New York Institute of Technology

Cross-Traffic Sensor System: Design and Simulation

AUTHOR

Siddharth Ketan Kenia

COURSE NAME AND NUMBER

Computer Architecture, CSCI-155

SEMESTER

FALL 2024

Contents

1 Abstract

This thesis describes the design and implementation of a cross-traffic sensor system in VHDL, which is an important component for improving safety in Advanced Driver Assistance Systems (ADAS). The system uses an infrared sensor to detect vehicles or obstacles approaching from the sides, calculates the distance, and sends a brake signal when a collision is imminent. A high-level block diagram depicts the system's architecture, which includes modules for clock generation, distance calculation, sensor data analysis, and output monitoring. The design is detailed, including sensor functionality, distance calculation via time-of-flight measurement, and braking system activation logic based on a predefined safety threshold. The VHDL code for the core sensor module is provided and fully explained. A detailed testbench is created and used in ModelSim simulations to validate the system's functionality in a variety of scenarios. Simulation results confirm accurate distance measurement and consistent brake signal generation within the safety parameters. The discussion section examines the simulation results, emphasizing the system's strengths, limitations, and potential for future improvements, such as incorporating real-world environmental factors and advanced algorithms to improve performance. The study concludes by emphasizing the significance of this work in advancing the larger goals of improved road safety and intelligent transportation systems.

2 Introduction

Transportation systems have emerged as a fundamental pillar of global economic growth and development. However, many cities face significant challenges due to uncontrolled growth of traffic volume, resulting in severe problems such as traffic congestion, delays, increased fuel consumption, higher CO2 emissions, accidents and deteriorated quality of life. Advancements in information and communication Technologies (ICT) have paved a way for Intelligent Transportation Systems (ITS). These systems focus on sustainability, integration, safety and responsiveness to improve mobility, reduce environmental impacts, and enhance economic efficiency [1]. The integration of sensor technologies within ITS plays a pivotal role in collecting and processing accurate data for real-time decisions, enabling vehicle-to-vehicle (V2V) and Vehicle-to-infrastructure (V2I) communication. This data is used to reduce CO2 emissions and traffic congestion while improving road safety. [2] [3]

This report discusses the design and implementation of a cross traffic sensor system in VHDL (VHSIC Hardware Description Language), demonstrating the importance of sensor technology in ITS. The proposed system uses an infrared sensor to emit rays, detect human presence and calculate the time of flight (TOF) of signal reflection. Using this TOF, the system computes the distance between the vehicle and the detected object using the standard distance formula. This information is then analyzed to determine whether the car's braking system should be activated to prevent a collision.

Modern vehicles already use a variety of sensors, including proximity, ultrasonic, radar and LIDAR sensors to improve functionality. Proximity and ultrasonic sensors aid in parking assistance and obstacle detection, while radar and LIDAR provide high accurate depth information for collision avoidance and autonomous driving [4]. Despite these advances, certain limitations remain, such as susceptibility to environmental factors like temperature and humidity, as well as integration challenges caused by a lack of standardization among sensor technologies. Low-visibility conditions can limit the performance of systems like Volvo's city safety, which use laser sensors. [5]

The cross traffic sensor addresses these challenges by combining sensor technology with an efficient and scalable designs methodology. Using VHDL improves hardware efficiency and allows robust simulation and testing real-world scenarios. This project aligns with the ITS goals of sustainability, safety, and responsiveness, while also contributing to the larger effort of improving transportation systems.

This report explores the use of sensor technology in collision prevention, highlighting the potential for such systems to create safer and more intelligent vehicles. This work's findings will also help to advance ITS by addressing critical issues like traffic congestion, fuel efficiency and road safety.

3 High Level Block Diagram

Cross traffic sensors in modern automotive systems play an important role in increasing driver safety by detecting vehicles approaching from the sides. This test bench setup simulates and evaluates the functionality of a cross traffic sensor system, ensuring that it detects across traffic correctly and signals the braking system.

The system architecture is made up of various components that work together to simulate and monitor the cross-traffic detection. Each component is critical to determining the sensor's performance and reliability. Here's a high level block diagram of cross traffic sensor.

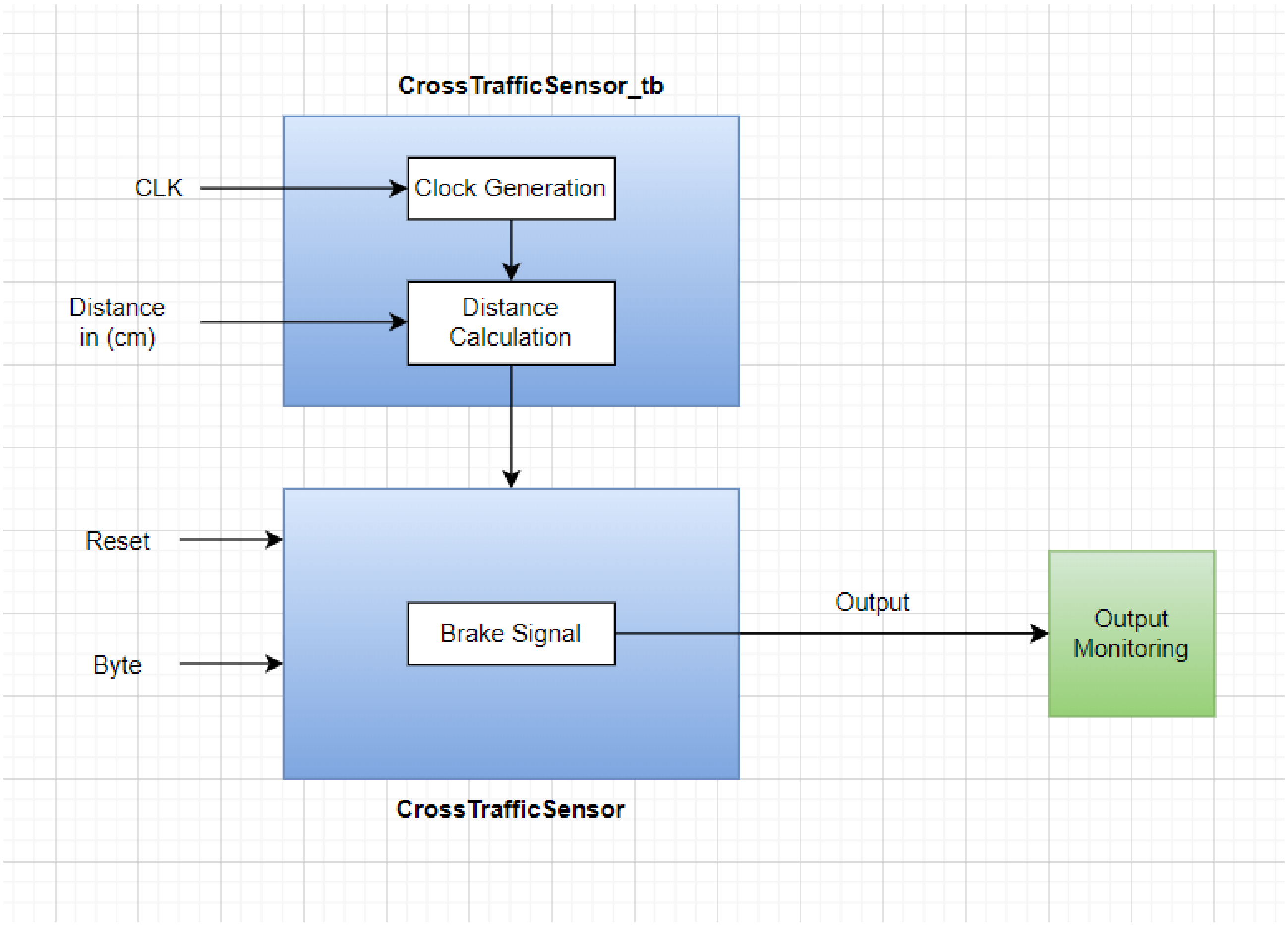


Figure 1: High Level Block Diagram of Cross Traffic Sensor.

Components and Functions of the above block diagram are as follows:-

1. **Clock generation** - The clock generation unit generates a timing signal (clk) that synchronizes all operations on the test bench. This ensures that the sensor's functionalities are simulated in real time and accurately represent realistic conditions.

2. **Distance Calculation** - The distance calculation module processes input data that represents the distance between objects in centimeters. This modules simulates the sensor's ability to calculate and track the distance between vehicles or obstacles, providing critical data for subsequent processing.

3. **Cross Traffic Sensor Module**

   - **Functionality** - The system's core component, the cross traffic sensor module, analyzes distance data to determine the presence of cross traffic. Its purpose is to detect vehicles or obstacles approaching from the sides.
   - **Reset Input** - This input allows the system to reset its operations, erasing all previous states and data. This feature is critical for reliable testing because it allows for repeated trails under various scenarios without introducing residual data interference.
   - **Byte Input** - This represents an interface for additional binary data inputs required for sensor's operation, such as configuration settings or test parameters.

4. **Output Monitoring** - The output monitoring component monitors the results of the cross traffic sensor module. Its primary function is to validate the system's responses and ensure that they are consistent with expected outcomes. The primary output monitored is the brake signal.

5. **Brake Signal** - The brake signal is an important output that indicates whether the system has detected cross traffic is detected at a critical distance, the system sends a brake signals triggering a response aimed at avoiding collisions.

4 Design Description

This section describes the comprehensive design and operational framework of an infrared distance measurement system designed specifically for cross-traffic detection in Advanced driver assistance systems (ADAS). This system is critical for improving vehicle safety by preventing potential collisions with pedestrians, cyclists and other vehicles, especially in urban areas where such interactions are common.

The system comprises several key components: an infrared sensor, a calculator, and a braking system. These components work in tandem to detect objects or obstacles ahead of a vehicle and initiate braking if necessary. System design would look like as follows -

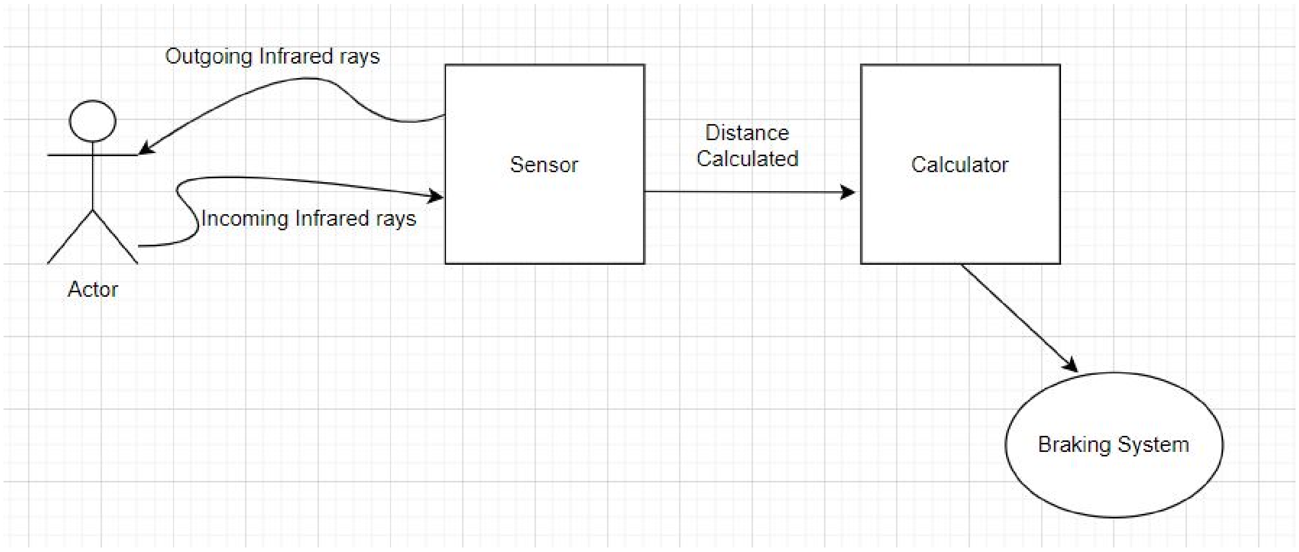


Figure 2: System Design of Cross traffic Sensor.

The above system design has the following design overview.

1. **System Overview**
   The cross traffic infrared distance measurement system is a critical component of an ADAS, serving as a proactive safety mechanism for monitoring and evaluating the surrounding environment. With urbanization leading to increased people walking and complicated driving scenarios, effective cross traffic detection has emerged as a top priority for vehicle manufacturers and safety regulators. [6]

2. **Actor**
   The term "Actor" refers to a variety of entities in the vehicle's vicinity, including pedestrians, vehicles, bicycles and pets. The system uses sophisticated algorithms to evaluate the behavior

of these actors, identifying movements that indicate potential hazards. [7]. By accurately identifying the proximity of these actors, the system improves the driver's situational awareness and assists in the implementation of timely safety measures.

3. **Sensor Functionality**
   Infrared Sensors: The system uses a network of infrared sensors arranged around the vehicle to form a 360-degree detection zone. These sensors emit a series of infrared rays that travel though the surrounding space.

   - **Detection Range and Resolution:** The range of these sensors can be adjusted based on the vehicle's speed and predicted traffic conditions. High-resolution sensors can detect even the smallest movements, ensuring that small pedestrians and cyclists are not overlooked. Recent studies show that increasing sensing range contributes significantly to safer navigation through crowded areas
   - **Multi-sensor Fusion:** To improve reliability and accuracy, the system may employ data fusion techniques, which combine inputs from multiple sensors (such as ultrasonic and radar) with infrared data. This multi-sensor approach overcomes the limitations of single sensor systems, resulting in a more robust detection platform. [8]

4. **Distance Calculation**
   When infrared rays are emitted, the time it takes for them to return after reflecting from an actor is precisely measured. The system's functionality depends on the ability to calculate distance in real time.

$$Distance = Speedoflight * time/2$$

   The formula allows real-time adjustments to the vehicle's speed and direction based on detected distances. According to research, increasing the computational speed of these calculations significantly reduces response times in emergency situations.[9]

5. **Data processing and Decision making**
   Real-time processing: The system's advanced microprocessors immediately analyze the incoming data. This capability enables immediate decision-making based on detected actor movements.

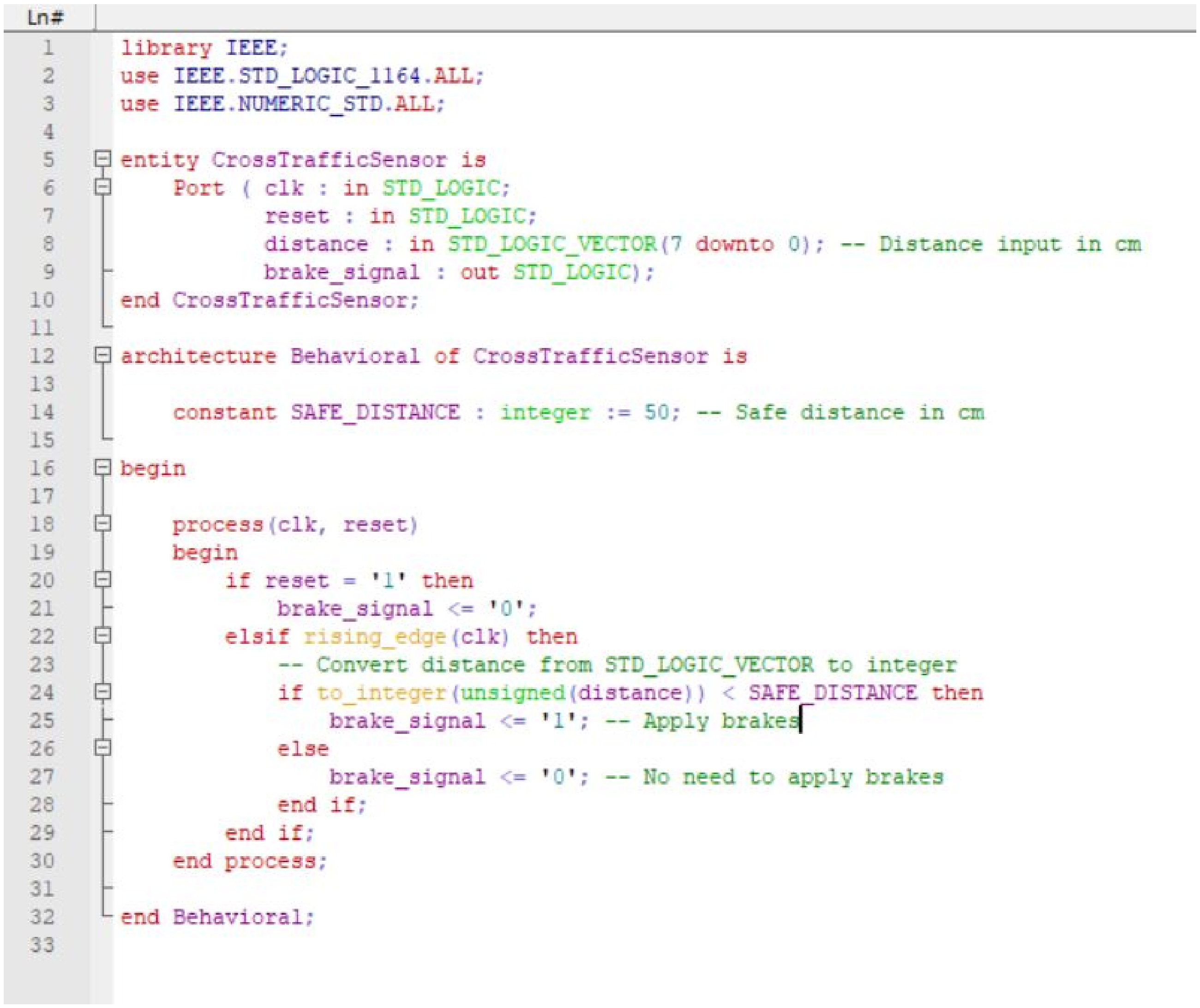
   - **Algorithm Development:** Advanced algorithms, such as machine learning techniques, can adapt and improve over time by learning from a wide range of driving scenarios. These algorithms use object speed trajectory, and other behavioral patterns to predict potential threats [10].

6. **Braking system Activation**
   The system's primary safety feature is its ability to activate the vehicle's braking system automatically when a potential collision is detected.

   - **Threshold Logic:** Predefined threshold for safe stopping distances are determined based on the vehicle's speed and the relative speed of the detected actor. When the calculated distance falls below its threshold, this system activates an alert mechanism (audio or visual) to warn the driver of an impending threat.
   - **Automatic Braking Mechanism:** When the driver does not respond to alerts, the system seamlessly switches to automatic braking, significantly reducing the vehicle's speed or completely stopping it to avoid a collision. This function is especially useful in urban environments where sudden stops and starts are common.

5 Design VHDL

The system described has been written in VHDL and modelSim has been used as an editor to view the simulations. Following is the code for Cross traffic sensor in vhdl with comments.



```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity CrossTrafficSensor is
    Port ( clk : in STD_LOGIC;
           reset : in STD_LOGIC;
           distance : in STD_LOGIC_VECTOR(7 downto 0); -- Distance input in cm
           brake_signal : out STD_LOGIC);
end CrossTrafficSensor;

architecture Behavioral of CrossTrafficSensor is

    constant SAFE_DISTANCE : integer := 50; -- Safe distance in cm

begin

    process(clk, reset)
    begin
        if reset = '1' then
            brake_signal <= '0';
        elsif rising_edge(clk) then
            -- Convert distance from STD_LOGIC_VECTOR to integer
            if to_integer(unsigned(distance)) < SAFE_DISTANCE then
                brake_signal <= '1'; -- Apply brakes
            else
                brake_signal <= '0'; -- No need to apply brakes
            end if;
        end if;
    end process;

end Behavioral;
```

Figure 3: VHDL Code for Cross traffic Sensor.

The VHDL code includes a simple traffic sensor called CrossTrafficSensor. This entity uses and input signal distance to measure the distance between a vehicle and an obstacle and then triggers a brake signal based on a predetermined safe distance.

1. **Entity Declaration**
   The CrossTrafficEntity specifies the following signals:
   - **clk:** A standard logic input that acts ass the clock signal during synchronous operation.
   - **Reset:** A standard logic input used to return the system to its initial stage.
   - **Distance:** An 8-bit standard logic vector representing the measured distance to an obstacle in centimeters.
2. **Outputs**
   - **brake-signal:** A standard logic that whether to apply the brakes ('1' for braking and '0' for no action).
3. **Architecture**
   The behavioral architecture employs a single process that is sensitive to changes in the clock and reset signals. The logic of process works as follows:
   - **Reset Condition:** If the reset signal is activated ('1'), the brake-signal is set to '0', indicating that no brakes should be applied. This allows you to initialize or clear the system's state.
   - **Clock edge Detection:** The process looks for a rising edge in the clock signal. This is common in synchronous designs to ensure that operations occur at time intervals defined by the clock.
   - **Distance Checking:**
     - During the clock edge condition, the code converts the distance from its standard logic vector format to an integer using the following line.
       $$to\_integer(unsigned(distance))$$
     - It then compares the converted distance to a predefined constant called "SAFE_DISTANCE", which is set to 50cm.
     - If the distance is shorter than the safe distance, the brake_signal is assterted ('1'), indicating that the brakes should be applied. In contrast, if the distance exceeds or equals the safe distance the brake_signal is negated ('0'), indicating that no braking is required.
4. **Constants**
   "SAFE_DISTANCE": This constant specifies the threshold distance for brake activation. It is set to 50cm in this design to represent the critical distance at which the system should apply brakes.

6 TestBench Simulations

Implementing a testbench is critical for verifying the functionality and performance of VHDL-designed hardware systems. This report focuses on CrossTrafficSensor_tb, which acts as a testing ground for the CrossTrafficSensor entity. The testbench simulates various distance scenarios to validate the sensor's braking logic.

The testbench below is defined as follows:

1. **Entity Declaration:**
   The testbench has no external ports because it is designed to be used independently to test the unit under test.

2. **Architectural Framework:**
   The behavioral architecture includes various components and processes required for testing.

3. **Component Declaration:**
   The CrossTrafficSensor entity is reference within the testbench, indicating that it is the component being tested. The UUT's ports are connected to the signals defined in the testbench:

   - **clk:**Input clock signal.
   - **reset:**Input a reset signal.
   - **distance:** An 8-bit input that represents the distance to obstacles.
   - **brake_signal:**An output signal that indicates the status of the brakes.

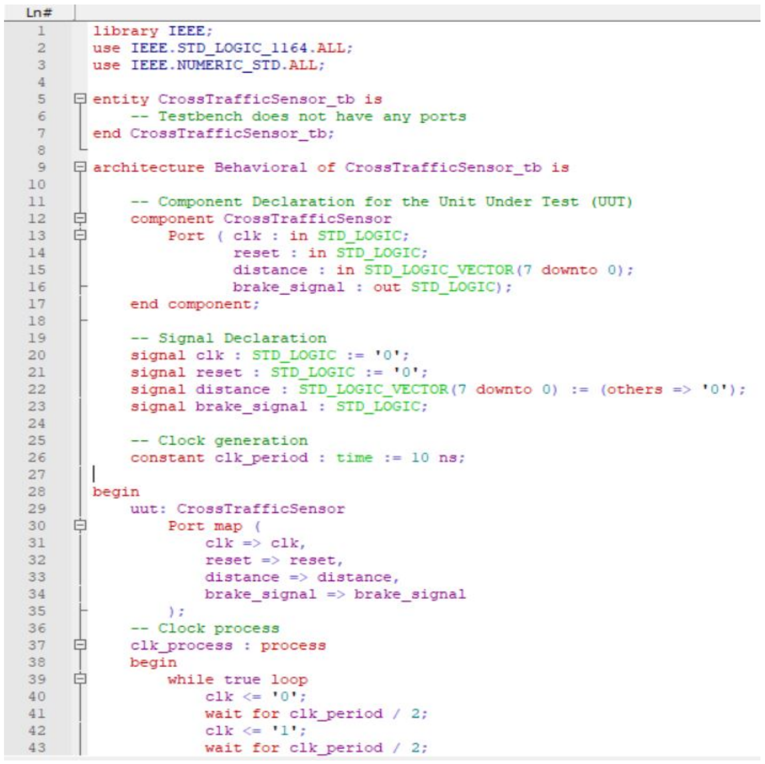
4. **Signal Declaration:**
   The testbench declares the following signals to facilitate functionality:

   - **clk:**A standard logic signal set to '0'.
   - **reset:** A standard logic signal that is set to '0.'
   - **Distance:** A STD_LOGIC_VECTOR(7 down to 0) set to '0.'
   - **brake_signal:**A standard logic signal to capture the brake output.

5. **Clock Generation:**
   A clock generation process called 'clk_process' is created to toggle clk signal. The clock period is set to 10 nanoseconds to ensure that the simulation follows a consistent timing framework.

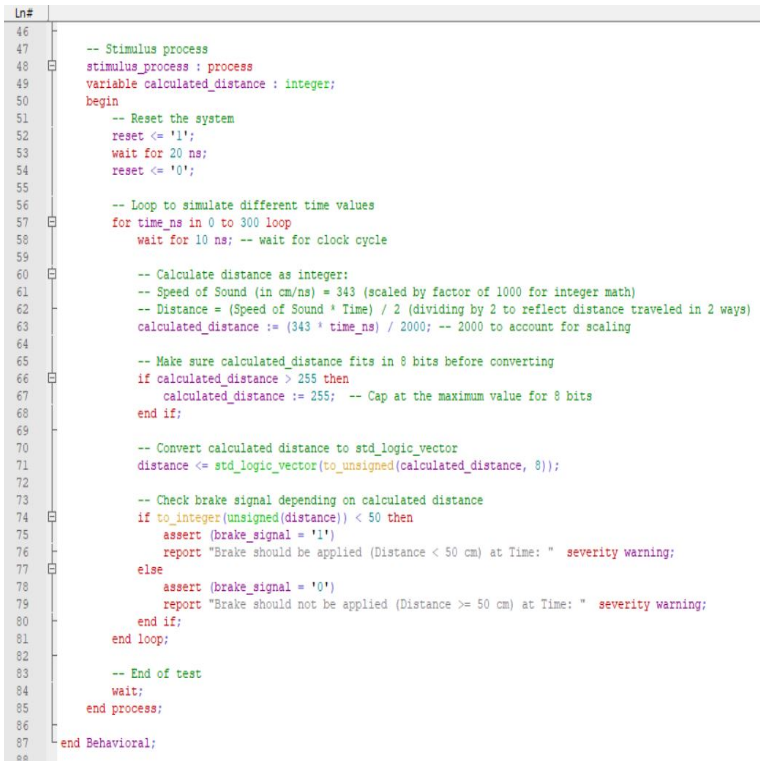
6. **Stimulus Generation:**
   The stimulus_process is in charge of delivering a series of distance inputs and monitoring the brake_signal output. The process works as follows:

   - The system is reset for 20 nanoseconds after a reset signal is pressed.



```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity CrossTrafficSensor_tb is
    -- Testbench does not have any ports
end CrossTrafficSensor_tb;

architecture Behavioral of CrossTrafficSensor_tb is

    -- Component Declaration for the Unit Under Test (UUT)
    component CrossTrafficSensor
        Port ( clk : in STD_LOGIC;
               reset : in STD_LOGIC;
               distance : in STD_LOGIC_VECTOR(7 downto 0);
               brake_signal : out STD_LOGIC);
    end component;

    -- Signal Declaration
    signal clk : STD_LOGIC := '0';
    signal reset : STD_LOGIC := '0';
    signal distance : STD_LOGIC_VECTOR(7 downto 0) := (others => '0');
    signal brake_signal : STD_LOGIC;

    -- Clock generation
    constant clk_period : time := 10 ns;

begin
    uut: CrossTrafficSensor
        Port map (
            clk => clk,
            reset => reset,
            distance => distance,
            brake_signal => brake_signal
        );
    -- Clock process
    clk_process : process
    begin
        while true loop
            clk <= '0';
            wait for clk_period / 2;
            clk <= '1';
            wait for clk_period / 2;
```

Figure 4: Snapshot of testbench part-1



```vhdl

    -- Stimulus process
    stimulus_process : process
    variable calculated_distance : integer;
    begin
        -- Reset the system
        reset <= '1';
        wait for 20 ns;
        reset <= '0';

        -- Loop to simulate different time values
        for time_ns in 0 to 300 loop
            wait for 10 ns; -- wait for clock cycle

            -- Calculate distance as integer:
            -- Speed of Sound (in cm/ns) = 343 (scaled by factor of 1000 for integer math)
            -- Distance = (Speed of Sound * Time) / 2 (dividing by 2 to reflect distance traveled in 2 ways)
            calculated_distance := (343 * time_ns) / 2000; -- 2000 to account for scaling

            -- Make sure calculated_distance fits in 8 bits before converting
            if calculated_distance > 255 then
                calculated_distance := 255;  -- Cap at the maximum value for 8 bits
            end if;

            -- Convert calculated distance to std_logic_vector
            distance <= std_logic_vector(to_unsigned(calculated_distance, 8));

            -- Check brake signal depending on calculated distance
            if to_integer(unsigned(distance)) < 50 then
                assert (brake_signal = '1')
                report "Brake should be applied (Distance < 50 cm) at Time: "  severity warning;
            else
                assert (brake_signal = '0')
                report "Brake should not be applied (Distance >= 50 cm) at Time: "  severity warning;
            end if;
        end loop;

        -- End of test
        wait;
    end process;

end Behavioral;
```

Figure 5: Snapshot of testbench part-2

- A loop iterates to simulate varying time intervals, with the distance calculated using the following sound propagation formula:
  - The formula used is

$$Distance = (SpeedofSound * Time)/2$$

  - The speed of sound is approximated as 343 cm/ns, with scaling adjustments made to maintain integer values.
- The calculated distance is clamped to fit within the 8-bit range, ensuring proper input to the distance signal.

7. **Brake signal Verification:**
   For each calculated distance, assertions are used to validate the brake_signal output. If the distance is less than 50cm, it is expected that the brake_signal will be '1'. In contrast, if the distance equals or exceeds 50cm, the brake_signal should remain '0'. Each condition is reported with appropriate messages, allowing for easy debugging and verification during simulation.

   The CrossTrafficSensor_tb testbench accurately simulates distance scenarios and observes the system's response to ensure that it works properly. This structured approach to testing not only ensures that the logic is correct, but also prepares the system for future integration into real-world vehicular safety applications.

7 Simulation Results

The simulation was carried out using a testbench (CrossTrafficSensor_tb), which verifies the functionality of the CrossTrafficSensor entity. The testbench generates clock signals and varying distance inputs to simulate the sensor's response across a predefined set of scenarios. The main objectives of the simulation were to:

- Check that the brake signal is activated when the distance falls below the 50-cm safety limit.
- Confirm that the brake signal is inactive when the distance is at or above the safe threshold.

Throughout the simulation, the distance input was continuously varied from 0 cm to 51cm. The following is a summary of the specific values generated and their expected outputs based on the braking logic:

Time (ns)	Distance (in cm)	Expected Brake Signal	Actual Brake Signal	Status
30	00000000 (0cm)	1	1	PASS
300	00000101 (5cm)	1	1	PASS
1500	00011110 (30cm)	1	1	PASS
3000	00110011 (51cm)	0	0	PASS

Table 1: Brake Signal Analysis

The following figure is the waveform formed after running the simulations using ModelSim.

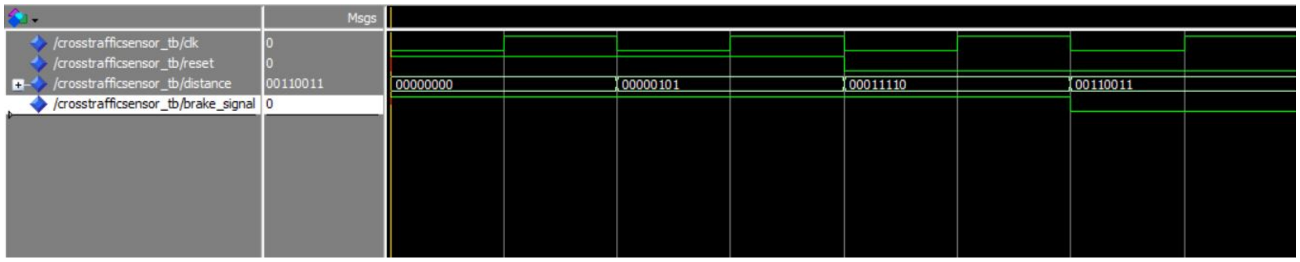

Msgs
/crosstrafficsensor_tb/clk 0
/crosstrafficsensor_tb/reset 0
/crosstrafficsensor_tb/distance 00110011
/crosstrafficsensor_tb/brake_signal 0
00000000
00000101
00011110
00110011

Figure 6: Brake Signal Waveform Simulation.

8 Discussions

This chapter examines the simulation results and discusses the implications of designing and implementing a cross-traffic system.

1. **Performance and Function:**
   The simulation results (Table 1, Figure 6) show that cross traffic sensor accurately detects cross-traffic within the defined safety threshold of 50cm. The brake signal is reliably asserted ('1') when the distance is less than the threshold and de-asserted ('0') when the distance is equal to or greater than the threshold. This confirms that the system works as intended, providing an important safety feature. The use of VHDL and ModelSim enabled efficient and rigorous testing, demonstrating the system's accuracy and responsiveness. The consistent performance across multiple scenarios demonstrates the design's robustness.

2. **Limitations and future work:**
   While the simulation result are positive, certain limitations must be addressed:

   - **Real-world environmental factors:** The simulation does not account for real-world environmental factors such as changing weather conditions (fog, rain, snow), lighting, or the presence of obstacles that could interfere with infrared signal transmission. More testing with a physical prototype in a variety of environments is required to assess the sensor's performance in real-world scenarios.
   - **Sensor accuracy and calibration:** The accuracy of distance measurement is heavily dependent on the accuracy of the infrared sensor and its calibration. Sensor readings may deviate, affecting the system's effectiveness. Comprehensive calibration procedures and real-world testing are essential for ensuring the sensor's accuracy and reliability.
   - **Integration with braking system:** The model simulates brake signal activation, but full integration with an actual braking system necessitates additional development and testing. This includes ensuring that the brake signal activates the braking mechanism correctly and safely. Additional safety mechanisms (redundancy) may also be considered to improve reliability.
   - **Algorithm refinement:** While basic distance calculation is implemented, advanced algorithms (machine learning, for example) that incorporate object recognition and movement prediction could further improve the system's capabilities. This would allow for earlier and more precise warnings, increasing safety.

   Future work will focus on:

   - Real-world field tests are being conducted in various environments to assess the system's performance under a variety of conditions.

- Implementing advanced algorithms to improve obstacle detection and prediction capabilities.
- Creating an adaptive threshold system that adjusts the safety distance dynamically in response to factors such as vehicle speed and road conditions.
- Exploring sensor fusion techniques (combining with other sensor modalities such as radar or ultrasonic) to improve robustness and accuracy in challenging environments.

This thesis presents the design and implementation of a cross-traffic sensor system, which lays the groundwork for a significant safety improvement. While simulations show promise, real-world testing and refinement are required before deploying in commercial vehicles. Addressing the limitations and future work outlined above will result in a more robust and dependable system capable of significantly increasing road safety.

References

[1] S. C.-C. J. Guerrero-IbÃ¡Ã±$ez, J.A.; Zeadally, “Integrationchallengesofintelligenttransportationsyst$

[2] C. J. Z. S. Guerrero-Ibanez, “Ja internet of vehicles: Architecture, protocols, and security,” *IEEE Internet of Things Journal*, vol. 5, no. 5, p. 3701, 2017.

[3] J. A. Guerrero-Ibanez, S. Zeadally, and J. Contreras-Castillo, “Integration challenges of intelligent transportation systems with connected vehicle, cloud computing, and internet of things technologies,” *IEEE Wireless Communications*, vol. 22, no. 6, pp. 122–128, 2015.

[4] N. Ziraknejad, P. D. Lawrence, and D. P. Romilly, “Vehicle occupant head position quantification using an array of capacitive proximity sensors,” *IEEE Transactions on Vehicular Technology*, vol. 64, no. 6, pp. 2274–2287, 2014.

[5] “City safety w/collision warning.” `http://volvo.custhelp.com/app/answers/detail/a_id/9766/~/city-safety-w/collision-warning`. Accessed: 2024-11-21.

[6] Q. A. . A.-S. . Khan, M., “Impact of driver assistance systems on road safety,”

[7] S. R. . K.-A. . Kumar, A., “Infrared sensors: Applications in vehicle automation,”

[8] . K. M. . Pannil, J., “"machine learning techniques for predictive traffic management.,”

[9] R. E. . S.-G. . Gonzalez, C., “"real-time distance measurement for automotive applications.,”

[10] Z. Y. . Q.-J. . Li, K., “"enhancing urban safety through advanced driving assistance systems.,”