EE-102 Introduction to Digital Circuit Design
Kaan Gürsoy 21704120
Section: 4 Project Report 31.05.2021

Targeting System

Abstract:

The purpose of this project is designing a targeting system by using BASYS 3 FPGA board and Vivado VHDL.

Design Specification Plan:

Equipments:

1- BASYS 3

It is the most important part of the project. It is an FPGA board that make user able to design projects. It is coded by VHDL in this project and provide working with servos and displaying image on monitor via VGA cable.

2- SG90 Servo Motor

Servo motor works with 20 ms (50 Hz) signal and by generating 1-2 ms duty cycle it can be rotated as it is desired. Also, it needs 4,8- 5V DC to work.

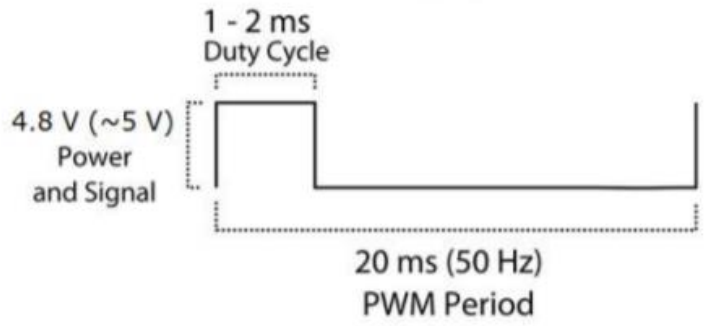

Figure 1: Necessary wave to work with servos

The signal in figure 1 is designed in Vivado and provided by BASYS 3 to servo.

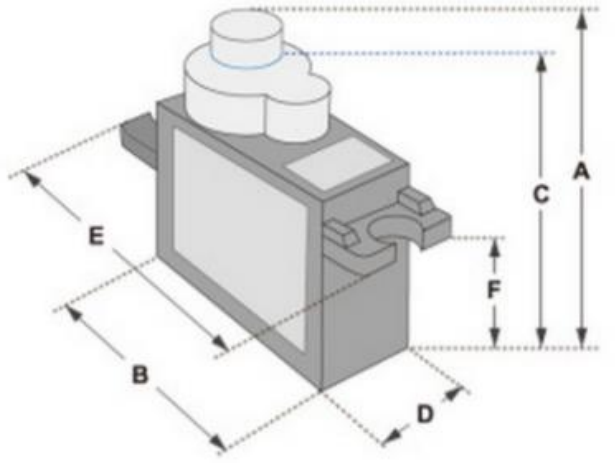

Figure 2: SG90 Servo and pulse-position examples.

3- VGA Cable

This provides connection between monitor and BASYS 3 to displaying desired image (in this project word “FIRE”)

4- Monitor

Image is displayed on this device.

5- 3 piece of LDR sensors (Light sensors)

Light sensors gives input ‘1’ when the quantity of light is reduced. The sensitivity can be adjusted via screwdriver. It need 3.3V to work.

6- Glove

Light sensors is attached to glove to control fire process better.

7- Jumper Cables

It provides connection between components.

8- Weapon model

It is attached to servo to construct more accurate and pleasant model.

9- Voltage Generator

It provides 5V DC to servo motor.

In this project, servo motor controlled via BASYS 3 and weapon model is attached to servo motor. Subsequently, the direction of weapon is controlled by BASYS 3, which is coded by VHDL. Seven switch of the BASYS 3 control the servo and when weapon’s aim is adjusted, it is fired by three piece of LDR sensors, attached to the glove. When the luminous flux (quantity of light) on all LDR sensors decrease, weapon open fire imaginatively. To show that weapon open fire, when fire director is given by the light sensors, “FIRE” word appears on monitor’s screen, which is connected to the BASYS 3 via VGA cable.

Project Design Methodology:

Project consist of two phase. The first phase is displaying the “FIRE” word on the screen. In order to perform this, 2 modules are created by VHDL in the name of “draw” and “colourgenerator”. “colourgenerator” module is created due to scanning the monitor. It scans the monitor from up to down and right to left by counters. Also, it restricts the limit of the monitor so that we can give a color and shape appropriately. After giving monitor a solid black screen, the shape and color have been given by “draw” module. “draw” module writes the ”FIRE” word by using color (it has been chosen as red in this project). By coloring between the bits using AND and OR gates, “FIRE” word is created on screen. Also, in “draw” module it is coded that “when the all sensors on glove is closed, generates the word “FIRE””.

The second phase is working with servo motor. To adjust the servo motor, three modules is created. Servo motor requires 20 ms pulse signal with 1-2 ms duty cycle. In order to provide that in first module “pulseconverter”, 50 MHz signal that BASYS 3 generates is converted to 64 kHz signal. This module is made for generating 1-2 ms duty cycle. After that, “servoo” module is created and generating 20 ms signal with 1-2 ms cycle according to switches is programmed. These two module is unified in module “servo_pwm_clk64kHz”. This module, “draw” module, “colourgenerator” module is unified in “topmodule”.

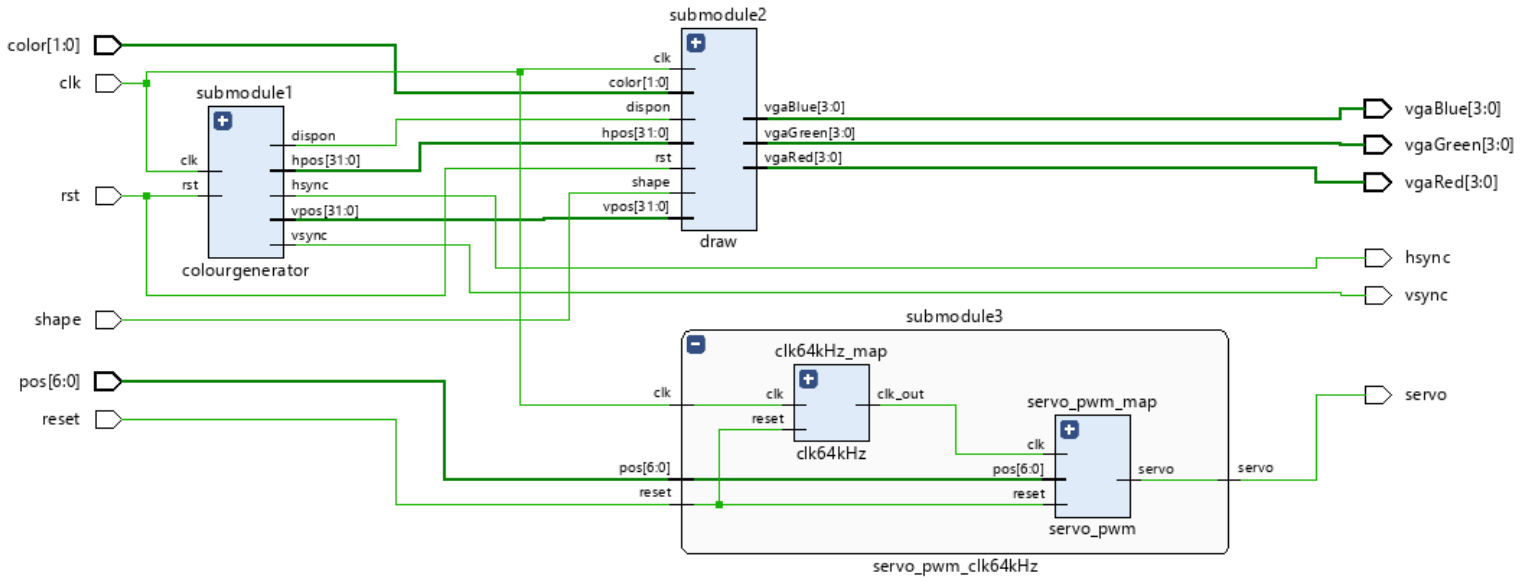

Figure 3: Schematic of the Project

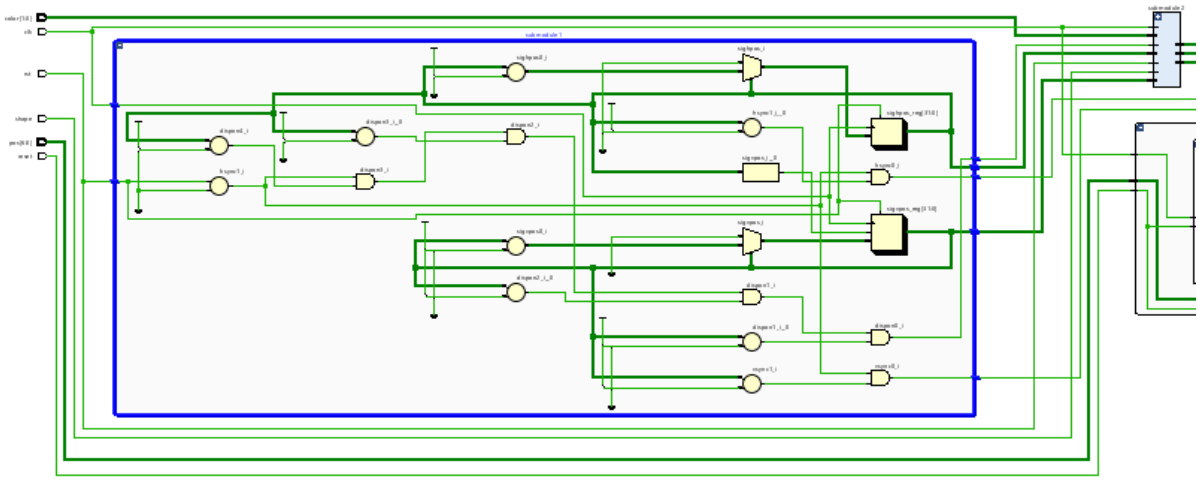

Figure 4: Schematic of the submodule 1 “colourgenerator”

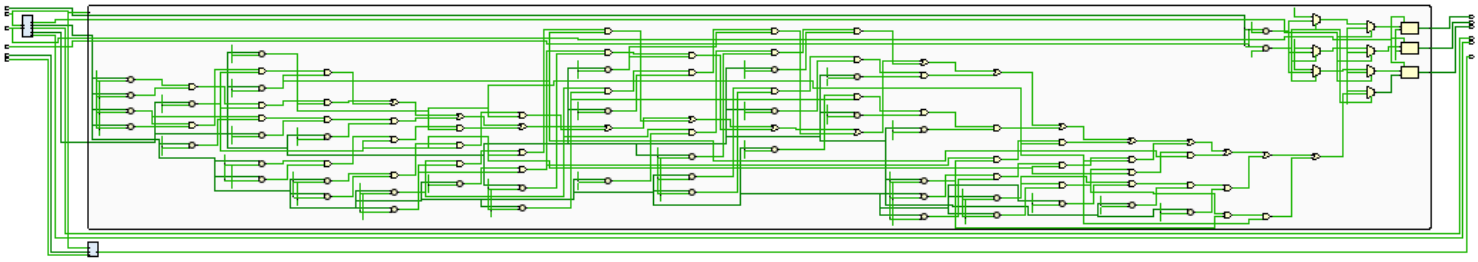

Figure 5: Schematic of the submodule 2 “draw”

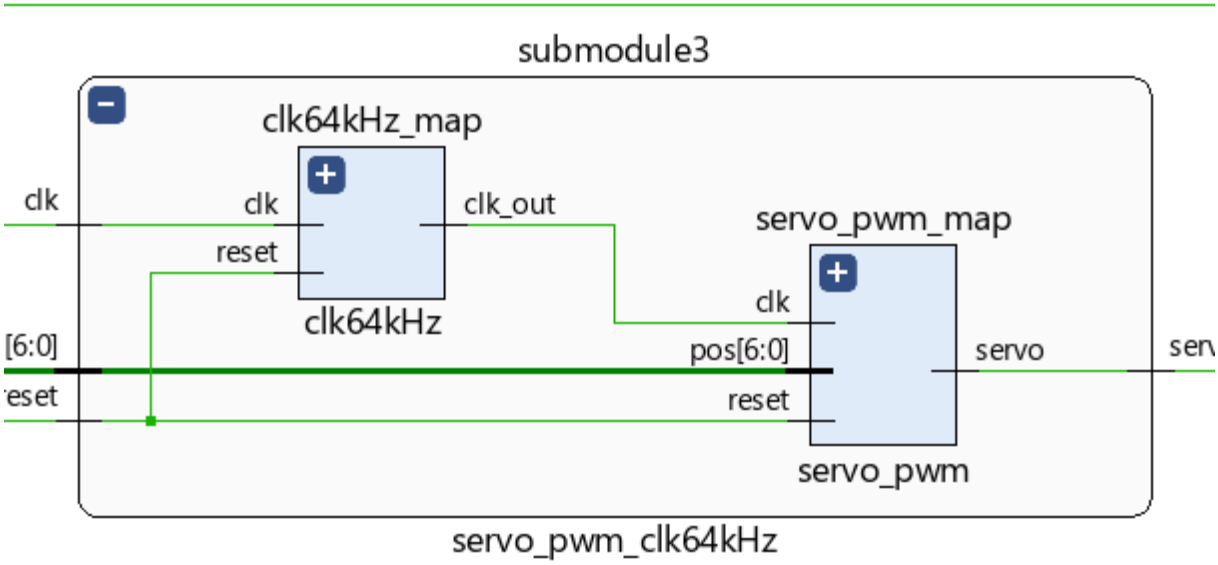

Figure 6: Schematic of the submodule 3 “servo_pwm_clk64kHz”

As it can be seen, submodule 3 consist of two modules “pulseconverter” and “servoo” and output of “pulseconverter” (64 kHz pulse signal) is used as an input by “servoo”. It is demonstrated in figure 7 and figure 8, this submodule, generates the clock signal and duty cycle by counters temporary registers. Also, it provides user to generate desired duty cycle via switches on BASYS 3.

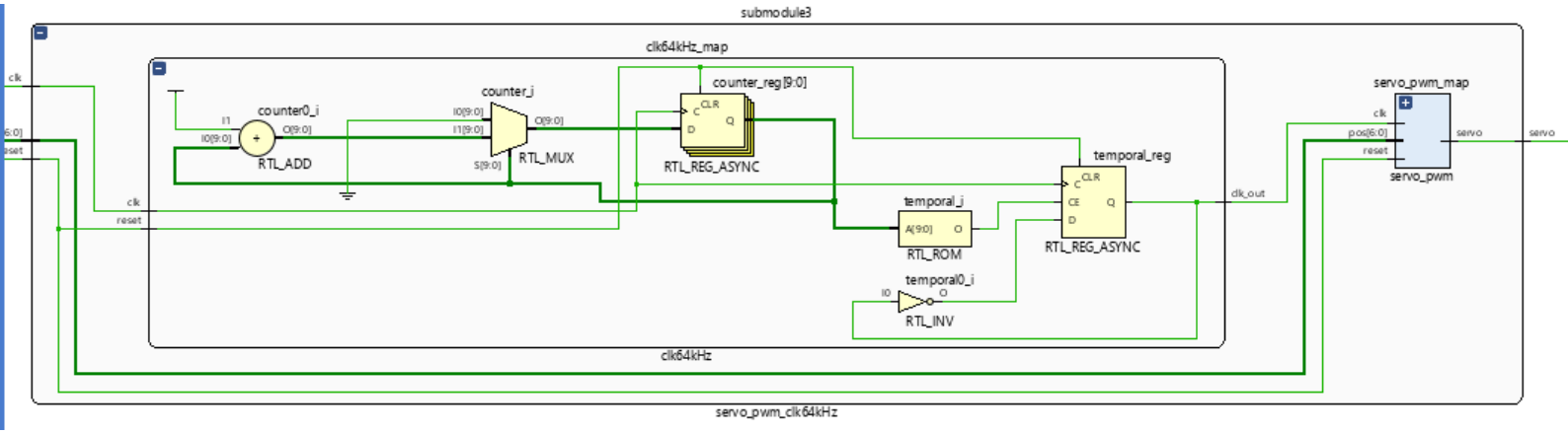

Figure 7: Schematic of “pulseconverter”

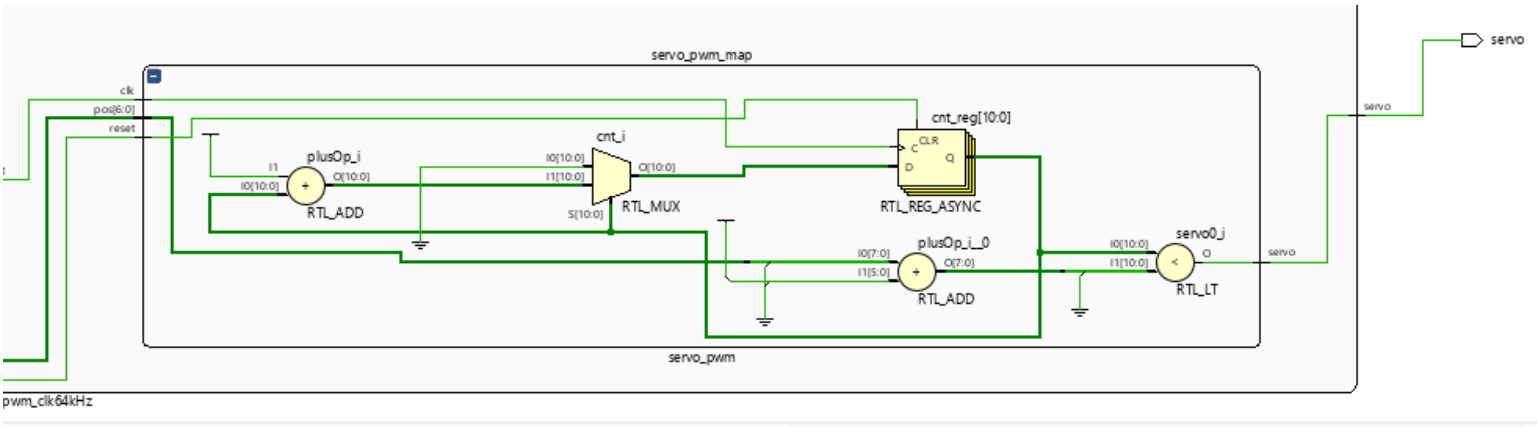

Figure 8: Schematic of “servoo”

After designing all the digital circuit, constrain code is constructed to finish the project.

Results:

After designing the coding part of the project, model is constructed. All components are attached with jumpers and all of them is attached to BASYS 3 via PMODs.

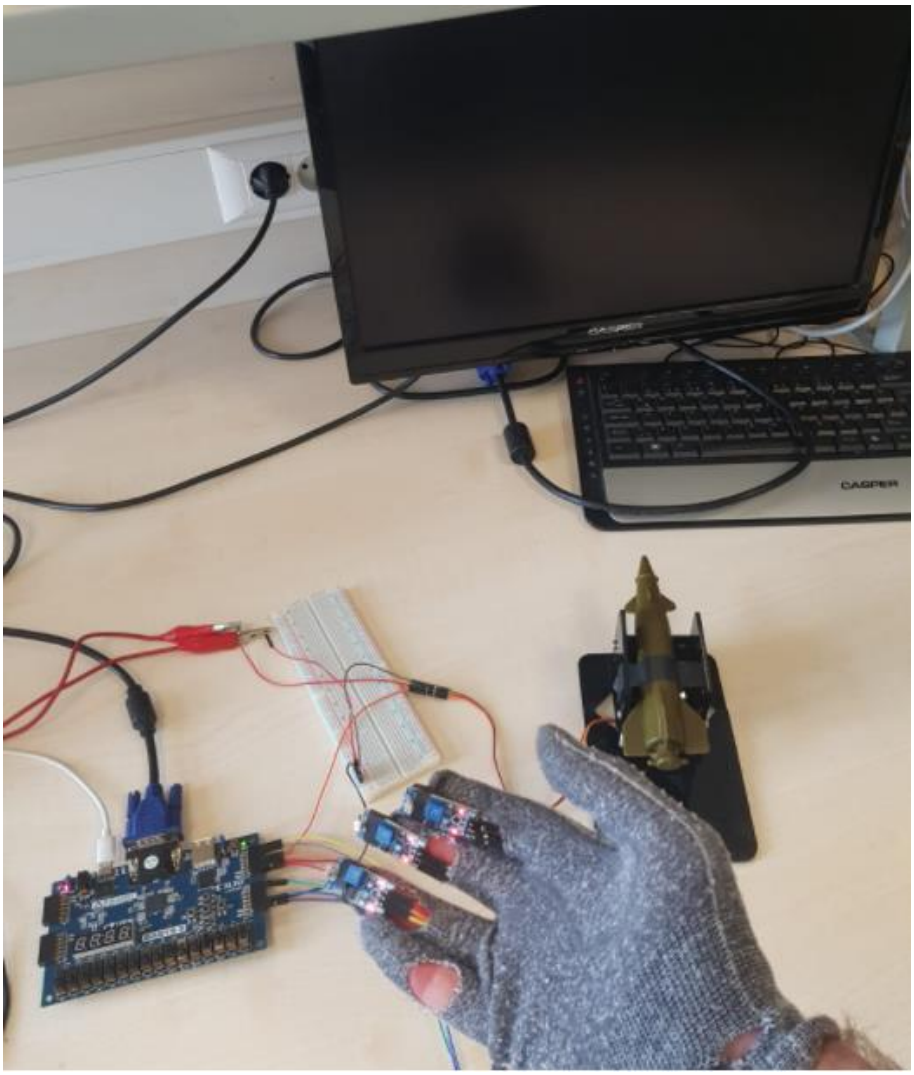

Figure 9: Targeting System Projects

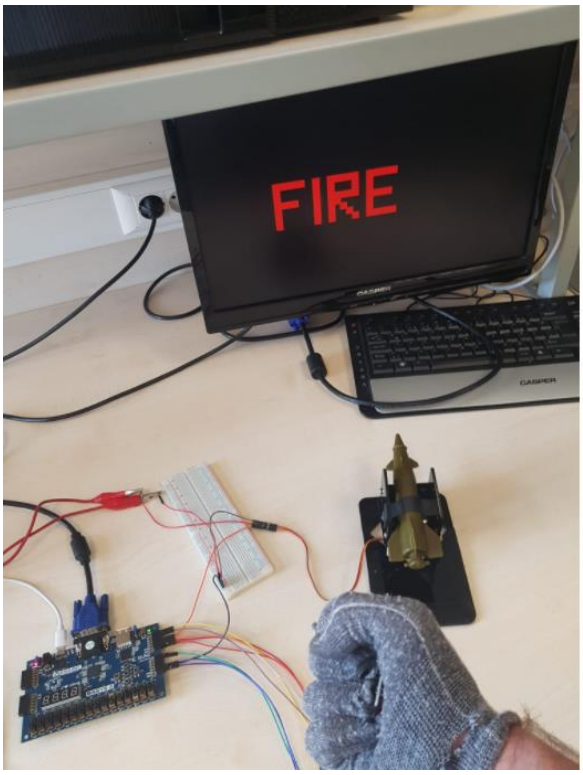

Figure 10: Targeting System Projects

As it can be seen in figure 9 and figure 10, firing process occurs when user makes his hand fist. When user prevent light sensor to be exposed to the light, firing process is occurred and to show this displaying the word “FIRE” is showed in the screen.

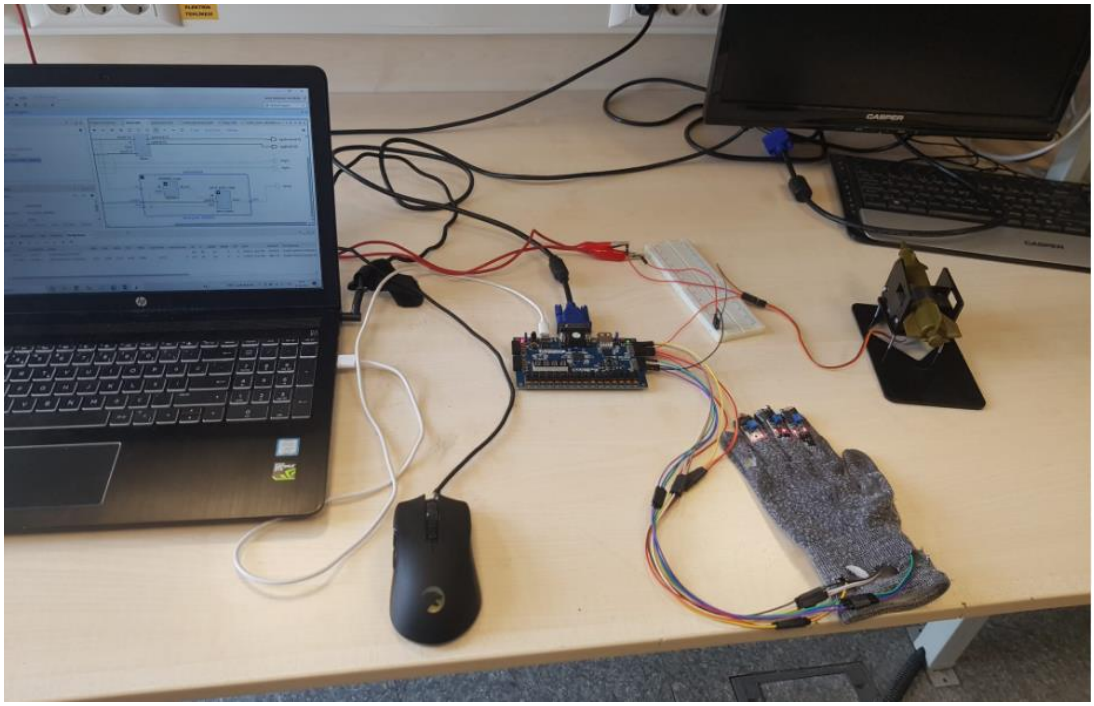

Figure 11: Targeting System Projects

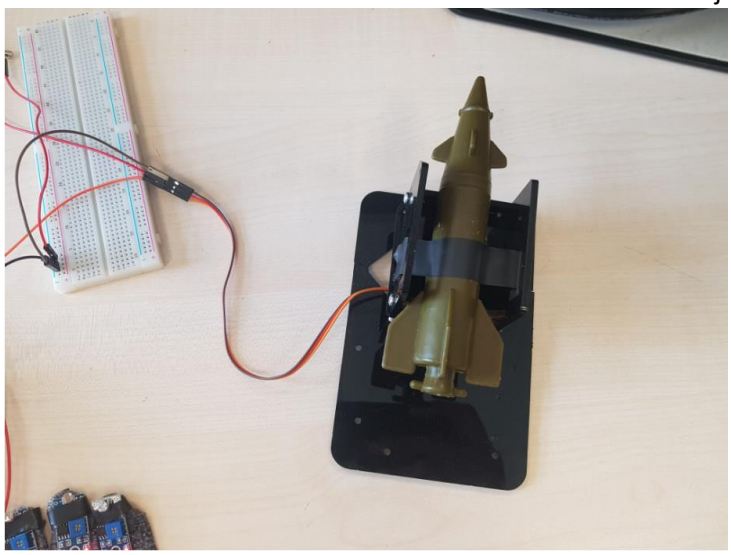

Figure 12: Weapon Model

Figure 13: Weapon Model (rotated left)

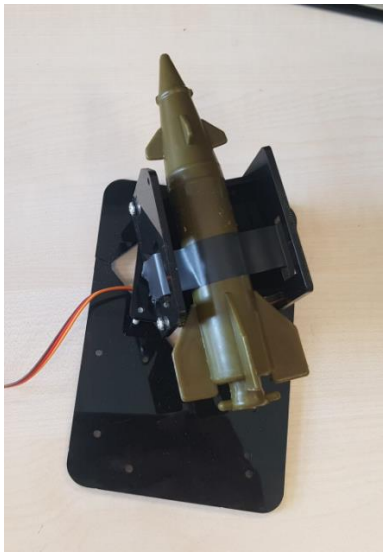

Figure 14: Weapon Model (rotated a little bit left)

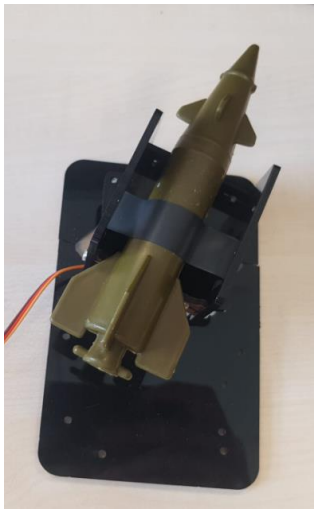

Figure 15: Weapon Model (rotated right)

Figure 16: Voltage generator

As it can be seen in figure 12, figure 13 and figure 14, servo motor is controlled as a result of the project. Seven switched controls the servo so 128 combination of rotation exists in the project.

Figure 17: “FIRE” word on the screen

All results are accurate and expected in design plan. Servo motor is rotated as the user wants. 128 combination of rotation is tested by opening and closing switches on BASYS 3 and acquired accurate results. Displaying the “FIRE” word is accomplished as it can be seen in figure 16.

Conclusion:

The purpose of the project was designing a targeting system by using BASYS 3 and VHDL language. Several and crucial circuit elements are researched and acquired. Working with servo and controlling it by using BASYS3 and VHDL is accomplished. Displaying an image on monitor by using VGA cable is accomplished and VGA output of BASYS3 is researched. VHDL skills are improved and how to design counters, registers, multiplexers are learned. Generating the desired clock signal by any FPGA board by using VHDL is learned.

References

1- Bobrowicz, S. (2018, March 12). *Basys 3 Reference*. Digilent Reference. https://reference.digilentinc.com/basys3/refmanual

2- Ramos, C. A. (2012a, August 20). *Frequency Divider with VHDL*. CODE PROJECT. https://www.codeproject.com/Articles/443644/Frequency-Divider-with-VHDL

3- Ramos, C. A. (2012b, December 20). *Servomotor Control with PWM and VHDL*. CODE PROJECT. https://www.codeproject.com/Articles/513169/Servomotor-Control-with-PWM-and-VHDL

4- *Servo Motor SG-90 Basics, Pinout, Wire Description, Datasheet*. (2017, September 18). Components101. https://components101.com/motors/servo-motor-basics-pinout-datasheet

Appendix:

VHDL Code:

Constrain:

```
set_property PACKAGE_PIN W5 [get_ports clk]

        set_property IOSTANDARD LVCMOS33 [get_ports clk]

        create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]
#reset button
#set_property PACKAGE_PIN V17 [get_ports {rst}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {rst}]
set_property PACKAGE_PIN T1 [get_ports {rst}]
        set_property IOSTANDARD LVCMOS33 [get_ports {rst}]


#VGA Constrains
set_property PACKAGE_PIN G19 [get_ports {vgaRed[0]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {vgaRed[0]}]
   set_property PACKAGE_PIN H19 [get_ports {vgaRed[1]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {vgaRed[1]}]
   set_property PACKAGE_PIN J19 [get_ports {vgaRed[2]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {vgaRed[2]}]
   set_property PACKAGE_PIN N19 [get_ports {vgaRed[3]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {vgaRed[3]}]
```

```
set_property PACKAGE_PIN N18 [get_ports {vgaBlue[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {vgaBlue[0]}]
set_property PACKAGE_PIN L18 [get_ports {vgaBlue[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {vgaBlue[1]}]
set_property PACKAGE_PIN K18 [get_ports {vgaBlue[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {vgaBlue[2]}]
set_property PACKAGE_PIN J18 [get_ports {vgaBlue[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {vgaBlue[3]}]
set_property PACKAGE_PIN J17 [get_ports {vgaGreen[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {vgaGreen[0]}]
set_property PACKAGE_PIN H17 [get_ports {vgaGreen[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {vgaGreen[1]}]
set_property PACKAGE_PIN G17 [get_ports {vgaGreen[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {vgaGreen[2]}]
set_property PACKAGE_PIN D17 [get_ports {vgaGreen[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {vgaGreen[3]}]
set_property PACKAGE_PIN P19 [get_ports hsync]
    set_property IOSTANDARD LVCMOS33 [get_ports hsync]
set_property PACKAGE_PIN R19 [get_ports vsync]
    set_property IOSTANDARD LVCMOS33 [get_ports vsync]
```

```
#PMOD SENSORS
```

```
##Pmod Header JB

##Sch name = JB1

set_property PACKAGE_PIN A14 [get_ports {color[1]}]

        set_property IOSTANDARD LVCMOS33 [get_ports {color[1]}]

##Sch name = JB2

set_property PACKAGE_PIN A16 [get_ports {color[0]}]

        set_property IOSTANDARD LVCMOS33 [get_ports {color[0]}]


 ##Pmod Header JC

##Sch name = JC1

set_property PACKAGE_PIN K17 [get_ports {shape}]

        set_property IOSTANDARD LVCMOS33 [get_ports {shape}]


        ##Sch name = JB4

set_property PACKAGE_PIN B16 [get_ports {sss}]

    set_property IOSTANDARD LVCMOS33 [get_ports {sss}]


#reset button

set_property PACKAGE_PIN R2 [get_ports {reset}]

        set_property IOSTANDARD LVCMOS33 [get_ports {reset}]


# Switches

set_property PACKAGE_PIN V17 [get_ports {roate[0]}]

        set_property IOSTANDARD LVCMOS33 [get_ports {rotate[0]}]
```

```
set_property PACKAGE_PIN V16 [get_ports {pos[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {rotate[1]}]
set_property PACKAGE_PIN W16 [get_ports {rotate[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {rotate[2]}]
set_property PACKAGE_PIN W17 [get_ports {rotate[3]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {rotate[3]}]
set_property PACKAGE_PIN W15 [get_ports {rotate[4]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {rotate[4]}]
set_property PACKAGE_PIN V15 [get_ports {rotate[5]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {rotate[5]}]
set_property PACKAGE_PIN W14 [get_ports {rotate[6]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {rotate[6]}]
```

Topmodule:

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


entity topmodule is
Port ( clk : in STD_LOGIC;
      rotate  : IN  STD_LOGIC_VECTOR(6 downto 0);
      rst : in STD_LOGIC;
      Green : out STD_LOGIC_VECTOR (3 downto 0);
      shape : in STD_LOGIC;
      Red : out STD_LOGIC_VECTOR (3 downto 0);
```

```
        veseync : out STD_LOGIC;
        color : in STD_LOGIC_VECTOR(1 downto 0);
        heseync : out STD_LOGIC;



        Blue : out STD_LOGIC_VECTOR (3 downto 0);
        reset: IN  STD_LOGIC;

        sss: OUT STD_LOGIC);



end topmodule;

architecture Behavioral of topmodule is

component colourgenerator is
Port ( clk: in std_logic;
        rst: in std_logic;
        hepeos: out integer;
        vepos: out integer;
        heseync: out std_logic;
        veseync: out std_logic;
        disepon: out std_logic);
```

```
end component;



component draw is
Port (clk: in std_logic;
      rst: in std_logic;
      hepos: in integer;
      vepos: in integer;
      disepon: in std_logic;
      shape : in STD_LOGIC;
      color : in STD_LOGIC_VECTOR(1 downto 0);
      Red: out std_logic_vector(3 downto 0);
      Green: out std_logic_vector(3 downto 0);
      Blue: out std_logic_vector(3 downto 0));

end component;
component servo_pwm_clk64kHz is
    PORT(
        clk  : IN  STD_LOGIC;
        reset: IN  STD_LOGIC;
        rotate  : IN  STD_LOGIC_VECTOR(6 downto 0);
        sss: OUT STD_LOGIC
    );
end component;
```

```
signal hpos: integer:= 0;

signal vpos: integer:= 0;

signal dispon: std_logic;


begin


submodule1: colourgenerator port map(clk => clk, rst => rst , hepos =>hepos , vepos =>
vepos , heseync => heseync , veseync => veseync , disepon => disepon );

submodule2: draw port map(clk => clk, rst => rst, hepos => hepos, vepos => vepos, dispon
=> disepon, color => color, shape => shape, Red => Red, Blue => Blue, Green => Green);

submodule3: servo_pwm_clk64kHz port map(clk => clk, reset => reset , rotate =>rotate , sss
=> sss);

end Behavioral;
```

“colourgenerator” Module:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

use IEEE.NUMERIC_STD.ALL;


entity colourgenerator is


Port ( clk : in STD_LOGIC;

       rst : in STD_LOGIC;

       heepos : out integer;

       veepos : out integer;
```

```
        heseync : out STD_LOGIC;
        veseync : out STD_LOGIC;
        disepon : out STD_LOGIC);
end colourgenerator;

architecture Behavioral of colourgenerator is
-- right to left


constant hedisp: integer:= 1440;
constant hefp: integer:= 80;
constant hesep: integer:= 152;
constant hebp: integer:= 232;
--down to up
constant vedisp: integer:= 900;
constant vefep: integer:= 1;
constant vesep: integer:= 3;
constant vebep: integer:= 28;
-- for casper monitor

signal sighepos: integer:= 0;
signal sigvepos: integer:= 0;

begin
horiz_pos_cnt:process(clk, rst)
```

```
begin


if(rst = '1')then

sighepos <= 0;

sigvepos <= 0;


elsif(clk'event and clk = '1')then


if(sighepos = hedisp + heefp + heesp + heebp)then

sighepos <= 0;

if (sigvepos = vdisp + vfp + vsp + vbp) then

sigvepos <= 0;

else

sigvepos <= sigvepos + 1;

end if;

else

sighepos <= sighepos + 1;

end if;

end if;

end process;

-- It scans all of the monitor bit by bit

heseyenc <= '1' when rst = '0' AND sighepos >= hesep else '0';

veseyenec <= '1' when rst = '0' AND sigvepos >= vesep else '0';

disepon <= '1' when rst = '0' AND sighepos >= hesep + hbp AND sighpos < hesep + hebep +
hedisp AND sigvepos >= vsp + vbp AND sigvepos < vesep + vbp + vdeisp else '0';
```

```
hepos <= sighepos;

vepos <= sigvepos;



end Behavioral;
```

"draw" Module:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

use IEEE.numeric_std.ALL;


entity draw is

Port ( clk : in std_logic;

rst : in STD_LOGIC;

   hpos : in integer;

   vpos : in integer;


   imagebit : in STD_LOGIC;

   imagepart : in STD_LOGIC_VECTOR(1 downto 0);

   dispon : in STD_LOGIC;

   Red : out STD_LOGIC_VECTOR (3 downto 0);

   Green : out STD_LOGIC_VECTOR (3 downto 0);

   Blue : out STD_LOGIC_VECTOR (3 downto 0));
```

```
end draw;

architecture Behavioral of draw is
signal last_point: integer range 0 to 4:= 0;

signal w: STD_LOGIC_VECTOR(2 downto 0);
--horizontal constants
constant hdisp: integer:= 1440;
constant hsp: integer:= 152;
constant hbp: integer:= 232;


constant vdisp: integer:= 900;
constant vsp: integer:= 3;
constant vbp: integer:= 28;
```

```
begin

w <= imagebit & imagepart;

process(rst, dispon, hpos, vpos, W)

variable pixel : integer range 0 to 1000;

begin

if(rst = '1')then

Red <= "0000";

Green <= "0000";

Blue <= "0000";

elsif(dispon = '1')then


      if (W = "111") then


          if(hpos > 700 AND hpos < 730 AND vpos > 400 AND vpos < 600) OR (hpos > 725 AND hpos < 830 AND vpos > 400 AND vpos <430) OR (hpos > 725 AND hpos < 800 AND vpos > 480 AND vpos < 510) OR (hpos > 860 AND hpos < 890 AND vpos > 400 AND vpos < 600) OR (hpos > 920 AND hpos < 950 AND vpos > 400 AND vpos < 600) OR (hpos > 945 AND hpos < 1050 AND vpos > 400 AND vpos < 430) OR (hpos > 1020 AND hpos < 1050 AND vpos > 425 AND vpos < 500) OR (hpos > 945 AND hpos < 1025 AND vpos > 470 AND vpos < 500) OR (hpos > 945 AND hpos < 975 AND vpos > 500 AND vpos < 525) OR (hpos > 970 AND hpos < 1000 AND vpos > 520 AND vpos < 550) OR (hpos > 995 AND hpos < 1025 AND vpos > 545 AND vpos < 575) OR (hpos > 1020 AND hpos < 1050 AND vpos > 570 AND vpos < 600)OR (hpos > 1080 AND hpos < 1110 AND vpos > 400 AND vpos < 600) OR (hpos > 1105 AND hpos < 1210 AND vpos > 400 AND vpos < 430) OR (hpos > 1105 AND hpos < 1190 AND vpos > 485 AND vpos < 515)OR (hpos > 1105 AND hpos < 1210 AND vpos > 570 AND vpos < 600)then

             Red <= "1111";

              Green <= "0000";

             Blue <= "0000";

          else

              Red <= "0000";
```

```
            Green <= "0000";

            Blue <= "0000";

        end if;

    end if;


    if (W = "000") then

        if(hpos > 520 AND hpos < 540 AND vpos > 300 AND vpos < 330) OR (hpos > 540 AND hpos < 560 AND vpos > 270 AND vpos < 300) OR (hpos > 560 AND hpos < 580 AND vpos > 240 AND vpos < 270) OR (hpos > 580 AND hpos < 600 AND vpos > 210 AND vpos < 240) OR (hpos > 600 AND hpos < 620 AND vpos > 240 AND vpos < 270) OR (hpos > 620 AND hpos < 640 AND vpos > 270 AND vpos < 300) OR (hpos > 640 AND hpos < 660 AND vpos > 300 AND vpos < 330) OR (hpos > 660 AND hpos < 680 AND vpos > 270 AND vpos < 300) OR (hpos > 680 AND hpos < 700 AND vpos > 240 AND vpos < 270) OR (hpos > 700 AND hpos < 720 AND vpos > 210 AND vpos < 240) OR (hpos > 720 AND hpos < 740 AND vpos > 240 AND vpos < 270) OR (hpos > 740 AND hpos < 760 AND vpos > 270 AND vpos < 300) OR (hpos > 760 AND hpos < 780 AND vpos > 300 AND vpos < 330)then

            Red <= "0000";

            Green <= "0000";

            Blue <= "0000";

        else



            Red <= "0000";

            Green <= "0000";

            Blue <= "0000";

        end if;

    end if;




end if;

end process;
```

```
end Behavioral;
```

“servo_pwm_clk64kHz” Module:

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


entity servo_pwm_clk64kHz is
   PORT(
      rotate  : IN  STD_LOGIC_VECTOR(6 downto 0);


      clk  : IN  STD_LOGIC;


      sss: OUT STD_LOGIC
   );
end servo_pwm_clk64kHz;


architecture Behavioral of servo_pwm_clk64kHz is
      COMPONENT servoo
      PORT (
         clk   : IN  STD_LOGIC;
         rotate   : IN  STD_LOGIC_VECTOR(6 downto 0);
         sss : OUT STD_LOGIC
      );
   END COMPONENT;
```

```
    COMPONENT pulseconverter
        PORT(
            clk    : in  STD_LOGIC;
            output: out STD_LOGIC
        );
    END COMPONENT;









    signal output : STD_LOGIC := '0';
begin
    pulseconverter_map: pulseconverter PORT MAP(
        clk, output
    );


    servo_pwm_map: servoo PORT MAP(
        clock, rotate, sss
    );
end Behavioral;
```

“pulseconverter” Module:

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


entity pulseconverter is
    Port (
        clk    : in  STD_LOGIC;
        output: out STD_LOGIC
    );
end pulseconverter;


architecture Behavioral of pulseconverter is
    signal temporary_reg: STD_LOGIC;
    signal counter : integer range 0 to 780 := 0;
begin
    freq_divider: process (clk) begin


        if rising_edge(clk) then
            if (counter = 780) then
                temporary_reg <= NOT(temporary_reg);
                counter  <= 0;
            else
                counter <= counter + 1;
            end if;
        end if;
```

```
    end process;


    output <= temporary_reg;
end Behavioral;
```

“servoo” Module:

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


entity servoo is
    PORT (
        clk   : IN  STD_LOGIC;
        rotate   : IN  STD_LOGIC_VECTOR(6 downto 0);
        sss : OUT STD_LOGIC
    );
end servoo;


architecture Behavioral of servoo is
    signal switches: unsigned(7 downto 0);
    signal count : unsigned(10 downto 0);


begin
    -- Minimum value should be 0.5ms.
    switches <= unsigned('0' & rotate);
```

```
    -- Counter process, from 0 to 1279.
    counter: process (clk) begin
        if rising_edge(clk) then
            if (count = 640) then
                count <= (others => '0');
            else
                count <= count + 1;
            end if;
        end if;
    end process;

    sss <= '1' when (count < switches) else '0';
end Behavioral;
```