

Lab 3: Implementation of Structural & Behavioral Verilog Code

Question 1: Implement the following expression using the Verilog HDL. Moreover, Verify your circuit against the waveform.

a. $F(A, B, C) = A'BC + AB'C + ABC$

testbench sv

```

1 module tb_exp;
2   reg K,L,M;
3   wire N;
4   build_exp x1 (.k(K), .l(L), .m(M), .n(N));
5   initial
6   begin
7     K = 1; L = 1; M = 1; #5;
8     K = 1; L = 1; M = 0; #5;
9     K = 1; L = 0; M = 1; #5;
10    K = 1; L = 0; M = 0; #5;
11    K = 0; L = 1; M = 1; #5;
12    K = 0; L = 1; M = 0; #5;
13    K = 0; L = 0; M = 1; #5;
14    K = 0; L = 0; M = 0; #5;
15  end
16  initial
17  begin
18    $dumpfile("dump.vcd");
19    $dumpvars(1);
20  end
21 endmodule

```

design sv

```

1 module and_gate(input a,b,c, output d);
2   assign d = a&b&c;
3 endmodule
4
5 module not_gate(input e, output f);
6   assign f = ~e;
7 endmodule
8
9 module or_gate(input p,q,r, output s);
10  assign t = p|q|r|s;
11 endmodule
12
13 module build_exp(input k,l,m, output n);
14   wire w,x,y, k_not,l_not,m_not;
15
16   not_gate n1 (.e(k), .f(k_not));
17   not_gate n2 (.e(l), .f(l_not));
18   not_gate n3 (.e(m), .f(m_not));
19
20   and_gate a1 (.a(k_not), .b(l), .c(m), .d(w));
21   and_gate a2 (.a(k), .b(l_not), .c(m), .d(x));
22   and_gate a3 (.a(k), .b(l), .c(m), .d(y));
23
24   or_gate o1 (.p(w), .q(x), .r(y), .s(z));
25 endmodule

```

b. $F(A, B, C, D) = ABCD' + A'BCD + AB'CD' + ABC$

testbench sv

```

1 module tb_exp;
2   reg K,L,M,N;
3   wire O;
4   build_exp x1 (.k(K), .l(L), .m(M), .n(N), .o(O));
5   initial
6   begin
7     K = 1; L = 1; M = 1; N = 1; #5;
8     K = 1; L = 1; M = 1; N = 0; #5;
9     K = 1; L = 1; M = 0; N = 1; #5;
10    K = 1; L = 1; M = 0; N = 0; #5;
11    K = 1; L = 0; M = 1; N = 1; #5;
12    K = 1; L = 0; M = 1; N = 0; #5;
13    K = 1; L = 0; M = 0; N = 1; #5;
14    K = 1; L = 0; M = 0; N = 0; #5;
15    K = 0; L = 1; M = 1; N = 1; #5;
16    K = 0; L = 1; M = 1; N = 0; #5;
17    K = 0; L = 1; M = 0; N = 1; #5;
18    K = 0; L = 1; M = 0; N = 0; #5;
19    K = 0; L = 0; M = 1; N = 1; #5;
20    K = 0; L = 0; M = 1; N = 0; #5;
21    K = 0; L = 0; M = 0; N = 1; #5;
22    K = 0; L = 0; M = 0; N = 0; #5;
23  end
24  initial
25  begin
26    $dumpfile("dump.vcd");
27    $dumpvars(1);
28  end
29 endmodule

```

design sv

```

1 module and_gate1(input g,h,i, output j);
2   assign j = g&h&i;
3 endmodule
4
5 module and_gate2(input a,b,c,d, output e);
6   assign e = a&b&c&d;
7 endmodule
8
9 module not_gate(input e, output f);
10  assign f = ~e;
11 endmodule
12
13 module or_gate(input p,q,r,s, output t);
14  assign t = p|q|r|s;
15 endmodule
16
17 module build_exp(input k,l,m,n, output o);
18   wire w,x,y,z,out, k_not,l_not,m_not,n_not;
19
20   not_gate n1 (.e(k), .f(k_not));
21   not_gate n2 (.e(l), .f(l_not));
22   not_gate n3 (.e(m), .f(m_not));
23   not_gate n4 (.e(n), .f(n_not));
24
25   and_gate2 a1 (.a(k), .b(l), .c(m), .d(n_not), .e(w));
26   and_gate2 a2 (.a(k), .b(l_not), .c(m), .d(n), .e(x));
27   and_gate2 a3 (.a(k), .b(l_not), .c(m), .d(n_not), .e(y));
28   and_gate1 a4 (.g(k), .h(l), .i(m), .j(z));
29
30   or_gate o1 (.p(w), .q(x), .r(y), .s(z), .t(out));
31 endmodule

```

c. $F(A, B) = (A' + B') (A + B') (A' + B) (A + B)$

testbench.sv

```

1 // Code your testbench here
2 // or browse Examples
3 module tbandgate;
4   reg A, B;
5   wire Y;
6   andgate a1 (.a(A),.b(B),.y(Y));
7   initial
8     begin
9       A=0;B=0;#5;
10      A=0;B=1;#5;
11      A=1;B=0;#5;
12      A=1;B=1;#5;
13    end
14  initial
15    begin
16      $dumpfile("dump.vcd");
17      $dumpvars(1);
18    end
19 endmodule
20

```

design.sv

```

1 // Code your design here
2 module andgate(
3   input a,b,
4   output y);
5   assign y = (~a | ~b) & (a | ~b) & (~a | b) & (a | b);
6 endmodule

```

d. $(A, B) = ((A.B')' + ((A)'(B)'))'$

testbench.sv

```

1 // Code your testbench here
2 // or browse Examples
3 module tbandgate;
4   reg A, B;
5   wire Y;
6   andgate a1 (.a(A),.b(B),.y(Y));
7   initial
8     begin
9       A=0;B=0;#5;
10      A=0;B=1;#5;
11      A=1;B=0;#5;
12      A=1;B=1;#5;
13    end
14  initial
15    begin
16      $dumpfile("dump.vcd");
17      $dumpvars(1);
18    end
19 endmodule
20

```

design.sv

```

1 // Code your design here
2 module andgate(
3   input a,b,
4   output y);
5   assign y = ~((a & ~b) | (~(~a & ~b)));
6 endmodule

```

e. $(A, B) = (((A)' + B)'. ((A)' + (B)')')$

testbench sv

```

1 // Code your testbench here
2 // or browse Examples
3 module tbandgate;
4   reg A, B;
5   wire Y;
6   andgate a1 (.a(A),.b(B),.y(Y));
7   initial
8   begin
9     A=0;B=0;#5;
10    A=0;B=1;#5;
11    A=1;B=0;#5;
12    A=1;B=1;#5;
13  end
14  initial
15  begin
16    $dumpfile("dump.vcd");
17    $dumpvars(1);
18  end
19 endmodule
20

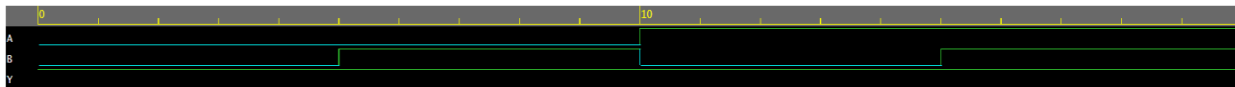
```

design sv

```

1 // Code your design here
2 module andgate(
3   input a,b,
4   output y);
5   assign y = ~(~((~a | b)) & ~(~a | ~b));
6 endmodule

```



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

design sv

```

1 // Behavioural Code
2 module not_gate (input e, output f);
3   assign f = ~e;
4 endmodule
5 module and_gate (input a, b, output c);
6   assign c = a & b;
7 endmodule
8 module or_gate (input p, q, output r);
9   assign r = p | q;
10 endmodule
11 module build_xor (input m, n, output o);
12   wire x, y, a_not, b_not;
13   not_gate n1 (.e(m),.f(a_not));
14   not_gate n2 (.e(n),.f(b_not));
15   and_gate a1 (.a(a_not),.b(n),.c(x));
16   and_gate a2 (.a(m),.b(b_not),.c(y));
17   or_gate o1 (.p(x),.q(y),.r(o));
18 endmodule
19
20 |
21 // Structural Code
22 module Lab4_Build_XOR(a, b, c);
23   input a,b;
24   output c;
25   wire a_not, b_not;
26   wire x,y;
27   not(a_not,a);
28   not(b_not,b);
29   and(x,a_not,b);
30   and(y,a,b_not);
31   or(c,x,y);
32 endmodule
33

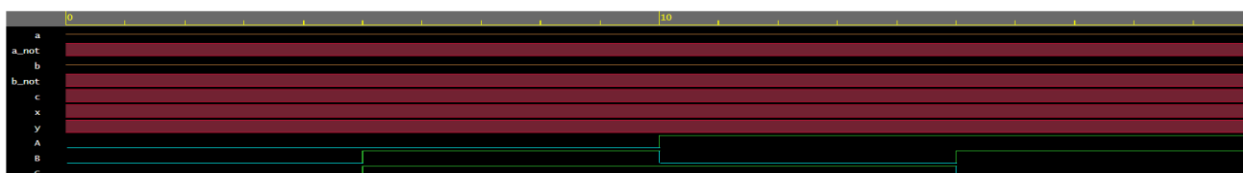
```

testbench sv

```

1 module tb_xor_str;
2   reg A,B;
3   wire C;
4   build_xor x1 (.m(A), .n(B), .o(C));
5   initial
6   begin
7     A = 0; B = 0; #5;
8     A = 0; B = 1; #5;
9     A = 1; B = 0; #5;
10    A = 1; B = 1; #5;
11  end
12  initial
13  begin
14    $dumpfile("dump.vcd");
15    $dumpvars(1);
16  end
17 endmodule

```



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

a. $F(A, B, C) = (A' + B' + C')(A + B' + C')(A' + B + C')(A' + B')$

testbench sv

```

1 // Code your testbench here
2 // or browse Examples
3 module tbxor;
4   reg A, B, C;
5   wire D;
6   buildxor x1 (.m(A), .n(B), .o(C), .t(D));
7   initial
8   begin
9     A = 0; B = 0; C = 0; #5;
10    A = 0; B = 1; C = 0; #5;
11    A = 1; B = 0; C = 0; #5;
12    A = 1; B = 1; C = 0; #5;
13    A = 0; B = 0; C = 1; #5;
14    A = 0; B = 1; C = 1; #5;
15    A = 1; B = 0; C = 1; #5;
16    A = 1; B = 1; C = 1; #5;
17  end
18  initial
19  begin
20    $dumpfile("dump.vcd");
21    $dumpvars(1);
22  end
23 endmodule
24

```

design sv

```

1 // Code your design here
2 module notgate (input e, output f);
3   assign f = ~e;
4 endmodule
5
6 module andgate (input a, b, c, output d);
7   assign d = a & b & c;
8 endmodule
9
10 module orgate (input p, q, output r);
11   assign r = p | q;
12 endmodule
13
14 module buildxor (input m, n, o, output t);
15   wire w, x, anot, bnot;
16   notgate n1 (.e(n), .f(bnot));
17   notgate n2 (.e(o), .f(cnot));
18   andgate a1 (.a(m), .b(bnot), .c(o), .d(w));
19   andgate a2 (.a(m), .b(bnot), .c(cnot), .d(x));
20   orgate o1 (.p(x), .q(w), .r(t));
21 endmodule
22

```

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

a. $F(A, B, C) = (AB'C) + (AB'C')$

testbench sv

```

1 module tb_exp;
2   reg K,L,M;
3   wire N;
4   build_exp x1 (.k(K), .l(L), .m(M), .n(N));
5   initial
6   begin
7     K = 1; L = 1; M = 1; #5;
8     K = 1; L = 1; M = 1; #5;
9     K = 1; L = 1; M = 0; #5;
10    K = 1; L = 1; M = 0; #5;
11    K = 1; L = 0; M = 1; #5;
12    K = 1; L = 0; M = 1; #5;
13    K = 1; L = 0; M = 0; #5;
14    K = 1; L = 0; M = 0; #5;
15    K = 0; L = 1; M = 1; #5;
16    K = 0; L = 1; M = 0; #5;
17    K = 0; L = 1; M = 0; #5;
18    K = 0; L = 1; M = 0; #5;
19    K = 0; L = 0; M = 1; #5;
20    K = 0; L = 0; M = 1; #5;
21    K = 0; L = 0; M = 0; #5;
22    K = 0; L = 0; M = 0; #5;
23  end
24  initial
25  begin
26    $dumpfile("dump.vcd");
27    $dumpvars(1);
28  end
29 endmodule

```

design sv

```

1 module build_exp(k,l,m,n);
2   input k,l,m;
3   output n;
4   wire w,x,y,z;
5   wire k_not,l_not,m_not;
6
7   not (k, k_not);
8   not (l, l_not);
9   not (m, m_not);
10
11   or (k_not, l_not, m_not, w);
12   or (k, l_not, m_not, x);
13   or (k_not, l, m_not, y);
14   or (k_not, l_not, l_not, z);
15
16   not (z, z_not);
17   and (w, x, y, z_not, out);
18
19 endmodule

```

b. $F(A, B) = A'B' + AB' + A'B + AB$

testbench.sv

```

1 // Code your testbench here
2 // or browse Examples
3 module tbxor;
4   reg A, B;
5   wire C;
6   buildxor x1 (.m(A), .n(B), .o(C));
7   initial
8   begin
9     A = 0; B = 0; #5;
10    A = 0; B = 1; #5;
11    A = 1; B = 0; #5;
12    A = 1; B = 1; #5;
13  end
14  initial
15  begin
16    $dumpfile("dump.vcd");
17    $dumpvars(1);
18  end
19 endmodule
20

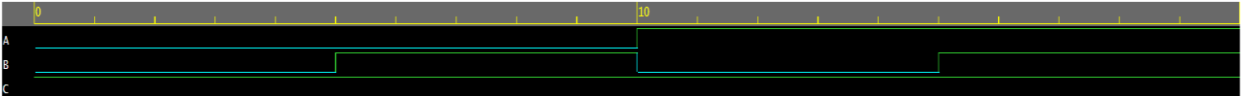
```

design.sv

```

1 // Code your design here
2 module notgate (input e, output f);
3   assign f = ~e;
4 endmodule
5
6 module andgate (input a, b, output c);
7   assign c = a & b;
8 endmodule
9
10 module orgate (input p, q, r, s, output t);
11   assign t = p | q | r | s;
12 endmodule
13
14 module buildxor (input m, n, output o);
15   wire w, x, y, z, anot, bnot;
16   notgate n1 (.e(m), .f(anot));
17   notgate n2 (.e(n), .f(bnot));
18   andgate a1 (.a(anot), .b(bnot), .c(x));
19   andgate a2 (.a(m), .b(bnot), .c(y));
20   andgate a3 (.a(anot), .b(n), .c(z));
21   andgate a4 (.a(m), .b(n), .c(w));
22   orgate o1 (.p(x), .q(y), .r(z), .s(w), .t(o));
23 endmodule
24

```



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.

testbench.sv

```

1 module tb_xor_str;
2   reg A,B;
3   wire C;
4   build_xor x1 (.m(A), .n(B), .o(C));
5   initial
6   begin
7     A = 0; B = 0; #5;
8     A = 0; B = 1; #5;
9     A = 1; B = 0; #5;
10    A = 1; B = 1; #5;
11  end
12  initial
13  begin
14    $dumpfile("dump.vcd");
15    $dumpvars(1);
16  end
17 endmodule

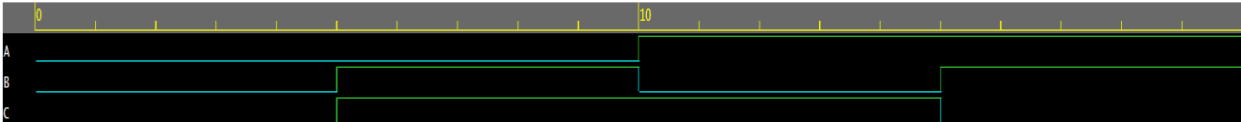
```

design.sv

```

1 module and_gate(input a, b, output c);
2   assign c = a&b;
3 endmodule
4
5 module not_gate(input e, output f);
6   assign f = ~e;
7 endmodule
8
9 module or_gate(input p,q, output r);
10  assign r = p|q;
11 endmodule
12
13 module build_xor(input m, n, output o);
14   wire x, y, a_not, b_not;
15   not_gate n1 (.e(m),.f(a_not));
16   not_gate n2 (.e(n),.f(b_not));
17   and_gate a1 (.a(a_not),.b(n),.c(x));
18   and_gate a2 (.a(m),.b(b_not),.c(y));
19   or_gate o1 (.p(x),.q(y),.r(o));
20 endmodule

```



Question 5 Consider the following expression: $Y = A'.B'.C' + A'.B.C' + A.B'.C' + A.B'.C$

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.

A	B	C	Y
F	F	F	T
F	F	T	F
F	T	F	T
F	T	T	F
T	F	F	T
T	F	T	T
T	T	F	F
T	T	T	F

Reduced Expression: $A'.B'.C'$

testbench.sv

```

1 module tb_exp;
2   reg K,L,M;
3   wire N;
4   build_exp x1 (.k(K), .l(L), .m(M), .n(N));
5   initial
6     begin
7       K = 1; L = 1; M = 1; #5;
8       K = 1; L = 1; M = 0; #5;
9       K = 1; L = 0; M = 1; #5;
10      K = 1; L = 0; M = 0; #5;
11      K = 0; L = 1; M = 1; #5;
12      K = 0; L = 1; M = 0; #5;
13      K = 0; L = 0; M = 1; #5;
14      K = 0; L = 0; M = 0; #5;
15    end
16    initial
17      begin
18        $dumpfile("dump.vcd");
19        $dumpvars(1);
20      end
21  endmodule

```

design.sv

```

1 module and_gate(input a,b,c , output d);
2   assign d = a&b&c;
3 endmodule
4
5 module not_gate(input e, output f);
6   assign f = ~e;
7 endmodule
8
9 module or_gate(input p,q,r,s, output t);
10  assign t = p|q|r|s;
11 endmodule
12
13 module build_exp(input k,l,m, output n);
14   wire w,x,y,z, k_not,l_not,m_not;
15
16   not_gate n1 (.e(k), .f(k_not));
17   not_gate n2 (.e(l), .f(l_not));
18   not_gate n3 (.e(m), .f(m_not));
19
20   and_gate a1 (.a(k_not), .b(l_not), .c(m_not), .d(w));
21   and_gate a2 (.a(k_not), .b(l), .c(m_not), .d(x));
22   and_gate a3 (.a(k), .b(l_not), .c(m_not), .d(y));
23   and_gate a4 (.a(k), .b(l_not), .c(m), .d(z));
24
25   or_gate o1 (.p(w), .q(x), .r(y), .s(z), .t(n));
26 endmodule

```

