

**ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)**

ORGANISATION OF ISLAMIC COOPERATION (OIC)

Department of Computer Science and Engineering (CSE)

**Project Report**

**CSE 4206 : Digital Logic Design Lab**

**Team Members**

Wasi Omar Syed (230042143)

Sheikh Alimun Alahi(230042115)

Md. Saif-Al Sarker (230042106 )

Md Nadid Hassan Khan (230042146)

Arian Abedin (230042145)

Shudipto Sarwar Mamun (230042142)

**Table of Contents**

[**Motivation**](#_wjf4uguakib2) **3**

[**Description**](#_uyedtmand0x0) **3**

[**Features**](#_4dz8uhnx3bbl) **3**

[**Components of the project and How they were used**](#_7sv7j7srrssa) **4**

[**Summary Table**](#_b1w9z1tm3chh) **5**

[**Project Diagram**](#_c2otm96ym6b6) **6**

[**Individual Contribution of each Team Member**](#_qx98llba06ll) **8**

[**Manual**](#_c2e2kvtvmn9z) **8**

[**Conclusion**](#_g689fa74dz19) **9**

**SecureVote**

Hardware-Based Digital Voting System with Fraud Prevention

# Motivation

Ensuring fairness, accuracy, and fraud prevention in voting systems is a critical challenge. Many existing systems allow loopholes for double voting or inaccurate tallying. This project aims to design a secure, hardware-driven voting machine capable of authenticating voters, ensuring one-vote-per-voter, and delivering real-time, reliable results without reliance on software-based security.

# Description

The proposed system is a voting machine designed to ensure secure and accurate voting through the use of digital logic components.

The system allows a voter to enter their 4-digit voter ID using push buttons. Upon entering the ID, the voter presses another button to verify their identity. The verification process checks whether the entered ID exists in the Read-Only Memory (ROM) that contains a list of authorized voter IDs. To ensure proper functioning, a clock pulse controls a 4-bit counter, which checks if the entered ID is present in the ROM. This verification is performed using logic gates to match the ID against the stored entries. If the ID is found, the voter is granted permission to vote; if not, voting is not allowed.

Additionally, the system includes 16 flip-flops, which ensure that each voter can cast their vote only once, preventing duplicate voting.

Once verified, the voter can choose from three candidates by selecting one of two buttons. After selecting their candidate, the system counts the votes. A comparator is then used to determine the winner of the election. The results are displayed on a 7-segment display, with 'A' representing the first candidate, 'B' for the second candidate, and 'C' for the third candidate. This design leverages a combination of ROM, flip-flops, counters, and logic gates to ensure the integrity and efficiency of the voting process

# Features

This voting system handles several important functionalities to ensure secure and accurate voting, including:

**1.Voter Authentication**: The system verifies each voter’s identity by checking their 4-digit voter ID against a list of authorized IDs stored in ROM.

**2.One-Vote-Per-Voter Enforcement**: The use of 16 flip-flops ensures that each voter can cast only one vote, preventing multiple votes from the same individual.

**3.Candidate Selection and Voting**: Voters can select one of three candidates using two buttons. After selecting their candidate, they can cast their vote.

**4.Voting Count Display**: The system keeps track of the votes cast for each candidate and displays the vote count.

**5.Winner Determination**: After the votes are counted, a comparator determines the winner. The winner is shown on a 7-segment display, with each candidate represented by a letter ('A' for the first candidate, 'B' for the second, and 'C' for the third).

# Components of the project and How they were used

The following components were used in this project for different purposes:

1. **JK Flip-Flops (74LS112)**
   * **Purpose**: Used for controlling the ROM and tracking the voting count. Four JK flip-flops were used for controlling the ROM, and twelve were used to track the votes for each candidate.
2. **D Flip-Flops (4013)**
   * **Purpose**: Used for holding the voting verification status and ensuring the one-vote-per-voter rule. One D flip-flop was used to hold the voting verification status, and sixteen were used to enforce the one-vote-per-voter rule by storing voting states.
3. **Comparator (7485)**
   * **Purpose**: Used to compare the vote counts for each candidate and determine the winner by selecting the highest vote count.
4. **7-Segment Display (N/A)**
   * **Purpose**: Used to display the election result. Three 7-segment displays were used to show the winner: 'A' for Candidate 1, 'B' for Candidate 2, and 'C' for Candidate 3.
5. **Multiplexer (4067)**
   * **Purpose**: Used for selecting the appropriate flip-flop during voter authentication, ensuring the system reads the correct voter ID.
6. **Decoder (74154)**
   * **Purpose**: Two decoders were used: one to decode voter IDs in ROM during authentication, and another to decode candidate selection for voting.
7. **Demux (4514)**
   * **Purpose**: Used to update the corresponding D flip-flop during voting, ensuring each voter can only vote once.
8. **Input-Logic States (N/A)**
   * **Purpose**: Used for managing the inputs: four states for the voter ID input, one for the authentication button, two for candidate selection, and one for voting.
9. **Output-Logic Probes (N/A)**
   * **Purpose**: Used to display the verification status and vote count. One probe shows the current verification status, and another shows the voting count.

# Summary Table

| Serial | Component Name | Count (optional) | Use Case |
| --- | --- | --- | --- |
| 1 | JK Flip-Flop (74LS112) | 16 | 4 for controlling ROM, 12 for voting count tracking. |
| 2 | D Flip-Flop (4013) | 17 | 1 for holding voting verification status, 16 for enforcing one-vote-per-voter. |
| 3 | Comparator (7485) | 3 | Used to compare vote counts for each candidate and determine the winner. |
| 4 | 7-Segment Display (N/A) | 3 | Used for showing the winner of the election (A for first, B for second, C for third). |
| 5 | Multiplexer (4067) | 1 | Selects the corresponding flip-flop during authentication. |
| 6 | Decoder (74154) | 2 | 1 for decoding voter IDs in ROM during authentication, 1 for candidate selection |
| 7 | Demux (4514) | 1 | Updates the corresponding D flip-flop to enforce one-vote-per-voter. |
| 8 | Input-Logic States (N/A | 8 | 4 for voter ID input, 1 for authentication button, 2 for candidate selection, 1 for voting. |
| 9 | Output-Logic Probes (N/A) | 2 | 1 for displaying vote count, 1 for showing verification status. |

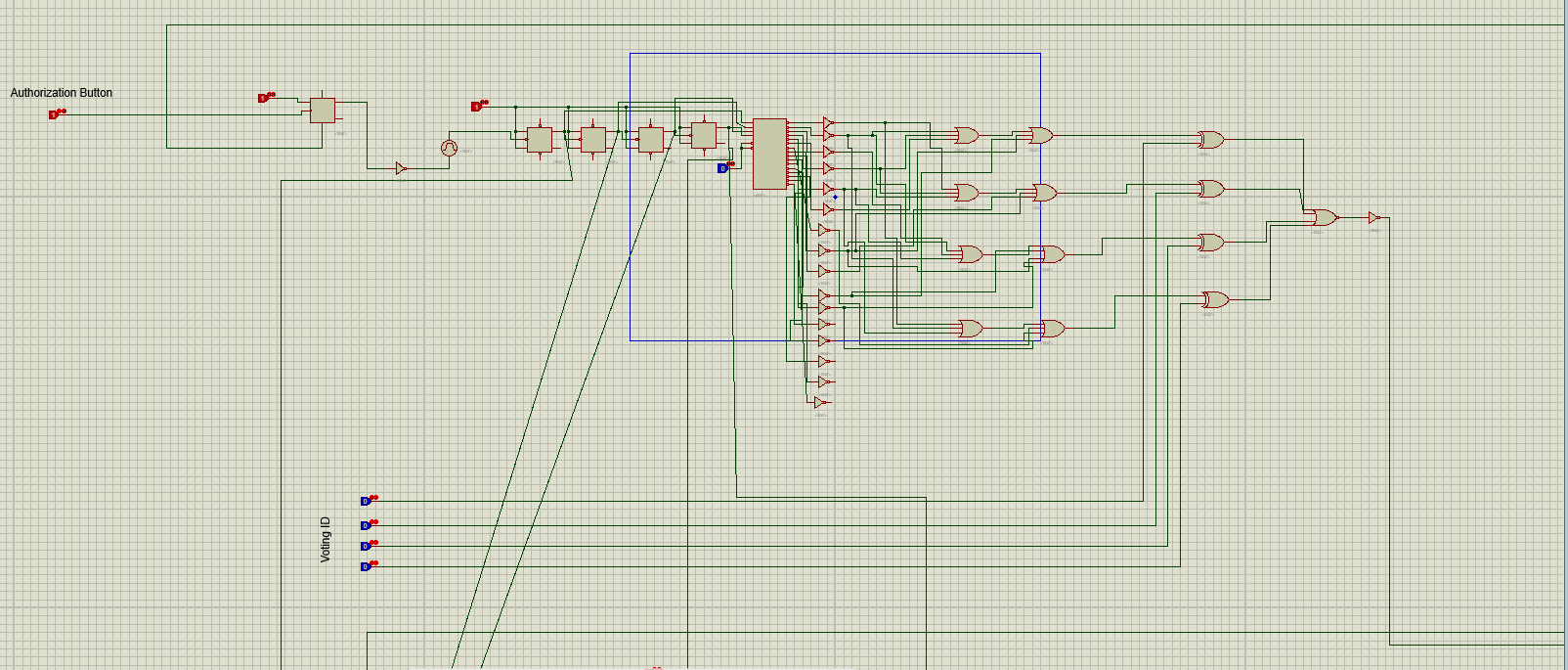
# 

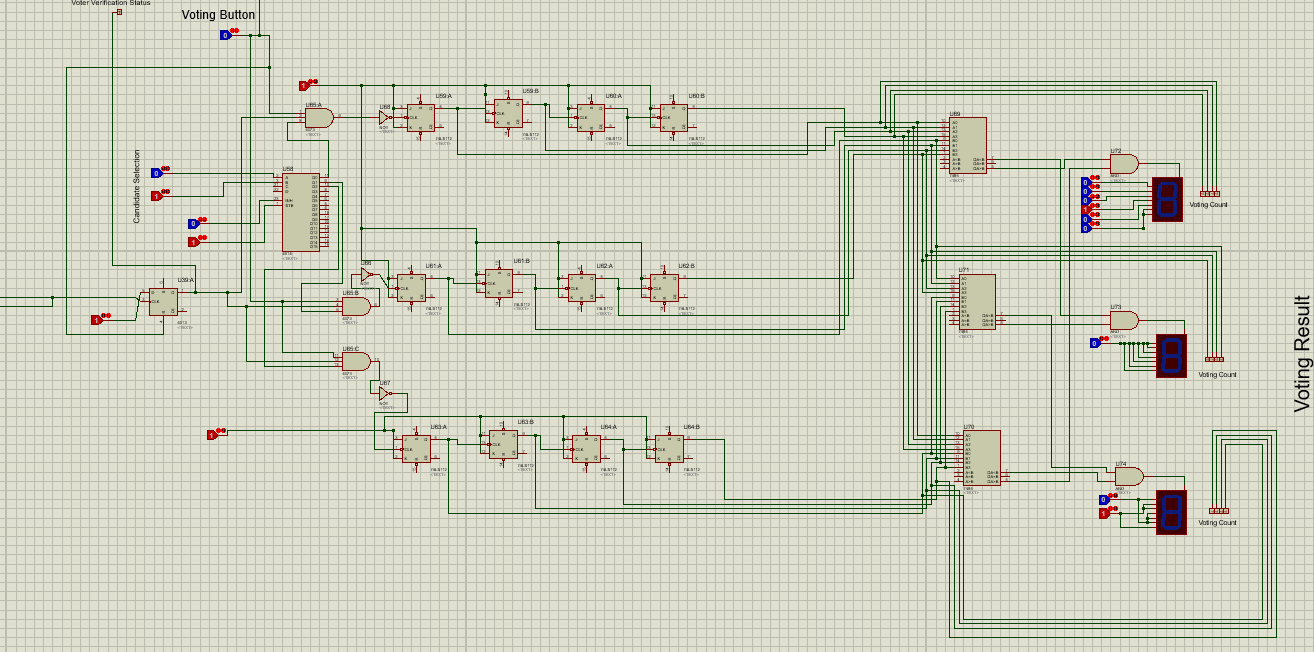
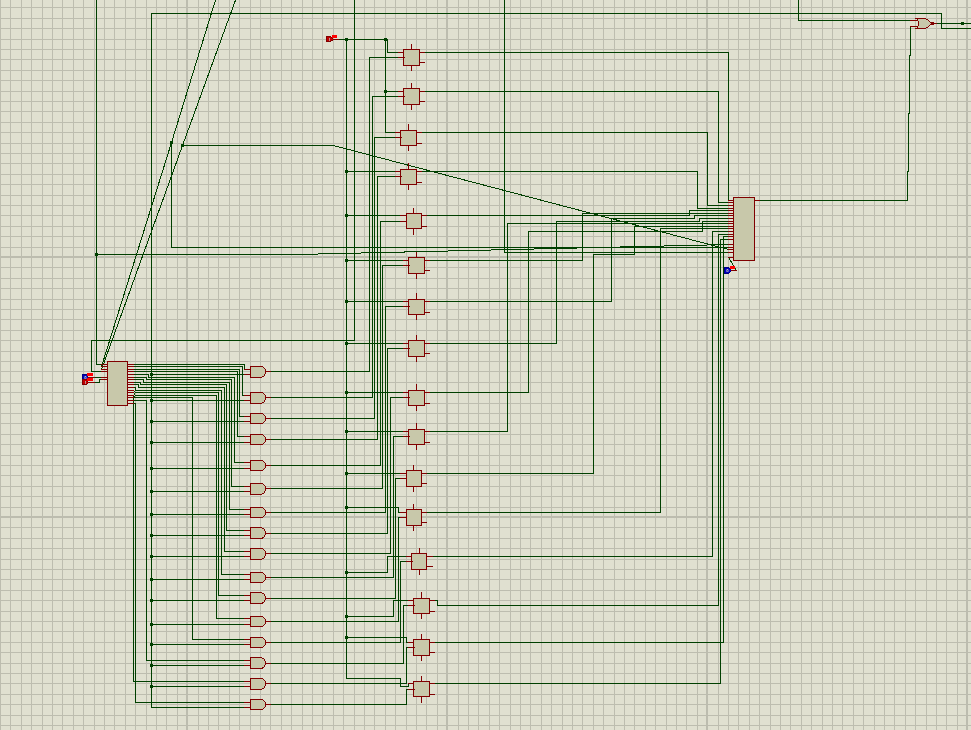
# Project Diagram

Diagram 1-Verifying voting id

Diagram 2-Ensuring one-vote-per-voter system

Diagram 3-Counting votes, Displaying winner





# Individual Contribution of each Team Member

| **Team Member** | **Contribution** |
| --- | --- |
| Wasi Omar Syed (230042143) | Overall planning of the project, integration of various components, and documentation. |
| Sheikh Alimun Alahi(230042115) | Implemented the candidate selection system, allowing voters to choose between three candidates |
| Md. Saif-Al Sarker (230042106) | Responsible for implementing the ROM authentication system to verify voter IDs |
| Md Nadid Hassan Khan (230042146) | Created the logic for comparing the vote counts and displaying the winner on the 7-segment display |
| Arian Abedin (230042145) | Developed the "One-vote-per-voter" enforcement mechanism using flip-flop flags |
| Shudipto Sarwar Mamun (230042142) | Designed and implemented the vote counting mechanism for the system |

# Manual

This section will guide you through the process of using the voting system.

1. **Enter Voter ID**
   * Use the **4 push buttons** to input your 4-digit voter ID. Each button corresponds to a binary digit (0 or 1).
2. **Verify Voter ID and One-Vote-Per-Voter Enforcement**
   * After entering your voter ID, press the **authentication button** to verify if your ID is stored in the ROM.
   * The system will check your ID against the list in the ROM. If your ID is valid, the system will verify that you haven’t already voted by checking the **16 flip-flops**. If you have already voted, the system will prevent you from voting again. If you're authorized, the system will proceed to the next step.
3. **Select a Candidate**
   * Once your ID is verified and you are authorized to vote, you can select your preferred candidate.
   * Use the **2 candidate selection buttons** to choose between the 3 candidates. Enter **00** for Candidate 1 (A), **01** for Candidate 2 (B), or **10** for Candidate 3 (C).
4. **Cast Your Vote**
   * After selecting your candidate, press the **vote button** to register your vote. The system will count the vote and store it in the voting tally.
5. **Viewing Vote Count**
   * Once all votes are cast, the system will display the total number of votes for each candidate on the **7-segment displays**.
   * The winner is determined by the **comparator**, and the result will be shown (A for the first candidate, B for the second, and C for the third).
6. **Next Vote**
   * For the next vote, ensure your **voting button** is set to **0**.
   * Select your **voter ID** again and verify by setting the **authentication button** to **0**, then press it to **1** for verification.

**Note:** The following voter IDs are **not allowed** to vote: **1001, 0110, 1101, 1111**.

# Conclusion

The proposed voting system successfully integrates key digital logic components to ensure a secure and accurate election process. By utilizing various elements such as JK flip-flops, D flip-flops, multiplexers, decoders, and comparators, the system efficiently handles voter authentication, one-vote-per-voter enforcement, candidate selection, and vote counting. The inclusion of a 7-segment display allows for clear and immediate visualization of the election results.

This design not only meets the requirements for a basic voting machine but also ensures integrity through its robust verification process, preventing unauthorized or multiple votes. With its practical application of digital logic concepts, this system serves as a reliable and straightforward model for voting systems, demonstrating the potential of digital circuits in real-world applications like electronic voting.

Future improvements could include expanding the system's functionality, such as adding more candidates or implementing a more advanced user interface, but this project successfully demonstrates the power of digital logic in creating secure, efficient, and user-friendly systems.

# 

# 