Introduction to Computing

Lectured by: Dr. Pham Tran Vu

t.v.pham@cse.hcmut.edu.vn

Assignment

- Research on the current issues in computing
- Assessment:
  - Report: 30%
  - Presentation: 10%
- Deadline: 29/3/2010

Assignment Topics

- Web search engines: history and development
- Online games: benefits and social issues
- Software licensing and opportunities for open source software
- Internet in Vietnam: development history and its social impacts

Lecture 2: Fundamental Concepts (cont’)

History of computer

Number systems

Data representation

Computer logic

Data Representation

- Data processed by computers has to be in binary form
- Main memory and external storage media, e.g. magnetic disk and tape, use electrical/magnetic patterns representing binary digits to record and handle data & instructions

Character & Numeric Codes

- Character codes used to represent data processed by computers and stored data
- Numeric codes used to represent numeric data for processing purposes
- Characters may be:
  - Alphabetic (upper and lower case)
  - Numeric
  - Special characters (apostrophe, comma, etc)
  - Control characters and codes

ASCII Character Set

- The range of characters which can be represented by a computer system is know as character set
- ASCII – American Standard Code for Information Interchange
- A character is represented by 7 binary digits
  - Total of 128 characters in ASCII character set
- A additional bit, known as parity-bit, in left most position, is used to detect single bit error during data transfer

Examples of ASCII Characters

Char	ASCII	Char	ASCII	Char	ASCII	Char	ASCII
0	0110000	9	0111001	I	1001001	R	1010010
1	0110001	A	1000001	J	1001010	S	1010011
2	0110010	B	1000010	K	1001011	T	1010100
3	0110011	C	1000011	L	1001100	U	1010101
4	0110100	D	1000100	M	1001101	V	1010110
5	0110101	E	1000101	N	1001110	W	1010111
6	0110110	F	1000110	O	1001111	X	1011000
7	0110111	G	1000111	P	1010000	Y	1011001
8	0111000	H	1001000	Q	1010001	Z	1011010

Structure of Main Memory (1)

- Main memory is divided into locations, each of which has a unique address
- Each location (an addressable unit) contains a memory word
- A memory word is a group of bits in memory, representing data or an instruction
- Memory word’s length is the number of bits can be stored at one location
- Word’s length can be different, depending on computer architecture (4, 8, 16, 32 or 64 bits)

Structure of Main Memory (2)

- Large words may be composed of smaller units called byte, which is 8-bit length
- Example: structure of 16-bit word

High order byte								Low order byte							
MSB															LSB
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Internal Numbers

- Numbers are represented by bits
- An n-bit number has range from $0..2^n - 1$
- Examples
  - 1-bit: 2 values 0 and 1
  - 1 byte: from 0 to 2^8 - 1(255)
  - 2 bytes: 0 to 2^{16} -1 (65535)

Representation of Signed Integers

- Sign-magnitude
  - Use the MSB as a sign bit
- One's complement
  - The inverse of a number formed by complementing each bit (0->1 and 1->0)
- Two's complement
  - One's complement of a number add 1

Sign and Magnitude

- Used in early computers
- Sign and magnitude of 8-bit number
- Range: - 127_{10} -> $+127_{10}$

MSB							LSB
7	6	5	4	3	2	1	0
Sign	Magnitude						

8 bit signed magnitude

Binary value	Signed magnitude interpretation	Unsigned interpretation
00000000	0	0
00000001	1	1
...	...	...
01111111	127	127
10000000	−0	128
...	...	...
11111111	−127	255

One's Complement

- Have two representations of 0:
  - +0: 00000000
  - -0: 11111111
- An 8-bit byte has value ranging from -127_{10} to 127_{10}

```
     binary      decimal
    11111110      -1
 +  00000010      +2
............     ...
  1 00000000       0    <-- not the correct answer
           1      +1    <-- add carry
............     ...
    00000001       1    <-- correct answer
```

8 bit ones' complement

Binary value	Ones' complement interpretation	Unsigned interpretation
00000000	0	0
00000001	1	1
...	...	...
01111101	125	125
01111110	126	126
01111111	127	127
10000000	−127	128
10000001	−126	129
10000010	−125	130
...	...	...
11111110	−1	254
11111111	−0	255

Two's Complement

- N-bit two's complement number in the range: -2^{N-1} to $2^{N-1}-1$
- 8-bit number ranging from -128 to 127

	MSB	Place value						LSB
	-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
+33	0	0	1	0	0	0	0	1
-33	1	1	0	1	1	1	1	1
bit	7	6	5	4	3	2	1	0

8 bit two's complement

Binary value	Two's complement interpretation	Unsigned interpretation
00000000	0	0
00000001	1	1
...	...	...
01111110	126	126
01111111	127	127
10000000	−128	128
10000001	−127	129
10000010	−126	130
...	...	...
11111110	−2	254
11111111	−1	255

Arithmetic Operations: Addition

- No need for special processing

```
   11111 111     (carry)
    0000 1111    (15)
  + 1111 1011    (-5)
  ==================
    0000 1010    (10)
```

Arithmetic Operations: Subtraction

- Direct subtraction can be used

```
 11110 000     (borrow)      11100 000     (borrow)
  0000 1111    (15)           0000 1111    (15)
- 1111 1011    (-5)         - 0010 0011    (35)
===========                 ===========
  0001 0100    (20)           1110 1100    (-20)
```

- Or negate the subtrahend and perform addition

Arithmetic Overflow

- Overflow happens when result of an arithmetic operation is larger than the range permitted by a word
- Can be detected by comparing the two right most carry bits

```
 0111      (carry)              11111 111    (carry)
  0111     (7)                    0000 1111  (15)
+ 0011     (3)                  + 1111 1011  (-5)
==============                  ===================
  1010     (-6)   invalid!        0000 1010  (10)
```

Real Numbers

- Computers also need to handle real numbers
- Two methods can be used:
  - Fixed-point representation
  - Floating-point representation

Fixed-point Representation

- Fixed-point numbers use conventional formats

Integer part	.	Fractional part

- The binary point can be placed any position within a memory word by the programmer

	Integer part	.	Fractional part
2.75_{10}	000010	.	11_2
28.25_{10}	011100	.	01_2

- Not commonly used

Floating-point Representation

- Represented in the form: $m \times r^e$
  - m: mantissa, can be positive or negative
  - r: radix or base
  - e: exponent, can be positive or negative
- Examples:
  - Denary: 6.8×10^6, 5.64×10^{-5}
  - Binary: 0.1010101×2^3, 0.11001×2^{-2}

Storage of Floating Point Numbers

- The length of mantissa determines the precision of a number
- The exponent determines the range, the length usually one-third or one-half of the mantissa
- The binary point is immediately to the right of the sign bit

	sign	Mantissa (fraction)										Exponent (int)			
bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Positive and Negative Floating-point Forms – using Two's Complement

- Positive form: the most significant digit to the right of binary point is 1, the sign bit is 0
- Negative form: the most significant digit to the right of binary point is 0, the sign bit is 1
- If the most significant digit and the sign-bit is the same, the number needs to be normalised

Positive floating form	
12 bits	4 bits
0.1**********	****
mantissa	exponent

Negative floating form	
12 bits	4 bits
1.0**********	****
mantissa	exponent

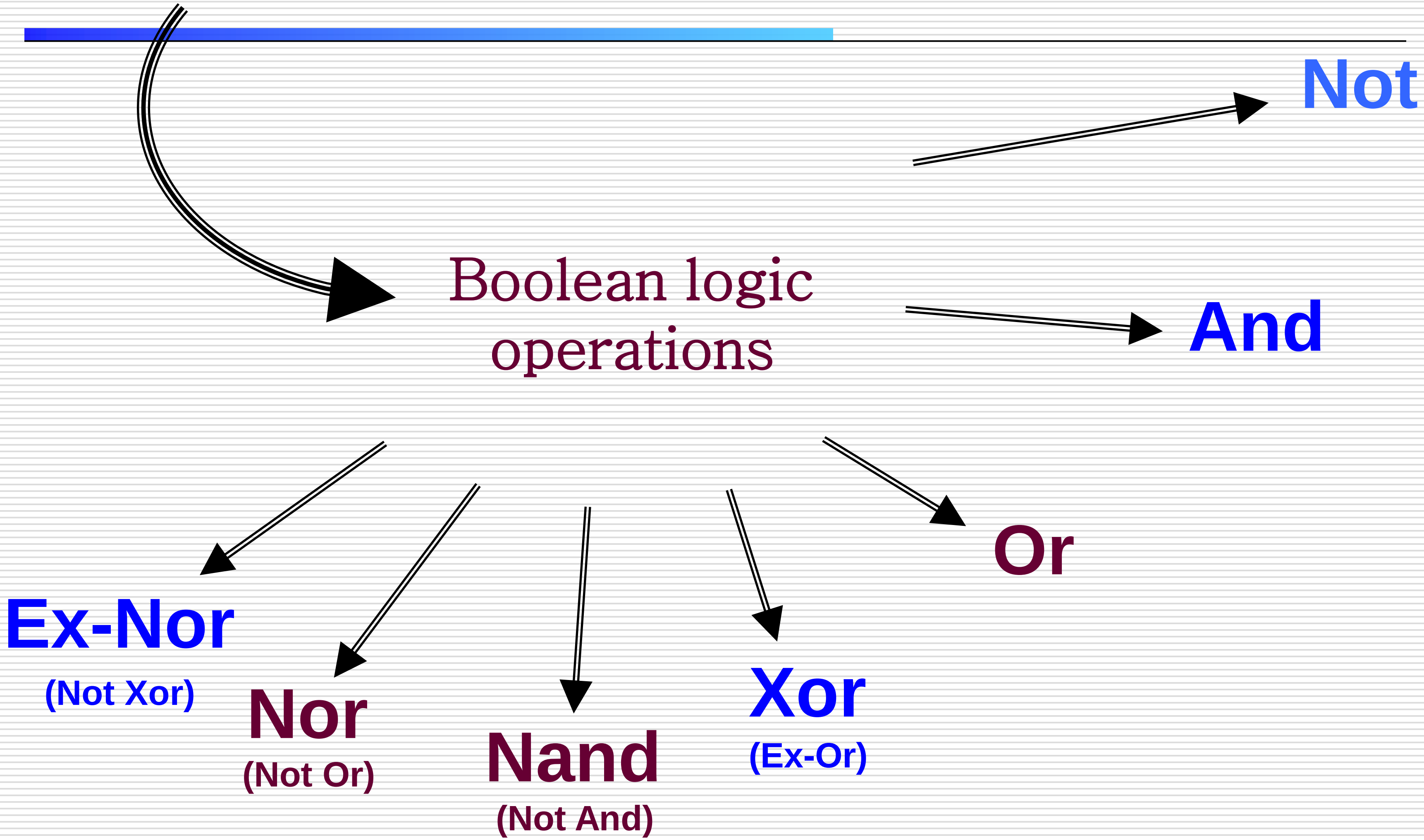
Double Precision Numbers

- Using two contiguous memory words for storing a number to increase precision

Computer Logic

- Boolean variables
  - Have two values: 0 or 1
- Boolean operations

Boolean Operations



Not Operation

Symbol

Truth table

x	$\overline{x}$
0	1
1	0

$$x = 1011 \Rightarrow \overline{x} = 0100$$

$$\Rightarrow \overline{\overline{x}} = 1011 = x$$

And Operation

Use dot symbol as in multiplication

Truth table

x	y	x.y
0	0	0
0	1	0
1	0	0
1	1	1

y . 0 = 0

y . 1 = y

Or Operation

Use + symbol as in addition

Truth table

x	y	x + y
0	0	0
0	1	1
1	0	1
1	1	1

y + 0 = y

y + 1 = 1

XOR (Exclusive OR) Operation

Use ⊕ symbol

Truth table

x	y	x ⊕ y
0	0	0
0	1	1
1	0	1
1	1	0

$$\mathbf{y \oplus 0 = y}$$

$$\mathbf{y \oplus 1 = \overline{y}}$$

Summary

Truth table

		NOT	AND	OR	XOR
x	y	not y	x and y	x or y	x xor y
0	0	1	0	0	0
0	1	0	0	1	1
1	0	1	0	1	1
1	1	0	1	1	0

y and 0 = 0
y and 1 = y

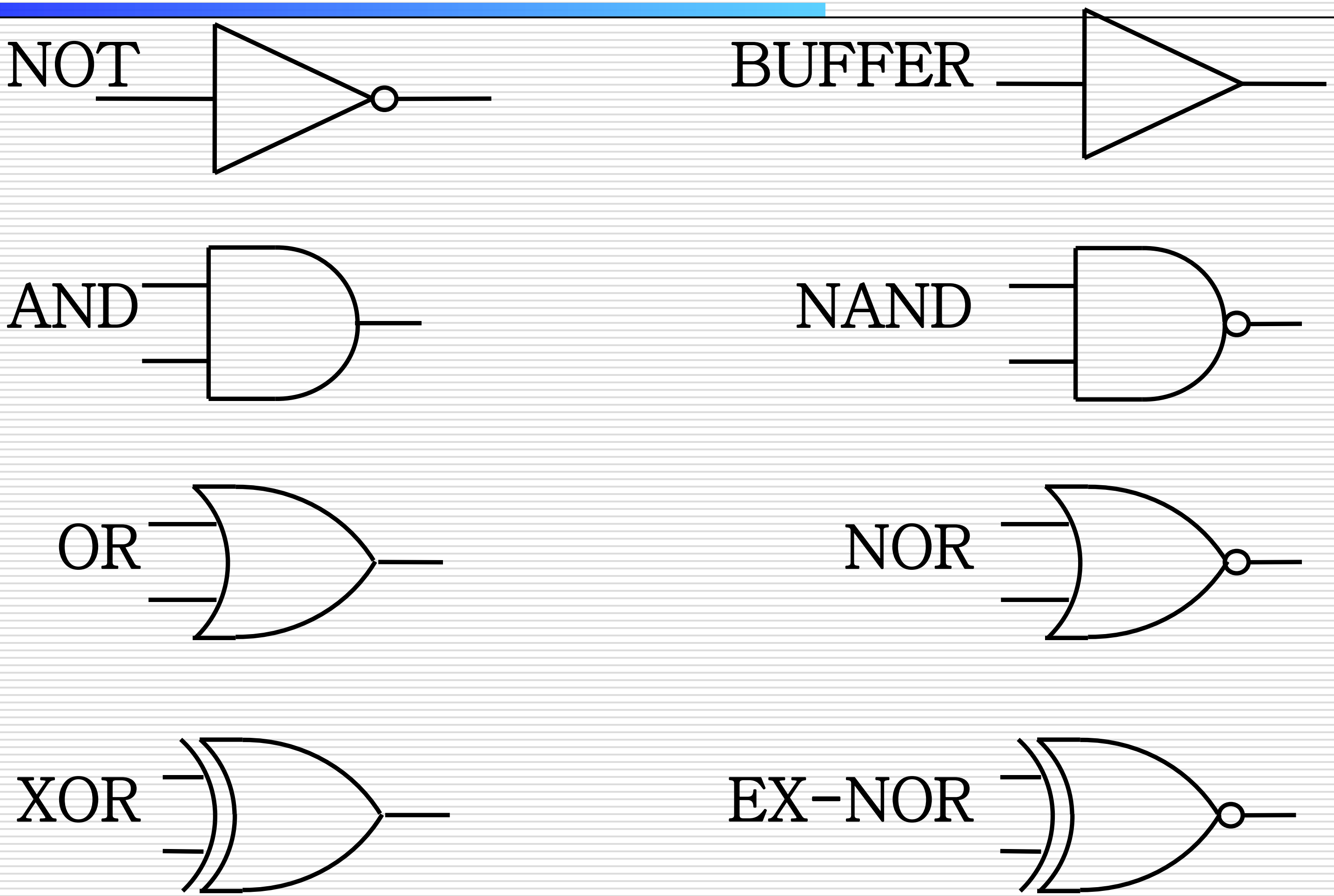
y or 0 = y
y or 1 = 1

y xor 0 = y
y xor 1 = not y

Laws of Boolean Algebra

- A Boolean expression
  - $A = X.Y.Z + X.\overline{Y}.Z + X.Y.\overline{Z}$
- Laws:
  - $X + Y = Y + X$; $X.Y = Y.X$
  - $X + (Y+Z) = (X + Y) + Z$; $X.(Y.Z) = (X.Y).Z$
  - $X.(Y+Z) = X.Y + X.Z$; $X + Y.Z = (X+Y).(X+Z)$
  - $\overline{(X+Y)}=\overline{X}.\overline{Y}$; $\overline{X.Y} = \overline{X} + \overline{Y}$
  - $X + X.Y = X$; $X.(X+Y) = Y$
  - $X + X = X$; $X.X = X$
  - $\overline{\overline{X}} = X$

Gates (1)

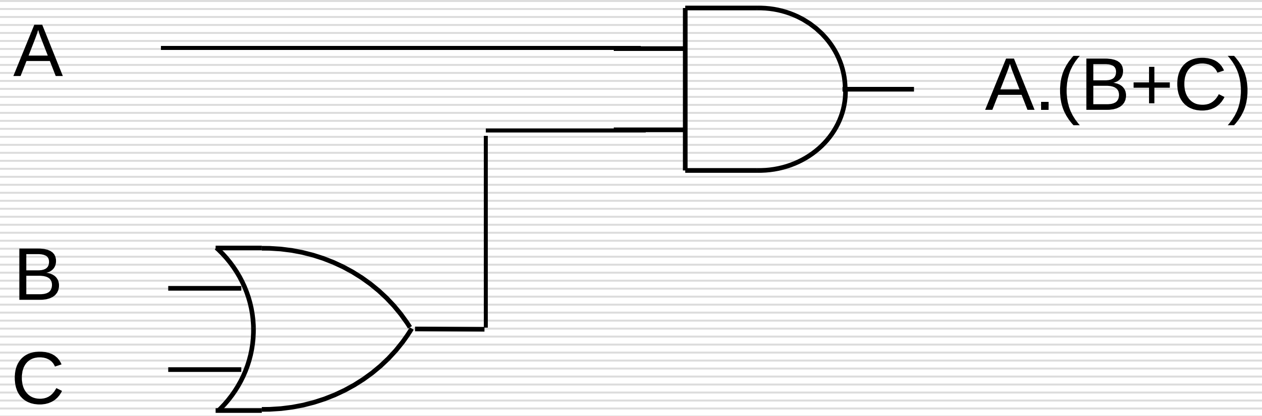


Gates (2)

- Gates are basic electronic components can be used to perform logical and arithmetic operations
- A combination of gates can be used for complex operations
- A logic circuit is a combination of gates

Circuit Logic Using Gates

- Logic circuits can be built from gates based directly on Boolean expressions



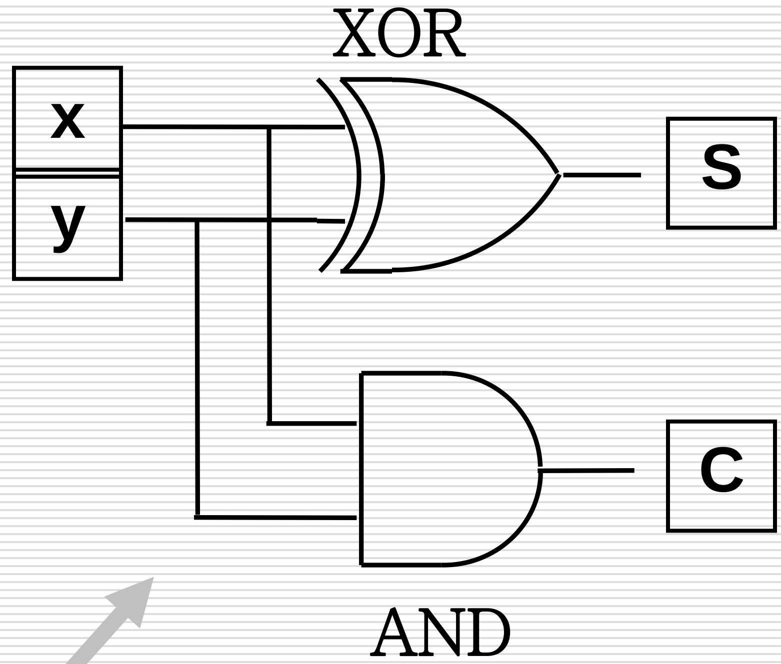
An Application of Logic Gates

- Half adder circuit: perform addition operation for 2 binary digits
- Full adder circuit can add 3 binary digits
- Two numbers of larger numbers of digits can be added by using a combination of full adder circuits

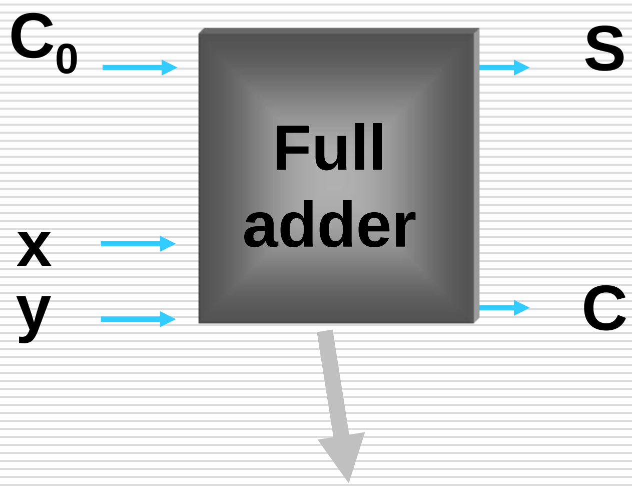
Half Adder Circuit

XOR AND

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



Full Adder Circuit



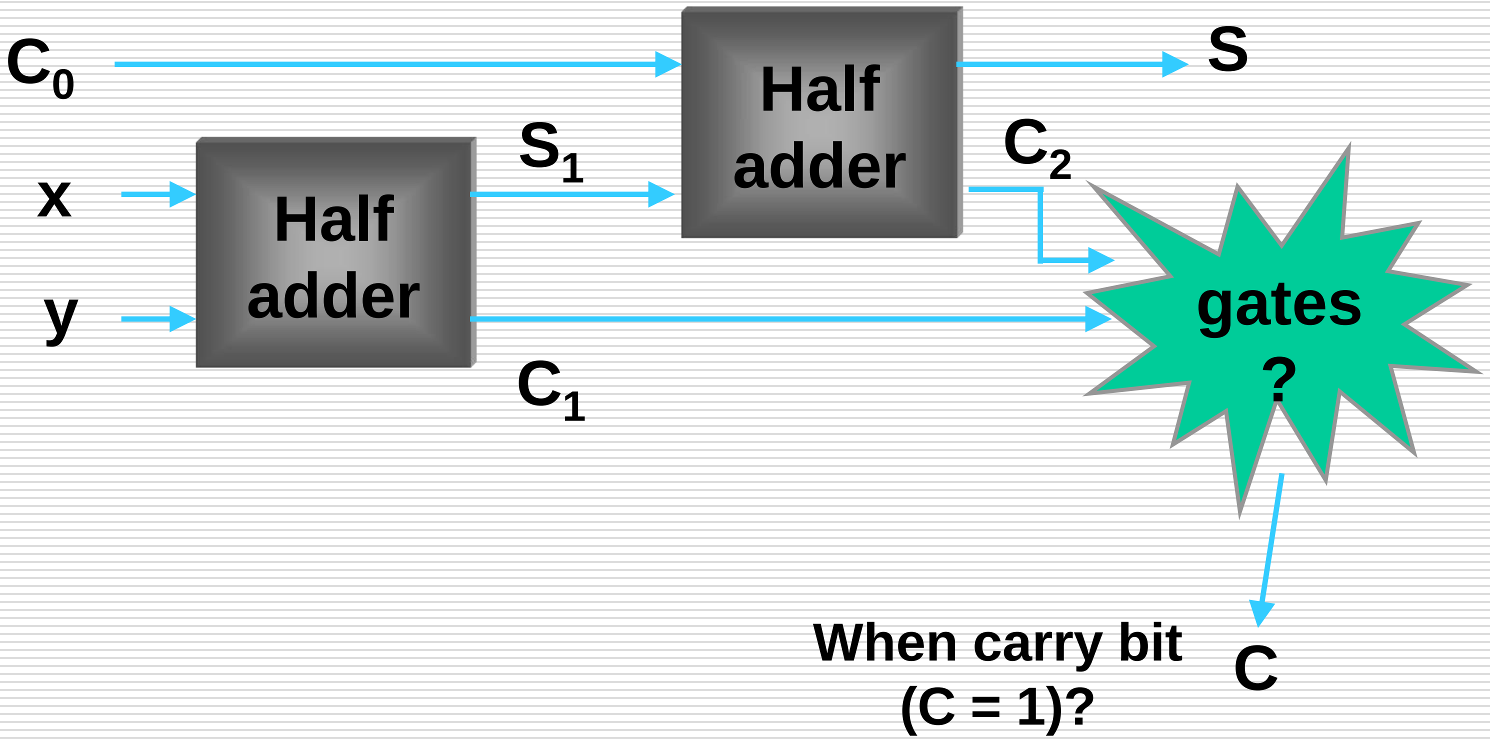
$S = x + y + C_0$

$S = (x + y) + C_0$

Tính: $S_1 = x + y$ → **Half adder 1**

Tính: $S_2 = S_1 + C_0$ → **Half adder 2**

Mạch cộng toàn phần (tt.)

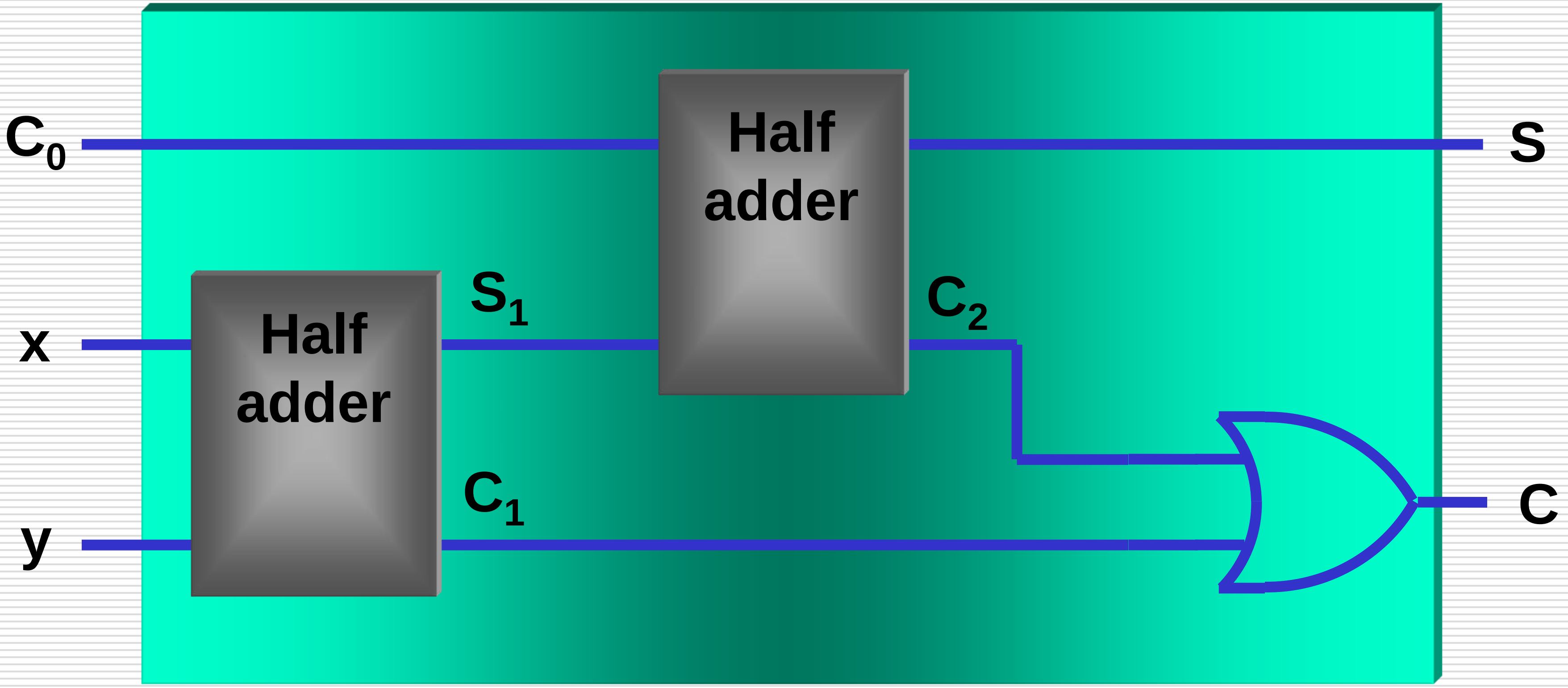


Full adder (2)

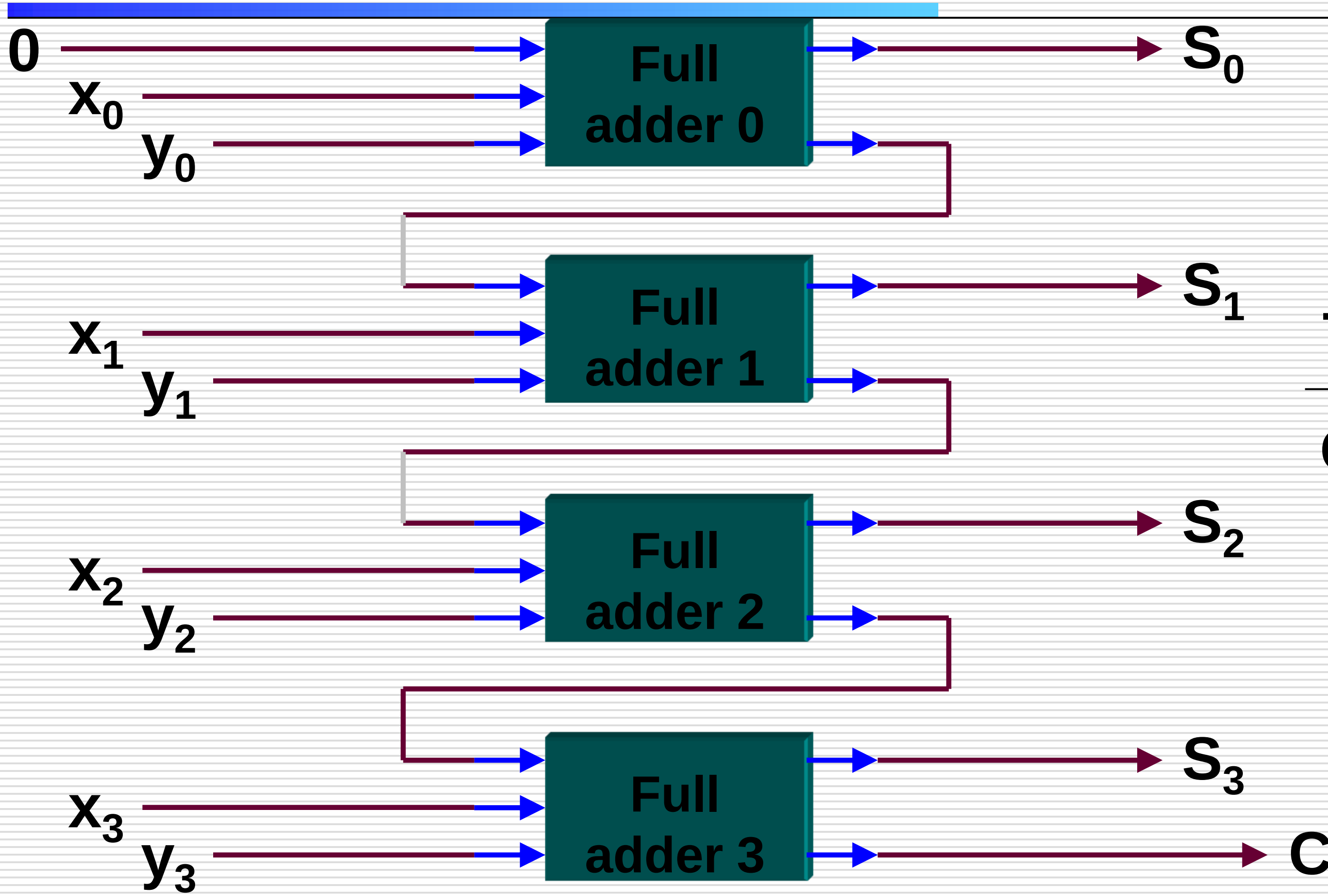
C_0	x	y	S	C	C_0	S_1	C_1	C_2	C
0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	0	0	0
0	1	0	1	0	0	1	0	0	0
0	1	1	0	1	0	0	1	0	1
1	0	0	1	0	1	0	0	0	0
1	0	1	0	1	1	1	0	1	1
1	1	0	0	1	1	1	0	1	1
1	1	1	1	1	1	0	1	0	1

C = 1 when C_1 = 1 orC_2 = 1

Full adder (3)



Adding Multiple Bits



$$\begin{array}{r} x_3x_2x_1x_0 \\ +\quad y_3y_2y_1y_0 \\ \hline C\,S_3S_2S_1S_0 \end{array}$$