EEE 2243
Digital System Design
Semester II 2011/12

Tutorial 1

Verilog Combinational

1. Write the Verilog code for the following 2-input Boolean gates:

   a. AND gate

```
module ANDGate(A, B, F);
input A, B;
output F;

assign F = A & B;

endmodule
```

   b. OR gate

```
module ORGate(A, B, F);
input A, B;
output F;

assign F = A | B;

endmodule
```

   c. NOR gate

```
module NORGate(A, B, F);
input A, B;
output F;

assign F = ~(A | B);

endmodule
```

   d. NAND gate

```
module NANDGate(A, B, F);
input A, B;
output F;

assign F = ~(A & B);

endmodule
```

e. XOR gate

```
module XORGate(A, B, F);
input A, B;
output F;

assign F = A ^ B;

endmodule
```

f. XNOR gate

```
module XNORGate(A, B, F);
input A, B;
output F;

assign F = ~(A ^ B);

endmodule
```

2. Write the Verilog code for the Boolean gates as in no. 1, but with 8 inputs, and forced with the following 8 bit inputs:

a. 2 dec

```
module ANDGate(F);
output F;
wire [7:0] inp;

assign inp = 8'd2;

assign F = inp[7] & inp[6] & inp[5] & inp[4] &
           inp[3] & inp[2] & inp[1] & inp[0];

endmodule
```

```
module ORGate(F);
output F;
wire [7:0] inp;

assign inp = 8'd2;

assign F = inp[7] | inp[6] | inp[5] | inp[4] |
           inp[3] | inp[2] | inp[1] | inp[0];

endmodule
```

```
module NORGate(F);
output F;
wire [7:0] inp;

assign inp = 8'd2;

assign F = ~(inp[7] | inp[6] | inp[5] | inp[4] |
             inp[3] | inp[2] | inp[1] | inp[0]);

endmodule
```

```
module NANDGate(F);
output F;
wire [7:0] inp;

assign inp = 8'd2;

assign F = ~(inp[7] & inp[6] & inp[5] & inp[4] &
             inp[3] & inp[2] & inp[1] & inp[0]);

endmodule
```

```
module XORGate(F);
output F;
wire [7:0] inp;

assign inp = 8'd2;

assign F = inp[7] ^ inp[6] ^ inp[5] ^ inp[4] ^
           inp[3] ^ inp[2] ^ inp[1] ^ inp[0];

endmodule
```

```
module XNORGate(F);
output F;
wire [7:0] inp;

assign inp = 8'd2;

assign F = ~(inp[7] ^ inp[6] ^ inp[5] ^ inp[4] ^
             inp[3] ^ inp[2] ^ inp[1] ^ inp[0]);

endmodule
```

b. 170 dec

```
module ANDGate(F);
output F;
wire [7:0] inp;

assign inp = 8'd170;

assign F = inp[7] & inp[6] & inp[5] & inp[4] &
           inp[3] & inp[2] & inp[1] & inp[0];

endmodule
```

```
module ORGate(F);
output F;
wire [7:0] inp;

assign inp = 8'd170;

assign F = inp[7] | inp[6] | inp[5] | inp[4] |
           inp[3] | inp[2] | inp[1] | inp[0];

endmodule
```

```
module NORGate(F);
output F;
wire [7:0] inp;

assign inp = 8'd170;

assign F = ~(inp[7] | inp[6] | inp[5] | inp[4] |
             inp[3] | inp[2] | inp[1] | inp[0]);

endmodule
```

```
module NANDGate(F);
output F;
wire [7:0] inp;

assign inp = 8'd170;

assign F = ~(inp[7] & inp[6] & inp[5] & inp[4] &
             inp[3] & inp[2] & inp[1] & inp[0]);

endmodule
```

```
module XORGate(F);
output F;
wire [7:0] inp;

assign inp = 8'd170;

assign F = inp[7] ^ inp[6] ^ inp[5] ^ inp[4] ^
           inp[3] ^ inp[2] ^ inp[1] ^ inp[0];

endmodule
```

```
module XNORGate(F);
output F;
wire [7:0] inp;

assign inp = 8'd170;

assign F = ~(inp[7] ^ inp[6] ^ inp[5] ^ inp[4] ^
             inp[3] ^ inp[2] ^ inp[1] ^ inp[0]);

endmodule
```

c. F0 hex

```
module ANDGate(F);
output F;
wire [7:0] inp;

assign inp = 8'hF0;

assign F = inp[7] & inp[6] & inp[5] & inp[4] &
           inp[3] & inp[2] & inp[1] & inp[0];

endmodule
```

```
module ORGate(F);
output F;
wire [7:0] inp;

assign inp = 8'hF0;

assign F = inp[7] | inp[6] | inp[5] | inp[4] |
           inp[3] | inp[2] | inp[1] | inp[0];

endmodule
```

```
module NORGate(F);
output F;
wire [7:0] inp;

assign inp = 8'hF0;

assign F = ~(inp[7] | inp[6] | inp[5] | inp[4] |
             inp[3] | inp[2] | inp[1] | inp[0]);

endmodule
```

```
module NANDGate(F);
output F;
wire [7:0] inp;

assign inp = 8'hF0;

assign F = ~(inp[7] & inp[6] & inp[5] & inp[4] &
             inp[3] & inp[2] & inp[1] & inp[0]);

endmodule
```

```
module XORGate(F);
output F;
wire [7:0] inp;

assign inp = 8'hF0;

assign F = inp[7] ^ inp[6] ^ inp[5] ^ inp[4] ^
           inp[3] ^ inp[2] ^ inp[1] ^ inp[0];

endmodule
```

```
module XNORGate(F);
output F;
wire [7:0] inp;

assign inp = 8'hF0;

assign F = ~(inp[7] ^ inp[6] ^ inp[5] ^ inp[4] ^
             inp[3] ^ inp[2] ^ inp[1] ^ inp[0]);

endmodule
```

d. 55 hex

```
module ANDGate(F);
output F;
wire [7:0] inp;

assign inp = 8'h55;

assign F = inp[7] & inp[6] & inp[5] & inp[4] &
           inp[3] & inp[2] & inp[1] & inp[0];

endmodule
```

```
module ORGate(F);
output F;
wire [7:0] inp;

assign inp = 8'h55;

assign F = inp[7] | inp[6] | inp[5] | inp[4] |
           inp[3] | inp[2] | inp[1] | inp[0];

endmodule
```

```
module NORGate(F);
output F;
wire [7:0] inp;

assign inp = 8'h55;

assign F = ~(inp[7] | inp[6] | inp[5] | inp[4] |
             inp[3] | inp[2] | inp[1] | inp[0]);

endmodule
```

```
module NANDGate(F);
output F;
wire [7:0] inp;

assign inp = 8'h55;

assign F = ~(inp[7] & inp[6] & inp[5] & inp[4] &
             inp[3] & inp[2] & inp[1] & inp[0]);

endmodule
```

```
module XORGate(F);
output F;
wire [7:0] inp;

assign inp = 8'h55;

assign F = inp[7] ^ inp[6] ^ inp[5] ^ inp[4] ^
           inp[3] ^ inp[2] ^ inp[1] ^ inp[0];

endmodule
```

```
module XNORGate(F);
output F;
wire [7:0] inp;

assign inp = 8'h55;

assign F = ~(inp[7] ^ inp[6] ^ inp[5] ^ inp[4] ^
             inp[3] ^ inp[2] ^ inp[1] ^ inp[0]);

endmodule
```

Which gate(s) is (are) good to tell whether the input has even number of 1-s and 0-s?

[This information is graduate (master) level answer. It is okey if you don't know, but if you do, it will be really good.] If we simulate the above code with enough no. values of `inp`, we can tell that, if the no. inputs is even, the XOR gate gives 1 if the no. 1-s at the inputs is odd, otherwise 0 if the no. 1-s at the inputs is even. If the no. inputs is odd, XNOR and XOR are the same.

3. Write the Verilog code for the following Boolean functions WITHOUT minimization:

a. $f_1 = \sum \mathrm{m}(0,2,4,5,6,7,8,10,11,12,14,15)$

$0 \Rightarrow 0000_2$
$2 \Rightarrow 0010_2$
$4 \Rightarrow 0100_2$
$5 \Rightarrow 0101_2$
$6 \Rightarrow 0110_2$
$7 \Rightarrow 0111_2$
$8 \Rightarrow 1000_2$
$10 \Rightarrow 1010_2$
$11 \Rightarrow 1011_2$
$12 \Rightarrow 1100_2$
$14 \Rightarrow 1110_2$
$15 \Rightarrow 1111_2$

```
module Q3a(a, b, c, d, f1);

input a, b, c, d;
output f1;

assign f1 =   (~a & ~b & ~c & ~d)
            | (~a & ~b &  c & ~d)
            | (~a &  b & ~c & ~d)
            | (~a &  b & ~c &  d)
            | (~a &  b &  c & ~d)
            | (~a &  b &  c &  d)
            | ( a & ~b & ~c & ~d)
            | ( a & ~b &  c & ~d)
            | ( a & ~b &  c &  d)
            | ( a &  b & ~c & ~d)
            | ( a &  b &  c & ~d)
            | ( a &  b &  c &  d);

endmodule
```

b. $f_2 = \sum \mathrm{m}(1,3,4,5,10,12,13)$

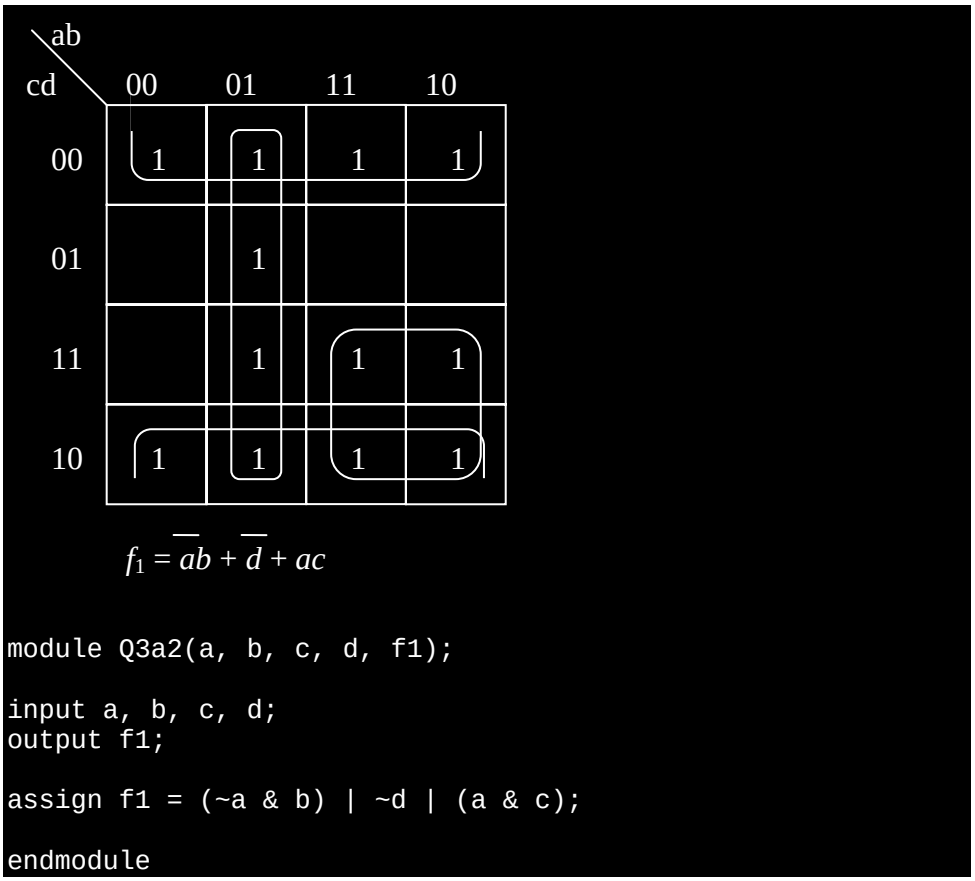
$1 \quad \Rightarrow \quad 0001_2$
$3 \quad \Rightarrow \quad 0011_2$
$4 \quad \Rightarrow \quad 0100_2$
$5 \quad \Rightarrow \quad 0101_2$
$10 \quad \Rightarrow \quad 1010_2$
$12 \quad \Rightarrow \quad 1100_2$
$13 \quad \Rightarrow \quad 1101_2$

```
module Q3b(a, b, c, d, f1);

input a, b, c, d;
output f1;

assign f1 =   (~a & ~b & ~c &  d)
            | (~a & ~b &  c &  d)
            | (~a &  b & ~c & ~d)
            | (~a &  b & ~c &  d)
            | ( a & ~b &  c & ~d)
            | ( a &  b & ~c & ~d)
            | ( a &  b & ~c &  d);

endmodule
```

c. $f_3 = \sum \mathrm{m}(0,2,3,5,6,7,8,10,11,14,15)$

```
0      ⇨     0000₂
2      ⇨     0010₂
3      ⇨     0011₂
5      ⇨     0101₂
6      ⇨     0110₂
7      ⇨     0111₂
8      ⇨     1000₂
10     ⇨     1010₂
11     ⇨     1011₂
14     ⇨     1110₂
15     ⇨     1111₂

module Q3c(a, b, c, d, f1);

input a, b, c, d;
output f1;

assign f1 =   (~a & ~b & ~c & ~d)
            | (~a & ~b &  c & ~d)
            | (~a & ~b &  c &  d)
            | (~a &  b & ~c &  d)
            | (~a &  b &  c & ~d)
            | (~a &  b &  c &  d)
            | ( a & ~b & ~c & ~d)
            | ( a & ~b &  c & ~d)
            | ( a & ~b &  c &  d)
            | ( a &  b &  c & ~d)
            | ( a &  b &  c &  d);

endmodule
```

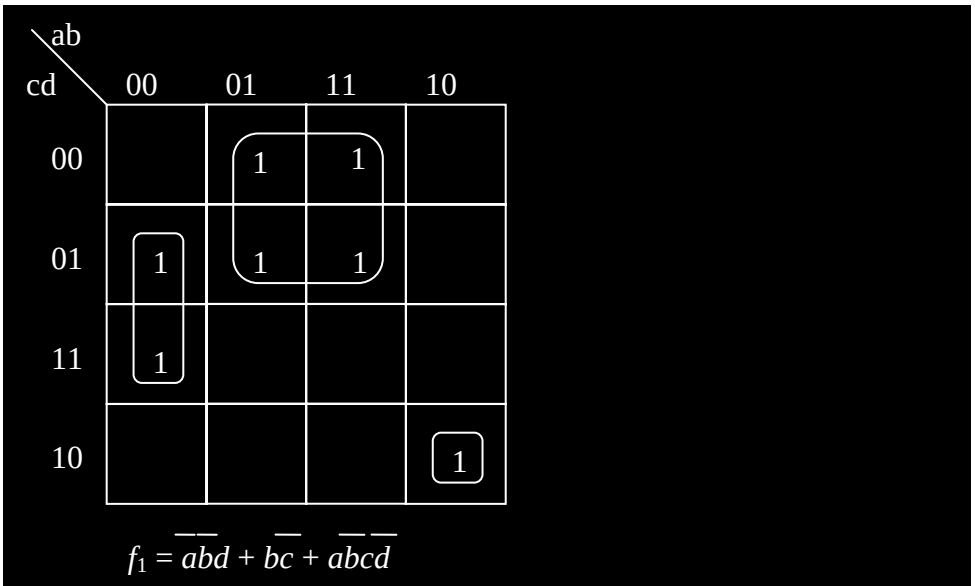
4. Redo number 3 using the minimized version of the functions, then compare the simulation – they should be identical.




```
module Q3a2(a, b, c, d, f1);

input a, b, c, d;
output f1;

assign f1 = (~a & b) | ~d | (a & c);

endmodule
```

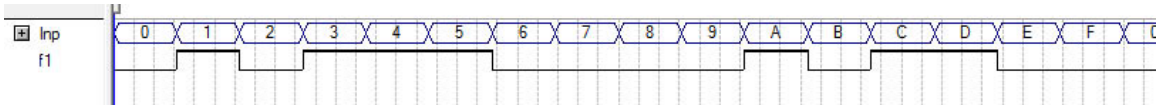
Simulation of unminimized Q3a:

Simulation of minimized Q3a:

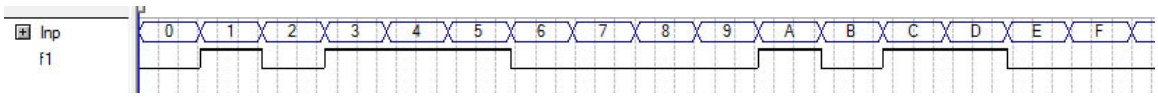


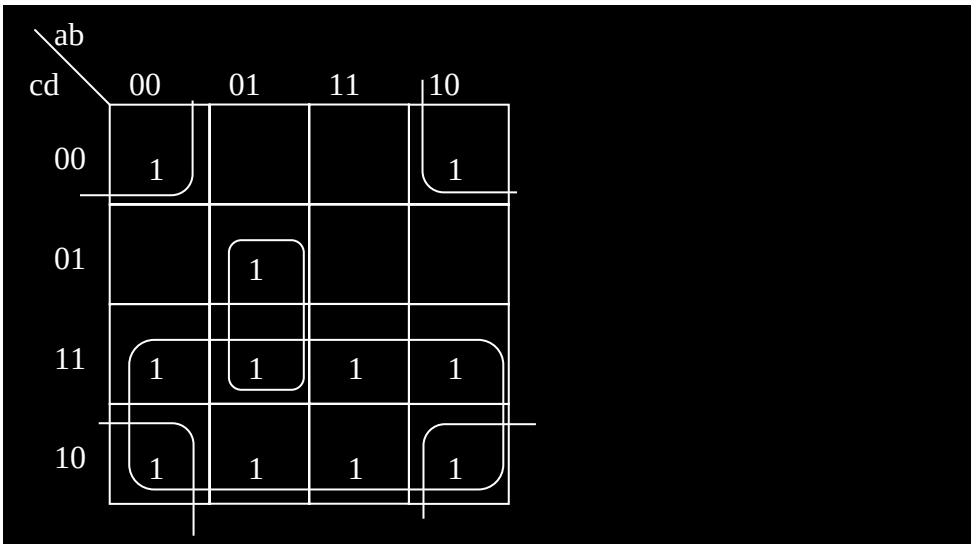

```
module Q3b2(a, b, c, d, f1);

input a, b, c, d;
output f1;

assign f1 =   (b & ~c)
            | (~a & ~b & d)
            | (a & ~b & c & ~d);

endmodule
```

Simulation of unminimized Q3b:



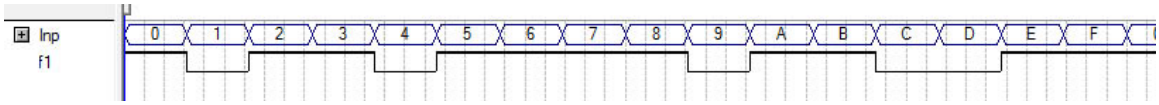
Simulation of minimized Q3b:



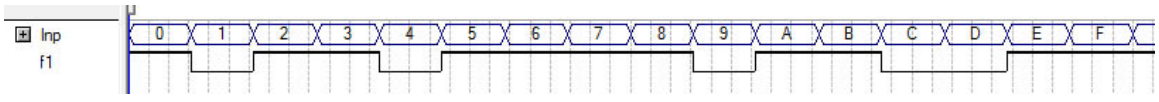



```
module Q3c2(a, b, c, d, f1);

input a, b, c, d;
output f1;

assign f1 = c | (~b & ~d) | (~a & b & d);

endmodule
```

Simulation of unminimized Q3c:



Simulation of minimized Q3c:



5. Write Verilog code for the circuit in Figure Q5 with and without wires. Compare the RTL view and simulation.

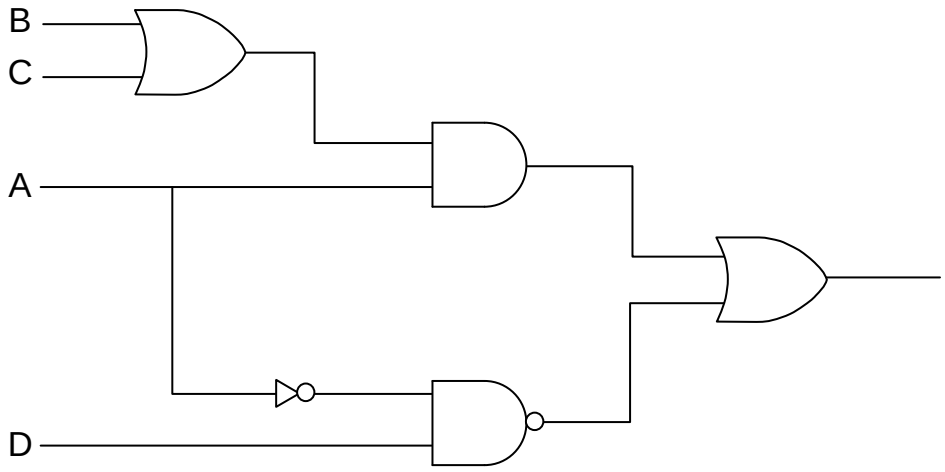
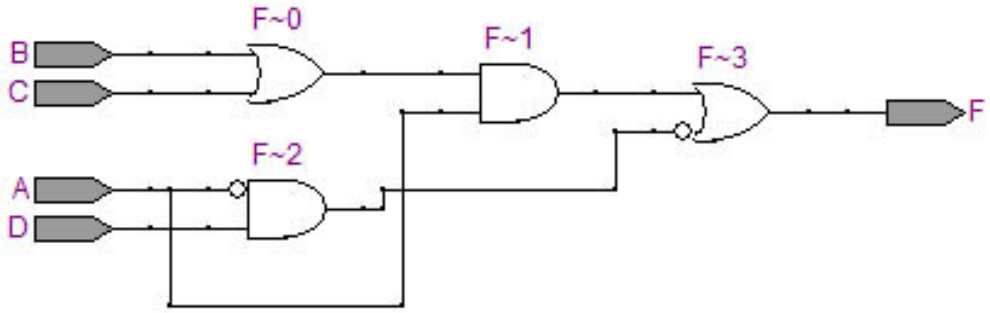
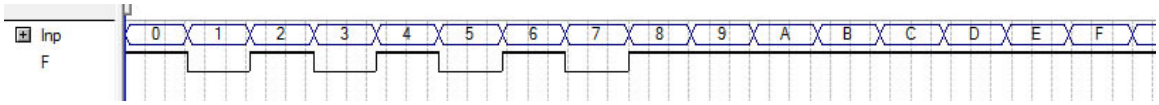


Figure Q5

Without wire:

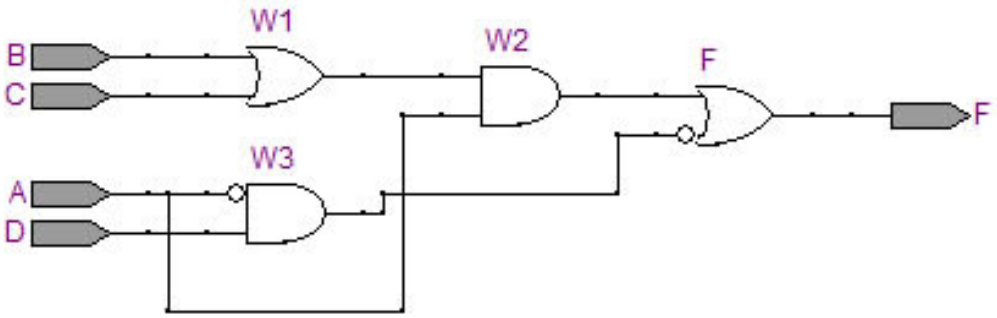
```
module Q5i(A, B, C, D, F);

input A, B, C, D;
output F;

assign F = ((B|C)&A)|~(~A&D);

endmodule
```

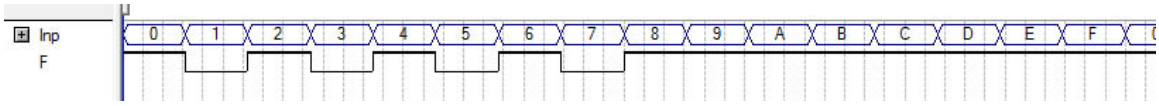




With wire:

```
module Q5ii(A, B, C, D, F);

input A, B, C, D;
output F;
wire W1, W2, W3;

assign W1 = B | C;
assign W2 = W1 & A;
assign W3 = ~(~A & D);
assign F = W2 | W3;

endmodule
```



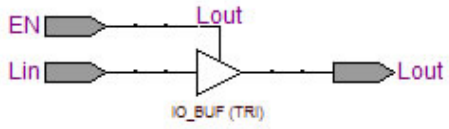


6. Write Verilog code for a tri-state buffer and another one for a tri-state inverter.

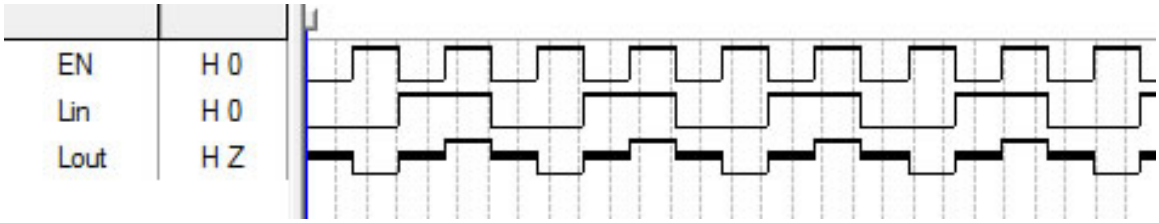
Tri-state buffer:

```
module TriStateBuffer(Lin, Lout, EN);

input Lin, EN;
output Lout;
reg Lout;

always@(Lin or EN)
begin
   if (EN)
      Lout <= Lin;
   else
      Lout <= 1'bz;
end

endmodule
```

RTL View:



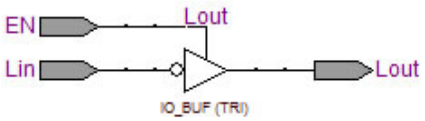
Simulation:



Tri-state buffer:

```
module TriStateInverter(Lin, Lout, EN);

input Lin, EN;
output Lout;
reg Lout;

always@(Lin or EN)
begin
   if (EN)
      Lout <= ~Lin;
   else
      Lout <= 1'bz;
end

endmodule
```

RTL View:



Simulation: