9. Benzetim Uygulamaları

Bu bölümde şu ana kadar anlatılan konuların kullanıldığı 15 örneğe yer verilmiştir. VHDL dili ile ilgil olarak anlatılan konuların pekiştirilmesi adına bu uygulamalar oldukça faydalı olacaktır. Burada anlatılan örnek uygulamalar sırasıyla aşağıda gösterildiği gibidir:

1. Yetki Girişli D-Mandalı
2. Yükselen Kenar Tetiklemeli D İki Durumlusu (D Flip-Flop)
3. Asenkron Resetli D İki Durumlusu (Asenkron Resetli D Flip-Flop)
4. Senkron Resetli D İki Durumlusu (Senkron Resetli D Flip-Flop)
5. Saklayıcı (Register)
6. Kaydırmalı Saklayıcı (Shift Register)
7. Sayaçlar
8. Saat (Clock) Frekans Bölme
9. VHDL'de metin dosyasından veri okuma
10. VHDL'de ROM Bloğu Oluşturmak
11. VHDL'de RAM Bloğu Oluşturmak
12. FIFO Tasarımı
13. Sinyal İşlemede Konvolüsyon
14. Temel İmge İşleme Algoritmaları
15. VHDL ile İmge'de Konvolisyon İşlemi

Bu bölümde verilen örnek uygulamaların tamamı benzetim ortamında gerçekleştirilmiş ve ilgili uygulamaya ait benzetim ekran görüntüleri her örnek için anlatımlar eklenmiştir. Ayrıca benzetim için kullanılan sınama kodu (test bench) ve tasarım kodu da her örnek için verilmiştir. Uygulamalar gerçekleştirilirken basit tasarımlardan, karmaşık tasarımlara doğru bir yol izlenmiştir.

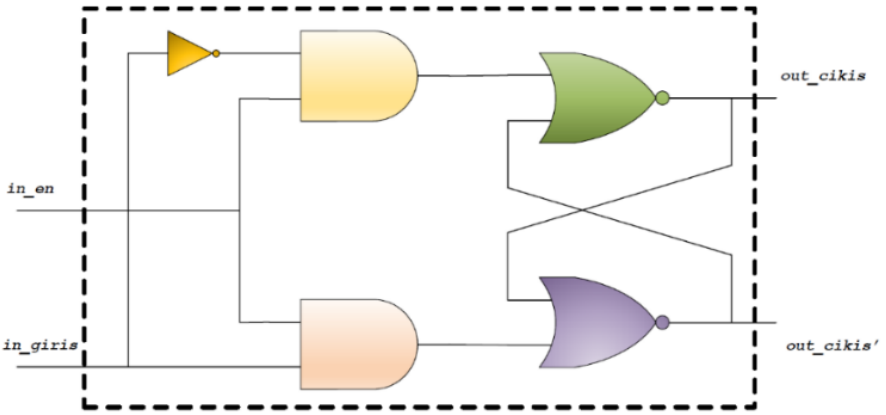
9.1. Yetki Girişli D Mandalı (D Latch)

D Mandalı devresi giriş sinyalinin durumu değiştiği zaman çıkış değerini güncelleyen, değişim olmadığı durumlarda ise çıkış değerini koruyan bir devre elemanıdır. Bu elemana bir yetkilendirme girişi eklendiğinde oluşan elemana "Yetki Girişli D Mandalı" adı verilmektedir. Bu elemanda çıkış değerinin güncellenmesi normal D Mandalından farklı olarak yetki girişinin de **'1'** olması gerekmektedir. Eğer yetki girişi **'0'** ise giriş değeri değişse bile çıkış değeri son durumunu korumaya devam edecektir.

Yetki girişli D mandalı doğruluk tablosu Tablo 9-1'de verilmiştir. Tablo 9-1'den de görüleceği üzere **in_en** giriş portu değeri **'0'** olduğunda çıkış değerleri **in_giris** giriş portu değerinin önemi olmadan bir önceki değerini korumaktadır. **in_en** girişi portu değeri **'1'** olduğunda ise **in_giris** giriş portu değeri **out_cikis** çıkış portuna ve **in_giris** giriş portu değerinin değili ise **out_cikis'** çıkış portuna aktarılmaktadır. Şekil 9-1'de ise yetki girişli D mandalı mantık devresi gösterilmiştir.

Tablo 9-1 Yetki girişli D mandalı doğruluk tablosu

in_en	in_giris	out_cikis(+)	out_cikis'(+)
0	0	out_cikis	out_cikis'
0	1	out_cikis	out_cikis'
1	0	0	1
1	1	1	0



Şekil 9-1 D mandalı mantık devresi

Örnek 9.1: Bu örnekte Yetki Girişli D Mandalı mantık devresinin gerçekleştirildiği **d_latch.vhd** VHDL kodu verilmiştir. Tablo 9-1 ve Şekil 9-1'den de görülceği üzere **d_latch** varlığımız iki giriş ve iki çıkış portuna sahiptir. Port tanımlamaya ilişkin bildirimler 5-10. satırları arasında yapılmatıdır. Port tanımalamaları **std_logic** veri tipinde yapıldığından dolayı 1. ve 2. satırlarda gerekli kütüphane bildirimleri yapılmaktadır.

15. satırda **std_logic** veri tipinde **r_cikis** sinyali tanımlanmaktadır. 19. satırda tanımlanan söz dizimi ile **process**'in **in_en** ve **in_giris** giriş portları değerlerinde meydana gelen değişiklikler ile aktif olacağı belirtilmektedir. 22. satırda tanımlı koşul ifadesinin sağlanması yani **in_en** giriş portu değerinin **'1'** olması durumunda **in_giris** giriş portu değeri **r_cikis** sinyaline atanmaktadır. Aksi durumlarda ise **r_cikis** sinyali bir önceki değerini korumaktadır. 28. satırda **out_cikis** çıkış portuna **r_cikis** sinyalinin değeri atanırken, 29. satırda ise **out_cikis_degil** çıkış portuna **r_cikis** sinyalinin değerinin değili atanmaktadır.

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.all;
3.
4. entity d_latch is
5.   port (
6.     in_en : in std_logic;
7.     in_giris : in std_logic;
8.     out_cikis : out std_logic;
9.     out_cikis_degil : out std_logic
10.  );
11.end d_latch;
12.
13.architecture Behavioral of d_latch is
14.
15.  signal r_cikis : std_logic := '0';
16.
17.begin
18.
19.  process(in_en, in_giris)
```

```
20.  begin
21.
22.    if in_en = '1' then
23.      r_cikis <= in_giris;
24.    end if;
25.
26.  end process;
27.
28.  out_cikis <= r_cikis;
29.  out_cikis_degil <= (not r_cikis);
30.
31.end Behavioral;
```

Aşağıda ise **d_latch** varlığının benzetim yapılabilmesi için **tb_d_latch.vhd** sınama kodu (test bench) verilmiştir ve sınama koduna ait benzetim çıktısı Şekil 9-2'de gösterilmiştir. Şekil 9-2'de gösterilen benzetim sonucunda:

1. adım: **in_en** giriş portu değeri **'1'** olduğundan **r_cikis** sinyaline **in_giris** giriş portu değeri atanmaktadır ve **r_cikis** sinyalinin değeri **'0'** olmaktadır. **r_cikis** sinyalinin değerinin **'0'** olması ile **out_cikis** çıkış portuna **'0'**, **out_cikis_degil** çıkış portuna **'1'** değerleri atanmaktadır.

2. adım: **in_en** giriş portu değerine **'0'** ve **in_giris** giriş portu değerine **'1'** atanmaktadır. **r_cikis** sinyalinin değeri **in_en** giriş portu değerinin **'0'** olması nedeniyle bir önceki adımdaki değerini korumuştur. **r_cikis** sinyalinin değerinin değişmemesi ile çıkış portlarının değerleride değişmemiştir.

3. adım: **in_en** giriş portu değerine **'1'** atanmaktadır. **in_en** girişi portu değeri **'1'** olmasıyla **r_cikis** sinyaline **in_giris** giriş portu değeri atanmıştır ve **r_cikis** sinyalinin değeri **'1'** olmaktadır. **r_cikis** sinyalinin değerinin **'1'** olması ile **out_cikis** çıkış portuna **'1'**, **out_cikis_degil** çıkış portuna **'0'** değerleri atanmaktadır.

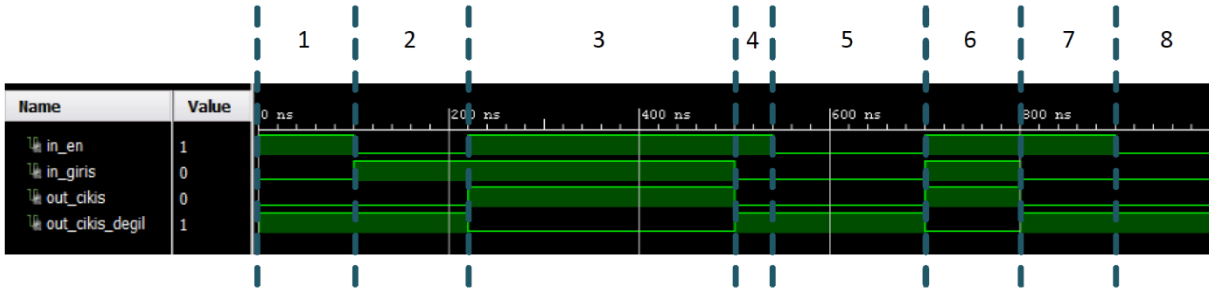
4. adım: **in_en** giriş portu değeri **'1'** olduğundan **r_cikis** sinyaline **in_giris** giriş portu değeri atanmıştır ve **r_cikis** sinyalinin değeri **'0'** olmaktadır. **r_cikis** sinyalinin değerinin **'0'** olması ile **out_cikis** çıkış portuna **'0'**, **out_cikis_degil** çıkış portuna **'1'** değerleri atanmaktadır.

5. adım: **in_en** giriş portu değeri **'0'** olmaktadır. **r_cikis** sinyalinin değeri **in_en** giriş portu değerinin **'0'** olması nedeniyle bir önceki adımdaki değerini korumuştur. **r_cikis** sinyalinin değerinin değişmemesi ile çıkış portlarının değeride değişmemiştir.

6. adım: **in_en** giriş portu değeri **'1'** olduğundan **r_cikis** sinyaline **in_giris** giriş portu değeri atanmıştır ve **r_cikis** sinyalinin değeri **'1'** olmaktadır. **r_cikis** sinyalinin değerinin **'1'** olması ile **out_cikis** çıkış portuna **'1'**, **out_cikis_degil** çıkış portuna **'0'** değerleri atanmaktadır.

7. adım: **in_en** giriş portu değeri **'1'** olduğundan **r_cikis** sinyaline **in_giris** giriş portu değeri atanmıştır ve **r_cikis** sinyalinin değeri **'0'** olmaktadır. **r_cikis** sinyalinin değerinin **'0'** olması ile **out_cikis** çıkış portuna **'0'**, **out_cikis_degil** çıkış portuna **'1'** değerleri atanmaktadır.

8. adım: **in_en** giriş portu değeri **'0'** olmaktadır. **r_cikis** sinyalinin değeri **in_en** giriş portu değerinin **'0'** olması nedeniyle bir önceki adımdaki değerini korumuştur. **r_cikis** sinyalinin değerinin değişmemesi ile çıkış portlarının değeride değişmemiştir.



Şekil 9-2 d_latch varlığı benzetim çıktısı

tb_d_latch.vhd VHDL kodun amacı **d_latch** varlığının benzetiminin yapmak olduğundan dolayı **tb_d_latch** varlığına ait bir port tanımlaması yapılmamıştır. **tb_d_latch.vhd** VHDL kodu sentezlenemez ve sadece benzetim için kullanılabilmektedir. Test kodlarının oluşturulması amacı ile kullanılacak olan sinyallerin tanımlama işlemleri 18-21. satırlarda yapılmıştır.

25-33. satırlar arasında tanımlanan **process** ile **in_en** yetkilendirme sinyalinin davranışı tanımlanmaktadır. **wait for** komutu ile tanımlanan süre kadar **process** içinde beklenmektedir.

- 27. satırda yapılan tanımlama ile **in_en** yetkilendirme sinyaline **'1'** değeri atanmaktadır ve **100ns** beklendikten sonra 28. satırda tanımlanan ifade gerçeklenecektir. Diğer bir ifade ile 27. satırda yapılan tanımlama ile **in_en** yetkilendirme sinyalinin **0-100 ns** aralığında alacağı değer tanımlanmıştır.
- 28. satırda tanımlanan ifade ile **in_en** yetkilendirme sinyalinin **100-220 ns** aralığında alacağı değerin **'0'** olacağı belirtilmektedir.
- 29. satırda tanımlanan ifade ile **in_en** yetkilendirme sinyalinin **220-540 ns** aralığında alacağı değerin **'1'** olacağı belirtilmektedir.
- 30. satırda tanımlanan ifade ile **in_en** yetkilendirme sinyalinin **540-700 ns** aralığında alacağı değerin **'0'** olacağı belirtilmektedir.
- satırda tanımlanan ifade ile **in_en** yetkilendirme sinyalinin **700-900 ns** aralığında alacağı değerin **'1'** olacağı belirtilmektedir.
- satırda tanımlanan ifade ile **in_en** yetkilendirme sinyalinin **900-1000 ns** aralığında alacağı değerin **'0'** olacağı belirtilmektedir.

35-42. satırları arasında tanımlanan **process** içerisinde **in_giris** sinyalinin davranışı tanımlanmaktadır.

- 37. satırda tanımlanan ifade ile **in_giris** sinyalinin **0-100 ns** aralığında alacağı değerin **'0'** olacağı belirtilmektedir.
- 38. satırda tanımlanan ifade ile **in_giris** sinyalinin **100-500 ns** aralığında alacağı değerin **'1'** olacağı belirtilmektedir.
- 39. satırda tanımlanan ifade ile **in_giris** sinyalinin **500-700 ns** aralığında alacağı değerin **'0'** olacağı belirtilmektedir.
- 40. satırda tanımlanan ifade ile **in_giris** sinyalinin **700-800 ns** aralığında alacağı değerin **'1'** olacağı belirtilmektedir.
- 41. satırda tanımlanan ifade ile **in_giris** sinyalinin **800-1000 ns** aralığında alacağı değerin **'0'** olacağı belirtilmektedir.

d_latch varlığının **tb_d_latch** varlığında alt devre olarak kullanılabilmesi için gerekli **component** tanımlama işlemleri 9-16. satırlar arasında yapılmıştır. 44-50. satırlar arasında ise **d_latch** alt devresine ilişkin bağlantılar yapılmaktadır.

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.all;
3.
4. entity tb_d_latch is
5. end tb_d_latch;
```

```
6.
7. architecture Behavioral of tb_d_latch is
8.
9.   component d_latch
10.  Port (
11.    in_en : in std_logic;
12.    in_giris : in std_logic;
13.    out_cikis : out std_logic;
14.    out_cikis_degil : out std_logic
15.  );
16.  end component;
17.
18.  signal in_en : std_logic := '0';
19.  signal in_giris : std_logic := '0';
20.  signal out_cikis : std_logic := '0';
21.  signal out_cikis_degil : std_logic := '0';
22.
23.begin
24.
25.  process
26.  begin
27.    in_en <= '1'; wait for 100 ns;
28.    in_en <= '0'; wait for 120 ns;
29.    in_en <= '1'; wait for 320 ns;
30.    in_en <= '0'; wait for 160 ns;
31.    in_en <= '1'; wait for 200 ns;
32.    in_en <= '0'; wait for 100 ns;
33.  end process;
34.
35.  process
36.  begin
37.    in_giris <= '0'; wait for 100 ns;
38.    in_giris <= '1'; wait for 400 ns;
39.    in_giris <= '0'; wait for 200 ns;
40.    in_giris <= '1'; wait for 100 ns;
41.    in_giris <= '0'; wait for 200 ns;
42.  end process;
43.
44.  d_latch_map : d_latch
45.  port map(
```

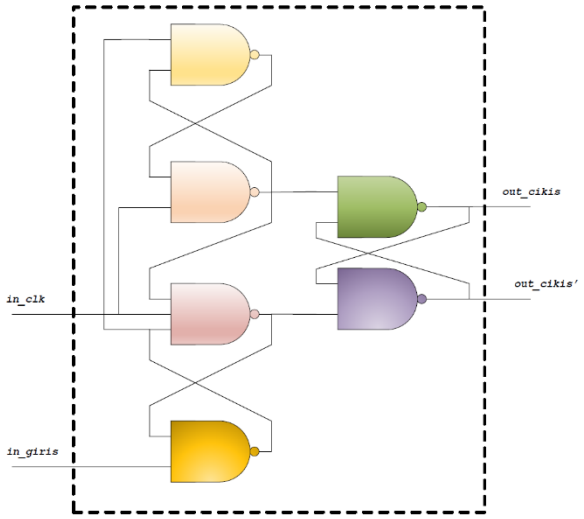
```
46.     in_en => in_en,
47.     in_giris => in_giris,
48.     out_cikis => out_cikis,
49.     out_cikis_degil => out_cikis_degil
50.   );
51.
52.end Behavioral;
```

9.2. Yükselen Kenar Tetiklemeli D İki Durumlusu (D Flip Flop)

Yükselen kenar tetiklemeli D iki durumlusunda, saat darbesinin yükselen kenarına bağlı olarak çıkış değeri güncelleme işlemi yapılmaktadır. Yükselen kenar tetiklemeli D iki durumlusuna ait doğruluk tablosu Tablo 9-2'de verilmiştir. Tablo 9-2'dende görüleceği üzere saat darbesin yükselen kenarında **`in_giris`** giriş portunun değeri **`out_cikis`** çıkış portuna ve **`in_giris`** giriş portunun değerinin değili ise **`out_cikis'`** çıkış portuna atanmaktadır. Saat darbesinin yükselen kenarı mevcut olmadığı durumlarda ise giriş değeri ne olursa olsun çıkış değerleri eski değerlerini korumaktadır. Şekil 9-3'de ise Yükselen kenar tetiklemeli D iki durumlusuna ait mantık devresi verilmiştir.

Tablo 9-2 Yükselen kenar tetiklemeli D iki durumlusu doğruluk tablosu

Saat Darbesi	in_giris	out_cikis(+)	out_cikis'(+)
Çıkan kenar	0	0	1
Çıkan kenar	1	1	0
Çıkan kenar yok	X	out_cikis	out_cikis'



Şekil 9-3 Yükselen kenar tetiklemeli D iki durumlusu mantık devresi

Örnek 9.2: Aşağıda yükselen kenar tetiklemeli D iki durumlu mantık devresinin gerçekleştirildiği **`d_mandali.vhd`** VHDL kodu verilmiştir. 1. ve 2. satırlarda tasarımda kullanılacak olan kütüphane bildirimleri

yapılmaktadır. Tablo 9-2 ve Şekil 9-3'den de görülceği üzere **d_mandali** varlığımız iki giriş ve iki çıkış portuna sahiptir. Port tanımlamaya ilişkin bildirimler 5-10 satırları arasında yapılmaktadır. Port tanımalamaları **std_logic** veri tipinde yapıldığından dolayı 1. ve 2. satırlarda gerekli kütüphane bildirimleri yapılmaktadır.

15. satırda **std_logic** veri tipinde **r_cikis** sinyali tanımlanmaktadır. 19. satırda tanımlanan söz dizimi ile **process**'in **in_clk** ve **in_giris** giriş port değerlerinde meydana gelen değişiklikler ile aktif olacağı belirtilmektedir. 22. satırda tanımlı koşul ifadesinin sağlanması yani **in_clk** giriş portunda yükselen kenarının meydana gelmesi durumunda **in_giris** giriş port değeri **r_cikis** sinyaline atanmaktadır. Aksi durumlarda ise **r_cikis** sinyali bir önceki değerini korumaktadır. 28. satırda **out_cikis** çıkış portuna **r_cikis** sinyalinin değeri atanırken, 29. satırda ise **out_cikis_degil** çıkış portuna **r_cikis** sinyalinin değerinin değili atanmaktadır.

Eğer tasarım düşen kenar olarak tasarlanmak istenirse 22. satırdaki `in_clk = '1'` koşulu `in_clk = '0'` şeklinde değiştirilerek kullanılabilir.

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.all;
3.
4. entity d_mandali is
5.   port (
6.     in_clk : in std_logic;
7.     in_giris : in std_logic;
8.     out_cikis : out std_logic;
9.     out_cikis_degil : out std_logic
10.  );
11.end d_mandali;
12.
13.architecture Behavioral of d_mandali is
14.
15.  signal r_cikis : std_logic := '0';
16.
17.begin
18.
19.  process(in_clk, in_giris)
20.  begin
21.
22.    if in_clk'event and in_clk = '1' then
23.      r_cikis <= in_giris;
24.    end if;
25.
26.  end process;
27.
28.  out_cikis <= r_cikis;
29.  out_cikis_degil <= (not r_cikis);
30.
```

```
31.end Behavioral;
```

Aşağıda ise **d_mandali** varlığının benzetim yapılabilmesi için **tb_d_mandali.vhd** sınama kodu (test bench) verilmiştir ve benzetim çıktısı Şekil 9-4'de gösterilmiştir. Şekil 9-4'de gösterilen benzetim sonucunda:

1. adım: **in_clk** giriş portunda yükselen kenar meydana gelmesi ile birlikte **in_giris** giriş portu değeri **r_cikis** sinyaline atanmaktadır ve **r_cikis** sinyalinin değeri **'0'** olmaktadır. **r_cikis** sinyalinin değerinin **'0'** olması ile **out_cikis** çıkış portu değerine **'0'**, **out_cikis_degil** çıkış portu değerine ise **'1'** atanmaktadır. 1. adım içerisinde **in_giris** giriş portu **'1'** değerini almakta ve **process** aktif olmaktadır. Fakat 22. satırda tanımlanan **in_clk** giriş portunda saat darbesi yükselen kenarının meydana gelip gelmediğini kontrol eden koşul ifadesi gerçeklenmediğinden dolayı çıkış portlarının değerleri değişmemiştir.

2. adım: **in_clk** giriş portunda yükselen kenar meydana gelmesi ile birlikte **in_giris** giriş portu değeri **r_cikis** sinyaline atanmaktadır ve **r_cikis** sinyalinin değeri **'1'** olmaktadır. **r_cikis** sinyalinin değerinin **'1'** olması ile **out_cikis** çıkış portu değerine **'1'**, **out_cikis_degil** çıkış portu değerine ise **'0'** atanmaktadır

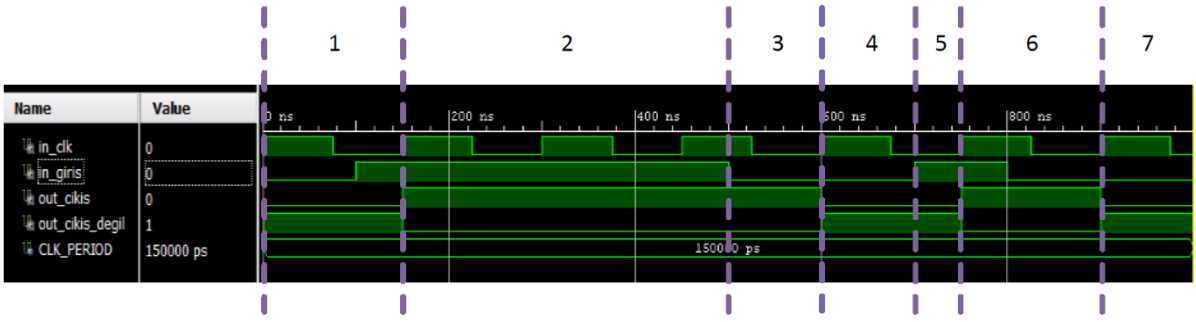
3. adım: **in_giris** giriş portu **'0'** değerini almakta ve **process** aktif olmaktadır. Fakat 22. satırda tanımlanan **in_clk** giriş portunda saat darbesi yükselen kenarının meydana gelip gelmediğini kontrol eden koşul ifadesi gerçeklenmediğinden dolayı çıkış portlarının değerleri değişmemiştir.

4. adım: **in_clk** giriş portunda yükselen kenar meydana gelmesi ile birlikte **in_giris** giriş portu değeri **r_cikis** sinyaline atanmaktadır ve **r_cikis** sinyalinin değeri**'0'** olmaktadır**. r_cikis** sinyalinin değerinin **'0'** olması ile **out_cikis** çıkış portu değerine **'0'**, **out_cikis_degil** çıkış portu değerine ise **'1'** atanmaktadır

5. adım : **in_giris** giriş portu **'1'** değerini almakta ve **process** aktif olmaktadır. Fakat 22. satırda tanımlanan **in_clk** giriş portunda saat darbesi yükselen kenarının meydana gelip gelmediğini kontrol eden koşul ifadesi gerçeklenmediğinden dolayı çıkış portlarının değerleri değişmemiştir.

6. adım : **in_clk** giriş portunda yükselen kenar meydana gelmesi ile birlikte **in_giris** giriş portu değeri **r_cikis** sinyaline atanmaktadır ve **r_cikis** sinyalinin değeri **'1'** olmaktadır. **r_cikis** sinyalinin değerinin **'1'** olması ile **out_cikis** çıkış portu değerine **'1'**, **out_cikis_degil** çıkış portu değerine ise **'0'** atanmaktadır. 6. adım içerisinde **in_giris** giriş portu **'0'** değerini almakta ve **process** aktif olmaktadır. Fakat 22. satırda tanımlanan **in_clk** giriş portunda saat darbesi yükselen kenarının meydana gelip gelmediğini kontrol eden koşul ifadesi gerçeklenmediğinden dolayı çıkış portlarının değerleri değişmemiştir.

7. adım : **in_clk** giriş portunda yükselen kenar meydana gelmesi ile birlikte **in_giris** giriş portu değeri **r_cikis** sinyaline atanmaktadır ve **r_cikis** sinyalinin değeri **'0'** olmaktadır. **r_cikis** sinyalinin değerinin **'0'** olması ile **out_cikis** çıkış portu değerine **'0'**, **out_cikis_degil** çıkış portu değerine ise **'1'** atanmaktadır.



Şekil 9-4 d_mandali varlığı benzetim çıktısı

tb_d_mandali.vhd VHDL kodun amacı **d_mandali** varlığının benzetiminin yapmak olduğundan dolayı **tb_d_mandali** varlığına ait bir port tanımlaması yapılmamıştır. **tb_d_mandali.vhd** VHDL kodu sentezlenemez sadece benzetim için kullanılabilmektedir. 18. satırda saat darbesinin periyodunun belirlendiği sabit

tanımlaması yapılmıştır. Tanımlamada saat darbesi periyodu **150 ns** olarak belirlenmiştir. Test kodlarının oluşturulması amacı ile kullanılacak olan sinyallerin tanımlama işlemleri 19-22. satırlarda yapılmıştır.

26-33. satırlar arasında tanımlanan **process** ile **in_clk** saat darbesi sinyalinin davranışı tanımlanmaktadır. 30. satırda **in_clk** sinyaline **'1'** değeri atanmakta ve 31. satırda tanımlanan ifade ile saat darbesi periyodunun yarısı kadar beklenmektedir. 32. satırda **in_clk** sinyaline **'0'** değeri atanmakta ve 33. satırda tanımlanan ifade ile saat darbesi periyodunun yarısı kadar beklenmektedir. Bu şekilde %50 doluluk oranında saat darbesi elde edilmiştir.

35-42. satırları arasında tanımlanan **process** içerisinde **in_giris** sinyalinin davranışı tanımlanmaktadır. 37. satırda tanımlanan ifade ile **in_giris** sinyalinin **0-100 ns** aralığında alacağı değerin **'0'** olacağı belirtilmektedir. 38. satırda tanımlanan ifade ile **in_giris** sinyalinin **100-500 ns** aralığında alacağı değerin **'1'** olacağı belirtilmektedir. 39. satırda tanımlanan ifade ile **in_giris** sinyalinin **500-700 ns** aralığında alacağı değerin **'0'** olacağı belirtilmektedir. 40. satırda tanımlanan ifade ile **in_giris** sinyalinin **700-800 ns** aralığında alacağı değerin **'1'** olacağı belirtilmektedir. 41. satırda tanımlanan ifade ile **in_giris** sinyalinin **800-1000 ns** aralığında alacağı değerin **'0'** olacağı belirtilmektedir.

d_mandali varlığının **tb_d_mandali** varlığında alt devre olarak kullanılabilmesi için gerekli **component** tanımlama işlemleri 9-16. satırlar arasında yapılmıştır. 44-50. satırlar arasında ise **d_mandali** alt devresine ilişkin bağlantılar yapılmaktadır.

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.all;
3.
4. entity tb_d_mandali is
5. end tb_d_mandali;
6.
7. architecture Behavioral of tb_d_mandali is
8.
9.   component d_mandali
10.  Port (
11.    in_clk : in std_logic;
12.    in_giris : in std_logic;
13.    out_cikis : out std_logic;
14.    out_cikis_degil : out std_logic
15.  );
16.  end component;
17.
18.  constant CLK_PERIOD : time := 150 ns;
19.  signal in_clk : std_logic := '0';
20.  signal in_giris : std_logic := '0';
21.  signal out_cikis : std_logic := '0';
22.  signal out_cikis_degil : std_logic := '0';
23.
24.begin
25.
26.  process
```

```
27.  begin
28.    in_clk <= '1';
29.    wait for CLK_PERIOD / 2;
30.
31.    in_clk <= '0';
32.    wait for CLK_PERIOD / 2;
33.  end process;
34.
35.  process
36.  begin
37.    in_giris <= '0'; wait for 100 ns;
38.    in_giris <= '1'; wait for 400 ns;
39.    in_giris <= '0'; wait for 200 ns;
40.    in_giris <= '1'; wait for 100 ns;
41.    in_giris <= '0'; wait for 200 ns;
42.  end process;
43.
44.  d_mandali_map : d_mandali
45.  Port map (
46.    in_clk => in_clk,
47.    in_giris => in_giris,
48.    out_cikis => out_cikis,
49.    out_cikis_degil => out_cikis_degil
50.  );
51.end Behavioral;
```

9.3. Eşzamanlı Olmayan Resetli Yükselen Kenar Tetiklemeli D İki Durumlusu

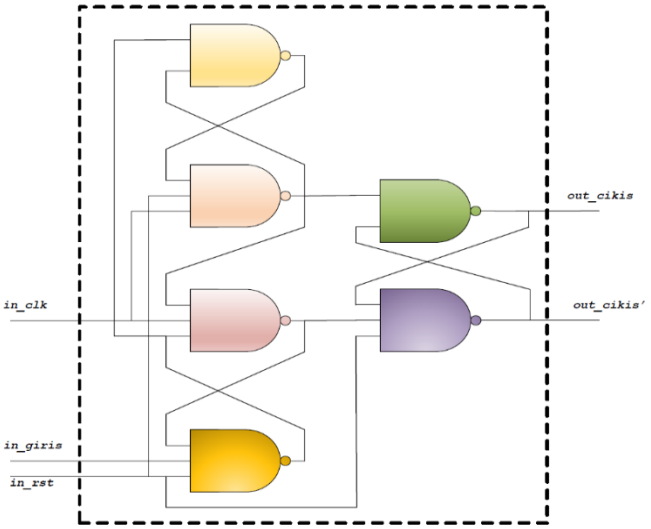
Bu örnekte bir önceki bölümde anlatılan "Yükselen Kenar Tetiklemeli D İki Durumlusu" örneğinden farklı olarak, tasarıma eşzamanlı olmayan bir reset girişi eklenmiştir. Eşzamanlı olmayan reset girişi, tetikleme işaretinden bağımsız olarak, çıkışların sıfırlanmasını sağlamaktadır. Gerçekleştirilen tasarıma ait doğruluk tablosu Tablo 9-3'de verilmiştir.

Tablo 9-3'den de görüleceği üzere **`in_rst`** giriş portu değeri **`'0'`** ve saat darbesi yükselen kenarı mevcut ise **`in_giris`** giriş port değeri **`out_cikis`** çıkış portuna, **`in_giris`** giriş port değerinin değili ise out**`_cikis'`** çıkış portuna atanmaktadır.

Yükselen kenar mevcut olmadığı durumlarda ise çıkış değerleri bir önceki durumlarını korumaktadır. Eğer **`in_rst`** giriş değeri **`'1'`** ise **`in_giris`** giriş portu değeri ve saat darbesi girişinin durumları önemsiz hale gelmektedir. Bu durumda **`out_cikis`** çıkış portu değerine **`'0'`** ve **`out_cikis'`** çıkış portu değerine **`'1'`** atanmaktadır. Şekil 9-5'de ise "Eşzamanlı Olmayan Resetli Yükselen Kenar Tetiklemeli D İki Durumlusu" tasarımına ait mantık devresi verilmiştir.

Tablo 9-3 Eşzamanlı olmayan resetli yükselen kenar tetiklemeli D iki durumlusu doğruluk tablosu

in_rst	in_clk	in_giris	out_cikis(+)	out_cikis'(+)
0	Çıkan kenar	0	0	1
	Çıkan kenar	1	1	0
	Çıkan kenar yok	X	out_cikis	out_cikis'
1	Çıkan kenar	0	0	1
	Çıkan kenar	1	0	1
	Çıkan kenar yok	X	0	1



Şekil 9-5 Eşzamanlı olmayan resetli yükselen kenar tetiklemeli D iki durumlusu mantık devresi

Örnek 9.3: Aşağıda eşzamanlı olmayan resetli yükselen kenar tetiklemeli D iki durumlu mantık devresinin gerçekleştirildiği **eszam_olmayan_rstli_d_mandali.vhd** VHDL kodu verilmiştir. Tablo 9-3 ve Şekil 9-5'den de görülceği üzere **eszam_olmayan_rstli_d_mandali** varlığımız üç giriş ve iki çıkış portuna sahiptir. Port tanımlamaya ilişkin bildirimler 5-11 satırları arasında yapılmatıdır.

16. satırda **r_cikis** sinyali tanımlanmaktadır. 20. satırda tanımlanan söz dizimi ile **process**'in **in_clk**, **in_rst** ve **in_giris** giriş portları değerlerinde meydana gelen değişiklikler ile aktif olacağı belirtilmektedir. 23. satırda tanımlı koşul ifadesi ile **in_rst** giriş portu değeri **'1'** olduğunda **in_clk** ve **in_giris** sinyallerinin durumu farketmeksiniz **r_cikis** sinyaline **'0'** değeri atanmaktadır. Bu durumda **out_cikis** çıkış portu değeri **'0'** ve **out_cikis_degil** çıkış portu değeri **'1'** olmaktadır. **in_rst** giriş portunun diğer durumlarında ise 25. satırda tanımlı koşul ifadesinin sağlanması yani **in_clk** giriş portunda yükselen kenar meydana gelmesi durumunda **in_giris** giriş port değeri **r_cikis** sinyaline atanmaktadır. Aksi durumlarda ise **r_cikis** sinyali bir önceki değerini korumaktadır. 31 satırda **out_cikis** portuna **r_cikis** sinyali atanırken, 32. satırda ise **out_cikis_degil** portuna **r_cikis** sinyalinin değili atanmaktadır.

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.all;
3.
4. entity eszam_olmayan_rstli_d_mandali is
5.   port (
6.     in_clk : in  std_logic;
7.     in_rst : in std_logic;
8.     in_giris : in  std_logic;
```

```
9.      out_cikis : out std_logic;
10.     out_cikis_degil : out std_logic
11.   );
12.end eszam_olmayan_rstli_d_mandali;
13.
14.architecture Behavioral of eszam_olmayan_rstli_d_mandali is
15.
16.   signal r_cikis : std_logic := '0';
17.
18.begin
19.
20.   process(in_clk, in_rst, in_giris)
21.   begin
22.
23.     if in_rst = '1' then
24.       r_cikis <= '0';
25.     elsif in_clk'event and in_clk = '1' then
26.       r_cikis <= in_giris;
27.     end if;
28.
29.   end process;
30.
31.   out_cikis <= r_cikis;
32.   out_cikis_degil <= (not r_cikis);
33.
34.end Behavioral;
```

Aşağıda ise **eszam_olmayan_rstli_d_mandali** varlığının benzetim yapılabilmesi için **tb_eszam_olmayan_rstli_d_mandali.vhd** sınama kodu (test bench) verilmiştir ve benzetim çıktısı Şekil 9-6'da gösterilmiştir. Şekil 9-6'da gösterilen benzetim sonucunda:

1. adım : in_rst giriş portu değerinin **'0'** olması ve **in_clk** giriş portunda meydana gelen yükselen kenarı ile birlikte **in_giris** giriş portu değeri **r_cikis** sinyaline atanmaktadır ve **r_cikis** sinyalinin değeri **'0'** olmaktadır. **r_cikis** sinyalinin değerinin **'0'** olması ile **out_cikis** çıkış portu değerine **'0'**, **out_cikis_degil** çıkış portu değerine ise **'1'** atanmaktadır.

2. adım: in_rst giriş portu değerinin **'1'** olması nedeniyle **r_cikis** sinyaline **'0'** atanmaktadır. **r_cikis** sinyalinin değerinin **'0'** olması ile **out_cikis** çıkış portu değerine **'0'**, **out_cikis_degil** çıkış portu değerine ise **'1'** atanmaktadır.

3. adım: in_rst giriş portu değerinin **'0'** olması ve **in_clk** giriş portunda meydana gelen yükselen kenar ile birlikte **in_giris** giriş portu değeri **r_cikis** sinyaline atanmaktadır ve **r_cikis** sinyalinin değeri **'1'** olmaktadır. **r_cikis** sinyalinin değerinin **'1'** olması ile **out_cikis** çıkış portu değerine **'1'**, **out_cikis_degil** çıkış portu değerine ise **'0'** atanmaktadır.

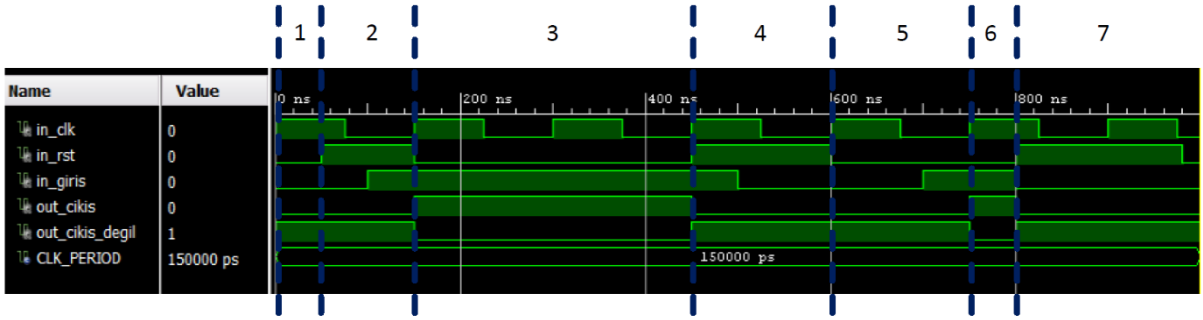
4. adım: in_rst giriş portu değerinin **'1'** olması nedeniyle **r_cikis** sinyaline **'0'** atanmaktadır. **r_cikis** sinyalinin değerinin **'0'** olması ile **out_cikis** çıkış portu değerine **'0'**, **out_cikis_degil**

çıkış portu değerine ise **'1'** atanmaktadır. 4. adım içerisinde **in_giris** giriş port değerinin **'1'** olmasına rağmen **in_rst** giriş portunun değerinin **'1'** olması nedeniyle çıkışa aktarılamamıştır

5. adım: in_rst giriş portu değerinin **'0'** olması ve **in_clk** giriş portunda meydana gelen yükselen kenar ile birlikte **in_giris** giriş portu değeri **r_cikis** sinyaline atanmaktadır ve **r_cikis** sinyalinin değeri **'0'** olmaktadır. **r_cikis** sinyalinin değerinin **'0'** olması ile **out_cikis** çıkış portu değerine **'0'**, **out_cikis_degil** çıkış portu değerine ise **'1'** atanmaktadır. 5. adım içerisinde **in_giris** giriş portu değeri **'1'** olmasına rağmen saat darbesi yükselen kenarı mevcut olmadığından **r_cikis** sinyaline atanamamıştır.

6. adım: in_rst giriş portu değerinin **'0'** olması ve **in_clk** giriş portunda meydana gelen yükselen kenar ile birlikte **in_giris** giriş portu değeri **r_cikis** sinyaline atanmaktadır ve **r_cikis** sinyalinin değeri **'1'** olmaktadır. **r_cikis** sinyalinin değerinin **'1'** olması ile **out_cikis** çıkış portu değerine **'1'**, **out_cikis_degil** çıkış portu değerine ise **'0'** atanmaktadır.

7. adım: in_rst giriş portu değerinin **'1'** olması nedeniyle **r_cikis** sinyaline **'0'** atanmıştır. **r_cikis** sinyalinin değerinin **'0'** olması ile **out_cikis** çıkış portu değerine **'0'**, **out_cikis_degil** çıkış portu değerine ise **'1'** atanmaktadır.



Şekil 9-6 eszam_olmayan_rstli_d_mandali varlığı benzetim çıktısı

tb_eszam_olmayan_rstli_d_mandali.vhd VHDL kodun amacı **eszam_olmayan_rstli_d_mandali** varlığının benzetiminin yapmak olduğundan dolayı **tb_eszam_olmayan_rstli_d_mandali** varlığına ait bir port tanımlaması yapılmamıştır. **tb_eszam_olmayan_rstli_d_mandali.vhd** VHDL kodu sentezlenemez sadece benzetim için kullanılabilmektedir. 19. satırda saat darbesinin periyodunun belirlendiği sabit tanımlaması yapılmıştır. Tanımlamada saat darbesi periyodu **150 ns** olarak belirlenmiştir. Test kodlarının oluşturulması amacı ile kullanılacak olan sinyallerin tanımlama işlemleri 20-24. satırlarda yapılmıştır.

28-35. satırlar arasında tanımlanan **process** ile **in_clk** saat darbesinin davranışı tanımlanmaktadır. 30. satırda **in_clk** sinyaline **'1'** değeri atanmakta ve 31. satırda tanımlanan ifade ile saat darbesi periyodunun yarısı kadar beklenmektedir. 33. satırda **in_clk** sinyaline **'0'** değeri atanmakta ve 34. satırda tanımlanan ifade ile saat darbesi periyodunun yarısı kadar beklenmektedir. Bu şekilde %50 doluluk oranında saat darbesi elde edilmiştir.

37-46 satırları arasında tanımlanan **process** içerisinde **in_rst** sinyalinin davranışı tanımlanmaktadır. 39. satırda tanımlanan ifade ile **in_rst** sinyalinin **0-50 ns** aralığında alacağı değerin **'0'** olacağı belirtilmektedir. 40. satırda tanımlanan ifade ile **in_rst** sinyalinin **50-150 ns** aralığında alacağı değerin **'1'** olacağı belirtilmektedir. 41. satırda tanımlanan ifade ile **in_rst** sinyalinin **150-450 ns** aralığında alacağı değerin **'0'** olacağı belirtilmektedir. 42 satırda tanımlanan ifade ile **in_rst** sinyalinin **450-600 ns** aralığında alacağı değerin **'1'** olacağı belirtilmektedir. 43. satırda tanımlanan ifade ile **in_rst** sinyalinin **600-800 ns** aralığında alacağı değerin **'0'** olacağı belirtilmektedir. 44. satırda tanımlanan ifade ile **in_rst** sinyalinin **800-980 ns** aralığında alacağı değerin **'1'** olacağı belirtilmektedir. 45. satırda tanımlanan ifade ile **in_rst** sinyalinin **980-100 ns** aralığında alacağı değerin **'0'** olacağı belirtilmektedir.

48-55. satırları arasında tanımlanan **process** içerisinde **in_giris** sinyalinin davranışı tanımlanmaktadır. 50. satırda tanımlanan ifade ile **in_giris** sinyalinin **0-100 ns** aralığında alacağı değerin **'0'** olacağı belirtilmektedir. 51. satırda tanımlanan ifade ile **in_giris** sinyalinin **100-500 ns** aralığında alacağı değerin

'1' olacağı belirtilmektedir. 52. satırda tanımlanan ifade ile **in_giris** sinyalinin **500-700 ns** aralığında alacağı değerin **'0'** olacağı belirtilmektedir. 53. satırda tanımlanan ifade ile **in_giris** sinyalinin **700-800 ns** aralığında alacağı değerin **'1'** olacağı belirtilmektedir. 54. satırda tanımlanan ifade ile **in_giris** sinyalinin **800-1000 ns** aralığında alacağı değerin **'0'** olacağı belirtilmektedir.

eszam_olmayan_rstli_d_mandali varlığının **tb_eszam_olmayan_rstli_d_mandali** varlığında alt devre olarak kullanılabilmesi için gerekli **component** tanımlama işlemleri 9-17. satırlar arasında yapılmıştır. 57-64. satırlar arasında ise **eszam_olmayan_rstli_d_mandali** alt devresine ilişkin bağlantılar yapılmaktadır.

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.all;
3.
4. entity tb_eszam_olmayan_rstli_d_mandali is
5. end tb_eszam_olmayan_rstli_d_mandali;
6.
7. architecture Behavioral of tb_eszam_olmayan_rstli_d_mandali is
8.
9.   component eszam_olmayan_rstli_d_mandali
10.  Port (
11.    in_clk : in std_logic;
12.    in_rst : in std_logic;
13.    in_giris : in std_logic;
14.    out_cikis : out std_logic;
15.    out_cikis_degil : out std_logic
16.  );
17.  end component;
18.
19.  constant CLK_PERIOD : time := 150 ns;
20.  signal in_clk : std_logic := '0';
21.  signal in_rst : std_logic := '0';
22.  signal in_giris : std_logic := '0';
23.  signal out_cikis : std_logic := '0';
24.  signal out_cikis_degil : std_logic := '0';
25.
26.begin
27.
28.  process
29.  begin
30.    in_clk <= '1';
31.    wait for CLK_PERIOD / 2;
32.
33.    in_clk <= '0';
```

```
34.     wait for CLK_PERIOD / 2;
35.   end process;
36.
37.   process
38.   begin
39.     in_rst <= '0'; wait for 50 ns;
40.     in_rst <= '1'; wait for 100 ns;
41.     in_rst <= '0'; wait for 300 ns;
42.     in_rst <= '1'; wait for 150 ns;
43.     in_rst <= '0'; wait for 200 ns;
44.     in_rst <= '1'; wait for 180 ns;
45.     in_rst <= '0'; wait for 20 ns;
46.   end process;
47.
48.   process
49.   begin
50.     in_giris <= '0'; wait for 100 ns;
51.     in_giris <= '1'; wait for 400 ns;
52.     in_giris <= '0'; wait for 200 ns;
53.     in_giris <= '1'; wait for 100 ns;
54.     in_giris <= '0'; wait for 200 ns;
55.   end process;
56.
57.   eszam_olmayan_rstli_d_mandali_map :
58.   eszam_olmayan_rstli_d_mandali port map(
59.     in_clk => in_clk,
60.     in_rst => in_rst,
61.     in_giris => in_giris,
62.     out_cikis => out_cikis,
63.     out_cikis_degil => out_cikis_degil
64.   );
65.end Behavioral;
```

9.4. Eşzamanlı Resetli Yükselen Kenar Tetiklemeli D İki Durumlusu

Eşzamanlı reset işlemi tetikleme işaretine (saat darbesine) bağlı olarak çalışan bir tasarımdır. Eşzamanlı olmayan reset işleminden farklı olarak reset girişi etkinleştirilse bile çıkışın sıfırlanması için saat darbesinin yükselen kenarı (tasarıma göre düşen kenarı da olabilir) beklenmektedir.

“Eşzamanlı Resetli Yükselen Kenar Tetiklemeli D İki Durumlusu” tasarımına ait doğruluk tablosu Tablo 9-4’de verilmiştir. Tablo 9-4’den de görüleceği üzere saat darbesi yükselen kenarı mevcut ve **in_rst** giriş portu değeri **'0'** ise **in_giris** giriş port değeri **out_cikis** çıkış portuna, **in_giris** giriş port değerinin değili ise **out_cikis'** çıkış portuna atanmaktadır.

Yükselen kenar mevcut olmadığı durumlarda ise çıkış değerleri bir önceki durumlarını korumaktadır. Eğer saat darbesi yükselen kenarı mevcut ve **in_rst** giriş değeri **'1'** ise **in_giris** giriş portu değeri önemsiz hale gelmektedir. Bu durumda **out_cikis** çıkış portu değerine **'0'** ve **out_cikis'** çıkış portu değerine **'1'** atanmaktadır. Saat darbesi yükselen kenarı mevcut olmadığı durumlarda ise **in_rst** ve **in_giris** giriş portu değerleri önemsiz hale gelmektedir ve çıkış değerleri bir önceki değerlerini korumaktadır.

Tablo 9-4 Eşzamanlı resetli yükselen kenar tetiklemeli D iki durumlusu doğruluk tablosu

in_clk	in_rst	in_giris	out_cikis+
Çıkan kenar	0	0	0
Çıkan kenar		1	1
Çıkan kenar yok		X	out_cikis
Çıkan kenar	1	0	0
Çıkan kenar		1	0
Çıkan kenar yok		X	out_cikis

Örnek 9.4: Aşağıda “Eşzamanlı Resetli Yükselen Kenar Tetiklemeli D İki Durumlusu” mantık devresinin gerçekleştirildiği **eszam_rstli_d_mandali.vhd** VHDL kodu verilmiştir. Tablo 9-4’den de görülceği üzere **eszam_olmayan_rstli_d_mandali** varlığımız üç giriş ve iki çıkış portuna sahiptir. Port tanımlamaya ilişkin bildirimler 5-11 satırları arasında yapılmatıdır.

16. satırda **r_cikis** sinyali tanımlanmaktadır. 20. satırda **out_cikis** portuna **r_cikis** sinyali atanırken, 21. satırda ise **out_cikis_degil** portuna **r_cikis** sinyalinin değili atanmaktadır. 23. satırda tanımlanan söz dizimi ile **process**’in **in_clk, in_rst** ve **in_giris** değerlerinden meydana gelen değişiklikler ile aktif olacağı belirtilmektedir. 25. satırda tanımlı koşul ifadesi ile koşulun sağlanması yani **in_clk** giriş sinyalinin yükselen kenarının meydana gelmesi durumunda 26. satırda tanımlı koşul ifadesi ile ile **in_rst** girişi **'1'** değerini aldığında **in_giris** sinyalinin durumu farketmeksiniz **r_cikis** sinyaline **'0'** değeri atanmaktadır. Bu durumda **out_cikis** değeri **'0'** ve **out_cikis_degil** değeri **'1'** olmaktadır. **in_rst** girişinin diğer durumlarında ise **in_giris** değeri **r_cikis** sinyaline atanmaktadır. **in_clk** giriş sinyalinin yükselen kenarının meydana gelmemesi durumunda ise **r_cikis** sinyali bir önceki değerini korumaktadır.

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.all;
3.
4. entity eszam_rstli_d_mandali is
5.   port (
6.     in_clk : in std_logic;
7.     in_rst : in std_logic;
8.     in_giris : in  std_logic;
9.     out_cikis : out std_logic;
10.    out_cikis_degil : out std_logic
11.  );
12.end eszam_rstli_d_mandali;
13.
14.architecture Behavioral of eszam_rstli_d_mandali is
15.
```

```
16.  signal r_cikis : std_logic := '0';
17.
18.begin
19.
20.  process(in_clk, in_rst, in_giris)
21.  begin
22.
23.    if in_clk'event and in_clk = '1' then
24.      if in_rst = '1' then
25.        r_cikis <= '0';
26.      else
27.        r_cikis <= in_giris;
28.      end if;
29.    end if;
30.  end process;
31.
32.  out_cikis <= r_cikis;
33.  out_cikis_degil <= (not r_cikis);
34.
35.end Behavioral;
```

Aşağıda ise **eszam_rstli_d_mandali** varlığının benzetim yapılabilmesi için **tb_eszam_rstli_d_mandali.vhd** sınama kodu (test bench) verilmiştir ve benzetim çıktısı Şekil 9-7'de gösterilmiştir. Şekil 9-7'de gösterilen benzetim sonucunda:

1. adım: **in_clk** giriş portunun yükselen kenarı ile birlikte **in_rst** giriş portu değerinin **'0'** olmasıyla **in_giris** giriş portu değeri **r_cikis** sinyaline atanmaktadır ve **r_cikis** sinyalinin değeri **'0'** olmaktadır. **r_cikis** sinyalinin değerinin **'0'** olması ile **out_cikis** çıkış portuna **'0'**, **out_cikis_degil** çıkış portuna **'1'** değerleri atanmaktadır. 1. adım içerisinde **in_rst** ve **in_giris** giriş portları **'1'** değerini almalarına rağmen saat darbesi yükselen kenarı meydana gelmediği için **r_cikis** sinyal değeri değişmemiştir.

2. adım: **in_clk** giriş portunun yükselen kenarı ile birlikte **in_rst** giriş portu değerinin **'0'** olmasıyla **in_giris** giriş portu değeri **r_cikis** sinyaline atanmıştır ve **r_cikis** sinyalinin değeri **'1'** olmaktadır. **r_cikis** sinyalinin değerinin **'1'** olması ile **out_cikis** çıkış portuna **'1'**, **out_cikis_degil** çıkış portuna **'0'** değerleri atanmaktadır.

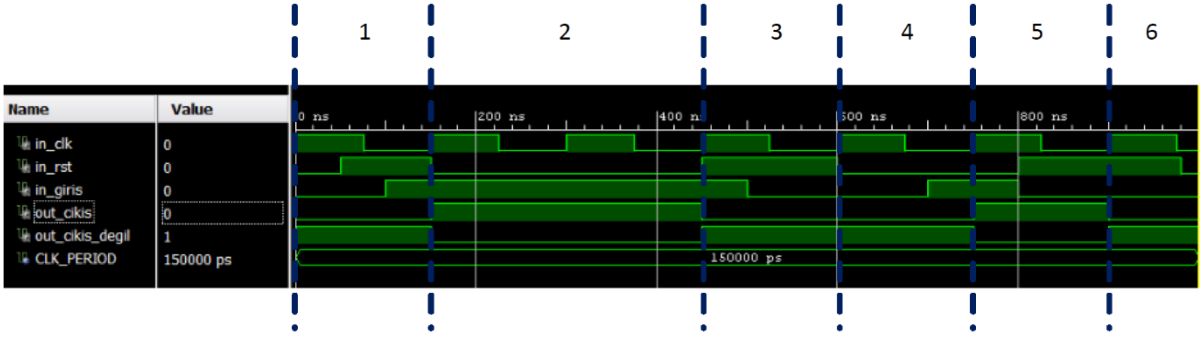
3. adım: **in_clk** giriş portunun yükselen kenarı ile birlikte **in_rst** giriş portu değerinin **'1'** olmasıyla **r_cikis** sinyaline **0'** atanmıştır. **r_cikis** sinyalinin değerinin **'0'** olması ile **out_cikis** çıkış portuna **'0'**, **out_cikis_degil** çıkış portuna **'1'** değerleri atanmaktadır.

4. adım: **in_clk** giriş portunun yükselen kenarı ile birlikte **in_rst** giriş portu değerinin **'0'** olmasıyla **in_giris** giriş portu değeri **r_cikis** sinyaline atanmıştır ve **r_cikis** sinyalinin değeri **'0'** olmaktadır. **r_cikis** sinyalinin değerinin **'0'** olması ile **out_cikis** çıkış portuna **'0'**, **out_cikis_degil** çıkış portuna **'1'** değerleri atanmaktadır. 4. adım içerisinde **in_giris** giriş portu **'1'** değerini almasına rağmen saat darbesi yükselen kenarı meydana gelmediği için **r_cikis** sinyali değeri değişmemiştir.

5. adım: **in_clk** giriş portunun yükselen kenarı ile birlikte **in_rst** giriş portu değerinin **'0'** olmasıyla **in_giris** giriş portu değeri **r_cikis** sinyaline atanmıştır ve **r_cikis** sinyalinin değeri **'1'** olmaktadır. **r_cikis** sinyalinin değerinin **'1'** olması ile **out_cikis** çıkış portuna **'1'**, **out_cikis_degil** çıkış portuna **'0'** değerleri atanmaktadır. 5. adım içerisinde **in_rst** giriş portu **'1'** ve **in_giris** giriş portu **'0'**

değerini almasına rağmen saat darbesi yükselen kenarı meydana gelmediği için **r_cikis** sinyali değeri değişmemiştir.

6. adım: in_clk giriş portunun yükselen kenarı ile birlikte **in_rst** giriş portu değerinin **'1'** olmasıyla **r_cikis** sinyaline **0'** atanmıştır. **r_cikis** sinyalinin değerinin **'0'** olması ile **out_cikis** çıkış portuna **'0'**, **out_cikis_degil** çıkış portuna **'1'** değerleri atanmaktadır. 6. adım içerisinde **in_rst** giriş portu **'0'** değerini almasına rağmen saat darbesi yükselen kenarı meydana gelmediği için **r_cikis** sinyali değeri değişmemiştir.



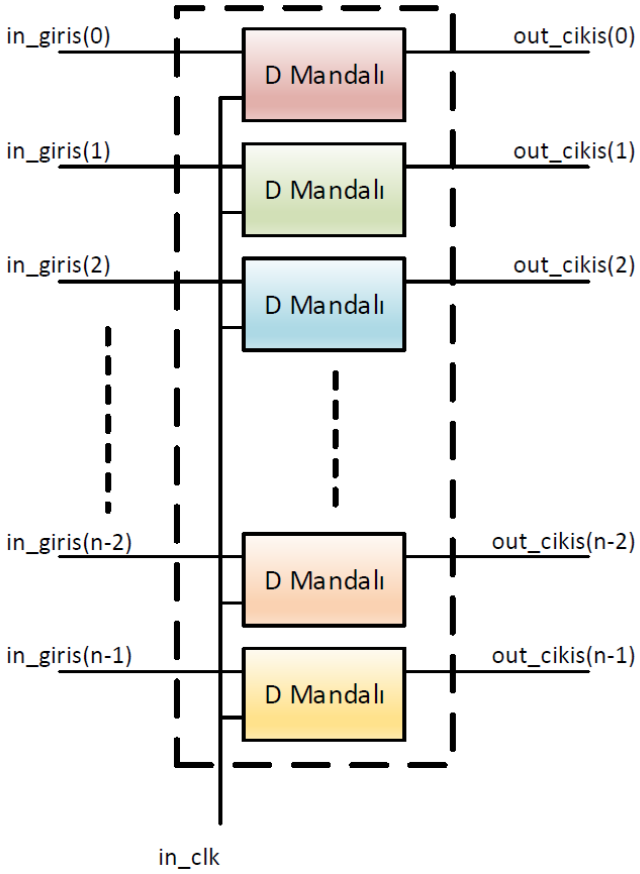
Şekil 9-7 eszam_rstli_d_mandali varlığı benzetim çıktısı

Örnek 9.3'te verilen **tb_eszam_olmayan_rstli_d_mandali.vhd** sınama kodunda (test bench) aşağıda verilen satır numaralarındaki tanımlamalarda değişiklikler yapılarak **eszam_rstli_d_mandali** varlığının benzetimi yapılabilmektedir.

```
9. component eszam_rstli_d_mandali
10.Port (
11.     in_clk : in std_logic;
12.     in_rst : in std_logic;
13.     in_giris : in std_logic;
14.     out_cikis : out std_logic;
15.     out_cikis_degil : out std_logic
16.);
17.end component;

57.eszam_olmayan_rstli_d_mandali_map :
58.eszam_olmayan_rstli_d_mandali port map(
59.     in_clk => in_clk,
60.     in_rst => in_rst,
61.     in_giris => in_giris,
62.     out_cikis => out_cikis,
63.     out_cikis_degil => out_cikis_degil
64.);
```

9.5. Saklayıcı (Register)

Bu kısıma kadar anlatılan örneklerden görüleceği üzere "D Mandalı" bilgiyi saklama özelliğine sahip bir elemandır. 1 Bit uzunluğundaki bilgi (`'1'` ya da `'0'`) "D Mandalı" üzerinde saklanabilmektedir. Bu özellik kullanılarak, birden fazla "D Mandalı" bir araya getirilerek basit bir hafıza elemanı tasarlamak münkündür. Bu şekilde tasarlanmış hafıza elemanlarına "saklayıcı" (register) adı verilmektedir. Şekil 9-8'de "D Mandalı" kullanılarak tasarlanmış bir saklayıcıya ait genel tasarım gösterimi verilmiştir. İstenilen sayıda D Mandalı kullanılarak, istenilen uzunlukta bir saklayıcı tasarlamak mümkündür.



Şekil 9-8 Saklayıcı tasarımı

Örnek 9.5.1: Aşağıda 4 bitlik saklayıcı devresinin gerçekleştirildiği **saklayici_4_bit.vhd** VHDL kodu verilmiştir. **saklayici_4_bit** varlığımıza ilişkin port bildirimleri 5-10. satırları arasında yapılmaktıdır. Tanımlamalardan da görüleceği üzere saklayıcımızın data giriş ve çıkış portları 4 bittir. 17. satırda 4 bitlik **r_saklayici** sinyali tanımlanmaktadır. 20. satırda **out_cikis** portuna **r_saklayici** sinyali atanmaktadır. 21. satırda tanımlanan söz dizimi ile **process**'in **in_clk, in_rst** ve **in_giris** değerlerinden meydana gelen değişiklikler ile aktif olacağı belirtilmektedir.

process içerisinde yapılan tanımlamardan da görüleceği üzere saklayıcı tasarımında eş zamanlı olmaya reset kullanılmıştır. 21. satırda **in_rst** giriş portu **'1'** değerini aldığında **in_giris** giriş portunun aldığı değerler farketmeksiniz **r_saklayici** sinyalinin tüm bitlerine **'0'** değeri atanmaktadır. Bu durumda **out_cikis** değerinin tüm bitleride **'0'** olmaktadır. **in_rst** girişinin diğer durumlarında ise **in_clk** giriş portunun yükselen kenarı ile birlikte **in_giris** değeri **r_cikis** sinyaline atanmaktadır. **in_clk** giriş sinyalinin yükselen kenarının meydana gelmemesi durumunda ise **r_cikis** sinyali bir önceki değerini korumaktadır.

```
1. library IEEE;
```

```
2. use IEEE.STD_LOGIC_1164.all;
3.
4. entity saklayici_4_bit is
5.   port (
6.     in_clk : in std_logic;
7.     in_rst : in std_logic;
8.     in_giris : in std_logic_vector(3 downto 0);
9.     out_cikis : out std_logic_vector(3 downto 0)
10.  );
11.end saklayici_4_bit;
12.
13.architecture Behavioral of saklayici_4_bit is
14.
15.  signal r_saklayici : std_logic_vector(3 downto 0);
16.
17.begin
18.
19.  out_cikis <= r_saklayici;
20.
21.  process(in_clk, in_rst, in_giris)
22.  begin
23.    if in_rst = '1' then
24.      r_saklayici <= (others => '0');
25.    elsif rising_edge(in_clk) then
26.      r_saklayici <= in_giris;
27.    end if;
28.  end process;
29.
30.end Behavioral;
```

Aşağıda ise **saklayici_4_bit** varlığının benzetim yapılabilmesi için **tb_saklayici_4_bit.vhd** sınama kodu (test bench) verilmiştir ve benzetim çıktısı Şekil 9-9'da gösterilmiştir. Şekil 9-9'da gösterilen benzetim sonucunda:

1. adım : **in_clk** giriş portunun yükselen kenarı ile birlikte **in_rst** giriş portu değerinin **'0'** olmasıyla **in_giris** giriş portu değeri **r_saklayici** sinyaline atanmaktadır ve **r_saklayici** sinyalinin değeri **"0000"** olmaktadır. **r_saklayici** sinyalinin değerinin **"0000"** olması ile **out_cikis** çıkış portuna **"0000"** değeri atanmaktadır.

2. adım : **in_rst** giriş portu değerinin **'1'** olması nedeniyle **r_saklayici** sinyaline **"0000"** değeri atanmaktadır. **r_saklayici** sinyalinin değerinin **"0000"** olması ile **out_cikis** çıkış portuna **"0000"**, değeri atanmaktadır. 2. adım içerisinde **in_giris** giriş portu değerinin değişmesi ile birlikte saat darbesi yükselen kenarının meydana gelmesine rağmen eş zamanlı olmayan reset ile tasarlanmış saklayıcının çıkışı değişmemektedir.

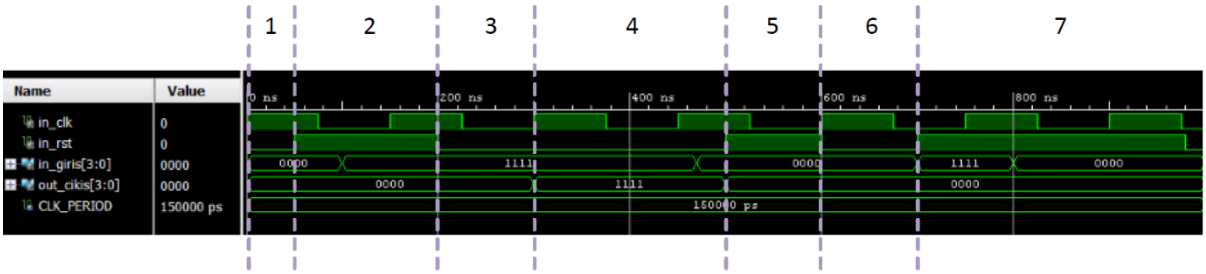
3. adım : **`in_rst`** giriş portu değeri **`'0'`** olmakta fakat saat darbesi yükselen kenarı meydana gelmemesi nedeniyle **`r_saklayici`** sinyalinin değeri değişmemektedir ve bu nedenle **`out_cikis`** çıkış portunun değeri de değişmemiştir.

4. adım : **`in_clk`** giriş portunun yükselen kenarı ile birlikte **`in_rst`** giriş portu değerinin **`'0'`** olmasıyla **`in_giris`** giriş portu değeri **`r_saklayici`** sinyaline atanmaktadır ve **`r_saklayici`** sinyalinin değeri **`"1111"`** olmaktadır. **`r_saklayici`** sinyalinin değerinin **`"1111"`** olması ile **`out_cikis`** çıkış portuna **`"1111"`** değeri atanmaktadır.

5. adım : **`in_rst`** giriş portu değerinin **`'1'`** olmasıyla **`r_saklayici`** sinyalinin değerine **`"0000"`** atanmaktadır. **`r_saklayici`** sinyalinin değerinin **`"0000"`** olması ile **`out_cikis`** çıkış portuna **`"0000"`**değeri atanmaktadır.

6. adım : **`in_clk`** giriş portunun yükselen kenarı ile birlikte **`in_rst`** giriş portu değerinin **`'0'`** olmasıyla **`in_giris`** giriş portu değeri **`r_saklayici`** sinyaline atanmaktadır ve **`r_saklayici`** sinyalinin değeri **`"0000"`** olmaktadır. **`r_saklayici`** sinyalinin değerinin **`"0000"`** olması ile **`out_cikis`** çıkış portuna **`"0000"`** değeri atanmaktadır.

7. adım : **`in_rst`** giriş portu değerinin **`'1'`** olmasıyla **`r_saklayici`** sinyaline **`"0000"`** değeri atanmaktadır. **`r_saklayici`** sinyalinin değerinin **`"0000"`** olması ile **`out_cikis`** çıkış portuna **`"0000"`** değeri atanmaktadır. 7. adım içerisinde **`in_giris`** giriş portu değerinin değişmesi ile birlikte saat darbesi yükselen kenarının meydana gelmesine rağmen eş zamanlı olmayan reset ile tasarlanmış saklayıcının çıkışı değişmemektedir.



Şekil 9-9 saklayici_4_bit varlığı benzetim çıktısı

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3.
4. entity tb_saklayici_4_bit is
5. end tb_saklayici_4_bit;
6.
7. architecture Behavioral of tb_saklayici_4_bit is
8.
9.   component saklayici_4_bit
10.  Port (
11.     in_clk : in std_logic;
12.     in_rst : in std_logic;
13.     in_giris : in std_logic_vector(3 downto 0);
14.     out_cikis : out std_logic_vector(3 downto 0)
15.  );
16.  end component;
```

```
17.
18.  constant CLK_PERIOD : time := 150 ns;
19.  signal in_clk : std_logic := '0';
20.  signal in_rst : std_logic := '0';
21.  signal in_giris : std_logic_vector(3 downto 0) := (others => '0');
22.  signal out_cikis : std_logic_vector(3 downto 0) := (others => '0');
23.
24.begin
25.
26.  process
27.  begin
28.    in_clk <= '1';
29.    wait for CLK_PERIOD / 2;
30.
31.    in_clk <= '0';
32.    wait for CLK_PERIOD / 2;
33.
34.  end process;
35.
36.  process
37.  begin
38.    in_rst <= '0'; wait for 50 ns;
39.    in_rst <= '1'; wait for 150 ns;
40.    in_rst <= '0'; wait for 300 ns;
41.    in_rst <= '1'; wait for 100 ns;
42.    in_rst <= '0'; wait for 100 ns;
43.    in_rst <= '1';  wait for 280 ns;
44.    in_rst <= '0'; wait for 20 ns;
45.  end process;
46.
47.  process
48.  begin
49.    in_giris <= (others => '0'); wait for 100 ns;
50.    in_giris <= (others => '1'); wait for 370 ns;
51.    in_giris <= (others => '0'); wait for 230 ns;
52.    in_giris <= (others => '1'); wait for 100 ns;
53.    in_giris <= (others => '0'); wait for 200 ns;
54.  end process;
55.
56.  saklayici_4_bit_map : saklayici_4_bit
```

```
57.   port map(
58.     in_clk => in_clk,
59.     in_rst => in_rst,
60.     in_giris => in_giris,
61.     out_cikis => out_cikis
62.   );
63.end Behavioral;
```

Örnek 9.5.2: Aşağıda etkinleştirme girişine sahip **n_bit** bitlik saklayıcı devresinin gerçekleştirildiği **saklayici_generic_aktif_sinyal.vhd** VHDL kodu verilmiştir. **saklayici_generic_aktif_sinyal** varlığımıza ilişkin generic bildirimleri 5-7. Satırlarda, port bildirimleri 8-14. satırları arasında yapılmaktıdır.

Tanımlamalardan da görüleceği üzere saklayıcımızın data giriş ve çıkış portları generic parametresi içersinde tanımlanan **n_bit** uzunluğundadır. 19. satırda **n_bit** bitlik **r_saklayici** sinyali tanımlanmaktadır. 23. satırda **out_cikis** portuna **r_saklayici** sinyali atanmaktadır. 25. satırda tanımlanan söz dizimi ile **process**'in **in_clk**, **in_rst** ve **in_giris** değerlerinden meydana gelen değişiklikler ile aktif olacağı belirtilmektedir.

process içerisinde yapılan tanımlamardan da görüleceği üzere saklayıcı tasarımında eş zamanlı olmaya reset kullanılmıştır. 27. satırda **in_rst** giriş portu **'1'** değerini aldığında **in_giris** giriş portunun aldığı değerler farketmeksiniz **r_saklayici** sinyalinin tüm bitlerine **'0'** değeri atanmaktadır. Bu durumda **out_cikis** değerinin tüm bitleride **'0'** olmaktadır. **in_rst** girişinin diğer durumlarında ise **in_clk** giriş portunun yükselen kenarı ile birlikte 30. satırda tanımlanan koşul ifadesi ile **in_en** giriş portunun aktif olması (değerinin **'1'** olması) ile birlikte **in_giris** giriş port değeri **r_cikis** sinyaline atanmaktadır. **in_en** giriş portunun pasif olması (değerinin **'0'** olması) veya **in_clk** giriş sinyalinin yükselen kenarının meydana gelmemesi durumunda ise **r_saklayici** sinyali bir önceki değerini korumaktadır.

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3.
4. entity saklayici_generic_aktif_signal is
5.   Generic(
6.     n_bit : integer := 4
7.   );
8.   Port (
9.     in_clk : in std_logic;
10.    in_rst : in std_logic;
11.    in_en : in std_logic;
12.    in_giris : in std_logic_vector(n_bit - 1 downto 0);
13.    out_cikis : out std_logic_vector(n_bit - 1 downto 0)
14.  );
15.end saklayici_generic_aktif_signal;
16.
17.architecture Behavioral of saklayici_generic_aktif_signal is
18.
```

```
19.   signal r_saklayici : std_logic_vector(n_bit - 1 downto 0);
20.
21.begin
22.
23.   out_cikis <= r_saklayici;
24.
25.   process(in_clk, in_rst, in_en, in_giris)
26.   begin
27.     if in_rst = '1' then
28.       r_saklayici <= (others => '0');
29.     elsif rising_edge(in_clk) then
30.       if in_en = '1' then
31.         r_saklayici <= in_giris;
32.       end if;
33.     end if;
34.   end process;
35.end Behavioral;
```

Aşağıda ise **saklayici_generic_aktif_signal** varlığının benzetim yapılabilmesi için **tb_saklayici_generic_aktif_signal.vhd** sınama kodu (test bench) verilmiştir ve benzetim çıktısı Şekil 9-10'da gösterilmiştir. Şekil 9-10'da gösterilen benzetim sonucunda:

1. adım : in_rst giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **in_en** giriş portunun **'1'** değerini almasıyla **in_giris** giriş portu değeri **r_saklayici** sinyaline atanmaktadır ve **r_saklayici** sinyalinin değeri **"0000"** olmaktadır. **r_saklayici** sinyalinin değerinin **"0000"** olması ile **out_cikis** çıkış portuna **"0000"** değeri atanmaktadır.

2. adım : in_rst giriş portunun değeri **'1'** olması nedeniyle **r_saklayici** sinyaline **"0000"** atanmaktadır. **r_saklayici** sinyalinin değerinin **"0000"** olması ile **out_cikis** değerine **"0000"** atanmaktadır. 2. adım içerisinde **in_giris** giriş portu değeri değişse de eş zamanlı olmayan reset ile tasarlanmış saklayıcının çıkışı değişmemektedir.

3. adım : in_rst giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı meydana gelmesine rağmen **in_en** giriş portu değerinin **'0'** olması nedeniyle **r_saklayici** sinyalinin değeri değişmemiştir ve bu sebeple çıkış değeri de değişmemiştir.

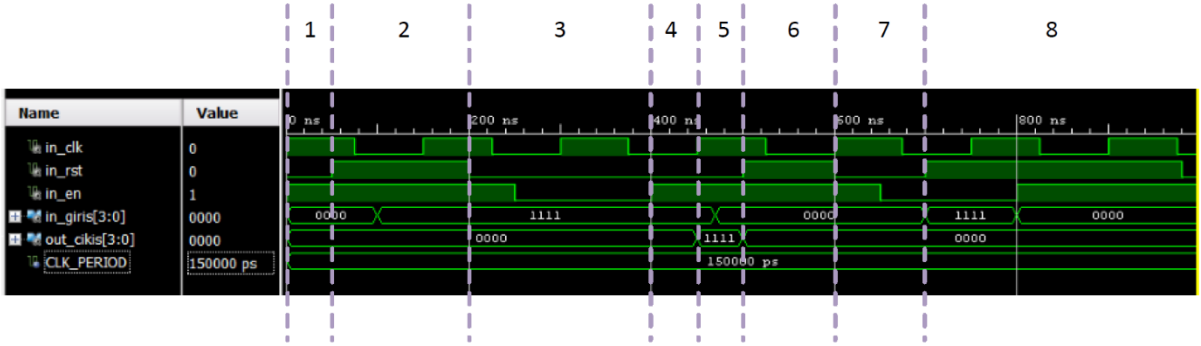
4. adım : in_rst giriş portunun değerinin **'0'** ve **in_en** giriş portu değerinin **'1'** olmasına rağmen **in_clk** giriş portunun yükselen kenarı meydana gelmemesi nedeniyle **r_saklayici** sinyalinin değeri değişmemiştir ve bu sebeple çıkış değeri de değişmemiştir.

5. adım: in_rst giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **in_en** giriş portunun **'1'** değerini almasıyla **in_giris** giriş portu değeri **r_saklayici** değerine atanmıştır ve **r_saklayici** sinyalinin değeri **"1111"** olmaktadır. **r_saklayici** değerinin **"1111"** olması ile **out_cikis** değerine **"1111"** atanmaktadır.

6. adım: in_rst giriş portunun değeri **'1'** olması nedeniyle **r_saklayici** değerine **"0000"** atanmaktadır. **r_saklayici** değerinin **"0000"** olması ile **out_cikis** çıkış portuna **"0000"** değeri atanmaktadır.

7. adım: in_rst giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **in_en** giriş portunun **'1'** değerini almasıyla **in_giris** giriş portu değeri **r_saklayici** değerine atanmıştır ve **r_saklayici** sinyalinin değeri **"0000"** olmaktadır. **r_saklayici** değerinin **"0000"** olması ile **out_cikis** çıkış portuna **"0000"** değeri atanmaktadır.

8. adım: **`in_rst`** giriş portunun değeri **`'1'`** olması nedeniyle **`r_saklayici`** değerine **`"0000"`** atanmaktadır. **`r_saklayici`** değerinin **`"0000"`** olması ile **`out_cikis`** çıkış portuna **`"0000"`** atanmaktadır.



Şekil 9-10 saklayici_generic_aktif_signal varlığı benzetim çıktısı

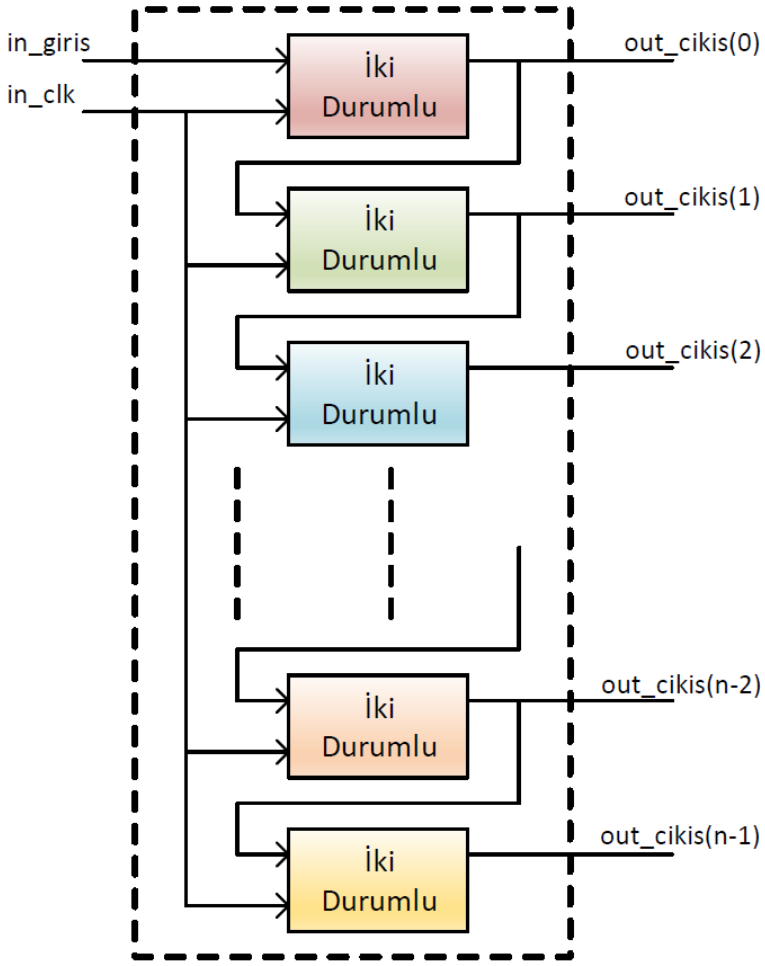
```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3.
4. entity tb_saklayici_generic_aktif_signal is
5. end tb_saklayici_generic_aktif_signal;
6.
7. architecture Behavioral of tb_saklayici_generic_aktif_signal is
8.
9.   component saklayici_generic_aktif_signal
10.  Generic(
11.    n_bit : integer := 4
12.  );
13.  Port (
14.    in_clk : in std_logic;
15.    in_rst : in std_logic;
16.    in_en : in std_logic;
17.    in_giris : in std_logic_vector(n_bit - 1 downto 0);
18.    out_cikis : out std_logic_vector(n_bit - 1 downto 0)
19.  );
20.  end component;
21.
22.  constant CLK_PERIOD : time := 150 ns;
23.  signal in_clk : std_logic := '0';
24.  signal in_rst : std_logic := '0';
25.  signal in_en: std_logic := '0';
26.  signal in_giris : std_logic_vector(3 downto 0) := (others => '0');
27.  signal out_cikis : std_logic_vector(3 downto 0) := (others => '0');
28.
```

```
29.begin
30.
31.
32.  Process
33.  Begin
34.    in_clk <= '1';
35.    wait for CLK_PERIOD / 2;
36.
37.    in_clk <= '0';
38.    wait for CLK_PERIOD / 2;
39.
40.  end process;
41.
42.  Process
43.  Begin
44.    in_rst <= '0'; wait for 50 ns;
45.    in_rst <= '1'; wait for 150 ns;
46.    in_rst <= '0'; wait for 300 ns;
47.    in_rst <= '1'; wait for 100 ns;
48.    in_rst <= '0'; wait for 100 ns;
49.    in_rst <= '1';  wait for 280 ns;
50.    in_rst <= '0'; wait for 20 ns;
51.  end process;
52.
53.  process
54.  begin
55.    in_en <= '1'; wait for 250 ns;
56.    in_en <= '0'; wait for 150 ns;
57.    in_en <= '1'; wait for 250 ns;
58.    in_en <= '0'; wait for 150 ns;
59.    in_en <= '1'; wait for 200 ns;
60.  end process;
61.
62.
63.  process
64.  begin
65.    in_giris <= (others => '0'); wait for 100 ns;
66.    in_giris <= (others => '1'); wait for 370 ns;
67.    in_giris <= (others => '0'); wait for 230 ns;
68.    in_giris <= (others => '1'); wait for 100 ns;
```

```
69.     in_giris <= (others => '0'); wait for 200 ns;
70.   end process;
71.
72.   saklayici_generic_aktif_signal_map :
73.   saklayici_generic_aktif_signal
74.   generic map(
75.     n_bit => 4
76.   )
77.   port map(
78.     in_clk => in_clk,
79.     in_rst => in_rst,
80.     in_en => in_en,
81.     in_giris => in_giris,
82.     out_cikis => out_cikis
83.   );
84.
85.end Behavioral;
```

9.6. Kaydırmalı Saklayıcı (Shift Register)

Sık kullanılan bir diğer saklayıcı tasarımı da “Kaydırmalı Saklayıcı” adı verilen tasarımdır. Bu tasarımda giriş portundan verilen veriler sırayla kayarak saklayıcı içinde D İki Durumluları (D Flip-Flop) arasında aktarılır. Şekil 9-11’de bu saklayıcıya mimari gösterimi verilmiştir. Kaydırmalı Saklayıcı çıkışları her saat darbesi ile güncellendiği için çıkışlar gelen veriye göre sürekli değişiklik gösterecektir.



Şekil 9-11 Kaydırmalı saklayıcı tasarımı

Örnek 9.6.1: Aşağıda 4 bitlik, sağa kaydırma yapan saklayıcı devresinin gerçekleştirildiği **kaydirmali_saklayici.vhd** VHDL kodu verilmiştir. **kaydirmali_saklayici** varlığımıza ilişkin port bildirimleri 5-10. satırlar arasında yapılmıştır. Kaydırmalı saklayıcı tasarımında 1 bitlik saat darbesi giriş portu, 1 bitlik reset giriş portu, 1 bitlik data giriş portu ve 4 bitlik çıkış portu mevcuttur. 15. satırda 4 bitlik **r_saklayici** sinyali tanımlanmaktadır. 19. satırda **out_cikis** portuna **r_saklayici** sinyali atanmaktadır. 21. satırda tanımlanan söz dizimi ile **process**'in **in_clk**, **in_rst** ve **in_giris** giriş portlarında meydana gelen değişiklikler ile aktif olacağı belirtilmektedir.

process içerisinde yapılan tanımlamardan da görüleceği üzere saklayıcı tasarımında eş zamanlı olmayan reset kullanılmıştır. 23. satırda **in_rst** giriş portu **'1'** değerini aldığında **in_giris** giriş portunun aldığı değerler farketmeksiniz **r_saklayici** sinyalinin tüm bitlerine **'0'** değeri atanmaktadır. Bu durumda **out_cikis** çıkış portunun tüm bitleride **'0'** atanmaktadır.

in_rst giriş portunun diğer durumlarında ise **in_clk** giriş portunun yükselen kenarı ile birlikte 26. satırda tanımlanan koşul ifadesinin gerçeklenmesiyle **in_giris** giriş port değeri **r_saklayici** sinyalinin en anlamlı bitine atanmaktadır. **r_saklayici** sinyalinin 3. biti 2.bite, 2.biti 1.bite ve 1. bit en anlamsız bite atanmtakdır. **r_saklayici** sinyal olarak tanımlandığından dolayı tüm atama işlemleri **process** sonunda yapılmaktadır. **in_clk** giriş portunda yükselen kenarının meydana gelmemesi durumunda ise **r_saklayici** sinyali bir önceki değerini korumaktadır.

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

```
3.
4. entity kaydirmali_saklayici is
5.   Port (
6.     in_clk : in std_logic;
7.     in_rst : in std_logic;
8.     in_giris : in std_logic;
9.     out_cikis : out std_logic_vector(3 downto 0)
10.  );
11.end kaydirmali_saklayici;
12.
13.architecture Behavioral of kaydirmali_saklayici is
14.
15.  signal r_saklayici : std_logic_vector(3 downto 0) := (others => '0');
16.
17.begin
18.
19.  out_cikis <= r_saklayici;
20.
21.  process(in_clk, in_rst, in_giris)
22.  begin
23.    if in_rst = '1' then
24.      r_saklayici <= (others => '0');
25.    elsif rising_edge(in_clk) then
26.      r_saklayici(3) <= in_giris ;
27.      r_saklayici(2) <= r_saklayici(3) ;
28.      r_saklayici(1) <= r_saklayici(2) ;
29.      r_saklayici(0) <= r_saklayici(1) ;
30.
31.    end if;
32.  end process;
33.
34.end Behavioral;
```

Aşağıda ise **kaydirmali_saklayici** varlığının benzetim yapılabilmesi için **tb_kaydirmali_saklayici.vhd** sınama kodu verilmiştir ve benzetim çıktısı Şekil 9-12'de gösterilmiştir. Şekil 9-12'de gösterilen benzetim sonucunda:

1. adım: **in_rst** giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **in_giris** giriş portu değeri (**'1'**) **r_saklayici** sinyalinin en anlamlı bitine atanmaktadır ve diğer bitlerin sırasıyla atama işlemleri yapılmaktadır. Atama işlemi ile birlikte **r_sakalyici** sinyalinin değeri **"1000"** olmaktadır. **r_saklayici** değerinin **"1000"** olması ile **out_cikis** çıkış portuna **"1000"** değeri atanmaktadır.

2. adım: **in_rst** giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **in_giris** giriş portu değeri (**'0'**) **r_saklayici** sinyalinin en anlamlı bitine atanmaktadır ve diğer bitlerin sırasıyla atama işlemleri yapılmaktadır. Atama işlemi ile birlikte **r_sakalyici** sinyalinin değeri **"0100"** olmaktadır. **r_saklayici** değerinin **"0100"** olması ile **out_cikis** çıkış portuna **"0100"** değeri atanmaktadır.

3. adım: **in_rst** giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **in_giris** giriş portu değeri (**'1'**) **r_saklayici** sinyalinin en anlamlı bitine atanmaktadır ve diğer bitlerin sırasıyla atama işlemleri yapılmaktadır. Atama işlemi ile birlikte **r_sakalyici** sinyalinin değeri **"1010"** olmaktadır. **r_saklayici** değerinin **"1010"** olması ile **out_cikis** çıkış portuna **"1010"** değeri atanmaktadır.

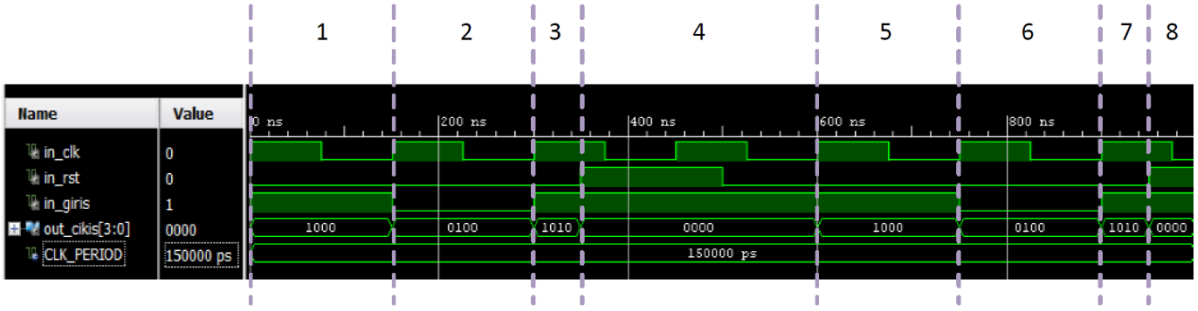
4. adım : **in_rst** giriş portunun değeri **'1'** olması nedeniyle **r_saklayici** değerine **"0000"** atanmaktadır. **r_saklayici** değerinin **"0000"** olması ile **out_cikis** çıkış portuna **"0000"** değeri atanmaktadır.

5. adım : **in_rst** giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **in_giris** giriş portu değeri (**'1'**) **r_saklayici** sinyalinin en anlamlı bitine atanmaktadır ve diğer bitlerin sırasıyla atama işlemleri yapılmaktadır. Atama işlemi ile birlikte **r_sakalyici** sinyalinin değeri **"1000"** olmaktadır. **r_saklayici** değerinin **"1000"** olması ile **out_cikis** çıkış portuna **"1000"** değeri atanmaktadır.

6. adım : **in_rst** giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **in_giris** giriş portu değeri (**'0'**) **r_saklayici** sinyalinin en anlamlı bitine atanmaktadır ve diğer bitlerin sırasıyla atama işlemleri yapılmaktadır. Atama işlemi ile birlikte **r_sakalyici** sinyalinin değeri **"0100"** olmaktadır. **r_saklayici** değerinin **"0100"** olması ile **out_cikis** çıkış portuna **"0100"** değeri atanmaktadır.

7. adım : **in_rst** giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **in_giris** giriş portu değeri (**'1'**) **r_saklayici** sinyalinin en anlamlı bitine atanmaktadır ve diğer bitlerin sırasıyla atama işlemleri yapılmaktadır. Atama işlemi ile birlikte **r_sakalyici** sinyalinin değeri **"1010"** olmaktadır. **r_saklayici** değerinin **"1010"** olması ile **out_cikis** çıkış portuna **"1010"** değeri atanmaktadır.

8. adım: **in_rst** giriş portunun değeri **'1'** olması nedeniyle **r_saklayici** değerine **"0000"** atanmaktadır. **r_saklayici** değerinin **"0000"** olması ile **out_cikis** çıkış portuna **"0000"** değeri atanmaktadır.



Şekil 9-12 signal veri nesnesi kullanılan kaydirmali_saklayici varlığı benzetim çıktısı

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3.
4. entity tb_kaydirmali_saklayici is
5. end tb_kaydirmali_saklayici;
6.
```

```
7. architecture Behavioral of tb_kaydirmali_saklayici is
8.
9.   component kaydirmali_saklayici
10.  Port (
11.    in_clk : in std_logic;
12.    in_rst : in std_logic;
13.    in_giris : in std_logic;
14.    out_cikis : out std_logic_vector(3 downto 0)
15.  );
16.  end component;
17.
18.  constant CLK_PERIOD : time := 150 ns;
19.  signal in_clk : std_logic := '0';
20.  signal in_rst : std_logic := '0';
21.  signal in_giris : std_logic := '0';
22.  signal out_cikis : std_logic_vector(3 downto 0) := (others => '0');
23.
24.begin
25.
26.  process
27.  begin
28.    in_clk <= '1';    wait for CLK_PERIOD / 2;
29.    in_clk <= '0';    wait for CLK_PERIOD / 2;
30.  end process;
31.
32.  process
33.  begin
34.    in_rst <= '0'; wait for 350 ns;
35.    in_rst <= '1'; wait for 150 ns;
36.    in_rst <= '0'; wait for 450 ns;
37.    in_rst <= '1'; wait for 50 ns;
38.  end process;
39.
40.  process
41.  begin
42.    in_giris <= '1'; wait for CLK_PERIOD;
43.    in_giris <= '0'; wait for CLK_PERIOD;
44.    in_giris <= '1'; wait for 3 * CLK_PERIOD;
45.    in_giris <= '0'; wait for CLK_PERIOD;
46.    in_giris <= '1'; wait for CLK_PERIOD;
```

```
47.     in_giris <= '0'; wait for CLK_PERIOD;
48.   end process;
49.
50.   kaydirmali_saklayic_map : kaydirmali_saklayici
51.   port map(
52.     in_clk => in_clk,
53.     in_rst => in_rst,
54.     in_giris => in_giris,
55.     out_cikis => out_cikis
56.   );
57.end Behavioral;
```

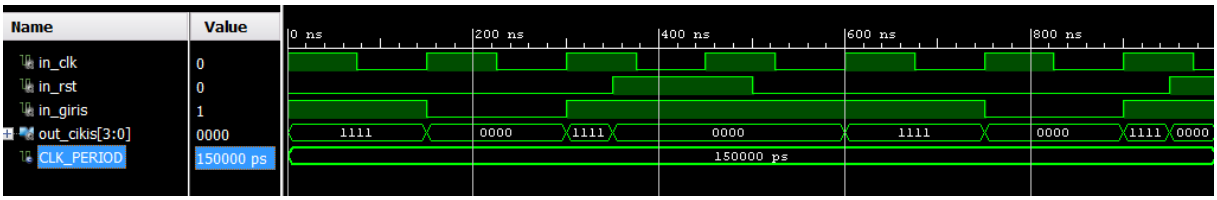
Örnek 9.6.2: **Örnek 9.6.1**'de verilen VHDL kodunda ufak bir değişiklik yapılarak tasarımda **signal** veri nesnesi yerine **variable** veri nesnesi kullanılmıştır. Bu tasarıma ait VHDL kodu aşağıda verilmiştir. 15. satırda 4 bitlik **r_saklayici** sinyali ve 22.satırda 4 bitlik **v_saklayici** değişkeni tanımlanmaktadır. 19. satırda **out_cikis** çıkış portuna **r_saklayici** sinyali atanmaktadır. 21. satırda tanımlanan söz dizimi ile **process**'in **in_clk, in_rst** ve **in_giris** giriş portlarında meydana gelen değişiklikler ile aktif olacağı belirtilmektedir.

process içerisinde yapılan tanımlamardan da görüleceği üzere saklayıcı tasarımında eş zamanlı olmaya reset kullanılmıştır. 24. satırda **in_rst** giriş portu **'1'** değerini aldığında **in_giris** giriş portunun aldığı değerler farketmeksiniz **v_saklayi** değişkeninin tüm bitlerine **'0'** değeri atanmaktadır. **in_rst** girişinin diğer durumlarında ise **in_clk** giriş portunun yükselen kenarı ile birlikte 27. satırda tanımlanan koşul ifadesi ile **in_giris** giriş port değeri **v_saklayici** değişkeninin en anlamlı bitine atanmaktadır. **v_saklayici** sinyalinin 3. biti 2.bite, 2.biti 1.bite ve 1. bit en anlamsız bite atanmtakdır.

34. satırda **process**'in sonunda **v_saklayici** değişkeninin aldığı değer **r_saklayici** değişkenine atanarak **process** dışına aktarılmaktadır. **variable** veri nesnesi tüm atama işlemleri o anda yapıldığından dolayı **in_giris** portunun değeri **v_saklayici** değişkeninin tüm bitlerine atanmaktadır. Şekil 9-13'den de görüleceği üzere **in_giris** giriş portu değerinin **'1'** olduğu zamanlarda **r_saklayici** sinyali ve **out_cikis** çıkış portu **"1111"** değerini almaktadır. Aynı şekilde **in_giris** giriş portu değerinin **'0'** olduğu zamanlarda **r_saklayici** sinyali ve **out_cikis** çıkış portu **"0000"** değerini almaktadır.

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3.
4. entity kaydirmali_saklayici is
5.   Port (
6.     in_clk : in std_logic;
7.     in_rst : in std_logic;
8.     in_giris : in std_logic;
9.     out_cikis : out std_logic_vector(3 downto 0)
10.  );
11.end kaydirmali_saklayici;
12.
13.architecture Behavioral of kaydirmali_saklayici is
```

```
14.
15.  signal r_saklayici : std_logic_vector(3 downto 0) := (others => '0');
16.
17.begin
18.
19.  out_cikis <= r_saklayici;
20.
21.  process(in_clk, in_rst, in_giris)
22.    variable v_saklayici : std_logic_vector(3 downto 0) := (others =>
   '0');
23.  begin
24.    if in_rst = '1' then
25.      v_saklayici := (others => '0');
26.
27.    elsif rising_edge(in_clk) then
28.      v_saklayici(3) := in_giris ;
29.      v_saklayici(2) := v_saklayici(3) ;
30.      v_saklayici(1) := v_saklayici(2) ;
31.      v_saklayici(0) := v_saklayici(1) ;
32.
33.    end if;
34.    r_saklayici <= v_saklayici;
35.  end process;
36.
37.end Behavioral;
```



Şekil 9-13 variable veri nesnesi kullanılan kaydirmali_saklayici varlığı benzetim çıktısı

Şekil 9-13'tende görüleceği üzere yukarıda verilen **kaydirmali_saklayici.vhd** VHDL kodu kaydırmalı saklayıcı işlevini yerine getirememektedir. Kodun kaydırmalı saklayıcı işlevini yapabilmesi için 28-31 nolu satırlar aşağıda verilen kod satırı ile değiştirilmelidir.

```
28.v_saklayici(0) := v_saklayici(1) ;
29.v_saklayici(1) := v_saklayici(2) ;
30.v_saklayici(2) := v_saklayici(3) ;
31.v_saklayici(3) := in_giris ;
```

Örnek 9.6.3: Aşağıda kaydırma yönünün tayin edilebildiği **n_bit** bitlik kaydırmalı saklayıcı devresinin gerçekleştirildiği **generic_kaydirmali_saklayici.vhd** VHDL kodu verilmiştir. **generic_kaydirmali_saklayici** varlığımıza ilişkin generic bildirimleri 5-7. satırlarda, port bildirimleri 8-14. satırları arasında yapılmaktıdır.

Kaydırmalı saklayıcı tasarımında 1 bitlik saat darbesi giriş portu, 1 bitlik reset giriş portu, 1 bitlik kaydırma yönünün tayin edildiği giriş portu, 1 bitlik data giriş portu ve **n_bit** bitlik çıkış portu mevcuttur.

20-31. satırlar arasında sola kaydırma işleminin yapıldığı **f_sola_kaydir** fonksiyonu tanımlanmıştır. 33-44. satırlar arasında sağa kaydırma işleminin yapıldığı **f_saga_kaydir** fonksiyonu tanımlanmıştır.

56. satırta tanımlı koşul ifadesinin gerçeklenmesi durumunda **r_saklayici** sinyalinin tüm bitleri sola kaydırılmakta ve en anlamsız bitine **in_giris** giriş portu değeri yazılmaktadır. 58. satırta tanımlı koşul ifadesinin gerçeklenmesi durumunda **r_saklayici** sinyalinin tüm bitleri sağa kaydırılmakta ve en anlamlı bitine **in_giris** giriş portu değeri yazılmaktadır.

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3.
4. entity generic_kaydirmali_saklayici is
5.   Generic(
6.     n_bit : integer := 4
7.   );
8.   Port (
9.     in_clk : in std_logic;
10.    in_rst : in std_logic;
11.    in_yon : in std_logic;
12.    in_giris : in std_logic;
13.    out_cikis : out std_logic_vector(n_bit - 1 downto 0)
14.  );
15.end generic_kaydirmali_saklayici;
16.
17.architecture Behavioral of generic_kaydirmali_saklayici is
18.  signal r_saklayici : std_logic_vector(n_bit - 1 downto 0) := (others
   => '0');
19.
20.  function  f_sola_kaydir(in_giris  :  std_logic;  in_saklayici  :
   std_logic_vector(n_bit - 1 downto 0) )
21.  return std_logic_vector is
22.    variable v_saklayici : std_logic_vector(n_bit - 1 downto 0);
23.  begin
24.    v_saklayici := in_saklayici;
25.    for n_i in n_bit - 2 downto 0 loop
26.      v_saklayici(n_i + 1) := v_saklayici(n_i);
27.    end loop;
28.    v_saklayici(0) := in_giris;
```

```
29.     return v_saklayici;
30.
31.   end f_sola_kaydir;
32.
33.   function   f_saga_kaydir(in_giris   :   std_logic;   in_saklayici   :
    std_logic_vector(n_bit - 1 downto 0) )
34.   return std_logic_vector is
35.     variable v_saklayici : std_logic_vector(n_bit - 1 downto 0);
36.   begin
37.     v_saklayici := in_saklayici;
38.     for n_i in  1 to n_bit - 1 loop
39.       v_saklayici(n_i - 1) := v_saklayici(n_i);
40.     end loop;
41.     v_saklayici(n_bit - 1) := in_giris;
42.     return v_saklayici;
43.
44.   end f_saga_kaydir;
45.begin
46.
47.   out_cikis <= r_saklayici;
48.
49.   process(in_clk, in_rst, in_giris)
50.   begin
51.     if in_rst = '1' then
52.       r_saklayici <= (others => '0');
53.
54.     elsif rising_edge(in_clk) then
55.
56.       if in_yon = '0' then
57.         r_saklayici <= f_sola_kaydir(in_giris, r_saklayici);
58.       elsif in_yon = '1' then
59.         r_saklayici <= f_saga_kaydir(in_giris, r_saklayici);
60.       end if;
61.     end if;
62.   end process;
63.
64.end Behavioral;
```

Aşağıda ise **kaydirmali_saklayici** varlığının benzetim yapılabilmesi için **tb_kaydirmali_saklayici.vhd** sınama kodu verilmiştir ve benzetim çıktısı Şekil 9-14'de gösterilmiştir. Şekil 9-14'de gösterilen benzetim sonucunda:

1. adım : **in_rst** giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **in_yon** giriş portunun **'0'** değerini alması ile 56. satırda tanımlı koşul ifadesine bağlı 57. satırda tanımlı **f_sola_kaydir** fonksiyon işlemi gerçekleştirilerilmektedir ve fonksiyonun döndürdüğü değer **r_saklayici** sinyaline atanmaktadır. **in_giris** giriş portu değeri (**'1'**) **r_saklayici** sinyalinin en anlamsız bitine atanmaktadır ve diğer bitlerin sırasıyla atama işlemleri yapılmaktadır. Atama işlemlerinin sonucunda **r_saklayici** sinyalinin değeri **"0001"** olmaktadır. **r_saklayici** değerinin **"0001"** olması ile **out_cikis** çıkış portuna **"0001"** değeri atanmaktadır.

2. adım : **in_rst** giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **in_yon** giriş portunun **'0'** değerini alması ile 56. satırda tanımlı koşul ifadesine bağlı 57. satırda tanımlı **f_sola_kaydir** fonksiyon işlemi gerçekleştirilerilmektedir ve fonksiyonun döndürdüğü değer **r_saklayici** sinyaline atanmaktadır. **in_giris** giriş portu değeri (**'0'**) **r_saklayici** sinyalinin en anlamsız bitine atanmaktadır ve diğer bitlerin sırasıyla atama işlemleri yapılmaktadır. Atama işlemlerinin sonucunda **r_saklayici** sinyalinin değeri **"0010"** olmaktadır. **r_saklayici** değerinin **"0010"** olması ile **out_cikis** çıkış portuna **"0010"** değeri atanmaktadır.

3. adım : **in_rst** giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **in_yon** giriş portunun **'0'** değerini alması ile 56. satırda tanımlı koşul ifadesine bağlı 57. satırda tanımlı **f_sola_kaydir** fonksiyon işlemi gerçekleştirilerilmektedir ve fonksiyonun döndürdüğü değer **r_saklayici** sinyaline atanmaktadır. **in_giris** giriş portu değeri (**'1'**) **r_saklayici** sinyalinin en anlamsız bitine atanmaktadır ve diğer bitlerin sırasıyla atama işlemleri yapılmaktadır. Atama işlemlerinin sonucunda **r_saklayici** sinyalinin değeri **"0101"** olmaktadır. **r_saklayici** değerinin **"0101"** olması ile **out_cikis** çıkış portuna **"0101"** değeri atanmaktadır.

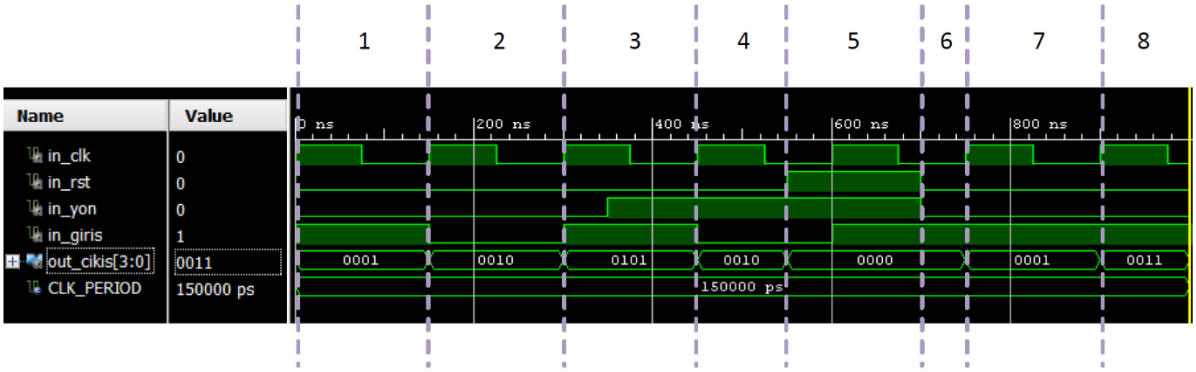
4. adım : **in_rst** giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **in_yon** giriş portunun **'1'** değerini alması ile 58. satırda tanımlı koşul ifadesine bağlı 59. satırda tanımlı **f_saga_kaydir** fonksiyon işlemi gerçekleştirilerilmektedir ve fonksiyonun döndürdüğü değer **r_saklayici** sinyaline atanmaktadır. **in_giris** giriş portu değeri (**'0'**) **r_saklayici** sinyalinin en anlamlı bitine atanmaktadır ve diğer bitlerin sırasıyla atama işlemleri yapılmaktadır. Atama işlemlerinin sonucunda **r_saklayici** sinyalinin değeri **"0010"** olmaktadır. **r_saklayici** değerinin **"0010"** olması ile **out_cikis** çıkış portuna **"0010"** değeri atanmaktadır.

5. adım : **in_rst** giriş portunun değeri **'1'** olması nedeniyle **r_saklayici** değerine **"0000"** atanmaktadır. **r_saklayici** değerinin **"0000"** olması ile **out_cikis** çıkış portuna **"0000"** değeri atanmaktadır.

6. adım : **in_rst** giriş portunun değeri **'0'** olmasına rağmen **in_clk** giriş portunda yükselen kenar meydan gelmediği için **r_saklayici** sinyalinin değerinde değişiklik olmamıştır.

7. adım : **in_rst** giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **in_yon** giriş portunun **'0'** değerini alması ile 56. satırda tanımlı koşul ifadesine bağlı 57. satırda tanımlı **f_sola_kaydir** fonksiyon işlemi gerçekleştirilerilmektedir ve fonksiyonun döndürdüğü değer **r_saklayici** sinyaline atanmaktadır. **in_giris** giriş portu değeri (**'1'**) **r_saklayici** sinyalinin en anlamsız bitine atanmaktadır ve diğer bitlerin sırasıyla atama işlemleri yapılmaktadır. Atama işlemlerinin sonucunda **r_saklayici** sinyalinin değeri **"0001"** olmaktadır. **r_saklayici** değerinin **"0001"** olması ile **out_cikis** çıkış portuna **"0001"** değeri atanmaktadır.

8. adım : **in_rst** giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **in_yon** giriş portunun **'0'** değerini alması ile 56. satırda tanımlı koşul ifadesine bağlı 57. satırda tanımlı **f_sola_kaydir** fonksiyon işlemi gerçekleştirilerilmektedir ve fonksiyonun döndürdüğü değer **r_saklayici** sinyaline atanmaktadır. **in_giris** giriş portu değeri (**'1'**) **r_saklayici** sinyalinin en anlamsız bitine atanmaktadır ve diğer bitlerin sırasıyla atama işlemleri yapılmaktadır. Atama işlemlerinin sonucunda **r_saklayici** sinyalinin değeri **"0011"** olmaktadır. **r_saklayici** değerinin **"0011"** olması ile **out_cikis** çıkış portuna **"0011"** değeri atanmaktadır.



Şekil 9-14 generic_kaydirmali_saklayici varlığı benzetim çıktısı

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3.
4. entity tb_generic_kaydirmali_saklayici is
5. end tb_generic_kaydirmali_saklayici;
6.
7. architecture Behavioral of tb_generic_kaydirmali_saklayici is
8.
9.   component generic_kaydirmali_saklayici
10.  Port (
11.    in_clk : in std_logic;
12.    in_rst : in std_logic;
13.    in_yon : in std_logic;
14.    in_giris : in std_logic;
15.    out_cikis : out std_logic_vector(3 downto 0)    );
16.  end component;
17.
18.  constant CLK_PERIOD : time := 150 ns;
19.  signal in_clk : std_logic := '0';
20.  signal in_rst : std_logic := '0';
21.  signal in_yon : std_logic := '0';
22.  signal in_giris : std_logic := '0';
23.  signal out_cikis : std_logic_vector(3 downto 0) := (others => '0');
24.
25.begin
26.
27.  process
28.  begin
```

```
29.    in_clk <= '1';    wait for CLK_PERIOD / 2;
30.    in_clk <= '0';    wait for CLK_PERIOD / 2;
31.  end process;
32.
33.  process
34.  begin
35.    in_rst <= '0'; wait for 550 ns;
36.    in_rst <= '1'; wait for 150 ns;
37.    in_rst <= '0'; wait for 300 ns;
38.  end process;
39.
40.  process
41.  begin
42.    in_yon <= '0'; wait for 350 ns;
43.    in_yon <= '1'; wait for 350 ns;
44.    in_yon <= '0'; wait for 300 ns;
45.  end process;
46.
47.  process
48.  begin
49.    in_giris <= '1'; wait for CLK_PERIOD;
50.    in_giris <= '0'; wait for CLK_PERIOD;
51.    in_giris <= '1'; wait for CLK_PERIOD;
52.    in_giris <= '0'; wait for CLK_PERIOD;
53.    in_giris <= '1'; wait for 3 * CLK_PERIOD;
54.    in_giris <= '0'; wait for CLK_PERIOD;
55.  end process;
56.
57.  generic_kaydirmali_saklayic_map : generic_kaydirmali_saklayici
58.  port map(
59.    in_clk => in_clk,
60.    in_rst => in_rst,
61.    in_yon => in_yon,
62.    in_giris => in_giris,
63.    out_cikis => out_cikis
64.  );
65.
66.end Behavioral;
```

9.7. Sayaçlar

VHDL ile tasasrım yapılırken sıklıkla kullanılan bileşenlerden biri de sayıcılardır. Sayıcılar gecikme yapmak, frekans bölmek v.b. pek çok amaçla kullanılabilen elemanlardandır. Bu uygulamalı örnekte ilk olarak 4 Bit uzunlukta basit bir sayıcı tasarımı yapılmış olup ardından aynı sayıcı **generic** hale getirilmiştir.

Sayıcı tasarlarken sayı aralığına bağlı olarak değişen bit uzunluğuna dikkat edilmelidir. Sayıcının alabileceği azami değer ve bit uzunluğu arasında aşağıda verilen matematiksel bağıntı mevcuttur, `'n'` toplam bit uzunluğu olmak üzere:

$$Sayıcının\ Azami\ Değeri = 2^n - 1$$

Örnek 9.7.1: Aşağıda 4 bitlik sayaç devresinin gerçekleştirildiği **sayac_4_bit.vhd** VHDL kodu verilmiştir. **sayac_4_bit** varlığımıza ilişkin port bildirimleri 6-10. satırlar arasında yapılmıştır. Sayaç tasarımında 1 bitlik saat darbesi giriş portu, 1 bitlik reset giriş portu ve 4 bitlik çıkış portu mevcuttur. 15. Satırda tanımlanan 4 bitlik **r_sayac** sinyali 19. satırda **out_cikis** çıkış portuna atanmaktadır. 21. satırda tanımlanan söz dizimi ile **process**'in **in_clk** ve **in_rst** giriş portlarında meydana gelen değişiklikler ile aktif olacağı belirtilmektedir.

process içerisinde yapılan tanımlamardan da görüleceği üzere saklayıcı tasarımında eş zamanlı olmaya reset kullanılmıştır. 23. satırda **in_rst** giriş portu **'1'** değerini aldığında **r_sayac** sinyalinin tüm bitlerine **'0'** değeri atanmaktadır. Bu durumda **out_cikis** çıkış portunun tüm bitleride **'0'** olmaktadır. **in_rst** girişinin diğer durumlarında ise **in_clk** giriş portunun yükselen kenarı ile birlikte **r_sayac** sinyalinin değeri bir artırılmaktadır. **in_clk** giriş sinyalinin yükselen kenarının meydana gelmemesi durumunda ise **r_sayac** sinyali bir önceki değerini korumaktadır.

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3. use IEEE.STD_LOGIC_UNSIGNED.ALL;
4.
5. entity sayac_4_bit is
6.   Port (
7.     in_clk : in std_logic;
8.     in_rst : in std_logic;
9.     out_cikis : out std_logic_vector(3 downto 0)
10.  );
11.end sayac_4_bit;
12.
13.architecture Behavioral of sayac_4_bit is
14.
15.  signal r_sayac : std_logic_vector(3 downto 0) := (others => '0');
16.
17.begin
18.  out_cikis <= r_sayac;
19.
20.  process(in_clk, in_rst)
21.  begin
```

```
22.     if in_rst = '1' then
23.       r_sayac <= (others => '0');
24.     elsif rising_edge(in_clk) then
25.       r_sayac <= r_sayac + 1;
26.     end if;
27.   end process;
28.
29.end Behavioral;
```

Aşağıda ise **sayac_4_bit** varlığının benzetim yapılabilmesi için **tb_sayac_4_bit.vhd** sınama kodu verilmiştir ve benzetim çıktısı Şekil 9-15'de gösterilmiştir. Şekil 9-15'de gösterilen benzetim sonucunda:

1. adım : in_rst giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **r_sayac** sinyalinin değeri bir artırılmaktadır ve değeri **"0001"** olmaktadır. **r_sayac** sinyalinin değerinin **"0001"** olması ile **out_cikis** çıkış portuna **"0001"** değeri atanmaktadır.

2. adım : in_rst giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **r_sayac** sinyalinin değeri bir artırılmaktadır ve değeri **"0010"** olmaktadır. **r_sayac** sinyalinin değerinin **"0010"** olması ile **out_cikis** çıkış portuna **"0010"** değeri atanmaktadır.

3. adım : in_rst giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **r_sayac** sinyalinin değeri bir artırılmaktadır ve değeri **"0011"** olmaktadır. **r_sayac** sinyalinin değerinin **"0011"** olması ile **out_cikis** çıkış portuna **"0011"** değeri atanmaktadır.

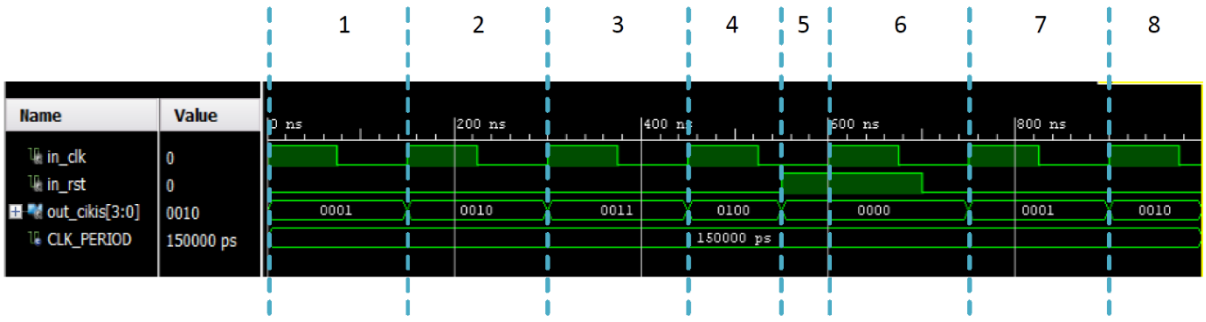
4. adım : in_rst giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **r_sayac** sinyalinin değeri bir artırılmaktadır ve değeri **"0100"** olmaktadır. **r_sayac** sinyalinin değerinin **"0100"** olması ile **out_cikis** çıkış portuna **"0100"** değeri atanmaktadır.

5. adım : in_rst giriş portunun değeri **'1'** olması nedeniyle **r_sayac** sinyalinin değerine **"0000"** atanmaktadır. **r_sayac** değerinin **"0000"** olması ile **out_cikis** değerine **"0000"** değeri atanmaktadır.

6. adım : in_rst giriş portunun değeri **'0'** olmasına rağmen **in_clk** giriş portunda yükselen kenar meydan gelmediği için **r_sayac** sinyalinin değerinde değişiklik olmamıştır.

7. adım : in_rst giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **r_sayac** sinyalinin değeri bir artırılmaktadır ve değeri **"0001"** olmaktadır. **r_sayac** sinyalinin değerinin **"0001"** olması ile **out_cikis** çıkış portuna **"0001"** değeri atanmaktadır.

8. adım : in_rst giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **r_sayac** sinyalinin değeri bir artırılmaktadır ve değeri **"0010"** olmaktadır. **r_sayac** sinyalinin değerinin **"0010"** olması ile **out_cikis** çıkış portuna **"0010"** değeri atanmaktadır.



Şekil 9-15 sayac_4_bit varlığı benzetim çıktısı

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3.
4. entity tb_sayac_4_bit is
5. end tb_sayac_4_bit;
6.
7. architecture Behavioral of tb_sayac_4_bit is
8.   component sayac_4_bit
9.   Port (
10.    in_clk : in std_logic;
11.    in_rst : in std_logic;
12.    out_cikis : out std_logic_vector(3 downto 0)
13.  );
14.  end component;
15.
16.  constant CLK_PERIOD : time := 150 ns;
17.  signal in_clk : std_logic := '0';
18.  signal in_rst : std_logic := '0';
19.  signal out_cikis : std_logic_vector(3 downto 0) := (others => '0');
20.
21.begin
22.
23.  process
24.  begin
25.    in_clk <= '1';
26.    wait for CLK_PERIOD / 2;
27.    in_clk <= '0';
28.    wait for CLK_PERIOD / 2;
29.  end process;
30.
31.  process
32.  begin
33.    in_rst <= '0'; wait for 550 ns;
34.    in_rst <= '1'; wait for 150 ns;
35.    in_rst <= '0'; wait for 300 ns;
36.  end process;
37.
38.  sayac_4_bit_map : sayac_4_bit
39.  port map(
40.    in_clk => in_clk,
```

```
41.     in_rst => in_rst,
42.     out_cikis => out_cikis
43.   );
44.end Behavioral;
```

Örnek 9.7.2: Aşağıda sayacın artan veya azalan durumunun tayin edilebildiği **n_bit** bitlik sayaç devresinin gerçekleştirildiği **generic_sayac.vhd** VHDL kodu verilmiştir. **generic_sayac** varlığımıza ilişkin generic bildirimleri 6-8. satırlarda, port bildirimleri 9-14. satırları arasında yapılmaktıdır.

Sayaç tasarımında 1 bitlik saat darbesi giriş portu, 1 bitlik reset giriş portu, 1 bitlik sayacın artması veya azalması durumunun tayin edildiği giriş portu ve **n_bit** bitlik çıkış portu mevcuttur. 25. satırda tanımlanan söz dizimi ile **process**'in **in_clk** ve **in_rst** giriş portlarında meydana gelen değişiklikler ile aktif olacağı belirtilmektedir.

process içerisinde yapılan tanımlamardan da görüleceği üzere saklayıcı tasarımında eş zamanlı olmaya reset kullanılmıştır. 27. satırda **in_rst** giriş portu **'1'** değerini aldığında **r_sayac** sinyalinin tüm bitlerine **'0'** değeri atanmaktadır. Bu durumda **out_cikis** çıkış portunun tüm bitleride **'0'** olmaktadır. **in_rst** girişinin diğer durumlarında ise **in_clk** giriş portunun yükselen kenarı ile birlikte **in_say** giriş portu değerinin **'0'** olması durumunda **r_sayac** sinyalinin değeri bir artırılmaktadır. **in_clk** giriş portunun yükselen kenarı ile birlikte **in_say** giriş portu değerinin **'1'** olması durumunda **r_sayac** sinyalinin değeri bir azaltılmaktadır.

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3. use IEEE.STD_LOGIC_UNSIGNED.ALL;
4.
5. entity generic_sayac is
6.   Generic(
7.     n_bit : integer := 4
8.   );
9.   Port (
10.    in_clk : in std_logic;
11.    in_rst : in std_logic;
12.    in_say : in std_logic;
13.    out_cikis : out std_logic_vector(n_bit - 1 downto 0)
14.  );
15.end generic_sayac;
16.
17.architecture Behavioral of generic_sayac is
18.
19.  signal r_sayac : std_logic_vector(n_bit - 1 downto 0) := (others =>
   '0');
20.
21.begin
22.
23.  out_cikis <= r_sayac;
```

```
24.
25.   process(in_clk, in_rst)
26.   begin
27.     if in_rst = '1' then
28.       r_sayac <= (others => '0');
29.
30.     elsif rising_edge(in_clk) then
31.       if in_say = '0' then
32.         r_sayac <= r_sayac + 1;
33.       elsif in_say = '1' then
34.         r_sayac <= r_sayac - 1;
35.       end if;
36.     end if;
37.   end process;
38.
39. end Behavioral;
```

Aşağıda ise **generic_sayac** varlığının benzetim yapılabilmesi için **tb_generic_sayac.vhd** sınama kodu verilmiştir ve benzetim çıktısı Şekil 9-16'de gösterilmiştir. Şekil 9-16'de gösterilen benzetim sonucunda:

1. adım : in_rst giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **in_say** giriş portunun **'0'** değerini alması ile **r_sayac** sinyalinin değeri bir artırılmaktadır ve değeri **"0001"** olmaktadır. **r_sayac** değerinin **"0001"** olması ile **out_cikis** çıkış portuna **"0001"** değeri atanmaktadır.

2. adım : in_rst giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **in_say** giriş portunun **'0'** değerini alması ile **r_sayac** sinyalinin değeri bir artırılmaktadır ve değeri **"0010"** olmaktadır. **r_sayac** değerinin **"0010"** olması ile **out_cikis** çıkış portuna **"0010"** değeri atanmaktadır.

3. adım : in_rst giriş portunun değeri **'1'** olması nedeniyle **r_sayac** değerine **"0000"** atanmaktadır. **r_sayac** değerinin **"0000"** olması ile **out_cikis** çıkış portuna **"0000"** değeri atanmaktadır.

4. adım : in_rst giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **in_say** giriş portunun **'0'** değerini alması ile **r_sayac** sinyalinin değeri bir artırılmaktadır ve değeri **"0001"** olmaktadır. **r_sayac** değerinin **"0001"** olması ile **out_cikis** çıkış portuna **"0001"** değeri atanmaktadır.

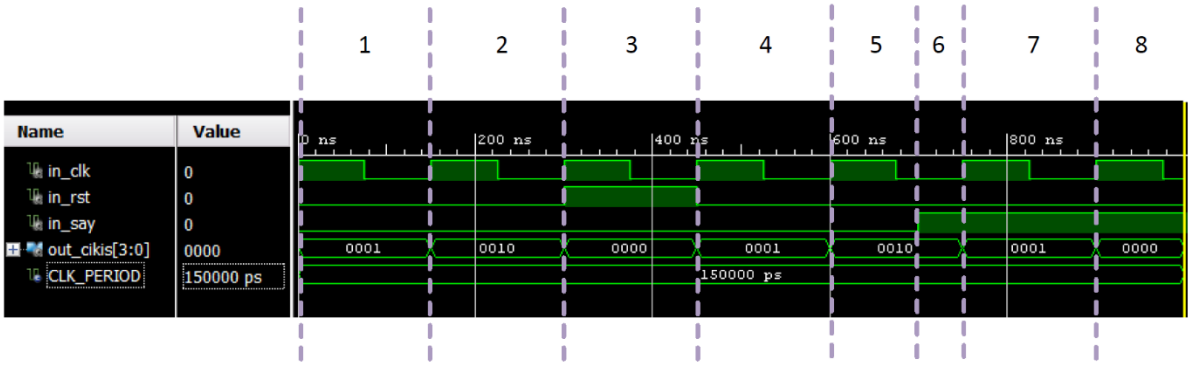
5. adım : in_rst giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **in_say** giriş portunun **'0'** değerini alması ile **r_sayac** sinyalinin değeri bir artırılmaktadır ve değeri **"0010"** olmaktadır. **r_sayac** değerinin **"0010"** olması ile **out_cikis** çıkış portuna **"0010"** değeri atanmaktadır.

6. adım : in_rst giriş portunun değerinin **'0'**ve **in_say** girişinin **'1'** olmasına rağmen **in_clk** giriş portunda yükselen kenar meydana gelmemsidnen dolayı **r_sayac** sinyalinin değeri değilmemiştir.

7. adım : in_rst giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **in_say** giriş portunun **'1'** değerini alması ile **r_sayac** sinyalinin değeri bir azaltılmaktadır ve değeri **"0001"** olmaktadır. **r_sayac** değerinin **"0001"** olması ile **out_cikis** çıkış portuna **"0001"** değeri atanmaktadır.

8. adım : in_rst giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **in_say** giriş portunun **'1'** değerini alması ile **r_sayac** sinyalinin değeri bir azaltılmaktadır ve değeri

"0000" olmaktadır. **r_sayac** değerinin **"0000"** olması ile **out_cikis** çıkış portuna **"0000"** değeri atanmaktadır.



Şekil 9-16 sayac_generic varlığı benzetim çıktısı

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3.
4. entity tb_generic_sayac is
5. end tb_generic_sayac;
6.
7. architecture Behavioral of tb_generic_sayac is
8.
9.   component generic_sayac
10.  Generic(
11.    n_bit : integer := 4
12.  );
13.  Port (
14.    in_clk : in std_logic;
15.    in_rst : in std_logic;
16.    in_say : in std_logic;
17.    out_cikis : out std_logic_vector(n_bit - 1 downto 0)
18.  );
19.  end component;
20.
21.  constant CLK_PERIOD : time := 150 ns;
22.  signal in_clk : std_logic := '0';
23.  signal in_rst : std_logic := '0';
24.  signal in_say : std_logic := '0';
25.  signal out_cikis : std_logic_vector(3 downto 0) := (others => '0');
26.
27.begin
```

```
28.
29.   process
30.   begin
31.     in_clk <= '1';
32.     wait for CLK_PERIOD / 2;
33.     in_clk <= '0';
34.     wait for CLK_PERIOD / 2;
35.   end process;
36.
37.   process
38.   begin
39.     in_rst <= '0'; wait for 300 ns;
40.     in_rst <= '1'; wait for 150 ns;
41.     in_rst <= '0'; wait for 550 ns;
42.   end process;
43.
44.   process
45.   begin
46.     in_say <= '0'; wait for 700 ns;
47.     in_say <= '1'; wait for 300 ns;
48.   end process;
49.
50.   generic_sayac_map : generic_sayac
51.   port map(
52.     in_clk => in_clk,
53.     in_rst => in_rst,
54.     in_say => in_say,
55.     out_cikis => out_cikis
56.   );
57.
58.end Behavioral;
```

9.8. Saat Frekans Bölücü

Zaman zaman tasarımlarımızda kullandığımız saat kaynağından daha yavaş çalışan, daha düşük frekanslı saat kaynaklarına ihtiyaç duyarız. Bu durumda kullanabileceğimiz çeşitli yöntemler mevcuttur. Örneğin FPGA üzreticileri tarafından sağlanan PLL (Phase Locked Loop), DCM (Digital Clock Manager) gibi hazır tasarım kütüphaneleri kullanılabilir. Bir diğer basit yaklaşım ise bir sayıcı kullanarak frekansı düşürmektir. Sayıcı kullanarak frekans bölmenin birden fazla yolu mevcuttur. Bu örnek uygulamada kullandığımız yol ise sayıcının her bir bitini saat kaynağı olarak kullanılması şeklindedir.

Bu kullanımın temelinde yatan fikir son derece basittir. Örneğin elimizde 3 bitlik bir sayıcı olduğunu kabul edelim. Bu sayıcının tüm durumları Tablo 9-5'de verildiği gibi olacaktır:

Tablo 9-5 3 itlik sayıcının tüm durumları.

Bit_2	Bit_1	Bit_0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Frekans bölme işleminin mantığını anlamak için şu kabulleri yapalım:

- Çalışma frekansımız $\boldsymbol{f}$ olsun.
- Sayıcımın yükselen kenar tetiklemeli çalışsın.

Bu durumda sayıcımızın bir çalışma döngüsünün uzunluğu (clock cycle), yani yükselen iki kenar arasındaki süremiz aşağıda verilen formül ile gösterilebilir. $\boldsymbol{t}$ süreyi temsil etmek üzere, frekans ve süre arasındaki ilişki:

$$t = \frac{1}{f}, \qquad f = \frac{1}{t}$$

Bu durumda sayıcının her sayma işlemi için $\boldsymbol{t}$ kadar süre gerekmektedir. Bu durumda Tablo 9-5'deki 0 ve 1 değişimlerine bakarsak şu yorumları yapabiliriz:

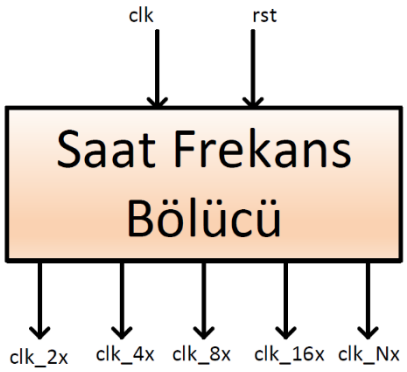
- Bit_0 için iki yükselen kenar arasındaki süre $\boldsymbol{2t}$ olmaktadır. '0' için $\boldsymbol{t}$, '1' için $\boldsymbol{t}$.
- Bit_1 için iki yükselen kenar arasındaki süre $\boldsymbol{4t}$ olmaktadır. '0' için $\boldsymbol{2t}$, '1' için $\boldsymbol{2t}$.
- Bit_3 için iki yükselen kenar arasındaki süre $\boldsymbol{8t}$ olmaktadır. '0' için $\boldsymbol{4t}$, '1' için $\boldsymbol{4t}$.

Yukarıda verilen formülü düzenleyip yeni frekans değerlerimizi hesaplarsak aşağıdaki bağıntıları elde ederiz:

- Bit_0 için : $f_{yeni} = \frac{1}{2t}$
- Bit_1 için : $f_{yeni} = \frac{1}{4t}$
- Bit_2 için : $f_{yeni} = \frac{1}{8t}$

Bu yöntemi genelleştirerek 2'nin kuvvetleri şeklinde istenilen frekans bölme değerleri elde edilebilmektedir. Eğer daha hassas bir şekilde çalışılmak isteniyor ve bu yöntemle istenilen frekans değeri elde edilemiyor ise FPGA üreticisi tarafından sağlanan PLL, DCM gibi hazır tasarım kütüphanelerinin kullanılması daha iyi olacaktır.

Örnek 9.8 : Şekil 9-17'de blok diyagramı verilen saat frekans bölücü devresi tasarımının gerçekleştirildiği **saat_frekans_bolucu.vhd** VHDL kodu aşağıda verilmiştir. Şekil 9-17'den de görüleceği üzere saat_frekans_bolucu varlığımız saat darbe frekansını 2, 4, 8, 16 ve generic olaran belirlenen N parametresine bölerek çıkış üretmektedir.



Şekil 9-17 sayac_generic varlığı benzetim çıktısı

sayac_generic varlığımıza ilişkin **generic** bildirimleri 6-8. satırlarda, port bildirimleri 9-17. satırları arasında yapılmaktıdır. **generic** atama işleminde **N** parametresi değerine 16 atanmaktadır. Bunun anlamı saat darbe frekansının 16'ya bölüneceğidir. Port tanımlama işlemlerindende görüleceği üzere **sayac_generic** varlığı **in_clk** ve **in_rst** giriş portlarına, **out_clk_2**, **out_clk_4**, **out_clk_8**, **out_clk_16** ve **out_clk_N** çıkış portlarına sahiptir. 21. satırda tanımlanan 4 bitlik **r_sayac** sinyali ile **out_clk_2**, **out_clk_4**, **out_clk_8** ve **out_clk_16** çıkış portlarının değerleri tanımlanmaktadır. 22. satırda tanımlan **r_sayac_N** sinyali ise de **out_clk_N** sinyalinin değerinin belirlenmesinde kullanılır.

25. satırda **r_sayac** değerinin 0. biti **out_clk_2** değerine atanmarak frekans değeri 2'ye bölünmektedir.

26. satırda **r_sayac** değerinin 1. biti **out_clk_4** değerine atanmarak frekans değeri 4'e bölünmektedir.

27. satırda **r_sayac** değerinin 2. biti **out_clk_8** değerine atanmarak frekans değeri 8'e bölünmektedir.

28. satırda **r_sayac** değerinin 3. biti **out_clk_16** değerine atanmarak frekans değeri 16'ya bölünmektedir.

29. satırda ise **r_sayac_N** değeri eğer generic parametre **N / 2** değerinden küçün ise **out_clk_N** çıkışına **'0'** değeri atanmaktadır. Aksi durumda ise **out_clk_N** çıkışına **'1'** değeri atanmaktadır.

38. satırda ise her saat darbesi yükselen kenarında **r_sayac** değeri bir artırılmkatadır. 39-43 satırları asında saat darbesi yükselen kenar darbesi ile birlikte **r_sayac_N** değeri **N - 1** değerine eşit ise **r_sayac_N** değeri sıfırlanmakatadır. Aksi durumlar da ise **r_sayac_N** değeri bir artırılmaktadır.

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3. use IEEE.STD_LOGIC_SIGNED.ALL;
4.
5. entity saat_frekans_bolucu is
6.   generic(
7.     N : integer := 16
8.   );
9.   Port (
10.    in_clk : in std_logic;
11.    in_rst : in std_logic;
12.    out_clk_2 : out std_logic;
13.    out_clk_4 : out std_logic;
14.    out_clk_8 : out std_logic;
15.    out_clk_16 : out std_logic;
16.    out_clk_N : out std_logic
17.  );
18.end saat_frekans_bolucu;
19.
20.architecture Behavioral of saat_frekans_bolucu is
21.  signal r_sayac : std_logic_vector(3 downto 0) := (others => '0');
22.  signal r_sayac_N : integer := 0;
23.begin
24.
25.  out_clk_2  <= r_sayac(0);
```

```
26.   out_clk_4  <= r_sayac(1);
27.   out_clk_8  <= r_sayac(2);
28.   out_clk_16 <= r_sayac(3);
29.   out_clk_N  <= '0' when r_sayac_N < N / 2 else '1';
30.
31.   process(in_clk, in_rst)
32.   begin
33.     if in_rst = '1' then
34.       r_sayac <= (others => '0');
35.       r_sayac_N <= 0;
36.
37.     elsif rising_edge(in_clk) then
38.       r_sayac <= r_sayac + 1;
39.       if r_sayac_N = N - 1 then
40.         r_sayac_N <= 0;
41.       else
42.         r_sayac_N <= r_sayac_N + 1;
43.       end if;
44.     end if;
45.   end process;
46. end Behavioral;
```

Aşağıda ise **sayac_generic** varlığının benzetim yapılabilmesi için **tb_sayac_generic.vhd** sınama kodu verilmiştir ve benzetim çıktıları Şekil 9-2 ve Şekil 9-19'de gösterilmiştir. Şekil 9-2'de gösterilen benzetim sonucunda:

1. adım: **in_rst** giriş portunun değeri **'1'** olması nedeniyle **r_sayac** ve **r_sayac_N** sinyallerinin değerleri sıfırlanmatakdır. Bu nedenle tüm çıkışlara **'0'** değeri atanmatakdır.

2. adım: **in_rst** giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **r_sayac** sinyalinin değeri bir artırılmaktadır ve değeri **"0001"** olmaktadır. Bu nedenle **out_clk_2 cikis** çıkış portu değeri **'1'** ve **out_clk_4** çıkış portu değeri **'0'** olmaktadır.

3. adım: **in_rst** giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **r_sayac** sinyalinin değeri bir artırılmaktadır ve değeri **"0010"** olmaktadır. Bu nedenle **out_clk_2 cikis** çıkış portu değeri **'0'** ve **out_clk_4** çıkış portu değeri **'1'** olmaktadır.

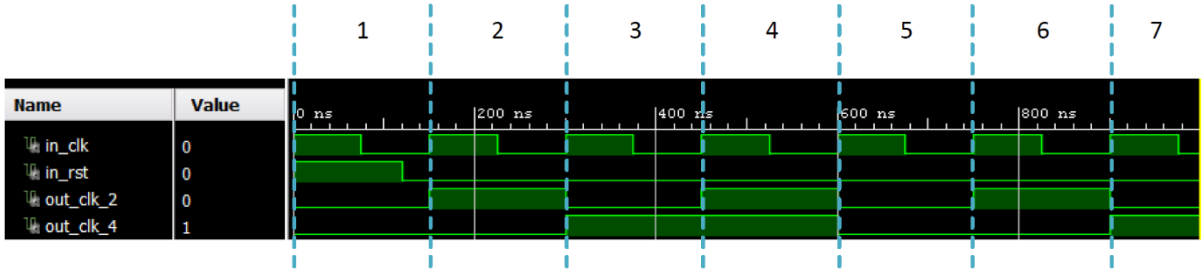
4. adım: **in_rst** giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **r_sayac** sinyalinin değeri bir artırılmaktadır ve değeri **"0011"** olmaktadır. Bu nedenle **out_clk_2 cikis** çıkış portu değeri **'1'** ve **out_clk_4** çıkış portu değeri **'1'** olmaktadır.

5. adım: **in_rst** giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **r_sayac** sinyalinin değeri bir artırılmaktadır ve değeri **"0100"** olmaktadır. Bu nedenle **out_clk_2 cikis** çıkış portu değeri **'0'** ve **out_clk_4** çıkış portu değeri **'0'** olmaktadır.

6. adım: **in_rst** giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **r_sayac** sinyalinin değeri bir artırılmaktadır ve değeri **"0101"** olmaktadır. Bu nedenle **out_clk_2 cikis** çıkış portu değeri **'1'** ve **out_clk_4** çıkış portu değeri **'0'** olmaktadır.

7. adım: **in_rst** giriş portunun değerinin **'0'** olması ve **in_clk** giriş portunun yükselen kenarı ile birlikte **r_sayac** sinyalinin değeri bir artırılmaktadır ve değeri **"0110"** olmaktadır. Bu nedenle **out_clk_2 cikis** çıkış portu değeri **'0'** ve **out_clk_4** çıkış portu değeri **'0'** olmaktadır.

Şekil 9-2'dende görüleceği üzere **out_clk_2** çıkış portunun 1 periyodluk zaman dilimi, **in_clk** giriş portunun 2 tam periyodluk zaman dilimine denk gelmektedir. Aynı şekilde **out_clk_4** çıkışının 1 periyodluk zaman dilimi, **in_clk** giriş portunun 4 tam periyodluk zaman dilimine denk eglmektedir.



Şekil 9-18 saat_frekans_bolucu varlığı benzetim çıktısı-1

Şekil 9-19'de gösterilen benzetim sonucunda:

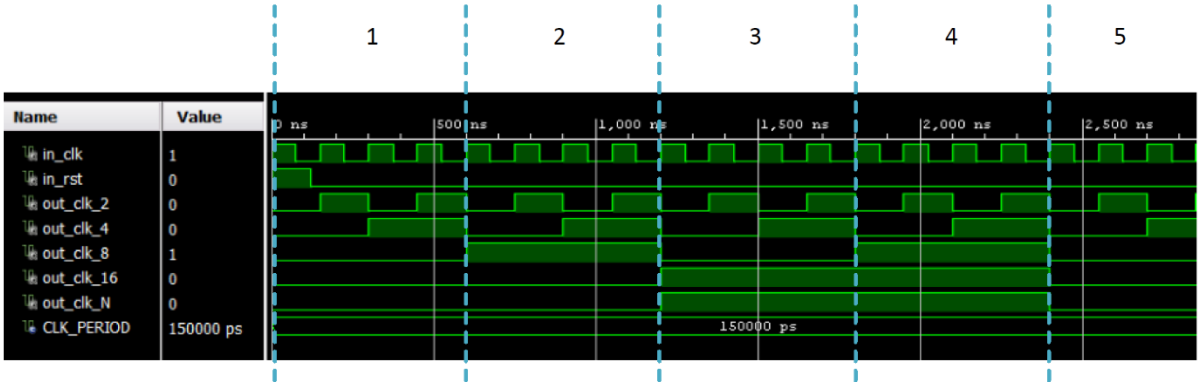
1. adım: **r_sayac** sinyalinin değerleri **"0000"** ile **"0011"** aralığında ve **r_sayac_N** sinyalinin değerleri **0** ile **3** aralığında olmaktadır. Bu nedenle **out_cikis_8** çıkış portu değeri **'0'**, **out_cikis_16** çıkış portu değeri **'0'** ve **out_cikis_N** çıkış değeri **'0'** olmaktadır.

2. adım: **r_sayac** sinyalinin değerleri **"0100"** ile **"0111"** aralığında ve **r_sayac_N** sinyalinin değerleri **4** ile **7** aralığında olmaktadır. Bu nedenle **out_cikis_8** çıkış portu değeri **'1'**, **out_cikis_16** çıkış portu değeri **'0'** ve **out_cikis_N** çıkış değeri **'0'** olmaktadır.

3. adım: **r_sayac** sinyalinin değerleri **"1000"** ile **"1011"** aralığında ve **r_sayac_N** sinyalinin değerleri **8** ile **11** aralığında olmaktadır. Bu nedenle **out_cikis_8** çıkış portu değeri **'0'**, **out_cikis_16** çıkış portu değeri **'1'** ve **out_cikis_N** çıkış değeri **'1'** olmaktadır.

4. adım: **r_sayac** sinyalinin değerleri **"1100"** ile **"1111"** aralığında ve **r_sayac_N** sinyalinin değerleri **12** ile **15** aralığında olmaktadır. Bu nedenle **out_cikis_8** çıkış portu değeri **'1'**, **out_cikis_16** çıkış portu değeri **'1'** ve **out_cikis_N** çıkış değeri **'1'** olmaktadır.

5. adım: **r_sayac** sinyalinin değerleri **"0000"** ile **"0011"** aralığında ve **r_sayac_N** sinyalinin değerleri **0** ile **3** aralığında olmaktadır. Bu nedenle **out_cikis_8** çıkış portu değeri **'0'**, **out_cikis_16** çıkış portu değeri **'0'** ve **out_cikis_N** çıkış değeri **'0'** olmaktadır.



Şekil 9-19 saat_frekans_bolucu varlığı benzetim çıktısı-2

```
1. library IEEE;
```

```
2. use IEEE.STD_LOGIC_1164.ALL;
3.
4. entity tb_saat_frekans_bolucu is
5. end tb_saat_frekans_bolucu;
6.
7. architecture Behavioral of tb_saat_frekans_bolucu is
8.
9.   component saat_frekans_bolucu
10.  generic(
11.    N : integer := 16
12.  );
13.  Port (
14.    in_clk : in std_logic;
15.    in_rst : in std_logic;
16.    out_clk_2 : out std_logic;
17.    out_clk_4 : out std_logic;
18.    out_clk_8 : out std_logic;
19.    out_clk_16 : out std_logic;
20.    out_clk_N : out std_logic
21.  );
22.  end component;
23.
24.  constant CLK_PERIOD : time := 150 ns;
25.  signal in_clk : std_logic := '0';
26.  signal in_rst : std_logic := '0';
27.  signal out_clk_2 : std_logic := '0';
28.  signal out_clk_4 : std_logic := '0';
29.  signal out_clk_8 : std_logic := '0';
30.  signal out_clk_16 : std_logic := '0';
31.  signal out_clk_N : std_logic := '0';
32.
33.begin
34.
35.  process
36.  begin
37.    in_clk <= '1';
38.    wait for CLK_PERIOD / 2;
39.    in_clk <= '0';
40.    wait for CLK_PERIOD / 2;
41.  end process;
```

```
42.   process
43.   begin
44.     in_rst <= '1'; wait for 120 ns;
45.     in_rst <= '0'; wait;
46.   end process;
47.
48.   saat_frekans_bolucu_map : saat_frekans_bolucu
49.   generic map( N => 16  )
50.   port map (
51.     in_clk => in_clk,
52.     in_rst => in_rst,
53.     out_clk_2 => out_clk_2,
54.     out_clk_4 => out_clk_4,
55.     out_clk_8 => out_clk_8,
56.     out_clk_16 => out_clk_16,
57.     out_clk_N => out_clk_N
58.   );
59.end Behavioral;
```

9.9. VHDL'de Dosya Okuma ve Yazma İşlemleri

Yaptığımız tasarımları bilgisayarda benzetim ortamında test ederken, tasarım tarafından işlenecek pek çok veri olabilir. Örneğin imge üzerinde çeşitli işlemler yapan bir tasarımı sınamak için imgenin de tasarlanan sisteme giriş olarak verilmesi gerekebilir. Böyle durumlar için VHDL bize dosyadan veri okuma ve yazma imkanı sunmaktadır.

VHDL ile dosyadan okuma ve yazma işlemi yapabilmek için kütüphane bildirim kısmına aşağıda verilen söz diziminin eklenmesi gerekmektedir. Bu söz dizimi ile birlikte dosya işlem komutları kullanılabilir hale gelmektedir.

```
use std.textio.ALL;
```

Tanımlanan veri yoluna ait dosyanın okuma modunda dosya değişkeninde açılmasına ilişkin söz dizimi aşağıda verilmiştir. Söz diziminde tanımlı **`text open`** söz dizimleri ile tanımlanan **`read_mode`** söz dizimi ile dosyanın okuma modunda açılıacağı tanımlanmaktadır.

```
file dosya : text open read_mode is VERI_YOLU;
```

Dosyadan okunacak olan satır, `variable` değişkeni olarak tanımlanmaktadır. **`satir`** değişkeni tanımlamaya ait söz dizimi aşağıda verilmiştir.

```
variable satir : line;
```

Dosyadan okuma işlemlerinde dosyanın sonuan gelinip gelinmediğinin kontrol işlemleri için aşağıdaki sözdizimi kullanılmaktadır.

```
if not endfile(dosya)  then
    ..
    ..
end if;
```

Dosyadan ilgili satırdan verilerin okunması işlemine ait söz dizimi aşağıda verilmiştir.

```
readline(dosya, satir);
read(satir, data);
```

Dosya üzerine yazım işlemi de okuma işlemine benzer şekilde yapılabilmektedir. Bunun için ilk olarak yazılacak dosyanın tanımlanması gerekmektedir. Aşağıda bu işlem için gerekli söz dizimi verilmiştir.

```
file dosya : text open write_mode is VERI_YOLU;
```

Dosya açma işlemi gerçekleştirdikten sonra yazılacak verilerin düzenlenmesi, dosyaya yazmaya hazır hale getirilmesi gerekmektedir. Bunun için gerekli söz dizimi aşağıda verilmiştir.

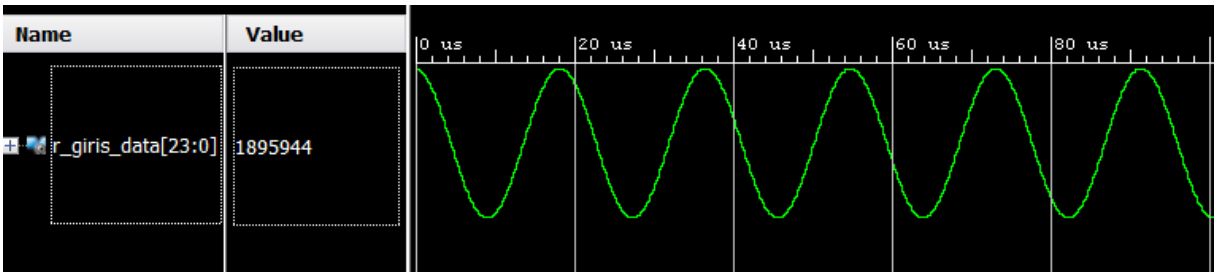
```
write(satir, data);
writeline(dosya, satir);
```

Örnek 9.9.1: Aşağıda sinüs örneklerinin var olduğu **sin.txt** dosyasında kayıtlı integer sayıların okunmasına ilişkin **dosya_okuma_integer.vhd** VHDL kodu verilmiştir. 4. satırda metin dosyasından okuma/yazma yapabilmek için kütüphane bildirimi yapılmıştır. Bu kod benzetim amacı ile kullanılacağından dolayı varlık içerisinde port tanımlaması yapılmamıştır.

12. satırda sinüs örneklerinin bulunduğu **sin.txt** dosyasına ait veri yolu tanımlama işlemi **string** tipinde **VERI_YOLU_OKUMA constant** veri nesnesi ile tanımlanmıştır. Okunan sinüs örnekerşnin yazılacağı dosya veri yolu 13. satırda tanımlanmıştır. 27. satırda **sin.txt** dosyası okuma modunda açılmıştır. 28. satırda **sin.txt** dosyası yazma modunda açılmıştır. 34. satırda dosyanın sonuna gelinip gelinmediğinin kotnrol işlemini yapan söz dizimi tanımlanmıştır. Dosyanın sonuna gelinmediği durumalrda ilgili satırda bulunan veriler dosyadan okunarak **data** değişkenine atanmaktadır. **integer** tipinde tanımlı **data** değişkeni 24 bit uzunluğunda tanımlı r**_giris_data** sinyaline tip dönüşümü yapılarak atanmaktadır. Aynı zamanda **data** değişkeninin sahip olduğu değer tekrar dosyaya yazılmaktadır. Dosyadan okuma işlemine ait benzetim çıktısı Şekil 9-17'de verilmiştir.

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3. use IEEE.STD_LOGIC_ARITH.ALL;
4. use std.textio.ALL;
5.
6. entity dosya_okuma_integer is
```

```
7. end dosya_okuma_integer;
8.
9. architecture Behavioral of dosya_okuma_integer is
10.
11.  constant CLK_PERIOD : time := 150 ns;
12.  constant VERI_YOLU_OKUMA : string := "C:\sin.txt";
13.  constant VERI_YOLU_YAZMA : string := "D:\sin.txt
14.  signal r_giris_data : std_logic_vector(23 downto 0) := (others=>
   '0');
15.  signal in_clk : std_logic := '0';
16.begin
17.
18.  process
19.  begin
20.    in_clk <= '1';
21.    wait for CLK_PERIOD / 2;
22.    in_clk <= '0';
23.    wait for CLK_PERIOD / 2;
24.  end process;
25.
26.  process(in_clk)
27.    file dosya_okuma: text open read_mode is VERI_YOLU_OKUMA;
28.    file dosya_yazma: text open write_mode is VERI_YOLU_YAZMA;
29.    variable satir_okuma : line;
30.    variable satir_yazma : line;
31.    variable data : integer;
32.  begin
33.    if rising_edge(in_clk) then
34.      if not endfile(dosya)  then
35.        readline(dosya_okuma, satir_okuma);
36.        read(satir_okuma, data);
37.        r_giris_data          <=          conv_std_logic_vector(data,
   r_giris_data'length);
38.        write(satir_yazma, data);
39.        writeline(dosya_yazma, satir_yazma);
40.      end if;
41.    end if;
42.  end process;
43.
44.end Behavioral;
```



Şekil 9-20 dosya_okuma_integer varlığı benzetim çıktısı

Sinüs örneklerinin bulunduğu **sin.txt** dosyasını oluşturmak için aşağıda verilen MATLAB kodu kullanılabilir.

```
clc, clear all, close all;

sin_file = fopen('C:\sin.txt', 'w');
f_s = 10000;
n_t = 1 / f_s : 1 / f_s : 2;

f_1 = 10;
A = 1 * sin( 2 * pi * n_t  * f_1 );
D = round(2^20 * (A + 1));
figure, plot(A);
figure, plot(D);

for n_i = 1 : length(D)
    fprintf(sin_file, '%d\n', D(n_i) );
end
```

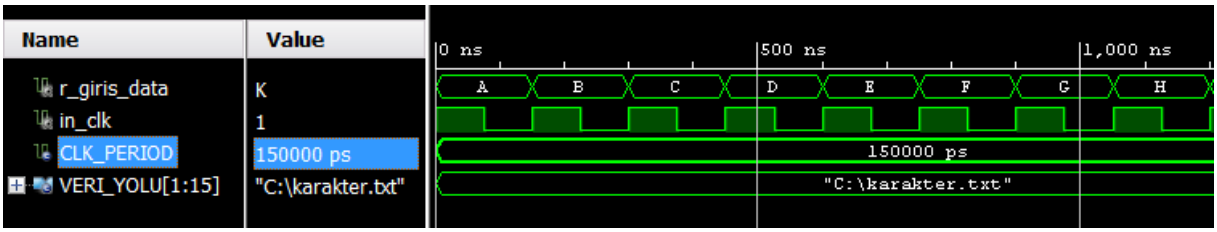
sin.txt dosyasına yazılan dataları okumak için aşağıda verilen MATLAB kodu kullanılabilir.

```
clc, clear all, close all;

sin_file = fopen('D:\sin.txt', 'r');
sin_okunan = fscanf(sin_file, '%d')';

figure, plot(sin_okunan);
```

Örnek 9.9.2: Aşağıda karakter örneklerinin var olduğu **karakter.txt** dosyasında kayıtlı karakterlerin okunmasına ilişkin **dosya_okuma_integer.vhd** VHDL kodu verilmiştir. 4. satırda metin dosyasından okuma/yazma yapabilmek için kütüphane bildirimi yapılmıştır. Bu kod benzetim amacı ile kullanılacağından dolayı varlık içerisinde port tanımlaması yapılmamıştır.

12. satırda karakter örneklerinin bulunduğu **karakter.txt** dosyasına ait veri yolu tanımlama işlemi **string** tipinde **VERI_YOLU constant** veri nesnesi tanımlanmıştır. 26. satırda **karakter.txt** dosyası okuma

modunda açılmıştır. 31. satırda dosyanın sonuna gelinip gelinmediğinin kotnrol işlemini yapan söz dizimi tanımlanmıştır. Dosyanın sonuna gelinmediği durumlarda ilgili satırda bulunan veriler dosyadan okunarak **data** değişkenine atanmaktadır. **character** tipinde tanımlı **data** değişkeni **character** tipinde tanımlı **r_giris_data** sinyaline atanmaktadır. Dosyadan okuma işlemine aiat benzetim çıktısı Şekil 9-21'de verilmiştir.



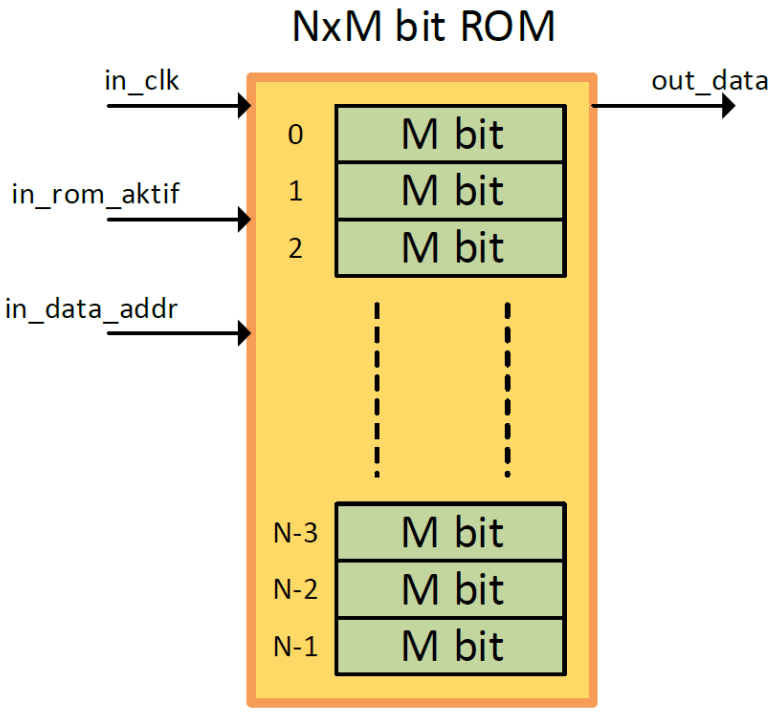
Şekil 9-21 dosya_okuma_karakter varlığı benzetim çıktısı

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3. use std.textio.ALL;
4.
5. entity dosya_okuma_karakter is
6. end dosya_okuma_karakter;
7.
8. architecture Behavioral of dosya_okuma_karakter is
9.
10.   constant CLK_PERIOD : time := 150 ns;
11.   constant VERI_YOLU : string := "C:\karakter.txt";
12.
13.   signal r_giris_data : character;
14.   signal in_clk : std_logic := '0';
15.
16.begin
17.
18.   process
19.   begin
20.     in_clk <= '1';
21.     wait for CLK_PERIOD / 2;
22.     in_clk <= '0';
23.     wait for CLK_PERIOD / 2;
24.   end process;
25.
26.   process(in_clk)
27.     file dosya : text open read_mode is VERI_YOLU;
28.     variable satir : line;
```

```
29.     variable data : character;
30.
31.   begin
32.     if rising_edge(in_clk) then
33.       if not endfile(dosya)  then
34.         readline(dosya, satir);
35.         read(satir, data);
36.       end if;
37.     end if;
38.     r_giris_data <= data;
39.   end process;
40.  end Behavioral;
```

9.10. VHDL'de ROM Bloğu oluşturmak

ROM (Read Only Memory – Sadece Okunabilir Hafıza) sadece okunabilen sayısal verilerin saklanması için kullanılan depolama birimidir. Şekil 9-22'de NxM bitlik ROM gösterimi verilmiştir. Şekil 9-22'den de görüleceği üzere **in_rom_aktif** giriş portunun aktif olması ile birlikte ile ROM'dan **in_data_addr** adresindeki data **out_data** çıkış portuna aktarılmaktadır.



Şekil 9-22 NxM bitlik ROM

Hafıza elemanları tasarlanırken genel olarak kullanılan bazı terimlere aşina olmak gerekmektedir. Bu bölümde verilen örneklerde geçen **VERI_UZUNLUGU** tanımlaması tasarladığımız hafıza biriminin veri yolu genişliğini bildirmektedir. Örneğin bu değer 8 ise tasarlanan hafıza elemanı her bir adreste 8 bit uzunluğunda veri saklayabiliyor demektir.

Hafıza elemanları ile ilgili bir diğer tanımlama ise **ROM_DERINLIGI**'dir. Bu tanımlama ile tasarladığımız hafıza elemanının kaç adet veri saklayacabileği belirtilmektedir. Bu değer aynı zaman doğrudan adresleme hattının uzunluğunun da belirlenmesini sağlamaktadır. Örneklemek gerekirse; 30 adet veri saklamak istersek **ROM_DERINLIGI** tanımlamasının değerinin 30 olması gerekmektedir. 30 adet veriyi adreslemek için gereken adres yolu genişliği ise $ceil(log_2^N) = ceil(log_2^{30}) \rightarrow n = 5$ hesaplamında gösterildiği şekilde 5 bit olacaktır. Bu işlemin gerçeklenmesi için kullanılcak olan **log2_int** fonksiyonu **ornekler_paket.vhd** paket dosyasında tanımlanmıştır. **Örnek 9.10.1** ve **Örnek 9.10.2**'de tanımlı **ornekler_paket.vhd** paket dosyası aşağıda verilmiştir.

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3.
4. package ornekler_paket is
5.   function log2_int(in_giris : integer) return integer;
6. end ornekler_paket;
7.
8. package body ornekler_paket is
9.
10.  function log2_int(in_giris : integer) return integer is
11.    variable sonuc : integer;
12.  begin
13.    for n_i in 0 to 31 loop
14.      if (in_giris <= (2 ** n_i)) then
15.        sonuc := n_i;
16.        exit;
17.      end if;
18.    end loop;
19.    return sonuc;
20.  end log2_int;
21.end package body;
```

Örnek 9.10.1: ROM datalarının değiştirlemez olmasından dolayı dataların **constant** veri nesnesinde tanımlandığı **rom.vhd** VHDL kodu aşağıda verilmiştir. **rom** varlığımıza ilişkin generic bildirimleri 7-10. satırlarda, port bildirimleri 11-16. satırları arasında yapılmaktıdır. ROM tasarımında 1 bitlik saat darbesi giriş portu, 1 bitlik ROM aktif giriş portu, generic **ROM_DERINLIGI** parametresine bağlı olarak hesaplanan adres giriş portu ve okununan adresindeki datanın ROM dışına aktarılması için **VERI_UZUNLUGU** uzunluğunda çıkış portu mevcuttur. ROM adres uzunluğunun belirlenmesi için kullanılacak olan **log2_int** fonskiyonu 4. satırda tanımlanan **ornekler_paket** paketi içerisinde tanımlanmıştır. 21. satırda **VERI_UZUNLUGU** genişliğine sahip **ROM_DERINLIGI** derinliğinde tip tanımlama işlemi yapılmıştır. 22. satırda ise ROM değerlerinin atama işlemleri yapılmaktadır.

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3. use IEEE.STD_LOGIC_UNSIGNED.ALL;
4. use work.ornekler_paket.all;
```

```
5.
6. entity rom is
7.   Generic(
8.     ROM_DERINLIGI : integer := 30;
9.     VERI_UZUNLUGU : integer := 4
10.  );
11.  Port (
12.    in_clk : in std_logic;
13.    in_rom_aktif : in std_logic;
14.    in_data_addr : in std_logic_vector(log2_int(ROM_DERINLIGI) - 1
   downto 0);
15.    out_data : out std_logic_vector(VERI_UZUNLUGU - 1 downto 0)
16.  );
17.end rom;
18.
19.architecture Behavioral of rom is
20.
21.  type t_ROM_DATA is array (0 to ROM_DERINLIGI - 1) of
   std_logic_vector(VERI_UZUNLUGU - 1 downto 0) ;
22.  constant r_ROM_DATA : t_ROM_DATA := ( X"0", X"3", X"6", X"9", X"C",
   X"F", X"C", X"9", X"6", X"3", X"0", X"3", X"6", X"9", X"C",
   X"F", X"C", X"9", X"6", X"3", X"0", X"3", X"6", X"9", X"C",
   X"F", X"C", X"9", X"6", X"3" );
23.
24.begin
25.
26.  process(in_clk)
27.  begin
28.    if rising_edge(in_clk) then
29.      if in_rom_aktif = '1' then
30.        out_data <= r_ROM_DATA(conv_integer(in_data_addr));
31.      end if;
32.    end if;
33.  end process;
34.end Behavioral;
```

Örnek 9.10.2: Aşağıda ROM datalarının dosyadan okuyan **rom_dosya.vhd** VHDL kodu verilmiştir. **rom_dosya** varlığımıza ilişkin generic bildirimleri 9-12. satırlarda, port bildirimleri 13-18. satırları arasında yapılmaktıdır. ROM tasarımında 1 bitlik saat darbesi giriş portu, 1 bitlik ROM aktif giriş portu, generic **ROM_DERINLIGI** parametresine bağlı olarak hesaplanan adres giriş portu ve okununan adresdeki datanın ROM dışına aktarılması için **VERI_UZUNLUGU** uzunluğunda çıkış portu mevcuttur. ROM adres uzunluğunun belirlenmesi için kullanılacak olan **log2_int** fonskiyonu 4. satırda tanımlanan **ornekler_paket** paketi içerisinde tanımlanmıştır. 24. satırda **VERI_UZUNLUGU** genişliğine sahip **ROM_DERINLIGI** derinliğinde tip

tanımlama işlemi yapılmıştır. 27-41. satırlar arasında tanımlı **ROM_DATA_YUKLE** procedure tanımlaması ile sinüs örneklerinin bulunduğu dosyadan datalar alınmaktadır. 45. satırda tanımlı söz dizimi ile procedure çağrılarak dosyadan okunan datalar ROM'a yüklenmektedir.

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3. use IEEE.STD_LOGIC_UNSIGNED.ALL;
4. use IEEE.STD_LOGIC_ARITH.ALL;
5. use std.textio.ALL;
6. use work.ornekler_paket.all;
7.
8. entity rom_dosya is
9.   Generic(
10.    ROM_DERINLIGI : integer := 500;
11.    VERI_UZUNLUGU : integer := 24
12.  );
13.  Port (
14.    in_clk : in std_logic;
15.    in_rom_aktif : in std_logic;
16.    in_data_addr  :  in  std_logic_vector(log2_int(ROM_DERINLIGI)  -  1
   downto 0);
17.    out_data : out std_logic_vector(VERI_UZUNLUGU - 1 downto 0)
18.  );
19.end rom_dosya;
20.
21.architecture Behavioral of rom_dosya is
22.
23.  constant VERI_YOLU : string := "C:\sin.txt";
24.  type   t_ROM_DATA   is   array   (0   to   ROM_DERINLIGI   -   1)   of
   std_logic_vector(VERI_UZUNLUGU - 1 downto 0) ;
25.  signal r_ROM_DATA : t_ROM_DATA;
26.
27.  procedure ROM_DATA_YUKLE(signal r_ROM_DATA : inout t_ROM_DATA) is
28.    file dosya : text open read_mode is VERI_YOLU;
29.    variable satir  :line;
30.    variable data : integer;
31.  begin
32.    for n_i in 0 to ROM_DERINLIGI - 1 loop
33.      if endfile(dosya) then
34.        exit;
35.      else
36.        readline(dosya, satir);
```

```
37.         read(satir, data);
38.         r_ROM_DATA(n_i) <= conv_std_logic_vector(data, VERI_UZUNLUGU);
39.       end if;
40.     end loop;
41.   end procedure;
42.
43.begin
44.
45.   ROM_DATA_YUKLE(r_ROM_DATA);
46.
47.   process(in_clk)
48.   begin
49.     if rising_edge(in_clk) then
50.       if in_rom_aktif = '1' then
51.         out_data <= r_ROM_DATA(conv_integer(in_data_addr));
52.       end if;
53.     end if;
54.   end process;
55.end Behavioral;
```

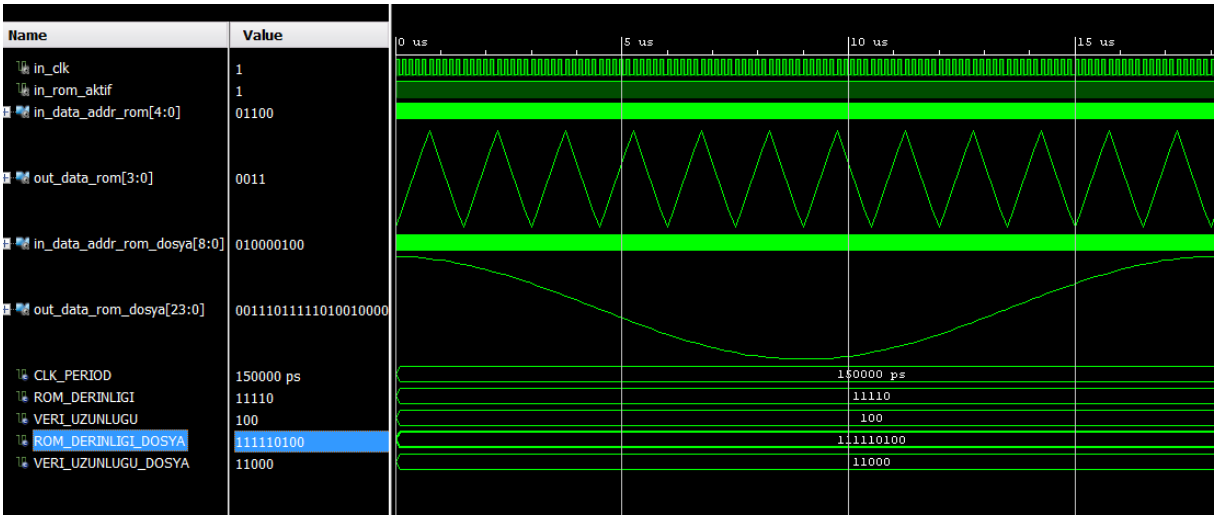
Örnek 9.10.1 ve **Örnek 9.10.2**'de verilen **rom** ve **rom_dosya** varlıklarının benzetiminin yapılabilmesi için aşağıda **tb_rom.vhd** VHDL sınama kodu verilmiştir. 10-21. satırlarda rom varlığına ilişkin component tanımlamaları yapılmıştır. 23-34. satırlarda **rom_dosya** varlığına ilişkin **component** tanımlamaları yapılmıştır. 37-38. satırlardaki yapılan sabit tanmlamalarında **rom** varlığına ait derinlik değeri **30** ve veri uzunluğu **4** olarak tanımlanmıştır. 39-40. satırlardaki yapılan sabit tanmlamalarında **rom_dosya** varlığına ait derinlik değeri **500** ve veri uzunluğu **24** olarak tanımlanmıştır. Şekil 9-23'de **rom** ve **rom_dosya** varlıklarının benzetim sonuçları verilmiştir.

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3. use IEEE.STD_LOGIC_UNSIGNED.ALL;
4. use work.ornekler_paket.all;
5.
6. entity tb_rom is
7. end tb_rom;
8.
9. architecture Behavioral of tb_rom is
10.   component rom
11.     Generic(
12.       ROM_DERINLIGI : integer := 500;
13.       VERI_UZUNLUGU : integer := 24
```

```
14.     );
15.     Port (
16.       in_clk : in std_logic;
17.       in_rom_aktif : in std_logic;
18.       in_data_addr : in std_logic_vector(log2_int(ROM_DERINLIGI) - 1
    downto 0);
19.       out_data : out std_logic_vector(VERI_UZUNLUGU - 1 downto 0)
20.     );
21.   end component;
22.
23.   component rom_dosya
24.     Generic(
25.       ROM_DERINLIGI : integer := 500;
26.       VERI_UZUNLUGU : integer := 24
27.     );
28.     Port (
29.       in_clk : in std_logic;
30.       in_rom_aktif : in std_logic;
31.       in_data_addr : in std_logic_vector(log2_int(ROM_DERINLIGI) - 1
    downto 0);
32.       out_data : out std_logic_vector(VERI_UZUNLUGU - 1 downto 0)
33.     );
34.   end component;
35.
36.   constant CLK_PERIOD : time := 150 ns;
37.   constant ROM_DERINLIGI : integer := 30;
38.   constant VERI_UZUNLUGU : integer := 4;
39.   constant ROM_DERINLIGI_DOSYA : integer := 500;
40.   constant VERI_UZUNLUGU_DOSYA : integer := 24;
41.
42.   signal in_clk : std_logic := '0';
43.   signal in_rom_aktif : std_logic := '0';
44.   signal in_data_addr_rom : std_logic_vector( log2_int(ROM_DERINLIGI)
    - 1 downto 0) := (others => '0');
45.   signal out_data_rom :  std_logic_vector(VERI_UZUNLUGU - 1 downto 0)
    := (others => '0');
46.   signal           in_data_addr_rom_dosya:          std_logic_vector(
    log2_int(ROM_DERINLIGI_DOSYA) - 1 downto 0) := (others => '0');
47.   signal out_data_rom_dosya :  std_logic_vector( VERI_UZUNLUGU_DOSYA -
    1 downto 0) := (others => '0');
48.
49.begin
```

```
50.
51.  process
52.  begin
53.    in_clk <= '1';
54.    wait for CLK_PERIOD / 2;
55.    in_clk <= '0';
56.    wait for CLK_PERIOD / 2;
57.  end process;
58.
59.  process(in_clk)
60.  begin
61.    if rising_edge(in_clk) then
62.      in_data_addr_rom <= in_data_addr_rom + 1;
63.      if in_data_addr_rom = ROM_DERINLIGI - 1 then
64.        in_data_addr_rom <= (others => '0');
65.      end if;
66.      in_data_addr_rom_dosya <= in_data_addr_rom_dosya + 1;
67.
68.      if in_data_addr_rom_dosya = ROM_DERINLIGI_DOSYA - 1 then
69.        in_data_addr_rom_dosya <= (others => '0');
70.      end if;
71.    end if;
72.  end process;
73.  in_rom_aktif <= '1';
74.  rom_map : rom generic map(
75.    ROM_DERINLIGI => ROM_DERINLIGI,
76.    VERI_UZUNLUGU => VERI_UZUNLUGU   )
77.  port map (
78.    in_clk => in_clk,
79.    in_rom_aktif => in_rom_aktif,
80.    in_data_addr => in_data_addr_rom,
81.    out_data => out_data_rom  );
82.
83.  rom_dosya_map : rom_dosya generic map(
84.    ROM_DERINLIGI => ROM_DERINLIGI_DOSYA,
85.    VERI_UZUNLUGU => VERI_UZUNLUGU_DOSYA  )
86.  port map (
87.    in_clk => in_clk,
88.    in_rom_aktif => in_rom_aktif,
89.    in_data_addr => in_data_addr_rom_dosya,
```

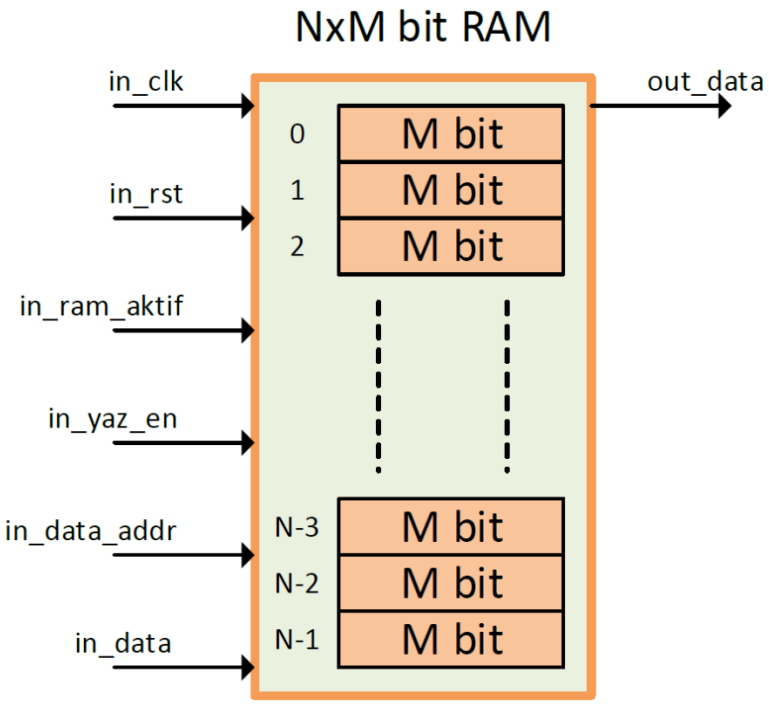
```
90.     out_data => out_data_rom_dosya);
91.
92.end Behavioral;
```



Şekil 9-23 rom ve rom_dosya varlıklarının benzetim çıktısı

9.11. VHDL'de RAM Bloğu oluşturmak

RAM'ler (Random Access Memory – Rastgele Erişimli Hafıza), ROM'lardan farklı olarak verilerin hem okunmasına hem de yazılmasına izin veren bir hafıza birimidir. Rastgele erişim kontrol devresi ile saklanmış verilere rastgele sırada direk erişim sağlanır. Şekil 9-24'de NxM bitlik RAM gösterimi bulunmaktadır. Şekil 9-24'den de görüleceği üzere **`in_ram_aktif`** giriş portunun aktif olması ile birlikte ile RAM'dan **`in_data_addr`** adresindeki data **`out_data`** çıkış portuna aktarılmaktadır. **`in_yaz_en`** giriş portunun aktif olması ile birlikte **`in_data`** giriş portundaki data **`in_data_addr`** adrsindeki yerine yazılmaktadır.



Şekil 9-24 NxM bitlik RAM

Hafıza elemanları tasarlanırken genel olarak kullanılan bazı terimlere aşina olmak gerekmektedir. Bu bölümde verilen örneklerde geçen **VERI_UZUNLUGU** tanımlaması tasarladığımız hafıza biriminin veri yolu genişliğini bildirmektedir. Örneğin bu değer 8 ise tasarlanan hafıza elemanı her bir adreste 8 bit uzunluğunda veri saklayabiliyor demektir.

Hafıza elemanları ile ilgili bir diğer tanımlama ise **ROM_DERINLIGI'**dir. Bu tanımlama ile tasarladığımız hafıza elemanının kaç adet veri saklayacabileği belirtilmektedir. Bu değer aynı zaman doğrudan adresleme hattının uzunluğunun da belirlenmesini sağlamaktadır. Örneklemek gerekirse; 32 adet veri saklamak istersek **ROM_DERINLIGI** tanımlamasının değerinin 32 olması gerekmektedir. 32 adet veriyi adreslemek için gereken adres yolu genişliği ise $2^n = 32 \rightarrow n = 5$ hesaplamında gösterildiği şekilde 5 bit olacaktır.

Örnek 9.11.1: Şekil 9-24'de gösterilen blok ram tasarımın yapıldığı **blok_ram.vhd** VHDL kodu aşağıda verilmiştir. **blok_ram** varlığımıza ilişkin generic bildirimleri 7-10. satırlarda, port bildirimleri 11-16. satırları arasında yapılmaktıdır.

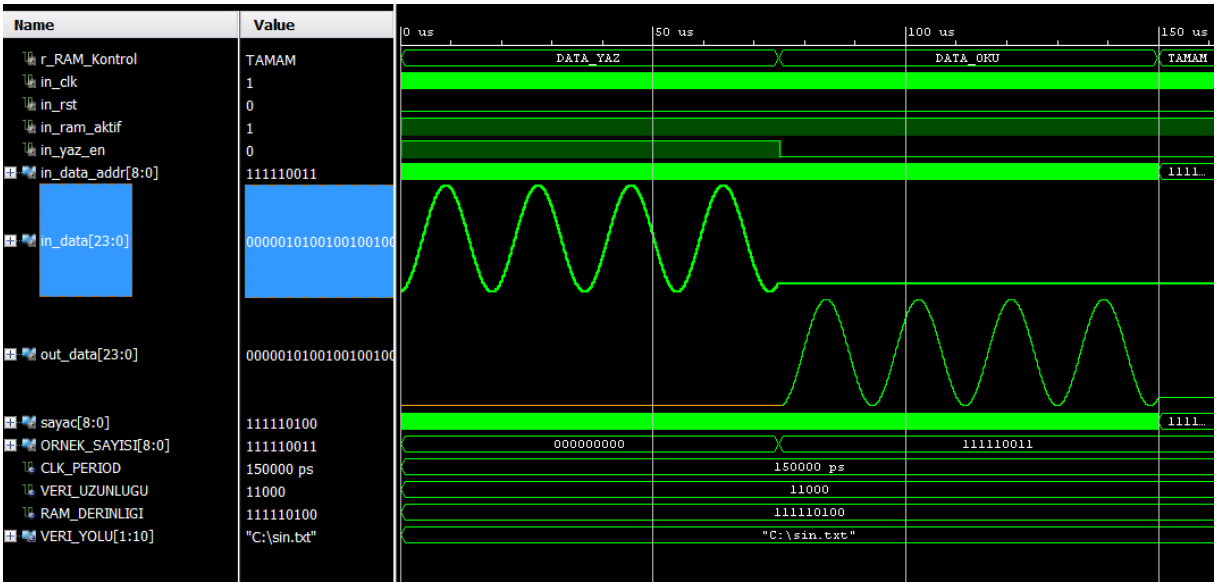
RAM tasarımında 1 bitlik saat darbesi giriş portu, 1 bitlik RAM aktif giriş portu, generic **ROM_DERINLIGI** parametresine bağlı olarak hesaplanan adres giriş portu, 1 bitlik yazma aktif giriş portu, RAM'a dataların yazılması için **VERI_UZUNLUGU** uzunluğunda data giriş portu ve okununan adresindeki datanın RAM dışına aktarılması için **VERI_UZUNLUGU** uzunluğunda çıkış portu mevcuttur.

RAM adres uzunluğunun belirlenmesi için kullanılacak olan **log2_int** fonskiyonu 4. satırda tanımlanan **ornekler_paket** (Bölüm 9.10'da verilmiştir) paketi içerisinde tanımlanmıştır. 24. satırda **VERI_UZUNLUGU** genişliğine sahip **RAM_DERINLIGI** derinliğinde tip tanımlama işlemi yapılmıştır.

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3. use IEEE.STD_LOGIC_UNSIGNED.ALL;
4. use work.ornekler_paket.all;
5.
```

```
6. entity blok_ram is
7.   generic(
8.     VERI_UZUNLUGU : integer := 8;
9.     RAM_DERINLIGI : integer := 110
10.  );
11.  port(
12.    in_clk : in std_logic;
13.    in_rst : in std_logic;
14.    in_ram_aktif : in std_logic;
15.    in_yaz_en : in std_logic;
16.    in_data_addr : in std_logic_vector(log2_int(RAM_DERINLIGI) - 1
   downto 0);
17.    in_data : in std_logic_vector(VERI_UZUNLUGU - 1 downto 0);
18.    out_data : out std_logic_vector(VERI_UZUNLUGU - 1 downto 0)
19.  );
20.end blok_ram;
21.
22.architecture Behavioral of blok_ram is
23.
24.  type t_BRAM_DATA is array (0 to RAM_DERINLIGI - 1) of
   std_logic_vector(VERI_UZUNLUGU - 1 downto 0) ;
25.  signal r_BRAM_DATA : t_BRAM_DATA := (others =>(others => '0'));
26.
27.begin
28.
29.  process(in_clk, in_rst)
30.  begin
31.    if in_rst = '1' then
32.      r_BRAM_DATA <= (others =>(others => '0'));
33.    elsif rising_edge(in_clk) then
34.      if in_ram_aktif = '1' then
35.        out_data <= r_BRAM_DATA(conv_integer(in_data_addr));
36.        if in_yaz_en = '1' then
37.          r_BRAM_DATA(conv_integer(in_data_addr)) <= in_data;
38.        end if;
39.      end if;
40.    end if;
41.  end process;
42.
43.end Behavioral;
```

Aşağıda verilen **tb_blok_ram.vhd** VHDL kodu ile **blok_ram** varlığının benzetim işlemleri yapılmaktadır. Benzetim işlemlerinde, sinüs örneklerinin kaydedildiği **sin.txt** dosyasından okuma işlemleri yapılmakta ve okunan datalar RAM'a yazılmaktadır. Verilerin yazılma işlemi dosyanın sonuna gelinmesi veya RAM'in dolması durumlarında sonlanmaktadır. Yazma işlenmesinin sonlanması ile birlikte toplmda RAM'a yazılan data sayısı hafızaya alınmaktadır. Daha sonra ise okuma işlemi başlamaktadır. Okuma işlemleri aslında başlangıç anından itibaren yapılmaktadır. RAM'in başlangıç değerlerinin tamamının sıfır olması nedeniyle verilerin yazılma işlemi anında okunan dataların değeri sıfır olmaktadır. Okuma işleminde sinüs datalarının RAM'a yazılmış olması ile birlikte **out_data** çıkış portunda sinüs örnekleri görülmektedir (Şekil 9-25).



Şekil 9-25 block_ram varlığı benzetim çıktısı

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3. use IEEE.STD_LOGIC_ARITH.ALL;
4. use IEEE.STD_LOGIC_UNSIGNED.ALL;
5. use std.textio.ALL;
6. use work.ornekler_paket.all;
7.
8. entity tb_blok_ram is
9. end tb_blok_ram;
10.
11.architecture Behavioral of tb_blok_ram is
12.
13.  component blok_ram
14.  generic(
15.    VERI_UZUNLUGU : integer := 8;
16.    RAM_DERINLIGI : integer := 110
17.  );
18.  port(
19.    in_clk : in std_logic;
20.    in_rst : in std_logic;
```

```
21.     in_ram_aktif : in std_logic;
22.     in_yaz_en : in std_logic;
23.     in_data_addr  :  in  std_logic_vector(log2_int(RAM_DERINLIGI)  -  1
    downto 0);
24.     in_data : in std_logic_vector(VERI_UZUNLUGU - 1 downto 0);
25.     out_data : out std_logic_vector(VERI_UZUNLUGU - 1 downto 0)
26.   );
27.   end component;
28.
29.   type t_RAM_Kontrol is (DATA_YAZ, DATA_OKU, TAMAM);
30.   signal r_RAM_Kontrol : t_RAM_Kontrol := DATA_YAZ;
31.
32.   constant CLK_PERIOD : time := 150 ns;
33.   constant VERI_UZUNLUGU : integer := 24;
34.   constant RAM_DERINLIGI : integer := 500;
35.   constant VERI_YOLU : string := "C:\sin.txt";
36.
37.   signal in_clk : std_logic := '0';
38.   signal in_rst : std_logic := '0';
39.   signal in_ram_aktif : std_logic := '0';
40.   signal in_yaz_en : std_logic := '0';
41.   signal  in_data_addr  :  std_logic_vector(log2_int(RAM_DERINLIGI)  -  1
    downto 0) := (others => '0');
42.   signal  in_data  :  std_logic_vector(VERI_UZUNLUGU  -  1  downto  0)  :=
    (others => '0');
43.   signal  out_data  :  std_logic_vector(VERI_UZUNLUGU  -  1  downto  0)  :=
    (others => '0');
44.   signal  sayac  :  std_logic_vector(log2_int(RAM_DERINLIGI)  -  1  downto
    0) := (others => '0');
45.   signal  ORNEK_SAYISI  :  std_logic_vector(log2_int(RAM_DERINLIGI)  -  1
    downto 0) := (others => '0');
46.
47.begin
48.
49.   process
50.   begin
51.     in_clk <= '1';
52.     wait for CLK_PERIOD / 2;
53.     in_clk <= '0';
54.     wait for CLK_PERIOD / 2;
55.   end process;
56.
```

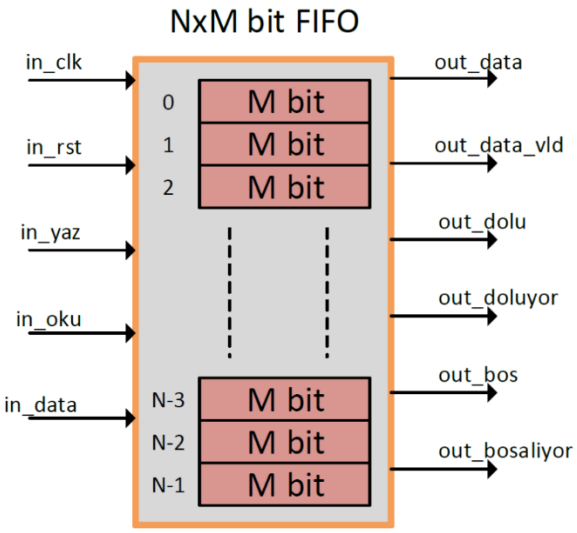
```
57.  process(in_clk)
58.    file dosya : text open read_mode is VERI_YOLU;
59.    variable satir : line;
60.    variable data : integer;
61.  begin
62.    if rising_edge(in_clk) then
63.      case r_RAM_Kontrol is
64.        when DATA_YAZ =>
65.          if (not endfile(dosya)) then
66.            readline(dosya, satir);
67.            read(satir, data);
68.            in_data <= conv_std_logic_vector(data, VERI_UZUNLUGU);
69.            in_data_addr <= sayac;
70.            in_yaz_en <= '1';
71.            in_ram_aktif <= '1';
72.            if sayac = RAM_DERINLIGI - 1 then
73.              sayac <= (others => '0');
74.              ORNEK_SAYISI <= sayac;
75.              r_RAM_Kontrol <= DATA_OKU;
76.            else
77.              sayac <= sayac + 1;
78.            end if;
79.          end if;
80.
81.        when DATA_OKU =>
82.          in_yaz_en <= '0';
83.          in_data_addr <= sayac;
84.          sayac <= sayac + 1;
85.          if sayac = ORNEK_SAYISI then
86.            r_RAM_Kontrol <= TAMAM;
87.          end if;
88.
89.        when TAMAM =>
90.          in_ram_aktif <= '0';
91.
92.        when others => NULL;
93.      end case;
94.    end if;
95.  end process;
96.
```

```
97.  blok_ram_map : blok_ram
98.  generic map(
99.    VERI_UZUNLUGU => VERI_UZUNLUGU,
100.     RAM_DERINLIGI => RAM_DERINLIGI
101.  )
102.  port map(
103.    in_clk => in_clk,
104.    in_rst => in_rst,
105.    in_ram_aktif => in_ram_aktif,
106.    in_yaz_en => in_yaz_en,
107.    in_data_addr => in_data_addr ,
108.    in_data => in_data,
109.    out_data => out_data
110.  );
111. end Behavioral;
```

9.12. VHDL'de FIFO tasarımı

Yaptığımız tasarımlarda zaman zaman farklı hızlarda çalışan modüller arasında veri alış-verişi yapmamız gerekebilir. Ya da bazı durumlarda işlenmesi gereken verilerin önce belli bir miktar biriktirilip daha sonra işlenmesi gerekebilir. Bu ve bunun gibi durumlarda en sık tercih edilen yaklaşım FIFO (First In First Out – İlk Giren İlk Çıkar) kullanımıdır. FIFO'ya giren veriler giriş sırasına çıkışından alınır. FIFO elemanını, hafıza adresini otomatik olarak ayarlayan, verileri geldiği sıraya göre çıkışa aktaran bir hafıza türü olarak da düşünmek mümkündür.

Şekil 9-26'den de görüleceği üzere FIFO okuma ve yazma aktif giriş portlarına, data giriş portuna, data çıkış portuna, çıkış data yürürlülükte portuna, FIFO dolu ve boş çıkış protlarına, FIFO doluyor ve boşalıyor çıkış portlarına sahiptir.



Şekil 9-26 NxM bitlik FIFO

Örnek 9.12: Şekil 9-26'de gösterilen FIFO tasarımın yapıldığı **FIFO.vhd** VHDL kodu aşağıda verilmiştir. **FIFO** varlığımıza ilişkin generic bildirimleri 6-11. satırlarda, port bildirimleri 12-24. satırları arasında yapılmaktıdır.

FIFO tasarımında 1 bitlik saat darbesi giriş portu, 1 bitlik FIFO reset giriş portu, 1 bitlik FIFO yazma aktif giriş portu, 1 bitlik FIFO okuma aktif giriş portu, **VERI_UZUNLUGU** uzunluğunda data giriş portu, **VERI_UZUNLUGU** uzunluğunda data çıkış portu, 1 bitlik çıkış datası gerçerli portu, 1 bitlik FIFO dolu uyarı portu, 1 bitlik FIFO boş uyarı portu, , 1 bitlik FIFO doluyor uyarı portu ve 1 bitlik FIFO boşalıyor uyarı portu mevcuttur.

29. satırda **VERI_UZUNLUGU** genişliğine sahip **FIFO_DERINLIGI** derinliğinde tip tanımlama işlemi yapılmıştır. 63-67. satırlarda tanımlı koşul ifadeleri ile yazma işleminde **r_fifo_sayac** sinyalinin değerinin bir artırılması, okuma işleminde ise **r_fifo_sayac** sinyalinin değerinin bir azaltılması işlemleri gerçekleştirilmektedir.

43-50. satırlar arasında ise **r_fifo_sayac** sinyalinin her saat darbesinde kontrolü ile FIFO doluluk boşluk durumları belirlenmektedir.

69-75. satırlarda her yazma işlemi yapıldığında **ind_yaz** sinyalinin değeri bir artırılmaktadır. **ind_yaz** sinyai değeri **FIFO_DERINLIGI - 1** değerine ulaştığında değeri sıfırlanmaktadır. 76-87. satırlarda her yazma işlemi yapıldığında **ind_oku** sinyalinin değeri bir artırılmaktadır.

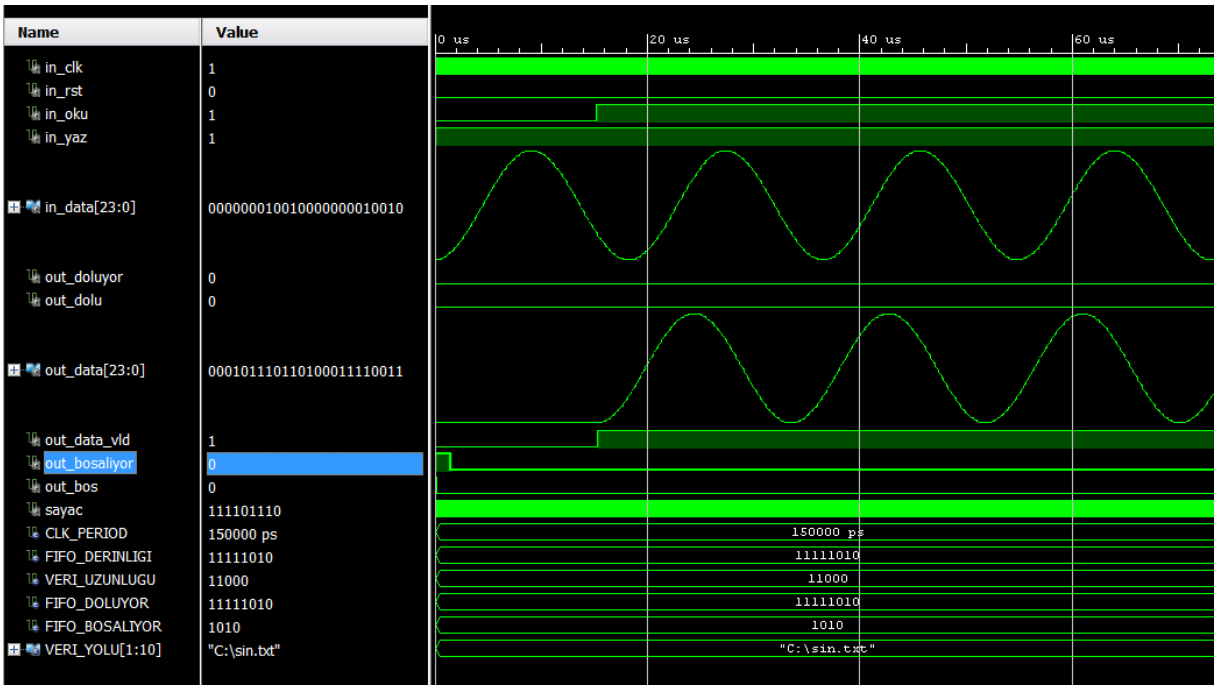
ind_oku sinyali değeri **FIFO_DERINLIGI - 1** değerine ulaştığında kendi değerini sıfırlamaktadır. **r_FIFO_DATA** sinyalinin **ind_oku** adresindeki veri değeri **r_data** sinyaline atanmaktadır. Atama işlemi ile birlikte **r_data_vld** sinyali **'1'** değerini almaktadır. 88-90. satırlarda **r_FIFO_DATA** sinyalinin **ind_yaz** adresine **in_data** sinyali atanmaktadır.

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3. use IEEE.STD_LOGIC_UNSIGNED.ALL;
4.
5. entity FIFO is
6.   Generic(
7.     FIFO_DERINLIGI : integer := 250;
8.     VERI_UZUNLUGU : integer := 24;
9.     FIFO_DOLUYOR : integer := 250;
10.    FIFO_BOSALIYOR : integer := 10
11.  );
12.  Port (
13.    in_clk : in std_logic;
14.    in_rst : in std_logic;
15.    in_yaz : in std_logic;
16.    in_oku : in std_logic;
17.    in_data : in std_logic_vector(VERI_UZUNLUGU - 1 downto 0);
18.    out_doluyor : out std_logic;
19.    out_dolu : out std_logic;
```

```
20.     out_data : out std_logic_vector(VERI_UZUNLUGU - 1 downto 0);
21.     out_data_vld : out std_logic;
22.     out_bosaliyor : out std_logic;
23.     out_bos : out std_logic
24.   );
25.end FIFO;
26.
27.architecture Behavioral of FIFO is
28.
29.   type t_FIFO_DATA is array (0 to FIFO_DERINLIGI - 1) of
   std_logic_vector(VERI_UZUNLUGU - 1 downto 0) ;
30.   signal r_FIFO_DATA : t_FIFO_DATA := (others =>(others => '0'));
31.   signal r_fifo_sayac : integer range -1 to FIFO_DERINLIGI + 1 := 0;
32.   signal ind_yaz : integer range 0 to FIFO_DERINLIGI - 1 := 0;
33.   signal ind_oku : integer range 0 to FIFO_DERINLIGI - 1 := 0;
34.   signal bayrak_dolu  : std_logic := '0';
35.   signal bayrak_bos : std_logic := '0';
36.   signal r_data_vld : std_logic := '0';
37.   signal r_data : std_logic_vector(VERI_UZUNLUGU - 1 downto 0) :=
   (others => '0');
38.
39.begin
40.
41.   out_data <= r_data;
42.   out_data_vld <= r_data_vld;
43.   out_dolu  <= '1' when r_fifo_sayac = FIFO_DERINLIGI else '0';
44.   out_bos <= '1' when r_fifo_sayac = 0 else '0';
45.
46.   bayrak_dolu  <= '1' when r_fifo_sayac = FIFO_DERINLIGI else '0';
47.   bayrak_bos <= '1' when r_fifo_sayac = 0 else '0';
48.
49.   out_doluyor <= '1' when r_fifo_sayac > FIFO_DOLUYOR else '0';
50.   out_bosaliyor <= '1' when r_fifo_sayac < FIFO_BOSALIYOR else '0';
51.
52.   process(in_clk, in_rst)
53.   begin
54.     if in_rst = '1' then
55.       r_FIFO_DATA <= (others =>(others => '0'));
56.       r_fifo_sayac <= 0;
57.       ind_yaz <= 0;
58.       ind_oku <= 0;
```

```
59.       r_data_vld <= '0';
60.       r_data <= (others => '0');
61.
62.     elsif rising_edge(in_clk) then
63.       if in_yaz = '1' and in_oku = '0' then
64.         r_fifo_sayac <= r_fifo_sayac + 1;
65.       elsif in_yaz = '0' and in_yaz = '1' then
66.         r_fifo_sayac <= r_fifo_sayac - 1;
67.       end if;
68.
69.       if in_yaz = '1' and bayrak_dolu = '0' then
70.         if ind_yaz = FIFO_DERINLIGI - 1 then
71.           ind_yaz <= 0;
72.         else
73.           ind_yaz <= ind_yaz + 1;
74.         end if;
75.       end if;
76.       if (in_oku = '1' and bayrak_bos = '0') then
77.         if ind_oku = FIFO_DERINLIGI - 1 then
78.           ind_oku <= 0;
79.         else
80.           ind_oku <= ind_oku + 1;
81.         end if;
82.         r_data <= r_FIFO_DATA(ind_oku);
83.         r_data_vld <= '1';
84.       else
85.         r_data_vld <= '0';
86.       end if;
87.     end if;
88.     if in_yaz = '1' then
89.       r_FIFO_DATA(ind_yaz) <= in_data;
90.     end if;
91.   end process;
92.end Behavioral;
```

Aşağıda verilen **tb_FIFO.vhd** VHDL kodu ile **FIFO** varlığının benzetim işlemleri yapılmaktadır. Benzetim işlemlerinde, sinüs örneklerinin kaydedildiği **sin.txt** dosyasından okuma işlemleri yapılmakta ve okunan datalar FIFO'ya yazılmaktadır. 100. örneğin FIFO'ya yazılmasından sonra FIFO'dan okuma aktif edilmektedir. Okuma işlemin aktif olması ile birlikte FIFO'ya yazılmış olması sinüs örnekleri **out_data** çıkış portunda görülmektedir (Şekil 9-27).



Şekil 9-27 FIFO varlığı benzetim çıktısı

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3. use IEEE.STD_LOGIC_ARITH.ALL;
4. use std.textio.ALL;
5.
6. entity tb_fifo is
7. end tb_fifo;
8.
9. architecture Behavioral of tb_fifo is
10.   component FIFO
11.   Generic(
12.     FIFO_DERINLIGI : integer := 250;
13.     VERI_UZUNLUGU : integer := 24;
14.     FIFO_DOLUYOR : integer := 250;
15.     FIFO_BOSALIYOR : integer := 10
16.   );
17.   Port (
18.     in_clk : in std_logic;
19.     in_rst : in std_logic;
20.     in_yaz : in std_logic;
21.     in_oku : in std_logic;
22.     in_data : in std_logic_vector(VERI_UZUNLUGU - 1 downto 0);
```

```
23.     out_doluyor : out std_logic;
24.     out_dolu : out std_logic;
25.     out_data : out std_logic_vector(VERI_UZUNLUGU - 1 downto 0);
26.     out_data_vld : out std_logic;
27.     out_bosaliyor : out std_logic;
28.     out_bos : out std_logic
29.   );
30.   end component;
31.
32.   constant CLK_PERIOD : time := 150 ns;
33.   constant FIFO_DERINLIGI : integer := 250;
34.   constant VERI_UZUNLUGU : integer := 24;
35.   constant FIFO_DOLUYOR : integer := 250;
36.   constant FIFO_BOSALIYOR : integer := 10;
37.   constant VERI_YOLU : string := "C:\sin.txt";
38.
39.   signal in_clk : std_logic := '0';
40.   signal in_rst : std_logic := '0';
41.   signal in_oku : std_logic := '0';
42.   signal in_yaz : std_logic := '0';
43.   signal in_data : std_logic_vector(VERI_UZUNLUGU - 1 downto 0) :=
    (others => '0');
44.   signal out_doluyor : std_logic := '0';
45.   signal out_dolu : std_logic := '0';
46.   signal out_data : std_logic_vector(VERI_UZUNLUGU - 1 downto 0) :=
    (others => '0');
47.   signal out_data_vld : std_logic := '0';
48.   signal out_bosaliyor : std_logic := '0';
49.   signal out_bos : std_logic := '0';
50.   signal sayac : integer := 0;
51.
52.begin
53.
54.   process
55.   begin
56.     in_clk <= '1';
57.     wait for CLK_PERIOD / 2;
58.     in_clk <= '0';
59.     wait for CLK_PERIOD / 2;
60.   end process;
61.
```

```
62.  process(in_clk)
63.    file dosya : text open read_mode is VERI_YOLU;
64.    variable satir : line;
65.    variable data : integer;
66.  begin
67.    if rising_edge(in_clk) then
68.      if (not endfile(dosya)) then
69.        readline(dosya, satir);
70.        read(satir, data);
71.        in_data <= conv_std_logic_vector(data, VERI_UZUNLUGU);
72.        in_yaz <= '1';
73.        sayac <= sayac + 1;
74.        if sayac > 100 then
75.          in_oku <= '1';
76.        end if;
77.      end if;
78.    end if;
79.  end process;
80.
81.  FIFO_map : FIFO
82.  generic map(
83.    FIFO_DERINLIGI => FIFO_DERINLIGI,
84.    VERI_UZUNLUGU => VERI_UZUNLUGU,
85.    FIFO_DOLUYOR => FIFO_DOLUYOR,
86.    FIFO_BOSALIYOR => FIFO_BOSALIYOR  )
87.  port map(
88.    in_clk => in_clk,
89.    in_rst => in_rst,
90.    in_yaz => in_yaz,
91.    in_oku => in_oku,
92.    in_data => in_data,
93.    out_doluyor => out_doluyor,
94.    out_dolu => out_dolu,
95.    out_data => out_data,
96.    out_data_vld => out_data_vld,
97.    out_bosaliyor => out_bosaliyor,
98.    out_bos => out_bos
99.  );
100.     end Behavioral;
```

9.13. Sinyal İşlemede Konvolüsyon

Bu kısma kadar olan örneklerde genelde sayısal tasarımla alakalı uygulamalar gerçekleştirdik. Uygulamalarla VHDL ile ilgili kullanım şekillerini ve tasasrım yollarını göstermeye çalıştık.

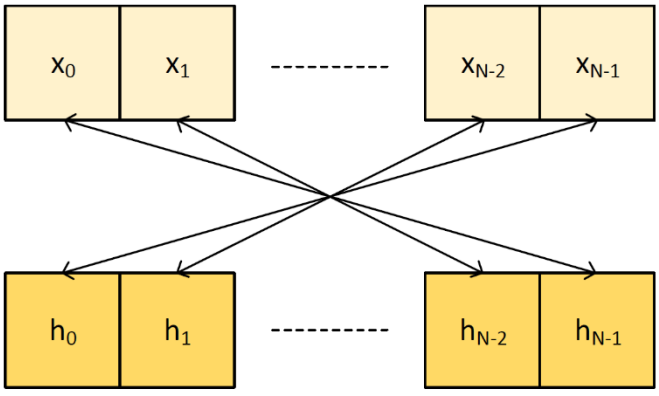
Bu örnekte ise temel bir işaret işleme uygulamasına geçiş yapıyoruz. Bu kısımda bahsedilen kavramları anlayabilmek için temel seviyede işaret işleme ile alakalı konuların bilinmesi gerekmektedir. Bu bölüme devam etmeden önce lütfen işaret işleme ile alakalı kaynakları gözden geçriniz.

İşaret işlemede çok kullanışlı olan araçlardan biri de konvolüsyon işlemidir. Konvolusyon, giriş sinyali ve doğrusal sistemin dürtü tepki (impulse response) fonksiyonu bilindiğinde çıkış işaretini bulmaya yarayan bir işlemdir. Sistemin dürtü tepki fonksiyonu $h[n]$'in N tane çarpanlı bir sonlu filtre olduğunu varsayarsak, giriş işareti $x[n]$ sonsuz uzunlukta olduğu durumda dahi, filtrenin çıkış işareti $y[n]$ konvolüsyon işlemi ile aşağıda verilen deknklemde gösterilmiştir.

$$y[n] = x[n] * h[n] = \sum_{k=0}^{N-1} h[k]x[n-k]$$

Dürtü tepki fonksiyonu katsayıları ve katsayıların sayısı tasarım aşamsından önce belirlenmelidir. Örneğin sonlu dürtü yanıtlı bir (N -1). dereceden filtrenin katsayısı N adettir. Bu tepki fonksiyonu için tasarımda N uzunluğunda bellek bloğu (**h**) ayrılması gerekmektedir.

Giriş işaretinin dürtü tepki fonksiyonu ile konvolüsyon işleminin yapılabilmesi için dürtü fonksiyonu katsayıları uzunluğunda giriş örneğinin saklaması gerekmektedir. Yukarıda verilen filtre örneği için N uzunluğunda bellek bloğu (**x**) ayrılması gerekmektedir. Giriş sinyali hafıza bloğu ile dürtü tepki fonksiyonunun katsayılarının bulunduğu bellek bloğu ile konvolüsyon işlemi, Şekil 9-28'de gösterilmiştir.



Şekil 9-28 Konvolüsyon işlemi

Örnek 9.13: Aşağıda sinyal işleme uygulamaları için katsayıların yüklenembilme özelliğine sahip ve konvolüsyon işlemi yapan **konvolusyon_signal.vhd** VHDL kodu verilmiştir. Uygulamada konvolüsyon işlemi için kullanıcı kendi oluşturduğu dürtü tepki fonksiyonun katsayılarını, katsayı hafıza bloğuna yükleyebilmektedir. Bu özellik ile **konvolusyon_signal** varlığı farklı dütrü tepki fonksiyonlarının gerçeklenebilmesine olanak sağlamaltadır.

konvolusyon_signal varlığına ilişkin generic tanımlamaları 8-13. satırlar arasında yapılmaktadır.

- 9. satırda tanımlı **VERI_UZUNLUGU** parametresi ile giriş data uzunluğu belirlenmektedir.
- 10. satırda tanımlı **KATSAYI** parametresi ile konvolüsyon için kullılacak katsayıların sayısı tanımlanmaktadır.
- 11. satırda tanımlı **KATSAYI_UZUNLUGU** parametresi ile katsayıların data uzunluğu belirlenmektedir.

- 12. satırda tanımlı **KATSAYI_CARPIM** paramatresi ise hesaplanan katsayıların 2'nin kaçıncı kuvveti ile çarpıldığını belirtmektedir. Bu parametre filtre çıkışında kullanılmaktadır.

konvolusyon_signal varlığına ilişkin port tanımlama işlemleri 14-26. satırlarda yapılmaktadır. 37. satırda **KATSAYI_UZUNLUGU** bit uzunluğunda ve **KATSAYI** boyunda tip tanımlama işlemi yapılarak konvolüsyon katsayılarının saklanacağı bellek tipi oluşturulmuştur.

40. satırda **VERI_UZUNLUGU** bit uzunluğunda **KATSAYI** boyunda tip tanımlama işlemi yapılarak giriş datalarının saklanacağı bellek tipi oluşturulmuştur.

Giriş datalarının saklanma işleminde giriş örnek datası belleğin 0. adresine yazılacak şekilde tasarım yapılmıştır. Bu nedenle 43-52. satırlarda tanımlı fonksiyon ile giriş datalarının tutulduğu bellekte dataların sağa kaydırma işlemi gerçekleştirilmektedir.

68-77. satırlarda tanımlı **process** ile konvolüsyon katsayılarının belleğe yazılma işlemi yapılmaktadır. Katsayı ve katsayı adres bilgisi ile birlikte aktif sinyali olması durumunda katsayı değeri **r_katsayi_bellek** sinyalinde ilgili adresadresteki yerine yazılmaktadır.

79-104. satırlarda tanımlı **process**'de data aktif sinyali ile **f_Bellek_kaydir** fonksiyonu çağrılarak **r_data_bellek** sinyali dataları sağa kayırdama işlemi yapılır ve giriş datası ilgili yerine yazılır. Data yazma işleminin tamamlanması ile birlikte **r_hesap_basla** sinyali aktif edilerek 106-144. satırlarda tanımlı **process**'de tanımlı konvolüsyon işlemi başlatılır.

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3. use IEEE.STD_LOGIC_SIGNED.all;
4. use IEEE.STD_LOGIC_ARITH.all;
5. use work.ornekler_paket.all;
6.
7. entity konvolusyon_sinyal is
8.   Generic(
9.     VERI_UZUNLUGU : integer := 24;
10.    KATSAYI : integer := 5;
11.    KATSAYI_UZUNLUGU : integer := 8;
12.    KATSAYI_CARPIM : integer := 3
13.  );
14.  Port (
15.    in_clk : in std_logic;
16.    in_rst : in std_logic;
17.    in_en : in std_logic;
18.    in_katsayi_vld : in std_logic;
19.    in_katsayi_addr : in std_logic_vector(log2_int(KATSAYI) downto 0);
20.    in_katsayi_data : in std_logic_vector(KATSAYI_UZUNLUGU - 1 downto
   0);
21.    in_data : in std_logic_vector(VERI_UZUNLUGU - 1 downto 0);
22.    in_data_vld : in std_logic;
23.    out_data : out std_logic_vector(VERI_UZUNLUGU - 1 downto 0);
24.    out_data_vld : out std_logic;
25.    out_calisiyor : out std_logic
```

```
26.  );
27.end konvolusyon_sinyal;
28.
29.architecture Behavioral of konvolusyon_sinyal is
30.
31.  type t_Kayma_Ctrl is (BOSTA, DATA_KAYDIR);
32.  signal r_Kayma_Ctrl : t_Kayma_Ctrl := BOSTA;
33.
34.  type t_Filtre_Hesap is (BOSTA, HESAPLA, TAMAM);
35.  signal r_Filtre_Hesap : t_Filtre_Hesap := BOSTA;
36.
37.  type t_katsayi_bellek is array (0 to KATSAYI - 1 ) of
   std_logic_vector(KATSAYI_UZUNLUGU - 1 downto 0);
38.  signal r_katsayi_bellek : t_katsayi_bellek := (others => (others =>
   '0'));
39.
40.  type t_data_bellek is array (0 to KATSAYI - 1 ) of
   std_logic_vector(VERI_UZUNLUGU - 1 downto 0);
41.  signal r_data_bellek : t_data_bellek := (others => (others => '0'));
42.
43.  function f_Bellek_Kaydir(r_data_bellek : t_data_bellek; in_data :
   std_logic_vector(VERI_UZUNLUGU - 1 downto 0)) return t_data_bellek is
44.    variable v_data_bellek : t_data_bellek;
45.  begin
46.    v_data_bellek := r_data_bellek;
47.    for n_i in KATSAYI - 2 downto 0 loop
48.        v_data_bellek(n_i + 1) := v_data_bellek(n_i);
49.    end loop;
50.    v_data_bellek(0) := in_data;
51.    return v_data_bellek;
52.  end f_Bellek_Kaydir;
53.
54.  signal r_hesap_basla : std_logic := '0';
55.  signal r_data : std_logic_vector(VERI_UZUNLUGU - 1 downto 0) :=
   (others => '0');
56.  signal r_toplam : std_logic_vector(VERI_UZUNLUGU + KATSAYI_UZUNLUGU
   + log2_int(KATSAYI) - 1 downto 0) := (others => '0');
57.  signal r_data_out : std_logic_vector(VERI_UZUNLUGU - 1 downto 0) :=
   (others => '0');
58.  signal r_data_out_vld : std_logic := '0';
59.  signal r_calisiyor : std_logic := '0';
60.  signal n_i : integer := 0;
61.
```

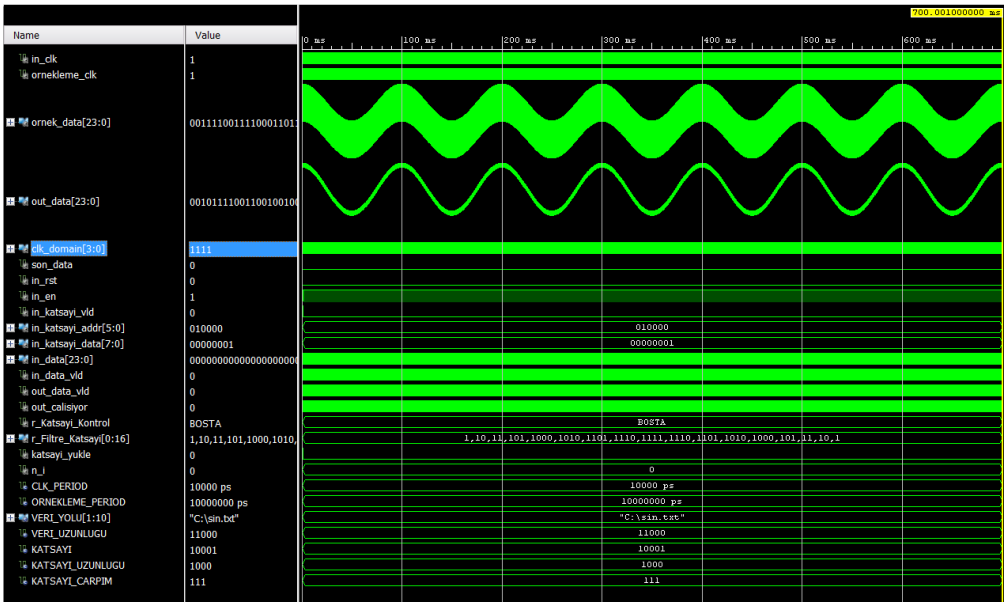
```
62.begin
63.
64.  out_data <= r_data_out;
65.  out_data_vld <= r_data_out_vld;
66.  out_calisiyor <= r_calisiyor;
67.
68.  process(in_clk, in_rst)
69.  begin
70.    if in_rst = '1' then
71.      r_katsayi_bellek <= (others => (others => '0'));
72.    elsif rising_edge(in_clk) then
73.      if in_katsayi_vld = '1' then
74.        r_katsayi_bellek(conv_integer(in_katsayi_addr))             <=
   in_katsayi_data;
75.      end if;
76.    end if;
77.    end process;
78.
79.  process(in_clk, in_rst)
80.  begin
81.    if in_rst = '1' then
82.      r_Kayma_Ctrl <= BOSTA;
83.      r_data_bellek <= (others => (others => '0'));
84.      r_hesap_basla <= '0';
85.      r_data <= (others => '0');
86.
87.    elsif rising_edge(in_clk) then
88.      r_hesap_basla <= '0';
89.      case r_Kayma_Ctrl is
90.        when BOSTA =>
91.          if in_data_vld = '1' then
92.            r_data <= in_data;
93.            r_Kayma_Ctrl <= DATA_KAYDIR;
94.          end if;
95.
96.        when DATA_KAYDIR =>
97.          r_data_bellek <= f_Bellek_Kaydir(r_data_bellek, r_data);
98.          r_Kayma_Ctrl <= BOSTA;
99.          r_hesap_basla <= '1';
100.
101.       when others => NULL;
```

```
102.           end case;
103.       end if;
104.     end process;
105.
106.     process(in_clk, in_rst)
107.     begin
108.         if in_rst = '1' then
109.             r_Filtre_Hesap <= BOSTA;
110.             r_toplam <= (others => '0');
111.             r_data_out_vld <= '0';
112.             r_data_out <= (others => '0');
113.             r_calisiyor <= '0';
114.             n_i <= 0;
115.
116.     elsif rising_edge(in_clk) then
117.             r_data_out_vld <= '0';
118.             case r_Filtre_Hesap is
119.                 when BOSTA =>
120.                     r_calisiyor <= '0';
121.                     if r_hesap_basla = '1' and in_en = '1' then
122.                         r_Filtre_Hesap <= HESAPLA;
123.                         r_calisiyor <= '1';
124.                     end if;
125.
126.                 when HESAPLA =>
127.                     r_toplam <= r_toplam + sxt((r_data_bellek(n_i))
    * (r_katsayi_bellek(KATSAYI - 1 - n_i)), r_toplam'length);
128.                     n_i <= n_i + 1;
129.                     if n_i = KATSAYI - 1 then
130.                         n_i <= 0;
131.                         r_Filtre_Hesap <= TAMAM;
132.                     end if;
133.
134.                 when TAMAM =>
135.                     r_toplam <= (others => '0');
136.                     r_calisiyor <= '0';
137.                     r_data_out_vld <= '1';
138.                     r_data_out    <=    r_toplam(KATSAYI_CARPIM    +
    VERI_UZUNLUGU - 1 downto KATSAYI_CARPIM);
139.                     r_Filtre_Hesap <= BOSTA;
140.
```

```
141.                 when others => NULL;
142.             end case;
143.         end if;
144.     end process;
145.
146.     end Behavioral;
```

Örnek 9.13'de tanımlı **ornekler_paket.vhd** paket dosyası Örnek 9.10'da verilmiştir.

konvolusyon_signal varlığının benzetiminin yapılabilmesi için aşağıda **tb_konvolusyon_signal.vhd** VHDL sınama kodu verilmiştir. Kodda tanımlı alçak geçiren süzgeç parametreleri öncelikle **konvolusyon_signal** varlığına yazılmaktadır. Yazma işleminin bitiminde konvolusyon hesaplama işlemi aktif hale gelmektedir. **konvolusyon_signal** varlığına giriş olarak 10 Hz ve 10 Khz frekansında iki sinüs sinyalinin toplamı100 KHz ile örneklenerek verilmiştir. Şekil 9-29'den de görüleceği üzere **ornek_data** sinyali'nin giriş olarak verildiği **konvolusyon_signal** varlığı çıkışında elde edilen **out_data** sinyali incelendiğinde filtreleme işlemi başarılı bir şekilde gerçekleştirilmiştir.



Şekil 9-29 Alçak geçiren filtre uygulaması

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3. use IEEE.STD_LOGIC_UNSIGNED.all;
4. use IEEE.STD_LOGIC_ARITH.all;
5. use work.ornekler_paket.all;
6. use std.textio.ALL;
7.
8. entity tb_konvolusyon_signal is
9. end tb_konvolusyon_signal;
10.
```

```
11.architecture Behavioral of tb_konvolusyon_signal is

12.  component konvolusyon_sinyal is
13.  Generic(
14.    VERI_UZUNLUGU : integer := 24;
15.    KATSAYI : integer := 5;
16.    KATSAYI_UZUNLUGU : integer := 8;
17.    KATSAYI_CARPIM : integer := 3
18.  );
19.  Port (
20.    in_clk : in std_logic;
21.    in_rst : in std_logic;
22.    in_en : in std_logic;
23.    in_katsayi_vld : in std_logic;
24.    in_katsayi_addr : in std_logic_vector(log2_int(KATSAYI)  downto 0);
25.    in_katsayi_data : in std_logic_vector(KATSAYI_UZUNLUGU - 1 downto
   0);
26.    in_data : in std_logic_vector(VERI_UZUNLUGU - 1 downto 0);
27.    in_data_vld : in std_logic;
28.    out_data : out std_logic_vector(VERI_UZUNLUGU - 1 downto 0);
29.    out_data_vld : out std_logic;
30.    out_calisiyor : out std_logic
31.  );
32.  end component;
33.
34.  constant CLK_PERIOD : time := 10 ns;
35.  constant ORNEKLEME_PERIOD : time := 10 us;
36.  constant VERI_YOLU : string := "C:\sin.txt";
37.  constant VERI_UZUNLUGU : integer := 24;
38.  constant KATSAYI : integer := 17;
39.  constant KATSAYI_UZUNLUGU : integer := 8;
40.  constant KATSAYI_CARPIM : integer := 7;
41.
42.  signal in_clk : std_logic  := '0';
43.  signal ornekleme_clk : std_logic := '0';
44.  signal ornek_data : std_logic_vector(23 downto 0) := (others => '0');
45.  signal clk_domain : std_logic_vector(3 downto 0) := (others => '0');
46.  signal son_data : std_logic := '0';
47.  signal in_rst : std_logic := '0';
48.  signal in_en : std_logic := '0';
49.  signal in_katsayi_vld : std_logic := '0';
```

```
50.  signal in_katsayi_addr : std_logic_vector(log2_int(KATSAYI) downto
   0) := (others => '0');
51.  signal in_katsayi_data : std_logic_vector(KATSAYI_UZUNLUGU - 1 downto
   0) := (others => '0');
52.  signal in_data : std_logic_vector(VERI_UZUNLUGU - 1 downto 0) :=
   (others => '0');
53.  signal in_data_vld : std_logic := '0';
54.  signal out_data : std_logic_vector(VERI_UZUNLUGU - 1 downto 0) :=
   (others => '0');
55.  signal out_data_vld : std_logic := '0';
56.  signal out_calisiyor : std_logic := '0';
57.
58.  type t_Katsayi_Kontrol is  (BOSTA, YUKLE);
59.  signal r_Katsayi_Kontrol : t_Katsayi_Kontrol := BOSTA;
60.
61.  type t_Filtre_Katsayi is array (0 to KATSAYI - 1) of integer;
62.  signal r_Filtre_Katsayi : t_Filtre_Katsayi := (1, 2, 3, 5, 8, 10, 13,
   14, 15, 14, 13, 10, 8, 5, 3, 2, 1);
63.
64.  signal katsayi_yukle : std_logic := '0';
65.  signal n_i : integer := 0;
66.
67.begin
68.
69.  process
70.  begin
71.    in_clk <= '1';
72.    wait for CLK_PERIOD / 2;
73.    in_clk <= '0';
74.    wait for CLK_PERIOD / 2;
75.  end process;
76.
77.  process
78.  begin
79.    ornekleme_clk <= '1';
80.    wait for ORNEKLEME_PERIOD / 2;
81.    ornekleme_clk <= '0';
82.    wait for ORNEKLEME_PERIOD / 2;
83.  end process;
84.
85.  process
86.  begin
```

```
87.    katsayi_yukle <= '0';
88.    wait for CLK_PERIOD * 2;
89.    katsayi_yukle <= '1';
90.    wait for CLK_PERIOD ;
91.    katsayi_yukle <= '0';
92.    wait;
93.  end process;
94.
95.  process (ornekleme_clk)
96.    file dosya : text open read_mode is VERI_YOLU ;
97.    variable satir : line;
98.    variable data : integer;
99.  begin
100.       if rising_edge(ornekleme_clk) then
101.           if not(endfile(dosya)) then
102.               readline(dosya, satir);
103.               read(satir, data);
104.               ornek_data <= conv_std_logic_vector(data, 24) ;
105.               son_data <= '0';
106.           else
107.               ornek_data <= conv_std_logic_vector(0, 24) ;
108.               son_data <= '1';
109.           end if;
110.       end if;
111.    end process;
112.
113.    process(in_clk)
114.    begin
115.    if rising_edge(in_clk) then
116.        clk_domain <= clk_domain(2 downto 0) & ornekleme_clk;
117.    end if;
118.    end process;
119.
120.    process (in_clk)
121.    begin
122.    if rising_edge(in_clk) then
123.        if son_data = '0' and clk_domain(3 downto 2) = "01" then
124.            in_data_vld <= '1' ;
125.            in_data <= ornek_data ;
126.        else
```

```
127.             in_data_vld <= '0' ;
128.             in_data <= (others=>'0') ;
129.         end if;
130.     end if;
131.     end process;
132.
133.     process(in_clk)
134.     begin
135.     if rising_edge(in_clk) then
136.         in_katsayi_vld <= '0';
137.         case r_Katsayi_Kontrol is
138.         when BOSTA =>
139.             if katsayi_yukle = '1' then
140.                 r_Katsayi_Kontrol <= YUKLE;
141.             end if;
142.
143.         when YUKLE =>
144.             in_katsayi_vld <= '1';
145.             in_katsayi_addr      <=      conv_std_logic_vector(n_i,
    log2_int(KATSAYI) + 1);
146.             in_katsayi_data        <=        conv_std_logic_vector(
    r_Filtre_Katsayi(n_i), KATSAYI_UZUNLUGU);
147.             if n_i = KATSAYI - 1 then
148.                 r_Katsayi_Kontrol <= BOSTA;
149.                 n_i <= 0;
150.                 in_en <= '1';
151.             else
152.                 n_i <= n_i + 1;
153.             end if;
154.         when others => NULL;
155.         end case;
156.     end if;
157.     end process;
158.
159.     konvolusyon_sinyal_map :  konvolusyon_sinyal
160.     Generic map(
161.     VERI_UZUNLUGU => VERI_UZUNLUGU,
162.     KATSAYI => KATSAYI,
163.     KATSAYI_UZUNLUGU => KATSAYI_UZUNLUGU,
164.     KATSAYI_CARPIM => KATSAYI_CARPIM
165.     )
```

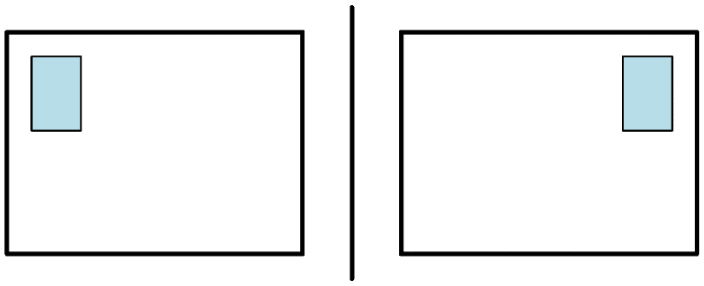
```
166.     Port map(
167.     in_clk => in_clk,
168.     in_rst => in_rst,
169.     in_en => in_en,
170.     in_katsayi_vld => in_katsayi_vld,
171.     in_katsayi_addr => in_katsayi_addr,
172.     in_katsayi_data => in_katsayi_data,
173.     in_data => in_data,
174.     in_data_vld => in_data_vld,
175.     out_data => out_data,
176.     out_data_vld => out_data_vld,
177.     out_calisiyor => out_calisiyor
178.     );
179.     end Behavioral;
```

9.14. Temel İmge İşleme Algoritmaları

Bu örnekte de işaret işleme uygulamalarına devam ediyoruz. Bir önceki başlık bir alçak geçiren filtre uygulaması gerçekleştirmiştik. Bu örnekte ise işaret işleme uygulamarı arasında bulunan temel imge (görüntü) işleme algoritmalarından ve VHDL ile tasarımından bahsedeceğiz.

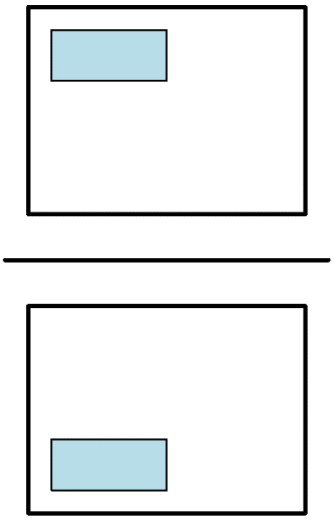
Uygulamaya geçmeden önce ilk olarak temel imge işleme işlemleri tanıtılacak olup ardından VHDL kodları verilecektir. Örnek uygulamada kullanılan görüntü 8 bit gri seviyeli bir görüntüdür. Bu yüzden her beneğin (pixel) alabileceği değerler 0 ile 255 arasında olmaktadır. (x,y) gösterimi ise imgenin her bir beneğinin koordinatlarını temsil etmektedir. Temel imge işleme algoritmaları sırasıyla verildiği gibidir:

Aynalama : Aynalama işleminde en soldaki sütun en sağdaki sütüna, en sağdaki sütün ise en soldaki sütuna yazılacak şekilde imge elemanları yer değiştirir (Şekil 9-30).



Şekil 9-30 İmgede aynalama

Ters Çevirme : Ters çevirme işleminde en alttaki satır en üstteki satıra, en üstteki satır ise en alttaki alttaki yazılacak şekilde imge elemanları yer değiştirir (Şekil 9-31).



Şekil 9-31 İmgede ters çevirme

Negatifleme : İmge değerlerinin ters çevrilmesi ile yapılmaktadır. Aşağıda negatifleme işlemine ilişkin denklem verilmiştir. 255 değeri 8bitlik bir beneğin alabileceği azami değer olup farklı bit uzunlukları için bu değer değişebilir.

$$Imge\_yeni(x,y) = 255 - Imge(x,y)$$

Eşikleme : İmge değerlerinin bir sayı değerinden büyük veya küçük olması durumuna göre eşikleme yapılmaktadır. Aşağıda eşikleme işlemine ilişkin denklem verilmiştir.

$$Imge\_yeni(x,y) = \begin{cases} 0, & Imge(x,y) < a \\ 255, & Imge(x,y) \geq a \end{cases}$$

Parlaklık : İmge değerlerinin bir sayı değeriyle toplanması veya çıkarılması ile parlaklık ayarı yapılmaktadır. Aşağıda parlaklık ayarlamasına ilişkin denklem verilmiştir. Bu işlem yapılırken her bir beneğin alabileceği azami ve asgari değerlere dikkat edilmelidir aksi halde istenmeyen sonuçlar oluşabilir. Örneğin 250 değerine sahip bir beneğe 10 eklenirse değerde taşma oluşacak ve beneğin yeni değeri 5 olacağı için siyaha dönecektir.(imgenin beneklerinin 8 bitlik uzunluğa sahip olduğu kabul edilmiştir.)

$$Imge\_yeni(x,y) = Imge(x,y) \pm a$$

Karşıtlık : İmge değerlerinin bir sayı değeriyle çarpılması ile karşıtlık ayarı yapılmaktadır. Aşağıda karşıtlık ayarlamasına ilişkin denklem verilmiştir. Bu işlem yapılırken her bir beneğin alabileceği azami ve asgari değerlere dikkat edilmelidir aksi halde istenmeyen sonuçlar oluşabilir.

$$Imge\_yeni(x,y) = a \times Imge(x,y)$$

Örnek 9.14 : Aşağıda temel imge algoritmalarının gerçekleştirildiği **temel_imge_isleme.vhd** VHDL kodu verilmiştir. **temel_imge_isleme** varlığında uygulanacak algoritmaya ilişkin RAM'dan okunacak datanın adres tanımlama işlemleri 83-91. satırlar arasında tanımlanmıştır. 101-131. satırlar arasında ise uygulanacak algoritmaya ilişkin hesaplama işlemleri yapılmaktadır.

```
1. library IEEE;
```

```
2. use IEEE.STD_LOGIC_1164.ALL;
3. use IEEE.STD_LOGIC_UNSIGNED.ALL;
4. use IEEE.STD_LOGIC_ARITH.ALL;
5. use work.ornekler_paket.all;
6.
7. entity temel_imge_isleme is
8.   generic(
9.     IMGE_SATIR : integer := 8;
10.    IMGE_SUTUN : integer := 8;
11.    VERI_UZUNLUGU : integer := 24
12.  );
13.  port(
14.    in_clk : in std_logic;
15.    in_rst : in std_logic;
16.    in_en  : in std_logic;
17.    in_basla : in std_logic;
18.    in_islem : in std_logic_vector(2 downto 0);
19.    in_data : in std_logic_vector(VERI_UZUNLUGU - 1 downto 0);
20.    in_data_vld : in std_logic;
21.    out_addr : out std_logic_vector(log2_int(IMGE_SATIR * IMGE_SUTUN)
   - 1 downto 0);
22.    out_addr_vld : out std_logic;
23.    out_data : out std_logic_vector(7 downto 0);
24.    out_data_vld : out std_logic;
25.    out_tamam : out std_logic
26.  );
27.end temel_imge_isleme;
28.
29.architecture Behavioral of temel_imge_isleme is
30.
31.  constant AYNALAMA : std_logic_vector(2 downto 0) := "000";
32.  constant TERS_CEVIRME : std_logic_vector(2 downto 0) := "001";
33.  constant NEGATIFLEME : std_logic_vector(2 downto 0) := "010";
34.  constant ESIKLEME : std_logic_vector(2 downto 0) := "011";
35.  constant PARLAKLIK_ARTIR : std_logic_vector(2 downto 0) := "100";
36.  constant PARLAKLIK_AZALT : std_logic_vector(2 downto 0) := "101";
37.  constant KARSITLIK_ARTIR : std_logic_vector(2 downto 0) := "110";
38.  constant KARSITLIK_AZALT : std_logic_vector(2 downto 0) := "111";
39.
40.  type t_Imge_Isleme is (BOSTA, RAMDAN_OKU, OKUMA_BEKLE, ISLEM_YAP,
   SAYAC_KONT, TAMAM );
```

```
41.  signal r_Imge_Isleme : t_Imge_Isleme := RAMDAN_OKU;
42.  signal n_i : integer := 0;
43.  signal n_j : integer := 0;
44.  signal r_addr : std_logic_vector(log2_int(IMGE_SATIR * IMGE_SUTUN) -
   1 downto 0) := (others => '0');
45.  signal r_addr_vld : std_logic := '0';
46.  signal r_data : std_logic_vector(7 downto 0) := (others => '0');
47.  signal r_data_vld : std_logic := '0';
48.  signal r_tamam : std_logic := '0';
49.
50.begin
51.
52.  out_addr <= r_addr;
53.  out_addr_vld <= r_addr_vld;
54.  out_data <= r_data;
55.  out_data_vld <= r_data_vld;
56.  out_tamam <= r_tamam;
57.
58.  process(in_clk, in_rst)
59.  begin
60.    if in_rst = '1' then
61.      r_Imge_Isleme <= BOSTA;
62.      n_i <= 0;
63.      n_j <= 0;
64.      r_addr <= (others => '0');
65.      r_addr_vld <= '0';
66.      r_data <= (others => '0');
67.      r_data_vld <= '0';
68.      r_tamam <= '0';
69.
70.    elsif rising_edge(in_clk) then
71.      if in_en = '1' then
72.        r_data_vld <= '0';
73.        r_addr_vld <= '0';
74.        r_tamam <= '0';
75.
76.      case r_Imge_Isleme is
77.        when BOSTA =>
78.          if in_basla = '1' then
79.            r_Imge_Isleme <= RAMDAN_OKU;
80.          end if;
```

```
81.
82.         when RAMDAN_OKU =>
83.           if in_islem = AYNALAMA then
84.             r_addr   <=   conv_std_logic_vector(n_i   *   IMGE_SUTUN   +
   (IMGE_SUTUN - 1 - n_j) ,  r_addr'length);
85.           elsif in_islem = TERS_CEVIRME then
86.             r_addr <= conv_std_logic_vector((IMGE_SATIR - 1 -n_i) *
   IMGE_SUTUN +  n_j ,  r_addr'length);
87.           elsif in_islem = NEGATIFLEME or in_islem = ESIKLEME or
88.           in_islem = PARLAKLIK_ARTIR or in_islem = PARLAKLIK_AZALT or
89.           in_islem = KARSITLIK_ARTIR or in_islem = KARSITLIK_AZALT then
90.             r_addr<= conv_std_logic_vector(n_i * IMGE_SUTUN +  n_j ,
   r_addr'length);
91.           end if;
92.           r_addr_vld <= '1';
93.           r_Imge_Isleme <= OKUMA_BEKLE;
94.
95.         when OKUMA_BEKLE =>
96.           if in_data_vld = '1' then
97.             r_data <= in_data;
98.             r_Imge_Isleme <= ISLEM_YAP;
99.           end if;
100.        when ISLEM_YAP =>
101.          if in_islem = AYNALAMA or in_islem = TERS_CEVIRME then
102.            r_data <= r_data;
103.          elsif in_islem = NEGATIFLEME then
104.            r_data <= 255 - r_data;
105.          elsif in_islem = ESIKLEME then
106.            if r_data > 128 then
107.              r_data <= conv_std_logic_vector(255, r_data'length);
108.            else
109.              r_data <= (others => '0');
110.            end if;
111.          elsif in_islem = PARLAKLIK_ARTIR then
112.            if r_data > 210 then
113.              r_data <= conv_std_logic_vector(255, r_data'length);
114.            else
115.              r_data <= r_data + 45;
116.            end if;
117.          elsif in_islem = PARLAKLIK_AZALT then
118.            if r_data < 45 then
```

```
119.              r_data <= conv_std_logic_vector(0, r_data'length);
120.            else
121.              r_data <= r_data - 45;
122.            end if;
123.          elsif in_islem = KARSITLIK_ARTIR then
124.            if r_data > 128 then
125.              r_data <= conv_std_logic_vector(255, r_data'length);
126.            else
127.              r_data <= r_data(6 downto 0) & '0';
128.            end if;
129.          elsif in_islem = KARSITLIK_AZALT then
130.            r_data <= '0' & r_data(7 downto 1);
131.          end if;
132.          r_data_vld <= '1';
133.          r_Imge_Isleme <= SAYAC_KONT;
134.        when SAYAC_KONT =>
135.          if n_j = IMGE_SUTUN - 1 then
136.            n_j <= 0;
137.            if n_i = IMGE_SATIR - 1 then
138.              n_i <= 0;
139.              r_Imge_Isleme <= TAMAM;
140.            else
141.              n_i <= n_i + 1;
142.              r_Imge_Isleme <= RAMDAN_OKU;
143.            end if;
144.          else
145.            n_j <= n_j + 1;
146.            r_Imge_Isleme <= RAMDAN_OKU;
147.          end if;
148.        when TAMAM =>
149.          r_tamam <= '1';
150.          r_Imge_Isleme <= BOSTA;
151.        when others => NULL;
152.      end case;
153.    end if;
154.    end if;
155.    end process;
156.    end Behavioral;
```

temel_imge_isleme varlığının benzetiminin yapılabilmesi için aşağıda **tb_temel_imge_isleme.vhd** VHDL sınama kodu verilmiştir. Kodda dosyadan okunan imge dataları RAM bloğuna yazılmaktadır. Dataların RAM bloğuna yazılma işlemi tamamlandıktan sonra **temel_imge_isleme** varlığı aktif edilemektedir.

temel_imge_isleme varlığında uygulanacak olan algoritmaya göre adres bilgisi üretilir ve o adreste bulunan data RAM üzerinden okunur. RAM üzerinden okunan data **temel_imge_isleme** varlığında uygulanacak algoritmaya göre işlendikten sonra çıkışa aktarılmaktadır. Çıkışa aktarılan data tekrardan dosyaya yazılmaktadır.

Eşikleme algoritmasının eşik değeri 128 olarak belirlenmiştir. Parlaklık artırmak/ için eklenecek/çıkarılacak sayı 45'tir. Karşıtlık artırma işlemi için 2 ile çarpma işlemi yapılmıştır. Karşıtlık azaltma işlemi için ise 0.5 ile çarpma işlemi yapılmıştır.

Yapılan işlemlerden sonra oluşacak görüntüler sırasıyla verilmiştir. Şekil 9-32.a'da orijinal imge, Şekil 9-32.b'de aynalama algoritması uygulanarak elde edilen yeni imge, Şekil 9-32.c'de ters çevirme algoritması uygulanarak elde edilen yeni imge, Şekil 9-32.d'de negatifleme algoritması uygulanarak elde edilen yeni imge, Şekil 9-32.e'de eşikleme algoritması uygulanarak elde edilen yeni imge, Şekil 9-32.f'de parlaklığın artırıldığı yeni imge, Şekil 9-32.g'de parlaklığın azaltıldığı yeni imge, Şekil 9-32.h'de karşıtlığın artırıldığı yeni imge ve Şekil 9-32.i'de karşıtlığın azaltıldığı yeni imge gösterilmiştir.

Şekil 9-32 Temel imge işleme algoritmaları çıktıları (a) Orijinal imge, (b)Aynalama yapılmış imge, (c)Ters çevrilmiş imge, (d) Negatifleme yapılmış imge, (e) Eşikleme yapılmış imge, (f)Parlaklık artırılmış imge, (g) Parlaklık azaltılmış imge, (h) Karşıtlık artırılmış imge, (i) Karşıtlık azaltılmış imge

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3. use IEEE.STD_LOGIC_ARITH.ALL;
4. use IEEE.STD_LOGIC_UNSIGNED.ALL;
5. use std.textio.ALL;
6. use work.ornekler_paket.all;
7.
8. entity tb_temel_imge_isleme is
9. end tb_temel_imge_isleme;
10.
11.architecture Behavioral of tb_temel_imge_isleme is
12.
13.  component temel_imge_isleme
14.  generic(
15.    IMGE_SATIR : integer := 8;
16.    IMGE_SUTUN : integer:= 8;
17.    VERI_UZUNLUGU : integer:= 24
18.  );
19.  port(
20.    in_clk : in std_logic;
21.    in_rst : in std_logic;
22.    in_en  : in std_logic;
23.    in_basla : in std_logic;
24.    in_islem : in std_logic_vector(2 downto 0);
25.    in_data : in std_logic_vector(VERI_UZUNLUGU - 1 downto 0);
26.    in_data_vld : in std_logic;
27.    out_addr : out std_logic_vector(log2_int(IMGE_SATIR * IMGE_SUTUN)
   - 1 downto 0);
28.    out_addr_vld : out std_logic;
29.    out_data : out std_logic_vector(7 downto 0);
30.    out_data_vld : out std_logic;
31.    out_tamam : out std_logic
32.  );
33.  end component;
34.
35.  component block_ram
36.  generic(
37.    VERI_UZUNLUGU : integer := 8;
38.    RAM_DERINLIGI : integer := 110
39.  );
40.  port(in_clk : in std_logic;
```

```
41.     in_rst : in std_logic;
42.     in_ram_aktif : in std_logic;
43.     in_yaz_en : in std_logic;
44.     in_oku_en : in std_logic;
45.     in_data_addr  :  in  std_logic_vector(log2_int(RAM_DERINLIGI)  -  1
    downto 0);
46.     in_data : in std_logic_vector(VERI_UZUNLUGU - 1 downto 0);
47.     out_data : out std_logic_vector(VERI_UZUNLUGU - 1 downto 0);
48.     out_data_vld : out std_logic
49.   );
50.   end component;
51.
52.   constant CLK_PERIOD : time := 20 ns;
53.   constant IMGE_SATIR : integer := 256;
54.   constant IMGE_SUTUN : integer := 256;
55.   constant VERI_UZUNLUGU : integer := 8;
56.   constant VERI_YOLU_OKUMA : string := "C:\cameraman.txt";
57.   constant VERI_YOLU_YAZMA : string := "D:\cameraman_sonuc.txt";
58.
59.   constant AYNALAMA : std_logic_vector(2 downto 0) := "000";
60.   constant TERS_CEVIRME : std_logic_vector(2 downto 0) := "001";
61.   constant NEGATIFLEME : std_logic_vector(2 downto 0) := "010";
62.   constant ESIKLEME : std_logic_vector(2 downto 0) := "011";
63.   constant PARLAKLIK_ARTIR : std_logic_vector(2 downto 0) := "100";
64.   constant PARLAKLIK_AZALT : std_logic_vector(2 downto 0) := "101";
65.   constant KARSITLIK_ARTIR : std_logic_vector(2 downto 0) := "110";
66.   constant KARSITLIK_AZALT : std_logic_vector(2 downto 0) := "111";
67.
68.   type t_Imge_Isleme  is (RAM_OKUMA, RAM_YAZMA, TAMAM);
69.   signal r_Imge_Isleme : t_Imge_Isleme := RAM_YAZMA;
70.
71.   signal in_clk : std_logic := '0';
72.   signal in_rst : std_logic := '0';
73.   signal in_basla : std_logic := '0';
74.   signal in_ram_aktif : std_logic := '1';
75.   signal out_data_vld : std_logic := '0';
76.   signal out_data : std_logic_vector(7 downto 0) := (others => '0');
77.   signal in_ram_data : std_logic_vector(VERI_UZUNLUGU - 1 downto 0) :=
    (others => '0');
78.   signal  in_ram_data_addr  :  std_logic_vector(log2_int(IMGE_SATIR  *
    IMGE_SUTUN) - 1 downto 0) := (others => '0');
```

```
79.  signal out_ram_data : std_logic_vector(VERI_UZUNLUGU - 1 downto 0)
   := (others => '0');
80.  signal   out_data_addr   :   std_logic_vector(log2_int(IMGE_SATIR   *
   IMGE_SUTUN) - 1 downto 0) := (others => '0');
81.  signal out_ram_data_vld : std_logic := '0';
82.  signal in_en : std_logic := '0';
83.  signal in_yaz_en : std_logic := '0';
84.  signal in_oku_en : std_logic := '0';
85.  signal data_sayac : integer := 0;
86.  signal out_data_addr_vld : std_logic := '0';
87.  signal r_imge_isleme_tamam : std_logic := '0';
88.
89.begin
90.
91.  process
92.  begin
93.    in_clk <= '1';
94.    wait for CLK_PERIOD / 2;
95.    in_clk <= '0';
96.    wait for CLK_PERIOD / 2;
97.  end process;
98.
99.  process
100.      begin
101.       in_basla <= '1';
102.       wait for CLK_PERIOD;
103.       in_basla <= '0'; wait;
104.    end process;
105.
106.    process(in_clk)
107.      file dosya_okuma : text open read_mode is VERI_YOLU_OKUMA;
108.      file dosya_yazma : text open write_mode is VERI_YOLU_YAZMA;
109.      variable satir_okuma : line;
110.      variable satir_yazma : line;
111.      variable data_okuma : integer;
112.    begin
113.      if rising_edge(in_clk) then
114.          if out_data_vld = '1' then
115.              write(satir_yazma, conv_integer(out_data));
116.              writeline(dosya_yazma, satir_yazma);
117.          end if;
```

```
118.
119.           case r_Imge_Isleme is
120.               when RAM_YAZMA =>
121.                   if not endfile(dosya_okuma) then
122.                       readline(dosya_okuma, satir_okuma);
123.                       read(satir_okuma, data_okuma);
124.                       in_ram_data      <=      conv_std_logic_vector(
   data_okuma, VERI_UZUNLUGU);
125.                       in_ram_data_addr   <=   conv_std_logic_vector(
   data_sayac, in_ram_data_addr'length);
126.                       in_en <= '0';
127.                       in_yaz_en <= '1';
128.                       data_sayac <= data_sayac + 1;
129.                   else
130.                       r_Imge_Isleme <= RAM_OKUMA;
131.                       in_yaz_en <= '0';
132.                   end if;
133.
134.               when RAM_OKUMA =>
135.                   in_en <= '1';
136.                   in_ram_data_addr <= out_data_addr;
137.                   in_oku_en <= out_data_addr_vld ;
138.                   if r_imge_isleme_tamam = '1' then
139.                       r_Imge_Isleme <= TAMAM;
140.                   end if;
141.
142.               when TAMAM => null;
143.               when others => NULL;
144.           end case;
145.       end if;
146.     end process;
147.
148.     temel_imge_isleme_map : temel_imge_isleme
149.     generic map(
150.         IMGE_SATIR => IMGE_SATIR,
151.         IMGE_SUTUN => IMGE_SUTUN,
152.         VERI_UZUNLUGU => VERI_UZUNLUGU
153.     )
154.     port map(
155.         in_clk => in_clk,
156.         in_rst => in_rst,
```

```
157.          in_en => in_en,
158.          in_basla => in_basla,
159.          in_islem => KARSITLIK_AZALT,
160.          in_data => out_ram_data,
161.          in_data_vld => out_ram_data_vld,
162.          out_addr => out_data_addr,
163.          out_addr_vld => out_data_addr_vld,
164.          out_data => out_data,
165.          out_data_vld => out_data_vld,
166.          out_tamam => r_imge_isleme_tamam
167.      );
168.
169.      block_ram_map : block_ram
170.      generic map(
171.           VERI_UZUNLUGU => VERI_UZUNLUGU,
172.           RAM_DERINLIGI => IMGE_SATIR * IMGE_SUTUN
173.      )
174.      port map(
175.          in_clk => in_clk,
176.          in_rst => in_rst,
177.          in_ram_aktif => in_ram_aktif,
178.          in_yaz_en => in_yaz_en,
179.          in_oku_en => in_oku_en,
180.          in_data_addr => in_ram_data_addr,
181.          in_data => in_ram_data,
182.          out_data => out_ram_data,
183.          out_data_vld => out_ram_data_vld
184.      );
185.      end Behavioral;
```

Aşağıda **Örnek 9.10**'da verilen RAM uygulamasına ek olarak data okuma işleminin geçerli olduğuna dair sinyal üreten çıkış portunun eklendiği **block_ram.vhd** VHDL kodu verilmiştir. **temel_imge_isleme** varlığında RAM'dan tanımlı adresteki data isteme işlemi yapıldıktan sonra, adresten okunan datanın geçerli olması beklenmektedir. Bu işlem 20. satırda tanımlı **out_data_vld** çıkış portundan sağlanmaktadır. Bu port değeri data okuma işlemi gerçekleştiğinde **'1'**, aksi durumlarda **'0'** çıkışı vermektedir.

```
1. library IEEE;
2. use IEEE.STD_LOGIC_1164.ALL;
3. use IEEE.STD_LOGIC_UNSIGNED.ALL;
4. use work.ornekler_paket.all;
5.
```

```
6. entity block_ram is
7.   generic(
8.     VERI_UZUNLUGU : integer := 8;
9.     RAM_DERINLIGI : integer := 110
10.  );
11.  port(
12.    in_clk : in std_logic;
13.    in_rst : in std_logic;
14.    in_ram_aktif : in std_logic;
15.    in_yaz_en : in std_logic;
16.    in_oku_en : in std_logic;
17.    in_data_addr : in std_logic_vector(log2_int(RAM_DERINLIGI) - 1
   downto 0);
18.    in_data : in std_logic_vector(VERI_UZUNLUGU - 1 downto 0);
19.    out_data : out std_logic_vector(VERI_UZUNLUGU - 1 downto 0);
20.    out_data_vld : out std_logic
21.  );
22.end block_ram;
23.
24.architecture Behavioral of block_ram is
25.
26.  type t_BRAM_DATA is array (0 to RAM_DERINLIGI - 1) of
   std_logic_vector(VERI_UZUNLUGU - 1 downto 0) ;
27.  signal r_BRAM_DATA : t_BRAM_DATA := (others =>(others => '0'));
28.
29.begin
30.
31.  process(in_clk, in_rst)
32.  begin
33.    if in_rst = '1' then
34.      r_BRAM_DATA <= (others =>(others => '0'));
35.
36.    elsif rising_edge(in_clk) then
37.      if in_ram_aktif = '1' then
38.        if in_oku_en = '1' then
39.          out_data <= r_BRAM_DATA( conv_integer( in_data_addr));
40.          out_data_vld <= '1';
41.        else
42.          out_data_vld <= '0';
43.        end if;
44.        if in_yaz_en = '1' then
```

```
45.           r_BRAM_DATA(conv_integer(in_data_addr)) <= in_data;
46.         end if;
47.       end if;
48.     end if;
49.   end process;
50.end Behavioral;
```

Aşağıda imgenin dosya işleminin yapıldığı MATLAB kodu verilmiştir.

```
clc, clear all, close all;

imge = imread('cameraman.tif');
dosya = fopen('cameraman.txt', 'w');

[satir sutun] = size(imge);
for n_i = 1 : satir
    for n_j = 1 : sutun
        if n_i == satir && n_j == sutun
            fprintf(dosya, '%d', imge(n_i, n_j));
        else
            fprintf(dosya, '%d\n', imge(n_i, n_j));
        end
    end
end

fclose(dosya);
```

Aşağıda dosyadan imge okuma işleminin yapıldığı MATLAB kodu verilmiştir.

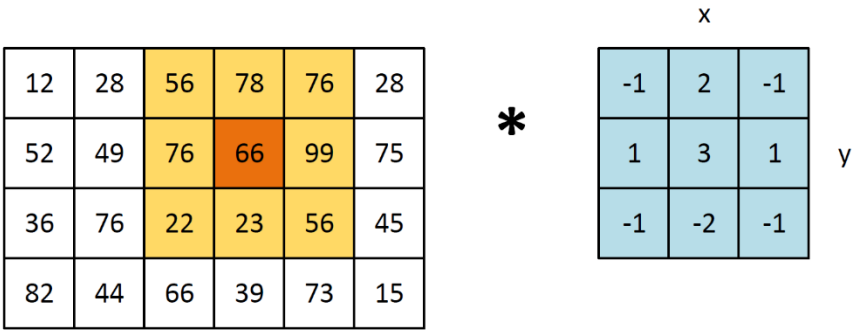
```
clc, clear all, close all;

imge = imread('cameraman.tif');
imshow(imge)
[satir sutun] = size(imge);
dosya = fopen('D:\cameraman_sonuc.txt', 'r');
imge_okunan = fscanf(dosya, '%d')';
COEF = [-1 -2 -1; 0 0 0; 1 2 1];

for n_i = 1 : satir
    for n_j = 1 : sutun
```

```
        yeni_imge(n_i, n_j) = imge_okunan((n_i - 1)* sutun + n_j);
    end
end


fclose(dosya);
figure, imshow(uint8(yeni_imge))
```

9.15. VHDL ile İmge'de Konvolisyon İşlemi

İmge işlemede konvolüsyon işleminde, işlem yapılacak olan pikselin değeri, çevresindeki piksellerin konvolisyon kerneli olarak adlandırılan bir matris ile çarpımıyla elde edilir. Konvolisyon işlemine ilişkin denklem aşağıda verilmiştir.

$$yeni\_imge(x,y) = \sum_{i=0}^{m-1}\sum_{i=0}^{n-1} k(i,j)imge(x-i,y-i) = k * imge$$



Şekil 9-33 Konvolüsyon işlemi

$$\begin{aligned} yeni\_imge(x,y) &= k(0,0)*imge(x+1,y+1)+k(0,1)*imge(x+1,y)+ \\ &k(0,2)*imge(x+1,y-1)+k(1,0)*imge(x,y+1)+k(1,1)*imge(x,y)+ \\ &k(1,2)*imge(x,y-1)+k(2,0)*imge(x-1,y+1)+ \\ &k(2,1)*imge(x-1,y)+k(2,2)*imge(x-1,y-1) \end{aligned}$$

Şekil 9-33'de verilen örnek piksel değeri ve konvolüsyon kerneline ilişkin hesaplama aşağıdaki gibi yapılmatakdır.

$$\begin{aligned} yeni\_imge(x,y) &= (-1)*56+2*23+(-1)*22+1*99+3*66+1*76+ \\ &(-1)*76+(-2)*78+(-1)*56 \\ yeni\_imge(x,y) &= 53 \end{aligned}$$

Örnek 9.15 : Aşağıda imge de konvolüsyon işleminin gerçekleştirildiği **konvolusyon_imge.vhd** VHDL kodu verilmiştir. Kodda RAM üzerinden okunan datalar 3x3 matriste saklanmaktadır. Matriste bulunan datalar konvolusyon işlemine tabi tutularak işlem sonucu elde edilmektedir.

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

```
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use work.konvolusyon_imge_paket.all;

entity konvolusyon_imge is
  port(
    in_clk : in std_logic;
    in_rst : in std_logic;
    in_en  : in std_logic;
    in_basla : in std_logic;
    in_data : in std_logic_vector(VERI_UZUNLUGU - 1 downto 0);
    in_data_vld : in std_logic;
    in_kernel : in std_logic_vector(2 downto 0);
    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 konvolusyon_imge;

architecture Behavioral of konvolusyon_imge is

  type t_Konvolusyon_Imge is (BOSTA, RAMDAN_OKU, OKUMA_BEKLE,
MATRIS_KAYDIR, KONV_HESAPLA, SAYAC_KONT, TAMAM );
  signal r_Konvolusyon_Imge : t_Konvolusyon_Imge := RAMDAN_OKU;

  signal VERI : m_VERI_MATRISI := (others => (others => (others =>
'0')));
  signal Tek_Sutun : v_VERI_DIZISI := (others => (others => '0'));

  signal n_i : integer := 0;
  signal n_j : integer := 0;
  signal n_k : integer := 0;
  signal n_s : 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_bayrak_oku : std_logic := '0';
  signal r_kenar_bulma_tmm : 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, in_en)
  begin
    if in_rst = '1' then
      Tek_Sutun <= (others => (others => '0'));
      n_s <= 0;
      r_bayrak_oku <= '0';
   elsif rising_edge(in_clk) then
      if in_en = '1' then
        if in_data_vld = '1' then
          Tek_Sutun(n_s) <= in_data;
          n_s <= n_s + 1;
          if n_s = 2 then
            n_s <= 0;
            r_bayrak_oku <= '1';
          end if;
        else
          r_bayrak_oku <= '0';
        end if;

      end if;

    end if;
  end process;

  process(in_clk, in_rst)
  begin
    if in_rst = '1' then
      VERI <= (others => (others => (others => '0')));
```

```
      r_Konvolusyon_Imge <= 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';
      n_k <= 0;

    elsif rising_edge(in_clk) then
      if in_en = '1' then
        r_addr_vld <= '0';
        r_data_vld <= '0';
        r_tamam <= '0';
        case r_Konvolusyon_Imge is
          when BOSTA =>
            if in_basla = '1' then
              r_Konvolusyon_Imge <= RAMDAN_OKU;
            end if;

          when RAMDAN_OKU =>
            r_addr <= conv_std_logic_vector((n_i + n_k) * IMGE_SUTUN +
n_j,  r_addr'length);
            r_addr_vld <= '1';
            n_k <= n_k + 1;
            if n_k = 2 then
              r_Konvolusyon_Imge <= OKUMA_BEKLE;
            end if;

          when OKUMA_BEKLE =>
            n_k <= 0;
            if r_bayrak_oku = '1' then
              r_Konvolusyon_Imge <= MATRIS_KAYDIR;
            end if;
          when MATRIS_KAYDIR =>
            n_j <= n_j + 1;
            VERI <= f_Matris_Kaydır(VERI, Tek_Sutun);
            if n_j < 2 then
              r_Konvolusyon_Imge <= RAMDAN_OKU;
            else
```

```
                r_Konvolusyon_Imge <= KONV_HESAPLA;
              end if;

            when KONV_HESAPLA =>
              r_data        <=                f_Konvolusyon_Imge(VERI,
r_KERNEL_LISTE(conv_integer(in_kernel)));
              r_data_vld <= '1';
              r_Konvolusyon_Imge <= SAYAC_KONT;
              if n_j = IMGE_SUTUN then
                n_i <= n_i + 1;
                n_j <= 0;
              end if;

            when SAYAC_KONT =>
              r_data_vld <= '0';
              if n_i < IMGE_SATIR - 2 then
                r_Konvolusyon_Imge <= RAMDAN_OKU;
              else
                n_i <= 0;
                r_Konvolusyon_Imge <= TAMAM;
              end if;

            when TAMAM =>
              r_tamam <= '1';
              r_Konvolusyon_Imge <= BOSTA;
            when others => NULL;
          end case;
        end if;
      end if;
    end process;
end Behavioral;
```

Bu örnekte tip, fonksiyon, sabit vb tanımlama işlemleri **konvolusyon_imge_paket.vhd** paketi içerisinde yapılmıştır. Aşağıda bu pakete ilişkin kod verilmiştir.

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

package konvolusyon_imge_paket is
```

```
  constant IMGE_SATIR : integer := 256;
  constant IMGE_SUTUN : integer := 256;
  constant VERI_UZUNLUGU : integer := 8;

  type v_VERI_DIZISI is array (0 to 2) of
std_logic_vector(VERI_UZUNLUGU - 1 downto 0);
  type m_VERI_MATRISI is array (0 to 2) of v_VERI_DIZISI;

  type v_KERNEL_DIZISI is array (0 to 2) of integer;
  type m_KERNEL_MATRISI is array (0 to 2) of v_KERNEL_DIZISI;

  type t_KERNEL_LISTE is array (0 to 7) of m_KERNEL_MATRISI;
  constant r_KERNEL_LISTE : t_KERNEL_LISTE :=
          (((1, 2, 1), (0, 0, 0), (-1, -2, -1)),
           ((1, 0, -1), (2, 0, -2), (1, 0, -1)),
           ((1, 1, 1), (0, 0, 0), (-1, -1, -1)),
           ((1, 0, -1), (2, 0, -2), (1, 0, -1)),
           ((0, 0, 0), (0, 1, 0), (0, 0, -1)),
           ((0, 1, 0), (1, 0, 1), (0, 1, 0)),
           ((-1, -1, -1), (-1, 8, -1), (-1, -1, -1)),
           ((1, 2, 1), (2, 4, 2), (1, 2, 1)));

  constant YATAY_SOBEL : std_logic_vector(2 downto 0) := "000";
  constant DIKEY_SOBEL : std_logic_vector(2 downto 0) := "001";
  constant YATAY_PREWIT : std_logic_vector(2 downto 0) := "010";
  constant DIKEY_PREWIT : std_logic_vector(2 downto 0) := "011";
  constant KAYDIR_CIKART : std_logic_vector(2 downto 0) := "100";
  constant ALCAK_GECIREN : std_logic_vector(2 downto 0) := "101";
  constant YUKSEK_GECIREN : std_logic_vector(2 downto 0) := "110";
  constant GAUSS : std_logic_vector(2 downto 0) := "111";

  function log2_int(in_giris : integer) return integer;
  function f_Matris_Kaydır(Kernel : m_VERI_MATRISI; Tek_Sutun :
v_VERI_DIZISI) return m_VERI_MATRISI;
  function f_Konvolusyon_Imge(VERI : m_VERI_MATRISI; KERNEL :
m_KERNEL_MATRISI) return std_logic_vector;

end konvolusyon_imge_paket;

package body konvolusyon_imge_paket is
```

```
  function f_Konvolusyon_Imge(VERI : m_VERI_MATRISI; KERNEL : m_KERNEL_MATRISI) return std_logic_vector is
    variable Toplam : integer;
  begin
    Toplam := 0;
    for n_i in 0 to 2 loop
      for n_j in 0 to 2 loop
        Toplam := Toplam +  conv_integer(VERI(n_i)(n_j)) * KERNEL(2 - n_i)(2 - n_j);
      end loop;
    end loop;
    if Toplam > 255 then
      Toplam := 255;
    elsif Toplam < 0 then
      Toplam := 0;
    end if;
    return conv_std_logic_vector(Toplam, 8);
  end f_Konvolusyon_Imge;


  function f_Matris_Kaydır(Kernel : m_VERI_MATRISI; Tek_Sutun : v_VERI_DIZISI) return m_VERI_MATRISI is
    variable Kernel_v : m_VERI_MATRISI;
    variable Tek_Sutun_v : v_VERI_DIZISI;
  begin
    Kernel_v := Kernel;
    Tek_Sutun_v := Tek_Sutun;
    for n_j in 0 to 1 loop
      for n_i in 0 to 2 loop
        Kernel_v(n_i)(n_j) := Kernel_v(n_i)(n_j + 1);
      end loop;
    end loop;
    for n_j in 0 to 2 loop
      Kernel_v(n_j)(2) := Tek_Sutun_v(n_j);
    end loop;
    return Kernel_v;
  end f_Matris_Kaydır;

  function log2_int(in_giris : integer) return integer is
    variable sonuc : integer;
  begin
```

```
    for n_i in 0 to 31 loop
      if (in_giris <= (2 ** n_i)) then
        sonuc := n_i;
        exit;
      end if;
    end loop;
    return sonuc;
  end function;
end package body;
```

konvolusyon_imge varlığının benzetiminin yapılabilmesi için aşağıda **tb_konvolusyon_imge.vhd** VHDL sınama kodu verilmiştir. Kodda dosyadan okunan imge dataları RAM bloğuna yazılmaktadır. Benzetim işlemleri sonuçları Şekil 9-34'de gösterilmiştir.

- Şekil 9-34.a'da orijinal imge,
- Şekil 9-34.b'de yatay Sobel kerneli ile konvolüsyon sonucunda elde edilen yeni imge,
- Şekil 9-34.c'de dikey Sobel kerneli ile konvolüsyon sonucunda elde edilen yeni imge,
- Şekil 9-34.d'de yatay Prewit kerneli ile konvolüsyon sonucunda elde edilen yeni imge,
- Şekil 9-34.e'de dikey Prewit kerneli ile konvolüsyon sonucunda elde edilen yeni imge,
- Şekil 9-34.f'de kaydır ve çıkart kerneli ile konvolüsyon sonucunda elde edilen yeni imge,
- Şekil 9-34.g'de alçak geçiren süzgeç kerneli ile konvolüsyon sonucunda elde edilen yeni imge,
- Şekil 9-34.h'de yüksek geçiren süzgeç kerneli ile konvolüsyon sonucunda elde edilen yeni imge ve
- Şekil 9-34.i'de gauss kerneli ile konvolüsyon sonucunda elde edilen yeni imge gösterilmiştir.

Şekil 9-34 İmge işlemede farklı kerneller için çıktılar (a) Orijinal imge, (b)Yatay Sobel, (c)Dikey Sobel, (d) Yatay Prewit, (e) Dikey Prewit, (f)Kaydır ve Çıkart, (g) Alçak Geçiren, (h) Yüksek Geçiren, (i) Gauss

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

entity tb_konvolusyon_imge is
end tb_konvolusyon_imge;

architecture Behavioral of tb_konvolusyon_imge is

  component konvolusyon_imge
```

```
  port(
    in_clk : in std_logic;
    in_rst : in std_logic;
    in_en  : in std_logic;
    in_basla : in std_logic;
    in_data : in std_logic_vector(VERI_UZUNLUGU - 1 downto 0);
    in_data_vld : in std_logic;
    in_kernel : in std_logic_vector(2 downto 0);
    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 block_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 VERI_YOLU_OKUMA : string := "C:\cameraman.txt";
  constant VERI_YOLU_YAZMA : string := "D:\cameraman_sonuc.txt";
```

```
  type t_Konvolusyon_Imge  is (RAM_OKUMA, RAM_YAZMA, TAMAM);
  signal r_Konvolusyon_Imge : t_Konvolusyon_Imge := 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_Konvolusyon_Imge 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_Konvolusyon_Imge <= 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_Konvolusyon_Imge <= TAMAM;
          end if;

        when TAMAM => null;

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

```
    end if;
  end process;

  konvolusyon_imge_map : konvolusyon_imge
  port map(
    in_clk => in_clk,
    in_rst => in_rst,
    in_en => in_en,
    in_basla => in_basla,
    in_data => out_ram_data,
    in_data_vld => out_ram_data_vld,
    in_kernel => GAUSS,
    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
  );

  block_ram_map : block_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;
```

Blok RAM için Örnek 9.14'te verilen **block_ram.vhd** VHDL dosyasında 4. satırda bulunan paket tanımlama ifadesi aşğıda verilen ifade ile değiştirilmelidir.

```
use work.konvolusyon_imge_paket.all;
```

Aşağıda dosyadan imge okuma işleminin yapıldığı MATLAB kodu verilmiştir.

```
clc, clear all, close all;

imge = imread('cameraman.tif');
imshow(imge)
[satir sutun] = size(imge);
dosya = fopen('D:\cameraman_sonuc.txt', 'r');
imge_okunan = fscanf(dosya, '%d')';
COEF = [-1 -2 -1; 0 0 0; 1 2 1];

for n_i = 1 : satir - 2
    for n_j = 1 : sutun - 2
        yeni_imge(n_i, n_j) = imge_okunan((n_i - 1) * (sutun - 2) +
n_j);

    end
end

fclose(dosya);
figure, imshow(uint8(yeni_imge))
```