

# DIGITAL LOGIC

## Chapter 4 part1: Combinational Logic

2024 Fall

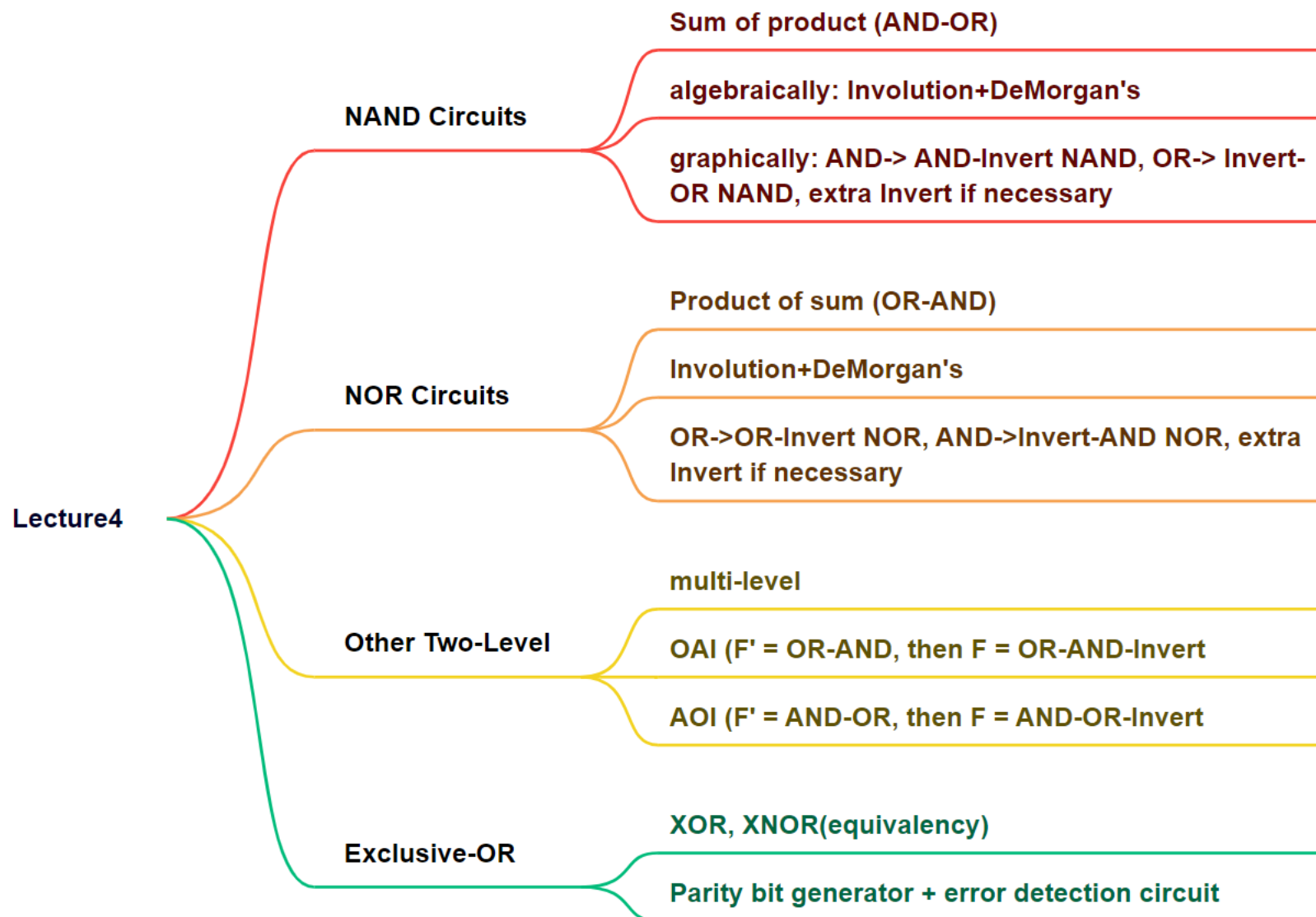
This PowerPoint is for internal use only at Southern University of Science and Technology.  
Please do not repost it on other platforms without permission from the instructor.



# Today's Agenda

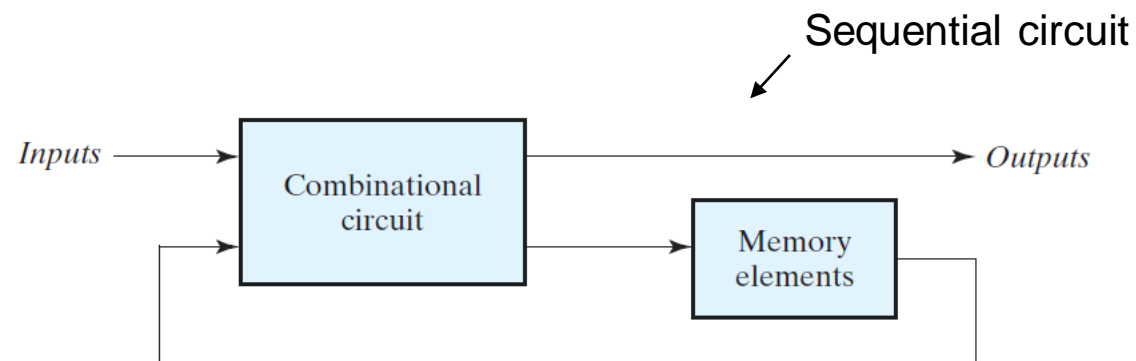
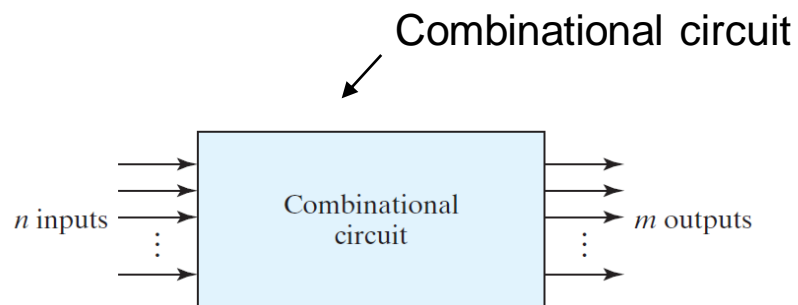
- Recap
- Context
  - Combinational Circuits
    - Analysis of Combinational Circuits
    - Design Procedure
  - Basic Components
    - Magnitude Comparator
- Reading: Textbook, Chapter 4.1-4.4, 4.8

# Recap



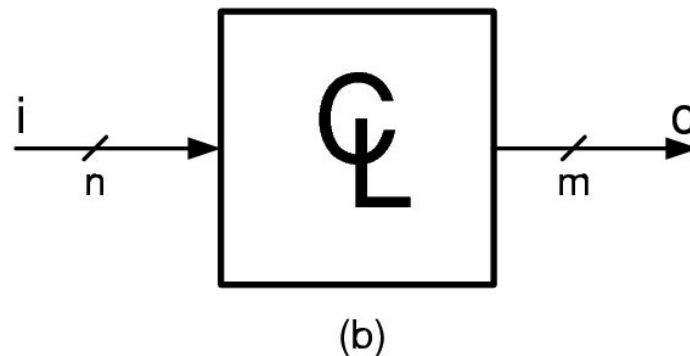
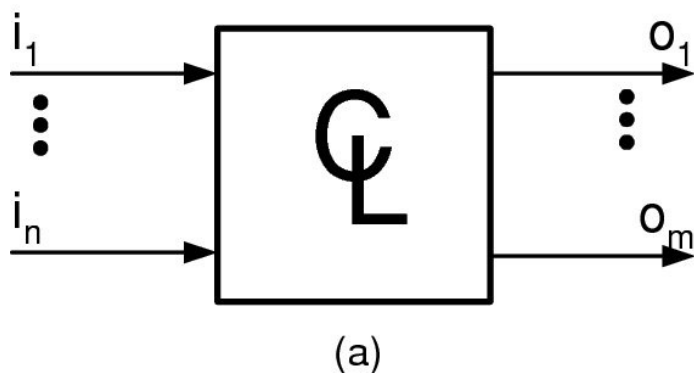
# Logic Circuits for the Digital System

- Combinational circuits
  - Logic circuits whose outputs at any time are determined directly and only from the present input combination.
- Sequential circuits (next chapter)
  - Circuits that employ memory elements + (combinational) logic gates
  - Outputs are determined from the present input combination as well as the state of the memory cells.



# Combinational Logic Circuits

- Memoryless:  $o=f(i)$ 
  - Used for control, arithmetic, and data steering.
  - such as adders, subtractors, comparators, decoders, encoders, and multiplexers.
  - These components are available in integrated circuits as medium-scale integration (MSI) circuits



# Outline

- **Analysis of Combinational Circuits**
- Design of Combinational Circuits

# Analysis Procedure

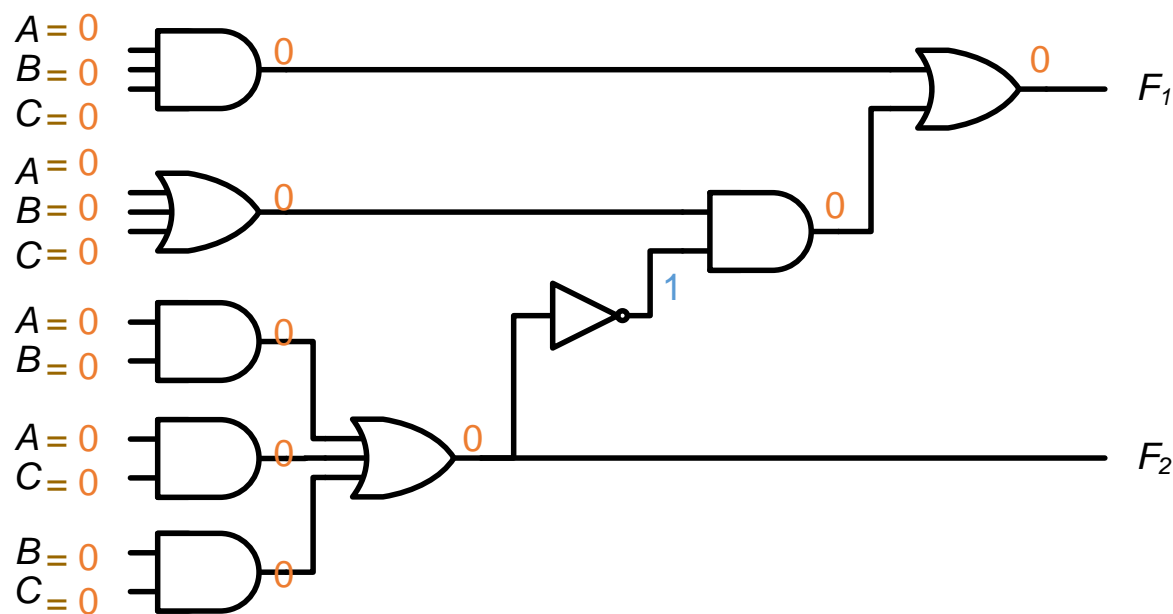
- Analysis of a combinational circuit: determine the function of the circuit.
  - From given logic diagram,
  - Aim to develop a set of Boolean functions, a truth table, an optional explanation of the circuit operation.
  - If a function name or an explanation is given along the circuit, just verify if the given information is correct.

# Analysis with Truth Table: Steps

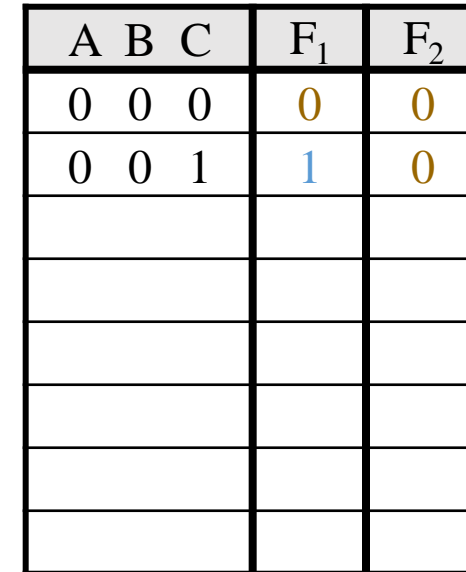
- Derivation of truth table for  $n$  input variables
  - List all the  $2^n$  input combinations from 0 to  $2^n-1$ .
  - Partition the circuit into small single-output blocks and label the output of each block.
  - Obtain the truth table of the blocks depending on the input variables only.
  - Proceed to obtain the truth tables for other blocks that depend on previously defined truth tables.
- Then determine the function of the circuit (e.g. using K-map)
- Explain the functionality of the circuit if necessary



- Truth Table Approach

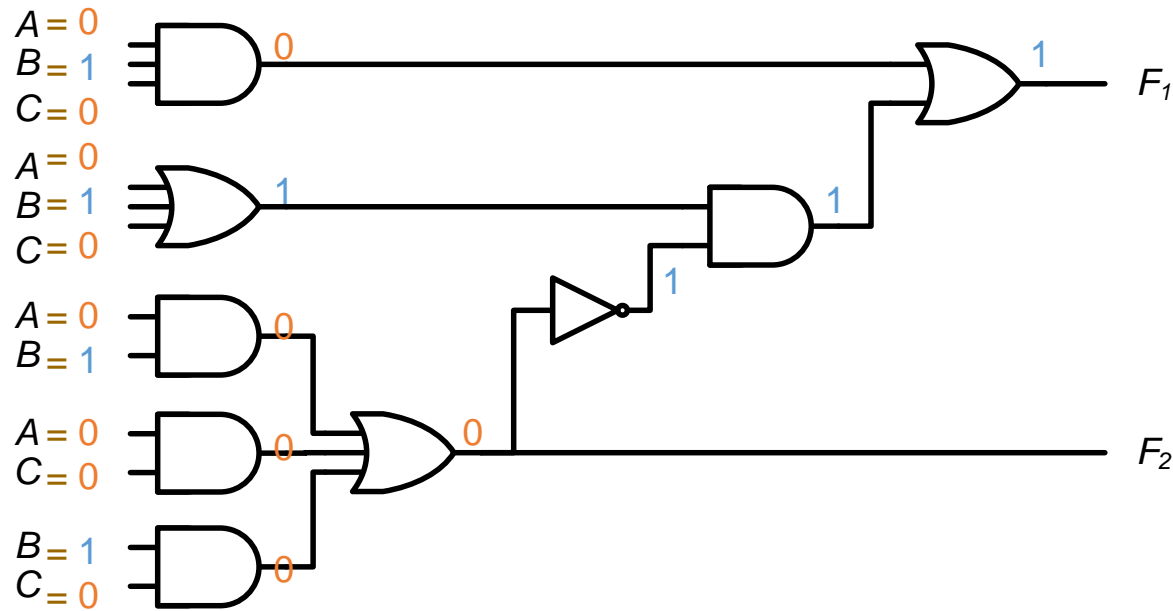
[illegible]

- Truth Table Approach



# Analysis with Truth Table: Example

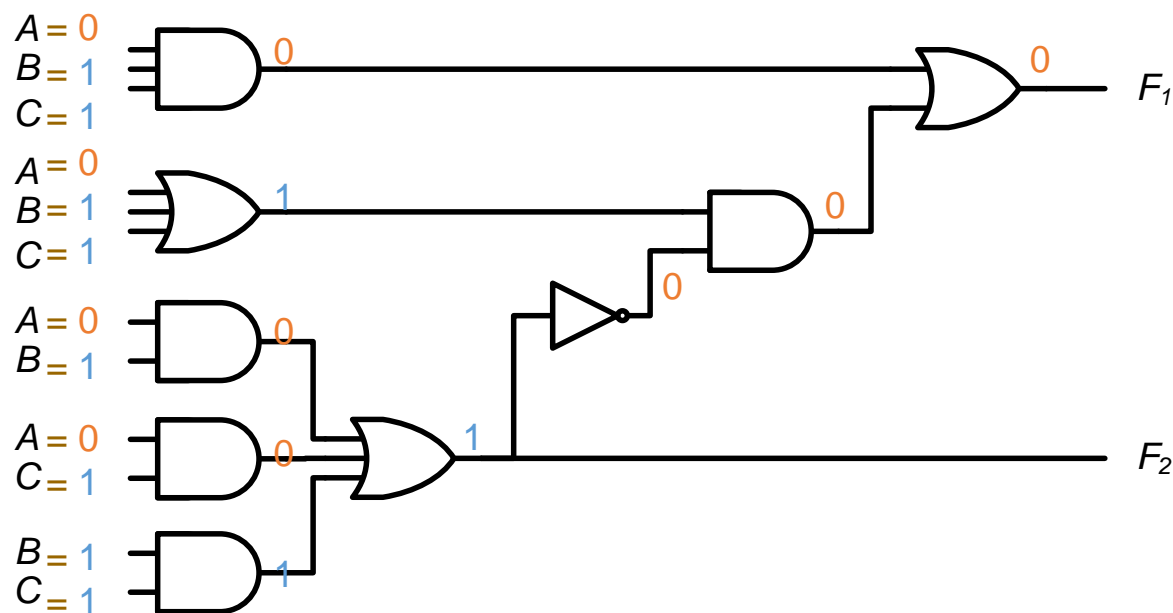
- Truth Table Approach



| A | B | C | $F_1$ | $F_2$ |
|---|---|---|-------|-------|
| 0 | 0 | 0 | 0     | 0     |
| 0 | 0 | 1 | 1     | 0     |
| 0 | 1 | 0 | 1     | 0     |
|   |   |   |       |       |
|   |   |   |       |       |
|   |   |   |       |       |
|   |   |   |       |       |
|   |   |   |       |       |

# Analysis with Truth Table: Example

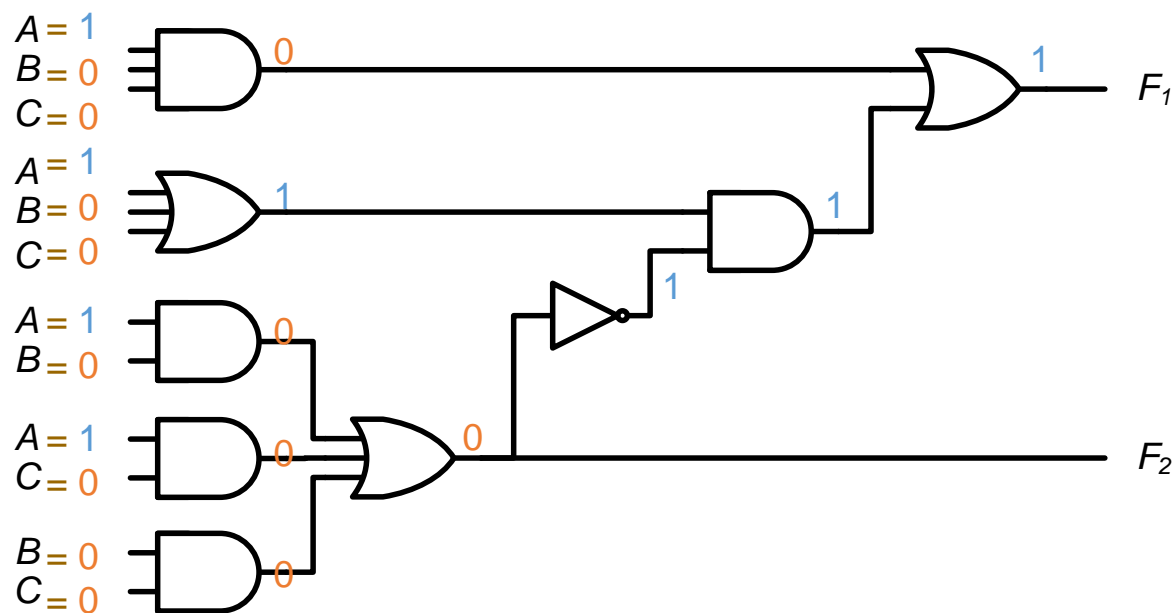
- Truth Table Approach



| 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     |
|   |   |   |       |       |
|   |   |   |       |       |
|   |   |   |       |       |
|   |   |   |       |       |

# Analysis with Truth Table: Example

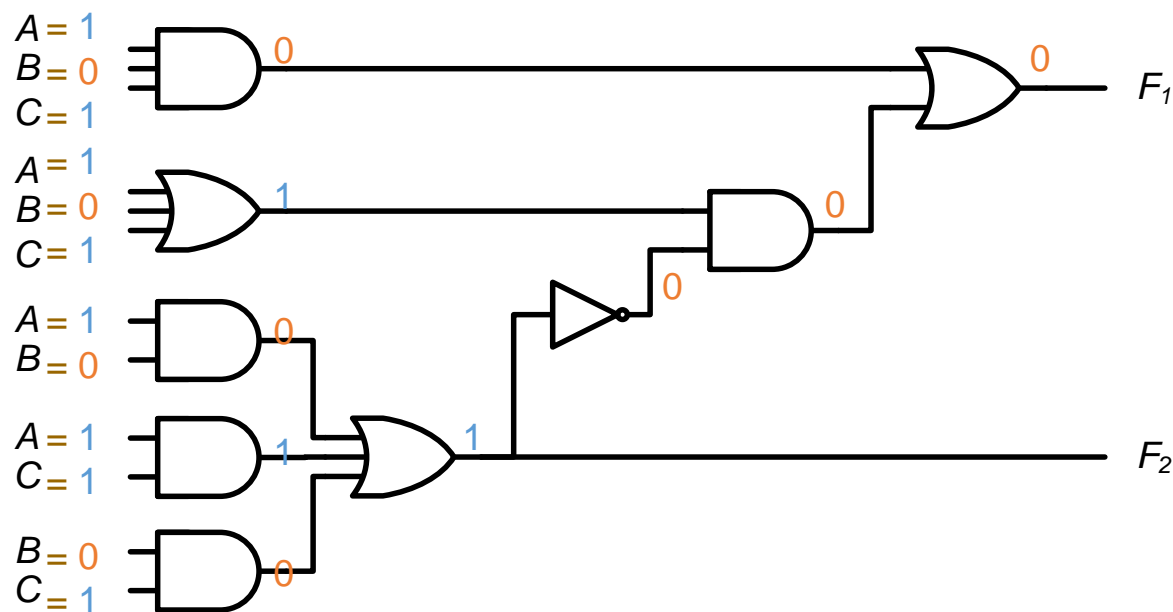
- Truth Table Approach



| 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     |
|   |   |   |       |       |
|   |   |   |       |       |
|   |   |   |       |       |

# Analysis with Truth Table: Example

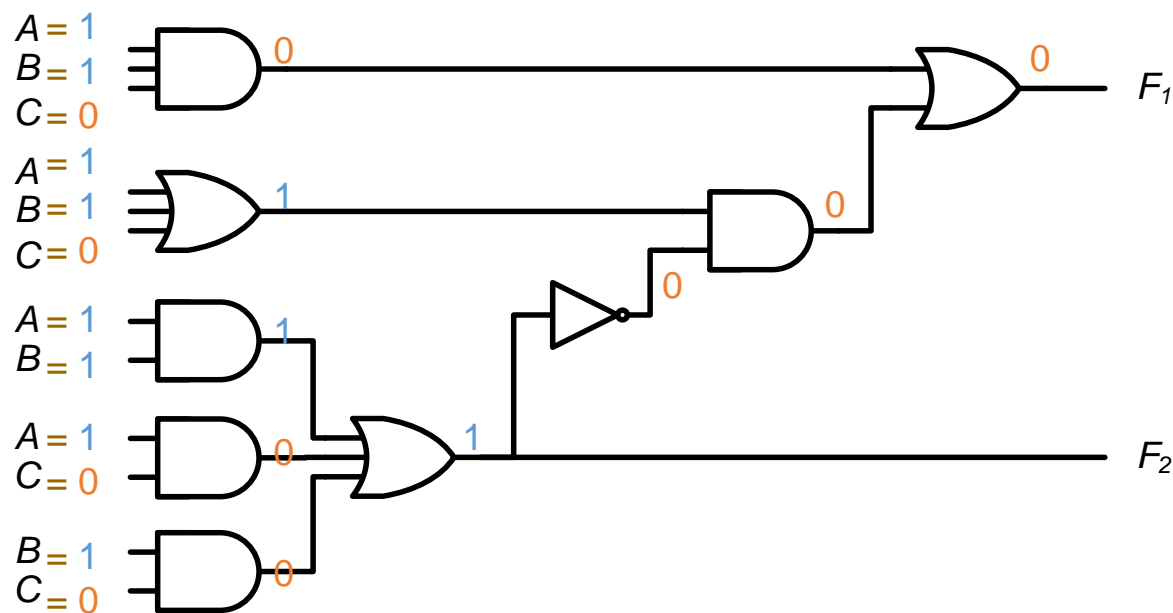
- Truth Table Approach



| A | B | C | F <sub>1</sub> | F <sub>2</sub> |
|---|---|---|----------------|----------------|
| 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              |
|   |   |   |                |                |
|   |   |   |                |                |

# Analysis with Truth Table: Example

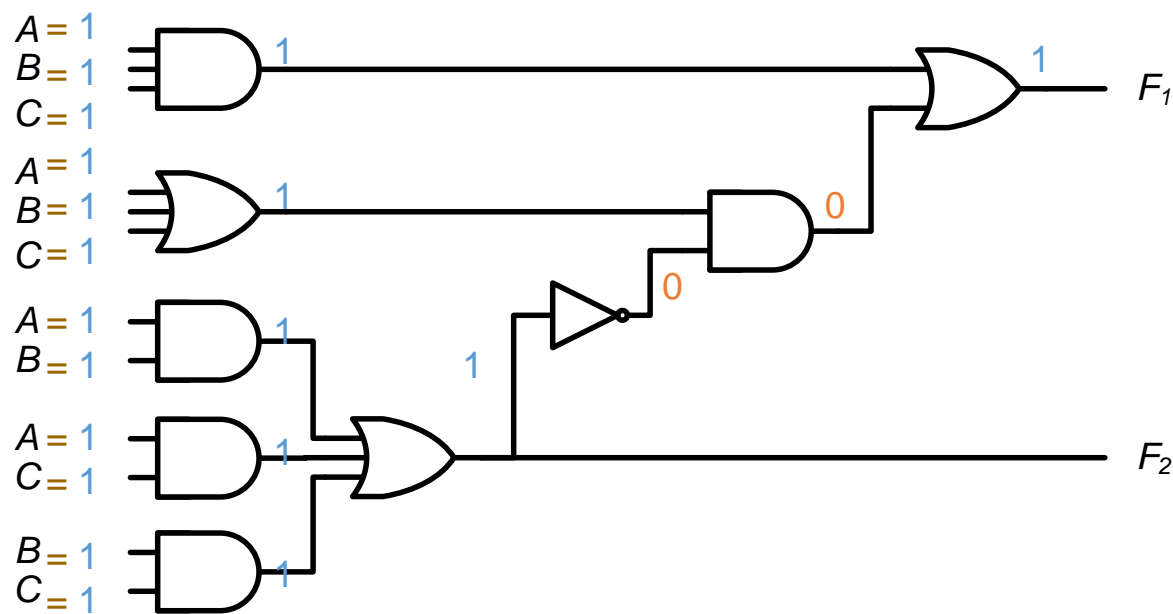
- Truth Table Approach



| 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     |
|   |   |   |       |       |

# Analysis with Truth Table: Example

- 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     |



# Analysis with Truth Table: Example

- Truth table ready
- Obtain Boolean function
- Functionality: the circuit is a Full adder ( $F_1$ : sum,  $F_2$ : carry. Arithmetic circuit will be covered in lecture 10)

| 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 | B          |            |            |            |
|--------|------------|------------|------------|------------|
|        | 00         | 01         | 11         | 10         |
| 0      | $m_0$<br>0 | $m_1$<br>1 | $m_3$<br>0 | $m_2$<br>1 |
| 1      | $m_4$<br>1 | $m_5$<br>0 | $m_7$<br>1 | $m_6$<br>0 |

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

$$= A \oplus B \oplus C$$

| A \ BC | B          |            |            |            |
|--------|------------|------------|------------|------------|
|        | 00         | 01         | 11         | 10         |
| 0      | $m_0$<br>0 | $m_1$<br>0 | $m_3$<br>1 | $m_2$<br>0 |
| 1      | $m_4$<br>0 | $m_5$<br>1 | $m_7$<br>1 | $m_6$<br>1 |

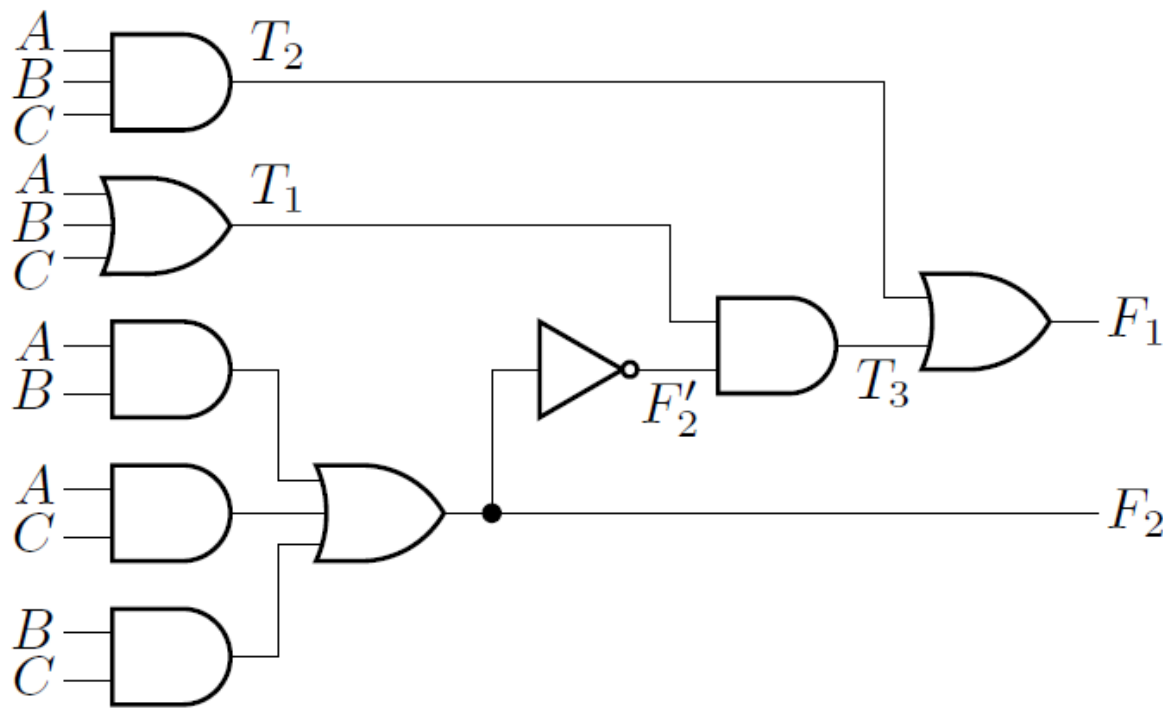
$$F_2 = AB + AC + BC$$

# Analysis with Boolean Function Derivation: Steps

- Obtain the output Boolean functions:
  - Label all gate outputs that are a function of only inputs, no other intermediate variables. Determine their Boolean functions.
  - Label all gates that are a function of inputs and the gates in the previous step. Determine their Boolean functions.
  - List output Boolean functions, squeeze the intermediate variables.
- Then simplify the function of the circuit if necessary
- Explain the functionality of the circuit if necessary

# Analysis with Boolean Function Derivation: Example 1

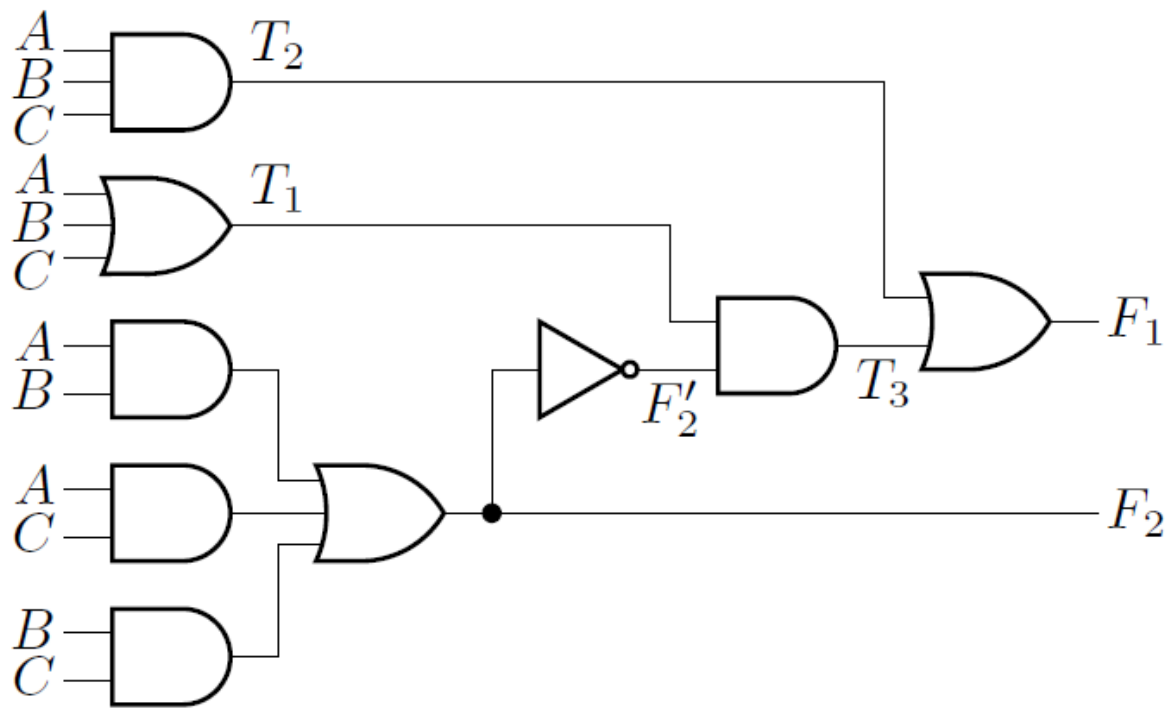
- Obtain the output Boolean functions:
  - Label all gate outputs that are a function of only inputs, no other intermediate variables. Determine their Boolean functions.



- $F_2 = AB + AC + BC$
- $T_1 = A + B + C$
- $T_2 = ABC$

# Analysis with Boolean Function Derivation: Example 1

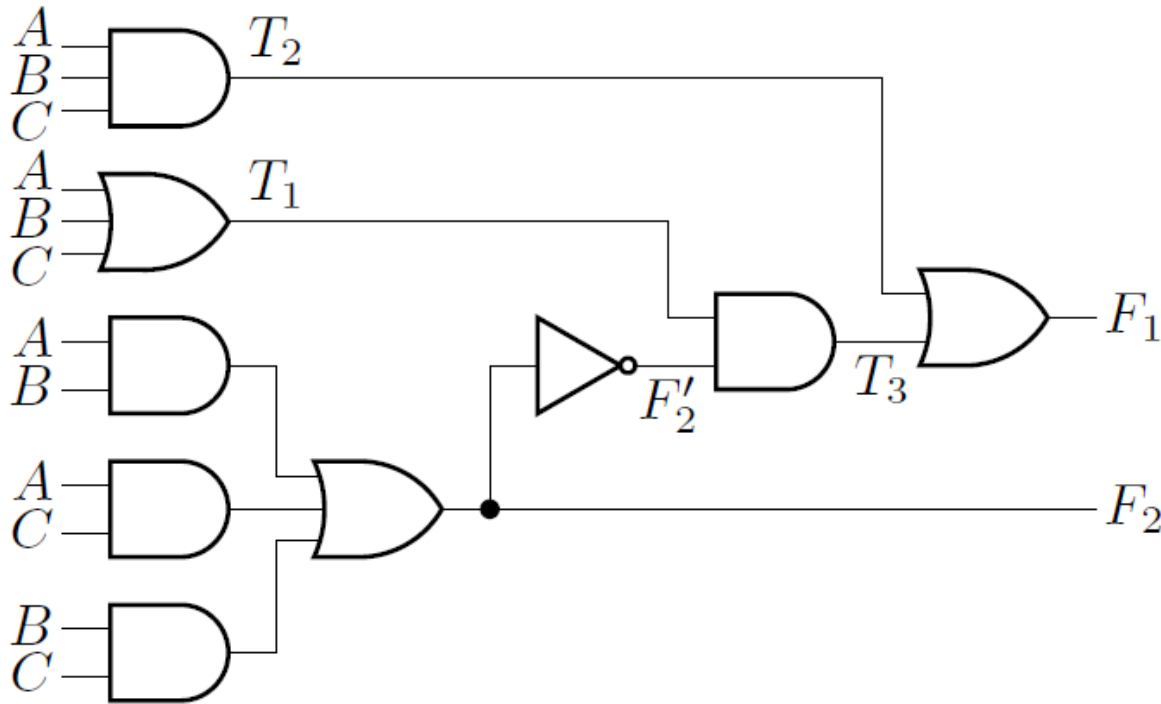
- Obtain the output Boolean functions:
  - Label all gates that are a function of inputs and the gates in the previous step. Determine their Boolean functions.



- $T_3 = F_2' T_1$
- $F_1 = T_3 + T_2$

# Analysis with Boolean Function Derivation: Example 1

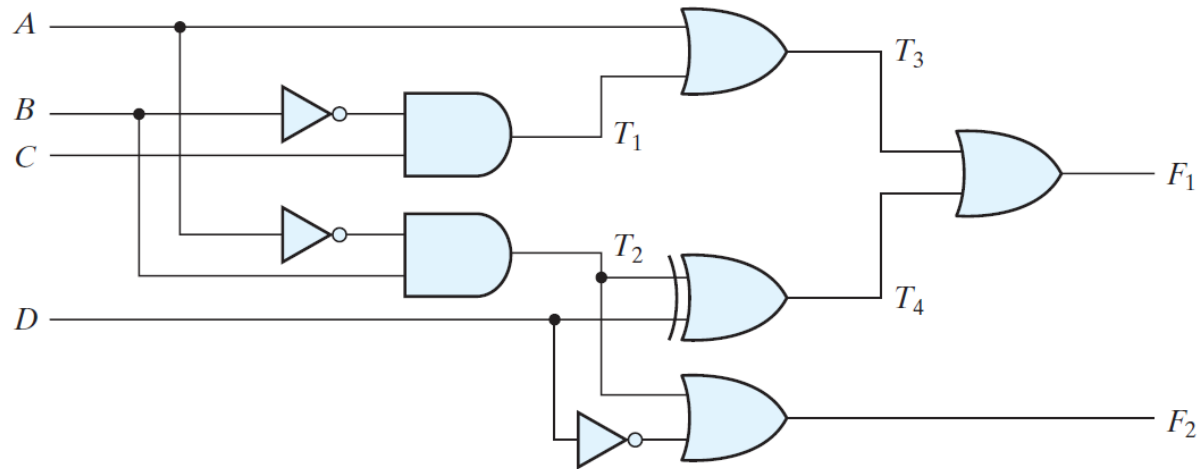
- Obtain the output Boolean functions:
  - List output Boolean functions, squeeze the intermediate variables.



- $$F_1 = T_2 + T_3$$
$$= ABC + F_2' T_1$$
$$= ABC + (AB + AC + BC)'(A + B + C)$$
$$= A'B'C + A'BC' + AB'C' + ABC$$
- $F_2 = AB + AC + BC$
- Full adder

# Analysis with Boolean Function Derivation: Exercise

- Derive the Boolean expressions for  $T_1$  through  $T_4$ . Evaluate the outputs  $F_1$  and  $F_2$  as a function of the four inputs.



# Outline

- Analysis of Combinational Circuits
- **Design of Combinational Circuits**

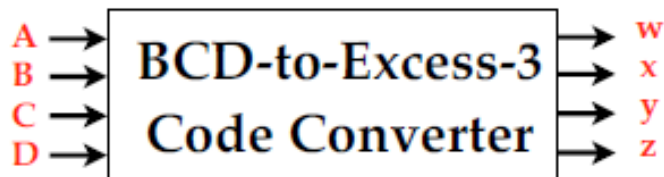
# Design Procedure

- Design of a combinational circuits: develop a logic circuit diagram or a set of Boolean functions.
  - From specification of the design objective
- Involves the following steps:
  1. Specification: From the specifications, determine the inputs, outputs, and their symbols.
  2. Formulation: Derive the truth table (functions) from the relationship between the inputs and outputs
  3. Optimization: Derive the simplified Boolean functions for each output.
  4. Logic diagram (optional): Draw a logic diagram for the resulting circuits using AND, OR, and inverters. (Or using required technology mapping)



# Example1: BCD-to-Excess-3 Code Converter

- Step1: Spec
  - input (ABCD)
  - output (wxyz) (MSB to LSB)
  - ABCD: 0000 ~ 1001 (0~9)
- Step2: Formulation
  - $wxyz = ABCD + 0011$



| 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 |

don't care

# Example1: BCD-to-Excess-3 Code Converter

## • Step3: Optimization

$$w = A + BC + BD$$

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

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

$$z = D'$$

from K-map

$$w = A + BC + BD$$

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

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

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

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

$$z = D'$$

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

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

# Example1: BCD-to-Excess-3 Code Converter

- Step3: Optimization
- We manipulate these Boolean functions to reuse common gates:

$$w=A+BC+BD = A+B(\textcolor{red}{C+D})$$

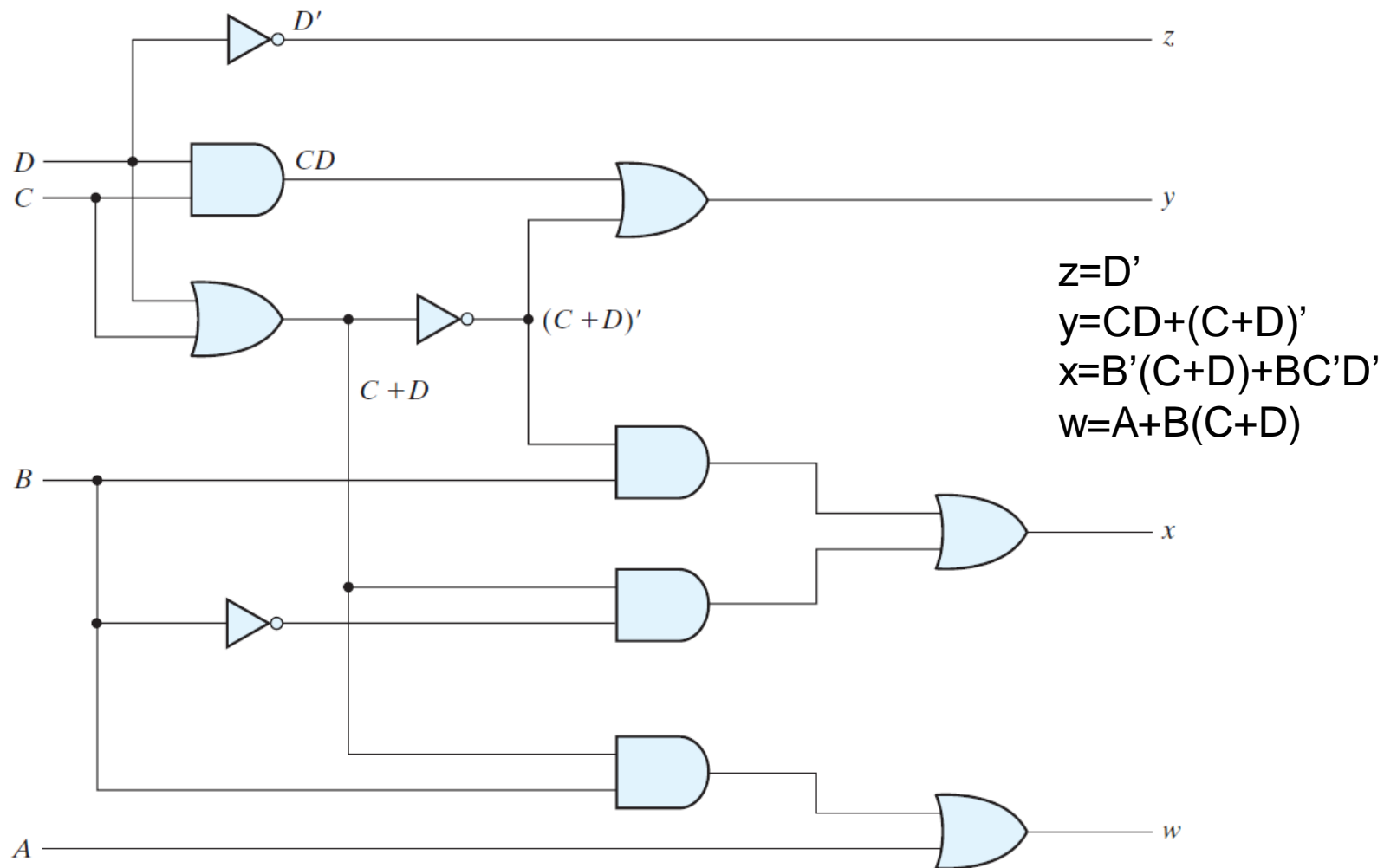
$$x=B'C+B'D+BC'D' = x=B'(\textcolor{red}{C+D})+BC'D'$$

$$y=CD+C'D' = CD+(\textcolor{red}{C+D})'$$

$$z=D'$$

# Example1: BCD-to-Excess-3 Code Converter

- Step4: Draw logic diagram



# Example2: 4-bit Equality Comparator

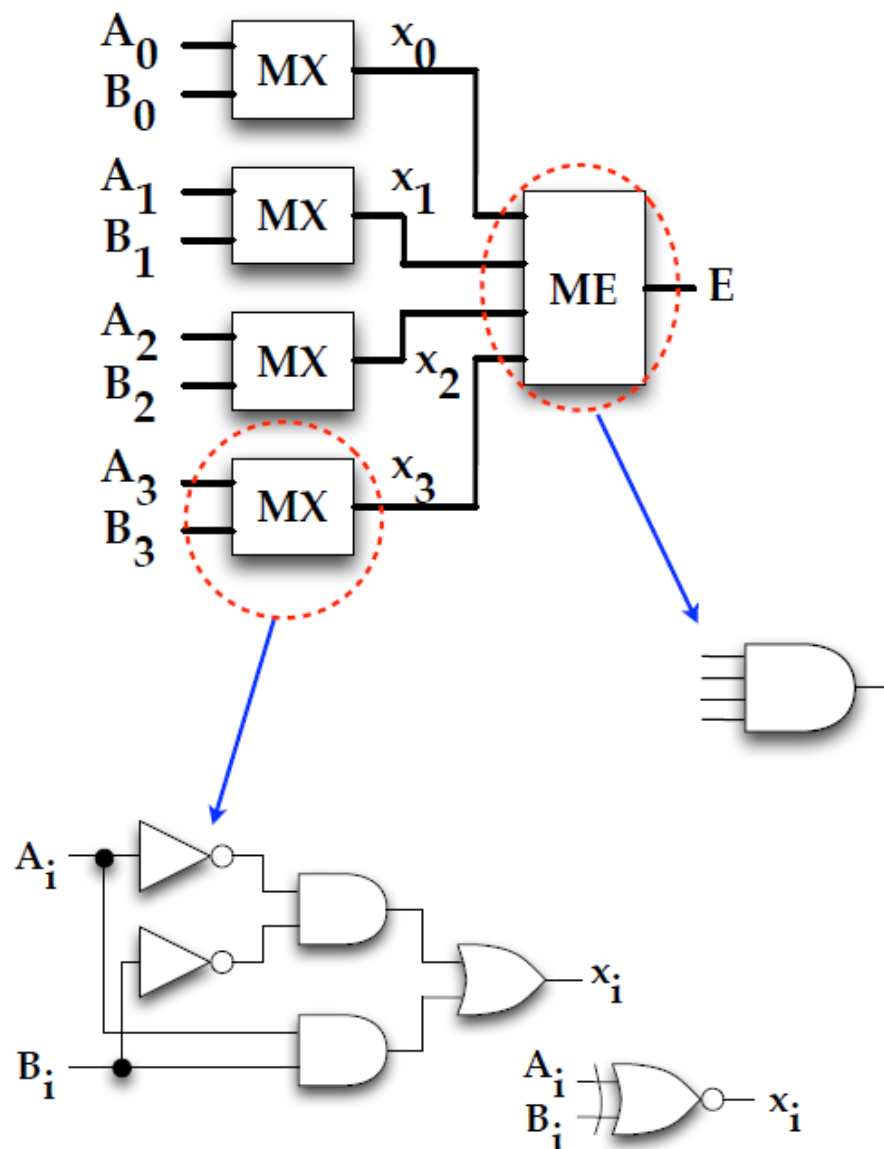
- Step1: Spec
  - input A(3:0), B(3:0);
  - output E (1/0 for equal/unequal)
- Step2: Formulation
  - By truth table:  $2^8$  rows => not practicable
  - By algorithm: build a regular circuit
    - 4-bit A:  $A_3A_2A_1A_0$ , 4-bit B:  $B_3B_2B_1B_0$
    - $A==B$ , if  $(A_3==B_3) \& (A_2==B_2) \& (A_1==B_1) \& (A_0==B_0)$
    - Suppose bit equality  $X_i=(A_i \oplus B_i)' = A_iB_i + A_i'B_i'$ , then  $(A==B) = X_3X_2X_1X_0$



Hint: XNOR gate's output  
Is 1 when input values  
Are same

# Example2: 4-bit Equality Comparator

- Step3: Optimization
  - Regularity
  - Reuse
- Step4: Draw diagram
  - MX : stands for XNOR
  - ME : stands for AND



# Example3: Magnitude Comparator

- Step1: Spec
  - Comparison of two numbers, three possible results ( $A > B$ ,  $A = B$ ,  $A < B$ )
- Step2: Formulation (for n-bit numbers)
  - By truth table:  $2^{2n}$  rows  $\Rightarrow$  not practicable
  - If the truth table is too cumbersome, the regularity of comparator circuit allows deriving the function using algorithm
- Step3: Optimization
  - By algorithm to build a regular circuit
  - 4-bit A:  $A_3A_2A_1A_0$ , 4-bit B:  $B_3B_2B_1B_0$
  - $A == B$ , if  $(A_3 == B_3) \text{ AND } (A_2 == B_2) \text{ AND } (A_1 == B_1) \text{ AND } (A_0 == B_0)$ 
    - equality  $x_i = A_i B_i + A_i' B_i'$ ,  $(A = B) = x_3 x_2 x_1 x_0$
  - $(A > B) = A_3 B_3' + x_3 A_2 B_2' + x_3 x_2 A_1 B_1' + x_3 x_2 x_1 A_0 B_0'$
  - $(A < B) = A_3' B_3 + x_3 A_2' B_2 + x_3 x_2 A_1' B_1 + x_3 x_2 x_1 A_0' B_0$

# Example3: Magnitude Comparator

- Step4

