**VLSI PROJECT REPORT**

***In partial fulfilment of the requirements for the award of the degree of***

**Bachelor of Technology**

**in**

**Electronics and Communication Engineering**

IV semester

*Submitted by*

**Student Name: Ankush Verma**

**Register Number: 200907058**



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

MANIPAL INSTITUTE OF TECHNOLOGY

(A Constituent institute of MAHE, Manipal)

MANIPAL – 576104, KARNATAKA, INDIA

**PROJECT: VERILOG BASED TRAIN COLLISION AVOIDANCE- SYSTEM**

**1. INTRODUCTION:**

Electronic devices play an important role in my daily life. Each of us in one or other way is related directly or indirectly to electronics. Electronics deals with electric circuits and electronic devices. Electronic devices are components that control the flow of electric current through them for information processing thus can be used in a wide variety of applications that help me in day-to-day life. Today electronics has gained much more importance due to its variety of applications in various industries. These electronic devices mostly perform functions like control, amplification, and rectification . In the last few years, electronics has witnessed a great revolution in the industry than any other technology in invention. The current trending technologies include Verilog, VHDL, Mems, and Nems. As I know, most of the people travel by trains as it is the easiest and cheapest way of transport. Annually 12 million people travel by train all over the globe, but sadly, nearly 10% of people meet with train accidents leading to severe injuries and fatal deaths .

Every hour on an average, at least 8-10 persons die in train accidents, and at least 2-3 million people are affected seriously, which was shown from the recent survey. I also estimate that the medical bill, damaged property due to accidents, and other nearly cost up to 2-3 percent of the world's gross domestic product. Train accidents may occur due to cracks on the railway track or due to short circuiting or ineffective Train Traffic Control System that is failing to find the train approaching in the opposite direction on the same track leading to collision. Recently, most of the train accidents occur due to train collision .

Due to these accidents, this global world is facing many fatal deaths . Here I have implemented one of the best and easy solutions to avoid such collisions. Earlier there are many implementations like anti-collision train system, etc., which is not effective all the time, and maintaining cost is high. And on-road train detection is one of the recent implementations which is not up to mark as it is complex. And one of the finest implementations is railway signalling using wireless communications which is efficient and effective and works fine. But in my project, I have decided to find an effective solution to avoid the collisions. In my project, I am implementing the train collision avoidance system using Verilog programming language. It is a Hardware Description Language (HDL) which is used for modelling electronic systems. It is like other programming languages and mostly like C programming language, so it is easy to learn for beginners. To implement this program, I use software called XILINX ISE (Integrated Synthesis Environment). It is a software tool developed by Xilinx for synthesizing and analysing HDL designs. The software is also used to synthesize the designs, performance timing analysis, examining the RTL designs, and configuring the target device with the programmer. It is primarily used for circuit synthesis and its design used for System-level testing.

Generally, when a single train is moving, there is no point of collision. The problem arises when two trains approaching towards each other on a single track. In my project, I worked on identifying the status of the opposite train and processing only one train based upon their importance or priority. When two trains come on the same track, there comes the chance of a collision. To avoid these accidents, we allow only one train based upon their priority.

Here in this project, I had considered 4 types of trains on each side, i.e., Right side and left side. The trains are categorized as left superfast, left express, left passenger, left goods, right superfast, right express, right passenger, right goods. And priority is given to a train as per below.

*1.1Priority Table*

Let's consider different types of trains denoted as Superfast (S), Express (E), Passenger (P), and Goods (G). The priorities assigned to each type of train in various combinations are presented below:

- S>E: Superfast is given priority over Express.

- S>P: Superfast is given priority over Passenger.

- S>G: Superfast is given priority over Goods.

- E>P: Express is given priority over Passenger.

- E>G: Express is given priority over Goods.

- P>G: Passenger is given priority over Goods.

In cases where the trains on both sides are of the same type (e.g., S==S, E==E, P==P, G==G), the priority is granted to the left train.

Let's consider the following two scenarios:

Case 1: When I have two trains with equal priorities, as shown in Fig. 1, the left train is "goods," and the right train is also "goods," approaching each other. In this situation, I need to determine which train is given priority. Referring to the priority table, when two trains have equal rankings, the left train is granted the highest priority, allowing it to move while the right train is stopped by a red signal.

Case 2: In this scenario presented in Fig. 2, the left train is categorized as "passenger," and the right train is identified as "superfast." To prevent a collision, one train is given priority over the other. As per priority table 1, "superfast" holds a higher priority than "passenger." Consequently, the right train (superfast) is allowed to proceed while the other train (passenger) is stopped.

**2. THEORETICAL ANALYSIS**

To prevent train collisions, I have implemented a priority-based diversion system. The complete process is detailed through a flow chart, which outlines the steps taken to ensure train safety. The primary challenge arises when two trains share the same track, potentially leading to collisions. To address this, sensors have been strategically placed on both sides of the track to detect the presence and type of trains moving in either direction. The sensor data is then forwarded to the next stage, where the priority table and hdl (hardware description language) code come into play. Based on the priority assigned to each train type, I decide which train should be granted priority and subsequently processed.

The results of this project demonstrate significant improvements in the reliability of railway transport safety. The trains considered in my implementation are Goods, Superfast, Express, and Passenger, as previously discussed. Among these train types, Superfast is given the highest preference, while Goods is assigned the least priority. However, the priority settings can be modified as required, allowing for flexibility in the system.

The flow of the process begins by checking whether any trains are approaching. If no trains are detected, the stop gates remain open, and I continually monitor for incoming trains, as they may arrive at any moment. The analysis commences with the status of the left train. If the left train is approaching from the right direction (status equals one), I then examine the right train's status. If the right train's status is also one (approaching from the left), I allow the specific train to proceed based on the assigned priority. However, if the right train's status is not one, both gates remain open since the presence of a single train poses no problem, and I continue to monitor the right train's status. Conversely, if the left train's status is not equal to one, both gates are opened, and I continuously check the status of the left train since it may arrive at any given moment. The process comes to a halt once the priority decision is made, and a new cycle begins for subsequent trains.

*2.1 Algorithm*

Step 1: Begin the process.

Step 2: Check the status of the left train.

Step 3: If the left train status is equal to 1, proceed to step 5.

Step 4: Otherwise, open the gates and return to step 2.

Step 5: Check the status of the right train.

Step 6: If the right train status is equal to 1, open the gates based on priority.

Step 7: Otherwise, open the gates and return to step 5.

Step 8: End the process.

*2.2 Theoretical Explanation*

In Table below, binary values are used to indicate the presence of trains. When the value is "0000," it denotes that no train is approaching, while other binary values like "0001," "0100," and "1000" signify the arrival of a train of a specific type.

| Left Side Trains | Right Side Trains | Left Side Status | Right Side Status | Status Stop Gate |

|---------------------|-----------------------|--------------------|-----------------------|---------------------|

| 0000 | 0001 | 0 | 1 | 1 |1

| 0100 | 0000 | 1 | 0 | 1 |1

| 0010 | 0001 | 1 | 0 | 1 |1

| 1000 | 1000 | 1 | 0 | 1 |1

Case 1: In this scenario, there is a train on the right side, denoted as "0001," indicating a superfast train. Since there is no train on the left side ("0000"), the gates remain open for both trains, following the specified algorithm.

Case 2: Here, a train is present on the left side, represented as "0100," indicating a passenger train. The algorithm checks the right side, where there is no train ("0000"), resulting in open gates for both trains.

Case 3: When trains are present on both sides, for instance, an express train on the left side and a goods train on the right side, priority is given to the train on the left side, as per Table.

Case 4: When there are trains with equal priority on both sides, the assumption is made that the left train is given the highest priority, following Table.

**3. FLOW CHART OF TRAIN COLLISION AVOIDANCE SYSTEM**

NO

NO

YES

YES

GATES ARE OPEN BASED ON PRIORITY

RIGHT TRAIN STATUS==1

GATES ARE OPEN FOR BOTH TRAINS

GATES ARE OPEN FOR BOTH TRAINS

SCAN FOR RIGHT TRAIN STATUS

LEFT TRAIN STATUS==1

SCAN FOR LEFT TRAIN STATUS

**4. CONCLUSION**

In this project, I successfully implemented the Train Collision Avoidance System using Verilog. When two trains appear on the same track, they collide, leading to severe accidents. To avoid train collisions, I have developed an efficient algorithm that is cost-effective and can be easily implemented in no time. Through proper implementation of this algorithm, many human lives can be saved, and a lot of property can be protected from being damaged.

This project's key advantage is its ability to work under any circumstances, as it is based on code and doesn't require significant human labor. The trains will deviate automatically according to the priority given, eliminating the need for constant human intervention. I have clearly explained the functionality with a few examples and verified the results through respective outputs and simulations.

The outcomes demonstrated in this project significantly enhance the reliability of safety in railway transportation. As a result, we can expect further developments in the railway system soon, which will contribute to a great push in the economy.

A screen shot of a computer

Description automatically generated

**Fig. 4.1:**  Gate stop open indicating that no two trains are arriving.

It indicates that no train is arriving on either side of the track so that gate stop should be opened and is indicated by low (0) signal.

A screenshot of a computer screen

Description automatically generated

**Fig. 4.2:** Gate stop close as right train is having priority.

It indicates that right passenger and left goods are arriving. As passenger is having high priority from Table , priority is given for the right-side train and gate stop should be closed for left train and is indicated by high (1) signal.