DIGITAL LOGIC

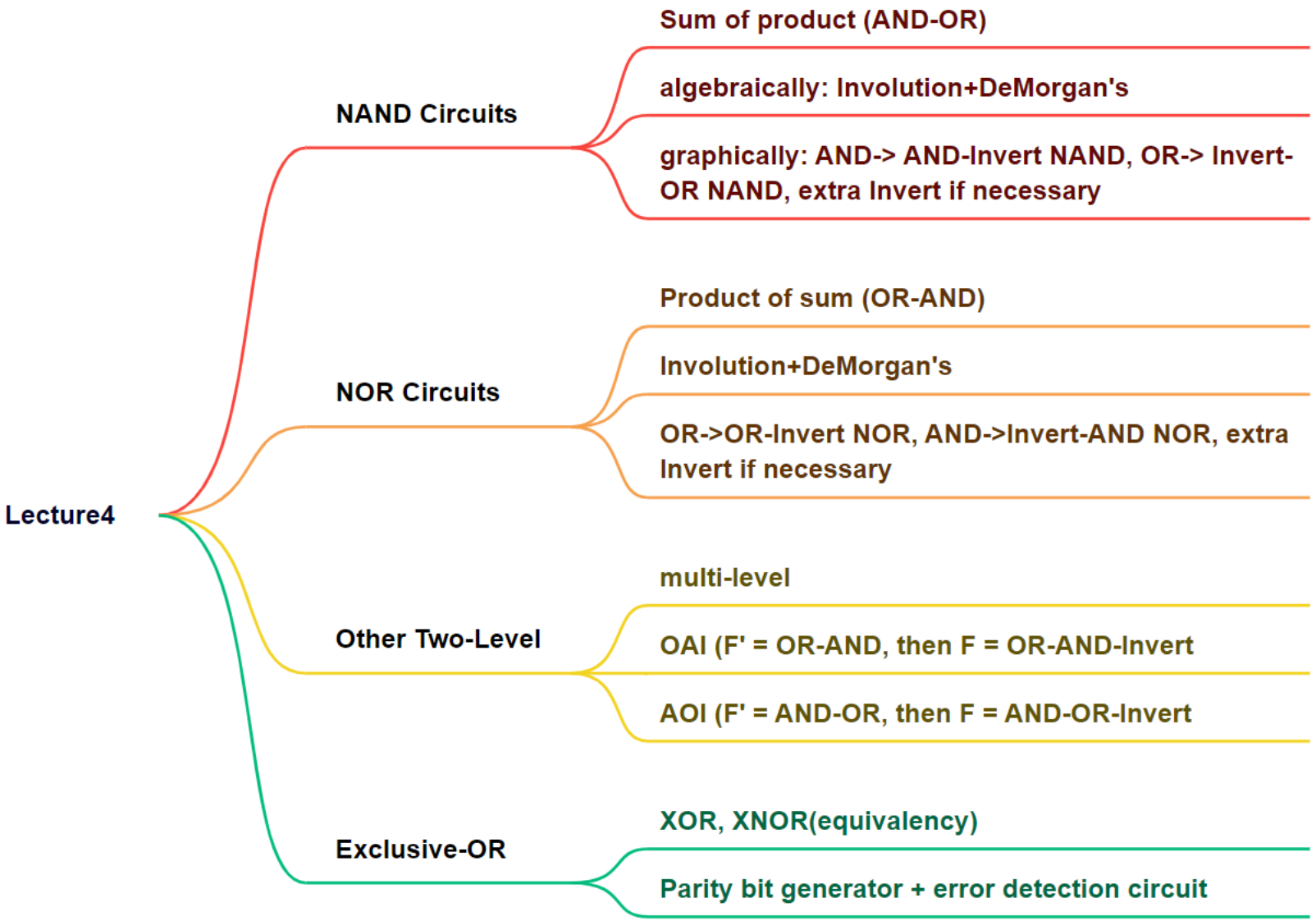
Chapter 4 part1:
Combinational Logic

2023 Fall

Today's Agenda

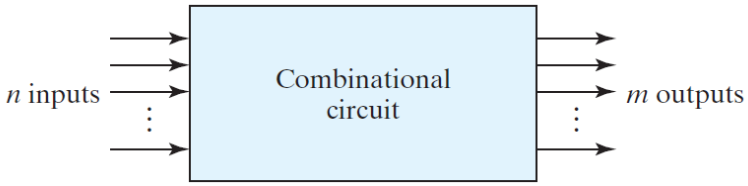
- Recap
- Context
  - Combinational Circuits
    - Analysis of Combinational Circuits
    - Design Procedure
  - Basic Components
    - Magnitude Comparator
- Reading: Textbook, Chapter 4.1-4.4, 4.8

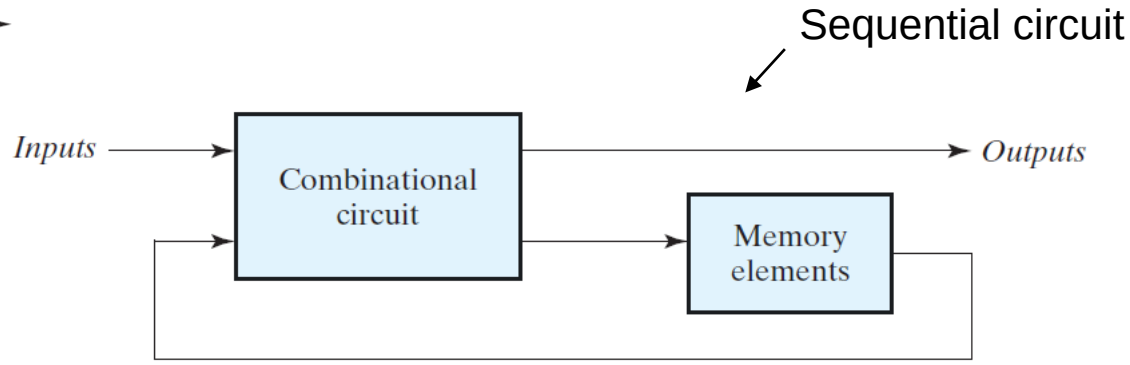
Recap


Lecture4
NAND Circuits
Sum of product (AND-OR)
algebraically: Involution+DeMorgan's
graphically: AND-> AND-Invert NAND, OR-> Invert-OR NAND, extra Invert if necessary
NOR Circuits
Product of sum (OR-AND)
Involution+DeMorgan's
OR->OR-Invert NOR, AND->Invert-AND NOR, extra Invert if necessary
Other Two-Level
multi-level
OAI (F' = OR-AND, then F = OR-AND-Invert
AOI (F' = AND-OR, then F = AND-OR-Invert
Exclusive-OR
XOR, XNOR(equivalency)
Parity bit generator + error detection circuit

Logic Circuits for the Digital System

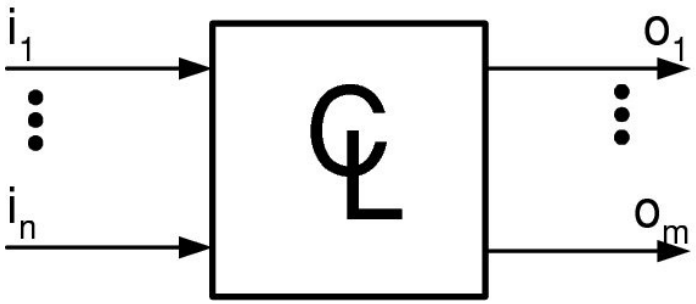
- Combinational circuits
  - Logic circuits whose outputs at any time are determined directly and only from the present input combination.
- Sequential circuits (next lecture)
  - Circuits that employ memory elements + (combinational) logic gates
  - Outputs are determined from the present input combination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



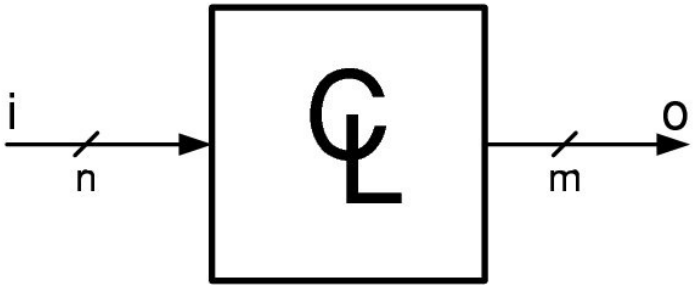


Combinational Logic Circuits

- 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



(a)



(b)

Outline

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

Analysis Procedure

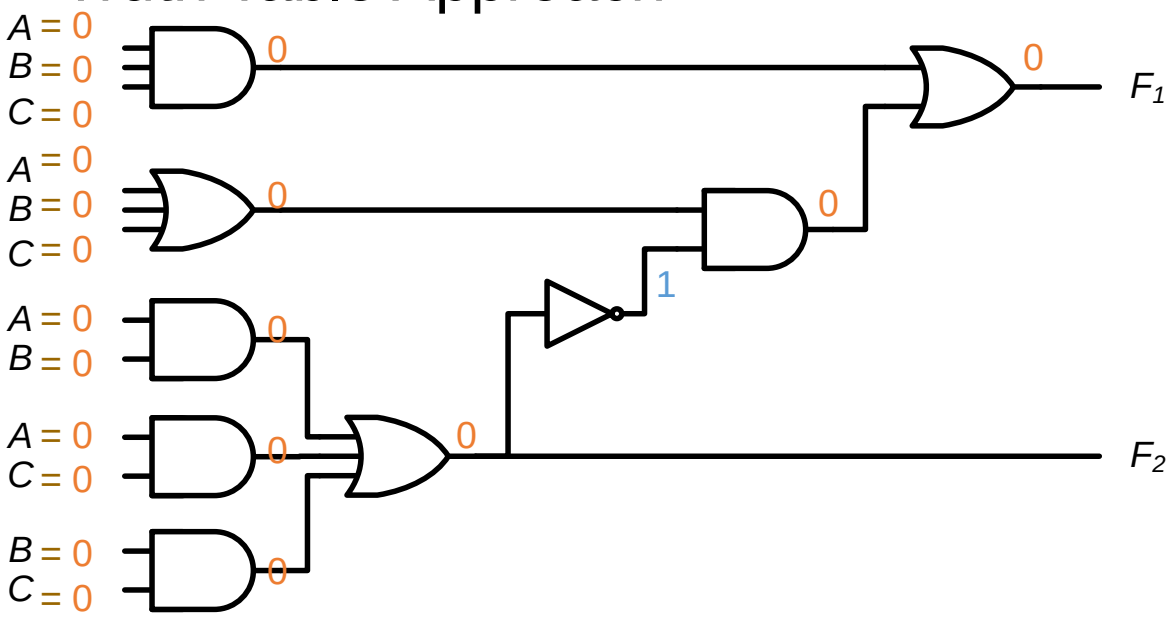
- Analysis of a combinational circuit: determine the function of the circuit.
  - Given logic diagram,
  - develop a set of Boolean functions, a truth table, an optional explanation of the circuit operation.
- If a function name or an explanation is given along the circuit, just verify if the given information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

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.

Example

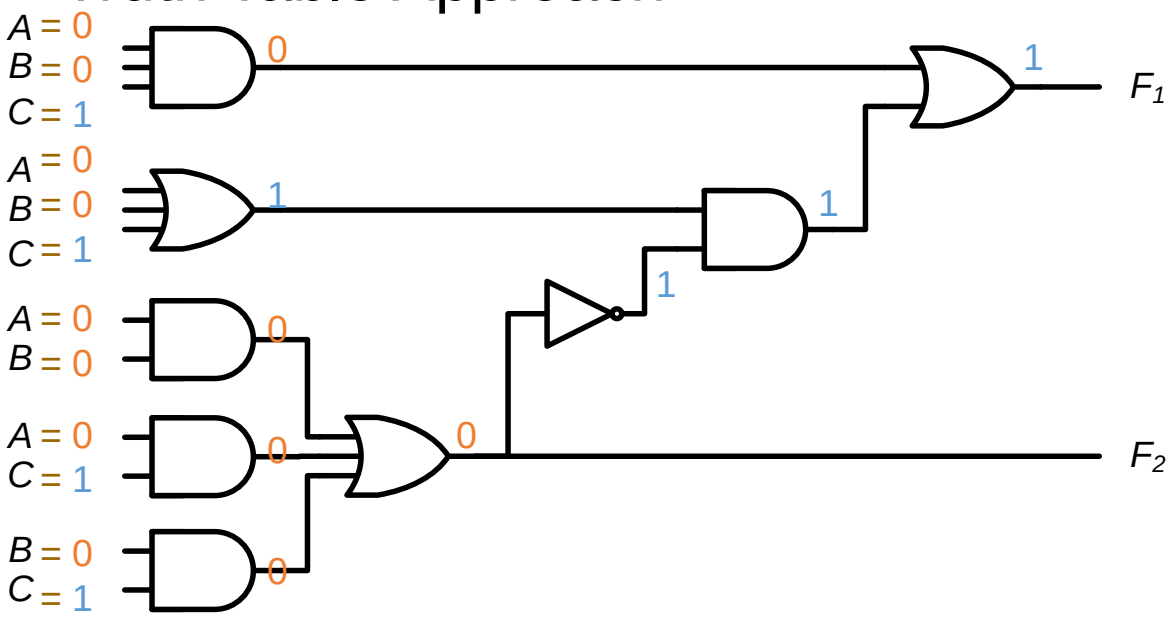
- Truth Table Approach



A B C	F_1	F_2
0 0 0	0	0

Example

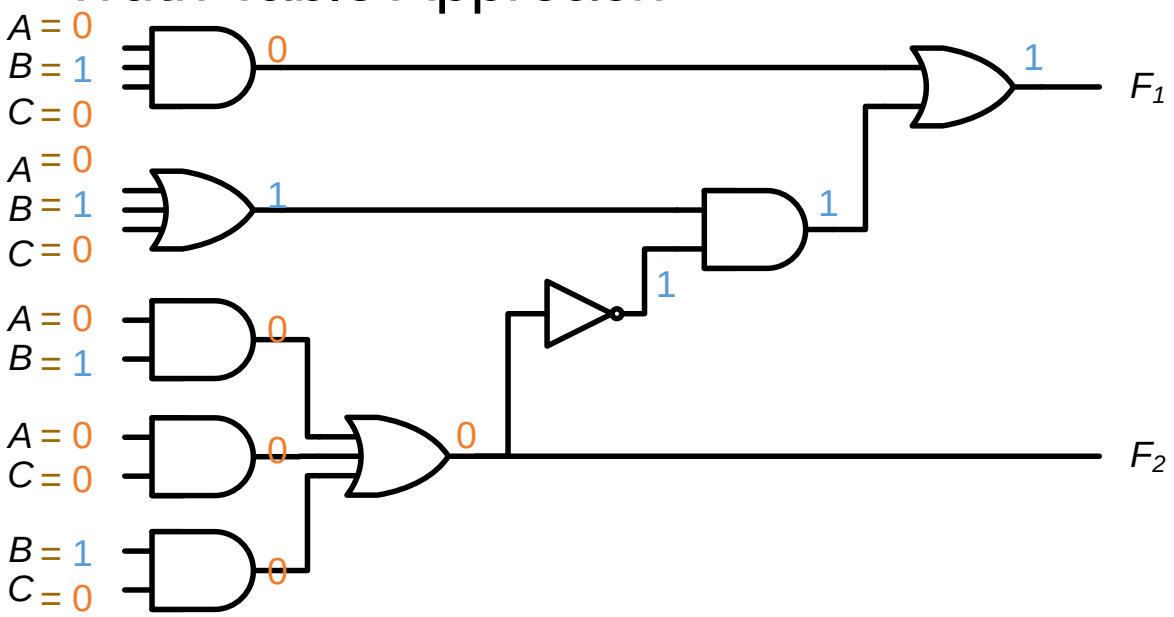
- Truth Table Approach



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

Example

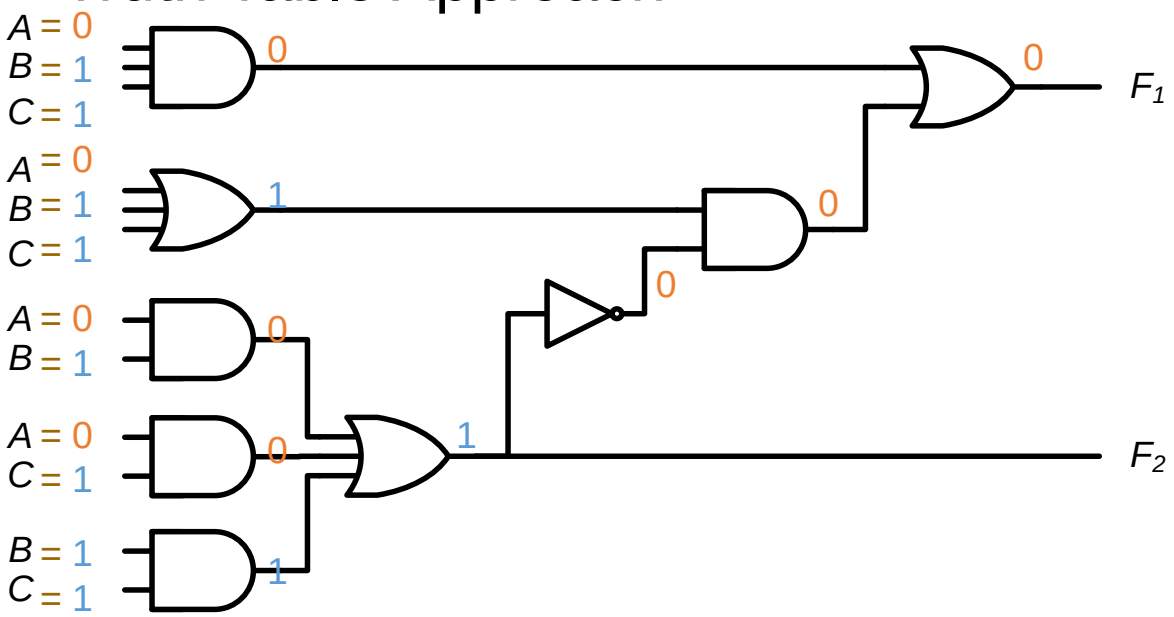
- Truth Table Approach



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

Example

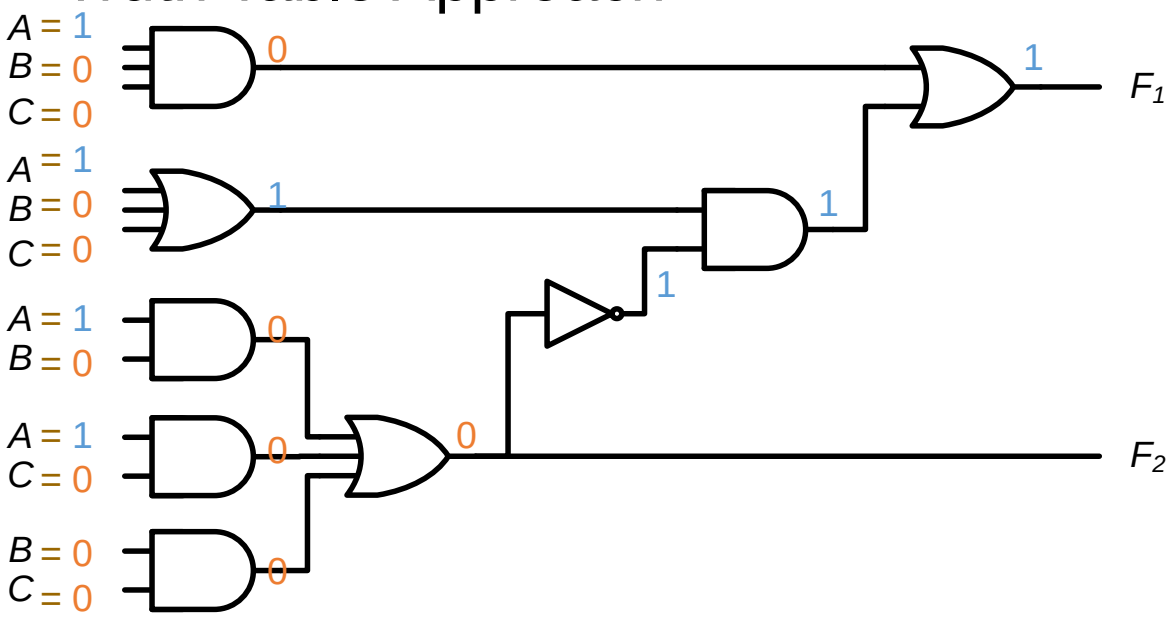
- Truth Table Approach



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

Example

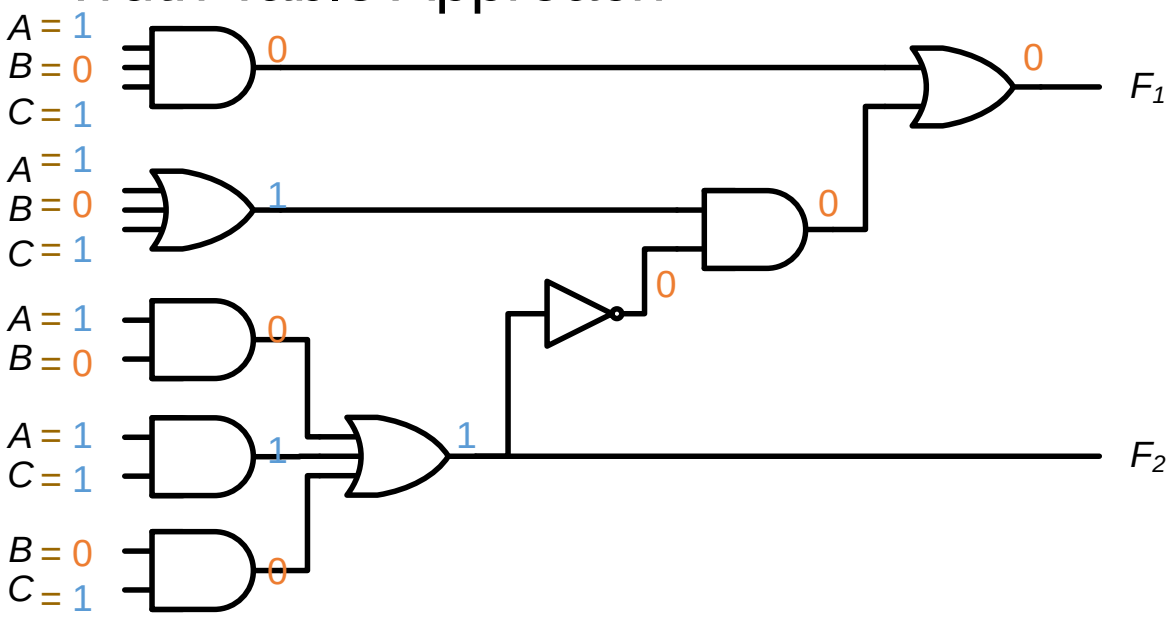
- Truth Table Approach



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

Example

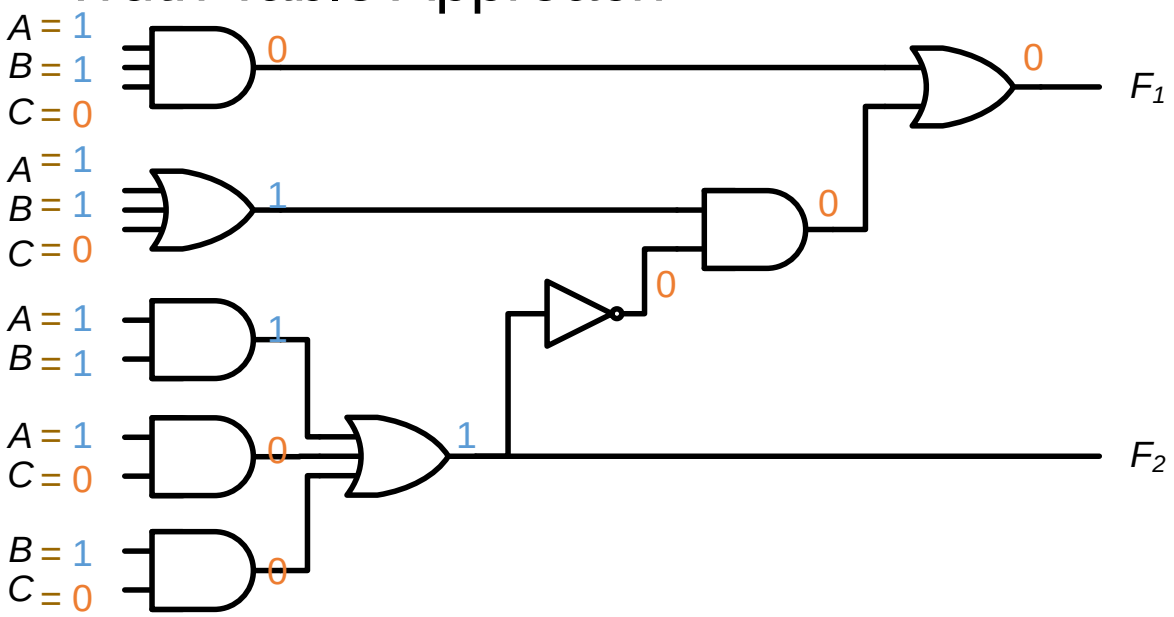
- Truth Table Approach



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

Example

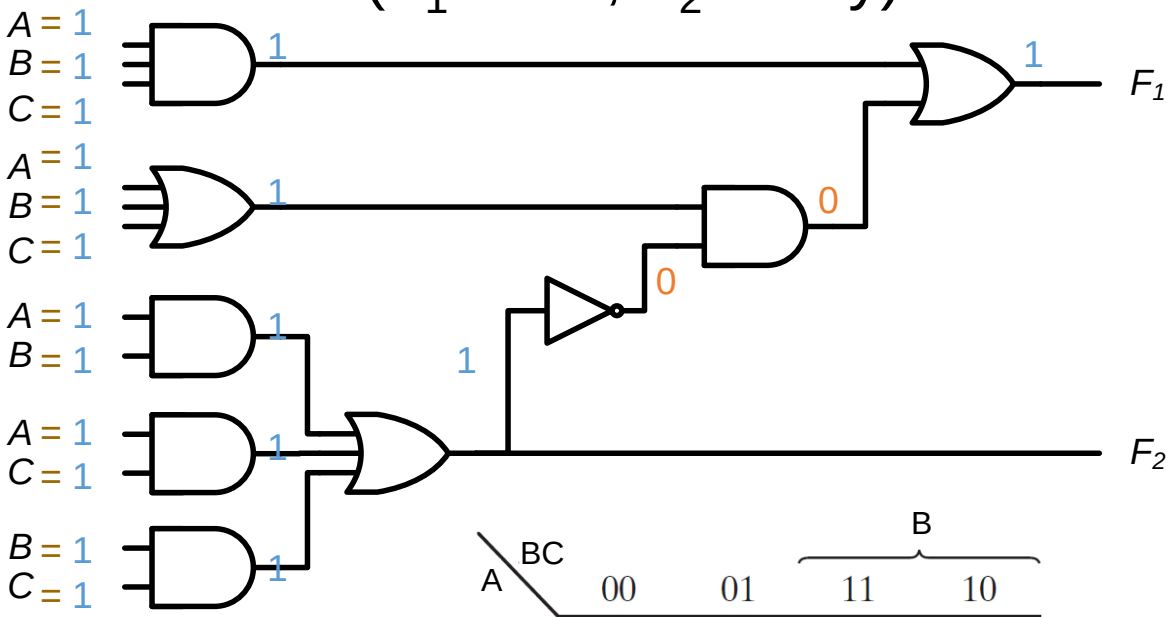
- Truth Table Approach



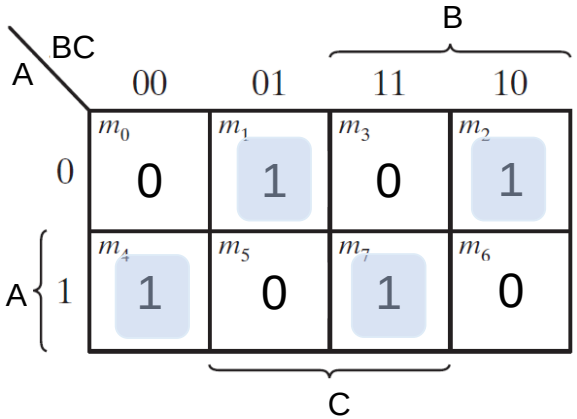
A	B	C	F_1	F_2
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1

Example

- Full adder (F_1: sum, F_2: carry)

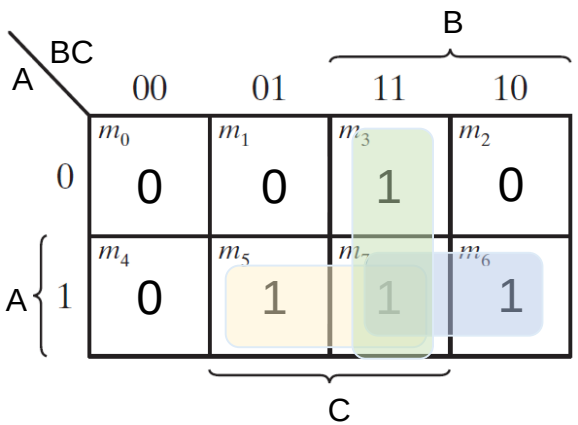


A B C	F_1	F_2
0 0 0	0	0
0 0 1	1	0
0 1 0	1	0
0 1 1	0	1
1 0 0	1	0
1 0 1	0	1
1 1 0	0	1
1 1 1	1	1



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

=A⊕B⊕C



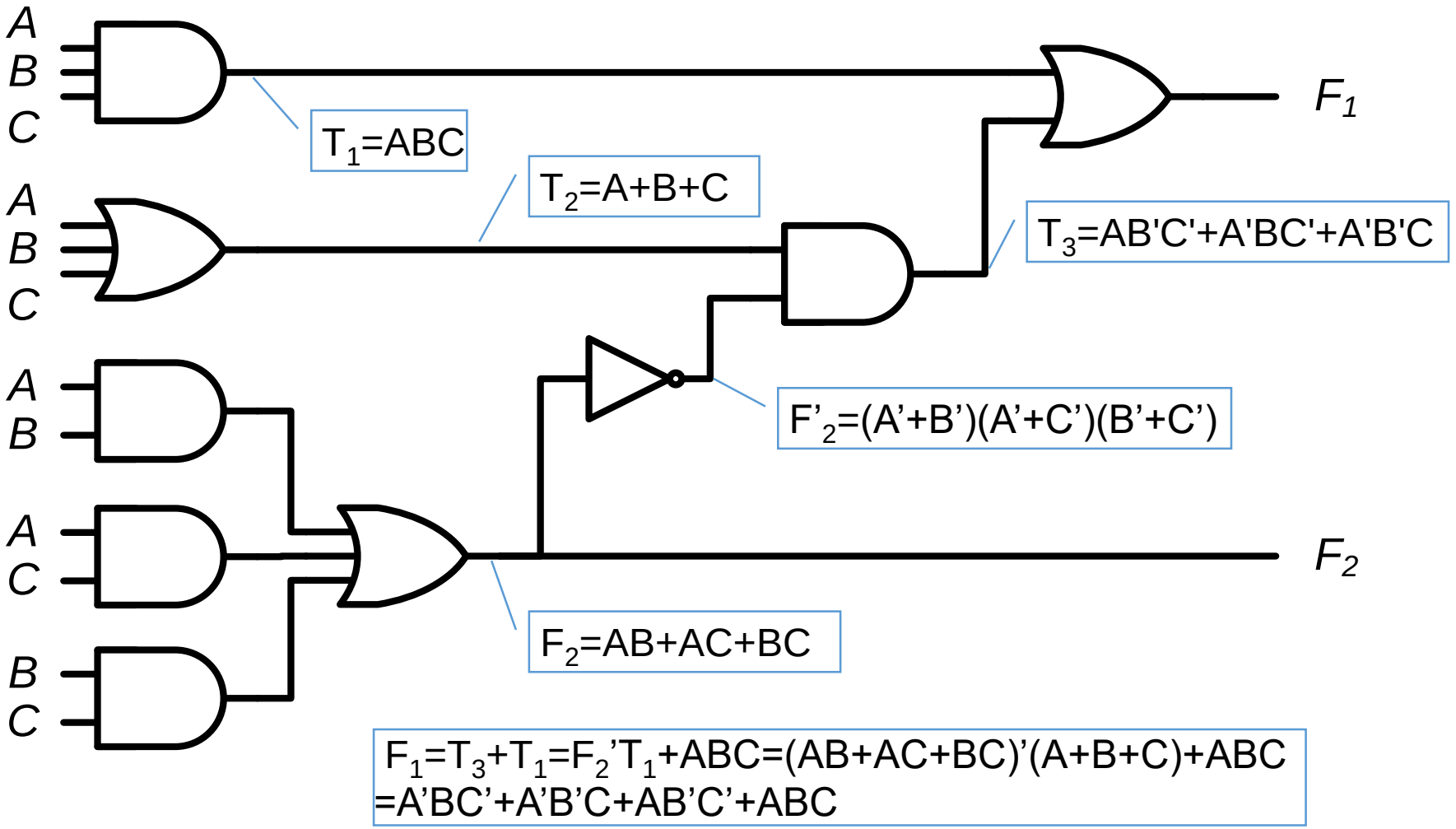
F_2=AB+AC+BC

Derivation of Boolean Functions

- Label all gate outputs that are functions of the input variables only. Determine the functions.
- Label all gate outputs that are functions of the input variables and previously labeled gate outputs, and find the functions.
- Repeat previous step until all the primary outputs are obtained.

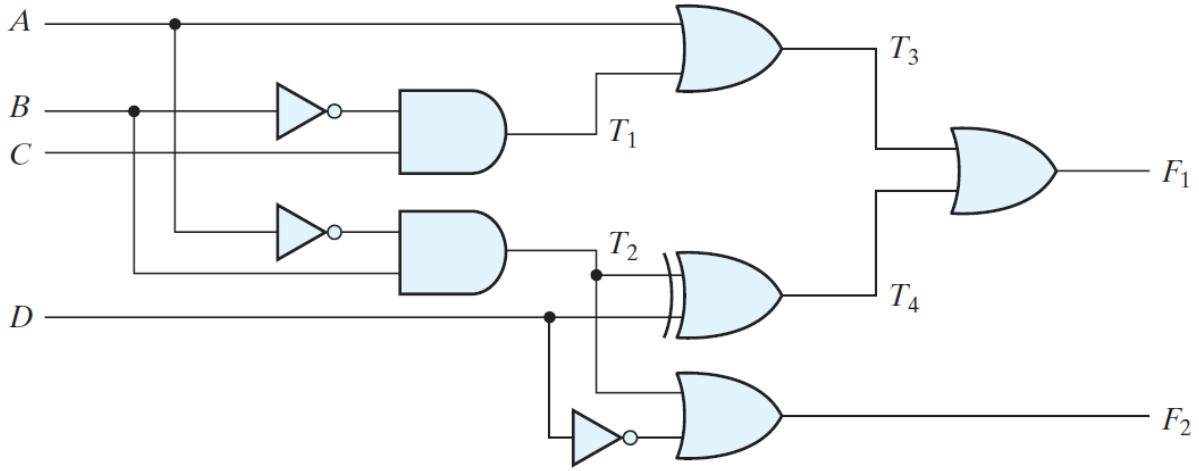
Example 1

- Boolean Expression Approach



Example 2

- 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.



$T_1 = B'C$ $\quad T_2 = A'B$ $\quad T_3 = A + T_1 = A + B'C$

$T_4 = T_2 \oplus D = T_2'D + T_2D' = (A'B)'D + A'BD' = AD + B'D + A'BD'$

$F_1 = T_3 + T_4 = A + B'C + AD + B'D + A'BD' = A(1+D) + A'BD' + B'C + B'D$

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

$F_2 = A'B + D'$

Outline

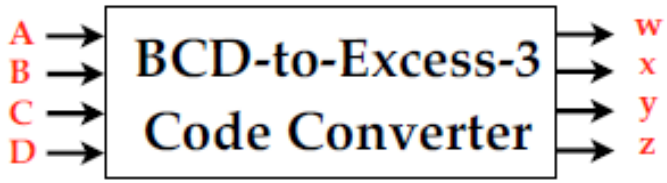
- Analysis of Combinational Circuits
- **Design of Combinational Circuits**

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

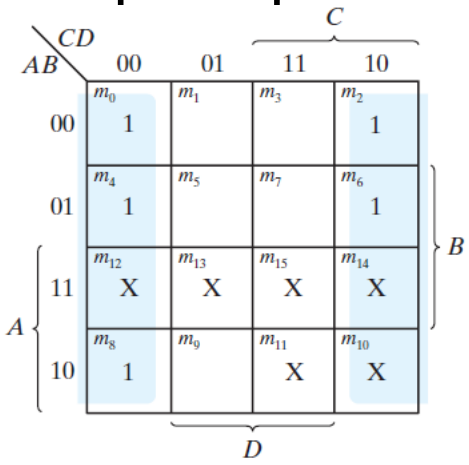


don't care

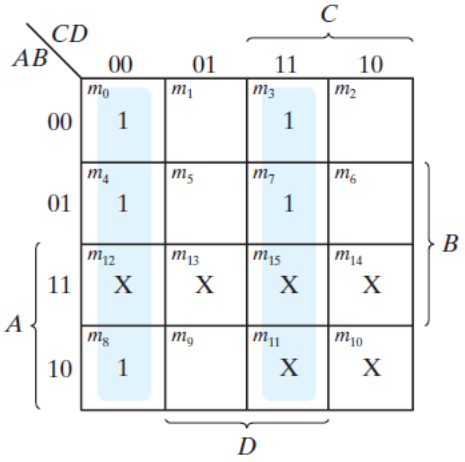
A	B	C	D	w	x	y	z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	x	x	x	x
1	0	1	1	x	x	x	x
1	1	0	0	x	x	x	x
1	1	0	1	x	x	x	X
1	1	1	0	x	x	x	x
1	1	1	1	x	x	x	x

Example1: BCD-to-Excess-3 Code Converter

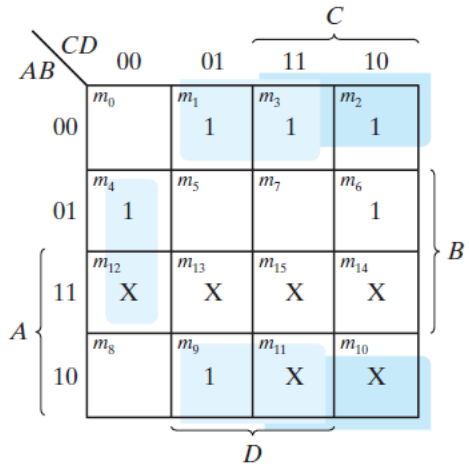
- Step3: Optimization



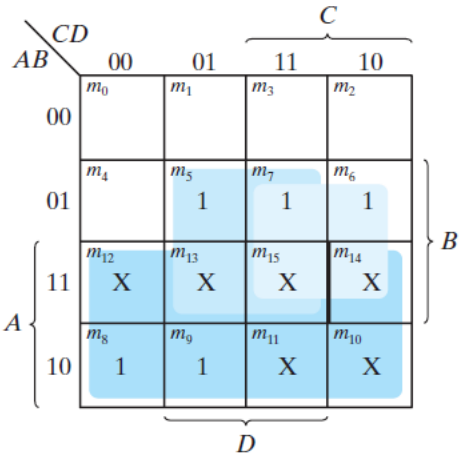
$z = D'$



$y = CD + C'D'$



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



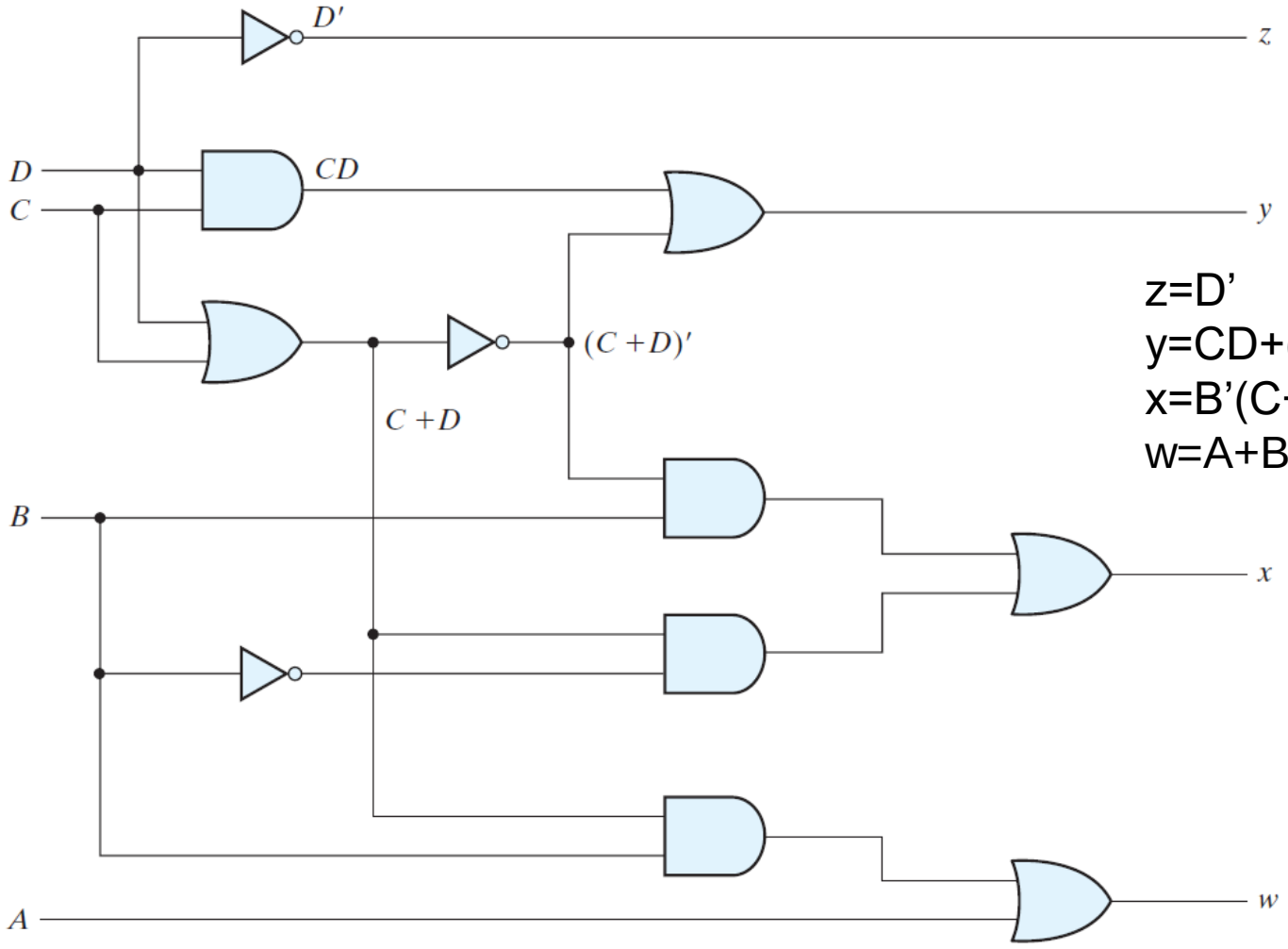
$w = A + BC + BD$

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

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

Example1: BCD-to-Excess-3 Code Converter

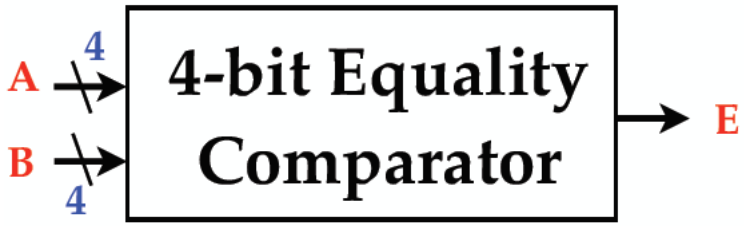
- Step4: Draw logic diagram



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

Example2: 4-bit Equality Comparator

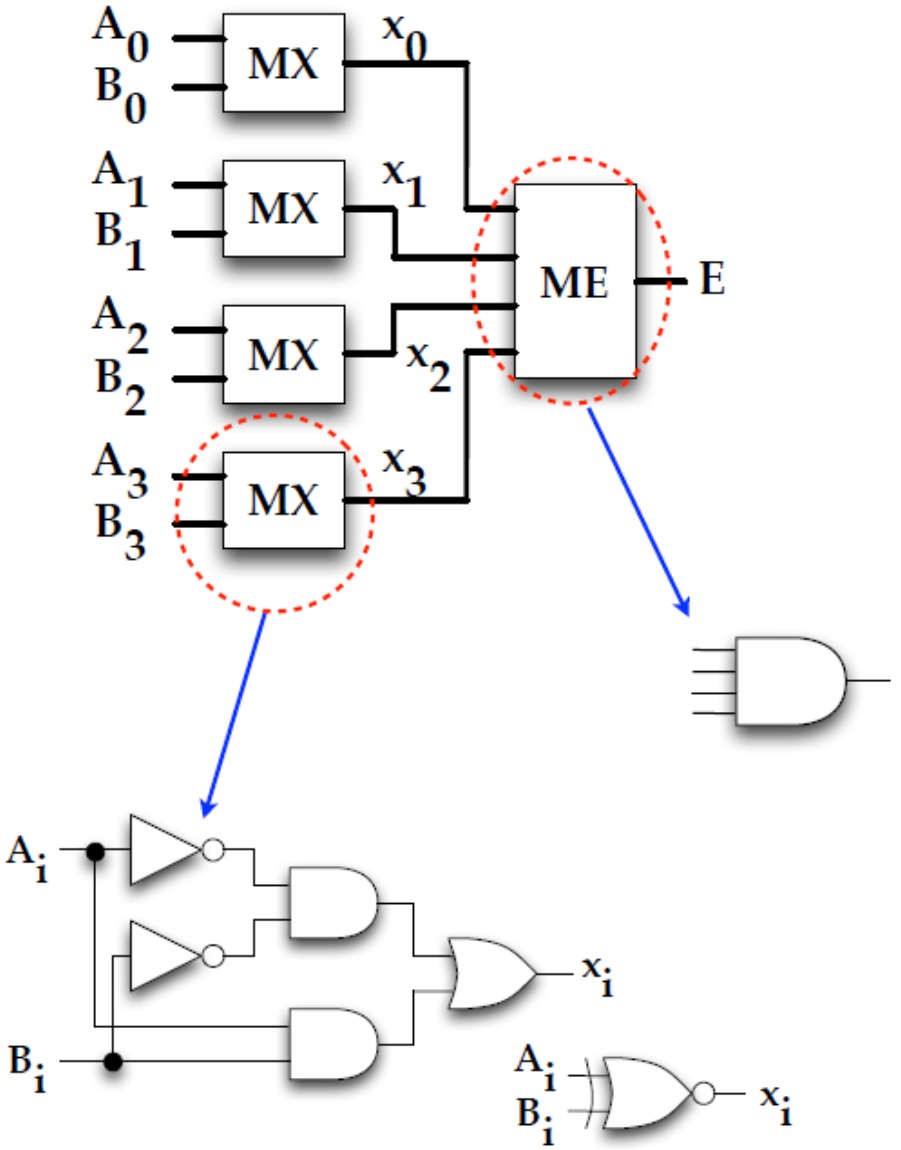
- Step1: Spec
  - input A(3:0), B(3:0);
  - output E (1/0 for equal/unequal)
- Step2: Formulation
  - Bypass the truth table approach due to its size (8 inputs)
  - By algorithm to build a regular circuit
    - $A=A_3A_2A_1A_0$, $B=B_3B_2B_1B_0$
    - A==B, if (A_3==B_3) AND (A_2==B_2) AND (A_1==B_1) AND (A_0==B_0)
    - Suppose bit equality $X_i=(A_i \oplus B_i)' = A_iB_i+A_i'B_i'$, $(A==B) = X_3X_2X_1X_0$



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

Example2: 4-bit Equality Comparator

- Step3: Optimization
  - Regularity
  - Reuse
- Step4: Draw diagram



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
  - $A=A_3A_2A_1A_0$, $B=B_3B_2B_1B_0$
  - 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