

## Bölüm 5. Karnaugh Haritaları (Devamı)

---

### ❖ Tablo Yöntemiyle (Tabulation Method) Lojik İfadelerin İndirgenmesi

- Asal Bileşenlerin Bulunması
- Asıl Asal Bileşenlerin Bulunması

### ❖ NAND veya NOR Kapılarının Üniversal Kapılar Olarak Kullanılması

- Üniversal Kapılar Kullanılarak Diğer Kapıların Elde Edilmesi
- Lojik İfadeleri Cebirsel Olarak NAND ya da NOR Formuna Getirme

# Tablo Yöntemiyle (Tabulation Method) Lojik İfadelerin İndirgenmesi

---

Quine-McCluskey yöntemi olarak bilinen tablo yöntemi, özellikle değişken sayısı fazla olan lojik ifadelerin indirgenmesinde algoritmik bir yaklaşım sunar.

İlk olarak mintermler, içerdikleri 1 sayısına göre gruplanır (Önemsiz birleşimler de bu işleme dahil edilir ).

Daha sonra birbirine komşu gruplar arasında sadece 1 değişken değişmişse, elde edilen mintermin o değişkenine önemsiz manasında ‘—’ işareti konur, diğerleri aynen alınır. Komşu gruptaki terimler arasında ilişki varsa, bu ilişkinin varlığını göstermek için ‘√’ sembolü kullanılır.

Komşu gruplar arasında ilişki içermeyen terimler, asal bileşenleri oluşturur. Bu ilişkilendirme işlemleri, elde edilen terimler üzerinde de devam eder (Ta ki ilişkilendirme işlemleri kalmayıncaya kadar). Daha sonra da asıl asal bileşenlerin bulunması için bir tablo düzenlenir.

# Quine-McCluskey Yöntemi

**Örnek:**  $f(A,B,C,D)=\sum(4,8,10,11,12,15)$  ifadesini tablo yöntemiyle sadeleştirelim.

- İlk aşamada mintermler içerdikleri 1'lere göre gruplandırılır;

Terimlerdeki 1 sayısı	Mintermler	İkili gösterimi
1	$m_4$	0100
	$m_8$	1000
2	$m_{10}$	1010
	$m_{12}$	1100
3	$m_{11}$	1011
4	$m_{15}$	1111

## Örnek: (devam)

Terimlerdeki 1 sayısı	Mintermler	İkili gösterimi	İlişkili terimler	Bileşenler
1	$m_4$	0100 ✓	$m(4,12)$	-100
	$m_8$	1000 ✓	$m(8,10)$	10-0
2	$m_{10}$	1010 ✓	$m(8,12)$	1-00
	$m_{12}$	1100 ✓	$m(10,11)$	101-
3	$m_{11}$	1011 ✓	$m(11,15)$	1-11
4	$m_{15}$	1111 ✓		




**Komşu gruplar arasında ilişki olmadığından,  
buradaki tüm terimler asal bileşenleri oluşturur.**

## Örnek: (devam)

Asal bileşenleri oluşturduktan sonra, asıl asal bileşenlerin bulunması süreci başlar. Asıl asal bileşen tablosu, satırlarda asal bileşenleri, sütunlarda ise mintermleri barındırır. Asal bileşenler hangi mintermleri içeriyorsa ilgili sütuna 'X' işareti konur. Daha sonra sütunlar taranır ve sadece tek 'X' işaretinin olduğu hücreler işaretlenir.

	Bileşenler	4	8	10	11	12	15
m(4,12)	-100 (BC'D')	X				X	
m(8,10)	10-0 (AB'D')		X	X			
m(8,12)	1-00 (AC'D')		X			X	
m(10,11)	101- (AB'C)			X	X		
m(11,15)	1-11 (ACD)				X		X

 **Asıl asal bileşenler**

$$F(A,B,C,D) = BC'D' + ACD + AB'D'$$

## Örnek: (devam)

F fonksiyonunun içerdiği mintermleri Karnaugh haritasına taşıdığımızda, indirgemeye girebilecek olan terimlerin, tablo yöntemiyle bulduğumuz asal bileşenler olduğunu görmekteyiz. Asıl asal bileşenler ise, haritadaki en iyi indirgeme işlemine karşılık gelen asal bileşenlerdir.

AB \ CD	00	01	11	10
00		1	1	1
01				
11			1	1
10				1

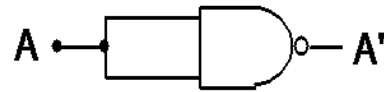
Gri ve kahverengi tonlamayla verilen gruplar, tablo yönteminde kesinlikle olması gereken asıl asal bileşenlere karşılık gelmektedir. Harita incelendiğinde  $m_4$  ve  $m_{15}$  mintermlerini örten başka bir grup olmadığı görülmektedir. Diğer asıl asal bileşen de mavi tonlamayla verilen gruptur.

# NAND veya NOR Kapılarının Üniversal Kapılar Olarak Kullanılması

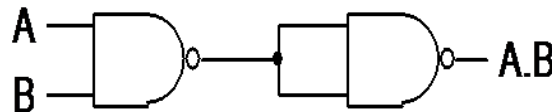
Şimdiye kadar ele alınan lojik ifadelerde NOT, AND ve OR kapıları kullanılarak kombinasyonel devrelerin tasarımı yapıldı. Aslında üniversal kapılar olarak bilinen NAND veya NOR kapılarıyla diğer kapı elemanlarını oluşturmak mümkündür. Böylece değişik tipte kapılar kullanmak yerine tek tipte kapılar kullanılarak, devrelerin oluşturulması daha ekonomik bir hale gelir.

## NAND kapıları kullanılarak diğer kapıların oluşturulması:

- $(A.A)' = A'$  olduğundan, NOT kapısı oluşturmak için NAND kapısının girişlerini birleştirmek gerekir.

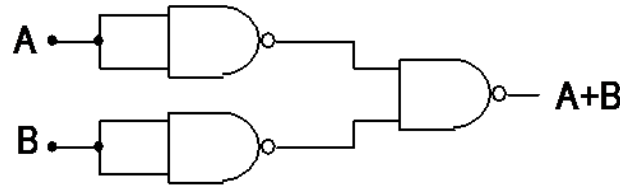


- $((A.B)')' = A.B$  olduğundan, NAND kapısının çıkışının değili alındığında AND kapısı elde edilir.

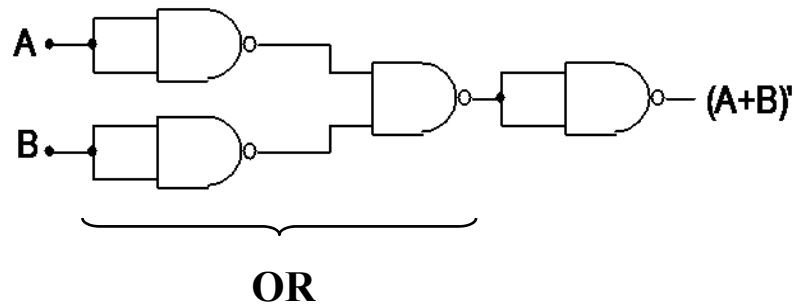


# NAND veya NOR Kapılarının Üniversal Kapılar Olarak Kullanılması

- $(A'.B')' = A+B$  olduğundan, A ve B girişlerinin değilleri alındıktan sonra NAND kapısının girişlerine uygulanırsa OR kapısı elde edilmiş olur.



- NOR işlevini elde etmek için de NAND kapılarından oluşturulan OR kapısının çıkışı değillenir.

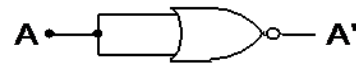




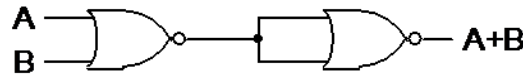
# NAND veya NOR Kapılarının Üniversal Kapılar Olarak Kullanılması

## NOR kapılarını kullanarak diğer kapıların oluşturulması:

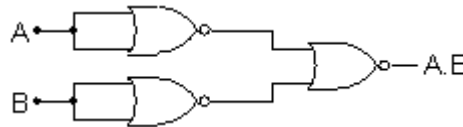
- $(A+A)' = A'$  olduğundan, NOT kapısı elde etmek için NAND kapısında olduğu gibi NOR kapısının girişlerinin birleştirilmesi gerekir.



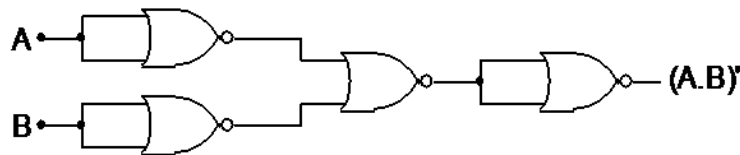
- $((A+B)')' = A+B$  olduğundan, OR kapısını elde etmek için NOR kapısının çıkışının değilin alınması gerekir.



- $(A'+B')' = A.B$  olduğundan, A ve B girişlerinin değilleri alındıktan sonra NOR kapısının girişlerine uygulanırsa AND kapısı elde edilmiş olur.

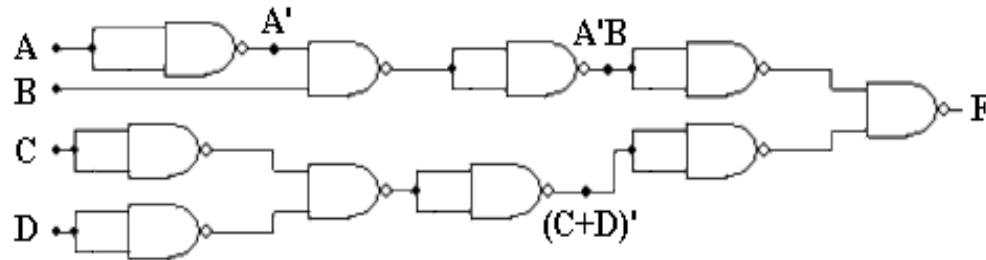


- NAND kapısı elde etmek için, NOR kapılarından oluşturulan AND kapısının çıkışının değilin alınması gerekir.

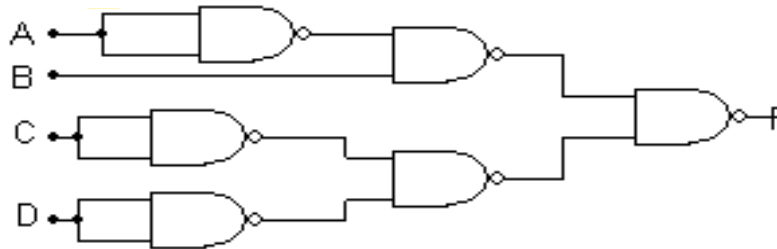


# Lojik İfadelerin NAND ile Gerçeklenmesi

**Örnek:**  $F(A,B,C,D) = A'B + (C+D)'$  lojik ifadesini NAND kapıları kullanarak gerçekleştirmek istersek NOT, AND, OR ve NOR işlemlerine karşılık gelen devreleri, yerlerine koymak gerekir.



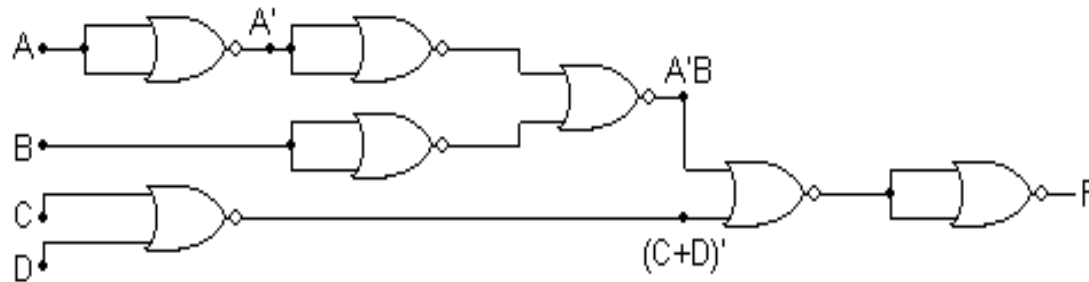
İki kere alınan değil işlemleri birbirini götürcektir;



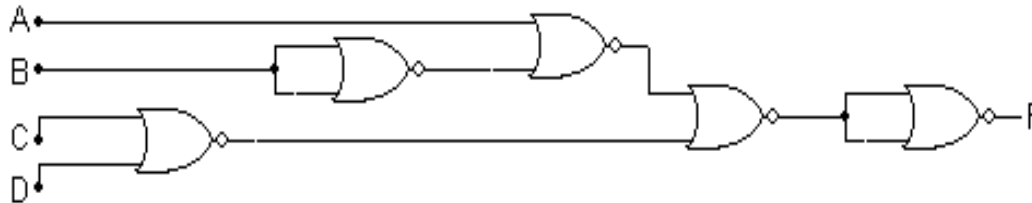
# Lojik İfadelerin NOR ile Gerçeklenmesi

**Örnek:** Aynı işlemi NOR kapıları kullanarak gerçekleştirelim.

$$F(A,B,C,D) = A'B + (C+D)'$$



İki kere alınan değil işlemleri birbirini götürcektir;



# Lojik İfadeleri Cebirsel Olarak NAND ya da NOR Formuna Getirme

---

Özellikle DeMorgan kuralları kullanılarak istenen form elde edilebilir.

- Çarpımlar toplamı şeklindeki ifadelerin iki kere değilleri alınmak suretiyle NAND formu elde edilebilir.
- Topamlar çarpımı şeklindeki ifadelerin de iki kere değilleri alınmak suretiyle NOR formu elde edilebilir.

**Örnek:**  $f(a,b,c,d) = ab' + c'd$  ifadesinin NAND formunu elde etmek istersek;

$$(f')' = f \text{ olduğundan, } f = [(ab' + c'd)']' = [(ab')' \cdot (c'd)']'$$

**Örnek:**  $f(a,b,c,d) = (a+b')(c'+d)$  ifadesinin NOR formunu elde etmek istersek;

$$(f')' = f \text{ olduğundan, } f = [(a+b')(c'+d)]' = [(a+b')' + (c'+d)']'$$

# Lojik İfadeleri Cebirsel Olarak NAND ya da NOR Formuna Getirme

**Örnek:**  $f(a,b,c,d) = ab' + c'd$  ifadesinin NOR formunu elde etmek istersek; ifadedeki tüm terimlerin ayrı ayrı iki kere değillerini almamız gerekir.

$$f(a,b,c,d) = [(ab')']' + [(c'd)']' = (a' + b)' + (c + d')'$$

Elde edilen ifadenin tamamının iki kere değilini alırsak;

$$f(a,b,c,d) = ([ (a' + b)' + (c + d')' ]')'$$

**Örnek:**  $f(a,b,c,d) = (a + b')(c' + d)$  ifadesinin NAND formunu elde etmek istersek, ifadedeki tüm terimlerin ayrı ayrı iki kere değillerini almamız gerekir.

$$f(a,b,c,d) = (a + b')(c' + d) = ([ (a + b') ]')' . ([ (c' + d) ]')' = (a'b)' . (cd')'$$

Elde edilen ifadenin tamamının iki kere değilini alırsak;

$$f(a,b,c,d) = ([ (a'b)' . (cd')' ]')'$$

# Mintermler Cinsinden Verilmiş Lojik İfadeleri, Karnaugh yardımıyla NAND ya da NOR Formuna Getirme

**Örnek:**  $f(a,b,c) = \sum(0,1,3,5,6,7)$  lojik ifadesini NAND ve NOR kapılarıyla ayrı ayrı gerçekleyelim.

Lojik ifadenin içerdiği mintermleri Karnaugh haritasına taşırsak;

$a \backslash bc$	00	01	11	10
0	1	1	1	
1		1	1	1

$f(a,b,c) = c + a'b' + ab$  olur. İfadeyi NAND formuna dönüştürmek istersek iki kere deęilini almamız yeterlidir.

$$f(a,b,c) = [(c + a'b' + ab)']' = [c' \cdot (a'b')' \cdot (ab)']'$$

Devreler genellikle 2 giriře sahip NAND kapıları kullanılarak gereklenir:

$$f(a,b,c) = [c' \cdot (a'b')' \cdot (ab)']' = [c' \cdot ((a'b')' \cdot (ab)')]'$$

## Örnek: (devam)

$f$  fonksiyonunu NOR kapıları kullanarak gerçeklemek istersek  $f$ 'nin tümleyenini kullanmamız daha uygun olur.  $f$ 'nin tümleyenini ( $f'$ ) bulmak için Karnaugh haritasındaki boş hücrelere 1 koymamız gerekir.

$f$

$a \backslash bc$	00	01	11	10
0	1	1	1	
1		1	1	1

$f'$

$a \backslash bc$	00	01	11	10
0				1
1	1			

$$f'(a,b,c) = ab'c' + a'bc'$$

$$f = (f')' = (a' + b + c).(a + b' + c)$$

Toplamlar çarpımı biçimindeki ifadenin iki kere değilini almak suretiyle NOR formuna getirebiliriz:

$$f = [(a' + b + c)' + (a + b' + c)']'$$

Bu ifadeyi 2 girişli NOR kapılarıyla gerçeklemek istersek,

$$f = ([((a' + b)')' + c]' + [((a + b')')' + c]')'$$