A PROJECT REPORT ON

FPGA BASED ELECTRONIC VOTING MACHINE

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA, SURATHKAL

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted by

Todeti Deekshitha (221EC162)

Medhinee Padorthy (221EC130)

Under the guidance of

Dr. Kalpana G Bhat

11 March 2024

Abstract

This project focuses on implementing an electronic voting machine (EVM) on a Field-Programmable Gate Array (FPGA) platform. The objective is to create a secure and efficient system for electronic voting, ensuring integrity and confidentiality. The FPGA serves as the hardware platform, enabling real-time processing and robust control. The design incorporates cryptographic techniques to safeguard voting data, and the system aims to provide a user-friendly interface while addressing key challenges in electronic voting systems. The project emphasizes the importance of reliability, security, and accessibility in modernizing the electoral process through FPGA-based EVM implementation.

Table of contents:

Tables and figures

Introduction:

Electronic voting machines (EVMs) have revolutionized the electoral process by providing a secure, accurate, and efficient means of voting. Unlike traditional paper ballots, EVMs are electronic devices that allow voters to cast their votes by pressing a button or touching a screen. These machines are equipped with advanced technology to ensure the integrity and confidentiality of the voting process. EVMs have been widely adopted around the world due to their ability to minimize errors, reduce the time required to count votes, and enhance the overall transparency of elections.

Designing an Electronic Voting Machine (EVM) on a Field-Programmable Gate Array (FPGA) offers a flexible and efficient solution for implementing secure and reliable voting systems. FPGA-based EVMs can provide real-time processing, high-speed data transfer, and robust security features. This presentation will explore the key components and advantages of an FPGA-based EVM design, highlighting its potential for enhancing the electoral process.

Motivation

1.Secure voting system:

FPGA-based EVMs can provide secure and tamper-resistant voting systems, ensuring the integrity of the voting process.

2.Real time processing:

FPGAs allow for real-time processing of votes, enabling quick and efficient tabultion bulation of results.

3.High speed data transfer:

FPGAs support high-speed data transfer, which is essential for transmitting voting data securely and quickly.

4.Auditability:

FPGAs can be programmed to provide auditability features, allowing for the verification of votes and ensuring transparency in
the voting process.

5.Scalability:

FPGA-based EVMs can be easily scaled to accommodate different voting needs, from small local elections to large national elections.

Literature Review:

[1] describes more about power efficiency. This evm works for 3 parties. It has a check input to select party and then "select" to vote. It has count for all 3 parties as output.

It lacks authentication input. It does not show voter wether vote is casted or not.It doesn't have a seperate access to voter and officer, even voter can see election results.

Evm works for 3 parties in [2]. It has voter_enable, given by poling officer which allows voters to cast their vote. It has 3 seven segment display outputs with active low enables to show numbers of votes casted for each party. It also has Dout output to check total number of votes casted. It has opled output corresponding to each party to show voters wether they casted vote to intended party.

It accepts more than 1 vote at a time. There is no need to use 4 different displays, when we have registers already to store values, we can select them using another input.

The Voting Machine is implemented on the FPGA NEXYS-4 board in [3]. It works for 4 candidates. It has mode input which enables to vote when 0 and shows count when 1. Vote is considered valid only if button is pressed for 1 second.

There is no authentication input in this model. It accepts more than 1 vote at a time.

Objectives

1. Ballot Casting

Design the FPGA logic to handle the process of voting securely .

2. Accuracy and Reliability

Ensure accurate and reliable vote counting.

3. Security

Implement robust security measures to prevent tampering and ensure the integrity of voting system.

4. Transperency

Provide a transparent and auditabpe system they allows voters and election authorities to verify the correctness of voting process.

5. Cost effective Design

Use FPGAs to create cost effective and scalable solution for electronic voting

6. Speed and Efficiency

Speed up voting process and enable quick and efficient tabultion for easy elections.

Methodology

Xilinx vivado tool is used for verilog coding. We have used Nexys4 FPGA.

This voting machine works for 255 voters and 4 candidates. Inputs are push buttons for each candidate and authentication input. We also have a "mode" input which enables voting when reset and enables to view vote count when it is set. Outputs are 4 leds to show voters which button they have pressed, one led to show whether their vote is casted, and 8 bit vote count output which shows the number of votes of candidate selected.

Our main module is voting_mc. It takes inputs and have instantiations of validation module which checks whether vote is valid or not based on time during the button is pressed and authentication. Then if vote for a candidate is valid and valid votes for all other candidates is zero , count of that particular candidate is increased by 1. Then it gives person_voted output so that poling officer makes authentication low(or some other model to make authentication low) , so that voter can't vote twice.All this happens when mode is zero.
When mode is 1, number of votes of that particular candidate are shown, if user is authenticated and pressed button for 1 sec.

Code

```
module voting_mc(
input clk,
input reset,
input mode,
input button_cand1,
input button_cand2,
input button_cand3,
input button_cand4,
input authentication,
output  led1,
output  led2,
output  led3,
output  led4,
output reg [7:0] count,
output wire person_voted );

wire valid1;
wire valid2;
wire valid3;
wire valid4;
```

```
reg [7:0]count_1;
reg [7:0]count_2;
reg [7:0]count_3;
reg [7:0]count_4;
reg [3:0]abcd;


reg pv;
assign person_voted = authentication && pv;

validation p(clk, reset, button_cand1,authentication, valid1,  led1);
validation q(clk, reset, button_cand2,authentication, valid2,  led2);
validation r(clk, reset, button_cand3,authentication, valid3,, led3);
validation s(clk, reset, button_cand4,authentication, valid4, led4);

always @(posedge clk)
begin
   if(reset)
   begin
      count_1<=0;
      count_2<=0;
      count_3<=0;
      count_4<=0;
      pv <= 1'b0;
   end
```

```
 else
      begin

      if(mode == 1'b0)
         begin
           if((valid1) && (!valid2) && (!valid3) && (!valid4))
             begin
              count_1 <= count_1 + 1;
               pv <= 1'b1;
             end
                 else if((!valid1) && (valid2) && (!valid3) &&
(!valid4))
             begin
              count_2 <= count_2 + 1;
                pv <= 1'b1;
             end
                 else if((!valid1) && (!valid2) && (valid3) &&
(!valid4) )
             begin
                count_3 <= count_3 + 1;
                 pv <= 1'b1;
             end
                 else if((!valid1) && (!valid2) && (!valid3) &&
(valid4))
             begin
                count_4 <= count_4 + 1;
                  pv <= 1'b1;
             end
           else if (authentication == 0)
                  pv <= 1'b0;
```

```
            end
     end
end
always @(posedge clk)
begin
   if(mode == 1'b1)
      begin
         if(valid1 || valid2 || valid3 || valid4)
             abcd <= {valid1, valid2, valid3, valid4};

      end
    else
         abcd <= 4'b0000;
end

always @(posedge clk)
begin
      case(abcd)
      4'b1000:    count <= count_1;
      4'b0100:    count <= count_2;
      4'b0010:    count <= count_3;
      4'b0001:    count <= count_4;
      default:    count <= 8'b0;
      endcase

end

endmodule
```

```
module validation(
input clk,
input reset,
input button,
input  auth,
output reg valid_vote,
output reg led
   );
reg [29:0] counter_a,
always @(posedge clk)
begin

   if(reset)
        counter_a <= 0;

   else

      begin
         if(button && auth)
            counter_a <= counter_a + 1;
         else if(counter_a > 0 && counter_a <= 200000000)
             counter_a <= counter_a + 1;
         else
            counter_a <= 0;

      end

end
```

```
always @(posedge clk)
begin
   if(reset)
      valid_vote <= 1'b0;
   else
   begin
      if(counter_a >= 100000000 )
         begin
           led <= 1'b1;
            if(counter_a == 100000000 )
             valid_vote <= 1'b1;
            else
               valid_vote <= 1'b0;
          end

      else
         begin
            valid_vote <= 1'b0;
            led <= 1'b0;
          end
   end
end
endmodule
```

Results and discussion

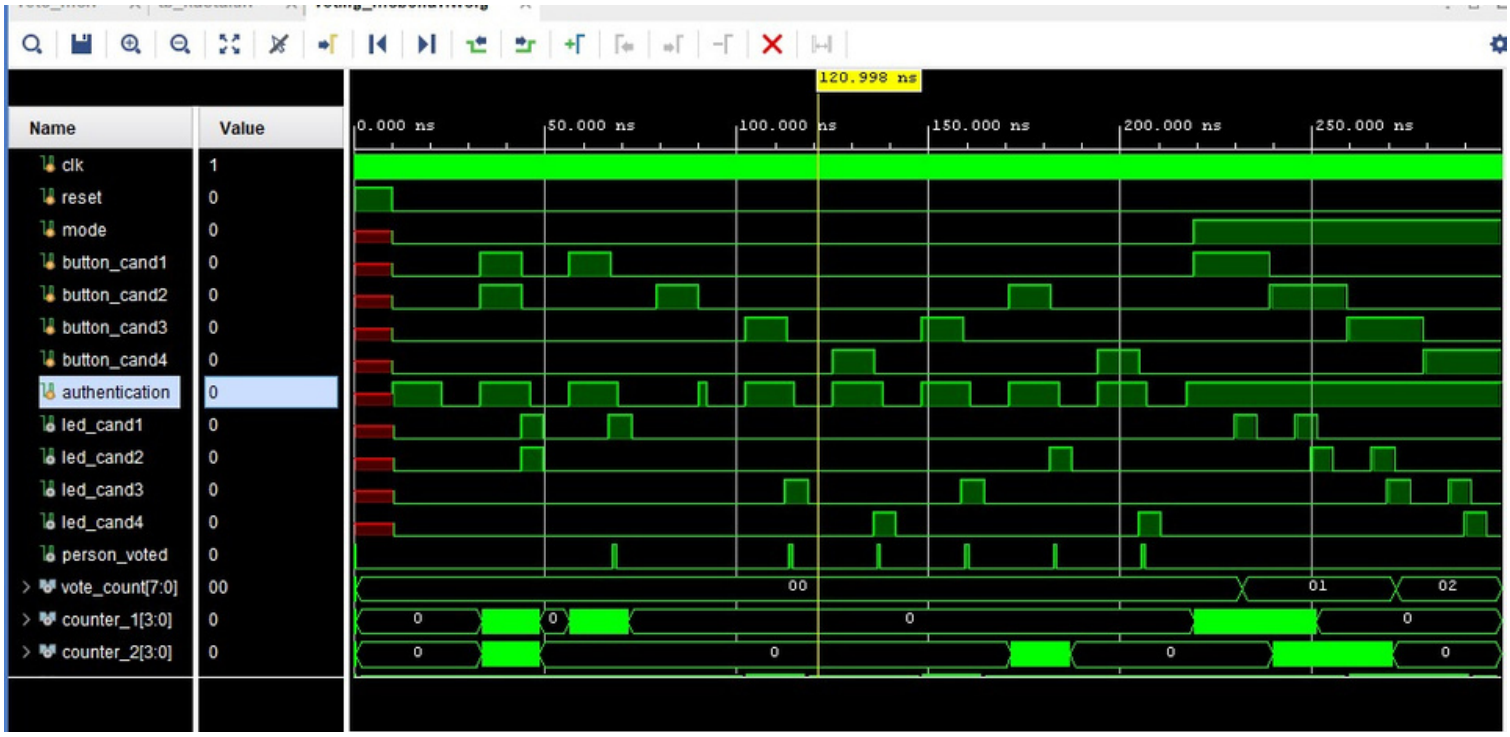

fig1

Leds of corresponding candidates are glown when button is pressed for 10 ns and authentication is high.
Vote is counted, and" person_voted" is high when only one valid vote is recorded.
After mode is given high input, count is stopped and vote_count is displayed according to the candidate selected.

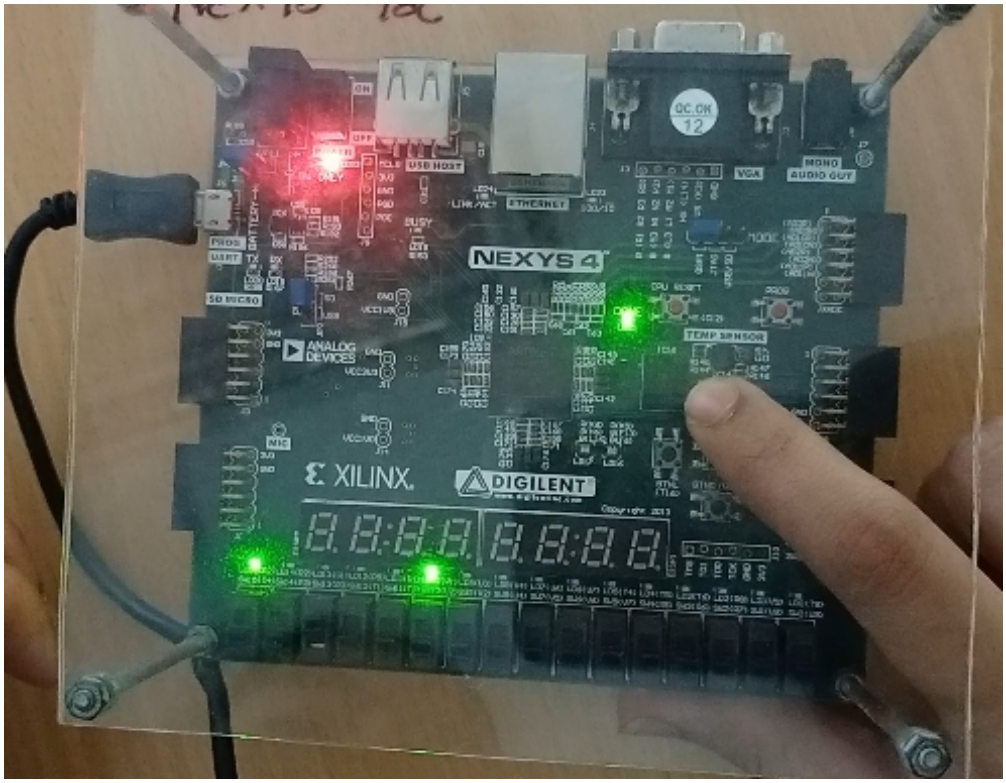

fig2

Four push buttons are for each candidate button inputs. First slide switch is for reset, second for mode and third for authentication. Left most leds represent led1, led2, led3, led4 corresponding to each candidate indicating that they have pressed the button for more than 1 sec. The middle led glows if the vote is taken for count, that is if the vote is a valid one.

In fig2, reset is 0 and mode is 0 and authentication is1. So, when button corresponding to first person is pressed for more than a second, and not any other button is pressed, led1 and person_voted led glows.

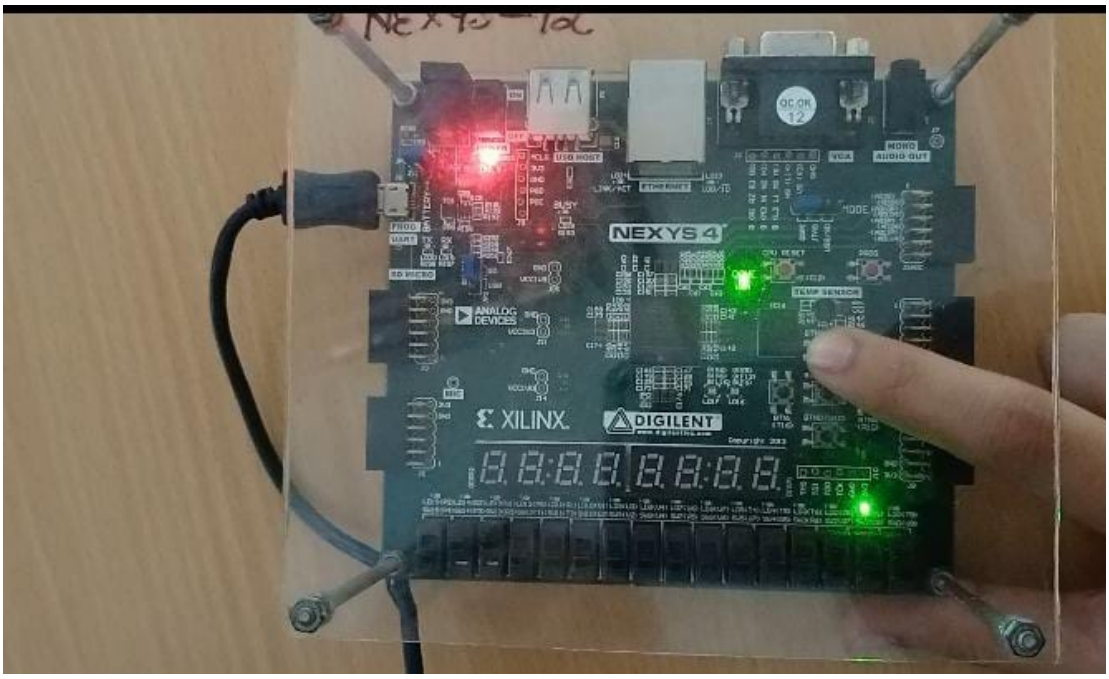

fig3

In fig3, mode is 1 and authentication is also 1. So when button corresponding to first candidate is pressed it is showing number of votes casted to that particular candidate using the 8 right most leds.

Conclusion

The voting_mc module is a versatile Verilog implementation of a voting system capable of counting votes for four candidates, validating votes, and managing vote counts based on different modes of operation. Its functionality includes vote counting, mode selection, vote validation, LED indication, vote confirmation, and vote count selection. The module provides a robust framework for implementing various types of voting systems in hardware, making it suitable for a wide range of applications.

Future scope

Accessibility Features: Consider adding features to make the voting machine more accessible to a wider range of voters, such as support for different languages or interfaces for voters with disabilities.

Data Analytics: Explore the use of data analytics to analyze voting patterns and trends, which could provide valuable insights for election organizers.

Voter Verification: Implement additional methods for verifying voters' identities, such as biometric authentication or two-factor authentication.

References

[1]. Prashant Kudesia,Manika,Aviral Kulshrestha,Anjan Kumar,"Nano Ballot Box: A Low Power, Thermal Resistant FPGA Based Electronic Voting Machine",2021 International Conference on Simulation, Automation & Smart Manufacturing (SASM) GLA University, Mathura, India. August 20-21, 2021

[2]. K. Gurucharan, B. Kiranmai, S. S. Kiran, M. Ravindra Kumar,"Xilinx Based Electronic Voting Machine,International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-9 Issue-1, October, 2019

[3]. Mritunjay Kumar,Gurpreet Singh Dhami,Ravi Shankar,"FPGA Based Voting Machine"May 9, 2023

Approval from guide

Medhinee Padorthy 11:35 AM

ToCourse instructor:Dr Kalpana G Bhat
National institute of technology karnataka

Kalpana B... 12:13 PM

to me

From Kalpana Bhat • kalpanabhat_ec@nitk.edu.in

To Medhinee Padorthy • medhineesrinivas@gmail.com

Date Apr 14, 2024, 12:13 PM

Standard encryption (TLS).

View security details

Approved.
Kalpana G Bhat