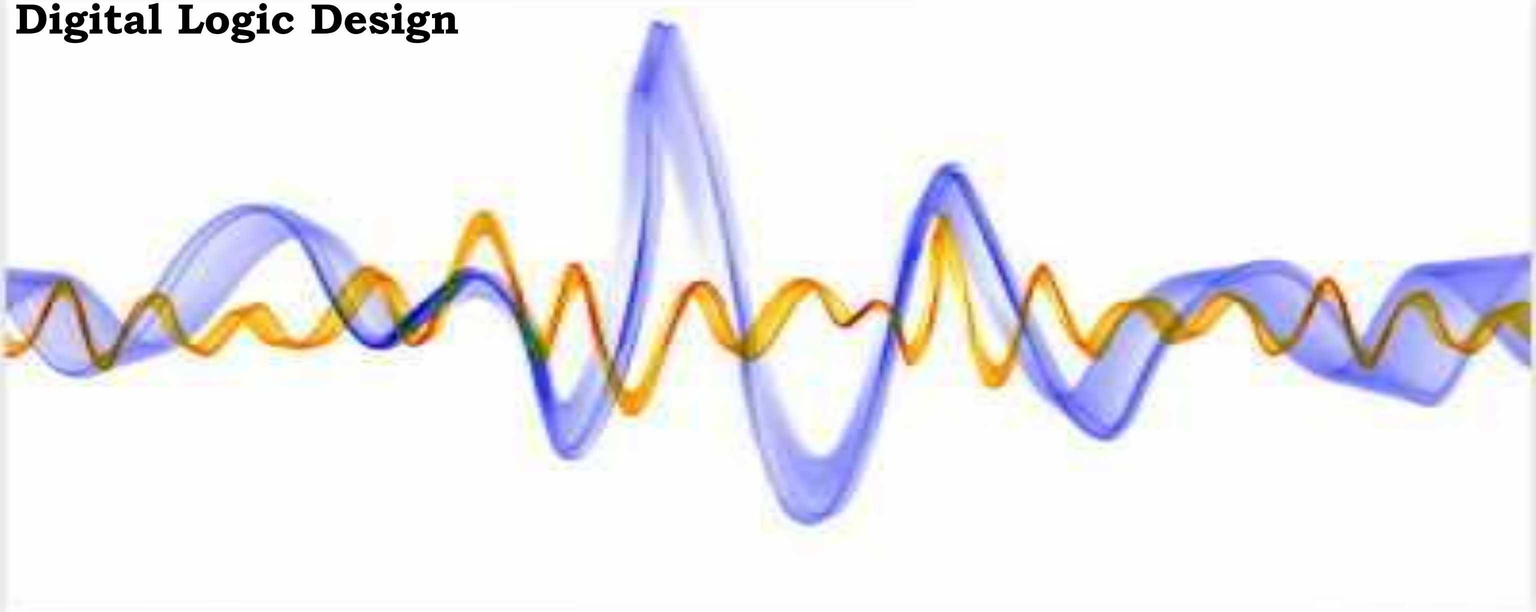
Digital Logic Design



Lecture 6:

Chapter 4: Combinational Logic

Mirvat Al-Qutt, Ph.D

Computer Systems Department , FCIS,

Ain Shams University

Design Procedure

- **Input:** the specification of the problem.
- **Output:** the logic circuit diagram or Boolean functions.

1. determine the required **number of inputs** and **outputs** from the specification
2. **derive the truth table** that defines the required relationship between inputs and outputs
3. obtain the **simplified Boolean function** for each output as a function of the input variables
4. **draw the logic diagram** and verify the correctness of the design.

Code Conversion Design Problems

- It is sometimes necessary to use the output of one system as the input to another.
  - A conversion circuit must be inserted between the two system if each uses different codes for the same information.
    - Thus, a code converter is a circuit that makes the two system compatible even though each uses a different binary code.
  - To convert from binary code A to binary code B, the input lines must supply the bit combination of elements as specified by code A and the output lines must generate the corresponding bit combination of code B.

Code Conversion Example

- **BCD to Excess-3 Code Converter**

2

1

- **Input BCD**
- **4 – Variables Input**

- **Output Excess-3**
- **4 – Variables output**

Input BCD				Output Excess-3 Code			
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

Code Conver

- **BCD to Excess-3 Code Converter**

1

- **Input BCD**
- **4 – Variables Input**
- **Output Excess-3**
- **4 – Variables output**

Input BCD- Code				Output Excess -3 Code			
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

Code Conversion Example

3

- **Boolean Expression :**
- The six don't care minterms (10~15) are marked with X.
- Each of four maps represents one of the four outputs of this circuit as a function of the four input variables.

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

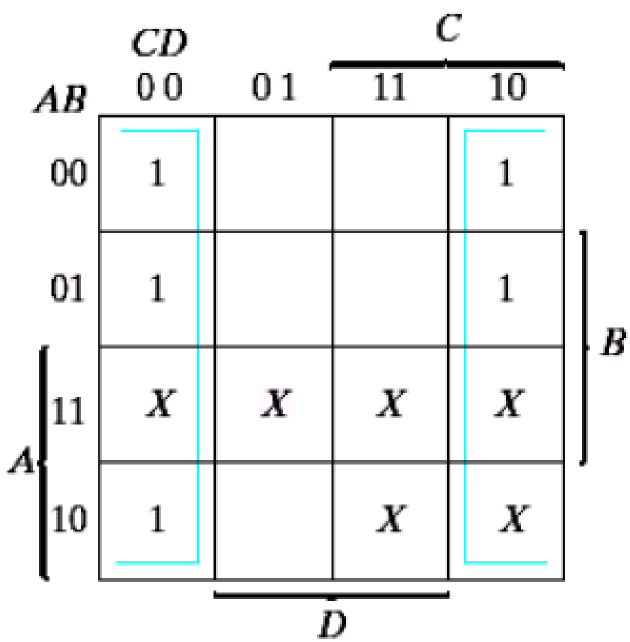
$X = B'C + B'D + BC'D'$

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

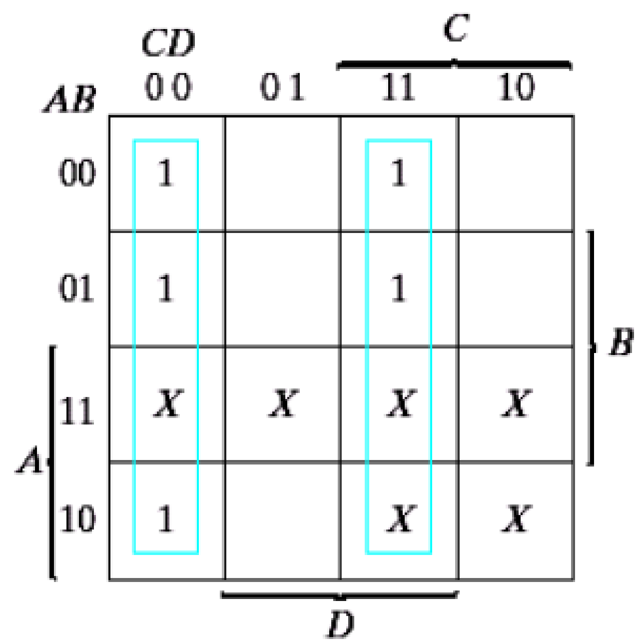
$w = A + BC + BD$

Code Conversion Example

- **Boolean Expression :** 3



$z = D'$



$y = CD + C'D'$

Code Conversion Example

- **Logic Diagram:** Reduce the number of gates used.

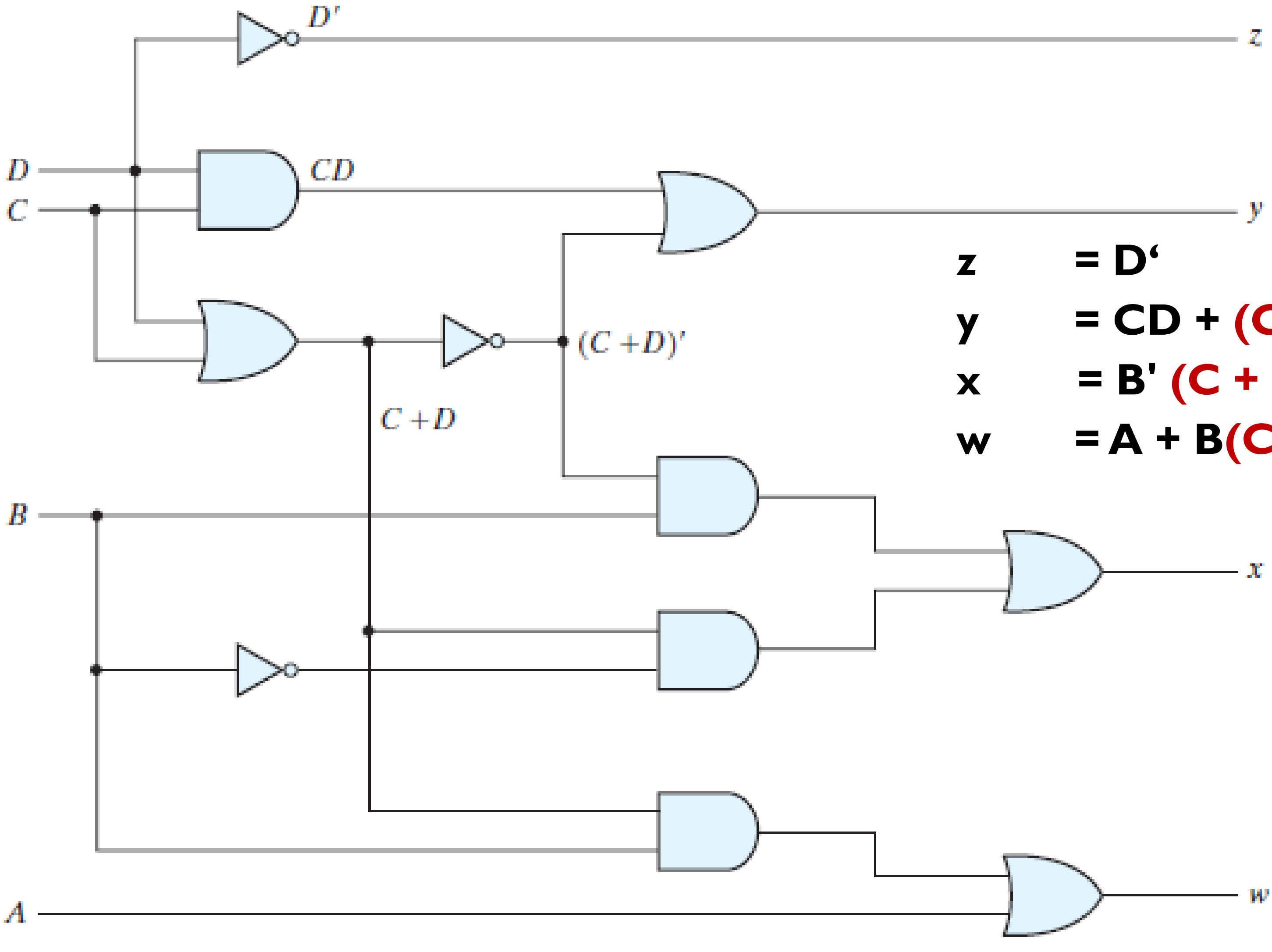
4

$z = D'$	$x = B'C + B'D + BC'D'$ $= B'(C + D) + BC'D'$ $= B'(C + D) + B(C + D)'$
$y = CD + C'D'$ $= CD + (C + D)'$	$w = A + BC + BD$ $= A + B(C + D)$

- **C + D is used to implement the three outputs.**

Code Conversion Example

4



$z = D'$

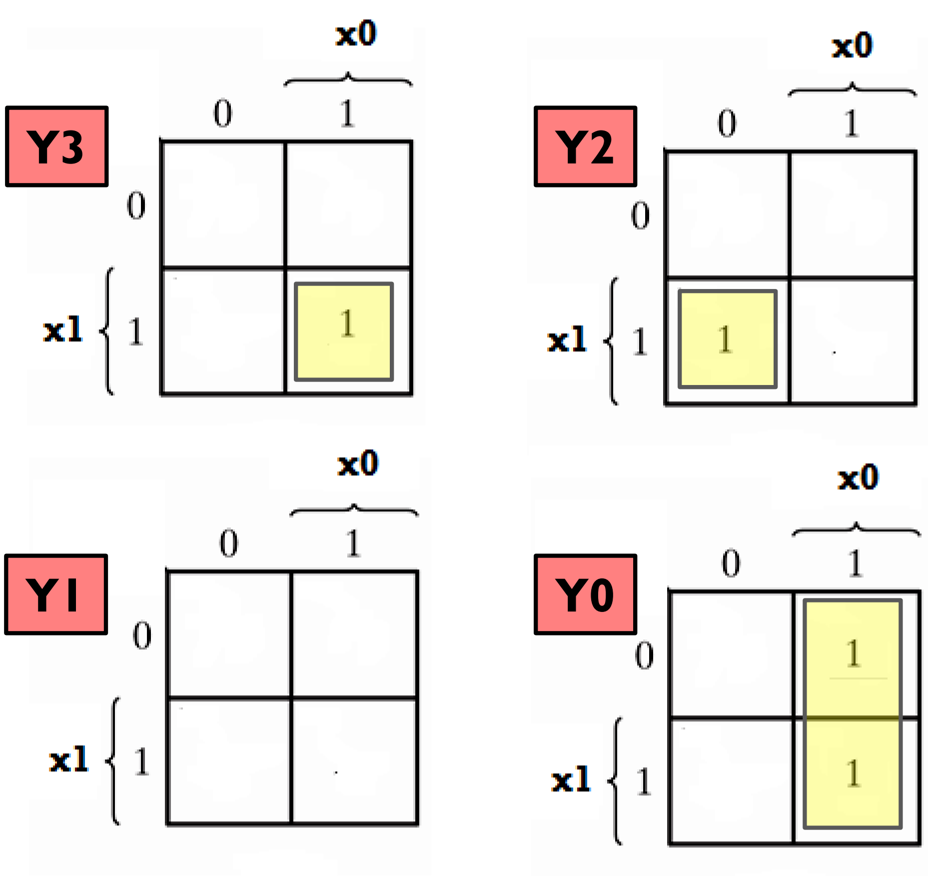
$y = CD + (C + D)'$

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

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

Design Examples

- Design a circuit that takes an input $X = x_1\ x_0$ and calculate the output $Y = X^2$



Input		Output			
X1	X0	Y3	Y2	Y1	Y0
0	0	0	0	0	0
0	1	0	0	0	1
1	0	0	1	0	0
1	1	1	0	0	1

$Y3 = x_1x_0$

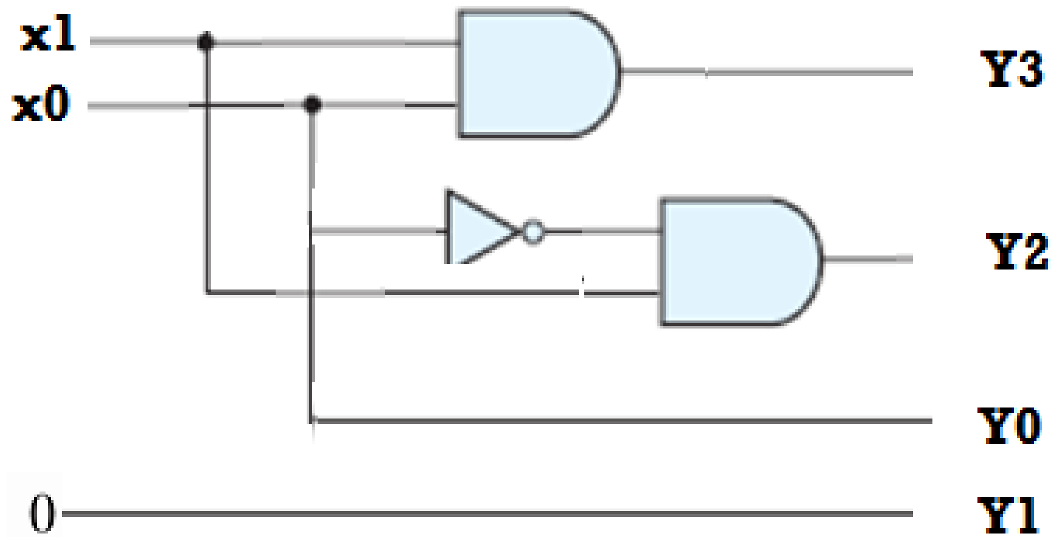
$Y2 = x_1x'_0$

$Y1 = 0$

$Y1 = x_0$

Design Examples

- Design a circuit that takes an input $X = x_1\ x_0$ and calculate the output $Y = X^2$



$Y3 = x_1x_0$

$Y2 = x_1x'_0$

$Y1 = 0$

$Y1 = x_0$

Design Examples

- Design a circuit that takes an input N = $n_2\ n_1\ n_0$ and calculates the output M, where M is calculated as the following:

$$M = \begin{cases} 2N & 0 \le N \le 3 \\ N+1 & 4 \le N < 7 \end{cases}$$

Input			Output		
n2	**n1**	**n0**	M2	M1	M0
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**1**	**0**	**1**	**0**
0	**1**	**0**	**1**	**0**	**0**
0	**1**	**1**	**1**	**1**	**0**
1	**0**	**0**	**1**	**0**	**1**
1	**0**	**1**	**1**	**1**	**0**
1	**1**	**0**	**1**	**1**	**1**
1	**1**	**1**	**x**	**x**	**x**

Design Examples

- Design a circuit that takes an input N = $n_2\ n_1\ n_0$ and calculates the output M, where M is calculated as the following:

$$M = \begin{cases} 2N & 0 \le N \le 3 \\ N+1 & 4 < N < 7 \end{cases}$$

M2

n2 \ n1 n0	00	01	11	10
0			1	1
1	1	1	x	1

M0

n2 \ n1 n0	00	01	11	10
0				
1	1		x	1

M1

n2 \ n1 n0	00	01	11	10
0		1	1	
1		1	x	1

$M_2 = n_2 + n_1$

$M_1 = n_0 + n_2 n_1$

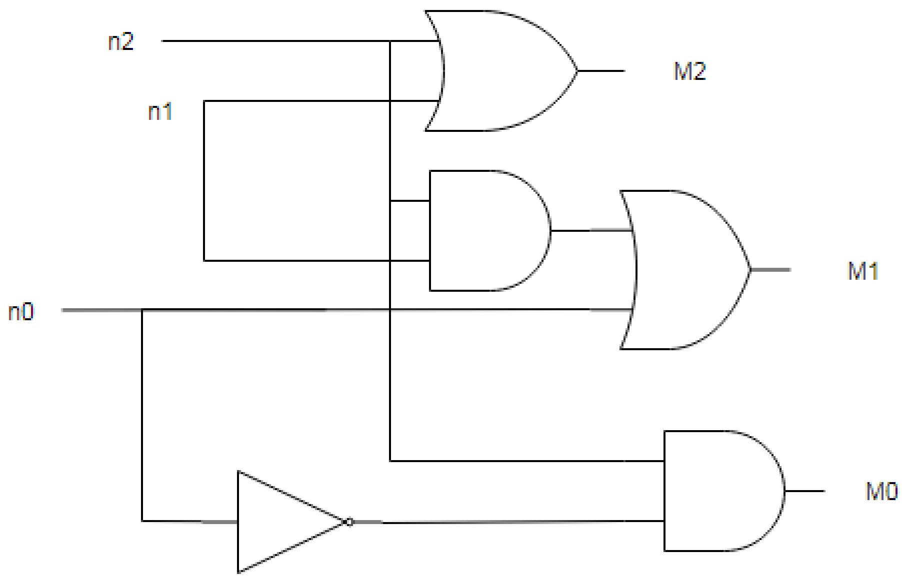
$M_0 = n_2 n'_0$

Output		
M2	M1	M0
0	0	0
0	1	0
1	0	0
1	1	0
1	0	1
1	1	0
1	1	1
x	x	x

Design Examples

- Design a circuit that takes an input $N = n_2\ n_1\ n_0$ and calculates the output M, where M is calculated as the following:

$$M = \begin{cases} 2N & 0 \le N \le 3 \\ N+1 & 4 \le N < 7 \end{cases}$$



$M_2 = n_2 + n_1$

$M_1 = n_0 + n_2 n_1$

$M_0 = n_2 n'_0$

Design Examples

- Design a code converter that converts a decimal digit from, The 8, 4, –2, –1 code to BCD

Table 1.5
Four Different Binary Codes for the Decimal Digits

Decimal Digit	BCD 8421	2421	Excess-3	8, 4, −2, −1
0	0000	0000	0011	0000
1	0001	0001	0100	0111
2	0010	0010	0101	0110
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1011
6	0110	1100	1001	1010
7	0111	1101	1010	1001
8	1000	1110	1011	1000
9	1001	1111	1100	1111
Unused bit combi- nations	1010	0101	0000	0001
	1011	0110	0001	0010
	1100	0111	0010	0011
	1101	1000	1101	1100
	1110	1001	1110	1101
	1111	1010	1111	1110

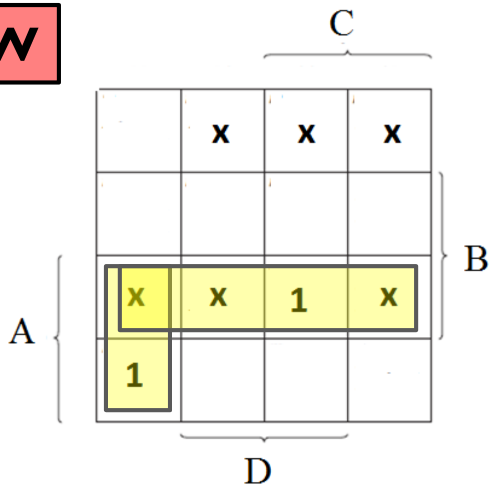
Design Example

- Design a code converter that converts a decimal digit from, The 8, 4, −2, −1 code to BCD

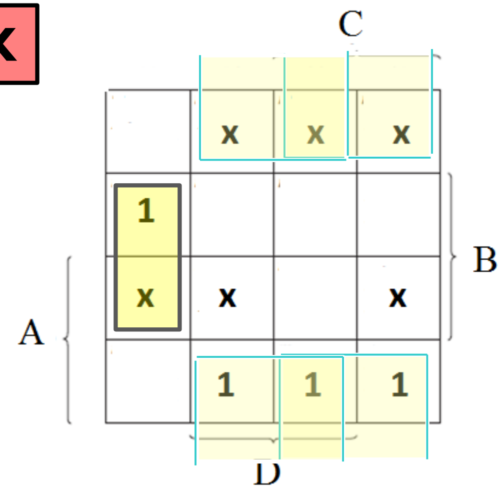
BCD 8421	8, 4, −2, −1
0000	0000
0001	0111
0010	0110
0011	0101
0100	0100
0101	1011
0110	1010
0111	1001
1000	1000
1001	1111
1010	0001
1011	0010
1100	0011
1101	1100
1110	1101
1111	1110

Input 8, 4, −2, −1 code				Output BCD			
A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	0
0	0	0	1	x	x	x	x
0	0	1	0	x	x	x	x
0	0	1	1	x	x	x	x
0	1	0	0	0	1	0	0
0	1	0	1	0	0	1	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	0	1
1	0	0	0	1	0	0	0
1	0	0	1	0	1	1	1
1	0	1	0	0	1	1	0
1	0	1	1	0	1	0	1
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	1	0	0	1

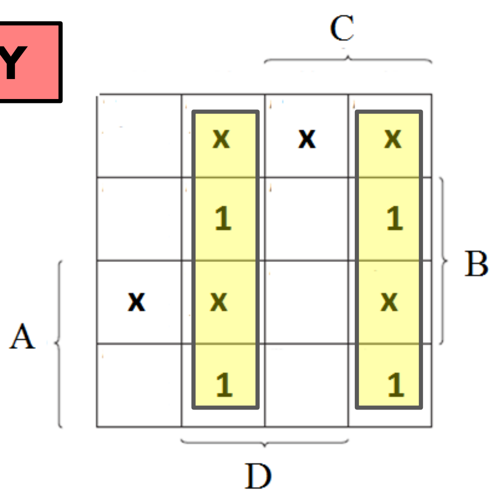
W



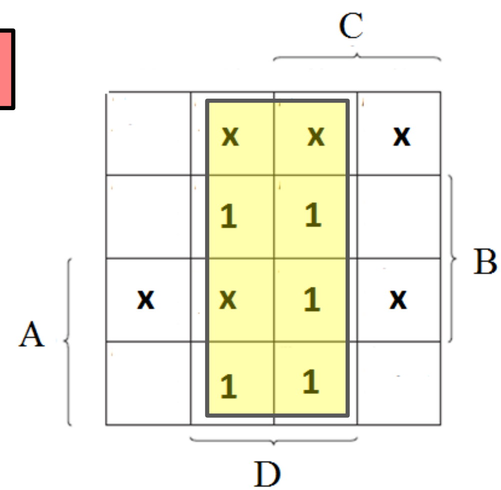
X



Y



Z



W = AB + AC'D'
X = B'C + B'D + B'C'D'
Y = CD' + C'D
Z = D

Output BCD			
W	X	Y	Z
0	0	0	0
x	x	x	x
x	x	x	x
x	x	x	x
0	1	0	0
0	0	1	1
0	0	1	0
0	0	0	1
1	0	0	0
0	1	1	1
0	1	1	0
0	1	0	1
x	x	x	x
x	x	x	x
x	x	x	x
1	0	0	1

Adder

- The most basic arithmetic operation is the addition of two binary digits
  - When both augend and addend bits are equal to 1, the binary sum consists of two digits (1 + 1 = 10)
  - The higher significant bit of this result is called a carry

- A combination circuit that performs the addition of two bits is **half adder**.

- A adder performs the addition of three bits (2 significant bits and a previous carry) is called a **full adder**

Half Adder

- **Inputs**: x and y
- **Outputs**: S (for sum) and C (for carry)
- The simplified Boolean functions for the two inputs can be obtained directly from the truth table.

1

2

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

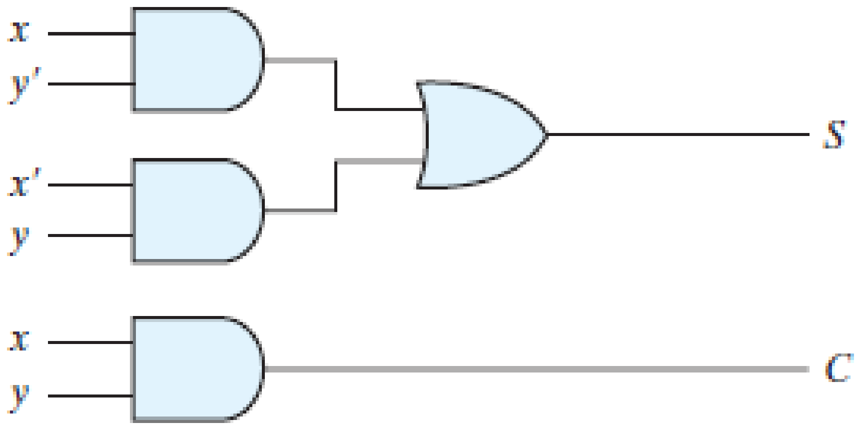
3

$$S = x'y + xy'$$
$$C = xy$$

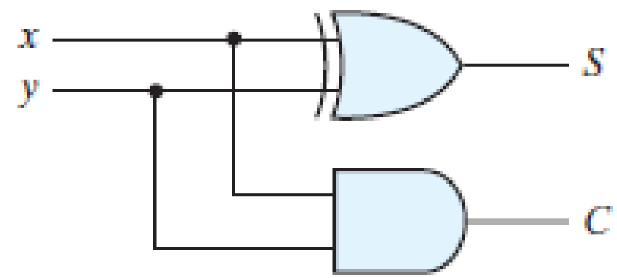
Half Adder

Implementation of Half Adder

4



(a) $S = xy' + x'y$
$C = xy$



(b) $S = x \oplus y$
$C = xy$

Full Adder

Full Adder

1

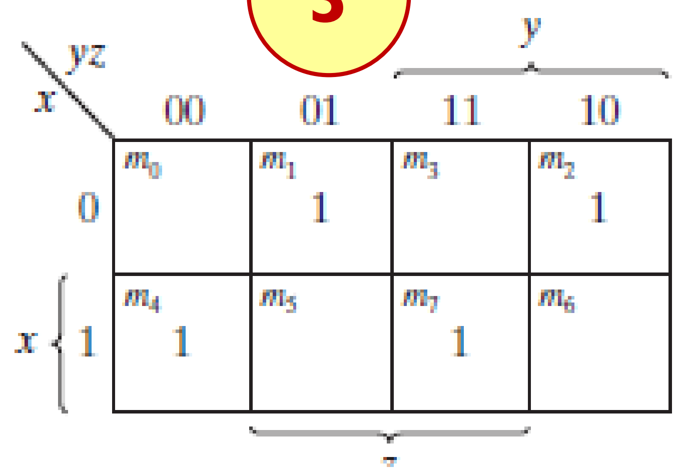
- A full adder is a combinational circuit that forms the arithmetic sum of three input bits
  - It consists of three inputs and two outputs.

2

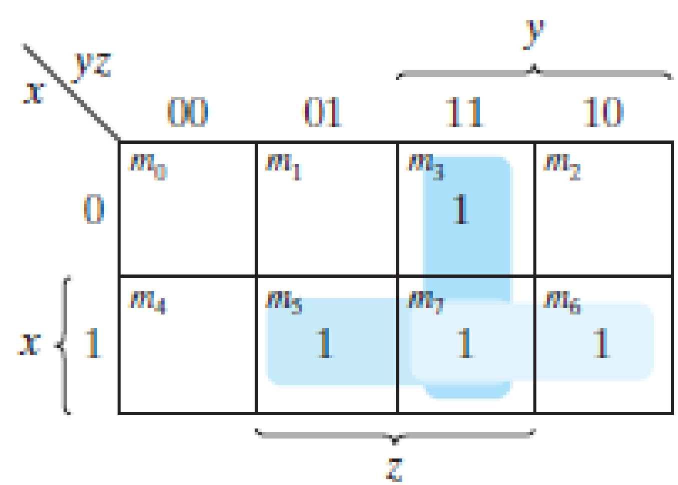
Full Adder

x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

3



(a) $S = x'y'z + x'yz' + xy'z' + xyz$



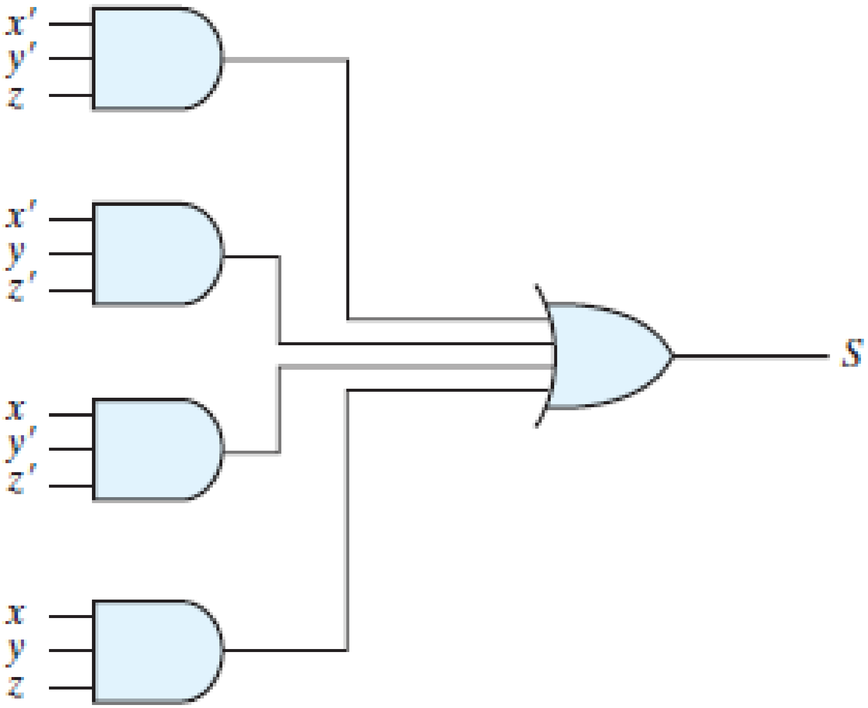
(b) $C = xy + xz + yz$

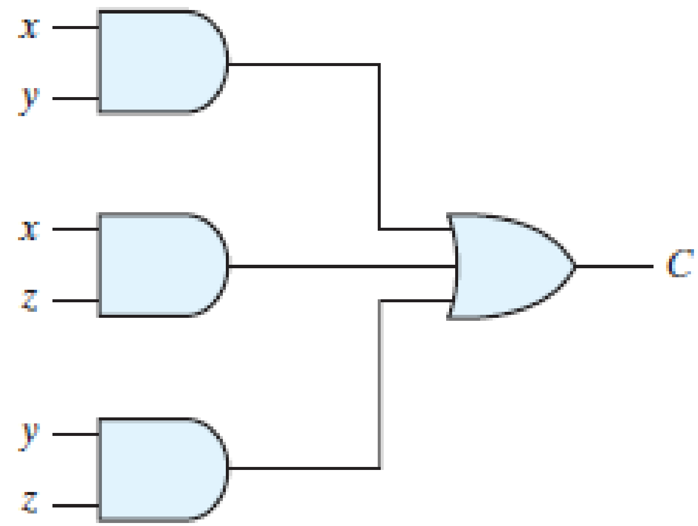
Implementation of Full Adder

(a) $S = x'y'z + x'yz' + xy'z' + xyz$

4

(b) $C = xy + xz + yz$

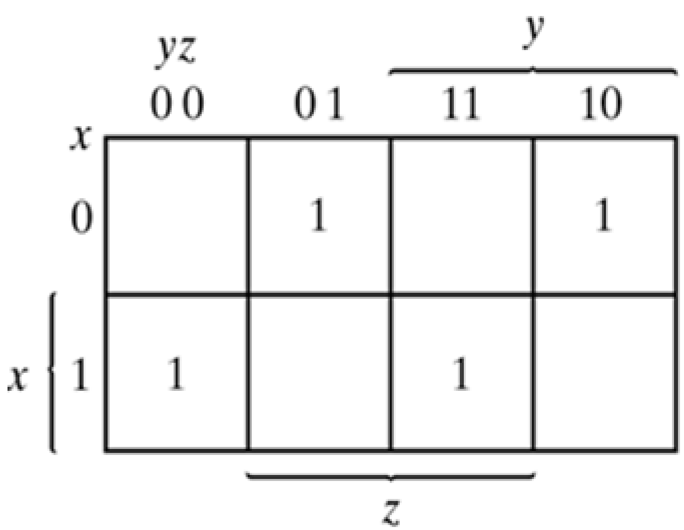




Full Adder

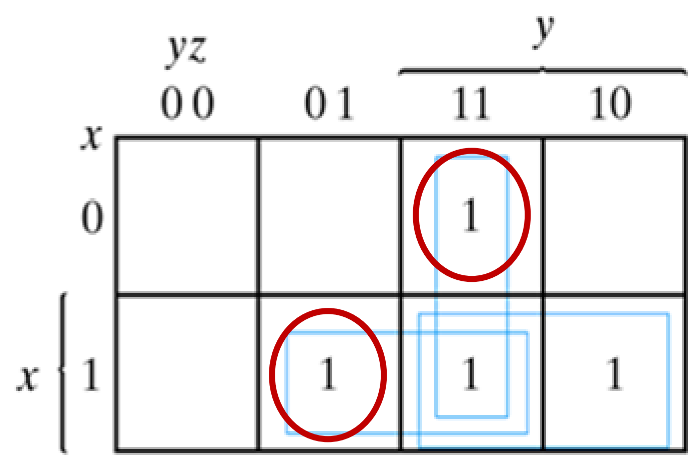
- A full adder is a combinational circuit that forms the arithmetic sum of three input bits
  - It consists of three inputs and two outputs.

2



$$S = x'y'z + x'yz' + xy'z' + xyz$$

$$= (x \oplus y) \oplus z$$



$$C = xy + xz + yz$$

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

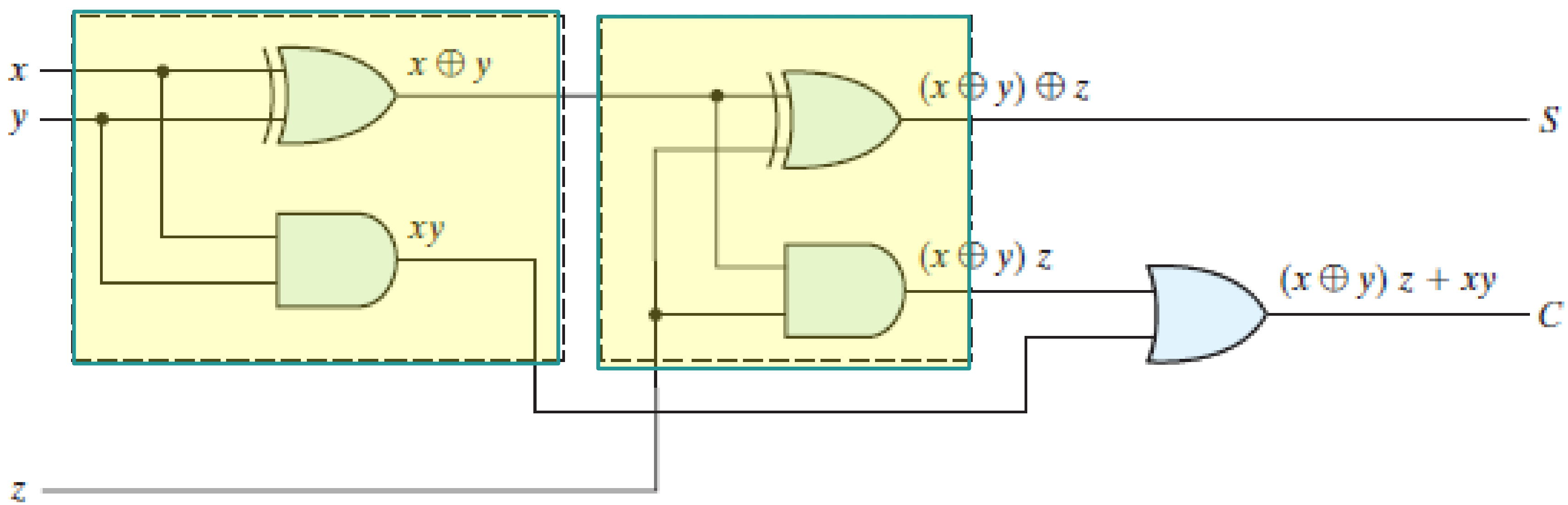
$$= xy + (x \oplus y)z$$

Implementation of Full Adder

- A full adder can be implemented with **two half adders and an OR gate**

$$S = x'y'z + x'yz' + xy'z' + xyz$$
$$= (x \oplus y) \oplus z$$

$$C = xy + xz + yz$$
$$= xy + xy'z + x'yz$$
$$= xy + (x \oplus y)z$$



4 Bits Binary Parallel Adder

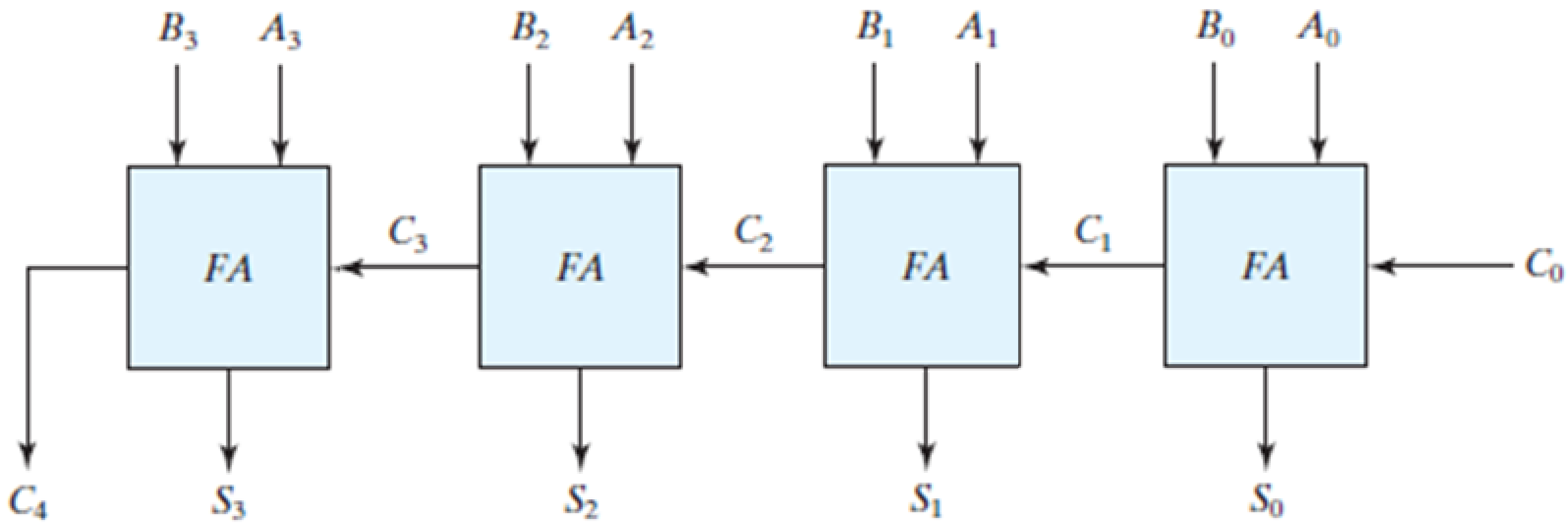
Binary Parallel Adder

- A binary adder produces the arithmetic sum of two binary numbers in parallel.
- Consider two binary number: A = 1011 and B = 0011

Subscript i :	3	2	1	0	
Input carry	0	1	1	0	C_i
Augend	1	0	1	1	A_i
Addend	0	0	1	1	B_i
Sum	1	1	1	0	S_i
Output carry	0	0	1	1	C_{i+1}

- **The output carry from each full adder is connected to input carry of the next full adder in the chain.**
- **An n-bit parallel adder requires n full-adder**
- **4-bit adder: Interconnection of four full adder (FA) circuits.**

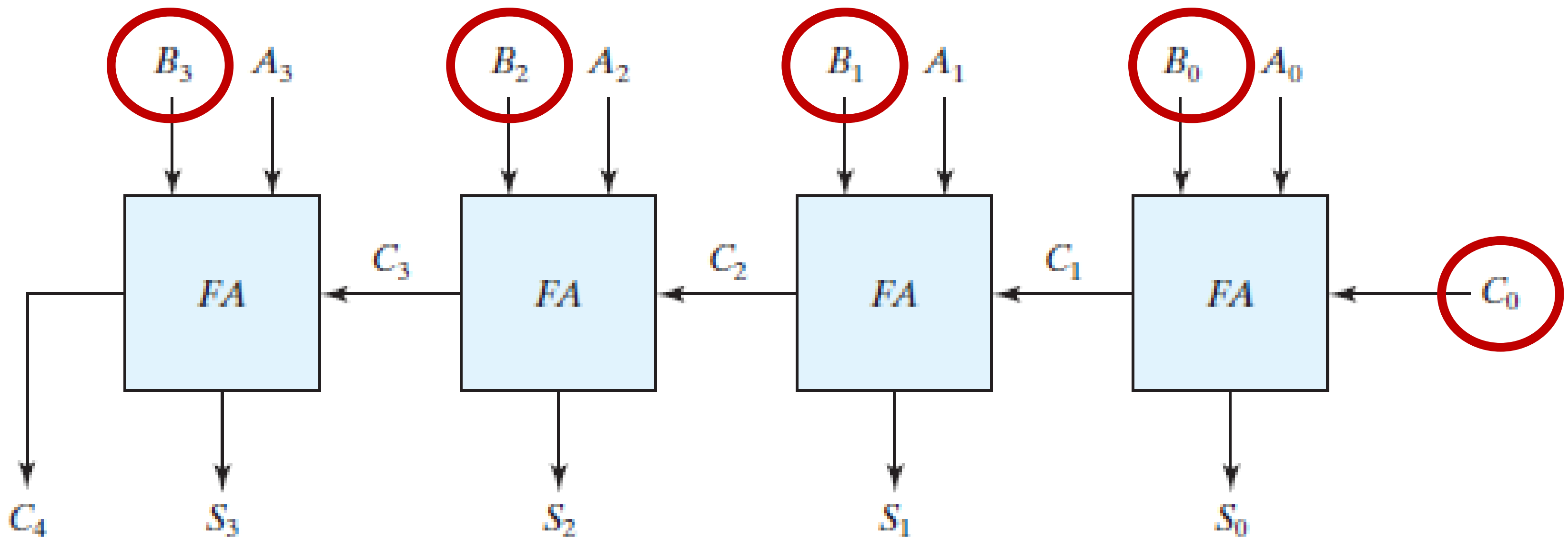
4-bit Adder Example



Subscript i :	3	2	1	0	
Input carry	0	1	1	0	C_i
Augend	1	0	1	1	A_i
Addend	0	0	1	1	B_i
Sum	1	1	1	0	S_i
Output carry	0	0	1	1	C_{i+1}

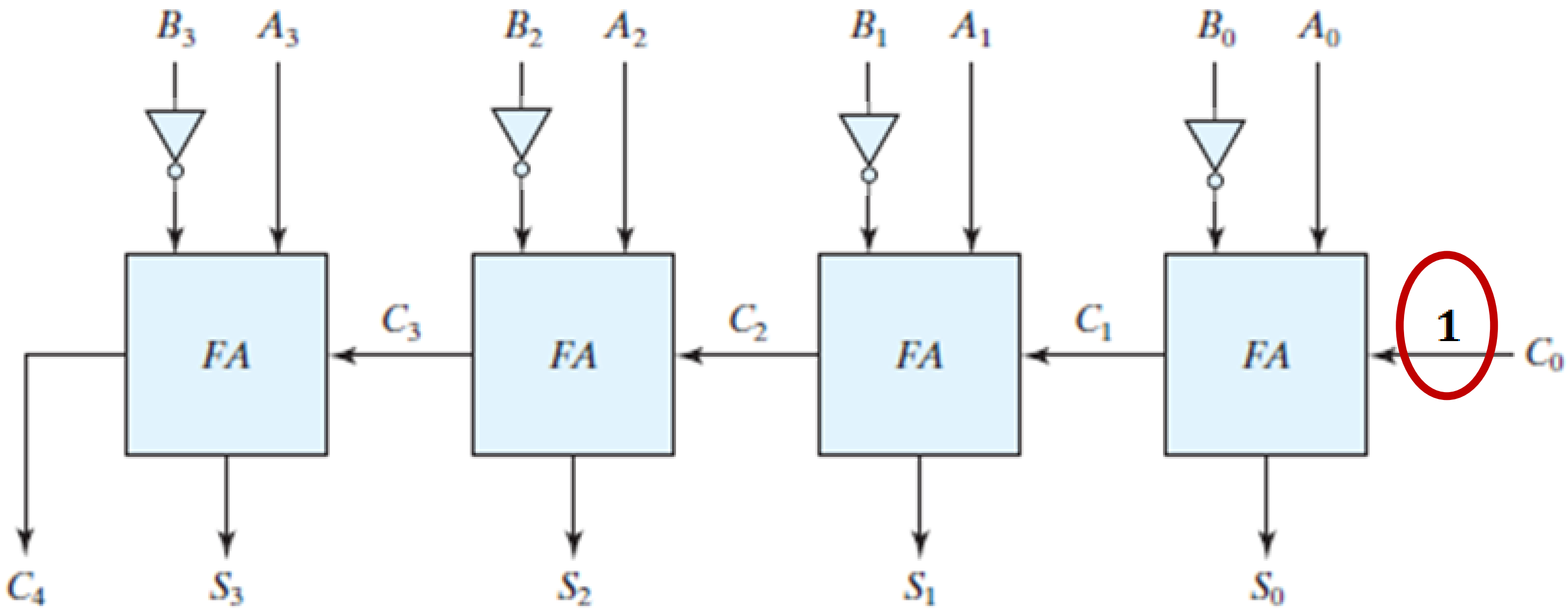
4-bit Binary Subtractor

4-bit Adder Example



Can you think about 4-bits binary subtractors ???

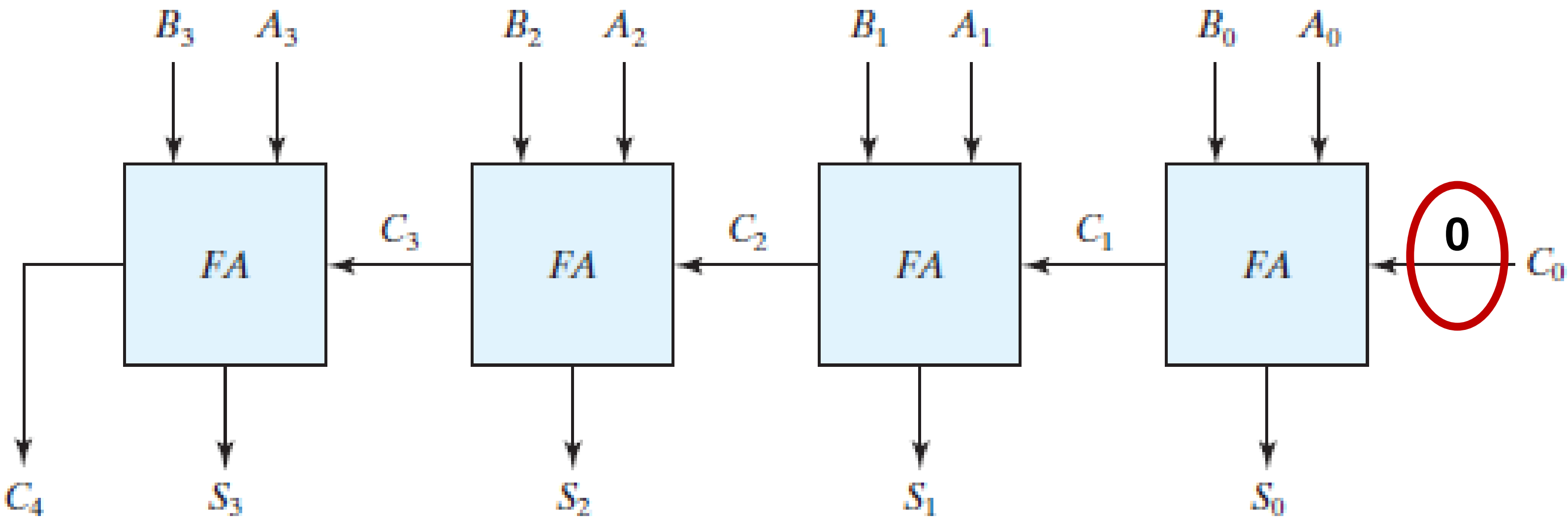
4-bit Subtractor Example



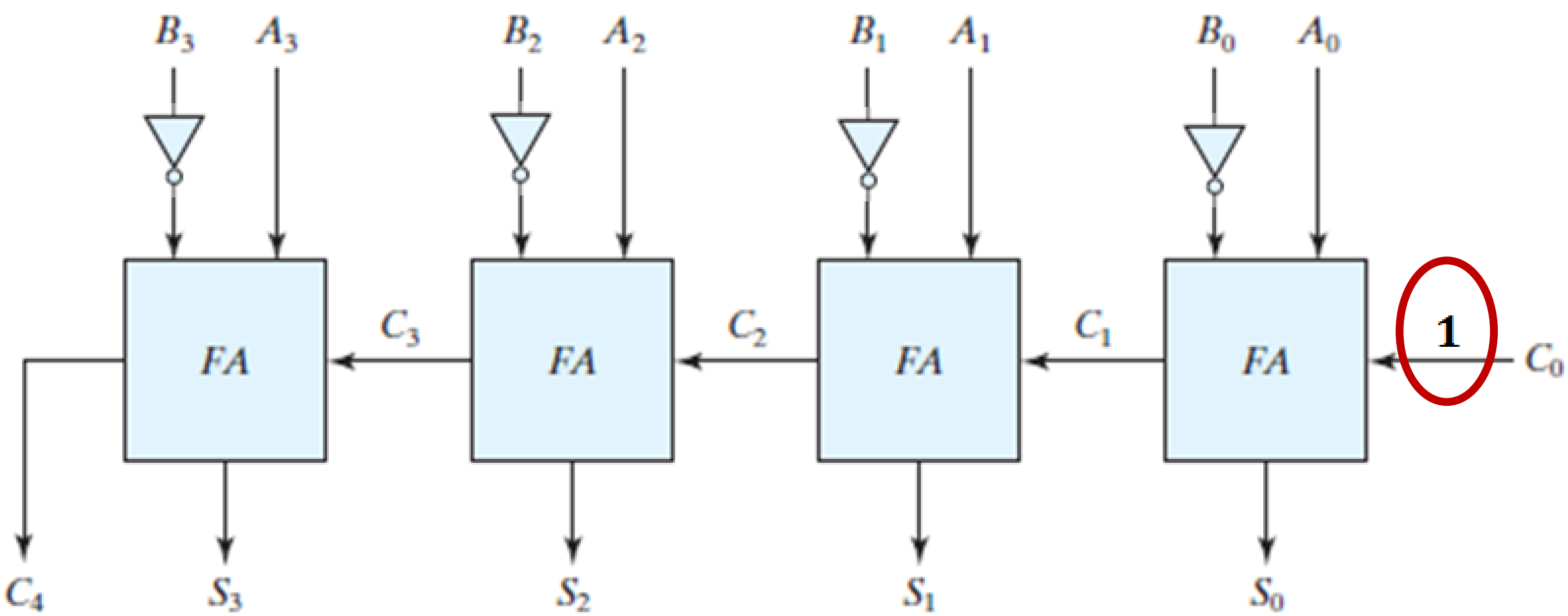
Can you think about 4-bits binary Adder - subtractors ???

4-bit Adder-Subtractor

Adder



Subtractor

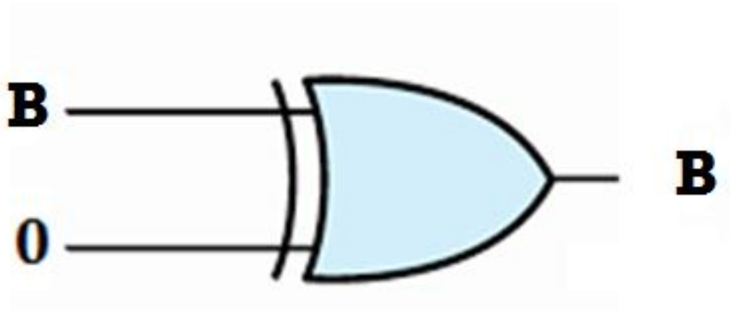


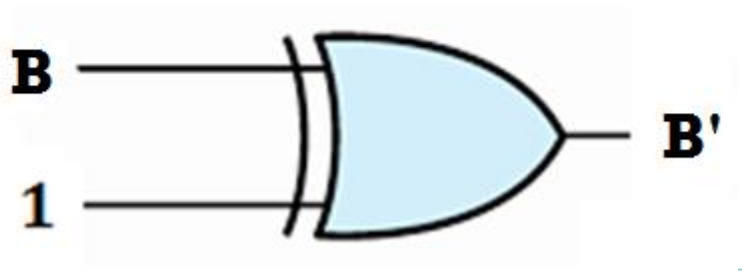
4-bit Adder-Subtractor

Function	C_0	Opr.
Adder	0	A+B
Subtractor	1	A+B’

Recall some XOR properties

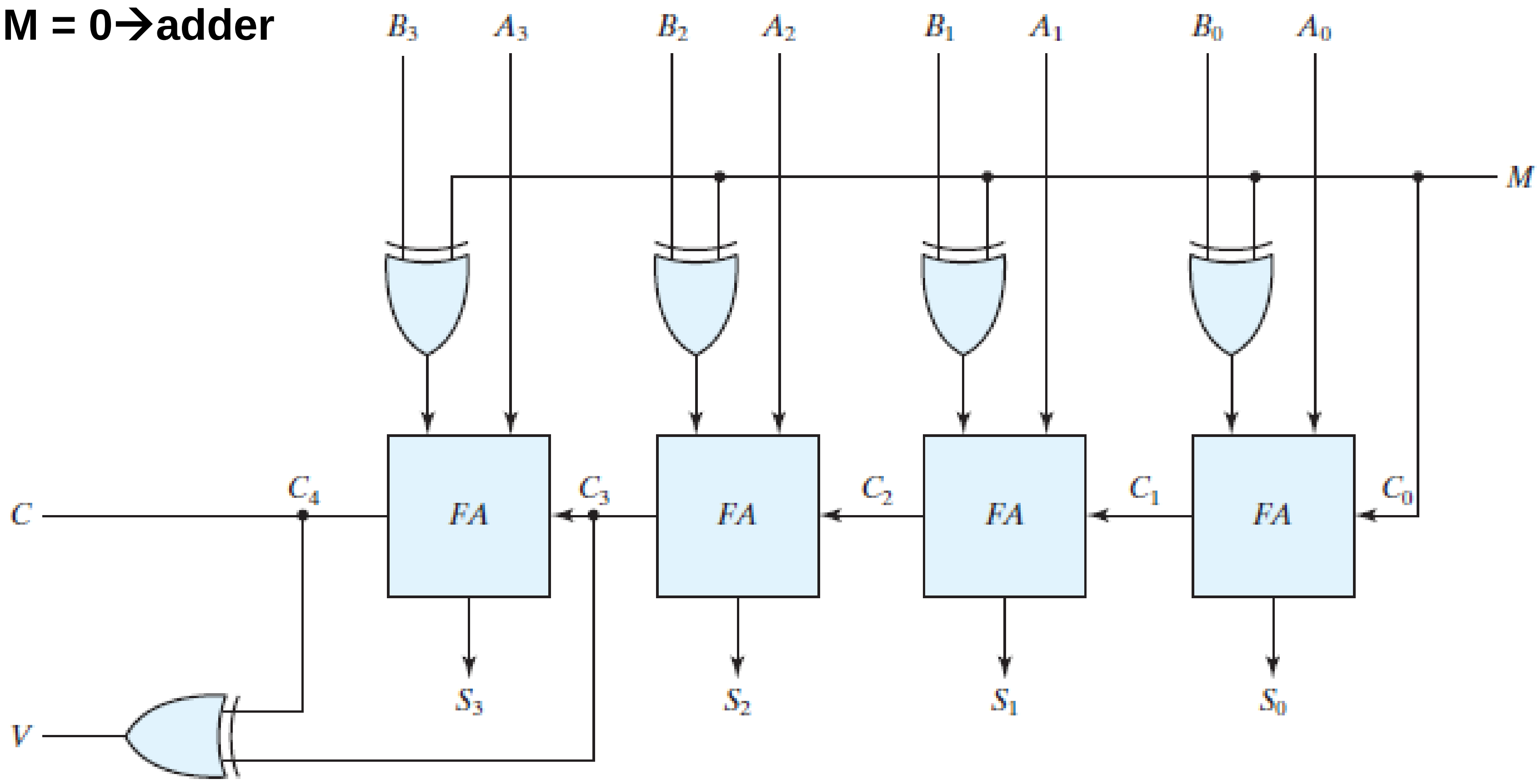
XOR Some identities	$B \oplus 0 = B$ $B \oplus 1 = B'$





4-bit Adder-Subtractor Example

M = 1→subtractor
M = 0→adder



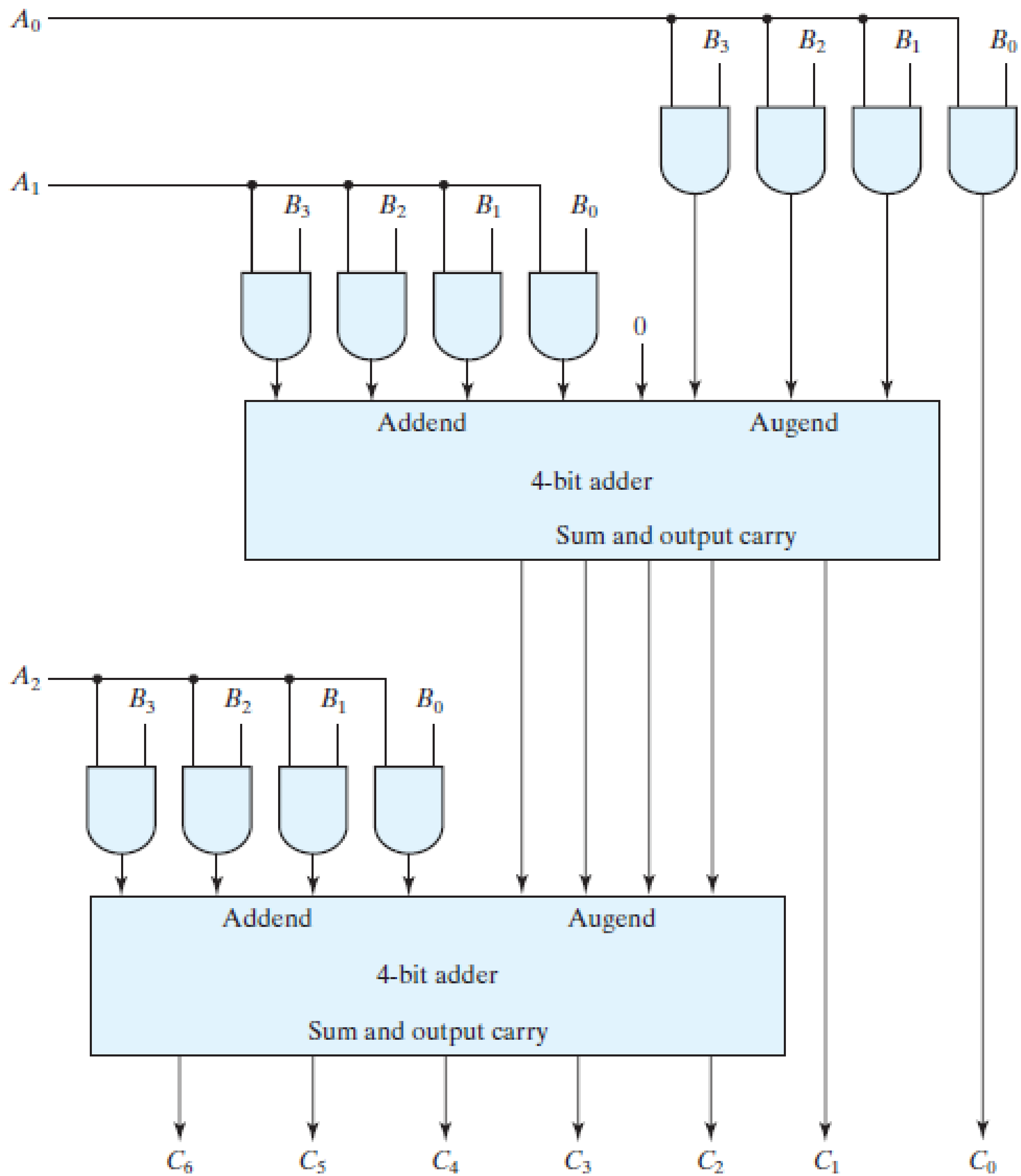
Overflow is a problem in digital computers because the number of bits that hold the number is finite and a result that contains n+1 bits cannot be accommodated.

Binary Multiplier

- Multiplication of binary numbers is performed in the same way as multiplication of decimal numbers.
- The multiplicand is multiplied by each bit of the multiplier, starting from the least significant bit.

Four-bit by three-bit binary multiplier

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



Magnitude Comparator

- The comparison of two numbers is an operation that determines whether one number is greater than, less than, or equal to the other number.

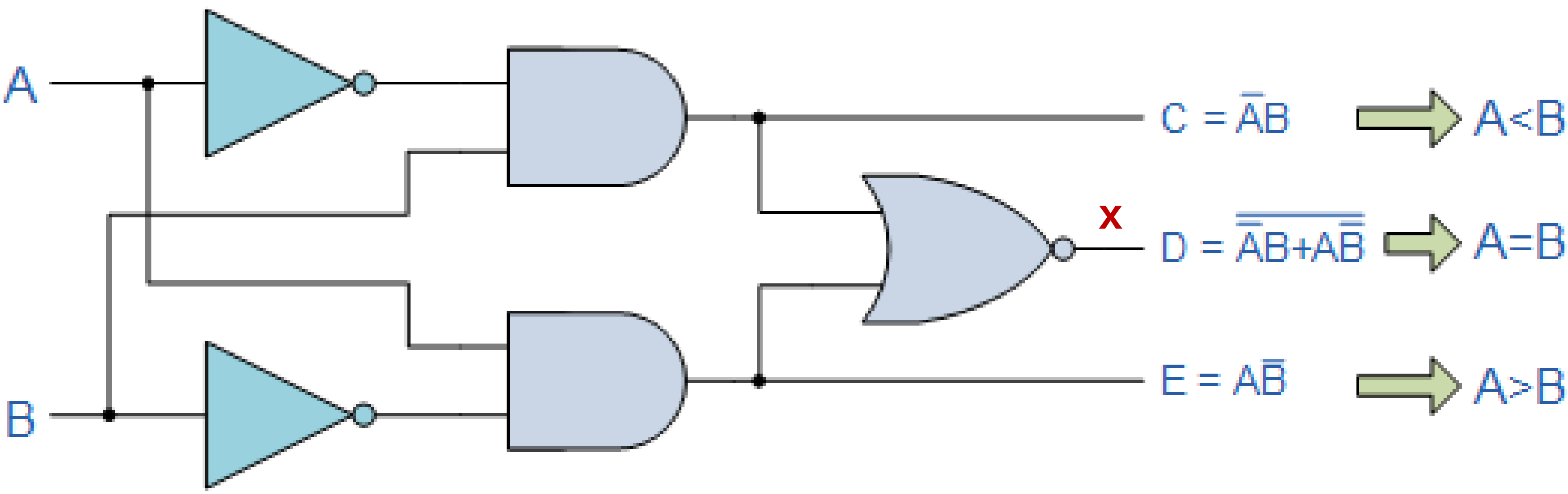
- A *magnitude comparator* is a combinational circuit that compares two numbers A and B and determines their relative magnitudes.

- The outcome of the comparison is specified by three binary variables that indicate whether
  - $A > B$,
  - $A = B$, or
  - $A < B$.

Magnitude Comparator

A	B	A<B	A=B	A>B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

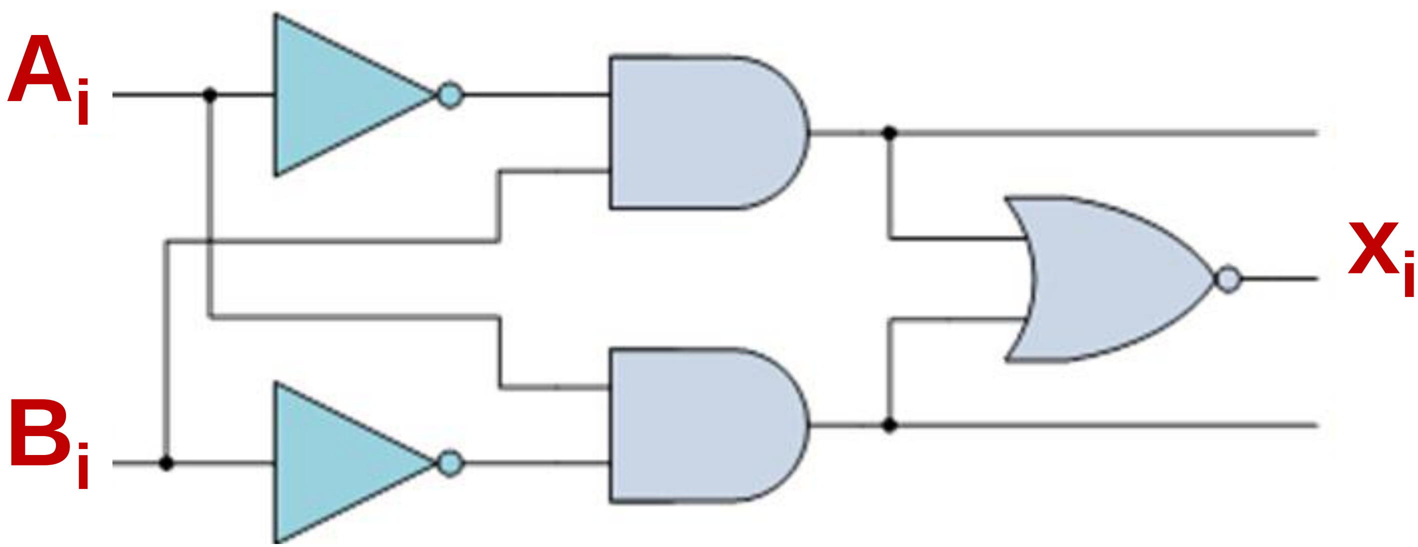
$x = (A'B + AB')'$
where x= 1 only if the pair of bits A and B are equal



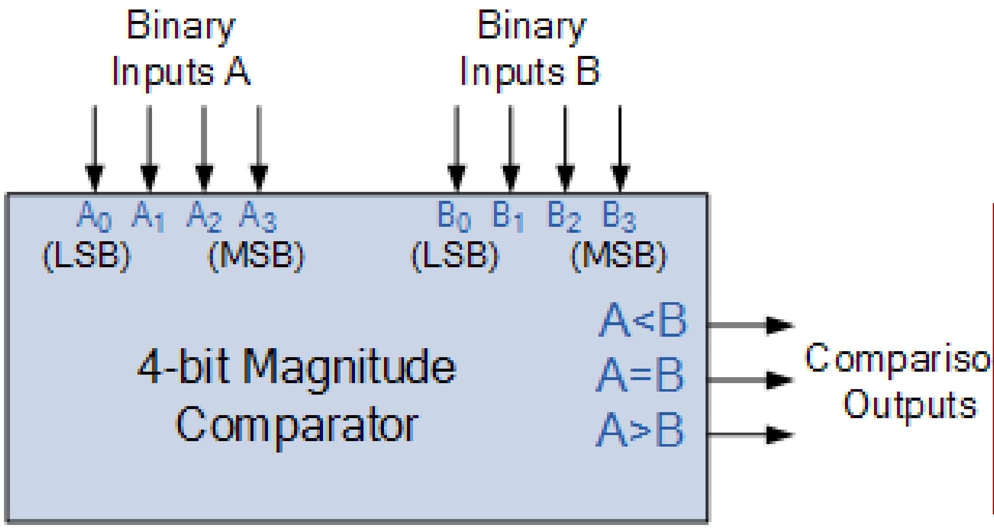
Magnitude Comparator

$$x_i = (A'_i B_i + A_i B'_i)' \text{ for } i = 0, 1, 2, 3$$

where $x_i = 1$ only if the pair of bits in position i are equal (i.e., if both are 1 or 0).



4 Bits Magnitude Comparator



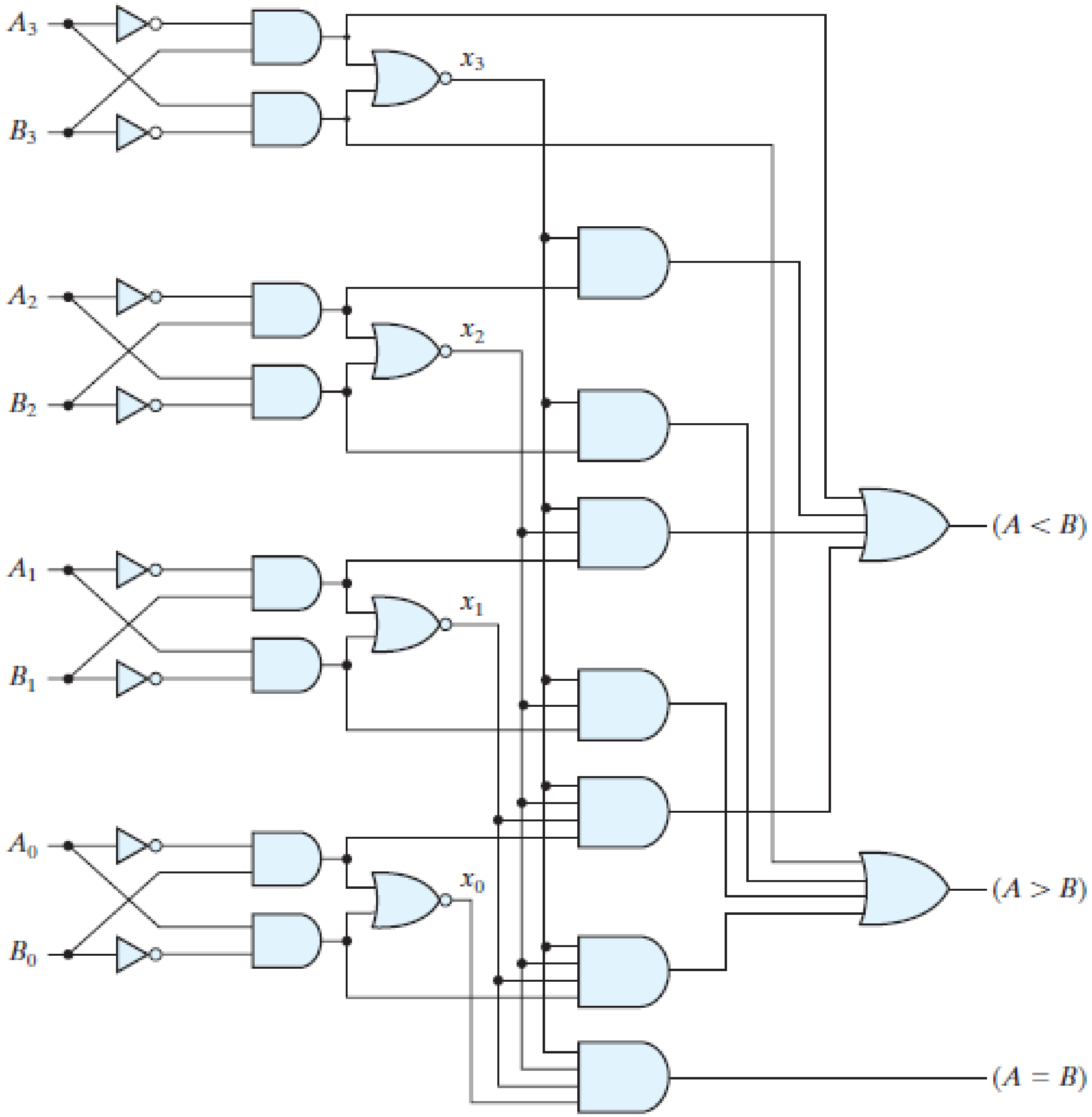
x_i = $(A'_iB_i + A_iB'_i)'$ for i = 0, 1, 2, 3
where x_i = 1 only if the pair of bits in position i are equal (i.e., if both are 1 or 0).

$$(A = B) = x_3x_2x_1x_0$$

$$(A > B) = A_3B'_3 + {\color{red}x_3}\,A_2B'_2 + {\color{red}x_3x_2}A_1B'_1 + {\color{red}x_3x_2x_1}A_0B'_0$$

$$(A < B) = A'_3B_3 + {\color{red}x_3}A'_2B_2 + {\color{red}x_3x_2}A'_1B_1 + {\color{red}x_3x_2x_1}A'_0B_0$$

4 Bits Magnitude Comparator




Thank You!