# Digital Systems Design and Laboratory
# [ 8. Combinational Circuit Design ]

Chung-Wei Lin
cwlin@csie.ntu.edu.tw
CSIE Department
National Taiwan University

We would like to thank Prof. Hui-Ru Jiang (NTU) for contributing the lecture material

# Outline

- ❑ **Review of Combinational Circuit Design**
- ❑ Design of Circuits with Limited Gate Fan-In
- ❑ Gate Delays and Timing Diagrams
- ❑ Hazards in Combinational Logic
- ❑ Simulation and Testing of Logic Circuits

# Combinational Circuit Design

不看記憶 ↙ sequential：有記憶

- Generic combinational circuit design steps
  - Translate the word description into a switching function (Unit 4)
  - Simplify the function
    - Boolean algebra (Units 2 and 3)
    - Karnaugh map (Unit 5)
    - Quine-McCluskey (Unit 6)
    - Other methods
  - Realize it using available logic gates (Unit 7)
    - Hardware cost = (#terms, #literals)
    - Start from minimum SOP ≡ AND-OR → NAND-NAND/OR-NAND/NOR-OR
    - Start from minimum POS ≡ OR-AND → NOR-NOR/AND-NOR/NAND-AND
    - # of levels = maximum # of cascaded gates between I/Os
      - Change level by factoring or multiplying out

# Outline

- ❑ Review of Combinational Circuit Design
- ❑ **Design of Circuits with Limited Gate Fan-In**
- ❑ Gate Delays and Timing Diagrams
- ❑ Hazards in Combinational Logic
- ❑ Simulation and Testing of Logic Circuits

# Example 1: Single Output

- Realize F(A, B, C, D) = ∑ m(0, 3, 4, 5, 8, 9, 10, 14, 15) using 3-input NOR gates (2個引入，但不能超過3)
  - F' = A'B'C'D + AB'CD + ABC' + A'BC + A'CD' (from K-map)
  - F = (A + B + C + D')(A' + B + C' + D')(A' + B' + C)(A + B' + C')(A + C' + D)
    - Two 4-input OR gates (X) → factoring
    - Three 3-input OR gates (O)
    - One 5-input AND gate (X)

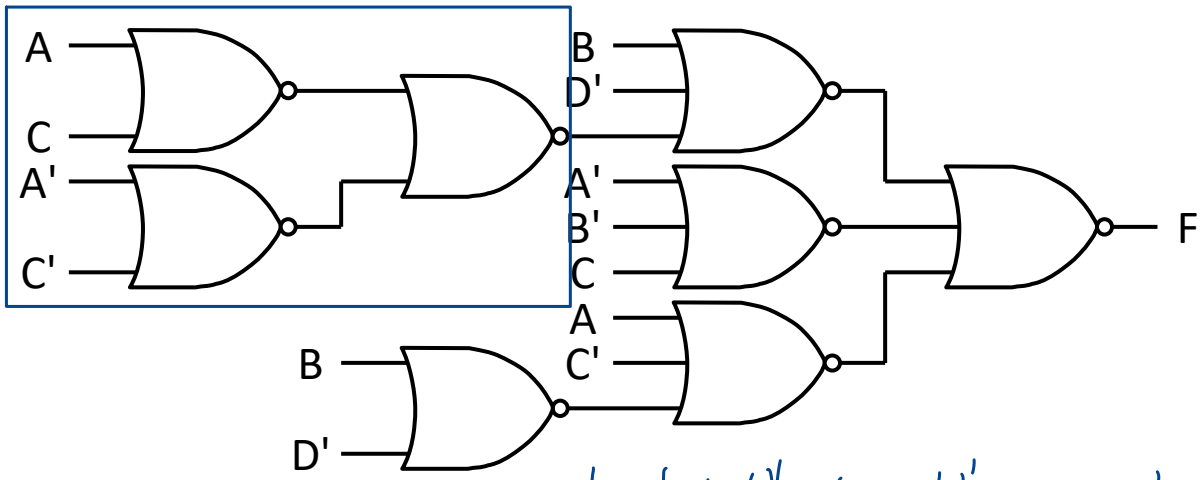
    $= [(A+C')+B'][(A+C')+D]$

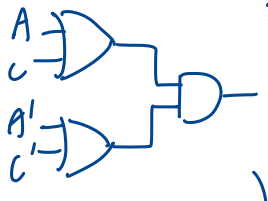
    $= (A+C')+B'D$

    $(X+Y)(X+Z) = X+YZ$

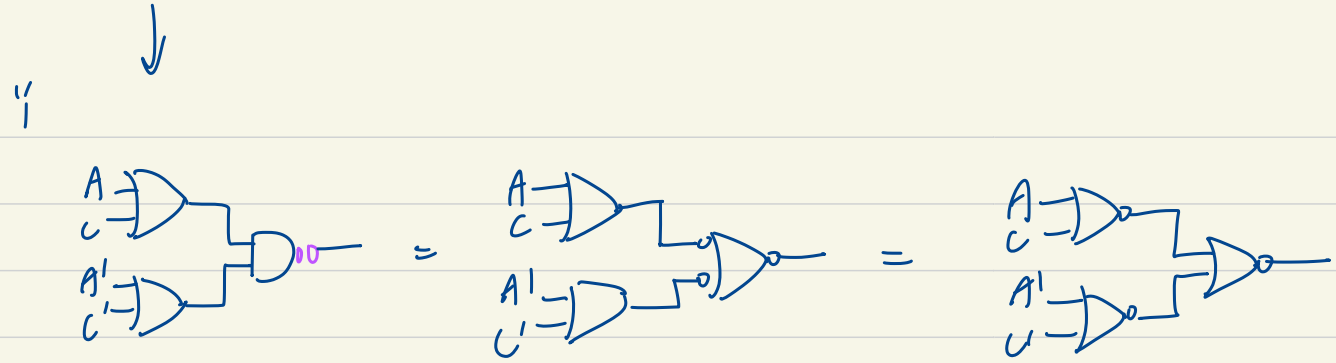
    $(A+B+C+D')(A'+B+C'+D')$

    $= [(A+C)+(B+D')][(A'+C')+(B+D')] = (B+D')+(A+C)(A'+C')$
  - F = [B + D' + (A + C)(A' + C')] [A' + B' + C] [A + C' + B'D]




$B'D = [(B'D)']' = (B+D')' =$ B NOR D'



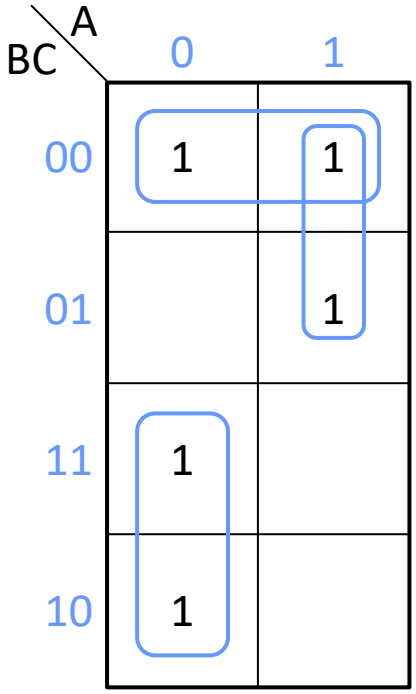

OR - AND

NOR - NOR

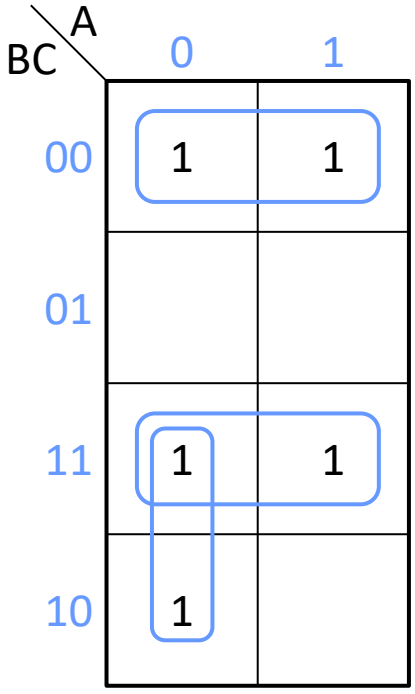
OR-AND Gate 就是 NOR-NOR Gate.

# Example 2: Multiple Outputs (1/2)

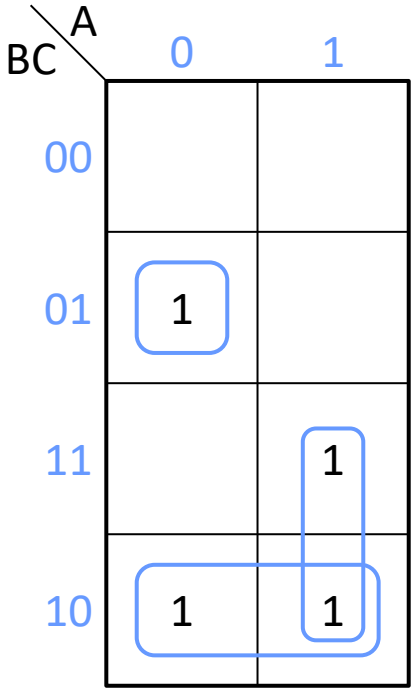
- Realize the functions using only 2-input NAND gates and inverters




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



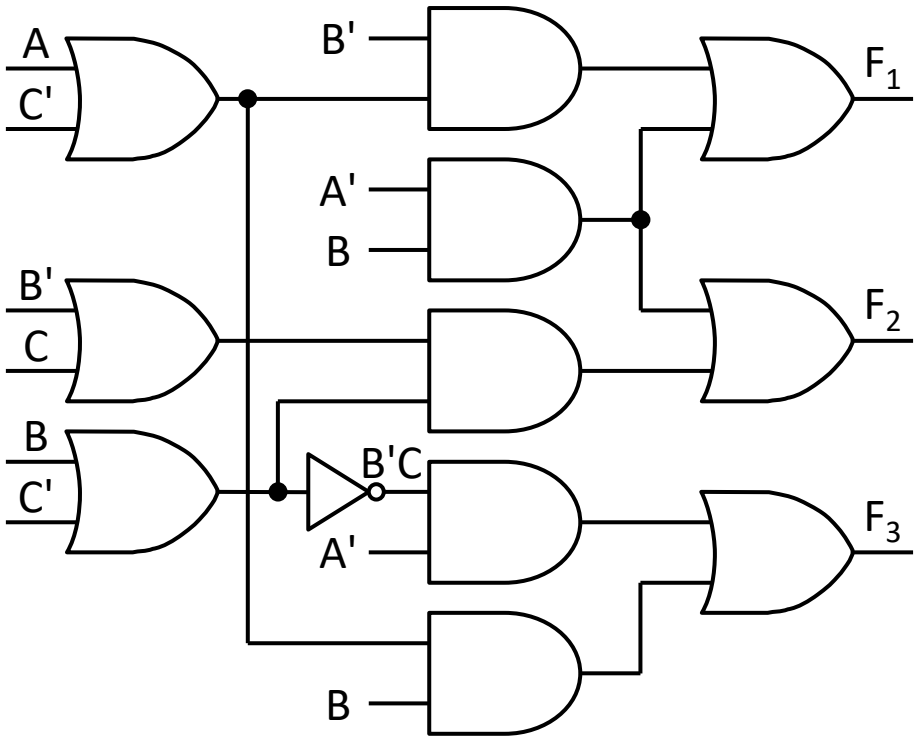

$F_2 = B'C' + BC + A'B$

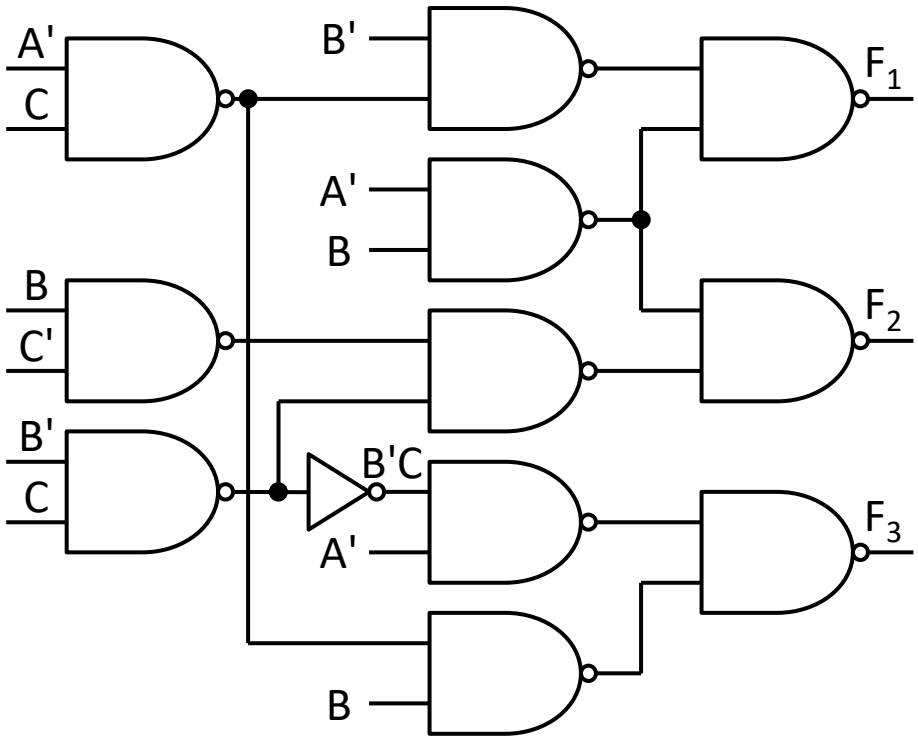



$F_3 = \underline{A'B'C} + AB + BC'$

# Example 2: Multiple Outputs (2/2)

- Two-level (NOT不限制)
  - $F_1 = B'C' + AB' + A'B = B'(A + C') + A'B$
  - $F_2 = B'C' + BC + A'B = A'B + (B' + C)(B + C')$
  - $F_3 = \underline{A'B'C} + AB + BC' = \underline{A'(B + C')'} + B(A + C')$



# Outline

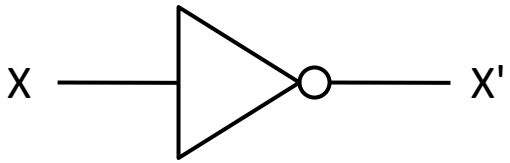
- ❑ Review of Combinational Circuit Design
- ❑ Design of Circuits with Limited Gate Fan-In
- ❑ **Gate Delays and Timing Diagrams**
- ❑ Hazards in Combinational Logic
- ❑ Simulation and Testing of Logic Circuits

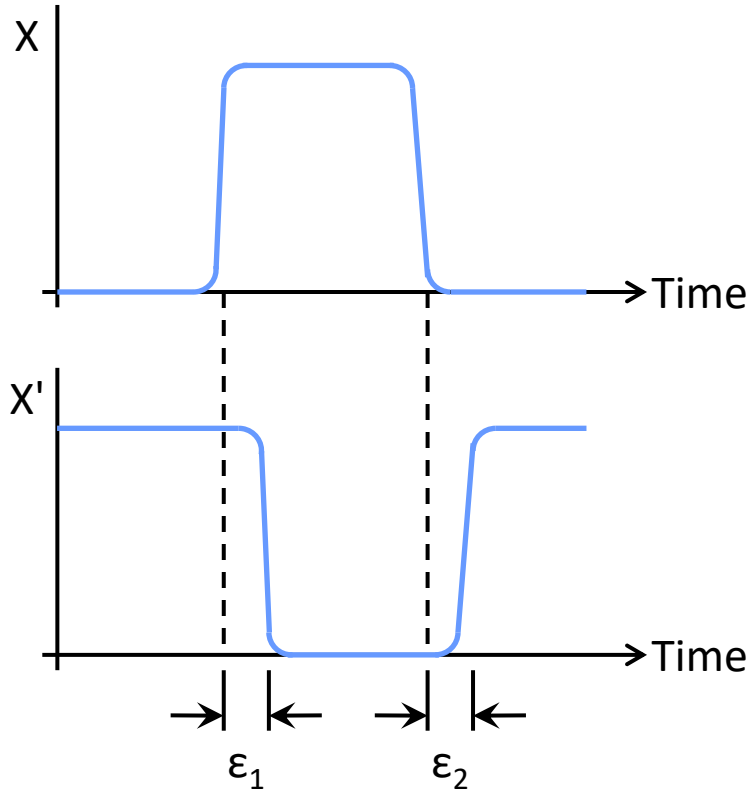
# Gate Delay

## Propagation delay

- When the input to a logic gate is changed, the output takes some time to change value
- Usually different for input rising 0 → 1 and falling 1 → 0

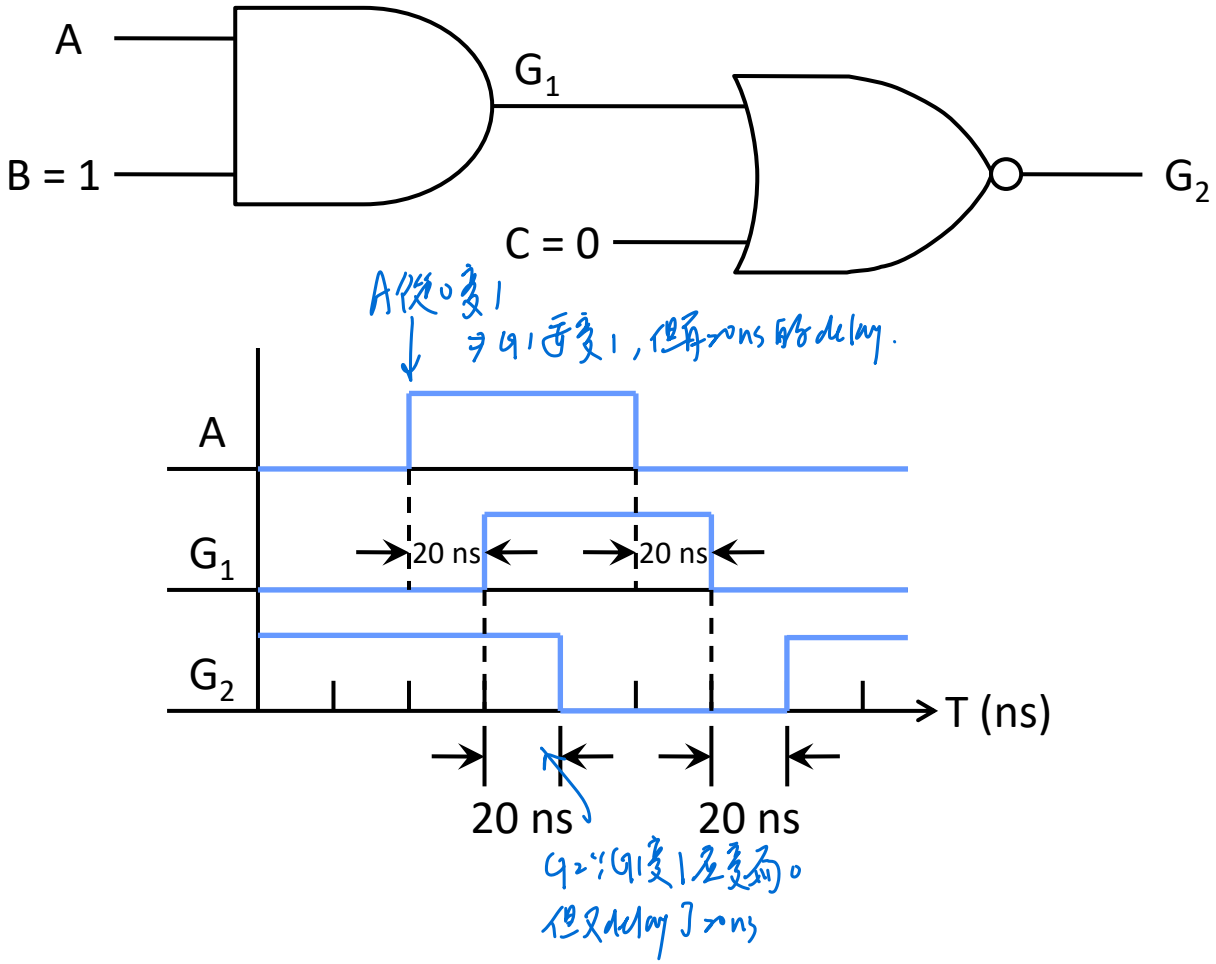
rising 和 falling delay 还可能不一样长 (门物理特性)



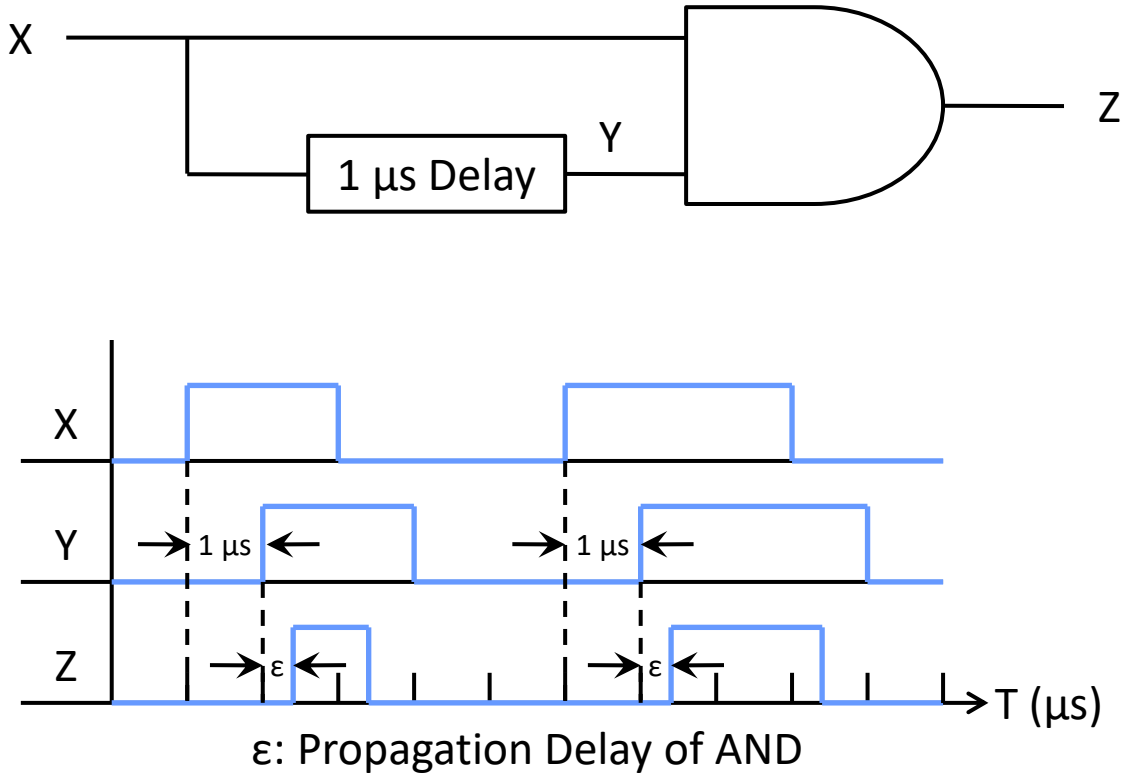
# Sample Timing Diagram (1/2)

- Assume each gate has a propagation delay of 20 ns



# Sample Timing Diagram (2/2)

- A circuit with delay element

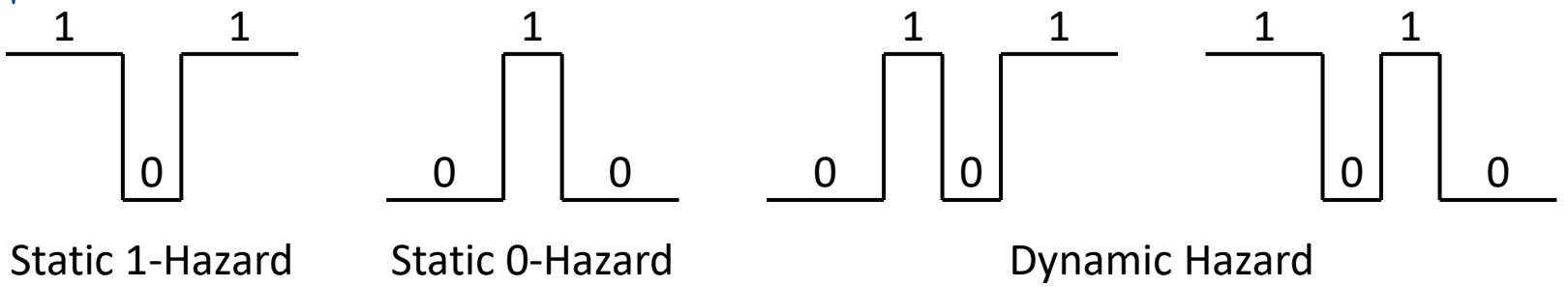


# Outline

- ❑ Review of Combinational Circuit Design
- ❑ Design of Circuits with Limited Gate Fan-In
- ❑ Gate Delays and Timing Diagrams
- ❑ **Hazards in Combinational Logic**
- ❑ Simulation and Testing of Logic Circuits

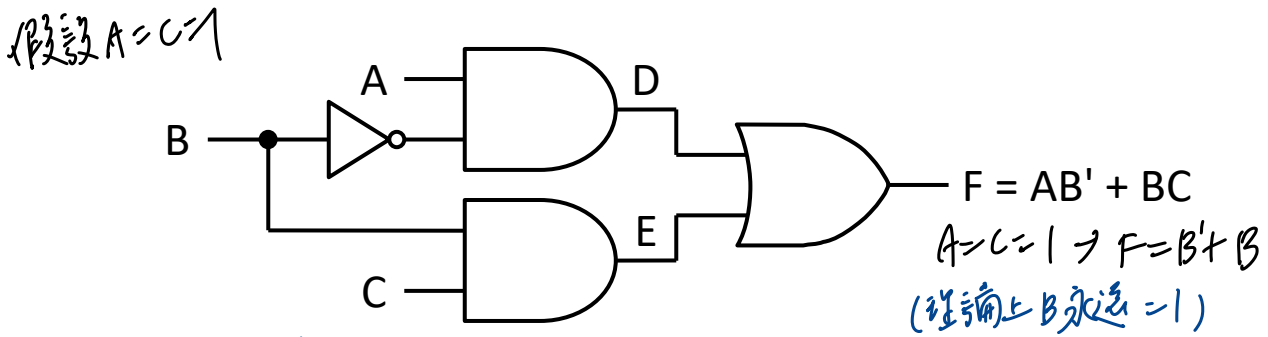
# Hazards

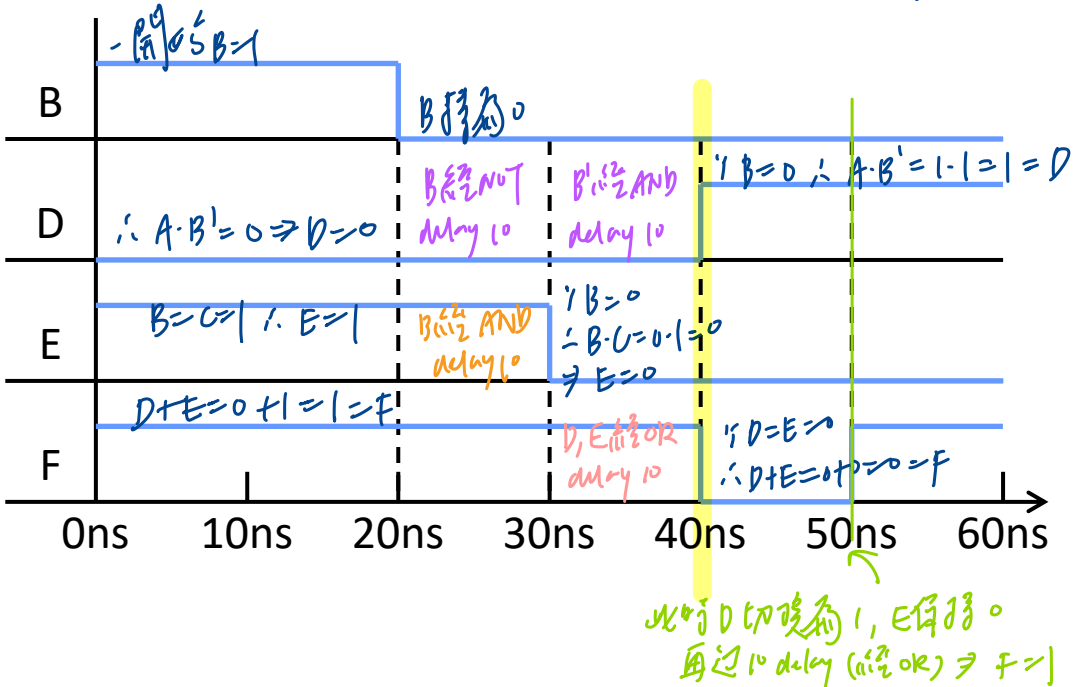
- **Hazard**
  - Unwanted switching transients appearing at the output when the input to a combinational circuit changes
    - Unequal propagation delays for different paths from inputs to outputs
  - Static 1-/0-hazard
    - Output momentarily goes to 0/1 when it should remain a constant 1/0
  - Dynamic hazard
    - Output change 3 or more times when the output changes from 0 to 1 (1 to 0)

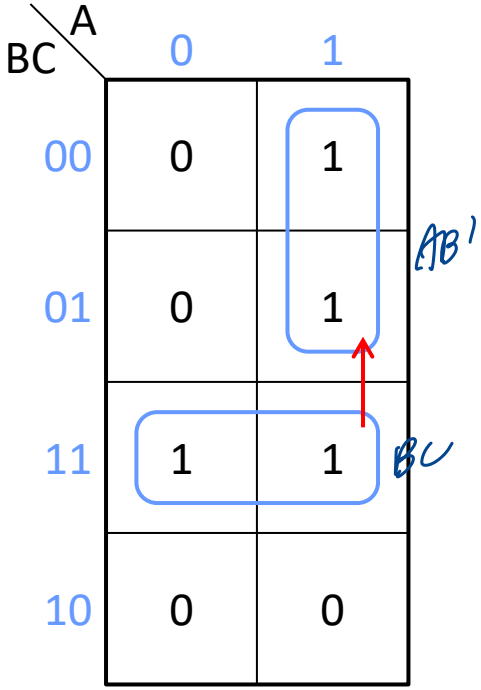


# Static 1-Hazard Example

- Example: assume each gate has a propagation delay of 10 ns
  - If A = C = 1, static-1 hazard occurs when B changes from 1 to 0



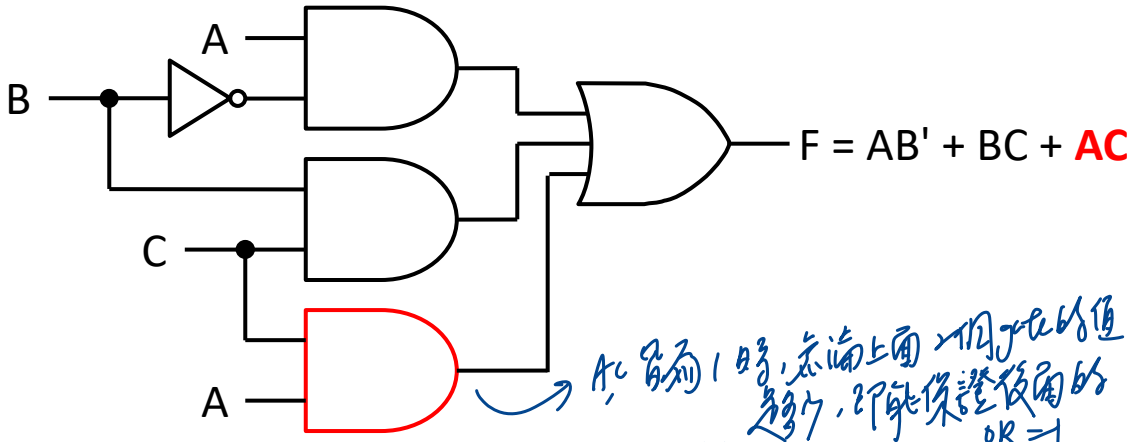
# Static 1-Hazard Removal

## When?

- In Karnaugh maps, if any two adjacent 1's are not covered by the same loop, a 1-hazard exists for the transition between the two 1's

## How to remove hazards?

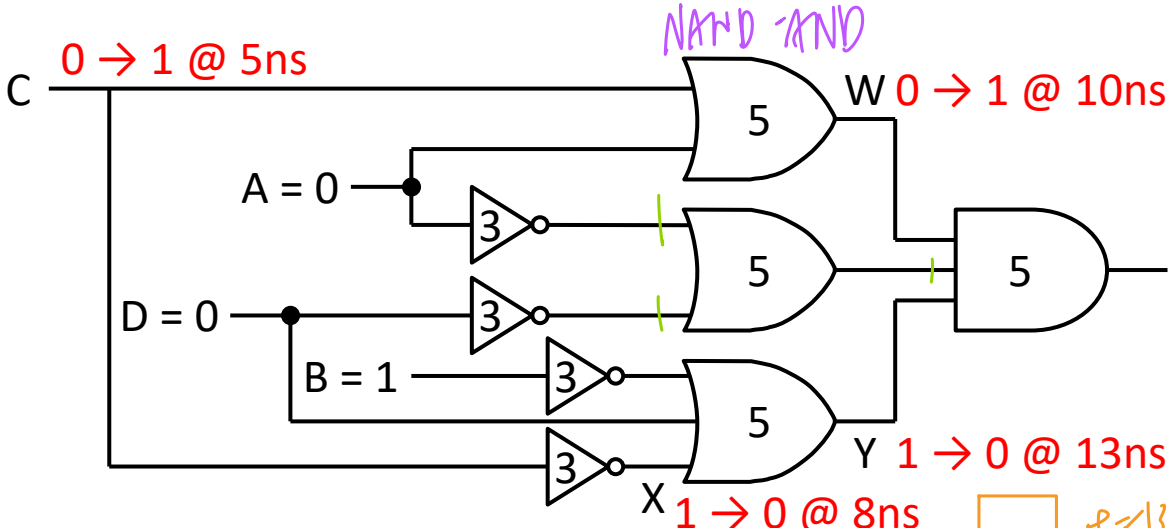
- Add additional loops such that all adjacent 1's are covered by some loop



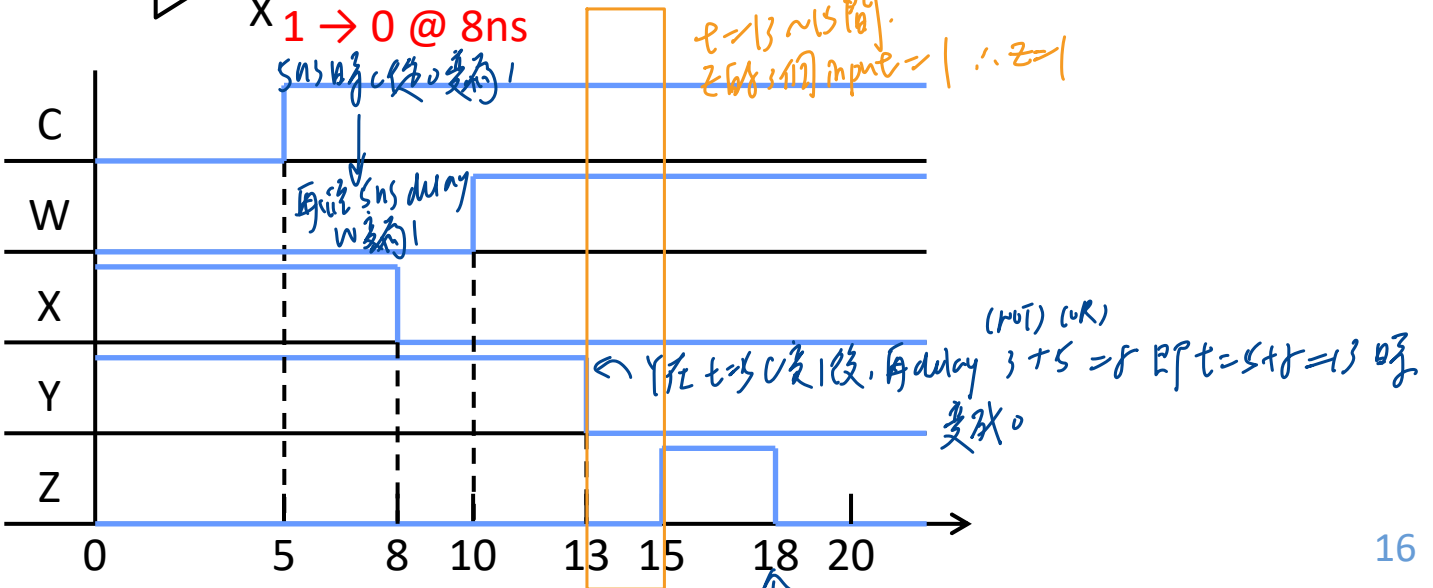

| BC \ A | 0 | 1 |
|---|---|---|
| 00 | 0 | 1 |
| 01 | 0 | 1 |
| 11 | 1 | 1 |
| 10 | 0 | 0 |

# Static 0-Hazard Example

- Example: A = 0, B = 1, D = 0, and C changes from 0 to 1

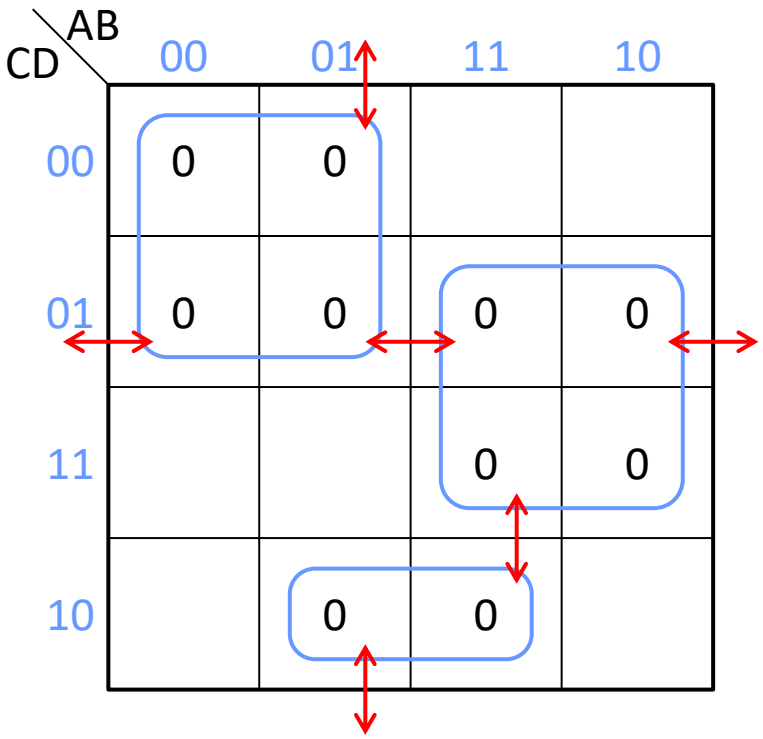



Propagation Delay
Inverter: 3 ns
AND/OR: 5 ns

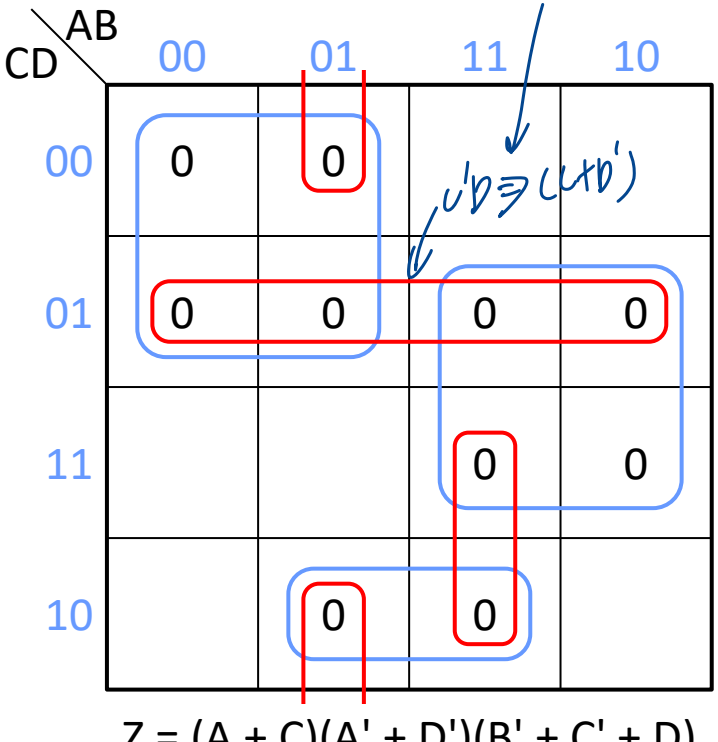


# Static 0-Hazard Removal

- Similarly, add additional loops for adjacent 0's




$Z = (A + C)(A' + D')(B' + C' + D)$




$Z = (A + C)(A' + D')(B' + C' + D)$

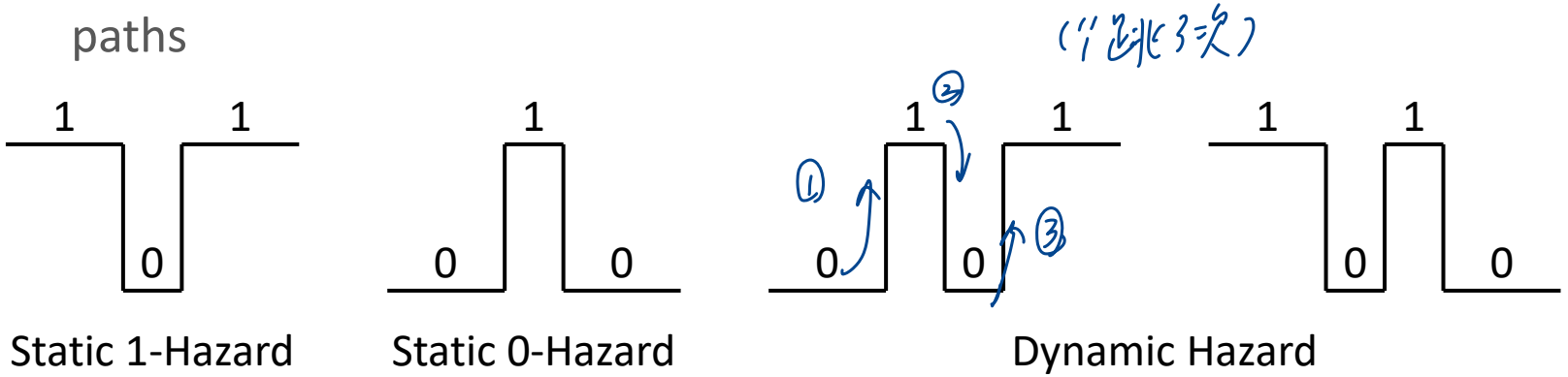
**$(C + D')(A + B' + D)(A' + B' + C')$**

# Hazards in Multilevel Logic (1/2)

- Derive a SOP/POS for a multilevel circuit, but the complementation laws (XX' = 0, X + X' = 1) are NOT used
  - Some redundant terms exist in the SOP (POS)
    - XX'α
      - α is a product of literals or it may be null
      - XX'α represents a pseudo gate that may temporarily have the output value 1 as X changes and if α = 1
    - X + X' + β
      - β is a sum of literals or it may be empty
      - X + X' + β represents a pseudo gate that may temporarily have the output value 0 as X changes and if β = 0
- For SOP, ensure every pair of adjacent 1's is covered together
  - The sum of all prime implicants is safe!
- Treat each X and X' as independent variables to prevent introduction of hazards

# Hazards in Multilevel Logic (2/2)

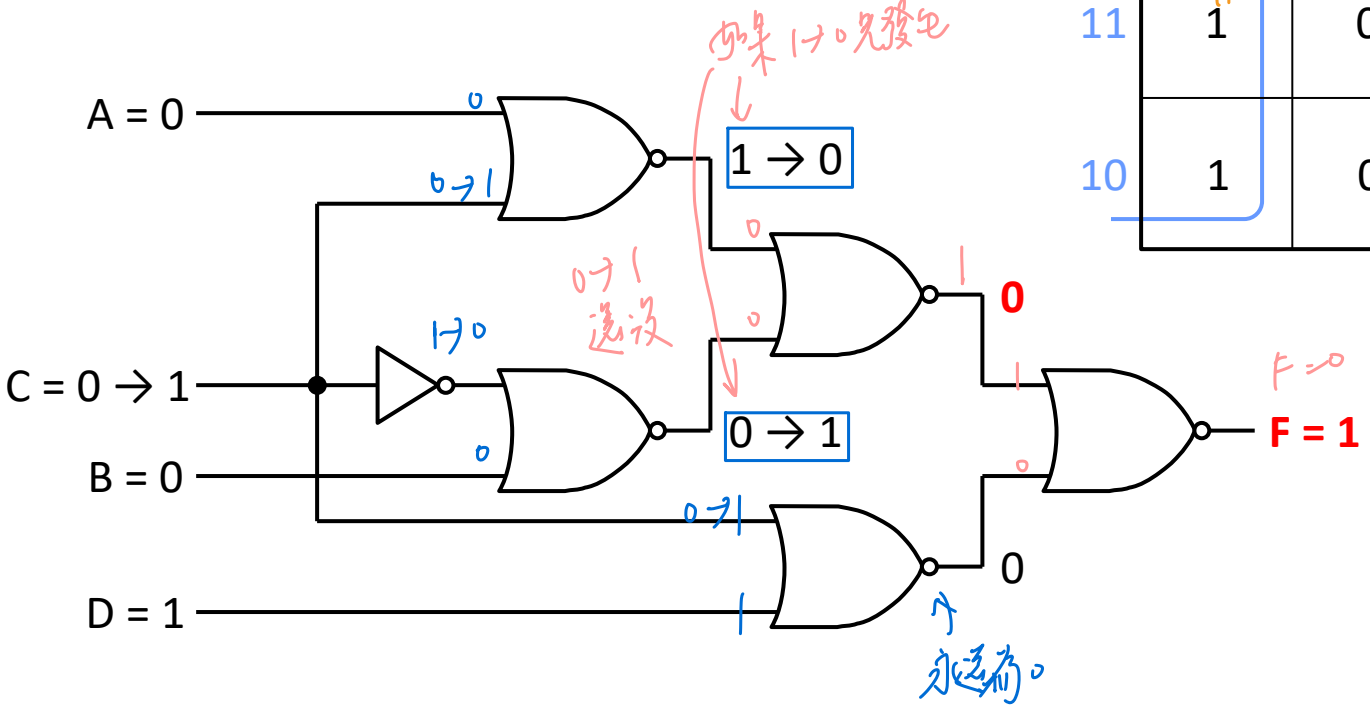
- Static 1-hazards: analyze the same as for 2-level
- Static 0-hazards: XX'α in SOP
  - Adjacent 0's differ in the value of X with α = 1
- Dynamic hazards: XX'α in SOP
  - Adjacent input combinations on K-map differ in the value of X with α = 1 and with opposite function values, and
  - For these combinations, the change in X propagates over ≥ 3 different paths



# Example (1/3)

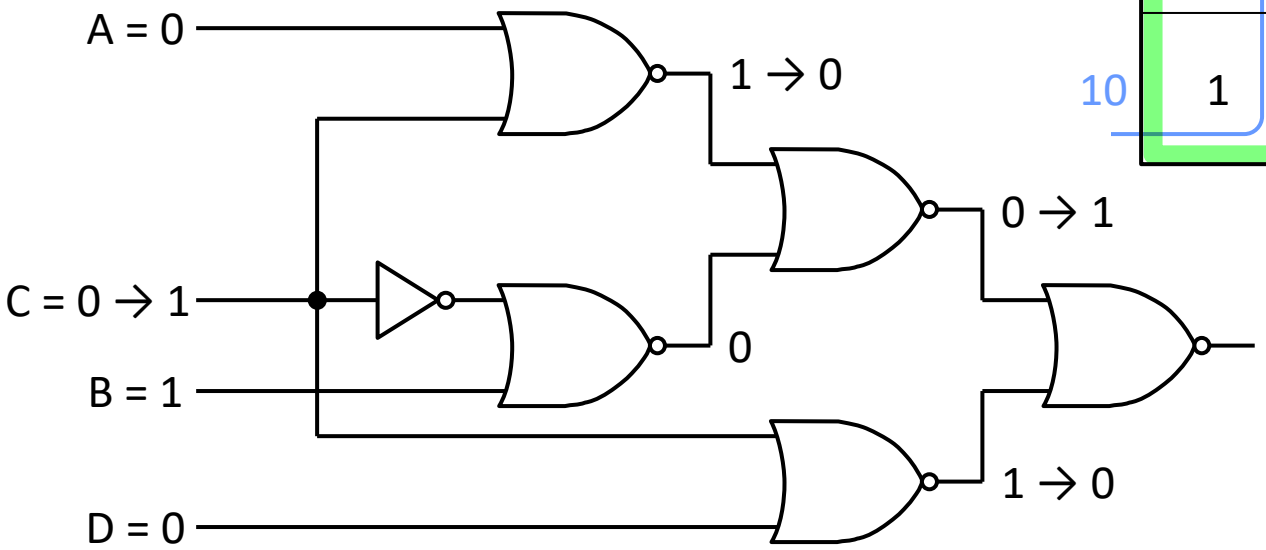
- F = (A'C' + B'C)(C + D)
  = A'C'C + A'C'D + B'C + B'CD
  = A'C'C + A'C'D + B'C
- Static 1-hazard
  - 0001 ↔ 0011

| CD \ AB | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 1 | 1 | 0 | 0 |
| 11 | 1 | 0 | 0 | 1 |
| 10 | 1 | 0 | 0 | 1 |



# Example (2/3)

- F = A'C'C + A'C'D + B'C
- Static 0-hazard
  - Adjacent 0's differ in the value of C with A' = 1
  - 0100 ↔ 0110



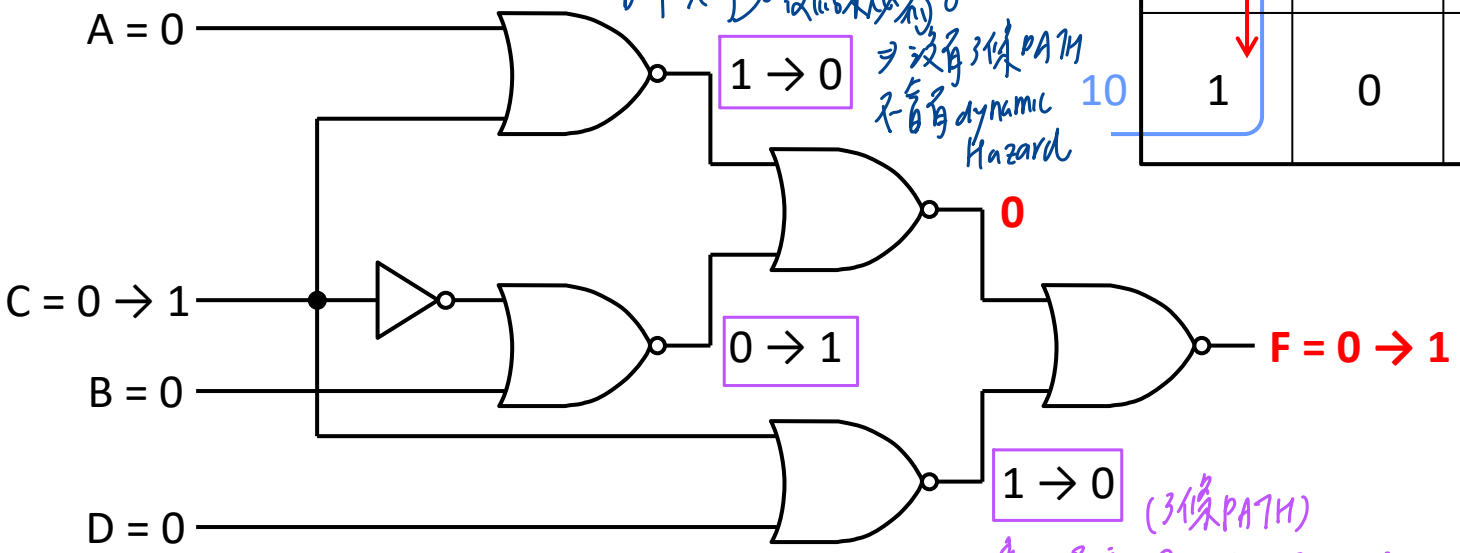

| CD \ AB | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 1 | 1 | 0 | 0 |
| 11 | 1 | 0 | 0 | 1 |
| 10 | 1 | 0 | 0 | 1 |

# Example (3/3)

dynamic 才有辦法的一個一個！

- F = A'C'C + A'C'D + B'C
  - Adjacent input combinations on K-map differ in the value of C with A' = 1 and with opposite function values, and
  - For these combinations, the change in C propagates over ≥ 3 different paths
  - 0000 ↔ 0010 (how about 0101 ↔ 0111?)

| CD \ AB | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 1 | 1 | 0 | 0 |
| 11 | 1 | 0 | 0 | 1 |
| 10 | 1 | 0 | 0 | 1 |



# Designing a Hazard-Free Circuit

## Hazard-free AND-OR circuit

- Find a SOP expression for the output where each pair of adjacent 1's is covered by a 1-term
  - The sum of all prime implicants always satisfies this condition
  - A two-level AND-OR circuit based on this will be free of static 1-, 0-, and dynamic hazards
- Transform the SOP expression into the desired form by simple factoring, DeMorgan's laws, etc.
  - Treat each variable X and X' as independent variables

## Hazard-free OR-AND circuit

- Alternatively, start with a POS form where every pair of adjacent 0's is covered by a 0-term
- Dual procedure

# Outline

- ❑ Review of Combinational Circuit Design
- ❑ Design of Circuits with Limited Gate Fan-In
- ❑ Gate Delays and Timing Diagrams
- ❑ Hazards in Combinational Logic
- ❑ **Simulation and Testing of Logic Circuits**

# Simulation

- Verify logic circuits by
  - Actually building them
  - Simulating them on a computer
- 4 logic values
  - 0 (low), 1 (high), X (unknown), Z (high-impedance/open circuit)
- AND and OR functions for 4-valued simulation

| AND | 0 | 1 | X | Z |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | X | X |
| X | 0 | X | X | X |
| Z | 0 | X | X | X |

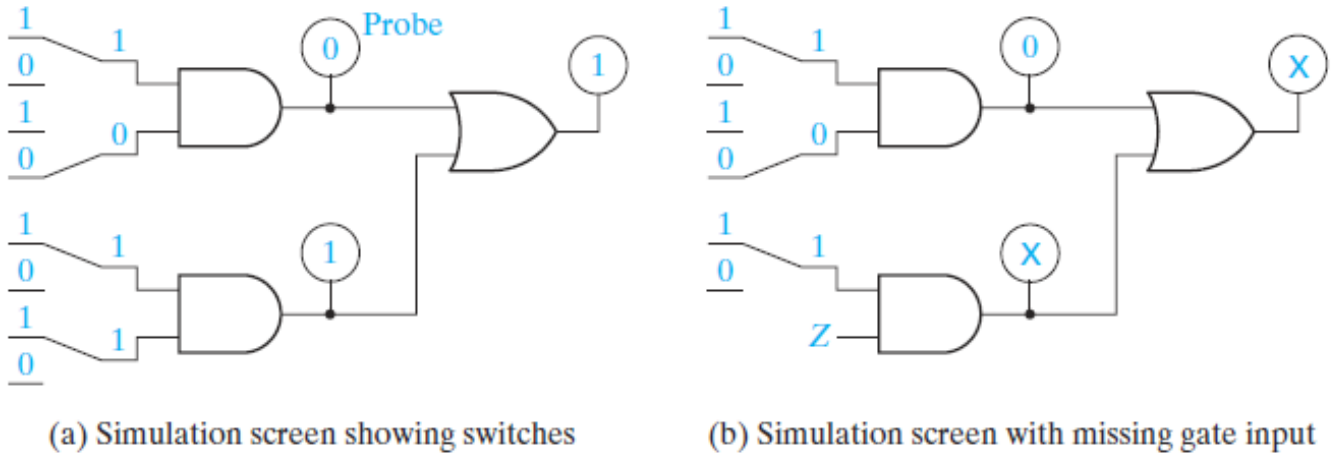
| OR | 0 | 1 | X | Z |
|---|---|---|---|---|
| 0 | 0 | 1 | X | X |
| 1 | 1 | 1 | 1 | 1 |
| X | X | 1 | X | X |
| Z | X | 1 | X | X |

# 4-Valued Logic Simulator (1/2)

- Simulate a circuit
  - Computations: level-by-level from inputs to outputs
  - Evaluations: once inputs change

FIGURE 8-13






(a) Simulation screen showing switches

(b) Simulation screen with missing gate input

# 4-Valued Logic Simulator (2/2)

- Possible errors in designs
  - In simulation 更好等去 ISIM 製造以前
    - Incorrect design
    - Gates with wrong connection
    - Wrong input signals to the circuits
  - In real circuits (testing) 晶片製造之後
    - Defective gates
    - Defective connecting wires
- Debug
  - Start from output, work back until the trouble is located

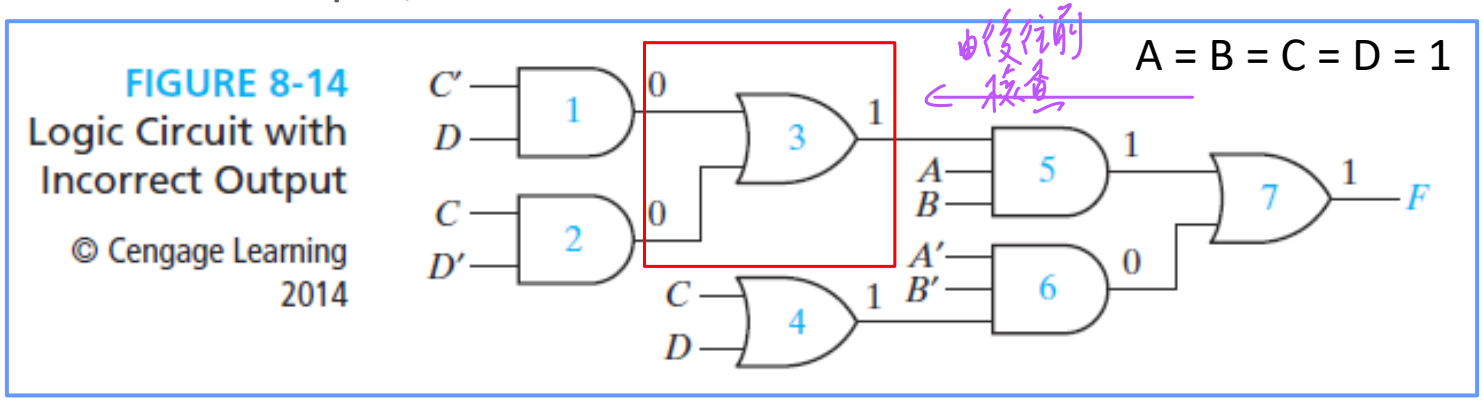



FIGURE 8-14
Logic Circuit with Incorrect Output

# Q&A