

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);

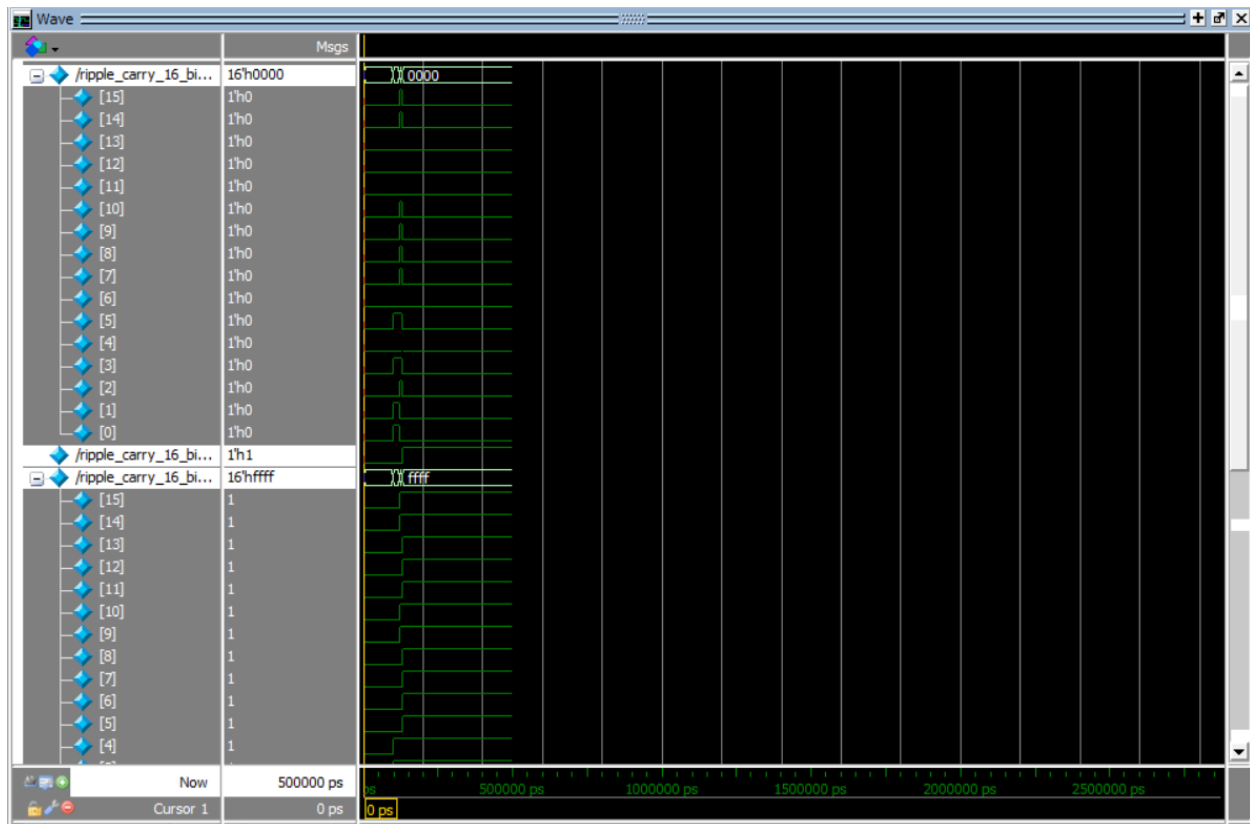
endmodule

```

## Testbench

```
C:/modeltech64_10.7/examples/rp_16_tb.v (/ripple_carry_16_bit_tb) - Default
Ln#
1 `timescale 1ns / 1ps
2 module ripple_carry_16_bit_tb;
3 wire [15:0] sum;//output
4 wire cout;//output
5 reg [15:0] a,b;//input
6 reg cin;//input
7
8 ripple_carry_16_bit uut(
9 .a(a),
10 .b(b),
11 .cin(cin),
12 .sum(sum),
13 .cout(cout));
14
15 initial begin
16 $display($time, " << Starting the Simulation >>");
17 a=0; b=0; cin=0;
18 #100 a= 16'b0000000000001111; b=16'b000000000001100; cin=1'b0;
19 #10 a= 16'b0000000000001111; b=16'b000000000001100; cin=1'b0;
20 #10 a= 16'b1100011000011111; b=16'b0000000110001100; cin=1'b1;
21 #10 a= 16'b1111111111111111; b=16'b0000000000000000; cin=1'b1;
22 end
23
24 initial
25 $monitor("time= ",
26 $time,
27 "A=%b,
28 B=%b,
29 Cin=%b
30 : Sum= %b,
31 Cout=%cout",
32 a,b,cin,sum,cout);
33 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

```

1 module ripple_tb;
2   reg clk; // Input
3   reg reset; // Input
4   wire [3:0] q; // Output
5
6   ripple_carry_counter r1 (.q(q), .clk(clk), .reset(reset));
7   initial
8     clk = 1'b0; // Set clk to 0
9   always
10    #5 clk = ~clk; // Toggle clk every 5 time units
11   initial
12   begin
13     reset = 1'b1;
14     #20 reset = 1'b0;
15     #180 reset = 1'b1;
16     #20 reset = 1'b0;
17   end endmodule
18

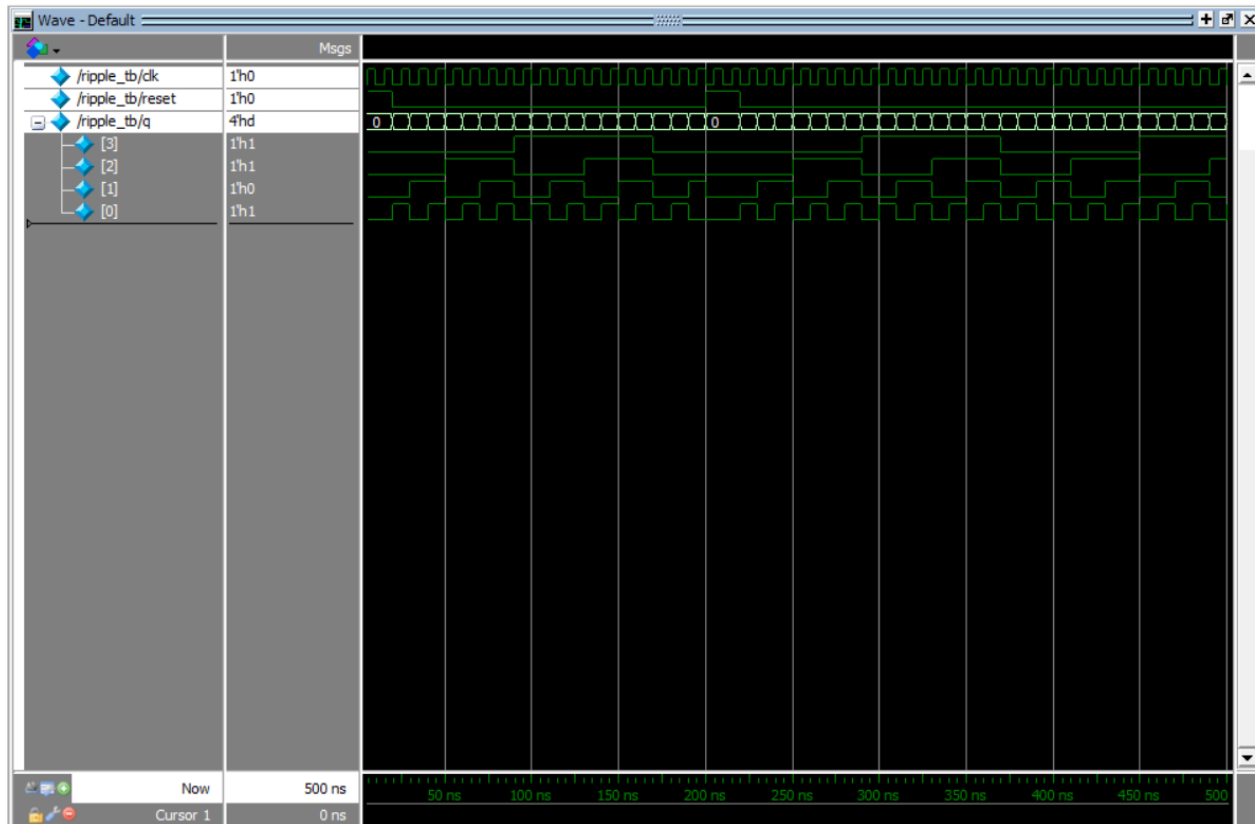
```

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:/modeltechno4_10./examples/gray.v (JTB/UD) - Default
Ln#
1 module gray_ctr
2   # (parameter N = 4)
3
4   ( input  clk,
5     input  rstn,
6     output reg [N-1:0] out);
7
8   reg [N-1:0] q;
9
10  always @ (posedge clk) begin
11    if (!rstn) begin
12      q <= 0;
13      out <= 0;
14    end else begin
15      q <= q + 1;
16      `ifdef FOR_LOOP
17        for (int i = 0; i < N-1; i= i+1) begin
18          out[i] <= q[i+1] ^ q[i];
19        end
20        out[N-1] <= q[N-1];
21      `else
22        out <= {q[N-1], q[N-1:1] ^ q[N-2:0]};
23      `endif
24    end
25  end
26 endmodule
27
```

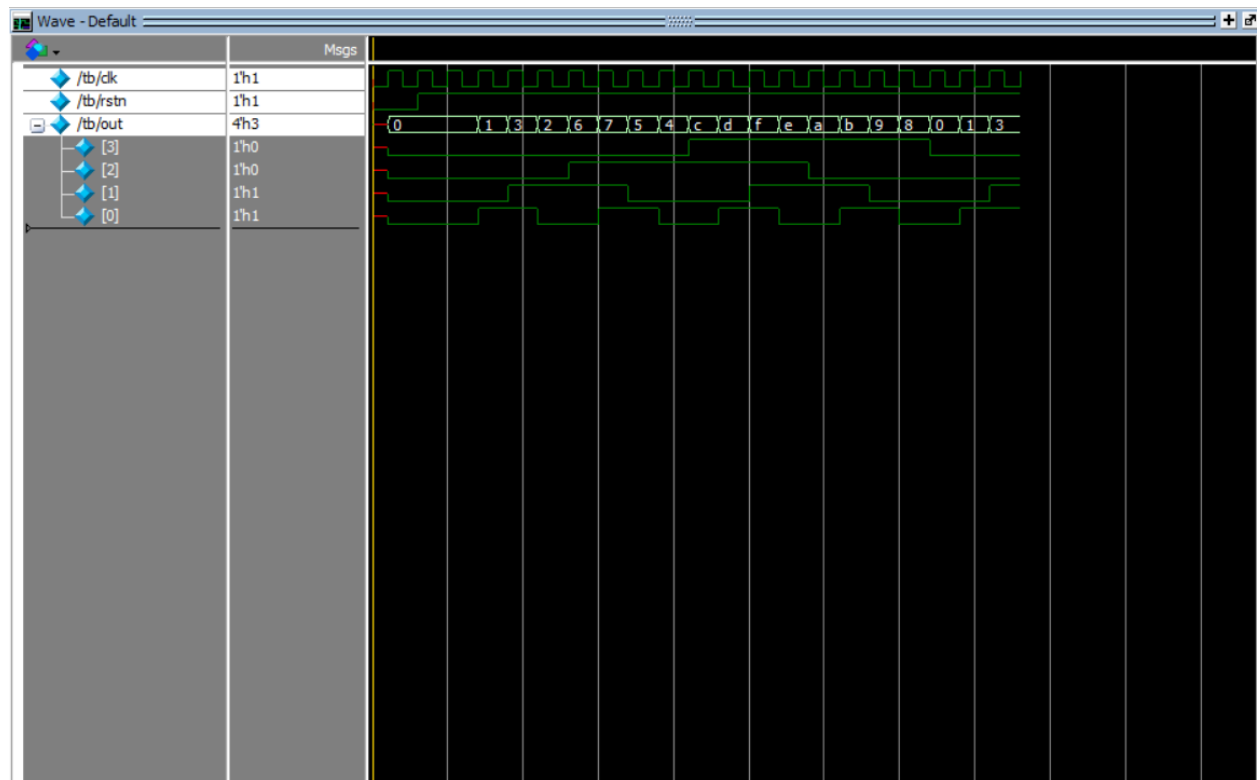
**Testbench:**

```

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



```

Transcript
# I=150 ratn=1 out=0x7
# I=170 ratn=1 out=0x5
# I=190 ratn=1 out=0x4
# I=210 ratn=1 out=0x3
# I=230 ratn=1 out=0x2
# I=250 ratn=1 out=0x1
# I=270 ratn=1 out=0xe
# I=290 ratn=1 out=0xa
# I=310 ratn=1 out=0xb
# I=330 ratn=1 out=0x9
# I=350 ratn=1 out=0x8
# I=370 ratn=1 out=0x0
# I=390 ratn=1 out=0x1
# I=410 ratn=1 out=0x3
** Note: $finish : C:/modeltech64_10.7/examples/gray_tb.v(22)
# Time: 430 ns Iteration: 1 Instance: /tb
# 1
# Break in Module tb at C:/modeltech64_10.7/examples/gray_tb.v line 22
# Compile of gray.v was successful.
# Compile of gray_tb.v was successful.
# 2 compiles, 0 failed with no errors.
VSI4> vsim -voptargs="+acc work1.tb"
# End time: 14:18:29 on Nov 06,2022, Elapsed time: 0:01:22
# Errors: 0, Warnings: 0
# vsim -voptargs="+acc" work1.tb
# Start time: 14:18:29 on Nov 06,2022
# ** Note: (vsim-3813) Design is being optimized due to module recompilation...
# Loading work.tb(fast)
# Loading work.gray_ctr(fast)
add wave -position insertpoint sim:/tb/*
VSI4> run
# I=0 ratn=x out=0xx
# ** Note: $finish : C:/modeltech64_10.7/examples/gray_tb.v(22)
# Time: 430 ns Iteration: 1 Instance: /tb
# 1
# Break in Module tb at C:/modeltech64_10.7/examples/gray_tb.v line 22
VSI4>

```

Ex4:

## Verilog code

```

1 module seven_seg_decoder(clk,bcd,seven_seg);
2
3     input [3:0] bcd;
4     input clk;
5     output reg [6:0] seven_seg;
6
7     always @(posedge clk)
8     begin
9         case (bcd)
10            4'b0000 : begin seven_seg = 7'b1111110; end
11            4'b0001 : begin seven_seg = 7'b0110000; end
12            4'b0010 : begin seven_seg = 7'b101101; end
13            4'b0011 : begin seven_seg = 7'b1111001; end
14            4'b0100 : begin seven_seg = 7'b0110011; end
15            4'b0101 : begin seven_seg = 7'b1011011; end
16            4'b0110 : begin seven_seg = 7'b1011111; end
17            4'b0111 : begin seven_seg = 7'b1110000; end
18            4'b1000 : begin seven_seg = 7'b1111111; end
19            4'b1001 : begin seven_seg = 7'b1110011; end
20            default : begin seven_seg = 7'b0000000; end
21        endcase
22    end
23 endmodule
24

```

## Testbench

```

1 module seven_seg_decoder_tb();
2   reg [3:0] bcd;
3   reg clk;
4   wire [6:0] seven_seg;
5   integer i;
6   // Instantiate the Unit Under Test (UUT)
7   seven_seg_decoder dut (
8     .bcd(bcd),
9     .clk(clk),
10    .seven_seg(seven_seg)
11  );
12
13  //Apply inputs
14  initial begin
15    for(i = 0; i < 16; i = i+1) //run loop for 0 to 15.
16    begin
17      bcd = i;
18      #10; //wait for 10 ns
19    end
20  end
21
22 endmodule
23

```

## Waveform

