COMPUTER ENGINEERING

# Kiểm tra 15’

- Rút gọn biểu thức sau

$$F = \Sigma_{W,X,Y,Z}(0,1,2,8,11) + d(3,9,15)$$

- Thiết kế mạch dựa trên bảng chân trị sau

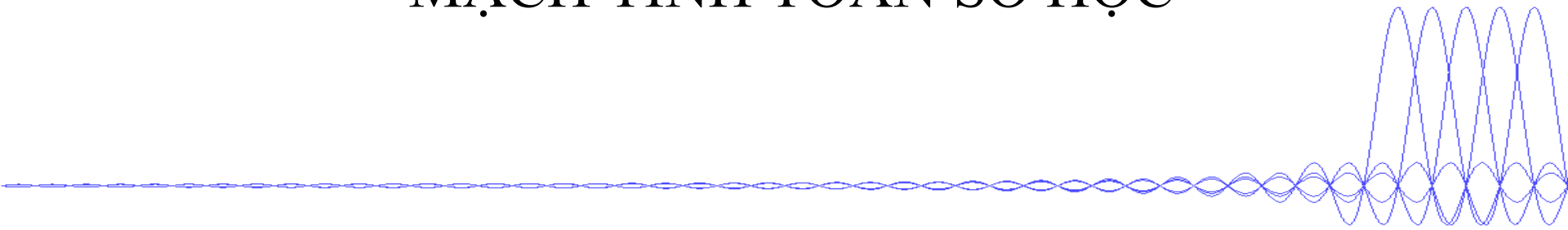
| $I_1$ | $I_0$ | $O_3$ | $O_2$ | $O_1$ | $O_0$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 |

COMPUTER ENGINEERING

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

# NHẬP MÔN MẠCH SỐ

## CHƯƠNG 5: MẠCH TỔ HỢP - MẠCH TÍNH TOÁN SỐ HỌC



COMPUTER ENGINEERING

# Nội dung

- Tổng quan
- Mạch cộng (Carry Ripple (CR) Adder)
- Mạch cộng nhìn trước số nhớ - (Carry Look-Ahead (CLA) Adder)
- Mạch cộng/ mạch trừ
- Đơn vị tính toán luận lý (Arithmetic Logic Unit)

COMPUTER ENGINEERING

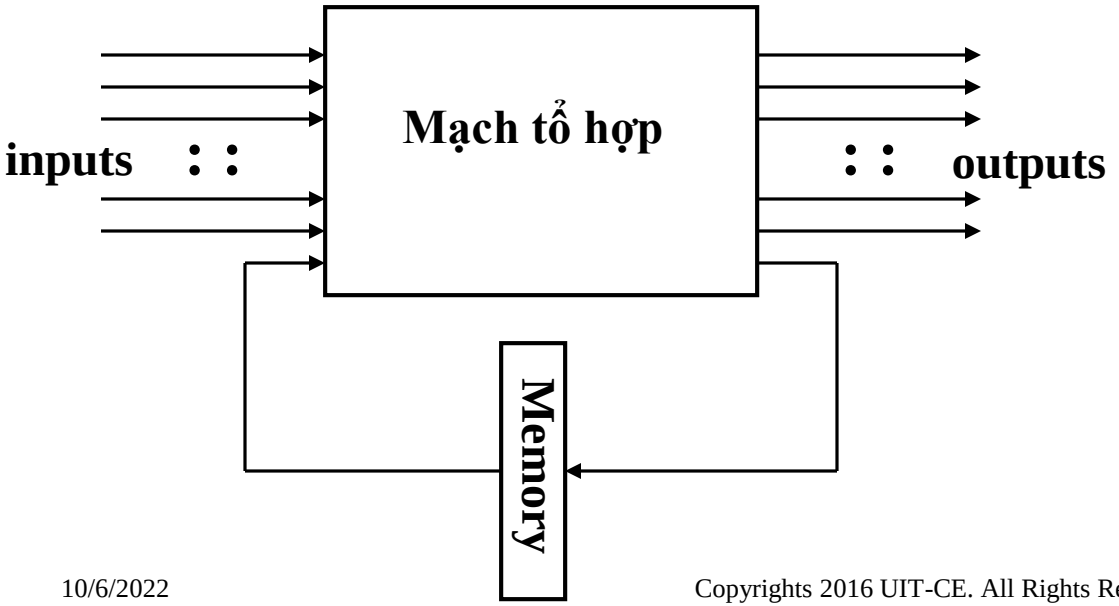
# Tổng quan

- Chương này sẽ học về:
  - Một số mạch logic tổ hợp thông dụng
  - Thiết kế các mạch logic tổ hợp phức tạp sử dụng các mạch logic tổ hợp thông dụng

# Phân biệt mạch tổ hợp và tuần tự


MẠCH TỔ HỢP

- Ngõ ra sẽ thay đổi lập tức khi ngõ vào thay đổi




MẠCH TUẦN TỰ

- Ngõ ra sẽ thay đổi phụ thuộc vào ngõ vào và trạng thái trước đó.
- Mạch có tính chất nhớ

COMPUTER ENGINEERING

# Nội dung

- Tổng quan
- Mạch cộng (Carry Ripple (CR) Adder)
- Mạch cộng nhìn trước số nhớ - (Carry Look-Ahead (CLA) Adder)
- Mạch cộng/ mạch trừ

# Mạch cộng bán phần (Half Adder)

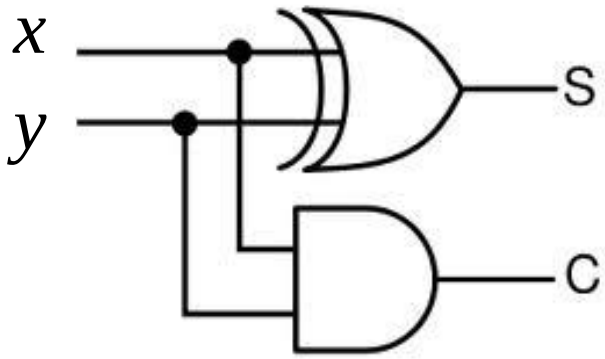
- Cộng 2 số 1 bit có 4 trường hợp

| | | | | |
|---|---|---|---|---|
| x | 0 | 0 | 1 | 1 |
| + y | + 0 | + 1 | + 0 | + 1 |
| c s | 0 0 | 0 1 | 0 1 | 1 0 |

Số nhớ Tổng

| x | y | c | s |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

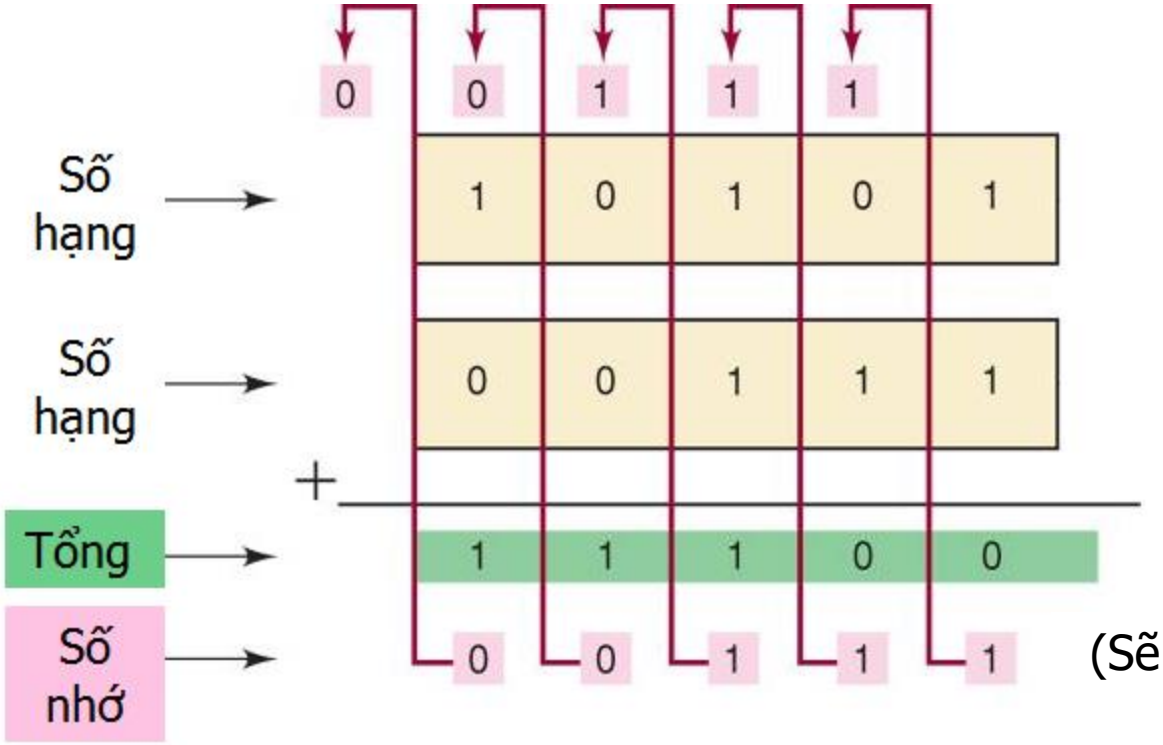
Mạch cộng 1 bit có tổng và số nhớ như thế này được gọi là mạch cộng bán phần (**HA**)




Sơ đồ mạch

# Mạch cộng toàn phần (Full Adder)

- Cộng những số có 2 hoặc nhiều bit
  - Cộng từng cặp bit bình thường
  - Nhưng ở vị trí cặp bit i, có thể có carry-in từ bit i-1



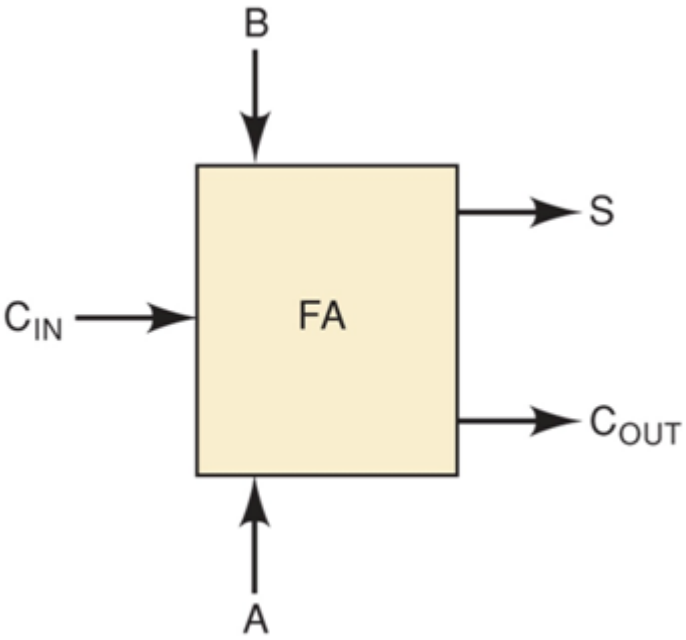

(Sẽ cộng vào vị trí kế tiếp)

COMPUTER ENGINEERING

# Mạch cộng toàn phần (Full Adder)

- Bộ cộng toàn phần (**FA**)
  - **3 ngõ vào** (2 ngõ vào cho 2 số 1-bit cần tính tổng, và 1 ngõ vào cho số nhớ đầu vào (**carry-in**))
  - **2 ngõ ra** (1 ngõ ra cho tổng và 1 cho số nhớ đầu ra (**carry-out**))

# Mạch cộng toàn phần (Full Adder)




Ký hiệu

Bảng sự thật

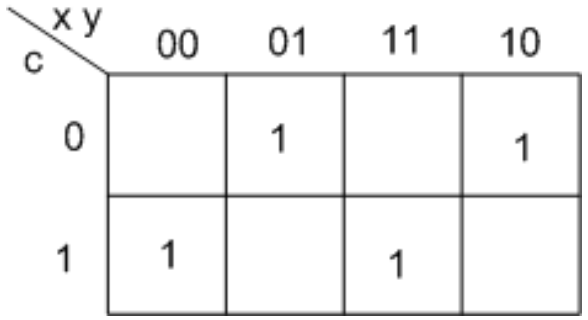
| Augend bit input | Addend bit input | Carry bit input | | Sum bit output | Carry bit output |
|---|---|---|---|---|---|
| A | B | $C_{IN}$ | | S | $C_{OUT}$ |
| 0 | 0 | 0 | | | |
| 0 | 0 | 1 | | | |
| 0 | 1 | 0 | | | |
| 0 | 1 | 1 | | | |
| 1 | 0 | 0 | | | |
| 1 | 0 | 1 | | | |
| 1 | 1 | 0 | | | |
| 1 | 1 | 1 | | | |

COMPUTER ENGINEERING

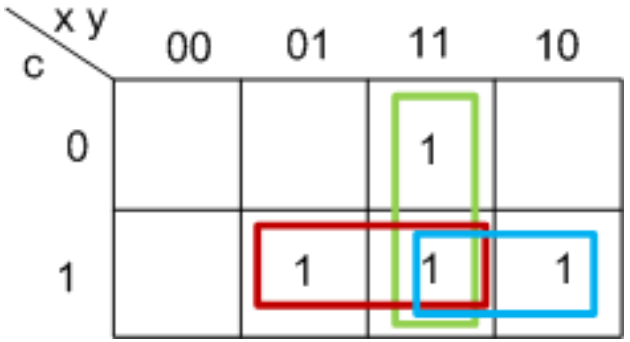
# Mạch cộng toàn phần (Full Adder)

Bảng sự thật

| Augend bit input | Addend bit input | Carry bit input | | Sum bit output | Carry bit output |
|---|---|---|---|---|---|
| A | B | $C_{IN}$ | | S | $C_{OUT}$ |
| 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 |




$$S_i = x_i \oplus y_i \oplus c_i$$




$$c_{i+1} = x_i y_i + x_i c_i + y_i c_i$$

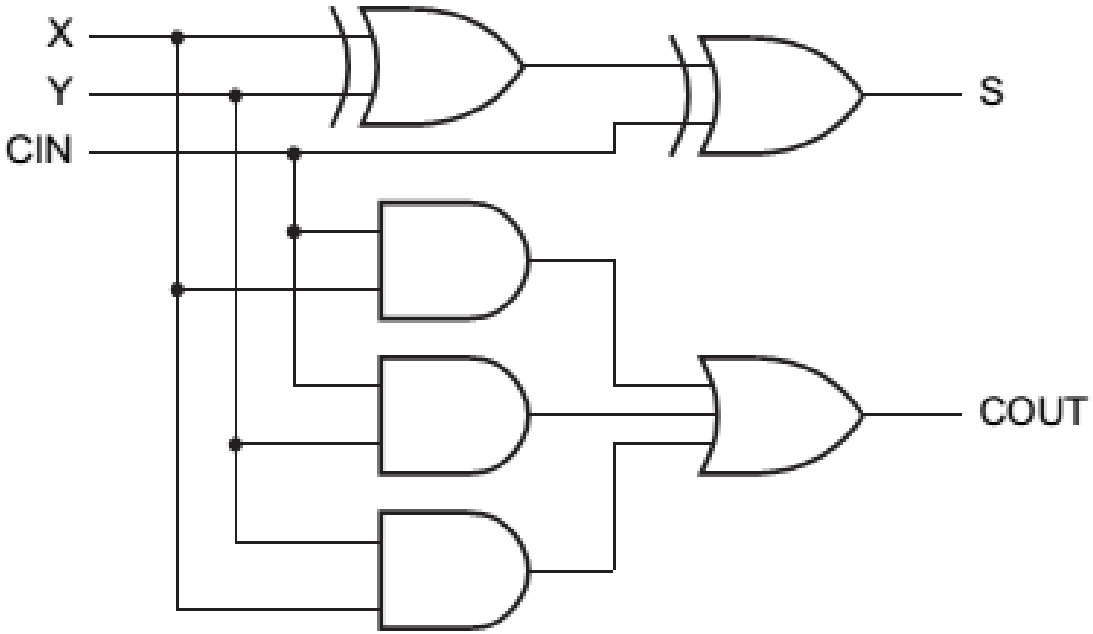
$$c_i = c_{IN} \qquad c_{i+1} = c_{OUT}$$

# Mạch cộng toàn phần (Full Adder)

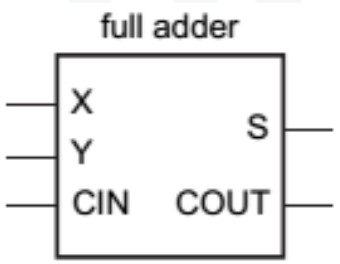
$$S_i = x_i \oplus y_i \oplus c_i$$

$$c_{i+1} = x_i y_i + x_i c_i + y_i c_i \qquad c_i = c_{IN}$$

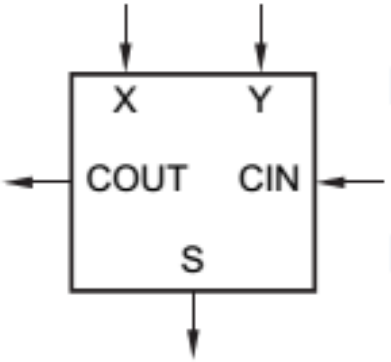
$$c_{i+1} = c_{OUT}$$




Sơ đồ mạch




Ký hiệu



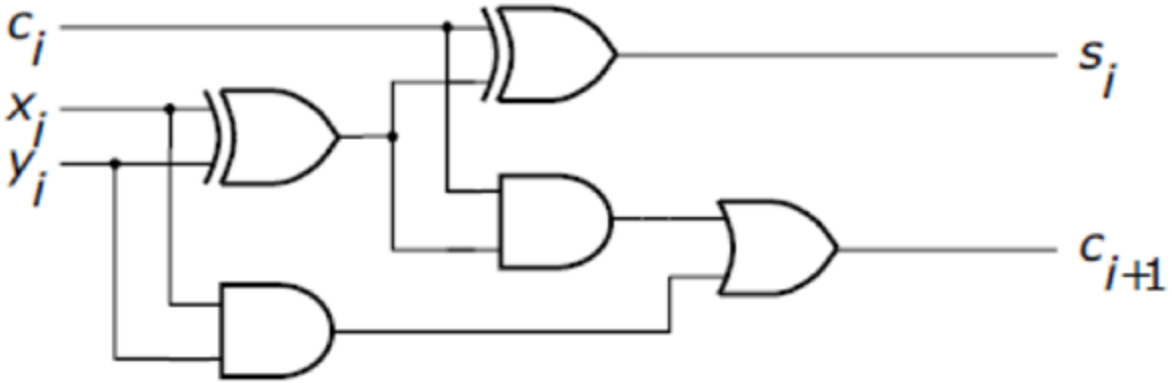

Ký hiệu khác

# Mạch cộng toàn phần (Full Adder)

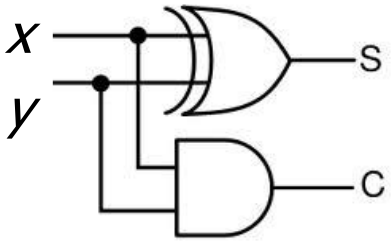
- Sử dụng lại HA

$$S_i = x_i \oplus y_i \oplus c_i$$

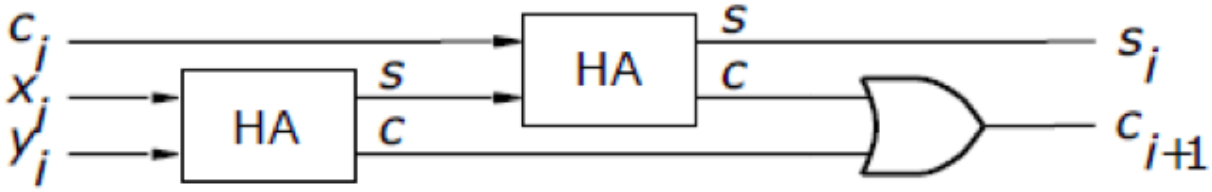
| c \ x y | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | | | 1 | |
| 1 | | 1 | 1 | 1 |




Sơ đồ mạch




Sơ đồ mạch HA

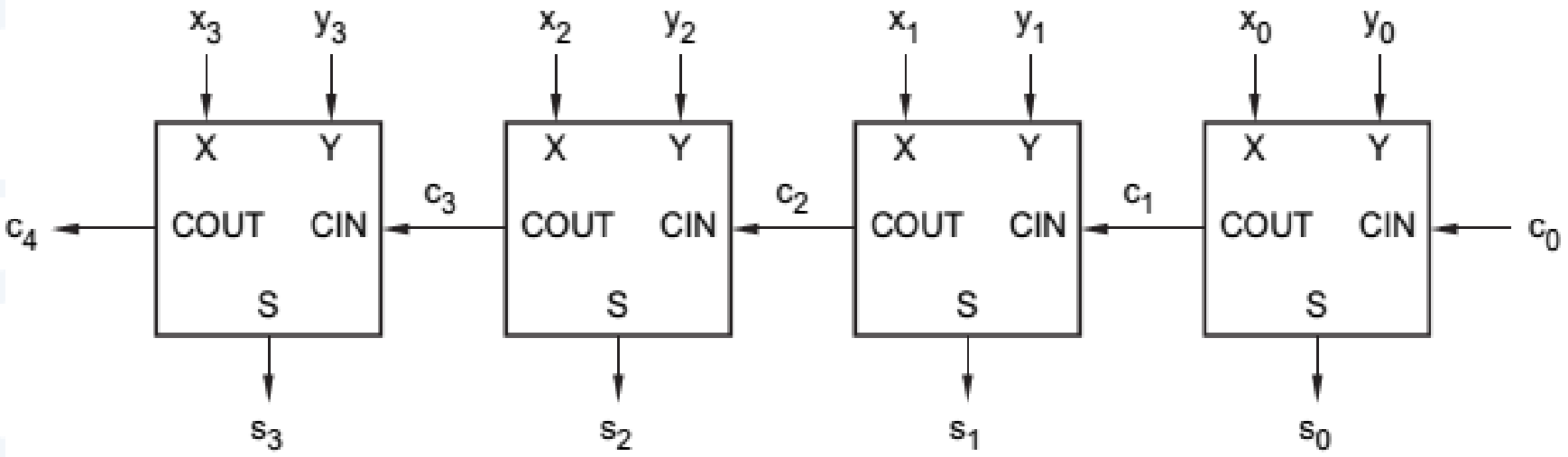



Sơ đồ mạch **FA** sử dụng lại HA

COMPUTER ENGINEERING

# Mạch cộng Carry Ripple (CR)

- Sơ đồ biểu diễn mạch cộng 4 bit song song sử dụng full adder



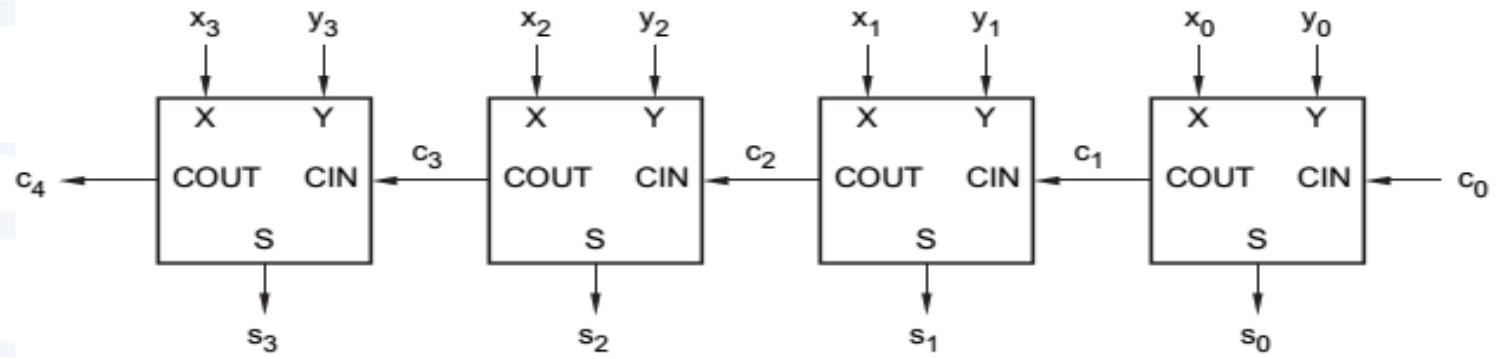
COMPUTER ENGINEERING

# Mạch cộng Carry Ripple (CR)

- Mạch FA bắt đầu với việc cộng các cặp bit từ LSB đến MSB
  - Nếu carry xuất hiện ở vị trí bit i, nó được cộng thêm vào phép cộng ở vị trí bit thứ i+1
- Việc kết hợp như vậy thường được gọi là mạch cộng **Carry-Ripple**
  - Vì carry được "ripple" từ FA này sang các FA kế tiếp
  - Tốc độ phép cộng bị giới hạn bởi quá trình truyền số nhớ

COMPUTER ENGINEERING

# Mạch cộng Carry Ripple (CR)




- Mỗi FA có một khoảng trễ (delay), giả sử là Δt
- Độ trễ phụ thuộc vào số lượng bit
  - Carry-out ở FA đầu tiên $C_1$ có được sau Δt
  - Carry-out ở FA đầu tiên $C_2$ có được sau 2Δt

  => $C_n$ được tính toán sau **n**Δt

Mô hình carry look ahead (CLA) thường được sử dụng để cải thiện tốc độ

COMPUTER ENGINEERING

# Nội dung

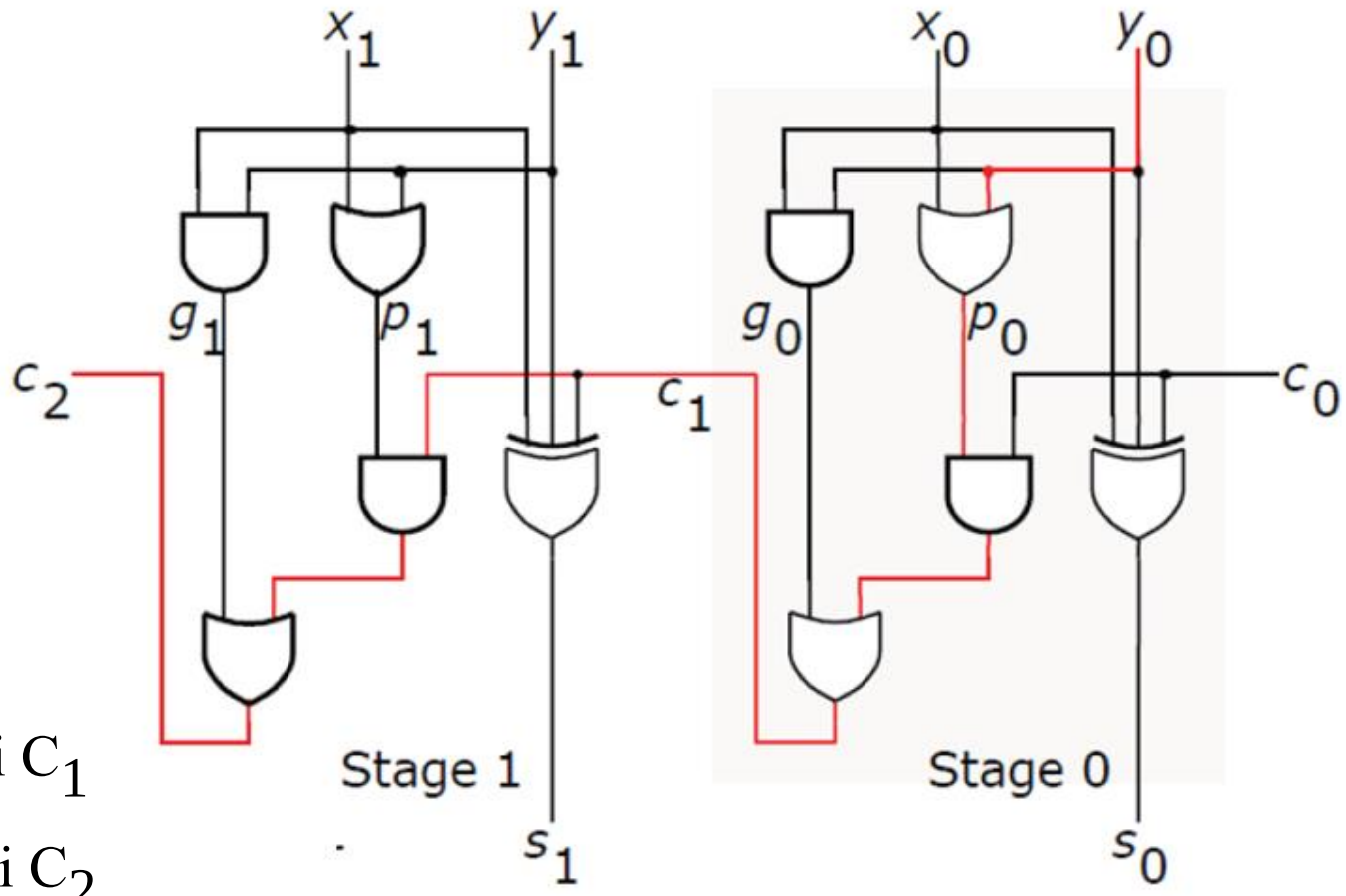
- Tổng quan
- Mạch cộng (Carry Ripple (CR) Adder)
- Mạch cộng nhìn trước số nhớ - (Carry Look-Ahead (CLA) Adder)
- Mạch cộng/ mạch trừ

COMPUTER ENGINEERING

# Critical path delay

- Tốc độ của mạch bị giới hạn bởi độ trễ lớn nhất dọc theo đường nối trong mạch
- Độ trễ lớn nhất được gọi là **critical path delay**
- Đường nối gây ra độ trễ đó gọi là **critical path**

# Mạch cộng Carry Ripple - critical path




Độ trễ 3 cổng đối với $C_1$

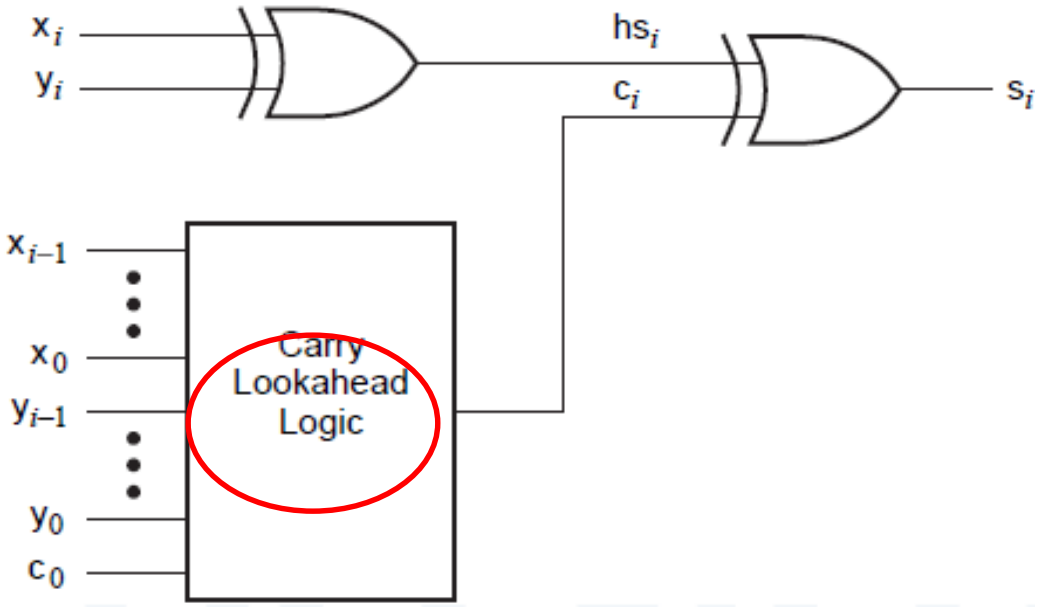
Độ trễ 5 cổng đối với $C_2$

Tổng quát, độ trễ **2n+1** cổng đối
với mạch cộng Carry Ripple **n-bit**

# Carry Look-Ahead Adder (CLA)

- Cải thiện tốc độ mạch cộng bằng cách
  - Tại mỗi tầng (stage), ta sẽ xác định nhanh giá trị **carry-in** ở tầng cộng trước đó sẽ có giá trị 0 hay 1

➔ Giảm Critical path delay



COMPUTER ENGINEERING

# Carry Look-Ahead Adder (CLA)

- Hàm xác định carry-out ở lần cộng thứ i

$$c_{i+1} = x_i y_i + x_i c_i + y_i c_i = x_i y_i + (x_i + y_i) c_i$$

- Đặt $g_i = x_i y_i$ và $p_i = x_i + y_i$ => $c_{i+1} = g_i + p_i c_i$
  - $g_i$ = 1 khi cả $x_i$ và $y_i$ đều bằng 1, không quan tâm $c_i$
    - g được gọi là **hàm generate,** vì carry-out luôn được generate ra khi g=1
  - $p_i$ = 1 khi $x_i$ = 1 hoặc $y_i$ = 1; carry-out = $c_i$
    - p được gọi là **hàm propagate**, vì carry-in = 1 được propagate (truyền) ở tầng cộng thứ i

# Carry Look-Ahead Adder (CLA)

- Xác định carry-out của mạch cộng n bit

$$c_n = g_{n-1} + p_{n-1}c_{n-1}$$

Mà $c_{n-1} = g_{n-2} + p_{n-2}c_{n-2}$

Do đó: $c_n = g_{n-1} + p_{n-1}(g_{n-2} + p_{n-2}c_{n-2})$

$$= g_{n-1} + p_{n-1}g_{n-2} + p_{n-1}p_{n-2}c_{n-2}$$

- Tiếp tục khai triển đến lần cộng đầu tiên

$$c_n = g_{n-1} + p_{n-1}g_{n-2} + p_{n-1}p_{n-2}g_{n-3} + \ldots + p_{n-1}p_{n-2}\ldots p_1g_0 + p_{n-1}p_{n-2}\ldots p_1p_0c_0$$

# Carry Look-Ahead Adder (CLA)

Số nhớ sinh ra ở lần cộng thứ **n-2** và được truyền qua các lần cộng còn lại

Số nhớ sinh ra ở lần cộng thứ 1 và được truyền qua các lần cộng còn lại

$$c_n = g_{n-1} + p_{n-1}g_{n-2} + p_{n-1}p_{n-2}g_{n-3} + \ldots + p_{n-1}p_{n-2}\cdots p_1 g_0 + p_{n-1}p_{n-2}\cdots p_1 p_0 c_0$$

Số nhớ sinh ra ở lần cộng cuối cùng

Số nhớ sinh ra ở lần cộng thứ **n-3** và được truyền qua các lần cộng còn lại

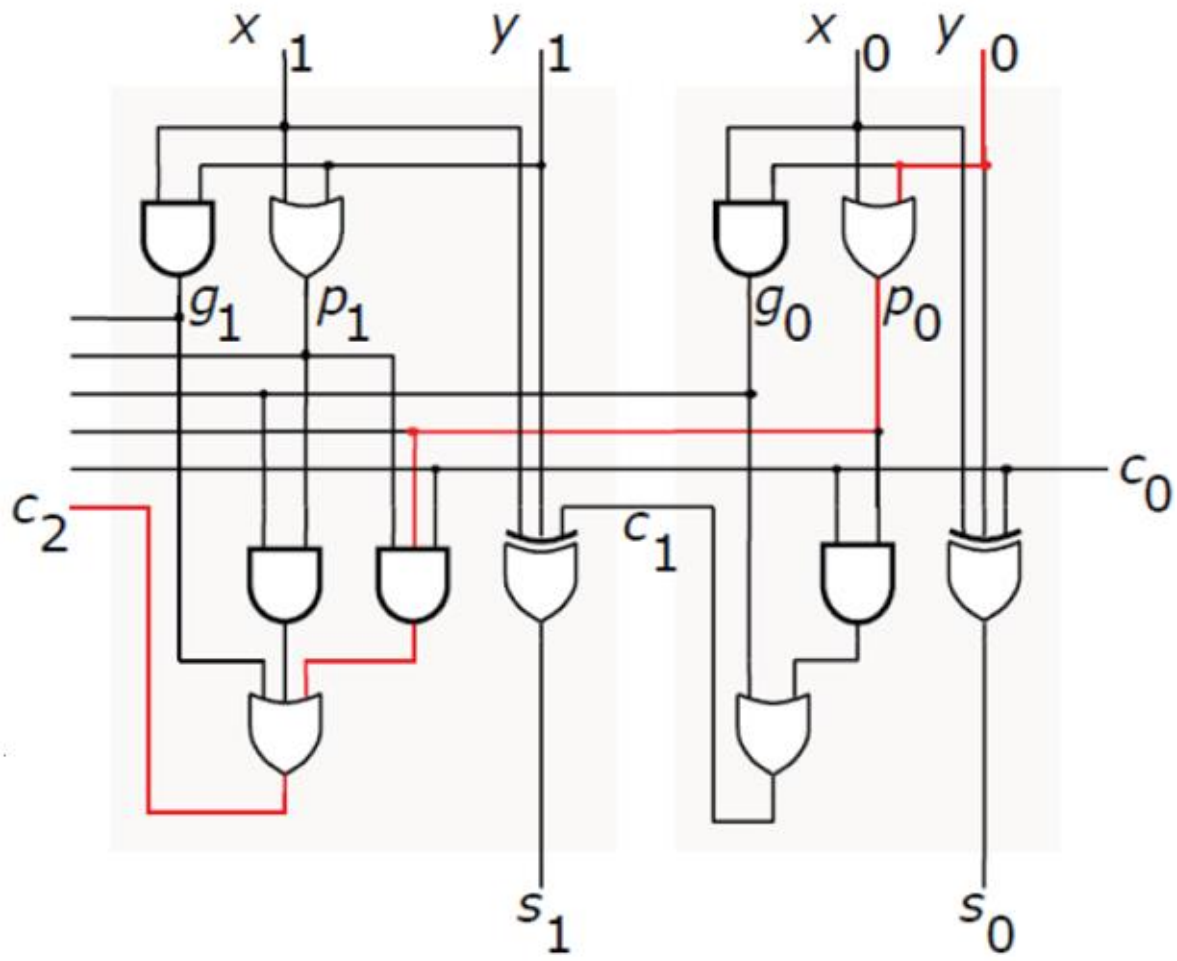
Số nhớ đầu vào $c_0$ được truyền qua tất cả các lần cộng

# Carry Look-Ahead Adder (CLA)

- Ví dụ: Trường hợp cộng 2 bit

$C_1 = G_0 + P_0.C_0$

$C_2 = G_1 + P_1.G_0 + P_1.P_0.C_0$



# Mạch cộng CLA - critical path

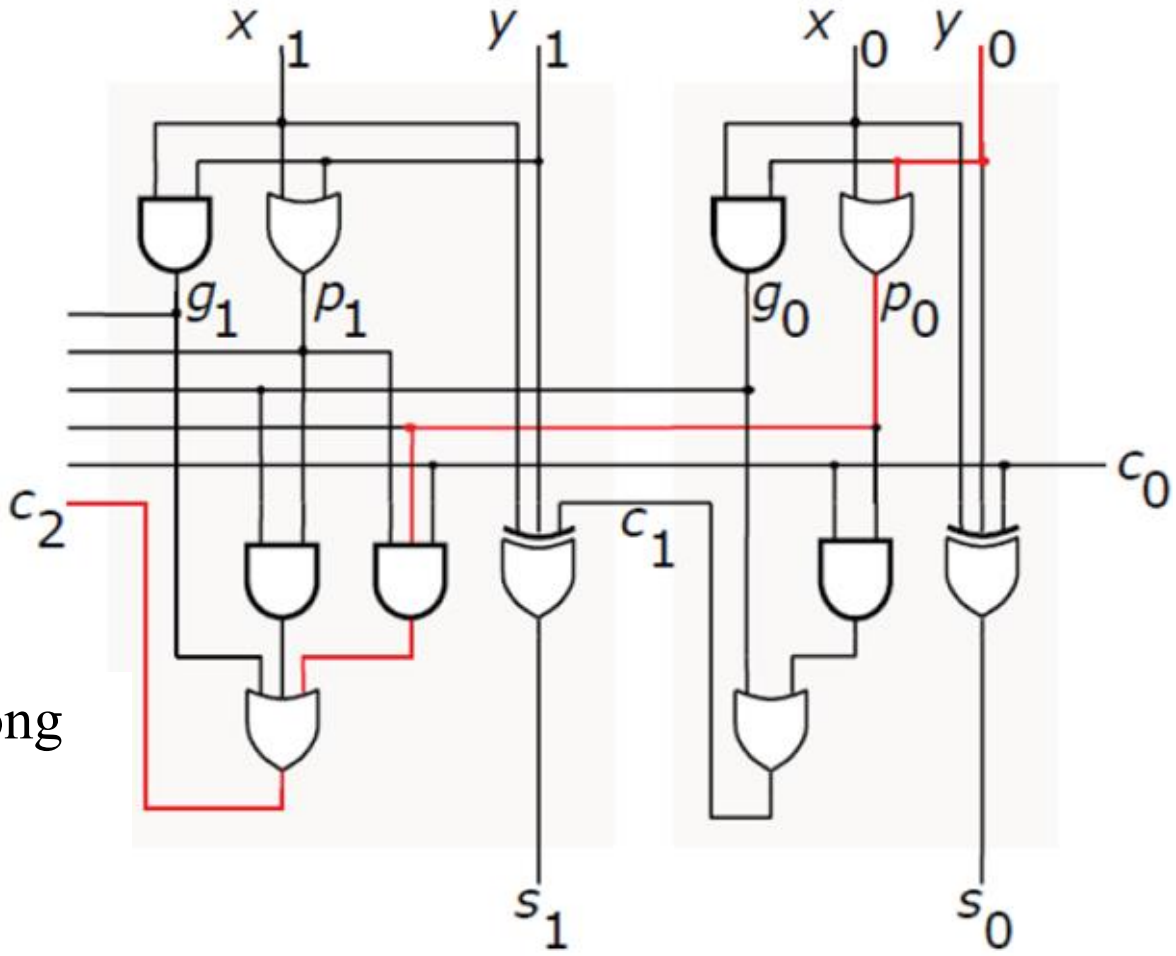
$C_1 = G_0 + P_0.C_0$

$C_2 = G_1 + P_1.G_0 + P_1.P_0.C_0$

Độ trễ 3 cổng đối với $C_1$

Độ trễ 3 cổng đối với $C_2$

Độ trễ 3 cổng đối với $C_n$




**Độ trễ** tổng cộng cho mạch cộng **CLA n-bit** là độ trễ **4 cổng**

- gi, pi: độ trễ 1 cổng
- Ci: độ trễ 2 cổng
- Độ trễ 1 cộng còn lại là do tính tổng s

COMPUTER ENGINEERING

# Giới hạn của mạch cộng CLA

- Biểu thức tính carry trong mạch cộng CLA

$$c_n = g_{n-1} + p_{n-1}g_{n-2} + p_{n-1}p_{n-2}g_{n-3} + \ldots + p_{n-1}p_{n-2}\ldots p_1 g_0 + p_{n-1}p_{n-2}\ldots p_1 p_0 c_0$$

- Độ phức tạp tăng lên nhanh chóng khi n lớn
- Vấn đề Fan-in có thể hạn chế tốc độ của mạch cộng CLA

COMPUTER ENGINEERING

# Nội dung

- Tổng quan
- Mạch cộng (Carry Ripple (CR) Adder)
- Mạch cộng nhìn trước số nhớ - (Carry Look-Ahead (CLA) Adder)
- Mạch trừ, mạch báo tràn, mạch cộng trừ

# Mạch trừ

- X, Y là 2 <u>số không dấu</u> n-bit

Phép cộng: $S = X + Y$

Phép trừ: $D = X - Y$

$= X + (-Y)$

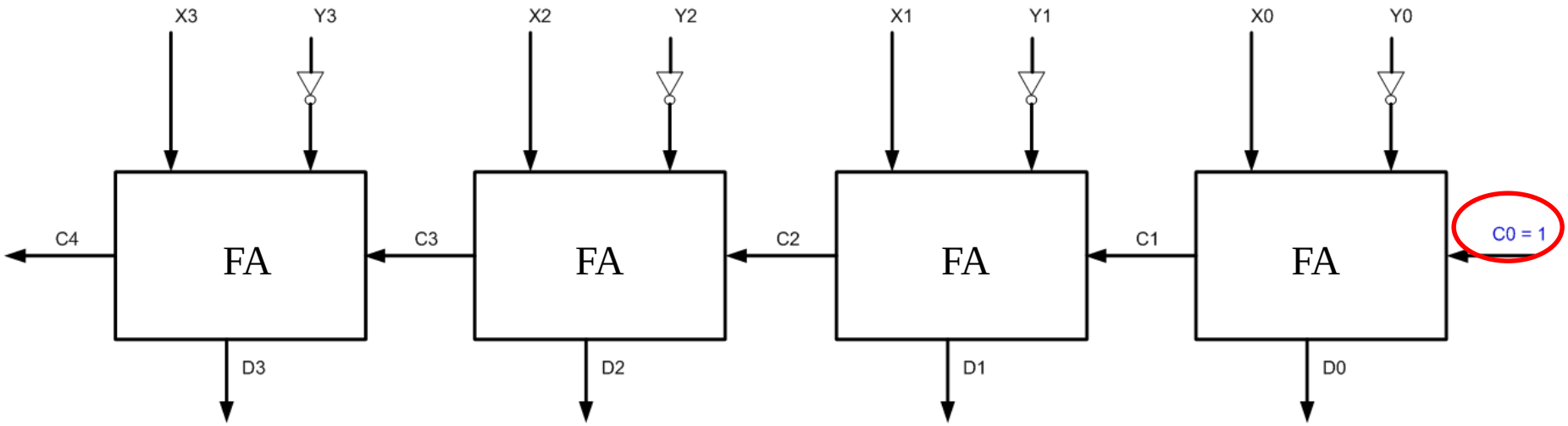
$= X+$ (Bù 2 của Y)

$= X+$ (Bù 1 của Y) + 1

$= X+ Y' + 1$

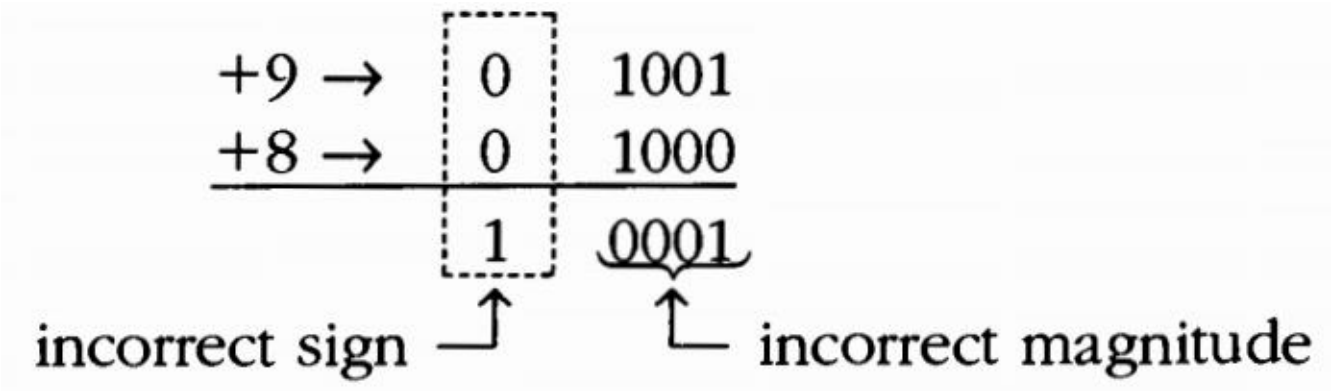
# Mạch trừ

- Mạch cộng Carry Ripple có thể được dùng để xây dựng mạch trừ Carry Ripple bằng cách đảo Y và đặt số nhớ đầu tiên là 1



# Mạch báo tràn

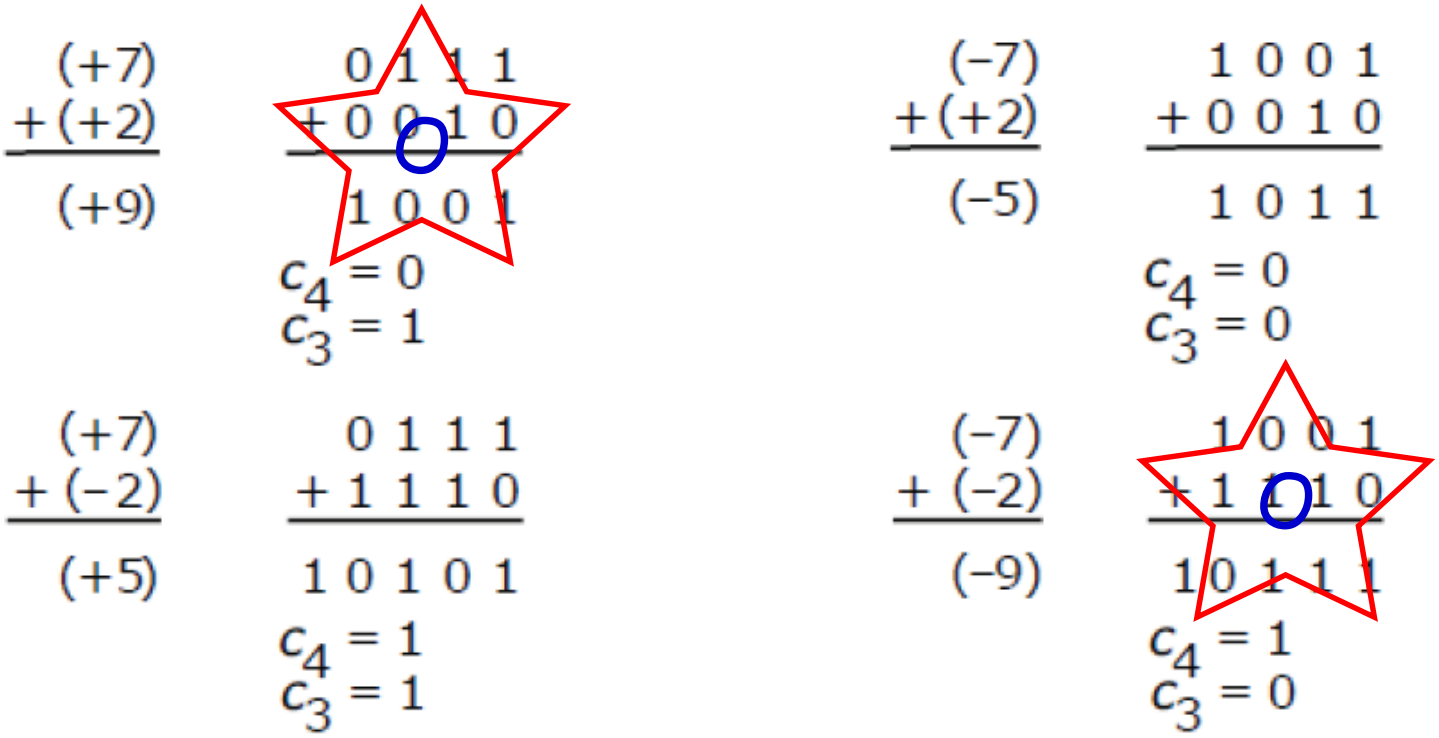
- Tràn (Overflow): là khi kết quả của phép toán vượt quá số bit biểu diễn phần giá trị
  - n bit biểu diễn được số từ $-2^{n-1}$ đến $+2^{n-1}-1$
  - Overflow luôn cho ra 1 kết quả sai




➔Mạch để xác định có overflow hay không

# Mạch báo tràn

- Ví dụ: Xét cộng 2 số 4 bit (3 bit giá trị và 1 bit dấu) sau:




- Overflow không xuất hiện khi cộng 2 số trái dấu

# Mạch báo tràn

- Tràn có thể phát hiện được bởi mạch phát hiện cờ tràn như sau:
  - Mạch cộng 4 bit:

$$Overflow = c_3\overline{c_4} + \overline{c_3}c_4$$
$$Overflow = c_3 \oplus c_4$$

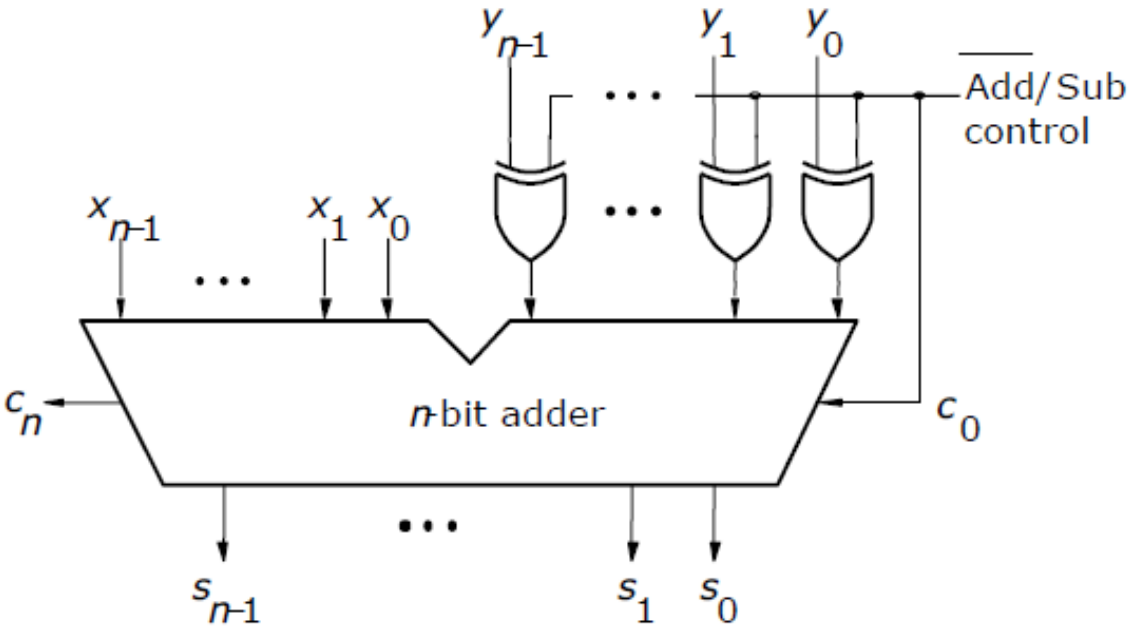
  - Với n bit

$$Overflow = c_{n-1} \oplus c_n$$

➔Mạch cộng/ trừ có thể bổ sung mạch kiểm tra tràn với 1 cổng XOR. Nếu sau khi thực hiện phép tính, cờ tràn có giá trị bằng "1" thì ta không cần quan tâm giá trị của phép tính vì giá trị đó bị sai.

# Mạch cộng/trừ

- Thiết kế một mạch cộng/ trừ với 1 ngõ điều khiển ADD/SUB
  - ADD = 0: mạch thực hiện cộng 2 số X + Y
  - SUB = 1: mạch thực hiện trừ 2 số X - Y



COMPUTER ENGINEERING

# Tóm tắt nội dung chương học

- Qua Phần 1 - Chương 5, sinh viên cần nắm những nội dung chính sau:
  - Sự khác biệt giữa mạch tổ hợp và mạch tuần tự? Khi nào thì ta cần thiết kế mạch tổ hợp và khi nào thì ta cần thiết kế mạch tuần tự trong thiết kế hệ thống mạch số
  - Phương pháp thiết kế mạch tổ hợp: Mạch cộng HA, FA, CRA, CLA, Mạch trừ, Mạch báo tràn. Ưu và khuyết của mạch CRA và CLA.

COMPUTER ENGINEERING

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

# Any question?