BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

EEE 304 (January 2023)
Digital Electronics Laboratory

Final Project Report

Section: A2 Group: 05

ELECTRONIC VOTING MACHINE

Course Instructors:
Nafis Sadik, Lecturer
Sadat Tahmeed Azad, Adjunct Lecturer

Signature of Instructor: _____

Academic Honesty Statement:

IMPORTANT! Please carefully read and sign the Academic Honesty Statement, below. Type the student ID and name, and put your signature. *You will not receive credit for this project*

"In signing this statement, We hereby certify that the work on this project is our own and that we have not copied the work of any other students (past or present), and cited all relevant sources while completing this project. We understand that if we fail to honor this agreement, We will each receive a score of ZERO for this project and be subject to failure of this course."

Signature: _____ **Full Name: Subhan Zawad Bihan** **Student ID: 1906037**	**Signature:** _____ **Full Name: K. M. Azmain Rafin** **Student ID: 1906045**
Signature: _____ **Full Name: Tapu Datta** **Student ID: 1906049**	**Signature:** _____ **Full Name: Mohammad Nafis Kamal** **Student ID: 1906061**

experiment unless this statement is signed in the presence of your lab instructor.

Table of Contents

1 Abstract

In this project, we have constructed a simplified Electronic Voting Machine (EVM) using FPGA technology. Initially, the system verifies a voter's identity by checking their serial number. Once it confirms that no vote has been registered under the same serial number, an authentication signal is generated. Subsequently, the voter is allowed to cast their vote for their preferred candidate. These votes are then tallied, along with the total number of authenticated votes. If any of these steps are not completed, the vote is considered invalid and will not be counted. This module offers two operational modes: voting, where votes are cast and stored, and display, which shows the total vote count. Thanks to the implementation of a two-step verification process, this EVM model ensures robust security against any attempts at tampering with the votes.

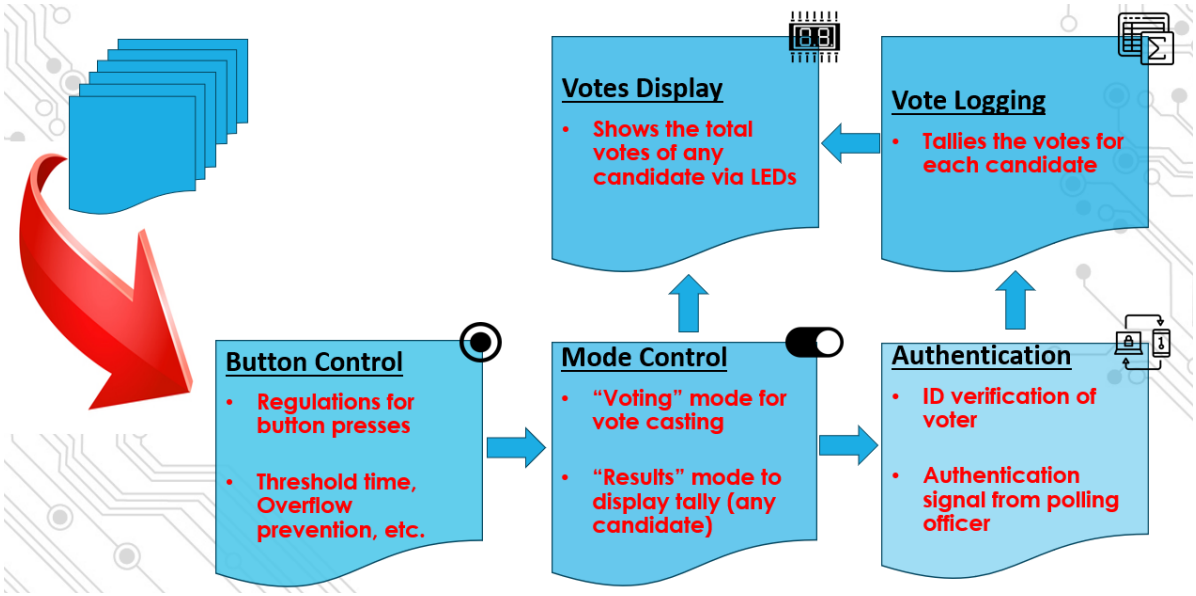
2 Introduction

Voting forms the foundation of a democratic system, and throughout history, various approaches have been employed to ensure the integrity of the voting process, guaranteeing accurate results that are not susceptible to manipulation. While government elections are widely known, many everyday procedures also involve some form of voting. Traditional paper ballots have been a common method, but they lack robust security against result tampering and introduce challenges like storage, manual counting, extensive preparation, and post-election waste. Electronic voting machines (EVMs) offer a solution to these issues, but it's essential to implement safeguards to make them tamper resistant. Notably, EVMs used in parliamentary elections are often prohibitively expensive. Our model presents an affordable and simplified design suitable for use by the public.

2.1 Complexity Analysis

The quest to establish a robust and efficient voting system is an ongoing endeavor. The credibility of any election hinges on the effectiveness of the voting system. Many electronic voting machines (EVMs) rely on battery power and incorporate one-time programmable chips to safeguard against external interference. These security measures come at a high cost, making it almost impractical to deploy EVMs for routine manual voting processes. Simplified voting machines with robust security often leverage digital electronic components like logic gates, flip-flops, FPGA modules, or microcontrollers. Excluding microcontrollers from our design enhances the clarity of how these devices operate. Certain aspects of this project, such as validity checks and counters, have potential applications beyond the scope of voting systems.

3 Design



3.1 Identification of Scope

This model can be used for elections in small scales, for example, in a school, or in a community. Also, it can be scaled up, and some modifications can be added. Thus, this model can be executed in larger scale.

3.2 Design Method

We have constructed a simplified Electronic Voting Machine (EVM) on an FPGA board using Verilog HDL. During this development, we incorporated both synchronous and asynchronous modules. One of the challenges addressed was the debouncing issue commonly encountered on FPGA boards, which was resolved through the creation of a button control module. This module ensures that button presses are correctly registered. Additionally, a pre-authentication module was devised to verify a voter's serial number and enable an authentication signal only if no prior vote has been recorded under that serial number. Subsequently, a polling officer must provide the authentication signal, without which a voter cannot cast their ballot. Once these two verification steps are successfully completed, the voter is permitted to vote for their chosen candidate. The system records and stores valid votes for each candidate. This module offers two modes: an open mode for voting and another mode for displaying the results.

3.3 Full Source Code

```
`timescale 1ns / 1ps

module authentication(
  input clock,
  input reset,
  input anyValidVote,
  input [7:0] voter_id,
  output reg [255:0] voted
);

// might have the valid_votes and voted as inout

  always @(posedge clock)
  begin

                if (reset) voted = 0;
                else if (anyValidVote) voted[voter_id] = 1;

  end

endmodule
```

```
`timescale 1ns / 1ps

module voteLogger(
input clock,
input reset,
input mode,
input cand1_vote_valid,
input cand2_vote_valid,
input cand3_vote_valid,
output reg [6:0] cand1_vote_recvd,
output reg [6:0] cand2_vote_recvd,
output reg [6:0] cand3_vote_recvd
);

always @(posedge clock)
begin
   if(reset)
   begin
      cand1_vote_recvd <= 0;
      cand2_vote_recvd <= 0;
      cand3_vote_recvd <= 0;
   end
   else if(~mode)
   begin
      if(cand1_vote_valid)
         cand1_vote_recvd = cand1_vote_recvd + 1;
      else if(cand2_vote_valid)
         cand2_vote_recvd = cand2_vote_recvd + 1;
      else if(cand3_vote_valid)
         cand3_vote_recvd = cand3_vote_recvd + 1;
   end
end

endmodule
```

Table: Source Code for Authentication and Vote Logger

```
`timescale 1ns / 1ps

module buttonControl(
input clock,
input reset,
input button,
input mode,
input pollsig,
input [7:0] voter_id,
input [255:0] voted,
output reg valid_vote
);

reg [24:0] counter;

always @(posedge clock)
begin
                if(reset | mode | ~pollsig)
      counter = 0;
   else
   begin
      if(button & counter < 25000001)
         counter = counter + 1;
      else if(!button)
         counter = 0;
   end
end


always @(posedge clock)
begin
   if(reset)
      valid_vote = 1'b0;
   else
   begin
      if(counter == 25000000 & ~voted[voter_id])
         valid_vote = 1'b1;
      else
         valid_vote = 1'b0;
   end
end

endmodule
```

```
`timescale 1ns / 1ps

module modeControl(
input clock,
input reset,
input mode,
input valid_vote_casted,
input [6:0] candidate1_vote,
input [6:0] candidate2_vote,
input [6:0] candidate3_vote,
input candidate1_button_press,
input candidate2_button_press,
input candidate3_button_press,
output [6:0] digit0,
output [6:0] digit1);

reg [6:0] num;
initial num = 100;

//mode0 -> voting mode, mode 1 -> result mode

always @(posedge clock)
begin
   if(reset) begin
      num = 0;
   end
   else
   begin
   if(~mode)         // mode = 0 -> voting mode
   num = 100;        // No digits displayed
   else              // mode = 1 -> result mode
   begin
   if(candidate1_button_press)
                num = candidate1_vote;
   else if(candidate2_button_press)
                num = candidate2_vote;
   else if(candidate3_button_press)
                num = candidate3_vote;
   end
   end
end
//digitsDisplay display_votes(.num(num), .digit0(digit0), digit1(digit1));
digitsDisplay display_votes(num, digit0, digit1);
endmodule
```

Table: Source Code for Button Control and Mode Control

```
`timescale 1ns / 1ps

module digitsDisplay (
                input [6:0] num,
    output reg [6:0] digit0, // 7-bit output for segments a to g of digit 0
    output reg [6:0] digit1 // 7-bit output for segments a to g of digit 1
);

wire [6:0] segment_data[0:9]; // Data for segments a to g for digits 0 to 9

// Define 7-segment patterns for digits 0 to 9

assign segment_data[0] = 7'b0000001; // 0
assign segment_data[1] = 7'b1001111; // 1
assign segment_data[2] = 7'b0010010; // 2
assign segment_data[3] = 7'b0000110; // 3
assign segment_data[4] = 7'b1001100; // 4
assign segment_data[5] = 7'b0100100; // 5
assign segment_data[6] = 7'b0100000; // 6
assign segment_data[7] = 7'b0001111; // 7
assign segment_data[8] = 7'b0000000; // 8
assign segment_data[9] = 7'b0000100; // 9


// Display the number
always @(*) begin
    if (num >= 0 && num <= 99) begin
        digit1 = segment_data[num / 10]; // Set segments for digit 1
        digit0 = segment_data[num % 10]; // Set segments for digit 0
    end
                else begin
        digit0 = 7'b1111111; // Display 'Err' if input is out of range
        digit1 = 7'b1111111;
    end
end

endmodule
```

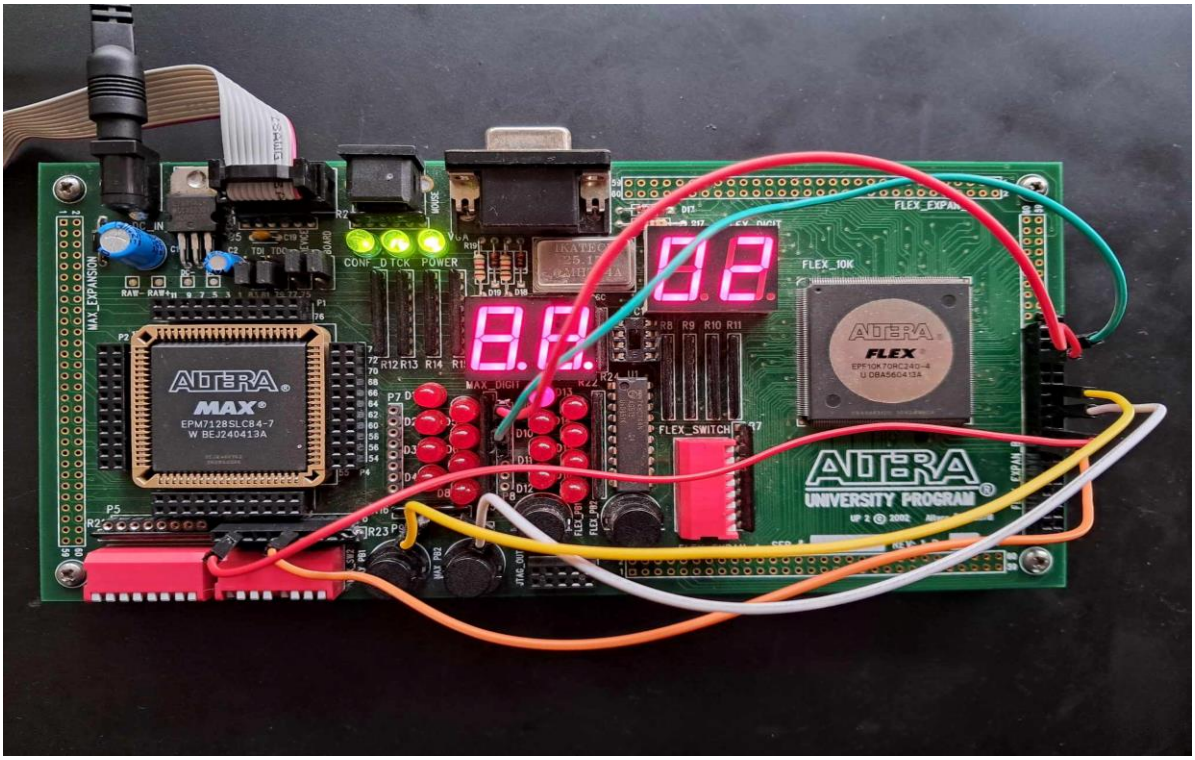
Table: Source Code for Digits Display

```
`timescale 1ns / 1ps

module votingMachine(
  input clock,
  input reset,
  input mode,
  input button1,
  input button2,
  input button3,
  input pollsig,
  input [7:0] voter_id,
  output already_voted,
  output pollsigON,
  output decimal_point0,
  output decimal_point1,
  output [6:0] digit0,
  output [6:0] digit1
);

  assign decimal_point0 = 1;
  assign decimal_point1 = 1;

  wire valid_vote_1;
  wire valid_vote_2;
  wire valid_vote_3;
  wire [6:0] cand1_vote_recvd;
  wire [6:0] cand2_vote_recvd;
  wire [6:0] cand3_vote_recvd;
  wire [255:0] voted;
  wire anyValidVote;

  wire reset_n;
  wire mode_n;
  wire button1_n;
  wire button2_n;
  wire button3_n;
  wire pollsig_n;
  wire [7:0] voter_id_n;

  assign reset_n = ~reset;
  assign mode_n = ~mode;
  assign button1_n = ~button1;
  assign button2_n = ~button2;
  assign button3_n = ~button3;
  assign pollsig_n = ~pollsig;
  assign voter_id_n = ~voter_id;

  assign pollsigON = ~(pollsig_n & mode);
  assign already_voted = ~(voted[voter_id_n] & mode);
  assign anyValidVote = valid_vote_1 | valid_vote_2 | valid_vote_3 ;

  buttonControl bc1(
    .clock(clock),
    .reset(reset_n),
    .button(button1_n),
    .mode(mode_n),
    .pollsig(pollsig_n),
    .voter_id(voter_id_n),
    .voted(voted),
    .valid_vote(valid_vote_1)
  );

  buttonControl bc2(
    .clock(clock),
    .reset(reset_n),
    .button(button2_n),
    .mode(mode_n),
    .pollsig(pollsig_n),
    .voter_id(voter_id_n),
    .voted(voted),
    .valid_vote(valid_vote_2)
  );

  buttonControl bc3(
    .clock(clock),
    .reset(reset_n),
    .button(button3_n),
    .mode(mode_n),
    .pollsig(pollsig_n),
    .voter_id(voter_id_n),
    .voted(voted),
    .valid_vote(valid_vote_3)
  );
```

```
authentication auth(
    .clock(clock),
    .reset(reset_n),
    .anyValidVote(anyValidVote),
    .voter_id(voter_id_n),
                .voted(voted)
  );

  voteLogger VL(
    .clock(clock),
    .reset(reset_n),
    .mode(mode_n),
    .cand1_vote_valid(valid_vote_1),
    .cand2_vote_valid(valid_vote_2),
    .cand3_vote_valid(valid_vote_3),
    .cand1_vote_recvd(cand1_vote_recvd),
    .cand2_vote_recvd(cand2_vote_recvd),
    .cand3_vote_recvd(cand3_vote_recvd)
  );

  modeControl MC(
    .clock(clock),
    .reset(reset_n),
    .mode(mode_n),
    .valid_vote_casted(anyValidVote),
    .candidate1_vote(cand1_vote_recvd),
    .candidate2_vote(cand2_vote_recvd),
    .candidate3_vote(cand3_vote_recvd),
    .candidate1_button_press(button1_n),
    .candidate2_button_press(button2_n),
    .candidate3_button_press(button3_n),
    .digit0(digit0),
    .digit1(digit1)
  );

endmodule
```

Table: Source Code for the Voting Machine

4 Implementation



4.1 Description

There are two modes, i.e. voting mode and result mode. In voting mode, after enabling the pole signal, only then voting is allowed, which is indicated by an LED. Otherwise, the vote will not be counted. The voting button must be pressed for minimum one second to register a vote. After registering the vote another LED will be on to indicate that the vote has been already casted for that specific ID. If the vote is cast once from an ID, it can not be cast again from that ID.

In result mode, a vote cannot be casted. In this mode, if we press the respective buttons, the total votes for the candidates will be shown on the digit display.

If the reset button is pressed at any time, the total system will be reset.

4.2 Results

We have checked it several times and this model worked perfectly.

5 Design Analysis and Evaluation

5.1 Novelty

In our system, we've implemented a two-step verification process to ensure the validity of each vote. Unlike traditional elections where physical marking is used to identify voters, we employ a distinctive serial number that allows only one vote per individual. Due to this dual verification approach in our EVM, even if someone attempts to provide an authentication signal for multiple votes under the same voter's name, those additional votes will not be considered valid. This serves as a protective measure against fraudulent voters and polling officials.

5.2 Design Considerations

5.2.1 Considerations to public health and safety

• The simplified EVM has the potential to replace paper ballots, thus diminishing paper waste.

• This semi-automated process demands less human labor in comparison to traditional methods.

• EVMs offer enhanced security compared to conventional paper ballot elections, reducing the likelihood of tampering and potentially lowering incidents of violence at polling centers.

• EVMs are faster and more user-friendly, reducing wait times for voters and providing a more comfortable voting experience, especially for elderly and physically challenged individuals.

• In some cases, ink contains heavy metals as pigments, which, if they come into contact with soil or water, can pose serious threats to public health.

5.2.2 Considerations to environment

• Standard election EVMs are battery-powered, operating with low energy consumption, making them a more environmentally friendly option.

• Elections utilizing paper ballots result in significant paper waste, whereas EVMs are a more environmentally responsible choice in this regard.

• The management of waste associated with ballot boxes and paper ballots consumes substantial energy, further emphasizing the advantages of EVMs.

• The production of paper for ballots necessitates the sacrifice of numerous trees, and the ink pigments employed in these papers contain heavy metals, posing environmental hazards.

5.2.3 Considerations to cultural and societal needs

• The youth of our nation are becoming less inclined to vote, primarily due to extended wait times and the convoluted identity verification procedures. The EVM we've developed is efficient and boasts a straightforward yet effective verification process. The implementation of user-friendly EVMs like this could potentially reverse this trend.

• Local elections in Bangladesh have a notorious reputation for manipulated outcomes. A secure EVM has the potential to restore the confidence of the general population in all electoral procedures.

5.3 Limitations of Tools

We had to execute the whole model in an FPGA device, that’s why there are some limitations. There are security and privacy problems, which will not occur in real life. When this model is used in larger scale, the poling switch, the voting or result switch, the display and the ID switch are to be to the poling officer. That’s how the limitations are to be solved.

6 Reflection on Individual and Teamwork

6.1 Individual Contribution of Each Member

ID	Contribution
1906037	Button control, Votes display and combining this discrete process, pin planning
1906045	Authentication process, display control and implementing FPGA
1906049	Base Module, mode control (Voting mode and result mode)
1906061	Counting valid votes received by each candidate (Vote logging) and operating FPGA

6.2 Mode of Teamwork

We organized both virtual and in-person meetings. Initially, we conducted online discussions using Zoom, during which we brainstormed, defined project objectives, and exchanged ideas. To maintain communication and facilitate idea and file sharing, we established a messenger group. Subsequently, we collaborated in person to develop and execute the project. We conducted code testing and debugging in Titumir Hall room.

6.3 Diversity Statement of Team

In our project group, we had a diverse composition of four members. Each member brought their unique characteristics to the team. This diversity posed its own set of challenges but also enriched our collective experience as we worked together to find common ground and create an inclusive working environment.

6.4 Log Book of Project Implementation

Week 4:Project proposal submission.

Week 5:Brainstroming Ideas.

Week 7: Borrowed FPGA Board from Lab

Week 8-10:Code Development.

Week 11:Debugging & Code adjustment.

Week12:Simulation & debugging.

Week 13:Implementation on FPGA & Project Presentation.

7 Project Management and Cost Analysis

The FPGA was borrowed from Digital Electronics Laboratory, Department of EEE, BUET.

8 Future Work

This model is currently implemented in small scale, but it can be executed in larger scale in future. THE ID button, reset button, pole signal switch and the digit display are to be separated from the users and to be kept in the pole officer's office. Also the digit display has to be increased from two digits to eight or nine digits. Moreover, the ID input system has to be more user-friendly.

9 References

https://www.ijeat.org/wp-content/uploads/papers/v9i1/A1484109119.pdf