Railway Traffic Management System

FPGA Mini Project | 26-04-2023

Submitted by 2nd year - ECE A-section
Shiva Santhosh – 21ECB0F02
Levin Joji Mathews – 21ECB0F04
Tanay Bhale- 21ECB0F05
Seth Ranjan Chodagam – 21ECB0F06

Submitted to:
Dr T V K Hanumantha Rao

National Institute of Technology Warangal
2021-2025

Index

Abstract

Problem Statement

Simulating a digital circuit, specially designed for Railway Traffic Management System in Xilinx vivado suite and thereafter implementing on a FPGA Board.

Introduction

Railway Traffic Management System using FPGA is a modern and efficient way of managing railway traffic. The FPGA-based system is designed to optimize the flow of trains on a railway network, reduce delays, and improve safety.

The system includes three major components: the train controller, and the platform scheduling system.

The train controller receives train position and speed data from sensors. It calculates train distance and time to next signal or station and sends commands to track switches and signals to control train movement. It monitors train movements to detect collisions or other safety hazards and sends status updates to other modules as needed

The platform scheduling module receives train arrival signals from sensors and checks availability of platforms and assigns the first available platform to the arriving train.

Methodology

The project will be implemented using Verilog as programming language. The methodology involves defining system requirements, developing a high-level architecture, designing and implementing the modules using Verilog, integrating the modules, and testing and validating the system. This approach ensures a reliable and efficient railway traffic management system that meets the needs of the industry.

Theory

- ## What is Verilog?

Verilog is a hardware description language (HDL) used to model electronic systems. It is most commonly used in the design and verification of digital circuits at the register-transfer level of abstraction. It is also used in the verification of analog circuits and mixed-signal circuits, as well as in the design of genetic circuits. In 2009, the Verilog standard (IEEE 1364-2005) was merged into the System Verilog standard, creating IEEE Standard 1800-2009. Since then, Verilog is officially part of the System Verilog language. The current version is IEEE standard 1800-2017.

- ## What is Railway Management System?

A Railway Management System (RMS) is a computer-based application that is used to control train operations and manage platform scheduling in a railway station. It is designed to optimize train traffic by providing real-time information on train movements and scheduling, as well as resource allocation and maintenance planning.

The system consists of various components such as a train control system, a station management system, and a train scheduling system. The train control system tracks train movements and provides automatic signals to control train speeds and routing. The station management system manages passenger flow and platform scheduling to ensure efficient train operations. The train scheduling system creates schedules for trains, taking into account factors such as train routes, train capacities, and maintenance requirements.

Overall, the RMS plays a crucial role in ensuring the smooth and safe operation of a railway system by providing real-time information, optimizing train traffic, and improving passenger experience.

- ## Steps in Railway Management

Railway management system contains of two key components:

❖ Train Control

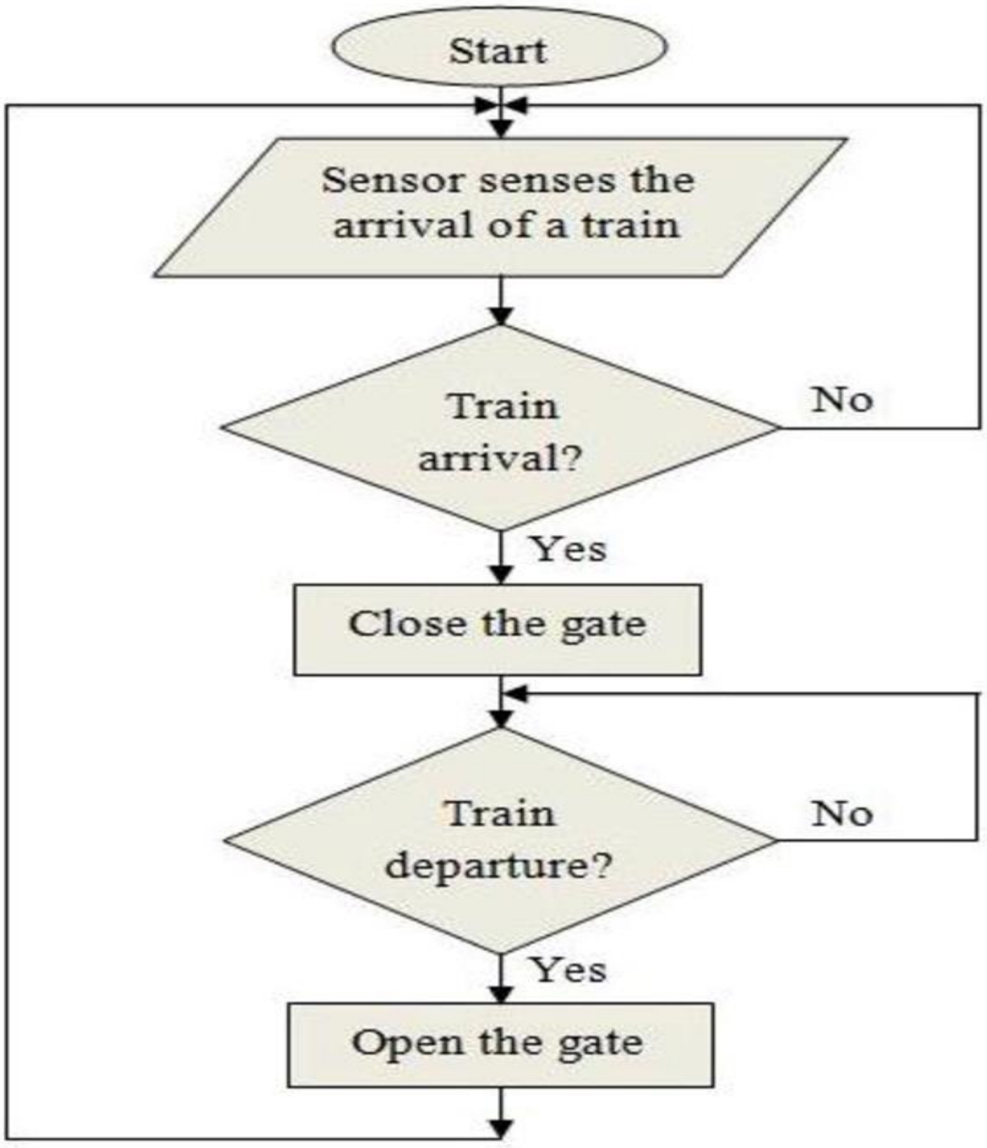
The train control module is responsible for controlling the movement of trains on the tracks. It receives train position and speed data from sensors and calculates the distance and time to the next signal or station. Based on this information, it sends commands to track switches and signals to control train movement. The train control module also monitors train movements to detect collisions or other safety hazards and sends status updates to other modules as needed.

❖ What is Platform Scheduling?

The platform scheduling module, on the other hand, is responsible for allocating platforms to arriving trains. It receives train arrival signals from sensors and checks the availability of the platforms. If a platform is available, it assigns it to the arriving train and maintains the state of the platforms using flip-flops. If both platforms are occupied, the module waits for a platform to become available before assigning it to the next train. The platform scheduling module outputs status signals indicating the status of each platform and sends status updates to other modules as needed.

The train control and platform scheduling modules work together to ensure safe and efficient train movements and platform allocations in a railway traffic management system. By coordinating the movements of trains and allocating platforms in a timely and efficient manner, these modules help to minimize delays and improve the overall performance of the railway system. Additionally, the use of sensors and other monitoring systems helps to enhance safety by detecting and mitigating potential hazards before they can cause accidents or other incidents.

- Flow Chart



- Steps to implement the project on FPGA Board

❖ Create a Vivado project: Open Vivado and create a new project by selecting "File" > "New Project". Follow the prompts to specify the project name, location, and target FPGA board.

- Add the Verilog source files: In the project manager, right-click on "Sources" and select "Add Sources". Choose the option to "Add or create design sources" and select the Verilog files for your project. Make sure that the top-level Verilog file is included in the project.

- Run synthesis: Select "Run Synthesis" from the Flow Navigator in the left panel of the Vivado window. This will synthesize the Verilog code and generate a netlist.

- Run implementation: Select "Run Implementation" from the Flow Navigator. This will map the synthesized netlist to the FPGA resources and generate a bitstream file.

- Program the FPGA: Connect the FPGA board to your computer using a USB cable or other interface cable. Make sure the board is properly powered and configured for programming. In Vivado, select "Open Hardware Manager" from the Flow Navigator. This will open the Hardware Manager window. Select "Open Target" and choose the target FPGA device. Select "Program Device" and choose the bitstream file generated in step 4. Follow the prompts to program the FPGA.

- Verify the design: Verify that the Verilog design has been correctly programmed onto the FPGA board. Test the design by sending input signals and verifying the output signals.

- Debug any issues: If the design does not work as expected, use the debugging tools in Vivado to identify and resolve any issues in the Verilog code.
- Finalize the design: Once the design is working as expected, finalize the Verilog code and the programming files. Keep these files for future use and documentation purposes.

- Pseudo-codes for Railway Management System

❖ Train Control:

- Read train position and speed data from sensors
- Calculate the distance and time to the next signal or station
- If the train is approaching a switch or signal:
- Send a command to switch the track or signal based on the train's direction
- If the train is approaching a station:
  - Check if the platform is available
  - If the platform is available, send a command to stop the train at the platform
  - If the platform is occupied, continue to the next station
- Monitor train movements to detect collisions or other safety hazards
- Send status updates to other modules as needed

❖ Platform Scheduling

- Read train arrival signals from sensors
- Check the availability of the platforms
- If a platform is available:
  - Assign it to the arriving train
  - Update the state of the platform using flip-flops
  - Send a command to the train control module to stop the train at the platform
- If both platforms are occupied:
  - Wait for a platform to become available
  - Repeat step 2
- Output status signals indicating the status of each platform
- Send status updates to other modules as needed.

Code

Train Control Module

```
module train_control(
    input clk,
    input reset,
    input [1:0] train_location,
    output [1:0] train_speed,
    output [1:0] train_stop
);

reg [1:0] current_location;
reg [1:0] next_location;
reg [1:0] speed;

always @(posedge clk or posedge reset) begin
    if(reset) begin
        current_location <= 2'b00;
        next_location <= 2'b01;
        speed <= 2'b10;
    end else begin
        if(train_location == next_location) begin
            current_location <= train_location;
            next_location <= (train_location == 2'b00) ?
2'b01 : 2'b00;
            speed <= (train_location == 2'b00) ? 2'b10 :
2'b01;
        end else begin
```

```
            current_location <= current_location;
            next_location <= next_location;
            speed <= speed;
        end
    end
end

assign train_speed = speed;
assign train_stop = (train_location == current_location) ?
1'b1 : 1'b0;

endmodule
```

This module uses a finite state machine to control the movement of the train on the track. The train_location input represents the current location of the train on the track, and the train_speed and train_stop outputs are used to control the speed and stopping of the train.

Platform Scheduling Module

```
module platform_scheduling(
    input clk,
    input reset,
    input train_arrival,
    output reg platform_1,
    output reg platform_2
);

reg [1:0] current_platform;
```

```
always @(posedge clk or posedge reset) begin
    if(reset) begin
        current_platform <= 2'b00;
        platform_1 <= 1'b0;
        platform_2 <= 1'b0;
    end else begin
        if(train_arrival) begin
            if(current_platform == 2'b00) begin
                current_platform <= 2'b01;
                platform_1 <= 1'b1;
            end else if(current_platform == 2'b01) begin
                current_platform <= 2'b10;
                platform_2 <= 1'b1;
            end else begin
                current_platform <= 2'b00;
                platform_1 <= 1'b0;
                platform_2 <= 1'b0;
            end
        end else begin
            current_platform <= current_platform;
            platform_1 <= platform_1;
            platform_2 <= platform_2;
        end
    end
end
endmodule
```

This module uses a simple state machine to manage platform scheduling. The train_arrival input represents the arrival of a train at the station, and the platform_1 and platform_2 outputs control the allocation of platforms for the trains. When a train arrives, the code determines the next available platform and sets the appropriate output to 1, indicating that the train can stop at that platform.

Test Bench:

```
module tb_traffic_management;


reg clk, reset, train_arrival;

reg [1:0] train_location;


wire [1:0] train_speed, train_stop;

wire platform_1, platform_2;


train_control train_control_inst(

    .clk(clk),

    .reset(reset),

    .train_location(train_location),

    .train_speed(train_speed),

    .train_stop(train_stop)

);


platform_scheduling platform_scheduling_inst(

    .clk(clk),

    .reset(reset),

    .train_arrival(train_arrival),
    .platform_1(platform_1),
    .platform_2(platform_2)

);

initial begin

    clk = 1'b0;
    reset = 1'b1;
    train_arrival = 1'b0;
    train_location = 2'b00;
    #10 reset = 1'b0;
end

always #5 clk = ~clk;

always @(posedge clk) begin

    train_location <= (train_location == 2'b00) ? 2'b01 :
2'b00;
end

initial begin



    $monitor("Time = %d | Train Location = %b | Train
Speed = %b | Train Stop = %b | Platform 1 = %b | Platform
```

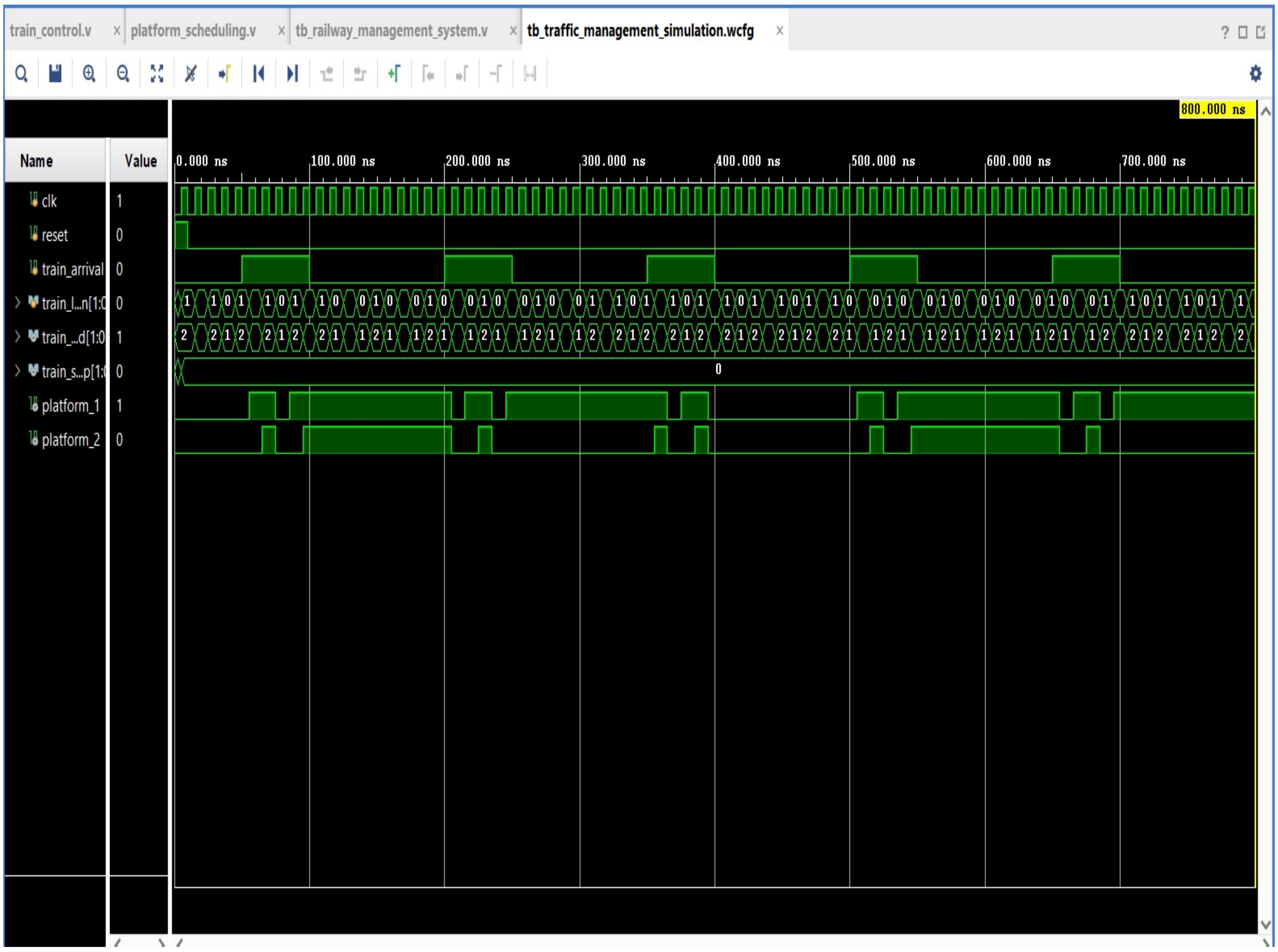
```
2 = %b", $time, train_location, train_speed, train_stop,
platform_1, platform_2);

    #50 train_arrival = 1'b1;

    #50 train_arrival = 1'b0;

    #100 train_arrival = 1'b1;

    #50 train_arrival = 1'b0;

    #100 train_arrival = 1'b1;

    #50 train_arrival = 1'b0;

    #100 train_arrival = 1'b1;

    #50 train_arrival = 1'b0;

    #100 train_arrival = 1'b1;

    #50 train_arrival = 1'b0;

    #100 $finish;

end


endmodule
```

This testbench code instantiates both the train_control and platform_scheduling modules and connects them to the testbench signals. The initial and always blocks are used to generate clock signals and simulate the movement of the train on the track. The train_arrival signal is used to simulate the arrival of trains at the station.

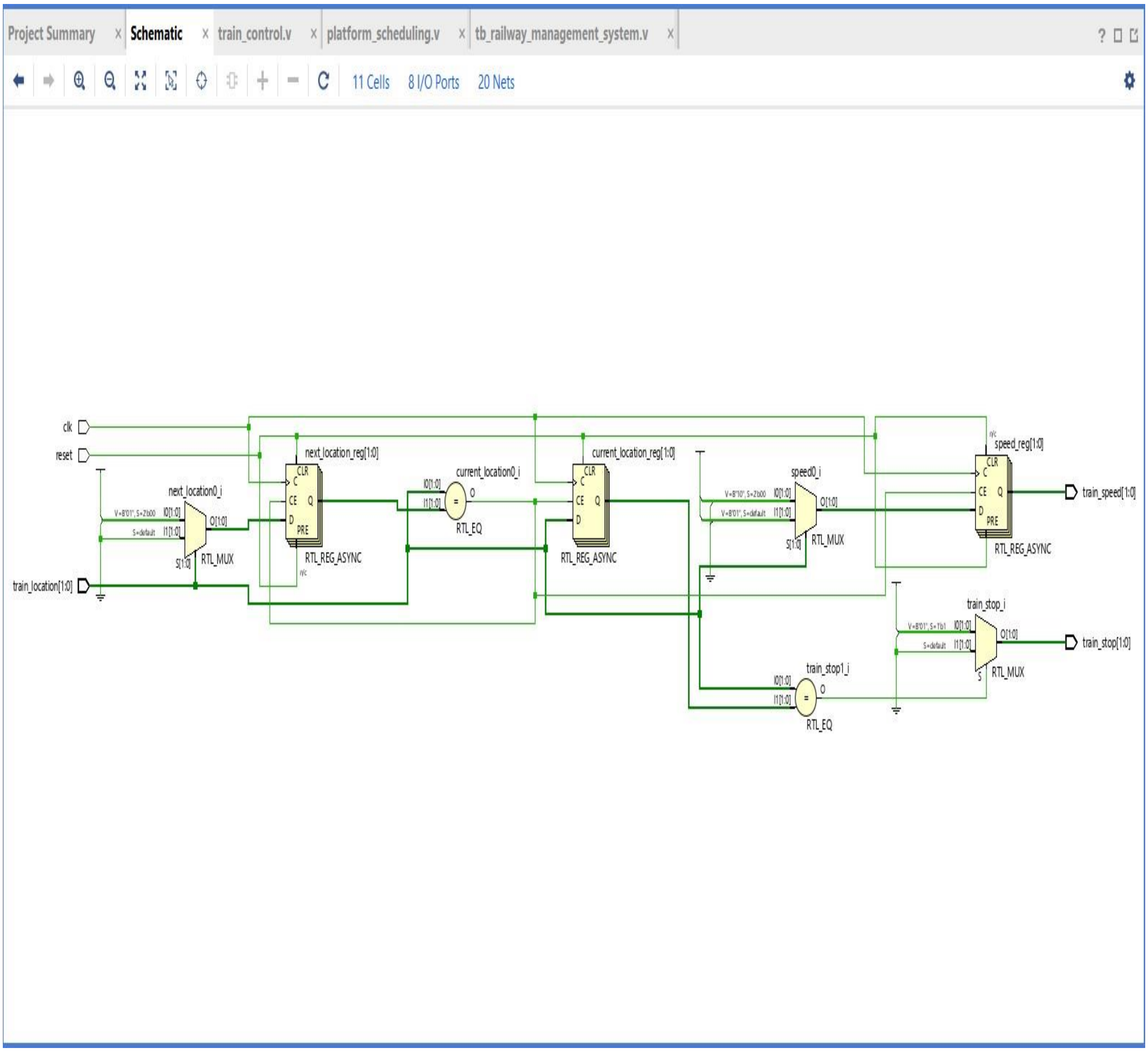
The $monitor statement is used to print the values of the signals at each clock cycle. This allows us to see the train location, speed, and stop signals, as well as the platform allocation for each train.

Results

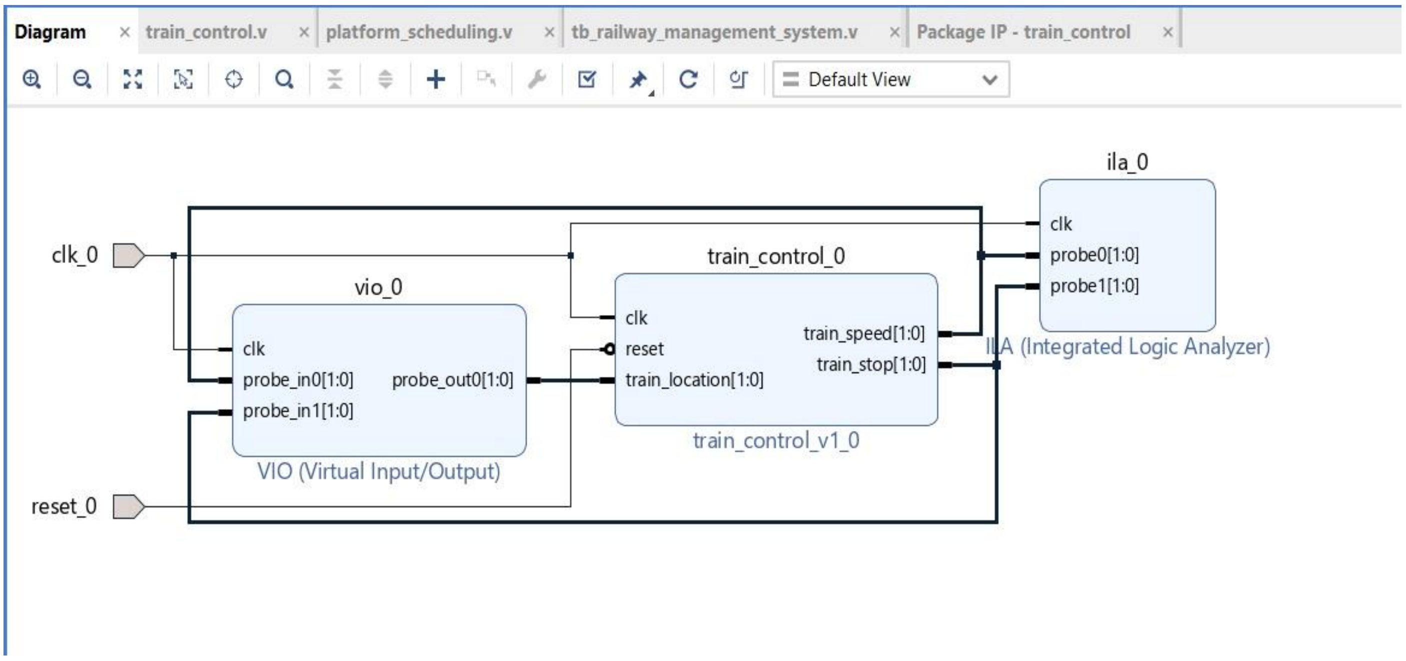
❖ Simulation



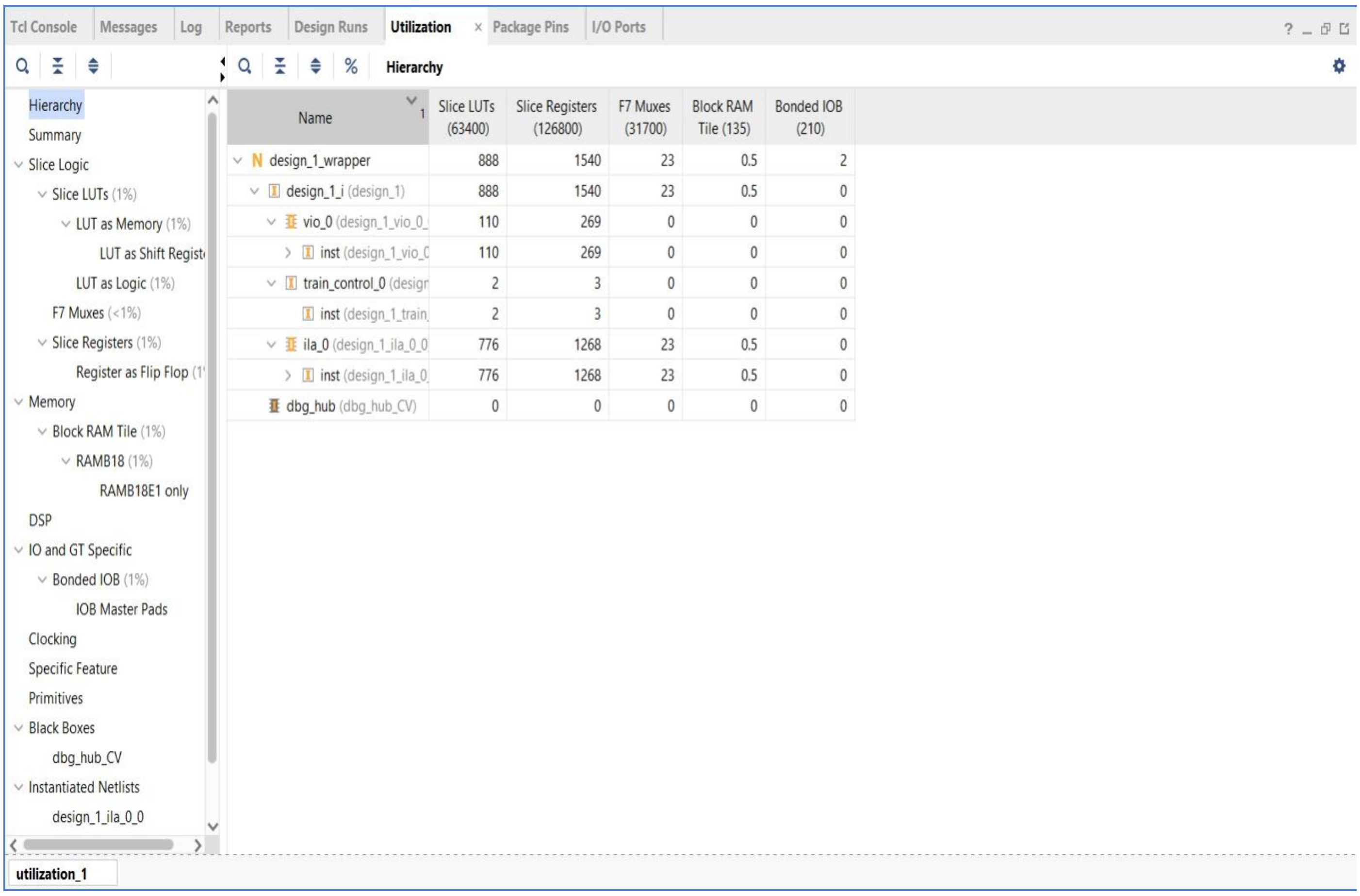
❖ Elaborated Design



❖ Block Design



❖ Utilization Report



Name	Slice LUTs (63400)	Slice Registers (126800)	F7 Muxes (31700)	Block RAM Tile (135)	Bonded IOB (210)
design_1_wrapper	888	1540	23	0.5	2
design_1_i (design_1)	888	1540	23	0.5	0
vio_0 (design_1_vio_0_	110	269	0	0	0
inst (design_1_vio_C	110	269	0	0	0
train_control_0 (design	2	3	0	0	0
inst (design_1_train_	2	3	0	0	0
ila_0 (design_1_ila_0_0	776	1268	23	0.5	0
inst (design_1_ila_0_	776	1268	23	0.5	0
dbg_hub (dbg_hub_CV)	0	0	0	0	0

How to enhance this project for future opportunities?

- ❖ There are several ways to for future opportunities in railway traffic management:

  - Implement artificial intelligence (AI) and machine learning algorithms to optimize train movements and platform allocations based on real-time data. This can help to improve efficiency, reduce delays, and enhance safety.
  - Use advanced sensors and monitoring systems, such as LiDAR and radar, to provide more accurate and detailed information about train and platform positions, speeds, and other parameters. This can help to improve the precision and reliability of the train control and platform scheduling modules.
  - Implement predictive maintenance techniques to detect and address potential equipment failures before they cause delays or other issues. This can help to minimize downtime and improve the overall performance of the railway system.
  - Incorporate wireless communication technologies, such as 5G and Wi-Fi 6, to provide faster and more reliable data transmission between modules and sensors. This can help to improve the responsiveness and efficiency of the train control and platform scheduling modules.
  - Integrate the train control and platform scheduling modules with other railway management systems, such as ticketing and scheduling systems, to provide a more seamless and integrated experience for passengers and operators.

How can the code developed be applied to other real-life scenarios?

- **Automated Warehouse Management System**: The train control and platform scheduling modules can be modified and applied to an automated warehouse management system to manage the movement of goods between storage locations and loading docks. The modules can help to optimize the movement of goods, reduce waiting times, and enhance safety.
- **Airport Traffic Management System**: The train control and platform scheduling modules can be modified and applied to an airport traffic management system to manage the movement of aircraft on runways and at gates. The modules can help to optimize aircraft movements, reduce waiting times, and enhance safety.
- **Autonomous Vehicle Traffic Management System**: The train control and platform scheduling modules can be modified and applied to an autonomous vehicle traffic management system to manage the movement of self-driving cars and other autonomous vehicles on roads and highways. The modules can help to optimize vehicle movements, reduce congestion, and enhance safety.
- **Seaport Traffic Management System**: The train control and platform scheduling modules can be modified and applied to a seaport traffic management system to manage the movement of ships and cargo. The modules can help to optimize ship movements, reduce waiting times, and enhance safety.

Conclusion

In conclusion, the train control and platform scheduling modules play critical roles in the efficient and safe management of railway traffic. These modules work together to control train movements and allocate platforms, minimizing delays and enhancing safety. By incorporating advanced technologies and techniques, such as AI, predictive maintenance, and wireless communication, these modules can be further enhanced to optimize train movements, improve precision and reliability, and provide a more seamless and integrated experience for passengers and operators. Overall, the train control and platform scheduling modules are essential components of a modern railway traffic management system, helping to ensure the safe and efficient transportation of people and goods. The railway traffic management system is a critical component of modern transportation infrastructure, and its continued development and improvement are essential for the sustainable growth of the global economy.

References

- "Railway Traffic Control and Management" by Xiaobo Qu and Wenhui Li, Springer, 2021.
- "Advanced Train Control Systems" by Paul A. Smith and Alistair P. Cook, The Institution of Engineering and Technology, 2014.
- The Institution of Railway Signal Engineers (IRSE): https://www.irse.org/
- The International Association of Public Transport (UITP): https://www.uitp.org/
- The Rail Safety and Standards Board (RSSB): https://www.rssb.co.uk/