**INDEX** **2**

**1. PURPOSE** **3**

**2. METHODOLOGY** **3**

*a.* Components and Design Specifications 3

b. Timer Module 5

c. Servo Driver Module 6

*d.* Top Module 7

**3. RESULTS** **8**

**4. CONCLUSION** **10**

**5. REFERENCES AND VIDEO** **11**

**6. APPENDIX** **11**

*a.* Top Module Code 11

b. Timer Module Code 15

c. Servo Driver Module Code 18

## 1) PURPOSE

Aim of this project is to design a line following car using external components and BASYS 3 FPGA board to implement the digital circuit design. It is aimed to exploit all the digital design sources instructed in EE102 digital circuit design lecture (VHDL language, BASYS 3 board and concepts of digital circuit design). It is planned to design a digital control circuit on the FPGA board with VHDL to control the car and realize the line following feature of the car.

## 2) METHODOLOGY

While designing the project first it is investigated which components are needed to achieve desired features and then it is investigated how to use them. Design of the digital circuit implemented in a modular form. Digital circuit modules needed to control external components are planned and implemented and in the top module the final design is realized using previously designed digital circuit modules.

### a) Components and Design Specifications

The line following feature is achieved by using black strip to mark the line and a digital IR light sensor, namely Reflective Optical Sensor TCRT 5000. The IR sensor includes an infra red light emitter and a IR light receiver that detects reflected IR light whose magnitude exceeds a threshold value. Since the sensor has digital output its output lead gives high voltage when IR light detected otherwise low voltage. As black strip absorbs IR light when the sensor is over the black strip the sensor gives 0v, otherwise if the sensor is over the

reflective surface it gives 3.3v. By using this feature of the sensor car detects the black line. Then in the top module it is designed that if the sensor output is 0 then car moves forward, else car starts to turn to left and scans 90 degrees area, a timer is started and limits the turning for 90 degrees left, searching for the line if it finds the line (sensor output is 0) it moves forward, otherwise it starts to turning right scanning 180 degrees area, again a timer is started to limit the turning for 180 degrees, if it finds the line it moves forward otherwise it stops.

I) Components

a) Continuous servo motors

b) IR light sensor

c) Additional physical parts of the car

A platform to place BASYS 3 board, 2 servos, light sensor and battery is needed. Also, a ball caster to make the car stand.

Continuous servo motors Feetech FS90R are used to generate the movement of the car. They are controlled by PWM signals servo motors stop if the pulse width is 1.5ms, turn in the counterclockwise direction if the pulse width is greater than 1.5 otherwise the servos turn in the clockwise direction.

IR light sensor is a digital infra red light sensor. TCRT5000 is used as a infra red light sensor. It both transmits and receives infra red light, if the car is on the black line the IR sensor gives low voltage output as the black surface absorbs the IR light, otherwise if the car is on a reflecting white surface it receives IR light and gives high voltage output.

Other additional physical components are used to place components and provide physical support such as ball caster and a external battery to power the BASYS 3 board and the external components.

b) **Timer Module**

To achieve the 180 degrees surface scanning feature a timer needed to limit the turning time otherwise car only search for right or left only. The timer module is designed such that it has a sr signal input (in the top module it will be assigned to the IR sensor output signal) when the sr signal is 0 the timer stops asynchronously otherwise it starts to count. As the internal clock frequency of BAYS 3 is 100 MHz, a clock divider is used to get a 1hz signal. In the top module the timer's 1s period clock signal is used. When the light sensor gives 1, high voltage the car is on white surface, the timer starts to count and the car turns left for 1 second (90 degrees ) and turns right for 2 seconds (180 degrees ). The counter counts from 0 to 7 and gives a 3 bit output when it reaches 111 it stops counting this feature causes car to stop after searching for 180 degrees area and not finding the black line. The schematic of the timer circuit can be seen bellow.

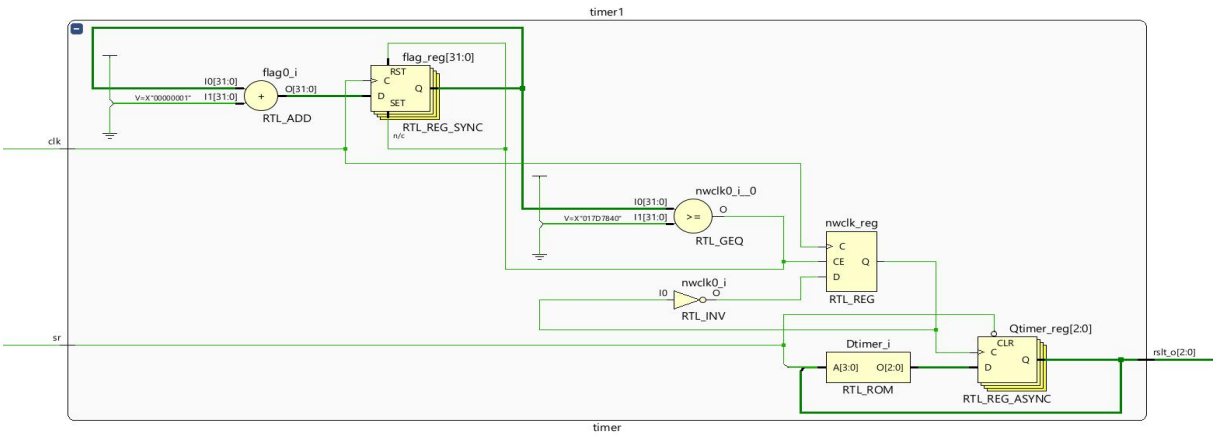


Figure 2b.1 schematic of timer circuit.

## c) Servo Driver Module

The servo motors are controlled by PWM signals generated by clock divider circuits using 100 MHz internal clock signal of BASYS 3, if the pulse width of the signal is 1.5 it stops, or if it is greater than 1.5 it turns counterclockwise otherwise in the clockwise direction**.** The servo driver module generates desired PWM signal to control the continuous servos. 3 different PWM signals needed to provide stopping, turning in the clockwise direction and in the counter clockwise direction. A two bits mode signal is given as a input signal it chooses the constant values with a multiplexer. If the mode signal is "10" then it chooses the constant value for the clock divider to generate a 2ms pulse width PWM signal (counterclockwise turning), if the mode signal is "01" it leads to the generation of 1ms PWM (clockwise turning) ,otherwise it leads to 1.5ms PWM (stopping) generation. And also a sw input as a start signal is given as a input if it is 1 the circuit starts to work. Consequently, the circuit outputs the desired PWM signals regarding to mode signal. The schematic of the servo driver circuit can be seen bellow.

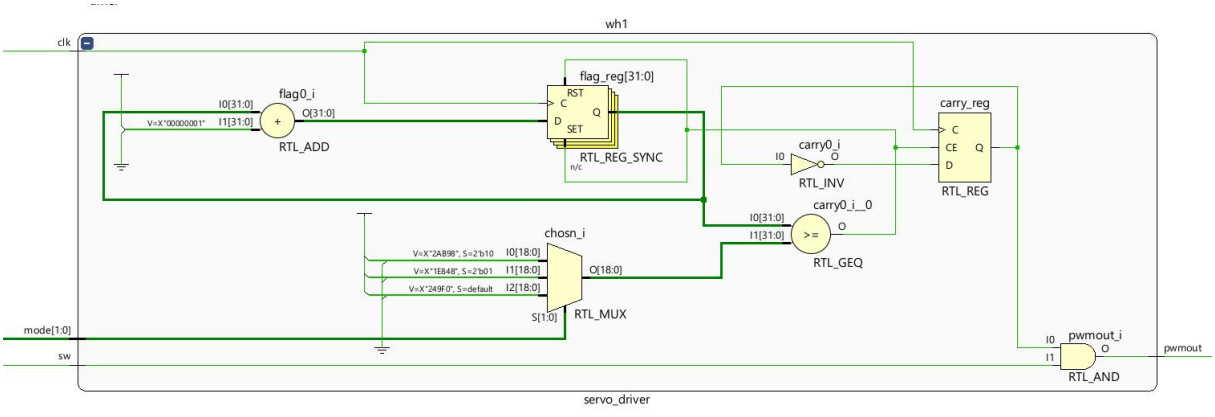


Figure 2c.1 schematic of servo driver circuit.

### d) Top Module

In the top module two servo driver components are instantiated to control two servos, and a timer module to generate the desired 180 degrees searching behaviour. The PWM signals for servos are controlled by the select signal created by concatenation of IR light sensor signal and 3 bits counter signal. The mode signals for servo driver modules are attained the values regarding the select signal. If the select signal is 0000 meaning the car is on the black line the right servo must turn in the clockwise direction and the left servo in the counter clockwise direction to make car go forward; hence, right servo's driver module is given "01" and left one is given "10" signal. If the IR light sensor outputs 1 meaning the car is on the white surface and the car should search the surface at first when the select signal is 1000 car stops for 1 seconds then it is assigned that when the select signal is 1001 the car turns to the left for 1 seconds searching 90 degree area (right servo in clockwise left servo is also in the clockwise direction mode signals are given as 01). Then if the car cannot find black line in the 90 degrees area it stops for 2 seconds ,select signal 1010 and 1011, then starts to search for 90 degrees right of the initial position to achieve that it should turn 90 degrees right for 1 second first to get initial position and 1 second right to scan the right area ,select signals 1100 and 1101, by turning both servos in counter clockwise direction , mode signals 10, and if it cannot find the black line in that time interval it stops. As a result, line following behaviour is achieved. The schematic of the top control circuit module can be seen below.

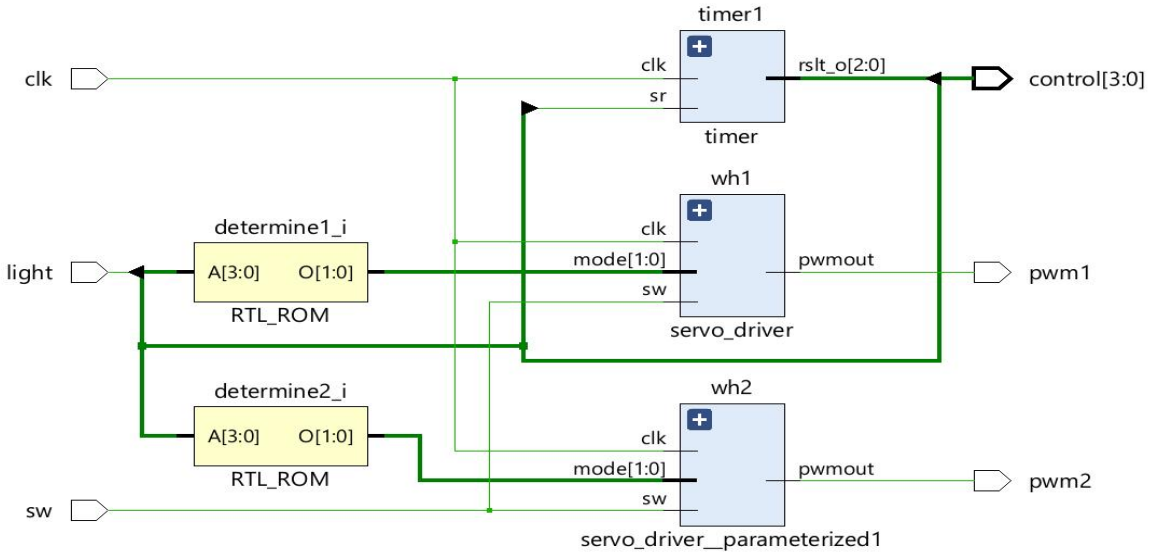


Figure 2d.1 schematic of top module.

### 3) RESULTS

Simulation results are added also the working project is exhibited on the youtube video, link can be found in chapter 5 references and video link chapter.

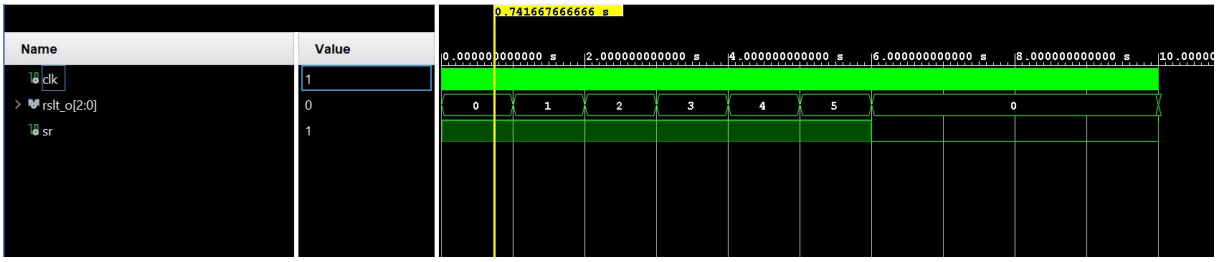


Figure 3.1 timer module simulation result. As the sr signal, IR light sensor output in the top module, is taken high voltage value counter counts in 1s steps and takes value 0 as sr is taken value 0. sr 0 means car is on the line otherwise 1 and to limit the scanning 180 degrees the timer is initiated when car is not on the line.

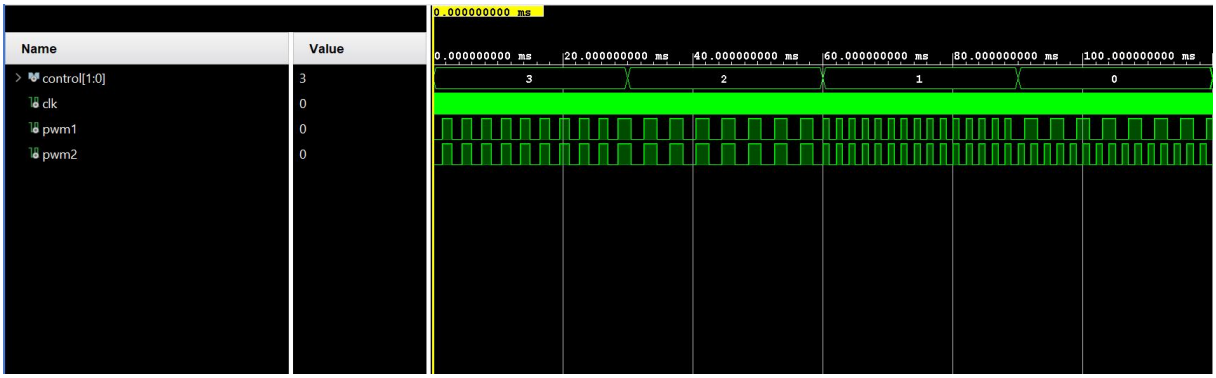


Figure 3.2 servo driver module simulation result. When different control signals given, circuit generates 3 different PWM signals with different pulse widths to stop, turn clockwise and counter clockwise the servo motors.

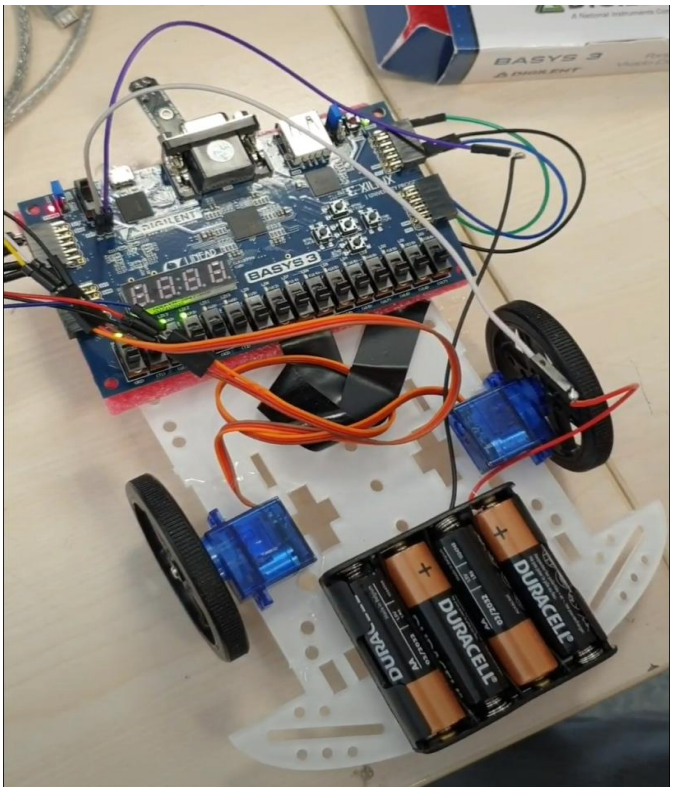


Figure 3.3 the complete car. PWM outs for the servos connected to PMOD I/O's of the BASYS 3 and also the IR light sensors output signal is taken from the PMOD. External power source connected to board.

## 4) CONCLUSION

In this project a line following car is implemented by using external components and BASYS 3 board and digital circuit design made using VHDL and realized by BASYS 3 board. The features of BASYS 3 board and VHDL language are aimed to be utilized and it is learned how to connect and use external components to BASYS 3 board to implement various digital design projects using them.

To realize line following car design different modules and digital circuits needed to be implemented. To control continuous servo motors a DC voltage output and a pulse width modulation signal are needed. As DC outputs BASYS 3 board's pmode 3.3v and gnd ports are used. To generate PWM signal BASYS 3 board's internal 100MHz, 10ns clock signal and clock divider circuits are used. It is needed to generate 3 PWM signal for achieving stop, clockwise and counterclockwise rotation of servos.

Car designed to has 1 infrared digital light sensor to follow the line. The sensor has 3 leads 2 for Vcc and gnd and one for the output signal. Absence of the IR light causes the output to be 0 otherwise 1. As the black line drown by black strip absorbs IR light if the car is on line signal is 0 otherwise 1. If the signal is 0 car moves forward otherwise scans the 180 degrees area of the surface. The scanning realized by turning the car around itself (one servo moving car to forward other backward causes it to rotate) in a predetermined period of time (time last for scanning 180 degrees). To implement timer also 100MHz clock signal of the board is used.

Consequently in the top module all the desired features such as line following or surface scanning are implemented on the same digital circuit.

## 5) REFERENCES AND VIDEO

### Video Link:

https://www.youtube.com/watch?v=IYBg6x32VF8

### References:

Brown, Stephen D., and Zvonko G. Vranesic. *Fundamentals of Digital Logic with VHDL Design*. McGraw Hill, 2023.

Shenzhen Feetech RC Model company, “Analog 360 Degree Continuous rotation Servo FS90R” FS90R datasheet.

Vishay Semiconductors, “Reflective Optical Sensor with Transistor Output” TCRT5000, TCRT5000L datasheet.

## 6) APPENDİX

### a) Top Module Code

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;



entity top_module is

    Port ( clk : in STD_LOGIC;

           sw : in std_logic;

           pwm1 : out STD_LOGIC;
```

```
        pwm2 : out STD_LOGIC;
        control: out std_logic_vector (3 downto 0);
        light : in STD_LOGIC);
end top_module;

architecture Behavioral of top_module is
component servo_driver is
   generic ( pwmp1 : integer:=150000;
             pwmp2 : integer:=200000;
             pwmp3 : integer:=100000 );
   Port (
        pwmout : out STD_LOGIC;
        clk : in STD_LOGIC;
        sw : in std_logic;
        mode : std_logic_vector (1 downto 0));
end component;

component timer is
```

```
    generic  ( const : integer := 100000000);
    Port ( clk : in STD_LOGIC;
           sr : in STD_LOGIC;
           rslt_o : out STD_LOGIC_VECTOR (2 downto 0));
end component;
signal timeing : std_logic_vector (2 downto 0):= "000";
signal selector : std_logic_vector (3 downto 0):="0000";
signal determine1 : std_logic_vector (1 downto 0):="00";
signal determine2 : std_logic_vector (1 downto 0):="00";
begin
wh1 : servo_driver generic map( pwmp1 => 150000, pwmp2 => 175000, pwmp3 => 125000 )
port map(pwmout=>pwm1,clk=>clk,mode=>determine1,sw=>sw);
wh2 : servo_driver generic map( pwmp1 => 150000, pwmp2 => 200000, pwmp3 => 100000 )
port map(pwmout=>pwm2,clk=>clk,mode=>determine2,sw=>sw);
timer1 : timer generic map(const=> 25000000 ) port map(clk=>clk,sr=>light,rslt_o=>timeing);
selector <= light & timeing;
with selector select
determine1 <= "01" when "1001",
```

```
        --"01" when "1010",

        --"01" when "1011",
        "10" when "1100",

        "10" when "1101",
        --"10" when "1110",

        "01" when "0000",

        "00" when others;

with selector select
determine2 <= "01" when "1001",
        --"01" when "1010",

        --"01" when "1011",
        "10" when "1100",
```

```
            "10" when "1101",
            --"10" when "1110",

            "10" when "0000",

            "00" when others;
control <= selector;
end Behavioral;
```

**b) Timer Module Code**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;

entity timer is
    generic ( const: integer:= 100000000 );
    Port ( clk : in STD_LOGIC;
```

```
        sr : in STD_LOGIC;
        rslt_o : out std_logic_vector (2 downto 0));
end timer;

architecture Behavioral of timer is
signal flag: integer:=0;
signal Dtimer: std_logic_vector (2 downto 0):="000";
signal Qtimer: std_logic_vector (2 downto 0):="000";
signal nwclk : std_logic:='0';
signal chose : std_logic_vector (3 downto 0):="0000";
begin
process(clk)
begin
if rising_edge(clk) then
   if flag >= const then
      nwclk <= not(nwclk);
      flag <= 1;
   else
```

```
        flag <= flag + 1;
    end if;
end if;
end process;
process(nwclk,sr)
begin
if sr = '0' then
    Qtimer <= "000";
else
    if rising_edge(nwclk) then
        Qtimer <= Dtimer;
    end if;
end if;
end process;
chose <= sr & Qtimer;
with chose select
Dtimer <= "111" when "1111",
          "001" when "1000",
```

```
        "010" when "1001",
        "011" when "1010",
        "100" when "1011",
        "101" when "1100",
        "110" when "1101",
        "111" when "1110",
        "000" when others;
rslt_o <= Qtimer;
end Behavioral;
```

**c) Servo Driver Module Code**

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

entity servo_driver is
    generic ( pwmp1 : integer:=150000;
              pwmp2 : integer:=200000;
              pwmp3 : integer:=100000 );
```

```
    Port ( pwmout : out STD_LOGIC;
           clk : in STD_LOGIC;
           sw : in std_logic:='0';
           mode : in STD_LOGIC_VECTOR (1 downto 0));
end servo_driver;

architecture Behavioral of servo_driver is
signal flag : integer :=0;
signal carry: std_logic:='0';
signal chosn : integer:=75000;
begin
with mode select
chosn <= pwmp2 when "10",
         pwmp3 when "01",
         pwmp1 when others;
process(clk)
begin
if rising_edge(clk) then
```

```
if flag >= chosn then

carry<= not(carry);

flag <= 1;

else

flag <= flag+1;

end if;

end if;

end process;

pwmout<= carry and sw;



end Behavioral;
```