SAVEETHA ENGINEERING COLLEGE AUTONOMOUS
AFFILIATED TO ANNA UNIVERSITY
TNEA CODE 1216

Skill Assessment

Subject Code	Subject Name
19EC526	Verilog HDL

Name of the Faculty	**Dr. Arunkumar K**
Name of the Student	**Royce Niran George A**
Register Number	**212223060231**
Department	**ECE**
Slot Name	**4X1-26**
Submitted Date	
Marks Obtained	

1. Problem Statement

1.1 Real-World Context

Railway level crossings are critical safety points where road vehicles and trains intersect. Manual gate operation often leads to delays, human errors, and unsafe conditions. Automatic railway gate systems help eliminate these risks by using sensors and control logic to manage the gate movement in real time. This improves safety, reduces accidents, and ensures smooth traffic flow without depending on human operators.

1.2 Project Objective

The objective of this project is to design and implement an automatic railway gate control system using Verilog HDL. The system should detect the presence of an approaching train, close the gate before the train reaches the crossing, keep the gate closed during the train's passage, and reopen the gate once the train exits. Additionally, visual indicators (red/green lights) must guide road traffic, ensuring safe and systematic operation.

1.3 Key Modules in the Design

1. Sensor Simulation (Entry/Exit):
   Simulates the proximity sensors placed on the track to detect when a train arrives or leaves the crossing area.

2. FSM Controller:
   A finite-state machine (FSM) processes sensor inputs and controls gate actions (open/close), signal lights, and motor activation.

3. Signal Lights (Red/Green):
   Visual indicators for road users.

   - Green: No train, Safe to cross
   - Red: Train coming or train crossing in-progress

1.4 Relevant Verilog Concepts Used

- FSM Design:
  Implemented using state registers, next-state logic, and output logic.

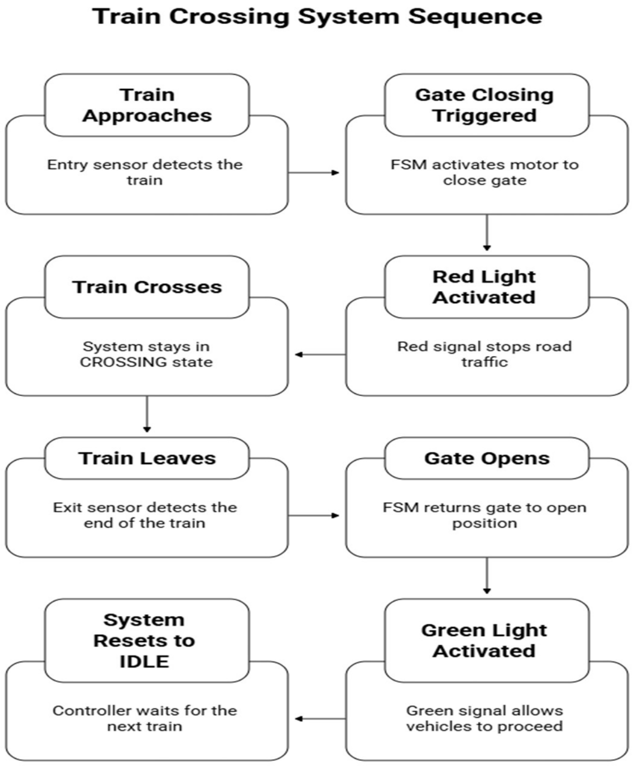
- Sequential Logic:
  Flip-flops used for state transitions on clock edge.

- Timing Control:
  Clock-based refresh for 7-segment display and optional delays.

- Combinational Logic:
  Used for decoding BCD values and controlling signals.

2. Design and Methodology

2.1 Block Diagram


Train Crossing System Sequence
Train Approaches
Entry sensor detects the train
Gate Closing Triggered
FSM activates motor to close gate
Red Light Activated
Red signal stops road traffic
Train Crosses
System stays in CROSSING state
Train Leaves
Exit sensor detects the end of the train
Gate Opens
FSM returns gate to open position
Green Light Activated
Green signal allows vehicles to proceed
System Resets to IDLE
Controller waits for the next train

2.2 Functional Description

>IDLE

No train is detected, so the gate stays open, and the green light is ON. The system simply waits for the entry sensor to trigger.

>INCOMING

The entry sensor detects an approaching train, so the red light turns ON. The motor starts closing the gate to secure the crossing.

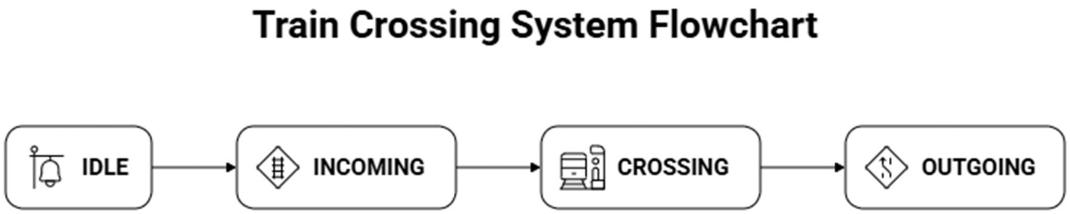
>CROSSING

The gate is fully closed while the train passes across the track. The system stays in this state until the exit sensor activates.

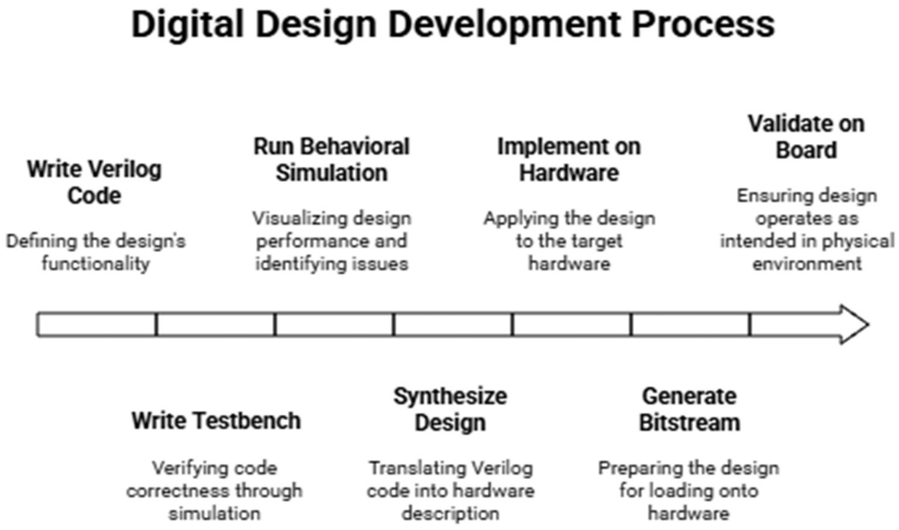
>OUTGOING

The exit sensor indicates the train has cleared the crossing. The motor opens the gate, and the system returns to IDLE.

2.3 FSM State Diagram



2.4 Design Flow


Digital Design Development Process
Write Verilog Code
Defining the design's functionality
Write Testbench
Verifying code correctness through simulation
Run Behavioral Simulation
Visualizing design performance and identifying issues
Synthesize Design
Translating Verilog code into hardware description
Implement on Hardware
Applying the design to the target hardware
Generate Bitstream
Preparing the design for loading onto hardware
Validate on Board
Ensuring design operates as intended in physical environment

3. Verilog Coding & Implementation

Inputs

- clk: Provides the timing signal for state transitions and display refresh.(clock)
- rst: Resets the system back to the IDLE state. (button)
- entry: Indicates that a train is approaching the crossing. (switch)
- exit: Indicates that the train has left the crossing. (switch)

Outputs

- red: Turns ON to stop vehicle movement during gate closure. (RGB led)
- green: Turns ON to allow vehicles when the gate is open. (RGB led)
- an: Selects which 7-segment display digit is active. (7-segment display)
- seg: Drives the segments to display graphical representation (7-segment display)

3.1 RTL Code

train_signal.v

```
`timescale 1ns / 1ps

module train_signal(clk, rst, entry, exit, red, green, an, seg);

input clk, rst, entry, exit;

output reg red, green;

output reg [3:0] an;

output reg [6:0] seg;

reg [2:0] current, next;

reg [12:0] motor;

parameter IDLE     = 3'b000,

INCOMING  = 3'b001,

CROSSING  = 3'b010,

OUTGOING  = 3'b011;

always @(posedge clk or posedge rst) begin

if (rst)

        current <= IDLE;

else

        current <= next;

end

always @(*) begin

        next = current;

        case (current)

                IDLE:    if (entry) next = INCOMING;

                INCOMING: next = CROSSING;

                CROSSING: if (exit) next = OUTGOING;

                OUTGOING: next = IDLE;

        endcase
```

```
end

always @(*) begin

        red   = 0;

        green = 0;

        motor = 12'd0;

        case (current)

                IDLE:     begin green = 1; motor = 12'd120; end

                INCOMING: begin red = 1;  motor = 12'd0; end

                CROSSING: begin red = 1;  motor = 12'd0; end

                OUTGOING: begin red = 1;  motor = 12'd120; end

        endcase

end

reg [15:0] refresh_cnt = 0;

reg refresh_clk = 0;

reg [1:0] digit = 0;

reg [3:0] bcd;

always @(posedge clk) begin

        refresh_cnt <= refresh_cnt + 1;

        refresh_clk <= refresh_cnt[15];

end

always @(posedge refresh_clk) begin

        digit <= digit + 1;

end

always @(*) begin

        case(digit)

                2'b00: begin an = 4'b1110; bcd = motor % 10; end

                2'b01: begin an = 4'b1101; bcd = (motor / 10) % 10; end

                2'b10: begin an = 4'b1011; bcd = (motor / 100) % 10; end
        2'b11: begin an = 4'b0111; bcd = (motor / 1000) % 10; end
    endcase
end
always @(*) begin
    case(bcd)
        4'd0: seg = 7'b0111111;
        4'd1: seg = 7'b1001111;
        4'd2: seg = 7'b1111001;
        default: seg = 7'b1111111;
    endcase
end
endmodule
```

3.2 Testbench Code

train_signal_tb.v

```
module train_signal_tb;
reg clk, rst, entry, exit;
wire red, green, an, seg;
train_signal uut(clk, rst, entry, exit, red, green, an, seg);
initial begin
    clk = 0;
    forever #5 clk = ~clk;
end
initial begin
    rst = 1; entry = 0; exit = 0;
    #20 rst = 0;
    entry = 1;
    #10 entry = 0;
```

```
        #50 exit = 1;

        #10 exit = 0;

        #50 $finish;

end

endmodule
```

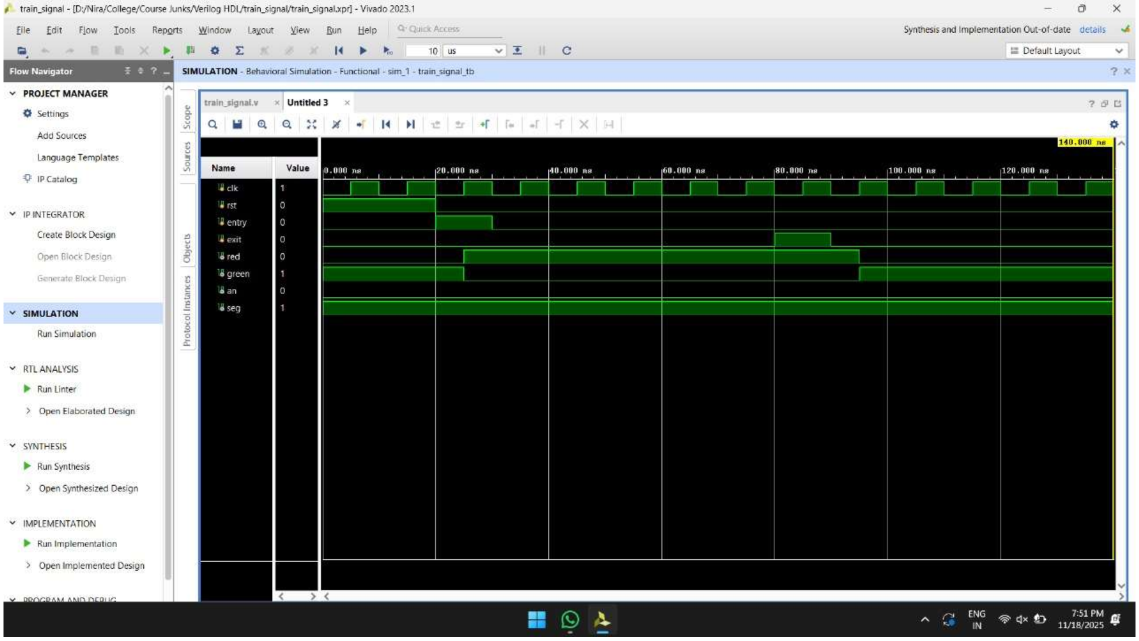
3.3 Constraint File

train_signal.xdc

```
set_property -dict {PACKAGE_PIN J2 IOSTANDARD LVCMOS33} [get_ports {rst}]

set_property -dict {PACKAGE_PIN V2 IOSTANDARD LVCMOS33} [get_ports {entry}]

set_property -dict {PACKAGE_PIN U2 IOSTANDARD LVCMOS33} [get_ports {exit}]

set_property -dict {PACKAGE_PIN V6 IOSTANDARD LVCMOS33} [get_ports {red}]

set_property -dict {PACKAGE_PIN V3 IOSTANDARD LVCMOS33} [get_ports {green}];

set_property -dict {PACKAGE_PIN F14 IOSTANDARD LVCMOS33} [get_ports {clk}]

set_property -dict {PACKAGE_PIN D5 IOSTANDARD LVCMOS33} [get_ports {an[0]}]

set_property -dict {PACKAGE_PIN C4 IOSTANDARD LVCMOS33} [get_ports {an[1]}]

set_property -dict {PACKAGE_PIN C7 IOSTANDARD LVCMOS33} [get_ports {an[2]}]

set_property -dict {PACKAGE_PIN A8 IOSTANDARD LVCMOS33} [get_ports {an[3]}]

set_property -dict {PACKAGE_PIN D7 IOSTANDARD LVCMOS33} [get_ports {seg[0]}]

set_property -dict {PACKAGE_PIN C5 IOSTANDARD LVCMOS33} [get_ports {seg[1]}]

set_property -dict {PACKAGE_PIN A5 IOSTANDARD LVCMOS33} [get_ports {seg[2]}]

set_property -dict {PACKAGE_PIN B7 IOSTANDARD LVCMOS33} [get_ports {seg[3]}]

set_property -dict {PACKAGE_PIN A7 IOSTANDARD LVCMOS33} [get_ports {seg[4]}]

set_property -dict {PACKAGE_PIN D6 IOSTANDARD LVCMOS33} [get_ports {seg[5]}]

set_property -dict {PACKAGE_PIN B5 IOSTANDARD LVCMOS33} [get_ports {seg[6]}]

set_property -dict {PACKAGE_PIN A6 IOSTANDARD LVCMOS33} [get_ports {seg[7]}]
```

4. Output and Result

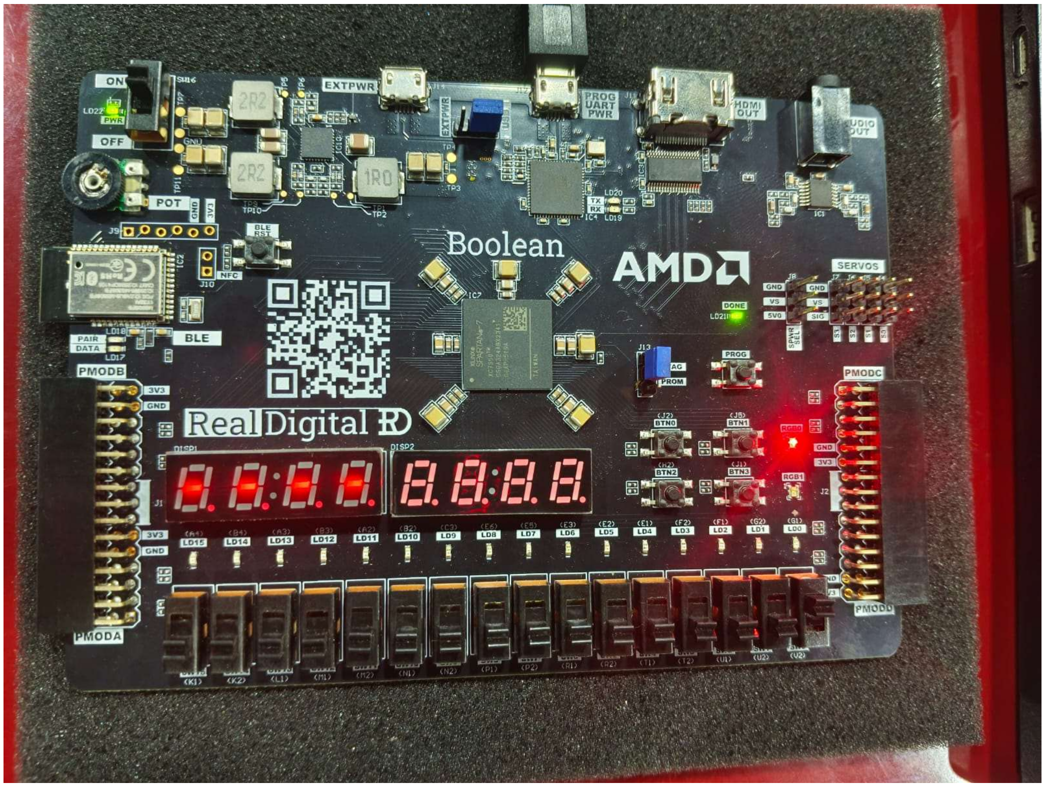
4.1. Testbench Output


train_signal - [D:/Nira/College/Course Junks/Verilog HDL/train_signal/train_signal.xpr] - Vivado 2023.1

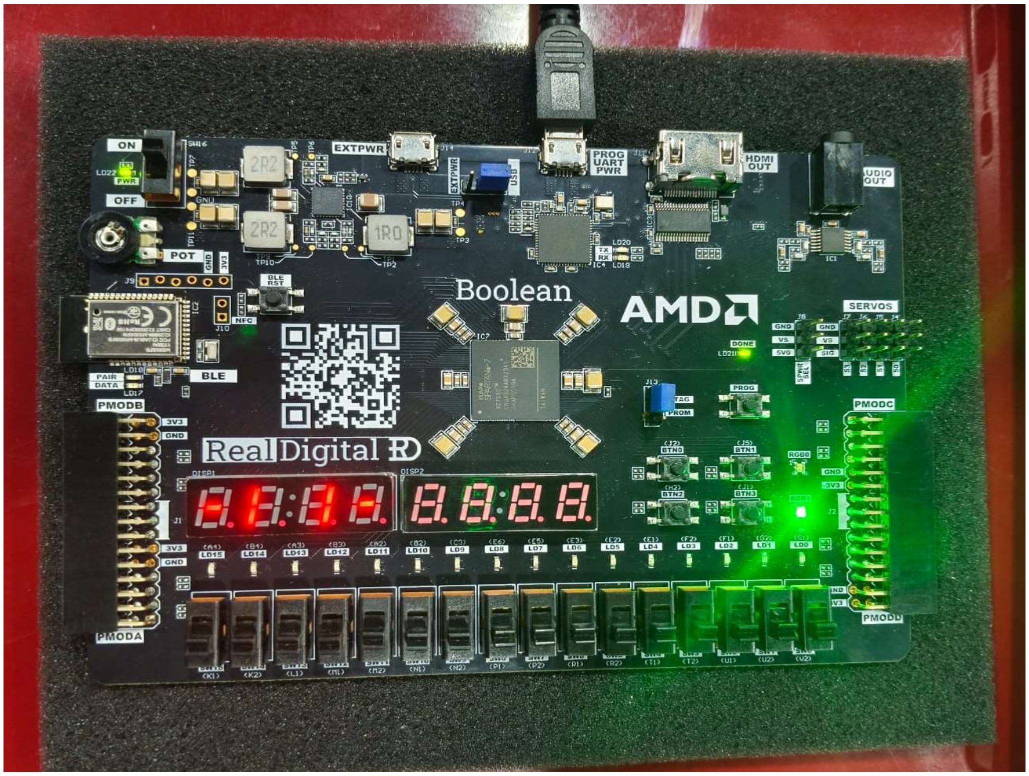
SIMULATION - Behavioral Simulation - Functional - sim_1 - train_signal_tb

Name	Value
clk	1
rst	0
entry	0
exit	0
red	0
green	1
an	0
seg	1

4.2. Spartan-7 Boolean FPGA Implementation Output



a) When Train Approaches



b) When Train Passes

4.3. Result

The Verilog design for automatic railway gate control was successfully implemented and verified using Vivado. FSM-based control ensured reliable operation, and the 7-segment display accurately reflected motor activity.