Ex1:

**Verilog code:**

module ripple\_carry\_16\_bit(a, b, cin,sum, cout);

input [15:0] a,b;

input cin;

output [15:0] sum;

output cout;

wire c1,c2,c3;

ripple\_carry\_4\_bit rca1 (

.a(a[3:0]),

.b(b[3:0]),

.cin(cin),

.sum(sum[3:0]),

.cout(c1));

ripple\_carry\_4\_bit rca2(

.a(a[7:4]),

.b(b[7:4]),

.cin(c1),

.sum(sum[7:4]),

.cout(c2));

ripple\_carry\_4\_bit rca3(

.a(a[11:8]),

.b(b[11:8]),

.cin(c2),

.sum(sum[11:8]),

.cout(c3));

ripple\_carry\_4\_bit rca4(

.a(a[15:12]),

.b(b[15:12]),

.cin(c3),

.sum(sum[15:12]),

.cout(cout));

endmodule

////////////////////////////////////

//4-bit Ripple Carry Adder

////////////////////////////////////

module ripple\_carry\_4\_bit(a, b, cin, sum, cout);

input [3:0] a,b;

input cin;

wire c1,c2,c3;

output [3:0] sum;

output cout;

full\_adder fa0(.a(a[0]), .b(b[0]),.cin(cin), .sum(sum[0]),.cout(c1));

full\_adder fa1(.a(a[1]), .b(b[1]), .cin(c1), .sum(sum[1]),.cout(c2));

full\_adder fa2(.a(a[2]), .b(b[2]), .cin(c2), .sum(sum[2]),.cout(c3));

full\_adder fa3(.a(a[3]), .b(b[3]), .cin(c3), .sum(sum[3]),.cout(cout));

endmodule

//////////////////////////////

//1bit Full Adder

/////////////////////////////

module full\_adder(a,b,cin,sum, cout);

input a,b,cin;

output sum, cout;

wire x,y,z;

half\_adder h1(.a(a), .b(b), .sum(x), .cout(y));

half\_adder h2(.a(x), .b(cin), .sum(sum), .cout(z));

or or\_1(cout,z,y);

endmodule

///////////////////////////

// 1 bit Half Adder

//////////////////////////

module half\_adder( a,b, sum, cout );

input a,b;

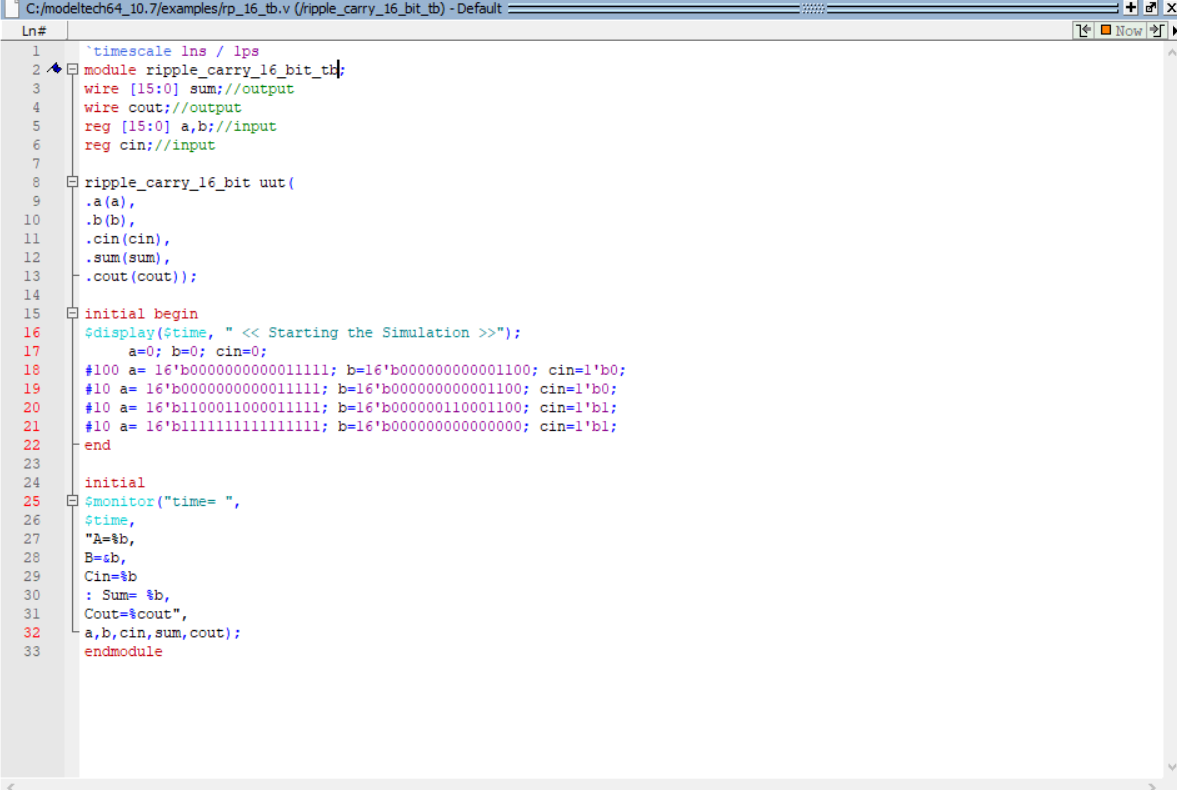
output sum, cout;

xor xor\_1 (sum,a,b);

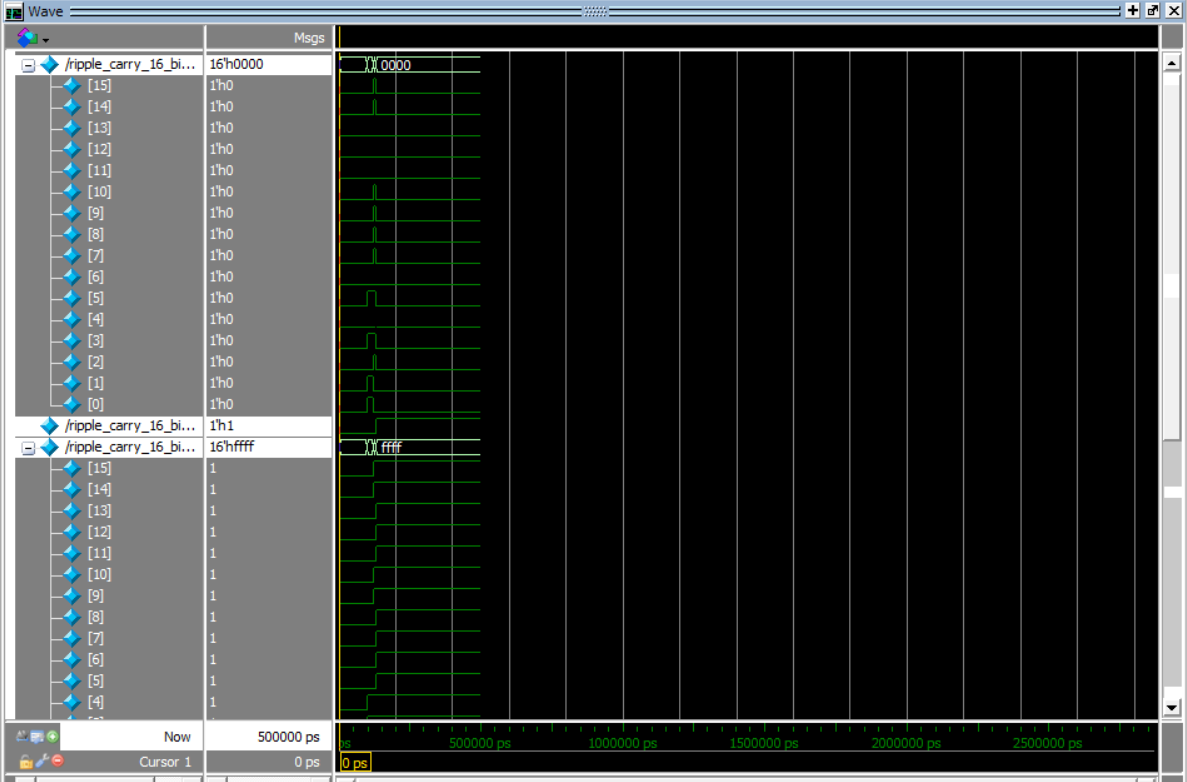
and and\_1 (cout,a,b);

endmodule

**Testbench**



**Waveform**

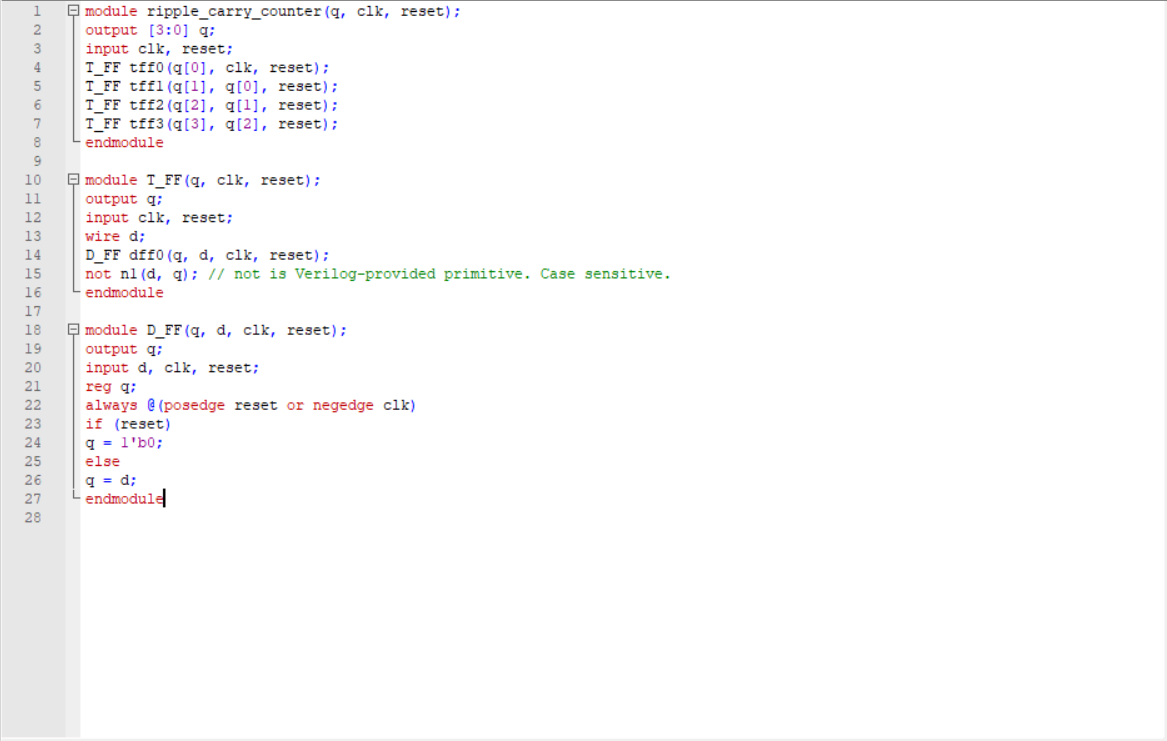


Ex2:

Counter is basically used to count the number of clock pulses applied to a flip flop.

Ripple counter is a cascaded arrangement of flip-flops drives the clock input of the following flip-flop. The number of flip flops in the cascaded arrangement depend on the num off different logic states it goes through before repeating the sequence a parameter known as the modulus of the counter.

**Verilog code**

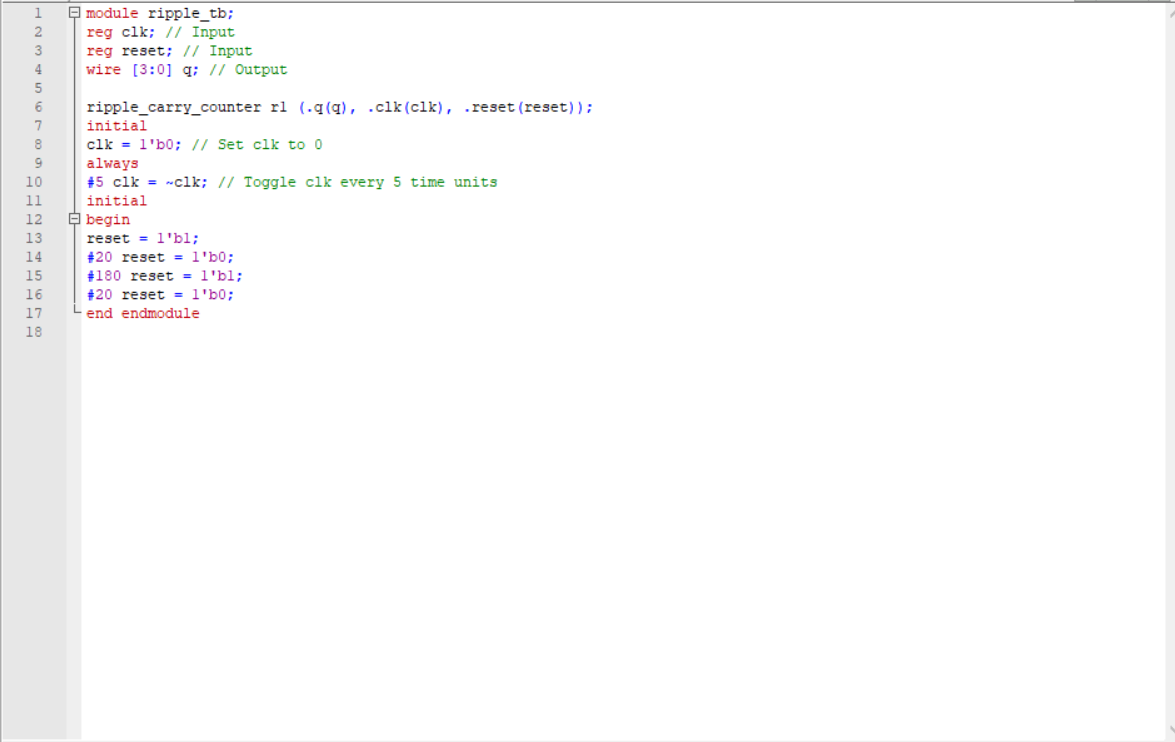


Comments:

//Line 4 to 7: Initiate 4 TFF to update the count of the counter system

// Line 10 to 15: Declare Tff module where tff takes clk and rest as input and q is the output. In here wwe instantiate d flip flop and not gate as t flip flop part

**Testbench**

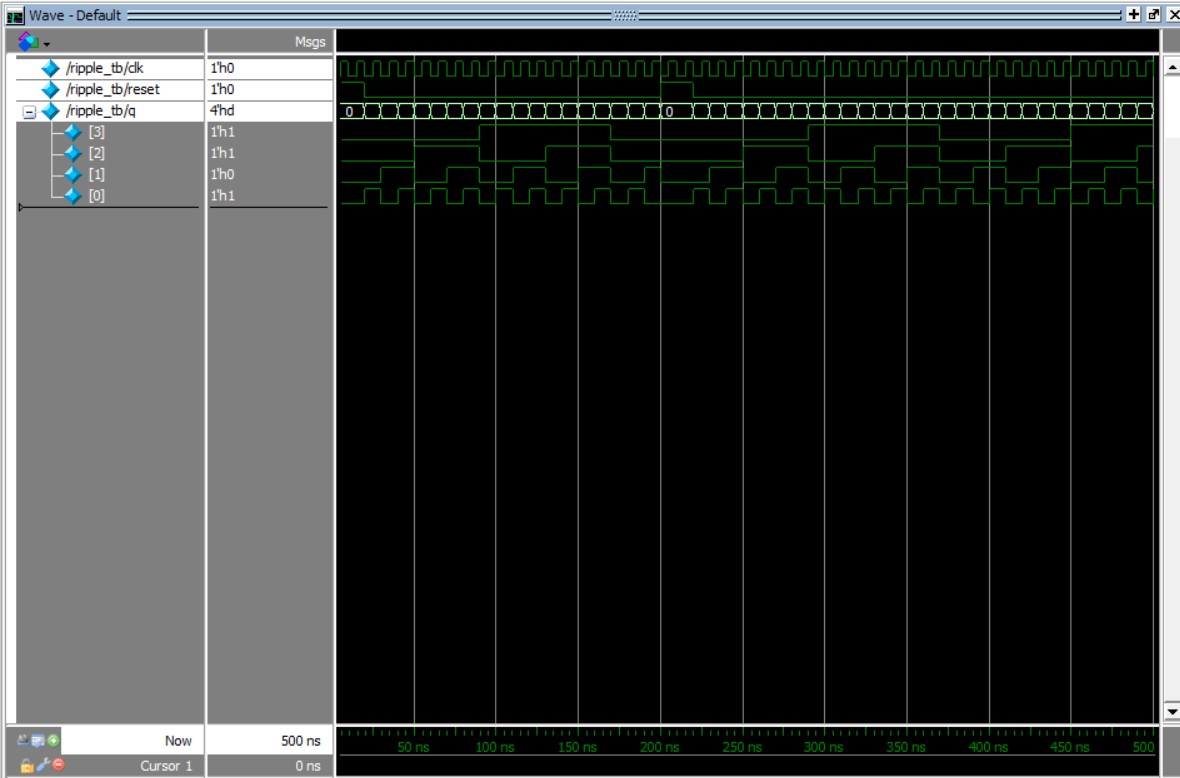


Comment:

// We set the initial clk to be 0 and the step for clock toggle is 5ns

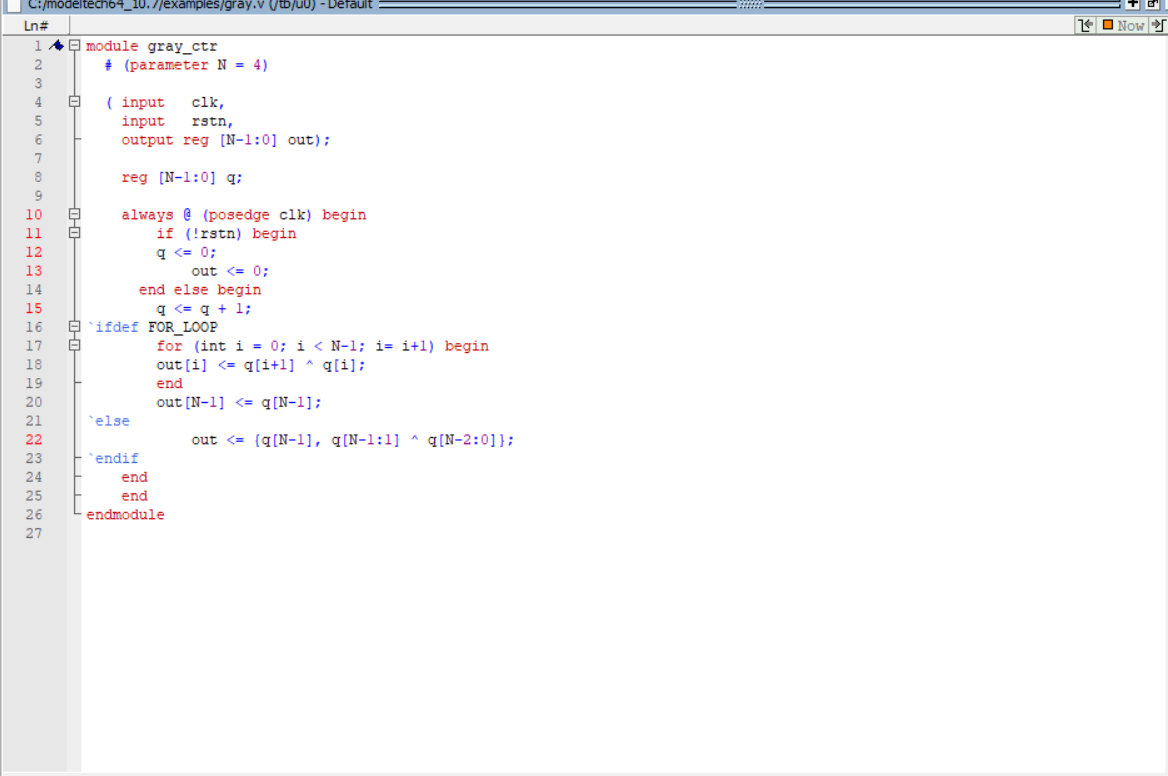
// This we provide reset values as the input where reset =1’b1, continue with re = 0,1,0,…

**Waveform**

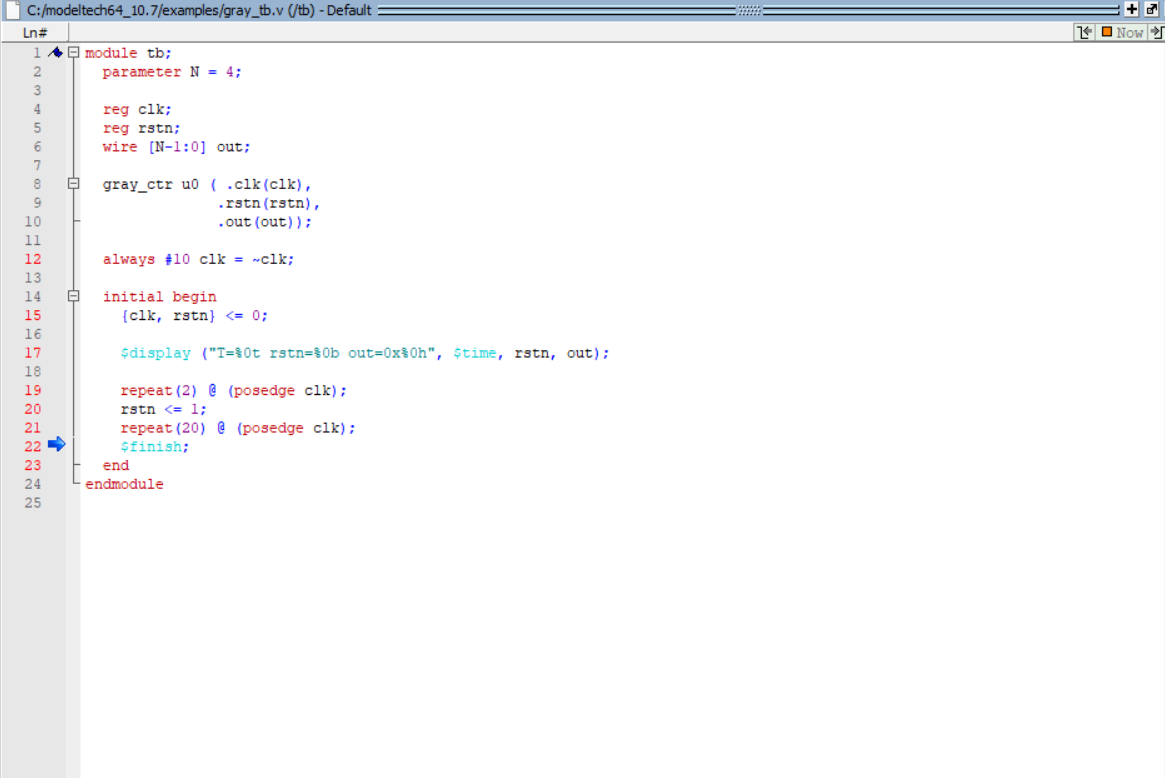


Ex3:

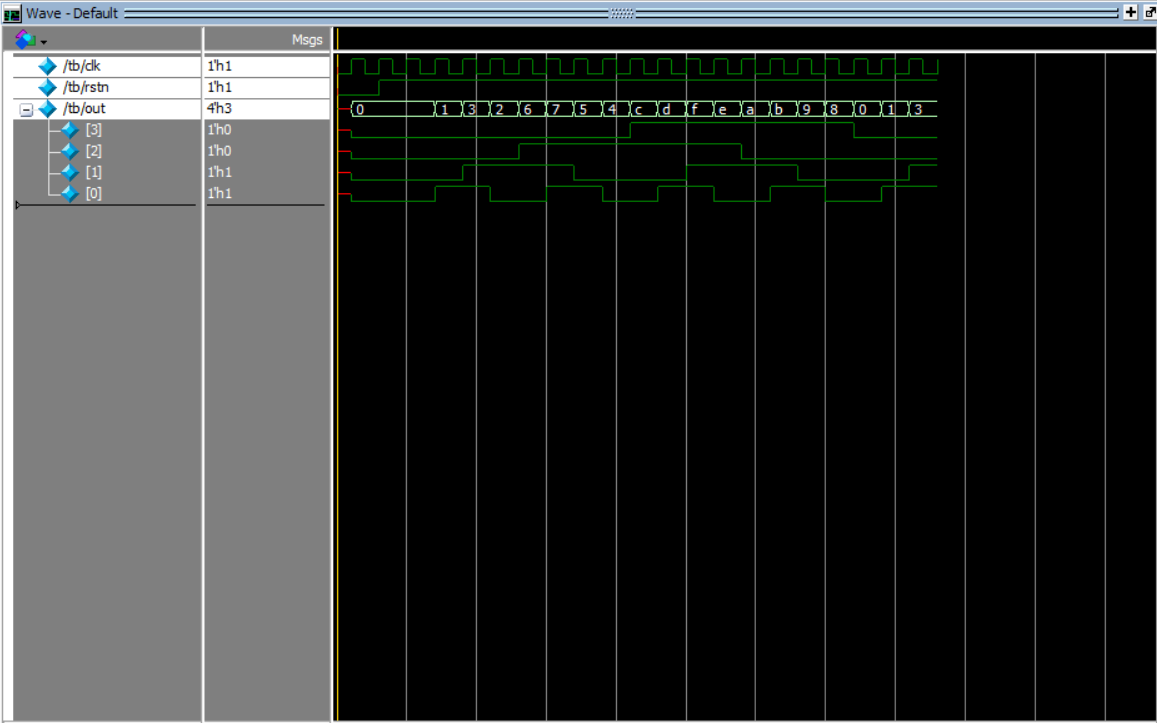
**Verilog code**

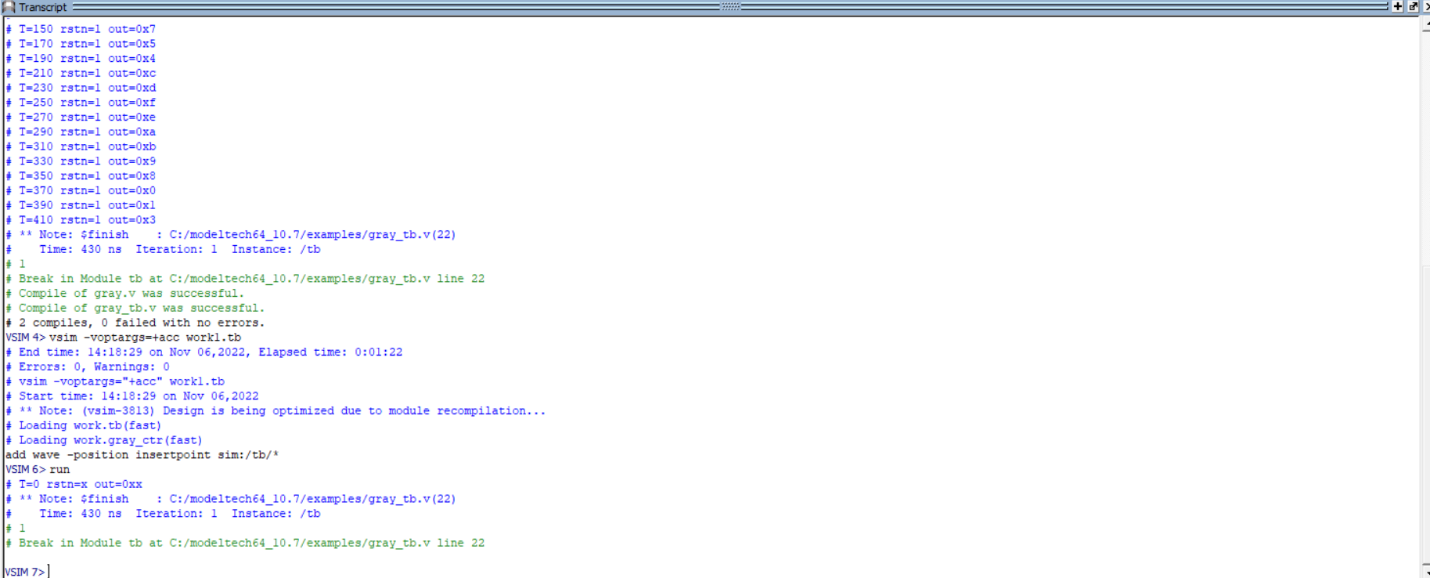
****

**Testbench:**

****

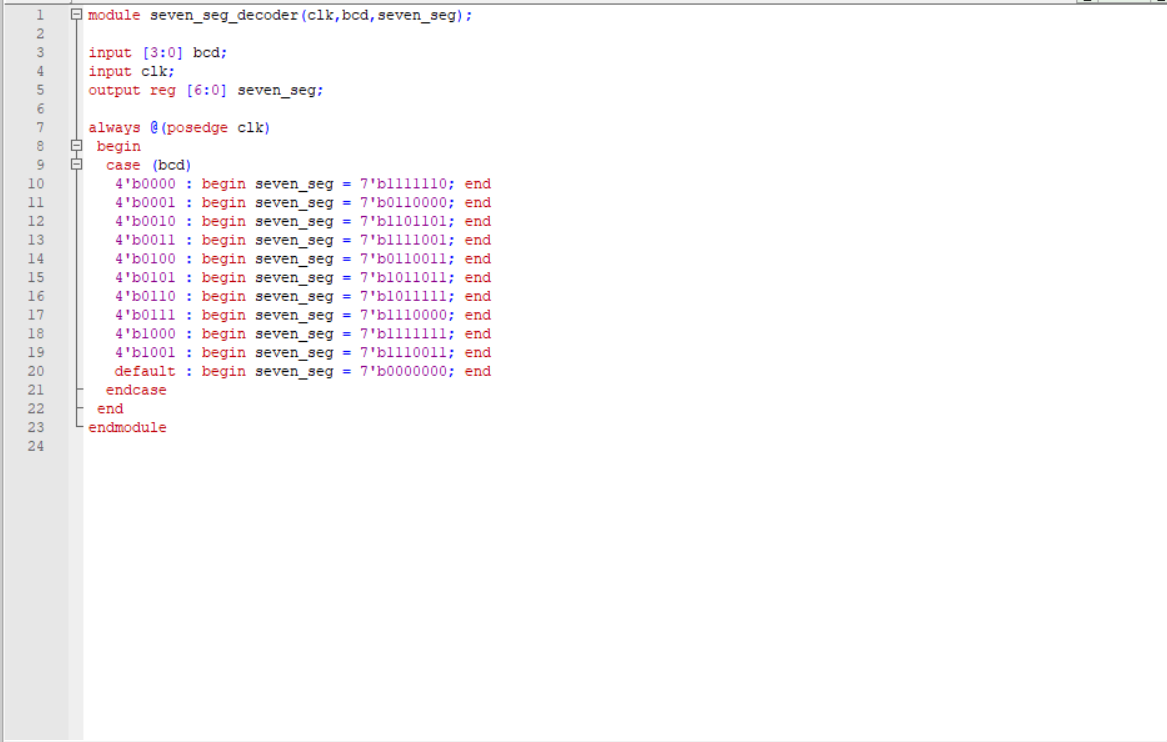
**Waveform**

****

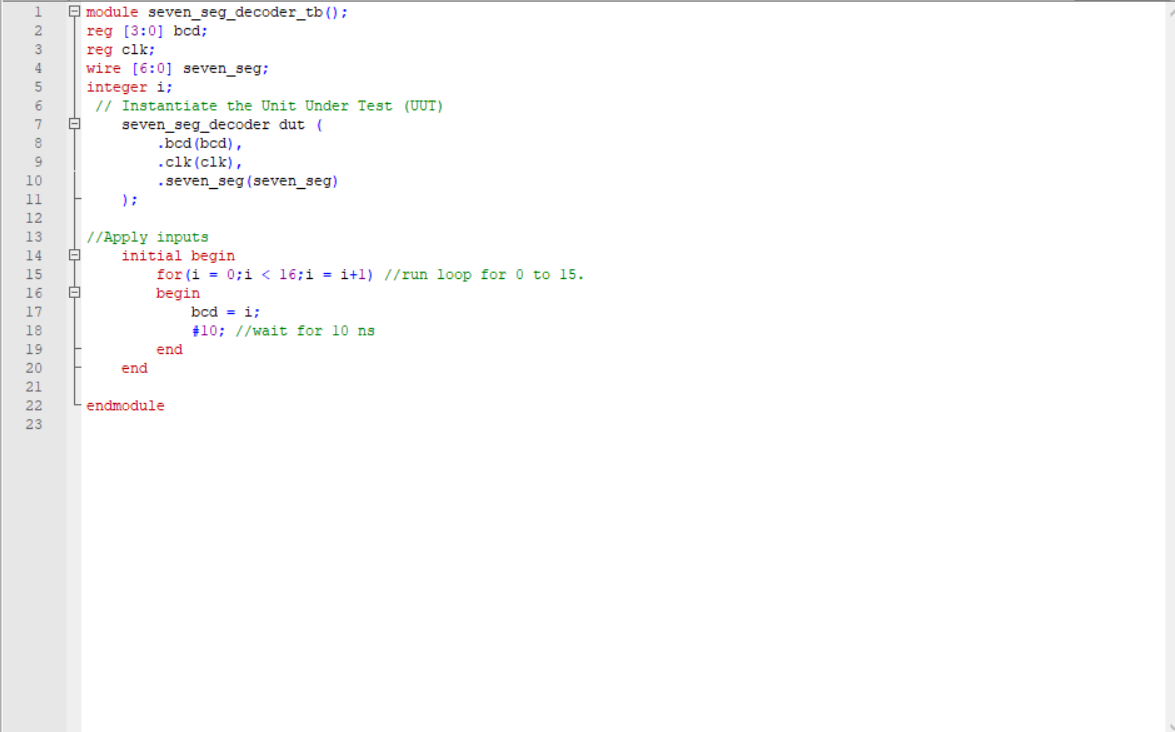
****

Ex4:

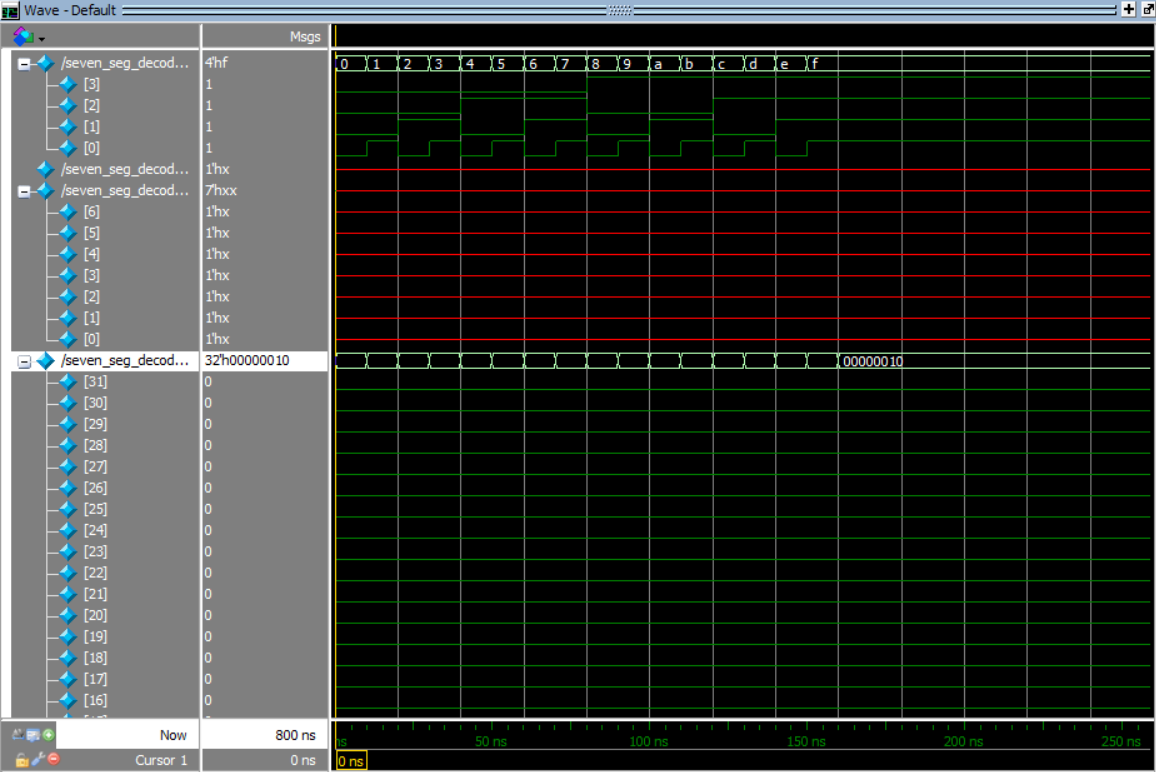
**Verilog code**

****

**Testbench**

****

**Waveform**

****