



# Chương 2: Khối xử lý trung tâm CPU

# Chương 2: Nội dung chính

---

- Sơ đồ khối tổng quát
- Chu kỳ xử lý lệnh
- Thanh ghi
- Khối điều khiển (CU)
- Khối số học và logic (ALU)
- Bus trong CPU

# CPU - Sơ đồ khối tổng quát

CU: (Control Unit) Khối điều khiển

IR: (Instruction Register) Thanh ghi lệnh

PC: (Program Counter) Bộ đếm chương trình

MAR: (Memory Address Register) Thanh ghi địa chỉ bộ nhớ

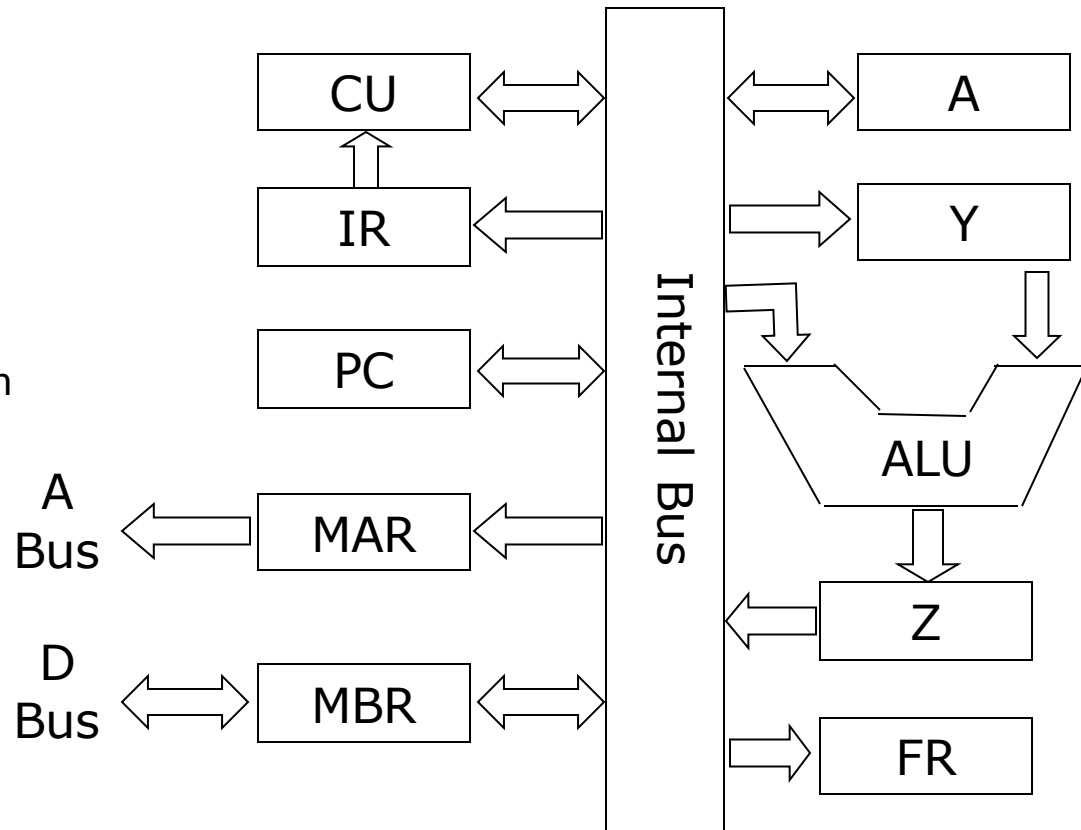
MBR: (Memory Buffer Register) Thanh ghi nhớ đệm

A: (Accumulator Register) Thanh ghi tích lũy

Y, Z: (Temporary Register) Thanh ghi tạm thời

FR: (Flag Register) Thanh ghi cờ

ALU: (Arithmetic and Logic Unit) Khối tính toán số học -logic



# Chu kỳ xử lý lệnh

---

1. Khi một chương trình được chạy, hệ điều hành tải mã chương trình vào bộ nhớ trong
2. Địa chỉ lệnh đầu tiên của chương trình được đưa vào thanh ghi PC
3. Địa chỉ của ô nhớ chứa lệnh được chuyển tới bus A qua thanh ghi MAR
4. Tiếp theo, bus A truyền địa chỉ tới khối quản lý bộ nhớ MMU (Memory Management Unit)
5. MMU chọn ô nhớ và sinh ra tín hiệu READ

# Chu kỳ xử lý lệnh

---

6. Lệnh chứa trong ô nhớ được chuyển tới thanh ghi MBR qua bus D
7. MBR chuyển lệnh tới thanh ghi IR. Sau đó IR lại chuyển lệnh tới CU
8. CU giải mã lệnh và sinh ra các tín hiệu xử lý cho các đơn vị khác, ví dụ như ALU để thực hiện lệnh
9. Địa chỉ trong PC được tăng lên để trỏ tới lệnh tiếp theo của chương trình sẽ được thực hiện
10. Thực hiện lại các bước 3->9 để chạy hết các lệnh của chương trình

# Thanh ghi

---

- Thanh ghi là thành phần nhớ ở bên trong CPU:
  - Lưu trữ tạm thời lệnh và dữ liệu cho CPU xử lý
  - Dung lượng nhỏ, số lượng ít
  - Tốc độ rất cao (bằng tốc độ CPU)
- Các CPU thế hệ cũ (80x86) có 16 – 32 thanh ghi.  
CPU thế hệ mới (Intel Pentium 4, Core 2 Duo) có hàng trăm thanh ghi
- Kích thước thanh ghi phụ thuộc vào thiết kế CPU: 8, 16, 32, 64, 128 và 256 bit
  - 8086 và 80286: 8 và 16 bit
  - 80386, Pentium II: 16 – 32 bit
  - Pentium IV, Core Duo: 32, 64 và 128 bit

# Thanh ghi tích lũy A (Accumulator)

---

- Thanh ghi tích lũy hay thanh ghi A là một trong những thanh ghi quan trọng nhất của CPU
  - Lưu trữ các toán hạng đầu vào
  - Lưu kết quả đầu ra
- Kích thước của thanh ghi A tương ứng với độ dài từ xử lý của CPU: 8, 16, 32, 64 bit
- Cũng được sử dụng để trao đổi dữ liệu với các thiết bị vào ra

# Thanh ghi tích lũy

---

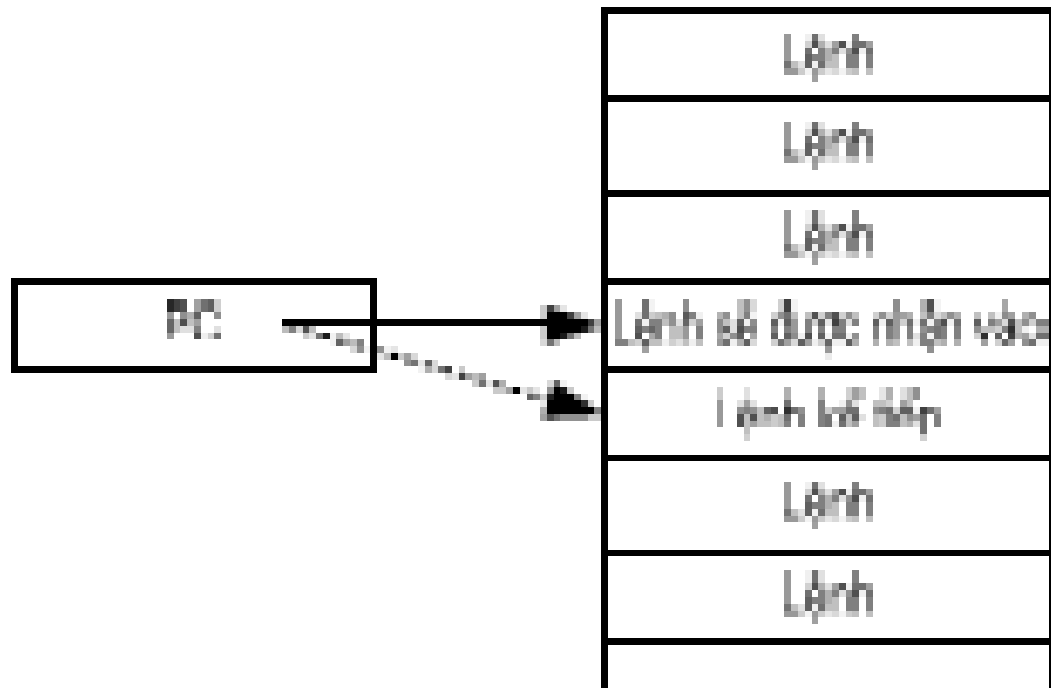
- Ví dụ: thực hiện phép tính  $x + y \rightarrow s$ 
  - Toán hạng  $x$  được đưa vào thanh ghi  $A$
  - Toán hạng  $y$  được đưa vào thanh ghi  $Y$
  - ALU thực hiện phép cộng  $A+Y$  , kết quả được lưu vào  $Z$
  - Kết quả sau đó lại được đưa vào  $A$



# Bộ đếm chương trình PC

---

- ❑ Program Counter hay Instruction Pointer lưu địa chỉ bộ nhớ của lệnh tiếp theo
- ❑ PC chứa địa chỉ ô nhớ chứa lệnh đầu tiên của chương trình khi nó được kích hoạt và được tải vào bộ nhớ
- ❑ Sau khi CPU chạy xong 1 lệnh, địa chỉ ô nhớ chứa lệnh tiếp theo được tải vào PC
- ❑ Kích thước của PC phụ thuộc vào thiết kế CPU: 8, 16, 32, 64 bit



# Thanh ghi trạng thái FR

---

- Mỗi bit của thanh ghi cờ lưu trữ trạng thái kết quả phép tính được ALU thực hiện
- Có 2 kiểu cờ:
  - Cờ trạng thái: CF, OF, AF, ZF, PF, SF
  - Cờ điều khiển: IF, TF, DF
- Các bit cờ thường được dùng là các điều kiện rẽ nhánh lệnh tạo logic chương trình
- Kích thước FR phụ thuộc thiết kế CPU

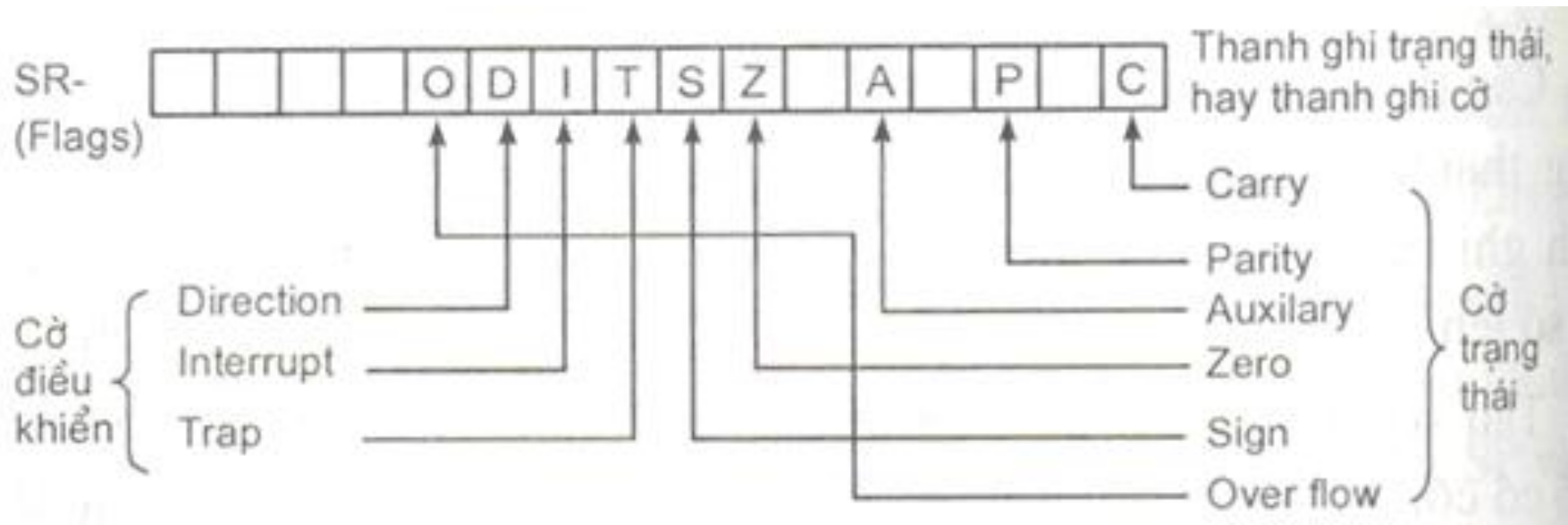
# CPU Registers - FR

---

Flag	ZF	SF	CF	AF	IF	OF	PF	1
Bit No	7	6	5	4	3	2	1	0

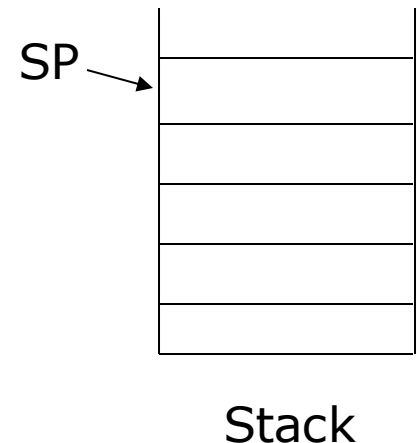
- ❑ ZF: Zero Flag, ZF=1 nếu kết quả =0 và ZF=0 nếu kết quả  $\neq 0$ .
- ❑ SF: Sign Flag, SF=1 nếu kết quả âm và SF=0 nếu kết quả dương
- ❑ CF: Carry Flag, CF=1 nếu có nhớ/mượn ở bit trái nhất
- ❑ AF: Auxiliary Flag, AF=1 nếu có nhớ ở bit trái nhất của nibble
- ❑ OF: Overflow Flag, OF=1 nếu có tràn, OF=0 ngược lại
- ❑ PF: Parity Flag, PF=1 nếu tổng số bit 1 trong kết quả là số lẻ, PF=0 ngược lại
- ❑ IF: Interrupt Flag, IF=1: ngắt được phép, IF=0: cấm ngắt

# Thanh ghi trạng thái của 8086



# Con trỏ ngăn xếp (SP: Stack Pointer)

- Ngăn xếp là 1 đoạn bộ nhớ đặc biệt hoạt động theo nguyên tắc vào sau ra trước (LIFO)
- Con trỏ ngăn xếp là thanh ghi luôn trỏ tới đỉnh của ngăn xếp
- 2 thao tác với ngăn xếp:
  - Push: đẩy dữ liệu vào ngăn xếp  
 $SP \leftarrow SP + 1$   
 $\{SP\} \leftarrow \text{Data}$
  - Pop: lấy dữ liệu ra khỏi ngăn xếp  
 $\text{Register} \leftarrow \{SP\}$   
 $SP \leftarrow SP - 1$



# Các thanh ghi đa năng

---

- Có thể sử dụng cho nhiều mục đích:
  - Lưu các toán hạng đầu vào
  - Lưu các kết quả đầu ra
- Ví dụ: CPU 8086 có 4 thanh ghi đa năng
  - AX: Accumulator Register
  - BX: Base Register
  - CX: Counter Register
  - DX: Data Register

# Thanh ghi lệnh IR

---

- ❑ Lưu trữ lệnh đang được xử lý
- ❑ IR lấy lệnh từ MBR và chuyển nó tới CU để giải mã lệnh





# Thanh ghi MBR và MAR

---

- MAR: thanh ghi địa chỉ bộ nhớ
  - Giao diện giữa CPU và bus địa chỉ
  - Nhận địa chỉ bộ nhớ của lệnh tiếp theo từ PC và chuyển nó tới bus địa chỉ
- MBR: thanh ghi đệm bộ nhớ
  - Giao diện giữa CPU và bus dữ liệu
  - Nhận lệnh từ bus dữ liệu và chuyển nó tới IR

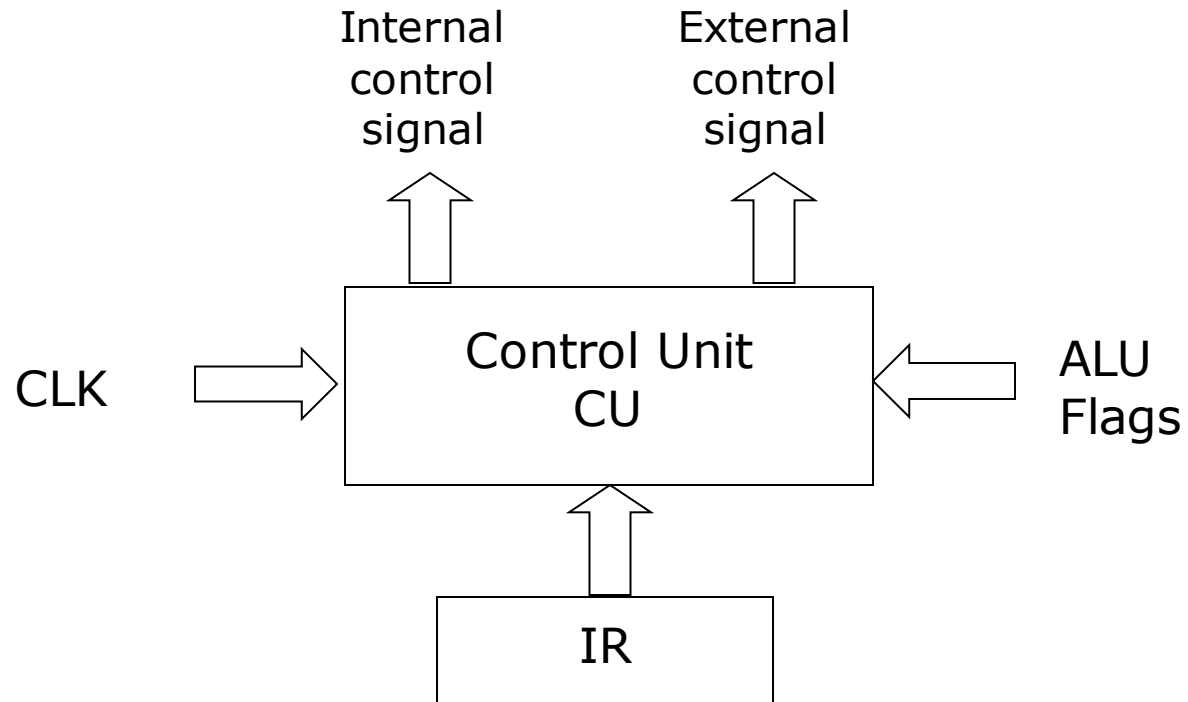
# Các thanh ghi tạm thời

---

- CPU thường sử dụng một số thanh ghi tạm thời để:
  - Lưu trữ các toán hạng đầu vào
  - Lưu các kết quả đầu ra
  - Hỗ trợ xử lý song song (tại một thời điểm chạy nhiều hơn 1 lệnh)
  - Hỗ trợ thực hiện lệnh theo cơ chế thực hiện tiên tiến kiểu không trật tự (OOO – Out Of Order execution)

# Khối điều khiển CU

---



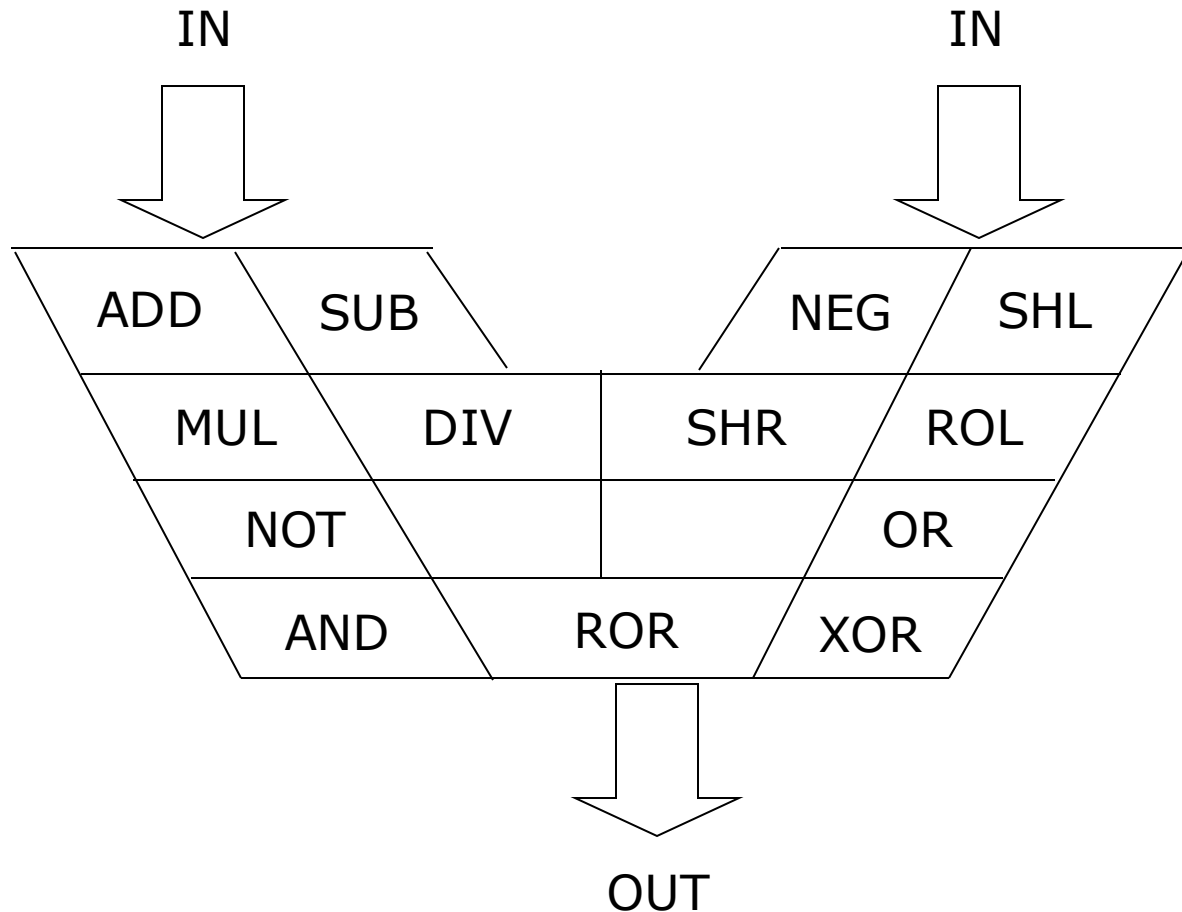
# Khối điều khiển CU

---

- ❑ Điều khiển tất cả các hoạt động của CPU theo xung nhịp đồng hồ
- ❑ Nhận 3 tín hiệu đầu vào:
  - Lệnh từ IR
  - Giá trị các cờ trạng thái
  - Xung đồng hồ
- ❑ CU sinh 2 nhóm tín hiệu đầu ra:
  - Nhóm tín hiệu điều khiển các bộ phận bên trong CPU
  - Nhóm tín hiệu điều khiển các bộ phận bên ngoài CPU
- ❑ Sử dụng nhịp đồng hồ để đồng bộ hóa các đơn vị bên trong CPU và giữa CPU với các thành phần bên ngoài

# Khối số học và logic ALU

---



# Khối số học và logic ALU

---

- Bao gồm các đơn vị chức năng con để thực hiện các phép toán số học và logic:
  - Bộ cộng (ADD), bộ trừ (SUB), bộ nhân (MUL), bộ chia (DIV), ...
  - Các bộ dịch (SHIFT) và quay (ROTATE)
  - Bộ phủ định (NOT), bộ và (AND), bộ hoặc (OR), và bộ hoặc loại trừ (XOR)
- ALU có:
  - 2 cổng IN để nhận đầu vào từ các thanh ghi
  - 1 cổng OUT được nối với bus trong để gửi kết quả tới các thanh ghi

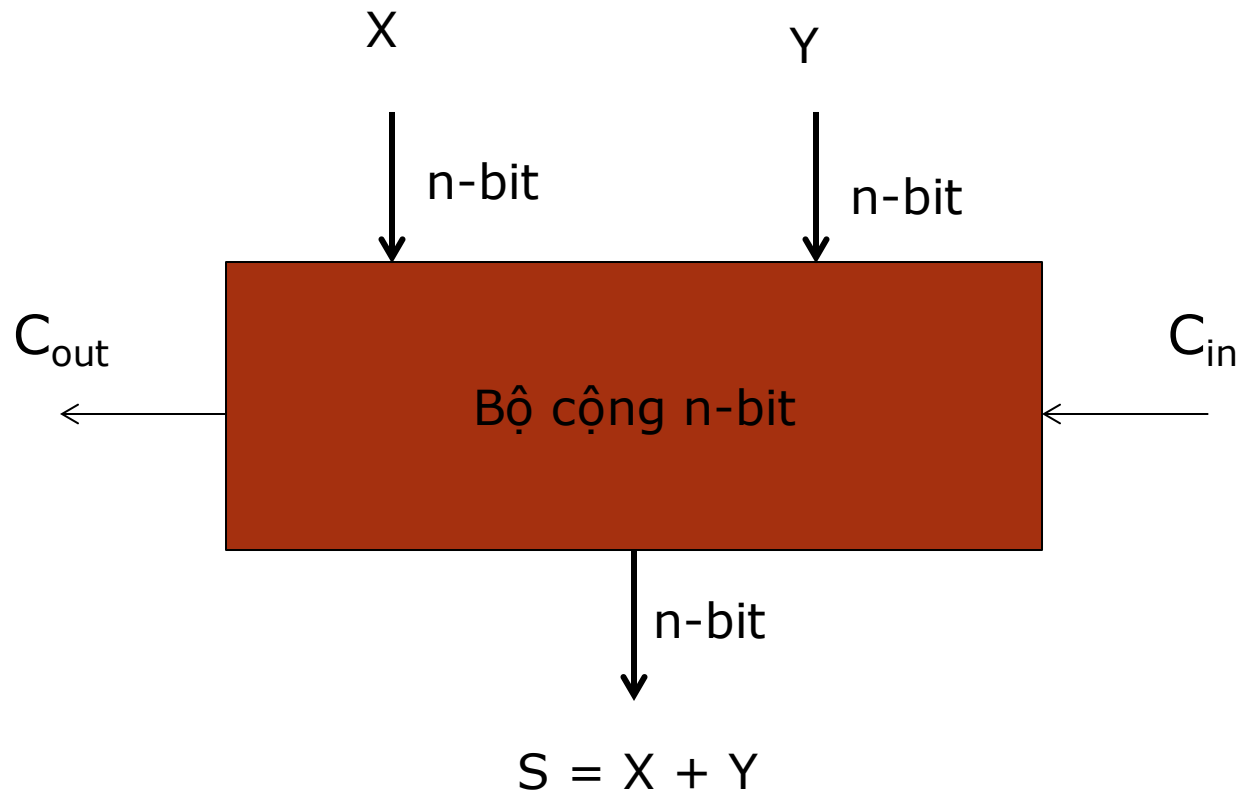
# Bus trong

---

- ❑ Bus trong là kênh liên lạc của tất cả các thành phần trong CPU
- ❑ Hỗ trợ liên lạc 2 chiều
- ❑ Bus trong có giao diện để trao đổi thông tin với bus ngoài (bus hệ thống)
- ❑ Bus trong luôn có băng thông lớn và tốc độ nhanh hơn so với bus ngoài

# Bộ Cộng Nhị phân n-bit

---





# Bộ Cộng Nhị phân không dấu n-bit

---

- Khi cộng hai số nguyên không dấu n-bit, kết quả nhận được là n-bit:
  - Nếu  $C_{out}=0 \rightarrow$  nhận được kết quả đúng.
  - Nếu  $C_{out}=1 \rightarrow$  nhận được kết quả sai, do tràn nhớ ra ngoài (Carry Out).
  - Tràn nhớ ra ngoài khi: tổng  $> (2^n - 1)$

# Bộ Cộng Nhị phân có dấu n-bit

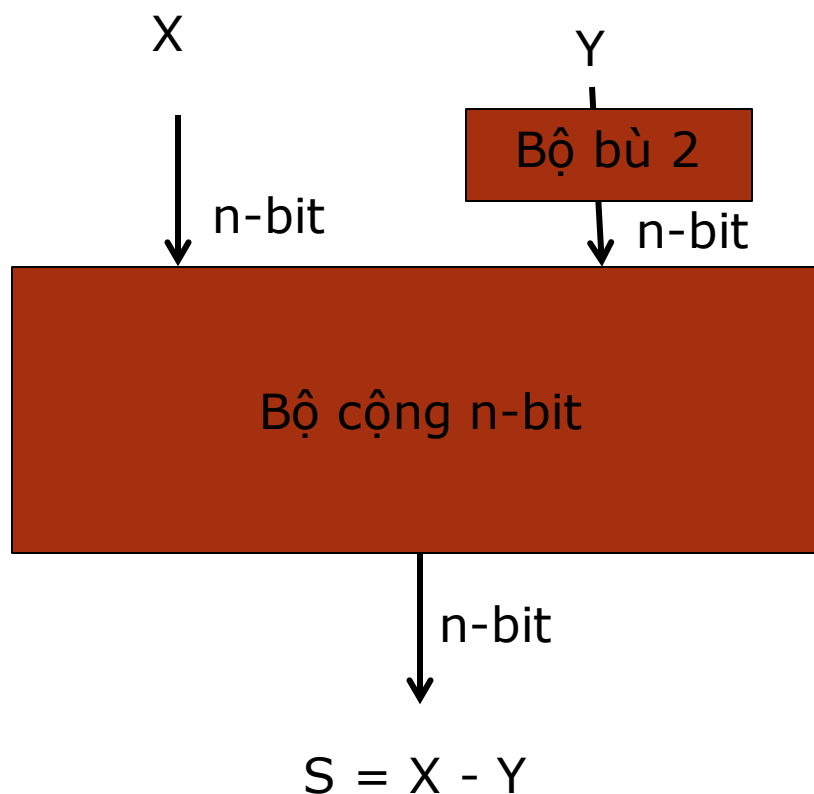
---

- Khi cộng hai số nguyên có dấu n-bit, kết quả nhận được là n-bit
- Cộng hai số khác dấu: kết quả luôn luôn đúng.
- Cộng hai số cùng dấu:
  - Nếu dấu kết quả cùng dấu với các số hạng thì kết quả là đúng .
  - nếu kết quả có dấu ngược lại, khi đó có tràn xảy ra (Overflow) và kết quả bị sai.
- Tràn xảy ra khi tổng nằm ngoài dải biểu diễn  $[-2^{n-1}, 2^{n-1}-1]$

# Bộ Trừ Nhị phân n-bit

---

- Nguyên tắc:  $X - Y = X + (-Y)$
- Thêm bộ bù 2 vào bộ cộng

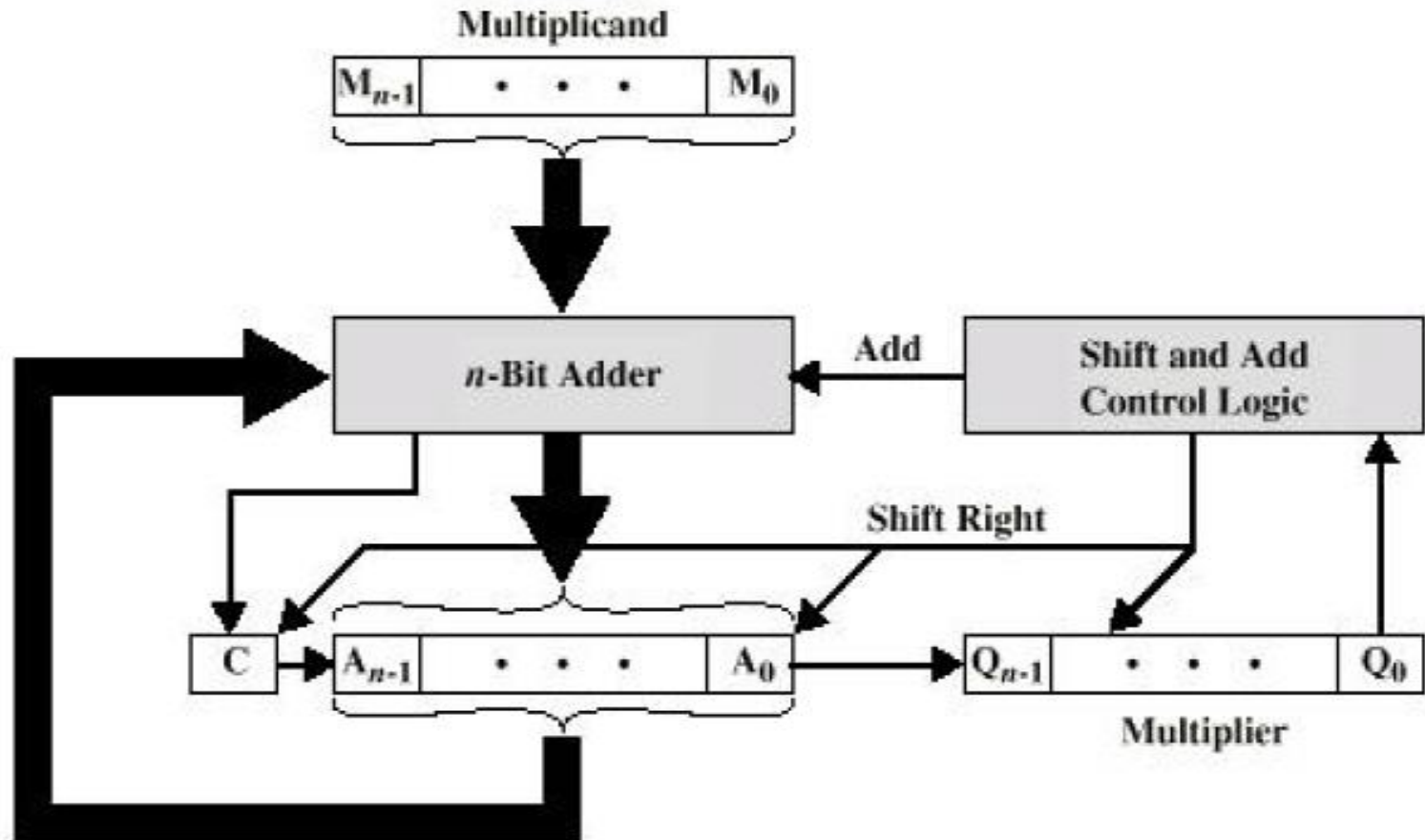


# Nhân Nhị phân không dấu n-bit

---

- Các tích riêng phần được xác định như sau:
  - Nếu bit của số nhân bằng 0  $\rightarrow$  tích riêng phần bằng 0.
  - Nếu bit của số nhân bằng 1  $\rightarrow$  tích riêng phần bằng số bị nhân.
  - Tích riêng phần tiếp theo được dịch trái một bit so với tích riêng phần trước đó.
- Tích bằng tổng các tích riêng phần
- Nhân hai số nguyên n-bit, tích có độ dài  $2n$  bit (không bao giờ tràn).

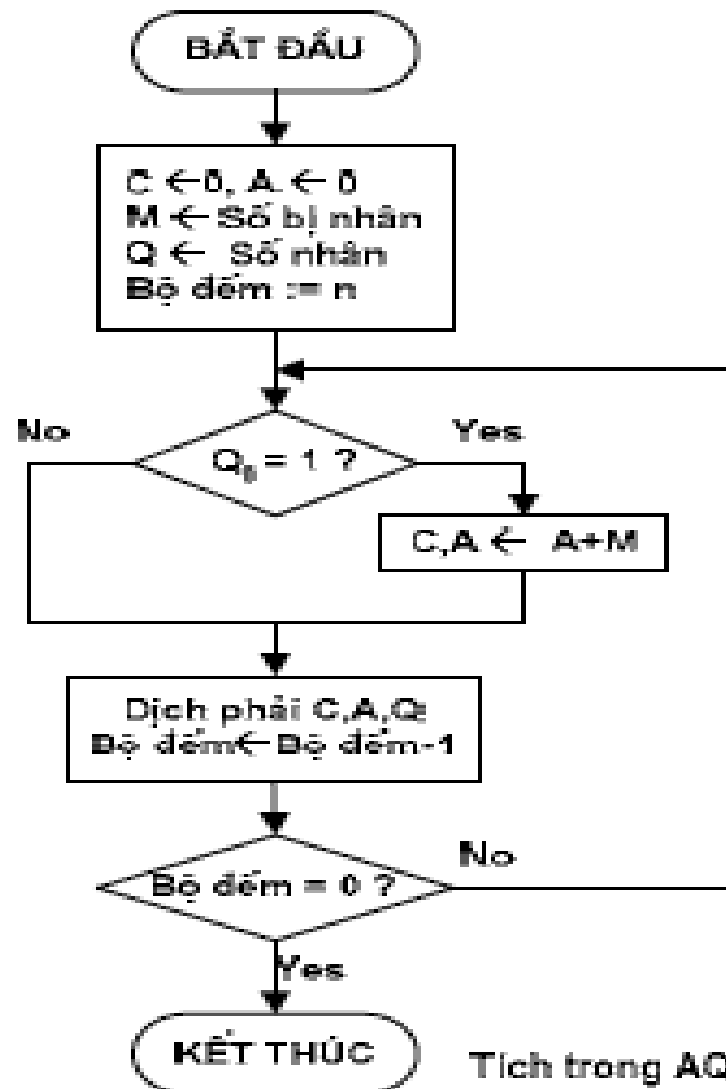
# Nhân Nhị phân không dấu n-bit



(a) Block Diagram

**FIG-8a**

# Nhân Nhị phân không dấu (lưu đồ)



Tích trong AQ

# Nhân Nhị phân không dấu n-bit

---

- Số bị nhân  $M = 1011$  (11)
- Số nhân  $Q = 1101$  (13)
- Tích =  $10001111$  (143)

C	A	Q	
0	0000	1101	Các giá trị khởi đầu
	+ 1011		
0	1011	1101	$A \leftarrow A + M$
0	0101	1110	Dịch phải
0	0010	1111	Dịch phải
	+ 1011		
0	1101	1111	$A \leftarrow A + M$
0	0110	1111	Dịch phải
	+ 1011		
1 0001	1111		$A \leftarrow A + M$
0 1000	1111		Dịch phải

# Nhân Nhị phân có dấu n-bit

---

- Hai thuật toán chủ yếu:
  - Dùng giải thuật nhân không dấu
  - Booth multiplication algorithm



# Nhân Nhị phân có dấu n-bit

---

Dùng giải thuật nhân không dấu

□ Bước 1. Chuyển đổi số bị nhân và số nhân thành số dương tương ứng

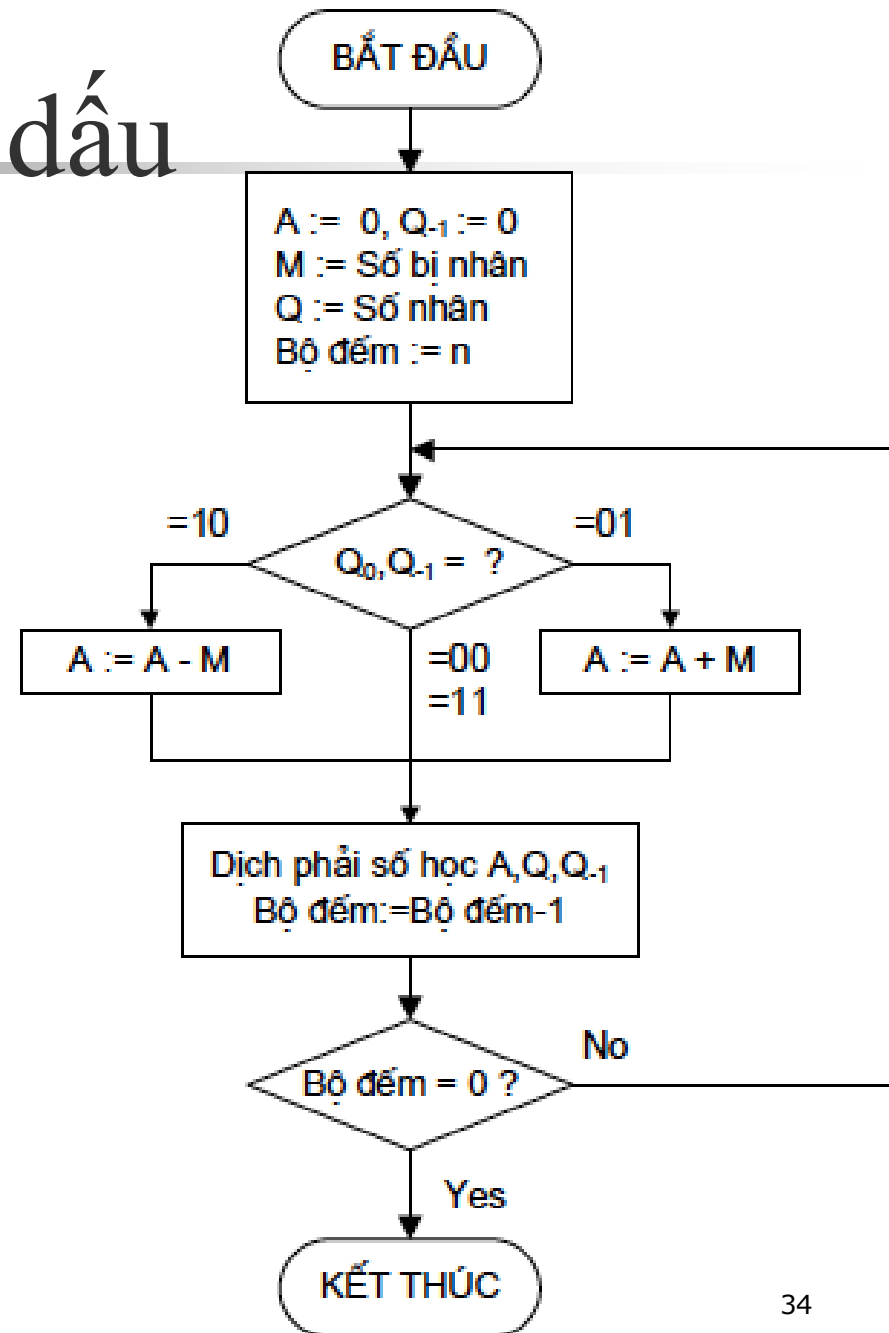
□ Bước 2. Nhân hai số dương bằng thuật giải nhân số nguyên không dấu, được tích của hai số dương.

□ Bước 3. Hiệu chỉnh dấu của tích:

- Nếu hai thừa số ban đầu cùng dấu thì giữ nguyên kết quả ở bước 2.
- Nếu hai thừa số ban đầu là khác dấu thì đảo dấu kết quả của bước 2.

# Nhân Nhị phân có dấu n-bit

## Booth Multiplication Algorithm



# Nhân Nhị Phân có dấu

□ Ex:  $3 \times 7 = 21$

$A = 0000, Q = 0011, M = 0111$

A	Q	Q <sub>-1</sub>	M	Initial Values	
0000	0011	0	0111		
1001	0011	0	0111	$A \leftarrow A - M$ Shift	} First Cycle
1100	1001	1	0111		
1110	0100	1	0111	Shift	} Second Cycle
0101	0100	1	0111	$A \leftarrow A + M$ Shift	} Third Cycle
0010	1010	0	0111		
0001	0101	0	0111	Shift	} Fourth Cycle

# Ví dụ nhân có dấu (booth algorithm)

$  \begin{array}{r}  0111 \\  \times 0011 \\  \hline  11111001 \\  00000000 \\  000111 \\  \hline  00010101  \end{array}  $	$  \begin{array}{r}  (0) \\  1-0 \\  1-1 \\  0-1 \\  (21)  \end{array}  $
$  \begin{array}{r}  0111 \\  \times 1101 \\  \hline  11111001 \\  0000111 \\  111001 \\  \hline  11101011  \end{array}  $	$  \begin{array}{r}  (0) \\  1-0 \\  0-1 \\  1-0 \\  (-21)  \end{array}  $

(a)  $(7) \times (3) = (21)$

(b)  $(7) \times (-3) = (-21)$

$  \begin{array}{r}  1001 \\  \times 0011 \\  \hline  00000111 \\  00000000 \\  111001 \\  \hline  11101011  \end{array}  $	$  \begin{array}{r}  (0) \\  1-0 \\  1-1 \\  0-1 \\  (-21)  \end{array}  $
$  \begin{array}{r}  1001 \\  \times 1101 \\  \hline  00000111 \\  1111001 \\  000111 \\  \hline  00010101  \end{array}  $	$  \begin{array}{r}  (0) \\  1-0 \\  0-1 \\  1-0 \\  (21)  \end{array}  $

(c)  $(-7) \times (3) = (-21)$

(d)  $(-7) \times (-3) = (21)$

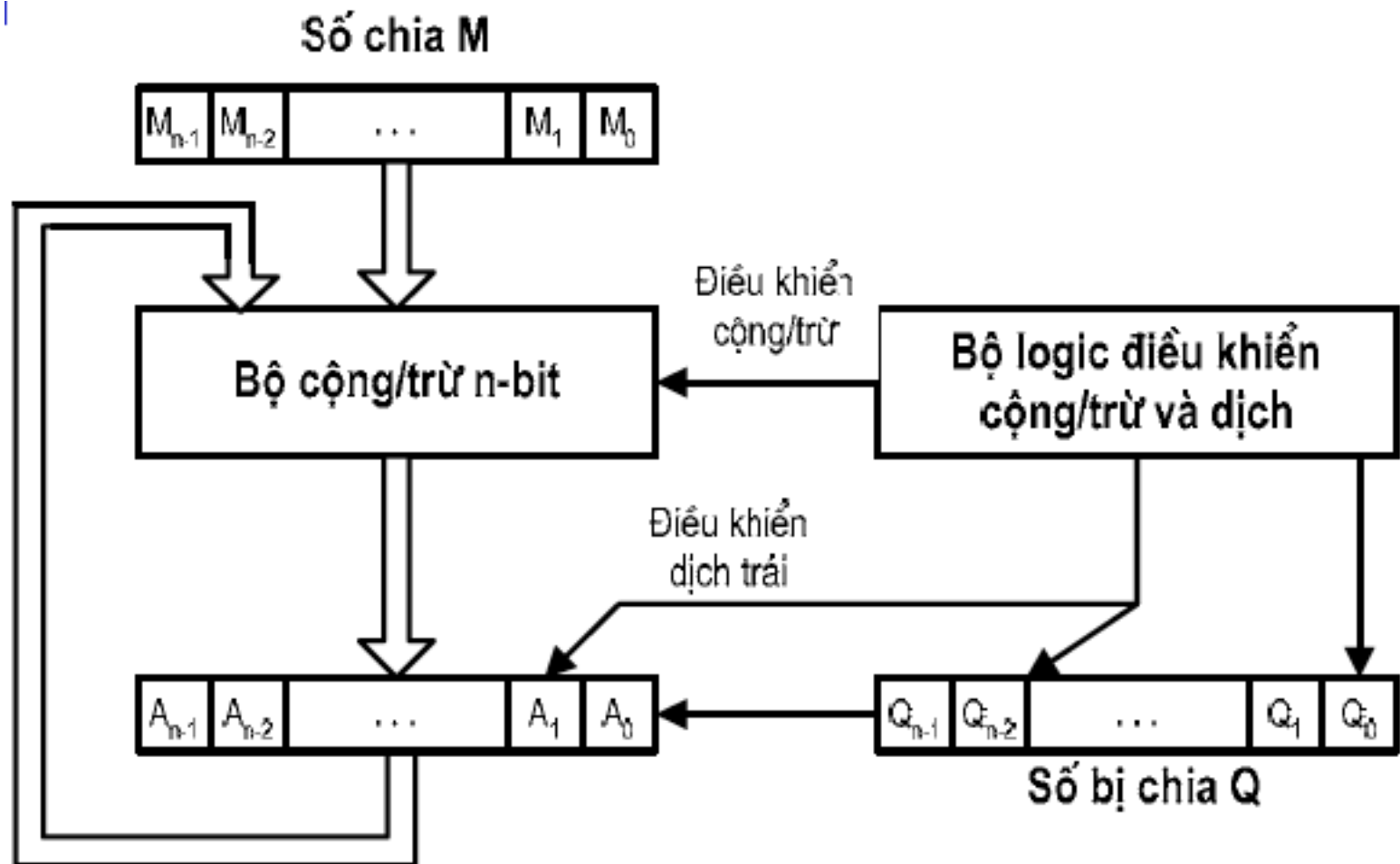
**Figure 10.14 Examples Using Booth's Algorithm**

# Chia nhị phân không dấu n-bit

---

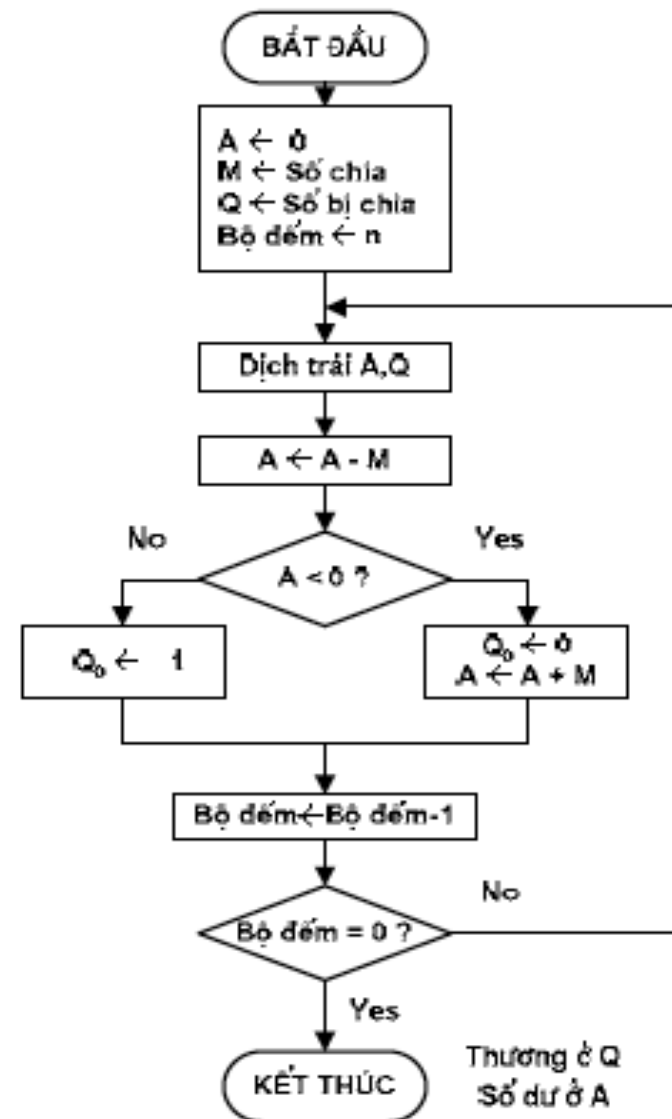
Số bị chia	10010011	$\overline{) 1011}$	Số chia
	<u>1011</u>	00001101	Thương
	001110		
	<u>1011</u>		
	001111		
	<u>1011</u>		
	100		Phần dư

# Chia nhị phân không dấu n-bit

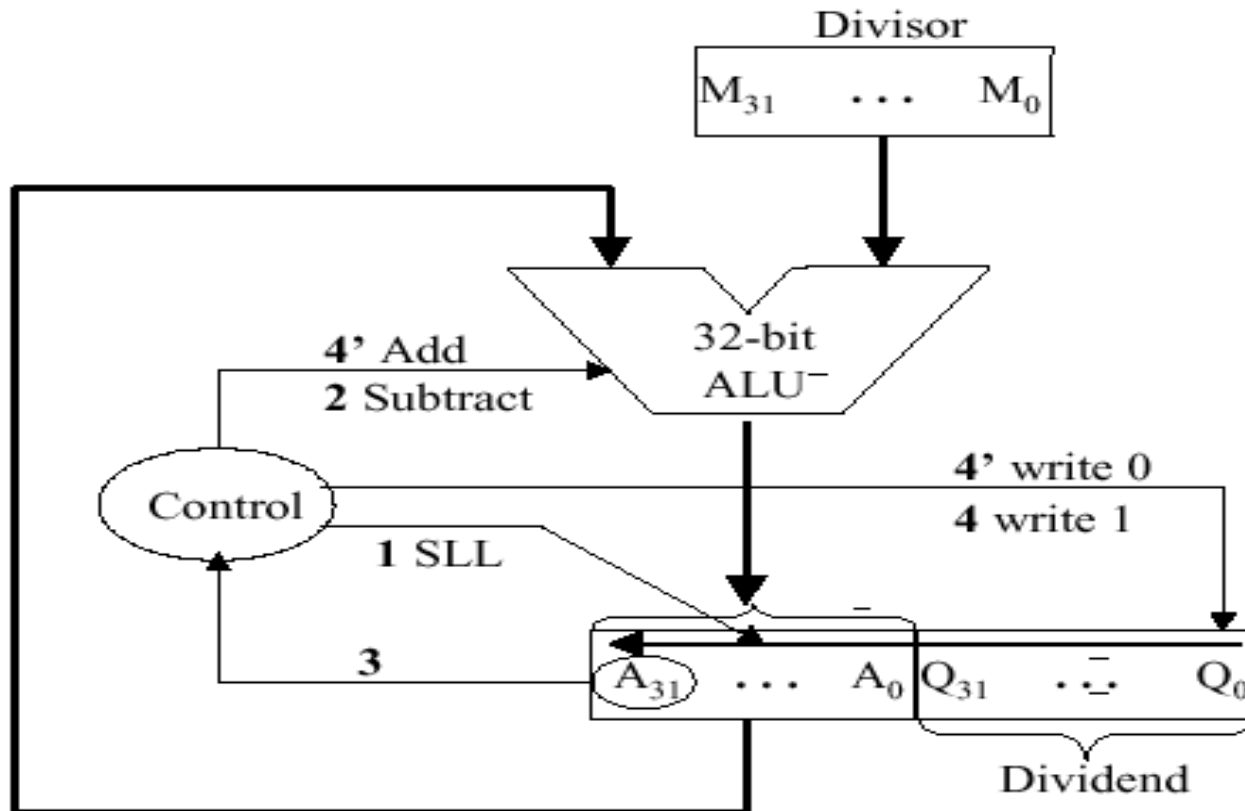


# Chia nhị phân không dấu n-bit (ví dụ: $7/3=2$ dư 1)

A	Q	M = 0011	
0000	0111	Initial values	
0000	1110	Shift	1
1101		$A = A - M$	
0000	1110	$A = A + M$	
0001	1100	Shift	2
1110		$A = A - M$	
0001	1100	$A = A + M$	
0011	1000	Shift	3
0000		$A = A - M$	
0000	1001	$Q_0 = 1$	
0001	0010	Shift	4
1110		$A = A - M$	
0001	0010	$A = A + M$	



# Chia nhị phân không dấu n-bit





# Nội dung Bài tập chương 2

---

1. Đại số Boole
2. Các cổng logic
3. Mạch tổ hợp
4. Mạch dãy



# 1. Đại số Boole

---

- Đại số Boole sử dụng các biến logic và phép toán logic
- Biến logic có thể nhận giá trị 1 (TRUE) hoặc 0 (FALSE)
- Phép toán logic cơ bản là AND, OR và NOT với ký hiệu như sau:
  - $A \text{ AND } B : A \cdot B$
  - $A \text{ OR } B : A + B$
  - $\text{NOT } A : \bar{A}$
- Thứ tự ưu tiên:  $\text{NOT} > \text{AND} > \text{OR}$



# Các phép toán logic (tiếp)

---

## ■ Các phép toán NAND, NOR, XOR:

■ A NAND B:  $\overline{A \bullet B}$

■ A NOR B:  $\overline{A + B}$

■ A XOR B:  $A \oplus B = A \bullet \bar{B} + \bar{A} \bullet B$



# Phép toán đại số Boole

A	B	NOT A $\overline{A}$	A AND B $A \cdot B$	A OR B $A + B$	A NAND B $\overline{A \cdot B}$	A NOR B $\overline{A + B}$	A XOR B $A \oplus B$
0	0	1	0	0	1	1	0
0	1	1	0	1	1	0	1
1	0	0	0	1	1	0	1
1	1	0	1	1	0	0	0



# Các đồng nhất thức của đại số Boole

$$A \cdot B = B \cdot A$$

$$A \cdot (B + C) = (A \cdot B) + (A \cdot C)$$

$$1 \cdot A = A$$

$$A \cdot \bar{A} = 0$$

$$0 \cdot A = 0$$

$$A \cdot A = A$$

$$A \cdot (B \cdot C) = (A \cdot B) \cdot C$$

$$\overline{A \cdot B} = \bar{A} + \bar{B} \text{ (Định lý De Morgan)}$$

$$A + B = B + A$$

$$A + (B \cdot C) = (A + B) \cdot (A + C)$$

$$0 + A = A$$

$$A + \bar{A} = 1$$

$$1 + A = 1$$

$$A + A = A$$

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

$$\overline{A + B} = \bar{A} \cdot \bar{B} \text{ (Định lý De Morgan)}$$

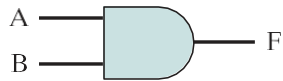
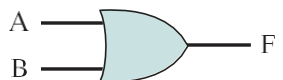
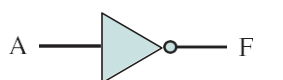

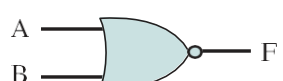



# Các cổng logic (Logic Gates)

---

- Thực hiện các hàm logic:
  - NOT, AND, OR, NAND, NOR, etc.
- Cổng logic một đầu vào:
  - Cổng NOT, bộ đệm (buffer)
- Cổng hai đầu vào:
  - AND, OR, XOR, NAND, NOR, XNOR
- Cổng nhiều đầu vào

# Các cổng logic

Name	Graphical Symbol	Algebraic Function	Truth Table															
AND		$F = A \bullet B$ or $F = AB$	<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	F	0	0	0	0	1	0	1	0	0	1	1	1
A	B	F																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OR		$F = A + B$	<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	F	0	0	0	0	1	1	1	0	1	1	1	1
A	B	F																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
NOT		$F = \overline{A}$ or $F = A'$	<table><tr><th>A</th><th>F</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	F	0	1	1	0									
A	F																	
0	1																	
1	0																	
NAND		$F = \overline{AB}$	<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	F	0	0	1	0	1	1	1	0	1	1	1	0
A	B	F																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
NOR		$F = \overline{A + B}$	<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	F	0	0	1	0	1	0	1	0	0	1	1	0
A	B	F																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
XOR		$F = A \oplus B$	<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	F	0	0	0	0	1	1	1	0	1	1	1	0
A	B	F																
0	0	0																
0	1	1																
1	0	1																
1	1	0																



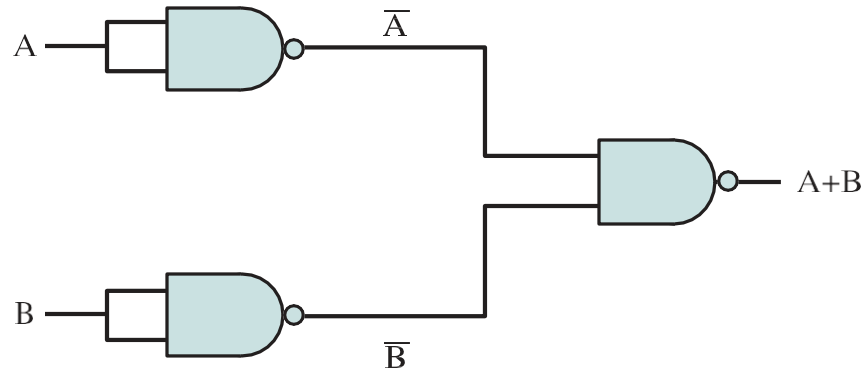
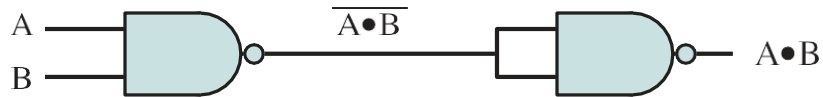
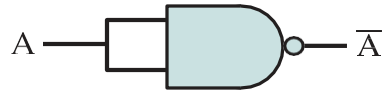
# Tập đầy đủ

---

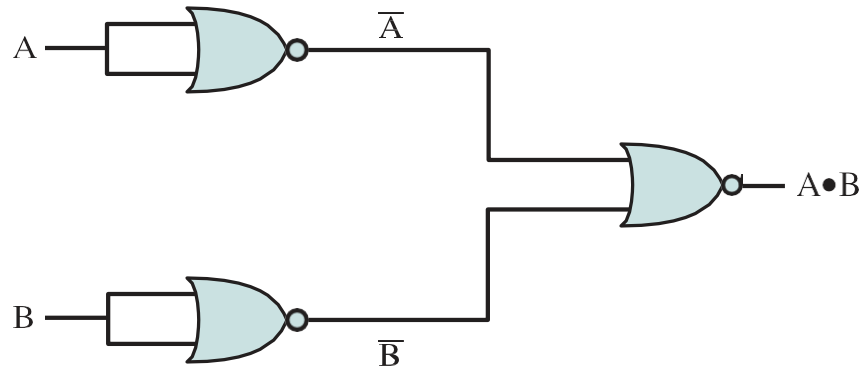
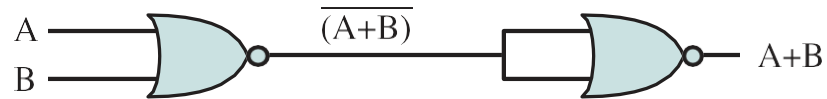
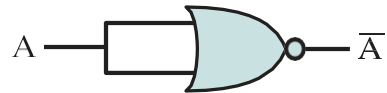
- Là tập các cổng có thể thực hiện được bất kỳ hàm logic nào từ các cổng của tập đó.
- Một số ví dụ về tập đầy đủ:
  - {AND, OR, NOT}
  - {AND, NOT}
  - {OR, NOT}
  - {NAND}
  - {NOR}



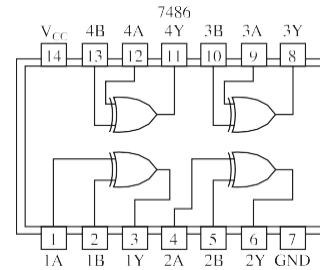
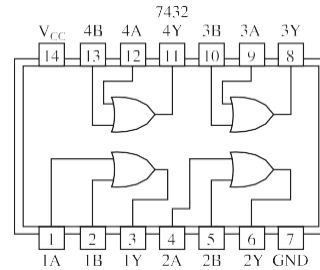
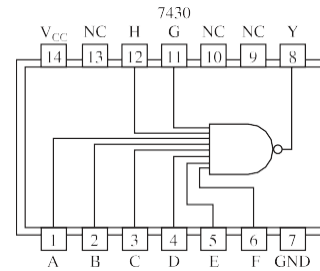
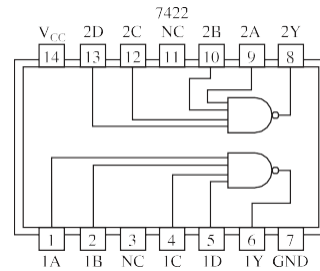
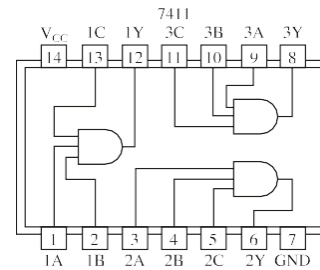
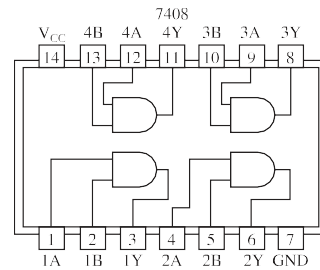
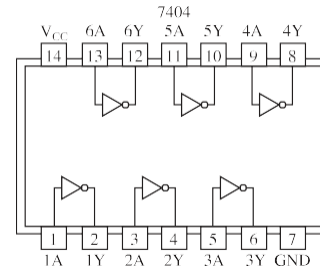
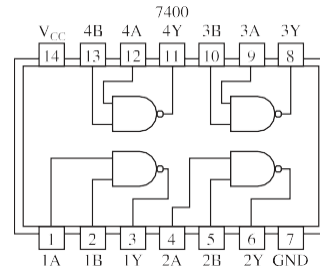
# Sử dụng cổng NAND



# Sử dụng cổng NOR



# Một số ví dụ vi mạch logic





# Mạch tổ hợp

---

- Mạch logic là mạch bao gồm:
  - Các đầu vào (Inputs)
  - Các đầu ra (Outputs)
  - Đặc tả chức năng (Functional specification)
  - Đặc tả thời gian (Timing specification)
- Các kiểu mạch logic:
  - Mạch logic tổ hợp (Combinational Logic)
    - Mạch không nhớ
    - Đầu ra được xác định bởi các giá trị hiện tại của đầu vào
  - Mạch logic dãy (Sequential Logic)
    - Mạch có nhớ
    - Đầu ra được xác định bởi các giá trị trước đó và giá trị hiện tại của đầu vào



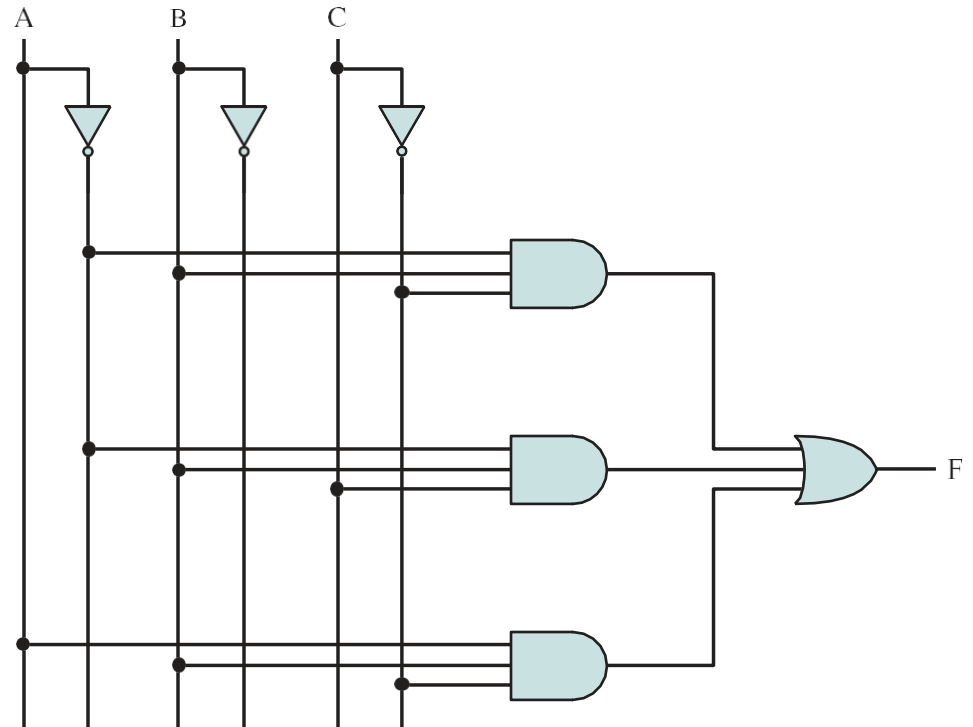
# Mạch tổ hợp

---

- Mạch tổ hợp là mạch logic trong đó đầu ra chỉ phụ thuộc đầu vào ở thời điểm hiện tại.
- Là mạch không nhớ và được thực hiện bằng các cổng logic cơ bản
- Mạch tổ hợp có thể được định nghĩa theo ba cách:
  - Bảng thật
  - Dạng sơ đồ
  - Phương trình Boole

# Ví dụ

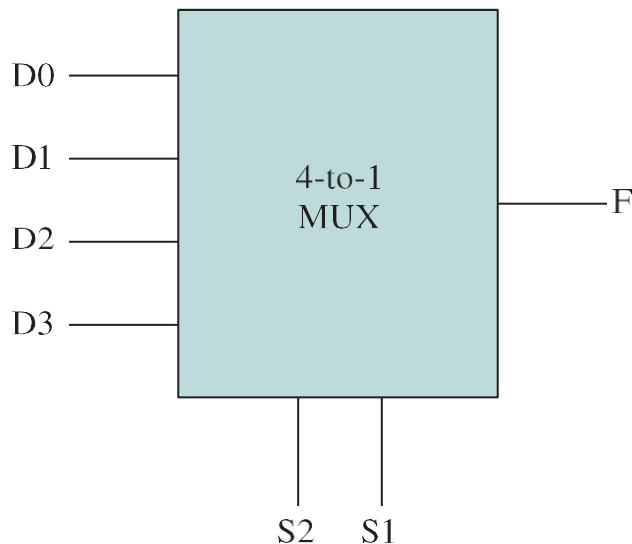
A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0



$$F = \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}\bar{C}$$

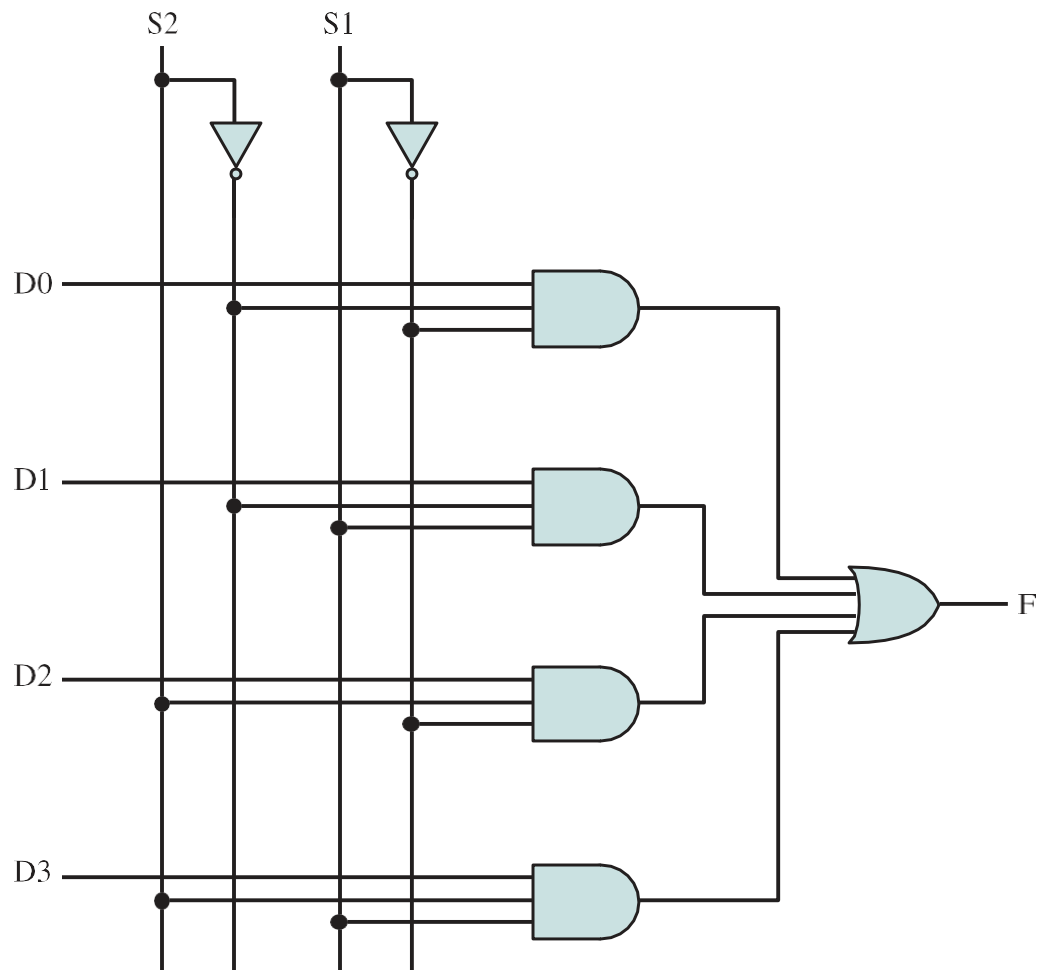
# Bộ dồn kênh (Multiplexer-MUX)

- $2^n$  đầu vào dữ liệu
- $n$  đầu vào chọn
- 1 đầu ra
- Đầu vào chọn (S) xác định đầu vào nào (D) sẽ được nối với đầu ra (F).



S2	S1	F
0	0	D0
0	1	D1
1	0	D2
1	1	D3

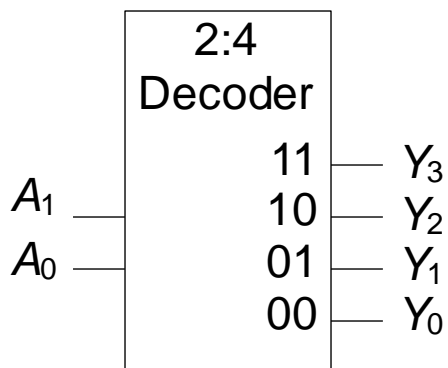
# Thực hiện MUX bốn đầu vào



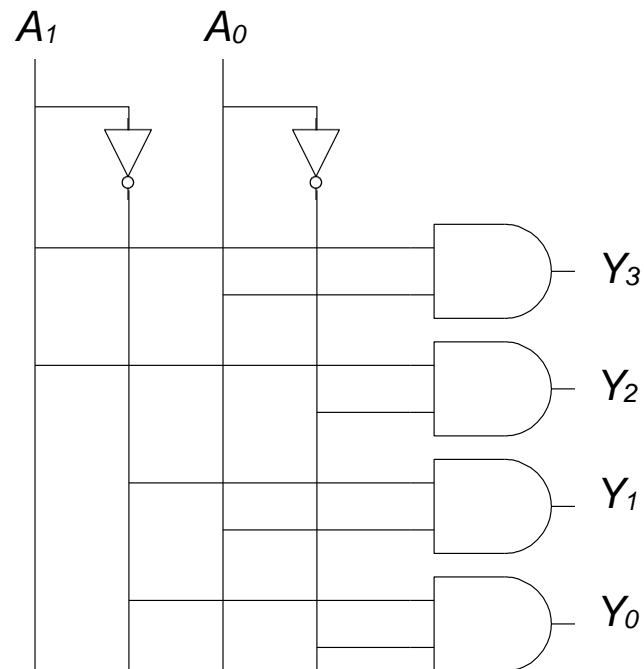


# Bộ giải mã (Decoder)

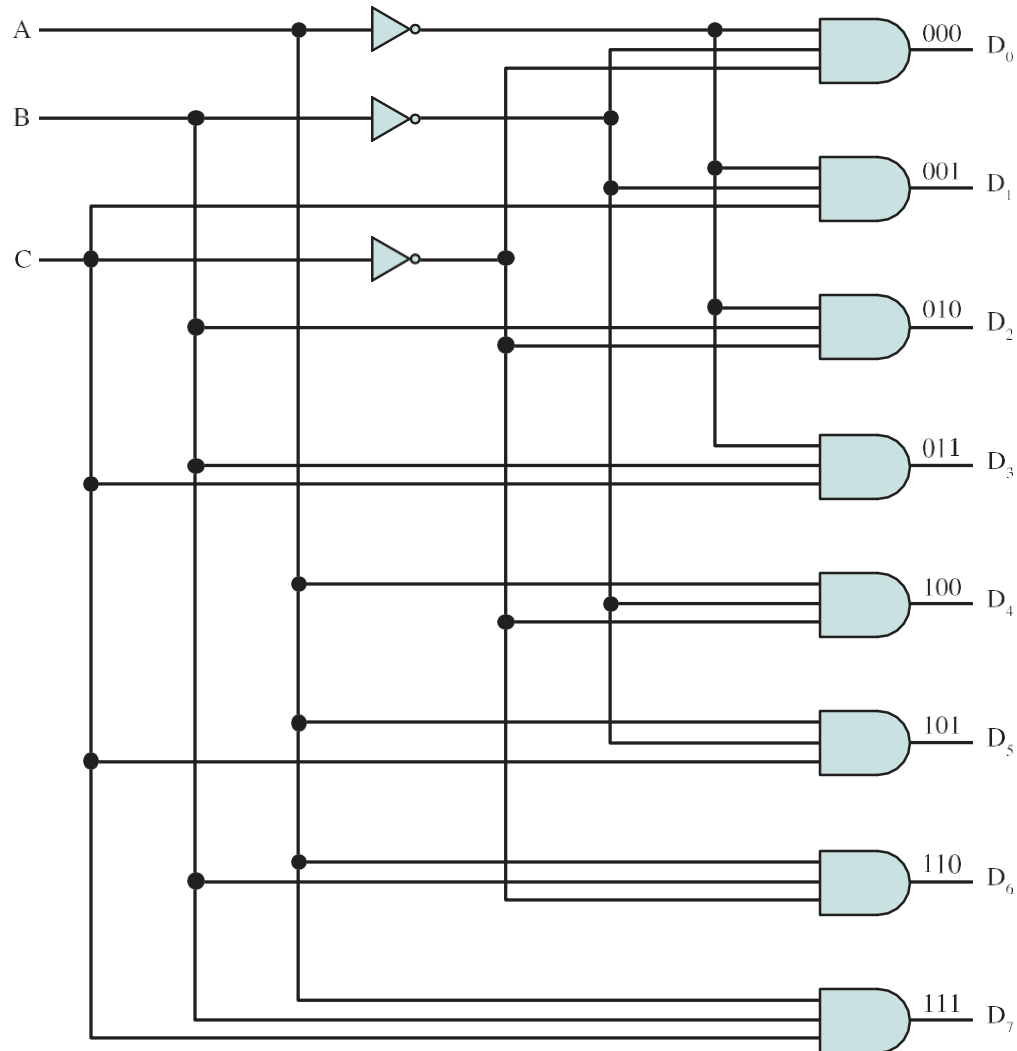
- $N$  đầu vào,  $2^N$  đầu ra
- Chỉ có một đầu ra tích cực (được chọn) tương ứng với một tổ hợp của  $N$  đầu vào.



$A_1$	$A_0$	$Y_3$	$Y_2$	$Y_1$	$Y_0$
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0



# Thực hiện bộ giải mã 3 ra 8





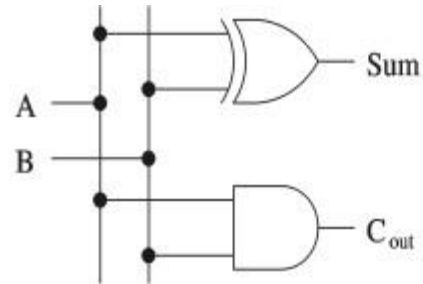
# Bộ cộng (Adder)

---

- Bộ cộng bán phần (Half-adder)
  - Cộng hai bit tạo ra bit tổng và bit nhớ
- Bộ cộng toàn phần (Full-adder)
  - Cộng 3 bit
  - Cho phép xây dựng bộ cộng N-bit

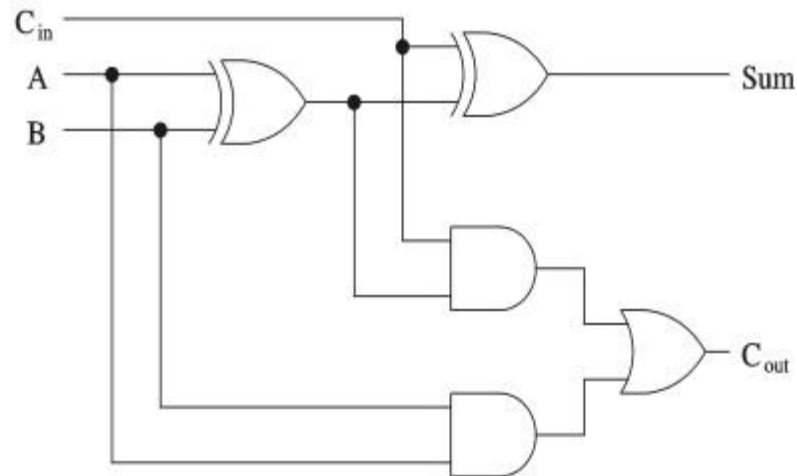
# Bộ cộng (tiếp)

A	B	Sum	C <sub>out</sub>
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

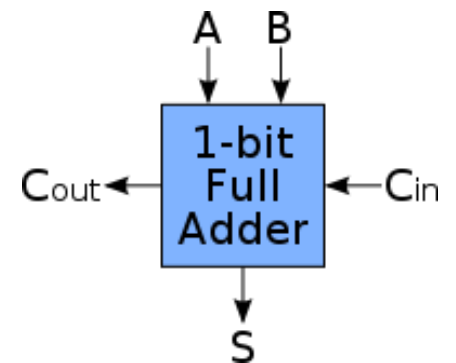


(a) Half-adder truth table and implementation

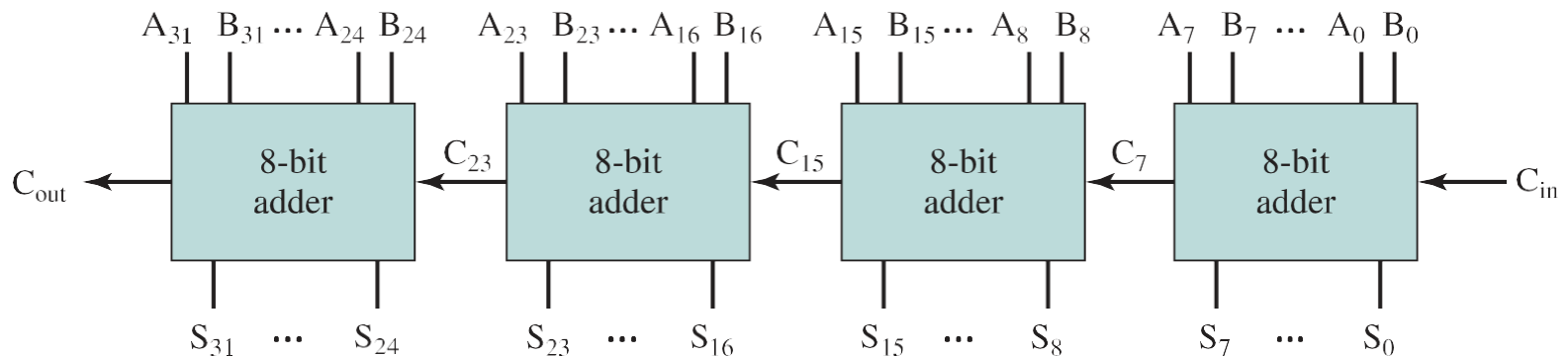
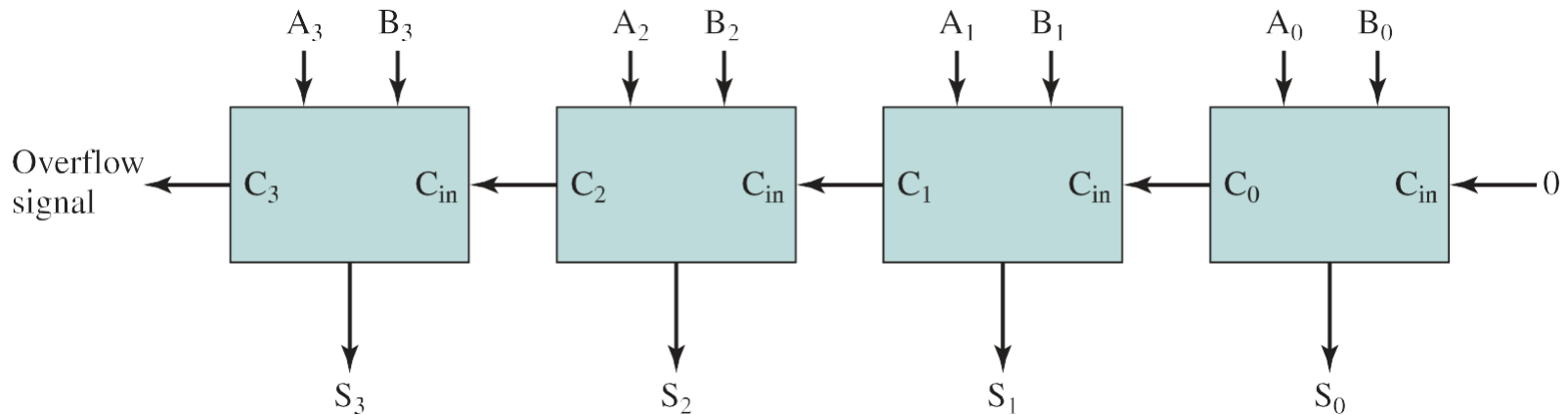
A	B	C <sub>in</sub>	Sum	C <sub>out</sub>
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



(b) Full-adder truth table and implementation



# Bộ cộng 4-bit và bộ cộng 32-bit



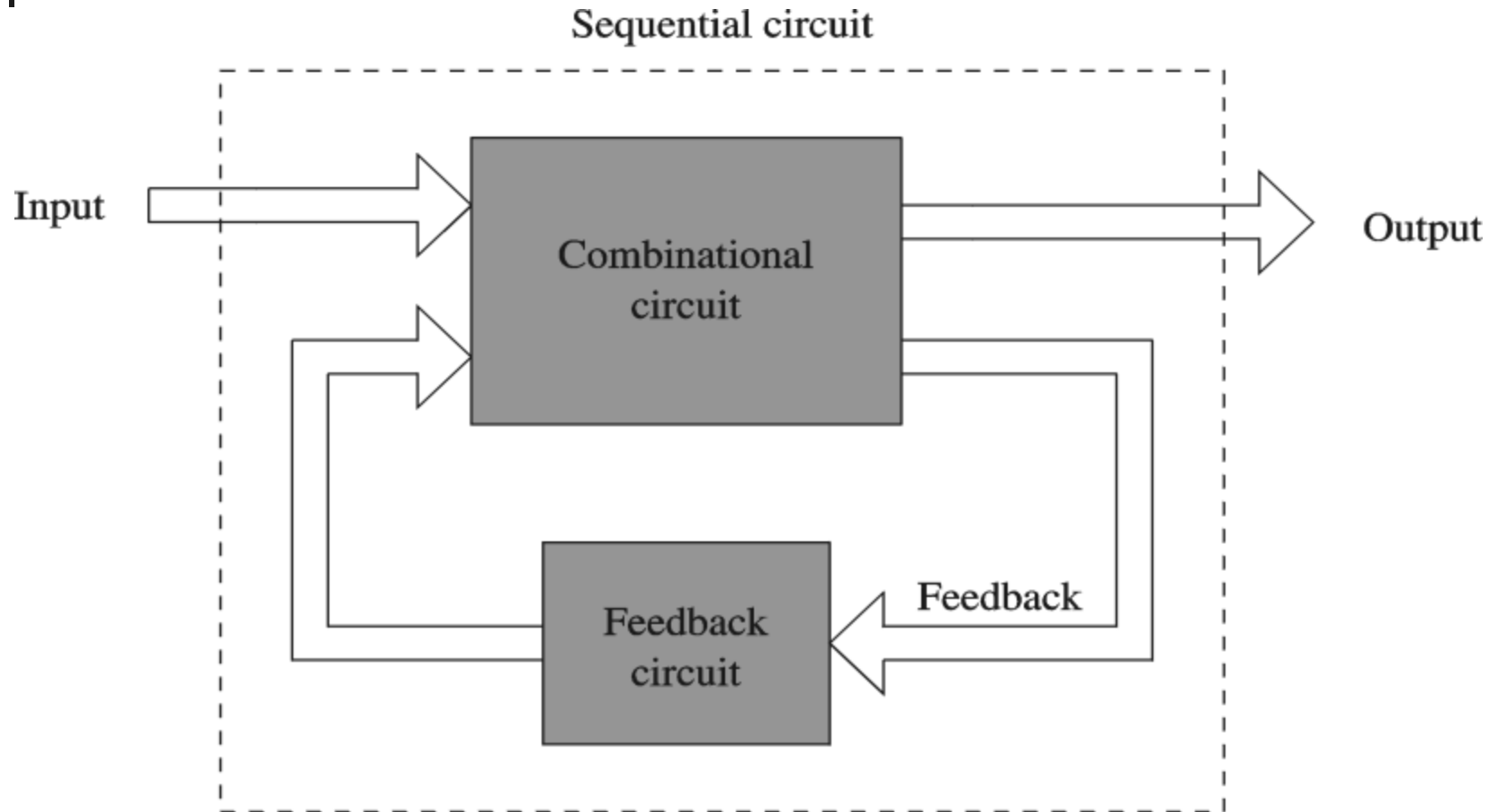


# Mạch dãy

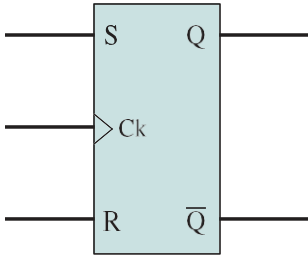
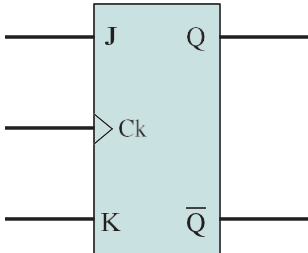
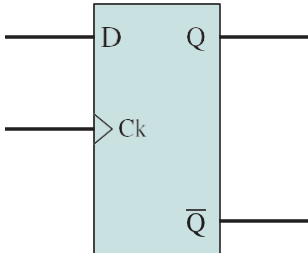
---

- Mạch dãy là mạch logic trong đó đầu ra phụ thuộc giá trị đầu vào ở thời điểm hiện tại và quá khứ
- Là mạch có nhớ, được thực hiện bằng phần tử nhớ (Latch, Flip-Flop) và có thể kết hợp với các cổng logic cơ bản
- Mạch dãy bao gồm:
  - Mạch tổ hợp
  - Mạch hồi tiếp

# Các thành phần chính của mạch dãy

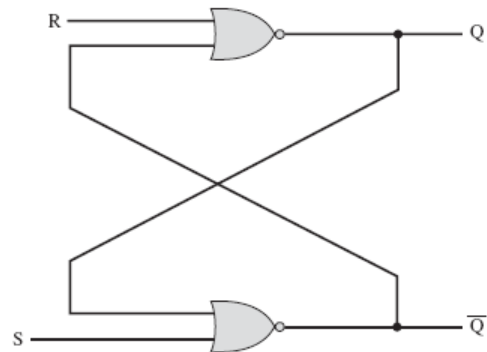


# Các Flip-Flop cơ bản

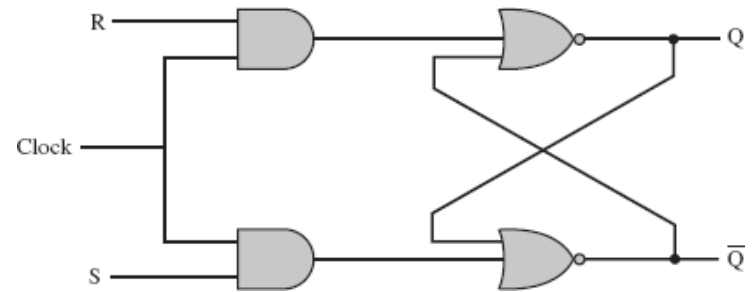
Name	Graphical Symbol	Truth Table															
S-R		<table> <tr> <th>S</th> <th>R</th> <th><math>Q_{n+1}</math></th> </tr> <tr> <td>0</td> <td>0</td> <td><math>Q_n</math></td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>—</td> </tr> </table>	S	R	$Q_{n+1}$	0	0	$Q_n$	0	1	0	1	0	1	1	1	—
S	R	$Q_{n+1}$															
0	0	$Q_n$															
0	1	0															
1	0	1															
1	1	—															
J-K		<table> <tr> <th>J</th> <th>K</th> <th><math>Q_{n+1}</math></th> </tr> <tr> <td>0</td> <td>0</td> <td><math>Q_n</math></td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td><math>\overline{Q_n}</math></td> </tr> </table>	J	K	$Q_{n+1}$	0	0	$Q_n$	0	1	0	1	0	1	1	1	$\overline{Q_n}$
J	K	$Q_{n+1}$															
0	0	$Q_n$															
0	1	0															
1	0	1															
1	1	$\overline{Q_n}$															
D		<table> <tr> <th>D</th> <th><math>Q_{n+1}</math></th> </tr> <tr> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> </tr> </table>	D	$Q_{n+1}$	0	0	1	1									
D	$Q_{n+1}$																
0	0																
1	1																



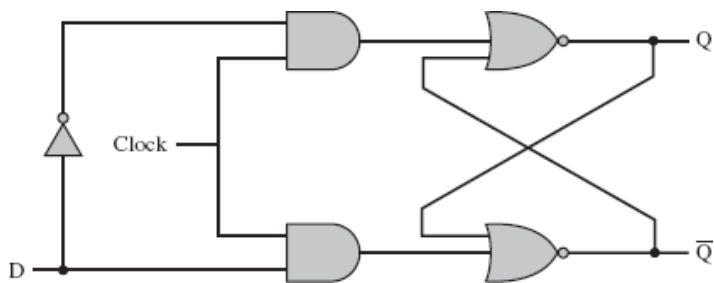
# R-S Latch và các Flip-Flop



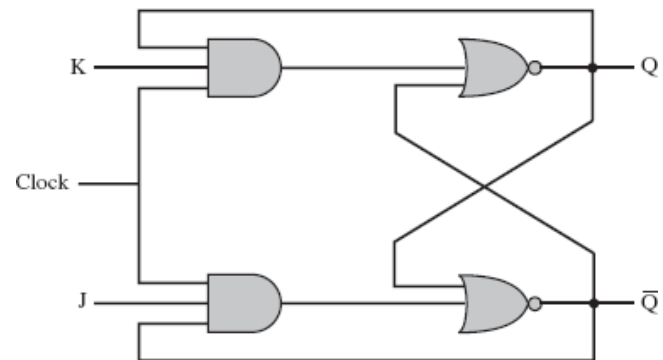
R-S Latch



R-S Flip Flop

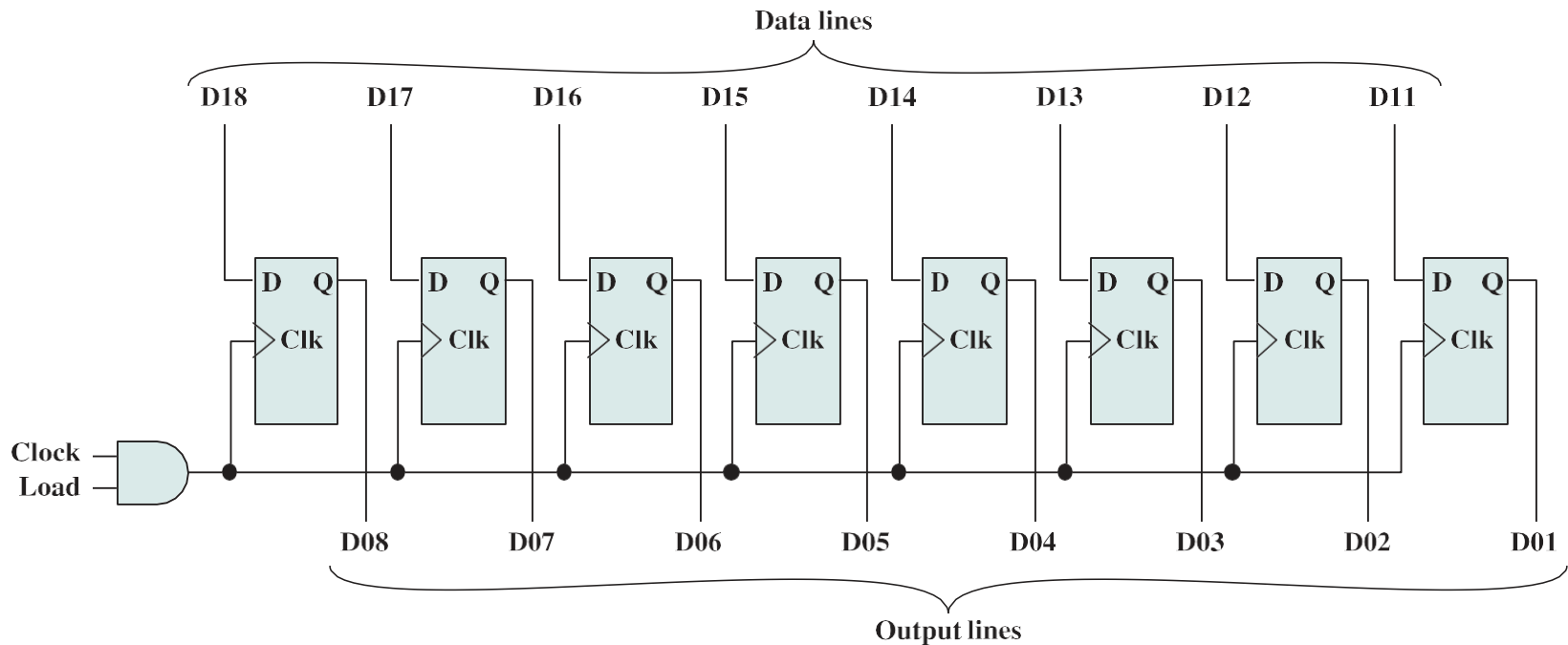


D Flip Flop

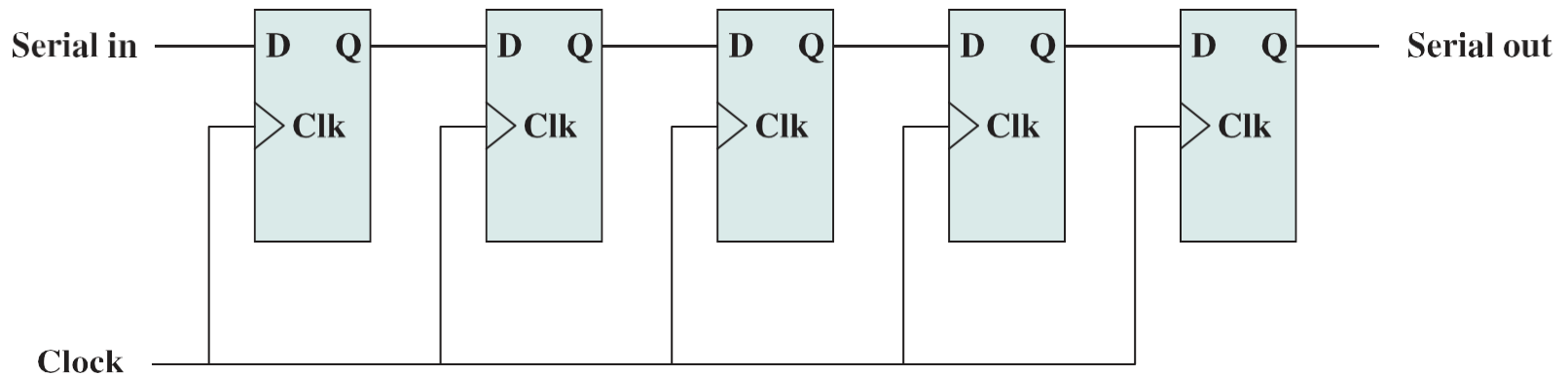


J-K Flip Flop

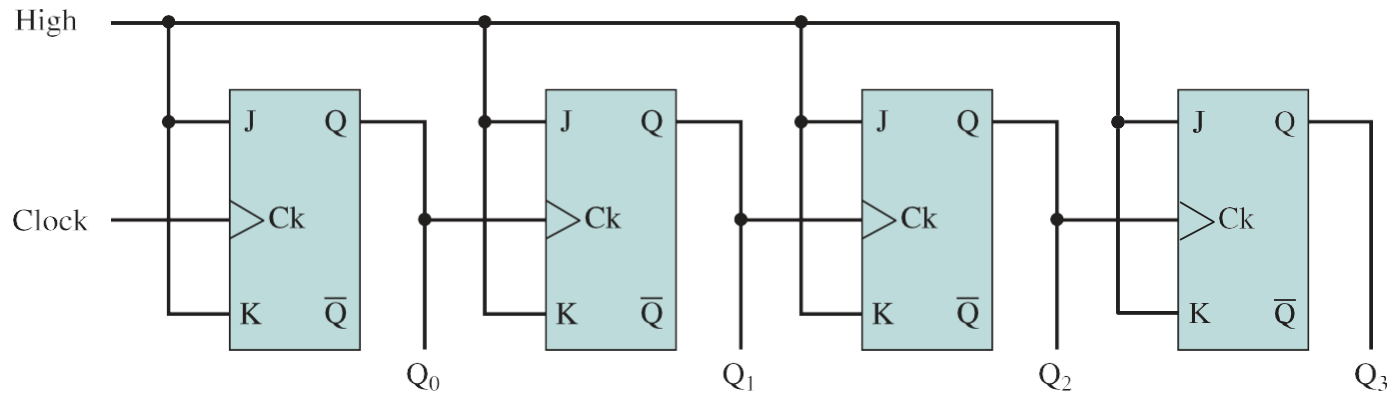
# Thanh ghi 8-bit song song



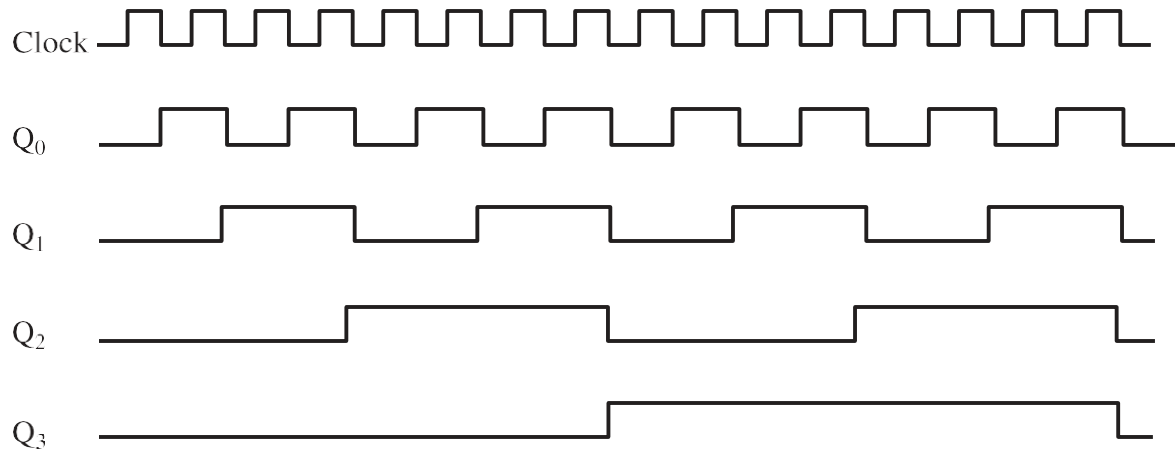
# Thanh ghi dịch 5-bit



# Bộ đếm 4-bit



(a) Sequential circuit



(b) Timing diagram

# Bài tập chương 2

---

Hãy vẽ sơ đồ mạch logic thực hiện phương trình Boole sau:

$$Y = ((\bar{A} + B).C) \oplus (\overline{B + C})$$

1. Vẽ sơ đồ mạch logic với 3 cổng AND, OR, NOT
2. Vẽ sơ đồ mạch với 1 loại cổng logic NAND
3. Vẽ sơ đồ mạch với 1 loại cổng NOR