

# **Boolean Cebri ve Lojik Kapılar**

# Cebirsel Sistem

- Cebirin Anlamı Nedir?
- Matematik Sistem
  - Bir dizi eleman
  - Bir dizi işlem
  - Aksiyomlar ve Kanunlar
- Niçin önemlidir?
- Örnek: Doğal Sayılar üzerinde Aritmetik İşlemler
  - Eleman dizisi :  $N = \{1, 2, 3, 4, \dots\}$
  - İşlem Operatörü  $+, -, *$
  - Aksiyomlar : Birleşme, Dağılma, Kapalılık, Birim Eleman özellikleri .....
- *Binary : İki girişli işlem operatörü*
- *Unary Bir girişli işlem operatörü*

# TEMEL TANIMLAMALAR

- Küme : Aynı özelliğe sahip elemanlar topluluğu
  - $S$ : küme ,  $x$  ve  $y$ : eleman
  - Örnek:  $S = \{1, 2, 3, 4\}$ 
    - $x = 2$  ise o halde  $x \in S$ .
    - $y = 5$  ise o halde  $y \notin S$ .
- $S$  kümesi elemanları üzerinde tanımlanan İkili operatör  $S$  kümesinden alınan her bir eleman çiftini  $S$  kümesindeki bir tek elemana dönüştüren kuraldır.
  - Örneğin: Verilen bir  $S$  kümesinde  
 $a * b = c$  işlemi düşünelim  $*$  bir ikili işlem operatörü gösterir.  
Eğer  $(a, b)$  eleman çiftine  $*$  işlemi uygulandığında  $c$  elemanı elde ediliyor ise,  $(a, b, c \in S)$   
o halde  $*$   $S$  kümesinin bir ikili operatörüdür.
  - Diğer yandan.  $*$   $S$  in bir ikili operatörü değil ise  $(a, b \in S)$   
o halde  $c \notin S$ .

# TEMEL TANIMLAMALAR

- Çeşitli cebirsel yapıları formüle etmek için kullanılan en çok bilinen kanunlar şunlardır:

## 1. Kapalılık (Closure):

S kümesinden alınan her eleman çifti için bir ikili operatör S nin tek bir elemanını elde etme kuralını belirliyorsa bu S kümesi ikili operatöre göre kapalıdır.

- Örneğin: Doğal sayılar  $N=\{1,2,3,\dots\}$  + operatörüne göre kapalıdır.

$a+b = c$  Her  $a,b \in N$  için bir  $c \in N$  dönüşümü tanımlıdır.

– operatörü ise doğal sayılar kümesi için kapalılık özelliğini sağlamaz.

$2-3 = -1$       $2, 3 \in N$ , ancak  $(-1) \notin N$ .

## 2. Birleşme Kanunu (Associative law):

S kümesi üzerinde tanımlı \* ikili operatörü birleşme kanunu sağlaması için:

$$(x * y) * z = x * (y * z) \text{ her } x, y, z \in S$$

$(x+y)+z = x+(y+z)$  + işlemi de bu özelliği sağlar...

## 3. Değişme Kanunu (Commutative law) :S kümesi üzerinde tanımlı \* ikili operatörü değişme kanununu sağlaması için:

$$x * y = y * x \text{ for all } x, y \in S$$

$$x+y = y+x$$

# TEMEL TANIMLAMALAR

4. *Birim eleman (Identity element)*: Bir  $S$  kümesi elemanı  $e \in S$  aşağıdaki özelliği sağlarsa,  $*$  operatörüne göre bu küme bir birim elemana sahiptir.

$$\square e * x = x * e = x \quad \text{her } x \in S$$

$$\square 0 + x = x + 0 = x \quad \text{her } x \in I. \quad I = \{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}.$$

$$\square 1 * x = x * 1 = x \quad \text{her } x \in I. \quad I = \{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}.$$

5. *Ters Eleman (Inverse element)* : Aşağıda tanımlı özelliği sağlayan her  $x \in S$  için bir  $y \in S$  mevcutsa, bu  $S$  kümesi  $*$  operatörüne göre bir ters elemana sahiptir.

$$\square x * y = e$$

$\square +$  operatörü  $I$  üzerinde, ( $e = 0$ ), ters eleman özelliği gösterir

$\square a$  elemanın tersi  $(-a)$ , ( $a + (-a) = 0$ ).

6. *Dağılma Kanunu (Distributive law)*:

$*$  ve  $.$  operatörleri  $S$  kümesi üzerinde tanımlı olsunlar.  $*$  operatörü  $.$  operatörü üzerinde dağılma özelliği göstermesi için:

$$x * (y . z) = (x * y) . (x * z) \quad \text{Her } x, y, z \in S$$

# Boolean Cebirsel Sistemi

- İkili değerler için cebri tanımlamak gerekmektedir.
- George Boole 1854 yılında ilk defa ortaya atmıştır.
- Boolean Cebri için Huntington Postulaları (1904):
- $B = \{0, 1\}$  ve iki ikili işlem,  $+$  ve  $\cdot$ 
  - operator  $+$  ve operator  $\cdot$  için **Kapalılık** Özelliği
  - operator  $+$  için **birim eleman** 0 and operator  $\cdot$  için 1
  - $+$  ve  $\cdot$  operatörleri için **Değişim özelliği**

$$x+y = y+x, \quad x \cdot y = y \cdot x$$

$+$  operatörünün  $\cdot$  üzerine ve  $\cdot$  operatörünün  $+$  üzerine **dağılma özelliği**

$$x \cdot (y+z) = (x \cdot y) + (x \cdot z) \quad \text{and} \quad x + (y \cdot z) = (x+y) \cdot (x+z)$$

Her  $x$  elemanının **tümleyeni**  $x'$  olsun  $x+x'=1$ ,  $x \cdot x'=0$

En az 2 ayırık eleman  $x$  ve  $y$  vardır ki  $x \neq y$

# Boolean Cebri

## ■ Terminoloji:

- *Literal*: Bir değişken veya tümleyeni

$x, y, x', y'$

- *Çarpım Terimi (Product term)*:  $\cdot$  ile bağlanmış değişken grubu

- $x.y \quad x.y' \quad x'.y \quad x'.y'$

- *Toplam Terimi (Sum term)*:  $+$  ile bağlanmış değişken grubu

$x+y \quad x+y' \quad x'+y \quad x'+y'$

# İki-değerli Boolean Cebri Postulaları

- $B = \{0, 1\}$  ve iki ikili işlemler  $+$  ve  $\cdot$
- İşlem kuralları : AND , OR ve NOT.

**AND**

$x$	$y$	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

**OR**

$x$	$y$	$x+y$
0	0	0
0	1	1
1	0	1
1	1	1

**NOT**

$x$	$x'$
0	1
1	0

1. Kapalılık ( $+$  ve  $\cdot$ )

2. Birim elemanlar

(1)  $+: 0$

(2)  $\cdot : 1$



# İki-değerli Boolean Cebri Postulaları

3. Değişim kanunları

4. Dağılma kanunları

$x$	$y$	$z$	$y+z$	$x \cdot (y+z)$	$x \cdot y$	$x \cdot z$	$(x \cdot y) + (x \cdot z)$
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

# İki-değerli Boolean Cebri Postulaları

## 5. Tümleme

- $x+x'=1 \rightarrow 0+0'=0+1=1; 1+1'=1+0=1$
- $x \cdot x'=0 \rightarrow 0 \cdot 0'=0 \cdot 1=0; 1 \cdot 1'=1 \cdot 0=0$

## 6. İki ayrık eleman varlığı 1 ve 0, $0 \neq 1$

### ■ Not

- İki elemanlı bir küme
- $+$  : OR işlemi;  $\cdot$  : AND işlemi
- Tümleme operatörü NOT işlemi
- İkili Mantık iki değerli Boolean cebridir.

# İkililik Prensibi 'Duality'

- İkilik prensibi önemli bir kavramdır. Bir ifade Boolaen Cebrinde tanımlı ise, bu ifadenin ikili karşılığında tanımlı bir ifadedir.
- Bir ifadenin ikili karşılığını bulmak için, orijinal ifadede + yerine . , . yerine +, 0 yerine 1, 1 yerine 0 konulur.
- İfadenin duali
$$a + (bc) = (a + b)(a + c)$$
$$a(b + c) = ab + ac$$

# Temel Teoremler

**Table 2.1**

*Postulates and Theorems of Boolean Algebra*

Postulate 2	(a) $x + 0 = x$	(b) $x \cdot 1 = x$
Postulate 5	(a) $x + x' = 1$	(b) $x \cdot x' = 0$
Theorem 1	(a) $x + x = x$	(b) $x \cdot x = x$
Theorem 2	(a) $x + 1 = 1$	(b) $x \cdot 0 = 0$
Theorem 3, involution	$(x')' = x$	
Postulate 3, commutative	(a) $x + y = y + x$	(b) $xy = yx$
Theorem 4, associative	(a) $x + (y + z) = (x + y) + z$	(b) $x(yz) = (xy)z$
Postulate 4, distributive	(a) $x(y + z) = xy + xz$	(b) $x + yz = (x + y)(x + z)$
Theorem 5, DeMorgan	(a) $(x + y)' = x'y'$	(b) $(xy)' = x' + y'$
Theorem 6, absorption	(a) $x + xy = x$	(b) $x(x + y) = x$

# Boolean Teoremleri

- Huntington'un postulları bazı kuralları tanımlar.

Post. 1: closure

Post. 2: (a)  $x+0=x$ , (b)  $x \cdot 1=x$

Post. 3: (a)  $x+y=y+x$ , (b)  $x \cdot y=y \cdot x$

Post. 4: (a)  $x(y+z) = xy+xz$ ,  
(b)  $x+yz = (x+y)(x+z)$

Post. 5: (a)  $x+x'=1$ , (b)  $x \cdot x'=0$

- Cebirsel ifadeleri değiştirmek için daha fazla kural gerekir.

- Teoremler postullardan türetilir.

- Teorem Nedir?

- Postullardan türetilen bir formül yada ifade  
(veya diğer ispatlanmış teoremler)

- Boolean cebri temel teoremleri

- Teorem 1 (a):  $x + x = x$  (b):  $x \cdot x = x$

# İspat: $x+x=x$

- Sadece Huntington postulları kullanarak

## Huntington postulları

**Post. 2:** (a)  $x+0=x$ , (b)  $x \cdot 1=x$

**Post. 3:** (a)  $x+y=y+x$ , (b)  $x \cdot y=y \cdot x$

**Post. 4:** (a)  $x(y+z) = xy+xz$ ,  
(b)  $x+yz = (x+y)(x+z)$

**Post. 5:** (a)  $x+x'=1$ , (b)  $x \cdot x'=0$

- $x+x=x$  olduğunun gösterimi.

$$\begin{aligned}x+x &= (x+x) \cdot 1 && 2(b) \\&= (x+x)(x+x') && 5(a) \\&= x+xx' && 4(b) \\&= x+0 && 5(b) \\&= x && 2(a)\end{aligned}$$

- Teorem 1(a) yı ileriki ispatlarda kullanacağız.

# İspat : $x \cdot x = x$

Huntington postulates:

**Post. 2:** (a)  $x+0=x$ , (b)  $x \cdot 1=x$

**Post. 3:** (a)  $x+y=y+x$ , (b)  $x \cdot y=y \cdot x$

**Post. 4:** (a)  $x(y+z) = xy+xz$ ,  
(b)  $x+yz = (x+y)(x+z)$

**Post. 5:** (a)  $x+x'=1$ , (b)  $x \cdot x'=0$

**Th. 1:** (a)  $x+x=x$

■  $x \cdot x = x$  olduğunun gösterimi.

$$x \cdot x = xx+0 \quad 2(a)$$

$$= xx+xx' \quad 5(b)$$

$$= x(x+x') \quad 4(a)$$

$$= x \cdot 1 \quad 5(a)$$

$$= x \quad 2(b)$$

# İspat: $x+1=1$

## ■ Teorem 2(a): $x + 1 = 1$

$$\begin{aligned}
 x + 1 &= 1 \cdot (x + 1) & 2(b) \\
 &= (x + x')(x + 1) & 5(a) \\
 &= x + x'1 & 4(b) \\
 &= x + x' & 2(b) \\
 &= 1 & 5(a)
 \end{aligned}$$

### Huntington postulları:

**Post. 2:** (a)  $x+0=x$ , (b)  $x \cdot 1=x$

**Post. 3:** (a)  $x+y=y+x$ , (b)  $x \cdot y=y \cdot x$

**Post. 4:** (a)  $x(y+z) = xy+xz$ ,  
(b)  $x+yz = (x+y)(x+z)$

**Post. 5:** (a)  $x+x'=1$ , (b)  $x \cdot x'=0$

**Th. 1:** (a)  $x+x=x$

## ■ Teorem 2(b): $x \cdot 0 = 0$ ikililik prensibi

## ■ Teorem 3: $(x')' = x$

□ Postula 5,  $x$  tümleyeni tanımlar,  $x + x' = 1$  ve  $x \cdot x' = 0$

□  $x'$  tümleyeni  $x = (x')'$



# Kapsama Özelliği (Covering)

Teorem 6(a):  $x + xy = x$

$$\begin{aligned}
 x + xy &= x \cdot 1 + xy && 2(b) \\
 &= x(1 + y) && 4(a) \\
 &= x(y + 1) && 3(a) \\
 &= x \cdot 1 && 2(a) \\
 &= x && 2(b)
 \end{aligned}$$

## Huntington postulları:

**Post. 2:** (a)  $x+0=x$ , (b)  $x \cdot 1=x$

**Post. 3:** (a)  $x+y=y+x$ , (b)  $x \cdot y=y \cdot x$

**Post. 4:** (a)  $x(y+z) = xy+xz$ ,  
(b)  $x+yz = (x+y)(x+z)$

**Post. 5:** (a)  $x+x'=1$ , (b)  $x \cdot x'=0$

**Th. 1:** (a)  $x+x=x$

Teorem 6(b):  $x(x + y) = x$  *İkililik Özelliği*

Doğruluk Tablosu kullanark (diğer bir ispat yolu )

$x$	$y$	$xy$	$x+xy$
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

# DeMorgan Teorem

- Teorem 5(a):  $(x + y)' = x'y'$
- Teorem 5(b):  $(xy)' = x' + y'$
- Doğruluk Tablosu yardımıyla ispat

$x$	$y$	$x'$	$y'$	$x+y$	$(x+y)'$	$x'y'$	$xy$	$x'+y'$	$(xy)'$
0	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	0	0	1	1
1	0	0	1	1	0	0	0	1	1
1	1	0	0	1	0	0	1	0	0

# Yutma Teoremi

1.  $xy + x'z + yz = xy + x'z$
2.  $(x+y) \cdot (x'+z) \cdot (y+z) = (x+y) \cdot (x'+z) \text{ -- (dual)}$

## ■ İspatı:

$$\begin{aligned} xy + x'z + yz &= xy + x'z + (x+x')yz \\ &= xy + x'z + xyz + x'yz \\ &= (xy + xyz) + (x'z + x'zy) \\ &= xy + x'z \end{aligned}$$

(İkililik prensibi ile diğer teoremi (2) ispatlayınız ).

- Boolean İfadelerini değerlendirmek için kullanılır:

- Parantez

- NOT

- AND

- OR

- Örnekler:

$$x y' + z$$

$$(x y + z)'$$

# Boolean Fonksiyonları:

## ■ Bir Boolean Fonksiyonu

- İkili Değişkenler
- İkili Operatörler OR ve AND
- Tekil Operatör NOT
- Parantezler

## ■ Örnekler:

- $F_1 = x y z'$
- $F_2 = x + y'z$
- $F_3 = x' y' z + x' y z + x y'$
- $F_4 = x y' + x' z$

# Boolean Fonksiyonları

- $2^n$  girişli Doğruluk Tablosu (truth table)

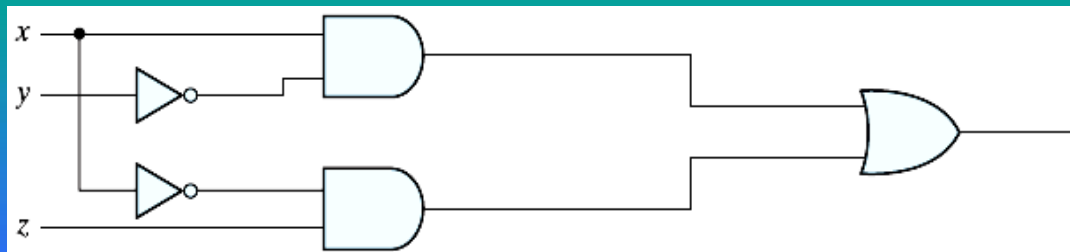
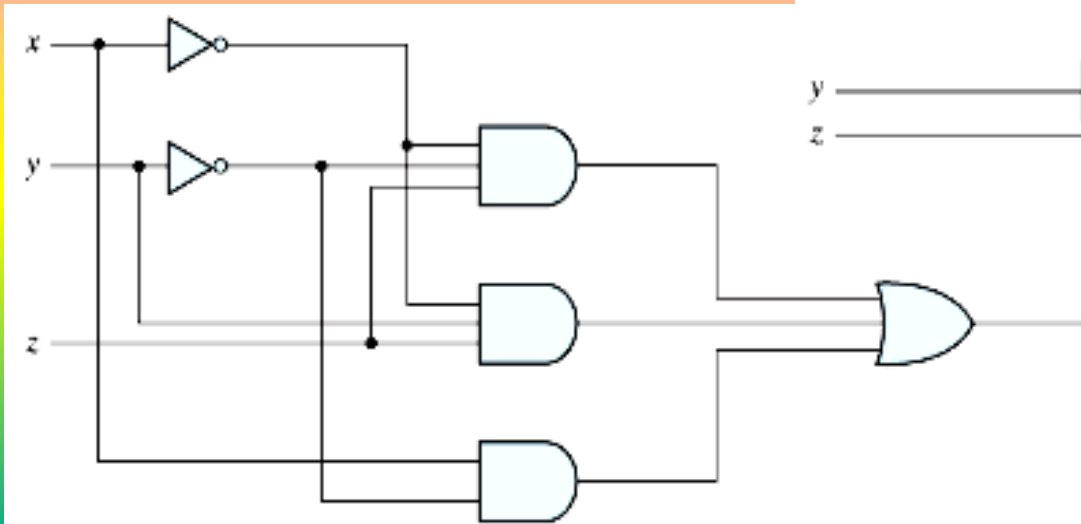
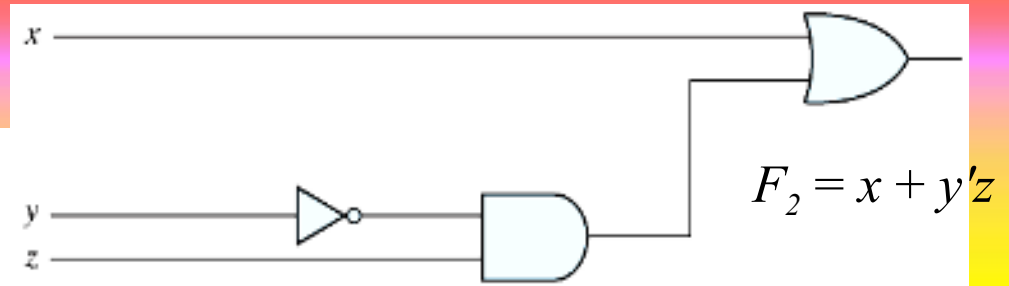
$x$	$y$	$z$	$F_1$	$F_2$	$F_3$	$F_4$
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	0	0	0	0
0	1	1	0	0	1	1
1	0	0	0	1	1	1
1	0	1	0	1	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	0

- İki Boolean fonksiyonu aynı doğruluk tablosuna sahipse bu fonksiyonların cebirsel ifadeleri de aynıdır.
- $F_3 = F_4$

# Boolean Fonksiyonları

## ■ Lojik kapılarla gerçekleştirme

■  $F_4$  çok daha ekonomiktir.



$$F_3 = x'y'z + x'yz + xy'$$

$$F_4 = xy' + x'z$$

# Cebirsel Manüplasyon

- Boolean ifadelerini minimize etmek için,
  - **Literal**: Tek değişken (bir kapının bir girişi)
  - **Term**: Bir kapılı gerçekleştirme
  - Literal veya term sayısını minimize etmek → daha az kapılı bir devre
  - Belirli bir kuralı yok

## ■ Example 2.1

1.  $x(x'+y) = xx' + xy = 0 + xy = xy$
2.  $x + x'y = (x+x')(x+y) = 1(x+y) = x+y$
3.  $(x+y)(x+y') = x + xy + xy' + yy' = x(1+y+y') = x$
4.  $xy + x'z + yz = xy + x'z + yz(x+x') = xy + x'z + yzx + yzx' = xy(1+z) + x'z(1+y) = xy + x'z$
5.  $(x+y)(x'+z)(y+z) = (x+y)(x'+z)$ , ikililik prensibi (duality).

Yutma teoremi



# Bir Fonksiyonun Tümleyeni

□ By DeMorgan's theorem

$$\square (A+B+C)' = (A+X)'$$

let  $B+C = X$

$$= A'X'$$

by theorem 5(a) (DeMorgan's)

$$= A'(B+C)'$$

substitute  $B+C = X$

$$= A'(B'C')$$

by theorem 5(a) (DeMorgan's)

$$= A'B'C'$$

by theorem 4(b) (associative)

■ *Genelleştirme: Bir fonksiyon ifadesinde AND ve OR operatörleri yer değiştirilerek ve her değişkenin tümleyeni alınarak fonksiyonun tümleyeni bulunur.*

$$\square (A+B+C+D+ \dots +F)' = A'B'C'D' \dots F'$$

$$\square (ABCD \dots F)' = A' + B' + C' + D' \dots + F'$$

# Örnekler

## ■ Örnek 2.2

$$\square F_1' = (x'yz' + x'y'z)' = (x'yz')' (x'y'z)' = (x+y'+z) (x+y+z')$$

$$\begin{aligned}\square F_2' &= [x(y'z'+yz)]' = x' + (y'z'+yz)' = x' + (y'z')' (yz)' \\ &= x' + (y+z) (y'+z') \\ &= x' + yz' + y'z\end{aligned}$$

## ■ Örnek 2.3: daha basit bir prosedür

□ Fonksiyonun dualini al ve her değişkeni tümleyenini koy.

1.  $F_1 = x'yz' + x'y'z.$

Dual  $F_1 \quad (x'+y+z') (x'+y'+z).$

Değişken tümleme :  $(x+y'+z)(x+y+z') = F_1'$

2.  $F_2 = x(y'z' + yz).$

Dual  $F_2 \quad x+(y'+z') (y+z).$

Değişken tümleme :  $x'+(y+z)(y'+z') = F_2'$

## 2.6 Kanonik ve Standard Formlar

Standart Çarpımlar (***Minterms***) ve Standart Toplamlar (***Maxterms***)

- Minterm: Normal veya tümlleme formunda bulunan tüm değişkenleri içeren bir AND terimidir.
  - Örnek, 2 ikili değişken  $x$  ve  $y$  ile oluşturulan mintermler
    - $xy, xy', x'y, x'y'$
  - $n$  değişken ile  $2^n$  minterm oluşturulabilir.
- Maksterm : Bir OR terimi
  - Örnek, 2 ikili değişken  $x$  ve  $y$  ile oluşturulan makstermler
    - $x+y, x+y', x'+y, x'+y'$
  - $n$  değişken ile  $2^n$  maksterm oluşturulabilir.

# Minterm ve Makstermler

- Her *maksterm* ona karşı gelen mintermin tümleyenidir ve tersi de doğrudur.

**Table 2.3**

*Minterms and Maxterms for Three Binary Variables*

<i>x</i>	<i>y</i>	<i>z</i>	Minterms		Maxterms	
			Term	Designation	Term	Designation
0	0	0	$x'y'z'$	$m_0$	$x + y + z$	$M_0$
0	0	1	$x'y'z$	$m_1$	$x + y + z'$	$M_1$
0	1	0	$x'yz'$	$m_2$	$x + y' + z$	$M_2$
0	1	1	$x'yz$	$m_3$	$x + y' + z'$	$M_3$
1	0	0	$xy'z'$	$m_4$	$x' + y + z$	$M_4$
1	0	1	$xy'z$	$m_5$	$x' + y + z'$	$M_5$
1	1	0	$xyz'$	$m_6$	$x' + y' + z$	$M_6$
1	1	1	$xyz$	$m_7$	$x' + y' + z'$	$M_7$

# Minterm ve Makstermler

- Bir Boolean fonksiyonu doğruluk tablosu veya minterm (veya maksterm) cinsinden ifade edilebilir:
  - Minterm toplamları
  - $f_1 = x'y'z + xy'z' + xyz = m_1 + m_4 + m_7$
  - $f_2 = x'yz + xy'z + xyz' + xyz = m_3 + m_5 + m_6 + m_7$

**Table 2.4**

*Functions of Three Variables*

<b>x</b>	<b>y</b>	<b>z</b>	<b>Function <math>f_1</math></b>	<b>Function <math>f_2</math></b>
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

# Minterm ve Makstermler

- Bir Boolean fonksiyonun tümleyeni
  - fonksiyonu 0 yapan mintermler
  - $f_1' = m_0 + m_2 + m_3 + m_5 + m_6 = x'y'z' + x'yz' + x'yz + xy'z + xyz'$
  - $f_1 = (f_1)'$
  - $= (x+y+z)(x+y'+z)(x+y'+z')(x'+y+z')(x'+y'+z) = M_0 M_2 M_3 M_5 M_6$
  - $f_2 = (x+y+z)(x+y+z')(x+y'+z)(x'+y+z) = M_0 M_1 M_2 M_4$
- Bir Boolean fonksiyonu aşağıdaki şekilde ifade edilebilir:
  - Mintermlerin toplamı. (toplama OR işlemi anlamında)
  - Makstermlerin çarpımı (çarpım AND işlemi anlamında).
  - Bu durumdaki Boolean Fonksiyonları Kanonik Formdadırlar.

# Mintermlerin Toplamı

- $n$  Boolean değişkenli fonksiyon  $2^n$  minterm ile ifade edilebilir.
- Örnek 2.4:  $F = A + BC'$  fonksiyonunu.
  - $F = A + B'C = A(B + B') + B'C = AB + AB' + B'C = AB(C + C') + AB'(C + C') + (A + A')B'C = ABC + ABC' + AB'C + AB'C' + A'B'C$
  - $F = A'B'C + AB'C' + AB'C + ABC' + ABC = m_1 + m_4 + m_5 + m_6 + m_7$
  - $F(A, B, C) = \Sigma(1, 4, 5, 6, 7)$
- Yada doğruluk tablosu yapılarak aynı ifade doğrudan elde edilir.

**Table 2.5**

*Truth Table for  $F = A + B'C$*

<b>A</b>	<b>B</b>	<b>C</b>	<b>F</b>
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

# Makstermlerin çarpımı

□ *Dağılma özelliğini kullanarak:*

□  $x + yz = (x + y)(x + z) = (x+y+zz')(x+z+yy') = (x+y+z)(x+y+z')$   
 $(x+y'+z)$

■ Örnek 2.5:  $F = xy + x'z$

□  $F = xy + x'z = (xy + x')(xy + z) = (x+x')(y+x')(x+z)(y+z) = (x'+y)$   
 $(x+z)(y+z)$

□  $x'+y = x' + y + zz' = (x'+y+z)(x'+y+z')$

□  $F = (x+y+z)(x+y'+z)(x'+y+z)(x'+y+z') = M_0M_2M_4M_5$

□  $F(x, y, z) = \Pi(0, 2, 4, 5)$



# Kanonik Formlar arasındaki dönüşüm

- Minterm toplamı olarak ifade edilen bir fonksiyonun tümleyeni orijinal fonksiyonda görünmeyen mintermlerin toplamına eşittir:
  - $F(A, B, C) = \Sigma(1, 4, 5, 6, 7)$
  - Buradan ,  $F'(A, B, C) = \Sigma(0, 2, 3)$
  - DeMorgan teoremi ile
$$F(A, B, C) = \Pi(0, 2, 3)$$
$$F'(A, B, C) = \Pi(1, 4, 5, 6, 7)$$
  - $m_j' = M_j$
  - Mintermlerin toplamı = Makstermlerin çarpımı
  - $\Sigma$  ve  $\Pi$  sembolleri değiştirilerek fonksiyon tümleyeni bulunabilir
  - $\Sigma$  fonksiyonu 1 yapan mintermlerden
  - $\Pi$  fonksiyonu 0 yapan makstermlerden

## ■ Örnek

□  $F = xy + x'z$

□  $F(x, y, z) = \Sigma(1, 3, 6, 7)$

□  $F(x, y, z) = \Pi(0, 2, 4, 6)$

**Table 2.6**

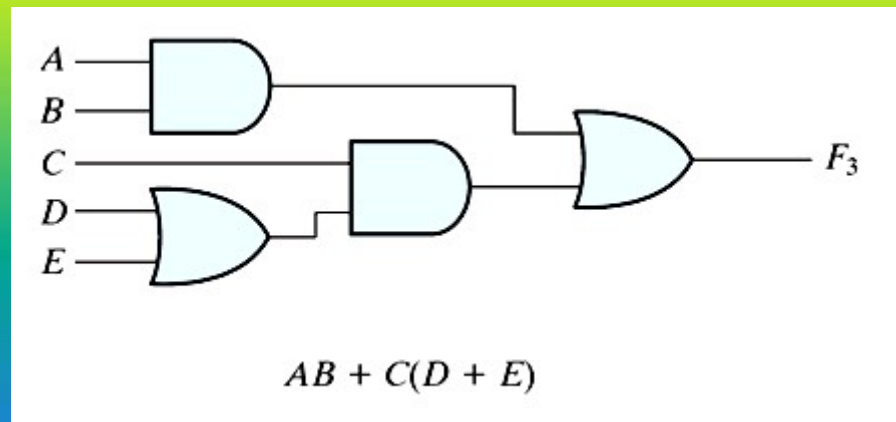
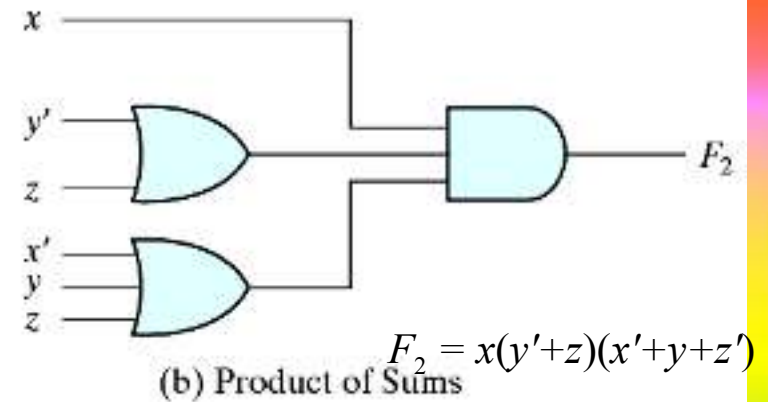
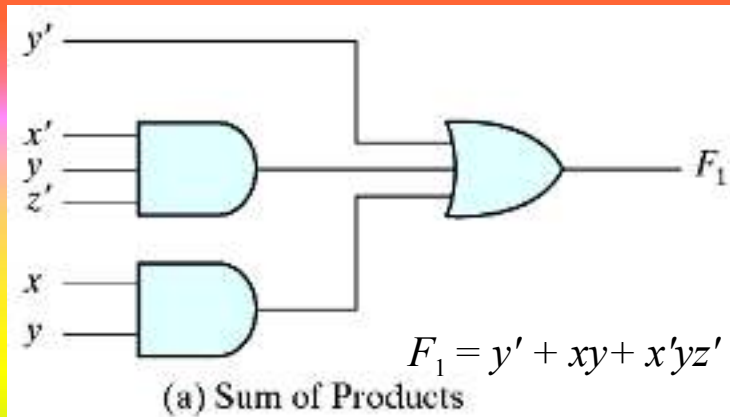
*Truth Table for  $F = xy + x'z$*

<b>x</b>	<b>y</b>	<b>z</b>	<b>F</b>
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

# Standart Formlar

- Kanonik formlar minterm veya maksterm formunda ifadedir.
- Standard formlar: 1, 2 veya herhangi sayıda değişkenden oluşan fonksiyon terimlerini içerir.
  - Çarpımların Toplamı:  $F_1 = y' + xy + x'yz'$
  - Toplamların Çarpımı :  $F_2 = x(y'+z)(x'+y+z')$
  - $F_3 = A'B'CD + ABC'D'$

# Gerçekleme



## 2.7 Diğer Lojik İşlemler

- $n$  değişkenli doğruluk tablosunda  $2^n$  satır
- $2^{2^n}$  fonksiyon tanımlanabilir
- $n=2$  için 16 fonksiyon

**Table 2.7**

*Truth Tables for the 16 Functions of Two Binary Variables*

$x$	$y$	$F_0$	$F_1$	$F_2$	$F_3$	$F_4$	$F_5$	$F_6$	$F_7$	$F_8$	$F_9$	$F_{10}$	$F_{11}$	$F_{12}$	$F_{13}$	$F_{14}$	$F_{15}$
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

# Boolean İfadeleri

**Table 2.8**

*Boolean Expressions for the 16 Functions of Two Variables*

Boolean Functions	Operator Symbol	Name	Comments
$F_0 = 0$		Null	Binary constant 0
$F_1 = xy$	$x \cdot y$	AND	$x$ and $y$
$F_2 = xy'$	$x/y$	Inhibition	$x$ , but not $y$
$F_3 = x$		Transfer	$x$
$F_4 = x'y$	$y/x$	Inhibition	$y$ , but not $x$
$F_5 = y$		Transfer	$y$
$F_6 = xy' + x'y$	$x \oplus y$	Exclusive-OR	$x$ or $y$ , but not both
$F_7 = x + y$	$x + y$	OR	$x$ or $y$
$F_8 = (x + y)'$	$x \downarrow y$	NOR	Not-OR
$F_9 = xy + x'y'$	$(x \oplus y)'$	Equivalence	$x$ equals $y$
$F_{10} = y'$	$y'$	Complement	Not $y$
$F_{11} = x + y'$	$x \subset y$	Implication	If $y$ , then $x$
$F_{12} = x'$	$x'$	Complement	Not $x$
$F_{13} = x' + y$	$x \supset y$	Implication	If $x$ , then $y$
$F_{14} = (xy)'$	$x \uparrow y$	NAND	Not-AND
$F_{15} = 1$		Identity	Binary constant 1

## 2.8 Dijital Lojik Kapılar

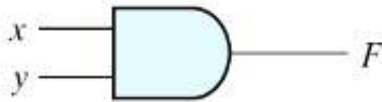

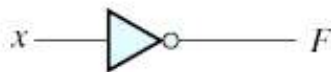
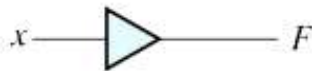
- Boolean ifadeleri: AND, OR ve NOT işlemleri
- Diğer lojik işlemlerin kapılarını oluşturulması
  - Ekonomik ve gerçekleştirilebilirlik
  - Kapı girişlerini genişletmek imkanı;
  - İkili işlemlerin temel özellikleri (değişme ve birleşme);
  - Boolean fonksiyonunu gerçeklemek için kapı yeteneği.

# Standart Kapılar


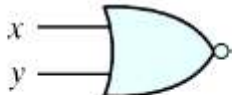
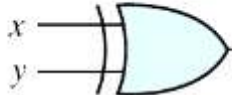
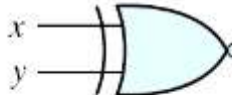
- 16 adet fonksiyonu hatırlayın (38 nolu slayt)
- Sabit fonksiyonlar( $F_0$  ve  $F_{15}$ ).
- 2 kez tekrarlı fonksiyonlar ( $F_4$ ,  $F_5$ ,  $F_{10}$  and  $F_{11}$ ).
- $F_4$  ile  $F_2$ ,  $F_5$   $F_3$ ,  $F_{10}$   $F_{12}$ ,  $F_{11}$  ile  $F_{13}$
- ( $F_2$ ) ve ( $F_{13}$ ) değişme ve birleşme özelliğine uymuyor.
- ( $F_2$ )  $x.y' \neq y.x'$
- Standart Kapılar: Tümleme ( $F_{12}$ ), transfer ( $F_3$ ), AND ( $F_1$ ), OR ( $F_7$ ), NAND ( $F_{14}$ ), NOR ( $F_8$ ), XOR ( $F_6$ ), ve eşdeğerlik (XNOR) ( $F_9$ )
- Transfer: buffer (kapının diğer kapıları sürme yeteneğini artırır.).



# Lojik Kapı Özeti

Name	Graphic symbol	Algebraic function	Truth table															
AND		$F = xy$	<table><tr><th><math>x</math></th><th><math>y</math></th><th><math>F</math></th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	$x$	$y$	$F$	0	0	0	0	1	0	1	0	0	1	1	1
$x$	$y$	$F$																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OR		$F = x + y$	<table><tr><th><math>x</math></th><th><math>y</math></th><th><math>F</math></th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	$x$	$y$	$F$	0	0	0	0	1	1	1	0	1	1	1	1
$x$	$y$	$F$																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
Inverter		$F = x'$	<table><tr><th><math>x</math></th><th><math>F</math></th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	$x$	$F$	0	1	1	0									
$x$	$F$																	
0	1																	
1	0																	
Buffer		$F = x$	<table><tr><th><math>x</math></th><th><math>F</math></th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	$x$	$F$	0	0	1	1									
$x$	$F$																	
0	0																	
1	1																	

# Lojik Kapı Özeti

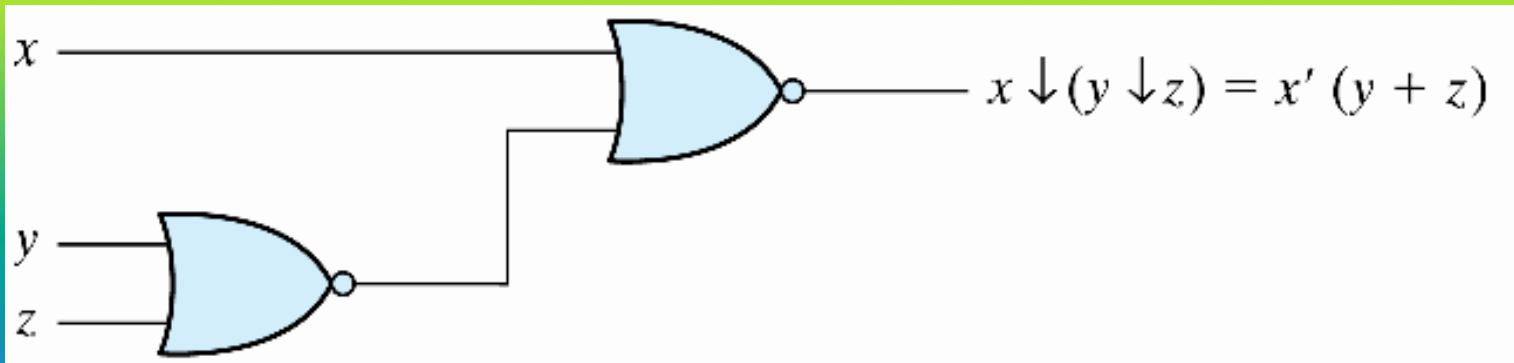
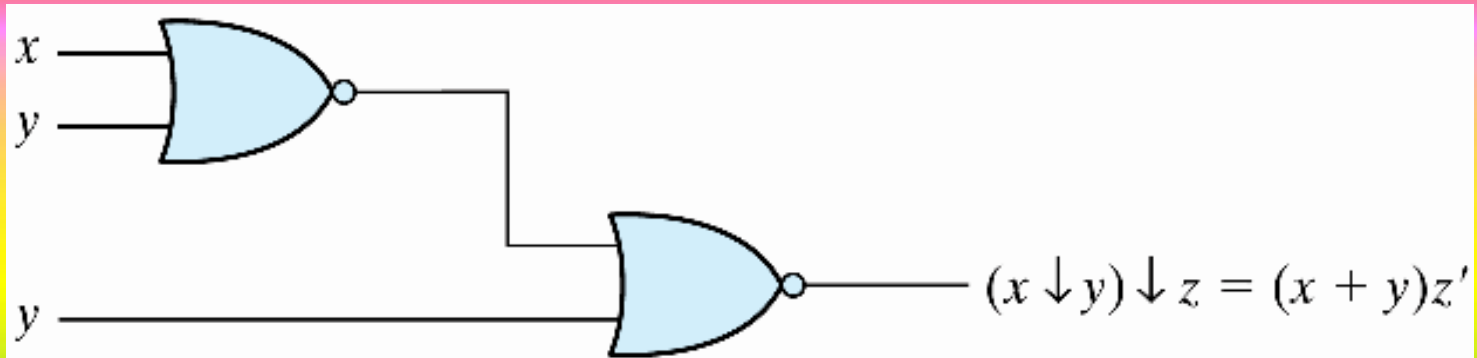
NAND	 $F = (xy)'$	<table> <tr> <th><math>x</math></th><th><math>y</math></th><th><math>F</math></th></tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table>	$x$	$y$	$F$	0	0	1	0	1	1	1	0	1	1	1	0
$x$	$y$	$F$															
0	0	1															
0	1	1															
1	0	1															
1	1	0															
NOR	 $F = (x + y)'$	<table> <tr> <th><math>x</math></th><th><math>y</math></th><th><math>F</math></th></tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table>	$x$	$y$	$F$	0	0	1	0	1	0	1	0	0	1	1	0
$x$	$y$	$F$															
0	0	1															
0	1	0															
1	0	0															
1	1	0															
Exclusive-OR (XOR)	 $F = xy' + x'y$ $= x \oplus y$	<table> <tr> <th><math>x</math></th><th><math>y</math></th><th><math>F</math></th></tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table>	$x$	$y$	$F$	0	0	0	0	1	1	1	0	1	1	1	0
$x$	$y$	$F$															
0	0	0															
0	1	1															
1	0	1															
1	1	0															
Exclusive-NOR or equivalence	 $F = xy + x'y'$ $= (x \oplus y)'$	<table> <tr> <th><math>x</math></th><th><math>y</math></th><th><math>F</math></th></tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	$x$	$y$	$F$	0	0	1	0	1	0	1	0	0	1	1	1
$x$	$y$	$F$															
0	0	1															
0	1	0															
1	0	0															
1	1	1															

# Çoklu Girişler

- Çoklu giriş genişletme
  - Bir kapı çoklu girişlere genişletilebilir:
  - İkili işlemleri değişme ve birleşme özelliklerini sağlıyorsa;
  - AND ve OR değişme ve birleşme özelliklerini sağlar
    - OR
      - $x+y = y+x$
      - $(x+y)+z = x+(y+z) = x+y+z$
    - AND
      - $xy = yx$
      - $(x y)z = x(y z) = x y z$

# Çoklu Girişler

- NAND ve NOR değişme özelliğini sağlar fakat birleşme özelliğine uymaz → Genişletilemez

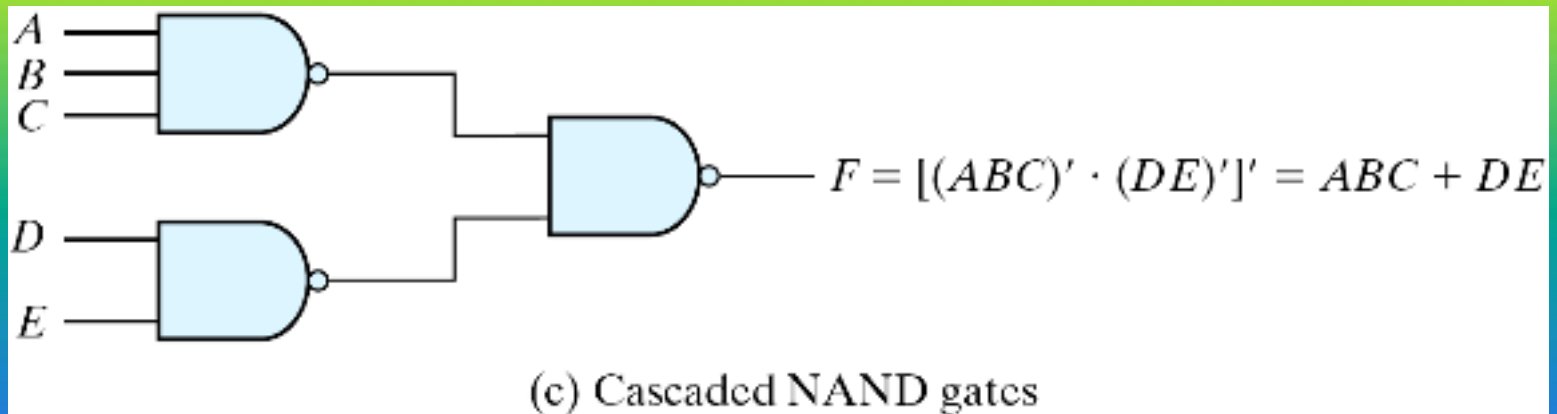
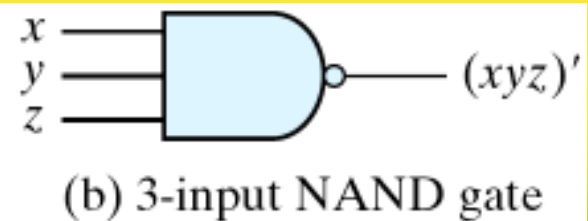
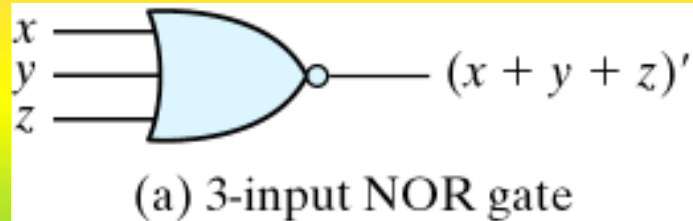


NOR işlemi birleşme özelliğini sağlamaz

$$(x \downarrow y) \downarrow z \neq x \downarrow (y \downarrow z)$$

# Çoklu Girişler

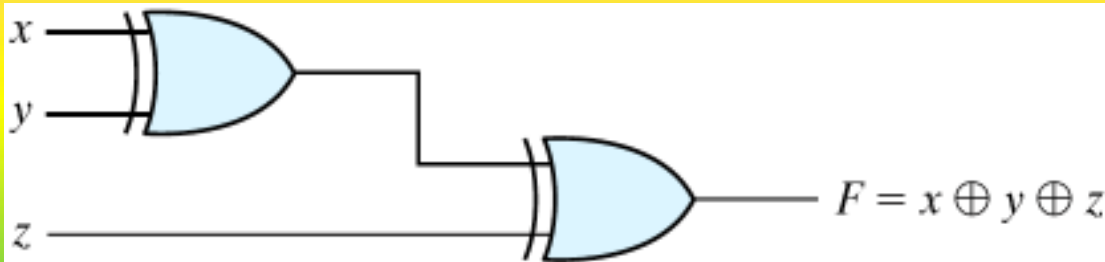
- ❑ Çok girişli NOR = OR kapısının tümleyeni
- ❑ Çok girişli NAND = AND kapısının tümleyeni
- ❑ Kaskad NAND = çarpımların toplamı
- ❑ Kaskad NOR = toplamaların çarpımı



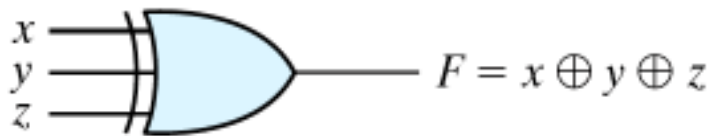
Çok girişli ve Kaskad NOR ve NAND kapıları

# Çoklu Girişler

- ❑ XOR ve XNOR kapıları değişme ve birleşme özelliklerini sağlarlar.
- ❑ Çok girişli XOR gerçekleştirilebilir
- ❑ XOR bir teklik belirleme fonksiyonudur.:
- ❑ Girişleri tek sayıda 1 içeriyorsa çıkış 1



(a) Using 2-input gates



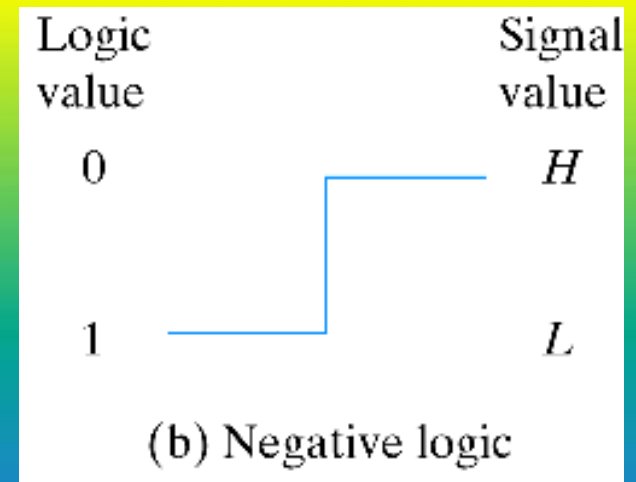
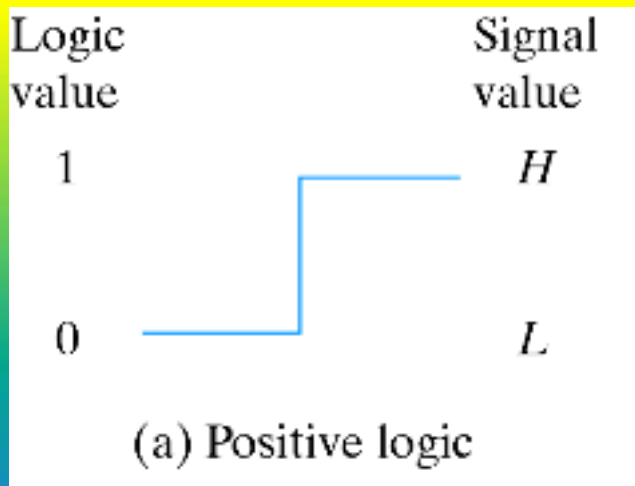
(b) 3-input gate

$x$	$y$	$z$	$F$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(c) Truth table

# Pozitif ve Negatif Lojik

- Pozitif ve Negatif Lojik
  - İki işaret değeri  $\Leftrightarrow$  iki lojik seviye
  - Pozitif lojik  $H=1$ ;  $L=0$
  - Negatif lojik  $H=0$ ;  $L=1$

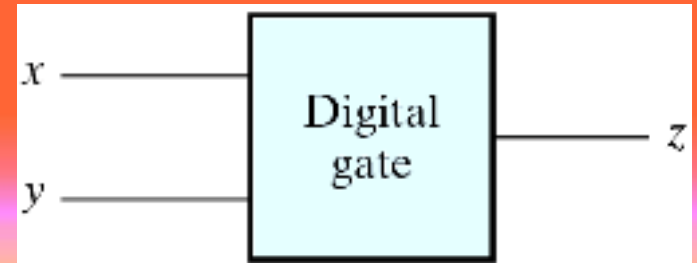


Lojik Seviyeler

# Pozitif ve Negatif Lojik

$x$	$y$	$z$
$L$	$L$	$L$
$L$	$H$	$L$
$H$	$L$	$L$
$H$	$H$	$H$

(a) Truth table with  $H$  and  $L$



(b) Gate block diagram

TTL kapı düşünelim

Bir pozitif lojik AND kapısı

Bir negatif lojik OR kapısı

Bu derste pozitif lojik kullanılacaktır.

$x$	$y$	$z$
0	0	0
0	1	0
1	0	0
1	1	1

(c) Truth table for positive logic



(d) Positive logic AND gate

$x$	$y$	$z$
1	1	1
1	0	1
0	1	1
0	0	0

(e) Truth table for negative logic



(f) Negative logic OR gate



# 2.9 Entegre Devreler

## Tümleştirme Seviyesi

- IC (a chip)
- Örnekler
  - Küçük çapta tümleştirme  
Small-scale Integration (SSI):  $< 10$  kapı
  - Orta çapta tümleştirme  
Medium-scale Integration (MSI):  $10 \sim 100$  kapı
  - Büyük çapta tümleştirme  
Large-scale Integration (LSI):  $100 \sim xk$  kapı
  - Çok büyük çapta tümleştirme  
Very Large-scale Integration (VLSI):  $> xk$  kapı
- VLSI
  - Küçük boyut                      Düşük güç tüketimi
  - Düşük fiyat                      Yüksek güvenilirlik
  - Low power consumption      Yüksek hız

# Dijital Lojik Aileler

- Kullandıkları Devre teknolojisi
  - TTL: transistor-transistor logic
  - ECL: emitter-coupled logic (high speed, high power consumption)
  - MOS: metal-oxide semiconductor (NMOS, high density)
  - CMOS: complementary MOS (low power)
  - BiCMOS: high speed, high density

# Dijital Lojik Aileler

- ❑ KARAKTERİSTİKLERİ
- ❑ Çıkış Yelpazesi: (Fan-out ) Tipik bir kapı çıkışının sürebileceği kapı girişi sayısı
- ❑ Güç tüketimi (Power dissipation)
- ❑ İletim Gecikmesi (propagation delay): işaretin girişten çıkışa ortalama iletim süresi
- ❑ Gürültü Bağışıklığı (Noise margin):
- ❑ Devre çıkışında istenmeyen değişime neden olan dış kaynaklı minimum gürültü gerilimi

# CAD

- CAD – Computer-Aided Design
  - Millions of transistors
  - Computer-based representation and aid
  - Automatic the design process
  - Design entry
    - Schematic capture
    - HDL – Hardware Description Language
      - Verilog, VHDL
  - Simulation
  - Physical realization
    - ASIC, FPGA, PLD

# Chip Design

- Why is it better to have more gates on a single chip?
  - Easier to build systems
  - Lower power consumption
  - Higher clock frequencies
- What are the drawbacks of large circuits?
  - Complex to design
  - Chips have design constraints
  - Hard to test
- Need tools to help develop integrated circuits
  - Computer Aided Design (CAD) tools
  - Automate tedious steps of design process
  - Hardware description language (HDL) describe circuits
  - VHDL (see the lab) is one such system