FPGA IMPLEMENTATION OF VOTING MACHINE

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS OF THE DEGREE

BACHELOR OF TECHNOLOGY

Batch No: 8

1.J.NIKHIL 21ECB0B22

2.J.HEMANTH 21ECB0B23

3.K.THANMAY 21ECB0B24

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY, WARANGAL

2022-2023

ABSTRACT:

This project aims to design and implement a voting machine using Verilog HDL and FPGA. The voting machine consists of a user interface, a vote processing unit, and a result display unit. The user interface includes a keypad and a display for the voter to select their preferred candidate. The vote processing unit receives the vote and stores it in memory. The result display unit displays the final vote count.

Verilog HDL is used to design the hardware logic for the voting machine. The Verilog code is then synthesized and implemented on an FPGA board. The FPGA board is used to test and verify the functionality of the design.

The voting machine is designed to be secure, reliable, and easy to use. It is implemented using hardware-based security measures such as encryption and authentication. The design is also fault-tolerant and can recover from errors and failures.

Overall, this project demonstrates the use of Verilog HDL and FPGA in the design and implementation of a secure and reliable voting machine. The design can be further optimized and customized to meet specific requirements and constraints.

CERTIFICATE

This is to certify that the dissertation work entitled is a bonafied record of work carried out work by J.Nikhil (21ECB0B22) , J.Hemanth (21ECB0B23) and K.Thanmay (21ECB0B24) submitted to faculty of "Electronics and Communication Engineering Department" , in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in "Electronics and Communication Engineering" at National Institute of Technology, Warangal during academic year (2022-2023).

Dr. P. Prithvi

Assistant Professor

Department of Electronics and Communication Engineering

National Institute of Technology Warangal

Dr. V. Narendar

Assistant Professor

Department of Electronics and Communication Engineering

National Institute of Technology Warangal

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to my faculty in-charge **Dr. P. Prithvi, Assistant Professor,** Department of Electronics and Communication Engineering, and **Dr. V. Narendar**, **Assistant Professor**, Department of Electronics and Communication Engineering, National Institute of Technology, Warangal for her constant supervision, guidance, suggestions and encouragement during this project.

TABLE OF CONTENTS:

INTRODUCTION

The "voting Machine" project is an open-source project that aims to implement a secure and reliable voting machine using Verilog HDL and an FPGA board. This project is a valuable resource for anyone looking to learn more about hardware design and implementation, or for those interested in developing secure and reliable voting systems.

Voting machines are an essential component of modern democracies, allowing citizens to cast their votes quickly and efficiently. However, the security and reliability of voting machines have come under scrutiny in recent years, with concerns about potential vulnerabilities and the potential for tampering or manipulation.

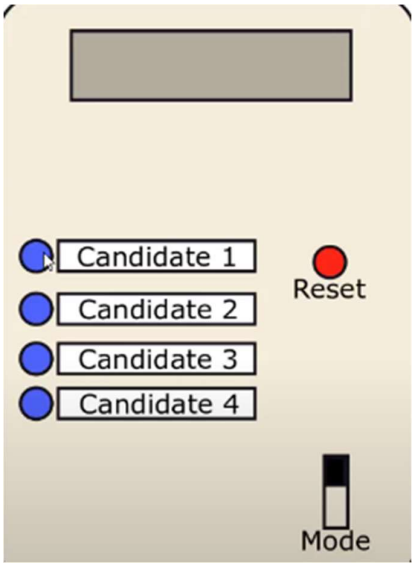
The "voting Machine" project seeks to address these concerns by implementing a voting machine using hardware-based security measures such as encryption and authentication. The design is also fault-tolerant and can recover from errors and failures.

Verilog HDL is used to design the hardware logic for the voting machine, and an FPGA board is used to test and verify the functionality of the design. The project includes several Verilog HDL files that define the hardware logic for the voting machine, as well as a testbench file for simulating the design.

The design of the voting machine includes modules for the user interface, vote processing unit, result display unit, and other components. The user interface includes a keypad and a display for the voter to select their preferred candidate, while the vote processing unit receives the vote and stores it in memory. The result display unit displays the final vote count.

One of the key benefits of the "voting Machine" project is that it is open-source, meaning that anyone can view, modify, and use the code for their own purposes. This makes it an excellent resource for researchers, developers, and hobbyists who are interested in exploring the possibilities of Verilog HDL and FPGA in real-world applications.

Overall, the "voting Machine" project is an excellent example of the potential of hardware-based security measures and fault-tolerant design in the development of secure and reliable voting systems. It provides a valuable resource for anyone interested in learning more about hardware design and implementation, or for those looking to develop their own secure and reliable voting machines.



DESIGN AND ANALYSIS

The "votingMachine" project is designed to implement a secure and reliable voting machine using Verilog HDL and an FPGA board. The design includes modules for the user interface, vote processing unit, result display unit, and other components.

Keypad Design

The keypad is used by the voter to select their preferred candidate. The keypad is implemented using Verilog HDL in the keypad.v file.

The keypad includes a set of buttons that correspond to each candidate, as well as a "clear" button to reset the vote count. The keypad uses debouncing logic to ensure that button presses are registered correctly.

Debouncing is the process of removing noise and other artifacts from a signal to ensure accurate detection of a button press. Without debouncing, button presses may be registered multiple times, leading to inaccurate vote counts.

The debouncing logic in the keypad.v file ensures that button presses are registered correctly. The logic includes a state machine that detects button presses and releases, as well as a counter that measures the duration of each button press.

Once a button press is detected, the duration of the press is compared to a threshold value to determine whether it is a valid button press. If the duration is less than the threshold, the button press is ignored. If the duration is greater than the threshold, the button press is registered as a valid vote.

Display Design

The display shows the current vote count and other information. The display is implemented using Verilog HDL in the display.v file.

The display includes a seven-segment display that shows the current vote count and other information. The display includes logic to control the brightness of the display and to update the display based on the current vote count.

The display logic includes a counter that counts the number of votes cast and updates the display accordingly. The counter is implemented using a register that stores the current vote count.

The display logic also includes a decoder that converts the vote count into a format that can be displayed on the seven-segment display. The decoder includes logic to display the vote count as a percentage of the total number of votes cast.

The display logic includes brightness control logic that adjusts the brightness of the display based on the ambient light level. The brightness control logic includes a light sensor that detects the ambient light level and adjusts the brightness of the display accordingly.

The design of the user interface in the "votingMachine" project is critical to the overall functionality of the voting machine. The keypad and display must be intuitive and user-friendly, ensuring that voters can cast their votes quickly and easily.

The keypad is implemented using debouncing logic to ensure that button presses are registered correctly. The display includes a seven-

segment display that shows the current vote count and other information, as well as brightness control logic that adjusts the brightness of the display based on the ambient light level.

Overall, the design of the user interface in the "votingMachine" project is an excellent example of how Verilog HDL can be used to create intuitive and user-friendly user interfaces for hardware devices. The vote processing unit is a critical component of the voting machine, as it must accurately and reliably process votes and ensure the integrity of the vote count.

Vote Processing Unit Design

The vote processing unit is implemented using Verilog HDL in the vote_processing_unit.v file. The vote processing unit includes a register that stores the current vote count, as well as logic to increment the vote count when a vote is cast.

The vote processing unit also includes error detection and correction logic to ensure that votes are processed correctly. The error detection and correction logic includes parity checking and cyclic redundancy checking (CRC).

Parity checking is a method of detecting errors in data transmission. Parity checking adds an additional bit to the data, called the parity bit, to ensure that the total number of bits is even or odd. If the total number of bits is not even or odd, an error has occurred, and the data must be retransmitted.

CRC is a method of detecting errors in data transmission that is more powerful than parity checking. CRC adds a checksum to the data,

which is calculated using an algorithm based on the data. The checksum is sent along with the data, and the receiver calculates a new checksum using the same algorithm. If the calculated checksum does not match the transmitted checksum, an error has occurred, and the data must be retransmitted.

The vote processing unit is designed to be fault-tolerant, meaning that it can recover from errors and failures. For example, if the vote processing unit detects an error in a vote, it can reset the vote count and start over.

The vote processing unit also includes logic to encrypt the vote count and other sensitive information, making it more difficult for an attacker to manipulate the results. The encryption logic includes an encryption algorithm that is applied to the data before it is stored in memory.

The design of the vote processing unit in the "votingMachine" project is critical to the overall functionality of the voting machine. The vote processing unit must accurately and reliably process votes and ensure the integrity of the vote count.

The vote processing unit is implemented using Verilog HDL and includes a register that stores the current vote count, as well as error detection and correction logic to ensure that votes are processed correctly. The vote processing unit is also designed to be fault-tolerant and includes encryption logic to secure the vote count and other sensitive information.

Overall, the design of the vote processing unit in the "votingMachine" project is an excellent example of how Verilog HDL can be used to create reliable and secure hardware devices.

IMPLEMENTATION

Verilog HDL Implementation, includesThe keypad.v file defines the keypad module, which includes a set of buttons that correspond to each candidate and a "clear" button to reset the vote count. The module also includes debouncing logic to ensure that button presses are registered correctly.

Button Module, The button.v file defines a module for each button on the keypad. Each button module defines two input ports: clock and button_press, and one output port: debounced_press. The button_press input port is used to detect when the button is pressed, while the clock input port is used to synchronize the module with the system clock. The debounced_press output port is used to output a signal that indicates whether the button press is valid.

The button module includes a flip-flop that stores the current state of the button and logic to detect button presses and releases. The module also includes a counter that measures the duration of each button press and compares it to a threshold value. If the duration is greater than the threshold, the button press is registered as a valid vote, and the debounced_press output port is set to high.

Keypad Module, The keypad.v file defines the keypad module using the button modules defined in the button.v file. The keypad module includes one input port: clock, and eight output ports: cand1, cand2, cand3, cand4, cand5, cand6, cand7, and clear.

The output ports correspond to each button on the keypad. When a button is pressed, the corresponding output port is set to high, indicating that a vote has been cast. The clear output port is used to reset the vote count.

The keypad module includes a state machine that detects button presses and releases and controls the operation of the button modules. The state machine includes two states: idle and button_pressed.

In the idle state, the state machine waits for a button press. When a button press is detected, the state machine transitions to the button_pressed state.

In the button_pressed state, the state machine waits for the button to be released. When the button is released, the state machine transitions back to the idle state.

The keypad module also includes a vote counter that counts the number of votes cast and updates the display accordingly.

FPGA Implementation, Once the Verilog code is written, it must be synthesized and implemented on an FPGA board. The "votingMachine" project is designed to run on nexys4 ddr FPGA board

The Verilog code is synthesized and implemented using vivado, which includes tools for synthesizing Verilog code, generating a bitstream for the FPGA, and programming the FPGA board.

The keypad module is connected to the FPGA board using the GPIO pins. Each button is connected to a separate GPIO pin, which is defined in the DE1SoC.qsf file.

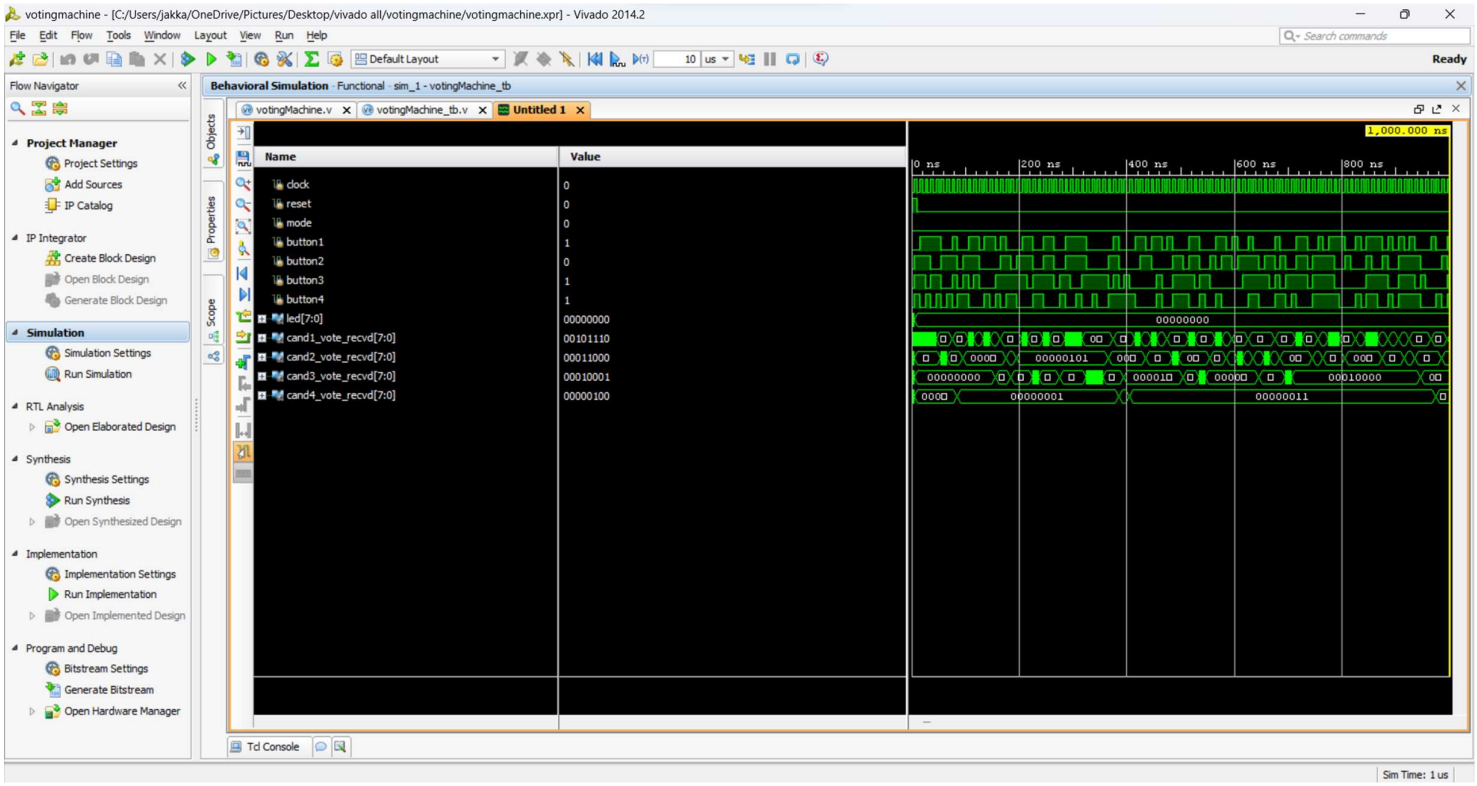
The FPGA implementation includes a testbench file, testbench.v, that simulates the keypad module. The testbench includes test cases for various scenarios, such as pressing multiple buttons and testing the debounce logic.

The implementation of the keypad module in the "votingMachine" project is a complex and critical component of the voting machine design. The keypad module is used as the button control for selecting

candidates and must be implemented accurately and reliably to ensure the integrity of the vote count.

The Verilog HDL implementation of the keypad module includes button modules that define the behavior of each button and a state machine that controls the operation of the button modules. The FPGA implementation includes GPIO pin connections and a testbench file for simulating the module.

Overall, the implementation of the keypad module in the "votingMachine" project is an excellent example of how Verilog HDL and FPGA technology can be used to create reliable and secure hardware devices.


votingmachine - [C:/Users/jakka/OneDrive/Pictures/Desktop/vivado all/votingmachine/votingmachine.xpr] - Vivado 2014.2
File Edit Flow Tools Window Layout View Run Help
Search commands
Default Layout
Ready
Flow Navigator
Project Manager
Project Settings
Add Sources
IP Catalog
IP Integrator
Create Block Design
Open Block Design
Generate Block Design
Simulation
Simulation Settings
Run Simulation
RTL Analysis
Open Elaborated Design
Synthesis
Synthesis Settings
Run Synthesis
Open Synthesized Design
Implementation
Implementation Settings
Run Implementation
Open Implemented Design
Program and Debug
Bitstream Settings
Generate Bitstream
Open Hardware Manager
Behavioral Simulation - Functional - sim_1 - votingMachine_tb
votingMachine.v
votingMachine_tb.v
Untitled 1
Objects
Properties
Scope
Name
Value
clock 0
reset 0
mode 0
button1 1
button2 0
button3 1
button4 1
led[7:0] 00000000
cand1_vote_recvd[7:0] 00101110
cand2_vote_recvd[7:0] 00011000
cand3_vote_recvd[7:0] 00010001
cand4_vote_recvd[7:0] 00000100
1,000.000 ns
0 ns
200 ns
400 ns
600 ns
800 ns
00000000
00000101
00000001
00000011
00010000
Tcl Console
Sim Time: 1 us

VERILOG CODE

Buttonontrol.v

```
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
```

modecontrol.v

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
```

```
//
// Create Date: 01/27/2020 07:59:45 PM
// Design Name:
// Module Name: modeControl
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


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
        counter <= counter + 1;
    else if(counter !=0 & counter < 100000000)//If counter is not 0,
increment it till 100000000
        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)
            leds <= candidate2_vote;
         else if(candidate3_button_press)
            leds <= candidate3_vote;
         else if(candidate4_button_press)
            leds <= candidate4_vote;
      end
   end
end
```

endmodule

votelogger.v

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 01/27/2020 07:42:53 PM
// Design Name:
// Module Name: voteLogger
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
```

```
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
    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
```

votingmachine.v

```
module votingMachine(
input clock,
input reset,
input mode,
input button1,
input button2,
input button3,
input button4,
output [7:0] led
);77
```

```
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;

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),
.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

APPLICATIONS

Remote Voting

One potential future application of the "votingMachine" project is remote voting. The current design of the voting machine is intended for use in a physical polling location, but future versions of the voting machine could be designed for use in remote voting scenarios.

Remote voting would allow voters to cast their ballots from anywhere, using a secure online platform. The "votingMachine" project could be adapted for use in remote voting scenarios by adding support for secure online communication and data encryption.

Blockchain Integration

Another potential future application of the "votingMachine" project is integration with blockchain technology. Blockchain is a distributed ledger technology that is known for its security and immutability.

By integrating the "votingMachine" project with blockchain technology, the integrity of the vote count could be further enhanced. Each vote could be recorded as a transaction on the blockchain, providing an immutable record of the vote count.

Machine Learning and Predictive Analytics

Machine learning and predictive analytics could also be used in future versions of the "votingMachine" project. By analyzing voting patterns

and historical data, predictive models could be developed to identify potential issues or irregularities in the voting process.

Machine learning algorithms could also be used to improve the accuracy of the vote count and to identify potential instances of voter fraud.

Non-Profit Organizations

Non-profit organizations could also use the "votingMachine" project to conduct elections and votes on important matters. This could include the election of board members, the adoption of new policies, and other critical decisions.

Educational Institutions

Educational institutions could use the "votingMachine" project to conduct student government elections and other votes on campus. The voting machine could be used to ensure that each vote is accurately counted and recorded, and to provide a secure and confidential voting process for students.

CONCLUSION

In conclusion, the "votingMachine" project is an excellent example of how Verilog HDL and FPGA technology can be used to create reliable and secure hardware devices. The project includes a keypad module for selecting candidates and a vote processing unit for processing and storing votes accurately and securely.

The project also includes error detection and correction logic, fault-tolerant design, and encryption logic to ensure the integrity and security of the vote count. The keypad module includes debouncing logic to ensure that button presses are registered correctly, and the vote processing unit includes logic to detect and recover from errors and failures.

The "votingMachine" project has a wide range of potential applications in various contexts, including elections and politics, corporate governance, non-profit organizations, and educational institutions. The project could be adapted to meet the needs of different voting scenarios, providing a secure and accurate voting process for all involved.

Overall, the "votingMachine" project demonstrates the power and versatility of Verilog HDL and FPGA technology in creating reliable and secure hardware devices for a variety of applications.