

# CHƯƠNG 3. BÀI\_2: BÌA KARNAUGH

2

- Các khái niệm cơ bản
- Các cách thiết kế một mạch logic
- Bìa Karnaugh
- Cổng XOR & XNOR
- Giới thiệu một số mạch logic

3

# Các khái niệm cơ bản

# Ví dụ

4

- Dùng định lý Boole để đơn giản hàm sau:

$$F(X, Y, Z) = (X + Y) (X + \bar{Y}) (\overline{XZ})$$

$(X + Y) (X + \bar{Y}) (\overline{XZ})$	Idempotent Law (Rewriting)
$(X + Y) (X + \bar{Y}) (\bar{X} + Z)$	DeMorgan's Law
$(XX + X\bar{Y} + XY + Y\bar{Y}) (\bar{X} + Z)$	Distributive Law
$((X + Y\bar{Y}) + X(Y + \bar{Y})) (\bar{X} + Z)$	Commutative & Distributive Laws
$((X + 0) + X(1)) (\bar{X} + Z)$	Inverse Law
$X(\bar{X} + Z)$	Idempotent Law
$X\bar{X} + XZ$	Distributive Law
$0 + XZ$	Inverse Law
$XZ$	Idempotent Law

# Phân tích

5

- Thông qua ví dụ rút gọn biểu thức ở trên, có nhiều cách để biểu diễn cùng một hàm Boole
  - ▣ Những cách này *tương đương về mặt logic (logically equivalent)*
  - ▣ Những biểu thức tương đương về mặt logic có *cùng bảng sự thật*
- Để giảm bớt những rắc rối có thể có, những nhà thiết kế biểu diễn hàm Boole dưới *dạng chính tắc (canonical)* hoặc *dạng chuẩn (standardized)*

# Minterms

6

- Minterms là các nhóm **AND** mà **tất cả các biến xuất hiện** ở dạng bình thường (nếu là 1) hoặc dạng bù (complement) (nếu là 0)
- Mỗi biến nhị phân có thể xuất hiện ở dạng bình thường (**X**) hoặc dạng bù (**X'**), có  $2^n$  tích chuẩn trong trường hợp có n biến
- Vd: 2 biến (X và Y) tạo  $2 \times 2 = 4$  tổ hợp:
  - XY** (cả 2 ở dạng bình thường)
  - XY'** (X bình thường, Y bù)
  - X'Y** (X bù, Y bình thường)
  - X'Y'** (cả 2 ở dạng bù)

# Maxterms

7

- Maxterms là các nhóm **OR** mà **tất cả các biến xuất hiện** ở dạng bình thường (nếu là 0) hoặc dạng bù (complement) (nếu là 1)
- Mỗi biến nhị phân có thể xuất hiện ở dạng bình thường (**X**) hoặc dạng bù (**X'**), có  $2^n$  tổng chuẩn trong trường hợp có n biến
- Vd: 2 biến (X và Y) tạo  $2 \times 2 = 4$  tổ hợp:
  - X + Y** (cả 2 ở dạng bình thường)
  - X + Y'** (X bình thường, Y bù)
  - X' + Y** (X bù, Y bình thường)
  - X' + Y'** (cả 2 ở dạng bù)

# Minterms & Maxterms

8

x	y	Index	Minterm	Maxterm
0	0	0	$m_0 = x' y'$	$M_0 = x + y$
0	1	1	$m_1 = x' y$	$M_1 = x + y'$
1	0	2	$m_2 = x y'$	$M_2 = x' + y$
1	1	3	$m_3 = x y$	$M_3 = x' + y'$

- minterm  $m_i$  sẽ là **1** đối với mỗi sự kết hợp của x và y
- maxterm là **bù của minterm**
- **$M_i = m_i'$**  hoặc  **$m_i = M_i'$**

#	x	y	z	Minterm	Maxterm
0	0	0	0	$m_0 = x' y' z'$	$M_0 = x + y + z$
1	0	0	1	$m_1 = x' y' z$	$M_1 = x + y + z'$
2	0	1	0	$m_2 = x' y z'$	$M_2 = x + y' + z$
3	0	1	1	$m_3 =$	$M_3 =$
4	1	0	0		



# Minterms & Maxterms (tt)

9

x	y	z	Index	Minterm	Maxterm
0	0	0	0	$m_0 = x' y' z'$	$M_0 = x + y + z$
0	0	1	1	$m_1 = x' y' z$	$M_1 = x + y + z'$
0	1	0	2	$m_2 = x' y z'$	$M_2 = x + y' + z$
0	1	1	3	$m_3 = x' y z$	$M_3 = x + y' + z'$
1	0	0	4	$m_4 = x y' z'$	$M_4 = x' + y + z$
1	0	1	5	$m_5 = x y' z$	$M_5 = x' + y + z'$
1	1	0	6	$m_6 = x y z'$	$M_6 = x' + y' + z$
1	1	1	7	$m_7 = x y z$	$M_7 = x' + y' + z'$

# Dạng chính tắc (Canonical Forms)

10

- Tổng các tích chuẩn (**Sum of minterms**)
  - ▣ Dựa vào bảng sự thật, viết biểu thức tổng các tích chuẩn bằng cách chọn các hàng mà tại đó hàm output có giá trị là **1**
  
- Tích các tổng chuẩn (**Product of maxterms**)
  - ▣ Dựa vào bảng sự thật, viết biểu thức tích các tổng chuẩn bằng cách chọn các hàng mà tại đó hàm output có giá trị là **0**

**Câu 24:**

Cho hàm logic dạng tuyển sau:

$$Z = F(A, B, C) = \sum (1, 2, 3, 5, 7)$$

Hãy tối giản hóa bằng phương pháp đại số.

**Câu 25:**

Cho hàm logic dạng hội sau:

$$Z = F(A, B, C) = \prod (0, 4, 6)$$

Hãy tối giản hóa bằng phương pháp đại số.

# Dạng chính tắc (Canonical Forms) (tt)

11

#	A	B	C	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

- Tổng các tích chuẩn (Sum of minterms)

$$F = \sum (1, 2, 5, 6)$$

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

- Tích các tổng chuẩn (Product of maxterms)

$$F = \prod (0, 3, 4, 7)$$

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

# Dạng chính tắc (Canonical Forms) (tt)

12

Sum of Minterms ( $\Sigma$ )	Product of Maxterms ( $\Pi$ )
Chỉ quan tâm tổ hợp input mà $F=1$	Chỉ quan tâm tổ hợp input mà $F=0$
$X = 0 \Rightarrow X'$	$X = 0 \Rightarrow X$
$X = 1 \Rightarrow X$	$X = 1 \Rightarrow X'$

# Dạng chuẩn (Standard Form)

13

- Dạng chính tắc có thể được đơn giản hoá để thành dạng chuẩn tương đương
  - ▣ Ít nhóm AND (hoặc OR) và/hoặc các nhóm này có ít biến hơn
- Tổng các tích - SoP (Sum-of-Product)
 
$$F(x, y, z) = xy + xz + yz$$
- Tích các tổng - PoS (Product-of-Sum)
 
$$F(x, y, z) = (x+y)(x+z)(y+z)$$

Có thể chuyển SoP về dạng chính tắc bằng cách AND thêm  $(x+x')$  và PoS về dạng chính tắc bằng cách OR thêm  $xx'$

14

# Thiết kế mạch logic

# Đặc tả đầu vào (Specification)

15

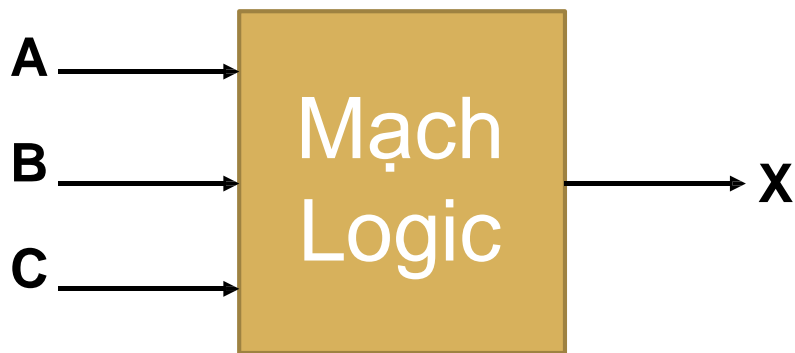
- Thiết kế một mạch logic với
  - ▣ 3 inputs
  - ▣ 1 output
  - ▣ Output ở mức **HIGH** khi đa số input ở mức **HIGH**



# Quy trình thiết kế mạch logic

16

## □ Bước 1: xây dựng bảng sự thật



A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

# Quy trình thiết kế mạch logic (tt)

17

## □ Bước 2: chuyển bảng sự thật sang biểu thức logic

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

→  $A'BC$

→  $AB'C$

→  $ABC'$

→  $ABC$

Biểu thức SOP cho ngõ ra X:

$$X = A'BC + AB'C + ABC' + ABC$$

# Quy trình thiết kế mạch logic (tt)

18

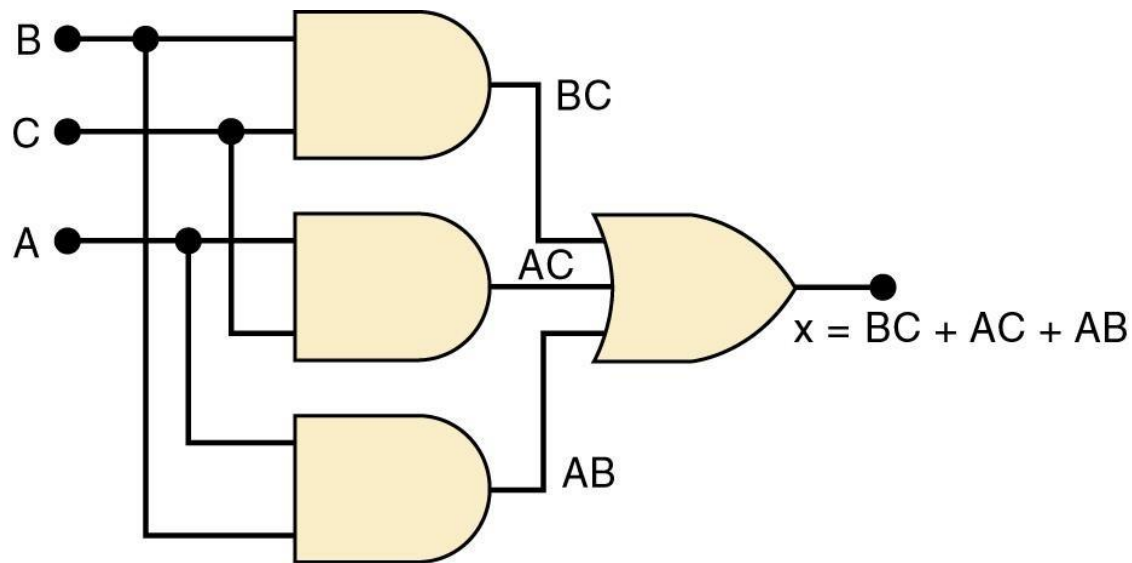
- Bước 3: đơn giản biểu thức logic bằng các biến đổi đại số
 
$$\begin{aligned}
 X &= A'BC + AB'C + ABC' + ABC \\
 &= A'BC + ABC + AB'C + ABC + ABC' + ABC \\
 &= BC + AC + AB
 \end{aligned}$$

# Quy trình thiết kế mạch logic (tt)

19

- Bước 4: vẽ sơ đồ mạch logic cho  

$$x = BC + AC + AB$$



# Quy trình thiết kế mạch logic (tt)

20

- Nhận xét hai vấn đề của biến đổi đại số:
  - ▣ Không có tính hệ thống
  - ▣ Rất khó để kiểm tra rằng giải pháp tìm ra đã là tối ưu hay không

=> **BÌA KARNAUGH**

21

# Bìa Karnaugh

# Chi phí để hiện thực một mạch logic

22

- Chi phí để tạo ra một mạch logic liên quan đến
  - ▣ Số cổng (gates) sử dụng và
  - ▣ Số đầu vào của mỗi cổng
- Chi phí của một biểu thức boolean B biểu diễn dưới dạng tổng của các tích (Sum-of-Product) được tính như sau:

$$C(B) = O(B) + \sum_{j=0}^{k-1} P_j(B)$$

- ***k***: số term trong B

$$O(B) = \begin{cases} m & \text{khi B có } m \text{ term} \\ 0 & \text{khi B có } 1 \text{ term} \end{cases}$$

$$P_j(B) = \begin{cases} m & \text{khi term thứ } j \text{ có } m \text{ literal} \\ 0 & \text{khi term thứ } j \text{ có } 1 \text{ literal} \end{cases}$$

- ***Literal***: biến Boolean ở dạng bình thường hoặc dạng bù

# Chi phí để hiện thực một mạch logic (tt)

23

- Tính chi phí của các biểu thức sau:

$$C(B) = O(B) + \sum_{j=0}^{k-1} P_j(B)$$

$$O(B) = \begin{cases} m & \text{khi } B \text{ có } m \text{ term} \\ 0 & \text{khi } B \text{ có } 1 \text{ term} \end{cases}$$

$$P_j(B) = \begin{cases} m & \text{khi term thứ } j \text{ có } m \text{ literal} \\ 0 & \text{khi term thứ } j \text{ có } 1 \text{ literal} \end{cases}$$

$$f1(w,x,y,z) = wxy'z + wxyz'$$

$$f2(w,x,y,z) = w' + x' + yz + y'z'$$

$$g1(XYZ) = XY + X'Z + YZ$$

$$g2(XYZ) = XY + X'Z$$

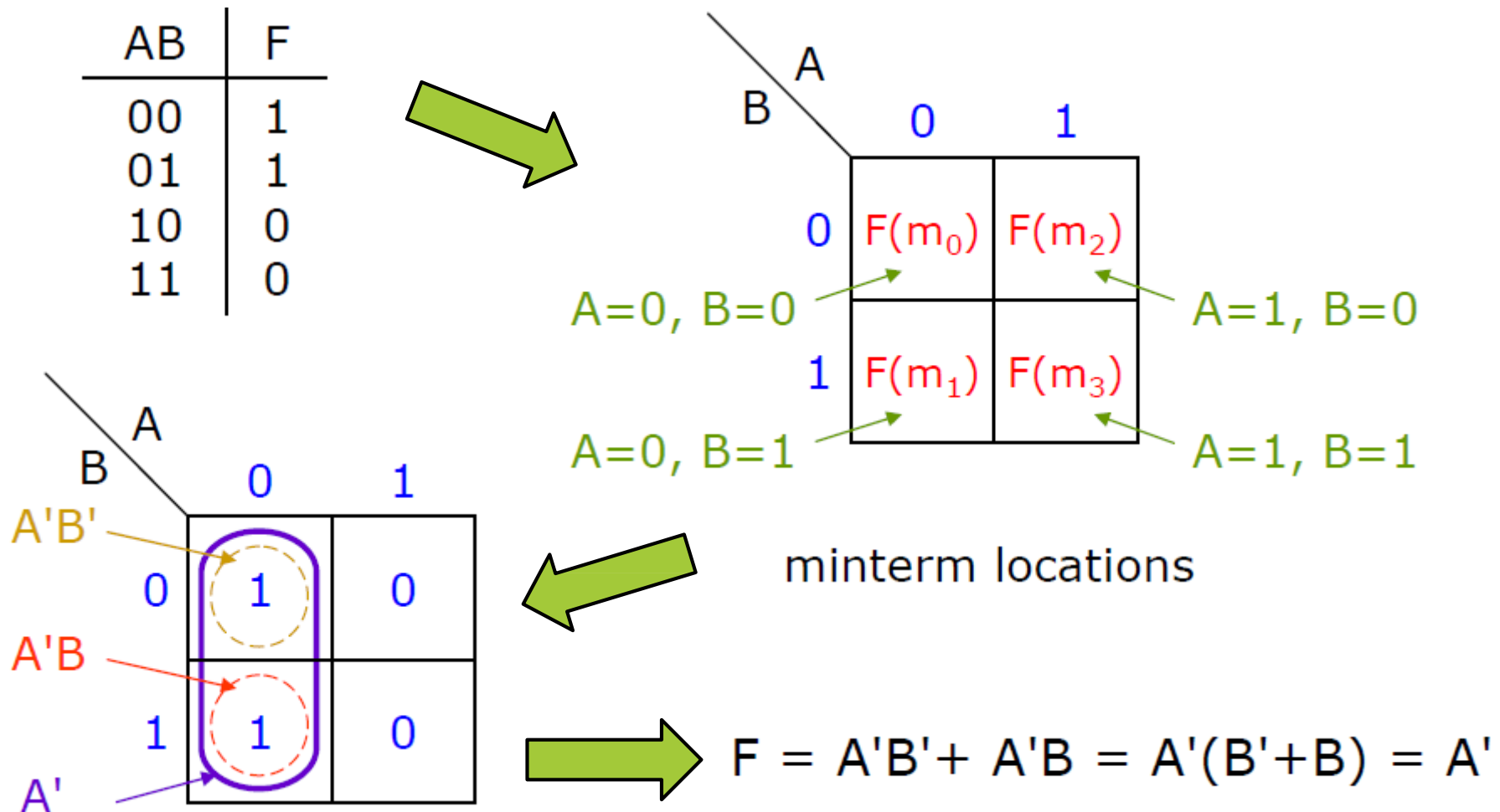
$$h1(a,b) = ab$$

$$h2(a,b) = b'$$



# Bìa Karnaugh 2 biến

24



# Bìa Karnaugh 3 biến

25

BC \ A	A	
	0	1
00	0	4
01	1	5
11	3	7
10	2	6

minterm locations

ABC	F
000	0
001	0
010	1
011	1
100	1
101	0
110	1
111	0

BC \ A	A	
	0	1
00	0	1
01	0	0
11	1	0
10	1	1

$$\begin{aligned}
 F &= A'BC' + A'BC + AB'C' + ABC' \\
 &= A'B + AC' + BC' \\
 &= A'B + AC'
 \end{aligned}$$

# Bìa Karnaugh 3 biến

26

$$F = m_1 + m_3 + m_5$$

$$= M_0 M_2 M_4 M_6 M_7$$

		a	
		0	1
bc	00	0	0
	01	1	1
	11	1	0
	10	0	0



$$F = a'c + b'c$$

		a	
		0	1
bc	00	0	0
	01	1	1
	11	1	0
	10	0	0

# Bìa Karnaugh 3 biến

27

$$G = (m_1 + m_3 + m_5)'$$

$$= (M_0 M_2 M_4 M_6 M_7)'$$

		a	
bc		0	1
00		1	1
01		0	0
11		0	1
10		1	1

Simplify

$$G = ab + c'$$

		a	
bc		0	1
00		1	1
01		0	0
11		0	1
10		1	1

# Bìa Karnaugh 3 biến

28

		x	
		0	1
yz	00		
	01	1	
	11	1	1
	10		1

hay

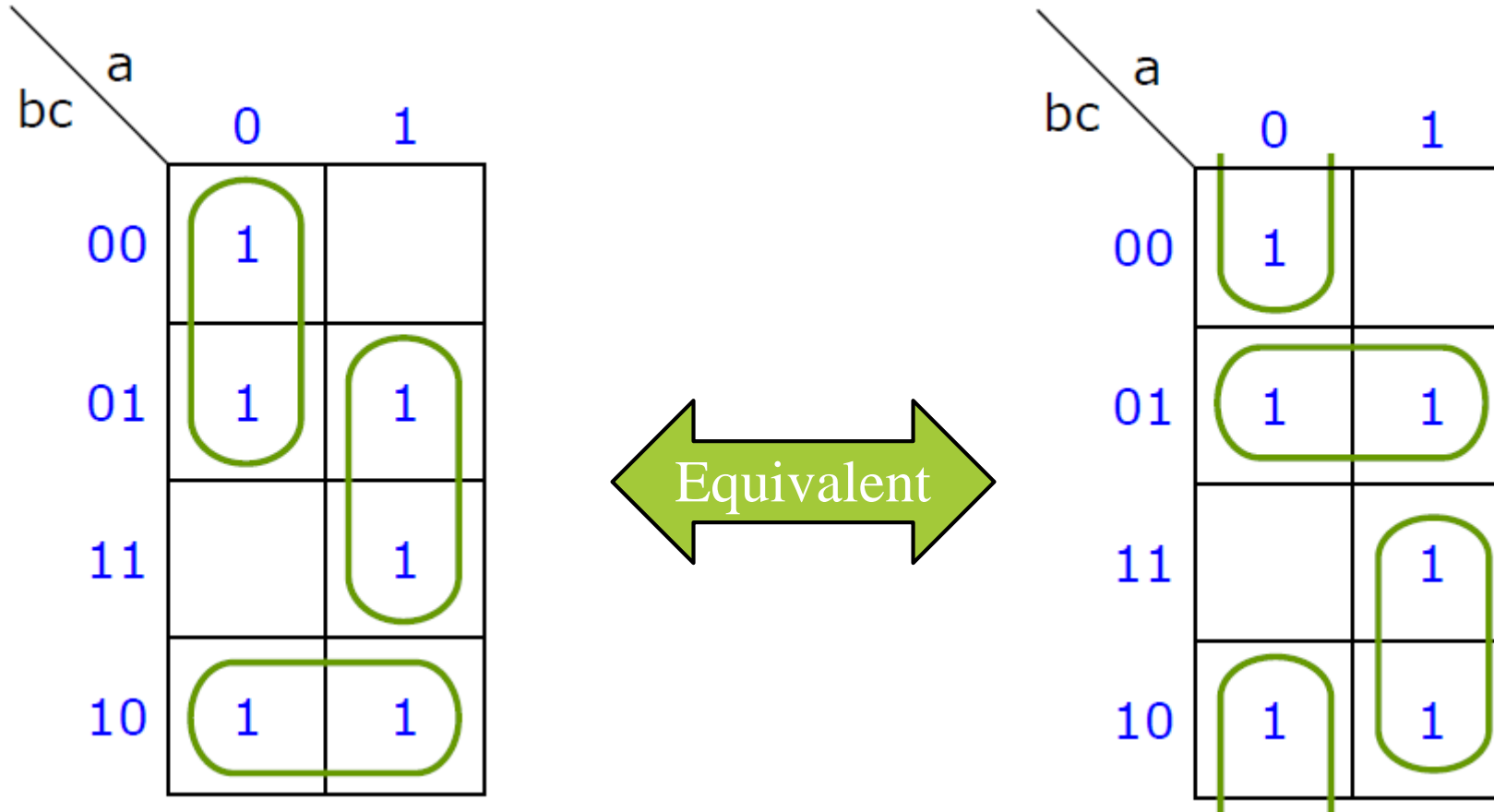
		x	
		0	1
yz	00		
	01	1	
	11	1	1
	10		1

Nhóm ô màu đỏ là dư thừa  
 $\Rightarrow$  kết quả không tối ưu

Chính xác

# Bìa Karnaugh 3 biến

29



$$F = a'b' + bc' + ac = a'c' + b'c + ab$$

# Bìa Karnaugh 4 biến

30

CD \ AB	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

minterm locations

Simplify

$F = ac + a'b + d'$

cd \ ab	00	01	11	10
00	1	1	1	1
01		1		
11		1	1	1
10	1	1	1	1

$$f_2 = \sum m(0, 2, 3, 5, 6, 7, 8, 10, 11, 14, 15)$$

		ab			
		00	01	11	10
cd	00	1			1
	01		1		
	11	1	1	1	1
	10	1	1	1	1

Simpl



# Bìa Karnaugh 4 biến

31

$f_1 = \sum m(1,3,4,5,10,12,13)$

cd \ ab	00	01	11	10
00		1	1	
01	1	1	1	
11	1			
10				1

Simplify

$f_1 = ab'cd' + a'b'd + bc'$

cd \ ab	00	01	11	10
00		1	1	
01	1	1	1	
11	1			
10				1

# Bìa Karnaugh 4 biến

32

$$f_2 = \sum m(0, 2, 3, 5, 6, 7, 8, 10, 11, 14, 15)$$

cd \ ab	00	01	11	10
00	1			1
01		1		
11	1	1	1	1
10	1	1	1	1

Simplify

$$f_2 = c + b'd' + a'bd$$

cd \ ab	00	01	11	10
00	1			1
01		1		
11	1	1	1	1
10	1	1	1	1

# Quy tắc rút gọn bìa Karnaugh

33

- **Khoanh vòng (looping)** là quá trình kết hợp các ô kề nhau lại với nhau. Thông thường ta khoanh các ô chứa giá trị 1.
- Ngõ xuất có thể được đơn giản hóa bằng cách khoanh vòng.

# Quy tắc tính giá trị của 1 vòng

34

- Khi một biến xuất hiện cả dang đảo và không đảo trong một vòng, biến đó sẽ được đơn giản khỏi biểu thức.
- Các biến chung cho mọi ô trong một vòng phải xuất hiện trong biểu thức cuối cùng.

# Khoanh vòng 2 ô kề nhau

35

**C**

		0	1
00		0	0
01	<b>AB</b>	1	0
11		1	0
10		0	0

$$X = \overline{A}B\overline{C} + AB\overline{C} = B\overline{C}$$

**C**

		0	1
00		0	0
01	<b>AB</b>	1	1
11		0	0
10		0	0

$$X = \overline{A}B\overline{C} + \overline{A}BC = \overline{A}B$$

**C**

		0	1
00		1	0
01	<b>AB</b>	0	0
11		0	0
10		1	0

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

# Khoanh vòng 2 ô kề nhau (tt)

36

**CD**

	00	01	11	10
00	1	0	0	0
01	0	1	1	0
11	0	0	0	0
10	1	0	0	0

**AB**

$$X = \overline{B}\overline{C}\overline{D} + \overline{A}BD$$

**CD**

	00	01	11	10
00	1	1	0	0
01	0	0	0	0
11	1	0	0	1
10	0	0	0	0

**AB**

$$X = \overline{A}\overline{B}\overline{C} + ABD$$

# Khoanh vòng 4 ô kề nhau

37

C

	0	1
00	1	0
01	1	0
11	1	0
10	1	0

AB

$$X = \bar{C}$$

CD

	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	0	0	0	0

AB

$$X = \bar{A}B$$

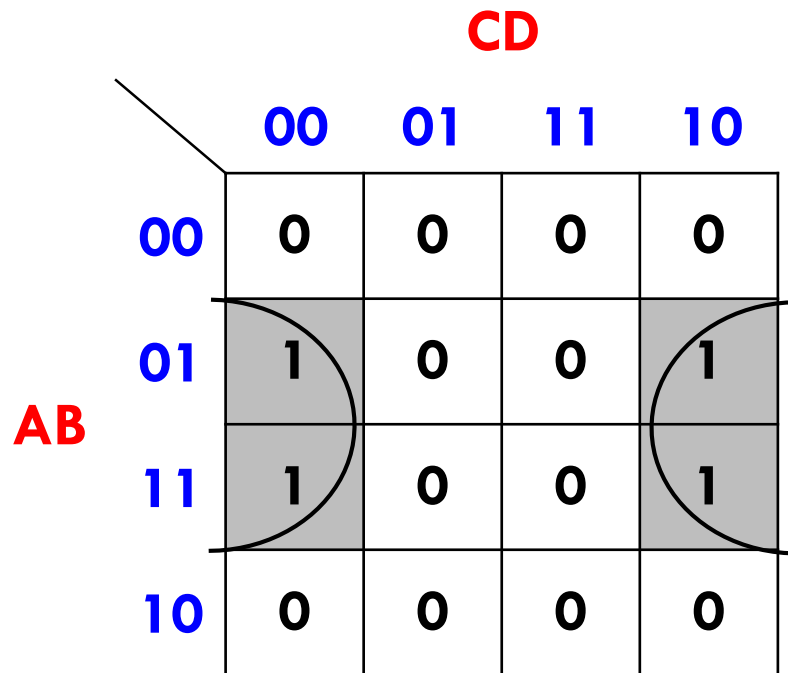
# Khoanh vòng 4 ô kề nhau (tt)

38

**CD**

	<b>00</b>	<b>01</b>	<b>11</b>	<b>10</b>
<b>00</b>	0	0	0	0
<b>01</b>	1	0	0	1
<b>11</b>	1	0	0	1
<b>10</b>	0	0	0	0

**AB**

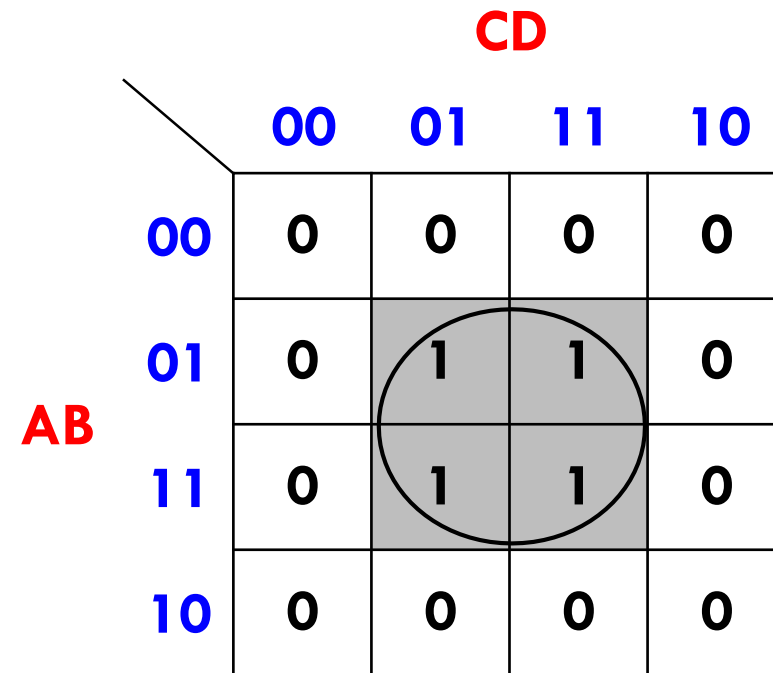


$$X = B\bar{D}$$

**CD**

	<b>00</b>	<b>01</b>	<b>11</b>	<b>10</b>
<b>00</b>	0	0	0	0
<b>01</b>	0	1	1	0
<b>11</b>	0	1	1	0
<b>10</b>	0	0	0	0

**AB**



$$X = BD$$



# Khoanh vòng 4 ô kề nhau (tt)

39

**CD**

	<b>00</b>	<b>01</b>	<b>11</b>	<b>10</b>
<b>00</b>	1	0	0	1
<b>01</b>	0	0	0	0
<b>11</b>	0	0	0	0
<b>10</b>	1	0	0	1

**AB**

$$X = \overline{B}\overline{D}$$

# Khoanh vòng 8 ô kề nhau

40

**CD**

	00	01	11	10
00	1	1	1	1
01	0	0	0	0
11	0	0	0	0
10	1	1	1	1

**AB**

$$X = \bar{B}$$

**CD**

	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	1	0	0	1
10	1	0	0	1

**AB**

$$X = \bar{D}$$

# Khoanh vòng 8 ô kề nhau (tt)

41

**CD**

	<b>00</b>	<b>01</b>	<b>11</b>	<b>10</b>
<b>00</b>	1	1	1	1
<b>01</b>	1	1	1	1
<b>11</b>	0	0	0	0
<b>10</b>	0	0	0	0

**AB**

$X = \bar{A}$

**CD**

	<b>00</b>	<b>01</b>	<b>11</b>	<b>10</b>
<b>00</b>	0	1	1	0
<b>01</b>	0	1	1	0
<b>11</b>	0	1	1	0
<b>10</b>	0	1	1	0

**AB**

$X = D$

# Quá trình đơn giản hóa

42

- Xây dựng bảng K-map và đặt 1 hoặc 0 trong các ô tương ứng với bảng sự thật.
- Khoanh vòng **các ô giá trị 1 đơn lẻ**, không tiếp giáp với các ô giá trị 1 khác (vòng đơn).
- Khoanh vòng **các cặp giá trị 1** không tiếp giáp với các ô giá trị 1 nào khác nữa (vòng kép).
- Khoanh vòng **các ô 8 giá trị 1** (nếu có) ngay cả nếu nó chứa 1 hoặc nhiều ô đã được khoanh vòng.
- Khoanh vòng **các ô 4 giá trị 1** (nếu có) chứa một hoặc nhiều ô chưa được khoanh vòng. Phải đảm bảo số vòng là ít nhất.
- Khoanh vòng các cặp giá trị 1 tương ứng với các ô giá trị 1 chưa được khoanh vòng. Phải đảm bảo **số vòng là ít nhất**.
- Tạo cổng OR các số hạng được tạo bởi mỗi vòng

# Ví dụ

43

CD

		00	01	11	10
00	0	0	0	0	1
01	0	1	1	1	0
11	0	1	1	1	0
10	0	0	1	1	0

AB

$$X = \overline{A}\overline{B}\overline{C}\overline{D} + ACD + BD$$

# Ví dụ (tt)

44

CD

		00	01	11	10
AB	00	0	0	1	0
	01	1	1	1	1
	11	1	1	0	0
	10	0	0	0	0

$$X = \bar{A}CD + \bar{A}B + B\bar{C}$$

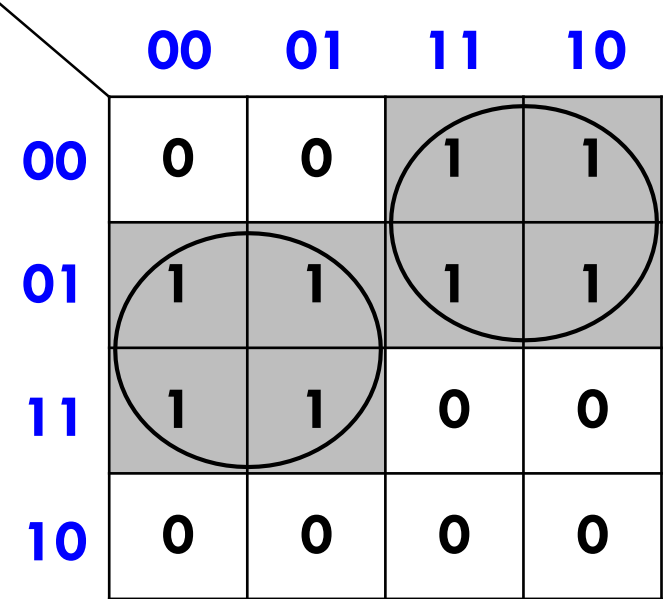
# Ví dụ (tt)

45

CD

		00	01	11	10
00		0	0	1	1
01		1	1	1	1
11		1	1	0	0
10		0	0	0	0

AB



$$X = B\bar{C} + \bar{A}C$$

# Ví dụ (tt)

46

CD

		00	01	11	10
AB	00	0	1	0	0
	01	0	1	1	1
	11	1	1	1	0
	10	0	1	1	0

$$X = A.B.\bar{C} + \bar{A}.\bar{C}.D + \bar{A}.B.C + A.C.D$$

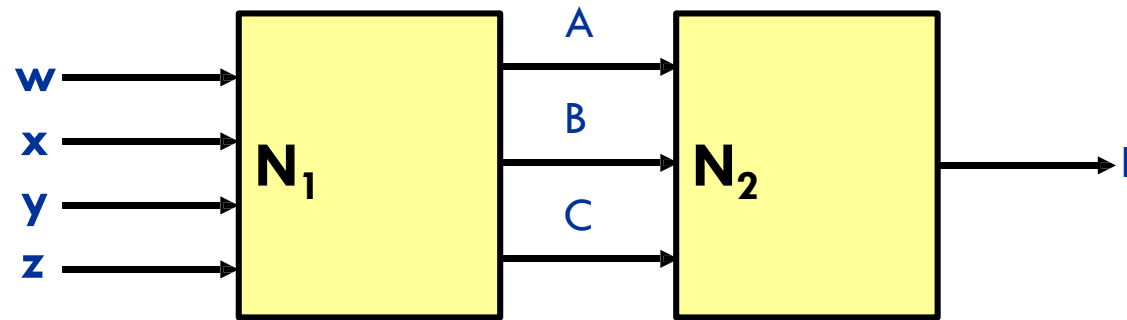


# PP bảng Karnaugh - Tóm tắt

- So sánh với phương pháp đại số, phương pháp dùng K-map có tính hệ thống hơn, ít bước hơn và luôn tạo ra được biểu thức tối giản nhất.
- Bảng Karnaugh có thể dùng **tối đa** là với hàm **6** biến. Đối với những mạch có số ngõ nhập lớn ( $>6$ ), người ta dùng thêm các kỹ thuật phức tạp để thiết kế.

# Trường hợp “Don’t Care”

48



**Giả thuyết:**  $N_1$  không bao giờ cho kết quả khi  
 $ABC = 001$  and  $ABC = 110$

**Câu hỏi:** F cho ra giá trị gì trong trường hợp  
 $ABC = 001$  and  $ABC = 110$ ?

**We don't care!!!**

# Trường hợp “Don’t Care” (tt)

49

- Trong trường hợp này thì chúng ta phải làm thế nào để đơn giản  $N_2$ ?

A	B	C	F
0	0	0	1
0	0	1	<del>X</del> 0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	<del>X</del> 0
1	1	1	1

Nếu ta giả sử  $F(0,0,1) = 0$  và  $F(1,1,0)=0$ , ta có biểu thức sau:

$$\begin{aligned}
 F(A,B,C) &= A'B'C' + A'BC' + A'BC + ABC \\
 &= A'C'(B' + B) + (A' + A)BC \\
 &= A'C' \cdot 1 + 1 \cdot BC \\
 &= A'C' + BC
 \end{aligned}$$

# Trường hợp “Don’t Care” (tt)

50

- Tuy nhiên, nếu giả sử sử  $F(0,0,1) = 1$  và  $F(1,1,0)=1$ , ta có biểu thức sau:

A	B	C	F
0	0	0	1
0	0	1	<del>X</del> 1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	<del>X</del> 1
1	1	1	1

$$F(A,B,C) = A'B'C' + A'B'C + A'BC' + A'BC + ABC' + ABC$$

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

$$= A'B' \cdot 1 + A'B \cdot 1 + AB \cdot 1$$

$$= A'B' + A'B + AB$$

$$= A'B' + A'B + A'B + AB$$

$$= A'(B' + B) + (A' + A)B$$

$$= A' \cdot 1 + 1 \cdot B$$

$$= A' + B$$

So sánh với giải pháp với giả thuyết trước đó:

$F(A,B,C) = A'C' + BC$ . Giải pháp nào rẻ hơn?

# Trường hợp “Don’t Care” (tt)

51

Tất cả các giá trị 1 phải được tính, nhưng X là tùy chọn và sẽ chỉ có giá trị 1 chỉ khi chúng giúp đơn giản biểu thức.

$$f = \sum m(1,3,5,7,9) + \sum d(6,12,13)$$

cd \ ab	00	01	11	10
00			X	
01	1	1	X	1
11	1	1		
10		X		

Simplify

$$f = a'd + c'd$$

cd \ ab	00	01	11	10
00			X	
01	1	1	X	1
11	1	1		
10		X		

**Câu 33:**

Tối thiểu hóa các hàm sau bằng bìa Các-nô:

$$F(A, B, C, D) = \sum (0, 2, 5, 6, 9, 11, 13, 14)$$

**Câu 34:**

Tối thiểu hóa các hàm sau bằng bìa Các-nô:

$$F(A, B, C, D) = \sum (0, 1, 3, 5, 8, 9, 13, 14, 15)$$

**Câu 35:**

Tối thiểu hóa các hàm sau bằng bìa Các-nô:

$$F(A, B, C, D) = \sum (2, 4, 5, 6, 7, 9, 12, 13)$$

**Câu 36:**

Tối thiểu hóa các hàm sau bằng bìa Các-nô:

$$F(A, B, C, D) = \prod (1, 4, 6, 7, 9, 10, 12, 13)$$

Câu 21. Tối thiểu hóa các hàm sau bằng bảng Các-nô.

$$F(A, B, C, D) = \sum (1, 3, 4, 5, 7, 9, 11, 14) + d(6, 12, 13)$$

**Bước 1. Lập bảng các nô**

**Bước 2. Tối giản các nô**

**Bước 3. Kết quả**

$$F(A, B, C, D) =$$

CD \ AB	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10



CD \ AB	00	01	11	10
00				
01				
11				
10				

**Câu 24:**

Cho hàm logic dạng tuyển sau:

$$Z = F(A, B, C) = \sum (1, 2, 3, 5, 7)$$

Hãy tối giản hóa bằng phương pháp đại số.

**Câu 25:**

Cho hàm logic dạng hội sau:

$$Z = F(A, B, C) = \prod (0, 4, 6)$$

Hãy tối giản hóa bằng phương pháp đại số.



**Câu 26:**

Cho hàm logic dạng tuyển sau:

$$Z = F(A, B, C, D) = \sum (1, 2, 4, 5, 6, 8, 9, 10, 14)$$

Xây dựng sơ đồ mạch logic thực hiện hàm chỉ dùng các phần tử NAND hai lối vào.

**Câu 27:**

Cho hàm số:

$$Y = F(A, B, C, D) = \prod (0, 1, 3, 7, 8, 9, 11, 12, 13, 15)$$

Xây dựng sơ đồ mạch logic thực hiện hàm chỉ dùng các phần tử NOR hai lối vào.

**Câu 28:**

Cho hàm logic dạng tuyển sau:

$$Z = F(A, B, C) = \sum (1, 2, 3, 5, 7)$$

Thiết kế mạch logic trên

Câu 21. Tối thiểu hóa các hàm sau bằng bảng Cáo-nô.

$$F(A, B, C, D) = \sum (1, 3, 4, 5, 7, 9, 11, 14) + d(6, 12, 13)$$

Câu 22. Cho hàm bool dùng bản đồ Cáo\_nô đề:

$$F(A, B, C, D) = \sum (0, 1, 6, 8, 9, 11, 14, 15) + d(2, 3, 10)$$

- Xác định dạng chuẩn tổng các tích của hàm f ( gọi là hàm g)
- Xác định dạng chuẩn tích các tổng của hàm f ( gọi là hàm h)
- So sánh hai hàm g và h
- Xây dựng sơ đồ hàm g và đánh giá OUTPUTs của mạch logic trên.

Câu 23. Cho hàm bool dùng bản đồ Cáo\_nô đề:

$$F(A, B, C, D) = \sum (3, 4, 5, 7, 10, 12, 13) + d(8, 9, 11)$$

- Xác định dạng chuẩn tổng các tích của hàm f ( gọi là hàm g)
- Xác định dạng chuẩn tích các tổng của hàm f ( gọi là hàm h)
- So sánh hai hàm g và h
- Xây dựng sơ đồ hàm g và đánh giá OUTPUTs của mạch logic trên.

Câu 24. Cho hàm bool  $f(A, B, C, D) = \prod (3, 4, 5, 6, 10, 12, 13) + d(8, 11)$ , Dùng bản đồ Karnaugh để rút gọn theo :

- Dạng tổng các tích của hàm f
- Dạng tích các tổng của hàm f

### 3.4 Cho hàm bool:

$f(A,B,C,D) = \sum (0,1,2,6,8,9,11,14,15) + d(3,10)$ , dùng bản đồ Karnaugh để:

- Xác định dạng chuẩn tổng các tích của hàm f (gọi là hàm g)
- Xác định dạng chuẩn tích các tổng của hàm f (gọi là hàm h)
- So sánh hai hàm g và h
- Vẽ sơ đồ mạch hàm g mà chỉ sử dụng cổng NAND.

### 3.5 Cho hàm bool:

$f(A,B,C,D) = \sum (3,4,5,7,10,12,13) + d(8,9,11)$ , Dùng bản đồ Karnaugh để:

- Xác định dạng chuẩn tổng các tích của hàm f (gọi là hàm g)
- Xác định dạng chuẩn tích các tổng của hàm f (gọi là hàm h)
- So sánh hai hàm g và h
- Vẽ sơ đồ mạch hàm g mà sử dụng cổng NOR.

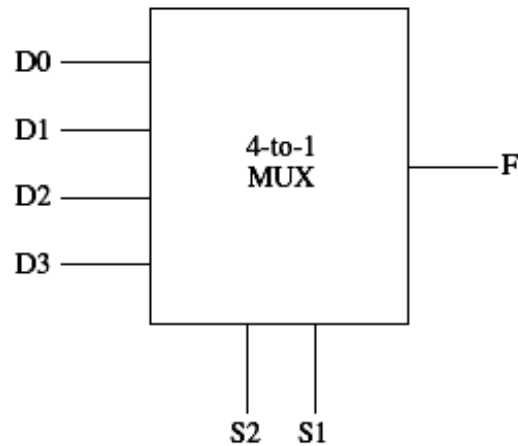
# Mạch tổ hợp

## ➤ Khái niệm

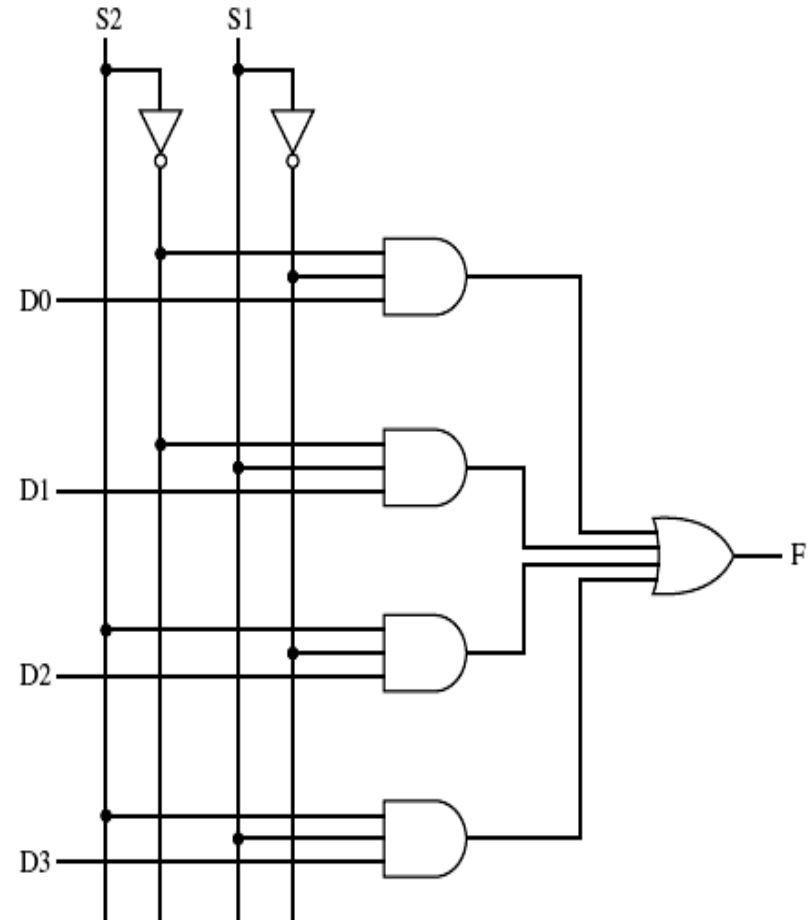
- Mạch tổ hợp (combinational circuit) là mạch logic trong đó tín hiệu ra chỉ phụ thuộc tín hiệu vào ở thời điểm hiện tại.
- Là mạch không nhớ (memoryless) và được thực hiện bằng các cổng logic cơ bản
- Mạch tổ hợp được cài đặt từ 1 hàm hoặc bảng chân trị cho trước
- Được ứng dụng nhiều trong thiết kế mạch máy tính

# Bộ dồn kênh (Multiplexer)

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

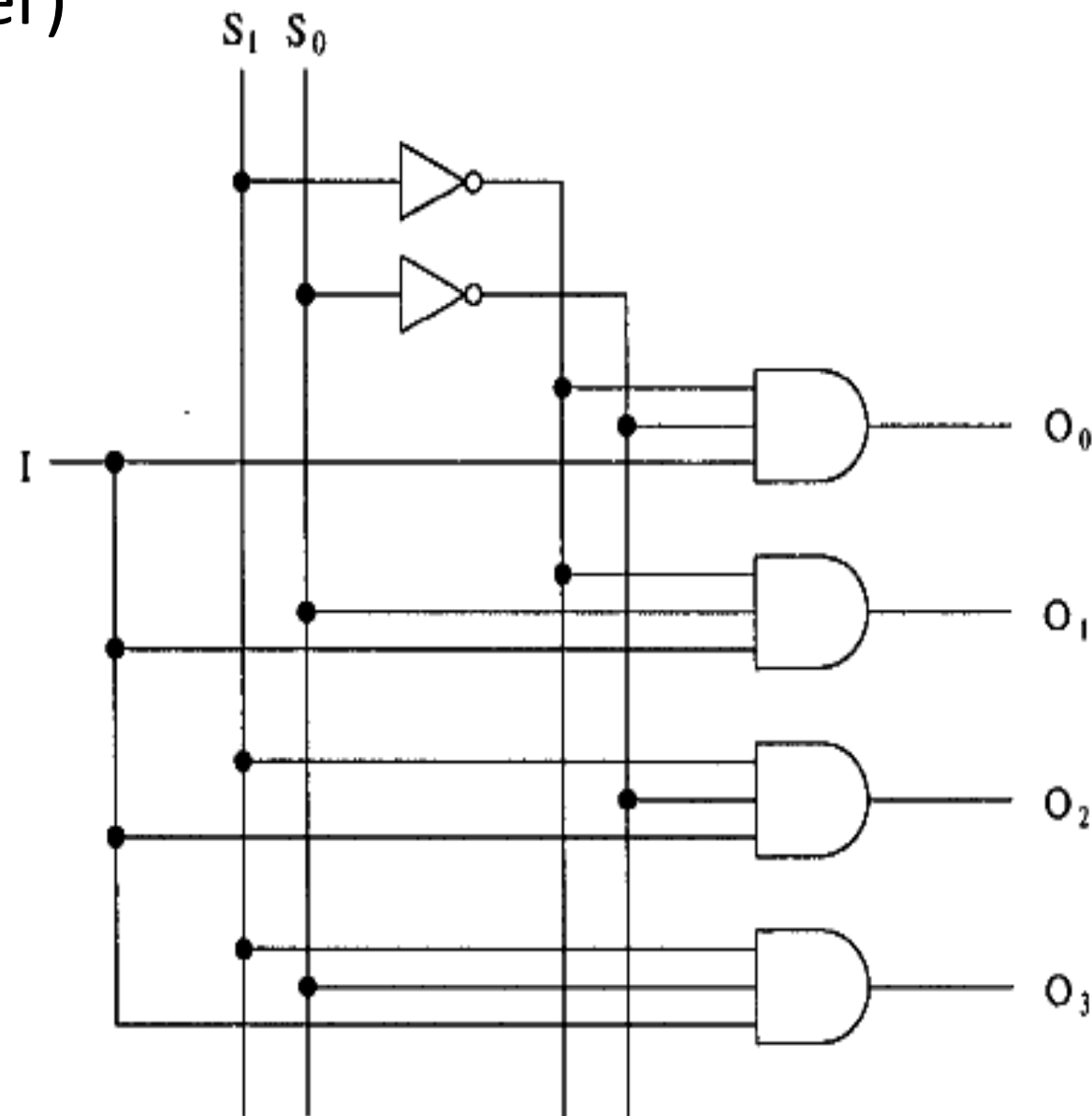
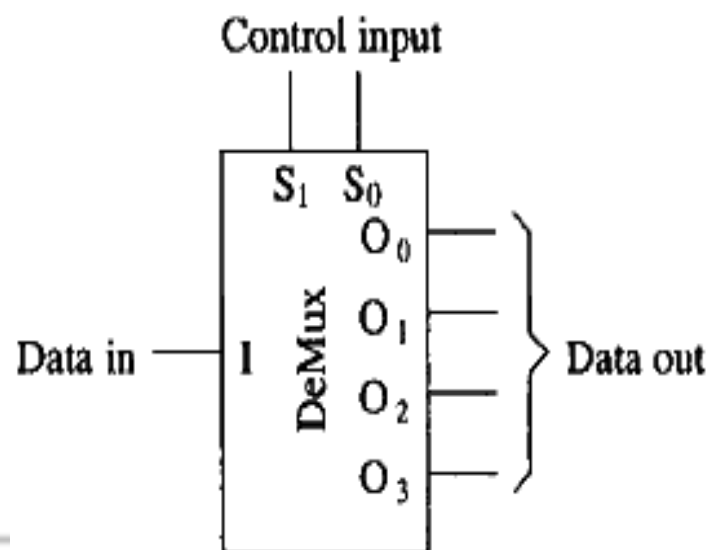


S2	S1	F
0	0	D0
0	1	D1
1	0	D2
1	1	D3



# Bộ phân kênh (Demultiplexer)

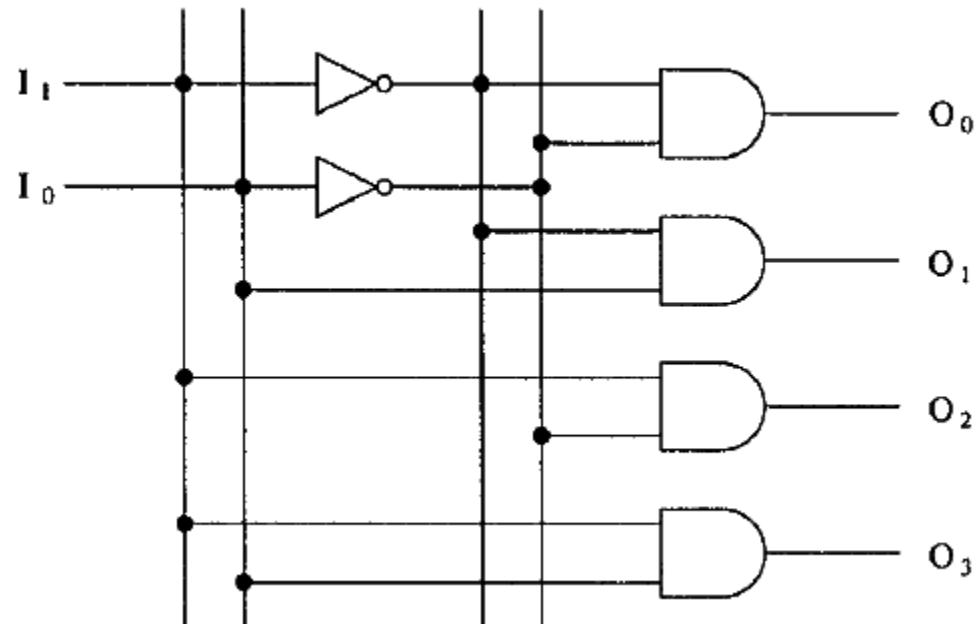
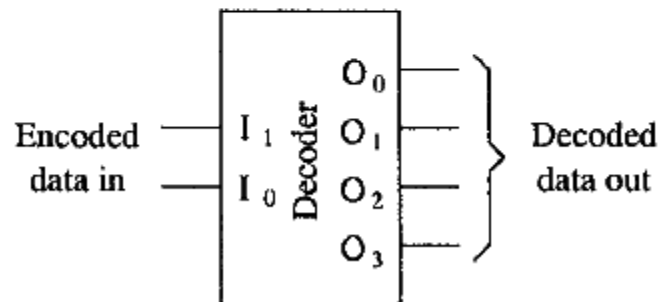
- Ngược với bộ dồn kênh
- Tín hiệu điều khiển (S) sẽ chọn đầu ra nào kết nối với đầu vào (I)
- Ví dụ: Demux 1-to-4



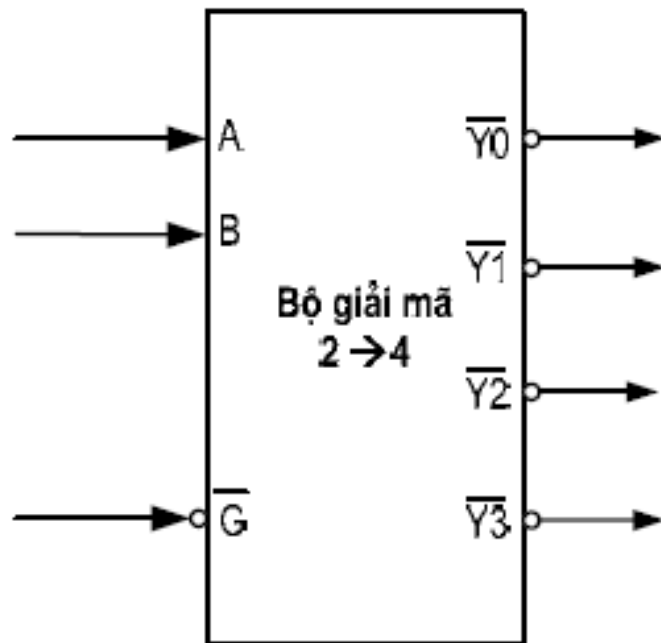
## Bộ giải mã (Decoder)

- Bộ giải mã chọn một trong  $2^n$  đầu ra (O) tương ứng với một tổ hợp của n đầu vào (I)
- Ví dụ : Mạch giải mã 2 ra 4

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



# Bộ giải mã 2- $\rightarrow$ 4



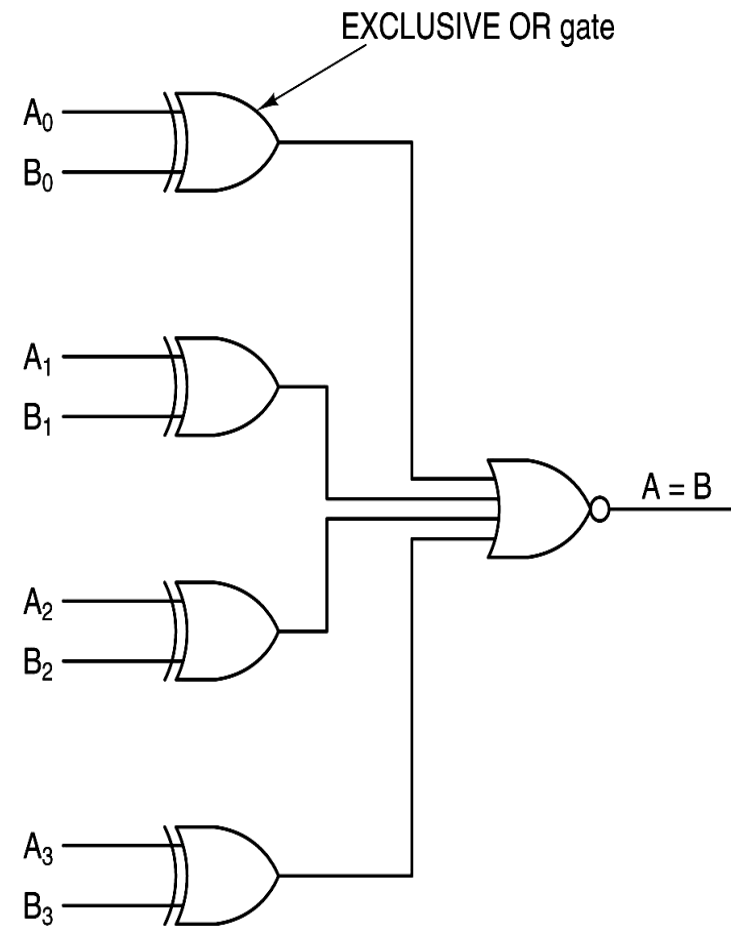
$\overline{G}$	$B$	$A$	$\overline{Y_0}$	$\overline{Y_1}$	$\overline{Y_2}$	$\overline{Y_3}$
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	x	x	1	1	1	1



# Mạch so sánh (Comparator)

- So sánh các bit của 2 ngõ vào và xuất kết quả 1 nếu bằng nhau.
- Ví dụ : Mạch so sánh 4 bit dùng các cổng XOR

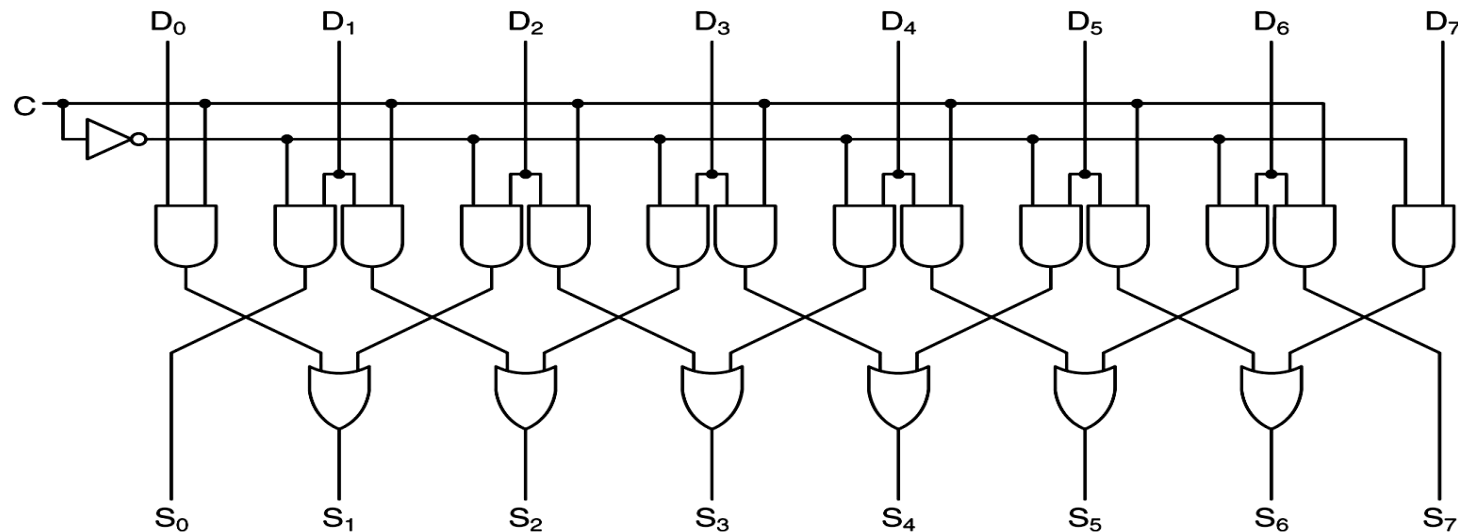
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0



# Mạch tính toán

## ➤ Mạch dịch (Shifter)

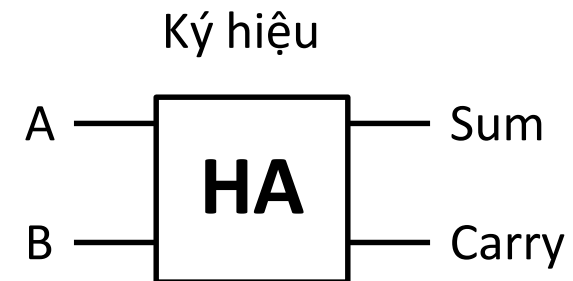
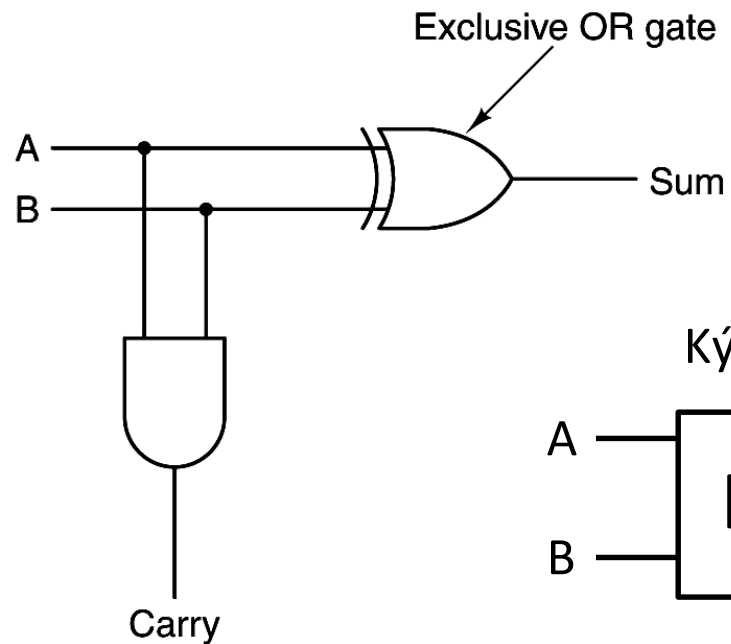
- Dịch các tín hiệu sang trái hoặc phải 1 vị trí. Ứng dụng cho phép nhân/chia cho 2.
- Ví dụ : mạch dịch 8 bit với tín hiệu điều khiển chiều dịch trái ( $C=0$ ) hay phải ( $C=1$ )



## ➤ Mạch cộng bán phần (Half adder)

- Cộng 2 bit đầu vào thành 1 bit đầu ra và 1 bit nhớ

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

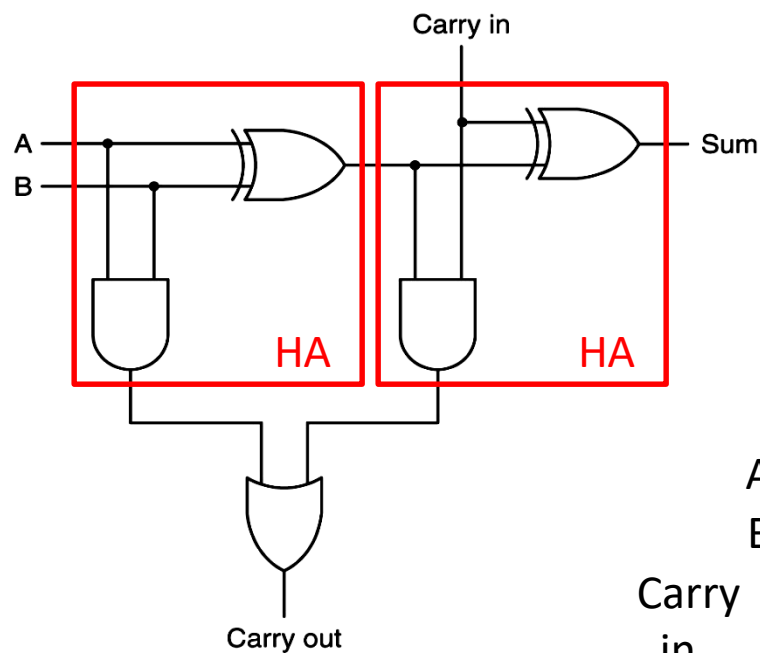


## ➤ Mạch cộng toàn phần (Full adder)

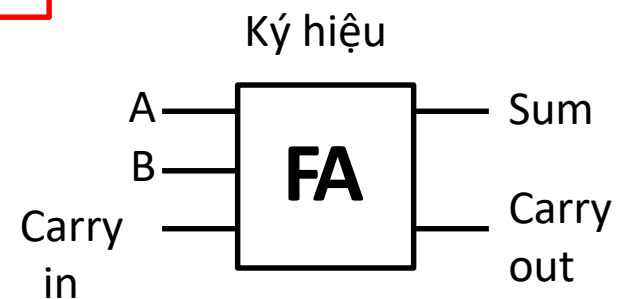
- Cộng 3 bit đầu vào thành 1 bit đầu ra và 1 bit nhớ
- Cho phép xây dựng bộ cộng nhiều bit

A	B	Carry in	Sum	Carry 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

(a)

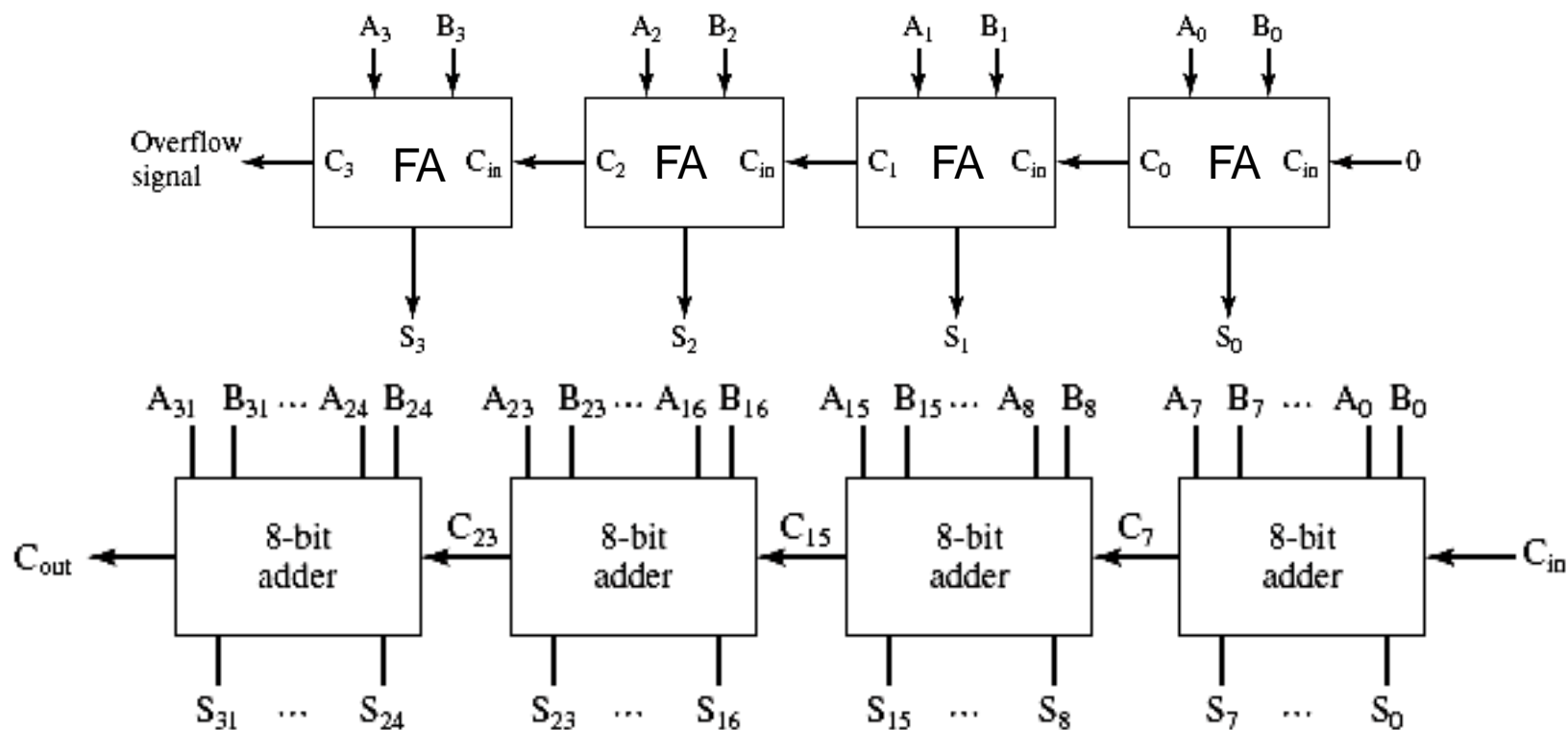


(b)



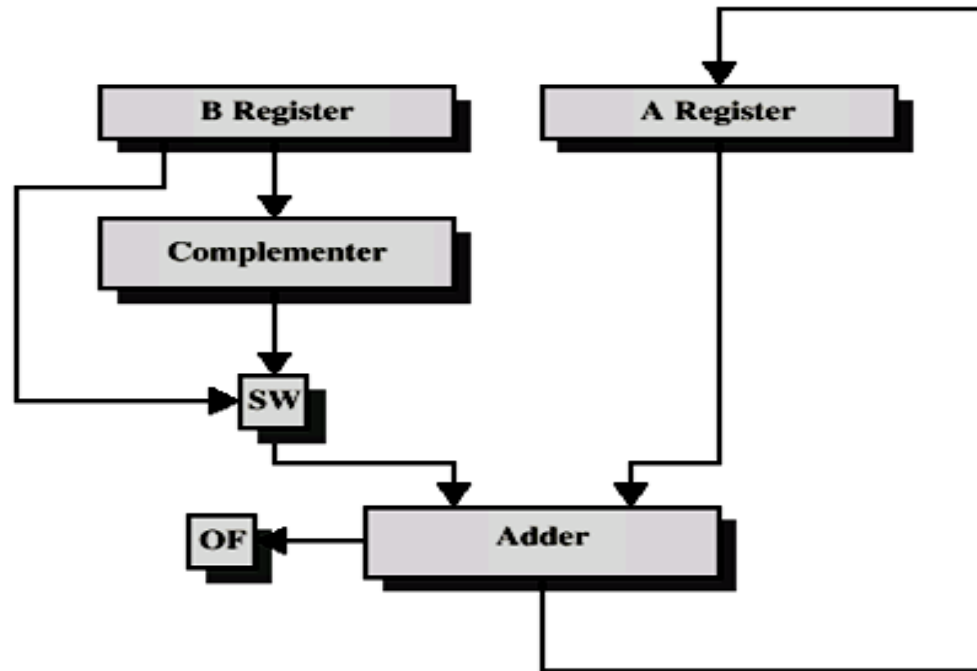
## ➤ Mạch cộng nhiều bit

- Ghép từ nhiều bộ cộng toàn phần



## ➤ Mạch cộng và trừ

- Mạch trừ: Đổi sang số bù 2 rồi cộng
- Có thể dùng chung mạch cộng để thực hiện phép trừ



OF = overflow bit  
SW = Switch (select addition or subtraction)

- Ví dụ ALU 1 bit

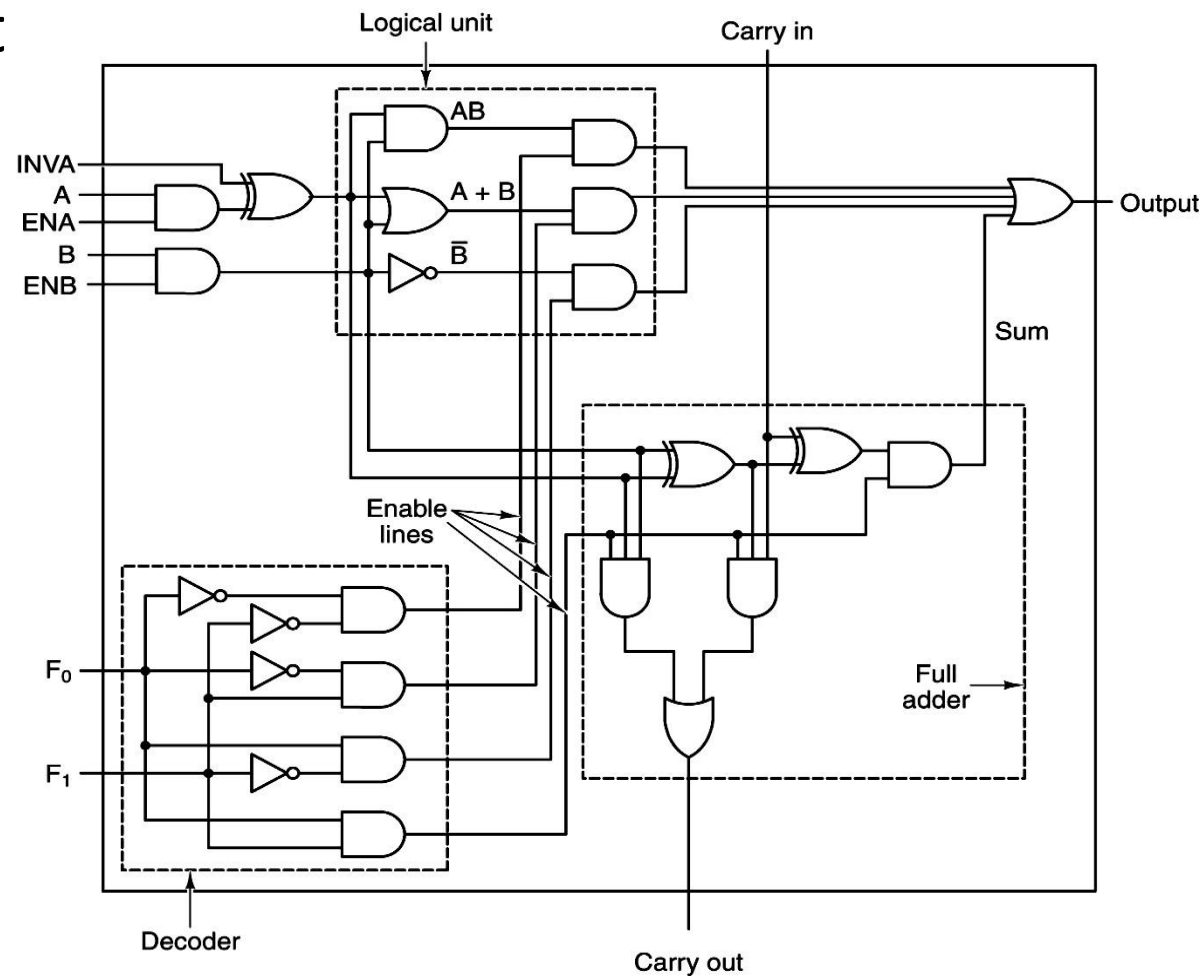
$F_0F_1$	Functions
00	<b>A AND B</b>
01	<b>A OR B</b>
10	
11	<b>A + B</b>

Điều kiện  
bình thường

ENA=1

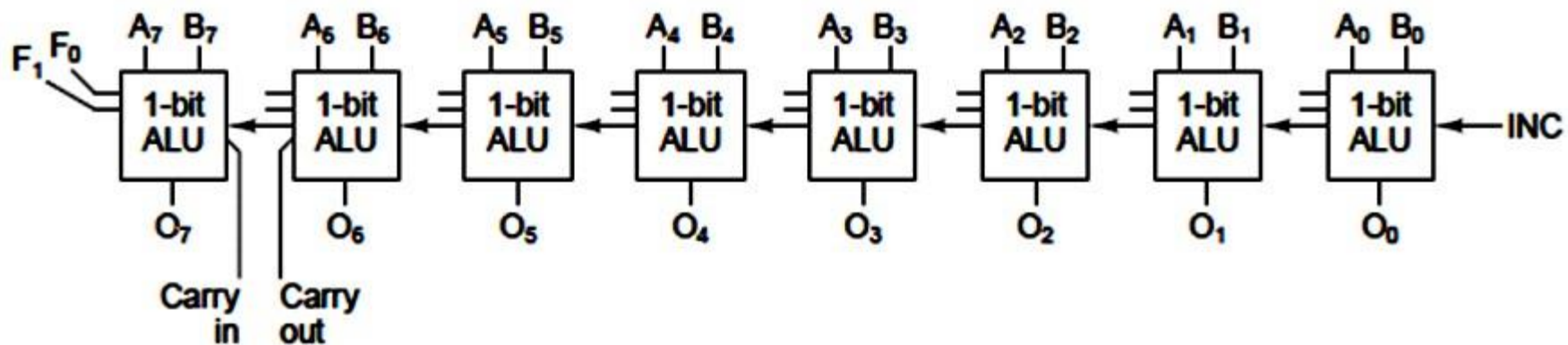
ENB=1

INVA=0



## ➤ ALU 8 bit

- Ví dụ tạo 1 mạch ALU 8 bit bằng cách ghép 8 bộ ALU 1 bit ở ví dụ trước





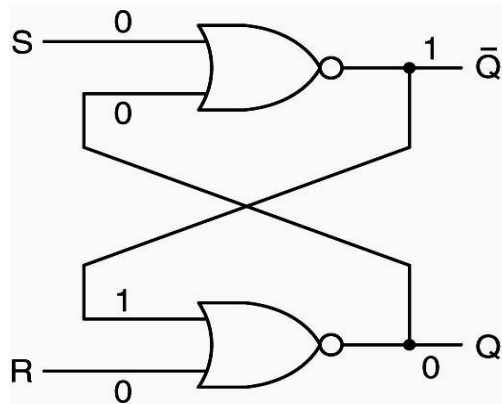
# Mạch tuần tự

## ➤ Khái niệm

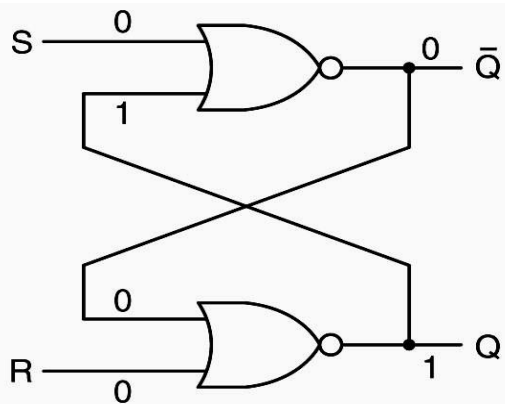
- Mạch tuần tự (sequential circuit) là mạch logic trong đó tín hiệu ra phụ thuộc tín hiệu vào ở hiện tại và quá khứ
- Là mạch có nhớ, được thực hiện bằng phần tử nhớ (Latch, Flip-Flop) và có thể kết hợp với các cổng logic cơ bản
- Ứng dụng làm bộ nhớ, thanh ghi, mạch đếm,... trong máy tính

## ➤ Mạch chốt (Latch)

- Dùng 2 cổng NOR mắc hồi tiếp với nhau. S, R là ngõ vào, Q và  $\bar{Q}$  là ngõ ra.
- Đây là mạch chốt SR. Nó có thể ở 1 trong 2 trạng thái  $Q=1$  hoặc  $Q=0$  khi  $S=R=0$ .
- Khi  $S=1 \rightarrow Q=1$  bất kể trạng thái trước đó (set)
- Khi  $R=1 \rightarrow Q=0$  bất kể trạng thái trước đó (reset)



(a)



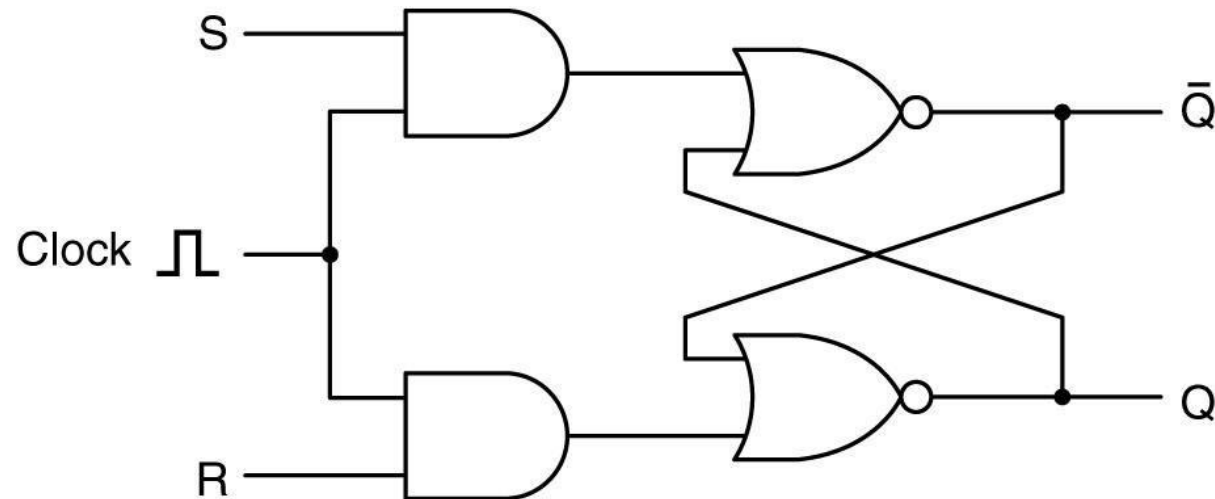
(b)

A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

(c)

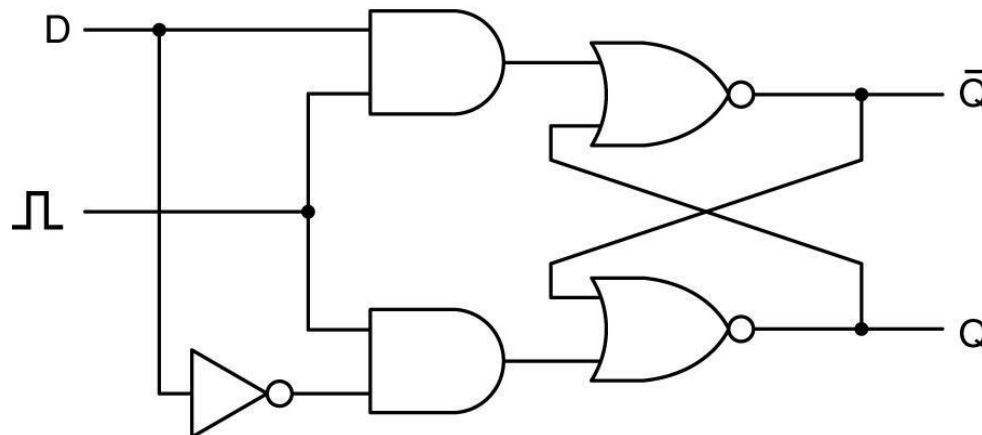
## ➤ Mạch chốt SR có xung Clock

- Thêm vào mạch chốt SR 2 cổng AND nối với xung đồng hồ để điều khiển trạng thái mạch chốt tại thời điểm xác định
- Tín hiệu vào chỉ có tác dụng khi xung clock=1 (mức cao)



## ➤ Mạch chốt D có xung Clock

- Mạch chốt SR sẽ ở trạng thái không xác định khi  $S=R=1$
- Khắc phục bằng cách chỉ dùng 1 tín hiệu vào và đấu nối R với S qua cổng NOT
- Đây chính là mạch bộ nhớ 1 bit với D là ngõ vào, Q là ngõ ra



**Câu 44.** Trình khái niệm mạch tổ hợp và Xây dựng mạch logic bộ dồn kênh (Multiplexer)? Lấy ví dụ minh họa?

**Câu 45.** Trình khái niệm mạch tổ hợp và Xây dựng mạch logic bộ phân kênh (Demultiplexer)? Lấy ví dụ minh họa?

**Câu 46.** Trình khái niệm mạch tổ hợp và Xây dựng mạch logic bộ giải mã (Decoder)? Lấy ví dụ minh họa?

**Câu 47.** Trình khái niệm mạch tổ hợp và Xây dựng mạch logic mạch so sánh (Comparator)? Lấy ví dụ minh họa?

**Câu 48.** Trình khái niệm mạch tính toán và Xây dựng mạch dịch (Shifter)? Lấy ví dụ minh họa?

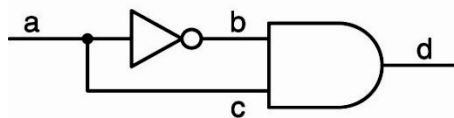
**Câu 49.** Trình khái niệm mạch tính toán và Xây dựng mạch cộng bán phần (Half adder)? Lấy ví dụ minh họa?

**Câu 50.** Trình khái niệm mạch tính toán và Xây dựng mạch cộng toàn phần (Full adder)? Lấy ví dụ minh họa?

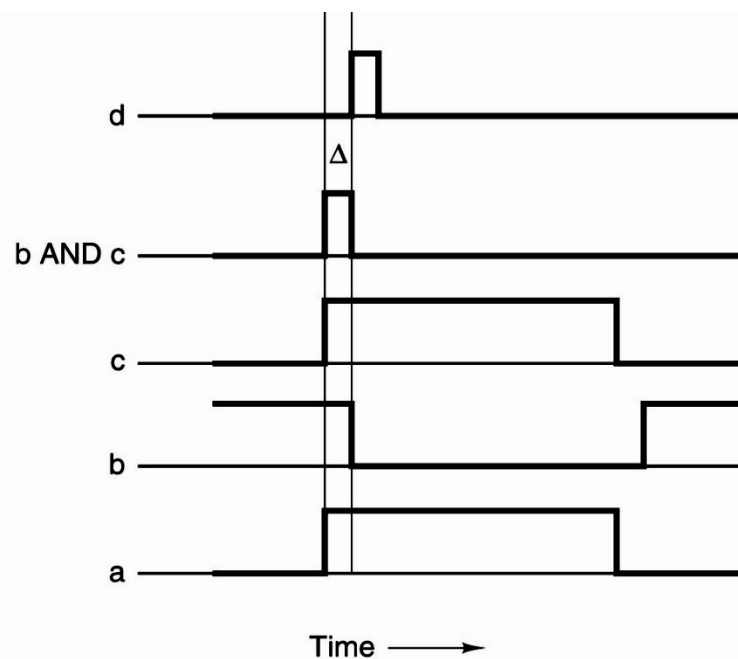
---

## ➤ Flip-Flop

- Trong thực tế ta muốn bộ nhớ chỉ được ghi trong 1 khoảng thời gian nhất định → cần thiết kế mạch xung Clock tác dụng theo cạnh (lên hoặc xuống)



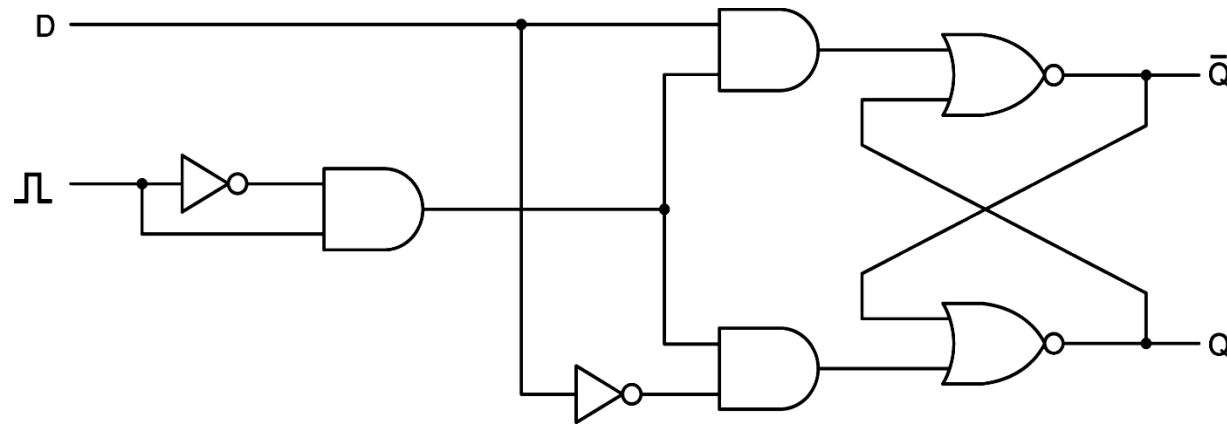
(a)



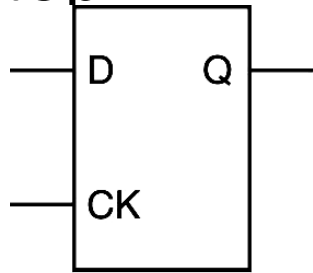
(b)

## ➤ D Flip-Flop

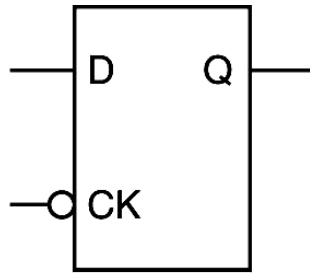
- Là mạch chốt D có xung Clock điều khiển bằng Flip-flop
- Phân biệt:
  - Flip-flop: edge triggered
  - Latch: level triggered



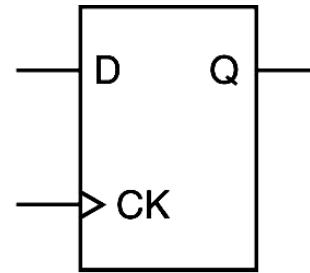
## ➤ Ký hiệu mạch chốt và Flip-Flop



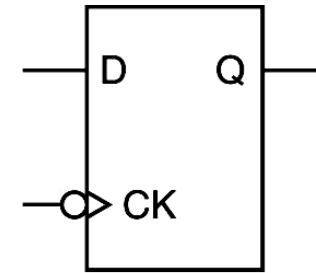
(a)



(b)



(c)



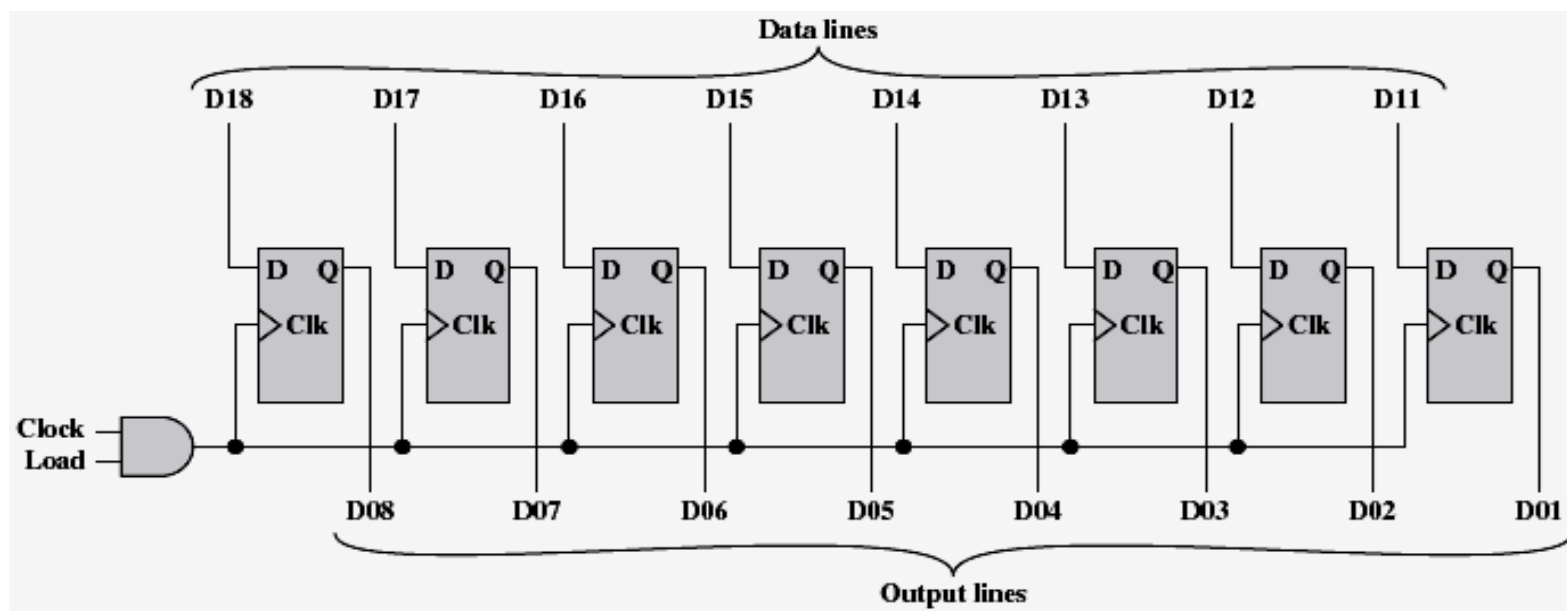
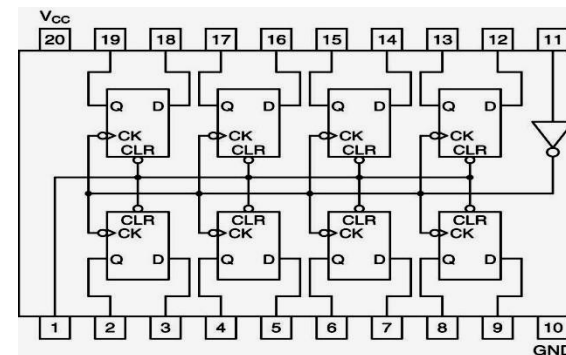
(d)

- a) Mạch chốt D tác động theo mức 1 (clock=1)
- b) Mạch chốt D tác động theo mức 0 (clock=0)
- c) Flip-flop D tác động theo cạnh lên (clock=  $0 \rightarrow 1$ )
- d) Flip-flop D tác động theo cạnh xuống (clock=  $1 \rightarrow 0$ )



## ➤ Thanh ghi (Register)

- Việc ghép nối nhiều ô nhớ 1 bit tạo thành các ô nhớ lớn hơn
- Ví dụ : Vi mạch 74273 gồm 8 D flip-flop ghép nối lại tạo thành 1 thanh ghi 8 bit



## ➤ Ví dụ : mạch bộ nhớ 4 ô x 3 bit

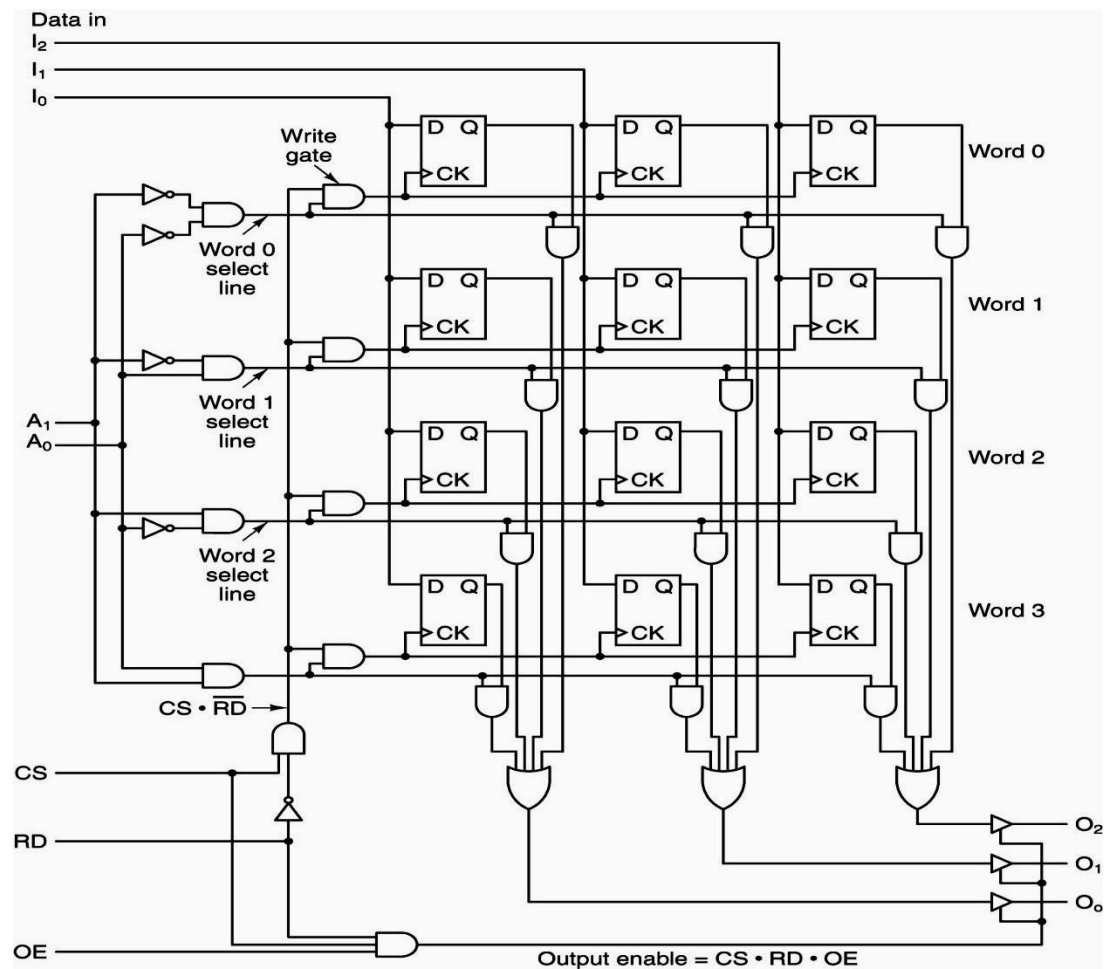
- A: Address
- I: Input data
- O: Output data
- CS: Chip select
- RD: Read/write
- OE: Output enable

Write:

CS=1, RD=0, OE=0

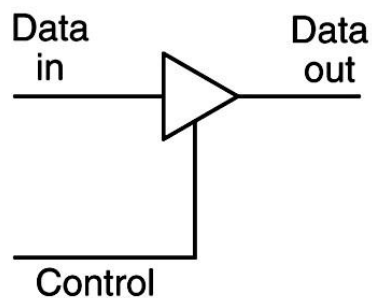
Read:

CS=1, RD=1, OE=1



## ➤ Mạch đệm (Buffer)

- Dùng để đọc dữ liệu đồng bộ trên nhiều đường tín hiệu bằng 1 đường điều khiển riêng.
- Sử dụng các cổng 3 trạng thái (tri-state devices)



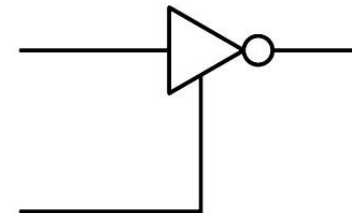
(a)



(b)



(c)

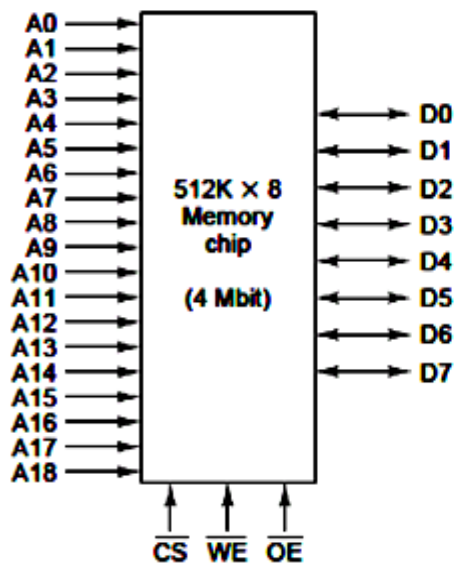


(d)

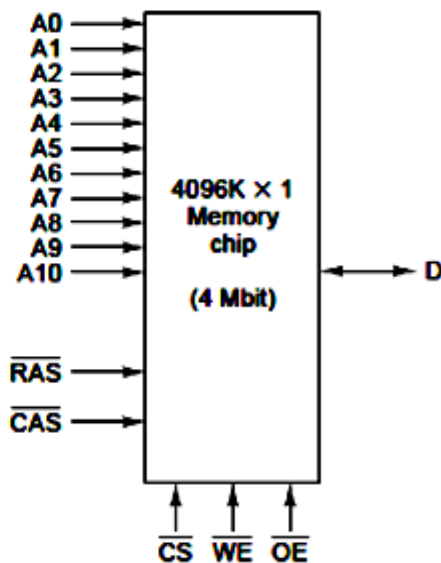
- a. Buffer không đảo.
- b. Khi control ở mức cao (=1).
- c. Khi control ở mức thấp (=0).
- d. Buffer đảo.

## ➤ Chip bộ nhớ

- Bộ nhớ thường gồm nhiều ô nhớ ghép lại
- Ví dụ 1: Chip bộ nhớ 4Mbit có thể tạo thành từ 512K ô 8 bit hoặc ma trận 2048x2048 ô 1 bit



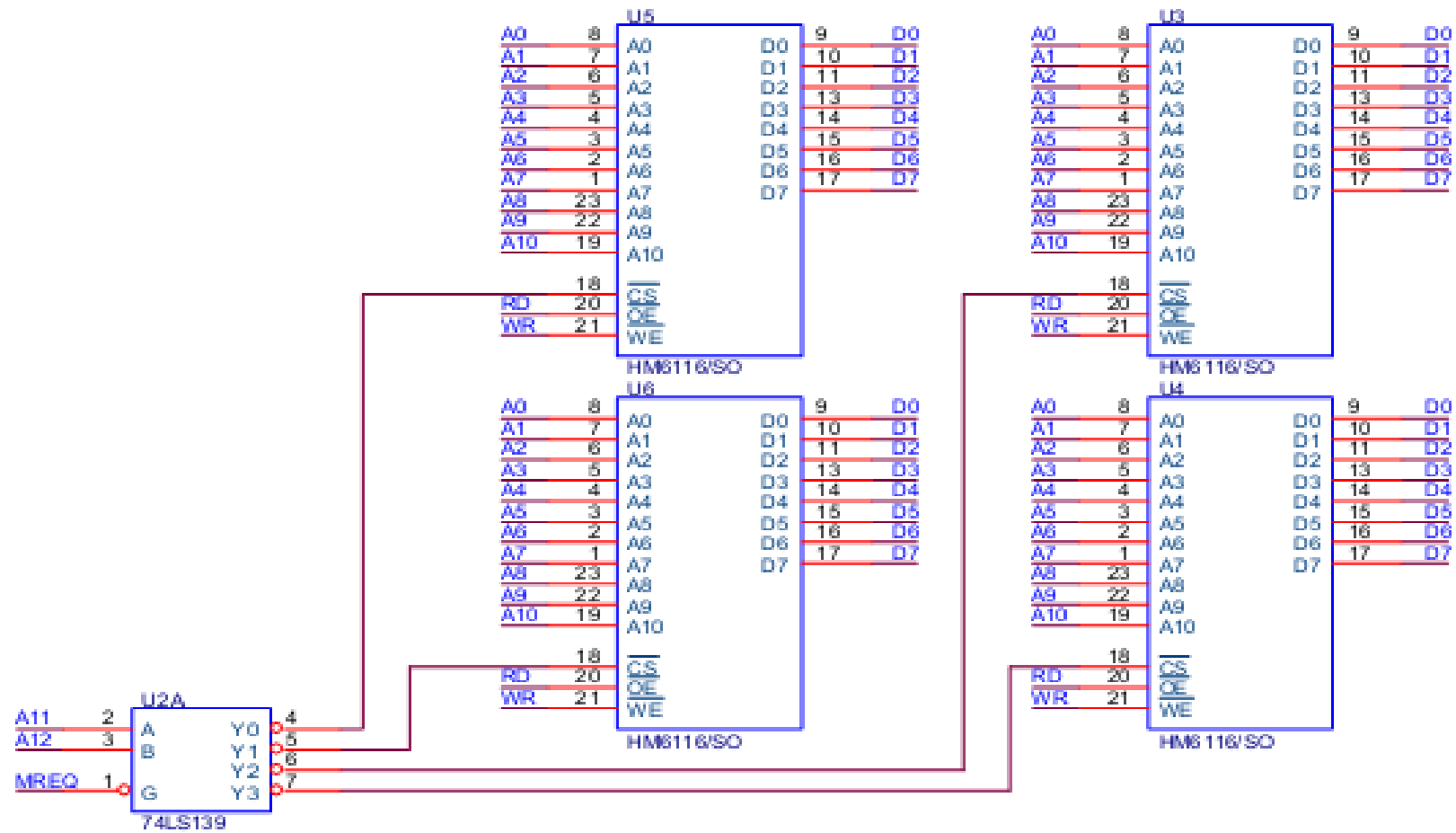
(a)



(b)

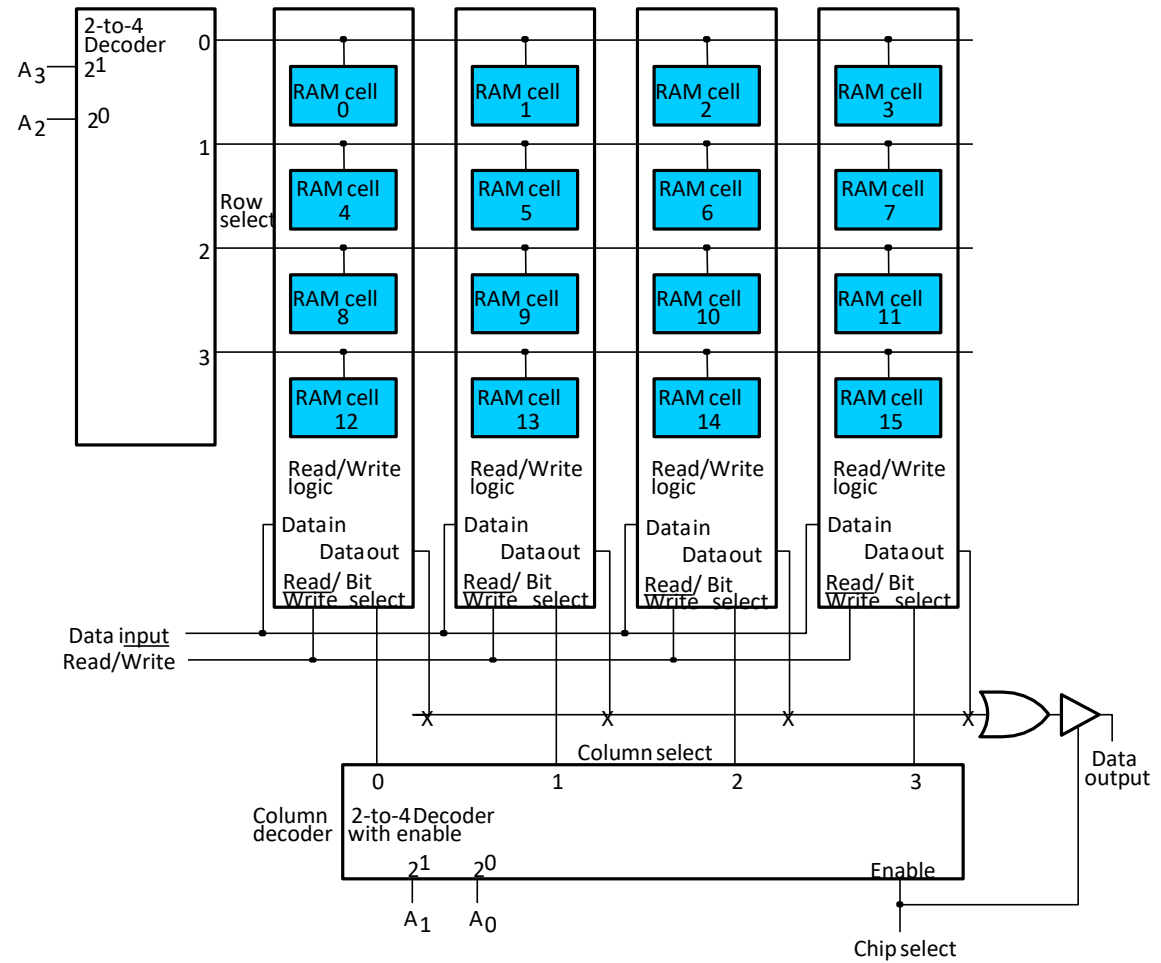
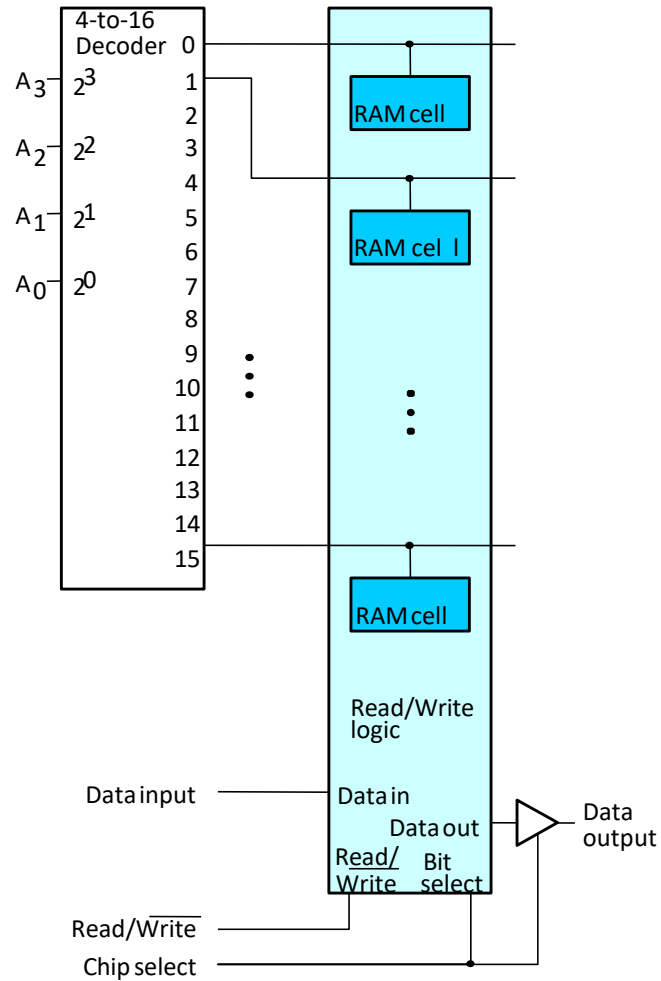
### Ghi chú:

RAS: Row Address Strobe  
CAS: Column Address Strobe  
CS: Chip select  
WE: Write enable  
OE: Output enable  
D: Data  
A: Address



## ➤ Chip bộ nhớ (tiếp)

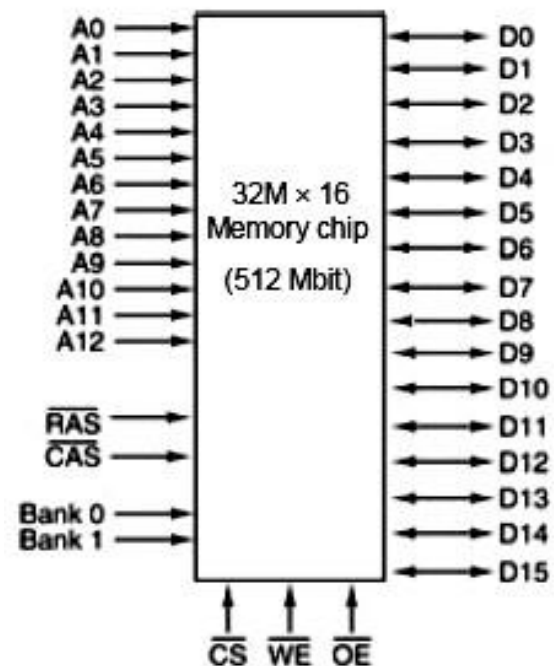
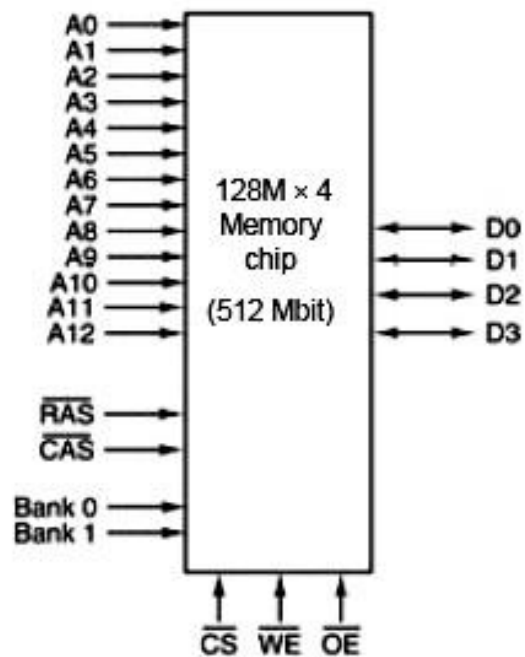
- Mạch giải mã địa chỉ  $n$  bit có thể giải mã cho  $2^n$  ô nhớ  
→ cần  $n$  chân tín hiệu địa chỉ
- Có thể giảm kích thước bộ giải mã còn bằng cách tổ chức thành ma trận các ô nhớ → sử dụng 2 bộ giải mã cho hàng và cột riêng
- Ví dụ: bộ nhớ 16 ô cần 4 bit địa chỉ có thể tổ chức thành ma trận  $4 \times 4$  → chỉ cần giải mã 2 bit cho hàng và 2 bit cho cột.
- Có thể ghép địa chỉ hàng và cột chung 1 chân tín hiệu → giảm số chân kết nối bus địa chỉ
- Nhược điểm: cần gấp đôi thời gian truy cập bộ nhớ



## ➤ Chip bộ nhớ (tiếp)

○ Ví dụ 2: Chip bộ nhớ 512Mbit = 4 bank 128Mbit

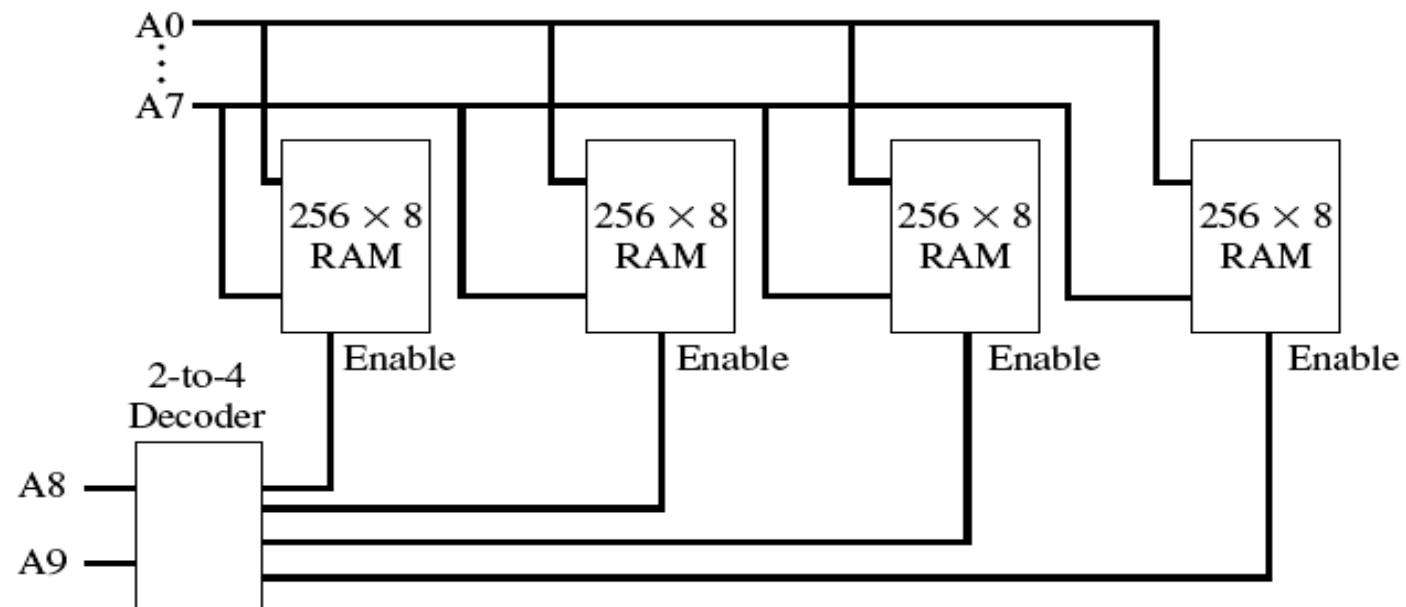
- Ma trận 13 bit hàng \* 12 bit cột \* ô nhớ 4 bit
- Ma trận 13 bit hàng \* 10 bit cột \* ô nhớ 16 bit





## ➤ Tổ chức bộ nhớ

- Bộ nhớ thường gồm nhiều chip nhớ dung lượng nhỏ ghép lại
- Dùng 1 mạch giải mã địa chỉ để chọn chip khi truy cập
- Ví dụ: Bộ nhớ 1KB gồm 4 chip 256B ghép lại



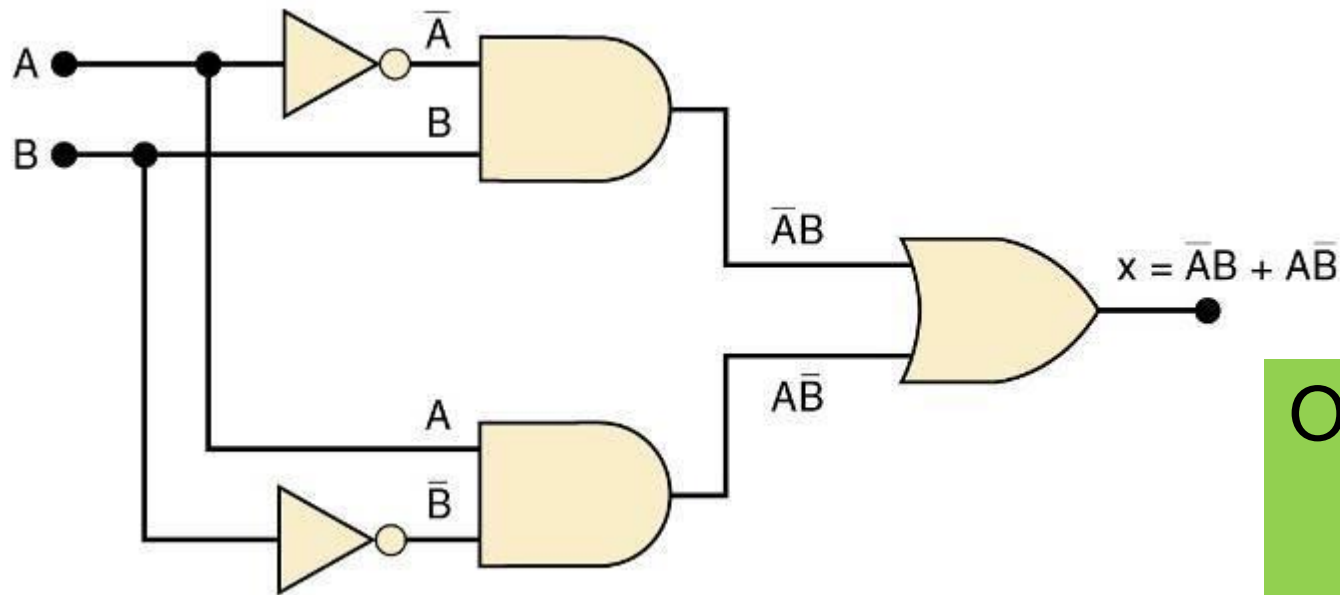
65

# XOR & XNOR Gate

# Exclusive OR

66

- Exclusive OR (XOR) cho ra kết quả HIGH khi hai đầu vào khác nhau

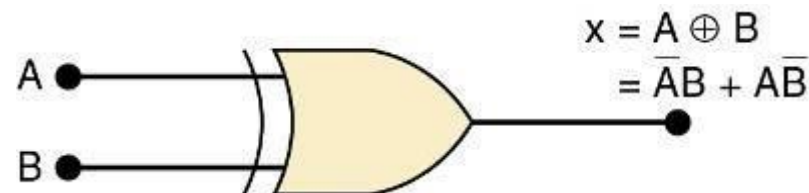


A	B	x
0	0	0
0	1	1
1	0	1
1	1	0

Output expression:

$$x = \bar{A}B + A\bar{B}$$

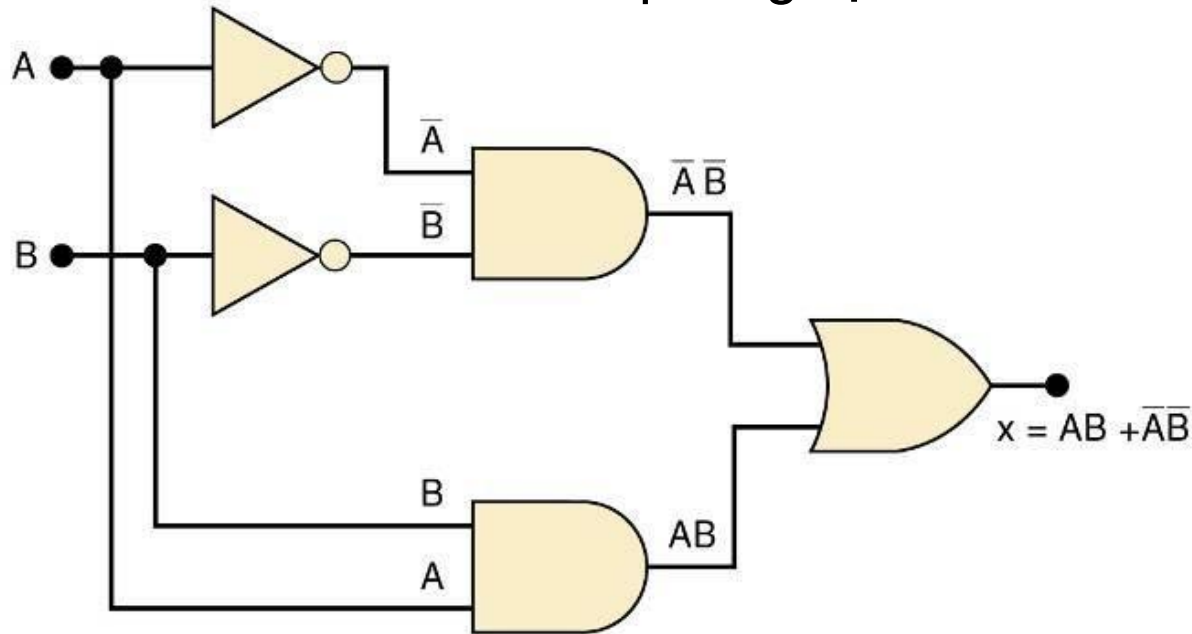
Ký hiệu XOR Gate



# Exclusive NOR

67

- Exclusive NOR (XNOR) cho ra kết quả HIGH khi hai đầu vào giống nhau
  - ▣ XOR và XNOR cho ra kết quả ngược nhau

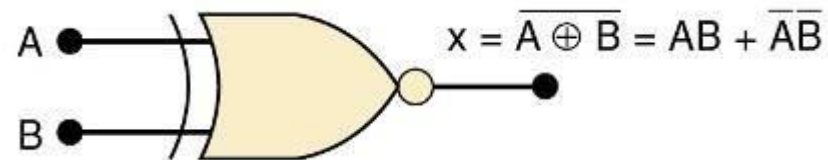


A	B	x
0	0	1
0	1	0
1	0	0
1	1	1

Output

$$x = AB + \bar{A}\bar{B}$$

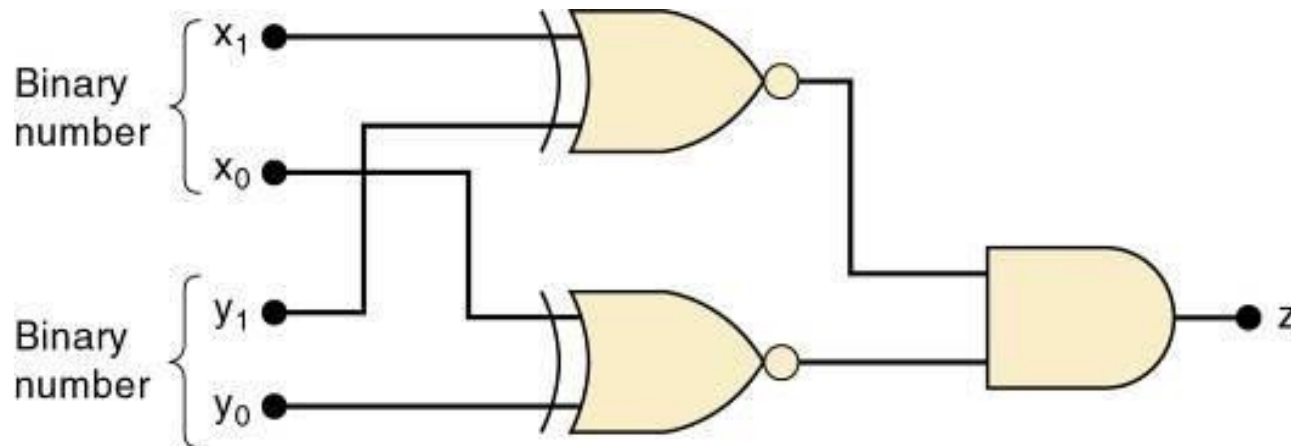
Ký hiệu XNOR



# Ví dụ

68

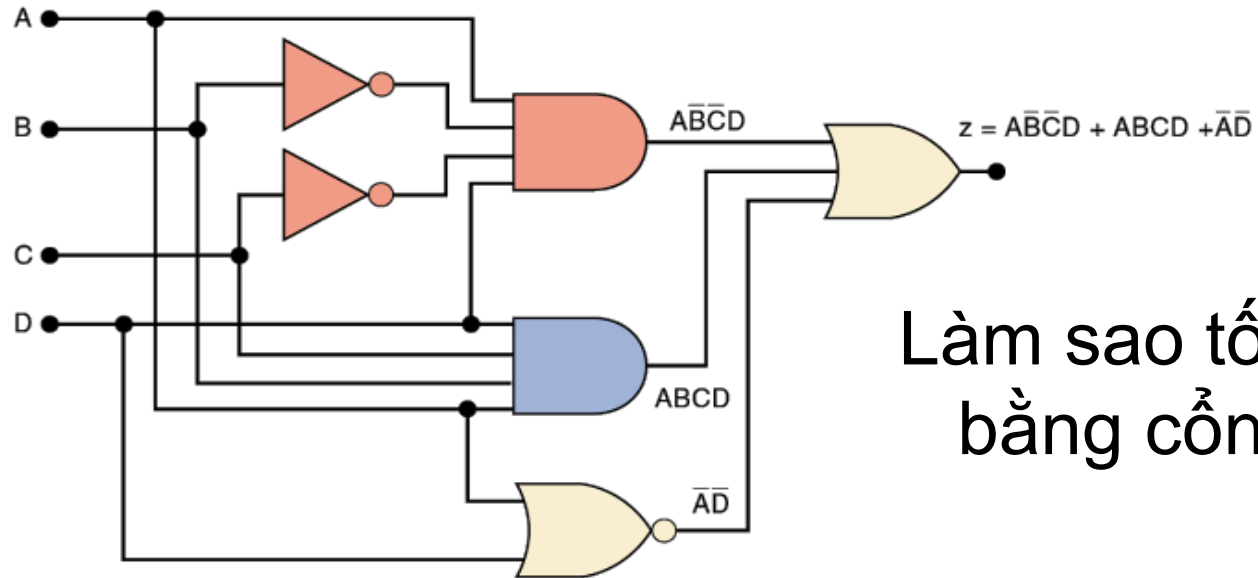
- Thiết kế một mạch để phát hiện ra 2 số nhị phân 2 bit có bằng nhau hay không



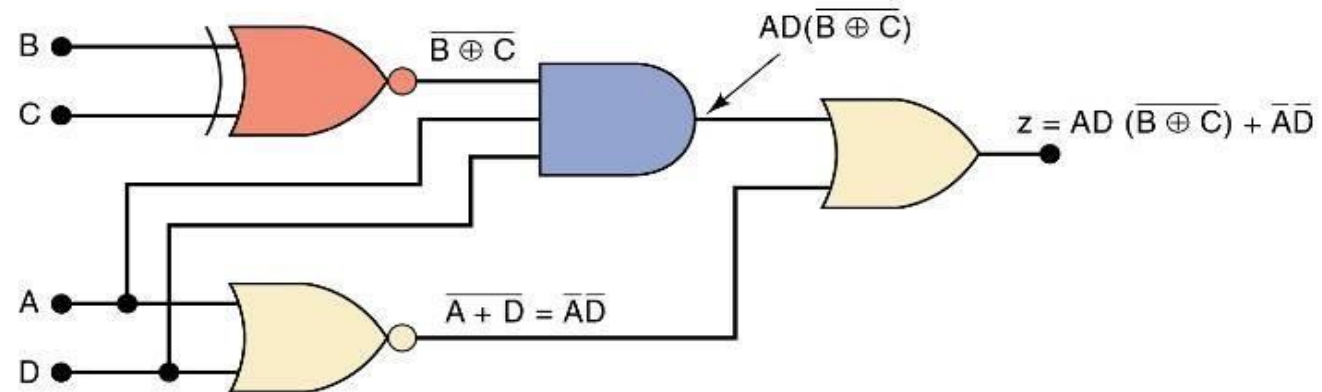
$x_1$	$x_0$	$y_1$	$y_0$	$z$ (Output)
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

# XOR & XNOR

69



Làm sao tối ưu mạch  
bằng cổng **XNOR**



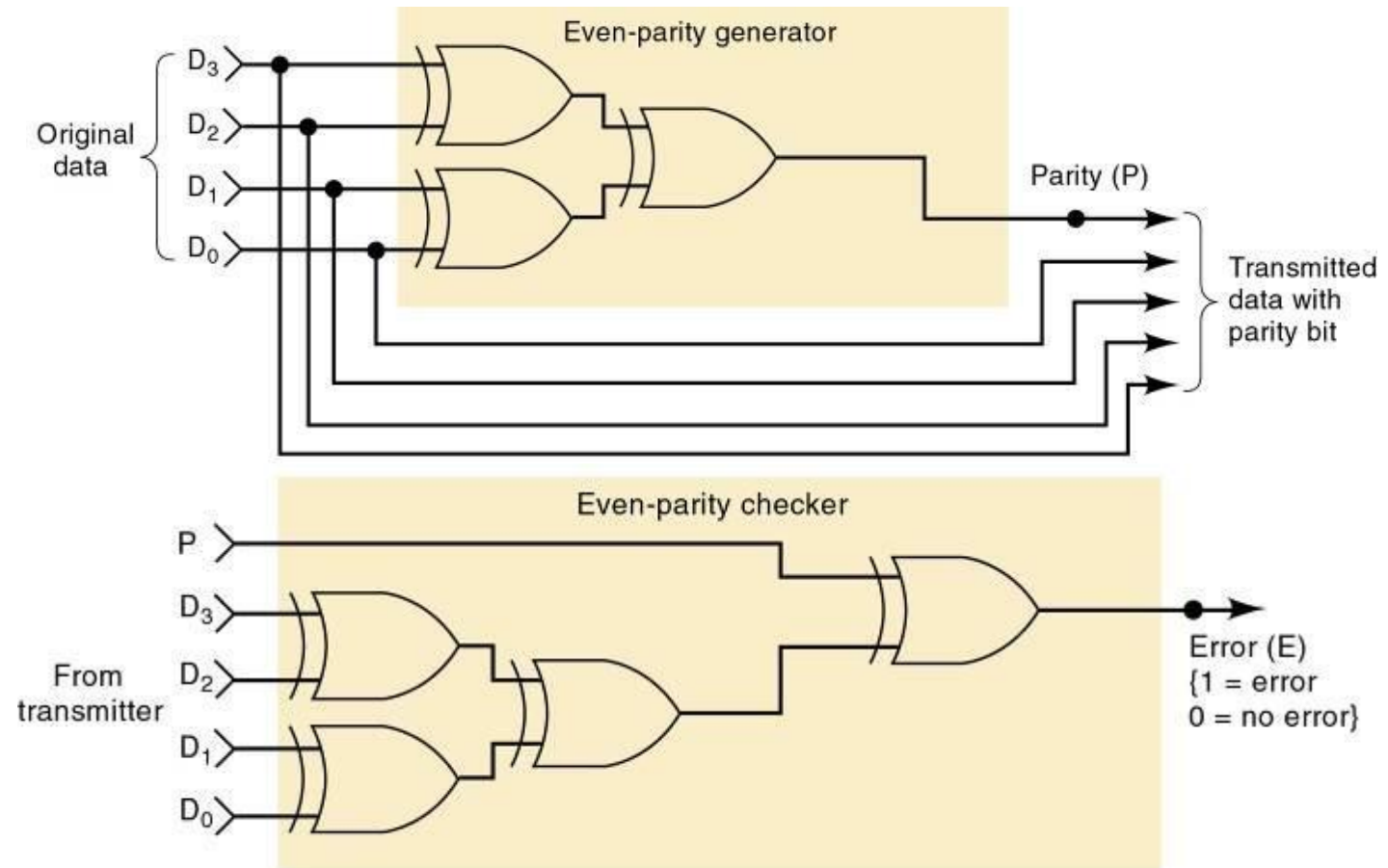
70

# Một số mạch logic

# Parity generator and checker

71

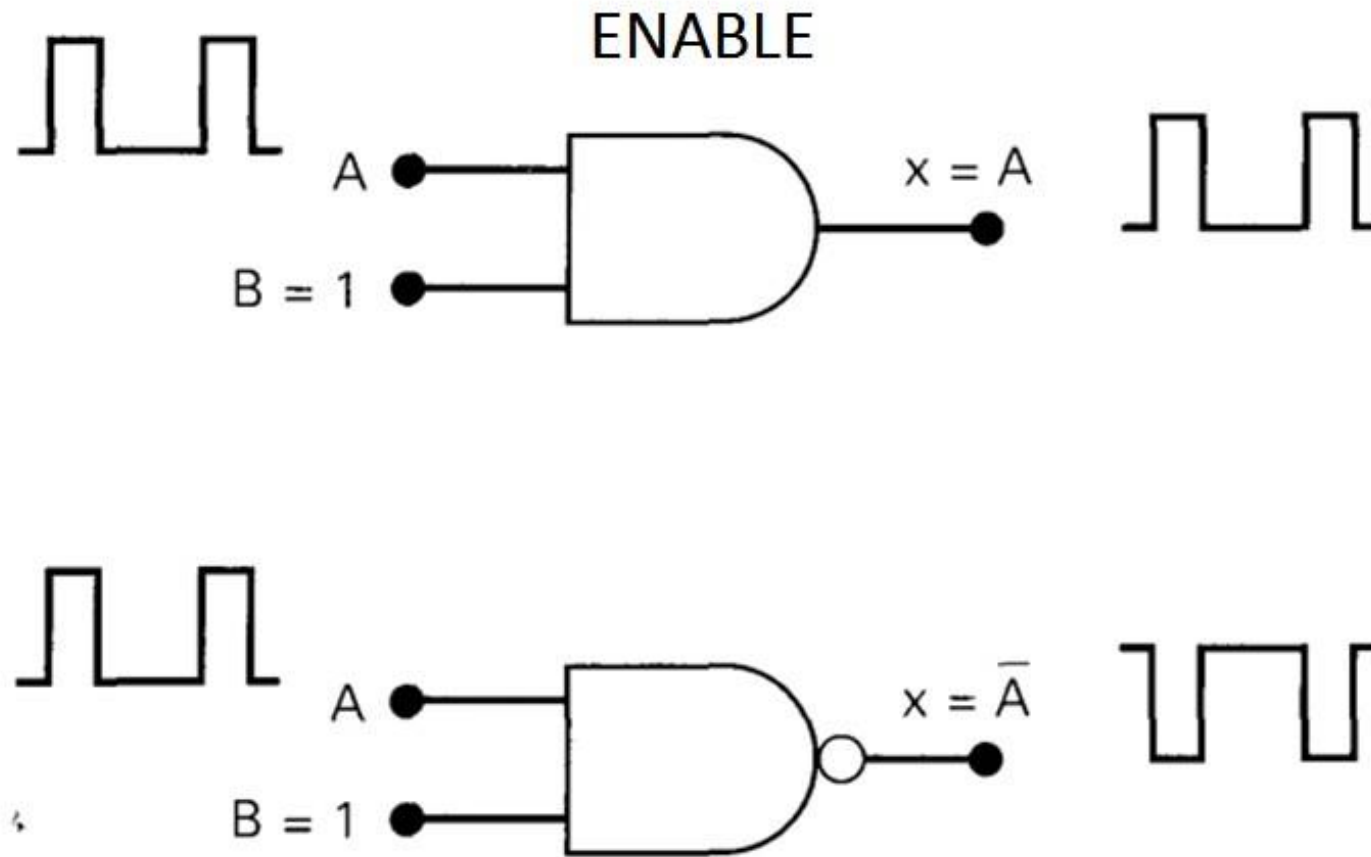
- Cổng XOR và XNOR rất hữu dụng trong các mạch với mục đích phát hiện và kiểm tra parity





# Mạch enable

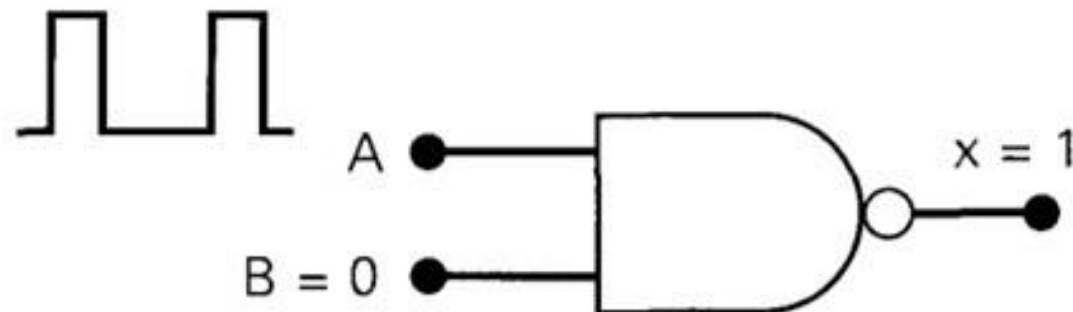
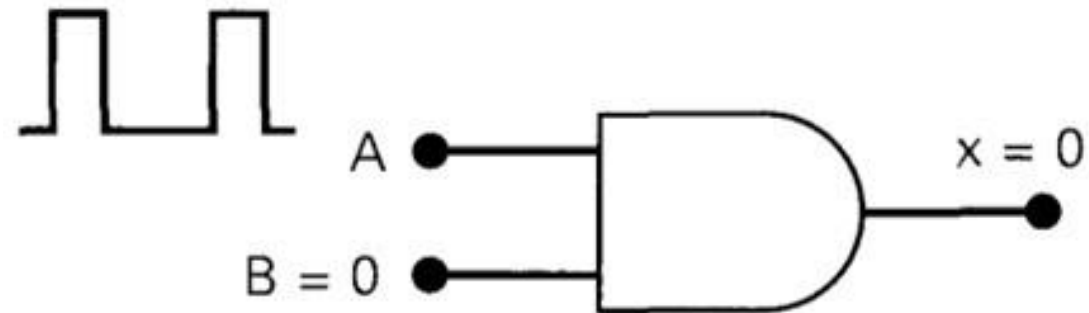
72



# Mạch disable

73

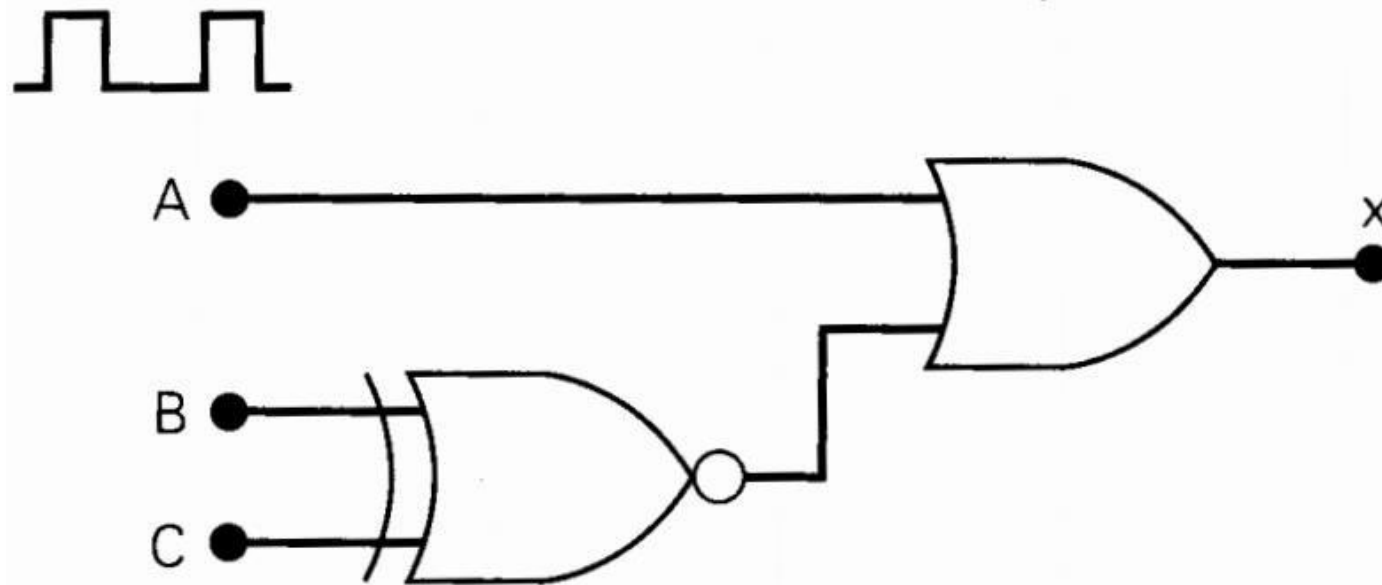
DISABLE



# Ví dụ

74

- Thiết kế mạch tổ hợp cho phép 1 tín hiệu truyền đến ngõ xuất khi một trong 2 tín hiệu điều khiển ở **mức 1** (không đồng thời). Các trường hợp khác ngõ xuất ở **mức 1** (HIGH).



3.5 Cho hàm bool:

$f(A,B,C,D) = \sum(3,4,5,7,10,12,13) + d(8,9,11)$ , Dùng bản đồ Karnaugh để:

- Xác định dạng chuẩn tổng các tích của hàm f (gọi là hàm g)
- Xác định dạng chuẩn tích các tổng của hàm f (gọi là hàm h)
- So sánh hai hàm g và h
- Vẽ sơ đồ mạch hàm g mà sử dụng cổng NOR.

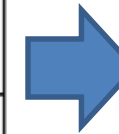
**Bước 1. Lập bảng các nô**

**Bước 2. Tối giản các nô**

**Bước 3. Kết quả**

$F(A,B,C,D) =$

AB \ CD	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10



AB \ CD	00	01	11	10
00				
01				
11				
10				