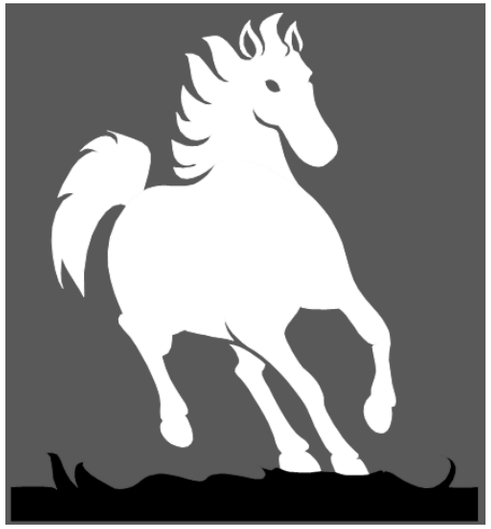
Digital Logic Design

"Combinational Logics"

Dr. Cahit Karakuş, February-2018



Basics

Digital Logic Basics

- Hardware consists of a few simple building blocks
  - These are called *logic gates*
    - AND, OR, NOT, …
    - NAND, NOR, XOR, …
- Logic gates are built using transistors
    - NOT gate can be implemented by a single transistor
    - AND gate requires 3 transistors
- Transistors are the fundamental devices
    - Pentium consists of 3 million transistors
    - Compaq Alpha consists of 9 million transistors
    - Now we can build chips with more than 100 million transistors

Temel Kavramlar -1

- Number of functions
  - With *N* logical variables, we can define
    2^{2^N} functions
  - Some of them are useful
    - AND, NAND, NOR, XOR, …
  - Some are not useful:
    - Output is always 1
    - Output is always 0
  - “Number of functions” definition is useful in proving completeness property

Temel Lojik Kapılar -1

- Simple gates
  - AND
  - OR
  - NOT
- Functionality can be expressed by a truth table
  - A truth table lists output for each possible input combination
- Precedence
  - NOT > AND > OR
  - F = A B + A B
    = (A (B)) + ((A) B)

Gate	Symbol	Truth-Table	Expression
NAND		X Y Z 0 0 1 0 1 1 1 0 1 1 1 0	$Z = \overline{X \cdot Y}$
AND		X Y Z 0 0 0 0 1 0 1 0 0 1 1 1	$Z = X \cdot Y$
NOR		X Y Z 0 0 1 0 1 0 1 0 0 1 1 0	$Z = \overline{X + Y}$
OR		X Y Z 0 0 0 0 1 1 1 0 1 1 1 1	$Z = X + Y$

Temel Lojik Kapılar -2

- Additional useful gates
  - NAND
  - NOR
  - XOR
- NAND = AND + NOT
- NOR = OR + NOT
- XOR implements exclusive-OR function
- NAND and NOR gates require only 2 transistors
  - AND and OR need 3 transistors!

XOR

$(X \oplus Y)$

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

$Z = X\overline{Y} + \overline{X}Y$

X or Y but not both ("inequality", "difference")

XNOR

$\overline{(X \oplus Y)}$

X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	1

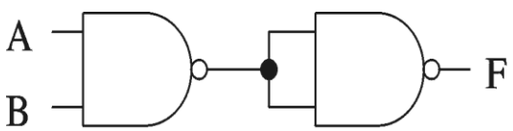
$Z = \overline{X}\,\overline{Y} + XY$

X and Y the same ("equality")

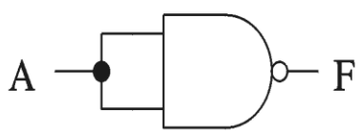
Widely used in arithmetic structures such as adders and multipliers

Temel Kavramlar -3

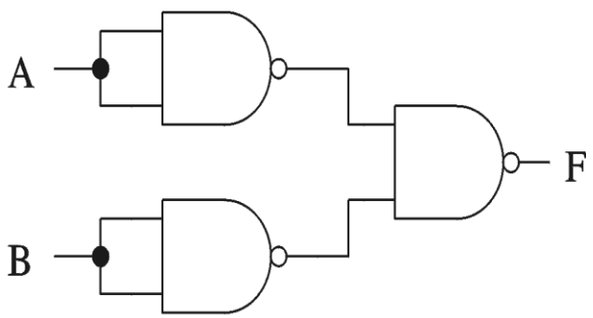
- Proving NAND gate is universal



AND gate

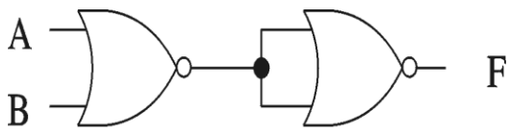


NOT gate

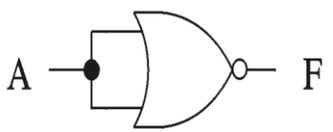


OR gate

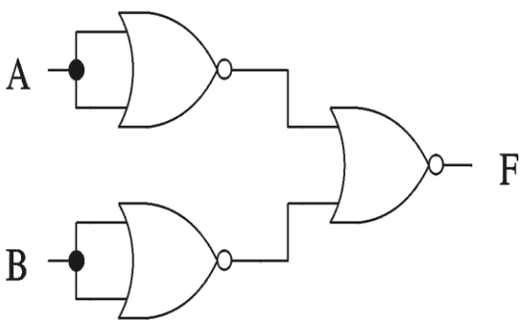
- Proving NOR gate is universal



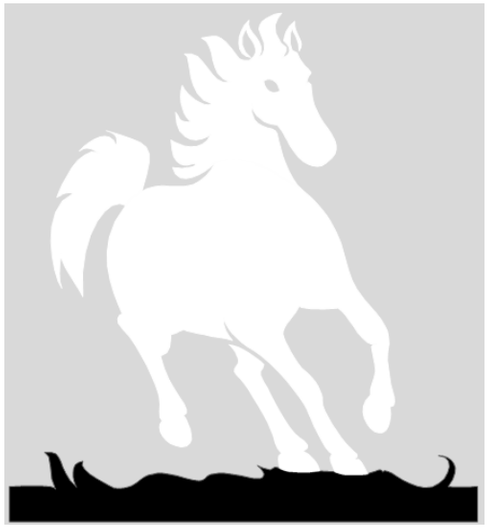
OR gate



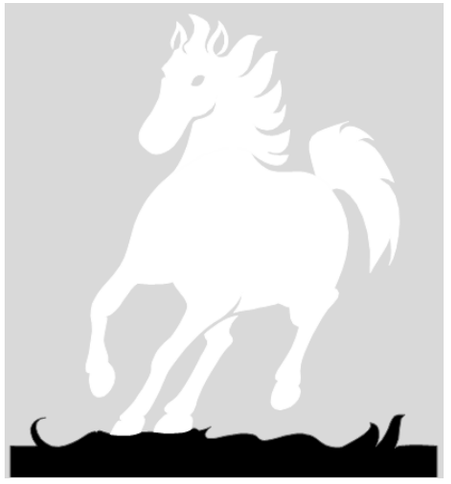
NOT gate



AND gate



Logic Functions



Logic Functions

- Logical functions can be expressed in several ways:
  - Truth table
  - Logical expressions
  - Graphical form
- Example:
  - Majority function
    - Output is one whenever majority of inputs is 1
    - We use 3-input majority function

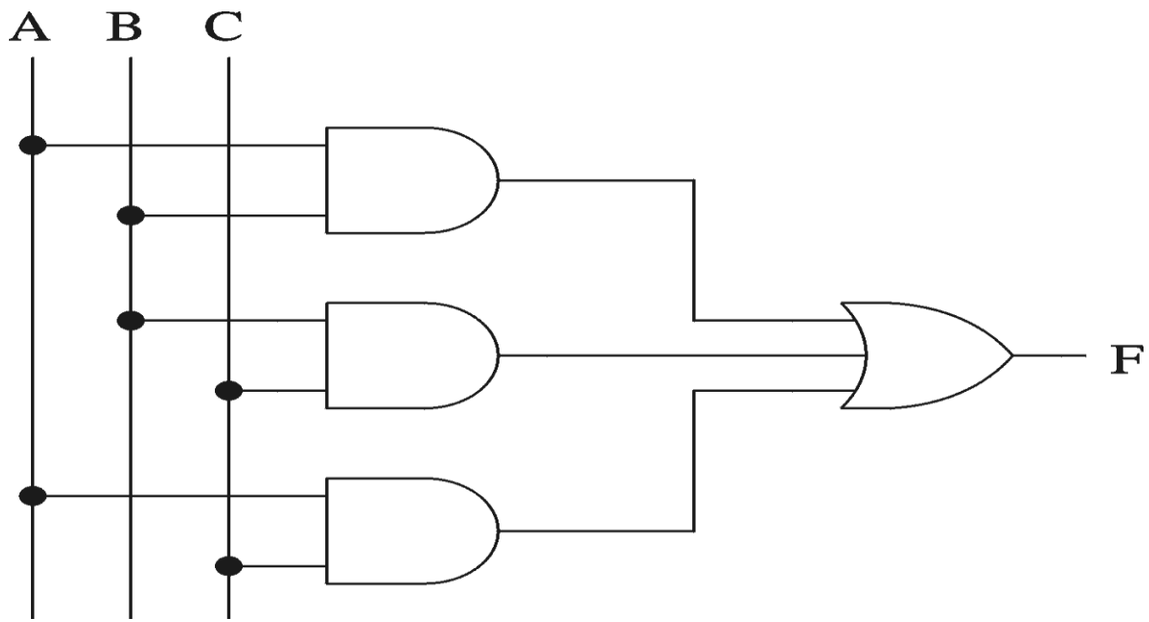
Logic Functions

3-input majority function

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

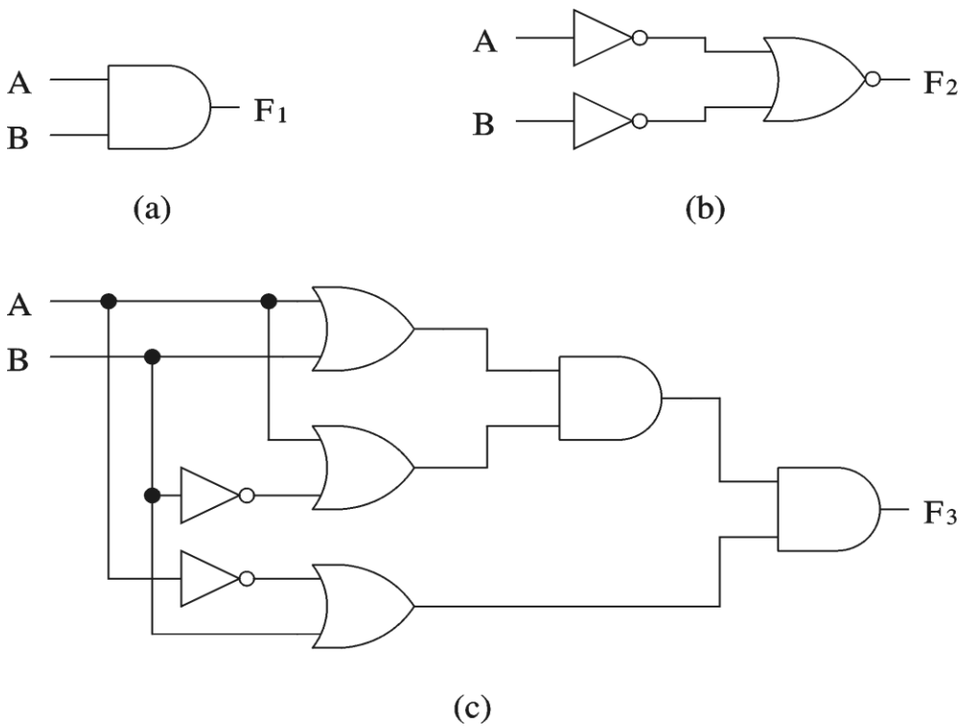
- Logical expression form

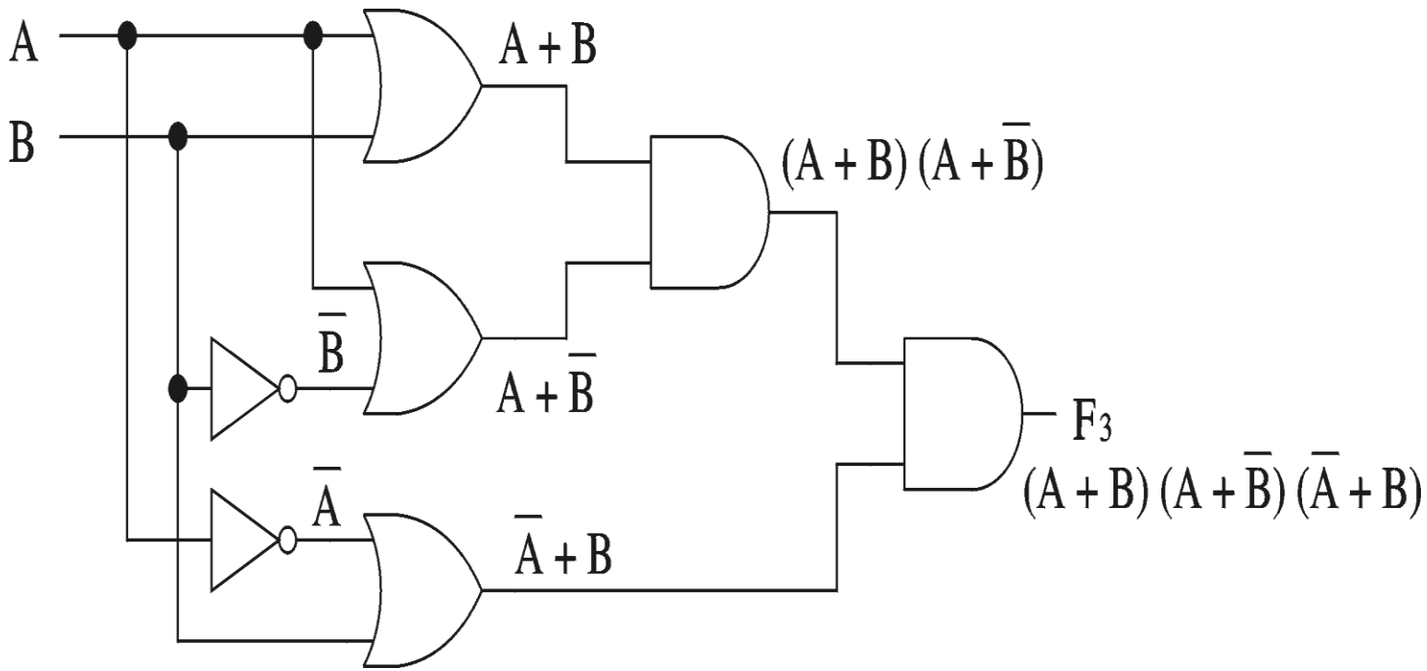
$$F = A\,B + B\,C + A\,C$$

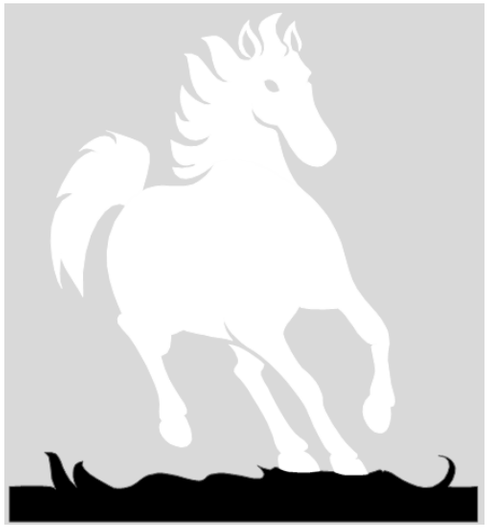


Logical Equivalence

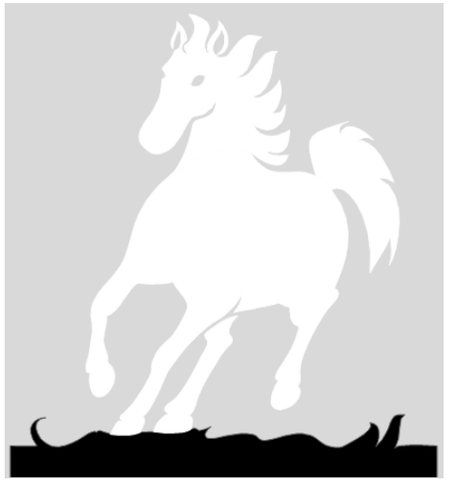
- All three circuits implement F = A B function







Boolean Algebra



Boole Cebri Aksiyomları

- Her bir değişken “0” veya “1” değerinden sadece birini alabilir.
- 1+1=1, Birbirine VEYA ile bağlı iki önermenin ikisi de doğru ise birleşik önerme de doğrudur.
  0·0=0 Birbirine VE ile bağlı iki önermenin ikisi de yanlış ise birleşik önerme de yanlıştır.
- 0+0=0 Birbirine VEYA ile bağlı iki önermenin ikisi de yanlış ise birleşik önerme de yanlıştır.
  1·1=1 Birbirine VE ile bağlı iki önermenin ikisi de doğru ise birleşik önerme de doğrudur.
- 1+0=1 Birbirine VEYA ile bağlı iki önermeden biri doğru ise birleşik önerme de doğrudur.
  0·1=0 Birbirine VE ile bağlı iki önermeden birisi yanlış ise birleşik önerme de yanlıştır.

Boole Cebri Teoremleri

1. a) $a+b=b+a$ Değişme Özelliği
   b) $a \cdot b=b \cdot a$
2. a) $a+b+c= a+b +c=a+(b+c)$ Birleşme Özelliği
   b) $a \cdot b \cdot c= a \cdot b \cdot c=a \cdot (b \cdot c)$
3. a) $a+b \cdot c= a+b \cdot (a+c)$ Dağılma Özelliği
   b) $a \cdot b+c = a \cdot b +(a \cdot c)$
4. a) $a+a=a$ Değişkende Fazlalık Özelliği
   b) $a \cdot a=a$
5. a) $a+a.b=a$ Yutma Özelliği
   b) $a \cdot (a+b)=a$
6. a) $(a) =a$ işlemde Fazlalık Özelliği
   b) $(a) =a$
7. a) $(a+b+c+\cdots) =a \cdot b \cdot c \cdots$ De Morgan Kuralı
   b) $(a \cdot b \cdot c \cdots) =a +b +c +\cdots$

Boole Cebri Teoremleri

8. a) $a+a$ =1 Sabit Özelliği
   b) $a \cdot a$ =0
9. a) 0+a=a Etkisizlik Özelliği
   b) 1$\cdot a$=a
10. a) 1+a=1 Yutan Sabit Özelliği
    b) 0$\cdot a$=0
11. a) (a+b)$\cdot b$=$a \cdot b$
    b) $a \cdot b$ +b=a+b
12. a) a+$b \cdot a$ +$c \cdot b$+c = a+b $\cdot$(a +c)
    b) $a \cdot b$+a $\cdot c$+$b \cdot c$=$a \cdot b$+a $\cdot c$
13. a) a+$b \cdot a$ +c =$a \cdot c$+a $\cdot b$
    b) $a \cdot b$+a $\cdot c$= a+c $\cdot$(a +b)
14. a) f $a,b,c,d,\cdots$ =[a+f(0,$b,c,d,\cdots$)]$\cdot$[a +f(1,$b,c,d,\cdots$)] Shannon Teoremi
    b) f $a,b,c,d,\cdots$ = $a \cdot f$ 1,$b,c,d,\cdots$ +[a $\cdot f$(0,$b,c,d,\cdots$)]

Laws and Rules of Boolean Algebra

- Laws of Boolean Algebra
  - Commutative Law
    - Commutative Law of Addition: $A + B = B + A$
    - Commutative Law of Multiplication: $AB = BA$
  - Associative Law
    - Associative Law of Addition: $A + (B + C) = (A + B) + C$
    - Associative Law of Multiplication: $A(BC) = (AB)C$
  - Distributive Law
    - $A(B + C) = AB + AC$

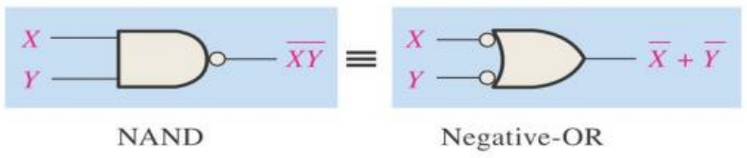
Laws and Rules of Boolean Algebra

- Laws of Boolean Algebra
  - The 12 Rules of Boolean Algebra
    - $A + 0 = A$
    - $A + 1 = 1$
    - $A \cdot 0 = 0$
    - $A \cdot 1 = A$
    - $A + A = A$
    - $A + \overline{A} = 1$
    - $A \cdot A = A$
    - $A \cdot \overline{A} = 0$
    - $\overline{\overline{A}} = A$
    - $A + AB = A$
    - $A + \overline{A}B = A + B$
    - $(A + B)(A + C) = A + BC$

DeMorgan's Theorem

Demorgan's Theorems

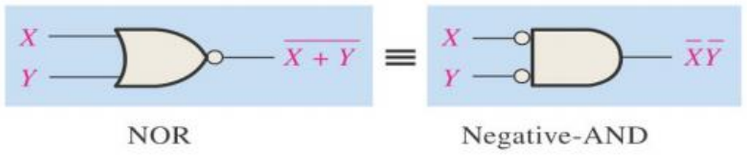
$\overline{XY} = \overline{X} + \overline{Y}$



NAND Negative-OR

Inputs		Output	
X	Y	$\overline{XY}$	$\overline{X} + \overline{Y}$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

$\overline{X + Y} = \overline{X}\,\overline{Y}$



NOR Negative-AND

Inputs		Output	
X	Y	$\overline{X + Y}$	$\overline{X}\,\overline{Y}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

$$\overline{a\,[b + c\,(d + \overline{e}\,)]}$$

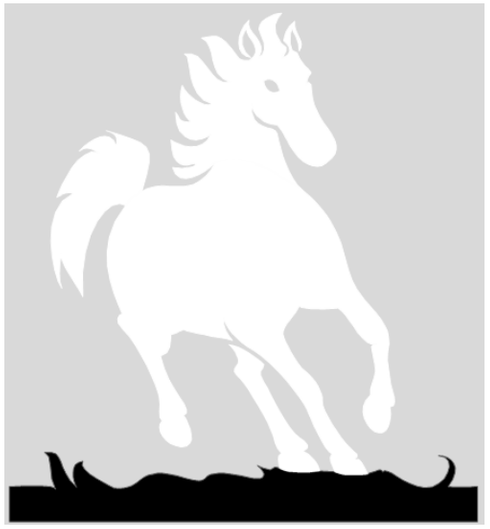
$$\overline{a} + \overline{[b + c\,(d + \overline{e}\,)]}$$

$$\overline{a} + \overline{b}\,\overline{(c\,(d + \overline{e}\,))}$$

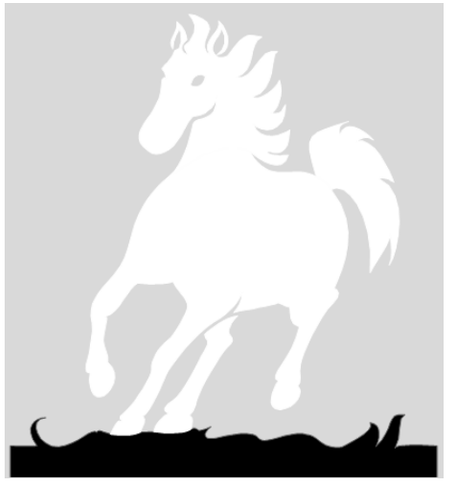
$$\overline{a} + \overline{b}\,(\overline{c} + \overline{(d + \overline{e}\,)})$$

$$\overline{a} + \overline{b}\,(\overline{c} + (\overline{d}\ \overline{e}))$$

$$\overline{a} + \overline{b}\,(\overline{c} + \overline{d}\ e)$$



Conversion of Minterm and Maxterm



Minimum ve Maksimum Terimler

- x ve y şeklinde iki terim için minimum terimler (minterm) ve maksimum terimler (maksterm) aşağıdaki tabloda verilmiştir.

x	y	Minterm	m'ye indis	Maksterm	M' ye indis
0	0	$\bar{x} \cdot \bar{y}$	m_0	$x + y$	M_0
0	1	$\bar{x} \cdot y$	m_1	$x + \bar{y}$	M_1
1	0	$x \cdot \bar{y}$	m_2	$\bar{x} + y$	M_2
1	1	$x \cdot y$	m_3	$\bar{x} + \bar{y}$	M_3

Minimum terimler kanonik biçimi

- Doğruluk tablosu kullanarak çarpımların toplamı çözümü

a	b	$F = a + b$	Minterm	m'ye indis
0	0	0	$\bar{a} \cdot \bar{b}$	m_0
0	1	1	$\bar{a} \cdot b$	m_1
1	0	1	$a \cdot \bar{b}$	m_2
1	1	1	$a \cdot b$	m_3

$$F = a + b = \bar{a} \cdot b + a \cdot \bar{b} + a \cdot b = m_1 + m_2 + m_3 = \sum (1,2,3)$$

Maksimum terimler kanonik biçimi

- Doğruluk tablosu kullanarak toplamların çarpımı çözümü

a	b	$F = a \cdot b$	Maksterm	M'ye indis
0	0	0	$a + b$	M_0
0	1	0	$a + \bar{b}$	M_1
1	0	0	$\bar{a} + b$	M_2
1	1	1	$\bar{a} + \bar{b}$	M_3

$$F = a \cdot b = (a + b) \cdot \left(a + \bar{b}\right) \cdot (\bar{a} + b) = M_0 \cdot M_1 \cdot M_2 = \prod(0,1,2)$$

- **Product term** (or minterm): ANDed product of literals – input combination for which output is true

A	B	C	minterms	
0	0	0	$\overline{A}\ \overline{B}\ \overline{C}$	m0
0	0	1	$\overline{A}\ \overline{B}\ C$	m1
0	1	0	$\overline{A}\ B\ \overline{C}$	m2
0	1	1	$\overline{A}\ B\ C$	m3
1	0	0	$A\ \overline{B}\ \overline{C}$	m4
1	0	1	$A\ \overline{B}\ C$	m5
1	1	0	$A\ B\ \overline{C}$	m6
1	1	1	$A\ B\ C$	m7

short-hand notation form in terms of 3 variables

F in canonical form:

$F(A, B, C) = \Sigma m(1,3,5,6,7)$

$= m1 + m3 + m5 + m6 + m7$

$F = \overline{A}\,\overline{B}\,C + \overline{A}\,B\,C + A\,\overline{B}\,C + A\,B\,\overline{C} + ABC$

canonical form ≠ minimal form

$F(A, B, C) = \overline{A}\,\overline{B}\,C + \overline{A}\,B\,C + A\overline{B}\,C + ABC + AB\overline{C}$

$= (\overline{A}\,\overline{B} + \overline{A}\,B + A\overline{B} + AB)C + AB\overline{C}$

$= ((\overline{A} + A)(\overline{B} + B))C + AB\overline{C}$

$= C + AB\overline{C} = AB\overline{C} + C = AB + C$

- **Sum term** (or maxterm) - ORed sum of literals – input combination for which output is false

A	B	C	maxterms	
0	0	0	$A + B + C$	M0
0	0	1	$A + B + \overline{C}$	M1
0	1	0	$A + \overline{B} + C$	M2
0	1	1	$A + \overline{B} + \overline{C}$	M3
1	0	0	$\overline{A} + B + C$	M4
1	0	1	$\overline{A} + B + \overline{C}$	M5
1	1	0	$\overline{A} + \overline{B} + C$	M6
1	1	1	$\overline{A} + \overline{B} + \overline{C}$	M7

short-hand notation for maxterms of 3 variables

F in canonical form:

$F(A, B, C) = \Pi M(0,2,4)$

$= M0 \cdot M2 \cdot M4$

$= (A + B + C)(A + \overline{B} + C)(\overline{A} + B + C)$

canonical form ≠ minimal form

$F(A, B, C) = (A + B + C)(A + \overline{B} + C)(\overline{A} + B + C)$

$= (A + B + C)(A + \overline{B} + C)$

$(A + B + C)(\overline{A} + B + C)$

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

Conversion of Minterm and Maxterm

$$F = \overline{X}\overline{Y}\overline{Z} + \overline{X}Y\overline{Z} + X\overline{Y}Z + XYZ = m_0 + m_2 + m_5 + m_7 = \sum m(0, 2, 5, 7)$$

$$\overline{F} = \overline{X}\overline{Y}Z + \overline{X}YZ + X\overline{Y}\overline{Z} + XY\overline{Z} = m_1 + m_3 + m_4 + m_6 = \sum m(1, 3, 4, 6)$$

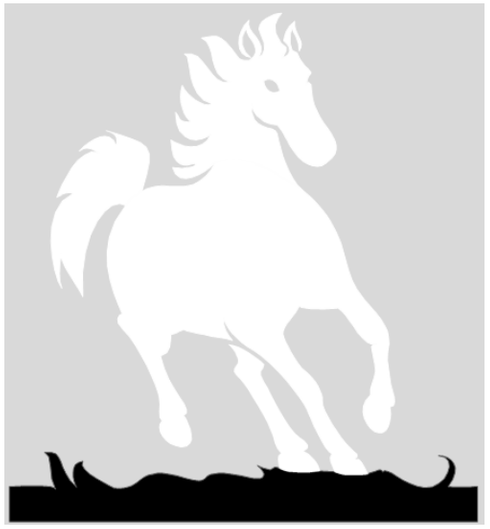
Conversion of Minterm and Maxterm

$$\overline{F} = m_1 + m_3 + m_4 + m_6$$

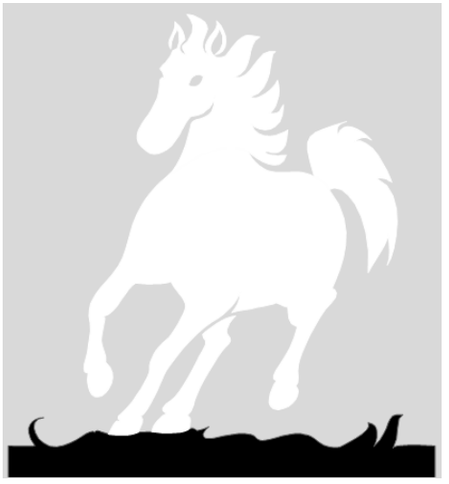
$$\Rightarrow F = \overline{m_1 + m_3 + m_4 + m_6} = \overline{m_1} \cdot \overline{m_3} \cdot \overline{m_4} \cdot \overline{m_6}$$

$$\Rightarrow F = M_1 \cdot M_3 \cdot M_4 \cdot M_6 = (X + Y + \overline{Z})(X + \overline{Y} + Z)(\overline{X} + Y + Z)(\overline{X} + \overline{Y} + Z)$$

$$= \prod M(1, 3, 4, 6)$$

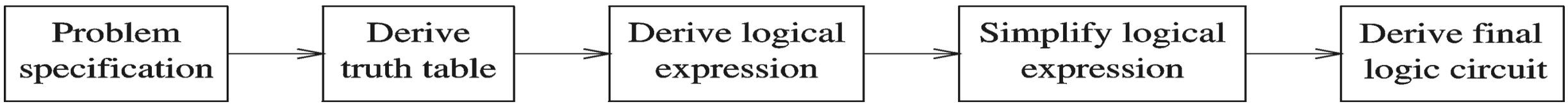


Simplify the function



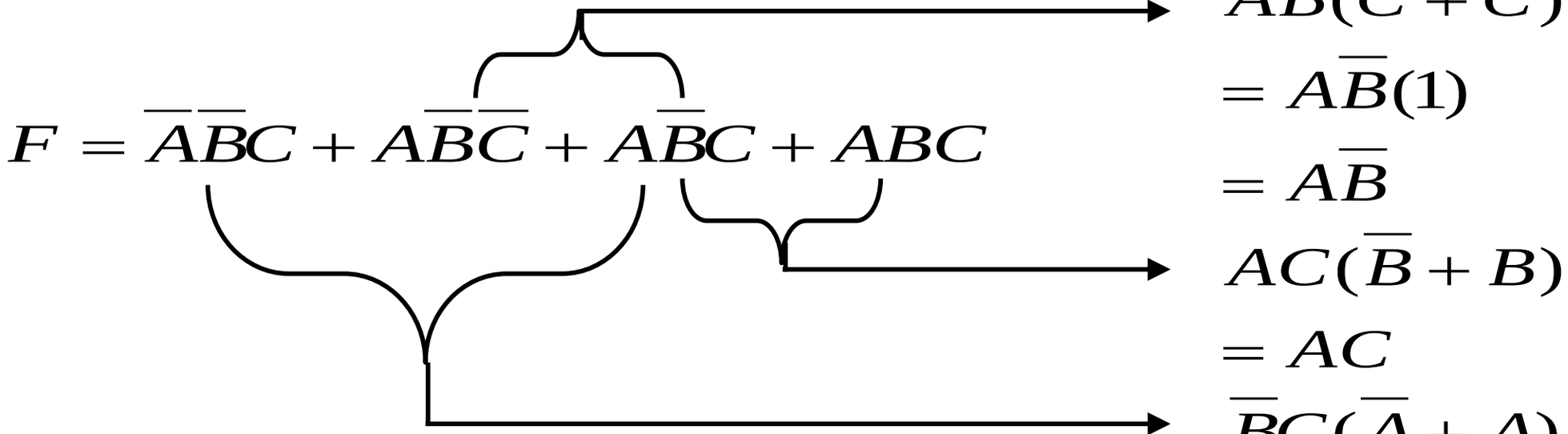
Logic Circuit Design Process

- A simple logic design process involves
  - Problem specification
  - Truth table derivation
  - Derivation of logical expression
  - Simplification of logical expression
  - Implementation



Standard Forms

- Sum of Products (SOP)



$$F = \overline{B}C(\overline{A} + A) + A\overline{B}(\overline{C} + C) + AC(\overline{B} + B)$$

$$F = \overline{B}C + A\overline{B} + AC$$

Boolean Algebra

- We can use Boolean identities to simplify the function:

  as follows: $F(X,Y,Z) = (X + Y)(X + \overline{Y})\,\overline{(X\overline{Z})}$

$(X + Y)(X + \overline{Y})\,\overline{(X\overline{Z})}$	Idempotent Law (Rewriting)
$(X + Y)(X + \overline{Y})(\overline{X} + Z)$	DeMorgan's Law
$(XX + X\overline{Y} + XY + Y\overline{Y})(\overline{X} + Z)$	Distributive Law
$((X + Y\overline{Y}) + X(Y + \overline{Y}))(\overline{X} + Z)$	Commutative & Distributive Laws
$((X + 0) + X(1))(\overline{X} + Z)$	Inverse Law
$X(\overline{X} + Z)$	Idempotent Law
$X\overline{X} + XZ$	Distributive Law
$0 + XZ$	Inverse Law
XZ	Idempotent Law

Logic simplification

- Example:
- Z = A'BC + AB'C' + AB'C + ABC' + ABC
  = A'BC + AB'(C’ + C) + AB(C' + C) distributive
  = A'BC + AB’ + AB complementary
  = A'BC + A(B' + B) distributive
  = A'BC + A complementary
  = BC + A absorption #2 Duality

  (X •Y')+Y=X+Y with X=BC and Y=A

- Simplify $A + AB + A\overline{B}C$
  - DeMorgan's theorems.

$$A + AB + A\overline{B}C$$
$$A + A\overline{B}C$$
$$A$$

- Simplify AB + A(B + C) + B(B + C)

$$AB + AB + AC + BB + BC$$
$$AB + AC + B + BC$$
$$AB + B + AC$$
$$B + AC$$

$$Y = A.B.C + A.\overline{B}.C + \overline{B}.\overline{C} + \overline{A}.\overline{B}$$

Using Boolean algebra:

$Y = A.B.C + A.\overline{B}.C + \overline{B}.\overline{C} + \overline{A}.\overline{B}$

$Y = A.B.C + A.\overline{B}.C + A.\overline{B}.\overline{C} + \overline{A}.\overline{B}.\overline{C} + \overline{A}.\overline{B}.C + \overline{A}.\overline{B}.\overline{C}$ **(Expanding all terms by multiplying by 1. i.e. (A + $\overline{A}$))**

$Y = A.B.C + \overline{B}.(A.C + A.\overline{C} + \overline{A}.\overline{C} + \overline{A}.C + \overline{A}.\overline{C})$ **(Take out the common factor)**

$Y = A.B.C + \overline{B}.\left(A.\left(C + \overline{C}\right) + \overline{A}.\left(\overline{C} + C + \overline{C}\right)\right)$ **(Group terms and take out the common factors)**

$Y = A.B.C + \overline{B}\left(A + \overline{A}\right)$ **(Simplify)**

$\underline{Y = A.B.C + \overline{B}}$

Structural Operations

Restructuring Problem: Given initial network, find best network.

Example: $f_1 = abcd+abce+ab'cd'+ab'c'd'+a'c+cdf+abc'd'e'+ab'c'df'$

$f_2 = bdg+b'dfg+b'd'g+bd'eg$

minimizing,

$f_1 = bcd+bce+b'd'+a'c+cdf+abc'd'e'+ab'c'df'$

$f_2 = bdg+dfg+b'd'g+d'eg$

factoring,

$f_1 = c(b(d+e)+b'(d'+f)+a')+ac'(bd'e'+b'df')$

$f_2 = g(d(b+f)+d'(b'+e))$

decompose,

$f_1 = c(x+a')+ac'x'$

$f_2 = gx$

$x = d(b+f)+d'(b'+e)$

Two problems:

- find good common subfunctions
- effect the division

Structural Operations

Basic Operations:

1. Decomposition (single function)

   $f = abc+abd+a'c'd'+b'c'd'$

   $\Downarrow$

   $f = xy+x'y' \qquad x = ab \quad y = c+d$

2. Extraction (multiple functions)

   $f = (az+bz')cd+e \quad g = (az+bz')e' \quad h = cde$

   $\Downarrow$

   $f = xy+e \quad g = xe' \quad h = ye \quad x = az+bz' \quad y = cd$

3. Factoring (series-parallel decomposition)

   $f = ac+ad+bc+bd+e$

   $\Downarrow$

   $f = (a+b)(c+d)+e$

4. Substitution

   $g = a+b \quad f = a+bc$

   $\Downarrow$

   $f = g(a+b)$

5. Collapsing (also called elimination)

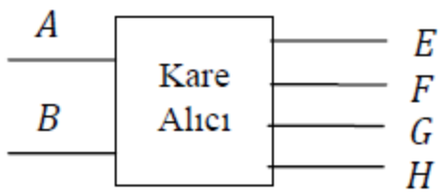
   $f = ga+g'b \qquad\qquad g = c+d$

   $\Downarrow$

   $f = ac+ad+bc'd' \quad g = c+d$

Örnek

- Girişine giren 2 bitlik sayıların karesini alan lojik devreyi doğruluk tablosu çıkararak hesaplayınız.



A	B	E	F	G	H
0	0	0	0	0	0
0	1	0	0	0	1
1	0	0	1	0	0
1	1	1	0	0	1

$E = A \cdot B$ $\quad$ $G = Lojik\ 0$ $\quad$ $F = A \cdot \bar{B}$ $\quad$ $H = \bar{A}B + AB = B(A + \bar{A}) = B$

$A + \overline{A}B$

↓ Applying the previous rule to expand **A** term
A + AB = A

$A + AB + \overline{A}B$

↓ Factoring **B** out of 2nd and 3rd terms

$A + B(A + \overline{A})$

↓ Applying identity $\mathbf{A + \overline{A} = 1}$

$A + B(1)$

↓ Applying identity **1A = A**

$A + B$

$A + B(A + C) + AC$

↓ Distributing terms

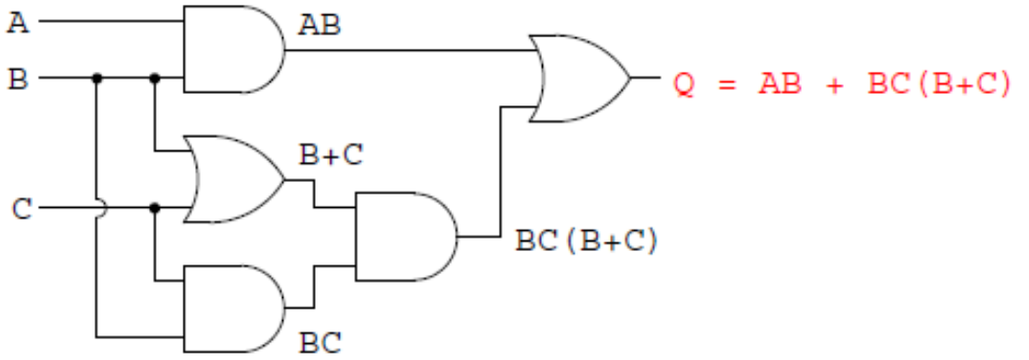
$A + AB + BC + AC$

↓ Applying rule **A + AB = A** to 1st and 2nd terms

$A + BC + AC$

↓ Applying rule **A + AB = A** to 1st and 3rd terms

$A + BC$



$AB + BC(B + C)$

↓ Distributing terms

$AB + BBC + BCC$

↓ Applying identity **AA = A** to 2nd and 3rd terms

$AB + BC + BC$

↓ Applying identity **A + A = A** to 2nd and 3rd terms

$AB + BC$

↓ Factoring **B** out of terms

$B(A + C)$

$\overline{\overline{A + BC} + \overline{AB}}$

↓ Breaking longest bar

$(\overline{\overline{A + BC}})\ (\overline{\overline{AB}})$

↓ Applying identity $\overline{\overline{A}} = A$ wherever double bars of equal length are found

$(A + BC)\ (A\overline{B})$

↓ Distributive property

$AA\overline{B} + BCA\overline{B}$

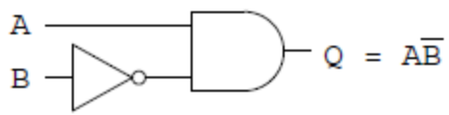
↓ Applying identity $AA = A$ to left term; applying identity $A\overline{A} = 0$ to B and $\overline{B}$ in right term

$A\overline{B} + 0$

↓ Applying identity $A + 0 = A$

$A\overline{B}$

The equivalent gate circuit for this much-simplified expression is as follows:



$\overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$

↓ Factoring **BC** out of 1st and 4th terms

$BC(\overline{A} + A) + A\overline{B}C + AB\overline{C}$

↓ Applying identity $A + \overline{A} = 1$

$BC(1) + A\overline{B}C + AB\overline{C}$

↓ Applying identity $1A = A$

$BC + A\overline{B}C + AB\overline{C}$

↓ Factoring **B** out of 1st and 3rd terms

$B(C + A\overline{C}) + A\overline{B}C$

↓ Applying rule $A + \overline{A}B = A + B$ to the $C + A\overline{C}$ term

$B(C + A) + A\overline{B}C$

↓ Distributing terms

$BC + AB + A\overline{B}C$

↓ Factoring **A** out of 2nd and 3rd terms

$BC + A(B + \overline{B}C)$

↓ Applying rule $A + \overline{A}B = A + B$ to the $B + \overline{B}C$ term

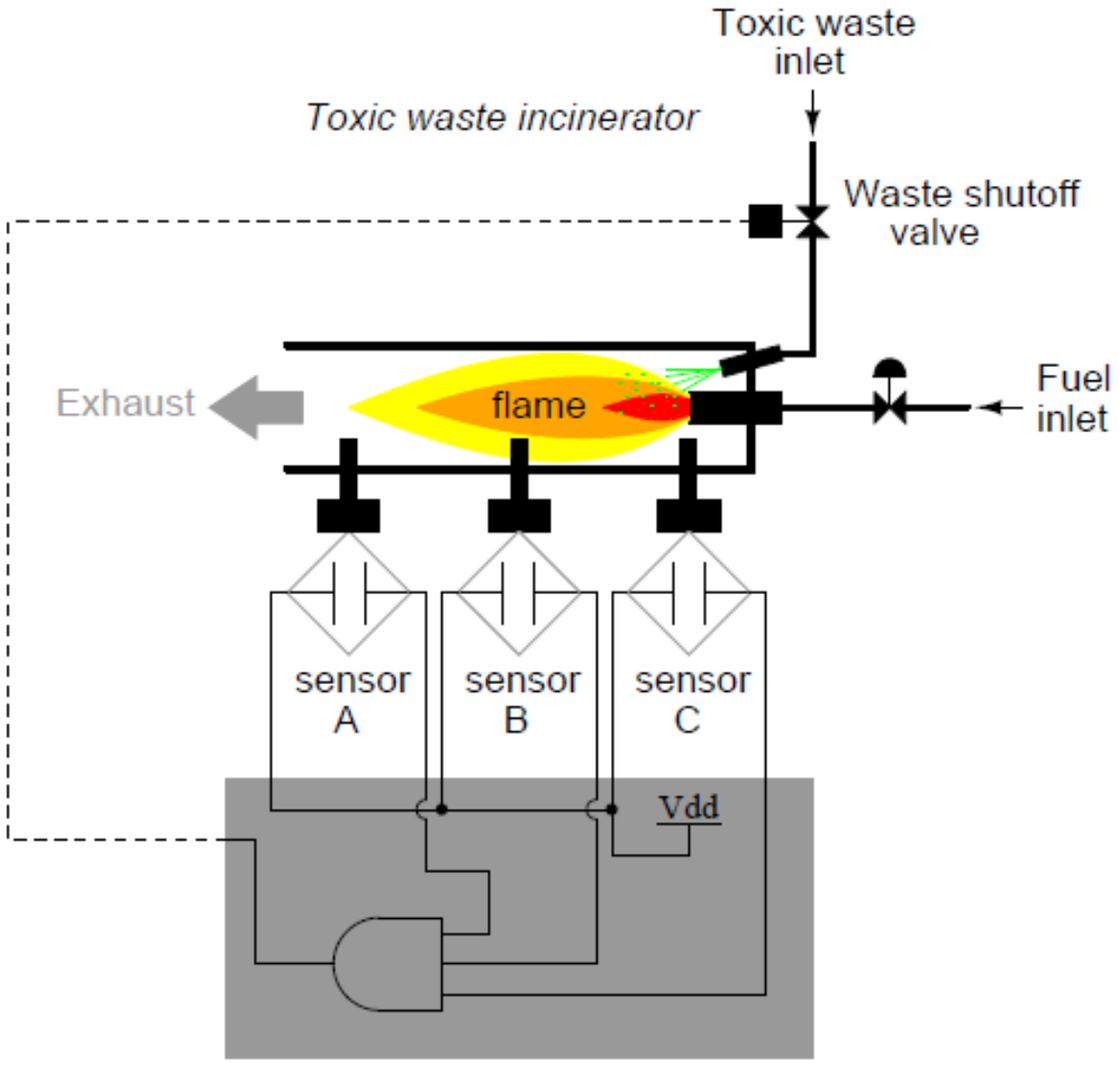
$BC + A(B + C)$

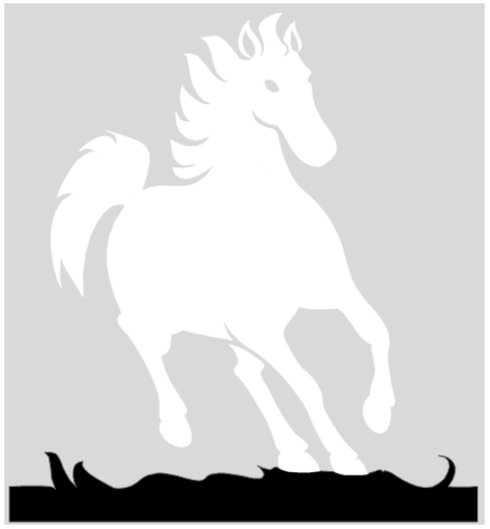
↓ Distributing terms

$BC + AB + AC$

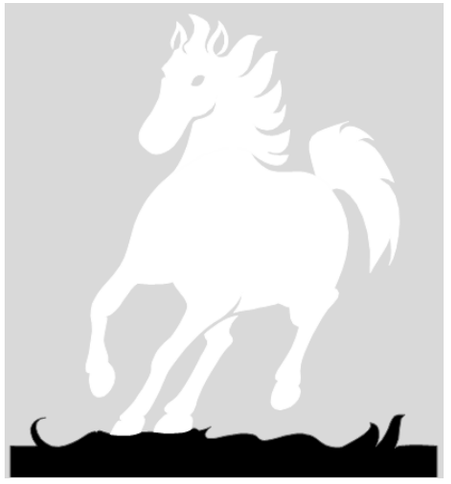
or Simplified result

$AB + BC + AC$


Toxic waste
inlet
Toxic waste incinerator
Waste shutoff
valve
Exhaust
flame
Fuel
inlet
sensor
A
sensor
B
sensor
C
Vdd



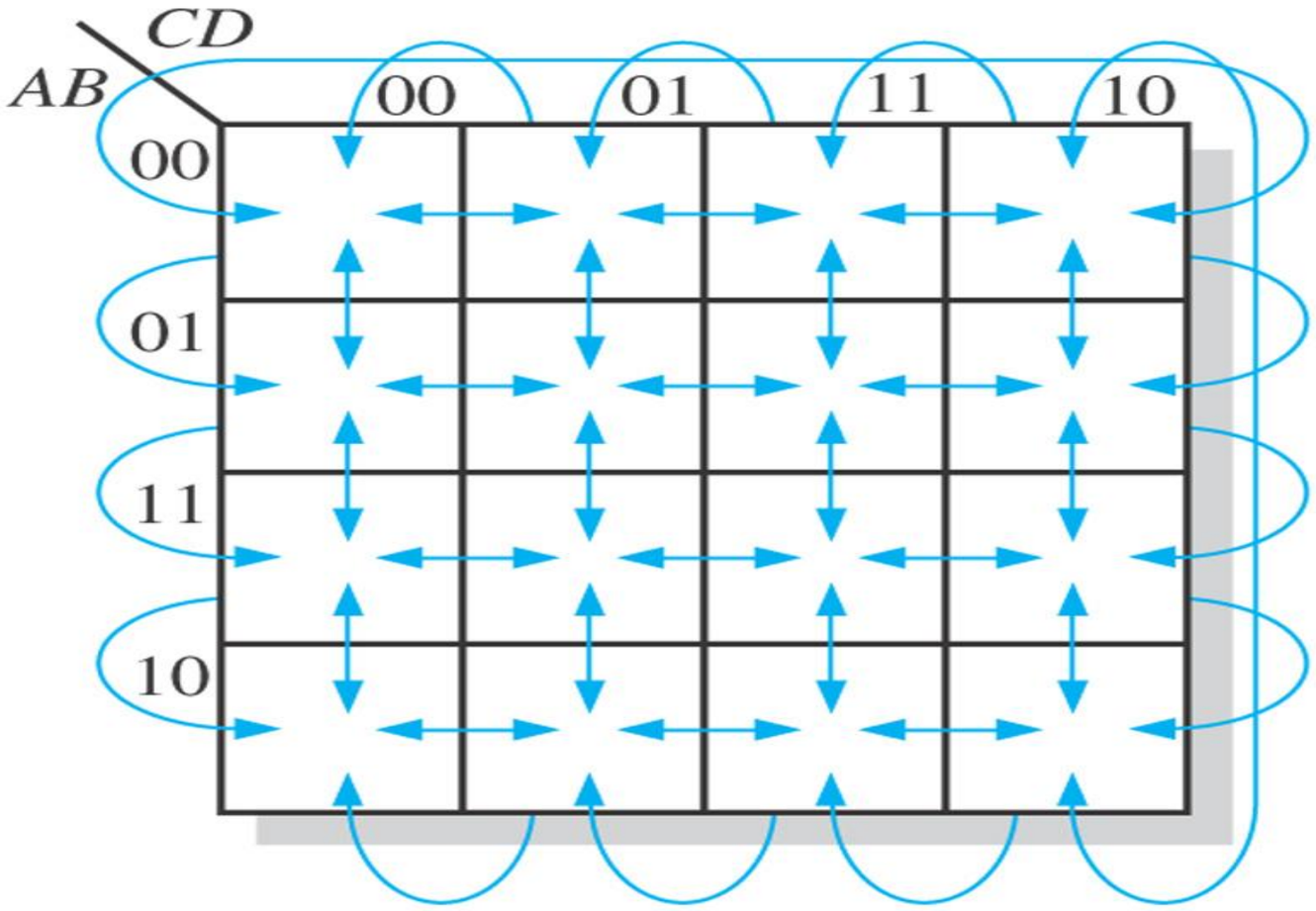
The Karnaugh Map

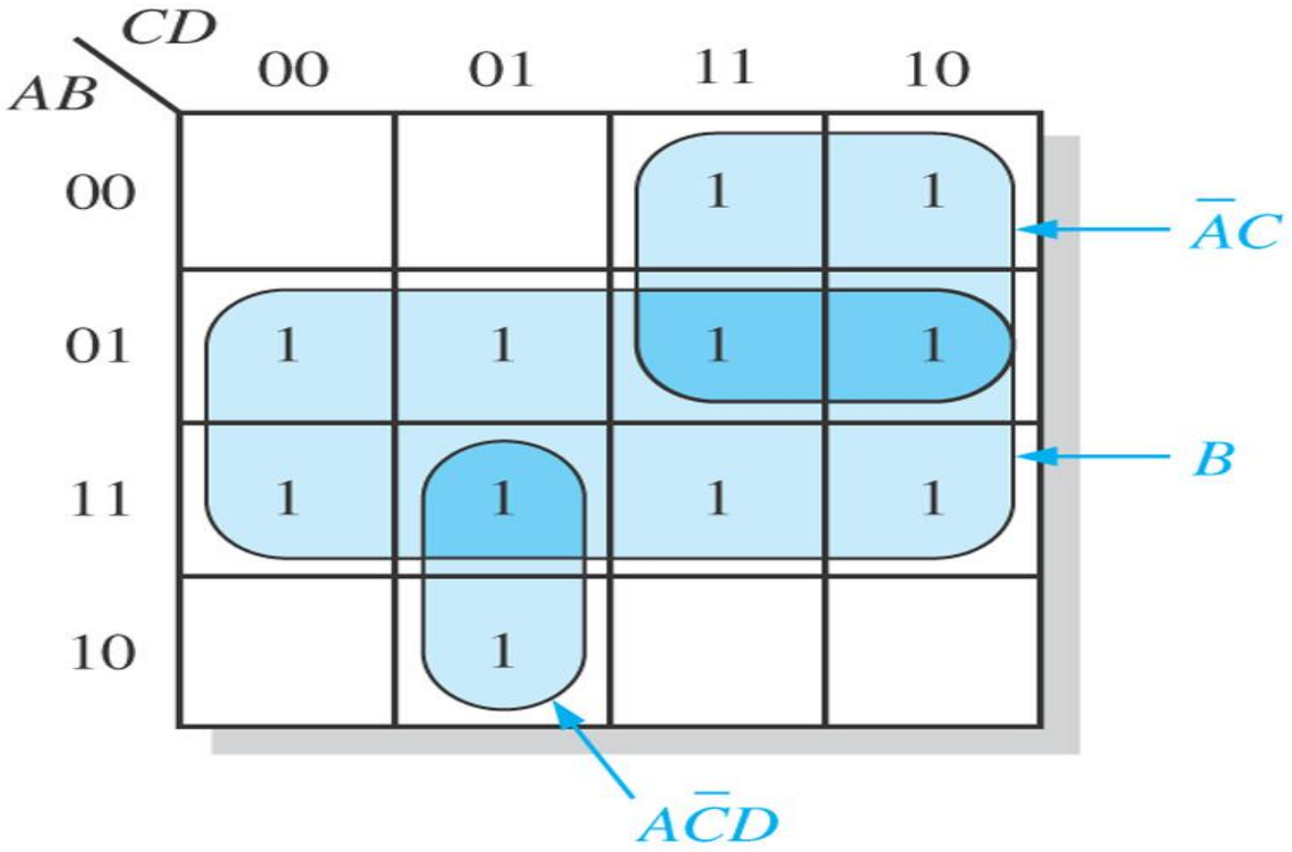


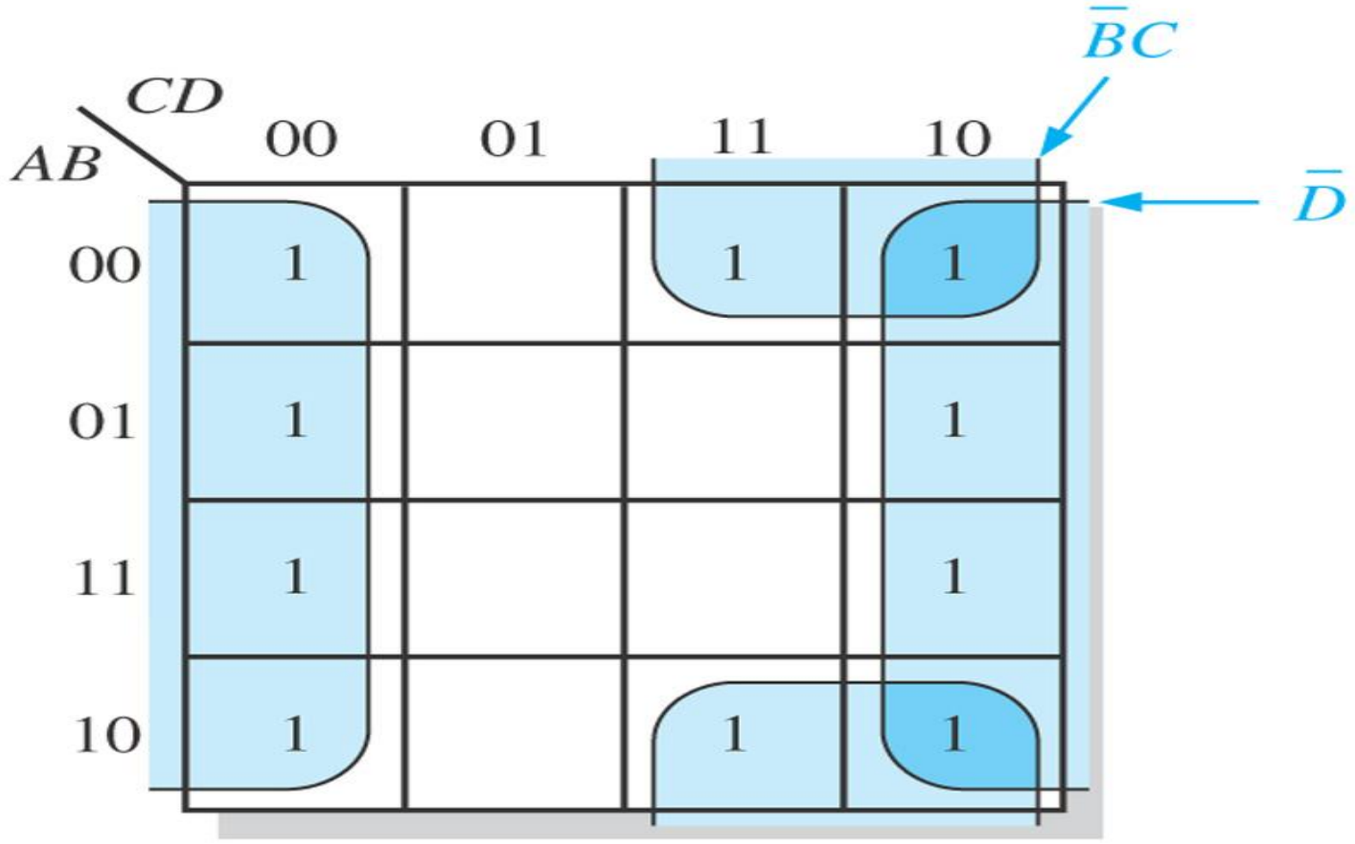
Lojik fonksiyonların sadeleştirilmesi

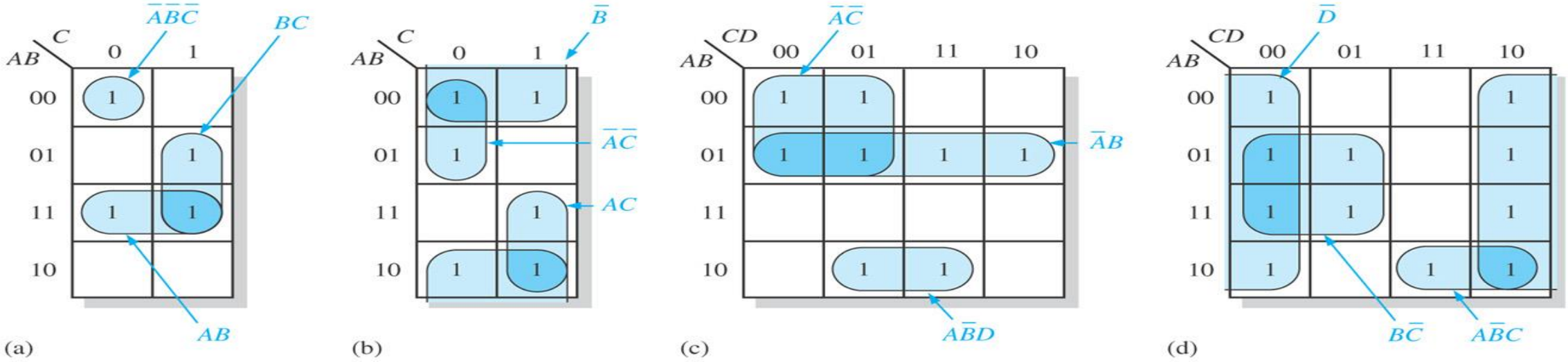
- Lojik fonksiyonların sadeleştirilmesinde en çok kullanılan iki yöntem şunlardır:
- 1. Karnaugh Diyagramı Yöntemi
- 2. Quine-McCluskey Tablo Yöntemi

Adjacent cells on a Karnaugh map are those that differ by only one variable. Arrows point between adjacent cells.




CD
AB
00
01
11
10
00
01
11
10
1
1
1
1
1
1
1
1
1
1
1
1
1
$\bar{A}C$
B
$A\bar{C}D$


$\bar{B}C$
CD
AB
00
01
11
10
$\bar{D}$
00
01
11
10
1
1
1
1
1
1
1
1
1
1
1
1


$\bar{A}\bar{B}\bar{C}$
BC
AB
C
AB
0
1
00
01
11
10
(a)
$\bar{B}$
$\bar{A}\bar{C}$
AC
(b)
$\bar{A}\bar{C}$
$\bar{A}B$
$A\bar{B}D$
CD
00
01
11
10
(c)
$\bar{D}$
$B\bar{C}$
$A\bar{B}C$
(d)

Problems

Simplify the following Boolean functions, using three-variable maps:

(a) $F(x, y, z) = \Sigma(0, 2, 4, 5)$

(b) $F(x, y, z) = \Sigma(0, 2, 4, 5, 6)$

(c) $F(x, y, z) = \Sigma(0, 1, 2, 3, 5)$

(d) $F(x, y, z) = \Sigma(1, 2, 3, 7)$

Simplify the following Boolean functions, using three-variable maps:

(a)* $F(x, y, z) = \Sigma(0, 1, 5, 7)$

(b)* $F(x, y, z) = \Sigma(1, 2, 3, 6, 7)$

(c) $F(x, y, z) = \Sigma(2, 3, 4, 5)$

(d) $F(x, y, z) = \Sigma(1, 2, 3, 5, 6, 7)$

(e) $F(x, y, z) = \Sigma(0, 2, 4, 6)$

(f) $F(x, y, z) = \Sigma(3, 4, 5, 6, 7)$

Simplify the following Boolean expressions, using three-variable maps:

(a)* $xy + x'y'z' + x'yz'$

(b)* $x'y' + yz + x'yz'$

(c)* $F(x, y, z) = x'y + yz' + y'z'$

(d) $F(x, y, z) = x'yz + xy'z' + xy'z$

Problems

Simplify the following Boolean functions, using *Karnaugh* maps:

(a)* $F(x, y, z) = \Sigma(2, 3, 6, 7)$

(b)* $F(A, B, C, D) = \Sigma(4, 6, 7, 15)$

(c)* $F(A, B, C, D) = \Sigma(3, 7, 11, 13, 14, 15)$

(d)* $F(w, x, y, z) = \Sigma(2, 3, 12, 13, 14, 15)$

(e) $F(w, x, y, z) = \Sigma(11, 12, 13, 14, 15)$

(f) $F(w, x, y, z) = \Sigma(8, 10, 12, 13, 14)$

Simplify the following Boolean functions, using four-variable maps:

(a)* $F(w, x, y, z) = \Sigma(1, 4, 5, 6, 12, 14, 15)$

(b) $F(A, B, C, D) = \Sigma(2, 3, 6, 7, 12, 13, 14)$

(c) $F(w, x, y, z) = \Sigma(1, 3, 4, 5, 6, 7, 9, 11, 13, 15)$

(d)* $F(A, B, C, D) = \Sigma(0, 2, 4, 5, 6, 7, 8, 10, 13, 15)$

Simplify the following Boolean expressions, using four-variable maps:

(a)* $A'B'C'D' + AC'D' + B'CD' + A'BCD + BC'D$

(b)* $x'z + w'xy' + w(x'y + xy')$

(c) $A'B'C'D + AB'D + A'BC' + ABCD + AB'C$

(d) $A'B'C'D' + BC'D + A'C'D + A'BCD + ACD'$

Simplify the following Boolean expressions, using four-variable maps:

(a)* $w'z + xz + x'y + wx'z$

(b) $AD' + B'C'D + BCD' + BC'D$

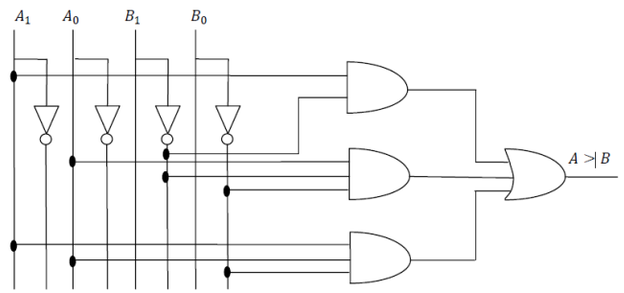
(c)* $AB'C + B'C'D' + BCD + ACD' + A'B'C + A'BC'D$

(d) $wxy + xz + wx'z + w'x$

Örnek

- A=$A1$, $A0$ ve B=$B1$, $B0$ sayıları karşılaştırılacaktır. A>B, A=B ve A<B çıkışlarını veren devreyi doğruluk tablosunu çıkartarak Karnaugh diyagramı ile gerçekleştiriniz.

A_1	A_0	B_1	B_0	$A > B$	$A = B$	$A < B$
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0



B_1B_0 / A_1A_0	00	01	11	10
00				
01	1			
11	1	1		1
10	1	1		

$$A > B$$

$$F_1 = A_1\overline{B_1} + A_0\overline{B_1}\,\overline{B_0} + A_1A_0\overline{B_0}$$

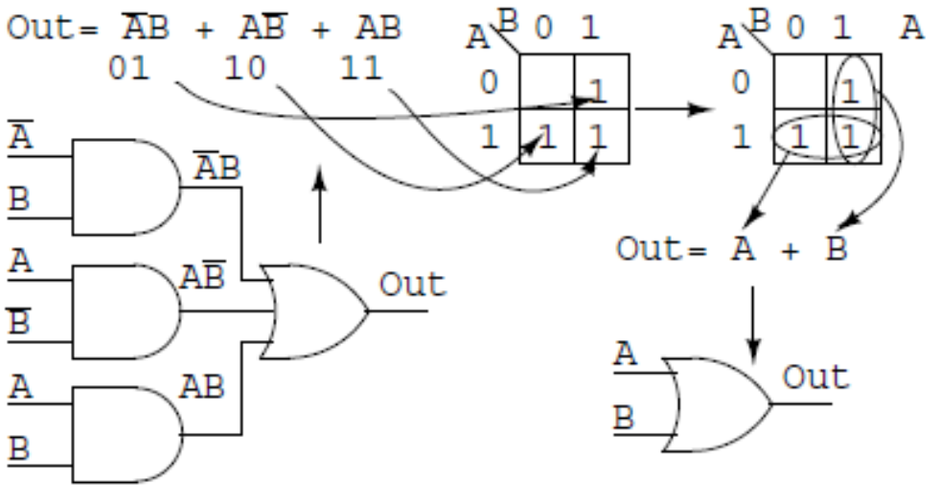
B_1B_0 / A_1A_0	00	01	11	10
00	1			
01		1		
11			1	
10				1

$$A = B$$

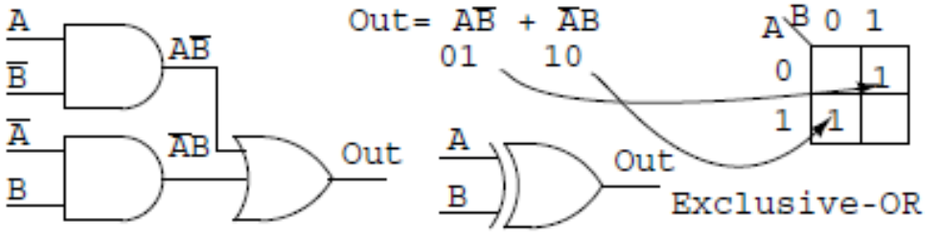
$$F_2 = \overline{A_1}\,\overline{A_0}\,\overline{B_1}\,\overline{B_0} + \overline{A_1}\,A_0\,\overline{B_1}\,B_0 + A_1\,A_0\,B_1\,B_0 + A_1\,\overline{A_0}\,B_1\,\overline{B_0}$$

B_1B_0 / A_1A_0	00	01	11	10
00		1	1	1
01			1	1
11				
10			1	

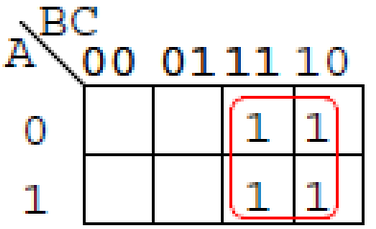
$$A < B \Rightarrow F_3 = \overline{A_1}\,B_1 + \overline{A_0}\,B_1\,B_0 + \overline{A_1}\,\overline{A_0}\,B_0$$



Simplify the logic diagram below.



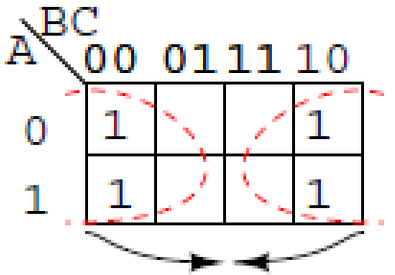
Out= $\overline{A}BC + \overline{A}B\overline{C} + ABC + AB\overline{C}$

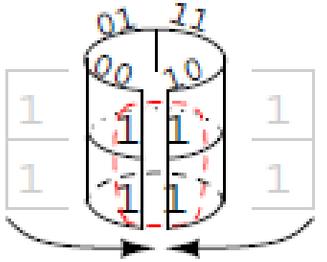


Out= B .

Mapping the four p-terms yields a single group of four, which is **B**

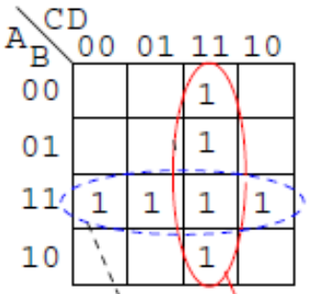
Out= $\overline{A}\overline{B}\overline{C} + A\overline{B}\overline{C} + \overline{A}B\overline{C} + AB\overline{C}$



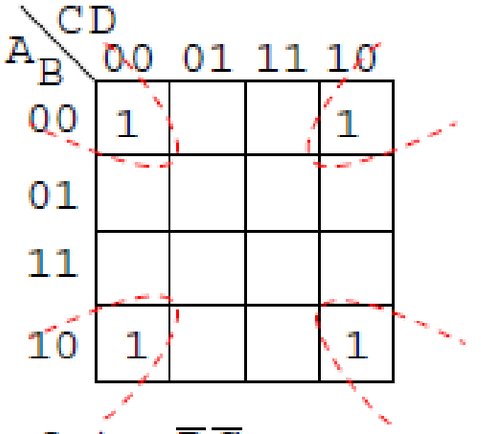


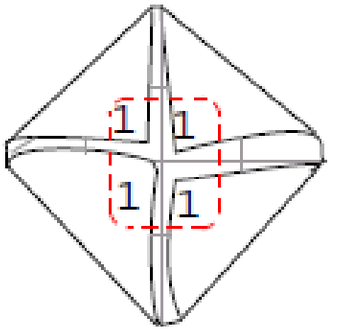
Out= $\overline{C}$.

Out= $\overline{A}\overline{B}CD+\overline{A}BCD+ABCD+A\overline{B}CD+AB\overline{C}\overline{D}+AB\overline{C}D+ABC\overline{D}$

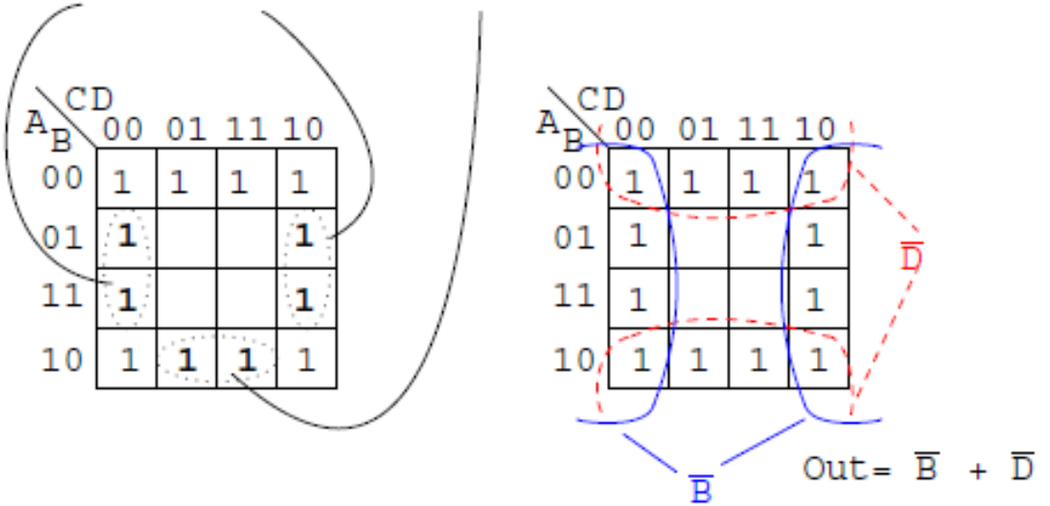


Out= $\overline{A}\overline{B}\overline{C}\overline{D}+\overline{A}\overline{B}C\overline{D}+A\overline{B}\overline{C}\overline{D}+A\overline{B}C\overline{D}$

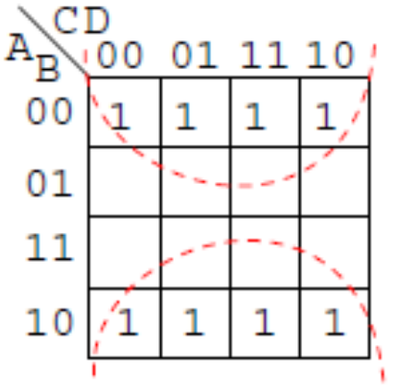




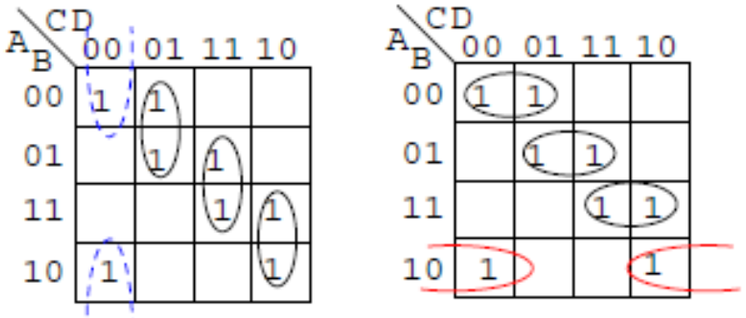
Out= $\overline{A}\overline{B}\overline{C}\overline{D}+\overline{A}\overline{B}\overline{C}D+\overline{A}\overline{B}CD+\overline{A}\overline{B}C\overline{D}$
$+B\overline{C}\overline{D}+BC\overline{D}+A\overline{B}\overline{C}\overline{D}+A\overline{B}D+A\overline{B}C\overline{D}$



Out= $\overline{A}\overline{B}\overline{C}\overline{D}+\overline{A}\overline{B}\overline{C}D+\overline{A}\overline{B}CD+\overline{A}\overline{B}C\overline{D}$
$+A\overline{B}\overline{C}\overline{D}+A\overline{B}\overline{C}D+A\overline{B}CD+A\overline{B}C\overline{D}$



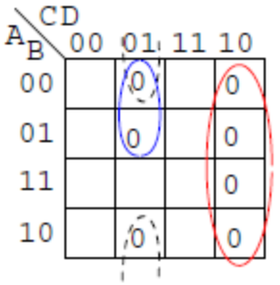
Out= $\overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}B\overline{C}D + \overline{A}BCD$
$+ ABCD + ABC\overline{D} + A\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}CD$

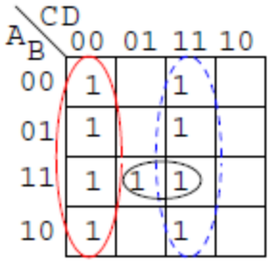


Out= $\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{C}D + BCD + AC\overline{D}$

Out= $\overline{A}B\overline{C} + \overline{A}BD + ABC + AB\overline{D}$

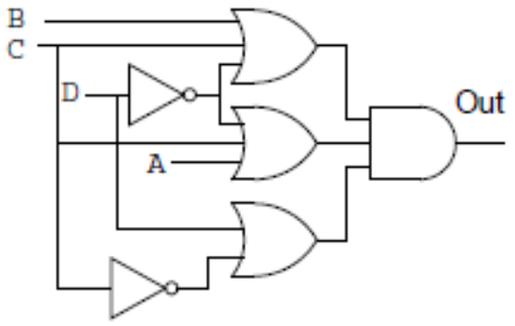
Out= $(A+B+C+\overline{D})(A+B+\overline{C}+D)(A+\overline{B}+C+\overline{D})(A+\overline{B}+\overline{C}+D)$
$(\overline{A}+\overline{B}+\overline{C}+D)(\overline{A}+B+C+\overline{D})(\overline{A}+B+\overline{C}+D)$



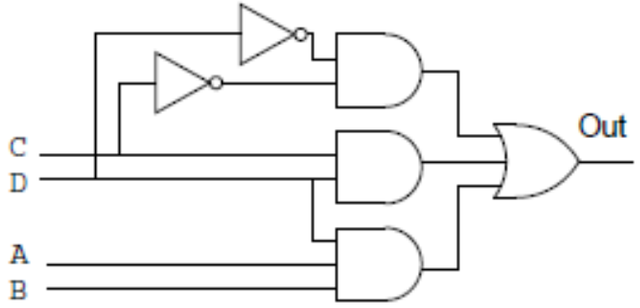


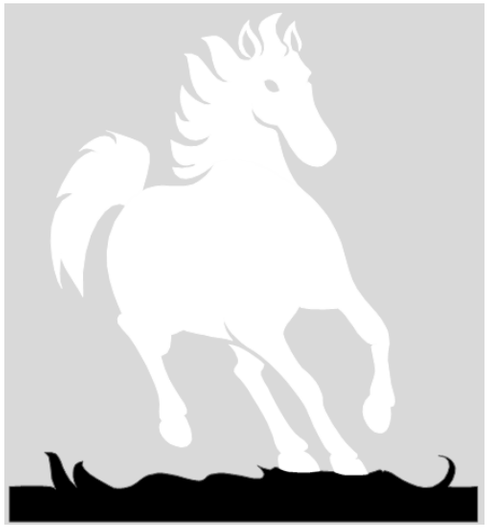
Out= $\overline{C}\,\overline{D} + CD + ABD$

Out= $(B+C+\overline{D})(A+C+\overline{D})(\overline{C}+D)$

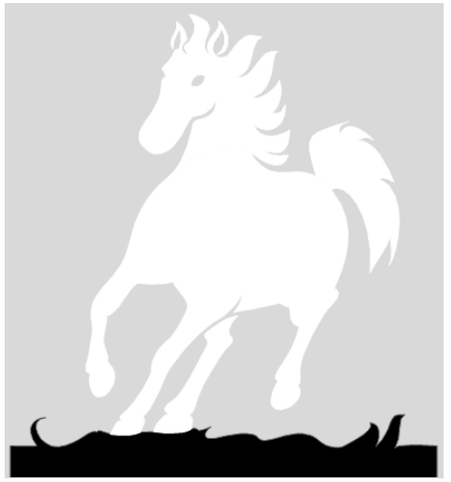


Out= $\overline{C}\,\overline{D} + CD + ABD$





Combinational Circuits



Tümleşik Kombinasyonel Devreler

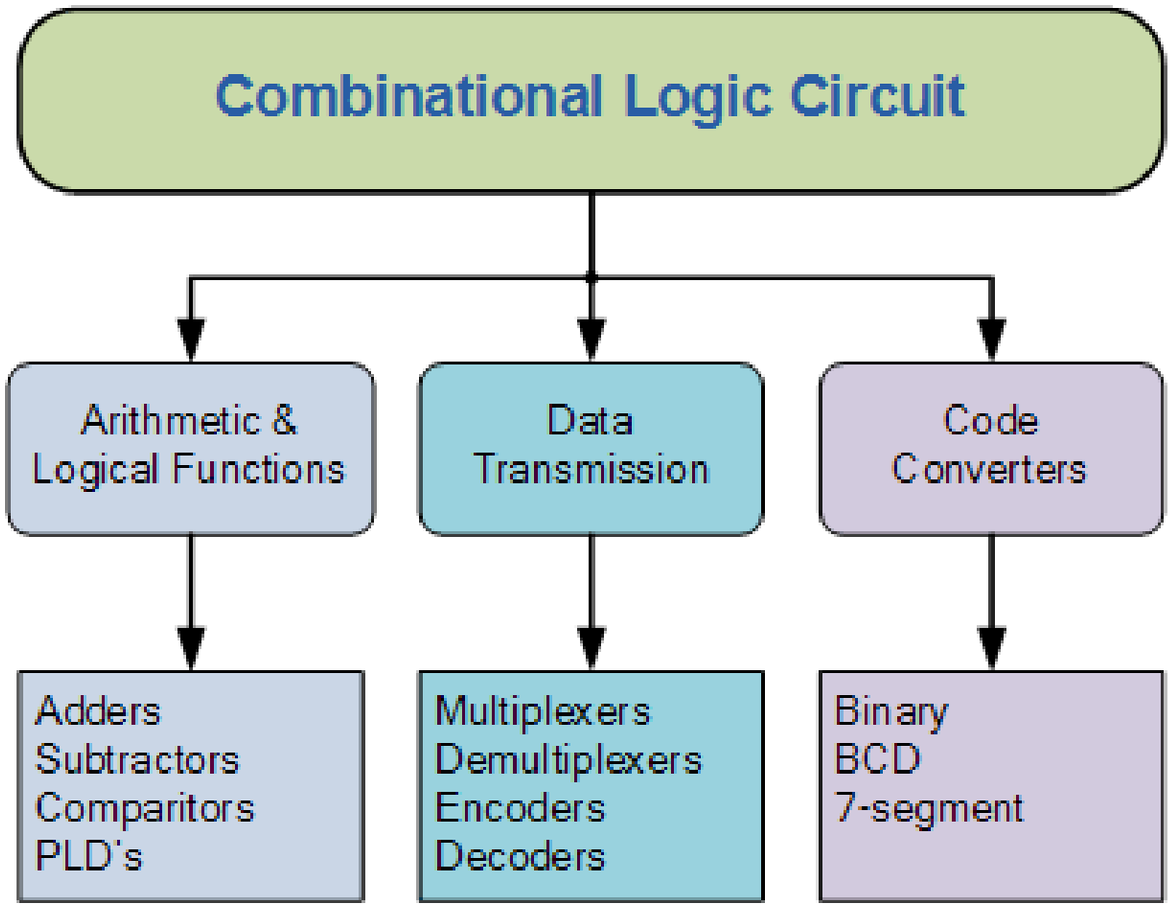
Belirli bir işii yerine getiren kapılarla tasarlanmış devrelerdir.

- Yarı ve tam toplayıcılar,
- seçiciler,
- kodlayıcılar,
- PAL (Programlanabilir Dizi Elemanı),
- PLA (Programlanabilir Lojik Dizi Elemanı)
- Çoğullayıcılar gibi elemanlar sayılabilir.

Introduction to Combinational Circuits

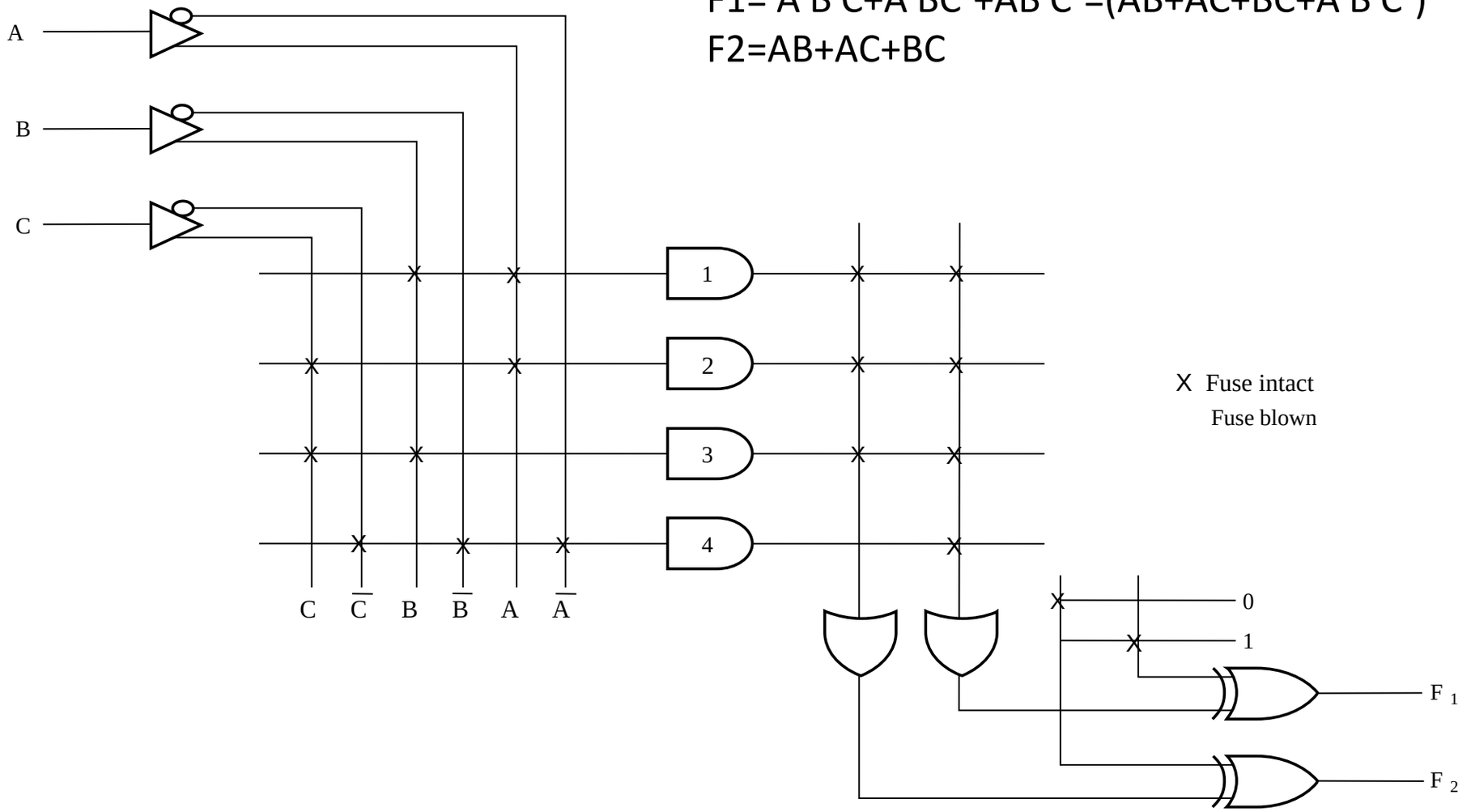
- Combinational circuits
    - Output depends only on the current inputs
- Combinational circuits provide a higher level of abstraction
  - Help in reducing design complexity
  - Reduce chip count
- We look at some useful combinational circuits

Classification of Combinational Logic



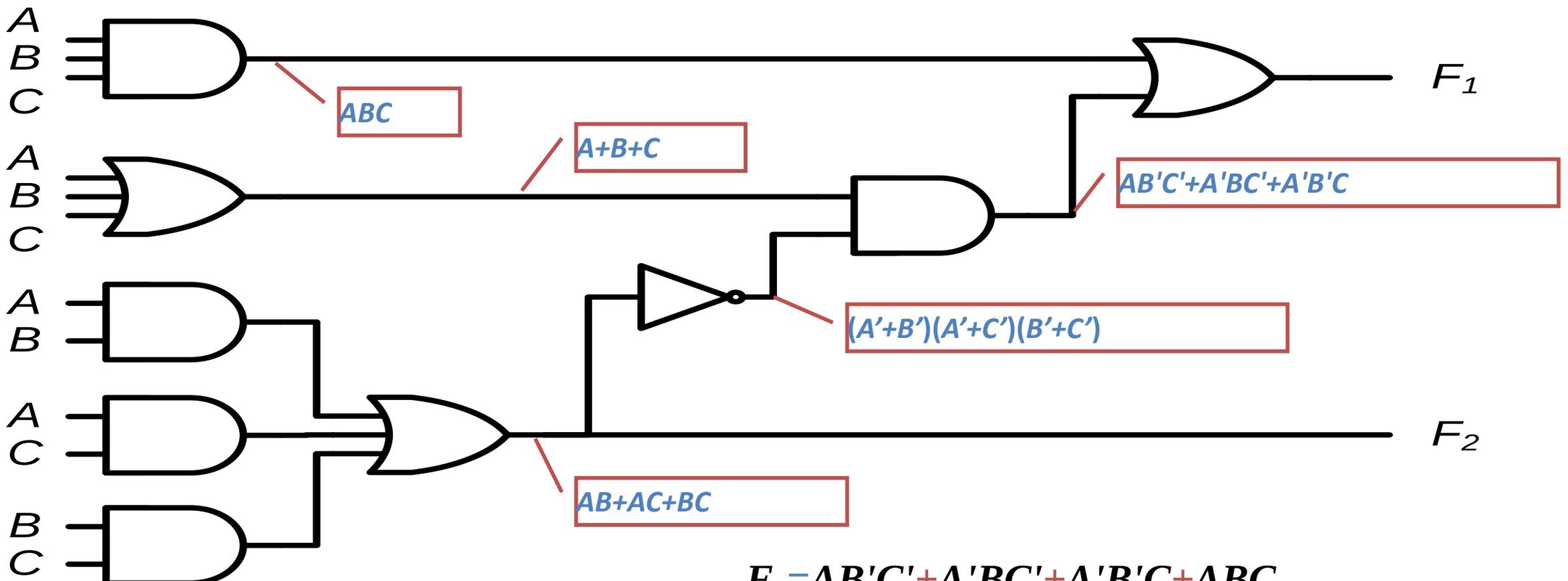
Programmable Logic Array Example

F1= A’B’C+A’BC’+AB’C’=(AB+AC+BC+A’B’C’)’
F2=AB+AC+BC



Analysis Procedure

- Boolean Expression Approach

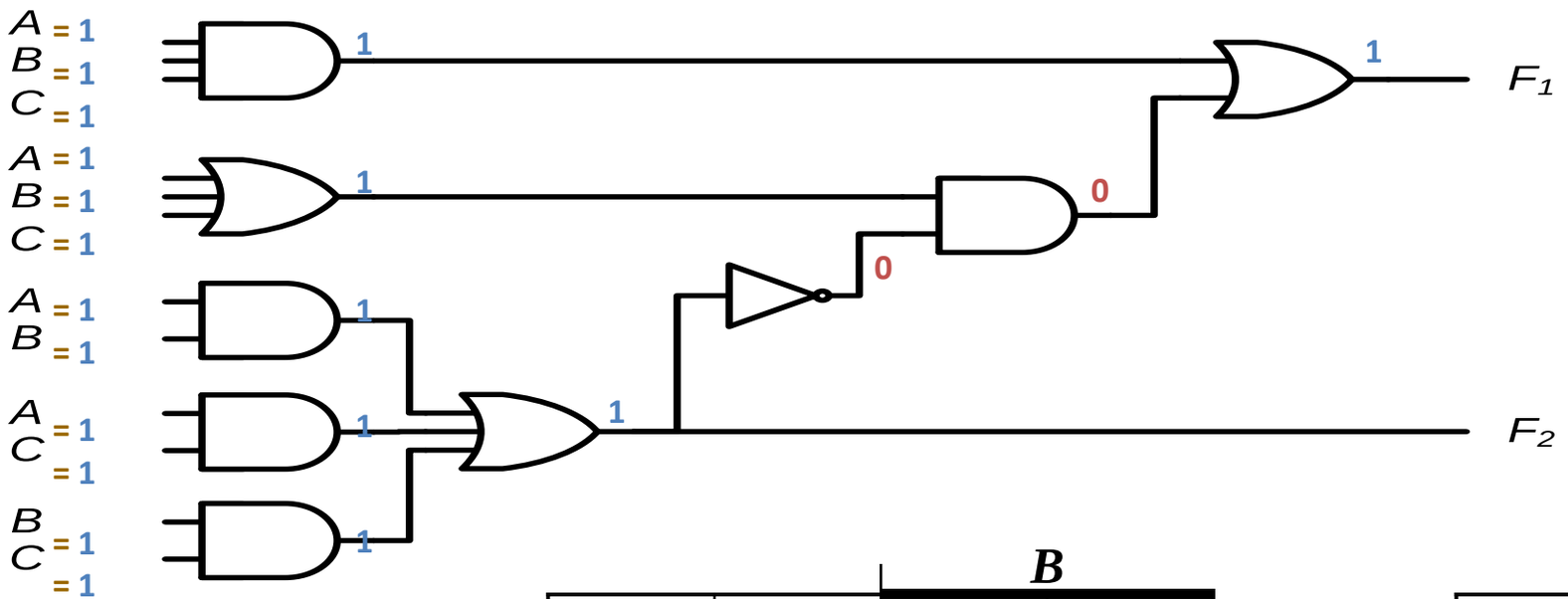


$$F_1 = AB'C' + A'BC' + A'B'C + ABC$$

$$F_2 = AB + AC + BC$$

Analysis Procedure

- Truth Table Approach



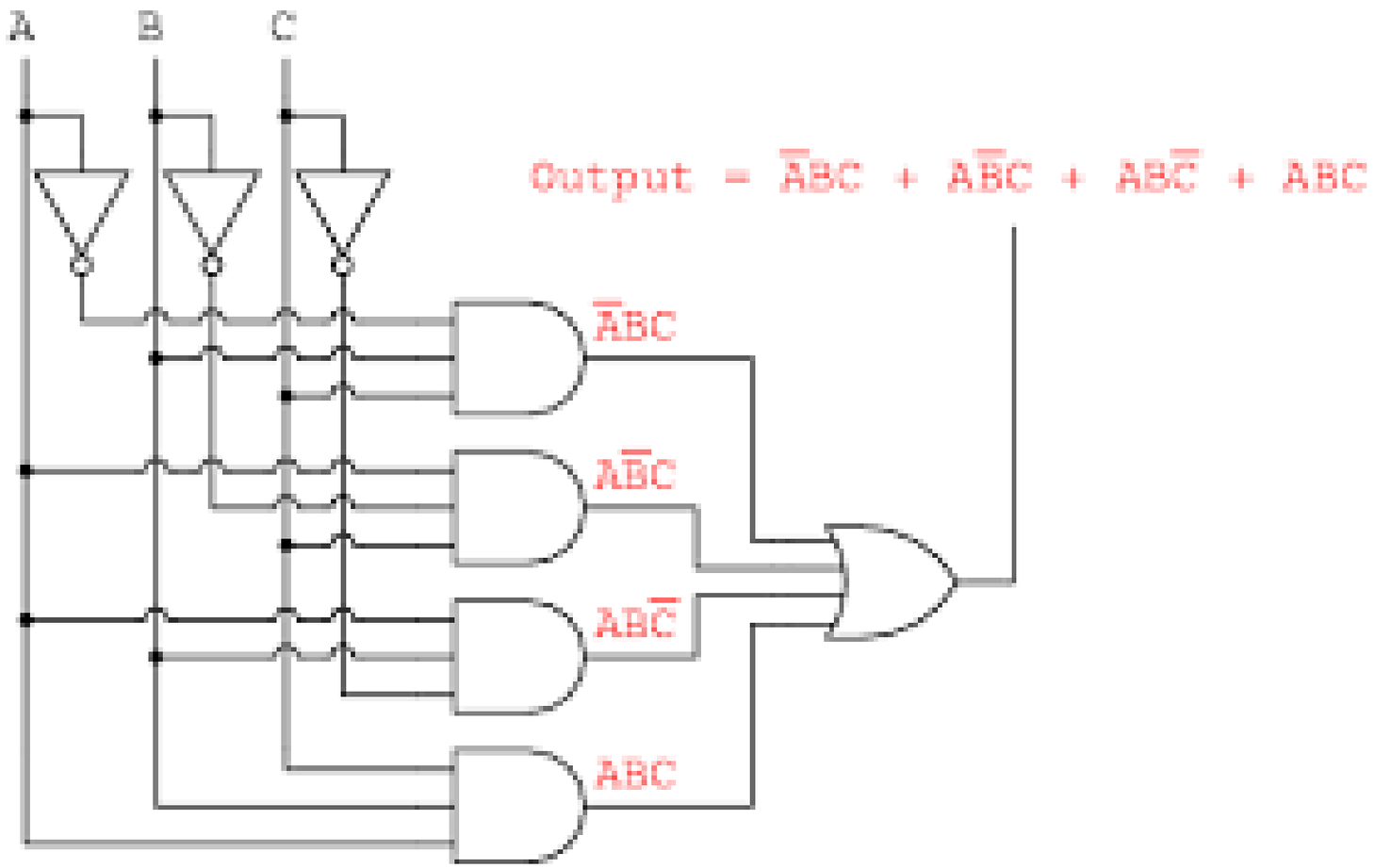
A B C	F_1	F_2
0 0 0	0	0
0 0 1	1	0
0 1 0	1	0
0 1 1	0	1
1 0 0	1	0
1 0 1	0	1
1 1 0	0	1
1 1 1	1 1	

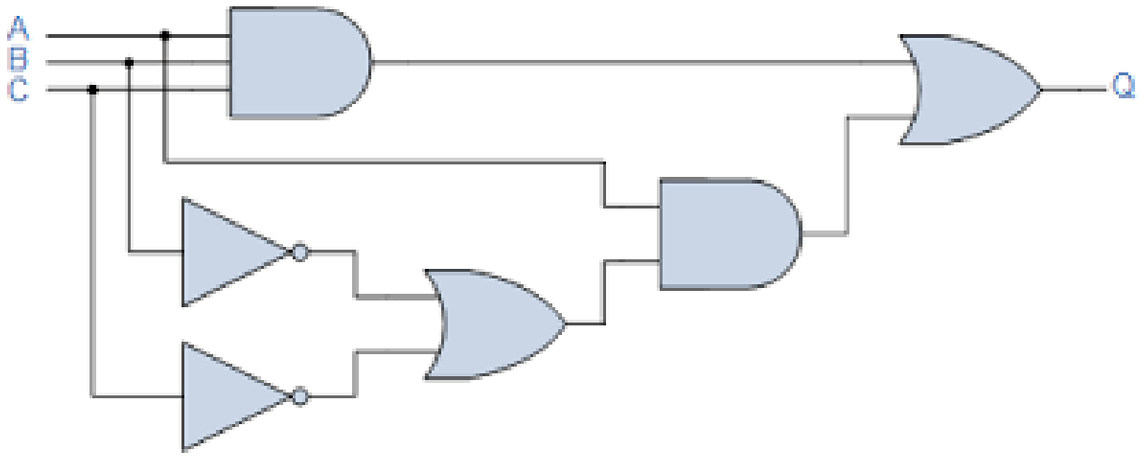
A \ BC	00	01	11	10
0	0	1	0	1
1	1	0	1	0

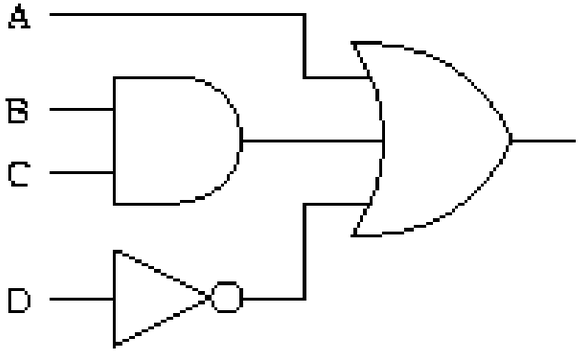
A \ BC	00	01	11	10
0	0	0	1	0
1	0	1	1	1

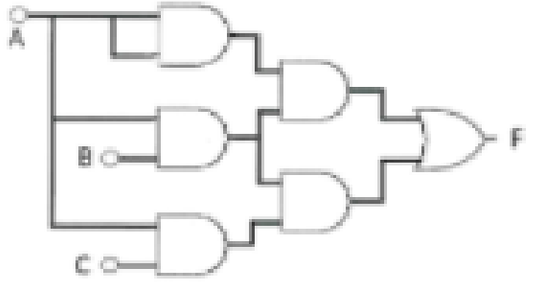
$F_1 = AB'C' + A'BC' + A'B'C + ABC$

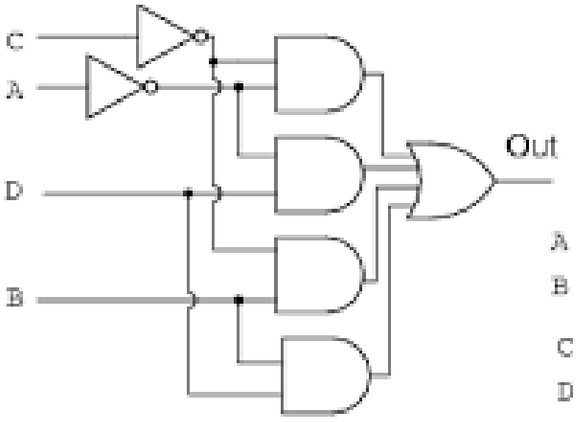
$F_2 = AB + AC + BC$


A
B
C
Output = $\overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$
$\overline{A}BC$
$A\overline{B}C$
$AB\overline{C}$
ABC

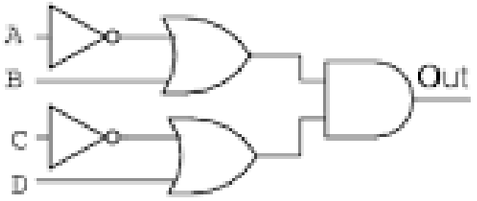

A
B
C
Q


A
B
C
D



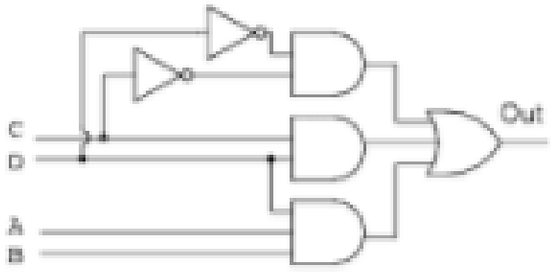


Out= $\overline{A}\overline{C} + \overline{A}D + B\overline{C} + BD$

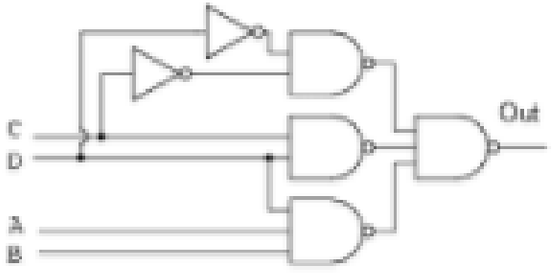


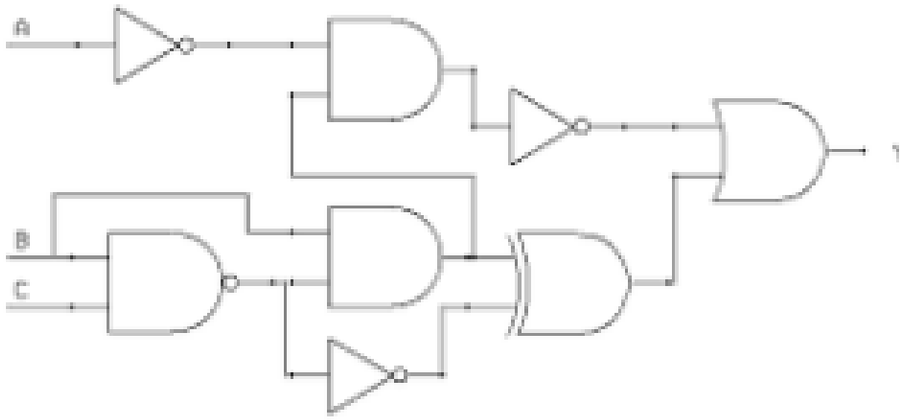
Out= $(\overline{A}+B)(\overline{C}+D)$

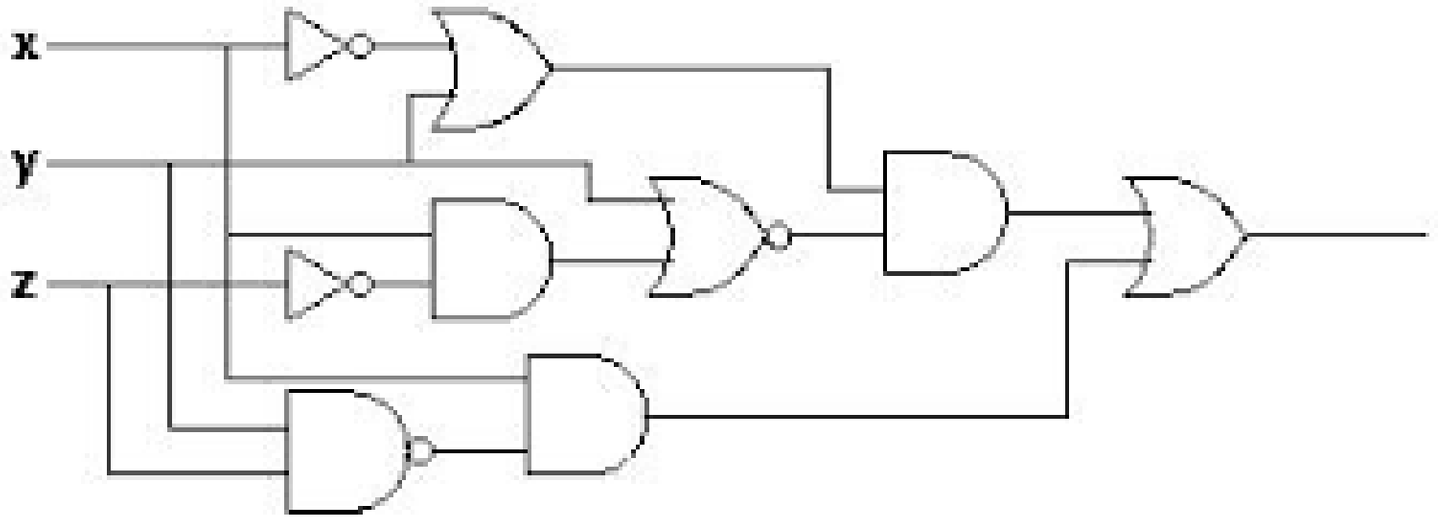
Out= $\overline{C}\overline{D} + CD + ABD$



Out= $\overline{C}\overline{D} + CD + ABD$



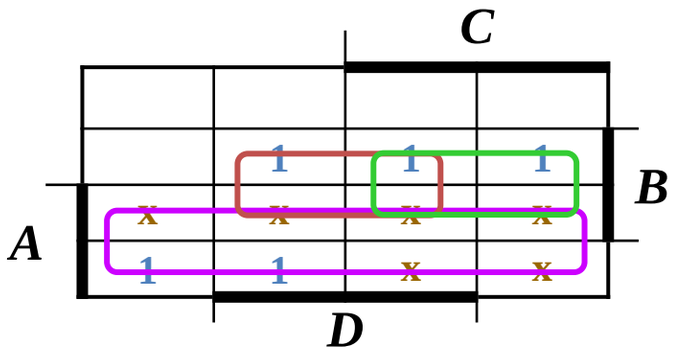



x
y
z

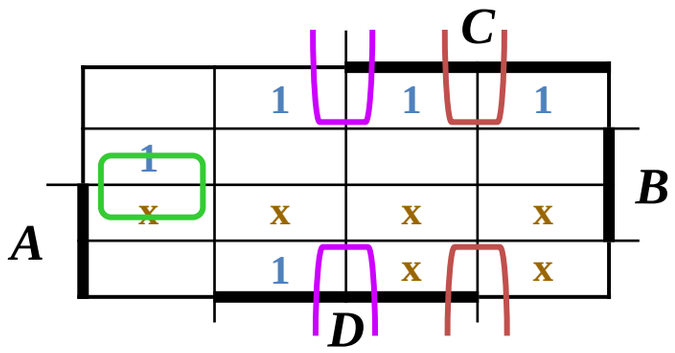
Design Procedure

- BCD-to-Excess 3 Converter

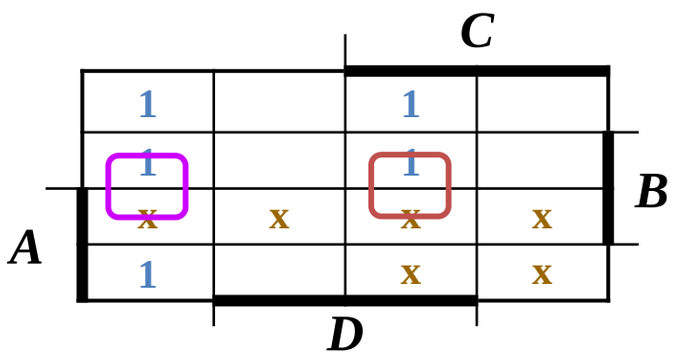
A B C D	w x y z
0 0 0 0	0 0 1 1
0 0 0 1	0 1 0 0
0 0 1 0	0 1 0 1
0 0 1 1	0 1 1 0
0 1 0 0	0 1 1 1
0 1 0 1	1 0 0 0
0 1 1 0	1 0 0 1
0 1 1 1	1 0 1 0
1 0 0 0	1 0 1 1
1 0 0 1	1 1 0 0
1 0 1 0	x x x x
1 0 1 1	x x x x
1 1 0 0	x x x x
1 1 0 1	x x x x
1 1 1 0	x x x x
1 1 1 1	x x x x



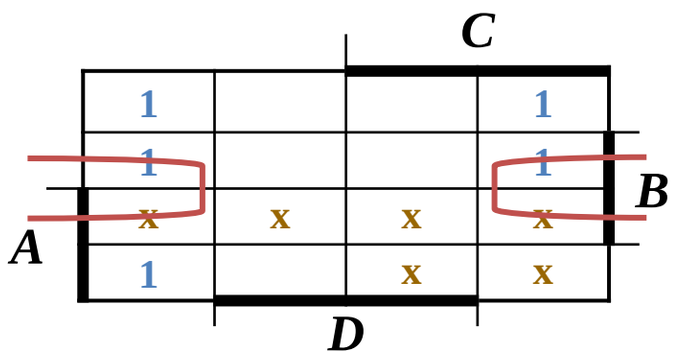
$w = A+BC+BD$



$x = B'C+B'D+BC'D'$



$y = C'D'+CD$

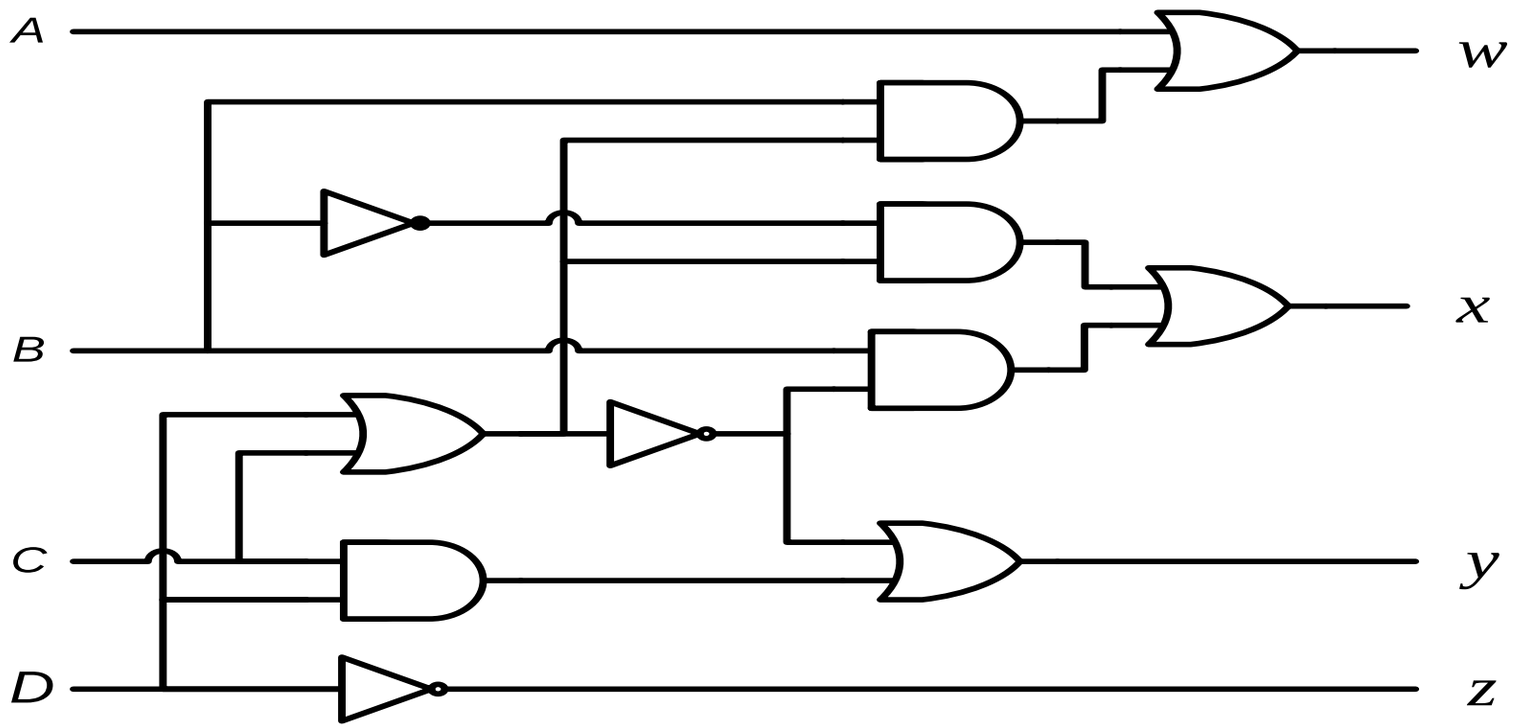


$z = D'$

Design Procedure

- BCD-to-Excess 3 Converter

A B C D	w x y z
0 0 0 0	0 0 1 1
0 0 0 1	0 1 0 0
0 0 1 0	0 1 0 1
0 0 1 1	0 1 1 0
0 1 0 0	0 1 1 1
0 1 0 1	1 0 0 0
0 1 1 0	1 0 0 1
0 1 1 1	1 0 1 0
1 0 0 0	1 0 1 1
1 0 0 1	1 1 0 0
1 0 1 0	x x x x
1 0 1 1	x x x x
1 1 0 0	x x x x
1 1 0 1	x x x x
1 1 1 0	x x x x
1 1 1 1	x x x x



$w = A + B(C+D)$

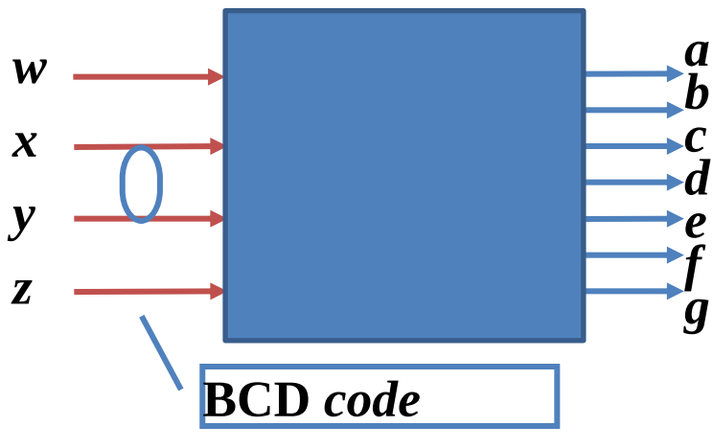
$x = B'(C+D) + B(C+D)'$

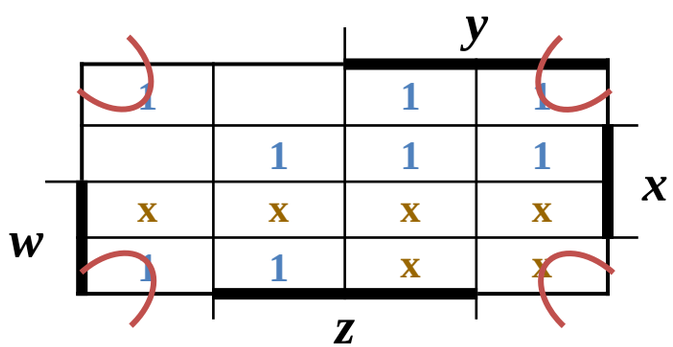
$y = (C+D)' + CD$

$z = D'$

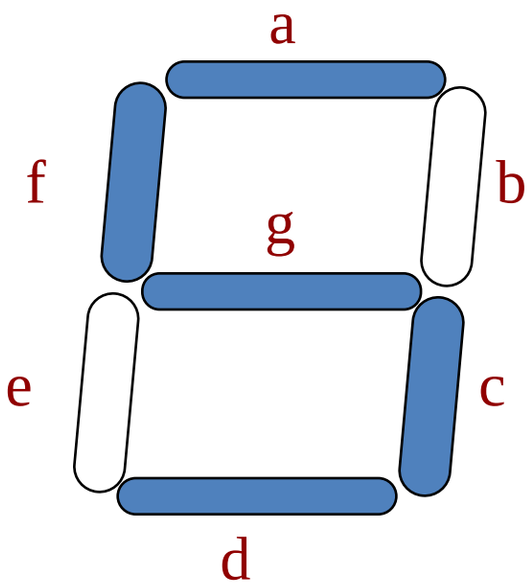
Seven-Segment Decoder

w x y z	a b c d e f g
0 0 0 0	1 1 1 1 1 1 0
0 0 0 1	0 1 1 0 0 0 0
0 0 1 0	1 1 0 1 1 0 1
0 0 1 1	1 1 1 1 0 0 1
0 1 0 0	0 1 1 0 0 1 1
0 1 0 1	1 0 1 1 0 1 1
0 1 1 0	1 0 1 1 1 1 1
0 1 1 1	1 1 1 0 0 0 0
1 0 0 0	1 1 1 1 1 1 1
1 0 0 1	1 1 1 1 0 1 1
1 0 1 0	x x x x x x x
1 0 1 1	x x x x x x x
1 1 0 0	x x x x x x x
1 1 0 1	x x x x x x x
1 1 1 0	x x x x x x x
1 1 1 1	x x x x x x x



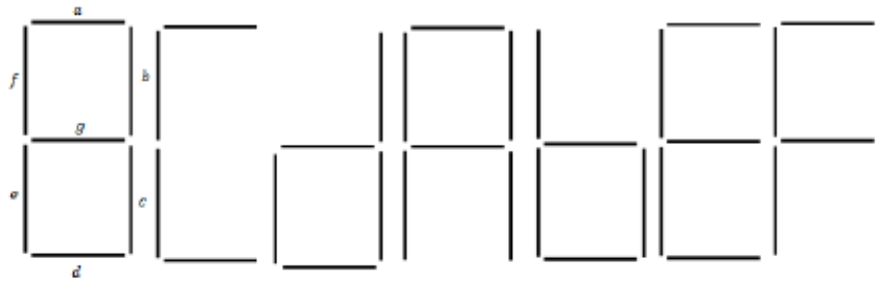


$a = w + y + xz + x'z'$

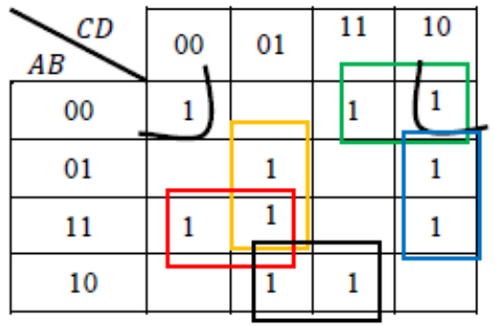


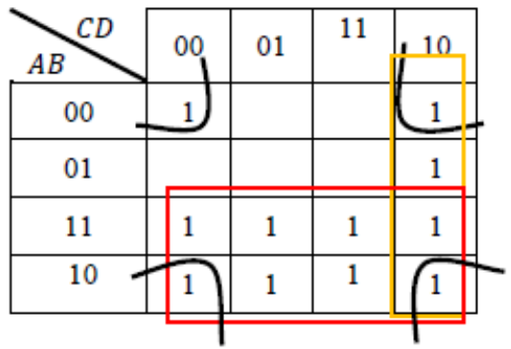
$b = \ldots$
$c = \ldots$
$d = \ldots$

Seven-Segment Decoder



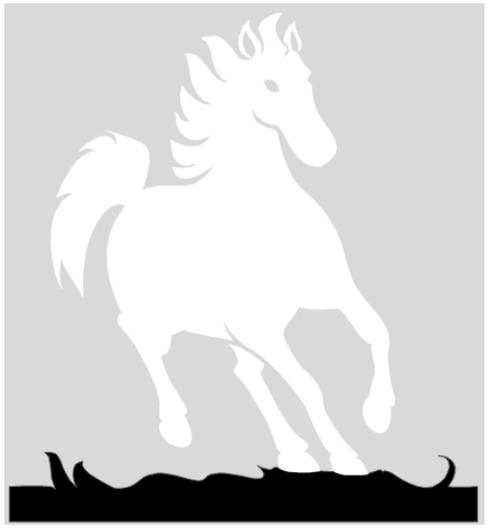
A	B	C	D	d	e
0	0	0	0	1	1
0	0	0	1	0	0
0	0	1	0	1	1
0	0	1	1	1	0
0	1	0	0	0	0
0	1	0	1	1	0
0	1	1	0	1	1
0	1	1	1	0	0
1	0	0	0	1	1
1	0	0	1	0	0
1	0	1	0	0	1
1	0	1	1	1	1
1	1	0	0	1	1
1	1	0	1	1	1
1	1	1	0	1	1
1	1	1	1	0	1





$$d = \bar{A}\bar{B}\bar{D} + \bar{A}\bar{B}C + B\bar{C}D + AB\bar{C} + BC\bar{D} + A\bar{B}D$$

$$e = \bar{B}\bar{D} + A + C\bar{D}$$



Circuits for Binary Addition

Half Adder

Circuits for Binary Addition

With twos complement numbers, addition is sufficient

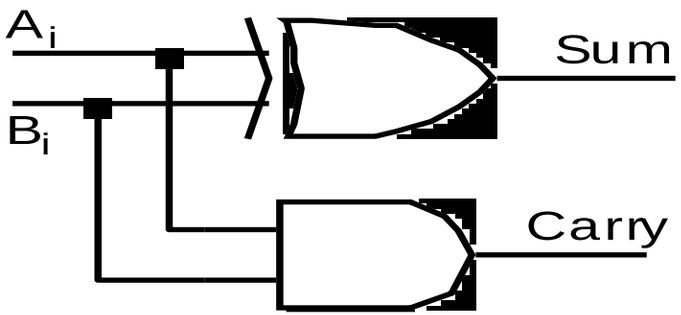
Ai	Bi	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Bi \ Ai	0	1
0	0	1
1	1	0

$$Sum = \overline{Ai}\,Bi + Ai\,\overline{Bi}$$
$$= Ai \oplus Bi$$

Bi \ Ai	0	1
0	0	0
1	0	1

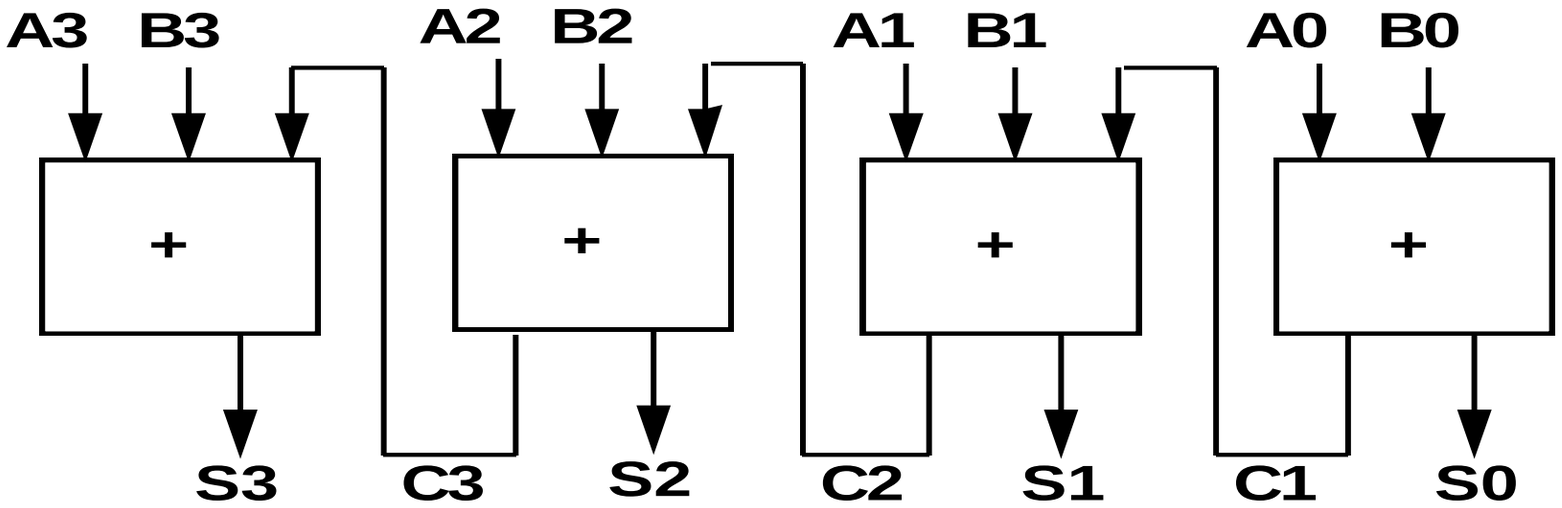
$$Carry = Ai\,Bi$$



Half-adder Schematic

Full Adder

Cascaded Multi-bit Adder

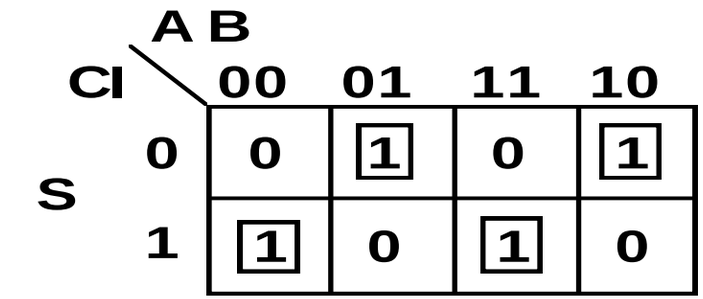


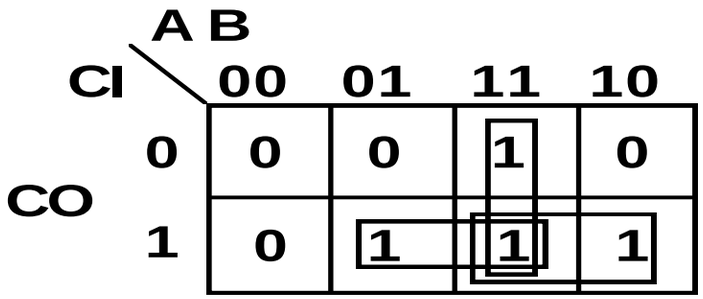
usually interested in adding more than two bits

this motivates the need for the full adder

Full Adder

A	B	CI	S	CO
0	0	0	0	0
0	0	1	1	0
0	1	0	1	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



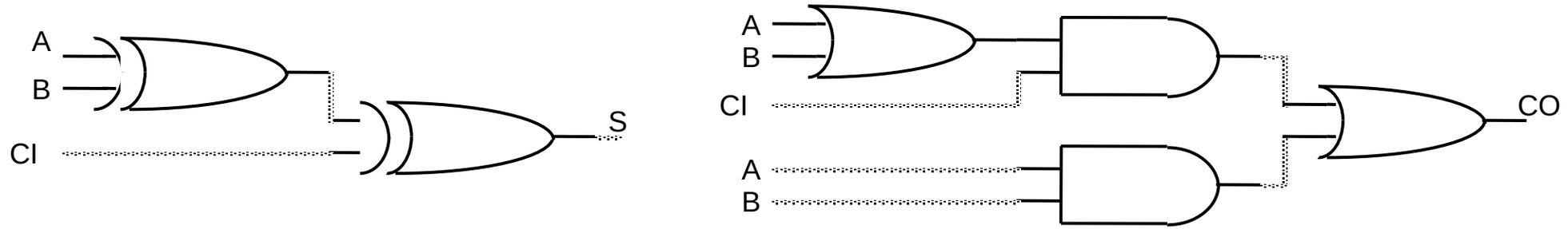


S = CI xor A xor B

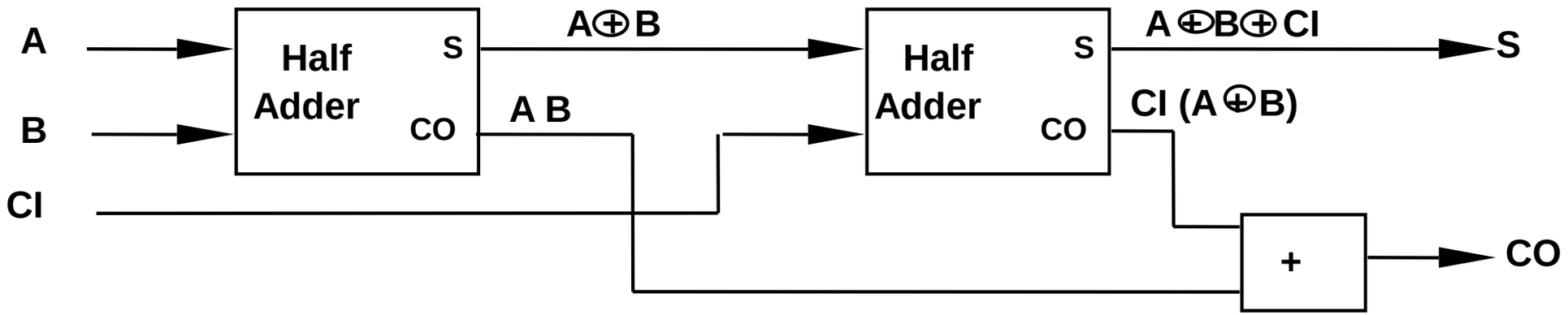
CO = B CI + A CI + A B = CI (A + B) + A B

Full Adder Circuit

Standard Approach: 6 Gates

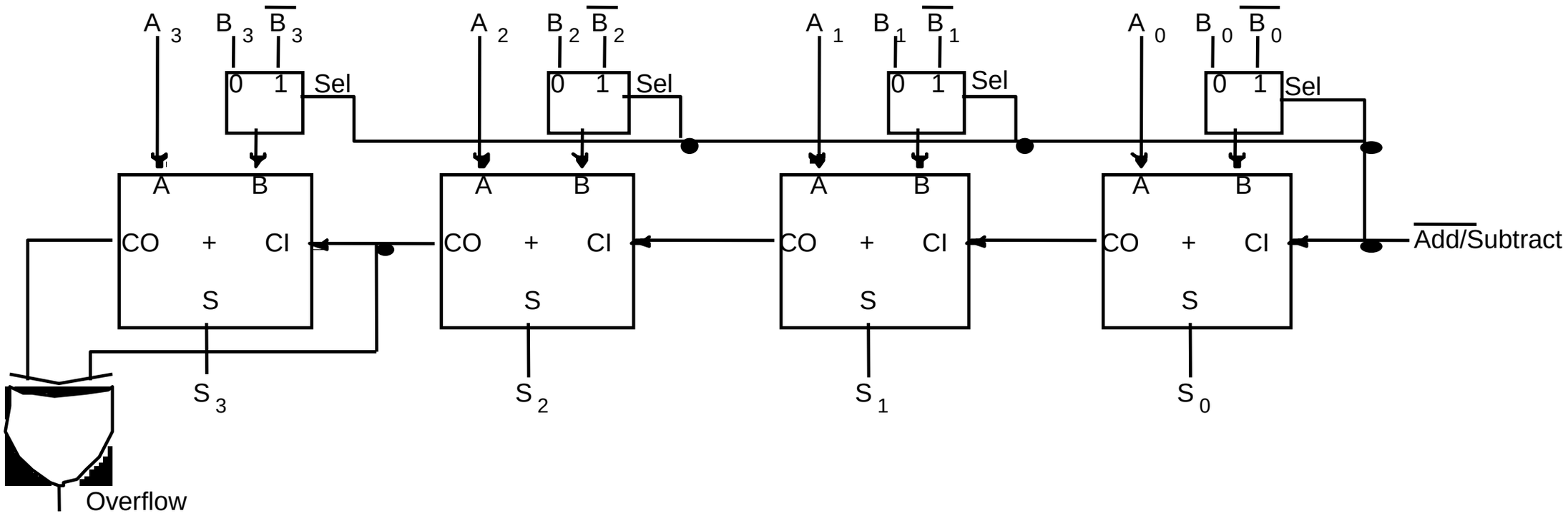


Alternative Implementation: 5 Gates



A B + CI (A xor B) = A B + B CI + A CI

Adder/Subtractor



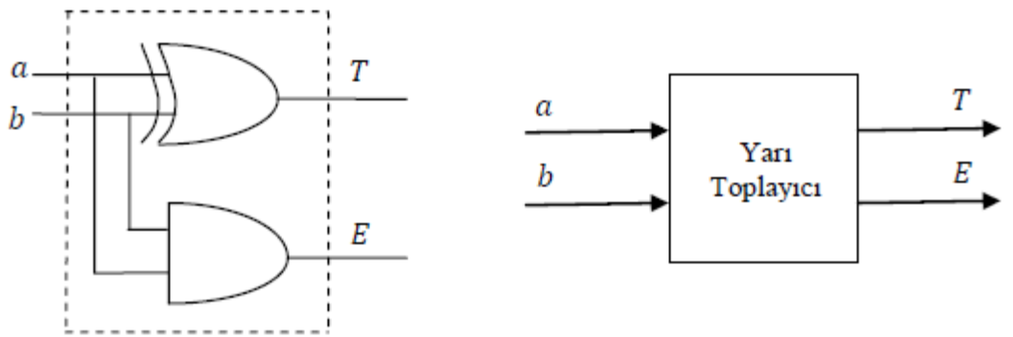
A - B = A + (-B) = A + B + 1

Yarı toplayıcı (Half Adder)

- Yarı toplayıcı elde girişi olmaksızın 1 bitlik iki sayının toplamını bulan bir kombinasyonel devredir.

a	*b*	Toplam *T*	Elde *E*
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

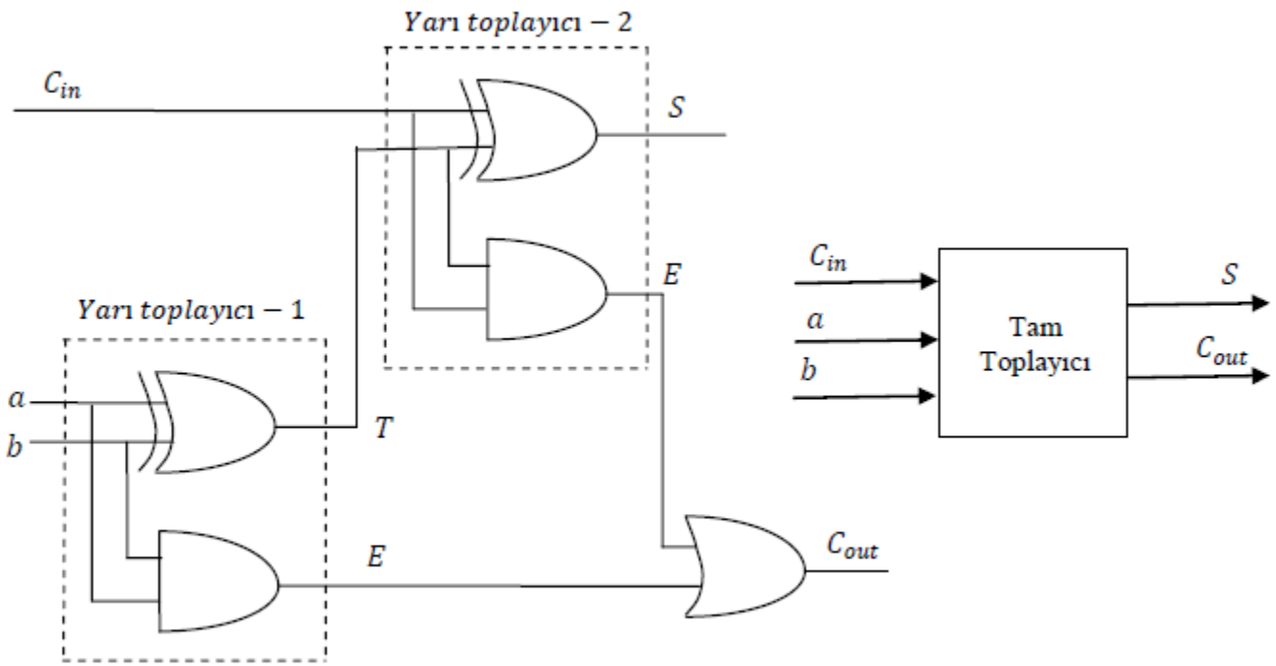
$$T = \bar{a}b + a\bar{b} \qquad E = ab$$



Yarı toplayıcı devresi

Tam toplayıcı (Full Adder)

- Girişinde elde bitinin olduğu ve bir bitlik iki sayı ile birlikte toplandığı bir kombinasyonel devredir.



Tam toplayıcı devresi

C_{in}	a	b	Toplam S	Elde C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	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

C_{in} \ ab	00	01	11	10
0	0	1	0	1
1	1	0	1	0

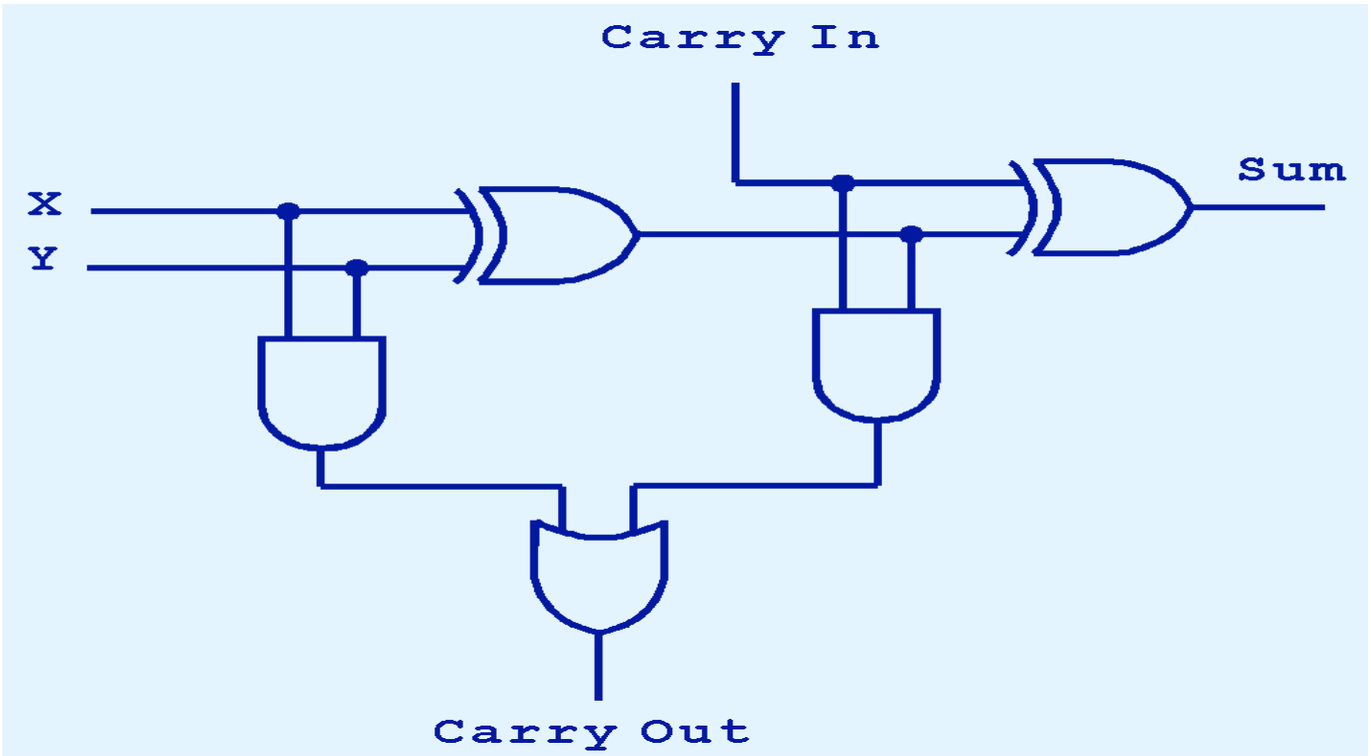
C_{in} \ ab	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$$T = C_{in}\bar{a}\bar{b} + \overline{C_{in}}\bar{a}b + C_{in}ab + \overline{C_{in}}a\bar{b} = \overline{C_{in}}(\bar{a}b + a\bar{b}) + C_{in}(ab + \bar{a}\bar{b}) \Longrightarrow T = a \oplus b \oplus C_{in}$$

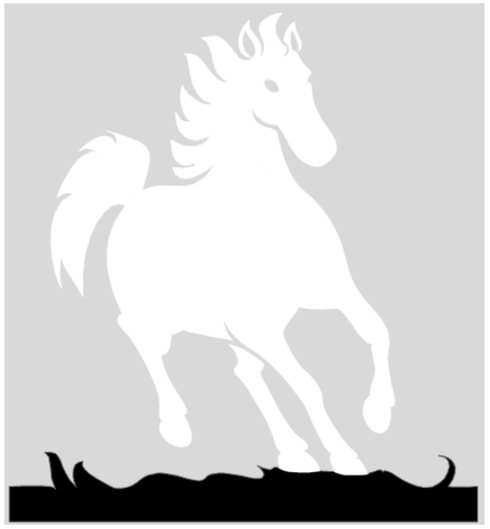
$$C_{out} = C_{in}b + C_{in}a + ab$$

Combinational Circuits

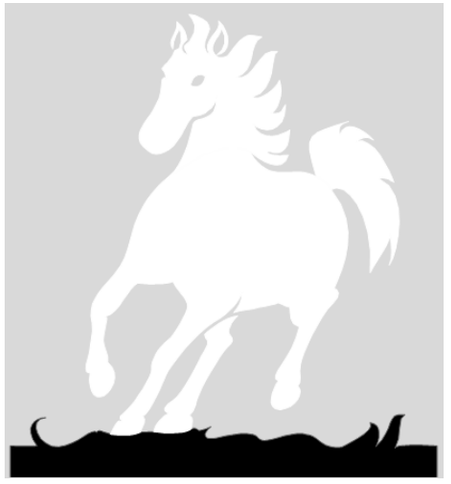
- full adder.



Inputs			Outputs	
X	Y	Carry In	Sum	Carry Out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	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



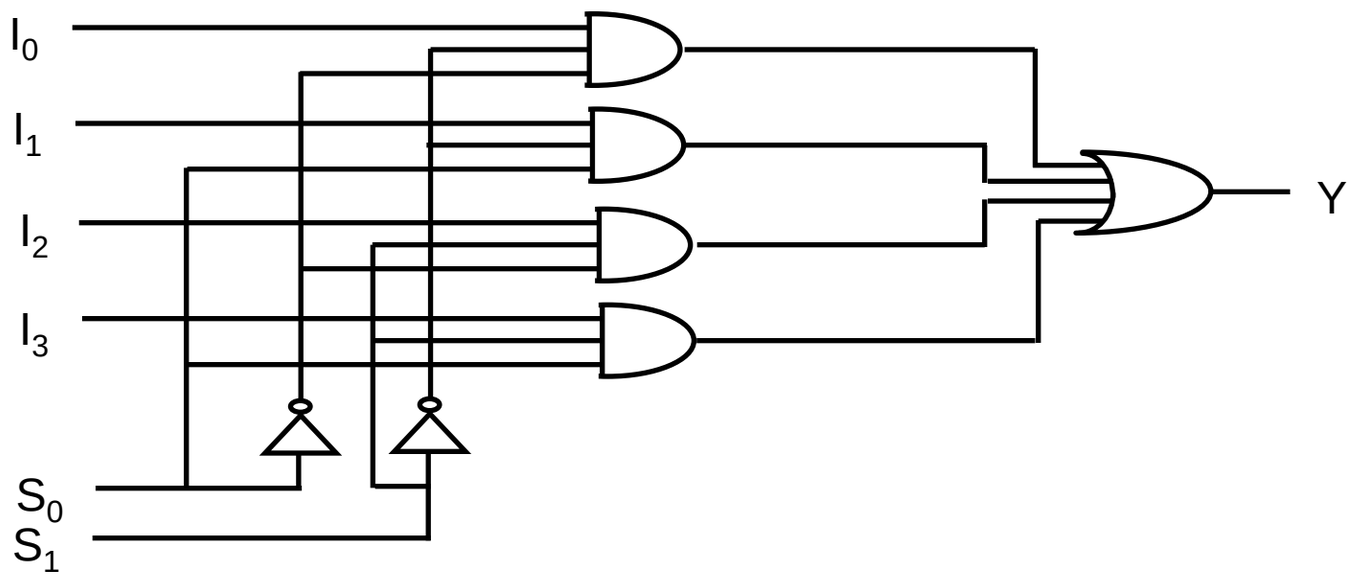
Multiplexers



MULTIPLEXER

4-to-1 Multiplexer

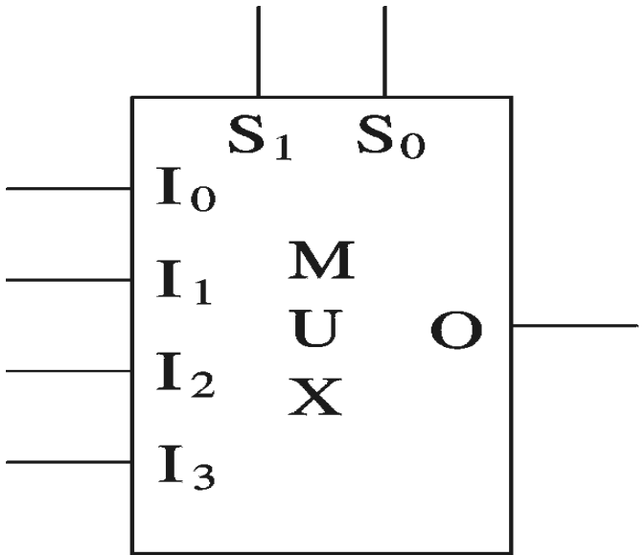
Select		Output
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



Multiplexers

- Multiplexer
  - 2^n data inputs
  - n selection inputs
  - a single output
- Selection input determines the input that should be connected to the output

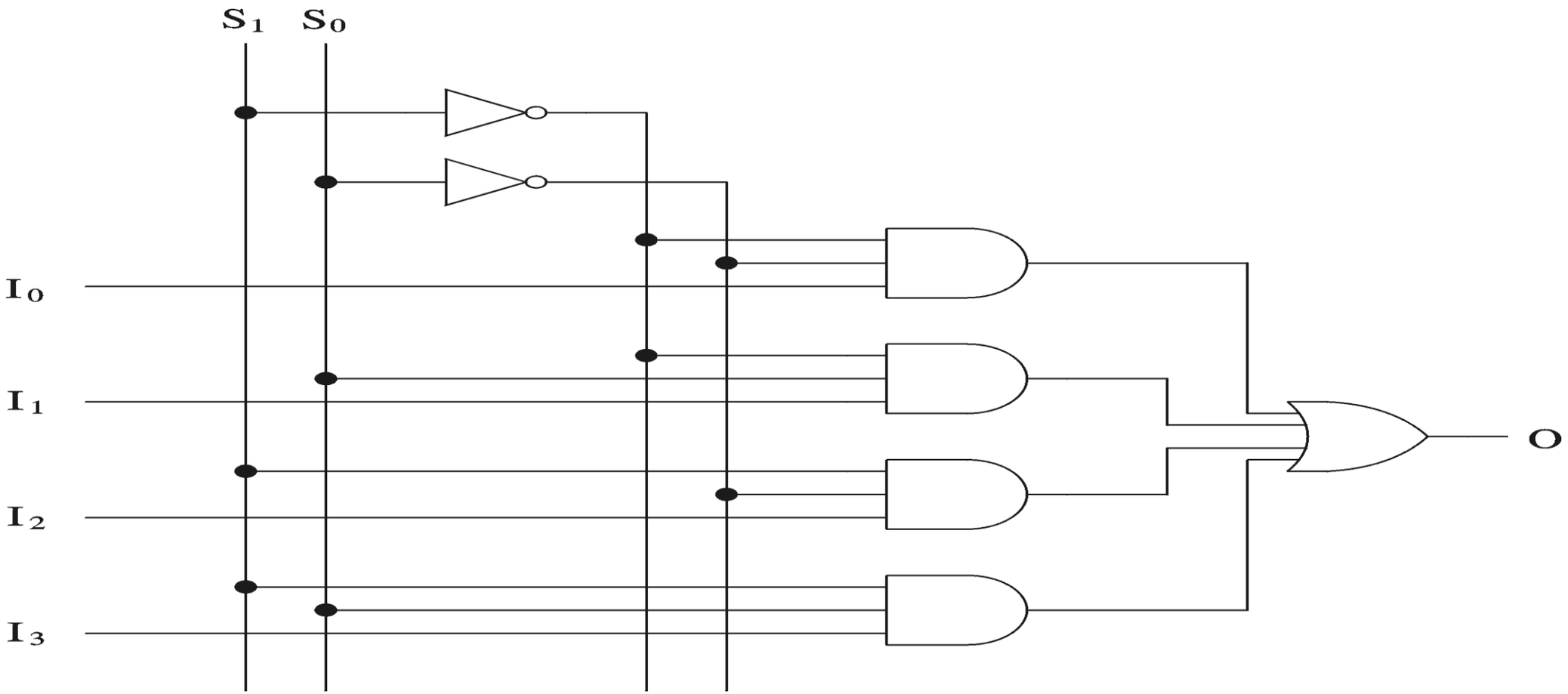
4-data input MUX



S_1	S_0	O
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

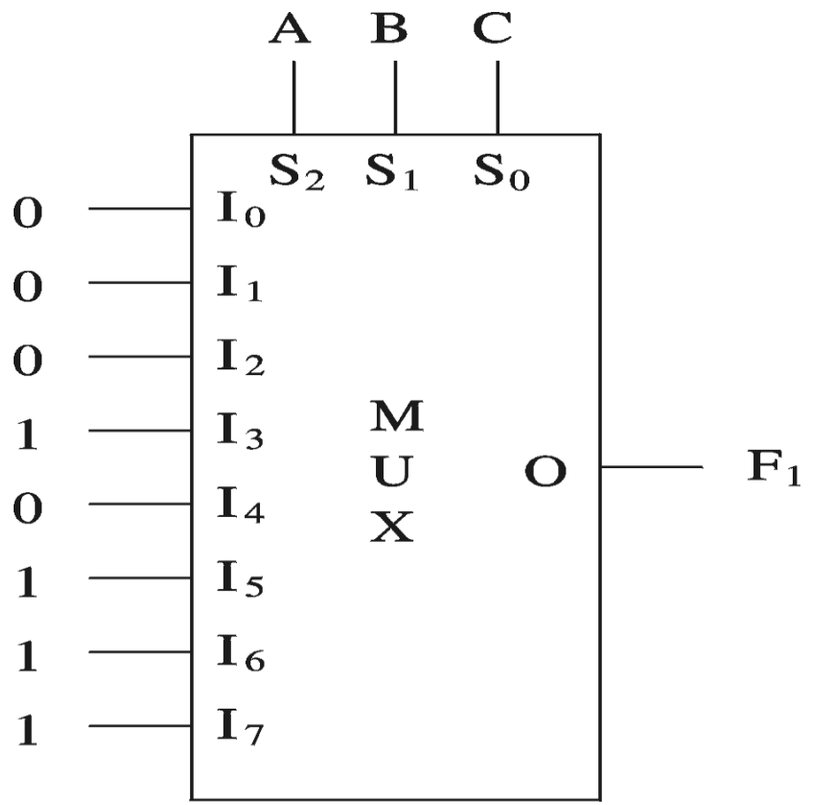
Multiplexers

4-data input MUX implementation

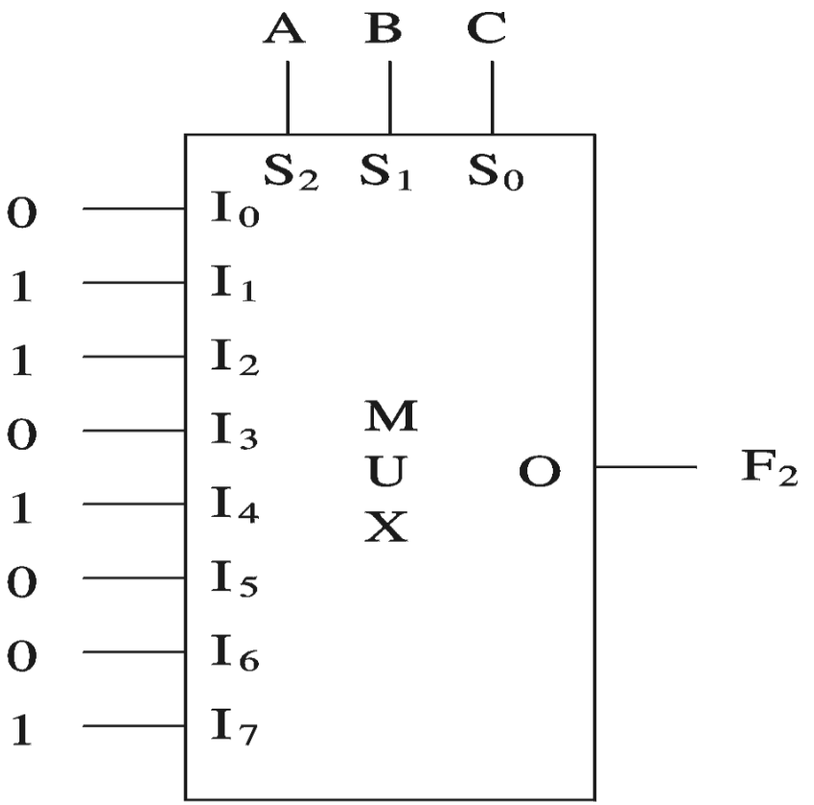


Multiplexers

MUX implementations



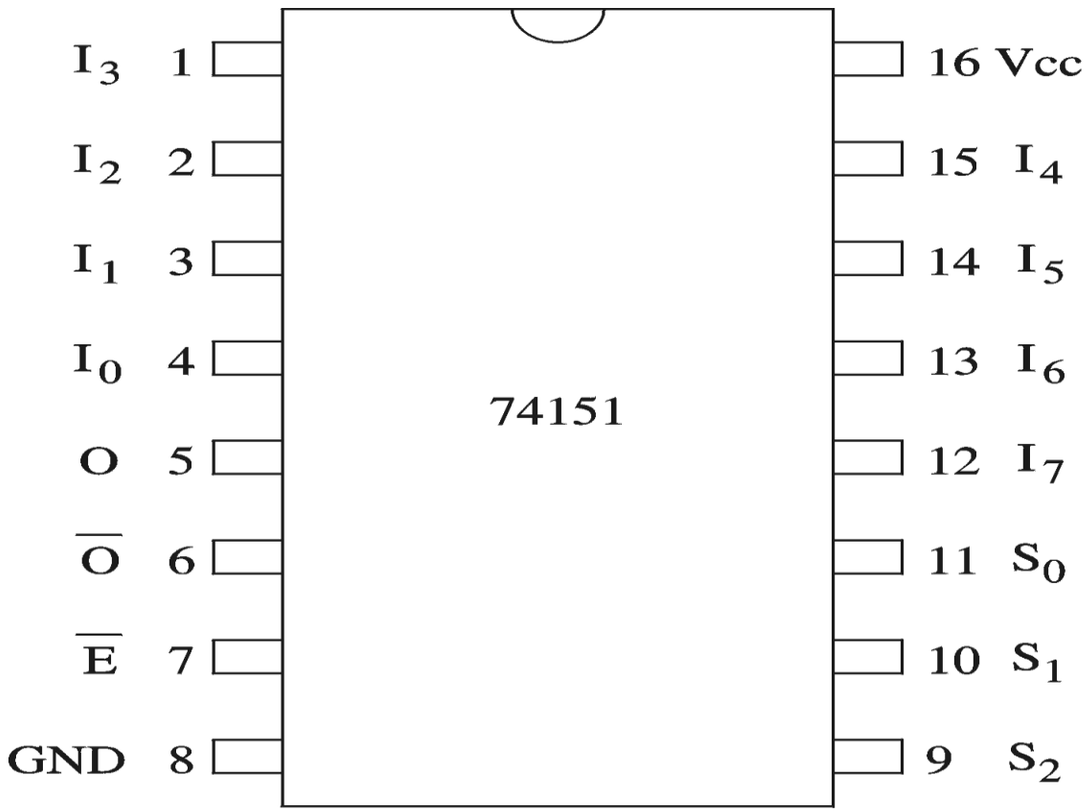
Majority function



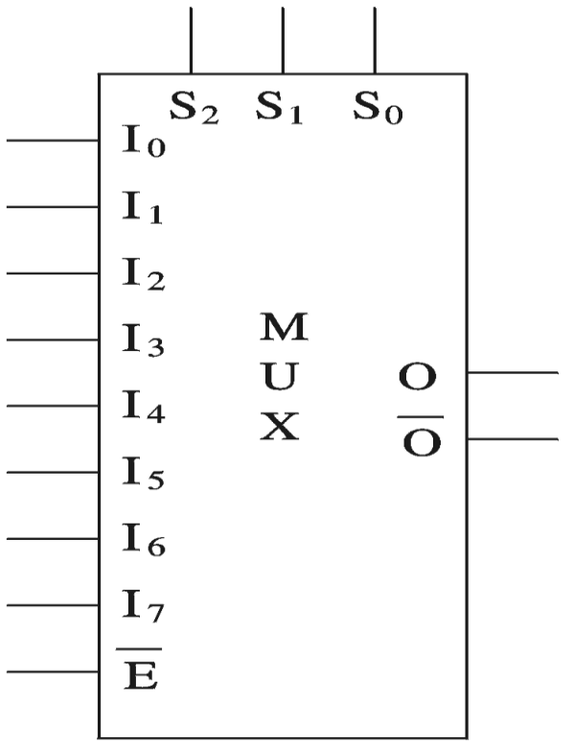
Even-parity function

Multiplexers

Example chip: 8-to-1 MUX



(a) Connection diagram



(b) Logic symbol

Multiplexers

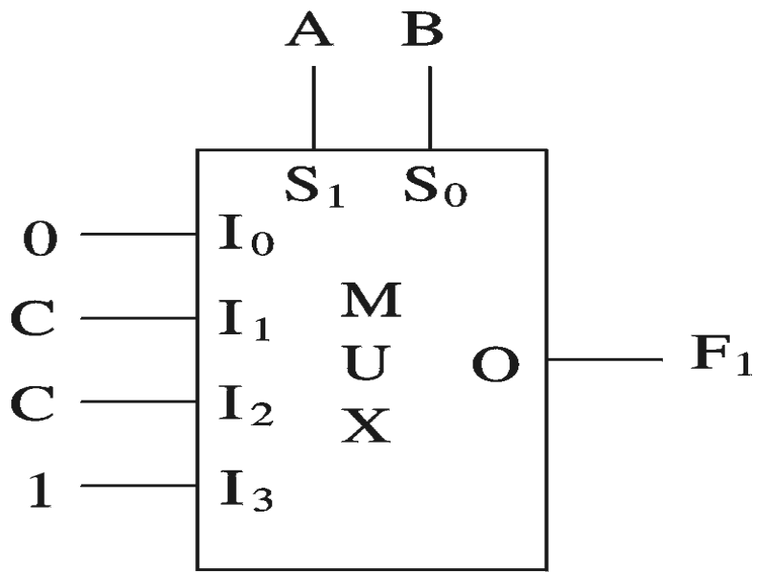
Efficient implementation: Majority function

Original truth table

A	B	C	F_1
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

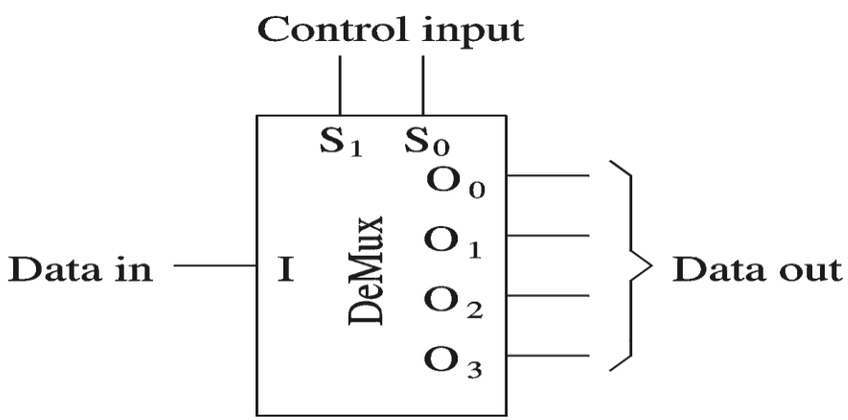
New truth table

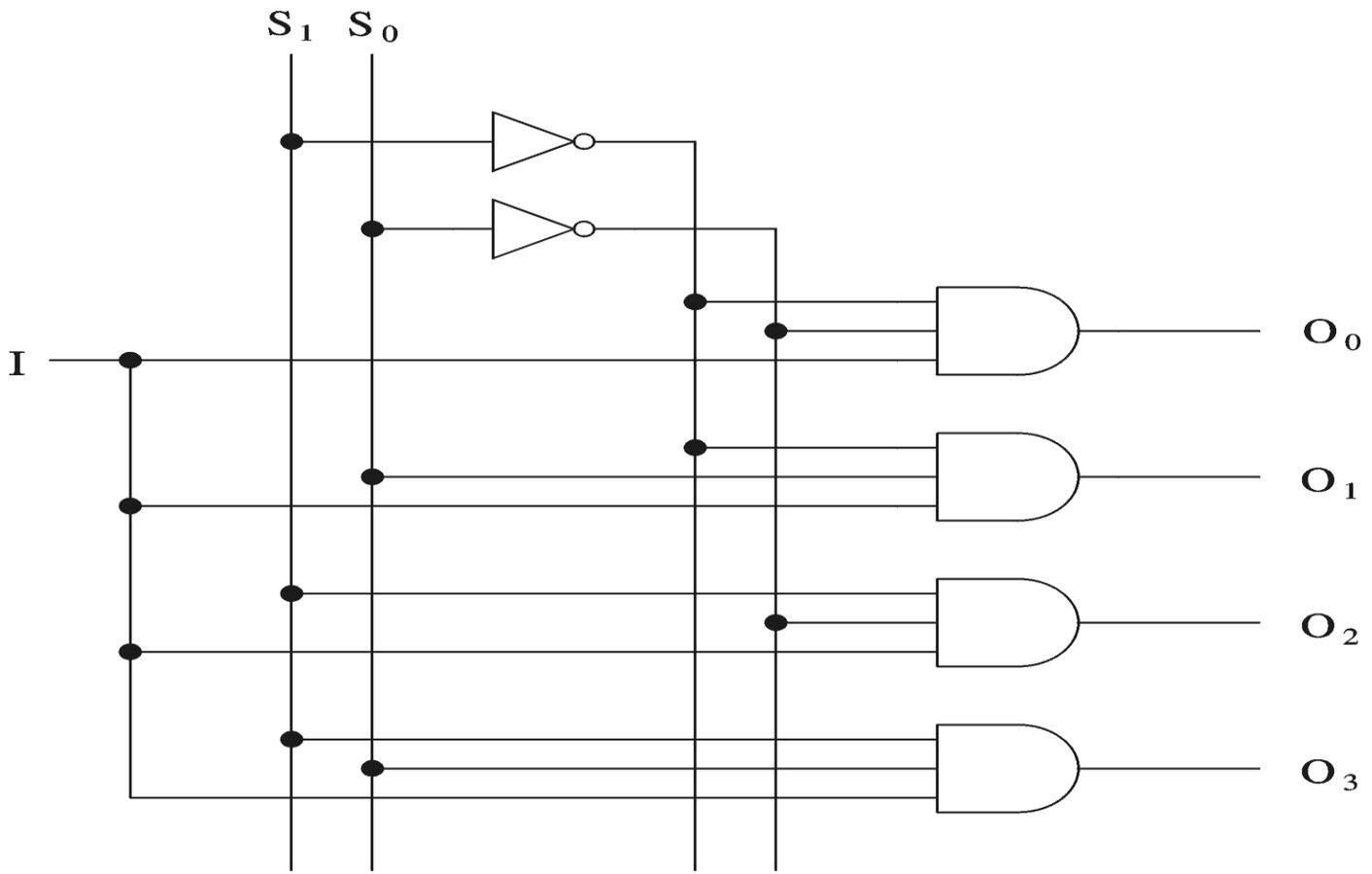
A	B	F_1
0	0	0
0	1	C
1	0	C
1	1	1

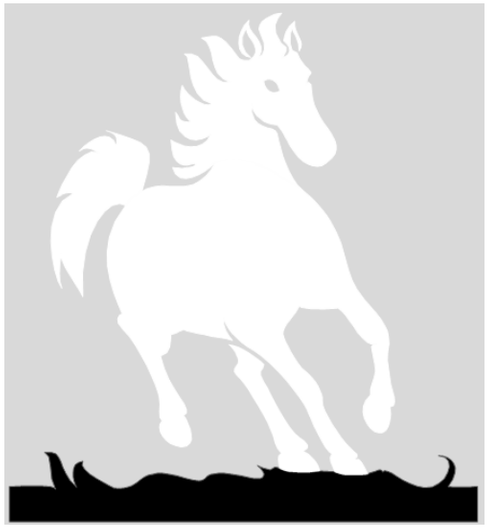


Demultiplexers

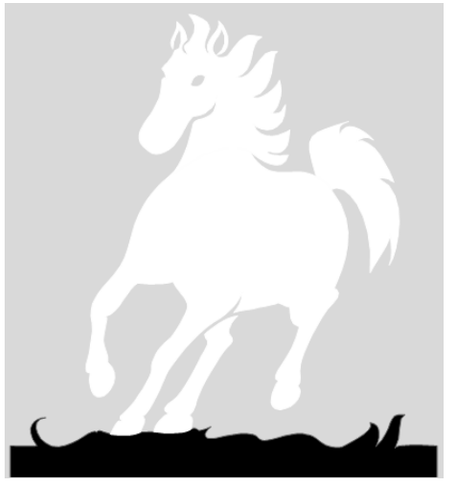
Demultiplexer (DeMUX)







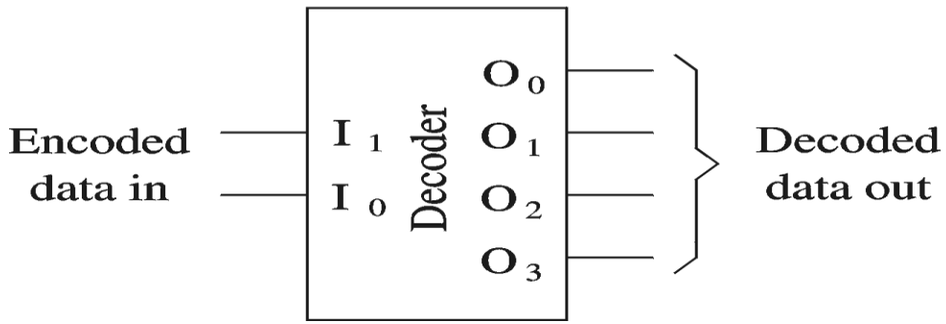
Decoders

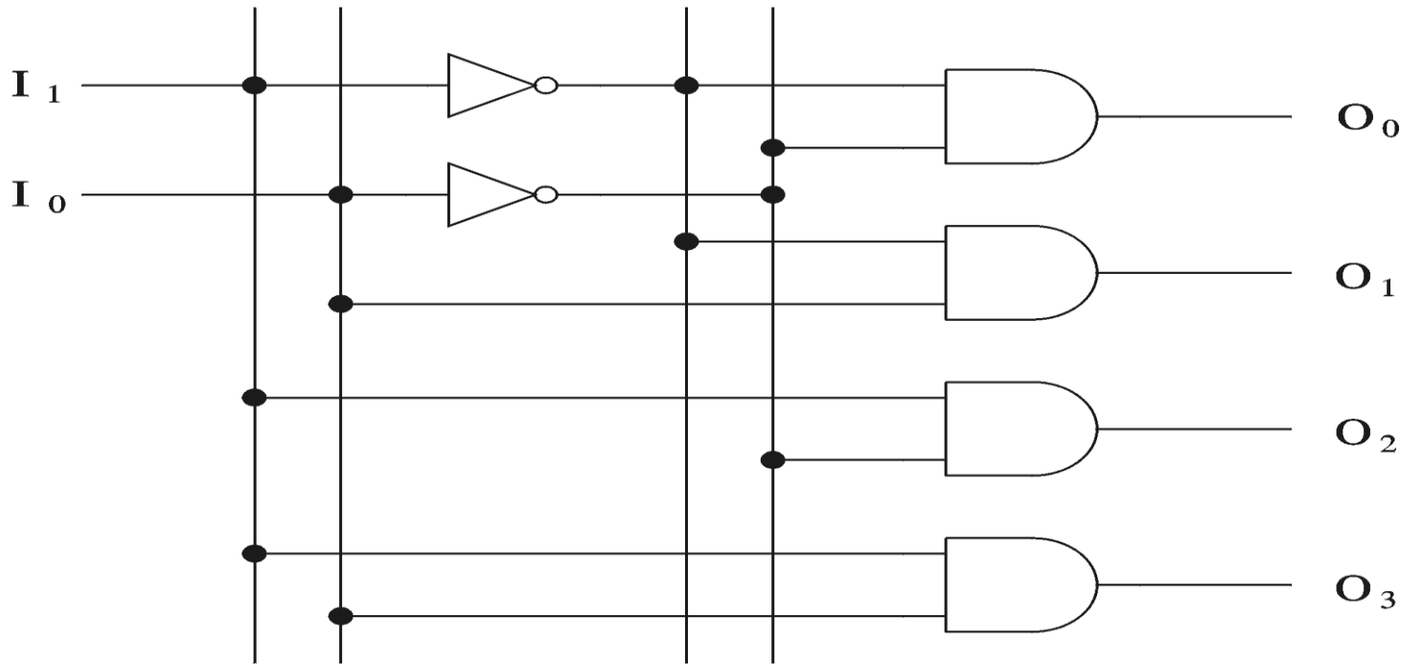


Decoders

- Decoder selects one-out-of-N inputs

I_1	I_0	O_3	O_2	O_1	O_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0



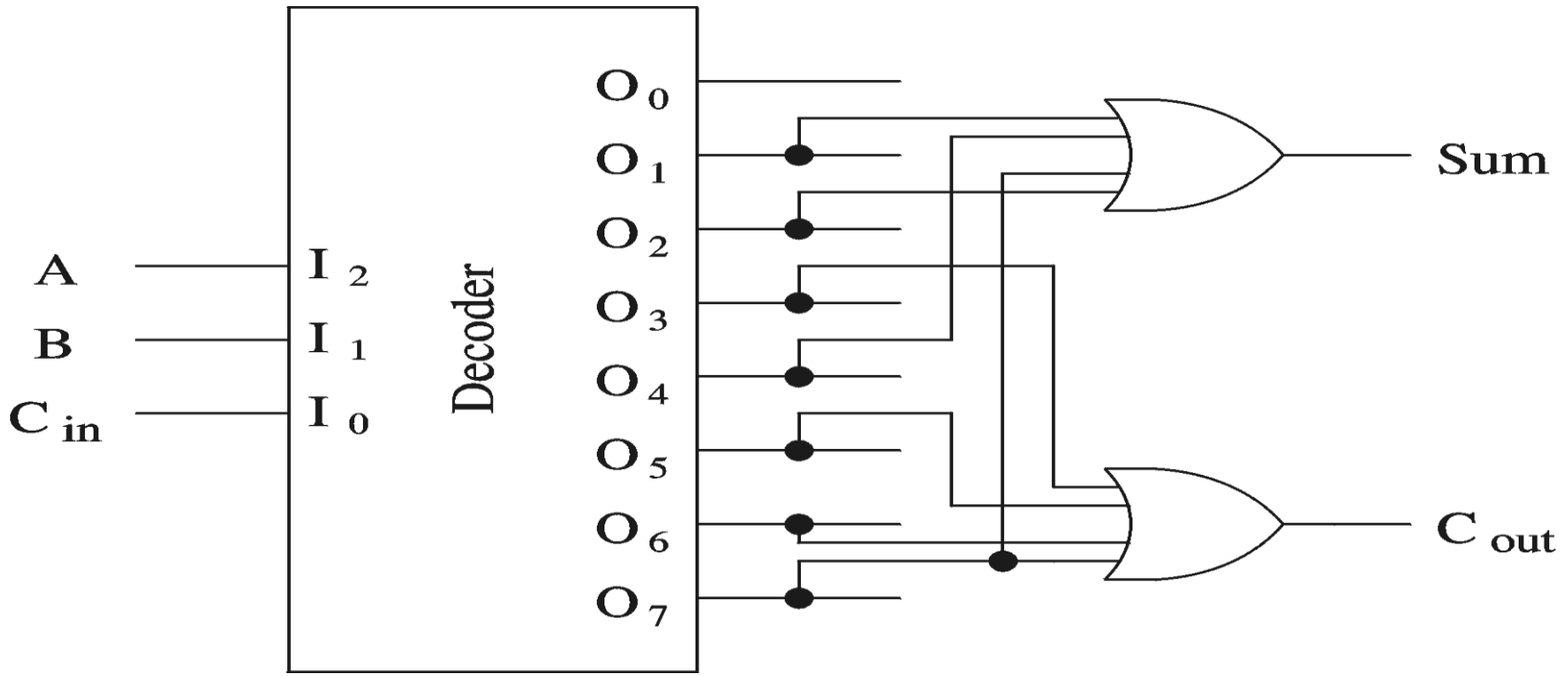


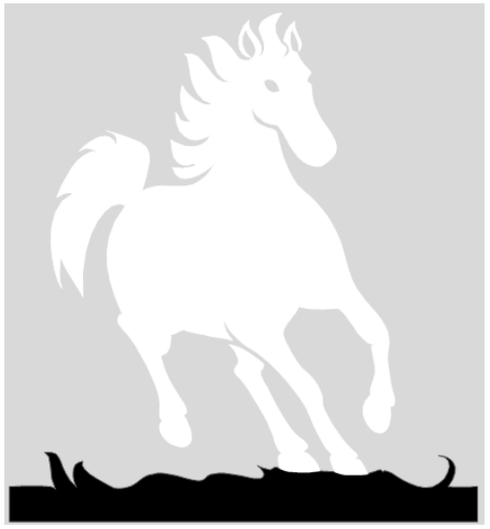
Decoders

Logic function implementation

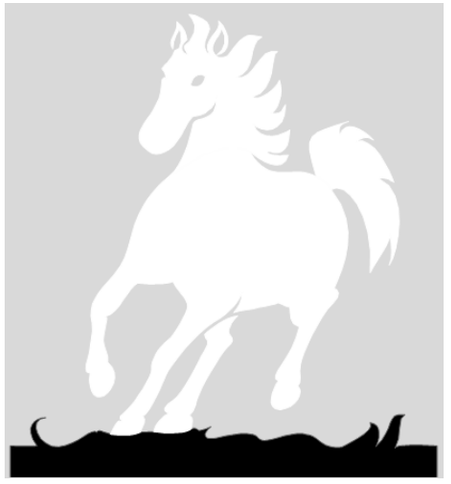
A	B	C_{in}	Sum	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

(Full Adder)



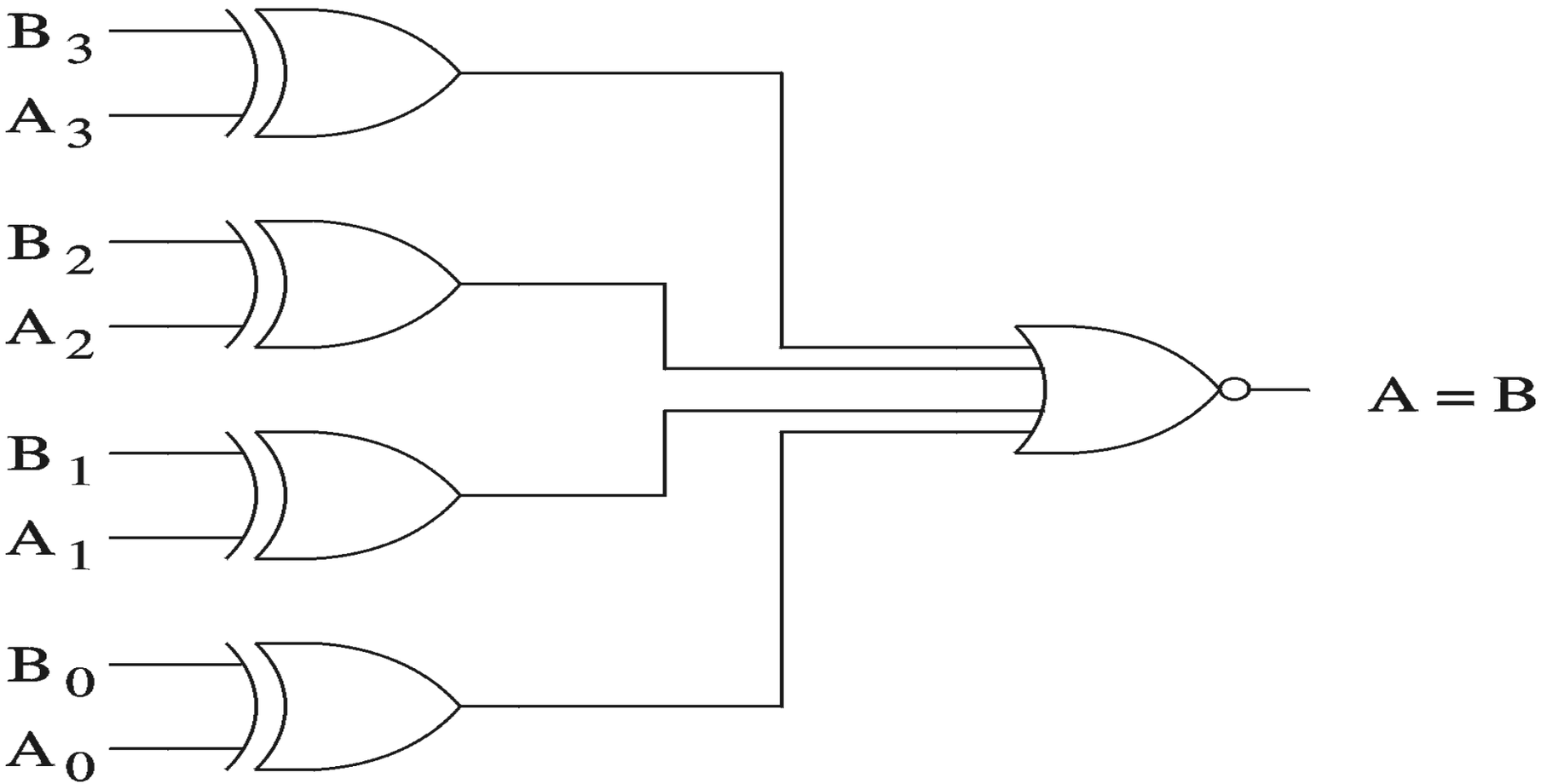


Comparator



Comparator

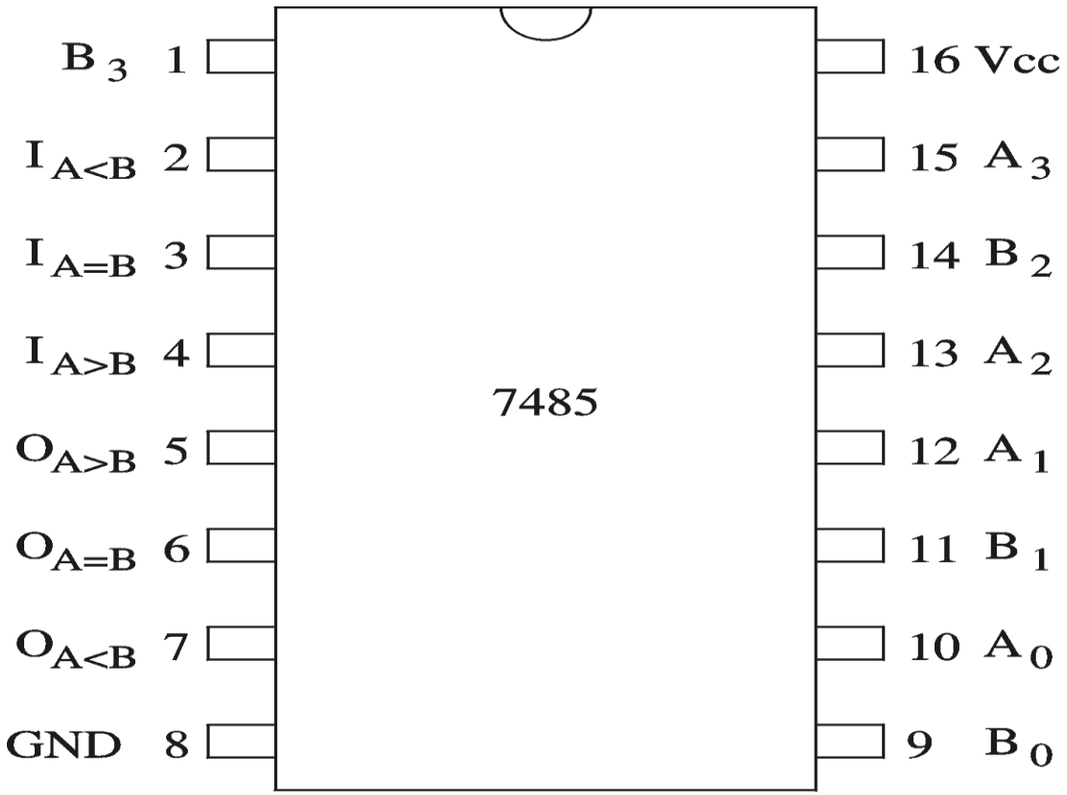
- Used to implement comparison operators (= , > , < , ≥ , ≤)



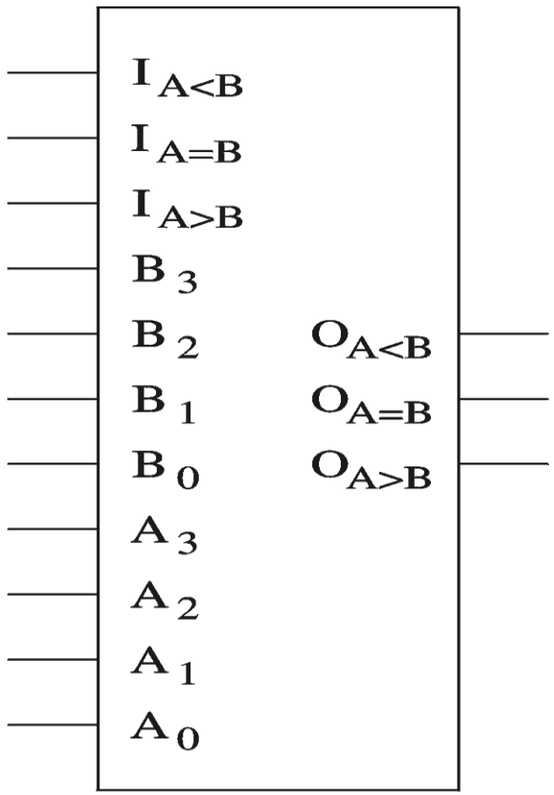
Comparator

A=B: $O_x = I_x$ (x=A<B, A=B, & A>B)

4-bit magnitude comparator chip



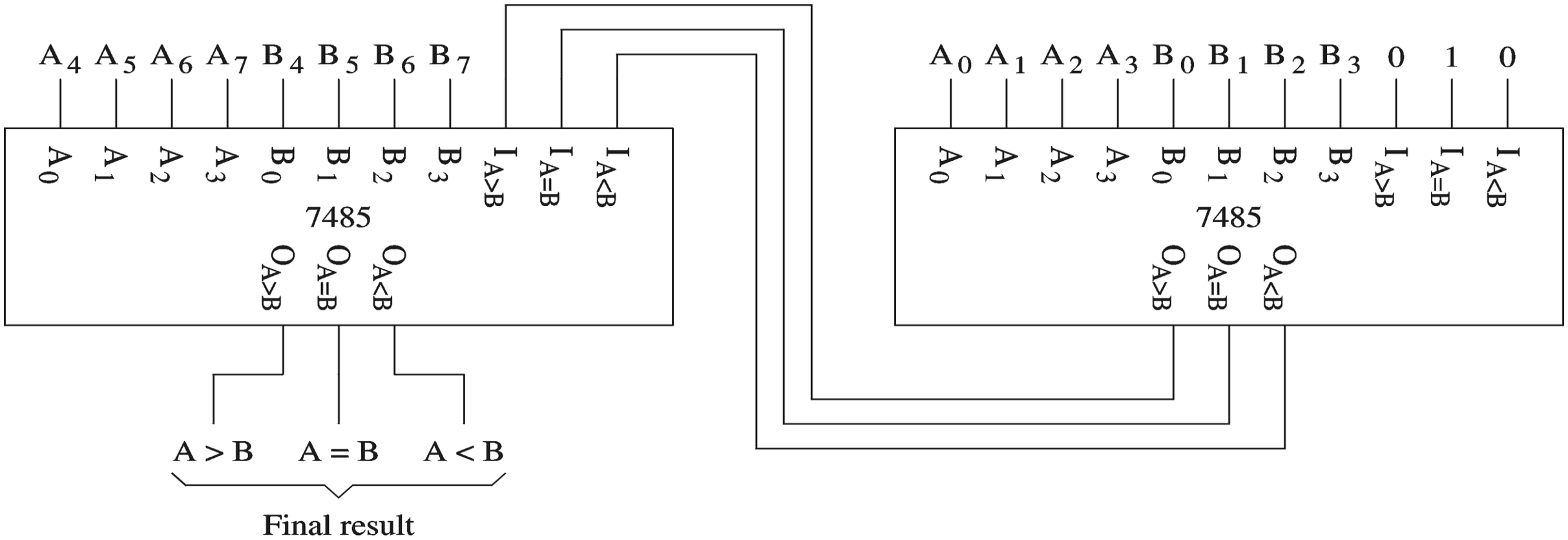
(a) Connection diagram



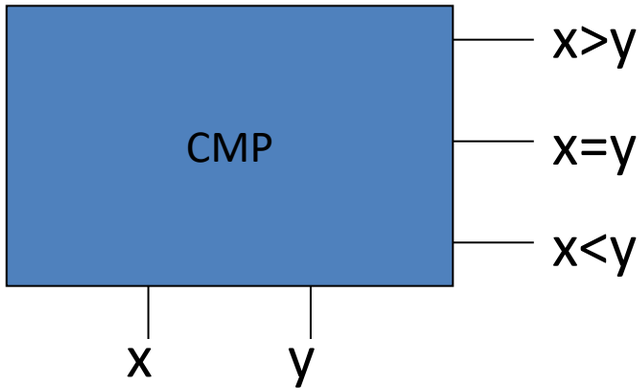
(b) Logic symbol

Comparator

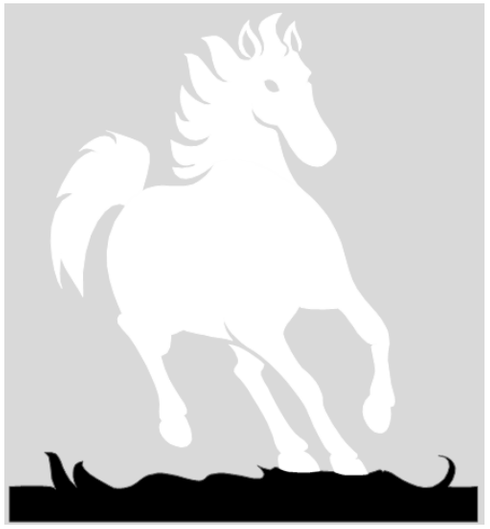
Serial construction of an 8-bit comparator



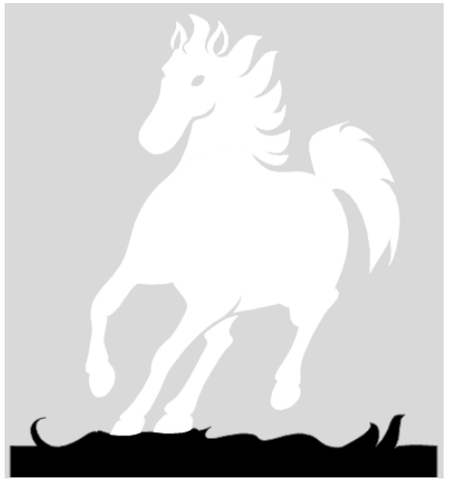
1-bit Comparator



x	y	x>y	x=y	x<y



Adders

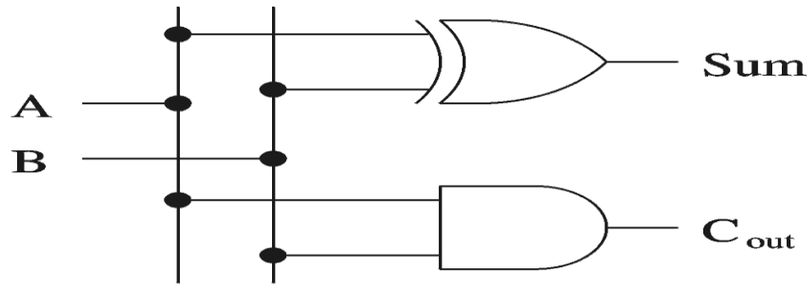


Adders

- Half-adder
  - Adds two bits
    - Produces a *sum* and *carry*
  - Problem: Cannot use it to build larger inputs
- Full-adder
  - Adds three 1-bit values
    - Like half-adder, produces a *sum* and *carry*
  - Allows building N-bit adders
    - Simple technique
      - Connect C_{out} of one adder to C_{in} of the next
    - These are called *ripple-carry adders*

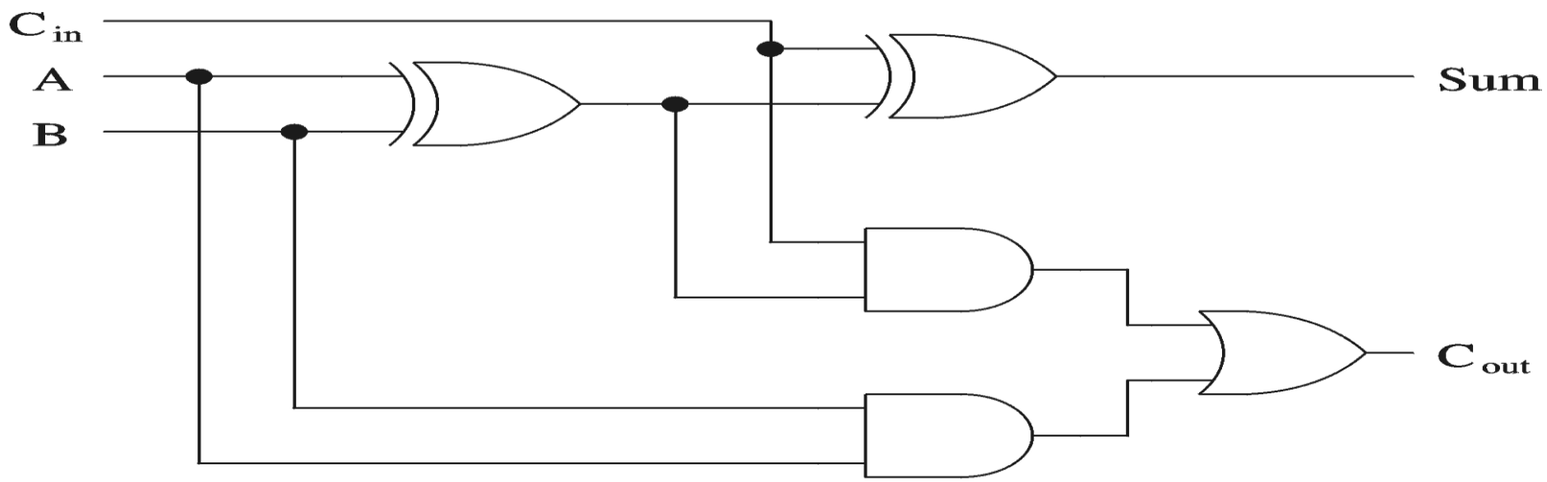
Adders

A	B	Sum	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



(a) Half-adder truth table and implementation

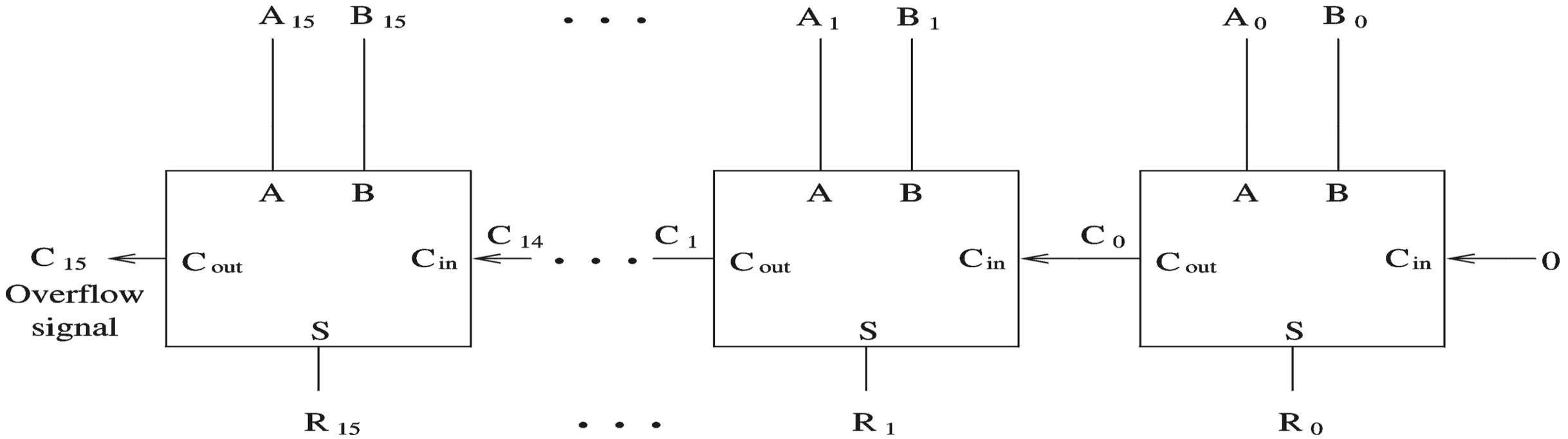
A	B	C_{in}	Sum	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



(b) Full-adder truth table and implementation

Adders

A 16-bit ripple-carry adder

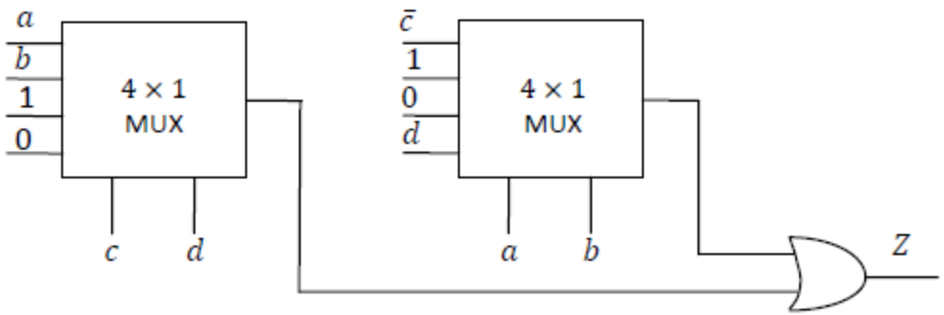


Adders

- Ripple-carry adders can be slow
  - Delay proportional to number of bits
- Carry lookahead adders
  - Eliminate the delay of ripple-carry adders
  - Carry-ins are generated independently
    - $C_0 = A_0 B_0$
    - $C_1 = A_0 B_0 A_1 + A_0 B_0 B_1 + A_1 B_1$
    - . . .
  - Requires complex circuits
  - Usually, a combination carry lookahead and ripple-carry techniques are used

Örnek

- Devreye ait Z lojik fonksiyonunu doğruluk tablosu ile oluşturarak Karnough diyagramı yardımıyla elde ediniz.



a	b	c	d	F_1	F_2	Z
0	0	0	0	0	1	1
0	0	0	1	0	1	1
0	0	1	0	1	0	1
0	0	1	1	0	0	0
0	1	0	0	0	1	1
0	1	0	1	1	1	1
0	1	1	0	1	1	1
0	1	1	1	0	1	1
1	0	0	0	1	0	1
1	0	0	1	0	0	0
1	0	1	0	1	0	1
1	0	1	1	0	0	0
1	1	0	0	1	0	1
1	1	0	1	1	1	1
1	1	1	0	1	0	1
1	1	1	1	0	1	1

c	d	F_1
0	0	a
0	1	b
1	0	1
1	1	0

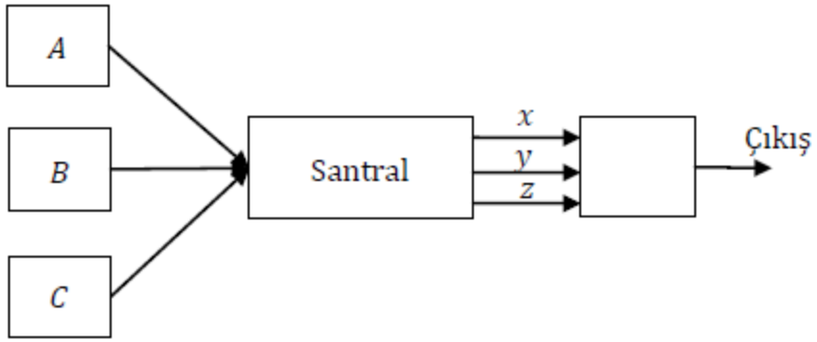
a	b	F_2
0	0	$\bar{c}$
0	1	1
1	0	-

ab \ cd	00	01	11	10
00	1	1	0	1
01	1	1	1	1
11	1	1	1	1
10	1	0	0	1

$$Z = b + \bar{a}\bar{c} + \bar{d}$$

Örnek

- Şekildeki telefon sisteminde konuşmada öncelik sırası A,B ve C' dir. Santral bu önceliği seçerek çıkış verecektir. Bu sistemi gerçekleştiriniz (Konuşma isteğinde santral 1 sinyali verecektir).



A	B	C	x	y	z
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	0	0
1	1	0	1	0	0
1	1	1	1	0	0

$$f(x) = A,$$

$$f(y) = \bar{A}B\bar{C} + \bar{A}BC = \bar{A}B(\bar{C} + C)$$

$$f(y) = \bar{A}B,$$

$$f(z) = \bar{A}\bar{B}C$$

Kaynakça

- https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-111-introductory-digital-systems-laboratory-spring-2006/lecture-notes/
- http://web.ee.nchu.edu.tw/~cpfan/FY92b-digital/Chapter-4.ppt
- http://www.cs.nccu.edu.tw/~whliao/ds2003/ds4.ppt
- http://www.just.edu.jo/~tawalbeh/cpe252/slides/CH1_2.ppt
- Lessons In Electric Circuits, Volume IV { Digital By Tony R. Kuphaldt Fourth Edition, last update July 30, 2004.
- Digital Electronics Part I – Combinational and Sequential Logic Dr. I. J. Wassell.
- Digital Design With an Introduction to the Verilog HDL, M. Morris Mano Emeritus Professor of Computer Engineering California State University, Los Angeles; Michael D. Ciletti Emeritus Professor of Electrical and Computer Engineering University of Colorado at Colorado Springs.
- Digital Logic Design Basics, Combinational Circuits, Sequential Circuits, Pu-Jen Cheng.