MADRAS INSTITUTE OF TECHNOLOGY, ANNA UNIVERSITY

EC5611-VLSI LABORATORY

PROJECT REPORT

IMPLEMENTATION OF LINE FOLLOWER BOT USING BAYSYS-3 ARTIX-7 FPGA BOARD

SUBMITTED BY :

1.Gokul G (2020504528)

2.Jashwin Mari K(2020504534)

INTRODUCTION:

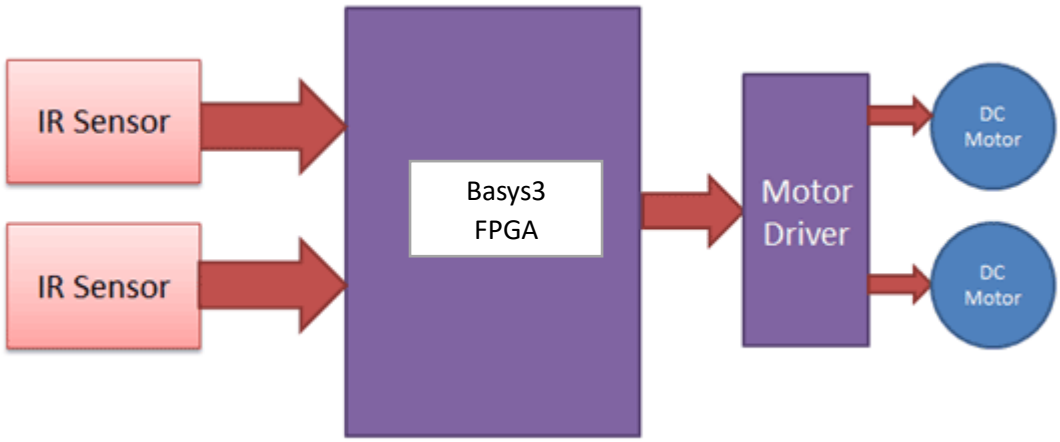
A line follower bot is a common robot design that uses sensors to detect and follow a line on the ground. It detects the line using IR sensors and makes decisions based on the sensor's output. Then a motor driver LM595 is used to control the two DC motor present in robot. The controlling is handled by Basys3 FPGA. The FPGA gets the input from IR sensor, makes decision based on it and actuates the motor accordingly by generating PWM pulses.

To build this, a hardware description language called Verilog was used. The Verilog code will be synthesized and implemented on a basys3 FPGA using Vivado Design suite 17.4.

COMPONENTS USED:

1. Basys3 FPGA board
2. LM595 motor driver
3. IR sensors
4. DC gear motors
5. Lithium ion batteries
6. Jumper wires
7. Robot base

BLOCK DIAGRAM:



PROCEDURE:

Here's a step-by-step explanation of the process:

1. Create a new project in Vivado: Start by creating a new project in Vivado and specify the target FPGA device.
2. Create a new Verilog module: Inside the project, create a new Verilog module. This module will have files that will have inputs, outputs, and internal logic to control bot.
3. Make the required hardware connections between PMOD pins and sensor.

4. Synthesize and implement the design: Once the Verilog module is complete, you can use Vivado's synthesis and implementation tools to convert the Verilog code into a hardware design that can be deployed on an basys3 FPGA. Vivado will generate a bitstream file that contains the configuration data for the FPGA.
5. Program the FPGA: Finally, program the FPGA with the generated bitstream file. This will configure the FPGA to behave as the dice roller, allowing you to simulate the rolling of a dice by interacting with the input signal and observing the output signal.

SOURCE CODE:

```
module generatePWM(input clk, input [3:0] speed, output reg PWM);
//speed 1 - 10
reg [19:0] DutyCounter = 20'd0;
parameter divideClk = 20'd100000;
always@(posedge clk)
begin
   if (DutyCounter < divideClk) DutyCounter <= DutyCounter + 20'd1;
   else DutyCounter <= 20'd0;
   PWM = (DutyCounter < (divideClk*speed/10)) ? 1'b1:1'b0;
end
endmodule
module motor(input clk, input m1EN, input [3:0] speed, input forward, output reg m1PWM, output reg m1IN1, output reg m1IN2, output reg statusLED);
reg [19:0] DutyCounter = 20'd0;
parameter divideClk = 20'd100000;
always@(posedge clk)
begin
```

```
    if (DutyCounter < divideClk) DutyCounter <= DutyCounter + 20'd1;
    else DutyCounter <= 20'd0;
    if(m1EN)
    begin
        statusLED = 1;
        if(forward)
        begin
            m1IN1 = 1;
            m1IN2 = 0;
        end
        else
        begin
            m1IN1 = 0;
            m1IN2 = 1;
        end
    end
    else
    begin
        statusLED = 0;
        m1IN1 = 0;
        m1IN2 = 0;
    end
    m1PWM = (DutyCounter < (divideClk*speed*m1EN/10)) ? 1'b1:1'b0;
end
```

```
endmodule
module controller(
input clk, input CONT_SWITCH,
input IR_L, input IR_R,
input [3:0] speed,
output reg L_M, output reg IN1, output reg IN2,
output reg R_M, output reg IN3, output reg IN4
);
reg [19:0] DutyCounter = 20'd0;
parameter divideClk = 20'd100000;


always@(IR_L or IR_R or CONT_SWITCH)
begin
   if(CONT_SWITCH == 1)
      begin
      case({IR_L, IR_R})
         2'b00: begin IN1 = 0; IN2 = 0; IN3 = 0; IN4 = 0; end
         2'b01: begin IN1 = 0; IN2 = 1; IN3 = 1; IN4 = 0; end
         2'b10: begin IN1 = 1; IN2 = 0; IN3 = 0; IN4 = 1; end
         2'b11: begin IN1 = 1; IN2 = 0; IN3 = 1; IN4 = 0; end
      endcase
      end
   else
      {IN1, IN2, IN3, IN4} = 0;
```

```
end


always@(posedge clk)
begin
    if (DutyCounter < divideClk) DutyCounter <= DutyCounter + 20'd1;
    else DutyCounter <= 20'd0;
    L_M = (DutyCounter < (divideClk*speed/10)) ? 1'b1:1'b0;
    R_M = L_M;
end
//generatePWM CONTROL_L(clk, speed, L_M);
//generatePWM CONTROL_R(clk, speed, R_M);
endmodule
module generatePWM(input clk, input [3:0] speed, output reg PWM);
//speed 1 - 10
reg [19:0] DutyCounter = 20'd0;
parameter divideClk = 20'd100000;
always@(posedge clk)
begin
    if (DutyCounter < divideClk) DutyCounter <= DutyCounter + 20'd1;
    else DutyCounter <= 20'd0;
    PWM = (DutyCounter < (divideClk*speed/10)) ? 1'b1:1'b0;
end
endmodule
```

```
module motor(input clk, input m1EN, input [3:0] speed, input forward, output reg m1PWM, output reg m1IN1, output reg m1IN2, output reg statusLED);

reg [19:0] DutyCounter = 20'd0;

parameter divideClk = 20'd100000;

always@(posedge clk)

begin

   if (DutyCounter < divideClk) DutyCounter <= DutyCounter + 20'd1;

   else DutyCounter <= 20'd0;

   if(m1EN)

   begin

      statusLED = 1;

      if(forward)

      begin

         m1IN1 = 1;

         m1IN2 = 0;

      end

      else

      begin

         m1IN1 = 0;

         m1IN2 = 1;

      end

   end

   else

   begin
```

```
        statusLED = 0;
        m1IN1 = 0;
        m1IN2 = 0;
    end
    m1PWM = (DutyCounter < (divideClk*speed*m1EN/10)) ? 1'b1:1'b0;
end
endmodule
```

TESTBENCH

```
module controller_tb();
reg clk, CONT_SWITCH, IR_L, IR_R;
reg [3:0] speed;
wire L_M, IN1, IN2, R_M, IN3, IN4;
controller con(clk, CONT_SWITCH, IR_L, IR_R, speed, L_M, IN1, IN2, R_M, IN3, IN4);
always #5 clk = ~clk;
initial
begin
clk = 0;
CONT_SWITCH = 1;
speed = 5;
{IR_L, IR_R} = 2'b00;
#4000000
{IR_L, IR_R} = 2'b10;
#4000000
```

{IR_L, IR_R} = 2'b01;

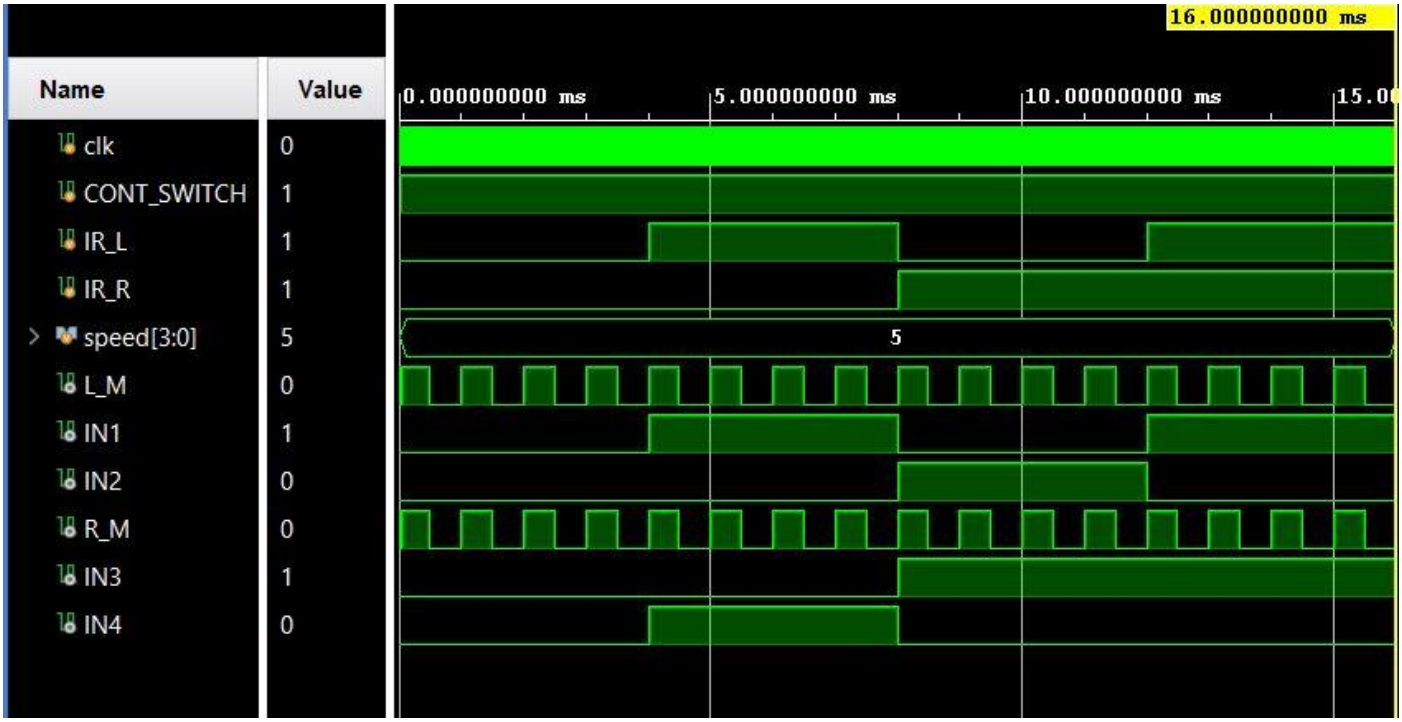
#4000000

{IR_L, IR_R} = 2'b11;

end

endmodule

WAVEFORM



XDC FILE:

Clock signal

set_property PACKAGE_PIN W5 [get_ports clk]

set_property IOSTANDARD LVCMOS33 [get_ports clk]

Switches

set_property PACKAGE_PIN R2 [get_ports {CONT_SWITCH}]

set_property IOSTANDARD LVCMOS33 [get_ports {CONT_SWITCH}]

```
set_property PACKAGE_PIN V17 [get_ports {speed[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {speed[0]}]

set_property PACKAGE_PIN V16 [get_ports {speed[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {speed[1]}]

set_property PACKAGE_PIN W16 [get_ports {speed[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {speed[2]}]

set_property PACKAGE_PIN W17 [get_ports {speed[3]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {speed[3]}]
##Pmod Header JA
##Sch name = JA1
set_property PACKAGE_PIN J1 [get_ports {L_M}]
        set_property IOSTANDARD LVCMOS33 [get_ports {L_M}]
##Sch name = JA2
set_property PACKAGE_PIN L2 [get_ports {IR_L}]
        set_property IOSTANDARD LVCMOS33 [get_ports {IR_L}]
##Sch name = JA3
set_property PACKAGE_PIN J2 [get_ports {IN1}]
        set_property IOSTANDARD LVCMOS33 [get_ports {IN1}]
##Sch name = JA4
set_property PACKAGE_PIN G2 [get_ports {IN2}]
        set_property IOSTANDARD LVCMOS33 [get_ports {IN2}]
```

##Sch name = JA7

set_property PACKAGE_PIN H1 [get_ports {R_M}]

set_property IOSTANDARD LVCMOS33 [get_ports {R_M}]

##Sch name = JA8

set_property PACKAGE_PIN K2 [get_ports {IR_R}]

set_property IOSTANDARD LVCMOS33 [get_ports {IR_R}]

##Sch name = JA9

set_property PACKAGE_PIN H2 [get_ports {IN3}]

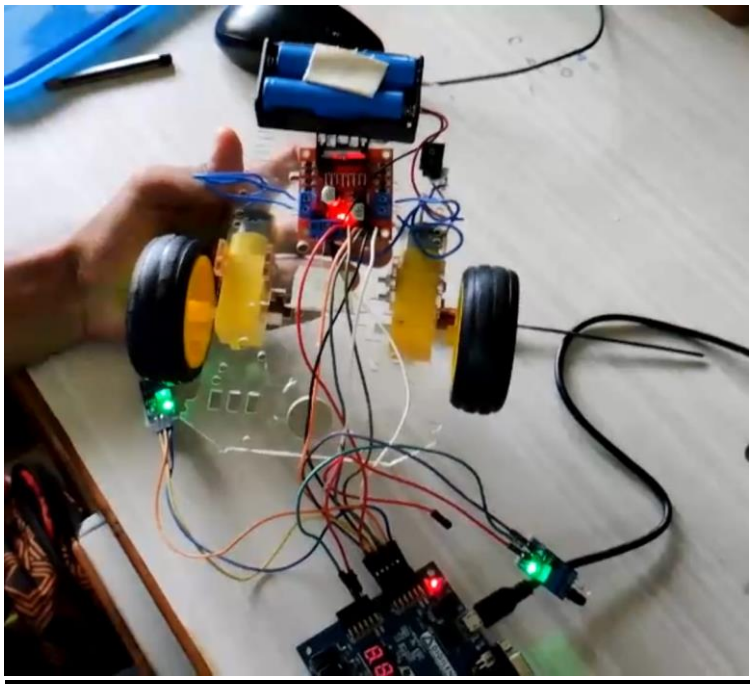
set_property IOSTANDARD LVCMOS33 [get_ports {IN3}]

##Sch name = JA10

set_property PACKAGE_PIN G3 [get_ports {IN4}]

set_property IOSTANDARD LVCMOS33 [get_ports {IN4}]

OUTPUT:



RESULT:

Thus a line follower robot was built successfully using basys3 FPGA .