GOVERNMENT COLLEGE OF TECHNOLOGY, COIMBATORE -641013

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

BIAS PROJECT – 2024

# FSM BASED RAILWAY TRACK INTERLOCKING SYSTEM

SUBMITTED BY,

HARI PREETH D M – 71772214115

MAGESH KUMAR E – 71772214127

PRAKASH M – 71772214136

SILAMBARASAN C V - 71772214147

## ABSTRACT :`

This project focuses on developing an Automated Railway Track Switching and Signaling System using Finite State Machine (FSM) logic within a Computer-Based Interlocking (CBI) framework, specifically aimed at addressing issues highlighted by recent railway accidents in Odisha. Such incidents often stem from human errors in track switching and signal operations, leading to devastating consequences. The FSM in our system controls the states of track switching and signal operations by dynamically transitioning between states like idle, safety check, route locking, switch configuration, and error handling based on real-time conditions. By automating these processes, the system enhances railway safety, minimizes human error, and optimizes train routing efficiency. The integration of FSM logic ensures that the system adapts to changing inputs with precise control, providing a scalable and reliable solution for modern railway networks. Ultimately, this project aims to eliminate the potential for human intervention errors, thus significantly improving the overall safety and reliability of railway operations, and preventing tragic accidents similar to those that have occurred in Odisha.

## ALGORITHM:

**STEP 1:** System Reset: When reset is asserted, the FSM goes to the IDLE state.

**STEP 2:** IDLE: The system stays idle, waiting for a train to be detected (train_detect = 1).

**STEP 3:** Train Detected: Once a train is detected, it moves to the ROUTE_REQUEST state**.**

**STEP 4**: Route Request: The system waits for a valid route request from the cloud (route_request = 1).

**STEP 5:** Platform Check: The FSM checks the assigned platform (platform_num).

**STEP 6**: Platform Availability: If the platform is occupied (platform_occupied = 1), it waits until it's free.

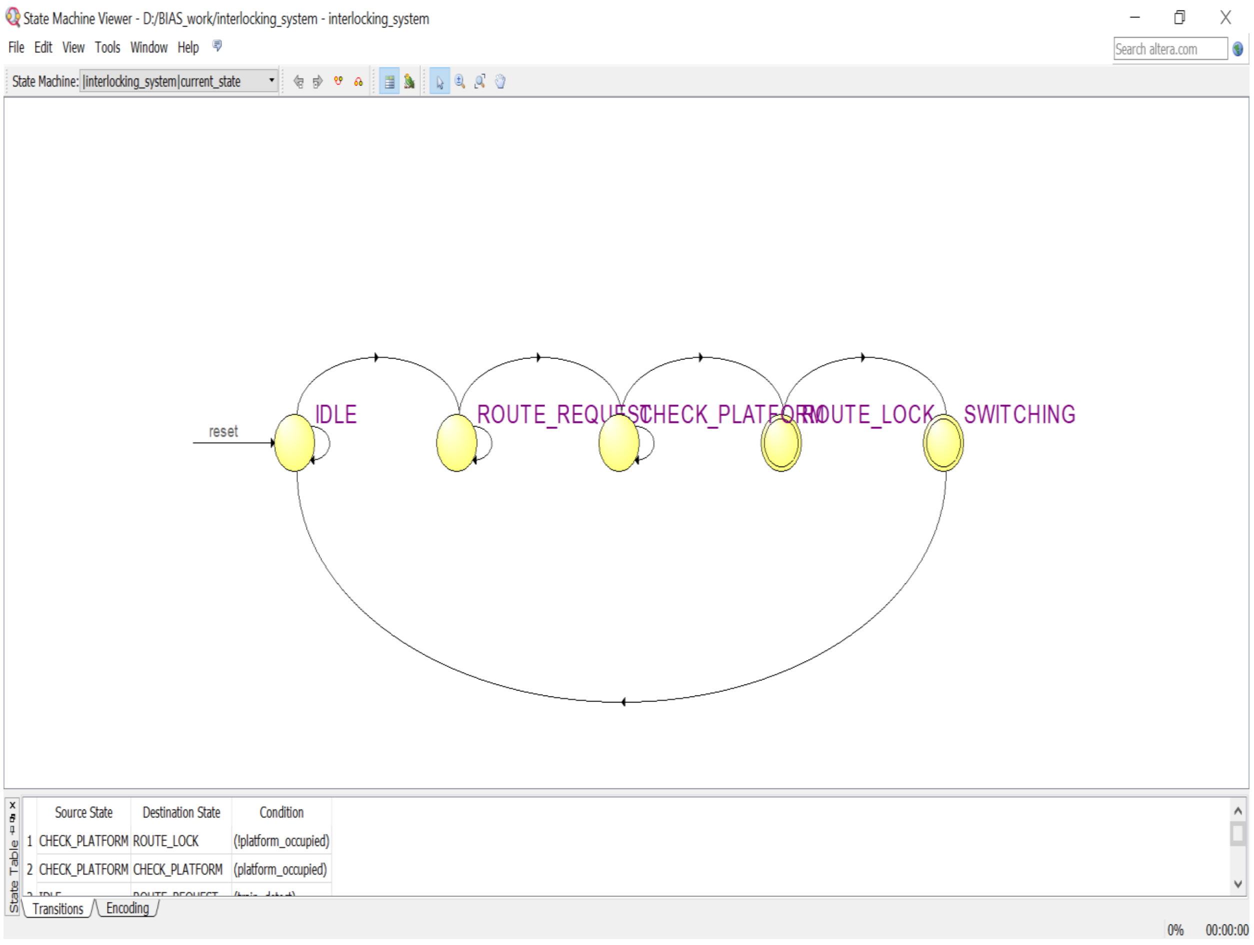
**STEP 7:** Platform Free: Once the platform is free (platform_occupied = 0), it proceeds to lock the route.

**STEP 8:** Route Lock: The system locks the track for the train (track_locked = 1).

**STEP 9:** Train Switching: The train is switched to the assigned track/platform (train_switched = 1).

**STEP 10:** Return to IDLE: After switching, the system resets and goes back to the IDLE state, ready for the next train**.**

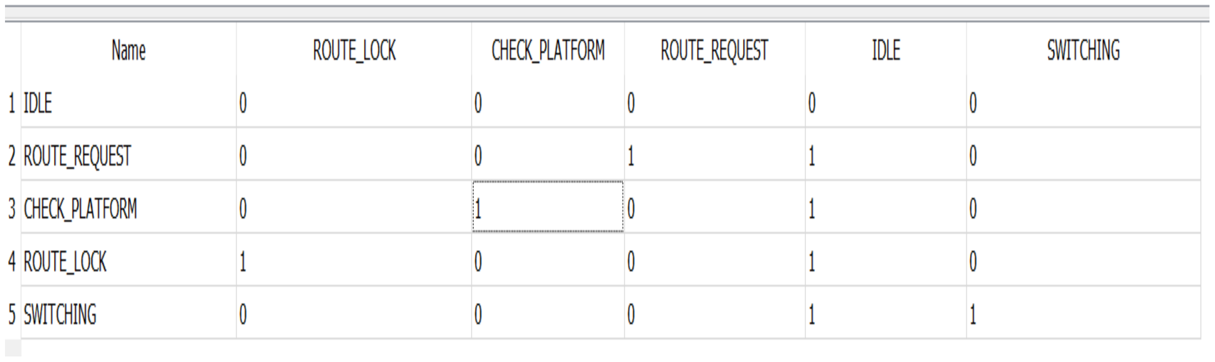
STATE DIAGRAM:



State Machine Viewer - D:/BIAS_work/interlocking_system - interlocking_system
File Edit View Tools Window Help
Search altera.com
State Machine: |interlocking_system|current_state
reset
IDLE
ROUTE_REQUEST
CHECK_PLATFORM
ROUTE_LOCK
SWITCHING
State Table
Transitions
Encoding
0%
00:00:00

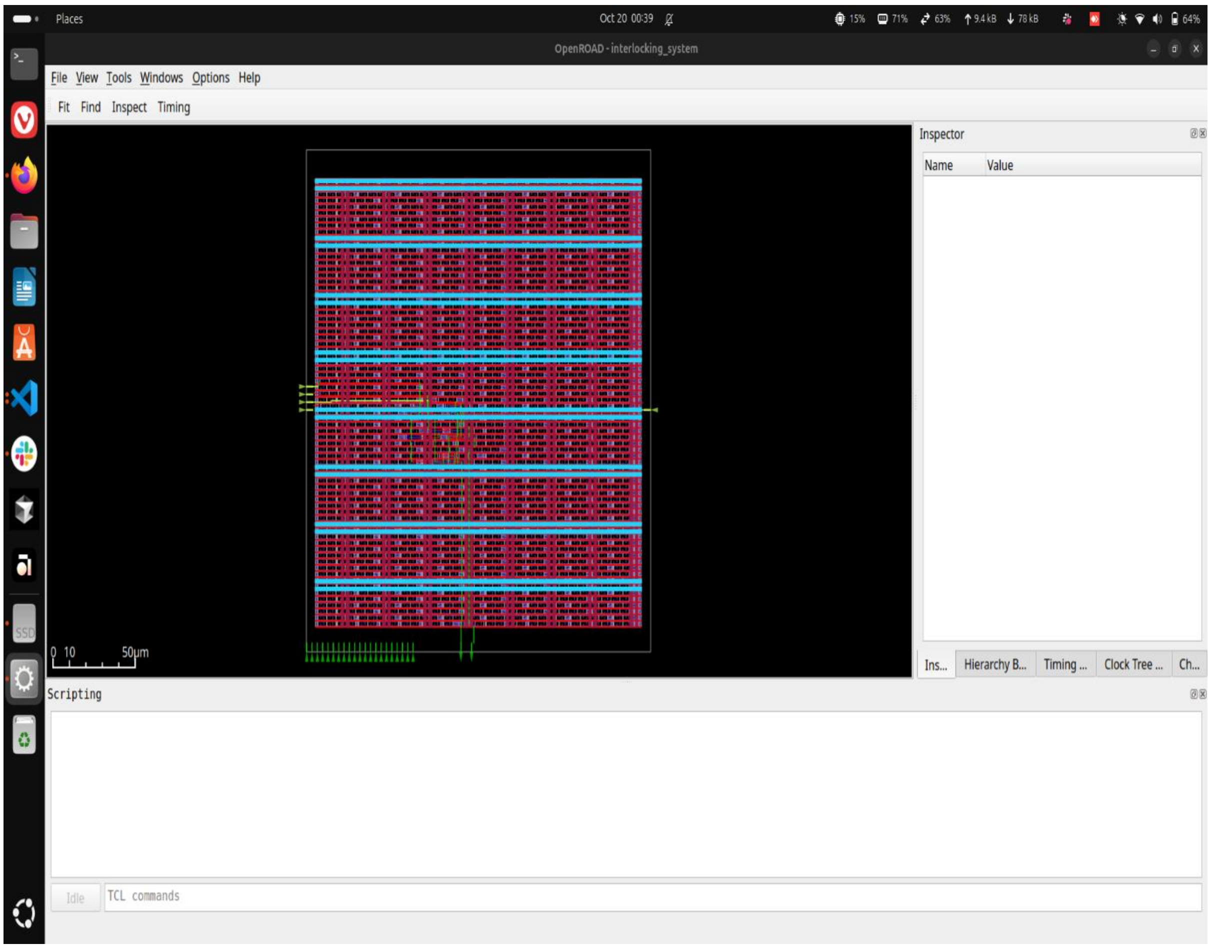
| | Source State | Destination State | Condition |
|---|---|---|---|
| 1 | CHECK_PLATFORM | ROUTE_LOCK | (!platform_occupied) |
| 2 | CHECK_PLATFORM | CHECK_PLATFORM | (platform_occupied) |

## STATE FLOW



| | Name | ROUTE_LOCK | CHECK_PLATFORM | ROUTE_REQUEST | IDLE | SWITCHING |
|---|---|---|---|---|---|---|
| 1 | IDLE | 0 | 0 | 0 | 0 | 0 |
| 2 | ROUTE_REQUEST | 0 | 0 | 1 | 1 | 0 |
| 3 | CHECK_PLATFORM | 0 | 1 | 0 | 1 | 0 |
| 4 | ROUTE_LOCK | 1 | 0 | 0 | 1 | 0 |
| 5 | SWITCHING | 0 | 0 | 0 | 1 | 1 |

GDS FILE :



**PROGRAM:**

```
`timescale 1ns / 1ps
 module tb_interlocking_system;
   reg clk;
   reg reset;
   reg train_detect;
   reg [18:0] train_number;
   reg platform_occupied;
   reg route_request;
   wire track_locked;
   wire train_switched
```

```
reg [1:0] platform_num;
  // Instantiate the DUT (Device Under Test)
  interlocking_system dut (
      .clk(clk),
      .reset(reset),
      .train_detect(train_detect),
      .route_request(route_request),
      .train_number(train_number),
      .platform_num(platform_num),
      .platform_occupied(platform_occupied),
      .track_locked(track_locked),
      .train_switched(train_switched)
  );
  // Clock generation
  initial begin
      clk = 0;
      forever #5 clk = ~clk; // 10 ns clock period
  end


  // Testbench initial block
  initial begin
      // Initialize signals
      reset = 1;
      train_detect = 0;
      train_number = 5'd0;
      platform_occupied = 0;
```

```
route_request = 0;
platform_num = 0;
// Deassert reset after 10ns
#10 reset = 0;
// Scenario 1: Train arrives at platform 2 (free)
train_detect = 1;
route_request = 1;
#10 train_number = 19'd34567;
    platform_num = 2'b10;
#15 platform_occupied = 0;
#50; // Wait for track locking and switching
// Reset inputs after processing
train_detect = 0;
route_request = 0;
#100; // Wait for FSM to return to idle
// Scenario 2: Train arrives at platform 1 (initially occupied)
train_detect = 1;
route_request = 1;
#10 train_number = 19'd38567;
    platform_num = 2'b01;
#15 platform_occupied = 1; // Platform is occupied
#50 platform_occupied = 0; // Platform becomes free after 50ns
#100; // Wait for track locking and switching
// Reset inputs
train_detect = 0;
route_request = 0;
#100;
```

```
        // Scenario 3: Train routed back to platform 2 (free)
        train_detect = 1;
        route_request = 1;
        #10 train_number = 19'd34567;
            platform_num = 2'b10;
        #15 platform_occupied = 0; // Platform is free
        #50; // Wait for track locking and switching
        // Reset inputs after processing
        train_detect = 0;
        route_request = 0;
        #100;
        // End of simulation
        #200 $finish;
    end
endmodule
```

## Applications of the Interlocking System Project:

1.Railway Safety Systems:

Ensures the safe management of train movements by controlling track switching and platform assignment, preventing collisions.

2.Automated Train Routing:

Automates the process of routing trains to the correct platform, reducing human error and improving operational efficiency.

3.Real-Time Control:

Enables dynamic and real-time track control based on train detection and platform availability, optimizing railway traffic flow.

4.Cloud Integration:

Allows route requests and updates from a centralized system, improving coordination between stations and remote operations.

5.Urban Railways and Metro Systems:

Can be applied to urban railway networks or metro systems where high-frequency train routing and platform switching are critical.

## CONCLUSION:

This project successfully demonstrates an efficient Finite State Machine (FSM) based design to automate and control train track switching and platform assignment, ensuring enhanced safety and improved management of railway traffic. By dynamically checking platform availability, locking routes, and switching trains, the system reduces risks of misrouting and collisions. It is a scalable and reliable solution that can be integrated with cloud-based management systems to handle complex rail networks in real time.