Lecture #17

Combinational Circuits

NUS
National University
of Singapore

School of
Computing

Lecture #17: Combinational Circuits

1. Introduction
2. Analysis Procedure
3. Design Methods
4. Gate-Level (SSI) Design
5. Block-Level Design
6. Summary of Arithmetic Circuits
7. Example: 6-Person Voting System
8. Magnitude Comparator
9. Circuit Delays

1. Introduction

- ## Two classes of logic circuits
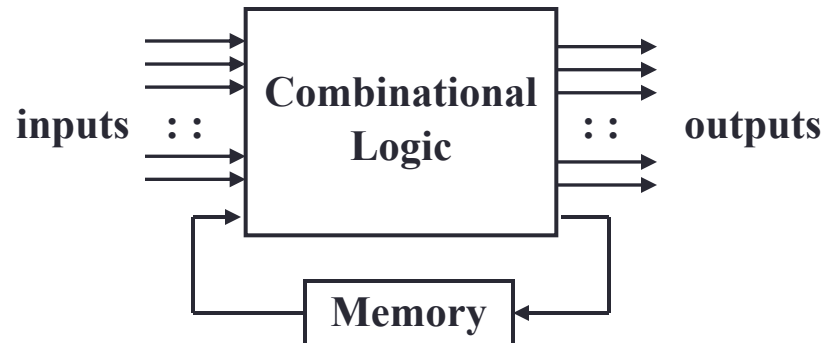 - Combinational
  - Sequential

- ## Combinational Circuit
  - Each output depends entirely on the immediate (present) inputs.
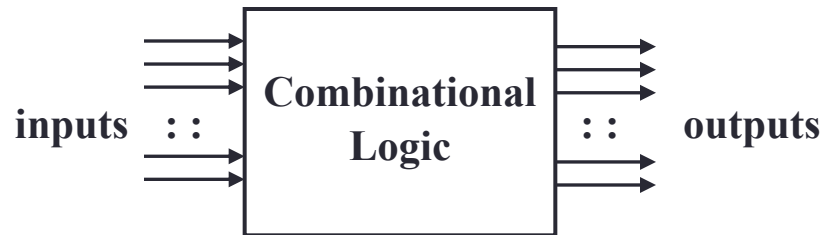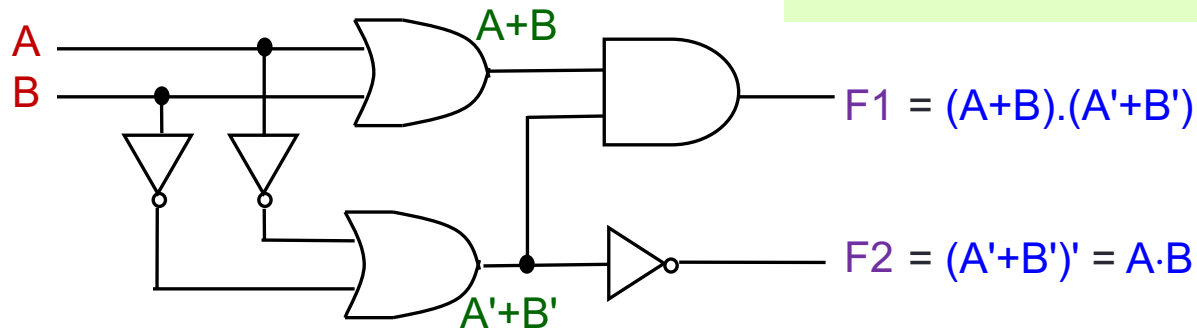

- ## Sequential Circuit
  - Each output depends on both present inputs and state.

2. Analysis Procedure

▪ Given a combinational circuit, how do you analyze its function?

What is this circuit?

A+B

F1 = (A+B).(A'+B')

F2 = (A'+B')' = A·B

A'+B'

▪ Steps:

1. Label the inputs and outputs.

2. Obtain the functions of intermediate points and the outputs.

3. Draw the truth table.

4. Deduce the functionality of the circuit ➲ Half adder.

A	B	(A+B)	(A'+B')	F1	F2
0	0	0	1	0	0
0	1	1	1	1	0
1	0	1	1	1	0
1	1	1	0	0	1

3. Design Methods

- Different combinational circuit design methods:
  - Gate-level design method (with logic gates)
  - Block-level design method (with functional blocks)
- Design methods make use of logic gates and useful function blocks
  - These are available as Integrated Circuit (IC) chips.
  - Types of IC chips (based on packing density): SSI, MSI, LSI, VLSI, ULSI.
- Main objectives of circuit design:
  - Reduce cost (number of gates for small circuits; number of IC packages for complex circuits)
  - Increase speed
  - Design simplicity (re-use blocks where possible)

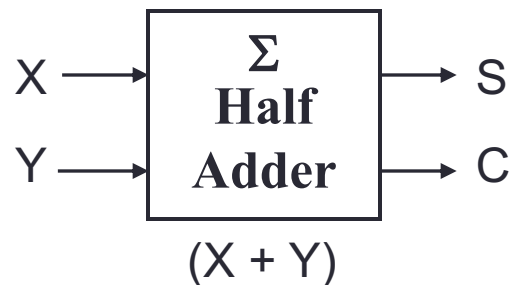# 4. Gate-Level (SSI) Design: Half Adder (1/2)

- Design procedure:
  1. State problem
     Example: Build a Half Adder.

  2. Determine and label the inputs and outputs of circuit.
     Example: Two inputs and two outputs labelled, as shown below.

X ⟶ | Σ **Half Adder** | ⟶ S

Y ⟶ | | ⟶ C

(X + Y)

X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

3. Draw the truth table.

4. Gate-Level (SSI) Design: Half Adder (2/2)
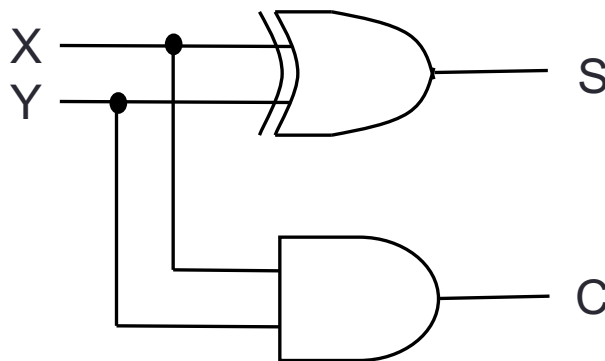
4. Obtain simplified Boolean functions.
   Example: $C = X \cdot Y$
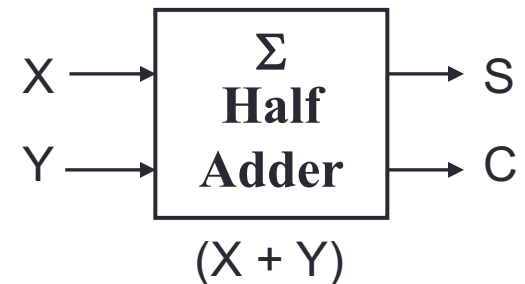 $S = X' \cdot Y + X \cdot Y' = X \oplus Y$

X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

5. Draw the logic diagram.

Half Adder

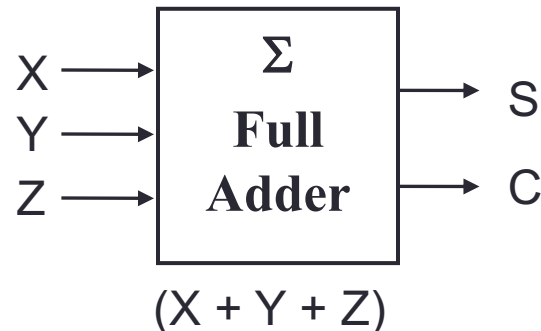Block diagram of Half Adder

$(X + Y)$

4. Gate-Level (SSI) Design: Full Adder (1/5)

- Half adder adds up only two bits.

- To add two binary numbers, we need to add 3 bits (including the *carry*).
  - Example:

$$
\begin{array}{ccccc}
1 & 1 & 1 & & \text{carry} \\
0 & 0 & 1 & 1 & X \\
+ \quad 0 & 1 & 1 & 1 & Y \\
\hline
1 & 0 & 1 & 0 & S \\
\hline
\end{array}
$$

- Need Full Adder (so called as it can be made from two half adders).

X ⟶ ⎡ Σ ⎤ ⟶ S
Y ⟶ | **Full** |
Z ⟶ | **Adder** | ⟶ C

(X + Y + Z)

4. Gate-Level (SSI) Design: Full Adder (2/5)

- Truth table:

X	Y	Z	C	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

Note:
  Z - carry in (to the current position)
  C - carry out (to the next position)

C

YZ\X	00	01	11	10
0	0	0	1	0
1	0	1	1	1

- Using K-map, simplified SOP form:

$C = X \cdot Y + X \cdot Z + Y \cdot Z$

$S = X' \cdot Y' \cdot Z + X' \cdot Y \cdot Z' + X \cdot Y' \cdot Z' + X \cdot Y \cdot Z$

S

YZ\X	00	01	11	10
0	0	1	0	1
1	1	0	1	0

4. Gate-Level (SSI) Design: Full Adder (3/5)

- Alternative formulae using algebraic manipulation:

C   = X·Y + X·Z + Y·Z
    = X·Y + (X + Y)·Z
    = X·Y + ((X⊕Y) + X·Y)·Z
    = X·Y + (X⊕Y)·Z + X·Y·Z
    = X·Y + (X⊕Y)·Z

S   = X'·Y'·Z + X'·Y·Z' + X·Y'·Z' + X·Y·Z
    = X'·(Y'·Z + Y·Z') + X·(Y'·Z' + Y·Z)
    = X'·(Y⊕Z) + X·(Y⊕Z)'
    = X⊕(Y⊕Z)

4. Gate-Level (SSI) Design: Full Adder (4/5)

- Circuit for above formulae:

C   = X·Y + (X⊕Y)·Z

S   = X⊕(Y⊕Z) = (X⊕Y)⊕Z (XOR is associative)
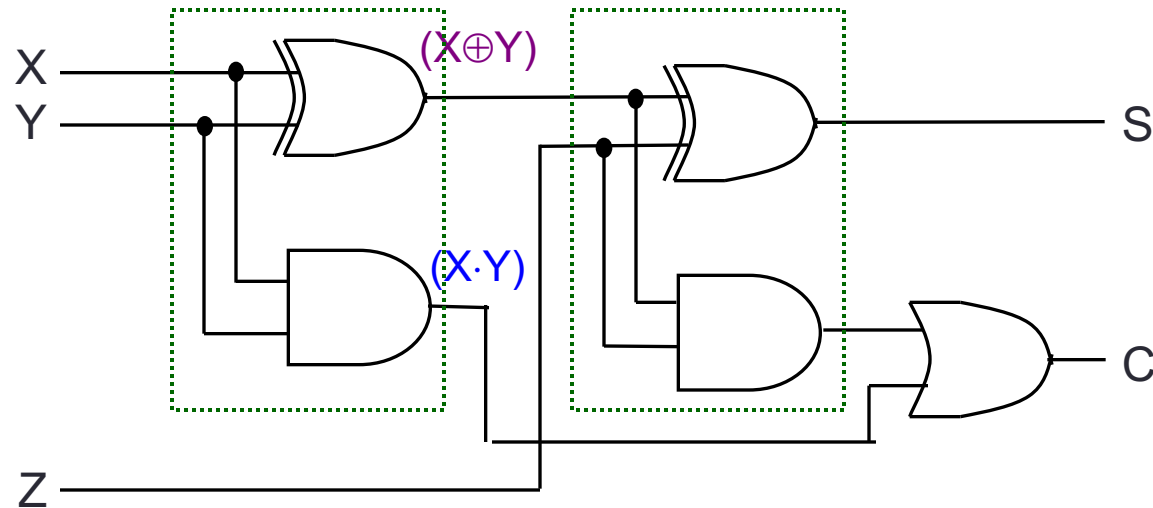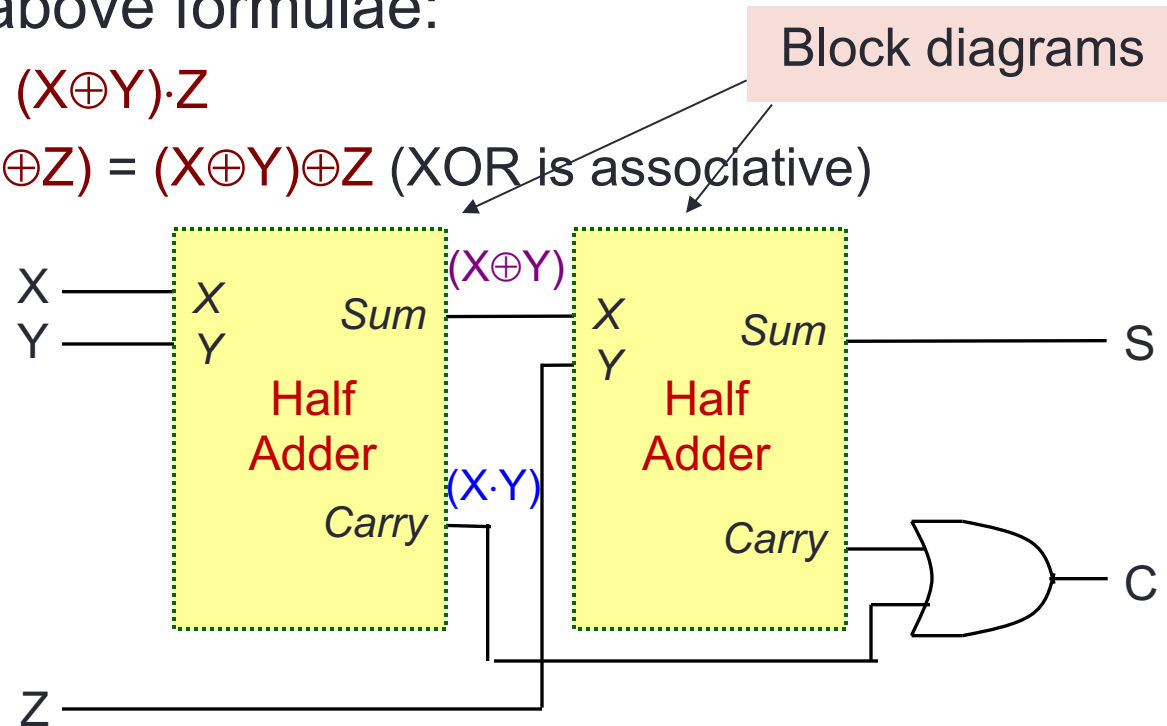

Full Adder made from two <u>Half-Adders</u> (+ an OR gate).

4. Gate-Level (SSI) Design: Full Adder (5/5)
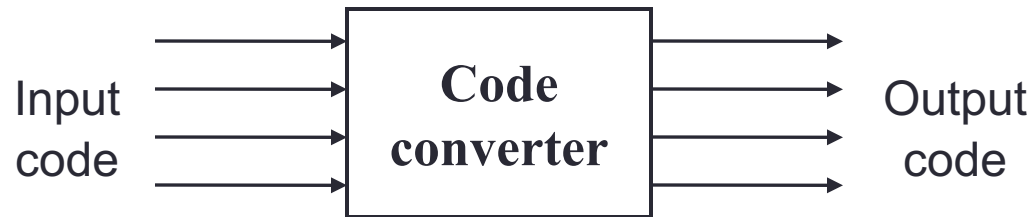
- Circuit for above formulae:

  C   = X·Y + (X⊕Y)·Z

  S   = X⊕(Y⊕Z) = (X⊕Y)⊕Z (XOR is associative)

Block diagrams

Full Adder made from two <u>Half-Adders</u> (+ an OR gate).

4. Gate-Level (SSI) Design: Code Converters

■ Code converter – takes an input code, translates to its equivalent output code.

Input
code
Code converter
Output
code

■ Example: BCD to Excess-3 code converter.

  ■ Input: BCD code
  ■ Output: Excess-3 code

4. BCD to Excess-3 Code Converter (1/3)

Digit	BCD code	Excess-3 code
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

4. BCD to Excess-3 Code Converter (2/3)

- Truth table:

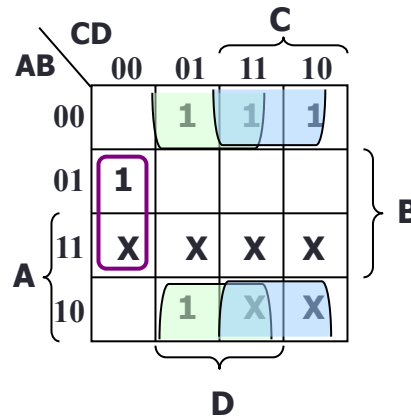|	BCD				Excess-3			
	A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0
10	1	0	1	0	X	X	X	X
11	1	0	1	1	X	X	X	X
12	1	1	0	0	X	X	X	X
13	1	1	0	1	X	X	X	X
14	1	1	1	0	X	X	X	X
15	1	1	1	1	X	X	X	X

- K-maps:

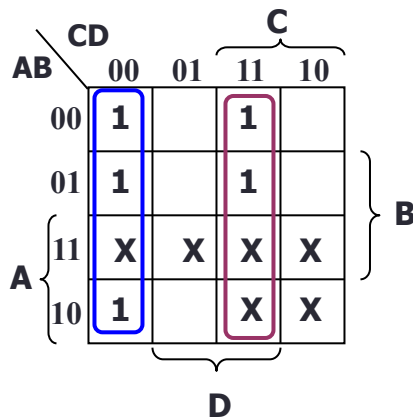# 4. BCD to Excess-3 Code Converter (3/3)

W

X

Y

Z

$$W = A + B \cdot C + B \cdot D$$
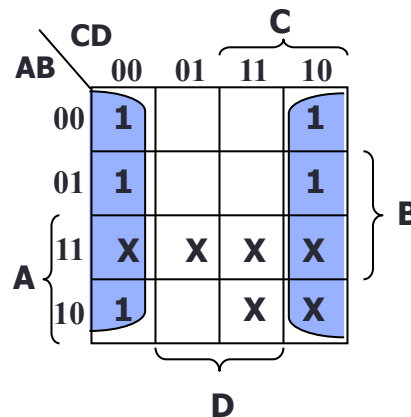
$$X = B' \cdot C + B' \cdot D + B \cdot C' \cdot D'$$

$$Y = C.D + C'.D'$$

$$Z = D'$$

5. Block-Level Design

- More complex circuits can also be built using block-level method.

- In general, block-level design method (as opposed to gate-level design) relies on algorithms or formulae of the circuit, which are obtained by decomposing the main problem to sub-problems recursively (until small enough to be directly solved by blocks of circuits).

- First example shows how to create a 4-bit parallel adder using block-level design.

- Using 4-bit parallel adders as building blocks, we can create the following:
    1. BCD-to-Excess-3 Code Converter
    2. 16-bit Parallel Adder

5. 4-bit Parallel Adder (1/4)

- Consider a circuit to add two 4-bit numbers together and a carry-in, to produce a 5-bit result.

Black-box view of 4-bit parallel adder
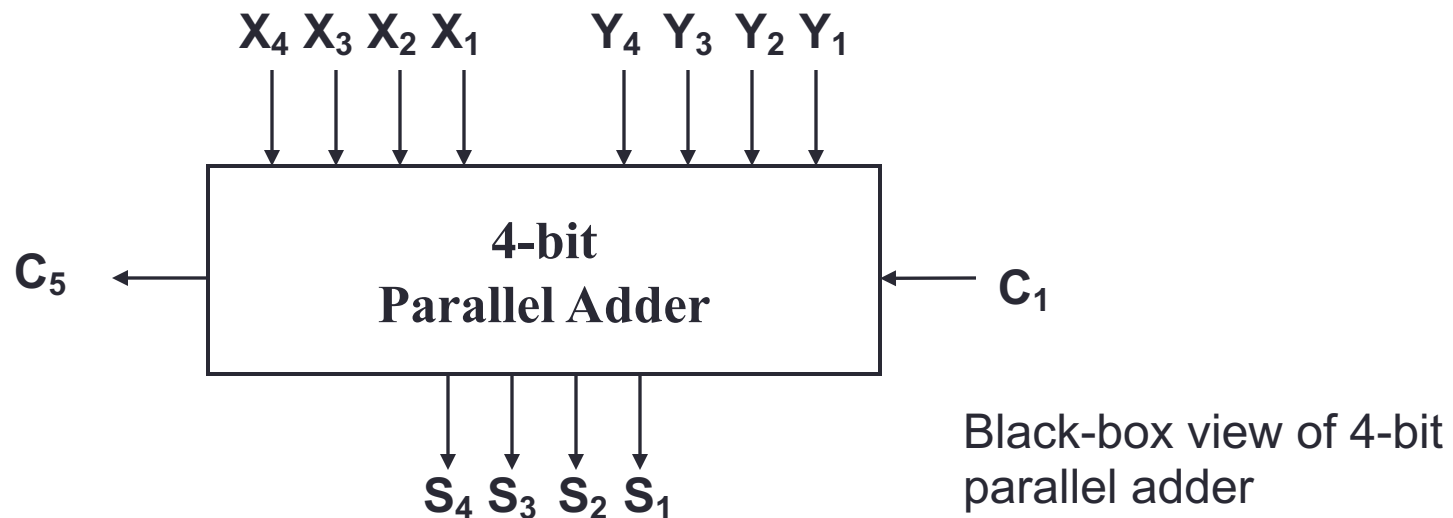
- 5-bit result is sufficient because the largest result is:
$$1111_2 + 1111_2 + 1_2 = 11111_2$$

5. 4-bit Parallel Adder (2/4)

- SSI design (gate-level design) technique should not be used here.
- Truth table for 9 inputs is too big: 2^9 = 512 rows!

$X_4X_3X_2X_1$	$Y_4Y_3Y_2Y_1$	C_1	C_5	$S_4S_3S_2S_1$
0 0 0 0	0 0 0 0	0	0	0 0 0 0
0 0 0 0	0 0 0 0	1	0	0 0 0 1
0 0 0 0	0 0 0 1	0	0	0 0 0 1
…	…	…	…	…
0 1 0 1	1 1 0 1	1	1	0 0 1 1
…	…	…	…	…
1 1 1 1	1 1 1 1	1	1	1 1 1 1

- Simplification becomes too complicated!

5. 4-bit Parallel Adder (3/4)

- Alternative design possible.

- Addition formula for each pair of bits (with carry in),

  $C_{i+1}S_i = X_i + Y_i + C_i$

  has the same function as a full adder:

  $C_{i+1} = X_i \cdot Y_i + (X_i \oplus Y_i) \cdot C_i$
  $S_i = X_i \oplus Y_i \oplus C_i$

C =	1 1 0 0
X =	1 0 1 0
Y =	1 1 1 1
X + Y =	1 1 0 0 1

5. 4-bit Parallel Adder (4/4)

- Cascading 4 full adders via their carries, we get:
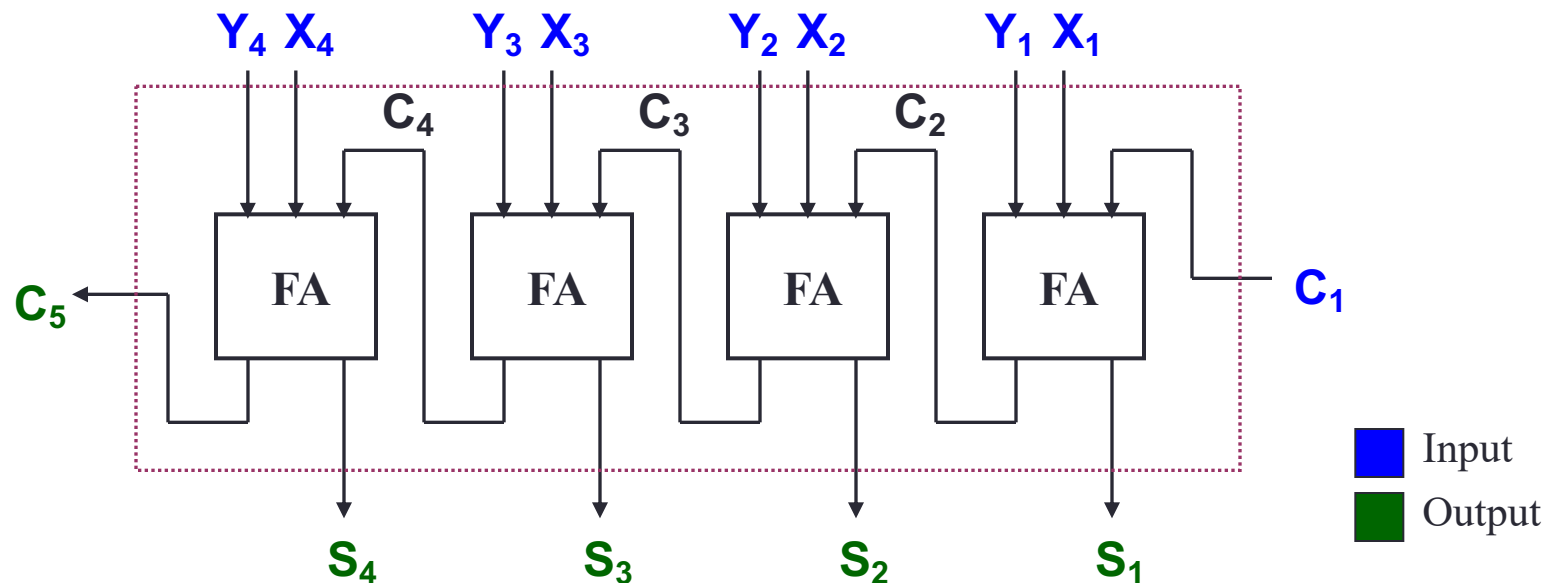
- Note that carry is propagated by cascading the carry from one full adder to the next.

- Called Parallel Adder because inputs are presented simultaneously (in parallel). Also called Ripple-Carry Adder.

5. BCD to Excess-3 Converter: Revisit (1/2)

- Excess-3 code can be converted from BCD code using truth table:

- Gate-level design can be used since only 4 inputs.

- However, alternative design is possible.

- Use problem-specific formula:

  Excess-3 code
   = BCD Code + 0011_2

	BCD				Excess-3			
	A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0
10	1	0	1	0	X	X	X	X
11	1	0	1	1	X	X	X	X
12	1	1	0	0	X	X	X	X
13	1	1	0	1	X	X	X	X
14	1	1	1	0	X	X	X	X
15	1	1	1	1	X	X	X	X

5. BCD to Excess-3 Converter: Revisit (2/2)

- **Block-level circuit:**

**A BCD to Excess-3
Code Converter**

unused

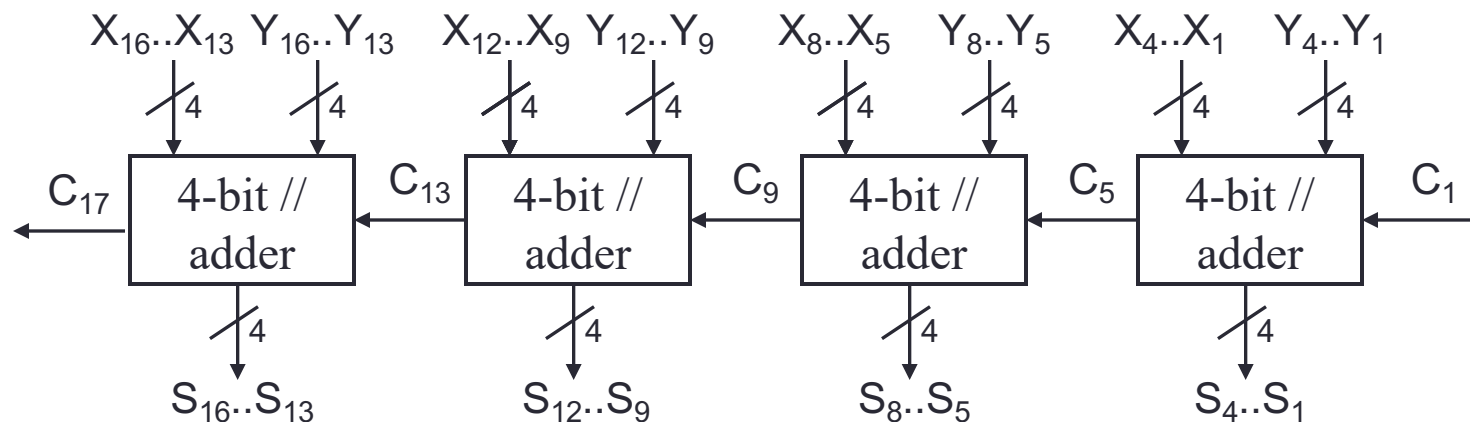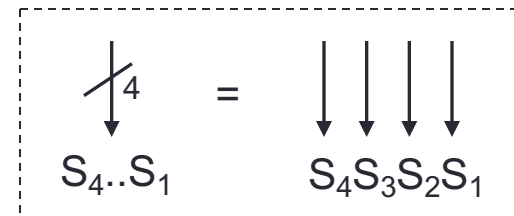|		
BCD code → X_4, X_3, X_2, X_1	**4-bit Parallel Adder** (Cout / Cin)	S_4, S_3, S_2, S_1 → Excess-3 code
0 → Y_4		
0 → Y_3		
1 → Y_2		
1 → Y_1		

0
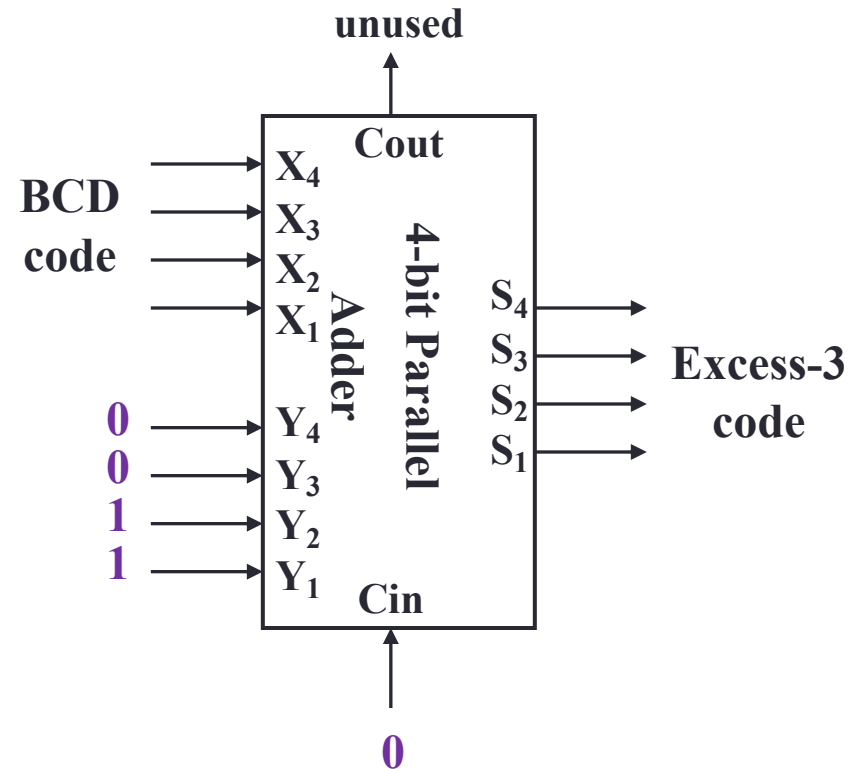
Note: In the lab, input 0 (low) is connected to GND,
1 (high) to Vcc.

5. 16-bit Parallel Adder

- Larger parallel adders can be built from smaller ones.

- Example: A 16-bit parallel adder can be constructed from four 4-bit parallel adders:

A 16-bit parallel adder

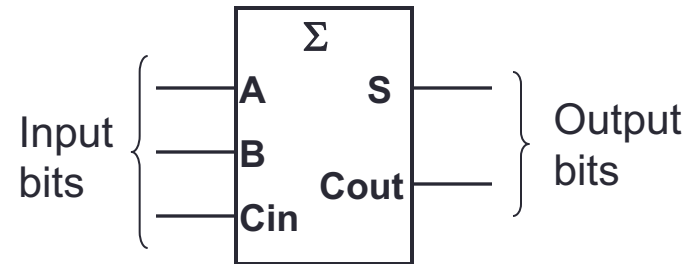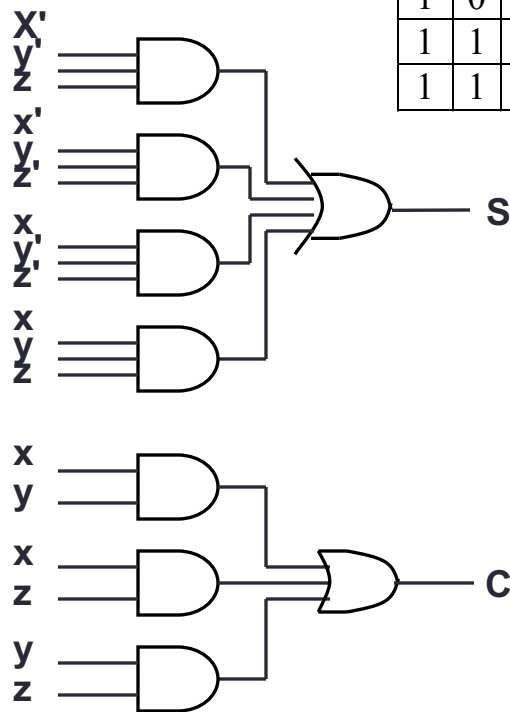# 6. Summary of Arithmetic Circuits (1/4)

- ## Half adder

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0
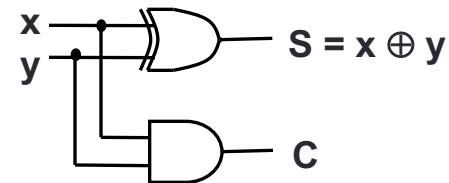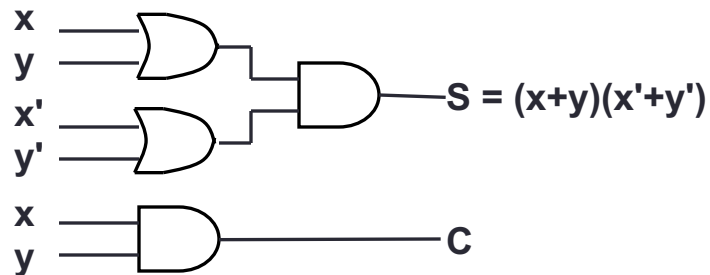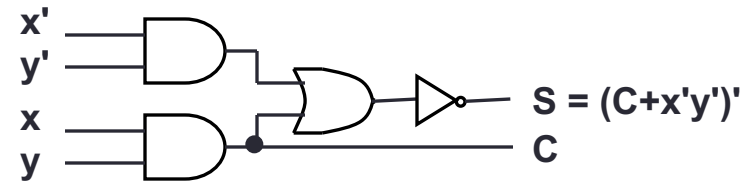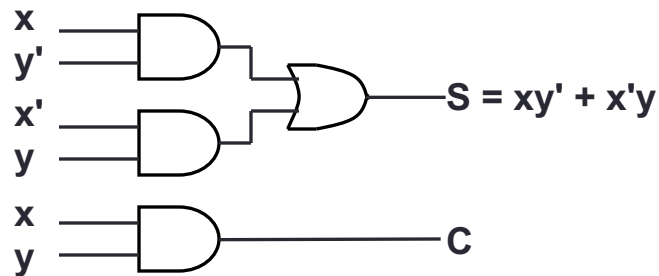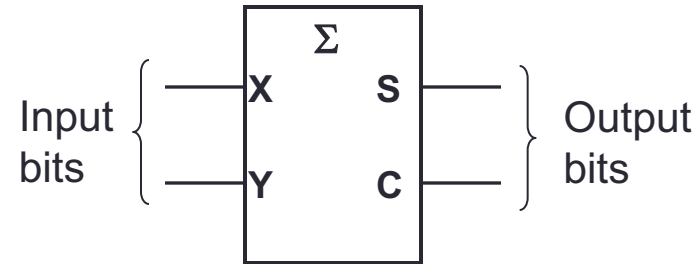
6. Summary of Arithmetic Circuits (2/4)

- Full adder

x	y	z	C	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

$C = xy + xz + yz$

$S = x'y'z + x'yz' + xy'z' + xyz$

$S = (x \oplus y) \oplus z$

$C = x \cdot y + (x \oplus y) \cdot z$
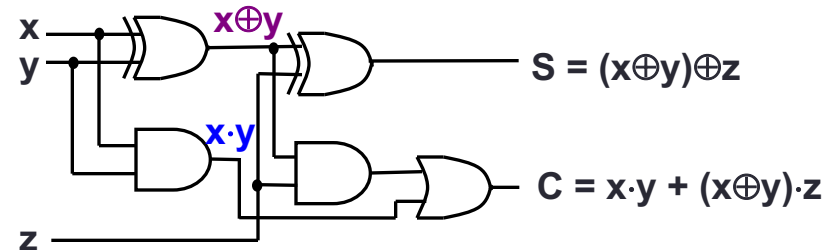
6. Summary of Arithmetic Circuits (3/4)

- 4-bit parallel adder

| Subscript i	4	3	2	1	
Input carry	0	1	1	0	C_i
Augend	1	0	1	1	A_i
Addend	0	0	1	1	B_i
Sum	1	1	1	0	S_i
Output carry	0	0	1	1	C_{i+1}

2 ways:

- ◆ Serial (one FA)
- ◆ Parallel (*n* FAs for *n* bits)
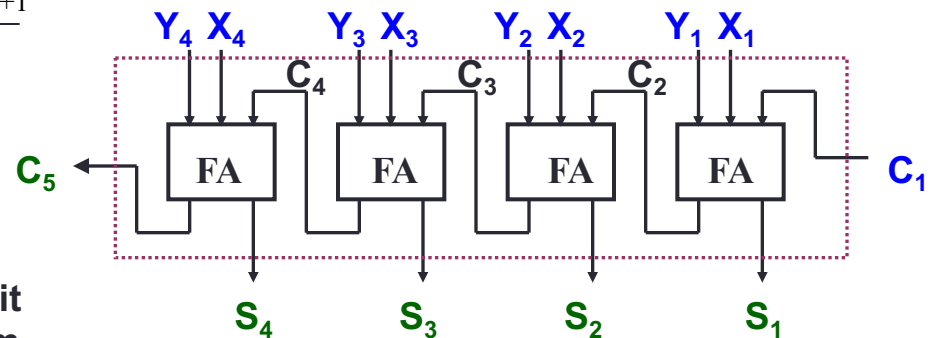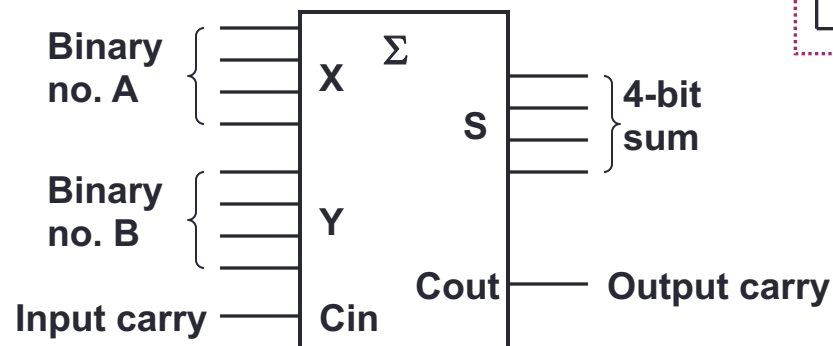
6. Summary of Arithmetic Circuits (4/4)

- Cascading 4 full adders (FAs) gives a 4-bit parallel adder.
  - Classical method: 9 input variables → 2^9 = 512 rows in truth table!
- Cascading method can be extended to larger adders.
  - Example: 16-bit parallel adder.
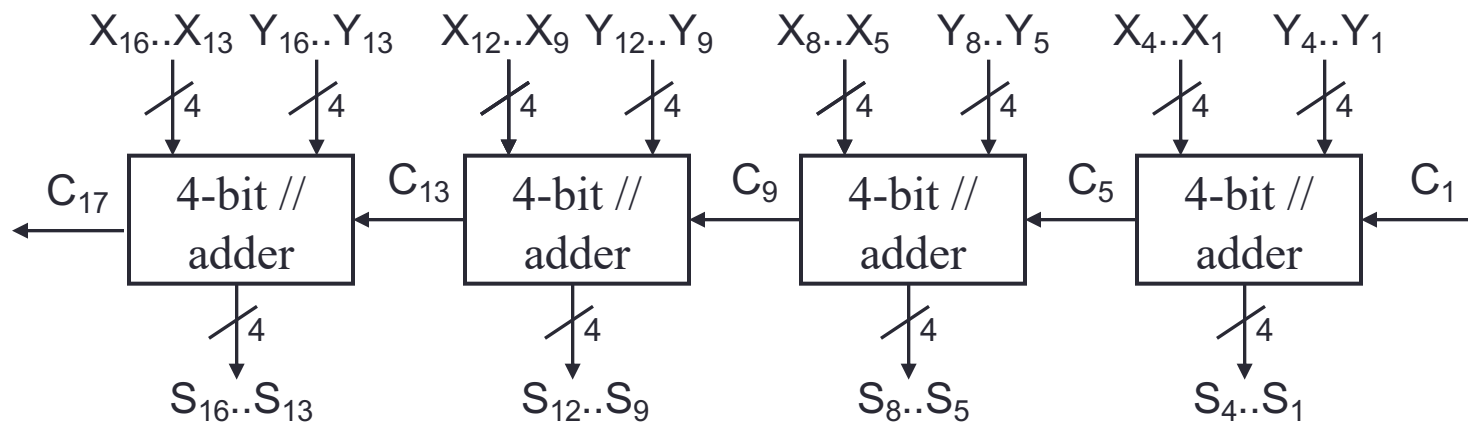
7. Example: 6-Person Voting System

- Application: 6-person voting system.
  - Use FAs and a 4-bit parallel adder.
  - Each FA can sum up to 3 votes.
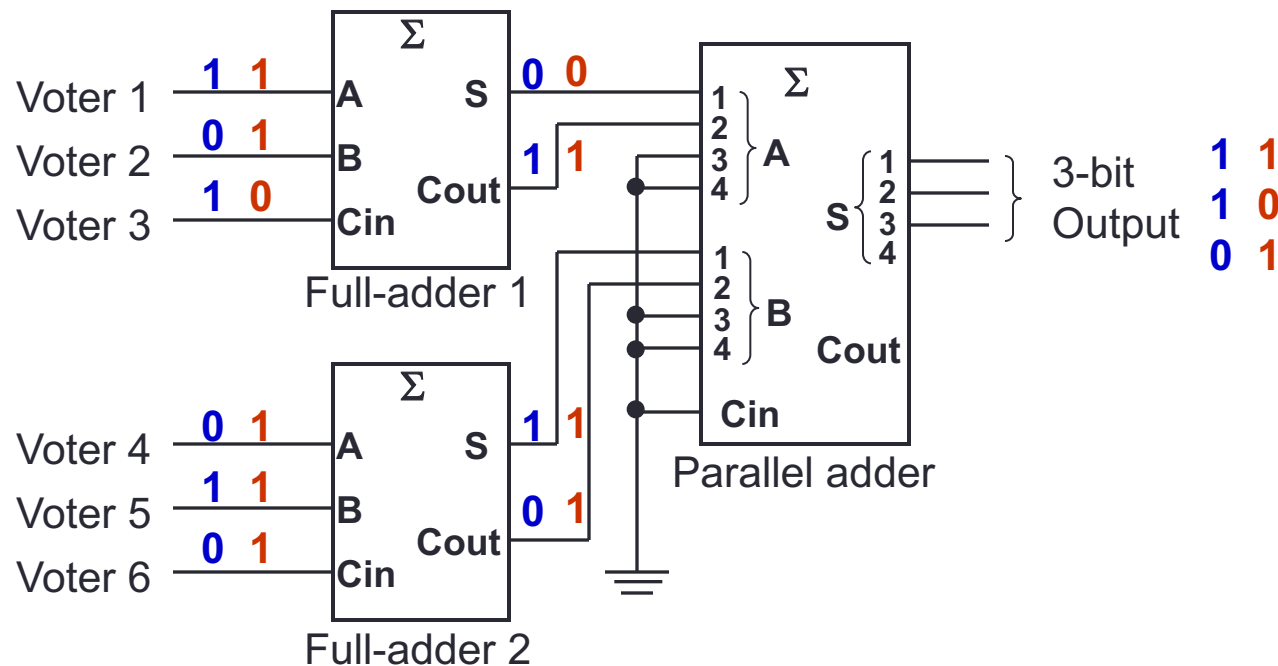
8. Magnitude Comparator (1/4)

- Magnitude comparator: compares 2 unsigned values A and B, to check if A>B, A=B, or A<B.

- To design an n-bit magnitude comparator using classical method, it would require 2^{2n} rows in truth table!

- We shall exploit regularity in our design.

- Question: How do we compare two 4-bit unsigned values A ($a_3a_2a_1a_0$) and B ($b_3b_2b_1b_0$)?
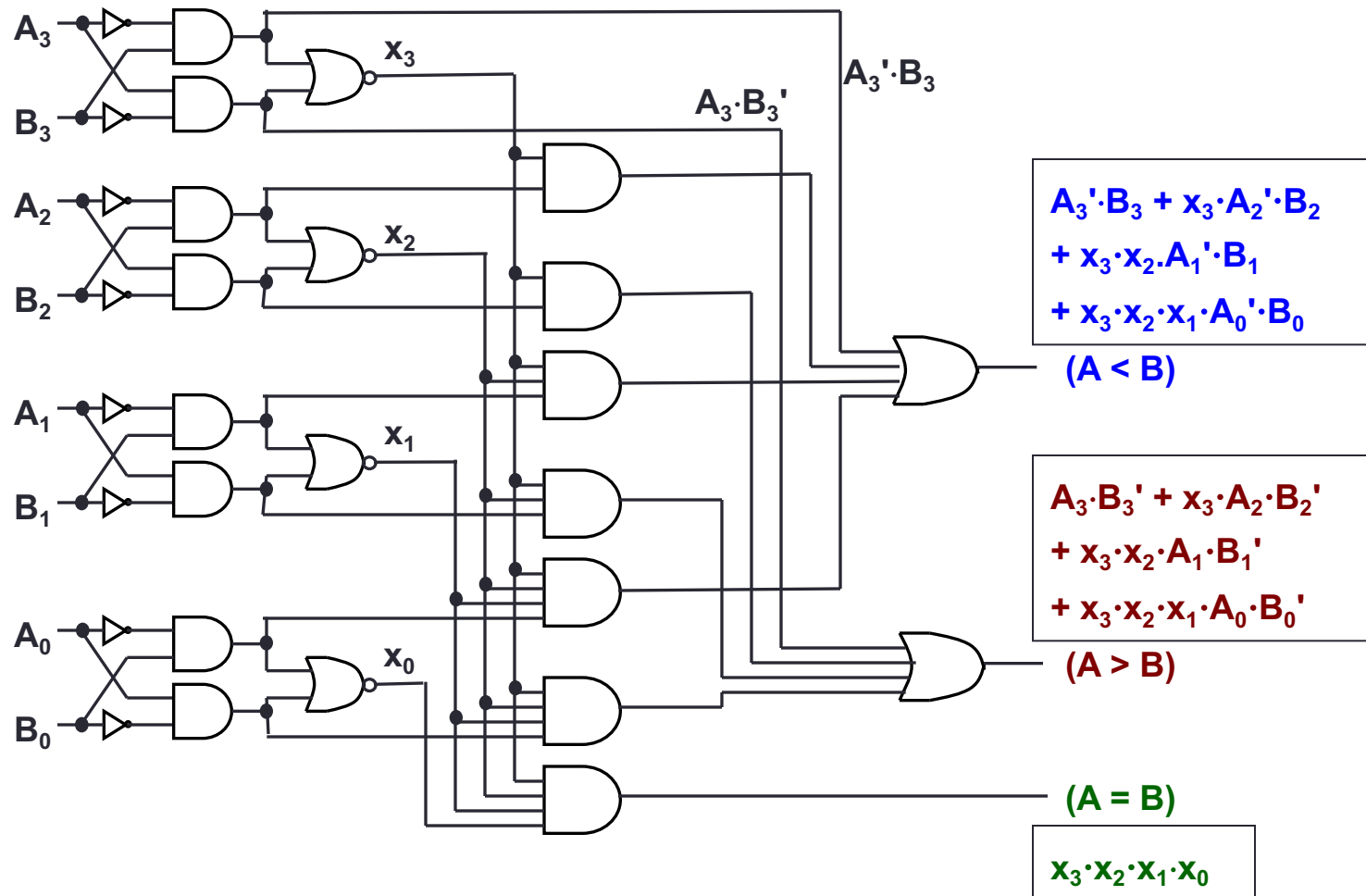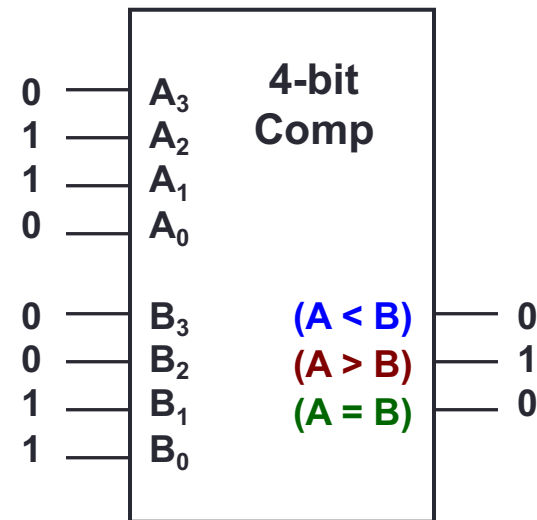
  If ($a_3 > b_3$) then A > B

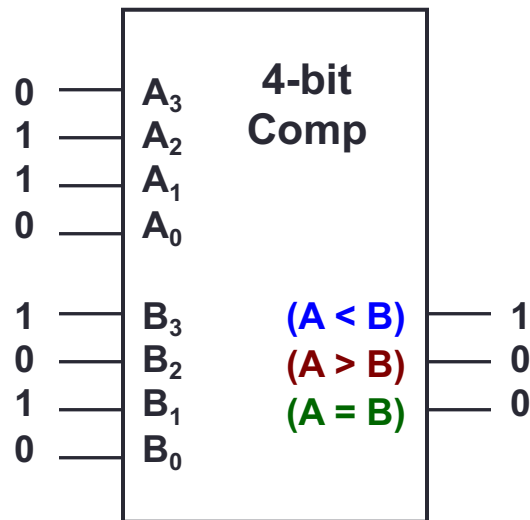 If ($a_3 < b_3$) then A < B

  If ($a_3 = b_3$) then if ($a_2 > b_2$) …

8. Magnitude Comparator (2/4)

Let $A = A_3A_2A_1A_0$, $B = B_3B_2B_1B_0$; $x_i = A_i \cdot B_i + A_i' \cdot B_i'$

$A_3' \cdot B_3 + x_3 \cdot A_2' \cdot B_2$
$+ x_3 \cdot x_2 . A_1' \cdot B_1$
$+ x_3 \cdot x_2 \cdot x_1 \cdot A_0' \cdot B_0$

(A < B)

$A_3 \cdot B_3' + x_3 \cdot A_2 \cdot B_2'$
$+ x_3 \cdot x_2 \cdot A_1 \cdot B_1'$
$+ x_3 \cdot x_2 \cdot x_1 \cdot A_0 \cdot B_0'$

(A > B)

(A = B)

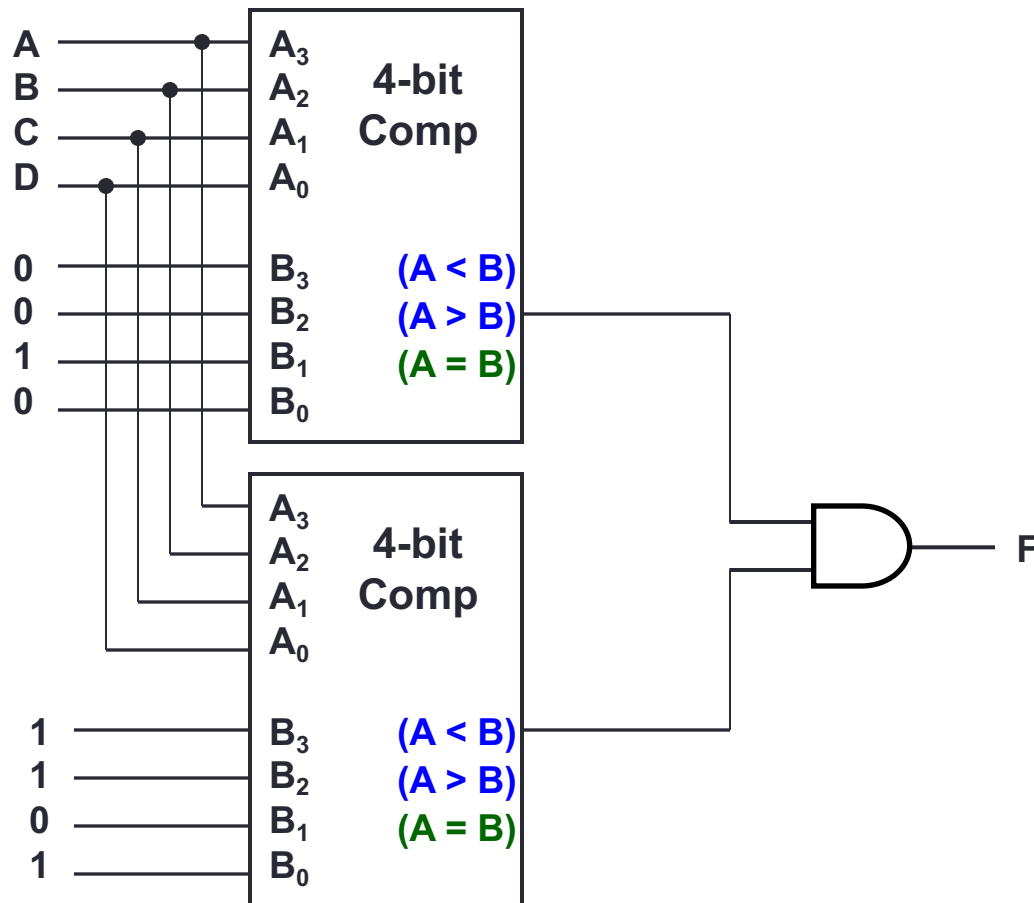$x_3 \cdot x_2 \cdot x_1 \cdot x_0$

8. Magnitude Comparator (3/4)

▪ Block diagram of a 4-bit magnitude comparator

8. Magnitude Comparator (4/4)
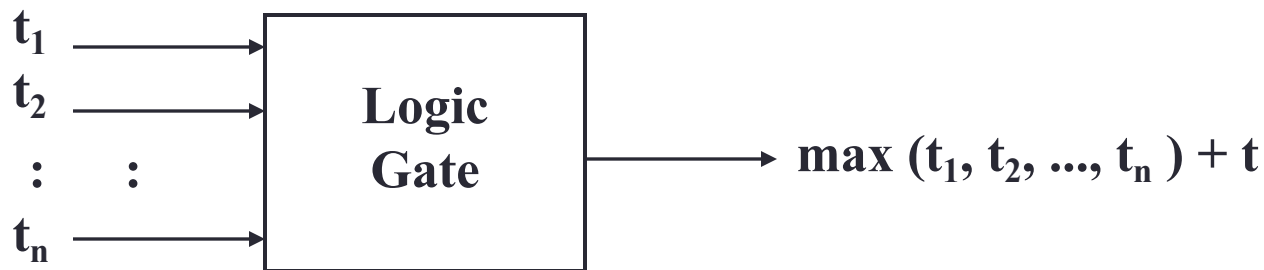
- A function F accepts a 4-bit binary value ABCD, and returns 1 if 3 ≤ ABCD ≤ 12, or 0 otherwise. How would you implement F using magnitude comparators and a suitable logic gate?

9. Circuit Delays (1/5)

- Given a logic gate with delay t. If inputs are stable at times t_1, t_2, …, t_n, then the earliest time in which the output will be stable is:
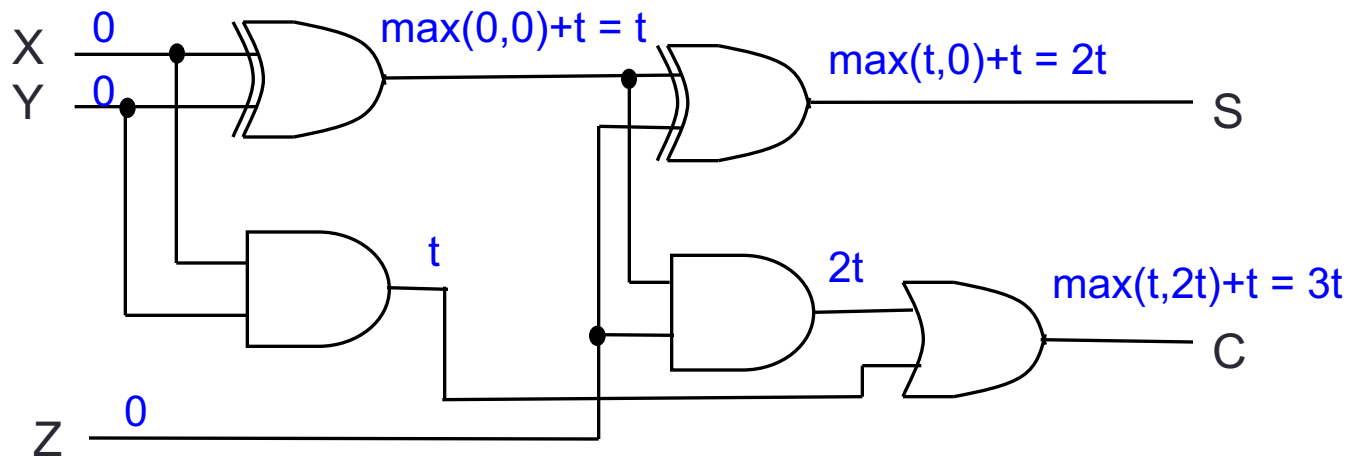
$$max(t_1, t_2, …, t_n) + t$$

- To calculate the delays of all outputs of a combinational circuit, repeat above rule for all gates.
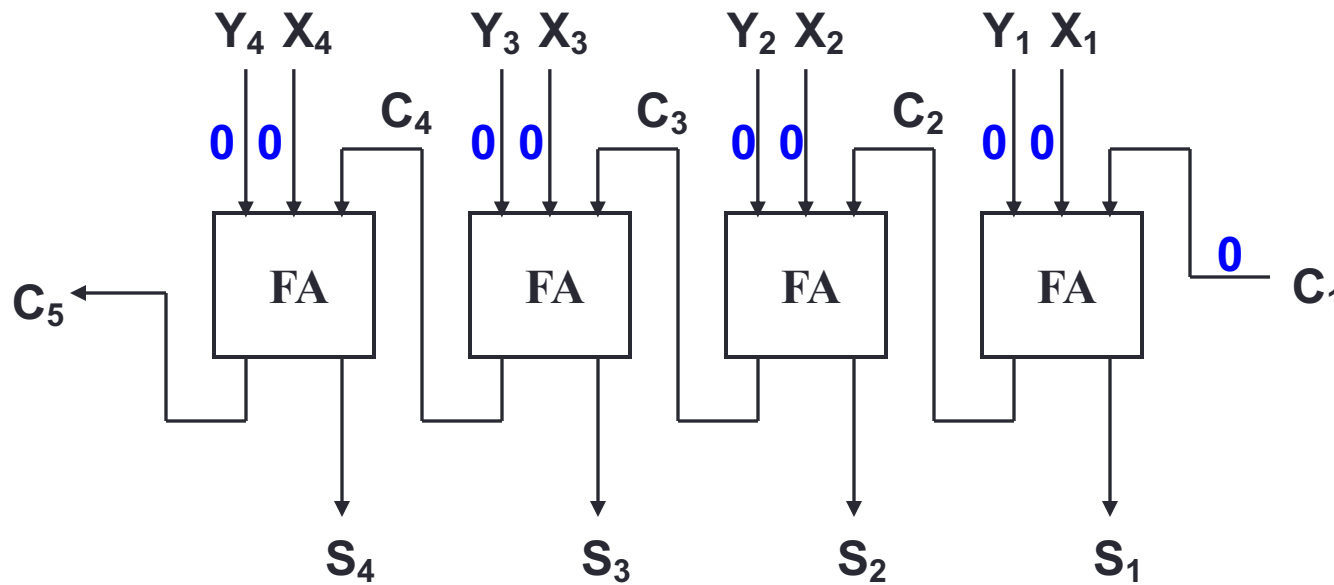
9. Circuit Delays (2/5)

- As a simple example, consider the full adder circuit where all inputs are available at time 0. Assume each gate has delay t.

- Outputs **S** and **C** experience delays of **2t** and **3t** respectively.

9. Circuit Delays (3/5)
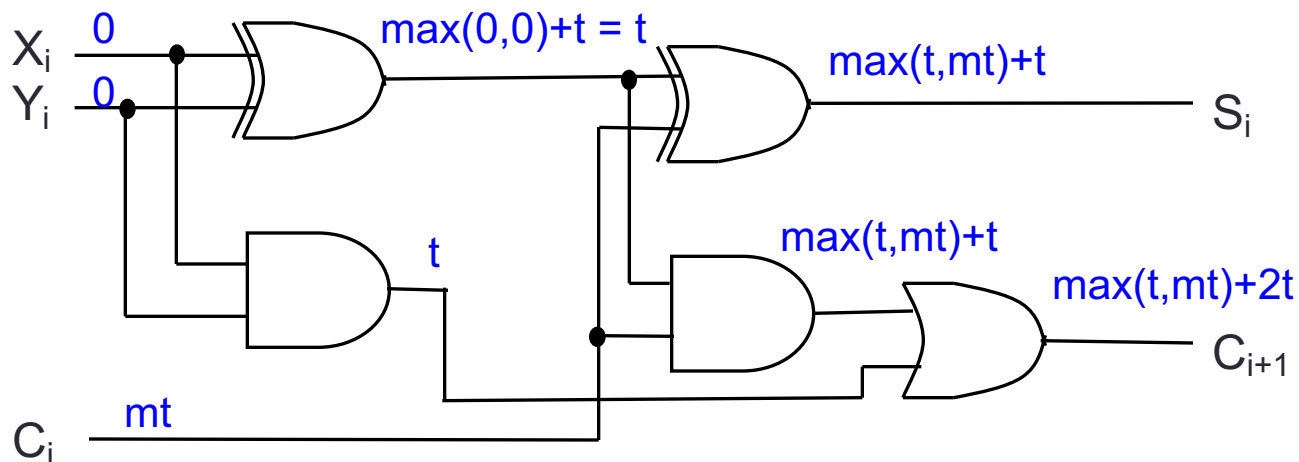
- More complex example: 4-bit parallel adder.

9. Circuit Delays (4/5)

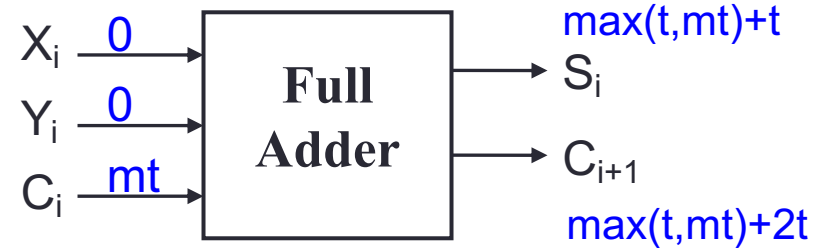- Analyse the delay for the repeated block.

where X_i, Y_i are stable at 0t, while C_i is assumed to be stable at mt.

- Performing the delay calculation:

9. Circuit Delays (5/5)

$X_i \xrightarrow{\ 0\ }$

$Y_i \xrightarrow{\ 0\ }$

$C_i \xrightarrow{\ mt\ }$

Full Adder

$\xrightarrow{}$ S_i — max(t,mt)+t

$\xrightarrow{}$ C_{i+1} — max(t,mt)+2t

▪ Calculating:

When i=1, m=0; $S_1 = 2t$ and $C_2 = 3t$

When i=2, m=3; $S_2 = 4t$ and $C_3 = 5t$

When i=3, m=5; $S_3 = 6t$ and $C_4 = 7t$

When i=4, m=7; $S_4 = 8t$ and $C_5 = 9t$

▪ In general, an *n*-bit ripple-carry parallel adder will experience the following delay times:

$S_n = ((n-1)2 + 2) t$

$C_{n+1} = ((n-1)2 + 3) t$

▪ Propagation delay of ripple-carry parallel adders is proportional to the number of bits it handles.

▪ Maximum delay: $((n-1)2 + 3) t$

Quick Review Questions

- DLD pages 128 – 129
  Questions 6-1 to 6-4.

End of File