Project DSD  
Pet Companion

By:

Mohamed Ibrahim Mohamed 58-16326

Youssif Soryal 58-6817

Khaled Korayem 58-8361

Youssef Eladi 58-10955

Ziad Mohamed 58-9941

Ahmed Ehab 58-0579

Project Report: "Catch Me If You Can" Pet Toy Car

Overview

This project involves designing and implementing a toy car that plays an interactive game with pets. The car moves away when it detects the approach of a pet and turns on LED lights in the dark, allowing continuous play. The design combines FPGA and Arduino with multiple sensors to achieve motion detection, light sensitivity, and dynamic response

Objectives

1. Pet Detection: Use sensors to detect when the pet is near the car.

2. Reactive Movement: Ensure the car moves away in the opposite direction of the detected motion.

3. Light Control: Monitor ambient light and control LEDs accordingly to ensure visibility during play.

Components Used

1. Sensors:

- 4 Ultrasonic Sensors (front, back, left, right): For detecting motion in all directions.

- Light Dependent Resistor: For detecting the light level.

2. Actuators:

- 2 Motors: For movement.

- 1 LED: For lighting.

3. Electronics:

- 2 Transistors: To control motor directions.

- Arduino: For handling sensor data and sending output signal to the FPGA.

- FPGA (MAX 10): Receives input bits from the Arduino and turns the motors accordingly, receives input from the LDR and controls the LED accordingly .

System Design and Implementation

1. Block Diagram

1. Sensors: Ultrasonic sensors are connected to the Arduino to measure distances and detect approaching objects (pet).

2. Light Sensor: Connects to the FPGA to monitor light intensity.

3. Motors and Transitor: Controlled by the FPGA to move the car based on sensor readings.

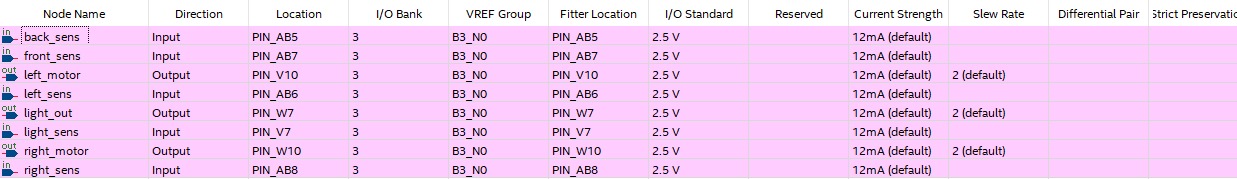
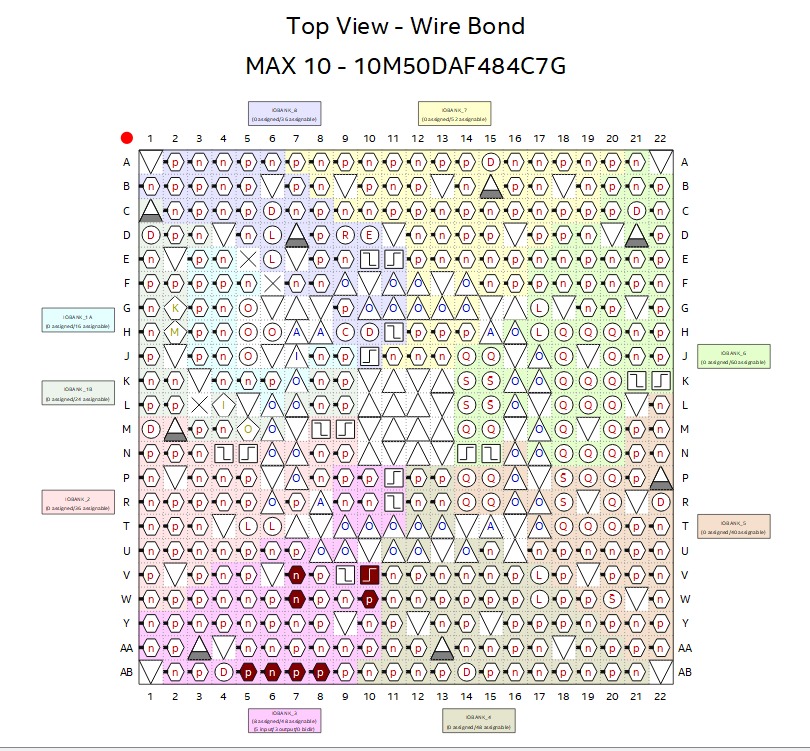
4. Arduino: Receives input from the ultrasonic sensors and outputs the closest sensor to the FPGA

4. FPGA: Manages signal processing and integrates control logic written in VHDL.

5. LED: Controlled by the Arduino to turn on/off based on light sensor data.

2. Pin Assignments

Below are the pin assignments for the FPGA as shown in the pin planner image:



3. Implementation

FPGA (VHDL)

- VHDL was used to program the FPGA, enabling real-time processing of sensor signals and motor control.

**Arduino (C++)**

- The Arduino handles 4 ultrasonic inputs controls the LED, and sends 4 output signals, 3 low, 1 high (high indicating the closest sensor). Code is commented with explanation:

const int prox0Echo = 4; // Back Echo

const int prox1Echo = 6; // Left Echo

const int prox2Echo = 8; // Forward Echo

const int prox3Echo = 10; // Right Echo

const int prox0Trig = 5; // Back Trig

const int prox1Trig = 7; // Left Trig

const int prox2Trig = 9; // Forward Trig

const int prox3Trig = 11; // Right Trig

// Define Output pins

const int backSig = 2;

const int leftSig = 3;

const int frontSig = 12;

const int rightSig = 13;

int duration0, duration1, duration2, duration3;

float distance0, distance1, distance2, distance3;

void setup() {

Serial.begin(9600);

// Initialize Trigger Pins as OUTPUT and set them LOW

pinMode(prox0Trig, OUTPUT);

digitalWrite(prox0Trig, LOW);

pinMode(prox1Trig, OUTPUT);

digitalWrite(prox1Trig, LOW);

pinMode(prox2Trig, OUTPUT);

digitalWrite(prox2Trig, LOW);

pinMode(prox3Trig, OUTPUT);

digitalWrite(prox3Trig, LOW);

// Initialize Echo Pins as INPUT

pinMode(prox0Echo, INPUT);

pinMode(prox1Echo, INPUT);

pinMode(prox2Echo, INPUT);

pinMode(prox3Echo, INPUT);

// Initialize Signal Pins as OUTPUT

pinMode(backSig, OUTPUT);

pinMode(leftSig, OUTPUT);

pinMode(frontSig, OUTPUT);

pinMode(rightSig, OUTPUT);

}

void loop() {

// Measure Back Distance

digitalWrite(prox0Trig, HIGH);

delayMicroseconds(10);

digitalWrite(prox0Trig, LOW);

duration0 = pulseIn(prox0Echo, HIGH);

distance0 = duration0 / 58.0;

// Measure Left Distance

digitalWrite(prox1Trig, HIGH);

delayMicroseconds(10);

digitalWrite(prox1Trig, LOW);

duration1 = pulseIn(prox1Echo, HIGH);

distance1 = duration1 / 58.0;

// Measure Forward Distance

digitalWrite(prox2Trig, HIGH);

delayMicroseconds(10);

digitalWrite(prox2Trig, LOW);

duration2 = pulseIn(prox2Echo, HIGH);

distance2 = duration2 / 58.0;

// Measure Right Distance

digitalWrite(prox3Trig, HIGH);

delayMicroseconds(10);

digitalWrite(prox3Trig, LOW);

duration3 = pulseIn(prox3Echo, HIGH);

distance3 = duration3 / 58.0;

// Variables to store signal states

bool backState = LOW;

bool leftState = LOW;

bool frontState = LOW;

bool rightState = LOW;

// Determine the minimum distance direction and set signals accordingly

if (distance0 < distance1 && distance0 < distance2 && distance0 < distance3) {

backState = HIGH;

} else if (distance1 < distance0 && distance1 < distance2 && distance1 < distance3) {

leftState = HIGH;

} else if (distance2 < distance0 && distance2 < distance1 && distance2 < distance3) {

frontState = HIGH;

} else if (distance3 < distance0 && distance3 < distance1 && distance3 < distance2) {

rightState = HIGH;

}

// Update Signal Pins

digitalWrite(backSig, backState);

digitalWrite(leftSig, leftState);

digitalWrite(frontSig, frontState);

digitalWrite(rightSig, rightState);

// Print all distance measurements to Serial

Serial.print("Back: ");

Serial.print(distance0);

Serial.print(" cm\t");

Serial.print("Left: ");

Serial.print(distance1);

Serial.print(" cm\t");

Serial.print("Forward: ");

Serial.print(distance2);

Serial.print(" cm\t");

Serial.print("Right: ");

Serial.print(distance3);

Serial.print(" cm\t");

// Print Signal States

Serial.print("BackSig: ");

Serial.print(backState == HIGH ? "HIGH" : "LOW");

Serial.print("\t");

Serial.print("LeftSig: ");

Serial.print(leftState == HIGH ? "HIGH" : "LOW");

Serial.print("\t");

Serial.print("FrontSig: ");

Serial.print(frontState == HIGH ? "HIGH" : "LOW");

Serial.print("\t");

Serial.print("RightSig: ");

Serial.println(rightState == HIGH ? "HIGH" : "LOW");

// Small delay to avoid flooding the serial monitor

delay(200);

}

**VHDL code:**

LIBRARY ieee;

USE ieee.std\_logic\_1164.all;

ENTITY the\_great\_sensor\_brain IS

PORT (

back\_sens : IN STD\_LOGIC;

left\_sens : IN STD\_LOGIC;

front\_sens : IN STD\_LOGIC;

right\_sens : IN STD\_LOGIC;

light\_sens : IN STD\_LOGIC;

light\_out : OUT STD\_LOGIC;

left\_motor : OUT STD\_LOGIC;

right\_motor : OUT STD\_LOGIC);

END the\_great\_sensor\_brain;

ARCHITECTURE structure OF the\_great\_sensor\_brain IS

signal sensors : STD\_LOGIC\_VECTOR (3 DOWNTO 0);

BEGIN

sensors <= back\_sens & left\_sens & front\_sens & right\_sens;

PROCESS (sensors, light\_sens)

BEGIN

IF light\_sens = '1' Then

light\_out <= '1';

ELSE

light\_out <= '0';

END IF;

CASE sensors IS

WHEN "0000" =>

left\_motor <= '0';

right\_motor <= '0';

WHEN "1000" =>

left\_motor <= '1';

right\_motor <='1';

WHEN "0100" =>

left\_motor <= '0';

right\_motor <= '1';

WHEN "0001"=>

left\_motor <= '1';

right\_motor <= '0';

WHEN others =>

left\_motor <= '0';

right\_motor <= '0';

END CASE;

END PROCESS;

END structure;

4. Results

1. Motion Detection: The car accurately detects pet movement from all four directions using the ultrasonic sensors.

2. Reactive Movement: Based on sensor inputs, the car moves away in the correct direction, ensuring an engaging game for the pet.

3. Light Control: The LED reliably turns on in low-light conditions, enabling play in dark environments.

Conclusion

This project successfully implemented a responsive and interactive toy car for pets using FPGA and Arduino. The car can detect motion and dynamically respond to movement and can detect light intensity to to change the LED on/off.