Mini-Project “Design and Implementation in Logisim and HDL”

Smart Commute

Team members:

1. 221CS223, HARSHINI.V, harshiniv.221cs223@nitk.edu.in , 9606982602

2. 221CS237, RANGALLA SANJANA, sanjanarangalla.221cs237@nitk.edu.in , 6360462299

3. 221CS259, VARAHI SUVARNA, varahisuvarna.221cs259@nitk.edu.in , 9008705532

Abstract:

Background:

In India, there can be a lack of stringent enforcement of regulations regarding passenger limits and overloading. In some cases, buses are allowed to operate despite exceeding their capacity. In recent years, the transportation landscape has undergone significant transformations driven by technological advancements, evolving passenger expectations, and the pressing need for sustainable mobility solutions. In this context, our project seeks to address critical challenges and opportunities within the public transportation

sector, with a focus on enhancing the overall passenger experience, safety, and efficiency of bus services. We'll keep a counter that keeps track of the number of people entering the bus and display the number of free seats to the passengers.

Motivation:

Ensuring the comfort and safety of bus passengers is paramount. Our project acknowledges the significance of creating a secure and pleasant environment for all travellers. As the world grapples with environmental concerns, public transportation plays a pivotal role in reducing emissions and promoting sustainability. Our project aligns with these global sustainability objectives. We plan to install energy efficient lights that adjust according to the time of the day.

Unique Contribution:

1. Safety Rules: The system shall prevent passengers from standing near the door when available seats are unoccupied. The bus shall not start in such a case until the person sits down.

2. Power Saving: When the bus is empty, the electrical appliances like lights inside the bus will be turned off leading to saving the power.

3. Persistent Overload Alert: Our bus will not start in case any of the conditions go false, which makes for an efficient way to make sure the rules are obeyed.

4. Regulatory Compliance: Our system includes a passenger counter to display current occupancy, aiding compliance with stringent safety regulations and preventing legal issues.

5. Real-time Occupancy Awareness: Passengers receive a live count of occupants, empowering them to assess safety.

Brief Description:

Idea:

To ensure safe and secure travel experience in local bus transportation by implementing the following conditions:

1. Sending warning signals when bus gets overloaded.

2.Displaying number of vacant seats.

3.Checking the ID's of passengers to avoid any illegal practices.

4.Giving a warning signal when someone is standing near the door.

5.Conserving energy by switching off lights if no one is inside the bus.

Existing issues:

1. Road accidents due to overloading.

2.Accidents due to people standing near the exit.

3.Unauthorized people travelling inside the public transportation.

Approach:

1.Using combinational and sequential circuits to design a model to count the number of passengers therefore avoiding overloading situations.

2.Displaying the number of vacant seats in the bus by using a subtractor and subtracting the counter value from the maximum number (30) in the bus.

3.Using two counters and checking if someone is standing at the door.

4.Verifying the ID's of the passengers with those ID's already present in the database.

Design:

1. The passenger should have primarily registered and should have the bus card which consists of ID which will be verified at the door of the bus before entering (manually entered in project). The 'ID VERIFIER' block consists of an 8- bit comparator which compares the two ids and return 1 if they are equal, as shown in Fig(1).

2. After verification if card is valid then the person can enter the bus, else a red light would be turned on to indicate the invalid passenger.

3. Two counters are used to keep track of number of people inside the bus. The 'EMPTY BUS' block compares the count from the second counter and checks if its equal to 0, if it is then the lights will be turned off as there are no people inside. The block uses a 5-bit comparator to compare the values of the output of the counter with a fixe input – 0.

4. The 'PERSON-ON-STEPS' block takes the input from both the counters as compares them using a 5-bit comparator. It returns 1 if both are equal.

5. After each entry of passenger the number of vacant seats as per maximum capacity of the bus (maximum limit is set to 30 in this project) is displayed outside the bus, using the 'DISPLAY' block. A 5-bit subtractor is used to calculate the number of vacancies or empty seats.

6. If the maximum limit is exceeded the bus will not move. In the 'OVERLOADING-CHECK' block, a 5 - bit comparator checks if the count of people inside is less than the maximum value (30) or equal to the maximum value, in such a case the bus will be allowed to operate.

7. Finally, all the three outputs are taken as inputs to and gate, which will give 1 only when all three are 1, i.e. when all the conditions are satisfied, only then the bus will move.

8. There are also 3 LED'S which turn on when any one of the conditions is not satisfied causing the bus to stop connected via 3 NOT gates.

8. The validity of the bus cards should be renewed every year.

WORKING:

Truth Tables :-

OR GATE

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

AND GATE

A	B	C	D
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

5 – Bit Comparator

A4B4	A3B3	A2B2	A1B1	A0B0	A<B	B==A	A>B
A4>B4	X	X	X	X	0	0	1
A4<B4	X	X	X	X	1	0	0
A4==B4	A3>B3	X	X	X	0	0	1
A4==B4	A3<B3	X	X	X	1	0	0
A4==B4	A3==B3	A2>B2	X	X	0	0	1
A4==B4	A3==B3	A2<B2	X	X	1	0	0
A4==B4	A3==B3	A2==B2	A1>B1	X	0	0	1
A4==B4	A3==B3	A2==B2	A1<B1	X	1	0	0
A4==B4	A3==B3	A2==B2	A1==B1	A0>B0	0	0	1
A4==B4	A3==B3	A2==B2	A1==B1	A0<B0	1	0	0
A4==B4	A3==B3	A2==B2	A1==B1	A0==B0	0	1	0

D Flip-Flop 5-bit Synchronous Up-Down Counter

M	Present State	Next State	D4	D3	D2	D1	D0
1	0	1	0	0	0	0	1
1	1	2	0	0	0	1	0
1	2	3	0	0	0	1	1
1	3	4	0	0	1	0	0
1	4	5	0	0	1	0	1
1	5	6	0	0	1	1	0
1	6	7	0	0	1	1	1
1	7	8	0	1	0	0	0
1	8	9	0	1	0	0	1
1	9	10	0	1	0	1	0
1	1	11	0	1	0	1	1

1	11	12	0	1	1	0	0
1	12	13	0	1	1	0	1
1	13	14	0	1	1	1	0
1	14	15	0	1	1	1	1
1	15	16	1	0	0	0	0
1	16	17	1	0	0	0	1
1	17	18	1	0	0	1	0
1	18	19	1	0	0	1	1
1	19	20	1	0	1	0	0
1	20	21	1	0	1	0	1
1	21	22	1	0	1	1	0
1	22	23	1	0	1	1	1
1	23	24	1	1	0	0	0
1	24	25	1	1	0	0	1
1	25	26	1	1	0	1	0
1	26	27	1	1	0	1	1
1	27	28	1	1	1	0	0
1	28	29	1	1	1	0	1
1	29	30	1	1	1	1	0
1	30	31	1	1	1	1	1
1	31	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	2	1	0	0	0	0	1
0	3	2	0	0	0	1	0
0	4	3	0	0	0	1	1
0	5	4	0	0	1	0	0
0	6	5	0	0	1	0	1
0	7	6	0	0	1	1	0
0	8	7	0	0	1	1	1
0	9	8	0	1	0	0	0
0	10	9	0	1	0	0	1
0	11	10	0	1	0	1	0
0	12	11	0	1	0	1	1
0	13	12	0	1	1	0	0
0	14	13	0	1	1	0	1
0	15	14	0	1	1	1	0
0	16	15	0	1	1	1	1
0	17	16	1	0	0	0	0
0	18	17	1	0	0	0	1
0	19	18	1	0	0	1	0
0	20	19	1	0	0	1	1
0	21	20	1	0	1	0	0
0	22	21	1	0	1	0	1
0	23	22	1	0	1	1	0

0	24	23	1	0	1	1	1
0	25	24	1	1	0	0	0
0	26	25	1	1	0	0	1
0	27	26	1	1	0	1	0
0	28	27	1	1	0	1	1
0	29	28	1	1	1	0	0
0	30	29	1	1	1	0	1
0	31	30	1	1	1	1	0
0	0	31	1	1	1	1	1

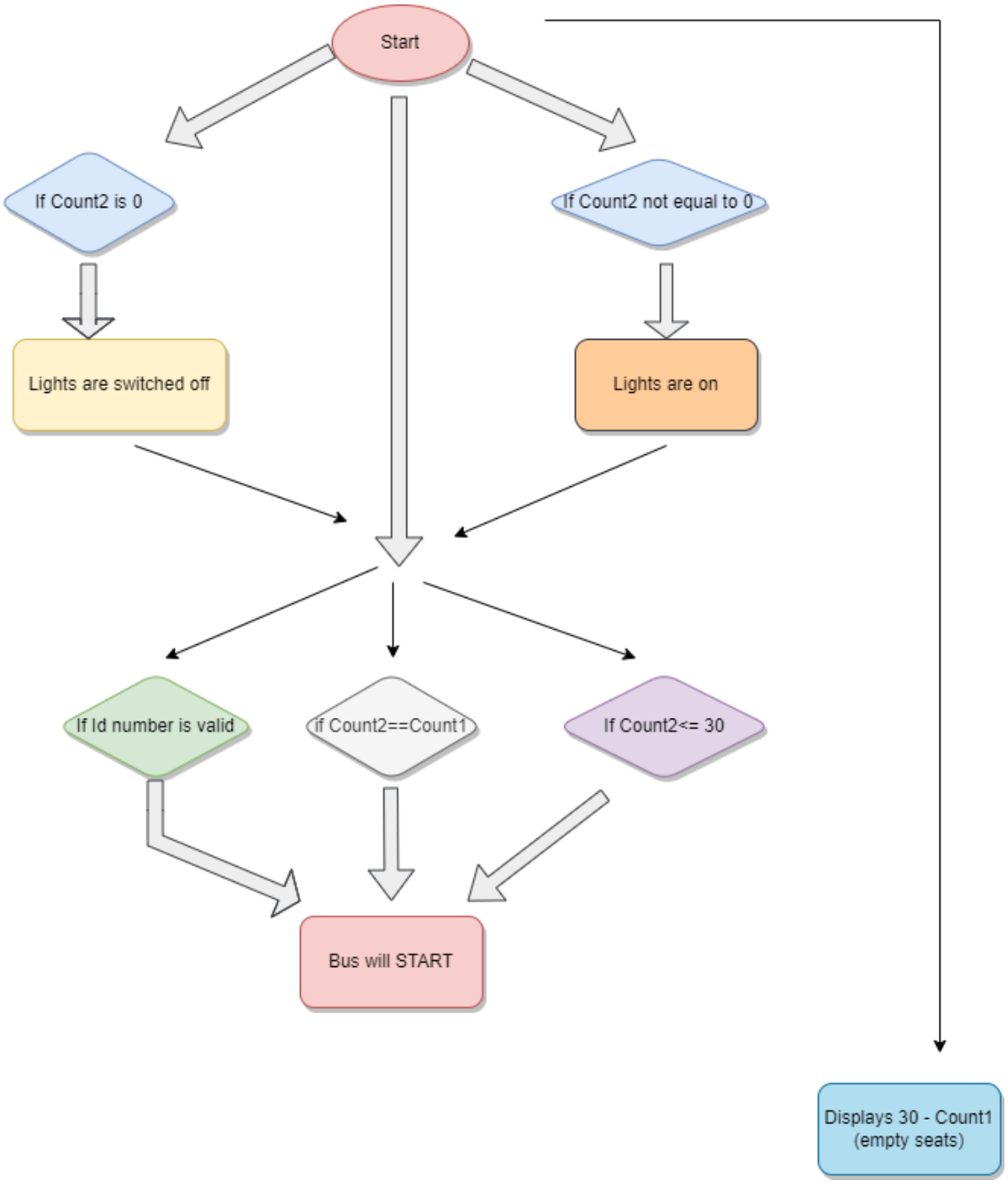
Full Subtractor

A	B	C	Difference	Borrow
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

NOT GATE

A	B
1	0
0	1

Flow Chart: -


Start
If Count2 is 0
If Count2 not equal to 0
Lights are switched off
Lights are on
If Id number is valid
if Count2==Count1
If Count2<= 30
Bus will START
Displays 30 - Count1
(empty seats)

Logisim Circuit Diagram:-

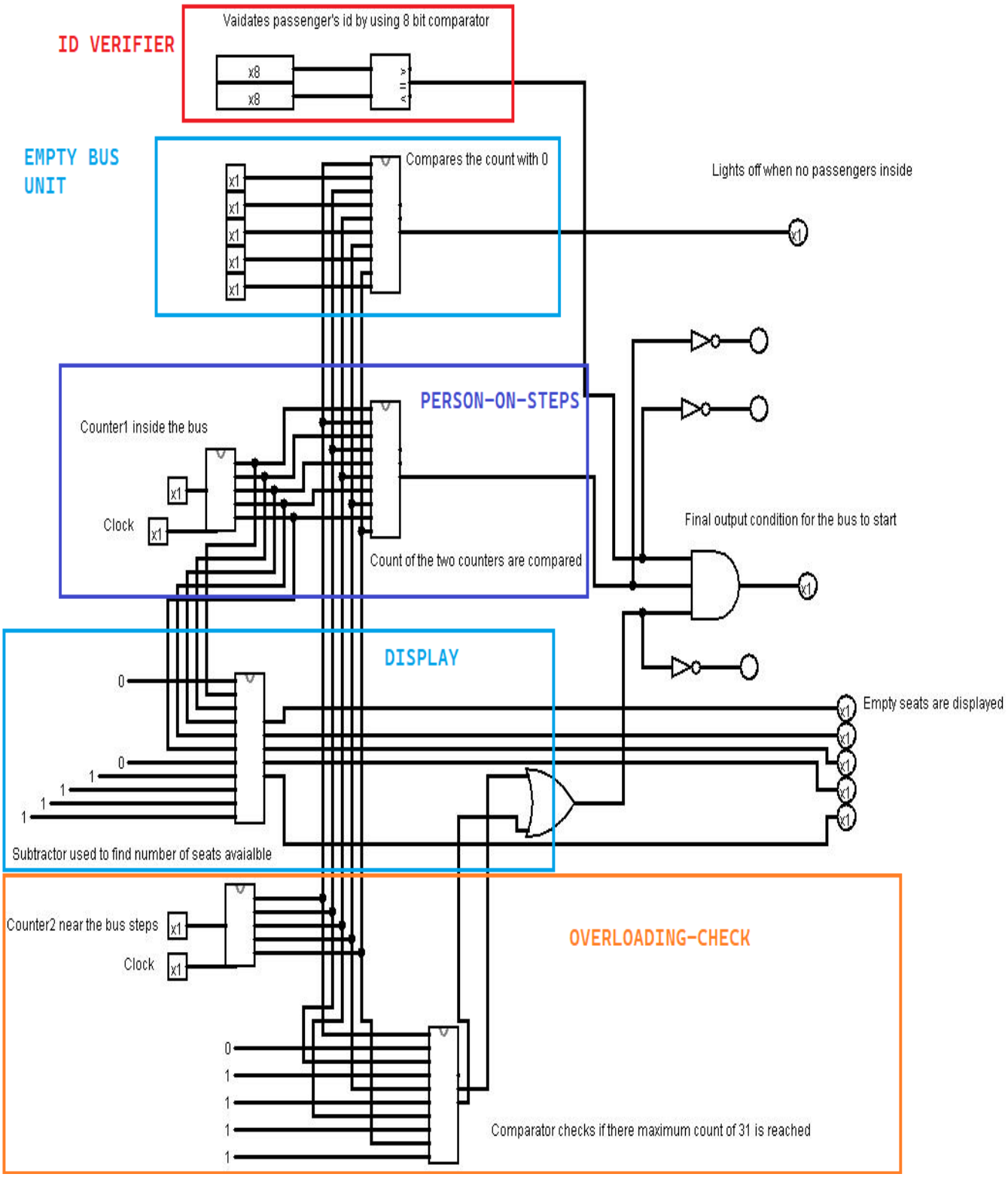


Fig-1.

Verilog Code:

main.v (testbench)

```
`include "comparator.v"
`include "bus.v"
`include "sub.v"
`include "comp2.v"
`include "finst.v"
`include "lig.v"
`include "id.v"

module main;

        reg Clk;reg Clk2;
        reg reset;reg reset2;reg [7:0] id1;reg [7:0] vaid;
        reg UpOrDown;reg UpOrDown2;

        wire x;wire [4:0] z;wire k;wire light;
        wire [4:0] Count;wire [4:0] Count2;
        wire finout;wire val_id;

        upordown_counter test(Clk,reset,UpOrDown,Count); // innermost counter
        upordown_counter test2(Clk2,reset2,UpOrDown2,Count2);

        sub test3(Count,Count2,z);// z is ppl standing

        comp2 test4(z,k);// k =1 if its 0 ppl standing
        comparator tests(Count,x);// x is 1 means 31 is reached
        id test6(id1,vaid,val_id);

        finst test5(x,k,val_id,finout);
        lig test7(Count2,light); // light is 0 if Count2=0
```

```
       initial begin
              Clk = 1;Clk2 = 1;
              forever #1 begin Clk = ~Clk; Clk2 = ~Clk2; end
       end

       initial begin
              $dumpfile("main.vcd");
              $dumpvars(0,main);

              $display("   User Id ||Valid Id||  Count1 ||   Count2 ||  Light   || Output ");
              $monitor("   %b    |  %b |   %b |    %b    |    %b    |    %b  ",id1,vaid,Count,Count2,light,finout);

       // Apply Inputs
              reset = 0;reset2=0;
              UpOrDown = 1;UpOrDown2=1;id1=01101101;vaid=01101101;
              #5;
              reset=1;
              UpOrDown = 0;UpOrDown2=1;id1=01101010;vaid=01100010;
              #5;
              reset = 0;reset2 = 0;
              UpOrDown = 1;UpOrDown2 = 1;id1=01000101;vaid=01010101;
              #5;
              reset = 0;  reset2=1;id1=00011111;vaid=00011111;
              #5;
       $finish;
   end

endmodule
```

comparator.v

```
//gives output as 1 if maximum value of passengers is exceeded(30 and becomes 31) are same
module comparator(A,x);
        input [4:0]A;

        output x;
        wire a,b,c,d,e;
        xnor(a,A[4],1);
        xnor(b,A[3],1);
        xnor(c,A[2],1);
        xnor(d,A[1],1);
        xnor(e,A[0],1);

        assign x=a&b&c&d&e;

endmodule
```

bus.v (counter)

```
// 5 bit up-down synchronous counter
module upordown_counter(
    Clk,
    reset,
    UpOrDown,  //high for UP counter and low for Down counter
    Count
    );


    //input ports and their sizes
    input Clk,reset,UpOrDown;
    //output ports and their size
    output [4 : 0] Count;
    //Internal variables
    reg [4 : 0] Count = 0;
    //output reg[4:0] occupancy;
     always @(posedge(Clk) or posedge(reset))
     begin
        if(reset == 1) begin
            Count <= 0;

        end else begin
            if(UpOrDown == 1) begin   //Up mode selected
                if(Count == 30 | Count == 31 ) begin

                    Count <= 0;
                end else begin
                    Count <= Count + 1; //Incremend Counter

                    end
```

```
                end else begin  //Down mode selected
                if(Count == 0) begin
                    Count <= 31;


                end else begin
                    Count <= Count - 1; //Decrement counter


     end
end
end
end


endmodule
```

sub.v (subtractor)

```
// counts the number of people standing near the door by subtracting the values in the two counters
module sub(input [4:0] a, input [4:0] b, output reg [4:0] res );

        always @* begin
                res=a-b;
        end
endmodule
```

comp2.v (returns 1 if there are 0 people standing)

```
// returns 1 if there are 0 people standing
module comp2(A,x);
        input [4:0]A;
        //input [4:0]B;
        output x;
        wire a,b,c,d,e;
        xnor(a,A[4],0);
        xnor(b,A[3],0);
        xnor(c,A[2],0);
        xnor(d,A[1],0);
        xnor(e,A[0],0);

        assign x=a&b&c&d&e;

endmodule
```

finst.v(final output of the bus state – stop or start)

```
// decides if the bus should finally be start or stop based on the 3 inputs
module finst(input x,input k,input y, output finout);
        assign finout = ~x&k&y;


endmodule
```

lig.v(light is off when there are no people inside)

```
// decides if light should be off if there are 0 people and on otherwise
module lig(input [4:0] count, output reg light);
        always @*
        begin
                if(count==0)
                        light<=0;

                else
                        light<=1;
        end

endmodule
```

Id.v(compares the 2 ids)

```
// returns 1 as output if both the ids - id in database and the passengers id match
module id(input [7:0] id, input [7:0] vaid, output reg y);

        always@(vaid)
                begin
                if(id == vaid)
                        y<=1;
                else
                        y<=0;

        end

endmodule
```

References:

1. https://www.javatpoint.com/verilog-behavioral-modelling-and-timing#:~:text=In%20Verilog%2C%20Behavioral%20models%20contain,activity%20flow%20associated%20with%20it

2 :https://www.youtube.com/watch?v=kcIyq2fiXtI

3. https://www.youtube.com/watch?app=desktop&v=wsVD-gjdanM

4. https://www.fpga4student.com/2017/03/verilog-code-for-counter-with-testbench.html?m=1

5. https://www.geeksforgeeks.org/synchronous-3-bit-up-down-counter/

*****END*****