=1
F=0
A=1 and B=0
F=0
A=1 and B=1
F=0
A=1 and B=0
F=0
```

Fig 1.4.2- output as truth table

Waveform:

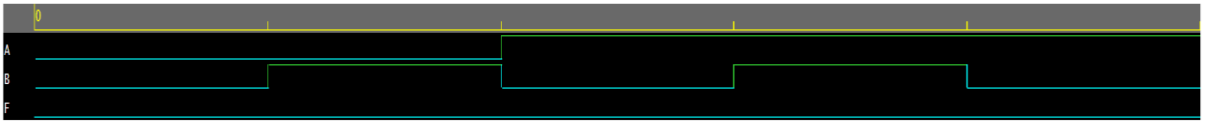


Fig 1.4.3- Waveform of output

1.5 F(A, B)= ((A.B')'+((A)'(B)')')'

Verilog Code:

testbench.sv

```
// Akshat Shah
// 21BCP322

module lab3_que1_tb;    // creating a module
  reg A,B;
  wire F;
  lab3_que1 a1(A,B,F);  // Instantiating by order as declared in module


  initial
    begin               // beginning execution

      // assigning values as input
      A=0;B=0;#1;
      $display("A=%b and B=%b ",A,B);
      $display("F=%b",F);

      A=0;B=1;#1;
      $display("A=%b and B=%b ",A,B);       // for display of values entered
      $display("F=%b",F);

      A=1;B=0;#1;
      $display("A=%b and B=%b",A,B);
      $display("F=%b",F);

      A=1;B=1;#1;
      $display("A=%b and B=%b ",A,B);
      $display("F=%b",F);

      A=1;B=0;#1;
      $display("A=%b and B=%b",A,B);
      $display("F=%b",F);

    end     // ending execution
  initial
    begin
      $dumpfile("dump.vcd");    // for storing information of waveform
      $dumpvars(1);         // specifying part of design to be stored as waveform
    end
endmodule       // enclosing the module
```

SV/Verilog Testbench

design.sv

```
// F(A, B)= ((A.B')'+((A)'(B)')')'

module lab3_que1(a,b,f);
  input a,b;
  output f;
  assign f = ~(~((~(a&(~b)))| (~((~a)&(~b))))) ;
endmodule
```

Fig 1.5.1- Testbench and design code

Output:

```
A=0 and B=0
F=1
A=0 and B=1
F=1
A=1 and B=0
F=1
A=1 and B=1
F=1
A=1 and B=0
F=1
```

Fig 1.5.2- Output as truth table

Waveform:

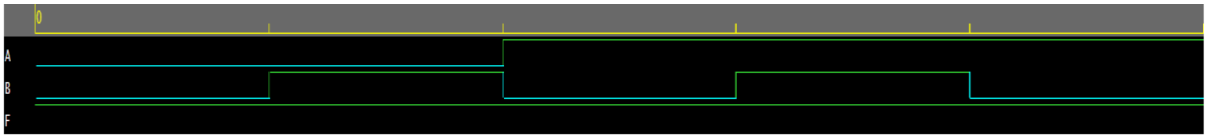


Fig 1.5.3- Waveform of output

1.6 F(A, B)= (((A)'+B)'. ((A)'+(B)')')'

Verilog code:

testbench.sv

```
// Akshat Shah
// 21BCP322

module lab3_que1_tb;    // creating a module
  reg A,B;
  wire F;
  lab3_que1 a1(A,B,F);  // Instantiating by order as declared in module


  initial
    begin               // beginning execution

      // assigning values as input
      A=0;B=0;#1;
      $display("A=%b and B=%b ",A,B);
      $display("F=%b",F);

      A=0;B=1;#1;
      $display("A=%b and B=%b ",A,B);       // for display of values entered
      $display("F=%b",F);

      A=1;B=0;#1;
      $display("A=%b and B=%b",A,B);
      $display("F=%b",F);

      A=1;B=1;#1;
      $display("A=%b and B=%b ",A,B);
      $display("F=%b",F);

      A=1;B=0;#1;
      $display("A=%b and B=%b",A,B);
      $display("F=%b",F);

    end     // ending execution
  initial
    begin
      $dumpfile("dump.vcd");    // for storing information of waveform
      $dumpvars(1);         // specifying part of design to be stored as waveform
    end
endmodule       // enclosing the module
```

design.sv

```
// F(A, B)= (((A)'+B)'. ((A)'+(B)')')'

module lab3_que1(a,b,f);
  input a,b;
  output f;
  assign f = ~(~((~a)|b)& (~((~a)|(~b)))) ;
endmodule

```

Fig 1.6.1- Testbench and design code

Output:

```
A=0 and B=0
F=1
A=0 and B=1
F=1
A=1 and B=0
F=1
A=1 and B=1
F=1
A=1 and B=0
F=1
```

Fig 1.6.2- Output as truth table

Waveform:

Fig 1.6.3- Waveform of output

Question 2: Implement the XOR gate using Behavioural and Structural code of Verilog Hardware Description Language.

Verilog Code:

testbench.sv

```
// Akshat Shah
// 21BCP322

// test bench code for XOR gate

module lab3_que2_tb;    // creating a module
  reg A,B;
  wire F;
  lab3_que2 a1(A,B,F);  // Instantiating by order as declared in module


  initial
    begin               // beginning execution

      // assigning values as input
      A=0;B=0;#1;
      $display("A=%b and B=%b ",A,B);
      $display("F=%b",F);

      A=0;B=1;#1;
      $display("A=%b and B=%b ",A,B);       // for display of values entered
      $display("F=%b",F);

      A=1;B=0;#1;
      $display("A=%b and B=%b",A,B);
      $display("F=%b",F);

      A=1;B=1;#1;
      $display("A=%b and B=%b ",A,B);
      $display("F=%b",F);



    end     // ending execution
  initial
    begin
      $dumpfile("dump.vcd");    // for storing information of waveform
      $dumpvars(1);         // specifying part of design to be stored as waveform
    end
endmodule       // enclosing the module
```

design.sv

```
// XOR GATE

// Behavioral Code

module lab3_que2(a,b,f);
  input a,b;
  output f;
  assign f = ((~a)&b| a&(~b)) ;
endmodule




// Structural code

/* module lab3_que2(a,b,c);
input a,b;
output f;
wire a_not,b_not,x,y;
not(a_not,a);
not(b_not,b);
and(x,a_not,b);
and(y,a,b_not);
or(f,x,y);
end module
*/
```

Fig 2.1- Testbench and design code

Output:

```
A=0 and B=0
F=0
A=0 and B=1
F=1
A=1 and B=0
F=1
A=1 and B=1
F=0
```

Fig 2.2 Output as truth table

Waveform:

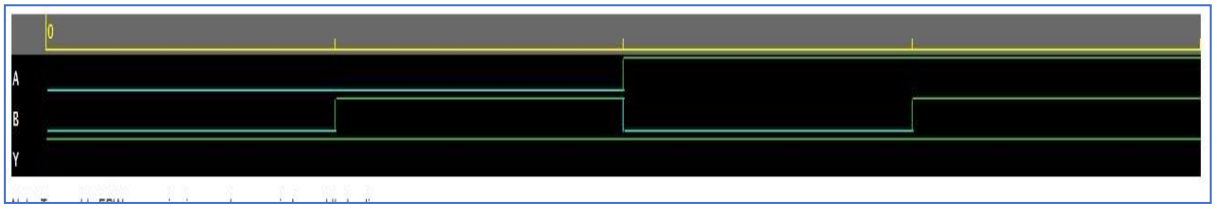


Fig 2.3- Waveform

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

3.1 F(A, B, C)=(A'+B'+C')(A+B'+C')(A'+B+C')(A'+B')'

Verilog Code:

testbench.sv

```
// Akshat Shah
// 21BCP322

// Test bench code


module lab3_que3_tb;    // creating a module
  reg A,B,C;
  wire F;
  lab3_que3 a1(A,B,C,F);    // Instantiating by order as declared in module


  initial
    begin               // beginning execution

      // assigning values as input
      A=0;B=0;C=0;#1;
      $display("A=%b and B=%b and C=%b",A,B,C);
      $display("F=%b",F);

      A=0;B=0;C=1;#1;
      $display("A=%b and B=%b and C=%b",A,B,C); // for display of values entered
      $display("F=%b",F);

     A=0;B=1;C=0;#1;
      $display("A=%b and B=%b and C=%b",A,B,C);
      $display("F=%b",F);

     A=0;B=1;C=1;#1;
      $display("A=%b and B=%b and C=%b",A,B,C);
      $display("F=%b",F);



    end     // ending execution
  initial
    begin
      $dumpfile("dump.vcd");    // for storing information of waveform
      $dumpvars(1);         // specifying part of design to be stored as waveform
    end
endmodule       // enclosing the module
```

design.sv

```
// F(A, B, C)=(A'+B'+C')(A+B'+C')(A'+B+C')(A'+B')'

// Structural code

module lab3_que3(a,b,c,f);
input a,b,c;
output f;
wire a_not,b_not,c_not;
wire p,q,r,s,t;
not(a_not,a);
not(b_not,b);
not(c_not,c);
or(r,a_not,b_not,c_not);
or(s,a,b_not,c_not);
or(t,a_not,b,c_not);
or(q,a_not,b_not);
   not(p,q);
   and(f,r,s,t,p);

  endmodule

```

Fig 3.1.1 - Testbench and design code

Output:

```
A=0 and B=0 and C=0
F=0
A=0 and B=0 and C=1
F=0
A=0 and B=1 and C=0
F=0
A=0 and B=1 and C=1
F=0
```

Fig 3.1.2- Output as truth table

Waveform:

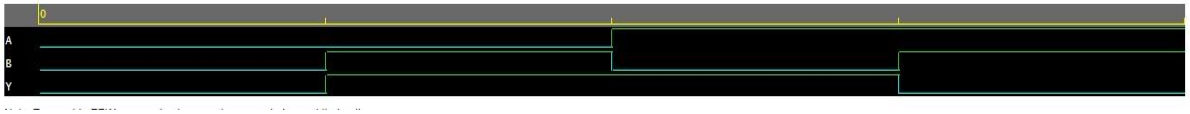


Fig 3.1.3- Output as truth table

3.2 F(A, B, C, D, E) = ((A+B).(C+D+E))’

Verilog code:

testbench.sv

```
// Akshat Shah
// 21BCP322

// Test bench code


module lab3_que3_tb;    // creating a module
  reg A,B,C,D,E;
  wire F;
  lab3_que3 a1(A,B,C,D,E,F);    // Instantiating by order as declared in module


  initial
    begin               // beginning execution

      // assigning values as input
      A=0;B=0;C=0;D=0;E=0;#1;
      $display("A=%b and B=%b and C=%b and D=%b and E=%b",A,B,C,D,E);
      $display("F=%b",F);

      A=0;B=0;C=0;D=0;E=1;#1;
      $display("A=%b and B=%b and C=%b and D=%b and E=%b",A,B,C,D,E);
      $display("F=%b",F);   //for display of values entered

     A=0;B=0;C=0;D=1;E=0;#1;
      $display("A=%b and B=%b and C=%b and D=%b and E=%b",A,B,C,D,E);
      $display("F=%b",F);

     A=0;B=0;C=0;D=1;E=1;#1;
      $display("A=%b and B=%b and C=%b and D=%b and E=%b",A,B,C,D,E);
      $display("F=%b",F);

      A=0;B=0;C=1;D=0;E=0;#1;
      $display("A=%b and B=%b and C=%b and D=%b and E=%b",A,B,C,D,E);
      $display("F=%b",F);

      end       // ending execution
  initial
    begin
      $dumpfile("dump.vcd");    // for storing information of waveform
      $dumpvars(1);         // specifying part of design to be stored as waveform
    end
endmodule       // enclosing the module

```

design.sv

```
// F(A, B, C, D, E) = ((A+B).(C+D+E))’

// Structural code

module lab3_que3(a,b,c,d,e,f);
input a,b,c,d,e;
output f;
wire p,q,r;

or(p,a,b);
or(q,c,d,e);

and(r,p,q);

not(f,r);

endmodule

```

Fig 3.2.1- Test bench and design code

Output:

```
A=0 and B=0 and C=0 and D=0 and E=0
F=1
A=0 and B=0 and C=0 and D=0 and E=1
F=1
A=0 and B=0 and C=0 and D=1 and E=0
F=1
A=0 and B=0 and C=0 and D=1 and E=1
F=1
A=0 and B=0 and C=1 and D=0 and E=0
F=1
```

Fig 3.2.2- output as truth table

Waveform:

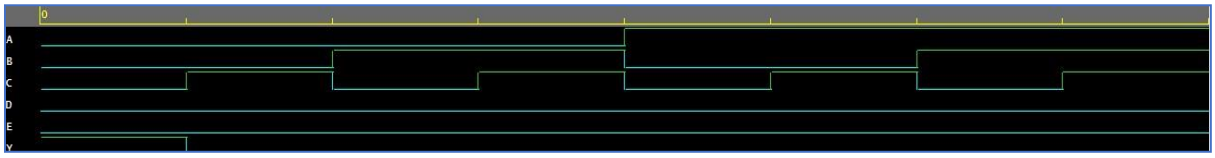


Fig 3.2.3- Waveform of output

Question 4: Implement the following expression using universal NAND and NOR gate. Write down Verilog Structural and Behavioral code for that expression.

4.1 F(A, B, C)=(AB'C) + (AB'C')

1. By NAND gate

Verilog Code:

testbench.sv

```
// Akshat Shah
// 21BCP322

// Test bench code


module lab3_que3_tb;    // creating a module
  reg A,B,C;
  wire F;
  lab3_que3 a1(A,B,C,F);    // Instantiating by order as declared in module


  initial
    begin               // beginning execution

      // assigning values as input
      A=0;B=0;C=0;#1;
      $display("A=%b and B=%b and C=%b",A,B,C);
      $display("F=%b",F);

      A=0;B=0;C=1;#1;
      $display("A=%b and B=%b and C=%b ",A,B,C);
      $display("F=%b",F);   //for display of values entered

     A=0;B=1;C=0;#1;
      $display("A=%b and B=%b and C=%b ",A,B,C);
      $display("F=%b",F);

     A=0;B=1;C=1;#1;
      $display("A=%b and B=%b and C=%b ",A,B,C);
      $display("F=%b",F);


      end       // ending execution
  initial
    begin
      $dumpfile("dump.vcd");    // for storing information of waveform
      $dumpvars(1);         // specifying part of design to be stored as waveform
    end
endmodule       // enclosing the module
```

design.sv

```
// F(A, B, C)=(AB'C) + (AB'C')
//By NAND gate

// Structural code

module lab3_que3(a,b,c,f);
input a,b,c;
output f;
wire b_not,c_not,p,q;
  nand(b_not,b);
  nand(c_not,c);
  nand(p,a,b_not,c);
  nand(q,a,b_not,c_not);
  nand(f,p,q);
endmodule


// behavioral code
/*
    module lab3_que3(a,b,c,f);
    input a,b,c;
    output f;
    assign f = ~((~(a&&(~(b&&b))&&(~(a&&(~(b&&b))&&(~(c&&c)))))));
    endmodule

 */

```

Fig 4.1.1- Test bench and design code

2. By NOR gate:

Verilog Code:

testbench.sv

```
// Akshat Shah
// 21BCP322

// Test bench code


module lab3_que3_tb;    // creating a module
  reg A,B,C;
  wire F;
  lab3_que3 a1(A,B,C,F);    // Instantiating by order as declared in module


  initial
    begin               // beginning execution

      // assigning values as input
      A=0;B=0;C=0;#1;
      $display("A=%b and B=%b and C=%b",A,B,C);
      $display("F=%b",F);

      A=0;B=0;C=1;#1;
      $display("A=%b and B=%b and C=%b ",A,B,C);
      $display("F=%b",F);   //for display of values entered

     A=0;B=1;C=0;#1;
      $display("A=%b and B=%b and C=%b ",A,B,C);
      $display("F=%b",F);

     A=0;B=1;C=1;#1;
      $display("A=%b and B=%b and C=%b ",A,B,C);
      $display("F=%b",F);


      end       // ending execution
  initial
    begin
      $dumpfile("dump.vcd");    // for storing information of waveform
      $dumpvars(1);          // specifying part of design to be stored as waveform
    end
endmodule       // enclosing the module
```

design.sv

```
// F(A, B, C)=(AB'C) + (AB'C')
//By NOR gate

// Structural code

module lab3_que3(a,b,c,f);
input a,b,c;
output f;
wire a_not,b_not,c_not,p,q,r,s;
  nor(a_not,a);
  nor(b_not,b);
  nor(c_not,c);
  nor(p,a_not,b,c_not);
  nor(q,a_not,b,c);
  nor(r,p,q);
  nor(f,r);
endmodule


// behavioral code
/*
    module lab3_que3(a,b,c,f);
    input a,b,c;
    output f;
    assign f = ~(~(~(~(a||a))||(b)||(~(~(a||a))||(c))));
    endmodule

 */
```

Fig 4.1.2- Test bench and design code

Output:

```
A=0 and B=0 and C=0
F=0
A=0 and B=0 and C=1
F=0
A=0 and B=1 and C=0
F=0
A=0 and B=1 and C=1
F=0
```

Fig 4.1.3- Output as truth table

Waveform:

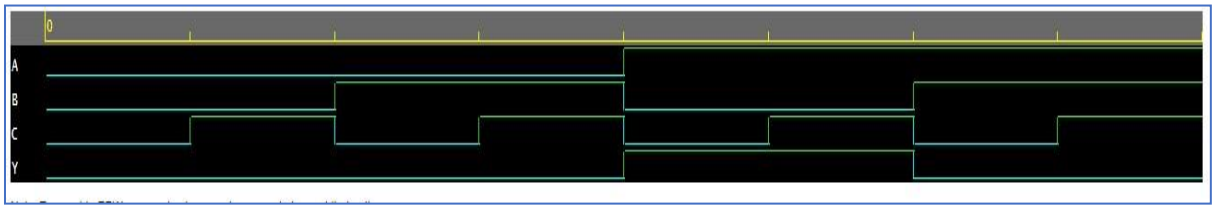


Fig 4.1.4- Waveform of output

4.2 F(A, B)= A'B' + AB' + A'B + AB

1. By NOR gate

Verilog Code:

testbench.sv

```
// Akshat Shah
// 21BCP322

// Test bench code


module lab3_que3_tb;    // creating a module
  reg A,B;
  wire F;
  lab3_que3 a1(A,B,F);  // Instantiating by order as declared in module


  initial
    begin               // beginning execution

      // assigning values as input
      A=0;B=0;#1;
      $display("A=%b and B=%b",A,B);
      $display("F=%b",F);

      A=0;B=0;#1;
      $display("A=%b and B=%b ",A,B);
      $display("F=%b",F);   //for display of values entered

     A=0;B=1;#1;
      $display("A=%b and B=%b  ",A,B);
      $display("F=%b",F);

     A=0;B=1;#1;
      $display("A=%b and B=%b  ",A,B);
      $display("F=%b",F);


      end       // ending execution
  initial
    begin
      $dumpfile("dump.vcd");    // for storing information of waveform
      $dumpvars(1);         // specifying part of design to be stored as waveform
    end
endmodule       // enclosing the module
```

design.sv

```
// F(A, B)= A'B' + AB' + A'B + AB
//By NOR gate

// Structural code

module lab3_que3(a,b,f);
input a,b;
output f;
wire a_not,b_not,p,q,r,s,t;
  nor(a_not,a);
  nor(b_not,b);
  nor(p,a,b);
  nor(q,a_not,b);
  nor(r,a,b_not);
  nor(s,a_not,b_not);
  nor(t,p,q,r,s);
  nor(f,t);
endmodule


// behavioral code
/*
    module lab3_que3(a,b,f);
    input a,b;
    output f;
    assign f = ~(~((~(~a|~b))|(~(a|~b))|(~(~a | b)) | (~(a|b))));
    endmodule

 */
```

Fig 4.2.1- Test bench and design code

2. By NAND gate:

Verilog code:

testbench.sv

```
// Akshat Shah
// 21BCP322

// Test bench code


module lab3_que4_tb;    // creating a module
  reg A,B;
  wire F;
  lab3_que4 a1(A,B,F);  // Instantiating by order as declared in module


  initial
    begin               // beginning execution

      // assigning values as input
      A=0;B=0;#1;
      $display("A=%b and B=%b",A,B);
      $display("F=%b",F);

      A=0;B=1;#1;
      $display("A=%b and B=%b ",A,B);
      $display("F=%b",F);   //for display of values entered

     A=1;B=0;#1;
      $display("A=%b and B=%b  ",A,B);
      $display("F=%b",F);

     A=1;B=1;#1;
      $display("A=%b and B=%b  ",A,B);
      $display("F=%b",F);


      end       // ending execution
  initial
    begin
      $dumpfile("dump.vcd");    // for storing information of waveform
      $dumpvars(1);         // specifying part of design to be stored as waveform
    end
endmodule       // enclosing the module
```

design.sv

```
// F(A,B) = A'B' + AB' + A'B + AB
//Structural Code

module lab3_que4(a,b,f);

  input a,b;
  output f;
  wire p,q,r,s;
  nand(a_not,a);
  nand(b_not,b);
  nand(p,a_not,b_not);
  nand(q,a,b_not);
  nand(r,a_not,b);
  nand(s,a,b);
  nand(f,p,q,r,s);
endmodule

/*
    Behavioral Code

    module lab3_que4(a,b,f);

  input a,b;
  output f;

  assign f= ~((~(~a&~b)) & (~(a&~b)) & (~(~a&b)) & (~(a&b)));

  */
```

Output:

```
A=0 and B=0
F=1
A=0 and B=0
F=1
A=0 and B=1
F=1
A=0 and B=1
F=1
```

Fig 4.2.2 Output as truth table

Waveform:

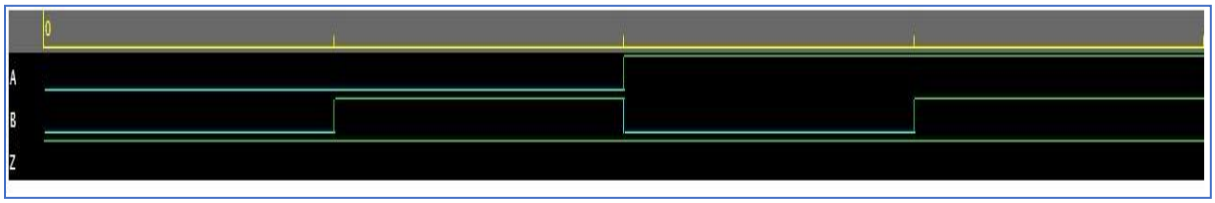


Fig 4.2.3- Waveform of output

Question 5: 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.

Verilog Code:

testbench.sv

```
// Akshat Shah
// 21BCP322

// Test bench code


module lab3_que5_tb;    // creating a module
  reg A,B;
  wire F;
  lab3_que5 a1(A,B,F);  // Instantiating by order as declared in module


  initial
    begin               // beginning execution

      // assigning values as input
      A=0;B=0;#1;
      $display("A=%b and B=%b",A,B);
      $display("F=%b",F);

      A=0;B=1;#1;
      $display("A=%b and B=%b ",A,B);
      $display("F=%b",F);   //for display of values entered

     A=1;B=0;#1;
      $display("A=%b and B=%b  ",A,B);
      $display("F=%b",F);

     A=1;B=1;#1;
      $display("A=%b and B=%b  ",A,B);
      $display("F=%b",F);


      end       // ending execution
  initial
    begin
      $dumpfile("dump.vcd");    // for storing information of waveform
      $dumpvars(1);         // specifying part of design to be stored as waveform
    end
endmodule       // enclosing the module
```

design.sv

```
// F(A,B) = A'B' + AB' + A'B + AB
//Structural Code

// AND GATE
module and_gate(output c, input a,b);
  assign c= a&b;

    endmodule

// OR GATE
module or_gate(output z, input x,y);
      assign z=x|y;
    endmodule

// NOT GATE
module not_gate(output h, input i);
  assign i = (~h);
endmodule



// XOR GATE using above 3 gates
module lab3_que5(p,q,f);
  input p,q;
  output f;
  wire a_not,b_not,v,w;
  not_gate n1(.i(a_not),.h(p));
  not_gate n2(.i(b_not),.h(q));
              and_gate a1(.c(v),.a(a_not),.b(q));
              and_gate a2(.c(w),.a(p),.b(b_not));
              or_gate o1(.z(f),.x(v),.y(w));
endmodule
```

Fig 5.1- Test bench and design code

Output:

```
A=0 and B=0
F=0
A=0 and B=1
F=1
A=1 and B=0
F=1
A=1 and B=1
F=0
```

Fig 5.2- Output as truth table

Waveform:

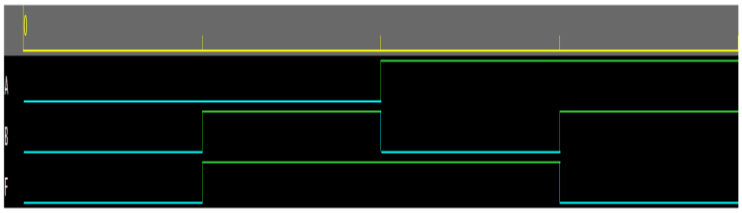


Fig 5.3- Waveform of output

Question 6: 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 Test Bench code for the verification purpose.

(i) Truth table:

A	B	C	A'B'C'	A'BC'	AB'C'	AB'C	Y
0	0	0	1	0	0	0	1
0	0	1	0	0	0	0	0
0	1	0	0	1	0	0	1
0	1	1	0	0	1	0	1
1	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0
1	1	0	0	0	0	0	0
1	1	1	0	0	0	0	0

(ii) Verilog Code:

SV/Verilog Testbench

```
// Akshat Shah
// 21BCP322

// test bench code for XOR gate

module lab3_que3_tb;    // creating a module
  reg A,B,C;
  wire F;
  lab3_que3 a1(A,B,C,F);    // Instantiating by order as declared in module


  initial
    begin               // beginning execution

      // assigning values as input
      A=0;B=0;C=0;#1;
      $display("A=%b and B=%b and C=%b",A,B,C);
      $display("F=%b",F);

      A=0;B=0;C=1;#1;
      $display("A=%b and B=%b and C=%b",A,B,C); // for display of values entered
      $display("F=%b",F);

     A=0;B=1;C=0;#1;
      $display("A=%b and B=%band C=%b",A,B,C);
      $display("F=%b",F);

     A=;B=1;C=1;#1;
      $display("A=%b and B=%b and C=%b",A,B,C);
      $display("F=%b",F);



    end     // ending execution
  initial
    begin
      $dumpfile("dump.vcd");    // for storing information of waveform
      $dumpvars(1);         // specifying part of design to be stored as waveform
    end
endmodule       // enclosing the module
```

```
// Y = A'.B'.C'+ A'.B.C'+ A.B'.C'+ A.B'.C

// 3 input AND gate
module and_gate(output d,input a,b,c);
  assign d = a&b&c;
endmodule

// 4 input OR gate
module or_gate(output z,input p,q,r,s);
  assign z = p|q|r|s;
endmodule


// instantiating given expression
module lab3_que3( input u,v,w ,output o);
  wire a_not,b_not,c_not,x,y,d,e;
  not(a_not,u);
  not(b_not,v);
  not(c_not,w);
  and_gate a1(.d(x),.a(a_not),.b(b_not),.c(c_not));
  and_gate a2(.d(y),.a(a_not),.b(v),.c(c_not));
  and_gate a3(.d(d),.a(u),.b(b_not),.c(c_not));
  and_gate a4(.d(e),.a(u),.b(b_not),.c(w));
  or_gate o1(.z(o),.p(x),.q(y),.r(d),.s(e));
endmodule
```

Fig 6.1- Test bench and design code

Output:

```
a=0, b=0, c=0
y=1
a=0, b=0, c=1
y=0
a=0, b=1, c=0
y=1
a=0, b=1, c=1
y=0
```

Fig 6.2- Output as truth table

(iii) Minimal Expression: Y = A'C' + AB'

(iv) Verilog Code:

```
// Akshat Shah
// 21BCP322

// test bench code for XOR gate

module lab3_que6_tb;    // creating a module
  reg A,B,C;
  wire F;
  lab3_que6 a1(A,B,C,F);    // Instantiating by order as declared in module


  initial
    begin               // beginning execution

      // assigning values as input
      A=0;B=0;C=0;#1;
      $display("A=%b and B=%b and C=%b",A,B,C);
      $display("F=%b",F);

      A=0;B=0;C=1;#1;
      $display("A=%b and B=%b and C=%b",A,B,C); // for display of values entered
      $display("F=%b",F);

     A=0;B=1;C=0;#1;
      $display("A=%b and B=%band C=%b",A,B,C);
      $display("F=%b",F);

     A=0;B=1;C=1;#1;
      $display("A=%b and B=%b and C=%b",A,B,C);
      $display("F=%b",F);



    end     // ending execution
  initial
    begin
      $dumpfile("dump.vcd");    // for storing information of waveform
      $dumpvars(1);         // specifying part of design to be stored as waveform
    end
endmodule       // enclosing the module
```

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

module or_gate(input x,y, output z);
  assign z= x|y;
endmodule

module not_gate(input h, output i);
  assign i = ~h;
endmodule

module lab3_que6(u,v,w,o);
input u,v,w;
output o;

  wire u_not,v_not,w_not,p,q;
  not_gate n1(u,u_not);
  not_gate n2(v,v_not);
  not_gate n3(w,w_not);

  and_gate a1(u,v_not,q);
  and_gate a2(u_not,w_not,p);
  or_gate o1(q,p,o);
endmodule
```

Fig 6.3- Test bench and design code

Output:

```
A=0 and B=0 and C=0
F=1
A=0 and B=0 and C=1
F=0
A=0 and B=1and C=0
F=1
A=0 and B=1 and C=1
F=0
```

Fig 6.4- Output as truth table

Waveform:

Fig 6.5- Waveform of output