Lab03 Report

Park Chaehyun

1. Lab Practice

(1) 74x139

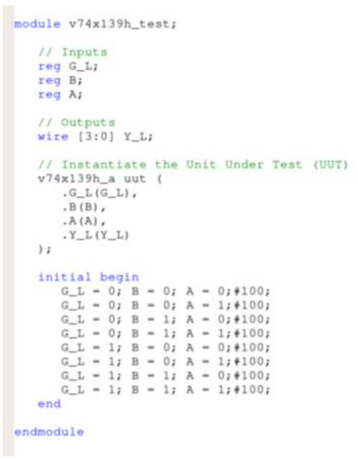
(i) Half 74x139

[1] Structural description

Code:

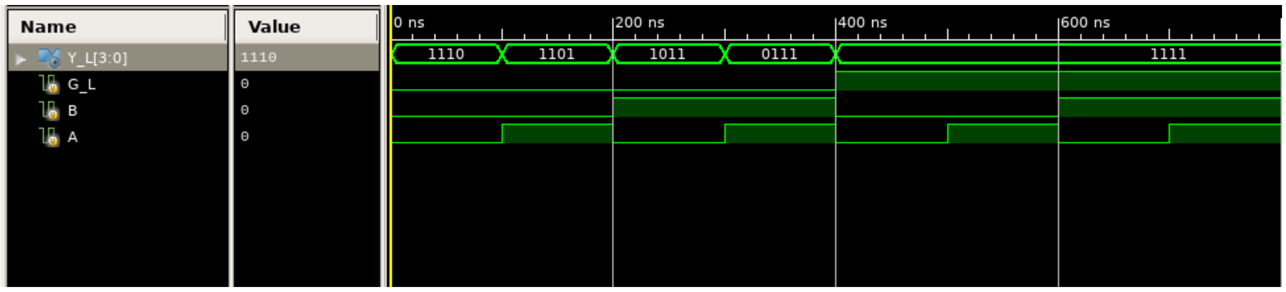
```
module v74x139h_a(
    input G_L,
    input A,
    input B,
    output [3:0] Y_L
    );

    wire N_A, N_B, N_G;

    not U1(N_G, G_L);
    not U2(N_A, A);
    not U3(N_B, B);

    nand U4(Y_L[0], N_G, N_A, N_B);
    nand U5(Y_L[1], N_G, A, N_B);
    nand U6(Y_L[2], N_G, N_A, B);
    nand U7(Y_L[3], N_G, A, B);

endmodule
```

Test code:



```
module v74x139h_test;

    // Inputs
    reg G_L;
    reg B;
    reg A;

    // Outputs
    wire [3:0] Y_L;

    // Instantiate the Unit Under Test (UUT)
    v74x139h_a uut (
        .G_L(G_L),
        .B(B),
        .A(A),
        .Y_L(Y_L)
    );

    initial begin
        G_L = 0; B = 0; A = 0;#100;
        G_L = 0; B = 0; A = 1;#100;
        G_L = 0; B = 1; A = 0;#100;
        G_L = 0; B = 1; A = 1;#100;
        G_L = 1; B = 0; A = 0;#100;
        G_L = 1; B = 0; A = 1;#100;
        G_L = 1; B = 1; A = 0;#100;
        G_L = 1; B = 1; A = 1;#100;
    end

endmodule
```

Simulation result:



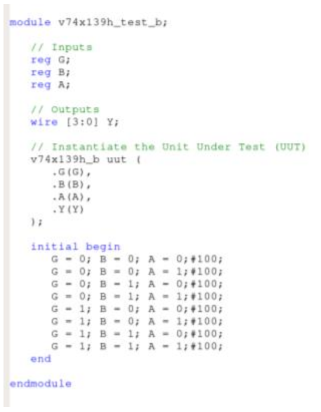
When G is 1, the decoder is not enabled and gives 1111. When G is 0, the decoder is enabled. When B is 0 and A is 0 the output is 1110, when B is 0 and A is 1 the output is 1101, when B is 1 and A is 0 the output is 1011 and when B is 1 and A is 1 the output is 0111, which is the desired behavior for the 2-to-4 decoder.

[2] Data-flow description

Code:

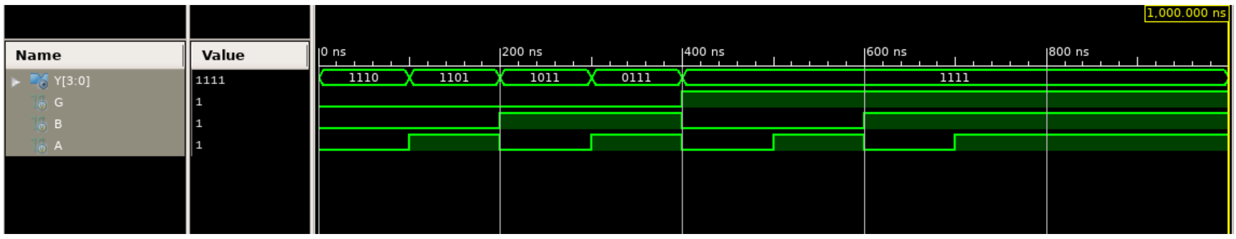
```
module v74x139h_b(
    input G,
    input A,
    input B,
    output [3:0] Y
    );

    wire [1:0] sel;
    wire [3:0] out;

    assign sel = {B, A};
    assign Y = ~out;

    assign out = (sel == 2'b00 && G == 1'b0) ? 4'b0001 :
                 (sel == 2'b01 && G == 1'b0) ? 4'b0010 :
                 (sel == 2'b10 && G == 1'b0) ? 4'b0100 :
                 (sel == 2'b11 && G == 1'b0) ? 4'b1000 :
                 4'b0000;

endmodule
```

Test code:



```
module v74x139h_test_b;

    // Inputs
    reg G;
    reg B;
    reg A;

    // Outputs
    wire [3:0] Y;

    // Instantiate the Unit Under Test (UUT)
    v74x139h_b uut (
        .G(G),
        .B(B),
        .A(A),
        .Y(Y)
    );

    initial begin
        G = 0; B = 0; A = 0;#100;
        G = 0; B = 0; A = 1;#100;
        G = 0; B = 1; A = 0;#100;
        G = 0; B = 1; A = 1;#100;
        G = 1; B = 0; A = 0;#100;
        G = 1; B = 0; A = 1;#100;
        G = 1; B = 1; A = 0;#100;
        G = 1; B = 1; A = 1;#100;
    end

endmodule
```

Simulation result:



The result is the same as (i)-[1], which is the desired result.

[3] Behavioral description

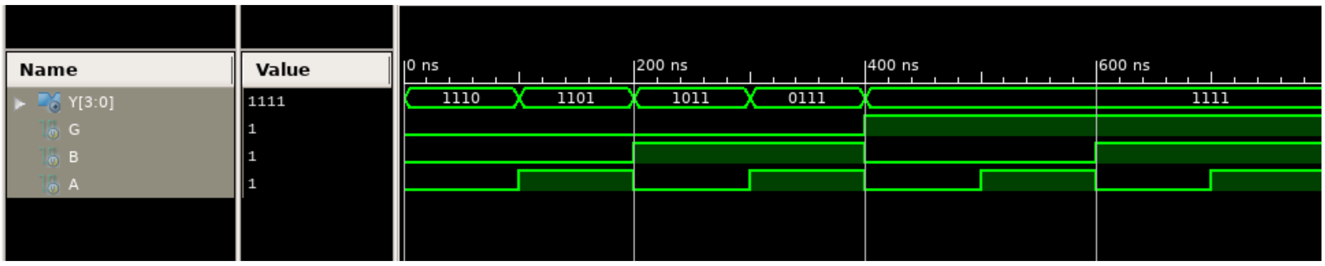
Code:

```
module v74x139h_c(
    input G,
    input A,
    input B,
    output [3:0] Y
    );

    wire [1:0] sel;
    reg [3:0] out;

    assign sel = {B, A};
    assign Y = ~out;

    always@(G or sel)
      begin
         if (G == 1'b0)
            begin
               case(sel)
                  2'b00 : out = 4'b0001;
                  2'b01 : out = 4'b0010;
                  2'b10 : out = 4'b0100;
                  2'b11 : out = 4'b1000;
               endcase
            end
         else
            begin
               out = 4'b0000;
            end
      end
endmodule
```

Test code:

```
module v17x139h_test_c;

   // Inputs
   reg G;
   reg B;
   reg A;

   // Outputs
   wire [3:0] Y;

   // Instantiate the Unit Under Test (UUT)
   v74x139h_c uut (
      .G(G),
      .B(B),
      .A(A),
      .Y(Y)
   );

   initial begin
      G = 0; B = 0; A = 0;#100;
      G = 0; B = 0; A = 1;#100;
      G = 0; B = 1; A = 0;#100;
      G = 0; B = 1; A = 1;#100;
      G = 1; B = 0; A = 0;#100;
      G = 1; B = 0; A = 1;#100;
      G = 1; B = 1; A = 0;#100;
      G = 1; B = 1; A = 1;#100;
   end
```

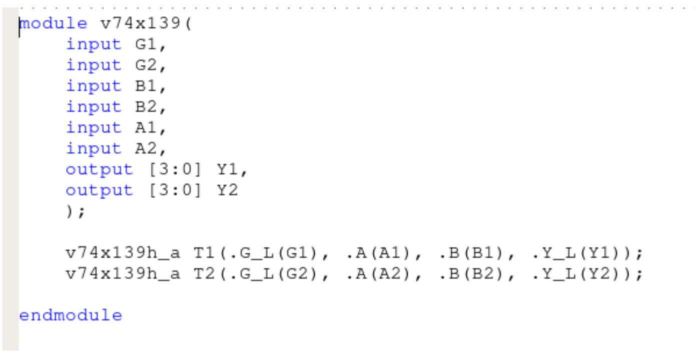
Simulation result:



The result is the same as (i)-[1], which is the desired result.

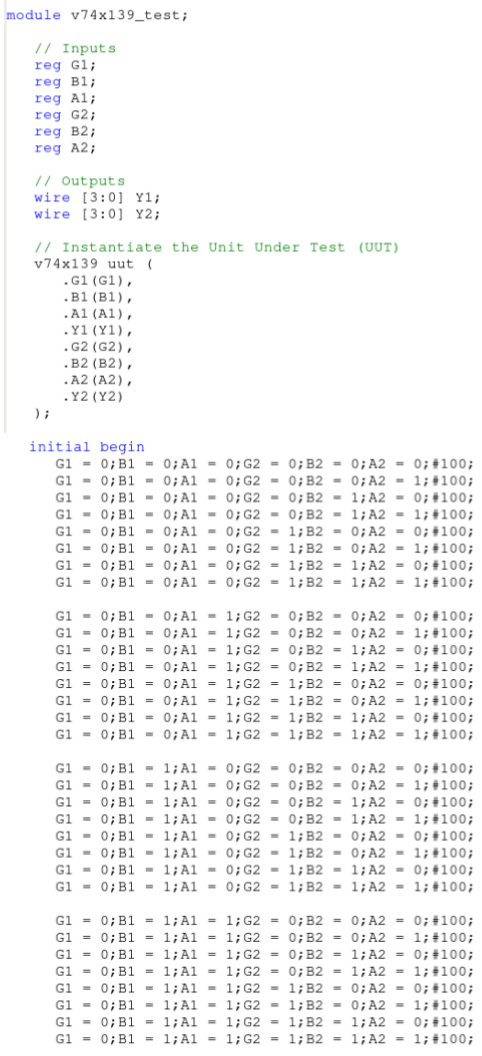
(ii) Full 74x139
Combined 2 half 74x139 using the implementation of (i)-[1]:

Code:



```
module v74x139(
    input G1,
    input G2,
    input B1,
    input B2,
    input A1,
    input A2,
    output [3:0] Y1,
    output [3:0] Y2
    );

    v74x139h_a T1(.G_L(G1), .A(A1), .B(B1), .Y_L(Y1));
    v74x139h_a T2(.G_L(G2), .A(A2), .B(B2), .Y_L(Y2));

endmodule
```

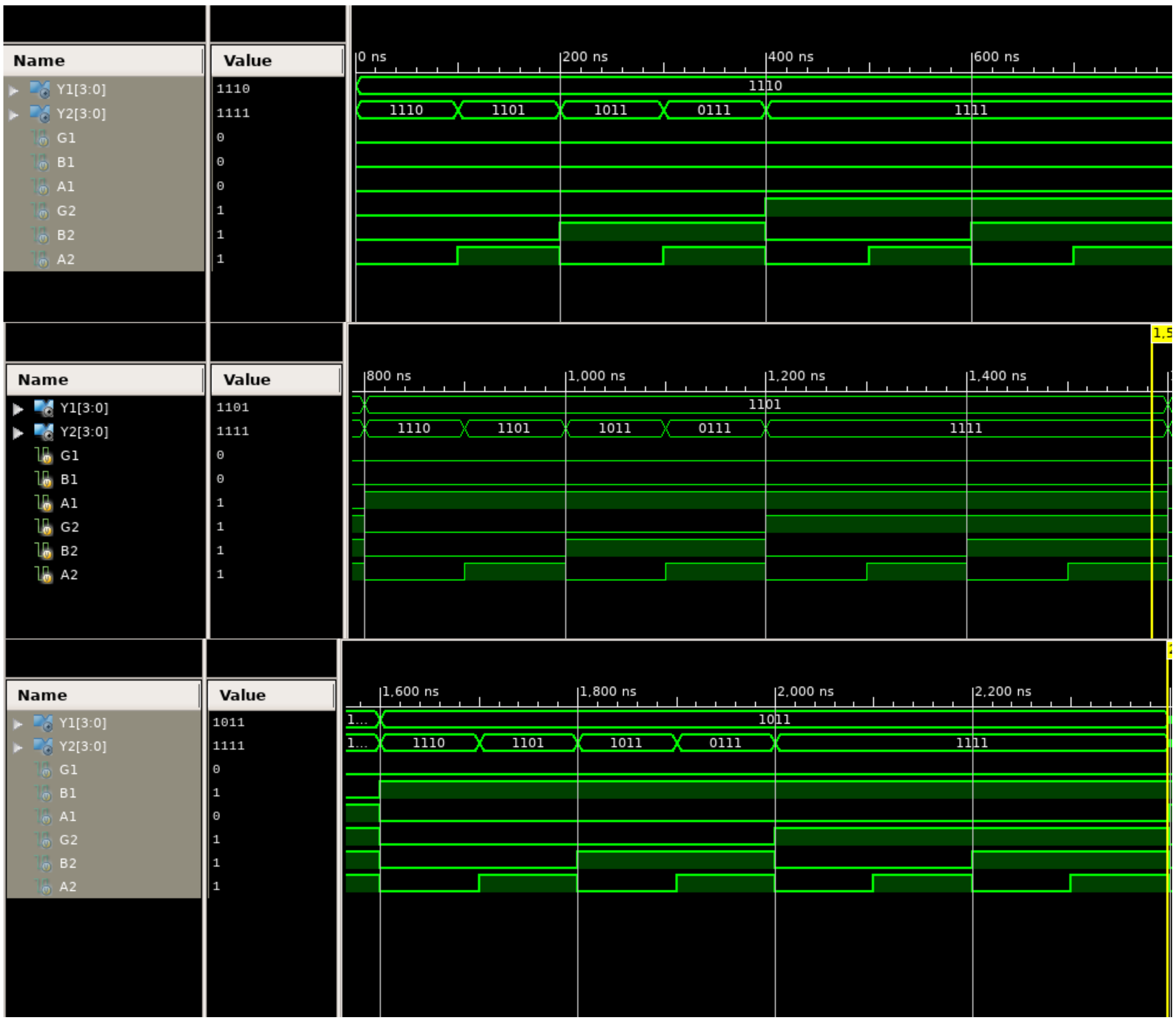
Test code:

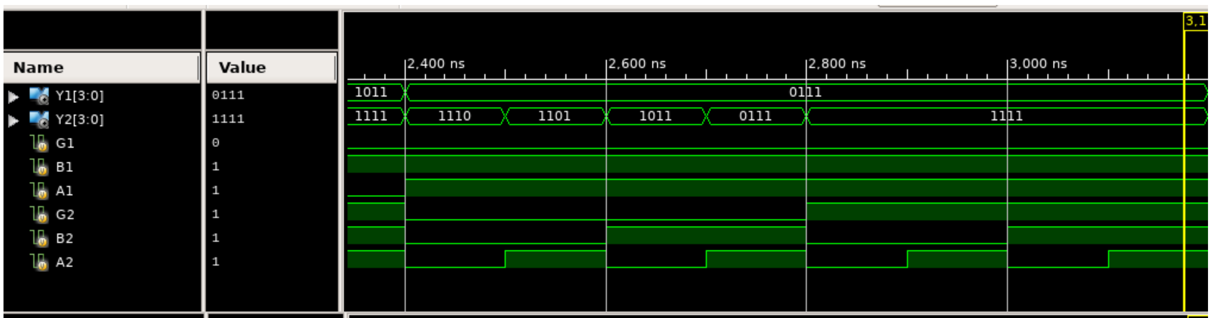


```
module v74x139_test;

   // Inputs
   reg G1;
   reg B1;
   reg A1;
   reg G2;
   reg B2;
   reg A2;

   // Outputs
   wire [3:0] Y1;
   wire [3:0] Y2;

   // Instantiate the Unit Under Test (UUT)
   v74x139 uut (
      .G1(G1),
      .B1(B1),
      .A1(A1),
      .Y1(Y1),
      .G2(G2),
      .B2(B2),
      .A2(A2),
      .Y2(Y2)
   );

  initial begin
     G1 = 0;B1 = 0;A1 = 0;G2 = 0;B2 = 0;A2 = 0;#100;
     G1 = 0;B1 = 0;A1 = 0;G2 = 0;B2 = 0;A2 = 1;#100;
     G1 = 0;B1 = 0;A1 = 0;G2 = 0;B2 = 1;A2 = 0;#100;
     G1 = 0;B1 = 0;A1 = 0;G2 = 0;B2 = 1;A2 = 1;#100;
     G1 = 0;B1 = 0;A1 = 0;G2 = 1;B2 = 0;A2 = 0;#100;
     G1 = 0;B1 = 0;A1 = 0;G2 = 1;B2 = 0;A2 = 1;#100;
     G1 = 0;B1 = 0;A1 = 0;G2 = 1;B2 = 1;A2 = 0;#100;
     G1 = 0;B1 = 0;A1 = 0;G2 = 1;B2 = 1;A2 = 1;#100;

     G1 = 0;B1 = 0;A1 = 1;G2 = 0;B2 = 0;A2 = 0;#100;
     G1 = 0;B1 = 0;A1 = 1;G2 = 0;B2 = 0;A2 = 1;#100;
     G1 = 0;B1 = 0;A1 = 1;G2 = 0;B2 = 1;A2 = 0;#100;
     G1 = 0;B1 = 0;A1 = 1;G2 = 0;B2 = 1;A2 = 1;#100;
     G1 = 0;B1 = 0;A1 = 1;G2 = 1;B2 = 0;A2 = 0;#100;
     G1 = 0;B1 = 0;A1 = 1;G2 = 1;B2 = 0;A2 = 1;#100;
     G1 = 0;B1 = 0;A1 = 1;G2 = 1;B2 = 1;A2 = 0;#100;
     G1 = 0;B1 = 0;A1 = 1;G2 = 1;B2 = 1;A2 = 1;#100;

     G1 = 0;B1 = 1;A1 = 0;G2 = 0;B2 = 0;A2 = 0;#100;
     G1 = 0;B1 = 1;A1 = 0;G2 = 0;B2 = 0;A2 = 1;#100;
     G1 = 0;B1 = 1;A1 = 0;G2 = 0;B2 = 1;A2 = 0;#100;
     G1 = 0;B1 = 1;A1 = 0;G2 = 0;B2 = 1;A2 = 1;#100;
     G1 = 0;B1 = 1;A1 = 0;G2 = 1;B2 = 0;A2 = 0;#100;
     G1 = 0;B1 = 1;A1 = 0;G2 = 1;B2 = 0;A2 = 1;#100;
     G1 = 0;B1 = 1;A1 = 0;G2 = 1;B2 = 1;A2 = 0;#100;
     G1 = 0;B1 = 1;A1 = 0;G2 = 1;B2 = 1;A2 = 1;#100;

     G1 = 0;B1 = 1;A1 = 1;G2 = 0;B2 = 0;A2 = 0;#100;
     G1 = 0;B1 = 1;A1 = 1;G2 = 0;B2 = 0;A2 = 1;#100;
     G1 = 0;B1 = 1;A1 = 1;G2 = 0;B2 = 1;A2 = 0;#100;
     G1 = 0;B1 = 1;A1 = 1;G2 = 0;B2 = 1;A2 = 1;#100;
     G1 = 0;B1 = 1;A1 = 1;G2 = 1;B2 = 0;A2 = 0;#100;
     G1 = 0;B1 = 1;A1 = 1;G2 = 1;B2 = 0;A2 = 1;#100;
     G1 = 0;B1 = 1;A1 = 1;G2 = 1;B2 = 1;A2 = 0;#100;
     G1 = 0;B1 = 1;A1 = 1;G2 = 1;B2 = 1;A2 = 1;#100;
```

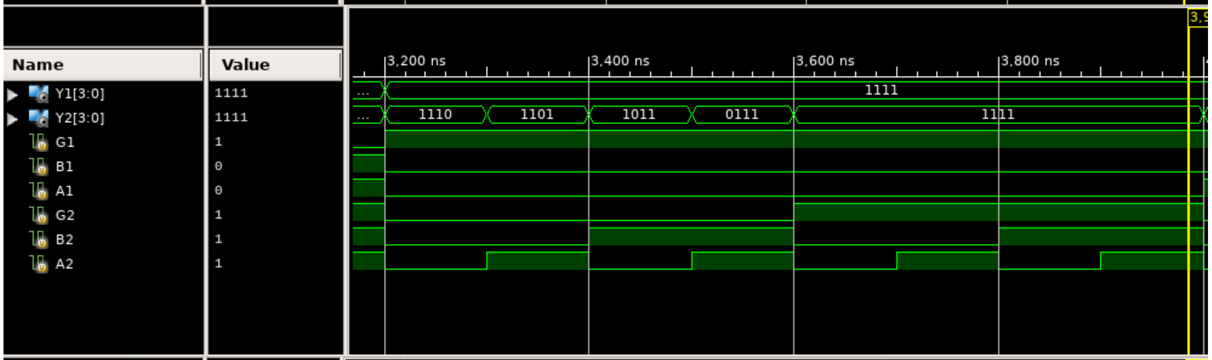
```
        G1 = 1;B1 = 0;A1 = 0;G2 = 0;B2 = 0;A2 = 0;#100;
        G1 = 1;B1 = 0;A1 = 0;G2 = 0;B2 = 0;A2 = 1;#100;
        G1 = 1;B1 = 0;A1 = 0;G2 = 0;B2 = 1;A2 = 0;#100;
        G1 = 1;B1 = 0;A1 = 0;G2 = 0;B2 = 1;A2 = 1;#100;
        G1 = 1;B1 = 0;A1 = 0;G2 = 1;B2 = 0;A2 = 0;#100;
        G1 = 1;B1 = 0;A1 = 0;G2 = 1;B2 = 0;A2 = 1;#100;
        G1 = 1;B1 = 0;A1 = 0;G2 = 1;B2 = 1;A2 = 0;#100;
        G1 = 1;B1 = 0;A1 = 0;G2 = 1;B2 = 1;A2 = 1;#100;

        G1 = 1;B1 = 0;A1 = 1;G2 = 0;B2 = 0;A2 = 0;#100;
        G1 = 1;B1 = 0;A1 = 1;G2 = 0;B2 = 0;A2 = 1;#100;
        G1 = 1;B1 = 0;A1 = 1;G2 = 0;B2 = 1;A2 = 0;#100;
        G1 = 1;B1 = 0;A1 = 1;G2 = 0;B2 = 1;A2 = 1;#100;
        G1 = 1;B1 = 0;A1 = 1;G2 = 1;B2 = 0;A2 = 0;#100;
        G1 = 1;B1 = 0;A1 = 1;G2 = 1;B2 = 0;A2 = 1;#100;
        G1 = 1;B1 = 0;A1 = 1;G2 = 1;B2 = 1;A2 = 0;#100;
        G1 = 1;B1 = 0;A1 = 1;G2 = 1;B2 = 1;A2 = 1;#100;

        G1 = 1;B1 = 1;A1 = 0;G2 = 0;B2 = 0;A2 = 0;#100;
        G1 = 1;B1 = 1;A1 = 0;G2 = 0;B2 = 0;A2 = 1;#100;
        G1 = 1;B1 = 1;A1 = 0;G2 = 0;B2 = 1;A2 = 0;#100;
        G1 = 1;B1 = 1;A1 = 0;G2 = 0;B2 = 1;A2 = 1;#100;
        G1 = 1;B1 = 1;A1 = 0;G2 = 1;B2 = 0;A2 = 0;#100;
        G1 = 1;B1 = 1;A1 = 0;G2 = 1;B2 = 0;A2 = 1;#100;
        G1 = 1;B1 = 1;A1 = 0;G2 = 1;B2 = 1;A2 = 0;#100;
        G1 = 1;B1 = 1;A1 = 0;G2 = 1;B2 = 1;A2 = 1;#100;

        G1 = 1;B1 = 1;A1 = 1;G2 = 0;B2 = 0;A2 = 0;#100;
        G1 = 1;B1 = 1;A1 = 1;G2 = 0;B2 = 0;A2 = 1;#100;
        G1 = 1;B1 = 1;A1 = 1;G2 = 0;B2 = 1;A2 = 0;#100;
        G1 = 1;B1 = 1;A1 = 1;G2 = 0;B2 = 1;A2 = 1;#100;
        G1 = 1;B1 = 1;A1 = 1;G2 = 1;B2 = 0;A2 = 0;#100;
        G1 = 1;B1 = 1;A1 = 1;G2 = 1;B2 = 0;A2 = 1;#100;
        G1 = 1;B1 = 1;A1 = 1;G2 = 1;B2 = 1;A2 = 0;#100;
        G1 = 1;B1 = 1;A1 = 1;G2 = 1;B2 = 1;A2 = 1;#100;
    end

endmodule
```

Simulation result:

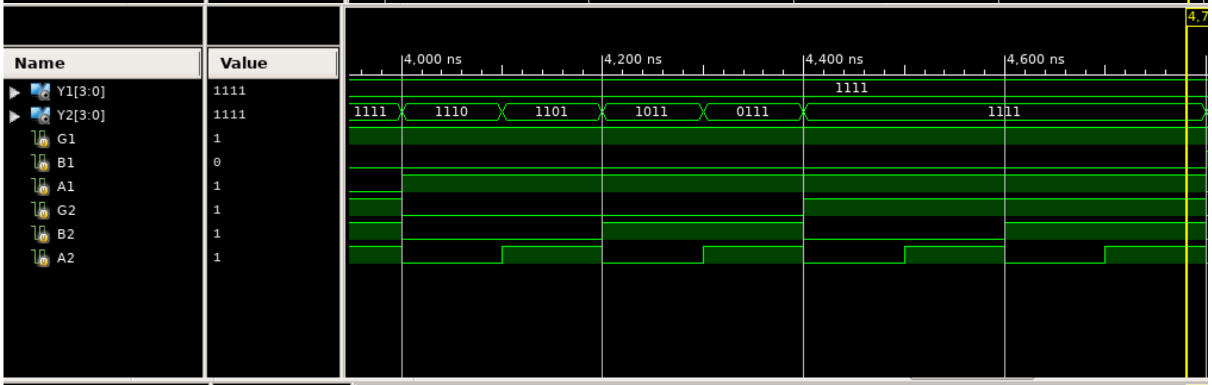




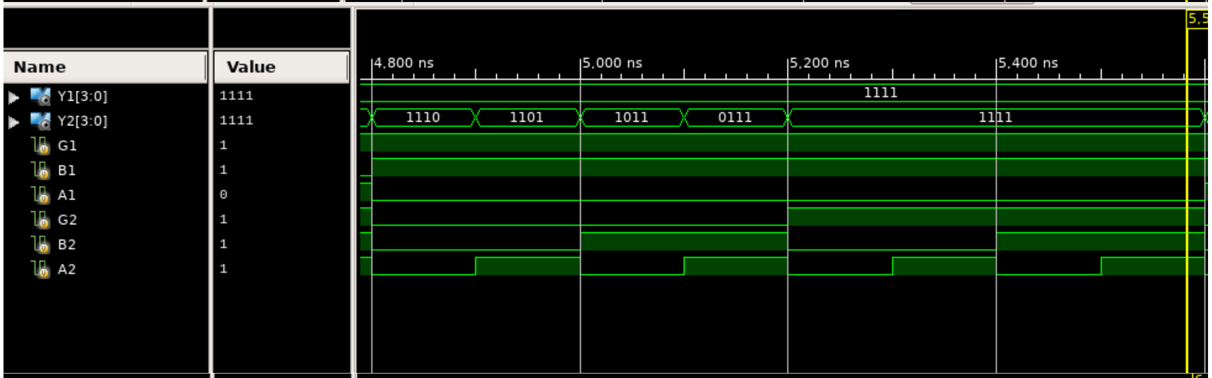
Name	Value
Y1[3:0]	0111
Y2[3:0]	1111
G1	0
B1	1
A1	1
G2	1
B2	1
A2	1



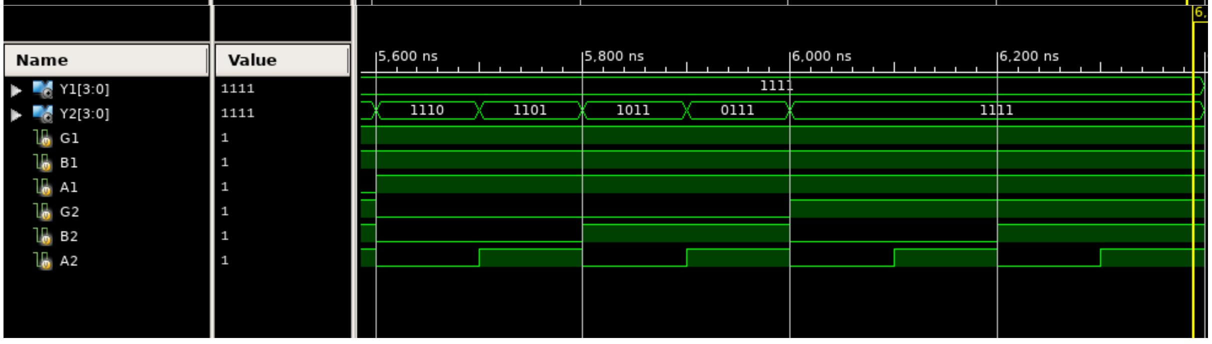
Name	Value
Y1[3:0]	1111
Y2[3:0]	1111
G1	1
B1	0
A1	0
G2	1
B2	1
A2	1



Name	Value
Y1[3:0]	1111
Y2[3:0]	1111
G1	1
B1	0
A1	1
G2	1
B2	1
A2	1



Name	Value
Y1[3:0]	1111
Y2[3:0]	1111
G1	1
B1	1
A1	0
G2	1
B2	1
A2	1



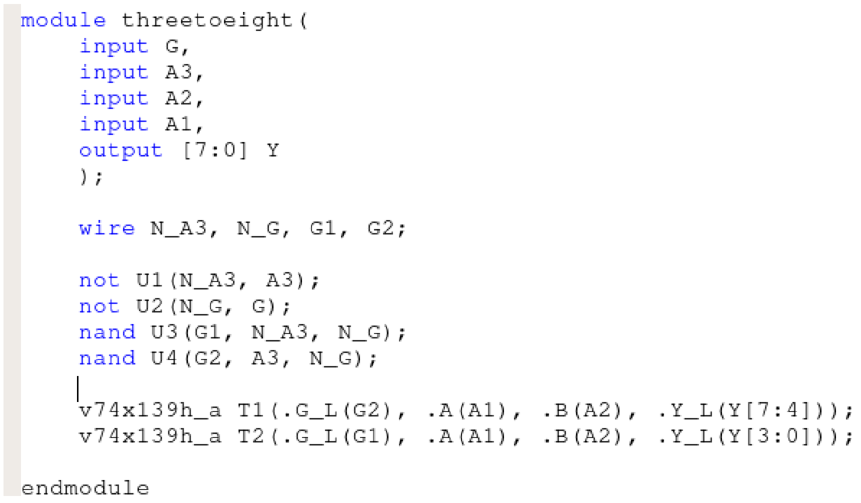
Name	Value
Y1[3:0]	1111
Y2[3:0]	1111
G1	1
B1	1
A1	1
G2	1
B2	1
A2	1

Each half of 74x139, which is a 2-to-4 decoder, gives the desired result, as described in (i)-[1].

(2) 3-to-8 decoder using 2-to-4 decoders

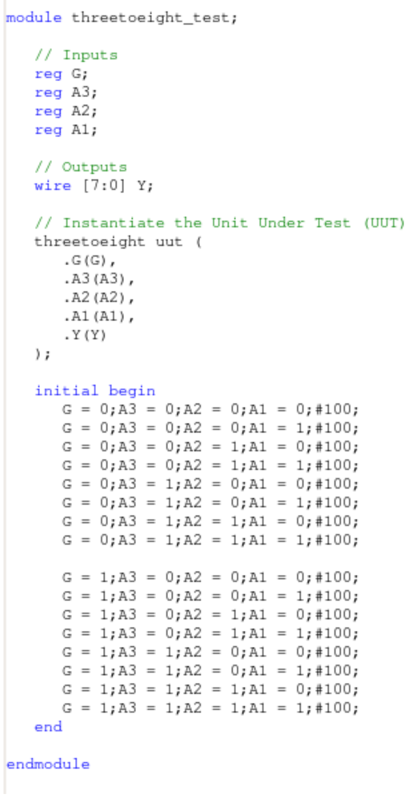
I used structural description. I used two instances of the 2-to-4 decoder, using the implementation done with structural description in (i)-[1]. G is the enable signal, and A3, A2 and A1 are inputs, A3 being MSB and A1 being LSB.

Code:



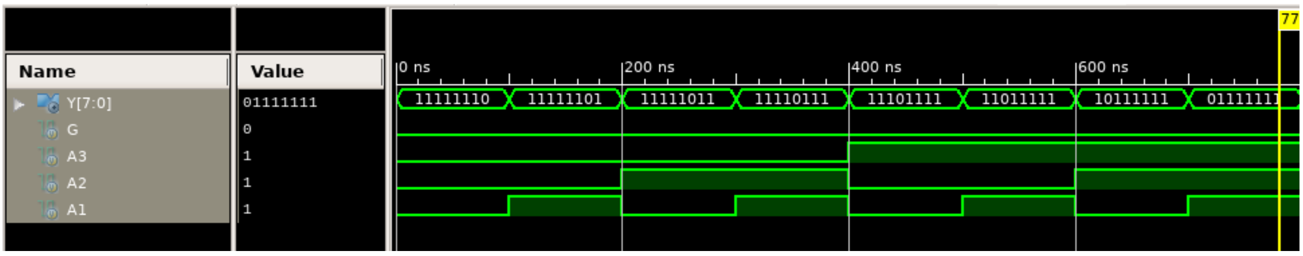
```
module threetoeight(
    input G,
    input A3,
    input A2,
    input A1,
    output [7:0] Y
    );

    wire N_A3, N_G, G1, G2;

    not U1(N_A3, A3);
    not U2(N_G, G);
    nand U3(G1, N_A3, N_G);
    nand U4(G2, A3, N_G);

    v74x139h_a T1(.G_L(G2), .A(A1), .B(A2), .Y_L(Y[7:4]));
    v74x139h_a T2(.G_L(G1), .A(A1), .B(A2), .Y_L(Y[3:0]));

endmodule
```

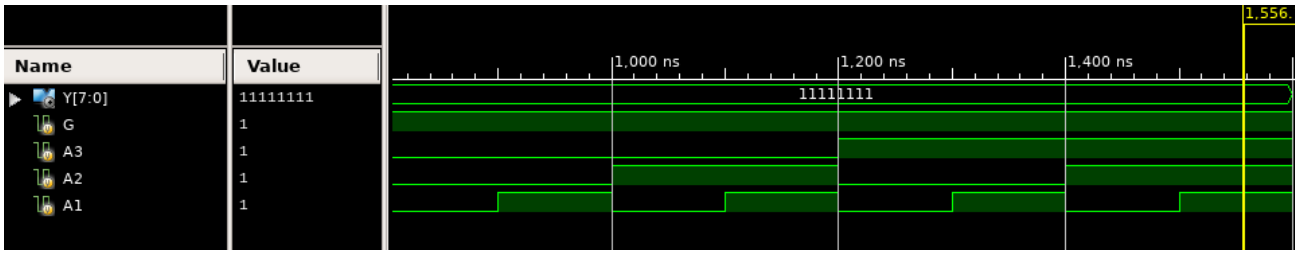
Test code:



```
module threetoeight_test;

   // Inputs
   reg G;
   reg A3;
   reg A2;
   reg A1;

   // Outputs
   wire [7:0] Y;

   // Instantiate the Unit Under Test (UUT)
   threetoeight uut (
      .G(G),
      .A3(A3),
      .A2(A2),
      .A1(A1),
      .Y(Y)
   );

   initial begin
      G = 0;A3 = 0;A2 = 0;A1 = 0;#100;
      G = 0;A3 = 0;A2 = 0;A1 = 1;#100;
      G = 0;A3 = 0;A2 = 1;A1 = 0;#100;
      G = 0;A3 = 0;A2 = 1;A1 = 1;#100;
      G = 0;A3 = 1;A2 = 0;A1 = 0;#100;
      G = 0;A3 = 1;A2 = 0;A1 = 1;#100;
      G = 0;A3 = 1;A2 = 1;A1 = 0;#100;
      G = 0;A3 = 1;A2 = 1;A1 = 1;#100;

      G = 1;A3 = 0;A2 = 0;A1 = 0;#100;
      G = 1;A3 = 0;A2 = 0;A1 = 1;#100;
      G = 1;A3 = 0;A2 = 1;A1 = 0;#100;
      G = 1;A3 = 0;A2 = 1;A1 = 1;#100;
      G = 1;A3 = 1;A2 = 0;A1 = 0;#100;
      G = 1;A3 = 1;A2 = 0;A1 = 1;#100;
      G = 1;A3 = 1;A2 = 1;A1 = 0;#100;
      G = 1;A3 = 1;A2 = 1;A1 = 1;#100;
   end

endmodule
```

Result:





When G is 1, the decoder is not enabled and all outputs are 1. When G is 0, for each of the numbers from 0 to 7, the corresponding bit is 0 and all other bits are 1. This is the desired behavior for the 3-to-8 decoder.

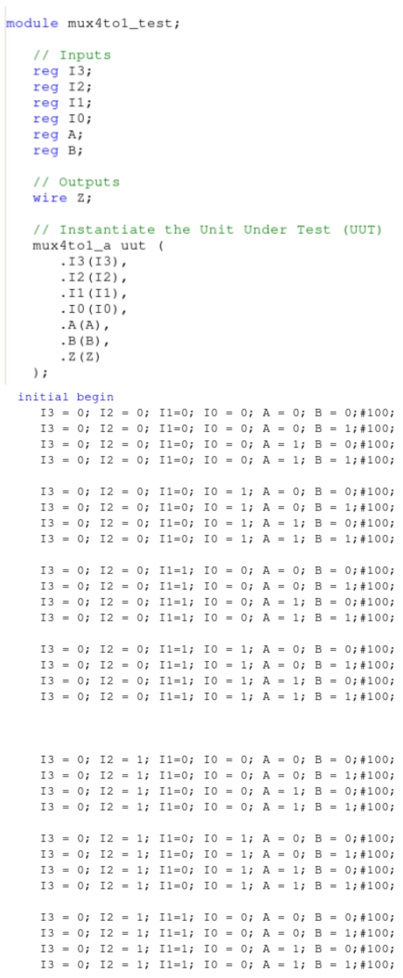
2. 4-to-1 MUX

(1) Structural description

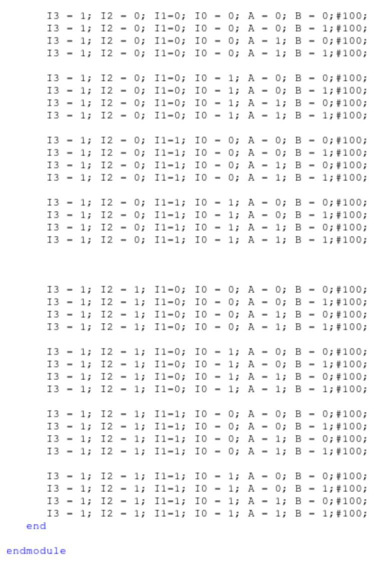
Code:

```
module threetoeight_test;

   // Inputs
   reg G;
   reg A3;
   reg A2;
   reg A1;

   // Outputs
   wire [7:0] Y;

   // Instantiate the Unit Under Test (UUT)
   threetoeight uut (
      .G(G),
      .A3(A3),
      .A2(A2),
      .A1(A1),
      .Y(Y)
   );

   initial begin
      G = 0;A3 = 0;A2 = 0;A1 = 0;#100;
      G = 0;A3 = 0;A2 = 0;A1 = 1;#100;
      G = 0;A3 = 0;A2 = 1;A1 = 0;#100;
      G = 0;A3 = 0;A2 = 1;A1 = 1;#100;
      G = 0;A3 = 1;A2 = 0;A1 = 0;#100;
      G = 0;A3 = 1;A2 = 0;A1 = 1;#100;
      G = 0;A3 = 1;A2 = 1;A1 = 0;#100;
      G = 0;A3 = 1;A2 = 1;A1 = 1;#100;

      G = 1;A3 = 0;A2 = 0;A1 = 0;#100;
      G = 1;A3 = 0;A2 = 0;A1 = 1;#100;
      G = 1;A3 = 0;A2 = 1;A1 = 0;#100;
      G = 1;A3 = 0;A2 = 1;A1 = 1;#100;
      G = 1;A3 = 1;A2 = 0;A1 = 0;#100;
      G = 1;A3 = 1;A2 = 0;A1 = 1;#100;
      G = 1;A3 = 1;A2 = 1;A1 = 0;#100;
      G = 1;A3 = 1;A2 = 1;A1 = 1;#100;
   end

endmodule
```

Test code:



```verilog
module mux4to1_test;

   // Inputs
   reg I3;
   reg I2;
   reg I1;
   reg I0;
   reg A;
   reg B;

   // Outputs
   wire Z;

   // Instantiate the Unit Under Test (UUT)
   mux4to1_a uut (
      .I3(I3),
      .I2(I2),
      .I1(I1),
      .I0(I0),
      .A(A),
      .B(B),
      .Z(Z)
   );
 initial begin
    I3 = 0; I2 = 0; I1=0; I0 = 0; A = 0; B = 0;#100;
    I3 = 0; I2 = 0; I1=0; I0 = 0; A = 0; B = 1;#100;
    I3 = 0; I2 = 0; I1=0; I0 = 0; A = 1; B = 0;#100;
    I3 = 0; I2 = 0; I1=0; I0 = 0; A = 1; B = 1;#100;

    I3 = 0; I2 = 0; I1=0; I0 = 1; A = 0; B = 0;#100;
    I3 = 0; I2 = 0; I1=0; I0 = 1; A = 0; B = 1;#100;
    I3 = 0; I2 = 0; I1=0; I0 = 1; A = 1; B = 0;#100;
    I3 = 0; I2 = 0; I1=0; I0 = 1; A = 1; B = 1;#100;

    I3 = 0; I2 = 0; I1=1; I0 = 0; A = 0; B = 0;#100;
    I3 = 0; I2 = 0; I1=1; I0 = 0; A = 0; B = 1;#100;
    I3 = 0; I2 = 0; I1=1; I0 = 0; A = 1; B = 0;#100;
    I3 = 0; I2 = 0; I1=1; I0 = 0; A = 1; B = 1;#100;

    I3 = 0; I2 = 0; I1=1; I0 = 1; A = 0; B = 0;#100;
    I3 = 0; I2 = 0; I1=1; I0 = 1; A = 0; B = 1;#100;
    I3 = 0; I2 = 0; I1=1; I0 = 1; A = 1; B = 0;#100;
    I3 = 0; I2 = 0; I1=1; I0 = 1; A = 1; B = 1;#100;



    I3 = 0; I2 = 1; I1=0; I0 = 0; A = 0; B = 0;#100;
    I3 = 0; I2 = 1; I1=0; I0 = 0; A = 0; B = 1;#100;
    I3 = 0; I2 = 1; I1=0; I0 = 0; A = 1; B = 0;#100;
    I3 = 0; I2 = 1; I1=0; I0 = 0; A = 1; B = 1;#100;

    I3 = 0; I2 = 1; I1=0; I0 = 1; A = 0; B = 0;#100;
    I3 = 0; I2 = 1; I1=0; I0 = 1; A = 0; B = 1;#100;
    I3 = 0; I2 = 1; I1=0; I0 = 1; A = 1; B = 0;#100;
    I3 = 0; I2 = 1; I1=0; I0 = 1; A = 1; B = 1;#100;

    I3 = 0; I2 = 1; I1=1; I0 = 0; A = 0; B = 0;#100;
    I3 = 0; I2 = 1; I1=1; I0 = 0; A = 0; B = 1;#100;
    I3 = 0; I2 = 1; I1=1; I0 = 0; A = 1; B = 0;#100;
    I3 = 0; I2 = 1; I1=1; I0 = 0; A = 1; B = 1;#100;
```



```verilog
    I3 = 1; I2 = 0; I1=0; I0 = 0; A = 0; B = 0;#100;
    I3 = 1; I2 = 0; I1=0; I0 = 0; A = 0; B = 1;#100;
    I3 = 1; I2 = 0; I1=0; I0 = 0; A = 1; B = 0;#100;
    I3 = 1; I2 = 0; I1=0; I0 = 0; A = 1; B = 1;#100;

    I3 = 1; I2 = 0; I1=0; I0 = 1; A = 0; B = 0;#100;
    I3 = 1; I2 = 0; I1=0; I0 = 1; A = 0; B = 1;#100;
    I3 = 1; I2 = 0; I1=0; I0 = 1; A = 1; B = 0;#100;
    I3 = 1; I2 = 0; I1=0; I0 = 1; A = 1; B = 1;#100;

    I3 = 1; I2 = 0; I1=1; I0 = 0; A = 0; B = 0;#100;
    I3 = 1; I2 = 0; I1=1; I0 = 0; A = 0; B = 1;#100;
    I3 = 1; I2 = 0; I1=1; I0 = 0; A = 1; B = 0;#100;
    I3 = 1; I2 = 0; I1=1; I0 = 0; A = 1; B = 1;#100;

    I3 = 1; I2 = 0; I1=1; I0 = 1; A = 0; B = 0;#100;
    I3 = 1; I2 = 0; I1=1; I0 = 1; A = 0; B = 1;#100;
    I3 = 1; I2 = 0; I1=1; I0 = 1; A = 1; B = 0;#100;
    I3 = 1; I2 = 0; I1=1; I0 = 1; A = 1; B = 1;#100;



    I3 = 1; I2 = 1; I1=0; I0 = 0; A = 0; B = 0;#100;
    I3 = 1; I2 = 1; I1=0; I0 = 0; A = 0; B = 1;#100;
    I3 = 1; I2 = 1; I1=0; I0 = 0; A = 1; B = 0;#100;
    I3 = 1; I2 = 1; I1=0; I0 = 0; A = 1; B = 1;#100;

    I3 = 1; I2 = 1; I1=0; I0 = 1; A = 0; B = 0;#100;
    I3 = 1; I2 = 1; I1=0; I0 = 1; A = 0; B = 1;#100;
    I3 = 1; I2 = 1; I1=0; I0 = 1; A = 1; B = 0;#100;
    I3 = 1; I2 = 1; I1=0; I0 = 1; A = 1; B = 1;#100;

    I3 = 1; I2 = 1; I1=1; I0 = 0; A = 0; B = 0;#100;
    I3 = 1; I2 = 1; I1=1; I0 = 0; A = 0; B = 1;#100;
    I3 = 1; I2 = 1; I1=1; I0 = 0; A = 1; B = 0;#100;
    I3 = 1; I2 = 1; I1=1; I0 = 0; A = 1; B = 1;#100;

    I3 = 1; I2 = 1; I1=1; I0 = 1; A = 0; B = 0;#100;
    I3 = 1; I2 = 1; I1=1; I0 = 1; A = 0; B = 1;#100;
    I3 = 1; I2 = 1; I1=1; I0 = 1; A = 1; B = 0;#100;
    I3 = 1; I2 = 1; I1=1; I0 = 1; A = 1; B = 1;#100;
  end

endmodule
```

(2) Data-flow description

Code:

```verilog
module mux4to1_b(
    input I0,
    input I1,
    input I2,
    input I3,
    input A,
    input B,
    output Z
    );

    assign Z = A ? (B ? I3 : I2) : (B ? I1 : I0);

endmodule
```

Test code:

```
module mux4to1_test_b;

   // Inputs
   reg I3;
   reg I2;
   reg I1;
   reg I0;
   reg A;
   reg B;

   // Outputs
   wire Z;

   // Instantiate the Unit Under Test (UUT)
   mux4to1_b uut (
      .I3(I3),
      .I2(I2),
      .I1(I1),
      .I0(I0),
      .A(A),
      .B(B),
      .Z(Z)
   );
```

```
 initial begin
    I3 = 0; I2 = 0; I1=0; I0 = 0; A = 0; B = 0;#100;
    I3 = 0; I2 = 0; I1=0; I0 = 0; A = 0; B = 1;#100;
    I3 = 0; I2 = 0; I1=0; I0 = 0; A = 1; B = 0;#100;
    I3 = 0; I2 = 0; I1=0; I0 = 0; A = 1; B = 1;#100;

    I3 = 0; I2 = 0; I1=0; I0 = 1; A = 0; B = 0;#100;
    I3 = 0; I2 = 0; I1=0; I0 = 1; A = 0; B = 1;#100;
    I3 = 0; I2 = 0; I1=0; I0 = 1; A = 1; B = 0;#100;
    I3 = 0; I2 = 0; I1=0; I0 = 1; A = 1; B = 1;#100;

    I3 = 0; I2 = 0; I1=1; I0 = 0; A = 0; B = 0;#100;
    I3 = 0; I2 = 0; I1=1; I0 = 0; A = 0; B = 1;#100;
    I3 = 0; I2 = 0; I1=1; I0 = 0; A = 1; B = 0;#100;
    I3 = 0; I2 = 0; I1=1; I0 = 0; A = 1; B = 1;#100;

    I3 = 0; I2 = 0; I1=1; I0 = 1; A = 0; B = 0;#100;
    I3 = 0; I2 = 0; I1=1; I0 = 1; A = 0; B = 1;#100;
    I3 = 0; I2 = 0; I1=1; I0 = 1; A = 1; B = 0;#100;
    I3 = 0; I2 = 0; I1=1; I0 = 1; A = 1; B = 1;#100;


    I3 = 0; I2 = 1; I1=0; I0 = 0; A = 0; B = 0;#100;
    I3 = 0; I2 = 1; I1=0; I0 = 0; A = 0; B = 1;#100;
    I3 = 0; I2 = 1; I1=0; I0 = 0; A = 1; B = 0;#100;
    I3 = 0; I2 = 1; I1=0; I0 = 0; A = 1; B = 1;#100;

    I3 = 0; I2 = 1; I1=0; I0 = 1; A = 0; B = 0;#100;
    I3 = 0; I2 = 1; I1=0; I0 = 1; A = 0; B = 1;#100;
    I3 = 0; I2 = 1; I1=0; I0 = 1; A = 1; B = 0;#100;
    I3 = 0; I2 = 1; I1=0; I0 = 1; A = 1; B = 1;#100;

    I3 = 0; I2 = 1; I1=1; I0 = 0; A = 0; B = 0;#100;
    I3 = 0; I2 = 1; I1=1; I0 = 0; A = 0; B = 1;#100;
    I3 = 0; I2 = 1; I1=1; I0 = 0; A = 1; B = 0;#100;
    I3 = 0; I2 = 1; I1=1; I0 = 0; A = 1; B = 1;#100;
```

```
    I3 = 1; I2 = 0; I1=0; I0 = 0; A = 0; B = 0;#100;
    I3 = 1; I2 = 0; I1=0; I0 = 0; A = 0; B = 1;#100;
    I3 = 1; I2 = 0; I1=0; I0 = 0; A = 1; B = 0;#100;
    I3 = 1; I2 = 0; I1=0; I0 = 0; A = 1; B = 1;#100;

    I3 = 1; I2 = 0; I1=0; I0 = 1; A = 0; B = 0;#100;
    I3 = 1; I2 = 0; I1=0; I0 = 1; A = 0; B = 1;#100;
    I3 = 1; I2 = 0; I1=0; I0 = 1; A = 1; B = 0;#100;
    I3 = 1; I2 = 0; I1=0; I0 = 1; A = 1; B = 1;#100;

    I3 = 1; I2 = 0; I1=1; I0 = 0; A = 0; B = 0;#100;
    I3 = 1; I2 = 0; I1=1; I0 = 0; A = 0; B = 1;#100;
    I3 = 1; I2 = 0; I1=1; I0 = 0; A = 1; B = 0;#100;
    I3 = 1; I2 = 0; I1=1; I0 = 0; A = 1; B = 1;#100;

    I3 = 1; I2 = 0; I1=1; I0 = 1; A = 0; B = 0;#100;
    I3 = 1; I2 = 0; I1=1; I0 = 1; A = 0; B = 1;#100;
    I3 = 1; I2 = 0; I1=1; I0 = 1; A = 1; B = 0;#100;
    I3 = 1; I2 = 0; I1=1; I0 = 1; A = 1; B = 1;#100;


    I3 = 1; I2 = 1; I1=0; I0 = 0; A = 0; B = 0;#100;
    I3 = 1; I2 = 1; I1=0; I0 = 0; A = 0; B = 1;#100;
    I3 = 1; I2 = 1; I1=0; I0 = 0; A = 1; B = 0;#100;
    I3 = 1; I2 = 1; I1=0; I0 = 0; A = 1; B = 1;#100;

    I3 = 1; I2 = 1; I1=0; I0 = 1; A = 0; B = 0;#100;
    I3 = 1; I2 = 1; I1=0; I0 = 1; A = 0; B = 1;#100;
    I3 = 1; I2 = 1; I1=0; I0 = 1; A = 1; B = 0;#100;
    I3 = 1; I2 = 1; I1=0; I0 = 1; A = 1; B = 1;#100;

    I3 = 1; I2 = 1; I1=1; I0 = 0; A = 0; B = 0;#100;
    I3 = 1; I2 = 1; I1=1; I0 = 0; A = 0; B = 1;#100;
    I3 = 1; I2 = 1; I1=1; I0 = 0; A = 1; B = 0;#100;
    I3 = 1; I2 = 1; I1=1; I0 = 0; A = 1; B = 1;#100;

    I3 = 1; I2 = 1; I1=1; I0 = 1; A = 0; B = 0;#100;
    I3 = 1; I2 = 1; I1=1; I0 = 1; A = 0; B = 1;#100;
    I3 = 1; I2 = 1; I1=1; I0 = 1; A = 1; B = 0;#100;
    I3 = 1; I2 = 1; I1=1; I0 = 1; A = 1; B = 1;#100;
   end

endmodule
```

(3) Behavioral description

Code:

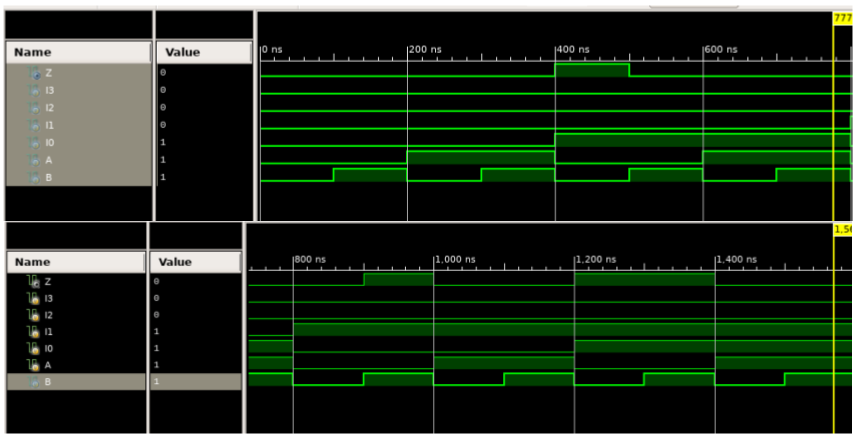
```
module mux4to1_c(
    input I0,
    input I1,
    input I2,
    input I3,
    input A,
    input B,
    output reg Z
    );

    always@ (I0 or I1 or I2 or I3 or A or B)
      begin
        if(A==0 & B==0)
          Z=I0;
        else if(A==0 & B==1)
          Z=I1;
        else if(A==1 & B==0)
          Z=I2;
        else
          Z=I3;
      end

endmodule
```

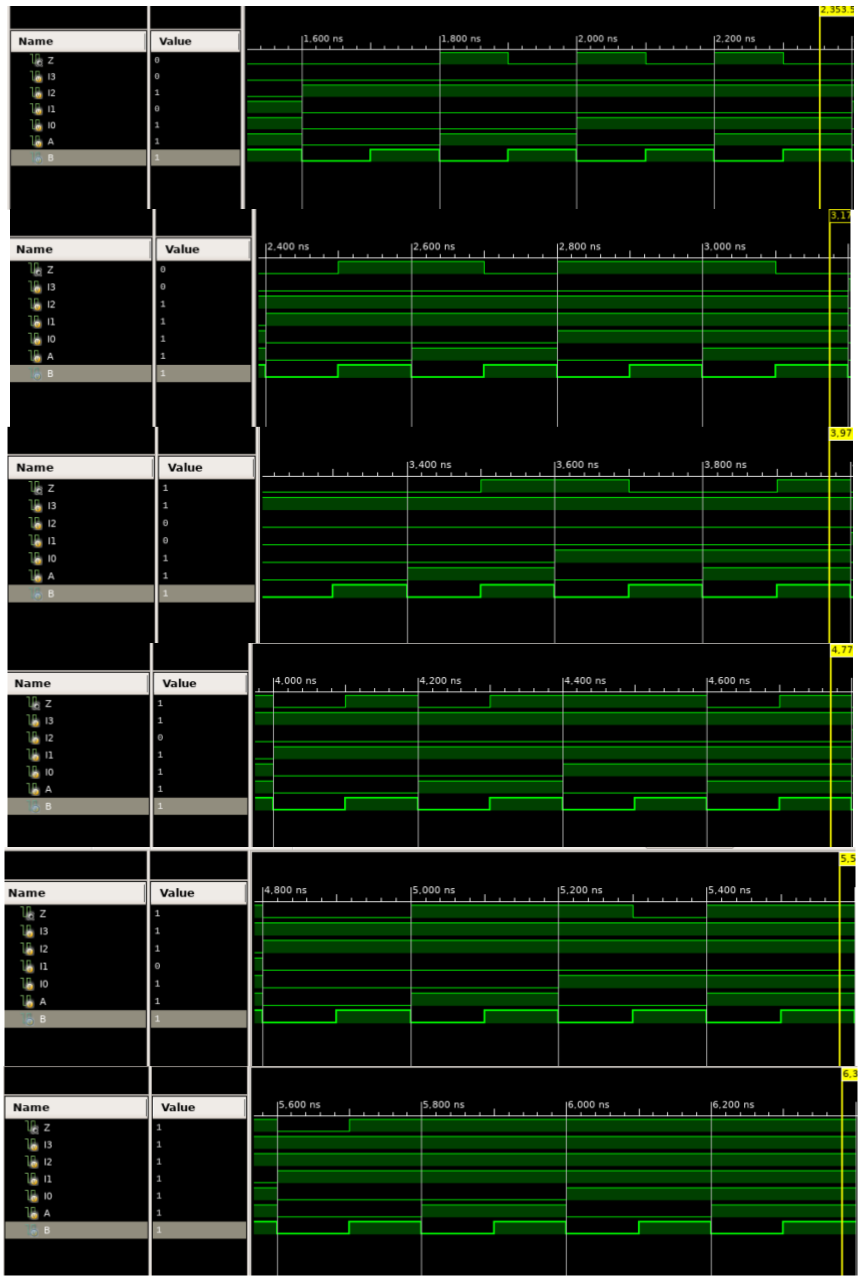
Test code:

```verilog
module mux4to1_test_c;

   // Inputs
   reg I3;
   reg I2;
   reg I1;
   reg I0;
   reg A;
   reg B;

   // Outputs
   wire Z;

   // Instantiate the Unit Under Test (UUT)
   mux4to1_c uut (
      .I3(I3),
      .I2(I2),
      .I1(I1),
      .I0(I0),
      .A(A),
      .B(B),
      .Z(Z)
   );


 initial begin
   I3 = 0; I2 = 0; I1=0; I0 = 0; A = 0; B = 0;#100;
   I3 = 0; I2 = 0; I1=0; I0 = 0; A = 0; B = 1;#100;
   I3 = 0; I2 = 0; I1=0; I0 = 0; A = 1; B = 0;#100;
   I3 = 0; I2 = 0; I1=0; I0 = 0; A = 1; B = 1;#100;

   I3 = 0; I2 = 0; I1=0; I0 = 1; A = 0; B = 0;#100;
   I3 = 0; I2 = 0; I1=0; I0 = 1; A = 0; B = 1;#100;
   I3 = 0; I2 = 0; I1=0; I0 = 1; A = 1; B = 0;#100;
   I3 = 0; I2 = 0; I1=0; I0 = 1; A = 1; B = 1;#100;

   I3 = 0; I2 = 0; I1=1; I0 = 0; A = 0; B = 0;#100;
   I3 = 0; I2 = 0; I1=1; I0 = 0; A = 0; B = 1;#100;
   I3 = 0; I2 = 0; I1=1; I0 = 0; A = 1; B = 0;#100;
   I3 = 0; I2 = 0; I1=1; I0 = 0; A = 1; B = 1;#100;

   I3 = 0; I2 = 0; I1=1; I0 = 1; A = 0; B = 0;#100;
   I3 = 0; I2 = 0; I1=1; I0 = 1; A = 0; B = 1;#100;
   I3 = 0; I2 = 0; I1=1; I0 = 1; A = 1; B = 0;#100;
   I3 = 0; I2 = 0; I1=1; I0 = 1; A = 1; B = 1;#100;



   I3 = 0; I2 = 1; I1=0; I0 = 0; A = 0; B = 0;#100;
   I3 = 0; I2 = 1; I1=0; I0 = 0; A = 0; B = 1;#100;
   I3 = 0; I2 = 1; I1=0; I0 = 0; A = 1; B = 0;#100;
   I3 = 0; I2 = 1; I1=0; I0 = 0; A = 1; B = 1;#100;

   I3 = 0; I2 = 1; I1=0; I0 = 1; A = 0; B = 0;#100;
   I3 = 0; I2 = 1; I1=0; I0 = 1; A = 0; B = 1;#100;
   I3 = 0; I2 = 1; I1=0; I0 = 1; A = 1; B = 0;#100;
   I3 = 0; I2 = 1; I1=0; I0 = 1; A = 1; B = 1;#100;

   I3 = 0; I2 = 1; I1=1; I0 = 0; A = 0; B = 0;#100;
   I3 = 0; I2 = 1; I1=1; I0 = 0; A = 0; B = 1;#100;
   I3 = 0; I2 = 1; I1=1; I0 = 0; A = 1; B = 0;#100;
   I3 = 0; I2 = 1; I1=1; I0 = 0; A = 1; B = 1;#100;

   I3 = 1; I2 = 0; I1=0; I0 = 0; A = 0; B = 0;#100;
   I3 = 1; I2 = 0; I1=0; I0 = 0; A = 0; B = 1;#100;
   I3 = 1; I2 = 0; I1=0; I0 = 0; A = 1; B = 0;#100;
   I3 = 1; I2 = 0; I1=0; I0 = 0; A = 1; B = 1;#100;

   I3 = 1; I2 = 0; I1=0; I0 = 1; A = 0; B = 0;#100;
   I3 = 1; I2 = 0; I1=0; I0 = 1; A = 0; B = 1;#100;
   I3 = 1; I2 = 0; I1=0; I0 = 1; A = 1; B = 0;#100;
   I3 = 1; I2 = 0; I1=0; I0 = 1; A = 1; B = 1;#100;

   I3 = 1; I2 = 0; I1=1; I0 = 0; A = 0; B = 0;#100;
   I3 = 1; I2 = 0; I1=1; I0 = 0; A = 0; B = 1;#100;
   I3 = 1; I2 = 0; I1=1; I0 = 0; A = 1; B = 0;#100;
   I3 = 1; I2 = 0; I1=1; I0 = 0; A = 1; B = 1;#100;

   I3 = 1; I2 = 0; I1=1; I0 = 1; A = 0; B = 0;#100;
   I3 = 1; I2 = 0; I1=1; I0 = 1; A = 0; B = 1;#100;
   I3 = 1; I2 = 0; I1=1; I0 = 1; A = 1; B = 0;#100;
   I3 = 1; I2 = 0; I1=1; I0 = 1; A = 1; B = 1;#100;



   I3 = 1; I2 = 1; I1=0; I0 = 0; A = 0; B = 0;#100;
   I3 = 1; I2 = 1; I1=0; I0 = 0; A = 0; B = 1;#100;
   I3 = 1; I2 = 1; I1=0; I0 = 0; A = 1; B = 0;#100;
   I3 = 1; I2 = 1; I1=0; I0 = 0; A = 1; B = 1;#100;

   I3 = 1; I2 = 1; I1=0; I0 = 1; A = 0; B = 0;#100;
   I3 = 1; I2 = 1; I1=0; I0 = 1; A = 0; B = 1;#100;
   I3 = 1; I2 = 1; I1=0; I0 = 1; A = 1; B = 0;#100;
   I3 = 1; I2 = 1; I1=0; I0 = 1; A = 1; B = 1;#100;

   I3 = 1; I2 = 1; I1=1; I0 = 0; A = 0; B = 0;#100;
   I3 = 1; I2 = 1; I1=1; I0 = 0; A = 0; B = 1;#100;
   I3 = 1; I2 = 1; I1=1; I0 = 0; A = 1; B = 0;#100;
   I3 = 1; I2 = 1; I1=1; I0 = 0; A = 1; B = 1;#100;

   I3 = 1; I2 = 1; I1=1; I0 = 1; A = 0; B = 0;#100;
   I3 = 1; I2 = 1; I1=1; I0 = 1; A = 0; B = 1;#100;
   I3 = 1; I2 = 1; I1=1; I0 = 1; A = 1; B = 0;#100;
   I3 = 1; I2 = 1; I1=1; I0 = 1; A = 1; B = 1;#100;
 end

endmodule
```

Result:

In all 3 descriptions, the result was identical, as follows:





We can check that for any combinations of I3, I2, I1 and I0, when A is 0 and B is 0 the output is I0, when A is 0 and B is 1 the output is I1, when A is 1 and B is 0 the output is I2 and when A is 1 and B is 1 the output is I3, which is the desired behavior for the 4-to-1 mux.

(3) Discussion

(i) Structural description

Structural description corresponds to the actual schematic with the same function, meaning that you can use logic gates, functioning the same as the ones in the schematic. So it is very intuitive, because you can look at the schematic and implement it in hardware description language directly. However, the code will get too big if it is a complicated schematic, where there are a lot of gates.

(ii) Data-flow description

It describes the flow of data from input to output, using operators. The statements are executed concurrently. It can be useful when the design involves complex logic operations that involve many gates, because we can simply describe the output for each of the input combinations. And they are easier to understand, because we can easily expect what the output will be for each input. However, they can be harder to understand for complex designs where states are involved.

(iii) Behavioral description
Like other programming languages, statements in behavioral description are executed sequentially. They are divided into ‘initial’ and ‘always’, the former executed only once and the latter executed repeatedly with respect to changes of the variables in the sensitivity list. With behavioral description it’s easier to describe state, so it’s useful for describing sequential logic. However, for simple circuits, using structural or data-flow description instead can result in shorter code.

(4) Verilog Grammar
There are blocking statements and non-blocking statements in Verilog. The former is done by using “=” and the latter by using “<=”. Blocking statements are executed in order, from top to bottom. Conversely, non-blocking statements are executed concurrently, without specific order. The non-blocking statements exist because of Verilog is describing hardware. In sequential logic, where there is state, the output may be different for each clock edge. Think of a circuit with 2 outputs A and B, where A and B have opposite values and their values are toggled each clock edge. Using the blocking statements “A=B;” and “B=A;” does not give the desired result: A and B will have the same value. Using the non-blocking statements “A<=B;” and “B<=A;” gives the desired result and describes the behavior of the hardware well. However, in combinational logic, implementing (A or B) and C for example, “A or B” should be executed before “(A or B) and C”, because the latter uses the output of the former. So in digital circuits, assignments that are done concurrently and ones that are done in order are both needed, hence the existence of both blocking and non-blocking statements.