UŞAK ÜNİVERSİTESİ

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ
2022-2023 GÜZ DÖNEMİ

DİJİTAL SİSTEMLER
DERSİ ÖDEV RAPORU

Ödev No : Ödev 1

Ödev Tarihi :12 Aralık Salı 2022 Ödev Teslim Tarihi: 25 Aralık Pazar 2022

Adı Soyadı: Şerif Batıkan Çobanoğlu Öğrenci No:190517012

ÖDEV SORULARI

Soru 1) Bir MATLAB ortamında bir ses sinyalini yükledikten sonra, sinyale 1 db gürültü ekleyiniz.

a) Bu işlemi yapan MATLAB kodlarını geliştiriniz. MATLAB ortamında sinyaller elde edildikten sonra, elde ettiğiniz diziyi bir .txt dosyasına yazınız.

b) Gerekli olan AGF'yi fdatool kullanarak oluşturunuz. Oluşturduğunuz filtre katsayılarını bir txt dosyasına yazınız.

c) VHDL'de oluşturduğunuz txt dosyasından ses sinyalini ve filtre katsayıları dizilerini oluşturunuz. (Satır satır okuma ve diziye atmayı yapınız)

d) VHDL'de sinyal filtreleme döngüsünü yazınız ve filtrelenmiş sinyali bir txt'ye yazarak MATLAB ortamında tekrar okutunuz ve sinyalin filtrelendiğini gösteriniz.

Sorunun Amacı: Bu sorunun amacı derste öğrenilen FPGA ve VHDL programını öğrendiğimiz teknikler ve önceki senelerden almış olduğumu Matlab dersine bağlı olarak sinyal filtrelemeyi, filtrelenen ve oluşturulan sinyali txt dosyasına yazmayı ve yazılan dosyayı okumaya yöneliktir.

Soru Adımları: Öncelikli olarak kendi sesime ait bir ses dosyasını Matlab'yükledik. Daha sonra bu ses dosyasına Matlab komutları ile istenildiği gibi 1db gürültü yüklendi. Daha sonra AGF tasarımında bulunan Fpass ve Fstop değerlerini belirlemek için sinyalin spekturumunu çizdirdik. Daha sonra sinyal katsayılarını ve sinyala ait değerleri txt dosyasına yazdırdık. Daha sonra VHDL programından önce sesin txt dosyasını okuduk ve filtre için gerekli işlemleri yaptırdıktan sonra yeni bir txt dosyasına yazdırdık. Tekrardan bu yazılan dosyayı Matlab üzerinden okuyup doğruluğunu test ettik.

Yazılan Matlab Kodları:

```
clc;

clear all;

[y Fs]=audioread('C:\Users\Batikan\Desktop\sound\test.ogg');%bilgisayardaki ses dosyasını okudum

%sound(y,Fs);%daha duyulabilir olması için sesi yükselttim

%1 db gürültü ekledik

x = awgn(y,1,'measured');

plot([y x]);

legend('Original Signal','Signal with AWGN')

%sound(y); %ses çaldırmak için

%ses sinyali ve gürültülü sinyali grafiğe çizdirdim
```

```
subplot(2,2,1);
title('ses sinyali');
plot(y);
subplot(2,2,2);
plot(x);
title('ses ve gürültü birleşimi sinyali');
writematrix(x,'C:\Users\Batikan\Desktop\sound\sinyal.txt')%oluşturulan x matrisini txt dosyasına yazdırdık.
N=length(x);
[F]=fft(x)/N;
plot(abs(F));
%Frekans spekturumları için bunu yazmadan kodu çalıştırırsak gürültülü
%sinyali veriyor.Bununla açınca speekturumu çizdiriyor.
 Freq_Spec=[F(1); 2*abs(F(2:round(N/2)+1))];
 f=linspace(0,Fs,N);

 figure;
 stem(f(1:round(N/2)+1),Freq_Spec)
%
figure;
Freq_Spec_Full=[F(1); 2*abs(F(2:end))];
stem(f,Freq_Spec_Full);
```

Matlab sinyal çıktıları ve fdatool üzerinden katsayı değerleri:

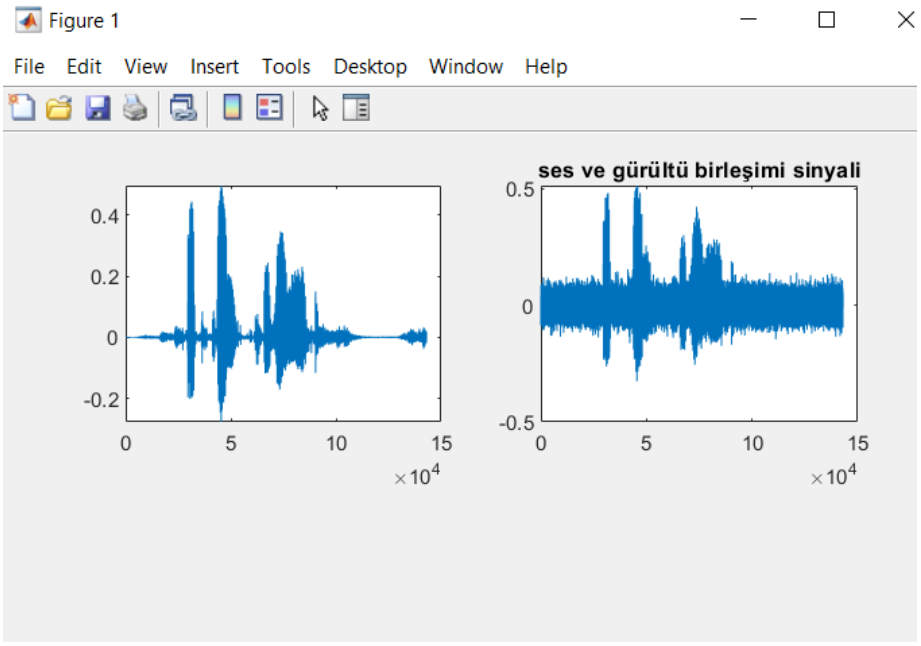

Figure 1 Matlab Sinyal ve Gürültü

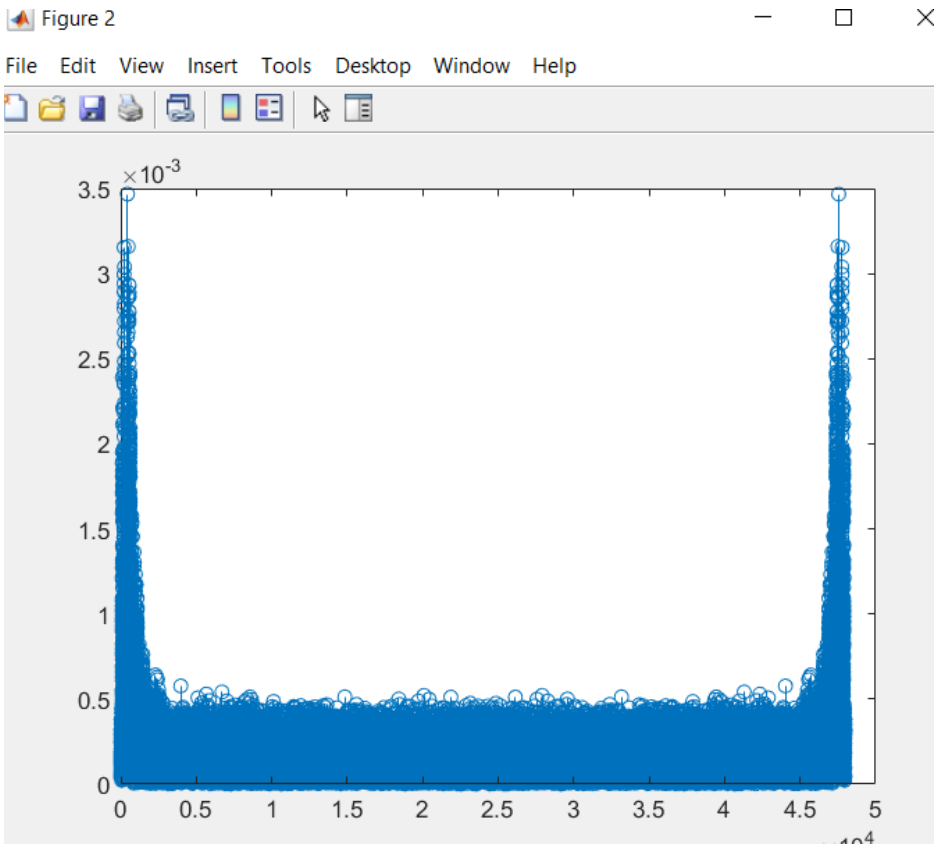

Figure 2 Frekans Spekturumu

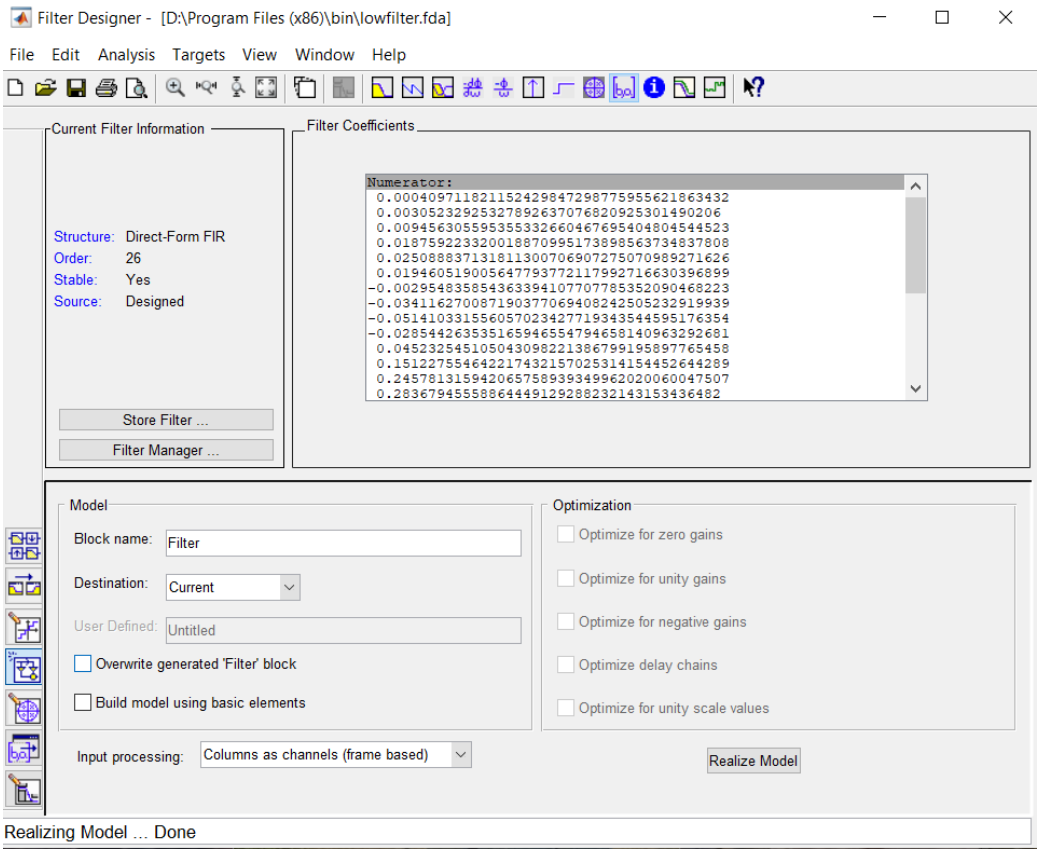

Figure 3 Filtre Katsayıları

VHDL KODLARI :

VHDL Kod Açıklaması: Filtre katsayı değerlerimiz noktalı sayılar olduğu için integer yerine real ile sinyallerimizi tanımladık.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use std.textio.all;


entity ghh is
-- Port (
```

```
end ghh;

architecture Behavioral of ghh is
signal clk :std_logic:='0';
type sin1 is array (0 to 999) of real ;
signal ss : sin1 :=(others=>0) ;
type filt is array (0 to 10) of real;
signal ff : filt :=(others=>0) ;
signal start : std_logic :='0';

begin
process
begin
clk <= '1';
wait for 300 us;
clk <= '0';
wait for 300 us;
end process;
process (clk)

file oku1 : text open read_mode is "C:\Users\Batikan\Desktop\sound\sinyal.txt" ;
file oku2 : text open read_mode is "C:\Users\Batikan\Desktop\sound\katsayilar.txt" ;
file yaz : text open write_mode is "C:\Users\Batikan\Desktop\sound\yn1.txt" ;
variable s_oku1 : line;
variable s_oku2 : line;
variable s_yaz : line;
variable s_yaz1 : line;
variable s_yaz2 : line;
variable data1 : real;
variable dgr1 : real;
```

```
variable data2 : real;
variable dgr2 : real;
variable cnt : real;



begin

if clk = '1' then

if not endfile(oku1) then

readline(oku1 , s_oku1);
read (s_oku1, data1);
ss(cnt)<=data1;
end if;
if not endfile(oku2) then
readline(oku2 , s_oku2);
read (s_oku2, data2);
ff(cnt)<=data2;
end if;
cnt :=cnt+1;


if endfile(oku1) then
start <= '1';
end if;
end if ;

end process;
```

```
process (start)
file yaz : text open write_mode is "C:\Users\Batikan\Desktop\sound\sinyeni.txt" ;
variable s_oku1 : line;
variable s_oku2 : line;
variable s_yaz : line;
variable x : real;
variable cnt1 : real ;
variable kmin : real ;
variable kmax : real ;
begin

if start = '1' then
for i in 0 to 1110 loop
x:=0;
        if i>=10 then
          kmin := i - 10;
        else
          kmin:= 0;
        end if;
        if i<999 then
          kmax:=i;
        else
          kmax:=1110;
        end if;
        for k in kmin to kmax loop
         x :=  x + ss(k) * ff(i-k+1)/1000000 ;
 end loop;
write(s_yaz,x);
writeline (yaz, s_yaz);
end loop;
```

```
end if ;

end process ;

end Behavioral;
```

Soru 2) $x(t)$ = 10 cos(2 ∗ pi ∗ 10000 ∗ t) sinyali 0.2 saniye boyunca 100 KHz örnekleniyor. Sonrasında sinyale ana harmoniğin %5'i kadar 3. Harmonik ve %10'u kadar 5. Harmonik ekleniyor. Daha sonra sinyalinizi bir Alçak Geçiren Filtre(AGF)'den geçirdikten sonra sinyali tekrar elde ediniz.

a) Bu işlemi yapan MATLAB kodlarını geliştiriniz. MATLAB ortamında sinyaller elde edildikten sonra, elde ettiğiniz diziyi bir .txt dosyasına yazınız.

b) Gerekli olan AGF'yi fdatool kullanarak oluşturunuz. Oluşturduğunuz filtre katsayılarını bir txt dosyasına yazınız.

c) VHDL'de oluşturduğunuz txt dosyasından sinüs sinyalini ve filtre katsayıları dizilerini oluşturunuz. (Satır satır okuma ve diziye atmayı yapınız)

d) VHDL'de sinyal filtreleme döngüsünü yazınız ve filtrelenmiş sinyali bir txt'ye yazarak MATLAB ortamında tekrar okutunuz ve sinyalin filtrelendiğini gösteriniz.

Sorunun Amacı: Matlab'de sinyal tanımlamayı ve harmonik açılımı yapmayı öğrenmek. Txt dosyasına bu sinyali yazdırmayı ve bu sinyali fdatool yardımı ile filtrelemeyi daha sonrasında ise bu işlemleri VHDL üzerinden yapabilmek.

Oluşturulan Matlab Kodları:

```
clc;

clear all;

fs = 100000;

Ts=1/fs;

fNy = fs / 2;

duration = 0.2;

t = 0 : Ts : duration-Ts;

noSamples = length(t);

x=10*cos(2*pi*10000*t);

x3 = 100.*sin(2 .* pi .* 500 .* t);

x5 = 100.*sin(2 .* pi .* 1000 .* t);

xn= x + x3+ x5;

f = 0 : fs/noSamples : fs - fs/noSamples;
```

```
x_fft = abs(fft(x));
xn_fft = abs(fft(xn));
figure(1);
subplot(2,2,1);
plot(t, x);
subplot(2,2,2);
plot(t, xn);
subplot(2,2,3);
plot(f,x_fft);
xlim([0 fNy]);
subplot(2,2,4);
plot(f,xn_fft);
xlim([0 fNy]);
writematrix(xn,'C:\Users\Batikan\Desktop\soru1\sinyals2.txt')
```

Yazılan VHDL Kodları:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use std.textio.all;


entity ghh is
-- Port (



end ghh;
```

```
architecture Behavioral of ghh is
signal clk :std_logic:='0';
type sin1 is array (0 to 999) of real ;
OKUNAN TXT'NİN İÇİNE YAZILACAK ARRAY/REAL TANIMLANDI
signal ss : sin1 :=(others=>0) ;
type filt is array (0 to 10) of real;
FİLTRE KATSAYILARI'NIN YAZILI OLDUGU TXT'NİN İÇİNE YAZILACAĞI
ARRAY/REAL TANIMLANDI
signal ff : filt :=(others=>0) ;
signal start : std_logic :='0';

begin
process
begin
clk <= '1';
wait for 300 us;
clk <= '0';
wait for 300 us;
end process;
MANUEL OLARAK CLK TANIMLANDI
process (clk)

file oku1 : text open read_mode is "C:\Users\Batikan\Desktop\soru1\sinyals2.txt" ;
file oku2 : text open read_mode is "C:\Users\Batikan\Desktop\soru1\agfkatsayilar.txt" ;
file yaz : text open write_mode is "C:\Users\Batikan\Desktop\soru1\sinyeni.txt" ;
variable s_oku1 : line;
variable s_oku2 : line;
variable s_yaz : line;
variable s_yaz1 : line;
variable s_yaz2 : line;
variable data1 : real;
```

```
variable dgr1 : real;
variable data2 : real;
variable dgr2 : real;
variable cnt : real;



begin

if clk = '1' then

if not endfile(oku1) then

readline(oku1 , s_oku1);
read (s_oku1, data1);
ss(cnt)<=data1;
end if;
```
TXT OKUNDU VE SİNYAL OLUŞTURUAL ARRAY'E YERLEŞTİRİLDİ
```
if not endfile(oku2) then
readline(oku2 , s_oku2);
read (s_oku2, data2);
ff(cnt)<=data2;
end if;
```
FİLTRE KATSAYILARI OKUNDU VE ARRAY'E YERLEŞTİRİLDİ
```
cnt :=cnt+1;
if endfile(oku1) then
start <= '1';
end if;
end if ;
end process;
```

```
process (start)

file yaz : text open write_mode is "C:\Users\Batikan\Desktop\sound\sinyeni.txt" ;
variable s_oku1 : line;
variable s_oku2 : line;
variable s_yaz : line;
variable x : real;
variable cnt1 : real ;
variable kmin : real ;
variable kmax : real ;
begin
if start = '1' then
for i in 0 to 1110 loop
x:=0;
      if i>=10 then
        kmin := i - 10;
      else
        kmin:= 0;
      end if;
      if i<999 then
        kmax:=i;
      else
        kmax:=1110;
      end if;
      for k in kmin to kmax loop
       x :=  x + ss(k) * ff(i-k+1)/1000000 ;
FİLTRE KATSAYILARI TXT'E YAZILIRKEN VHDL'DEN OKUNABİLSİN DİYE 10^6
İLE ÇARPILDI VE ÇARPILAN İŞLEM GERİ ALINDI
 end loop;
```

```
write(s_yaz,x);
writeline (yaz, s_yaz);
end loop;
end if ;
end process ;
end Behavioral;
```

Matlab Harmonik Çıktıları:

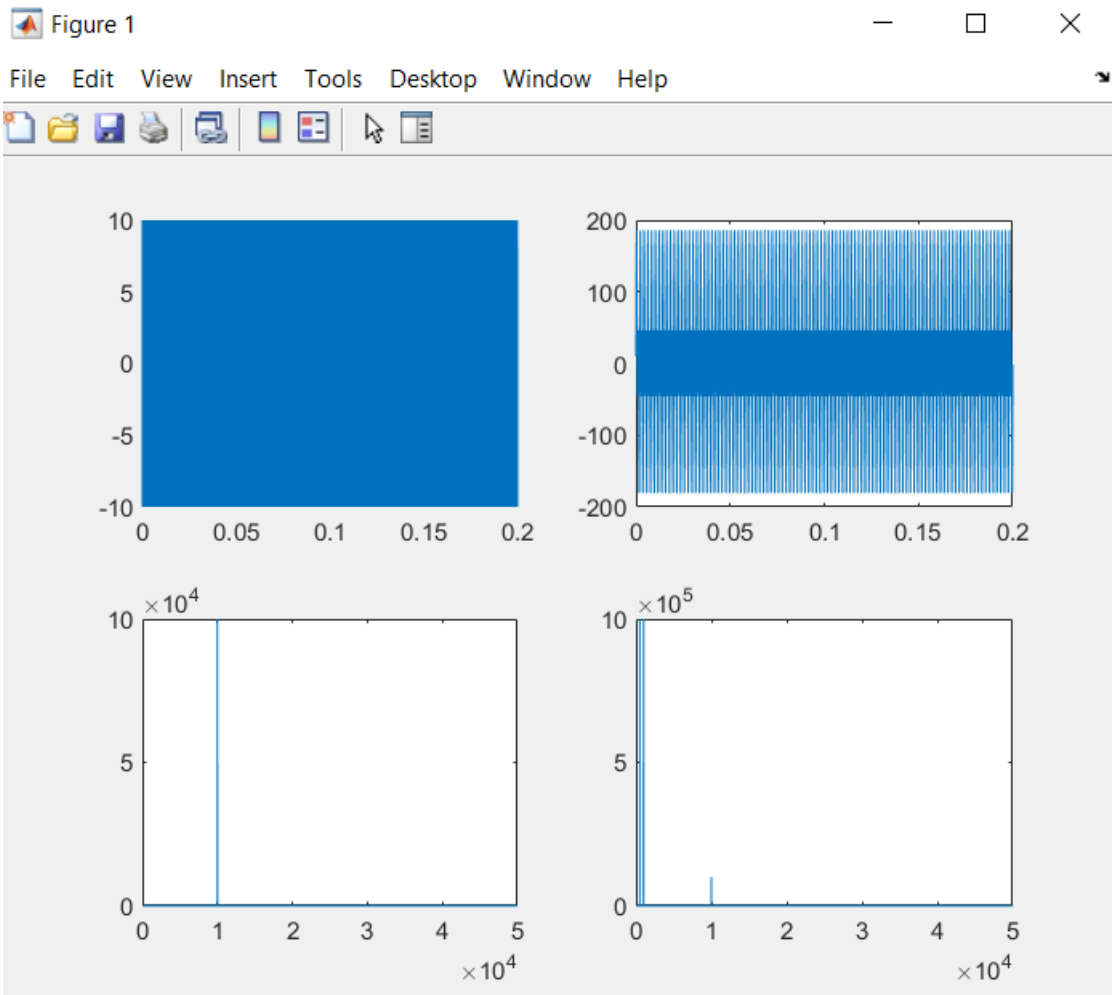

Figure 4 Matlab Harmonik Çıktıları

Soru 3) VHDL ortamında bir imge için (bu aşamada herkes kendi resmini çekip, o dosyayı kullanacaktır)

a. Aynalama

b. Ters Döndürme

c. Negatifleme

d. Karşıtlık Değiştirme

e. Parlaklık Değiştirme

f. Eşikleme

g. Sobel, Prewitt Kenar bulma filtrelerini Gerçekleştirecek VHDL kodlarını yazınız.

Sorunun Amacı: Matlab üzerinden görüntü elde edebilme elde edilen görüntüyü txt dosyasına yazdırabilme ve daha sonrasında yazdırılan txt dosyasını VHDL ortamından okuyup gerekli işlemleri yaptırdıktan sonra tekrardan txt dosyasına yazdırıp Matlab ortamından yapılan işlem çıktısını elde edebilme.

Yazılan VHDL Kodları:

İşlemlerin Yapıldığı Vhd Kodları:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED. ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use work.ornek_paket.all;


entity temel_isleme is
generic (
IMGE_SATIR : integer := 8;
IMGE_SUTUN : integer := 8;
VERI_UZUNLUGU : integer := 8
 );
 Port (
 in_clk : in std_logic;
in_rst : in std_logic;
in_en : in std_logic;
in_basla : in std_logic;
in_islem : in std_logic_vector(2 downto 0);
in_data : in std_logic_vector(VERI_UZUNLUGU - 1 downto 0);
in_data_vld : in std_logic;
out_addr : out std_logic_vector(log2_int(IMGE_SATIR * IMGE_SUTUN) - 1 downto 0);
```

```
out_addr_vld : out std_logic;
out_data : out std_logic_vector(7 downto 0);
out_data_vld : out std_logic;
out_tamam : out std_logic

 );
end temel_isleme;

architecture Behavioral of temel_isleme is
--constant AYNALAMA : std_logic_vector(2 downto 0) := "000";
--constant TERS_CEVIRME : std_logic_vector(2 downto 0) := "001";
--constant NEGATIFLEME : std_logic_vector(2 downto 0) := "000";
--constant ESIKLEME : std_logic_vector(2 downto 0) := "011";
--constant PARLAKLIK_ARTIR : std_logic_vector(2 downto 0) := "100";
constant PARLAKLIK_AZALT : std_logic_vector(2 downto 0) := "101";
--constant KARSITLIK_ARTIR : std_logic_vector(2 downto 0) := "110";
--constant KARSITLIK_AZALT : std_logic_vector(2 downto 0) := "111";

type t_Imge_Isleme is (BOSTA, RAMDAN_OKU, OKUMA_BEKLE, ISLEM_YAP, SAYAC_KONT,
TAMAM );

signal r_Imge_Isleme : t_Imge_Isleme := RAMDAN_OKU;
signal n_i: integer := 0;
signal n_j : integer := 0;
signal r_addr : std_logic_vector(log2_int (IMGE_SATIR * IMGE_SUTUN) - 1 downto 0) := (others =>
'0');
signal r_addr_vld : std_logic := '0';
signal r_data : std_logic_vector(7 downto 0) := (others => '0');
signal r_data_vld : std_logic := '0';
signal r_tamam : std_logic := '0';
begin
out_addr <= r_addr;
out_addr_vld <= r_addr_vld;
out_data <= r_data;
out_data_vld <= r_data_vld;
```

```
out_tamam <= r_tamam;
process (in_clk, in_rst)
begin
if in_rst = '1' then
r_Imge_Isleme <= BOSTA;
n_i <= 0;
n_j <= 0;
r_addr <= (others => '0');
r_addr_vld <= '0';
r_data <= (others => '0');
r_data_vld <= '0';
r_tamam <= '0';

elsif rising_edge(in_clk) then
if in_en = '1' then

r_data_vld <= '0';
r_addr_vld <= '0';
r_tamam <= '0';

case r_Imge_Isleme is
when BOSTA =>
if in_basla = '1' then
r_Imge_Isleme <= RAMDAN_OKU;
end if;
when RAMDAN_OKU =>
--if in_islem = AYNALAMA then
--r_addr <= conv_std_logic_vector(n_i * IMGE_SUTUN +(IMGE_SUTUN - 1 - n_j) , r_addr'length);
--if in_islem = TERS_CEVIRME then
--r_addr <= conv_std_logic_vector((IMGE_SATIR - 1 - n_i) * IMGE_SUTUN + n_j, r_addr'length);
--if in_islem = NEGATIFLEME then --or --
--if in_islem = ESIKLEME or in_islem = PARLAKLIK_ARTIR or in_islem = PARLAKLIK_AZALT or
in_islem = KARSITLIK_ARTIR or in_islem = KARSITLIK_AZALT then
if in_islem = PARLAKLIK_AZALT then
```

```
--if in_islem = KARSITLIK_ARTIR then
--if in_islem = ESIKLEME then
r_addr<= conv_std_logic_vector(n_i * IMGE_SUTUN + n_j, r_addr' length);
end if;

--if in_islem =AYNALAMA then
 --if in_islem = KARSITLIK_AZALT then
--r_addr<= conv_std_logic_vector(n_i * IMGE_SUTUN + n_j, r_addr' length);
--end if;

r_addr_vld <= '1';

r_Imge_Isleme <= OKUMA_BEKLE;

when OKUMA_BEKLE =>
if in_data_vld = '1' then
r_data <= in_data;
r_Imge_Isleme <= ISLEM_YAP;
end if;

when ISLEM_YAP =>
--if in_islem = AYNALAMA then --or in_islem = TERS_CEVIRME then
--if in_islem = TERS_CEVIRME then
--r_data<= r_data;
--end if;
--if in_islem = KARSITLIK_ARTIR then
--r_data <= r_data;
--if in_islem= NEGATIFLEME then --elsif in_islem = NEGATIFLEME then
--r_data <= 255 - r_data;
--if in_islem = ESIKLEME then
--if r_data > 128 then
--r_data <= conv_std_logic_vector(255, r_data'length);
--else
--r_data <= (others => '0');
```

```
--end if;
--end if;
--if in_islem = PARLAKLIK_ARTIR then
--if r_data > 210 then
--r_data <= conv_std_logic_vector (255, r_data' length);
--else
--r_data <= r_data + 45;
--end if;
if in_islem = PARLAKLIK_AZALT then
if r_data < 45 then
r_data <= conv_std_logic_vector(0, r_data' length) ;
else
r_data <= r_data - 45;
end if;
--if in_islem = KARSITLIK_ARTIR then
--if r_data>128 then
--r_data <= conv_std_logic_vector(255, r_data' length) ;
--else
--r_data <= r_data(6 downto 0) & '0';
--end if;
--end if;
--if in_islem = KARSITLIK_AZALT then
--if r_data>128 then
--else
--r_data <= '0' & r_data(7 downto 1);
end if;
--end if;
r_data_vld <= '1';
r_Imge_Isleme <= SAYAC_KONT;
when SAYAC_KONT =>
if n_j = IMGE_SUTUN - 1 then
n_j <= 0;
if n_i = IMGE_SATIR - 1 then
n_i <= 0;
```

```
r_Imge_Isleme <= TAMAM;
else
n_i<=n_i + 1;
r_Imge_Isleme <= RAMDAN_OKU;
end if;
else
n_j <= n_j + 1;
r_Imge_Isleme <= RAMDAN_OKU;
end if;

when TAMAM =>
r_tamam <= '1';

r_Imge_Isleme <= BOSTA;
when others => NULL;
end case;
end if;
end if;
end process;

end Behavioral;
```

Block Ram Kodları:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_unsigned.ALL;
use work.ornek_paket.all;

entity blok_ram is
generic(
VERI_UZUNLUGU : integer := 8;
RAM_DERINLIGI : integer := 110
);
  Port (
  in_clk : in std_logic;
```

```
in_rst : in std_logic;
in_ram_aktif: in std_logic;
in_yaz_en : in std_logic;
in_oku_en : in std_logic;
in_data_addr : in std_logic_vector (log2_int(RAM_DERINLIGI) - 1 downto 0);
in_data : in std_logic_vector(VERI_UZUNLUGU - 1 downto 0);
out_data : out std_logic_vector(VERI_UZUNLUGU - 1 downto 0);
out_data_vld : out std_logic
 );
end blok_ram;
architecture Behavioral of blok_ram is
type t_BRAM_DATA is array (0 to RAM_DERINLIGI - 1) of std_logic_vector(VERI_UZUNLUGU - 1
downto 0);
signal r_BRAM_DATA: t_BRAM_DATA:=(others=>(others=>'0'));
begin
process(in_clk,in_rst)
begin
if in_rst='1' then
r_BRAM_DATA <=(others=>(others=>'0'));
elsif rising_edge(in_clk)then
if  in_ram_aktif='1' then
if in_oku_en='1' then
out_data <= r_BRAM_DATA(conv_integer(in_data_addr));
out_data_vld <= '1';
else
out_data_vld <='0';
end if;
if in_yaz_en ='1' then
r_BRAM_DATA(conv_integer(in_data_addr)) <= in_data;
end if;
end if;
end if;
end process;

end Behavioral;
```

log2_int KOMUTUNUN TANIMLANDIĞI PACKAGE KODU:

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

package ornekler_paket is
function log2_int(in_giris : integer) return integer;
end ornekler_paket;

package body ornekler_paket is
function log2_int(in_giris : integer) return integer is
variable sonuc : integer;
begin
for i in 0 to 31 loop
if in_giris <= (2**i) then
sonuc := i;
exit;
end if;
end loop;
return sonuc;
end log2_int;
end package body;
```

Test Bench Kodları:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

use std.textio.ALL;

use work.ornek_paket.all;



entity test_b is

--  Port ( );

end test_b;


architecture Behavioral of test_b is

component temel_isleme

generic(


IMGE_SATIR : integer := 8;

IMGE_SUTUN : integer:= 8;
```

```
VERI_UZUNLUGU : integer:= 24
);
port (
in_clk : in std_logic;
in_rst : in std_logic;
in_en : in std_logic;
in_basla : in std_logic;
in_islem : in std_logic_vector(2 downto 0);
in_data : in std_logic_vector(VERI_UZUNLUGU - 1 downto 0);
in_data_vld : in std_logic;
out_addr : out std_logic_vector(log2_int(IMGE_SATIR * IMGE_SUTUN) - 1 downto 0);
out_addr_vld : out std_logic;
out_data : out std_logic_vector(7 downto 0);
out_data_vld : out std_logic;
out_tamam : out std_logic
);

end component;

component blok_ram

generic(
VERI_UZUNLUGU : integer := 8;
RAM_DERINLIGI : integer := 110
);
port(
in_clk : in std_logic;
in_rst : in std_logic;
in_ram_aktif : in std_logic;
in_yaz_en : in std_logic;
in_oku_en : in std_logic;
in_data_addr : in std_logic_vector (log2_int(RAM_DERINLIGI) - 1 downto 0);
in_data : in std_logic_vector(VERI_UZUNLUGU - 1 downto 0);
out_data : out std_logic_vector(VERI_UZUNLUGU - 1 downto 0);
```

```
out_data_vld : out std_logic
);
end component;

constant CLK_PERIOD : time := 20 ns;
constant IMGE_SATIR : integer :=1024 ;
constant IMGE_SUTUN : integer := 1024;

constant VERI_UZUNLUGU : integer := 8;
constant VERI_YOLU_OKUMA : string := "C:\Users\Batikan\Desktop\islemeer\bb.txt";
constant VERI_YOLU_YAZMA : string := "C:\Users\Batikan\Desktop\islemeer\parlaklikaz.txt";

--constant AYNALAMA : std_logic_vector(2 downto 0) := "000";
--constant TERS_CEVIRME : std_logic_vector(2 downto 0) := "001";
--constant NEGATIFLEME : std_logic_vector(2 downto 0) := "000";
--constant ESIKLEME : std_logic_vector(2 downto 0) := "011";
--constant PARLAKLIK_ARTIR : std_logic_vector(2 downto 0) := "100";
constant PARLAKLIK_AZALT : std_logic_vector(2 downto 0) := "101";
--constant KARSITLIK_ARTIR : std_logic_vector(2 downto 0) := "110";
--constant KARSITLIK_AZALT : std_logic_vector(2 downto 0) := "111";
type t_Imge_Isleme is (RAM_OKUMA, RAM_YAZMA, TAMAM);
signal r_Imge_Isleme : t_Imge_Isleme := RAM_YAZMA;
signal in_clk : std_logic := '0';
signal in_rst : std_logic := '0';
signal in_basla : std_logic := '0';
signal in_ram_aktif : std_logic := '1';
signal out_data_vld : std_logic := '0';
signal out_data : std_logic_vector(7 downto 0) := (others => '0');
signal in_ram_data : std_logic_vector (VERI_UZUNLUGU - 1 downto 0) := (others => '0');
signal in_ram_data_addr : std_logic_vector(log2_int (IMGE_SATIR * IMGE_SUTUN) - 1 downto 0) :=
(others => '0');
signal out_ram_data : std_logic_vector (VERI_UZUNLUGU - 1 downto 0) := (others => '0');
signal out_data_addr : std_logic_vector (log2_int (IMGE_SATIR * IMGE_SUTUN) - 1 downto 0) :=
(others => '0');
signal out_ram_data_vld: std_logic := '0';
```

```
signal in_en : std_logic := '0';
signal in_yaz_en : std_logic := '0';
signal in_oku_en : std_logic := '0';
signal data_sayac : integer := 0;
signal out_data_addr_vld : std_logic := '0';
signal r_imge_isleme_tamam : std_logic := '0';

 begin

process

begin
in_clk <= '1';
wait for CLK_PERIOD / 2;
in_clk <= '0';
wait for CLK_PERIOD / 2;
end process;

process
begin
in_basla <= '1';
wait for CLK_PERIOD ;
in_basla <= '0';
wait;
end process;

process (in_clk)
file dosya_okuma : text open read_mode is VERI_YOLU_OKUMA;
file dosya_yazma : text open write_mode is VERI_YOLU_YAZMA;
variable satir_okuma : line;
variable satir_yazma : line;
variable data_okuma : integer;

begin
```

```
if rising_edge(in_clk) then
if out_data_vld = '1' then
write(satir_yazma, conv_integer(out_data));
writeline(dosya_yazma, satir_yazma) ;
end if;

case r_Imge_Isleme is
when RAM_YAZMA =>
if not endfile(dosya_okuma) then
readline (dosya_okuma, satir_okuma) ;
read(satir_okuma, data_okuma) ;

in_ram_data <= conv_std_logic_vector( data_okuma, VERI_UZUNLUGU) ;

in_ram_data_addr <= conv_std_logic_vector (data_sayac, in_ram_data_addr' length);
in_en <= '0';
in_yaz_en <= '1';
data_sayac <= data_sayac + 1;
else

r_Imge_Isleme <= RAM_OKUMA;
in_yaz_en <= '0';
end if;

when RAM_OKUMA =>

in_en <= '1' ;
in_ram_data_addr <= out_data_addr;
in_oku_en <= out_data_addr_vld ;
if r_imge_isleme_tamam = '1' then
r_Imge_Isleme <= TAMAM;
end if;

when TAMAM => null;
```

```
when others => NULL;
end case;
end if;
end process;

temel_isleme_map : temel_isleme
generic map (
IMGE_SATIR => IMGE_SATIR,
IMGE_SUTUN => IMGE_SUTUN,
VERI_UZUNLUGU => VERI_UZUNLUGU
)
port map(
in_clk => in_clk,
in_rst => in_rst,
in_en => in_en,
in_basla => in_basla,
in_islem =>PARLAKLIK_AZALT,
in_data => out_ram_data,
in_data_vld => out_ram_data_vld,
out_addr => out_data_addr,
out_addr_vld => out_data_addr_vld,
out_data => out_data,
out_data_vld => out_data_vld,
out_tamam => r_imge_isleme_tamam
);

blok_ram_map : blok_ram

generic map (
VERI_UZUNLUGU => VERI_UZUNLUGU,
RAM_DERINLIGI => IMGE_SATIR * IMGE_SUTUN
)

port map (
```

```
in_clk => in_clk,
in_rst => in_rst,
in_ram_aktif => in_ram_aktif,
in_yaz_en => in_yaz_en,
in_oku_en => in_oku_en,
in_data_addr=> in_ram_data_addr,
in_data => in_ram_data,
out_data => out_ram_data,
out_data_vld=> out_ram_data_vld
);

end Behavioral;
```

Yazılan Matlab Kodları:

```
clear all;
clc;
close all;

 %%Görüntü griye çevirilde ve okundu

imge = imread('C:\Users\Batikan\Desktop\islemeer\bsg.png');
image = rgb2gray(imge);
imshow(image)
[satir sutun] = size(image);

 %%Yapılan işlemin txt dosyası okundu
dosya = fopen('C:\Users\Batikan\Desktop\islemeer\parlaklikaz.txt', 'r')';
image_okunan = fscanf(dosya, '%d')';

COEF = [-1 -2 -1; 0 0 0; 1 2 1];
 %image_okunan(262144) = 0;
for n_i = 1 : satir
for n_j = 1 : sutun
yeni_image(n_i,n_j)= image_okunan(((n_i-1)*sutun) + n_j);

end
end

fclose (dosya) ;

figure,imshow(uint8(yeni_image))
```

Ekran Çıktıları :

Figure 5 Orijinal Fotoğraf

Figure 6 Aynalama İşlemi

Figure 7 Ters Döndürme İşlemi

Figure 8 Negatifleme İşlemi

Figure 9 Karşıtlık Azaltma

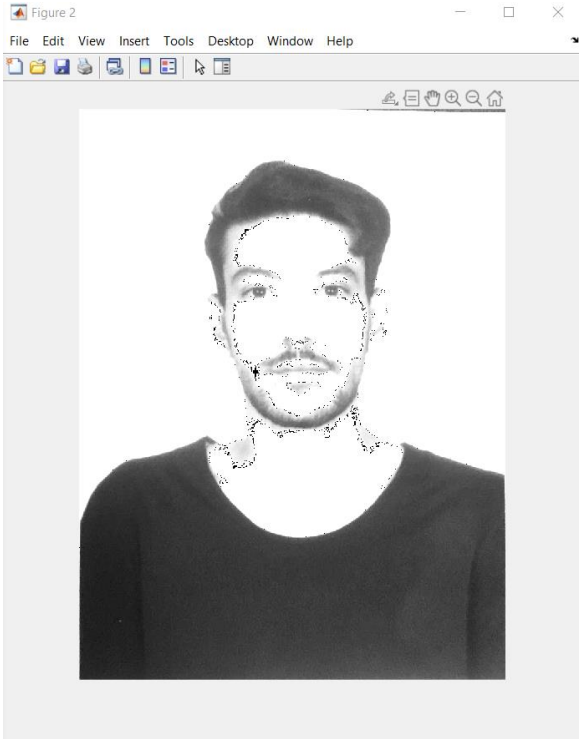

Figure 10 Karşıtlık Arttırma

Figure 11 Parlaklık Arttırma

Figure 12 Parlaklık Azaltma

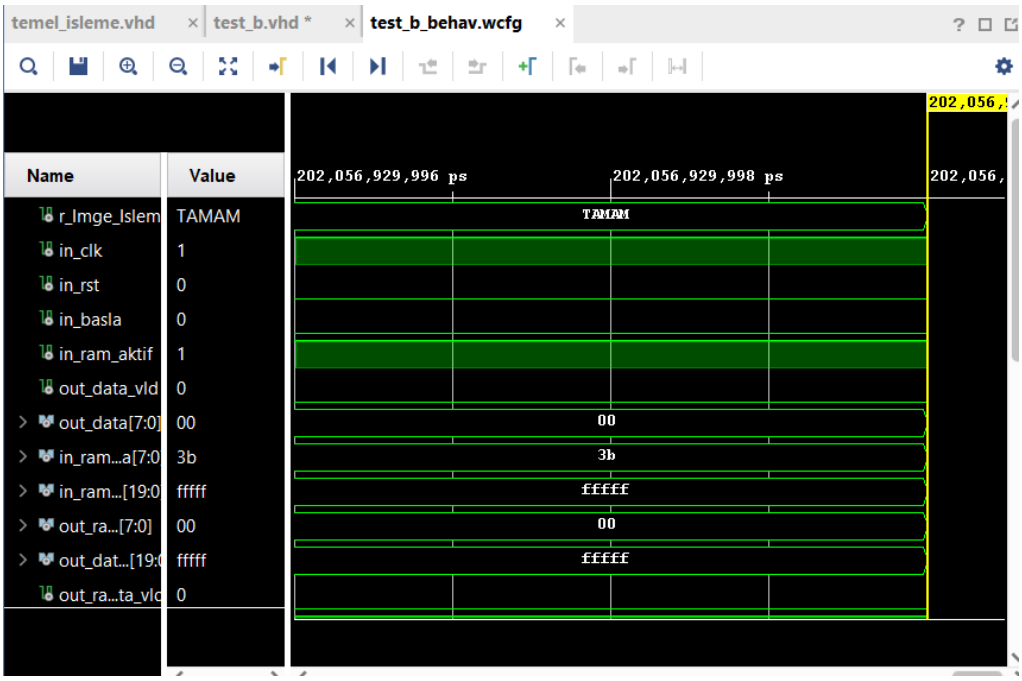

Figure 13 Test Bench Simulasyon

Sobell, Prewitt Kenar Bulma Filtreleri:

Sobel Matlab Kodları:

```
clear all;
clc;
close all;

imge = imread('C:\Users\Batikan\Desktop\islemeer\bsg.png');
y = rgb2gray(imge);
z=edge(y,'sobel');
imshow(z);
function [ ] = Sobel( adres )
r=imread(adres);
g=rgb2gray(r);
t=double(g);
[x,y]=size(t);
for i=2:x-1
for j=2:y-1
k1=t(i-1,j+1)+2*t(i,j+1)+t(i+1,j+1)...
-(t(i-1,j-1)+2*t(i,j-1)+t(i+1,j-1));
k2=t(i-1,j-1)+2*t(i-1,j)+t(i-1,j+1)...
-(t(i+1,j-1)+2*t(i+1,j)+t(i+1,j+1));

g(i,j)=sqrt(k1^2+k2^2);
end
end
imshow(g);
end
```

Prewitt Matlab Kodları:

```
clear all;
clc;
close all;

imge = imread('C:\Users\Batikan\Desktop\islemeer\bsg.png');
y = rgb2gray(imge);
z=edge(y,'prewitt');
imshow(z);
function [ ] = Prewitt(adres)
r=imread(adres);
g=rgb2gray(r);
t=double(g);
[x,y]=size(t);
for i=2:x-1
for j=2:y-1
k1=t(i-1,j+1)+t(i,j+1)+t(i+1,j+1)...
-(t(i-1,j-1)+t(i,j-1)+t(i+1,j-1));
k2=t(i+1,j-1)+t(i+1,j)+t(i+1,j+1)...
-(t(i-1,j-1)+t(i-1,j)+t(i-1,j+1));
g(i,j)=sqrt(k1^2+k2^2);
end
end
imshow(g);
end
```

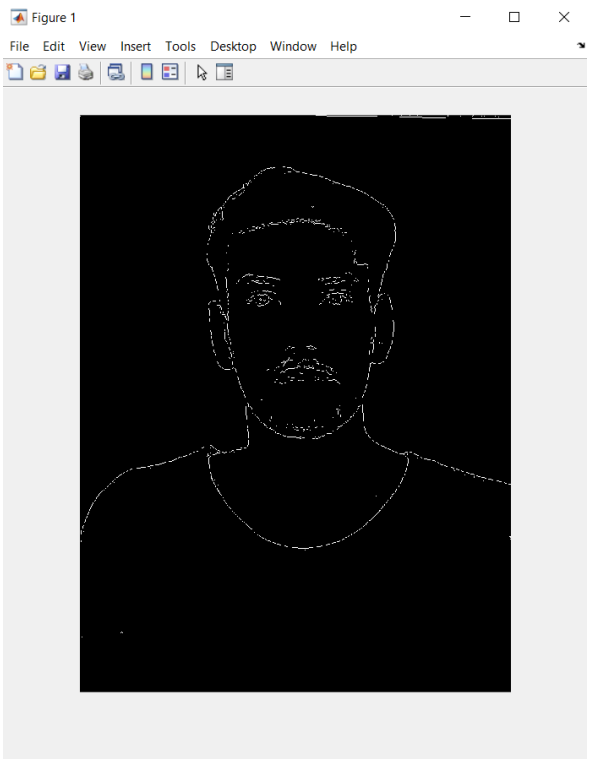

Figure 14 Sobel Ekran Çıktısı

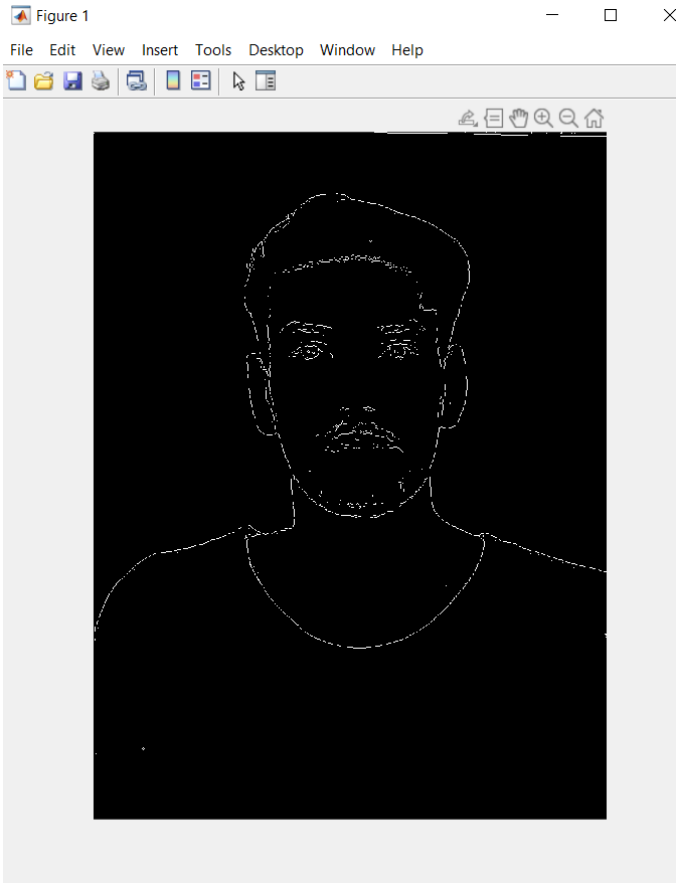

Figure 15 Prewitt Ekran Çıktısı

Soru 4)

a) Uart üzerinden gelen bir veri dizini için baudrate'i otomatik olarak hesaplayan VHDL kodunu yazınız.

b) Incremental bir encoder her bir turda, kanal başına 32000 pulse vermektedir. Buna göre bu encoderin bağlandığı motorun hızını rpm cinsinden hesaplayan VHDL kodunu yazınız.

A şıkkı için baud hızını hesaplayabilmek için clk freakansının üzerinden gitmemiz lazım. Bunun için bir clk frekans bölücü yapıldı. Daha sonra Ts=1/fs formülünden baud hızına ulaşıldı. Aynı zamanda simüle etmek için de test bench dosyası yazıldı.

VHDL Kodları:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use ieee.numeric_std.all;



entity clk_frek is

  Port (
```

```
clk_in : in  STD_LOGIC;
        reset  : in  STD_LOGIC;
        baud : out std_logic;
        clk_out: out STD_LOGIC
  );
end clk_frek;

architecture Behavioral of clk_frek is
signal temporal: STD_LOGIC;
    signal counter : integer range 0 to 124999 := 0;

begin
frequency_divider: process (reset, clk_in) begin
        if (reset = '1') then
            temporal <= '0';
            counter <= 0;
        elsif rising_edge(clk_in) then
            if (counter = 124999) then
                temporal <= NOT(temporal);
                counter <= 0;
            else
                counter <= counter + 1;
            end if;
        end if;
    end process;

    clk_out <= temporal;
 baud <= 1/temporal;
end Behavioral;
```

Test Bench Kodları:

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

```
entity clck is
 -- Port ( );
end clck;

architecture Behavioral of clck is
component clk_frek
Port(
clk_in : IN  std_logic;
                reset  : IN  std_logic;
                clk_out: OUT std_logic
);
end component;
signal clk_in  : std_logic := '0';
        signal reset   : std_logic := '0';
        signal clk_out : std_logic;
        constant clk_in_t : time := 20 ns;
begin
uut: clk_frek Port map(
clk_in  => clk_in,
                reset   => reset,
                clk_out => clk_out

);
entrada_process :process
                begin
                clk_in <= '0';
                wait for clk_in_t / 2;
                clk_in <= '1';
                wait for clk_in_t / 2;
        end process;


        stimuli: process
        begin
```

```
                reset <= '1';
                wait for 100 ns;
                reset <= '0';
        wait;
            end process;

end Behavioral;
```

Figure 16 Simulasyon Çıktısı

Soru 5) SPI üzerinden haberleşen bir ad7476 ADC entegresinin birinci kanalına 0-3V arasında değişen bir sinyal, ikinci kanalına ise -10 ile 10V arasında değişen bir sinyal veriliyor. Bu ADC'den alınan verilerin toplamını bulan hesaplayan VHDL kodunu yazınız.

VHDL Kodları:

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


entity soru_5 is
 Port (
```

```
clk_250: in std_logic;
sampling_pulse: in std_logic;
postscaler_in: in std_logic_vector(15 downto 0);
data_out: out std_logic_vector(15 downto 0) := (others => '0');

spi_sck: out std_logic;
spi_cs_n: out std_logic;
spi_din: in std_logic
   );
end soru_5;

architecture Behavioral of soru_5 is
TYPE states is (ACQ,CONV);
SIGNAL state : states := ACQ;
SIGNAL postscaled_clk : std_logic := '0';
SIGNAL postscaled_clk_rising_pulse : std_logic := '0';
SIGNAL pulse_detected : std_logic := '0';
begin

   spi_sck <= postscaled_clk;
   spi_cs_n <= '1' when state=ACQ else '0';
POSTSCALER: process(clk_250)
      variable postscaler_cnt: unsigned(15 downto 0):=(others=>'0');
   begin
      if rising_edge(clk_250) then
         postscaled_clk_rising_pulse <= '0';


         if postscaler_cnt+1 >= unsigned(postscaler_in) then
            if postscaled_clk = '0' then
               postscaled_clk_rising_pulse <= '1';
            end if;
            postscaler_cnt := (others => '0');
            postscaled_clk <= not postscaled_clk;
```

```
        else
            postscaler_cnt := postscaler_cnt + 1;
        end if;
    end if;
end process POSTSCALER;

-- Generate pulse_detected
SAMPLING: process(clk_250)
begin
    if rising_edge(clk_250) then
        if sampling_pulse = '1' then
            pulse_detected <= '1';
        elsif postscaled_clk_rising_pulse = '1' then
            pulse_detected <= '0';
        end if;
    end if;
end process SAMPLING;



FSM : process(clk_250)
    variable bit_cnt : unsigned(4 downto 0) := (others=>'0');
begin
    if rising_edge(clk_250) and postscaled_clk_rising_pulse = '1' then
        case state is

            when ACQ =>
                bit_cnt := (others => '0');
                if pulse_detected = '1' then
                    state <= CONV;
                end if;

            when CONV =>
                bit_cnt := bit_cnt + 1;
                if bit_cnt >= 16 then
```

```
                    state <= ACQ;
                end if;

            when others => null;
        end case;
    end if;
end process FSM;


SHIFT_REG: process (clk_250)
    variable data_reg: std_logic_vector(15 downto 0):=(others=>'0');
begin
    if rising_edge(clk_250) then
        if state = CONV and postscaled_clk_rising_pulse = '1' then
            data_reg := data_reg(14 downto 0) & spi_din;
        elsif state = ACQ then
            data_out <= "0" & data_reg(15 downto 1);
        end if;
    end if;
end process SHIFT_REG;


end Behavioral;
```

Test Bench Kodları:

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


entity tb_s5 is
--  Port ( );
end tb_s5;
```

```
architecture Behavioral of tb_s5 is
constant NBLANKBITS : positive := 1;

constant SCK_POSTSCALER : std_logic_vector := "0000000000000010";
constant CLK_PERIOD : time := 4.0 ns; -- 250 MHz
signal rawdata : unsigned(13 downto 0) := (others=>'0');
signal clk_250, sampling_pulse : std_logic := '0';
signal SPI_DIN, SPI_nCS, SPI_CLK : std_logic := '0';

begin

primary_clock: clk_250 <= not clk_250 after CLK_PERIOD / 2;
 DUT: entity work.soru_5
    port map(
        clk_250 => clk_250,
        sampling_pulse => sampling_pulse,
        postscaler_in => SCK_POSTSCALER,
        spi_sck => SPI_CLK,
        spi_cs_n => SPI_nCS,
        spi_din => SPI_DIN,
        data_out => open);

DATA_SAMPLE: process
    begin
        wait for CLK_PERIOD*100;

        rawdata <= to_unsigned(12345,14);
        sampling_pulse <= '1';
        wait for CLK_PERIOD;
        sampling_pulse <= '0';

        wait for CLK_PERIOD*100;

        rawdata <= to_unsigned(5782,14);
```

```
        sampling_pulse <= '1';
        wait for CLK_PERIOD;
        sampling_pulse <= '0';

        wait for CLK_PERIOD*100;

        rawdata <= to_unsigned(777,14);
        sampling_pulse <= '1';
        wait for CLK_PERIOD;
        sampling_pulse <= '0';

    end process DATA_SAMPLE;

    SPI_TARGET: process(SPI_nCS,SPI_CLK,SPI_DIN)
    variable counter : integer := 0;
    begin
        if SPI_nCS='1' then
            SPI_DIN <= 'Z';
            counter := 13 + NBLANKBITS;
        elsif SPI_nCS='0' and falling_edge(SPI_CLK) then
            if (counter > 13 or counter < 0) then
                SPI_DIN <= '0';
            else
                SPI_DIN <= std_logic(rawdata(counter));
            end if;
            counter := counter - 1;
        end if;
    end process SPI_TARGET
end Behavioral;
```

Simulasyon Çıktısı:

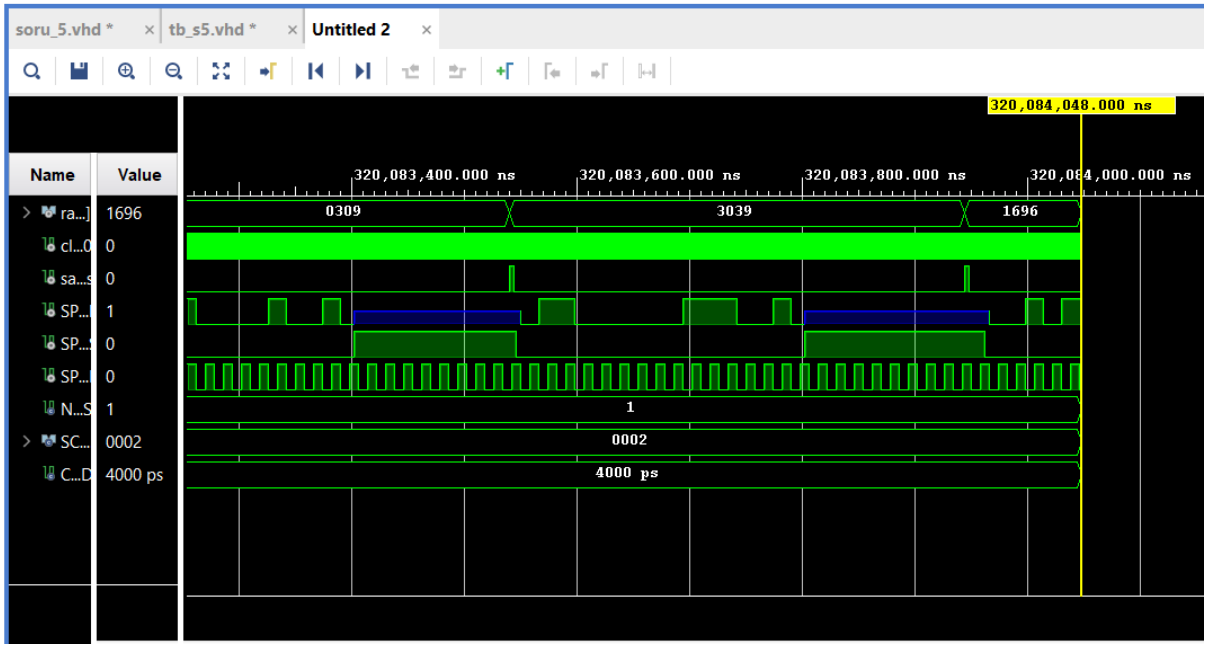

Figure 17 Simulasyon Çıktısı