NHẬP MÔN MẠCH SỐ

Chương 4

Bìa Karnaugh

Tổng quan

Chương này sẽ học về:

- Phương pháp đánh giá ngõ ra của một mạch logic cho trước.
- Phương pháp thiết kế một mạch logic từ biểu thức đại số cho trước.
- Phương pháp thiết kế một mạch logic từ yêu cầu cho trước.
- Các phương pháp để đơn giản/tối ưu một mạch logic → giúp cho mạch thiết kế được tối ưu về diện tích, chi phí và tốc độ.

Nội dung

1. Mạch logic số
2. Thiết kế một mạch số
3. Bìa Karnaugh (bản đồ Karnaugh)
4. Cổng XOR/XNOR

1. Mạch logic số (logic circuit)

- Dùng định lý Boolean để đơn giản hàm sau:

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

Tên	Dạng AND	Dạng OR
Định luật thống nhất	1A = A	0 + A = A
Định luật không	OA = O	1+ A = 1
Định luật Idempotent	AA = A	A + A = A
Định luật nghịch đảo	$A\overline{A} = 0$	$A + \overline{A} = 1$
Định luật giao hoán	AB = BA	A + B = B + A
Định luật kết hợp	(AB)C = A(BC)	(A+B)+C = A + (B+C)
Định luật phân bố	A + BC = (A + B)(A + C)	A(B+C) = AB + AC
Định luật hấp thụ	A(A + B) = A	A + AB = A
Định luật De Morgan	$\overline{AB} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A}.\overline{B}$

Tích chuẩn và Tổng chuẩn

- **<u>Tích chuẩn</u>** <u>(minterm):</u> **m_i** là các *số hạng tích* (AND) mà tất cả các biến xuất hiện ở dạng bình thường (nếu là 1) hoặc dạng bù (complement) (nếu là 0)
- **<u>Tổng chuẩn</u>** <u>(Maxterm):</u> **M_i** là các *số hạng tổng* (OR) mà tất cả các biến xuất hiện ở dạng bình thường (nếu là 0) hoặc dạng bù (complement) (nếu là 1)

x	y	z	Minterms	Maxterms
0	0	0	$m_0 = \bar{x}\,\bar{y}\,\bar{z}$	$M_0 = x + y + z$
0	0	1	$m_1 = \bar{x}\,\bar{y}\,z$	$M_1 = x + y + \bar{z}$
0	1	0	$m_2 = \bar{x}\,y\,\bar{z}$	$M_2 = x + \bar{y} + z$
0	1	1	$m_3 = \bar{x}\,y\,z$	$M_3 = x + \bar{y} + \bar{z}$
1	0	0	$m_4 = x\,\bar{y}\,\bar{z}$	$M_4 = \bar{x} + y + z$
1	0	1	$m_5 = x\,\bar{y}\,z$	$M_5 = \bar{x} + y + \bar{z}$
1	1	0	$m_6 = x\,y\,\bar{z}$	$M_6 = \bar{x} + \bar{y} + z$
1	1	1	$m_7 = x\,y\,z$	$M_7 = \bar{x} + \bar{y} + \bar{z}$

Dạng chính tắc (Canonical Form)

- **<u>Dạng chính tắc 1:</u>** là dạng *tổng của các tích chuẩn_1 (minterm_1)* (*tích chuẩn_1* là tích chuẩn mà tại tổ hợp đó hàm Boolean có giá trị 1).

x	y	z	Minterms	Maxterms	F
0	0	0	$m_0 = \bar{x}\,\bar{y}\,\bar{z}$	$M_0 = x + y + z$	0
0	0	1	$m_1 = \bar{x}\,\bar{y}\,z$	$M_1 = x + y + \bar{z}$	1
0	1	0	$m_2 = \bar{x}\,y\,\bar{z}$	$M_2 = x + \bar{y} + z$	0
0	1	1	$m_3 = \bar{x}\,y\,z$	$M_3 = x + \bar{y} + \bar{z}$	1
1	0	0	$m_4 = x\,\bar{y}\,\bar{z}$	$M_4 = \bar{x} + y + z$	1
1	0	1	$m_5 = x\,\bar{y}\,z$	$M_5 = \bar{x} + y + \bar{z}$	0
1	1	0	$m_6 = x\,y\,\bar{z}$	$M_6 = \bar{x} + \bar{y} + z$	0
1	1	1	$m_7 = x\,y\,z$	$M_7 = \bar{x} + \bar{y} + \bar{z}$	0

$F(x, y, z) =$

Dạng chính tắc (Canonical Form) (tt)

- **Dạng chính tắc 2:** là dạng *tích của các tổng chuẩn_0* *(Maxterm_0)* (*tổng chuẩn_0* là tổng chuẩn mà tại tổ hợp đó hàm Boolean có giá trị 0).

$$F(x, y, z) =$$

x	y	z	Minterms	Maxterms	F
0	0	0	$m_0 = \bar{x}\,\bar{y}\,\bar{z}$	$M_0 = x + y + z$	$\boxed{0}$
0	0	1	$m_1 = \bar{x}\,\bar{y}\,z$	$M_1 = x + y + \bar{z}$	1
0	1	0	$m_2 = \bar{x}\,y\,\bar{z}$	$M_2 = x + \bar{y} + z$	$\boxed{0}$
0	1	1	$m_3 = \bar{x}\,y\,z$	$M_3 = x + \bar{y} + \bar{z}$	1
1	0	0	$m_4 = x\,\bar{y}\,\bar{z}$	$M_4 = \bar{x} + y + z$	1
1	0	1	$m_5 = x\,\bar{y}\,z$	$M_5 = \bar{x} + y + \bar{z}$	$\boxed{0}$
1	1	0	$m_6 = x\,y\,\bar{z}$	$M_6 = \bar{x} + \bar{y} + z$	$\boxed{0}$
1	1	1	$m_7 = x\,y\,z$	$M_7 = \bar{x} + \bar{y} + \bar{z}$	$\boxed{0}$

$$Mi = \overline{mi} \text{ and } mi = \overline{Mi}$$

- **Trường hợp tùy định (don't care)**

Hàm Boolean theo dạng chính tắc:

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

Ví dụ

- Câu hỏi: Trong các biểu thức sau, biểu thức nào ở dạng chính tắc?
  a. X'YZ + X'Y'
  b. X'YZ + XY'Z + XYZ'
  c. X + YZ
  d. X + Y + Z
  e. (X+Y)(Y+Z)

- Trả lời:

Dạng chính tắc (Canonical Forms) (tt)

Tổng các tích chuẩn Sum of Minterms	Tích các tổng chuẩn Product of Maxterms
Σ	Π
Chỉ quan tâm hàng có giá trị 1	Chỉ quan tâm hàng có giá trị 0
X = 0: viết X’	X = 0: viết X
X = 1: viết X	X = 1: viết X’

Dạng chuẩn (Standard Form)

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

Ví dụ

- Câu hỏi: Trong các biểu thức sau, biểu thức nào ở dạng chuẩn?
  a. XYZ + X’Y’
  b. X’YZ + XY’Z + XYZ’
  c. X + YZ
  d. X + Y + Z
  e. (X+Y)(Y+Z)

- Trả lời:

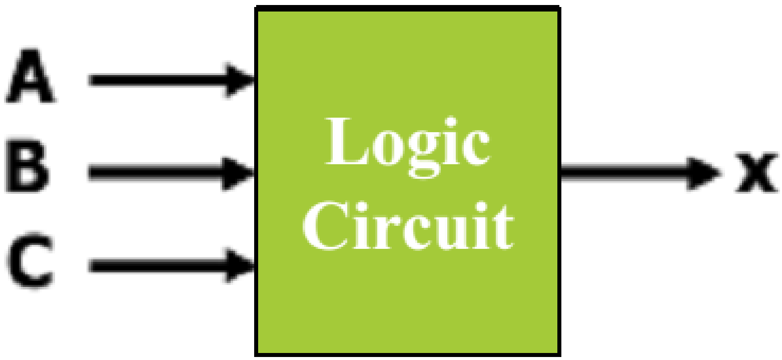
2. Thiết kế một mạch logic

Ví dụ

- Thiết kế một mạch logic số với
  - 3 ngõ vào
  - 1 ngõ ra
  - Kết quả ngõ ra bằng 1 khi có từ 2 ngõ vào trở lên có giá trị bằng 1

Các bước thiết kế một mạch logic số

- **Bước 1**: xây dựng bảng sự thật/chân trị



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

Các bước thiết kế một mạch logic số

- 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	$\overline{A}.B.C$
1	0	0	0	
1	0	1	1	$A.\overline{B}.C$
1	1	0	1	$A.B.\overline{C}$
1	1	1	1	$A.B.C$

Biểu thức SOP cho ngõ ra X:

$$x = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

Các nhóm **AND** cho mỗi trường hợp ngõ ra là 1

Các bước thiết kế một mạch logic số

- **Bước 3:** đơn giản biểu thức logic qua biến đổi đại số

$$x = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

Hạn chế của biến đổi đại số

- Hai vấn đề của biến đổi đại số
  1. Không có hệ thống
  2. Rất khó để kiểm tra rằng giải pháp tìm ra đã là tối ưu hay chưa?

- **Bìa Karnaugh** sẽ khắc phục những nhược điểm này
  – Tuy nhiên, bìa Karnaugh chỉ để giải quyết các hàm Boolean có không quá 5 biến

Các bước thiết kế một mạch logic số

- **Bước 4:** vẽ sơ đồ mạch logic cho

$$x = BC + AC + AB$$

3. Bìa Karnaugh

Chi phí để tạo ra một mạch logic

- Chi phí (cost) để tạo ra một mạch logic liên quan đến:
  - Số cổng (gates) được sử dụng
  - Số đầu vào của mỗi cổng

- Một **literal** là một biến kiểu Boolean hay bù của nó

Chi phí để tạo ra một mạch logic

- Chi phí của một biểu thức Boolean **B** được biểu diễn dưới dạng tổng của các tích (Sum-of-Product) như sau:

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

Trong đó k là số các term (thành phần tích) trong biểu thức B

O(B) : số các term trong biểu thức B

P_J(B): số các literal trong term thứ j của biểu thức B

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

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

Chi phí để tạo ra một mạch logic Ví dụ

- Tính chi phí của các biểu thức sau:

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

$$O(B) = \begin{cases} m & \text{if } B \text{ has } m \text{ terms} \\ 0 & \text{if } B \text{ has } one \text{ term} \end{cases}$$

$$P_j(B) = \begin{cases} m & \text{if the } j^{\text{th}} \text{ term of } B \text{ has } m \text{ literals} \\ 0 & \text{if the } j^{\text{th}} \text{ term of } B \text{ has } one \text{ literal} \end{cases}$$

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

- M. Karnaugh, “The Map Method for Synthesis of combinatorial Logic Circuits”, Transactions of the American Institute of Electrical Engineers, Communications and Electronics, Vol. 72, pp. 593-599, November 1953.
- Bìa Karnaugh là một công cụ hình học để đơn giản hóa các biểu thức logic
- Tương tự như bảng sự thật, bìa Karnaugh sẽ xác định giá trị ngõ ra cụ thể tại các tổ hợp của các đầu vào tương ứng.

Bìa Karnaugh (bìa K)

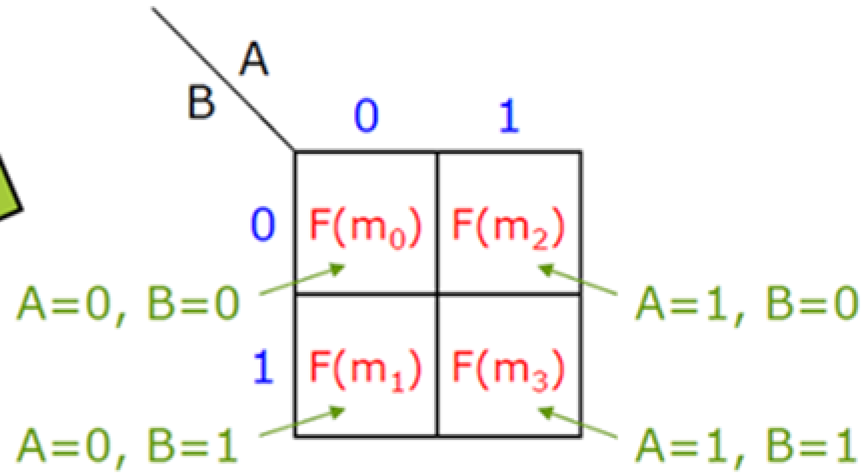
- **Bìa Karnaugh** là *biểu diễn của bảng sự thật* dưới dạng một ma trận các ô (matrix of squares/cells) trong đó mỗi ô tương ứng với một dạng tích chuẩn (minterm) hay dạng tổng chuẩn (Maxterm).

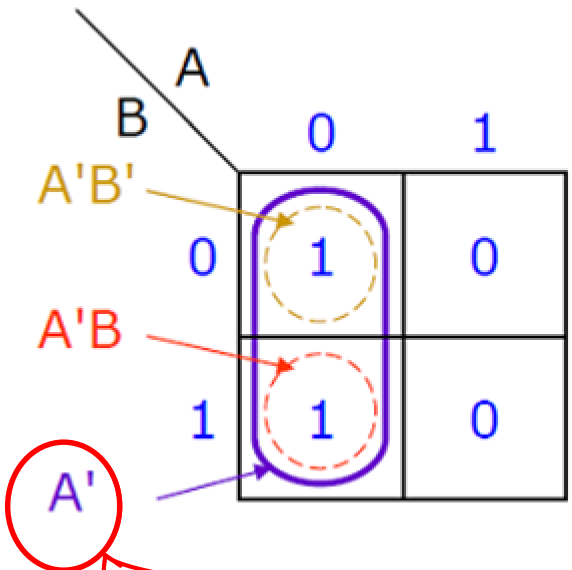
- Với một hàm có **n biến**, chúng ta cần một bảng sự thật có $\mathbf{2^n}$ hàng, tương ứng bìa Karnaugh có $\mathbf{2^n}$ ô (cell).

- Để biểu diễn một hàm logic, *một giá trị ngõ ra* trong bảng sự thật sẽ được copy sang *một ô tương ứng* trong bìa K

Bìa Karnaugh 2 biến

	AB	F
(m_0)	00	1
(m_1)	01	1
(m_2)	10	0
(m_3)	11	0





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

Bìa Karnaugh 3 biến

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

Ví dụ:

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

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

$F = A'BC' + A'BC + AB'C' + ABC'$ (đại số)

$F = A'B + AC' + BC'$ (chưa tối ưu)

$= A'B + AC'$ (tối ưu)

Bìa Karnaugh 3 biến

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

Cách 1

C \ AB	00	01	11	10
0	0	2	6	4
1	1	3	7	5

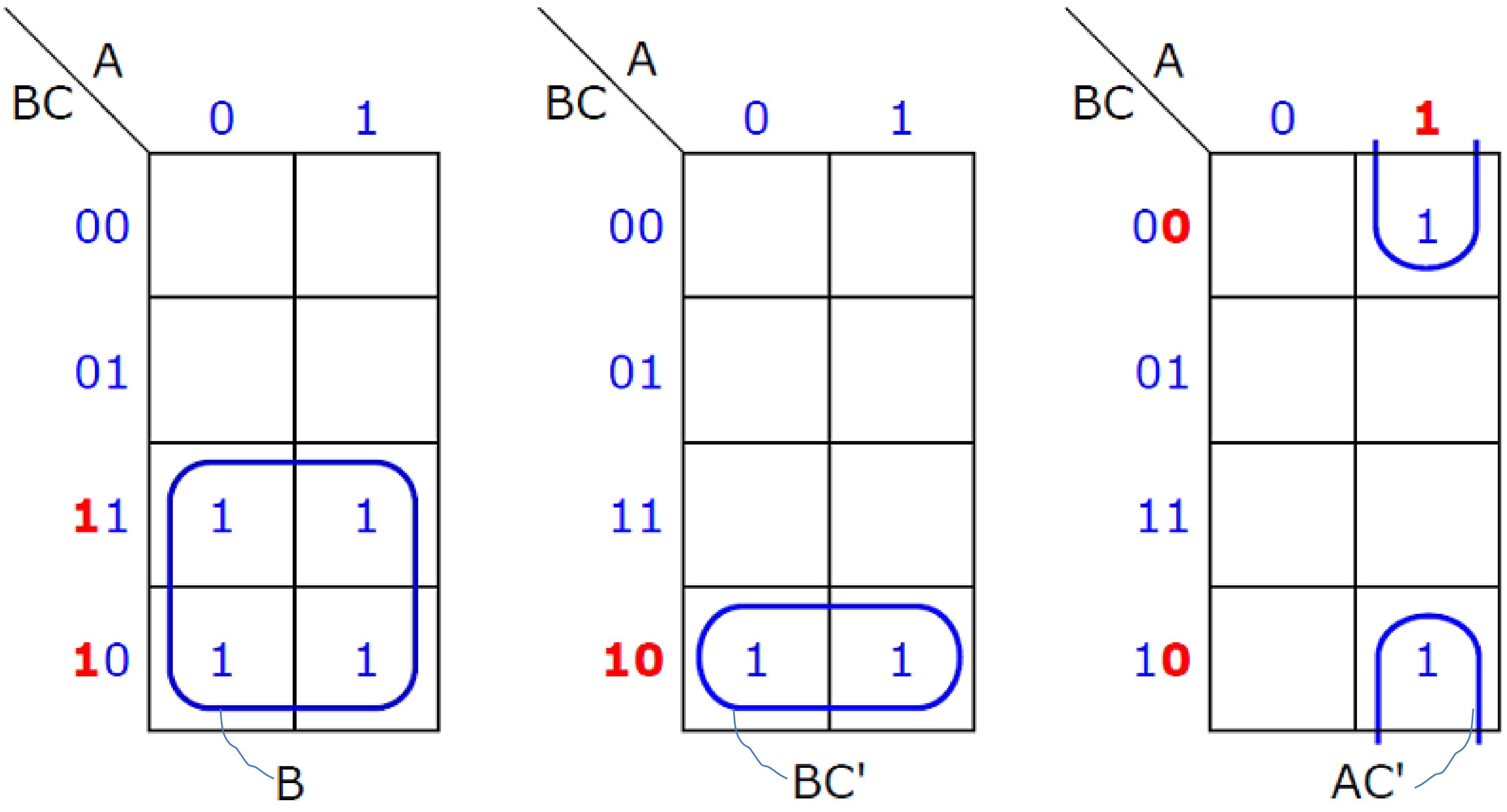
Cách 2

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

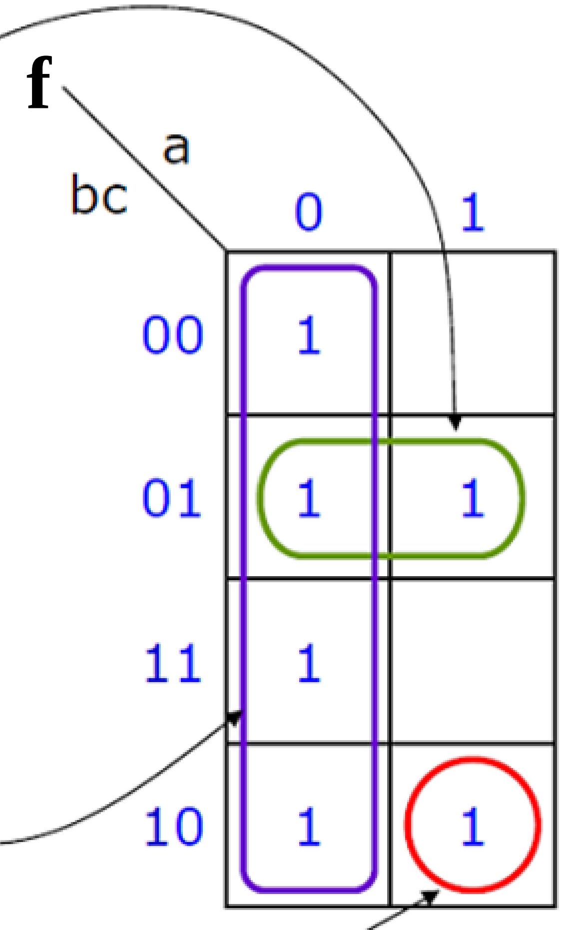
Cách 3

Lưu ý: có thể sử dụng cách nào để biểu diễn bìa-K cũng được, nhưng phải lưu ý *trọng số của các biến* thì mới đảm bảo thứ tự các ô theo giá trị thập phân.

Bìa Karnaugh 3 biến



Bìa Karnaugh 3 biến



$f(a,b,c) = abc' + b'c + a'$

(chưa tối ưu)

Bìa Karnaugh 3 biến

$F = m_1 + m_3 + m_5$

$\quad = M_0 M_2 M_4 M_6 M_7$

F

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

Simplify

$F = a'c + b'c$

Bìa Karnaugh 3 biến

$$G = (m_1+m_3+m_5)'$$
$$= (M_0M_2M_4M_6M_7)'$$

G

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

Simplify

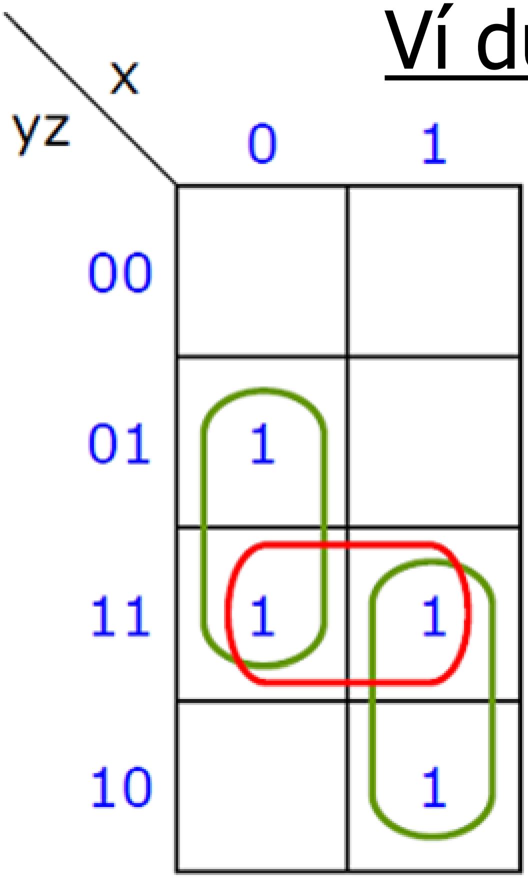
G

bc \ a

G = F’

Bìa Karnaugh 3 biến

Ví dụ:

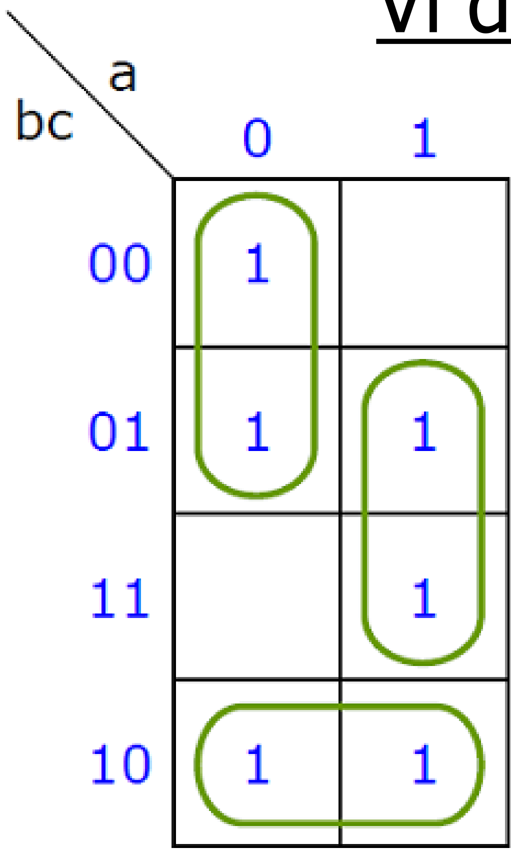


Rút gọn chưa tối ưu

Rút gọn tối ưu

Bìa Karnaugh 3 biến

Ví dụ:



F =

Bìa Karnaugh 4 biến

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 =

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

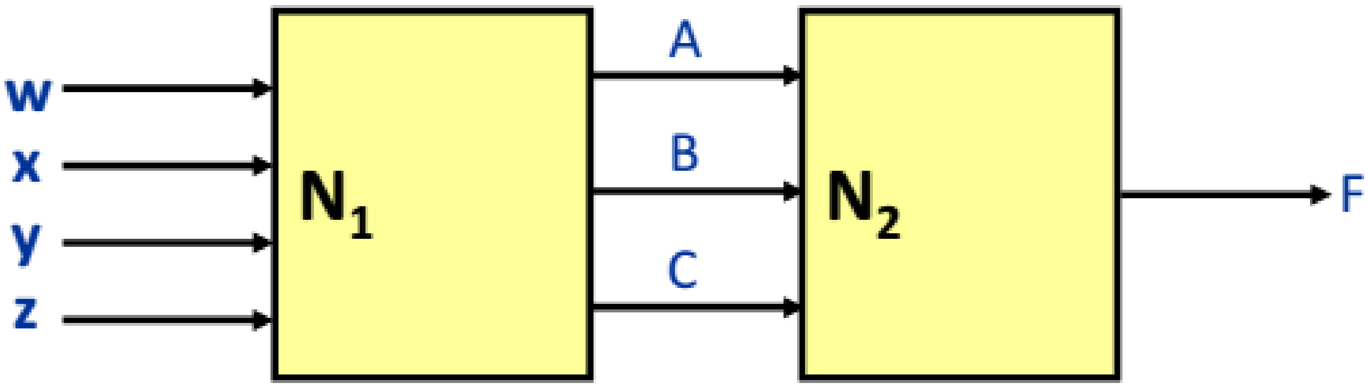
Bìa Karnaugh 4 biến

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

Bìa Karnaugh 4 biến

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

Hàm đặc tả không đầy đủ
(Incompletely Specified Functions)



- Giả thuyết: N1 không bao giờ cho kết quả ABC = 001 và ABC = 110
- Câu hỏi : F cho ra giá trị gì trong trường hợp ABC = 001 và ABC = 110 ?

We don’t care!!!

Hàm đặc tả không đầy đủ (tt)
(Incompletely Specified Functions)

- Trong trường hợp trên thì chúng ta phải làm thế nào để đơn giản N2?

A	B	C	F
0	0	0	1
0	0	1	~~X~~ 0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	~~X~~ 0
1	1	1	1

Giả sử F(0,0,1) = 0 và F(1,1,0)=0, ta có biểu thức sau:

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

Hàm đặc tả không đầy đủ (tt)
(Incompletely Specified Functions)

- Tuy nhiên, nếu giả 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	~~X~~ 1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	~~X~~ 1
1	1	1	1

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

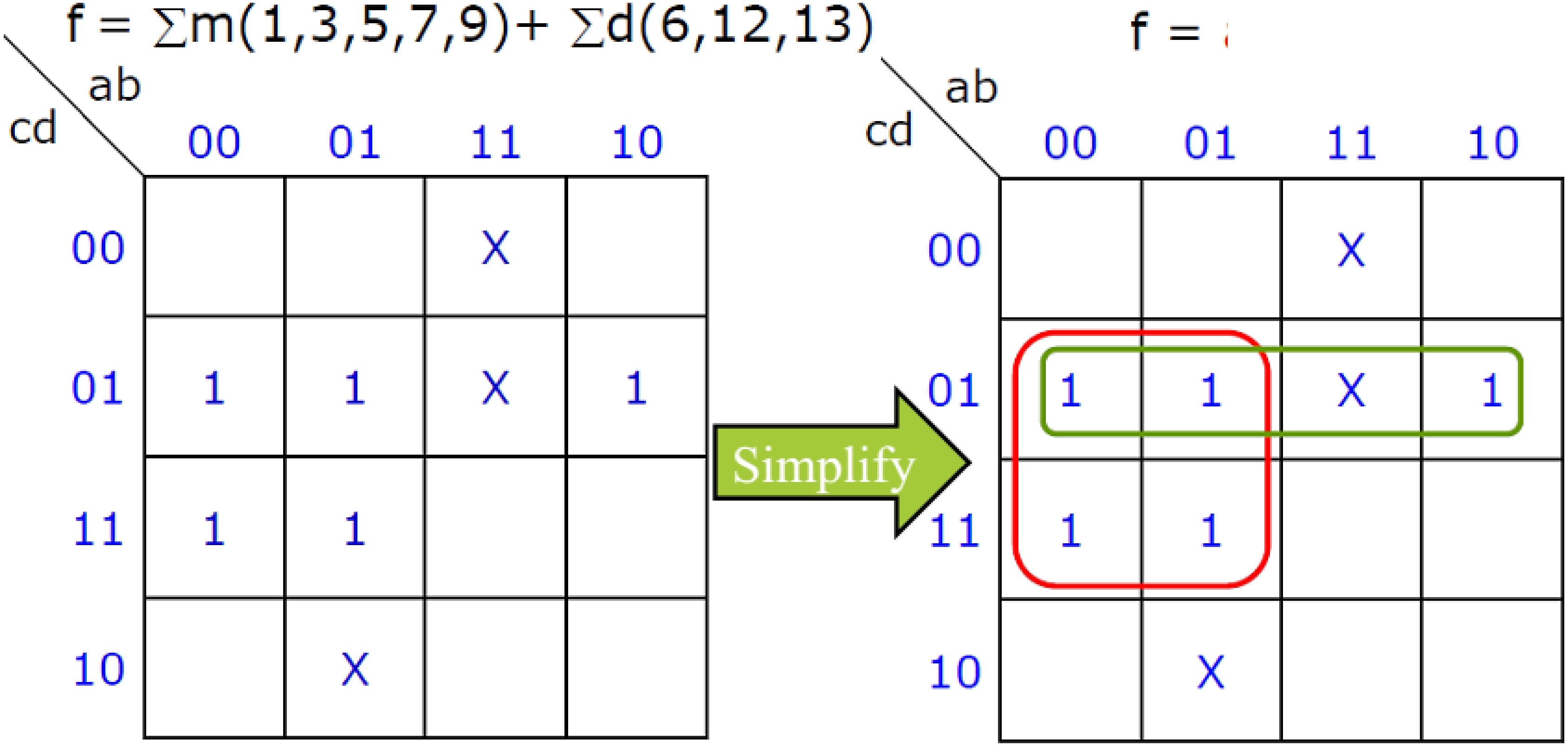
So sánh với giả thuyết trước đó:
F(A,B,C) = A'C' + BC, giải pháp nào chi phí ít hơn (tốt hơn)?

Hàm đặc tả không đầy đủ (tt)
(Incompletely Specified Functions)

Tất cả các ô 1 phải được khoanh tròn, nhưng với ô có giá trị X thì tùy chọn, các ô này chỉ được

- xem xét là **1** nếu đơn giản biểu thức theo dạng **SOP**
- hoặc xem xét là **0** nếu đơn giản biểu thức theo dạng **POS**

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



Đơn giản POS (Product of Sum)

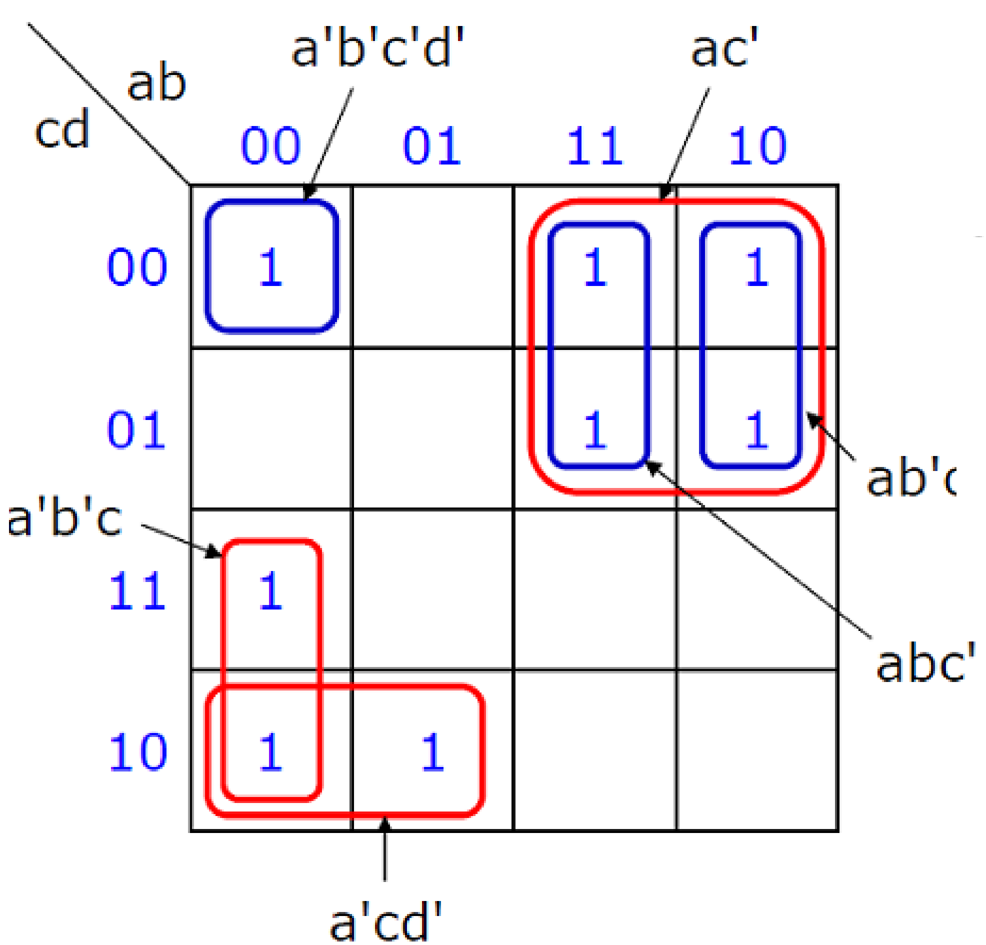
- **Khoanh tròn giá trị 0 thay vì giá trị 1**

Ví dụ: $\mathbf{f = x'z' + wyz + w'y'z' + x'y}$

Implicant cơ bản (Prime Implicant)

- **Implicant**: là dạng *tích chuẩn* của một hàm
  - Một nhóm các ô 1 hoặc một ô 1 đơn lẻ trên một bìa-K kết hợp với nhau tạo ra một dạng tích chuẩn
- **Implicant cơ bản** (prime implicant):
  - Implicant không thể kết hợp với bất kì ô 1 nào khác để loại bỏ một biến
- Tất cả các prime implicant của 1 hàm có thể đạt được bằng cách phát triển các nhóm 1 trong bìa-K lớn nhất có thể

Ví dụ



- a'b'c, a'cd', ac' là các prime implicants
- a'b'c'd', abc', ab'c' là các implicants (nhưng không phải là prime implicants)

Tối thiểu biểu thức sử dụng Essential Prime Implicant (EPI)

- Xác định tất cả các prime implicants
  - Để xác định các prime implicant, các giá trị tùy định (don't care) được coi như là giá trị 1.

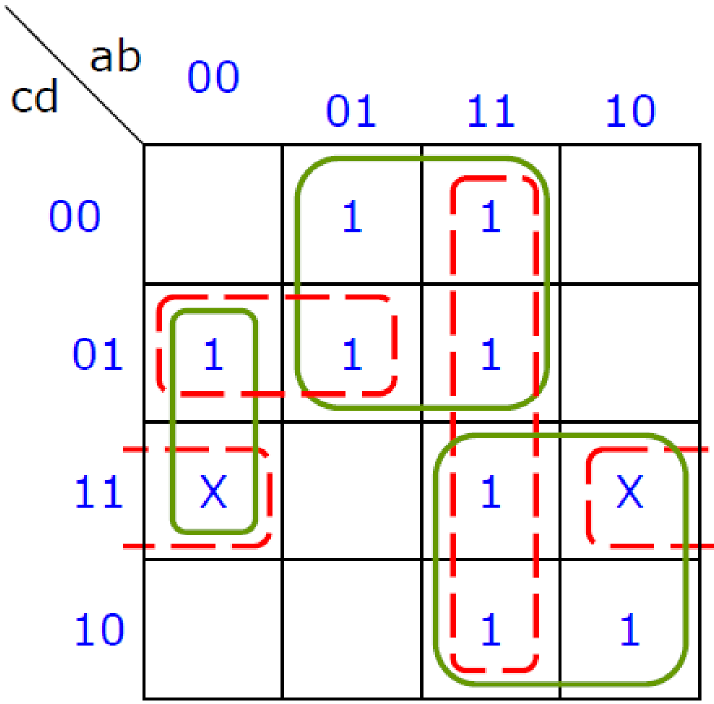
    Tuy nhiên, một prime implicant chỉ gồm các giá trị tùy định thì không cần cho biểu thức ngõ ra.
  - Không phải tất cả các prime implicant đều cần thiết để tạo ra *minimum SOP*

- Ví dụ
  - Tất cả các prime implicants:

    a'b'd, bc', ac, a'c'd, ab,

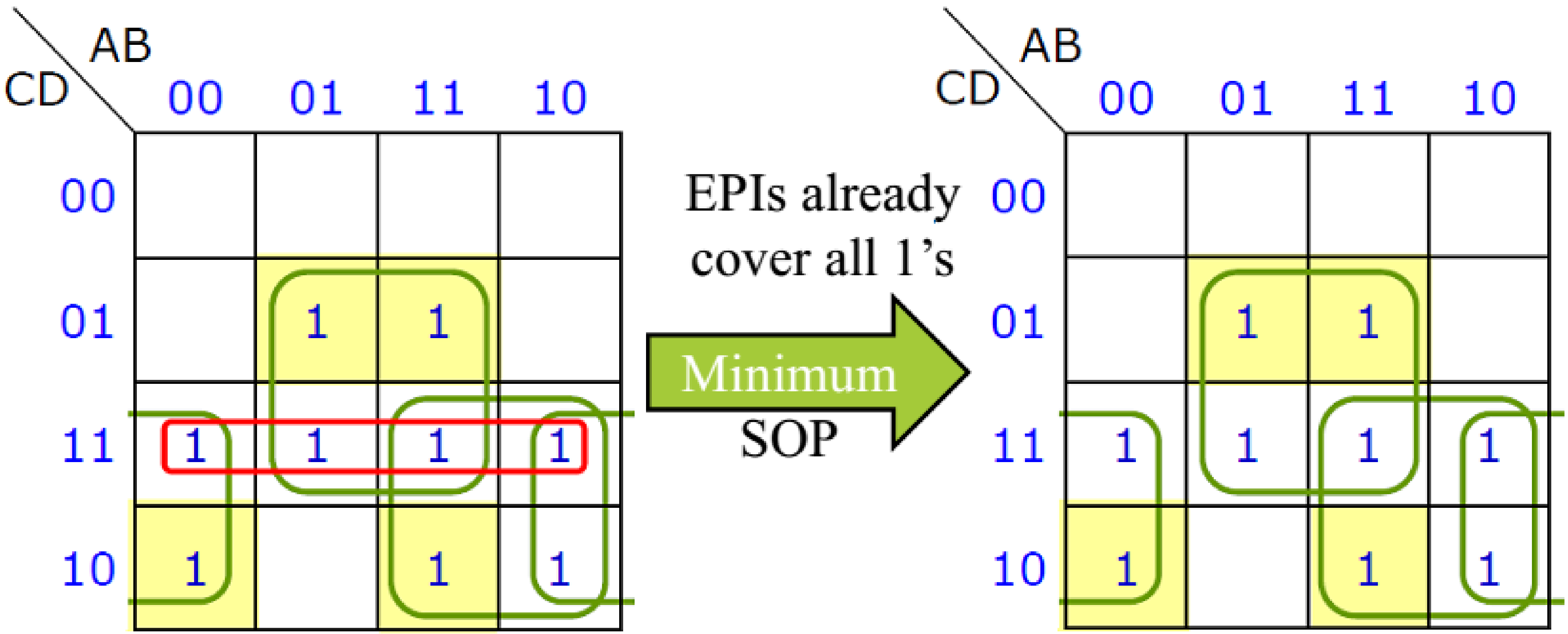
    b'cd (chỉ gồm các giá trị không xác định)
  - Minimum solution:

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



Tối thiểu biểu thức sử dụng Essential Prime Implicant (EPI) (tt)

- **Essential prime implicant** (EPI):
  prime implicant có ít nhất 1 ô không bị gom bởi các prime implicant khác



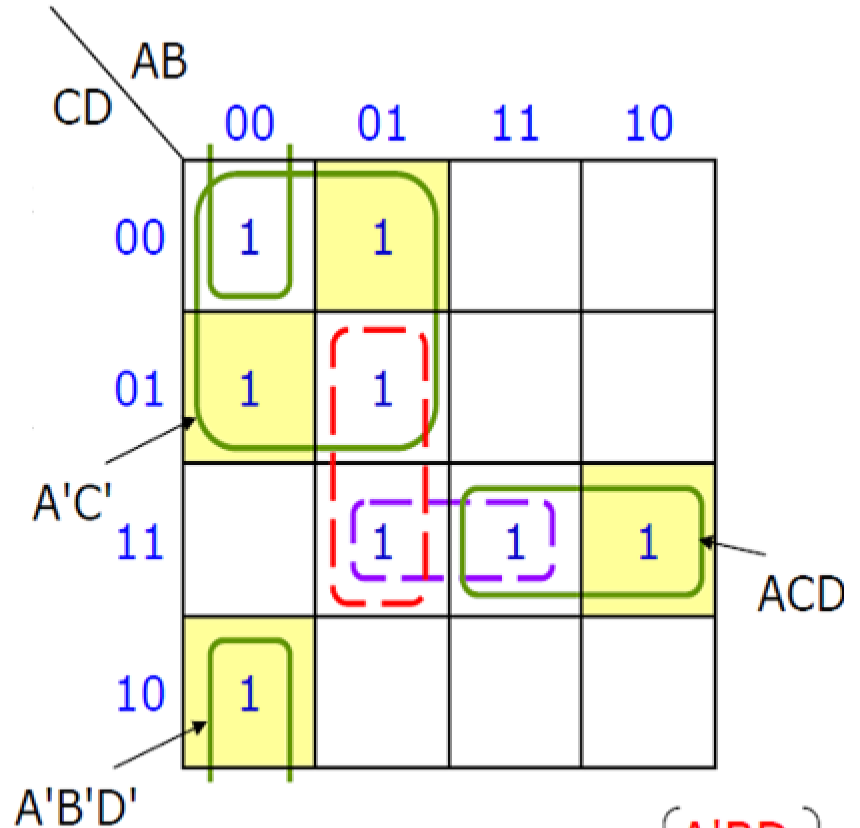
f = CD+BD+B'C+AC

f = BD+B'C+AC

Tối thiểu biểu thức sử dụng Essential Prime Implicant (EPI) (tt)

1. Chọn ra tất cả **EPI**

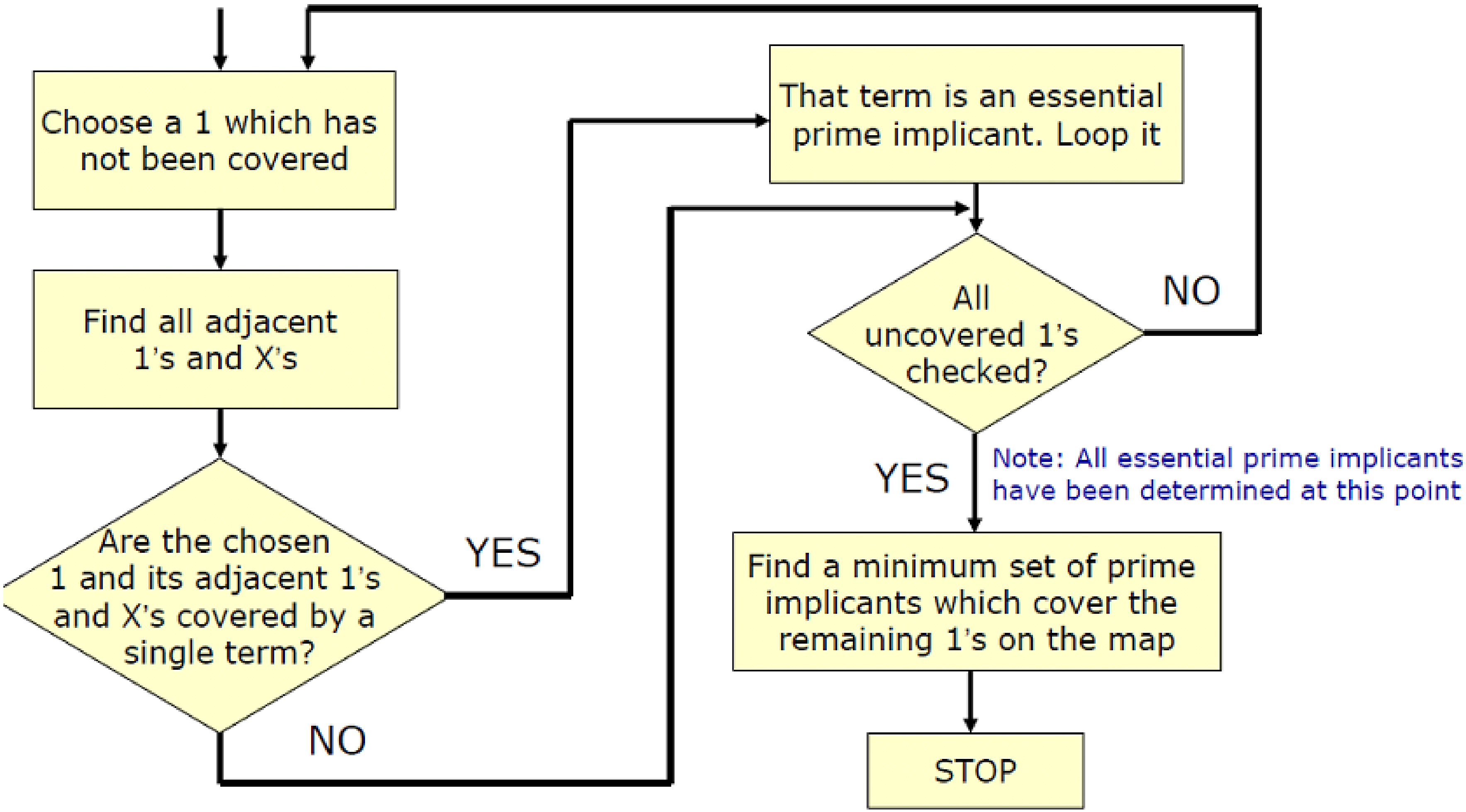
2. Tìm ra một tập nhỏ nhất các prime implicant gom được tất cả các minterm còn lại (các minterm không bị gom bởi các EPI)



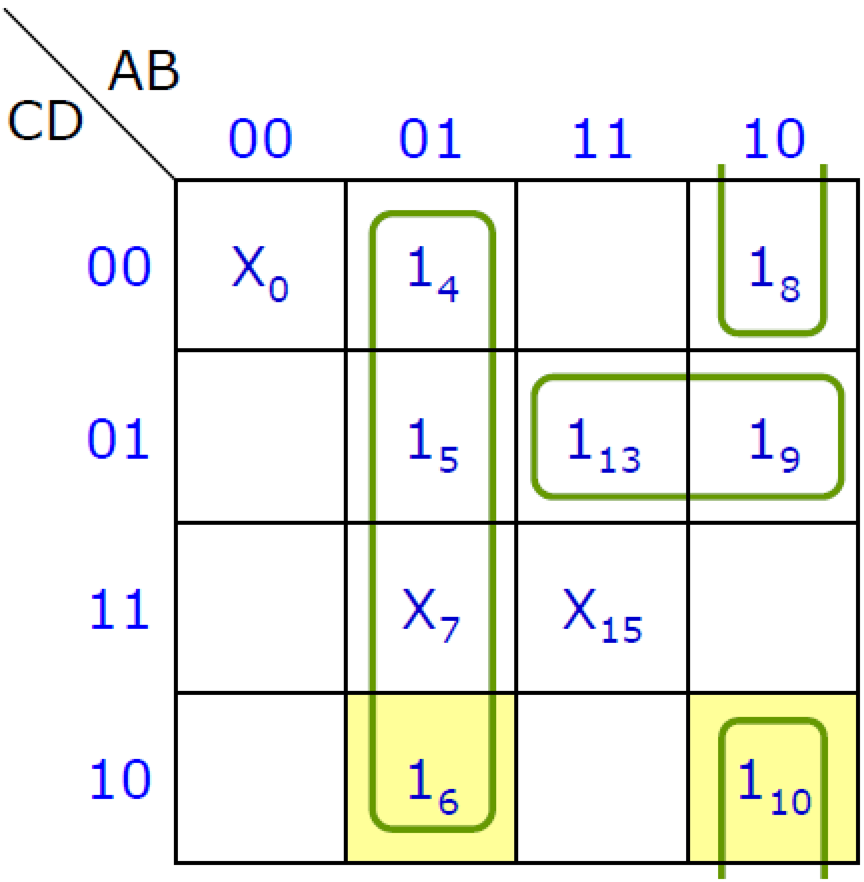
$$F = A'C' + A'B'D' + ACD + \left\{ \begin{matrix} A'BD \\ \text{or} \\ BCD \end{matrix} \right\}$$

Tối thiểu biểu thức sử dụng Essential Prime Implicant (EPI) (tt)

- Lưu đồ để xác định một **minimum SOP** sử dụng K-map



Ví dụ

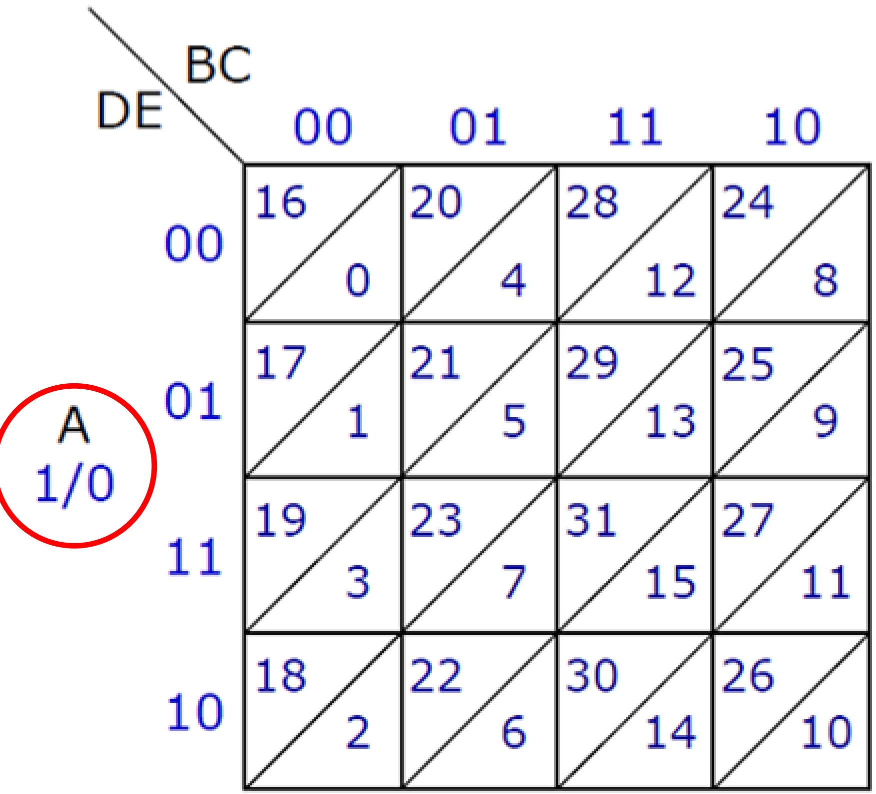


- Step 1: đánh dấu 1_4
- Step 2: đánh dấu 1_5
- Step 3: đánh dấu 1_6
  - EPI => A'B được chọn
- Step 4: đánh dấu 1_8
- Step 5: đánh dấu 1_9
- Step 6: đánh dấu 1_{10}
  - EPI => AB'D' được chọn
- Step 7: đánh dấu 1_{13}

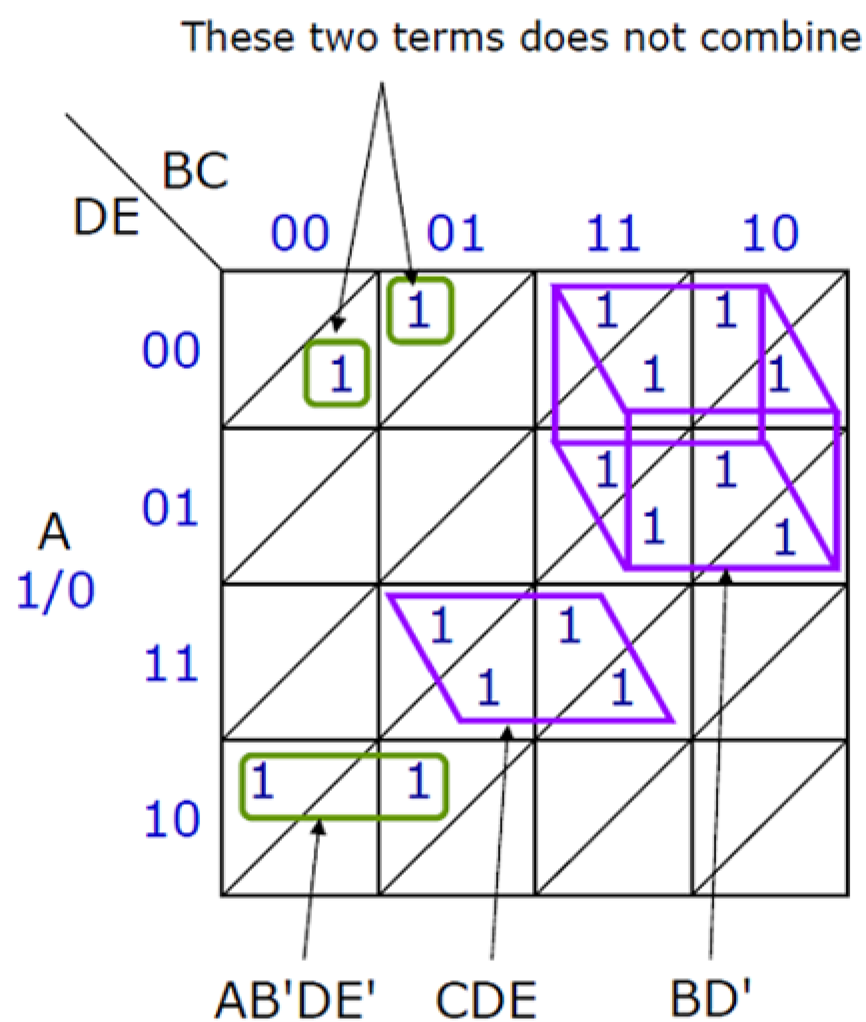
(tại điểm này tất cả EPIs đã được xác định)

- Step 8: AC'D được chọn để gom các số 1 còn lại

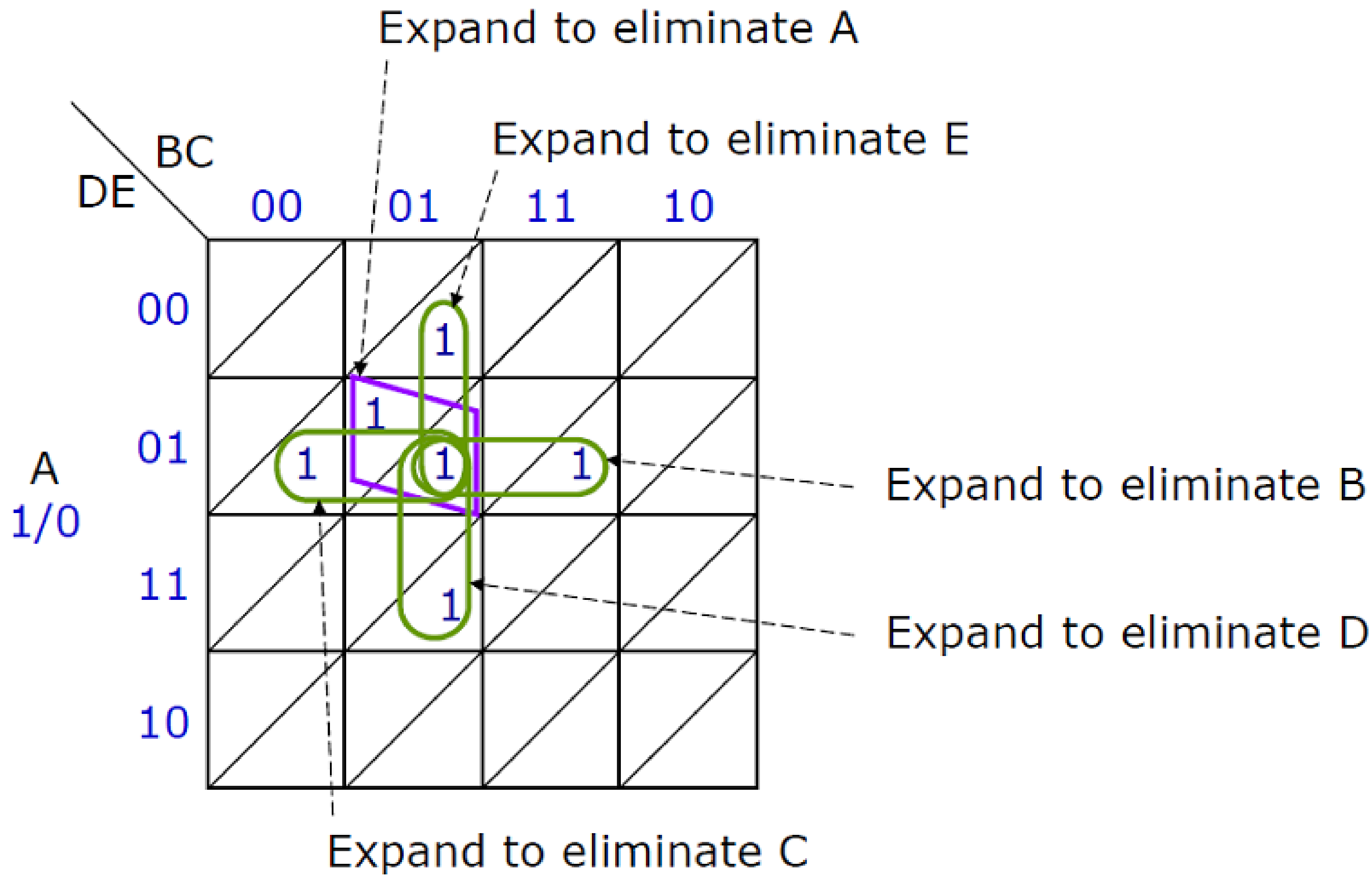
Bìa Karnaugh 5 biến



minterm locations

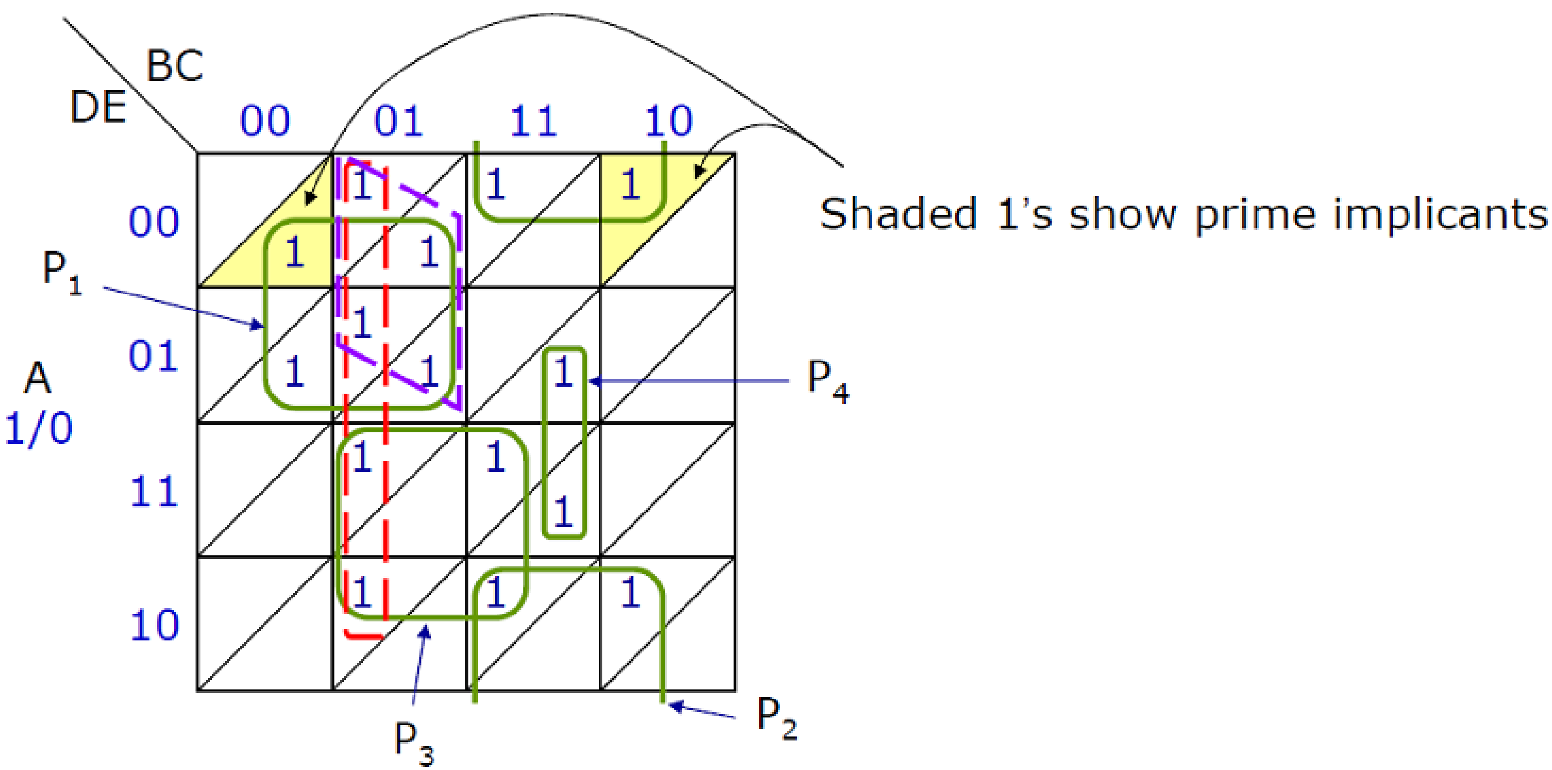


Bìa Karnaugh 5 biến



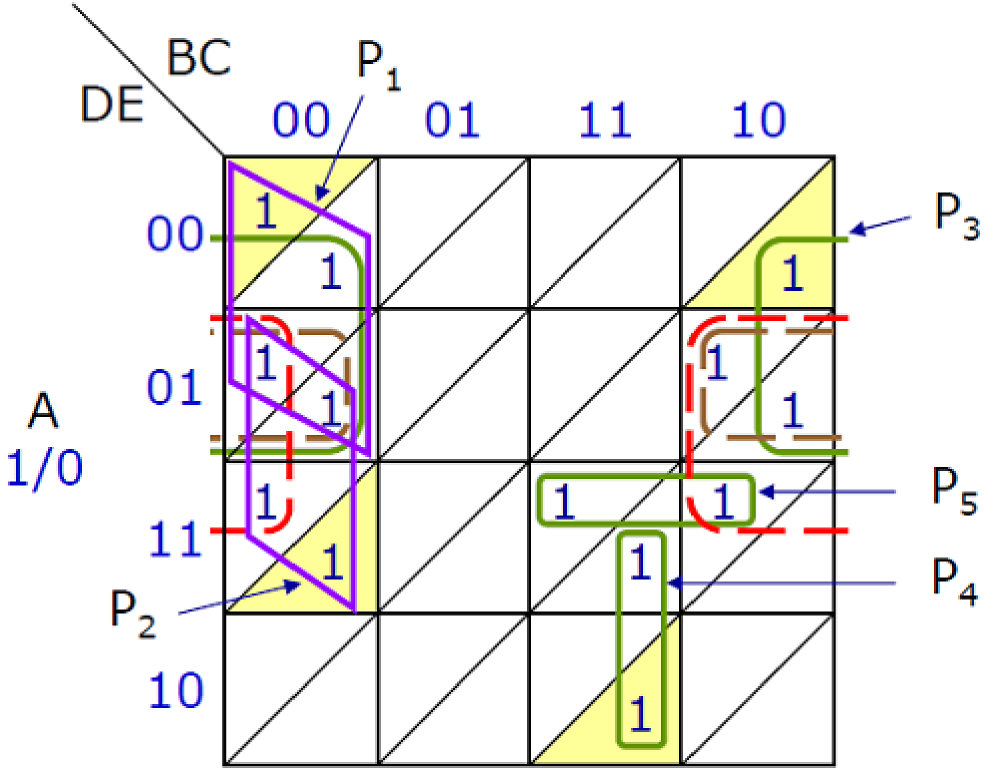
Bìa Karnaugh 5 biến

$$F = \underbrace{A'B'D'}_{P_1} + \underbrace{ABE'}_{P_2} + \underbrace{ACD}_{P_3} + \underbrace{A'BCE}_{P_4} + \left\{ \begin{array}{c} AB'C \\ \text{or} \\ B'CD' \end{array} \right\}$$



Bìa Karnaugh 5 biến

$$F = \underbrace{B'C'D'}_{P_1} + \underbrace{B'C'E}_{P_2} + \underbrace{A'C'D'}_{P_3} + \underbrace{A'BCD}_{P_4} + \underbrace{ABDE}_{P_5} + \left\{ \begin{array}{c} AC'E \\ \text{or} \\ C'D'E \end{array} \right\}$$



Bìa Karnaugh 5 biến

Phương pháp khác

Ví dụ 1

$$F = \sum(31, 30, 29, 27, 25, 22, 21, 20, 17, 16, 15, 13, 11, 9, 6, 4, 1, 0)$$

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

A = 0

DE \ BC	00	01	11	10
00	16	20	28	24
01	17	21	29	25
11	19	23	31	27
10	18	22	30	26

A = 1

Bìa Karnaugh 5 biến

Ví dụ 1 (tt)

$$F = \sum(31,30,29,27,25,22,21,20,17,16,15,13,11,9,6,4,1,0)$$

DE \ BC	00	01	11	10
00	1	1		
01	1		1	1
11			1	1
10		1		

A=0

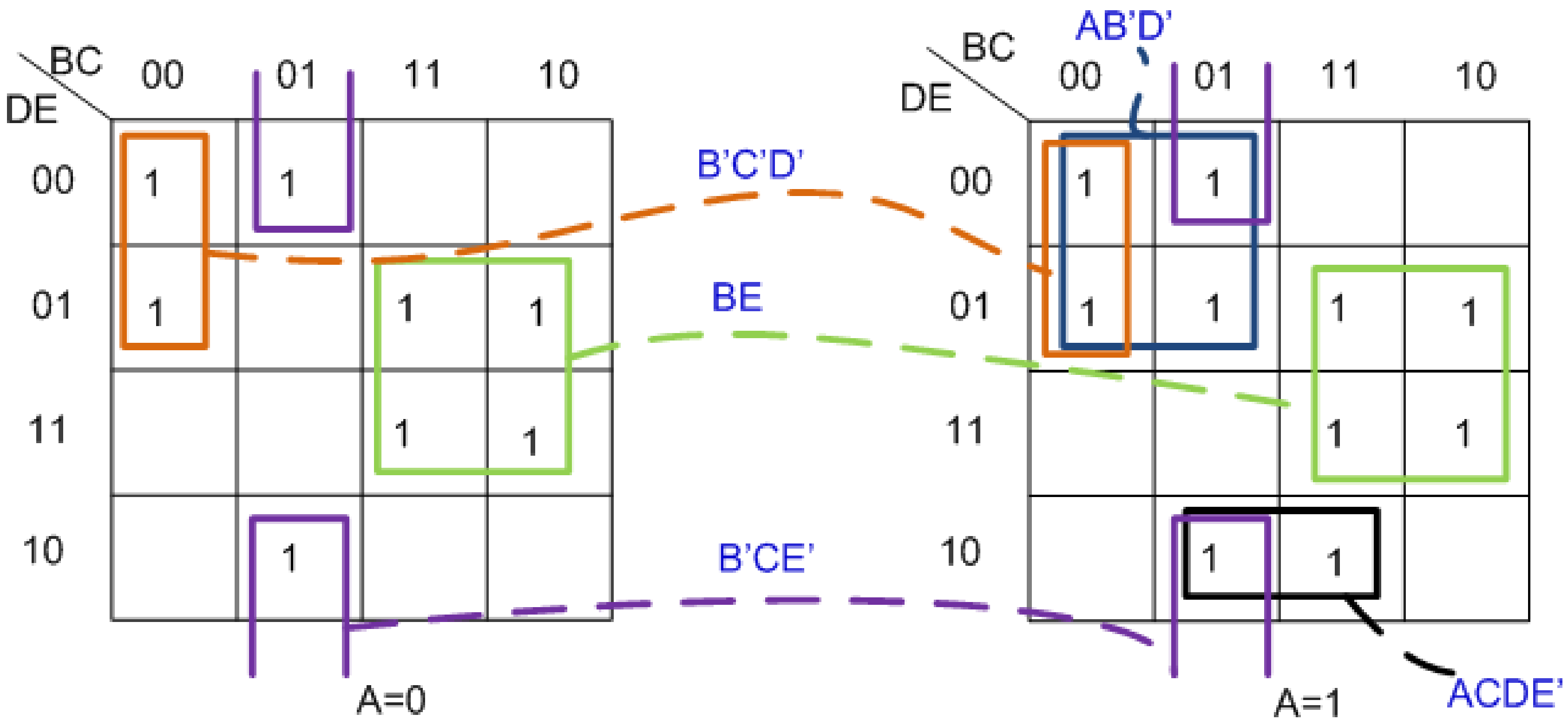
DE \ BC	00	01	11	10
00	1	1		
01	1	1	1	1
11			1	1
10		1	1	

A=1

Bìa Karnaugh 5 biến

Ví dụ 1 (tt)

$$F = \sum(31,30,29,27,25,22,21,20,17,16,15,13,11,9,6,4,1,0)$$

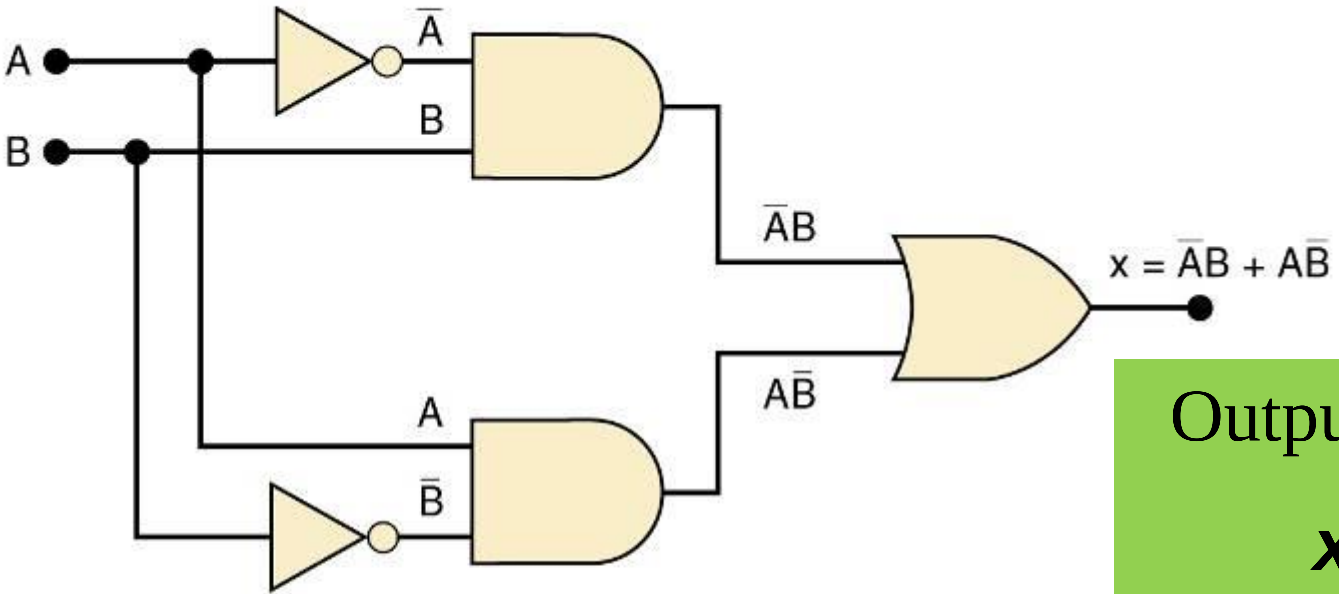


F = ACDE' + B'CE' + BE + B'C'D' + AB'D'

4. Cổng XOR và XNOR

Mạch Exclusive OR (XOR)

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

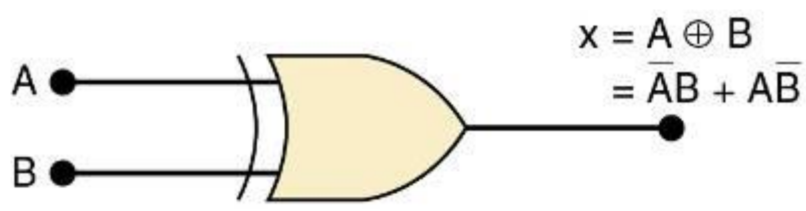


A	B	x
0	0	0
0	1	1
1	0	1
1	1	0

Output expression:

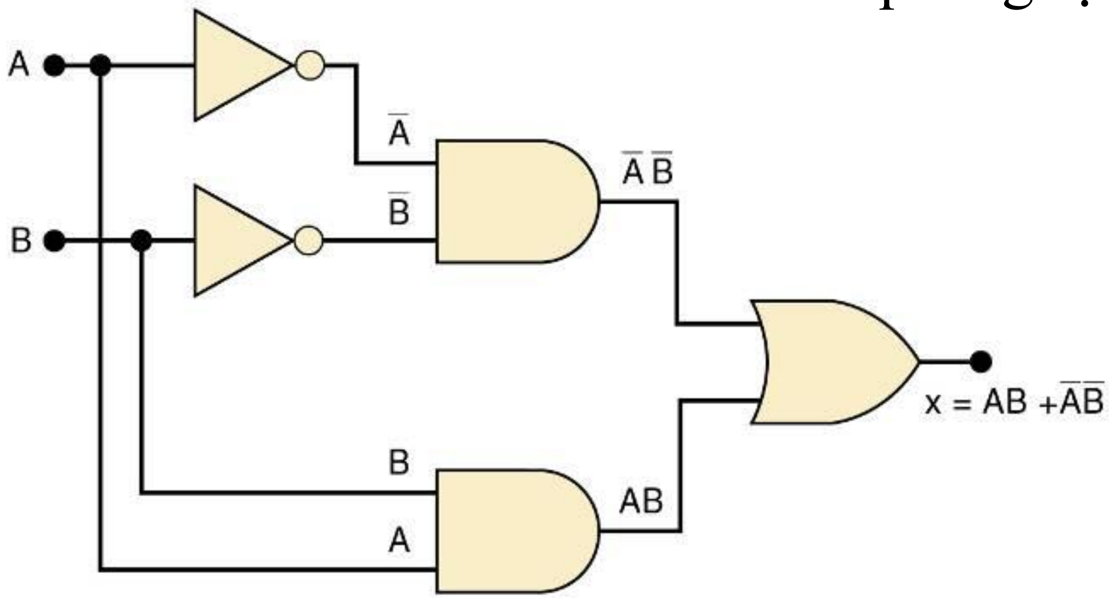
$$x = \bar{A}B + A\bar{B}$$

XOR Gate Symbol



Mạch Exclusive NOR (XNOR)

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

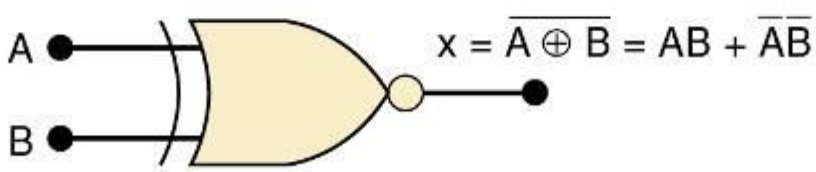


A	B	x
0	0	1
0	1	0
1	0	0
1	1	1

Output expression

$$x = AB + \overline{A}\,\overline{B}$$

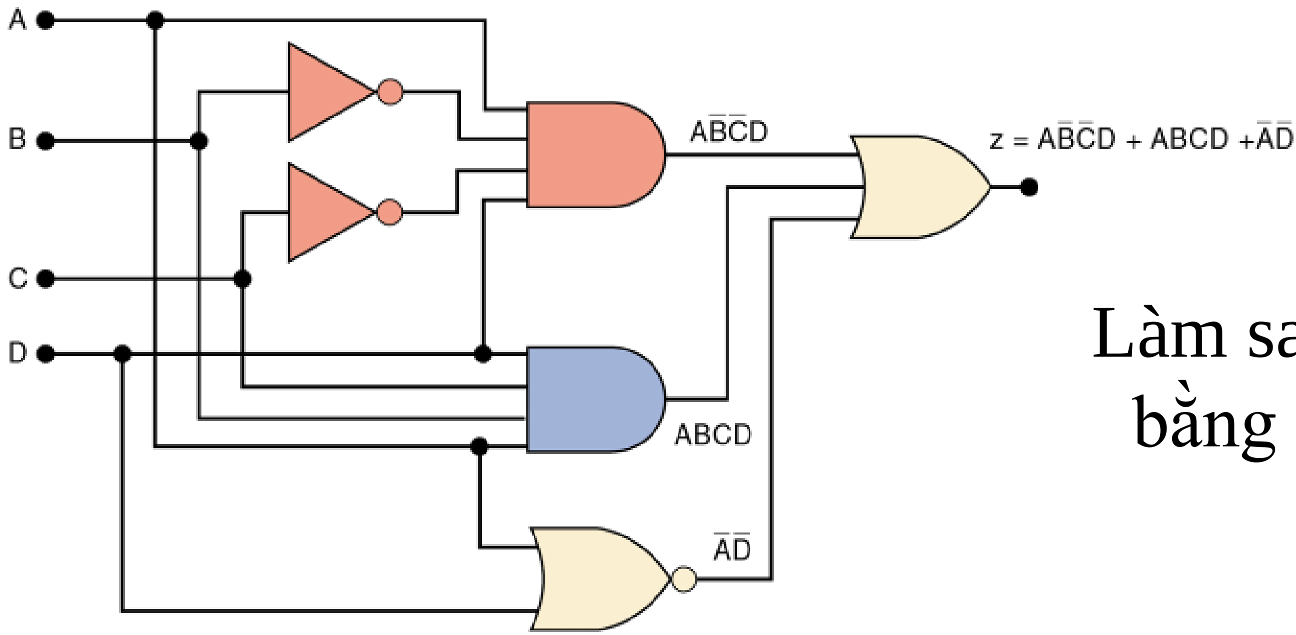
XNOR Gate Symbol



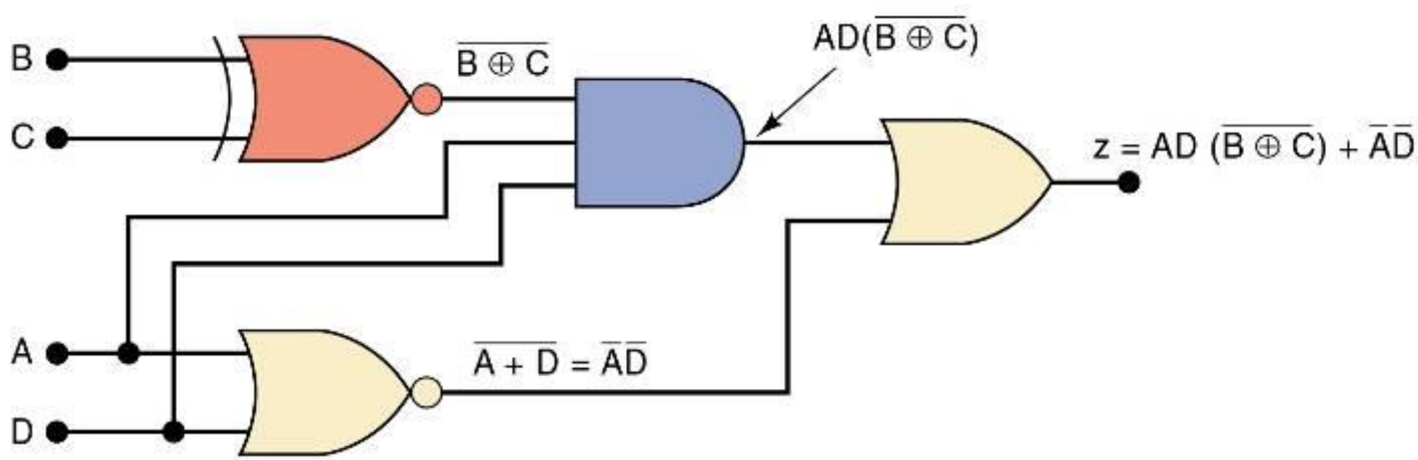
Ví dụ

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

TỐI ƯU MẠCH BẰNG CỔNG XOR VÀ XNOR

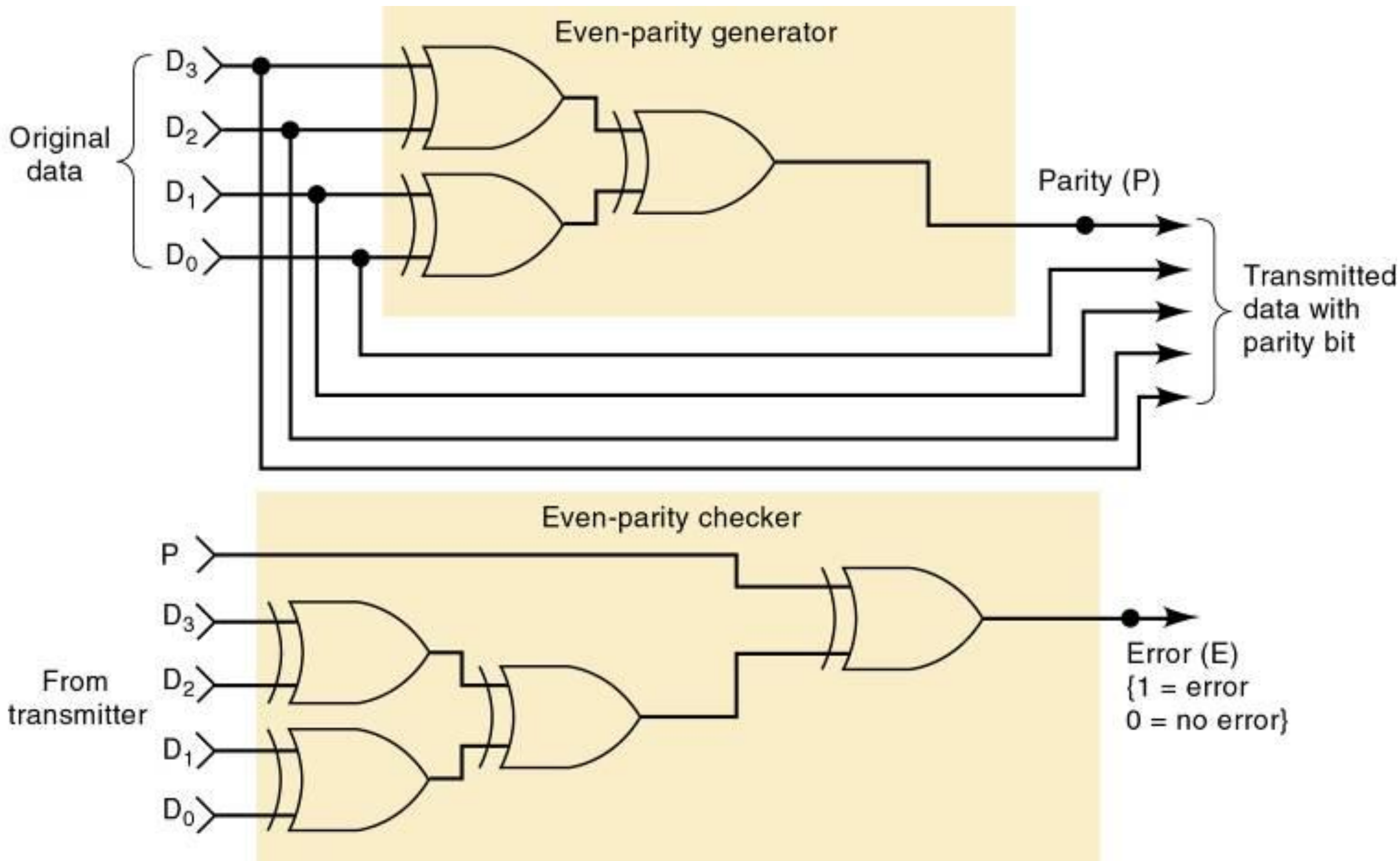


Làm sao tối ưu mạch bằng cổng **XNOR**



Bộ tạo và kiểm tra Parity (Parity generator and checker)

- Cổng XOR và XNOR rất hữu dụng trong các mạch với mục đích **tạo** (bộ phát) và **kiểm tra** (bộ nhận) parity bit



Any question?