“Digital Design”

Dr. Cahit Karakuş, February-2019

İçerik

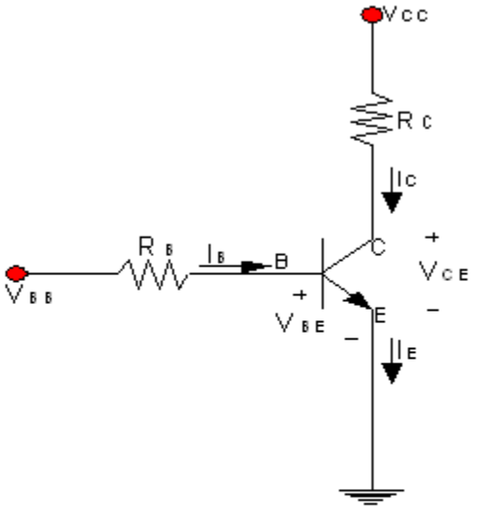
Sayısal Tasarım

- Boolean Algebra
- Simplify the function
- The Karnaugh Map
- Combinational Circuits
- Sequential Logic
- Counters
- State Tables

Sayısal Mantığın Temelleri

- Sayısal mantık değerlerini temsil etmek için ikili sayı sistemi kullanılır: 1/0 (Doğru /Yanlış, İyi /Kötü, Gece/Gündüz, 0V /5V)
- Sayısal mantık değerlerinin matematiksel işlemleri, Boole cebirinin kurallarında belirtilen yasalar tabidir.
- Boole cebirinin kurallarında belirtilen yasaların matematiksel girdileri ve çıktıları ikili sayı (1/0) sistemi ile temsil edilir.
- Bilgisayar sistemlerinin donanımını oluşturan mantıksal kapıları
    - AND, OR, NOT, NAND, NOR, XOR, …
- Mantık kapıları, transistörler kullanılarak oluşturulur.
    - NOT gate can be implemented by a single transistor
    - AND gate requires 3 transistors
- Transistörler bilgisayar sistemlerinin temel devre elemanlarıdır.
    - Pentium consists of 3 million transistors
    - Compaq Alpha consists of 9 million transistors
    - Now we can build chips with more than 100 million transistors

Gates and Transistors



$$V_{CC} = R_C * I_C + V_{CE}$$

$$V_{BB} = R_B * I_B + V_{BE}$$

$$I_C = \beta * I_B$$

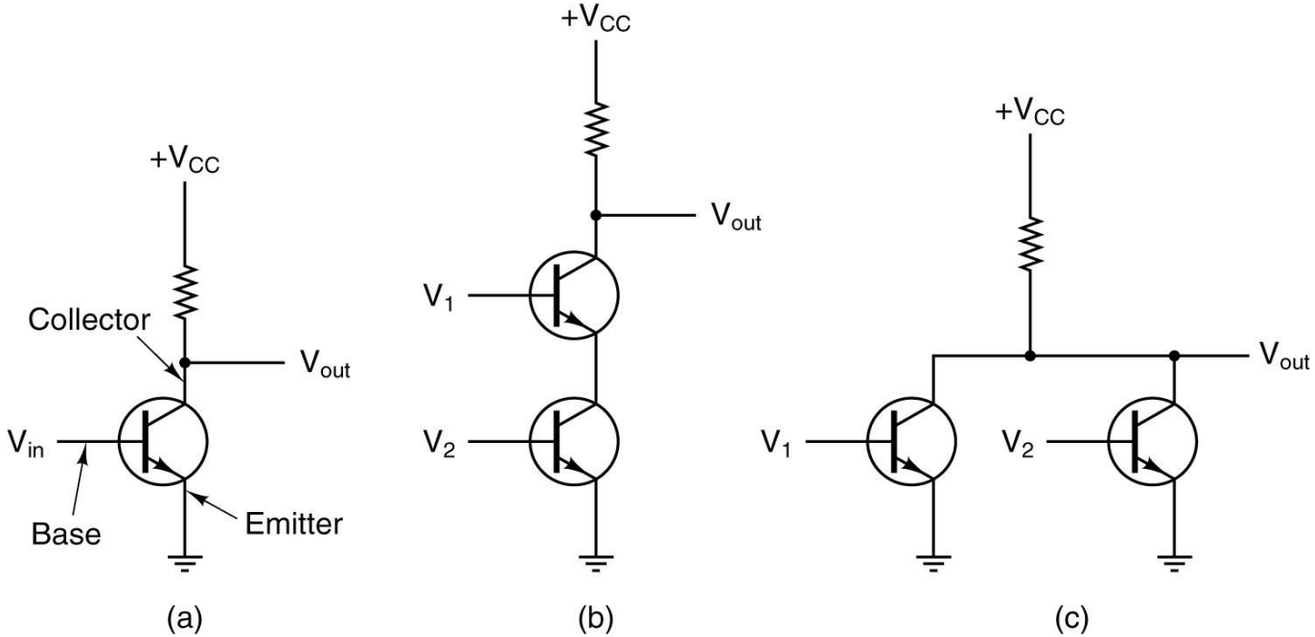
$$I_{C\ SAT} = \frac{V_{CC}}{R_C}$$

$$I_B = \frac{V_{BB} - V_{BE}}{R_B}$$

$V_{CE} \leq 0\ V$ *ya da* $I_c \geq I_{c\ SAT}$; *Saturasyon* $V_{CE} = 0V$ *Olur*

$I_B \leq 0\ A$ *ise* ; *Kesmede* $I_B = I_C = 0\ A$ *Olur*

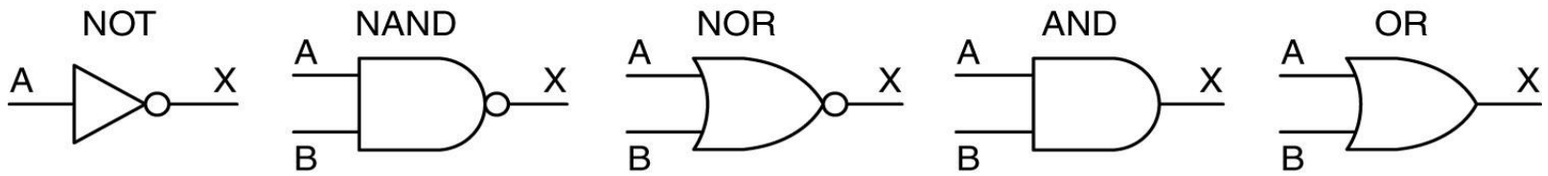
VCE=VCC kesme durumunda



(a) A transistor inverter.
(b) A NAND gate.
(c) A NOR gate.

Transistör elektron akışını kontrol eden, yarı iletken teknolojisinde üretilen bir devre elemanıdır.

Mantık Kapıları için semboller ve işlevsel davranış



A	X
0	1
1	0

(a)

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

(b)

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

(c)

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

(d)

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

(e)

AND Kapısı: Girişlerden herhangi biri 0 ise çıkış 0 dır. Girişlerin tümü 1 ise çıkış 1 dir.

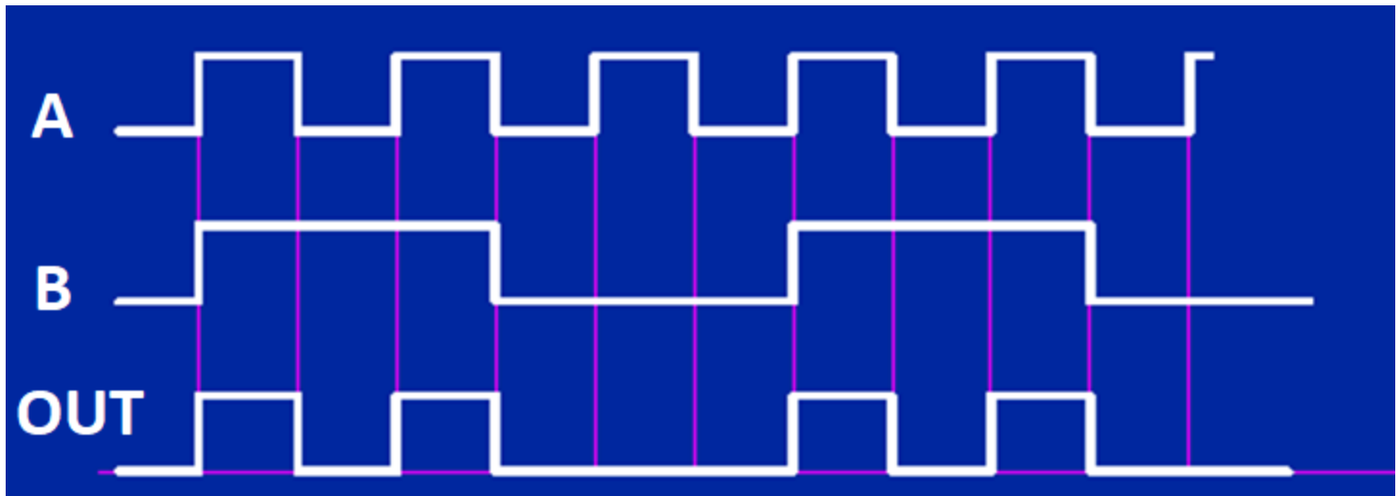
OR Kapısı: Girişlerden herhangi biri 1 ise çıkış 1 dır. Girişlerin tümü 0 ise çıkış 0 dir.

NOT Kapısı: girişin evriğini alır.

Mantık kapısı, bir Boole işlevini uygulayan idealleştirilmiş veya fiziksel bir devredir, yani bir veya daha fazla mantık girişinde mantıksal bir işlem gerçekleştirir ve tek bir mantık çıkışı üretir.

Giriş ve Çıkış mantıksal sevilere bakılarak mantık kapısını belirleme

- Giriş dalga formları A ve B bir mantık kapısının iki girişine uygulandığında çıkış dalga formu belli ise bu kapının türünü belirleyiniz. (AND Kapısı)



A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

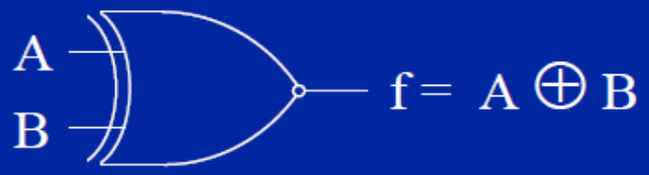
Logic Gates

Toplama

The EXCLUSIVE OR Truth Table

A	B	f = A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

The XOR Gate



Karşılaştırma

The EXCLUSIVE NOR Truth Table

A	B	f = A XOR B
0	0	1
0	1	0
1	0	0
1	1	1

EXCLUSIVE NOR Gate

This is called the equivalence gate

Karşılaştırma ve Aritmetik toplama işlemlerinde XOR kapıları kullanılır.
Girişlerin tümü birbirine eşit (0 ya da 1) çıkış sıfır ise XOR, çıkış 1 ise XNOR kapısıdır.

Logic Functions

Truth Table: 3-input majority function

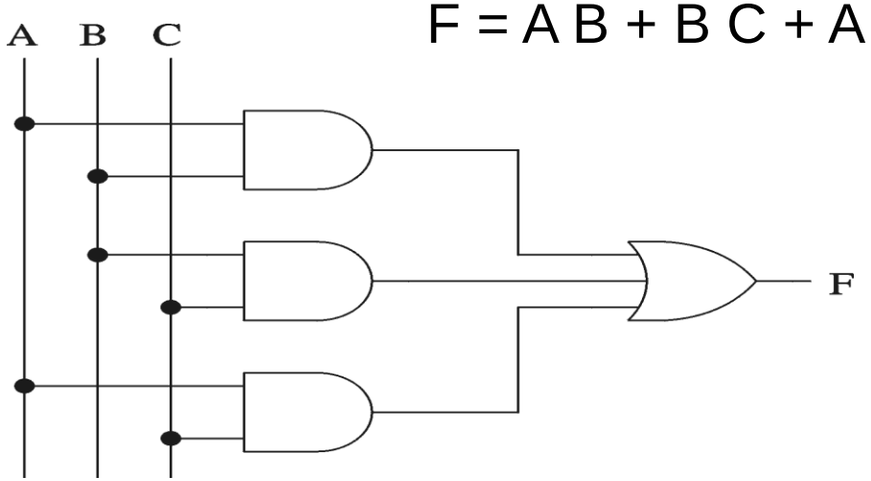
A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

- **Logical functions can be expressed in several ways:**
  - **Doğruluk Tablosu**
  - **Mantıksal İfade**
  - **Gösterim**

- **Logical expression form**

F=A'BC+AB'C+ABC'+ABC

F = A B + B C + A C



- Verilen denklemlerdeki değişkenlerin toplamı giriş sayısını verir.
- Mantık kapılarında mantıksal denklemin sonucu 1 ya da 0 dır.

Giriş sayısı, m olursa kaç farklı durum vardır? S=2^m durum vardır. 2 tabanı olmasının nedeni ikili sayı (binary) sisteminden kaynaklanmaktadır: 0 yada 1, bit

Boolean Algebra

Boolean Algebra

- Definition: a logic variable x can have only one of two possible values or states
  x = TRUE
  x = FALSE
- In binary notation, we can say
  x = TRUE = 1
  x = FALSE = 0
- This is called positive logic or high-true logic. We could also say
  x = TRUE = 0
  x = FALSE = 1
- This is called negative logic or low-true logic. Usually we use the positive logic convention.
- Electrically,
  – 1 is represented by a more positive voltage than zero and
  – 0 is represented by zero volts
  x = TRUE = 1 = 5 volts
  x = FALSE = 0 = 0 volts

Boole Cebirinin Kuralları ve Yasaları

Tek değişkenli temel kurallar:

- Boole Cebirinin Kurallarının ve Yasalarının her birinin bir kanıtı, değişkenin yalnızca iki bit(0/1) değere sahip olabileceği gerçeğinden yararlanılarak kolayca ispat edilebilir.
- Not: A=0 ya da 1 olur.
- A +A+A+A+A....+A+ 1 = 1 ; OR kapısında girişlerden herhangi biri 1 ise çıkış birdir. Diğer ispat etme yöntemi, 1 ve 0 değerleri verilerek doğruluk aranır.
- A+A+A+ ... + A=A (Neden? İki değişkenli 0 ya da 1 girişler mevcuttur)
- AAA ... A=A
  - If A = 0 then 0 + 1 = 1
  - If A = 1 then 1 + 1 = 1

$A + 0 = A$	$A \cdot 0 = 0$
$A + 1 = 1$	$A \cdot 1 = A$
$A + A = A$	$A \cdot A = A$
$A + \overline{A} = 1$	$A \cdot \overline{A} = 0$

It should be noted that $\overline{\overline{A}} = A$

Mantıksal matematiği ikili (0/1) sayı sisteminde hesaplama yapılır.
Soru: Boole cebrini kullanarak aşağıdaki işlemi yapınız. A=9, A+1=?
a) 0 b)1 c)10 d) 8 e)hiçbiri
Soru: Boole cebrinde A hangi değerleri alır? a) 0 b)1 c) 0/1 d) 0,1,2, …, 9 d)Her değeri e)hiçbir değeri
Soru: Boole cebrinde A=1 ise, A+A+A+A+A=? A)1 B)0 C)A D)5 D)5A E) Hiçbiri
Soru: Boole Cebrinde A*A*A*A=? A)A B) A^4

Boole Cebirinin Kuralları ve Yasaları

DeMorgan Yasaları, NAND ve NOR mantığı ile uğraşırken özellikle yararlıdır.

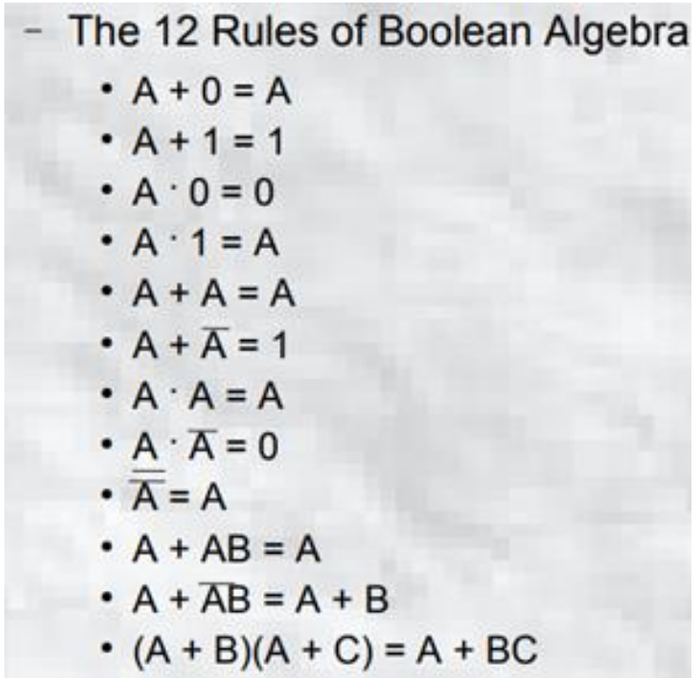
$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

$$\overline{A \cdot B} = \overline{A} + \overline{B}$$

Boole Cebri Teoremleri

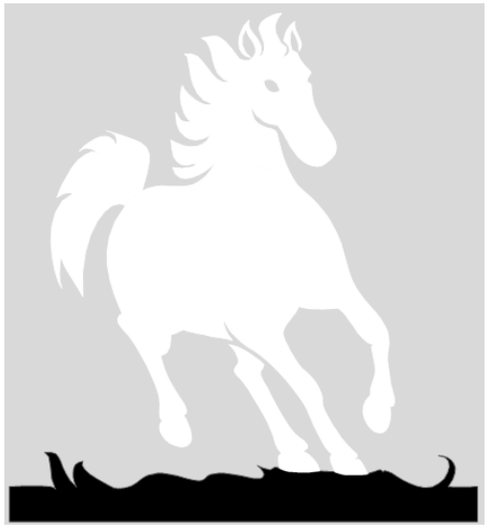
1. a) $a+b=b+a$ Değişme Özelliği
   $b)$ $a \cdot b=b \cdot a$
2. a) $a+b+c=a+(b+c)$ Birleşme Özelliği
   $b)$ $a \cdot b \cdot c=a \cdot (b \cdot c)$
3. a) $a+b \cdot c=(a+b) \cdot (a+c)$ Dağılma Özelliği
   $b)$ $a \cdot (b+c) = a \cdot b + a \cdot c$
4. a) $a+a=a$ Değişkende Fazlalık Özelliği
   $b)$ $a \cdot a=a$
5. a) $a+a.b=a$ Yutma Özelliği
   $b)$ $a \cdot (a+b)=a$
6. a) (a)^n $=a$ işlemde Fazlalık Özelliği
   $b)$ $(a$ x n$) =a$
7. a) $\overline{(a+b)}=\bar{a} \cdot \bar{b}$ De Morgan Kuralı
   $b)$ $\overline{(a \cdot b)}=\bar{a} + \bar{b}$

8. a) $a+\bar{a}$ =1 Sabit Özelliği
   $b)$ $a \cdot \bar{a}$ =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) $(a+b)\cdot b=b$
    $b)$ $a \cdot b + b=b$

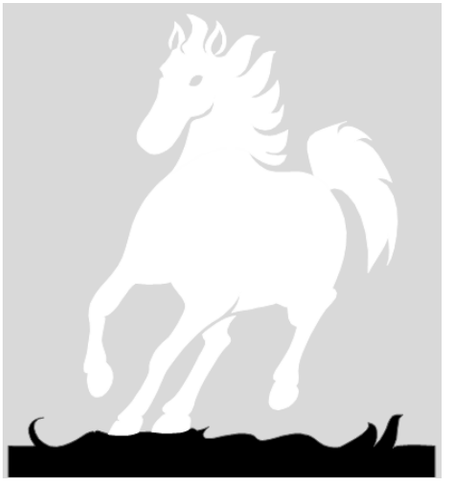


Mantıksal işlemlerin sonucu daima 1 ya da 0 dır.

- a + a + a + a + ... + a =a
- a*a*a* .... *a =a
- 1+a+b+c+ ... + z=1
- ab’c + ab’c=ab’c; benzer ifadelerin toplamı her zaman benzer bir tanesine eşittir.

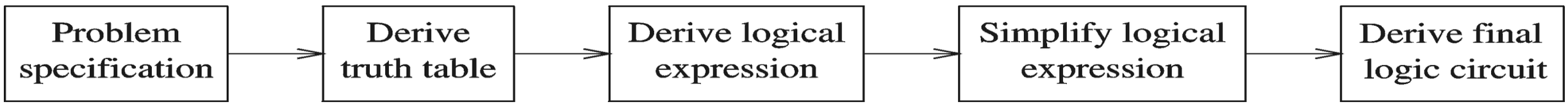


Simplify the function



Logic Circuit Design Process

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



Rules and Laws of Boolean Algebra

- Some useful theorems

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

A+AB=A(1+B)=A; 1+B=1
A+A'B=(A+A')(A+B)=A+B; A+A'=1;
A+BC=(A+B)(A+C)
AB+AB'=A(B+B')=A; B+B'=1
A(A+B)=AA+AB=A+AB=A(1+B)=A; AA=A
A(A'+B)=AA'+AB=AB; AA'=0
(A+B)(A+B')=AA+AB'+AB+BB'=A+A(B+B')+0=A+A=A

- **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

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

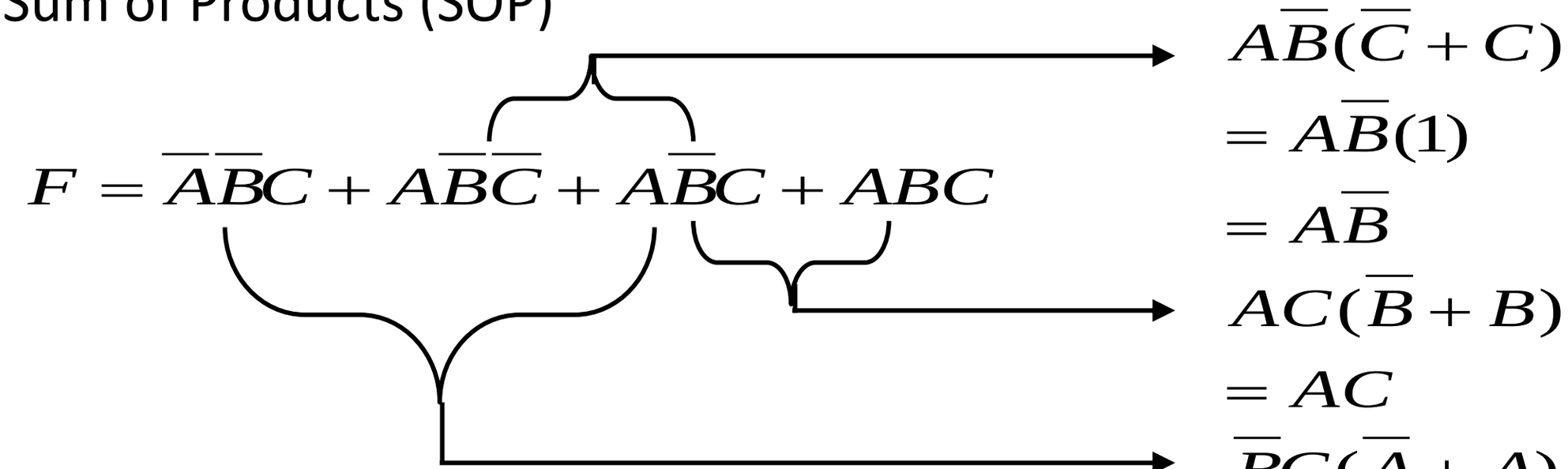
$= (A + B + C)(A + \overline{B} + C)$

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

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

Standard Forms

- Sum of Products (SOP)



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

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

$$AC(\overline{B} + B)$$
$$= AC$$

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

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

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

- İndirgeme yapılırken: bir ifadeden çok sayıda ilave edilip çıkarılabilir. (Çünkü: abd + abd + abd + ...=abd)
- İndirgemede ifadedeki değişken sayısının bir eksiği ifadenin bezerliği aranır. (bu örnekte ifade 3 değişken oluşmaktadır. O halde 2 benzer aranır.)

Simplification of Expressions using Boolean Algebra

AB + A(B+C) + B(B+C)	
AB + AB + AC + BB + BC	{distribution}
AB + AC + BB + BC	{X + X = X }
AB + AC + B + BC	{X . X = X }
AB + B.1 + BC + AC	{X . 1 = X }
B(A+1+C) + AC	{distribution}
B.1 + AC	{1 + X = 1 }
B + AC	{X . 1 = X }

Örnek: 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

Örnek: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)

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

Örnek: Boolean Algebra

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

↓ 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

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

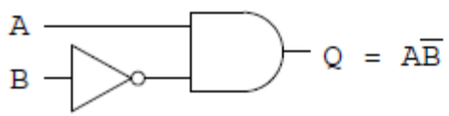
↓ Applying identity $AA = A$ to left term; applying identity $A\overline{A} = 0$ to B and $\overline{B}$ in right term

$A\overline{B} + 0$

↓ 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

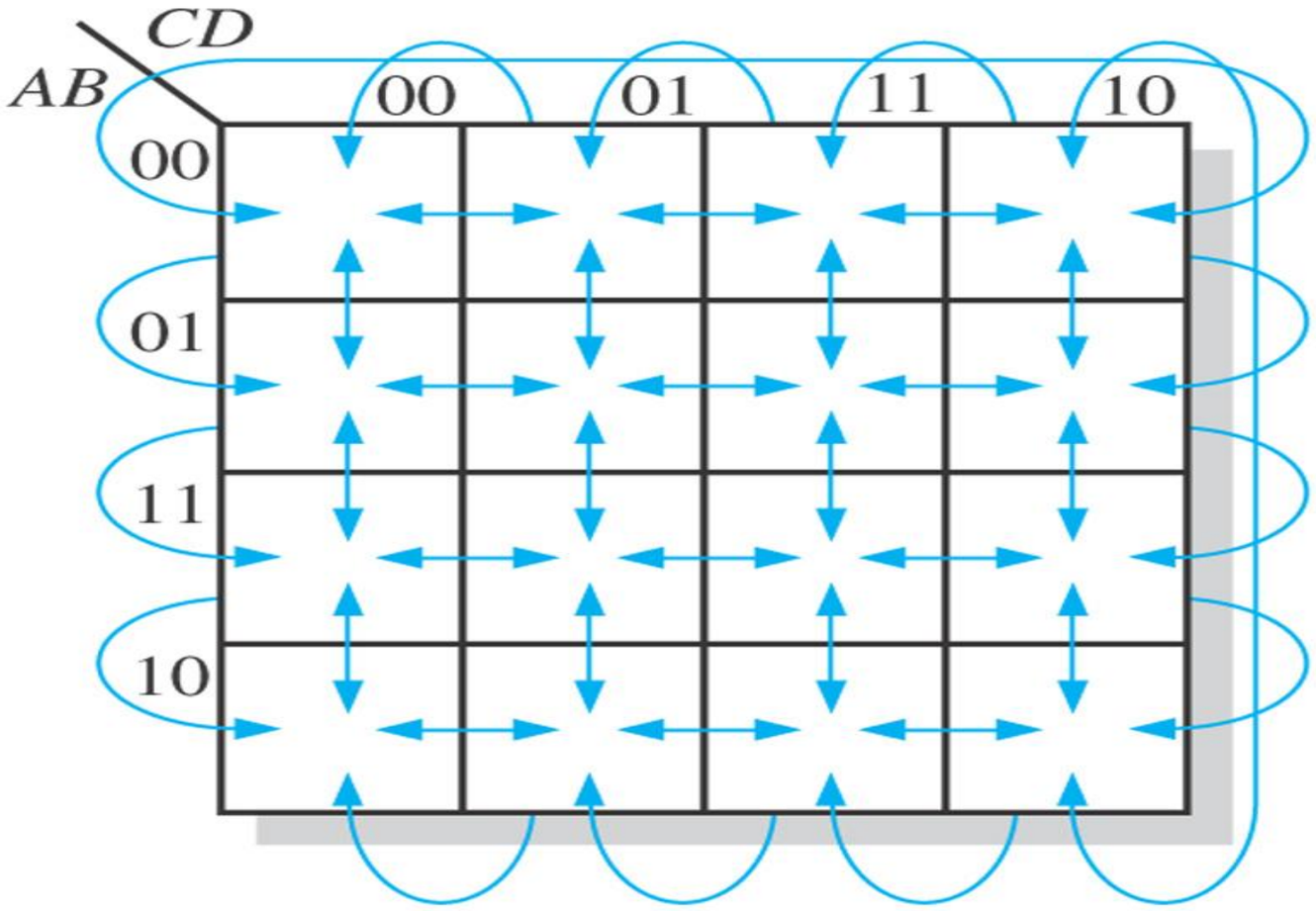
$AB + BC + AC$

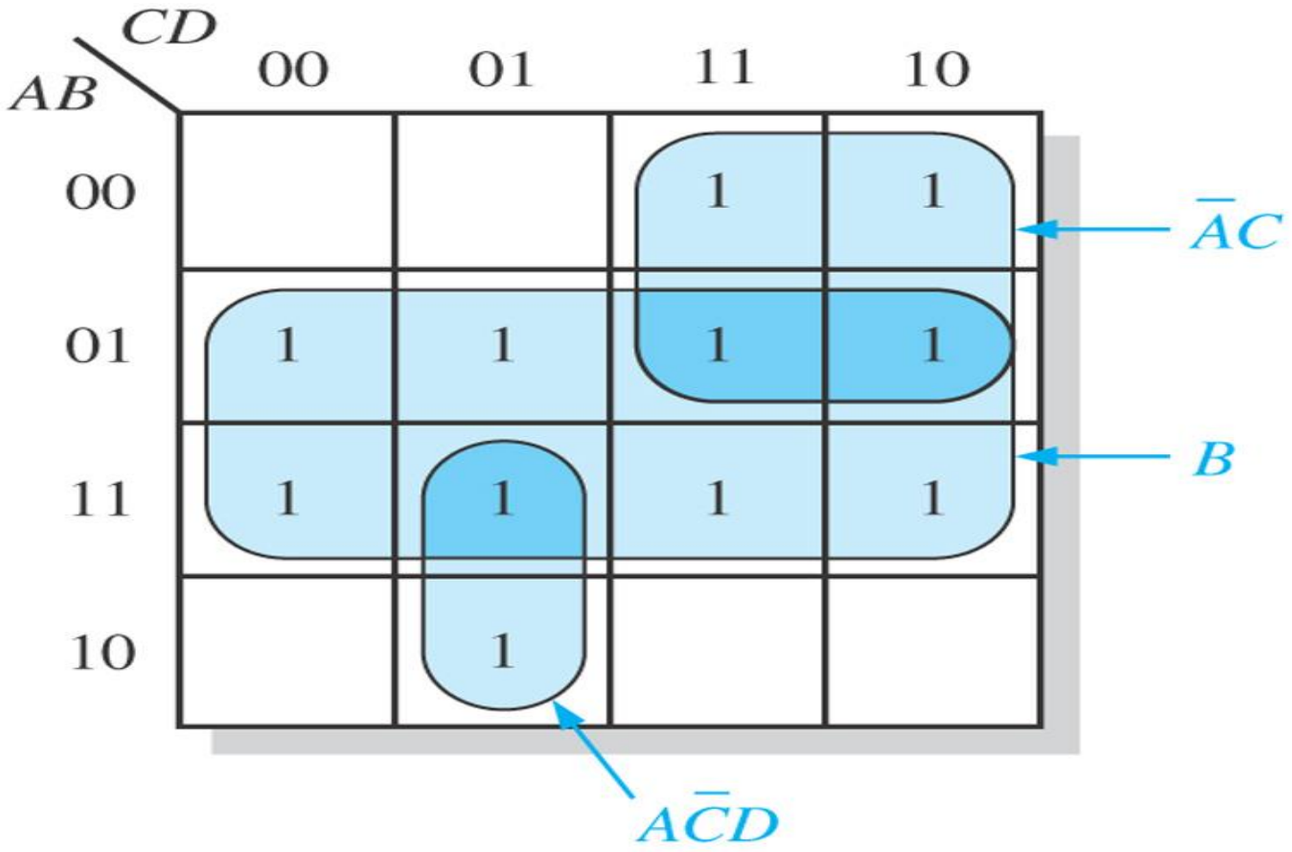
The Karnaugh Map

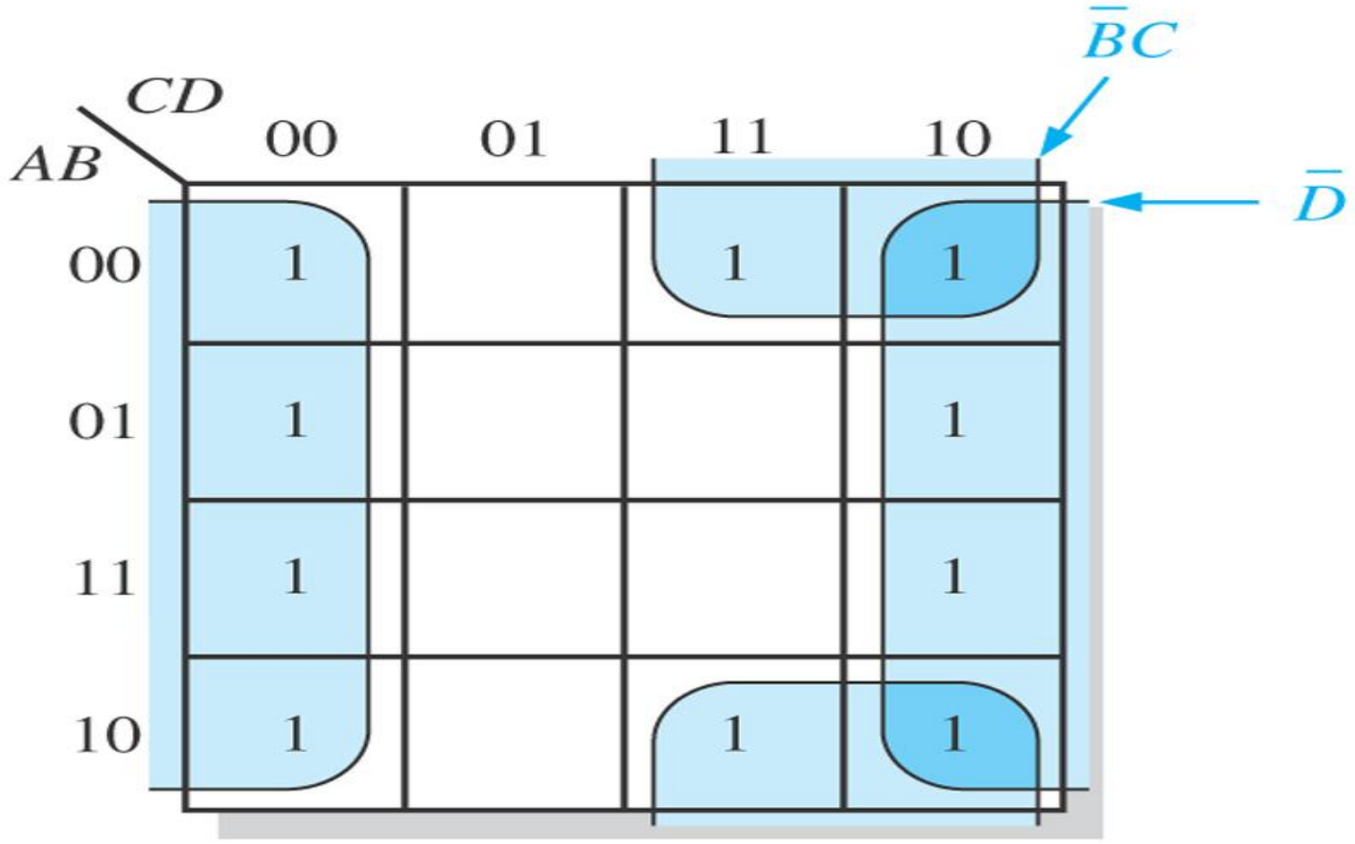
Lojik fonksiyonların sadeleştirilmesi

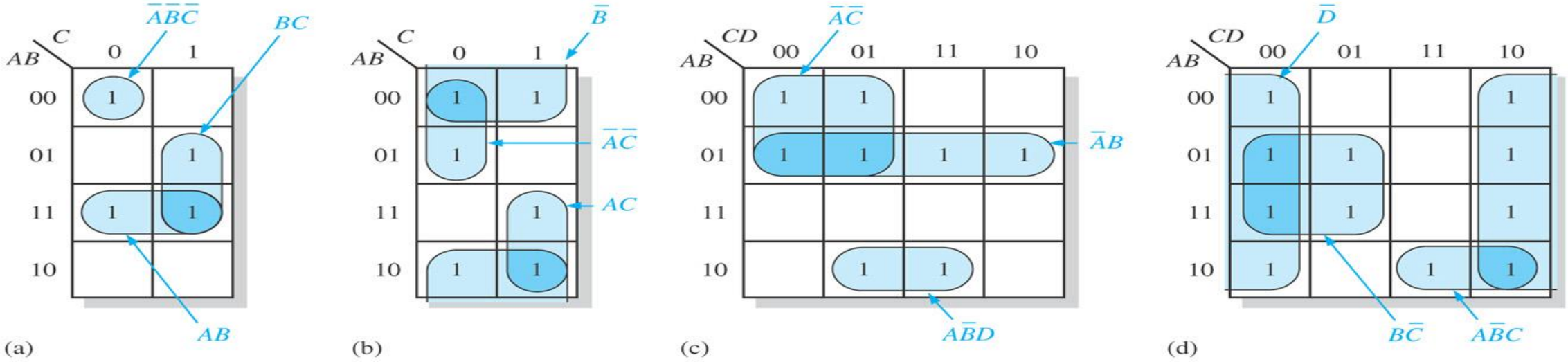
- Lojik fonksiyonların sadeleştirilmesinde en çok kullanılan iki yöntem şunlardır:
- 1. Karnaugh Diyagramı Yöntemi
- 2. Quine-McCluskey Tablo Yöntemi

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

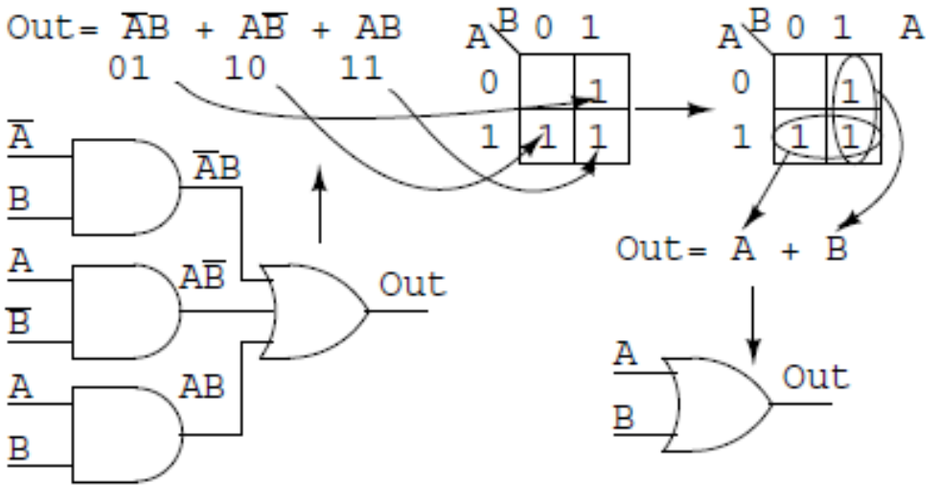
Simplify the following Boolean expressions, using three-variable maps:

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

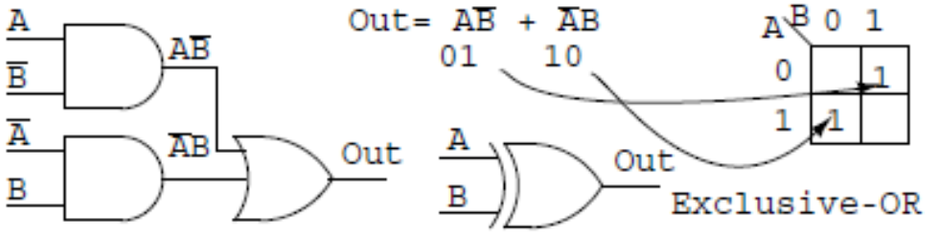
(b)* $x'y' + yz + x'yz'$

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

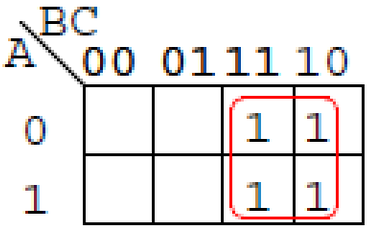
(d) $F(x, y, z) = x'yz + xy'z' + xy'z$



Simplify the logic diagram below.



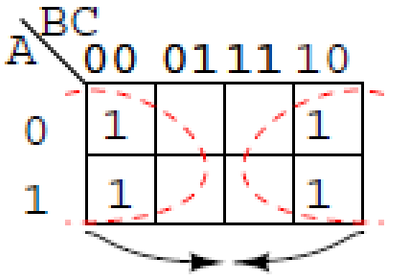
Out= $\overline{A}BC + \overline{A}B\overline{C} + ABC + AB\overline{C}$

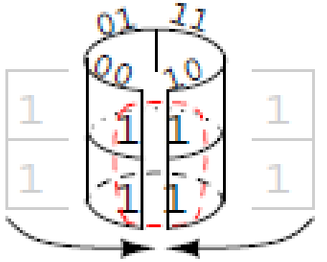


Out= B .

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

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

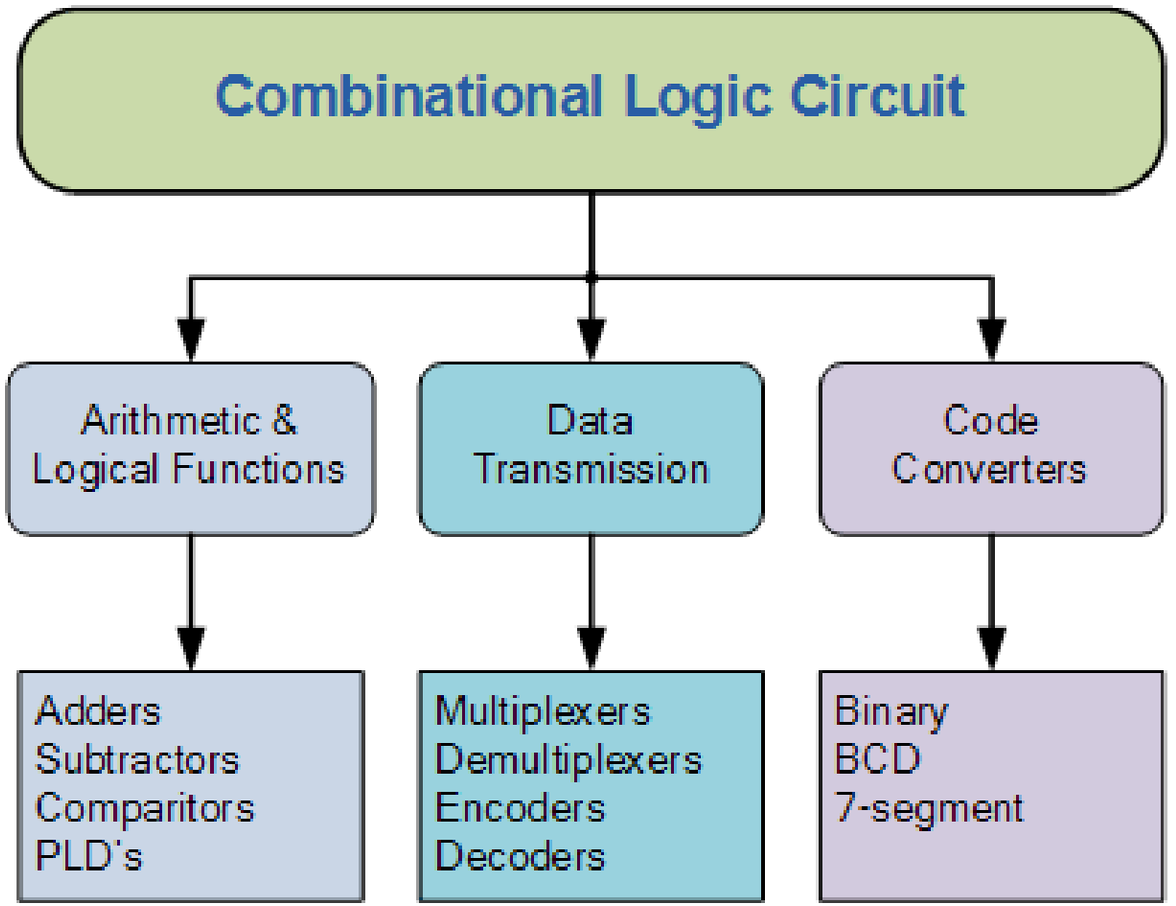




Out= $\overline{C}$.

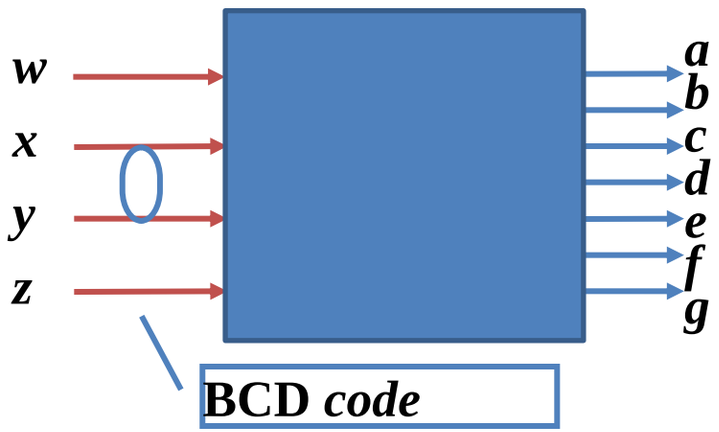
Kombinasyon Devreleri

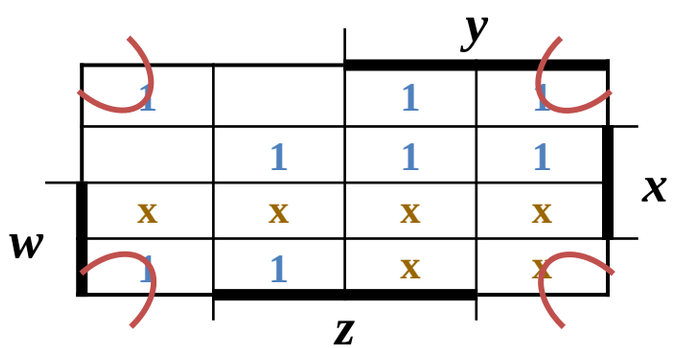
Classification of Combinational Logic



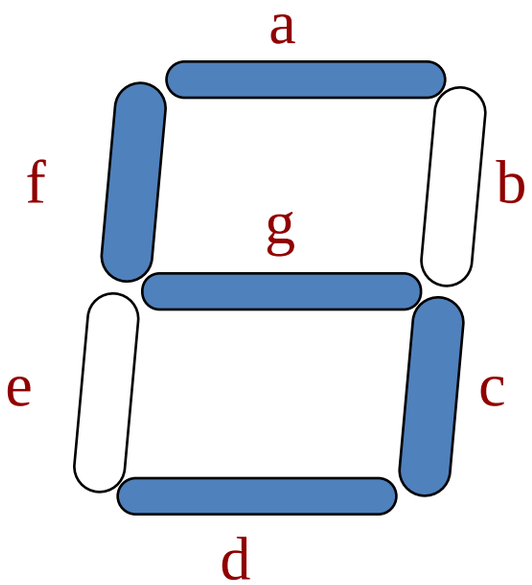
Seven-Segment Decoder

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



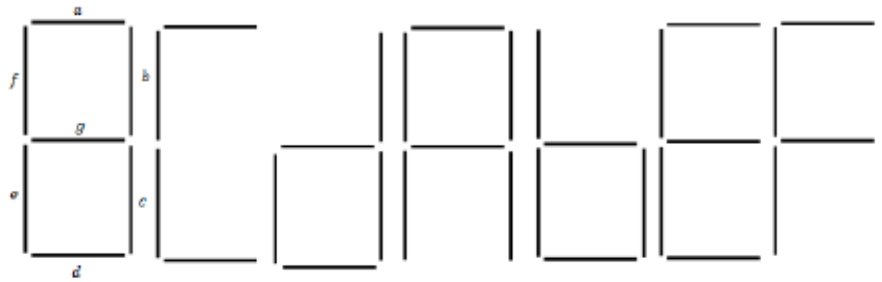


$a = w + y + xz + x'z'$

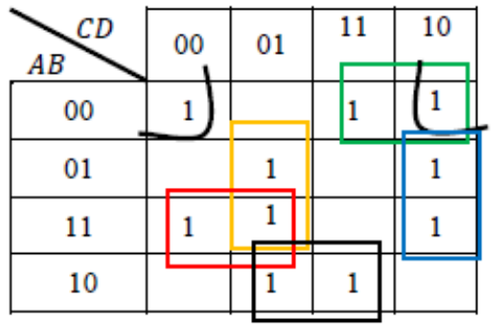


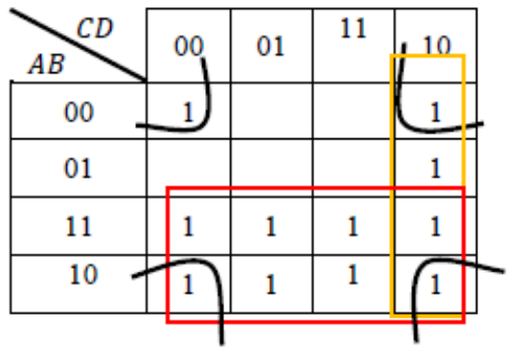
$b = \ldots$
$c = \ldots$
$d = \ldots$

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





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

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

Half Adder

Circuits for Binary Addition

With twos complement numbers, addition is sufficient

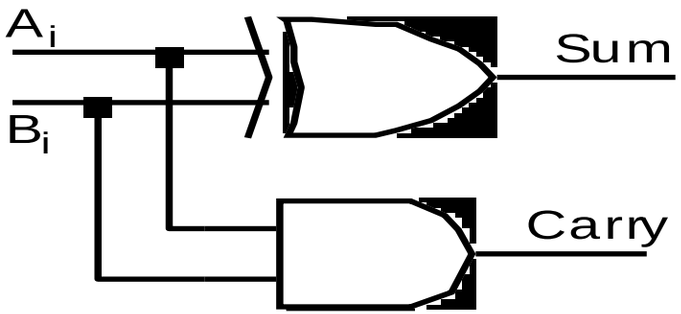
Ai	Bi	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

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

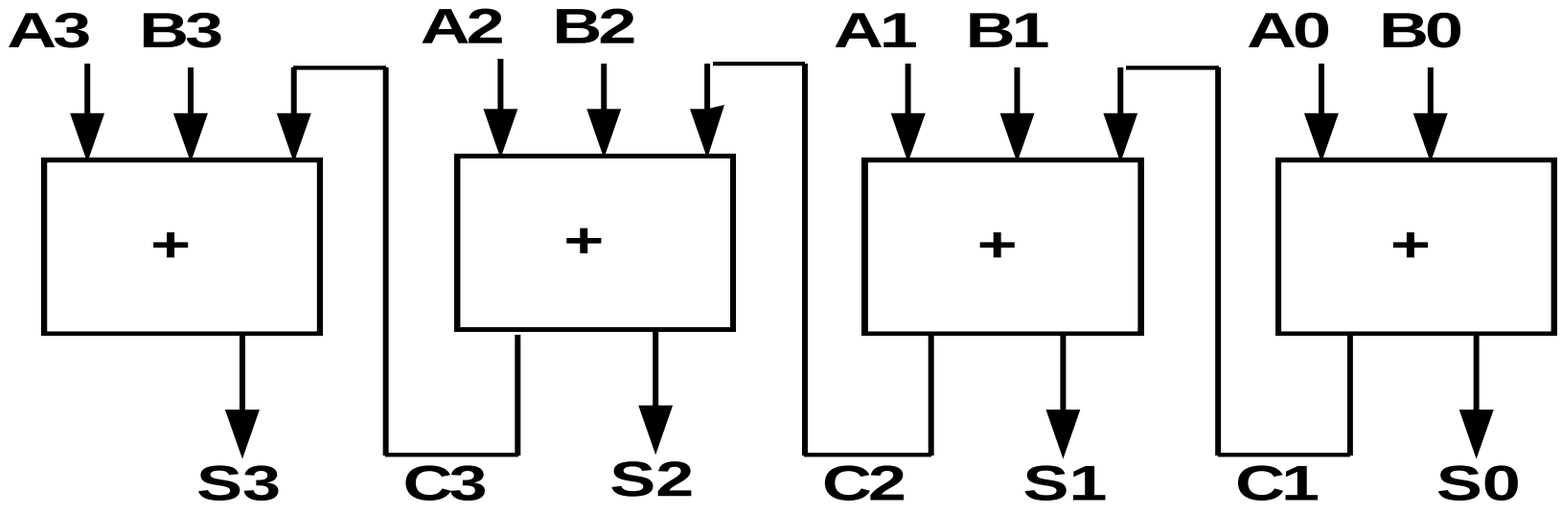
$$Carry = Ai\,Bi$$



Half-adder Schematic

Full Adder

Cascaded Multi-bit Adder

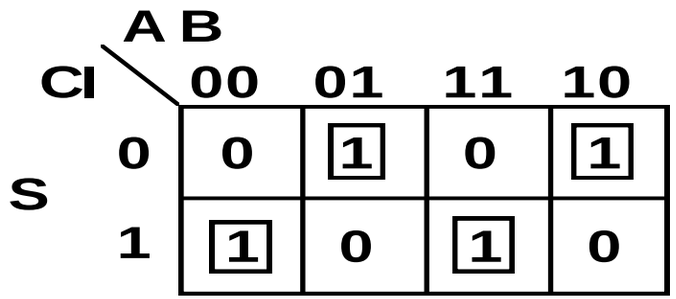


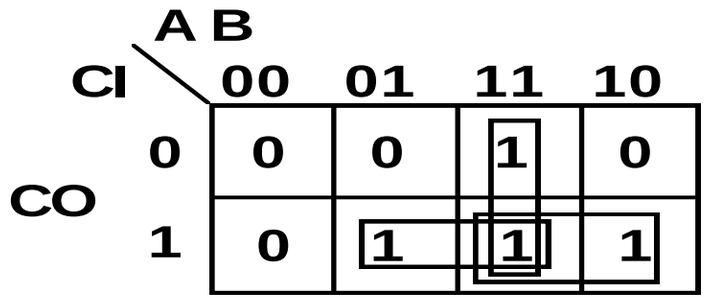
usually interested in adding more than two bits

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



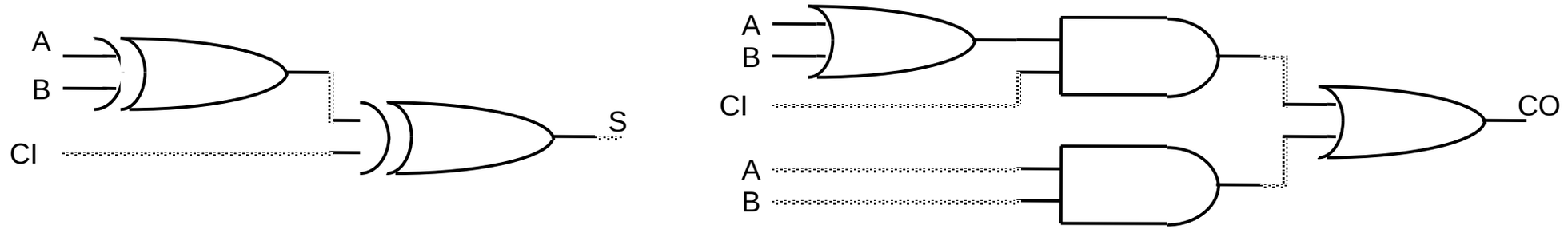


S = CI xor A xor B

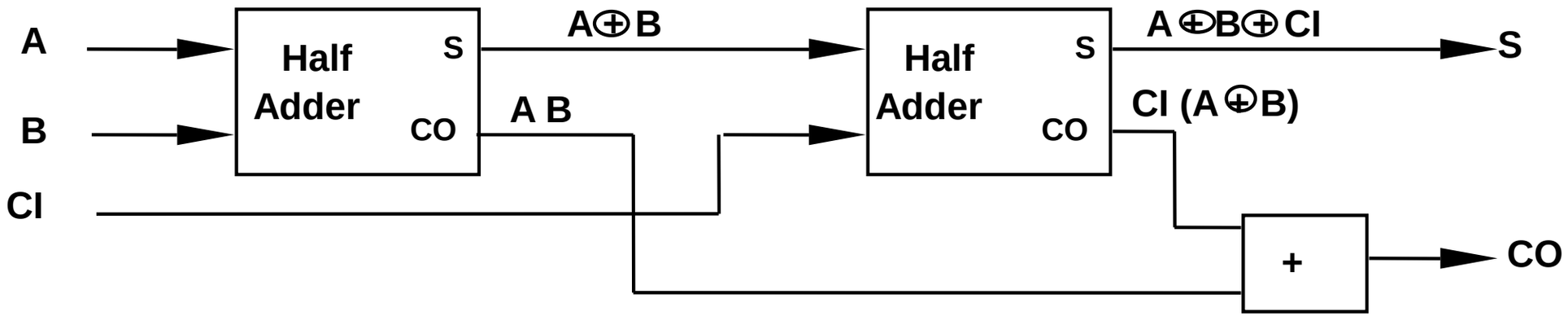
CO = B CI + A CI + A B = CI (A + B) + A B

Full Adder Circuit

Standard Approach: 6 Gates

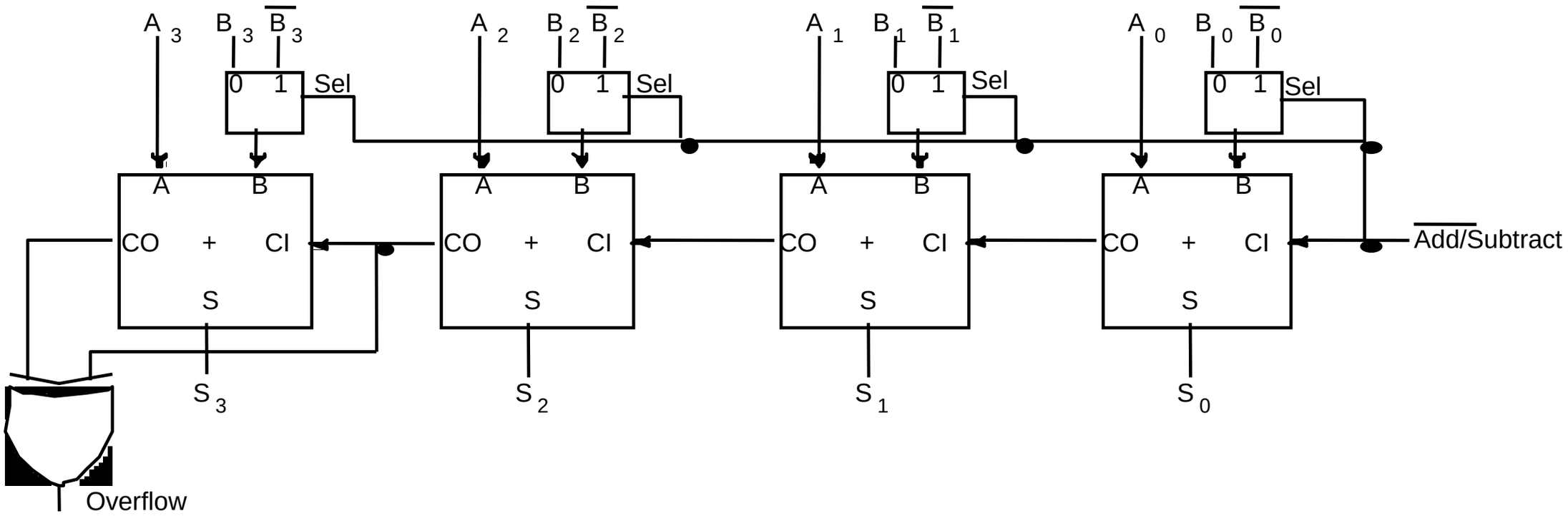


Alternative Implementation: 5 Gates



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

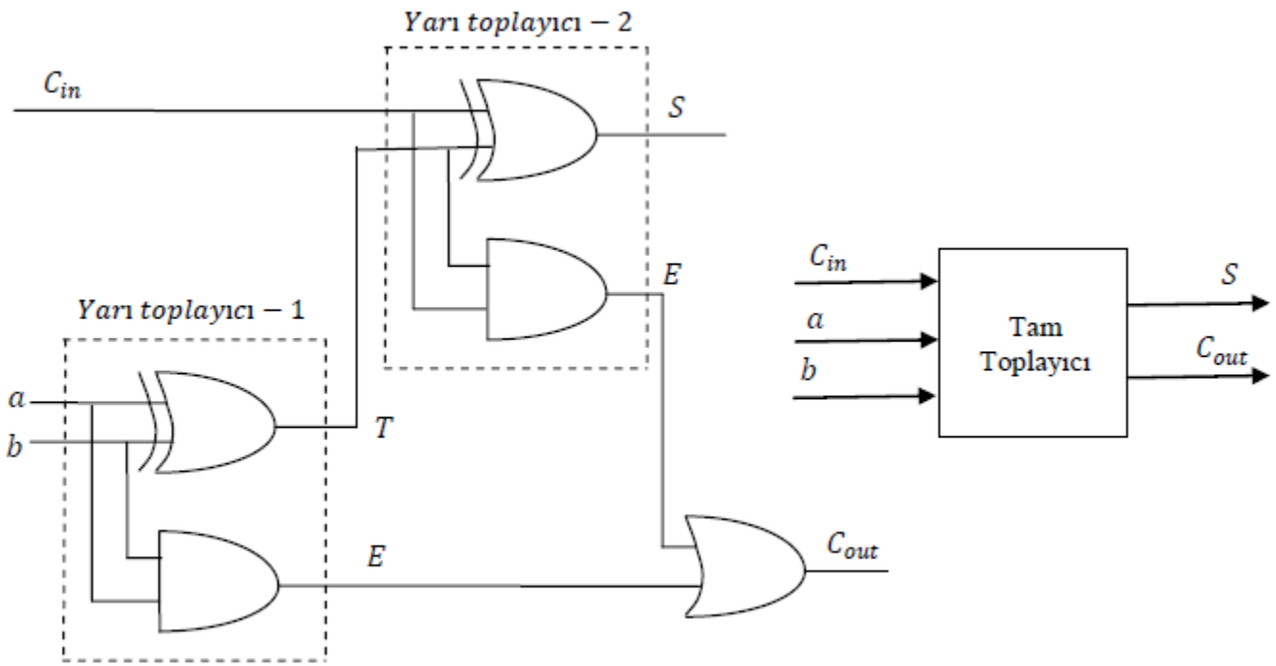
Adder/Subtractor



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

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

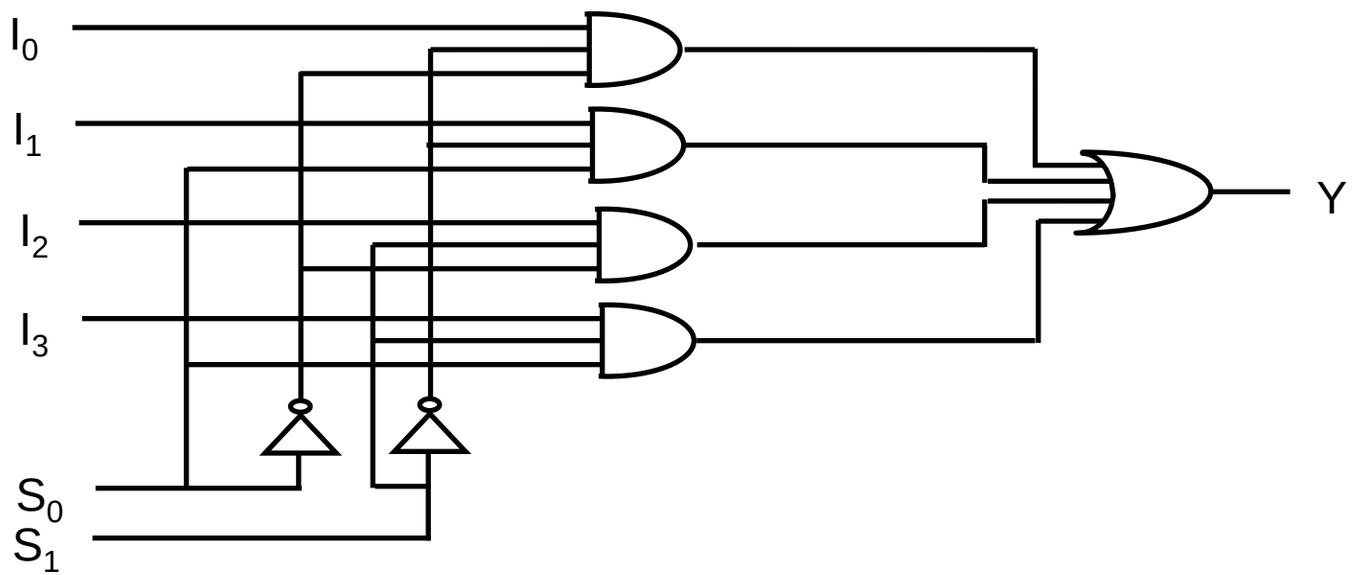
$$T = C_{in}\bar{a}\bar{b} + \overline{C_{in}}\bar{a}b + C_{in}ab + \overline{C_{in}}a\bar{b} = \overline{C_{in}}(\bar{a}b + a\bar{b}) + C_{in}(ab + \bar{a}\bar{b}) \Longrightarrow T = a \oplus b \oplus C_{in}$$

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

Multiplexer

4-to-1 Multiplexer

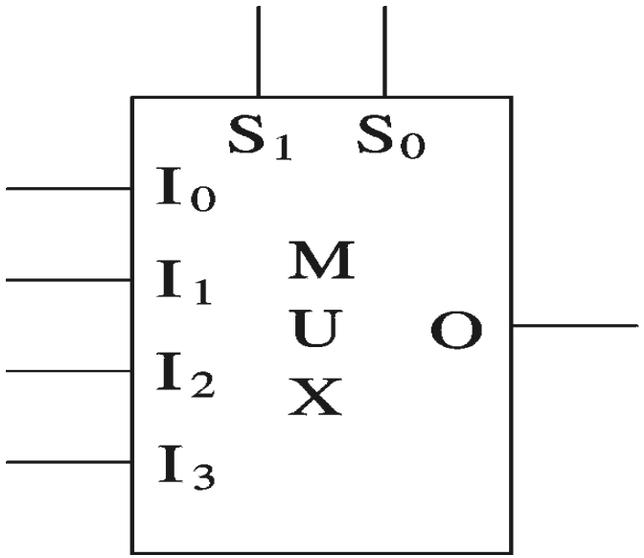
Select		Output
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



Multiplexers

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

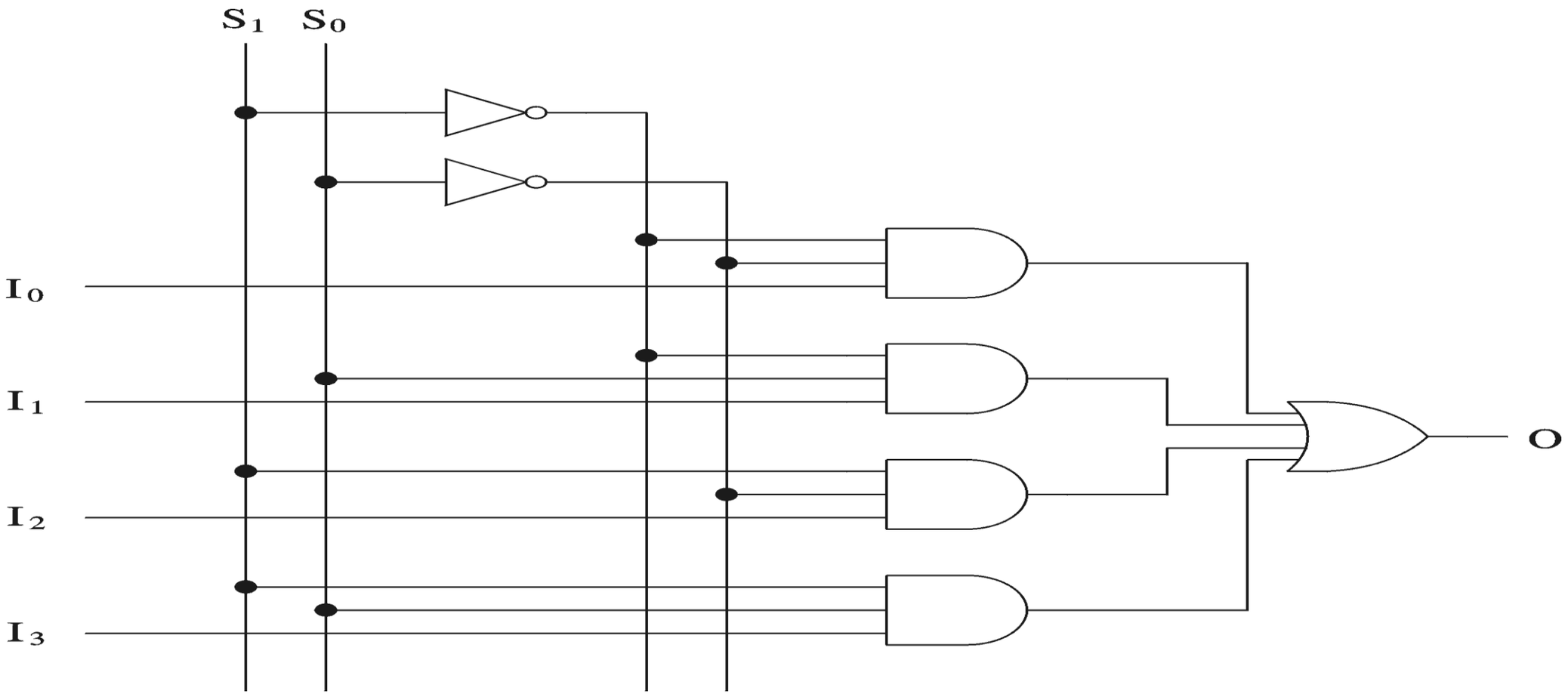
4-data input MUX



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

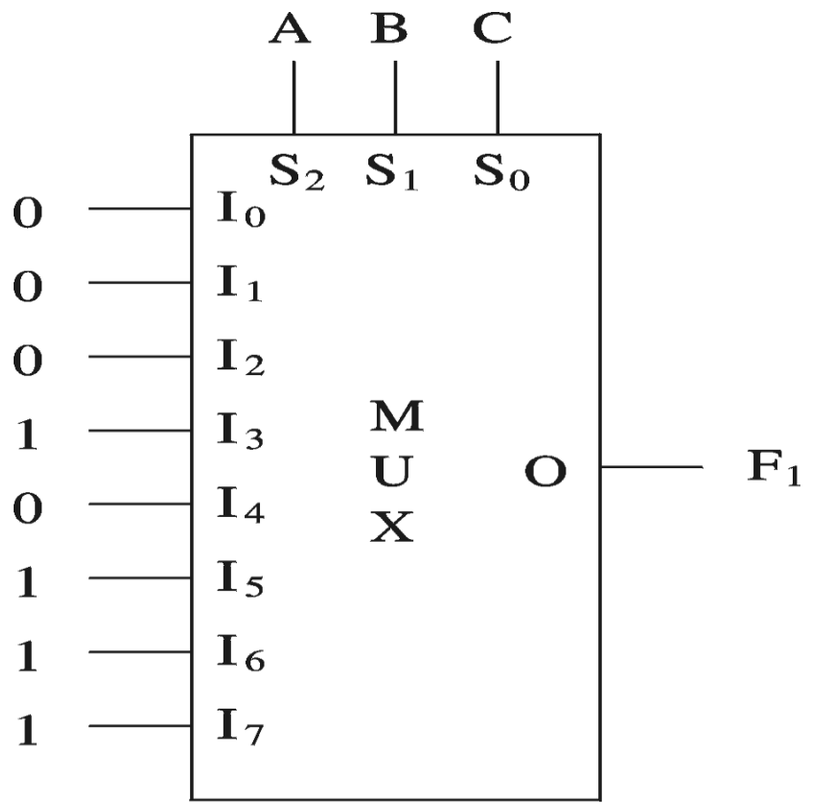
Multiplexers

4-data input MUX implementation

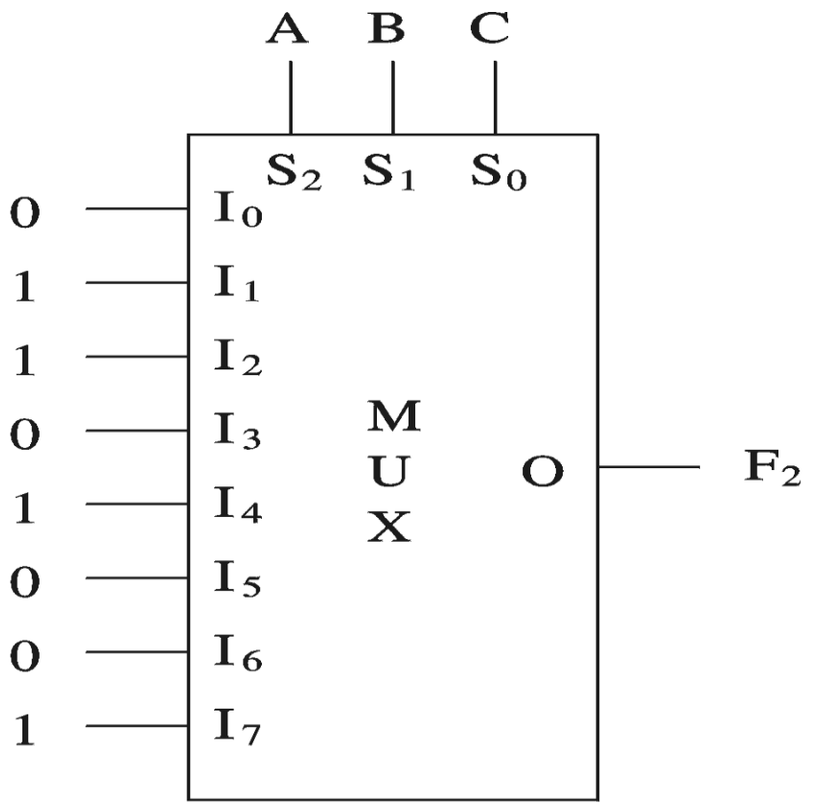


Multiplexers

MUX implementations



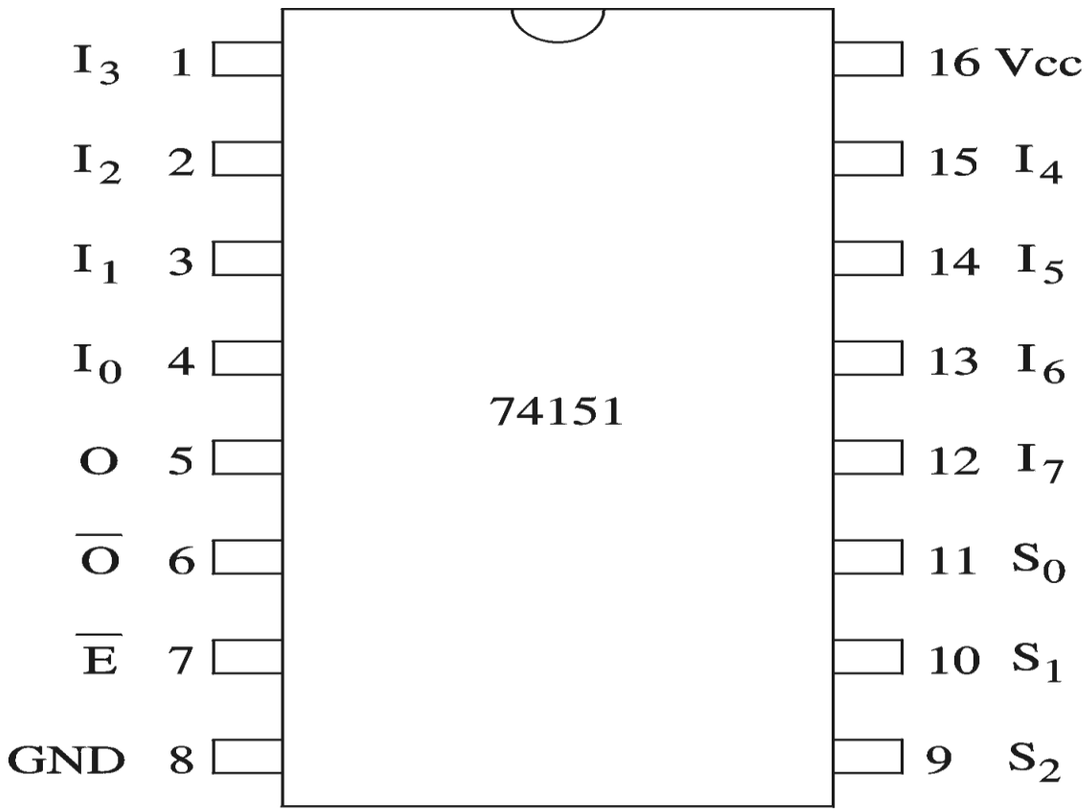
Majority function



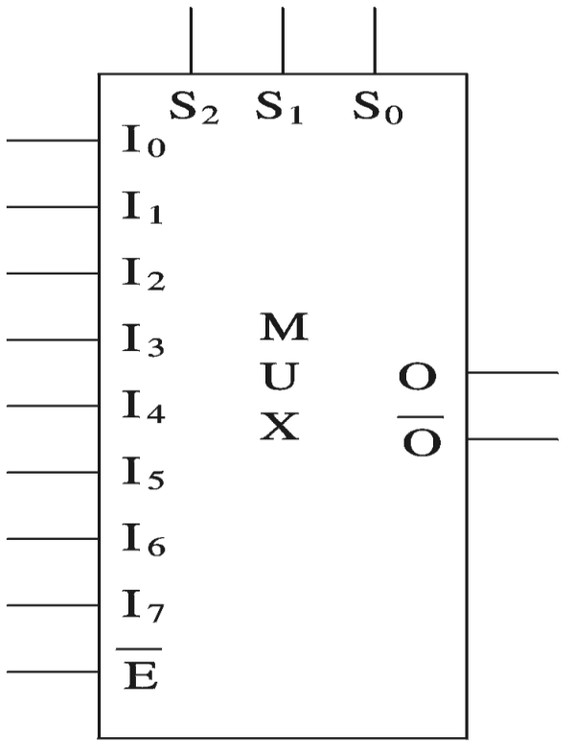
Even-parity function

Multiplexers

Example chip: 8-to-1 MUX



(a) Connection diagram



(b) Logic symbol

Multiplexers

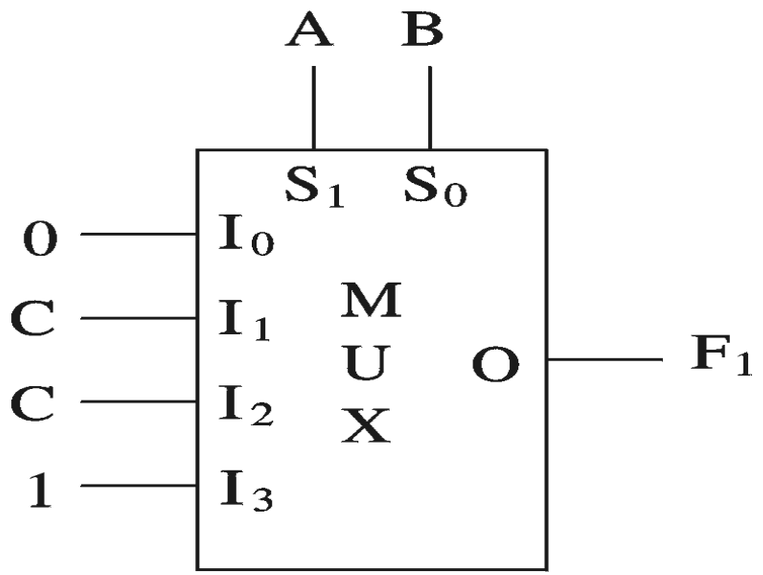
Efficient implementation: Majority function

Original truth table

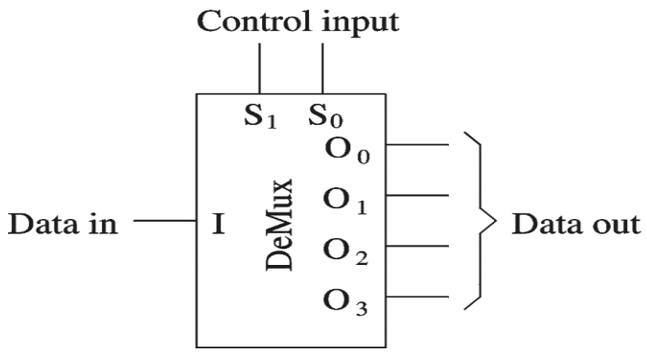
A	B	C	F_1
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

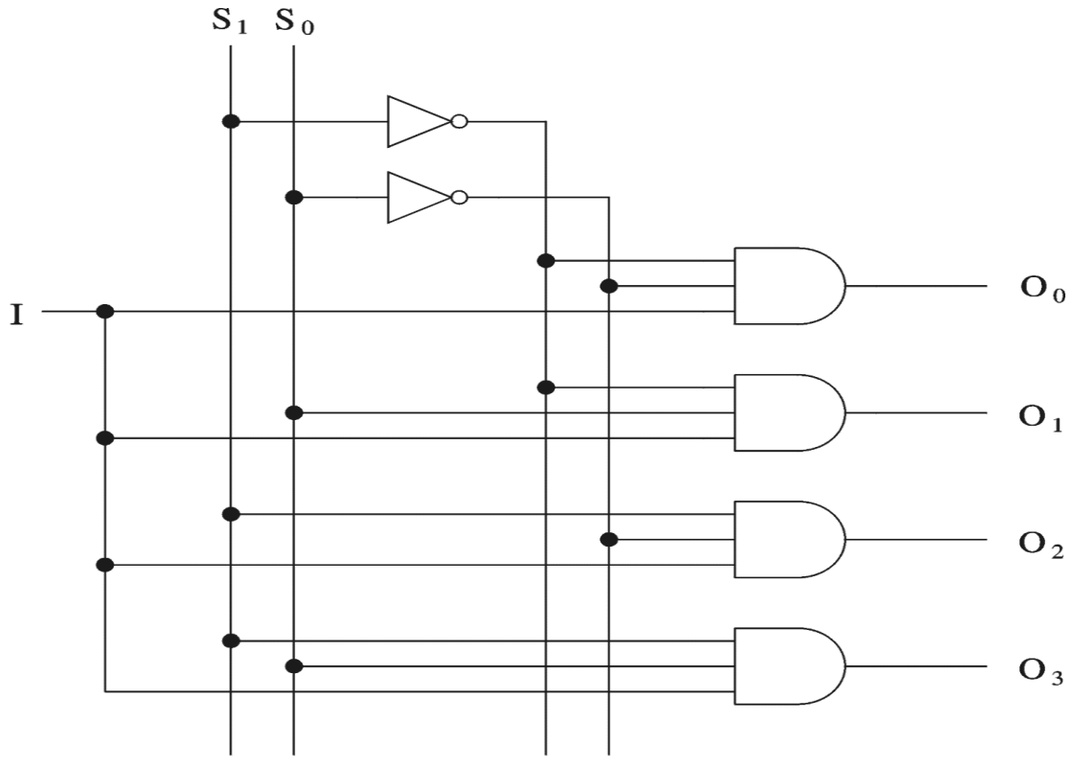
New truth table

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



Demultiplexer (DeMUX)

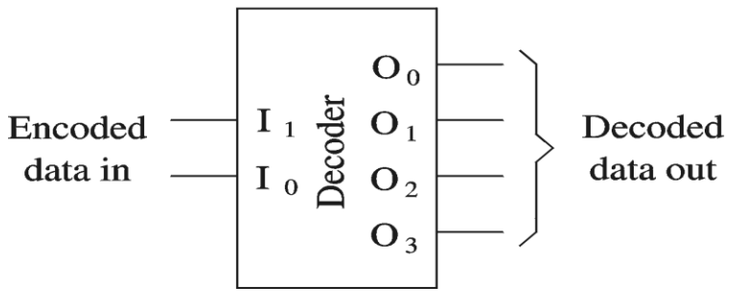


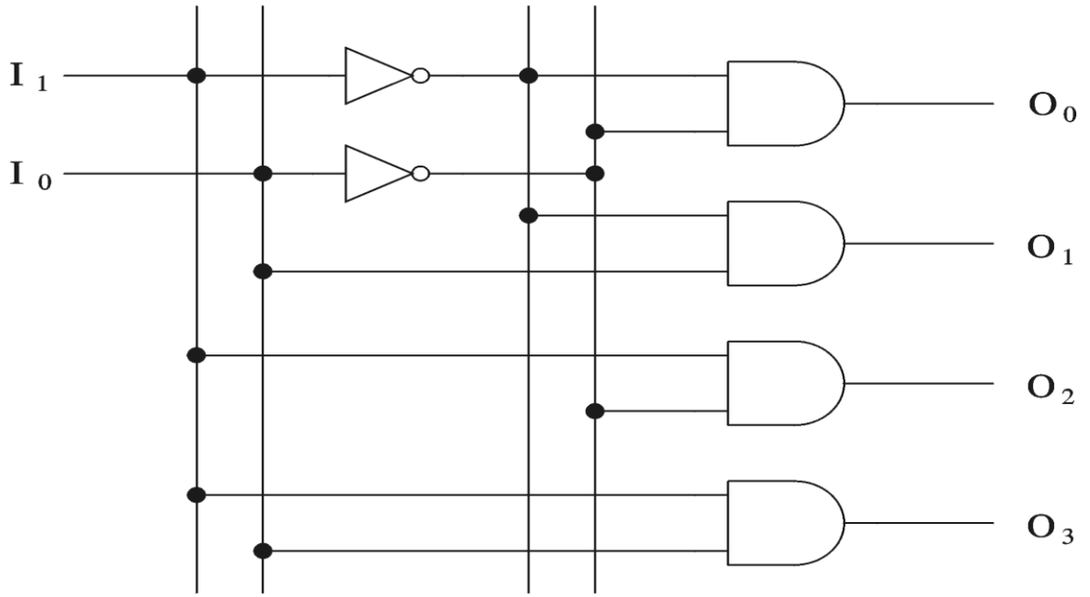


Decoders

- 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



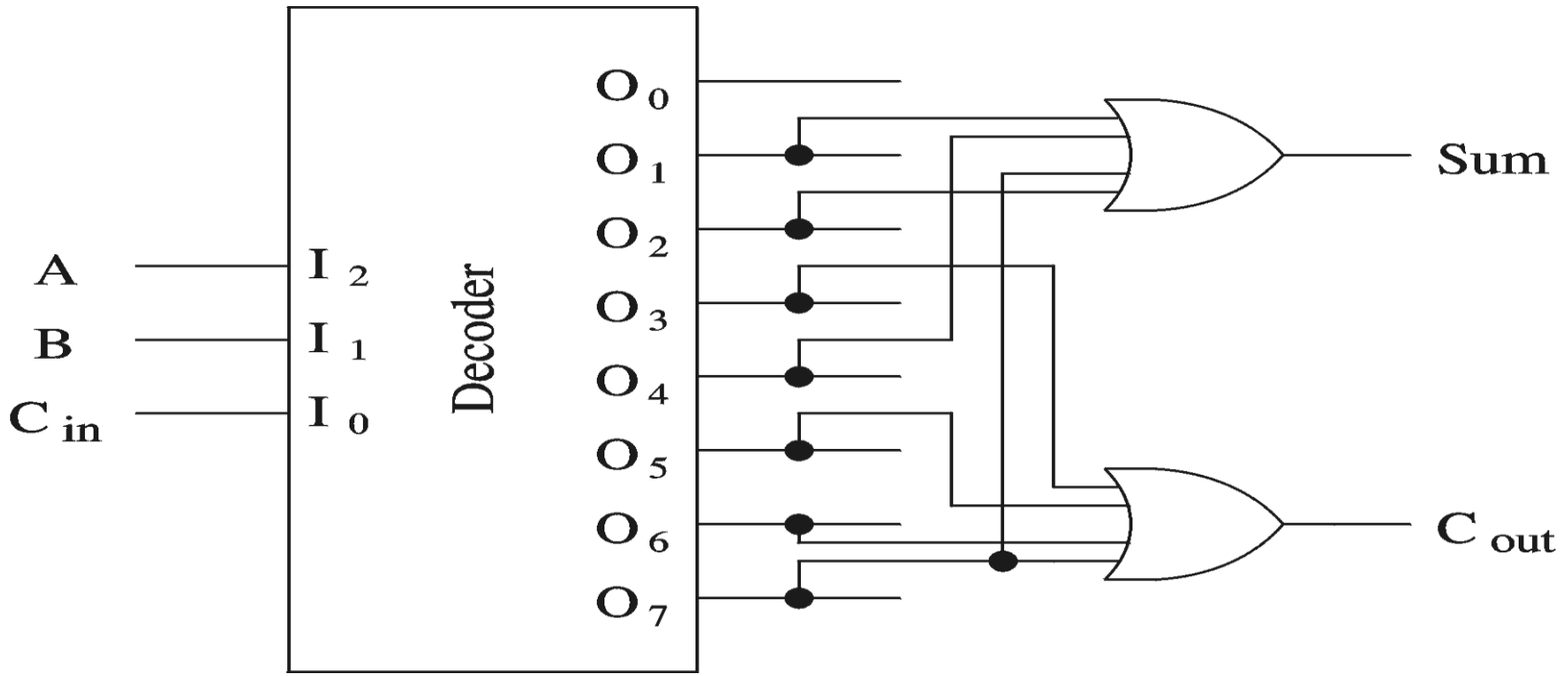


Decoders

Logic function implementation

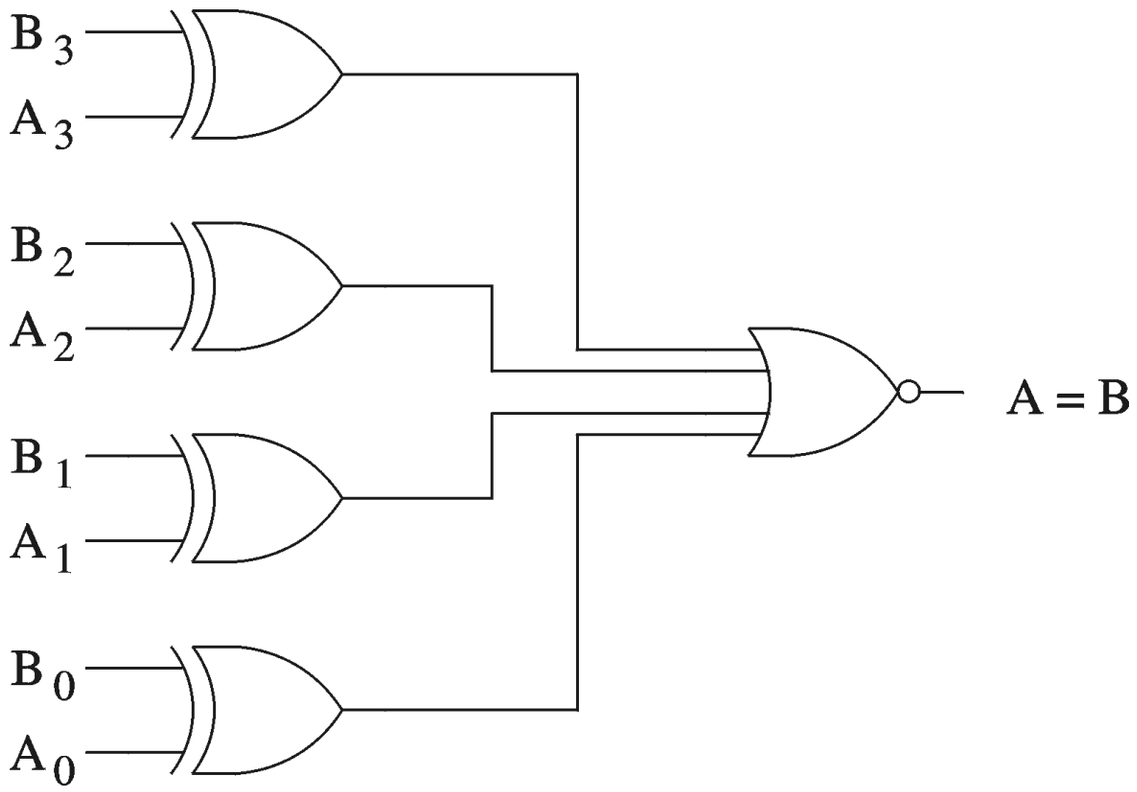
A	B	C_{in}	Sum	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

(Full Adder)

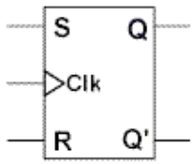
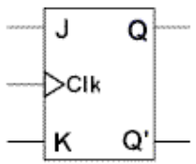
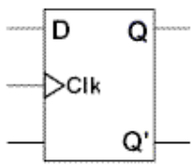
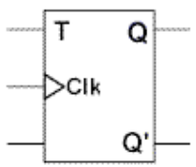


Comparator

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



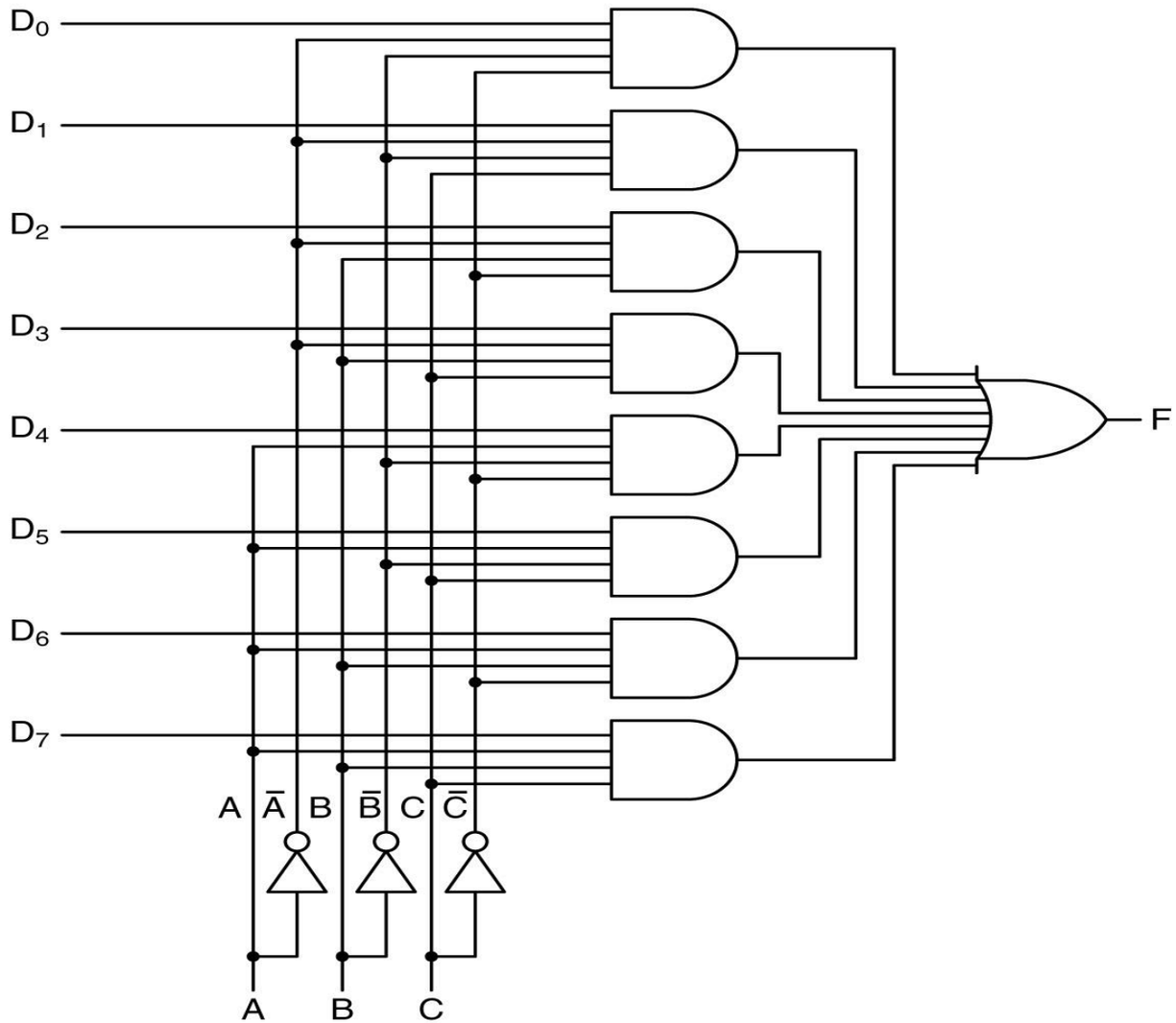
Sequential (Sıralı Mantık) Logic

FLIP-FLOP NAME	FLIP-FLOP SYMBOL	CHARACTERISTIC EQUATION	EXCITATION TABLE
SR		$Q_{(next)} = S + R'Q$ $SR = 0$	**Q** | **$Q_{(next)}$** | **S** | **R** 0 | 0 | 0 | X 0 | 1 | 1 | 0 1 | 0 | 0 | 1 1 | 1 | X | 0
JK		$Q_{(next)} = JQ' + K'Q$	**Q** | **$Q_{(next)}$** | **J** | **K** 0 | 0 | 0 | X 0 | 1 | 1 | X 1 | 0 | X | 1 1 | 1 | X | 0
D		$Q_{(next)} = D$	**Q** | **$Q_{(next)}$** | **D** 0 | 0 | 0 0 | 1 | 1 1 | 0 | 0 1 | 1 | 1
T		$Q_{(next)} = TQ' + T'Q$	**Q** | **$Q_{(next)}$** | **T** 0 | 0 | 0 0 | 1 | 1 1 | 0 | 1 1 | 1 | 0

Address Decoding Circuit

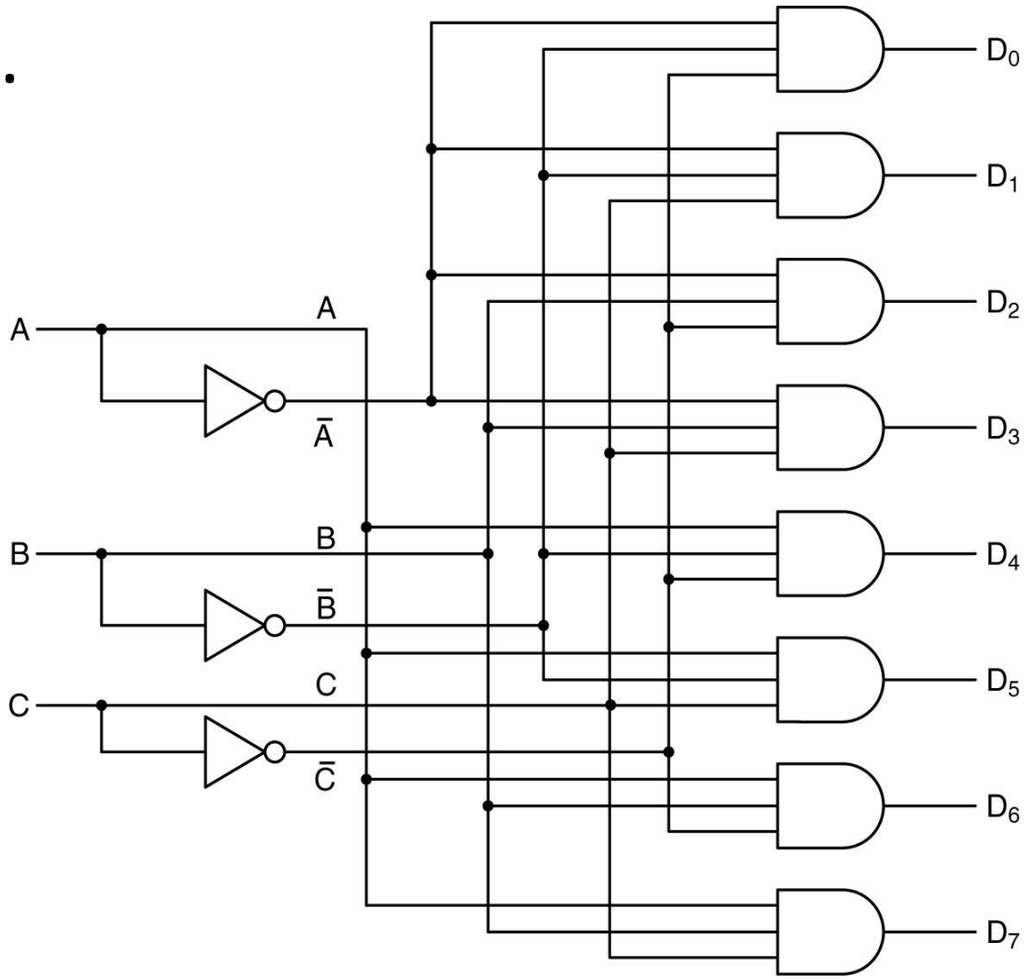
Multiplexers

An eight-input multiplexer circuit.



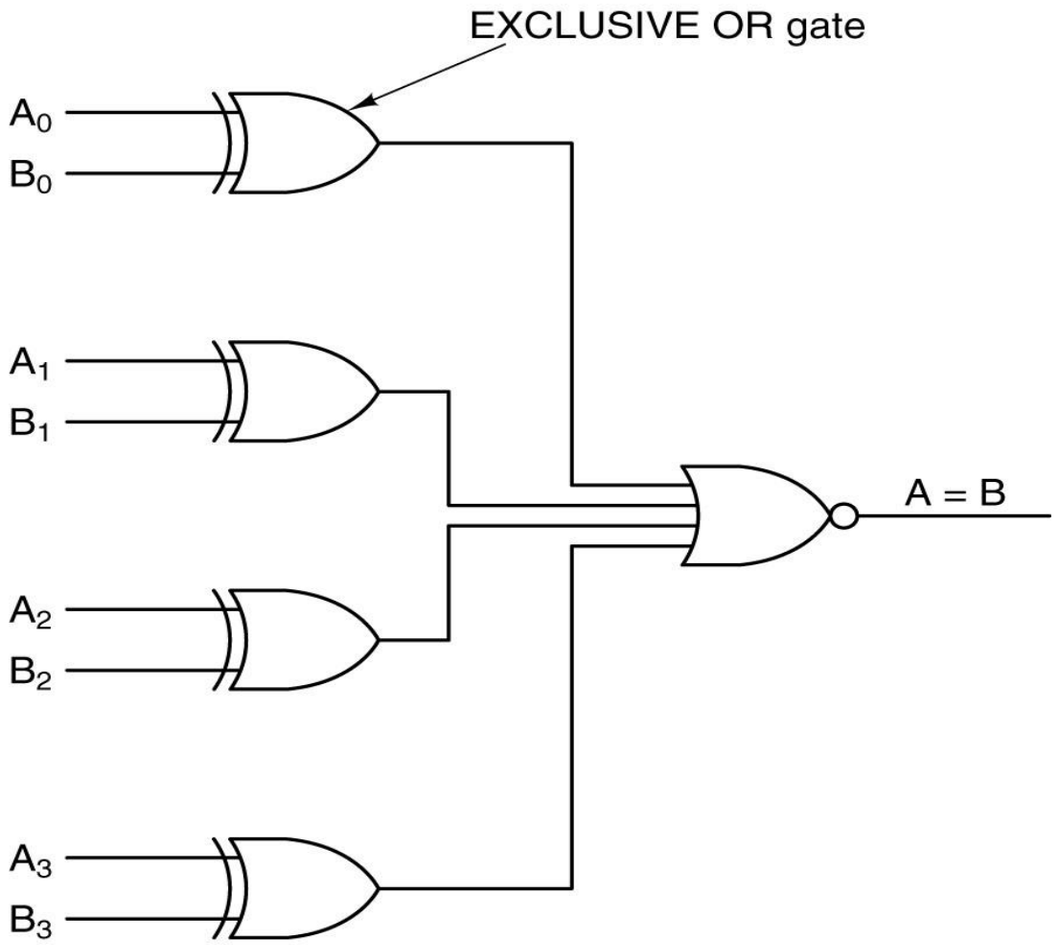
Decoders

A 3-to-8 decoder circuit.



Comparators

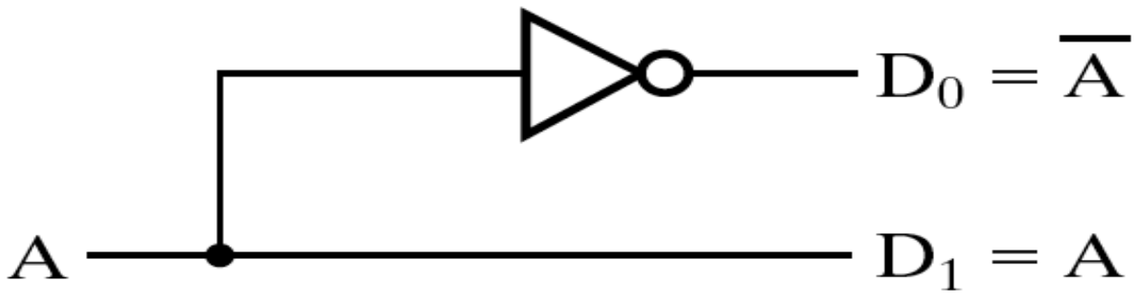
A simple 4-bit comparator.



1-2 Decoder

A	D_0	D_1
0	1	0
1	0	1

(a)

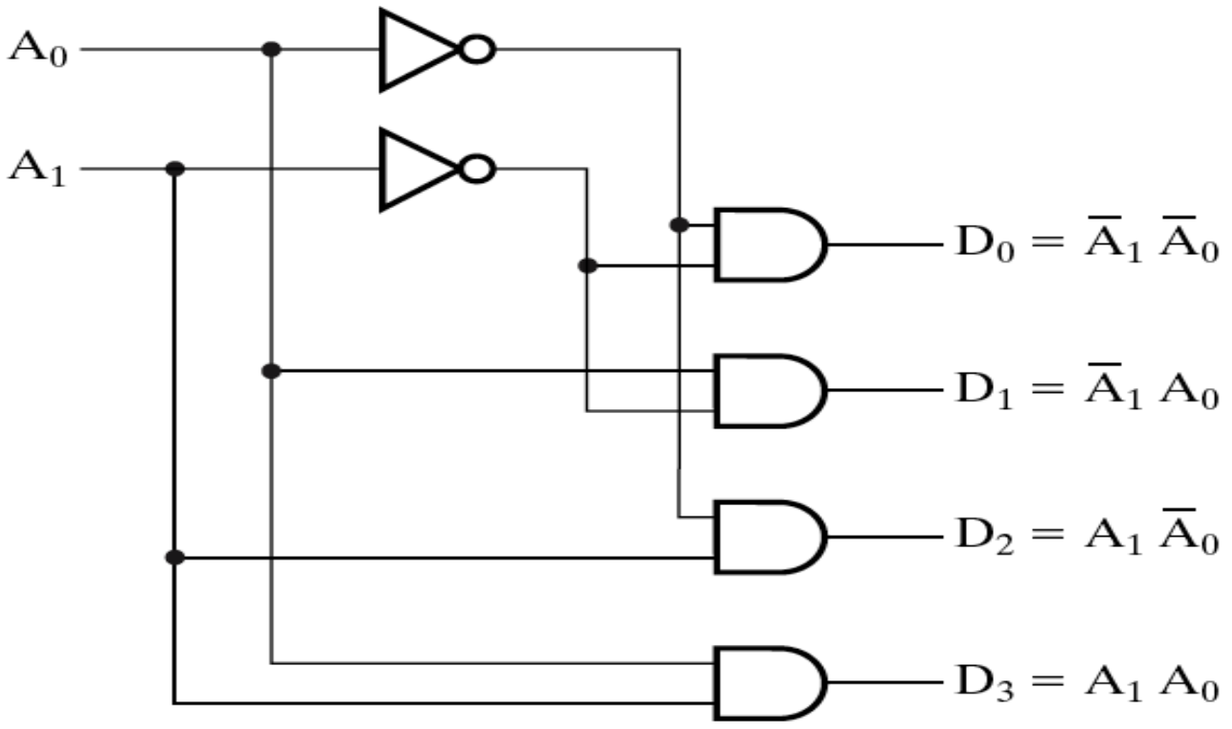


(b)

2-to-4 Decoder

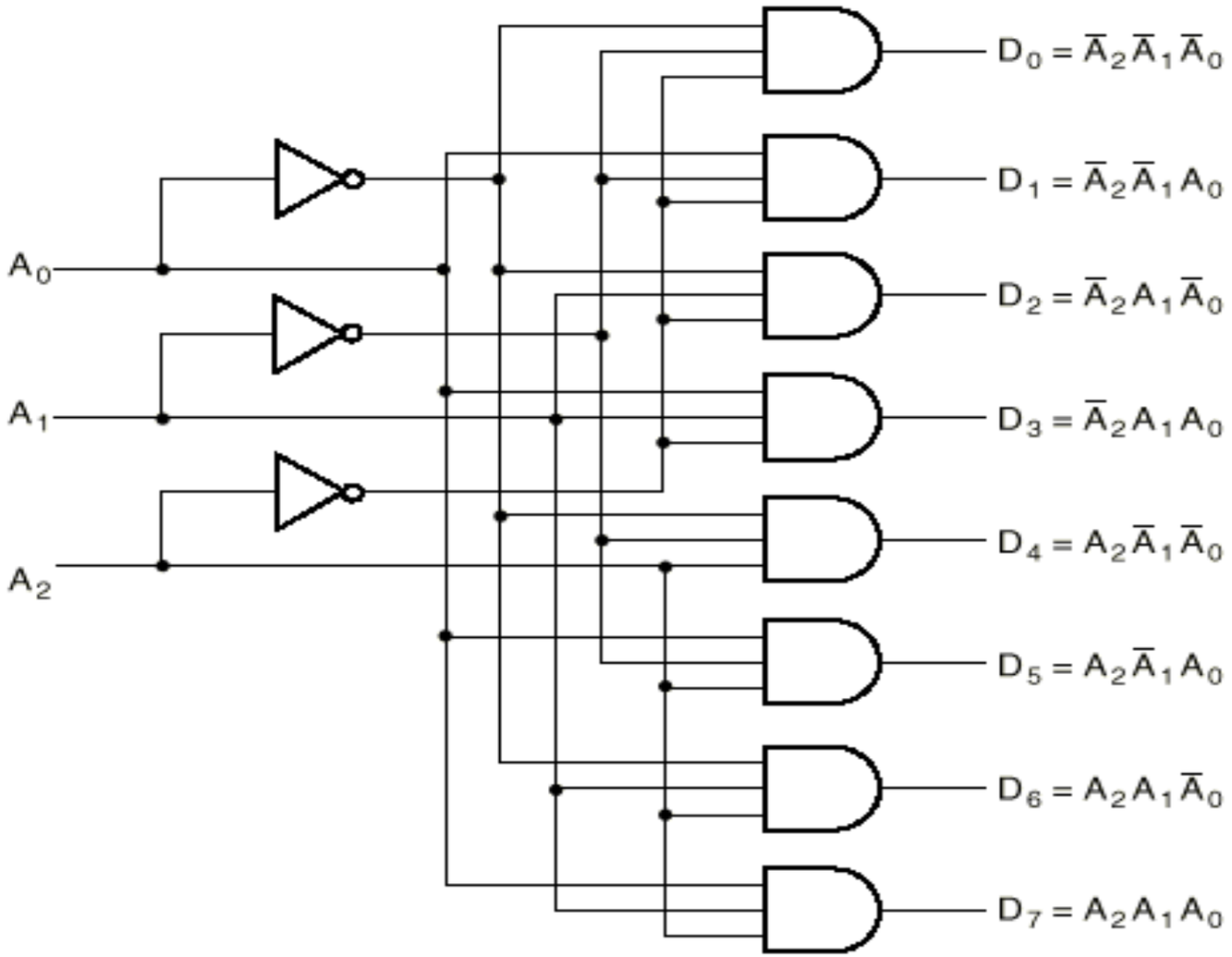
A_1	A_0	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

(a)

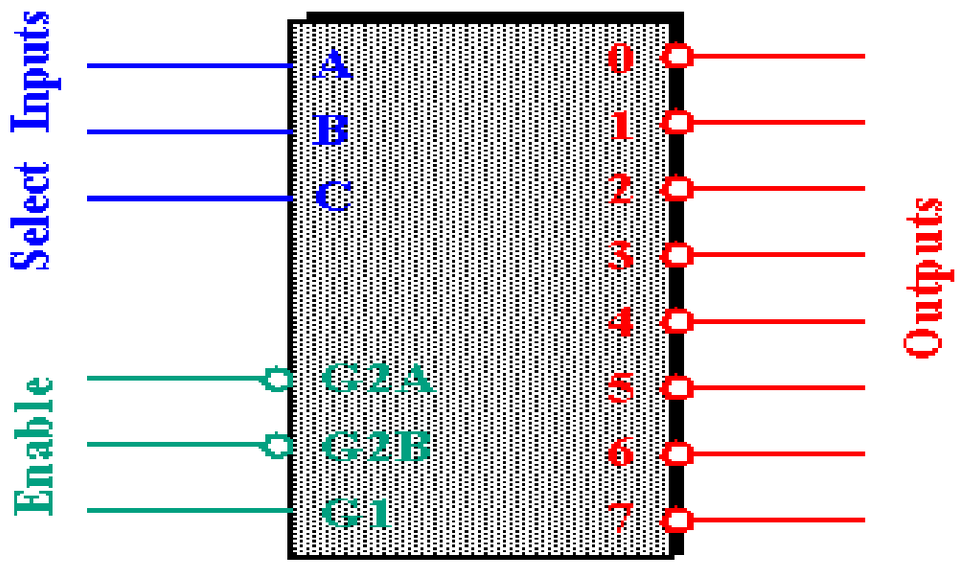


(b)

3-to-8 Decoder



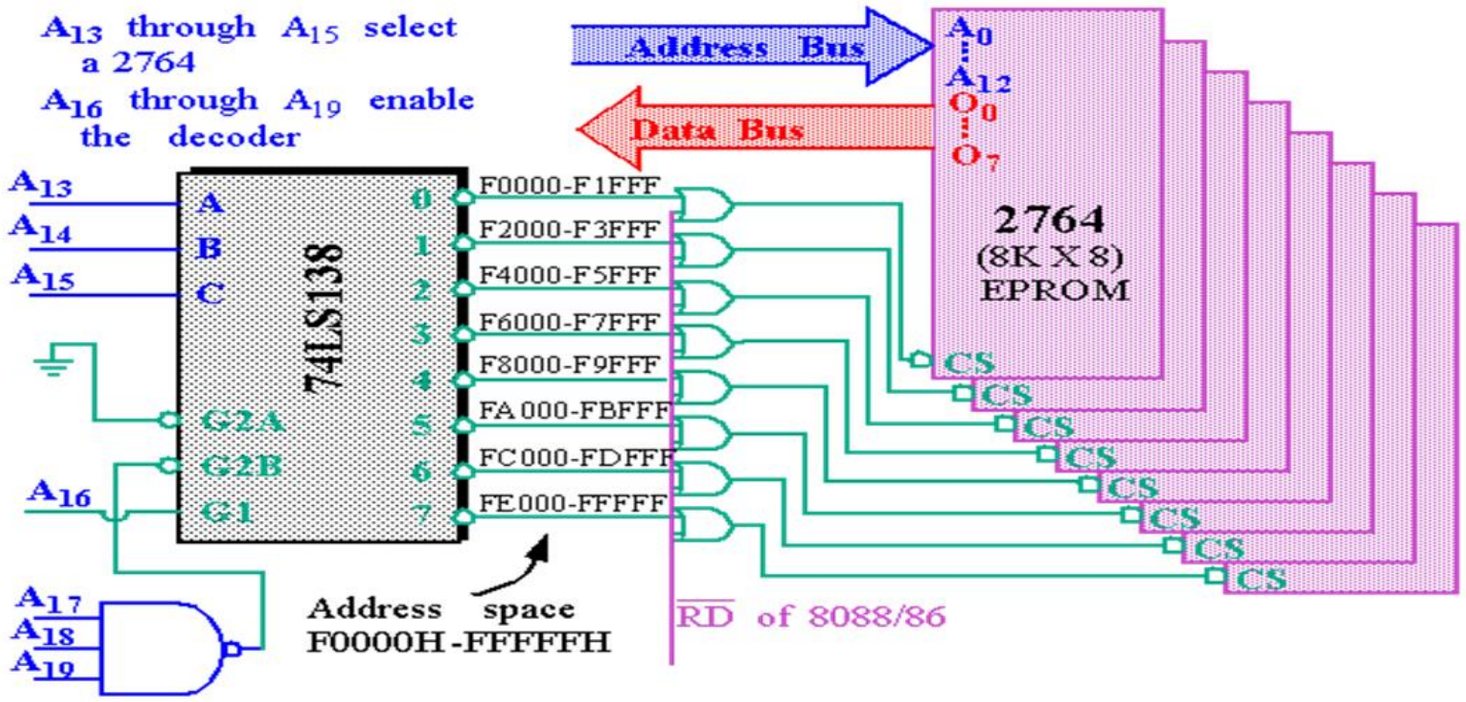
The 3-to-8 Line Decoder (74LS138)



Inputs						Output							
Enable			Select										
G2A	G2B	G1	C	B	A	0	1	2	3	4	5	6	7
1	X	X	X	X	X	1	1	1	1	1	1	1	1
X	1	X	X	X	X	1	1	1	1	1	1	1	1
X	X	0	X	X	X	1	1	1	1	1	1	1	1
0	0	1	0	0	0	0	1	1	1	1	1	1	1
0	0	1	0	0	1	1	0	1	1	1	1	1	1
0	0	1	0	1	0	1	1	0	1	1	1	1	1
0	0	1	0	1	1	1	1	1	0	1	1	1	1
0	0	1	1	0	0	1	1	1	1	0	1	1	1
0	0	1	1	0	1	1	1	1	1	1	0	1	1
0	0	1	1	1	0	1	1	1	1	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	1	0

Figure 10–14 The 74LS138 3-to-8 line decoder and function table.

Figure 10–15 A circuit that uses eight 2764 EPROMs for a 64K × 8 section of memory in an 8088 microprocessor-based system. The addresses selected in this circuit are F0000H–FFFFFH.



Address Decoding

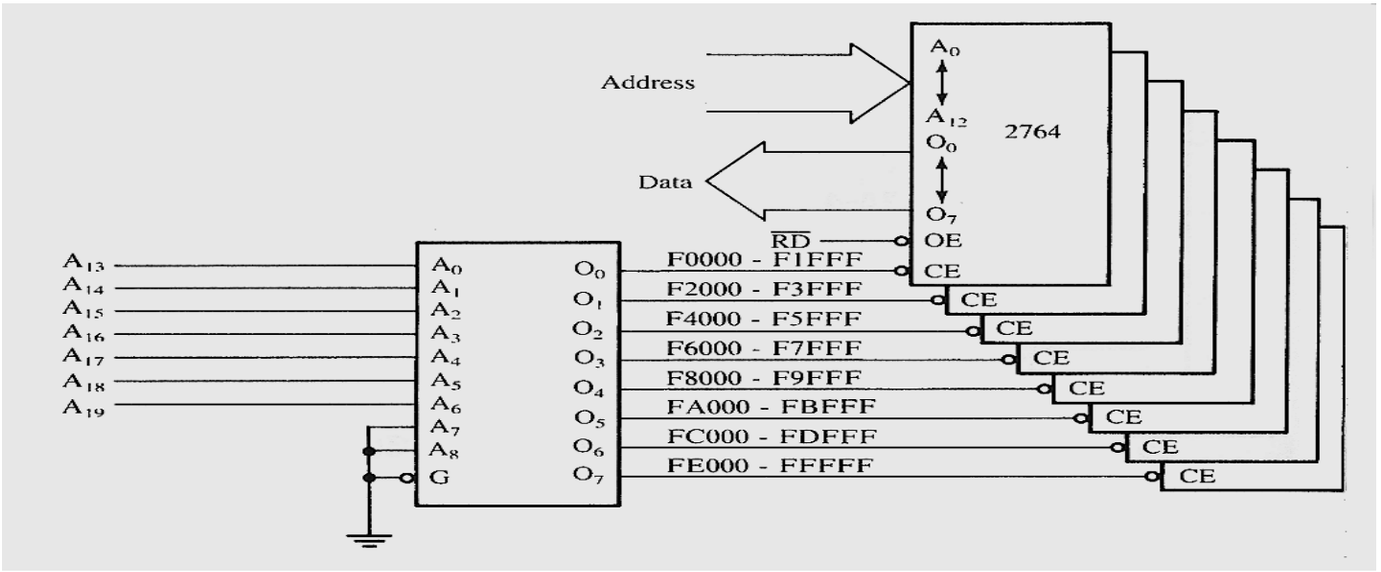


TABLE 10–1 The 82S147 PROM programming pattern for the circuit of Figure 10–17.

Inputs										*Outputs*							
$\overline{OE}$	A8	A7	A6	A5	A4	A3	A2	A1	A0	O0	O1	O2	O3	O4	O5	O6	O7
0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1
0	0	0	1	1	1	1	0	0	1	1	0	1	1	1	1	1	1
0	0	0	1	1	1	1	0	1	0	1	1	0	1	1	1	1	1
0	0	0	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1
0	0	0	1	1	1	1	1	0	0	1	1	1	1	0	1	1	1
0	0	0	1	1	1	1	1	0	1	1	1	1	1	1	0	1	1
0	0	0	1	1	1	1	1	1	0	1	1	1	1	1	1	0	1
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
all	other	combinations								1	1	1	1	1	1	1	1

Address Decoding

- Prevent overlap
- Input of decoding: CPU- address bus «m» wires
- Output off decoding: 2^m
- cs, en, ... Address decoding

Example: 10 adet belleğim var. CPU dan Address decoding için kaç adet adre hattına ihtiyaç var?

m=?, m=4, 10 → 16

Addressing of the Memory

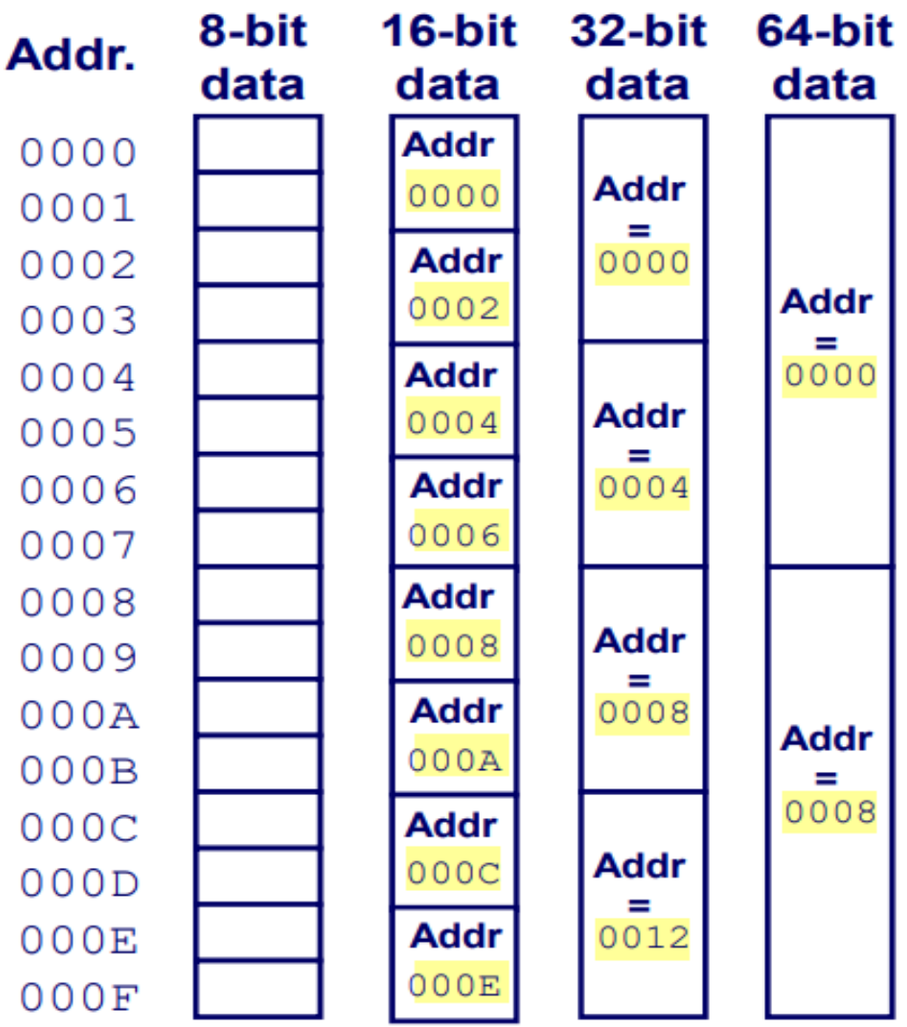
- The total capacity of a memory is calculated by using the number of wires on address bus: 2^n
- The number of wires on address bus is n
- Addressing index of the memory: $A_{n-1}\ A_{n-2}\ \ldots.\ A_2\ A_1\ A_0$
- The capacity range of the memory: 0,1,2, … , 2^{n-1}
- The number of wires of chip select input for decoding: m
- The number of wires of chip select output for decoding: 2^m, also,

  $2^m \geq$ the number of the rams

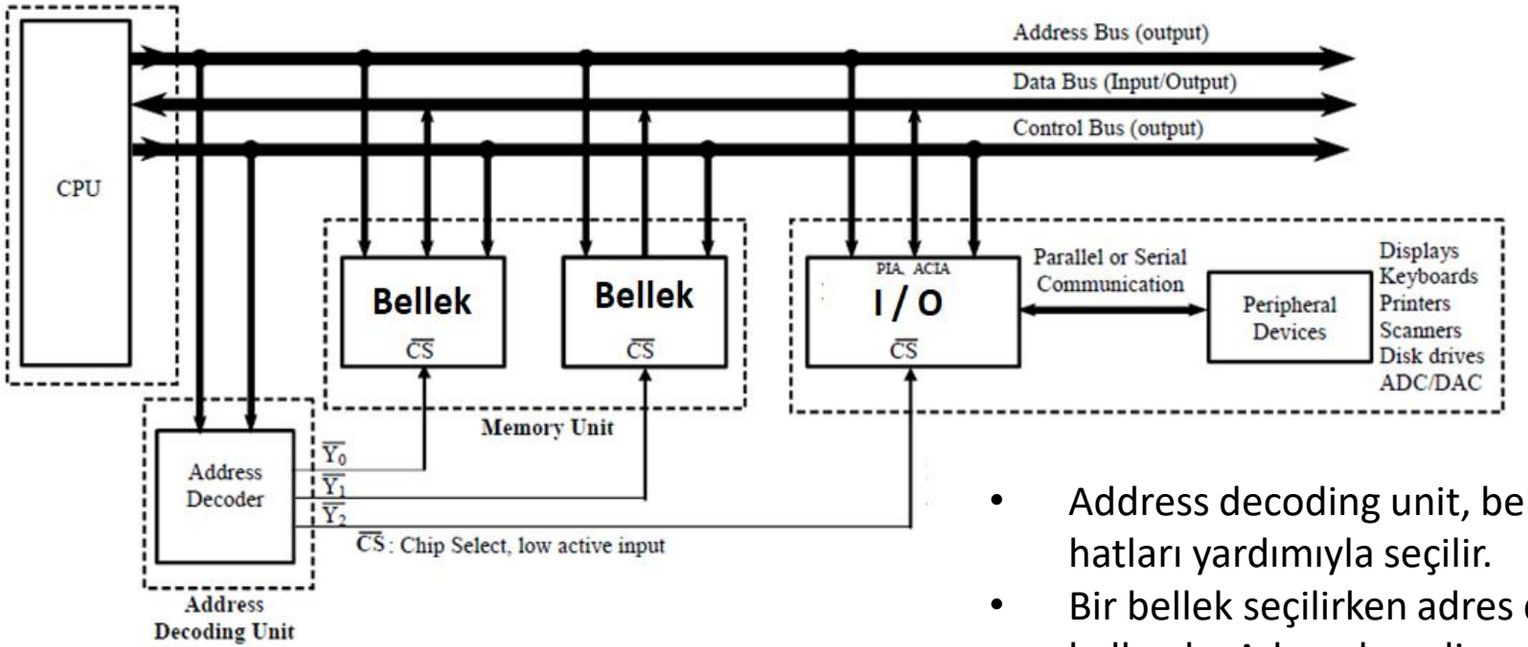
- Overlap: It must be written only selected memory cell. The decoding prevents from overlaps.

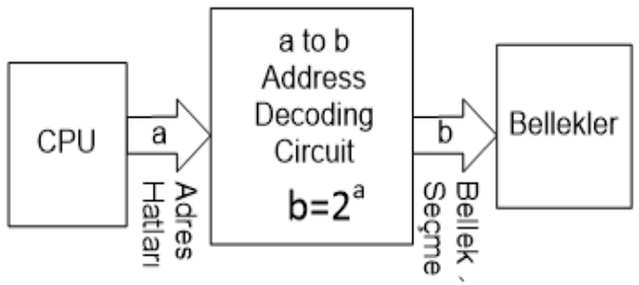
Address of Multi-byte Data

- **Every byte has a unique address**
- ***So, if data spans multiple bytes, what is address?***
- **Data always addressed by its lowest address**
  - address of first byte in memory
- **Alignment**
  - Data elements are aligned by size
    - for a primitive (single datum) with *K* bits, address must be multiple of *K*
    - **chars, booleans** at any address
    - **shorts** at even addresses
    - **ints, floats,** pointers every 4th addr
    - **doubles** every 8th address
    - etc.
  - Arrays, structures, and classes
    - alignment determined by size of largest primitive (single datum)



Address Decoding Unit





- Address decoding unit, bellek seçer. Belleğin gözü ise CPU'dan gelen adres hatları yardımıyla seçilir.
- Bir bellek seçilirken adres decoding devresine CPU'dan gelen adres hatları kullanılır. Adres decoding devresinin çıkışları bellekleri seçmede kullanılır.
- Seçilecek bellek sayısı=Address decoding devresi çıkış sayısı=2^m dir. Burada m CPU'dan gelen adres hattı sayısıdır.
- Herbir belleğin kapasitesini belirleyen (Kapasite=2^n) n adet hat CPU'dan gelir.
- Amaç aynı anda bir belleğin ilgili veri gözünün seçilmesidir.
- Adres decoding devresinin çıkış sayısı bellek sayısına eşit ya da büyük olmak olmak sorundadır.
- Adres decoding devresini girişi, CPU dan gelecek adres hattı sayısı belirlenir. Özeldir.

Address Decoding Unit

- Seçilecek bellek sayısı=Address decoding devresi çıkış sayısı=2^m dir.
- Örnek: 172 adet bellek var ise 2^m =172 ise m=8 alınır. Çünkü 2^7 =128 dir. Bellek sayısına eşit ya da büyük olan seçilri.
- O halde CPU'dan Address Decoding Devresinin girişine gelecek hat sayısı=8 dir. Bu durumda seçilecek bellek sayısı 256 olur.
- Bellek Address Decoding Devresi yardımıyla seçilir. Seçilen belleğin yazılıp ya da okunacakilgili gözü ise CPU dan parelel gelecek n_i adres hattı ile belirlenir. Burada, i=1 ... 2^m dir.
- Seçilen belleğin kapasitesi=2^{ni} byte olur.

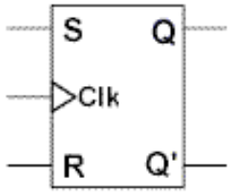
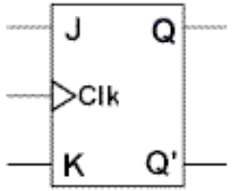
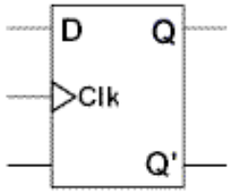
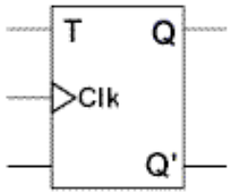
“Sıralı Mantık”

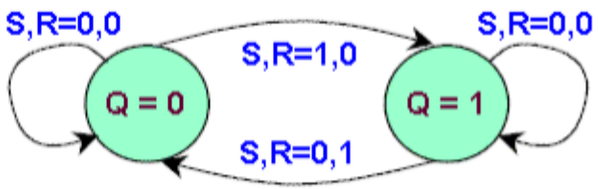
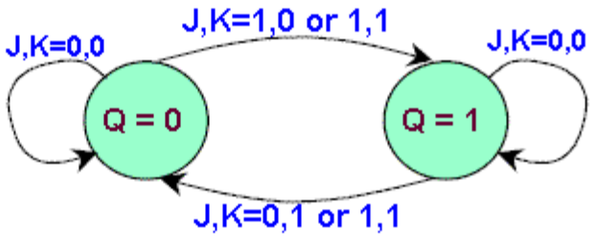
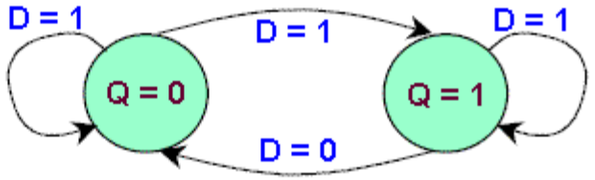
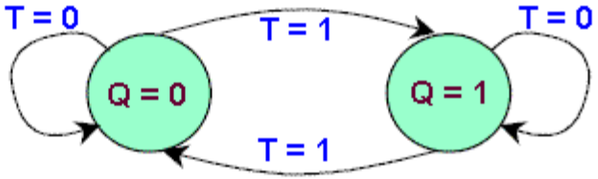
Sıralı Mantık (Sequential Logic)

- Sequential logic has **memory;** the circuit stores the result of the previous set of inputs. The current output depends on inputs **in the past** as well as present inputs.
  - The basic element in sequential logic is the **bistable latch** or **flip-flop**, which acts as a memory element for one bit of data.

Sıralı Mantık (Sequential Logic)

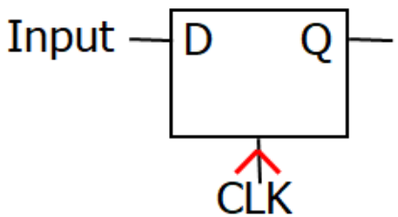
- Sıralı mantık belleğe sahiptir; devre, önceki giriş setinin sonucunu depolar. Çıkış, geçmişteki girişlerin yanı sıra mevcut girişlere bağlıdır.
- Sıralı mantıktaki temel öğe, bir bitlik veri için bellek öğesi olarak işlev gören iki durumlu mandal (latch) veya iki durumlu tetikleme devresidir (flip – flop).

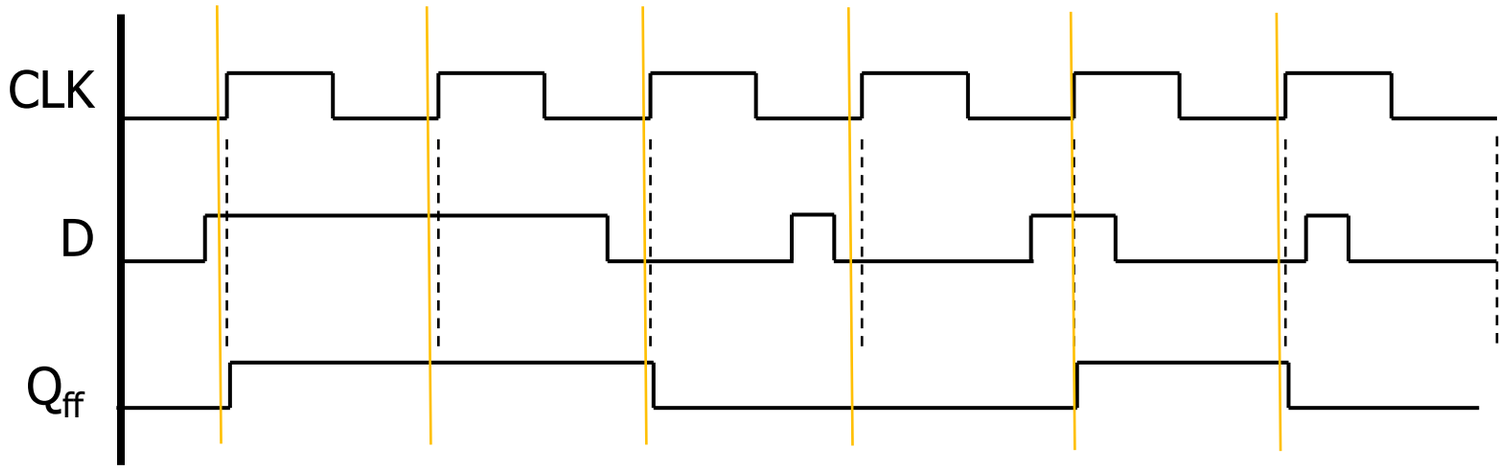
FLIP-FLOP NAME	FLIP-FLOP SYMBOL	CHARACTERISTIC EQUATION	EXCITATION TABLE
SR		$Q_{(next)} = S + R'Q$ $SR = 0$	**Q** | **$Q_{(next)}$** | **S** | **R** 0 | 0 | 0 | X 0 | 1 | 1 | 0 1 | 0 | 0 | 1 1 | 1 | X | 0
JK		$Q_{(next)} = JQ' + K'Q$	**Q** | **$Q_{(next)}$** | **J** | **K** 0 | 0 | 0 | X 0 | 1 | 1 | X 1 | 0 | X | 1 1 | 1 | X | 0
D		$Q_{(next)} = D$	**Q** | **$Q_{(next)}$** | **D** 0 | 0 | 0 0 | 1 | 1 1 | 0 | 0 1 | 1 | 1
T		$Q_{(next)} = TQ' + T'Q$	**Q** | **$Q_{(next)}$** | **T** 0 | 0 | 0 0 | 1 | 1 1 | 0 | 1 1 | 1 | 0

NAME	STATE DIAGRAM
SR	
JK	
D	
T	

The D flip-flop

- Input sampled at clock edge
  - Rising edge: Input passes to output
  - Otherwise: Flip-flop holds its output
- Flip-flops can be rising-edge triggered or falling-edge triggered
- Clok işaretinin yükselen kenarında çıkış girişe eşit olur (Q=D). Clok işeretinin diğer tüm durumlarında çıkış değişmez.
- Şu anki durum (Q) ve bir sonraki durum (D) birlikte göz önüne alınır.

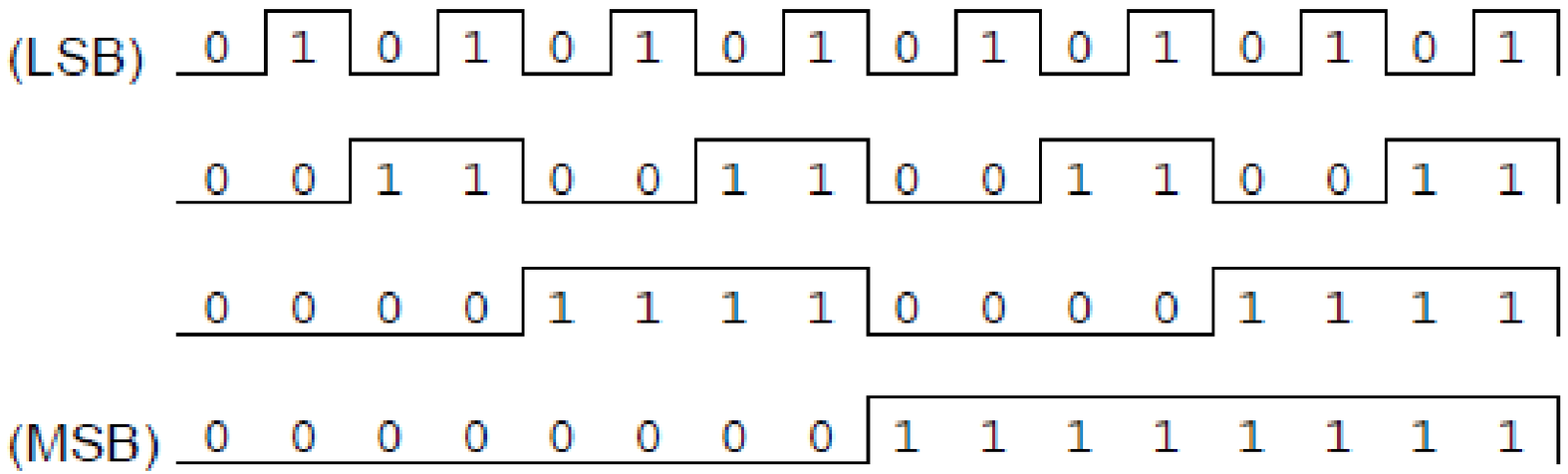




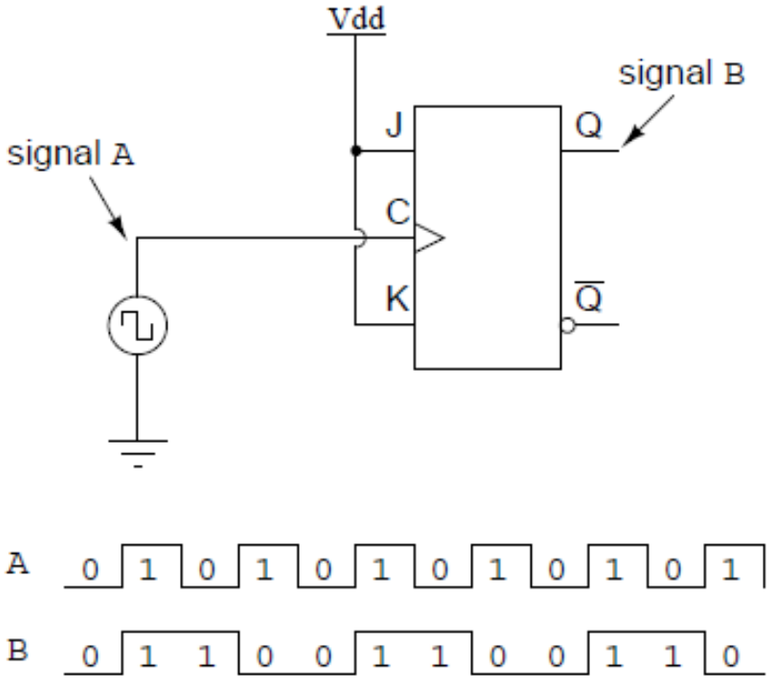
COUNTERS

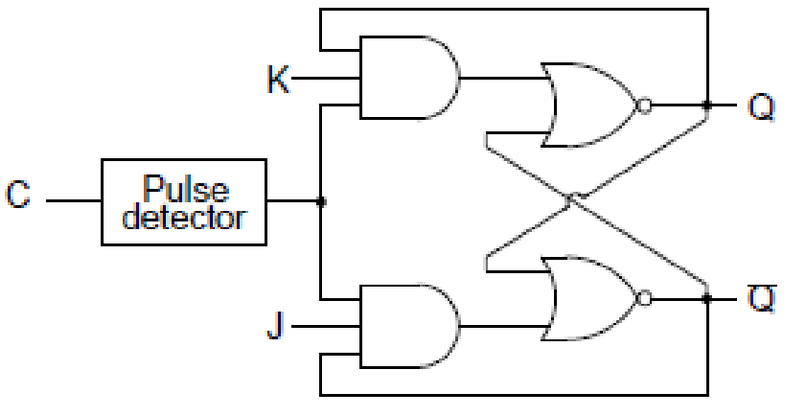
Binary count sequence

- If we examine a four-bit binary count sequence from 0000 to 1111, a definite pattern will be evident in the "oscillations" of the bits between 0 and 1



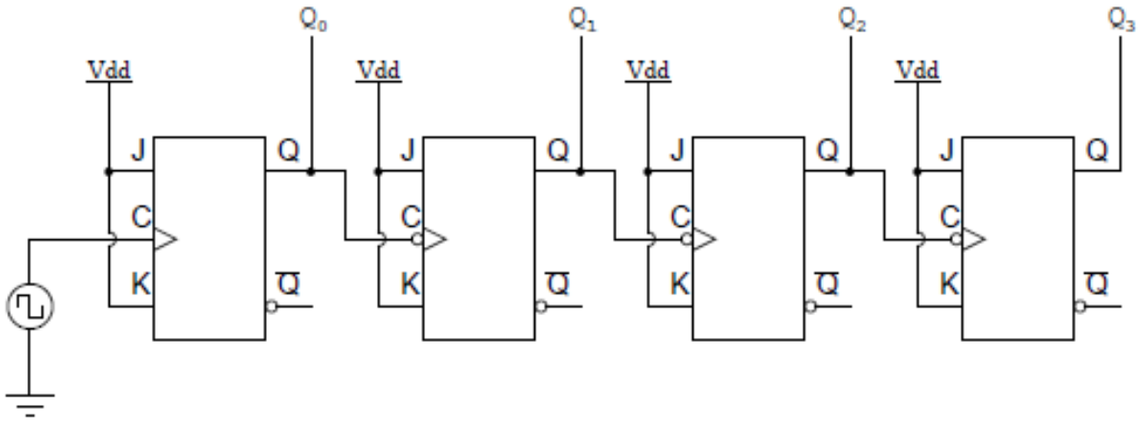
0 0 0 0
0 0 0 1
0 0 1 0
0 0 1 1
0 1 0 0
0 1 0 1
0 1 1 0
0 1 1 1
1 0 0 0
1 0 0 1
1 0 1 0
1 0 1 1
1 1 0 0
1 1 0 1
1 1 1 0
1 1 1 1

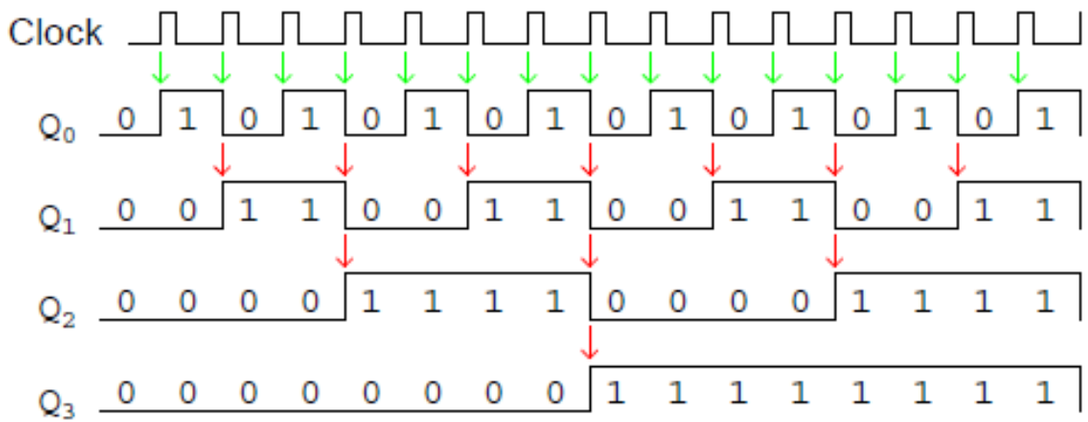




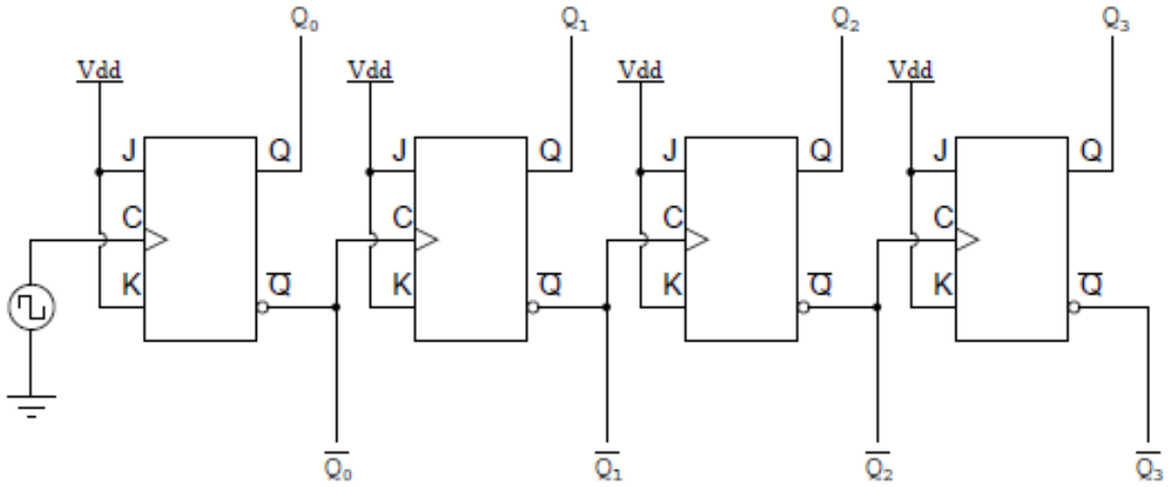
C	J	K	Q	$\overline{Q}$
⌐	0	0	latch	latch
⌐	0	1	0	1
⌐	1	0	1	0
⌐	1	1	toggle	toggle
x	0	0	latch	latch
x	0	1	latch	latch
x	1	0	latch	latch
x	1	1	latch	latch

A four-bit "up" counter

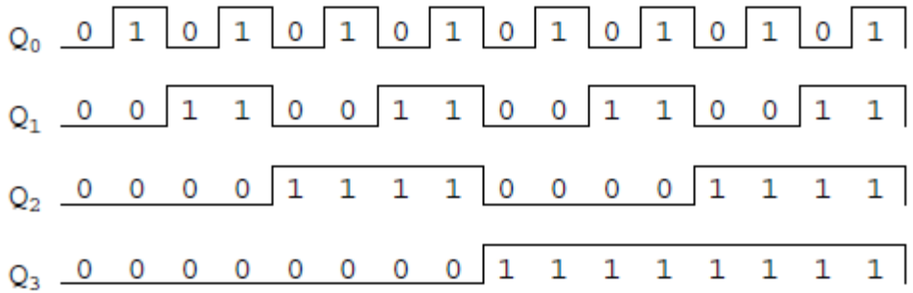




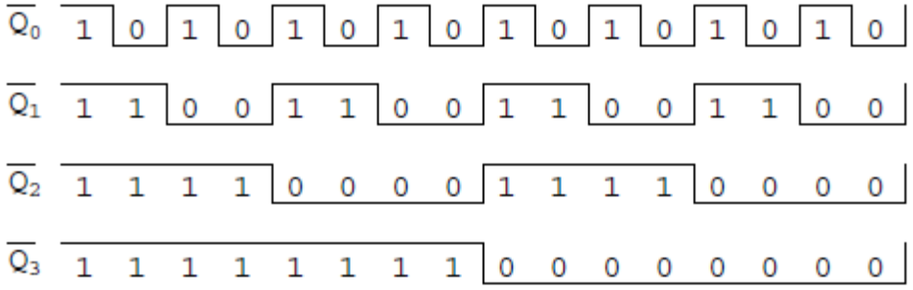
A simultaneous "up" and "down" counter



"Up" count sequence

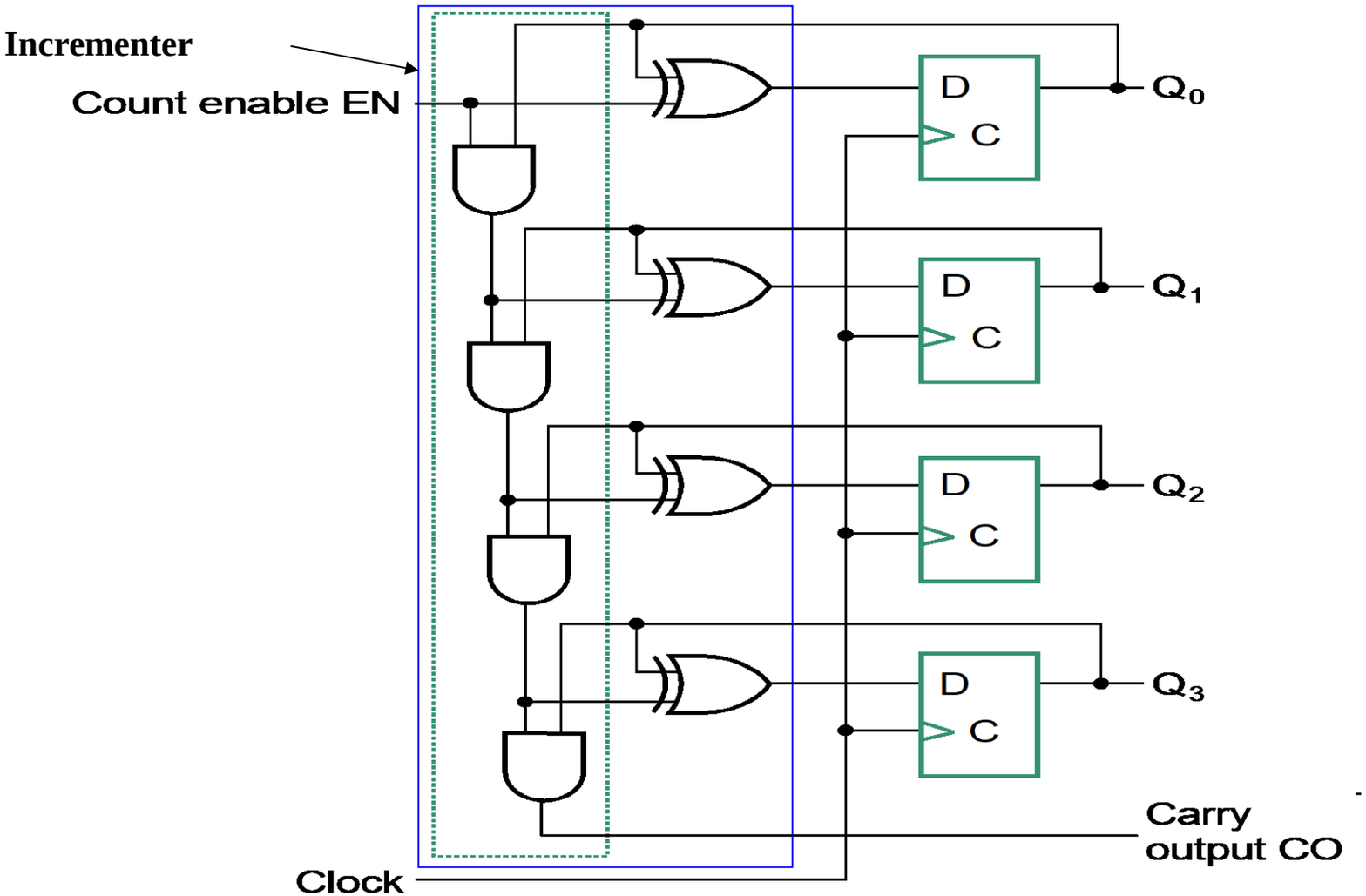


"Down" count sequence



Synchronous Counters (continued)

- Internal details =>
- Internal Logic
  - XOR complements each bit
  - AND chain causes complement of a bit if all bits toward LSB from it equal 1
- Count Enable
  - Forces all outputs of AND chain to 0 to “hold” the state
- Carry Out
  - Added as part of incrementer
  - Connect to Count Enable of additional 4-bit counters to form larger counters



(a) Logic Diagram-Serial Gating

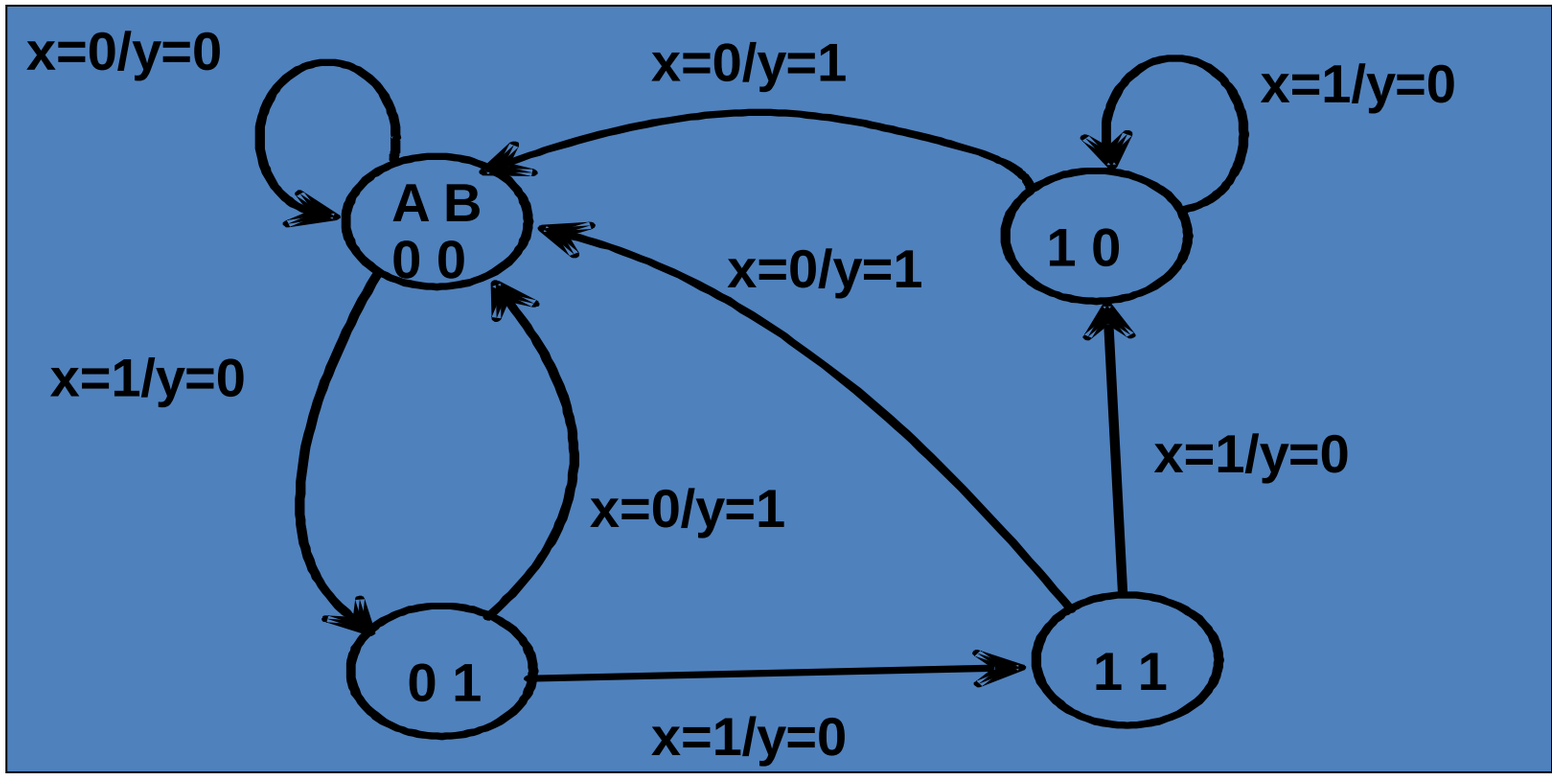
State Table

State Table

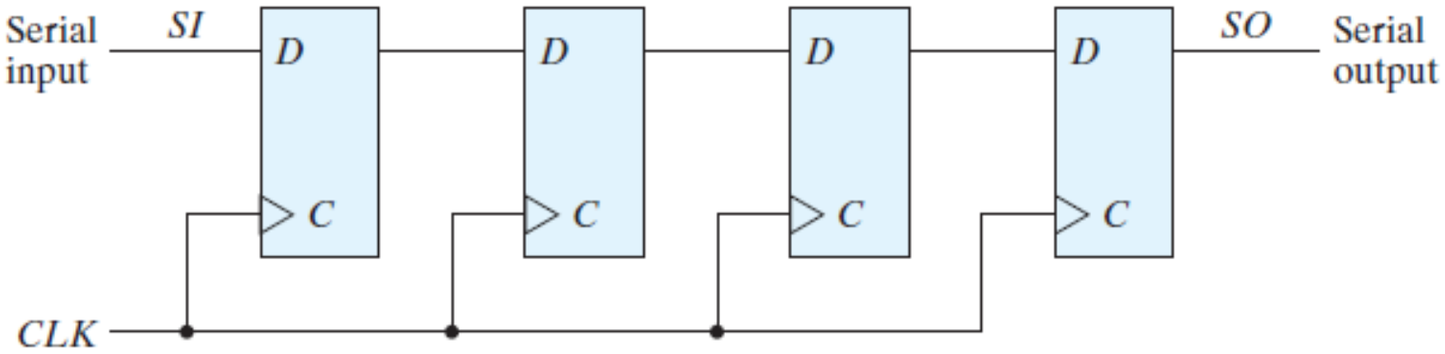
- *State table* – a multiple variable table with the following four sections:
  - *Present State* – the values of the state variables for each allowed state.
  - *Input* – the input combinations allowed.
  - *Next-state* – the value of the state at time (t+1) based on the present state and the input.
  - *Output* – the value of the output as a function of the present state and (sometimes) the input.
- From the viewpoint of a truth table:
  - the inputs are Input, Present State
  - and the outputs are Output, Next State

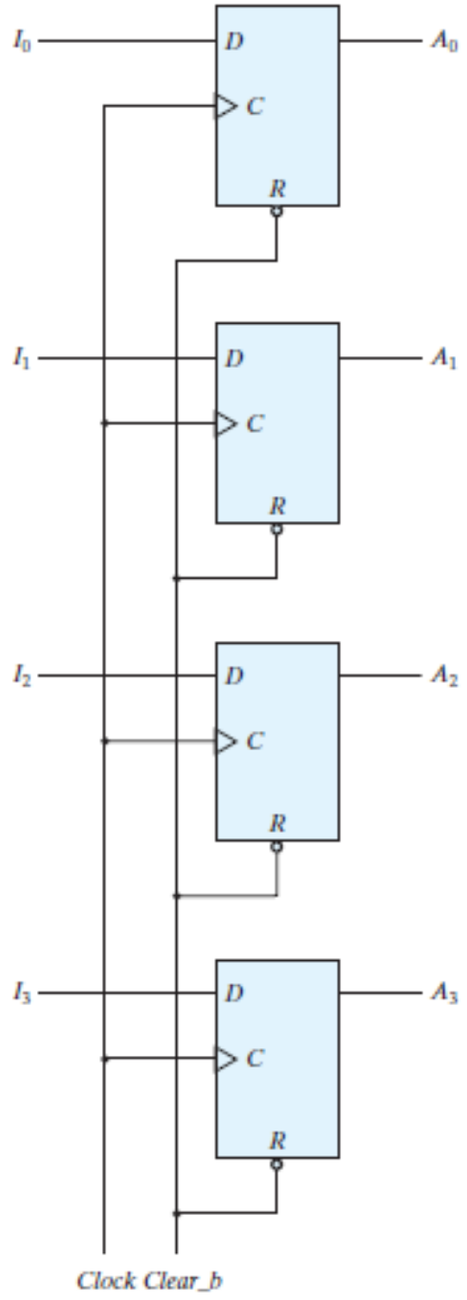
Example-1 : State Diagram

- Which type?
- Diagram gets confusing for large circuits
- For small circuits, usually easier to understand than the state table

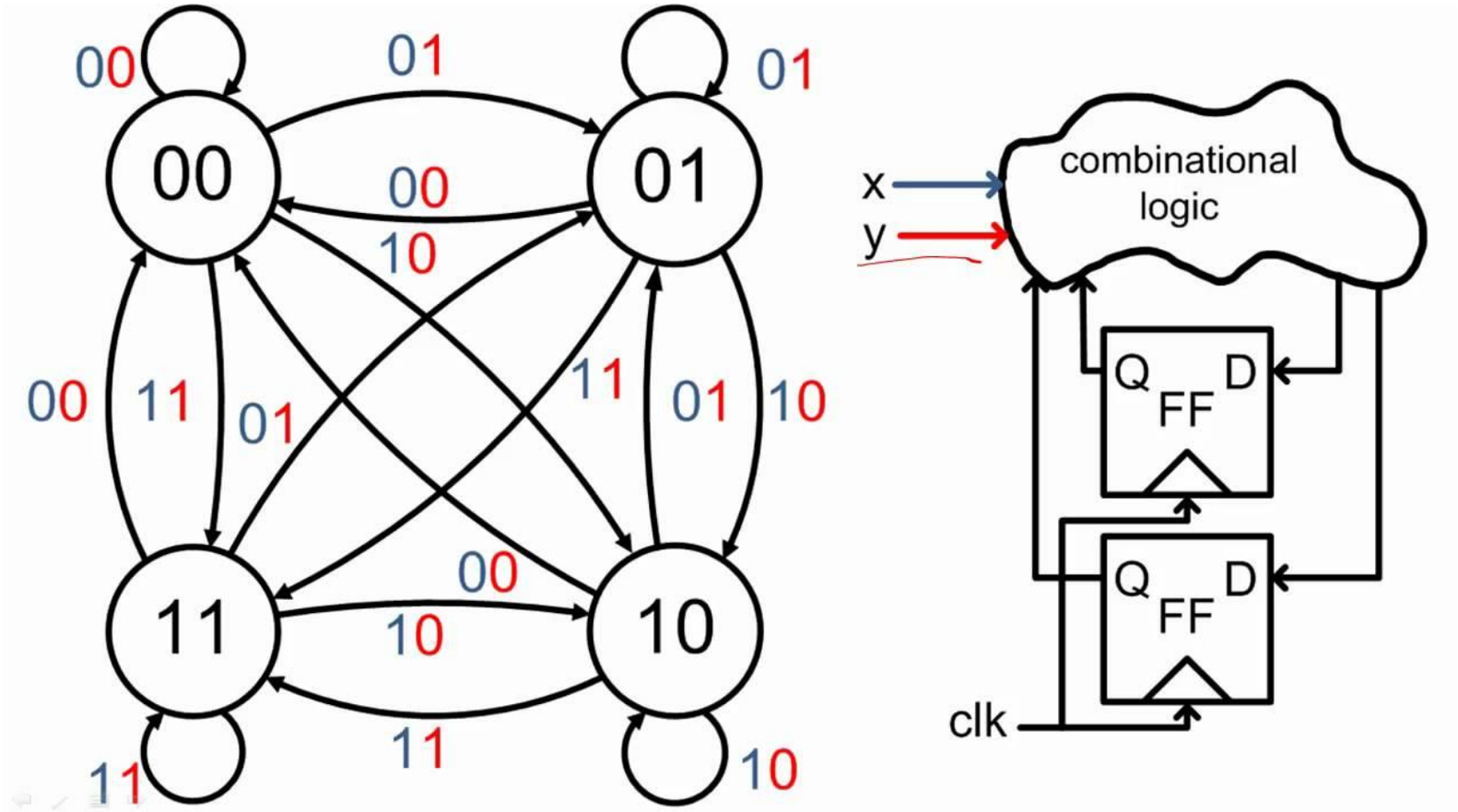


Example-2 : Registers





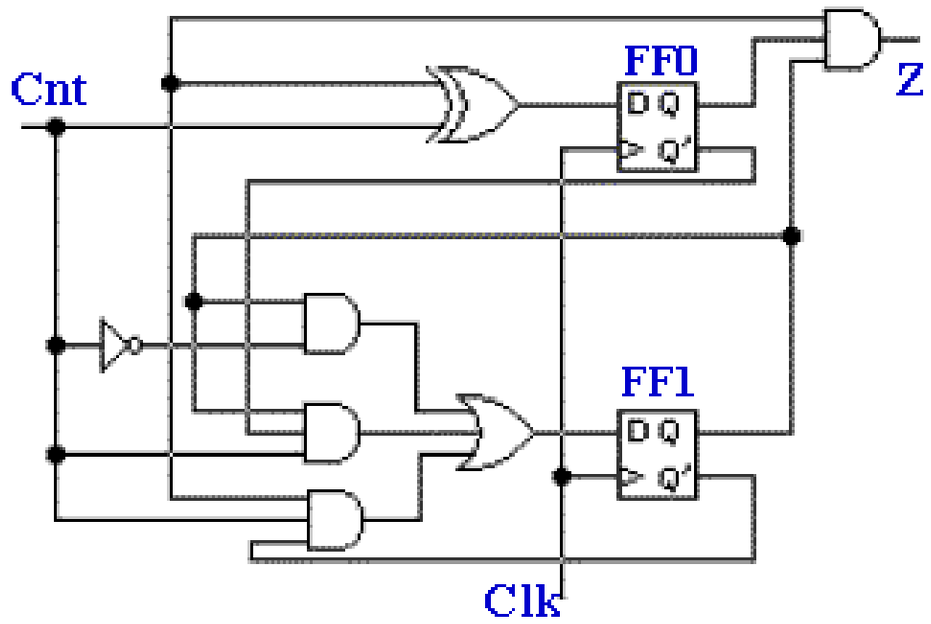
Example-3: Durum Diyagramı



Example-4 : Analysis of sequential circuits

Analysis of Sequential Circuits.

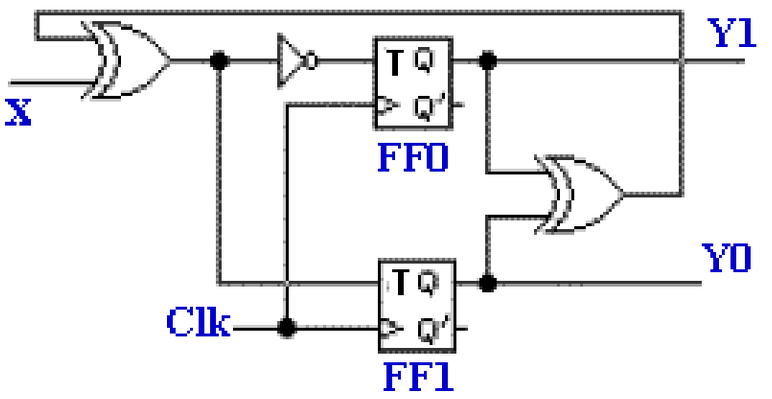
1. Derive a) excitation equations, b) next state equations,



c) a state/output table, and

d) a state diagram

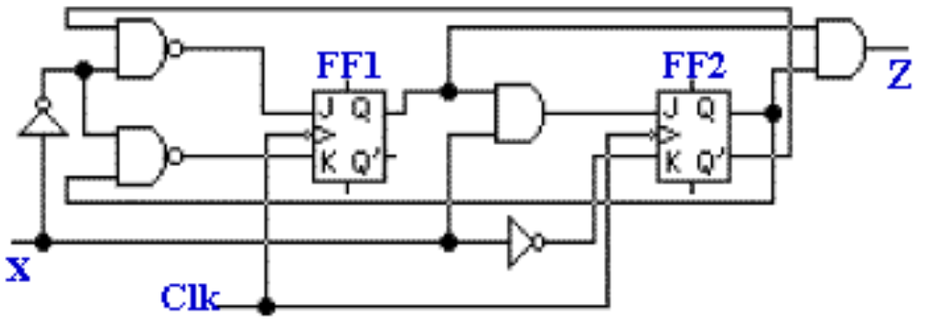
2. Derive a) excitation equations, b) next state equations,



c) a state/output table,

and d) a state diagram

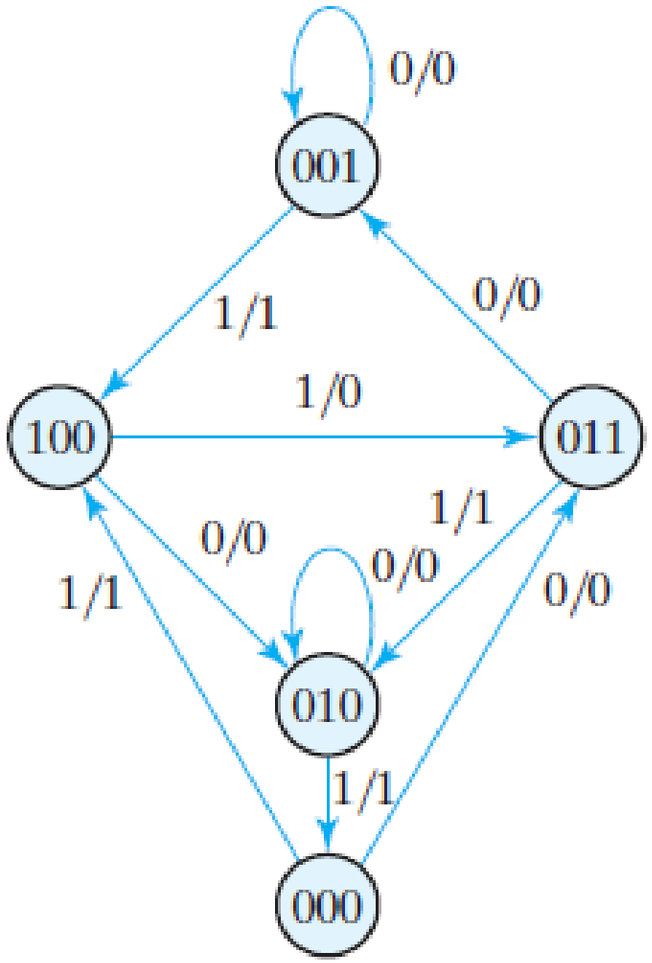
3. Derive a) excitation equations, b) next state equations,



c) a state/output table,

and d) a state diagram

Example-5

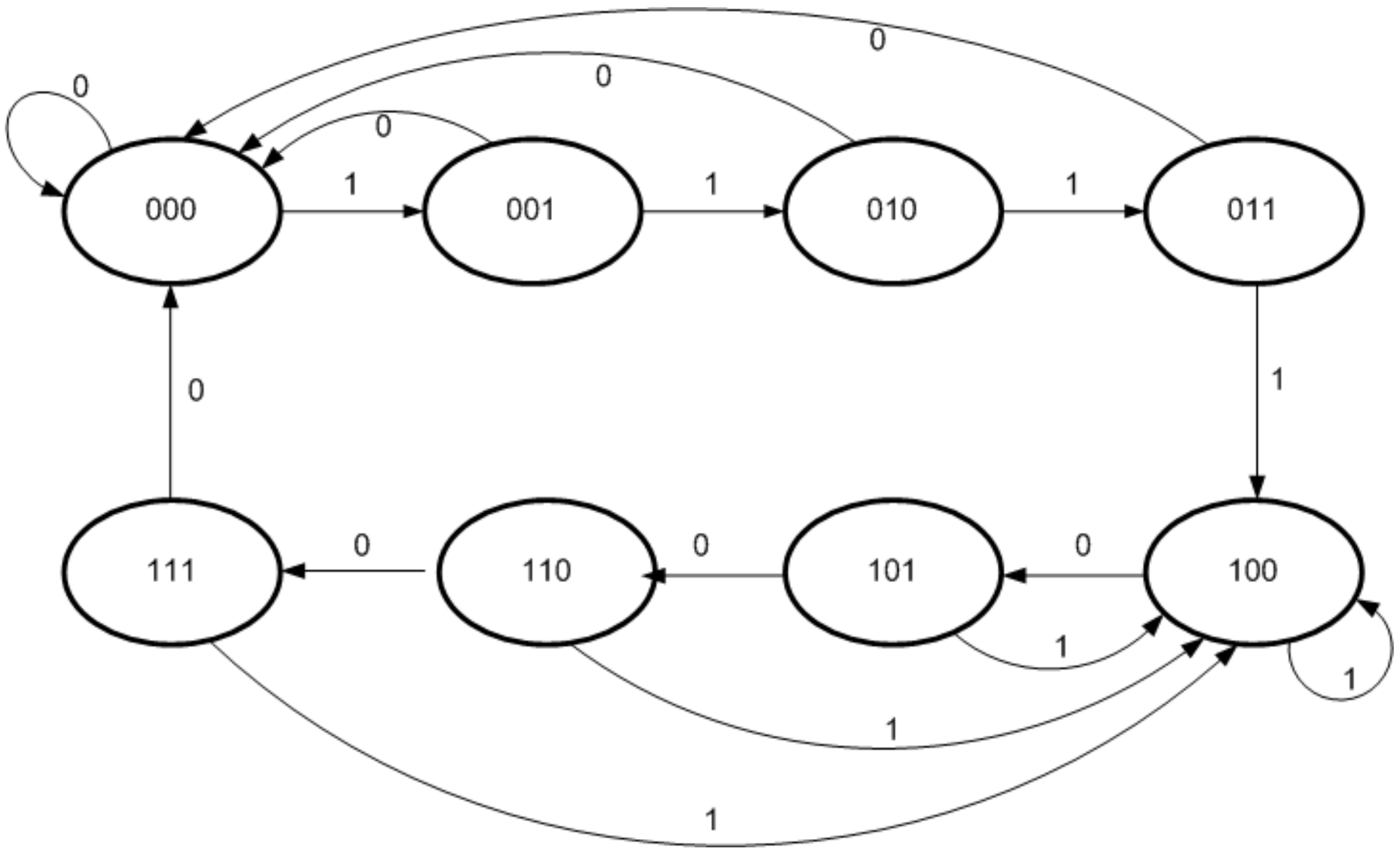


Example-6:

Durum diyagramı verilen devrenin

- Şuanki ve birsonraki durumlara göre D -ikili devre sayısını bulun
- Durum Tablosunu oluşturun
- Karnaugh Diyagramı ile indirgeyerek çıkış denklemlerini bulun
- Devreyi çizin
- Yorumlayın

6- Durum Diyagramı

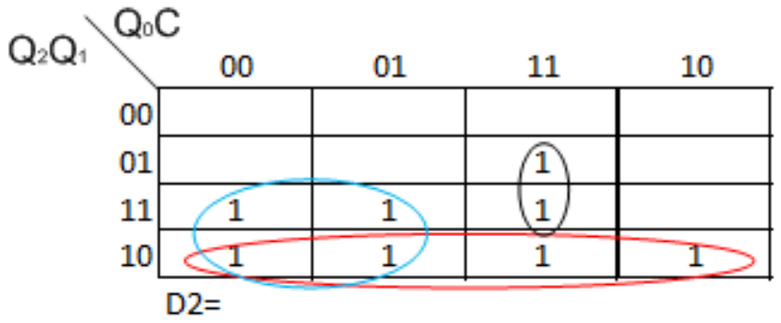


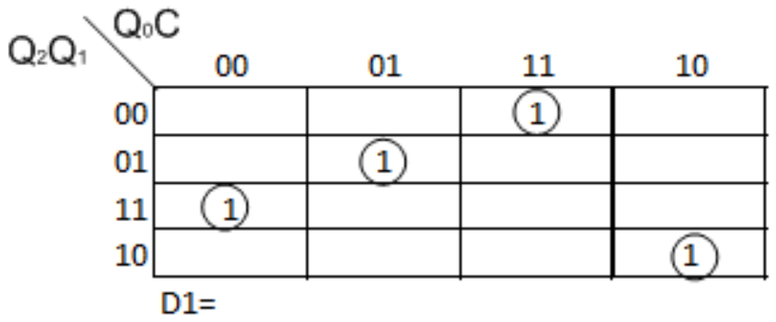
6- Şu anki ve bir sonraki durumlara göre ikili devre sayısını bulunması

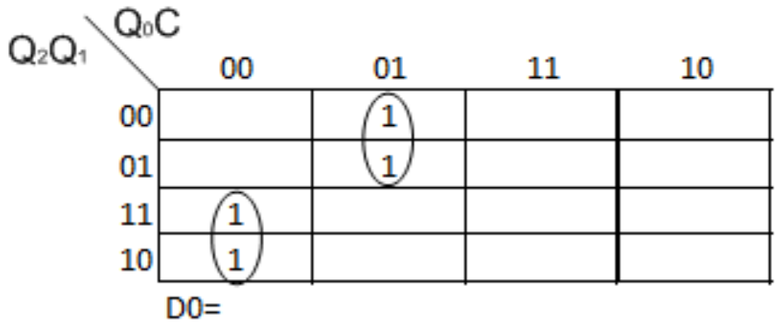
- İkili sayısı=3 adettir. Çünkü Durum diyagramında tüm durumlar 0 ile 7 arasında değişmektedir. Toplam durum sayısı=8=2^3 dür.
- Şu anki durumlar D-ikili devresini Q çıkışlarında bulunmaktadır. Bir sonraki durum ise D-ikili devresinin D girişlerinde bulunmaktır.
- Clok'un yükselen kenarı ile D-ikili devresi tetiklendiğinde Q-çıkışları D-girişlerine eşit olur.

6-Durum Tablosunun oluşturulması ve Karnaugh diyagramı yardımıyla indirgnmesi

Şu anki durum			Giriş	Bir sonraki durum		
Q2	Q1	Q0	C	D2	D1	D0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	0
0	0	1	1	0	1	0
0	1	0	0	0	0	0
0	1	0	1	0	1	1
0	1	1	0	0	0	0
0	1	1	1	1	0	0
1	0	0	0	1	0	1
1	0	0	1	1	0	0
1	0	1	0	1	1	0
1	0	1	1	1	0	0
1	1	0	0	1	1	1
1	1	0	1	1	0	0
1	1	1	0	0	0	0
1	1	1	1	1	0	0







6-Denklemler

- $D_2 = Q_2Q_1' + Q_2Q_0' + Q_1Q_0C$
- $D_1 = Q_2Q_1Q_0'C' + Q_2'Q_1Q_0'C + Q_2'Q_1'Q_0C + Q_2Q_1'Q_0C'$
- $D_0 = Q_2'Q_0'C' + Q_2Q_0'C$

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.

Usage Notes

- A lot of slides are adopted from the presentations and documents published on internet by experts who know the subject very well.
- I would like to thank who prepared slides and documents.
- Also, these slides are made publicly available on the web for anyone to use
- If you choose to use them, I ask that you alert me of any mistakes which were made and allow me the option of incorporating such changes (with an acknowledgment) in my set of slides.

Sincerely,

Dr. Cahit Karakuş

cahitkarakus@gmail.com