P1). 3-Bit upcounter

Design code:

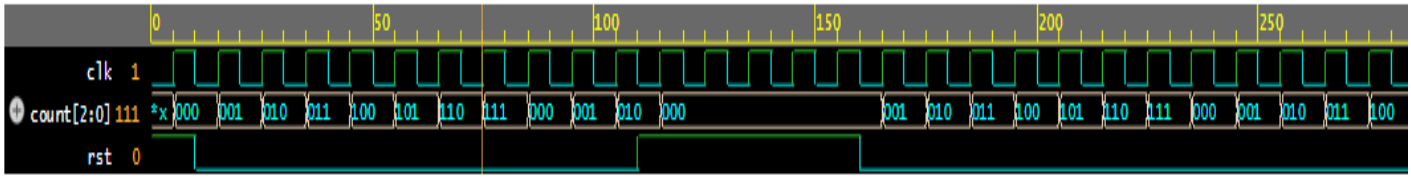
```
module counter(input clk, rst, output reg
[2:0]count);
  reg [2:0]temp;

  always@(posedge clk )begin
    if(rst)
      temp <= 3'b0;
    else
      temp <= temp + 1;
  end
  assign count = temp;
  endmodule
```

Test bench code:

```
module tb;
  reg clk, rst;
  wire [2:0]count;

  counter dut(clk, rst, count);

   always #5 clk = ~clk;
  initial begin
    clk = 0;
    rst = 1;
    #10 rst = 0;
    $monitor("count = %0b", count);
    #50 finish;
  end

  initial begin
    $dumpfile("dump.vcd");
    $dumpvars(0, clk, rst, count);
  end
endmodule
```

OUTPUT:

```
reset = 0, count = 0
reset = 0, count = 1
reset = 0, count = 10
reset = 0, count = 11
reset = 0, count = 100
reset = 0, count = 101
reset = 0, count = 110
reset = 0, count = 111
reset = 0, count = 0
reset = 0, count = 1
reset = 0, count = 10
reset = 0, count = 11
```



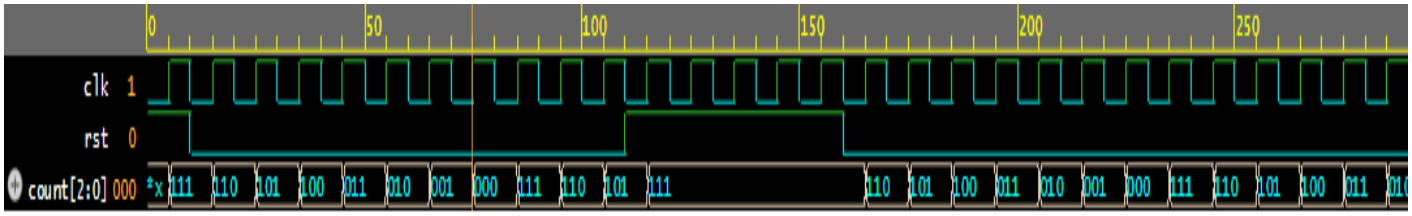
P2). 3-Bit down counter

Design code:	*Test bench code:*

module counter(input clk, rst,
output [2:0]count);
reg [2:0]temp;
 	
always@(posedge clk)begin
if(rst)
temp <= 3'd7;
else
temp <= temp - 1;
end
assign count = temp;
endmodule
	module tb;
	reg clk, rst;
	wire [2:0]count;
	
	counter dut(clk, rst, count);
	
	always #5 clk = ~clk;
	initial begin
	clk = 0;
	rst = 1;
	#10 rst = 0;
	#100 rst = 1;
	#50 rst = 0;
	$monitor("reset = %0b, count = %0b", rst,count);
	#200 $finish;
	end
	
	initial begin
	$dumpfile("dump.vcd");
	$dumpvars(0, clk, rst, count);
	end
	endmodule

OUTPUT:

```
reset = 0, count = 110
reset = 0, count = 101
reset = 0, count = 100
reset = 0, count = 11
reset = 0, count = 10
reset = 0, count = 1
reset = 0, count = 0
reset = 0, count = 111
reset = 0, count = 110
reset = 0, count = 101
reset = 0, count = 100
```

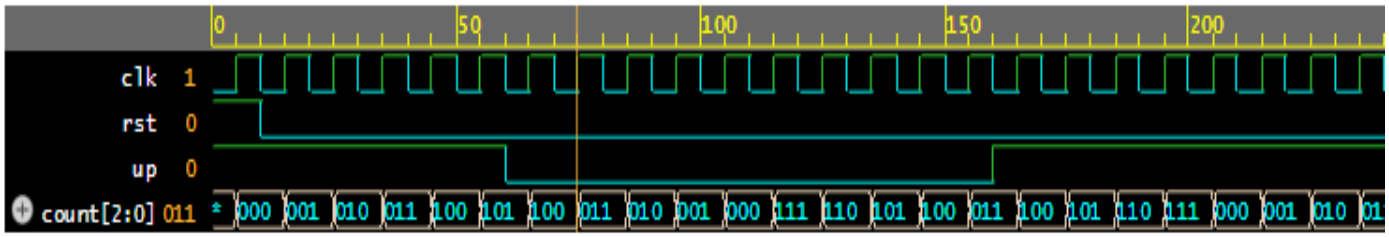


P3). 3- bit updown counter

Design code:	*Test bench code:*

module counter(input clk, rst, up, output [2:0]count);
reg [2:0]temp;
assign count = temp;
always @(posedge clk) begin
if (rst)
temp <= 3'b000;
else begin
if (up)
temp <= temp + 1;
else
temp <= temp - 1;
end
end
endmodule
	// Code your testbench here
	// or browse Examples
	module tb;
	reg clk, rst, up, down;
	wire [2:0]count;
	
	counter dut(clk, rst, up, count);
	
	always #5 clk = ~clk;
	initial begin
	clk = 0;
	rst = 1;
	up = 1;
	#10 rst = 0;
	up = 1;
	#100
	up = 0;
	$monitor("count = %0b", count);
	#500 $finish;
	end
	
	initial begin
	$dumpfile("dump.vcd");
	$dumpvars(0, clk, rst, up,count);
	end
	endmodule

OUTPUT:

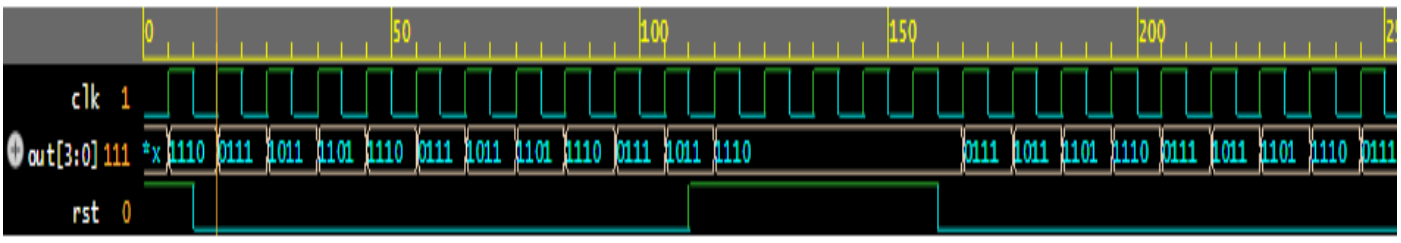


P4). Ring counter

Design code:	*Test bench code:*

module counter(input clk, rst, output [3:0]out);
reg [3:0]temp;
 	
always@(posedge clk)begin
if(rst)
temp <= 4'b1110;//declare with non zero value
else
temp <= {temp[0], temp[3:1]};
end
assign out = temp;
endmodule
	module tb;
	reg clk, rst;
	wire [3:0]out;
	
	counter dut(clk, rst, out);
	
	always #5 clk = ~clk;
	initial begin
	clk = 0;
	rst = 1;
	#10 rst = 0;
	#100 rst = 1;
	#50 rst = 0;
	$monitor("reset = %0b, out = %0b", rst,out);
	#200 $finish;
	end
	
	initial begin
	$dumpfile("dump.vcd");
	$dumpvars(0, clk, rst, out);
	end
	endmodule

OUTPUT:

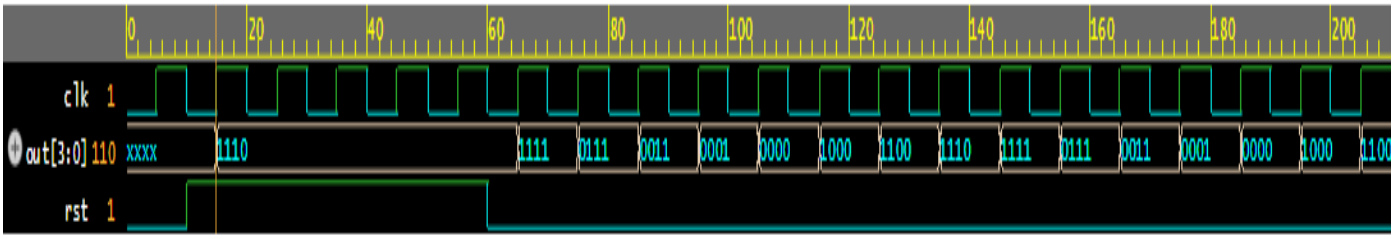


P4). Johnson counter

Design code:	*Test bench code:*

module counter(input clk, rst, output [3:0]out);
reg [3:0]temp;
 	
always@(posedge clk)begin
if(rst)
temp <= 4'b1110;//declare with non zero value
else
temp <= {~temp[0], temp[3:1]};
end
assign out = temp;
endmodule
	module tb;
	reg clk, rst;
	wire [3:0]out;
	
	counter dut(clk, rst, out);
	
	always #5 clk = ~clk;
	initial begin
	clk = 0;
	rst = 1;
	#10 rst = 0;
	#100 rst = 1;
	#50 rst = 0;
	$monitor("reset = %0b, out = %0b", rst,out);
	#200 $finish;
	end
	
	initial begin
	$dumpfile("dump.vcd");
	$dumpvars(0, clk, rst, out);
	end
	endmodule

OUTPUT:



P5). Gradually incrementing and decrementing counter

Design code:	*Test bench code:*

OUTPUT: