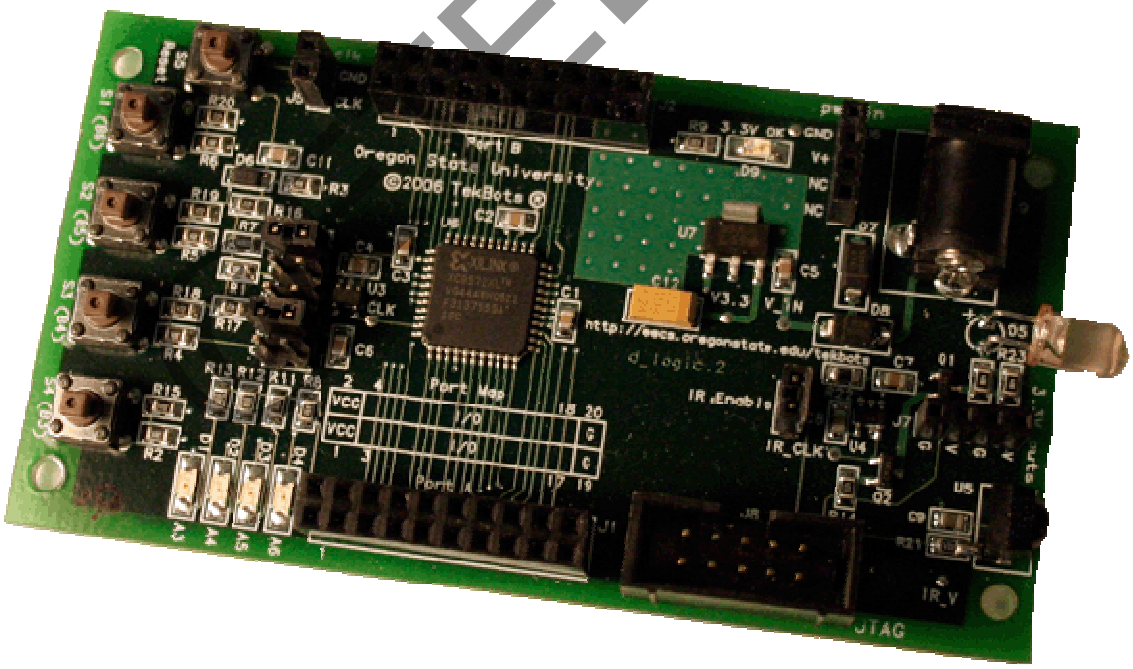
ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN TỬ - VIỄN THÔNG

----- oOo -----

BÀI GIẢNG

Kỹ Thuật Số

(Lưu hành nội bộ)



Đà Nẵng, 2013

Chương 1

HỆ THỐNG SỐ ĐẾM VÀ KHÁI NIỆM VỀ MÃ

1.1. HỆ THỐNG SỐ ĐẾM

1.1.1. Hệ đếm

1. Khái niệm

Hệ đếm là tập hợp các phương pháp gọi và biểu diễn các con số bằng các kí hiệu có giá trị số lượng xác định gọi là các chữ số.

2. Phân loại

Có thể chia các hệ đếm làm hai loại: hệ đếm theo vị trí và hệ đếm không theo vị trí.

a. Hệ đếm theo vị trí:

Hệ đếm theo vị trí là hệ đếm mà trong đó giá trị số lượng của chữ số còn phụ thuộc vào vị trí của nó đứng trong con số cụ thể.

Ví dụ: Hệ thập phân là một hệ đếm theo vị trí. Số **1991** trong hệ thập phân được biểu diễn bằng 2 chữ số "1" và "9", nhưng do vị trí đứng của các chữ số này trong con số là khác nhau nên sẽ mang các giá trị số lượng khác nhau, chẳng hạn chữ số "1" ở vị trí hàng đơn vị biểu diễn cho giá trị số lượng là 1 song chữ số "1" ở vị trí hàng nghìn lại biểu diễn cho giá trị số lượng là 1000, hay chữ số "9" khi ở hàng chục biểu diễn giá trị là 90 còn khi ở hàng trăm lại biểu diễn cho giá trị là 900.

b. Hệ đếm không theo vị trí:

Hệ đếm không theo vị trí là hệ đếm mà trong đó giá trị số lượng của chữ số không phụ thuộc vào vị trí của nó đứng trong con số.

Hệ đếm La Mã là một hệ đếm không theo vị trí. Hệ đếm này sử dụng các ký tự "I", "V", "X"... để biểu diễn các con số, trong đó "I" biểu diễn cho giá trị số lượng 1, "V" biểu diễn cho giá trị số lượng 5, "X" biểu diễn cho giá trị số lượng 10... mà không phụ thuộc vào vị trí các chữ số này đứng trong con số cụ thể.

Các hệ đếm không theo vị trí sẽ không được đề cập đến trong giáo trình này.

1.1.2. Cơ số của hệ đếm

Một số A bất kỳ có thể biểu diễn bằng dãy sau:

$$A = a_{m-1}a_{m-2}.....a_0a_{-1}.....a_{-n}$$

Trong đó a_i là các chữ số, ($i = \overline{-n \div m-1}$); i là các hàng số, i nhỏ: hàng trẻ, i lớn: hàng già.

Giá trị số lượng của các chữ số a_i sẽ nhận một giá trị nào đó sao cho thỏa mãn bất đẳng thức sau:

$$0 \le a_i \le N-1 \qquad (a_i \text{ nguyên})$$

N được gọi là cơ số của hệ đếm. **Cơ số của một hệ đếm là số lượng ký tự phân biệt được sử dụng trong một hệ đếm.** Các hệ thống số đếm được phân biệt với nhau bằng một cơ số N của hệ đếm đó. Mỗi ký tự biểu diễn một chữ số.

Trong đời sống hằng ngày chúng ta quen sử dụng hệ đếm thập phân (*decimal*) với N=10. Trong hệ thống số còn sử dụng những hệ đếm khác là hệ đếm nhị phân (*binary*) với N=2, hệ đếm bát phân (*octal*) với N=8 và hệ đếm thập lục phân (*hexadecimal*) với N=16.

- Hệ nhị phân : N =2 $\Rightarrow$ a_i = 0, 1.
- Hệ thập phân : N =10 $\Rightarrow$ a_i = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
- Hệ bát phân : N =8 $\Rightarrow$ a_i = 0, 1, 2, 3, 4, 5, 6, 7.
- Hệ thập lục phân : N =16 $\Rightarrow$ a_i = 0, 1, 2, …8, 9, A, B, C,D, E, F.

Khi đã xuất hiện cơ số N, ta có thể biểu diễn số A dưới dạng một đa thức theo cơ số N, được ký hiệu là $A_{(N)}$:

$$A_{(N)} = a_{m-1}.N^{m-1} + a_{m-2}.N^{m-2} +...+ a_0.N^0 + a_{-1}.N^{-1} + ... + a_{-n}.N^{-n}$$

Hay:

$$A_{(N)} = \sum_{i=-n}^{m-1} a_i N^i \qquad (1.1)$$

Với N=10 (hệ thập phân):

$$A_{(10)} = a_{m-1}.10^{m-1} + a_{m-2}.10^{m-2} +....+ a_0.10^0 +...+ a_{-n}.10^{-n}$$

$$1999{,}959_{(10)} = 1.10^3 + 9.10^2 + 9.10^1 + 9.10^0 + 9.10^{-1} + 5.10^{-2} + 9.10^{-3}$$

Với N=2 (hệ nhị phân):

$$A_{(2)} = a_{m-1}.2^{m-1} + a_{m-2}.2^{m-2} +...+ a_0.2^0 ....+a_{-n}2^{-n}$$

$$1101_{(2)} = 1.2^3 +1.2^2 + 0.2^1 + 1.2^0 = 13_{(10)}$$

Với N=16 (hệ thập lục phân):

$$A_{(16)} = a_{m-1}.16^{m-1} + a_{m-2}.16^{m-2} +...+ a_0.16^0 + a_{-1}16^{-1} + ... + a_{-n}16^{-n}$$

$$3FF_{(16)} = 3.16^2 + 15.16^1 + 15.16^0 = 1023_{(10)}$$

Với N=8 (hệ bát phân):

$$A_{(8)} = a_{m-1}.8^{m-1} + a_{m-2}.8^{m-2} +...+ a_0.8^0 + a_{-1}.8^{-1} + ... + a_{-n}.8^{-n}$$

$$376_{(8)} = 3.8^2 + 7.8^1 + 6.8^0 = 254_{(10)}$$

Như vậy, biểu thức (1.1) cho phép đổi các số ở bất kỳ hệ nào sang hệ thập phân (hệ 10).

1.1.3. Đổi cơ số

1. Đổi từ cơ số d sang cơ số 10

Để chuyển đổi một số ở hệ đếm cơ số d sang hệ đếm cơ số 10 người ta khai triển con số trong cơ số d dưới dạng đa thức theo cơ số của nó (theo biểu thức 1.3).

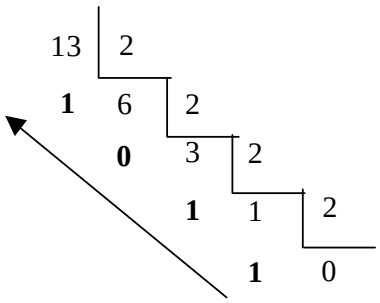
Ví dụ 1.1 Đổi số $1101_{(2)}$ ở hệ nhị phân sang hệ thập phân như sau:

$$1011_{(2)} = 1.2^3 + 0.2^2 + 1.2^1 + 1.2^0 = 11_{(10)}$$

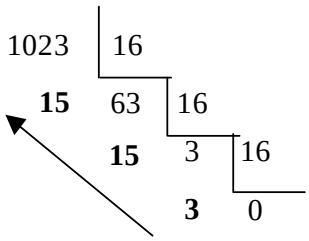
2. Đổi từ cơ số 10 sang cơ số d

Để chuyển đổi một số từ cơ số 10 sang cơ số d (d = 2, 8, 16) người ta lấy con số trong cơ số 10 chia liên tiếp cho d đến khi thương số bằng không thì dừng lại. Kết quả chuyển đổi có được trong hệ đếm cơ số d là tập hợp các số dư của phép chia được viết theo thứ tự ngược lại, nghĩa là số dư đầu tiên có trọng số nhỏ nhất. (xem ví dụ 1.2)

Ví dụ 1.2:



$A_{(10)}=13 \rightarrow A_{(2)}=1101$



$A_{(10)}=1023 \rightarrow A_{(16)}=3FFH$

Kết luận: Gọi d_1, d_2, ..,d_n lần lượt là dư số của phép chia số thập phân cho cơ số d ở lần thứ 1, 2, 3, 4, .., n thì kết quả chuyển đổi một số từ hệ đếm cơ số 10 (thập phân) sang hệ đếm cơ số d sẽ là:

$$\mathbf{d_n d_{n-1} d_{n-2} ... d_1,}$$

nghĩa là dư số sau cùng của phép chia là bít có trọng số cao nhất (MSB), còn dư số đầu tiên là bít có trọng số nhỏ nhất (LSB).

Trong các ví dụ trên, cơ số của hệ đếm được ghi ở dạng chỉ số bên dưới. Ngoài ra cũng có thể ký tự chữ để phân biệt như sau:

B - Hệ nhị phân (Binary) O - Hệ bát phân (Octal)
D - Hệ thập phân (Decmal) H - Hệ thập lục phân (Hexadecimal)

Ví dụ: 1010B có nghĩa là $1010_{(2)}$
37FH có nghĩa là $37F_{(16)}$

& *Quy tắc chuyển đổi giữa các hệ đếm cơ số 2, 8, 16 ?*

1.2. HỆ ĐẾM NHỊ PHÂN VÀ KHÁI NIỆM VỀ MÃ

1.2.1. Hệ đếm nhị phân

1. Khái niệm

Hệ đếm nhị phân, còn gọi là hệ đếm cơ số 2, là hệ đếm trong đó người ta chỉ sử dụng hai kí hiệu 0 và 1 để biểu diễn tất cả các số. Hai ký hiệu đó gọi chung là bit hoặc digit, nó đặc trưng cho mạch điện tử có hai trạng thái ổn định hay còn gọi là 2 trạng thái bền của FLIP- FLOP (ký hiệu là FF).

Trong hệ đếm nhị phân người ta quy ước như sau:

- Một nhóm 4 bít gọi là 1 nibble.
- Một nhóm 8 bít gọi là 1 byte.
- Nhóm **nhiều bytes gọi là từ (word),** có thể có từ 2 bytes (16 bít), từ 4 bytes (32 bít), ...

Để hiểu rõ hơn một số khái niệm, ta xét số nhị phân 4 bít: $a_3a_2a_1a_0$. Biểu diễn dưới dạng đa thức theo cơ số của nó là:

$$a_3a_2a_1a_{0\,(2)} = a_3.2^3 + a_2.2^2 + a_1.2^1 + a_0.2^0$$

Trong đó:

- 2^3, 2^2, 2^1, 2^0 (hay 8, 4, 2, 1) được gọi là các trọng số.
- a_0 được gọi là bit có trọng số nhỏ nhất, hay còn gọi bit có ý nghĩa nhỏ nhất (**LSB** - **L**east **S**ignificant **B**it), còn gọi là bít trẻ nhất.

- a_3 được gọi là bit có trọng số lớn nhất, hay còn gọi là bít có ý nghĩa lớn nhất (**MSB - M**ost **S**ignificant **B**it), còn gọi là bít già nhất.

Như vậy, với số nhị phân 4 bit $a_3a_2a_1a_0$ trong đó mỗi chữ số a_i (i từ 0 đến 3) chỉ nhận được hai giá trị {0,1} ta có $\mathbf{2^4 = 16}$ tổ hợp nhị phân phân biệt.

Bảng sau đây liệt kê các tổ hợp mã nhị phân 4 bít cùng các giá trị số thập phân, số bát phân và số thập lục phân tương ứng.

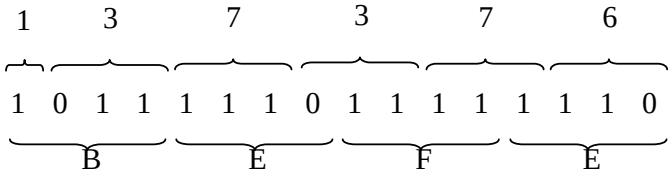
& *Từ bảng này hãy cho biết mối quan hệ giữa các số trong hệ nhị phân với các số trong hệ bát phân (N=8) và hệ thập lục phân (N=16)? Từ đó suy ra phương pháp chuyển đổi nhanh giữa các hệ này?*

Số thập phân	$a_3a_2a_1a_0$	Số bát phân	Số thập lục phân
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Bảng 1.1. Các tổ hợp mã nhị phân 4 bít

Sự chuyển đổi giữa các hệ thống số đếm khác nhau giữ vai trò quan trọng trong máy tính số. Chúng ta biết rằng $2^3 = 8$ và $2^4 = 16$, từ bảng mã trên có thể nhận thấy mỗi chữ số trong hệ bát phân tương đương với một nhóm ba chữ số (3 bít) trong hệ nhị phân, mỗi chữ số trong hệ thập lục phân tương đương với một nhóm bốn chữ số (4 bít) trong hệ nhị phân. Do đó, khi biểu diễn số nhị phân nhiều bit trên máy tính để tránh sai sót người ta thường biểu diễn thông qua số thập phân hoặc thập lục phân hoặc bát phân.

Ví dụ 1.3: Xét việc biểu diễn số nhị phân $1011111011111110_{(2)}$.



Vậy, có thể biểu diễn : $137376_{(8)}$ theo hệ bát phân

hoặc : $BEFE_{(H)}$ theo hệ thập lục phân.

& *Với số nhị phân **n bít** có bao nhiêu tổ hợp nhị phân khác nhau? Xét trường hợp số nhị phân **8 bít** (n=8) $a_7a_6a_5a_4a_3a_2a_1a_0$ có bao nhiêu tổ hợp nhị phân (từ mã nhị phân) khác nhau?*

2. Các phép tính trên số nhị phân

a. Phép cộng

Để cộng hai số nhị phân, người ta dựa trên qui tắc cộng như sau:

0 + 0 = 0 nhớ 0
0 + 1 = 1 nhớ 0
1 + 0 = 1 nhớ 0
1 + 1 = 0 nhớ 1

Ví dụ 1.4:

$$\begin{array}{rcl} +\ 3 & \rightarrow & +\ 0011 \\ \underline{\ 2\ } & \rightarrow & \underline{\ 0010\ } \\ 5 & \rightarrow & 0101 = 1.2^2 + 1.2^0 = 5_{(10)} \end{array}$$

b. Phép trừ

0 - 0 = 0 mượn 0
0 - 1 = 1 mượn 1
1 - 0 = 1 mượn 0
1 - 1 = 0 mượn 0

Ví dụ 1.5:

$$\begin{array}{rcl} -\ 7 & \rightarrow & -\ 0111 \\ \underline{\ 5\ } & \rightarrow & \underline{\ 0101\ } \\ 2 & \rightarrow & 0010 = 0.2^3 + 0.2^2 + 1.2^1 + 0.2^0 = 2_{(10)} \end{array}$$

c. Phép nhân

0 . 0 = 0
0 . 1 = 0
1 . 0 = 0
1 . 1 = 1

Ví dụ 1.6:

$$\begin{array}{rcl} x\ 7 & \rightarrow & x\ 0111 \\ \underline{\ 5\ } & \rightarrow & \underline{\ 0101\ } \\ 35 & & 0111 \\ & & 0000 \\ & & 0111 \\ & & \underline{\ 0000\ } \\ & & 0100011 = 1.2^5 + 1.2^1 + 1.2^0 = 35_{(10)} \end{array}$$

d. Phép chia

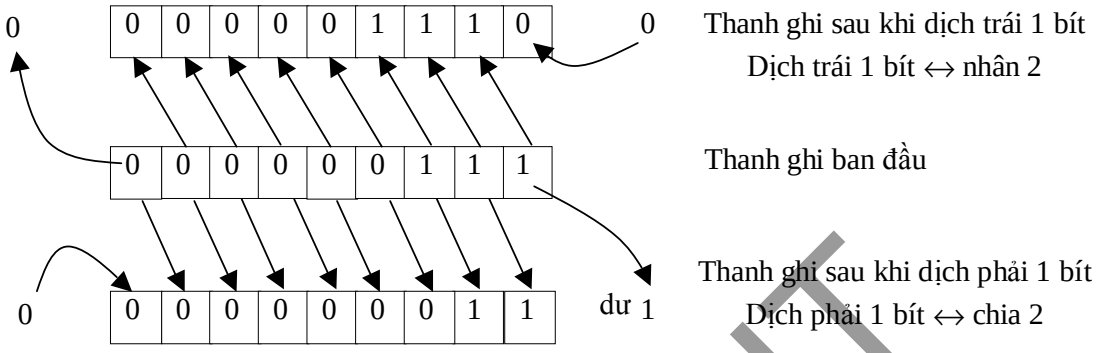
0 : 1 = 0
1 : 1 = 1

Lưu ý: Khi chia số chia phải khác 0

Ví dụ 1.7:

$$10 \mid \underline{5} \qquad \rightarrow 1010 \mid \underline{101}$$
$$2 \qquad\quad 101 \qquad 10_{(2)} = 2_{(10)}$$
$$00$$
$$0$$

Ứng dụng thanh ghi dịch thực hiện phép toán nhân hai, chia hai:



Hình 1.1. Ứng dụng thanh ghi dịch thực hiện phép toán nhân và chia 2

1.2.2. Khái niệm về mã

1. Đại cương

Trong đời sống hàng ngày, con người giao tiếp với nhau thông qua một hệ thống ngôn ngữ qui ước, nhưng trong máy tính và các hệ thống số chỉ xử lý các dữ liệu nhị phân. Do đó, một vấn đề đặt ra là làm thế nào tạo ra một giao diện dễ dàng giữa người và máy tính, nghĩa là máy tính thực hiện được những bài toán do con người đặt ra.

Vì các máy tính số hiện nay chỉ hiểu các số 0 và số 1, nên bất kỳ thông tin nào dưới dạng các chữ số, chữ cái hoặc các ký tự phải được biến đổi thành dạng số nhị phân trước khi nó có thể được xử lý bằng các mạch số.

Để thực hiện điều đó, người ta đặt ra vấn đề về mã hóa dữ liệu. Như vậy, mã hóa là quá trình biến đổi những ký hiệu quen thuộc của con người sang những ký hiệu quen thuộc với máy tính. Những số liệu đã mã hóa này được nhập vào máy tính, máy tính tính toán xử lý và sau đó máy tính thực hiện quá trình ngược lại là giải mã để chuyển đổi các bít thông tin nhị phân thành các ký hiệu quen thuộc với con người mà con người có thể hiểu được.

Các lĩnh vực mã hóa bao gồm:

- Mã hóa số thập phân
- Mã hóa ký tự
- Mã hóa tập lệnh
- Mã hóa tiếng nói
- Mã hóa hình ảnh ..v..v..

Phần tiếp theo chúng ta khảo sát lĩnh vực mã hóa đơn giản nhất là mã hóa số thập phân bằng cách sử dụng các từ mã nhị phân. Việc mã hóa ký tự, tập lệnh, tiếng nói, hình ảnh... đều dựa trên cơ sở mã hóa số thập phân.

2. Mã hóa số thập phân

a. Khái niệm

Trong thực tế để mã hóa số thập phân người ta sử dụng các số nhị phân 4 bit ($a_3a_2a_1a_0$) theo quy tắc sau:

$0 \rightarrow 0000$;	$5 \rightarrow 0101$
$1 \rightarrow 0001$;	$6 \rightarrow 0110$
$2 \rightarrow 0010$;	$7 \rightarrow 0101$
$3 \rightarrow 0011$;	$8 \rightarrow 1000$
$4 \rightarrow 0100$;	$9 \rightarrow 1001$

Các số nhị phân dùng để mã hóa các số thập phân được gọi là các số BCD (Binary Coded Decimal: Số thập phân được mã hóa bằng số nhị phân).

b. Phân loại

Khi sử dụng số nhị phân 4 bit để mã hóa các số thập phân tương ứng với $2^4 = 16$ tổ hợp mã nhị phân phân biệt.

Do việc chọn 10 tổ hợp trong 16 tổ hợp để mã hóa các ký hiệu thập phân từ 0 đến 9 mà trong thực tế xuất hiện nhiều loại mã BCD khác nhau.

Mặc dù tồn tại nhiều loại mã BCD khác nhau, nhưng có thể chia làm hai loại chính: *Mã BCD có trọng số* và *mã BCD không có trọng số*.

b1. Mã BCD có trọng số là loại mã cho phép phân tích thành đa thức theo trọng số của nó. Mã BCD có trọng số được chia làm 2 loại là: mã BCD tự nhiên và mã BCD số học.

Mã BCD tự nhiên là loại mã mà trong đó các trọng số thường được sắp xếp theo thứ tự tăng dần. Ví dụ: Mã BCD 8421, BCD 5421.

Mã BCD số học là loại mã mà trong đó có tổng các trọng số luôn luôn bằng 9.Ví dụ: BCD 2421, BCD 5121, BCD8 4-2-1

Đặc trưng của mã BCD số học là có tính chất đối xứng qua một đường trung gian. Do vậy, để tìm từ mã BCD của một số thập phân nào đó ta lấy bù (đảo) từ mã BCD của số bù 9 tương ứng.

Ví dụ xét mã BCD 2421. Đây là mã BCD số học (tổng các trọng số bằng 9), trong đó số 3 (thập phân) có từ mã là 0011, số 6 (thập phân) là bù 9 của 3. Do vậy, có thể suy ra từ mã của 6 bằng cách lấy bù từ mã của 3, nghĩa là lấy bù 0011, ta sẽ có từ mã của 6 là 1100.

b2. Mã BCD không có trọng số là loại mã không cho phép phân tích thành đa thức theo trọng số của nó. Các mã BCD không có trọng số là: Mã Gray, Mã Gray thừa 3.

Đặc trưng của mã Gray là bộ mã trong đó hai từ mã nhị phân đứng kế tiếp nhau bao giờ cũng chỉ khác nhau 1 bit.

Ví dụ: Mã Gray:			Còn với mã BCD 8421:		
2	$\rightarrow$	0011	3	$\rightarrow$	0011
3	$\rightarrow$	0010	4	$\rightarrow$	0100
4	$\rightarrow$	0110			

Các bảng dưới đây trình bày một số loại mã thông dụng.

Bảng 1.2: *Các mã BCD tự nhiên.*

BCD 8421				BCD 5421				BCD quá 3				Số thập phân
a_3	a_2	a_1	a_0	b_3	b_2	b_1	b_0	c_3	c_2	c_1	c_0	
0	0	0	0	0	0	0	0	0	0	1	1	0
0	0	0	1	0	0	0	1	0	1	0	0	1
0	0	1	0	0	0	1	0	0	1	0	1	2
0	0	1	1	0	0	1	1	0	1	1	0	3
0	1	0	0	0	1	0	0	0	1	1	1	4
0	1	0	1	1	0	0	0	1	0	0	0	5
0	1	1	0	1	0	0	1	1	0	0	1	6
0	1	1	1	1	0	1	0	1	0	1	0	7
1	0	0	0	1	0	1	1	1	0	1	1	8
1	0	0	1	1	1	0	0	1	1	0	0	9

Bảng 1.3: *Các mã BCD số học*

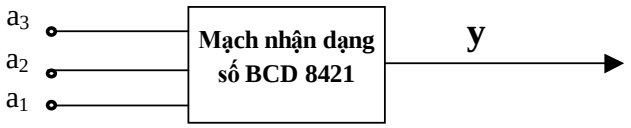
BCD 2421				BCD 5121				BCD 84-2-1				Số thập phân
a_3	a_2	a_1	a_0	b_3	b_2	b_1	b_0	c_3	c_2	c_1	c_0	
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	0	1	1	1	1
0	0	1	0	0	0	1	0	0	1	1	0	2
0	0	1	1	0	0	1	1	0	1	0	1	3
0	1	0	0	0	1	1	1	0	1	0	0	4
1	0	1	1	1	0	0	0	1	0	1	1	5
1	1	0	0	1	1	0	0	1	0	1	0	6
1	1	0	1	1	1	0	1	1	0	0	1	7
1	1	1	0	1	1	1	0	1	0	0	0	8
1	1	1	1	1	1	1	1	1	1	1	1	9

Bảng 1.4: *BCD tự nhiên và mã Gray.*

BCD 8421				BCD quá 3				Mã Gray				Gray quá 3				Số thập phân
a_3	a_2	a_1	a_0	c_3	c_2	c_1	c_0	G_3	G_2	G_1	G_0	g_3	g_2	g_1	g_0	
0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0
0	0	0	1	0	1	0	0	0	0	0	1	0	1	1	0	1
0	0	1	0	0	1	0	1	0	0	1	1	0	1	1	1	2
0	0	1	1	0	1	1	0	0	0	1	0	0	1	0	1	3
0	1	0	0	0	1	1	1	0	1	1	0	0	1	0	0	4
0	1	0	1	1	0	0	0	0	1	1	1	1	1	0	0	5
0	1	1	0	1	0	0	1	0	1	0	1	1	1	0	1	6
0	1	1	1	1	0	1	0	0	1	0	0	1	1	1	1	7
1	0	0	0	1	0	1	1	1	1	0	0	1	1	1	0	8
1	0	0	1	1	1	0	0	1	1	0	1	1	0	1	0	9

Chú ý: Mã Gray được suy ra từ mã BCD 8421 bằng cách: các bit 0,1 đứng sau bit 0 (ở mã BCD 8421) khi chuyển sang mã Gray được giữ nguyên, còn các bit 0,1 đứng sau bit 1 (ở mã BCD 8421) khi chuyển sang mã Gray thì đảo bít, nghĩa là từ bit 1 thành bit 0 và bit 0 thành bit 1.

3. Mạch nhận dạng số BCD 8421:



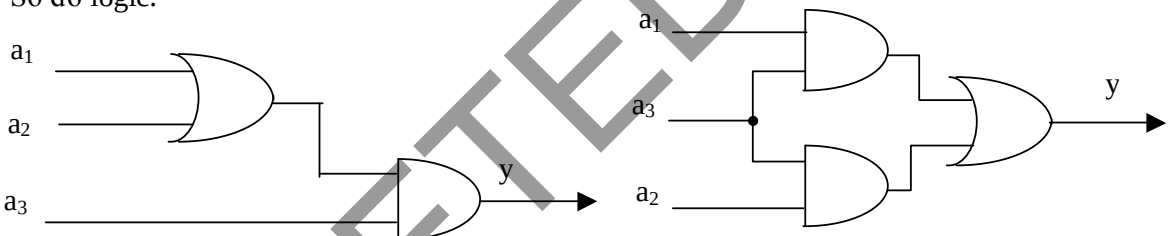
Mạch nhận dạng số BCD 8421 nhận tín hiệu vào là các bít a_3, a_2, a_1 của số nhị phân 4 bít $a_3a_2a_1a_0$, đầu ra y được quy định như sau:

- Nếu y = 1 thì$a_3a_2a_1a_0$ không phải số BCD 8421
- Nếu y = 0 thì$a_3a_2a_1a_0$ là số BCD 8421

Như vậy, nếu một số nhị phân 4 bit không phải là một số BCD 8421 thì ngõ ra y = 1. Từ bảng 1.1 ta thấy một số nhị phân 4 bít không phải là số BCD 8421 khi ***bít a_3 luôn luôn bằng 1 và (bit a_1 bằng 1 hoặc bít a_2 bằng 1)***.

Suy ra phương trình logic của ngõ ra y: $y = a_3(a_1 + a_2) = a_3a_1 + a_3a_2$

Sơ đồ logic:



Cũng do việc xuất hiện số BCD nên có hai cách nhập dữ liệu vào máy tính: nhập số nhị phân, nhập bằng mã BCD.

Để nhập số BCD thập phân hai chữ số thì máy tính chia số thập phân thành các đềcác và mỗi đềcác được biểu diễn bằng số BCD tương ứng. Chẳng hạn: $11_{(10)}$ có thể được nhập vào máy tính theo 2 cách:

- Số nhị phân : 1011
- Mã BCD : 0001 0001

4. Các phép tính trên số BCD

a. Phép cộng

Do số BCD chỉ có từ 0 đến 9 nên đối với những số thập phân lớn hơn sẽ chia số thập phân thành nhiều đềcác, mỗi đềcác được biểu diễn bằng số BCD tương ứng.

Ví dụ 1.8 Cộng 2 số BCD một đềcác:

```
+ 5 → + 0101          + 7 → + 0111
  3 → + 0011            5 → + 0101
  8     1000           12     1100
                      + 0110  ← Số hiệu chỉnh
                        0001  0010
```

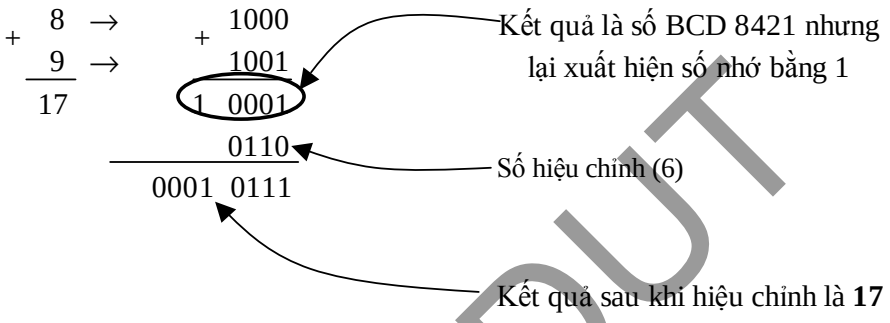
Có hai trường hợp phải hiệu chỉnh kết quả của phép cộng 2 số BCD 8421:

- *Khi kết quả của phép cộng là một số không phải là số BCD 8421*
- *Khi kết quả của phép cộng là một số BCD 8421 nhưng lại xuất hiện số nhớ bằng 1.*

Việc hiệu chỉnh được thực hiện bằng cách cộng kết quả với số hiệu chỉnh là 6 (0110_2).

Ở *ví dụ 1.8* đã xem xét trường hợp hiệu chỉnh khi kết quả không phải là một số BCD 8421. Trường hợp hiệu chỉnh khi kết quả là một số BCD 8421 nhưng phép cộng lại xuất hiện số nhớ bằng 1 được xem xét trong ví dụ sau đây:

Ví dụ 1.9 Hiệu chỉnh kết quả cộng 2 số BCD một đềcác khi xuất hiện số nhớ bằng 1:



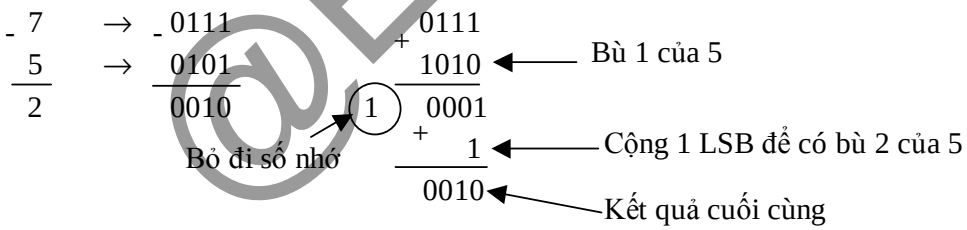
b. Phép trừ

Phép toán trừ 2 số BCD được thực hiện theo quy tắc sau đây:

$$A - B = A + \overline{B}$$

Trong đó $\overline{B}$ là số bù 2 của B.

Ví dụ 1.10 Thực hiện trừ 2 số BCD một đềcác:



Lưu ý:

- *Bù 1 của một số nhị phân là lấy đảo tất cả các bít của số đó (bit 0 thành 1, bit 1 thành 0).*
- *Bù 2 của một số nhị phân bằng số bù 1 cộng thêm 1 vào bít LSB.*

Xét các trường hợp mở rộng sau đây:

1. Thực hiện trừ 2 số BCD 1 đềcác mà số bị trừ nhỏ hơn số trừ ?

2. Mở rộng cho cộng và trừ 2 số BCD nhiều đềcác ?

Chương 2

ĐẠI SỐ BOOLE

2.1. CÁC TIÊN ĐỀ VÀ ĐỊNH LÝ ĐẠI SỐ BOOLE

Trong các mạch số, các tín hiệu thường được cho ở 2 mức điện áp, ví dụ: 0V và 5V. Những linh kiện điện tử dùng trong mạch số làm việc ở một trong hai trạng thái, ví dụ Transistor lưỡng cực (BJT) làm việc ở hai chế độ là tắt hoặc dẫn bão hoà… Do vậy, để mô tả các mạch số người ta dùng hệ nhị phân (binary), hai trạng thái của các linh kiện trong mạch số được mã hoá tương ứng là 0 hoặc 1.

Một bộ môn đại số phát triển từ cuối thế kỷ 19 mang tên người sáng lập ra nó: đại số Boole, còn được gọi là đại số logic, thích hợp cho việc mô tả mạch số. Đại số Boole là công cụ toán học quan trọng để phân tích và thiết kế các mạch số, được dùng làm chìa khoá để đi sâu vào mọi lĩnh vực liên quan đến kỹ thuật số.

2.1.1. Các tiên đề của đại số Boole

Cho một tập hợp B hữu hạn trong đó ta trang bị các phép toán + (cộng logic), x (nhân logic), - (bù logic/nghịch đảo logic) và hai phần tử 0 và 1 lập thành một cấu trúc đại số Boole (đọc là Bun).

$\forall$ x,y $\in$ B thì: x+y $\in$ B, x*y $\in$ B và thỏa mãn 5 tiên đề sau:

1. Tiên đề giao hoán

$\forall x,y \in B$: $x + y = y + x$

2. Tiên đề phối hợp

$\forall x,y,z \in B$: $(x+y)+z = x+(y+z) = x+y+z$

$(x.y).z = x.(y.z) = x.y.z$

3. Tiên đề phân phối

$\forall x,y,z \in B$: $x.(y + z) = x.y + x.z$

$x + (y.z) = (x + y).(x + z)$

4. Tiên đề về phần tử trung hòa

Trong tập B tồn tại hai phần tử trung hòa là **phần tử đơn vị** và **phần tử không.** Phần tử đơn vị ký hiệu là 1, phần tử không ký hiệu là 0.

$\forall x \in B$: $x + 1 = 1$

$x . 1 = x$

$x + 0 = x$

$x . 0 = 0$

5. Tiên đề về phần tử bù

$\forall x \in B$, bao giờ cũng tồn tại phần tử bù tương ứng, ký hiệu $\bar{x}$, sao cho luôn thỏa mãn:

$x + \bar{x} = 1$ và $x.\bar{x} = 0$

Nếu B = B* = {0,1} (B* chỉ gồm 2 phần tử 0 và 1) và thỏa mãn 5 tiên đề trên thì cũng lập thành cấu trúc đại số Boole nhưng là cấu trúc đại số Boole nhỏ nhất.

2.1.2. Các định lý cơ bản của đại số Boole

1. Vấn đề đối ngẫu trong đại số Boole

Hai mệnh đề (hai biểu thức, hai định lý) được gọi là đối ngẫu với nhau nếu trong mệnh đề này người ta thay phép toán cộng thành phép toán nhân và ngược lại, thay 0 bằng 1 và ngược lại, thì sẽ suy ra được mệnh đề kia.

Khi hai mệnh đề đối ngẫu với nhau, nếu 1 trong 2 mệnh đề được chứng minh là đúng thì mệnh đề còn lại là đúng. Dưới đây là ví dụ về các cặp mệnh đề đối ngẫu với nhau.

Ví dụ 2.1: $x.(y+z) = (x.y) + (x.z)$ ⟷ $x + (y.z) = (x+y).(x+z)$ *Hai mệnh đề này là đối ngẫu*

Ví dụ 2.2: $x + \bar{x} = 1$ ⟷ $x.\bar{x} = 0$ *Hai mệnh đề này là đối ngẫu*

2. Các định lý

a. Định lí 1 (Định lý về phần tử bù là duy nhất)

$\forall x, y \in B$, ta có:

$$\left.\begin{array}{l} x+y=1 \\ x.y=0 \end{array}\right\} \Rightarrow y=\bar{x}$$ là duy nhất (x và y là 2 phần tử bù của nhau)

Phần tử bù của một phần tử bất kỳ là duy nhất.

b. Định lí 2 (Đlý về sự đồng nhất của phép cộng và phép nhân logic)

$\forall x \in B$, ta có:

$$x + x + \ldots\ldots + x = x$$
$$x.\,x.\,x.\ldots\ldots x = x$$

c. Định lý 3 (Định lý về phủ định hai lần)

$\forall x \in B$, ta có: $\bar{\bar{x}} = x$

d. Định lí 4 (Định lý De Morgan)

$\forall x, y, z \in B$, ta có:

$$\overline{x+y+z} = \bar{x}.\bar{y}.\bar{z}$$
$$\overline{x.y.z} = \bar{x} + \bar{y} + \bar{z}$$

Hệ quả: $\forall x, y, z \in B$, ta có:

$$x + y + z = \bar{\bar{x} + y + z} = \bar{\bar{x}}.\bar{\bar{y}}.\bar{\bar{z}}$$
$$x.\,y.\,z = \bar{\bar{x}.\bar{y}.\bar{z}} = \bar{\bar{x} + \bar{y} + \bar{z}}$$

e. Định lí 5 (Định lý dán)

$\forall x, y \in B$, ta có:

$$x.\,(\bar{x} + y) = x.y$$
$$x + (\bar{x}.y) = x + y$$

f. Định lí 6 (Định lý nuốt)

$\forall x, y \in B$, ta có:

$$x + x.y = x$$
$$x.(x + y) = x$$

g. Định lí 7 (Quy tắc tính đối với hằng)

Với $0, 1 \in B$, ta có:

$$\bar{0} = 1$$
$$\bar{1} = 0$$

2.2. HÀM BOOLE VÀ CÁC PHƯƠNG PHÁP BIỂU DIỄN

2.2.1. Hàm Boole

1. Định nghĩa

Hàm Boole là một ánh xạ từ đại số Boole vào chính nó. Nghĩa là $\forall x, y \in B$ được gọi là các biến Boole thì hàm Boole, ký hiệu là f, được hình thành trên cơ sở liên kết các biến Boole bằng các phép toán **+ (cộng logic)**, **x / . (nhân logic)**, **nghịch đảo logic (-)**.

Hàm Boole đơn giản nhất là hàm Boole theo 1 biến Boole, được cho như sau:

$$f(x) = x, \quad f(x) = \bar{x}, \quad f(x) = \alpha \ (\alpha \text{ là hằng số })$$

Trong trường hợp tổng quát, ta có hàm Boole theo n biến Boole được ký hiệu như sau:

$$f(x_1, x_2, ..., x_n)$$

2. Các tính chất của hàm Boole

Nếu $f(x_1, x_2, ..., x_n)$ là một hàm Boole thì:

- $\alpha.f(x_1, x_2, ..., x_n)$ cũng là một hàm Boole.
- $\bar{f}(x_1, x_2, ..., x_n)$ cũng là một hàm Boole.

Nếu $f_1(x_1, x_2, ..., x_n)$ và $f_2(x_1, x_2, ..., x_n)$ là những hàm Boole thì:

- $f_1(x_1, x_2, ..., x_n) + f_2(x_1, x_2, ..., x_n)$ cũng là một hàm Boole.
- $f_1(x_1, x_2, ..., x_n).f_2(x_1, x_2, ..., x_n)$ cũng là một hàm Boole.

Vậy, một hàm Boole f cũng được hình thành trên cơ sở liên kết các hàm Boole bằng các phép toán + (cộng logic), x (.) (nhân logic) hoặc nghịch đảo logic (-).

3. Giá trị của hàm Boole

Giả sử $f(x_1, x_2, ..., x_n)$ là một hàm Boole theo n biến Boole.

Trong f người ta thay các biến x_i bằng các giá trị cụ thể α_i ($i = \overline{1, n}$) thì giá trị $f(\alpha_1, \alpha_2, ..., \alpha_n)$ được gọi là giá trị của hàm Boole theo n biến.

Ví dụ 2.3:

Xét hàm $f(x_1, x_2) = x_1 + x_2$

Xét trong tập $B = B^* = \{0,1\}$ ta có các trường hợp sau (lưu ý đây là phép *cộng logic* hay còn gọi phép toán *HOẶC / phép OR*):

- $x_1 = 0, x_2 = 0 \quad \rightarrow f(0,0) = 0 + 0 = 0$

- $x_1 = 0, x_2 = 1 \rightarrow f(0,1) = 0 + 1 = 1$
- $x_1 = 1, x_2 = 0 \rightarrow f(1,0) = 1 + 0 = 1$
- $x_1 = 1, x_2 = 1 \rightarrow f(1,1) = 1 + 1 = 1$

Ta lập được bảng giá trị của hàm trên.

x_1	x_2	$f(x_1, x_2) = x_1 + x_2$
0	0	0
0	1	1
1	0	1
1	1	1

***Ví dụ 2.4*:**

Xét hàm cho bởi biểu thức sau: $f(x_1, x_2, x_3) = x_1 + x_2.x_3$

Xét tập B = B* = {0,1}. Hoàn toàn tương tự ta lập được bảng giá trị của hàm:

x_1	x_2	x_3	$f(x_1, x_2, x_3) = x_1 + x_2.x_3$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

2.2.2. Các phương pháp biểu diễn hàm Boole

1. Phương pháp biểu diễn hàm bằng bảng giá trị

Đây là phương pháp thường dùng để biểu diễn hàm số nói chung và cũng được sử dụng để biểu diễn các hàm logic. Phương pháp này gồm một bảng được chia làm hai phần:

- Một phần dành cho biến để ghi các tổ hợp giá trị có thể có của biến vào.
- Một phần dành cho hàm để ghi các giá trị của hàm ra tương ứng với các tổ hợp biến vào.

Bảng giá trị còn được gọi là bảng chân trị hay bảng chân lý (TRUE TABLE). Như vậy với một hàm Boole n biến bảng chân lý sẽ có:

- **(n+1) cột**: n cột tương ứng với n biến vào, 1 cột tương ứng với giá trị ra của hàm.
- **2^n hàng**: 2^n giá trị khác nhau của tổ hợp n biến.

***Ví dụ 2.5*:** Hàm 3 biến $f(x_1, x_2, x_3)$ có thể được cho bằng bảng giá trị như sau:

x_1	**x_2**	**x_3**	**$f(x_1, x_2, x_3)$**
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Trong các ví dụ 2.3 và 2.4 chúng ta cũng đã quen thuộc với phương pháp biểu diễn hàm bằng bảng giá trị.

2. Phương pháp giải tích

Đây là phương pháp biểu diễn hàm logic bằng các biểu thức đại số. Phương pháp này có 2 dạng: ***tổng của các tích số*** hoặc ***tích của các tổng số***.

Dạng tổng của các tích số gọi là dạng chính tắc thứ nhất (Dạng chính tắc 1 – CT1).

Dạng tích của các tổng số gọi là dạng chính tắc thứ hai (Dạng chính tắc 2 – CT2).

Hai dạng chính tắc này là đối ngẫu nhau.

Dạng tổng các tích số còn gọi là dạng *chuẩn tắc tuyển* (CTT), dạng tích các tổng số còn gọi là dạng *chuẩn tắc hội* (CTH).

a. Dạng chính tắc 1(Dạng tổng của các tích số)

Xét các hàm Boole một biến đơn giản: f(x) = x, f(x) = $\bar{x}$, f(x) = α (α là hằng số).

Đây là những trường hợp có thể có đối với hàm Boole 1 biến.

Chúng ta sẽ đi chứng minh biểu thức tổng quát của hàm logic 1 biến số đối với dạng chính tắc 1. Sau đó áp dụng biểu thức tổng quát của hàm 1 biến để tìm biểu thức tổng quát của hàm 2 biến với việc xem 1 biến là hằng số. Cuối cùng, chúng ta suy ra biểu thức tổng quát của hàm logic n biến cho trường hợp dạng chính tắc 1 (tổng các tích số).

Xét f(x) = x:

Ta có: $x = 0.\bar{x} + 1.x$

mặt khác:

$$f(x) = x \Rightarrow \begin{cases} f(1) = 1 \\ f(0) = 0 \end{cases}$$

Suy ra: f(x) = x có thể biểu diễn:

$$f(x) = x = f(0).\bar{x} + f(1).x$$

trong đó: f (0), f (1) được gọi là các giá trị của hàm Boole theo một biến.

Xét f(x) = $\bar{x}$:

Ta có: $\bar{x} = 1.\bar{x} + 0.x$

Mặt khác:

$$f(x) = \bar{x} \Rightarrow \begin{cases} f(1) = 0 \\ f(0) = 1 \end{cases}$$

Suy ra: f(x) = $\bar{x}$ có thể biểu diễn:

$$f(x) = \bar{x} = f(0).\bar{x} + f(1).x$$

Xét f(x) = α (α là hằng số):

Ta có: $\alpha = \alpha.1 = \alpha.(x + \bar{x}) = \alpha.\bar{x} + \alpha.x$

Mặt khác:

$$f(x) = \alpha \Rightarrow \begin{cases} f(1) = \alpha \\ f(0) = \alpha \end{cases}$$

Suy ra f(x) = α có thể biểu diễn:

$$f(x) = \alpha = f(0).\bar{x} + f(1).x$$

Kết luận: Dù f(x) = x, f(x) = $\bar{x}$ hay f(x) = α, ta đều có biểu thức tổng quát của hàm một biến viết theo dạng chính tắc thứ nhất như sau:

$$\mathbf{f(x) = f(0).\bar{x} + f(1).x}$$

Vậy **f(x) = f(0).$\bar{x}$ + f(1).x**, trong đó f(0), f(1) là giá trị của hàm Boole theo một biến, được gọi là biểu thức tổng quát của hàm 1 biến viết *ở dạng chính tắc thứ nhất (dạng tổng của các tích).*

<u>Biểu thức tổng quát của hàm hai biến f(x_1, x_2)</u>:

Biểu thức tổng quát của hàm 2 biến viết theo dạng chính tắc thứ nhất cũng hoàn toàn dựa trên cách biểu diễn của dạng chính tắc thứ nhất của hàm 1 biến, trong đó xem một biến là hằng số.

Cụ thể là: nếu xem x_2 là hằng số, x_1 là biến số và áp dụng biểu thức tổng quát của dạng chính tắc thứ nhất cho hàm 1 biến, ta có:

$$f(x_1,x_2) = f(0,x_2).\bar{x}_1 + f(1,x_2).x_1$$

Bây giờ, các hàm $f(0,x_2)$ và $f(1,x_2)$ trở thành các hàm 1 biến số theo x_2. Tiếp tục áp dụng biểu thức tổng quát của dạng chính tắc thứ nhất cho hàm 1 biến, ta có:

$$f(0,x_2) = f(0,0).\bar{x}_2 + f(0,1).x_2$$

$$f(1,x_2) = f(1,0).\bar{x}_2 + f(1,1).x_2$$

Suy ra:

$$f(x_1,x_2) = f(0,0).\bar{x}_1\bar{x}_2 + f(0,1).\bar{x}_1x_2 + f(1,0).x_1\bar{x}_2 + f(1,1).x_1x_2$$

Đây chính là biểu thức tổng quát của dạng chính tắc thứ nhất (dạng tổng của các tích số) viết cho hàm Boole hai biến số $f(x_1,x_2)$.

Biểu thức tổng quát này có thể biểu diễn bằng công thức sau:

$$f(x_1,x_2) = \sum_{e=0}^{2^2-1} f(\alpha_1,\alpha_2)x_1^{\alpha_1}x_2^{\alpha_2}$$

Trong đó e là số thập phân tương ứng với mã nhị phân (α_1,α_2) và:

$$x_1^{\alpha_1} = \begin{cases} x_1 & \text{nếu } \alpha_1 = 1 \\ \bar{x}_1 & \text{nếu } \alpha_1 = 0 \end{cases}$$

$$x_2^{\alpha_2} = \begin{cases} x_2 & \text{nếu } \alpha_2 = 1 \\ \bar{x}_2 & \text{nếu } \alpha_2 = 0 \end{cases}$$

<u>Biểu thức tổng quát cho hàm Boole n biến</u>:

Từ biểu thức tổng quát viết ở dạng chính tắc thứ nhất của hàm Boole 2 biến, ta có thể tổng quát hoá cho hàm Boole **n biến f(x_1,x_2, ..,x_n)** như sau:

$$f(x_1,x_2,..,x_n) = \sum_{e=0}^{2^n-1} f(\alpha_1,\alpha_2,....,\alpha_n)x_1^{\alpha_1}x_2^{\alpha_2}...x_n^{\alpha_n}$$

trong đó e là số thập phân tương ứng với mã nhị phân $(\alpha_1,\alpha_2, ...,\alpha_n)$;

và:
$$x_i^{\alpha_i} = \begin{cases} x_i & \text{nếu } \alpha_i = 1 \\ \bar{x}_i & \text{nếu } \alpha_i = 0 \end{cases} \quad (\text{với } i = 1, 2, 3,\ldots,n)$$

Ví dụ 2.6:

Viết biểu thức của hàm 3 biến theo dạng chính tắc 1:

$$f(x_1,x_2,x_3) = \sum_{e=0}^{2^3-1} f(\alpha_1,\alpha_2,\alpha_3).x_1^{\alpha 1}.x_2^{\alpha 2}.x_3^{\alpha 3}$$

Bảng dưới đây cho ta giá trị của số thập phân e và tổ hợp mã nhị phân $(\alpha_1,\alpha_2,\alpha_3)$ tương ứng:

e	α_1	α_2	α_3
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Biểu thức của hàm 3 biến viết theo dạng tổng các tích như sau:

$$f(x_1, x_2, x_3) = f(0,0,0)\bar{x}_1\bar{x}_2\bar{x}_3 + f(0,0,1)\bar{x}_1\bar{x}_2 x_3$$
$$+ f(0,1,0)\bar{x}_1 x_2\bar{x}_3 + f(0,1,1)\bar{x}_1 x_2 x_3 + f(1,0,0)x_1\bar{x}_2\bar{x}_3$$
$$+ f(1,0,1)x_1\bar{x}_2 x_3 + f(1,1,0)x_1 x_2\bar{x}_3 + f(1,1,1)x_1 x_2 x_3$$

Vậy dạng chính tắc thứ nhất là dạng tổng của các tích số mà trong mỗi tích số chứa đầy đủ các biến Boole dưới dạng thật hoặc dạng bù (nghịch đảo).

b. Dạng chính tắc 2 (tích của các tổng số):

Dạng chính tắc 2 là dạng đối ngẫu của dạng chính tắc 1 nên **biểu thức tổng quát của dạng chính tắc 2 cho n biến** được viết như sau:

$$f(x_1, x_2, ..., x_n) = \prod_{e=0}^{2^n-1} [f(\alpha_1,\alpha_2,\alpha_3) + x_1^{\alpha 1} + x_2^{\alpha 2} + ... + x_n^{\alpha n})]$$

trong đó e là số thập phân tương ứng với mã nhị phân $(\alpha_1,\alpha_2, ...,\alpha_n)$;
và:

$$x_i^{\alpha i} = \begin{cases} \bar{x}_i & \text{nếu } \alpha_i = 1 \\ x_i & \text{nếu } \alpha_i = 0 \end{cases} \quad (\text{với } i = 1, 2, 3,\ldots,n)$$

Ví dụ 2.7: Biểu thức của hàm Boole 2 biến ở dạng tích các tổng số (dạng chính tắc 2) được viết như sau:

$$f(x_1,x_2)=[f(0,0)+x_1+x_2][f(0,1)+x_1+\bar{x}_2][f(1,0)+\bar{x}_1+x_2][f(1,1)+\bar{x}_1+\bar{x}_2]$$

Ví dụ 2.8: Biểu thức của hàm Boole 3 biến ở dạng chính tắc 2:

$$f(x_1,x_2,x_3) = [f(0,0,0)+x_1+x_2+x_3].[f(0,0,1)+x_1+x_2+\bar{x}_3].$$
$$[f(0,1,0)+x_1+\bar{x}_2+x_3].[f(0,1,1)+x_1+\bar{x}_2+\bar{x}_3].$$
$$[f(1,0,0)+\bar{x}_1+x_2+x_3].[f(1,0,1)+\bar{x}_1+x_2+\bar{x}_3].$$
$$[f(1,1,0)+\bar{x}_1+\bar{x}_2+x_3].[f(1,1,1)+\bar{x}_1+\bar{x}_2+\bar{x}_3]$$

Vậy, dạng chính tắc thứ hai là dạng tích của các tổng số mà trong đó mỗi tổng số này chứa đầy đủ các biến Boole dưới dạng thật hoặc dạng bù.

Ví dụ 2.9:

Hãy viết biểu thức biểu diễn cho hàm Boole 2 biến $f(x_1,x_2)$ ở dạng chính tắc 1, với bảng giá trị của hàm được cho như sau:

x_1	x_2	$f(x_1,x_2)$
0	0	0
0	1	1
1	0	1
1	1	1

Viết dưới dạng chính tắc 1 ta có:

$$f(x_1,x_2) = f(0,0).\bar{x}_1\bar{x}_2 + f(0,1).\bar{x}_1.x_2 + f(1,0).x_1.\bar{x}_2 + f(1,1).x_1.x_2$$
$$= 0.\bar{x}_1\bar{x}_2 + 1.\bar{x}_1.x_2 + 1.x_1.\bar{x}_2 + 1.x_1.x_2$$
$$= \bar{x}_1.x_2 + x_1.\bar{x}_2 + x_1.x_2$$

Nhận xét:

- *Dạng chính tắc thứ nhất, tổng của các tích số, là dạng liệt kê tất cả các tổ hợp nhị phân các biến vào sao cho tương ứng với những tổ hợp đó giá trị của hàm ra bằng 1 $\rightarrow$ chỉ cần liệt kê những tổ hợp biến làm cho giá trị hàm ra bằng 1.*
- *Khi liệt kê nếu biến tương ứng bằng 1 được viết ở dạng thật (x_i), nếu biến tương ứng bằng 0 được viết ở dạng bù ($\bar{x}_i$).*

***Ví dụ 2.10*:**

Viết biểu thức biểu diễn hàm $f(x_1,x_2,x_3)$ ở dạng chính tắc 2 với bảng giá trị của hàm ra được cho như sau:

x_3	x_2	x_1	$f(x_1,x_2,x_3)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Viết dưới dạng chính tắc 2 (tích các tổng số):

$$f(x_1,x_2,x_3) = (0+x_1+x_2+x_3).(0+x_1+x_2+\bar{x}_3).(0+x_1+\bar{x}_2+x_3).$$
$$(1+x_1+\bar{x}_2+\bar{x}_3).(1+\bar{x}_1+x_2+x_3).(1+\bar{x}_1+x_2+\bar{x}_3).$$
$$(1+\bar{x}_1+\bar{x}_2+x_3).(1+\bar{x}_1+\bar{x}_2+\bar{x}_3)$$

Áp dụng tiên đề về phần tử trung hòa 0 và 1 ta có:

$x + 1 = 1, \quad x . 1 = x$

$x + 0 = x, \quad x . 0 = 0$

nên suy ra biểu thức trên có thể viết gọn lại:

$f(x_1,x_2,x_3) = (x_1+x_2+x_3).(x_1+x_2+\bar{x}_3).(x_1+\bar{x}_2+x_3)$

Nhận xét:

- *Dạng chính tắc thứ hai là dạng liệt kê tất cả các tổ hợp nhị phân các biến vào sao cho tương ứng với những tổ hợp đó giá trị của hàm ra bằng 0 → chỉ cần liệt kê những tổ hợp biến làm cho giá trị hàm ra bằng 0.*
- *Khi liệt kê nếu biến tương ứng bằng 0 được viết ở dạng thật (x_i), nếu biến tương ứng bằng 1 được viết ở dạng bù ($\bar{x}_i$).*

Ví dụ đơn giản sau giúp SV hiểu rõ hơn về cách thành lập bảng giá trị của hàm, tìm hàm mạch và thiết kế mạch.

Ví dụ 2.11

Hãy thiết kế mạch điện sao cho khi công tắc 1 đóng thì đèn đỏ, khi công tắc 2 đóng đèn đỏ, khi cả hai công tắc đóng đèn đỏ ?

Lời giải:

Đầu tiên, ta qui định trạng thái của các công tắc và bóng đèn:

- Công tắc hở : 0 Đèn tắt : 0
- Công tắc đóng: 1 Đèn đỏ : 1

Bảng trạng thái mô tả hoạt động của mạch như sau:

Công tắc 1	Công tắc 2	Trạng thái đèn
x_1	x_2	$f(x_1,x_2)$
0	0	0
0	1	1
1	0	1
1	1	1

Từ bảng trạng thái có thể viết biểu thức của hàm $f(x_1,x_2)$ theo dạng chính tắc 1 hoặc chính tắc 2.

- Theo dạng chính tắc 1 ta có:

$$f(x_1, x_2) = \bar{x}_1.x_2 + x_1.\bar{x}_2 + x_1.x_2$$
$$= \bar{x}_1.x_2 + x_1(\bar{x}_2 + x_2)$$
$$= \bar{x}_1.x_2 + x_1$$
$$= x_1 + x_2$$

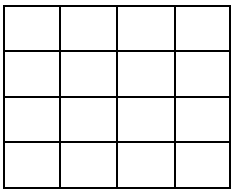
- Theo dạng chính tắc 2 ta có:

$$f(x_1, x_2) = (0+x_1+x_2) = x_1 + x_2$$

Từ biểu thức mô tả trạng thái đỏ/tắt của đèn $f(x_1,x_2)$ thấy rằng có thể thực hiện mạch bằng phần tử logic HOẶC có 2 ngõ vào (cổng OR 2 ngõ vào).

***Bài tập áp dụng**: Một hội đồng giám khảo gồm 3 thành viên. Mỗi thành viên có thể lựa chọn ĐỒNG Ý hoặc KHÔNG ĐỒNG Ý. Kết quả gọi là ĐẠT khi đa số các thành viên trong hội đồng giám khảo ĐỒNG Ý, ngược lại là KHÔNG ĐẠT. Hãy thiết kế mạch giải quyết bài toán trên.*

3. Biểu diễn hàm bằng bảng Karnaugh (bìa Karnaugh)



Đây là cách biểu diễn lại của phương pháp bảng dưới dạng bảng gồm các ô vuông như hình bên.

Trên bảng này người ta bố trí các biến vào theo hàng hoặc theo cột của bảng. Trong trường hợp số lượng biến vào là chẵn, người ta bố trí số lượng biến vào theo hàng ngang bằng số lượng biến vào theo cột dọc của bảng. Trong trường hợp số lượng biến vào là lẻ, người ta bố trí số lượng biến vào theo hàng ngang nhiều hơn số lượng biến vào theo cột dọc 1 biến hoặc ngược lại.

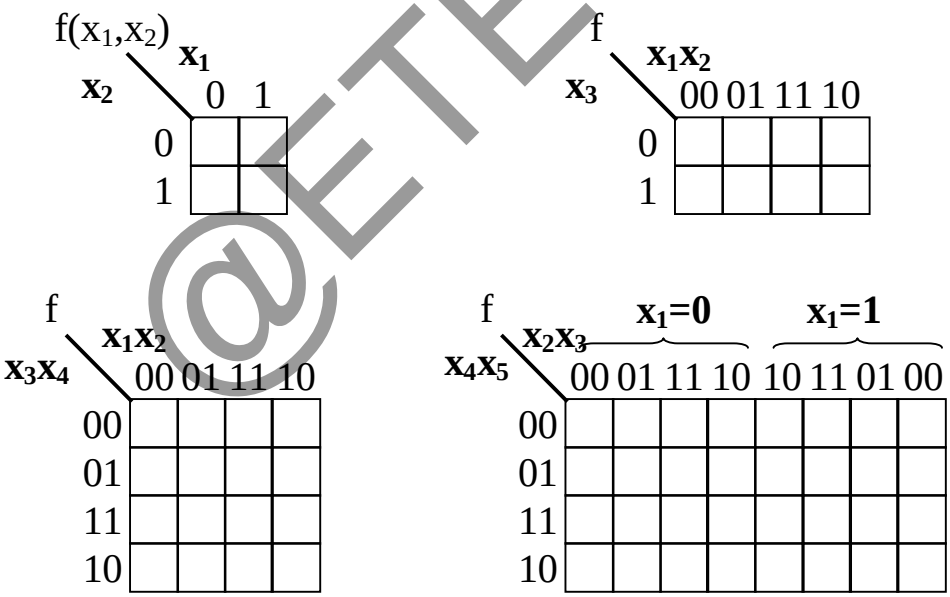
Các tổ hợp giá trị của biến vào theo hàng ngang hoặc theo cột dọc của bảng được bố trí sao cho khi ta đi từ một ô sang một ô lân cận với nó chỉ làm thay đổi một giá trị của biến, như vậy thứ tự bố trí hay sắp xếp các tổ hợp giá trị của biến vào theo hàng ngang hoặc theo cột dọc của bảng Karnaugh hoàn toàn tuân thủ theo mã Gray.

Giá trị ghi trong mỗi ô vuông này chính là giá trị của hàm ra tương ứng với các tổ hợp giá trị của biến vào. Ở những ô mà giá trị hàm là không xác định (có thể bằng 0 hay bằng 1), có nghĩa là giá trị của hàm là tùy ý (hay tùy định), người ta kí hiệu bằng chữ X.

Nếu hàm có n biến vào sẽ có 2^n ô vuông.

Phương pháp biểu diễn hàm bằng bảng Karnaugh chỉ thích hợp cho hàm có tối đa 6 biến, nếu vượt quá việc biểu diễn sẽ rất rắc rối.

Dưới đây là bảng Karnaugh cho các trường hợp hàm 2 biến, 3 biến, 4 biến và 5 biến:



2.3. TỐI THIỂU HÓA HÀM BOOLE

2.3.1. Đại cương

Trong thiết bị máy tính người ta thường thiết kế gồm nhiều modul (khâu) và mỗi modul này được đặc trưng bằng một phương trình logic. Trong đó, mức độ phức tạp của sơ đồ tùy thuộc vào phương trình logic biểu diễn chúng. Việc đạt được độ ổn định cao hay không là tùy thuộc vào phương trình logic biểu diễn chúng ở dạng tối thiểu hóa hay chưa. Để thực hiện được điều đó, khi thiết kế mạch số người ta đặt ra vấn đề tối thiểu hóa các hàm logic. Điều đó có nghĩa là phương

trình logic biểu diễn sao cho thực sự gọn nhất (số lượng các phép tính và số lượng các số được biểu diễn dưới dạng thật hoặc bù là ít nhất).

Các kỹ thuật để đạt được sự thực hiện hàm Boole một cách đơn giản nhất phụ thuộc vào nhiều yếu tố mà chúng ta cần cân nhắc:

Một là số lượng các phép tính và số lượng các số (số lượng literal) được biểu diễn dưới dạng thật hoặc bù là ít nhất, điều này đồng nghĩa với việc số lượng dây nối và số lượng đầu vào của mạch là ít nhất.

Hai là số lượng cổng cần thiết để thực hiện mạch phải ít nhất, chính số lượng cổng xác định kích thước của mạch. Một thiết kế đơn giản nhất phải ứng với số lượng cổng ít nhất chứ không phải số lượng literal ít nhất.

Ba là số mức logic của các cổng. Giảm số mức logic sẽ giảm trễ tổng cộng của mạch vì tín hiệu sẽ qua ít cổng hơn. Tuy nhiên nếu chú trọng đến vấn đề giảm trễ sẽ phải trả giá số lượng cổng tăng lên.

Bởi vậy trong thực tế không phải lúc nào cũng đạt được lời giải tối ưu cho bài toán tối thiểu hóa.

2.3.2. Các bước tiến hành tối thiểu hóa

- Dùng các phép tối thiểu để tối thiểu hóa các hàm số logic.
- Rút ra những thừa số chung nhằm mục đích tối thiểu hóa thêm một bước nữa các phương trình logic.

2.3.3. Các phương pháp tối thiểu hóa

Có nhiều phương pháp thực hiện tối thiểu hoá hàm Boole và có thể đưa về 2 nhóm là *biến đổi đại số* và dùng *thuật toán*. Phương pháp biến đổi đại số (phương pháp giải tích) dựa vào các tiên đề, định lý, tính chất của hàm Boole để thực hiện tối thiểu hoá.

Ở nhóm *thuật toán* có 2 phương pháp thường được dùng là: phương pháp bảng Karnaugh (còn gọi là bìa Karnaugh – bìa K) dùng cho các hàm có từ 6 biến trở xuống, và phương pháp Quine-Mc.Cluskey có thể sử dụng cho hàm có số biến bất kỳ cũng như cho phép thực hiện tự động theo chương trình được viết trên máy tính.

Trong phần này chỉ giới thiệu 2 phương pháp đại diện cho 2 nhóm:

- Phương pháp *biến đổi đại số* (nhóm biến đổi đại số).
- Phương pháp *bảng Karnaugh* (nhóm thuật toán).

1. Phương pháp biến đổi đại số

Đây là phương pháp tối thiểu hóa hàm Boole (phương trình logic) dựa vào các tiên đề, định lý, tính chất của đại số Boole.

Ví dụ 2.12 Tối thiểu hoá hàm $f(x_1,x_2) = \bar{x}_1x_2 + x_1\bar{x}_2 + x_1x_2$

$$\begin{aligned} f(x_1,x_2) &= \bar{x}_1x_2 + x_1\bar{x}_2 + x_1x_2 \\ &= (\bar{x}_1 + x_1).x_2 + x_1\bar{x}_2 \\ &= x_2 + x_1\bar{x}_2 \\ &= x_2 + x_1 \end{aligned}$$

Ví dụ 2.13 Tối thiểu hoá hàm 3 biến sau

$$f(x_1,x_2,x_3) = \bar{x}_1x_2x_3 + x_1\bar{x}_2\bar{x}_3 + x_1\bar{x}_2x_3 + x_1x_2\bar{x}_3 + x_1x_2x_3$$

$= \bar{x}_1 x_2 x_3 + x_1 \bar{x}_2 \bar{x}_3 + x_1 \bar{x}_2 x_3 + x_1 x_2 (\bar{x}_3 + x_3)$

$= \bar{x}_1 x_2 x_3 + x_1 \bar{x}_2 (\bar{x}_3 + x_3) + x_1 x_2$

$= \bar{x}_1 x_2 x_3 + x_1 (\bar{x}_2 + x_2)$

$= \bar{x}_1 x_2 x_3 + x_1$

$= x_1 + x_2 x_3$

Ví dụ 2.14 Rút gọn biểu thức: f = $\overline{AB+\bar{C}} + A\bar{C} + B$

Áp dụng định lý De Morgan ta có:

f = $\overline{AB}.C + A\bar{C} + B$

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

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

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

= $(\bar{A}+1).C + A\bar{C} + B$

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

= $A + B + C$

Vậy, để thực hiện mạch này có thể dùng cổng OR 3 ngõ vào.

2. Phương pháp bảng Karnaugh

Để tối thiểu hóa hàm Boole bằng phương pháp bảng Karnaugh phải tuân thủ theo qui tắc về ô kế cận: ***"Hai ô được gọi là kế cận nhau là hai ô mà khi ta từ ô này sang ô kia chỉ làm thay đổi giá trị của 1 biến."***

Quy tắc chung của phương pháp rút gọn bằng bảng Karnaugh là gom (kết hợp) các ô kế cận lại với nhau.

Khi gom 2 ô kế cận sẽ loại được 1 biến ($2=2^1$ loại 1 biến).

Khi gom 4 ô kế cận vòng tròn sẽ loại được 2 biến ($4=2^2$ loại 2 biến).

Khi gom 8 ô kế cận vòng tròn sẽ loại được 3 biến ($8=2^3$ loại 3 biến).

Tổng quát, khi gom 2^n ô kế cận vòng tròn sẽ loại được n biến. Những biến bị loại là những biến khi ta đi vòng qua các ô kế cận mà giá trị của chúng thay đổi.

Những điều cần lưu ý:

Vòng gom được gọi là hợp lệ khi trong vòng gom đó có ít nhất 1 ô chưa thuộc vòng gom nào.

Các ô kế cận muốn gom được phải là kế cận vòng tròn nghĩa là ô kế cận cuối cũng là ô kế cận đầu tiên.

Việc kết hợp những ô kế cận với nhau còn tùy thuộc vào phương pháp biểu diễn hàm Boole theo dạng chính tắc 1 hoặc chính tắc 2, cụ thể là:

- *Nếu biểu diễn hàm theo dạng chính tắc 1 (tổng các tích số)* ta chỉ quan tâm những ô kế cận có giá trị bằng 1 và tùy định. Kết quả mỗi vòng gom lúc này sẽ là một tích rút gọn. Kết quả của hàm biểu diễn theo dạng chính tắc 1 sẽ là tổng tất cả các tích số rút gọn của tất cả các vòng gom.
- *Nếu biểu diễn hàm theo dạng chính tắc 2 (tích các tổng số)* ta chỉ quan tâm những ô kế cận có giá trị bằng 0 và tùy định. Kết quả mỗi vòng gom lúc này sẽ là một tổng rút gọn.

Kết quả của hàm biểu diễn theo dạng chính tắc 2 sẽ là tích tất cả các tổng số rút gọn của tất cả các vòng gom.

Ta quan tâm những ô tùy định (X) sao cho những ô này kết hợp với những ô có giá trị bằng 1 (nếu biểu diễn theo dạng chính tắc 1) hoặc bằng 0 (nếu biểu diễn theo dạng chính tắc 2) **làm cho số lượng ô kế cận là 2^n lớn nhất.** Lưu ý các ô tùy định (X) chỉ là những ô thêm vào vòng gom để rút gọn hơn các biến mà thôi.

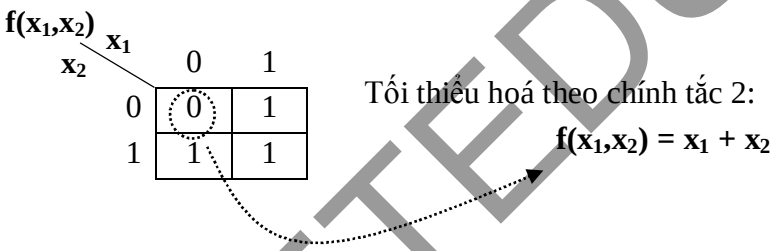
Các vòng gom bắt buộc phải phủ hết tất cả các ô có giá trị bằng 1 có trong bảng (nếu tối thiểu theo dạng chính tắc 1), tương tự các vòng gom bắt buộc phải phủ hết tất cả các ô có giá trị bằng 0 có trong bảng (nếu tối thiểu theo dạng chính tắc 2) thì kết quả tối thiểu hoá mới hợp lệ.

Các trường hợp đặc biệt:

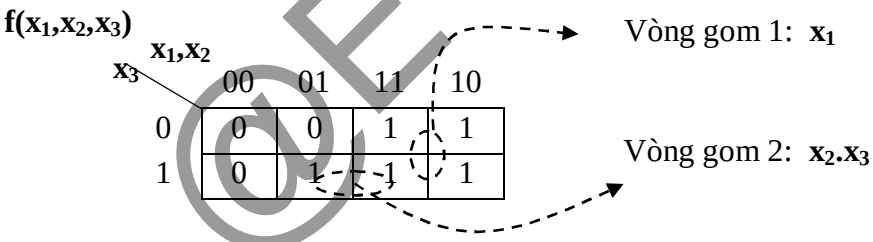
Nếu tất cả các ô của bảng Karnaugh đều bằng 1 và tuỳ định (X) nghĩa là tất cả các ô đều kế cận → giá trị của hàm bằng 1.

Nếu tất cả các ô của bảng Karnaugh đều bằng 0 và tuỳ định (X) nghĩa là tất cả các ô đều kế cận → giá trị của hàm bằng 0.

Ví dụ 2.15: Tối thiểu hóa hàm sau



Ví dụ 2.16:



Tối thiểu theo chính tắc 1: Ta chỉ quan tâm đến những ô có giá trị bằng 1 và tùy định (X), như vậy sẽ có 2 vòng gom để phủ hết các ô có giá trị bằng 1: vòng gom 1 gồm 4 ô kế cận, và vòng gom 2 gồm 2 ô kế cận (hình vẽ).

Đối với vòng gom 1: Có 4 ô = 2^2 nên loại được 2 biến. Khi đi vòng qua 4 ô kế cận trong vòng gom chỉ có giá trị của biến x_1 không đổi (luôn bằng 1), còn giá trị của biến x_2 thay đổi (từ 1→0) và giá trị của biến x_3 thay đổi (từ 0→1) nên các biến x_2 và x_3 bị loại, chỉ còn lại biến x_1 trong kết quả của vòng gom 1. Vì x_1=1 nên kết quả của vòng gom 1 theo dạng chính tắc 1 sẽ có x_1 viết ở dạng thật: x_1

Đối với vòng gom 2: Có 2 ô = 2^1 nên sẽ loại được 1 biến. Khi đi vòng qua 2 ô kế cận trong vòng gom giá trị của biến x_2 và x_3 không đổi, còn giá trị của biến x_1 thay đổi (từ 0→1) nên các biến x_2 và x_3 được giữ lại, chỉ có biến x_1 bị loại. Vì x_2=1 và x_3=1 nên kết quả của vòng gom 2 theo dạng chính tắc 1 sẽ có x_2 và x_3 viết ở dạng thật: $x_2.x_3$

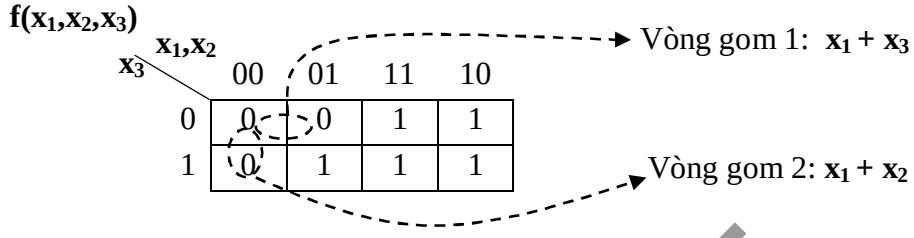
Kết hợp 2 vòng gom ta có kết quả tối giản theo chính tắc 1:

$$f(x_1,x_2,x_3) = x_1 + x_2.x_3$$

Tối thiểu theo chính tắc 2: Ta quan tâm đến những ô có giá trị bằng 0 và tùy định (X), như vậy cũng có 2 vòng gom (hình vẽ), mỗi vòng gom đều gồm 2 ô kế cận.

Đối với vòng gom 1: Có 2 ô = 2^1 nên loại được 1 biến, biến bị loại là x_2 (vì có giá trị thay đổi từ 0→1). Vì x_1=0 và x_3=0 nên kết quả của vòng gom 1 theo dạng chính tắc 2 sẽ có x_1 và x_3 ở dạng thật: $x_1 + x_3$.

Đối với vòng gom 2: Có 2 ô = 2^1 nên loại được 1 biến, biến bị loại là x_3 (vì có giá trị thay đổi từ 0→1). Vì x_1=0 và x_2=0 nên kết quả của vòng gom 2 theo dạng chính tắc 2 sẽ có x_1 và x_2 ở dạng thật: x_1+x_2.



Kết hợp 2 vòng gom có kết quả của hàm f viết theo dạng chính tắc 2 như sau:

$$f(x_1,x_2,x_3) = (x_1+x_3).(x_1+x_2)$$
$$= x_1.x_1 + x_1.x_2 + x_1.x_3 + x_2.x_3$$
$$= x_1 + x_1.x_2 + x_1.x_3 + x_2.x_3$$
$$= x_1(1+ x_2 + x_3) + x_2.x_3$$
$$= x_1 + x_2.x_3$$

Nhận xét: Trong ví dụ này, hàm ra viết theo dạng chính tắc 1 và hàm ra viết theo dạng chính tắc 2 là giống nhau. Tuy nhiên có trường hợp hàm ra của hai dạng chính tắc 1 và 2 là khác nhau, nhưng giá trị của hàm ra ứng với một tổ hợp biến đầu vào là duy nhất trong cả 2 dạng chính tắc.

Chú ý: Người ta thường cho hàm Boole dưới dạng biểu thức rút gọn. Vì có 2 cách biểu diễn hàm Boole theo dạng chính tắc 1 hoặc 2 nên sẽ có 2 cách cho giá trị của hàm Boole ứng với 2 dạng chính tắc đó:

Dạng chính tắc 1: Tổng các tích số.

$$f(x_1,x_2,x_3) = \Sigma(3,4,7) + d(5,6)$$

Trong đó ký hiệu d chỉ giá trị các ô này là tùy định (d: Don't care)

f(x1,x2,x3)

x3 \ x1,x2	00	01	11	10
0	0	0	X	1
1	0	1	1	X

Lúc đó bảng Karnaugh sẽ được cho như hình trên. Từ biểu thức rút gọn của hàm ta thấy tại các ô ứng với tổ hợp nhị phân các biến vào có giá trị là 3, 4, 7 hàm ra có giá trị bằng 1; tại các ô ứng với tổ hợp nhị phân các biến vào có giá trị là 5, 6 hàm ra có giá trị là tùy định; hàm ra có giá trị bằng 0 ở những ô còn lại ứng với tổ hợp các biến vào có giá trị là 0, 1, 2.

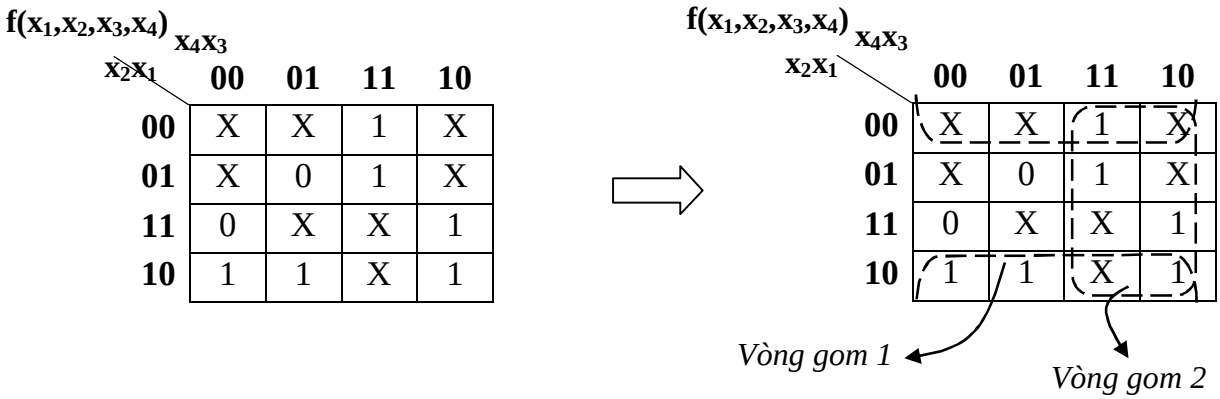
Dạng chính tắc 2: Tích các tổng số.

Phương trình trên cũng tương đương với cách cho hàm như sau:

$$f(x_1,x_2,x_3) = \Pi(0, 1, 2) + d(5, 6)$$

Ví dụ 2.17: Tối thiểu hóa hàm 4 biến cho dưới dạng biểu thức sau:

$f(x_1,x_2,x_3,x_4) = \Sigma(2,6,10,11,12,13) + d(0,1,4,7,8,9,14,15)$



Thực hiện tối thiểu hóa theo dạng chính tắc 1: từ bản đồ Karnaugh ta có 2 vòng gom, *vòng gom 1* gồm 8 ô kế cận và *vòng gom 2* gồm 8 ô kế cận. Kết quả tối thiểu hóa như sau:

Vòng gom 1: $\overline{x}_1$

Vòng gom 2: x_4

Vậy: $f(x_1,x_2,x_3,x_4) = \overline{x}_1 + x_4$

Chương 3

CÁC PHẦN TỬ LOGIC CƠ BẢN

3.1. KHÁI NIỆM VỀ MẠCH SỐ

3.1.1. Mạch tương tự

Mạch tương tự (còn gọi là mạch Analog) là mạch dùng để xử lý các tín hiệu tương tự. Tín hiệu tương tự là tín hiệu có biên độ biến thiên liên tục theo thời gian.

Việc xử lý bao gồm các vấn đề: Chỉnh lưu, khuếch đại, điều chế, tách sóng…

Nhược điểm của mạch tương tự:

- Khả năng chống nhiễu thấp (nhiễu dễ xâm nhập).
- Việc phân tích thiết kế mạch phức tạp.

Để khắc phục những nhược điểm này người ta sử dụng mạch số.

3.1.2. Mạch số

Mạch số (còn gọi là mạch Digital) là mạch dùng để xử lý tín hiệu số. Tín hiệu số là tín hiệu có biên độ biến thiên không liên tục theo thời gian hay còn gọi là tín hiệu gián đoạn, được biểu diễn dưới dạng sóng xung với 2 mức điện thế cao và thấp mà tương ứng với hai mức điện thế này là hai mức logic 1 và 0 của mạch số.

Việc xử lý trong mạch số bao gồm các vấn đề như:

- Lọc số.
- Điều chế số / Giải điều chế số.
- Mã hóa / Giải mã …

Ưu điểm của mạch số so với mạch tương tự :

- Độ chống nhiễu cao (nhiễu khó xâm nhập).
- Phân tích thiết kế mạch số tương đối đơn giản.

Vì vậy, hiện nay mạch số được sử dụng khá phổ biến trong tất cả các lĩnh vực như: Đo lường số, truyền hình số, điều khiển số. . .

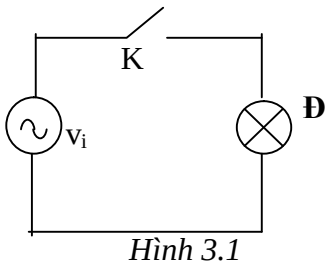
3.1.3. Họ logic dương/âm

Trạng thái logic của mạch số có thể biểu diễn bằng mạch điện đơn giản như trên hình 3.1:

Hoạt động của mạch điện này như sau:

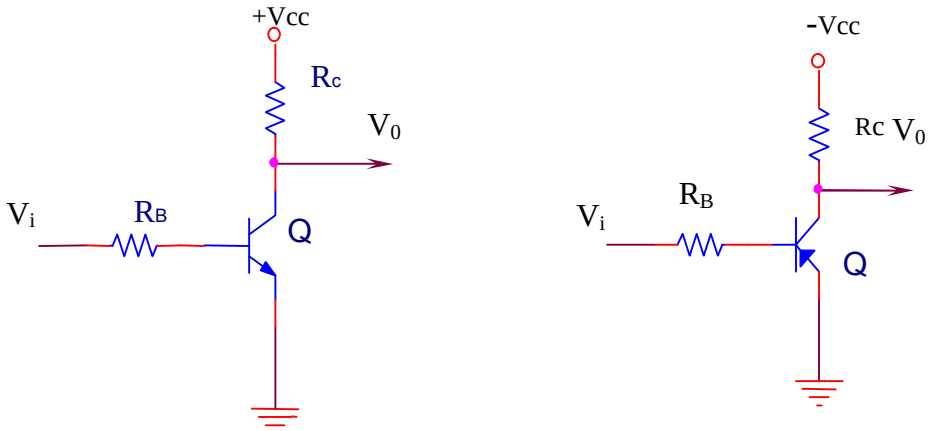
- K Mở : Đèn Tắt
- K Đóng : Đèn Sáng

Trạng thái **Đóng/Mở** của khóa K hoặc trạng thái **Sáng/Tắt** của đèn Đ cũng được đặc trưng cho hai trạng thái logic của mạch số.



Hình 3.1

Cũng có thể thay khóa K bằng khóa điện tử dùng BJT như sau (hình 3.2):



a) b)

Hình 3.2. Biểu diễn trạng thái logic của mạch số bằng khóa điện tử dùng BJT

Giải thích các sơ đồ mạch:

Hình 3.2a:

- Khi $V_i = 0$: BJT tắt $\rightarrow V_0 = +Vcc$
- Khi $V_i > a$: BJT dẫn bão hòa $\rightarrow V_0 = V_{ces} = 0{,}2$ (V) ≈ 0 (V).

Hình 3.2b:

- Khi $V_i = 0$: BJT tắt $\rightarrow V_0 = -Vcc$
- Khi $V_i < -a$: BJT dẫn bão hòa $\rightarrow V_0 = V_{ces} = -V_{ecs} = -0{,}2$ (V) ≈ 0 (V).

Vậy, trong cả 2 sơ đồ mức điện thế vào/ra của khoá điện tử dùng BJT cũng tương ứng với 2 trạng thái logic của mạch số.

Người ta phân biệt ra hai họ logic tùy thuộc vào mức điện áp:

- Nếu chọn : $V_{logic\ 1} > V_{logic\ 0} \rightarrow$ họ logic dương
- Nếu chọn : $V_{logic\ 1} < V_{logic\ 0} \rightarrow$ họ logic âm

Logic dương và logic âm là những họ logic tỏ, ngoài ra còn có họ logic mờ (Fuzzy Logic) hiện đang được ứng dụng khá phổ biến trong các thiết bị điện tử và các hệ thống điều khiển tự động.

3.2. CỔNG LOGIC (LOGIC GATE)

3.2.1. Khái niệm

Cổng logic là một trong các thành phần cơ bản để xây dựng mạch số. Cổng logic được chế tạo trên cơ sở các linh kiện bán dẫn như Diode, BJT, FET để hoạt động theo bảng trạng thái cho trước.

3.2.2 Phân loại

Có ba cách phân loại cổng logic:

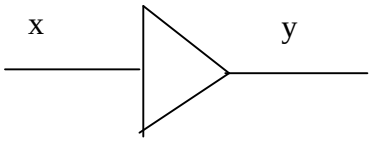
- Phân loại cổng theo chức năng.
- Phân loại cổng theo phương pháp chế tạo.
- Phân loại cổng theo ngõ ra.

1. Phân loại cổng logic theo chức năng

a. Cổng ĐỆM (BUFFER)

Cổng đệm (BUFFER) hay còn gọi là cổng không đảo là cổng có một ngõ vào và một ngõ ra với ký hiệu và bảng trạng thái hoạt động như hình vẽ.

Phương trình logic mô tả hoạt động của cổng đệm: $y = x$



Bảng trạng thái

x	y
0	0
1	1

Hình 3.3. Ký hiệu và bảng trạng thái của cổng đệm

Trong đó:

- x là ngõ vào có trở kháng vào Zv vô cùng lớn → do đó dòng vào của cổng đệm rất nhỏ.
- y là ngõ ra có trở kháng ra Zra nhỏ → cổng đệm có khả năng cung cấp dòng ngõ ra lớn.

Chính vì vậy người ta sử dụng cổng đệm theo 2 ý nghĩa sau:

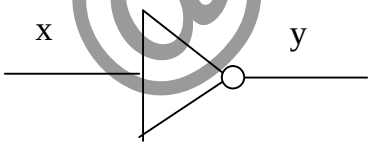
- Dùng để phối hợp trở kháng.
- Dùng để cách ly và nâng dòng cho tải.

Về phương diện mạch điện có thể xem cổng đệm (cổng không đảo) giống như mạch khuyếch đại C chung (đồng pha).

b.Cổng ĐẢO (NOT)

Cổng ĐẢO (còn gọi là cổng NOT) là cổng logic có 1 ngõ vào và 1 ngõ ra, với ký hiệu và bảng trạng thái hoạt động như hình vẽ:

Bảng trạng thái:



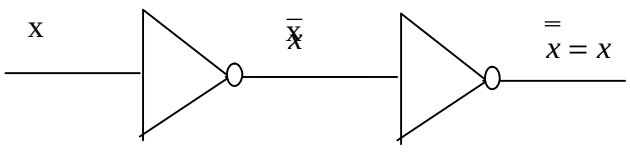
x	y
0	1
1	0

Hình 3.4. Ký hiệu và bảng trạng thái hoạt động của cổng đảo

Phương trình logic mô tả hoạt động của cổng ĐẢO: $y = \overline{x}$

Cổng đảo giữ chức năng như một cổng đệm, nhưng người ta gọi là đệm đảo vì tín hiệu ngõ ra ngược mức logic (ngược pha) với tín hiệu ngõ vào.

Trong thực tế ta có thể ghép hai cổng ĐẢO nối tầng với nhau để thực hiện chức năng của cổng ĐỆM (cổng không đảo) (hình 3.5):



Hình 3.5. Sử dụng 2 cổng ĐẢO tạo ra cổng ĐỆM

Về phương diện mạch điện, cổng ĐẢO giống như tầng khuyếch đại E chung.

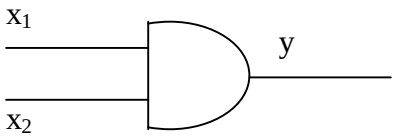
c. Cổng VÀ (AND)

Cổng AND là cổng logic thực hiện chức năng của phép toán nhân logic các tín hiệu vào. Cổng AND 2 ngõ vào có 2 ngõ vào 1 ngõ ra ký hiệu như hình vẽ:

Phương trình logic mô tả hoạt động của cổng AND:

$$y = x_1.x_2$$

Bảng trạng thái hoạt động của cổng AND 2 ngõ vào:



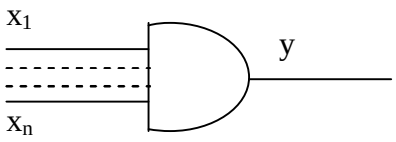
Hình 3.6. Cổng AND

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Từ bảng trạng thái này có nhận xét: Ngõ ra y chỉ bằng 1 (mức logic 1) khi cả 2 ngõ vào đều bằng 1, ngõ ra y bằng 0 (mức logic 0) khi có một ngõ vào bất kỳ (x_1 hoặc x_2) bằng 0.

Xét trường hợp tổng quát cho cổng AND có n ngõ vào x_1, x_2 ... x_n:

$$y_{AND} = \begin{cases} 0 & \exists x_i = 0 \\ 1 & \forall x_i = 1 \quad (i = \overline{1,n}) \end{cases}$$



Hình 3.7. Cổng AND với n ngõ vào

Vậy, đặc điểm của cổng AND là: ngõ ra y chỉ bằng 1 khi tất cả các ngõ vào đều bằng 1, ngõ ra y bằng 0 khi có ít nhất một ngõ vào bằng 0.

Sử dụng cổng AND để đóng mở tín hiệu:

Cho cổng AND có hai ngõ vào x_1 và x_2. Ta chọn:

- x_1 đóng vai trò ngõ vào điều khiển (control).
- x_2 đóng vai trò ngõ vào dữ liệu (data).

Xét các trường hợp cụ thể sau đây:

- Khi $x_1 = 0$: y = 0 bất chấp trạng thái của x_2, ta nói ***cổng AND khóa*** lại không cho dữ liệu đưa vào ngõ vào x_2 qua cổng AND đến ngõ ra.
- Khi $x_1 = 1$ $\begin{cases} x_2 = 0 \Rightarrow y = 0 \\ x_2 = 1 \Rightarrow y = 1 \end{cases} \Rightarrow y = x_2$

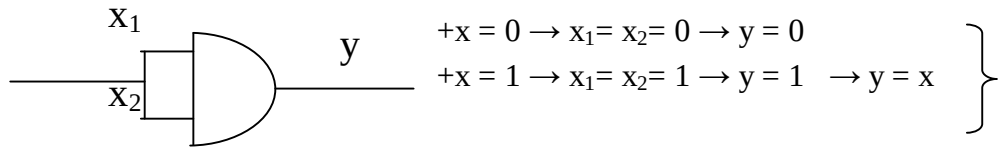
  Ta nói ***cổng AND mở*** cho dữ liệu đưa vào ngõ vào x_2 qua cổng AND đến ngõ ra.

Vậy, có thể sử dụng một ngõ vào bất kỳ của cổng AND đóng vai trò tín hiệu điều khiển cho phép hoặc không cho phép luồng dữ liệu đi qua cổng AND.

Sử dụng cổng AND để tạo ra cổng logic khác:

Nếu sử dụng 2 tổ hợp đầu và cuối trong bảng giá trị của cổng AND và nối cổng AND theo sơ đồ như hình 3.8 thì có thể sử dụng cổng AND để tạo ra cổng đệm.

Trong thực tế, có thể tận dụng hết các cổng chưa dùng trong IC để thực hiện chức năng của các cổng logic khác.

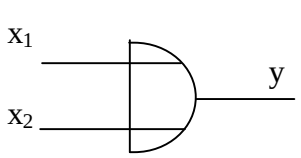


Hình 3.8. Sử dụng cổng AND tạo ra cổng đệm.

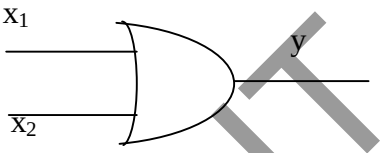
d. Cổng HOẶC (OR)

Cổng OR là cổng thực hiện chức năng của phép toán cộng logic các tín hiệu vào. Trên hình vẽ là ký hiệu của cổng OR 2 ngõ vào:

Phương trình logic cổng OR 2 ngõ vào: $y = x_1 + x_2$



Ký hiệu Châu Âu



Ký hiệu theo Mỹ, Nhật, Úc

Hình 3.9a Cổng OR 2 ngõ vào

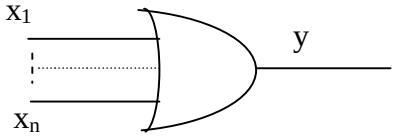
Bảng trạng thái mô tả hoạt động:

x_1	x_2	$y = x_1 + x_2$
0	0	0
0	1	1
1	0	1
1	1	1

Xét trường hợp tổng quát đối với cổng OR có n ngõ vào.

Phương trình logic:

$$y_{OR} = \begin{cases} 1 & \exists x_i = 1 \\ 0 & \forall x_i = 0 \end{cases} \quad (i = \overline{1,n})$$



Hình 3.9b Cổng OR n ngõ vào

Đặc điểm của cổng OR là: Tín hiệu ngõ ra chỉ bằng 0 khi và chỉ khi tất cả các ngõ vào đều bằng 0, ngược lại tín hiệu ngõ ra bằng 1 khi chỉ cần có ít nhất một ngõ vào bằng 1.

Sử dụng cổng OR để đóng mở tín hiệu:

Xét cổng OR có 2 ngõ vào x_1, x_2. Nếu chọn x_1 là ngõ vào điều khiển (control), x_2 ngõ vào dữ liệu (data), ta có các trường hợp cụ thể sau đây:

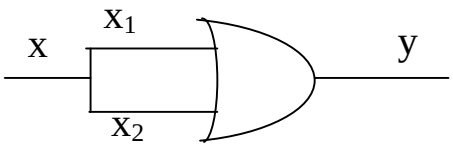
- x_1= 1: y = 1, y luôn bằng 1 bất chấp x_2 → Ta nói ***cổng OR khóa*** không cho dữ liệu đi qua.

- x_1= 0: $\begin{cases} x_2 = 0 \Rightarrow y = 0 \\ x_2 = 1 \Rightarrow y = 1 \end{cases} \Rightarrow y = x_2 \rightarrow$ Ta nói ***cổng OR mở*** cho dữ liệu từ ngõ vào x_2 qua cổng đến ngõ ra y.

<u>Sử dụng cổng OR để thực hiện chức năng cổng logic khác</u>: Sử dụng hai tổ hợp giá trị đầu và cuối của bảng trạng thái của cổng OR và nối mạch cổng OR như sơ đồ hình 3.10:

- x = 0, $x_1 = x_2 = 0 \Rightarrow y = 0$
- x = 1, $x_1 = x_2 = 1 \Rightarrow y = 1$

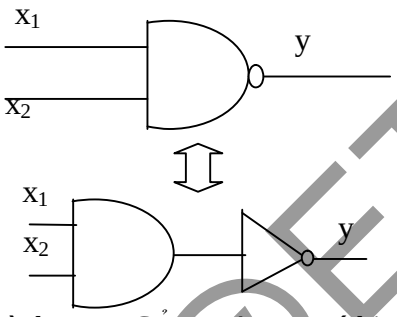
$\Rightarrow$ y = x: cổng OR đóng vai trò như cổng đệm.



Hình 3.10. Sử dụng cổng OR làm cổng đệm

<u>e. Cổng NAND</u>

Đây là cổng thực hiện phép toán nhân đảo, về sơ đồ logic cổng NAND gồm 1 cổng AND mắc nối tầng với 1 cổng NOT, ký hiệu và bảng trạng thái cổng NAND được cho như hình 3.11:



x_1	x_2	y
0	0	1
0	1	1
1	0	1
1	1	0

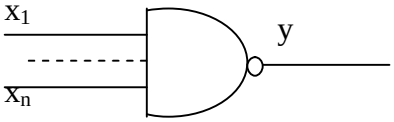
Hình 3.11. Cổng NAND: Ký hiệu, sơ đồ logic tương đương và bảng trạng thái

Phương trình logic mô tả hoạt động của cổng NAND 2 ngõ vào:

$$y = \overline{x_1.x_2}$$

Xét trường hợp tổng quát: Cổng NAND có n ngõ vào.

$$y_{NAND} = \begin{cases} 1 & \exists x_i = 0 \\ 0 & \forall x_i = 1 \quad (i = \overline{1,n}) \end{cases}$$



Hình 3.12.Cổng NAND n ngõ vào

Vậy, đặc điểm của cổng NAND là: tín hiệu ngõ ra chỉ bằng 0 khi tất cả các ngõ vào đều bằng 1, và tín hiệu ngõ ra sẽ bằng 1 khi chỉ cần ít nhất một ngõ vào bằng 0.

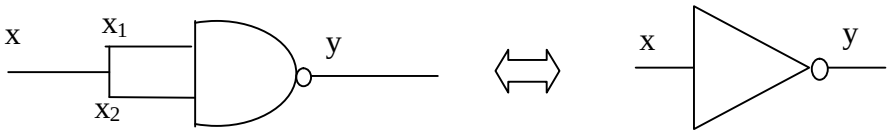
<u>Sử dụng cổng NAND để đóng mở tín hiệu</u>:

Xét cổng NAND có hai ngõ vào. Chọn x_1 là ngõ vào điều khiển (control), x_2 là ngõ vào dữ liệu (data), lần lượt xét các trường hợp sau:

- x_1= 0: y = 1 (y luôn bằng 1 bất chấp giá trị của x_2) ta nói ***cổng NAND khóa***.
- x_1= 1: $\begin{cases} x_2 = 0 \Rightarrow y = 1 \\ x_2 = 1 \Rightarrow y = 0 \end{cases} \Rightarrow y = \overline{x_2} \rightarrow$ ***Cổng NAND mở*** cho dữ liệu vào ngõ vào x_2 đến ngõ ra đồng thời đảo mức tín hiệu ngõ vào x_2, lúc này cổng NAND đóng vai trò là cổng ĐẢO.

Sử dụng cổng NAND để tạo các cổng logic khác:

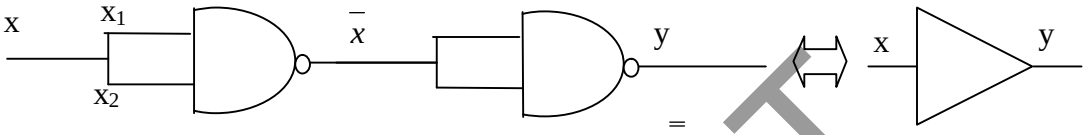
- dùng cổng NAND tạo cổng NOT:



$$y = \overline{x_1 x_2} = \overline{x_1} + \overline{x_2} = \bar{x}$$

Hình 3.13a.Dùng cổng NAND tạo cổng NOT

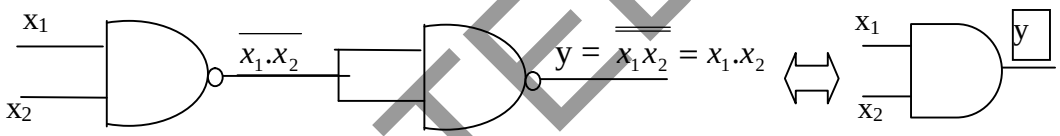
- dùng cổng NAND tạo cổng BUFFER (cổng đệm):



$$y = \bar{\bar{x}} = x$$

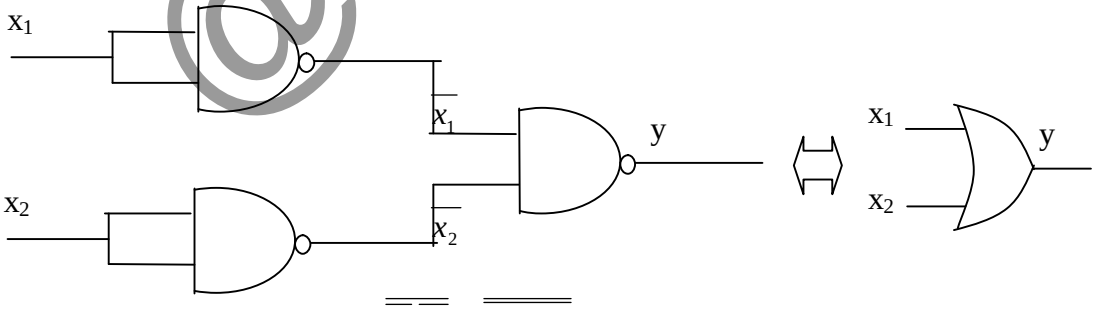
Hình 3.13b.Dùng cổng NAND tạo cổng ĐỆM (BUFFER)

- dùng cổng NAND tạo cổng AND:



Hình 3.13c. Sử dụng cổng NAND tạo cổng AND

- dùng cổng NAND tạo cổng OR:



$$y = \overline{\bar{x}_1.\bar{x}_2} = \overline{\overline{x_1 + x_2}} = x_1 + x_2$$

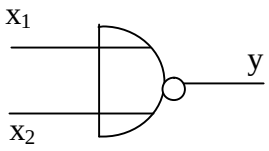
Hình 3.13d. Dùng cổng NAND tạo cổng OR

f. Cổng NOR

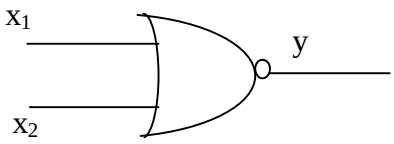
Cổng NOR, còn gọi là cổng Hoặc-Không, là cổng thực hiện chức năng của phép toán cộng đảo logic, là cổng có hai ngõ vào và một ngõ ra có ký hiệu như hình vẽ:

Phương trình logic mô tả hoạt động của cổng :

$$y = \overline{x_1 + x_2}$$



Ký hiệu theo Châu Âu



Ký hiệu theo Mỹ, Nhật

Hình 3.14. Ký hiệu cổng NOR

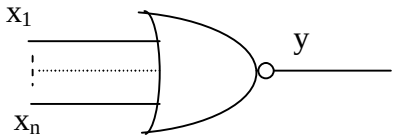
Bảng trạng thái mô tả hoạt động của cổng NOR :

x_1	x_2	y
0	0	1
0	1	0
1	0	0
1	1	0

Xét trường hợp tổng quát cho cổng NOR có n ngõ vào.

$$y_{NOR} = \begin{cases} 0 & \exists x_i = 1 \\ 1 & \forall x_i = 0 \quad (i = \overline{1,n}) \end{cases}$$

Vậy đặc điểm của cổng NOR là: Tín hiệu ngõ ra chỉ bằng 1 khi tất cả các ngõ vào đều bằng 0, tín hiệu ngõ ra sẽ bằng 0 khi có ít nhất một ngõ vào bằng 1.



Hình 3.15. Cổng NOR n ngõ vào

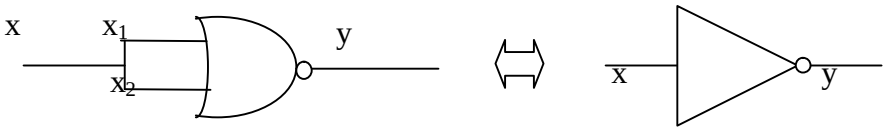
Sử dụng cổng NOR để đóng mở tín hiệu:

Xét cổng NOR có 2 ngõ vào, chọn x_1 là ngõ vào điều khiển, x_2 là ngõ vào dữ liệu. Ta có:

- x_1= 1: y = 0 (y luôn bằng 0 bất chấp x_2), ta nói ***cổng NOR khóa*** không cho dữ liệu đi qua.
- x_1= 0: $\begin{cases} x_2 = 0 \Rightarrow y = 1 \\ x_2 = 1 \Rightarrow y = 0 \end{cases} \Rightarrow y = \overline{x_2}$ → ta nói ***cổng NOR mở*** cho dữ liệu từ ngõ vào x_2 qua cổng NOR đến ngõ ra đồng thời đảo mức tín hiệu ngõ vào x_2, lúc này cổng NOR đóng vai trò là cổng ĐẢO.

Sử dụng cổng NOR để thực hiện chức năng cổng logic khác:

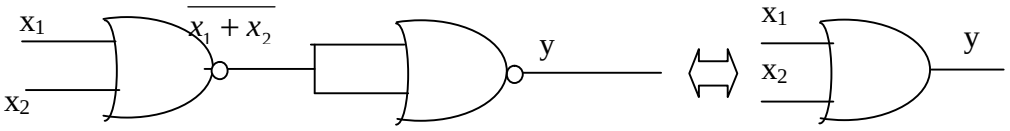
- Dùng cổng NOR làm cổng NOT:



$$y = \overline{x_1 + x_2} = \overline{x_1}.\overline{x_2} = \overline{x}$$

Hình 3.16a. Sử dụng cổng NOR tạo cổng NOT

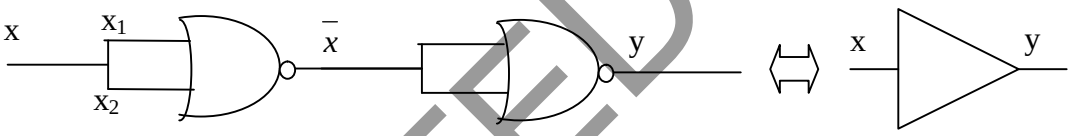
- Dùng cổng NOR làm cổng OR :



$$y = \overline{\overline{x_1 + x_2}} = x_1 + x_2$$

Hình 3.16b. Sử dụng cổng NOR tạo cổng OR

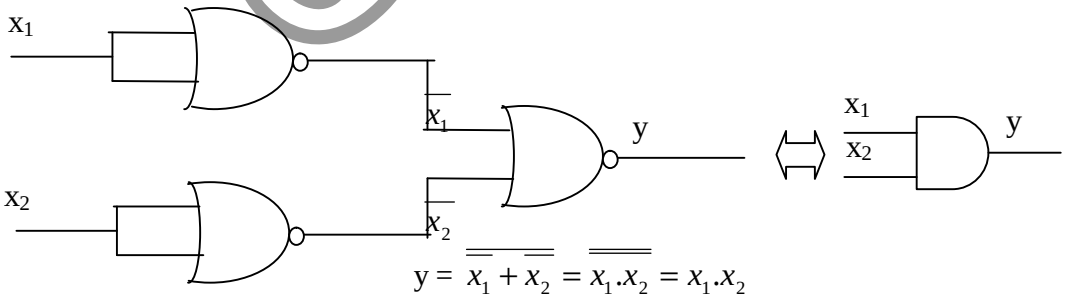
- Dùng cổng NOR làm cổng BUFFER :



$$y = \overline{\overline{x}} = x$$

Hình 3.16c. Sử dụng cổng NOR tạo cổng BUFFER

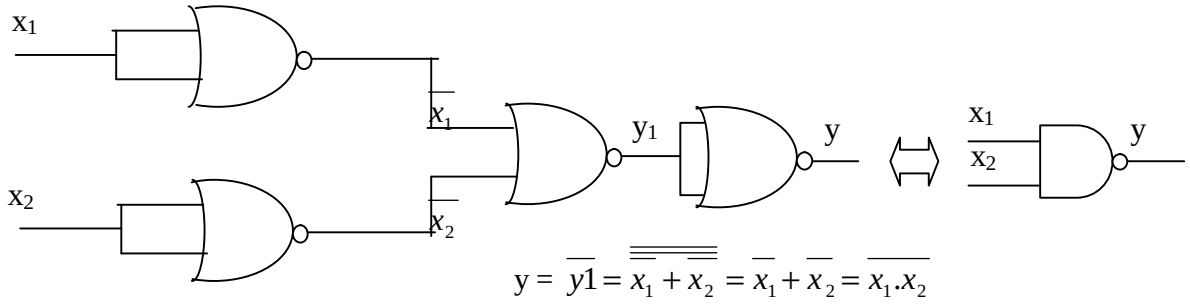
- Dùng cổng NOR làm cổng AND :



$$y = \overline{\overline{x_1} + \overline{x_2}} = \overline{\overline{x_1}.\overline{x_2}} = x_1.x_2$$

Hình 3.16d. Sử dụng cổng NOR làm cổng AND

- Dùng cổng NOR làm cổng NAND:



$$y = \overline{y1} = \overline{\overline{x_1} + \overline{x_2}} = \overline{x_1} + \overline{x_2} = \overline{x_1.x_2}$$

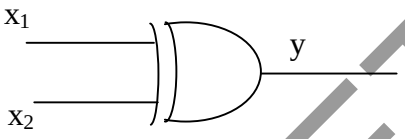
Hình 3.16e. Sử dụng cổng NOR làm cổng NAND

g. Cổng XOR (EX - OR)

Đây là cổng logic thực hiện chức năng của mạch cộng modulo 2 (cộng không nhớ), là cổng có hai ngõ vào và một ngõ ra có ký hiệu và bảng trạng thái như hình vẽ.

Phương trình logic mô tả hoạt động của cổng XOR :

$$y_{XOR} = x_1\overline{x_2} + \overline{x_1}.x_2 = x_1 \oplus x_2$$



Hình 3.17. Cổng XOR

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

Cổng XOR được dùng để so sánh hai tín hiệu vào:

- Nếu hai tín hiệu vào là bằng nhau thì tín hiệu ngõ ra bằng 0
- Nếu hai tín hiệu vào là khác nhau thì tín hiệu ngõ ra bằng 1.

Các tính chất của phép toán XOR:

1. $\mathbf{x_1 \oplus x_2 = x_2 \oplus x_1}$
2. $\mathbf{x_1 \oplus x_2 \oplus x_3 = (x_1 \oplus x_2) \oplus x_3 = x_1 \oplus (x_2 \oplus x_3)}$
3. $\mathbf{x_1.(x_2 \oplus x_3) = (x_1.x_2) \oplus (x_3.x_1)}$

Chứng minh:

$$\text{Vế trái} = x_1(x_2 \oplus x_3) = x_1(x_2.\overline{x}_3 + \overline{x}_2.x_3) = x_1x_2\overline{x}_3 + x_1\overline{x}_2x_3 + x_1\overline{x}_1.x_3 + x_1\overline{x}_1.x_2$$
$$= x_1x_2\overline{x}_3 + x_1\overline{x}_2x_3 + x_1\overline{x}_1.x_3 + x_1\overline{x}_1.x_2 = x_1x_2(\overline{x}_3 + x_1) + x_1x_3(\overline{x}_2 + \overline{x}_1)$$
$$= x_1x_2\overline{x_1x_3} + \overline{x_1x_2}\,x_1x_3 = (x_1x_2) \oplus (x_1x_3) = \text{Vế phải (đpcm).}$$

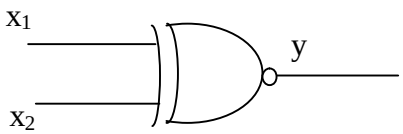
4. $\mathbf{x_1 \oplus (x_2.x_3) = (x_1 \oplus x_3).(x_1 \oplus x_2)}$
5. $\mathbf{x \oplus 0 = x}$; $\mathbf{x \oplus 1 = \overline{x}}$; $\mathbf{x \oplus x = 0}$; $\mathbf{x \oplus \overline{x} = 1}$

Mở rộng tính chất 5: **Nếu $x_1 \oplus x_2 = x_3$ thì $x_1 \oplus x_3 = x_2$**

h. Cổng XNOR (EX – NOR)

Đây là cổng logic thực hiện chức năng của mạch cộng đảo modulo 2 (cộng không nhớ), là cổng có hai ngõ vào và một ngõ ra có ký hiệu và bảng trạng thái như trên hình 3.19.

Phương trình logic mô tả hoạt động của cổng: $y = \overline{x_1\overline{x_2} + \overline{x_1}x_2} = \overline{x_1 \oplus x_2}$



Hình 3.19. Cổng XNOR

x_1	x_2	y
0	0	1
0	1	0
1	0	0
1	1	1

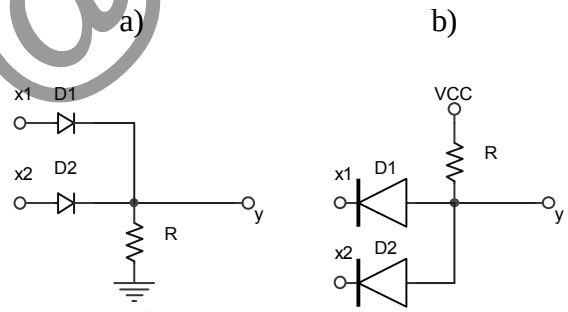
Tính chất của cổng XNOR:

1. $\overline{(x_1 \oplus x_2)(x_3 \oplus x_4)} = \overline{(x_1 \oplus x_2)} + \overline{(x_3 \oplus x_4)}$
2. $\overline{(x_1 \oplus x_2) + (x_3 \oplus x_4)} = \overline{(x_1 \oplus x_2)}\overline{(x_3 \oplus x_4)}$
3. $\overline{x_1 \oplus x_2} = \overline{x_1} \oplus x_2 = x_1 \oplus \overline{x_2}$
4. $x_1 \oplus x_2 = \overline{x_1} \oplus \overline{x_2}$
5. $\overline{x_1 \oplus x_2} = x_3 \Leftrightarrow \overline{x_1 \oplus x_3} = x_2$

Câu hỏi: Hãy thử chứng minh các tính chất từ 1 đến 5 ?

2. Phân loại cổng logic theo phương pháp chế tạo

a. Cổng logic dùng Diode



Hình 3.20. Sơ đồ mạch cổng logic dùng diode
a.Cổng OR - b.Cổng AND

Xét sơ đồ mạch đơn giản trên hình 3.20

Sơ đồ hình a:

- $Vx_1 = Vx_2 = 0V \rightarrow D_1, D_2$ tắt: $V_y = V_R = 0V \rightarrow y = 0$
- $Vx_1 = 0V, Vx_2 = 5V \rightarrow D_1$ tắt, D_2 dẫn: $V_y = V_R = 5V \rightarrow y = 1$
- $Vx_1 = 5V, Vx_2 = 0V \rightarrow D_1$ dẫn, D_2 tắt: $V_y = V_R = 5V \rightarrow y = 1$
- $Vx_1 = Vx_2 = 5V \rightarrow D_1, D_2$ dẫn: $V_y = V_R = 5V \rightarrow y = 1$

x_1	**x_2**	**y**
0	0	0
0	1	1
1	0	1
1	1	1

Đây chính là cổng OR được chế tạo trên cơ sở diode và điện trở hay còn gọi là họ DRL (Diode Resistor Logic) hoặc DL (Diode logic).

Sơ đồ hình b:

- $Vx_1 = Vx_2 = 0V$ $\rightarrow D_1, D_2$ dẫn: $V_y = V_R = 0V \rightarrow y = 0$
- $Vx_1 = 0V, Vx_2 = 5V$ $\rightarrow D_1$ dẫn, D_2 tắt: $V_y = V_R = 0V \rightarrow y = 0$
- $Vx_1 = 5V, Vx_2 = 0V$ $\rightarrow D_1$ tắt, D_2 dẫn: $V_y = V_R = 0V \rightarrow y = 0$
- $Vx_1 = Vx_2 = 5V$ $\rightarrow D_1, D_2$ tắt: $V_y = V_R = 5V \rightarrow y = 1$

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Đây chính là mạch thực hiện chức năng của cổng AND được chế tạo trên cơ sở diode và điện trở (họ DRL hoặc DL).

b. Cổng logic dùng BJT

Họ RTL (Resistor Transistor Logic)

Cổng NOT (hình 3.21a)

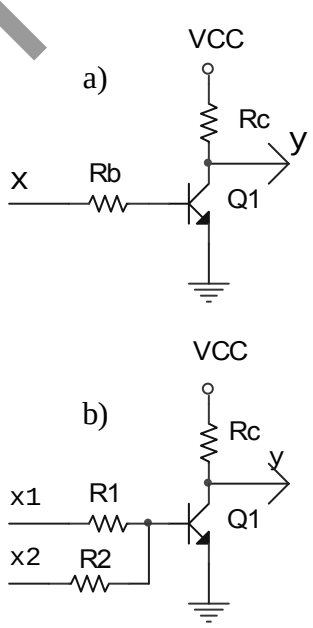
- $x = 0 \rightarrow$ BJT tắt $\rightarrow V_y = V_{cc} = 5V \rightarrow y = 1$
- $x = 1 \rightarrow$ BJT dẫn bão hòa $\rightarrow V_y = V_{ces} \approx 0V \rightarrow y = 0$

Đây là cổng NOT họ RTL (Resistor Transistor Logic).

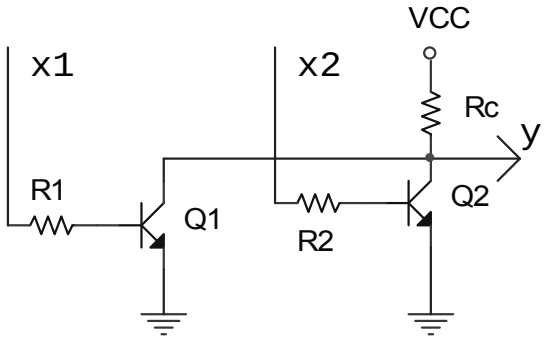
Cổng NOR (hình 3.21b)

- $x_1 = x_2 = 0 \rightarrow$ BJT tắt
  $\Rightarrow V_y = V_{cc} = 5V \Rightarrow y = 1$
- $x_1 = 0, x_2 = 1 \rightarrow$ BJT dẫn bão hoà
  $\Rightarrow V_y = V_{ces} \approx 0V \Rightarrow y = 0$
- $x_1 = 1, x_2 = 0 \rightarrow$ BJT dẫn bão hoà
  $\Rightarrow Vy = V_{ces} \approx 0V \Rightarrow y = 0$
- $x_1 = x_2 = 1 \rightarrow$ BJT dẫn bão hoà
  $\Rightarrow V_y = V_{ces} \approx 0V \Rightarrow y = 0$

Đây chính là cổng NOR họ RTL (Resistor Transistor Logic).



Hình 3.21.(a,b)



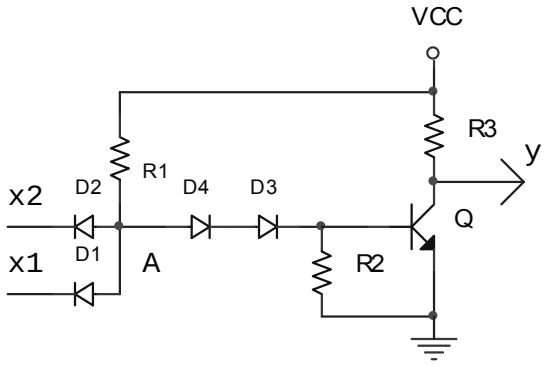
Hình 3.21c. Cổng NOR dùng 2 BJT

Tuy nhiên mạch này có nhược điểm là sự ảnh hưởng giữa các ngõ vào x_1 và x_2 rất lớn đặc biệt là khi hai ngõ vào có mức điện áp (mức logic) ngược nhau. Để khắc phục nhược điểm này người ta cải tiến mạch bằng cách sử dụng 2 BJT ở 2 ngõ vào độc lập với nhau như sơ đồ trên hình 3.21c.

Hãy giải thích hoạt động của mạch này?

Họ DTL (Diode-Transistor-Logic)

Trên hình 3.22 là sơ đồ mạch cổng NAND họ DTL.



Hình 3.22. Cổng NAND họ DTL

- Khi $\mathbf{x_1 = x_2 = 0}$: các diode D_1, D_2 được phân cực thuận nên D_1, D_2 dẫn → $V_A = V\gamma = 0,7V$ (diode ghim điện áp). Mà điều kiện để các diode D_3, D_4 và BJT Q dẫn là:

$$V_A \geq 2V_{\gamma/D} + V_{\gamma/BJT} = 2.0,7 + 0,6 = 2\ (V)$$

→ Khi D_1, D_2 dẫn → D_3, D_4 tắt → BJT tắt → ngõ ra **y = 1**.

- Khi $\mathbf{x_1 = 0, x_2 = 1}$: D_1 dẫn, D_2 tắt → $V_A = 0,7V$ (diode D_1 ghim điện áp) → D_3, D_4, BJT tắt → ngõ ra y = 1.
- Khi $\mathbf{x_1 = 1, x_2 = 0}$: D_1 tắt, D_2 dẫn → $V_A = 0,7V$ (diode D_2 ghim điện áp) → D_3, D_4, BJT tắt → ngõ ra **y = 1**.
- Khi $\mathbf{x_1 = x_2 = 1}$: cả hai diode D_1, D_2 đều tắt → $V_A \approx V_{cc}$, (thực tế $V_A = V_{cc} - V_{R1}$) → điều kiện để diode D_3, D_4 dẫn thoả mãn nên D_3, D_4 dẫn → BJT dẫn bão hòa → ngõ ra **y = 0**.

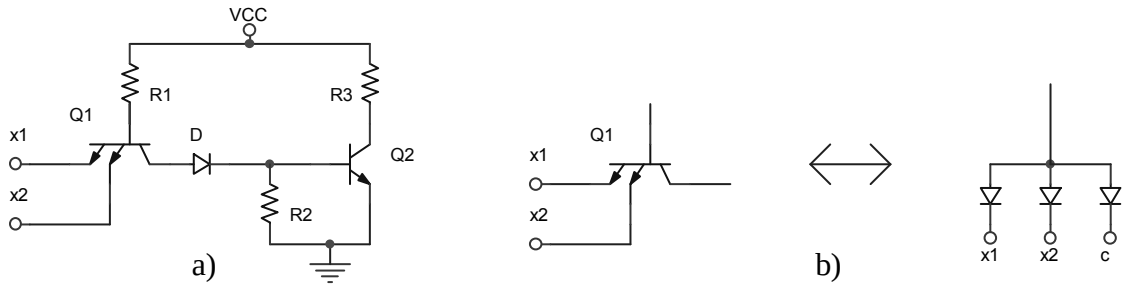
Vậy đây chính là sơ đồ mạch thực hiện cổng NAND họ DTL.

Nhiệm vụ của các linh kiện:

Nếu chỉ có một diode D_3, giả sử $x_1 = x_2 = 0$, ngõ ra y=1, lúc này D_1 và D_2 dẫn, ta có $V_A = V\gamma/D_3 = 0,7(V)$. Nếu có một tín hiệu nhiễu bên ngoài chỉ khoảng 0,6V tác động vào mạch sẽ làm điện áp tại A tăng lên thành 1,3(V), và sẽ làm cho diode D_3 và Q dẫn. Nhưng nếu mắc nối tiếp thêm D_4 mạch có thể ngăn tín hiệu nhiễu lên đến $2V\gamma = 1,2(V)$. Vậy, D_3 và D_4 có tác dụng nâng cao khả năng chống nhiễu của mạch.

Ngoài ra, R_2 làm tăng tốc độ chuyển đổi trạng thái của Q, vì lúc đầu khi Q dẫn sẽ có dòng đổ qua R_2 tạo một phân áp cho tiếp giáp J_E của Q để phân cực thuận làm cho Q nhanh chóng dẫn, và khi Q tắt thì lượng điện tích sẽ xã qua R_2 nên BJT nhanh chóng tắt.

Họ TTL (Transistor - Transistor -Logic)



Hình 3.23. Cổng NAND họ TTL

a. Sơ đồ mạch, b.Transistor 2 tiếp giáp và sơ đồ tương đương

Transistor Q_1 được sử dụng gồm 2 tiếp giáp BE_1, BE_2 và một tiếp giáp BC. Tiếp giáp BE_1, BE_2 của Q_1 thay thế cho D_1, D_2 và tiếp giáp BC thay thế cho D_3 trong sơ đồ mạch cổng NAND họ DTR (hình 3.22).

Giải thích hoạt động của mạch (hình 3.23):

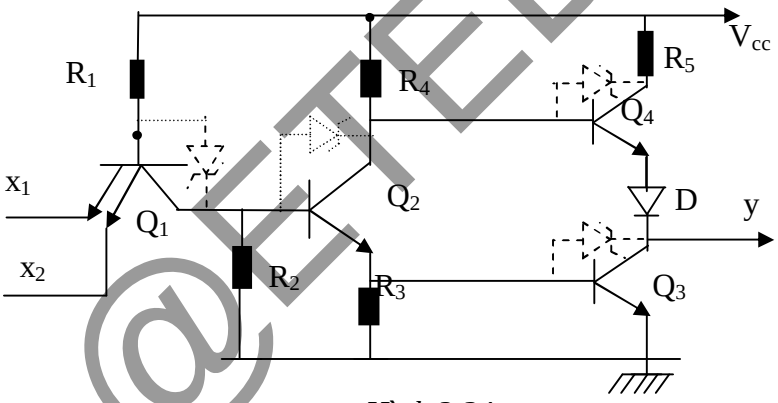
- $\mathbf{x_1 = x_2 = 0}$ các tiếp giáp BE_1, BE_2 sẽ được mở làm cho điện áp cực nền của Q_1 : $V_B = V\gamma$ = 0,6V. Mà điều kiện để cho tiếp giáp BC, diode D và Q_2 dẫn thì điện thế ở cực nền của Q_1 phải bằng:

  $V_B = V_{\gamma/BC} + V_{\gamma/BE1} + V_{\gamma/BE2} = 0,6 + 0,7 + 0,6 = 1,9V$

  Chứng tỏ khi các tiếp giáp BE_1, BE_2 mở thì tiếp giáp BC, diode D và BJT Q_2 tắt $\rightarrow$ **y = 1**.
- $\mathbf{x_1 = 0, x_2 = 1}$ các tiếp giáp BE_1 mở, BE_2 tắt thì tiếp giáp BC, diode D và BJT Q_2 tắt $\rightarrow$ **y = 1**.
- $\mathbf{x_1 = 1, x_2 = 0}$ các tiếp giáp BE_1 tắt, BE_2 mở thì tiếp giáp BC, diode D và BJT Q_2 tắt $\rightarrow$ **y = 1**.
- $\mathbf{x_1 = x_2 = 1}$ các tiếp giáp BE_1, BE_2 tắt thì tiếp giáp BC, diode D dẫn và BJT Q_2 dẫn bão hòa $\rightarrow$ **y = 0**

Vậy, đây chính là mạch thực hiện cổng NAND theo công nghệ TTL.

Để nâng cao khả năng tải của cổng, người ta thường mắc thêm ở ngõ ra một tầng khuếch đại kiểu C chung (CC) như sơ đồ mạch trên hình 3.24:

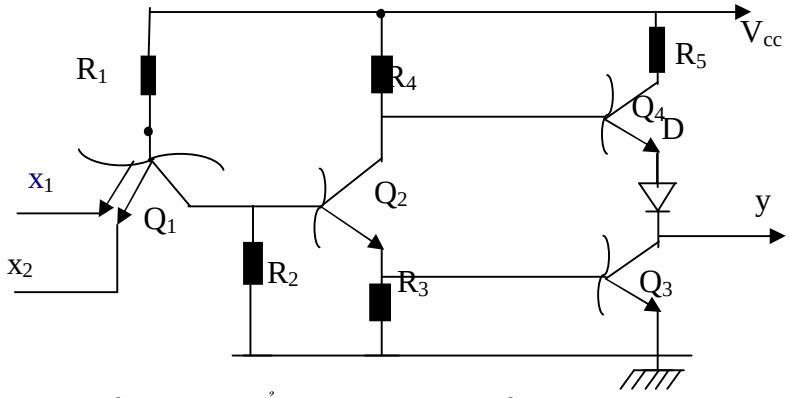


Hình 3.24

Để nâng cao tần số làm việc của cổng, người ta cho các BJT làm việc ở chế độ khuếch đại, điều đó có nghĩa là người ta khống chế để sao cho các tiếp xúc J_C của BJT bao giờ cũng ở trạng thái phân cực ngược. Bằng cách mắc song song với tiếp giáp J_C của BJT một diode Schottky. Đặc điểm của diode Schottky là tiếp xúc của nó gồm một chất bán dẫn với một kim loại, nên nó không tích lũy điện tích trong trạng thái phân cực thuận nghĩa là thời gian chuyển từ phân cực thuận sang phân cực ngược nhanh hơn, nói cách khác BJT sẽ chuyển đổi trạng thái nhanh hơn.

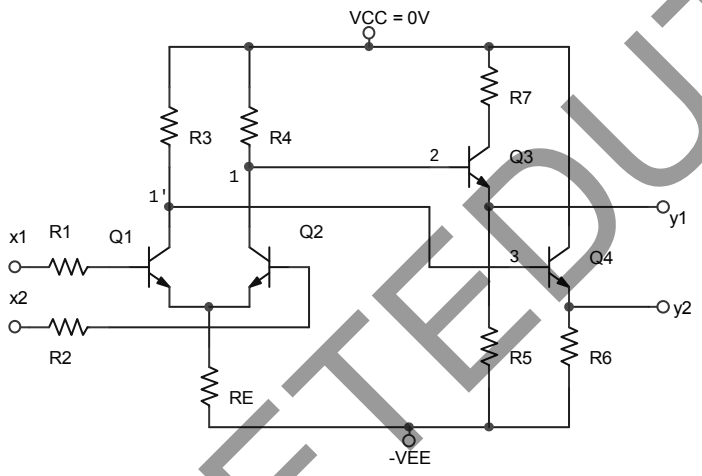
Lưu ý: Người ta cũng không dùng diode Zener bởi vì tiếp xúc của diode Zener là chất bán dẫn nên sẽ tích trữ điện tích dư.

Sơ đồ mạch cải tiến có diode Schottky trên sẽ vẽ tương đương như sau (hình 3.25):



Hình 3.25. Cổng logic họ TTL dùng diode Schottky

Họ ECL (Emitter-Coupled-Logic)



Hình 3.26. Cổng logic họ ECL (Emitter Coupled Logic)

Logic ghép emitter chung (ECL) là họ logic có tốc độ hoạt động rất cao và thường được dùng trong các ứng dụng đòi hỏi tốc độ cao. Tốc độ cao đạt được là nhờ vào các transistor được thiết kế để hoạt động trong chế độ khuyếch đại, vì vậy chúng không bao giờ rơi vào trạng thái bão hoà và do đó thời gian tích luỹ hoàn toàn bị loại bỏ. Họ ECL đạt được thời gian trễ lan truyền nhỏ hơn 1ns trên mỗi cổng.

Nhược điểm của họ ECL: Ngõ ra có điện thế âm nên nó không tương thích về mức logic với các họ logic khác.

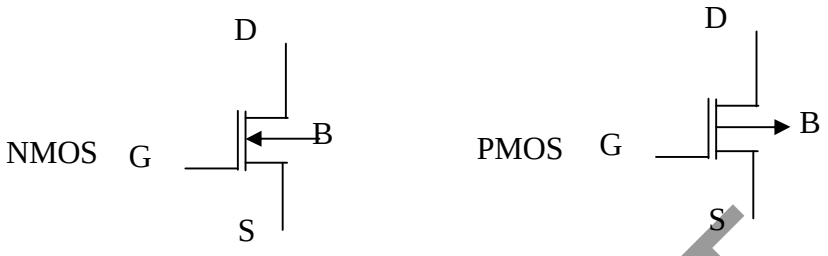
Giải thích hoạt động của mạch (hình 3.26):

- Khi $x_1 = x_2 = 0$: Q_1, Q_2 dẫn nên điện thế tại cực nền (2), (3) của Q_3, Q_4 càng âm (do 1 và 1' âm) nên Q_3, Q_4 tắt $\rightarrow y_1 = 1, y_2 = 1$.
- Khi $x_1 = 0$, $x_2 = 1$: Q_1 dẫn, Q_2 tắt nên điện thế tại cực nền (2) của Q_3 dương, điện thế tại cực nền (3) của Q_4 càng âm nên Q_3 dẫn, Q_4 tắt $\rightarrow y_1 = 0, y_2 = 1$.
- Khi $x_1 = 1$, $x_2 = 0$: Q_1 tắt, Q_2 dẫn nên điện thế tại cực nền (2) của Q_3 âm, điện thế tại cực nền (3) của Q_4 càng dương nên Q3 dẫn, Q_4 tắt $\rightarrow y_1 = 1, y_2 = 0$.
- Khi $x_1 = x_2 = 1$: Q_1, Q_2 tắt nên điện thế tại cực nền (2), (3) của Q_3, Q_4 càng dương nên Q_3, Q_4 dẫn $\rightarrow y_1 = 0, y_2 = 0$.

c. Cổng logic dùng MOSFET

MOSFET (***M****etal* ***O****xyt* ***S****emiconductor* ***F****ield* ***E****ffect* ***T****ransistor*), còn gọi là IGFET (***I****solated* ***G****ate* FET - Transistor trường có cực cổng cách ly).

MOSFET có hai loại: Loại có kênh đặt sẵn và loại có kênh cảm ứng.



a. MOSFET kênh đặt sẵn

b. MOSFET kênh cảm ứng

Hình 3.27. Ký hiệu các loại MOSFET khác nhau

Dù là MOSFET có kênh đặt sẵn hay kênh cảm ứng đều có thể phân chia làm hai loại:

- MOSFET kênh N gọi là NMOS
- MOSFET kênh P gọi là PMOS.

Đặc điểm của 2 loại này khác nhau như sau:

- PMOS: Tiêu thụ công suất thấp, tốc độ chuyển đổi trạng thái chậm.
- NMOS: Tiêu thụ công suất lớn hơn, tốc độ chuyển đổi trạng thái nhanh hơn.

Trên hình 3.27 là ký hiệu của các loại MOSFET khác nhau.

Chú ý: MOSFET kênh đặt sẵn có thể làm việc ở hai chế độ giàu kênh và nghèo kênh trong khi MOSFET kênh cảm ứng chỉ làm việc ở chế độ giàu kênh.

Dùng NMOS kênh cảm ứng chế tạo các cổng logic

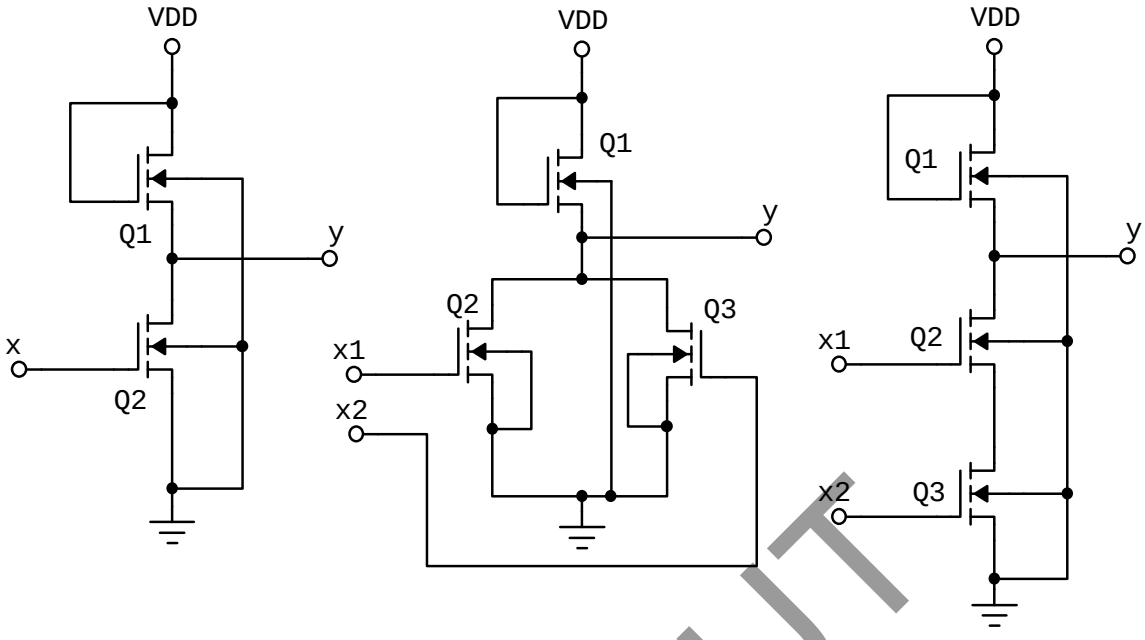
Xét các cổng logic loại NMOS trên hình 3.28.

Điều kiện để cổng NMOS dẫn: $V_D > V_S$, $V_G > V_B$

Trong tất cả hình vẽ ta có :

$$Q_1\begin{cases} R_{DS(ON)} = 200K\Omega \\ R_{DS(OF)} = \end{cases} \qquad Q_2, Q_3\begin{cases} R_{DS(ON)} = 1K\Omega \\ R_{DS(OF)} = 10^7\, K\Omega \end{cases}$$

Hình 3.28a (cổng NOT)



a) Cổng NOT *b) Cổng NOR* *c) Cổng NAND*

Hình 3.28 Các cổng logic chế tạo bằng NMOS

Theo điều kiện để cổng NMOS dẫn: $V_D > V_S$, $V_G > V_B$

Ta thấy Q_1 có B nối mass thỏa mãn điều kiện nên: ***Q_1 luôn luôn dẫn***.

- Khi x = 0: Q_1 dẫn, Q_2 tắt (vì $V_{G2} = V_{B2} = 0$ nên không hình thành điện trường giữa G và B → không hút được các e- là hạt dẫn thiểu số ở vùng đế B → không hình thành được kênh dẫn). Lúc này, theo sơ đồ tương đương (hình 3.29a) ta có:

$$V_y = \frac{R_{DS(OFF)/Q2}}{R_{DS(ON)/Q1} + R_{DS(OFF)/Q2}} V_{DD}$$

$$= \frac{10^7 K}{200K + 10^7 K} V_{DD}$$

$$\Rightarrow V_y \approx V_{DD} \Rightarrow y = 1$$

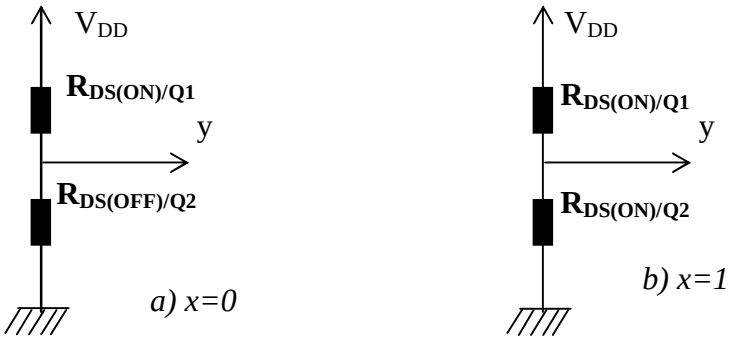
- Khi x = 1: lúc này $V_{G/Q2} > V_{B/Q2}$ → hình thành một điện trường hướng từ G đến B, điện trường này hút các điện tử là các hạt dẫn thiểu số trong vùng đế B di chuyển theo chiều ngược lại về mặt đối diện, hình thành kênh dẫn nối liền giữa G và B và có dòng điện i_D đi từ D qua → Q_2 dẫn. Như vậy Q_1, Q_2 đều dẫn, ta sẽ có sơ đồ tương đương (hình 3.29b). Theo sơ đồ này ta có:

$$V_y = \frac{R_{DS(ON)/Q2}}{R_{DS(ON)/Q1} + R_{DS(ON)/Q2}} V_{DD}$$

$$= \frac{1K}{200K + 1K} V_{DD}$$

$$\Rightarrow V_y \approx \frac{1}{200} V_{DD} = 0{,}025V \Rightarrow y = 0$$

Vậy mạch ở hình 3.28a là mạch thực hiện cổng NOT.



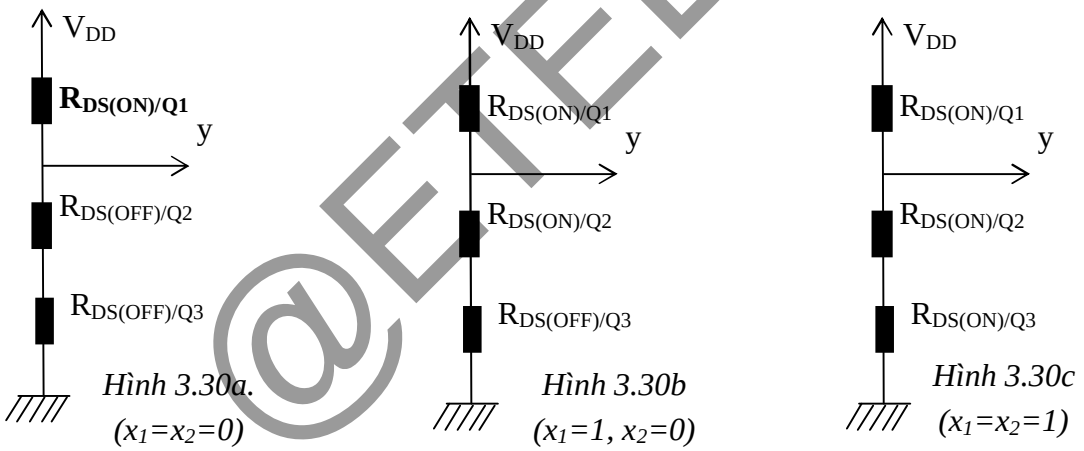
Hình 3.29 Sơ đồ tương đương mạch hình 3.28a

Hình 3.28c (cổng NAND)

- Khi $x_1 = x_2 = 0$ (hình 3.30a): Q_1 luôn dẫn, Q_2 và Q_3 đều tắt, lúc đó theo sơ đồ tương đương ta có:

$$V_y = \frac{R_{DS(OFF)/Q2} + R_{DS(OFF)/Q3}}{R_{DS(ON)/Q1} + R_{DS(OFF)/Q2} + R_{DS(OFF)/Q3}} V_{DD}$$

$$= \frac{10^7 K + 10^7 K}{200K + 10^7 K + 10^7 K} V_{DD} \quad \Rightarrow V_y \approx V_{DD} \Rightarrow y = 1.$$



Hình 3.30a. ($x_1=x_2=0$) *Hình 3.30b ($x_1=1, x_2=0$)* *Hình 3.30c ($x_1=x_2=1$)*

- Khi $x_1 = 1$, $x_2 = 0$ (hình 3.30b): Q_1, Q_2 dẫn và Q_3 tắt lúc đó theo sơ đồ tương đương ta có:

$$V_y = \frac{R_{DS(ON)/Q2} + R_{DS(OFF)/Q3}}{R_{DS(ON)/Q1} + R_{DS(ON)/Q2} + R_{DS(OFF)/Q3}} V_{DD} = \frac{1K + 10^7 K}{200K + 1K + 10^7 K} V_{DD}$$

$\Rightarrow V_y \approx V_{DD} \Rightarrow y = 1$

- Khi $x_1 = 0$, $x_2 = 1$: Q_1, Q_3 dẫn và Q_2 tắt, giải thích tương tự ta có $Vy \approx VDD \rightarrow y = 1$.
- Khi $x_1 = 1$, $x_2 = 1$ (hình 3.30c): Q_1, Q_2 và Q_3 đều dẫn, lúc đó theo sơ đồ tương đương ta có:

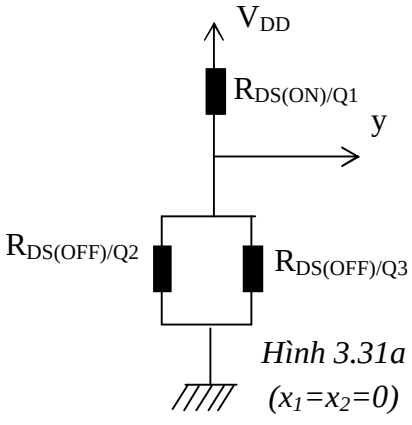
$$V_y = \frac{R_{DS(ON)/Q2} + R_{DS(ON)/Q3}}{R_{DS(ON)/Q1} + R_{DS(ON)/Q2} + R_{DS(ON)/Q3}} V_{DD} = \frac{1\ K + 1K}{200K + 1K + 1K} V_{DD}$$

$\Rightarrow V_y \approx 0{,}05V \Rightarrow y = 0.$

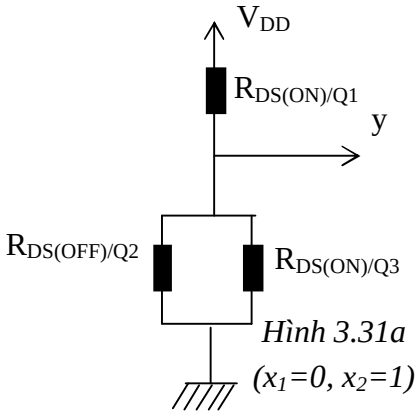
Vậy hình 3.28c là mạch thực hiện cổng NAND.

Hình 3.28b (cổng NOR)

Ta lần lượt xét các trường hợp sau: (sơ đồ tương đương hình 3.31)



Hình 3.31a
($x_1=x_2=0$)



Hình 3.31a
($x_1=0$, $x_2=1$)

- Khi $x_1 = x_2 = 0$ (hình 3.31a) : Q_1 dẫn, Q_2 và Q_3 đều tắt, lúc đó theo sơ đồ tương đương ta có:

$$V_y = \frac{(R_{DS(OFF)/Q2})//(R_{DS(OFF)/Q3})}{R_{DS(ON)/Q1}+[(R_{DS(OFF)/Q2})//(R_{DS(OFF)/Q3})]}V_{DD} = \frac{10^7K//10^7K}{200K+(10^7K//10^7K)}V_{DD}$$

$$\Rightarrow V_y \approx V_{DD} \Rightarrow y = 1$$

- Khi $x_1=0$, $x_2=1$ (hình 3.31b): Q_1 và Q_3 dẫn, Q_2 tắt, ta có:

$$V_y = \frac{(R_{DS(OFF)/Q2})//(R_{DS(ON)/Q3})}{R_{DS(ON)/Q1}+[(R_{DS(OFF)/Q2})//(R_{DS(ON)/Q3})]}V_{DD} = \frac{10^7K//1K}{200K+(10^7K//1K)}V_{DD}$$

$$\Rightarrow V_y \approx \frac{1}{201}V_{DD} \approx 0{,}005V \Rightarrow y = 0$$

- Khi $x_1=1$, $x_2=0$: Q_1 và Q_2 dẫn, Q_3 tắt, giải thích tương tự ta có:

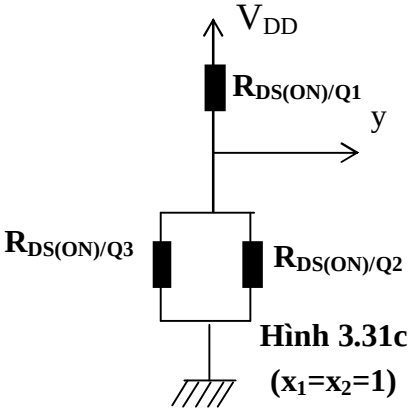
$$V_y \approx \frac{1}{201}V_{DD} \approx 0{,}005V \Rightarrow y = 0$$

- Khi $x_1=x_2=1$ (hình 3.31c): Q_1, Q_2, Q_3 đều dẫn, ta có:

$$V_y = \frac{(R_{DS(ON)/Q2})//(R_{DS(ON)/Q3})}{R_{DS(ON)/Q1}+[(R_{DS(ON)/Q2})//(R_{DS(ON)/Q3})]}V_{DD} = \frac{1K//1K}{200K+(1K//1K)}V_{DD}$$

$$\Rightarrow V_y \approx \frac{0{,}5}{200}V_{DD} \Rightarrow y = 0.$$

Vậy, sơ đồ mạch trên hình 3.28b chính là mạch thực hiện cổng NOR.



Hình 3.31c
($x_1=x_2=1$)

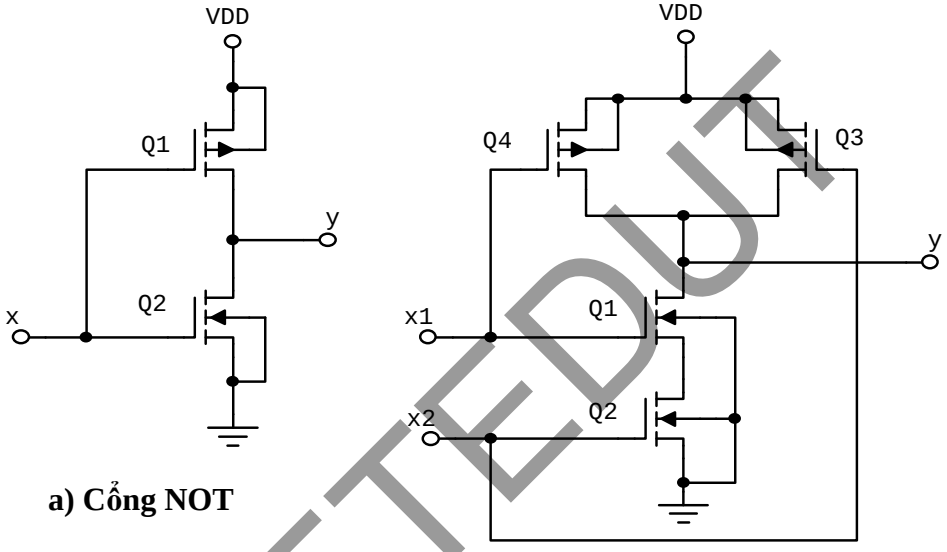
Các cổng logic họ CMOS (Complementation MOS)

Đây là loại cổng trong đó các transistor được sử dụng thuộc loại MOSFET và luôn có sự kết hợp giữa PMOS và NMOS, vì vậy mà người ta gọi là CMOS. Nhờ cấu trúc này mà vi mạch CMOS có những ưu điểm sau:

- Công suất tiêu thụ ở trạng thái tĩnh rất nhỏ.
- Tốc độ chuyển đổi trạng thái cao.
- Khả năng chống nhiễu tốt.
- Khả năng tải cao.

Trên hình 3.32 là các cổng logic họ CMOS, chúng ta sẽ lần lượt giải thích hoạt động của mỗi sơ đồ mạch.

Hình 3.32a (cổng NOT)



a) Cổng NOT **b) Cổng NAND**

Hình 3.32 Các cổng logic họ CMOS

Điều kiện để cổng PMOS dẫn : $V_S > V_D$, $V_G < V_B$
Điều kiện để cổng NMOS dẫn : $V_D > V_S$, $V_G > V_B$

- Khi x = 0 (hình 3.33a): Q_1 dẫn, Q_2 tắt, từ sơ đồ tương đương ta có:

$$V_y = \frac{R_{DS(OFF)/Q2}}{R_{DS(ON)/Q1} + R_{DS(OFF)/Q2}} V_{DD} = \frac{10^7 K}{1K + 10^7 K} V_{DD}$$

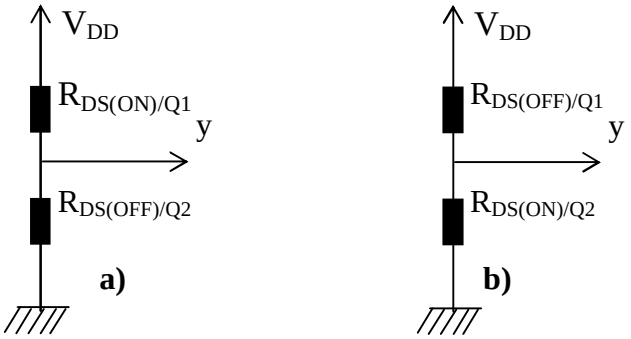
$\Rightarrow V_y \approx V_{DD} \Rightarrow y = 1$

- Khi x =1 (hình 3.33b): Q_1 tắt, Q_2 dẫn, ta có:

$$V_y = \frac{R_{DS(ON)/Q2}}{R_{DS(OFF)/Q1} + R_{DS(ON)/Q2}} V_{DD} = \frac{1K}{1K + 10^7 K} V_{DD} \Rightarrow V_y \approx \frac{1}{10^7} V_{DD}$$

vì rất nhỏ so với điện thế bão hòa của CMOS ở mức logic 0 $\rightarrow$ y = 0.

Vậy mạch ở hình 3.32a là mạch thực hiện cổng NOT theo công nghệ CMOS. Sơ đồ tương đương tương ứng với 2 trường hợp x=0 và x=1 được cho trên hình 3.33.



Hình 3.33.Sơ đồ tương đương: a.Khi x=0 b.Khi x=1

Hình 3.32b (cổng NAND)

Sơ đồ tương đương của mạch cổng NAND họ CMOS được cho trên hình 3.34.

- Khi $x_1=x_2=0$: Q_4 và Q_3 dẫn, Q_2 và Q_1 tắt, ta có:

$$V_y = \frac{(R_{DS(OFF)/Q2})//(R_{DS(OFF)/Q1})}{R_{DS(OFF)/Q1} + R_{DS(OFF)/Q2} + [(R_{DS(ON)/Q4})//(R_{DS(ON)/Q3})]} V_{DD} = \frac{10^7K//10^7K}{10^7K//10^7K + (1K//1K)} V_{DD}$$

$\Rightarrow V_y \approx V_{DD} \Rightarrow y = 1$

- Khi $x_1 = 0$, $x_2 = 1$: Q_2 và Q_3 dẫn, Q_1 và Q_4 tắt, ta có :

$$V_y = \frac{(R_{DS(OFF)/Q1})//(R_{DS(ON)/Q2})}{R_{DS(OFF)/Q1} + R_{DS(OFF)/Q2} + [(R_{DS(ON)/Q3})//(R_{DS(OF)/Q4})]} V_{DD} = \frac{10^7K + 1K}{10^7K + 1K + (10^7K//1K)} V_{DD}$$

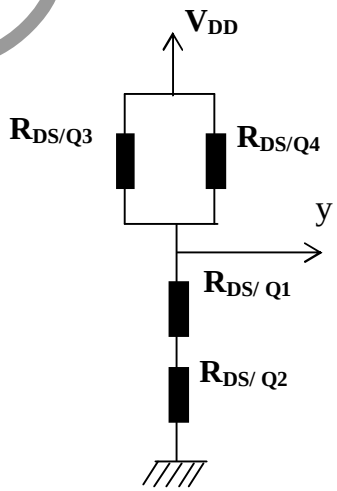
$\Rightarrow V_y \approx V_{DD} \Rightarrow y = 1$

- Khi $x_1= 1$, $x_2 = 0$: Q_3 và Q_2 dẫn, Q_1 và Q_4 tắt: $V_y \approx V_{DD} \Rightarrow y = 1$

- Khi $x_1 = x_2 = 1$: Q_2 và Q_1 dẫn, Q_3 và Q_4 tắt, ta có:

$$V_y = \frac{(R_{DS(ON)/Q1})//(R_{DS(ON)/Q2})}{R_{DS(ON)/Q1} + R_{DS(ON)/Q2} + [(R_{DS(OFF)/Q4})//(R_{DS(OFF)/Q3})]} V_{DD} = \frac{1K + 1K}{1K + 1K + (10^7K//10^7K)} V_{DD}$$

$\Rightarrow V_y \approx 0V \Rightarrow y = 0 \Rightarrow$ Đây chính là mạch thực hiện cổng NAND.

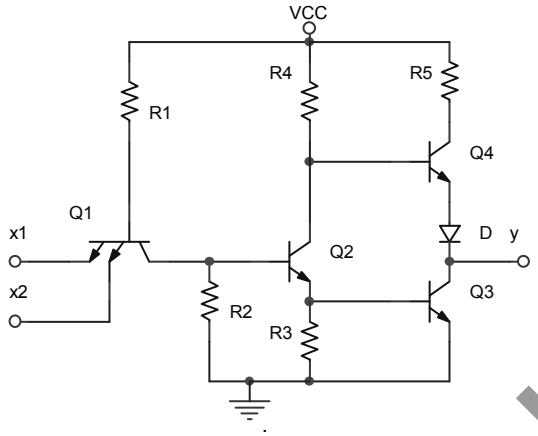


Hình 3.34.

3. Phân loại cổng logic theo ngõ ra

a. Ngõ ra cột cham (Totem Pole Output)

Xét cổng logic họ TTL với sơ đồ mạch như hình 3.35.



Hình 3.35. Ngõ ra cột chạm

- Khi $x_1=x_2=1$: Tiếp giáp BE_1, BE_2 của Q_1 phân cực ngược nên Q_1 tắt. Điện thế tại cực nền của Q_1 làm cho tiếp giáp BC/Q_1 mở, có dòng điện chảy qua tiếp giáp BC/Q_1 đổ vào cực nền của Q_2, Q_2 được phân cực thuận nên dẫn bão hòa. Do Q_2 dẫn bão hòa dẫn tới Q_3 dẫn bão hòa.

Khi Q2 dẫn bão hòa thì điện thế tại cực C/Q2

$$V_{C/Q2}= V_{B/Q4} = V_{ces/Q2} + V_{bes/Q3} = 0,2 + 0,8 = 1V$$

Mà điều kiện cần cho Q4 dẫn là:

$$V_{C/Q2}=V_{B/Q4} = V_{be/Q4} + V_{\gamma/D} + V_{ces/Q3} = 0,6 + 0,8 + 0,2= 1,6V$$

Ta thấy điều kiện này không thỏa mãn khi Q2 dẫn bão hòa, do đó khi Q2 dẫn bão hòa → Q4 tắt → cắt nguồn V_{CC} ra khỏi mạch. Lúc này ta nói rằng cổng sẽ hút dòng vào và dòng từ ngoài qua tải đổ vào ngõ ra của cổng đi qua Q_3, người ta nói Q_3 là nơi nhận dòng và dòng đổ vào Q_3 gọi là dòng ngõ ra mức thấp, ký hiệu I_{OL}.

Về mặt thiết kế mạch: ta thấy rằng dòng tải It cũng chính là dòng ngõ ra mức thấp I_{OL} và là dòng đổ từ ngoài vào qua Q_3, dòng này phải nằm trong giới hạn chịu đựng dòng của Q_3 để Q_3 không bị đánh thủng thì mạch sẽ làm việc bình thường.

Dòng I_{OL} thay đổi tùy thuộc vào công nghệ chế tạo:

+ TTL : dòng ngõ ra mức thấp I_{OL} lớn nhất 16mA.
+ TTL/LS : dòng ngõ ra mức thấp I_{OL} lớn nhất 8mA.

Đây là những thông số rất quan trọng cần chú ý trong quá trình thiết kế mạch số họ TTL để đảm bảo độ an toàn và ổn định của mạch.

- Các trường hợp còn lại ($x_1=0,x_2=1$; $x_1=1,x_2=0$; $x_1=x_2=0$): Lúc này Q_2 và Q_3 tắt còn Q_4 dẫn → y = 1. Ta nói cổng cấp dòng ra, dòng này đổ từ nguồn qua Q_4 và diode D xuống cung cấp cho tải, người ta gọi là dòng ngõ ra mức cao, ký hiệu I_{OH}.

Điện áp ngõ ra V_Y được tính phụ thuộc vào dòng tải I_{OH}:

$$V_Y = V_{logic1} = V_{cc}- I_{OH}R_5 - V_{ces/\ Q4} - V_{\gamma/D}$$

Thông thường khi có tải $V_{logic1\ max}$ = (3,4V → 3,6V)

I_{OH} cũng chính là dòng qua tải It, nếu I_{OH} càng tăng thì V_{logic1} càng giảm và ngược lại. Song V_{logic1} chỉ được phép giảm đến một giá trị cho phép $V_{logic1\ min}$ = 2,2V.

Về mặt thiết kế mạch: ta chọn $V_{logic1\ min}$ = 2,4V để bảo đảm cổng cấp dòng ra khi ở mức logic 1 không được nhỏ hơn $V_{logic1\ min}$ và đảm bảo cổng hút dòng vào khi ở mức logic 0 thì dòng tải ở mức logic 0 không được lớn hơn dòng I_{OL}.

Nhược điểm của ngõ ra cột chạm: ***Không cho phép nối chung các ngõ ra lại với nhau có thể làm hỏng cổng.***

b. Ngõ ra cực thu để hở (Open Collector Output)

Về phương diện cấu tạo gần giống với ngõ ra cột chạm nhưng khác với ngõ ra cột chạm là không có Q_4, diode D, R_5 và lúc này cực thu (cực C) của Q_3 để hở.

Do đó để cổng làm việc trong thực tế ta nối ngõ ra của cổng (cực C của Q_3) lên nguồn V'_{CC} bằng phần tử thụ động R. Nguồn V'_{CC} có thể cùng giá trị với V_{CC} hoặc khác tùy thuộc vào mục đích thiết kế.

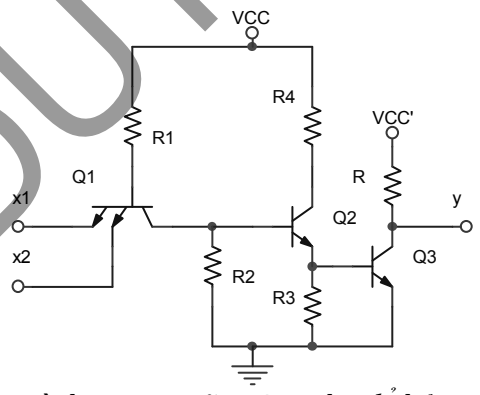
Chúng ta lần lượt phân tích các trường hợp hoạt động của mạch:

- Khi $x_1=x_2=1$: Tiếp giáp BE_1, BE_2 phân cực ngược, điện thế tại cực nền của Q_1 làm cho tiếp giáp BC/Q_1 mở nên Q_2 dẫn bão hòa, Q_2 dẫn bão hòa kéo theo Q_3 dẫn bão hòa → y = 0, do đó điện áp tại ngõ ra y:

  $V_Y = V_{logic0} = V_{C/Q3} = V_{ces/Q3}$

  $= 0,2V \approx 0V$

  Lúc này cổng sẽ hút dòng vào và Q_3 là nơi nhận dòng, ta gọi là **dòng ngõ ra mức thấp I_{OL}.**



Hình 3.36. Ngõ ra cực thu để hở

- Các trường hợp còn lại ($x_1=0,x_2=1$; $x_1=1,x_2=0$; $x_1=x_2=0$): Có ít nhất một tiếp giáp BE/Q_1 mở, ghim điện thế tại cực nền Q_1 làm cho tiếp giáp BC/Q_1, Q_2, Q_3 đều tắt, lúc này cổng cấp dòng ra đổ từ nguồn V'_{CC} qua điện trở R cấp cho tải ở mạch ngoài → y=1, người ta gọi là **dòng ngõ ra mức cao I_{OH}.**

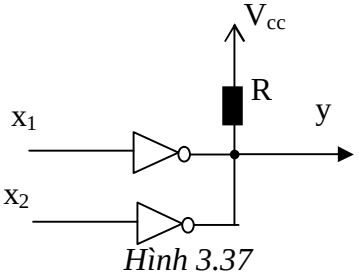
  Ta có:

  $$V_Y = V_{logic1} = V'_{CC} - I_{OH}.R$$

Ưu điểm của ngõ ra có cực thu để hở:

- Cho phép nối chung các ngõ ra lại với nhau.
- Trong một vài trường hợp khi nối chung các ngõ ra lại với nhau có thể tạo thành cổng logic khác.

Ví dụ: Mạch ở hình 3.37 sử dụng các cổng NOT có ngõ ra cực thu để hở, khi nối chung các ngõ ra lại với nhau có thể tạo thành cổng NOR. *(Hãy giải thích hoạt động của mạch này?)*



Hình 3.37

c. Ngõ ra ba trạng thái (Three States Output)

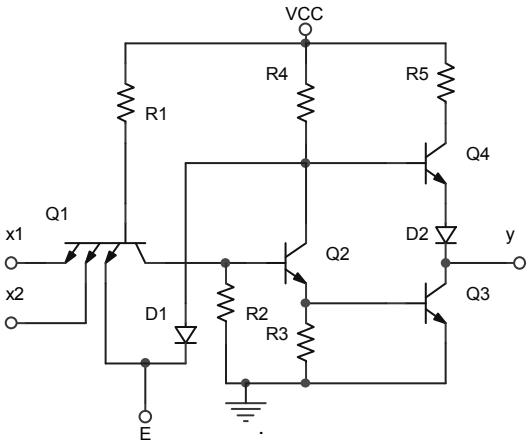
Về mặt cấu trúc và cấu tạo hoàn toàn giống ngõ ra cột chạm, tuy nhiên có thêm ngõ vào thứ 3 cho phép mạch hoạt động kí hiệu là E (Enable).

- **E=1:** diode D_1 tắt, mạch làm việc hoàn toàn giống cổng NAND ngõ ra cột chạm. Lúc đó mạch tồn tại một trạng thái y = 0 hoặc y = 1 tùy thuộc vào các trạng thái logic của 2 ngõ vào x_1, x_2.

- **E=0:** diode tiếp giáp BE_3 mở, ghim áp trên cực nền của Q_1 làm cho tiếp giáp BC/Q_1 tắt và Q_2, Q_3 cũng tắt. Lúc này diode D_1 dẫn ghim điện thế ở cực C của Q_2:

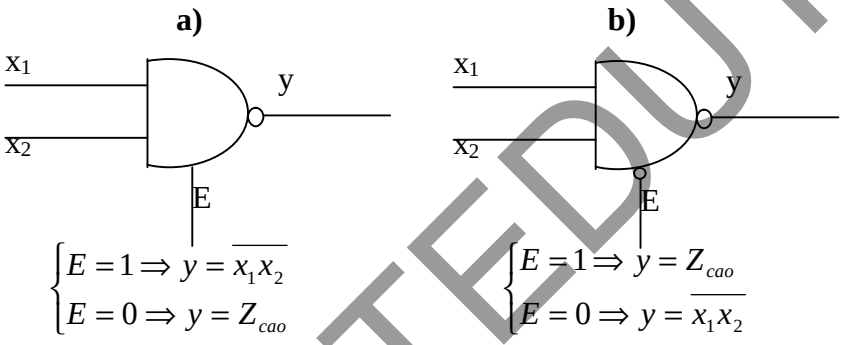
$$V_{C/Q2} = V_{B/Q4} = V\gamma/_{D1} = 0{,}7V \Rightarrow Q_4 \text{ tắt.}$$

Nên cổng không cấp dòng ra và cũng không hút dòng vào. Lúc này, ngõ ra y chỉ nối với cổng về phương diện vật lý nhưng lại cách ly về phương diện điện, tương đương với trạng thái trở kháng cao. Chính vì vậy mà người ta gọi là trạng thái thứ ba là trạng thái tổng trở cao.



Hình 3.38. Ngõ ra 3 trạng thái

Trong trường hợp này ngõ vào cho phép E tích cực mức cao (mức logic 1). Thực tế các cổng logic với ngõ ra 3 trạng thái có thể có ngõ vào điều khiển E tích cực mức cao (mức 1) hoặc tích cực mức thấp (mức 0). Chẳng hạn một cổng NAND với ngõ ra 3 trạng thái có thể được ký hiệu như trên hình vẽ 3.39.

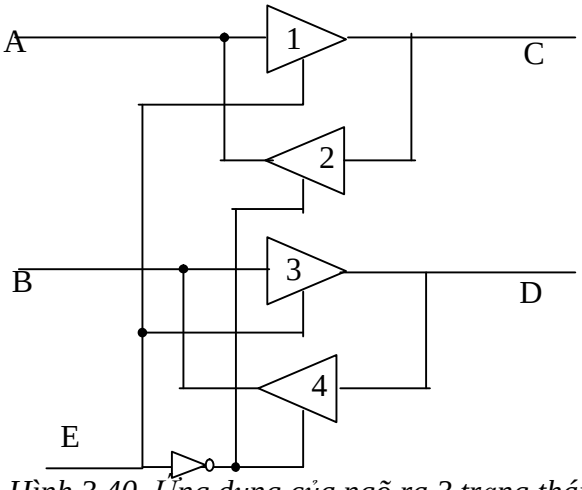


Hình 3.39. Cổng NAND 3 trạng thái với ngõ vào E
a. E tích cực mức cao - b. E tích cực mức thấp

<u>Ứng dụng của ngõ ra 3 trạng thái</u>:

- Sử dụng ngõ ra ba trạng thái để chế tạo ra cổng đệm 2 chiều.
- Chế tạo các chíp nhớ của bộ vi xử lý.

Một ứng dụng của ngõ ra ba trạng thái trong mạch xuất/nhập dữ liệu 2 chiều có thể cho trên sơ đồ 3.40. *Hãy thử giải thích sơ đồ này ?*



Hình 3.40. Ứng dụng của ngõ ra 3 trạng thái

- E=1: Cổng đệm 1 và 3 mở, 2 và 4 treo lên tổng trở cao: dữ liệu đi từ A→C, B→D. Vậy dữ liệu được xuất ra.
- E=0: Cổng đệm 2 và 4 mở, 1 và 3 treo lên tổng trở cao: dữ liệu đi từ C→A, D→B. Vậy dữ liệu được nhập vào.

3.2.3. Các thông số kỹ thuật của cổng logic

1. Công suất tiêu tán P_{tt}

Một phần tử logic khi làm việc phải trải qua các giai đoạn sau:

- Ở trạng thái tắt.
- Chuyển từ trạng thái tắt sang trạng thái dẫn.
- Ở trạng thái dẫn.
- Chuyển từ trạng thái dẫn sang tắt.

Ở mỗi giai đoạn, phần tử logic đều tiêu thụ ở nguồn một công suất.

Đối với các phần tử logic họ TTL: các phần tử TTL tiêu thụ công suất của nguồn chủ yếu khi ở trạng thái tĩnh (đang dẫn hoặc đang tắt).

- Nếu gọi P^0 là công suất tiêu thụ ứng với ngõ ra của phần tử logic tồn tại ở mức logic 0.
- Nếu gọi P^1 là công suất tiêu thụ ứng với ngõ ra của phần tử logic tồn tại ở mức logic 1.
- Gọi P là công suất tiêu tán trung bình thì:

$$P = \frac{P^0 + P^1}{2}$$

Đối với cả vi mạch (IC – **I**ntegrated **C**ircuit) người ta tính như sau:

- Gọi I_{CL} dòng do nguồn cung cấp khi ngõ ra ở mức logic 0.
- Gọi I_{CH} dòng do nguồn cung cấp khi ngõ ra ở mức logic 1.
- Gọi I_C là dòng trung bình thì :

$$I_C = \frac{I_{CL} + I_{CH}}{2}$$

- Thì công suất tiêu tán cho cả vi mạch được tính:

$$\mathbf{P_{tt} = I_C . V_{CC}}$$

Đối với vi mạch họ CMOS: chỉ tiêu thụ công suất chủ yếu trong trạng thái động (trong thời gian chuyển mạch). Công suất tiêu tán:

$$P_{tt} = C_L . f . V_{DD}^2$$

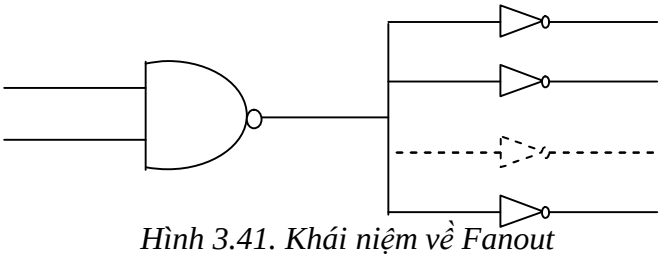
Trong đó: C_L là điện dung của tải (điện dung tải)

Như vậy ta thấy đối với vi mạch CMOS tần số hoạt động (tần số chuyển mạch) càng lớn công suất tiêu tán càng tăng.

2. Fanout (Hệ số mắc mạch ngõ ra)

Fanout là hệ số mắc mạch ở ngõ ra hay còn gọi là khả năng tải của một phần tử logic.

Gọi N là Fanout của một phần tử logic, thì nó được định nghĩa như sau: Số ngõ vào logic cực đại được nối đến một ngõ ra của phần tử logic cùng họ mà mạch vẫn hoạt động bình thường (hình 3.41).



Hình 3.41. Khái niệm về Fanout

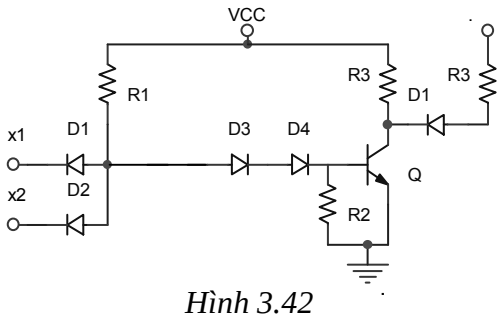
Xét ví dụ đối với họ DTL: (Hình 3.42)

- y=1: mạch hoạt động bình thường.
- y=0: BJT dẫn bão hòa, dòng bão hòa gồm hai thành phần:

$$I_{C S} = I_{R3} + N\, I_1$$

(với N là số phần tử tải mắc ở ngõ ra)

Mặt khác: $I_B=I_{R1}-I_{R2}$= const, mà I_{cs} tăng lên do có dòng ghép đổ vào → điều kiện dẫn bão hòa không thỏa mãn → BJT ra khỏi chế độ dẫn bão hòa và đi vào chế độ khuếch đại, lúc đó V_Y tăng lên nên ngõ ra không còn đảm bảo ở mức logic 0 nữa. Vậy, điều kiện để mạch hoạt động bình thường là:



Hình 3.42

$$I_{R3} + N\, I_1 < \beta_{min}\, I_B \;\Rightarrow N < \frac{\beta_{min} I_B - I_{R3}}{I_1} \quad (*)$$

N: số lớn nhất thỏa mãn điều kiện (*) được gọi là Fanout của phần tử logic DTL.

3. Fanin (Hệ số mắc mạch ngõ vào)

Gọi M là Fanin của 1 phần tử logic thì M được định nghĩa như sau: Đó chính là "số ngõ vào logic cực đại của một phần tử logic".

Đối với các phần tử logic thực hiện chức năng cộng logic, thì số lượng M lớn nhất là 4 ngõ vào. Đối với các phần tử logic thực hiện chức năng nhân logic, thì số lượng M lớn nhất là 6 ngõ vào.

Đối với họ logic CMOS thì có M nhiều hơn nhưng cũng không quá 8 ngõ vào.

4. Độ chống nhiễu

Độ ổn định nhiễu là tiêu chuẩn đánh giá độ nhạy của mạch logic đối với tạp âm xung trên đầu vào. Độ ổn định nhiễu (tĩnh) là giá trị điện áp nhiễu tối đa trên đầu vào không làm thay đổi trạng thái logic của mạch, còn gọi là mức ổn định nhiễu.

5. Trễ truyền đạt

Trễ truyền đạt là khoảng thời gian để đầu ra của mạch có đáp ứng đối với sự thay đổi mức logic của đầu vào.

Trễ truyền đạt là tiêu chuẩn để đánh giá tốc độ làm việc của mạch. Tốc độ làm việc của mạch tương ứng với tần số mà mạch vẫn còn hoạt động đúng. Như vậy, trễ truyền đạt càng nhỏ càng tốt hay tốc độ làm việc càng lớn càng tốt.

Đối với hầu hết các vi mạch số hiện nay, trễ truyền đạt là rất nhỏ, cỡ vài nano giây (ns). Một vài loại mạch logic có thời gian trễ lớn cỡ vài trăm nano giây.

Khi mắc liên tiếp nhiều mạch logic thì trễ truyền đạt của toàn mạch sẽ bằng tổng các trễ truyền đạt của mỗi tầng.

3.3. FLIP – FLOP (FF)

3.3.1. Khái niệm

Flip-Flop (viết tắt là FF) là mạch dao động đa hài hai trạng thái bền, được xây dựng trên cơ sở các cổng logic và hoạt động theo một bảng trạng thái cho trước.

3.3.2. Phân loại

Có hai cách phân loại:

- Phân loại theo tín hiệu điều khiển.
- Phân loại theo chức năng.

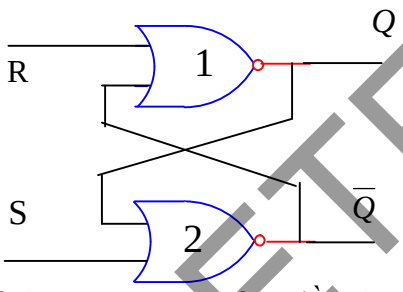
1. Phân loại FF theo tín hiệu điều khiển đồng bộ

Gồm có hai loại:

- Không có tín hiệu điều khiển đồng bộ (FF không đồng bộ).
- Có tín hiệu điều khiển đồng bộ (FF đồng bộ).

a. FF không đồng bộ

Dạng 1: RSFF không đồng bộ dùng cổng NOR (sơ đồ hình 3.43)



S	R	Q
0	0	Q^0
0	1	0
1	0	1
1	1	X

Hình 3.43. RSFF không đồng bộ sử dụng cổng NOR và bảng trạng thái

Dựa vào bảng chân trị của cổng NOR để giải thích hoạt động của sơ đồ mạch này:

- S = 0, R = 1 $\Rightarrow$ Q = 0. Q=0 hồi tiếp về cổng NOR 2 nên cổng NOR 2 có hai ngõ vào bằng 0 $\Rightarrow \overline{Q} = 1$. Vậy, Q = 0 và $\overline{Q} = 1$.
- S = 1, R = 0 $\Rightarrow$ $\overline{Q} = 0$. $\overline{Q} = 0$ hồi tiếp về cổng NOR 1 nên cổng NOR 1 có hai ngõ vào bằng 0 $\Rightarrow$ Q = 1. Vậy, Q = 1 và $\overline{Q} = 0$.
- Giả sử ban đầu: S = 0, R = 1 $\Rightarrow$ Q = 0 và $\overline{Q} = 1$.

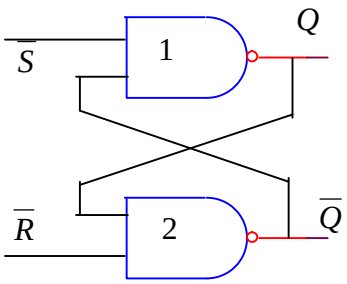
  Nếu tín hiệu ngõ vào thay đổi thành: S = 0, R = 0 (R chuyển từ 1 $\rightarrow$ 0) ta có:

  + S = 0 và Q = 0 $\Rightarrow \overline{Q} = 1$
  + R = 0 và $\overline{Q}$ = 1 $\Rightarrow$ Q = 0 $\Rightarrow$ RSFF giữ nguyên trạng thái cũ trước đó.
- Giả sử ban đầu: S = 1, R = 0 $\Rightarrow$ Q = 1 và $\overline{Q} = 0$.

  Nếu tín hiệu ngõ vào thay đổi thành: R = 0, S = 0 (S chuyển từ 1 $\rightarrow$ 0) ta có:

  + R = 0 và $\overline{Q} = 0 \Rightarrow$ Q = 1
  + S = 0 và Q = 1 $\Rightarrow \overline{Q} = 0$ $\Rightarrow$ RSFF giữ nguyên trạng thái cũ trước đó.

Dạng 2: RSFF không đồng bộ dùng cổng NAND (sơ đồ hình 3.44)



$\overline{S}$	$\overline{R}$	Q
0	0	X
0	1	1
1	0	0
1	1	Q^0

Hình 3.44. RSFF không đồng bộ sử dụng cổng NAND và bảng trạng thái

Dựa vào bảng chân trị của cổng NAND:

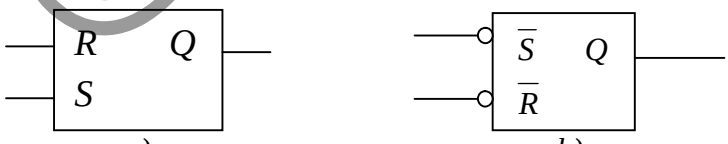
$$y = \begin{cases} 0 & \forall x_i = 1 \\ 1 & \exists x_i = 0 \end{cases}$$

Ta có:

- $\overline{S}$ = 0, $\overline{R}$ = 1 $\Rightarrow$ Q = 1. Q = 1 hồi tiếp về cổng NAND 2 nên cổng NAND 2 có hai ngõ vào bằng 1 vậy $\overline{Q}$ = 0.
- $\overline{S}$ = 0, $\overline{R}$ = 1 $\Rightarrow$ $\overline{Q}$ = 1. $\overline{Q}$ = 1 hồi tiếp về cổng NAND 1 nên cổng NAND 1 có hai ngõ vào bằng 1 vậy Q = 0.
- $\overline{S}$ = $\overline{R}$ = 0 $\Rightarrow$ $\overline{Q}$ = Q = 1 đây là trạng thái cấm.
- $\overline{S}$ = $\overline{R}$ = 1: Giả sử trạng thái trước đó có Q = 1, $\overline{Q}$ = 0 $\Rightarrow$ hồi tiếp về cổng NAND 1 nên cổng NAND 1 có một ngõ vào bằng 0 vậy Q = 1 $\Rightarrow$ RSFF giữ nguyên trạng thái cũ.

Như vậy gọi là FF không đồng bộ bởi vì chỉ cần một trong hai ngõ vào S hay R thay đổi thì ngõ ra cũng thay đổi theo.

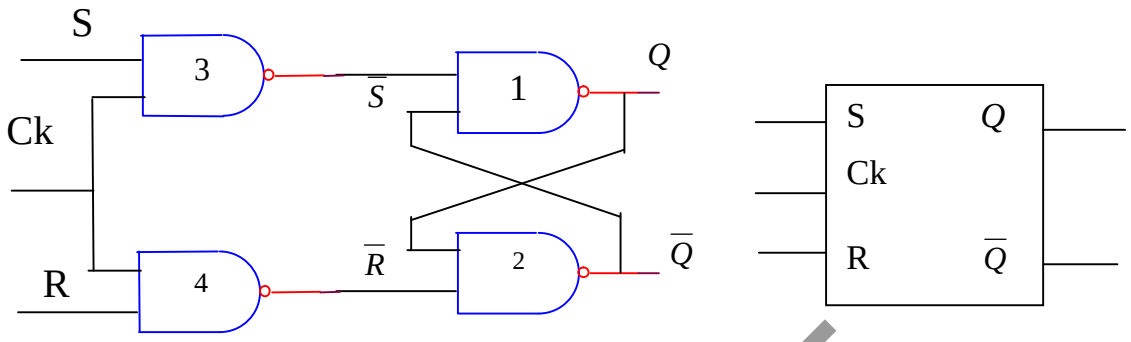
Về mặt kí hiệu, các RSFF không đồng bộ được ký hiệu như sau:



Hình 3.45. Ký hiệu các FF không đồng bộ
a. R,S tác động mức 1 - b. R,S tác động mức 0

b. FF đồng bộ

Xét sơ đồ RSFF đồng bộ với sơ đồ mạch, ký hiệu và bảng trạng thái hoạt động như hình 3.46. Trong đó: Ck là tín hiệu điều khiển đồng bộ hay tín hiệu đồng hồ (Clock). Khảo sát hoạt động của mạch:



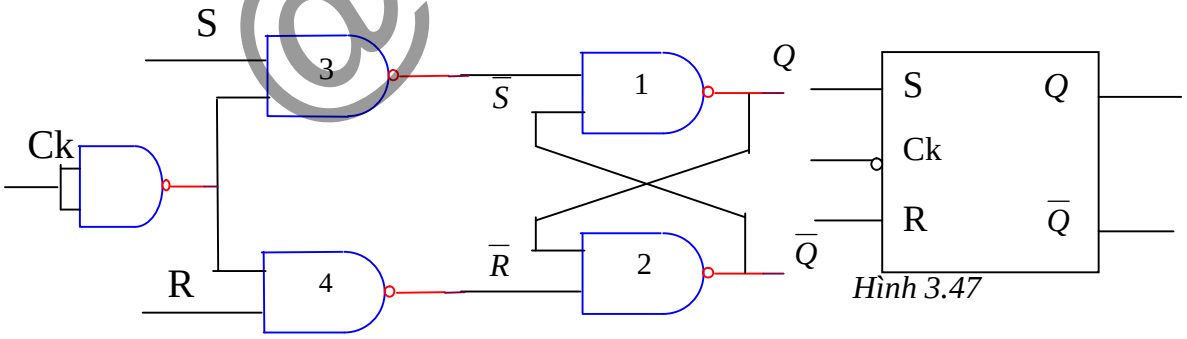
Hình 3.46. RSFF đồng bộ: Sơ đồ logic và ký hiệu

- Ck = 0: cổng NAND 3 và 4 khóa không cho dữ liệu đưa vào. Vì cổng NAND 3 và 4 đều có ít nhất một ngõ vào Ck = 0 $\Rightarrow \overline{S} = \overline{R} = 1 \Rightarrow Q = Q^0$: RSFF giữ nguyên trạng thái cũ.
- Ck = 1: cổng NAND 3 và 4 mở. Ngõ ra Q sẽ thay đổi tùy thuộc vào trạng thái của S và R.

  + S = 0, R = 0 $\Rightarrow \overline{S} = 1, \overline{R} = 1 \Rightarrow Q = Q^0$
  + S = 0, R = 1 $\Rightarrow \overline{S} = 1, \overline{R} = 0 \Rightarrow Q = 0$
  + S = 1, R = 0 $\Rightarrow \overline{S} = 0, \overline{R} = 1 \Rightarrow Q = 1$
  + S = 1, R = 1 $\Rightarrow \overline{S} = 0, \overline{R} = 0 \Rightarrow Q = X$

S	**R**	**Ck**	**Q**
X	X	0	Q^0
0	0	1	Q^0
0	1	1	0
1	0	1	1
1	1	1	X

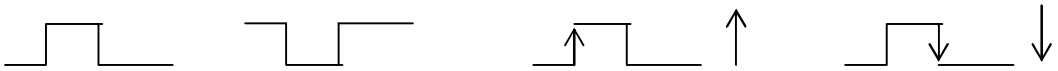
Trong trường hợp này tín hiệu đồng bộ Ck tác động mức 1. Trong trường hợp Ck tác động mức 0 thì ta mắc thêm cổng đảo như sau (hình 3.47):



Hình 3.47

Tùy thuộc vào mức tích cực của tín hiệu đồng bộ Ck, chúng ta có các loại tín hiệu điều khiển:

- Ck điều khiển theo mức 1.
- Ck điều khiển theo mức 0.
- Ck điều khiển theo sườn lên (sườn trước).
- Ck điều khiển theo sườn xuống (sườn sau).



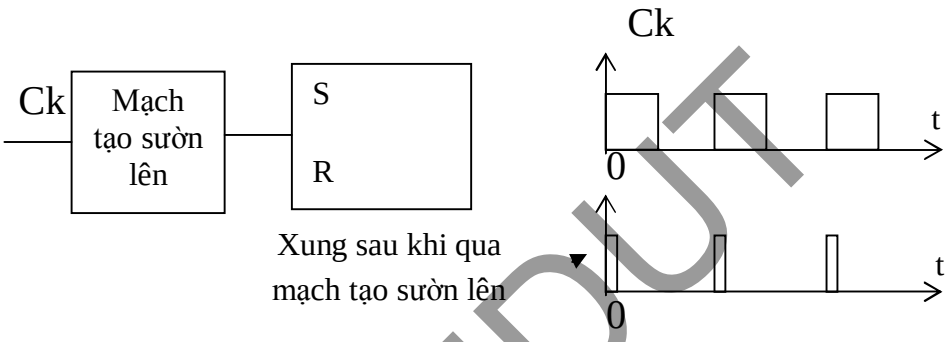
a. Mức 1 *b. Mức 0* *c. Sườn lên* *d. Sườn xuống*

Hình 3.48. Các loại tín hiệu điều khiển Ck khác nhau

<u>Xét FF có Ck điều khiển theo sườn lên (sườn trước)</u>:

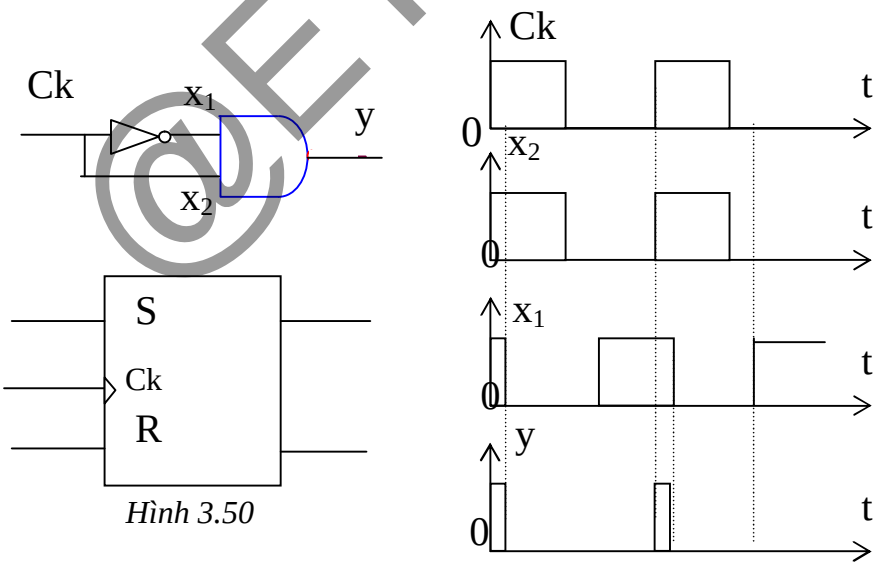
Sườn lên và mức logic 1 có mối quan hệ với nhau, vì vậy mạch tạo sườn lên là mạch cải tiến của mạch tác động theo mức logic 1.

Sườn lên thực chất là một xung dương có thời gian tồn tại rất ngắn. Để cải tiến các FF tác động theo mức logic 1 thành FF tác động theo sườn lên ta mắc vào trước FF đó một mạch tạo sườn lên như hình 3.49.



Hình 3.49. Sơ đồ khối FF tác động theo sườn lên và dạng sóng

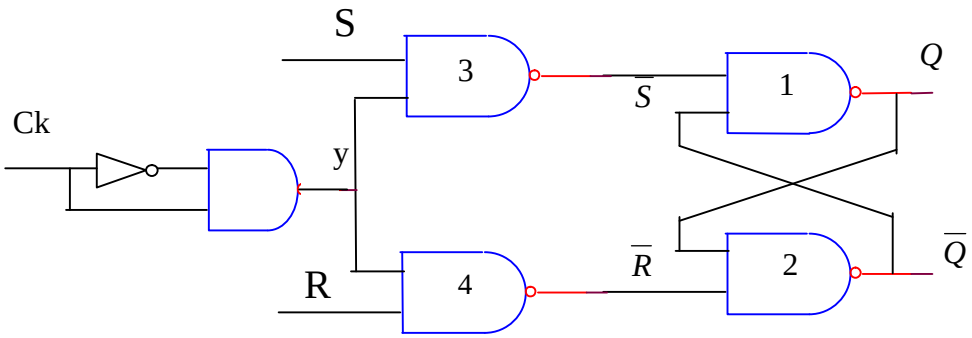
Ở mạch tạo sườn người ta lợi dụng thời gian trễ của tín hiệu khi đi qua phần tử logic. Đối với mạch tạo sườn người ta lợi dụng thời gian trễ của tín hiệu khi đi qua cổng NOT.



Hình 3.50

Xét sơ đồ mạch tạo sườn lên và dạng sóng như hình 3.50 : Mạch tạo sườn lên gồm một cổng AND 2 ngõ vào và một cổng NOT. Tín hiệu x1 từ cổng NOT được đưa đến cổng AND cùng với tín hiệu x_2 đi trực tiếp (x_2 = Ck). Do tính chất trễ của tín hiệu Ck khi đi qua cổng NOT nên x_1 bị trễ một khoảng thời gian, vì vậy tín hiệu ngõ ra của cổng AND có dạng một xung dương rất hẹp với thời gian tồn tại chính bằng thời gian trễ (trễ truyền đạt) của cổng NOT. Xung dương hẹp này được đưa đến ngõ vào đồng bộ của FF điều khiển theo mức logic 1. Tại các thời điểm có sườn lên của tín hiệu xung nhịp Ck sẽ xuất hiện một xung dương tác động vào ngõ vào đồng bộ của FF điều khiển ngõ ra

Q thay đổi trạng thái theo các ngõ vào. Sơ đồ mạch FF có tín hiệu Ck điều khiển theo sườn lên như

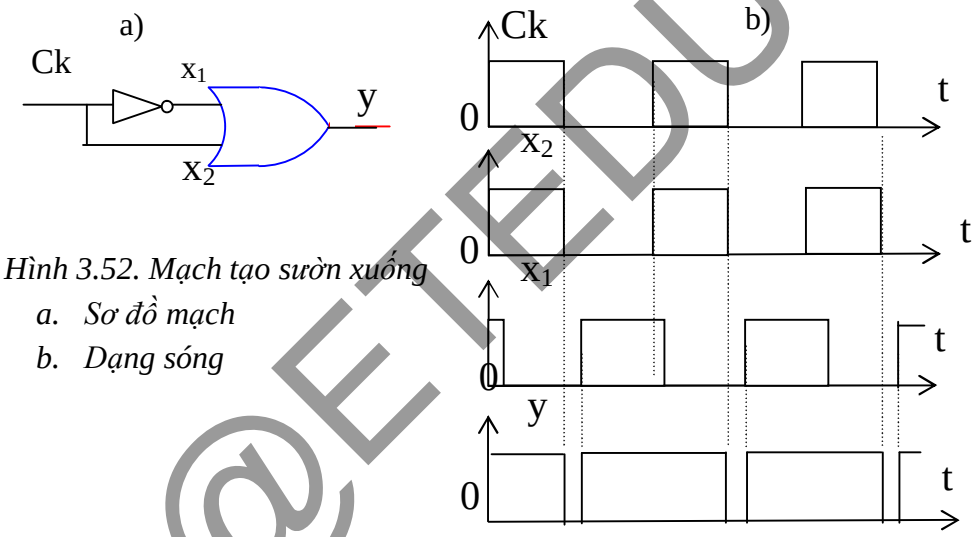


Hình 3.51. FF có tín hiệu Ck điều khiển theo sườn lên

hình 3.51.

<u>Xét FF có Ck điều khiển theo sườn xuống (sườn sau)</u>:

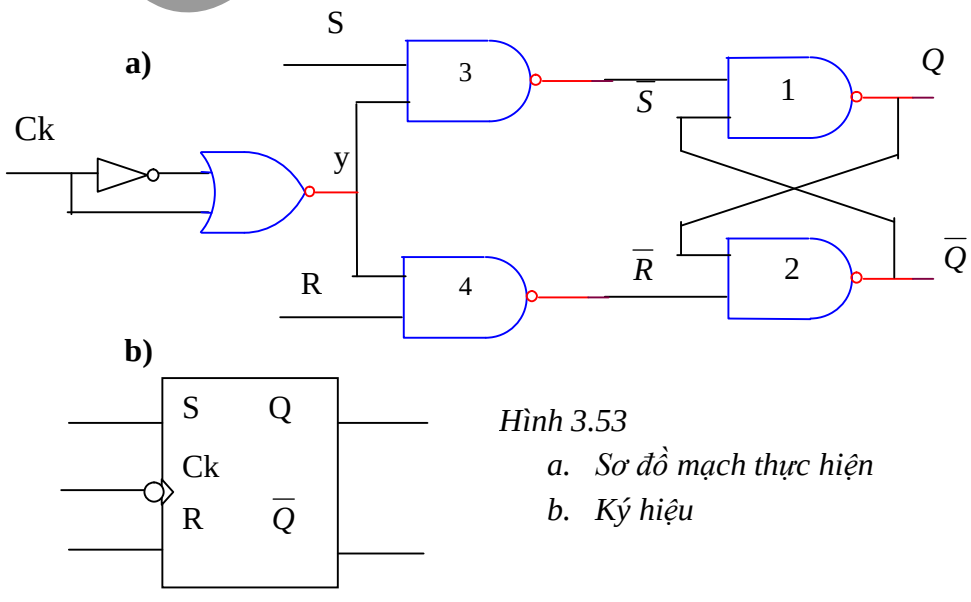
Mạch tạo sườn xuống là mạch cải tiến tác động mức logic 0. Sơ đồ mạch và dạng sóng được cho ở hình 3.52. Trên hình 3.53 là ký hiệu trên sơ đồ mạch và sơ đồ thực hiện Flip-Flop tác động theo sườn xuống.



Hình 3.52. Mạch tạo sườn xuống

a. Sơ đồ mạch

b. Dạng sóng



Hình 3.53

a. Sơ đồ mạch thực hiện

b. Ký hiệu

(Sinh viên tự giải thích hoạt động của các mạch này).

Ý nghĩa của tín hiệu đồng bộ Ck:

Đối với các FF đồng bộ, các ngõ ra chỉ thay đổi trạng thái theo ngõ vào DATA khi xung Ck tồn tại mức 1 (đối với FF tác động mức 1), hoặc xung Ck tồn tại mức 0 (đối với FF tác động mức 0), hoặc xung Ck ở sườn lên (đối với FF tác động sườn lên), xung Ck ở sườn xuống (đối với FF tác động sườn xuống), còn tất cả các trường hợp khác của Ck thì ngõ ra không thay đổi trạng thái theo các ngõ vào mặc dù lúc đó các ngõ vào có thay đổi trạng thái.

Phương pháp điều khiển theo kiểu chủ tớ (Master - Slaver):

Đối với phương pháp này khi xung Ck tồn tại mức logic 1 dữ liệu sẽ được nhập vào FF, còn khi Ck tồn tại mức logic 0 thì dữ liệu chứa trong FF được xuất ra ngoài.

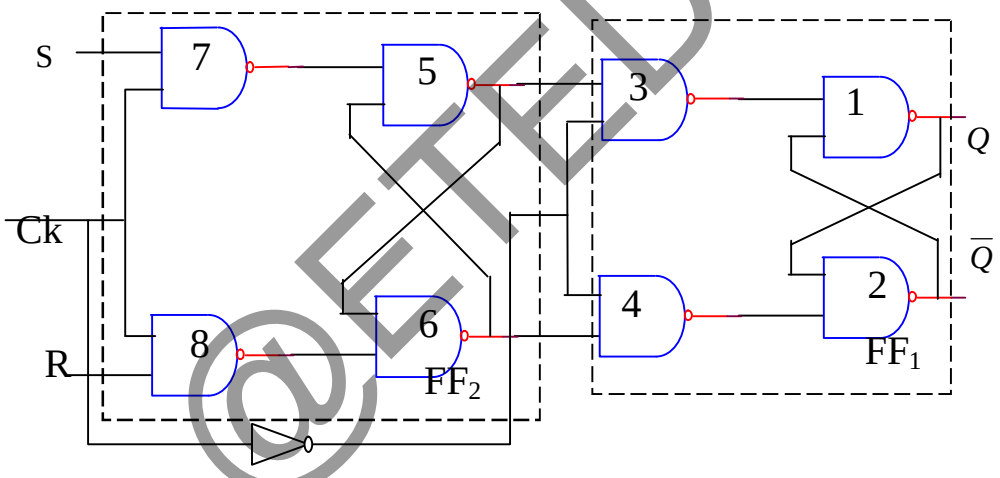
Về mặt cấu tạo bên trong gồm 2 FF: một FF thực hiện chức năng chủ (Master) và một FF thực hiện chức năng tớ (Slaver).

Hoạt động của FF điều khiển theo kiểu chủ/tớ: (hình 3.54)

+ Ck = 1: FF2 mở, dữ liệu được nhập vào FF2. Qua cổng đảo Ck = 0 (FF1 khóa nên giữ nguyên trạng thái cũ trước đó.

+ Ck = 0: FF2 khóa nên giữ nguyên trạng thái cũ trước đó. Qua cổng đảo Ck = 1 (FF1 mở, dữ liệu được xuất ra ngoài.

Chú ý: Tín hiệu Ck có thể được tạo ra từ mạch dao động đa hài không trạng thái bền.

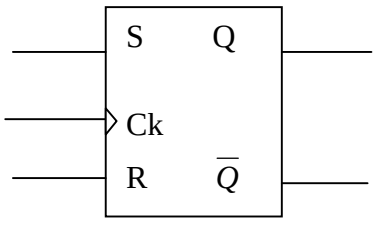


Hình 3.54. Phương pháp điều khiển theo kiểu chủ tớ

3.3.2.2. Phân loại FF theo chức năng

a. RSFF

Đó là FF có các ngõ vào và ngõ ra ký hiệu như hình vẽ.



Hình 3.55. Ký hiệu RSFF

Trong đó:

- S, R : các ngõ vào dữ liệu.
- *Q, $\overline{Q}$: các ngõ ra.*
- Ck : tín hiệu xung đồng bộ

Gọi S^n và R^n là trạng thái ngõ vào Data ở xung Ck thứ n.

Q^n , Q^{n+1} là trạng thái của ngõ ra Q ở xung Ck thứ n và thứ (n+1).

Lúc đó ta có bảng trạng thái mô tả hoạt động của RSFF:

S^n	R^n	Q^{n+1}
0	0	Q^n
0	1	0
1	0	1
1	1	X

Lưu ý rằng trạng thái khi cả 2 ngõ vào S = R = 1 lúc đó cả 2 ngõ ra có cùng mức logic, đây là trạng thái cấm của RSFF (thường được ký hiệu X).

Tiếp theo chúng ta sẽ đi xây dựng bảng đầu vào kích của RSFF. ***Bảng đầu vào kích gồm 2 phần, phần bên trái liệt kê ra các yêu cầu*** cần chuyển đổi của FF, và phần bên phải là các điều kiện tín hiệu đầu vào kích cần đảm bảo để đạt được các sự chuyển đổi ấy. Nếu các điều kiện đầu vào được đảm bảo thì FF sẽ chuyển đổi theo đúng yêu cầu. Thực chất bảng đầu vào kích của FF là sự khai triển bảng trạng thái của FF.

Ta viết lại bảng trạng thái của RSFF ở dạng khai triển như sau:

S^n	R^n	Q^n	Q^{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	X
1	1	1	X

Trong bảng này, tín hiệu ngõ ra ở trạng thái tiếp theo (Q^{n+1}) sẽ phụ thuộc vào tín hiệu các ngõ vào data (S, R) và tín hiệu ngõ ở ra trạng thái hiện tại (Q^n).

Từ bảng khai triển trên ta xây dựng được bảng đầu vào kích cho RSFF:

Q^n	Q^{n+1}	S^n	R^n
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Cũng từ bảng trạng thái khai triển ta có thể tìm được phương trình logic của RSFF bằng cách lập sơ đồ Karnaugh như sau:

Q^{n+1} / Q^n \ S^nR^n	00	01	11	10
0	0	0	X	1
1	1	0	X	1

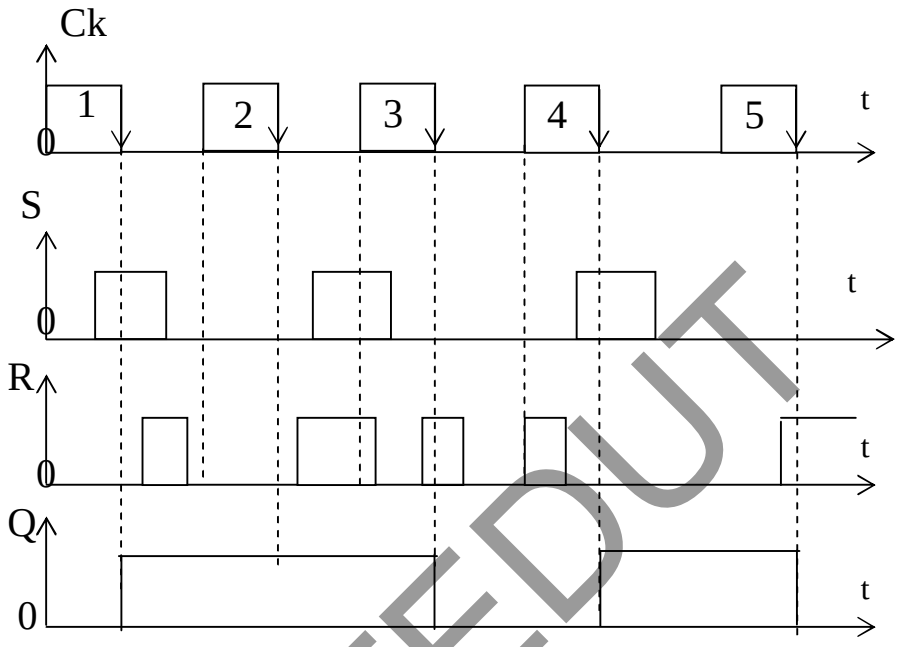
Từ bảng Karnaugh này ta có phương trình logic của RSFF:

$$Q^{n+1} = S^n + \overline{R^n} Q^n$$

Vì điều kiện của RSFF là S.R= 0 nên ta có phương trình logic của RSFF được viết đầy đủ như sau:

$$\begin{cases} Q^{n+1} = S^n + \overline{R}^n Q^n \\ SR=0 \end{cases}$$

Dạng sóng minh họa hoạt động của RSFF trên hình 3.56:

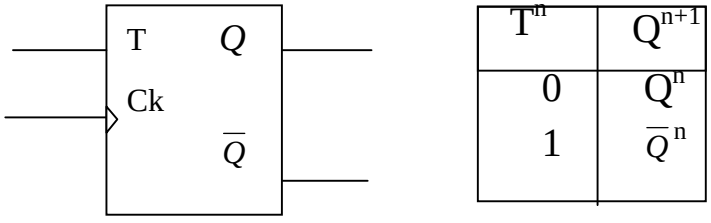


Hình 3.56. Đồ thị thời gian dạng sóng RSFF

b. TFF

TFF là FF có ngõ vào và ngõ ra ký hiệu và bảng trạng thái hoạt động như hình vẽ (hình 3.57):
Trong đó:

- T: ngõ vào dữ liệu
- *Q,Ġ: các ngõ ra*
- Ck: tín hiệu xung đồng bộ.



T^n	Q^{n+1}
0	Q^n
1	$\overline{Q}^n$

Hình 3.57. Ký hiệu TFF và bảng trạng thái hoạt động

Gọi T^n là trạng thái của ngõ vào DATA T ở xung Ck thứ n.
Gọi Q^n , Q^{n+1} là trạng thái của ngõ ra ở xung Ck thứ n và (n+1).
Lúc đó ta có bảng trạng thái hoạt động khai triển của TFF.
Từ bảng trạng thái này ta có nhận xét:

+ Khi T=0: mỗi khi có xung Ck tác động ngõ ra Q giữ nguyên trạng thái cũ trước đó.
+ Khi T=1: mỗi khi có xung Ck tác động ngõ ra Q đảo trạng thái.

T^n	Q^n	Q^{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

Từ bảng trạng thái khai triển của TFF ta tìm được bảng đầu vào kích của TFF như sau:

Q^n	Q^{n+1}	T^n
0	0	0
0	1	1
1	0	1
1	1	0

Phương trình logic của TFF:

$$Q^{n+1} = \overline{T^n}.Q^n + T^n.\overline{Q^n} \qquad \text{(dạng chính tắc 1)}$$

Hoặc: $Q^{n+1} = (T^n + Q^n)(\overline{T^n} + \overline{Q^n})$ (dạng chính tắc 2).

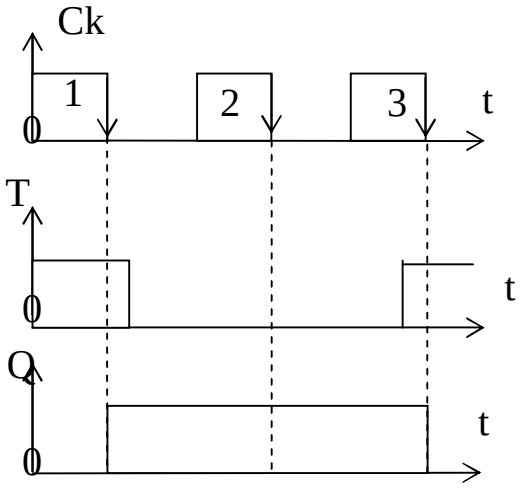
Viết gọn hơn:

$$Q^{n+1} = T^n \oplus Q^n$$

(SV có thể lập Karnaugh và tối thiểu hóa để tìm phương trinh logic của TFF).

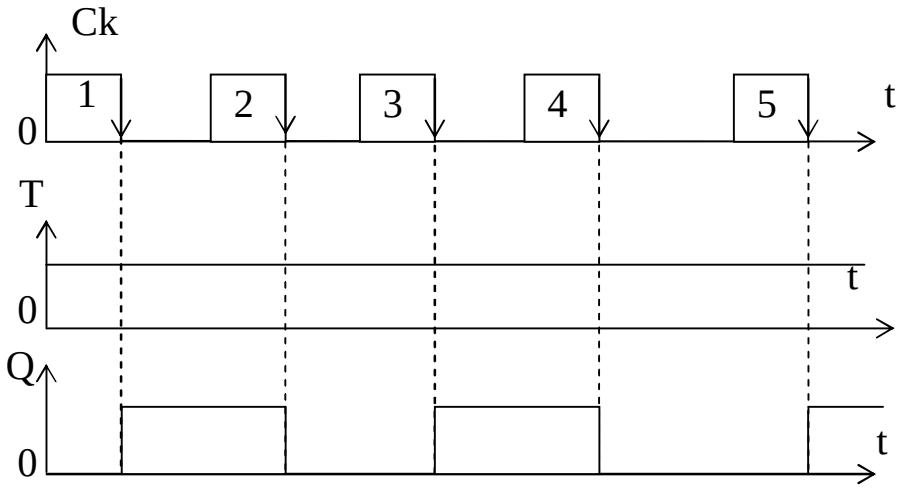
Trên hình 3.58 minh họa đồ thị thời gian dạng sóng của TFF.

- Tín hiệu ra Q đầu tiên luôn luôn ở mức logic 0
- Tín hiệu Ck(1) điều khiển theo sườn xuống nhìn tín hiệu T dưới mức logic 1. Theo bảng trạng thái : $T^0 = 1$ và $Q^0 = 0 \Rightarrow Q^1 = \overline{Q^0} = 1$.
- Tín hiệu Ck(2) điều khiển theo sườn xuống nhìn tín hiệu T dưới mức logic 0. Theo bảng trạng thái : $T^1 = 0$ và $Q^1 = 1 \Rightarrow Q^2 = Q^1 = 1$ (Giữ nguyên trạng thái trước đó).
- Tín hiệu Ck(3) điều khiển theo sườn xuống nhìn tín hiệu T dưới mức logic 1. Theo bảng trạng thái: $T^2 = 1$ và $Q^2 = 1 \Rightarrow Q^3 = \overline{Q^2} = 0$.



Hình 3.58

Trường hợp ngõ vào T luôn luôn bằng 1 (luôn ở mức logic 1):



Hình 3.59. Dạng sóng ngõ ra khi T=1

Khi T=1 thì dạng sóng ngõ ra Q được cho trên hình vẽ. Ta có nhận xét rằng chu kỳ của ngõ ra Q bằng 2 lần chu kỳ tín hiệu xung Ck nên tần số của ngõ ra là:

$$f_Q = \frac{f_{CK}}{2}$$

Vậy, khi T=1 thì TFF giữ vai trò mạch chia tần số xung vào Ck.

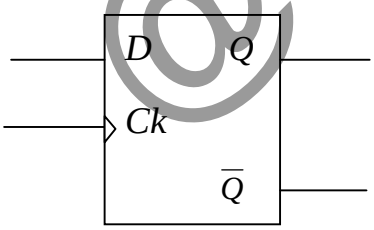
Tổng quát: Ghép nối tiếp n TFF với nhau sao cho ngõ ra của TFF trước sẽ nối với ngõ vào của TFF đứng sau (Ck_{i+1} nối với Q_i) và lúc bây giờ tất cả các ngõ vào DATA T ở tất cả các TFF đều giữ mức logic 1, lúc đó tần số tín hiệu ngõ ra sẽ là:

$$f_{Q_n} = \frac{f_{CK}}{2^n}$$

với Q_n là tín hiệu ngõ ra của TFF thứ n; f_{CK} là tần số xung clock ở ngõ vào đồng bộ TFF đầu tiên.

c. DFF

DFF là FF có ngõ vào và ngõ ra ký hiệu như hình 3.60.



Hình 3.60. Ký hiệu DFF

Bảng trạng thái

D^n	Q^{n+1}
0	0
1	1

Trong đó: D là ngõ vào dữ liệu. Q, $\overline{Q}$: các ngõ ra. Ck: tín hiệu xung đồng bộ.

Gọi D^n là trạng thaïi của ngõ vào DATA D ở xung Ck thứ n.

Gọi Q^n, Q^{n+1} là trạng thái của ngõ ra ở xung Ck thứ n và (n+1).

Khai triển bảng trạng thái của DFF để tìm bảng đầu vào kích của DFF, ta có:

D^n	Q^n	Q^{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

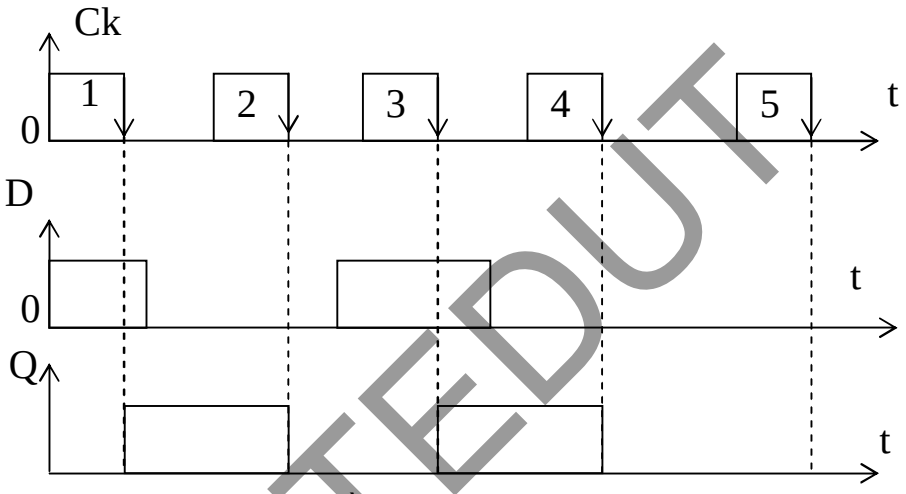
Bảng đầu vào kích của DFF:

Q^n	Q^{n+1}	D^n
0	0	0
0	1	1
1	0	0
1	1	1

Phương trình logic của DFF:

$$Q^{n+1} = D^n$$

Trên hình 3.61 là đồ thị thời gian dạng sóng của DFF:

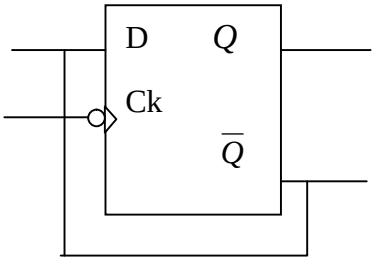


Hình 3.61. Đồ thị thời gian dạng sóng của DFF

Giải thích dạng sóng của tín hiệu trên hình 3.61:

- Tín hiệu ra Q đầu tiên luôn luôn ở mức logic 0, $Q^0 = 0$
- Tín hiệu Ck(1) điều khiển theo sườn xuống nhìn tín hiệu D dưới mức logic 1. Theo bảng trạng thái ta có: $D^0 = 1 \Rightarrow Q^1 = 1$
- Tín hiệu Ck(2) điều khiển theo sườn xuống nhìn tín hiệu D dưới mức logic 0. Theo bảng trạng thái ta có :$D^1 = 0 \Rightarrow Q^2 = 0$

..v..v..



Hình 3.62.

<u>DFF đóng vai trò mạch chia tần số</u>:

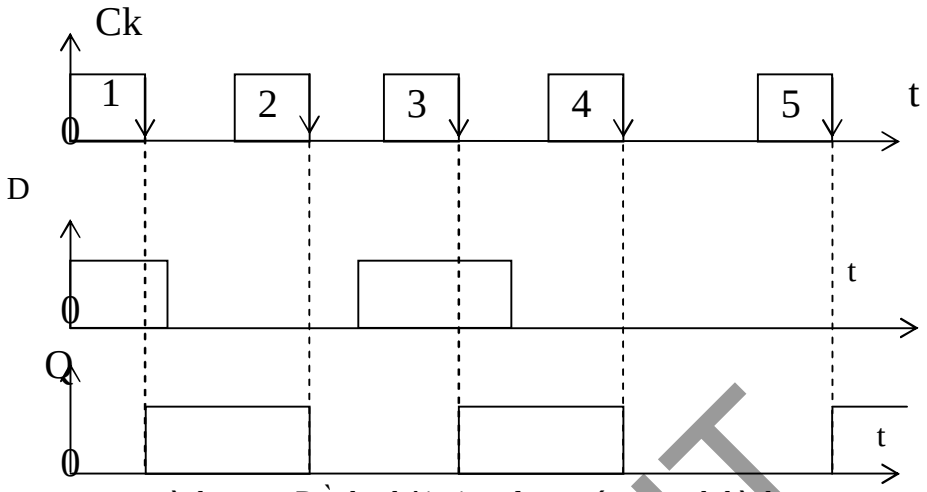
Trên hình 3.62 là sơ đồ mạch DFF thực hiện chức năng chia tần số. Ở mạch này ngõ ra $\overline{Q}$ được nối ngược trở về ngõ vào D.

- Tín hiệu ra Q^0 đầu tiên luôn ở mức logic 0:

$$Q^0 = 0 \Rightarrow \overline{Q^0} = D^1 = 1$$

- Tín hiệu Ck(1) điều khiển theo sườn xuống nhìn tín hiệu D^1 dưới mức logic 1. $D^1 = 1 \Rightarrow Q^1 = 1 \Rightarrow \overline{Q^1} = D^2 = 0$.
- Tín hiệu Ck(2) điều khiển theo sườn xuống nhìn tín hiệu D^2 dưới mức logic 0. $D^2 = 0 \Rightarrow Q^2 = 0 \Rightarrow \overline{Q^2} = D^3 = 1$.

- Tín hiệu Ck(3) điều khiển theo sườn xuống nhìn tín hiệu D^3 dưới mức logic 1. $D^3 = 1 \Rightarrow Q^3 = 1 \Rightarrow \overline{Q^3} = D^4 = 0$.
- Tín hiệu Ck(4) điều khiển theo sườn xuống nhìn tín hiệu D^4 dưới mức logic 0. $\Rightarrow Q^4 = 0$ ..v..v..



Hình 3.63. Đồ thị thời gian dạng sóng mạch hình 3.62

Nhận xét về tần số ngõ ra:

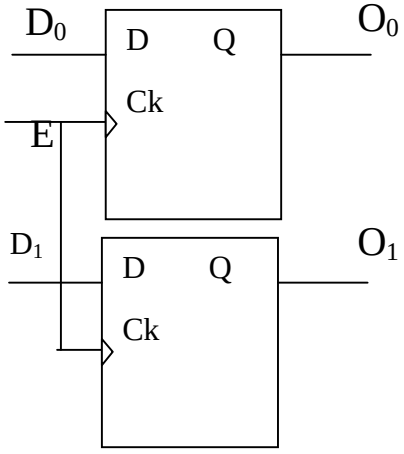
$$f_Q = \frac{f_{CK}}{2} \quad \Rightarrow \text{DFF giữ vai trò như mạch chia tần số.}$$

<u>Ứng dụng của DFF</u>:

- Dùng DFF để chia tần số.
- Dùng DFF để lưu trữ dữ liệu để chế tạo các bộ nhớ và thanh ghi.
- Dùng DFF để chốt dữ liệu.

Trên hình 3.64 là sơ đồ mạch ứng dụng DFF để chốt dữ liệu. Hoạt động của mạch như sau:

- E=1: $O_0 = D_0$, $O_1 = D_1$ nên tín hiệu được đưa đến các FF.
- E=0: $O_0 = D_0$, $O_1 = D_1 \rightarrow$ chốt dữ liệu trở lại.



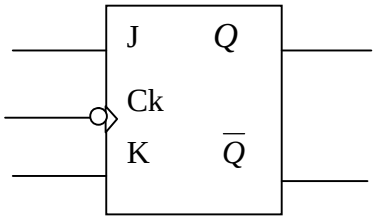
Hình 3.64. Chốt dữ liệu dùng DFF

<u>d. JKFF</u>

JKFF là FF có ngõ vào và ngõ ra ký hiệu như hình vẽ :

Trong đó:

- J, K là các ngõ vào dữ liệu.
- Q, $\overline{Q}$ là các ngõ ra.
- Ck là tín hiệu xung đồng bộ.



Hình 3.65. JKFF

Gọi J^n , K^n là trạng thái ngõ vào J,K ở xung Ck thứ n.

Gọi Q^n, Q^{n+1} là trạng thái ngõ ra Q ở xung Ck thứ n và (n+1).

Lúc đó ta có bảng trạng thái mô tả hoạt động của JKFF:

J	**K**	**Q^{n+1}**
0	0	Q^n
0	1	0

1 1	0 1	1 $\overline{Q}^n$

Phương trình logic của JKFF:

$$Q^{n+1} = J^n \overline{Q^n} + \overline{K^n}.Q^n$$

Từ bảng trạng thái ta thấy JKFF khắc phục được trạng thái cấm của RSFF, khi J=K=1 ngõ ra ở trạng thái kế tiếp đảo mức logic so với ngõ ra ở trạng thái hiện tại.

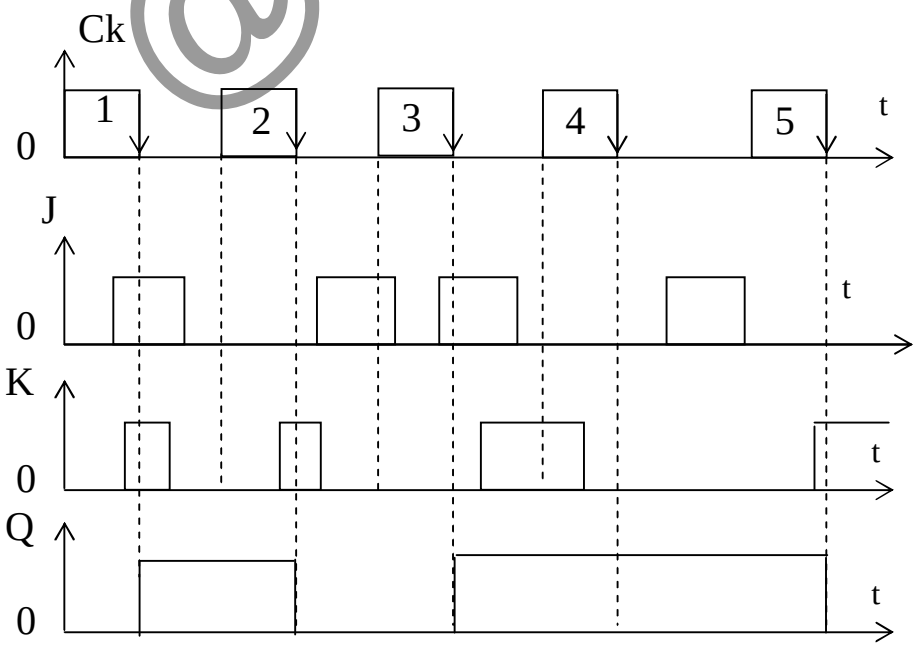
Để tìm bảng đầu vào kích của JKFF ta khai triển bảng trạng thái như sau:

J^n	K^n	Q^n	Q^{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Từ bảng khai triển trên ta xây dựng được bảng đầu vào kích cho JKFF như sau:

Q^n	Q^{n+1}	S^n	R^n
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

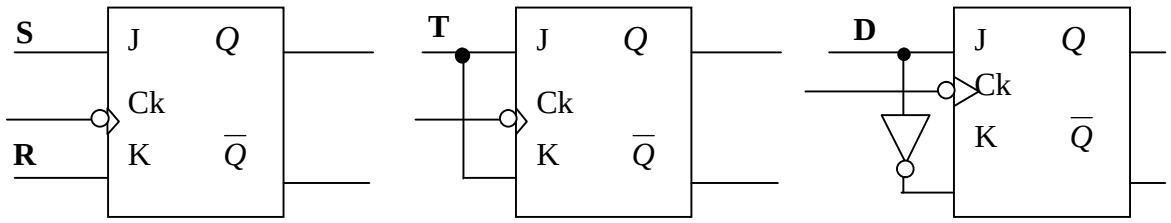
Đồ thị thời gian dạng sóng của JKFF:



Hình 3.66. Đồ thị thời gian dạng sóng JKFF

Nhận xét quan trọng: JKFF là mạch điện có chức năng thiết lập trạng thái 0, trạng thái 1, chuyển đổi trạng thái và duy trì trạng thái căn cứ vào các tín hiệu đầu vào J, K và xung nhịp đồng bộ Ck. Như vậy có thể nói JKFF là một FF rất vạn năng.

Trong thực tế, chúng ta có thể dùng JKFF để thực hiện chức năng của các FF khác: JKFF thay thế cho RSFF, JKFF thực hiện chức năng của TFF và DFF, các sơ đồ thực hiện được trình bày trên



Hình 3.67. Dùng JKFF thực hiện chức năng của RSFF, TFF, DFF

hình 3.67:

Trên cơ sở khảo sát về 4 loại FF phân chia theo chức năng, chúng ta có thể xây dựng một bảng đầu vào kích tổng hợp cho cả 4 loại FF như sau:

Q^n	Q^{n+1}	S^n	R^n	J^n	K^n	T^n	D^n
0	0	0	X	0	X	0	0
0	1	1	0	1	X	1	1
1	0	0	1	X	1	1	0
1	1	X	0	X	0	0	1

3.3.3. Sự chuyển đổi lẫn nhau giữa các loại FF

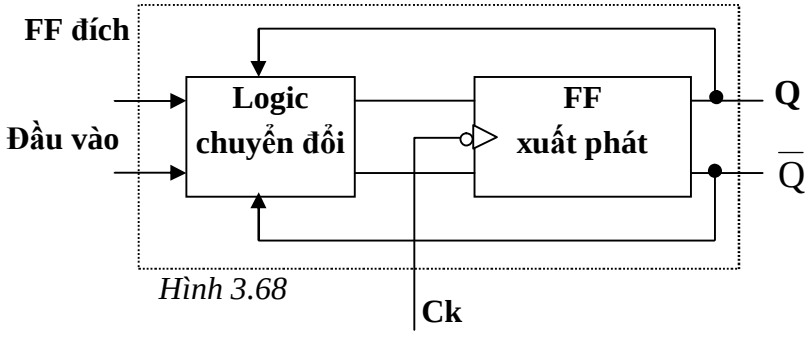
Đa số FF trên thị trường là loại JK, D trong khi kỹ thuật số yêu cầu tất cả các loại FF. Nếu biết cách chuyển đổi giữa các loại FF với nhau thì có thể phát huy tác dụng của loại FF sẵn có.

Trên thực tế, có thể chuyển đổi qua lại giữa các loại FF khác nhau. Có 2 phương pháp để thực hiện chuyển đổi giữa các loại FF:

- phương pháp biến đổi trực tiếp.
- phương pháp dùng bảng đầu vào kích và bảng Karnaugh.

a. Phương pháp biến đổi trực tiếp:

Đây là phương pháp sử dụng các định lý, tiên đề của đại số Boole để tìm phương trình logic tín hiệu kích thích đối với FF xuất phát. Sơ đồ khối thực hiện phương pháp này như sau (hình 3.68):



Hình 3.68

TFF chuyển đổi thành DFF, RSFF, JKFF:

- TFF $\rightarrow$ RSFF:

  RSFF có pt: $\begin{cases} Q^{n+1} = S^n + \overline{R^n}\, Q^n & (1) \\ S^n R^n = 0 & \text{(điều kiện của RSFF)} \end{cases}$

  TFF có pt: $Q^{n+1} = T^n \oplus Q^n$ (2)

  So sánh (1) và (2) ta có:

  $$S^n + \overline{R^n}\, Q^n = T^n \oplus Q^n$$

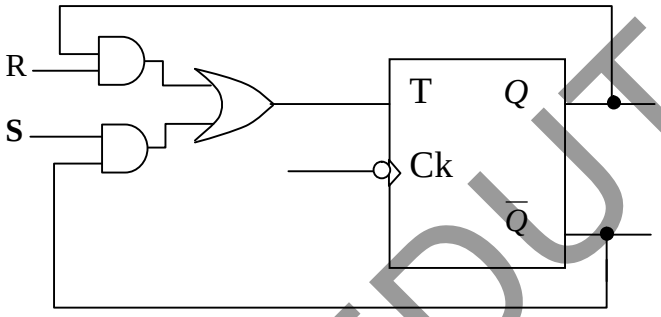
  Theo tính chất của phép toán XOR, ta có:

  $$T^n = Q^n \oplus (S^n + \overline{R^n}\, Q^n) = Q^n\, \overline{(S^n + \overline{R^n} Q^n)} + \overline{Q^n}\, (S^n + \overline{R^n}\, Q^n)$$

  $$= Q^n\, \overline{S^n}\, R^n + S^n \overline{Q^n} = Q^n\, \overline{S^n}\, R^n + S^n \overline{Q^n} + S^n R^n = Q^n R^n + S^n \overline{Q^n}$$

  Vậy: $T^n = Q^n R^n + S^n \overline{Q^n}$

  Sơ đồ mạch thực hiện:



Hình 3.69. Chuyển đổi TFF thành RSFF

- TFF$\rightarrow$ DFF:

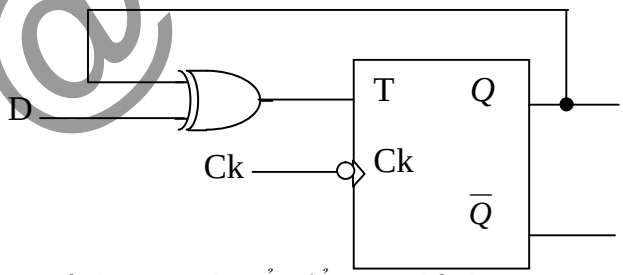
  DFF có phương trình logic: $Q^{n+1} = D^n$

  TFF có phương trình logic: $Q^{n+1} = T^n \oplus Q^n$

  Đồng nhất 2 phương trình: $D^n = T^n \oplus Q^n$

  Theo tính chất của phép XOR ta suy ra: $T^n = D^n \oplus Q^n$

  Sơ đồ mạch thực hiện:

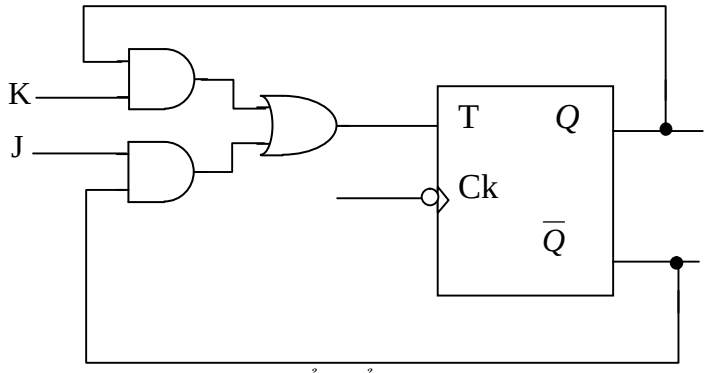


Hình 3.70. Chuyển đổi TFF thành DFF

- TFF$\rightarrow$ DFF: Thực hiện biến đổi hoàn toàn tương tự (như trường hợp chuyển đổi từ TFF sang RSFF) ta có logic chuyển đổi:

$$T^n = K^n Q^n + J^n \overline{Q^n}$$

Sơ đồ mạch chuyển đổi từ TFF sang JKFF



Hình 3.71. Chuyển đổi TFF thành JKFF

<u>DFF chuyển đổi thành TFF, RSFF, JKFF</u>:

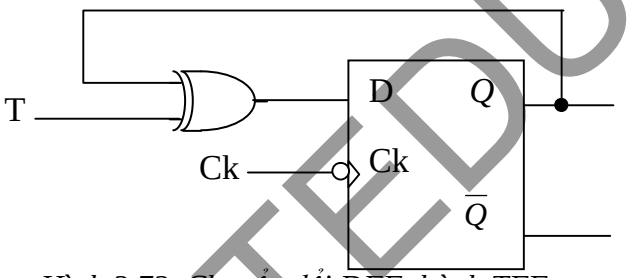
- DFF→ TFF:

  DFF có phương trình logic: $Q^{n+1} = D^n$

  TFF có phương trình logic: $Q^{n+1} = T^n \oplus Q^n$

  Đồng nhất 2 phương trình ta có: $D^n = T^n \oplus Q^n$

  Sơ đồ mạch thực hiện chuyển đổi (hình 3.72):



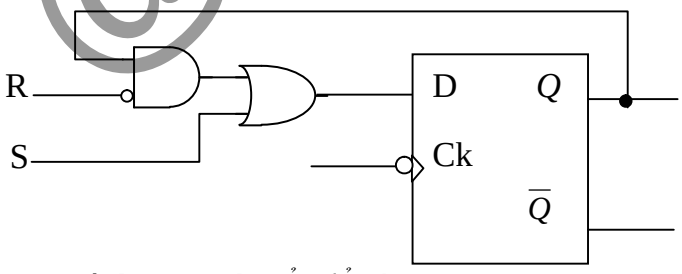
Hình 3.72. Chuyển đổi DFF thành TFF

- DFF→ RSFF:

  RSFF có phương trình logic: $Q^{n+1} = S^n + \overline{R^n}\, Q^n$

  Đồng nhất với phương trình của DFF ta có: $D^n = S^n + \overline{R^n}\, Q^n$

  Sơ đồ mạch thực hiện chuyển đổi:



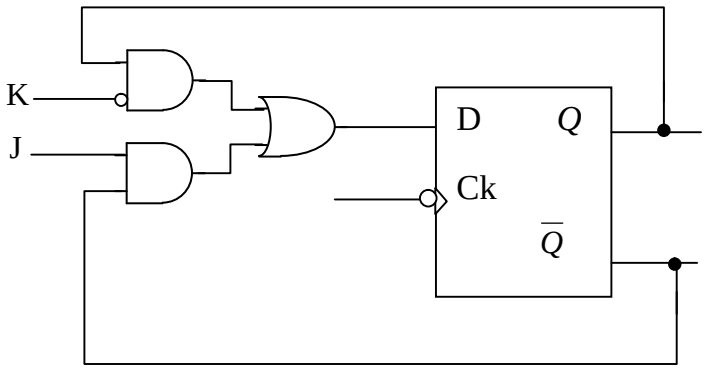
Hình 3.73. Chuyển đổi từ DFF sang RSFF

- DFF→ JKFF:

  Hoàn toàn tương tự ta có logic chuyển đổi từ DFF sang JKFF:

  $$D^n = J^n\, \overline{Q^n} + \overline{K^n}\, Q^n$$

  Sơ đồ mạch chuyển đổi trên hình 3.74:



Hình 3.74. Chuyển đổi DFF thành JKFF

RSFF chuyển đổi thành TFF, DFF, JKFF:

RSFF có pt: $\begin{cases} Q^{n+1} = S^n + \overline{R^n}\, Q^n \\ S^n R^n = 0 \quad \text{(điều kiện của RSFF)} \end{cases}$

Khi thực hiện chuyển đổi từ RSFF sang các FF khác cần kiểm tra điều kiện ràng buộc của RSFF đó là: $\mathbf{R^n S^n = 0}$.

- RSFF→ TFF:

  TFF có phương trình logic: $Q^{n+1} = T^n \oplus Q^n$

  Đồng nhất với phương trình của RSFF ta có:

  $$S^n + \overline{R^n}\, Q^n = T^n \oplus Q^n = T^n \overline{Q^n} + \overline{T^n}\, Q^n$$

  Từ biểu thức này, nếu ta đồng nhất:

  $\begin{cases} S^n = T^n \overline{Q^n} \\ R^n = T^n \end{cases}$

  thì suy ra:

  $$S^n R^n = T^n \overline{Q^n} . T^n = T^n \overline{Q^n} \neq 0$$

  nên không thỏa mãn điều kiện của RSFF.

  Thực hiện biến đổi tiếp:

  $$S^n + \overline{R^n}\, Q^n = T^n \overline{Q^n} + \overline{T^n}\, Q^n = T^n \overline{Q^n} + \overline{T^n}\, Q^n + \overline{Q^n}\, Q^n$$

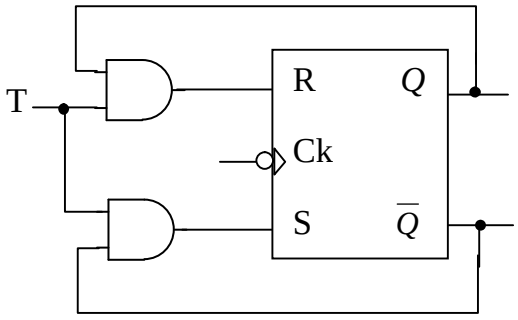
  $$S^n + \overline{R^n}\, Q^n = T^n \overline{Q^n} + (\overline{T^n} + \overline{Q^n}) Q^n = T^n \overline{Q^n} + \overline{T^n} \overline{Q^n}\, Q^n$$

  Đồng nhất 2 vế ta có:

  $\begin{cases} S^n = T^n \overline{Q^n} \\ R^n = T^n Q^n \end{cases}$

  thỏa mãn điều kiện: $R^n S^n = 0$.

  Sơ đồ thực hiện: hình 3.75.



Hình 3.75. Chuyển đổi RSFF sang TFF

- RSFF→ DFF: $Q^{n+1} = D^n$

  Đồng nhất 2 phương trình: $S^n + \overline{R^n}\, Q^n = D^n$

  Thực hiện biến đổi:

  $$S^n + \overline{R^n}\, Q^n = D^n = D^n (Q^n + \overline{Q^n}) = D^n Q^n + D^n \overline{Q^n} \qquad \textbf{(a)}$$

  Mặt khác biểu thức của RSFF có thể biến đổi như sau:

$$S^n + \overline{R^n} Q^n = S^n(Q^n + \overline{Q^n}) + \overline{R^n} Q^n = S^nQ^n + S^n\overline{Q^n} + \overline{R^n} Q^n$$
$$= S^nQ^n (R^n + \overline{R^n}) + S^n\overline{Q^n} + \overline{R^n} Q^n$$
$$= S^nQ^n \overline{R^n} + S^n\overline{Q^n} + \overline{R^n} Q^n$$
$$= \overline{R^n} Q^n (1 + S^n) + S^n\overline{Q^n}$$
$$= \overline{R^n} Q^n + S^n\overline{Q^n} \quad \textbf{(b)}$$

Từ (a) và (b) ta có:

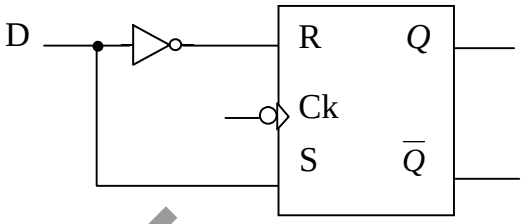
$$D^n Q^n + D^n\overline{Q^n} = \overline{R^n} Q^n + S^n\overline{Q^n}$$

Đồng nhất 2 vế suy ra:

$$\begin{cases} S^n = D^n \\ R^n = \overline{D^n} \end{cases}$$

thỏa mãn điều kiện $R^nS^n = 0$.

Sơ đồ thực hiện: hình 3.76.



Hình 3.76. RSFF→ DFF

- RSFF→ JKFF:

Đồng nhất 2 phương trình logic của RSFF và JKFF ta có:

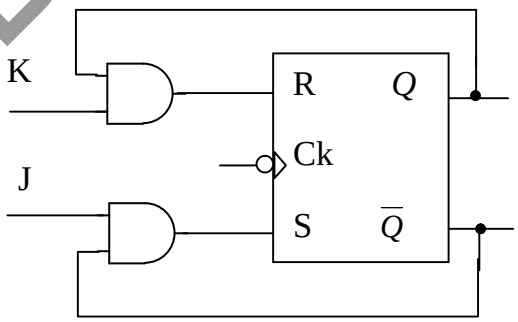
$$Q^{n+1} = S^n + \overline{R^n} Q^n = J^n \overline{Q^n} + \overline{K^n} Q^n$$
$$= J^n \overline{Q^n} + \overline{K^n} Q^n + Q^n\overline{Q^n} = J^n \overline{Q^n} + (\overline{K^n} + \overline{Q^n})Q^n = J^n \overline{Q^n} + \overline{K^n}\overline{Q^n} Q^n$$

So sánh ta có:

$$\begin{cases} S^n = J^n\overline{Q^n} \\ R^n = K^nQ^n \end{cases}$$

thỏa mãn điều kiện của RSFF.

Sơ đồ thực hiện: hình 3.77.



Hình 3.77. RSFF→ JKFF

JKFF chuyển đổi thành TFF, DFF, RSFF:

Như đã trình bày ở trên, JKFF là một FF vạn năng, có thể dùng JKFF để thay thế cho RSFF hoặc dùng JKFF thực hiện chức năng DFF, TFF. Sơ đồ thực hiện các mạch này như ở hình 3.67. Phần này tập trung chứng minh các biểu thức logic chuyển đổi từ JKFF sang các FF khác.

JKFF có phương trình logic: $Q^{n+1} = J^n \overline{Q^n} + \overline{K^n} Q^n$

- JKFF→ TFF:

TFF có phương trình logic: $Q^{n+1} = T^n \oplus Q^n = T^n\overline{Q^n} + \overline{T^n} Q^n$

So sánh với phương trình của JKFF ta suy ra logic chuyển đổi:

$$\begin{cases} J^n = T^n \\ K^n = T^n \end{cases}$$

- JKFF→ DFF:

DFF có phương trình logic: $Q^{n+1} = D^n$

Viết lại biểu thức này ta có: $Q^{n+1}=D^n=D^n(Q^n + \overline{Q^n}) = D^nQ^n + D^n\overline{Q^n}$

So sánh với biểu thức của JKFF ta có logic chuyển đổi:

$$\begin{cases} J^n = D^n \\ K^n = \overline{D^n} \end{cases}$$

- JKFF→ RSFF:

  Đối với RSFF có phương trình logic đã tìm được ở công thức (b):

  $$Q^{n+1} = S^n + \overline{R^n} Q^n = S^n \overline{Q^n} + \overline{R^n} Q^n \qquad \textbf{(b)}$$

  So sánh với phương trình logic của JKFF ta có logic chuyển đổi:

  $$\begin{cases} J^n = S^n \\ K^n = R^n \end{cases}$$

b. Phương pháp dùng bảng đầu vào kích và bảng Karnaugh:

Trong phương pháp này, các đầu vào dữ liệu (data) của FF ban đầu là hàm ra với các biến là trạng thái ngõ ra Qn và các đầu vào data của FF cần chuyển đổi. Để thực hiện chuyển đổi ta dựa vào bảng tín hiệu đầu vào kích của các FF và lập bảng Karnaugh, thực hiện tối giản để tìm logic chuyển đổi. Bảng tín hiệu đầu vào kích tổng hợp như sau:

Q^n	Q^{n+1}	S^n	R^n	J^n	K^n	T^n	D^n
0	0	0	X	0	X	0	0
0	1	1	0	1	X	1	1
1	0	0	1	X	1	1	0
1	1	X	0	X	0	0	1

Xét các trường hợp cụ thể:

- chuyển đổi từ JKFF → TFF : J = f (T,Q^n) và K = f (T,Q^n)
- chuyển đổi từ JKFF → DFF : J = f (D,Q^n) và K = f (D,Q^n)
- chuyển đổi từ JKFF → RSFF : J = f (S,R,Q^n) và K = f (S,R,Q^n)
- chuyển đổi từ RSFF → TFF : R = f (T,Q^n) và S = f (T,Q^n)
- chuyển đổi từ RSFF → DFF : R = f (D,Q^n) và S = f (D,Q^n)
- chuyển đổi từ RSFF → JKFF : R = f (J, K,Q^n) và S = f (J,K,Q^n)
- chuyển đổi từ TFF → DFF : T = f (D,Q^n)
- chuyển đổi từ TFF → RSFF : T = f (R,S,Q^n)
- chuyển đổi từ TFF → JKFF : T = f (J,K,Q^n)
- chuyển đổi từ DFF → TFF : D = f (T,Q^n)
- chuyển đổi từ DFF → RSFF : D = f (R,S,Q^n)
- chuyển đổi từ DFF → JKFF : D = f (J,K,Q^n)

Ví dụ 1: Chuyển đổi từ JKFF → DFF dùng phương pháp bảng.

Ta có các hàm cần tìm:

J = f (D, Q^n) vaì K = f (D, Q^n)

Dựa vào bảng đầu vào kích tổng hợp ta lập bảng Karnaugh:

J / Q^n \ D	0	1
0	0	1
1	X	X

J = D

K / Q^n \ D	0	1
0	X	X
1	1	0

K = $\overline{D}$

Tối giản theo dạng chính tắc 1 ta có: J = D và K = $\overline{D}$.

Ví dụ 2: Chuyển đổi từ JKFF → RSFF dùng phương pháp bảng.

Ta có các hàm cần tìm:

J = f (S,R,Q^n)

K = f (S,R,Q^n)

Dựa vào bảng đầu vào kích tổng hợp lập bảng Karnaugh (xem bảng).

Tối giản theo dạng chính tắc 1 ta có: J = S và K = R.

J / Q^n \ SR	00	01	11	10
0	0	0	X	1
1	X	X	X	X

J = S

K / Q^n \ SR	00	01	11	10
0	X	X	X	X
1	0	1	X	0

K = R

Chương 4

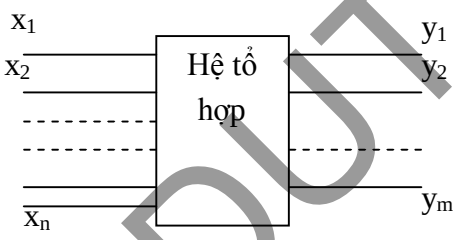
HỆ TỔ HỢP

4.1.KHÁI NIỆM CHUNG

Các phần tử logic AND, OR, NOR, NAND là các phần tử logic cơ bản còn được gọi là hệ tổ hợp đơn giản. Như vậy, hệ tổ hợp là hệ có các ngõ ra là các hàm logic theo ngõ vào, điều này nghĩa là khi một trong các ngõ vào thay đổi trạng thái lập tức làm cho ngõ ra thay đổi trạng thái ngay (nếu bỏ qua thời gian trễ của các phần tử logic) mà không chịu ảnh hưởng của trạng thái ngõ ra trước đó.

Xét một hệ tổ hợp có n ngõ vào và có m ngõ ra (hình 4.1), ta có:

$y_1 = f(x_1, x_2, ..., x_n)$
$y_2 = f(x_1, x_2, ..., x_n)$
.....
$y_m = f(x_1, x_2, ..., x_n)$



Hình 4.1

Như vậy, sự thay đổi của ngõ ra y_j (**j = 1 ÷ m**) theo các biến vào xi (**i = 1 ÷ n**) là tuỳ thuộc vào bảng trạng thái mô tả hoạt động của hệ tổ hợp.

Đặc điểm cơ bản của hệ tổ hợp là tín hiệu ra tại mỗi thời điểm chỉ phụ thuộc vào giá trị các tín hiệu vào ở thời điểm đó mà không phụ thuộc vào giá trị các tín hiệu ngõ ra ở thời điểm trước đó.

Trình tự để thiết kế hệ tổ hợp theo các bước sau:

1. Từ yêu cầu thực tế ta lập bảng trạng thái mô tả hoạt động của mạch (hệ tổ hợp).
2. Dùng các phương pháp tối thiểu để tối thiểu hoá các hàm logic.
3. Thành lập sơ đồ logic (Dựa vào phương trình logic đã tối giản).
4. Thành lập sơ đồ hệ tổ hợp.

Các mạch tổ hợp thông dụng:

- Mạch mã hoá - giải mã
- Mạch chọn kênh - phân đường
- Mạch so sánh
- Mạch số học ....v....v....

4.2. MẠCH MÃ HOÁ & MẠCH GIẢI MÃ

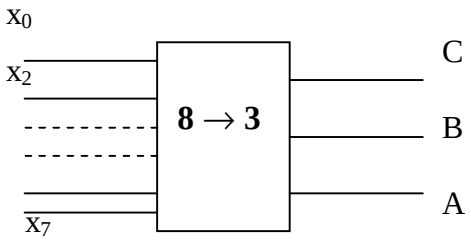
4.2.1. Khái niệm:

Mạch mã hoá (ENCODER) là mạch có nhiệm vụ biến đổi những ký hiệu quen thuộc với con người sang những ký hiệu không quen thuộc con người. Ngược lại, mạch giải mã (DECODER) là mạch làm nhiệm vụ biến đổi những ký hiệu không quen thuộc với con người sang những ký hiệu quen thuộc với con người.

4.2.2. Mạch mã hoá (Encoder)

1. Mạch mã hoá nhị phân

Xét mạch mã hóa nhị phân từ 8 sang 3 (8 ngõ vào và 3 ngõ ra). Sơ đồ khối của mạch được cho trên hình 4.2.



Hình 4.2 Sơ đồ khối mạch mã hóa nhị phân từ 8 sang 3

Trong đó:

- x_0, x_1,..., x_7 là 8 đường tín hiệu vào
- A, B, C là 3 ngõ ra.

Mạch mã hóa nhị phân thực hiện biến đổi tín hiệu ngõ vào thành một từ mã nhị phân tương ứng ở ngõ ra, cụ thể như sau:

$0 \rightarrow 000$ $3 \rightarrow 011$ $6 \rightarrow 100$

$1 \rightarrow 001$ $4 \rightarrow 100$ $7 \rightarrow 111$

$2 \rightarrow 010$ $5 \rightarrow 101$

Chọn mức tác động (tích cực) ở ngõ vào là mức logic 1, ta có bảng trạng thái mô tả hoạt động của mạch :

x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	C	B	A
1	0	0	0	0	0	0	0	0	0	0
0	**1**	0	0	0	0	0	0	0	0	1
0	0	**1**	0	0	0	0	0	0	1	0
0	0	0	**1**	0	0	0	0	0	1	1
0	0	0	0	**1**	0	0	0	1	0	0
0	0	0	0	0	**1**	0	0	1	0	1
0	0	0	0	0	0	**1**	0	1	1	0
0	0	0	0	0	0	0	**1**	1	1	1

Giải thích bảng trạng thái: Khi một ngõ vào ở trạng thái tích cực (mức logic 1) và các ngõ vào còn lại không được tích cực (mức logic 0) thì ngõ ra xuất hiện từ mã tương ứng. Cụ thể là: khi ngõ vào x0=1 và các ngõ vào còn lại bằng 0 thì từ mã ở ngõ ra là 000, khi ngõ vào x1=1 và các ngõ vào còn lại bằng 0 thì từ mã nhị phân ở ngõ ra là 001, ..v..v..

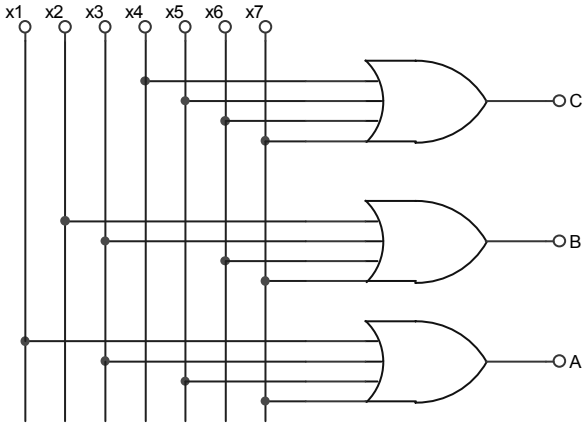
Phương trình logic tối giản:

$A = x_1 + x_3 + x_5 + x_7$

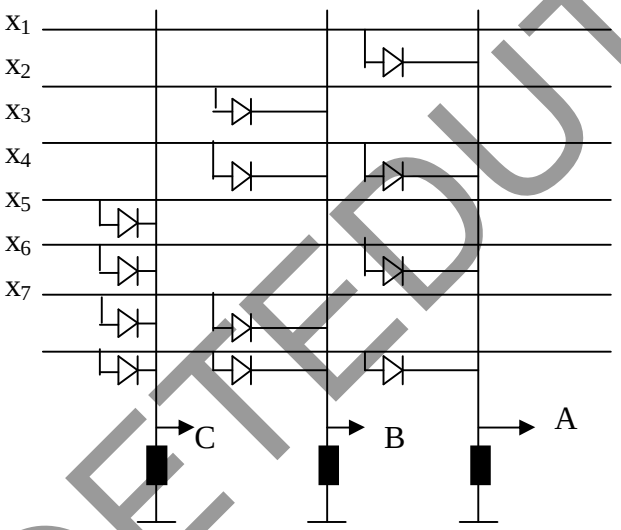
$B = x_2 + x_3 + x_6 + x_7$

$C = x_4 + x_5 + x_6 + x_7$

Sơ đồ logic thực hiện mạch mã hóa nhị phân từ 8 sang 3 (hình 4.3):
Biểu diễn bằng cổng logic dùng Diode (hình 4.4):



Hình 4.3 Mạch mã hóa nhị phân từ 8 sang 3



Hình 4.4 Mạch mã hóa nhị phân từ 8 sang 3 sử dụng diode

Nếu chọn mức tác động tích cực ở ngõ vào là mức logic 0, bảng trạng thái mô tả hoạt động của mạch lúc này như sau:

x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	C	B	A
0	1	1	1	1	1	1	1	0	0	0
1	**0**	1	1	1	1	1	1	0	0	1
1	1	**0**	1	1	1	1	1	0	1	0
1	1	1	**0**	1	1	1	1	0	1	1
1	1	1	1	**0**	1	1	1	1	0	0
1	1	1	1	1	**0**	1	1	1	0	1
1	1	1	1	1	1	**0**	1	1	1	0
1	1	1	1	1	1	1	**0**	1	1	1

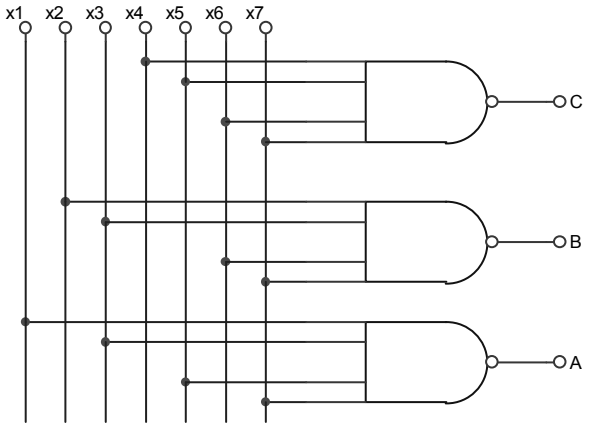
Phương trình logic tối giản :

$$A = \overline{x}_1 + \overline{x}_3 + \overline{x}_5 + \overline{x}_7 = \overline{x_1 x_3 x_5 x_7}$$

$$B = \overline{x}_2 + \overline{x}_3 + \overline{x}_6 + \overline{x}_7 = \overline{x_2 x_3 x_6 x_7}$$

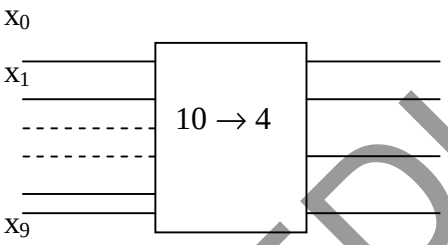
$$C = \overline{x}_4 + \overline{x}_5 + \overline{x}_6 + \overline{x}_7 = \overline{x_4 x_5 x_6 x_7}$$

Sơ đồ mạch thực hiện cho trên hình 4.5



Hình 4.5 Mạch mã hóa nhị phân 8 sang 3 ngõ vào tích cực mức 0

2. Mạch mã hoá thập phân



Hình 4.6 Sơ đồ khối mạch mã hóa từ 10 sang 4

Bảng trạng thái mô tả hoạt động của mạch :

x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	D	C	B	A
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	**1**	0	0	0	0	0	0	0	0	0	0	0	1
0	0	**1**	0	0	0	0	0	0	0	0	0	1	0
0	0	0	**1**	0	0	0	0	0	0	0	0	1	1
0	0	0	0	**1**	0	0	0	0	0	0	1	0	0
0	0	0	0	0	**1**	0	0	0	0	0	1	0	1
0	0	0	0	0	0	**1**	0	0	0	0	1	1	0
0	0	0	0	0	0	0	**1**	0	0	0	1	1	1
0	0	0	0	0	0	0	0	**1**	0	1	0	0	0
0	0	0	0	0	0	0	0	0	**1**	1	0	0	1

Phương trình logic đã tối giản:

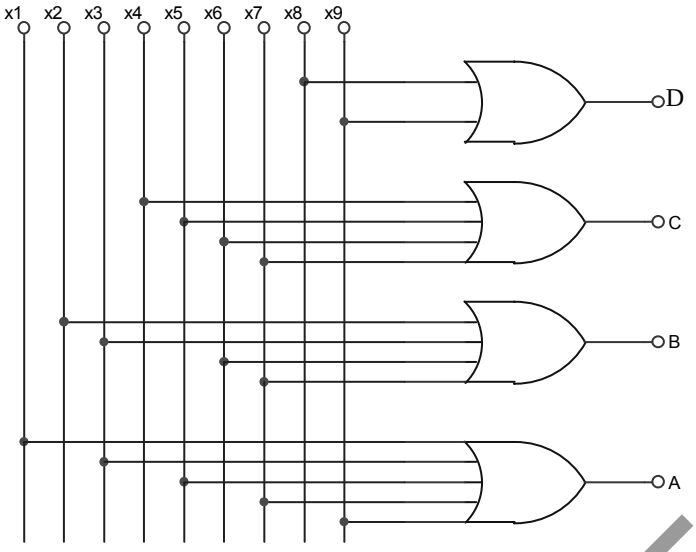
$A = x_1 + x_3 + x_5 + x_7 + x_9$

$B = x_2 + x_3 + x_6 + x_7$

$C = x_4 + x_5 + x_6 + x_7$

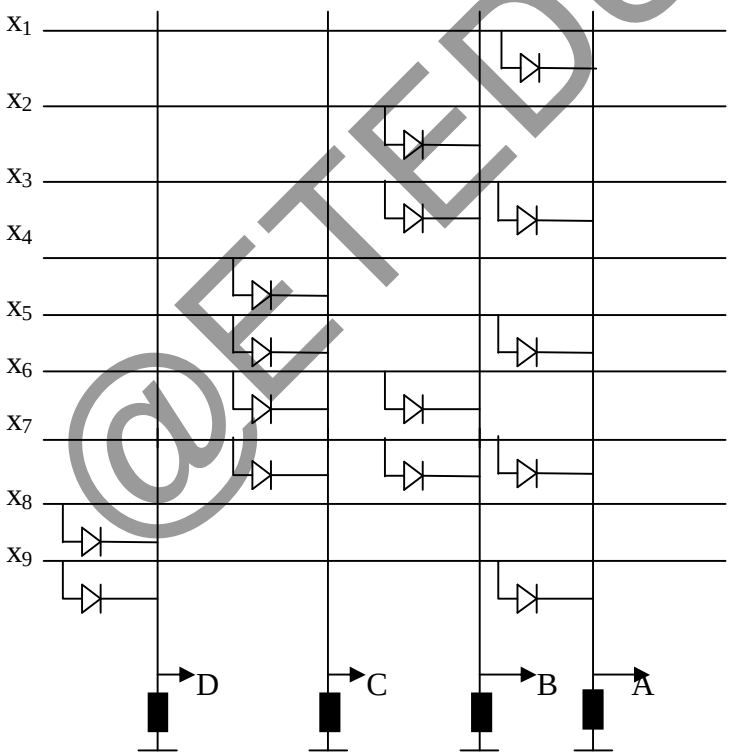
$D = x_8 + x_9$

Biểu diễn bằng sơ đồ logic (hình 4.7)



Hình 4.7 Sơ đồ mạch mã hóa thập phân từ 10 → 4

Biểu diễn sơ đồ này bằng cổng logic sử dụng Diode được cho trên *hình 4.8*



Hình 4.8

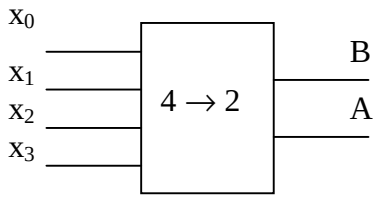
3. Mạch mã hoá ưu tiên

Trong hai mạch mã hoá đã xét ở trên, tín hiệu đầu vào tồn tại độc lập tức là không có tình huống có 2 tín hiệu trở lên đồng thời tác động ở mức logic 1 (nếu ta chọn mức tích cực ở ngõ vào là mức logic 1), thực tế đây là tình huống hoàn toàn có thể xảy ra, do đó cần phải đặt ra vấn đề ưu tiên.

Vấn đề ưu tiên: Khi có nhiều tín hiệu vào đồng thời tác động, tín hiệu nào có mức ưu tiên cao hơn ở thời điểm đang xét sẽ được ưu tiên tác động, tức là nếu ngõ vào có độ ưu tiên cao hơn bằng 1

trong khi những ngõ vào có độ ưu tiên thấp hơn nếu bằng 1 thì mạch sẽ tạo ra từ mã nhị phân ứng với ngõ vào có độ ưu tiên cao nhất.

Xét mạch mã hoá ưu tiên 4 → 2 (4 ngõ vào, 2 ngõ ra) (hình 4.9).



Hình 4.9

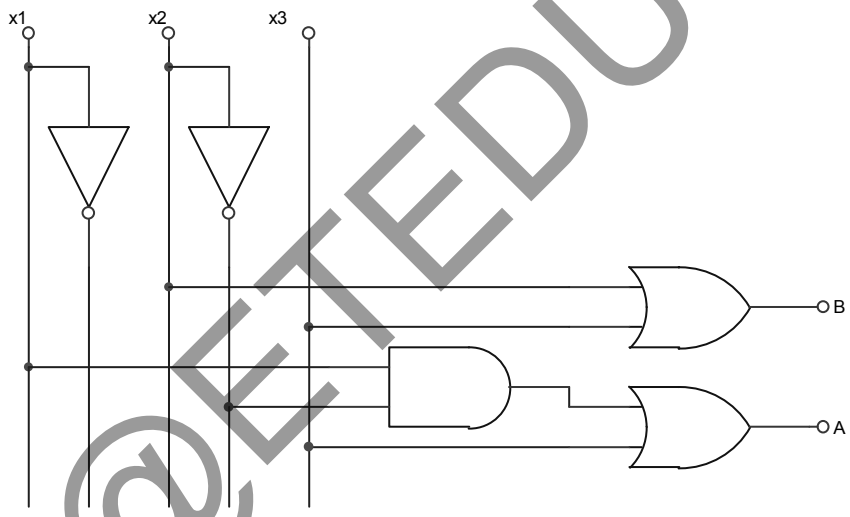
Bảng trạng thái

x_0	x_1	x_2	x_3	B	A
1	0	0	0	0	0
x	1	0	0	0	1
x	x	1	0	1	0
x	x	x	1	1	1

Từ bảng trạng thái có thể viết được phương trình logic các ngõ ra A và B:

$$A = x_1.\overline{x_2}.\overline{x_3} + x_3 = x_1.\overline{x_2} + x_3$$

$$B = x_2.\overline{x_3} + x_3 = x_2 + x_3$$



Hình 4.10 Sơ đồ logic mạch mã hóa ưu tiên 4 → 2

Sơ đồ logic: hình 4.10.

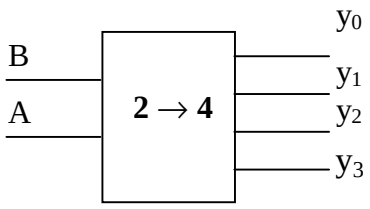
Một số vi mạch mã hóa ưu tiên thông dụng: 74LS147, 74LS148.

4.2.3. Mạch giải mã (Decoder)

1. Mạch giải mã nhị phân

Xét mạch giải mã nhị phân 2 → 4 (2 ngõ vào, 4 ngõ ra) như trên hình 4.11

Chọn mức tích cực ở ngõ ra là mức logic 1.



Hình 4.11 Mạch giải mã 2 sang 4

Baíng traûng thaïi mä taí hoaût âäüng cuía maûch

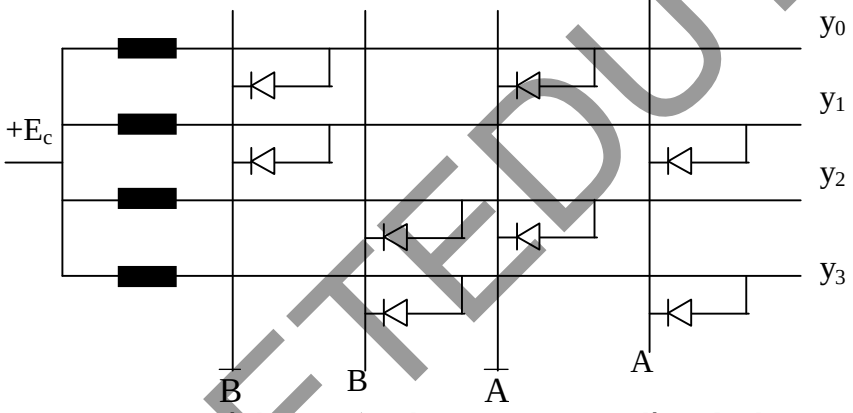
B	**A**	**y_0**	**y_1**	**y_2**	**y_3**
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Phương trình logic tối giản và sơ đồ mạch thực hiện

$$y_0 = \overline{B}.\overline{A} \quad y_1 = \overline{B}.A$$

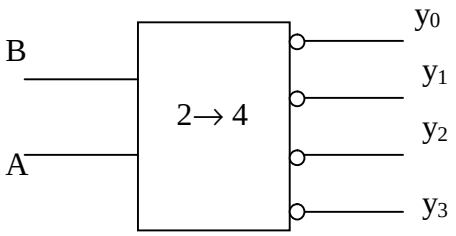
$$y_2 = B.\overline{A} \quad y_3 = A.B$$

Biểu diễn bằng cổng logic dùng Diode.



Hình 4.13. Mạch giải mã 2 → 4 dùng diode

Trường hợp chọn mức tích cực ở ngõ ra là mức logic 0 (mức logic thấp) ta có sơ đồ khối mạch giải mã được cho trên hình 4.14.



Hình 4.14. Mức tích cực ngõ ra là mức thấp

Bảng trạng thái

B	A	y_0	y_1	y_2	y_3
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

Phương trình logic:

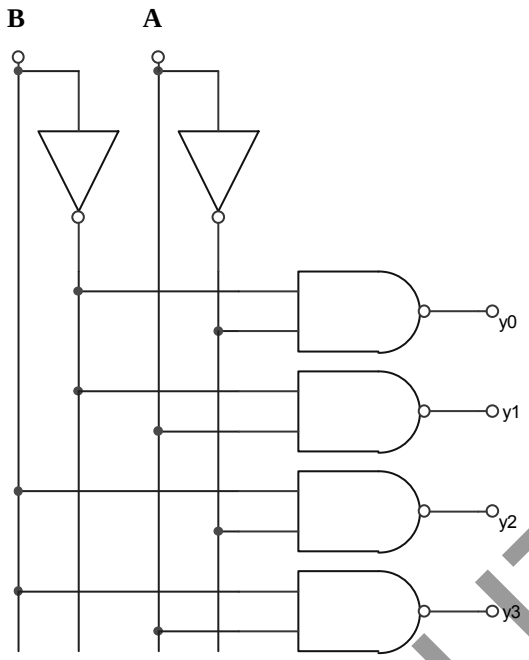
$$y_0 = B + A = \overline{\overline{B}.\overline{A}}$$

$$y_1 = B + \overline{A} = \overline{\overline{B}.A}$$

$$y_2 = \overline{B} + A = \overline{B\overline{A}}$$

$$y_3 = \overline{B} + \overline{A} = \overline{B.A}$$

Sơ đồ mạch thực hiện:



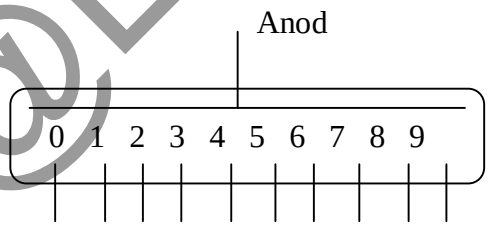
Hình 4.15. Mạch giải mã 2 → 4 với ngõ ra mức tích cực thấp

2. Mạch giải mã thập phân

a. Giải mã đèn NIXIE

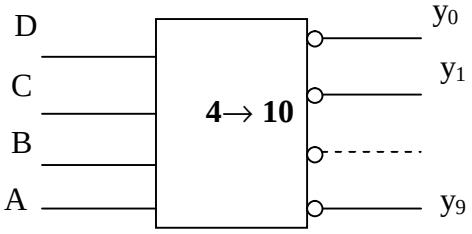
Đèn NIXIE là loại đèn điện tử loại Katod lạnh (Katod không được nung nóng bởi tim đèn), có cấu tạo gồm một Anod và 10 Katod mang hình các số từ 0 đến 9.

Sơ đồ khai triển của đèn được cho trên hình 4.16:



Hình 4.16. Sơ đồ khai triển của đèn NIXIE

Sơ đồ khối của mạch giải mã dèn NIXIE



Hình 4.17. Sơ đồ khối mạch giải mã đèn NIXIE

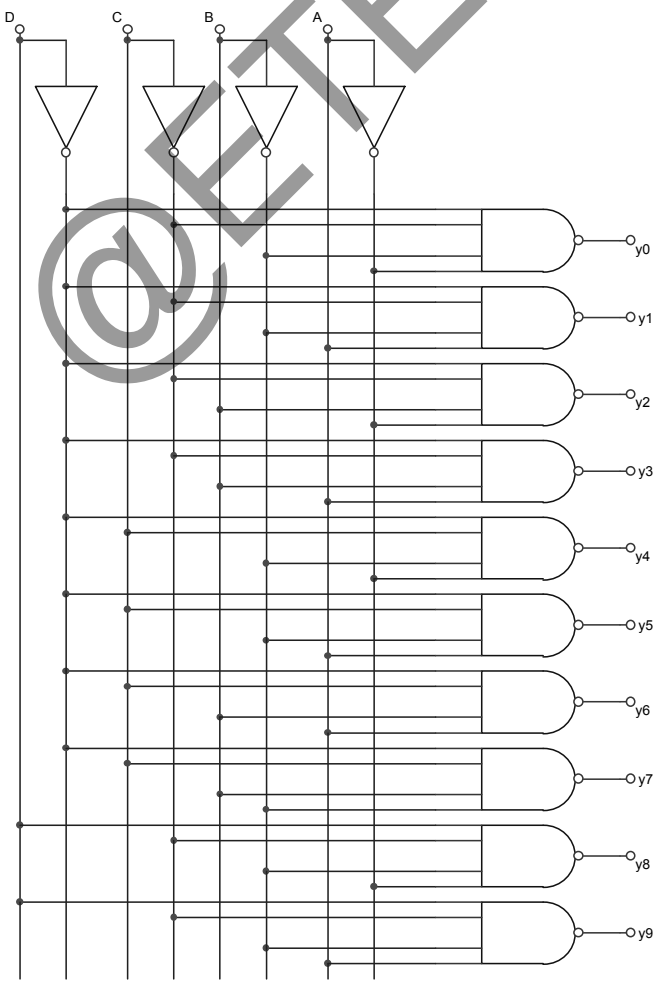
Chọn mức tích cực ở ngõ ra là mức logic 1, lúc đó bảng trạng thái hoạt động của mạch như sau:

D	C	B	A	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9
0	0	0	0	**1**	0	0	0	0	0	0	0	0	0
0	0	0	1	0	**1**	0	0	0	0	0	0	0	0
0	0	1	0	0	0	**1**	0	0	0	0	0	0	0
0	0	1	1	0	0	0	**1**	0	0	0	0	0	0
0	1	0	0	0	0	0	0	**1**	0	0	0	0	0
0	1	0	1	0	0	0	0	0	**1**	0	0	0	0
0	1	1	0	0	0	0	0	0	0	**1**	0	0	0
0	1	1	1	0	0	0	0	0	0	0	**1**	0	0
1	0	0	0	0	0	0	0	0	0	0	0	**1**	0
1	0	0	1	0	0	0	0	0	0	0	0	0	**1**

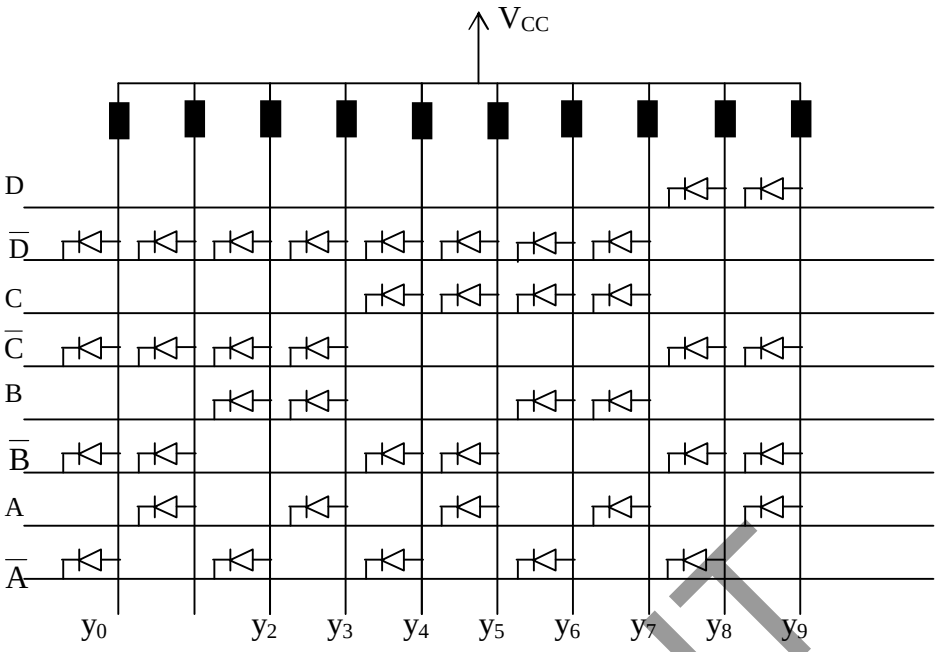
Phương trình logic:

$$y_0 = \overline{D}\overline{C}\overline{B}\overline{A} \qquad y_1 = \overline{D}\overline{C}\overline{B}A \qquad y_2 = \overline{D}\overline{C}B\overline{A} \qquad y_3 = \overline{D}\overline{C}BA$$
$$y_4 = \overline{D}C\overline{B}\overline{A} \qquad y_5 = \overline{D}C\overline{B}A \qquad y_6 = \overline{D}CB\overline{A} \qquad y_7 = \overline{D}CBA$$
$$y_8 = D\overline{C}\overline{B}\overline{A} \qquad y_9 = D\overline{C}\overline{B}A$$

Sơ đồ thực hiện mạch giải mã đèn NIXIE được cho trên hình 4.18 và 4.19:



Hình 4.18. Sơ đồ thực hiện bằng cổng logic

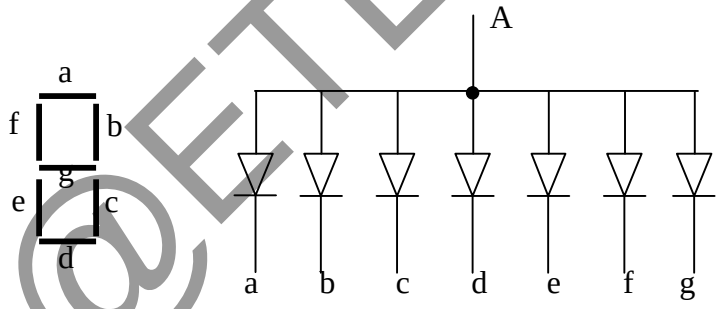


Hình 4.19. Sơ đồ thực hiện dùng diode

b. Giải mã đèn LED 7 đoạn

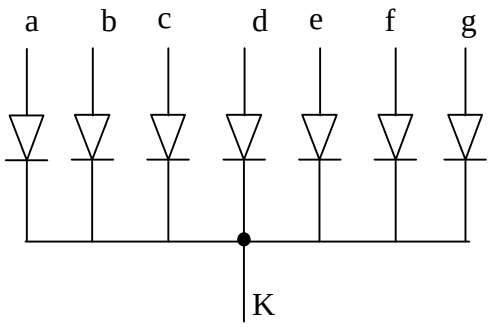
Đèn LED 7 đoạn có cấu tạo gồm 7 đoạn, mỗi đoạn là 1 đèn LED. Tuỳ theo cách nối các Kathode (Catốt) hoặc các Anode (Anốt) của các LED trong đèn, mà người ta phân thành hai loại:

LED 7 đoạn loại Anode chung:



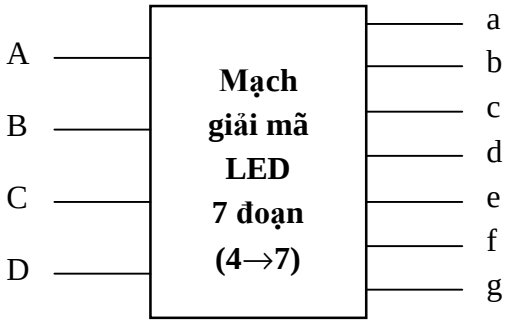
Hình 4.20. LED 7 đoạn loại Anode chung

LED 7 đoạn loại Kathode chung :



Hình 4.21. LED 7 đoạn loại Kathode chung

Ứng với mỗi loại LED khác nhau ta có một mạch giải mã riêng. Sơ đồ khối của mạch giải mã LED 7 đoạn như sau:



Hình 4.22. Sơ đồ khối mạch giải mã LED 7đoạn

Giải mã LED 7 đoạn loại Anode chung:

Đối với LED bảy đoạn loại anode chung, vì các anode của các đoạn led được nối chung với nhau và đưa lên mức logic 1 (5V), nên muốn đoạn led nào tắt ta nối kathode tương ứng lên mức logic 1 (5V) và ngược lại muốn đoạn led nào sáng ta nối kathode tương ứng xuống mass (mức logic 0).

Ví dụ: Để hiển thị số 0 ta nối kathode của đèn g lên mức logic 1 để đèn g tắt, và nối các kathode của đèn a, b, c, d, e, f xuống mass nên ta thấy số 0.

Lúc đó bảng trạng thái mô tả hoạt động của mạch giải mã LED bảy đoạn loại Anode chung như sau:

D	B	C	A	a	b	c	d	e	f	g	Số hiển thị
0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	1	0	0	1	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0	2
0	0	1	1	0	0	0	0	1	1	0	3
0	1	0	0	1	0	0	1	1	0	0	4
0	1	0	1	0	1	0	0	1	0	0	5
0	1	1	0	0	1	0	0	0	0	0	6
0	1	1	1	0	0	0	1	1	1	1	7
1	0	0	0	0	0	0	0	0	0	0	8
1	0	0	1	0	0	0	0	1	0	0	9
1	0	1	0	X	X	X	X	X	X	X	X
1	0	1	1	X	X	X	X	X	X	X	X
1	1	0	0	X	X	X	X	X	X	X	X
1	1	0	1	X	X	X	X	X	X	X	X
1	1	1	0	X	X	X	X	X	X	X	X
1	1	1	1	X	X	X	X	X	X	X	X

Dùng bảng Karnaugh để tối thiểu hóa mạch trên. Phương trình tối thiểu hóa có thể viết ở dạng chính tắc 1 (tổng của các tích số) hoặc dạng chính tắc 2 (tích của các tổng số):

Phương trình logic của ngõ ra a:

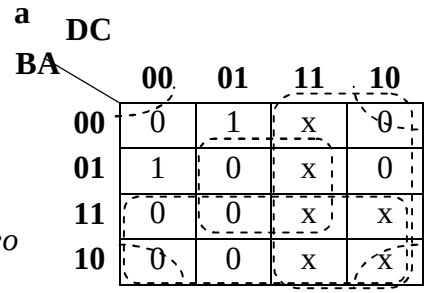
Dạng chính tắc 2:

$\mathbf{a} = \overline{B}.\overline{D}.(\overline{C}+\overline{A})(C+A) = \overline{B}\overline{C}\overline{D}A + \overline{B}\overline{D}C\overline{A}$

Dạng chính tắc 1:

$\mathbf{a} = C\overline{B}A + \overline{D}C\overline{B}A$

Lưu ý: Trên bảng Karnaugh chúng ta đã thực hiện tối thiểu hóa theo dạng chính tắc 2.



Phương trình logic của ngõ ra b:

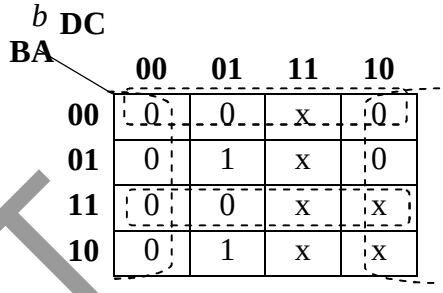
Dạng chính tắc 2:

$\mathbf{b} = .C(A+B)(\overline{A}+\overline{B}) = C(A\overline{B}+\overline{A}B)$

$= C(A \oplus B)$

Dạng chính tắc 1:

$\mathbf{b} = C\overline{B}A + CB\overline{A} = C(A \oplus B)$



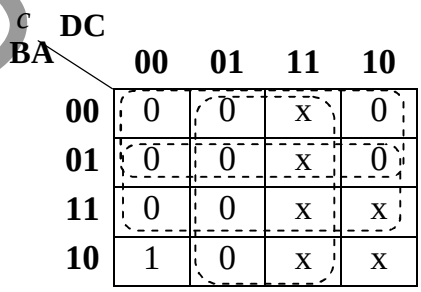
Phương trình logic của ngõ ra c:

Dạng chính tắc 2:

$\mathbf{c} = B\overline{A}\overline{C}$

Dạng chính tắc 1:

$\mathbf{c} = \overline{D}\overline{C}B\overline{A}$



Phương trình logic của ngõ ra d:

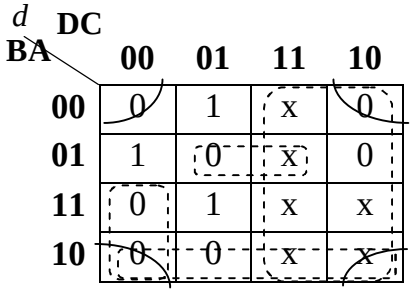
Dạng chính tắc 2:

$\mathbf{d} = \overline{D}(\overline{A}+B+\overline{C})(\overline{B}+C+D)(A+\overline{B})(A+C)$

$= \overline{A}\overline{B}C\overline{D} + ABC\overline{D} + A\overline{B}C\overline{D}$

Dạng chính tắc 1:

$\mathbf{d} = C\overline{B}\overline{A} + \overline{D}C\overline{B}A + CBA$



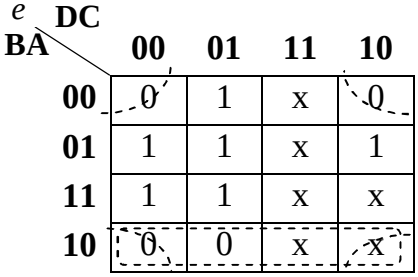
Phương trình logic của ngõ ra e:

Dạng chính tắc 2:

$\mathbf{e} = .(\overline{B}+A)(C+A)$

Dạng chính tắc 1:

$\mathbf{e} = C\overline{B} + A$



Phương trình logic của ngõ ra f:

Dạng chính tắc 2:

$$\mathbf{f} = (A+B)(B+\overline{C})(A+\overline{B}+\overline{C})\overline{D}$$
$$= AB\overline{D} + A\overline{C}\overline{D} + B\overline{C}\overline{D}$$

Dạng chính tắc 1:

$$\mathbf{f} = BA + \overline{D}\overline{C}A + \overline{D}\overline{C}B$$

f BA \ DC	00	01	11	10
00	0	0	x	0
01	1	0	x	0
11	1	1	x	x
10	1	0	x	x

Phương trình logic của ngõ ra g:

Dạng chính tắc 2:

$$\mathbf{g} = \overline{D}(A+\overline{B})(\overline{C}+B)(\overline{B}+C)$$
$$= \overline{B}\overline{C}\overline{D} + \overline{D}CBA$$

Dạng chính tắc 1:

$$\mathbf{g} = \overline{D}CBA + \overline{D}\overline{C}\overline{B}$$

g BA \ DC	00	01	11	10
00	1	0	x	0
01	1	0	x	0
11	0	1	x	x
10	0	0	x	x

Xét mạch giải mã đèn led 7 đoạn loại Kathode chung:

Chọn mức tích cực ở ngõ ra là mức logic 1. Vì Kathode của các đoạn led được nối chung và được nối xuống mức logic 0 (0V-mass) nên muốn đoạn led nào tắt ta đưa Anode tương ứng xuống mức logic 0 (0V-mass).

Ví dụ: Để hiển thị số 0 ta nối Anode của đoạn led g xuống mức logic 0 để đoạn g tắt, đồng thời các kathode của đoạn a, b, c, d, e, f được nối lên nguồn nên các đoạn này sẽ sáng do đó ta thấy số 0.

Lúc đó bảng trạng thái mô tả hoạt động của mạch như sau:

D	B	C	A	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	X	X	X	X	X	X	X
1	0	1	1	X	X	X	X	X	X	X
1	1	0	0	X	X	X	X	X	X	X
1	1	0	1	X	X	X	X	X	X	X
1	1	1	0	X	X	X	X	X	X	X
1	1	1	1	X	X	X	X	X	X	X

Tương tự như trường hợp trên, ta cũng dùng bảng Karnaugh để tối thiểu hóa hàm mạch và đi tìm phương trình logic tối giản các ngõ ra của các đoạn led: (Lưu ý trong những bảng đồ *Karnaugh sau ta thực hiện tối thiểu hóa theo dạng chính tắc 1)*

Phương trình logic của ngõ ra a:

Dạng chính tắc 1:

$$\mathbf{a} = D + B + \overline{A}\,\overline{C} + AC$$

Dạng chính tắc 2:

$$\mathbf{a} = (\overline{A} + B + C + D)(A + B + \overline{C})$$
$$= AD + B + AC + \overline{A}\,\overline{C}$$

a

BA \ DC	00	01	11	10
00	1	0	x	1
01	0	1	x	1
11	1	1	x	x
10	1	1	x	x

Phương trình logic của ngõ ra b:

Dạng chính tắc 1:

$$\mathbf{b} = \overline{C} + BA + \overline{B}\,\overline{A} = \overline{C} + \overline{A \oplus B}$$

Dạng chính tắc 2:

$$\mathbf{b} = (\overline{C} + B + \overline{A})(\overline{C} + \overline{B} + A)$$
$$= \overline{C} + AB + \overline{A}\,\overline{B} = \overline{C} + \overline{A \oplus B}$$

b

BA \ DC	00	01	11	10
00	1	1	x	1
01	1	0	x	1
11	1	1	x	x
10	1	0	x	x

Phương trình logic của ngõ ra c:

Dạng chính tắc 1:

$$\mathbf{c} = \overline{B} + A + C$$

Dạng chính tắc 2:

$$\mathbf{c} = C + \overline{B} + A$$

c

BA \ DC	00	01	11	10
00	1	1	x	1
01	1	1	x	1
11	1	1	x	x
10	0	1	x	x

Phương trình logic của ngõ ra d:

Dạng chính tắc 1:

$$\mathbf{d} = D + B\overline{A} + \overline{C}\,\overline{A} + B\overline{C} + A\overline{B}C$$

Dạng chính tắc 2:

$$\mathbf{d} = (A + B + \overline{C})(\overline{A} + \overline{B} + \overline{C})(\overline{A} + B + C + D)$$
$$= (\overline{C} + A\overline{B} + \overline{A}B)(\overline{A} + B + C + D)$$
$$= (C + A \oplus B)(\overline{A} + B + C + D)$$

d

BA \ DC	00	01	11	10
00	1	0	x	1
01	0	1	x	1
11	1	0	x	x
10	1	1	x	x

Phương trình logic của ngõ ra e:

Dạng chính tắc 1:

$$\mathbf{e} = \overline{A}.B + \overline{C}\,\overline{A}$$

Dạng chính tắc 2:

$$\mathbf{e} = \overline{A}\,(\overline{C} + B) = \overline{A}\,\overline{C} + \overline{A}.B$$

e

BA \ DC	00	01	11	10
00	1	0	x	1
01	0	0	x	0
11	0	0	x	x
10	1	1	x	x

Phương trình logic của ngõ ra f:

Dạng chính tắc 1:

$\mathbf{f} = D + C\overline{B} + \overline{B}\,\overline{A} + C\overline{A}$

Dạng chính tắc 2:

$\mathbf{f} = (\overline{B} + \overline{A})(D + C + \overline{A})(C + \overline{B})$

$= D + \overline{B}C + \overline{A}C + \overline{A}\,\overline{B}$

f BA \ DC	00	01	11	10
00	1	1	x	1
01	0	1	x	1
11	0	0	x	x
10	0	1	x	x

Phương trình logic của ngõ ra g:

Dạng chính tắc 1:

$\mathbf{g} = D + C\overline{B} + B\overline{A} + B\overline{C}$

Dạng chính tắc 2:

$\mathbf{g} = (\overline{C} + \overline{B} + \overline{A})(B + C + D)$

g BA \ DC	00	01	11	10
00	0	1	x	1
01	0	1	x	1
11	1	0	x	x
10	1	1	x	x

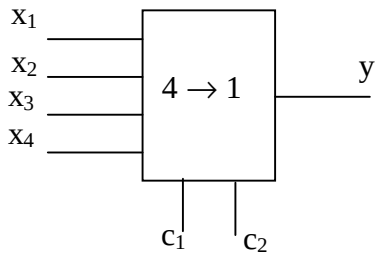
4.3. MẠCH CHỌN KÊNH - PHÂN ĐƯỜNG

4.3.1. Đại cương

Mạch chọn kênh còn gọi là mạch hợp kênh (ghép kênh) là mạch có chức năng chọn lần lượt 1 trong N kênh vào để đưa đến ngõ ra duy nhất (ngõ ra duy nhất đó gọi là đường truyền chung). Do đó, mạch chọn kênh còn gọi là mạch chuyển dữ liệu song song ở ngõ vào thành dữ liệu nối tiếp ở ngõ ra, được gọi là Multiplex (viết tắt là MUX).

Mạch chọn kênh thực hiện chức năng ở đầu phát còn mạch phân đường thực hiện chức năng ở đầu thu. Mạch phân đường còn gọi là mạch tách kênh (phân kênh, giải đa hợp), mạch này có nhiệm vụ tách N nguồn dữ liệu khác nhau ở cùng một đầu vào để rẽ ra N ngõ ra khác nhau. Do đó, mạch phân đường còn gọi là mạch chuyển dữ liệu nối tiếp ở ngõ vào thành dữ liệu song song ở ngõ ra, được gọi là Demultiplex (viết tắt là DEMUX).

4.3.2. Mạch chọn kênh



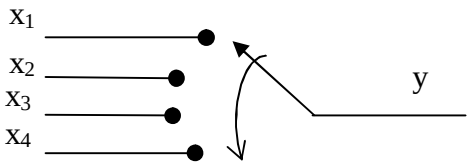
Hình 4.23a. Mạch chọn kênh

Xét mạch chọn kênh đơn giản có 4 ngõ vào và 1 ngõ ra như hình 4.23a.

Trong đó:

+ x_1, x_2, x_3, x_4 : Các kênh dữ liệu vào.
+ Ngõ ra y : Đường truyền chung.
+ c1, c2 : Các ngõ vào điều khiển

Vậy mạch này giống như 1 chuyển mạch (hình 4.23b):



Hình 4.23b

Để thay đổi lần lượt từ $x_1 \rightarrow x_4$ phải có điều khiển do đó đối với mạch chọn kênh để chọn lần lượt từ 1 trong 4 kênh vào cần có các ngõ vào điều khiển c_1, c_2. Nếu có N kênh vào thì cần có n ngõ vào điều khiển thỏa mãn quan hệ: $\mathbf{N=2^n}$. Nói cách khác: Số tổ hợp ngõ vào điều ***khiển bằng số lượng các kênh vào.***

Việc chọn dữ liệu từ 1 trong 4 ngõ vào để đưa đến đường truyền chung là tùy thuộc vào tổ hợp tín hiệu điều khiển tác động đến hai ngõ vào điều khiển c_1, c_2.

+ $c_1 = 0, c_2 = 0 \rightarrow y = x_1$ (x_1 được nối tới ngõ ra y).
+ $c_1 = 0, c_2 = 1 \rightarrow y = x_2$ (x_2 được nối tới ngõ ra y).
+ $c_1 = 1, c_2 = 0 \rightarrow y = x_3$ (x_3 được nối tới ngõ ra y).
+ $c_1 = 1, c_2 = 1 \rightarrow y = x_4$ (x_4 được nối tới ngõ ra y).

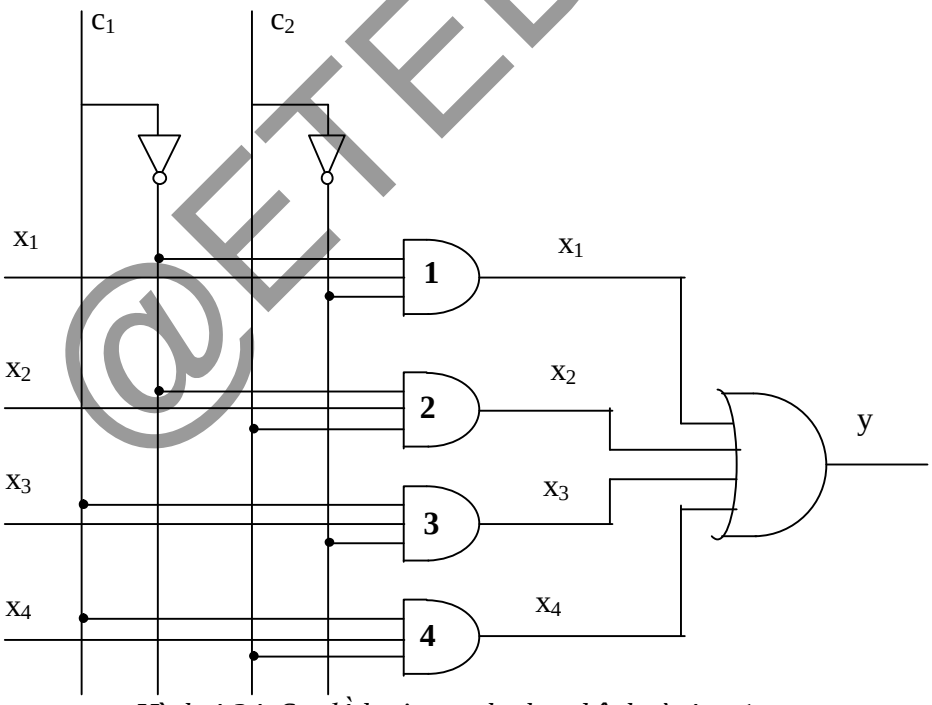
Vậy tín hiệu điều khiển phải liên tục để dữ liệu từ các kênh được liên tục đưa đến ngõ ra. Từ đó ta lập được bảng trạng thái mô tả hoạt động của mạch chọn kênh.

c_1	c_2	y
0	0	x_1
0	1	c_2
1	0	c_3
1	1	c_4

Phương trình logic mô tả hoạt động của mạch :

$$y = \overline{c_1}\,\overline{c_2}.x_1 + \overline{c_1}\,c_2.x_2 + c_1\overline{c_2}.x_3 + c_1.c_2.x_4$$

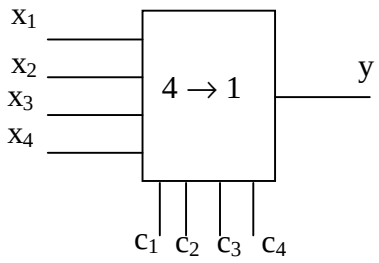
Sơ đồ logic của mạch:



Hình 4.24. Sơ đồ logic mạch chọn kênh từ 4→ 1

Bây giờ, xét mạch chọn kênh có 4 ngõ vào và 1 ngõ ra, nhưng lại có 4 ngõ điều khiển. Lúc này, ta không dựa vào tổ hợp tín hiệu tác động lên ngõ vào điều khiển, mà chỉ xét đến mức tích cực ở ngõ vào điều khiển. Ta sẽ chọn một trong hai mức logic 1 hoặc mức logic 0 làm mức tích cực, nếu 1 ngõ vào trong số 4 ngõ vào điều khiển tồn tại mức logic tích cực (mức 1 hoặc mức 0) thì kênh dữ liệu vào có cùng chỉ số với ngõ vào điều khiển đó sẽ được kết nối với ngõ ra. Trên hình 4.25 biểu diễn mạch chọn kênh với số lượng ngõ vào điều khiển bằng số lượng kênh vào.

Nếu chọn mức tích cực của các ngõ vào điều khiển là mức logic 1, ta có bảng trạng thái mô tả hoạt động của mạch như sau:



Hình 4.25. Mạch chọn kênh với số lượng ngõ vào điều khiển bằng số kênh vào

c_1	c_2	c_3	c_4	y
1	0	0	0	x_1
0	**1**	0	0	x_2
0	0	**1**	0	x_3
0	0	0	**1**	x_4

Phương trình logic:

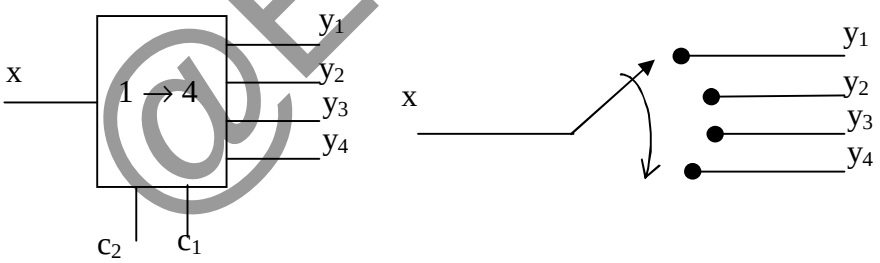
$$y = c_1. x_1 + c_2. x_2 + c_3. x_3 + c_4. x_4$$

Ý nghĩa trong thực tế của mạch:

+ c_1, c_2, c_3, c_4 : Có thể hiểu là các địa chỉ (nguồn và đích).
+ x_1, x_2, x_3, x_4 : Thông tin cần truyền đi.

4.3.3. Mạch phân đường

Xét mạch phân đường đơn giản có 1 ngõ vào và 4 ngõ ra ký hiệu như sau :



Hình 4.26. Mạch phân đường đơn giản từ 1 → 4

Trong đó:

+ x là kênh dữ liệu vào.
+ y1, y2, y3, y4 các ngõ ra dữ liệu; c1, c2 các ngõ vào điều khiển.

Ta có thể thấy mạch này thực hiện chức năng như 1 chuyển mạch (hình vẽ 4.26).

Tùy thuộc vào tổ hợp tín hiệu điều khiển tác dụng vào mạch mà lần lượt tín hiệu từ ngõ vào x sẽ chuyển đến ngõ ra y1, y2, y3, y4 một cách tương ứng.

Lúc đó bảng trạng thái mô tả hoạt động của mạch :

c_1	c_2	y_1	y_2	y_3	y_4
0	0	**x**	0	0	0
0	1	0	**x**	0	0
1	0	0	0	**x**	0
1	1	0	0	0	**x**

Phương trình logic các ngõ ra:

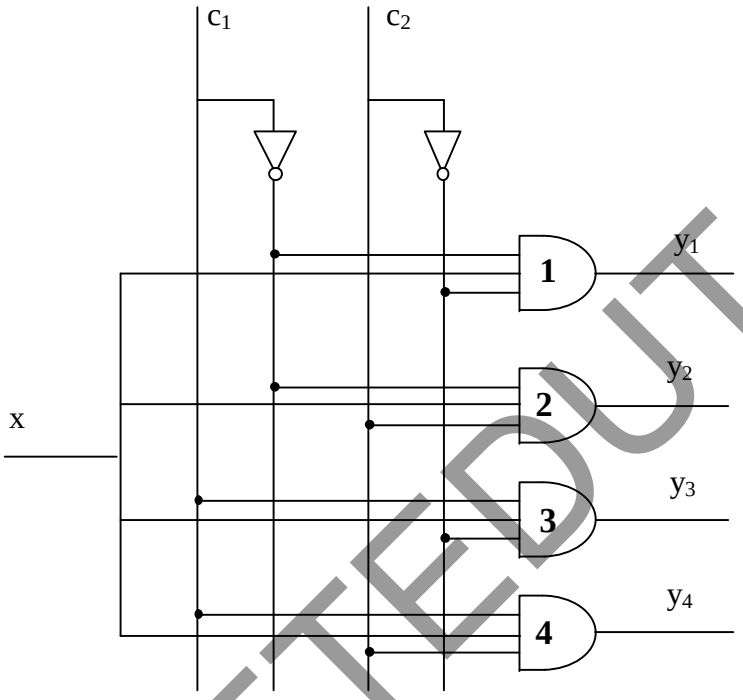
$y_1 = \overline{c_1}\ \overline{c_2}.x$

$y_2 = \overline{c_1}\, c_2.x$

$y_3 = c_1 \overline{c_2}.x$

$y_4 = c_1\, c_2.x$

Sơ đồ logic được cho trên hình 4.27:

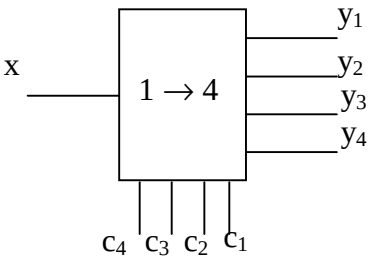


Hình 4.27. Sơ đồ logic thực hiện mạch phân đường

Nếu x = 1 và hoán đổi ngõ vào điều khiển thành ngõ vào dữ liệu thì mạch phân đường chuyển thành mạch giải mã nhị phân. Vì vậy, nhà sản xuất đã chế tạo IC đảm bảo cả hai chức năng: giải mã và giải đa hợp (Decode/Demultilex). Ví dụ: các IC 74138, 74139, 74154: giải mã và phân đường tùy thuộc vào cách nối chân.

Trong trường hợp tổng quát, mạch phân đường có 1 ngõ vào và 2^n ngõ ra: để tách $N=2^n$ nguồn dữ liệu khác nhau cần có n ngõ vào điều khiển, lúc đó số tổ hợp ngõ vào điều khiển bằng số lượng ngõ ra.

Tuy nhiên trong thực tế, ta còn gặp mạch phân đường có số lượng ngõ vào điều khiển bằng số ngõ ra (hình 4.28). Lúc đó chỉ xét đến mức tích cực ở ngõ vào điều khiển, người ta chọn một trong hai mức logic 1 hoặc mức logic 0 làm mức tích cực. Giả sử chọn mức logic 1 là mức tích cực: nếu 1 ngõ vào trong số 4 ngõ vào điều khiển tồn tại mức logic 1 (mức tích cực), thì ngõ ra dữ liệu tương ứng có cùng chỉ số với ngõ vào điều khiển đó sẽ được nối với ngõ vào dữ liệu chung x.



Hình 4.28

Ví dụ:

$c_1 = 1 \rightarrow x = y_1$ $\quad c_2 = 1 \rightarrow x = y_2$

$c_3 = 1 \rightarrow x = y_3$ $\quad c_4 = 1 \rightarrow x = y_4$

Lúc đó bảng trạng thái hoạt động của mạch:

c_1	c_2	c_3	c_4	y_1	y_2	y_3	y_4
1	0	0	0	X	0	0	0
0	1	0	0	0	X	0	0
0	0	1	0	0	0	X	0
0	0	0	1	0	0	0	X

Phương trình logic và sơ đồ logic được cho trên hình 4.29:

$y_1 = c_1 x$ $y_2 = c_2 x$

$y_3 = c_3 x$ $y_4 = c_4 x$

Giải thích hoạt động của mạch:

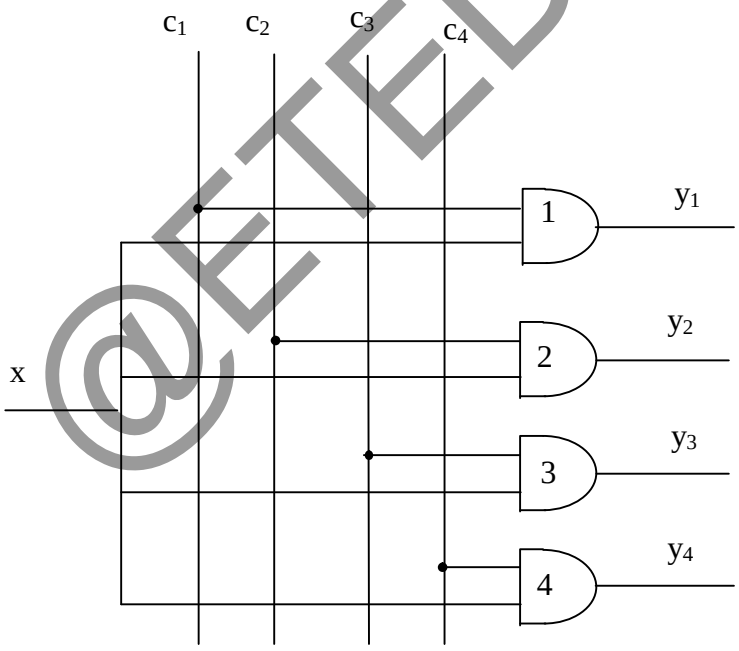
+ Khi c_1=1, c_2= c_3 = c_4 = 0 chỉ có cổng AND(1) thông cho dữ liệu từ x nối đến đầu ra y_1.

+ Khi c_2=1, c_1= c_3 = c_4 = 0 chỉ có cổng AND(2) thông cho dữ liệu từ x nối đến đầu ra y_2.

+ Khi c_3=1, c_2 = c_1= c_4 = 0 chỉ có cổng AND(3) thông cho dữ liệu từ x nối đến đầu ra y_3.

+ Khi c_4=1, c_2= c_3 = c_1= 0 chỉ có cổng AND(4) thông cho dữ liệu từ x nối đến đầu ra y_4.

Vì mạch chọn kênh được thực hiện ở đầu phát và mạch phân đường được thực hiện ở đầu thu nên để đảm bảo dữ liệu được chuyển đúng kênh thì mạch chọn kênh và mạch phân đường phải đồng bộ với nhau.



Hình 4.29. Mạch phân đường số lượng ngõ vào điều khiển bằng số ngõ ra

4.4. MẠCH SO SÁNH

4.4.1. Đại cương

- Mạch so sánh dùng để so sánh các số nhị phân về mặt độ lớn.
  Ví dụ: So sánh a và b: a = 0, b = 1 (a< b.
- Có hai mạch so sánh:
  + So sánh hai số nhị phân 1 bit.
  + So sánh hai số nhị phân nhiều bit.

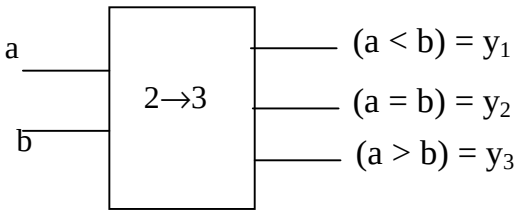
4.4.2. Mạch so sánh 1 bit

Là mạch thực hiện chức năng so sánh hai số nhị phân 1 bit.

Xét hai số nhị phân 1 bit a và b. Có các trường hợp sau đây:

+ a = 0, b = 0 ⇒ a = b.
+ a = 1, b = 1 ⇒ a = b.
+ a = 0, b = 1 ⇒ a < b.
+ a = 1, b = 0 ⇒ a > b.

Về phương diện mạch điện, mạch so sánh 1 bit có 2 ngõ vào và 3 ngõ ra. Các ngõ vào a, b là các bít cần so sánh; các ngõ ra thể hiện kết quả so sánh: y1 (a < b), y2 (a=b) và y3 (a > b). Sơ đồ khối mạch so sánh trên hình 4.30.



Hình 4.30. Mạch so sánh 1 bit

Bảng trạng thái

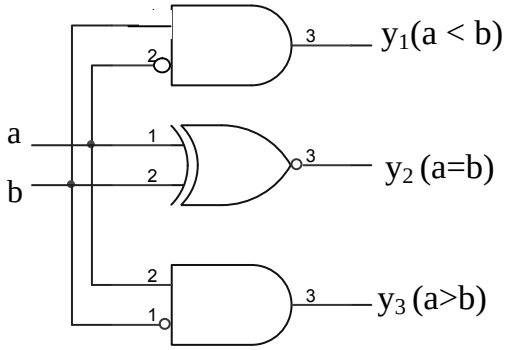
a	b	y_1	y_2	y_3
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

Chọn mức tích cực ở ngõ ra là mức logic 1. Ta lập được bảng trạng thái mô tả hoạt động của mạch. Từ bảng trạng thái, ta có phương trình logic:

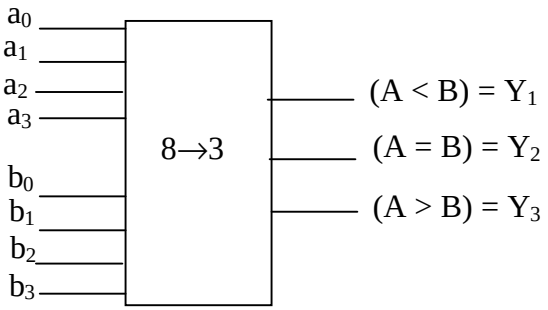
$y_1 = \overline{a}.b$

$y_2 = \overline{a}.\overline{b} + a.b = \overline{a \oplus b}$

$y_3 = a.\overline{b}$



Hình 4.31. Sơ đồ mạch so sánh 1 bit



Hình 4.32. Sơ đồ khối mạch so sánh nhiều bit

4.4.3. Mạch so sánh nhiều bit

Mạch có 8 ngõ vào và 3 ngõ ra, thực hiện so sánh 2 số nhị phân 4 bít A ($a_3a_2a_1a_0$) và B ($b_3b_2b_1b_0$). Có hai phương pháp thực hiện mạch so sánh nhiều bít:

- Thực hiện trực tiếp.
- Thực hiện mạch so sánh nhiều bít trên cơ sở mạch so sánh 1 bít.

Chúng ta lần lượt xét từng phương pháp.

1. Phương pháp trực tiếp

Ta có bảng trạng thái hoạt động của mạch

INPUT				OUTPUT		
a_3 và b_3	a_2 và b_2	a_1 và b_1	a_0 và b	A < B	A = B	A > B
<	x	x	x	1	0	0
>	x	x	x	0	0	1
=	<	x	x	1	0	0
=	>	x	x	0	0	1
=	=	<	x	1	0	0
=	=	>	x	0	0	1
=	=	=	<	1	0	0
=	=	=	>	0	0	1
=	=	=	=	0	1	0

Phương trình logic của mạch:

$Y_1 = (A < B)$

$= (a_3 < b_3) + (a_3 = b_3)(a_2 < b_2) + (a_3 = b_3)(a_2 = b_2)(a_1 < b_1)$

$+ (a_3 = b_3)(a_2 = b_2)(a_1 = b_1)(a_0 < b_0)$

$Y_2 = (A = B)$

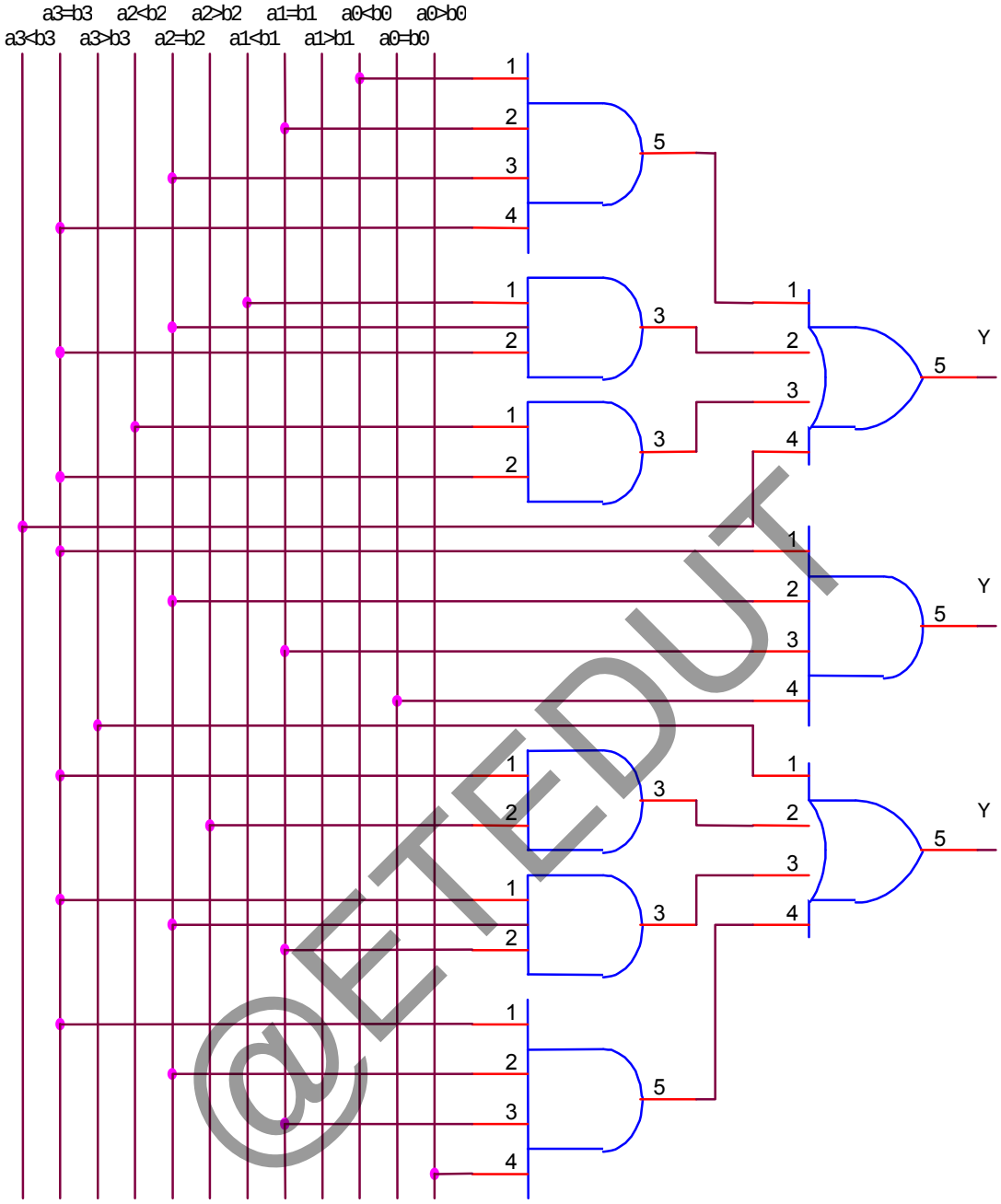
$= (a_3 = b_3)(a_2 = b_2) (a_1 = b_1)(a_0 = b_0)$

$Y_3 = (A > B)$

$= (a_3 > b_3) + (a_3 = b_3)(a_2 > b_2) + (a_3 = b_3)(a_2 = b_2)(a_1 > b_1)$

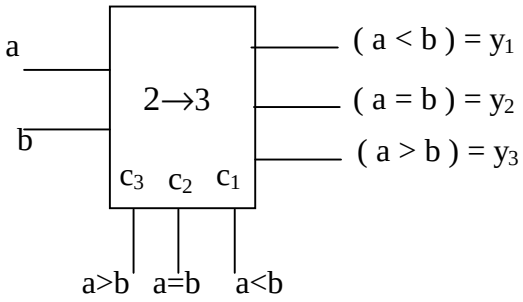
$+ (a_3 = b_3)(a_2 = b_2)(a_1 = b_1)(a_0 > b_0).$

Sơ đồ mạch thực hiện trên hình 4.33.



Hình 4.33. Thực hiện mạch so sánh nhiều bít theo cách trực tiếp

2. Phương pháp xây dựng trên cơ sở mạch so sánh 1 bit



Hình 4.34. Mạch so sánh 1 bít cải tiến

Để mạch so sánh hai số nhị phân 1 bit có thể thực hiện công việc xây dựng mạch so sánh hai số nhị phân nhiều bit ta cải tiến lại mạch so sánh 1 bit như sau: ngoài các ngõ vào và ngõ ra giống như mạch so sánh 1 bit ta đã khảo sát ở trên, còn có các ngõ vào điều khiển a< b, a> b, a = b, với sơ đồ mạch như sau :

Bảng trạng thái mô tả hoạt động của mạch so sánh nhị phân 1 bit đầy đủ như sau:

Ngõ vào điều khiển			Ngõ vào DATA		Ngõ ra		
a<b	a=b	a>b	a	b	(a<b)	(a=b)	(a>b)
1	0	0	x	x	1	0	0
0	0	1	x	x	0	0	1
0	1	0	0	0	0	1	0
0	1	0	0	1	1	0	0
0	1	0	1	0	0	0	1
0	1	0	1	1	0	1	0

Phương trình logic:

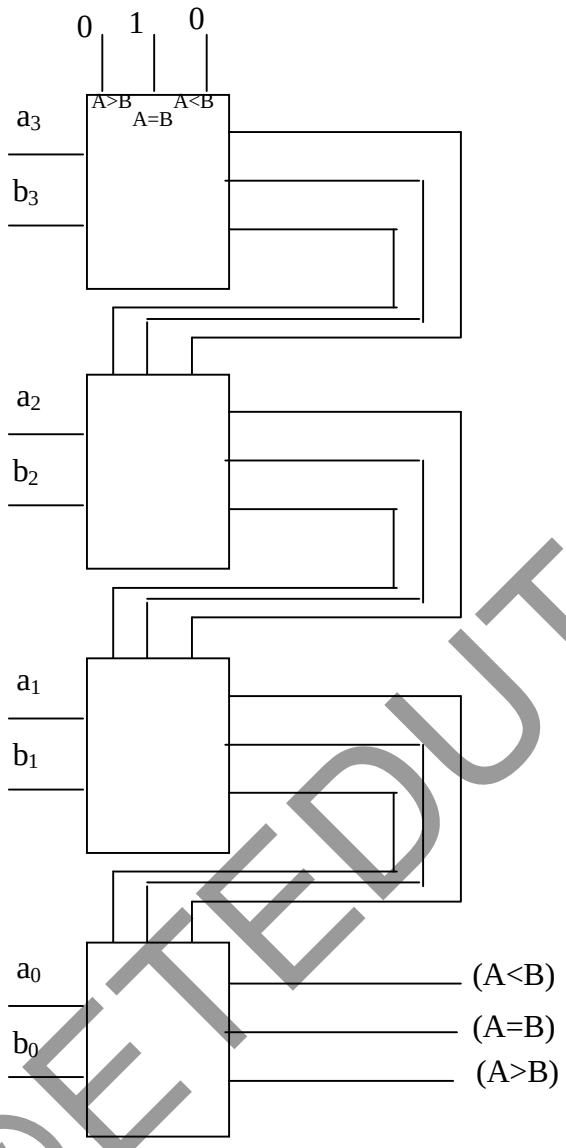
$y_1 = (a<b) = c_1 + c_2(\overline{a}\, b)$.

$y_2 = (a=b) = c_2(\overline{a \oplus b})$.

$y_3 = (a>b) = c_3 + c_2(a\overline{b})$.

Dựa vào vi mạch so sánh đầy đủ này, người ta thực hiện mạch so sánh hai số nhị phân 4 bit bằng cách sử dụng các vi mạch so sánh 1 bit đầy đủ này gữa a_3 với b_3, a_2 với b_2, a_1 với b_1, a_0 với b_0 với cách nối theo sơ đồ như trên hình 4.35.

<u>Lưu ý đối với mạch trên hình 4.35</u>: mạch có 3 ngõ vào điều khiển (A>B), (A=B), (A<B) nên để mạch làm việc được thì bắt buộc cho ngõ vào điều khiển (A=B) = 1 (tức là xem như a_4, a_4 trở về trước bằng nhau, nếu $a_4 > a_4$ thì ngõ ra A>B).



Hình 4.35. Mạch so sánh nhiều bít

4.5. MẠCH SỐ HỌC

4.5.1. Đại cương

Mạch số học là mạch có chức năng thực hiện các phép toán số học +, -, x, / các số nhị phân. Đây là cơ sở để xây dựng đơn vị luận lý và số học (ALU) trong µp (µicro Processor) hoặc CPU (Centre Processing Unit).

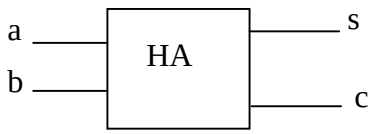
4.5.2. Bộ cộng (Adder)

1. Bộ bán tổng (HA-Half Adder)

Bộ bán tổng thực hiện cộng 2 số nhị phân một bít.

Quy tắc cộng như sau:

0 + 0 = 0 nhớ 0
0 + 1 = 1 nhớ 0
1 + 0 = 1 nhớ 0
1 + 1 = 0 nhớ 1
(a) (b) (s) (c)



Hình 4.36. Mạch cộng 1 bít

Trong đó a, b là số cộng, s là tổng, c là số nhớ.

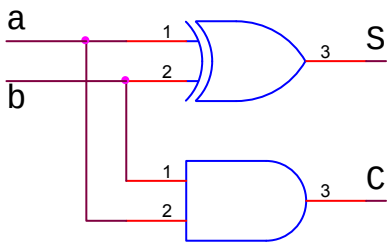
Bảng trạng thái mô tả hoạt động của mạch và phương trình logic:

$s = a.\bar{b} + \bar{a}.b = a \oplus b$

$c = a.b$

a	b	s	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Mạch cộng này chỉ cho phép cộng hai số nhị phân 1 bit mà không thực hiện cộng hai số nhị phân nhiều bit.

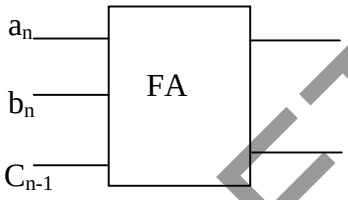


Hình 4.37. Sơ đồ mạch cộng bán phần

2.Bộ tổng (Bộ cộng toàn phần - FA: Full Adder)

Về phương diện mạch có sơ đồ khối như sau:

Trong đó:



Hình 4.38. Bộ cộng toàn phần

a_n	b_n	C_{n-1}	S_n	C_n
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

+ C_{n-1} : Số nhớ của lần cộng trước đó.
+ C_n : Số nhớ của lần cộng hiện tại.
+ S_n : Tổng hiện tại.

Từ bảng trạng thái mô tả hoạt động của mạch ta viết được phương trình logic:

$S_n = f(a_n, b_n, C_{n-1})$

$C_n = f(a_n, b_n, C_{n-1})$

S_n	a_nb_n 00	01	11	10
C_{n-1} 0	0	1	0	1
1	1	0	1	0

C_n	a_nb_n 00	01	11	10
C_{n-1} 0	0	0	1	0
1	0	1	1	1

$$S_n = \overline{a_n}\,\overline{b_n}C_{n-1} + \overline{a_n}b_n\overline{C_{n-1}} + a_n\overline{b_n}\,\overline{C_{n-1}} + a_nb_nC_{n-1}$$

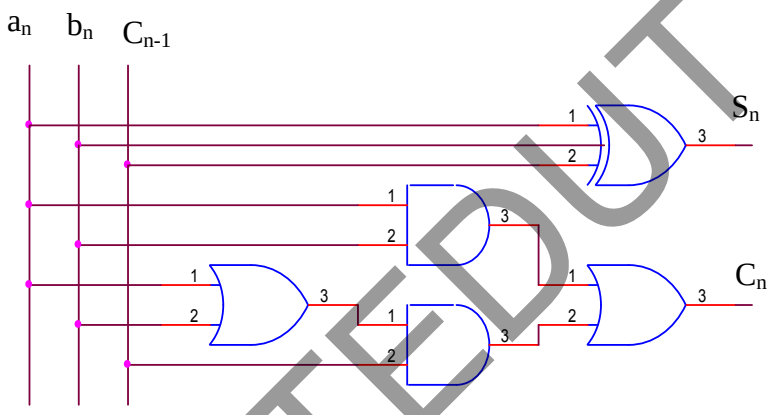
$$S_n = a_n \oplus b_n \oplus C_{n-1}$$

$$C_n = \overline{a_n}C_{n-1} + b_nC_{n-1} + a_nb_n$$

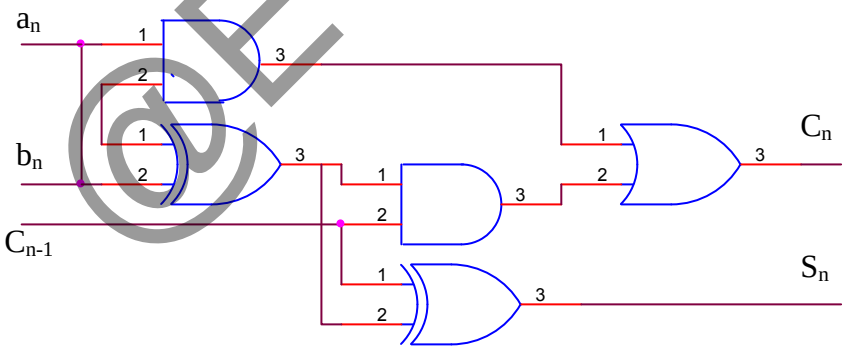
$$C_n = a_nb_n + C_{n-1}(a_n + b_n)$$

Lập bảng Karnaugh và tối thiểu hóa, ta có:

Có thể thực hiện trực tiếp (sơ đồ 4.39) hoặc sử dụng HA để thực hiện FA (sơ đồ 4.40):



Hình 4.39. Mạch cộng toàn phần trực tiếp



Hình 4.40. Thực hiện mạch cộng toàn phần từ bộ bán tổng

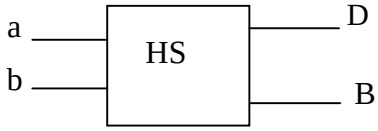
4.5.3. Bộ trừ (Subtractor)

1. Bộ bán trừ (Bộ trừ bán phần - HS: Half subtractor)

Bộ bán trừ thực hiện trừ 2 số nhị phân 1 bit.

Quy tắc trừ như sau:

0	- 0 =	0	mượn	0
0	- 1 =	1	mượn	1
1	- 0 =	1	mượn	0
1	- 1 =	0	mượn	0
(a)	(b)	(D)		(B)

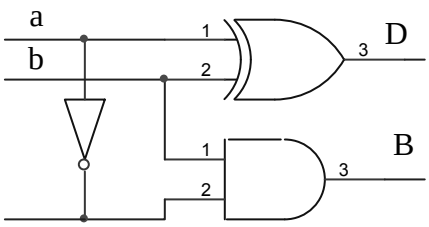


Hình 4.41 Mạch trừ bán phần

Trong đó a là số bị trừ, b là số trừ, D là hiệu, B là số mượn.

Bảng trạng thái mô tả hoạt động :

a	b	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



Hình 4.42. Sơ đồ logic

Phương trình logic :

$D = a.\overline{b} + \overline{a}.b = a \oplus b$

$B = \overline{a}.b$

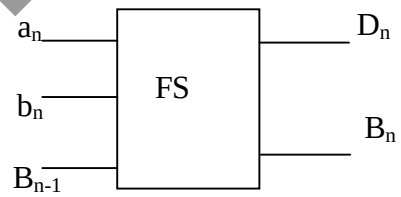
Mạch trừ này chỉ cho phép trừ hai số nhị phân 1 bit mà không thực hiện việc trừ hai số nhị phân nhiều bit.

2. Bộ trừ toàn phần (FS - Full Subtractor)

Mạch có sơ đồ khối và bảng trạng thái mô tả hoạt động như sau:

Trong đó: Bn-1 : Số mượn của lần trừ trước đó.
Bn : Số mượn của lần trừ hiện tại.
Dn : Hiệu số hiện tại.

a_n	b_n	B_{n-1}	D_n	B_n
0	0	0	0	0
0	1	0	1	1
1	0	0	1	0
1	1	0	0	0
0	0	1	1	1
0	1	1	0	1
1	0	1	0	0
1	1	1	1	1



Hình 4.43. Mạch trừ toàn phần

Lập bảng Karnaugh và tối thiểu hóa, ta có:

D_n	a_nb_n 00	01	11	10
B_{n-1} 0	0	1	0	1
1	1	0	1	0

$$D_n = \overline{a_n}\,\overline{b_n}B_{n-1} + \overline{a_n}b_n\overline{B_{n-1}} + a_n\overline{b_n}\,\overline{B_{n-1}} + a_nb_nB_{n-1}$$

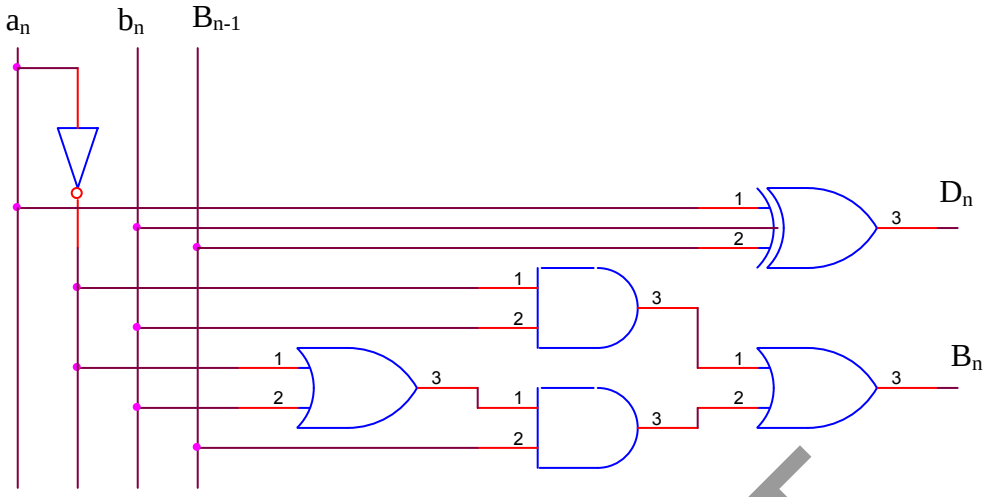
$$D_n = a_n \oplus b_n \oplus B_{n-1}$$

B_n	a_nb_n 00	01	11	10
B_{n-1} 0	0	1	0	0
1	1	1	1	0

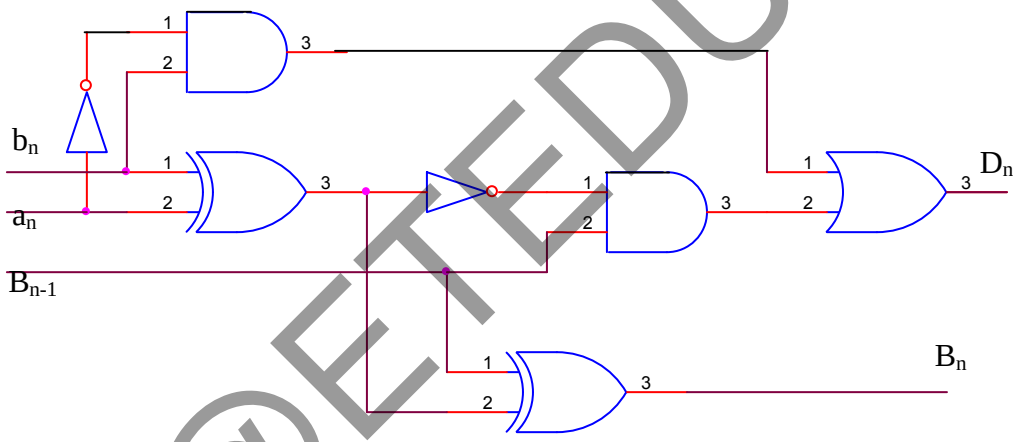
$$B_n = \overline{a_n}B_{n-1} + b_nB_{n-1} + \overline{a_n}b_n$$

$$B_n = a_nb_n + B_{n-1}(a_n + b_n)$$

Có 2 cách thực hiện bộ trừ toàn phần theo biểu thức logic đã tìm được: hoặc thực hiện trực tiếp (hình 4.44) hoặc sử dụng HS để thực hiện FS (hình 4.45).



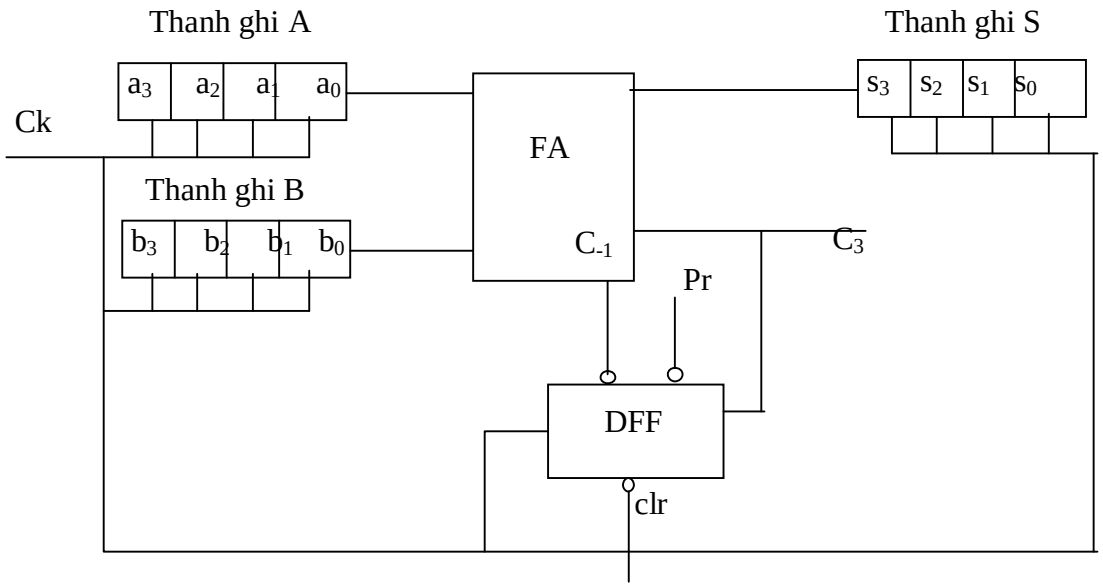
Hình 4.44. Thực hiện mạch trừ toàn phần trực tiếp



Hình 4.45. Thực hiện FS trên cơ sở HS

Từ bộ cộng toàn phần, ta xây dựng mạch cộng hai số nhị phân nhiều bit bằng 2 phương pháp: Nối tiếp và Song Song.

Phương pháp nối tiếp:



Hình 4.46. Mạch cộng 2 số nhị phân nhiều bit theo theo kiểu nối tiếp

Thanh ghi A chứa số A : a_3, a_2, a_1, a_0

Thanh ghi B chứa số B : b_3, b_2, b_1, b_0

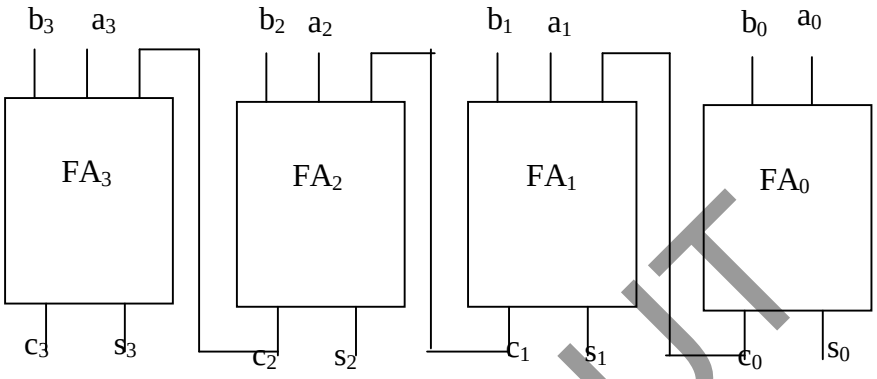
Thanh ghi S chứa số S : s_3, s_2, s_1, s_0

Nhược điểm của phương pháp này là thời gian thực hiện lâu.

Phương pháp song song:

Để khắc phục nhược điểm đó, người ta dùng phương pháp cộng song song (hình 4.47).

Do tín hiệu điều khiển Ck (điều khiển cộng) đồng thời nên thời gian thực hiện phép cộng nhanh hơn phương pháp nối tiếp, song do số nhớ vẫn phải chuyển nối tiếp nên ảnh hưởng tốc độ xử lý.



Hình 4.47. Mạch cộng song song, số nhớ chuyển nối tiếp

Mạch cộng nhớ nhanh - Mạch cộng với số nhớ nhìn thấy trước:

Người ta cải tiến mạch trên thành mạch cộng song song với số nhớ nhìn thấy trước còn gọi là mạch cộng nhớ nhanh (Fast Carry, Carry Look Ahead). Bằng cách dựa vào sự phân tích mạch cộng toàn phần như sau:

Ta có:

$$S_n = (a_n \oplus b_n) \oplus C_{n-1}$$

$$C_n = a_n . b_n + (a_n \oplus b_n) C_{n-1}$$

Ta âàût:

$$P_n = a_n \oplus b_n$$

$$G_n = a_n . b_n$$

Suy ra:

$$\mathbf{S_n = P_n \oplus C_{n-1}}$$

$$\mathbf{C_n = G_n + P_n . C_{n-1}}$$

Khi n= 0 (LSB):

$$S_0 = P_0 \oplus C_{-1}$$

$$C_0 = G_0 + P_0 . C_{-1}$$

Khi n=1:

$$S_1 = P_1 \oplus C_0 = P_1 \oplus (G_0 + P_0 . C_{-1})$$

$$C_1 = G_1 + P_1 . C_0 = G_1 + P_1 . (G_0 + P_0 . C_{-1})$$

Khi n=2:

$$S_2 = P_2 \oplus C_1 = P_2 \oplus [G_1 + P_1 . (G_0 + P_0 . C_{-1})]$$

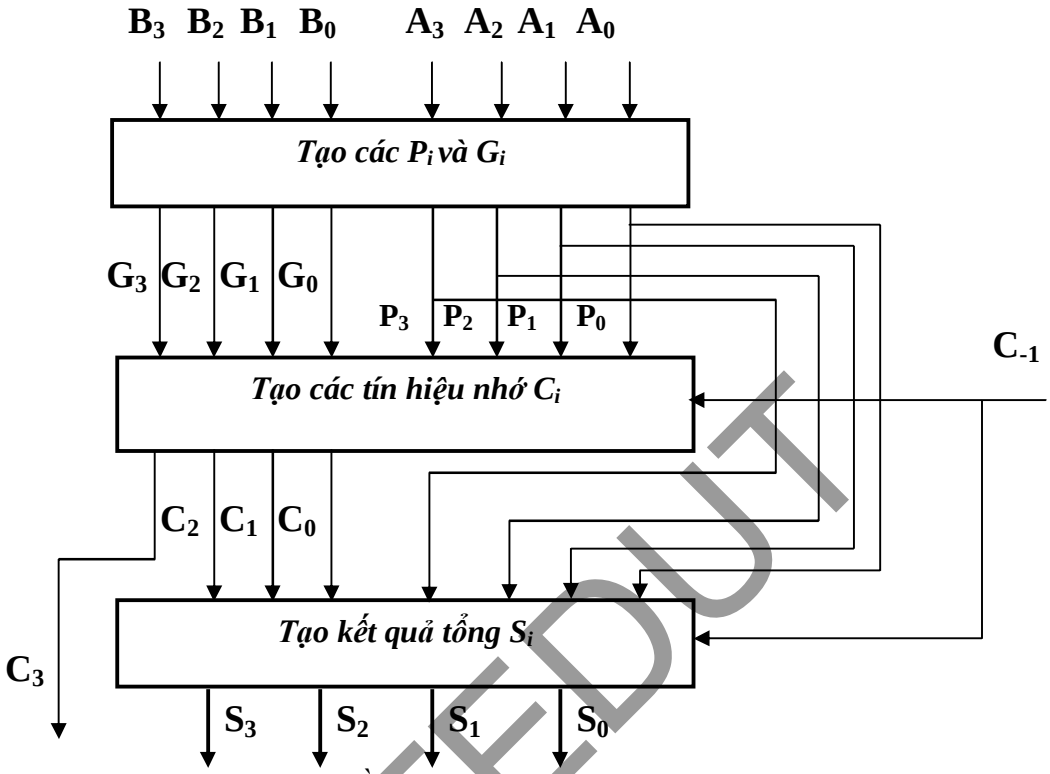
$$C_2 = G_2 + P_2 . C_1 = G_2 + P_2 . [G_1 + P_1 . (G_0 + P_0 . C_{-1})]$$

Khi n=3:

$$S_3 = P_3 \oplus C_2 = P_3 \oplus \{G_2 + P_2 . [G_1 + P_1 . (G_0 + P_0 . C_{-1})]\}$$

$$C_3 = G_3 + P_3 . C_2 = G_3 + P_3 . \{G_2 + P_2 . [G_1 + P_1 . (G_0 + P_0 . C_{-1})] \}$$

Đây chính là cơ sở tính toán để tạo ra số nhớ C1, C2, C3 và S3 tùy thuộc vào an, bn. Sơ đồ khối mạch cộng song song 4 bít nhớ nhanh được cho trên hình 4.48



Hình 4.48. Sơ đồ mạch cộng song song 4 bít nhớ nhanh

Trên thực tế người ta đã chế tạo ra các vi mạch cộng nhớ nhanh, ví dụ: IC 7483.

Chương 5

HỆ TUẦN TỰ

5.1. KHÁI NIỆM CHUNG

Mạch số được chia thành hai loại chính : Hệ tổ hợp và hệ tuần tự.

Đối với hệ tổ hợp: tín hiệu ngõ ra ở trạng thái kế tiếp chỉ phụ thuộc vào trạng thái hiện tại của ngõ vào, mà bất chấp trạng thái hiện tại của ngõ ra. Như vậy, khi các ngõ vào thay đổi trạng thái (bỏ qua thời gian trễ của tín hiệu đi qua phần tử logic) thì lập tức ngõ ra thay đổi trạng thái.

Đối với hệ tuần tự: Các ngõ ra ở trạng thái kế tiếp vừa phụ thuộc vào trạng thái hiện tại của ngõ vào, đồng thời còn phụ thuộc trạng thái hiện tại của ngõ ra.

Do đó, vấn đề thiết kế hệ tuần tự sẽ khác so với hệ tổ hợp và cơ sở thiết kế hệ tuần tự là dựa trên các Flip - Flop (trong khi việc thiết kế hệ tổ hợp dựa trên các cổng logic).

Mặûc khác, đối với hệ tuần tự, khi các ngõ vào thay đổi trạng thái thì các ngõ ra không thay đổi trạng thái ngay mà chờ đến cho đến khi có một xung điều khiển (gọi là xung đồng hồ Ck) thì lúc đó các ngõ ra mới thay đổi trạng thái theo các ngõ vào. Như vậy hệ tuần tự còn có tính đồng bộ và tính nhớ (có khả năng lưu trữ thông tin, lưu trữ dữ liệu), nên hệ tuần tự là cơ sở để thiết kế các bộ nhớ.

5.2. BỘ ĐẾM

5.2.1. Đại cương

Bộ đếm được xây dựng trên cơ sở các Flip - Flop (FF) ghép với nhau sao cho hoạt động theo một bảng trạng thái (qui luật) cho trước.

Số lượng FF sử dụng là số hàng của bộ đếm.

Bộ đếm còn được sử dụng để tạo ra một dãy địa chỉ của lệnh điều kiển, đếm số chu trình thực hiện phép tính, hoặc có thể dùng trong vấn đề thu và phát mã.

Có thể phân loại bộ đếm theo nhiều cách:

- **Phân loại theo cơ sở các hệ đếm:** Bộ đếm thập phân, bộ đếm nhị phân.
  Trong đó bộ đếm nhị phân được chia làm hai loại:
  + Bộ đếm với dung lượng đếm 2n.
  + Bộ đếm với dung lượng đếm khác 2n (đếm modulo M).
- **Phân loại theo hướng đếm gồm:** Mạch đếm lên (đếm tiến), mạch đếm xuống (đếm lùi), mạch đếm vòng.
- **Phân loại mạch đếm theo tín hiệu chuyển: bộ đếm nối tiếp, bộ đếm song song,** bộ đếm hỗn hợp.
- **Phân loại dựa vào chức năng điều khiển:**
  + Bộ đếm đồng bộ: Sự thay đổi ngõ ra phụ thuộc vào tín hiệu điều kiển Ck.
  + Bộ đếm không đồng bộ.

Mặc dù có rất nhiều cách phân loại nhưng chỉ có ba loại chính: **Bộ đếm nối tiếp** (không đồng bộ), **Bộ đếm song song** (đồng bộ), **Bộ đếm hỗn hợp**.

5.2.2. Bộ đếm nối tiếp

1. Khái niệm

Bộ đếm nối tiếp là bộ đếm trong đó các TFF hoặc JKFF giữ chức năng của TFF được ghép nối tiếp với nhau và hoạt động theo một loại mã duy nhất là BCD 8421. Đối với loại bộ đếm này, các ngõ ra thay đổi trạng thái không đồng thời với tín hiệu điều khiển Ck (tức không chịu sự điều khiển của tín hiệu điều khiển Ck) do đó mạch đếm nối tiếp còn gọi là mạch đếm không đồng bộ.

2. Phân loại

- Đếm lên.
- Đếm xuống.
- Đếm lên /xuống.
- Đếm Modulo M.

a. Đếm lên

Đây là bộ đếm có nội dung tăng dần. Nguyên tắc ghép nối các TFF (hoặc JKFF thực hiện chức năng TFF) để tạo thành bộ đếm nối tiếp còn phụ thuộc vào tín hiệu đồng bộ Ck. Có 2 trường hợp khác nhau:

- Tín hiệu Ck tác động theo sườn xuống: TFF hoặc JKFF được ghép nối với nhau theo qui luật sau:

$$\mathbf{Ck_{i+1} = Q_i}$$

- Tên hiệu Ck tác động theo sườn lên: TFF hoặc JKFF được ghép nối với nhau theo qui luật sau:

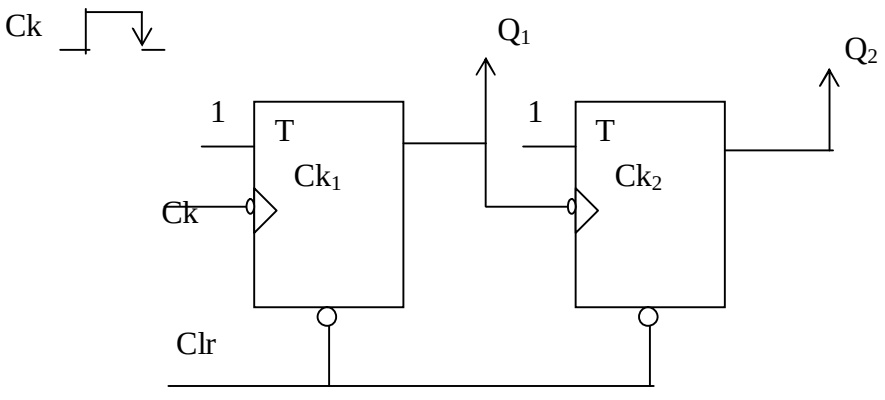
$$\mathbf{Ck_{i+1}} = \overline{Q_i}$$

Trong đó T luôn luôn giữ ở mức logic 1 (T = 1) và ngõ ra của TFF đứng trước nối với ngõ vào Ck của TFF đứng sau.

Để minh họa chúng ta xét ví dụ về một mạch đếm nối tiếp, đếm 4, đếm lên, dùng TFF.

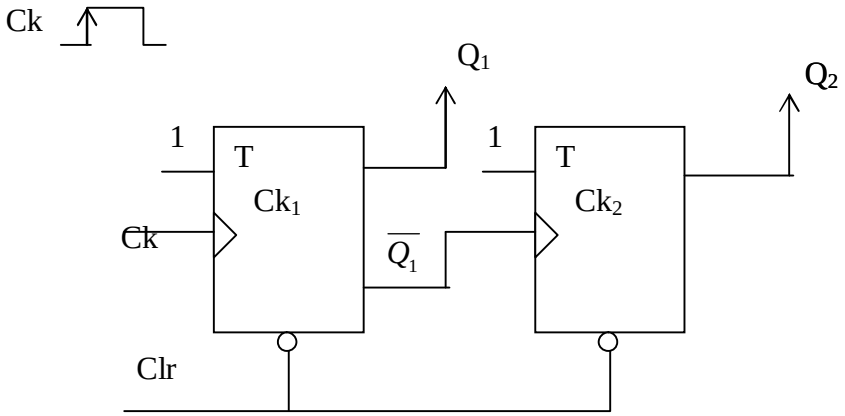
Số lượng TFF cần dùng: $4 = 2^2 \rightarrow$ dùng 2 TFF.

Trường hợp Ck tác động theo sườn xuống (hình 5.1a):



Hình 5.1a

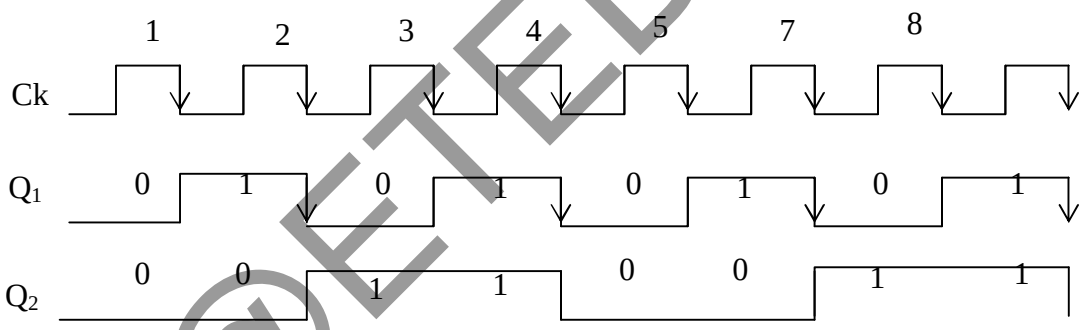
<u>Trường hợp Ck tác động theo sườn lên</u> (hình 5.1b):



H 5.1b

Trong các sơ đồ mạch này Clr (Clear) là ngõ vào xóa của TFF. Ngõ vào Clr tác động mức thấp, khi Clr = 0 thì ngõ ra Q của FF bị xóa về 0 (Q=0).

Giản đồ thời gian của mạch ở hình 5.1a :

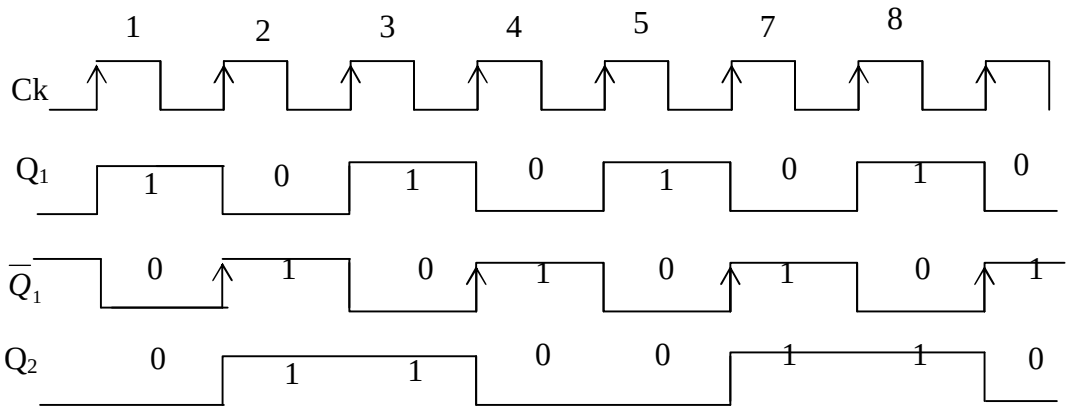


Hình 5.2a. Giản đồ thời gian mạch hình 5.1a

Bảng trạng thái hoạt động của mạch hình 5.1a:

Xung vào	Trạng thái hiện tại		Trạng thái kế tiếp	
Ck	**Q_2**	**Q_1**	**Q_2**	**Q_1**
1	0	0	0	1
2	0	1	1	0
3	1	0	1	1
4	1	1	0	0

Giản đồ thời gian mạch hình 5.1b :



Hình 5.2b. Giản đồ thời gian mạch hình 5.1b

Bảng trạng thái hoạt động của mạch hình 5.1b :

Xung vào	Trạng thái hiện tại		Trạng thái kế tiếp	
Ck	**Q_2**	**Q_1**	**Q_2**	**Q_1**
1	0	0	0	1
2	0	1	1	0
3	1	0	1	1
4	1	1	0	0

b. Đếm xuống

Đây là bộ đếm có nội dung đếm giảm dần. Nguyên tắc ghép các FF cũng phụ thuộc vào tín hiệu điều khiển Ck:

- Tín hiệu Ck tác động sườn xuống: TFF hoặc JKFF được nghép nối với nhau theo qui luật sau:

$$\mathbf{Ck_{i+1}} = \overline{Q_i}$$

- Tín hiệu Ck tác động sườn xuống: TFF hoặc JKFF được nghép nối với nhau theo qui luật sau:

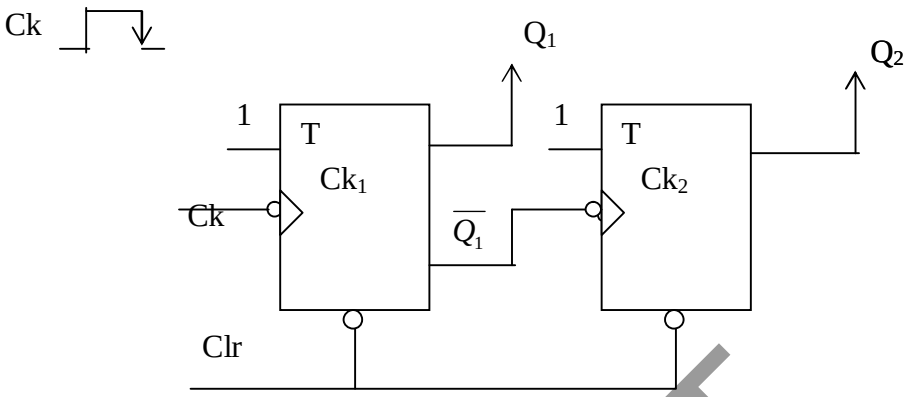
$$\mathbf{Ck_{i+1}} = \mathbf{Q_i}$$

Trong đó T luôn luôn giữ ở mức logic 1 (T = 1) và ngõ ra của TFF đứng trước nối với ngõ vào Ck của TFF đứng sau.

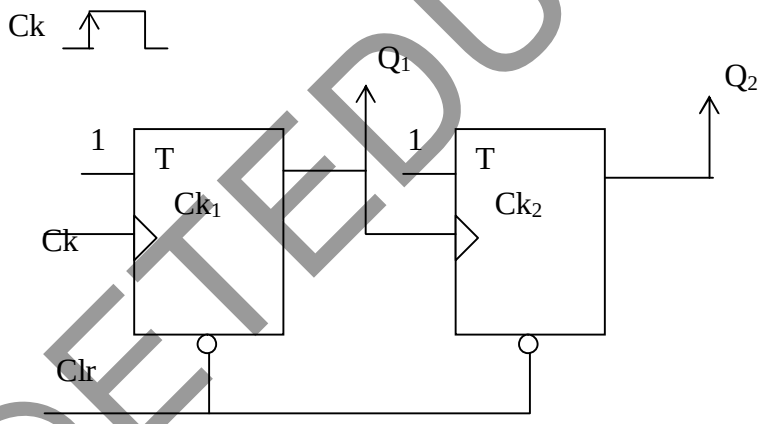
Ví dụ: Xét một mạch đếm 4, đếm xuống, đếm nối tiếp dùng TFF.

Số lượng TFF cần dùng: $4 = 2^2 \Rightarrow$ dùng 2 TFF.

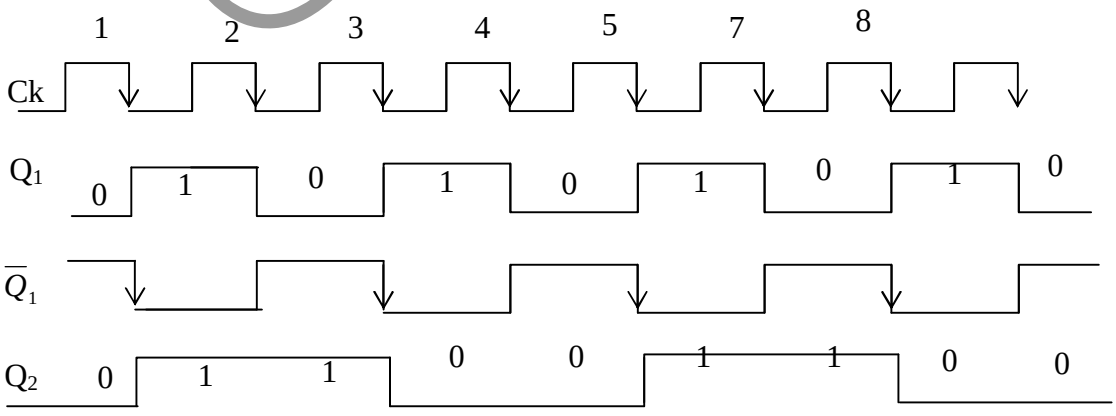
Sơ đồ mạch thực hiện khi sử dụng Ck tác động sườn xuống và Ck tác động sườn lên lần lượt được cho trên hình 5.3a và 5.3b :



Hình 5.3a



H 5.3b

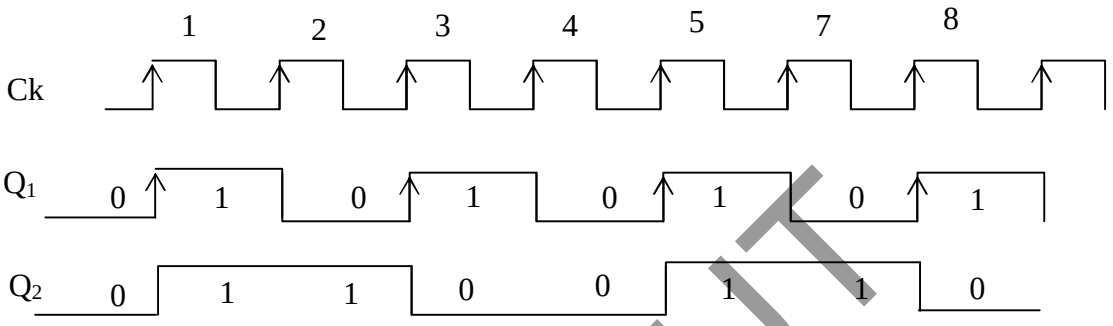


Hình 5.4a. Giản đồ thời gian mạch H 5.3a

Bảng trạng thái hoạt động của mạch hình 5.3a:

Xung vào	Trạng thái hiện tại		Trạng thái kế tiếp	
Ck	**Q_2**	**Q_1**	**Q_2**	**Q_1**
1	0	0	1	1
2	1	1	1	0
3	1	0	0	1
4	0	1	0	0

Giản đồ thời gian của mạch hình 5.3b:



Hình 5.4b. Giản đồ thời gian mạch hình 5.3b

Bảng trạng thái hoạt động của mạch hình 5.3b :

Xung vào	Trạng thái hiện tại		Trạng thái kế tiếp	
Ck	**Q_2**	**Q_1**	**Q_2**	**Q_1**
1	1	1	1	0
2	1	0	0	1
3	0	1	0	0
4	0	0	1	1

c. Đếm lên/xuống:

Gọi X là tín hiệu điều khiển chiều đếm, ta quy ước:

+ Nếu X = 0 thì mạch đếm lên.
+ Nếu X = 1 thì đếm xuống.

Ta xét 2 trường hợp của tín hiệu Ck:

- **Xét tín hiệu Ck tác động sườn xuống:**
  Lúc đó ta có phương trình logic:

$$Ck_{i+1} = \overline{X}.Q_i + X\overline{Q_i} = X \oplus Q_i$$

- **Xét tín hiệu Ck tác động sườn lên:**
  Lúc đó ta có phương trình logic:

$$Ck_{i+1} = \overline{X}.\overline{Q}_i + X.Q_i = \overline{X \oplus Q_i}$$

d. Đếm modulo M:

Đây là bộ đếm nối tiếp, theo mã BCD 8421, có dung lượng đếm khác 2^n.

Ví dụ: Xét mạch đếm 5, đếm lên, đếm nối tiếp.

Số lượng TFF cần dùng: Vì $2^2 = 4 < 5 < 8 = 2^3 \Rightarrow$ duùng 3 TFF.

Vậy bộ đếm này sẽ có 3 đầu ra (chú ý: Số lượng FF tương ứng với số đầu ra).

Bảng trạng thái hoạt động của mạch:

Xung vào	Trạng thái hiện tại			Trạng thái kế tiếp		
Ck	**Q_3**	**Q_2**	**Q_1**	**Q_3**	**Q_2**	**Q_1**
1	0	0	0	0	0	1
2	0	0	1	0	1	0
3	0	1	0	0	1	1
4	0	1	1	1	0	0
5	1	0	0	1/0	0	1/0

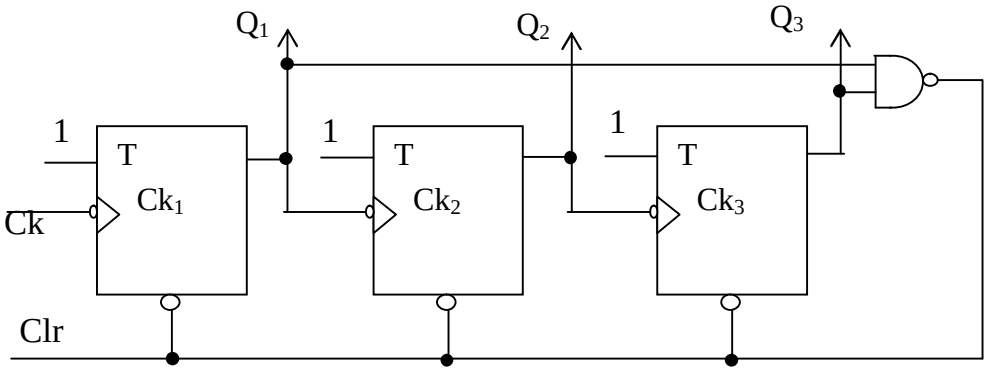
Nếu dùng 3 FF thì mạch có thể đếm được 8 trạng thái phân biệt (000 → 111 tương ứng 0→7). Do đó, để sử dụng mạch này thực hiện đếm 5, đếm lên, thì sau xung Ck thứ 5 ta tìm cách đưa tổ hợp 101 về 000 có nghĩa là mạch thực hiện việc đếm lại từ tổ hợp ban đầu. Như vậy, bộ đếm sẽ đếm từ 000 → 100 và quay về 000 trở lại, nói cách khác ta đã đếm được 5 trạng thái phân biệt.

Để xóa bộ đếm về 000 ta phân tích: Do tổ hợp 101 có 2 ngõ ra Q_1, Q_3 đồng thời bằng 1 (khác với các tổ hợp trước đó) (đây chính là dấu hiệu nhận biết để điều khiển xóa bộ đếm. Vì vậy để xóa bộ đếm về 000:

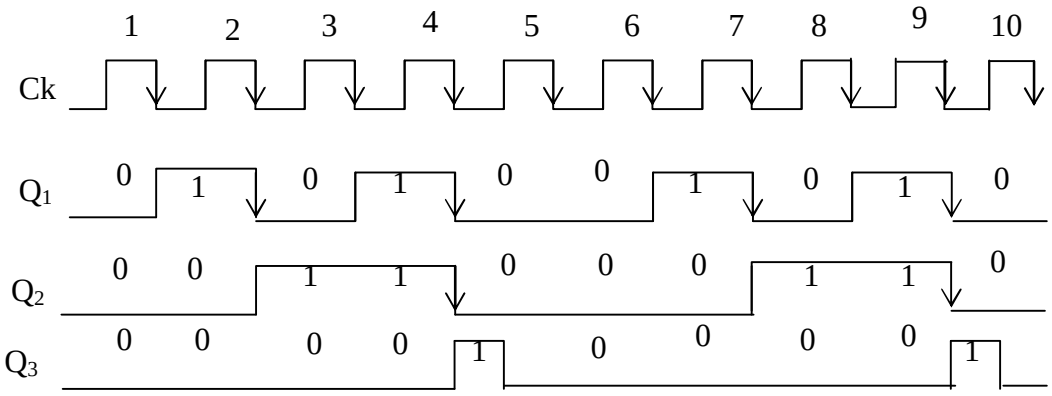
- Đối với FF có ngõ vào Clr tác động mức 0 thì ta dùng cổng NAND 2 ngõ vào.
- Đối với FF có ngõ vào Clr tác động mức 1 thì ta dùng cổng AND có 2 ngõ vào.

Như vậy sơ đồ mạch đếm 5 là sơ đồ cải tiến từ mạch đếm 8 bằng cách mắc thêm phần tử cổng NAND (hoặc cổng AND) có hai ngõ vào (tùy thuộc vào chân Clr tác động mức logic 0 hay mức logic 1) được nối đến ngõ ra Q_1 và Q_3, và ngõ ra của cổng NAND (hoặc AND) sẽ được nối đến ngõ vào Clr của bộ đếm (cũng chính là ngõ vào Clr của các FF).

Trong trường hợp Clr tác động mức thấp sơ đồ mạch thực hiện đếm 5 như trên hình 5.5 :



Hình 5.5. Mạch đếm 5, đếm lên

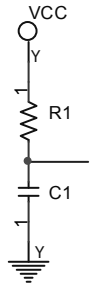


Hình 5.6. Giản đồ thời gian mạch đếm 5, đếm lên

Chú ý:

Do trạng thái của ngõ ra là không biết trước nên để mạch có thể đếm từ trạng thái ban đầu là 000 ta phải dùng thêm mạch xóa tự động ban đầu để xóa bộ đếm về 0 (còn gọi là mạch RESET ban đầu). Phương pháp thực hiện là dùng hai phần tử thụ động R và C.

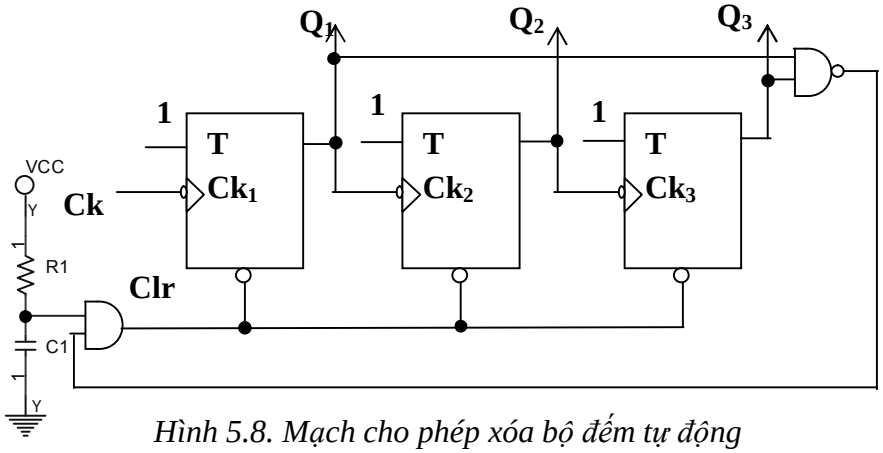
Trên hình 5.7 là mạch Reset mức 0 (tác động mức 0). Mạch hoạt động như sau: Do tính chất điện áp trên tụ C không đột biến được nên ban đầu mới cấp nguồn Vcc thì $V_C = 0$ (ngõ ra Clr = 0 và mạch có tác động Reset xóa bộ đếm, sau đó tụ C được nạp điện từ nguồn qua điện trở R với thời hằng nạp là $\tau = RC$ nên điện áp trên tụ tăng dần, cho đến khi tụ C nạp đầy thì điện áp trên tụ xấp xỉ bằng Vcc $\Rightarrow$ ngõ ra Clr = 1, mạch không còn tác dụng reset.



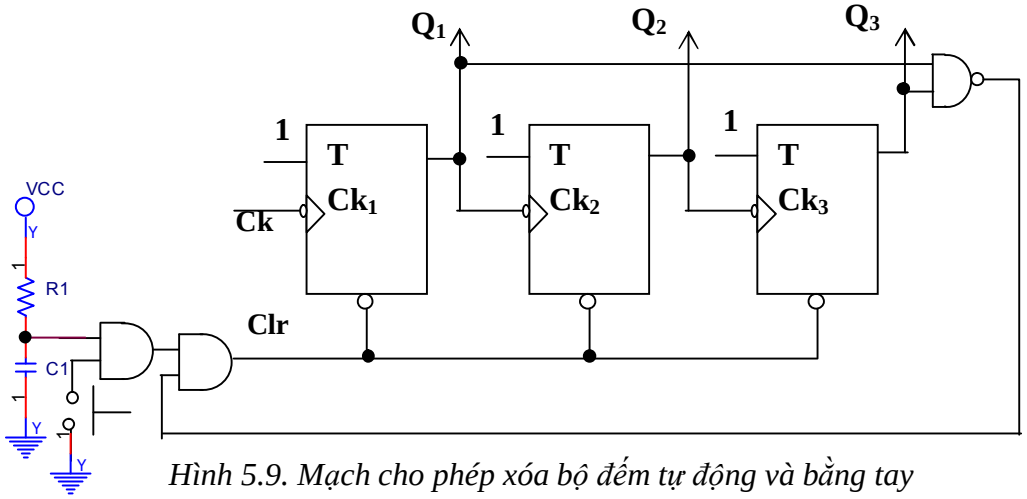
Hình 5.7. Mạch Reset mức 0

Chú ý khi thiết kế: Với một FF, ta biết được thời gian xóa (có trong Datasheet do nhà sản xuất cung cấp), do đó ta phải tính toán sao cho thời gian tụ C nạp điện từ giá trị ban đầu đến giá trị điện áp ngưỡng phải lớn hơn thời gian xóa cho phép thì mới đảm bảo xóa được các FF.

Mạch cho phép xóa bộ đếm tự động (H 5.8) và bằng tay (H 5.9):



Hình 5.8. Mạch cho phép xóa bộ đếm tự động



Hình 5.9. Mạch cho phép xóa bộ đếm tự động và bằng tay

Ưu điểm của bộ đếm nối tiếp: Đơn giản, dễ thiết kế.

Nhược điểm: Với dung lượng đếm lớn, số lượng FF sử dụng càng nhiều thì thời gian trễ tích lũy khá lớn. Nếu thời gian trễ tích lũy lớn hơn một chu kỳ tín hiệu xung kích thì lúc bấy giờ kết quả đếm sẽ sai. Do đó, để khắc phục nhược điểm này, người ta sử dụng bộ đếm song song.

5.2.3. Bộ đếm song song

1. Khái niệm

Bộ đếm song song là bộ đếm trong đó các FF mắc song song với nhau và các ngõ ra sẽ thay đổi trạng thái dưới sự điều khiển của tín hiệu Ck. Chính vì vậy mà người ta còn gọi bộ đếm song song là bộ đếm đồng bộ.

Mạch đếm song song được sử dụng với bất kỳ FF loại nào và có thể đếm theo qui luật bất kỳ cho trước. Vì vậy, để thiết kế bộ đếm đồng bộ (song song) người ta dựa vào các bảng đầu vào kích của FF.

2. Mạch thực hiện

Đối với bộ đếm song song dù đếm lên hay đếm xuống, hoặc là đếm Modulo M (đếm lên/đếm xuống) đều có cách thiết kế chung và không phụ thuộc vào tín hiệu Ck tác động sườn lên, sườn xuống, mức 0 hay mức 1.

Các bước thực hiện :

- Từ yêu cầu thực tế xây dựng bảng trạng thái hoạt động của bộ đếm.
- Dựa vào bảng đầu vào kích của FF tương ứng để xây dựng các bảng hàm giá trị của các ngõ vào dữ liệu (DATA) theo ngõ ra.
- Dùng các phương pháp tối thiểu để tối thiểu hóa các hàm logic trên.
- Thành lập sơ đồ logic.

Ví dụ:

Thiết kế mạch đếm đồng bộ, đếm 5, đếm lên theo mã BCD 8421 dùng JKFF.

Trước hết xác định số JKFF cần dùng: Vì $2^2 = 4 < 5 < 8 = 2^3 \Rightarrow$ dùng 3 JKFF $\Rightarrow$ có 3 ngõ ra Q_1, Q_2, Q_3.

Ta có bảng trạng thái mô tả hoạt động của bộ đếm như sau:

Xung vào	Trạng thái hiện tại			Trạng thái kế tiếp		
Ck	**Q_3**	**Q_2**	**Q_1**	**Q_3**	**Q_2**	**Q_1**
1	0	0	0	0	0	1
2	0	0	1	0	1	0
3	0	1	0	0	1	1
4	0	1	1	1	0	0
5	1	0	0	0	0	0

Ở chương 3 chúng ta đã xây dựng được bảng đầu vào kích cho các FF và đã có được bảng đầu vào kích tổng hợp như sau:

Q^n	Q^{n+1}	S^n	R^n	J^n	K^n	T^n	D^n
0	0	0	X	0	X	0	0
0	1	1	0	1	X	1	1
1	0	0	1	X	1	1	0
1	1	X	0	X	0	0	1

Từ đó ta suy ra bảng hàm giá trị của các ngõ vào data theo các ngõ ra như sau :

Xung vào	**Trạng thái hiện tại**			**Trạng thái kế tiếp**								
	Q_3	**Q_2**	**Q_1**	**Q_3**	**Q_2**	**Q_1**	**J_3**	**K_3**	**J_2**	**K_2**	**J_1**	**K_1**
1	0	0	0	0	0	1	0	X	0	X	1	X
2	0	0	1	0	1	0	0	X	1	X	X	1
3	0	1	0	0	1	1	0	X	X	0	1	X
4	0	1	1	1	0	0	1	X	X	1	X	1
5	1	0	0	0	0	0	X	1	0	X	0	X

Lập bảng Karnaugh để tối thiểu hóa ta được:

J_1 \ Q_3Q_2 / Q_1	00	01	11	10
0	1	1	x	0
1	x	x	x	x

$J_1 = Q_1$

K_1 \ Q_3Q_2 / Q_1	00	01	11	10
0	x	x	x	x
1	1	1	x	x

$K_1 = 1 = Q_1$

J_2 \ Q_3Q_2 / Q_1	00	01	11	10
0	0	x	x	0
1	1	x	x	x

$J_2 = Q_1$

K_2 \ Q_3Q_2 / Q_1	00	01	11	10
0	x	0	x	0
1	x	1	x	x

$K_2 = Q_1$

J_3 \ Q_3Q_2 / Q_1	00	01	11	10
0	0	0	x	X
1	0	1	x	x

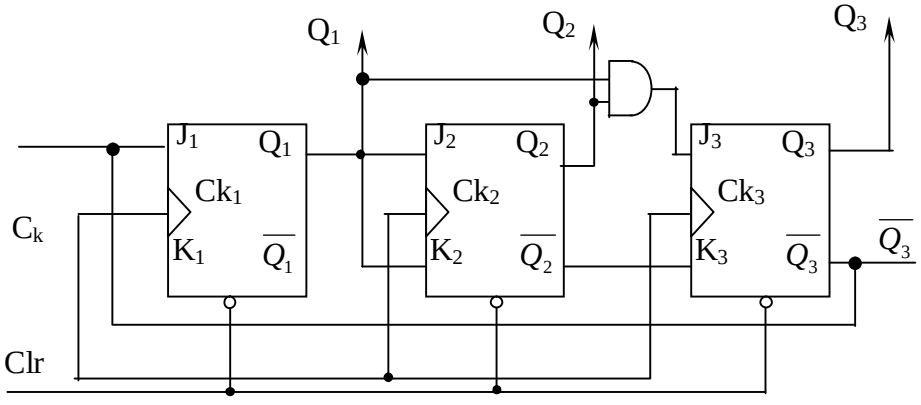
$J_2 = Q_1Q_2$

K_3 \ Q_3Q_2 / Q_1	00	01	11	10
0	x	0	x	0
1	x	1	x	x

$K_3 = 1 = Q_3 = \overline{Q_1} = \overline{Q_2}$

Lưu ý: Khi thiết kế tính toán ta dùng các phương pháp tối thiểu để đưa về phương trình logic tối giản. Nhưng trong thực tế thì đôi lúc không phải như vậy. Ví dụ: $K_3 = 1$, $K_3 = Q_3$ hay $K_3 = \overline{Q_2}$ đều đúng, nhưng khi lắp ráp thực tế ta chọn K3 = $\overline{Q_2}$ để tránh dây nối dài gây nhiễu cho mạch.

Sơ đồ logic: **Hình 5.10**



Hình 5.10. Sơ đồ mạch đếm lên đếm 5, đếm song song

<u>Giải thích hoạt động của bộ đếm</u>:

- Ban đầu dùng mạch RC xóa về 0 $\Rightarrow Q_1 = Q_2 = Q_3 = 0$.

  $J_1 = K_1 = 1$; $J_2 = K_2 = Q_2 = 0$; $J_3 = 0$, $K_3 = 1$.

- Khi Ck_1 ↓ : Các trạng thái ngõ ra đều thay đổi theo trạng thái ngõ vào DATA trước đó.

  $J_1 = K_1 = 1 \quad \Rightarrow Q_1 = \overline{Q_1^0} = 1$.

  $J_2 = K_2 = 1 \quad \Rightarrow Q_2 = Q_2^0 = 0$.

  $J_3 = 0, K_3 = 1 \Rightarrow Q_3 = 1$ bất chấp trạng thái trước đó.

  (Hoặc $J_3 = 0, K_3 = 0 \Rightarrow Q_3 = Q_3^0 = 0$) $\Rightarrow Q_3Q_2Q_1 = 001$.

  Lúc đó: $J_1 = K_1 = \overline{Q_3} = 1$; $J_2 = K_2 = Q_1 = 1$; $J_3 = Q_2.Q_1 = 0$, $K_3 = 1$.

  (Hoặc $K_3 = Q_3 = 0$).

- Khi Ck_2 ↓

  $J_1 = K_1 = 1 \quad \Rightarrow Q_1 = \overline{Q_1^1} = 0$.

  $J_2 = K_2 = 1 \quad \Rightarrow Q_2 = \overline{Q_2^1} = 1$.

  $J_3 = 0, K_3 = 1 \Rightarrow Q_3 = 0$.

  (Hoặc $J_3 = 0, K_3 = 0 \Rightarrow Q_3 = Q_3^1 = 0$) $\Rightarrow Q_3\,Q_2\,Q_1 = 010$.

  Lúc đó: $J_1 = K_1 = \overline{Q_3} = 1$; $J_2 = K_2 = Q_1 = 0$; $J_3 = 0$, $K_3 = 1$.

  (Hoặc $K_3 = \overline{Q_2} = 0$).

- Khi Ck_3 ↓

  $J_1 = K_1 = 1 \quad \Rightarrow Q_1 = \overline{Q_1^2} = 1$.

  $J_2 = K_2 = 0 \quad \Rightarrow Q_2 = Q_2^0 = 1$.

  $J_3 = 0, K_3 = 1 \Rightarrow Q_3 = 0$ bất chấp trạng thái trước đó.

  (Hoặc $J_3 = 0, K_3 = 0 \Rightarrow Q_3 = Q_3^2 = 0$) $\Rightarrow Q_3\,Q_2\,Q_1 = 011$.

  Lúc đó: $J_1 = K_1 = \overline{Q_3} = 1$; $J_2 = K_2 = Q_1 = 1$; $J_3 = Q_2.Q_1 = 1$, $K_3 = 0$.

  (Hoặc $K_3 = 1$).

- Khi Ck_4 ↓

  $J_1 = K_1 = 1 \quad \Rightarrow Q_1 = \overline{Q_1^3} = 0$.

  $J_2 = K_2 = 1 \quad \Rightarrow Q_2 = \overline{Q_2^3} = 0$.

  $J_3 = 0, K_3 = 1 \Rightarrow Q_3 = 1$ bất chấp trạng thái trước đó.

  (Hoặc $J_3 = 0, K_3 = 0 \Rightarrow Q_3 = Q_3^0 = 0$) $\Rightarrow Q_3\,Q_2\,Q_1 = 100$.

  Lúc đó: $J_1 = K_1 = \overline{Q_3} = 1$; $J_2 = K_2 = Q_1 = 0$; $J_3 = Q_2.Q_1 = 0$, $K_3 = 1$.

  (Hoặc $K_3 = Q_3 = 0$).

- Khi Ck_5 ↓

  $J_1 = K_1 = 1 \quad \Rightarrow Q_1 = Q_1^4 = 0$.

  $J_2 = K_2 = 1 \quad \Rightarrow Q_2 = Q_2^4 = 0$.

  $J_3 = 0, K_3 = 1 \Rightarrow Q_3 = 0$ bất chấp trạng thái trước đó.

  $\Rightarrow Q_3\,Q_2\,Q_1 = 000$.

  Lúc đó: $J_1 = K_1 = \overline{Q_3} = 1$; $J_2 = K_2 = Q_1 = 0$; $J_3 = Q_2.Q_1 = 0$, $K_3 = 1$.

  Mạch trở về trạng thái ban đầu.

5.2.4. Đếm thuận nghịch

Để thiết kế mạch cho phép vừa đếm lên vừa đếm xuống, ta thực hiện như sau:

- **Cách 1:** Lập hàm $J_{lên}$, $J_{xuống}$, $K_{lên}$, $K_{xuống}$ (giả sử ta dùng JKFF).
  Gọi X là tín hiệu điều khiển. Xét 2 trường hợp:
  + Nếu quy ước X = 0: đếm lên; X = 1: đếm xuống.
    Lúc đó ta có phương trình logic:
    $J = \overline{X} . J_{lên} + X. J_{xuống}$
    $K = \overline{X} . K_{lên} + X. K_{xuống}$
  + Nếu quy ước X = 1: đếm lên; X = 0: đếm xuống.
    Lúc đó ta có phương trình logic:
    $J = X. J_{lên} + \overline{X} . J_{xuống}$
    $K = X. K_{lên} + \overline{X} .K_{xuống}$

- **Cách 2:** Lập bảng trạng thái tổng hợp cho cả đếm lên và đếm xuống.

Xung vào	X	Trạng thái h.tại	Trạng thái kế	J_3 K_3	J_2 K_2	J_1 K_1
1						
2						

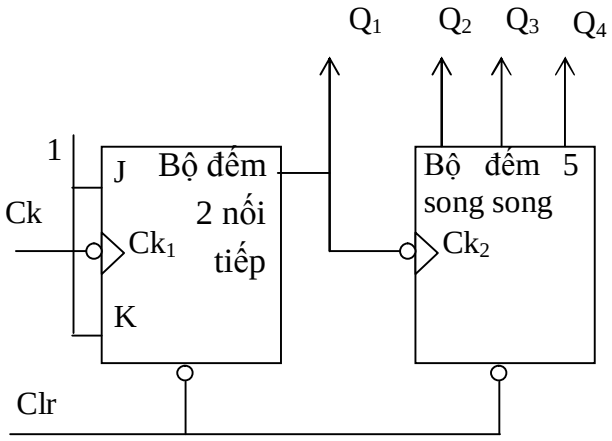
Sau đó thực hiện các bước giống như bộ đếm đồng bộ.

5.2.5. Đếm hỗn hợp

Bộ đếm hỗn hợp là bộ đếm mà trong đó bao gồm cả đếm nối tiếp và đếm song song. Đây là bộ đếm chế tạo khá nhiều trong thực tế và khả năng ứng dụng của bộ đếm hỗn hợp khá lớn so với bộ đếm song song.

Ví dụ: Bộ đếm 7490 bên trong bao gồm 2 bộ đếm đó là bộ đếm 2 nối tiếp và bộ đếm 5 song song. Hai bộ đếm này tách rời nhau. Do đó, tùy thuộc vào việc ghép hai bộ đếm này lại với nhau mà mạch có thể thực hiện được việc đếm thập phân hoặc chia tần số.

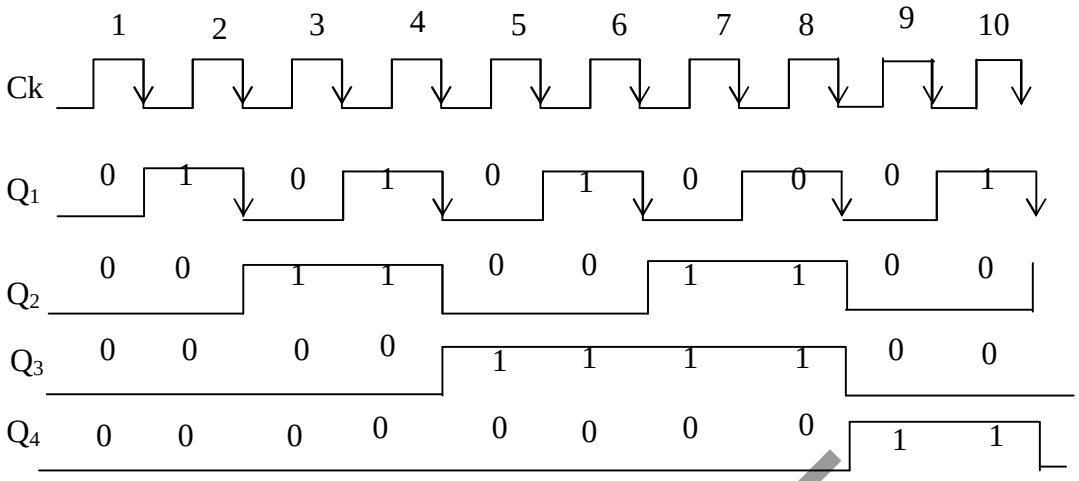
Trường hợp 1: 2 nối tiếp, 5 song song (hình 5.11).



Hình 5.11. Bộ đếm 2 nối tiếp ghép với bộ đếm 5 song song

Q_1 của bộ đếm 2 giữ vai trò xung Ck cho bộ đếm 5 song song.

Giản đồ thời gian của 2 nối tiếp 5 song song (hình 5.12) :



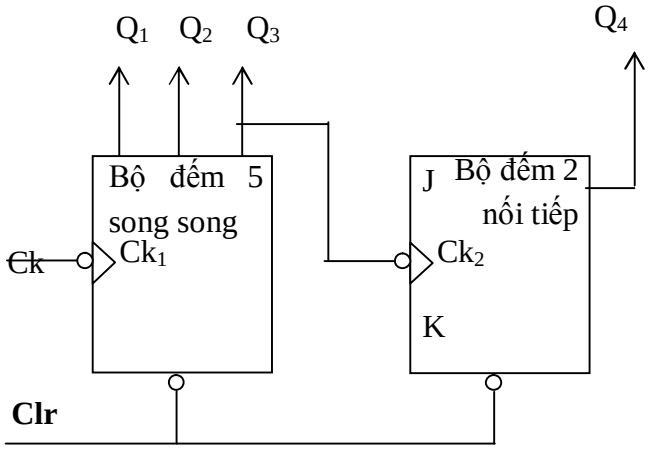
Hình 5.12. Giản đồ thời gian 2 nối tiếp ghép với 5 song song

Nhận xét: Cách ghép này dùng để đếm thập phân, nhưng không dùng để chia tần số.

Bảng trạng thái mô tả hoạt động của mạch:

Xung vào	Trạng thái hiện tại				Trạng thái kế tiếp			
Ck	Q_4	Q_3	Q_2	Q_1	Q_4	Q_3	Q_2	Q_1
1	0	0	0	0	0	0	0	1
2	0	0	0	1	0	0	1	0
3	0	0	1	0	0	0	1	1
4	0	0	1	1	0	1	0	0
5	0	1	0	0	0	1	0	1
6	0	1	0	1	0	1	1	0
7	0	1	1	0	0	1	1	1
8	0	1	1	1	1	0	0	0
9	1	0	0	0	1	0	0	1
10	1	0	0	1	0	0	0	0

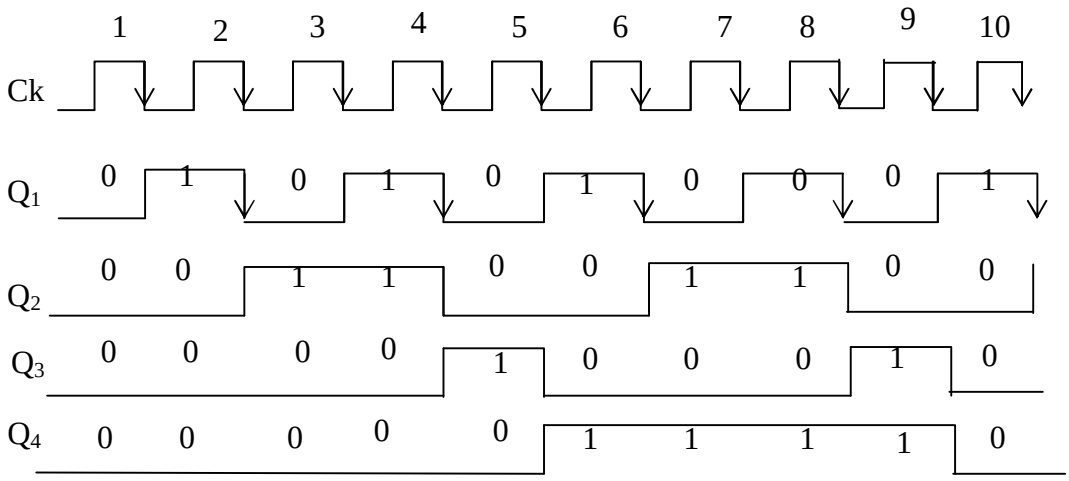
Trường hợp 2: 5 song song, 2 nối tiếp.



Hình 5.13. Bộ đếm 5 song song ghép với 2 nối tiếp

Q_3 của bộ đếm 5 song song giữ vai trò xung Ck cho bộ đếm 2.

Giản đồ thời gian của 5 song song nối tiếp 2.



Hình 5.14. Giản đồ thời gian đếm 5 song song ghép 2 nối tiếp

Nhận xét: Cách ghép này không được dùng để đếm thập phân, nhưng lại thích hợp cho việc chia tần số.

Bảng trạng thái mô tả hoạt động của mạch :

Xung vào	Trạng thái hiện tại				Trạng thái kế tiếp			
Ck	**Q_4**	**Q_3**	**Q_2**	**Q_1**	**Q_4**	**Q_3**	**Q_2**	**Q_1**
1	0	0	0	0	0	0	0	1
2	0	0	0	1	0	0	1	0
3	0	0	1	0	0	0	1	1
4	0	0	1	1	0	1	0	0
5	0	1	0	0	0	1	0	1
6	1	0	0	0	1	0	0	1
7	1	0	0	1	1	0	1	0
8	1	0	1	0	1	0	1	1
9	1	0	1	1	1	1	0	0
10	1	1	0	1	0	0	0	0

5.3. THANH GHI DỊCH CHUYỂN VÀ BỘ NHỚ

5.3.1. Khái niệm

Thanh ghi dịch và bộ nhớ đều được ứng dụng trong lưu trữ dữ liệu, trong đó thanh ghi do khả năng lưu trữ của nó có hạn nên chỉ được sử dụng như bộ nhớ tạm thời (lưu kết quả các phép tính). Còn bộ nhớ có khả năng lưu trữ các bit dữ liệu khá lớn, về mặc cấu tạo bộ nhớ được xây dựng trên cơ sở các thanh ghi (Nhiều thanh ghi hợp thành bộ nhớ)

5.3.2. Thanh ghi dịch chuyển

1. Khái niệm

Thanh ghi được xây dựng trên cơ sở các DFF (hoặc các FF khác thực hiện chức năng của DFF) và trong đó mỗi DFF sẽ lưu trữ 1 bit dữ liệu.

Để tạo thanh ghi nhiều bit, người ta ghép nhiều DFF lại với nhau theo qui luật như sau:

- Ngõ ra của DFF đứng trước được nối với ngõ vào DATA của DFF sau ($D_{i+1} = Q_i$) (thanh ghi có khả năng dịch phải.
- Hoặc ngõ ra của DFF đứng sau được nối với ngõ vào DATA của DFF đứng trước ($D_i = Q_{i+1}$) (thanh ghi có khả năng dịch trái.

2. Phân loại

Phân loại theo số bit dữ liệu lưu trữ: 4 bit, 5 bit, 8 bit, 16 bit, 32 bit. Đối với thanh ghi lớn 8 bit, người ta không dùng họ TTL mà dùng họ CMOS.

Phân loại theo hướng dịch chuyển dữ liệu trong thanh ghi:

- Thanh ghi dịch trái.
- Thanh ghi dịch phải.
- Thanh ghi vừa dời phải vừa dời trái.

Phân loại theo ngõ vào dữ liệu:

- Ngõ vào dữ liệu nối tiếp.
- Ngõ vào dữ liệu song song: Song song không đồng bộ, song song đồng bộ.

Phân loại theo ngõ ra:

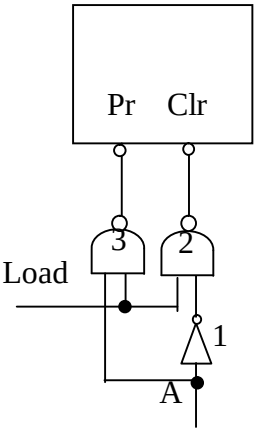
- Ngõ ra nối tiếp.
- Ngõ ra song song.
- Ngõ ra vừa nối tiếp vừa song song.

3. Nhập dữ liệu vào FF

Nhập dữ liệu vào FF bằng chân Preset (Pr): (xem hình 5.15)

- Khi Load = 0 : Cổng NAND 3 và 2 khóa → ngõ vào Pr = Clr = 1 → FF tự do → dữ liệu A không nhập vào được FF.
- Khi Load = 1 : Cổng NAND 2 và 3 mở, ta có: Pr = $\overline{A}$, Clr = A.
  Nếu A = 0 → Pr = 1, Clr = 0 → Q = A = 0.
  Nếu A = 1 → Pr = 0, Clr = 1 → Q = A = 1.
  Vậy Q = A→ dữ liệu A được nhập vào FF.

Tuy nhiên, cách này phải dùng nhiều cổng logic không kinh tế và phải dùng chân Clr là chân xóa nên phải thiết kế xóa đồng bộ.



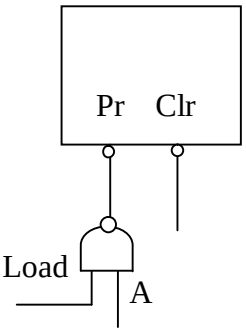
Hình 5.15

Để khắc phục những nhược điểm đó dùng mạch như trên hình 5.16 :

- Chân Clr để trống tương đương với mức logic 1.
- Khi Load = 0 : cổng NAND khóa → Pr = Clr =1 → FF tự do. Dữ liệu không được nhập vào FF.
- Khi Load = 1 : cổng NAND mở → Pr = $\overline{A}$.
  Giả sử ban đầu : Q = 0.
  Nếu A = 0 → Pr = 1, Clr = 1 ⇒ Q = Q^0 = 0.
  Nếu A = 1 → Pr = 0, Clr = 1 ⇒ Q = 1.

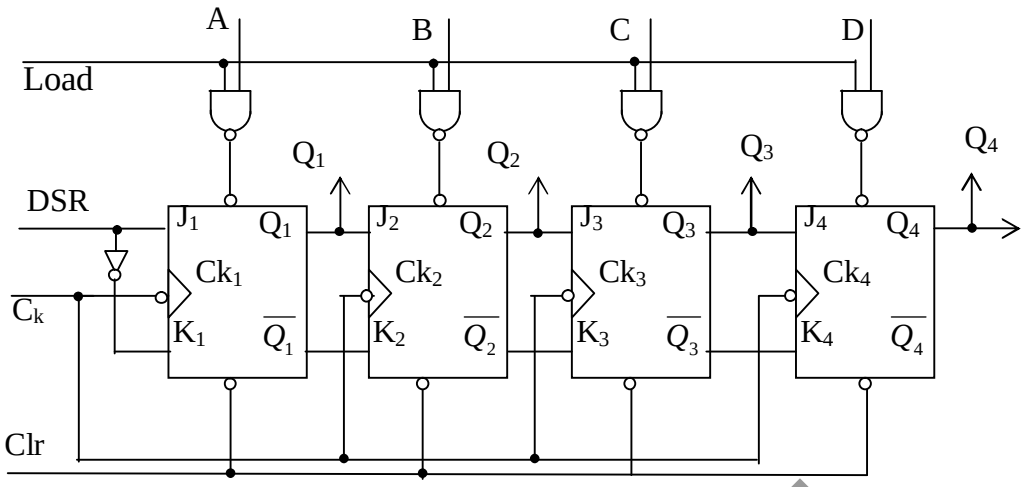
Vậy Q = A → Dữ liệu A được nhập vào FF.

Chú ý: Phương pháp này đòi hỏi trước khi nhập phải xóa FF về 0.



Hình 5.16

Ví dụ: Xét một thanh 4 bit có khả năng dời phải (hình 5.17).



Hình 5.17. Thanh ghi dịch phải

Trong đó:

- DSR (Data Shift Right): Ngõ vào Data nối tiếp (ngõ vào dịch phải).
- Q_1, Q_2,Q_3, Q_4 : các ngõ ra song song.

Để giải thích hoạt động của mạch, ta dựa vào bảng trạng thái của DFF.

Giả sử ban đầu : Ngõ vào nhập Load = 1 → A, B, C, D được nhập vào thanh ghi dịch:

$Q_1 = A$, $Q_2 = B$, $Q_3 = C$, $Q_4 = D$.

Hoạt động dịch phải của thanh ghi:

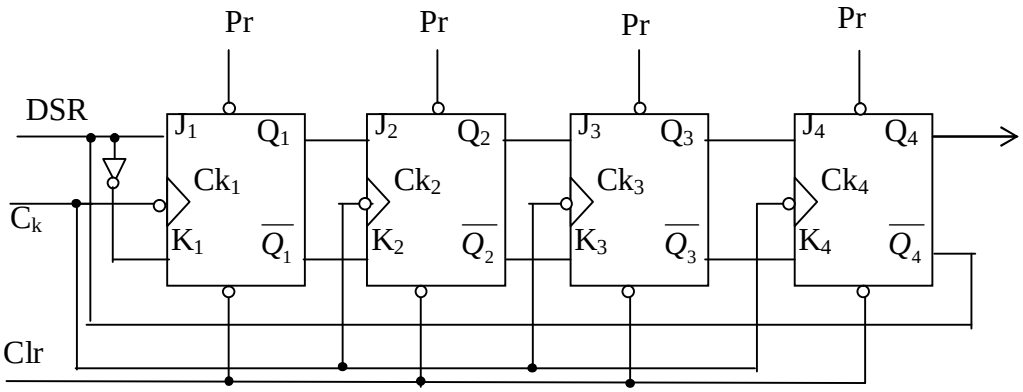
- Xét FF_1: $D = DSR_1$, $Q_1 = A$.
  Nếu $DSR_1 = 0 \rightarrow Q = 0$; nếu $DSR_1 = 1 \rightarrow Q = 1$.
  Kết luận: Sau một xung Ck tác động sườn xuống thì $Q_1 = DSR_1$.
- Lúc đó FF_2, FF_3,FF_4 : $Q_2 = A$, $Q_3 = B$, $Q_4 = C$.

Tức là sau khi Ck tác động sườn xuống thì nội dung trong thanh ghi được dời sang phải 1 bit. Sau 4 xung, dữ liệu trong thanh ghi được xuất ra ngoài và nội dung DFF được thay thế bằng các dữ liệu từ ngõ vào DATA nối tiếp DSR_1, DSR_2, DSR_3, DSR_4.

Ta có bảng trạng thái hoạt động của mạch:

Xung vào	**Trạng thái hiện tại**				**Trạng thái kế**			
	Q_1	**Q_2**	**Q_3**	**Q_4**	**Q_1**	**Q_2**	**Q_3**	**Q_4**
1	A	B	C	D	DSR_1	A	B	C
2	DSR_1	A	B	C	DSR_2	DSR_1	A	B
3	DSR_2	DSR_1	A	B	DSR_3	DSR_2	DSR_1	A
4	DSR_3	DSR_2	DSR_1	A	DSR_4	DSR_3	DSR_2	DSR_1

Trường hợp ngõ ra $\overline{Q}$ được nối với ngõ vào dữ liệu nối tiếp DSR (hình 5.18).



Hình 5.18.

Ta có bảng trạng thái hoạt động của mạch hình 5.18:

Xung vào	Trạng thái hiện tại				Trạng thái kế			
	Q_1	**Q_2**	**Q_3**	**Q_4**	**Q_1**	**Q_2**	**Q_3**	**Q_4**
1	0	0	0	0	1	0	0	0
2	1	0	0	0	1	1	0	0
3	1	1	0	0	1	1	1	0
4	1	1	1	0	1	1	1	1
5	1	1	1	1	0	1	1	1
6	0	1	1	1	0	0	1	1
7	0	0	1	1	0	0	0	1
8	0	0	0	1	0	0	0	0

Đây là mạch được ứng dụng nhiều trong thực tế.

5.3.3. Bộ nhớ

1. Các khái niệm

- **Tế bào nhớ (Memory cell)**

  Đó là thiết bị hay mạch điện tử dùng để lưu trữ 1 bit.

  Ví dụ: FF để lưu trữ 1 bit, tụ điện khi nạp điện thì lưu trữ 1 bit, hoặc một điểm trên băng từ.

- **Từ nhớ (Memory word)**

  Là nhóm các bit ở trong một bộ nhớ.

  Ví dụ: Một thanh ghi gồm 8 DFF có thể lưu trữ từ nhớ là 8 bit.

  Trong thực tế, kích thước của từ nhớ có thể thay đổi trong các loại máy tính từ 4 (64 bit.

- **Byte:**

  Một nhóm từ nhớ 8 bit.

- **Dung lượng bộ nhớ**

  Chỉ khả năng lưu trữ của bộ nhớ.

  Ví dụ: 1K = 2^{10} ; 2K = 2^{11}; 4K = 2^{12} ; 1M = 2^{20}.

- **Địa chỉ**

  Dùng để xác định các vùng của các từ trong bộ nhớ.

  Xét bộ nhớ gồm 16 ngăn nhớ tương đương 16 từ, ta cần dùng 4 đường địa chỉ ($2^4 = 16 \rightarrow$ có 4 đường địa chỉ). Như vậy có mối quan hệ giữa địa chỉ và dung lượng bộ nhớ.

Ví dụ : Để quản lý được bộ nhớ có dung lượng là 8 Kbytes thì cần 13 đường địa chỉ.

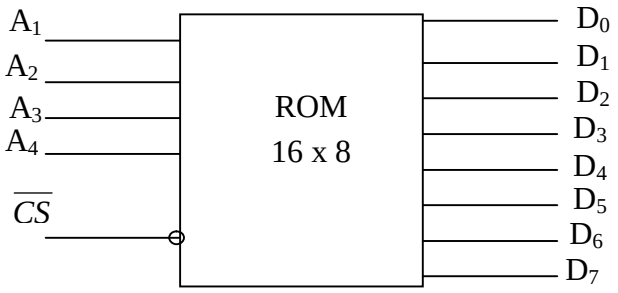
- **Hoạt động đọc (READ)**
  Đọc là xuất dữ liệu từ bộ nhớ ra ngoài.
  Để đọc nội dung một ô nhớ cần thực hiện:
  + Đưa địa chỉ tương ứng vào các đường địa chỉ A.
  + Khi tín hiệu điều khiển đọc tác động thì lúc bấy giờ dữ liệu chứa trong các ngăn nhớ tương ứng với vùng địa chỉ xác định ở trên sẽ được xuất ra ngoài.
- **Hoạt động viết (WRITE)**
  Viết là ghi dữ liệu từ bên ngoài vào bên trong bộ nhớ.
  Muốn viết phải thực hiện:
  + Đặt các địa chỉ tương ứng lên các đường địa chỉ.
  + Đặt dữ liệu cần viết vào bộ nhớ lên các đường dữ liệu.
  + Tích cực tín hiệu điều khiển ghi.

  Khi ghi dữ liệu từ bên ngoài vào bên trong bộ nhớ thì dữ liệu cũ sẽ mất đi và được thay thế bằng dữ liệu mới.
- **Bộ nhớ không bay hơi**
  Chỉ loại bộ nhớ mà dữ liệu không mất đi khi mất nguồn điện.
- **Bộ nhớ bay hơi**
  Chỉ loại bộ nhớ lưu trữ dữ liệu khi còn nguồn điện và khi mất nguồn điện thì dữ liệu sẽ bị mất.
- **RAM** (**R**andom **A**ccess **M**emory)
  Bộü nhớ truy xuất ngẫu nhiên, đọc viết tùy ý, còn được gọi là RWM (Read/Write Memory). Đây là loại bộ nhớ cho phép đọc dữ liệu chứa bên trong ra ngoài và cho phép nhập dữ liệu từ bên ngoài vào trong.
- **ROM** (**R**ead **O**nly **M**emory)
  Bộ nhớ chỉ đọc. Chỉ cho phép đọc dữ liệu trong ROM ra ngoài mà không cho phép dữ liệu ghi dữ liệu từ bên ngoài vào trong bộ nhớ.
- **SM (Static Memory)**
  Bộ nhớ tĩnh là loại bộ nhớ lưu trữ dữ liệu cho đến khi mất điện áp cung cấp mà không cần làm tươi dữ liệu bên trong. Ví dụ: SRAM.
- **DM (Dynamic Memory)**
  Bộ nhớ động là loại bộ nhớ có thể mất dữ liệu khi điện áp cung cấp chưa bị mất, vì vậy cần có cơ chế làm tươi dữ liệu. Ưu điểm của loại bộ nhớ này là tốc độ truy xuất nhanh, giá thành hạ. Ví dụ: DRAM.
- **Bộ nhớ tuần tự**
  Ví dụ: Đĩa mềm, đĩa cứng, băng từ.

2.ROM (Read Only Memory)

- **MROM (Mask ROM): Được lập trình bởi nhà sản xuất.**
  Ưu và nhược điểm: Chỉ có tính kinh tế khi sản xuất hàng loạt nhưng lại không phục hồi được khi chương trình bị sai hỏng.
- **PROM (Programmable ROM): Đây là loại ROM cho phép lập** trình bởi nhà sản xuất. Nhược điểm: Nếu hỏng không phục hồi được.
- **EPROM (Erasable PROM): Đó là loại PROM có thể xóa và** lập trình lại. Có hai loại EPROM: EPROM được xóa bằng tia cực tím (Ultralviolet EPROM) và EPROM xóa bằng xung điện (Electrical EPROM). Tuổi thọ của EPROM phụ thuộc vào thời gian xóa.

Ứng dụng của ROM: Chứa chương trình điều khiển vào ra của máy tính, PC, µP, µC, ROM BIOS (ROM Basic Input/Output System). Dùng để chứa ký tự: ROM ký tự. Dùng để chứa các biến đổi hàm.



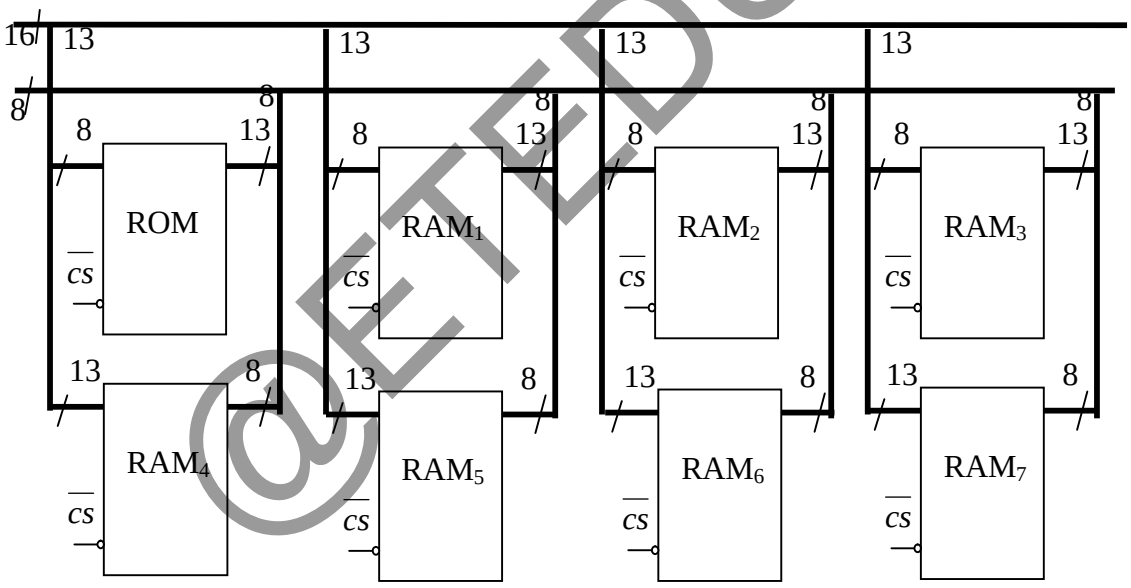
Hình 5.19. Sơ đồ khối của ROM 16x8 = 128 bit

3.RAM (Random Access Memory)

DRAM: RAM động, làm việc theo hai pha. Một pha chọn địa chỉ hàng, một pha chọn địa chỉ cột. Do đó, số chân địa chỉ thực hiện trên IC nhỏ hơn một nữa so với RAM hoặc ROM.

SRAM : RAM tĩnh, có tốc độ truy xuất nhanh hơn DRAM, do đó giá thành chế tạo đắt hơn.

4.Tổ chức bộ nhớ



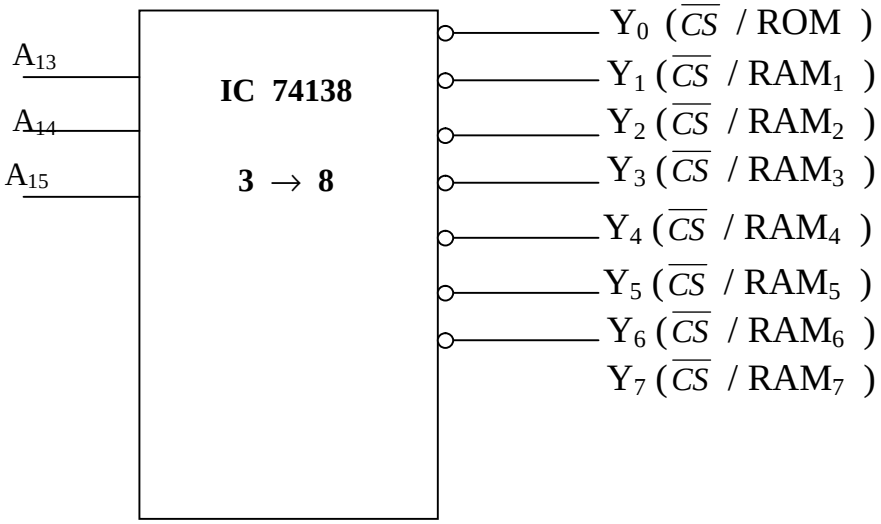
Hình 5.20. Tổ chức bộ nhớ

Giả sử CPU hay µP có 16 đường địa chỉ và 8 đường dữ liệu. Nếu dùng để quản lý bộ nhớ thì quản lý được dung lượng bộ nhớ tối đa là 64 KBytes (2^{16} = 64K).

Giả sử 64 KBytes phân thành các loại sau: 1 ROM 8K, và 7 RAM 8K.

Để chọn lần lượt từng bộ nhớ để xuất dữ liệu và vì còn thừa 3 dường địa chỉ là A_{13}, A_{14}, A_{15} nên ta dùng mạch giải mã từ 3 $\rightarrow$ 8.

Trên hình 5.21 là sơ đồ mạch giải mã địa chỉ dùng IC 74138.



Hình 5.21. Mạch giải mã địa chỉ

Bản đồ bộ nhớ của hệ thống:

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Địa chỉ Hex	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0 0 0 0 H	ROM
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1 F F F H	
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2 0 0 0 H	RAM_1
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3 F F F H	
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4 0 0 0 H	RAM_2
0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	5 F F F H	
0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	6 0 0 0 H	RAM_3
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	7 F F F H	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8 0 0 0 H	RAM_4
1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	9 F F F H	
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	A 0 0 0 H	RAM_5
1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	B F F F H	
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	C 0 0 0 H	RAM_6
1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	D F F F H	
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	E 0 0 0 H	RAM_7
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	F F F F H	

SỐ CÓ DẤU VÀ CỘNG TRỪ SỐ CÓ DẤU

1. Số bù –1 và số bù –2

a. Số bù –1:

Bù –1 của một số nhị phân là một số khi cộng với số nhị phân đã cho thì tổng bằng 1 ở tất cả các bít.

Để tìm số bù –1 của một số nhị phân ta lấy đảo tất cả các bít của số nhị phân đó.

Nếu A là một số nhị phân thì số bù –1 của A, kí hiệu A^{-1}, sẽ là:

$$A^{-1} = \overline{A}$$

Ví dụ 1:

Số nhị phân	Số bù -1
10110	01001
11010	00101
110011	001100

b. Số bù –2:

Bù –2 của một số nhị phân bằng số bù –1 của nó cộng thêm 1 vào bít có trọng số thấp nhất LSB.

Nếu kí hiệu số bù –2 của số nhị phân A là A^{-2} thì:

$$A^{-2} = A^{-1} + 1_{LSB}$$

Ví dụ 2:

Số nhị phân	Số bù -1	Số bù -2
10010	01101	01110
11010	00101	00110
110011	001100	001101

2. Biểu diễn các số có dấu

Có 2 phương pháp thông dụng để biểu diễn các số nhị phân có dấu:

- biểu diễn theo ký hiệu bù –1
- biểu diễn theo ký hiệu bù –2

a. Biểu diễn số nhị phân có dấu theo ký hiệu bù –1:

Số có dấu n-bít được biểu diễn theo mã bù –1 theo nguyên tắc sau:

- Bít lớn nhất (MSB) là bít dấu, trong đó bít 0 tương ứng với số dương và bít 1 tương ứng với số âm.
- Các bít còn lại biểu diễn giá trị thực của số dương hay giá trị bù –1 của số âm.
- Dải giá trị biểu diễn đối với số nhị phân n-bít theo ký hiệu bù –1:

$$-(2^{n-1}-1) \le R \le (2^{n-1}-1)$$

Nếu dùng 4 bít để biểu diễn số có dấu theo ký hiệu bù –1 thì dải giá trị biểu diễn sẽ là: -7 đến +7.

b. Biểu diễn số nhị phân có dấu theo ký hiệu bù –2:

Số có dấu n-bít được biểu diễn theo mã bù –2 theo nguyên tắc sau:

- Bít lớn nhất (MSB) là bít dấu, trong đó bít 0 tương ứng với số dương và bít 1 tương ứng với số âm (tương tự nguyên tắc biểu diễn theo ký hiệu bù –1).
- Các bít còn lại biểu diễn giá trị thực của số dương hay giá trị bù –2 của số âm.
- Dải giá trị biểu diễn đối với số nhị phân n-bít theo ký hiệu bù –2:

$$-2^{n-1} \le R \le 2^{n-1}-1$$

Nếu dùng 4 bít để biểu diễn số có dấu theo ký hiệu bù –2 thì dải giá trị biểu diễn sẽ là: -8 đến +7.

Bảng dưới đây trình bày các phương pháp biểu diễn số nhị phân có dấu 4 bít theo ký hiệu bù –1 và bù –2 tương ứng:

Số thập phân	Biểu diễn theo bù -1	Biểu diễn theo bù -2
-8		1 000
-7	1 000	1 001
-6	1 001	1 010
-5	1 010	1 011
-4	1 011	1 100
-3	1 100	1 101
-2	1 101	1 110
-1	1 110	1 111
-0	1 111	
+0	0 000	0 000
+1	0 001	0 001
+2	0 010	0 010
+3	0 011	0 011
+4	0 100	0 100
+5	0 101	0 101
+6	0 110	0 110
+7	0 111	0 111

Từ bảng này chúng ta thấy rằng cách biểu diễn theo ký hiệu bù –1 có nhược điểm là có đến 2 giá trị +0 và –0.

3. Cộng trừ số có dấu

Tương ứng với 2 phương pháp biểu diễn theo ký hiệu bù –1 và bù –2 cũng sẽ có 2 phương pháp thực hiện các phép toán cộng và trừ các số có dấu.

a. Cộng trừ số biểu diễn bằng ký hiệu bù –1:

Thực hiện phép cộng đối với số n-bít biểu diễn theo ký hiệu bù –1 cũng giống như cộng các số nhị phân không dấu, cộng cả bít dấu. ***Lưu ý cộng luôn cả số nhớ của bít lớn nhất (MSB) vào bít nhỏ nhất (LSB)***.

Ví dụ:

```
             1111                      ┌─11
   13       001101            -13      │ 110010
 + 11     + 001011            -11      │+110100
 ----     --------           ----      │ ------
   24       011000            -24      │ 100110
                                       └──►+ 1
                                         ------
                                         100111
```

Phép trừ được thay thế bằng phép cộng, trong đó số bị trừ cộng với số bù –1 của số trừ, số nhớ của bít lớn nhất (nếu có) cũng sẽ được cộng tiếp vào bít nhỏ nhất tương tự như khi thực hiện phép toán cộng:

A - B = A + (-B)

Ví dụ:

6	0110	11
- 3	- 0011	+ 0110
		1100
3		0010
		+1
		0011
4	0100	0100
- 7	- 0111	+ 1000
- 3		1100

b. Cộng trừ số biểu diễn bằng ký hiệu bù –2:

Thực hiện phép cộng đối với số n-bít biểu diễn theo ký hiệu bù –2 cũng giống như cộng các số nhị phân không dấu, cộng cả bít dấu. ***Lưu ý số nhớ của bít lớn nhất (MSB) nếu có sẽ bị loại bỏ***.

Ví dụ:

	1		1 1 1
12	001100	-12	110100
+ 9	+ 001001	- 9	+ 110111
21	010101	-21	(1)1101011
			→ bỏ

5	00101	00101
- 10	- 01010	+ 10110
- 5		11011

4. Hiện tượng tràn

Khi thực hiện cộng trừ với số có dấu cần lưu ý nếu kết quả nằm ngoài phạm vi biểu diễn của số có dấu n-bít thì kết quả sai. Đó chính là ***hiện tượng tràn***. Để khắc phục hiện tượng này chúng ta cần tăng số bít biểu diễn số có dấu. Hãy xét ví dụ sau:

Thực hiện phép cộng bằng ký hiệu bù –2 sử dụng số 4 bít:

	1		1 1
- 4	1100	- 4	11100
-5	+ 1011	- 5	+ 11011
- 9	(1)0111 → +7	- 9	(1)10111
	→ bỏ		→ bỏ

Kết quả phép tính là +7 thay vì –9. Đây là một kết quả sai bởi do –9 nằm ngoài phạm vi biểu diễn của số có dấu 4 bít theo ký hiệu bù –2 (phạm vi biểu diễn từ -8 đến +7). Chúng ta khắc phục hiện tượng trên bằng cách tăng số bít biểu diễn theo ký hiệu bù –2 lên 5 bít (lúc này phạm vi biểu diễn từ -16 đến 15) và thực hiện lại ví dụ trên thì sẽ được kết quả đúng.

---------- o0o ----------

3.3.CHUYỂN ĐỔI GIỮA CÁC LOẠI FLIP-FLOP

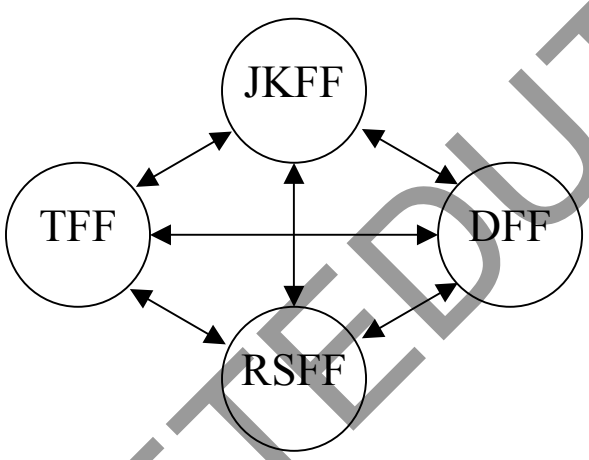
3.3.1. Ý nghĩa và phương pháp

1.Ý nghĩa

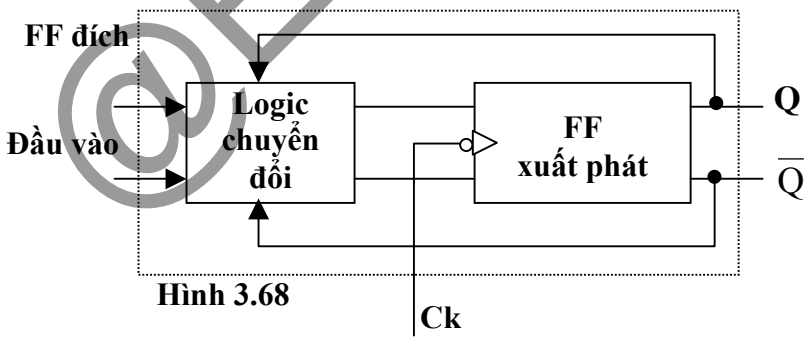
Đa số FF trên thị trường là loại JK, D trong khi kỹ thuật số yêu cầu tất cả các loại FF. Nếu biết cách chuyển đổi giữa các loại FF với nhau thì có thể phát huy tác dụng của loại FF sẵn có.

Bên cạnh đó, việc chuyển đổi giữa những loại FF khác nhau giúp đi sâu tìm hiểu chức năng của các loại FF.

Trên thực tế, có thể chuyển đổi qua lại giữa các loại FF khác nhau.



2. Phương pháp chuyển đổi



Hình 3.68

Trên hình 3.68 là sơ đồ khối biểu diễn tư duy về chuyển đổi từ một FF xuất phát để thực hiện chức năng của một FF đích.

Phương pháp chuyển đổi là các công việc cần phải làm để tìm logic chuyển đổi. Cụ thể theo sơ đồ khối hình 3.68 là: các đầu vào dữ liệu (data) của FF xuất phát là hàm ra với các biến là trạng thái ngõ ra Q^n và các đầu vào dữ liệu (data) của FF đích.

Xét các trường hợp chuyển đổi cụ thể ta có các hàm logic cần tìm:

- chuyển đổi từ JKFF → TFF : $J = f(T,Q^n)$ và $K = f(T,Q^n)$
- chuyển đổi từ JKFF → DFF : $J = f(D,Q^n)$ và $K = f(D,Q^n)$
- chuyển đổi từ JKFF → RSFF : $J = f(S,R,Q^n)$ và $K = f(S,R,Q^n)$

- chuyển đổi từ RSFF → TFF : $R = f(T,Q^n)$ và $S = f(T,Q^n)$
- chuyển đổi từ RSFF → DFF : $R = f(D,Q^n)$ và $S = f(D,Q^n)$
- chuyển đổi từ RSFF → JKFF : $R = f(J, K,Q^n)$ và $S = f(J,K,Q^n)$

- chuyển đổi từ TFF → DFF : $T = f(D,Q^n)$
- chuyển đổi từ TFF → RSFF : $T = f(R,S,Q^n)$
- chuyển đổi từ TFF → JKFF : $T = f(J,K,Q^n)$

- chuyển đổi từ DFF → TFF : $D = f(T,Q^n)$
- chuyển đổi từ DFF → RSFF : $D = f(R,S,Q^n)$
- chuyển đổi từ DFF → JKFF : $D = f(J,K,Q^n)$

Có 2 phương pháp để thực hiện chuyển đổi giữa các loại FF:
- phương pháp biến đổi trực tiếp.
- phương pháp dùng bảng đầu vào kích.

3.3.2. Phương pháp biến đổi trực tiếp

Đây là phương pháp sử dụng các định lý, tiên đề của đại số Boole để tìm phương trình logic tín hiệu kích thích đối với FF xuất phát. Cách làm này tiện cho việc trình bày viết, có thể dùng đại số Boole để xử lý nhưng cần một số kỹ xảo nhất định, trong phạm vi 4 loại FF chúng ta có thể nắm vững và vận dụng phương pháp này.

1. TFF chuyển đổi thành DFF, RSFF, JKFF:

a. TFF → RSFF

RSFF có pt: $\begin{cases} Q^{n+1} = S^n + \overline{R^n}\,Q^n & (1) \\ S^n R^n = 0 & \text{(điều kiện của RSFF)} \end{cases}$

TFF có pt: $Q^{n+1} = T^n \oplus Q^n$ (2)

So sánh (1) và (2) ta có:

$$S^n + \overline{R^n}\,Q^n = T^n \oplus Q^n$$

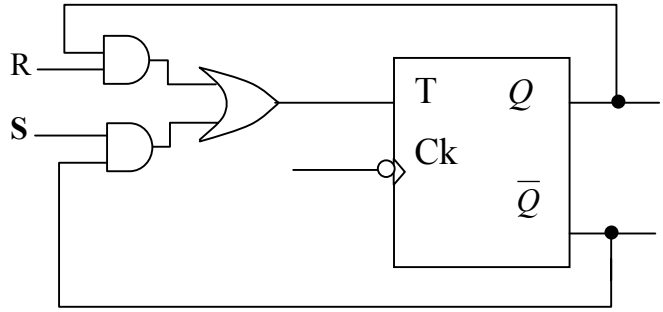
Theo tính chất của phép toán XOR, ta có:

$$T^n = Q^n \oplus (S^n + \overline{R^n}\,Q^n) = Q^n\,\overline{(S^n + \overline{R^n}Q^n)} + \overline{Q^n}(S^n + \overline{R^n}\,Q^n)$$

$$= Q^n\,\overline{S^n}\,R^n + S^n\overline{Q^n} = Q^n\,\overline{S^n}\,R^n + S^n\overline{Q^n} + S^n R^n = Q^n R^n + S^n\overline{Q^n}$$

Vậy: $\mathbf{T^n = Q^n R^n + S^n\overline{Q^n}}$

Sơ đồ mạch thực hiện:



Hình 3.69. Chuyển đổi TFF → RSFF

b.TFF → DFF:

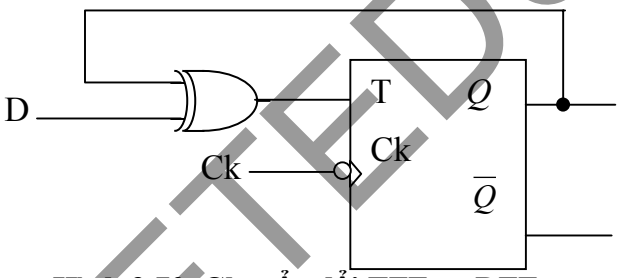
DFF có phương trình logic: $Q^{n+1} = D^n$

TFF có phương trình logic: $Q^{n+1} = T^n \oplus Q^n$

So sánh 2 phương trình: $D^n = T^n \oplus Q^n$

Theo tính chất của phép XOR ta suy ra: $\mathbf{T^n = D^n \oplus Q^n}$

Sơ đồ mạch thực hiện:

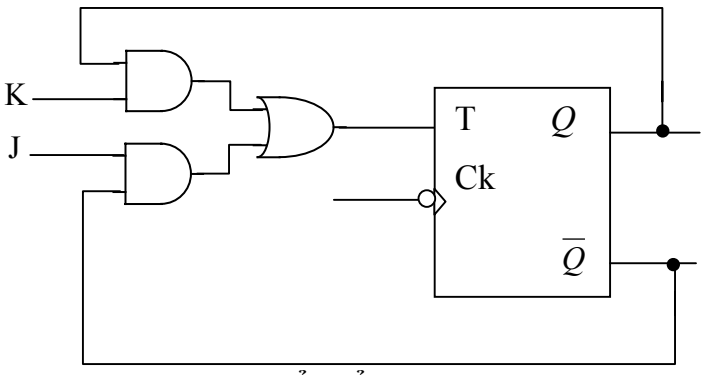


Hình 3.70. Chuyển đổi TFF → DFF

c.TFF → DFF:

Thực hiện biến đổi hoàn toàn tương tự (như trường hợp chuyển đổi từ TFF sang RSFF) ta có logic chuyển đổi:

$$\mathbf{T^n = K^nQ^n + J^n\overline{Q^n}}$$



Hình 3.71. Chuyển đổi TFF thành JKFF

2. DFF chuyển đổi thành TFF, RSFF, JKFF:

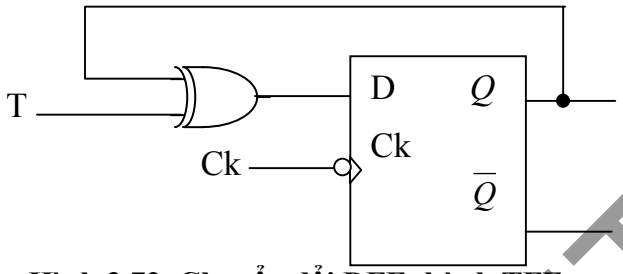
a. DFF → TFF:

DFF có phương trình logic: $Q^{n+1} = D^n$

TFF có phương trình logic: $Q^{n+1} = T^n \oplus Q^n$

So sánh 2 phương trình ta có: $\mathbf{D^n = T^n \oplus Q^n}$

Sơ đồ mạch thực hiện chuyển đổi (hình 3.72):



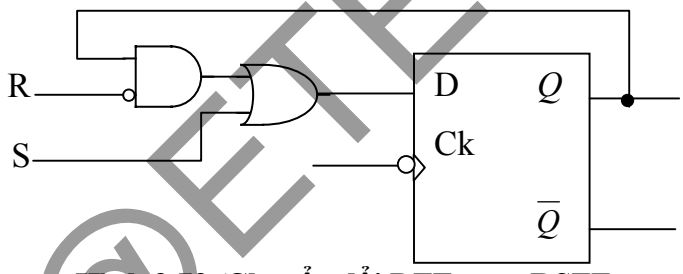
Hình 3.72. Chuyển đổi DFF thành TFF

b. DFF → RSFF:

RSFF có phương trình logic: $Q^{n+1} = S^n + \overline{R^n} Q^n$

So sánh với phương trình của DFF ta có: $\mathbf{D^n = S^n + \overline{R^n} Q^n}$

Sơ đồ mạch thực hiện chuyển đổi:



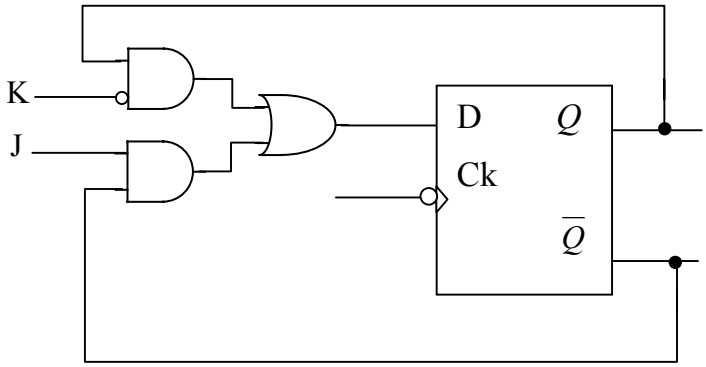
Hình 3.73. Chuyển đổi DFF sang RSFF

c. DFF → JKFF:

Hoàn toàn tương tự ta có logic chuyển đổi từ DFF sang JKFF:

$$\mathbf{D^n = J^n \overline{Q^n} + \overline{K^n} Q^n}$$

Sơ đồ mạch chuyển đổi trên hình 3.74:



Hình 3.74. Chuyển đổi DFF thành JKFF

3. RSFF chuyển đổi thành TFF, DFF, JKFF:

RSFF có pt: $\begin{cases} Q^{n+1} = S^n + \overline{R^n}\,Q^n \\ S^n R^n = 0 \end{cases}$ (điều kiện của RSFF)

Lưu ý: Khi thực hiện chuyển đổi từ RSFF sang các FF khác cần kiểm tra điều kiện ràng buộc của RSFF đó là: $\mathbf{R^nS^n = 0}$.

a. RSFF → TFF:

TFF có phương trình logic: $Q^{n+1} = T^n \oplus Q^n$

So sánh với phương trình của RSFF ta có:

$$S^n + \overline{R^n}\,Q^n = T^n \oplus Q^n = T^n\overline{Q^n} + \overline{T^n}\,Q^n$$

Từ biểu thức này, nếu ta cho:

$$\begin{cases} S^n = T^n\overline{Q^n} \\ R^n = T^n \end{cases}$$

thì suy ra:

$$S^n R^n = T^n\overline{Q^n}.T^n = T^n\overline{Q^n} \neq 0$$

nên không thỏa mãn điều kiện của RSFF.

Thực hiện biến đổi tiếp:

$$S^n + \overline{R^n}\,Q^n = T^n\overline{Q^n} + \overline{T^n}\,Q^n = T^n\overline{Q^n} + \overline{T^n}\,Q^n + \overline{Q^n}\,Q^n$$

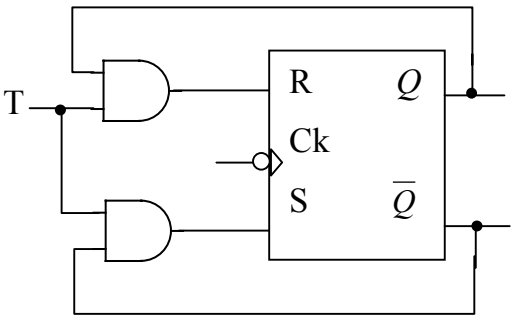
$$S^n + \overline{R^n}\,Q^n = T^n\overline{Q^n} + (\overline{T^n} + \overline{Q^n})Q^n = T^n\overline{Q^n} + \overline{T^nQ^n}\,Q^n$$

So sánh 2 vế ta có:

$$\begin{cases} \mathbf{S^n = T^n\overline{Q^n}} \\ \mathbf{R^n = T^n Q^n} \end{cases}$$

thỏa mãn điều kiện: $R^nS^n = 0$.

Sơ đồ thực hiện: hình 3.75.



Hình 3.75. Chuyển RSFF sang TFF

b. RSFF → DFF:

DFF có phương trình: $Q^{n+1} = D^n$

So sánh 2 phương trình: $S^n + \overline{R^n}\,Q^n = D^n$

Thực hiện biến đổi:

$$S^n + \overline{R^n}\,Q^n = D^n = D^n(Q^n + \overline{Q^n}) = D^nQ^n + D^n\overline{Q^n} \qquad (a)$$

Mặt khác biểu thức của RSFF có thể biến đổi như sau:

$$\begin{aligned} S^n + \overline{R^n}\,Q^n &= S^n(Q^n + \overline{Q^n}) + \overline{R^n}\,Q^n = S^nQ^n + S^n\overline{Q^n} + \overline{R^n}\,Q^n \\ &= S^nQ^n(R^n + \overline{R^n}) + S^n\overline{Q^n} + \overline{R^n}\,Q^n \\ &= S^nQ^n\,\overline{R^n} + S^n\overline{Q^n} + \overline{R^n}\,Q^n \\ &= \overline{R^n}\,Q^n(1 + S^n) + S^n\overline{Q^n} \\ &= \overline{R^n}\,Q^n + S^n\overline{Q^n} \qquad (b) \end{aligned}$$

Từ (a) và (b) ta có:

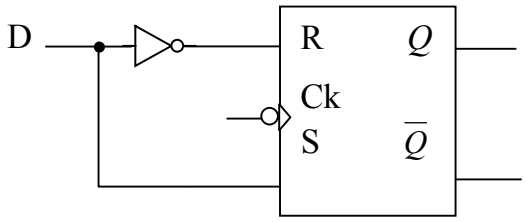
$$D^n Q^n + D^n \overline{Q^n} = \overline{R^n} Q^n + S^n \overline{Q^n}$$

So sánh 2 vế suy ra:

$$\begin{cases} \mathbf{S^n = D^n} \\ \mathbf{R^n = \overline{D^n}} \end{cases}$$

thỏa mãn điều kiện $R^nS^n = 0$.

Sơ đồ thực hiện: hình 3.76.



Hình 3.76. RSFF→ DFF

c. RSFF→JKFF:

So sánh 2 phương trình logic của RSFF và JKFF ta có:

$$Q^{n+1} = S^n + \overline{R^n} Q^n = J^n \overline{Q^n} + \overline{K^n} Q^n$$

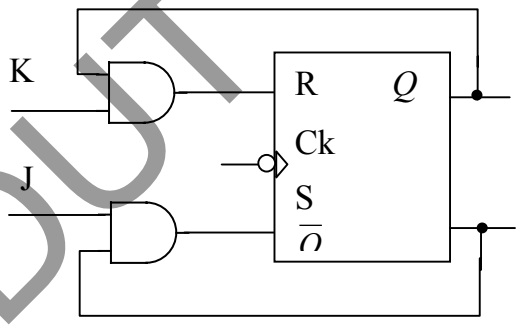
$$= J^n \overline{Q^n} + \overline{K^n} Q^n + Q^n \overline{Q^n} = J^n \overline{Q^n} + (\overline{K^n} + \overline{Q^n}) Q^n = J^n \overline{Q^n} + \overline{K^n Q^n} Q^n$$

So sánh ta có:

$$\begin{cases} \mathbf{S^n = J^n \overline{Q^n}} \\ \mathbf{R^n = K^n Q^n} \end{cases}$$

thỏa mãn điều kiện của RSFF.

Sơ đồ thực hiện: hình 3.77.



Hình 3.77. RSFF→ JKFF

4. JKFF chuyển đổi thành TFF, DFF, RSFF:

Như đã trình bày ở trên, JKFF là một FF vạn năng, có thể dùng JKFF để thay thế cho RSFF hoặc dùng JKFF thực hiện chức năng DFF, TFF. Sơ đồ thực hiện các mạch này như ở hình 3.67. Phần này tập trung chứng minh các biểu thức logic chuyển đổi từ JKFF sang các FF khác.

JKFF có phương trình trạng thái: $Q^{n+1} = J^n \overline{Q^n} + \overline{K^n} Q^n$

a. JKFF→ TFF:

TFF có phương trình logic: $Q^{n+1} = T^n \oplus Q^n = T^n \overline{Q^n} + \overline{T^n} Q^n$

So sánh với phương trình của JKFF ta suy ra logic chuyển đổi:

$$\begin{cases} \mathbf{J^n = T^n} \\ \mathbf{K^n = T^n} \end{cases}$$

b. JKFF→ DFF:

DFF có phương trình logic: $Q^{n+1} = D^n$

Viết lại biểu thức này ta có: $Q^{n+1} = D^n = D^n (Q^n + \overline{Q^n}) = D^n Q^n + D^n \overline{Q^n}$

So sánh với biểu thức của JKFF ta có logic chuyển đổi:

$$\begin{cases} \mathbf{J^n = D^n} \\ \mathbf{K^n} = \overline{D^n} \end{cases}$$

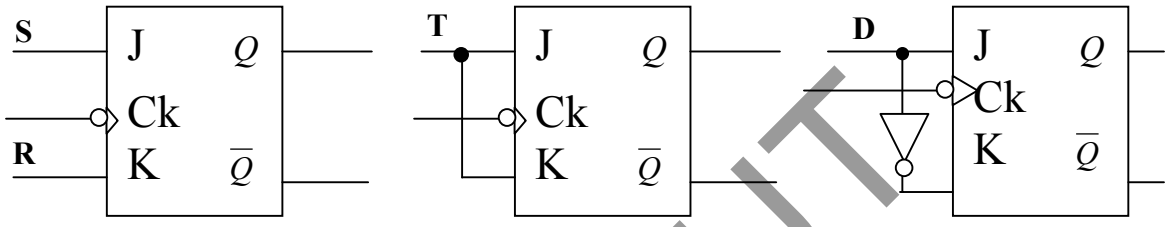
c. JKFF → RSFF:

Đối với RSFF có phương trình logic đã tìm được ở công thức (b):

$$Q^{n+1} = S^n + \overline{R^n}\,Q^n = S^n\overline{Q^n} + \overline{R^n}\,Q^n \qquad (b)$$

So sánh với phương trình logic của JKFF ta có logic chuyển đổi:

$$\begin{cases} \mathbf{J^n = S^n} \\ \mathbf{K^n = R^n} \end{cases}$$



Hình 3.67. Dùng JKFF thực hiện chức năng của RSFF, TFF, DFF

3.3.3. Phương pháp dùng bảng đầu vào kích

Để thực hiện chuyển đổi giữa các FF theo phương pháp này ta dựa vào bảng tín hiệu đầu vào kích của các FF và lập bảng Karnaugh, thực hiện tối giản để tìm logic chuyển đổi, phương pháp này có ưu điểm là trực quan, ít sai.

Bảng tín hiệu đầu vào kích tổng hợp như sau:

Q^n	Q^{n+1}	S^n	R^n	J^n	K^n	T^n	D^n
0	0	0	X	0	X	0	0
0	1	1	0	1	X	1	1
1	0	0	1	X	1	1	0
1	1	X	0	X	0	0	1

Ví dụ 1: Chuyển đổi từ JKFF → DFF dùng bảng đầu vào kích.

Ta có các hàm cần tìm:

$$J = f(D, Q^n) \text{ và } K = f(D, Q^n)$$

Dựa vào bảng đầu vào kích tổng hợp ta lập bảng Karnaugh:

J: Q^n \ D	0	1
0	0	1
1	X	X

J = D

K: Q^n \ D	0	1
0	X	X
1	1	0

K = $\overline{D}$

Tối giản theo dạng chính tắc 1 ta có: J = D và K = $\overline{D}$.

Ví dụ 2: Chuyển đổi từ JKFF → RSFF dùng bảng đầu vào kích.

Ta có các hàm cần tìm:

J = f (S, R, Q^n)

K = f (S, R, Q^n)

Dựa vào bảng đầu vào kích tổng hợp ta lập bảng Karnaugh:

J / Q^n \ SR	00	01	11	10
0	0	0	X	1
1	X	X	X	X

J = S

K / Q^n \ SR	00	01	11	10
0	X	X	X	X
1	0	1	X	0

K = R

Tối giản theo dạng chính tắc 1 ta có: J = S và K = R.

Các trường hợp chuyển đổi còn lại cũng hoàn toàn tương tự và kết quả chuyển đổi của cả 2 phương pháp (phương pháp biến đổi trực tiếp và phương pháp lập bảng Karnaugh) hoàn toàn giống nhau.

Chương 5
Hệ Tuần Tự
(Sequential Circuits)

Huỳnh Việt Thắng

Email: hvthang.ete.dut@gmail.com
Url: https://sites.google.com/site/hvthangete/

Đà Nẵng, 2013

Khái niệm và Đặc điểm

- Hệ Tuần Tự (Sequential Circuits) là hệ có các ngõ ra ở trạng thái kế tiếp là hàm của:
  - các ngõ vào ở trạng thái hiện tại, VÀ
  - các ngõ ra ở trạng thái hiện tại

  $Q^{n+1} = f$ (các tín hiệu vào hiện tại, Q^n)
- Đặc điểm
  - được thiết kế dựa trên Flip-Flop (FF)
  - ngõ ra thay đổi trạng thái ***đồng bộ*** với tín hiệu ***Clock***
  - có tính đồng bộ
  - có tính nhớ

2

Các hệ tuần tự tiêu biểu

- Bộ đếm (Counter)
  - Bộ đếm nối tiếp (Bộ đếm không đồng bộ)
- Máy trạng thái hữu hạn – Finite State Machine (FSM)
  - Bộ đếm song song (Bộ đếm đồng bộ)
  - Các ví dụ khác
- Bộ đếm hỗn hợp (Nối tiếp + Song song)
- Thanh ghi (Register)
- Thanh ghi dịch (Shift Registers)
- Bộ nhớ (Memory)

3

Bộ đếm (Counter)

- Bộ đếm được xây dựng trên cơ sở các FF ghép lại với nhau và hoạt động theo 1 bảng trạng thái cho trước
- Ứng dụng:
  - Tạo địa chỉ của lệnh điều khiển, địa chỉ bộ nhớ
  - Đếm số chu trình thực hiện phép tính / lệnh
  - Thu phát mã trong truyền số liệu
  - etc.

4

Bộ đếm (tt)

- Phân loại bộ đếm
  - Theo cơ sở các hệ đếm:
    - Đếm thập phân, đếm nhị phân
    - Đếm dung lượng 2^n, đếm dung lượng M bất kỳ
  - Theo hướng đếm
    - Đếm lên (nội dung bộ đếm tăng dần)
    - Đếm xuống (nội dung bộ đếm giảm dần)
    - Đếm thuận/nghịch
    - Đếm vòng
  - Theo tín hiệu điều khiển
    - **Đếm không đồng bộ (đếm nối tiếp) ← Quan tâm**
    - **Đếm đồng bộ (đếm song song) ← Quan tâm**
    - Đếm hỗn hợp

5

Bộ đếm không đồng bộ (nối tiếp)

- **Cấu trúc**: Bộ đếm không đồng bộ / bộ đếm nối tiếp bao gồm nhiều TFF (hoặc JKFF thực hiện chức năng của TFF) ghép nối tiếp với nhau – ngõ ra của FF đứng trước đóng vai trò là xung Clock cho FF đứng sau
- Ngõ vào T=1 (nếu dùng JKFF thì J=K=1)
- Hoạt động theo bộ mã duy nhất 8421
- Thực chất đếm nối tiếp chính là bộ chia tần số
- Phân loại
  - Đếm nối tiếp đếm lên
  - Đếm nối tiếp đếm xuống
  - Đếm nối tiếp đếm Modulo M
  - Đếm nối tiếp thuận nghịch

6

Thiết kế bộ đếm không đồng bộ

- **Cách ghép nối**: các TFF (hoặc JKFF thực hiện chức năng TFF với J=K=1) mắc nối tiếp với nhau tùy thuộc vào ***1) tín hiệu xung Clock*** (Ck) và ***2) hướng đếm***, tuân theo bảng sau đây

	Ck tích cực sườn xuống	Ck tích cực sườn lên
Đếm lên (Đếm thuận)	$Ck_{i+1} = Q_i$	$Ck_{i+1} = /Q_i$
Đếm xuống (Đếm ngược)	$Ck_{i+1} = /Q_i$	$Ck_{i+1} = Q_i$

$/Q_i = \overline{Q_i}$

7

Thiết kế bộ đếm không đồng bộ (tt)

- Số lượng FF cần sử dụng?
  - Bộ đếm có N trạng thái → cần dùng

    $\log_2 N$ TFF

    với **T=1** (hoặc JKFF đóng vai trò của TFF với J=K=1)

8

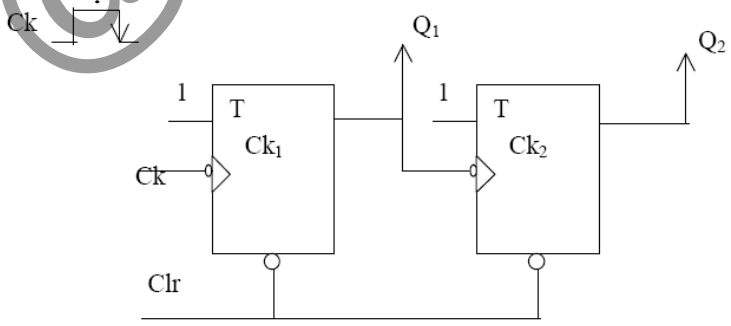
Ví dụ 1

- Thiết kế bộ đếm nối tiếp (đếm không đồng bộ) đếm lên, đếm 4 (0, 1, 2, 3), sử dụng TFF *có tín hiệu Ck tích cực theo sườn xuống*?
- Số lượng TFF cần dùng?
  - Đếm 4 trạng thái phân biệt (0→3) cần dùng $\log_2 4 = 2$ TFF
- Cách ghép nối các TFF: vì đếm lên, sử dụng TFF có tín hiệu xung clock Ck tích cực theo sườn xuống nên cần ghép nối tiếp các TFF sao cho: **$Ck_{i+1} = Q_i$**
  - $Ck_2 = Q1$
  - Ck_1 nhận xung clock từ nguồn phát bên ngoài
- Các ngõ vào dữ liệu $T_1=T_2=1$ (nối lên mức logic 1)

9

Ví dụ về đếm không đồng bộ (tt)

- Sơ đồ mạch

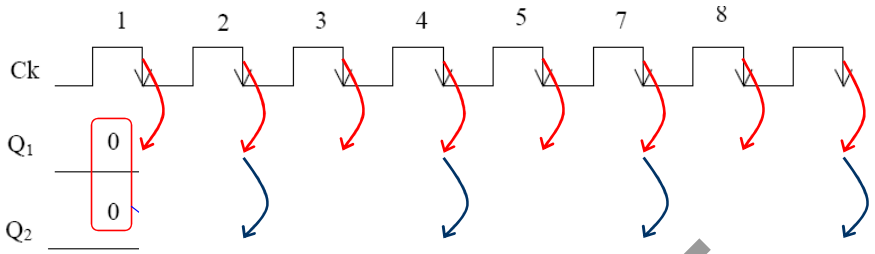


- Tác dụng của tín hiệu Clr (Clear): tín hiệu vào tích cực mức 0, dùng để xóa ngõ ra Q của FF về không
  - Nếu Clr = 0 → Q = 0

10

Ví dụ về đếm không đồng bộ (tt)

- Giản đồ thời gian và bảng trạng thái hoạt động



Xung vào	Trạng thái hiện tại		Trạng thái kế tiếp	
Ck	**Q_2**	**Q_1**	**Q_2**	**Q_1**
1	0	0	0	1
2	0	1	1	0
3	1	0	1	1
4	1	1	0	0

11

Các ví dụ khác về đếm nối tiếp

- Đếm lên, đếm 4, dùng TFF có Ck tích cực sườn lên
- Đếm xuống, đếm 4, TFF, Ck tích cực sườn xuống
- Đếm xuống, đếm 4, TFF, Ck tích cực sườn lên
- Đếm lên (xuống), đếm 8/16, TFF, Ck sườn xuống
- Đếm lên (xuống), đếm 8/16, TFF, Ck sườn lên
- Đếm lên (xuống) với JKFF (J=K=1): 8, 16
- Đếm Modulo M (dung lượng đếm khác 2^n)
- ..v..v...

➔ **SV tự đọc trong bài giảng!**

12

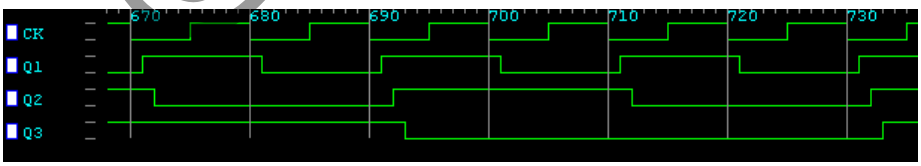
Ví dụ 2

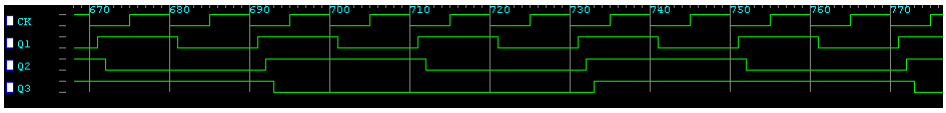
Thiết kế bộ đếm nối tiếp, đếm xuống, đếm 8 trạng thái dùng JKFF có xung Ck tác động tích cực theo sườn xuống. Lưu ý: trạng thái ban đầu của bộ đếm khi tín hiệu Clr=0 thì các ngõ ra đều bị xóa về 0.

- CircuitMaker Simulation
- **Proteus Simulation (nên sử dụng)**

13

Dạng sóng theo thời gian của Ví dụ 2





14

Ví dụ 3

Thiết kế bộ đếm nối tiếp, đếm lên, đếm 6 trạng thái dùng TFF có xung Ck tác động theo sườn lên. (gợi ý: xem bài giảng phần đếm modulo M)

15

Bộ đếm nối tiếp thuận/nghịch

- Đây là bộ đếm nối tiếp thực hiện chức năng vừa đếm lên (đếm thuận), vừa đếm xuống (đếm nghịch)
- Cần có thêm 1 tín hiệu điều khiển chiều đếm

Gọi X là tín hiệu điều khiển chiều đếm, ta quy ước:

+ Nếu X = 0 thì mạch đếm lên.
+ Nếu X = 1 thì đếm xuống.

Ta xét 2 trường hợp của tín hiệu Ck:

- **Xét tín hiệu Ck tác động sườn xuống:**

  Lúc đó ta có phương trình logic:

$$Ck_{i+1} = \overline{X}.Q_i + X\overline{Q_i} = X \oplus Q_i$$

- **Xét tín hiệu Ck tác động sườn lên:**

  Lúc đó ta có phương trình logic:

$$Ck_{i+1} = \overline{X}.\overline{Q_i} + X.Q_i = \overline{X \oplus Q_i}$$

16

Ưu, nhược điểm của đếm nối tiếp

- Ưu điểm: Đơn giản, dễ thiết kế
- Nhược điểm: Trễ truyền dẫn bị tích lũy → kết quả đếm ***không đồng bộ*** với tín hiệu xung clock ở đầu vào, nếu thời gian trễ tích lũy lớn hơn 1 chu kỳ tín hiệu xung clock thì sẽ đếm sai
- **Khắc phục**: Sử dụng bộ đếm song song, kết quả đếm xuất hiện đồng bộ với xung clock, thời gian trễ chỉ bằng thời gian trễ của 1 FF, và các ưu điểm khác của bộ đếm song song!

17

Các mạch đếm nối tiếp

- Đếm lên (đếm thuận)
- Đếm xuống (đếm nghịch)
- Đếm thuận / nghịch
- Đếm Modulo M

18

Nhắc lại: Các hệ tuần tự tiêu biểu

- *Bộ đếm (Counter)*
  - *Bộ đếm nối tiếp (Bộ đếm không đồng bộ)*
- **Máy trạng thái hữu hạn**
  Finite State Machine (FSM)
  - **Sequence Detector**
  - **Bộ đếm song song (Bộ đếm đồng bộ)**
  - **Các ví dụ khác**
- Bộ đếm hỗn hợp (Nối tiếp + Song song)
- Thanh ghi (Register)
- Thanh ghi dịch (Shift Registers)
- Bộ nhớ (Memory)

19

Finite State Machine (FSM)
Máy trạng thái hữu hạn

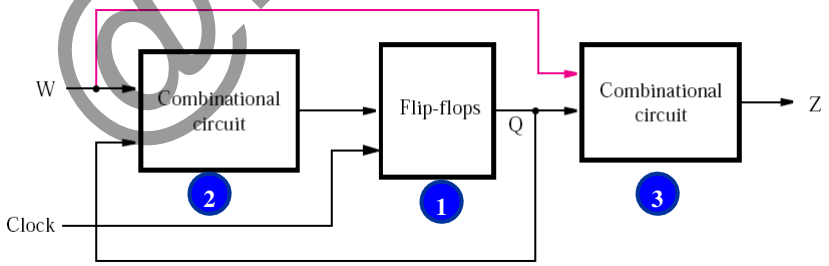
20

Khái niệm

- Mạch tuần tự (Sequential Circuits) được xây dựng trên cơ sở các Flip-Flop (FF) và các khối mạch logic tổ hợp (các cổng logic)
- Có thể chia làm 2 nhóm mạch
  - Mạch tuần tự đồng bộ (Synchronous Sequential Circuits)
  - Mạch tuần tự không đồng bộ (Asynchronous Sequential Circuits)
- ***Chỉ xem xét mạch tuần tự đồng bộ***
- Mạch tuần tự còn được gọi Máy trạng thái hữu hạn (Finite State Machine or FSM)
  - FSM loại Moore
  - FSM loại Mealy

21

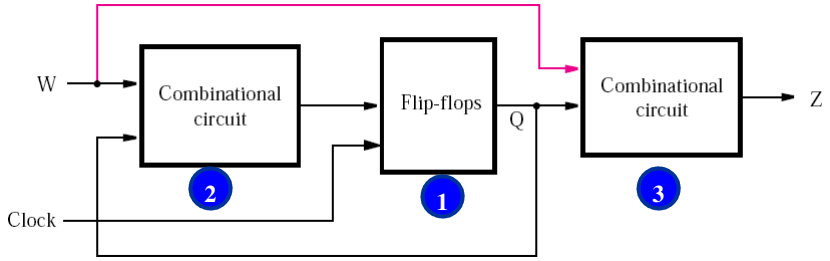
Sơ đồ khối FSM



- W: các tín hiệu vào; Q: trạng thái hiện tại của FSM; Z: tín hiệu ra của FSM
- 3 khối chính
  - Khối 1 (**State Register**): Khối thanh ghi trạng thái gồm các FFs xác định trạng thái hiện tại (***Current State***) của FSM
  - Khối 2 (**Next State Logic**): Logic tổ hợp xác định trạng thái kế tiếp (***Next State***) của FSM
  - Khối 3 (**Output Logic**): Logic tổ hợp xác định tín hiệu ra của FSM
- Clock: xung đồng bộ, FSM sẽ chuyển (cập nhật) trạng thái tại mỗi chu kỳ của xung Clock (giá trị các FFs được cập nhật)

22

FSM loại Moore vs. Mealy



- Tín hiệu ra Z phụ thuộc vào trạng thái Q hiện tại
- Tùy thuộc vào tính chất của tín hiệu ra Z có phụ thuộc vào tín hiệu vào W hay không mà FSM có thể phân chia thành 2 loại:
  - **FSM loại Moore**: tín hiệu ra chỉ phụ thuộc vào trạng thái hiện tại Q
  - **FSM loại Mealy**: tín hiệu ra phụ thuộc vào cả trạng thái hiện tại Q và tín hiệu vào W (tín hiệu màu đỏ trong sơ đồ khối)
- Edward Moore & George Mealy

23

FSM loại Moore

24

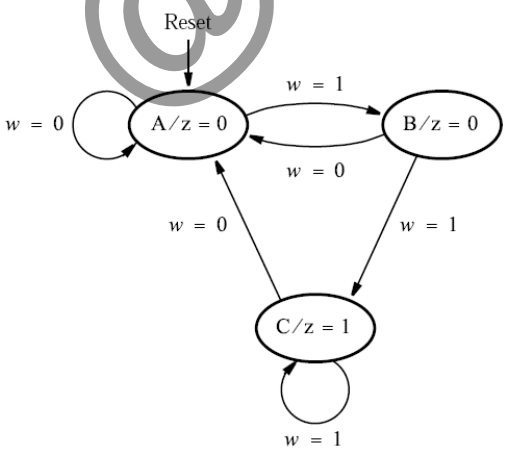
Ví dụ thiết kế FSM loại Moore đơn giản

- Thiết kế một mạch phát hiện chuỗi bít vào (sequence detector) hoạt động như sau:
  - mạch có 1 tín hiệu vào w, 1 tín hiệu ra z,
  - xung đồng bộ Ck tích cực sườn lên (positive edge)
  - tín hiệu ra z=1 nếu tại ngõ vào w xuất hiện 2 bit 1 liên tiếp trước đó, ngược lại tín hiệu ra z=0
- Ví dụ

Clock cycle:	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
w:	0	1	0	1	1	0	1	1	1	0	1
z:	0	0	0	0	0	1	0	0	1	1	0

25

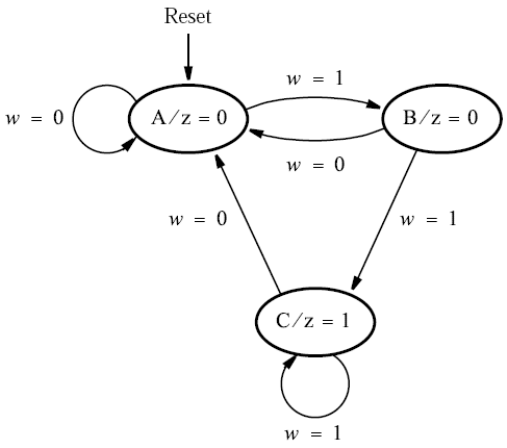
Bước 1. Vẽ giản đồ trạng thái FSM



- Đây là FSM loại Moore, có 3 trạng thái A, B, C
- Mỗi trạng thái được biểu diễn bằng 1 hình elip (tròn)
- Trạng thái ban đầu là A, khi có tín hiệu Reset
- Việc chuyển giữa các trạng thái tùy thuộc vào tín hiệu vào w, và dưới sự điều khiển đồng bộ của tín hiệu Clock
- Đây là FSM loại Moore nên tín hiệu ra z nằm bên trong elip biểu diễn trạng thái
  - Ở trạng thái A và B: z=0
  - Ở trạng thái C: z=1

26

Bước 2. Tối thiểu hóa số trạng thái



- Đối với mạch và giản đồ trạng thái đã cho thì số lượng trạng thái đã tối thiểu
- Trong trường hợp tổng quát thì tối thiểu hóa số trạng thái là cần thiết,
- Ít trạng thái ←→ mạch thực hiện đơn giản

27

Bước 3. Mã hóa trạng thái

- Có nhiều phương pháp khác nhau để mã hóa trạng thái của FSM, mỗi phương pháp sẽ cho độ phức tạp của mạch thực hiện khác nhau
- Trong ví dụ này để đơn giản chúng ta sử dụng các tổ hợp mã nhị phân 8421 để mã hóa các trạng thái
  - Có 3 trạng thái → cần tối thiểu 2 bits để mã hóa
  - Tương ứng cần sử dụng 2 FFs
  - Gán trạng thái:
    - A: 00
    - B: 01
    - C: 10

28

Bước 4. Lựa chọn loại FF sử dụng

- Cả 4 loại FF đều có thể sử dụng để thiết kế FSM
- Lựa chọn FF khác nhau dẫn đến mạch tổ hợp được thiết kế với độ phức khác nhau
- Bảng đầu vào kích của FF được lựa chọn sẽ được sử dụng để thiết kế FSM → ***1) tìm phương trình logic của mạch tổ hợp xác định trạng thái kế tiếp*** và ***2) tìm phương trình logic của mạch tổ hợp xác định tín hiệu ra z***
- **Trong ví dụ này để đơn giản chúng ta lựa chọn dùng DFF**

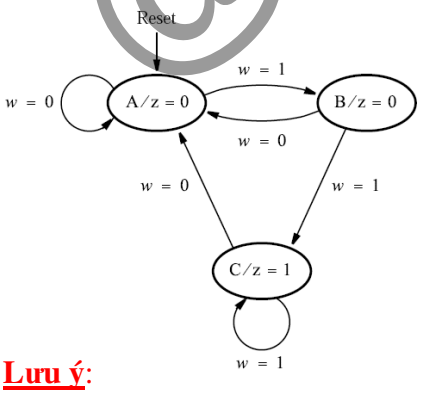
Bảng đầu vào kích của DFF

Q^n	Q^{n+1}	D^n
0	0	0
0	1	1
1	0	0
1	1	1

29

Bước 5. Thực hiện mạch tổ hợp

- Đầu tiên biểu diễn từ giản đồ trạng thái sang Bảng trạng thái



Present state	Next state $w=0$	Next state $w=1$	Output z
A	A	B	0
B	A	C	0
C	A	C	1

	Present state y_2y_1	Next state $w=0$ Y_2Y_1	Next state $w=1$ Y_2Y_1	Output z
A	00	00	01	0
B	01	00	10	0
C	10	00	10	1
	11	*dd*	*dd*	*d*

Lưu ý:
- d = don't care (X)
- $y_2 = Q_2^n$; $y_1 = Q_1^n$
- $Y_2 = Q_2^{n+1}$; $Y_1 = Q_1^{n+1}$

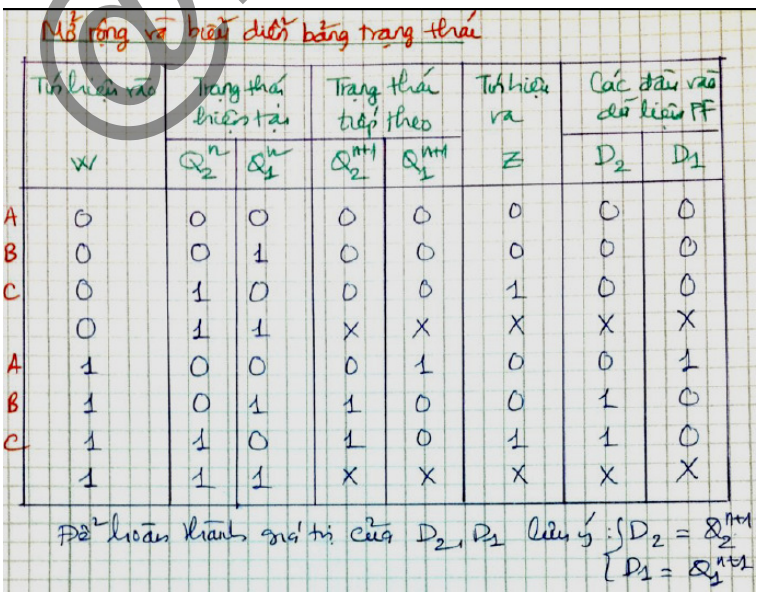
30

Bước 5. Thực hiện mạch tổ hợp (tt)

- Mở rộng và biểu diễn Bảng trạng thái bao gồm các tín hiệu sau đây:
  - Trạng thái hiện tại (Current State)
  - Trạng thái tiếp theo (Next State)
  - Các tín hiệu vào W
  - Các đầu vào dữ liệu của FF được sử dụng, lưu ý các đầu vào dữ liệu này được xác định dựa trên bảng đầu vào kích của FF
  - Các tín hiệu ra Z

31

Bước 5. Thực hiện mạch tổ hợp (tt)



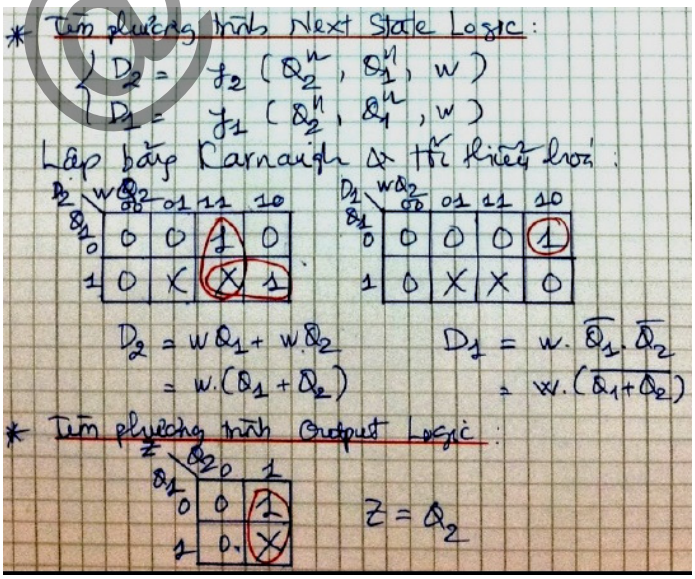
32

Bước 5. Thực hiện mạch tổ hợp (tt)

- Xác định phương trình của ***Next State Logic***
  - **Các đầu vào dữ liệu FF = f (Current State, Input W)**
- Xác định phương trình của ***Output Logic*** của FSM loại Moore
  - **Z = g (Current State)**
- Cho bài toán Sequence Detector cần tìm các hàm:
  - $D_1 = f_1 (Q_2^n, Q_1^n, w)$ và $D_2 = f_2 (Q_2^n, Q_1^n, w)$
  - $z = g (Q_2^n, Q_1^n)$
- Sử dụng K-Map để tối thiểu hóa các hàm để xác định
  - Next State Logic
  - Output Logic
- Vẽ sơ đồ thực hiện FSM

33

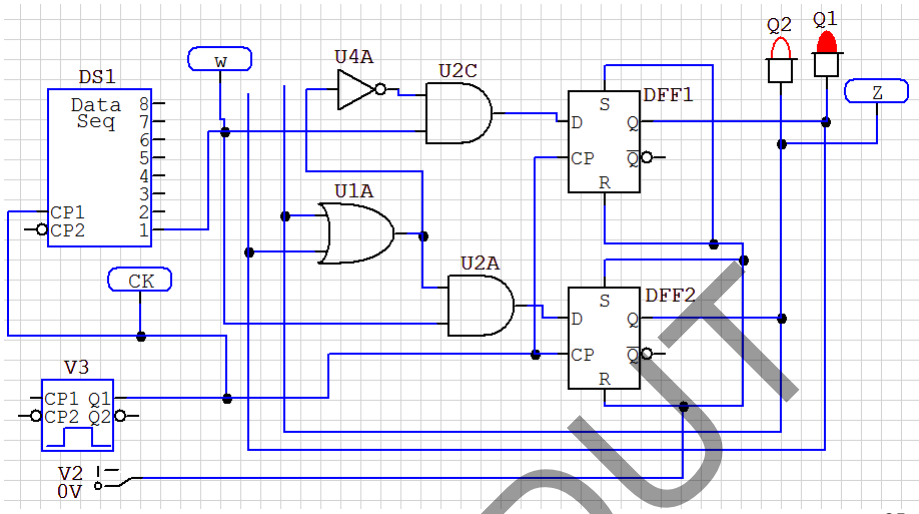
Bước 5. Thực hiện mạch tổ hợp (tt)



34

Bước 5. Thực hiện mạch tổ hợp (tt)

- Sơ đồ FSM Sequence Detector vẽ bằng CircuitMaker



35

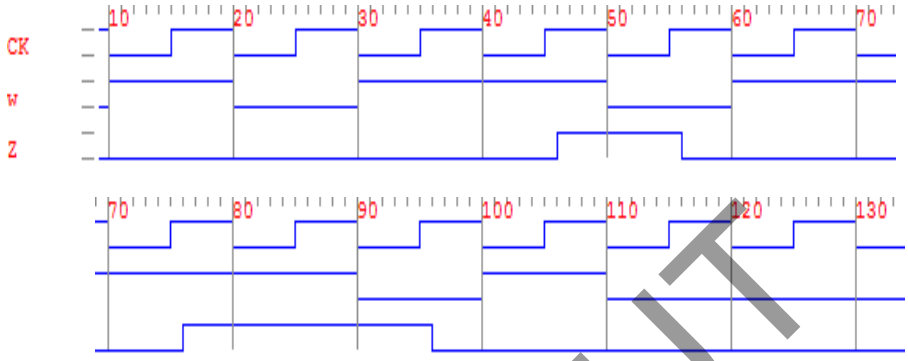
Bước 6. Phân tích tín hiệu theo thời gian

- Phân tích bằng tay
- Phân tích tự động: có thể sử dụng các phần mềm mô phỏng mạch số (e.g. CircuitMaker, Proteus, etc.)
- Mô tả FSM bằng HDL (Verilog hoặc VHDL) và phân tích mô phỏng bằng công cụ mô phỏng (e.g. ModelSIM của Mentor Graphics)

36

Bước 6. Phân tích tín hiệu theo thời gian (tt)

- Dạng sóng minh họa cho mạch Sequence Detector



37

Tóm tắt

Các bước thiết kế FSM

1. Biểu diễn giản đồ trạng thái
2. Tối thiểu hóa số trạng thái
3. Mã hóa trạng thái
4. Lựa chọn FF
5. Thực hiện mạch tổ hợp
6. Phân tích tín hiệu theo thời gian

38

Homework

- Thiết kế một mạch phát hiện chuỗi bít vào (sequence detector) hoạt động như sau:
  - mạch có 1 tín hiệu vào w, 1 tín hiệu ra z,
  - xung đồng bộ Ck tích cực sườn lên (positive edge)
  - **tín hiệu ra z=1 nếu tại ngõ vào w = 1001 hoặc w = 101**,
  - ngược lại tín hiệu ra z=0

39

Bộ đếm đồng bộ
(bộ đếm song song)

40

Bộ đếm đồng bộ (đếm song song)

- Bộ đếm đồng bộ, hay bộ đếm song song, là 1 trường hợp đặc biệt của máy trạng thái hữu hạn FSM → cách thiết kế và phân tích tương tự như FSM
  - FSM loại Moore
  - FSM loại Mealy

41

Bộ đếm đồng bộ (đếm song song)

- Đây là bộ đếm có các ngõ ra thay đổi đồng bộ (cùng 1 lúc) với tín hiệu xung Clock (Ck)
- Đặc điểm:
  - Sử dụng FF bất kỳ (T, RS, D, JK)
  - Quy luật đếm bất kỳ (bộ mã bất kỳ: 8421, Gray, .v..v..)
  - Không phụ thuộc xung clock tích cực sườn âm/dương
- Sử dụng bảng đầu vào kích để thiết kế bộ đếm đồng bộ

Q^n	Q^{n+1}	S^n	R^n	J^n	K^n	T^n	D^n
0	0	0	X	0	X	0	0
0	1	1	0	1	X	1	1
1	0	0	1	X	1	1	0
1	1	X	0	X	0	0	1

42

Thiết kế bộ đếm đồng bộ

- Yêu cầu thiết kế:
  - Bộ đếm đồng bộ (đếm song song)
  - Số trạng thái đếm N, quy luật đếm, và loại FF sử dụng
- **Các bước thiết kế bộ đếm đồng bộ**
  1. Xác định số lượng FF cần sử dụng: tương tự đếm nối tiếp
  2. Xây dựng bảng trạng thái mô tả hoạt động bộ đếm
  3. Dựa vào bảng đầu vào kích của FF tương ứng để mở rộng bảng trạng thái và xây dựng bảng giá trị các ngõ vào dữ liệu của FF tương ứng theo các ngõ ra Q
  4. Tối thiểu hóa tìm hàm các ngõ vào dữ liệu của FF theo **các ngõ ra Q ở trạng thái hiện tại**
  5. Vẽ sơ đồ mạch thực hiện

43

Ví dụ thiết kế bộ đếm đồng bộ

- Thiết kế mạch đếm đồng bộ, đếm 5, đếm lên, theo mã BCD 8421 sử dụng JKFF có xung Ck tích cực sườn lên?

1. Số JKFF cần dùng: 3 JKFF, có 3 ngõ ra Q3, Q2, Q1
   - *Vai trò của xung Ck tích cực sườn lên?*
2. Xây dựng bảng trạng thái mô tả hoạt động bộ đếm

Xung vào	Trạng thái hiện tại			Trạng thái kế tiếp		
Ck	**Q_3**	**Q_2**	**Q_1**	**Q_3**	**Q_2**	**Q_1**
1	0	0	0	0	0	1
2	0	0	1	0	1	0
3	0	1	0	0	1	1
4	0	1	1	1	0	0
5	1	0	0	0	0	0

44

Ví dụ thiết kế bộ đếm đồng bộ (tt)

3. Mở rộng bảng trạng thái, dựa vào bảng đầu vào kích của JKFF để xây dựng hàm các ngõ vào dữ liệu J và K theo các ngõ ra

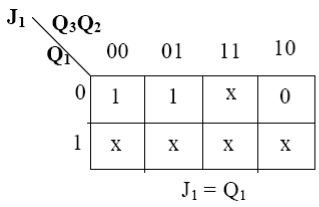
Q^n	Q^{n+1}	J^n	K^n
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Xung vào	Trạng thái hiện tại			Trạng thái kế tiếp								
	Q_3	Q_2	Q_1	Q_3	Q_2	Q_1	J_3	K_3	J_2	K_2	J_1	K_1
1	0	0	0	0	0	1	0	X	0	X	1	X
2	0	0	1	0	1	0	0	X	1	X	X	1
3	0	1	0	0	1	1	0	X	X	0	1	X
4	0	1	1	1	0	0	1	X	X	1	X	1
5	1	0	0	0	0	0	X	1	0	X	0	X

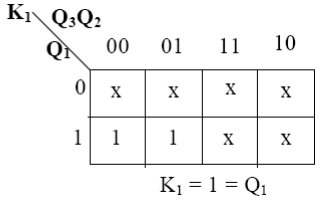
45

Ví dụ thiết kế bộ đếm đồng bộ (tt)

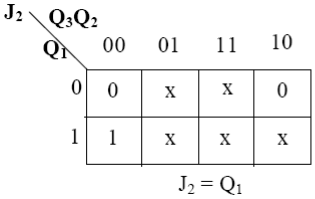
4. Tối thiểu hóa tìm hàm của J_i và K_i theo các ngõ ra Q_i ở trạng thái hiện tại (i = 1, 2, 3)



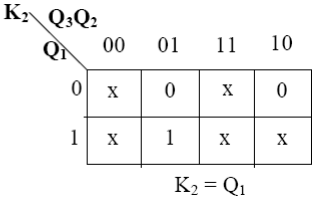
$J_1 = Q_1$



$K_1 = 1 = Q_1$



$J_2 = Q_1$



$K_2 = Q_1$

46

Ví dụ thiết kế bộ đếm đồng bộ (tt)

J_3 \ Q_3Q_2 / Q_1	00	01	11	10
0	0	0	x	X
1	0	1	x	x

$J_2 = Q_1Q_2$

K_3 \ Q_3Q_2 / Q_1	00	01	11	10
0	x	0	x	0
1	x	1	x	x

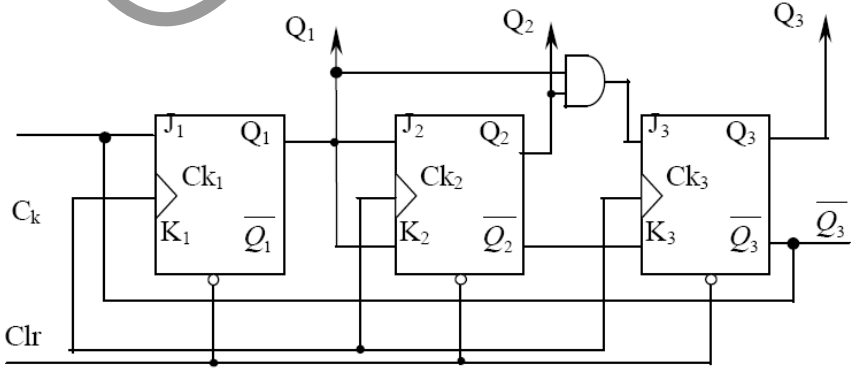
$K_3 = 1 = Q_3 = \overline{Q_1} = \overline{Q_2}$

Lưu ý: Khi thiết kế tính toán ta dùng các phương pháp tối thiểu để đưa về phương trình logic tối giản. Nhưng trong thực tế thì đôi lúc không phải như vậy. Ví dụ: $K_3 = 1$, $K_3 = Q_3$ hay $K_3 = \overline{Q_2}$ đều đúng, nhưng khi lắp ráp thực tế ta chọn $K3 = \overline{Q_2}$ để tránh dây nối dài gây nhiễu cho mạch.

47

Ví dụ thiết kế bộ đếm đồng bộ (tt)

5. Vẽ sơ đồ thực hiện bộ đếm



Giải thích hoạt động của mạch: xem bài giảng

48

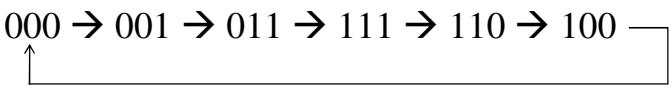
Các ví dụ khác về đếm đồng bộ

- Thiết kế bộ đếm đồng bộ bằng cách kết hợp các yêu cầu sau đây:
  - Số trạng thái bộ đếm N = 5, 6, 7, 8, 9, 10, 12, 14, 16, ...
  - Chiều đếm: đếm lên, đếm xuống, đếm thuận nghịch
  - Dùng DFF, JKFF, RSFF, TFF
  - Mã 8421, Mã Gray, 5421, 84-2-1, 2421, 5121, ...
- Đếm đồng bộ thuận nghịch: xem bài giảng
- Đếm hỗn hợp = Đếm nối tiếp + Đếm song song
  - Tăng dung lượng đếm

49

Các ví dụ khác về đếm đồng bộ (tt)

- ***Nhận xét***: có thể xem thiết kế bộ đếm đồng bộ là một trường hợp đặc biệt và đơn giản của thiết kế các máy trạng thái hữu hạn Finite State Machines (FSM)
- Ví dụ: Thiết kế bộ đếm đồng bộ thực hiện đếm theo quy luật sau đây sử dụng RSFF:



➔ Homework? why not?

50

Bài tập

- Thiết kế bộ đếm đồng bộ, đếm thuận nghịch, đếm 4, theo mã 8421, với DIR là tín hiệu điều khiển chiều đếm
  - DIR = 0: đếm lên
  - DIR = 1: đếm xuống
- Khi đếm đến 3 thì báo ngõ ra Y=1 (***i.e., khi trạng thái hiện tại của bộ đếm =3 thì ngõ ra Y=1***)
- Sử dụng JKFF với xung clock tích cực theo sườn lên

51

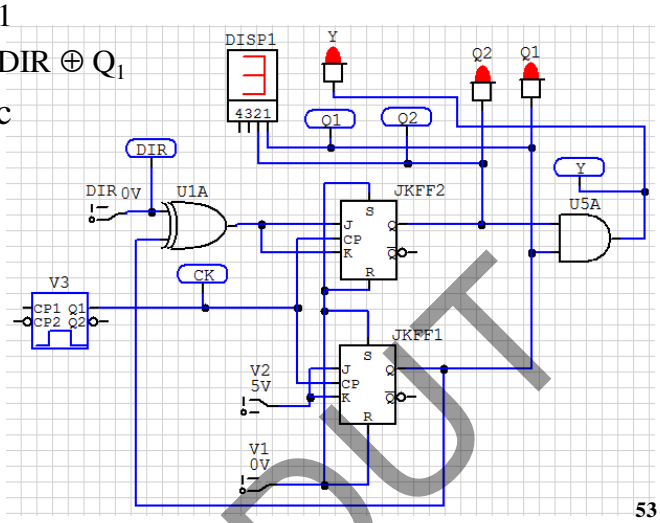
Bài tập (tt)

- Đây là máy trạng thái loại Moore
- Các bước thiết kế:
  1. Vẽ giản đồ trạng thái
  2. Tối thiểu hóa số trạng thái
     - *đã tối thiểu*
  3. Mã hóa trạng thái
     - *Mã lựa chọn là mã nhị phân 8421*
  4. Lựa chọn FF
     - *Đã lựa chọn sử dụng JKFF*
  5. Thực hiện mạch tổ hợp cho Next State Logic & Output Logic (và vẽ sơ đồ mạch thực hiện)
  6. Phân tích tín hiệu theo thời gian

52

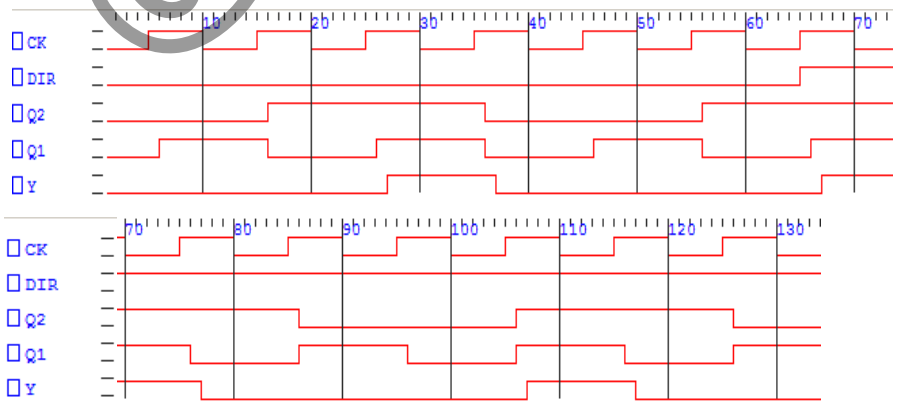
Bài tập (tt)

- Next State Logic
  - $J_1 = K_1 = 1$
  - $J_2 = K_2 = DIR \oplus Q_1$
- Output Logic
  - $Y = Q_1Q_2$



Bài tập (tt)

- Phân tích tín hiệu theo thời gian



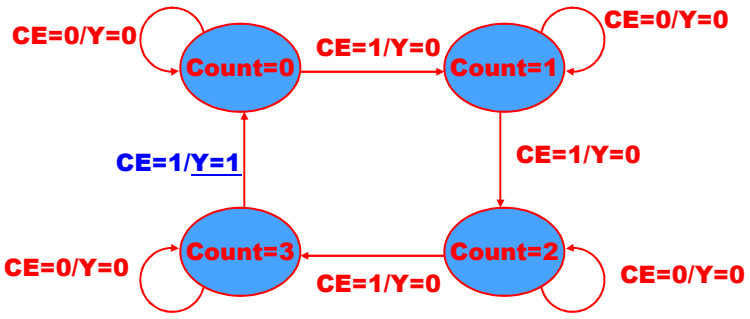
- Nhận xét về dạng sóng của Q_1, Q_2 và Y?

FSM loại Mealy

55

Máy trạng thái hữu hạn loại Mealy

- **Ví dụ 1**: Thiết kế bộ đếm đồng bộ, có 4 trạng thái, sử dụng JKFF có xung Clock tác động tích cực theo sườn âm (sườn xuống). Bộ đếm sẽ đếm lên mỗi khi tín hiệu CE=1 (Count Enable). Nếu tín hiệu cho phép đếm CE=1 và giá trị đếm = 3 thì tín hiệu ra Y=1.



56

Máy trạng thái hữu hạn loại Mealy (tt)

- Kết quả:
  - $J_2 = K_2 = CE.Q_1$
  - $J_1 = K_1 = CE$
  - $Y = CE.Q_1.Q_2$

57

Máy trạng thái hữu hạn loại Mealy (tt)

- Đặc điểm:
  - Tín hiệu ra được xác định cho mỗi trạng thái **và các tín hiệu vào tại trạng thái đó**
  - Tín hiệu ra phụ thuộc vào trạng thái hiện tại **và các giá trị đầu vào tại trạng thái đó**
  - Do đó giá trị tín hiệu ra **ghi tại mũi tên chuyển trạng thái**

58

Máy trạng thái hữu hạn loại Mealy (tt)

- Các bước thiết kế FSM loại Mealy tương tự như các bước thiết kế FSM loại Moore (6 bước)
- ***FSM loại Mealy***: Do tín hiệu ra phụ thuộc vào các giá trị đầu vào tại mỗi trạng thái → nên phương trình logic xác định tín hiệu ra (Output Logic) ở **Bước 5** như sau:

Z = g (Current State, Input W)

59

Nhắc lại

Các bước thiết kế FSM

1. *Biểu diễn giản đồ trạng thái*
2. *Tối thiểu hóa số trạng thái*
3. ***Mã hóa trạng thái***
4. ***Lựa chọn FF***
5. *Thực hiện mạch tổ hợp*
6. *Phân tích tín hiệu theo thời gian*

Mã hóa trạng thái và lựa chọn FF như thế nào?

60

Các phương pháp mã hóa trạng thái

- n trạng thái cần ít nhất $\log_2 n$ flip-flops.
- Có tất cả ***n!*** khả năng mã hoá (n lựa chọn cho trạng thái đầu tiên, n-1 cho trạng thái thứ 2 ...)

No.	s0	s1	s2	s3	No.	s0	s1	s2	s3
1	00	01	10	11	13	10	00	01	11
2	00	01	11	10	14	10	00	11	01
3	00	10	01	11	15	10	01	00	11
4	00	10	11	01	16	10	01	11	00
5	00	11	01	10	17	10	11	00	01
6	00	11	10	01	18	10	11	01	00
7	01	00	10	11	19	11	00	01	10
8	01	00	11	10	20	11	00	10	01
9	01	10	00	11	21	11	01	00	10
10	01	10	11	00	22	11	01	10	00
11	01	11	00	10	23	11	10	00	01
12	01	11	10	00	24	11	10	01	00

61

Các phương pháp mã hóa trạng thái (tt)

- Có cần thiết phải lựa chọn cách mã hoá?
  - Có, bởi mỗi sự lựa chọn sẽ cho ta độ phức tạp của mạch tổ hợp cũng như trễ của toàn bộ mạch
- Các kiểu mã hoá thường dùng:
  - Straightforward (mã nhị phân 8421)
  - Minimum-bit-change (thường dùng mã Gray)
  - One-hot

62

Mã hóa StraightForward

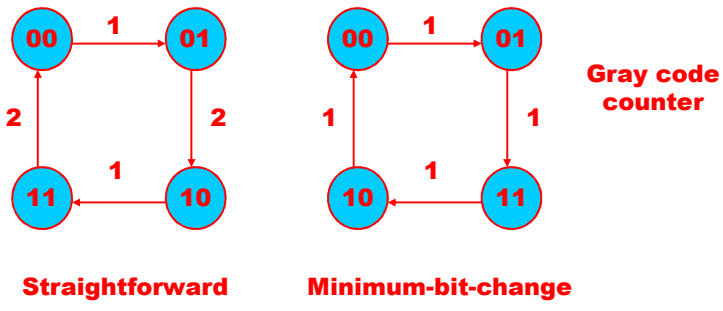
- Kiểu mã hoá này sử dụng giá trị nhị phân của thứ tự trạng thái để làm mã cho trạng thái đó
  - $s_0 \rightarrow 000$,
  - $s_5 \rightarrow 101$,
  - ...

	Present state $y_2 y_1$	Next state $w = 0$ $Y_2 Y_1$	Next state $w = 1$ $Y_2 Y_1$	Output z
A	00	00	01	0
B	01	00	10	0
C	10	00	10	1
	11	*dd*	*dd*	*d*

63

Mã hóa Minimum-Bit-Change

- Mã sẽ được gán cho trạng thái sao cho tổng số bít thay đổi khi chuyển trạng thái của tất cả các trạng thái là nhỏ nhất
- Phương pháp mã hoá này hay được sử dụng khi muốn tối thiểu hoá kích thước chip cũng như công suất tiêu thụ



64

Mã hóa One-hot

- Mỗi trạng thái ứng với một flip-flop, Q của 1 FF =1, Q của các FF khác =0
- Dùng khi số lượng trạng thái nhỏ
- Rất dễ thực hiện, mạch tổ hợp đơn giản
- Cấu trúc của FPGA rất phù hợp với kiểu mã hoá này

65

Vấn đề lựa chọn FF sử dụng

- **JKFF**
  - Đây là loại flip-flop đắt nhất
  - Khó thiết kế nhất
  - Có nhiều trạng thái don't cares nên mạch tổ hợp nhanh và rẻ nhất
- **SRFF**
  - flip-flop rẻ
  - Khó thiết kế

66

Vấn đề lựa chọn FF sử dụng (tt)

- **DFF**
  - flip-flop rẻ
  - Dễ thiết kế nhất
  - Không có don't cares dẫn tới mạch tổ hợp phức tạp và chậm nhất
- **TFF**
  - flip-flop rẻ
  - Dễ thiết kế
  - Không có don't cares dẫn tới mạch tổ hợp phức tạp và chậm nhất
  - Thường được dùng để thiết kế bộ đếm và bộ chia tần

67

Bài tập: Mạch điều khiển đèn giao thông

- Thiết kế mạch điều khiển đèn giao thông theo yêu cầu sau đây:
  - Thời gian đèn vàng: $\mathbf{t_{vàng} = 5\ s}$
  - Thời gian đèn xanh: $\mathbf{t_{xanh} = 20\ s}$
  - Thời gian đèn đỏ: $\mathbf{25\ s = t_{xanh} + t_{vàng}}$
- Hiển thị và đếm lùi thời gian của các đèn

68

Shift Register
(Thanh ghi dịch)

69

Khái niệm về thanh ghi

- Một FF có thể lưu trữ 1 bit thông tin
- Tập hợp gồm ***n*** FFs được sử dụng để lưu trữ ***n*** bit thông tin, chẳng hạn 1 số nhị phân *n*-bit, được gọi là một thanh ghi *n*-bit (*n-bit* register).
- Thông thường DFF được sử dụng để chế tạo thanh ghi (hoặc sử dụng các loại FF khác thực hiện chức năng của DFF)
- Các DFF được ghép nối theo 1 cấu trúc, trong đó ngõ vào Clock nối chung với nhau (*common clock*) để điều khiển hoạt động của thanh ghi

70

Thanh ghi dịch (Shift Register)

- Thông thường, thanh ghi có khả năng dịch chuyển nội dung chứa bên trong đó, và được gọi là ***Thanh ghi dịch*** – Shift Register
- Qui luật ghép nối để tạo Shift Register
  - **Right-Shift Register**: ngõ ra của DFF đứng trước được nối với ngõ vào D của DFF sau ($D_{i+1} = Q_i$) → thanh ghi có khả năng dịch phải
  - **Left-Shift Register**: ngõ ra của DFF đứng sau được nối với ngõ vào D của DFF đứng trước ($D_i = Q_{i+1}$) → thanh ghi có khả năng dịch trái.

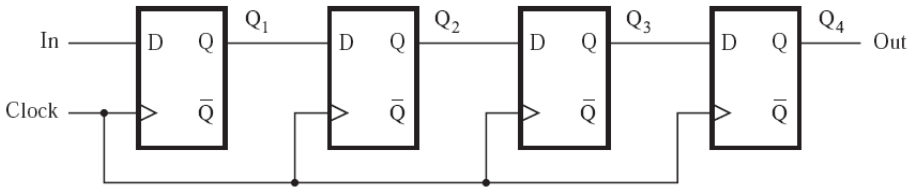
71

Phân loại

- Theo kích thước thanh ghi: 4 bit, 8 bit, 16 bit, 32 bit,...
- Theo hướng dịch chuyển dữ liệu:
  - Thanh ghi dịch trái
  - Thanh ghi dịch phải
  - Thanh ghi vừa dời phải vừa dời trái
- Theo ngõ vào dữ liệu:
  - Ngõ vào dữ liệu nối tiếp
  - Ngõ vào dữ liệu song song
- Phân loại theo ngõ ra:
  - Ngõ ra nối tiếp.
  - Ngõ ra song song.
  - Ngõ ra vừa nối tiếp vừa song song.

72

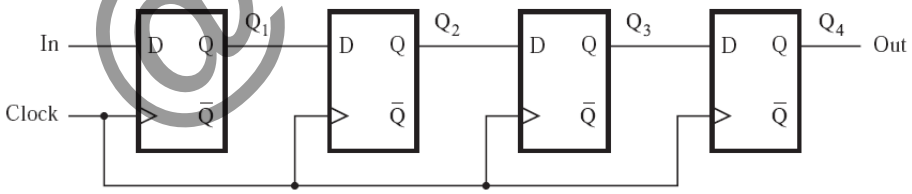
Thanh ghi dịch phải



- Xét 1 thanh ghi dịch phải 4-bit đơn giản
  - In: ngõ vào dữ liệu nối tiếp
  - Out: ngõ ra dữ liệu nối tiếp (dữ liệu dịch phải)
  - Clock: xung clock chung tích cực theo sườn lên

73

Thanh ghi dịch phải (tt)

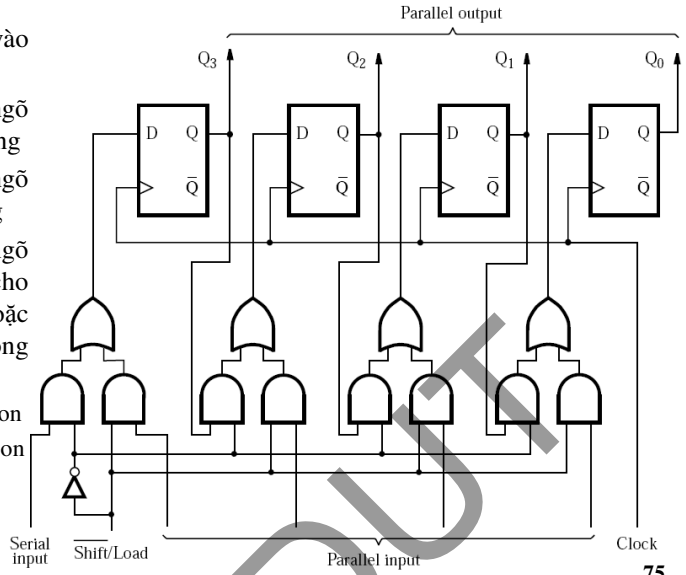


	In	Q_1	Q_2	Q_3	Q_4 = Out
t_0	1	0	0	0	0
t_1	0	1	0	0	0
t_2	1	0	1	0	0
t_3	1	1	0	1	0
t_4	1	1	1	0	1
t_5	0	1	1	1	0
t_6	0	0	1	1	1
t_7	0	0	0	1	1

74

Thanh ghi dịch phải với ngõ ra song song

- Serial Input: ngõ vào dữ liệu nối tiếp
- Parallel Input: ngõ vào dữ liệu song song
- Parallel Output: ngõ ra dữ liệu song song
- /Shift or Load: ngõ vào điều khiển cho phép dịch (Shift) hoặc nạp dữ liệu song song (Load)
  - 0 : Shift operation
  - 1 : Load operation

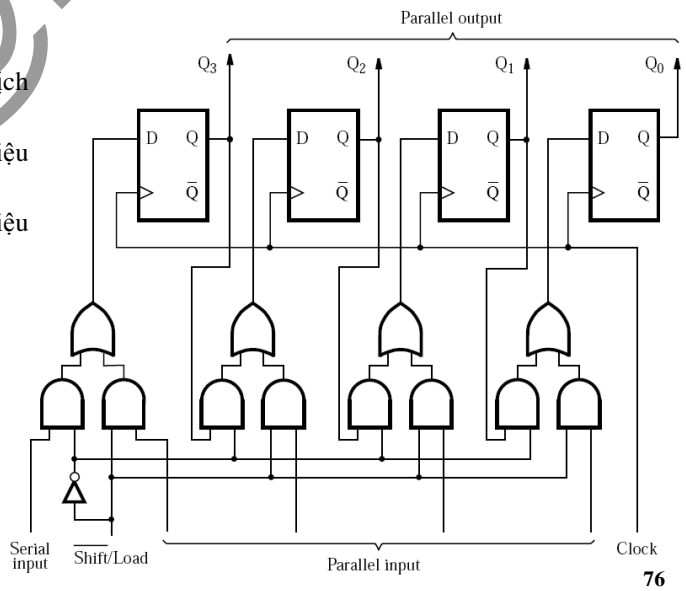


75

Thanh ghi dịch phải với ngõ ra song song (tt)

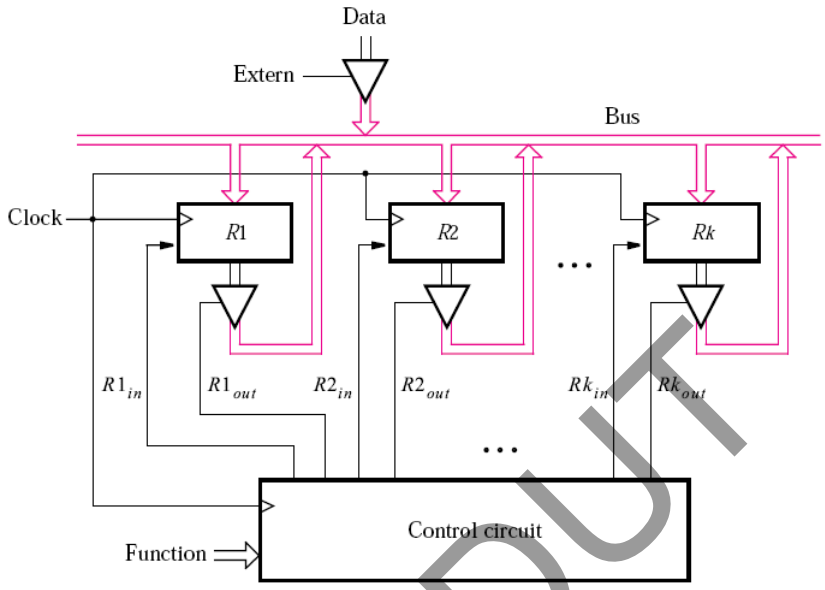
Giải thích:

- Hoạt động dịch chuyển dữ liệu?
- Hoạt động nạp dữ liệu vào song song?
- Hoạt động nạp dữ liệu vào nối tiếp?



76

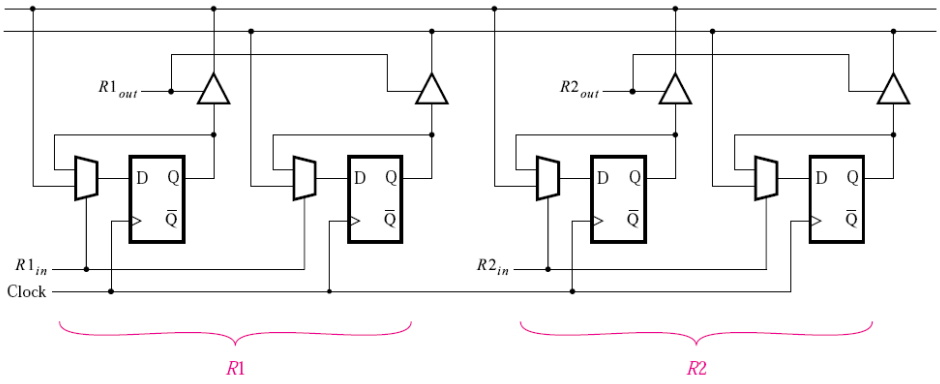
Bus structure



77

Bus structure (tt)

- Details for connecting registers to a bus: Two 2-bit registers *R1* and *R2* are connected to a bus



78

Các chủ đề khác về thanh ghi dịch

- Thanh ghi dịch trái?
- Thanh ghi vừa dịch trái vừa dịch phải?

79

Memory (Bộ nhớ bán dẫn)

- Khái niệm
- ROM (Read-Only-Memory)
- RAM (Random Access Memory)
- Memory Organization

80

Một số khái niệm

- **Tế bào nhớ (Memory cell)** là thiết bị hay mạch điện tử dùng để lưu trữ 1 bit.
  - ví dụ: FF để lưu trữ 1 bit, tụ điện khi nạp điện thì lưu trữ 1 bit, hoặc một điểm trên băng từ.
- **Từ nhớ (MEMORY WORD)** là nhóm các bit ở trong một bộ nhớ.
  - Ví dụ: Một thanh ghi gồm 8 DFF có thể lưu trữ từ nhớ với độ rộng là 8 bit.
  - Trong thực tế, kích thước của từ nhớ có thể thay đổi trong các loại máy tính từ 4 đến 64 bit.

81

Một số khái niệm (tt)

- **BYTE:** Một nhóm từ nhớ 8 bit.
- **DUNG LƯỢNG BỘ NHỚ** chỉ khả năng lưu trữ của bộ nhớ.
  - Ví dụ: $1K = 2^{10}$; $2K = 2^{11}$; $4K = 2^{12}$; $1M = 2^{20}$.
- **ĐỊA CHỈ** dùng để xác định các vùng của các từ trong bộ nhớ.
  - Xét bộ nhớ gồm 16 ngăn nhớ tương đương 16 từ, ta cần dùng 4 đường địa chỉ ($2^4 = 16 \rightarrow$ có 4 đường địa chỉ) → có mối quan hệ giữa địa chỉ và dung lượng bộ nhớ.
  - Ví dụ : Để quản lý được bộ nhớ có dung lượng là 8 Kbytes thì cần 13 đường địa chỉ (8 KB = 2^{13} bytes).

82

Một số khái niệm (tt)

- **RAM** (**R**andom **A**ccess **M**emory) là Bộ nhớ truy xuất ngẫu nhiên, đọc viết tùy ý, còn được gọi là RWM (Read/Write Memory). Đây là loại bộ nhớ cho phép đọc dữ liệu chứa bên trong ra ngoài và cho phép nhập dữ liệu từ bên ngoài vào trong.
- **ROM** (**R**ead **O**nly **M**emory) là Bộ nhớ chỉ đọc, chỉ cho phép đọc dữ liệu trong ROM ra ngoài mà không cho phép dữ liệu ghi dữ liệu từ bên ngoài vào trong bộ nhớ. Việc ghi dữ liệu vào ROM thông thường được thực hiện trong quá trình chế tạo hoặc trong quá trình sử dụng bằng các thiết bị ghi đặc biệt.

83

Một số khái niệm (tt)

- **BỘ NHỚ KHÔNG BAY HƠI** là khái niệm dùng để chỉ loại bộ nhớ mà dữ liệu không mất đi khi mất nguồn điện, còn gọi là **Non-Volatile**.
  - ROM là bộ nhớ không bay hơi (non-volatile)
- **BỘ NHỚ BAY HƠI** là khái niệm dùng để chỉ loại bộ nhớ lưu trữ dữ liệu khi còn nguồn điện và khi mất nguồn điện thì dữ liệu sẽ bị mất, còn gọi là **Volatile**.
  - RAM là loại bộ nhớ bay hơi (volatile)

84

Một số khái niệm (tt)

- **HOẠT ĐỘNG ĐỌC (READ)**
  - Đọc là xuất dữ liệu từ bộ nhớ ra ngoài.
  - Để đọc nội dung một ô nhớ cần thực hiện:
    - Đưa địa chỉ tương ứng vào các đường địa chỉ A.
    - Khi tín hiệu điều khiển đọc tác động dữ liệu chứa trong các ngăn nhớ tương ứng với vùng địa chỉ yêu cầu sẽ được xuất ra ngoài trên các đường dữ liệu.
- **HOẠT ĐỘNG GHI (WRITE)**
  - Viết là ghi dữ liệu từ bên ngoài vào bên trong bộ nhớ.
  - Muốn ghi dữ liệu vào bộ nhớ cần thực hiện:
    - Đặt các địa chỉ tương ứng lên các đường địa chỉ.
    - Đặt dữ liệu cần viết vào bộ nhớ lên các đường dữ liệu.
    - Tích cực tín hiệu điều khiển ghi.
    - Khi ghi dữ liệu từ bên ngoài vào bên trong bộ nhớ thì dữ liệu cũ sẽ mất đi và được thay thế bằng dữ liệu mới.

85

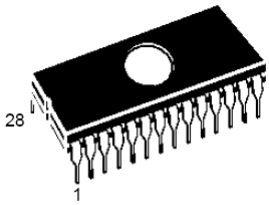
ROM (Read Only Memory)

- **MROM (Mask ROM):** nội dung bộ nhớ được lập trình trước bởi nhà sản xuất. Chỉ có tính kinh tế khi sản xuất hàng loạt nhưng lại không phục hồi được khi chương trình bị sai hỏng.
- **PROM (Programmable ROM):** Đây là loại ROM cho phép lập trình bởi nhà sản xuất. Nếu hỏng không phục hồi được.
- **EPROM (Erasable PROM):** là loại PROM có thể xóa và lập trình lại. Có hai loại EPROM:
  - EPROM được xóa bằng tia cực tím (UV EPROM - Ultralviolet EPROM), lập trình bằng xung điện.
  - EPROM xóa và lập trình bằng xung điện (EEPROM - Electrically EPROM).
  - Tuổi thọ của EPROM phụ thuộc vào thời gian xóa.
- **FLASH:** là loại bộ nhớ được phát triển từ EEPROM, tương tự như EEPROM nhưng ưu điểm hơn là cho phép xóa theo khối.

86

EPROM 2764

- 8K x 8 (64K) UV EPROM
- Xóa bằng tia cực tím
- Lập trình bằng xung điện 12.5V
- Thời gian lập trình nhanh (< 1 min)



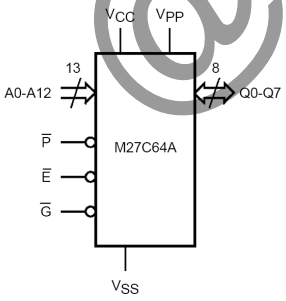
FDIP28W (F)

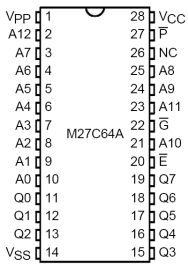
PLCC32 (C)

87

EPROM 2764 (tt)

- Sơ đồ khối, các chân tín hiệu, các mode hoạt động





A0 - A12	Address Inputs
Q0 - Q7	Data Outputs
$\overline{E}$	Chip Enable
$\overline{G}$	Output Enable
$\overline{P}$	Program
V_{PP}	Program Supply
V_{CC}	Supply Voltage
V_{SS}	Ground

Mode	$\overline{E}$	$\overline{G}$	$\overline{P}$	A9	V_{PP}	Q0 - Q7
Read	V_{IL}	V_{IL}	V_{IH}	X	V_{CC}	Data Out
Output Disable	V_{IL}	V_{IH}	V_{IH}	X	V_{CC}	Hi-Z
Program	V_{IL}	V_{IH}	V_{IL} Pulse	X	V_{PP}	Data In
Verify	V_{IL}	V_{IL}	V_{IH}	X	V_{PP}	Data Out
Program Inhibit	V_{IH}	X	X	X	V_{PP}	Hi-Z
Standby	V_{IH}	X	X	X	V_{CC}	Hi-Z
Electronic Signature	V_{IL}	V_{IL}	V_{IH}	V_{ID}	V_{CC}	Codes

Note: X = V_{IH} or V_{IL}, V_{ID} = 12V ± 0.5V

88

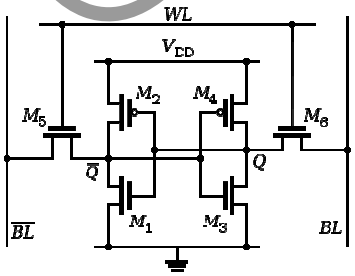
RAM (Random Access Memory)

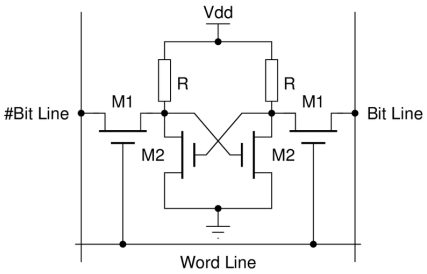
- SRAM (Static RAM): RAM tĩnh
- DRAM (Dynamic RAM): RAM động
- NVRAM
- Ferroelectric RAM (FRAM)

89

Static RAM (SRAM) – RAM tĩnh

- Được chế tạo trên cơ sở FF, theo công nghệ MOS
- Mỗi bit nhớ có thể bao gồm 6 hoặc 4 transistors (hình vẽ)



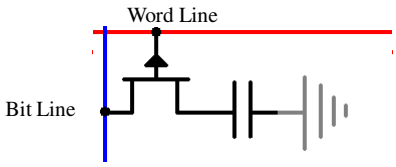


- 2 tín hiệu **Word Line** & **Bit Line** điều khiển truy nhập bit nhớ
- Việc đọc không làm mất nội dung thông tin SRAM
- SRAM là bộ nhớ truy nhập nhanh vì không cần thực hiện chu trình làm tươi (REFRESH) như DRAM

90

Dynamic RAM (DRAM) – RAM động

- DRAM: mỗi bit nhớ gồm một transistor và một tụ điện.



- Việc ghi nhớ dữ liệu dựa vào việc duy trì điện tích nạp vào tụ điện → việc đọc một bit nhớ làm nội dung bit này bị hủy → sau mỗi lần đọc một ô nhớ, bộ phận điều khiển bộ nhớ phải viết lại nội dung ô nhớ đó.
- Việc lưu giữ thông tin trong bit nhớ chỉ là tạm thời vì tụ điện sẽ phóng hết điện tích đã nạp → và như vậy phải làm tươi bộ nhớ.
- Việc làm tươi (Refresh) được thực hiện với tất cả các ô nhớ trong bộ nhớ, được thực hiện tự động bởi một vi mạch bộ nhớ.

91

SRAM versus DRAM

- **SRAM**
  - Thời gian truy cập nhanh hơn, do ko phải làm tươi
  - Mật độ tích hợp không cao
  - Giá thành đắt
  - Thường được dùng cho các bộ nhớ dung lượng bé, thời gian Đọc/Ghi nhanh.
  - Ví dụ: các bộ nhớ tạm thời Cache, dùng trong các hệ vi xử lý (microprocessor) nhỏ, làm bộ nhớ cho FPGA
- **DRAM**
  - Thời gian truy cập chậm hơn, do ko phải làm tươi
  - Mật độ tích hợp cao
  - Giá thành rẻ hơn SRAM
  - Rất thích hợp sử dụng cho các bộ nhớ dung lượng lớn và cực lớn
  - Ví dụ: bộ nhớ chính trong máy tính và các hệ vi xử lý

92

Các loại DRAM

- SDRAM (**S**ynchronous **D**ynamic **RAM**) được gọi là DRAM đồng bộ,
- SDRAM gồm các loại: SDR, DDR, DDR2, DDR3
  - SDR SDRAM = Single Data Rate SDRAM
  - DDR SDRAM = Double Data Rate SDRAM
  - DDR2 SDRAM = Double Data Rate 2 SDRAM
  - DDR3 SDRAM = Double Data Rate type 3 SDRAM
- Lưu ý **tránh nhầm lẫn** giữa
  - **SDRAM (RAM động đồng bộ), và**
  - **SRAM (Static RAM = RAM tĩnh)**

93

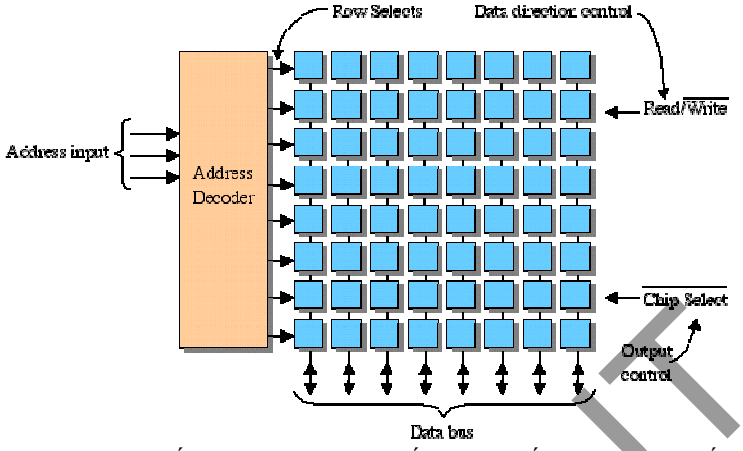
NVRAM & FRAM

- These devices are for your own study!

94

Kiến trúc ma trận của bộ nhớ

- Xét kiến trúc RAM 8 x 8 (dung lượng 64 bits: 8 words, mỗi word 8 bits)



- Các bit được bố trí thành ma trận, số hàng = số lượng word, số cột = độ rộng word, và mỗi hàng (row) tương ứng với 1 word
- Sử dụng mạch giải mã (Decoder) để giải mã địa chỉ và chọn word tương ứng để thực hiện hoạt động Đọc/Ghi

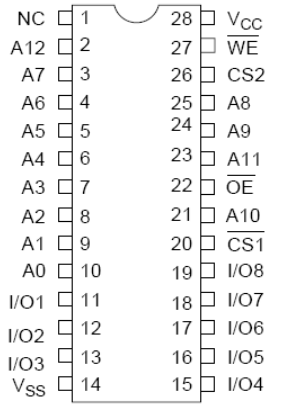
95

RAM 6264

- 64K SRAM (8-kword x 8-bit)
- Thời gian truy xuất nhanh (85 – 100 ns)
- Nguồn 5V, công suất thấp
- Tương thích họ TTL

Pin name	Function
A0 to A12	Address input
I/O1 to I/O8	Data input/output
$\overline{CS1}$	Chip select 1
CS2	Chip select 2
$\overline{WE}$	Write enable
$\overline{OE}$	Output enable
NC	No connection
V_{cc}	Power supply
V_{ss}	Ground

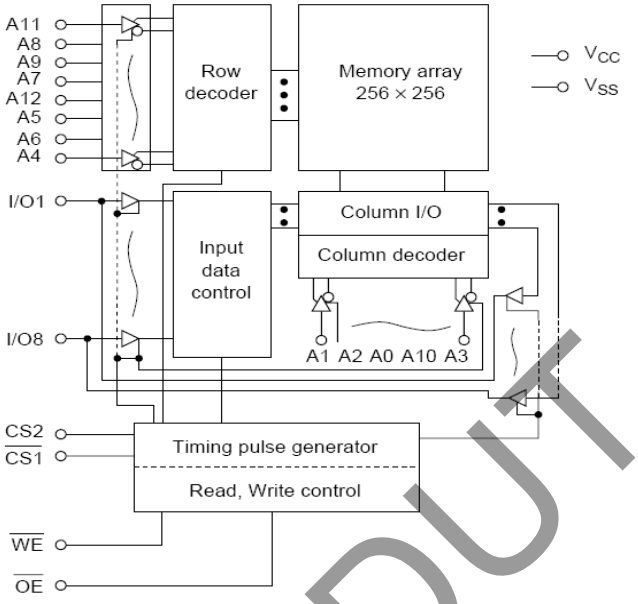
HM6264BLP/BLSP/BLFP Series



(Top view)

96

RAM 6264 (tt)



97

RAM 6264 (tt)

Function Table

$\overline{WE}$	$\overline{CS1}$	CS2	$\overline{OE}$	Mode	V_{CC} current	I/O pin	Ref. cycle
×	H	×	×	Not selected (power down)	I_{SB}, I_{SB1}	High-Z	—
×	×	L	×	Not selected (power down)	I_{SB}, I_{SB1}	High-Z	—
H	L	H	H	Output disable	I_{CC}	High-Z	—
H	L	H	L	Read	I_{CC}	Dout	Read cycle (1)–(3)
L	L	H	H	Write	I_{CC}	Din	Write cycle (1)
L	L	H	L	Write	I_{CC}	Din	Write cycle (2)

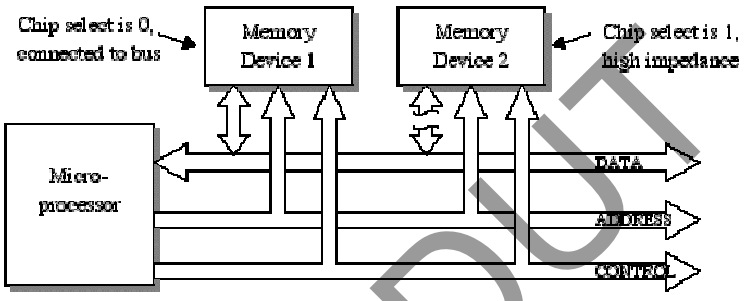
Note: ×: H or L

- CS1 và CS2: các chân tín hiệu chọn chip (Chip Select)
- WE, CS1, OE: tích cực mức thấp (L)
- CS2: tích cực mức cao (H)

98

Tổ chức bộ nhớ cho hệ Vi Xử Lý

- Ghép nối các chíp nhớ vào một hệ Vi Xử Lý (VXL)
- Sử dụng 3 loại BUS:
  - DATA BUS (BUS dữ liệu)
  - ADDRESS BUS (BUS địa chỉ)
  - CONTROL BUS (BUS điều khiển)
- Và các cổng đệm 3 trạng thái (3-state buffer)



99

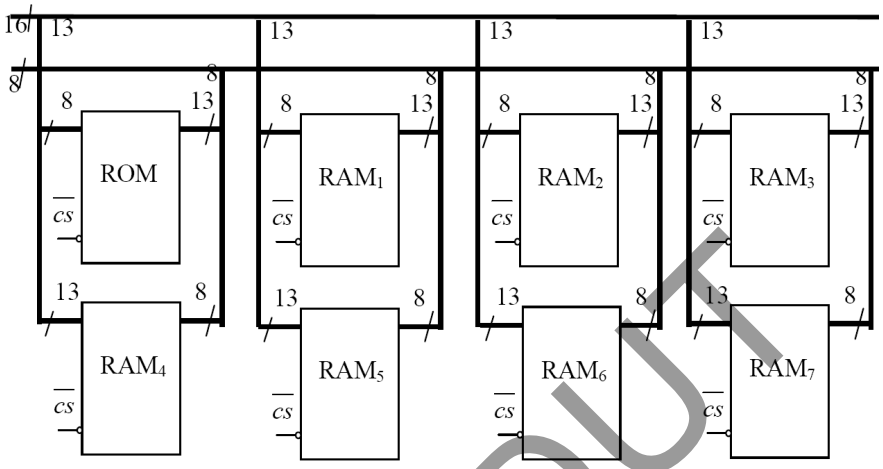
Tổ chức bộ nhớ cho hệ Vi Xử Lý (tt)

- **Bài toán**:
  - Ghép các vi mạch nhớ dung lượng nhỏ để tạo thành hệ thống nhớ với kích thước lớn hơn.
  - Tổ chức bộ nhớ cho 1 hệ vi xử lý cho trước.
- **Ví dụ**: Giả sử CPU có 16 đường địa chỉ và bus dữ liệu 8 bit, dung lượng bộ nhớ tối đa mà CPU có thể quản lý được là
  - 2^{16} = 64 KB → Độ rộng BUS địa chỉ quyết định dung lượng bộ nhớ tối đa mà CPU có thể quản lý được
- Phân chia dung lượng này thành các khối nhớ sau đây
  - 1 ROM dung lượng 8 KB, sử dụng 1 chíp EPROM 2764 (8Kx8)
  - 7 RAM với tổng dung lượng 56 KB, sử dụng 7 chíp SRAM 6264 (8Kx8)
- ***Câu hỏi: Vẽ sơ đồ tổ chức bộ nhớ của hệ thống trên?***

100

Tổ chức bộ nhớ cho hệ Vi Xử Lý (tt)

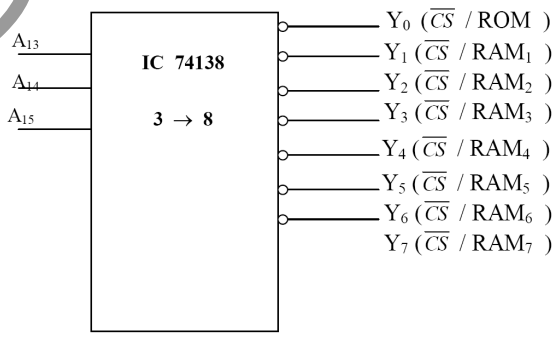
- Sơ đồ tổ chức bộ nhớ



101

Tổ chức bộ nhớ cho hệ Vi Xử Lý (tt)

- Sử dụng vi mạch giải mã 3→8 là 74138 để chọn lần lượt từng chíp nhớ tương ứng



- Tại mỗi thời điểm, duy nhất 1 chân tín hiệu chọn chip tương ứng CS = 0 → chip nhớ tương ứng được lựa chọn sử dụng, các chip nhớ khác không được chọn (do các tín hiệu đầu ra khác của vi mạch 74138 có mức logic 1)

102

Tổ chức bộ nhớ cho hệ Vi Xử Lý (tt)

- Bản đồ bộ nhớ của hệ thống được thiết kế

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Địa chỉ Hex				
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	H
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	F	F	F	H
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	H
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3	F	F	F	H
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	H
0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	5	F	F	F	H
0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	H
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	7	F	F	F	H
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	H
1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	9	F	F	F	H
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	A	0	0	0	H
1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	B	F	F	F	H
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	C	0	0	0	H
1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	D	F	F	F	H
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	E	0	0	0	H
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	F	F	F	F	H

ROM
RAM_1
RAM_2
RAM_3
RAM_4
RAM_5
RAM_6
RAM_7

103

Bài tập về thiết kế bộ nhớ

- Một hệ thống vi xử lý (µP - VXL) có độ rộng bus địa chỉ 16 bit, độ rộng bus dữ liệu 8 bit. Thiết kế hệ thống bộ nhớ (memory) cho hệ VXL đã cho sử dụng các chip nhớ sau đây:
  - ROM1: 8K x 8 dùng vi mạch 27C64
  - ROM2: 4K x 8 dùng vi mạch 27C32
  - ROM3: 4K x 8 dùng vi mạch 27C32
  - RAM1: 8K x 8 dùng vi mạch 6264
  - RAM2: 8K x 8 dùng vi mạch 6264
  - RAM3: 8K x 8 dùng vi mạch 6264
  - RAM4: 8K x 8 dùng vi mạch 6264
  - RAM5: 16K x 8 dùng vi mạch 62128

104

Hint

- Độ rộng bus địa chỉ của VXL là: 16 (A0 – A15)
- ROM1: 8K x 8 dùng vi mạch 27C64 →13 (A0 – A12)
- ROM2: 4K x 8 dùng vi mạch 27C32 → 12 (A0 – A11)
- ROM3: 4K x 8 dùng vi mạch 27C32 → 12
- RAM1: 8K x 8 dùng vi mạch 6264 →13
- RAM2: 8K x 8 dùng vi mạch 6264 →13
- RAM3: 8K x 8 dùng vi mạch 6264 →13
- RAM4: 8K x 8 dùng vi mạch 6264 →13
- RAM5: 16K x 8 dùng vi mạch 62128 →14 (A0 – A13)

- **Tín hiệu chọn chip /CS được nối như thế nào?**

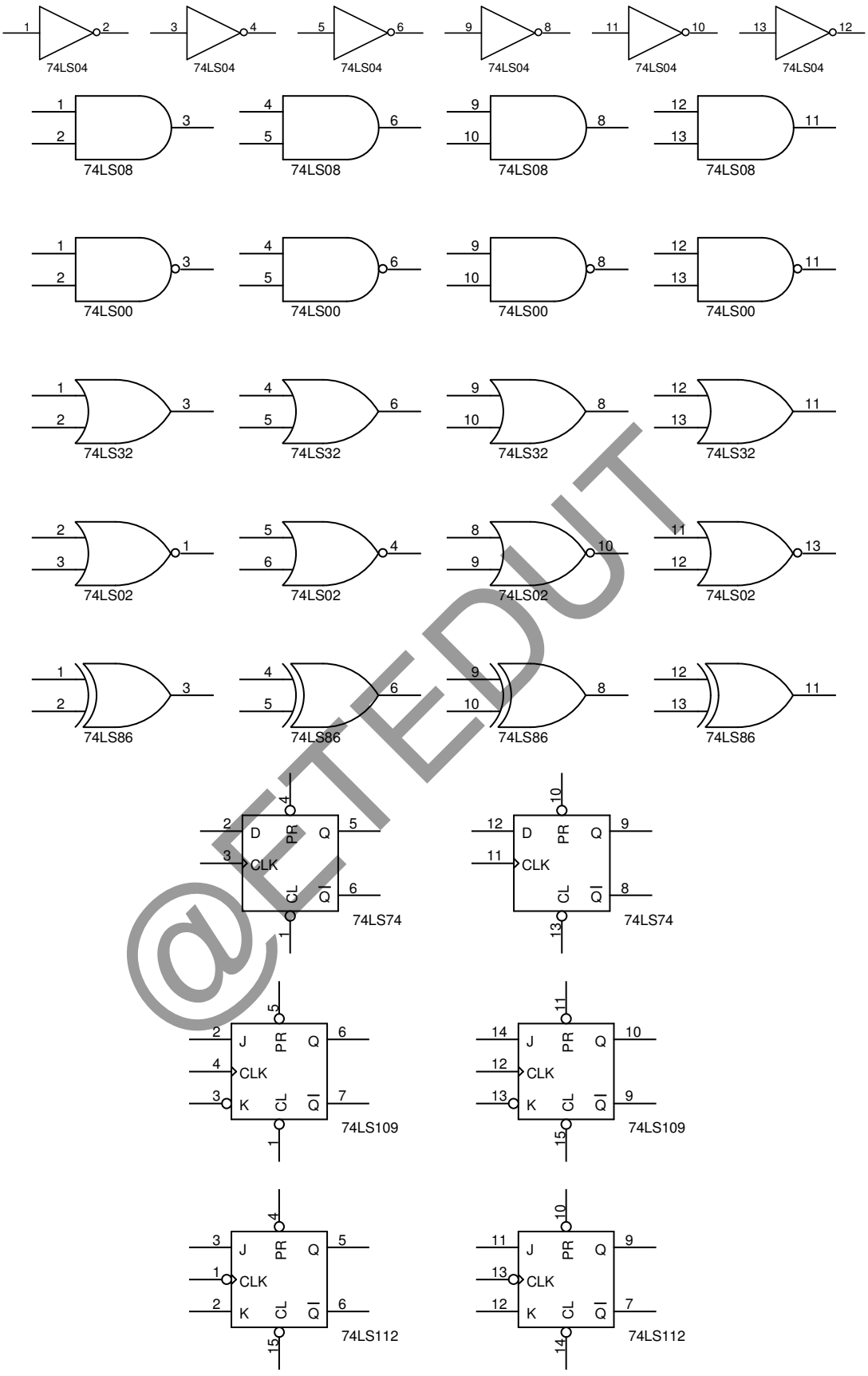
105

Tổng kết chương 5

- Khái niệm và đặc điểm của hệ tuần tự
- Đếm nối tiếp
- FSM
- Thanh ghi dịch
- Bộ nhớ bán dẫn

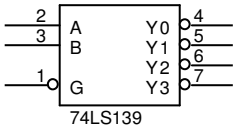
106

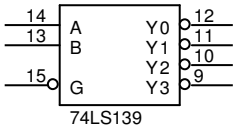
Phụ lục A: Các vi mạch cổng và FF thông dụng

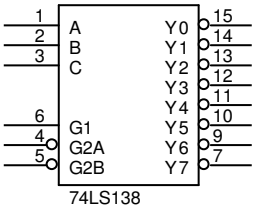


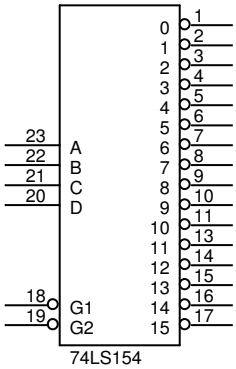
Phụ lục B: Các vi mạch tổ hợp thông dụng

Mạch giải mã (decoder) 2→4, 3→8, 4→16

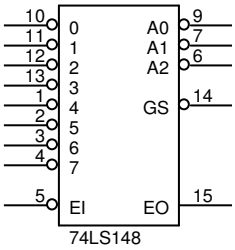


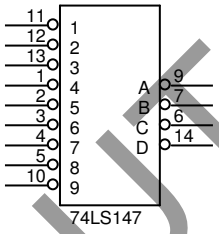




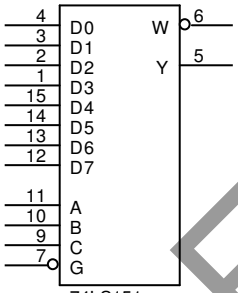


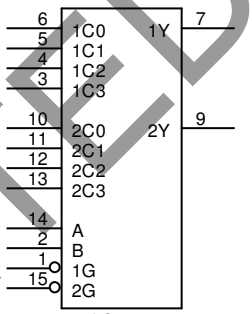
Mạch mã hóa (encoder) có ưu tiên 8→3, 10→4

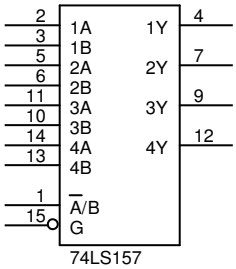




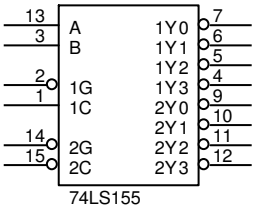
Mạch chọn kênh (mux) 8→1, 4→1, 2→1



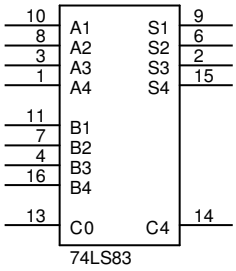




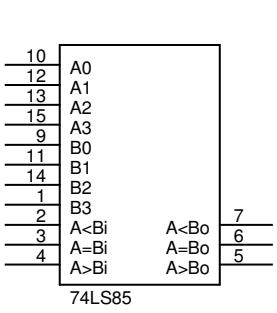
Mạch phân kênh (demux) 1→4

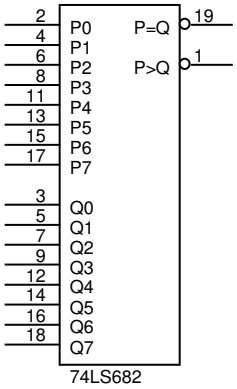


Mạch cộng nhị phân 4 bit

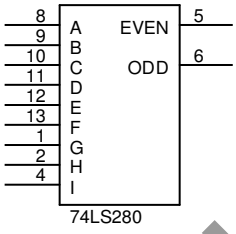


Mạch so sánh 4 bit, 8 bit

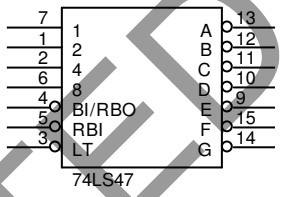




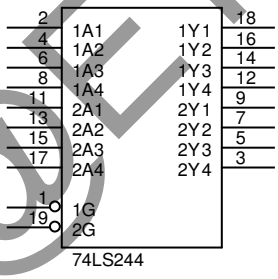
Mạch tạo/kiểm tra parity

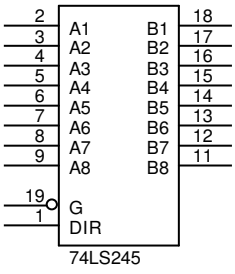


Mạch chuyển mã BCD →mã LED 7 đoạn anode chung



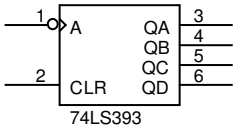
Mạch đệm 8 bit

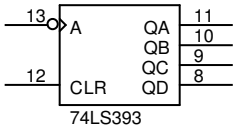


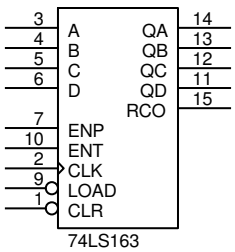


Phụ lục C: Các vi mạch tuần tự thông dụng

Mạch đếm nhị phân 4 bit đồng bộ



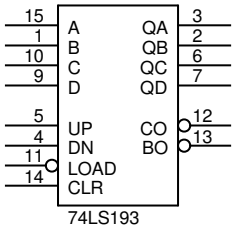




Các ngõ vào					Các ngõ ra				Chức năng
$\overline{CLR}$	$\overline{LOAD}$	ENP	ENT	CLK	Q_A	Q_B	Q_C	Q_D	
L	x	x	x	↑	L	L	L	L	Reset veà 0
H	L	x	x	↑	D	C	B	A	Nhaäp döõ lieäu vaøo
H	H	x	L	↑	Khoâng thay ñoåi				Khoâng ñeám
H	H	L	x	↑	Khoâng thay ñoåi				Khoâng ñeám
H	H	H	H	↑	Ñeám leân				Ñeám
x	x	x	x	↓	Khoâng thay ñoåi				Khoâng ñeám

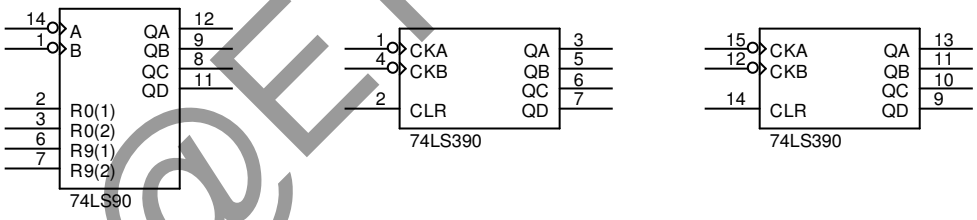
RCO (Ripple Carry Out) = ENT.Q_A.Q_B.Q_C.Q_D

Mạch đếm lên/xuống đồng bộ nhị phân 4 bit

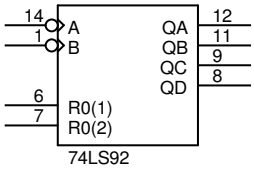


UP	DN	$\overline{LOAD}$	CLR	Chöùc naêng
↑	H	H	L	Ñeám leân
↓	H	H	L	Khoâng ñeám
H	↑	H	L	Ñeám xuoáng
H	↓	H	L	Khoâng ñeám
x	x	L	L	Nhaäp döõ lieäu vaøo
x	x	x	H	Reset veà 0

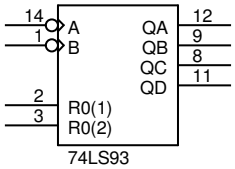
Mạch đếm mod 10 (mod 2 và mod 5)



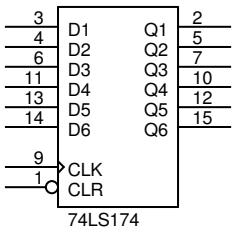
Mạch đếm mod 12 (mod 2 và mod 6)



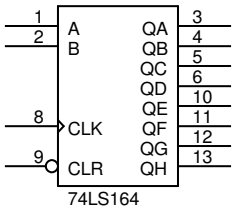
Mạch đếm mod 16 (mod 2 và mod 8)



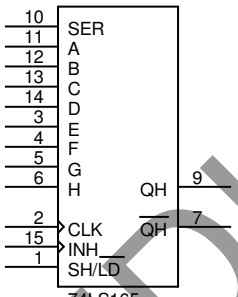
Thanh ghi dịch PIPO



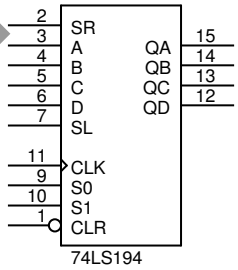
Thanh ghi dịch SIPO



Thanh ghi dịch PISO



Thanh ghi dịch trái/ phải PIPO



Mạch chốt 8 bit