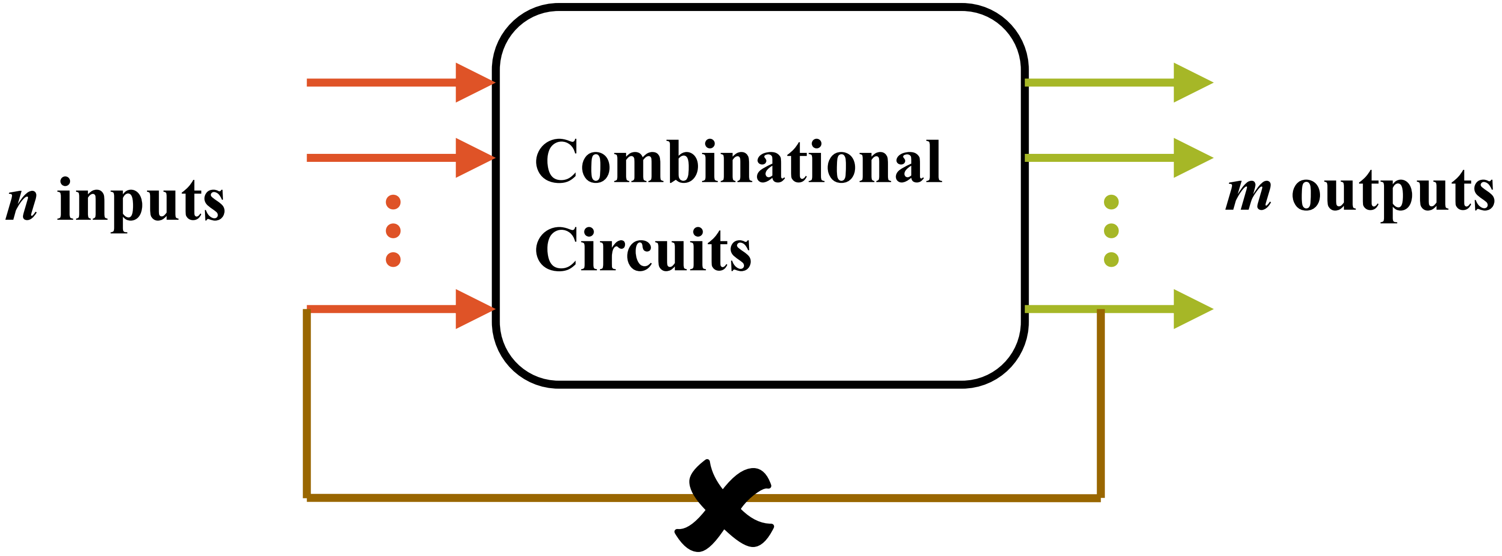
CHAPTER 4 – COMBINATIONAL LOGIC

Combinational Circuits

- Two classes of logic circuits:
  - Combinational Circuits
  - Sequential Circuits
- A **Combinational circuit** consists of logic gates
  - Output depends only on input
- A **Sequential circuit** consists of logic gates and memory
  - Output depends on current inputs and previous ones (stored in memory)
  - Memory defines the state of the circuit.

Combinational Circuits

- Output is function of input only i.e. no feedback



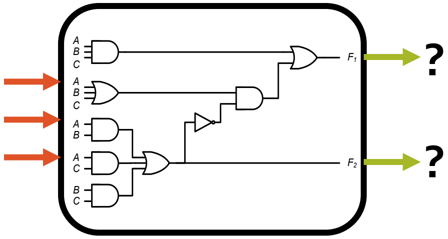
When input changes, output may change (after a delay)

Classification of Combinational Logic

- Arithmetic & logical functions (adder, subtractor, comparator)
- Data transmission(decoder, encoder, multiplexer, de-multiplexer)
- Code converters(BCD, grey code,7-segment)

Combinational Circuits

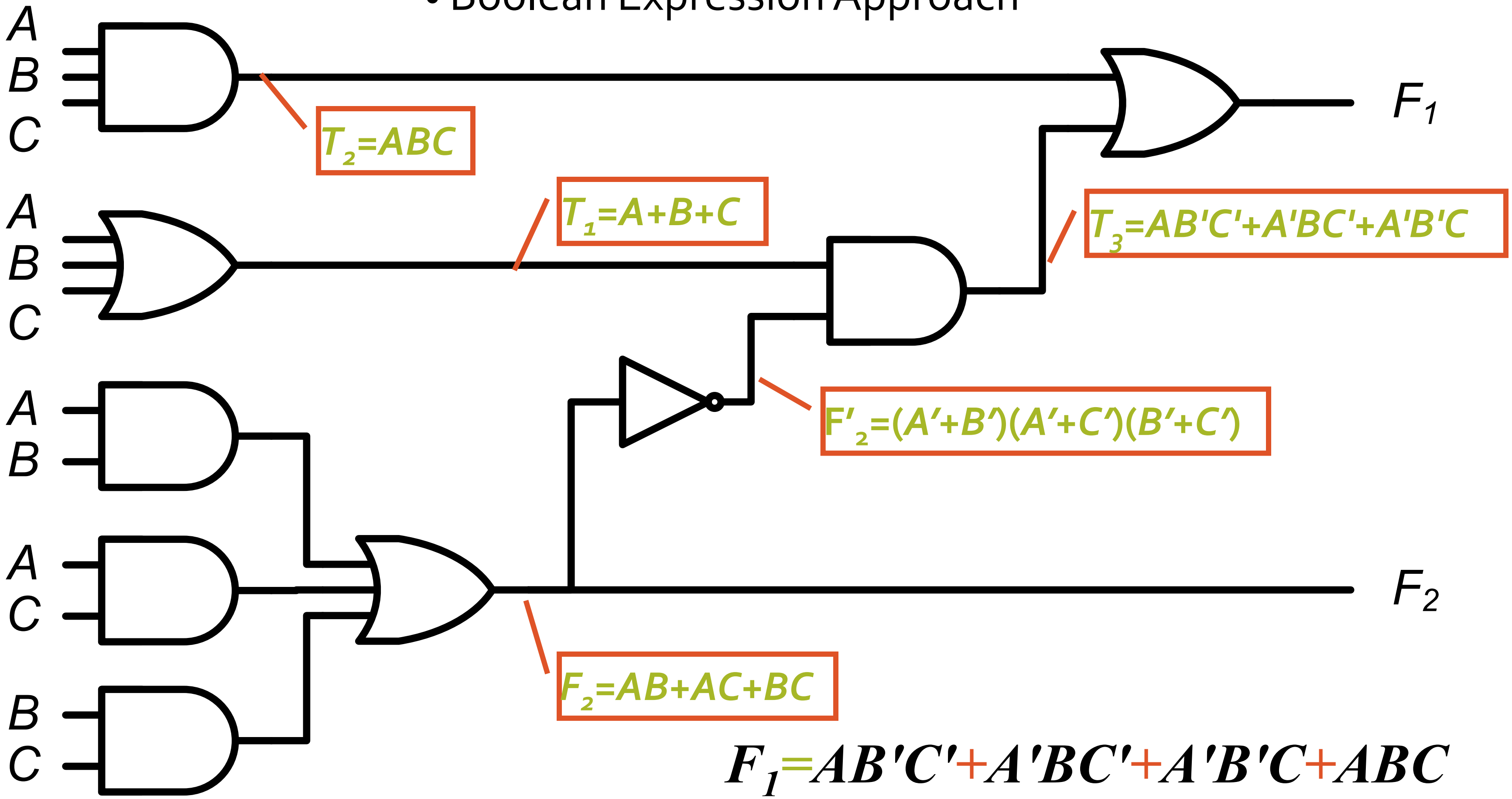
- Analysis
  - Given a circuit, find out its *function*
  - Function may be expressed as:
    - Boolean function
    - Truth table



- Design
  - Given a desired function, determine its *circuit*
  - Function may be expressed as:
    - Boolean function
    - Truth table

Analysis Procedure

- Boolean Expression Approach

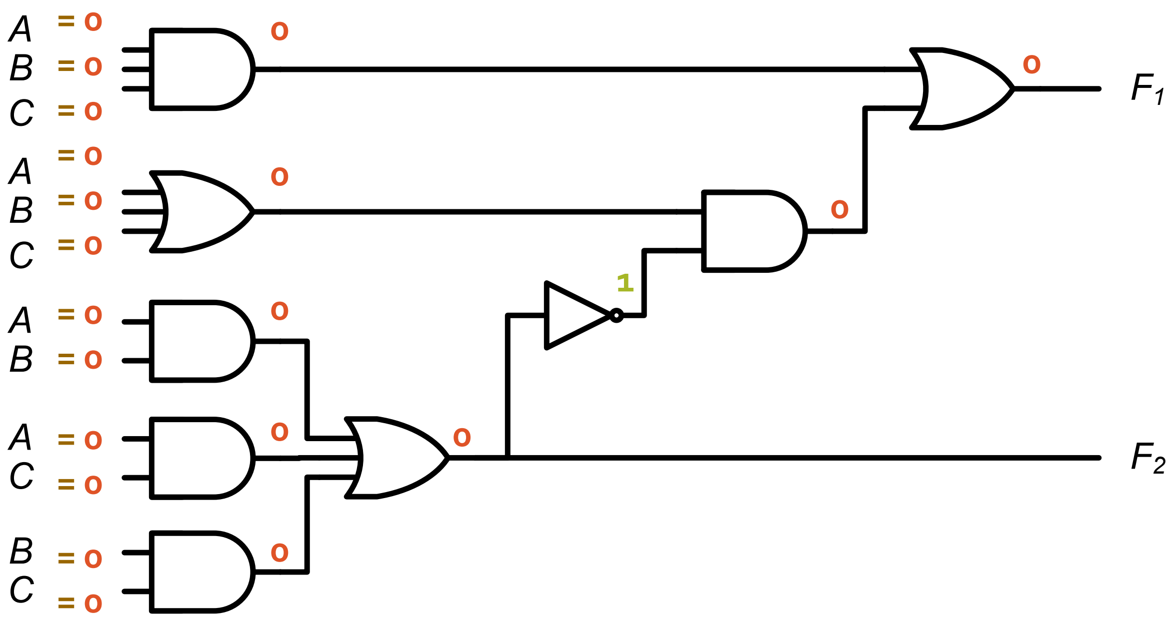


$F_1 = AB'C' + A'BC' + A'B'C + ABC$

$F_2 = AB + AC + BC$

Analysis Procedure

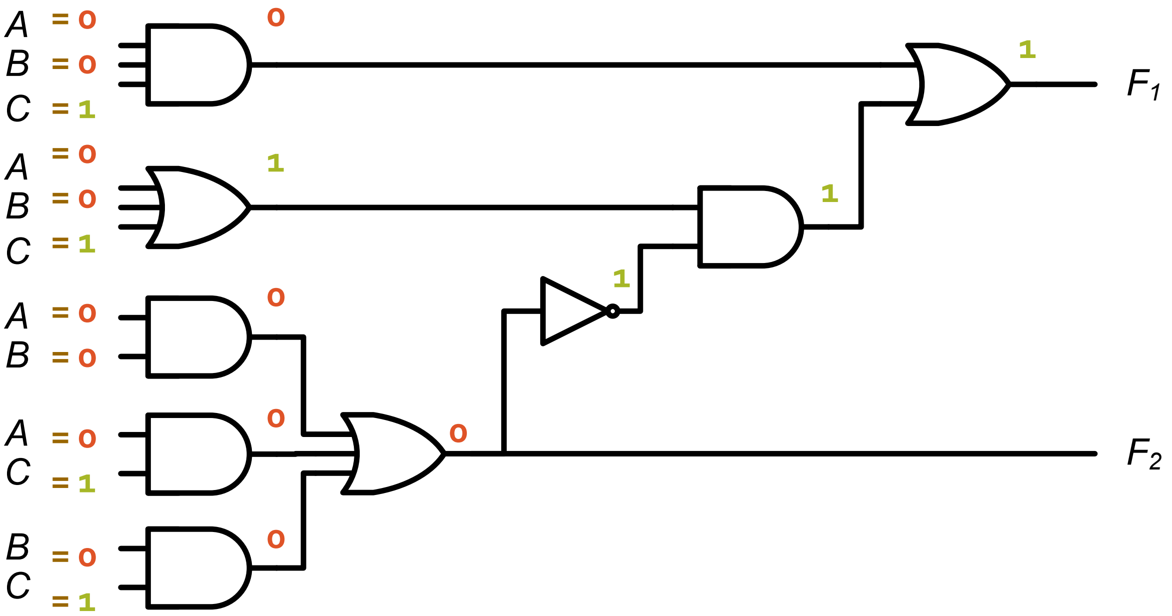
- Truth Table Approach



A B C	F_1	F_2
0 0 0	0	0

Analysis Procedure

- Truth Table Approach



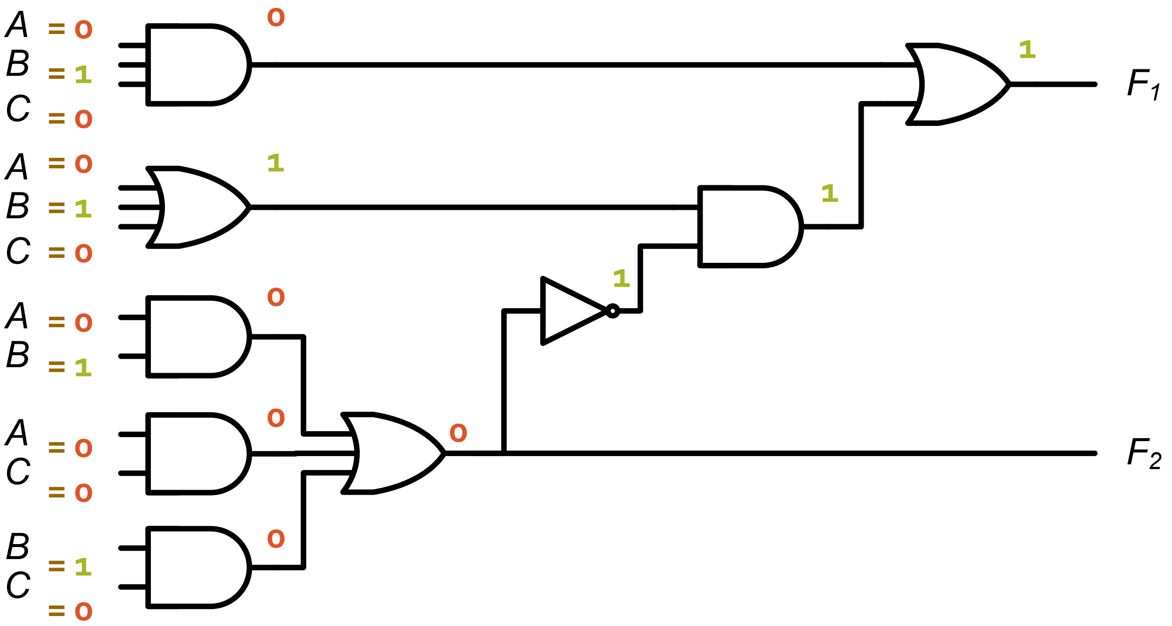
A B C	F_1	F_2
0 0 0	0	0
0 0 1	1	0

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

$$F_2 = AB + AC + BC$$

Analysis Procedure

- Truth Table Approach



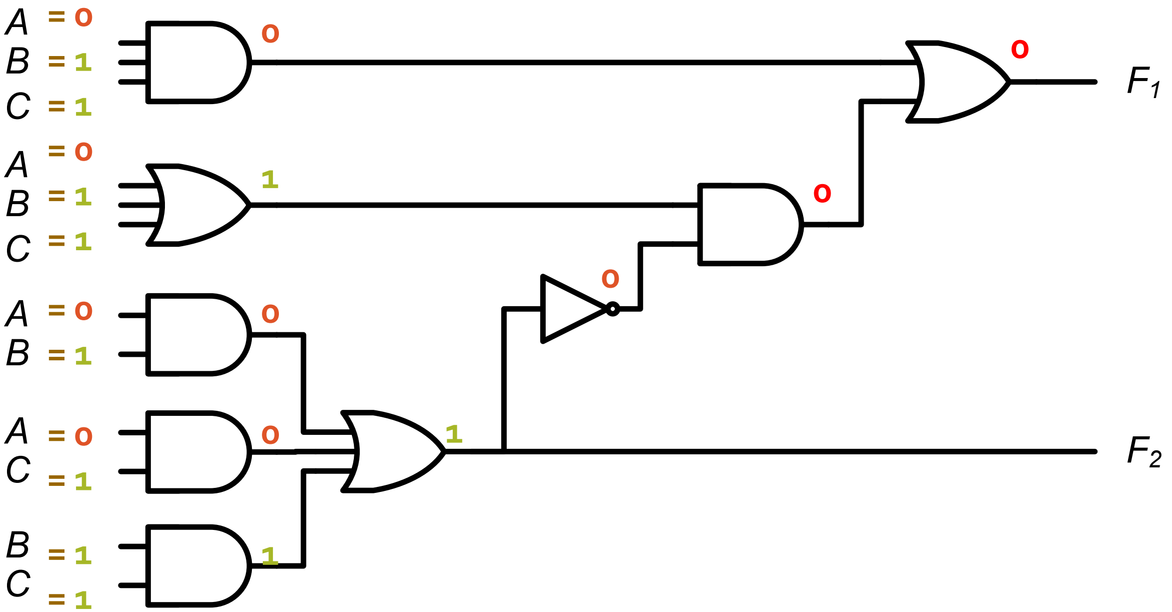
A	B	C	F_1	F_2
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0

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

$$F_2=AB+AC+BC$$

Analysis Procedure

- Truth Table Approach



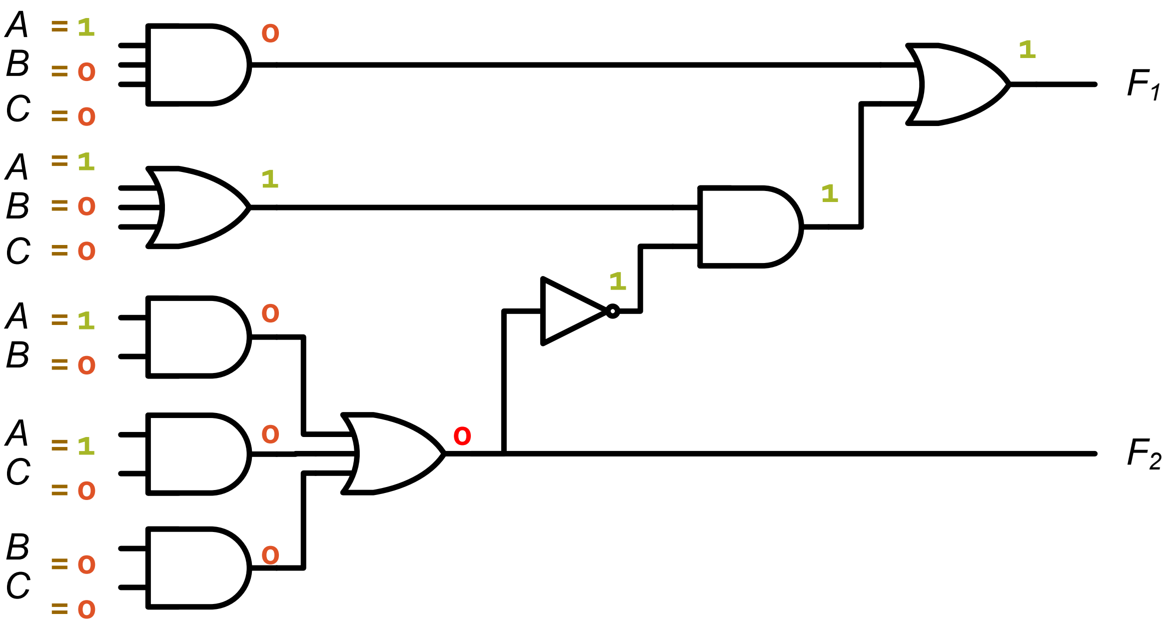
A B C	F_1	F_2
0 0 0	0	0
0 0 1	1	0
0 1 0	1	0
0 1 1	0	1

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

$$F_2 = AB + AC + BC$$

Analysis Procedure

- Truth Table Approach



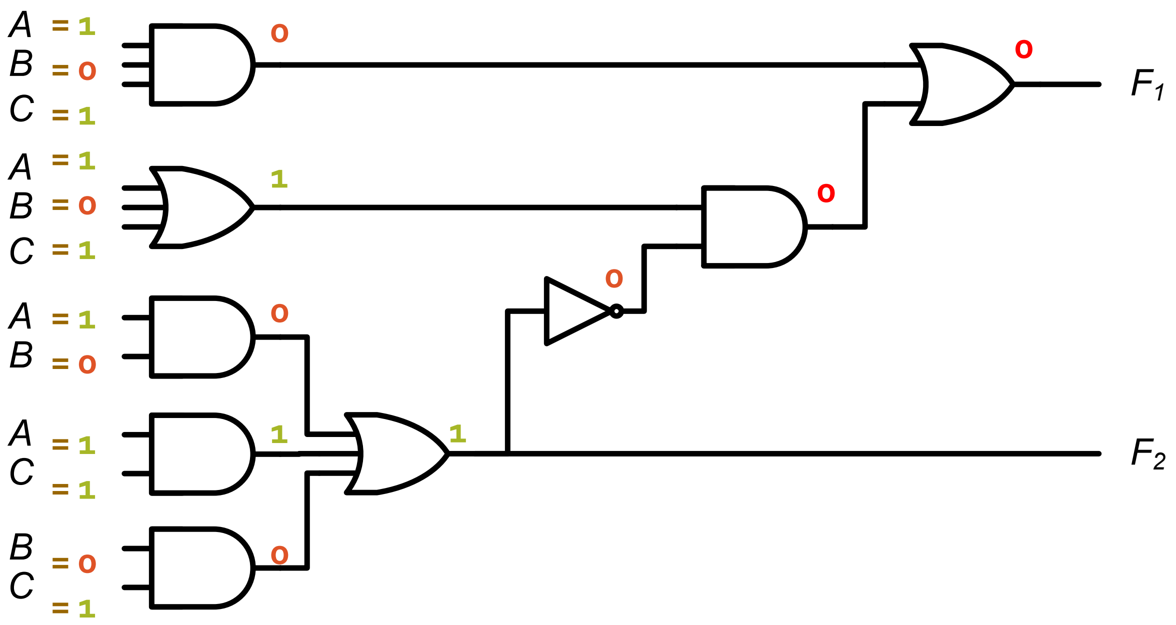
A B C	F_1	F_2
0 0 0	0	0
0 0 1	1	0
0 1 0	1	0
0 1 1	0	1
1 0 0	1	0

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

$$F_2 = AB + AC + BC$$

Analysis Procedure

- Truth Table Approach



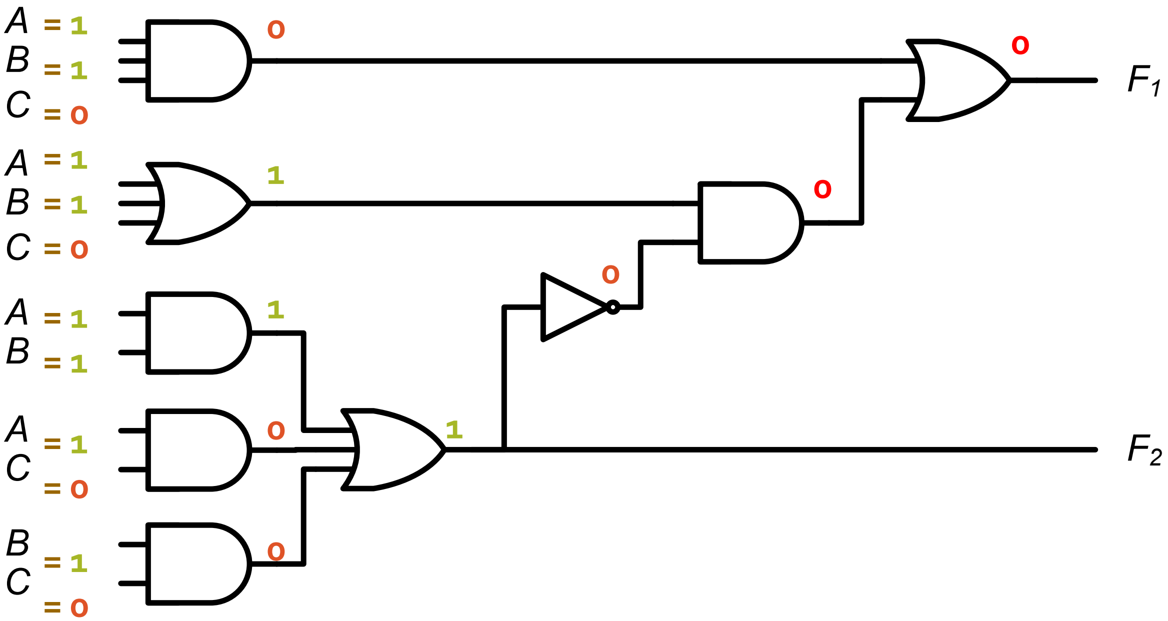
A B C	F_1	F_2
0 0 0	0	0
0 0 1	1	0
0 1 0	1	0
0 1 1	0	1
1 0 0	1	0
1 0 1	0	1

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

$$F_2 = AB + AC + BC$$

Analysis Procedure

- Truth Table Approach



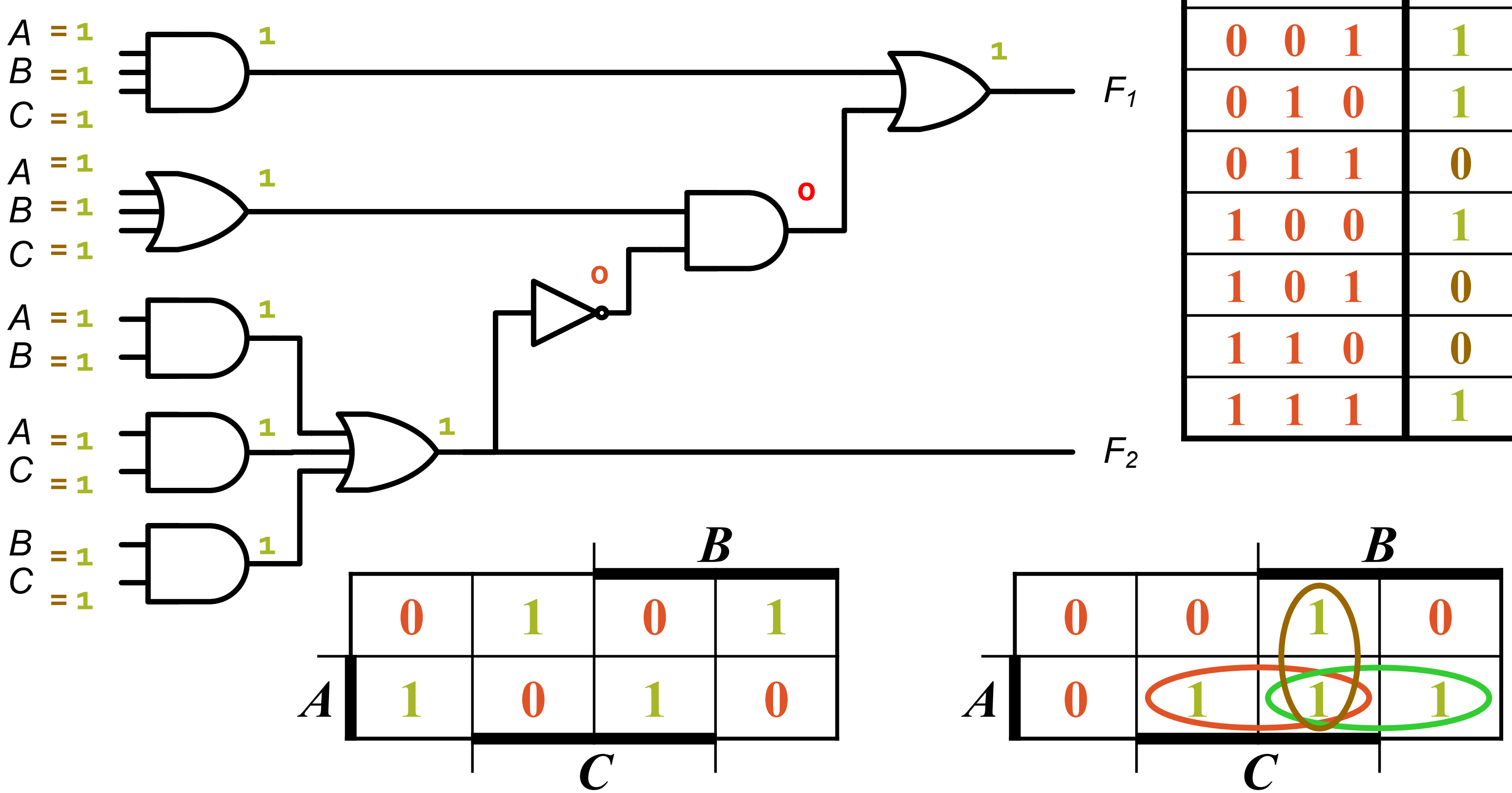
A B C	F₁	F₂
0 0 0	0	0
0 0 1	1	0
0 1 0	1	0
0 1 1	0	1
1 0 0	1	0
1 0 1	0	1
1 1 0	0	1

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

$$F_2 = AB + AC + BC$$

Analysis Procedure

- 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

	0	1	0	1
A	1	0	1	0

	0	0	1	0
A	0	1	1	1

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

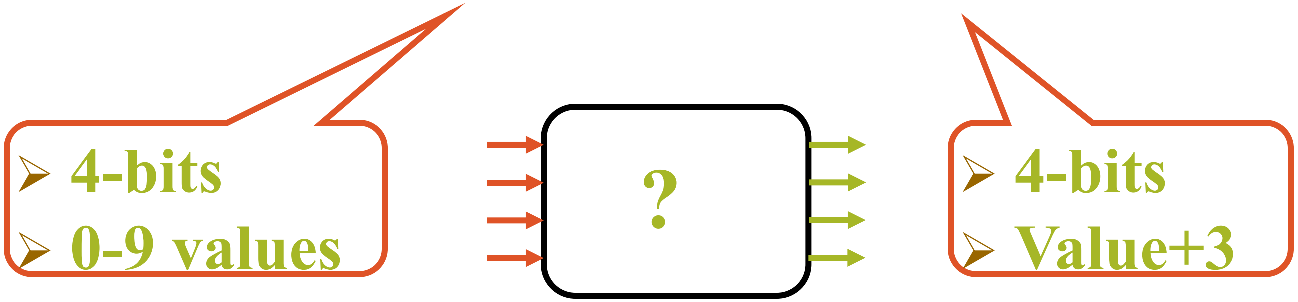
$$F_2 = AB + AC + BC$$

Design Procedure

- Given a problem statement:
  - Determine the number of *inputs* and *outputs*
  - Derive the truth table
  - Simplify the Boolean expression for each output
  - Produce the required circuit

Example:

Design a circuit to convert a “BCD” code to “Excess 3” code



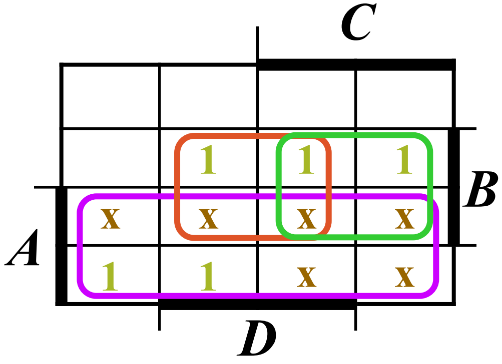
- **4-bits**
- **0-9 values**

- **4-bits**
- **Value+3**

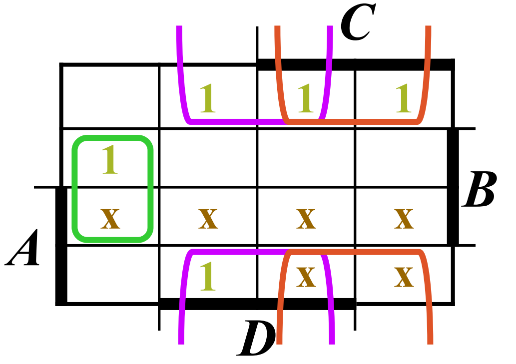
Design Procedure

- BCD-to-Excess 3 Converter

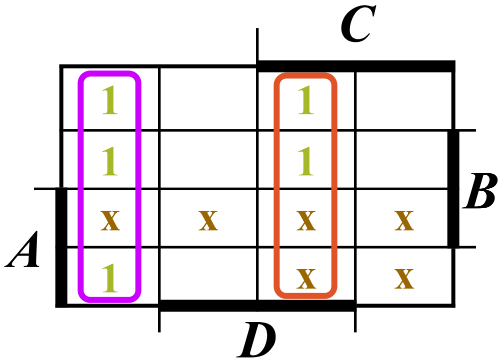
A	B	C	D	w	x	y	z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	x	x	x	x
1	0	1	1	x	x	x	x
1	1	0	0	x	x	x	x
1	1	0	1	x	x	x	x
1	1	1	0	x	x	x	x
1	1	1	1	x	x	x	x



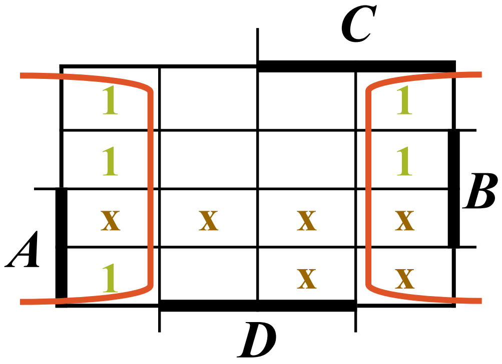
$w = A + BC + BD$



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



$y = C'D' + CD$

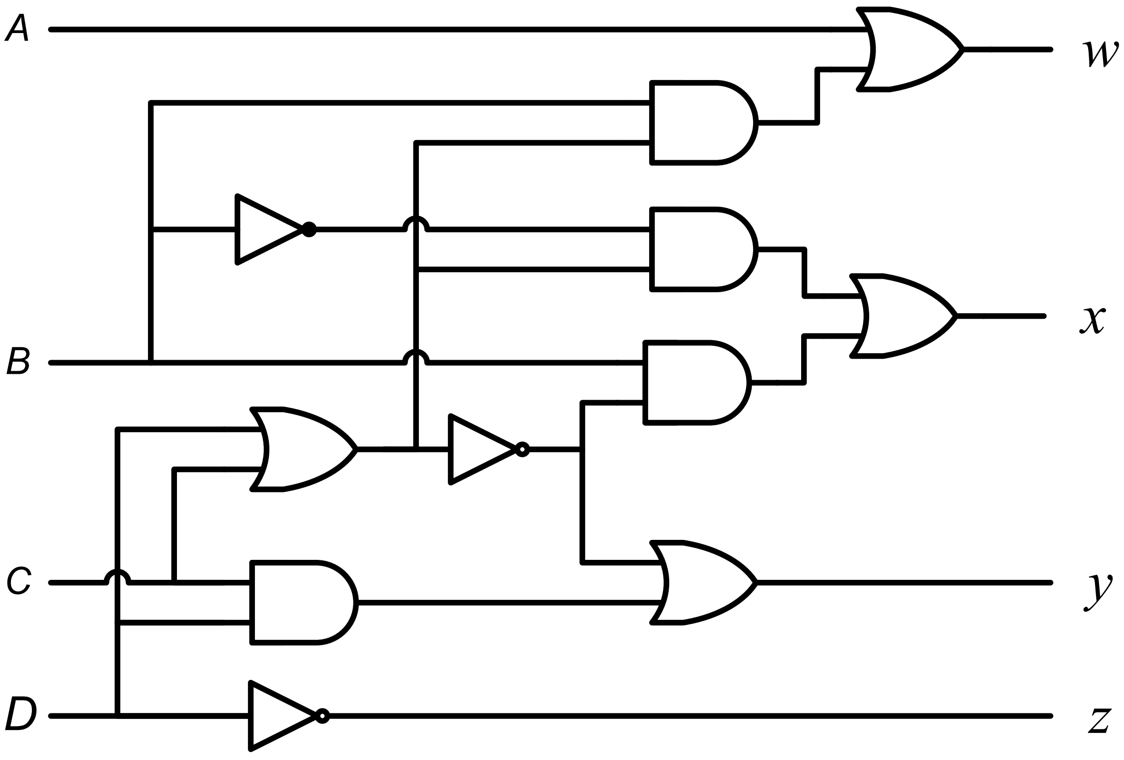


$z = D'$

Design Procedure

- BCD-to-Excess 3 Converter

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



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

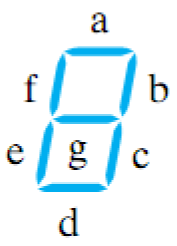
$x = B'(C+D) + B(C+D)'$

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

$z = D'$

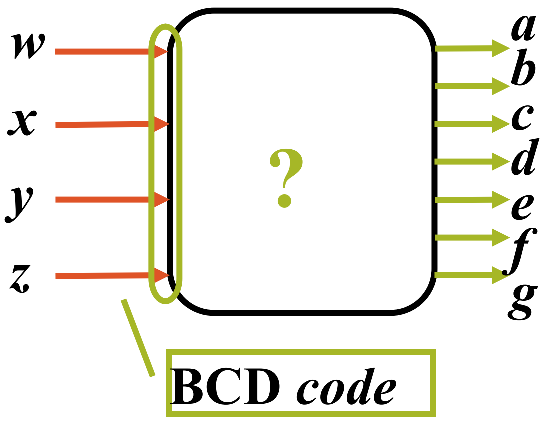
Seven-Segment converter

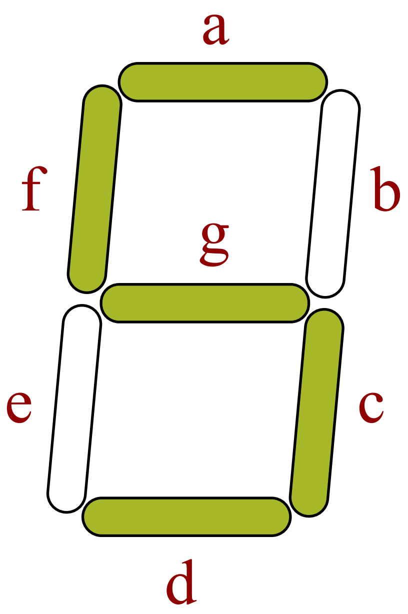
- Seven-Segment Display
- A seven-segment display is digital readout found in electronic devices like clocks, TVs, etc.
  - Made of seven light-emitting diodes (LED) segments; each segment is controlled separately.

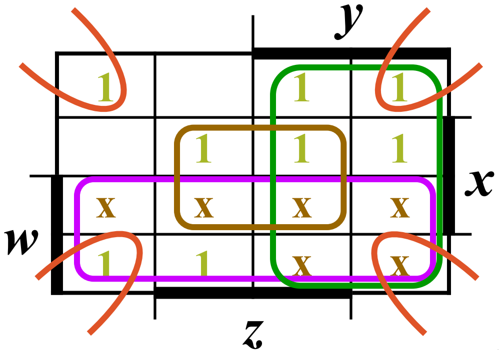


Seven-Segment converter

w x y z	*a b c d e f g*
0 0 0 0	1 1 1 1 1 1 0
0 0 0 1	0 1 1 0 0 0 0
0 0 1 0	1 1 0 1 1 0 1
0 0 1 1	1 1 1 1 0 0 1
0 1 0 0	0 1 1 0 0 1 1
0 1 0 1	1 0 1 1 0 1 1
0 1 1 0	1 0 1 1 1 1 1
0 1 1 1	1 1 1 0 0 0 0
1 0 0 0	1 1 1 1 1 1 1
1 0 0 1	1 1 1 1 0 1 1
1 0 1 0	x x x x x x x
1 0 1 1	x x x x x x x
1 1 0 0	x x x x x x x
1 1 0 1	x x x x x x x
1 1 1 0	x x x x x x x
1 1 1 1	x x x x x x x







1		1	1
	1	1	1
x	x	x	x
1	1	x	x

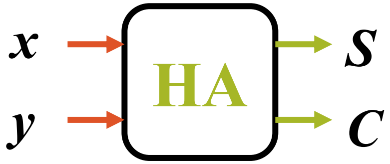
$a = w + y + xz + x'z'$

$b = \ldots$

$c = \ldots$

$d = \ldots$

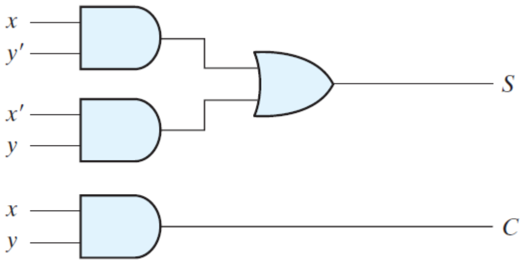
Binary Adder



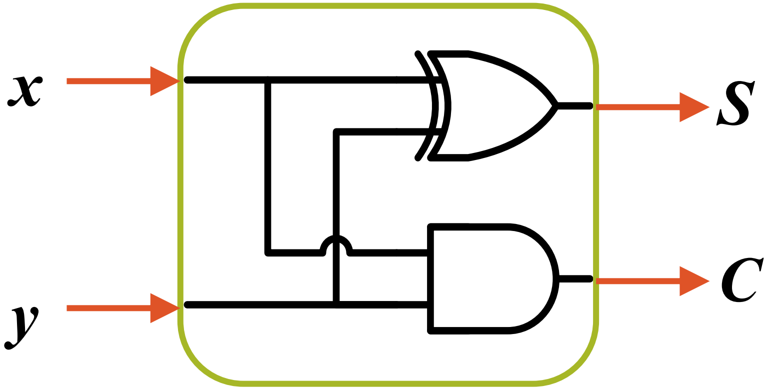
$$\begin{array}{r} x \\ +\quad y \\ \hline C\quad S \end{array}$$

- Half Adder
  - Adds 1-bit plus 1-bit
  - Produces Sum and Carry

x *y*	*C* *S*
0 0	0 0
0 1	0 1
1 0	0 1
1 1	1 0

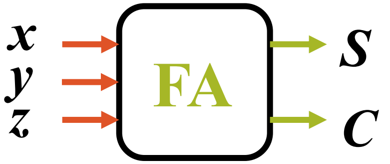


(a) $S = xy' + x'y$
$C = xy$



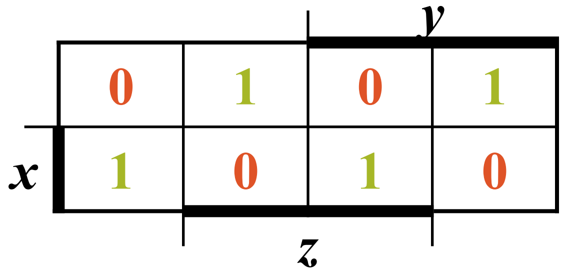
Binary Adder

- Full Adder
  - Adds 1-bit plus 1-bit plus 1-bit
  - Produces Sum and Carry



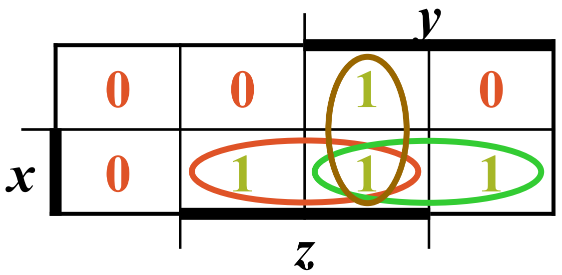
$$\begin{array}{rr} & x \\ + & y \\ + & z \\ \hline C & S \end{array}$$

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



0	1	0	1
1	0	1	0

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



0	0	1	0
0	1	1	1

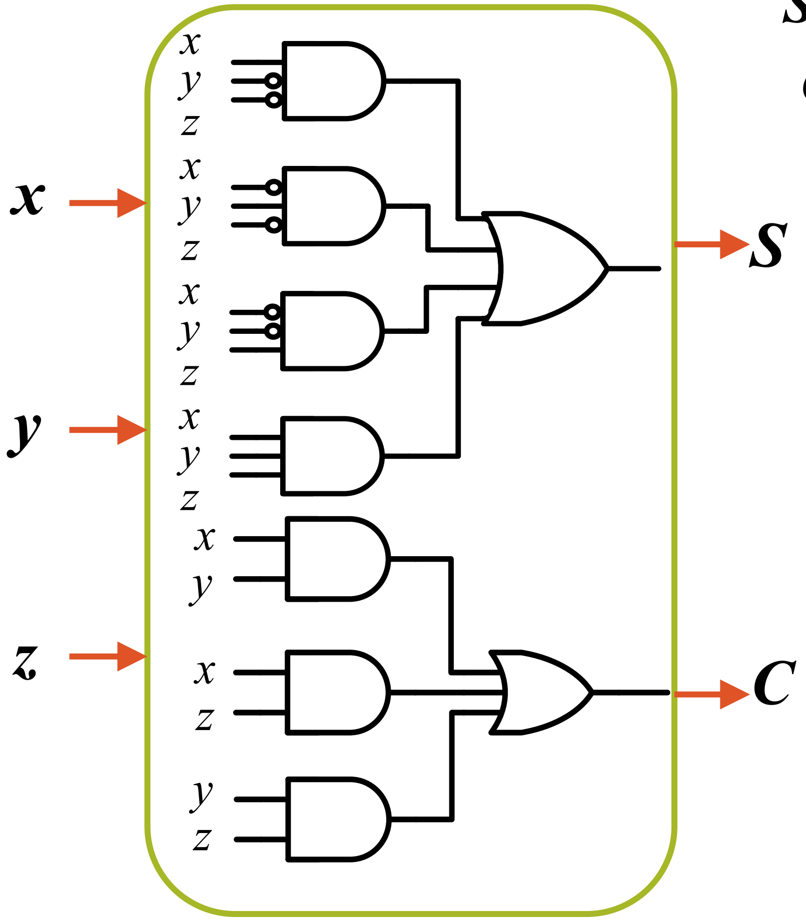
$$C = xy + xz + yz$$

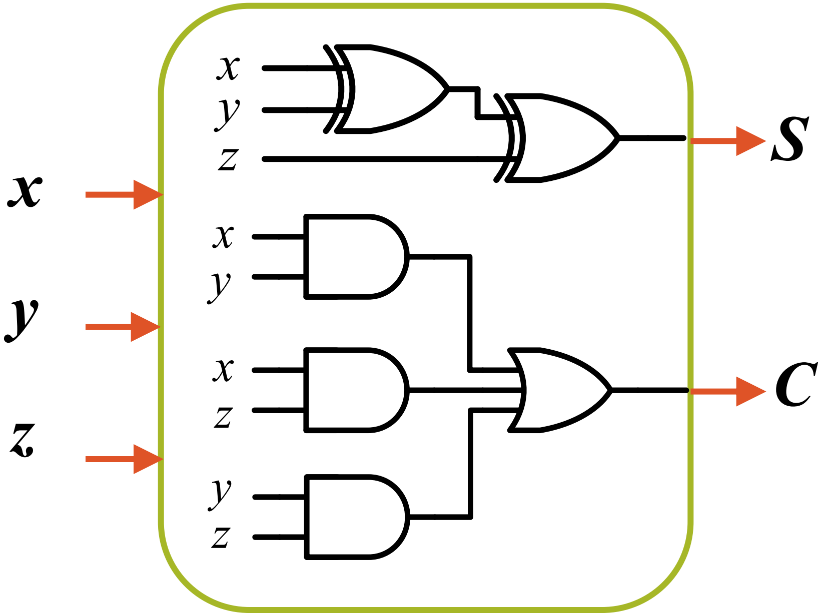
Binary Adder

- Full Adder

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

$$C = xy + xz + yz$$



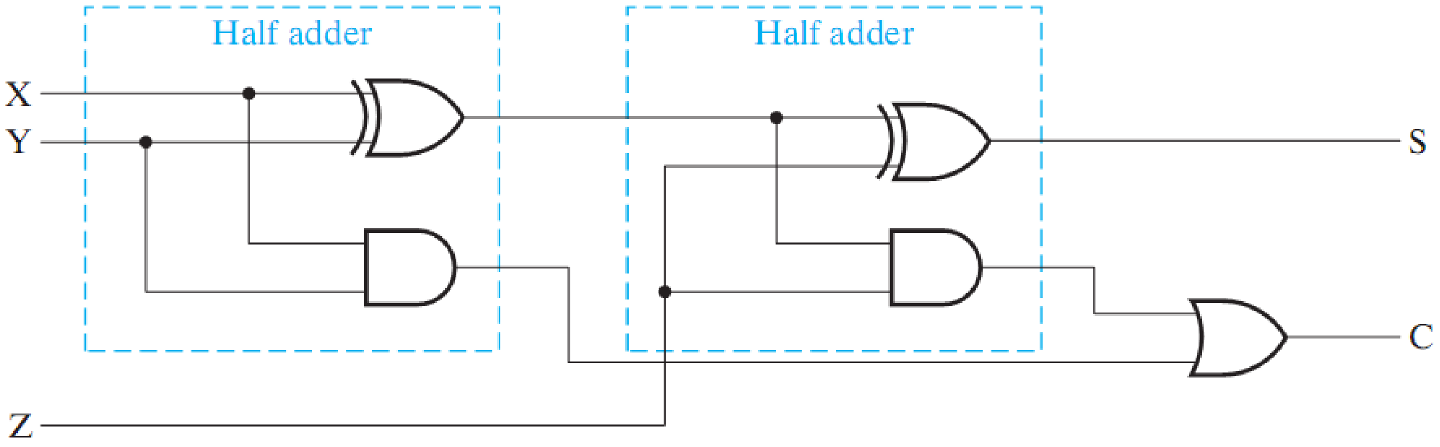


Full Adder = 2 Half Adders

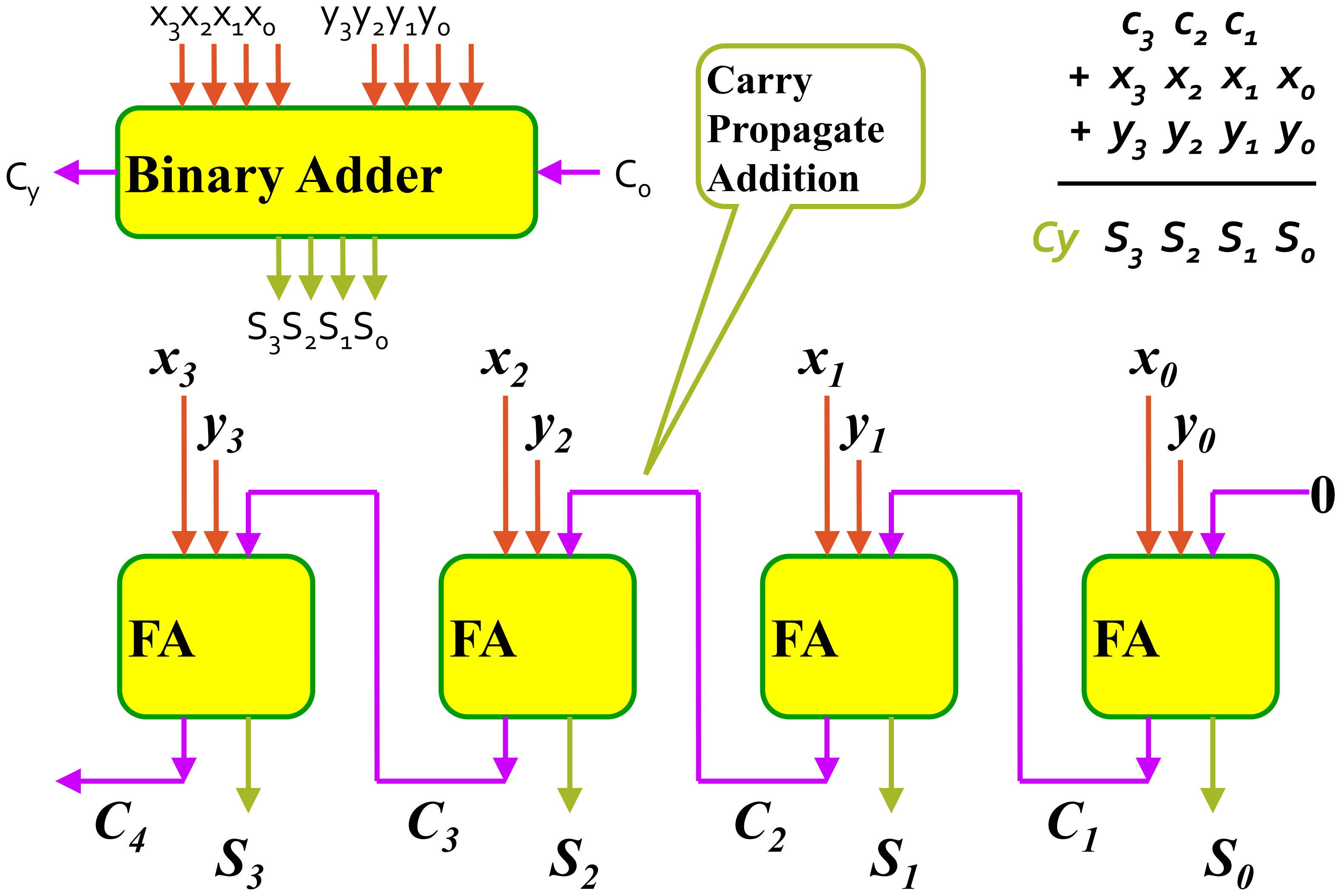
Manipulating the Equations:

$S = (X \oplus Y) \oplus Z$

$C = XY + XZ + YZ = XY + Z(X \oplus Y)$



Binary Adder



Carry Propagate Addition

$$\begin{array}{rcccc} & c_3 & c_2 & c_1 & \\ + & x_3 & x_2 & x_1 & x_0 \\ + & y_3 & y_2 & y_1 & y_0 \\ \hline Cy & S_3 & S_2 & S_1 & S_0 \end{array}$$

Bigger Adders

- How to build an adder for n-bit numbers?
  - Example: 4-Bit Adder
    - Inputs ?
    - Outputs ?
    - What is the size of the truth table?
    - How many functions to optimize?

Bigger Adders

- How to build an adder for n-bit numbers?
  - Example: 4-Bit Adder
    - Inputs ? 9 inputs
    - Outputs ? 5 outputs
    - What is the size of the truth table? 512 rows!
    - How many functions to optimize? 5 functions

Binary Adder

1 0 0 0 ← Carry in

$$\begin{array}{r} 0101 \\ +\ \underline{0110} \\ 1011 \end{array}$$

- To add n-bit numbers:
- Use n Full-Adders in Cascade.
- The carries propagates as in addition by hand.

This adder is called ***ripple carry adder***

Binary Subtractor

- Half Subtractor

  A logic circuit which is used for subtracting one single bit binary number from another single bit binary number is called half subtractor.

Binary Subtractor

- Half Subtractor

S.No	INPUT		OUTPUT	
	A	B	DIFF	BORR
1.	0	0	0	0
2.	0	1	1	1
3.	1	0	1	0
4.	1	1	0	0

Binary Subtractor

- Full Subtractor

  The Full subtractor is a combinational circuit which is used to perform subtraction of three bits.

Binary Subtractor

- Full Subtractor

S.No	INPUT			OUTPUT	
	A	B	C	DIFF	BORR
1.	0	0	0	0	0
2.	0	0	1	1	1
3.	0	1	0	1	1
4.	0	1	1	0	1
5.	1	0	0	1	0
6.	1	0	1	0	0
7.	1	1	0	0	0
8.	1	1	1	1	1

Binary Subtractor

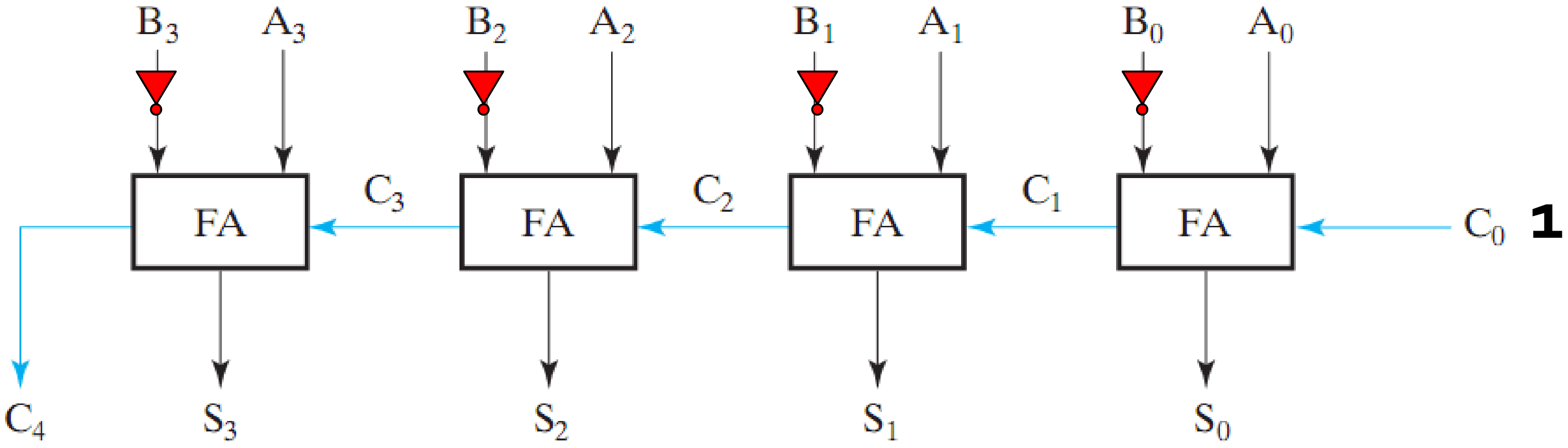
- Derive simplified Boolean expressions for Half Subtractor and Full Subtractor.

Subtraction (2’s Complement)

- How to build a subtractor using 2’s complement?

Subtraction (2's Complement)

- How to build a subtractor using 2's complement?

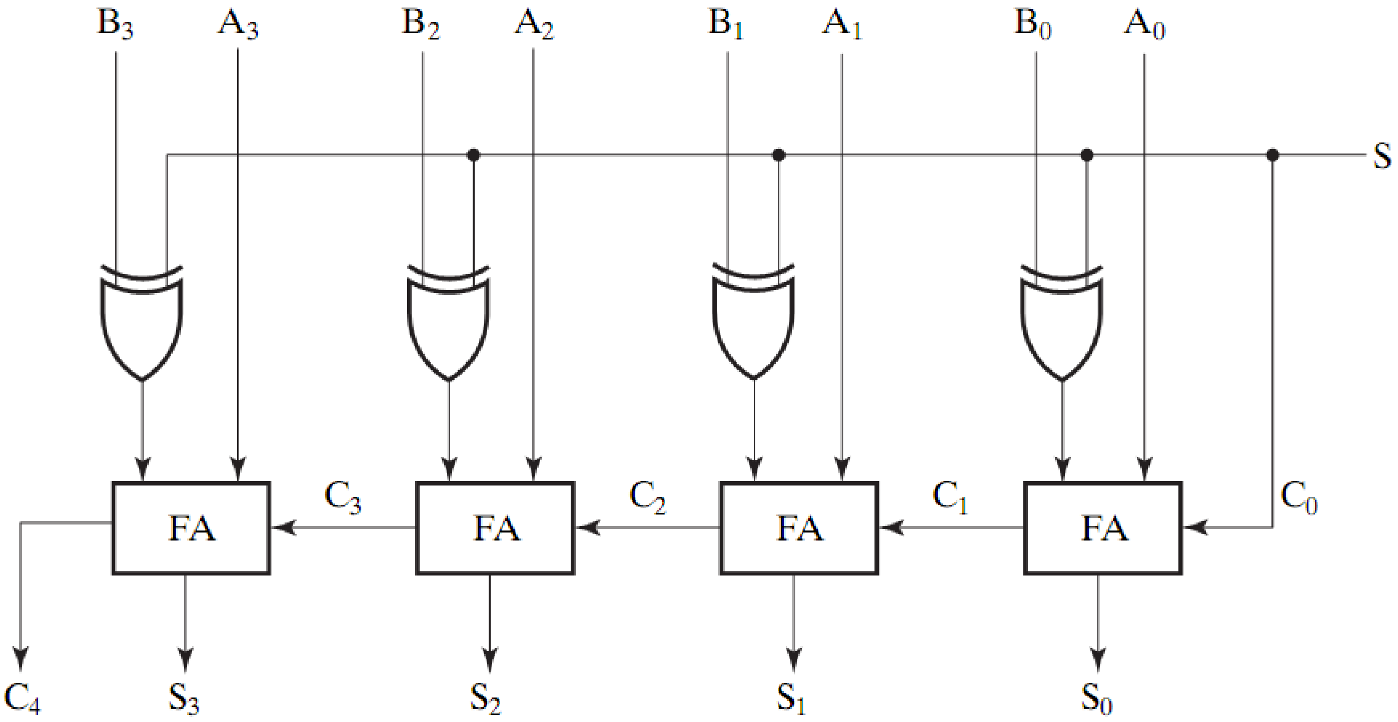


$S = A + (-B)$

Adder/Subtractor

- How to build a circuit that performs both addition and subtraction?

Adder/Subtractor



0 : Add
1: subtract

Using full adders and XOR we can build an Adder/Subtractor!