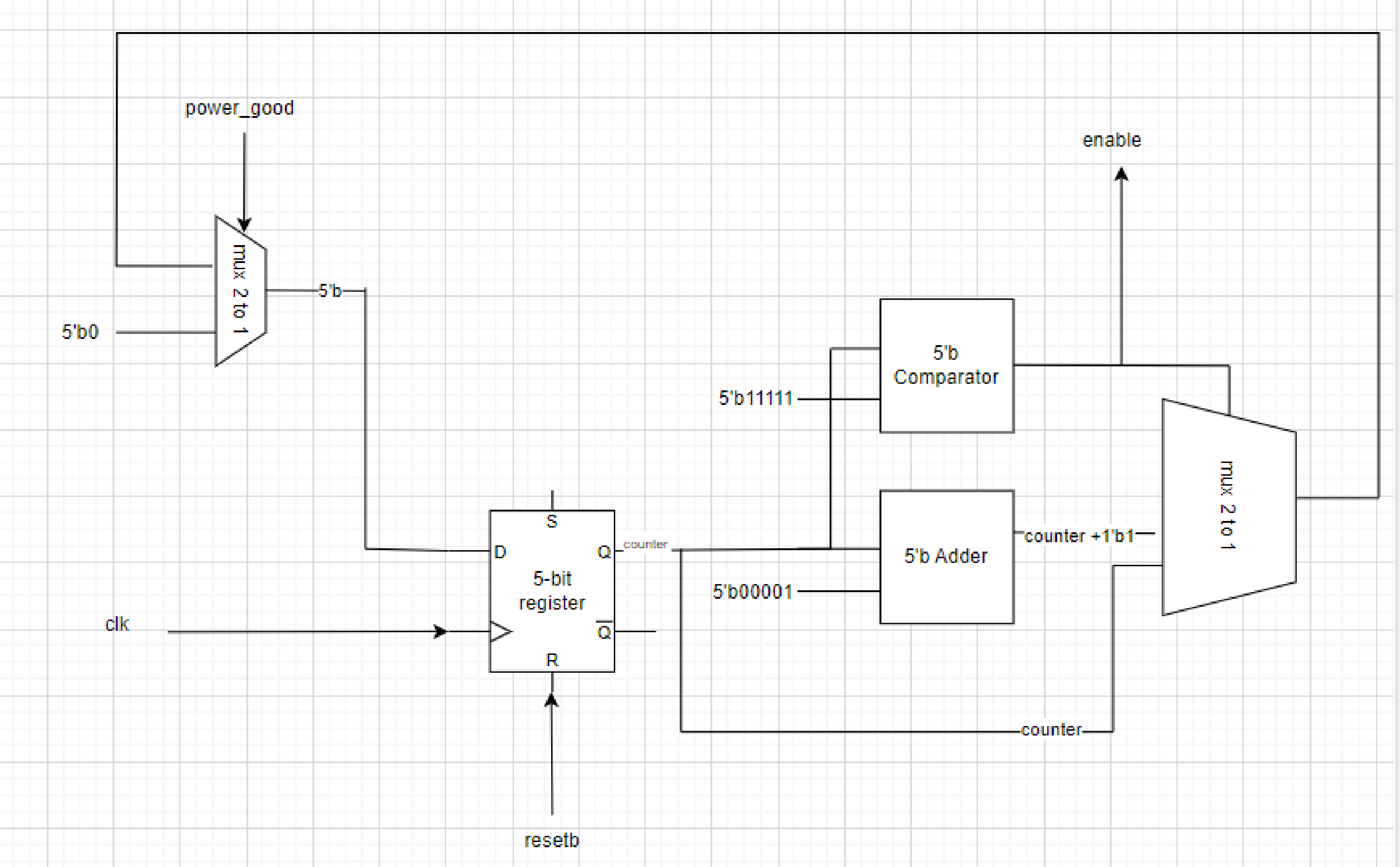
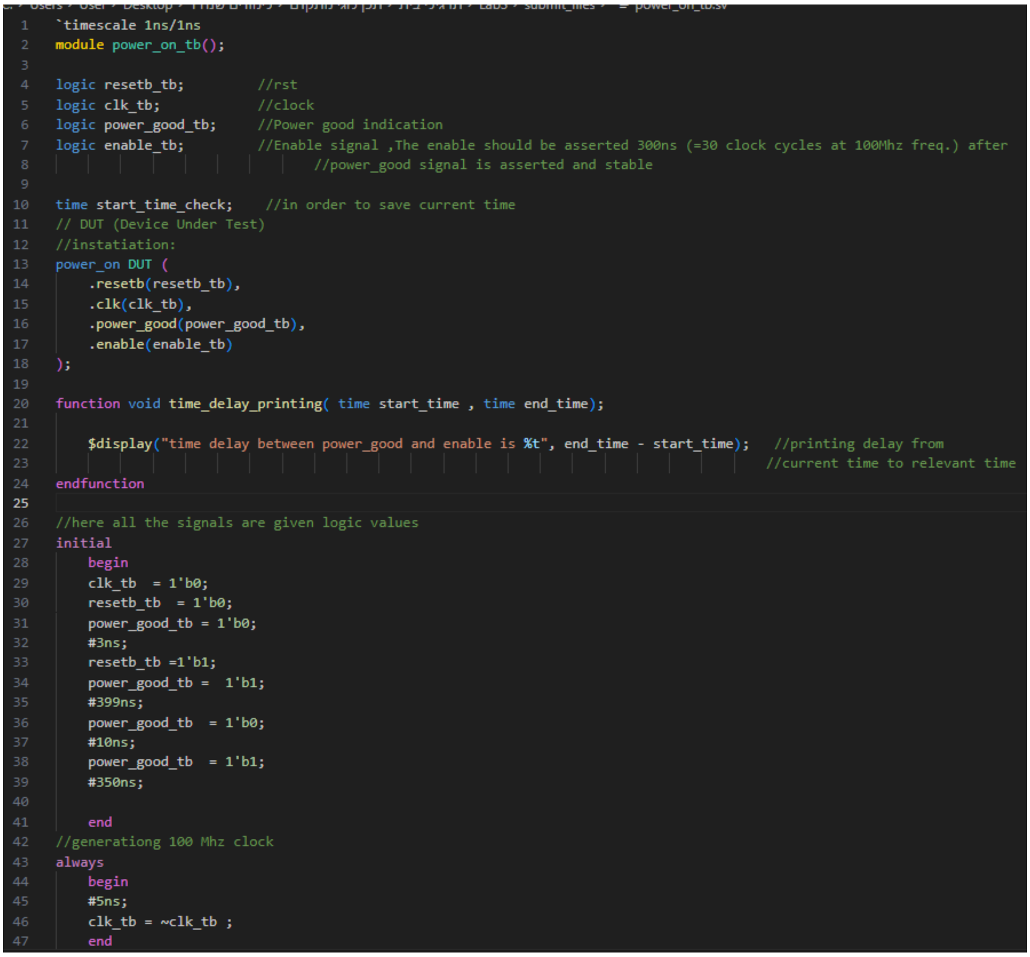
Advanced logic design - Lab3

Question 3:

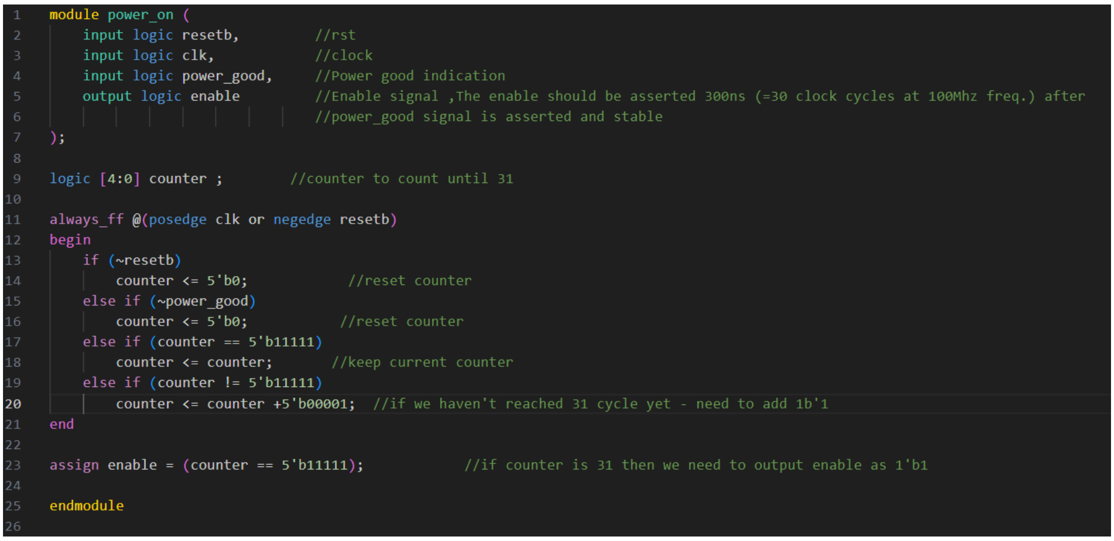
a: Block Diagram:



***B:** power_on.sv :*

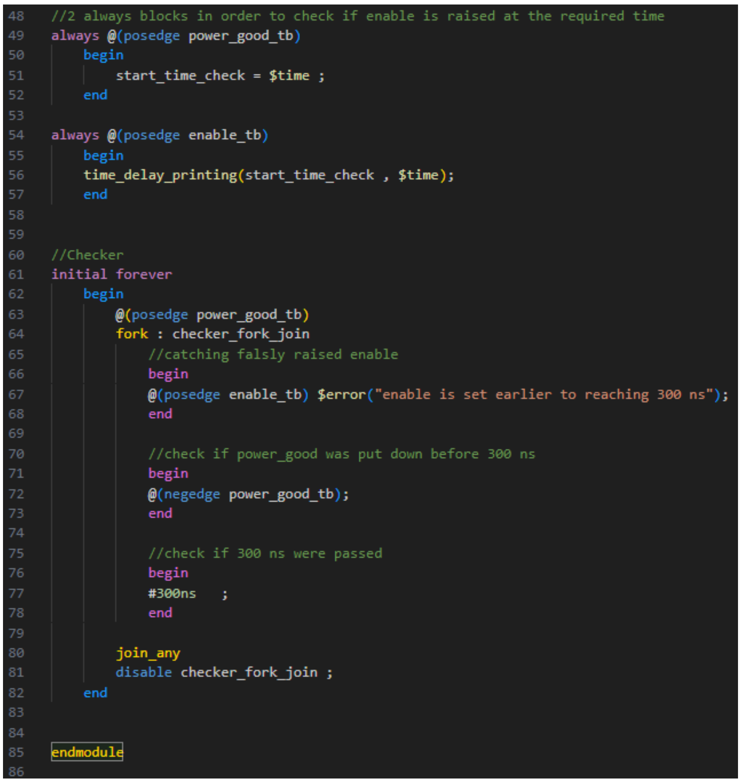


```systemverilog
`timescale 1ns/1ns
module power_on_tb();

logic resetb_tb;         //rst
logic clk_tb;            //clock
logic power_good_tb;     //Power good indication
logic enable_tb;         //Enable signal ,The enable should be asserted 300ns (=30 clock cycles at 100Mhz freq.) after
                                //power_good signal is asserted and stable

time start_time_check;    //in order to save current time
// DUT (Device Under Test)
//instatiation:
power_on DUT (
    .resetb(resetb_tb),
    .clk(clk_tb),
    .power_good(power_good_tb),
    .enable(enable_tb)
);

function void time_delay_printing( time start_time , time end_time);

    $display("time delay between power_good and enable is %t", end_time - start_time);   //printing delay from
                                                                                         //current time to relevant time
endfunction

//here all the signals are given logic values
initial
    begin
    clk_tb  = 1'b0;
    resetb_tb  = 1'b0;
    power_good_tb = 1'b0;
    #3ns;
    resetb_tb =1'b1;
    power_good_tb =  1'b1;
    #399ns;
    power_good_tb  = 1'b0;
    #10ns;
    power_good_tb  = 1'b1;
    #350ns;

    end
//generationg 100 Mhz clock
always
    begin
    #5ns;
    clk_tb = ~clk_tb ;
    end
```



```systemverilog
module power_on (
    input logic resetb,         //rst
    input logic clk,            //clock
    input logic power_good,     //Power good indication
    output logic enable         //Enable signal ,The enable should be asserted 300ns (=30 clock cycles at 100Mhz freq.) after
                                //power_good signal is asserted and stable
);

logic [4:0] counter ;        //counter to count until 31

always_ff @(posedge clk or negedge resetb)
begin
    if (~resetb)
        counter <= 5'b0;            //reset counter
    else if (~power_good)
        counter <= 5'b0;           //reset counter
    else if (counter == 5'b11111)
        counter <= counter;        //keep current counter
    else if (counter != 5'b11111)
        counter <= counter +5'b00001;  //if we haven't reached 31 cycle yet - need to add 1b'1
end

assign enable = (counter == 5'b11111);            //if counter is 31 then we need to output enable as 1'b1

endmodule
```

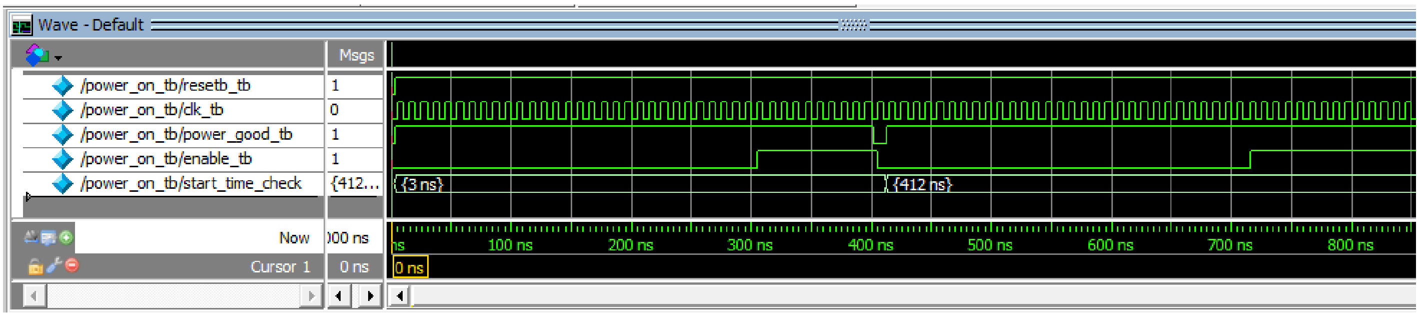
C: *power_on_tb.sv*



```
//2 always blocks in order to check if enable is raised at the required time
always @(posedge power_good_tb)
    begin
        start_time_check = $time ;
    end

always @(posedge enable_tb)
    begin
    time_delay_printing(start_time_check , $time);
    end


//Checker
initial forever
    begin
        @(posedge power_good_tb)
        fork : checker_fork_join
            //catching falsly raised enable
            begin
            @(posedge enable_tb) $error("enable is set earlier to reaching 300 ns");
            end

            //check if power_good was put down before 300 ns
            begin
            @(negedge power_good_tb);
            end

            //check if 300 ns were passed
            begin
            #300ns    ;
            end

        join_any
        disable checker_fork_join ;
    end


endmodule
```

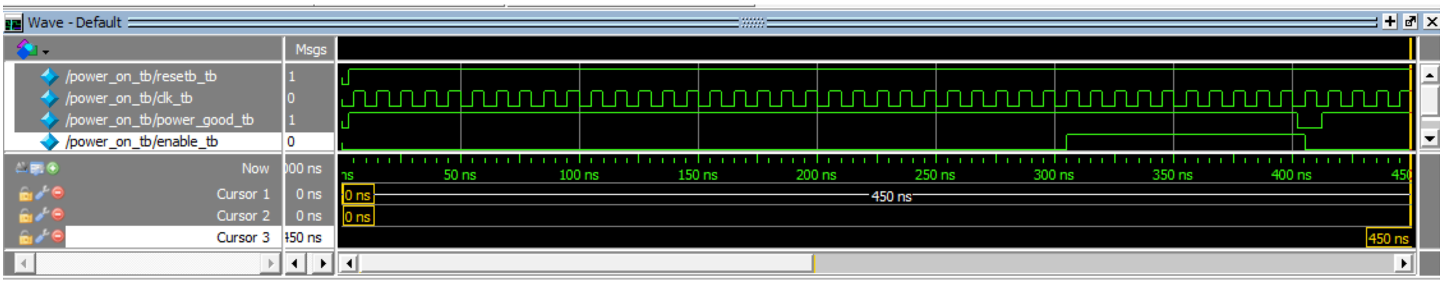
We have included waveform to prove the function correctness, when the difference between power_good and enable is less than 300 ns, we get error on the screen

Simulation zoom out:



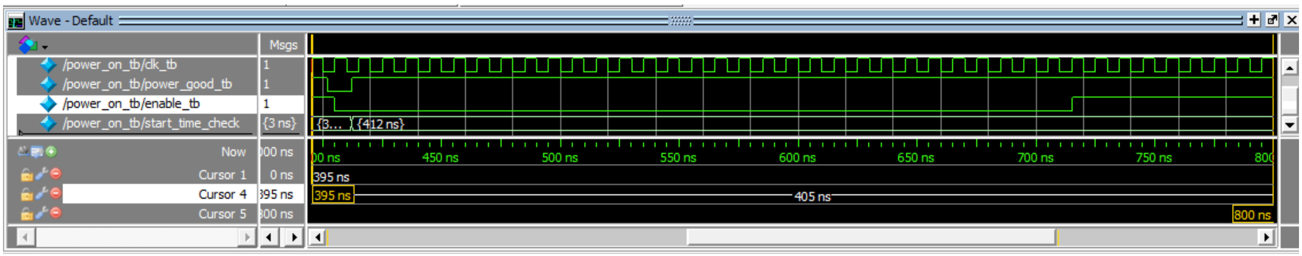
	Msgs
/power_on_tb/resetb_tb	1
/power_on_tb/clk_tb	0
/power_on_tb/power_good_tb	1
/power_on_tb/enable_tb	1
/power_on_tb/start_time_check	{412...

Zoom in: 0 - 450 ns :

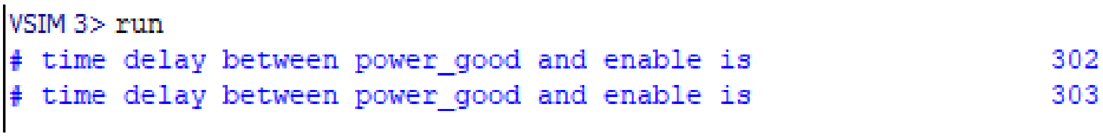


	Msgs
/power_on_tb/resetb_tb	1
/power_on_tb/clk_tb	0
/power_on_tb/power_good_tb	1
/power_on_tb/enable_tb	0

Zoom in: 395 - 800 ns :



Messages we got when running the simulation:

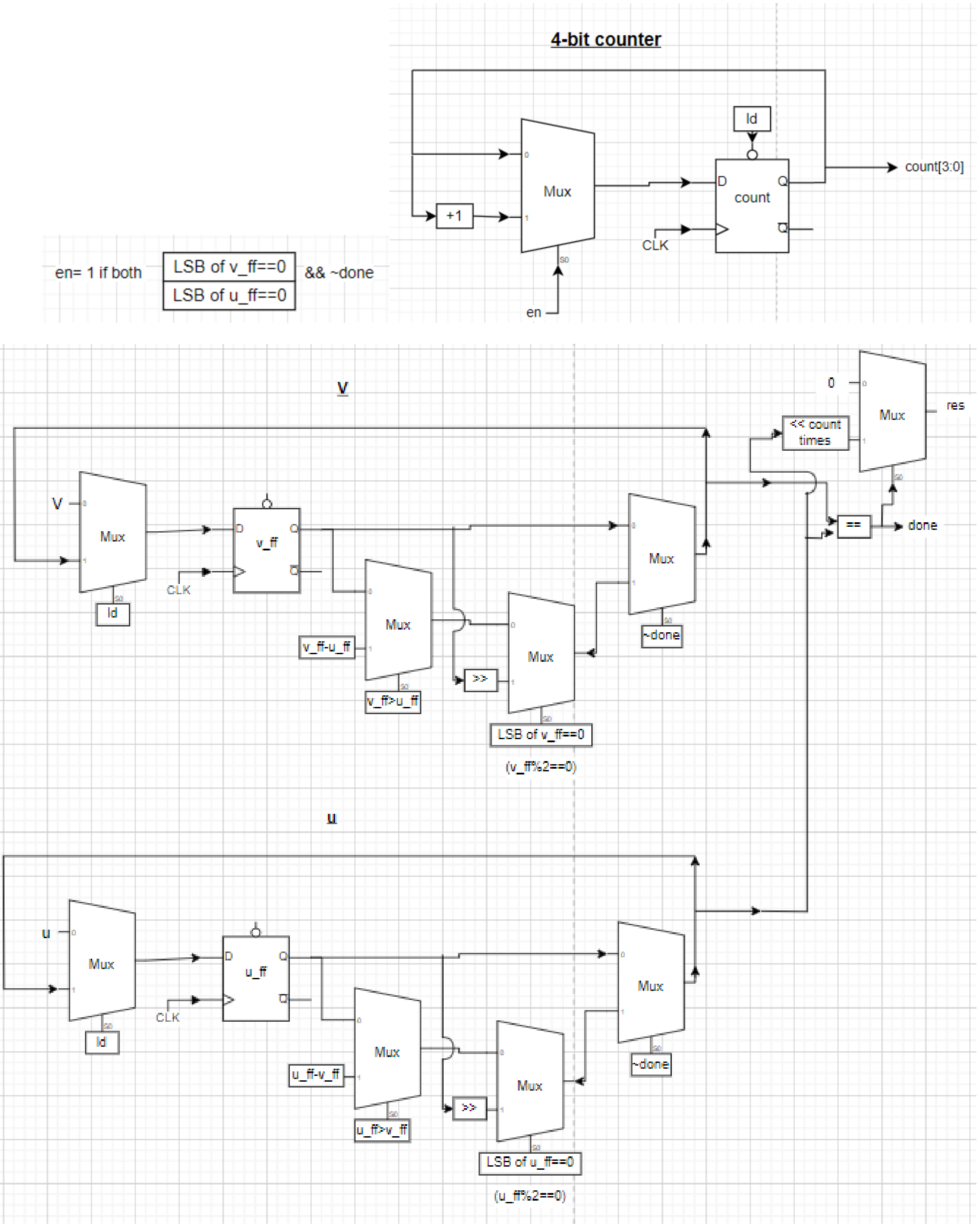


```
VSIM 3> run
# time delay between power_good and enable is                     302
# time delay between power_good and enable is                     303
```

The additional 2 and 3 ns are due to the fact that in the simulation we need to raise power_good a bit before the posedge rising of the clock, so the correct value will be sampled.

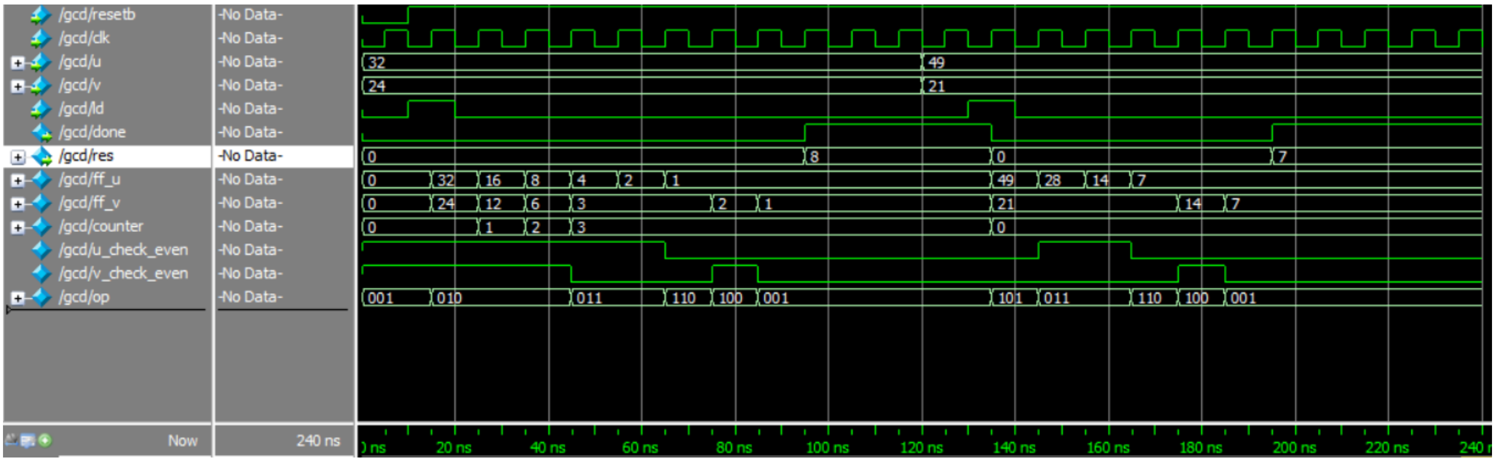
Question 4 – GCD:

A: Block Diagram



***B, C :** gcd.sv, gcd_tb.sv files are attached*

Waveform to prove the correctness of the design:


/gcd/resetb -No Data-
/gcd/clk -No Data-
/gcd/u -No Data- 32 49
/gcd/v -No Data- 24 21
/gcd/ld -No Data-
/gcd/done -No Data-
/gcd/res -No Data- 0 8 0 7
/gcd/ff_u -No Data- 0 32 16 8 4 2 1 49 28 14 7
/gcd/ff_v -No Data- 0 24 12 6 3 2 1 21 14 7
/gcd/counter -No Data- 0 1 2 3 0
/gcd/u_check_even -No Data-
/gcd/v_check_even -No Data-
/gcd/op -No Data- 001 010 011 110 100 001 101 011 110 100 001
Now 240 ns
) ns
20 ns
40 ns
60 ns
80 ns
100 ns
120 ns
140 ns
160 ns
180 ns
200 ns
220 ns
240 r