# Eşzamanlı Atama İfadeleri

Eşzamanlı atama ifadeleri mimari bölgesinde bir değerin bir sinyale atama işleminde kullanılır. VHDL 4 çeşit eşzamanlı atama ifadesi mevcuttur. Bu atama ifadeleri aşağıda tanıtılacaktır.

## Basit Sinyal Atamaları

Basit sinyal atamaları lojik veya aritmetik ifadelerde kullanılır. Atama işlemine ait genel form aşağıda verilmiştir.

sinyal\_adi <= ifade;

**<=** operatörü VHDL’de atama operatörüdür. Aşağıdaki verilen örnekte bu operatörün kullanımını daha detaylı olarak göstermektedir. **sinyal\_sonuc** sinyaline, **sinyal\_1** ve **sinyal\_2** sinyallerinin **xor** sonucunun **sinyal\_3** ile **and** işlemine tabi tutulmasında elde edilen sonuç atanmaktadır. Bu tanımlamada **sinyal\_sonuc** ifadesi, bir bitlik sonucu içermektedir.

..

..

**signal** sinyal\_1 : std\_logic;

**signal** sinyal\_2 : std\_logic;

**signal** sinyal\_3 : std\_logic;

**signal** sinyal\_sonuc : std\_logic;

..

..

sinyal\_sonuc <= (sinyal\_1 **xor** sinyal\_2) **and** sinyal\_3;

..

..

VHDL’de aynı zamanda çoklu bit atamaları da yapılabilmektedir. Aşağıda verilen örnekte **sinyal\_1** ve **sinyal\_2** sinyalleri 3 bitlik **std\_logic\_vector** tipinde tanımlanmıştır. **sinyal\_1** ve **sinyal\_2** sinyallerinin **or** işleminin sonucu ise yine 3 bitlik **std\_logic\_vector** tipinde tanımlanan **sinyal\_sonuc** sinyaline atanmaktadır.

..

..

**signal** sinyal\_1 : std\_logic\_vector(**2** downto **0**);

**signal** sinyal\_2 : std\_logic\_vector(**2** downto **0**);

**signal** sinyal\_sonuc : std\_logic\_vector(**2** downto **0**);

..

..

sinyal\_sonuc <= sinyal\_1 **or** sinyal\_2;

..

..

Bu tanımlamada tek bitli olarak aslında aşağıdaki işlemler yapılmaktadır. **sinyal\_sonuc** ifadesinin 0. bitine **sinyal\_1** ile **sinyal\_2**’nin 0. bitlerinin **or** işlemlerinin sonucu atanmaktadır. Aynı şekilde **sinyal\_sonuc** ifadesinin 1. bitine **sinyal\_1** ile **sinyal\_2**’nin 1. bitlerinin **or** işlemlerinin sonucu ve **sinyal\_sonuc** ifadesinin 2. bitine **sinyal\_1** ile **sinyal\_2**’nin 2. bitlerinin **or** işlemlerinin sonucu atanmaktadır.

sinyal\_sonuc(0) <= sinyal\_1(0) **or** sinyal\_2(0);

sinyal\_sonuc(1) <= sinyal\_1(1) **or** sinyal\_2(1);

sinyal\_sonuc(2) <= sinyal\_1(2) **or** sinyal\_2(2);

Aritmetik işlemlerde atama işlemlerine ilişkin örnek aşağıda verilmiştir. Verilen örnekte **sinyal\_1** ve **sinyal\_2** sinyalleri 4 bitlik **std\_logic\_vector** tipinde tanımlanmıştır. **sinyal\_1** ve **sinyal\_2** sinyallerinin **toplama** işleminin sonucu ise yine 4 bitlik **std\_logic\_vector** tipinde tanımlanan **sinyal\_sonuc** sinyaline atanmaktadır.

..

..

**signal** sinyal\_1 : std\_logic\_vector(**3** downto **0**);

**signal** sinyal\_2 : std\_logic\_vector(**3** downto **0**);

**signal** sinyal\_sonuc : std\_logic\_vector(**3** downto **0**);

..

..

sinyal\_sonuc <= sinyal\_1 + sinyal\_2;

..

..

Yukarda belirtilen tanımlamaya alternatif olarak aşağıdaki tanımlamada kullanılabilmektedir. Verilen örnekte **sinyal\_1** ve **sinyal\_2** sinyalleri 4 bitlik **std\_logic\_vector** tipinde tanımlanmıştır. **sinyal\_3** ise 1 bitlik **std\_logic** tipinde tanımlanmıştır. **sinyal\_1** sinyali **&** operatörü ile başına **'0'** eklenerek 5 bitlik hale getirilmiştir. Daha sonra elde edilen bu değer **sinyal\_2** ve **sinyal\_3** ile toplanarak işleminin sonucu ise 5 bitlik **std\_logic\_vector** tipinde tanımlanan **sinyal\_sonuc** sinyaline atanmaktadır.

..

..

**signal** sinyal\_1 : std\_logic\_vector(**3** downto **0**);

**signal** sinyal\_2 : std\_logic\_vector(**3** downto **0**);

**signal** sinyal\_3 : std\_logic;

**signal** sinyal\_sonuc : std\_logic\_vector(**4** downto **0**);

..

..

sinyal\_sonuc<= ('0' & sinyal\_1) + sinyal\_2 + sinyal\_3 ;

..

..

**Örnek 7.1 :** Aşağıda verilen **tam\_toplayici.vhd** VHDL kodunda 21. satırda yukarda anlatılan ifadenin kullanımıyla, 4 bitlik tam toplayıcı devresinde kullanılması gösterilmektedir. Toplam değerinin, en anlamlı biti elde değerini tutmaktadır ve **out\_cikis\_elde** değerine atanmaktadır. Geri kalan 4 bit ise toplam sonucu olarak **out\_cikis** değerine atanmaktadır.

1. **library** IEEE;
2. **use** IEEE.STD\_LOGIC\_1164.all;
3. **use** IEEE.STD\_LOGIC\_SIGNED.all;
4. **entity** tam\_toplayici **is**
5. **port** (
6. in\_giris\_elde : **in** std\_logic;
7. in\_giris\_1 : **in** std\_logic\_vector(**3** downto **0**);
8. in\_giris\_2 : **in** std\_logic\_vector(**3** downto **0**);
9. out\_cikis : **out** std\_logic\_vector(**3** downto **0**);
10. out\_cikis\_elde : **out** std\_logic
11. );
12. **end** tam\_toplayici;
13. **architecture** Behavioral **of** tam\_toplayici **is**
14. **signal** r\_Toplam : std\_logic\_vector(**4** downto **0**);
15. **begin**
16. r\_Toplam<= ('0' & in\_giris\_1) + in\_giris\_2 + in\_giris\_elde;
17. out\_cikis\_elde<= r\_Toplam(4);
18. out\_cikis<= r\_Toplam(**3** downto **0**);
19. **end** Behavioral;

### OTHERS Kullanarak Sinyal Değeri Atama

Bir sinyal değerinin belli bir kısmına aynı bitler yazılacaksa **others** kullanılarak bu işlem yapılabilmektedir. Aşağıda verilen tanımlamada **sinyal\_1** sinyalinin tüm bitlerine **0**, **sinyal\_2** sinyalinin tüm bitlerine ise **1** atanmaktadır.

..

..

**signal** sinyal\_1 : std\_logic\_vector(**7** downto **0**)

**signal** sinyal\_2 : std\_logic\_vector(**3** downto **0**);

..

..

sinyal\_1<= (**others** => '0');

sinyal\_2<= (**others** => '1');

..

..

## Seçilmiş Sinyal Atama

Seçilmiş sinyal ifadesi, seçim koşullarına uygun birkaç alternatif değerden bir sinyalin değer olarak atanmasında kullanılır. Genel tanımlama ifadesi aşağıdaki gibidir.

**with** ifade **select**

sinyal\_adi<= ifade **when** sabit\_deger

{,ifade **when** sabit\_deger } ;

Aşağıda verilen tanımlamada **sinyal\_cikis** ifadesine yapılacak atama değeri **sinyal\_secme** sinyalinin değerine bağlı olarak yapılmaktadır. Eğer **sinyal\_secme** sinyali **'0'** ise **sinyal\_cikis** değerine **sinyal\_1** sinyali atanmaktadır. **sinyal\_secme** sinyali 0 haricinde başka değerler alması durumunda ise **sinyal\_cikis** değerine **sinyal\_2** sinyali atanmaktadır.

..

..

**signal** sinyal\_1 : std\_logic;

**signal** sinyal\_2 : std\_logic;

**signal** sinyal\_secme : std\_logic;

**signal** sinyal\_sonuc : std\_logic;

..

..

**with** sinyal\_secme **select**

sinyal\_sonuc <= sinyal\_1 **when** '0',

sinyal\_2 **when** **others**;

..

..

## Şartlı Sinyal Atamaları

Seçilmiş sinyal atamalarına benzer olarak; şartlı sinyal atamaları da birkaç alternatif değerden bir sinyal değerinin atanması için kullanılır. Genel gösterim aşağıdaki gibidir:

sinyal\_adi<= ifade **when** lojik\_ifade **else**

{ifade **when** lojik\_ifade **else**}

ifade ;

Aşağıda verilen tanımlamada **sinyal\_cikis** ifadesine yapılacak atama değeri 3 farklı koşulda belirlenmektedir. Eğer **sinyal\_secme** sinyali **'0'** ise **sinyal\_cikis** değerine **sinyal\_1** sinyali atanmaktadır. Eğer **sinyal\_secme** sinyali **'1'** ise **sinyal\_cikis** değerine **sinyal\_2** sinyali atanmaktadır. **sinyal\_secme** sinyali **'0'** ve **'1'** haricinde başka değerler alması durumunda ise **sinyal\_cikis** değerine 0 atanmaktadır.

..

..

**signal** sinyal\_1 : std\_logic;

**signal** sinyal\_2 : std\_logic;

**signal** sinyal\_secme : std\_logic;

**signal** sinyal\_sonuc : std\_logic;

..

..

sinyal\_cikis <= sinyal\_1 **when** sinyal\_secme = '0'

**else** sinyal\_2 **when** sinyal\_secme = '1'

**else** '0';

..

..

**Örnek 7.2:** Aşağıda verilen **oncelikli\_atama.vhd** VHDL kodunda şartlı sinyal atama kullanılarak tasarım yapılmıştır. Kodda **in\_giris\_1**, **in\_giris\_2** ve **in\_giris\_3** girişlerini 1 olması durumunda **out\_cikis** değerine sırası ile **"01**", **"10"** ve **"11"** atanmaktadır. Bu şartların dışında meydana gelebilecek durumlarda ise çıkışa **"00"** atanmaktadır.

1. **library** IEEE;
2. **use** IEEE.STD\_LOGIC\_1164.all;
3. **entity** oncelikli\_atama **is**
4. port (
5. in\_giris\_1 : **in** std\_logic;
6. in\_giris\_2 : **in** std\_logic;
7. in\_giris\_3 : **in** std\_logic;
8. out\_cikis : **out** std\_logic\_vector(**1** downto **0**)
9. );
10. **end** oncelikli\_atama;
11. **architecture** Behavioral **of** oncelikli\_atama **is**
12. **begin**
13. out\_cikis<= "01" **when** in\_giris\_1 = '1'
14. **else** "10" **when** in\_giris\_2 = '1'
15. **else** "11" **when** in\_giris\_3 = '1'
16. **else** "00" ;
17. **end** Behavioral;

## GENERATE İfadeleri

**generate** ifadesi, VHDL’de tekrarlanan lojikler eşitlikler veya **component** örneklerinde kullanılır. İki tip **generate** ifadesi mevcuttur :

* **if generate**
* **for generate**

Bu ifadelerden **if generate** çok nadir kullanılır. Fakat **for generate** sıklıkla kullanılan bir ifadedir ve genel gösterimi aşağıda verilmiştir.

**for** değer **in** aralik **generate**

ifade ;

{ifade ;}

**end generate;**

Aşağıda verilen örnekte 4 bitlik **std\_logic\_vector** tipinde **sinyal\_1** sinyaline başlangıç değeri olarak **"0001"** atanmıştır. **sinyal\_2** sinyali de 5 bitlik **std\_logic\_vector** tipinde tanımlanmıştır. **sinyal\_2** sinyalinin en anlamsız bitine **'1'** değeri atanmaktadır. Daha sonra **for generate** döngüsü içerisinde **sinyal\_1** ve **sinyal\_2**’ye ait **n\_i.** bitler **xor** işlemine tabi tutulduktan sonra **sinyal\_2**’nin **n\_i + 1.** bitine atamaktadır. Koda ilişkin devre benzetimi Şekil 6.1’de verilmiştir.

..

..

**signal** sinyal\_1 : std\_logic\_vector(**3** downto **0**) := “0001”;

**signal** sinyal\_2 : std\_logic\_vector(**4** downto **0**);

..

..

sinyal\_2(0) <= ‘1’;

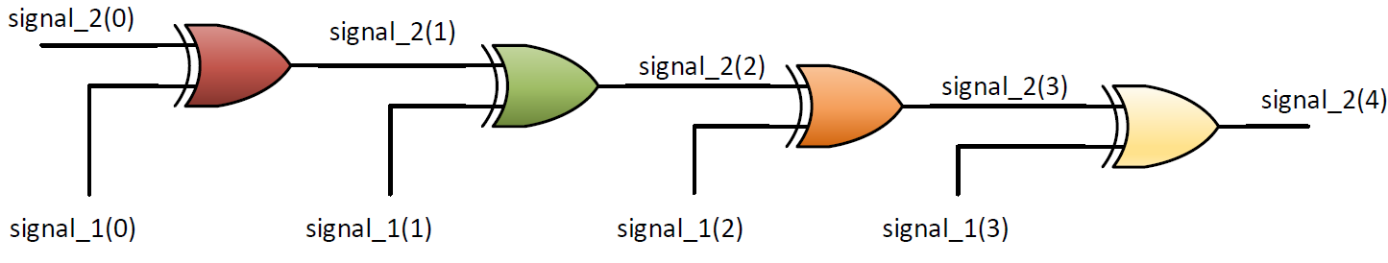
for\_kontrol: **for** n\_i **0** to **3** **generate**

sinyal\_2(n\_i + 1) <= sinyal\_2(n\_i) **xor** sinyal\_1(n\_i)

**end** **generate** for\_kontrol;

..

..



Şekil 6‑1 for generate için verilen örneğe ilişkin lojik gösterim

**if generate** ifadesinin genel gösterimi aşağıda verilmiştir.

**if** koşul **generate**

ifade ;

{ifade ;}

**end** **generate**;

**Örnek 7.3:** Aşağıda verilen **for\_if\_generate.vhd** VHDL kodunda **for generate** ve **if generate** ifadeleri kullanılarak 8 bitlik toplayıcı tasarımı yapılmıştır. **for\_kontrol** etiketli **for generate** döngüsü ile ardışık olarak oluşturulan toplayıcı devreleri ile toplama sonucu elde edilmektedir. Döngü içerisinde bulunan **if\_kontrol\_EAB** etiketli **if generate** söz dizimi ile sadece **n\_i** değerinin sıfır olduğu durumda söz dizimi içerisinde bulunan yarı toplayıcı lojik eşitlikleri aktif hale gelmektedir. **n\_i**’nin diğer durumlarında ise bu blok pasif durumda olacaktır. **if\_kontrol\_DB** etiketli **if generate** söz dizimi ile sadece **n\_i** değerinin sıfırdan farklı olduğu durumlarda söz dizimi içerisinde bulunan tam toplayıcı lojik eşitlikleri aktif hale getirmektedir.

1. **library** IEEE;
2. **use** IEEE.STD\_LOGIC\_1164.all;
3. **entity** for\_if\_generate **is**
4. **port** (
5. in\_giris\_1 : **in** std\_logic\_vector(**7** downto **0**);
6. in\_giris\_2 : **in** std\_logic\_vector(**7** downto **0**);
7. out\_cikis : **out** std\_logic\_vector(**7** downto **0**);
8. out\_cikis\_elde : **out** std\_logic
9. );
10. **end** for\_if\_generate;
11. **architecture** Behavioral **of** for\_if\_generate **is**
12. **signal** r\_toplam : std\_logic\_vector(8 downto 1);
13. **begin**
14. for\_kontrol : **for** n\_i **in** **0** to **7** **generate**
15. if\_kontrol\_EAB : **if** n\_i = 0 **generate**
16. out\_cikis(n\_i)<= in\_giris\_1(n\_i) **xor** in\_giris\_2(n\_i);
17. r\_toplam(n\_i + 1)<= in\_giris\_1(n\_i) **and** in\_giris\_2(n\_i);
18. **end** **generate** if\_kontrol\_EAB;
19. if\_kontrol\_DB : **if** n\_i> 0 **generate**
20. out\_cikis(n\_i) <= r\_toplam(n\_i) **xor**
21. in\_giris\_1(n\_i) **xor** in\_giris\_2(n\_i);
23. r\_toplam(n\_i + 1)<= (r\_toplam(n\_i) **and**
24. in\_giris\_1(n\_i)) **or** (in\_giris\_1(n\_i) **and**
25. in\_giris\_2(n\_i)) **or** (in\_giris\_2(n\_i) **and**
26. r\_toplam(n\_i));
28. **end** **generate** if\_kontrol\_DB;
29. **end** **generate** for\_kontrol;
30. out\_cikis\_elde <= r\_toplam(8);
31. **end** Behavioral;

**Örnek 7.4:** Yukarıda verilen **for\_if\_generate.vhd** VHDL kodunda yarı toplayıcı ve tam toplayıcı lojik eşitlikleri yerine **yari\_toplayici** ve **tam\_toplayici** alt devrelerinin kullanıldığı **port\_map\_for\_if\_generate.vhd** VHDL kodu aşağıda verilmiştir. Kodda 40-46. satırlarda **yari\_toplayici** alt devre tasarımı yapılmıştır. Kodda 50-47. satırlarda **tam\_toplayici** alt devre tasarımı yapılmıştır.

1. **library** IEEE;
2. **use** IEEE.STD\_LOGIC\_1164.all;
3. **entity** port\_map\_for\_if\_generate **is**
4. **Port** (
5. in\_giris\_1 : **in** std\_logic\_vector(**7** downto **0**);
6. in\_giris\_2 : **in** std\_logic\_vector(**7** downto **0**);
7. out\_cikis\_elde : **out** std\_logic;
8. out\_cikis : **out** std\_logic\_vector(**7** downto **0**)
9. );
10. **end** port\_map\_for\_if\_generate;
11. **architecture** Behavioral **of** port\_map\_for\_if\_generate **is**
12. **component** yari\_toplayici
13. **port**(
14. in\_giris\_1 : **in** std\_logic;
15. in\_giris\_2 : **in** std\_logic;
16. out\_cikis : **out** std\_logic;
17. out\_cikis\_elde : **out** std\_logic
18. );
19. **end** **component**;
20. **component** tam\_toplayici
21. **port**(
22. in\_giris\_elde : **in** std\_logic;
23. in\_giris\_1 : **in** std\_logic;
24. in\_giris\_2 : **in** std\_logic;
25. out\_cikis : **out** std\_logic;
26. out\_cikis\_elde : **out** std\_logic
27. );
28. **end** **component**;
29. **signal** r\_toplam : std\_logic\_vector(**8** downto **1**);
30. **begin**
31. for\_kontrol : **for** n\_i **in** **0** to **7** **generate**
32. if\_kontrol\_EAB : **if** n\_i = 0 **generate**
33. yari\_toplayici\_map : yari\_toplayici
34. **port** **map**(
35. in\_giris\_1 => in\_giris\_1(n\_i),
36. in\_giris\_2 => in\_giris\_2(n\_i),
37. out\_cikis => out\_cikis(n\_i),
38. out\_cikis\_elde => r\_toplam(n\_i + 1)
39. );
40. **end** **generate** if\_kontrol\_EAB;
41. if\_kontrol\_DB : **if** n\_i > 0 **generate**
42. tam\_toplayici\_map : tam\_toplayici
43. **port** **map**(
44. in\_giris\_elde => r\_toplam(n\_i),
45. in\_giris\_1 => in\_giris\_1(n\_i),
46. in\_giris\_2 => in\_giris\_2(n\_i),
47. out\_cikis => out\_cikis(n\_i),
48. out\_cikis\_elde => r\_toplam(n\_i + 1)
49. );
50. **end generate** if\_kontrol\_DB;
51. **end generate** for\_kontrol;
52. out\_cikis\_elde <= r\_toplam(8);
53. **end** Behavioral;