Bilkent University Electrical and Electronics Engineering Department

EEE102 Term Project

Engineer-Pong

***N*ame:** Emre Şaş

Section: 2

ID: 22102764

Instructor: Ergin Atalar

Date: 20.12.2023

Implementation of the project:

https://www.youtube.com/watch?v=1iFI331BV7Q&t=1s

Purpose:

The aim of this project is to design a system for the game Beer-Pong using BASYS 3 and IR sensors compatible with BASYS 3. In this project, we will be able to play advanced beer- pong. Our setup will consist of 6 cups in total (Figure 1), and the aim is to throw three balls out of 5 into adjacent 3 cups (Balls must perform a straight line). Thus, at that point, our game differs from the classical game. If the player manages to put three balls in adjacent three cups, they will win.

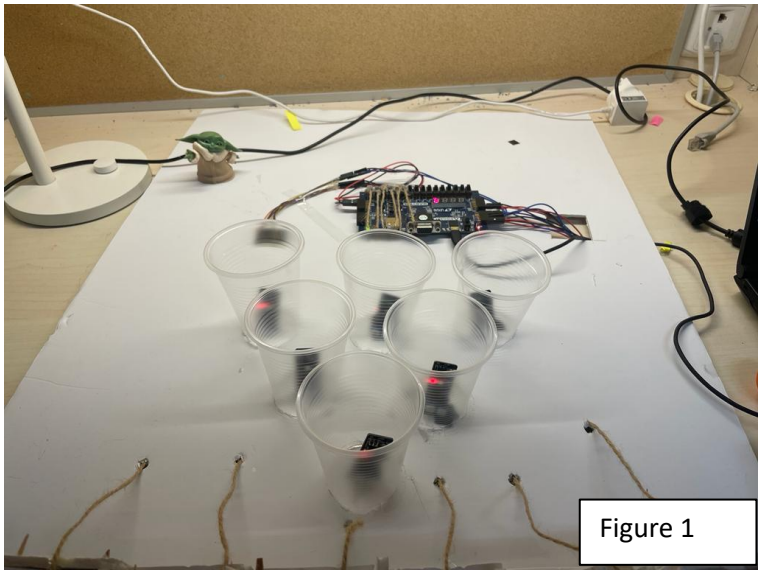


Figure 1

Components:

• 6 Plastic cups

• 6 IR sensors (KY-032)

• BASYS 3

• Jumper Cables and rope

• Ping-pong balls

• Boxes and cable organizers

Methodology:

First aim is to finding an efficient algorithm for the project. Since the project depends on data from switches and sensors the design must surely contain multiplexers to pick the wanted conditions and clocks to drive the seven segment display properly. To prevents players from cheating boxes are locked and it is not possible to open box without triggerring the switch. Therefore system is capable of detecting the unknown objects. When the game is played by the rules and the winning condition is met number '6' will be seen after switching up the last switch. Otherwise letter 'F' will be seen. After writing the code physical structure of the project is set. Boxes, cups,Basys and the sensors are placed onto a surface (Figure 2).

Design specifications:

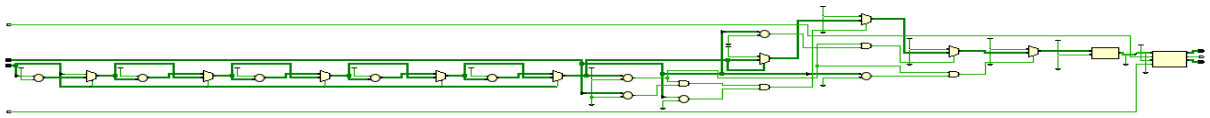
Write a VHDL code by dividing the project into 3 modules which are System where the main algorithm lies, SSD_Driver where we control the seven segment display and the full adder module which is used to make addition of switches ('sw'). Schematics of modules can be seen in below (Figure 2-5)

1. **System module:** The purpose of this module is to create the algorithm for the project. The top module named as System which contains various input and output ports, including clock (clk), switches (sw), joystick inputs (JA), a button (btnC), and outputs for a seven-segment display (seg), display multiplexing control (an), and a decimal point indicator (dp).

    1.1. The code instantiates two components, namely SSD_Driver and Adder. The SSD_Driver is responsible for driving the seven-segment display based on the provided inputs, including the clock signal and the display data (HEX). HEX is a 16 bit long vector. It is divided into 4 (HEX 0, HEX1, HEX2, HEX3) to ease the process of seven segment display. The

Adder component appears in the code, but its instantiation will be done as a seperate module (Module FullAdder).

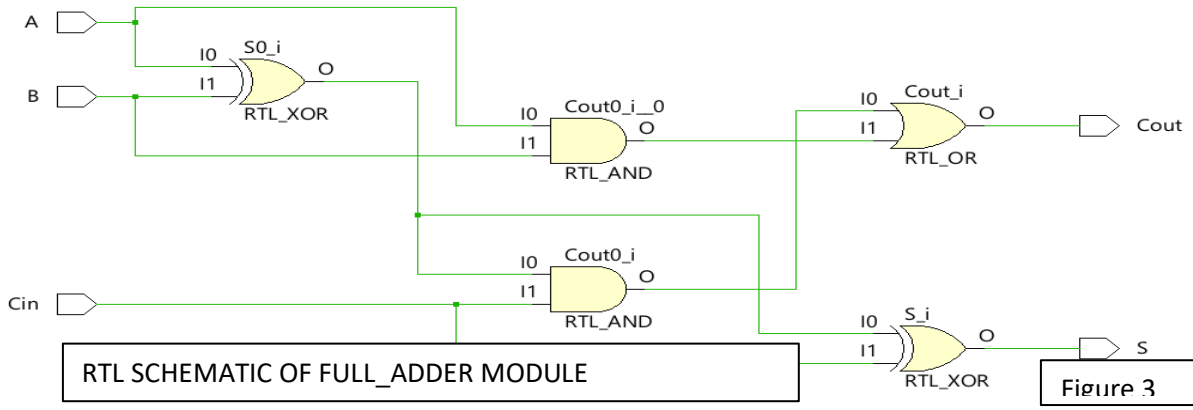
1.2. Throughout the architecture, there is logic for updating a display register (display_reg) based on certain conditions related to the joystick inputs (JA) and the count of active switches (sw). Switches are triggered by pulling the lid of locked boxes (Figure 2). The count of active switches is used to determine the value displayed on the lower four bits of the display register. The algorithm that checks for the winning condition is depending on the inputs JA (signals coming from sensors) and SW. Algorithm uses multiplexers to check whether JA'S and SW's are meeting the winning conditions. Reset button is added to eliminate any unwanted scenarios manually.



RTL SCHEMATIC OF SYSTEM MODULE

Figure 2

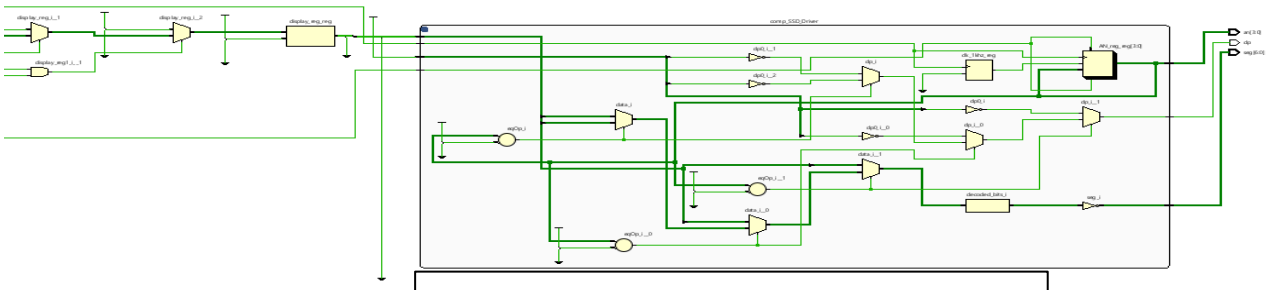
2. Full Adder Module: Purpose of this module is to create a full adder which is going to count the number of balls used by doing binary addition.



RTL SCHEMATIC OF FULL_ADDER MODULE

Figure 3

3. SSD_Driver moduler: The purpose of this module is to define the behavior of a module drive the seven-segment display. This module takes input signals, including the clock (clk), a reset signal (reset), a 16-bit vector representing the data to be displayed (HEX), and a 4-bit vector representing the state of the decimal points (dp_in). The module outputs signals to control a multiplexed display (AN), the seven-segment display segments (seg), and the decimal point (dp).
    3.1. First i generated a clock (clk_1khz) that generates a 1kHz clock signal which is used for driving the anodes of seven segment display. The AN_reg signal is used to control the multiplexing of the display, and it is updated based on the rising edges of the created clock signal.
    3.2. The module includes a decoding process that translates the input data (HEX) into a seven-segment display pattern (decoded_bits). The specific seven-segment patterns are determined based on the input data, and the resulting pattern is negated and assigned to the seg output signal. ROM is used to decode.
    3.3. To prevent unwanted glitches extra glitch removal logics are added.



RTL SCHEMATIC OF SSD_DRIVER MODULE

Figure 4

RTL SCHEMATIC OF SSD_DRIVER AND SYSTEM COMBINED

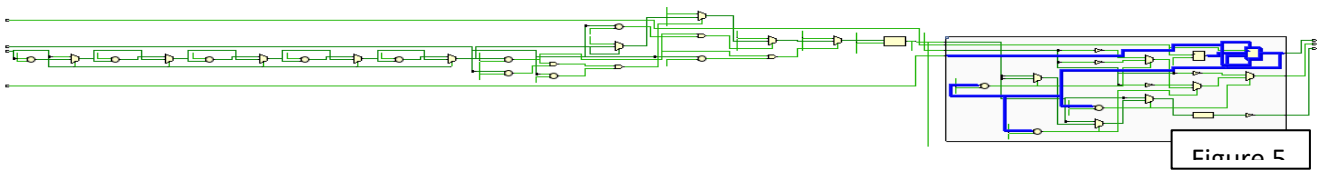


Figure 5

Results:

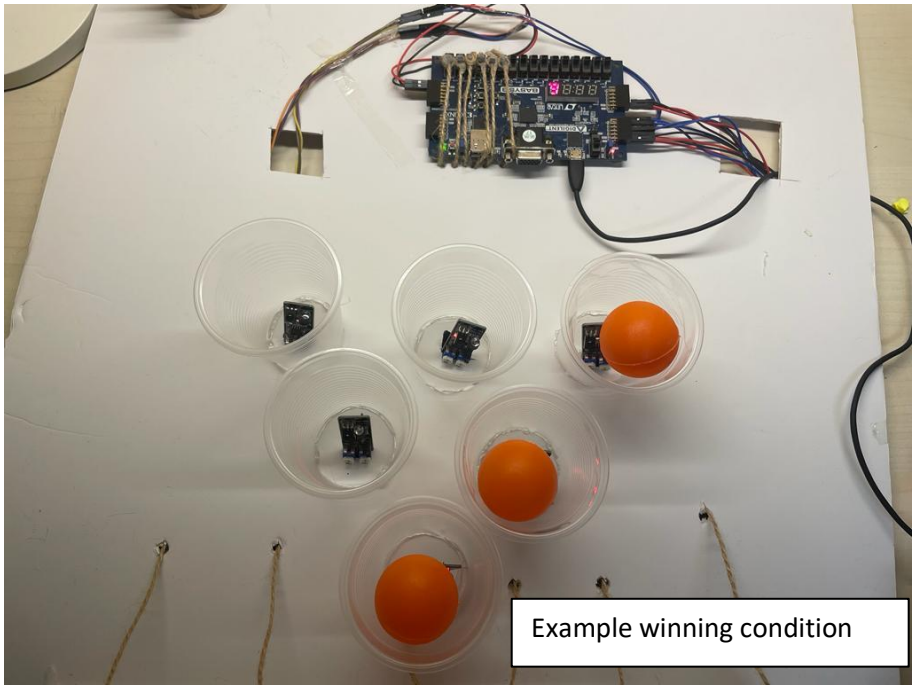
Project is implemented successfully, and it is easy to be enhanced since the code can be easily arranged in a way that this game can be played with more than 10 balls. As can be seen in the presentation (YouTube link is above) the system is working as expected.

Conclusion:

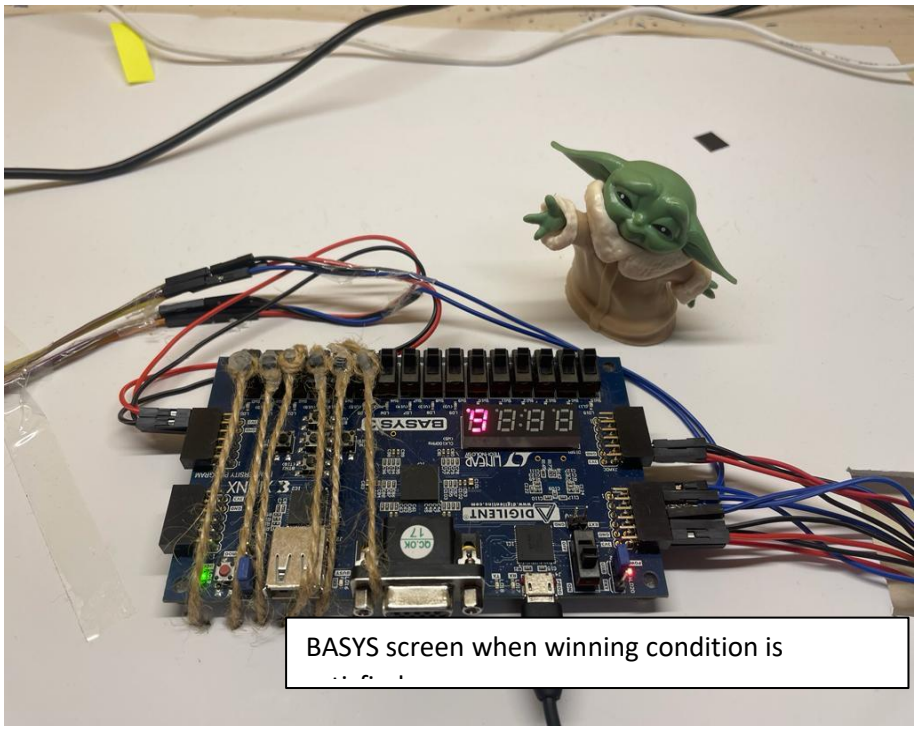
Engineer-pong is designed and implemented by using VHDL, BASYS3 and other external components. Project is designed for 10 cups initially but implemented as a 6-cup system because of the cost of sensors will be increased. Therefore, it is easy to enhance the project. Code was working perfectly. However, minor problems occurred during the physical set up. One of the problems is about the ropes that are connected to switches. I used a fine cotton rope which is strong and has a high fraction. Due to the friction between the edge of the boxes and the ropes boxes were getting harmed. This problem can be solved by using plastic boxes instead of cardboard. The other problem related to physical set up is the instability of the sensors. If the ball lands fast into the board sensors position may change a bit. Although this is not a major problem it can be solved by using supportive columns around the sensor.

Appendices:

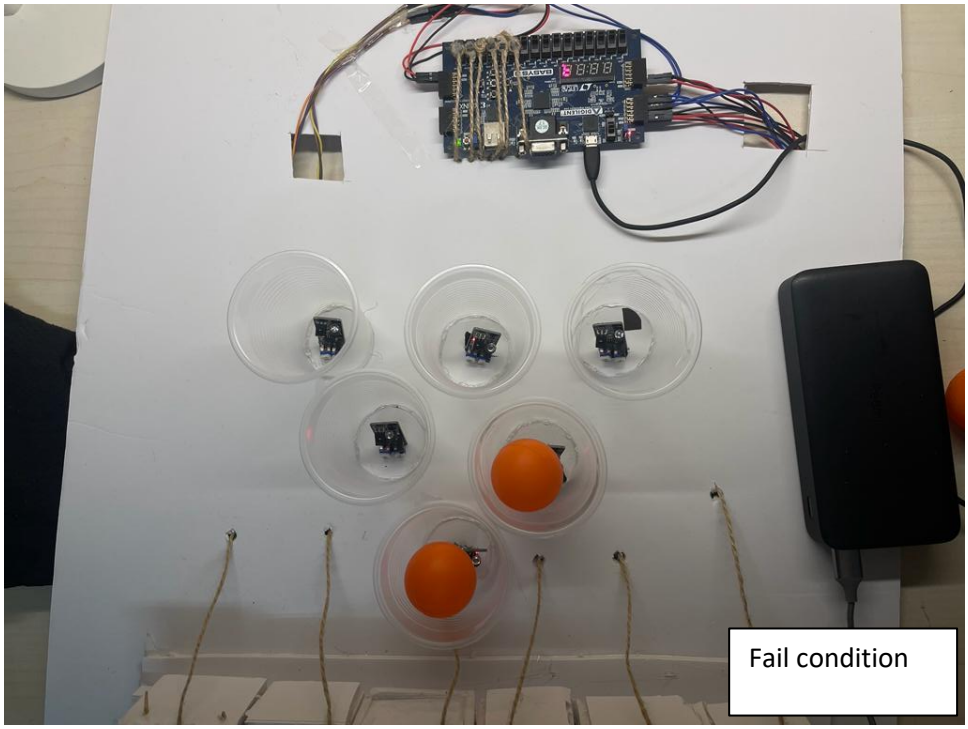
Photos of the project

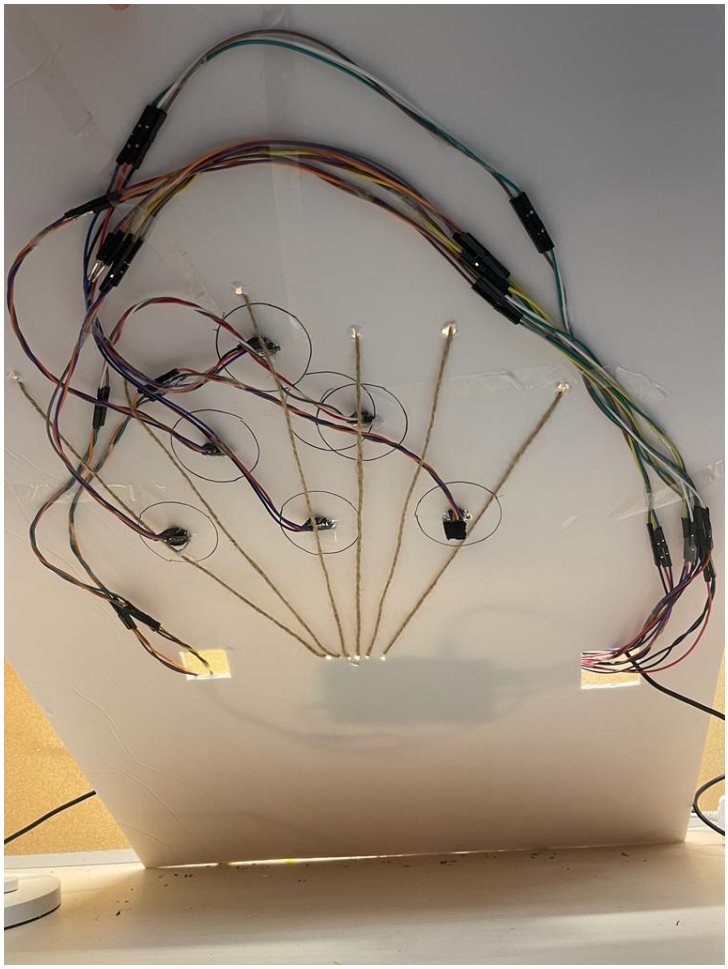


Example winning condition



BASYS screen when winning condition is


Fail condition



VHDL CODES:

SYSTEM MODULE

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 04.12.2023 13:53:48
-- Design Name:
-- Module Name: d9 - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;

entity System is
```

```
    generic(g_delay: natural := 1000000);
    Port ( clk : in STD_LOGIC;
           sw : in STD_LOGIC_VECTOR (5 downto 0);
           JA : in STD_LOGIC_VECTOR (5 downto 0);
           btnC : in STD_LOGIC;

           seg : out STD_LOGIC_VECTOR (6 downto 0);
           an : out STD_LOGIC_VECTOR (3 downto 0);
           dp : out STD_LOGIC);
end System;

architecture Behavioral of System is
    signal display_reg : std_logic_vector(15 downto 0):=(others=>'0');
    signal sw_c : unsigned(3 downto 0):=(others=>'0');
    -- ssd nin componentleri deklare et
    component SSD_Driver is
        generic(g_delay: natural := 1000000);
        Port ( clk : in STD_LOGIC;

               seg  : out STD_LOGIC_VECTOR (6 downto 0);

               dp_in  : in STD_LOGIC_VECTOR (3 downto 0);
               AN   : out STD_LOGIC_VECTOR (3 downto 0);
               dp   : out STD_LOGIC);
               reset : in STD_LOGIc;
               HEX  : in STD_LOGIC_VECTOR (15 downto 0));

    end component SSD_Driver;
    component Adder is ---modülü dışarda oluştur burda kalabalık yapmasın
        Port ( Cin : in  STD_LOGIC;
               A : in  STD_LOGIC_VECTOR (3 downto 0);
```

```
            B : in  STD_LOGIC_VECTOR (3 downto 0);
            Result : out  STD_LOGIC_VECTOR (4 downto 0));
    end component Adder;
begin

   comp_SSD_Driver : SSD_Driver --- generic delay oluştur sonradan değiştirebilirsin
     generic map(g_delay => g_delay)
     port map (
            clk   => clk,
            reset => btnC,


            AN    => an,
            seg   => seg,
            dp    => dp dp_in => "1111";
            HEX   => display_reg,
   );


display_reg(15 downto 4) <= (others=>'0'); ---kazanma conditionlarını ekle 123 -345 -561 sw countu eke
display_reg(3 downto 0) <=
        std_logic_vector(sw_c) when (sw_c=6 and JA(2 downto 0)="000") else
        std_logic_vector(sw_c) when (sw_c=6 and JA(4 downto 2)="000") else
        std_logic_vector(sw_c) when (sw_c=6 and JA(5 downto 4)="00" and JA(0)='0')
        else x"F"  when (sw_c=6)
        else std_logic_vector(sw_c);


process(sw) --- switchleri say
    variable v_sw_count : unsigned(3 downto 0);
begin
v_sw_count := (others=>'0');
for i in 0 to 5 loop
    if(sw(i)='1')then
```

```
        v_sw_count := v_sw_count + 1;
    end if;
end loop;
sw_c <= v_sw_count;
end process;
end Behavioral;
```

SSD_Driver

-- Company:

-- Engineer:

--

-- Create Date: 04.12.2023 13:58:31

-- Design Name:

-- Module Name: SSD_Driver - Behavioral

-- Project Name:

-- Target Devices:

-- Tool Versions:

-- Description:

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

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

entity SSD_Driver is
    generic(g_delay: natural := 1000000);  -- Generic delay for creating a 1 kHz clock
    Port ( clk : in STD_LOGIC;
           reset : in STD_LOGIC;
           HEX  : in STD_LOGIC_VECTOR (15 downto 0);
           dp_in  : in STD_LOGIC_VECTOR (3 downto 0);
           AN   : out STD_LOGIC_VECTOR (3 downto 0);
           sseg  : out STD_LOGIC_VECTOR (6 downto 0);
           dp   : out STD_LOGIC);
end SSD_Driver;

architecture Behavioral of SSD_Driver is
    signal dp : STD_LOGIC_VECTOR(6 downto 0);
    signal data : STD_LOGIC_VECTOR(3 downto 0);
    signal anode_reg : std_logic_vector(3 downto 0):="1110";  -- Anode register for
multiplexing
    signal count : STD_LOGIC_VECTOR(16 downto 0);
    signal 1khz : std_logic;  -- 1 kHz signal for clock generation
    signal HEX0 : STD_LOGIC_VECTOR (3 downto 0);
    signal HEX1 : STD_LOGIC_VECTOR (3 downto 0);
    signal HEX2 : STD_LOGIC_VECTOR (3 downto 0);
    signal HEX3 : STD_LOGIC_VECTOR (3 downto 0);
BEGIN
    -- HEX girişinden ayrı basamakları alır
    HEX3 <= HEX(15 downto 12);
    HEX2 <= HEX(11 downto 8);
    HEX1 <= HEX(7 downto 4);
    HEX0 <= HEX(3 downto 0);
```

```
-- 1 kHz sinyalini oluşturmak için clock işlemi
process(clk)
begin
    if rising_edge(clk) then
        if reset='1' then
            delay := 0;
            1khz <= '0';
        elsif(delay = g_delay-1) then
            delay := 0;
            1khz <= '1';
        else
            1khz <= '0';
            delay := delay + 1;
        end if;
    end if;
end process;

-- Multiplexing için aktif basamağı seçme işlemi
process(clk)
begin
    if rising_edge(clk) then
        if reset='1' then
            anode_reg <= "1110";
        elsif(1khz='1') then
            anode_reg <= anode_reg(2 downto 0) & anode_reg(3);
        end if;
    end if;
end process;

-- AN çıkışına aktif basamağı atama
```

```
AN <= anode_reg;

-- Giriş verisine bağlı olarak uygun HEX değerini atama
data <=
    HEX0 when anode_reg = "1110" else
    HEX1 when anode_reg = "1101" else
    HEX2 when anode_reg = "1011" else
    HEX3;

-- Aktif basamağa bağlı olarak noktayı görüntüleme
dp <=
    not(dp_in(0)) when anode_reg = "1110" else
    not(dp_in(1)) when anode_reg = "1101" else
    not(dp_in(2)) when anode_reg = "1011" else
    not(dp_in(3));

-- Giriş verisine bağlı olarak 7-segment desenini görüntüleme
case data is
    when "0000" => dp <= "1111110"; -- 0
    when "0001" => dp <= "0110000"; -- 1
    when "0010" => dp <= "1101101"; -- 2
    when "0011" => dp <= "1111001"; -- 3
    when "0100" => dp <= "0110011"; -- 4
    when "0101" => dp <= "1011011";
    when "0110" => dp <= "1011111";
    when "0111" => dp <= "1110000";
    when "1000" => dp <= "1111111";
    when "1001" => dp <= "1111011";
    when "1010" => dp <= "1110111";
    when "1011" => dp <= "1111111";
    when "1100" => dp <= "1001110";
```

```
        when "1101" => dp <= "1111110";
        when "1110" => dp <= "1001111";
        when "1111" => dp <= "1110001";
        when others => dp <= "0000000"; -- S off
    end case;

    -- 7-segment görüntüsünü göstermek için dp'yi ters çevirme
    sseg <= not(dp);
end Behavioral;
```

FullAdder

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 04.12.2023 13:31:31
-- Design Name:
-- Module Name: FullAdder - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
----------------------------------
```

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



entity FullAdder is
    Port ( A : in -STD_LOGIC;
           B : in - STD_LOGIC;
           Cin : in  S-TD_LOGIC;
           S : out  STD_-LOGIC;
           Cout : out  STD_LOGIC);
end Full-Adder;

architecture Behavioral of    FullAdder is

begin
S <= (A xor B) xor Cin;
Cout <= ((A xor B) and Cin) or (A and B);
end Behavioral;
```

Constraints

```
# Clock signal
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]

# Switches
set_property PACKAGE_PIN V17 [get_ports {sw[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw[0]}]
```

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


# LED
set_property PACKAGE_PIN U16 [get_ports {ja[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {ja[0]}]
set_property PACKAGE_PIN E19 [get_ports {ja[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {ja[1]}]
set_property PACKAGE_PIN U19 [get_ports {ja[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {ja[2]}]
set_property PACKAGE_PIN V19 [get_ports {ja[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {ja[3]}]
set_property PACKAGE_PIN W18 [get_ports {ja[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {ja[4]}]
set_property PACKAGE_PIN U15 [get_ports {ja[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports ja[5]}]


#7 segment display
set_property PACKAGE_PIN W7 [get_ports {seg[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[6]}]
set_property PACKAGE_PIN W6 [get_ports {seg[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[5]}]
```

```
set_property PACKAGE_PIN U8 [get_ports {seg[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[4]}]
set_property PACKAGE_PIN V8 [get_ports {seg[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[3]}]
set_property PACKAGE_PIN U5 [get_ports {seg[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[2]}]
set_property PACKAGE_PIN V5 [get_ports {seg[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[1]}]
set_property PACKAGE_PIN U7 [get_ports {seg[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[0]}]

set_property PACKAGE_PIN V7 [get_ports dp]

set_property IOSTANDARD LVCMOS33 [get_ports dp]

set_property PACKAGE_PIN U2 [get_ports {an[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {an[0]}]
set_property PACKAGE_PIN U4 [get_ports {an[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {an[1]}]
set_property PACKAGE_PIN V4 [get_ports {an[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {an[2]}]
set_property PACKAGE_PIN W4 [get_ports {an[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {an[3]}]


#Buttons
set_property PACKAGE_PIN U18 [get_ports btnC]
set_property IOSTANDARD LVCMOS33 [get_ports btnC]
set_property PACKAGE_PIN T18 [get_ports btnU]
set_property IOSTANDARD LVCMOS33 [get_ports btnU]
set_property PACKAGE_PIN W19 [get_ports btnL]
set_property IOSTANDARD LVCMOS33 [get_ports btnL]
set_property PACKAGE_PIN T17 [get_ports btnR]
```

```
set_property IOSTANDARD LVCMOS33 [get_ports btnR]
set_property PACKAGE_PIN U17 [get_ports btnD]
set_property IOSTANDARD LVCMOS33 [get_ports btnD]



#Pmod Header JA
#Sch name = JA1
set_property PACKAGE_PIN J1 [get_ports {JA[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {JA[0]}]
#Sch name = JA2
set_property PACKAGE_PIN L2 [get_ports {JA[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {JA[1]}]
#Sch name = JA3
set_property PACKAGE_PIN J2 [get_ports {JA[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {JA[2]}]
#Sch name = JA4
set_property PACKAGE_PIN G2 [get_ports {JA[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {JA[3]}]
#Sch name = JA7
set_property PACKAGE_PIN H1 [get_ports {JA[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {JA[4]}]
#Sch name = JA8
set_property PACKAGE_PIN K2 [get_ports {JA[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {JA[5]}]
##Sch name = JA9
#set_property PACKAGE_PIN H2 [get_ports {JA[6]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {JA[6]}]
##Sch name = JA10
#set_property PACKAGE_PIN G3 [get_ports {JA[7]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {JA[7]}]
```

##Pmod Header JB

##Sch name = JB1

#set_property PACKAGE_PIN A14 [get_ports {JB[0]}]

#set_property IOSTANDARD LVCMOS33 [get_ports {JB[0]}]

##Sch name = JB2

#set_property PACKAGE_PIN A16 [get_ports {JB[1]}]

#set_property IOSTANDARD LVCMOS33 [get_ports {JB[1]}]

##Sch name = JB3

#set_property PACKAGE_PIN B15 [get_ports {JB[2]}]

#set_property IOSTANDARD LVCMOS33 [get_ports {JB[2]}]

##Sch name = JB4

#set_property PACKAGE_PIN B16 [get_ports {JB[3]}]

#set_property IOSTANDARD LVCMOS33 [get_ports {JB[3]}]

##Sch name = JB7

#set_property PACKAGE_PIN A15 [get_ports {JB[4]}]

#set_property IOSTANDARD LVCMOS33 [get_ports {JB[4]}]

##Sch name = JB8

#set_property PACKAGE_PIN A17 [get_ports {JB[5]}]

#set_property IOSTANDARD LVCMOS33 [get_ports {JB[5]}]

##Sch name = JB9

#set_property PACKAGE_PIN C15 [get_ports {JB[6]}]

#set_property IOSTANDARD LVCMOS33 [get_ports {JB[6]}]

##Sch name = JB10

#set_property PACKAGE_PIN C16 [get_ports {JB[7]}]

#set_property IOSTANDARD LVCMOS33 [get_ports {JB[7]}]

##USB HID (PS/2)

#set_property PACKAGE_PIN C17 [get_ports PS2Clk]

#set_property IOSTANDARD LVCMOS33 [get_ports PS2Clk]

#set_property PULLUP true [get_ports PS2Clk]

#set_property PACKAGE_PIN B17 [get_ports PS2Data]

#set_property IOSTANDARD LVCMOS33 [get_ports PS2Data]

#set_property PULLUP true [get_ports PS2Data]

##Quad SPI Flash

##Note that CCLK_0 cannot be placed in 7 series devices. You can access it using the

##STARTUPE2 primitive.

#set_property PACKAGE_PIN D18 [get_ports {QspiDB[0]}]

#set_property IOSTANDARD LVCMOS33 [get_ports {QspiDB[0]}]

#set_property PACKAGE_PIN D19 [get_ports {QspiDB[1]}]

#set_property IOSTANDARD LVCMOS33 [get_ports {QspiDB[1]}]

#set_property PACKAGE_PIN G18 [get_ports {QspiDB[2]}]

#set_property IOSTANDARD LVCMOS33 [get_ports {QspiDB[2]}]

#set_property PACKAGE_PIN F18 [get_ports {QspiDB[3]}]

#set_property IOSTANDARD LVCMOS33 [get_ports {QspiDB[3]}]

#set_property PACKAGE_PIN K19 [get_ports QspiCSn]

#set_property IOSTANDARD LVCMOS33 [get_ports QspiCSn]