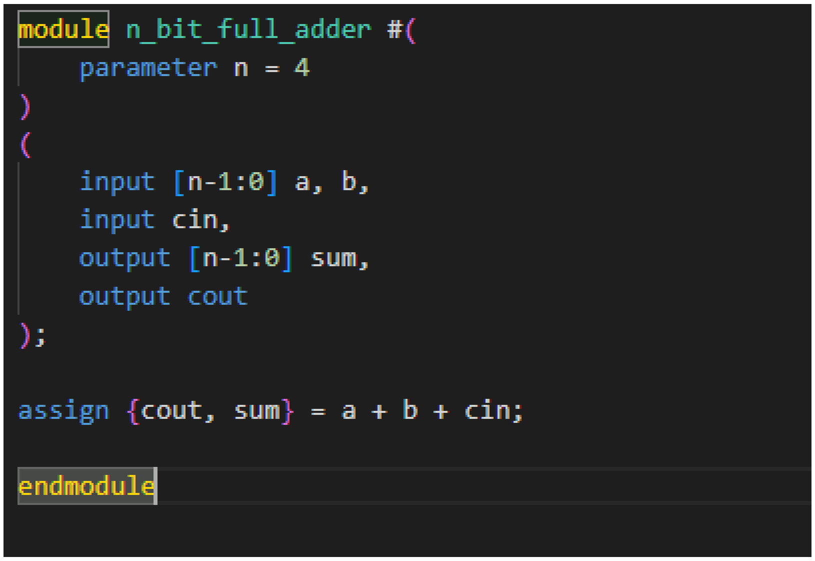
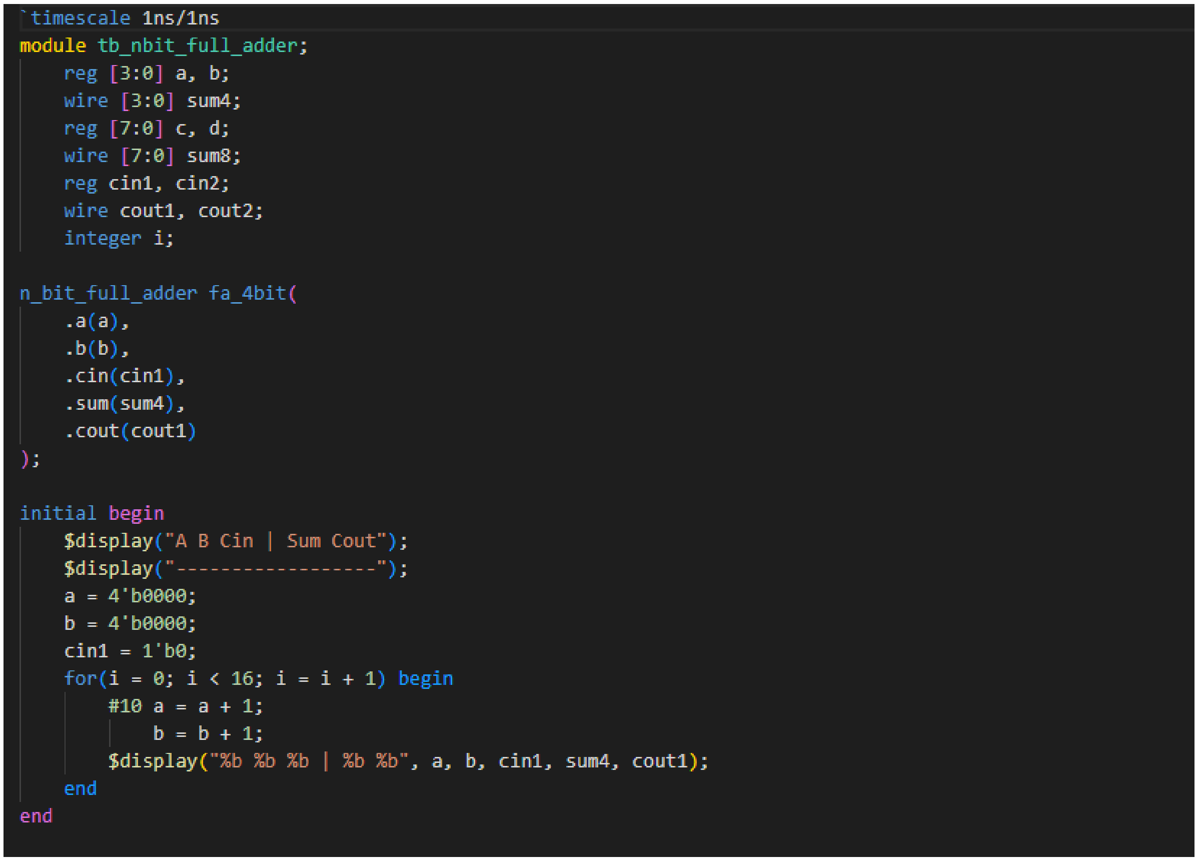
1. Full Adder n-bit:

- Code:

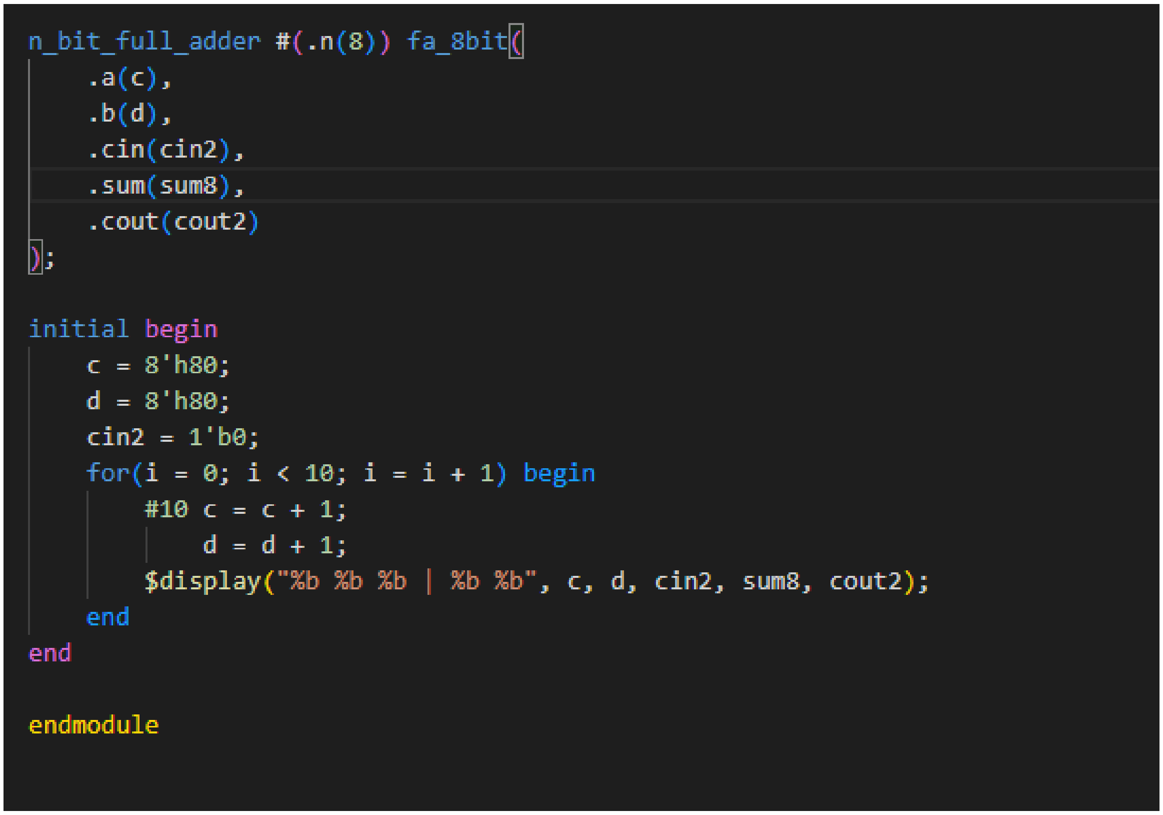


```
module n_bit_full_adder #(
    parameter n = 4
)
(
    input [n-1:0] a, b,
    input cin,
    output [n-1:0] sum,
    output cout
);

assign {cout, sum} = a + b + cin;

endmodule
```

- Testbench:

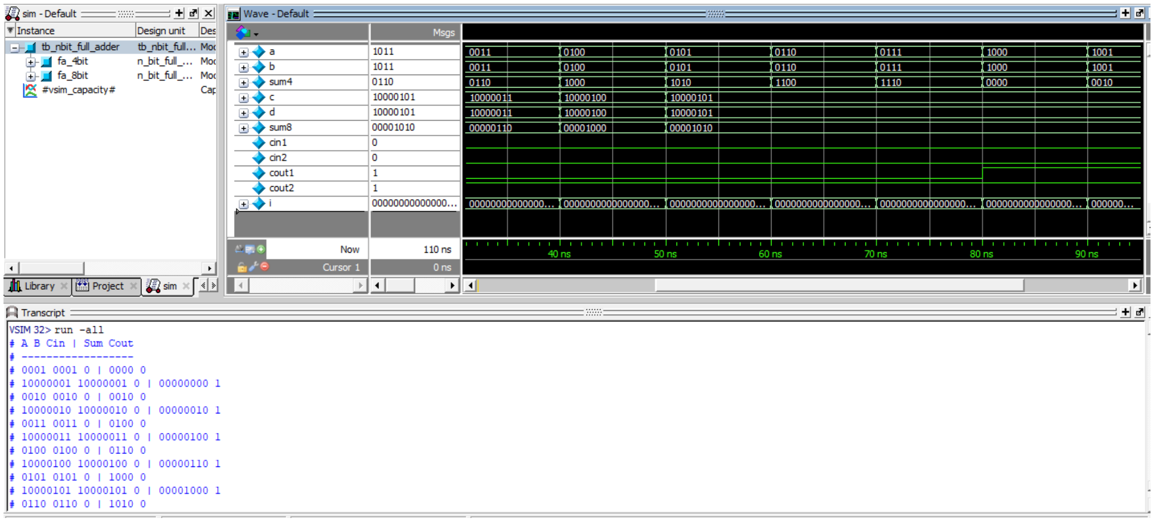


```
`timescale 1ns/1ns
module tb_nbit_full_adder;
    reg [3:0] a, b;
    wire [3:0] sum4;
    reg [7:0] c, d;
    wire [7:0] sum8;
    reg cin1, cin2;
    wire cout1, cout2;
    integer i;

n_bit_full_adder fa_4bit(
    .a(a),
    .b(b),
    .cin(cin1),
    .sum(sum4),
    .cout(cout1)
);

initial begin
    $display("A B Cin | Sum Cout");
    $display("------------------");
    a = 4'b0000;
    b = 4'b0000;
    cin1 = 1'b0;
    for(i = 0; i < 16; i = i + 1) begin
        #10 a = a + 1;
            b = b + 1;
        $display("%b %b %b | %b %b", a, b, cin1, sum4, cout1);
    end
end
```



```
n_bit_full_adder #(.n(8)) fa_8bit(
    .a(c),
    .b(d),
    .cin(cin2),
    .sum(sum8),
    .cout(cout2)
);

initial begin
    c = 8'h80;
    d = 8'h80;
    cin2 = 1'b0;
    for(i = 0; i < 10; i = i + 1) begin
        #10 c = c + 1;
            d = d + 1;
        $display("%b %b %b | %b %b", c, d, cin2, sum8, cout2);
    end
end

endmodule
```

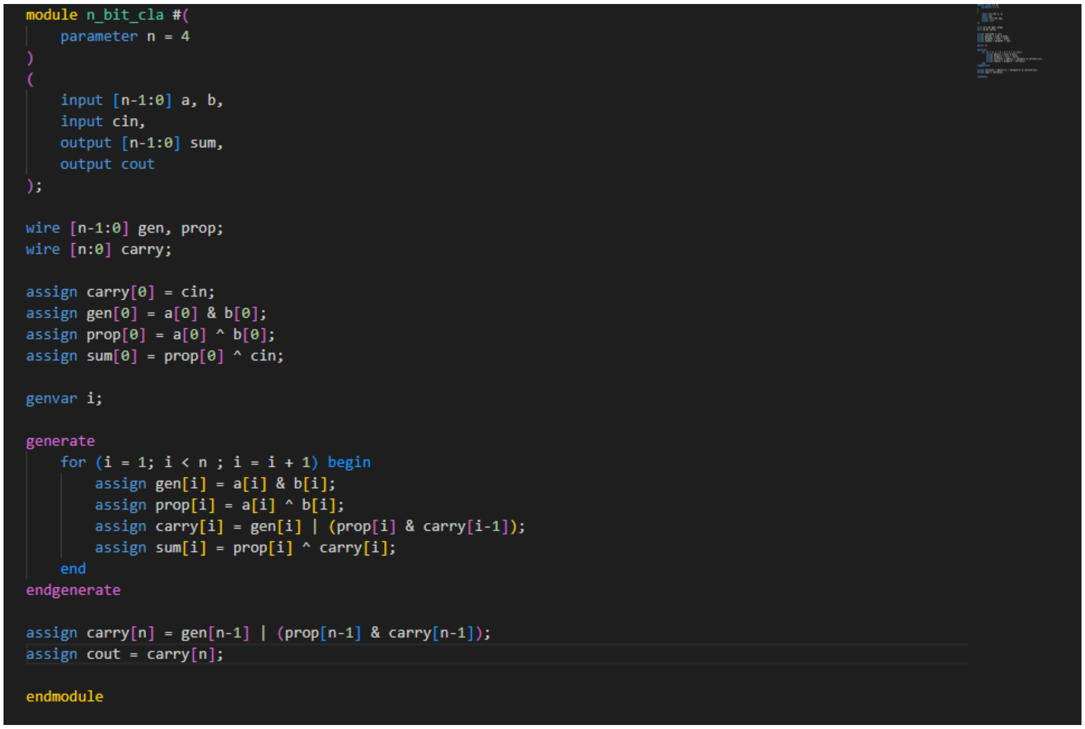
- Simulation and transcript:



```
VSIM 32> run -all
# A B Cin | Sum Cout
# ------------------
# 0001 0001 0 | 0000 0
# 10000001 10000001 0 | 00000000 1
# 0010 0010 0 | 0010 0
# 10000010 10000010 0 | 00000010 1
# 0011 0011 0 | 0100 0
# 10000011 10000011 0 | 00000100 1
# 0100 0100 0 | 0110 0
# 10000100 10000100 0 | 00000110 1
# 0101 0101 0 | 1000 0
# 10000101 10000101 0 | 00001000 1
# 0110 0110 0 | 1010 0
```

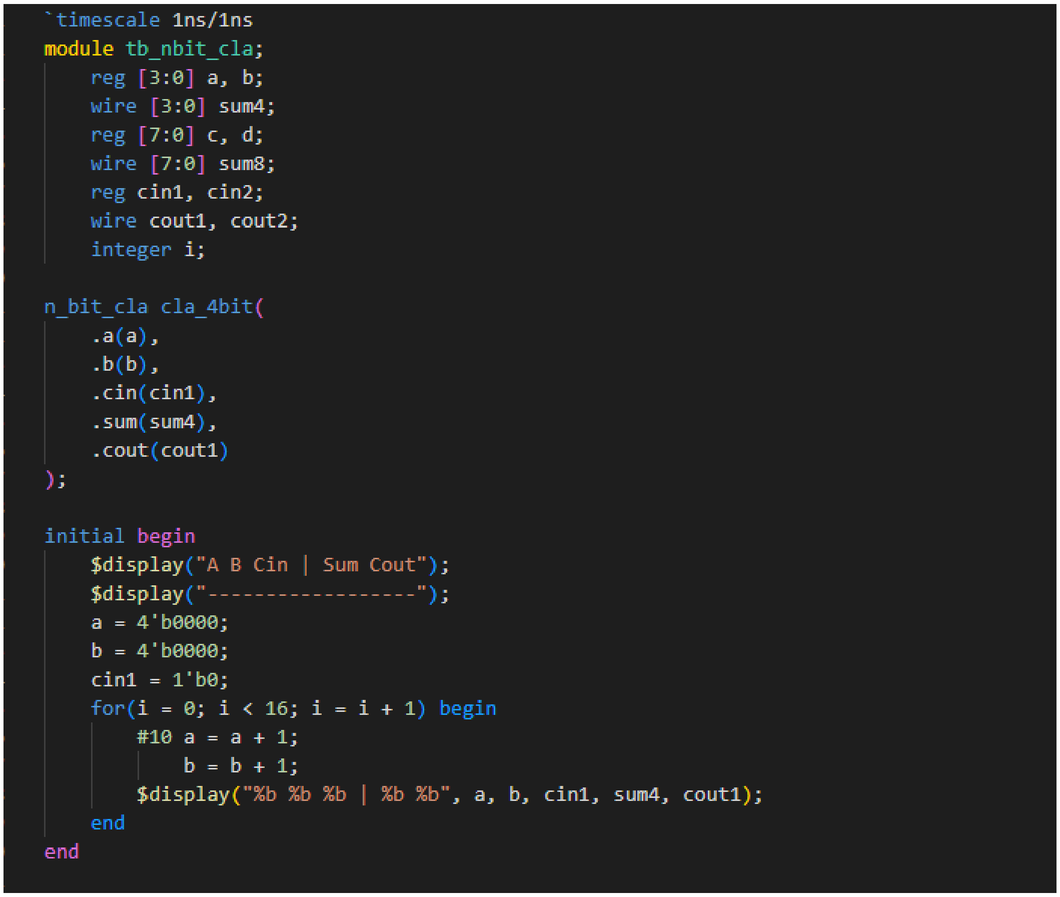
2. Carry Look Ahead Adder n-bit:

- Code

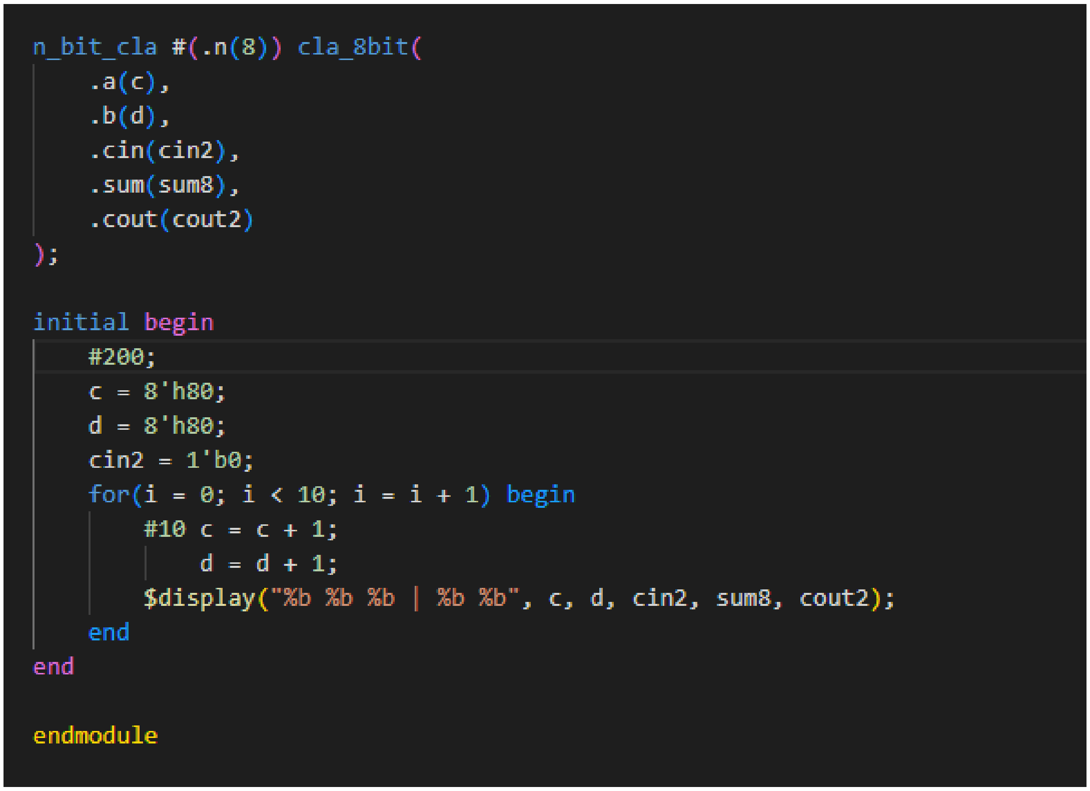


```verilog
module n_bit_cla #(
    parameter n = 4
)
(
    input [n-1:0] a, b,
    input cin,
    output [n-1:0] sum,
    output cout
);

wire [n-1:0] gen, prop;
wire [n:0] carry;

assign carry[0] = cin;
assign gen[0] = a[0] & b[0];
assign prop[0] = a[0] ^ b[0];
assign sum[0] = prop[0] ^ cin;

genvar i;

generate
    for (i = 1; i < n ; i = i + 1) begin
        assign gen[i] = a[i] & b[i];
        assign prop[i] = a[i] ^ b[i];
        assign carry[i] = gen[i] | (prop[i] & carry[i-1]);
        assign sum[i] = prop[i] ^ carry[i];
    end
endgenerate

assign carry[n] = gen[n-1] | (prop[n-1] & carry[n-1]);
assign cout = carry[n];

endmodule
```

- Testbench:

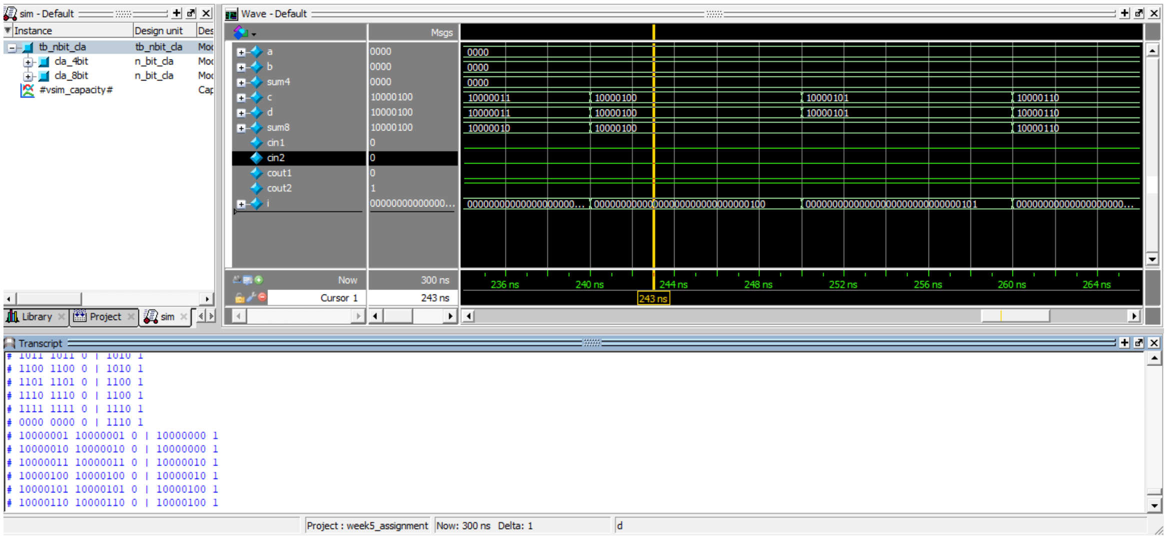


```verilog
`timescale 1ns/1ns
module tb_nbit_cla;
    reg [3:0] a, b;
    wire [3:0] sum4;
    reg [7:0] c, d;
    wire [7:0] sum8;
    reg cin1, cin2;
    wire cout1, cout2;
    integer i;

n_bit_cla cla_4bit(
    .a(a),
    .b(b),
    .cin(cin1),
    .sum(sum4),
    .cout(cout1)
);

initial begin
    $display("A B Cin | Sum Cout");
    $display("------------------");
    a = 4'b0000;
    b = 4'b0000;
    cin1 = 1'b0;
    for(i = 0; i < 16; i = i + 1) begin
        #10 a = a + 1;
            b = b + 1;
        $display("%b %b %b | %b %b", a, b, cin1, sum4, cout1);
    end
end
```



```
n_bit_cla #(.n(8)) cla_8bit(
    .a(c),
    .b(d),
    .cin(cin2),
    .sum(sum8),
    .cout(cout2)
);

initial begin
    #200;
    c = 8'h80;
    d = 8'h80;
    cin2 = 1'b0;
    for(i = 0; i < 10; i = i + 1) begin
        #10 c = c + 1;
            d = d + 1;
        $display("%b %b %b | %b %b", c, d, cin2, sum8, cout2);
    end
end

endmodule
```

- Simulation and transcript:



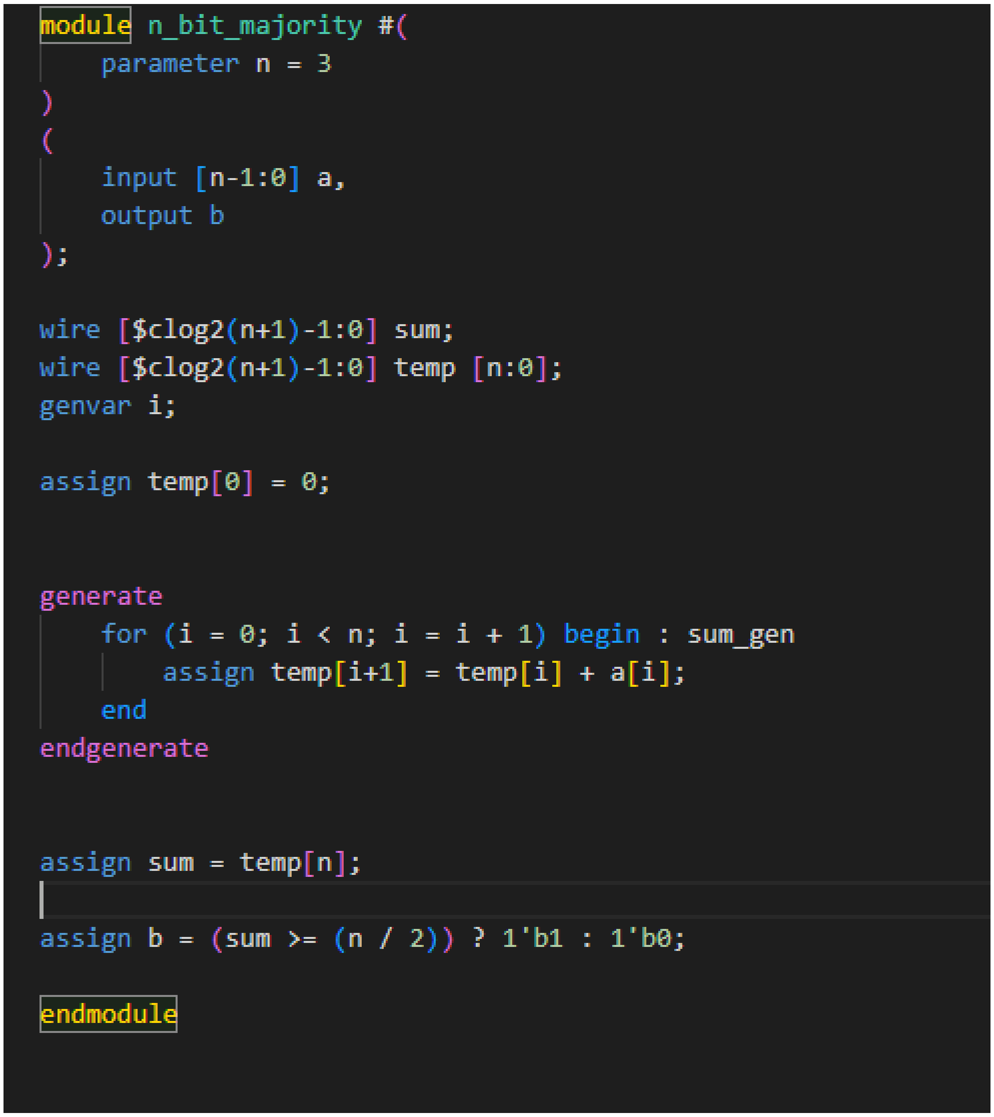
	Msgs
a	0000
b	0000
sum4	0000
c	10000100
d	10000100
sum8	10000100
cin1	0
cin2	0
cout1	0
cout2	1
i	00000000000000...

```
# 1011 1011 0 | 1010 1
# 1100 1100 0 | 1010 1
# 1101 1101 0 | 1100 1
# 1110 1110 0 | 1100 1
# 1111 1111 0 | 1110 1
# 0000 0000 0 | 1110 1
# 10000001 10000001 0 | 10000000 1
# 10000010 10000010 0 | 10000000 1
# 10000011 10000011 0 | 10000010 1
# 10000100 10000100 0 | 10000010 1
# 10000101 10000101 0 | 10000100 1
# 10000110 10000110 0 | 10000100 1
```

Project : week5_assignment   Now: 300 ns Delta: 1

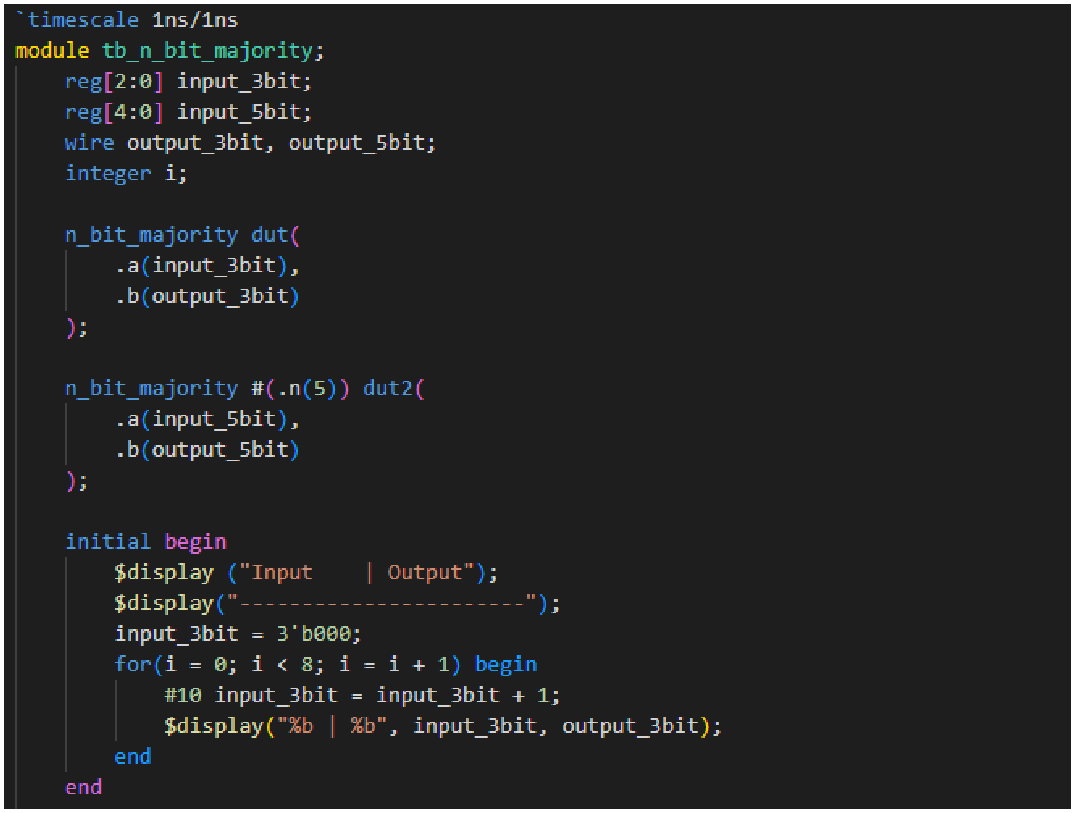
3. Majority n-bit

- Code:

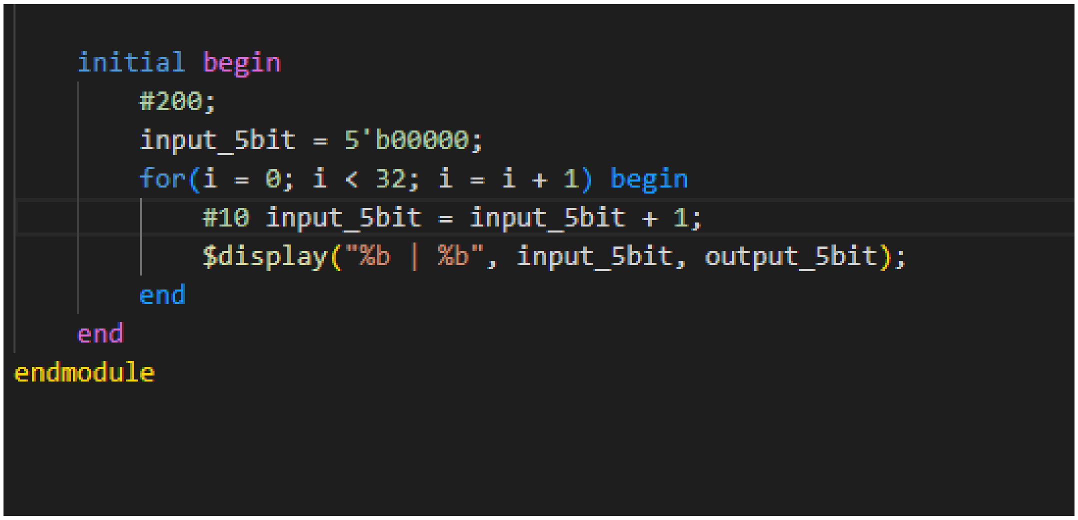


```
module n_bit_majority #(
    parameter n = 3
)
(
    input [n-1:0] a,
    output b
);

wire [$clog2(n+1)-1:0] sum;
wire [$clog2(n+1)-1:0] temp [n:0];
genvar i;

assign temp[0] = 0;


generate
    for (i = 0; i < n; i = i + 1) begin : sum_gen
        assign temp[i+1] = temp[i] + a[i];
    end
endgenerate


assign sum = temp[n];

assign b = (sum >= (n / 2)) ? 1'b1 : 1'b0;

endmodule
```

- Testbench:

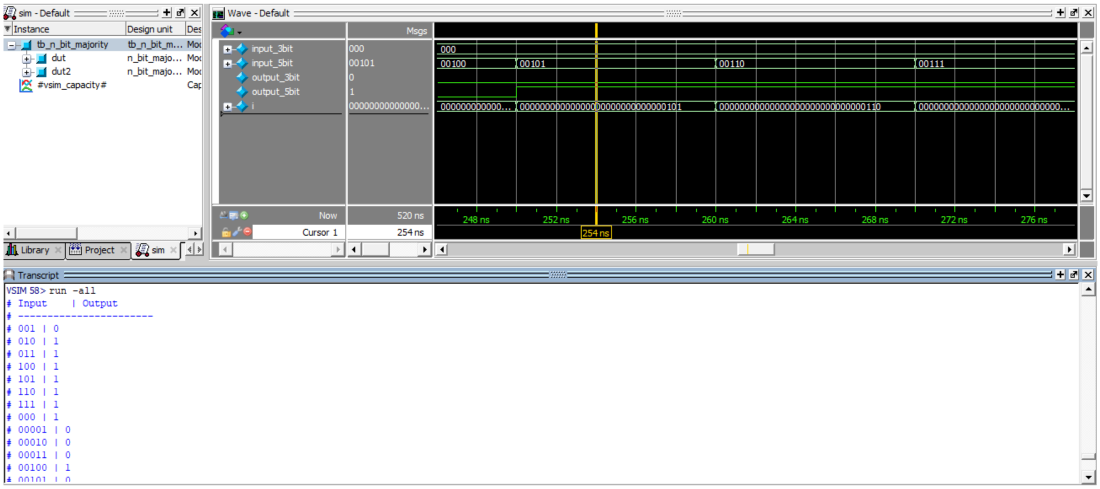


```
`timescale 1ns/1ns
module tb_n_bit_majority;
    reg[2:0] input_3bit;
    reg[4:0] input_5bit;
    wire output_3bit, output_5bit;
    integer i;

    n_bit_majority dut(
        .a(input_3bit),
        .b(output_3bit)
    );

    n_bit_majority #(.n(5)) dut2(
        .a(input_5bit),
        .b(output_5bit)
    );

    initial begin
        $display ("Input    | Output");
        $display("-----------------------");
        input_3bit = 3'b000;
        for(i = 0; i < 8; i = i + 1) begin
            #10 input_3bit = input_3bit + 1;
            $display("%b | %b", input_3bit, output_3bit);
        end
    end
```



```
    initial begin
        #200;
        input_5bit = 5'b00000;
        for(i = 0; i < 32; i = i + 1) begin
            #10 input_5bit = input_5bit + 1;
            $display("%b | %b", input_5bit, output_5bit);
        end
    end
endmodule
```

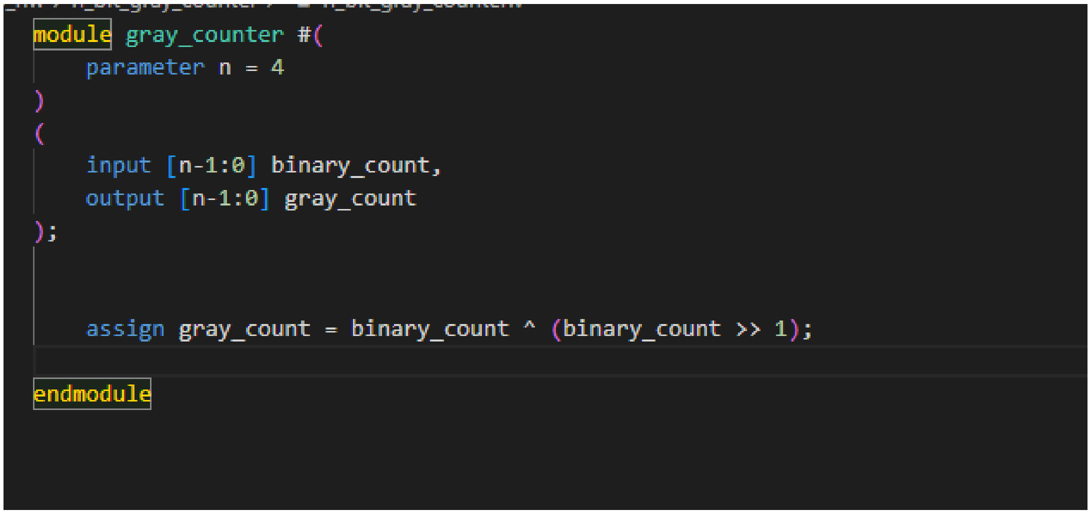
- Simulation and transcript:



```
VSIM 58> run -all
# Input    | Output
# -----------------------
# 001 | 0
# 010 | 1
# 011 | 1
# 100 | 1
# 101 | 1
# 110 | 1
# 111 | 1
# 000 | 1
# 00001 | 0
# 00010 | 0
# 00011 | 0
# 00100 | 1
# 00101 | 0
```

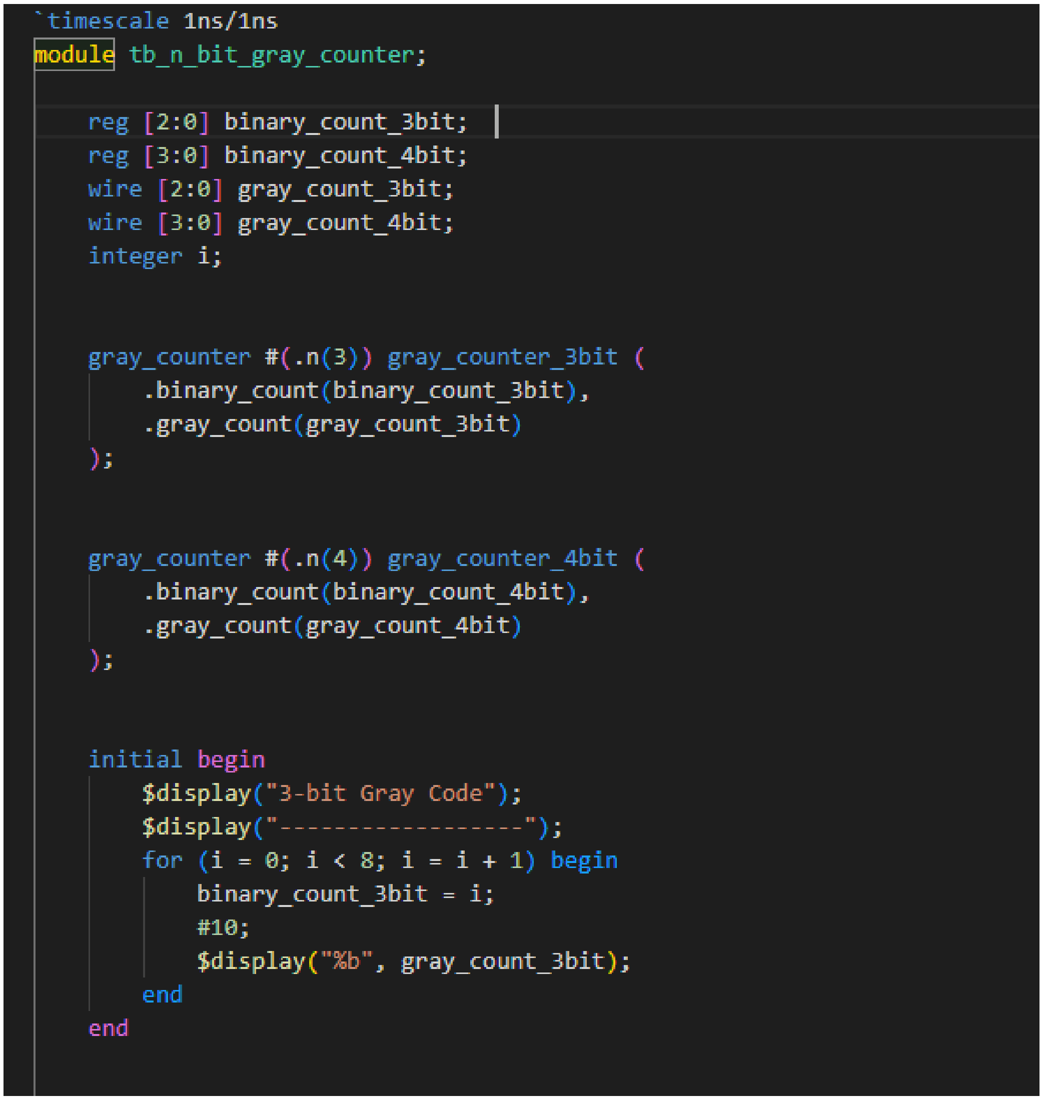
4. Gray counter n-bit

- Code:

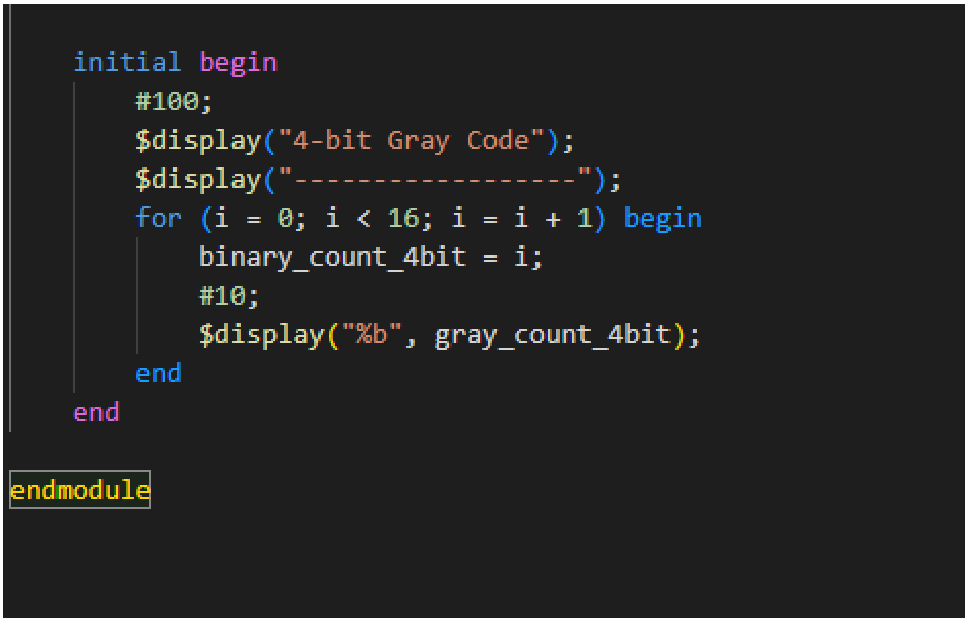


```
module gray_counter #(
    parameter n = 4
)
(
    input [n-1:0] binary_count,
    output [n-1:0] gray_count
);


    assign gray_count = binary_count ^ (binary_count >> 1);

endmodule
```

- Testbench:

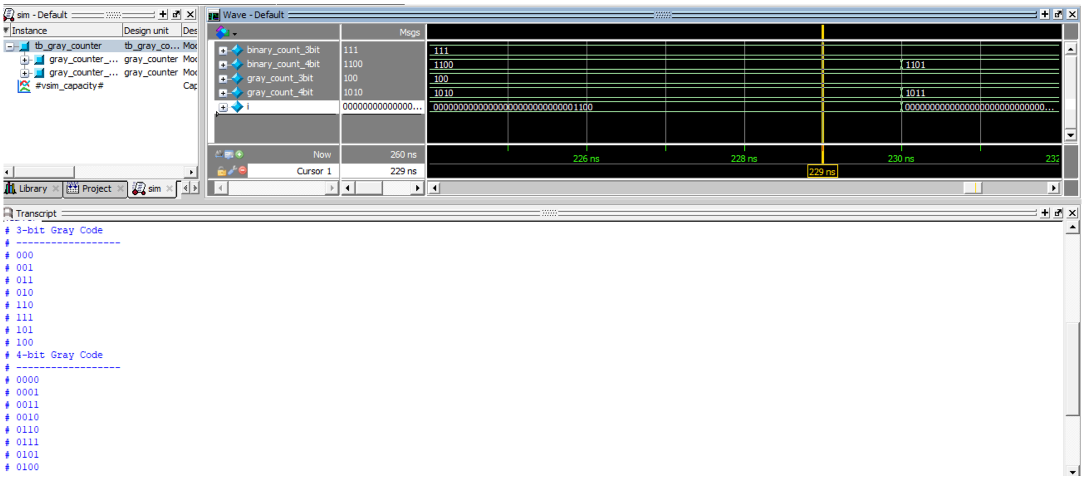


```
`timescale 1ns/1ns
module tb_n_bit_gray_counter;

    reg [2:0] binary_count_3bit;
    reg [3:0] binary_count_4bit;
    wire [2:0] gray_count_3bit;
    wire [3:0] gray_count_4bit;
    integer i;


    gray_counter #(.n(3)) gray_counter_3bit (
        .binary_count(binary_count_3bit),
        .gray_count(gray_count_3bit)
    );


    gray_counter #(.n(4)) gray_counter_4bit (
        .binary_count(binary_count_4bit),
        .gray_count(gray_count_4bit)
    );


    initial begin
        $display("3-bit Gray Code");
        $display("------------------");
        for (i = 0; i < 8; i = i + 1) begin
            binary_count_3bit = i;
            #10;
            $display("%b", gray_count_3bit);
        end
    end
```



```
initial begin
    #100;
    $display("4-bit Gray Code");
    $display("-----------------");
    for (i = 0; i < 16; i = i + 1) begin
        binary_count_4bit = i;
        #10;
        $display("%b", gray_count_4bit);
    end
end

endmodule
```

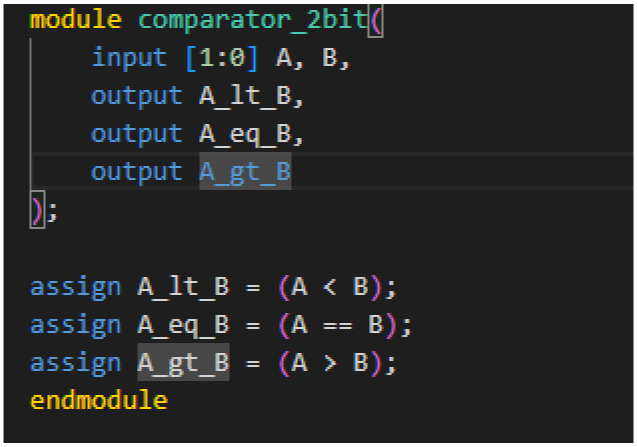
- Simulation and transcript:



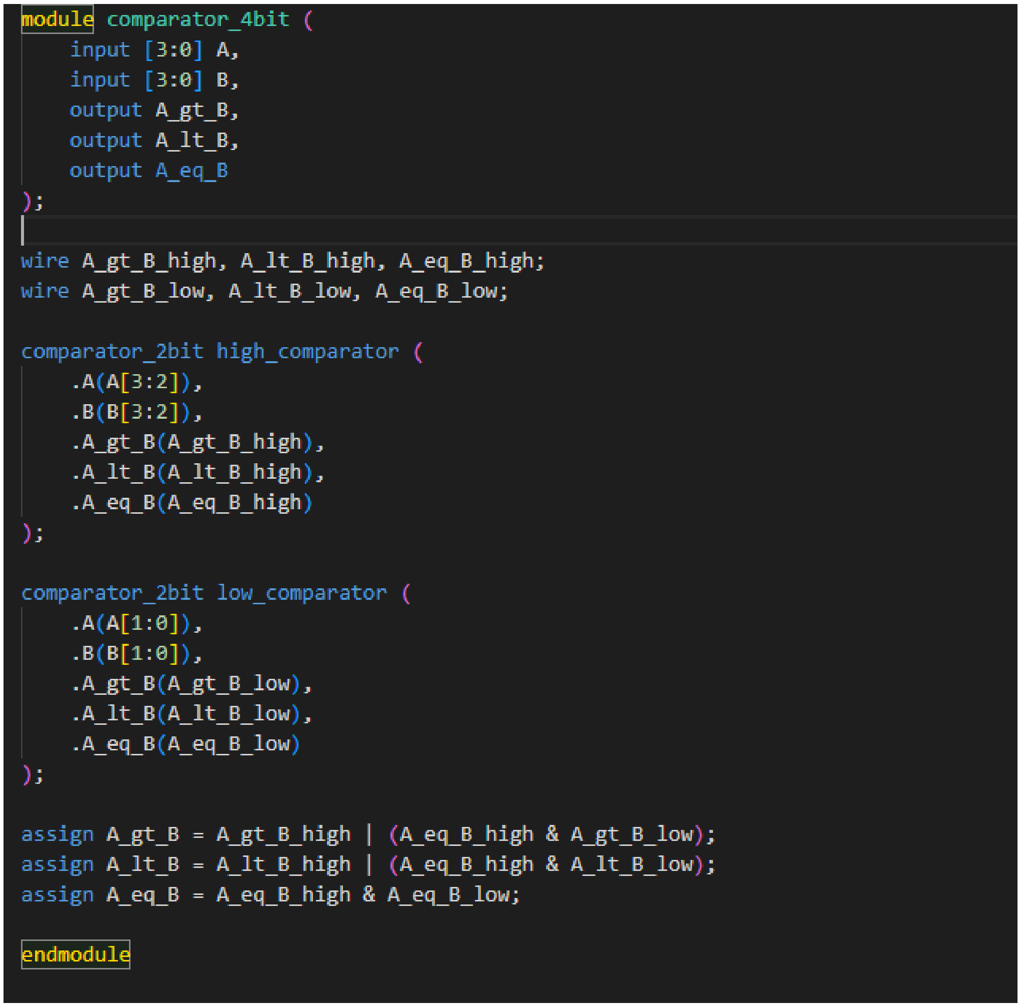
```
# 3-bit Gray Code
# -----------------
# 000
# 001
# 011
# 010
# 110
# 111
# 101
# 100
# 4-bit Gray Code
# -----------------
# 0000
# 0001
# 0011
# 0010
# 0110
# 0111
# 0101
# 0100
```

5. Comparator 4-bit:

- Code

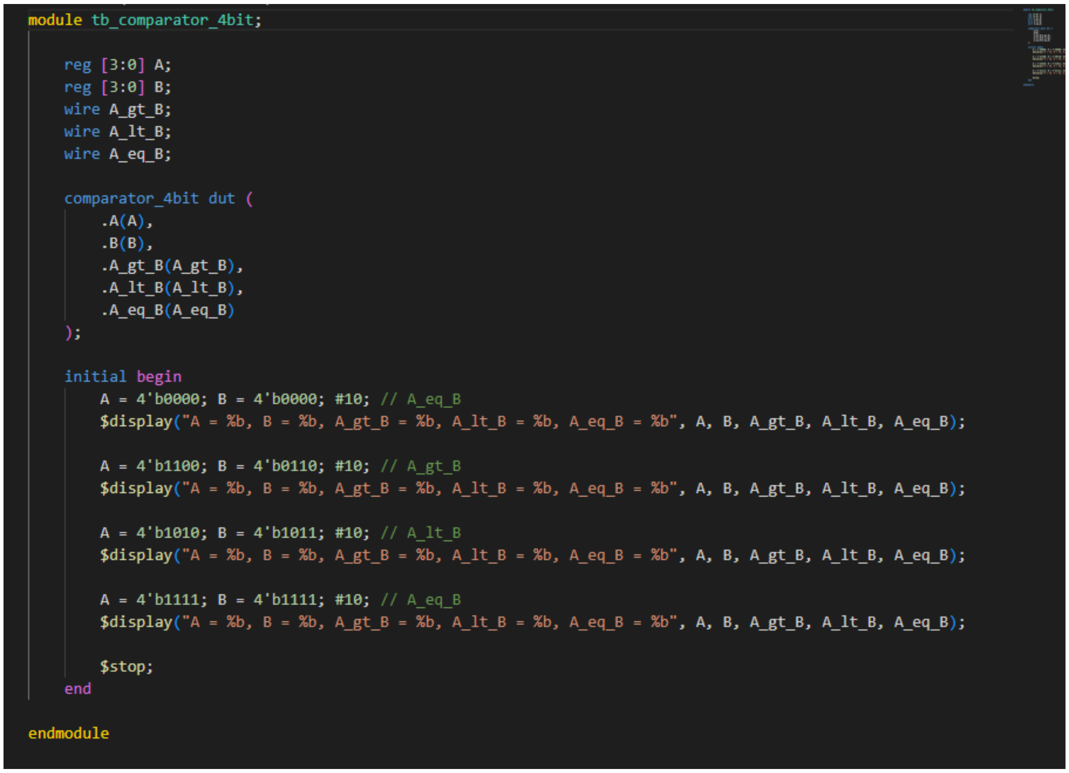


```verilog
module comparator_2bit(
    input [1:0] A, B,
    output A_lt_B,
    output A_eq_B,
    output A_gt_B
);

assign A_lt_B = (A < B);
assign A_eq_B = (A == B);
assign A_gt_B = (A > B);
endmodule
```



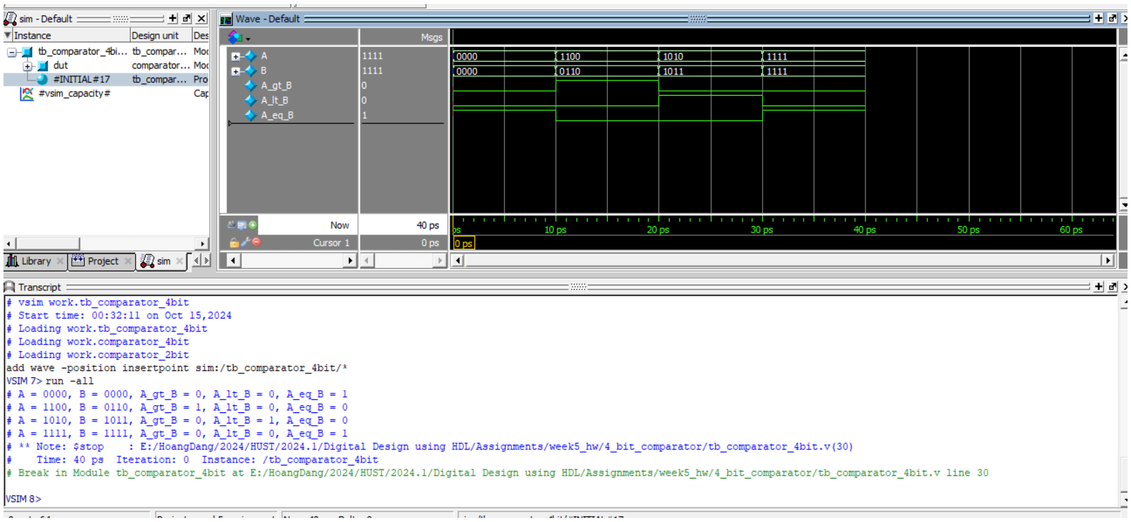
```verilog
module comparator_4bit (
    input [3:0] A,
    input [3:0] B,
    output A_gt_B,
    output A_lt_B,
    output A_eq_B
);

wire A_gt_B_high, A_lt_B_high, A_eq_B_high;
wire A_gt_B_low, A_lt_B_low, A_eq_B_low;

comparator_2bit high_comparator (
    .A(A[3:2]),
    .B(B[3:2]),
    .A_gt_B(A_gt_B_high),
    .A_lt_B(A_lt_B_high),
    .A_eq_B(A_eq_B_high)
);

comparator_2bit low_comparator (
    .A(A[1:0]),
    .B(B[1:0]),
    .A_gt_B(A_gt_B_low),
    .A_lt_B(A_lt_B_low),
    .A_eq_B(A_eq_B_low)
);

assign A_gt_B = A_gt_B_high | (A_eq_B_high & A_gt_B_low);
assign A_lt_B = A_lt_B_high | (A_eq_B_high & A_lt_B_low);
assign A_eq_B = A_eq_B_high & A_eq_B_low;

endmodule
```

- Testbench:



```verilog
module tb_comparator_4bit;

    reg [3:0] A;
    reg [3:0] B;
    wire A_gt_B;
    wire A_lt_B;
    wire A_eq_B;

    comparator_4bit dut (
        .A(A),
        .B(B),
        .A_gt_B(A_gt_B),
        .A_lt_B(A_lt_B),
        .A_eq_B(A_eq_B)
    );

    initial begin
        A = 4'b0000; B = 4'b0000; #10; // A_eq_B
        $display("A = %b, B = %b, A_gt_B = %b, A_lt_B = %b, A_eq_B = %b", A, B, A_gt_B, A_lt_B, A_eq_B);

        A = 4'b1100; B = 4'b0110; #10; // A_gt_B
        $display("A = %b, B = %b, A_gt_B = %b, A_lt_B = %b, A_eq_B = %b", A, B, A_gt_B, A_lt_B, A_eq_B);

        A = 4'b1010; B = 4'b1011; #10; // A_lt_B
        $display("A = %b, B = %b, A_gt_B = %b, A_lt_B = %b, A_eq_B = %b", A, B, A_gt_B, A_lt_B, A_eq_B);

        A = 4'b1111; B = 4'b1111; #10; // A_eq_B
        $display("A = %b, B = %b, A_gt_B = %b, A_lt_B = %b, A_eq_B = %b", A, B, A_gt_B, A_lt_B, A_eq_B);

        $stop;
    end

endmodule
```

- Simulation and transcript:



```
# vsim work.tb_comparator_4bit
# Start time: 00:32:11 on Oct 15,2024
# Loading work.tb_comparator_4bit
# Loading work.comparator_4bit
# Loading work.comparator_2bit
add wave -position insertpoint sim:/tb_comparator_4bit/*
VSIM 7> run -all
# A = 0000, B = 0000, A_gt_B = 0, A_lt_B = 0, A_eq_B = 1
# A = 1100, B = 0110, A_gt_B = 1, A_lt_B = 0, A_eq_B = 0
# A = 1010, B = 1011, A_gt_B = 0, A_lt_B = 1, A_eq_B = 0
# A = 1111, B = 1111, A_gt_B = 0, A_lt_B = 0, A_eq_B = 1
# ** Note: $stop    : E:/HoangDang/2024/HUST/2024.1/Digital Design using HDL/Assignments/week5_hw/4_bit_comparator/tb_comparator_4bit.v(30)
#    Time: 40 ps  Iteration: 0  Instance: /tb_comparator_4bit
# Break in Module tb_comparator_4bit at E:/HoangDang/2024/HUST/2024.1/Digital Design using HDL/Assignments/week5_hw/4_bit_comparator/tb_comparator_4bit.v line 30

VSIM 8>
```