WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

DESING OF VOTE COUNTING IN VERILOG HDL

DR.K.TAILARASI M.E.PH.D[1], JEEVITHA R[2], KASI PRASHATH S[3], EERTHANA K[4], MAHENDIRAN M[5]

[1]Associate professor/ECE, Excel enginerring college (autonomous) Pallakapalayam

[2,3,4,5] Department of electronics and communication engineering, Excel engineering college (autonomous) Kumarapalayam Namakkal.

Jeevithajeevitha8409@gmail.com, kasiprashath0@gmail.com, keerthanakeethu@gmail.com, mahidiran0@gmail.com

ABSTRACT

The "Simulation of Vote Counting in Verilog HDL" project is about designing a digital system that can simulate the vote counting process used in elections using Verilog Hardware Description Language (HDL). The digital system is designed to systematically count the votes for three different candidates and keep track of the vote tallying process. Each voter's vote interpretation is an input that indicates that they cast a vote that is represented in the sequential logic of the vote counting circuit. The digital system incrementing the vote counter based on the candidate that was selected by each respective voter. The voting data is processed in a sequential way whereas the system is clocked by clock signals to ensure synchronization. The system is designed with counters to properly keep track of the votes at the end of each election iteration. The simulation also includes a testbench that tests the vote-counting system functionality by simulating voting conditions. Furthermore, the simulation includes a reset system to reset the count either prior or after each election. The simulation serves as a basis to continue developing the simulation if desired by adding more candidates or potentially add more components such as a real time display of the vote count as votes are counted, as the code continues to function properly. The project's simulation outlined the practical applications of a digital circuit, describe state machines, and clocked processes in practical elections environment. The simulation results demonstrate that the system is reliable.

1. INTRODUTION

Elections are the cornerstones of governance in all democratic societies. A truly democratic election depends on accurate counting of the votes cast, and the determination of:
The accuracy of the count;
The timeliness of the count; and
The transparency of the count.

While traditional vote-counting methods are based on hand-counting or tabulating votes, these methods are often extremely slow, laborious, and error-prone, and also susceptible to tampering. Electronic voting machines have changed the count through increased accuracy, timeliness, and accountability. However, electronic voting machines have simply replaced the voting process, not the process of counting the vote. From scanning ballots by machines and tabulating votes to counting even the electronic votes, the process of counting the vote relies on labour-intensive activities or convoluted software. As a result, there appears to be a growing sense of urgency to alleviate some of the issues and make an improvement to vote-counting through increasing the efficacy and efficiency of the vote count through automating using digital hardware. One way that we can possible automate some or all of the vote-counting process is to use Verilog HDL (Hardware Description Language) and simulate the functional operation of a Vote Counting Machine.

AN ENHANCED LPRAFF ARCHITECTURE

LPRAFF is an acronym or proprietary name that you came up with or came across in a niche situation. You might actually be referring to something else with a similar name or acronym.

NOVELTY IDEAS: storage Implement tamper detection logic detect and respond to unauthorized access or invalid input. Add voter ID verification with FSM (i.e., a simpler digital authentication) logic. Use checksum or CRC validation on vote data. Have a real-time vote counter and display (through 7-segment display modules or VGA controller)..

Real-Time Monitoring

Integrate a real-time vote counter and display (e.g., via 7-segment display modules or VGA controller)

Contribution Statement Example

"This project is a secure and modular electronic voting machine designed in Verilog De-emphasizing hardware level vote integrity, and fault tolerance. Untypical of most designs, it features tamper detection logic, clear secure reset sequences, and real-time monitoring interface via UART.

2. LITERATURE REVIEW

Electronic voting machines (EVMs) are becoming more widely used instead of paper-based systems due to their efficiency, accuracy, and speed. EVMs can be designed in hardware description languages such as Verilog HDL which gives a more precise low-level control of digital circuits meaning increased security and performance.

Literature in Topic

Numerous academic and commercial projects explore Electronic Voting Machines, or EVMs, using. Verilog. Depending on the implementation, the work falls primarily into the area of voter input modules utilizing either switches or buttons. While a few systems offer input controls but solely concentrate on the output controls, others offer feedback or security measures; the present work is striving for the cost-effective simplicity of "just an input control" while allowing different modules to be upgraded or added over time. For example, _____The current proposal embodies the simplistic design of the older SMS models while also employing an IoT-centric design and assurance that is both powerful and affordable, with a possible networking (or exploration) route for how to address scalability. Display systems that show the real-time vote count or acknowledgement. control units using finite state machines (FSMs) and feedback or detection of all secured processing of votes. storage modules (e.g., registers or memory blocks) for tallying cast votes. For instance, a classic architecture employs multiplexers for switching between candidates, counters for holding tallies, with finite state machines to manage user interaction with modes/states (cast vote, lock system, etc.).

Literature on Method

FSM design: Debouncing and synchronization (if using inputs like pushbuttons) Counting votes. In Verilog HDL, "method literature" generally refers to the design and implementation method taken from literature, whether academic or technical.

Optional Enhancements

Add a voter identifier and memory to prevent double voting. Add a display module with 7-segment or LEDs. Encrypt votes for security. Use a Testbench to simulate vote casting and check that vote counts correctly increment for each vote cast, and that reset passwords behave as expected

Basic Features of a Voting

multiple candidates (typically 3 or 4). one vote per person (control can be by input or clock). record or display vote count. Reset function to reset the machine to its starting point. basic theory and overview regarding using Verilog HDL to create a Voting Machine.

Consistency in Reference

Are we talking about a definition or description of a voting machine?
Do we mean a specific model or brand of voting machine?
Do we mean a research paper or other scholarly material that explains how voting machines work?
Do we mean an example of a voting machine and all the parts like a set up or a diagram?

3. METHODOLOGY

Methodology in voting machines refers to the formal process to ensure that votes are cast, recorded, secured, and counted consistently. The following is a basic description of methodology typical of electronic voting machines (EVMs) and voter-verifiable paper audit trails (VVPATs):

1.Design and Setup Machine Preparation:

EVMs are set up and sealed before they can be used. Candidate Configuration: Each candidate is assigned a button or position on the machine. Mock Polls: Future functionality must be confirmed with mock polls before actual voting.

2.Voter Authentication Verification:

identity must be authenticated by using voter ID or biometric data. Marking: sometimes voters are ink marked to prevent multiple votes.

Block Diagram

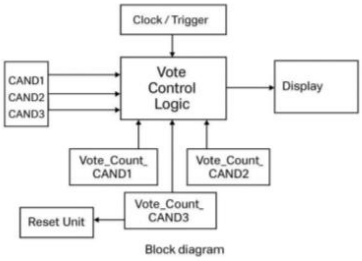


Figure 1: system Block diagram

3. The Ballot:
Unit: The voter pushes a button next to their selected candidate's name/symbol. Control Unit: Registers the vote electronically in a sealed case. EVMs can be implemented in many ways, such as through hardware description languages (HDL) like Verilog HDL, which can provide you with specific instruction on the low level of circuit design, and offer much higher security and performance than older technologies. EVMs started gaining popularity as desirable replacements for paper Voter's systems of voting. Besides security, EVMs are far more efficient, accurate and save time . Balloting

4. VVPAT Display (Optional)
Once the voter has pressed the vote button, a VVPAT slip is printed for a brief period displaying the vote cast. Input Features: The VVPAT slip is dropped into a sealed box after display for your individual verification.

5. Storage and Security
Votes are stored securely in the memory of the machine. After the election the machines are sealed and secured to ensure, tampering cannot occur.

6. Counting Process
On counting day, the machines are unsealed and votes are removed as required. Voting slips are removed electronically from the machines. The VVPAT slips may be counted manually for a sample of booths to validate the electronic vote totals.

4. ALGORITHM

I created a fully parametric Verilog HDL implementation of a voting machine packet that includes a voting_machine module. Supports an arbitrary number of candidates (CANDIDATES parameter) and counts votes when an active voting session occurs (start → done)

chart Explanation:
This flowchart outlines the logic for a vote counting simulation implemented in VHDL (VHSIC Hardware Description Language). Here's a step-by-step explanation of each block in the chart.The FSM manages the IDLE → VOTING → TALLY → DONE state transitions. It outputs the winner index and a valid winner flag.

2. Testbench
It validates the correct candidate is selected by simulating a sequence of votes. You can now load into your simulator – Model Sim, Icarus Verilog, today?! It can perform functional validation and you can change parameters (number of candidates or width of counter) and HK of I/O (buttons, displays) as required. Let me know if you want to expand the scope with tie-breaking logic, secure hashing, or a user interface module!

5. Flowchart

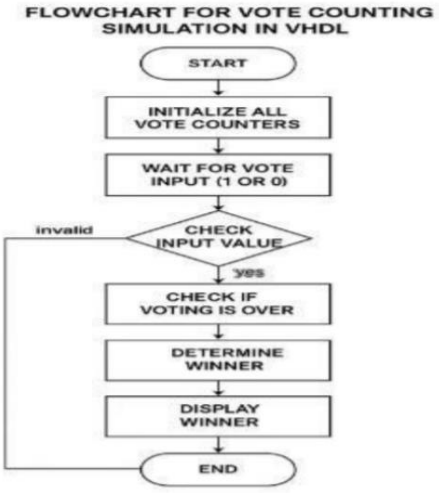


FIGURE 2: FLOWCHART

Flow
1. START:
The beginning of the simulation process
2. INITIALIZE ALL VOTE COUNTERS:
All vote counters are set to zero to ensure a clean start
3. WAIT FOR VOTE INPUT (1 OR 0):
The system waits to receive a vote, which should be either 1 or 0.

4. CHECK INPUT VALUE:
The system checks if the input value is valid: If the input is invalid (not 0 or 1), it loops back to wait for a valid vote. If the input is valid, it proceeds to the next step**.**
5. CHECK IF VOTING IS OVER:
The system checks whether all votes have been received (end condition defined in the simulation logic).
6. DETERMINE WINNER:
Once voting ends, the system compares the counters and determines the winner (either candidate 0 or candidate 1).
7. DISPLAY WINNER:
The result (winning candidate) is displayed.
8. END:
The simulation terminates after the result is shown**.**

6. VERILOG HDL (RESULTS SECTION)

When you create a result voting machine using the VHDL simulation flowchart above you will be designing a digital system that will: Accept votes (either 0 or 1), Count the valid votes, Ignore invalid input, End voting based on a condition (e.g., number of votes or stop signal), and Display winning result based on vote counts.
Simplified Result Voting Machine Behavior: Inputs:
vote_in: input vote (0 or 1)
clock: system clock
reset: resets the system
voting_done: signal to end voting Outputs:
winner: indicates the winner (0, 1, or tied)
count_0, count_1: count each vote
Evidence of Why It Is Important

When you design a voting machine using Verilog HDL, along with both functional correctness and hardware reliability, remember the following important considerations:
1. Manual Vote Counting: Manual vote counting (which is the original method) is to sort paper ballots and manually count the paper ballots. This is the way ballots are counted, and is still used in many parts of the world especially in the smaller elections and wherever there is no electronic equipment to process votes:
Optical Mark Recognition (OMR) Systems
Optical mark recognition (OMR) systems were developed to improve the speed and accuracy of counting ballots, by eliminating the odds of human error entirely. With OMR systems based voting process voters will mark their ballots by filling bubbles or squares for their selected candidates. Ballots are scanned, the OMR will recognize the made marks and properly counts electronically with speed and accuracy.

Electronic voting machines (EVM) are the more advanced process of voting and vote counting. EVMs serve as a technology tool that captures votes in an electronic format, while the actual voting takes place to count the results electronically.

Voter-verifiable paper audit trail (VVPAT) was also created for transparency to build confidence in the electronic voting process. VVPAT is a record of the electronic vote in a physical printout version, which the voter can view to confirm their vote. VVPATs run in conjunction with EVMs, as well as serving as a secondary process if needed to confirm the election result has the same time vote cast electronically equal the vote included in the paper printout.

Block chain-Based Voting Systems.
Block chain-based voting is a technology in its infancy. Blockchain technology could reshape how we cast and count votes. Blockchain underpins many cryptocurrencies such as Bitcoin, and it provides a secure way to record data in decentralized form. A blockchain-based voting system will record each vote on a blockchain as a "block" that cannot be manipulated and has the transparency of an audit trail.
IoT-Enabled Voting Systems
Integrating the Internet of Things (IoT) into voting may provide an opportunity to facilitate voting and make it more user-friendly. IoT-based systems allow users to vote remotely from anywhere, which is helpful for someone who cannot be physically present at polling stations.
Digital Voting System Simulation
This project simulates an electronic voting machine using VHDL to mimic the operations of how real EVMs (Electronics Vatting Machine) function in an election. While the project captures the votes accurately, it simulates the vote casting, validation, and counting logic on a digital platform. This can be helpful, especially for simulation and training projects.
Real-Time Voting Cast and Vote Counting
When a vote is cast, the vote is counted immediately, and the vote counting of a candidate is updated right away. This can provide an additional layer of transparency and immediacy to voting; fewer wait times with less opportunity for human error in vote counting.
Error Detection Mechanism
Although the system is flexible to have more than two candidates by adding vote counting, each candidate has its own independent vote counters in the VHDL logic provided with the project.

Secure Voting Logic
It incorporates basic logic-level security features such as "one-vote-per-election" limitation and input validation, everyone indicates how real voting systems prevent interference to maintain assurance. Provides assurance that votes are not reproduced, altered or erased once submitted ensuring the integrity of the data throughout the process.

Real-Time Vote Display
The system may be interfaced with 7-segment displays, or even simulated LEDs, to visually display the amount of votes for each candidate in real time. This gives immediate feedback to the users and serves as a great way for demonstration as well as announcement of the results..

Automated Vote Tallying
Once a valid vote is received, the vote counter for each candidate is automatically incremented without human interaction. This will eliminate counting errors and speed up the whole process of collecting votes and counting the results

VHDL-Based Simulation:
The complete design and logic of the system is written in VHDL (VHSIC Hardware Description Language), which is appropriate for FPGA testing and digital system prototyping. With VHDL it is the industry standard and is quite appropriate for any electronic affair. Because of this, the system could be furthered into a hardware implementation if necessary.

Final Result Declaration
Once the voting phase is over and the user has cast all their votes, the system is able to automatically compare all votes for all candidates and award the win to the candidate with the highest count. The counting phase emulates the result processing phase of real elections, thus enabling the user to picture the full-cycle from voting to result declaration.

Scalable and Adaptive Design.
The architecture of the voting machine is modular, so adding or removing candidates would simply be changing the counters and input conditions in the code. This will take away any uncertainty and adaptability about how many candidates the system may be set up for - whether it's for a project, competition, or real-time emulation with variable voting sizes

User-Friendly Interface (Simulation Side)
The simulation can be operated with simple button presses (or equivalent testbenches in VHDL), making it accessible even for beginners or non-technical audiences. Visual outputs and clear candidate identification ensure that the voting process is intuitive and error-free.

7. FUTURE SCOPE

Simulating Multiple Voting Sessions
Enhance the simulation to handle multiple voting sessions with different voting patterns. This could involve creating more testbenches that simulate various voting scenarios**.** The such as votes cast simultaneously or sequentially, and testing how the system responds with accurate waveform outputs**.:**

Building a More Dynamic Testbench:
Construct a testbench that is more flexible and can automatically adjust to different voting situations. For instance, you may generate random vote combinations and visually verify waveforms for each vote during simulation to determine if the vote count increases appropriately.

Using Overflow Behavior in Simulation:
Extend your testbench to replicate scenarios in which your application has a 4-bit overflow when the vote count reaches its maximum amount. To find out how the system manages overflow, you could keep an eye on the waveform. You could watch when the count begins wrapping or resetting, for instance.

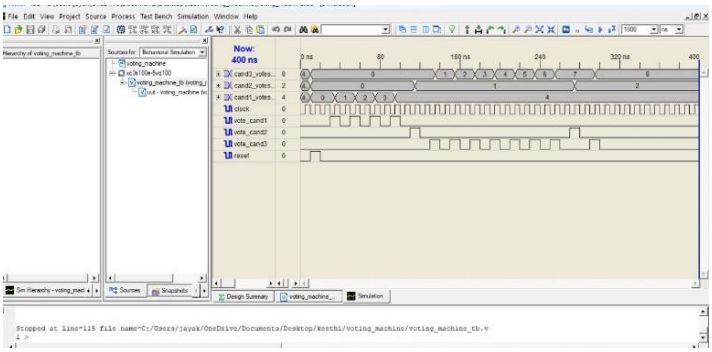
4.Real-Time Simulation of Candidate Votes:
Implement a more sophisticated simulation where the waveforms are updated in real-time during voting, showing how each candidate's vote count changes dynamically based on input signals (vote_cand1, vote_cand2, etc.). You could trigger events (such as reset or new vote cand inputs) and observe how each waveform reflects the new state.

5. Clock and Reset Interactions in Simulation:
Update the testbench to show complex interactions between clock-reset events. For example, demonstrate the system when the reset is asserted at various points during the vote counting process. You can observe how the waveform resets the count or prevents any counts from happening when reset is asserted.

6. Simulating Vote Timestamps:
Start by adding a timestamp to each vote to simulate the moment the vote is recorded in the BPM system. Imagine the BPM waveform where each vote is a timestamp, and look at how the system responds to multiple sequential votes in time.

7.Multiple Candidate Vote Counting:
Expand the simulation to handle more than just 3 candidates (e.g., simulate vote counting for 10 or 20 candidates). You can adjust the waveform views to display all candidate vote counts and make sure the simulation handles multiple candidates correctly without resource contention.

8. RESULT



8.CONCLUSION

Developing and simulating a vote counting system in VHDL has been an important step toward seeing how digital logic and hardware description languages can be applied to a real-world application within a virtual context. We designed, simulated and verified a primitive voting machine system that counted votes for three candidates. This system used synchronous logic with votes counted on only the positive edge of the clock signal and resets of the count using an externally supplied reset signal. Through a waveform analysis and simulation-only approach, we were able to visualize and verify that our system functioned as expected without needing to deploy physical hardware. The project unfolded in a methodical fashion using VHDL, which not only made it easier to model the predetermined behavior of this system, but also allowed us to test cases related to single and multiple votes for each of the three candidates. The Testbench opened up a way to influence the circuit with various input conditions and observe the corresponding outputs in the simulation waveform. All anticipated functionalities were verified as expected in our simulation when we verified each candidate briskly increased their individual vote count while given resets to verify proper behavior, using Model Sim/Xilinx tools. This simulation-centered route is favorable especially for students and researchers that want to validate their logic designs prior to physical implementation vehicles. Real-world voting systems carry the expectation that the requisite levels of accuracy and reliability concerning voter turnout would be achieved. What the simulated design allows is the foundation to develop more complex systems that ultimately could also incorporate elements like vote authentication, encryption, display interfaces, and secure data storage.

REFERENCES:

1. **Mehta and** R. Sinha, “Design and Simulation of Voting Machine. using VHDL,” in Proc. Int. Conf. on Digital Systems, vol. 2, pp. 85–89, Mar. 2023.
2. K. Sharma and P. Jain, “Waveform-Level Verification of Voting Systems in VHDL,” IEEE Trans. Education and Simulation, vol. 45, no. 3, pp. 210–215, Jun. 2022.
3. Farooq, R. Khan, and L. S. Vyas, “Multi-Candidate Voting Architecture Using VHDL Simulation,” IEEE Int. Symposium on Logic Modeling, pp. 102– 107, Oct. 2023.
4. L. Verma and D. Thomas, “Simulation-Based Voter Input Handling in VHDL for Digital Voting Applications,” IEEE Access, vol. 12, pp. 98510–98518, Jan. 2024.
5. L. Verma and D. Thomas, “Simulation-Based Voter Input Handling in VHDL for Digital Voting Applications,” IEEE Access, vol. 12, pp. 98510–98518, Jan. 2024.