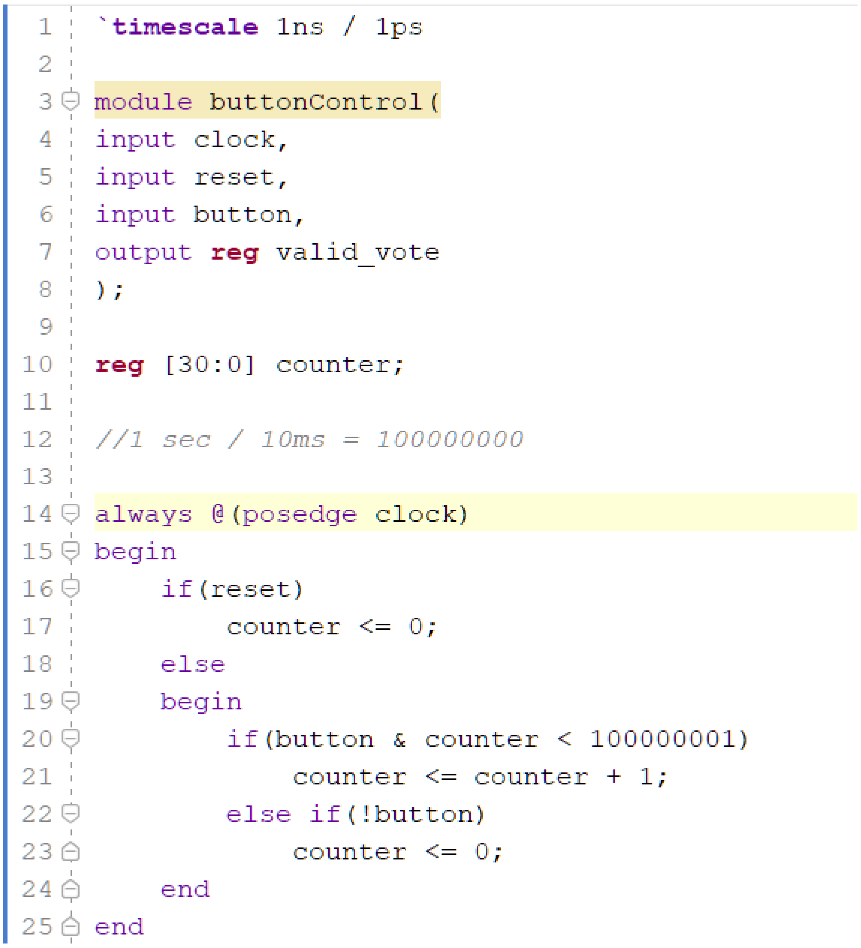
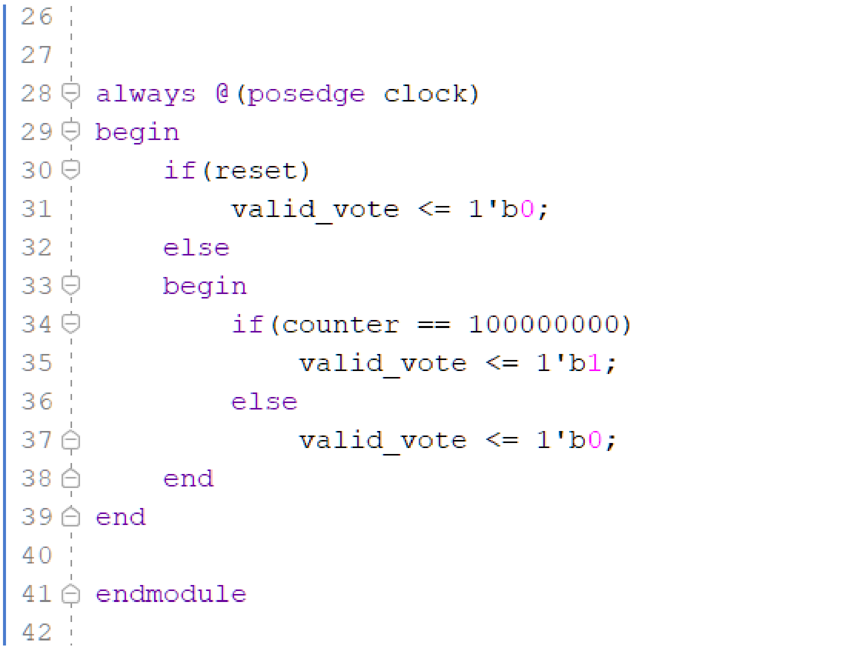
SIMPE VOTING MACHINE

BUTTON CONTROL

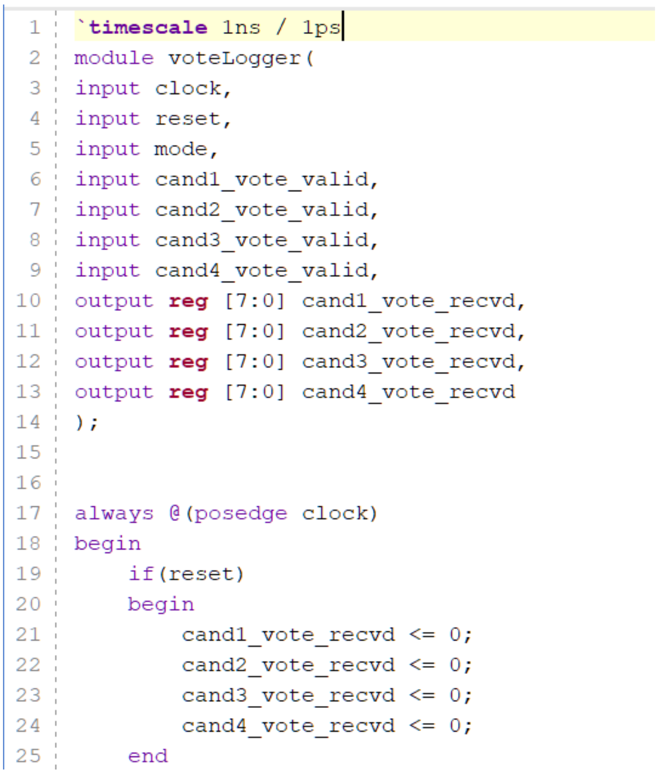


```
`timescale 1ns / 1ps

module buttonControl(
input clock,
input reset,
input button,
output reg valid_vote
);

reg [30:0] counter;

//1 sec / 10ms = 100000000

always @(posedge clock)
begin
    if(reset)
        counter <= 0;
    else
    begin
        if(button & counter < 100000001)
            counter <= counter + 1;
        else if(!button)
            counter <= 0;
    end
end
```

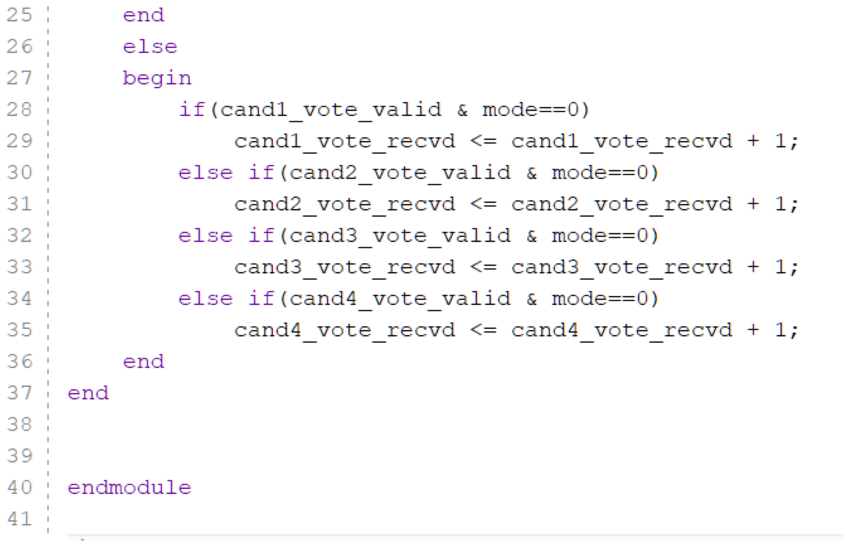


```


always @(posedge clock)
begin
    if(reset)
        valid_vote <= 1'b0;
    else
    begin
        if(counter == 100000000)
            valid_vote <= 1'b1;
        else
            valid_vote <= 1'b0;
    end
end

endmodule

```

VOTE LOGGER

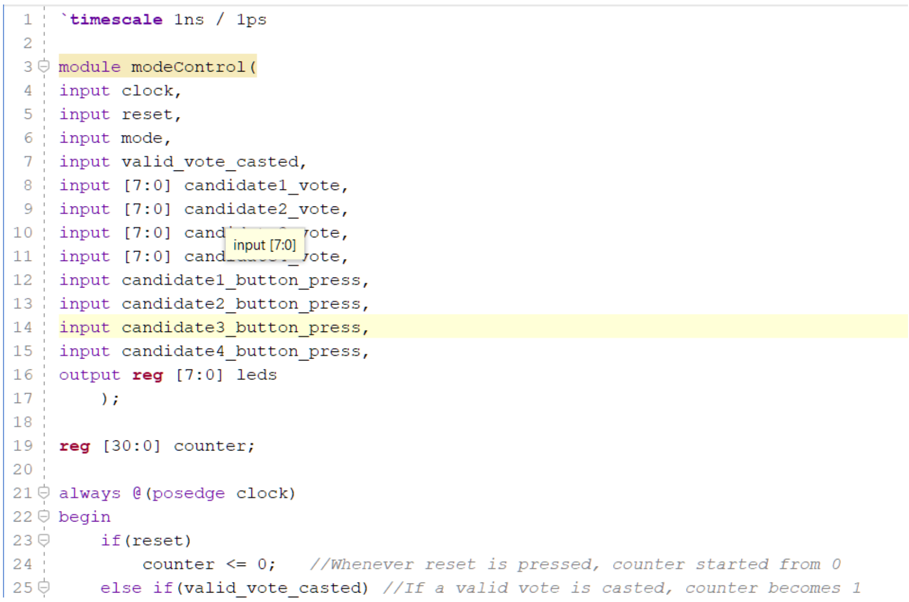


```
`timescale 1ns / 1ps
module voteLogger(
input clock,
input reset,
input mode,
input cand1_vote_valid,
input cand2_vote_valid,
input cand3_vote_valid,
input cand4_vote_valid,
output reg [7:0] cand1_vote_recvd,
output reg [7:0] cand2_vote_recvd,
output reg [7:0] cand3_vote_recvd,
output reg [7:0] cand4_vote_recvd
);


always @(posedge clock)
begin
    if(reset)
    begin
        cand1_vote_recvd <= 0;
        cand2_vote_recvd <= 0;
        cand3_vote_recvd <= 0;
        cand4_vote_recvd <= 0;
    end
```

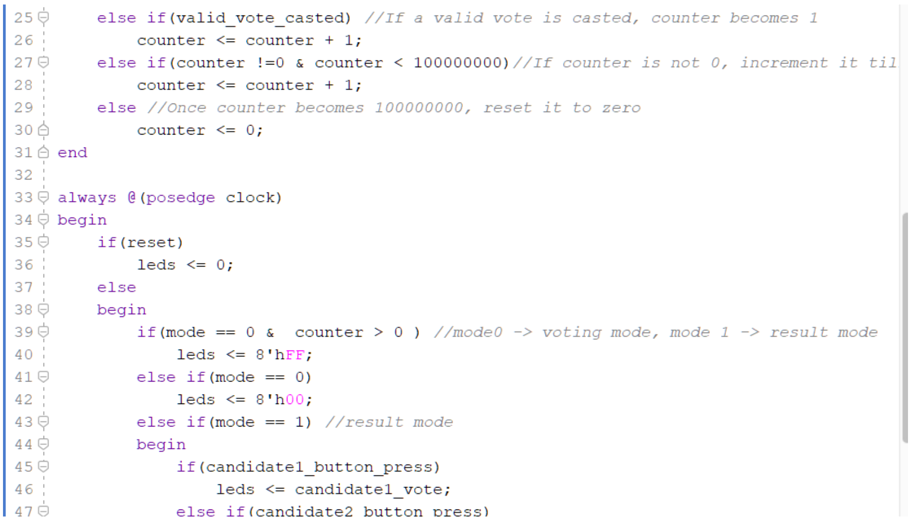


```
    end
    else
    begin
        if(cand1_vote_valid & mode==0)
            cand1_vote_recvd <= cand1_vote_recvd + 1;
        else if(cand2_vote_valid & mode==0)
            cand2_vote_recvd <= cand2_vote_recvd + 1;
        else if(cand3_vote_valid & mode==0)
            cand3_vote_recvd <= cand3_vote_recvd + 1;
        else if(cand4_vote_valid & mode==0)
            cand4_vote_recvd <= cand4_vote_recvd + 1;
    end
end


endmodule
```

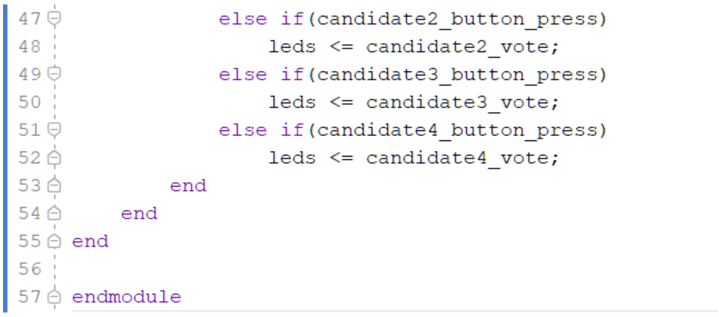
MODE CONTROL



```
`timescale 1ns / 1ps

module modeControl(
input clock,
input reset,
input mode,
input valid_vote_casted,
input [7:0] candidate1_vote,
input [7:0] candidate2_vote,
input [7:0] candidate3_vote,
input [7:0] candidate4_vote,
input candidate1_button_press,
input candidate2_button_press,
input candidate3_button_press,
input candidate4_button_press,
output reg [7:0] leds
    );

reg [30:0] counter;

always @(posedge clock)
begin
    if(reset)
        counter <= 0;   //Whenever reset is pressed, counter started from 0
    else if(valid_vote_casted) //If a valid vote is casted, counter becomes 1
```

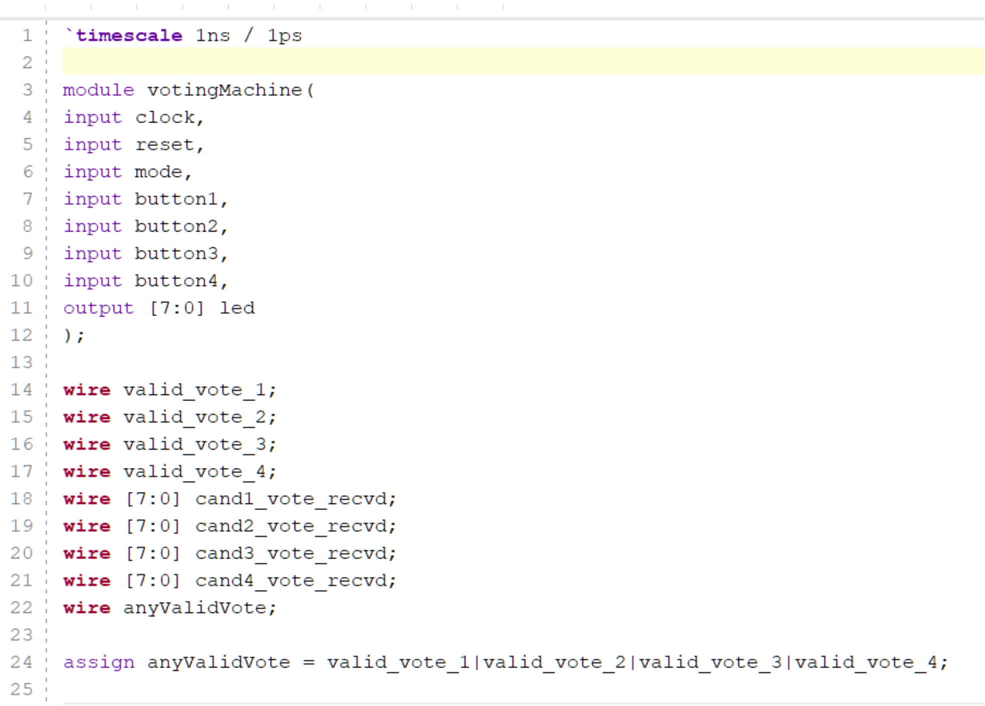


```
    else if(valid_vote_casted) //If a valid vote is casted, counter becomes 1
        counter <= counter + 1;
    else if(counter !=0 & counter < 100000000)//If counter is not 0, increment it til
        counter <= counter + 1;
    else //Once counter becomes 100000000, reset it to zero
        counter <= 0;
end

always @(posedge clock)
begin
    if(reset)
        leds <= 0;
    else
    begin
        if(mode == 0 &  counter > 0 ) //mode0 -> voting mode, mode 1 -> result mode
            leds <= 8'hFF;
        else if(mode == 0)
            leds <= 8'h00;
        else if(mode == 1) //result mode
        begin
            if(candidate1_button_press)
                leds <= candidate1_vote;
            else if(candidate2_button_press)
```

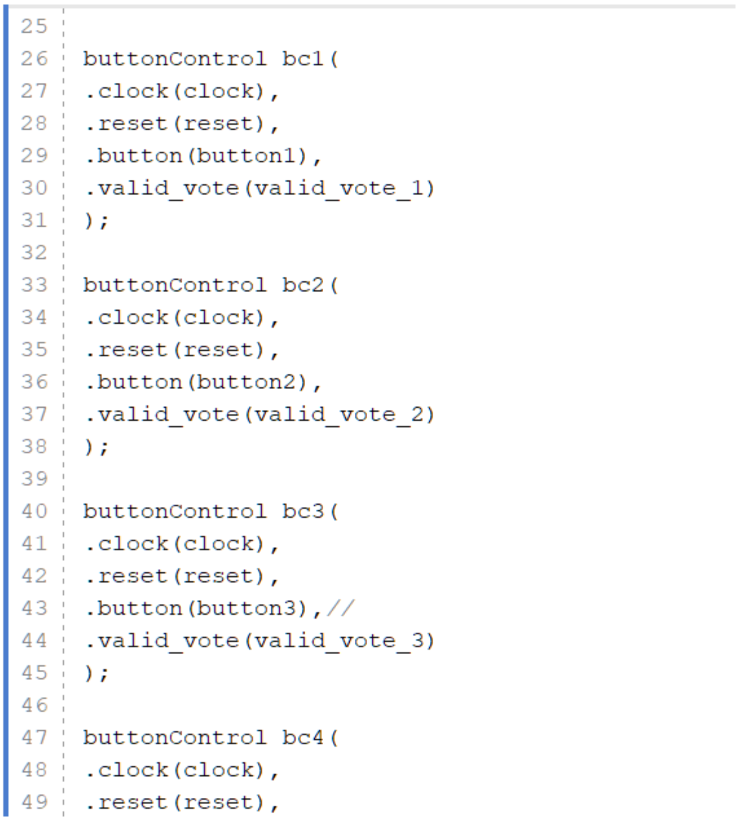


```
            else if(candidate2_button_press)
                leds <= candidate2_vote;
            else if(candidate3_button_press)
                leds <= candidate3_vote;
            else if(candidate4_button_press)
                leds <= candidate4_vote;
        end
    end
end

endmodule
```

TOP MODULE VOTING MACHINE

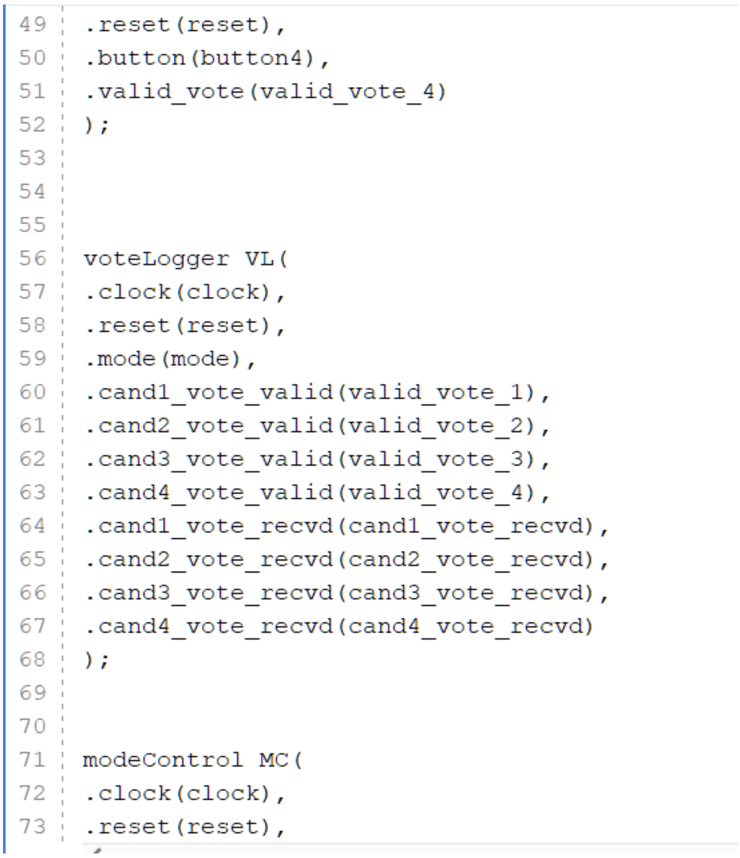


```verilog
`timescale 1ns / 1ps

module votingMachine(
input clock,
input reset,
input mode,
input button1,
input button2,
input button3,
input button4,
output [7:0] led
);

wire valid_vote_1;
wire valid_vote_2;
wire valid_vote_3;
wire valid_vote_4;
wire [7:0] cand1_vote_recvd;
wire [7:0] cand2_vote_recvd;
wire [7:0] cand3_vote_recvd;
wire [7:0] cand4_vote_recvd;
wire anyValidVote;

assign anyValidVote = valid_vote_1|valid_vote_2|valid_vote_3|valid_vote_4;

```

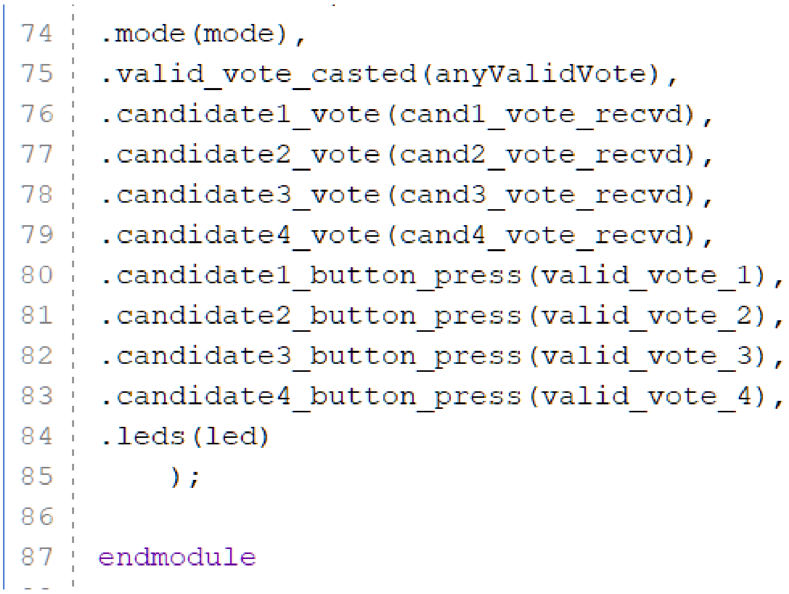


```verilog

buttonControl bc1(
.clock(clock),
.reset(reset),
.button(button1),
.valid_vote(valid_vote_1)
);

buttonControl bc2(
.clock(clock),
.reset(reset),
.button(button2),
.valid_vote(valid_vote_2)
);

buttonControl bc3(
.clock(clock),
.reset(reset),
.button(button3),//
.valid_vote(valid_vote_3)
);

buttonControl bc4(
.clock(clock),
.reset(reset),
```

TOP MODULE VOTING MACHINE

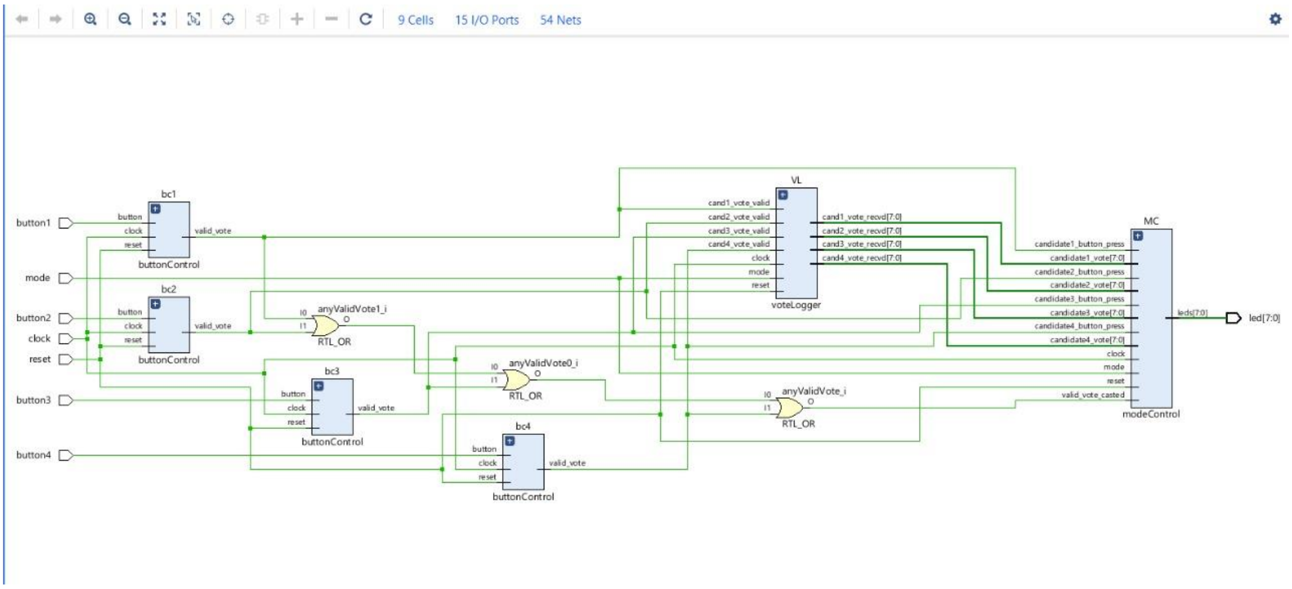


```
.reset(reset),
.button(button4),
.valid_vote(valid_vote_4)
);



voteLogger VL(
.clock(clock),
.reset(reset),
.mode(mode),
.cand1_vote_valid(valid_vote_1),
.cand2_vote_valid(valid_vote_2),
.cand3_vote_valid(valid_vote_3),
.cand4_vote_valid(valid_vote_4),
.cand1_vote_recvd(cand1_vote_recvd),
.cand2_vote_recvd(cand2_vote_recvd),
.cand3_vote_recvd(cand3_vote_recvd),
.cand4_vote_recvd(cand4_vote_recvd)
);


modeControl MC(
.clock(clock),
.reset(reset),
```

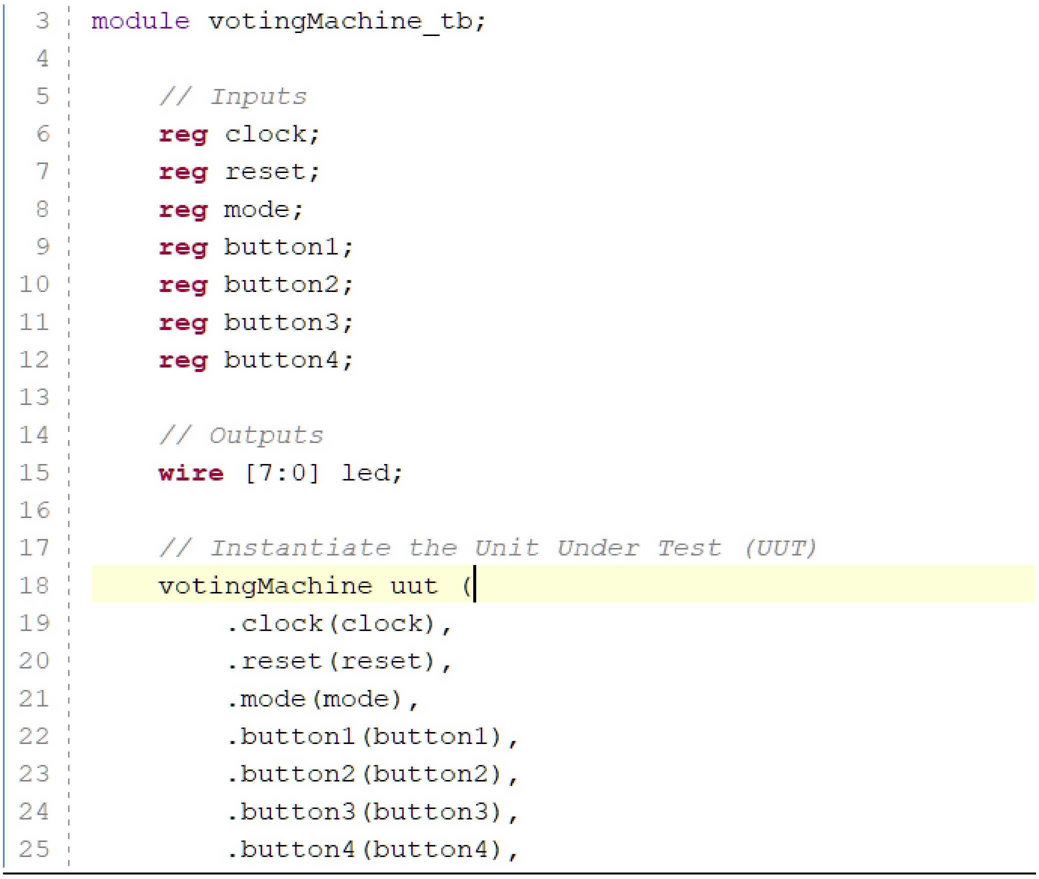


```
.mode(mode),
.valid_vote_casted(anyValidVote),
.candidate1_vote(cand1_vote_recvd),
.candidate2_vote(cand2_vote_recvd),
.candidate3_vote(cand3_vote_recvd),
.candidate4_vote(cand4_vote_recvd),
.candidate1_button_press(valid_vote_1),
.candidate2_button_press(valid_vote_2),
.candidate3_button_press(valid_vote_3),
.candidate4_button_press(valid_vote_4),
.leds(led)
    );

endmodule
```

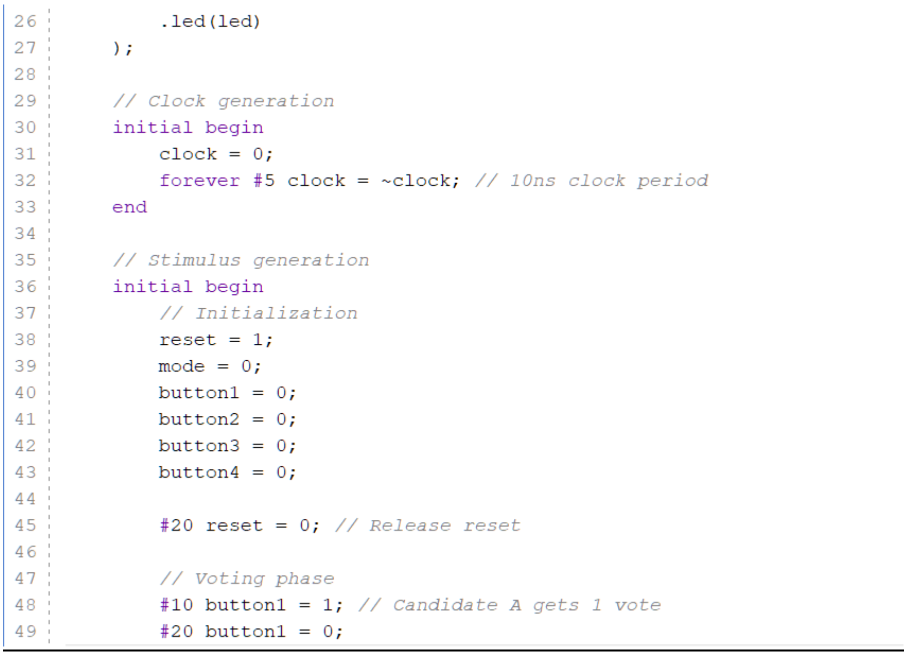
- **RTL SCHEMATIC**



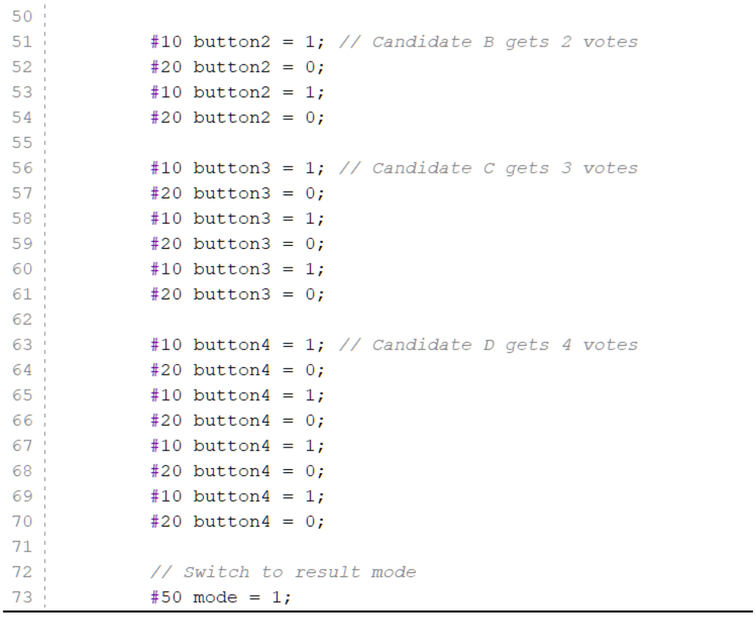
- **TESTBENCH**



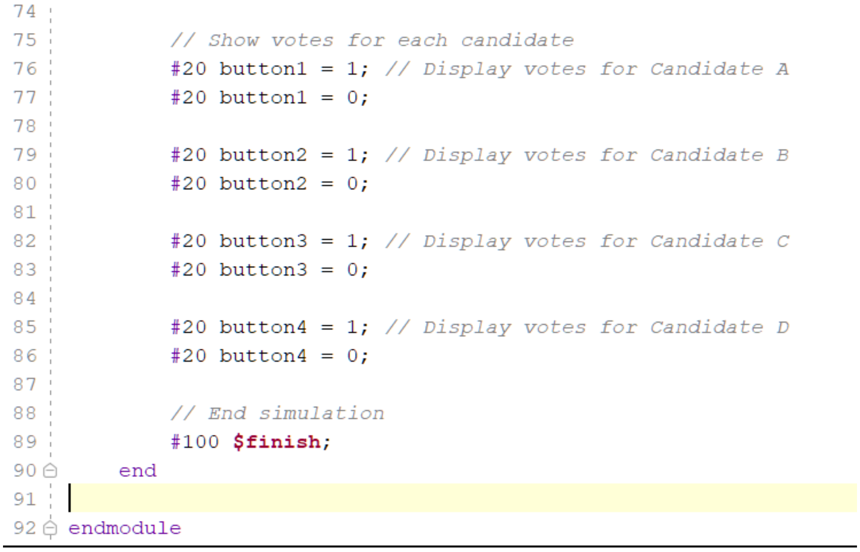
```verilog
module votingMachine_tb;

    // Inputs
    reg clock;
    reg reset;
    reg mode;
    reg button1;
    reg button2;
    reg button3;
    reg button4;

    // Outputs
    wire [7:0] led;

    // Instantiate the Unit Under Test (UUT)
    votingMachine uut (
        .clock(clock),
        .reset(reset),
        .mode(mode),
        .button1(button1),
        .button2(button2),
        .button3(button3),
        .button4(button4),
```



```verilog
        .led(led)
    );

    // Clock generation
    initial begin
        clock = 0;
        forever #5 clock = ~clock; // 10ns clock period
    end

    // Stimulus generation
    initial begin
        // Initialization
        reset = 1;
        mode = 0;
        button1 = 0;
        button2 = 0;
        button3 = 0;
        button4 = 0;

        #20 reset = 0; // Release reset

        // Voting phase
        #10 button1 = 1; // Candidate A gets 1 vote
        #20 button1 = 0;
```

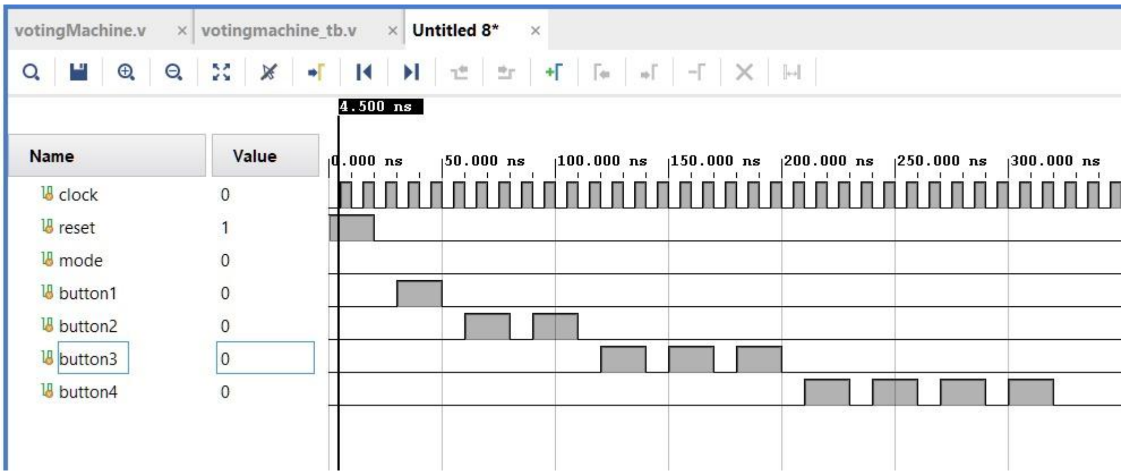


```verilog

        #10 button2 = 1; // Candidate B gets 2 votes
        #20 button2 = 0;
        #10 button2 = 1;
        #20 button2 = 0;

        #10 button3 = 1; // Candidate C gets 3 votes
        #20 button3 = 0;
        #10 button3 = 1;
        #20 button3 = 0;
        #10 button3 = 1;
        #20 button3 = 0;

        #10 button4 = 1; // Candidate D gets 4 votes
        #20 button4 = 0;
        #10 button4 = 1;
        #20 button4 = 0;
        #10 button4 = 1;
        #20 button4 = 0;
        #10 button4 = 1;
        #20 button4 = 0;

        // Switch to result mode
        #50 mode = 1;
```

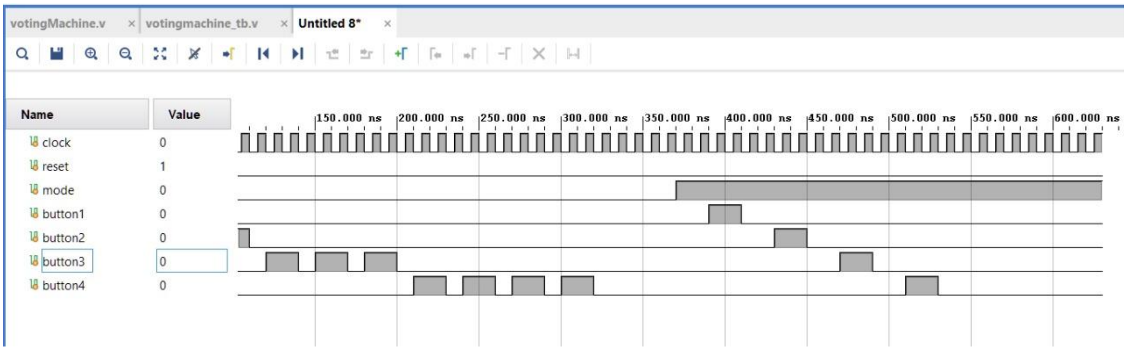


```verilog

        // Show votes for each candidate
        #20 button1 = 1; // Display votes for Candidate A
        #20 button1 = 0;

        #20 button2 = 1; // Display votes for Candidate B
        #20 button2 = 0;

        #20 button3 = 1; // Display votes for Candidate C
        #20 button3 = 0;

        #20 button4 = 1; // Display votes for Candidate D
        #20 button4 = 0;

        // End simulation
        #100 $finish;
    end

endmodule
```

- **SIMULATION RESULTS**



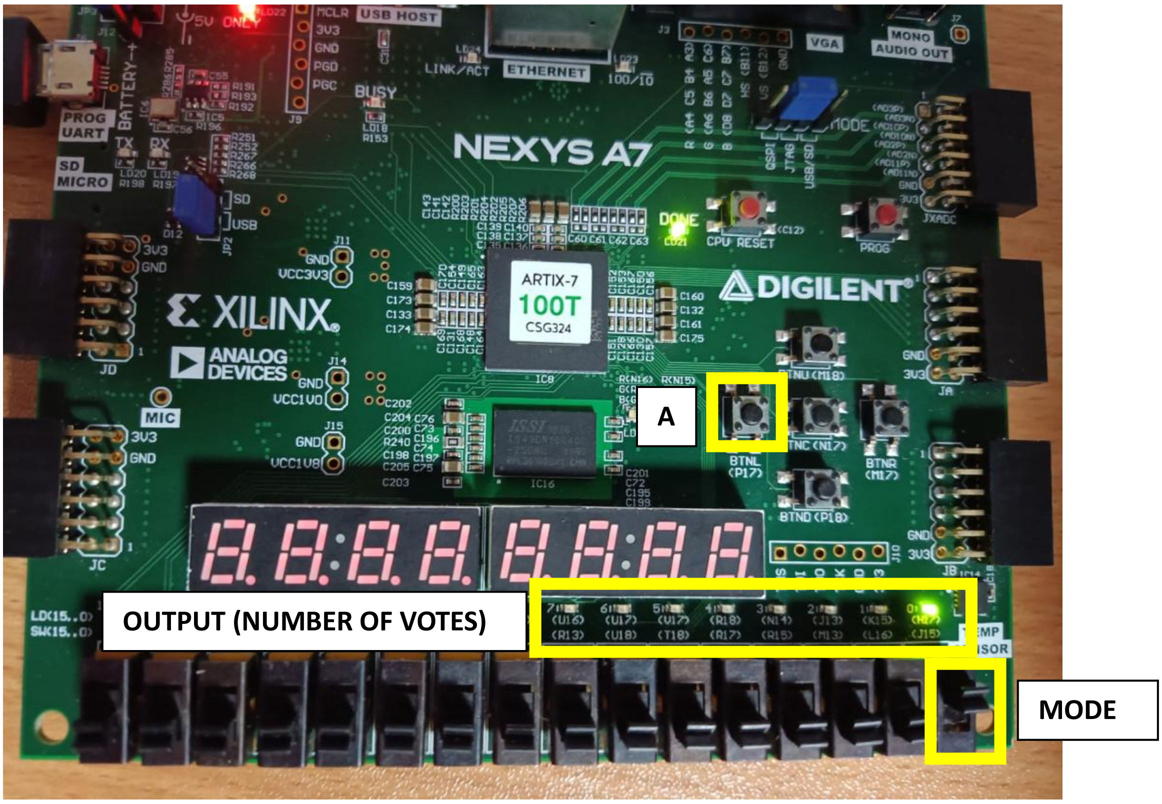
Name	Value
clock	0
reset	1
mode	0
button1	0
button2	0
button3	0
button4	0



Name	Value
clock	0
reset	1
mode	0
button1	0
button2	0
button3	0
button4	0

- **HARDWARE RESULTS**

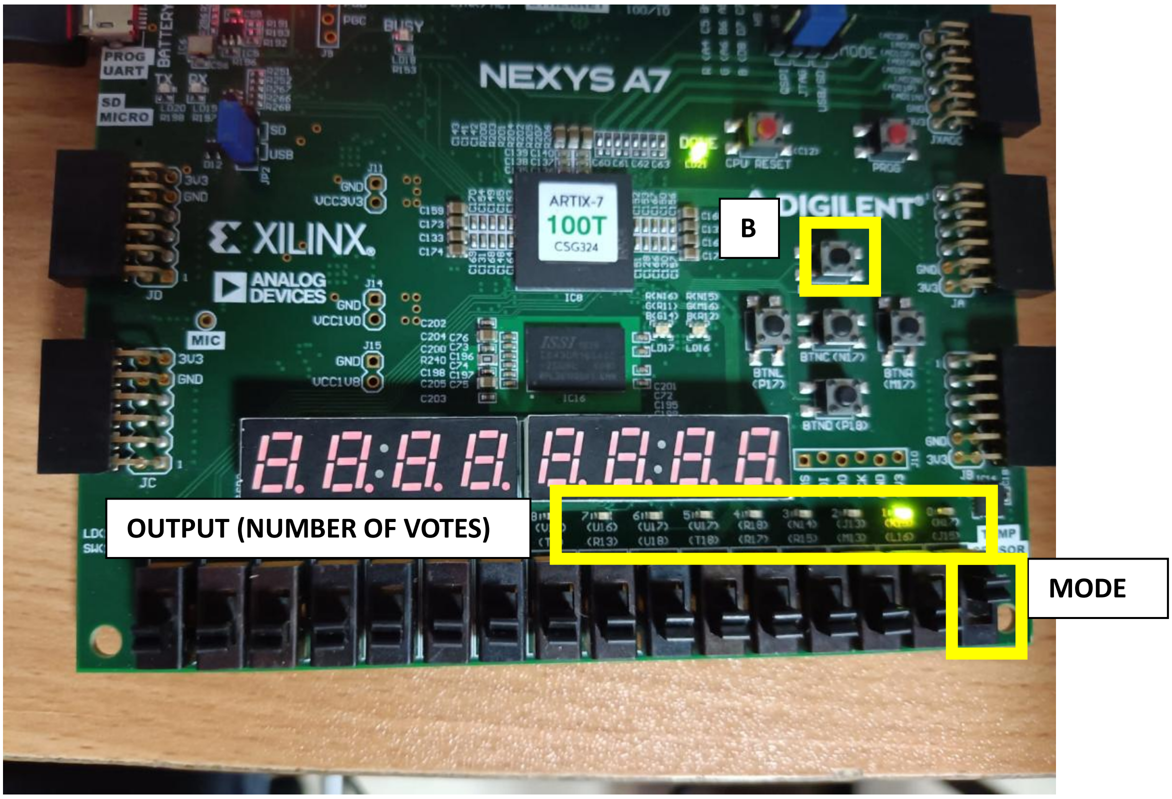
CANDIDATE	NO. OF VOTES (INPUT)
A	1
B	2
C	3
D	4



A

OUTPUT (NUMBER OF VOTES)

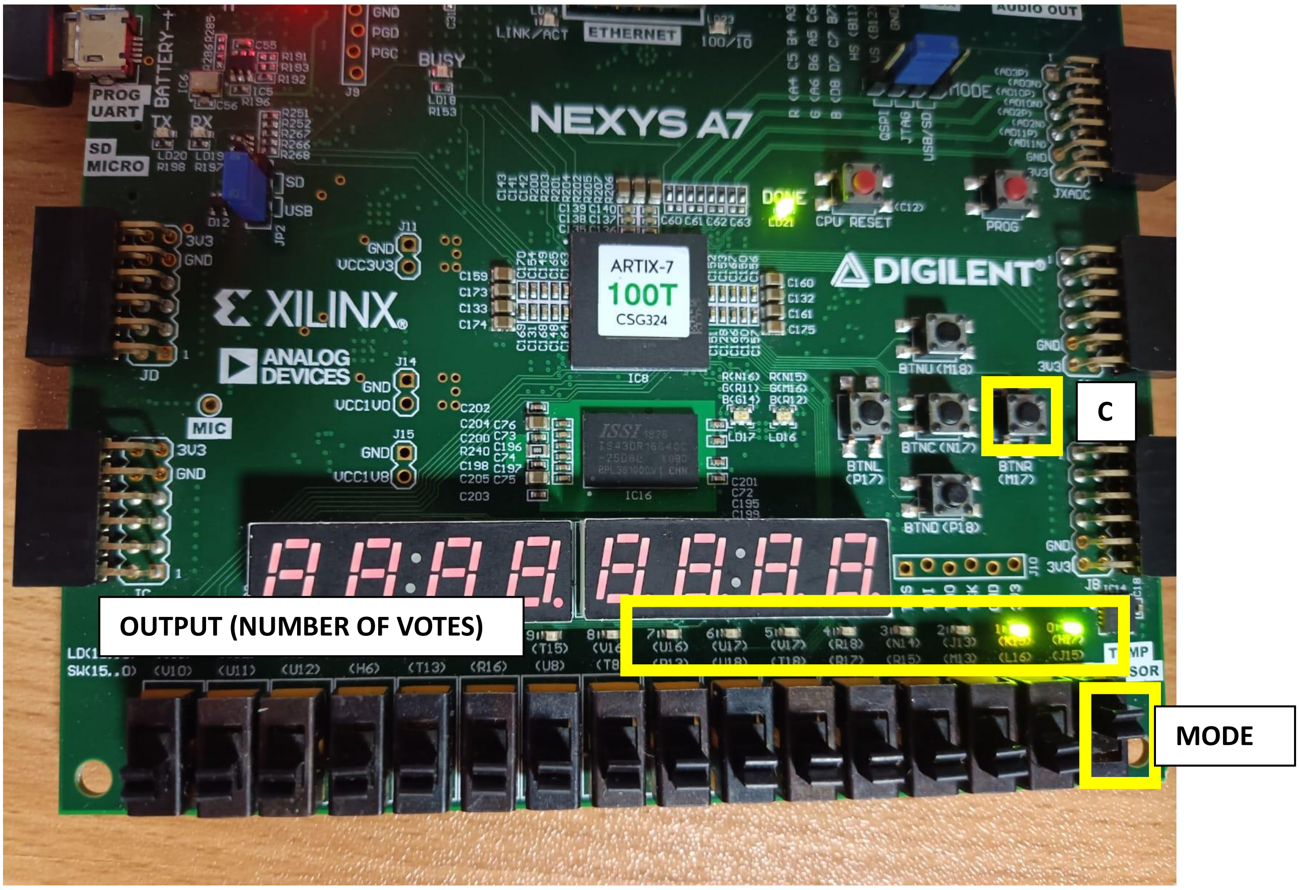
MODE

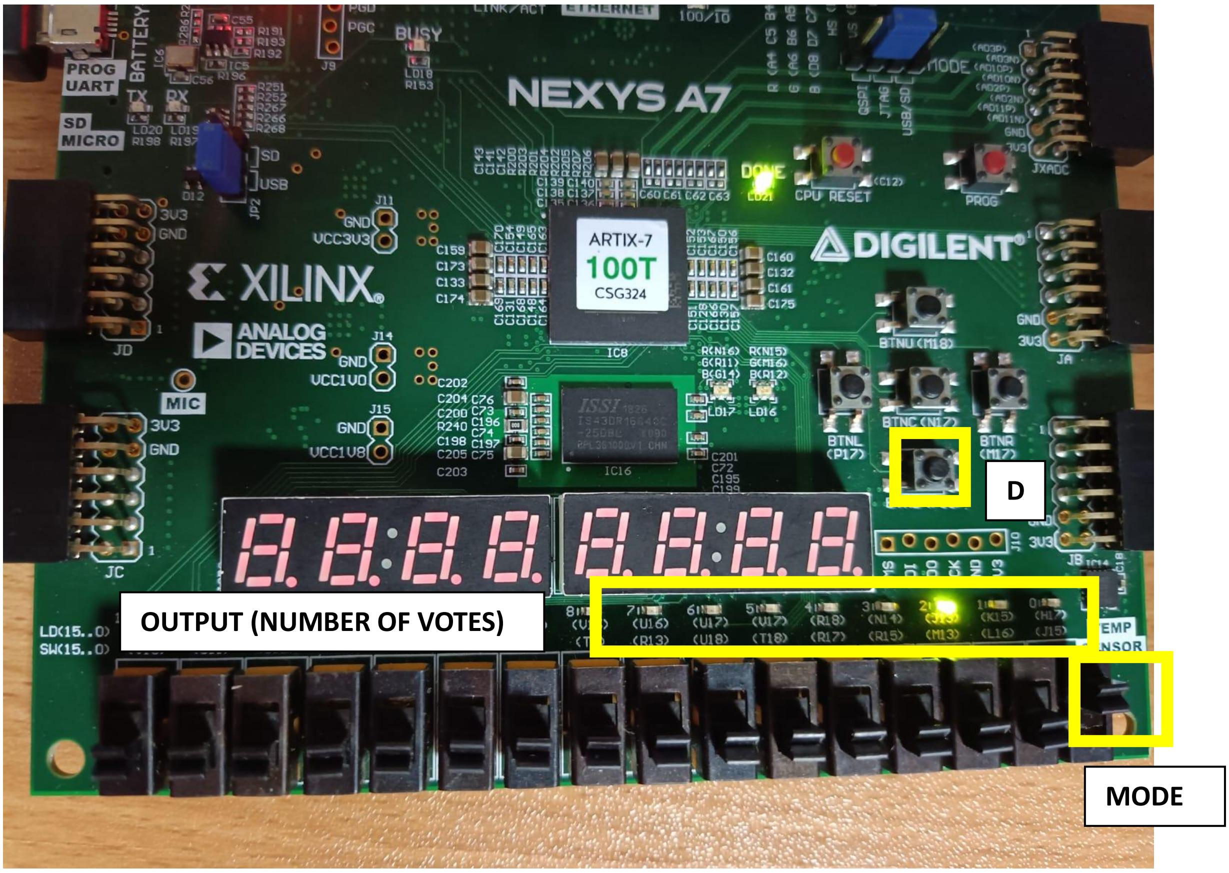


B

OUTPUT (NUMBER OF VOTES)

MODE


C
OUTPUT (NUMBER OF VOTES)
MODE


D
OUTPUT (NUMBER OF VOTES)
MODE