Digital Systems Design and Laboratory
[4. Applications of Boolean Algebra]

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

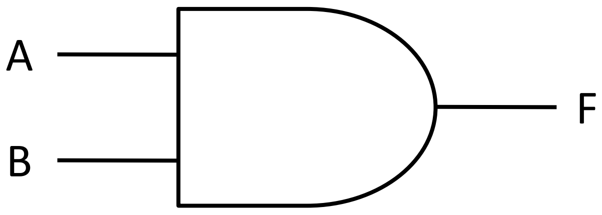
We would like to thank Prof. Hui-Ru Jiang (NTU) for contributing the lecture material

Outline

- **Conversion of English Sentences to Boolean Equations**
- Combinational Logic Design Using a Truth Table
- Minterm and Maxterm Expansions
- General Minterm and Maxterm Expansions
- Incompletely Specified Functions
- Examples of Truth Table Construction
- Design of Binary Adders and Subtracters

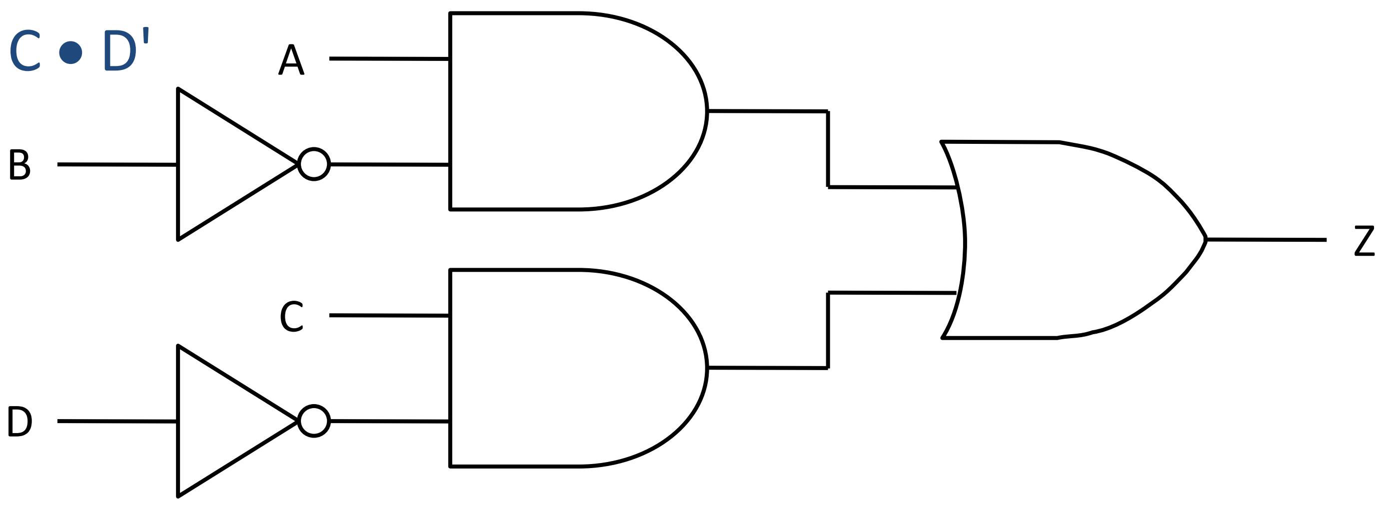
Objectives

- Design a combinational logic circuit starting with a **word description (specification)** of the desired circuit behavior
- Steps
  - Translate the word description into a switching function
    - Boolean expression or truth table
  - Simplify the function
  - Realize it using available logic gates
- Example
  - Mary watches TV **if and only if** it is Monday night **and** she has finished her homework
    - F: Mary watches TV
    - A: It is Monday night
    - B: Mary has finished her homework
  - F = A • B



Another Example

- The alarm will ring **<u>if and only if</u>** the alarm switch is turned on **<u>and</u>** the door is **<u>not</u>** closed, **<u>or</u>** it is after 6pm **<u>and</u>** the window is **<u>not</u>** closed
  - Z: The alarm will ring
  - A: the alarm switch is on
  - B: The door is closed
  - C: It is after 6pm
  - D: The window is closed
- $Z = A \bullet B' + C \bullet D'$



Outline

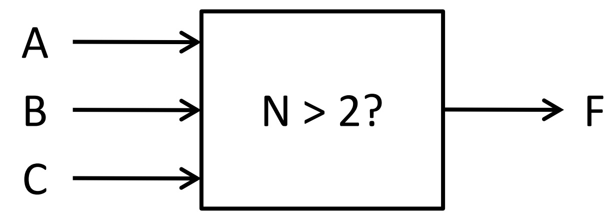
- ❑ Conversion of English Sentences to Boolean Equations
- ❑ **<u>Combinational Logic Design Using a Truth Table</u>**
- ❑ Minterm and Maxterm Expansions
- ❑ General Minterm and Maxterm Expansions
- ❑ Incompletely Specified Functions
- ❑ Examples of Truth Table Construction
- ❑ Design of Binary Adders and Subtracters

Threshold Detector (1/2)

- Design a detector that outputs 1 when input is greater than 2
  - Inputs $(A, B, C)_2$ represent a binary number N
  - If $N = (A, B, C)_2 \geq 3$, output F = 1; otherwise F = 0

A	B	C	F	F'
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

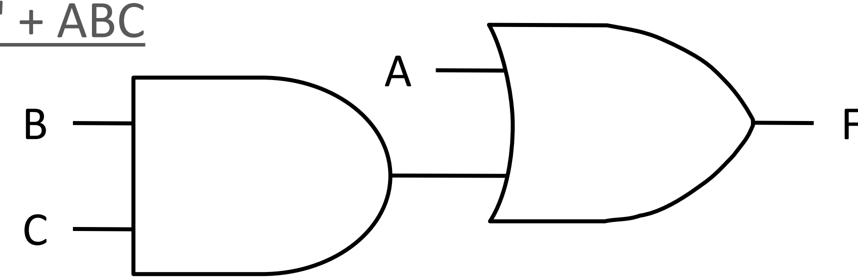
Show the condition to make output = 1



  - F = A'BC + AB'C' + AB'C + ABC' + ABC (SOP)

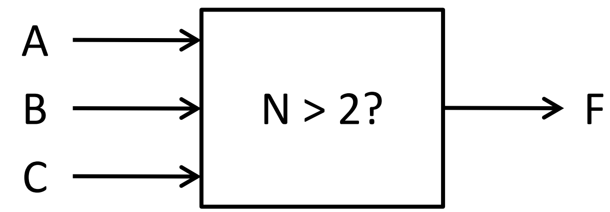
    = $\underline{A'BC + ABC}$ + $\underline{AB'C' + AB'C + ABC' + ABC}$

    = A + BC



Threshold Detector (2/2)

- By counting 1's, we have SOP
- What if counting 0's



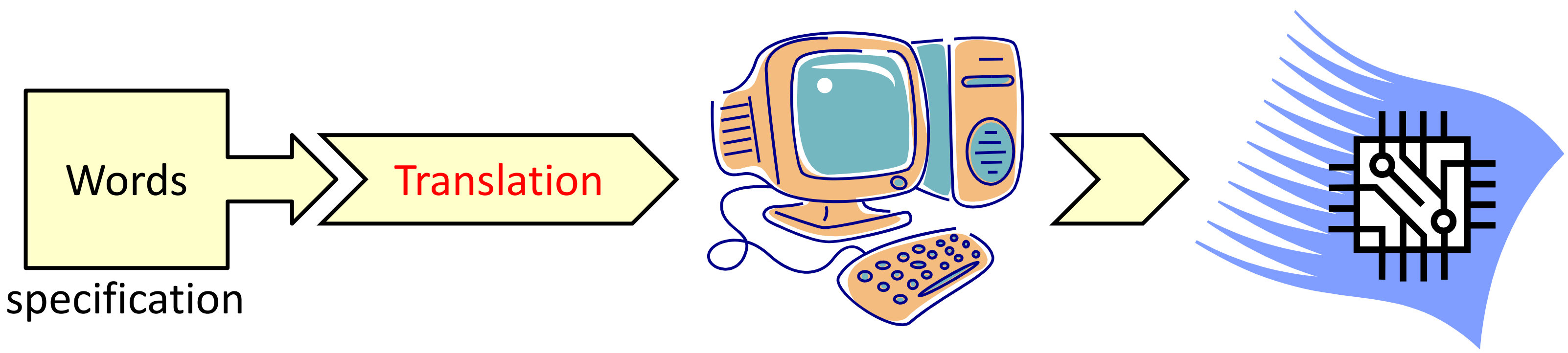
A	B	C	F	F'
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

Show the condition to make output = 0

- $F = (ABC \neq 000) \bullet (ABC \neq 001) \bullet (ABC \neq 010)$
  $= (A'B'C')' \bullet (A'B'C)' \bullet (A'BC')' = (A + B + C) \bullet (A + B + C') \bullet (A + B' + C)$
- $F = F'' = (A'B'C' + A'B'C + A'BC')'$
  $= (A'B'C')' \bullet (A'B'C)' \bullet (A'BC')' = (A + B + C) \bullet (A + B + C') \bullet (A + B' + C)$

Logic Design Using a Truth Table

- Steps
  - Make a truth table according to the word description
  - Generate a Boolean expression
    - Sum-of-products (SOP): check 1's
    - Product-of-sums (POS): check 0's
      - Have F' in SOP and then derive F in POS
  - Simplify the Boolean expression



Outline

- Conversion of English Sentences to Boolean Equations
- Combinational Logic Design Using a Truth Table
- **Minterm and Maxterm Expansions**
- General Minterm and Maxterm Expansions
- Incompletely Specified Functions
- Examples of Truth Table Construction
- Design of Binary Adders and Subtracters

Minterm and Maxterm

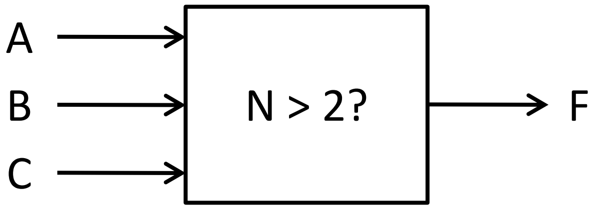
- Definition: A **minterm/maxterm** of n variables is a product/sum of n literals in which each variable appears exactly once in either true or complement form (but not both)
  - A literal is a variable or its complement (A or A')
  - Examples of 3 variables
    - Minterm: A'BC, AB'C'
    - Maxterm: A + B + C, A + B + C'

$(m_i)' = M_i$

Row No.	ABC	Minterm m_i	Maxterm M_i
0	000	m_0 = A'B'C'	M_0 = A + B + C
1	001	m_1 = A'B'C	M_1 = A + B + C'
2	010	m_2 = A'BC'	M_2 = A + B' + C
3	011	m_3 = A'BC	M_3 = A + B' + C'
4	100	m_4 = AB'C'	M_4 = A' + B + C
5	101	m_5 = AB'C	M_5 = A' + B + C'
6	110	m_6 = ABC'	M_6 = A' + B' + C
7	111	m_7 = ABC	M_7 = A' + B' + C'

Minterm Expansion

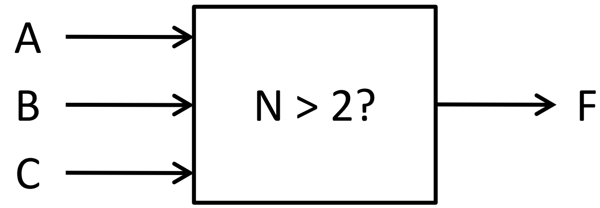
- A **minterm expansion** or a **standard sum of products** is a function written as a sum of minterms
  - Counting 1's
- Example
  - F = A'BC + AB'C' + AB'C + ABC' + ABC
    $= m_3 + m_4 + m_5 + m_6 + m_7$ (m-notation)
    $= \sum m(3, 4, 5, 6, 7)$



Row No.	ABC	Minterm m_i	Maxterm M_i
0	000	m_0 = A'B'C'	M_0 = A + B + C
1	001	m_1 = A'B'C	M_1 = A + B + C'
2	010	m_2 = A'BC'	M_2 = A + B' + C
3	011	m_3 = A'BC	M_3 = A + B' + C'
4	100	m_4 = AB'C'	M_4 = A' + B + C
5	101	m_5 = AB'C	M_5 = A' + B + C'
6	110	m_6 = ABC'	M_6 = A' + B' + C
7	111	m_7 = ABC	M_7 = A' + B' + C'

Maxterm Expansion

- A **maxterm expansion** or a **standard product of sums** is a function written as a product of maxterms
  - Counting 0's
- Example
  - $F = (A + B + C) \bullet (A + B + C') \bullet (A + B' + C)$
    $= M_0M_1M_2$ (M-notation)
    $= \prod M(0, 1, 2)$



Row No.	ABC	Minterm m_i	Maxterm M_i
0	000	$m_0 = A'B'C'$	$M_0 = A + B + C$
1	001	$m_1 = A'B'C$	$M_1 = A + B + C'$
2	010	$m_2 = A'BC'$	$M_2 = A + B' + C$
3	011	$m_3 = A'BC$	$M_3 = A + B' + C'$
4	100	$m_4 = AB'C'$	$M_4 = A' + B + C$
5	101	$m_5 = AB'C$	$M_5 = A' + B + C'$
6	110	$m_6 = ABC'$	$M_6 = A' + B' + C$
7	111	$m_7 = ABC$	$M_7 = A' + B' + C'$

Complement by Minterms/Maxterms

- $(m_i)' = M_i$
- Complement of F
  - Counting 0's in F (find F' directly)
    - $F' = m_0 + m_1 + m_2 = \sum m(0, 1, 2)$
    - $F' = M_3M_4M_5M_6M_7 = \prod M(3, 4, 5, 6, 7)$
  - Counting 1's in F (find F and then complement it)
    - $F' = (m_3 + m_4 + m_5 + m_6 + m_7)'$
      $= m_3'm_4'm_5'm_6'm_7'$
      $= M_3M_4M_5M_6M_7$
      $= \prod M(3, 4, 5, 6, 7)$
    - $F' = (M_0M_1M_2)'$
      $= M_0' + M_1' + M_2'$
      $= m_0 + m_1 + m_2$
      $= \sum m(0, 1, 2)$

A	B	C	F	F'
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

Another Example

- Example: $F(A, B, C, D) = A'(B' + D) + ACD'$

$$
\begin{aligned}
& F(A, B, C, D) \\
&= A'(B' + D) + ACD' \\
&= A'B' + A'D + ACD' \\
&= A'B'(C + C')(D + D') + A'D(B + B')(C + C') + ACD'(B + B') \\
&= A'B'C'D' + A'B'C'D + A'B'CD' + A'B'CD + A'BC'D + A'BCD + ABCD' + AB'CD' \\
&\quad\ 0000 \qquad 0001 \qquad 0010 \qquad 0011 \qquad 0101 \qquad 0111 \qquad 1110 \qquad 1010 \\
&= \sum m(0, 1, 2, 3, 5, 7, 10, 14) \qquad \text{... Minterm Expansion}
\end{aligned}
$$

$$
\begin{aligned}
& F(A, B, C, D) \\
&= (A' + CD')(A + B' + D) = (A' + C)(A' + D')(A + B' + D) \\
&= (A' + BB' + C + DD')(A' + BB' + CC' + D')(A + B' + CC' + D) \\
&= \ldots \\
&= \prod M(4, 6, 8, 9, 11, 12, 13, 15) \qquad \text{... Maxterm Expansion}
\end{aligned}
$$

Summary

- Convert a Boolean expression to a minterm/maxterm expansion
  - Use truth table
    - Sometimes there are too many terms
- Use Boolean algebra
  - SOP: multiply out and use (X + X') = 1 → minterm expansion
  - POS: factor and use XX' = 0 → maxterm expansion

Outline

- Conversion of English Sentences to Boolean Equations
- Combinational Logic Design Using a Truth Table
- Minterm and Maxterm Expansions
- **General Minterm and Maxterm Expansions**
- Incompletely Specified Functions
- Examples of Truth Table Construction
- Design of Binary Adders and Subtracters

General Truth Table

- Given n Boolean variables, how many different Boolean functions can you produce?
  - Each a_i can be assigned with either 0 or 1
  - $2^{(2^n)}$

A	B	C	F
0	0	0	a_0
0	0	1	a_1
0	1	0	a_2
0	1	1	a_3
1	0	0	a_4
1	0	1	a_5
1	1	0	a_6
1	1	1	a_7

AND of Minterm Expansions

- Given $F_1 = \sum m(0, 2, 3, 5, 9, 11)$ and $F_2 = \sum m(0, 3, 9, 11, 13, 14)$, find F_1F_2 = ?
  - AND: take the numbers that appear in both expansions
  - $F_1F_2 = \sum m(0, 3, 9, 11)$
- AND for two maxterm expansions?
- OR for two minterm expansions?
- OR for two maxterm expansions?

Conversion of Forms

- Convert between a minterm and a maxterm expansion

Given Form \ Desired Form	Minterm Expansion of F	Maxterm Expansion of F	Minterm Expansion of F'	Maxterm Expansion of F'
Minterm Expansion of F	---	The nos. not on the minterm list for F	The minterms not present in F	The same as minterm nos. of F
Maxterm Expansion of F	The nos. not on the maxterm list for F	---	The same as maxterm nos. of F	The maxterms not present in F

Given Form \ Desired Form	Minterm Expansion of F	Maxterm Expansion of F	Minterm Expansion of F'	Maxterm Expansion of F'
$F = \sum m(3, 4, 5, 6, 7)$	---	$\prod M(0, 1, 2)$	$\sum m(0, 1, 2)$	$\prod M(3, 4, 5, 6, 7)$
$F = \prod M(0, 1, 2)$	$\sum m(3, 4, 5, 6, 7)$	---	$\sum m(0, 1, 2)$	$\prod M(3, 4, 5, 6, 7)$

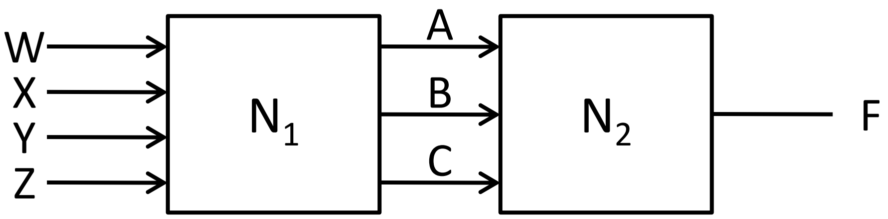
Outline

- ❑ Conversion of English Sentences to Boolean Equations
- ❑ Combinational Logic Design Using a Truth Table
- ❑ Minterm and Maxterm Expansions
- ❑ General Minterm and Maxterm Expansions
- ❑ **Incompletely Specified Functions**
- ❑ Examples of Truth Table Construction
- ❑ Design of Binary Adders and Subtracters

Incompletely Specified Functions (1/2)

- A large digital system is usually divided into subcircuits
- Assume N_1 never generates ABC = 001/110 for any W, X, Y, Z

A	B	C	F
0	0	0	1
0	0	1	X
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	X
1	1	1	1



- F: Incompletely specified function
- A'B'C, ABC': don't care terms
  - "don't care" (DC) terms can be assigned with either 0 or 1

Incompletely Specified Functions (2/2)

Impact of don't care terms on Boolean simplification

- Try exhaustive combinations of DCs to find the best
  - (may be stupid but works for now)
- Assign 0 to both "X"
  - F = A'B'C' + A'BC + ABC = A'B'C' + BC
- Assign 1 to 1st "X" and 0 to 2nd "X" (seems to be the simplest)
  - F = A'B'C' + <u>A'B'C</u> + A'BC + ABC = A'B' + BC
- Assign 1 to both "X"
  - F = A'B'C' + <u>A'B'C</u> + A'BC + <u>ABC'</u> + ABC = A'B' + BC + AB
- ...
- 2. is the simplest solution

A	B	C	F
0	0	0	1
0	0	1	X
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	X
1	1	1	1

Notation

- $F = \sum m(0, 3, 7) + \sum d(1, 6)$
- $F = \prod M(2, 4, 5) \bullet \prod D(1, 6)$

Outline

- Conversion of English Sentences to Boolean Equations
- Combinational Logic Design Using a Truth Table
- Minterm and Maxterm Expansions
- General Minterm and Maxterm Expansions
- Incompletely Specified Functions
- **Examples of Truth Table Construction**
- Design of Binary Adders and Subtracters

Error Detector for 6-3-1-1 Codes (1/2)

- Design an error detector for 6-3-1-1 codes:
  - The output F = 1 iff inputs (A, B, C, D) represent an **invalid** code combination
  - Step 1: construct the truth table

Decimal Digit	6-3-1-1 Code
0	0000
1	0001
2	0011
3	0100
4	0101
5	0111
6	1000
7	1001
8	1011
9	1100

$\Rightarrow$

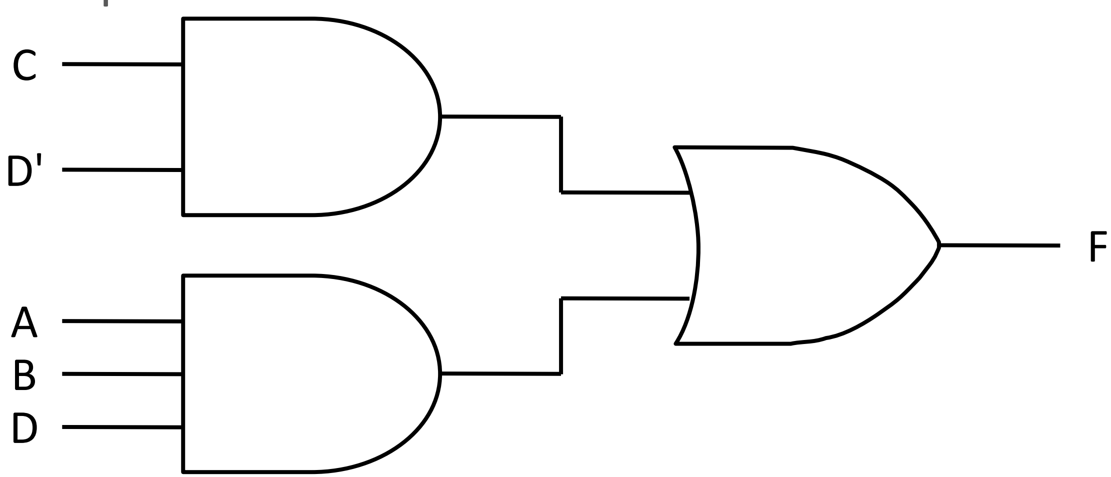
ABCD	F
0000	0
0001	0
0010	1
0011	0
0100	0
0101	0
0110	1
0111	0
1000	0
1001	0
1010	1
1011	0
1100	0
1101	1
1110	1
1111	1

Error Detector for 6-3-1-1 Codes (2/2)

- Design an error detector for 6-3-1-1 codes:
  - The output F = 1 iff inputs (A, B, C, D) represent an **invalid** code combination
  - Step 2: simplify the function

$$
\begin{aligned}
&F(A, B, C, D) \\
&= \sum m(2, 6, 10, 13, 14, 15) \\
&= \underline{A'B'CD' + A'BCD'} + \underline{AB'CD' + ABCD'} + \underline{ABC'D + ABCD} \\
&= \underline{A'CD' + ACD'} + ABD \\
&= CD' + ABD
\end{aligned}
$$

  - Step 3: realize it



ABCD	F
0000	0
0001	0
0010	1
0011	0
0100	0
0101	0
0110	1
0111	0
1000	0
1001	0
1010	1
1011	0
1100	0
1101	1
1110	1
1111	1

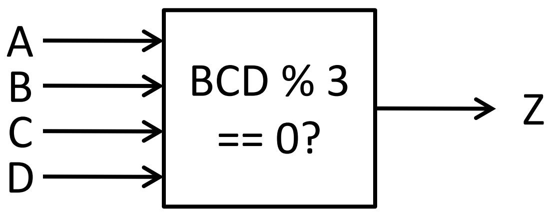
Another Example

- The output Z = 1 iff the 8-4-2-1 BCD number (A, B, C, D) is divisible by 3
  - Step 1: construct the truth table
  - Step 2: simplify the function

    Z(A, B, C, D)

    $= \sum m(0, 3, 6, 9) +$

    $\sum d(10, 11, 12, 13, 14, 15)$
  - Step 3: realize it
    - ?
    - Unit 5!



Decimal Digit	8-4-2-1 Code
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

ABCD	Z
0000	1
0001	0
0010	0
0011	1
0100	0
0101	0
0110	1
0111	0
1000	0
1001	1
1010	X
1011	X
1100	X
1101	X
1110	X
1111	X

Outline

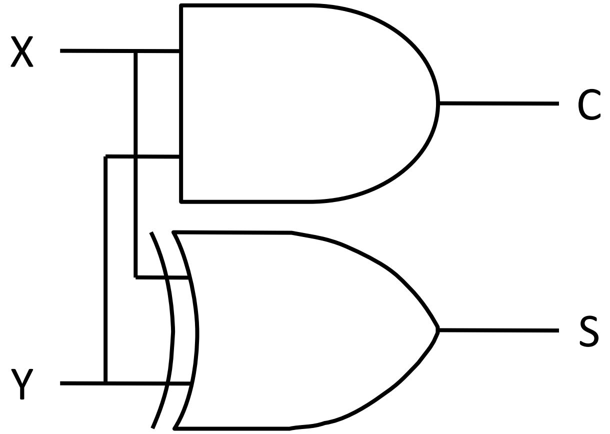
- ❑ Conversion of English Sentences to Boolean Equations
- ❑ Combinational Logic Design Using a Truth Table
- ❑ Minterm and Maxterm Expansions
- ❑ General Minterm and Maxterm Expansions
- ❑ Incompletely Specified Functions
- ❑ Examples of Truth Table Construction
- ❑ **Design of Binary Adders and Subtracters**

1-Bit Half Adder (HA)

- Step 1: construct the truth table

X	Y	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

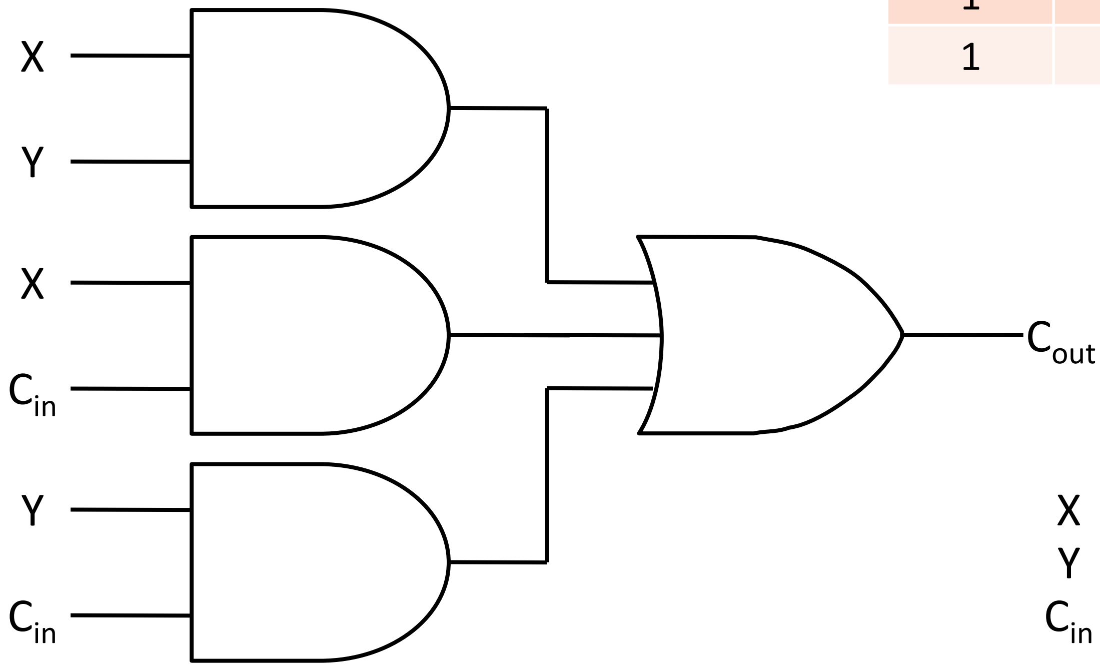
- Step 2: simplify the function
  - $C = XY$
  - $S = X'Y + XY' = X \oplus Y$
- Step 3: realize it

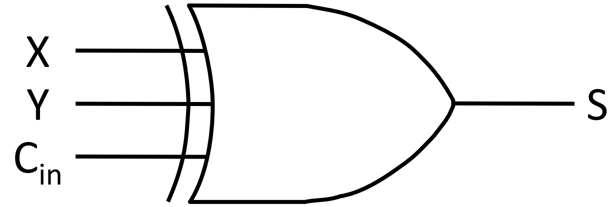


1-Bit Full Adder (FA)

- Step 1: construct the truth table
- Step 2: simplify the function
  - $C_{out} = X'YC_{in} + XY'C_{in} + XYC_{in}' + XYC_{in}$
    $= XY + XC_{in} + YC_{in}$
  - $S = X'Y'C_{in} + X'YC_{in}' + XY'C_{in}' + XYC_{in}$
- Step 3: realize it

X	Y	C_{in}	C_{out}	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





4-Bit Parallel Adder (1/3)

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

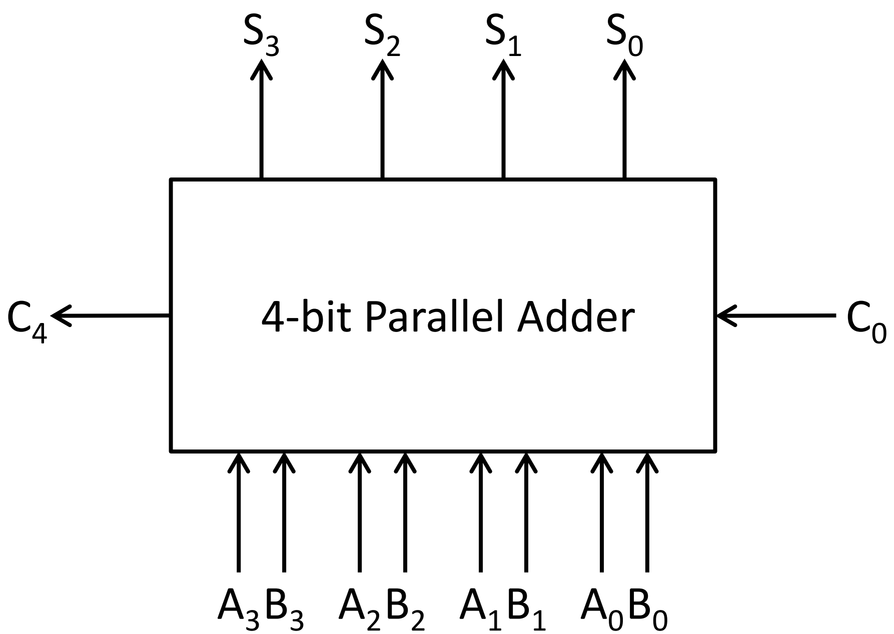
$$\begin{array}{lllllllll} & & & A_3 & A_2 & A_1 & A_0 & & \\ & + & & B_3 & B_2 & B_1 & B_0 & & \\ \hline C_4 & \leftarrow & & S_3 & S_2 & S_1 & S_0 & \leftarrow & C_0 \end{array}$$

carry-out carry-in

- Example

$$\begin{array}{lllllllll} & & & 1 & 0 & 1 & 1 & 0 & \leftarrow \text{carries} \\ & & & & 1 & 0 & 1 & 1 & \\ & + & & & 1 & 0 & 1 & 1 & \\ \hline 1 & \leftarrow & & 1 & 0 & 1 & 1 & 0 & \leftarrow 0 \end{array}$$

- How?
  - Step 1: construct the truth table
  - ...

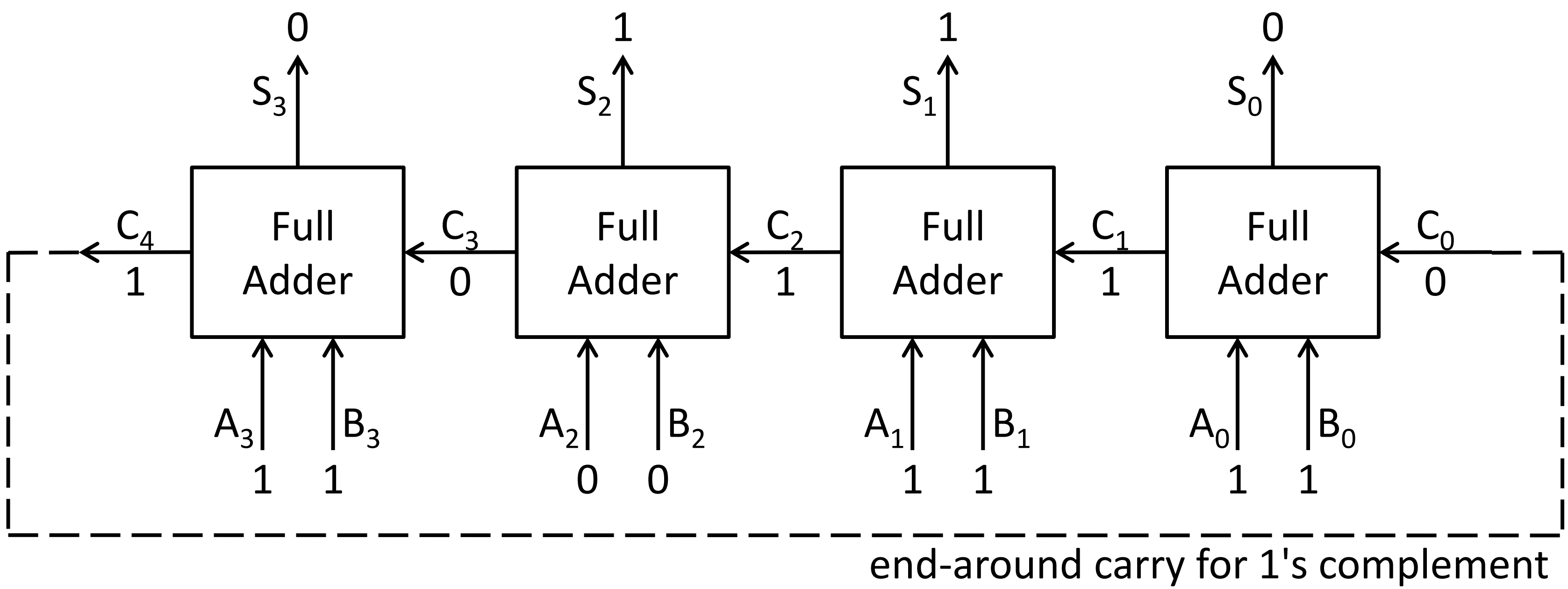


4-Bit Parallel Adder (2/3)

- Decompose the 4 bit adder into four modules
  - Each module adds two bits and a carry → use full adder
- Extend to negative numbers
  - Consider 1's complement
    - Add just as if all numbers are positive
    - Add the carry out back to the rightmost bit
  - How to detect overflow?
    - Check the sign
      - (+) + (+) becomes (-)
      - (-) + (-) becomes (+)

Case 1	Case 2	Case 3	Case 4	Case 5	Case 6
+3 0011	+5 0101	+5 0101	−5 1010	−3 1100	−5 1010
+4 0100	+6 0110	−6 1001	+6 0110	−4 1011	−6 1001
+7 0111	1011	−1 1110	+1 (1) 0000	−7 (1) 0111	(1) 0011
			1	1	1
			0001	1000	0100

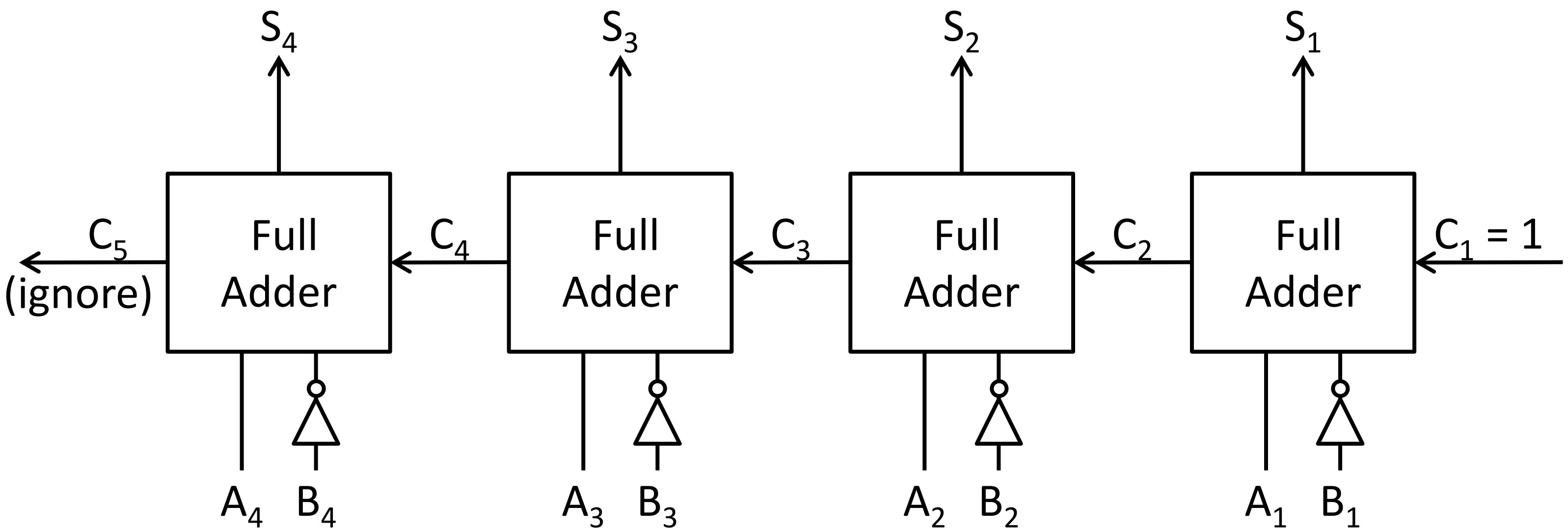
4-Bit Parallel Adder (3/3)



- Overflow detection?
  - $V = A_3'B_3'S_3 + A_3B_3S_3'$
  - Why?

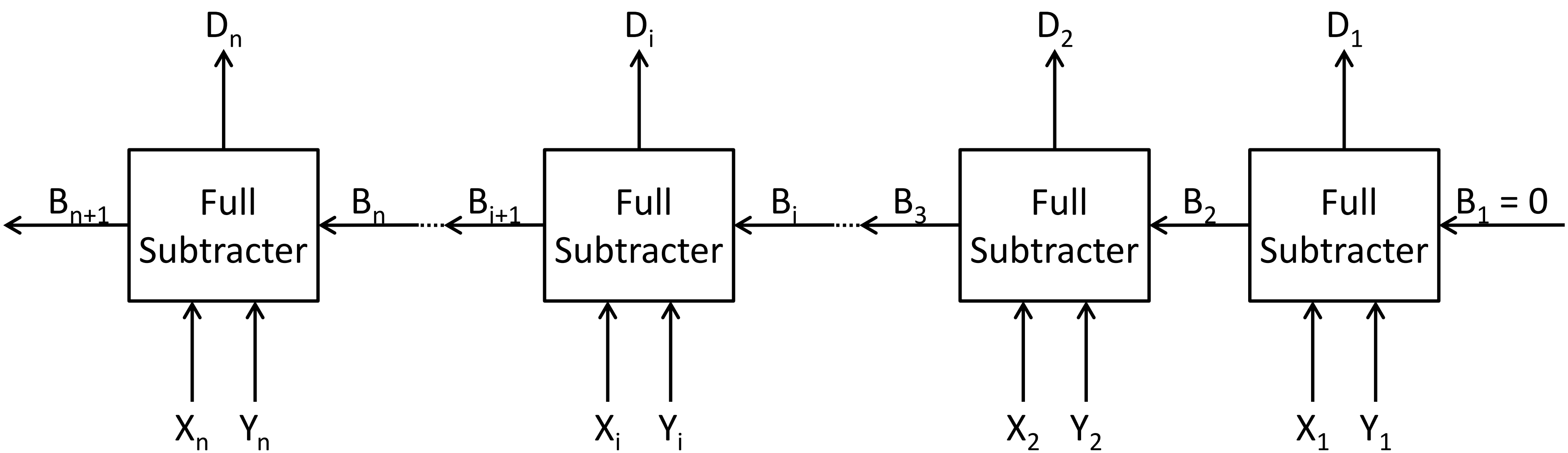
Binary Subtracter (1/2)

- Consider A – B = A + (-B) in 2's complement
  - A – B = A + (-B) = A + B* = A + $\overline{B}$ + 1
  - Convert B to 2's complement: inverse and then add 1
- Discard the carry from the sign bit



Binary Subtracter (2/2)

- Or design a full subtracter
  - D = X – Y: difference
  - B: borrow



Q&A