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;
```

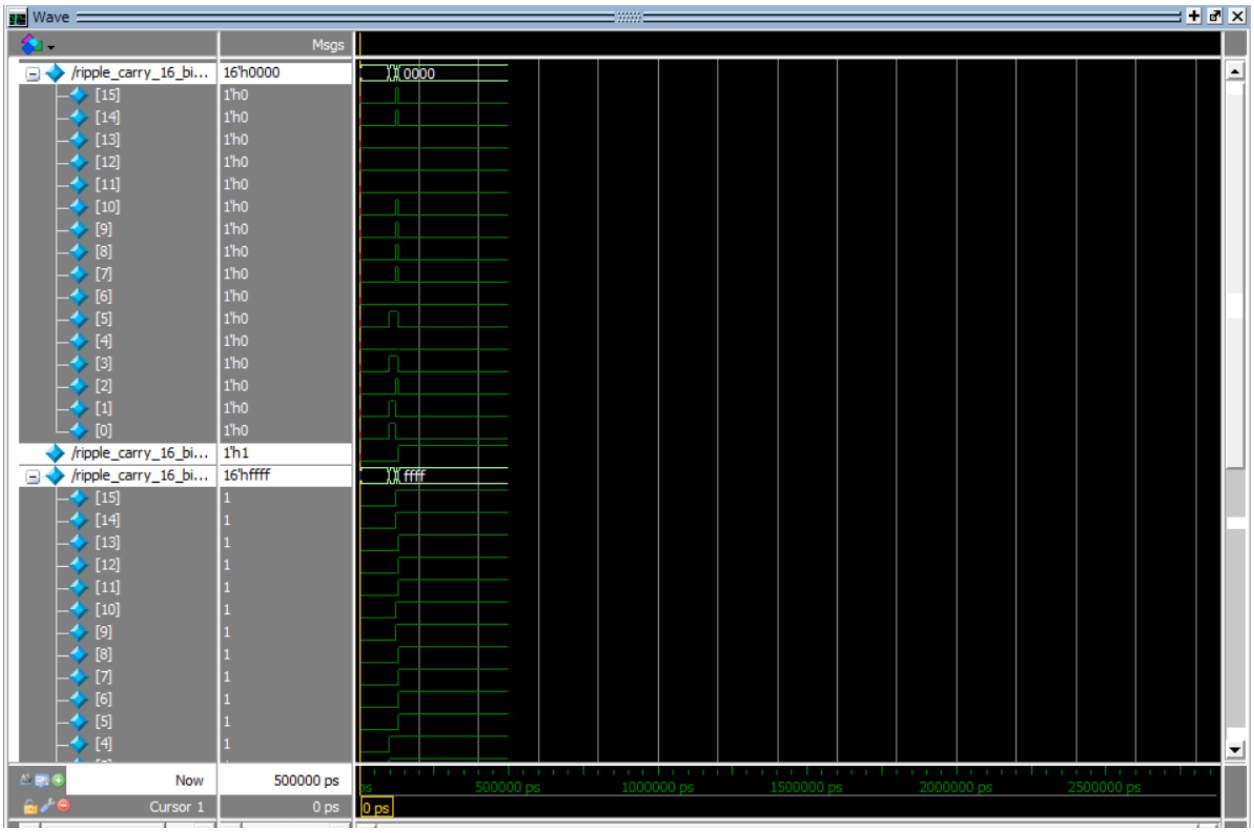
```
output sum, cout;
wire x,y,z;
half_adder h1(.a(a), .b(b), .sum(x), .cout(y));
half_adder h2(.a(x), .b(cin), .sum(sum), .cout(z));
or or_1(cout,z,y);
endmodule

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

C:/modeltech64_10.7/examples/rp_16_tb.v (/ripple_carry_16_bit_tb) - Default

```
`timescale 1ns / 1ps
module ripple_carry_16_bit_tb;
wire [15:0] sum;//output
wire cout;//output
reg [15:0] a,b;//input
reg cin;//input

ripple_carry_16_bit uut(
.a(a),
.b(b),
.cin(cin),
.sum(sum),
.cout(cout));

initial begin
$display($time, " << Starting the Simulation >>");
     a=0; b=0; cin=0;
#100 a= 16'b0000000000011111; b=16'b0000000000001100; cin=1'b0;
#10 a= 16'b0000000000011111; b=16'b0000000000001100; cin=1'b0;
#10 a= 16'b1100011000011111; b=16'b0000001100011100; cin=1'b1;
#10 a= 16'b1111111111111111; b=16'b0000000000000000; cin=1'b1;
end

initial
$monitor("time= ",
$time,
"A=%b,
B=&b,
Cin=%b
: Sum= %b,
Cout=%cout",
a,b,cin,sum,cout);
endmodule
```

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

```
1  module ripple_carry_counter(q, clk, reset);
2  output [3:0] q;
3  input clk, reset;
4  T_FF tff0(q[0], clk, reset);
5  T_FF tff1(q[1], q[0], reset);
6  T_FF tff2(q[2], q[1], reset);
7  T_FF tff3(q[3], q[2], reset);
8  endmodule
9
10 module T_FF(q, clk, reset);
11 output q;
12 input clk, reset;
13 wire d;
14 D_FF dff0(q, d, clk, reset);
15 not n1(d, q); // not is Verilog-provided primitive. Case sensitive.
16 endmodule
17
18 module D_FF(q, d, clk, reset);
19 output q;
20 input d, clk, reset;
21 reg q;
22 always @(posedge reset or negedge clk)
23 if (reset)
24 q = 1'b0;
25 else
26 q = d;
27 endmodule
28
```

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

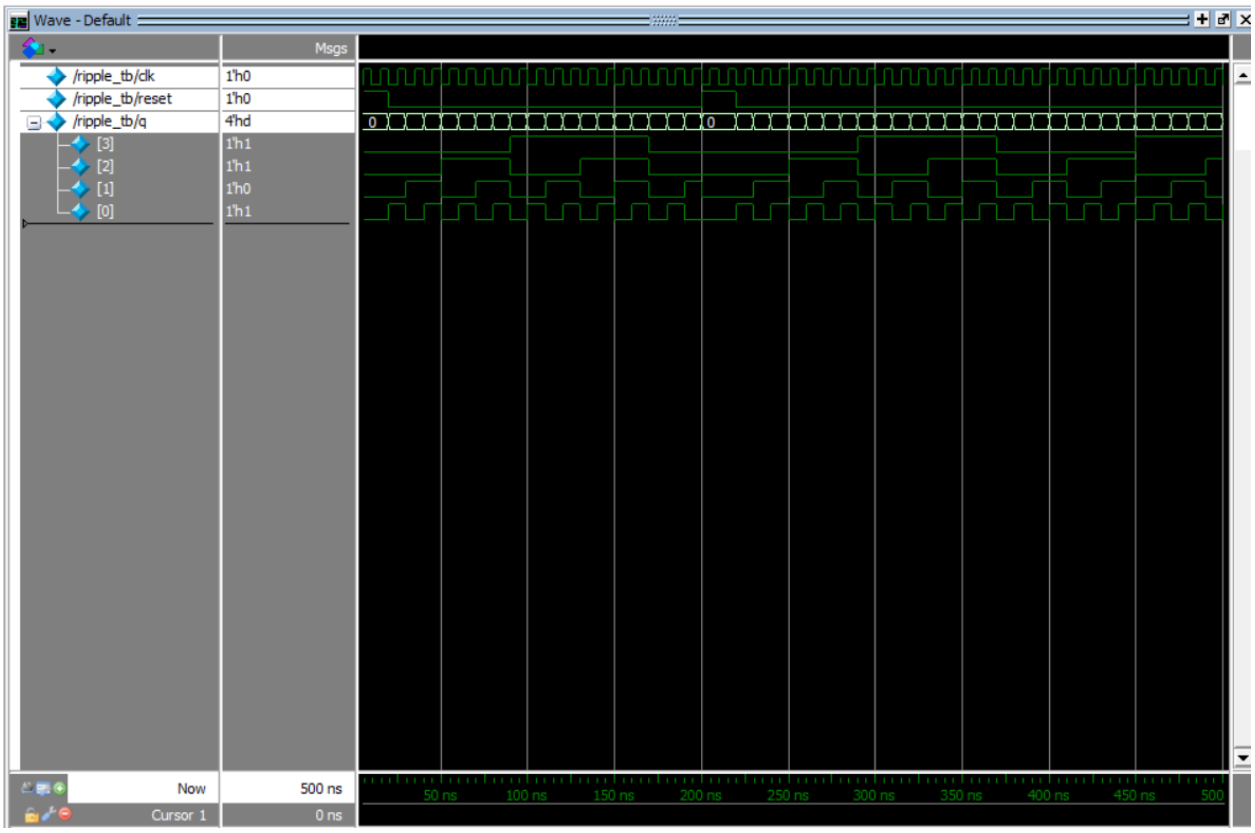
```
module ripple_tb;
reg clk; // Input
reg reset; // Input
wire [3:0] q; // Output

ripple_carry_counter r1 (.q(q), .clk(clk), .reset(reset));
initial
clk = 1'b0; // Set clk to 0
always
#5 clk = ~clk; // Toggle clk every 5 time units
initial
begin
reset = 1'b1;
#20 reset = 1'b0;
#180 reset = 1'b1;
#20 reset = 1'b0;
end endmodule

```

Comment:

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

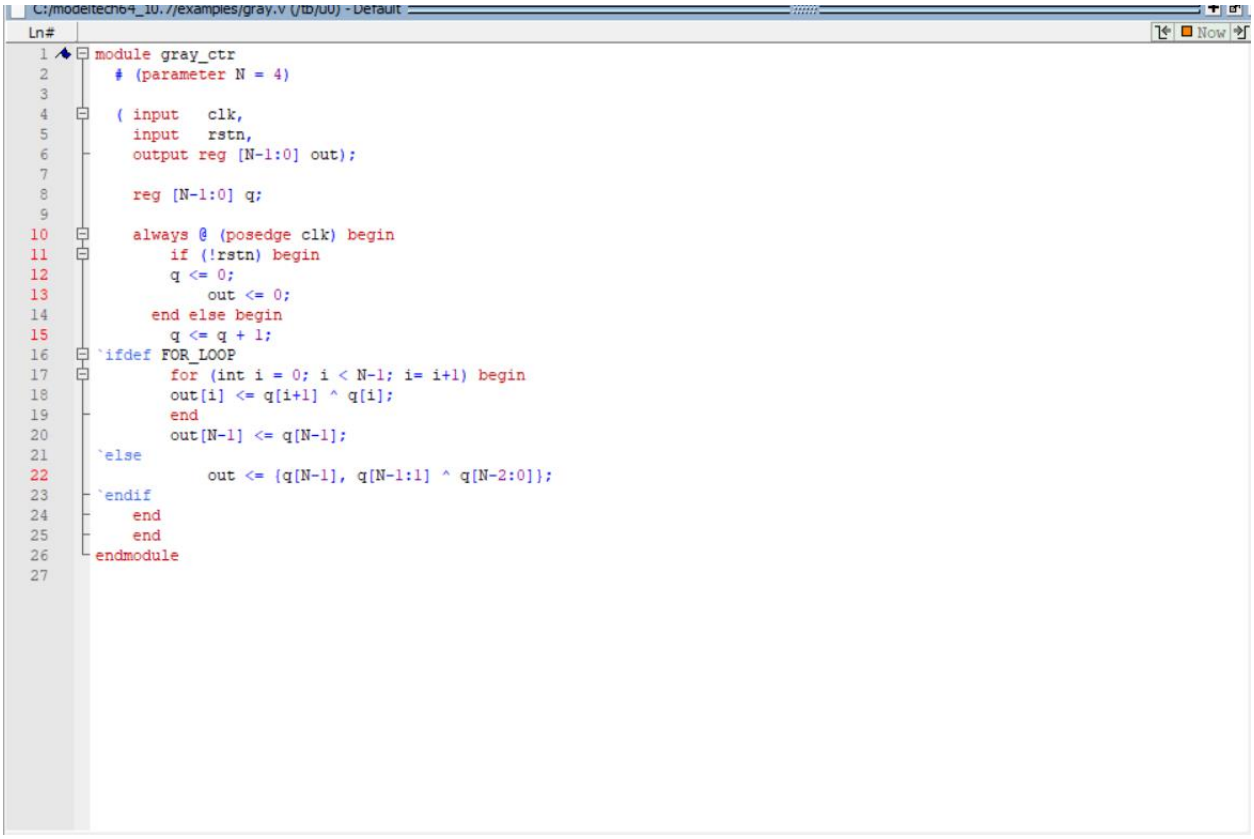
// This we provide reset values as the input where reset =1’b1, continue with re = 0,1,0,…

Waveform



Ex3:

Verilog code

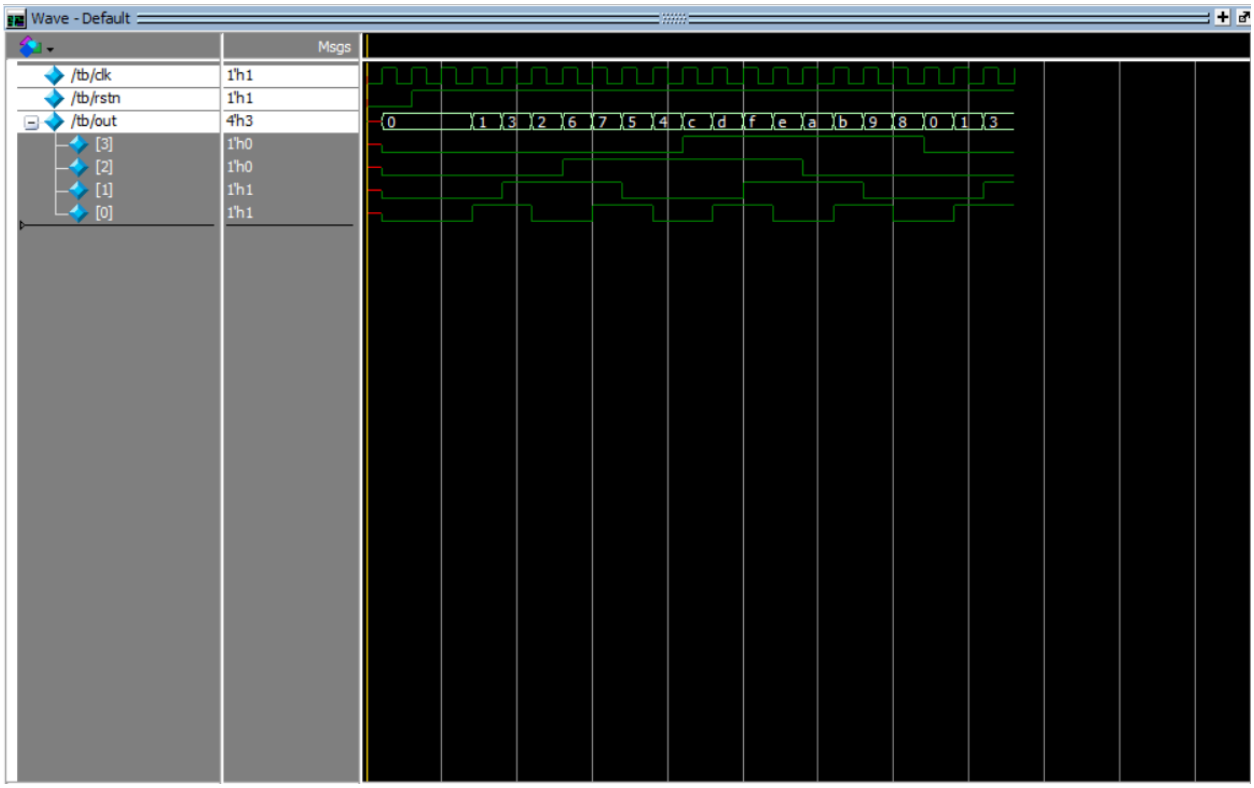

C:/modeltech64_10.7/examples/gray.v (/tb/u0) - Default
Ln#
Now
module gray_ctr
(parameter N = 4)
(input clk,
input rstn,
output reg [N-1:0] out);
reg [N-1:0] q;
always @ (posedge clk) begin
if (!rstn) begin
q <= 0;
out <= 0;
end else begin
q <= q + 1;
`ifdef FOR_LOOP
for (int i = 0; i < N-1; i= i+1) begin
out[i] <= q[i+1] ^ q[i];
end
out[N-1] <= q[N-1];
`else
out <= {q[N-1], q[N-1:1] ^ q[N-2:0]};
`endif
end
end
endmodule

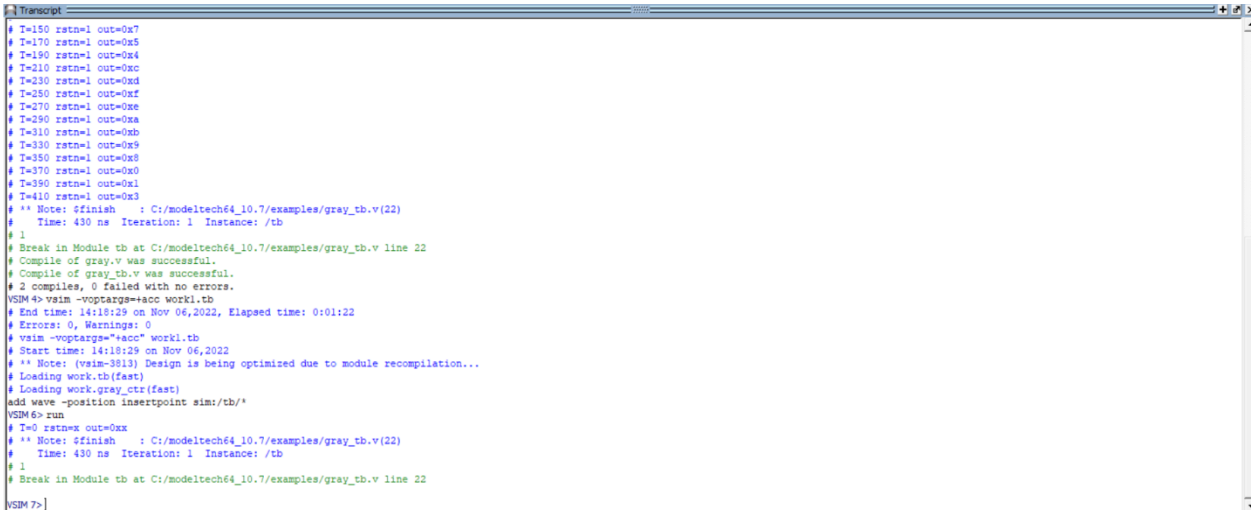
Testbench:

C:/modeltech64_10.7/examples/gray_tb.v (/tb) - Default

```
Ln#
 1  module tb;
 2    parameter N = 4;
 3
 4    reg clk;
 5    reg rstn;
 6    wire [N-1:0] out;
 7
 8    gray_ctr u0 ( .clk(clk),
 9                  .rstn(rstn),
10                  .out(out));
11
12    always #10 clk = ~clk;
13
14    initial begin
15      {clk, rstn} <= 0;
16
17      $display ("T=%0t rstn=%0b out=0x%0h", $time, rstn, out);
18
19      repeat(2) @ (posedge clk);
20      rstn <= 1;
21      repeat(20) @ (posedge clk);
22      $finish;
23    end
24  endmodule
25
```

Waveform





Ex4:

Verilog code

```
module seven_seg_decoder(clk,bcd,seven_seg);

input [3:0] bcd;
input clk;
output reg [6:0] seven_seg;

always @(posedge clk)
 begin
  case (bcd)
    4'b0000 : begin seven_seg = 7'b1111110; end
    4'b0001 : begin seven_seg = 7'b0110000; end
    4'b0010 : begin seven_seg = 7'b1101101; end
    4'b0011 : begin seven_seg = 7'b1111001; end
    4'b0100 : begin seven_seg = 7'b0110011; end
    4'b0101 : begin seven_seg = 7'b1011011; end
    4'b0110 : begin seven_seg = 7'b1011111; end
    4'b0111 : begin seven_seg = 7'b1110000; end
    4'b1000 : begin seven_seg = 7'b1111111; end
    4'b1001 : begin seven_seg = 7'b1110011; end
    default : begin seven_seg = 7'b0000000; end
  endcase
 end
endmodule
```

Testbench

```
module seven_seg_decoder_tb();
reg [3:0] bcd;
reg clk;
wire [6:0] seven_seg;
integer i;
 // Instantiate the Unit Under Test (UUT)
    seven_seg_decoder dut (
        .bcd(bcd),
        .clk(clk),
        .seven_seg(seven_seg)
    );

//Apply inputs
    initial begin
        for(i = 0;i < 16;i = i+1) //run loop for 0 to 15.
        begin
            bcd = i;
            #10; //wait for 10 ns
        end
    end

endmodule
```

Waveform