Visvesvaraya Technological University
Belagavi

A Mini Project Report

on

ELECTRIC VEHICLE CHARGING CONTROLLER

Submitted by

Name: Anupama S	**USN:1NH22EC020**
Name: Harika T	**USN:1NH22EC057**

In partial fulfilment for the award of the

degree of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

NEW HORIZON
COLLEGE OF ENGINEERING
New Horizon Knowledge Park, Ring Road, Marathalli
Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade, Accredited by NBA

Bengaluru – 560 103

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

Certified that the Mini project entitled **"ELECTRIC VEHICLE CHARGING CONTROLLER"** is carried out by **ANUPAMA S** bearing USN: **1NH22EC020** and **HARIKA T** bearing USN: **1NH22EC057**, bonafide students of NHCE, Bengaluru in partial fulfilment for the award of Bachelor of Engineering in Electronics and Communication of the Visvesvaraya Technological University, Belagavi during the year 2023-24. It is certified that all corrections and suggestions indicated for Internal Assessment have been incorporated in the report deposited in the department library. The mini project report has been approved as it satisfies the academic requirements in respect of the mini project work prescribed for the said degree.

Signature of the guide	Signature of the HoD
Mr. Krishnakumar R N	Dr. Aravinda K
Assistant Professor	Professor & HoD
Department of ECE	Department of ECE
NHCE, Bengaluru	NHCE, Bengaluru

Name of the Examiner	Signature of the Examiner
1.	1.
2.	2.

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be impossible without the mention of the people who made it possible, whose constant guidance and encouragement crowned our efforts with success.

We have great pleasure in expressing gratitude to **Dr. Mohan Manghnani**, Chairman, New Horizon Educational Institutions, for providing necessary infrastructure and creating good environment.

We take this opportunity to express our profound gratitude to **Dr. Manjunatha**, Principal, New Horizon College of Engineering, for his constant support and encouragement.

We would like to thank **Dr. R J Anandhi**, Professor and Dean-Academics, NHCE, for her valuable guidance.

We would also like to thank **Dr. Aravinda K** Professor and HOD, Department of Electronics and Communication Engineering, for his constant support.

We also express our gratitude to **Mr.Krishnakumar R N**, our project guide, for constantly monitoring the development of the project and setting up precise deadlines. Their valuable suggestions were the motivating factors in completing the work.

Finally, a note of thanks to all the teaching and non-teaching staff of Dept of Electronics and communication Engineering, for their cooperation extended to us, and our parents and friends, who helped us directly or indirectly in the course of the project work.

ANUPAMA S (1NH22EC020)
HARIKA T (1NH22EC057)

ABSTRACT

The development of electric vehicles (EVs) is increasing at a very fast pace and new technology to store and manage enough energy for sustainable energy management but what really is of concern is the sustainability of the charging activities. This project is about simulating a charging controller for electric vehicles using Verilog which is a hardware description language (HDL) and a digital system modeler at the same time.
The controller manages essential tasks, including:
Dynamic Slot Allocation: It is a system that manages the charging of electric vehicles (EVs) by assigning slots to different EVs dynamically based on the state of charge (SoC) and charging power of each EV.
Priority Management: These vehicles are poorly charged and their delivery rate of power needs to be disturbed to ensure effective energy distribution.
Charging Time Estimation: Calculating the estimated charging duration considering the state of charge (SOC) of the vehicle.
Charging Control Modes: Current limiting and voltage limiting were implemented in the simulation as well as voltage and current monitoring in the CC and CV charging modes, respectively.

A model made of Verilog is subjected to total simulation and validation that uses tools such as ModelSim and Quartus, guaranteeing the controller's functionality and performance. The simulated scenario gives details about different battery chargers asked for charging by the users of a charging station, one of them getting out of order which causes user disturbance and also the power flow manager not having enough power.

This simulation-based approach is a cheap and flexible picture of how the design of the EV charging infrastructure would turn out and especially now the infrastructural units can be upgraded further. Therefore, it is an effective approach for evaluation and design of the EV charging system

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

The rise of electric vehicles (EVs) in our country has significantly increased the call for more efficient, safer, and smarter solutions of charging. One of the important parts of the EV infrastructure is its charging controller which is in charge of charging, ensuring safety, and optimizing energy utilization. The design and functionality of the charging controller should be evaluated before going for hardware implementation. The most cost-effective and flexible way to evaluate it is through Verilog, a hardware description language which is widely used.

The main aim of this work is to simulate an EV charging controller with various innovative features for charging stations. The virtual controller dynamically controls the charging slot that gives a guarantee of resource optimization and the accommodation of different vehicle requirements. A priority-based system is activated to make vehicles with a critically low battery the first to be attended to. In addition, the controller uses the estimated average charging time based on the car's depleted state of charge (SOC) and the maximum available power, which is a smart approach to the management of energy.

The controller doesn't only perform the simulation of the charge but also implements a priority-based system to switch the charging current during car's state of charging detectable if the SOC abruptly changes. The main functions of the system such as real-time monitoring of voltage and current, fault detection and dynamic power management are exactly the ones being integrated so that they will work perfectly in real-life situations. The simulation environment, consisting of Verilog and executed in tools like ModelSim or Quartus, Has enabled a detailed analysis of the controller's aforementioned behavior under diverse conditions, including plume stress and fault.

The complexity of hardware implementation is absent in the simulation method, as a result, the controller's behaviour and characteristics is looked at in a more profound way. The knowledge gained from these simulations can help in polishing the process and the hardware-output, seeing the light of day it will let us to have a substantial step forward in the development of smarter, more efficient EV charging systems in highly dynamic environment.

LITERATURE REVIEW

Dynamic Charge Allocation and Prioritization

- Author: Zhang
- Published: Published in 2020.
- Focus: A system for dynamic assignment of available charging slots of EVs according to their State of Charge (SOC).

Book outcomes:

1) Subordinating: Operating vehicles with minimal battery are charged first in order to meet the install requirements.
2) Dynamic Alteration of Slot Assignment: Slot assignment is done with respect to the SOC of each vehicle in order to increase throughput at the station.
3) Simulation Method: The controller's decision-making process is modeled and implemented through Verilog and it is able to receive multiple charging requests at the same time and process them accordingly.
4) Testing: This system validation simulation proves the system is effective in real life providing different charging interaction without equipment implementation.

Constant Current and Constant Voltage Charging Simulation

Authors: Kumar

Publication Year: 2019

Focus: Simulation of constant current (CC) and constant voltage (CV) charging modes for battery management in Electric Vehicles (EVs).

Book outcomes:

1) Key Features: Modes of Charge: Initially, RC charging mode uses constant current to charge at full rates and switches over to CV after the battery reaches full charge to avoid overcharging.
2) Real-Time Monitoring: Voltage and current levels are continuously monitored throughout the charging process.
3) Fault Detection: Integrating fault detection for monitoring the health of the system to avoid an overload situation or any other damage.
4) Simulation Method: The complete charge-discharge cycle from switching between CC

5) and CV mode is modeled in Verilog for simulation of behavior of various charging scenarios by a change in a system response.
6) Testing: The simulations permit detailed performance evaluation of the controller under overload,
7) underload conditions, and changing demand with charging.
8) Contribution: This study demonstrates how adaptive charging control and real-time measurements in controlling the charging process would lead to better EV safety and efficient charge using Verilog to model complex charging behaviors and safety features.

OBJECTIVES

1. Build a model of a charging controller.

- Perform the design and modeling of an EV charging controller employing logic in Verilog that will supervise the entire process of charging.

2. Allocate the Charging Slots Automatically

- Devise a system that is able to allocate charging slots to many EVs efficiently depending on the number of resources at the charging station.

3. Implement a fully automated prioritization system which prioritizes any vehicles which are low on battery.

- In order to heighten the level of the system efficiency, a charging order is created that helps the vehicles which have low left charge on their batteries to be charged first.

4. Compute charging time.

- Come up with a formula that is able to yawn out estimates of charging time for every EV based on its SOC as well as the amount of power supplied in order to enable great scheduling.

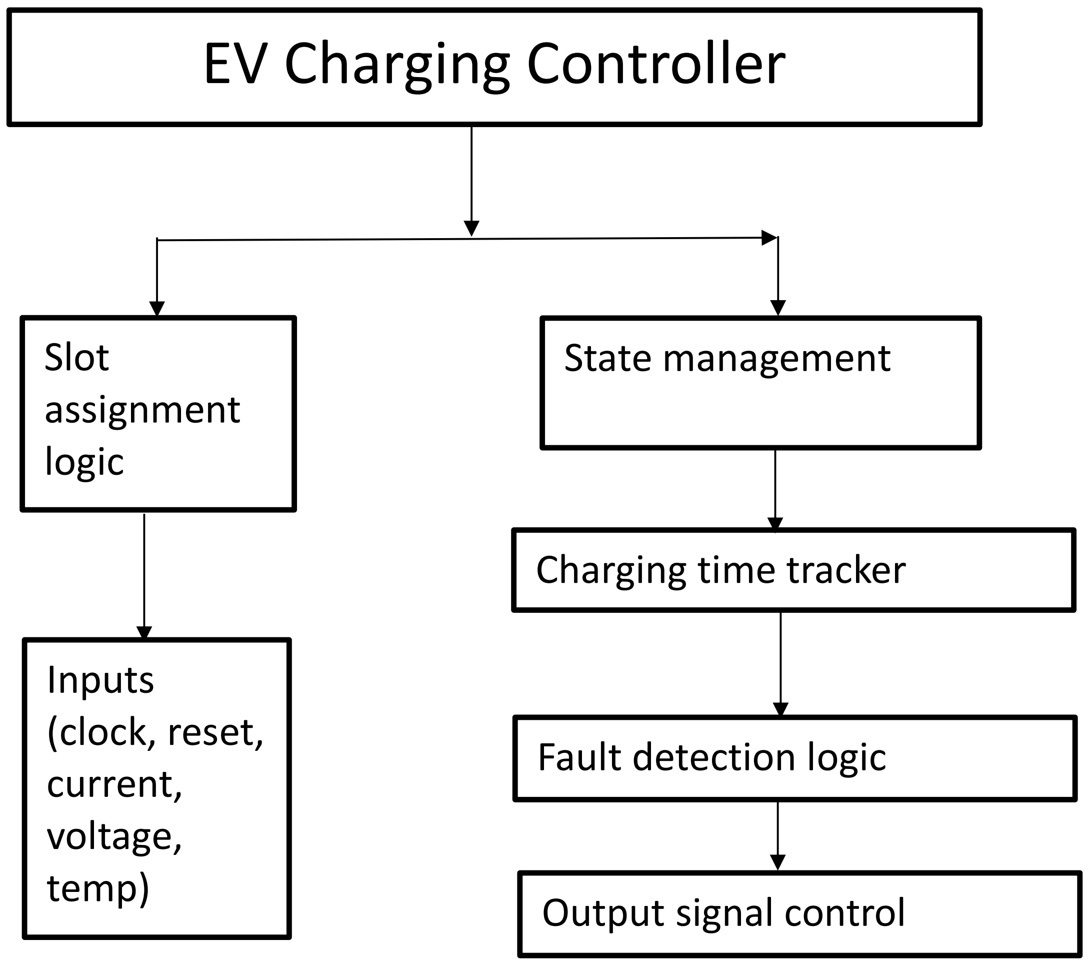
5. Test the different charging modes.

- Construct and validate the model which encompasses the transitioning of EV batteries from constant current (CC) charging mode to constant voltage (CV) charging and vice versa.

6. Test the system through simulation.

- Perform simulations and check what the outcomes are in terms of the behavior and the effectiveness of the charging controller in situations like peak times when there are more batteries that need charging.

BLOCK DIAGRAM



Explanation

The block diagram gives an overview of the EV Charging Controller structure along with its major components describing its logical flow of operations and interaction. Each block represents some functionality that needs to be implemented to provide charging control and state management and ensure system reliability. The following gives an elaborate explanation.

EV Charging Controller (Main Block):
This is the core block that orchestrates the working of all the other sub-blocks. It consists of input processing, slot allocation (assignment), state processing, fault detection, and control of outputs to appropriately control the charging process.

Slot Assignment Logic:
This module dynamically assigns charging slots to EVs. In addition, it ensures the availability of slots and gives priority to vehicles with lower battery levels. It also updates the slot allocation logic based on real-time input current and voltage.

Inputs (Clock, Reset, Current, Voltage, Temperature):
These are the basic inputs to the system.
Clock: Provides timing signals for synchronous operations.
Reset: Resets the controller to the marked state.
Current/Voltage: Monitors the battery's charging status.
Temperature: Ensures safe operations by preventing any overheating conditions.

State Management:
This block implements the finite state machine that manages states like IDLE, CHARGING, FULL, and ERROR. It transitions through states based on input conditions, ensuring a safe and seamless charging process.

Charging Time Tracker:
This module looks after tracking the time taken for each charging. It monitors the charging time of each EV and gives input for switching states, whereby charging moves to FULL.

Fault Detection Logic:
This continuously monitors the system's inputs (e.g., temperature) against certain faults: over-temperature and abnormal current levels. Whenever a fault is identified, it enters the ERROR-state, placing it in a safer operating range.

Output Signal Control:

It generates the necessary signals to control charging operations such as switching the charger on, off, etc.

METHODOLOGY

Requirements Analysis

The processes of functional and non-functional requirements analysis for the EV charging controller are set to be the first step.
The following aspects are notable:
Inputs: clock, reset, current, voltage, and temperature;
Outputs: charging status (on/off), assigned slot ID, charging duration, and fault codes;
Safety: over-temperature and over-voltage fault detection;
Prioritization: dynamic charging slot allocation, taking into consideration the EV with the least charged battery.

System Design

Block Diagram Development

The overall system design is illustrated in the block diagram above:

The main blocks include:

- **Slot assignment logic:** Dynamically assigns charging slots.
- **State Management:** Implements state machines with states such as IDLE, CHARGING, FULL, and ERROR.
- **Charge Time Clock:** Measures the time for which charging is done.
- **Fault Detection Logic:** Detects conditions such as overheating.
- **Output Signal Control:** Manages charging signals.

Finite State Machine (FSM) Design

The FSM is the core of the controller, specifying operating states and transitions with respect to input conditions. Such states are as follows:
IDLE: The system is initialized and waits for the voltage to cross some threshold.
CHARGING: The charging process is in work.
FULL: The charging cycle is finished.
ERROR: A fault condition has been declared.

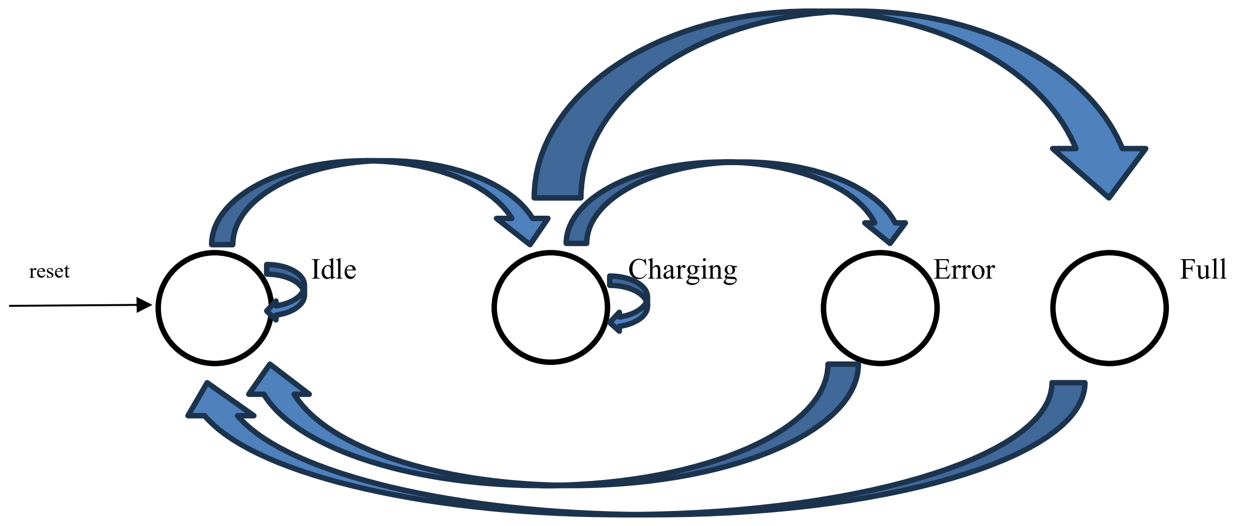


Fig 1.1: State Diagram

Current State	Input Condition	Next State	Output (if any)
Idle	Start Charging	Charging	None
Idle	Reset	Idle	None
Charging	Battery Full	Full	None
Charging	Error Detected	Error	None
Charging	Reset	Idle	None
Error	Reset	Idle	None
Full	Reset	Idle	None

Table 1.1: State table of EV Charging Controller

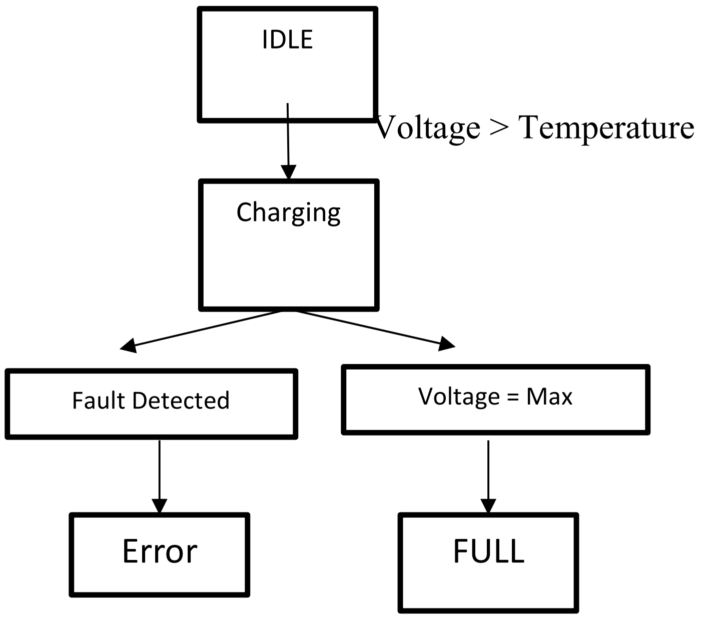


Fig 1.2: Flow Chart for State Transition

The flowchart below depicts the state transitions of a charging system.

1. **IDLE**:

- This is the initial state whereby the system is waiting for an input condition to start charging.
- Transition-if the voltage crosses the threshold, it enters the CHARGING state.

2. **CHARGING:**

- The system charges the battery or the connected device.
- Transitions:
  - If a fault is detected, it goes to the ERROR state.
  - If the voltage reaches a maximum limit, it transits to the FULL state.
  - It continues in the CHARGING state in other cases.

3. **ERROR:**

- The system identifies an issue or fault during charging.
- Transition-It continuously remains in the ERROR state until a reset signal is received.

4. **FULL:**

- The system is finished charging because the voltage is at its maximum level.
- Transition-Resets to IDLE once more for another cycle.

Important points:

- The system switches between these states based on input conditions, mainly voltage values, fault detection, and reset signals.

- In terms of error handling, the flow enables safe and effective charging by stopping when maximum voltage is reached.

Implementation

The EV Charging Controller Project carries with it multiple functional modules, addressing tasks requiring their specifications.

1. Slot Assignment Logic

Purpose:

- Dynamic assignment of charging bays to EVs.
- Prioritization according to the starting battery charge level and availability.

Functionality:

- It uses the counter to increment and allocate available slots.
- It reuses commanded slots effectively after charging is complete.

Inputs: Reset, clock signal and battery priority levels.

Outputs: Assigned slot ID.

Key Features:

- Maintains fairness for slot assignment.
- Eliminates deadlocks or resource conflicts.

2. State Machine

Purpose:

- Controls the entire operation of the EV charging controller.
- Manages the states of the system, IDLE, CHARGING, FULL, and ERROR.

Functionality:

- Monitors input (voltage, temperature) to prompt state transitions.
- Determines whether to charge, stop, or take corrective action.

Inputs: Voltage, temperature, and fault-detection signals.

Outputs: Current state of the system.

Key Features:

- Incorporation of a sturdy state machine.
- Unique state provision for unexpected inputs.

3. Charging Time Tracker

Purpose:

- Monitors EV charging duration.

Functionality:

- A counter remains active during charging by registering a clock signal.
- It tracks total time charged and updates the value of charging time.

Inputs: Clock signal, states.

Outputs: Total charging time (either in "seconds or cycles").

Key Features:

- Creates a fair system of tracking charging time.
- Resets up the cartridges of total charge when charging has been performed.

4. Fault Detection Logic

Purpose: Monitors certain characteristics of the system (voltage, temperature) to find faults.

Functionality:

- It can detect violations such as:
- Over-temperature (e.g., temperature > 80°C).
- Over-voltage (e.g., voltage > maximum possible threshold).
- Generates fault codes that indicate a type of error.

Inputs: Temperature, voltage, and state signal.

Outputs:

- Fault codes (e.g., over-temperature or over-voltage).
- State transition into ERROR state.

Key Features:

- Provides additional safety measures in the system, protecting it from damage.
- Allows the system to reset after the faults have been cleared.

Future Work

Hardware Implementation

1. **FPGA Prototyping**: Put the design on an FPGA development board (e.g. from Xilinx or intel FPGA) for testing in a real-world scenario. This will allow the validation of simulation results in a physical environment along with verification for timing constraints and power consumption.
2. **Hardware-in-the-Loop (HIL) Testing:** Alter a combination of the FPGA with actual hardware components like chargers, power supplies, and communication interfaces, etc., to validate the interaction of the software and hardware in a real-time manner**.**

Safety and Fault Detection

1. **Fault Detection System:** Develop fault detection and mitigation mechanisms that control safe charging under a variety of conditions (overvoltage, overheating, short circuits, etc.). Moreover, implementing diagnostic modules, and reporting faults in real time.
2. **Overcurrent Protection:** Employ real-time protection algorithms that would disconnect the EV from the charging station whenever any unauthorized variables were detected.

SOFTWARE REQURIMENTS

Software used: Quartus Prime Lite Edition (Intel FPGA) Version 18.0 and 18.1 does not need a license.

Family: MAX 10 (DA/DF/DC/SA/SC)

Name filter (Device):

10M50DAF484C7G **EDA Tool Options:**

In ModelSim-Altera

C:\intelFPGA_lite\18.0\modelsim_ase\win32aloem

RTL Viewer:

Tools

Netlist Viewers

RTL Viewer

Simulation:

Tool name: ModelSim-Altera

Compile test bench

Run Simulation

Tool RTL

Simulation

RESULTS AND WAVEFORMS

RTL Schematic View

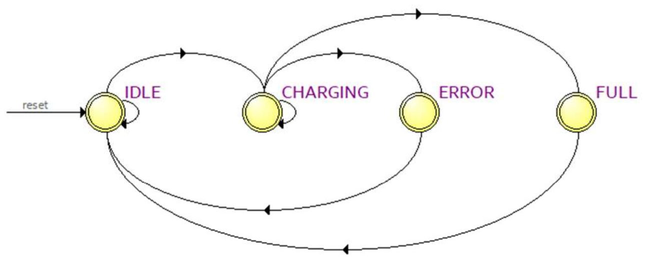


FIG 2.1: Finite State Machine

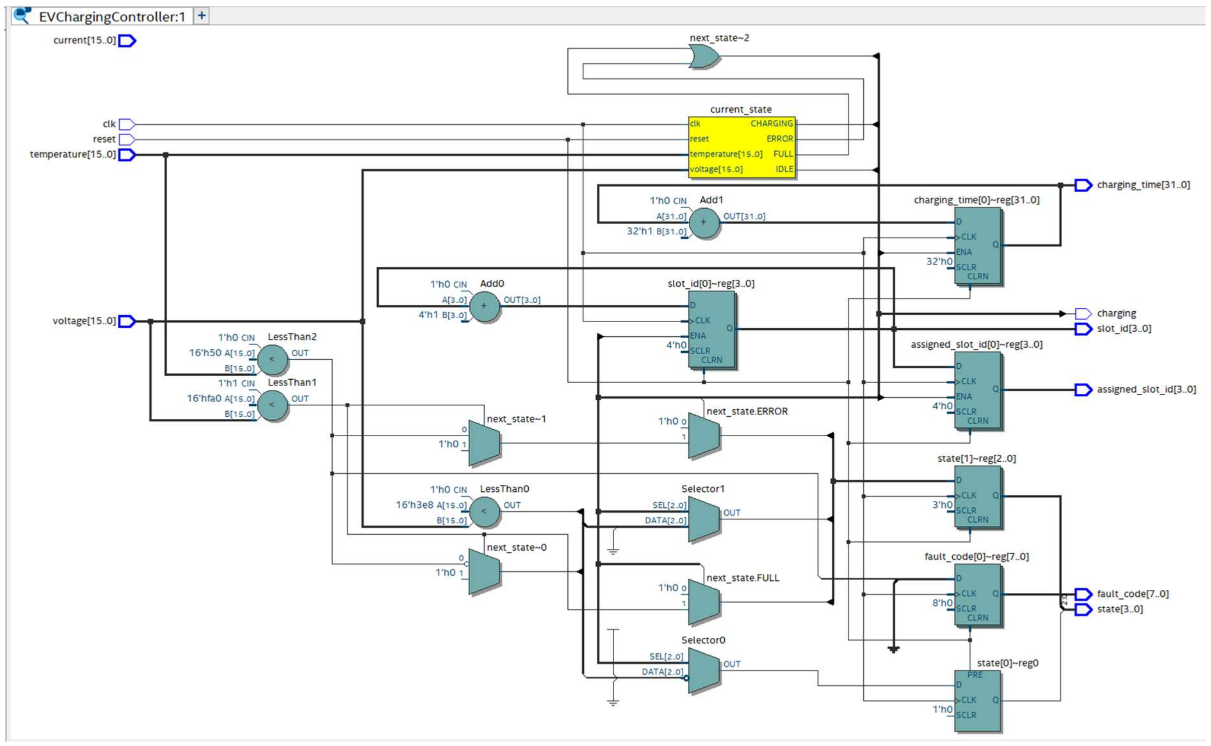


FIG 2.2: RTL View

Output Waveforms

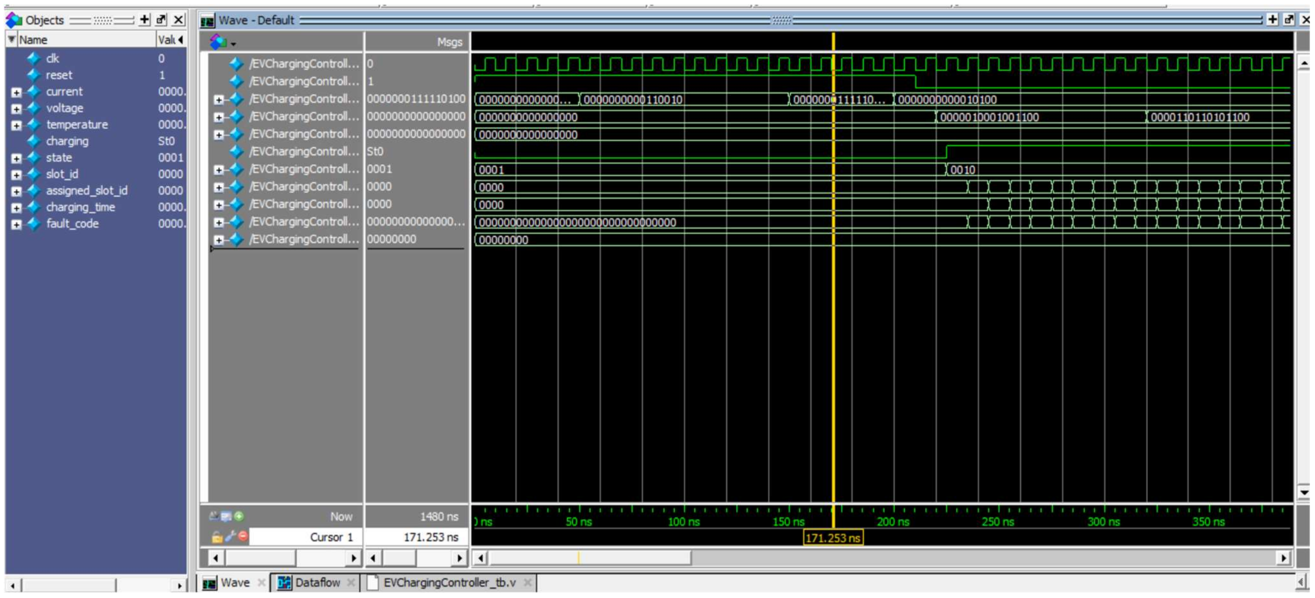


FIG 2.3: Simulation Output for EV Charging Control

CONCLUSION

The EV Charging Controller project makes a huge stride in developing electric vehicle infrastructure. Some highlights of this project are summarized below:

Efficient Charging Management:

- The controller was capable of dynamic slot allocation, providing maximum resource utilization.
- Low battery vehicles are given priority for charging, leading to reduced wait time and enhanced overall user satisfaction.

Fault Detection and Safety:

- Fault detection logic increases the reliability of the system by providing early detection and handling of occurrences such as over-temperature and over-voltage conditions.
- Safety features make the charging process safer.

Finite State Machine (FSM):

- Utilization of FSM ensures systematic and predictable operations.
- States such as IDLE, CHARGING, FULL, and ERROR allow the controller to effectively adapt to real-time inputs.

Real-Time Tracking:

- The charging time tracker gives sufficient information about the ongoing status of the charging process.
- Users are kept apprised of progress and time duration.

Modular Design:

- The modular structure allows for scalability and easy maintenance.
- Each module, that is, slot assignment logic, fault detection, and output signal control, is designed for the efficient performance of the system.

Implementation and Validation:

- The implementation and simulation results in Verilog validate the functionality and robustness of the system.
- The controller was successfully tested for a multitude of real-world scenarios.

This project highlights implementing an effective balance of safety, efficiency, and user-centered design into a reliable and innovative EV Charging Controller.

REFRENCES

1. Williamson S. S., A. Emadi, and K. Rajashekara, (2007). Comprehensive efficiency modeling of electric traction motor drives for hybrid electric vehicle propulsion applications. IEEE Trans. Veh. Technol., vol. 56, no. 4, pp. 1561–1572.Digital Logic Design: Principles and Practices by Norman Balabanian and Bradley Carlson

2. Abd El, A. A. E. B., Halim, E. H. E. B., El-Khattam, W., & Ibrahim, A. M. (2022). Electric vehicles: a review of their components and technologies. International Journal of Power Electronics and Drive Systems (IJPEDS), 13(4), 2041-2061

3. Xu M., Q. Meng (2020). Optimal deployment of charging stations considering path deviation and nonlinear elastic demand. Transp. Res. Part B Methodol. 135, 120–142.

4. Stoychev, Ivan, et al (2018). Sensor data fusion in the context of electric vehicles charging stations using a Network-on-Chip. Microprocessors and Microsystems 56, 134-143.

APPENDIX

RTL CODE

```
module EVChargingController (
    input wire clk,
    input wire reset,
    input wire [15:0] current,
    input wire [15:0] voltage,
    input wire [15:0] temperature,
    output reg charging,
    output reg [3:0] state,
    output reg [3:0] slot_id,
    output reg [3:0] assigned_slot_id,
    output reg [31:0] charging_time,
    output reg [7:0] fault_code
);

// State encoding
localparam IDLE = 4'b0001, CHARGING = 4'b0010, FULL = 4'b0100, ERROR = 4'b1000;

// State register
reg [3:0] current_state, next_state;

// Charging time calculation
reg [31:0] start_time;

// Assign slot logic: Ensure incremental behavior
always @(posedge clk or posedge reset) begin
    if (reset) begin
        slot_id <= 4'b0000;
        assigned_slot_id <= 4'b0000;
    end else if (current_state == CHARGING) begin
        slot_id <= slot_id + 1;
        assigned_slot_id <= slot_id;
    end
end

// State transition
always @(posedge clk or posedge reset) begin
    if (reset)
        current_state <= IDLE;
    else
        current_state <= next_state;
end

// Next state logic: Refine the logic to match any conditions properly
always @(*) begin
    case (current_state)
        IDLE: begin
            next_state = (voltage > 1000) ? CHARGING : IDLE;
```

```
 end
        CHARGING: begin
  if (voltage >= 4000) next_state = FULL;
            else if (temperature > 80) next_state = ERROR;
            else next_state = CHARGING;
        end
        FULL: next_state = IDLE; // Reset to IDLE after full charge
        ERROR: next_state = IDLE; // Reset to IDLE after error
        default: next_state = IDLE;
    endcase
end

// Charging time tracking
always @(posedge clk or posedge reset) begin
    if (reset) begin
        charging_time <= 0;
        start_time <= 0;
    end
    else if (current_state == CHARGING) begin
        if (start_time == 0)
            start_time <= charging_time;
        charging_time <= charging_time + 1; // Increment counter
    end
end

// Fault detection: Add any specific fault detections here
always @(posedge clk or posedge reset) begin
    if (reset)
        fault_code <= 8'b00000000;
    else if (temperature > 80)
        fault_code <= 8'b00000001; // Over-temperature fault
    else
        fault_code <= 8'b00000000; // No fault
end

// Output logic
always @(*) begin
    case (current_state)
        IDLE: charging = 1'b0;
        CHARGING: charging = 1'b1;
        FULL: charging = 1'b0;
        ERROR: charging = 1'b0;
    endcase
end
// Assign state
always @(posedge clk or posedge reset) begin
    if (reset)
        state <= IDLE;
    else
        state <= next_state;
end

endmodule
```

TESTBENCH

```
`timescale 1ns/1ps

module EVChargingController_tb;

// Testbench signals
reg clk;
reg reset;
reg [15:0] current;
reg [15:0] voltage;
reg [15:0] temperature;
wire charging;
wire [3:0] state;
wire [3:0] slot_id;
wire [3:0] assigned_slot_id;
wire [31:0] charging_time;
wire [7:0] fault_code;

// Clock generation
always #5 clk = ~clk; // 100MHz clock

// Instantiate the EV Charging Controller
EVChargingController uut (
   .clk(clk),
   .reset(reset),
   .current(current),
   .voltage(voltage),
   .temperature(temperature),
   .charging(charging),
   .state(state),
   .slot_id(slot_id),
   .assigned_slot_id(assigned_slot_id),
   .charging_time(charging_time),
   .fault_code(fault_code)
);

// Initialize the inputs
initial begin
   // Monitor important signals
   $monitor($time, " state=%b, slot_id=%d, assigned_slot_id=%d, charging_time=%d,
fault_code=%b",
            state, slot_id, assigned_slot_id, charging_time, fault_code);

   // Initialize signals
   clk = 0;
   reset = 1;
   current = 16'd0;
   voltage = 16'd0;
   temperature = 16'd0;
```

```
#50 current = 16'd50;  // Normal current

#100 current = 16'd500; // Over-current scenario
#50 current = 16'd20;  // Low current


    // Reset the system
    #10 reset = 0;

    // Apply test vectors
    #10 voltage = 16'd1100; // Start charging
    #100 voltage = 16'd3500; // Still charging
    #100 voltage = 16'd4500; // Full charge
    #100 voltage = 16'd0; // Back to idle

    #10 temperature = 16'd90; // Over-temperature fault
    #100 temperature = 16'd70; // Normal temperature

    // Simulate additional charging cycles
    #50 voltage = 16'd1200; // Start charging
    #100 voltage = 16'd3700; // Still charging
    #100 voltage = 16'd4600; // Full charge
    #100 voltage = 16'd0; // Back to idle

    // End simulation
    #500 $finish;
end

endmodule
```