CS 221
Logic Design

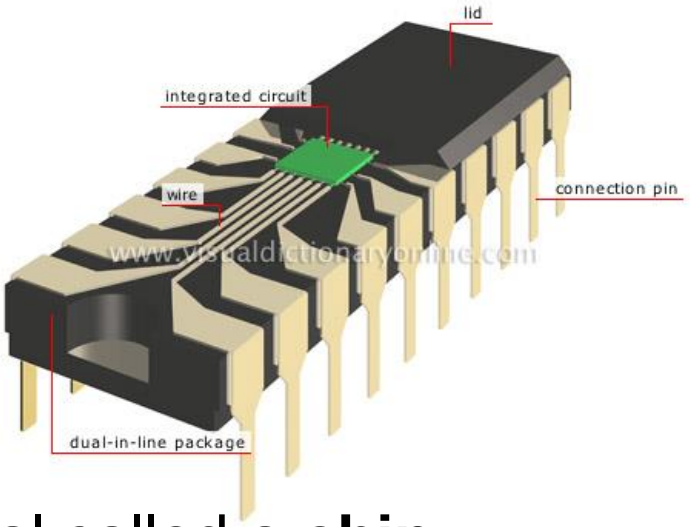
Fall 2021

By Wessam El-Behaidy & Salwa Osama

Analysis and Design of Combinational Circuit

Lecture 5

Integrated Circuits



- Abbreviated IC
- Is a silicon semiconductor crystal called a **chip**
- The various gates are interconnected inside the chip to form the required circuit
- The chip is mounted in a ceramic or plastic container and connections are welded to external pins to form the integrated circuits.
- Number of pins range from 14 on small IC package to several thousand on a larger package.

Levels of Integration

- Digital ICs are often categorized according to the number of gates in a single package:
  - Small-scale integration (SSI)
    - gates <10
  - Medium-scale integration (MSI)
    - 10-1000 gates, elementary digital operations
    - As decoders, adders, multiplexers, registers and counters
  - Large-scale integration (LSI)
    - Thousands of gates
    - Digital systems: processors, memory chips, and programmable logic devices.
  - Very large-scale integration (VLSI)
    - Hundred of thousands
    - Large memory arrays and complex microcomputer chips.

Standard combinational circuits

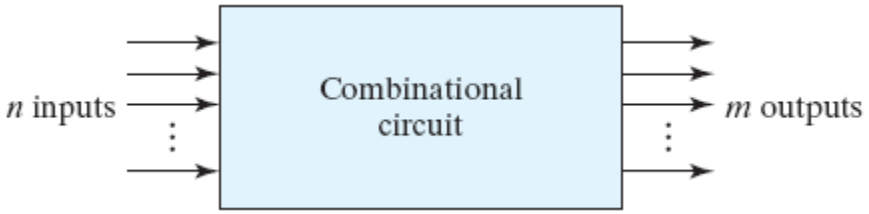
- Several combinational circuits that perform specific digital functions commonly needed in the design of digital systems.
- These components are available in integrated circuits as *standard components* (i.e. MSI circuits).
- The most important standard combinational circuits:
  - Adders
  - Subtractors
  - Comparators
  - Decoders
  - Encoders
  - Multiplexers
- Also, they are used as standard cells in complex VLSI

Current Chapter Purpose

- The purpose of the current chapter is:

  To use the knowledge acquired in previous chapters to formulate systematic analysis and design procedures for combinational circuits.

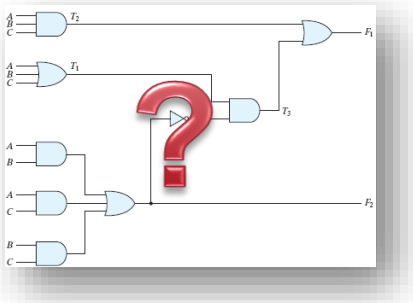
- So, **3 tasks** are applied on combinational circuits used as examples:
  - Analyze the behavior of a given logic circuit
  - Design a circuit that will have a given behavior
  - Write HDL models for some.



COMBINATIONAL CIRCUITS

- A combinational circuit consists of:
  - Input variable
  - Logic gates
  - Output variable
- For **n** input variables,
  - there are $\mathbf{2^n}$ possible binary input combinations
  - For each input combination, there is one possible output variable
- Thus, combinational circuit can be specified with a *truth table* or *Boolean function*

Analysis procedure



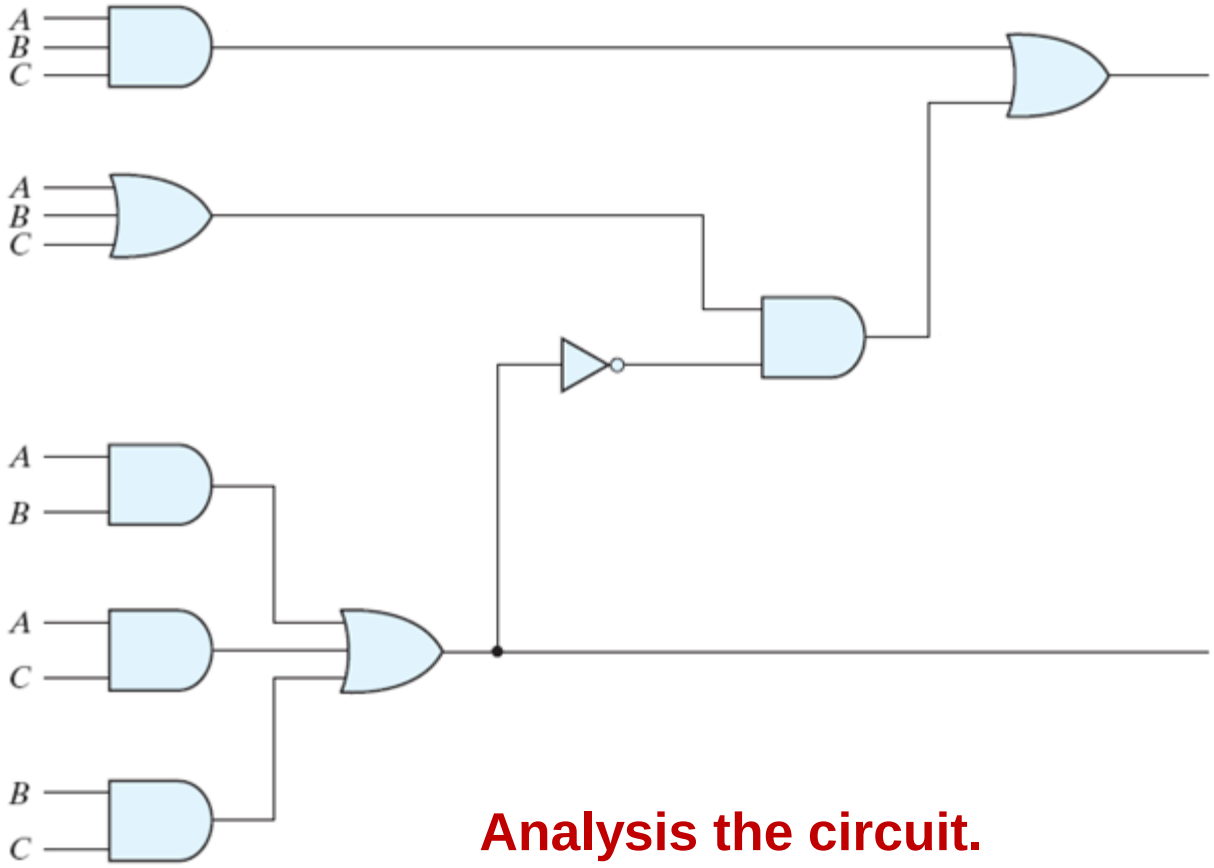
- **Analysis of a combinational circuit is:**

  the determination of the function that the circuit implements

- **The analysis can be performed:**
  - Manually by finding:
    1. the Boolean functions or
    2. truth table
  - Using a computer simulation program

- ✓ Make sure that the given circuit is combinational
- ✓ Proceed to obtain the output Boolean functions or truth table

EXAMPLE

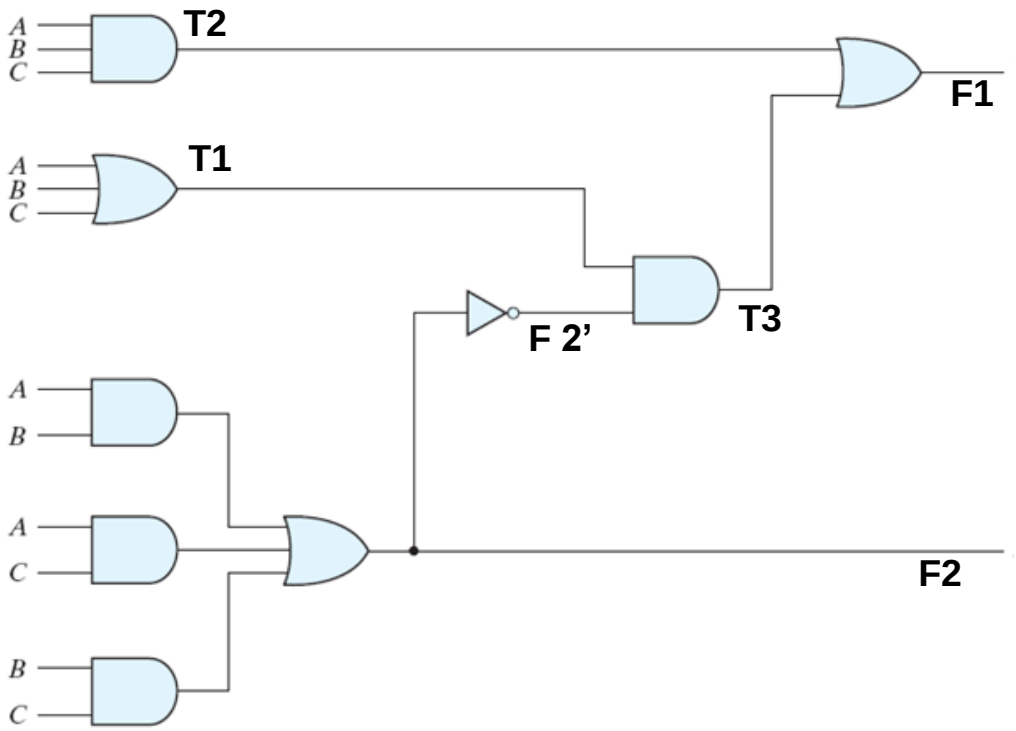


Analysis the circuit.
= What is its function?
= What is the output of this circuit?

EXAMPLE (CONT.)

T1 = A + B + C
T2 = ABC
F2= AB + AC + BC

T3=F2' T1
F1=T3 + T2



1

To obtain the **output Boolean function** from **a logic diagram**:

1. Label all gate outputs that are function of input variables. Determine the Boolean functions for each gate
2. Label the gates that are function of input variables and previously labeled gates. Determine the Boolean functions for these gates
3. Repeat the last process until the outputs of the circuit are obtained.
4. By repeated substitution , obtain the output functions in terms of input variables

EXAMPLE (CONT.)

T1 = A + B + C
T2 = ABC
F2= AB + AC + BC

T3=F'2 T1
F1=T3 + T2

- By substitution,

  F1 = T3 + T2

  = F2' T1 + ABC

  = (AB +AC + BC)' (A + B +C) +ABC

  = (A' +B')(A' +C')(B' + C') (A + B +C) +ABC (x+y)(x+z)=x+yz

  = (A' +B'C')(AB' +AC' +BC'+B'C) + ABC x'. x=0 , x+0=x

  = A'BC' +A'B'C +AB'C' +ABC **We can stop here**

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

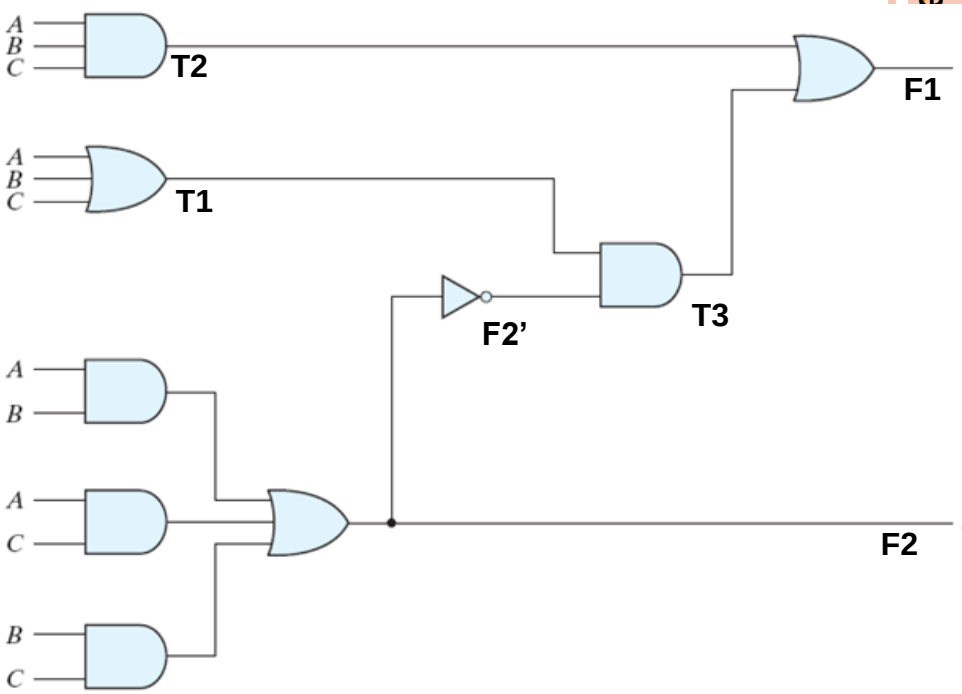
  =A'(B ⊕ C) +A (B ⊕ C)' **B ⊕ C= Z**

  =A'Z+ AZ' =A ⊕ B ⊕ C

- Finding Boolean representation of a circuit doesn't provide insight into its behavior
- We will observe that this circuit is a *full adder* circuit.

EXAMPLE(CONT.)

A	B	C	T_1	T_2	F_2	F_2'	T_3	F_1
0	0	0	0	0	0	1	0	0
0	0	1	1	0	0	1	1	1
0	1	0	1	0	0	1	1	1
0	1	1	1	0	1	0	0	0
1	0	0	1	0	0	1	1	1
1	0	1	1	0	1	0	0	0
1	1	0	1	0	1	0	0	0
1	1	1	1	1	1	0	0	1



2

To obtain the **truth table** from **a logic diagram**:

1. Determine the number of input variables in the circuit. For n inputs, form the 2^n possible input combinations
2. Label the output of selected gates with arbitrary symbols
3. Obtain the truth table for the outputs of those gates which are a function of previously defined values .

EXAMPLE(CONT.)

- Another way of analyzing a combinational circuit is by means of logic simulation.
- In sec. 4.12, logic simulation and verification of the circuit is demonstrated using Verilog HDL.

Design Procedure

The design of combinational circuits:

- Starts from:
  - the specification of the design objective
- Ends with:
  - a logic diagram or
  - a set of Boolean functions from which logic diagram is obtained

DESIGN PROCEDURE(CONT.)

The design procedure involves the following steps:

- From the specification, determine
  - **number of inputs and outputs** and assign a symbol to each.
- Derive **the truth table**, that
  - Relate the input to the output
- Obtain the **simplified Boolean functions** for each output as a function of input
- Draw **logic diagram** and verify it (manually or by simulation)

Code Conversion Example

- Given specification:

  The use of different codes by different digital system, a conversion circuit is needed to be inserted between 2 systems.

  Design a *code converter circuit* that converts binary coded decimal (BCD) to the excess-3 code for the decimal digits.

- Following the design procedure:

  1. Determine #inputs and outputs:

  Since both codes use 4 bits to represent a decimal digit

  ➔ we need 4 input variables and 4 output variables

Code Conversion Example (Cont.)

- Following the design procedure:
  2. Perform the truth table

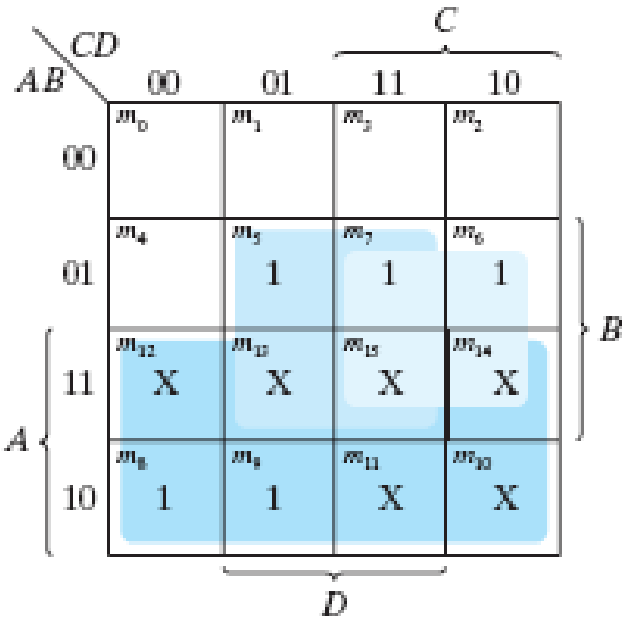
Table 4.2
Truth Table for Code-Conversion Example

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

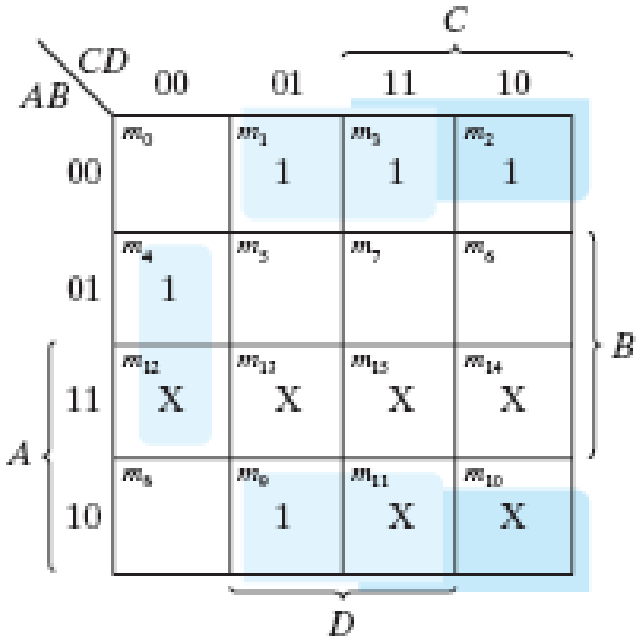
Note: We only use 10 combinations out of 16, so the rest will be don't care conditions

Code Conversion Example (Cont.)

- Following the design procedure:
  3. Simplify each output Boolean function



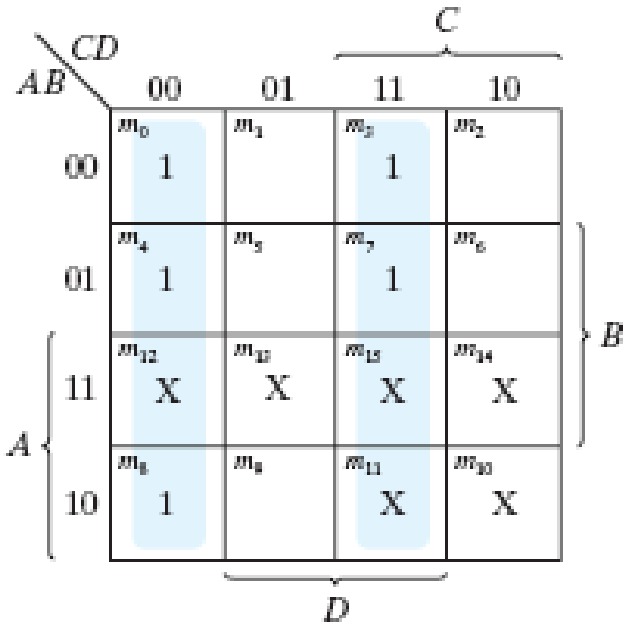
$w = A + BC + BD$



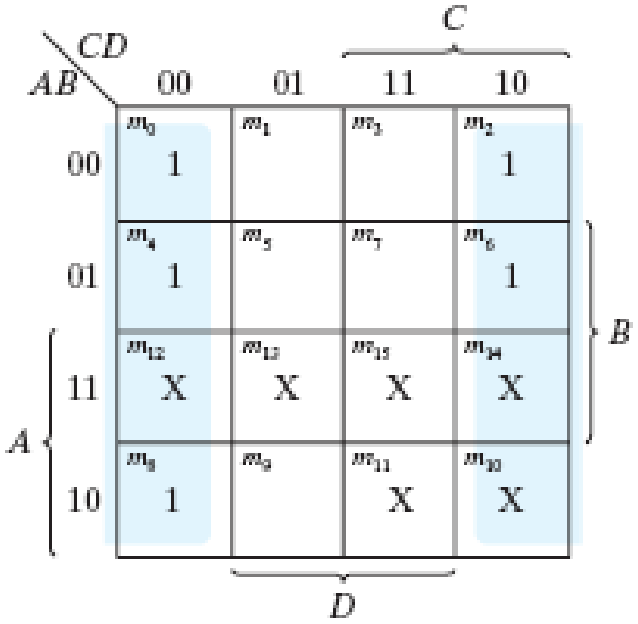
$x = B'C + B'D + BC'D'$

Code Conversion Example (Cont.)

- Following the design procedure:
  3. Simplify each output Boolean function



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



$$z = D'$$

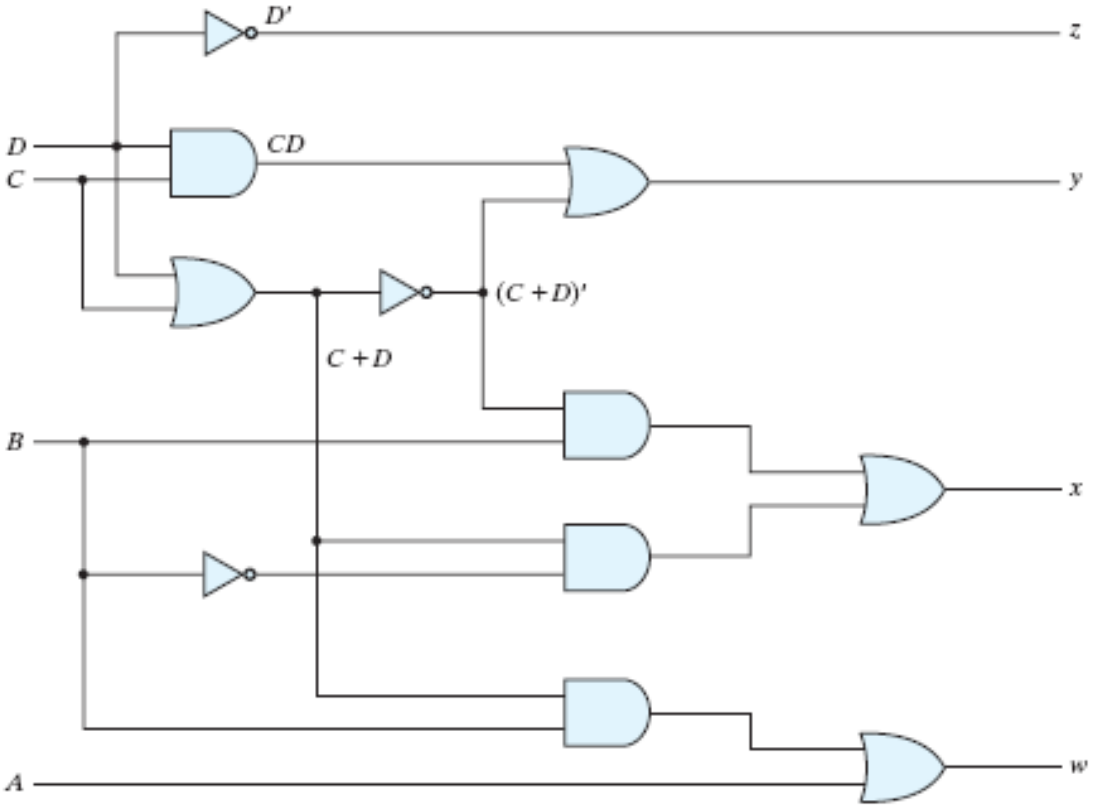
Code Conversion Example (Cont.)

- Following the design procedure:
  4. Draw logic diagram

Two-level implementation	Three-level implementation
z = D’ y = CD +C’D’ x = B’C + B’D + BC’D’ w= A +BC +BD	z = D’ y = CD +C’D’ = CD + (C + D)’ x = B’C + B’D + BC’D’ = B’(C +D) + B (C +D)’ w= A +BC +BD = A + B (C + D)
Its implementation requires: 7 AND gates, 3 OR gates and 3 inverters	Its implementation requires: 4 AND gates, 4 OR gates and 3 inverters
Thus, three-level requires fewer gates, all of which require only 2 inputs.	

Code Conversion Example (Cont.)

- Following the design procedure:
  4. Draw logic diagram with 3-level implementation



EXAMPLE2: HALF-ADDER CIRCUIT

- **Design a circuit that add 2 bits.**

1. # of inputs=? **2** , their labels=? **x, y**

   # of outputs=? **2** their labels=? **C(carry), S(Sum)**

2. Get truth table

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

3. Get the simplified Boolean functions

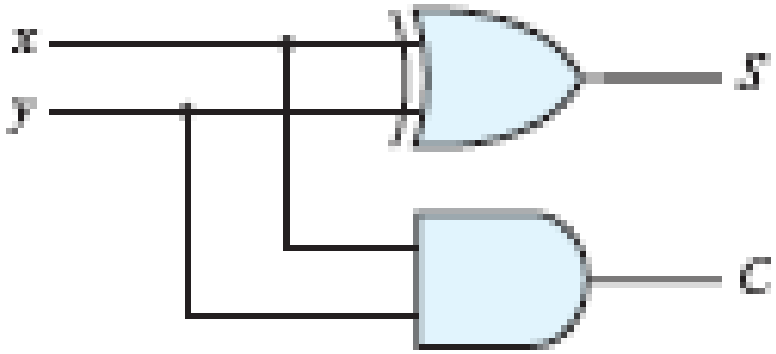
   $S = x'y + xy' = x \oplus y \quad C = xy$

EXAMPLE 2: HALF-ADDER CIRCUIT(CONT.)

- So, the simplified Boolean functions are:

  S= x'y +xy' = x⊕y C= xy

4. Draw the logic diagram



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

EXAMPLE3: FULL-ADDER CIRCUIT

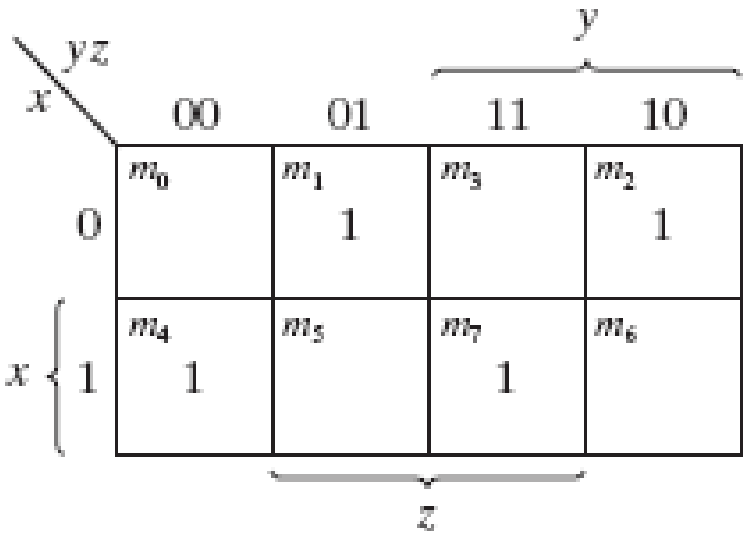
- **Design a circuit that add 3 bits.**

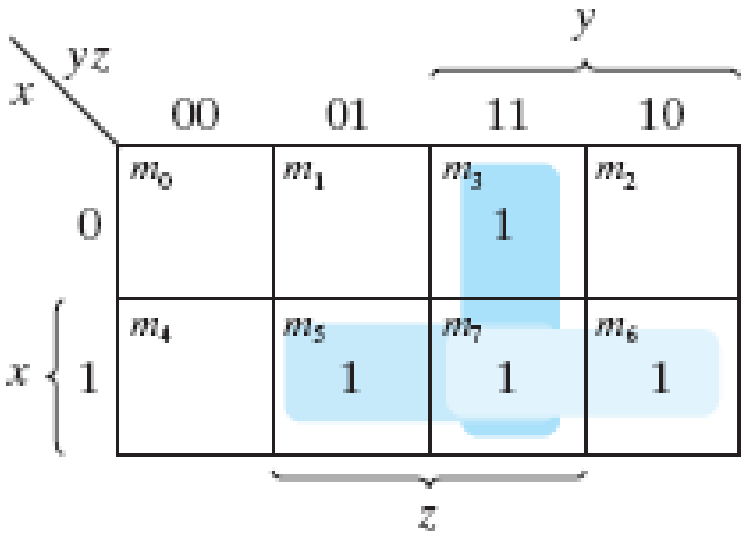
1. # of inputs=3 , their labels= x, y, z
   # of outputs=2 , their labels=C(carry), S(Sum)
2. Get truth table

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

EXAMPLE3: FULL-ADDER CIRCUIT(CONT.)

3. Get the simplified Boolean functions





= x ⊕ y ⊕ z

EXAMPLE 3: FULL-ADDER CIRCUIT (CONT.)

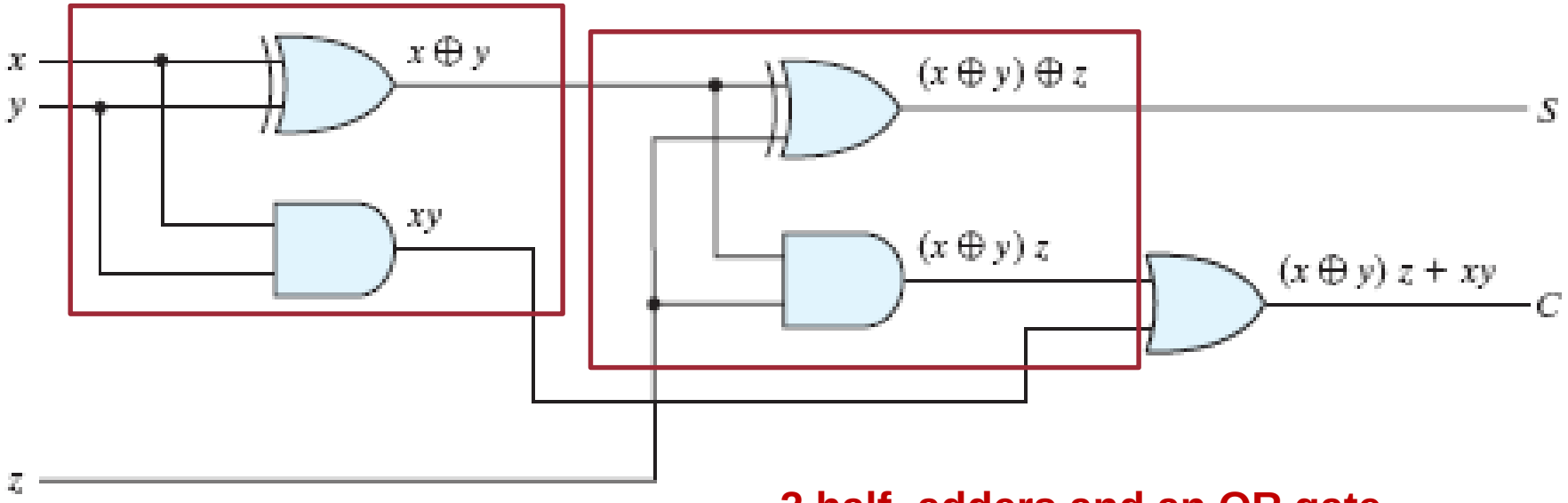
4. Draw the logic diagram using:

   $S = x \oplus y \oplus z$

Before Simplification: $C = x'yz + xy'z + xyz' + xyz$

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

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



2 half- adders and an OR gate

EXAMPLE4: BINARY ADDER

- A binary adder is a digital circuit that produces the arithmetic sum of two *n-bit* binary numbers;

  $x = A_n \ldots A_2 A_1 A_0$ and $y = B_n \ldots B_2 B_1 B_0$

- Example:

  Consider x= 1011 and y=0011, their sum=?

  $= A_3 A_2 A_1 A_0$ $= B_3 B_2 B_1 B_0$

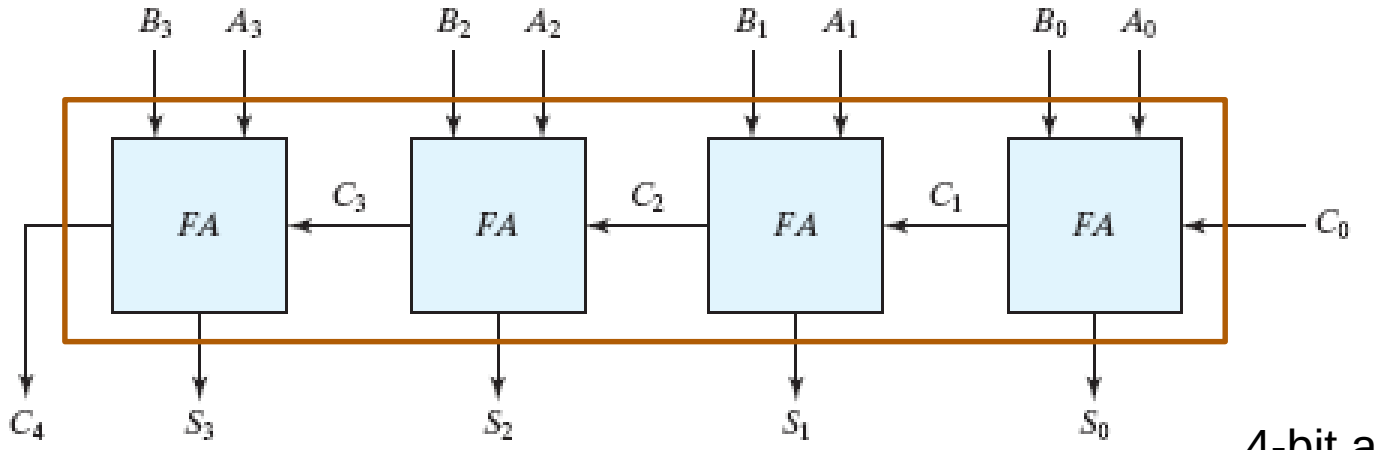
Must be zero

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

So, we need a circuit that:
Add 3 bits, and
Output:
sum and carry,
Which circuit?
How many?
At bit1, inputs?

EXAMPLE4: BINARY ADDER (CONT.)

- A *n-bit* binary adder can be constructed with:
  - *n* full adders connected in cascade, with
  - the output carry from each full adder connected to the input carry of the next full adder in chain



4-bit adder

EXAMPLE3: BINARY ADDER (CONT.)

- A 4-*bit* adder is a typical example of a *standard* component.
- It can be used in many applications involving arithmetic operations.
- The design of this circuit by the classical method would require:

2^9 =512 entries

Nine inputs?

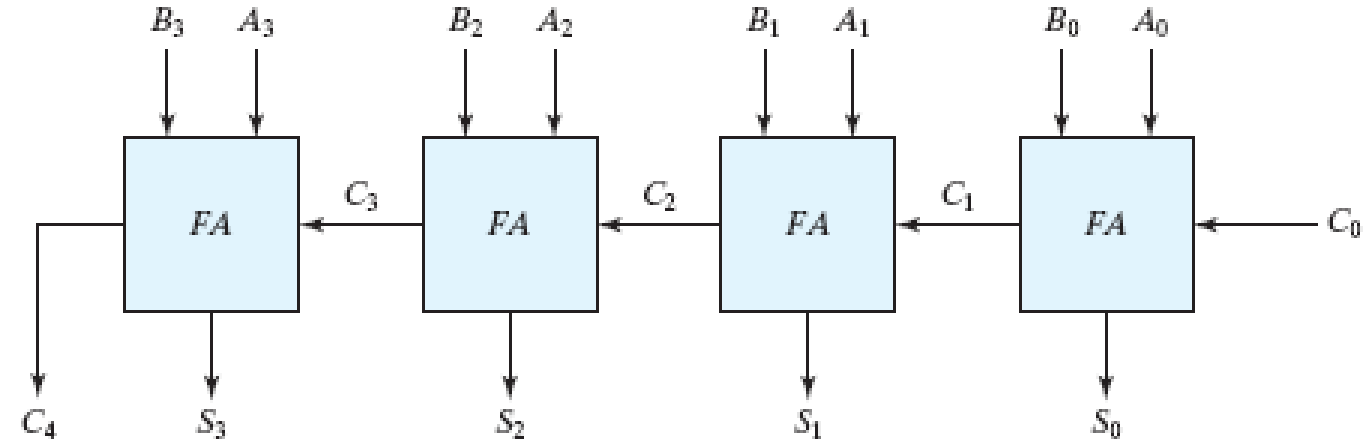
Carry propagation

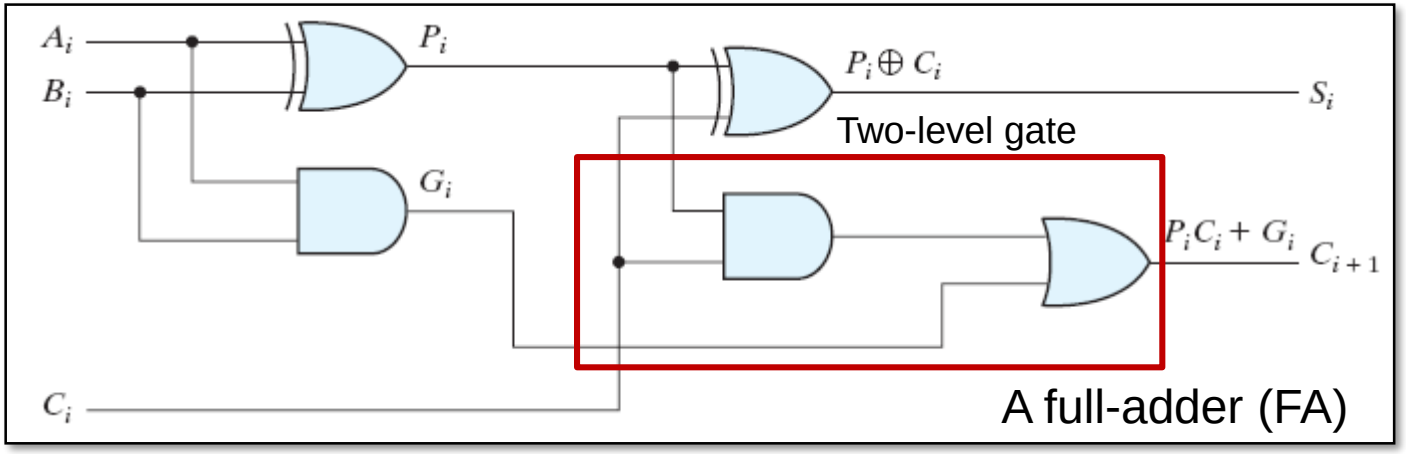
- As in any combinational circuit, the signal must propagate through the gates before correct output sum is available in the output terminals.

In adder, the longest propagation delay is the time it takes the carry to propagate through the full adders.

Total propagation = gate delay + # gate level

CARRY PROPAGATION (CONT.)





Since, Ci to Ci+1 require 2 gate level.

So, 4-bit full adder would require 2 x 4 =8 gate levels from C0 to C4

For n-bit adder, there are 2n gate levels for carry propagation

Carry propagation (Cont.)

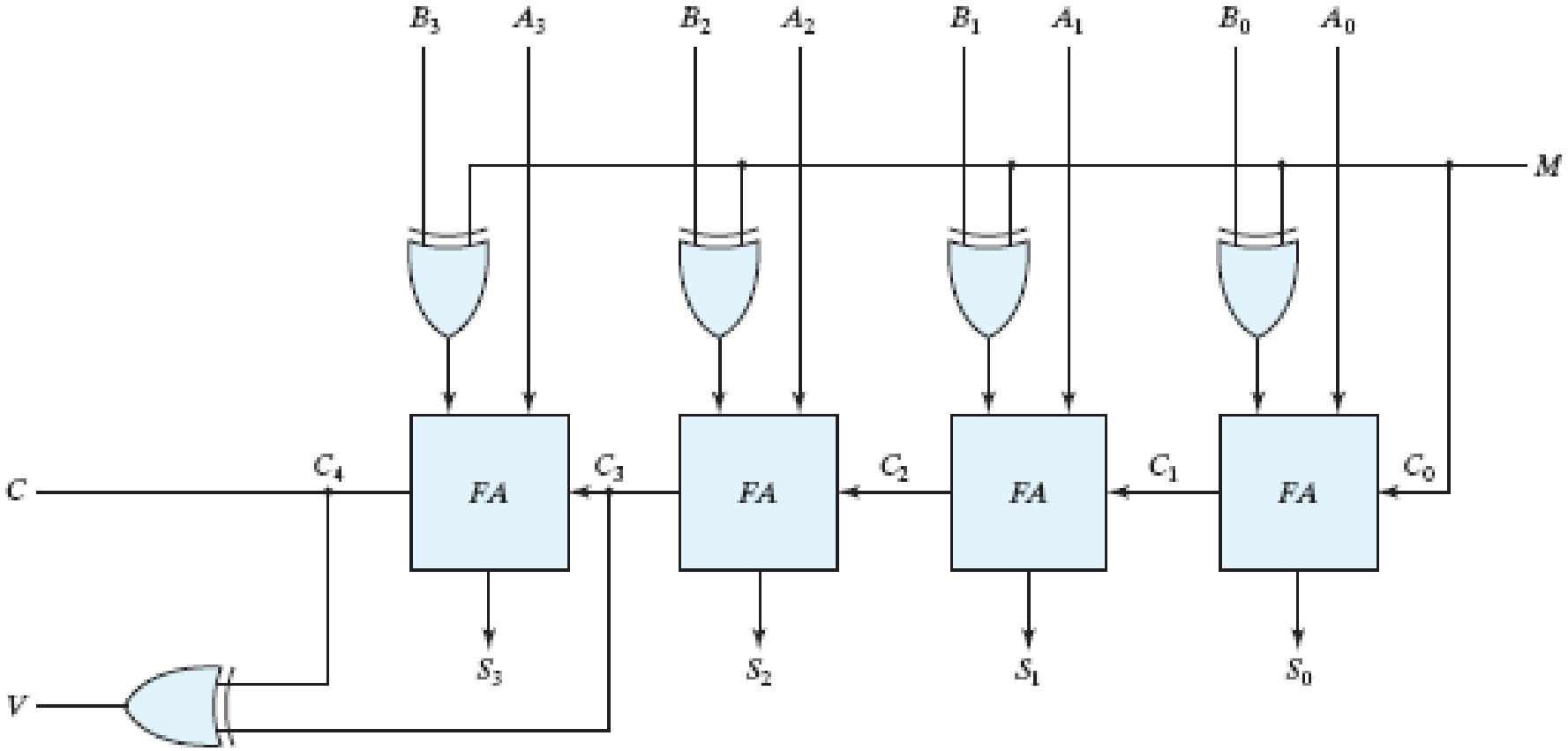
- The carry propagation time is an important attribute since:
  - The output will not be correct unless the signals given enough time to propagate
  - It limits the speed which 2 numbers are added.
  - All arithmetic operations are implemented by successive additions
- Carry propagation solutions:
  - Increase equipment complexity

    ➔ widely used technique "*carry lookahead logic*"

    ➔All output carries are generated after a delay of 2-level gates

BINARY ADDER-SUBTRACTOR

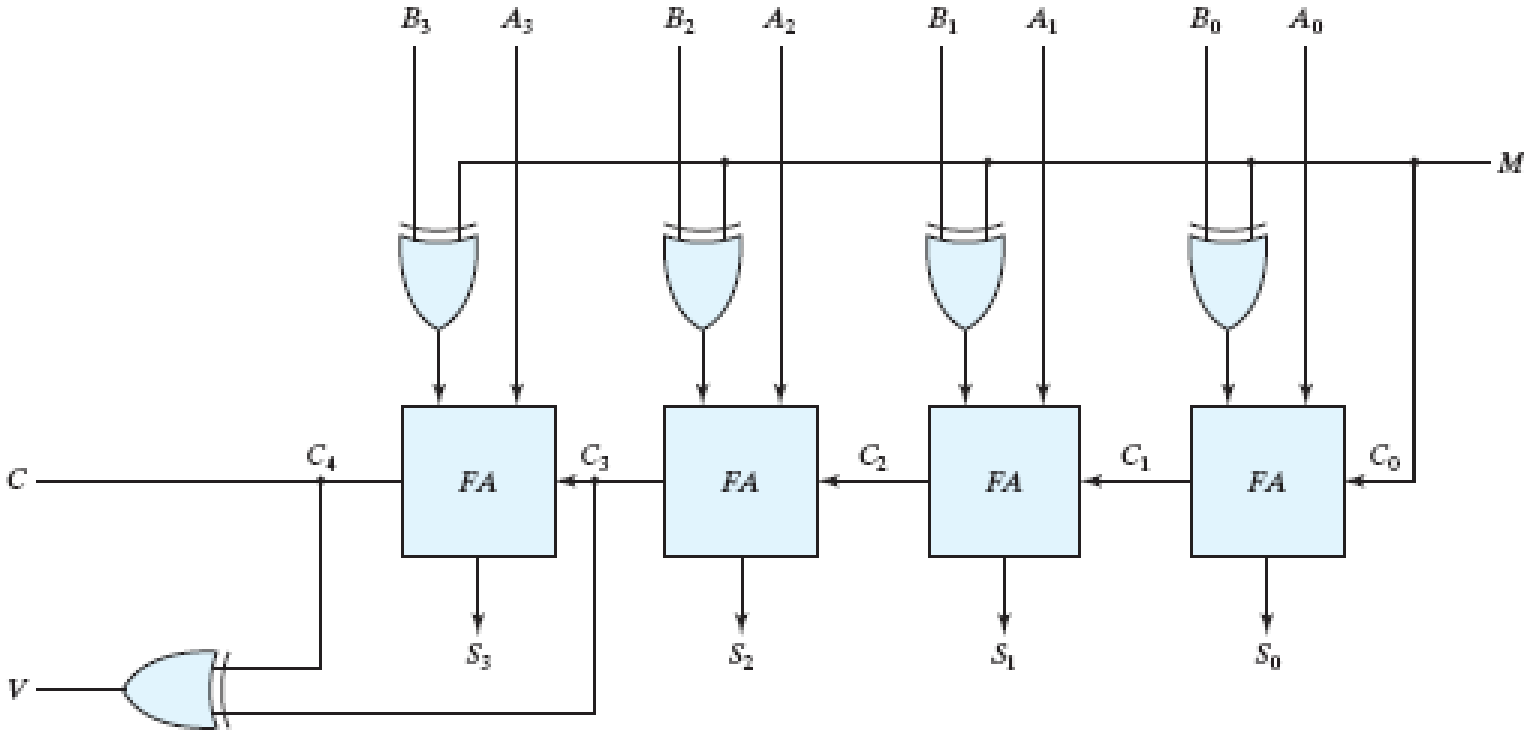
- M=0 ➔Adder (B⊕0=B and C_0=0)
- M=1 ➔ Subtractor (B⊕1=B' and C_0=1)



BINARY ADDER-SUBTRACTOR

If the numbers considered to be signed, V bit detects overflow

- If V=0, no overflow, the n-bit result is correct
- If V=1, overflow occurred and the results contains n+1 bits



OVERFLOW

- When two numbers with n digits each are added and the sum is a number occupying n+1 digits

  ➔ Overflow occurred

- It 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 can't be hold by an n bit word
- For this reason, many computers detect the occurrence of overflow, and if it occurs a corresponding flip-flop is set

OVERFLOW(CONT.)

Ex: Addition of +50 and +20 = +70,

Carries:	0 0	
+50	0	0110010
+20	0	0010100
+70	0	1000110

Addition of :

- Two unsigned numbers, overflow is detected from:
  - end carry out of MSB (Most Significant Bit)
- Two signed numbers both positive or negative, overflow can be detected by:
  - observing the carry into the sign bit and the carry out of the sign bit.
  - If these two carries are not equal, an overflow has occurred

Ex: Addition of +70 and +80 = +150, while 8-bit location can hold a range from +127 to -128

Carries:	0	1		carries:	1	0	
+70		0	1000110	-70		1	0111010
+80		0	1010000	-80		1	0110000
+150		1	0010110	-150		0	1101010

Quiz 1

- Quiz1 next week in sections until K-map
- Week starting Sun. 14 Nov.

Thanks