

Chương 2

CƠ BẢN VỀ LOGIC SỐ

Nguyễn Kim Khánh

Trường Đại học Bách khoa Hà Nội

Nội dung học phần

Chương 1. Giới thiệu chung

Chương 2. Cơ bản về logic số

Chương 3. Hệ thống máy tính

Chương 4. Số học máy tính

Chương 5. Kiến trúc tập lệnh

Chương 6. Bộ xử lý

Chương 7. Bộ nhớ máy tính

Chương 8. Hệ thống vào-ra

Chương 9. Các kiến trúc song song



Nội dung của chương 2

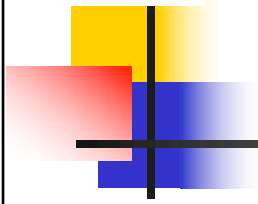
- 2.1. Các hệ đếm cơ bản
- 2.2. Đại số Boole
- 2.3. Các cổng logic
- 2.4. Mạch tổ hợp
- 2.5. Mạch dãy

2.1. Các hệ đếm cơ bản

- Hệ thập phân (Decimal System)
→ con người sử dụng
- Hệ nhị phân (Binary System)
→ máy tính sử dụng
- Hệ mười sáu (Hexadecimal System)
→ dùng để viết gọn cho số nhị phân

1. Hệ thập phân

- Cơ số 10
- 10 chữ số: 0,1,2,3,4,5,6,7,8,9
- Dùng n chữ số thập phân có thể biểu diễn được 10^n giá trị khác nhau:
 - $00\dots000 = 0$
 - $99\dots999 = 10^n - 1$



Dạng tổng quát của số thập phân

$$A = a_n a_{n-1} \dots a_1 a_0, a_{-1} \dots a_{-m}$$

Giá trị của A được hiểu như sau:

$$A = a_n 10^n + a_{n-1} 10^{n-1} + \dots + a_1 10^1 + a_0 10^0 + a_{-1} 10^{-1} + \dots + a_{-m} 10^{-m}$$

$$A = \sum_{i=-m}^n a_i 10^i$$

Ví dụ số thập phân

$$472.38 = 4 \times 10^2 + 7 \times 10^1 + 2 \times 10^0 + 3 \times 10^{-1} + 8 \times 10^{-2}$$

■ Các chữ số của phần nguyên:

- $472 : 10 = 47$ dư 2
- $47 : 10 = 4$ dư 7
- $4 : 10 = 0$ dư 4



■ Các chữ số của phần lẻ:

- $0.38 \times 10 = 3.8$ phần nguyên = 3
- $0.8 \times 10 = 8.0$ phần nguyên = 8



2. Hệ nhị phân

- Cơ số 2
- 2 chữ số nhị phân: 0 và 1
- Chữ số nhị phân được gọi là **bit** (*binary digit*)
- **bit** là đơn vị thông tin nhỏ nhất
- Dùng n bit có thể biểu diễn được 2^n giá trị khác nhau:
 - $00...000 = 0$
 - $11...111 = 2^n - 1$
- Các lệnh của chương trình và dữ liệu trong máy tính đều được mã hóa bằng số nhị phân

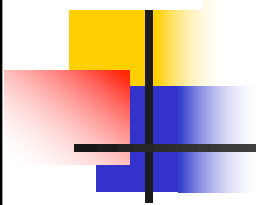


Biểu diễn số nhị phân

Số nhị phân				Số thập phân
1-bit	2-bit	3-bit	4-bit	
0	00	000	0000	0
1	01	001	0001	1
	10	010	0010	2
	11	011	0011	3
		100	0100	4
		101	0101	5
		110	0110	6
		111	0111	7
			1000	8
			1001	9
			1010	10
			1011	11
			1100	12
			1101	13
			1110	14
			1111	15

Đơn vị dữ liệu và thông tin trong máy tính

- **bit** – chữ số nhị phân (**b**inary **d**igit): là đơn vị thông tin nhỏ nhất, cho phép nhận một trong hai giá trị: 0 hoặc 1.
- **byte** là một tổ hợp 8 bit: có thể biểu diễn được 256 giá trị (2^8)
- **Qui ước các đơn vị dữ liệu:**
 - **KB** (Kilobyte) = 2^{10} bytes = 1024 bytes
 - **MB** (Megabyte) = 2^{10} KB = 2^{20} bytes ($\sim 10^6$)
 - **GB** (Gigabyte) = 2^{10} MB = 2^{30} bytes ($\sim 10^9$)
 - **TB** (Terabyte) = 2^{10} GB = 2^{40} bytes ($\sim 10^{12}$)
 - **PB** (Petabyte) = 2^{10} TB = 2^{50} bytes
 - **EB** (Exabyte) = 2^{10} PB = 2^{60} bytes



Qui ước mới về ký hiệu đơn vị dữ liệu

Theo thập phân			Theo nhị phân		
Đơn vị	Viết tắt	Giá trị	Đơn vị	Viết tắt	Giá trị
kilobyte	KB	10^3	kibibyte	KiB	$2^{10} = 1024$
megabyte	MB	10^6	mebibyte	MiB	2^{20}
gigabyte	GB	10^9	gibibyte	GiB	2^{30}
terabyte	TB	10^{12}	tebibyte	TiB	2^{40}
petabyte	PB	10^{15}	pebibyte	PiB	2^{50}
exabyte	EB	10^{18}	exbibyte	EiB	2^{60}

Dạng tổng quát của số nhị phân

$$A = a_n a_{n-1} \dots a_1 a_0, a_{-1} \dots a_{-m} \quad \text{với } a_i = 0 \text{ hoặc } 1$$

Giá trị của A được tính như sau:

$$A = a_n 2^n + a_{n-1} 2^{n-1} + \dots + a_1 2^1 + a_0 2^0 + a_{-1} 2^{-1} + \dots + a_{-m} 2^{-m}$$

$$A = \sum_{i=-m}^n a_i 2^i$$



Ví dụ số nhị phân

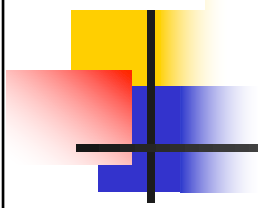
$$1101001.1011_{(2)} =$$

6 5 4 3 2 1 0 -1 -2 -3 -4

$$= 2^6 + 2^5 + 2^3 + 2^0 + 2^{-1} + 2^{-3} + 2^{-4}$$

$$= 64 + 32 + 8 + 1 + 0.5 + 0.125 + 0.0625$$

$$= 105.6875_{(10)}$$



Chuyển đổi số nguyên thập phân sang nhị phân

- Phương pháp 1: chia dần cho 2 rồi lấy phần dư
- Phương pháp 2: Phân tích thành tổng của các số 2^i → nhanh hơn

Phương pháp chia dần cho 2

■ Ví dụ: chuyển đổi $105_{(10)}$

■	$105 : 2 =$	52	dư	1
■	$52 : 2 =$	26	dư	0
■	$26 : 2 =$	13	dư	0
■	$13 : 2 =$	6	dư	1
■	$6 : 2 =$	3	dư	0
■	$3 : 2 =$	1	dư	1
■	$1 : 2 =$	0	dư	1

biểu diễn
số dư
theo chiều
mũi tên

■ Kết quả: $105_{(10)} = 1101001_{(2)}$

Phương pháp phân tích thành tổng của các 2^i

- Ví dụ 1: chuyển đổi $105_{(10)}$

- $105 = 64 + 32 + 8 + 1 = 2^6 + 2^5 + 2^3 + 2^0$

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
0	1	1	0	1	0	0	1

- Kết quả: $105_{(10)} = 0110\ 1001_{(2)}$

- Ví dụ 2: $17000_{(10)} = 16384 + 512 + 64 + 32 + 8$
 $= 2^{14} + 2^9 + 2^6 + 2^5 + 2^3$

$$17000_{(10)} = \underset{\substack{15\ 14\ 13\ 12\ 11\ 10\ 9\ 8\ 7\ 6\ 5\ 4\ 3\ 2\ 1\ 0}}{0100\ 0010\ 0110\ 1000}_{(2)}$$

Chuyển đổi số lẻ thập phân sang nhị phân

■ Ví dụ 1: chuyển đổi $0.6875_{(10)}$

- $0.6875 \times 2 = 1.375$ phần nguyên = 1
- $0.375 \times 2 = 0.75$ phần nguyên = 0
- $0.75 \times 2 = 1.5$ phần nguyên = 1
- $0.5 \times 2 = 1.0$ phần nguyên = 1

biểu diễn
theo
chiều
mũi tên

■ Kết quả : $0.6875_{(10)} = 0.1011_{(2)}$

Chuyển đổi số lẻ thập phân sang nhị phân (tiếp)

■ Ví dụ 2: chuyển đổi $0.81_{(10)}$

■	0.81	x 2 =	1.62	phần nguyên	=	1
■	0.62	x 2 =	1.24	phần nguyên	=	1
■	0.24	x 2 =	0.48	phần nguyên	=	0
■	0.48	x 2 =	0.96	phần nguyên	=	0
■	0.96	x 2 =	1.92	phần nguyên	=	1
■	0.92	x 2 =	1.84	phần nguyên	=	1
■	0.84	x 2 =	1.68	phần nguyên	=	1

■ $0.81_{(10)} \approx 0.1100111_{(2)}$

3. Hệ mười sáu (Hexa)

- Cơ số 16
- 16 chữ số: 0,1,2,3,4,5,6,7,8,9, A,B,C,D,E,F
- Dùng để viết gọn cho số nhị phân: cứ một nhóm 4-bit sẽ được thay bằng một chữ số Hexa

Quan hệ giữa số nhị phân và số Hexa

Ví dụ:

- $1011\ 0011_{(2)} = B3_{(16)}$
- $0000\ 0000_{(2)} = 00_{(16)}$
- $0010\ 1101\ 1001\ 1010_{(2)} = 2D9A_{(16)}$
- $1111\ 1111\ 1111\ 1111_{(2)} = FFFF_{(16)}$

4-bit	Số Hexa	Thập phân
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

2.2. 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)
- Các phép toán logic cơ bản: AND, OR và NOT
 - A AND B : $A \bullet B$ hay AB
 - A OR B : $A + B$
 - NOT A : \overline{A}
 - Thứ tự ưu tiên: NOT > AND > OR
- Thêm các phép toán logic: NAND, NOR, XOR
 - A NAND B: $\overline{A \bullet B}$
 - A NOR B : $\overline{A + B}$
 - A XOR B: $A \oplus B = A \bullet \overline{B} + \overline{A} \bullet B$

Phép toán đại số Boole với hai biến

A	B	A AND B $A \bullet B$
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A OR B $A + B$
0	0	0
0	1	1
1	0	1
1	1	1

A	NOT A \overline{A}
0	1
1	0

NOT là phép toán 1 biến

A	B	A NAND B $\overline{A \bullet B}$
0	0	1
0	1	1
1	0	1
1	1	0

A	B	A NOR B $\overline{A + B}$
0	0	1
0	1	0
1	0	0
1	1	0

A	B	A XOR B $A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Các đồng nhất thức của đại số Boole

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

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

$$1 \cdot A = A$$

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

$$0 \cdot A = 0$$

$$A \cdot A = A$$

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

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

$$A + B = B + A$$

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

$$0 + A = A$$

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

$$1 + A = 1$$

$$A + A = A$$


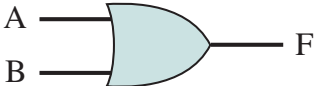
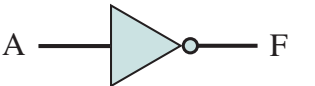

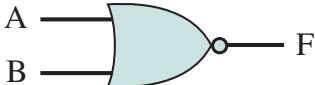
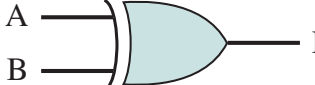
$$A + (B + C) = (A + B) + C$$

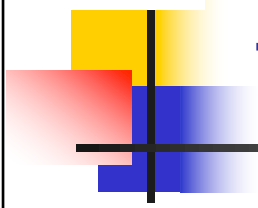
$$\overline{A + B} = \bar{A} \cdot \bar{B} \text{ (Định lý De Morgan)}$$

2.3. Các cổng logic (Logic Gates)

- Thực hiện các hàm logic:
 - NOT, AND, OR, NAND, NOR, XOR
- Cổng logic một đầu vào:
 - Cổng NOT
- Cổng hai đầu vào:
 - AND, OR, XOR, NAND, NOR
- Cổng nhiều đầu vào

Ký hiệu các cổng logic

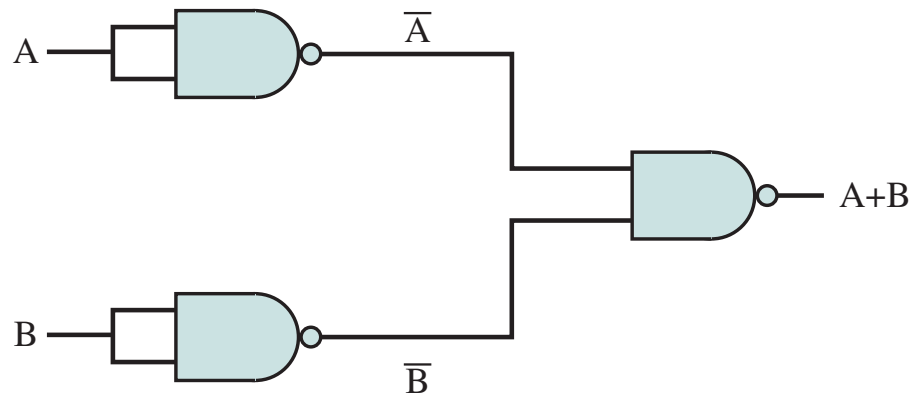
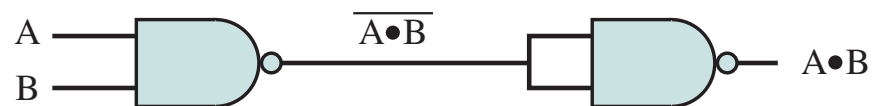
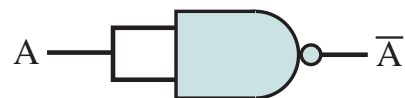
Name	Graphical Symbol	Algebraic Function	Truth Table															
AND		$F = A \bullet B$ or $F = AB$	<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></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																



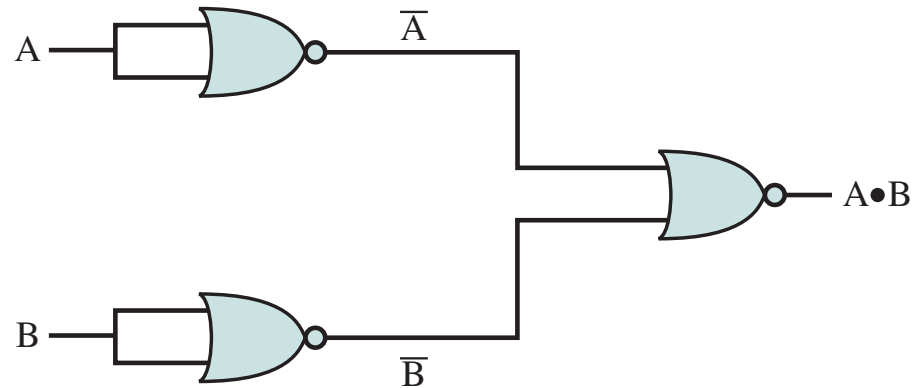
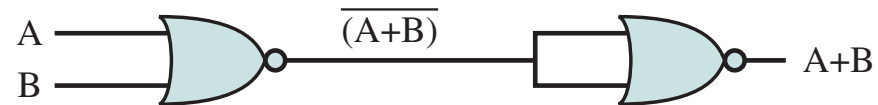
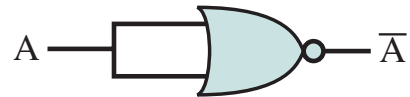
Tập đầy đủ

- 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}

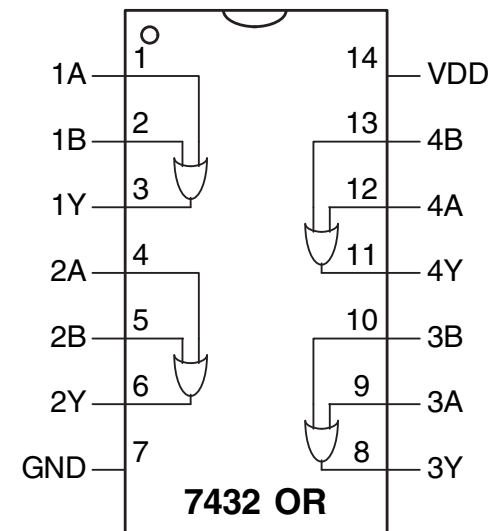
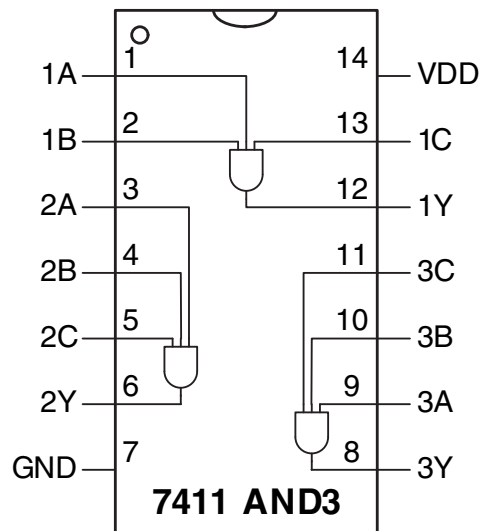
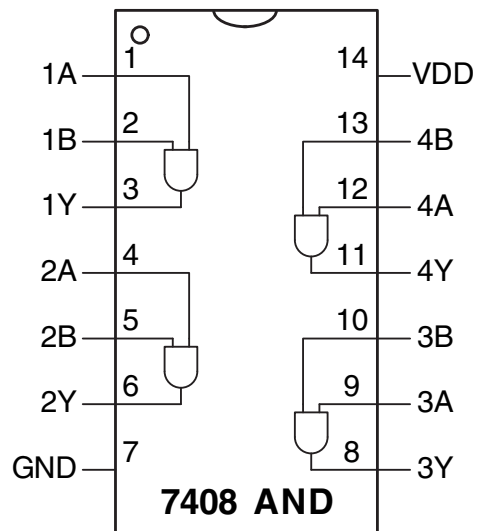
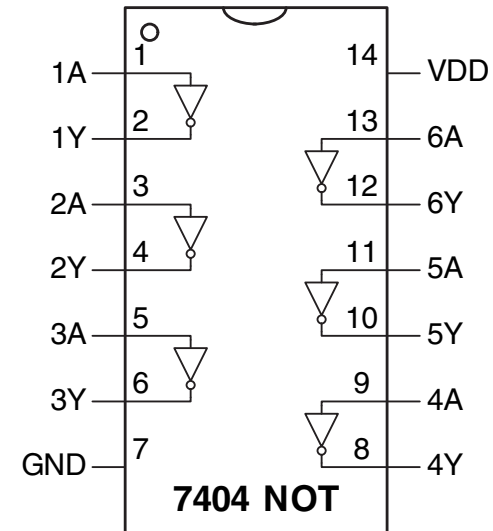
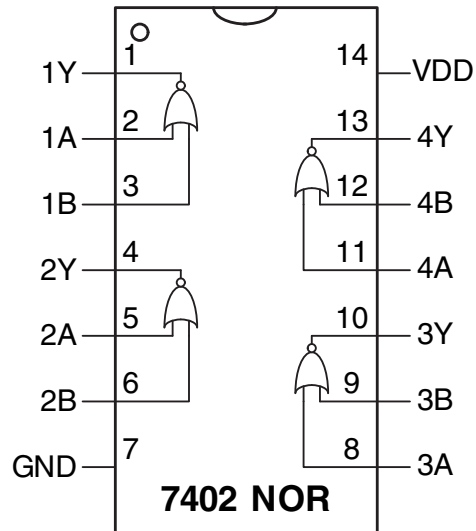
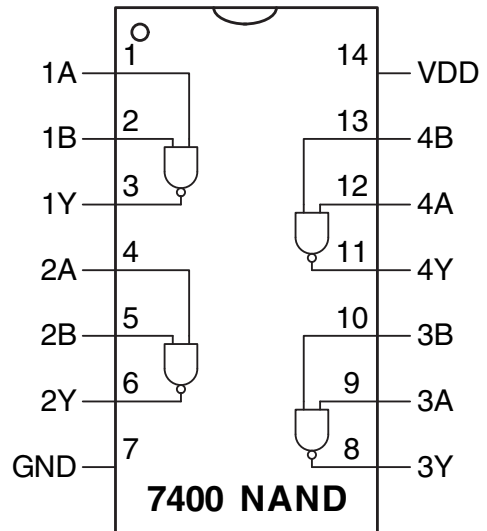
Sử dụng cổng NAND



Sử dụng cổng NOR



Một số vi mạch logic



2.4. 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 tổ hợp (Combinational Circuits)
 - Mạch không nhớ
 - Đầu ra được xác định bởi các giá trị hiện tại của đầu vào
 - Mạch dãy (Sequential Circuits)
 - Mạch có nhớ
 - Đầu ra được xác định bởi các giá trị trước đó và giá trị hiện tại của đầu vào

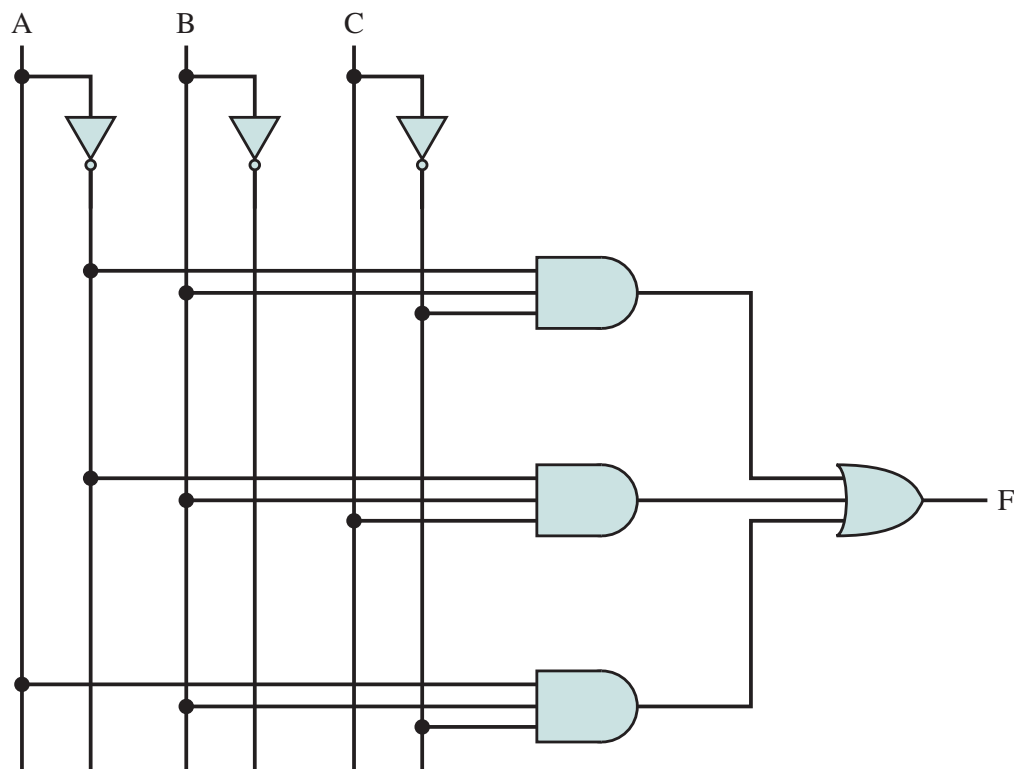


Mạch tổ hợp

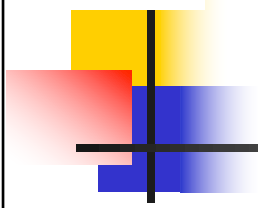
- Mạch tổ hợp là mạch logic trong đó đầu ra chỉ phụ thuộc đầu vào ở thời điểm hiện tại
- Là mạch không nhớ và được thực hiện bằng các cổng logic
- Mạch tổ hợp có thể được định nghĩa theo ba cách:
 - Bảng thật (True Table)
 - Dạng sơ đồ
 - Phương trình Boole

Ví dụ

Đầu vào			Đầu ra
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



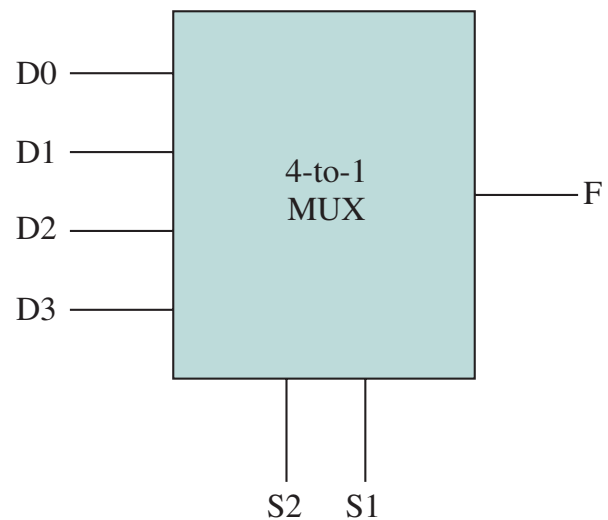
$$F = \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}\bar{C}$$



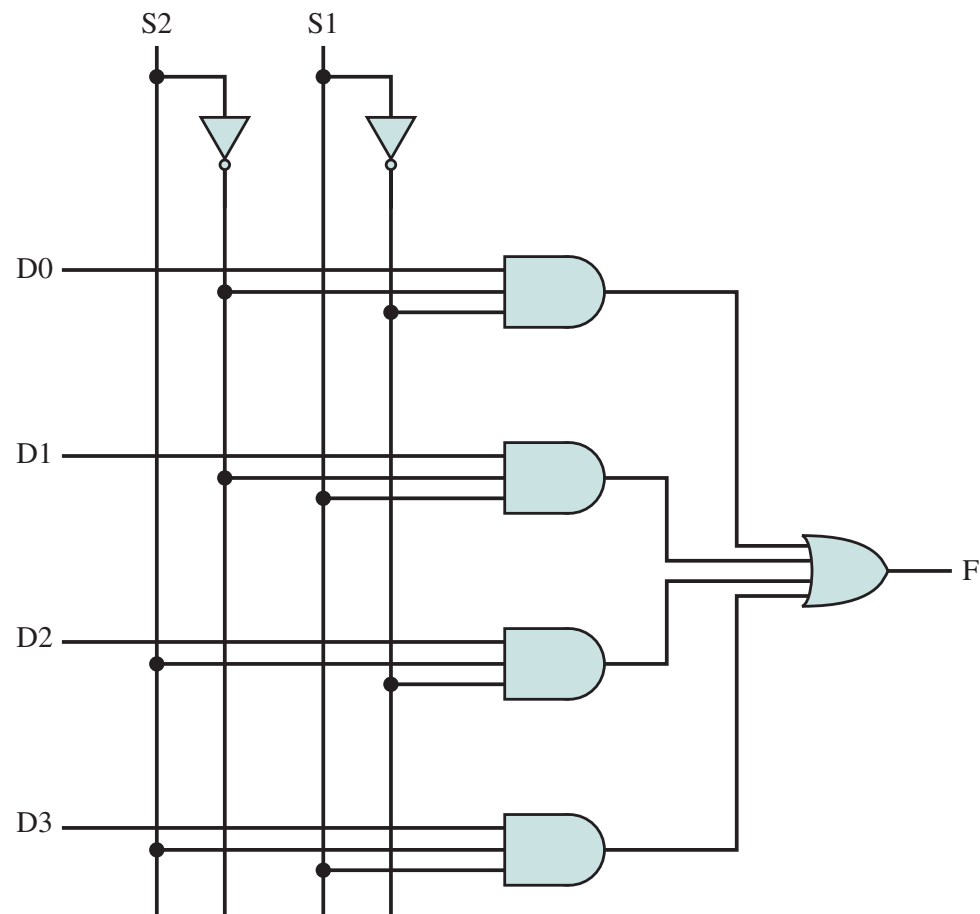
Bộ chọn kênh (Multiplexer - MUX)

- 2^n đầu vào dữ liệu
- n đầu vào chọn
- 1 đầu ra dữ liệu
- Mỗi tổ hợp đầu vào chọn (S) xác định đầu vào dữ liệu nào (D) sẽ được nối với đầu ra (F)

Bộ chọn kênh 4 đầu vào



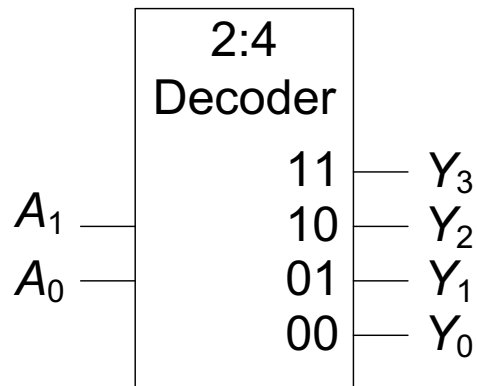
Đầu vào chọn		Đầu ra
S2	S1	F
0	0	D0
0	1	D1
1	0	D2
1	1	D3



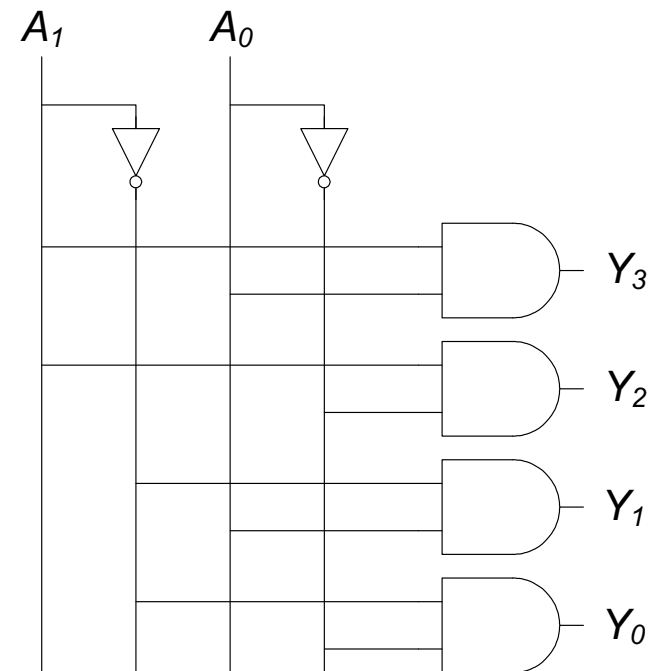
$$F = D0 \cdot \overline{S2} \cdot \overline{S1} + D1 \cdot \overline{S2} \cdot S1 + D2 \cdot S2 \cdot \overline{S1} + D3 \cdot S2 \cdot S1$$

Bộ giải mã (Decoder)

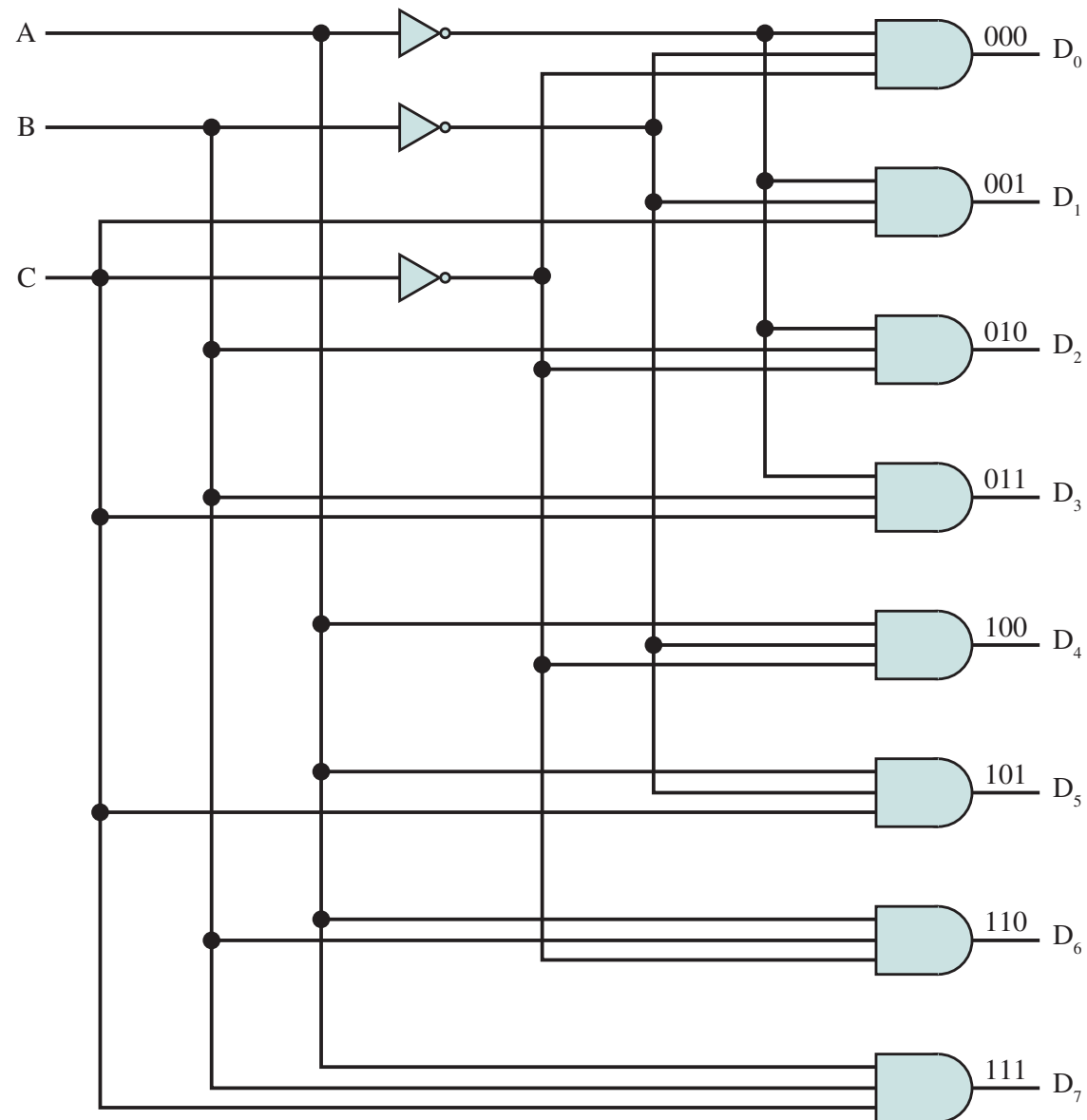
- N đầu vào, 2^N đầu ra
- Với một tổ hợp của N đầu vào, chỉ có một đầu ra tích cực (khác với các đầu ra còn lại)
- Ví dụ: Bộ giải mã 2 ra 4

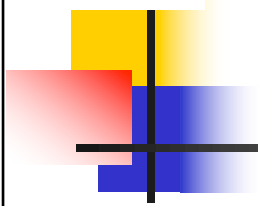


A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0



Thực hiện bộ giải mã 3 ra 8





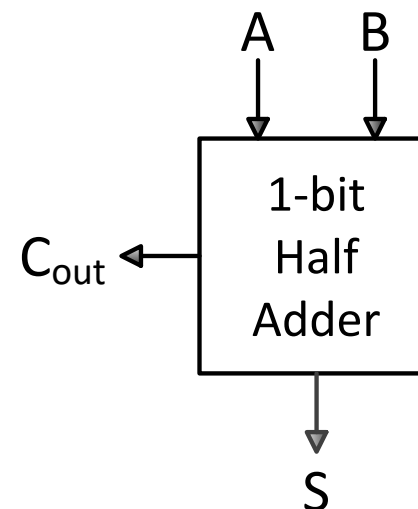
Bộ cộng

- Bộ cộng bán phần 1-bit (Half-adder)
 - Cộng hai bit tạo ra bit tổng và bit nhớ ra
- Bộ cộng toàn phần 1-bit (Full-adder)
 - Cộng 3 bit
 - Cho phép xây dựng bộ cộng N-bit

Bộ cộng bán phần 1-bit

0	0	1	1
+ 0	+ 1	+ 0	+ 1
0	1	1	10

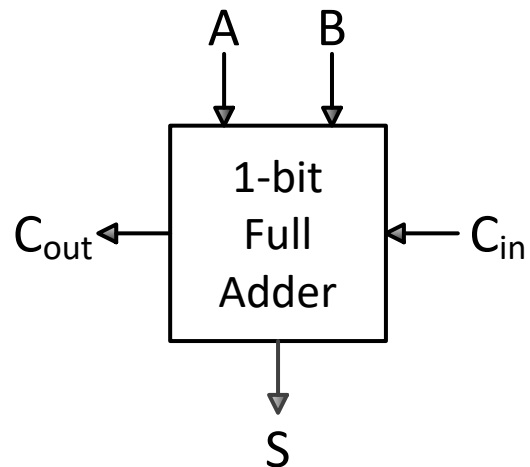
Đầu vào		Đầu ra	
A	B	S	C _{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



$$S = A \oplus B$$

$$C_{out} = AB$$

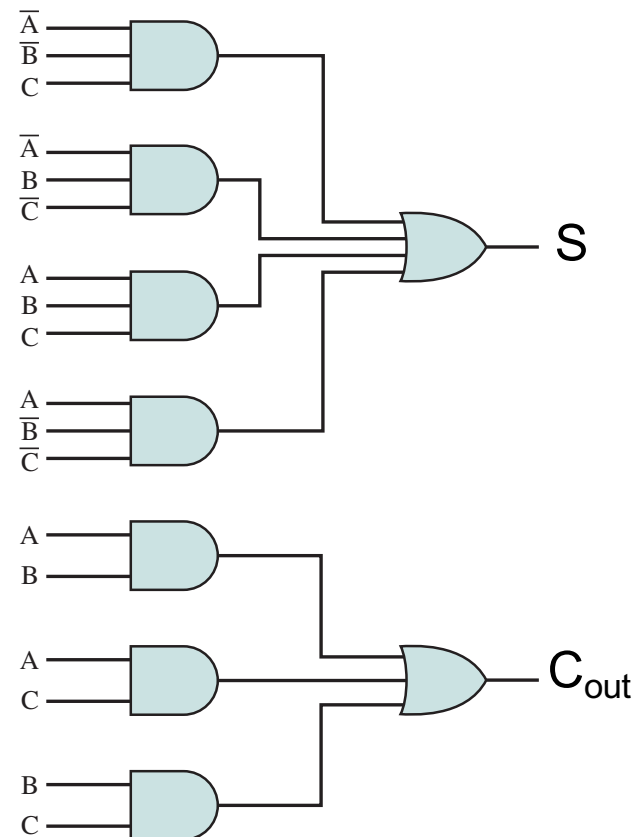
Bộ cộng toàn phần 1-bit



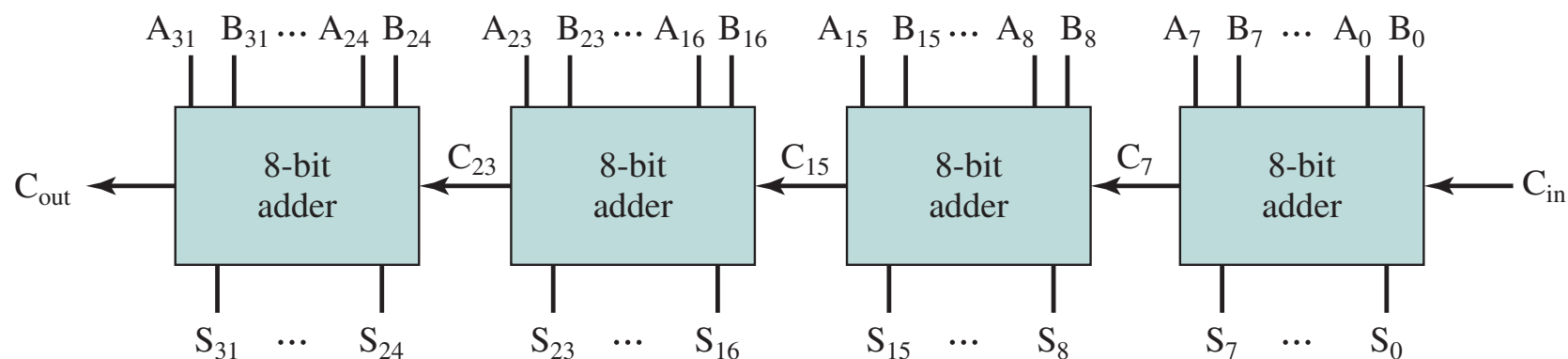
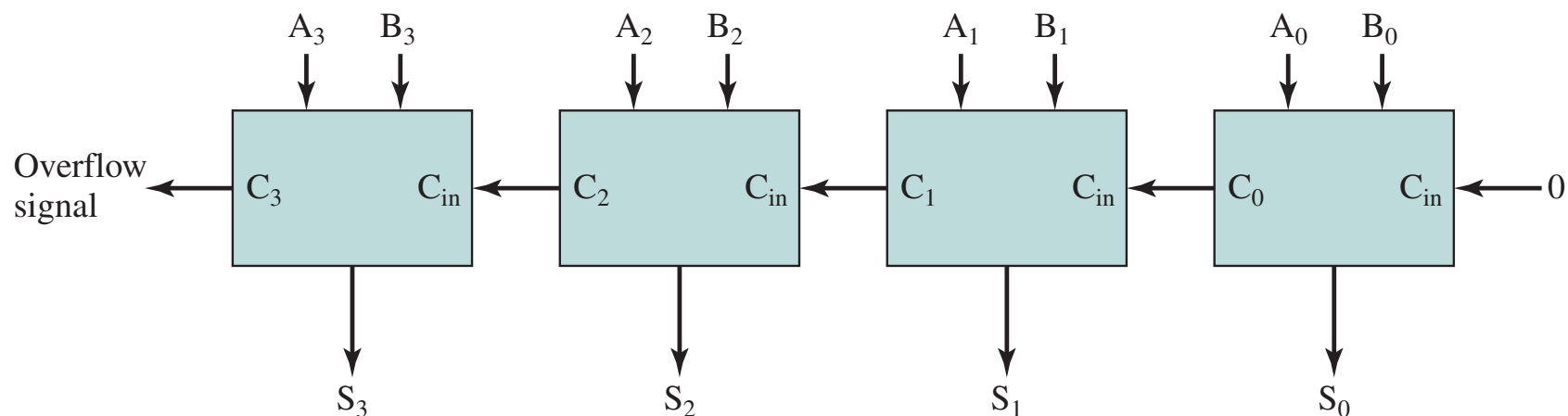
Đầu vào			Đầu ra	
C _{in}	A	B	S	C _{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$S = \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + A\overline{B}C$$

$$C_{out} = AB + AC + BC$$



Bộ cộng 4-bit và bộ cộng 32-bit

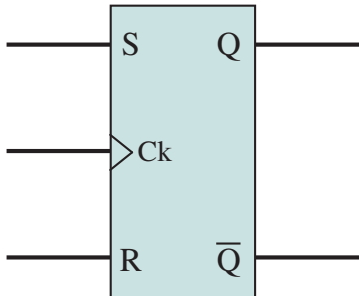
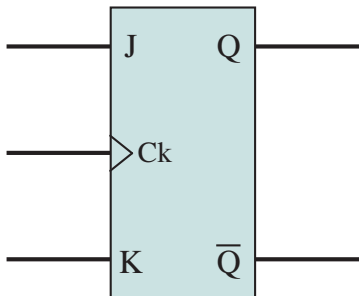
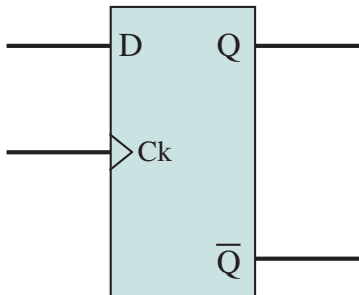




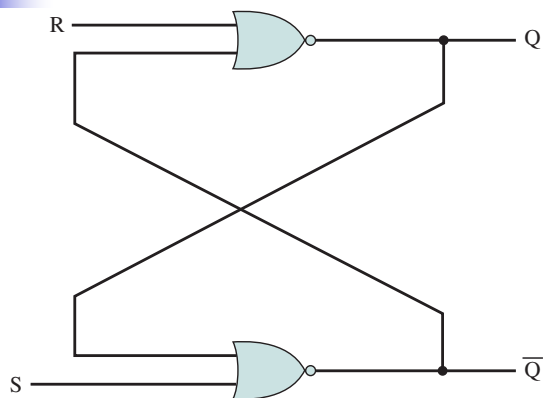
2.5. Mạch dẫy

- Mạch dẫy là mạch logic trong đó đầu ra phụ thuộc giá trị đầu vào ở thời điểm hiện tại và đầu vào ở thời điểm 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

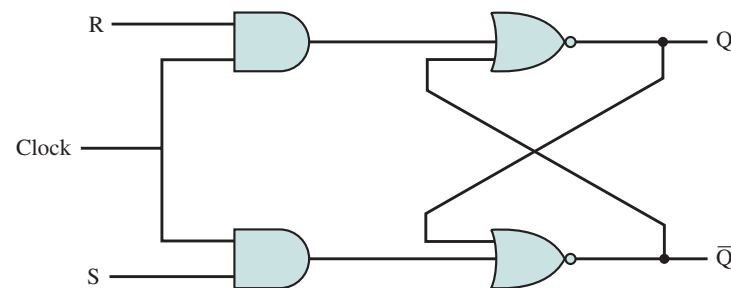
Các Flip-Flop cơ bản

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

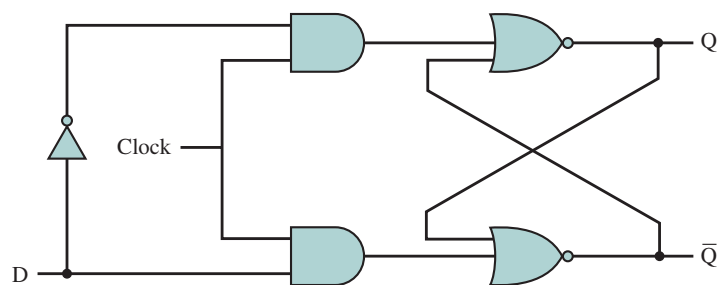
S-R Latch và các Flip-Flop



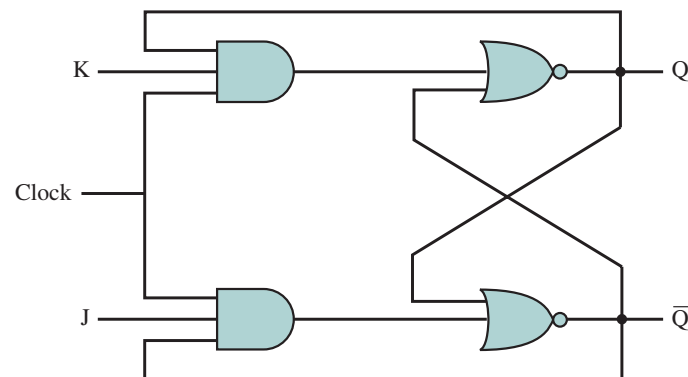
S-R Latch



S-R Flip-Flop

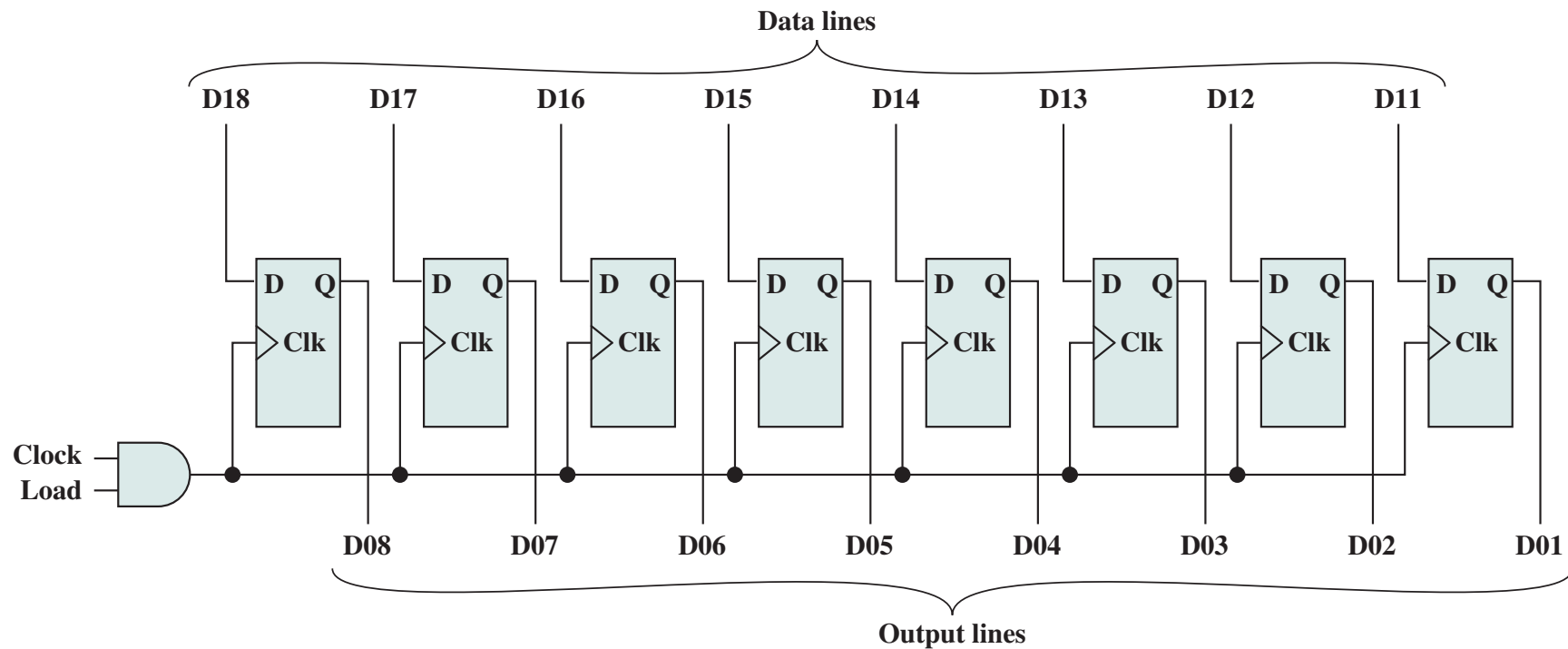


D Flip Flop

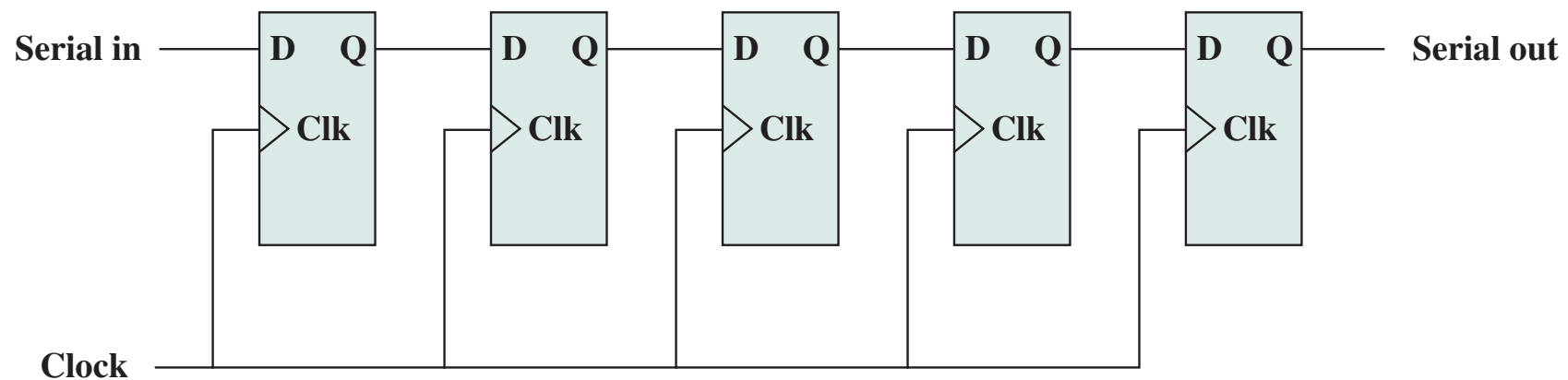


J-K Flip-Flop

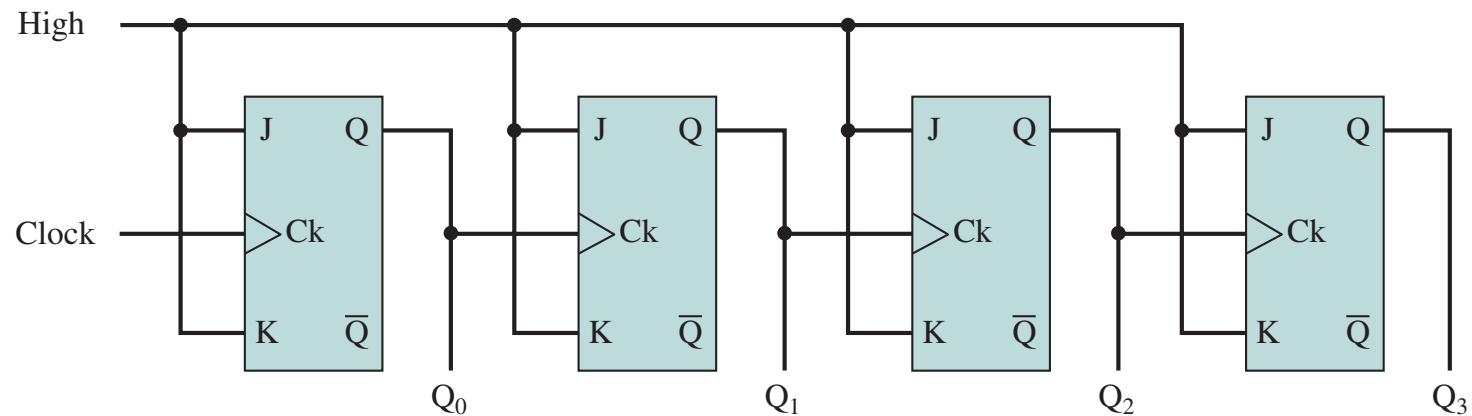
Thanh ghi 8-bit song song



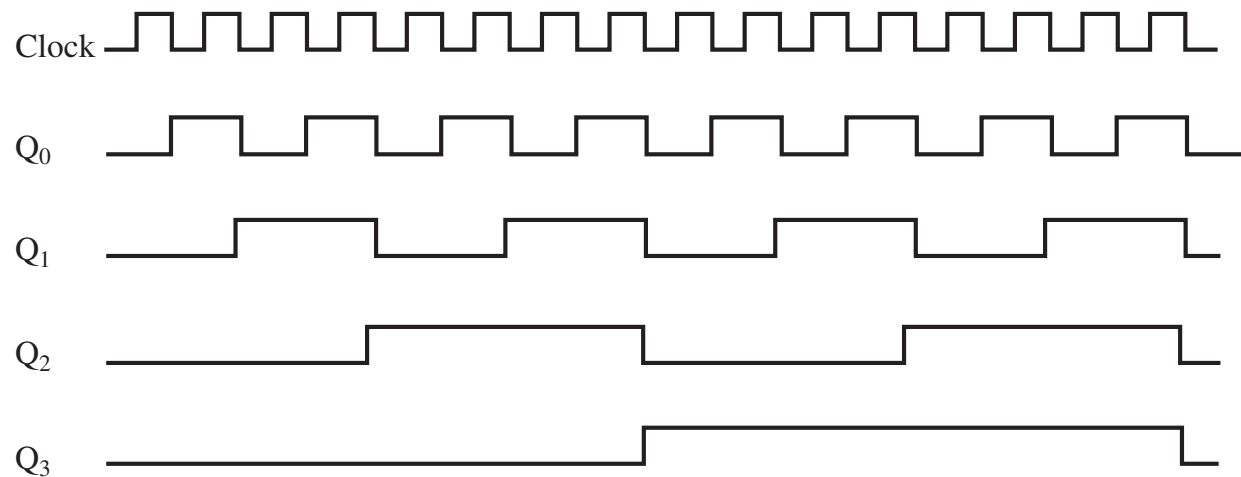
Thanh ghi dịch 5-bit



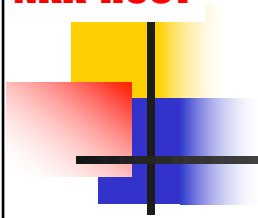
Bộ đếm 4-bit



(a) Sequential circuit



(b) Timing diagram



Hết chương 2