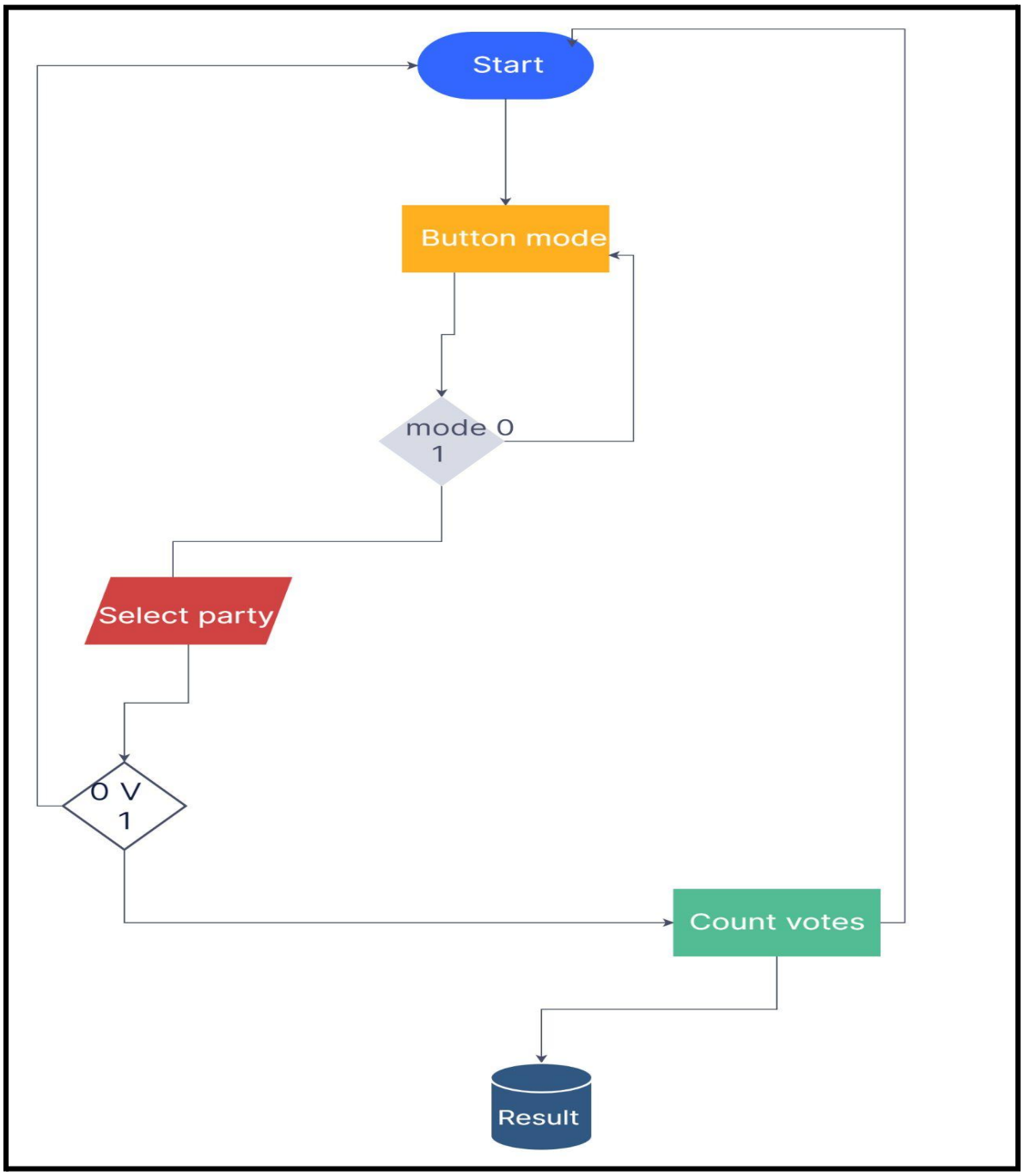
VERILOG CODE:

FLOWCHART & Skeletal functioning:



In the flowchart, The following execution is taking place in reference to the verilog programme written for Voting machine. Basically, a behavioral modeling style has been implemented to overcome the execution procedure.

Start:

Here the programme starts executing or reading the main module. A module is a block of Verilog code that implements certain functionality. Modules can be embedded within other modules, and a higher level module can communicate with its lower-level modules using their input and output ports.

Button mode:

In the voting machine module, there is always a button mode associated with voting buttons so that candidates may vote only at that level or period the button is on. Technically, in verilog votes input will be accessible if button mode is on.

mode:

After button mode procedure, mode will be checked out, so that no more than candidates are allowed to give multiple votes at a time.

Select party:

It is the input portion, where parties are selected by each candidate one at a time. In verilog, it has been designated so that no two votes could be recorded at the same button mode. Moreover, not more than one vote for the same party. For this verification a Valid **V** is also associated there.

Count votes:

Once the votes are recorded, it is necessary to count them. so a counter is there to increment the vote counter every instant the vote triggered for a given party. Counters are used in digital electronics for counting purposes, they can count specific events happening in the circuit. For example, in the UP counter a counter increases the count for every rising edge of the clock.

RESULT:

This is the database where the final result will be stored and could be utilized for displaying. This will be enabled after the voting procedure.

The voting process in the first phase begins with the input clock signal (upto 100 MHz). At each positive edge of the clock signal the voting_en signal is synchronized. Voting_en is an active high signal which when enabled then the voting process begins. This signal plays a key role in election process because the vote casted by the voter is valid if and only if this voting_en signal is made high. So, this signal can be controlled by a polling officer at the polling booth. Voter_switch is another input signal used in our design. This signal determines the total no. of parties contesting in the election process, in the present design it [3 down to 0] switch. This signal allocates a switch to each party contesting in the election process. In the proposed design we have taken three parties to contest the election process. At the point when the voter is prepared to make his choice the surveying official ought to confirm whether the clock and voting_en the two signals are high, now the voter can cast his vote of his own choice by enabling the corresponding Voter_switch signal. Now, after casting the vote the output of the corresponding party glows which indicates that his vote has been casted successfully. In the present design we have allocated three led‟s to three parties who are contesting in the elections. So whenever the clock signal is high and Voting_en is high and say Voter_switch [0] is enabled the corresponding op led [0] glows. This process continues for the entire election process until the voting process concludes that all the voters in that polling booth cast their votes to the parties of their choice.Now, once when the entire process completes, the dout register consists of the total no of valid votes polled in the election process. This helps in knowing the total votes polled. The party, party1 and party2 registers will contain the votes polled to the corresponding parties individually in order to determine the winner.

VERILOG CODE:

```
//`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
/
// Company:
// Engineer:
//
// Create Date:    21:05:58 04/09/2022
// Design Name:
// Module Name:    EVM
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
/
module
EVM(clk,button_control,tally,reset,part_y,vote_s0,vote_s1,vote_s2,vote_s3,vote_s4
,vote_s5,vote_s6,vote_s7,vote_s8,vote_s9);
input clk,button_control,reset,tally;
```

```
input [9:0] part_y;

output [3:0]
vote_s0,vote_s1,vote_s2,vote_s3,vote_s4,vote_s5,vote_s6,vote_s7,vote_s8,vote_s9;

reg
vote_s0,vote_s1,vote_s2,vote_s3,vote_s4,vote_s5,vote_s6,vote_s7,vote_s8,vote_s9;
// make them of 4 bits



reg flag;


always @(posedge part_y or button_control)

begin

case(part_y)

10'b0000000001:flag=1;

10'b0000000010:flag=1;

10'b0000000100:flag=1;

10'b0000001000:flag=1;

10'b0000010000:flag=1;

10'b0000100000:flag=1;

10'b0001000000:flag=1;

10'b0010000000:flag=1;

10'b0100000000:flag=1;

10'b1000000000:flag=1;

default:flag=0;

endcase

end
```

```
reg [3:0] vote_counter0;
reg [3:0] vote_counter1;
reg [3:0] vote_counter2;
reg [3:0] vote_counter3;
reg [3:0] vote_counter4;
reg [3:0] vote_counter5;
reg [3:0] vote_counter6;
reg [3:0] vote_counter7;
reg [3:0] vote_counter8;
reg [3:0] vote_counter9;   // maximum 100 voters are available

always @(posedge clk)
begin
 if(reset)
     begin
     vote_counter0 =4'b000;
      vote_counter1 =4'b000;
       vote_counter2 =4'b000;
         vote_counter3 =4'b000;
          vote_counter4 =4'b000;
           vote_counter5 =4'b000;
```

```
            vote_counter6 =4'b000;
             vote_counter7 =4'b000;
              vote_counter8 =4'b000;
                vote_counter9 =4'b000;
     end
end

always @(button_control)
begin
    if(button_control && flag)
    begin

    if(part_y[0]==1 && vote_counter0<100)
    begin
    vote_counter0=vote_counter0 + 1;
    $display("A");
    end

    else if(part_y[1]==1 && vote_counter1<100)
    begin
    vote_counter1=vote_counter1 + 1;
    $display("B");
    end

    else if(part_y[2]==1 && vote_counter2<100)
    begin
```

```
vote_counter2=vote_counter2 + 1;
$display("C");
end

else if(part_y[3]==1 && vote_counter3<100)
begin
vote_counter3=vote_counter3 + 1;
$display("D");
end

else if(part_y[4]==1 && vote_counter4<100)
begin
vote_counter4=vote_counter4 + 1;
$display("E");
end

else if(part_y[5]==1 && vote_counter5<100)
begin
vote_counter5=vote_counter5 + 1;
$display("F");
end

else if(part_y[6]==1 && vote_counter6<100)
begin
vote_counter6=vote_counter6 + 1;
$display("G");
```

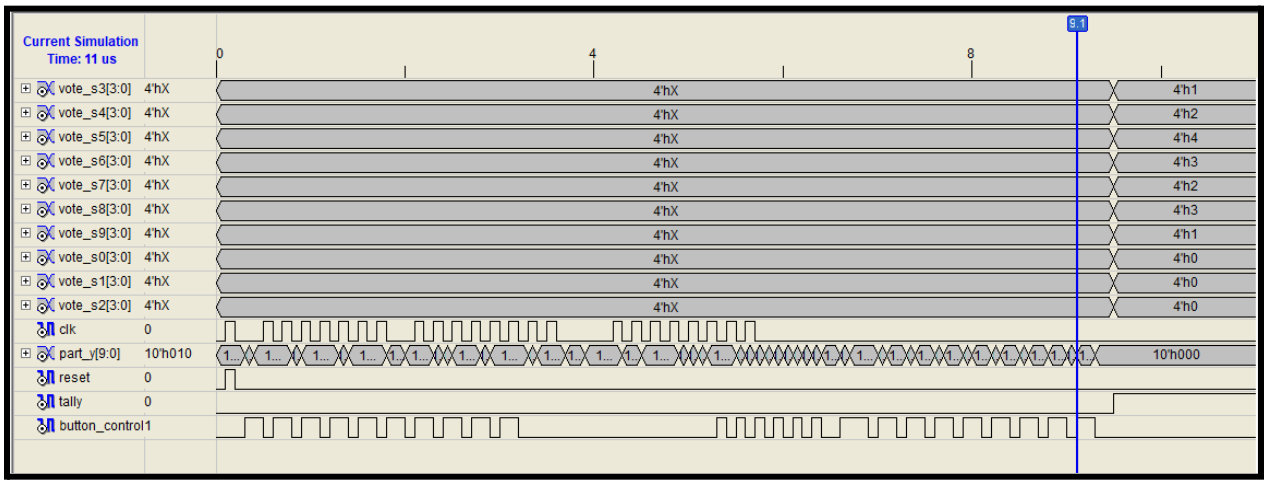
```
end

else if(part_y[7]==1 && vote_counter7<100)
begin
vote_counter7=vote_counter7 + 1;
$display("H");
end

else if(part_y[8]==1 && vote_counter8<100)
begin
vote_counter8=vote_counter8 + 1;
$display("I");
end

else if(part_y[9]==1 && vote_counter9<100)
begin
vote_counter9=vote_counter9 + 1;
$display("J");
end

else
begin
if(!flag)
$display("No two parties can be selected at the same moment!!");
end
 end
```

```
        end


initial         // vote_s calculation
begin
    if(!flag)
    $display("No two parties can be selected at the same moment!!");
    end


always @(posedge tally)
begin
vote_s0<=vote_counter0;
vote_s1<=vote_counter1;
vote_s2<=vote_counter2;
vote_s3<=vote_counter3;
vote_s4<=vote_counter4;
vote_s5<=vote_counter5;
vote_s6<=vote_counter6;
vote_s7<=vote_counter7;
vote_s8<=vote_counter8;
vote_s9<=vote_counter9;


end


endmodule
//done!!
```

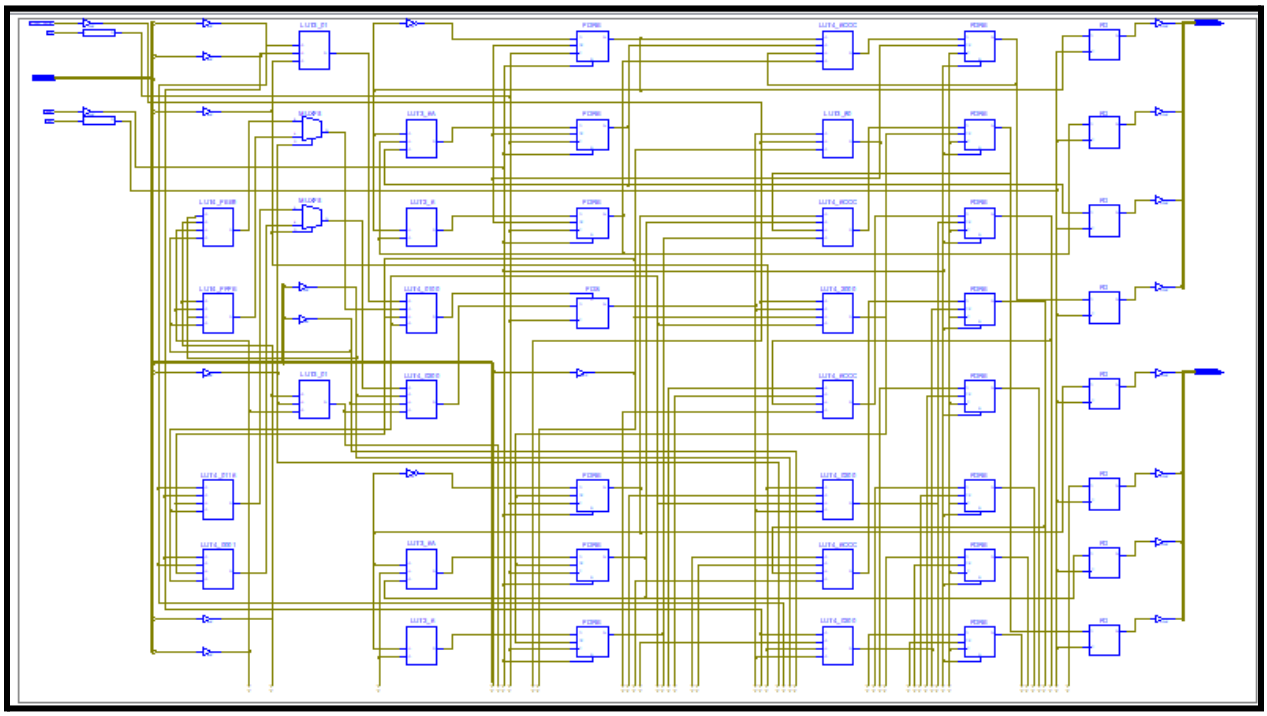
```
module voting_test;

reg clk,button_control,reset,tally;

reg [9:0] part_y;

wire [3:0]
vote_s0,vote_s1,vote_s2,vote_s3,vote_s4,vote_s5,vote_s6,vote_s7,vote_s8,vote_s9;

EVM uut
(clk,button_control,tally,reset,part_y,vote_s0,vote_s1,vote_s2,vote_s3,vote_s4,vo
te_s5,vote_s6,vote_s7,vote_s8,vote_s9);

initial
begin
  clk=0;
  button_control=0;
  tally=0;
  reset=1;

  part_y=10'b0000000100;
```

```
$dumpfile("voting_test.vcd");
$dumpvars(0,voting_test);


end


always #5 clk=~clk;
always #10 button_control=~button_control;
initial #400 tally=1;
initial
begin
  part_y=10'b0000100000;
 #50 part_y=10'b0100000000;
 #50 part_y=10'b0000100100;
 #50 part_y=10'b0000100000;
  #50 part_y=10'b0000001000;
 #50 part_y=10'b0000000010;
  #50 part_y=10'b1000000000;
 #50 part_y=10'b0000000001;
end
initial #20 reset=0;
initial #450 $finish;




endmodule
```

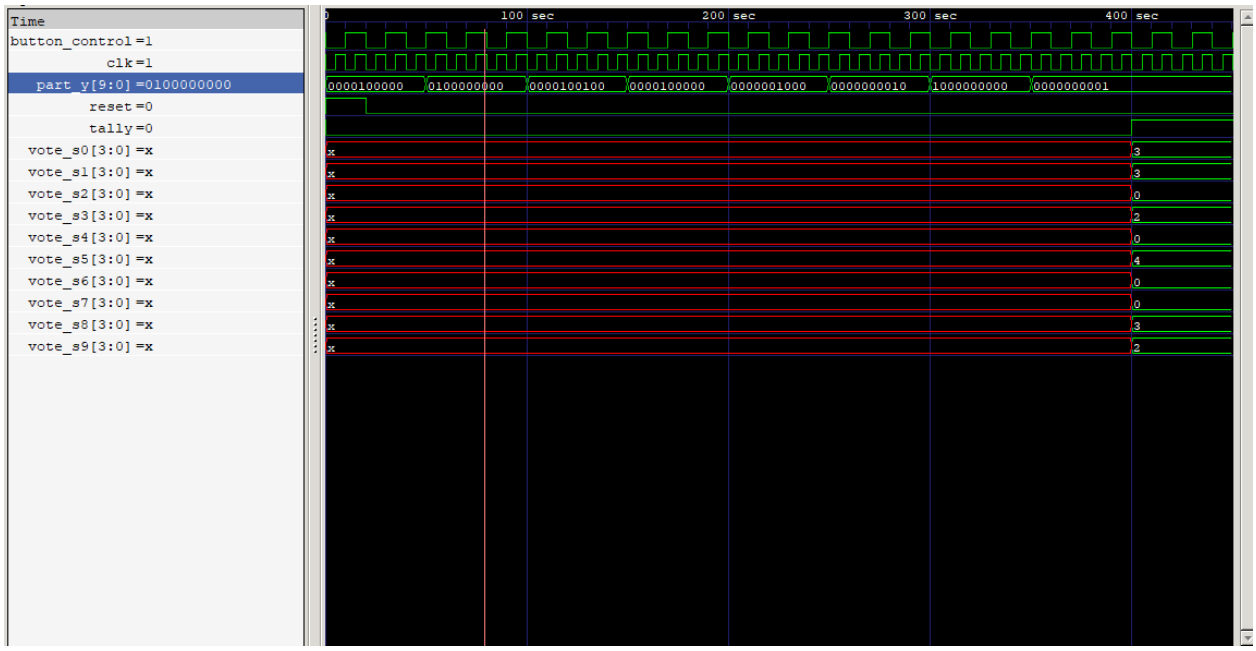
SIMULATION RESULT:



SYNTHESIS RESULT:



GTKWAVE:


Time
button_control =1
clk =1
part_y[9:0] =0100000000
reset =0
tally =0
vote_s0[3:0] =x
vote_s1[3:0] =x
vote_s2[3:0] =x
vote_s3[3:0] =x
vote_s4[3:0] =x
vote_s5[3:0] =x
vote_s6[3:0] =x
vote_s7[3:0] =x
vote_s8[3:0] =x
vote_s9[3:0] =x
100 sec
200 sec
300 sec
400 sec
0000100000
0100000000
0000100100
0000100000
0000001000
0000000010
1000000000
0000000001
3
3
0
2
0
4
0
0
3
2