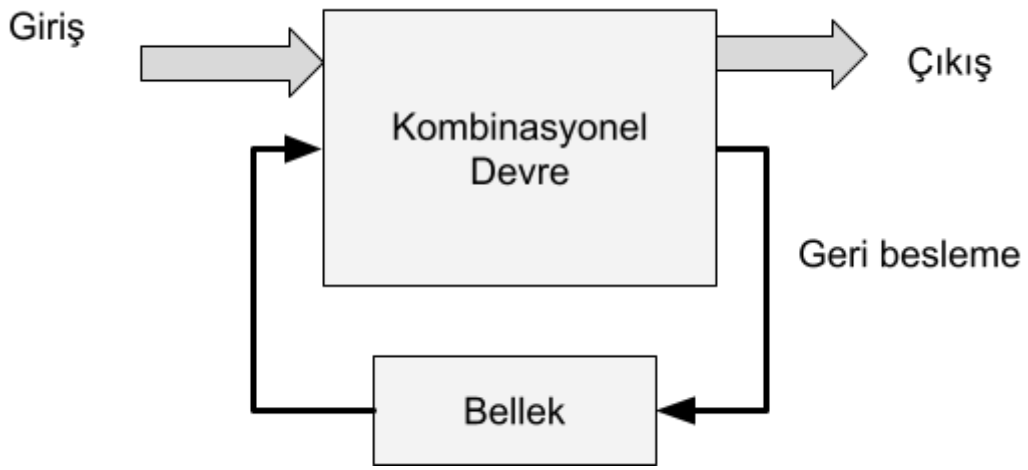
# Karadeniz Teknik Üniversitesi/Bilgisayar Mühendisliği Bölümü

# Sayısal Tasarım Laboratuvarı

## VHDL ile Ardışıl Devre Tasarımı

Sayısal Tasarım uygulamalarında tasarlanan entegre devreler kombinasyonel (tümleşik) ve ardışıl olmak üzere iki türe ayrılır. Bu deneyde temel olarak ardışıl devrelerin VHDL dili ile tasarımı ve simulasyonu ele alınacaktır.

Kombinasyonel devrelerin çıkışları girişlerin kombinasyonuna bağlı olarak değişmektedir. Ardışıl devrelerde ise kombinasyonel devrelerinin aksine çıkış durumu, "mevcut girdi", "geçmiş girdi" ve/veya "geçmiş çıktı" olmak üzere aşağıdaki üç durumun bir fonksiyonudur. Ardışıl devreler sahip olduğu bellek birimi ile geçmiş girdi ya da çıktıları depolayarak bir sonraki çıkışın hesaplanması için kullanmaktadır. Şekil 1'de ardışıl devrelerin blok şeması verilmektedir.




Şekil 1: Ardışıl devrelerin blok şeması

Ardışıl devrelerde "ardışıl" kelimesi saat sinyali ile bir sonraki girdilerin sıralı olarak devreye uygulanması anlamına gelmektedir. Ardışıl devrelere örnek olarak Flip-Flop'lar, Sayıcılar, tampon bellek birimleri verilebilir.

### Durum Makineleri

Trafik Lambası için geliştirilen durum makinesinin VHDL kodu aşağıda verilmiştir.

```
Type tDurumlar is (KRMZ, SAR, YSL); Signal
dSimdi, dSonra: tDurumlar := KRMZ; Process(s, r) - -
Beklenen sinyaller
Begin
    CASE dSimdi IS - - Durum makinesi When
        KRMZ =>
            dSonra <= SAR; When
        SAR =>
```

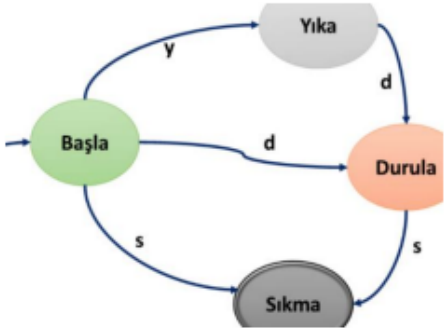
```
                    dSonra <= YSL; When
            KYBT =>
                    dSonra <= KRMZ;
        End case; End
Process;

Process(s) - - Durumu guncellemek icin Process Begin --
durumu guncelleme
  if s' event and s = '1' then dSimdi
     <= dSonra;
  end if;
End Procecss;
```

## Basit Bir Çamaşır Makinesi Tasarımı

Karmaşık sıralı devreler bir kaç evreden oluşabilir. Her bir evreyi SDM'de bir durumla ifade edebiliriz. Bu tip ardışıl devreler, içinde bulunduğu durum bilgisini kullanarak bir sonraki durumda ne yapılacağını bilir. Bir sıralı devre tasarlanırken, öncelikle, tasarımcı tarafından devrenin durumlarının belirlenmesi gerekir.

Durum makineleri, akış diyağramlarına benzer bir şema/graf ile gösterilir. Şekilde basit bir çamaşır makinesi için geliştireceğimiz durum makinesinin şeması verilmiştir. Grafa baktığımızda bu makinede dört durum bulunmaktadır ve durumlar arası geçişleri gösteren beş ok vardır.

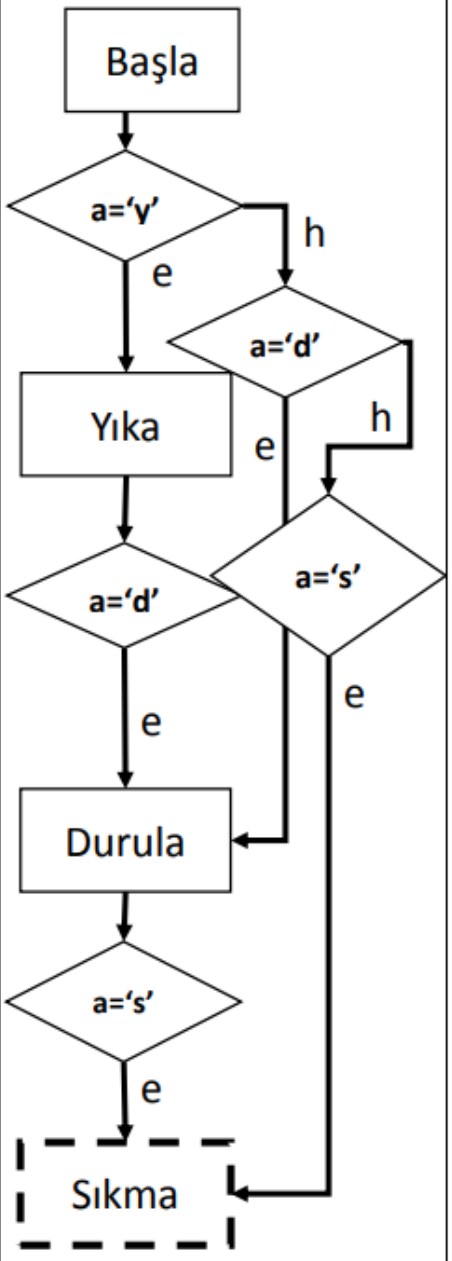



Şekil incelendiğinde ilk durum olan 'Başla', boşluktan gelen bir ok ile gösterilmiştir. 'Sıkma' durumu makinenin son evresidir. Geçişler üzerindeki harflerle verilen etiketler makinenin alfabesinde olan karekterlerdir ve bir durumdan diğerine geçmek için okunması gereken sembollerdir.

Durumları belirlenen SDM'nin durumlar arası geçiş tablosunun oluşturulması devrenin tasarım süresini kısaltan bir yöntemdir. Yanda verilen tablo, yukarıda grafı verilen makinenin durum geçişlerini özetleyen tablodur. Örneğin makine 'başla' durumunda ise ve giriş portundan 'y' sembolü okunmuşsa makinenin yeni durumu 'Yıka' olacaktır.

| Durum | Okunan Değer | Yeni Durum |
|---|---|---|
| Başla | y | Yıka |
| Başla | d | Durula |
| Başla | s | Sıkma |
| Yıka | d | Durula |
| Durula | s | Sıkma |

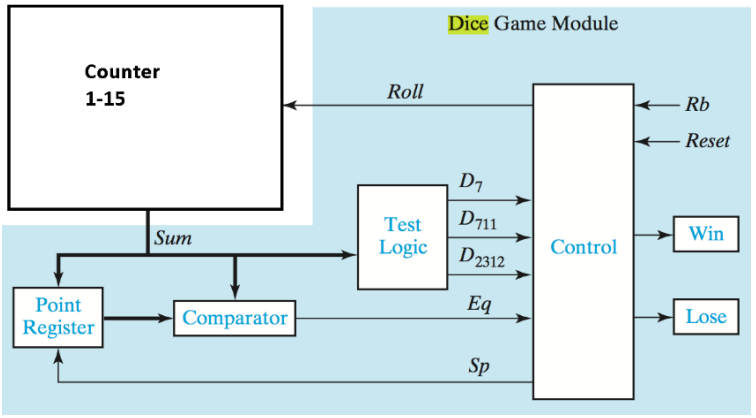
Yan tarafta VHDL kodu verilen durum makinesinde, iki bitlik A girişinden semboller kodlanmış olarak okunuyor. Yıkama komutu için y= '00', durulama için d = '01', ve son olarak sıkma komutu için s='10' kodları belirlenmiştir. Makinenin durumuna göre ve A portundaki bilgiye göre makine durumunu güncellemektedir. Devrenin portunda tanımlanan f çıkış portu makinenin işlemini bitirip bitirmediğini gösteren değerdir. Makinenin çalışır durumda olduğu 'başla', 'yıka', ve 'durula' evrelerinde f portuna '0' yazılmakta, 'sıkma' durumuna geçildiğinde ise çıkışa '1' aktarılmaktadır.




```
-------------------------------------------
-- Basit Camasir makinesi icin
-- Sonlu Durum Makinesi (SDM)
-- 4 Durumlu: BSL, YK, DRL, SKM
-- okunan semboller: y='00', d='01', s='10'.
-------------------------------------------
Library IEEE;
Use IEEE.std_logic_1164.all;
Entity eSDM is
Port( s : in std_logic;  -- saat
      A : in std_logic_vector(1 downto 0);
      f : out std_logic );
End eSDM;

Architecture Behv of eSDM is
-- Durumlar icin tip tanimi.
Type tDurumlar is ( BSL, YK, DRL, SKM );
Signal dSimdi : tDurumlar := Bsl ; -- ilk durum
Begin
Process(s)  -- Durum makinesi
Begin
  If( Rising_edge(s) ) then
    Case dSimdi is
      When BSL =>   -- y girisi
        If( A="00" ) Then  -- bitmedi
          f <= '0';  dSimdi <= YK;
        Elsif( A="01" ) Then -- d
          f <= '0'; dSimdi <= Drl;
        Elsif( A="10" ) Then -- s
          f <= '0'; dSimdi <= SKM;
        End if;
      When YK =>
        If( A="01" ) Then    -- d
          f <= '0'; dSimdi <= DRL;
        End if;
      When DRL =>
        If( A="10" ) Then    -- s
          f <= '0'; dSimdi <= SKM;
        End if;
      When SKM => f <= '1';   -- BITTI
      End Case;
   End if; -- her saat sinyalinde
End process;
End Behv;
```

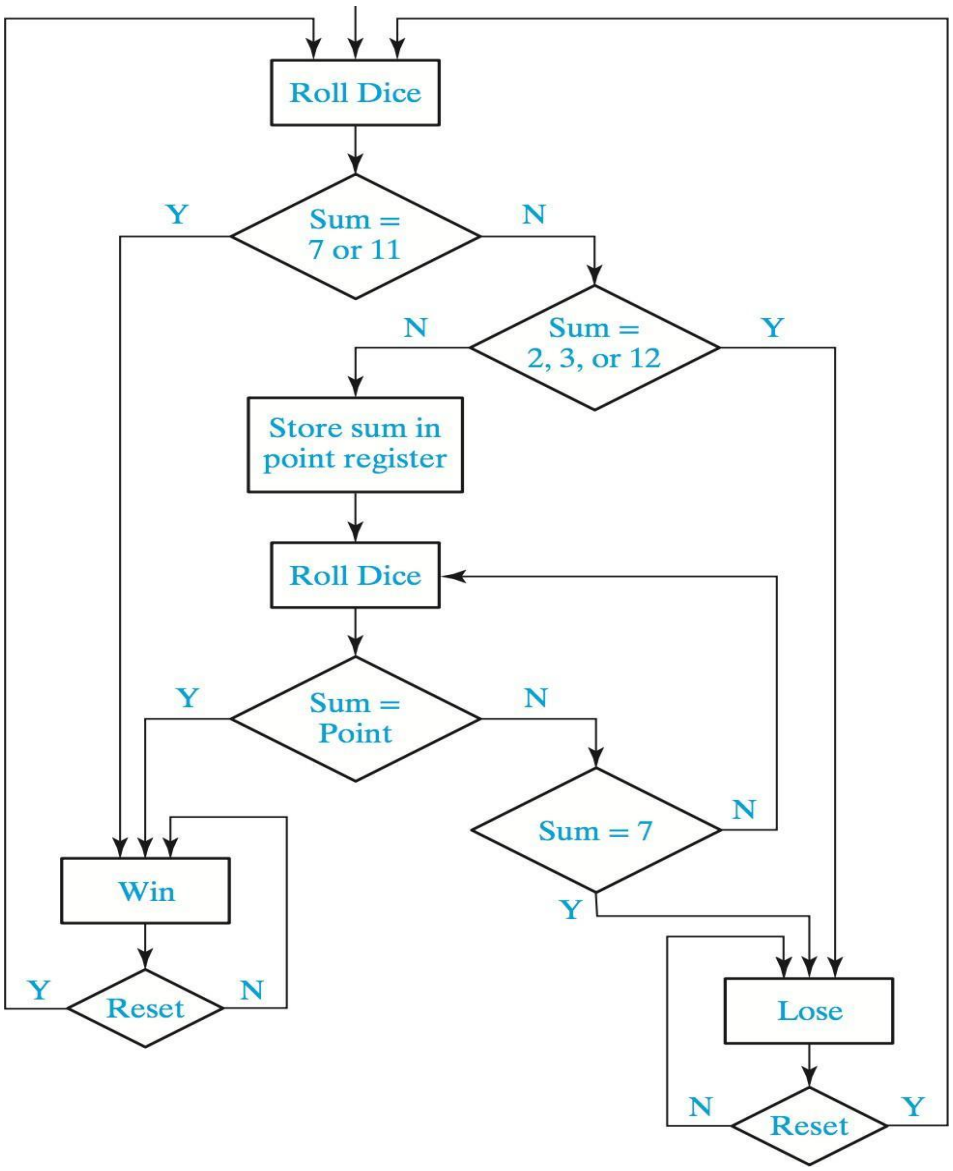
# Tahmin Oyunu




***Şekil 3**- Tahmin oyununun devre şeması.*

Tahmin oyununun blok şeması yukarıda verilmiştir. Tahmin oyununda 3 giriş (reset, Rb- say butonu-, ve saat) ve 3 çıkış bulunmaktadır. İlk durumda Kyt saklayıcısında herhangi bir değer yoktur. Reset butonuna basılarak istenildiği zaman devre ilk durumuna getirilir. Say butonu ile sayıcı aktif hale gelir ve sistem frekansına bağlı olarak sayma devam eder. Say butonundan el çekildiğinde Oyun için hazırlanan durum makinesi, içinde bulunduğu duruma göre gerekli düzenlemeleri yapmaktadır. Tahmin oyunu için hazırlanan VHDL kodu aşağıda verilmiştir. Durum makinesinin içindeki bazı kısımlar öğrenciler tarafından doldurulacaktır.

1- Tahmin oyununda say butonuna (Rb) basılınca sayıcı hızla saymaya başlar.
2- Buton bırakılınca, ilk denemede eğer sayıcı değeri 7 veya 11 ise oyuncu tahmin oyununu kazanır, yok eğer sonuç 2, 3, veya 12 ise oyunu kaybeder. Eğer elde edilen ilk sonuç bu değerlerden birisi değilse bu değer saklanır.
3- Oyun bitinceye kadar sayma butonuna tekrar tekrar basılarak saklanan ilk değer tahmin edilmeye çalışılır. Eğer sayıcı değeri herhangi bir durumda 7 olursa oyuncu oyunu kaybeder.




***Şekil 4****- ModelSim uygulamasında Veya kapısının simülasyonu.*

Tahmin oyununun durum makinesi yukarıda verilmiştir. Verilen bilgiler ışığında sizden tahmin oyunu için verilen VHDL kodunun boş bırakılan yerlerini doldurmanızdır.

---Tahmin Oyununun VHDL Kodu

**Library IEEE;**
**Use IEEE.STD_LOGIC_1164. all; Use**
**IEEE.STD_LOGIC_ARITH. all;**
**Use IEEE.STD_LOGIC_UNSIGNED. all; Use**
**IEEE.NUMERIC_STD.ALL;**

**Entity eOyun is**
**port (btnsay, r, s: in bit; kazan,**

```
        kaybet: out bit;
        F: out std_logic_vector(3 downto 0) ); end
  eOyun;
  Architecture Oyun of eOyun is signal
    say: bit;
   signal Tplm : std_logic_vector(3 downto 0):="0000"; - - sayici signal
   Kyt              : std_logic_vector(3 downto 0):="0000";

   TYPE tDurumlar is (BSL, KZN, KYBT, DVM);
   Signal dSimdi, dSonra: tDurumlar := BSL;

  Begin
     F <= Tplm;

    Process(btnsay, r, Tplm, dSimdi) - - Beklenen sinyaller Begin
      sakla, say, kazan ve kaybet sinyallarini sıfırla
      CASE dSimdi IS  - -Durum makinesi
        When BSL =>
        Bu durumda eğer btnsay butonuna basılmışsa saymaya devam edilir. Yok eğer
        tplm degeri 7 veya 11 ise KZN durumuna geç
        Yok eğer toplam değer 2,3 veya 12 ise KYBT durumuna geç Değilse
        sakla sinyalini set et ki ilk değerin kaydetmesini sağla ve DVM
        durumuna geç.
        When KZN =>
         Kazandınız: kazan çıkışını set et
        Eğer reset tuşuna basılmışsa BSL durumuna geç
       When KYBT =>
        Kaybettiniz: kaybet çıkışını 1 yap.
        Eğer reset tuşuna basılmışsa BSL durumuna geç
        When DVM =>
        Bu durumda eğer btnsay butonuna basılmışsa saymaya devam edilir. Yok eğer
          Tplm = kayit ise kazandınız durumuna geç
          Yok eğer Tplm = 7 ise kaybettiniz durumuna geç
End Case; End Process;

 - - saat sinyalinin yukselen kenarinda onceki bilgiyi sakla ve durumu degistir.

 Process(s)
 Begin -- Kyt' e sakla
        If s' event and s = '1' then
                dSimdi <= dSonra; - - Durumu guncelle If sakla
```

```
= '1' then - - ilk deger saklanir
        Kyt <= Tplm;
    End if;
    If btnsay = '1' then
        If Tplm = “1111” then Tplm <=
            ‘1’;
        Else
            Tplm <= Tplm + ‘1’;
        End if; End if;
End if; End
process;
End Oyun; - - Oyun sonu
```

## Sayıcı Tasarımı

Yukarıda VEYA kapısı için hazırlanan devrede kaydedici ünite (register) bulunmadığı için bir tümleşik devredir. Sayıcı gibi bilgi saklayıp daha sonra kullanan devreler için bir bellek birimi gerekmektedir. Bu uygulamada VHDL donanım tanımlama dilini kullanarak bir sayıcı tasarımını öğreneceğiz. Sayıcı devresi için aşağıda verilen adımları gerçekleştiriniz.

1. ModelSimde yeni bir proje oluşturunuz ve projeye uygun bir isim veriniz (örneğin Sayici gibi).
2. Kütüphane ve paketleri belirtiniz.
3. Uygun bir Entity adı (eSayici gibi) verdikten sonra port tanımını yapınız
    1. s: saat sinyali, (1bit giriş)
    2. r: reset sinyali, (1bit giriş)
    3. F: sayıcı çıkışı (4bit).

Sayıcı devresinin eleman tanımı aşağıdaki gibi olabilir.

```
Entity eSayici is -- devre elemani adi
Port( s, r: in std_logic; -- giris ve cikis portlari F: out
    std_logic_vector(3 downto 0) );
End eSayici;
```

Projenin mimari kısmını process içinde her saat sinyalinde sayıcıyı bir artacak şekilde oluşturunuz.

4. Projeyi derleyiniz, varsa hataları gideriniz ve devrenin simülasyonunu yapınız.

# Deney Uygulama Adımları

1. Yeni bir proje oluşturunuz.
2. Kodunuzu derleyiniz.
    1. Derlemede karşılaştığınız hataları açıklayınız.
3. Xilinx ve Altera (Quartus) uygulamaları kullanılan FPGA’in teknolojisine bağlı olarak hazırlanan projenin genel özelliklerini belirtmektedir. Bu bilgilerden yola çıkarak aşağıdaki sorulara cevap veriniz.
    1. Devrenizde kaç mantık elemanı kullanılmaktadır?
    2. Devrenizin maksimum çalışma frekansı (Fmax) nedir?

3. Kodunuzun 4-bitlik versiyonunu derleyip RTL Viewer yardımıyla devrenin şemasını elde ediniz.

4. Devrenin simulasyonunu yaparak devrenizin doğru çalıştığından emin olunuz.
5. Devrenizi FPGA kartına yükleyiniz ve çalışmasını test ediniz.

## Alıştırma projeler

Deney tamamlandığında deney grubundaki öğrenciler aşağıdaki örnek alıştırmalardan bir tanesini yaparak deney raporuna eklemelidir.

1. Standart VE Kapısı
2. 3 girişli VEYA kapısı
3. Toplama / çıkarma devresi
4. Çoklayıcı / tekleyici
5. Kodlama / çözümleme
6. Flip-flop
7. Mantıksal işlemler
8. Sağa / sola kaydırıcı
9. Aritmetik saga / sola kaydırıcı
10. Çevirme devresi