# 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.

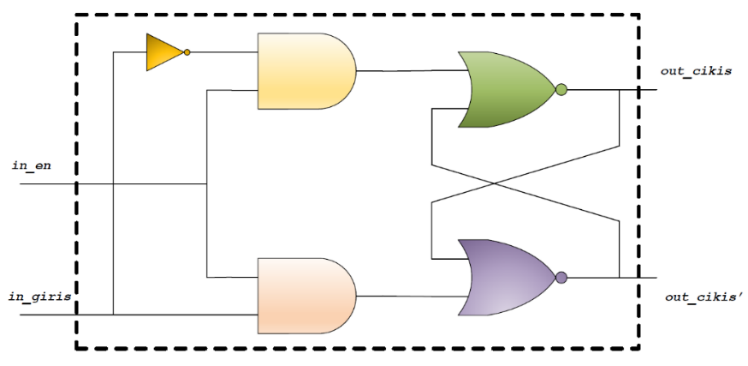
## Yetki Girişli D Mandalı (D Latch)

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 ‑ 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. **entity** d\_latch **is**
4. **port** (
5. in\_en : **in** std\_logic;
6. in\_giris : **in** std\_logic;
7. out\_cikis : **out** std\_logic;
8. out\_cikis\_degil : **out** std\_logic
9. );
10. **end** d\_latch;
11. **architecture** Behavioral **of** d\_latch **is**
12. **signal** r\_cikis : std\_logic := '0';
13. **begin**
14. **process**(in\_en, in\_giris)
15. **begin**
16. **if** in\_en = '1' **then**
17. r\_cikis <= in\_giris;
18. **end** if**;**
19. **end** **process**;
20. out\_cikis <= r\_cikis;
21. out\_cikis\_degil <= (**not** r\_cikis);
22. **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** girişportu 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.

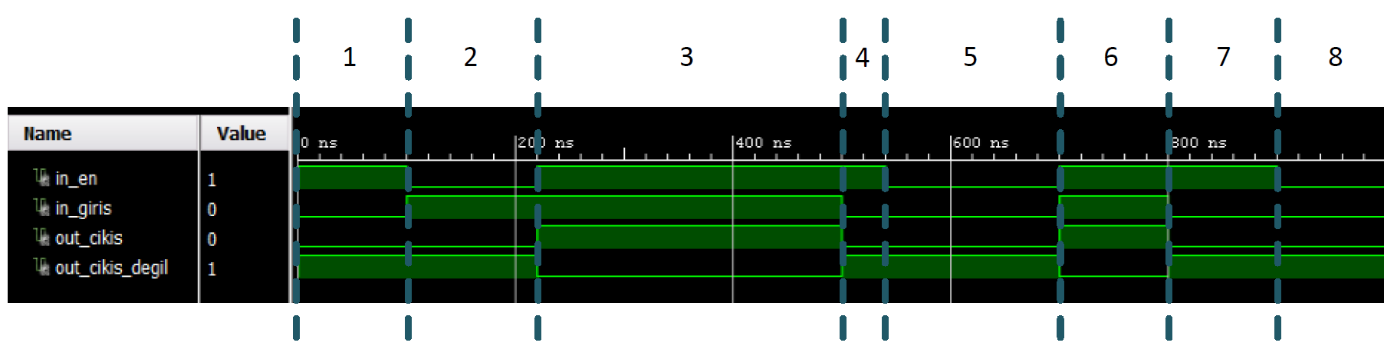
**4. adım:in\_en** giriş portu değeri **'1'** olduğundan **r\_cikis** sinyaline **in\_giris** giriş portu değeri atanmıştır ve **r\_cikis** sinyalinin değeri **'0'** olmaktadır. **r\_cikis** sinyalinin değerinin **'0'** olması ile **out\_cikis** çıkış portuna **'0'**, **out\_cikis\_degil** çıkış portuna **'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 ‑ 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.

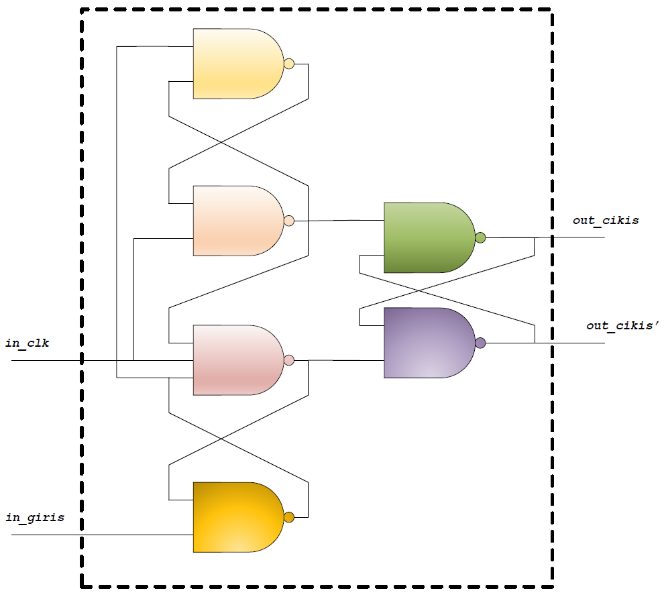
1. **library** IEEE;
2. **use** IEEE.STD\_LOGIC\_1164.all;
3. **entity** tb\_d\_latch **is**
4. **end** tb\_d\_latch;
5. **architecture** Behavioral **of** tb\_d\_latch **is**
6. **component** d\_latch
7. **Port** (
8. in\_en : **in** std\_logic;
9. in\_giris : **in** std\_logic;
10. out\_cikis : **out** std\_logic;
11. out\_cikis\_degil : **out** std\_logic
12. );
13. **end** **component**;
14. **signal** in\_en : std\_logic := '0';
15. **signal** in\_giris : std\_logic := '0';
16. **signal** out\_cikis : std\_logic := '0';
17. **signal** out\_cikis\_degil : std\_logic := '0';
18. **begin**
19. **process**
20. **begin**
21. in\_en <= '1'; **wait** **for** 100 ns;
22. in\_en <= '0'; **wait** **for** 120 ns;
23. in\_en <= '1'; **wait** **for** 320 ns;
24. in\_en <= '0'; **wait** **for** 160 ns;
25. in\_en <= '1'; **wait** **for** 200 ns;
26. in\_en <= '0'; **wait** **for** 100 ns;
27. **end** **process**;
28. **process**
29. **begin**
30. in\_giris <= '0'; **wait** **for** 100 ns;
31. in\_giris <= '1'; **wait** **for** 400 ns;
32. in\_giris <= '0'; **wait** **for** 200 ns;
33. in\_giris <= '1'; **wait** **for** 100 ns;
34. in\_giris <= '0'; **wait** **for** 200 ns;
35. **end** **process**;
36. d\_latch\_map : d\_latch
37. **port** **map**(
38. in\_en => in\_en,
39. in\_giris => in\_giris,
40. out\_cikis => out\_cikis,
41. out\_cikis\_degil => out\_cikis\_degil
42. );
43. **end** Behavioral;

## 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 ‑ 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. **entity** d\_mandali **is**
4. **port** (
5. in\_clk : **in** std\_logic;
6. in\_giris : **in** std\_logic;
7. out\_cikis : **out** std\_logic;
8. out\_cikis\_degil : **out** std\_logic
9. );
10. **end** d\_mandali;
11. **architecture** Behavioral **of** d\_mandali **is**
12. **signal** r\_cikis : std\_logic := '0';
13. **begin**
14. **process**(in\_clk, in\_giris)
15. **begin**
16. **if** in\_clk'event **and** in\_clk = '1' **then**
17. r\_cikis <= in\_giris;
18. **end** **if**;
19. **end** **process**;
20. out\_cikis <= r\_cikis;
21. out\_cikis\_degil <= (**not** r\_cikis);
22. **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

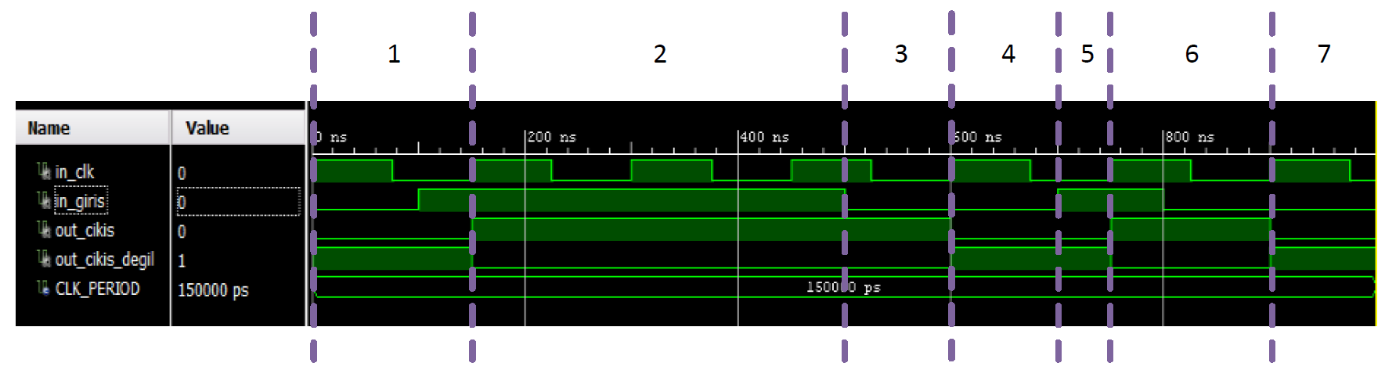
**3. adım:** **in\_giris** giriş portu **'0'** değerini almakta ve **process** aktif olmaktadır. Fakat 22. satırda tanımlanan **in\_clk** giriş portunda saat darbesi yükselen kenarının meydana gelip gelmediğini kontrol eden koşul ifadesi gerçeklenmediğinden dolayı çıkış portlarının değerleri değişmemiştir.

**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 ‑ 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. **entity** tb\_d\_mandali **is**
4. **end** tb\_d\_mandali;
5. **architecture** Behavioral **of** tb\_d\_mandali **is**
6. **component** d\_mandali
7. **Port** (
8. in\_clk : **in** std\_logic;
9. in\_giris : **in** std\_logic;
10. out\_cikis : **out** std\_logic;
11. out\_cikis\_degil : **out** std\_logic
12. );
13. **end** component;
14. **constant** CLK\_PERIOD : time := 150 ns;
15. **signal** in\_clk : std\_logic := '0';
16. **signal** in\_giris : std\_logic := '0';
17. **signal** out\_cikis : std\_logic := '0';
18. **signal** out\_cikis\_degil : std\_logic := '0';
19. **begin**
20. **process**
21. **begin**
22. in\_clk <= '1';
23. **wait** **for** CLK\_PERIOD / 2;
24. in\_clk <= '0';
25. **wait** **for** CLK\_PERIOD / 2;
26. **end** **process**;
27. **process**
28. **begin**
29. in\_giris <= '0'; **wait** **for** 100 ns;
30. in\_giris <= '1'; **wait** **for** 400 ns;
31. in\_giris <= '0'; **wait** **for** 200 ns;
32. in\_giris <= '1'; **wait** **for** 100 ns;
33. in\_giris <= '0'; **wait** **for** 200 ns;
34. **end** **process**;
35. d\_mandali\_map : d\_mandali
36. **Port** **map** (
37. in\_clk => in\_clk,
38. in\_giris => in\_giris,
39. out\_cikis => out\_cikis,
40. out\_cikis\_degil => out\_cikis\_degil
41. );
42. **end** Behavioral;

## Eşzamanlı Olmayan Resetli Yükselen Kenar Tetiklemeli D İki Durumlusu

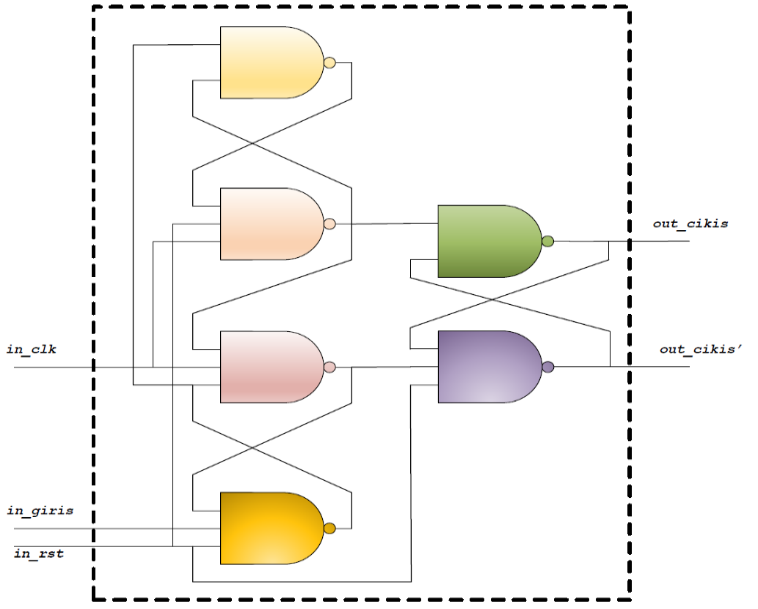
Bu örnekte bir önceki bölümde anlatılan “Yükselen Kenar Tetiklemeli D İki Durumlusu” örneğinden farklı olarak, tasarıma eşzamanlı olmayan bir reset girişi eklenmiştir. Eşzamanlı olmayan reset girişi, tetikleme işaretinden bağımsız olarak, çıkışların sıfırlanmasını sağlamaktadır. Gerçekleştirilen tasarıma 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 ‑ 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. **entity** eszam\_olmayan\_rstli\_d\_mandali **is**
4. **port** (
5. in\_clk : **in** std\_logic;
6. in\_rst : **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** eszam\_olmayan\_rstli\_d\_mandali;
12. **architecture** Behavioral **of** eszam\_olmayan\_rstli\_d\_mandali **is**
13. **signal** r\_cikis : std\_logic := '0';
14. **begin**
15. **process**(in\_clk, in\_rst, in\_giris)
16. **begin**
18. **if** in\_rst = '1' **then**
19. r\_cikis <= '0';
20. **elsif** in\_clk'event **and** in\_clk = '1' **then**
21. r\_cikis <= in\_giris;
22. **end** **if**;
23. **end process;**
24. out\_cikis <= r\_cikis;
25. out\_cikis\_degil <= (**not** r\_cikis);
26. **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 portudeğerinin **'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 portudeğerinin **'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.

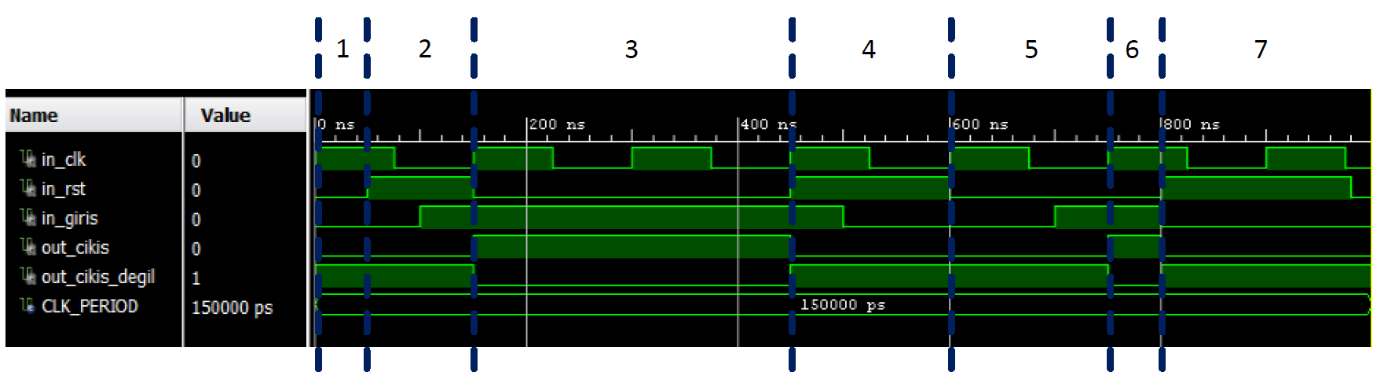
**3. adım:in\_rst**giriş portudeğerinin **'0'** olması ve **in\_clk** giriş portunda meydana gelen yükselen kenar ile birlikte **in\_giris** giriş portu değeri **r\_cikis** sinyaline atanmaktadır ve **r\_cikis** sinyalinin değeri **'1'** olmaktadır. **r\_cikis** sinyalinin değerinin **'1'** olması ile **out\_cikis** çıkış portu değerine **'1'**, **out\_cikis\_degil** çıkış portu değerine ise **'0'** atanmaktadır.

**4. adım:in\_rst**giriş portudeğerinin **'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 portudeğerinin **'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 portudeğerinin **'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 portudeğerinin **'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 ‑ 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. **entity** tb\_eszam\_olmayan\_rstli\_d\_mandali **is**
4. **end** tb\_eszam\_olmayan\_rstli\_d\_mandali;
5. **architecture** Behavioral **of** tb\_eszam\_olmayan\_rstli\_d\_mandali **is**
6. **component** eszam\_olmayan\_rstli\_d\_mandali
7. **Port** (
8. in\_clk : **in** std\_logic;
9. in\_rst : **in** std\_logic;
10. in\_giris : **in** std\_logic;
11. out\_cikis : **out** std\_logic;
12. out\_cikis\_degil : **out** std\_logic
13. );
14. **end component**;
15. **constant** CLK\_PERIOD : time := 150 ns;
16. **signal** in\_clk : std\_logic := '0';
17. **signal** in\_rst : std\_logic := '0';
18. **signal** in\_giris : std\_logic := '0';
19. **signal** out\_cikis : std\_logic := '0';
20. **signal** out\_cikis\_degil : std\_logic := '0';
21. **begin**
22. **process**
23. **begin**
24. in\_clk <= '1';
25. **wait for** CLK\_PERIOD / 2;
26. in\_clk <= '0';
27. **wait for** CLK\_PERIOD / 2;
28. **end process**;
29. **process**
30. **begin**
31. in\_rst <= '0'; **wait for** 50 ns;
32. in\_rst <= '1'; **wait for** 100 ns;
33. in\_rst <= '0'; **wait for** 300 ns;
34. in\_rst <= '1'; **wait for** 150 ns;
35. in\_rst <= '0'; **wait for** 200 ns;
36. in\_rst <= '1'; **wait for** 180 ns;
37. in\_rst <= '0'; **wait for** 20 ns;
38. **end process**;
39. **process**
40. **begin**
41. in\_giris <= '0'; **wait for** 100 ns;
42. in\_giris <= '1'; **wait for** 400 ns;
43. in\_giris <= '0'; **wait for** 200 ns;
44. in\_giris <= '1'; **wait for** 100 ns;
45. in\_giris <= '0'; **wait for** 200 ns;
46. **end process**;
47. eszam\_olmayan\_rstli\_d\_mandali\_map :
48. eszam\_olmayan\_rstli\_d\_mandali **port map**(
49. in\_clk => in\_clk,
50. in\_rst => in\_rst,
51. in\_giris => in\_giris,
52. out\_cikis => out\_cikis,
53. out\_cikis\_degil => out\_cikis\_degil
54. );
55. **end** Behavioral;

## 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. **entity** eszam\_rstli\_d\_mandali **is**
4. **port** (
5. in\_clk : **in** std\_logic;
6. in\_rst : **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** eszam\_rstli\_d\_mandali;
12. **architecture** Behavioral **of** eszam\_rstli\_d\_mandali **is**
13. **signal** r\_cikis : std\_logic := '0';
14. **begin**
15. **process**(in\_clk, in\_rst, in\_giris)
16. **begin**
17. **if** in\_clk'event **and** in\_clk = '1' **then**
18. **if** in\_rst = '1' **then**
19. r\_cikis <= '0';
20. **else**
21. r\_cikis <= in\_giris;
22. **end** **if**;
23. **end** **if**;
24. **end** **process**;
25. out\_cikis <= r\_cikis;
26. out\_cikis\_degil <= (**not** r\_cikis);
27. **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 portudeğerinin **'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.

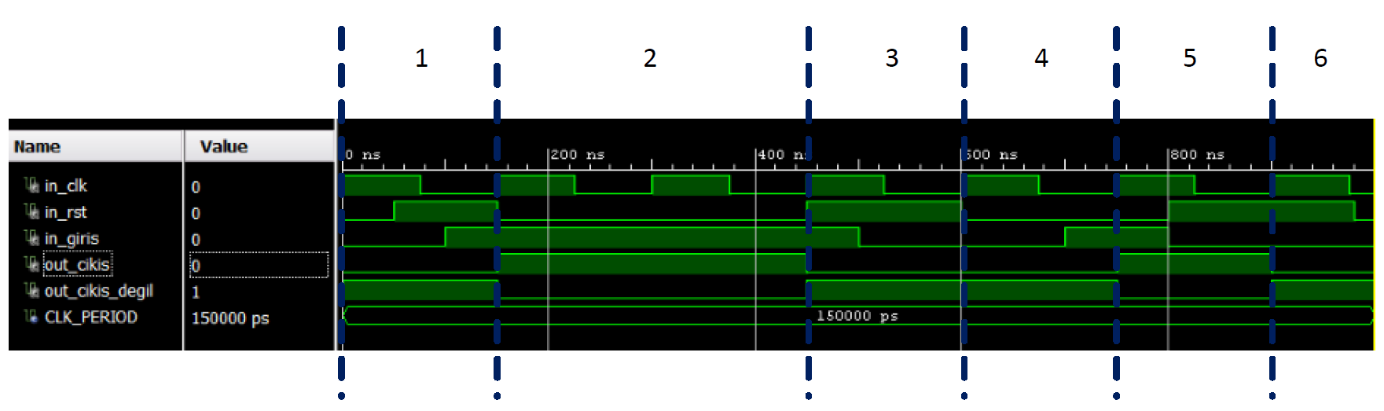
**2. adım:in\_clk** giriş portunun yükselen kenarı ile birlikte**in\_rst**giriş portudeğerinin **'0'** olmasıyla **in\_giris** giriş portu değeri **r\_cikis** sinyaline atanmıştır ve **r\_cikis** sinyalinin değeri **'1'** olmaktadır. **r\_cikis** sinyalinin değerinin **'1'** olması ile **out\_cikis** çıkış portuna **'1'**, **out\_cikis\_degil** çıkış portuna **'0'** değerleri atanmaktadır.

**3. adım:in\_clk** giriş portunun yükselen kenarı ile birlikte**in\_rst**giriş portudeğerinin **'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 portudeğerinin **'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 portudeğerinin **'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 portudeğerinin **'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.



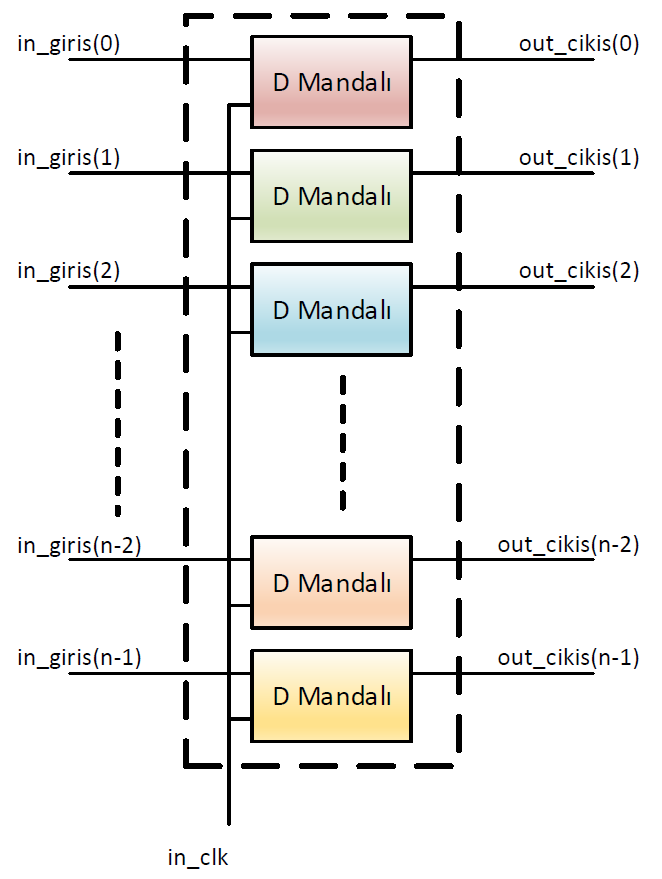
Şekil ‑ eszam\_rstli\_d\_mandali varlığı benzetim çıktısı

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.

1. **component** eszam\_rstli\_d\_mandali
2. **Port** (
3. in\_clk : **in** std\_logic;
4. in\_rst : **in** std\_logic;
5. in\_giris : **in** std\_logic;
6. out\_cikis : **out** std\_logic;
7. out\_cikis\_degil : **out** std\_logic
8. );
9. **end** **component**;
10. eszam\_olmayan\_rstli\_d\_mandali\_map :
11. eszam\_olmayan\_rstli\_d\_mandali **port** **map**(
12. in\_clk => in\_clk,
13. in\_rst => in\_rst,
14. in\_giris => in\_giris,
15. out\_cikis => out\_cikis,
16. out\_cikis\_degil => out\_cikis\_degil
17. );

## 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 ‑ 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. **entity** saklayici\_4\_bit **is**
4. **port** (
5. in\_clk : **in** std\_logic;
6. in\_rst : **in** std\_logic;
7. in\_giris : **in** std\_logic\_vector(**3** downto **0**);
8. out\_cikis : **out** std\_logic\_vector(**3** downto **0**)
9. );
10. **end** saklayici\_4\_bit;
11. **architecture** Behavioral **of** saklayici\_4\_bit **is**
12. **signal** r\_saklayici : std\_logic\_vector(**3** downto **0**);
13. **begin**
14. out\_cikis <= r\_saklayici;
15. **process**(in\_clk, in\_rst, in\_giris)
16. **begin**
17. **if** in\_rst = '1' **then**
18. r\_saklayici <= (**others** => '0');
19. **elsif** rising\_edge(in\_clk) **then**
20. r\_saklayici <= in\_giris;
21. **end** **if**;
22. **end** process;
23. **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 portudeğerinin **'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 portudeğerinin **'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.

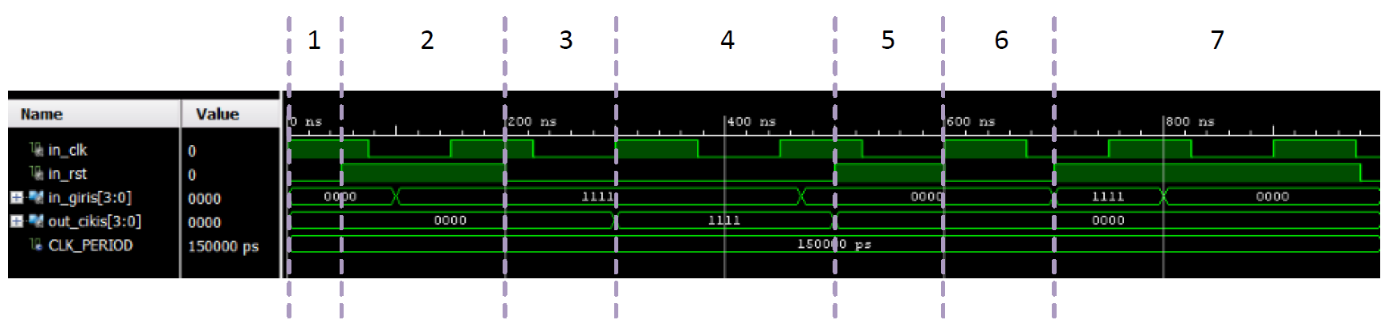
**3. adım :in\_rst**giriş portudeğeri **'0'** olmakta fakat saat darbesi yükselen kenarı meydana gelmemesi nedeniyle **r\_saklayici** sinyalinin değeri değişmemektedir ve bu nedenle **out\_cikis** çıkış portunun değeri de değişmemiştir.

**4. adım :in\_clk**giriş portunun yükselen kenarı ile birlikte**in\_rst**giriş portudeğerinin **'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 portudeğerinin **'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 portudeğerinin **'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 portudeğerinin **'1'** olmasıyla **r\_saklayici** sinyaline **"0000"**değeriatanmaktadır. **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 ‑ saklayici\_4\_bit varlığı benzetim çıktısı

1. **library** IEEE;
2. **use** IEEE.STD\_LOGIC\_1164.ALL;
3. **entity** tb\_saklayici\_4\_bit **is**
4. **end** tb\_saklayici\_4\_bit;
5. **architecture** Behavioral **of** tb\_saklayici\_4\_bit **is**
6. **component** saklayici\_4\_bit
7. **Port** (
8. in\_clk : **in** std\_logic;
9. in\_rst : **in** std\_logic;
10. in\_giris : **in** std\_logic\_vector(**3** downto **0**);
11. out\_cikis : **out** std\_logic\_vector(**3** downto **0**)
12. );
13. **end** **component**;
14. **constant** CLK\_PERIOD : time := 150 ns;
15. **signal** in\_clk : std\_logic := '0';
16. **signal** in\_rst : std\_logic := '0';
17. **signal** in\_giris : std\_logic\_vector(**3** downto **0**) := (**others** => '0');
18. **signal** out\_cikis : std\_logic\_vector(**3** downto **0**) := (**others** => '0');
19. **begin**
20. **process**
21. **begin**
22. in\_clk <= '1';
23. **wait** **for** CLK\_PERIOD / 2;
24. in\_clk <= '0';
25. **wait** **for** CLK\_PERIOD / 2;
26. **end** **process**;
27. **process**
28. **begin**
29. in\_rst <= '0'; **wait** **for** 50 ns;
30. in\_rst <= '1'; **wait** **for** 150 ns;
31. in\_rst <= '0'; **wait** **for** 300 ns;
32. in\_rst <= '1'; **wait** **for** 100 ns;
33. in\_rst <= '0'; **wait** **for** 100 ns;
34. in\_rst <= '1'; **wait** **for** 280 ns;
35. in\_rst <= '0'; **wait** **for** 20 ns;
36. **end** **process**;
37. **process**
38. **begin**
39. in\_giris <= (**others** => '0'); **wait** **for** 100 ns;
40. in\_giris <= (**others** => '1'); **wait** **for** 370 ns;
41. in\_giris <= (**others** => '0'); **wait** **for** 230 ns;
42. in\_giris <= (**others** => '1'); **wait** **for** 100 ns;
43. in\_giris <= (**others** => '0'); **wait** **for** 200 ns;
44. **end** **process**;
45. saklayici\_4\_bit\_map : saklayici\_4\_bit
46. **port** **map**(
47. in\_clk => in\_clk,
48. in\_rst => in\_rst,
49. in\_giris => in\_giris,
50. out\_cikis => out\_cikis
51. );
52. **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. **entity** saklayici\_generic\_aktif\_signal **is**
4. **Generic**(
5. n\_bit : integer := 4
6. );
7. **Port** (
8. in\_clk : **in** std\_logic;
9. in\_rst : **in** std\_logic;
10. in\_en : **in** std\_logic;
11. in\_giris : **in** std\_logic\_vector(**n\_bit - 1** downto **0**);
12. out\_cikis : **out** std\_logic\_vector(**n\_bit - 1** downto **0**)
13. );
14. **end** saklayici\_generic\_aktif\_signal;
15. **architecture** Behavioral **of** saklayici\_generic\_aktif\_signal **is**
16. **signal** r\_saklayici : std\_logic\_vector(**n\_bit - 1** downto **0**);
17. **begin**
18. out\_cikis <= r\_saklayici;
19. **process**(in\_clk, in\_rst, in\_en, in\_giris)
20. **begin**
21. **if** in\_rst = '1' **then**
22. r\_saklayici <= (**others** => '0');
23. **elsif** rising\_edge(in\_clk) **then**
24. **if** in\_en = '1' **then**
25. r\_saklayici <= in\_giris;
26. **end** **if**;
27. **end** **if**;
28. **end** **process**;
29. **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 portunundeğerinin **'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 portunundeğeri **'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 portunundeğerinin **'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.

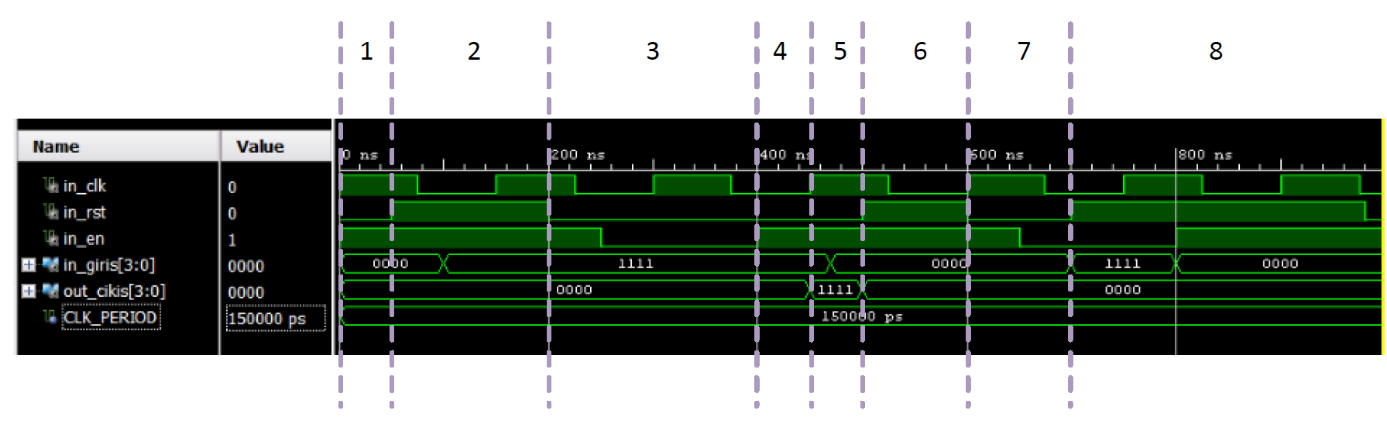
**4. adım :in\_rst**giriş portunundeğerinin **'0'** ve **in\_en** giriş portu değerinin **‘1’** olmasına rağmen **in\_clk**giriş portunun yükselen kenarı meydana gelmemesi nedeniyle **r\_saklayici** sinyalinin değeri değişmemiştir ve bu sebeple çıkış değeri de değişmemiştir.

**5. adım:in\_rst**giriş portunundeğerinin **'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 portunundeğeri **'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 portunundeğerinin **'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 portunundeğeri **'1'** olması nedeniyle **r\_saklayici** değerine **"0000"**atanmaktadır. **r\_saklayici** değerinin **"0000"** olması ile **out\_cikis** çıkış portuna **"0000"** atanmaktadır.

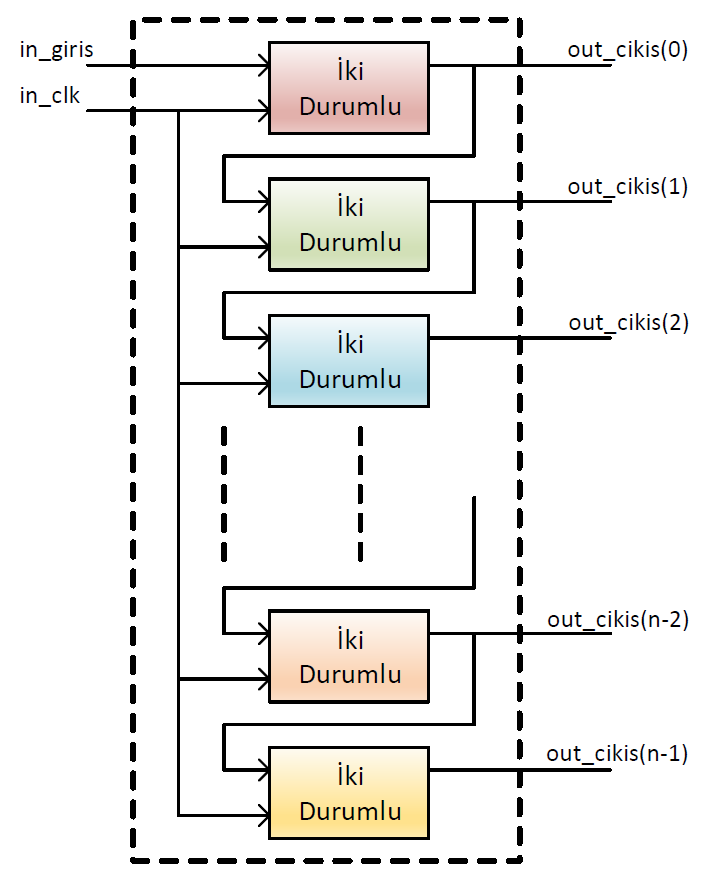


Şekil ‑ saklayici\_generic\_aktif\_signal varlığı benzetim çıktısı

1. **library** IEEE;
2. **use** IEEE.STD\_LOGIC\_1164.ALL;
3. **entity** tb\_saklayici\_generic\_aktif\_signal **is**
4. **end** tb\_saklayici\_generic\_aktif\_signal;
5. **architecture** Behavioral **of** tb\_saklayici\_generic\_aktif\_signal **is**
6. **component** saklayici\_generic\_aktif\_signal
7. **Generic**(
8. n\_bit : integer := 4
9. );
10. **Port** (
11. in\_clk : **in** std\_logic;
12. in\_rst : **in** std\_logic;
13. in\_en : **in** std\_logic;
14. in\_giris : **in** std\_logic\_vector(**n\_bit - 1** downto **0**);
15. out\_cikis : **out** std\_logic\_vector(**n\_bit - 1** downto **0**)
16. );
17. **end** **component**;
18. **constant** CLK\_PERIOD : time := 150 ns;
19. **signal** in\_clk : std\_logic := '0';
20. **signal** in\_rst : std\_logic := '0';
21. **signal** in\_en: std\_logic := '0';
22. **signal** in\_giris : std\_logic\_vector(**3** downto **0**) := (**others** => '0');
23. **signal** out\_cikis : std\_logic\_vector(**3** downto **0**) := (**others** => '0');
24. **begin**
25. **Process**
26. **Begin**
27. in\_clk <= '1';
28. **wait** **for** CLK\_PERIOD / 2;
29. in\_clk <= '0';
30. **wait** **for** CLK\_PERIOD / 2;
31. **end** **process**;
32. **Process**
33. **Begin**
34. in\_rst <= '0'; **wait** **for** 50 ns;
35. in\_rst <= '1'; **wait** **for** 150 ns;
36. in\_rst <= '0'; **wait** **for** 300 ns;
37. in\_rst <= '1'; **wait** **for** 100 ns;
38. in\_rst <= '0'; **wait** **for** 100 ns;
39. in\_rst <= '1'; **wait** **for** 280 ns;
40. in\_rst <= '0'; **wait** **for** 20 ns;
41. **end** **process**;
42. **process**
43. **begin**
44. in\_en <= '1'; **wait** **for** 250 ns;
45. in\_en <= '0'; **wait** **for** 150 ns;
46. in\_en <= '1'; **wait** **for** 250 ns;
47. in\_en <= '0'; **wait** **for** 150 ns;
48. in\_en <= '1'; **wait** **for** 200 ns;
49. **end** **process**;
50. **process**
51. **begin**
52. in\_giris <= (**others** => '0'); **wait** **for** 100 ns;
53. in\_giris <= (**others** => '1'); **wait** **for** 370 ns;
54. in\_giris <= (**others** => '0'); **wait** **for** 230 ns;
55. in\_giris <= (**others** => '1'); **wait** **for** 100 ns;
56. in\_giris <= (**others** => '0'); **wait** **for** 200 ns;
57. **end** **process**;
58. saklayici\_generic\_aktif\_signal\_map :
59. saklayici\_generic\_aktif\_signal
60. **generic** **map**(
61. n\_bit => 4
62. )
63. **port** **map**(
64. in\_clk => in\_clk,
65. in\_rst => in\_rst,
66. in\_en => in\_en,
67. in\_giris => in\_giris,
68. out\_cikis => out\_cikis
69. );
70. **end** Behavioral;

## 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 ‑ 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. **entity** kaydirmali\_saklayici **is**
4. **Port** (
5. in\_clk : **in** std\_logic;
6. in\_rst : **in** std\_logic;
7. in\_giris : **in** std\_logic;
8. out\_cikis : **out** std\_logic\_vector(**3** downto **0**)
9. );
10. **end** kaydirmali\_saklayici;
11. **architecture** Behavioral **of** kaydirmali\_saklayici **is**
12. **signal** r\_saklayici : std\_logic\_vector(**3** downto **0**) := (**others** => ‘0’);
13. **begin**
14. out\_cikis <= r\_saklayici;
15. **process**(in\_clk, in\_rst, in\_giris)
16. **begin**
17. **if** in\_rst = '1' **then**
18. r\_saklayici <= (**others** => '0');
19. **elsif** rising\_edge(in\_clk) **then**
20. r\_saklayici(3) <= in\_giris ;
21. r\_saklayici(2) <= r\_saklayici(3) ;
22. r\_saklayici(1) <= r\_saklayici(2) ;
23. r\_saklayici(0) <= r\_saklayici(1) ;
24. **end** **if**;
25. **end** **process**;
26. **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 portunundeğerinin **'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 portunundeğerinin **'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 portunundeğerinin **'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.

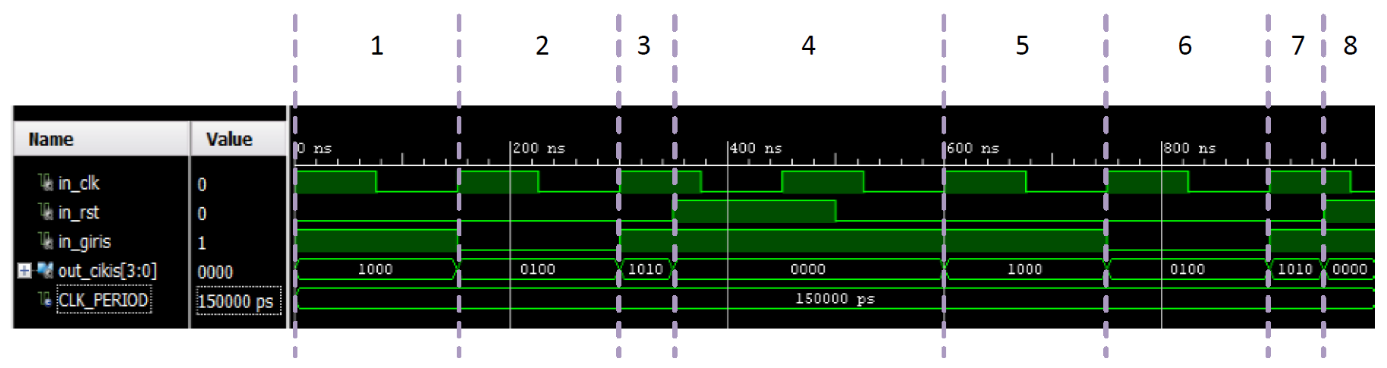
**4. adım :in\_rst**giriş portunundeğeri **'1'** olması nedeniyle **r\_saklayici** değerine **"0000"**atanmaktadır. **r\_saklayici** değerinin **"0000"** olması ile **out\_cikis** çıkış portuna **"0000"** değeri atanmaktadır.

**5. adım *:* in\_rst**giriş portunundeğerinin **'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 portunundeğerinin **'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 portunundeğerinin **'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 portunundeğeri **'1'** olması nedeniyle **r\_saklayici** değerine **"0000"**atanmaktadır. **r\_saklayici** değerinin **"0000"** olması ile **out\_cikis** çıkış portuna **"0000"** değeri atanmaktadır.



Şekil ‑ signal veri nesnesi kullanılan kaydirmali\_saklayici varlığı benzetim çıktısı

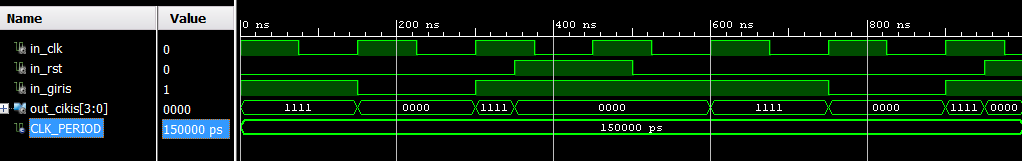
1. **library** IEEE;
2. **use** IEEE.STD\_LOGIC\_1164.ALL;
3. **entity** tb\_kaydirmali\_saklayici **is**
4. **end** tb\_kaydirmali\_saklayici;
5. **architecture** Behavioral **of** tb\_kaydirmali\_saklayici **is**
6. **component** kaydirmali\_saklayici
7. **Port** (
8. in\_clk : **in** std\_logic;
9. in\_rst : **in** std\_logic;
10. in\_giris : **in** std\_logic;
11. out\_cikis : **out** std\_logic\_vector(**3** downto **0**)
12. );
13. **end** **component**;
14. **constant** CLK\_PERIOD : time := 150 ns;
15. **signal** in\_clk : std\_logic := '0';
16. **signal** in\_rst : std\_logic := '0';
17. **signal** in\_giris : std\_logic := '0';
18. **signal** out\_cikis : std\_logic\_vector(**3** downto **0**) := (**others** => '0');
19. **begin**
20. **process**
21. **begin**
22. in\_clk <= '1'; **wait** **for** CLK\_PERIOD / 2;
23. in\_clk <= '0'; **wait** **for** CLK\_PERIOD / 2;
24. **end** **process**;
25. **process**
26. **begin**
27. in\_rst <= '0'; **wait** **for** 350 ns;
28. in\_rst <= '1'; **wait** **for** 150 ns;
29. in\_rst <= '0'; **wait** **for** 450 ns;
30. in\_rst <= '1'; **wait** **for** 50 ns;
31. **end** **process**;
32. **process**
33. **begin**
34. in\_giris <= '1'; **wait** **for** CLK\_PERIOD;
35. in\_giris <= '0'; **wait** **for** CLK\_PERIOD;
36. in\_giris <= '1'; **wait** **for** 3 \* CLK\_PERIOD;
37. in\_giris <= '0'; **wait** **for** CLK\_PERIOD;
38. in\_giris <= '1'; **wait** **for** CLK\_PERIOD;
39. in\_giris <= '0'; **wait** **for** CLK\_PERIOD;
40. **end** **process**;
41. kaydirmali\_saklayic\_map : kaydirmali\_saklayici
42. **port** **map**(
43. in\_clk => in\_clk,
44. in\_rst => in\_rst,
45. in\_giris => in\_giris,
46. out\_cikis => out\_cikis
47. );
48. **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. **entity** kaydirmali\_saklayici **is**
4. **Port** (
5. in\_clk : **in** std\_logic;
6. in\_rst : **in** std\_logic;
7. in\_giris : **in** std\_logic;
8. out\_cikis : **out** std\_logic\_vector(**3** downto **0**)
9. );
10. **end** kaydirmali\_saklayici;
11. **architecture** Behavioral **of** kaydirmali\_saklayici **is**
12. **signal** r\_saklayici : std\_logic\_vector(**3** downto **0**) := (**others** => '0');
13. **begin**
14. out\_cikis <= r\_saklayici;
15. **process**(in\_clk, in\_rst, in\_giris)
16. **variable** v\_saklayici : std\_logic\_vector(**3** downto **0**) := (**others** => '0');
17. **begin**
18. **if** in\_rst = '1' **then**
19. v\_saklayici := (**others** => '0');
20. **elsif** rising\_edge(in\_clk) then
21. v\_saklayici(3) := in\_giris ;
22. v\_saklayici(2) := v\_saklayici(3) ;
23. v\_saklayici(1) := v\_saklayici(2) ;
24. v\_saklayici(0) := v\_saklayici(1) ;
25. **end** **if**;
26. r\_saklayici <= v\_saklayici;
27. **end** **process**;
28. **end** Behavioral;



Şekil ‑ 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.

1. v\_saklayici(0) := v\_saklayici(1) ;
2. v\_saklayici(1) := v\_saklayici(2) ;
3. v\_saklayici(2) := v\_saklayici(3) ;
4. 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. **entity** generic\_kaydirmali\_saklayici **is**
4. **Generic**(
5. n\_bit : integer := 4
6. );
7. **Port** (
8. in\_clk : **in** std\_logic;
9. in\_rst : **in** std\_logic;
10. in\_yon : **in** std\_logic;
11. in\_giris : **in** std\_logic;
12. out\_cikis : **out** std\_logic\_vector(**n\_bit - 1** downto **0**)
13. );
14. **end** generic\_kaydirmali\_saklayici;
15. **architecture** Behavioral **of** generic\_kaydirmali\_saklayici **is**
16. **signal** r\_saklayici : std\_logic\_vector(**n\_bit - 1** downto **0**) := (**others** => '0');
17. **function** f\_sola\_kaydir(in\_giris : std\_logic; in\_saklayici : std\_logic\_vector(**n\_bit - 1** downto **0**) )
18. **return** std\_logic\_vector **is**
19. **variable** v\_saklayici : std\_logic\_vector(**n\_bit - 1** downto **0**);
20. **begin**
21. v\_saklayici := in\_saklayici;
22. **for** n\_i **in** **n\_bit - 2** downto **0** **loop**
23. v\_saklayici(n\_i + 1) := v\_saklayici(n\_i);
24. **end** **loop**;
25. v\_saklayici(0) := in\_giris;
26. **return** v\_saklayici;
27. **end** f\_sola\_kaydir;
28. **function** f\_saga\_kaydir(in\_giris : std\_logic; in\_saklayici : std\_logic\_vector(**n\_bit - 1** downto **0**) )
29. **return** std\_logic\_vector **is**
30. **variable** v\_saklayici : std\_logic\_vector(**n\_bit - 1** downto **0**);
31. **begin**
32. v\_saklayici := in\_saklayici;
33. **for** n\_i **in** **1** to **n\_bit - 1** **loop**
34. v\_saklayici(n\_i - 1) := v\_saklayici(n\_i);
35. **end** **loop**;
36. v\_saklayici(n\_bit - 1) := in\_giris;
37. **return** v\_saklayici;
38. **end** f\_saga\_kaydir;
39. **begin**
40. out\_cikis <= r\_saklayici;
41. **process**(in\_clk, in\_rst, in\_giris)
42. **begin**
43. **if** in\_rst = '1' **then**
44. r\_saklayici <= (others => '0');
45. **elsif** rising\_edge(in\_clk) **then**
46. **if** in\_yon = '0' **then**
47. r\_saklayici <= f\_sola\_kaydir(in\_giris, r\_saklayici);
48. **elsif** in\_yon = '1' **then**
49. r\_saklayici <= f\_saga\_kaydir(in\_giris, r\_saklayici);
50. **end** **if**;
51. **end** **if**;
52. **end** **process**;
53. **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 portunundeğerinin **'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 portunundeğerinin **'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 portunundeğerinin **'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.

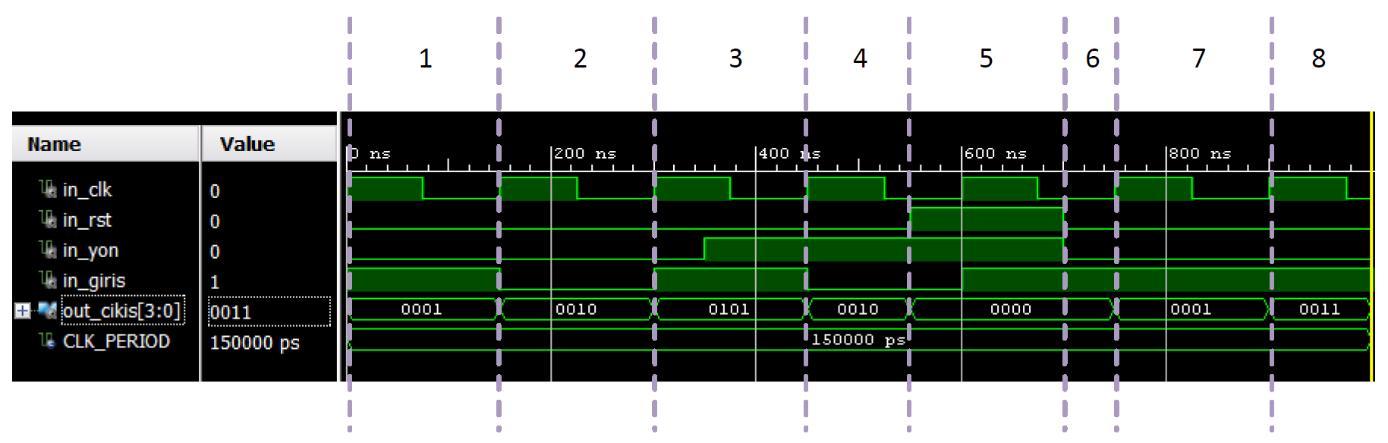
**4. adım :in\_rst**giriş portunundeğerinin **'0'** olması ve **in\_clk**giriş portunun yükselen kenarı ile birlikte **in\_yon** giriş portunun **‘1’** değerini alması ile 58. satırda tanımlı koşul ifadesine bağlı 59. satırda tanımlı **f\_saga\_kaydir** fonksiyon işlemi gerçekleştirilerilmektedir ve 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 portunundeğeri **'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 portunundeğeri **'0'** olmasına rağmen **in\_clk** giriş portunda yükselen kenar meydan gelmediği için **r\_saklayici** sinyalinin değerinde değişiklik olmamıştır.

**7. adım *:* in\_rst**giriş portunundeğerinin **'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 portunundeğerinin **'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 ‑ generic\_kaydirmali\_saklayici varlığı benzetim çıktısı

1. **library** IEEE;
2. **use** IEEE.STD\_LOGIC\_1164.ALL;
3. **entity** tb\_generic\_kaydirmali\_saklayici **is**
4. **end** tb\_generic\_kaydirmali\_saklayici;
5. **architecture** Behavioral **of** tb\_generic\_kaydirmali\_saklayici **is**
6. **component** generic\_kaydirmali\_saklayici
7. **Port** (
8. in\_clk : **in** std\_logic;
9. in\_rst : **in** std\_logic;
10. in\_yon : **in** std\_logic;
11. in\_giris : **in** std\_logic;
12. out\_cikis : **out** std\_logic\_vector(**3** downto **0**) );
13. **end** **component**;
14. **constant** CLK\_PERIOD : time := 150 ns;
15. **signal** in\_clk : std\_logic := '0';
16. **signal** in\_rst : std\_logic := '0';
17. **signal** in\_yon : std\_logic := '0';
18. **signal** in\_giris : std\_logic := '0';
19. **signal** out\_cikis : std\_logic\_vector(**3** downto **0**) := (**others** => '0');
20. **begin**
21. **process**
22. **begin**
23. in\_clk <= '1'; **wait** **for** CLK\_PERIOD / 2;
24. in\_clk <= '0'; **wait** **for** CLK\_PERIOD / 2;
25. **end** **process**;
26. **process**
27. **begin**
28. in\_rst <= '0'; **wait** **for** 550 ns;
29. in\_rst <= '1'; **wait** **for** 150 ns;
30. in\_rst <= '0'; **wait** **for** 300 ns;
31. **end** **process**;
32. **process**
33. **begin**
34. in\_yon <= '0'; **wait** **for** 350 ns;
35. in\_yon <= '1'; **wait** **for** 350 ns;
36. in\_yon <= '0'; **wait** **for** 300 ns;
37. **end** **process**;
38. **process**
39. **begin**
40. in\_giris <= '1'; **wait** **for** CLK\_PERIOD;
41. in\_giris <= '0'; **wait** **for** CLK\_PERIOD;
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. **end** **process**;
47. generic\_kaydirmali\_saklayic\_map : generic\_kaydirmali\_saklayici
48. **port** **map**(
49. in\_clk => in\_clk,
50. in\_rst => in\_rst,
51. in\_yon => in\_yon,
52. in\_giris => in\_giris,
53. out\_cikis => out\_cikis
54. );
55. **end** Behavioral;

## 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:

**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. **entity** sayac\_4\_bit **is**
5. **Port** (
6. in\_clk : **in** std\_logic;
7. in\_rst : **in** std\_logic;
8. out\_cikis : **out** std\_logic\_vector(**3** downto **0**)
9. );
10. **end** sayac\_4\_bit;
11. **architecture** Behavioral **of** sayac\_4\_bit **is**
12. **signal** r\_sayac : std\_logic\_vector(**3** downto **0**) := (**others** => '0');
13. **begin**
14. out\_cikis <= r\_sayac;
15. **process**(in\_clk, in\_rst)
16. **begin**
17. **if** in\_rst = '1' **then**
18. r\_sayac <= (**others** => '0');
19. **elsif** rising\_edge(in\_clk) **then**
20. r\_sayac <= r\_sayac + 1;
21. **end** **if**;
22. **end** **process**;
23. **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 portunundeğerinin **'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 portunundeğerinin **'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 portunundeğerinin **'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.

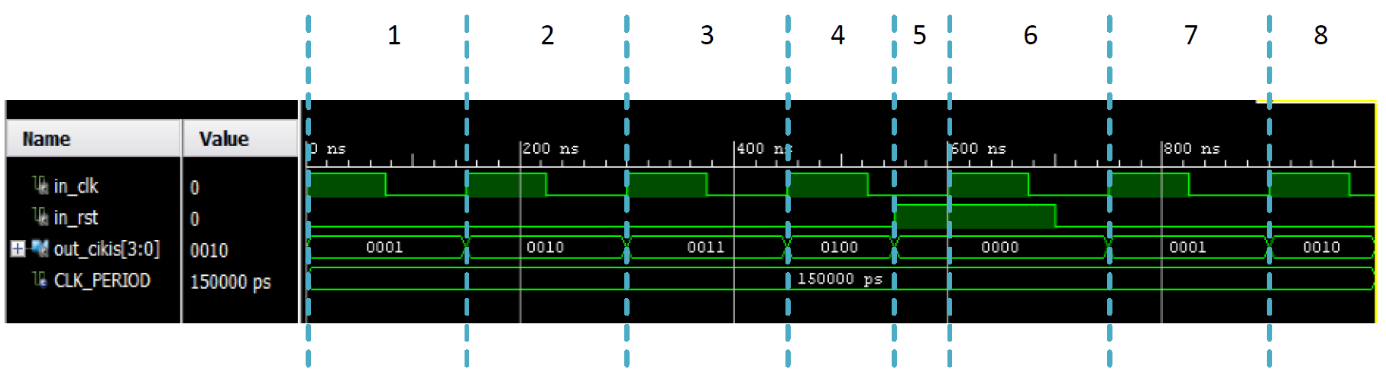
**4. adım :in\_rst**giriş portunundeğerinin **'0'** olması ve **in\_clk**giriş portunun yükselen kenarı ile birlikte **r\_sayac** sinyalinin değeri bir artırılmaktadır ve değeri **"0100"** olmaktadır. **r\_sayac** sinyalinin değerinin **"0100"** olması ile **out\_cikis** çıkış portuna **"0100"** değeri atanmaktadır.

**5. adım :in\_rst**giriş portunundeğeri **'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 portunundeğeri **'0'** olmasına rağmen **in\_clk** giriş portunda yükselen kenar meydan gelmediği için **r\_sayac** sinyalinin değerinde değişiklik olmamıştır.

**7. adım : in\_rst**giriş portunundeğerinin **'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 portunundeğerinin **'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 ‑ sayac\_4\_bit varlığı benzetim çıktısı

1. **library** IEEE;
2. **use** IEEE.STD\_LOGIC\_1164.ALL;
3. **entity** tb\_sayac\_4\_bit **is**
4. **end** tb\_sayac\_4\_bit;
5. **architecture** Behavioral **of** tb\_sayac\_4\_bit **is**
6. **component** sayac\_4\_bit
7. **Port** (
8. in\_clk : **in** std\_logic;
9. in\_rst : **in** std\_logic;
10. out\_cikis : **out** std\_logic\_vector(**3** downto **0**)
11. );
12. **end** **component**;
13. **constant** CLK\_PERIOD : time := 150 ns;
14. **signal** in\_clk : std\_logic := '0';
15. **signal** in\_rst : std\_logic := '0';
16. **signal** out\_cikis : std\_logic\_vector(**3** downto **0**) := (**others** => '0');
17. **begin**
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. **process**
26. **begin**
27. in\_rst <= '0'; **wait** **for** 550 ns;
28. in\_rst <= '1'; **wait** **for** 150 ns;
29. in\_rst <= '0'; **wait** **for** 300 ns;
30. **end** **process**;
31. sayac\_4\_bit\_map : sayac\_4\_bit
32. **port** **map**(
33. in\_clk => in\_clk,
34. in\_rst => in\_rst,
35. out\_cikis => out\_cikis
36. );
37. **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. **entity** generic\_sayac **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\_say : **in** std\_logic;
12. out\_cikis : **out** std\_logic\_vector(n\_bit - 1 downto 0)
13. );
14. **end** generic\_sayac;
15. **architecture** Behavioral **of** generic\_sayac **is**
16. **signal** r\_sayac : std\_logic\_vector(n\_bit - 1 downto 0) := (**others** => '0');
17. **begin**
18. out\_cikis <= r\_sayac;
19. **process**(in\_clk, in\_rst)
20. **begin**
21. **if** in\_rst = '1' **then**
22. r\_sayac <= (**others** => '0');
23. **elsif** rising\_edge(in\_clk) **then**
24. **if** in\_say = '0' **then**
25. r\_sayac <= r\_sayac + 1;
26. **elsif** in\_say = '1' **then**
27. r\_sayac <= r\_sayac - 1;
28. **end** **if**;
29. **end** **if**;
30. **end** **process**;
31. **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 portunundeğerinin **'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 portunundeğerinin **'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 portunundeğeri **'1'** olması nedeniyle **r\_sayac** değerine **"0000"**atanmaktadır. **r\_sayac** değerinin **"0000"** olması ile **out\_cikis** çıkış portuna **"0000"** değeri atanmaktadır.

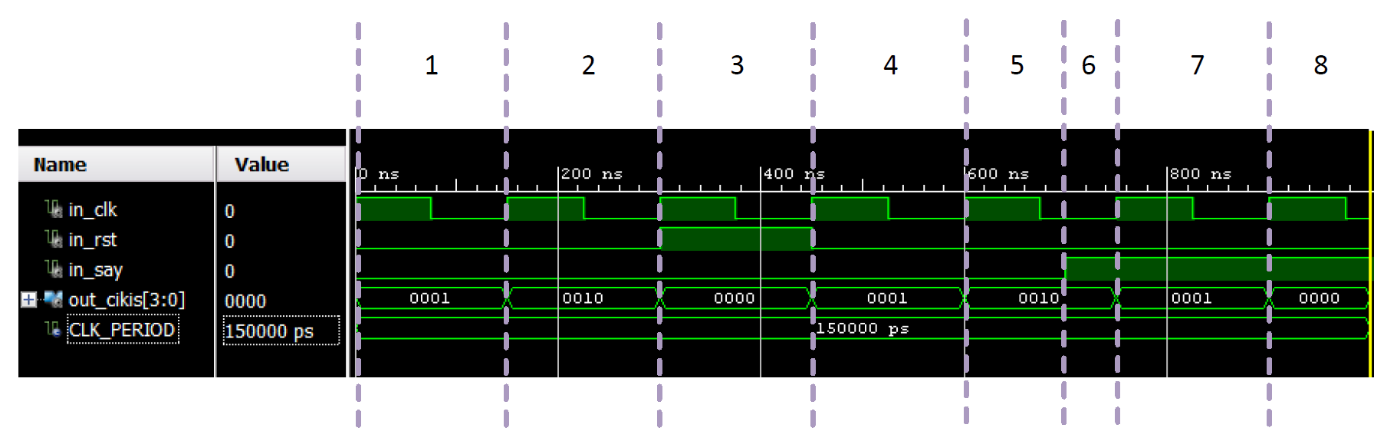
**4. adım :in\_rst**giriş portunundeğerinin **'0'** olması ve **in\_clk**giriş portunun yükselen kenarı ile birlikte **in\_say** giriş portunun **‘0’** değerini alması ile **r\_sayac** sinyalinin değeri bir artırılmaktadır ve değeri **"0001"** olmaktadır. **r\_sayac** değerinin **"0001"** olması ile **out\_cikis** çıkış portuna **"0001"**değeri atanmaktadır.

**5. adım :in\_rst**giriş portunundeğerinin **'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 portunundeğerinin **'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 portunundeğerinin **'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 portunundeğerinin **'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 ‑ sayac\_generic varlığı benzetim çıktısı

1. **library** IEEE;
2. **use** IEEE.STD\_LOGIC\_1164.ALL;
3. **entity** tb\_generic\_sayac **is**
4. **end** tb\_generic\_sayac;
5. **architecture** Behavioral **of** tb\_generic\_sayac **is**
6. **component** generic\_sayac
7. **Generic**(
8. n\_bit : integer := 4
9. );
10. **Port** (
11. in\_clk : **in** std\_logic;
12. in\_rst : **in** std\_logic;
13. in\_say : **in** std\_logic;
14. out\_cikis : **out** std\_logic\_vector(**n\_bit - 1** downto **0**)
15. );
16. **end** **component**;
17. **constant** CLK\_PERIOD : time := 150 ns;
18. **signal** in\_clk : std\_logic := '0';
19. **signal** in\_rst : std\_logic := '0';
20. **signal** in\_say : std\_logic := '0';
21. **signal** out\_cikis : std\_logic\_vector(**3** downto **0**) := (**others** => '0');
22. **begin**
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. **process**
31. **begin**
32. in\_rst <= '0'; **wait** **for** 300 ns;
33. in\_rst <= '1'; **wait** **for** 150 ns;
34. in\_rst <= '0'; **wait** **for** 550 ns;
35. **end** **process**;
36. **process**
37. **begin**
38. in\_say <= '0'; **wait** **for** 700 ns;
39. in\_say <= '1'; **wait** **for** 300 ns;
40. **end** **process**;
41. generic\_sayac\_map : generic\_sayac
42. **port** **map**(
43. in\_clk => in\_clk,
44. in\_rst => in\_rst,
45. in\_say => in\_say,
46. out\_cikis => out\_cikis
47. );
48. **end** Behavioral;

## 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 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. süreyi temsil etmek üzere, frekans ve süre arasındaki ilişki:

Bu durumda sayıcının her sayma işlemi için kadar süre gerekmektedir. Bu durumda Tablo 9‑5’deki 0 ve 1 değişimlerine bakarsak şu yorumları yapabiliriz:

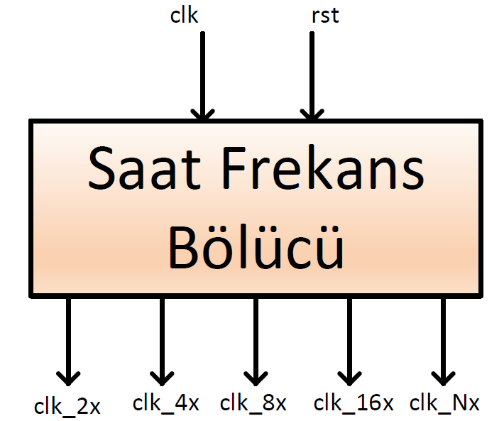
* Bit\_0 için iki yükselen kenar arasındaki süre olmaktadır. ‘0’ için **,** ‘1’ için **.**
* Bit\_1 için iki yükselen kenar arasındaki süre olmaktadır. ‘0’ için **,** ‘1’ için **.**
* Bit\_3 için iki yükselen kenar arasındaki süre olmaktadır. ‘0’ için **,** ‘1’ için **.**

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

* Bit\_0 için :
* Bit\_1 için :
* Bit\_2 için :

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 ‑ 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**, ou**t\_clk\_4**, **out\_clk\_8** ve **out\_clk\_16** çıkış portlarınının 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. **entity** saat\_frekans\_bolucu **is**
5. **generic**(
6. N : integer := 16
7. );
8. **Port** (
9. in\_clk : **in** std\_logic;
10. in\_rst : **in** std\_logic;
11. out\_clk\_2 : **out** std\_logic;
12. out\_clk\_4 : **out** std\_logic;
13. out\_clk\_8 : **out** std\_logic;
14. out\_clk\_16 : **out** std\_logic;
15. out\_clk\_N : **out** std\_logic
16. );
17. **end** saat\_frekans\_bolucu;
18. **architecture** Behavioral **of** saat\_frekans\_bolucu **is**
19. **signal** r\_sayac : std\_logic\_vector(**3** downto **0**) := (**others** => '0');
20. **signal** r\_sayac\_N : integer := 0;
21. **begin**
22. out\_clk\_2 <= r\_sayac(0);
23. out\_clk\_4 <= r\_sayac(1);
24. out\_clk\_8 <= r\_sayac(2);
25. out\_clk\_16 <= r\_sayac(3);
26. out\_clk\_N <= '0' **when** r\_sayac\_N < N / 2 **else** '1';
27. **process**(in\_clk, in\_rst)
28. **begin**
29. **if** in\_rst = '1' **then**
30. r\_sayac <= (**others** => '0');
31. r\_sayac\_N <= 0;
32. **elsif** rising\_edge(in\_clk) **then**
33. r\_sayac <= r\_sayac + 1;
34. **if** r\_sayac\_N = N - 1 **then**
35. r\_sayac\_N <= 0;
36. **else**
37. r\_sayac\_N <= r\_sayac\_N + 1;
38. **end** **if**;
39. **end** **if**;
40. **end** **process**;
41. **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 portunundeğeri **'1'** olması nedeniyle **r\_sayac** ve **r\_sayac\_N** sinyallerinin değerleri sıfırlanmatakdır. Bu nedenle tüm çıkışlara **'0'** değeri atanmatakdır.

**2. adım:in\_rst**giriş portunundeğerinin **'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 portunundeğerinin **'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.

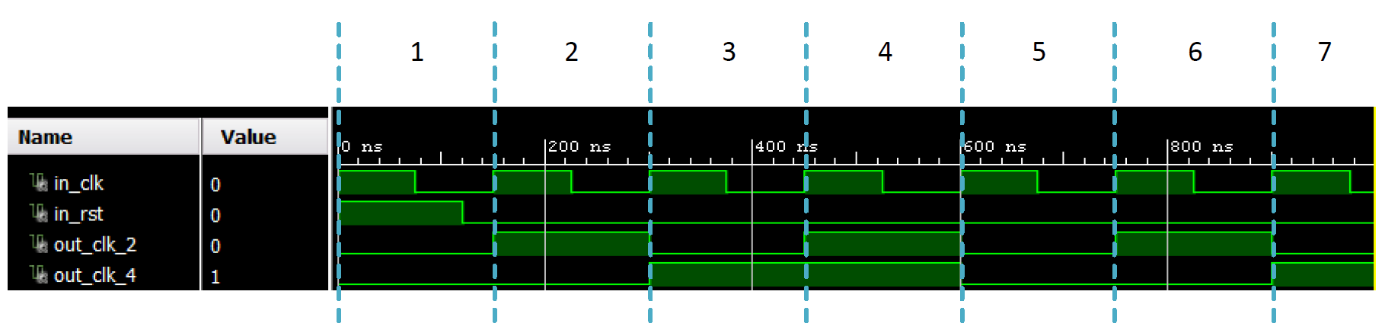
**4. adım:in\_rst**giriş portunundeğerinin **'0'** olması ve **in\_clk**giriş portunun yükselen kenarı ile birlikte **r\_sayac** sinyalinin değeri bir artırılmaktadır ve değeri **"0011"** olmaktadır. Bu nedenle **out\_clk\_2 cikis** çıkış portu değeri **'1'** ve **out\_clk\_4** çıkış portu değeri **'1'** olmaktadır.

**5. adım:in\_rst**giriş portunundeğerinin **'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 portunundeğerinin **'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 portunundeğerinin **'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 ‑ saat\_frekans\_bolucu varlığı benzetim çıktısı-1

Şekil 9‑19’de gösterilen benzetim sonucunda:

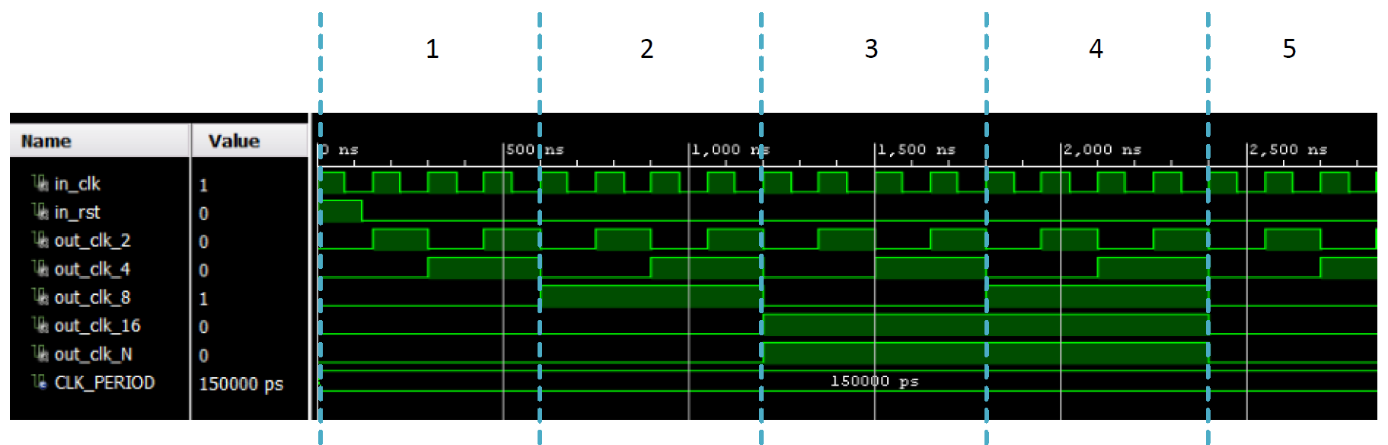
**1. adım:** **r\_sayac** sinyalinin değerleri **“0000”** ile **“0011”** aralığında ve **r\_sayac\_N** sinyalinin değerleri **0** ile **3** aralığında olmaktadır. Bu nedenle **out\_cikis\_8** çıkış portu değeri **'0'**, **out\_cikis\_16** çıkış portu değeri **'0'** ve **out\_cikis\_N** çıkış değeri **'0'** 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 ‑ saat\_frekans\_bolucu varlığı benzetim çıktısı-2

1. **library** IEEE;
2. **use** IEEE.STD\_LOGIC\_1164.ALL;
3. **entity** tb\_saat\_frekans\_bolucu **is**
4. **end** tb\_saat\_frekans\_bolucu;
5. **architecture** Behavioral **of** tb\_saat\_frekans\_bolucu **is**
6. **component** saat\_frekans\_bolucu
7. **generic**(
8. N : integer := 16
9. );
10. **Port** (
11. in\_clk : **in** std\_logic;
12. in\_rst : **in** std\_logic;
13. out\_clk\_2 : **out** std\_logic;
14. out\_clk\_4 : **out** std\_logic;
15. out\_clk\_8 : **out** std\_logic;
16. out\_clk\_16 : **out** std\_logic;
17. out\_clk\_N : **out** std\_logic
18. );
19. **end** **component**;
20. **constant** CLK\_PERIOD : time := 150 ns;
21. **signal** in\_clk : std\_logic := '0';
22. **signal** in\_rst : std\_logic := '0';
23. **signal** out\_clk\_2 : std\_logic := '0';
24. **signal** out\_clk\_4 : std\_logic := '0';
25. **signal** out\_clk\_8 : std\_logic := '0';
26. **signal** out\_clk\_16 : std\_logic := '0';
27. **signal** out\_clk\_N : std\_logic := '0';
28. **begin**
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. **process**
37. **begin**
38. in\_rst <= '1'; **wait** **for** 120 ns;
39. in\_rst <= '0'; **wait**;
40. **end** **process**;
41. saat\_frekans\_bolucu\_map : saat\_frekans\_bolucu
42. **generic** **map**( N => 16 )
43. **port** **map** (
44. in\_clk => in\_clk,
45. in\_rst => in\_rst,
46. out\_clk\_2 => out\_clk\_2,
47. out\_clk\_4 => out\_clk\_4,
48. out\_clk\_8 => out\_clk\_8,
49. out\_clk\_16 => out\_clk\_16,
50. out\_clk\_N => out\_clk\_N
51. );
52. **end** Behavioral;

## 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.

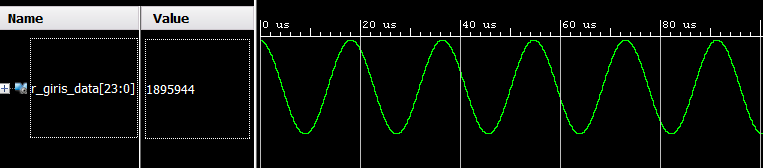
write(satir, data);

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. **entity** dosya\_okuma\_integer **is**
6. **end** dosya\_okuma\_integer;
7. **architecture** Behavioral **of** dosya\_okuma\_integer **is**
9. **constant** CLK\_PERIOD : time := 150 ns;
10. **constant** VERI\_YOLU\_OKUMA : string := "C:\sin.txt";
11. **constant** VERI\_YOLU\_YAZMA : string := "D:\sin.txt
12. **signal** r\_giris\_data : std\_logic\_vector(**23** downto **0**) := (**others**=> '0');
13. **signal** in\_clk : std\_logic := '0';
14. **begin**
16. **process**
17. **begin**
18. in\_clk <= '1';
19. **wait** **for** CLK\_PERIOD / 2;
20. in\_clk <= '0';
21. **wait** **for** CLK\_PERIOD / 2;
22. **end** **process**;
23. **process**(in\_clk)
24. **file** dosya\_okuma: **text** **open** read\_mode **is** VERI\_YOLU\_OKUMA;
25. **file** dosya\_yazma: **text** **open** write\_mode **is** VERI\_YOLU\_YAZMA;
26. **variable** satir\_okuma : line;
27. **variable** satir\_yazma : line;
28. **variable** data : integer;
29. **begin**
30. **if** rising\_edge(in\_clk) **then**
31. **if** **not** endfile(dosya) **then**
32. readline(dosya\_okuma, satir\_okuma);
33. read(satir\_okuma, data);
34. r\_giris\_data <= conv\_std\_logic\_vector(data, r\_giris\_data'length);
35. write(satir\_yazma, data);
36. writeline(dosya\_yazma, satir\_yazma);
37. **end** **if**;
38. **end** **if**;
39. **end** **process**;
40. **end** Behavioral;



Şekil ‑ 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;

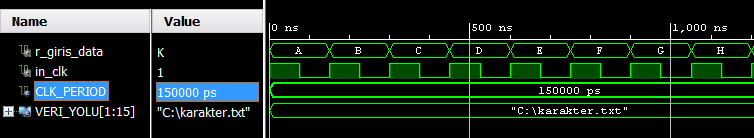
sin\_file = fopen('D:\sin.txt', 'r');

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.

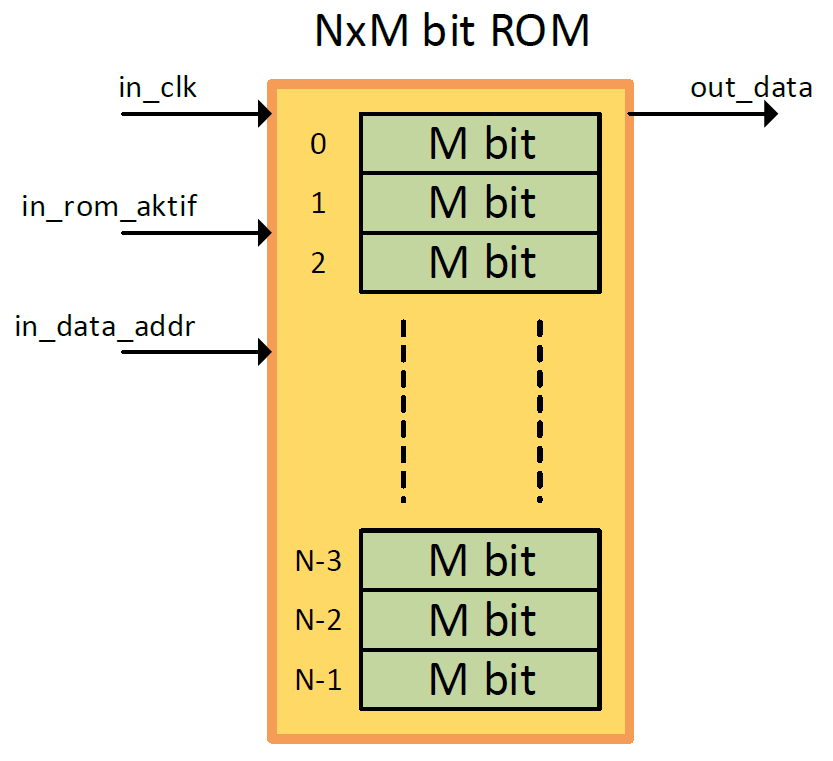


Şekil ‑ dosya\_okuma\_karakter varlığı benzetim çıktısı

1. **library** IEEE;
2. **use** IEEE.STD\_LOGIC\_1164.ALL;
3. **use** std.textio.ALL;
4. **entity** dosya\_okuma\_karakter **is**
5. **end** dosya\_okuma\_karakter;
6. **architecture** Behavioral **of** dosya\_okuma\_karakter **is**
7. **constant** CLK\_PERIOD : time := 150 ns;
8. **constant** VERI\_YOLU : string := "C:\karakter.txt";
9. **signal** r\_giris\_data : character;
10. **signal** in\_clk : std\_logic := '0';
11. **begin**
12. **process**
13. **begin**
14. in\_clk <= '1';
15. **wait** **for** CLK\_PERIOD / 2;
16. in\_clk <= '0';
17. **wait** **for** CLK\_PERIOD / 2;
18. **end** **process**;
19. **process**(in\_clk)
20. **file** dosya : text **open** read\_mode **is** VERI\_YOLU;
21. **variable** satir : line;
22. **variable** data : character;
23. **begin**
24. **if** rising\_edge(in\_clk) **then**
25. **if** **not** endfile(dosya) **then**
26. readline(dosya, satir);
27. read(satir, data);
28. **end** **if**;
29. **end** **if**;
30. r\_giris\_data <= data;
31. **end** **process**;
32. **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 ‑ 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 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. **package** ornekler\_paket **is**
4. **function** log2\_int(in\_giris : integer) **return** integer;
5. **end** ornekler\_paket;
6. **package** **body** ornekler\_paket **is**
7. **function** log2\_int(in\_giris : integer) **return** integer **is**
8. **variable** sonuc : integer;
9. **begin**
10. **for** n\_i **in** 0 to 31 **loop**
11. **if** (in\_giris <= (2 \*\* n\_i)) **then**
12. sonuc := n\_i;
13. **exit**;
14. **end** **if**;
15. **end** **loop**;
16. **return** **sonuc**;
17. **end** log2\_int;
18. **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. **entity** rom **is**
6. **Generic**(
7. ROM\_DERINLIGI : integer := 30;
8. VERI\_UZUNLUGU : integer := 4
9. );
10. **Port** (
11. in\_clk : **in** std\_logic;
12. in\_rom\_aktif : **in** std\_logic;
13. in\_data\_addr : **in** std\_logic\_vector(**log2\_int(ROM\_DERINLIGI) - 1** downto **0**);
14. out\_data : **out** std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**)
15. );
16. **end** rom;
17. **architecture** Behavioral **of** rom **is**
18. **type** t\_ROM\_DATA is array (**0** to **ROM\_DERINLIGI - 1**) of std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**) ;
19. **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" );

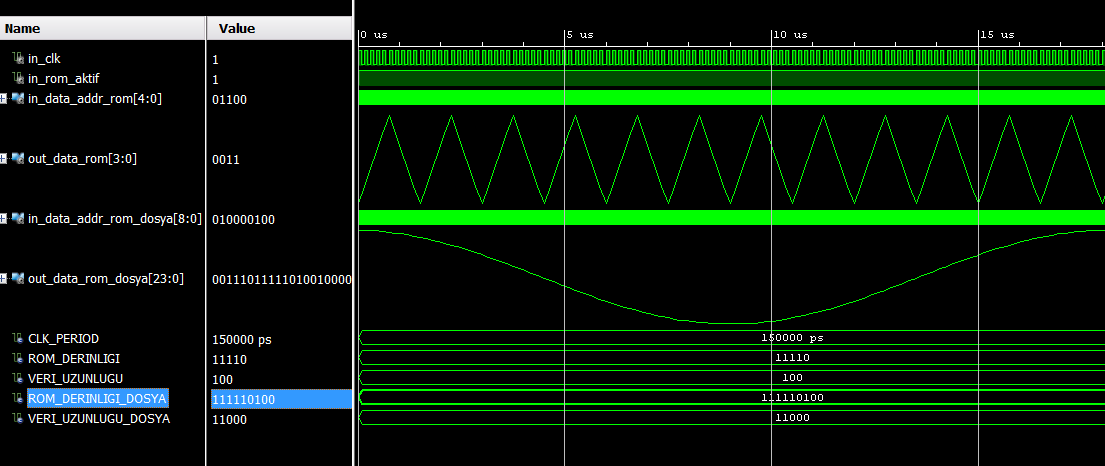
1. **begin**
3. **process**(in\_clk)
4. **begin**
5. **if** rising\_edge(in\_clk) **then**
6. **if** in\_rom\_aktif = '1' **then**
7. out\_data <= r\_ROM\_DATA(conv\_integer(in\_data\_addr));
8. **end** **if**;
9. **end** **if**;
10. **end** **process**;
11. **end** Behavioral;

**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. **entity** rom\_dosya **is**
8. **Generic**(
9. ROM\_DERINLIGI : integer := 500;
10. VERI\_UZUNLUGU : integer := 24
11. );
12. **Port** (
13. in\_clk : **in** std\_logic;
14. in\_rom\_aktif : **in** std\_logic;
15. in\_data\_addr : **in** std\_logic\_vector(**log2\_int(ROM\_DERINLIGI) - 1** downto **0**);
16. out\_data : **out** std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**)
17. );
18. **end** rom\_dosya;
19. **architecture** Behavioral **of** rom\_dosya **is**
20. **constant** VERI\_YOLU : string := "C:\sin.txt";
21. **type** t\_ROM\_DATA **is array** (**0** to **ROM\_DERINLIGI - 1**) of std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**) ;
22. **signal** r\_ROM\_DATA : t\_ROM\_DATA;
23. **procedure** ROM\_DATA\_YUKLE(signal r\_ROM\_DATA : **inout** t\_ROM\_DATA) **is**
24. **file** dosya : text **open** read\_mode **is** VERI\_YOLU;
25. **variable** satir :line;
26. **variable** data : integer;
27. **begin**
28. **for** n\_i **in** **0** to **ROM\_DERINLIGI - 1** **loop**
29. **if** endfile(dosya) **then**
30. **exit**;
31. **else**
32. readline(dosya, satir);
33. read(satir, data);
34. r\_ROM\_DATA(n\_i) <= conv\_std\_logic\_vector(data, VERI\_UZUNLUGU);
35. **end** **if**;
36. **end** **loop**;
37. **end** **procedure**;
38. **begin**
39. ROM\_DATA\_YUKLE(r\_ROM\_DATA);
41. **process**(in\_clk)
42. **begin**
43. **if** rising\_edge(in\_clk) **then**
44. **if** in\_rom\_aktif = '1' **then**
45. out\_data <= r\_ROM\_DATA(conv\_integer(in\_data\_addr));
46. **end** **if**;
47. **end** **if**;
48. **end** **process**;
49. **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.

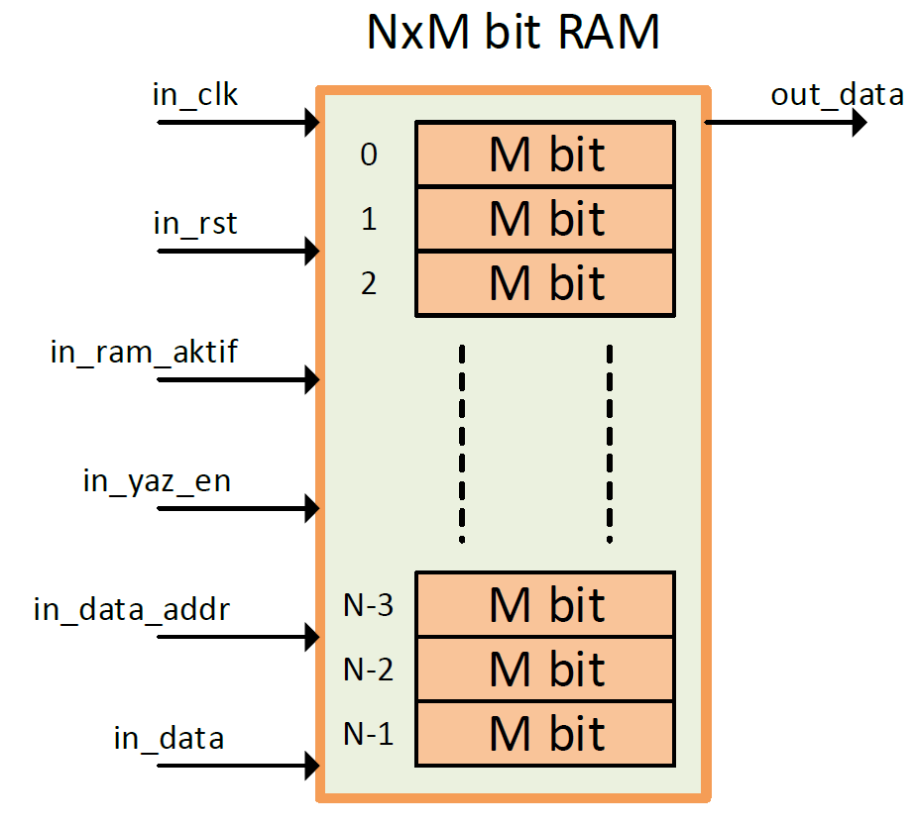
1. **library** IEEE;
2. **use** IEEE.STD\_LOGIC\_1164.ALL;
3. **use** IEEE.STD\_LOGIC\_UNSIGNED.ALL;
4. **use** work.ornekler\_paket.all;
5. **entity** tb\_rom **is**
6. **end** tb\_rom;
7. **architecture** Behavioral **of** tb\_rom **is**
8. **component** rom
9. **Generic**(
10. ROM\_DERINLIGI : integer := 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** **component**;
20. **component** rom\_dosya
21. **Generic**(
22. ROM\_DERINLIGI : integer := 500;
23. VERI\_UZUNLUGU : integer := 24
24. );
25. **Port** (
26. in\_clk : **in** std\_logic;
27. in\_rom\_aktif : **in** std\_logic;
28. in\_data\_addr : **in** std\_logic\_vector(**log2\_int(ROM\_DERINLIGI) - 1** downto **0**);
29. out\_data : **out** std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**)
30. );
31. **end** **component**;
32. **constant** CLK\_PERIOD : time := 150 ns;
33. **constant** ROM\_DERINLIGI : integer := 30;
34. **constant** VERI\_UZUNLUGU : integer := 4;
35. **constant** ROM\_DERINLIGI\_DOSYA : integer := 500;
36. **constant** VERI\_UZUNLUGU\_DOSYA : integer := 24;
38. **signal** in\_clk : std\_logic := '0';
39. **signal** in\_rom\_aktif : std\_logic := '0';
40. **signal** in\_data\_addr\_rom : std\_logic\_vector( **log2\_int(ROM\_DERINLIGI) - 1** downto **0**) := (**others** => '0');
41. **signal** out\_data\_rom : std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**) := (**others** => '0');
42. **signal** in\_data\_addr\_rom\_dosya: std\_logic\_vector( **log2\_int(ROM\_DERINLIGI\_DOSYA) - 1** downto **0**) := (**others** => '0');
43. **signal** out\_data\_rom\_dosya : std\_logic\_vector( **VERI\_UZUNLUGU\_DOSYA - 1** downto 0) := (**others** => '0');
44. **begin**
45. **process**
46. **begin**
47. in\_clk <= '1';
48. **wait** **for** CLK\_PERIOD / 2;
49. in\_clk <= '0';
50. **wait** **for** CLK\_PERIOD / 2;
51. **end** **process**;
52. **process**(in\_clk)
53. **begin**
54. **if** rising\_edge(in\_clk) **then**
55. in\_data\_addr\_rom <= in\_data\_addr\_rom + 1;
56. **if** in\_data\_addr\_rom = ROM\_DERINLIGI - 1 **then**
57. in\_data\_addr\_rom <= (**others** => '0');
58. **end** **if**;
59. in\_data\_addr\_rom\_dosya <= in\_data\_addr\_rom\_dosya + 1;
61. **if** in\_data\_addr\_rom\_dosya = ROM\_DERINLIGI\_DOSYA - 1 **then**
62. in\_data\_addr\_rom\_dosya <= (**others** => '0');
63. **end** **if**;
64. **end** **if**;
65. **end** **process**;
66. in\_rom\_aktif <= '1';
67. rom\_map : rom **generic** **map**(
68. ROM\_DERINLIGI => ROM\_DERINLIGI,
69. VERI\_UZUNLUGU => VERI\_UZUNLUGU )
70. **port** **map** (
71. in\_clk => in\_clk,
72. in\_rom\_aktif => in\_rom\_aktif,
73. in\_data\_addr => in\_data\_addr\_rom,
74. out\_data => out\_data\_rom );
75. rom\_dosya\_map : rom\_dosya **generic** **map**(
76. ROM\_DERINLIGI => ROM\_DERINLIGI\_DOSYA,
77. VERI\_UZUNLUGU => VERI\_UZUNLUGU\_DOSYA )
78. **port** **map** (
79. in\_clk => in\_clk,
80. in\_rom\_aktif => in\_rom\_aktif,
81. in\_data\_addr => in\_data\_addr\_rom\_dosya,
82. out\_data => out\_data\_rom\_dosya);
83. **end** Behavioral;



Şekil ‑ 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 ‑ 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 hesaplamında gösterildiği şekilde 5 bit olacaktır.

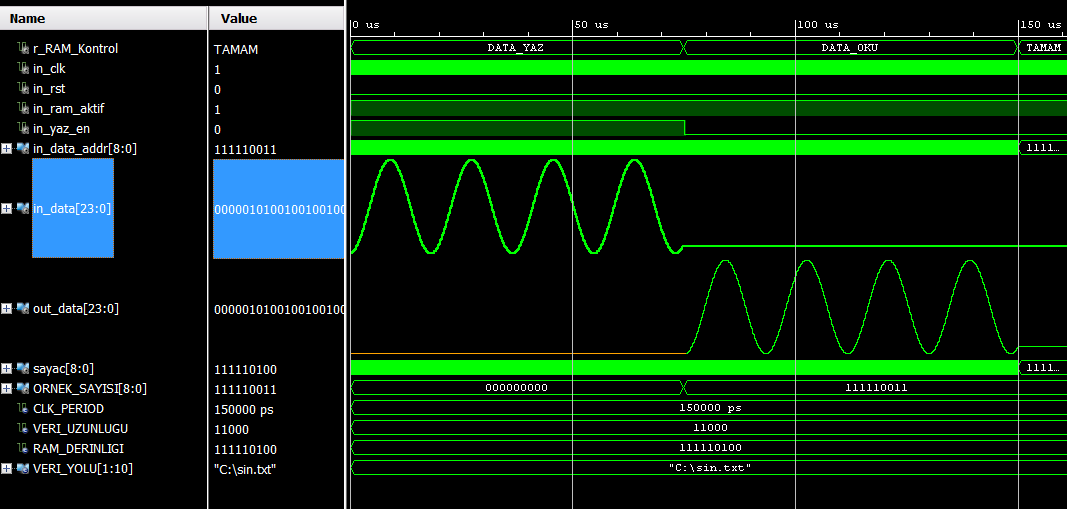
**Örnek 9.11.1:** Şekil 9‑24’de gösterilen blok ram tasarımın yapıldığı **blok\_ram.vhd** VHDL kodu aşağıda verilmiştir. **blok\_ram** varlığımıza ilişkin generic bildirimleri 7-10. satırlarda, port bildirimleri 11-16. satırları arasında yapılmaktıdır.

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. **entity** blok\_ram **is**
6. **generic**(
7. VERI\_UZUNLUGU : integer := 8;
8. RAM\_DERINLIGI : integer := 110
9. );
10. **port**(
11. in\_clk : **in** std\_logic;
12. in\_rst : **in** std\_logic;
13. in\_ram\_aktif : **in** std\_logic;
14. in\_yaz\_en : **in** std\_logic;
15. in\_data\_addr : **in** std\_logic\_vector(**log2\_int(RAM\_DERINLIGI) - 1** downto **0**);
16. in\_data : **in** std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**);
17. out\_data : **out** std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**)
18. );
19. **end** blok\_ram;
20. **architecture** Behavioral **of** blok\_ram **is**
21. **type** t\_BRAM\_DATA **is array** (**0** to **RAM\_DERINLIGI - 1**) **of** std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**) ;
22. **signal** r\_BRAM\_DATA : t\_BRAM\_DATA := (**others** =>(**others** => '0'));
23. **begin**
24. **process**(in\_clk, in\_rst)
25. **begin**
26. **if** in\_rst = '1' **then**
27. r\_BRAM\_DATA <= (**others** =>(**others** => '0'));
28. **elsif** rising\_edge(in\_clk) **then**
29. **if** in\_ram\_aktif = '1' **then**
30. out\_data <= r\_BRAM\_DATA(conv\_integer(in\_data\_addr));
31. **if** in\_yaz\_en = '1' **then**
32. r\_BRAM\_DATA(conv\_integer(in\_data\_addr)) <= in\_data;
33. **end if**;
34. **end if**;
35. **end if**;
36. **end process**;
37. **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).

****

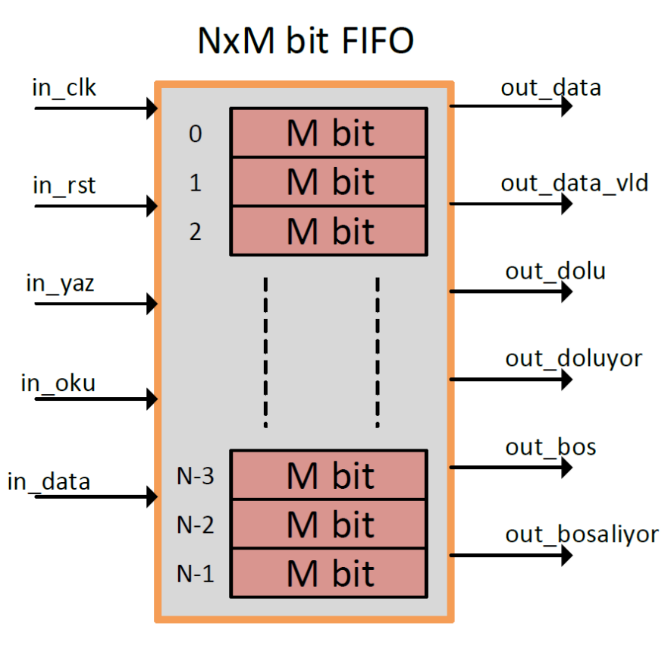
Şekil ‑ block\_ram varlığı benzetim çıktısı

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. **entity** tb\_blok\_ram **is**
8. **end** tb\_blok\_ram;
9. **architecture** Behavioral **of** tb\_blok\_ram **is**
10. **component** blok\_ram
11. **generic**(
12. VERI\_UZUNLUGU : integer := 8;
13. RAM\_DERINLIGI : integer := 110
14. );
15. **port**(
16. in\_clk : **in** std\_logic;
17. in\_rst : **in** std\_logic;
18. in\_ram\_aktif : **in** std\_logic;
19. in\_yaz\_en : **in** std\_logic;
20. in\_data\_addr : **in** std\_logic\_vector(**log2\_int(RAM\_DERINLIGI) - 1** downto **0**);
21. in\_data : **in** std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**);
22. out\_data : **out** std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**)
23. );
24. **end** **component**;
25. **type** t\_RAM\_Kontrol is (DATA\_YAZ, DATA\_OKU, TAMAM);
26. **signal** r\_RAM\_Kontrol : t\_RAM\_Kontrol := DATA\_YAZ;
28. **constant** CLK\_PERIOD : time := 150 ns;
29. **constant** VERI\_UZUNLUGU : integer := 24;
30. **constant** RAM\_DERINLIGI : integer := 500;
31. **constant** VERI\_YOLU : string := "C:\sin.txt";
33. **signal** in\_clk : std\_logic := '0';
34. **signal** in\_rst : std\_logic := '0';
35. **signal** in\_ram\_aktif : std\_logic := '0';
36. **signal** in\_yaz\_en : std\_logic := '0';
37. **signal** in\_data\_addr : std\_logic\_vector(**log2\_int(RAM\_DERINLIGI) - 1** downto **0**) := (**others** => '0');
38. **signal** in\_data : std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**) := (**others** => '0');
39. **signal** out\_data : std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**) := (**others** => '0');
40. **signal** sayac : std\_logic\_vector(**log2\_int(RAM\_DERINLIGI) - 1** downto **0**) := (**others** => '0');
41. **signal** ORNEK\_SAYISI : std\_logic\_vector(**log2\_int(RAM\_DERINLIGI) -** 1 downto **0**) := (**others** => '0');
42. **begin**
43. **process**
44. **begin**
45. in\_clk <= '1';
46. **wait** **for** CLK\_PERIOD / 2;
47. in\_clk <= '0';
48. **wait** **for** CLK\_PERIOD / 2;
49. **end** **process**;
50. **process**(in\_clk)
51. **file** dosya : text **open** read\_mode **is** VERI\_YOLU;
52. **variable** satir : line;
53. **variable** data : integer;
54. **begin**
55. **if** rising\_edge(in\_clk) **then**
56. **case** r\_RAM\_Kontrol **is**
57. **when** DATA\_YAZ =>
58. **if** (**not** endfile(dosya)) **then**
59. readline(dosya, satir);
60. read(satir, data);
61. in\_data <= conv\_std\_logic\_vector(data, VERI\_UZUNLUGU);
62. in\_data\_addr <= sayac;
63. in\_yaz\_en <= '1';
64. in\_ram\_aktif <= '1';
65. **if** sayac = RAM\_DERINLIGI - 1 **then**
66. sayac <= (**others** => '0');
67. ORNEK\_SAYISI <= sayac;
68. r\_RAM\_Kontrol <= DATA\_OKU;
69. **else**
70. sayac <= sayac + 1;
71. **end** **if**;
72. **end if**;
73. **when** DATA\_OKU =>
74. in\_yaz\_en <= '0';
75. in\_data\_addr <= sayac;
76. sayac <= sayac + 1;
77. **if** sayac = ORNEK\_SAYISI **then**
78. r\_RAM\_Kontrol <= TAMAM;
79. **end** **if**;
80. **when** TAMAM =>
81. in\_ram\_aktif <= '0';
83. **when** **others** => **NULL**;
84. **end** **case**;
85. **end** **if**;
86. **end** **process**;
87. blok\_ram\_map : blok\_ram
88. **generic** **map**(
89. VERI\_UZUNLUGU => VERI\_UZUNLUGU,
90. RAM\_DERINLIGI => RAM\_DERINLIGI
91. )
92. **port** **map**(
93. in\_clk => in\_clk,
94. in\_rst => in\_rst,
95. in\_ram\_aktif => in\_ram\_aktif,
96. in\_yaz\_en => in\_yaz\_en,
97. in\_data\_addr => in\_data\_addr ,
98. in\_data => in\_data,
99. out\_data => out\_data
100. );
101. **end** Behavioral;

## 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 ‑ 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.

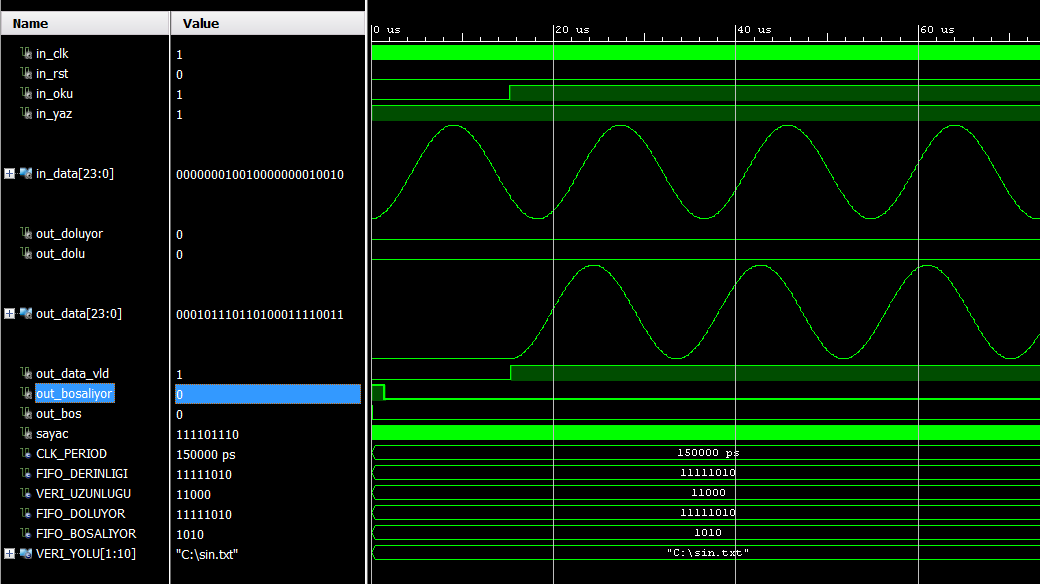
43-50. satırlar arasında ise **r\_fifo\_sayac** sinyalinin her saat darbesinde kontrolü ile FIFO doluluk boşluk durumları belirlenmektedir.

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. **entity** FIFO **is**
5. **Generic**(
6. FIFO\_DERINLIGI : integer := 250;
7. VERI\_UZUNLUGU : integer := 24;
8. FIFO\_DOLUYOR : integer := 250;
9. FIFO\_BOSALIYOR : integer := 10
10. );
11. **Port** (
12. in\_clk : **in** std\_logic;
13. in\_rst : **in** std\_logic;
14. in\_yaz : **in** std\_logic;
15. in\_oku : **in** std\_logic;
16. in\_data : **in** std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**);
17. out\_doluyor : **out** std\_logic;
18. out\_dolu : **out** std\_logic;
19. out\_data : **out** std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**);
20. out\_data\_vld : **out** std\_logic;
21. out\_bosaliyor : **out** std\_logic;
22. out\_bos : **out** std\_logic
23. );
24. **end** FIFO;
25. **architecture** Behavioral **of** FIFO **is**
26. **type** t\_FIFO\_DATA **is** **array** (**0** to **FIFO\_DERINLIGI - 1**) **of** std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**) ;
27. **signal** r\_FIFO\_DATA : t\_FIFO\_DATA := (**others** =>(**others** => '0'));
28. **signal** r\_fifo\_sayac : integer range **-1** to **FIFO\_DERINLIGI + 1** := 0;
29. **signal** ind\_yaz : integer range **0** to **FIFO\_DERINLIGI - 1** := 0;
30. **signal** ind\_oku : integer range **0** to **FIFO\_DERINLIGI - 1** := 0;
31. **signal** bayrak\_dolu : std\_logic := '0';
32. **signal** bayrak\_bos : std\_logic := '0';
33. **signal** r\_data\_vld : std\_logic := '0';
34. **signal** r\_data : std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**) := (**others** => '0');
35. **begin**
36. out\_data <= r\_data;
37. out\_data\_vld <= r\_data\_vld;
38. out\_dolu <= '1' **when** r\_fifo\_sayac = FIFO\_DERINLIGI **else** '0';
39. out\_bos <= '1' **when** r\_fifo\_sayac = 0 **else** '0';
41. bayrak\_dolu <= '1' **when** r\_fifo\_sayac = FIFO\_DERINLIGI **else** '0';
42. bayrak\_bos <= '1' **when** r\_fifo\_sayac = 0 **else** '0';
43. out\_doluyor <= '1' **when** r\_fifo\_sayac > FIFO\_DOLUYOR **else** '0';
44. out\_bosaliyor <= '1' **when** r\_fifo\_sayac < FIFO\_BOSALIYOR **else** '0';
45. **process**(in\_clk, in\_rst)
46. **begin**
47. **if** in\_rst = '1' **then**
48. r\_FIFO\_DATA <= (**others** =>(**others** => '0'));
49. r\_fifo\_sayac <= 0;
50. ind\_yaz <= 0;
51. ind\_oku <= 0;
52. r\_data\_vld <= '0';
53. r\_data <= (**others** => '0');
54. **elsif** rising\_edge(in\_clk) **then**
55. **if** in\_yaz = '1' **and** in\_oku = '0' **then**
56. r\_fifo\_sayac <= r\_fifo\_sayac + 1;
57. **elsif** in\_yaz = '0' **and** in\_yaz = '1' **then**
58. r\_fifo\_sayac <= r\_fifo\_sayac - 1;
59. **end** **if**;
61. **if** in\_yaz = '1' **and** bayrak\_dolu = '0' **then**
62. **if** ind\_yaz = FIFO\_DERINLIGI - 1 **then**
63. ind\_yaz <= 0;
64. **else**
65. ind\_yaz <= ind\_yaz + 1;
66. **end** **if**;
67. **end** **if**;
68. **if** (in\_oku = '1' **and** bayrak\_bos = '0') **then**
69. **if** ind\_oku = FIFO\_DERINLIGI - 1 **then**
70. ind\_oku <= 0;
71. **else**
72. ind\_oku <= ind\_oku + 1;
73. **end** **if;**
74. r\_data <= r\_FIFO\_DATA(ind\_oku);
75. r\_data\_vld <= '1';
76. **else**
77. r\_data\_vld <= '0';
78. **end** **if**;
79. **end** **if**;
80. **if** in\_yaz = '1' **then**
81. r\_FIFO\_DATA(ind\_yaz) <= in\_data;
82. **end** **if**;
83. **end** **process**;
84. **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 ‑ 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. **entity** tb\_fifo **is**
6. **end** tb\_fifo;
7. **architecture** Behavioral **of** tb\_fifo **is**
8. **component** FIFO
9. **Generic**(
10. FIFO\_DERINLIGI : integer := 250;
11. VERI\_UZUNLUGU : integer := 24;
12. FIFO\_DOLUYOR : integer := 250;
13. FIFO\_BOSALIYOR : integer := 10
14. );
15. **Port** (
16. in\_clk : **in** std\_logic;
17. in\_rst : **in** std\_logic;
18. in\_yaz : **in** std\_logic;
19. in\_oku : **in** std\_logic;
20. in\_data : **in** std\_logic\_vector(VERI\_UZUNLUGU - 1 downto 0);
21. out\_doluyor : **out** std\_logic;
22. out\_dolu : **out** std\_logic;
23. out\_data : **out** std\_logic\_vector(VERI\_UZUNLUGU - 1 downto 0);
24. out\_data\_vld : **out** std\_logic;
25. out\_bosaliyor : **out** std\_logic;
26. out\_bos : **out** std\_logic
27. );
28. **end** **component**;
29. **constant** CLK\_PERIOD : time := 150 ns;
30. **constant** FIFO\_DERINLIGI : integer := 250;
31. **constant** VERI\_UZUNLUGU : integer := 24;
32. **constant** FIFO\_DOLUYOR : integer := 250;
33. **constant** FIFO\_BOSALIYOR : integer := 10;
34. **constant** VERI\_YOLU : string := "C:\sin.txt";
35. **signal** in\_clk : std\_logic := '0';
36. **signal** in\_rst : std\_logic := '0';
37. **signal** in\_oku : std\_logic := '0';
38. **signal** in\_yaz : std\_logic := '0';
39. **signal** in\_data : std\_logic\_vector(VERI\_UZUNLUGU - 1 downto 0) := (**others** => '0');
40. **signal** out\_doluyor : std\_logic := '0';
41. **signal** out\_dolu : std\_logic := '0';
42. **signal** out\_data : std\_logic\_vector(VERI\_UZUNLUGU - 1 downto 0) := (**others** => '0');
43. **signal** out\_data\_vld : std\_logic := '0';
44. **signal** out\_bosaliyor : std\_logic := '0';
45. **signal** out\_bos : std\_logic := '0';
46. **signal** sayac : integer := 0;
47. **begin**
48. **process**
49. **begin**
50. in\_clk <= '1';
51. **wait** **for** CLK\_PERIOD / 2;
52. in\_clk <= '0';
53. **wait** **for** CLK\_PERIOD / 2;
54. **end** **process**;
56. **process**(in\_clk)
57. **file** dosya : text **open** read\_mode **is** VERI\_YOLU;
58. **variable** satir : line;
59. **variable** data : integer;
60. **begin**
61. **if** rising\_edge(in\_clk) **then**
62. **if** (**not** endfile(dosya)) **then**
63. readline(dosya, satir);
64. read(satir, data);
65. in\_data <= conv\_std\_logic\_vector(data, VERI\_UZUNLUGU);
66. in\_yaz <= '1';
67. sayac <= sayac + 1;
68. **if** sayac > 100 **then**
69. in\_oku <= '1';
70. **end** **if**;
71. **end** **if**;
72. **end** **if**;
73. **end** **process**;
74. FIFO\_map : FIFO
75. **generic** **map**(
76. FIFO\_DERINLIGI => FIFO\_DERINLIGI,
77. VERI\_UZUNLUGU => VERI\_UZUNLUGU,
78. FIFO\_DOLUYOR => FIFO\_DOLUYOR,
79. FIFO\_BOSALIYOR => FIFO\_BOSALIYOR )
80. **port** **map**(
81. in\_clk => in\_clk,
82. in\_rst => in\_rst,
83. in\_yaz => in\_yaz,
84. in\_oku => in\_oku,
85. in\_data => in\_data,
86. out\_doluyor => out\_doluyor,
87. out\_dolu => out\_dolu,
88. out\_data => out\_data,
89. out\_data\_vld => out\_data\_vld,
90. out\_bosaliyor => out\_bosaliyor,
91. out\_bos => out\_bos
92. );
93. **end** Behavioral;

## Sinyal İşlemede Konvolüsyon

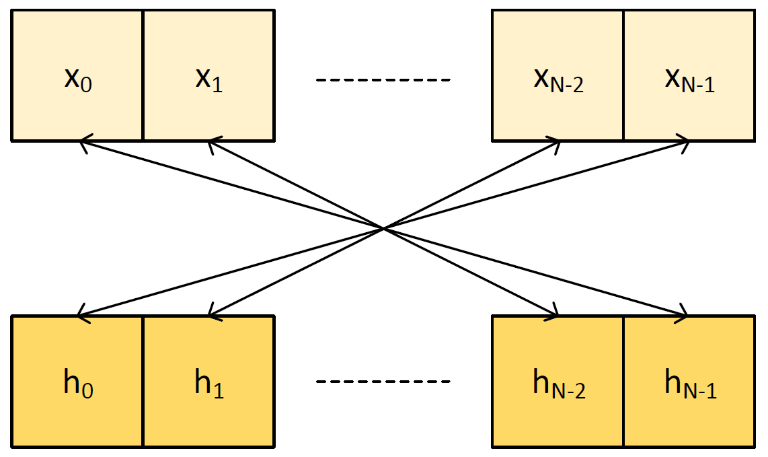
Bu kısma kadar olan örneklerde genelde sayısal tasarımla alakalı uygulamalar gerçekleştirdik. Uygulamalarla VHDL ile ilgili kullanım şekillerini ve tasasrım yollarını göstermeye çalıştık.

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 'in N tane çarpanlı bir sonlu filtre olduğunu varsayarsak, giriş işareti sonsuz uzunlukta olduğu durumda dahi, filtrenin çıkış işareti konvolüsyon işlemi ile aşağıda verilen deknklemde gösterilmiştir.

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 ‑ 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.

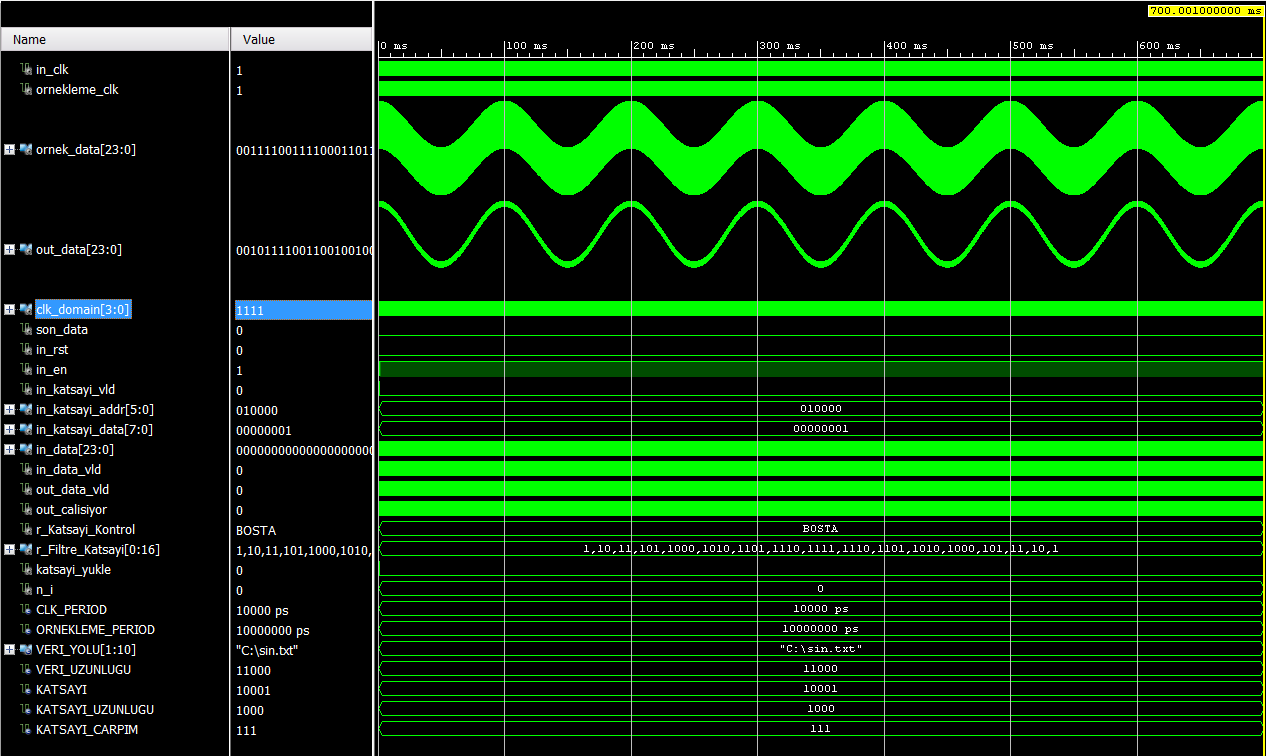
68-77. satırlarda tanımlı **process** ile konvolüsyon katsayılarının belleğe yazılma işlemi yapılmaktadır. Katsayı ve katsayı adres bilgisi ile birlikte aktif sinyali olması durumunda katsayı değeri **r\_katsayi\_bellek** sinyalinde ilgili adresadresteki yerine yazılmaktadır.

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. **entity** konvolusyon\_sinyal **is**
7. **Generic**(
8. VERI\_UZUNLUGU : integer := 24;
9. KATSAYI : integer := 5;
10. KATSAYI\_UZUNLUGU : integer := 8;
11. KATSAYI\_CARPIM : integer := 3
12. );
13. **Port** (
14. in\_clk : **in** std\_logic;
15. in\_rst : **in** std\_logic;
16. in\_en : **in** std\_logic;
17. in\_katsayi\_vld : **in** std\_logic;
18. in\_katsayi\_addr : **in** std\_logic\_vector(**log2\_int(KATSAYI)** downto **0**);
19. in\_katsayi\_data : **in** std\_logic\_vector(**KATSAYI\_UZUNLUGU - 1** downto **0**);
20. in\_data : **in** std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**);
21. in\_data\_vld : **in** std\_logic;
22. out\_data : **out** std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**);
23. out\_data\_vld : **out** std\_logic;
24. out\_calisiyor : **out** std\_logic
25. );
26. **end** konvolusyon\_sinyal;
27. **architecture** Behavioral **of** konvolusyon\_sinyal **is**
28. **type** t\_Kayma\_Ctrl **is** (BOSTA, DATA\_KAYDIR);
29. **signal** r\_Kayma\_Ctrl : t\_Kayma\_Ctrl := BOSTA;
30. **type** t\_Filtre\_Hesap **is** (BOSTA, HESAPLA, TAMAM);
31. **signal** r\_Filtre\_Hesap : t\_Filtre\_Hesap := BOSTA;
32. **type** t\_katsayi\_bellek **is array** (**0** to **KATSAYI - 1** ) **of** std\_logic\_vector(**KATSAYI\_UZUNLUGU - 1** downto **0**);
33. **signal** r\_katsayi\_bellek : t\_katsayi\_bellek := (**others** => (**others** => '0'));
34. **type** t\_data\_bellek **is** **array** (**0** to **KATSAYI - 1** ) **of** std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**);
35. **signal** r\_data\_bellek : t\_data\_bellek := (**others** => (**others** => '0'));
36. **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**
37. **variable** v\_data\_bellek : t\_data\_bellek;
38. **begin**
39. v\_data\_bellek := r\_data\_bellek;
40. **for** n\_i **in** **KATSAYI - 2** downto **0** **loop**
41. v\_data\_bellek(n\_i + 1) := v\_data\_bellek(n\_i);
42. **end** **loop**;
43. v\_data\_bellek(0) := in\_data;
44. **return** v\_data\_bellek;
45. **end** f\_Bellek\_Kaydir;
46. **signal** r\_hesap\_basla : std\_logic := '0';
47. **signal** r\_data : std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**) := (**others** => '0');
48. **signal** r\_toplam : std\_logic\_vector(**VERI\_UZUNLUGU + KATSAYI\_UZUNLUGU + log2\_int(KATSAYI) - 1** downto **0**) := (**others** => '0');
49. **signal** r\_data\_out : std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**) := (**others** => '0');
50. **signal** r\_data\_out\_vld : std\_logic := '0';
51. **signal** r\_calisiyor : std\_logic := '0';
52. **signal** n\_i : integer := 0;
53. **begin**
54. out\_data <= r\_data\_out;
55. out\_data\_vld <= r\_data\_out\_vld;
56. out\_calisiyor <= r\_calisiyor;
57. **process**(in\_clk, in\_rst)
58. **begin**
59. **if** in\_rst = '1' **then**
60. r\_katsayi\_bellek <= (**others** => (**others** => '0'));
61. **elsif** rising\_edge(in\_clk) **then**
62. **if** in\_katsayi\_vld = '1' **then**
63. r\_katsayi\_bellek(conv\_integer(in\_katsayi\_addr)) <= in\_katsayi\_data;
64. **end** **if**;
65. **end** **if**;
66. **end** **process**;
67. **process**(in\_clk, in\_rst)
68. **begin**
69. **if** in\_rst = '1' **then**
70. r\_Kayma\_Ctrl <= BOSTA;
71. r\_data\_bellek <= (others => (others => '0'));
72. r\_hesap\_basla <= '0';
73. r\_data <= (others => '0');
74. **elsif** rising\_edge(in\_clk) **then**
75. r\_hesap\_basla <= '0';
76. **case** r\_Kayma\_Ctrl **is**
77. **when** BOSTA =>
78. **if** in\_data\_vld = '1' **then**
79. r\_data <= in\_data;
80. r\_Kayma\_Ctrl <= DATA\_KAYDIR;
81. **end** if;
82. **when** DATA\_KAYDIR =>
83. r\_data\_bellek <= f\_Bellek\_Kaydir(r\_data\_bellek, r\_data);
84. r\_Kayma\_Ctrl <= BOSTA;
85. r\_hesap\_basla <= '1';
87. **when** **others** => **NULL**;
88. **end** **case**;
89. **end** if**;**
90. **end** **process**;
91. **process**(in\_clk, in\_rst)
92. **begin**
93. **if** in\_rst = '1' **then**
94. r\_Filtre\_Hesap <= BOSTA;
95. r\_toplam <= (others => '0');
96. r\_data\_out\_vld <= '0';
97. r\_data\_out <= (others => '0');
98. r\_calisiyor <= '0';
99. n\_i <= 0;
100. **elsif** rising\_edge(in\_clk) **then**
101. r\_data\_out\_vld <= '0';
102. **case** r\_Filtre\_Hesap **is**
103. **when** BOSTA =>
104. r\_calisiyor <= '0';
105. **if** r\_hesap\_basla = '1' **and** in\_en = '1' **then**
106. r\_Filtre\_Hesap <= HESAPLA;
107. r\_calisiyor <= '1';
108. **end** **if**;
109. **when** HESAPLA =>
110. r\_toplam <= r\_toplam + sxt((r\_data\_bellek(n\_i)) \* (r\_katsayi\_bellek(KATSAYI - 1 - n\_i)), r\_toplam'length);
111. n\_i <= n\_i + 1;
112. **if** n\_i = KATSAYI - 1 **then**
113. n\_i <= 0;
114. r\_Filtre\_Hesap <= TAMAM;
115. **end** **if**;
116. **when** TAMAM =>
117. r\_toplam <= (others => '0');
118. r\_calisiyor <= '0';
119. r\_data\_out\_vld <= '1';
120. r\_data\_out <= r\_toplam(**KATSAYI\_CARPIM + VERI\_UZUNLUGU - 1** downto **KATSAYI\_CARPIM**);
121. r\_Filtre\_Hesap <= BOSTA;
122. **when** **others** => **NULL**;
123. **end** **case**;
124. **end** **if**;
125. **end** **process**;
126. **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 ‑ Alçak geçiren filtre uygulaması

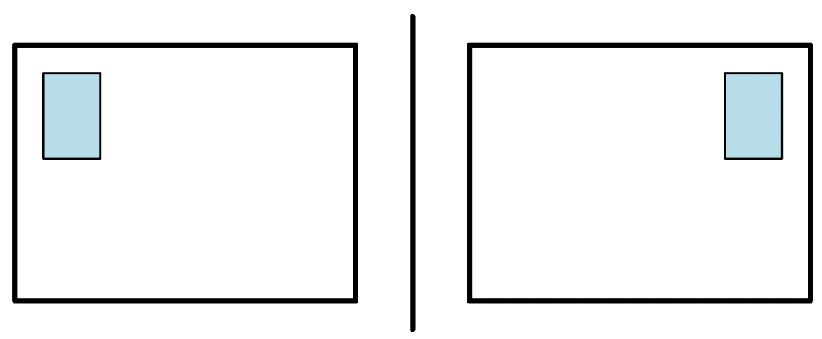
1. **library** IEEE;
2. **use** IEEE.STD\_LOGIC\_1164.ALL;
3. **use** IEEE.STD\_LOGIC\_UNSIGNED.all;
4. **use** IEEE.STD\_LOGIC\_ARITH.all;
5. **use** work.ornekler\_paket.all;
6. **use** std.textio.ALL;
7. **entity** tb\_konvolusyon\_signal **is**
8. **end** tb\_konvolusyon\_signal;
9. **architecture** Behavioral **of** tb\_konvolusyon\_signal **is**
10. **component** konvolusyon\_sinyal **is**
11. **Generic**(
12. VERI\_UZUNLUGU : integer := 24;
13. KATSAYI : integer := 5;
14. KATSAYI\_UZUNLUGU : integer := 8;
15. KATSAYI\_CARPIM : integer := 3
16. );
17. **Port** (
18. in\_clk : **in** std\_logic;
19. in\_rst : **in** std\_logic;
20. in\_en : **in** std\_logic;
21. in\_katsayi\_vld : **in** std\_logic;
22. in\_katsayi\_addr : **in** std\_logic\_vector(**log2\_int(KATSAYI)** downto **0**);
23. in\_katsayi\_data : **in** std\_logic\_vector(**KATSAYI\_UZUNLUGU - 1** downto **0)**;
24. in\_data : **in** std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**);
25. in\_data\_vld : **in** std\_logic;
26. out\_data : **out** std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**);
27. out\_data\_vld : **out** std\_logic;
28. out\_calisiyor : **out** std\_logic
29. );
30. **end** **component**;
31. **constant** CLK\_PERIOD : time := 10 ns;
32. **constant** ORNEKLEME\_PERIOD : time := 10 us;
33. **constant** VERI\_YOLU : string := "C:\sin.txt";
34. **constant** VERI\_UZUNLUGU : integer := 24;
35. **constant** KATSAYI : integer := 17;
36. **constant** KATSAYI\_UZUNLUGU : integer := 8;
37. **constant** KATSAYI\_CARPIM : integer := 7;
39. **signal** in\_clk : std\_logic := '0';
40. **signal** ornekleme\_clk : std\_logic := '0';
41. **signal** ornek\_data : std\_logic\_vector(**23** downto **0**) := (**others** => '0');
42. **signal** clk\_domain : std\_logic\_vector(**3** downto **0**) := (**others** => '0');
43. **signal** son\_data : std\_logic := '0';
44. **signal** in\_rst : std\_logic := '0';
45. **signal** in\_en : std\_logic := '0';
46. **signal** in\_katsayi\_vld : std\_logic := '0';
47. **signal** in\_katsayi\_addr : std\_logic\_vector(**log2\_int(KATSAYI)** downto **0**) := (others => '0');
48. **signal** in\_katsayi\_data : std\_logic\_vector(**KATSAYI\_UZUNLUGU -** 1 downto **0**) := (**others** => '0');
49. **signal** in\_data : std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**) := (**others** => '0');
50. **signal** in\_data\_vld : std\_logic := '0';
51. **signal** out\_data : std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**) := (**others** => '0');
52. **signal** out\_data\_vld : std\_logic := '0';
53. **signal** out\_calisiyor : std\_logic := '0';
55. **type** t\_Katsayi\_Kontrol **is** (BOSTA, YUKLE);
56. **signal** r\_Katsayi\_Kontrol : t\_Katsayi\_Kontrol := BOSTA;
58. **type** t\_Filtre\_Katsayi **is array** (**0** to **KATSAYI - 1**) **of** integer;
59. **signal** r\_Filtre\_Katsayi : t\_Filtre\_Katsayi := (1, 2, 3, 5, 8, 10, 13, 14, 15, 14, 13, 10, 8, 5, 3, 2, 1);
60. **signal** katsayi\_yukle : std\_logic := '0';
61. **signal** n\_i : integer := 0;
62. **begin**
63. **process**
64. **begin**
65. in\_clk <= '1';
66. **wait** **for** CLK\_PERIOD / 2;
67. in\_clk <= '0';
68. **wait** **for** CLK\_PERIOD / 2;
69. **end** **process**;
70. **process**
71. **begin**
72. ornekleme\_clk <= '1';
73. **wait** **for** ORNEKLEME\_PERIOD / 2;
74. ornekleme\_clk <= '0';
75. **wait** **for** ORNEKLEME\_PERIOD / 2;
76. **end** **process**;
77. **process**
78. **begin**
79. katsayi\_yukle <= '0';
80. **wait** **for** CLK\_PERIOD \* 2;
81. katsayi\_yukle <= '1';
82. **wait** **for** CLK\_PERIOD ;
83. katsayi\_yukle <= '0';
84. **wait**;
85. **end** **process**;
86. **process** (ornekleme\_clk)
87. **file** dosya : text **open** read\_mode **is** VERI\_YOLU ;
88. **variable** satir : line;
89. **variable** data : integer;
90. **begin**
91. **if** rising\_edge(ornekleme\_clk) **then**
92. **if** **not**(endfile(dosya)) **then**
93. readline(dosya, satir);
94. read(satir, data);
95. ornek\_data <= conv\_std\_logic\_vector(data, 24) ;
96. son\_data <= '0';
97. **else**
98. ornek\_data <= conv\_std\_logic\_vector(0, 24) ;
99. son\_data <= '1';
100. **end** **if**;
101. **end** **if**;
102. **end** **process**;
103. **process**(in\_clk)
104. **begin**
105. **if** rising\_edge(in\_clk) **then**
106. clk\_domain <= clk\_domain(2 downto 0) & ornekleme\_clk;
107. **end** **if**;
108. **end** **process**;
109. **process** (in\_clk)
110. **begin**
111. **if** rising\_edge(in\_clk) **then**
112. **if** son\_data = '0' **and** clk\_domain(3 downto 2) = "01" **then**
113. in\_data\_vld <= '1' ;
114. in\_data <= ornek\_data ;
115. **else**
116. in\_data\_vld <= '0' ;
117. in\_data <= (others=>'0') ;
118. **end** **if**;
119. **end** **if**;
120. **end** **process**;
121. **process**(in\_clk)
122. **begin**
123. **if** rising\_edge(in\_clk) **then**
124. in\_katsayi\_vld <= '0';
125. **case** r\_Katsayi\_Kontrol **is**
126. **when** BOSTA =>
127. **if** katsayi\_yukle = '1' **then**
128. r\_Katsayi\_Kontrol <= YUKLE;
129. **end** **if**;
131. **when** YUKLE =>
132. in\_katsayi\_vld <= '1';
133. in\_katsayi\_addr <= conv\_std\_logic\_vector(n\_i, log2\_int(KATSAYI) + 1);
134. in\_katsayi\_data <= conv\_std\_logic\_vector( r\_Filtre\_Katsayi(n\_i), KATSAYI\_UZUNLUGU);
135. **if** n\_i = KATSAYI - 1 **then**
136. r\_Katsayi\_Kontrol <= BOSTA;
137. n\_i <= 0;
138. in\_en <= '1';
139. **else**
140. n\_i <= n\_i + 1;
141. **end** **if**;
142. **when** **others** => **NULL**;
143. **end** **case**;
144. **end** **if**;
145. **end** **process**;
146. konvolusyon\_sinyal\_map : konvolusyon\_sinyal
147. **Generic** **map**(
148. VERI\_UZUNLUGU => VERI\_UZUNLUGU,
149. KATSAYI => KATSAYI,
150. KATSAYI\_UZUNLUGU => KATSAYI\_UZUNLUGU,
151. KATSAYI\_CARPIM => KATSAYI\_CARPIM
152. )
153. **Port** **map**(
154. in\_clk => in\_clk,
155. in\_rst => in\_rst,
156. in\_en => in\_en,
157. in\_katsayi\_vld => in\_katsayi\_vld,
158. in\_katsayi\_addr => in\_katsayi\_addr,
159. in\_katsayi\_data => in\_katsayi\_data,
160. in\_data => in\_data,
161. in\_data\_vld => in\_data\_vld,
162. out\_data => out\_data,
163. out\_data\_vld => out\_data\_vld,
164. out\_calisiyor => out\_calisiyor
165. );
166. **end** Behavioral;

## 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.

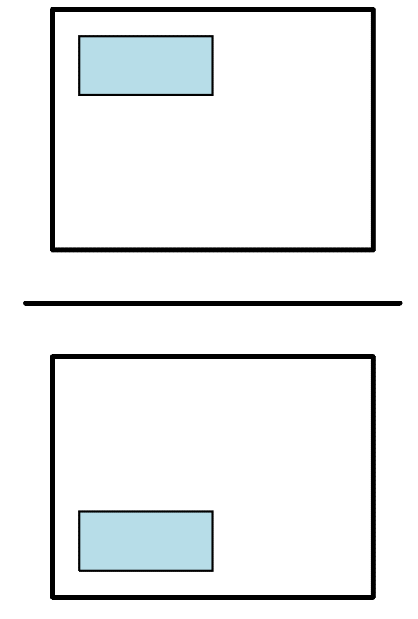
Uygulamaya geçmeden önce ilk olarak temel imge işleme işlemleri tanıtılacak olup ardından VHDL kodları verilecektir. Örnek uygulamada kullanılan görüntü 8 bit gri seviyeli bir görüntüdür. Bu yüzden her beneğin (pixel) alabileceği değerler 0 ile 255 arasında olmaktadır. (x,y) gösterimi ise imgenin her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

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

**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.)

**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.

**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. **entity** temel\_imge\_isleme **is**
7. **generic**(
8. IMGE\_SATIR : integer := 8;
9. IMGE\_SUTUN : integer := 8;
10. VERI\_UZUNLUGU : integer := 24
11. );
12. **port**(
13. in\_clk : **in** std\_logic;
14. in\_rst : **in** std\_logic;
15. in\_en : **in** std\_logic;
16. in\_basla : **in** std\_logic;
17. in\_islem : **in** std\_logic\_vector(**2** downto **0**);
18. in\_data : **in** std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**);
19. in\_data\_vld : **in** std\_logic;
20. out\_addr : **out** std\_logic\_vector(**log2\_int(IMGE\_SATIR \* IMGE\_SUTUN) - 1** downto **0**);
21. out\_addr\_vld : **out** std\_logic;
22. out\_data : **out** std\_logic\_vector(**7** downto **0**);
23. out\_data\_vld : **out** std\_logic;
24. out\_tamam : **out** std\_logic
25. );
26. **end** temel\_imge\_isleme;
27. **architecture** Behavioral **of** temel\_imge\_isleme **is**
28. **constant** AYNALAMA : std\_logic\_vector(**2** downto **0**) := "000";
29. **constant** TERS\_CEVIRME : std\_logic\_vector(**2** downto **0**) := "001";
30. **constant** NEGATIFLEME : std\_logic\_vector(**2** downto **0**) := "010";
31. **constant** ESIKLEME : std\_logic\_vector(**2** downto **0**) := "011";
32. **constant** PARLAKLIK\_ARTIR : std\_logic\_vector(**2** downto **0**) := "100";
33. **constant** PARLAKLIK\_AZALT : std\_logic\_vector(**2** downto **0**) := "101";
34. **constant** KARSITLIK\_ARTIR : std\_logic\_vector(**2** downto **0**) := "110";
35. **constant** KARSITLIK\_AZALT : std\_logic\_vector(**2** downto **0**) := "111";
36. **type** t\_Imge\_Isleme **is** (BOSTA, RAMDAN\_OKU, OKUMA\_BEKLE, ISLEM\_YAP, SAYAC\_KONT, TAMAM );
37. **signal** r\_Imge\_Isleme : t\_Imge\_Isleme := RAMDAN\_OKU;
38. **signal** n\_i : integer := 0;
39. **signal** n\_j : integer := 0;
40. **signal** r\_addr : std\_logic\_vector(**log2\_int(IMGE\_SATIR \* IMGE\_SUTUN) - 1** downto **0**) := (**others** => '0');
41. **signal** r\_addr\_vld : std\_logic := '0';
42. **signal** r\_data : std\_logic\_vector(**7** downto **0**) := (**others** => '0');
43. **signal** r\_data\_vld : std\_logic := '0';
44. **signal** r\_tamam : std\_logic := '0';
45. **begin**
47. out\_addr <= r\_addr;
48. out\_addr\_vld <= r\_addr\_vld;
49. out\_data <= r\_data;
50. out\_data\_vld <= r\_data\_vld;
51. out\_tamam <= r\_tamam;
52. **process**(in\_clk, in\_rst)
53. **begin**
54. **if** in\_rst = '1' **then**
55. r\_Imge\_Isleme <= BOSTA;
56. n\_i <= 0;
57. n\_j <= 0;
58. r\_addr <= (**others** => '0');
59. r\_addr\_vld <= '0';
60. r\_data <= (**others** => '0');
61. r\_data\_vld <= '0';
62. r\_tamam <= '0';
63. **elsif** rising\_edge(in\_clk) **then**
64. **if** in\_en = '1' **then**
65. r\_data\_vld <= '0';
66. r\_addr\_vld <= '0';
67. r\_tamam <= '0';
69. **case** r\_Imge\_Isleme **is**
70. **when** BOSTA =>
71. **if** in\_basla = '1' **then**
72. r\_Imge\_Isleme <= RAMDAN\_OKU;
73. **end** if**;**
75. **when** RAMDAN\_OKU =>
76. **if** in\_islem = AYNALAMA **then**
77. r\_addr <= conv\_std\_logic\_vector(n\_i \* IMGE\_SUTUN + (IMGE\_SUTUN - 1 - n\_j) , r\_addr'length);
78. **elsif** in\_islem = TERS\_CEVIRME **then**
79. r\_addr <= conv\_std\_logic\_vector((IMGE\_SATIR - 1 -n\_i) \* IMGE\_SUTUN + n\_j , r\_addr'length);
80. **elsif** in\_islem = NEGATIFLEME **or** in\_islem = ESIKLEME **or**
81. in\_islem = PARLAKLIK\_ARTIR **or** in\_islem = PARLAKLIK\_AZALT **or**
82. in\_islem = KARSITLIK\_ARTIR **or** in\_islem = KARSITLIK\_AZALT then
83. r\_addr<= conv\_std\_logic\_vector(n\_i \* IMGE\_SUTUN + n\_j , r\_addr'length);
84. **end** **if**;
85. r\_addr\_vld <= '1';
86. r\_Imge\_Isleme <= OKUMA\_BEKLE;
87. **when** OKUMA\_BEKLE =>
88. **if** in\_data\_vld = '1' **then**
89. r\_data <= in\_data;
90. r\_Imge\_Isleme <= ISLEM\_YAP;
91. **end** **if**;
92. **when** ISLEM\_YAP =>
93. **if** in\_islem = AYNALAMA **or** in\_islem = TERS\_CEVIRME **then**
94. r\_data <= r\_data;
95. **elsif** in\_islem = NEGATIFLEME **then**
96. r\_data <= 255 - r\_data;
97. **elsif** in\_islem = ESIKLEME **then**
98. **if** r\_data > 128 **then**
99. r\_data <= conv\_std\_logic\_vector(255, r\_data'length);
100. **else**
101. r\_data <= (**others** => '0');
102. **end** **if**;
103. **elsif** in\_islem = PARLAKLIK\_ARTIR **then**
104. **if** r\_data > 210 **then**
105. r\_data <= conv\_std\_logic\_vector(255, r\_data'length);
106. **else**
107. r\_data <= r\_data + 45;
108. **end** **if**;
109. **elsif** in\_islem = PARLAKLIK\_AZALT **then**
110. **if** r\_data < 45 **then**
111. r\_data <= conv\_std\_logic\_vector(0, r\_data'length);
112. **else**
113. r\_data <= r\_data - 45;
114. **end** **if**;
115. **elsif** in\_islem = KARSITLIK\_ARTIR **then**
116. **if** r\_data > 128 **then**
117. r\_data <= conv\_std\_logic\_vector(255, r\_data'length);
118. **else**
119. r\_data <= r\_data(6 downto 0) & '0';
120. **end** **if**;
121. **elsif** in\_islem = KARSITLIK\_AZALT **then**
122. r\_data <= '0' & r\_data(7 downto 1);
123. **end** **if**;
124. r\_data\_vld <= '1';
125. r\_Imge\_Isleme <= SAYAC\_KONT;
126. **when** SAYAC\_KONT =>
127. **if** n\_j = IMGE\_SUTUN - 1 **then**
128. n\_j <= 0;
129. **if** n\_i = IMGE\_SATIR - 1 **then**
130. n\_i <= 0;
131. r\_Imge\_Isleme <= TAMAM;
132. **else**
133. n\_i <= n\_i + 1;
134. r\_Imge\_Isleme <= RAMDAN\_OKU;
135. **end** **if**;
136. **else**
137. n\_j <= n\_j + 1;
138. r\_Imge\_Isleme <= RAMDAN\_OKU;
139. **end** **if**;
140. **when** TAMAM =>
141. r\_tamam <= '1';
142. r\_Imge\_Isleme <= BOSTA;
143. **when** **others** => **NULL**;
144. **end** **case**;
145. **end** **if**;
146. **end** **if**;
147. **end** **process**;
148. **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.

(a)

(b)

(c)

(d)

(e)

(f)

(g)

(h)

(i)

Şekil ‑ 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. **entity** tb\_temel\_imge\_isleme **is**
8. **end** tb\_temel\_imge\_isleme;
9. **architecture** Behavioral **of** tb\_temel\_imge\_isleme **is**
10. **component** temel\_imge\_isleme
11. **generic**(
12. IMGE\_SATIR : integer := 8;
13. IMGE\_SUTUN : integer:= 8;
14. VERI\_UZUNLUGU : integer:= 24
15. );
16. **port**(
17. in\_clk : **in** std\_logic;
18. in\_rst : **in** std\_logic;
19. in\_en : **in** std\_logic;
20. in\_basla : **in** std\_logic;
21. in\_islem : **in** std\_logic\_vector(**2** downto **0**);
22. in\_data : **in** std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**);
23. in\_data\_vld : **in** std\_logic;
24. out\_addr : **out** std\_logic\_vector(**log2\_int(IMGE\_SATIR \* IMGE\_SUTUN) - 1** downto **0**);
25. out\_addr\_vld : **out** std\_logic;
26. out\_data : **out** std\_logic\_vector(**7** downto **0**);
27. out\_data\_vld : **out** std\_logic;
28. out\_tamam : **out** std\_logic
29. );
30. **end** **component**;
31. **component** block\_ram
32. **generic**(
33. VERI\_UZUNLUGU : integer := 8;
34. RAM\_DERINLIGI : integer := 110
35. );
36. **port**(in\_clk : **in** std\_logic;
37. in\_rst : **in** std\_logic;
38. in\_ram\_aktif : **in** std\_logic;
39. in\_yaz\_en : **in** std\_logic;
40. in\_oku\_en : **in** std\_logic;
41. in\_data\_addr : **in** std\_logic\_vector(**log2\_int(RAM\_DERINLIGI) - 1** downto **0**);
42. in\_data : **in** std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**);
43. out\_data : **out** std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**);
44. out\_data\_vld : **out** std\_logic
45. );
46. **end** **component**;
47. **constant** CLK\_PERIOD : time := 20 ns;
48. **constant** IMGE\_SATIR : integer := 256;
49. **constant** IMGE\_SUTUN : integer := 256;
50. **constant** VERI\_UZUNLUGU : integer := 8;
51. **constant** VERI\_YOLU\_OKUMA : string := "C:\cameraman.txt";
52. **constant** VERI\_YOLU\_YAZMA : string := "D:\cameraman\_sonuc.txt";
53. **constant** AYNALAMA : std\_logic\_vector(**2** downto **0**) := "000";
54. **constant** TERS\_CEVIRME : std\_logic\_vector(**2** downto **0**) := "001";
55. **constant** NEGATIFLEME : std\_logic\_vector(**2** downto **0**) := "010";
56. **constant** ESIKLEME : std\_logic\_vector(**2** downto **0**) := "011";
57. **constant** PARLAKLIK\_ARTIR : std\_logic\_vector(**2** downto **0**) := "100";
58. **constant** PARLAKLIK\_AZALT : std\_logic\_vector(**2** downto **0**) := "101";
59. **constant** KARSITLIK\_ARTIR : std\_logic\_vector(**2** downto **0**) := "110";
60. **constant** KARSITLIK\_AZALT : std\_logic\_vector(**2** downto **0**) := "111";
61. **type** t\_Imge\_Isleme **is** (RAM\_OKUMA, RAM\_YAZMA, TAMAM);
62. **signal** r\_Imge\_Isleme : t\_Imge\_Isleme := RAM\_YAZMA;
63. **signal** in\_clk : std\_logic := '0';
64. **signal** in\_rst : std\_logic := '0';
65. **signal** in\_basla : std\_logic := '0';
66. **signal** in\_ram\_aktif : std\_logic := '1';
67. **signal** out\_data\_vld : std\_logic := '0';
68. **signal** out\_data : std\_logic\_vector(**7** downto **0**) := (**others** => '0');
69. **signal** in\_ram\_data : std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**) := (**others** => '0');
70. **signal** in\_ram\_data\_addr : std\_logic\_vector(**log2\_int(IMGE\_SATIR \* IMGE\_SUTUN) - 1** downto **0**) := (**others** => '0');
71. **signal** out\_ram\_data : std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**) := (**others** => '0');
72. **signal** out\_data\_addr : std\_logic\_vector(**log2\_int(IMGE\_SATIR \* IMGE\_SUTUN) - 1** downto **0**) := (**others** => '0');
73. **signal** out\_ram\_data\_vld : std\_logic := '0';
74. **signal** in\_en : std\_logic := '0';
75. **signal** in\_yaz\_en : std\_logic := '0';
76. **signal** in\_oku\_en : std\_logic := '0';
77. **signal** data\_sayac : integer := 0;
78. **signal** out\_data\_addr\_vld : std\_logic := '0';
79. **signal** r\_imge\_isleme\_tamam : std\_logic := '0';
80. **begin**
81. **process**
82. **begin**
83. in\_clk <= '1';
84. **wait** **for** CLK\_PERIOD / 2;
85. in\_clk <= '0';
86. **wait** **for** CLK\_PERIOD / 2;
87. **end** **process**;
88. **process**
89. **begin**
90. in\_basla <= '1';
91. **wait** **for** CLK\_PERIOD;
92. in\_basla <= '0'; wait;
93. **end** **process**;
94. **process**(in\_clk)
95. **file** dosya\_okuma : **text** **open** read\_mode **is** VERI\_YOLU\_OKUMA;
96. **file** dosya\_yazma : **text** **open** write\_mode **is** VERI\_YOLU\_YAZMA;
97. **variable** satir\_okuma : line;
98. **variable** satir\_yazma : line;
99. **variable** data\_okuma : integer;
100. **begin**
101. **if** rising\_edge(in\_clk) **then**
102. **if** out\_data\_vld = '1' **then**
103. write(satir\_yazma, conv\_integer(out\_data));
104. writeline(dosya\_yazma, satir\_yazma);
105. **end** **if**;
107. **case** r\_Imge\_Isleme **is**
108. **when** RAM\_YAZMA =>
109. **if** **not** endfile(dosya\_okuma) **then**
110. readline(dosya\_okuma, satir\_okuma);
111. read(satir\_okuma, data\_okuma);
112. in\_ram\_data <= conv\_std\_logic\_vector( data\_okuma, VERI\_UZUNLUGU);
113. in\_ram\_data\_addr <= conv\_std\_logic\_vector( data\_sayac, in\_ram\_data\_addr'length);
114. in\_en <= '0';
115. in\_yaz\_en <= '1';
116. data\_sayac <= data\_sayac + 1;
117. **else**
118. r\_Imge\_Isleme <= RAM\_OKUMA;
119. in\_yaz\_en <= '0';
120. **end** **if**;
121. **when** RAM\_OKUMA =>
122. in\_en <= '1';
123. in\_ram\_data\_addr <= out\_data\_addr;
124. in\_oku\_en <= out\_data\_addr\_vld ;
125. **if** r\_imge\_isleme\_tamam = '1' **then**
126. r\_Imge\_Isleme <= TAMAM;
127. **end** **if**;
128. **when** **TAMAM** => **null**;
129. **when** **others** => **NULL**;
130. **end** **case**;
131. **end** **if**;
132. **end** **process**;
133. temel\_imge\_isleme\_map : temel\_imge\_isleme
134. **generic** **map**(
135. IMGE\_SATIR => IMGE\_SATIR,
136. IMGE\_SUTUN => IMGE\_SUTUN,
137. VERI\_UZUNLUGU => VERI\_UZUNLUGU
138. )
139. **port** **map**(
140. in\_clk => in\_clk,
141. in\_rst => in\_rst,
142. in\_en => in\_en,
143. in\_basla => in\_basla,
144. in\_islem => KARSITLIK\_AZALT,
145. in\_data => out\_ram\_data,
146. in\_data\_vld => out\_ram\_data\_vld,
147. out\_addr => out\_data\_addr,
148. out\_addr\_vld => out\_data\_addr\_vld,
149. out\_data => out\_data,
150. out\_data\_vld => out\_data\_vld,
151. out\_tamam => r\_imge\_isleme\_tamam
152. );
153. block\_ram\_map : block\_ram
154. **generic** **map**(
155. VERI\_UZUNLUGU => VERI\_UZUNLUGU,
156. RAM\_DERINLIGI => IMGE\_SATIR \* IMGE\_SUTUN
157. )
158. **port** **map**(
159. in\_clk => in\_clk,
160. in\_rst => in\_rst,
161. in\_ram\_aktif => in\_ram\_aktif,
162. in\_yaz\_en => in\_yaz\_en,
163. in\_oku\_en => in\_oku\_en,
164. in\_data\_addr => in\_ram\_data\_addr,
165. in\_data => in\_ram\_data,
166. out\_data => out\_ram\_data,
167. out\_data\_vld => out\_ram\_data\_vld
168. );
169. **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. **entity** block\_ram **is**
6. **generic**(
7. VERI\_UZUNLUGU : integer := 8;
8. RAM\_DERINLIGI : integer := 110
9. );
10. **port**(
11. in\_clk : **in** std\_logic;
12. in\_rst : **in** std\_logic;
13. in\_ram\_aktif : **in** std\_logic;
14. in\_yaz\_en : **in** std\_logic;
15. in\_oku\_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(VE**RI\_UZUNLUGU - 1** downto **0**);
19. out\_data\_vld : **out** std\_logic
20. );
21. **end** block\_ram;
22. **architecture** Behavioral **of** block\_ram **is**
23. **type** t\_BRAM\_DATA **is array** (**0** to **RAM\_DERINLIGI - 1**) **of** std\_logic\_vector(**VERI\_UZUNLUGU - 1** downto **0**) ;
24. **signal** r\_BRAM\_DATA : t\_BRAM\_DATA := (**others** =>(**others** => '0'));
25. **begin**
26. **process**(in\_clk, in\_rst)
27. **begin**
28. **if** in\_rst = '1' **then**
29. r\_BRAM\_DATA <= (**others** =>(**others** => '0'));
30. **elsif** rising\_edge(in\_clk) **then**
31. **if** in\_ram\_aktif = '1' **then**
32. **if** in\_oku\_en = '1' **then**
33. out\_data <= r\_BRAM\_DATA( conv\_integer( in\_data\_addr));
34. out\_data\_vld <= '1';
35. **else**
36. out\_data\_vld <= '0';
37. **end** **if**;
38. **if** in\_yaz\_en = '1' **then**
39. r\_BRAM\_DATA(conv\_integer(in\_data\_addr)) <= in\_data;
40. **end** if**;**
41. **end** **if**;
42. **end** **if**;
43. **end** **process**;
44. **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

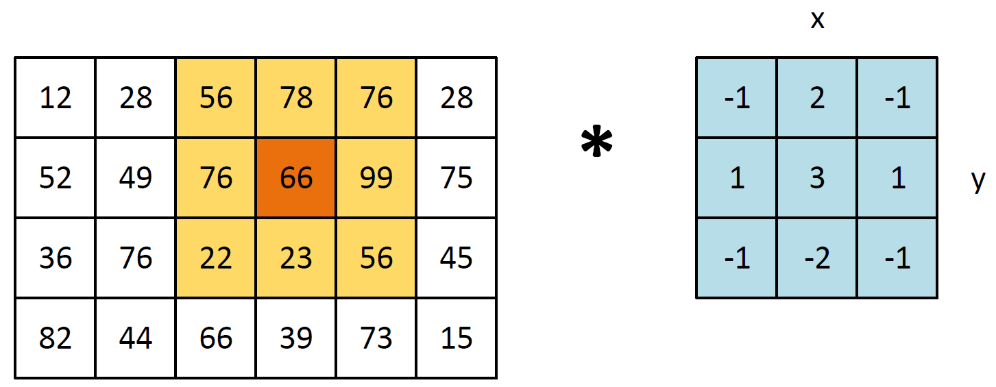
fclose(dosya);

figure, imshow(uint8(yeni\_imge))

## 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.





Şekil ‑ Konvolüsyon işlemi



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



**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.

(a)

(b)

(c)

(d)

(e)

(f)

(g)

(h)

(i)

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))