T.C AFYON KOCATEPE ÜNİVERSİTESİ
TEKNOLOJİ FAKÜLTESİ
ELEKTRİK-ELEKTRONİK
MÜHENDİSLİĞİ BÖLÜMÜ

Konu :FPGA ile XADC Uygulaması

Ders : Sayısal Sistemler Tasarımı

Dr. Öğr. Üyesi İsmail KOYUNCU

Numara	Adı-Soyadı
154210006	Mustafa ESGİN
154208033	Hüseyin KUZUCU

İÇİNDEKİLER

FPGA ve VHDL Nedir

- Alan/alanda programlanabilir kapı dizileri olarak Türkçeye çevirebileceğimiz FPGA çipleri, ilk üretiminin ardından tekrar tekrar programlanabilme özelliğine sahip kullanıcı/tasarımcı tarafından yapılan sistem tasarımına göre yapısının değiştiril
- Diğer platformlara göre paralel çalışma, hızlı ilk üretim ve yüksek performans gibi özellikleriyle ön plana çıkmaktadırlar.
- FPGA çipleri Xilinx ve Altera (Intel) gibi çeşitli firmalar tarafından üretilmektedir.
- Her çip üreticisi üretmiş olduğu çiplerin tasarımı için farklı programlar geliştirmişlerdir.
- Bu programlara Vivado (Xilinx) ve Quartus (Altera) örnek olarak verilebilir. Çiplerin tasarımı şematik, Matlab HDL Coder ve bazı programlama dilleri ile gerçekleştirilebilmektedir.
- Bu programlama dillerinden en çok kullanılan Verilog ve VHDL dilleridir.ebileceği tüm devrelerdir.

FPGA İLE XADC PROJESİ

➢ Özet

- Bu projede, VHDL programlama dilinde 0V-1V arasındaki değerleri ondalık sayısal değere çevirip 7 segmentli display üzerinde 0-255 arası değerlerde gösterilecektir.
- Bu 0-255 arası değerlere göre basys-3 kartı üzerindeki ledler tarafından kontrol edilecektir.
- 0 ile 255 değerleri arasında yanan ledlerin sayısı azalıp, artıyor.
- 7 segmentli display 255 sayısını gösterdiğinde ledler sönüyor.

➢ Amaç

- Bu proje, bir gerilim değerinin belirli aralıklar içerisinde 2'lik sayı sistemine çevirtilip bu değerleri ondalık olarak okumamızı sağlamaktır. Ayrıca VHDL kod programlamayı basys-3 kartını kullanmayı ve isteğe göre basys kartını harici devreler bağlayarak öğrenmeyi amaçlamaktadır.
- Herhangi bir su çıkışının akışını ölçmek için kullanılabilecek bir su debimetresidir. Harici olarak su çıkışına bağlanabilmesi hedeflenmiştir.

Bu proje aşağıdaki ana kısımlardan oluşmaktadır;

- Öncelikle vivado programında yazdığımız kodları çalıştırıp aktarabileceğimiz ana unsur olan basys-3 kartı tahsis edilmelidir.
- Bu proje, programda belirttiğimiz analog verileri dijital veriye çevirip ve gereken voltaj değerleri arasında değerleri ayarlamamızı sağlar, bunun için bir potansiyometreye ihtiyacımız var.
- Potansiyometreyi bağlamak için bread board ve bilgileri potansiyometreye aktaracak bağlantı kablosu jumperlara ihtiyacımız vardır.
- Basys-3 kartı üzerindeki 7 segmentli display yerine harici display kullanılmak istenilirse ekstra bir 7 segment display ve dirençler kullanılmaktadır.
- Bu dirençler sayesinde her bir display de bulunan ledlerin üzerindeki akımı kontrol ederek ledlerin zarar görmesini engellemiş oluruz.

Kısaca Özellikler

- FPGA üzerinde veri işlemeyi, analog veriyi, dijital veriye çevirmeye olanak sağlıyor.
- Basys-3 kartını öğrenmeyi amaçlayanlar için ve harici devrelerle basys-3 kartını kontrol etmek isteyenler için bütün imkanı sağlıyor.
- Bu devre sayesinde önemli denilebilecek kod yazımında vivadoda program yazmayı ve dışarıya aktarmayı, uygulama olarak hayata geçirmekte uzmanlaşmak için fırsat sunuyor.

Gerekli Digilent Ürünler

- Baysy-3 Artix-7 FPGA Kartı

Gerekli Donanımlar

- Baysy-3 Artix-7 FPGA Kartı
- Potansiyometre
- Jumper
- Bread Board

Yazılım Kısmı

- İşletim sistemi windows 7-10 service pack 1 veya üstü bilgisayarlarda kullanılabilir.
- Vivado 2018.1

Tasarım Durumu

- Vivado programı ile gereken led bacaklarına basys-3 kartı üzerindeki 7 segmentli display bacak kodları tanımlanmıştır.
- İsteğe göre JB VE JC konnektör bacak kodları atanarak JB VE JC den çıkış alınarak harici bir display bağlanılabilinir veya başka devrelerde bu konnektörler kullanılabilinir.
- JX konnektörü sayesinde potansiyometre bağlantısı yapıldı.
- Potansiyometre ile 0V-1V arası gerilim değerlerini basys-3 kartı üzerinde 7 segmentli display ve ledler sayesinde hem sayısal hem de ledler tarafından gözlemlendi.

Dizayn ve Özellikler

- Her şeyden önce , bu proje için basys-3 kartına ihtiyaç duyulacaktır.
- Devremizin gerilim ayarlamasını yapabilmemiz için potansiyometre, bread board, jumper kablolarına ihtiyacımız vardır.

Projenin Çalışma Algoritması

- Basys-3 kartı üzerindeki display çalıştırılmak üzere gereken değerler, vivado programı tarafından basys-3 kartına gönderildi.
- Farklı gerilim değerlerini sağlayacak potansiyometremiz bread board üzerine kuruldu.
- Displayden alacağımız değerleri ve belirli sayıda yakacağımız ledler için basys-3 kartına potansiyometre bağlantısı yapıldı.
- Bağlantılar tamamlandıktan sonra basys-3 kartı çalıştırılıp vivado programı ile gereken bilgiler display üzeride gösterildi ve ledler yakıldı.

XADC Nedir?

- FPGA'ın yardımcı analog giriş pinlerine bağlanır.
- Konfigürasyona bağlı olarak, bu konnektör Artix-7(XADC) içindeki analog digital dönüştürücüye diferansiyel analog sinyalleri girmek için kullanılabilir.
- Konnektördeki herhangi bir çift veya tüm çiftler analog giriş veya digital giriş çıkışı olarak yapılandırılabilir.
- 8 veri sinyali 4 çift halinde gruplandırılmıştır, bu gruplandırılmış çiftler daha iyi analog gürültü bağışıklığı için yakından bağlanmıştır.
- Ayrıca her çift PCB üzerine yerleştirilmiş kısmen yüklenmiş bir kenar yumuşatma filtrelerine sahiptir.

BASYS3 KARTI

- Basys3 ile uygulama devreleri arasındaki bağlantının hangi pinler ile sağlandığına dikkat edilmelidir.
- Basys3’ ün dış dünyaya açılan portlardan JB ve JC pmod konnektörü, set üzerine, J1 ve J2 portları olarak taşınmıştır.
- Her iki portun (J1 ve J2) paralel olduğuna dikkat edilmelidir.
- Flat kablo ile de bağlantılar sağlanmaktadır.

Şekil1: Basys 3 kartı

Basys-3 Kartı Görünüşü

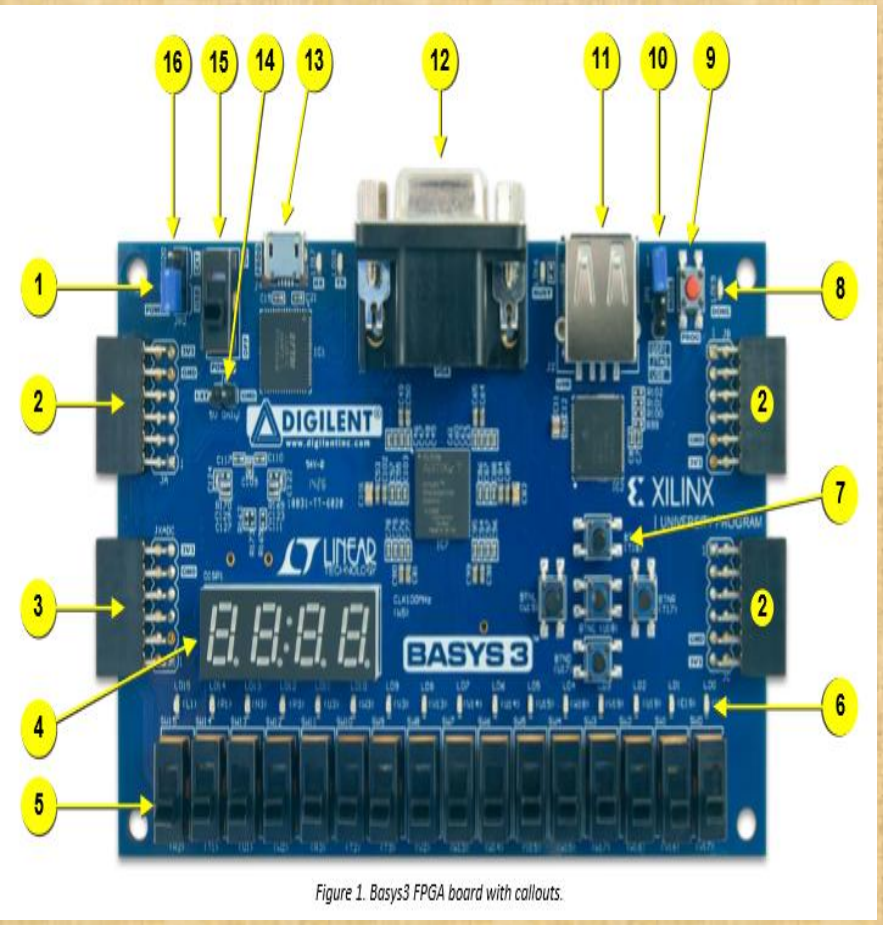


Figure 1. Basys3 FPGA board with callouts.

1= Güç Ledi
2= JA Pmod Konnektörü
3= XADC Analog Sinyal Pmod Konnektörü
4= 4 Haneli 7 Segmentli Display
5= Sürgülü Şalterler
6= Led
7= Butonlar
8= FPGA Programlama Ledi
9= FPGA Yapılandırma/Sıfırlama Düğmesi
10= Programlama modu atlama kablosu
11= USB Ana Bilgisayar Konnektörü
12= VGA Konnektörü
13= Paylaşımlı UART/JTAG USB Bağlantı Noktası
14= Harici Güç Konnektörü
15= Güç Düğmesi
16= Güç Seçimi Atlama Teli

Şekil4: Basys3 Kartı Tanıtımı

POTANSİYOMETRE

- Direnci dışardan el ile manuel olarak ayarlanabilen küçük değerlere sabit ayarlı dirençdir.
- Hem doğru akım, hem alternatif akım devrelerinde kullanılır.
- Potansiyometrenin dönebilen ayar düğmesini sağa ve sola çevirdiğimizde direnci bizim çevirme miktarına göre orantılı olarak direnç değeri azalır ve artar.

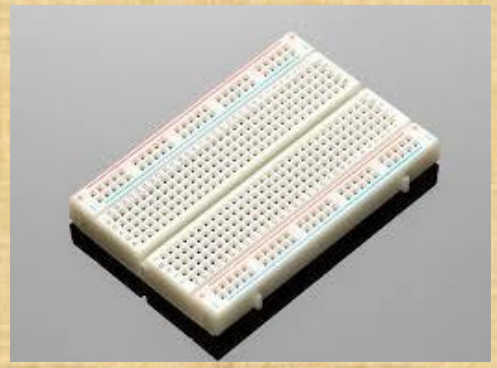
Şekil2: Potansiyometre datasheet

JUMPER

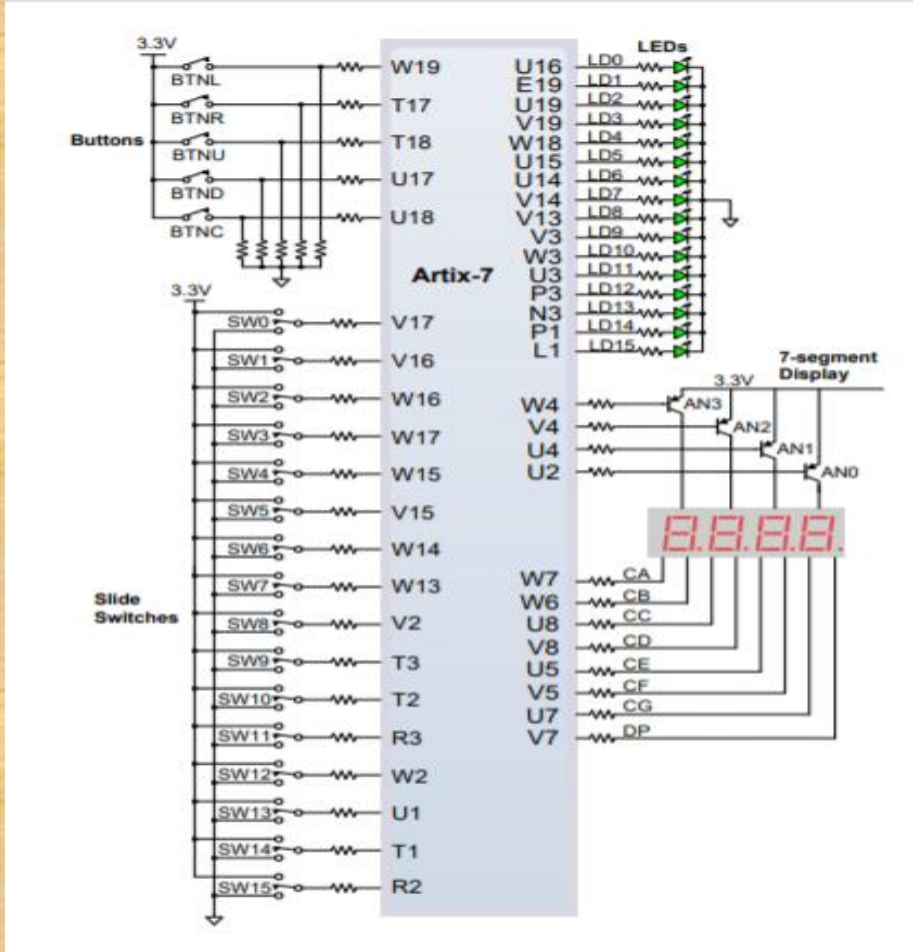
- Basys3 kartı ile uygulamalar arasındaki bağlantı bu bağlantı kabloları ile gerçekleştirilir.

BREAD BOARD

- Devreleri tak çıkar mantığı ile oluşturmamıza yarayan, belli satır ve sütunları kendi aralarında iletken edilmiş devre tahtasıdır.



Basys-3 Kartı Display Bağlantı Şeması



Şekil3: Bağlantı Bacakları

VHDL Kodlar Akış Ölçer;

```
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;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity FLOW_METER_SYSTEM is
    Port ( ADC : in STD_LOGIC_VECTOR (1 downto 0);
           clk : in STD_LOGIC;
           an : out STD_LOGIC_VECTOR (3 downto 0);
           seg : out STD_LOGIC_VECTOR (7 downto 0);
           led : out STD_LOGIC_VECTOR (7 downto 0):= "00000000"
           );
end FLOW_METER_SYSTEM;

architecture Behavioral of FLOW_METER_SYSTEM is

component xadc_wiz_0 is
  port( daddr_in   :   in  STD_LOGIC_VECTOR (6 downto 0);
        den_in     :   in  STD_LOGIC;
        di_in      :   in  STD_LOGIC_VECTOR (15 downto 0);
        dwe_in     :   in  STD_LOGIC;
        do_out     :   out STD_LOGIC_VECTOR (15 downto 0);
        drdy_out   :   out STD_LOGIC;
        dclk_in    :   in  STD_LOGIC;
        reset_in   :   in  STD_LOGIC;
        vauxp14    :   in  STD_LOGIC;
        vauxn14    :   in  STD_LOGIC;
        busy_out   :   out STD_LOGIC;
        channel_out:   out STD_LOGIC_VECTOR (4 downto 0);
        eoc_out    :   out STD_LOGIC;
        eos_out    :   out STD_LOGIC;
        alarm_out  :   out STD_LOGIC;
        vp_in      :   in  STD_LOGIC;
        vn_in      :   in  STD_LOGIC );
end component;
```

```
component sseg_dec is
  port( ALU_VAL : in std_logic_vector (7 downto 0);
        SIGN    : in std_logic;
        VALID   : in std_logic;
        CLK     : in std_logic;
        DISP_EN : out std_logic_vector (3 downto 0);
        SEGMENTS: out std_logic_vector (7 downto 0) );

end component;

component clk_div2 is
  port  ( clk : in std_logic;
          sclk: out std_logic );

end component;

component EightBitDataPassDelay is
  port ( FastIn     : in STD_LOGIC_VECTOR (7 downto 0);
         UpdateNow  : in STD_LOGIC;
         HeldSample : out STD_LOGIC_VECTOR (7 downto 0) );
end component;

component EightBitBarMeter is
  port ( BarDatIn  : in  STD_LOGIC_VECTOR (7 downto 0);
         BarDatOut : out STD_LOGIC_VECTOR (7 downto 0):= "00000000" );
end component;

signal ADCintcon      : STD_LOGIC_VECTOR (7 downto 0);
signal ADCslowintcon  : STD_LOGIC_VECTOR (7 downto 0);
signal Waistintcon    : STD_LOGIC_VECTOR (7 downto 0);
signal EnableInt      : STD_LOGIC:='1';
signal ReadyInt       : STD_LOGIC;
signal SlowClock      : STD_LOGIC;
```

```
begin
    clockDiv: clk_div2
        port map ( clk => CLK,
                   sclk=> SlowClock );
        ADCimp: xadc_wiz_0

        port map ( daddr_in             => "0011110",
                   den_in               => EnableInt,
                   di_in                => (others => '0'),
                   dwe_in               => '0',
                   do_out (15 downto 8) => ADCintcon,
                   do_out (7 downto 0)  => Waistintcon,
                   drdy_out             => open,
                   dclk_in              => clk,
                   reset_in             => '0',
                   vauxp14              => ADC(0),
                   vauxn14              => ADC(1),
                   busy_out             => open,
                   channel_out          => open,
                   eoc_out              => EnableInt,
                   eos_out              => open,
                   alarm_out            => open,
                   vp_in                => '0',
                   vn_in                => '0' );

Delaytimer: EightBitDataPassDelay
   port map ( FastIn     => ADCintcon (7 downto 0),
              UpdateNow  => SlowClock,
              HeldSample => ADCslowintcon (7 downto 0) );
SSegDisp: sseg_dec
  port map ( ALU_VAL     => ADCslowintcon (7 downto 0),
             SIGN        => '0',
             VALID       => '1',
             CLK         => clk,
             DISP_EN     => an,
             SEGMENTS    => seg );
  Barmeter: EightBitBarMeter
     port map ( BarDatIn  => ADCintcon (7 downto 0),
                BarDatOut => led (7 downto 0) );


end Behavioral;
```

(8-bit) Bar Metre;

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


entity EightBitBarMeter is
    Port (  BarDatIn  : in  STD_LOGIC_VECTOR (7 downto 0);
            BarDatOut : out STD_LOGIC_VECTOR (7 downto 0) := "00000000"   );
end EightBitBarMeter;

architecture Behavioral of EightBitBarMeter is
begin
    Bardisp: process (BarDatIn)
            begin
            if    (BarDatIn >= "00000000"  and BarDatIn < "00100000") then
                BarDatOut <= "00000001";
            elsif (BarDatIn >= "00100000"  and BarDatIn < "01000000") then
                BarDatOut <= "00000011";
            elsif (BarDatIn >= "01000000"  and BarDatIn < "01100000") then
                BarDatOut <= "00000111";
            elsif (BarDatIn >= "01100000"  and BarDatIn < "10000000") then
                BarDatOut <= "00001111";
            elsif (BarDatIn >= "10000000"  and BarDatIn < "10100000") then
                BarDatOut <= "00011111";
            elsif (BarDatIn >= "10100000"  and BarDatIn < "11000000") then
                BarDatOut <= "00111111";
            elsif (BarDatIn >= "11000000"  and BarDatIn < "11100000") then
                BarDatOut <= "01111111";
            elsif (BarDatIn >= "11100000"  and BarDatIn < "11111111") then
                BarDatOut <= "11111111";
            else BarDatOut <= "00000000";
        end if;
```

```
if    (BarDatIn >= "00000000"  and BarDatIn < "00010000") then
    BarDatOut <= "0000000000000001";
elsif (BarDatIn >= "00010000"  and BarDatIn < "00100000") then
    BarDatOut <= "0000000000000011";
elsif (BarDatIn >= "00100000"  and BarDatIn < "00110000") then
    BarDatOut <= "0000000000000111";
elsif (BarDatIn >= "00110000"  and BarDatIn < "01000000") then
    BarDatOut <= "0000000000001111";
elsif (BarDatIn >= "01000000"  and BarDatIn < "01010000") then
    BarDatOut <= "0000000000011111";
elsif (BarDatIn >= "01010000"  and BarDatIn < "01100000") then
    BarDatOut <= "0000000000111111";
elsif (BarDatIn >= "01100000"  and BarDatIn < "01110000") then
    BarDatOut <= "0000000001111111";
elsif (BarDatIn >= "01110000"  and BarDatIn < "10000000") then
    BarDatOut <= "0000000011111111";
elsif (BarDatIn >= "10000000"  and BarDatIn < "10010000") then
    BarDatOut <= "0000000111111111";
elsif (BarDatIn >= "10010000"  and BarDatIn < "10100000") then
    BarDatOut <= "0000001111111111";
elsif (BarDatIn >= "10100000"  and BarDatIn < "10110000") then
    BarDatOut <= "0000011111111111";
elsif (BarDatIn >= "10110000"  and BarDatIn < "11000000") then
    BarDatOut <= "0000111111111111";
elsif (BarDatIn >= "11000000"  and BarDatIn < "11010000") then
    BarDatOut <= "0001111111111111";
elsif (BarDatIn >= "11010000"  and BarDatIn < "11100000") then
    BarDatOut <= "0011111111111111";
elsif (BarDatIn >= "11100000"  and BarDatIn < "11110000") then
    BarDatOut <= "0111111111111111";
elsif (BarDatIn >= "11110000"  and BarDatIn <= "11111111") then
    BarDatOut <= "1111111111111111";
else BarDatOut <= "0000000000000000";
end if;
end process;
end Behavioral;
```

Clock_div2;

```
--Giriş saati frekansını daha yavaş bir frekansa böler.
----------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
----------------------------------------------------------------------------
-- saati bölmek için modül
----------------------------------------------------------------------------
entity clk_div2 is
    Port (  clk : in std_logic;
            sclk : out std_logic);
end clk_div2;

architecture my_clk_div of clk_div2 is
   constant max_count : integer := (3000000);
   signal tmp_clk : std_logic := '0';
begin
   my_div: process (clk,tmp_clk)
      variable div_cnt : integer := 0;
   begin
      if (rising_edge(clk)) then
         if (div_cnt = MAX_COUNT) then
            tmp_clk <= not tmp_clk;
            div_cnt := 0;
         else
            div_cnt := div_cnt + 1;
         end if;
      end if;
      sclk <= tmp_clk;
   end process my_div;
end my_clk_div;
```

(8-bit) Veri Geçiş Gecikmesi;

```
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;

-- Uncomment the following library declaration if instantiat
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity EightBitDataPassDelay is
    Port ( FastIn : in STD_LOGIC_VECTOR (7 downto 0);
           UpdateNow : in STD_LOGIC;
           HeldSample : out STD_LOGIC_VECTOR (7 downto 0));
end EightBitDataPassDelay;

architecture Behavioral of EightBitDataPassDelay is

component DFF is
 Port ( DATAIN     : in  STD_LOGIC;
        CLK        : in  STD_LOGIC;
        ENABLE     : in  STD_LOGIC;
        DATAOUT    : out STD_LOGIC := '0';
        NOTDATAOUT : out STD_LOGIC := '1' );
end component;

begin

    FF1 : DFF
        port map( DATAIN     => FastIn(0),
                  CLK        => UpdateNow,
                  ENABLE     => '1',
                  DATAOUT    => HeldSample(0),
                  NOTDATAOUT => open );

    FF2 : DFF
        port map( DATAIN     => FastIn(1),
                  CLK        => UpdateNow,
                  ENABLE     => '1',
                  DATAOUT    => HeldSample(1),
                  NOTDATAOUT => open );

    FF3 : DFF
        port map( DATAIN     => FastIn(2),
                  CLK        => UpdateNow,
                  ENABLE     => '1',
                  DATAOUT    => HeldSample(2),
                  NOTDATAOUT => open );

    FF4 : DFF
        port map( DATAIN     => FastIn(3),
                  CLK        => UpdateNow,
                  ENABLE     => '1',
                  DATAOUT    => HeldSample(3),
                  NOTDATAOUT => open );

    FF5 : DFF
        port map( DATAIN     => FastIn(4),
                  CLK        => UpdateNow,
                  ENABLE     => '1',
                  DATAOUT    => HeldSample(4),
                  NOTDATAOUT => open );
    FF6 : DFF
        port map( DATAIN     => FastIn(5),
                  CLK        => UpdateNow,
                  ENABLE     => '1',
                  DATAOUT    => HeldSample(5),
                  NOTDATAOUT => open );

    FF7 : DFF
        port map( DATAIN     => FastIn(6),
                  CLK        => UpdateNow,
                  ENABLE     => '1',
                  DATAOUT    => HeldSample(6),
                  NOTDATAOUT => open );

    FF8: DFF
        port map( DATAIN     => FastIn(7),
                  CLK        => UpdateNow,
                  ENABLE     => '1',
                  DATAOUT    => HeldSample(7),
                  NOTDATAOUT => open );
end Behavioral;
```

D- Flip Flop(DFF);

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity DFF is
    Port ( DATAIN      :  in STD_LOGIC;
           CLK         :  in STD_LOGIC;
           ENABLE      :  in STD_LOGIC;
           DATAOUT     : out STD_LOGIC :='0';
           NOTDATAOUT  : out STD_LOGIC :='1');
end DFF;

architecture Behavioral of DFF is

begin
 dff: process (DATAIN,CLK)
begin
    if(rising_edge(CLK)) then
        if (ENABLE = '1') then
            DATAOUT    <= DATAIN;
            NOTDATAOUT <= not DATAIN;
end if;
end if;
end process dff;
end Behavioral;
```

7-Segment Display;

```
-- Açıklama: Özel yedi segment ekran sürücüsü;
--
--  iki özel giriş:
--
--      VALID: if valid = 0, dört çizgi gösterilecek
--             if valid = 1, ekranda ondalık sayı görünüyor
--
--       SIGN: if sign = 1, en sol hanede bir eksi işareti görünüyor
--             if sign = 0, eksi işareti görünmüyor

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

----------------------------------------------------------------
-- 4 basamaklı yedi bölümlü ekran sürücüsü. Çıkışlar aktif
-- "segment" çıkışında düşük ve yapılandırılmış ABCEDFG.
----------------------------------------------------------------
entity sseg_dec is
    Port (      ALU_VAL : in std_logic_vector(7 downto 0);
                                          SIGN : in std_logic;
                                             VALID : in std_logic
                   CLK : in std_logic;
                DISP_EN : out std_logic_vector(3 downto 0);
               SEGMENTS : out std_logic_vector(7 downto 0));
end sseg_dec;

architecture my_sseg of sseg_dec is

   -- 2-bitlik BCD dönüştürücüye 8-bit ikili bildirimi --
   component bin2bcdconv
       Port ( BIN_CNT_IN : in std_logic_vector(7 downto 0);
                 LSD_OUT : out std_logic_vector(3 downto 0);
                 MSD_OUT : out std_logic_vector(3 downto 0);
                MMSD_OUT : out std_logic_vector(3 downto 0));
   end component;

      component clk_div
               Port (  clk : in std_logic;
                            sclk : out std_logic);
      end component;

-- ara sinyal bildirimi -----------------------
signal   cnt_dig : std_logic_vector(1 downto 0);
signal   digit : std_logic_vector (3 downto 0);
signal   lsd,msd,mmsd : std_logic_vector(3 downto 0);
     signal   sclk : std_logic;

begin

-- bcd dönüştürücüye bin örneği -----------------
my_conv: bin2bcdconv
     port map ( BIN_CNT_IN => ALU_VAL,
                LSD_OUT => lsd,
                MSD_OUT => msd,
               MMSD_OUT => mmsd);

my_clk: clk_div
     port map (clk => clk,
               sclk => sclk );

-- sayımı ilerletmek (ekran çoğullama için kullanılır) -----
process (SCLK)
begin
   if (rising_edge(SCLK)) then
      cnt_dig <= cnt_dig + 1;
   end if;
end process;


-- select the display sseg data abcdefg (active low) -----
segments <= "00000011" when digit = "0000"  else
            "10011111" when digit = "0001"  else
            "00100101" when digit = "0010"  else
            "00001101" when digit = "0011"  else
            "10011001" when digit = "0100"  else
            "01001001" when digit = "0101"  else
            "01000001" when digit = "0110"  else
            "00011111" when digit = "0111"  else
            "00000001" when digit = "1000"  else
            "00001001" when digit = "1001"  else
                                    "11111101" when digit = "1110" else   -- dash
                                    "11111111" when digit = "1110" else   -- blank

            "11111111";
```

```
-- Doğru ekranı çalıştır -------------------------
disp_en <= "1110" when cnt_dig = "00" else
           "1101" when cnt_dig = "01" else
           "1011" when cnt_dig = "10" else
           "0111" when cnt_dig = "11" else
           "1111";


    process (cnt_dig, lsd, msd, mmsd, sign, valid)
       variable mmsd_v, msd_v : std_logic_vector(3 downto 0);
    begin
       mmsd_v := mmsd;
            msd_v := msd;

            -- do the lead zero blanking for two msb's
            if (mmsd_v = X"0") then
               if (msd_v = X"0") then
                       msd_v := X"F";
                    end if;
               mmsd_v := X"F";
            end if;

            if (valid = '1') then
                    if (sign = '0') then
                            case cnt_dig is
                                    when "00" => digit <= "1111";
                                    when "01" => digit <= mmsd_v;
                                    when "10" => digit <= msd_v;
                                    when "11" => digit <= lsd;
                                    when others => digit <= "0000";
                            end case;
                    else
                            case cnt_dig is
                                    when "00" => digit <= "1110";
                                    when "01" => digit <= mmsd_v;
                                    when "10" => digit <= msd_v;
                                    when "11" => digit <= lsd;
                                    when others => digit <= "0000";
                            end case;
                    end if;
            else digit <= "1110";
            end if;
    end process;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
----------------------------------------------------------------------
-- bcd dönüştürücü bin için arayüz açıklaması
----------------------------------------------------------------------
entity bin2bcdconv is
    Port ( BIN_CNT_IN : in std_logic_vector(7 downto 0);
              LSD_OUT : out std_logic_vector(3 downto 0);
              MSD_OUT : out std_logic_vector(3 downto 0);
             MMSD_OUT : out std_logic_vector(3 downto 0));
end bin2bcdconv;

----------------------------------------------------------------------
-- 3-bit BCD dönüştürücünün 8-bit ikili açıklaması
----------------------------------------------------------------------
architecture my_ckt of bin2bcdconv is
begin
   process(bin_cnt_in)
       variable cnt_tot : INTEGER range 0 to 255 := 0;
            variable lsd,msd,mmsd : INTEGER range 0 to 9 := 0;
   begin

        -- girdi ikili değerini ondalık değerine dönüştür
        cnt_tot := 0;
    if (bin_cnt_in(7) = '1') then cnt_tot := cnt_tot + 128;  end if;
    if (bin_cnt_in(6) = '1') then cnt_tot := cnt_tot +  64;  end if;
    if (bin_cnt_in(5) = '1') then cnt_tot := cnt_tot +  32;  end if;
    if (bin_cnt_in(4) = '1') then cnt_tot := cnt_tot +  16;  end if;
    if (bin_cnt_in(3) = '1') then cnt_tot := cnt_tot +   8;  end if;
    if (bin_cnt_in(2) = '1') then cnt_tot := cnt_tot +   4;  end if;
    if (bin_cnt_in(1) = '1') then cnt_tot := cnt_tot +   2;  end if;
    if (bin_cnt_in(0) = '1') then cnt_tot := cnt_tot +   1;  end if;

        -- ara sinyalleri başlat
    msd  := 0;
        mmsd := 0;
        lsd  := 0;

    --  MMSB'yi hesapla
    for I in 1 to 2 loop
            exit when (cnt_tot >= 0 and cnt_tot < 100);
```

```
        mmsd := mmsd + 1; -- increment the mmds count
        cnt_tot := cnt_tot - 100;
end loop;

 --  MSB'yi hesaplamak
for I in 1 to 9 loop
   exit when (cnt_tot >= 0 and cnt_tot < 10);
   msd := msd + 1; -- increment the mds count
        cnt_tot := cnt_tot - 10;
end loop;

     lsd := cnt_tot;   -- lsd is what is left over

     -- LSD'yi ikiliye dönüştürmek
     case lsd is
        when 9 =>  lsd_out <= "1001";
        when 8 =>  lsd_out <= "1000";
        when 7 =>  lsd_out <= "0111";
        when 6 =>  lsd_out <= "0110";
        when 5 =>  lsd_out <= "0101";
        when 4 =>  lsd_out <= "0100";
        when 3 =>  lsd_out <= "0011";
        when 2 =>  lsd_out <= "0010";
        when 1 =>  lsd_out <= "0001";
        when 0 =>  lsd_out <= "0000";
        when others =>  lsd_out <= "0000";
end case;

     -- msd'yi ikiliye dönüştür
     case msd is
        when 9 =>  msd_out <= "1001";
        when 8 =>  msd_out <= "1000";
        when 7 =>  msd_out <= "0111";
        when 6 =>  msd_out <= "0110";
        when 5 =>  msd_out <= "0101";
        when 4 =>  msd_out <= "0100";
        when 3 =>  msd_out <= "0011";
        when 2 =>  msd_out <= "0010";
        when 1 =>  msd_out <= "0001";
        when 0 =>  msd_out <= "0000";
        when others =>  msd_out <= "0000";
  end case;
```

```
        -- msd'yi ikiliye dönüştür
        case mmsd is
           when 2 =>  mmsd_out <= "0010";
           when 1 =>  mmsd_out <= "0001";
           when 0 =>  mmsd_out <= "0000";
           when others =>  mmsd_out <= "00
     end case;
end process;
end my_ckt;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
----------------------------------------
-- Saati bölmek için modül
----------------------------------------
entity clk_div is
    Port (  clk : in std_logic;
            sclk : out std_logic);
end clk_div;

architecture my_clk_div of clk_div is
   constant max_count : integer := (2200);
   signal tmp_clk : std_logic := '0';
begin
   my_div: process (clk,tmp_clk)
      variable div_cnt : integer := 0;
   begin
      if (rising_edge(clk)) then
         if (div_cnt = MAX_COUNT) then
            tmp_clk <= not tmp_clk;
            div_cnt := 0;
         else
            div_cnt := div_cnt + 1;
         end if;
      end if;
      sclk <= tmp_clk;
   end process my_div;
end my_clk_div;
```

XADC;

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
Library UNISIM;
use UNISIM.VCOMPONENTS.ALL;

entity xadc_wiz_0 is
   port
   (
    daddr_in       : in  STD_LOGIC_VECTOR (6 downto 0);     -- Dinamik yeniden yapılandırma portu için adres veriyolu
    den_in         : in  STD_LOGIC;                         -- Dinamik yeniden yapılandırma portu için sinyali etkinleştirme
    di_in          : in  STD_LOGIC_VECTOR (15 downto 0);    -- Dinamik yeniden yapılandırma portu için giriş veri yolu
    dwe_in         : in  STD_LOGIC;                         -- Dinamik yeniden yapılandırma portu için Enable (Yaz) Etkinleştir
    do_out         : out  STD_LOGIC_VECTOR (15 downto 0);   -- Dinamik yeniden yapılandırma portu için çıkış veri yolu
    drdy_out       : out  STD_LOGIC;                        -- Dinamik yeniden yapılandırma portu için veri hazır sinyali
    dclk_in        : in  STD_LOGIC;                         -- Dinamik yeniden yapılandırma portu için saat girişi
    reset_in       : in  STD_LOGIC;                         -- Sistem Monitörü kontrol mantığı için sinyali sıfırla
    vauxp14        : in  STD_LOGIC;                         -- Yardımcı Kanal 14
    vauxn14        : in  STD_LOGIC;
    busy_out       : out  STD_LOGIC;                        -- ADC Meşgul sinyali
    channel_out    : out  STD_LOGIC_VECTOR (4 downto 0);    -- Kanal Seçim Çıktıları
    eoc_out        : out  STD_LOGIC;                        -- Dönüşüm Sonu Sinyali
    eos_out        : out  STD_LOGIC;                        -- Sıra Sonu Sinyali
    alarm_out      : out STD_LOGIC;                         -- VEYA Tüm Alarmların çıktısı
    vp_in          : in  STD_LOGIC;                         -- Özel Analog Giriş Çifti
    vn_in          : in  STD_LOGIC
);
end xadc_wiz_0;

architecture xilinx of xadc_wiz_0 is

  attribute CORE_GENERATION_INFO : string;
  attribute CORE_GENERATION_INFO of xilinx : architecture is "xadc_wiz_0,xadc_wiz_v3_3_5,{component_name=xadc_wiz_0,enable_axi=false,enable_axi4stream=false,dclk_frequency=100,enable_busy=t
```

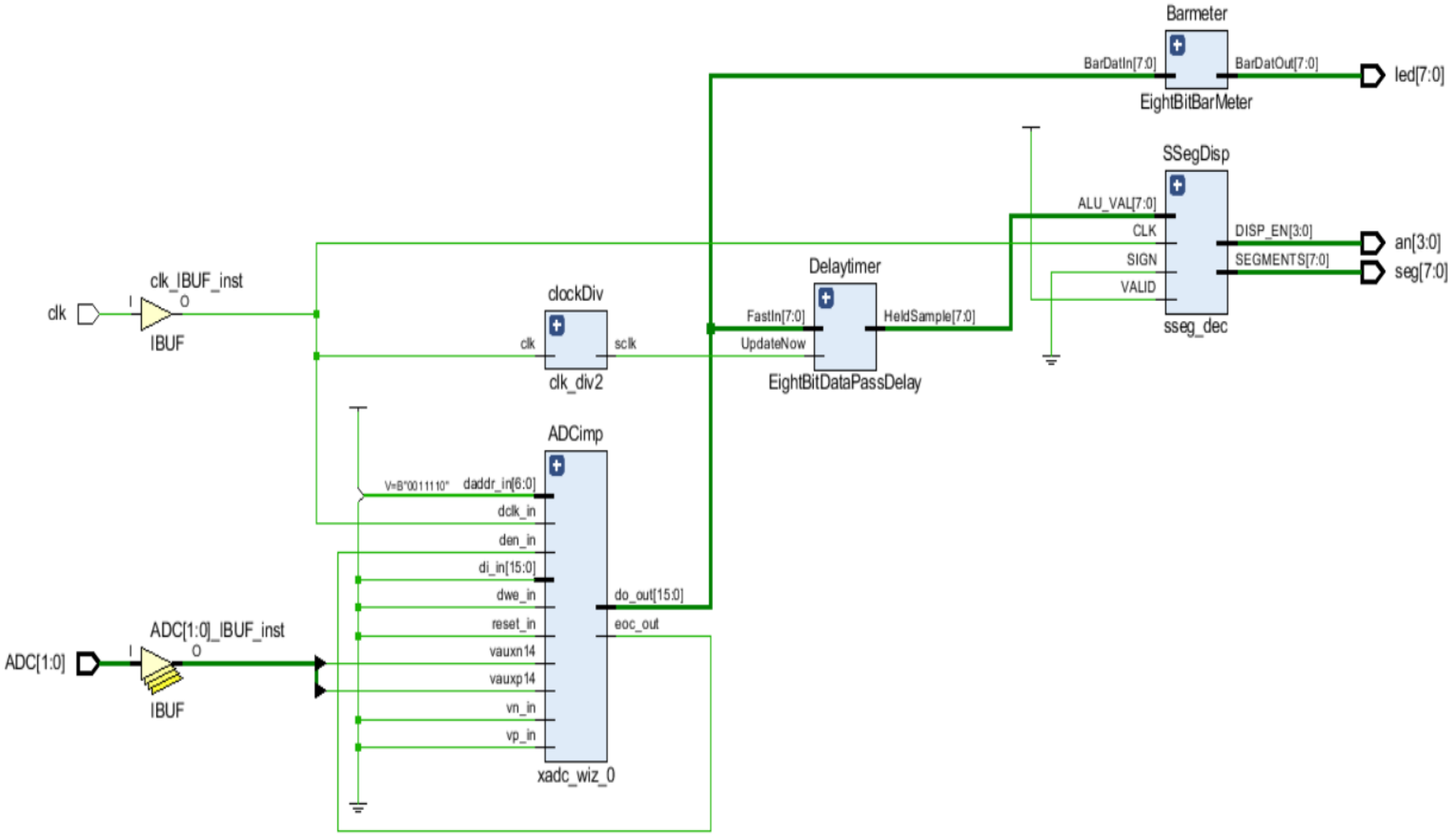
```
begin

    alarm_out <= alm_int(7);

     aux_channel_p(0) <= '0';
     aux_channel_n(0) <= '0';

     aux_channel_p(1) <= '0';
     aux_channel_n(1) <= '0';

     aux_channel_p(2) <= '0';
     aux_channel_n(2) <= '0';

     aux_channel_p(3) <= '0';
     aux_channel_n(3) <= '0';

     aux_channel_p(4) <= '0';
     aux_channel_n(4) <= '0';

     aux_channel_p(5) <= '0';
     aux_channel_n(5) <= '0';

     aux_channel_p(6) <= '0';
     aux_channel_n(6) <= '0';

     aux_channel_p(7) <= '0';
     aux_channel_n(7) <= '0';

     aux_channel_p(8) <= '0';
     aux_channel_n(8) <= '0';

     aux_channel_p(9) <= '0';
     aux_channel_n(9) <= '0';
     aux_channel_p(13) <= '0';
     aux_channel_n(13) <= '0';

     aux_channel_p(14) <= vauxp14;
     aux_channel_n(14) <= vauxn14;

     aux_channel_p(15) <= '0';
     aux_channel_n(15) <= '0';

U0 : XADC
    generic map(
       INIT_40 => X"001E", -- yapılandırma reg 0
       INIT_41 => X"31AF", -- yapılandırma reg 1
       INIT_42 => X"0400", -- yapılandırma reg 2
       INIT_48 => X"0100", -- Sıralayıcı kanalı seçimi
       INIT_49 => X"0000", -- Sıralayıcı kanalı seçimi
       INIT_4A => X"0000", -- Sıralayıcı Ortalama seçim
       INIT_4B => X"0000", -- Sıralayıcı Ortalama seçim
       INIT_4C => X"0000", -- Sıralayıcı Bipolar seçimi
       INIT_4D => X"0000", -- Sıralayıcı Bipolar seçimi
       INIT_4E => X"0000", -- Sıralayıcı Acq zaman seçimi
       INIT_4F => X"0000", -- Sıralayıcı Acq zaman seçimi
       INIT_50 => X"B5ED", -- Sıcaklık alarmı tetikleyicisi
       INIT_51 => X"57E4", -- Vccint üst alarm limiti
       INIT_52 => X"A147", -- Vccaux üst alarm limiti
       INIT_53 => X"CA33",  -- Sıcaklık alarmı OT üst
       INIT_54 => X"A93A", -- Sıcaklık alarmı sıfırlama
       INIT_55 => X"52C6", -- Vccint alt alarm limiti
       INIT_56 => X"9555", -- Vccaux alt alarm limiti
       INIT_57 => X"AE4E",  -- Sıcaklık alarmı OT reset
       INIT_58 => X"5999",  -- Vccbram üst alarm limiti
       INIT_5C => X"5111",  -- Vccbram alt alarm limiti
       SIM_DEVICE => "7SERIES",
       SIM_MONITOR_FILE => "design.txt"
       )
port map (
       CONVST             => '0',
       CONVSTCLK          => '0',
       DADDR(6 downto 0)  => daddr_in(6 downto 0),
       DCLK               => dclk_in,
       DEN                => den_in,
       DI(15 downto 0)    => di_in(15 downto 0),
       DWE                => dwe_in,
       RESET              => reset_in,
       VAUXN(15 downto 0) => aux_channel_n(15 downto 0),
       VAUXP(15 downto 0) => aux_channel_p(15 downto 0),
       ALM                => alm_int,
       BUSY               => busy_out,
       CHANNEL(4 downto 0) => channel_out(4 downto 0),
       DO(15 downto 0)    => do_out(15 downto 0),
       DRDY               => drdy_out,
       EOC                => eoc_out,
       EOS                => eos_out,
       JTAGBUSY           => open,
       JTAGLOCKED         => open,
       JTAGMODIFIED       => open,
       OT                 => open,

       MUXADDR            => FLOAT_MUXADDR,
       VN                 => vn_in,
       VP                 => vp_in
        );
end xilinx;
```

Master XDC;

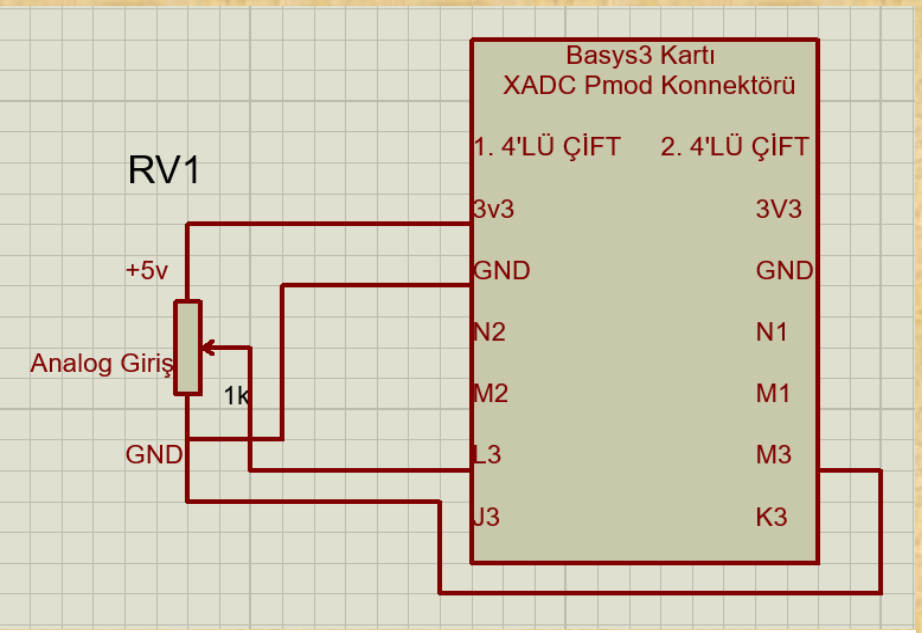
```
## LEDs
set_property PACKAGE_PIN U16 [get_ports {led[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {led[0]}]
set_property PACKAGE_PIN E19 [get_ports {led[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {led[1]}]
set_property PACKAGE_PIN U19 [get_ports {led[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {led[2]}]
set_property PACKAGE_PIN V19 [get_ports {led[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {led[3]}]
set_property PACKAGE_PIN W18 [get_ports {led[4]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {led[4]}]
set_property PACKAGE_PIN U15 [get_ports {led[5]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {led[5]}]
set_property PACKAGE_PIN U14 [get_ports {led[6]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {led[6]}]
set_property PACKAGE_PIN V14 [get_ports {led[7]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {led[7]}]
set_property PACKAGE_PIN V13 [get_ports {led[8]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {led[8]}]
set_property PACKAGE_PIN V3 [get_ports {led[9]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {led[9]}]
set_property PACKAGE_PIN W3 [get_ports {led[10]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {led[10]}]
set_property PACKAGE_PIN U3 [get_ports {led[11]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {led[11]}]
set_property PACKAGE_PIN P3 [get_ports {led[12]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {led[12]}]
set_property PACKAGE_PIN N3 [get_ports {led[13]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {led[13]}]
set_property PACKAGE_PIN P1 [get_ports {led[14]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {led[14]}]
set_property PACKAGE_PIN L1 [get_ports {led[15]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {led[15]}]
```

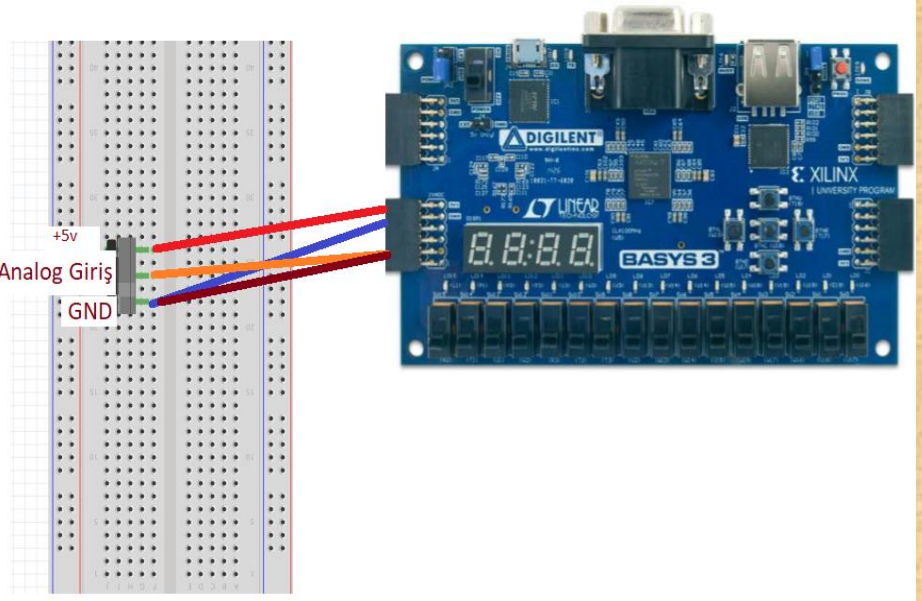
```
##7 segment display
set_property PACKAGE_PIN W7 [get_ports {seg[7]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {seg[7]}]
set_property PACKAGE_PIN W6 [get_ports {seg[6]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {seg[6]}]
set_property PACKAGE_PIN U8 [get_ports {seg[5]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {seg[5]}]
set_property PACKAGE_PIN V8 [get_ports {seg[4]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {seg[4]}]
set_property PACKAGE_PIN U5 [get_ports {seg[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {seg[3]}]
set_property PACKAGE_PIN V5 [get_ports {seg[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {seg[2]}]
set_property PACKAGE_PIN U7 [get_ports {seg[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {seg[1]}]

set_property PACKAGE_PIN V7 [get_ports {seg[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {seg[0]}]

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

Vhdl Kodları ile Oluşturulan RTL Şeması



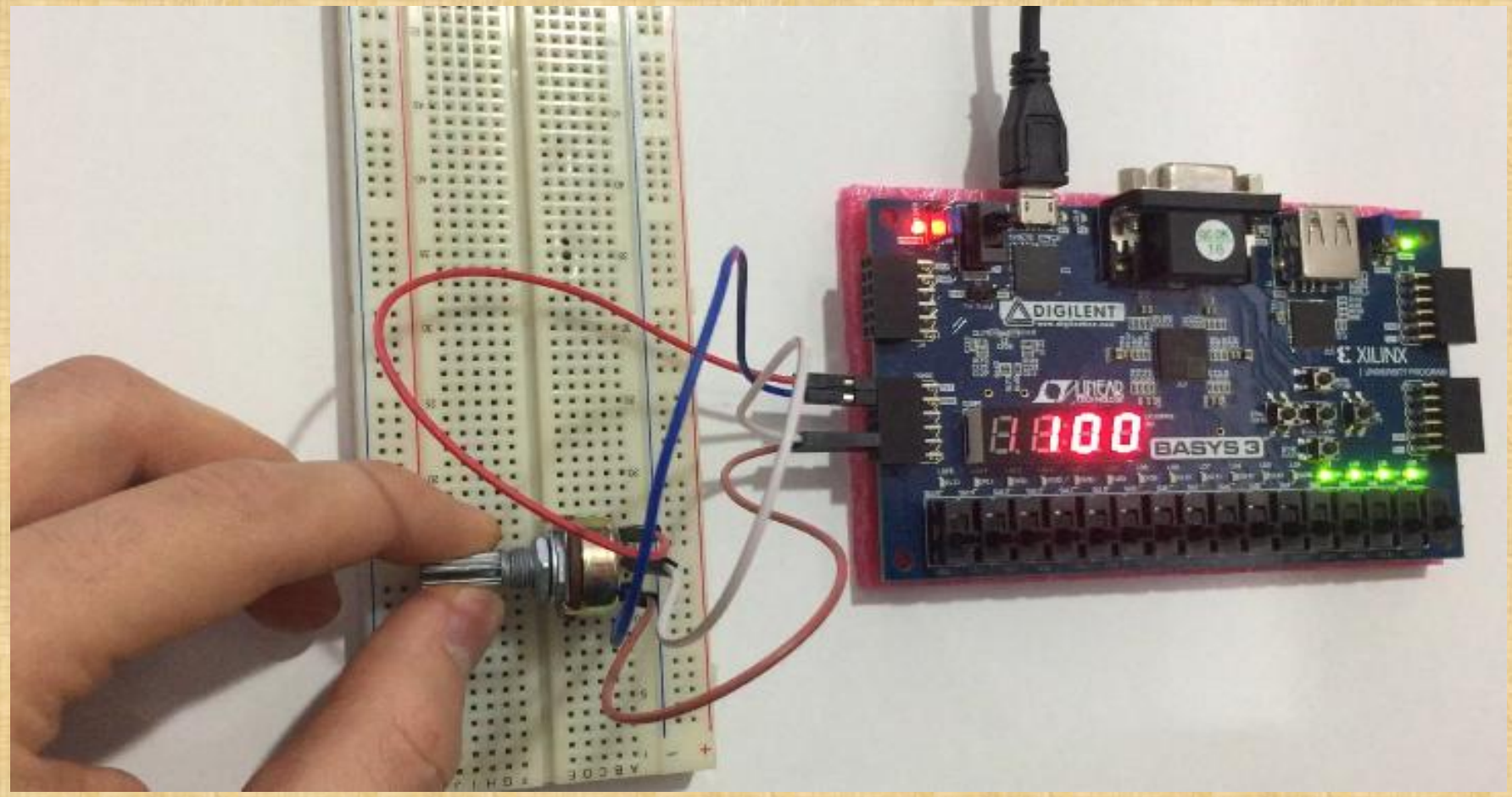
Potansiyometre’nin Kart ile Bağlantısı





Potansiyometremizin
1. bacağını(+5v) Basys-3 kartımızdaki 3v3 çıkışına bağlıyoruz.
2.bacağını (Analog Giriş) ise L3 çıkışına bağlıyoruz.
3. bacağını(GND) ise ilk önce gnd kısmına aynı 3. bacaktan bir çıkış daha alıp L3’e paralel bağlı olan M3 bacağına bağlıyoruz.

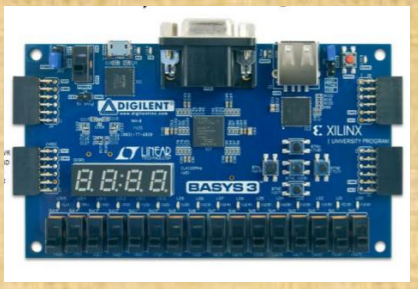
UYGULAMA DEVRE ŞEMASI



Şekil5: XADC UYGULAMASI

Maliyet Hesabı

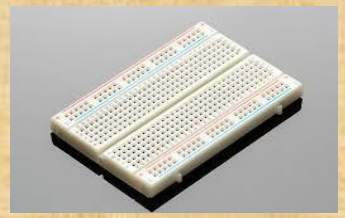
- Basys-3 kartı: 1200tl



- Potansiyometre: 1tl

- Jumper: 5 tl

- Bread Board : 10 tl



- Toplam: 1216 tl

Kaynakça

- https://yadi.sk/i/INNJKcSsQ8pv5A
- http://blog.aku.edu.tr/ismailkoyuncu/fpga/
- https://www.elektrovadi.com/BASYS3,PR-1964.html
- http://www.elektrikde.com/potansiyometre-nedir/http://www.robotiksistem.com/transistor_nedir_transistor_cesitleri.html
- https://www.xilinx.com/products/technology/analog-mixed-signal.html
- https://www.xilinx.com/support/documentation/user_guides/ug580-ultrascale-sysmon.pdf
- https://www.youtube.com/watch?v=bWWxZ9fWnKg