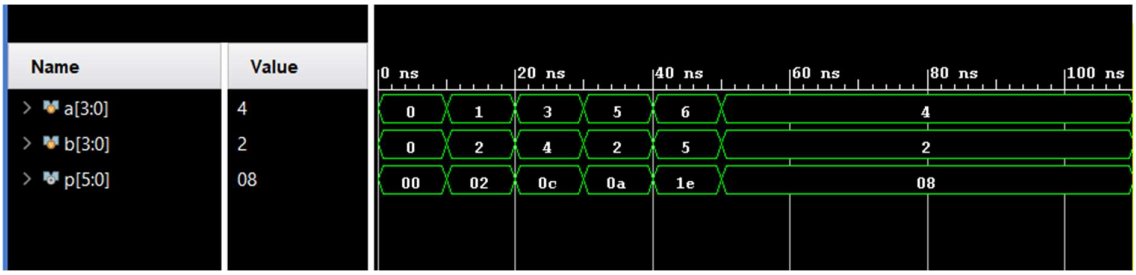
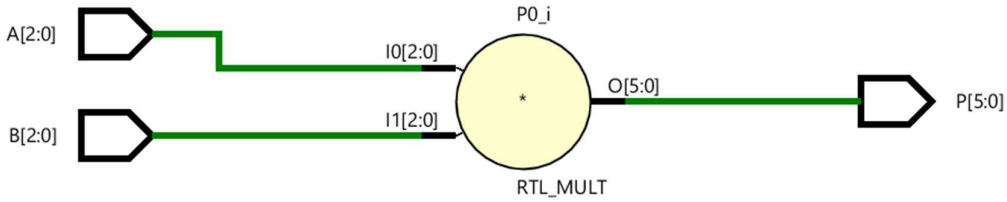
VERILOG ASSIGNMENT

3 BIT BINARY MULTIPLIER

```
module bitmultiplier3(input [2:0]A,B,
output reg [5:0]P);
always@(*)
P=(A*B);
Endmodule
module multiplierbit3_tb();
reg [3:0]a,b;
wire [5:0]p;
bitmultiplier3 dut (a,b,p);
initial
begin
a=3'b000;b=3'b000;#10;
a=3'b001;b=3'b010;#10;
a=3'b011;b=3'b100;#10;
a=3'b101;b=3'b010;#10;
a=3'b110;b=3'b101;#10;
a=3'b100;b=3'b010;#10;
$display("a=%0b,b=%0b,p=%0b",a,b,p);
#50;
$stop;
end
endmodule
```



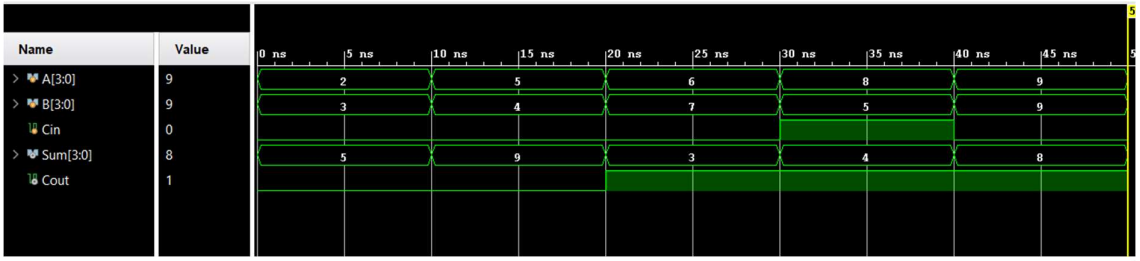


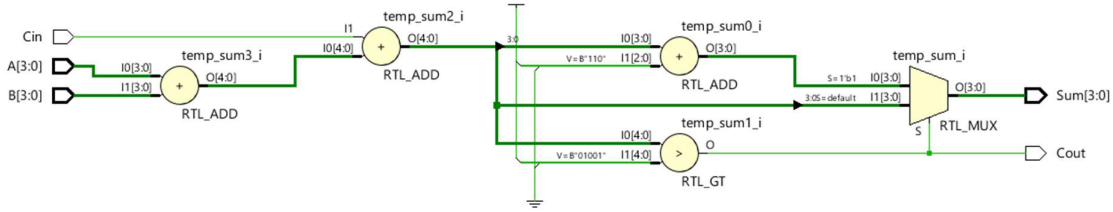
2.BINARY ADDER

```
module bcd_adder(
input [3:0] A, input [3:0] B, input Cin,
            output reg [3:0] Sum, output reg Cout);
reg [4:0] temp_sum;
```

```
always @(*) begin
   temp_sum = A + B + Cin;
   if (temp_sum > 9) begin
      temp_sum = temp_sum + 6;
      Cout = 1;
   end else Cout = 0;
   Sum = temp_sum[3:0];
end
endmodule

module bcd_adder_tb();
reg [3:0] A, B;
reg Cin;
wire [3:0] Sum;
wire Cout;

bcd_adder dut (.A(A), .B(B), .Cin(Cin), .Sum(Sum), .Cout(Cout));

initial begin
   $display("Time\tA\tB\tCin\t|\tSum\tCout");
   $monitor("%0t\t%0d\t%0d\t%b\t|\t%0d\t%b", $time, A, B, Cin,
Sum, Cout);
   A=4'd2; B=4'd3; Cin=0; #10;
   A=4'd5; B=4'd4; Cin=0; #10;
```

```
    A=4'd6; B=4'd7; Cin=0; #10;
    A=4'd8; B=4'd5; Cin=1; #10;
    A=4'd9; B=4'd9; Cin=0; #10;
    $display("Simulation Completed."); $finish;
end
endmodule
```

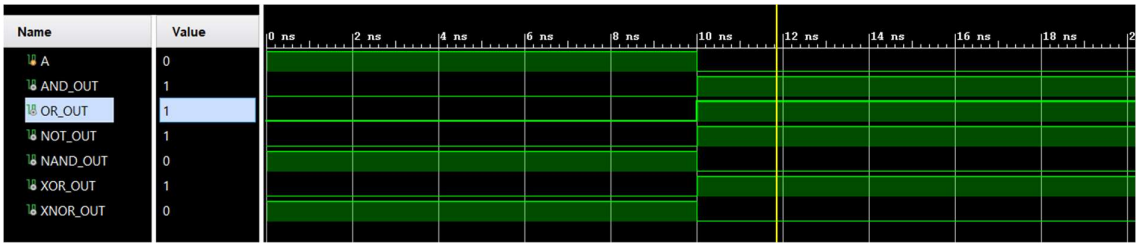


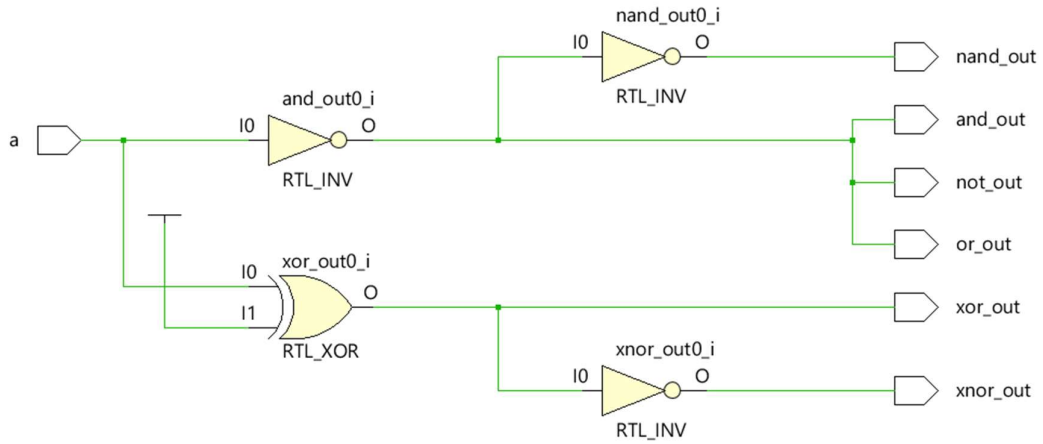


INVERTER USING BASIC GATES

```
module inverter_basicgates(input a,
output reg and_out,output reg or_out,output reg not_out,output reg nand_out,output reg xor_out,output reg xnor_out);
reg a_bar;
always@(*)
begin
a_bar=~a;
```

```
and_out=a_bar&1'b1;
or_out=a_bar|1'b0;
not_out=~a;
nand_out=~(a_bar&1'b1);
xor_out=a^1'b1;
xnor_out=~(a^1'b1);
end
endmodule

module invereter_tb();
reg A;
wire
AND_OUT,OR_OUT,NOT_OUT,NAND_OUT,XOR_OUT,XNOR_OUT;
inverter_basicgates
dut(A,AND_OUT,OR_OUT,NOT_OUT,NAND_OUT,XOR_OUT,XNOR_O
UT);
initial
begin
A=1'b1;#10;
A=1'b0;
end
endmodule
```

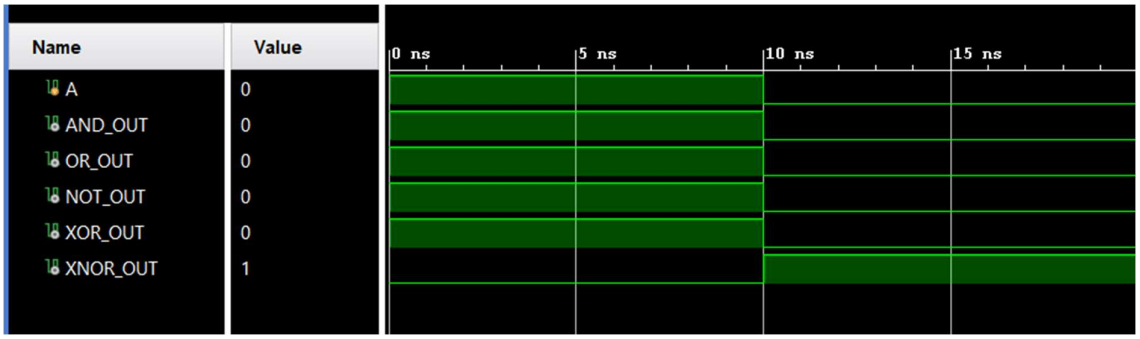


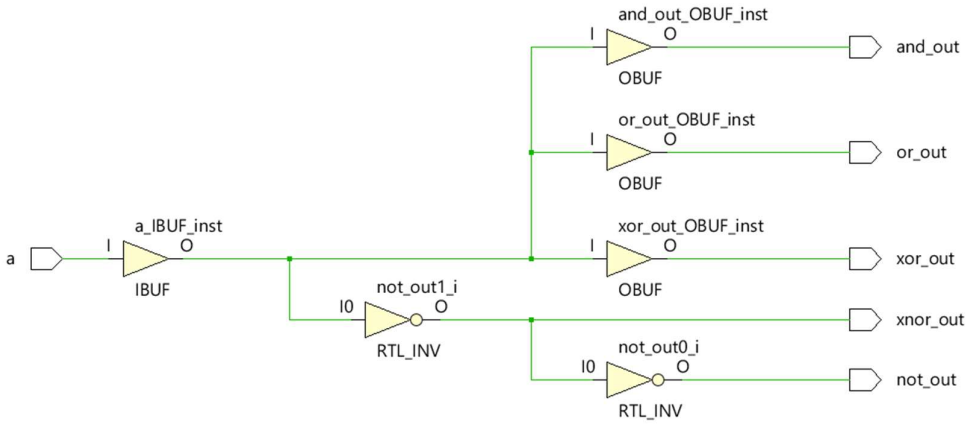


BUFFER USING BASIC GATES

```
module buffer_gates(input a,output reg
and_out,or_out,not_out,xor_out,xnor_out);
always@(*)
begin
and_out=a&1'b1;
or_out=a|1'b0;
not_out=~(~a);
xor_out=a^1'b0;
xnor_out=~(a^1'b0);
end
```

```
endmodule


module buffer_tb();
reg A;
wire AND_OUT,OR_OUT,NOT_OUT,XOR_OUT,XNOR_OUT;
buffer_gates dut (A,AND_OUT,OR_OUT,NOT_OUT,XOR_OUT,XNOR_OUT);
initial
begin
A=1'b1;#10;
A=1'b0;
#10;
$finish;
end
endmodule
```

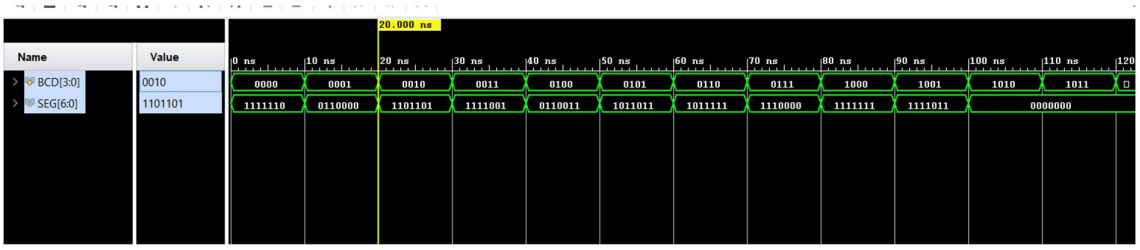


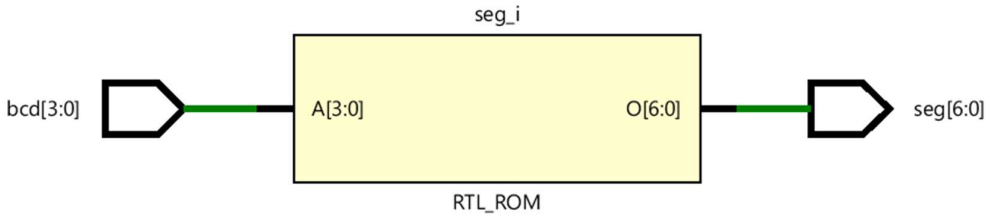


BCD TO 7 SEGMENT DISPLAY

```
module bcd_7_segment(input[3:0] bcd,output reg [6:0]seg);
always@(*)
begin
case(bcd)
4'b0000:seg=7'b1111110;
4'b0001:seg=7'b0110000;
4'b0010:seg=7'b1101101;
4'b0011:seg=7'b1111001;
4'b0100:seg=7'b0110011;
4'b0101:seg=7'b1011011;
4'b0110:seg=7'b1011111;
4'b0111:seg=7'b1110000;
4'b1000:seg=7'b1111111;
4'b1001:seg=7'b1111011;
default:seg=7'b0000000;
endcase
end
endmodule

module bcd_7_seg_tb();
reg [3:0]BCD;
wire[6:0]SEG;
```

```
bcd_7_segment dut (BCD,SEG);
initial
begin
BCD=4'd0;#10;
BCD=4'd1;#10;
BCD=4'd2;#10;
BCD=4'd3;#10;
BCD=4'd4;#10;
BCD=4'd5;#10;
BCD=4'd6;#10;
BCD=4'd7;#10;
BCD=4'd8;#10;
BCD=4'd9;#10;
BCD=4'd10;#10;
BCD=4'd11;#10;
BCD=4'd12;#10;
BCD=4'd13;#10;
BCD=4'd14;#10;
BCD=4'd15;#10;
$monitor("BCD=%0b,SEG=%0b",BCD,SEG);
$finish;
end
endmodule
```



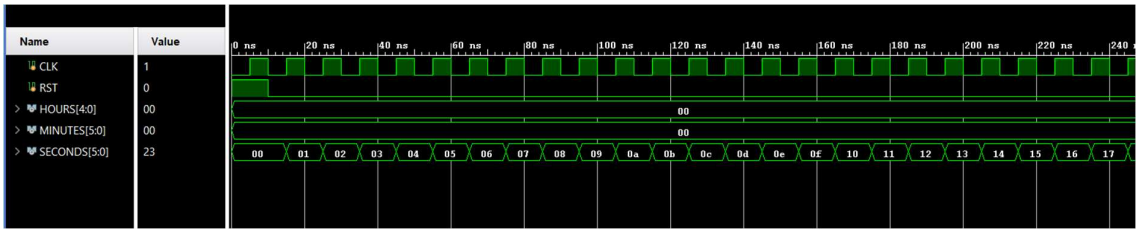


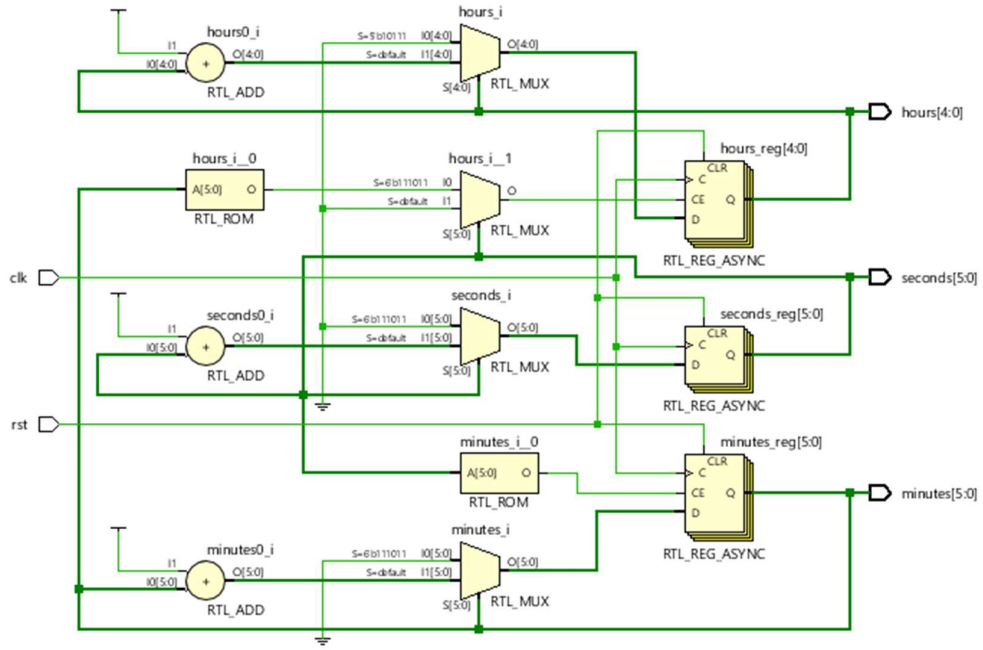
24 HOUR TIMER

```
module timer_24(input clk,rst,
output reg [4:0]hours,
output reg [5:0]minutes,
output reg [5:0]seconds);
always@(posedge clk or posedge rst)
if (rst)
begin
hours<=0;
minutes<=0;
seconds<=0;
end
else
begin
if(seconds==59)
begin
```

```
seconds<=0;
if(minutes==59)
begin
minutes<=0;
if(hours==23)
hours<=0;
else
hours<=hours+1;
end else
begin
minutes<=minutes+1;
end
end
else
begin
seconds<=seconds+1;
end
end
endmodule

module timer_24_tb( );
reg CLK,RST;
wire[4:0]HOURS;
wire[5:0]MINUTES,SECONDS;
timer_24 dut(CLK,RST,HOURS,MINUTES,SECONDS);
initial
```

```
begin
CLK=0;
forever #5 CLK=~CLK;
end
initial
begin
$display("CLK=%0b,RST=%0b,HOURS=%0b,MINUTES=%0b,SECONDS=%0b",CLK,
RST,HOURS,MINUTES,SECONDS);
RST=1;#10;RST=0;
repeat(3700)@(posedge CLK);
RST=1;#10;RST=0;
repeat(10)@(posedge CLK);
$display("simulation complete");
$finish;
end
endmodule
```

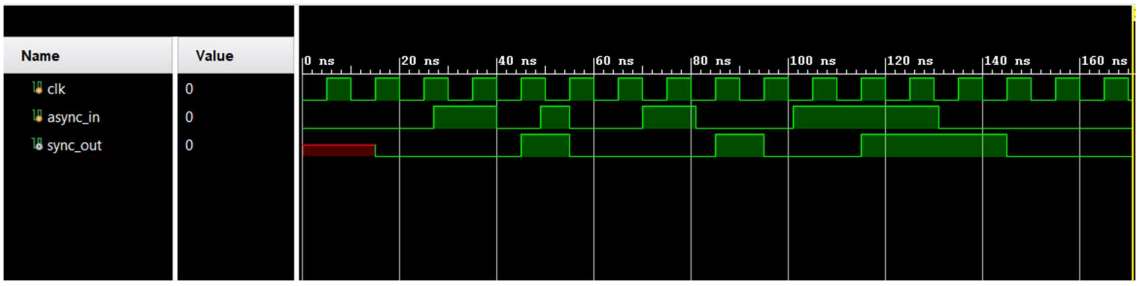


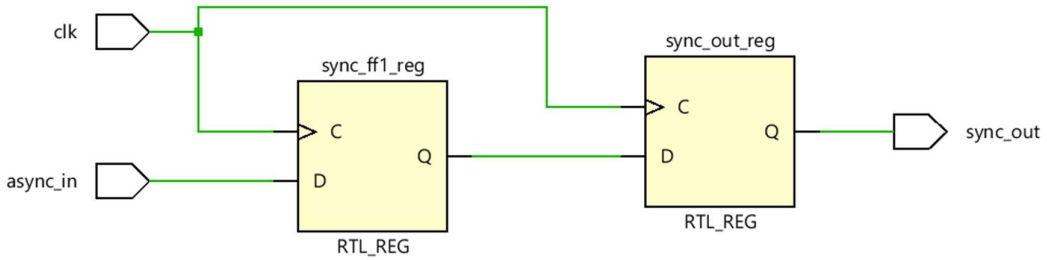


2 FLIPFLOP SYNCHRONIZER

```
module two_ff_synch(input wire clk, input wire async_in,
 output reg sync_out);
reg sync_ff1;
always @(posedge clk) begin
    sync_ff1 <= async_in;
    sync_out <= sync_ff1;
end
endmodule

module two_ff_syn_tb();
reg clk, async_in;
wire sync_out;
  two_ff_synch dut (clk, async_in, sync_out);
```

```
initial begin
   clk = 0; forever #5 clk = ~clk;
end

initial begin
   $display("Time\tclk\tasync_in\tsync_out");
   $monitor("%0t\t%b\t%b\t\t%b", $time, clk, async_in, sync_out);
   async_in = 0; #20;
   #7  async_in = 1;
   #13 async_in = 0;
   #9  async_in = 1;
   #6  async_in = 0;
   #15 async_in = 1;
   #11 async_in = 0;
   #20 async_in = 1;
   #30 async_in = 0;

   #40;
   $display("Simulation Complete.");
   $finish;
end
endmodule
```



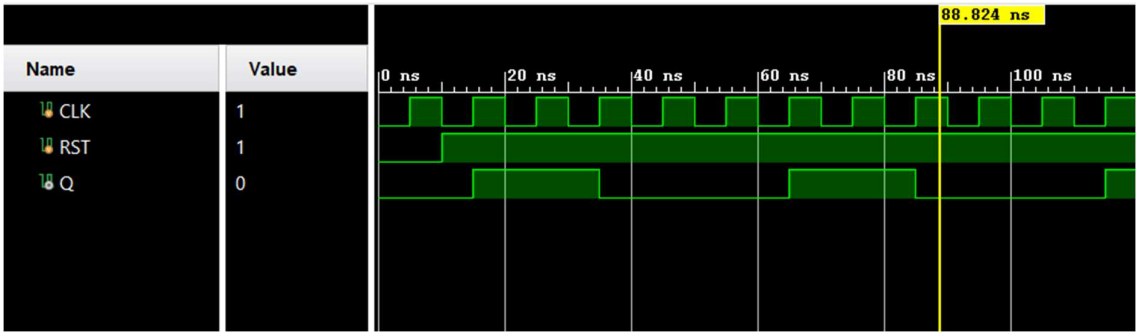


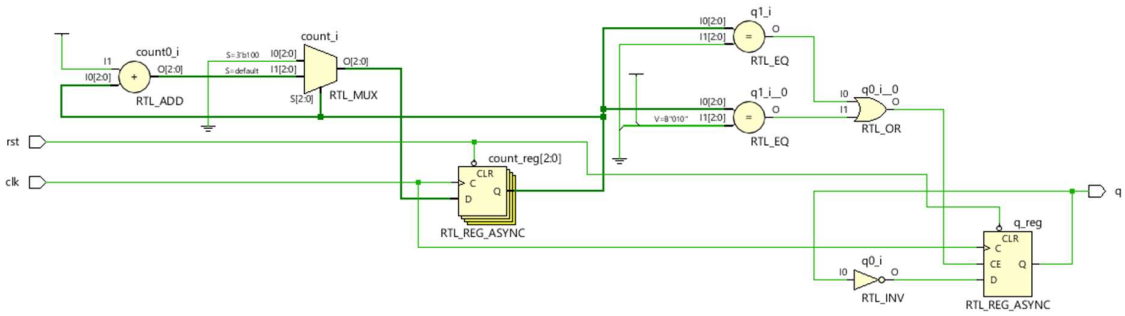
MOD 5 UPCOUNTER WITH 50% DUTY CYCLE

```
module mod5 (input  clk, rst,
 output reg  q);
 reg [2:0] count;
 always @(posedge clk or negedge rst)
  begin
 if (!rst) begin
 count <= 3'd0;
  q <= 1'b0;
  end else begin
  if (count == 3'd4)
  count <= 3'd0;
  else
```

```
  count <= count + 3'd1;
 if (count == 3'd0 || count == 3'd2)
  q <= ~q;
  end
  end
endmodule

module mod_5_tb();
reg CLK,RST;
wire Q;
mod5 dut (CLK,RST,Q);
initial
begin
CLK=0;
forever #5 CLK=~CLK;
end
initial
begin
RST=1'b0;
#10;
RST=1'b1;
#100;
$display("CLK=%0b,RST=%0b,Q=%0b",CLK,RST,Q);
#10;
$stop();
end
```

endmodule





AUTOMATIC DOOR OPENING

Module automatic_door(clk,rst,sensor,timer,motor_out);

input clk,rst,sensor,timer;

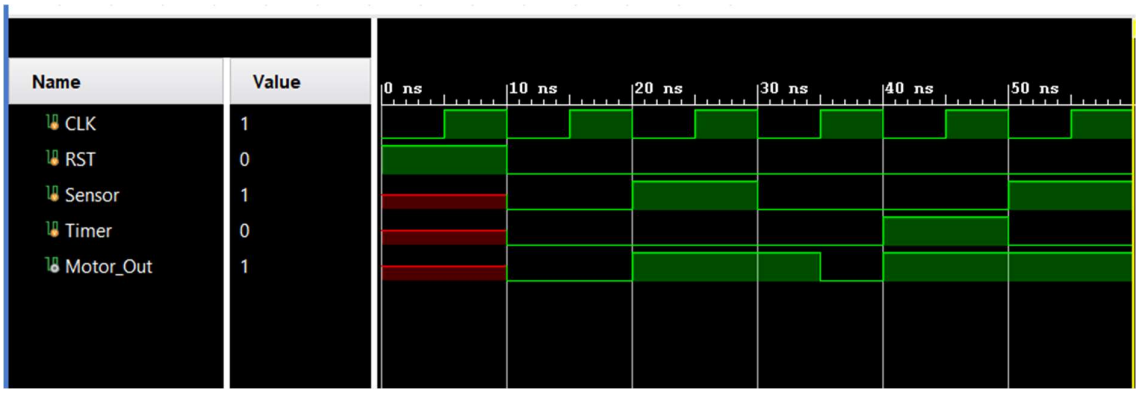
output reg motor_out;

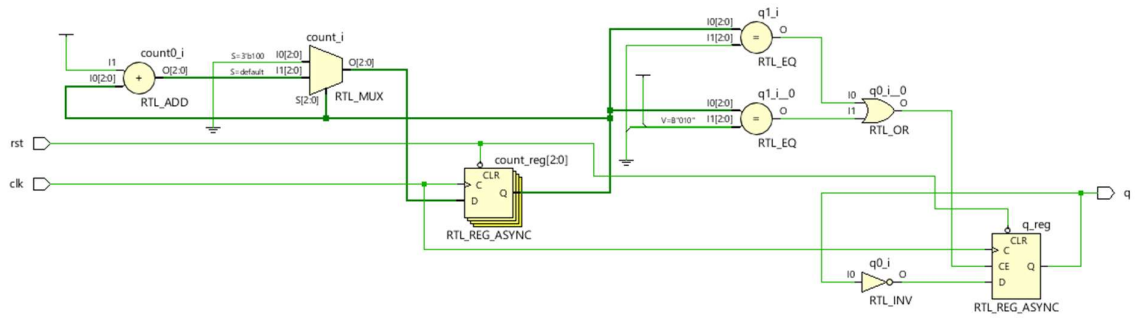
reg [1:0]ps,ns;

parameter closed=2'b00,opening=2'b01,open=2'b10,closing=2'b11;

```
always@(posedge clk) //reset logic
begin
if(rst)
ps<=closed;
else
ps<=ns;
end
always@(*) //ns & output logic
begin
case(ps)
closed:begin
ns<=sensor?opening:closed;
motor_out<=sensor?1:0;
end
opening:begin
ns<=open;
motor_out<=1'b1;
end
open:begin
ns<=timer?closing:open;
motor_out<=timer?1:0;
end
closing:begin
ns<=sensor?opening:closed;
motor_out<=1'b1;
end
```

```
default:ns<=closed;
endcase
end
endmodule

module automatic_door_tb();
reg CLK,RST,Sensor,Timer;
wire Motor_Out;
automatic_door dut(CLK,RST,Sensor,Timer,Motor_Out);
initial
begin
CLK=0;
forever #5 CLK=~CLK;
end
initial
begin
RST=1'b1; #10;
RST=1'b0; Sensor=1'b0; Timer=1'b0; #10;
Sensor=1'b1; #10;
Sensor=1'b0; #10;
Timer=1'b1; #10;
Timer=1'b0; Sensor=1'b1; #10;
$stop;
$display("CLK=%0b,RST=%0b,Sensor=%0b,Timer=%0b,Motor_Out=%0b",CLK,R
ST,Sensor,Timer,Motor_Out);
end
endmodule
```



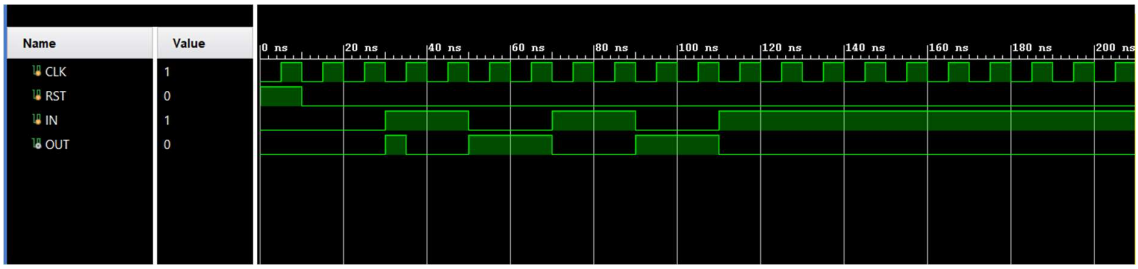


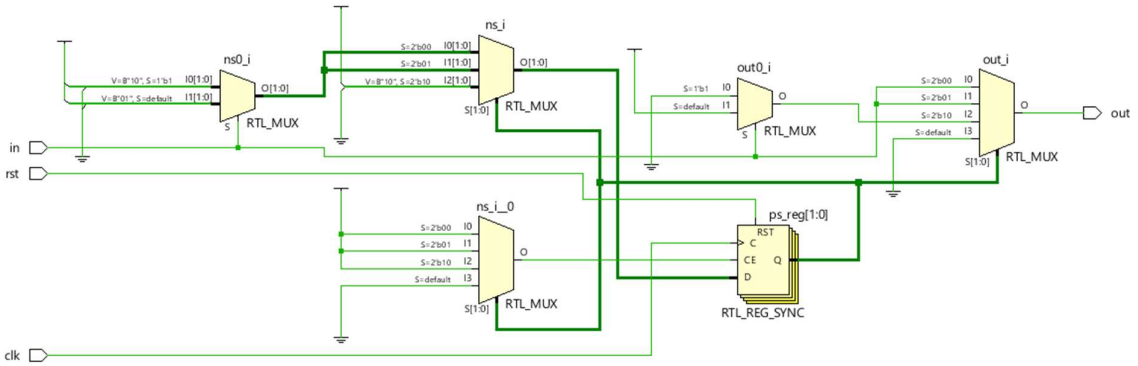
2s COMPLEMENT FSM

```
module complement_2s(
input clk,
input rst,
input in,
   output reg out
);
reg [1:0] ps, ns;
parameter s0 = 2'b00;
```

```
parameter s1 = 2'b01;
parameter s2 = 2'b10;
always@(posedge clk) begin
if (rst)
ps<=s0;
else
ps<=ns;
end
always@(*) begin
case(ps)
s0:begin
ns<=in?s2:s1;
out<=in?1:0;
end
s1:begin
ns<=in?s2:s1;
out<=in?1:0;
end
s2:begin
ns<=in?s2:s2;
out<=in?0:1;
end
default:begin
  ns<= ps;
  out<= 0;
  end
```

```
endcase
end
endmodule

module complement_2_tb();
 reg CLK, RST;
   reg IN;
   wire OUT;
   complement_2s dut(.clk(CLK), .rst(RST), .in(IN), .out(OUT));
   initial begin
      CLK = 0;
      forever #5 CLK = ~CLK;
   end
   initial begin
      RST = 1'b1; IN =1'b0; #10;
      RST = 1'b0;
      IN = 1'b0; #20;
      IN = 1'b1; #20;
      IN = 1'b0;#20;
IN = 1'b1;#20;
IN = 1'b0;#20;
IN = 1'b1;#100;
$stop();
$display("time=%0t clk=%b rst=%b in=%b out=%b", $time, CLK, RST, IN, OUT);
end
endmodule
```





1’s COMPLEMENT FSM

```
module complement_1s(clk,rst,in,out);
input clk,rst,in;
output reg out;
reg [1:0]ps,ns;
parameter s0=2'b00,s1=2'b01;
always@(posedge clk) //reset logic
begin
if(rst)
ps<=s0;
else
ps<=ns;
```

```
end
always@(*)
begin
case(ps)
s0:begin
ns<=in?s0:s1;
out<=in?0:1;
end
s1:begin
ns<=in?s0:s1;
out<=in?0:1;
end
endcase
end
endmodule

module complement_1s_tb();
reg CLK,RST,IN;
wire OUT;
complement_1s dut(CLK,RST,IN,OUT);
initial
begin
CLK=0;
forever #5 CLK=~CLK;
end
initial
```

```
begin
RST=1'b1; #10; RST=1'b1;
IN=1'b0; #10; IN=1'b0; #10;
IN=1'b1; #10; IN=1'b0; #10;
IN=1'b1; #10; IN=1'b1; #10;
IN=1'b0; #10;
$stop;
$display("CLK=%0b,RST=%0b,IN=%0b,OUT=%0b",CLK,RST,IN,OUT);
end
endmodule
```