VENDING MACHINE

EEE102 Introduction to Digital Circuit Design Course Project

Şevval Erbay
22202658

Table of Contents

Objective:

The aim of this project is to design and implement a coinless vending machine by using BASYS3 FPGA board and VHDL. The project is designed as a pantry organization system. If the ultrasonic sensor detects a hand in front of the corresponding lane, the servo motor in that lane will turn 90° and give exactly one product.

Methodology:

In this project, a simple vending machine is implemented. Initially, it looked like a single ramp with 3 different slots for products to roll. After the project demo, the design was changed and the whole system was separated into three different ramps for simplicity reasons. At the base of each ramp, there is a servo motor to control the product distribution. Each of these servos are controlled by their corresponding ultrasonic sensors in front of them.

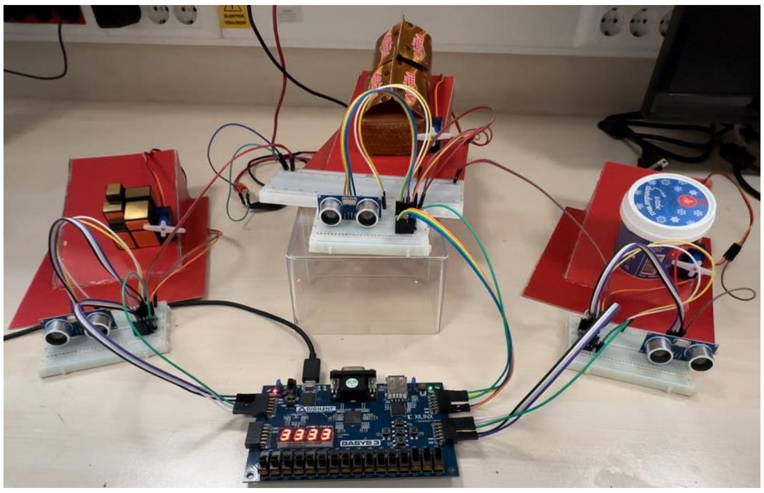

Figure.1 Finalized vending machine

A BASYS3 FPGA board is utilized to control the distribution process. It takes data from the ultrasonic sensors and give the appropriate output to the servo motors. 3 of the switches on the BASYS3 board are used to choose which lanes will be available to service. The number of in-service lanes are displayed on the seven segment display. Initially, the aim was to show the in-service lanes by their corresponding LEDs on the BASYS3 FPGA board. After the project demo, this design choice was changed. In the final product, the LEDs light up when that particular lane is handing out products. This choice was implemented to give both visual and sensory output to the person using the machine. This way, it is easier to check whether there is a problem with the machine or not. In other words, LEDs are used as a self-check system.

3 ultrasonic sensors (HC-SR04) and 3 servo motors (SG90) are used in this project. Since those component use 5V DC and the BASYS3 board works with 3.3V DC, 3 logic level converters are used. The job of logic level converters could be done by designing a voltage divider circuit but for simplicity reasons, converters are used in this project.

- **HC-SR04 Ultrasonic Sensor:** Ultrasonic sensor has 4 pins: Vcc, TRIG, ECHO, GND. Vcc and GND are used to feed 5V to the component. To use an ultrasonic sensor, TRIG pin is set HIGH for 10μs. Then, the sensor transmits an ultrasonic wave for eight pulses at 40 kHz. At the same time, ECHO pin goes HIGH to initiate the echo-back signal. If those pulses are reflected back, ECHO pin obtains LOW.
- **SG90 Servo Motor:** This particular servo motor can rotate between 0-180°. In this project, it is chosen for the servo to rotate 90°. The servo motor has 3 pins: Vcc, GND, PWM. Vcc and GND is used to power the servo with 5V DC. A servo motor works with PWM (Pulse Width modulation) signals. PWM signals help us control analog components by digital controllers. This signal is made of square wave pulses. This way, the wave will either be HIGH or LOW at any given time. In order to generate this PWM signal, we use BASYS3 FPGA board. The internal clock frequency of the BASYS3 FPGA board is 100 MHz. Since

this frequency is much larger than what we need to use for our project, a 64 KHz clock divider is used. A PWM pulse is generated every 20 ms to ensure that the servo motor stays in its intended position. The length of that pulse determines the position of the servo motor.

- **Logic Level Converter:** Creates communication between components and BASYS3 FPGA board by converting different levels of voltage.

The cost table is below (prices of the electronics are taken from https://www.robotistan.com/ at 31.12.2023):

Component	Number	Cost (TL)
HC-SR04	3	107.76
SG90	3	146.55
Logic Level Converter	3	32.34
Mini breadboard	3	122.97
Large Breadboard	1	42.24
Jumper Cables	40	40.59
BASYS3 FPGA	1	7764.49
Cardboard	1	15
	Total:	8271.94 TL

Table.1 Cost of the project

The connection between the components and the BASYS3 is shown below.

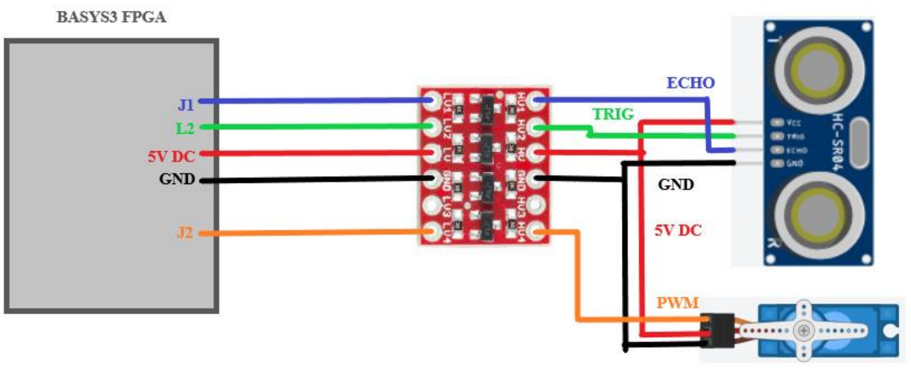

Figure.2 Connections between parts (lane 1)

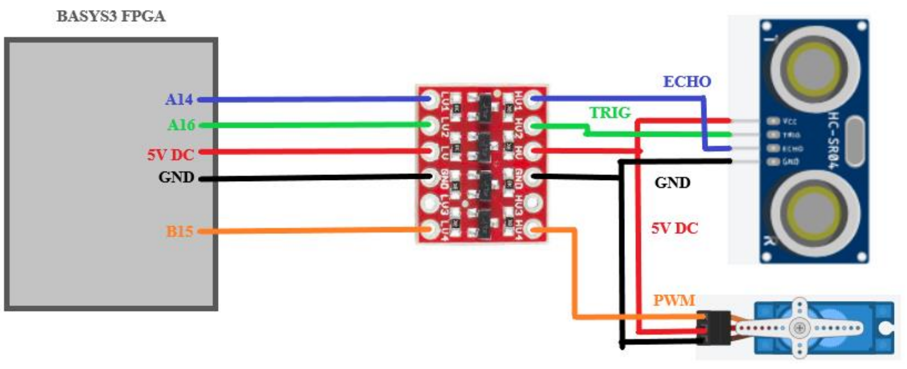

Figure.3 Connections between parts (lane 2)

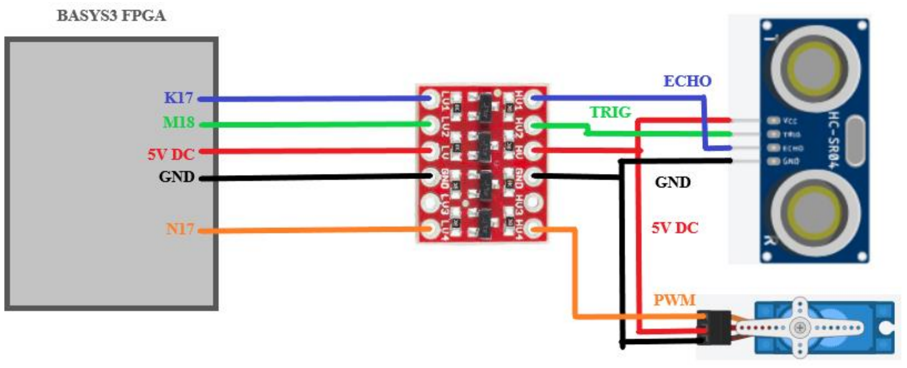

Figure.4 Connections between parts (lane 3)

The duties of the codes and their respective RTL schematics are explained below:

- **design:** Combines all of the sub modules together. Checks the value coming from each ultrasonic sensor and hands out products as needed. Also controls the disabling-by-switch functions and lighting up the LEDs. Inputs are:
    - *clk: 100 MHz internal clock of the BASYS3*
    - *echo0: ECHO of ultrasonic sensor 1*
    - *echo1: ECHO of ultrasonic sensor 2*
    - *echo2: ECHO of ultrasonic sensor 3*
    - *trig0: buffer 1*
    - *trig1: buffer 2*
    - *trig2: buffer 3*
    - *switch : Right-most 3 switches of the BASYS3*

Outputs are:

- *servo1: servo motor 1*
- *servo2: servo motor 2*
- *servo3 : servo motor 3*
- *led : Right-most 3 LEDs of the BASYS3*
- *display : seven-segment display*

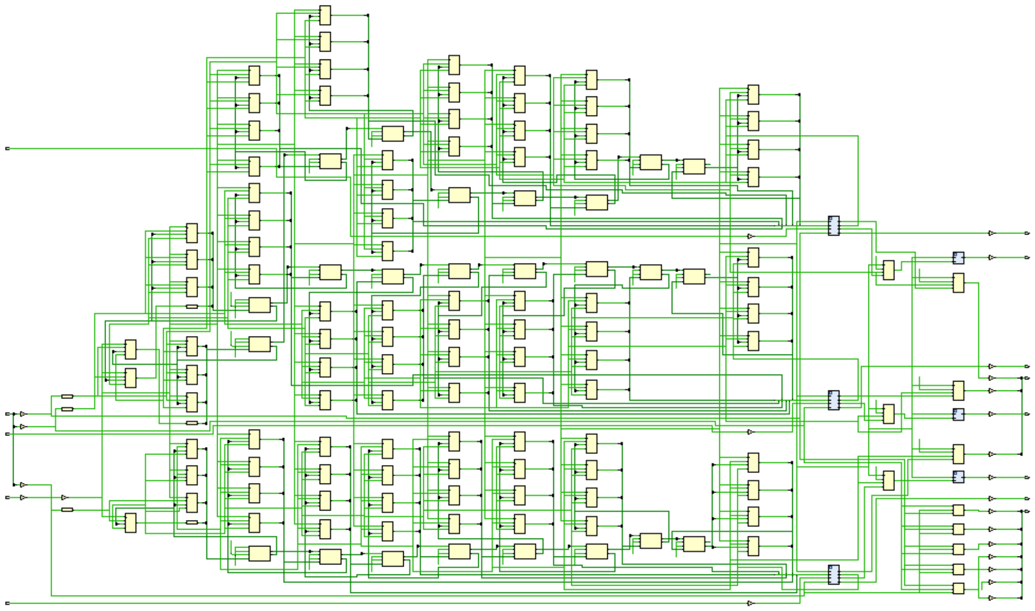

Figure.5 RTL schematic of design.vhd

- **distance:** Sends TRIG signals and receives them. Then checks the time and assigns the corresponding distance values. The output is given as binary numbers.

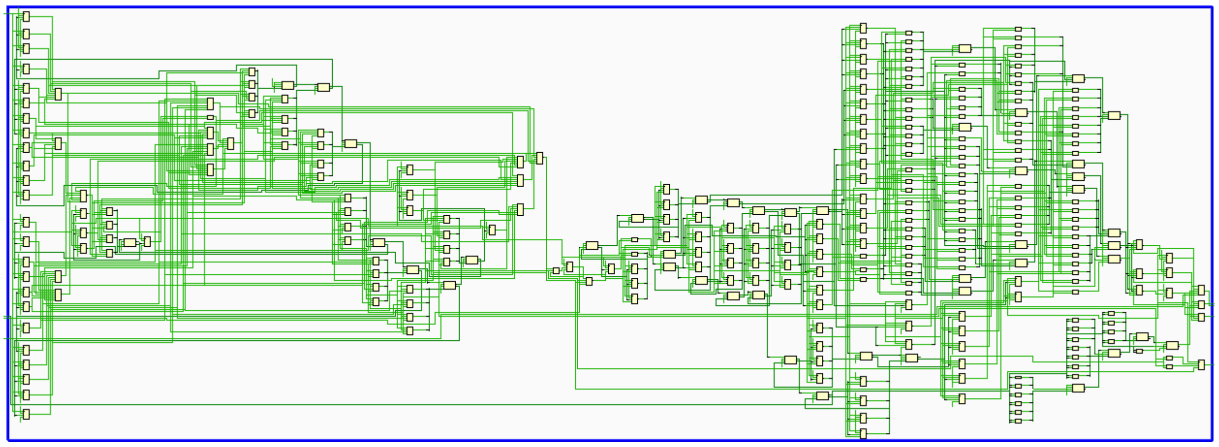

Figure.6 RTL schematic of distance.vhd

- **ssd:** Determines the value to be displayed on the seven-segment display depending on the configuration of the switch positions.

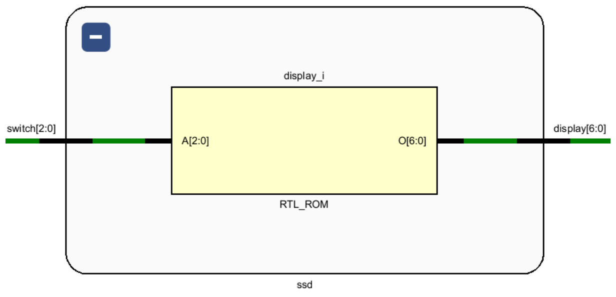

Figure.7 RTL schematic of ssd.vhd

- **servo:** Combines the 2 sub modules together.

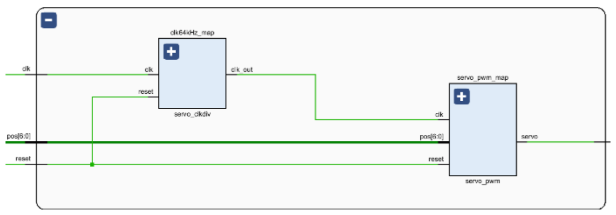

Figure.8 RTL schematic of servo.vhd

- **clkdiv:** The clock divider code used to create a 64 KHz clock to use for controlling the servo motors.

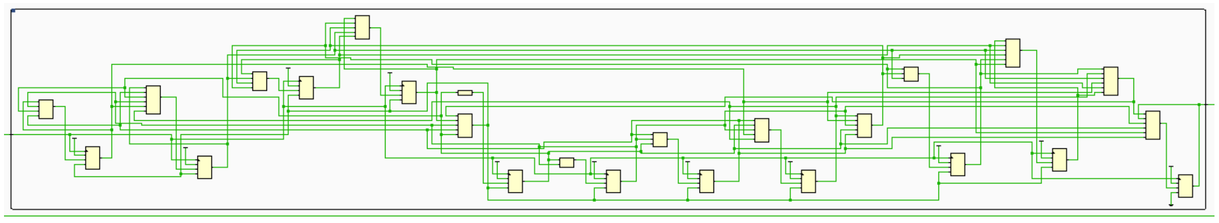

Figure.9 RTL schematic of clkdiv.vhd

- **pwm:** Creates the PWM pulses to control the servo motor.

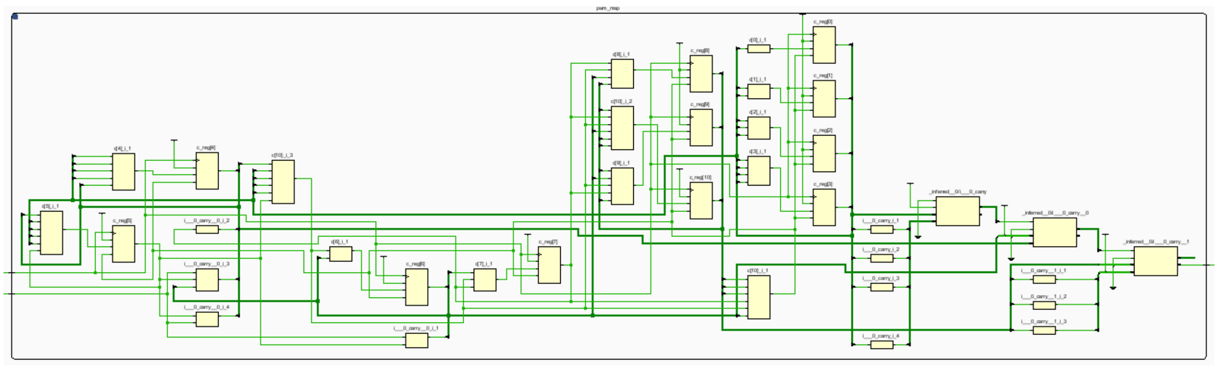

Figure.10 RTL schematic of pwm.vhd

Results:

All the parts of the project worked accordingly. All of the distribution lanes are tested and the results are shown in the following figures. Since all of the 3 switches are ON, the seven-segment display lights up number 3. Notice that when a product is being distributed, the corresponding LED lights up.

- Switch V17 (right- most): Lane 3, LED is U19
- Switch V16 (middle): Lane 2, LED is E19
- Switch W16 (left-most): Lane 1, LED is U16

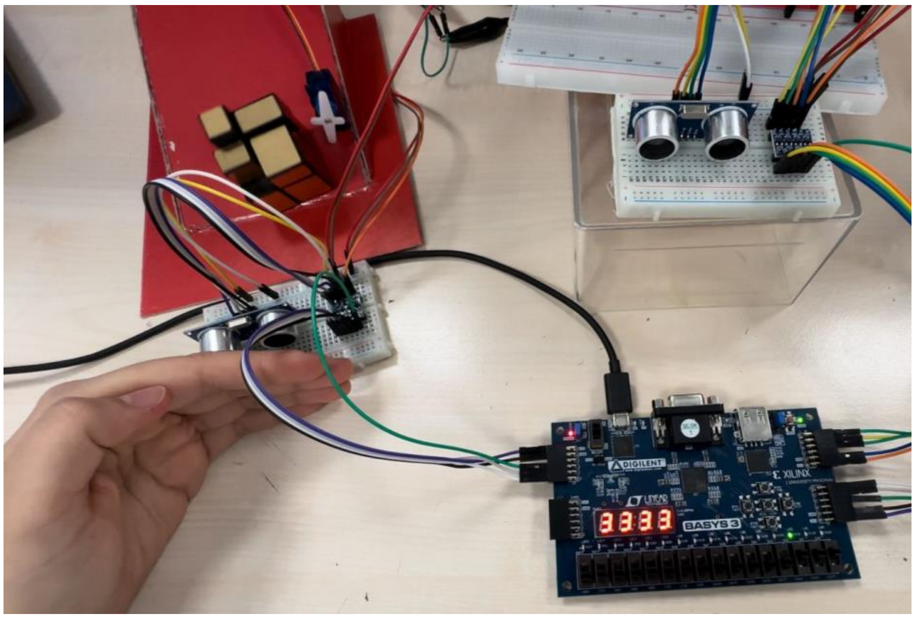

Figure.11 Demonstration of lane 1 (switch is ON)

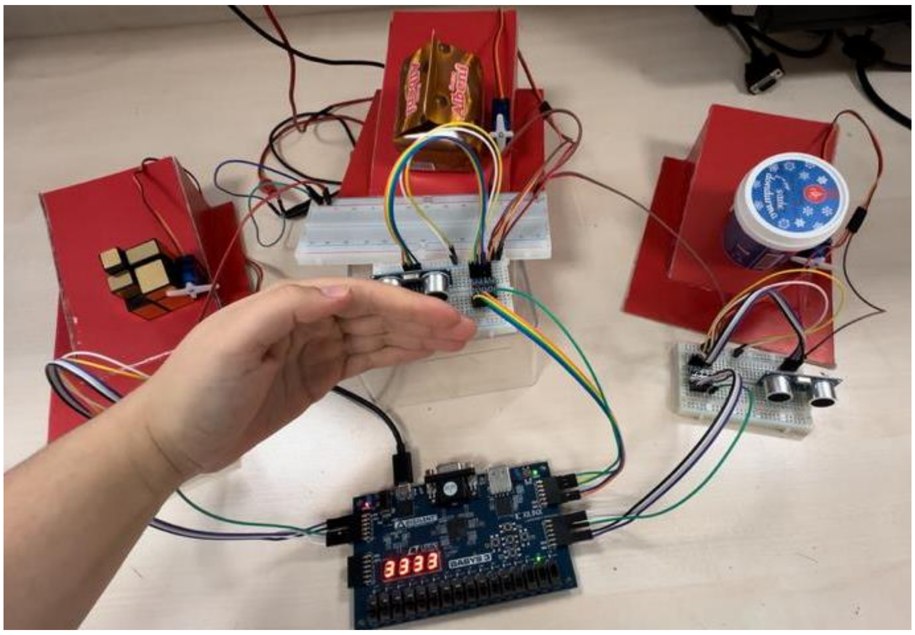

Figure.12 Demonstration of lane 2

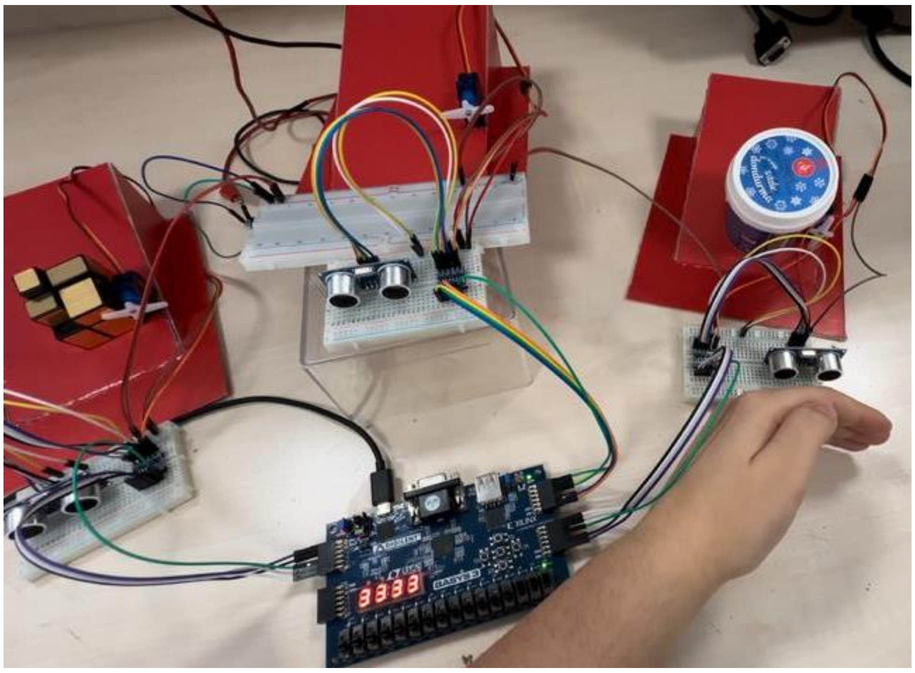

Figure.13 Demonstration of lane 3

For the next figure, lane 1 is disabled by switching off the switch W16. This change caused the seven-segment display to change from 3 to 2. Also, now no matter what you do, lane one will never hand out products. Notice the corresponding LED is off.

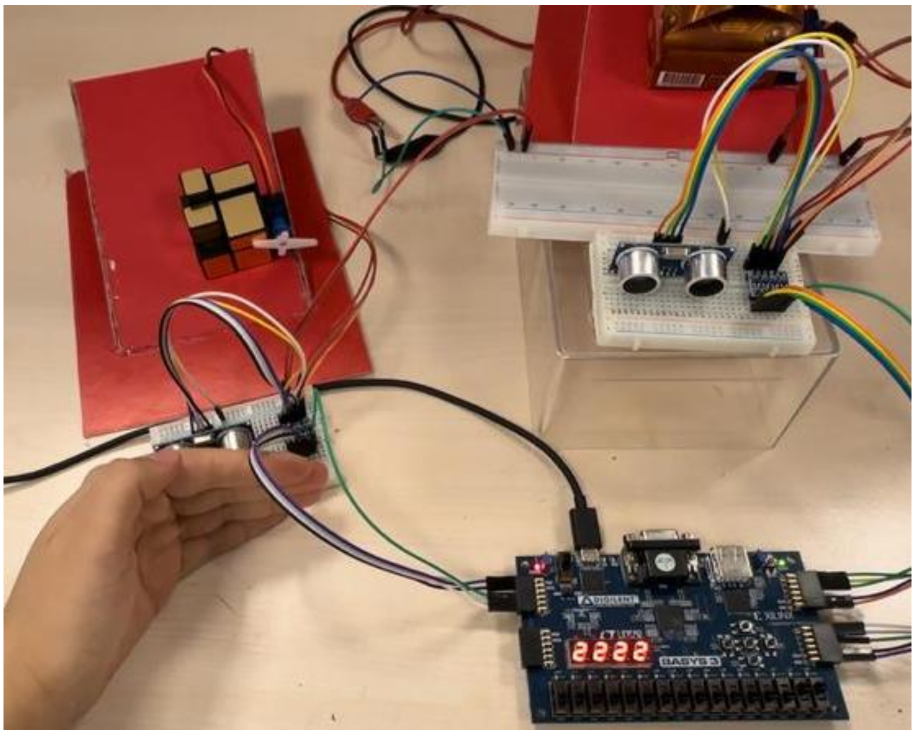

Figure.14 Demonstration of lane 1 (switch is OFF)

Conclusion:

In this project, aim was to implement a vending machine that could be used as a pantry organization system. In order to do that, a BASYS3 FPGA board is used. All of the codes are written in VHDL. External components (ultrasonic sensor HC-SR04 and servo motor SG90) are also used. The voltage supply in the lab is utilized to provide 5V DC to the circuits. The switches and the LEDs on the board are used. The project can be improved by creating closed boxes instead of using ramps and increasing the number of the lanes.

The most difficult part of this project was to write the code that created PWM pulses. The sources on the Web were not working and I could not figure out how to write the code from scratch. At the end, I combined two different Verilog codes and converted that to VHDL. Also, in the first project demo I used a different ultrasonic sensor code. That code showed the calculated distance on the seven-segment display. Since I was not going to show that value in the final, I wrote a different code. This code is not included in this document. This project helped me to implement what we learned in the class. I also included a cost table to show how much (in average) it cost to do a similar system. All in all, the project was successful.

Video Presentation:

https://youtu.be/CFh6LGy2-8E

References:

- https://lastminuteengineers.com/arduino-sr04-ultrasonic-sensor-tutorial/#:~:text=An%20HC%2DSR04%20ultrasonic%20distance,listens%20for%20the%20transmitted%20pulses.
- https://robocraze.com/blogs/post/all-about-servo-motor-sg90
- https://alchitry.com/servos-verilog
- https://github.com/bunyaminarslan/fpga-radar-project-with-basys3/tree/master
- https://github.com/josh-macfie/Verilog_Servo_Control
- https://github.com/matheustguimaraes/ultrars
- https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf
- http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf
- https://digilent.com/reference/_media/basys3:basys3_rm.pdf

Appendix:

- **constraints.xdc**

```
set_property PACKAGE_PIN W5 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports clk]

set_property PACKAGE_PIN J1 [get_ports echo0]
set_property PACKAGE_PIN A14 [get_ports echo1]
set_property PACKAGE_PIN K17 [get_ports echo2]
set_property IOSTANDARD LVCMOS33 [get_ports echo0]
set_property IOSTANDARD LVCMOS33 [get_ports echo1]
set_property IOSTANDARD LVCMOS33 [get_ports echo2]

set_property PACKAGE_PIN L2 [get_ports trig0]
set_property PACKAGE_PIN A16 [get_ports trig1]
set_property PACKAGE_PIN M18 [get_ports trig2]
set_property IOSTANDARD LVCMOS33 [get_ports trig0]
set_property IOSTANDARD LVCMOS33 [get_ports trig1]
set_property IOSTANDARD LVCMOS33 [get_ports trig2]

set_property PACKAGE_PIN J2 [get_ports servo1]
set_property PACKAGE_PIN B15 [get_ports servo2]
set_property PACKAGE_PIN N17 [get_ports servo3]
set_property IOSTANDARD LVCMOS33 [get_ports servo1]
set_property IOSTANDARD LVCMOS33 [get_ports servo2]
set_property IOSTANDARD LVCMOS33 [get_ports servo3]

set_property PACKAGE_PIN W16 [get_ports {switch[0]}]
set_property PACKAGE_PIN V16 [get_ports {switch[1]}]
set_property PACKAGE_PIN V17 [get_ports {switch[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {switch[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {switch[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {switch[2]}]
set_property PACKAGE_PIN U19 [get_ports {led[0]}]
set_property PACKAGE_PIN E19 [get_ports {led[1]}]
set_property PACKAGE_PIN U16 [get_ports {led[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[2]}]

set_property PACKAGE_PIN W7 [get_ports {display[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {display[6]}]
set_property PACKAGE_PIN W6 [get_ports {display[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {display[5]}]
set_property PACKAGE_PIN U8 [get_ports {display[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {display[4]}]
set_property PACKAGE_PIN V8 [get_ports {display[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {display[3]}]
set_property PACKAGE_PIN U5 [get_ports {display[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {display[2]}]
set_property PACKAGE_PIN V5 [get_ports {display[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {display[1]}]
set_property PACKAGE_PIN U7 [get_ports {display[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {display[0]}]
```

- **design.vhd**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity design is
  Port (clk: in std_logic;
    echo0: in std_logic;
    echo1: in std_logic;
    echo2: in std_logic;
    trig0: buffer std_logic;
    trig1: buffer std_logic;
    trig2: buffer std_logic;
    servo1: out std_logic;
    servo2: out std_logic;
    servo3 : out std_logic;
    led : out std_logic_vector(2 downto 0);
    display : out std_logic_vector(6 downto 0);
    switch : in std_logic_vector(2 downto 0));
end design;
architecture Behavioral of design is
  signal loc1, loc2, loc3: std_logic_vector(6 downto 0);
  signal d1: std_logic_vector(2 downto 0);
  signal d2: std_logic_vector(2 downto 0);
  signal d3: std_logic_vector(2 downto 0);
  signal counter0: integer := 0;
  signal counter1: integer := 0;
  signal counter2: integer := 0;
begin
  distance0: entity work.distance(Behavioral) port map(clk => clk, echo => echo0, trig => trig0,
d => d1);
  distance1: entity work.distance(Behavioral) port map(clk => clk,echo => echo1,trig => trig1,d
=> d2);
```

```
  distance2: entity work.distance(Behavioral) port map(clk => clk,echo => echo2,trig => trig2,d
=> d3);
    ssd1: entity work.ssd(Behavioral) port map(switch => switch, display => display);
  SERVO_1: entity work.servo(Behavioral) port map(clk => clk,pos => loc1,servo0 => servo1);
  SERVO_2: entity work.servo(Behavioral) port map(clk => clk,pos => loc2,servo0 => servo2);
  SERVO_3: entity work.servo(Behavioral) port map(clk => clk,pos => loc3,servo0 => servo3);
  process(clk, switch)
  begin
  if switch(0) = '0' then
    if rising_edge(clk) then
      if d1 = "001" then
        if counter0 < 1000000 then
          counter0 <= counter0 + 1;
          loc1 <= "0100000";
          led(0) <= '1';
        else
          counter0 <= 0;
          loc1 <= "0000000";
          led(0) <= '0';
        end if;
      else
        loc1 <= "0000000";
        counter0 <= 0;
        led(0) <= '0';
      end if;
    end if;
  else
    loc1 <= "0000000";
    led(0) <= '0';
  end if;
  if switch(1) = '0' then
    if rising_edge(clk) then
      if d2 = "001" then
        if counter1 < 1000000 then
          counter1 <= counter1 + 1;
          loc2 <= "0100000";
          led(1) <= '1';
        else
          counter1 <= 0;
          loc2 <= "0000000";
          led(1) <= '0';
        end if;
      else
        loc2 <= "0000000";
        counter1 <= 0;
        led(1) <= '0';
      end if;
    end if;
  else
    loc2 <= "0000000";
    led(1) <= '0';
  end if;
  if switch(2) = '0' then
    if rising_edge(clk) then
      if d3 = "001" then
        if counter2 < 1000000 then
          counter2 <= counter2 + 1;
          loc3 <= "0100000";
          led(2) <= '1';
        else
          counter2 <= 0;
          loc3 <= "0000000";
          led(2) <= '0';
        end if;
      else
        loc3 <= "0000000";
        counter2 <= 0;
        led(2) <= '0';
      end if;
    end if;
  else
    loc3 <= "0000000";
    led(2) <= '0';
  end if;
  end process;
end Behavioral;
```

- **distance.vhd**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
entity distance is
  Port (clk: in std_logic;
        echo: in std_logic;
        trig: buffer std_logic;
        d : buffer std_logic_vector (2 downto 0));
end distance;
architecture Behavioral of distance is
signal ech: integer:=0;
begin
process(clk, echo)
variable a,b:integer:=0;
variable y :std_logic:='0';
begin
if rising_edge(clk) then
if(a=0) then
    trig<='1';
elsif(a=10000) then
    trig<='0';
    y:='1';
elsif(a=10000000) then
    a:=0;
    trig<='1';
end if;
a:=a+1;
if(echo = '1') then
    b:=b+1;
end if;
```

```
if(echo = '0' and y='1') then
    ech<= b;
    b:=0;
    y:='0';
end if;
if(ech < 50000) then
    d <= "001";
elsif (50000 < ech) and (ech < 280000) then
    d<= "011";
elsif (280000 < ech) then
    d <= "110";
end if;
end if;
end process;
end Behavioral;
```

- **ssd.vhd**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity ssd is
  Port (display: out std_logic_vector(6 downto 0);
        switch: in std_logic_vector(2 downto 0));
end ssd;
architecture Behavioral of ssd is
begin
process(switch)
begin
 case switch is
 when "000" => display <= "0000001"; -- "0"
 when "001" => display <= "1001111"; -- "1"
 when "010" => display <= "1001111"; -- "1"
 when "011" => display <= "0010010"; -- "2"
 when "100" => display <= "1001111"; -- "1"
 when "101" => display <= "0010010"; -- "2"
 when "110" => display <= "0010010"; -- "2"
 when "111" => display <= "0000110"; -- "3"
 when others => display <= "0111000"; -- F
 end case;
end process;
end Behavioral;
```

servo.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity servo is
 PORT(clk : IN STD_LOGIC;
 pos : IN STD_LOGIC_VECTOR(6 downto 0);
 servo0: OUT STD_LOGIC);
end servo;
architecture Behavioral of servo is
 signal clk_out : STD_LOGIC := '0';
begin
 clk64kHz_map: entity work.clkdiv(behavioral) PORT MAP(clk, clk_out);
 pwm_map: entity work.pwm(Behavioral) PORT MAP(clk_out, pos, servo0);
 end Behavioral;
```

- **clkdiv.vhd**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity clkdiv is
 Port (clk : in STD_LOGIC;
 clk_out: out STD_LOGIC);
end clkdiv;
architecture Behavioral of clkdiv is
 signal temp: STD_LOGIC;
 signal counter : integer range 0 to 780 := 0;
begin
 freq_divider: process (clk) begin
 temp <= '0';
 counter <= 0;
 if rising_edge(clk) then
 if (counter = 780) then
 temp <= NOT(temp);
 counter <= 0;
 else
 counter <= counter + 1;
 end if;
 end if;
 end process;
 clk_out <= temp;
end Behavioral;
```

- **pwm.vhd**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity pwm is
 PORT (clk : IN STD_LOGIC;
 pos : IN STD_LOGIC_VECTOR(6 downto 0);
 servo0 : OUT STD_LOGIC);
end pwm;
architecture Behavioral of pwm is
 signal c : unsigned(10 downto 0);
 signal i: unsigned(7 downto 0);
begin
 i <= unsigned('0' & pos) + 16;
 process (clk) begin
 if rising_edge(clk) then
 if (c = 1279) then
 c <= (others => '0');
 else
 c <= c + 1;
 end if;
 end if;
 end process;
 servo0 <= '1' when (c < i) else '0';
end Behavioral;
```