DIGITAL LOGIC(H)

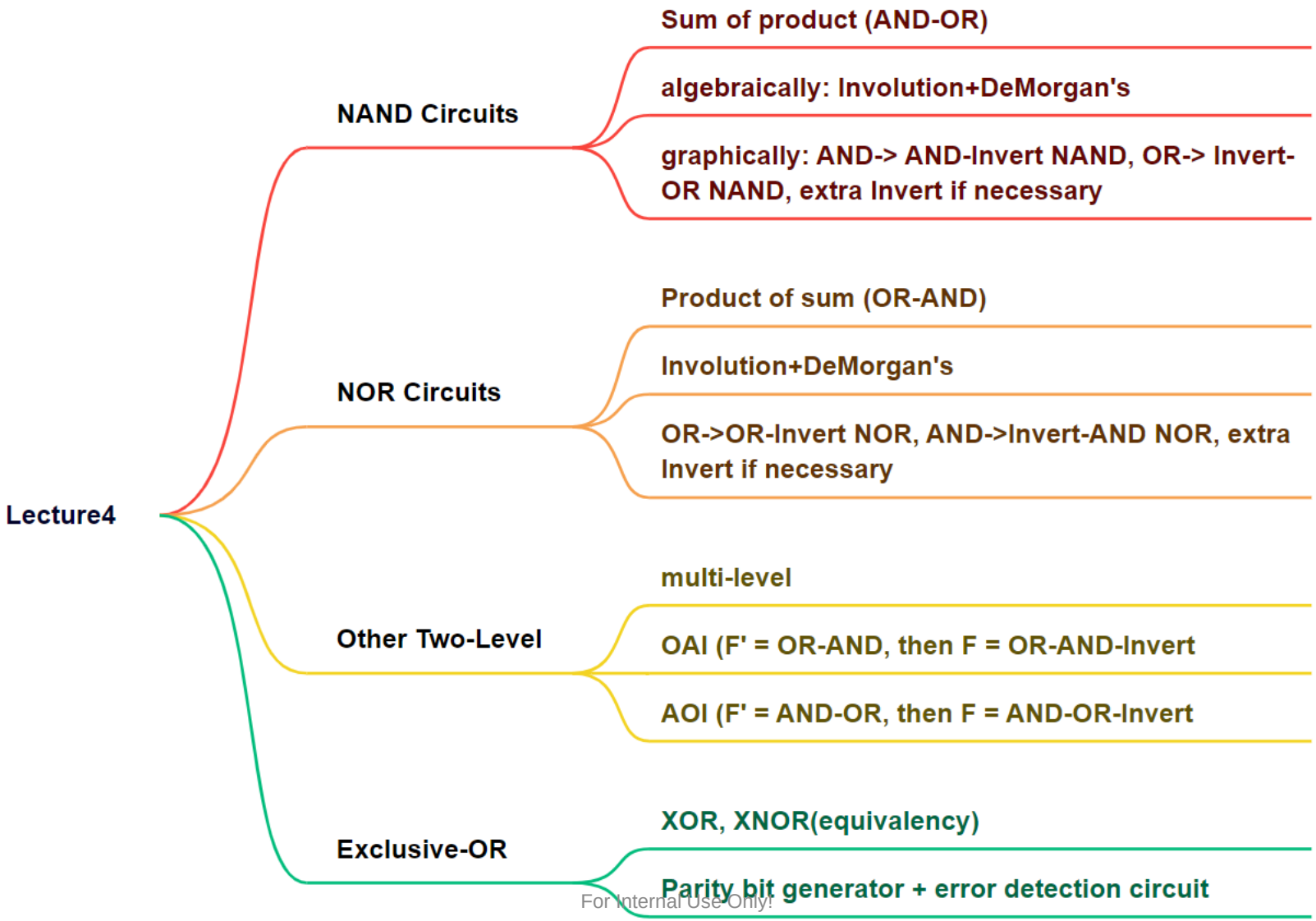
Chapter 4 part1:
Combinational Logic

2024 Fall

Today's Agenda

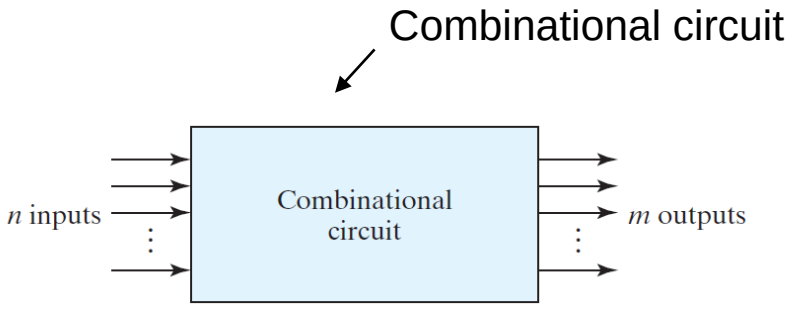
- Recap
- Context
  - Combinational Circuits
    - Analysis of Combinational Circuits
    - Design Procedure
  - Basic Components
    - Magnitude Comparator
- Reading: Textbook, Chapter 4.1-4.4, 4.8

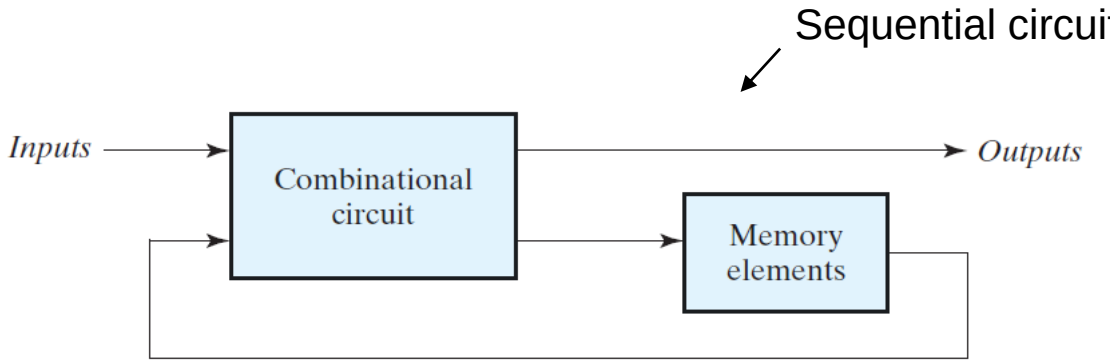
Recap



Logic Circuits for the Digital System

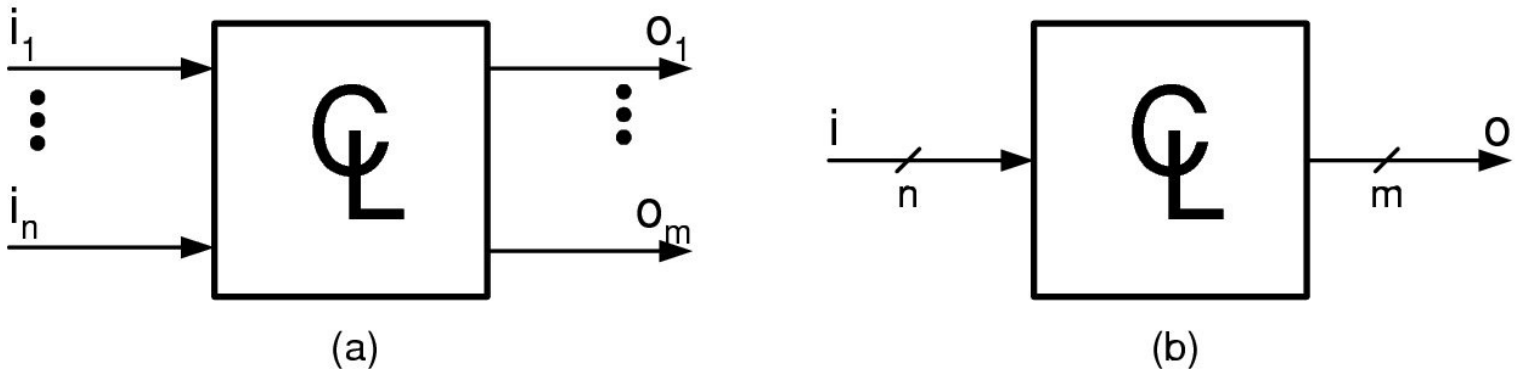
- Combinational circuits
  - Logic circuits whose outputs at any time are determined directly and only from the present input combination.
- Sequential circuits (next chapter)
  - Circuits that employ memory elements + (combinational) logic gates
  - Outputs are determined from the present input combination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

Analysis Procedure

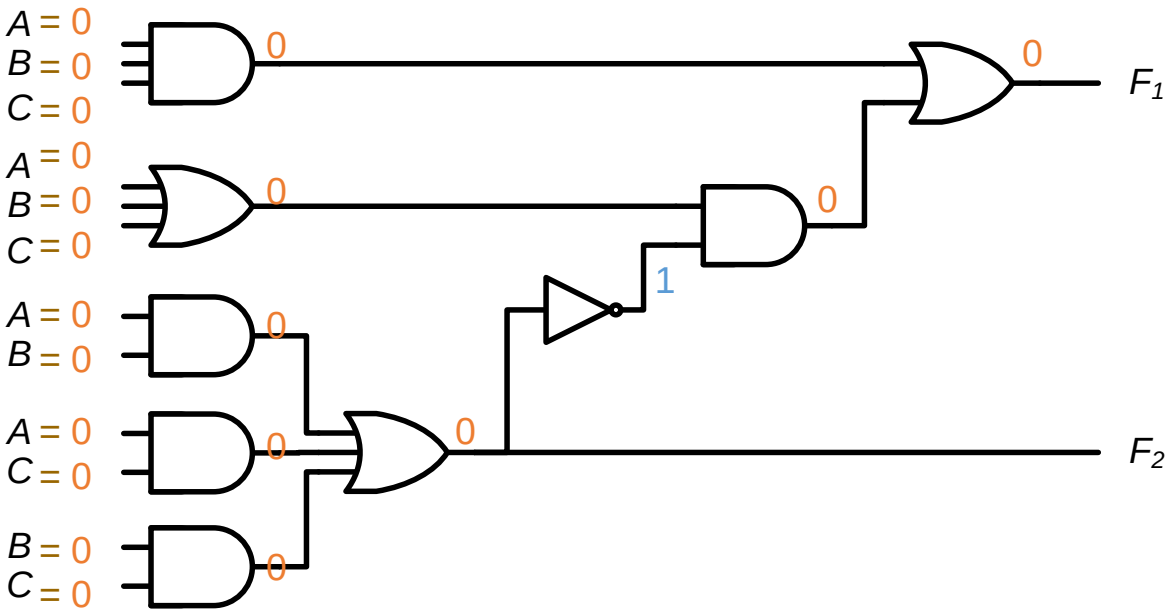
- Analysis of a combinational circuit: determine the function of the circuit.
  - From given logic diagram,
  - Aim to develop a set of Boolean functions, a truth table, an optional explanation of the circuit operation.
  - If a function name or an explanation is given along the circuit, just verify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

Analysis with Truth Table: Example

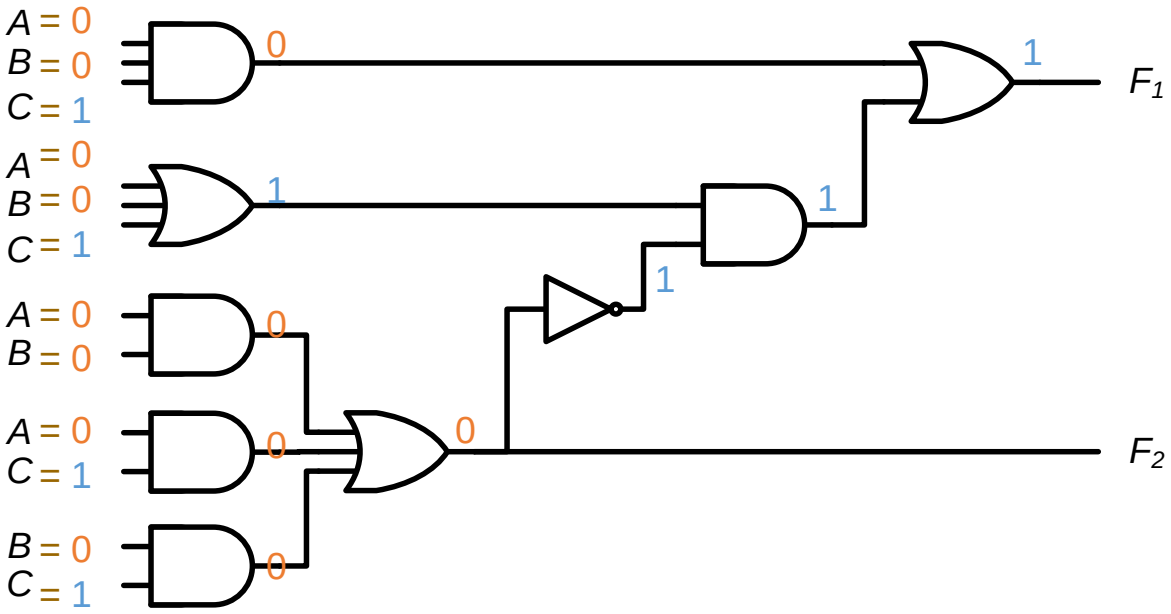
- Truth Table Approach



A B C	F_1	F_2
0 0 0	0	0

Analysis with Truth Table: Example

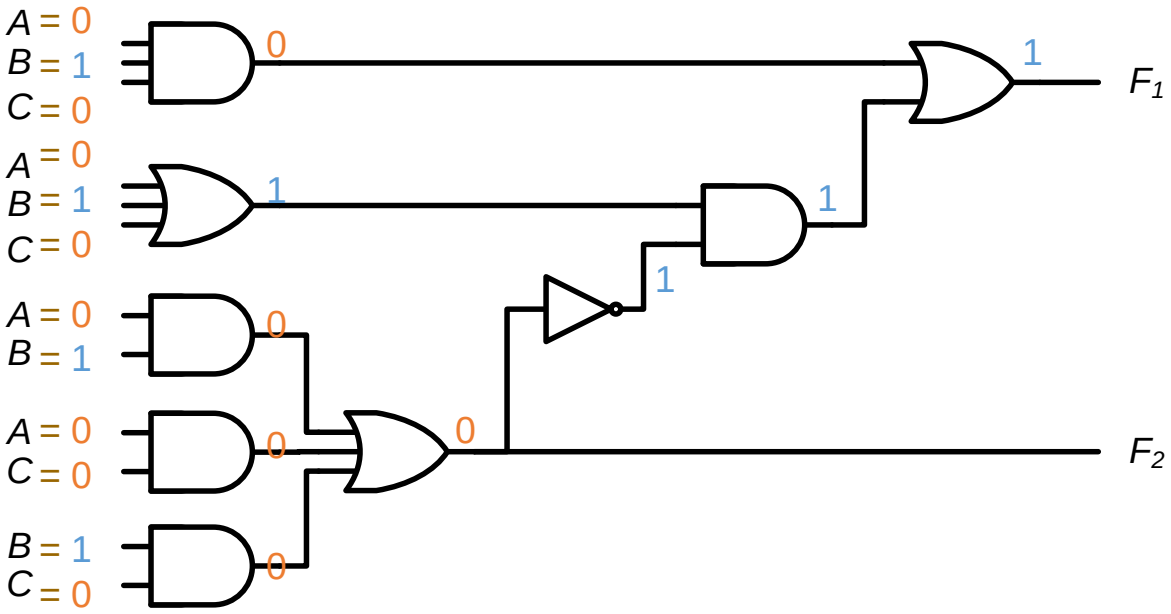
- Truth Table Approach



A B C	F_1	F_2
0 0 0	0	0
0 0 1	1	0

Analysis with Truth Table: Example

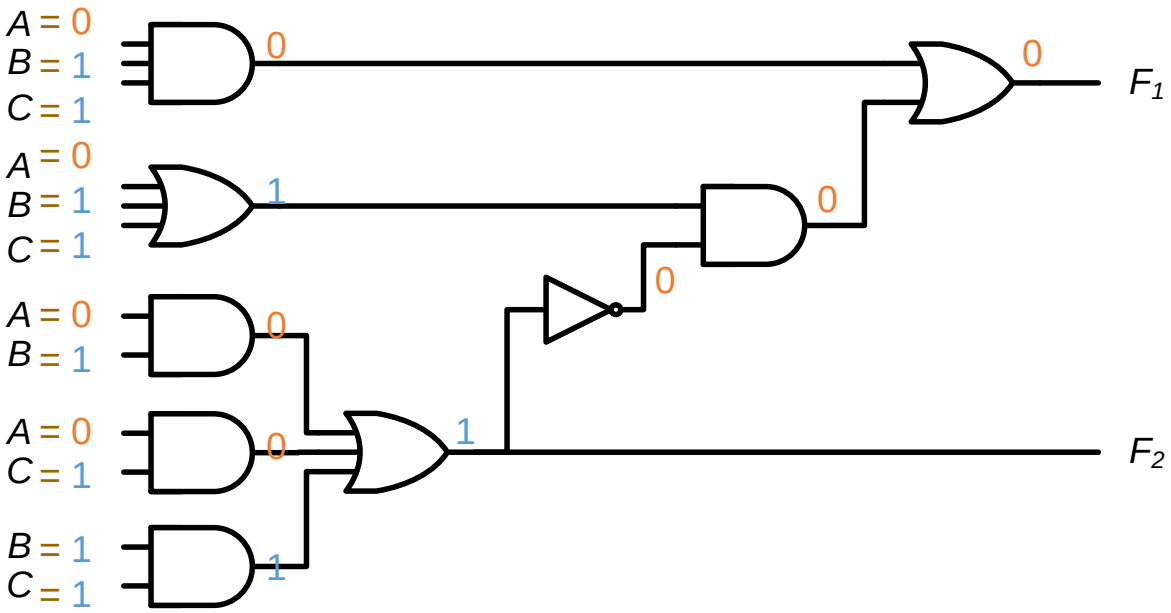
- Truth Table Approach



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

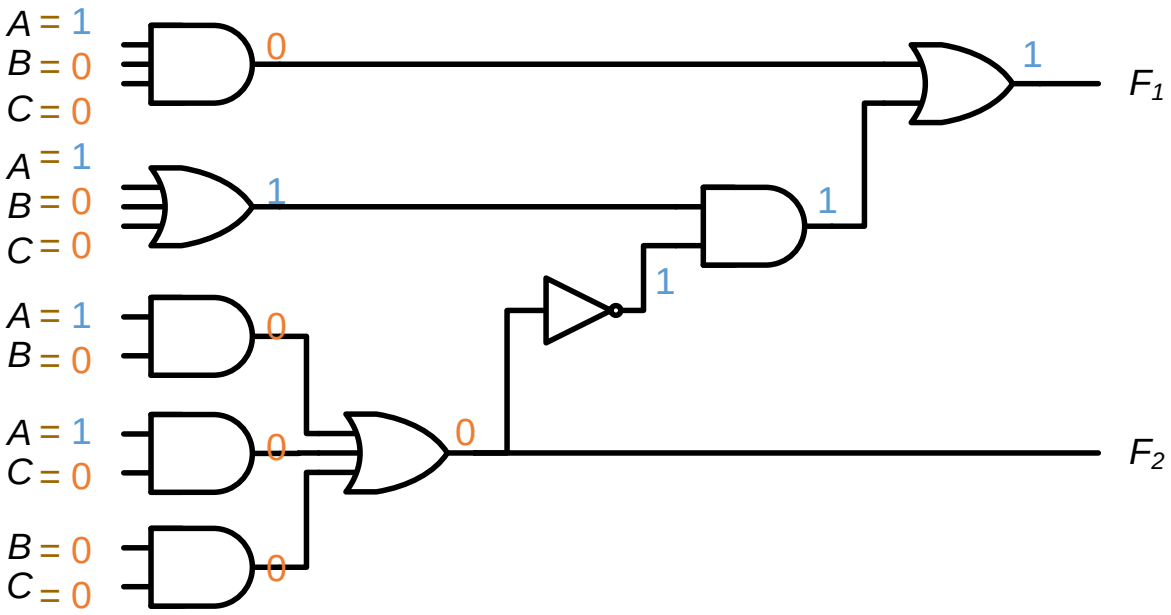
- Truth Table Approach



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

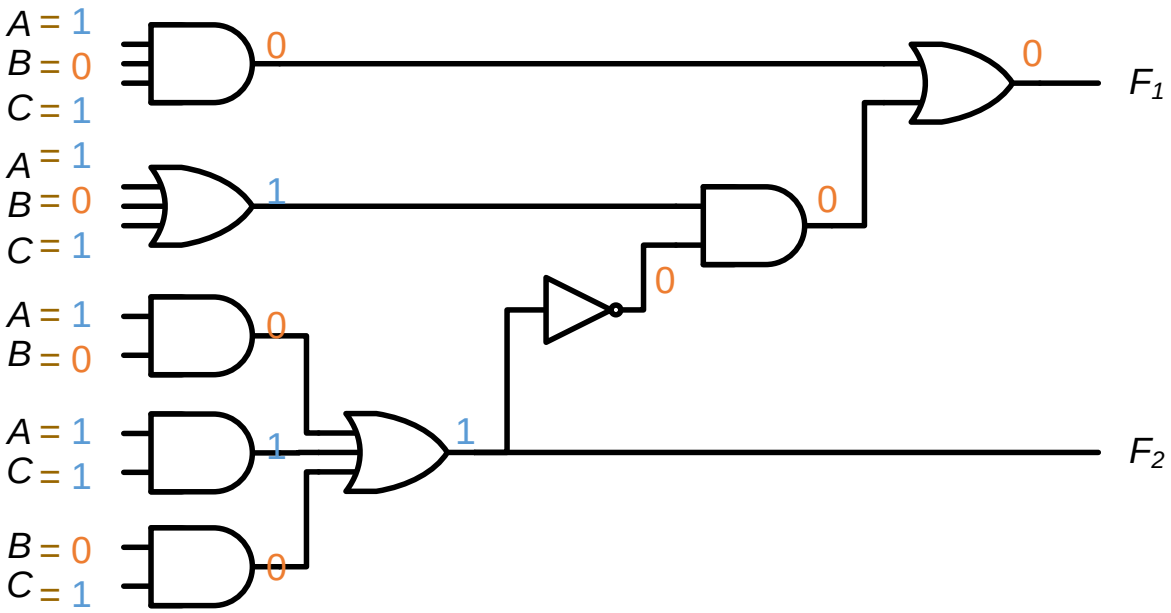
- Truth Table Approach



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

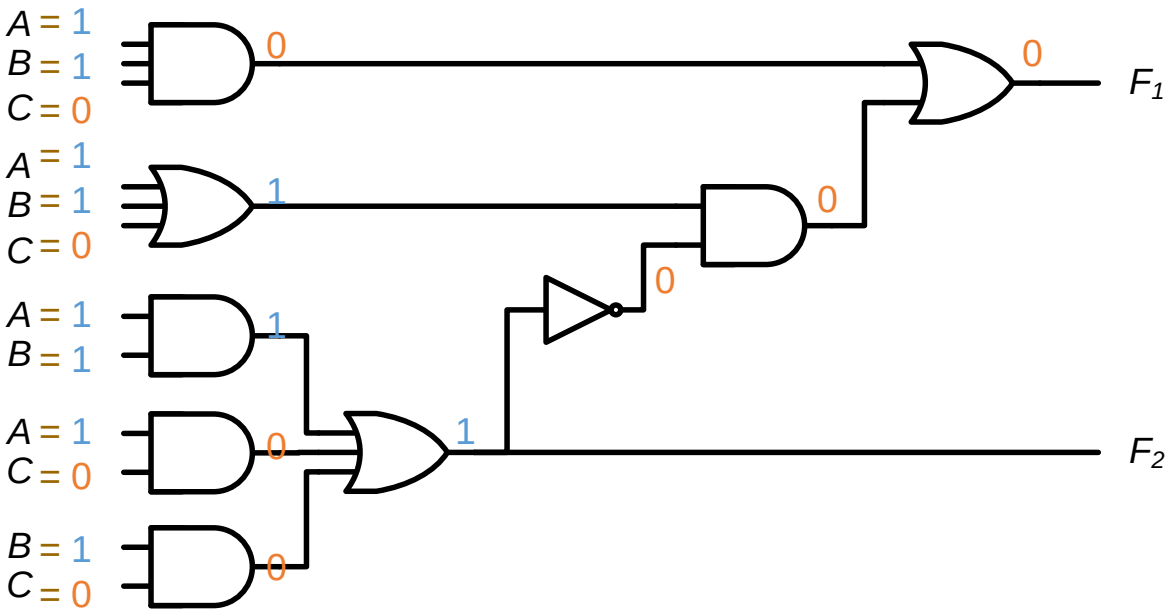
- Truth Table Approach



A	B	C	F_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

Analysis with Truth Table: Example

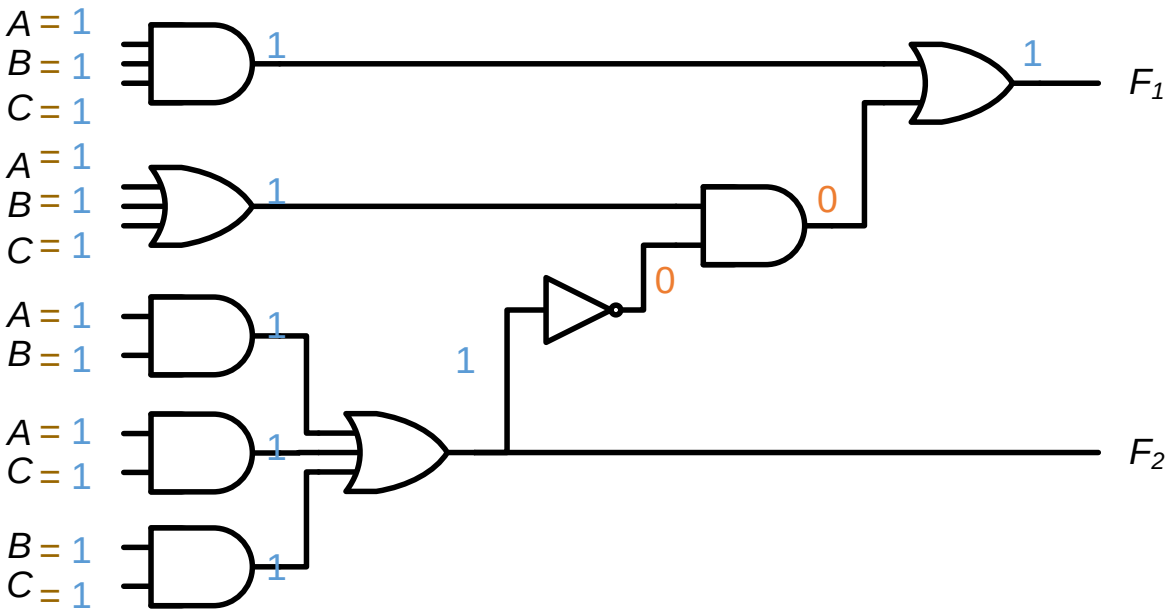
- Truth Table Approach



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

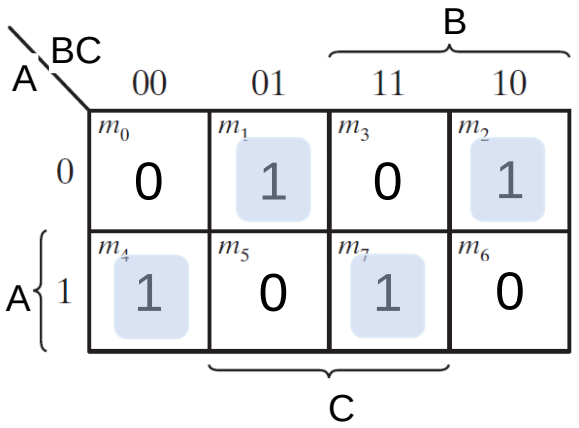


A	B	C	F_1	F_2
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

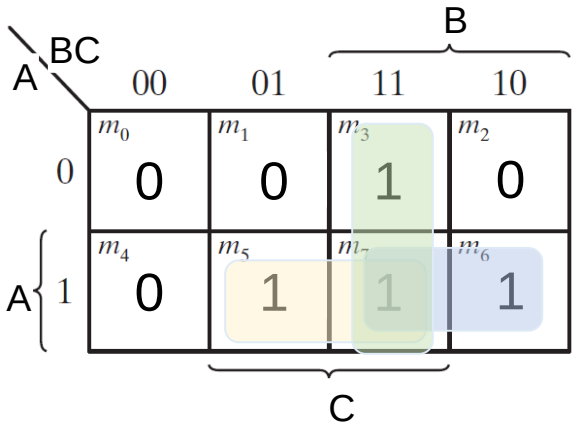
Analysis with Truth Table: Example

- 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



F_1=AB'C'+A'BC'+A'B'C+ABC
=A⊕B⊕C



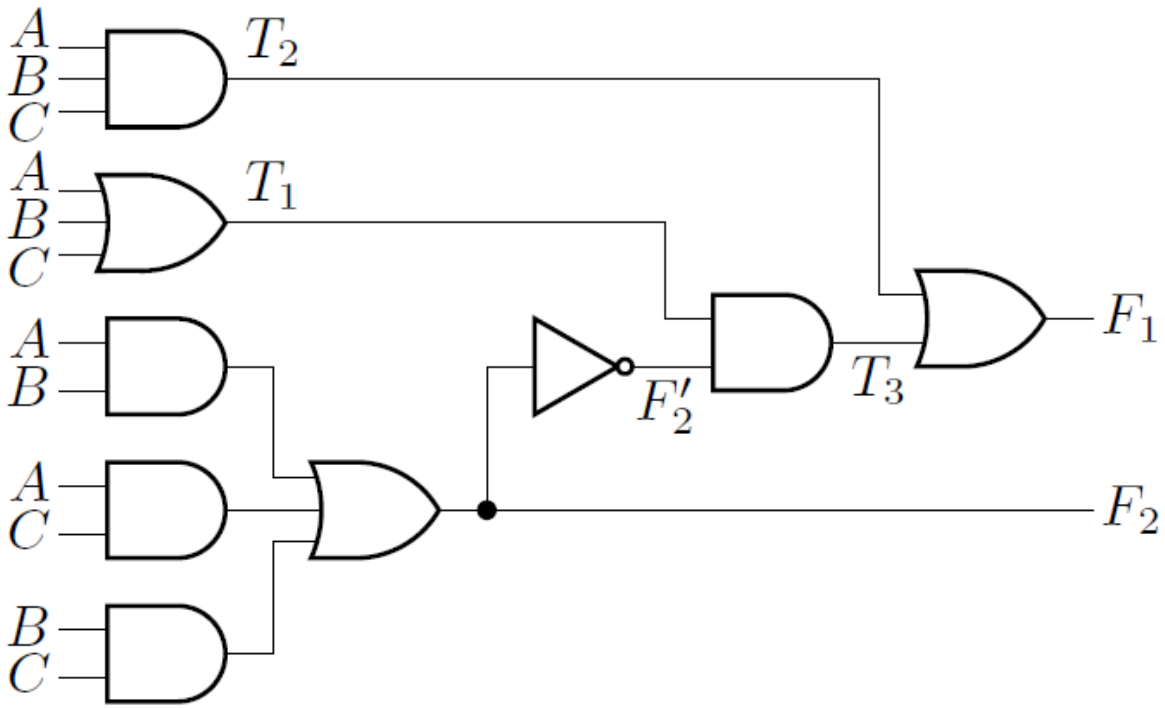
F_2=AB+AC+BC

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

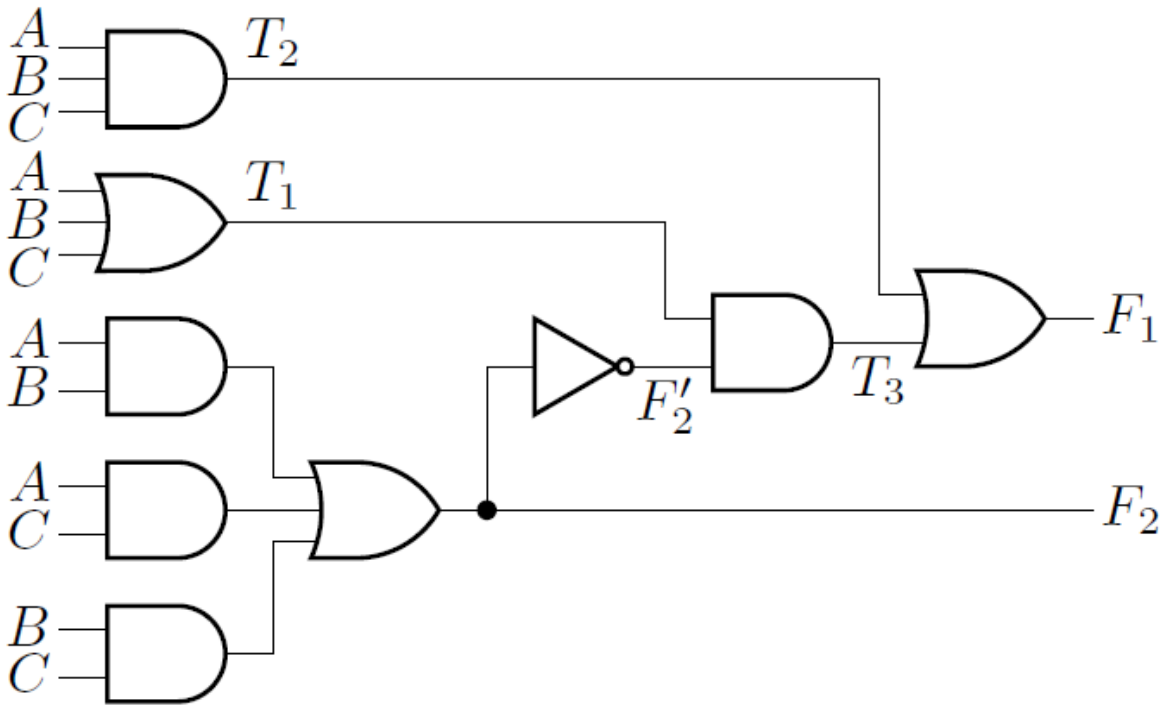
- Obtain the output Boolean functions:
  - Label all gate outputs that are a function of only inputs, no other intermediate variables. Determine their Boolean functions.



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

Analysis with Boolean Function Derivation: Example 1

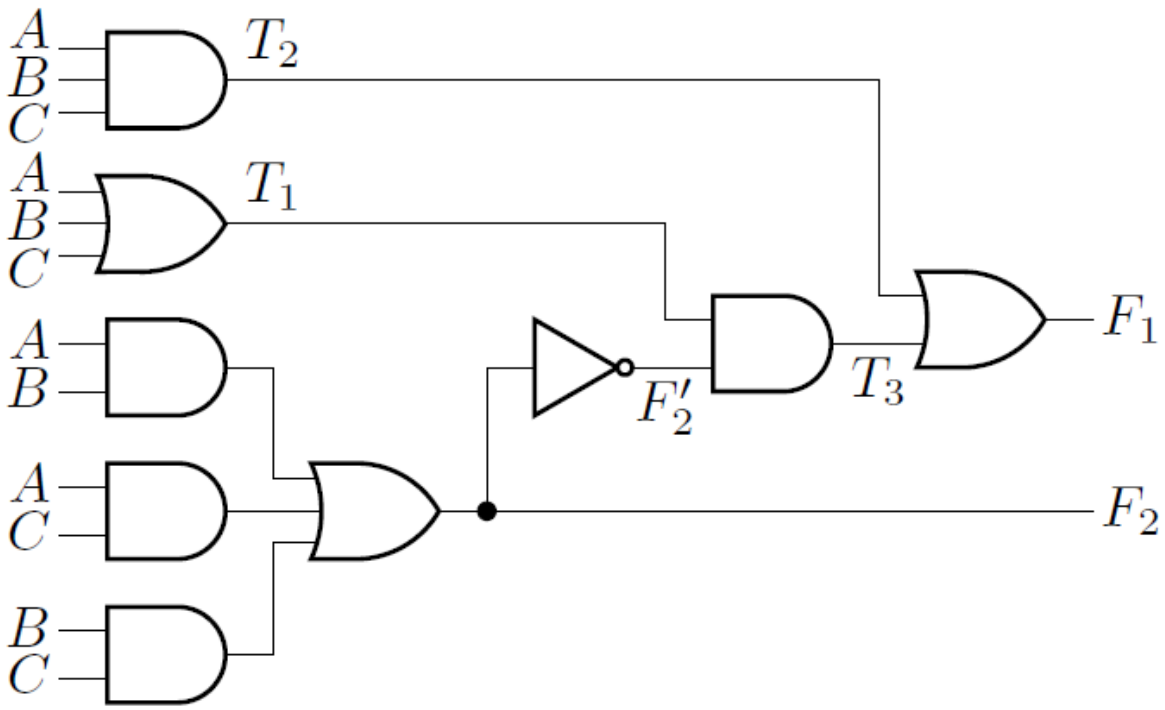
- Obtain the output Boolean functions:
  - Label all gates that are a function of inputs and the gates in the previous step. Determine their Boolean functions.



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

Analysis with Boolean Function Derivation: Example 1

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

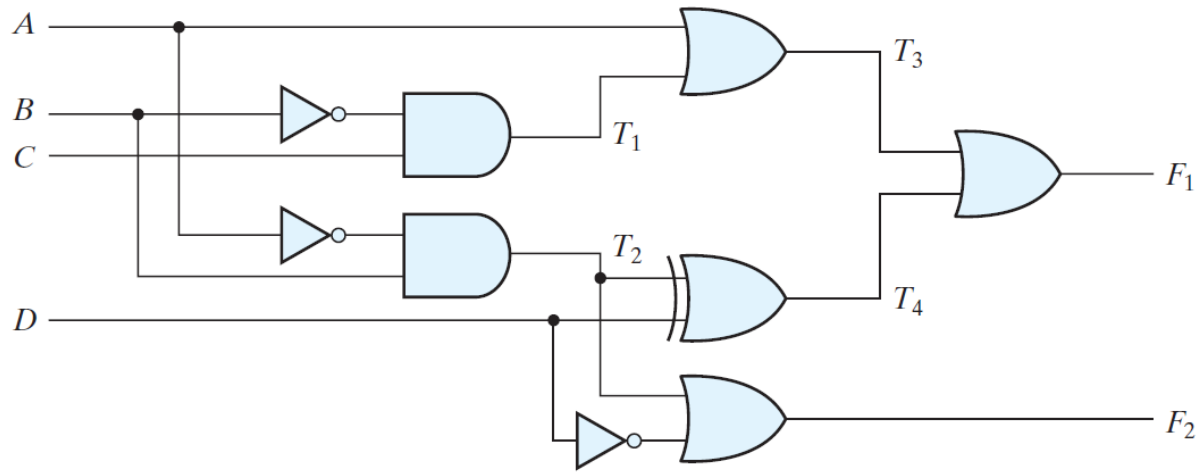


- $F_1 = T_2 + T_3$
  $= ABC+F_2'T_1$
  $= ABC+(AB+AC+BC)'(A+B+C)$
  $= A'B'C+A'BC'+AB'C'+ABC$
- $F_2 = AB+AC+BC$

- 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

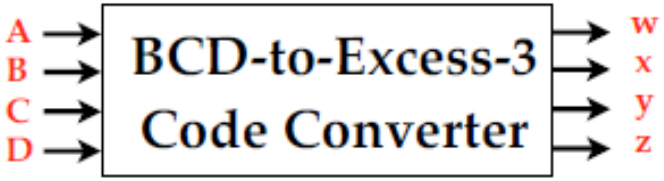
- Analysis of Combinational Circuits
- **Design of Combinational Circuits**
- Gate Delays

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

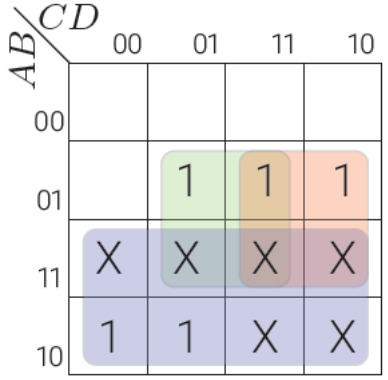
don’t care

Example1: BCD-to-Excess-3 Code Converter

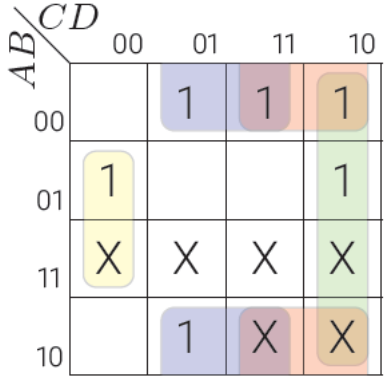
- Step3: Optimization

w=A+BC+BD
x=B’C+B’D+BC’D’
y=CD+C’D’
z=D’
from K-map

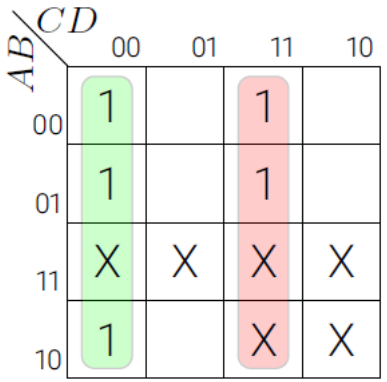
$$w = A + BC + BD$$



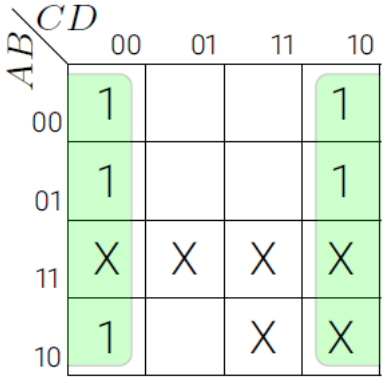
$$x = B'C + B'D + BC'D' + CD'$$



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



$$z = D'$$



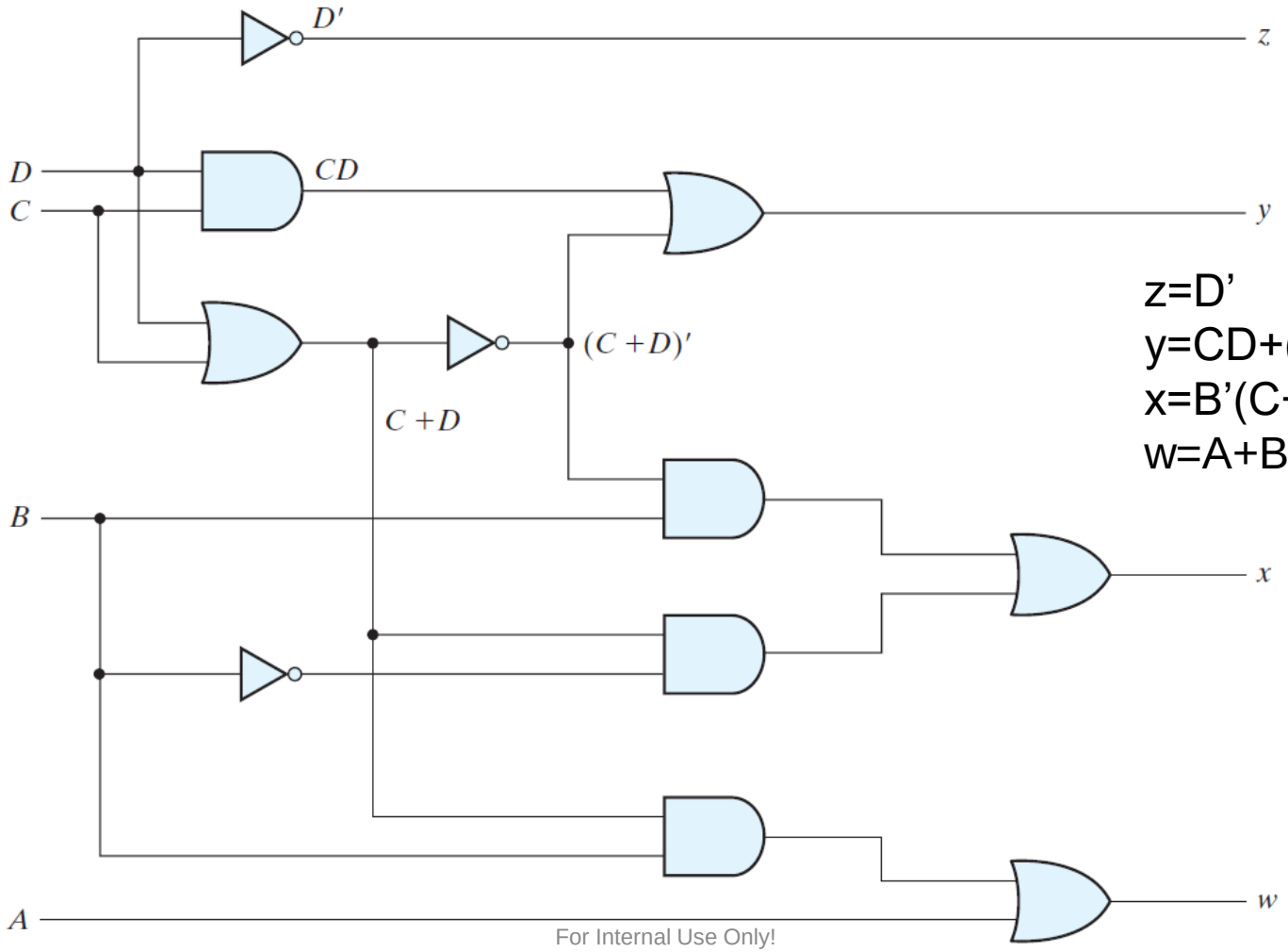
Example1: BCD-to-Excess-3 Code Converter

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

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

Example1: BCD-to-Excess-3 Code Converter

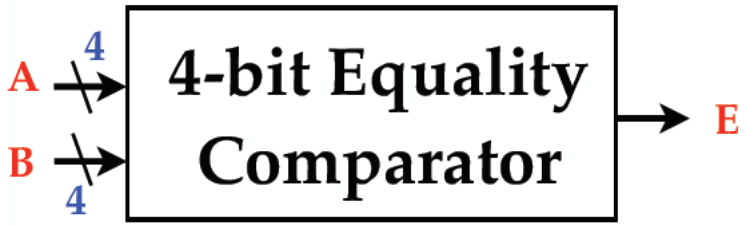
- Step4: Draw logic diagram



z=D’
y=CD+(C+D)’
x=B’(C+D)+BC’D’
w=A+B(C+D)

Example2: 4-bit Equality Comparator

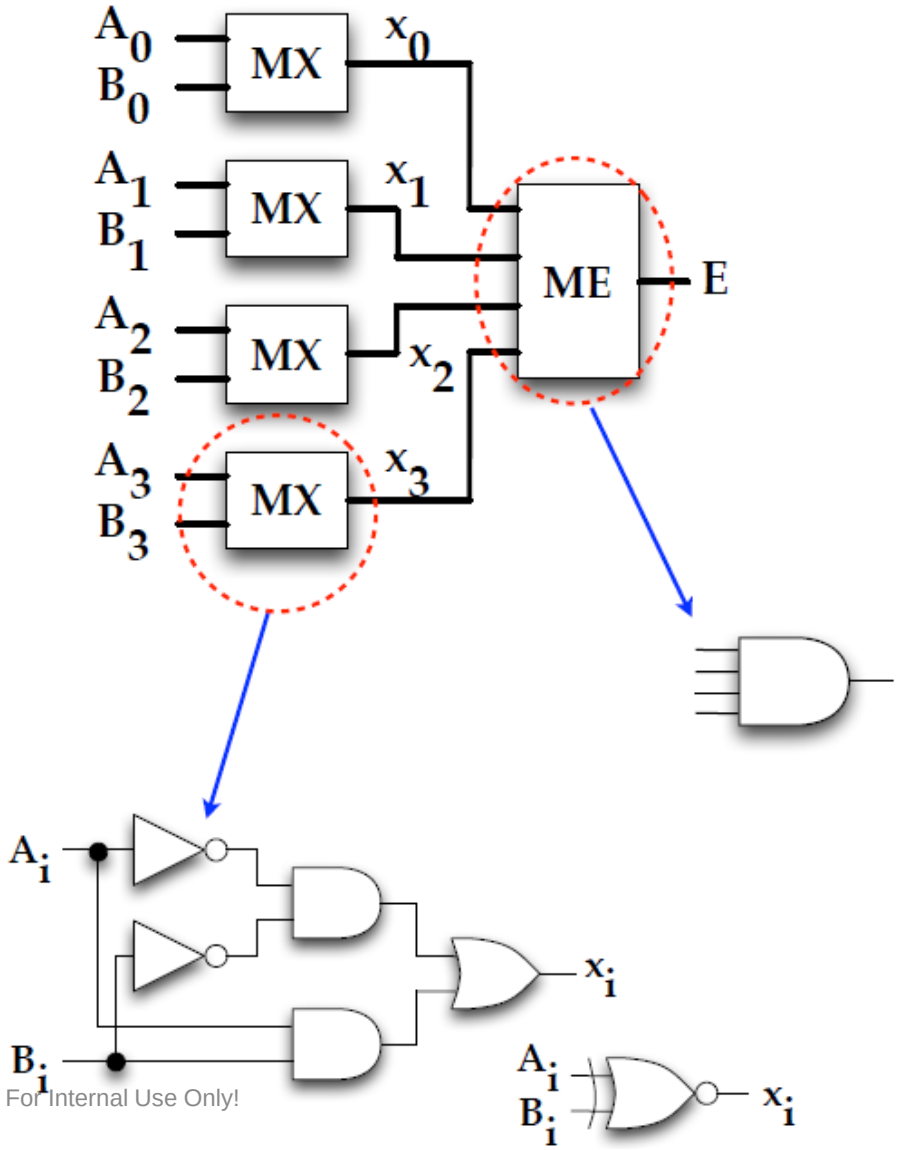
- Step1: Spec
  - input A(3:0), B(3:0);
  - output E (1/0 for equal/unequal)
- Step2: Formulation
  - By truth table: 2^8 rows => not practicable
  - By algorithm: build a regular circuit
    - 4-bit A: $A_3A_2A_1A_0$, 4-bit B: $B_3B_2B_1B_0$
    - A==B, if $(A_3==B_3)$ & $(A_2==B_2)$ & $(A_1==B_1)$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

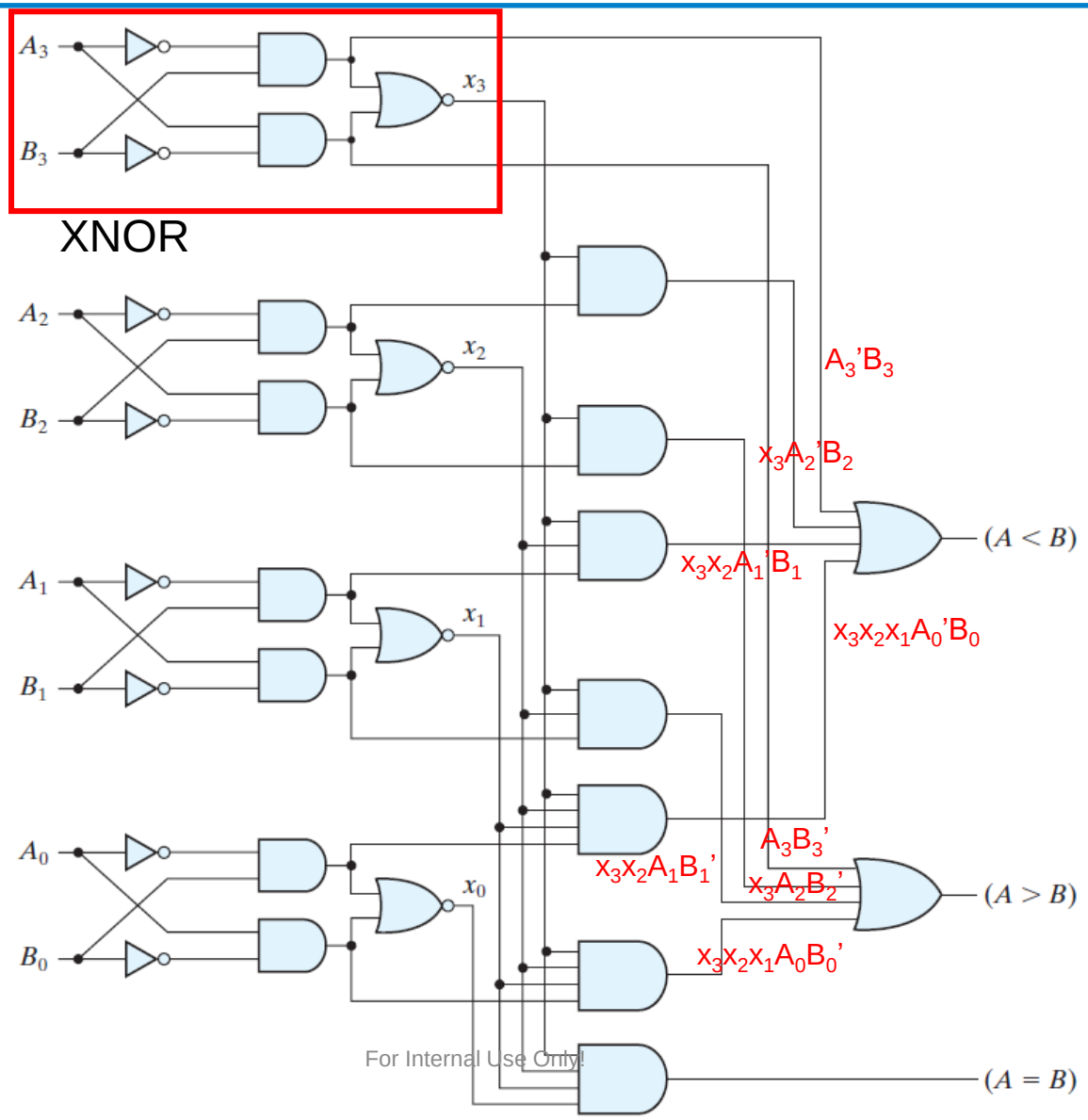


Example3: Magnitude Comparator

- 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 => 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) AND (A_2==B_2) AND (A_1==B_1) AND (A_0==B_0)
    - equality $x_i=A_iB_i+A_i'B_i'$, (A=B) = $x_3x_2x_1x_0$
  - (A>B) = $A_3B_3'+x_3A_2B_2'+x_3x_2A_1B_1' +x_3x_2x_1A_0B_0'$
  - (A<B) = $A_3'B_3+x_3A_2'B_2+x_3x_2A_1'B_1 +x_3x_2x_1A_0'B_0$

Example3: Magnitude Comparator

- Step4

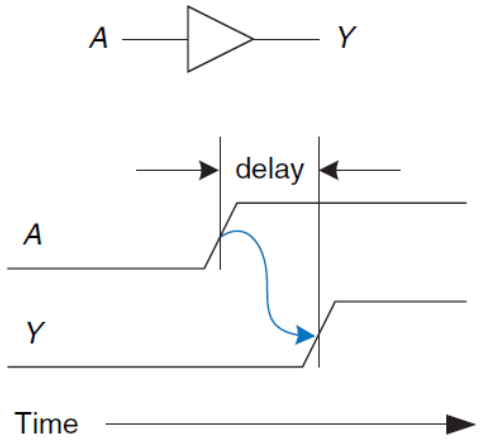


Outline

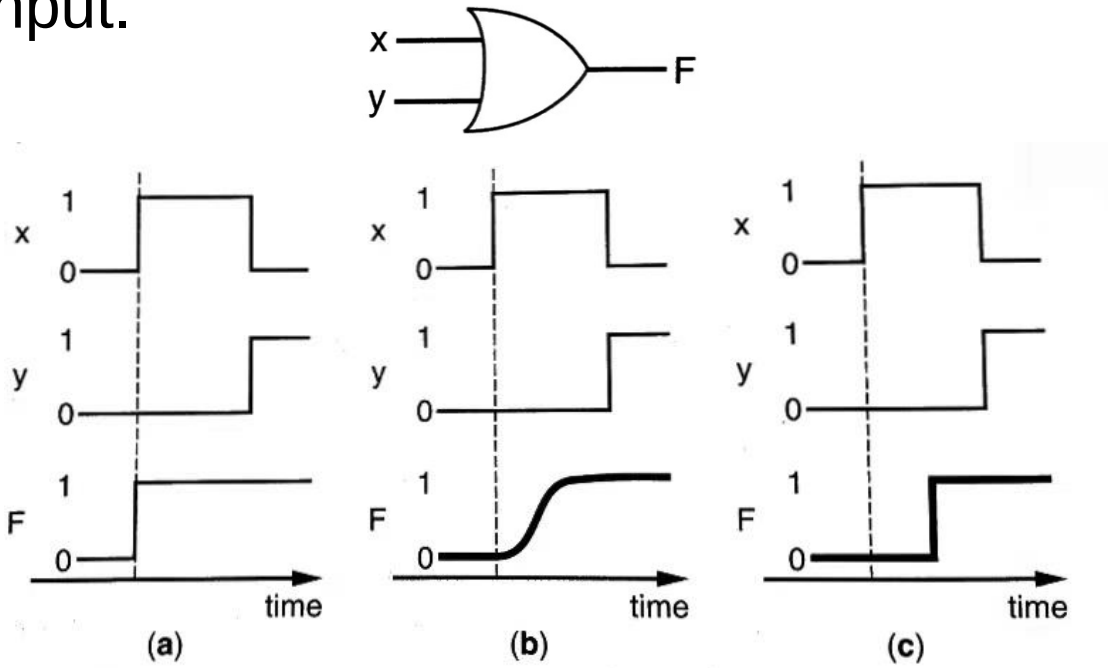
- Analysis of Combinational Circuits
- Design of Combinational Circuits
- **Gate Delays**

Gate Delays

- When the input to a logic gate is changed, the output will not change immediately. The output of the gate experiences a **propagation delay** in response to changes in the input.



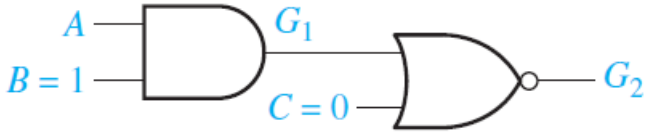
delay between an input change and the subsequent output change for a buffer

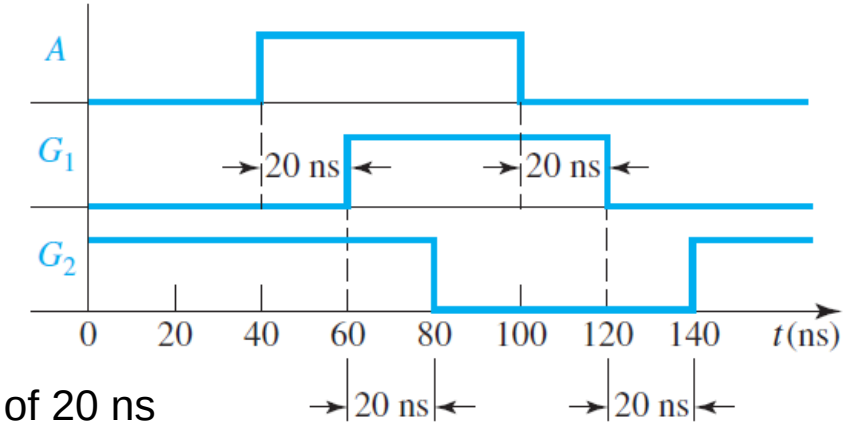


(a) ideal behavior without gate delay
(b) a more realistic illustration
(c) switching incorporating the delay

Effect of gate delays

- The analysis of a combinational circuit ignoring delays can predict only its steady-state behavior.
- Predicts a circuit's output as a function of its inputs assuming that the inputs have been stable for a long time, relative to the delays in the circuit's electronics.
- Because of circuit delays, the transient behavior of a combinational logic circuit may differ from what is predicted by steady-state analysis.





assume each gate has a propagation delay of 20 ns

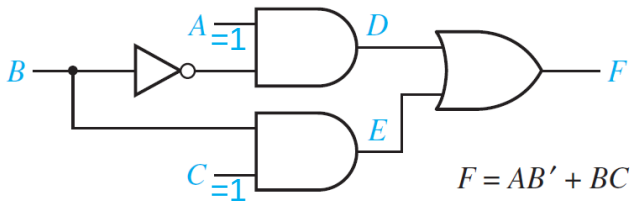
Hazard (Glitches)

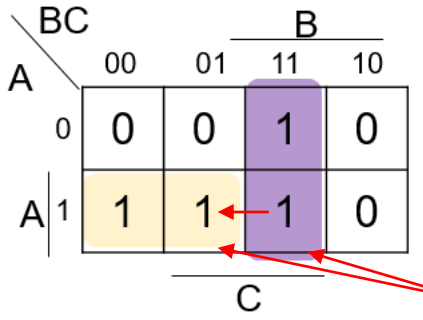
- Timing hazard: **Unwanted** switching transients (glitches) appearing in the output while the input to a combinational circuit changes.
  - static-1 hazard is a short 0 glitch when we expect (by logic theorems) the output to remain constant 1.
  - static-0 hazard is a short 1 glitch when we expect the output to remain constant 0.

(a) Static 1-hazard (b) Static 0-hazard

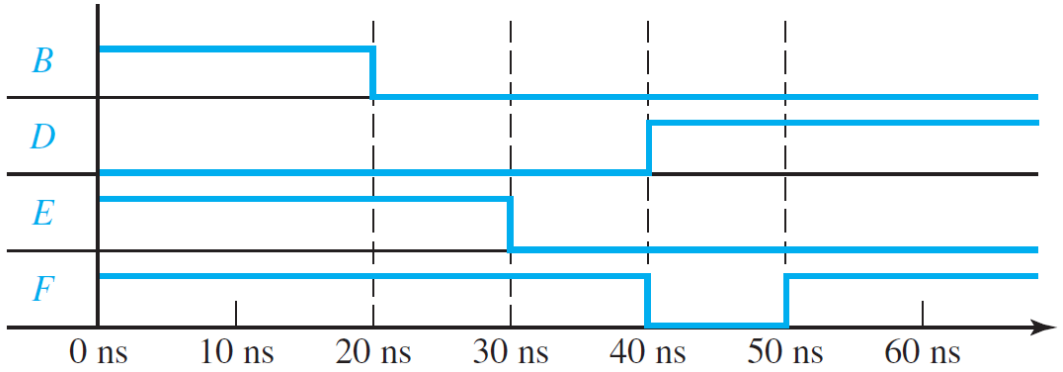
Circuit with a static 1 Hazard

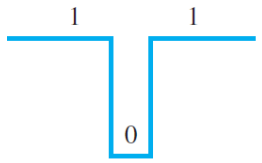
- Assume each gate has a propagation delay of 10 ns
  - if A = C = 1 and B changes from 1 to 0, F should be a stable 1. Change propagates to output F along two paths with different delays, resulting in a glitch in F.





If any two adjacent 1's are not covered by the same circle, a 1-hazard exists for the transition between the two 1's.

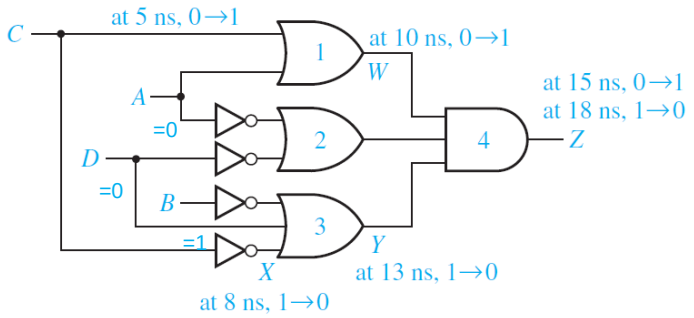




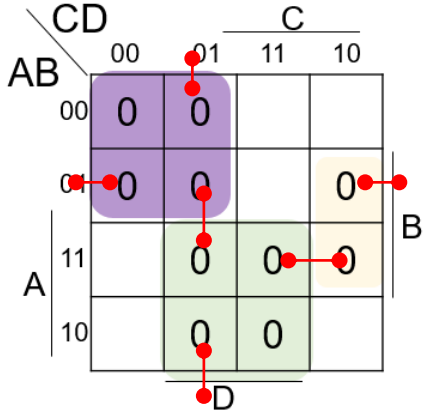
Static 1-hazard

Circuit with a static 0 Hazard

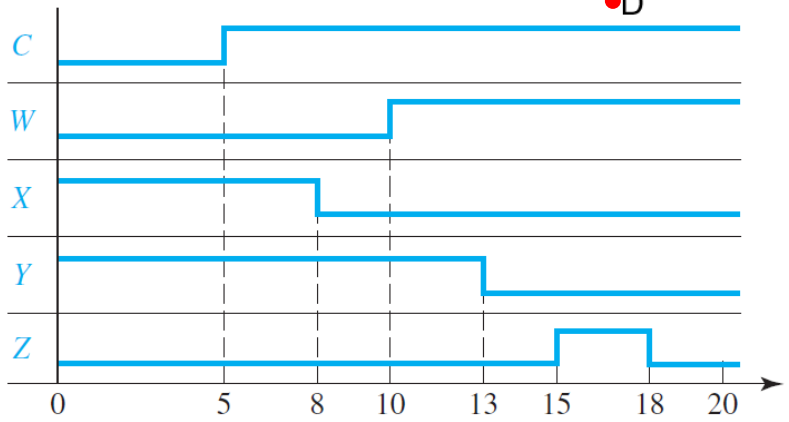
- Assume gate delay: Inverter = 3ns, AND/OR = 5ns
  - if A = 0, B = 1, D = 0, and C changes from 0 to 1, F should be a stable 0. Change propagates to output F along two paths with different delays., resulting in a glitch in F.

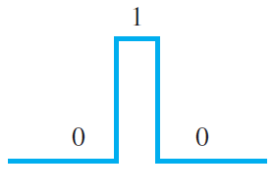


$$F = (A + C)(A' + D')(B' + C' + D)$$



If any two adjacent 0's are not covered by the same circle, a 0-hazard exists for the transition between the two 0's.

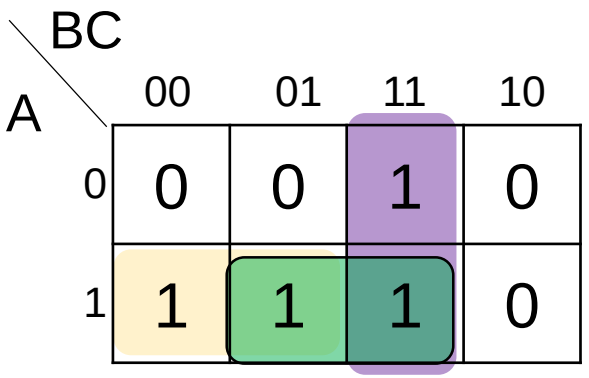




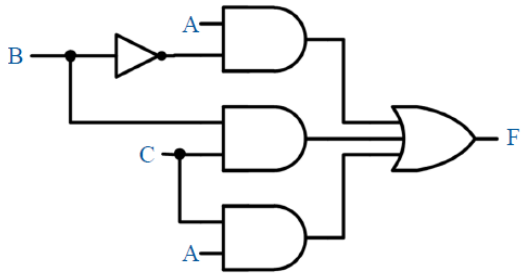
Static 0-hazard

Removing Hazard

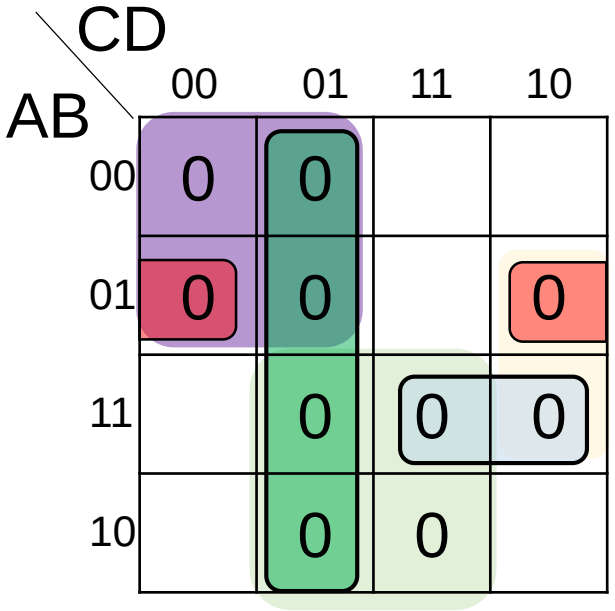
- We can eliminate the 1-hazard or 0-hazards by adding additional circle that covers the adjacent 1's or 0's not already covered within a common circle.



Term AC is added to eliminate 1 hazard



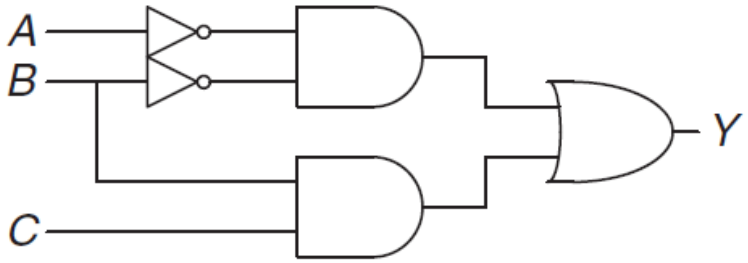
$F = AB' + BC + AC$



Term (C+D'), (A+B'+D), (A'+B'+C') are added to eliminate 0 hazard

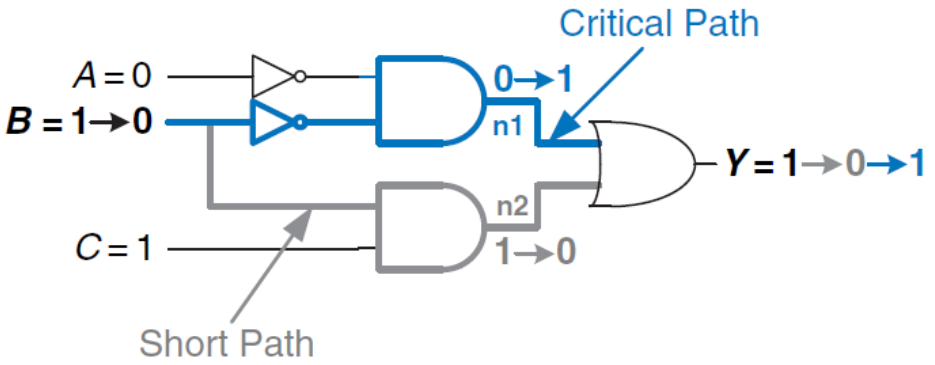
Example

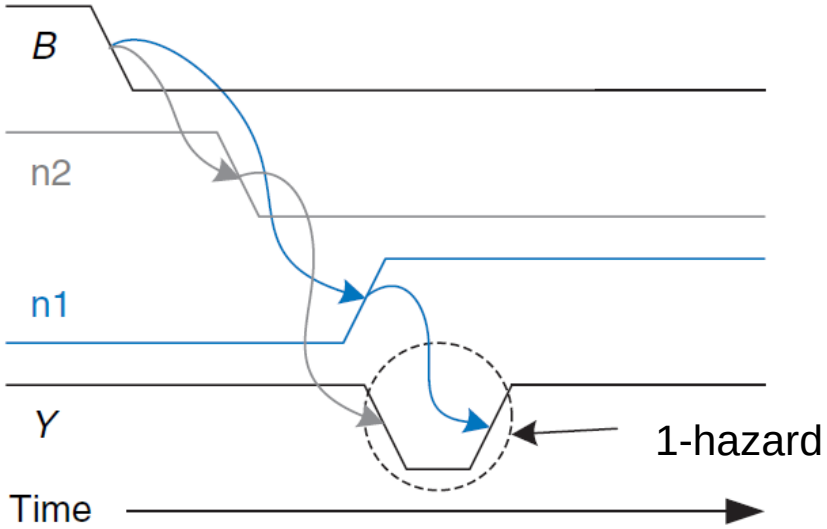
- What happens when A = 0, C = 1, B falls?



Y	AB	00	01	11	10
C	0	1	0	0	0
	1	1	1	1	0

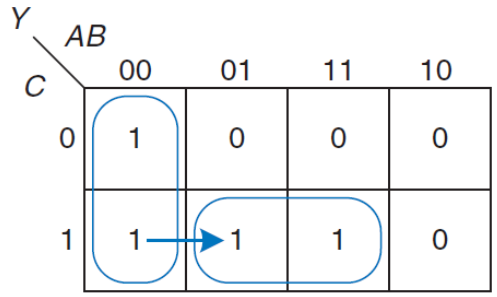
Y = A'B' + BC



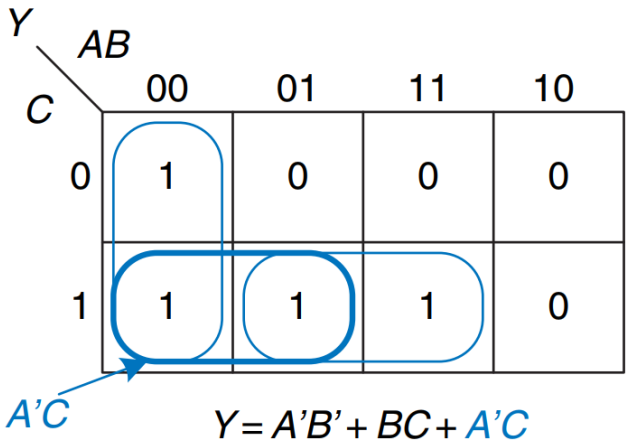


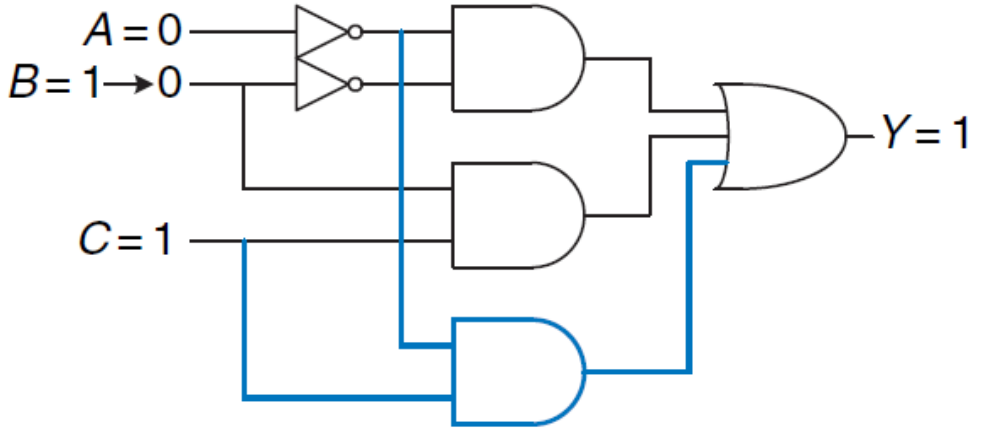
Example

- To eliminate the 1-hazard, add an additional circles that cover the adjacent 1's not already covered by a common circle.



Minimal cost
but with hazard:
Y = A'B' + BC





Eliminate hazard by
adding a product term