CHƯƠNG 1: CÁC HỆ THỐNG SỐ & MÃ

Nhu cầu về định lượng trong quan hệ giữa con người với nhau, nhất là trong những trao đổi thương mại, đã có từ khi xã hội hình thành. Đã có rất nhiều cố gắng trong việc tìm kiếm các vật dụng, các ký hiệu . . . dùng cho việc định lượng này như các que gỗ, vỏ sò, số La mã . . . Hiện nay số Ả rập tỏ ra có nhiều ưu điểm khi được sử dụng trong định lượng, tính toán. . . ..

Việc sử dụng hệ thống số hằng ngày trở nên quá quen thuộc khiến chúng ta có thể đã quên đi sự hình thành và các qui tắc để viết các con số.

Chương này nhắc lại một cách sơ lược nguyên lý của việc viết số và giới thiệu các hệ thống số khác ngoài hệ thống thập phân quen thuộc, phương pháp biến đổi qua lại của các số trong các hệ thống khác nhau. Chúng ta sẽ đặc biệt quan tâm đến hệ thống nhị phân là hệ thống được dùng trong lãnh vực điện tử-tin học như là một phương tiện để giải quyết các vấn đề mang tính logic.

Phần cuối của chương sẽ giới thiệu các loại mã thông dụng để chuẩn bị cho các chương kế tiếp.

1.1 Nguyên lý của việc viết số

Một số được viết bằng cách đặt kề nhau các **ký hiệu**, được chọn trong một tập hợp xác định. Mỗi ký hiệu trong một số được gọi là **số mã** (số hạng, digit).

Thí dụ, trong hệ thống thập phân (cơ số 10) tập hợp này gồm 10 ký hiệu rất quen thuộc, đó là các con số từ 0 đến 9:

$$S_{10} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

Khi một số gồm nhiều số mã được viết, giá trị của các số mã tùy thuộc vị trí của nó trong số đó. Giá trị này được gọi là **trọng số** của số mã.

Thí dụ số 1998 trong hệ thập phân có giá trị xác định bởi triển khai theo đa thức của 10:

$1998_{10} = 1x10^3 + 9x10^2 + 9x10^1 + 9x10^0 = 1000 + 900 + 90 + 8$

Trong triển khai, số mũ của đa thức chỉ vị trí của một ký hiệu trong một số với qui ước vị trí của hàng đơn vị là 0, các vị trí liên tiếp về phía trái là 1, 2, 3, ... Nếu có phần lẻ, vị trí đầu tiên sau dấu phẩy là -1, các vị trí liên tiếp về phía phải là -2, -3, ...

Ta thấy, số 9 đầu tiên (sau số 1) có trọng số là 900 trong khi số 9 thứ hai chỉ là 90.

Có thể nhận xét là với 2 ký hiệu giống nhau trong hệ 10, ký hiệu đứng trước có trọng số gấp 10 lần ký hiệu đứng ngay sau nó. Điều này hoàn toàn đúng cho các hệ khác, thí dụ, đối với hệ nhị phân (cơ số 2) thì tỉ lệ này là 2.

Tổng quát, một hệ thống số được gọi là **hệ b** sẽ gồm b ký hiệu trong một tập hợp:

$$S_b = \{S_0, S_1, S_2, . \ . \ ., S_{b-1}\}$$

Một số N được viết:

$N = (a_n a_{n-1} a_{n-2} . \ . \ . a_i \ . \ . \ . a_0 \ , a_{-1} a_{-2} \ . \ . \ . a_{-m})_b$ với $a_i \in S_b$

Sẽ có giá trị:

$N = a_n \ b^n + a_{n-1} b^{n-1} + a_{n-2} b^{n-2} + \ . \ . \ . + a_i b^i + . \ . \ . + a_0 b^0 + a_{-1} \ b^{-1} + a_{-2} \ b^{-2} + . \ . \ . + a_{-m} b^{-m}.$

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

$a_i b^i$ chính là trọng số của một ký hiệu trong S_b ở vị trí thứ i.

1.2 Các hệ thống số

1.2.1 Hệ cơ số 10 (thập phân, Decimal system)

Hệ thập phân là hệ thống số rất quen thuộc, gồm 10 số mã như nói trên.

Dưới đây là vài ví dụ số thập phân:

$N = 1998_{10} = 1x10^3 + 9x10^2 + 9x10^1 + 8x10^0 = 1x1000 + 9x100 + 9x10 + 8x1$

$N = 3{,}14_{10} = 3x10^0 + 1x10^{-1} + 4x10^{-2} = 3x1 + 1x1/10 + 4x1/100$

1.2.2 Hệ cơ số 2 (nhị phân, Binary system)

Hệ nhị phân gồm hai số mã trong tập hợp

$$S_2 = \{0, 1\}$$

Mỗi số mã trong một số nhị phân được gọi là một **bit** (viết tắt của binary digit).

Số N trong hệ nhị phân:

$N = (a_n a_{n-1} a_{n-2} . \ . \ . a_i \ . \ . \ . a_0 \ , a_{-1} a_{-2} \ . \ . \ . a_{-m})_2$ (với $a_i \in S_2$)

Có giá trị là:

$N = a_n \ 2^n + a_{n-1} 2^{n-1} + \ . \ . \ . + a_i 2^i + . \ . \ . + a_0 2^0 + a_{-1} \ 2^{-1} + a_{-2} \ 2^{-2} + \ . \ . \ . + a_{-m} 2^{-m}$

a_n là bit có **trọng số lớn nhất**, được gọi là bit **MSB** (Most significant bit) và a_{-m} là bit có **trọng số nhỏ nhất**, gọi là bit **LSB** (Least significant bit).

Thí dụ: $N = 1010{,}1_2 = 1x2^3 + 0x2^2 + 1x2^1 + 0x2^0 + 1x2^{-1} = 10{,}5_{10}$

1.2.3 Hệ cơ số 8 (bát phân ,Octal system)

Hệ bát phân gồm tám số trong tập hợp

$$S_8 = \{0,1, 2, 3, 4, 5, 6, 7\}.$$

Số N trong hệ bát phân:

$N = (a_n a_{n-1} a_{n-2} . \ . \ . a_i \ . \ . \ . a_0 \ , a_{-1} a_{-2} \ . \ . \ . a_{-m})_8$ (với $a_i \in S_8$)

Có giá trị là:

$N = a_n\ 8^n + a_{n-1}8^{n-1} + a_{n-2}8^{n-2} + . . . + a_i8^i . . . + a_08^0 + a_{-1}\ 8^{-1} + a_{-2}\ 8^{-2} + . . . + a_{-m}8^{-m}$

Thí dụ: $N = 1307,1_8 = 1x8^3 + 3x8^2 + 0x8^1 + 7x8^0 + 1x8^{-1} = 711,125_{10}$

1.2.4 Hệ cơ số 16 (thập lục phân, Hexadecimal system)

Hệ thập lục phân được dùng rất thuận tiện để con người giao tiếp với máy tính, hệ này gồm mười sáu số trong tập hợp

$S_{16} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$

(A tương đương với 10_{10} , B $=11_{10}$, ... , F$=15_{10}$) .

Số N trong hệ thập lục phân:

$N = (a_na_{n-1}a_{n-2}. . . a_i . . . a_0\ ,\ a_{-1}a_{-2} . . . a_{-m})_{16}$ (với $a_i \in S_{16}$)

Có giá trị là:

$N = a_n\ 16^n + a_{n-1}16^{n-1} + a_{n-2}16^{n-2} + . . + a_i16^i . . . + a_016^0 + a_{-1}\ 16^{-1} + a_{-2}\ 16^{-2} + . . . + a_{-m}16^{-m}$

Người ta thường dùng chữ H (hay h) sau con số để chỉ số thập lục phân.

Thí dụ: $N = 20EA,8H = 20EA,8_{16} = 2x16^3 + 0x16^2 + 14x16^1 + 10x16^0 + 8x16^{-1}$
$= 4330,5_{10}$

1.3 Biến đổi qua lại giữa các hệ thống số

Khi đã có nhiều hệ thông số, việc xác định giá trị tương đương của một số trong hệ này so với hệ kia là cần thiết. Phần sau đây cho phép ta biến đổi qua lại giữa các số trong bất cứ hệ nào sang bất cứ hệ khác trong các hệ đã được giới thiệu.

1.3.1 Đổi một số từ hệ b sang hệ 10

Để đổi một số từ hệ b sang hệ 10 ta triển khai trực tiếp đa thức của b

Một số N trong hệ b:

$N = (a_na_{n-1}a_{n-2}. . . a_i . . . a_0\ ,\ a_{-1}a_{-2} . . . a_{-m})_b$ với $a_i \in S_b$

Có giá trị tương đương trong hệ 10 là:

$N = a_n\ b^n + a_{n-1}b^{n-1} + . . . + a_ib^i + . . . + a_0b^0 + a_{-1}\ b^{-1} + a_{-2}\ b^{-2} + . . . + a_{-m}b^{-m}.$

Thí dụ:

* Đổi số $10110,11_2$ sang hệ 10

$10110,11_2 = 1x2^4 + 0 + 1x2^2 + 1x2 + 0 + 1x2^{-1} + 1x2^{-2} = 22,75_{10}$

* Đổi số 4BE,ADH sang hệ 10

$4BE,ADH = 4x16^2 + 11x16^1 + 14x16^0 + 10x16^{-1} + 13x16^{-2} = 1214,675_{10}$

1.3.2 Đổi một số từ hệ 10 sang hệ b

Đây là bài toán tìm một dãy ký hiệu cho số N viết trong hệ b.

Tổng quát, một số N cho ở hệ 10, viết sang hệ b có dạng:

$N = (a_na_{n-1} . . . a_0\ ,\ a_{-1}a_{-2} . . . a_{-m})_b = (a_na_{n-1} . . . a_0)_b + (0,a_{-1}a_{-2} . . . a_{-m})_b$

Trong đó

$(a_na_{n-1} . . . a_0)_b$ = PE(N) là phần nguyên của N

và $(0,a_{-1}a_{-2} . . . a_{-m})_b$ = PF(N) là phần lẻ của N

Phần nguyên và phần lẻ được biến đổi theo hai cách khác nhau:

◆ **Phần nguyên**:

Giá trị của phần nguyên xác định nhờ triển khai:

$$PE(N) = a_n b^n + a_{n-1}b^{n-1} + \ldots + a_1 b^1 + a_0 b^0$$

Hay có thể viết lại

$$PE(N) = (a_n b^{n-1} + a_{n-1}b^{n-2} + \ldots + a_1)b + a_0$$

Với cách viết này ta thấy nếu chia PE(N) cho b, ta được **thương số** là $\mathbf{PE'(N) = (a_n b^{n-1} + a_{n-1}b^{n-2} + \ldots + a_1)}$ và **số dư** là $\mathbf{a_0}$.

Vậy **số dư của lần chia thứ nhất** này chính là **số mã có trọng số nhỏ nhất** ($\mathbf{a_0}$) của phần nguyên.

Lặp lại bài toán chia PE'(N) cho b:

$$PE'(N) = a_n b^{n-1} + a_{n-1}b^{n-2} + \ldots + a_1 = (a_n b^{n-2} + a_{n-1}b^{n-3} + \ldots + a_2)b + a_1$$

Ta được **số dư thứ hai**, chính là số mã có trọng số lớn hơn kế tiếp ($\mathbf{a_1}$) và thương số là $PE''(N) = a_n b^{n-2} + a_{n-1}b^{n-3} + \ldots + a_2$.

Tiếp tục bài toán chia thương số có được với b, cho đến khi được **số dư của phép chia cuối cùng**, đó chính là số mã có trọng số lớn nhất ($\mathbf{a_n}$)

◆ **Phần lẻ**:

Giá trị của phần lẻ xác định bởi:

$$PF(N) = a_{-1} b^{-1} + a_{-2} b^{-2} + \ldots + a_{-m}b^{-m}$$

Hay viết lại

$$PF(N) = b^{-1} (a_{-1} + a_{-2} b^{-1} + \ldots + a_{-m}b^{-m+1})$$

Nhân PF(N) với b, ta được : $bPF(N) = a_{-1} + (a_{-2} b^{-1} + \ldots + a_{-m}b^{-m+1}) = a_{-1} + PF'(N)$.

Vậy lần nhân thứ nhất này ta được **phần nguyên của phép nhân**, chính là **số mã có trọng số lớn nhất của phần lẻ** ($\mathbf{a_{-1}}$) (số a_{-1} này có thể vẫn là số 0).

PF'(N) là phần lẻ xuất hiện trong phép nhân.

Tiếp tục nhân PF'(N) với b, ta tìm được $\mathbf{a_{-2}}$ và **phần lẻ PF"(N)**.

Lặp lại bài toán nhân phần lẻ với b cho đến khi kết quả có phần lẻ bằng không, ta sẽ tìm được dãy số ($a_{-1}a_{-2} \ldots a_{-m}$).

Chú ý: *Phần lẻ của số N khi đổi sang hệ b có thể gồm vô số số hạng (do kết quả của phép nhân luôn khác 0), điều này có nghĩa là ta không tìm được một số trong hệ b có **giá trị đúng bằng** phần lẻ của số thập phân, vậy tùy theo yêu cầu về độ chính xác khi chuyển đổi mà người ta lấy một số số hạng nhất định.*

Thí dụ:

* Đổi $25,3_{10}$ sang hệ nhị phân

Phần nguyên:
- 25 : 2 = 12 dư 1 ⇒ $a_0 = 1$
- 12 : 2 = 6 dư 0 ⇒ $a_1 = 0$
- 6 : 2 = 3 dư 0 ⇒ $a_2 = 0$
- 3 : 2 = 1 dư 1 ⇒ $a_3 = 1$

thương số cuối cùng là 1 cũng chính là bit a_4:

⇒ $a_4 = 1$

Vậy PE(N) = 11001

Phần lẻ:
- 0,3 * 2 = 0,6 ⇒ $a_{-1} = 0$
- 0,6 * 2 = 1,2 ⇒ $a_{-2} = 1$
- 0,2 * 2 = 0,4 ⇒ $a_{-3} = 0$
- 0,4 * 2 = 0,8 ⇒ $a_{-4} = 0$
- 0,8 * 2 = 1,6 ⇒ $a_{-5} = 1$ . . .

Nhận thấy kết quả của các bài toán nhân luôn khác không, do phần lẻ của lần nhân cuối cùng là 0,6, đã lặp lại kết quả của lần nhân thứ nhất, như vậy bài toán không thể kết thúc với kết quả đúng bằng 0,3 của hệ 10.

Giả sử bài toán yêu cầu lấy 5 số lẻ thì ta có thể dừng ở đây và

PF(N) = 0,01001.

Kết quả cuối cùng là:

$$25,3_{10} = 11001,01001_2$$

* Đổi $1376,85_{10}$ sang hệ thập lục phân

Phần nguyên: 1376 : 16 = 86 số dư = 0 $\Rightarrow a_0 = 0$

86 : 16 = 5 số dư = 6 $\Rightarrow a_1 = 6$ & $\Rightarrow a_2 = 5$

$1376_{10} = 560H$

Phần lẻ: 0,85 * 16 = 13,6 $\Rightarrow a_{-1} = 13_{10}$=DH

0,6 * 16 = 9,6 $\Rightarrow a_{-2} = 9$

0,6 * 16 = 9,6 $\Rightarrow a_{-3} = 9$

Nếu chỉ cần lấy 3 số lẻ: $0,85_{10}$= 0,D99H

Và kết quả cuối cùng:

$$1376,85_{10} = 560,D99H$$

1.3.3 Đổi một số từ hệ b sang hệ b^k và ngược lại

Từ cách triển khai đa thức của số N trong hệ b, ta có thể nhóm thành từng k số hạng từ dấu phẩy về hai phía và đặt thành thừa số chung

$N = a_n b^n + \ldots + a_5 b^5 + a_4 b^4 + a_3 b^3 + a_2 b^2 + a_1 b^1 + a_0 b^0 + a_{-1} b^{-1} + a_{-2} b^{-2} + a_{-3} b^{-3} \ldots + a_{-m} b^{-m}$

Để dễ hiểu, chúng ta lấy thí dụ k = 3, N được viết lại bằng cách nhóm từng 3 số hạng, kể từ dấu phẩy về 2 phía

$N = \ldots + (a_5 b^2 + a_4 b^1 + a_3 b^0) b^3 + (a_2 b^2 + a_1 b^1 + a_0 b^0) b^0 + (a_{-1} b^2 + a_{-2} b^1 + a_{-3} b^0) b^{-3} + \ldots$

Phần chứa trong mỗi dấu ngoặc luôn luôn nhỏ hơn b^3 , vậy số này tạo nên một số trong hệ b^3 và lúc đó được biểu diễn bởi ký hiệu tương ứng trong hệ này.

Thật vậy, số N có dạng:

$N = \ldots + A_2 B^2 + A_1 B^1 + A_0 B^0 + A_{-1} B^{-1} + \ldots$

Trong đó:

$B = b^3$ ($B^0 = b^0$; $B^1 = b^3$; $B^2 = b^6$, $B^{-1} = b^{-3}$ ....)

$A_2 = a_8 b^2 + a_7 b^1 + a_6 b^0 = b^3(a_8 b^{-1} + a_7 b^{-2} + a_6 b^{-3}) < B = b^3$

$A_1 = a_5 b^2 + a_4 b^1 + a_3 b^0 = b^3(a_5 b^{-1} + a_4 b^{-2} + a_3 b^{-3}) < B = b^3$

$A_0 = a_2 b^2 + a_1 b^1 + a_0 b^0 = b^3(a_2 b^{-1} + a_1 b^{-2} + a_0 b^{-3}) < B = b^3$

Các số A_i luôn luôn nhỏ hơn $B = b^3$ như vậy nó chính là một phần tử của tập hợp số tạo nên hệ $B = b^3$

Ta có kết quả biến đổi tương tự cho các hệ số k khác.

Tóm lại, để đổi một số từ hệ b sang hệ b^k, từ dấu phẩy đi về hai phía, ta nhóm từng k số hạng, giá trị của mỗi k số hạng này (tính theo hệ b) chính là số trong hệ b^k .

Thí dụ:

* Đổi số $N = 10111110101\ ,\ 01101_2$ sang hệ $8 = 2^3$

Từ dấu phẩy, nhóm từng 3 số hạng về hai phía (nếu cần, thêm số 0 vào ở nhóm đầu và cuối để đủ 3 số hạng mà không làm thay đổi giá trị của số N):

$$N = 010\ 111\ 110\ 101\ ,\ 011\ \ 010_2$$

Ghi giá trị tương ứng của các số 3 bit, ta được số N trong hệ 8

$$N = 2\quad 7\quad 6\quad 5\ ,\ 3\quad 2_{\ 8}$$

* Đổi số N trên sang hệ $16 = 2^4$

Cũng như trên nhưng nhóm từng 4 số hạng

$N = 0101\ \ 1111\ \ 0101\ ,\ 0110\ \ 1000_2$

$N = \ \ 5 \ \ \ F \ \ \ 5 \ \ ,\ \ 6 \ \ \ 8_{16}$

Từ kết quả của phép đổi số từ hệ b sang hệ b^k, ta có thể suy ra cách biến đổi ngược một cách dễ dàng: Thay mỗi số hạng của số trong hệ b^k bằng một số gồm k số hạng trong hệ b.

Thí dụ để đổi số $N = 5\ F5,\ 68_{16}$ (hệ 2^4) sang hệ nhị phân (2) ta dùng 4 bit để viết cho mỗi số hạng của số này:

$N = 0101\ \ 1111\ \ 0101\ ,\ 0110\ \ 1000_2$

1.3.4 Đổi một số từ hệ b^k sang hệ b^p

Qua trung gian của hệ b, ta có thể đổi từ hệ b^k sang hệ b^p. Muốn đổi số N từ hệ b^k sang hệ b^p, trước nhất đổi số N sang hệ b rồi từ hệ b tiếp tục đổi sang hệ b^p.

Thí dụ:

- Đổi số $1234,67_8$ sang hệ 16

$1234,67_8 = 001\ 010\ 011\ 100,110\ 111_2 = 0010\ 1001\ 1100,1101\ 1100_2 = 29C,DCH$

- Đổi số ABCD,EFH sang hệ 8

$ABCD,EFH = 1010\ 1011\ 1100\ 1101,1110\ 1111_2 = 1\ 010\ 101\ 111\ 001\ 101,111\ 011\ 110_2 = 125715,736_8$

Dưới đây là bảng kê các số đầu tiên trong các hệ khác nhau:

Thập phân	Nhị phân	Bát phân	Thập lục phân	Thập phân	Nhị phân	Bát phân	Thập lục phân
0	0	0	0	13	1101	15	D
1	1	1	1	14	1110	16	E
2	10	2	2	15	1111	17	F
3	11	3	3	16	10000	20	10
4	100	4	4	17	10001	21	11
5	101	5	5	18	10010	22	12
6	110	6	6	19	10011	23	12
7	111	7	7	20	10100	24	14
8	1000	10	8	21	10101	25	15
9	1001	11	9	22	10110	26	16
10	1010	12	A	23	10111	17	17
11	1011	13	B	24	11000	30	18
12	1100	14	C	25	11001	31	19

Bảng 1.1

1.4 Các phép tính trong hệ nhị phân

Các phép tính trong hệ nhị phân được thực hiện tương tự như trong hệ thập phân, tuy nhiên cũng có một số điểm cần lưu ý

1.4.1 Phép cộng

Là phép tính làm cơ sở cho các phép tính khác.
Khi thực hiện phép cộng cần lưu ý:
0 + 0 = 0 ;
0 + 1 = 1 ;
1 + 1 = 0 nhớ 1 (đem qua bít cao hơn).

Ngoài ra nếu cộng nhiều số nhị phân cùng một lúc ta nên nhớ :
- Nếu số bit 1 chẵn, kết quả là 0;
- Nếu số bit 1 lẻ kết quả là 1
- Và cứ 1 cặp số 1 cho 1 số nhớ (bỏ qua số 1 dư, thí dụ với 5 số 1 ta kể là 2 cặp)

Thí dụ: Tính 011 + 101 + 011 + 011

```
    1 1   ← số nhớ
  1 1 1   ← số nhớ
    0 1 1
+   1 0 1
    0 1 1
    0 1 1
  --------
  1 1 1 0
```

1.4.2 Phép trừ

Cần lưu ý:
0 - 0 = 0 ;
1 - 1 = 0 ;
1 - 0 = 1 ;
0 - 1 = 1 nhớ 1 cho bit cao hơn
Thí dụ: Tính 1011 - 0101

```
    1       ← số nhớ
    1 0 1 1
-   0 1 0 1
    ---------
    0 1 1 0
```

1.4.3 Phép nhân

Cần lưu ý:
0 x 0 = 0 ;
0 x 1 = 0 ;
1 x 1 = 1
Thí dụ: Tính 1101 x 101

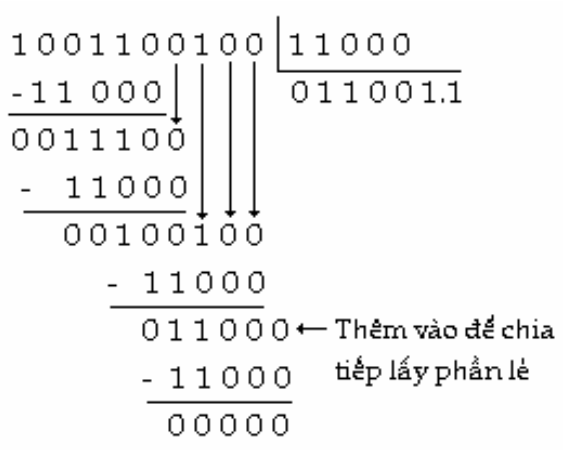
```
      1 1  0 1
x       1  0 1
      ---------
       1 1 0 1
     0 0 0 0
   1 1 0 1
  ---------------
```

1 0 0 0 0 0 1

1.4.4 Phép chia

Thí dụ: Chia 1001100100 cho 11000

Lần chia đầu tiên, 5 bit của số bị chia nhỏ hơn số chia nên ta được kết quả là 0, sau đó ta lấy 6 bit của số bị chia để chia tiếp (tương ứng với việc dịch phải số chia 1 bit trước khi thực hiện phép trừ)



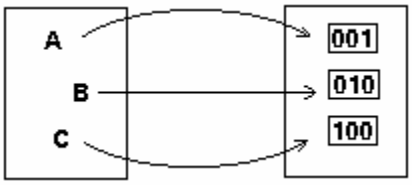
Kết quả : $(11001.1)_2 = (25.5)_{10}$

1.5 Mã hóa

1.5.1 Tổng quát

Mã hóa là gán một ký hiệu cho một đối tượng để thuận tiện cho việc thực hiện một yêu cầu cụ thể nào đó.

Một cách toán học, mã hóa là một phép áp một đối một từ một **tập hợp nguồn** vào một tập hợp khác gọi là **tập hợp đích**.



(H 1.1)

Tập hợp nguồn có thể là tập hợp các số, các ký tự, dấu, các lệnh dùng trong truyền dữ liệu . . . và tập hợp đích thường là tập hợp chứa các tổ hợp thứ tự của các số nhị phân.

Một tổ hợp các số nhị phân tương ứng với một số được gọi là **từ mã**. Tập hợp các từ mã được tạo ra theo một qui luật cho ta một **bộ mã**. Việc chọn một bộ mã tùy vào mục đích sử dụng.

Thí dụ để biểu diễn các chữ và số, người ta có **mã ASCII** (American Standard Code for Information Interchange), **mã Baudot, EBCDIC** . . .. Trong truyền dữ liệu ta có **mã dò lỗi, dò và sửa lỗi, mật mã** . . ..

Vấn đề ngược lại mã hóa gọi là **giải mã**.

Cách biểu diễn các số trong các hệ khác nhau cũng có thể được xem là một hình thức mã hóa, đó là các mã thập phân, nhị phân, thập lục phân . . . và việc chuyển từ mã này sang mã khác cũng thuộc loại bài toán mã hóa.

Trong kỹ thuật số ta thường dùng các mã sau đây:

1.5.2 Mã BCD (Binary Coded Decimal)

Mã BCD dùng **số nhị phân 4 bit** có giá trị tương đương thay thế cho **từng số hạng** trong số thập phân.

Thí dụ:

Số 625_{10} có mã BCD là 0110 0010 0101.

Mã BCD dùng rất thuận lợi : mạch điện tử đọc các số BCD và hiển thị ra bằng đèn bảy đoạn (led hoặc LCD) hoàn toàn giống như con người đọc và viết ra số thập phân.

1.5.3 Mã Gray

Mã **Gray** hay còn gọi là mã **cách khoảng đơn vị**.

Nếu quan sát thông tin ra từ một máy đếm đang đếm các sự kiện tăng dần từng đơn vị, ta sẽ được các số nhị phân dần dần thay đổi. Tại thời điểm đang quan sát có thể có những lỗi rất quan trọng. Thí dụ giữa số 7(0111) và 8 (1000), các phần tử nhị phân đều phải thay đổi trong quá trình đếm, nhưng sự giao hoán này không bắt buộc xảy ra đồng thời, ta có thể có các trạng thái liên tiếp sau:

$$0111 \rightarrow 0110 \rightarrow 0100 \rightarrow 0000 \rightarrow 1000$$

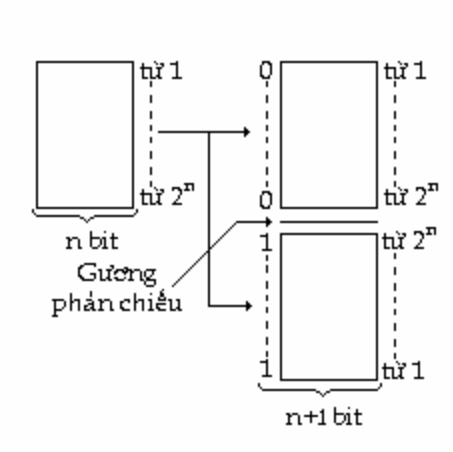
Trong một quan sát ngắn các kết quả thấy được khác nhau. Để tránh hiện tượng này, người ta cần mã hóa mỗi số hạng sao cho hai số liên tiếp chỉ khác nhau một phần tử nhị phân (1 bit) gọi là mã **cách khoảng đơn vị** hay mã **Gray**.

Tính kề nhau của các tổ hợp mã Gray (tức các mã liên tiếp chỉ khác nhau một bit) được dùng rất có hiệu quả để **rút gọn hàm logic tới mức tối giản**.

Ngoài ra, mã Gray còn được gọi là mã **phản chiếu** (do tính đối xứng của các số hạng trong tập hợp mã, giống như phản chiếu qua gương)

Người ta có thể thiết lập mã Gray bằng cách dựa vào tính đối xứng này:

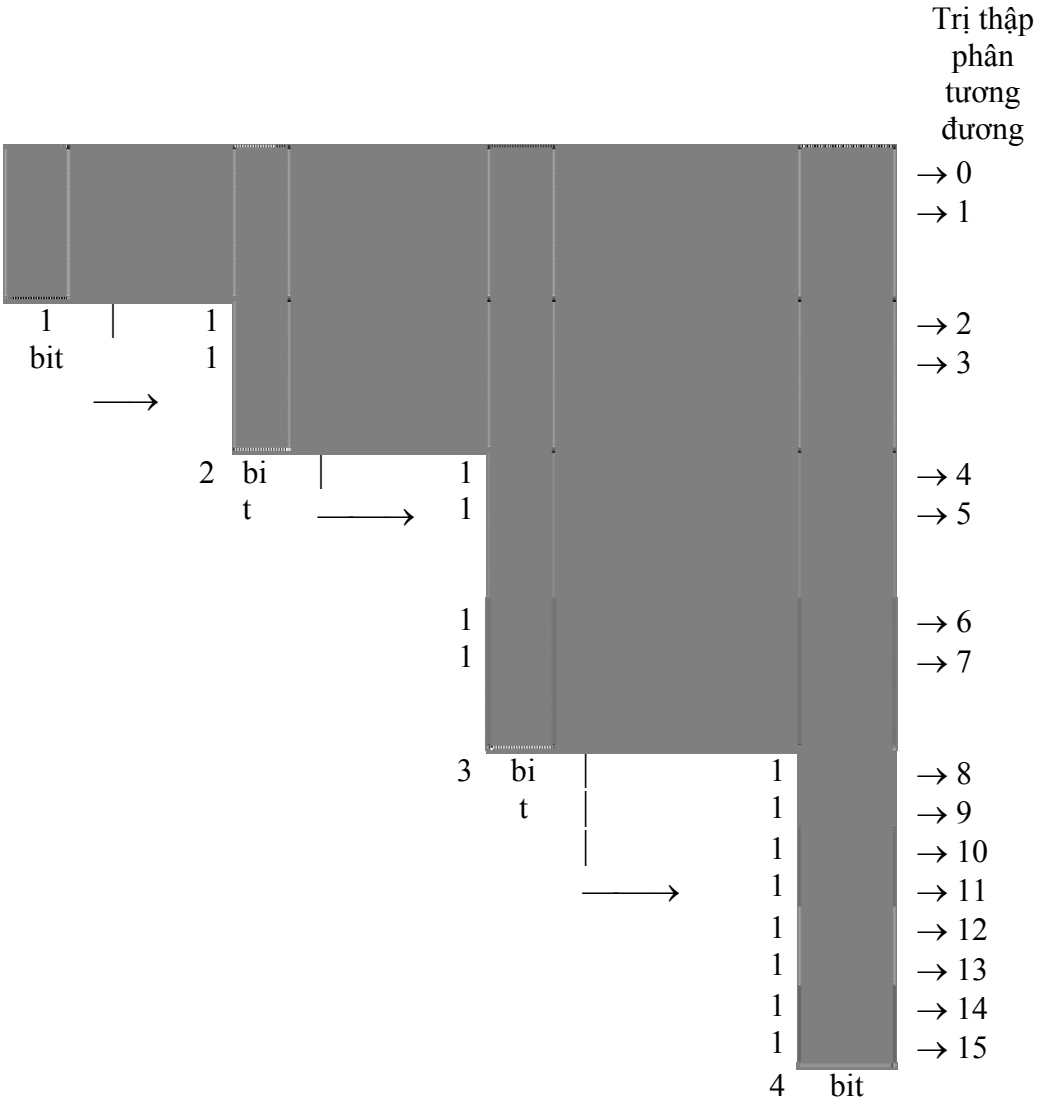
- Giả sử ta đã có tập hợp 2^n từ mã của số n bit thì có thể suy ra tập hợp 2^{n+1} từ mã của số (n+1) bit bằng cách:
- Viết ra 2^n từ mã theo thứ tự từ nhỏ đến lớn
- Thêm số 0 vào trước tất cả các từ mã đã có để được một phần của tập hợp từ mã mới
- Phần thứ hai của tập hợp gồm các từ mã giống như phần thứ nhất nhưng trình bày theo thứ tự ngược lại (giống như phản chiếu qua gương) và phía trước thêm vào số 1 thay vì số 0 (H 1.2).



(H 1.2)

Để thiết lập mã Gray của số nhiều bit ta có thể thực hiện các bước liên tiếp từ tập hợp đầu tiên của số một bit (gồm hai bit 0, 1).

Dưới đây là các bước tạo mã Gray của số 4 bit. Cột bên phải của bảng mã 4 bit cho giá trị tương đương trong hệ thập phân của mã Gray tương ứng (H 1.3).



(H 1.3)

Nhận xét các bảng mã của các số Gray (1 bit, 2 bit, 3 bit và 4 bit) ta thấy các số gần nhau luôn luôn khác nhau một bit, ngoài ra, trong từng bộ mã, các số đối xứng nhau qua gương cũng khác nhau một bit.

Bài Tập

1. Đổi các số thập phân dưới đây sang hệ nhị phân và hệ thập lục phân :
a/ 12 b/ 24 c/ 192 d/ 2079 e/ 15492
f/ 0,25 g/ 0,375 h/ 0,376 i/ 17,150 j/ 192,1875

2. Đổi sang hệ thập phân và mã BCD các số nhị phân sau đây:
a/ 1011 b/ 10110 c/ 101,1 d/ 0,1101
e/ 0,001 f/ 110,01 g/ 1011011 h/ 10101101011

3. Đổi các số thập lục phân dưới đây sang hệ 10 và hệ 8:
a/ FF b/ 1A c/ 789 d/ 0,13 e/ ABCD,EF

4. Đổi các số nhị phân dưới đây sang hệ 8 và hệ 16:
a/ 111001001,001110001 b/ 10101110001,00011010101
c/ 1010101011001100,1010110010101 d/ 1111011100001,01010111001

5. Mã hóa số thập phân dưới đây dùng mã BCD :
a/ 12 b/ 192 c/ 2079 d/15436 e/ 0,375 f/ 17,250

✍ CHƯƠNG 2 HÀM LOGIC

☸ HÀM LOGIC CƠ BẢN
☸ CÁC DẠNG CHUẨN CỦA HÀM LOGIC
- Dạng tổng chuẩn
- Dạng tích chuẩn
- Dạng số
- Biến đổi qua lại giữa các dạng chuẩn

☸ RÚT GỌN HÀM LOGIC
- Phương pháp đại số
- Phương pháp dùng bảng Karnaugh
- Phương pháp Quine Mc. Cluskey

Năm 1854 Georges Boole, một triết gia đồng thời là nhà toán học người Anh cho xuất bản một tác phẩm về lý luận logic, nội dung của tác phẩm đặt ra những mệnh đề mà để trả lời người ta chỉ phải dùng một trong hai từ đúng (có, yes) hoặc sai (không, no).

Tập hợp các thuật toán dùng cho các mệnh đề này hình thành môn Đại số Boole. Đây là môn toán học dùng hệ thống số nhị phân mà ứng dụng của nó trong kỹ thuật chính là các mạch logic, nền tảng của kỹ thuật số.

Chương này không có tham vọng trình bày lý thuyết Đại số Boole mà chỉ giới hạn trong việc giới thiệu các **hàm logic cơ bản** và các **tính chất cần thiết** để giúp sinh viên hiểu vận hành của một hệ thống logic.

2.1. HÀM LOGIC CƠ BẢN

2.1.1. Một số định nghĩa

- **Trạng thái logic**: trạng thái của một thực thể. Xét về mặt logic thì một thực thể chỉ tồn tại ở một trong hai trạng thái. Thí dụ, đối với một bóng đèn ta chỉ quan tâm nó đang ở trạng thái nào: tắt hay cháy. Vậy tắt / cháy là 2 trạng thái logic của nó.

- **Biến logic** dùng đặc trưng cho các trạng thái logic của các thực thể. Người ta biểu diễn biến logic bởi một ký hiệu (chữ hay dấu) và nó chỉ nhận 1 trong 2 giá trị : 0 hoặc 1.

Thí dụ trạng thái logic của một công tắc là đóng hoặc mở, mà ta có thể đặc trưng bởi trị 1 hoặc 0.

- **Hàm logic** diễn tả bởi một nhóm biến logic liên hệ nhau bởi các phép toán logic. Cũng như biến logic, hàm logic chỉ nhận 1 trong 2 giá trị: 0 hoặc 1 tùy theo các điều kiện liên quan đến các biến.

Thí dụ, một mạch gồm một nguồn hiệu thế cấp cho một bóng đèn qua hai công tắc mắc nối tiếp, bóng đèn chỉ cháy khi cả 2 công tắc đều đóng. Trạng thái của bóng đèn là một hàm theo 2 biến là trạng thái của 2 công tắc.

Gọi A và B là tên biến chỉ công tắc, công tắc đóng ứng với trị 1 và hở ứng với trị 0. Y là hàm chỉ trạng thái bóng đèn, 1 chỉ đèn cháy và 0 khi đèn tắt. Quan hệ giữa hàm Y và các biến A, B được diễn tả nhờ bảng sau:

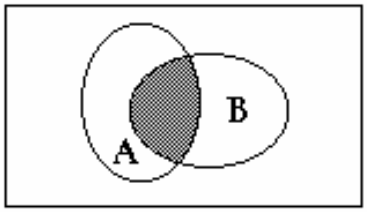
A	B	Y=f(A,B)
0 (hở)	0 (hở)	0 (tắt)
0 (hở)	1 (đóng)	0 (tắt)
1 (đóng)	0 (hở)	0 (tắt)
1 (đóng)	1 (đóng)	1 (cháy)

2.1.2. Biểu diễn biến và hàm logic

2.1.2.1. Giản đồ Venn

Còn gọi là giản đồ Euler, đặc biệt dùng trong lãnh vực tập hợp. Mỗi biến logic chia không gian ra 2 vùng không gian con, một vùng trong đó giá trị biến là đúng (hay=1), và vùng còn lại là vùng phụ trong đó giá trị biến là sai (hay=0).

Thí dụ: Phần giao nhau của hai tập hợp con A và B (gạch chéo) biểu diễn tập hợp trong đó A và B là đúng (A AND B) (H 2.1)



(H 2.1)

2.1.2.2. Bảng sự thật

Nếu hàm có n biến, bảng sự thật có n+1 cột và 2^n + 1 hàng. Hàng đầu tiên chỉ tên biến và hàm, các hàng còn lại trình bày các tổ hợp của n biến trong 2^n tổ hợp có thể có. Các cột đầu ghi giá trị của biến, cột cuối cùng ghi giá trị của hàm tương ứng với tổ hợp biến trên cùng hàng (gọi là trị riêng của hàm).

Thí dụ: Hàm OR của 2 biến A, B: f(A,B) = (A OR B) có bảng sự thật tương ứng.

A	B	f(A,B) = A OR B
0	0	0
0	1	1
1	0	1
1	1	1

2.1.2.3. Bảng Karnaugh

Đây là cách biểu diễn khác của bảng sự thật trong đó mỗi hàng của bảng sự thật được thay thế bởi một ô mà tọa độ (gồm hàng và cột) xác định bởi tổ hợp đã cho của biến.

Bảng Karnaugh của n biến gồm 2^n ô. Giá trị của hàm được ghi tại mỗi ô của bảng. Bảng Karnaugh rất thuận tiện để đơn giản hàm logic bằng cách nhóm các ô lại với nhau.

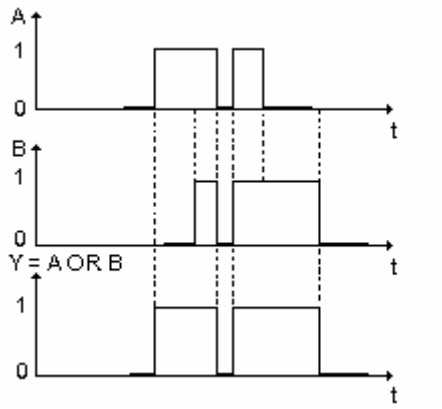
Thí dụ: Hàm OR ở trên được diễn tả bởi bảng Karnaugh sau đây

A \ B	0	1
0	0	1
1	1	1

2.1.2.4. Giản đồ thời gian

Dùng để diễn tả quan hệ giữa các hàm và biến theo thời gian, đồng thời với quan hệ logic.

Thí dụ: Giản đồ thời gian của hàm OR của 2 biến A và B, tại những thời điểm có một (hoặc 2) biến có giá trị 1 thì hàm có trị 1 và hàm chỉ có trị 0 tại những thời điểm mà cả 2 biến đều bằng 0.



(H 2.2)

2.1.3. Qui ước

Khi nghiên cứu một hệ thống logic, cần xác định qui ước logic. Qui ước này không được thay đổi trong suốt quá trình nghiên cứu.

Người ta dùng 2 mức điện thế thấp và cao để gán cho 2 trạng thái logic 1 và 0.

Qui ước **logic dương** gán điện thế thấp cho logic 0 và điện thế cao cho logic 1

Qui ước **logic âm** thì ngược lại.

2.1.4. Hàm logic cơ bản (Các phép toán logic)

2.1.4.1. Hàm NOT (đảo, bù) : $Y = \overline{A}$

Bảng sự thật

A	$Y = \overline{A}$
0	1
1	0

2.1.4.2. Hàm AND [tích logic, toán tử (.)] : Y = A.B

Bảng sự thật

A	B	Y=A.B
0	0	0
0	1	0
1	0	0

1	1	1

Nhận xét: Tính chất của hàm AND có thể được phát biểu như sau:
- Hàm AND của 2 (hay nhiều) biến chỉ có giá trị 1 khi tất cả các biến đều bằng 1
hoặc
- Hàm AND của 2 (hay nhiều) biến có giá trị 0 khi có một biến bằng 0.

2.1.4.3. Hàm OR [tổng logic, toán tử (+)] : Y = A + B

Bảng sự thật

A	B	Y=A + B
0	0	0
0	1	1
1	0	1
1	1	1

Nhận xét: Tính chất của hàm OR có thể được phát biểu như sau:
- Hàm OR của 2 (hay nhiều) biến chỉ có giá trị 0 khi tất cả các biến đều bằng 0
hoặc
- Hàm OR của 2 (hay nhiều) biến có giá trị 1 khi có một biến bằng 1.

2.1.4.4.Hàm EX-OR (OR loại trừ) $Y = A \oplus B$

Bảng sự thật

A	B	$Y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Nhận xét: Một số tính chất của hàm EX - OR:

- Hàm EX - OR của 2 biến chỉ có giá trị 1 khi hai biến khác nhau và ngược lại. Tính chất này được dùng để so sánh 2 biến.
- Hàm EX - OR của 2 biến cho phép thực hiện cộng hai số nhị phân 1 bit mà không quan tâm tới số nhớ.
- Từ kết quả của hàm EX-OR 2 biến ta suy ra bảng sự thật cho hàm 3 biến

A	B	C	$Y = A \oplus B \oplus C$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

- Trong trường hợp 3 biến (và suy rộng ra cho nhiều biến), hàm EX - OR có giá trị 1 khi số biến bằng 1 là số lẻ. Tính chất này được dùng để nhận dạng một chuỗi dữ liệu có số bit 1 là chẵn hay lẻ trong thiết kế mạch phát chẵn lẻ.

2.1.5. Tính chất của các hàm logic cơ bản:

2.1.5.1. Tính chất cơ bản:

♦ Có một phần tử trung tính duy nhất cho mỗi toán tử (+) và (.):
A + 0 = A ; **0** là phần tử trung tính của hàm OR
A . 1 = A ; **1** là phần tử trung tính của hàm AND

♦ Tính giao hoán:
A + B = B + A
A . B = B . A

♦ Tính phối hợp:
(A + B) + C = A + (B + C) = A + B + C
(A . B) . C = A . (B . C) = A . B . C

♦ Tính phân bố:
- Phân bố đối với phép nhân: **A . (B + C) = A . B + A . C**
- Phân bố đối với phép cộng: **A + (B . C) = (A + B) . (A + C)**

Phân bố đối với phép cộng là một tính chất đặc biệt của phép toán logic

♦ Không có phép tính lũy thừa và thừa số:
A + A + ... + A = A
A . A . . . . . A = A

♦ Tính bù:
$\overline{\overline{A}} = A$
$A + \overline{A} = 1$
$A.\overline{A} = 0$

2.1.5.2. Tính song đối (duality):

Tất cả biểu thức logic vẫn đúng khi [thay phép toán (+) bởi phép (.) và 0 bởi 1] hay ngược lại. Điều này có thể chứng minh dễ dàng cho tất cả biểu thức ở trên.

Thí dụ :

$$A + B = B + A \quad \Leftrightarrow \quad A.B = B.A$$
$$A + \overline{A}B = A + B \quad \Leftrightarrow \quad A(\overline{A} + B) = A.B$$
$$A + 1 = 1 \quad \Leftrightarrow \quad A.0 = 0$$

2.1.5.3. Định lý De Morgan

Định lý De Morgan được phát biểu bởi hai biểu thức:

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

Định lý De Morgan cho phép biến đổi qua lại giữa hai phép cộng và nhân nhờ vào phép đảo.

Định lý De Morgan được chứng minh bằng cách lập bảng sự thật cho tất cả trường hợp có thể có của các biến A, B, C với các hàm AND, OR và NOT của chúng.

2.1.5.4. Sự phụ thuộc lẫn nhau của các hàm logic cơ bản

Định lý De Morgan cho thấy các hàm logic không độc lập với nhau, chúng có thể biến đổi qua lại, sự biến đổi này cần có sự tham gia của hàm NOT. Kết quả là ta có thể dùng hàm (AND và NOT) hoặc (OR và NOT) để diễn tả tất cả các hàm.

Thí dụ:

Chỉ dùng hàm AND và NOT để diễn tả hàm sau: $\mathbf{Y = A.B + B.C + \overline{A}.C}$

Chỉ cần đảo hàm Y hai lần, ta được kết quả:

$$\mathbf{Y = \overline{\overline{Y}} = \overline{\overline{A.B + B.C + \overline{A}.C}} = \overline{\overline{A.B}.\overline{B.C}.\overline{\overline{A}.C}}}$$

Nếu dùng hàm OR và NOT để diễn tả hàm trên làm như sau:

$$\mathbf{Y = \overline{\overline{A.B} + \overline{B.C} + \overline{\overline{A}.C}} = \overline{\overline{A} + \overline{B} + \overline{B} + \overline{C} + A + \overline{C}}}$$

2.2. CÁC DẠNG CHUẨN CỦA HÀM LOGIC

Một hàm logic được biểu diễn bởi một tổ hợp của những tổng và tích logic.

♦ Nếu biểu thức là tổng của những tích, ta có dạng tổng

Thí dụ : $f(X,Y,Z) = XY + XZ + Y\overline{Z}$

♦ Nếu biểu thức là tích của những tổng, ta có dạng tích

Thí dụ : $f(X,Y,Z) = (X+Y).(X+Z).(Y+\overline{Z})$

Một hàm logic được gọi là hàm chuẩn nếu mỗi số hạng chứa đầy đủ các biến, ở dạng nguyên hay dạng đảo của chúng.

Thí dụ : $f(X,Y,Z) = XYZ + X\overline{Y}Z + XY\overline{Z}$ là một tổng chuẩn.

Mỗi số hạng của tổng chuẩn được gọi là **minterm**.

$f(X,Y,Z) = (X+Y+Z).(X+\overline{Y}+Z).(\overline{X}+Y+\overline{Z})$ là một tích chuẩn.

Mỗi số hạng của tích chuẩn được gọi là **maxterm**.

Phần sau đây cho phép chúng ta viết ra một hàm dưới dạng tổng chuẩn hay tích chuẩn khi có bảng sự thật diễn tả hàm đó.

2.2.1. Dạng tổng chuẩn

Để có được hàm logic dưới dạng chuẩn, ta áp dụng các **định lý triển khai của Shanon**.

Dạng tổng chuẩn có được từ triển khai theo **định lý Shanon thứ nhất**:

Tất cả các hàm logic có thể triển khai theo một trong những biến dưới dạng tổng của hai tích như sau:

$$\mathbf{f(A,B,...,Z) = A.f(1,B,...,Z) + \overline{A}.f(0,B,...,Z)} \qquad (1)$$

Hệ thức (1) có thể được chứng minh rất dễ dàng bằng cách lần lượt cho A bằng 2 giá trị 0 và 1, ta có kết quả là 2 vế của (1) luôn luôn bằng nhau. Thật vậy

Cho A=0: $\mathbf{f(0,B,...,Z) = 0.f(1,B,...,Z) + 1.\ f(0,B,...,Z) = f(0,B,...,Z)}$

Cho A=1: $\mathbf{f(1,B,...,Z) = 1.f(1,B,...,Z) + 0.\ f(0,B,...,Z) = f(1,B,...,Z)}$

Với 2 biến, hàm f(A,B) có thể triển khai theo biến A :

$$f(A,B) = A.f(1,B) + \overline{A}.f(0,B)$$

Mỗi hàm trong hai hàm vừa tìm được lại có thể triển khai theo biến B

$$f(1,B) = B.f(1,1) + B.f(1,0) \quad \& \quad f(0,B) = B.f(0,1) + \overline{B}.f(0,0)$$

Vậy: $f(A,B) = AB.f(1,1) + \overline{A}.B.f(0,1) + A\overline{B}.f(1,0) + \overline{A}\,\overline{B}.f(0,0)$

f(i,j) là giá trị riêng của f(A,B) khi A=i và B=j trong bảng sự thật của hàm.

Với 3 biến, trị riêng của f(A, B, C) là f(i, j, k) khi A=i, B=j và C=k ta được:

$$f(A,B,C) = A.B.C.f(1,1,1) + A.B.\overline{C}.f(1,1,0) + A.\overline{B}.C.f(1,0,1) + A.\overline{B}.\overline{C}.f(1,0,0) + \overline{A}.B.C.f(0,1,1) + \overline{A}.B.\overline{C}.f(0,1,0) + \overline{A}.\overline{B}.C.f(0,0,1) + \overline{A}.\overline{B}.\overline{C}.f(0,0,0)$$

Khi triển khai hàm 2 biến ta được tổng của $2^2 = 4$ số hạng

Khi triển khai hàm 3 biến ta được tổng của $2^3 = 8$ số hạng

Khi triển khai hàm n biến ta được tổng của 2^n số hạng

Mỗi số hạng là tích của một tổ hợp biến và một trị riêng của hàm. Hai trường hợp có thể xảy ra:

- Giá trị riêng = 1, số hạng thu gọn lại chỉ còn các biến:

  $\overline{A}.\overline{B}.C.f(0,0,1) = \overline{A}.\overline{B}.C$ nếu f(0,0,1) = 1

- Giá trị riêng = 0, tích bằng 0 :

  $\overline{A}.\overline{B}.\overline{C}.f(0,0,0) = 0$ nếu f(0,0,0) = 0

và số hạng này biến mất trong biểu thức của tổng chuẩn.

Thí dụ:

Cho hàm 3 biến A,B,C xác định bởi bảng sự thật:

Hàng			C	Z=f(A,B,C)
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

Với hàm Z cho như trên ta có các trị riêng f(i, j, k) xác định bởi:

f(0,0,1) = f(0,1,0) = f(0,1,1) = f(1,0,1) = f(1,1,1) =1

f(0,0,0) = f(1,0,0) = f(1,1,0) = 0

- Hàm Z có trị riêng f(0,0,1)=1 tương ứng với các giá trị của tổ hợp biến ở hàng (1) là A=0, B=0 và C=1 đồng thời, vậy $\overline{A}.\overline{B}.C$ là một số hạng trong tổng chuẩn
- Tương tự với các tổ hợp biến tương ứng với các hàng (2), (3), (5) và (7) cũng là các số hạng của tổng chuẩn, đó là các tổ hợp: $\overline{A}.B.\overline{C}$, $\overline{A}.B.C$, $A.\overline{B}.C$ và A.B.C
- Với các hàng còn lại (hàng 0,4,6), trị riêng của f(A,B,C) = 0 nên không xuất hiện trong triển khai.

Tóm lại ta có: $Z = \overline{A}.\overline{B}.C + \overline{A}.B.\overline{C} + \overline{A}.B.C + A.\overline{B}.C + A.B.C$

- Ý nghĩa của định lý Shanon thứ nhất:

Nhắc lại tính chất của các hàm AND và OR: $b_1.b_2.... b_n = 1$ khi $b_1, b_2..., b_n$ đồng thời bằng 1 và để $a_1 + a_2 + ... + a_p = 1$ chỉ cần ít nhất một biến $a_1, a_2, ..., a_p$ bằng 1

Trở lại thí dụ trên, biểu thức logic tương ứng với hàng 1 (A=0, B=0, C=1) được viết $\overline{A}.\overline{B}.C = 1$ vì $\overline{A} = 1$, $\overline{B} = 1$, C = 1 đồng thời.

Biểu thức logic tương ứng với hàng 2 là $\overline{A}.B.\overline{C} = 1$ vì A=0 ($\overline{A} = 1$), B=1, C=0 ($\overline{C} = 1$) đồng thời

Tương tự, với các hàng 3, 5 và 7 ta có các kết quả: $\overline{A}.B.C$, $A.\overline{B}.C$ và A.B.C

Như vậy, trong thí dụ trên

Z = hàng 1 + hàng 2 + hàng 3 + hàng 5 + hàng 7

$Z = \overline{A}.\overline{B}.C + \overline{A}.B.\overline{C} + \overline{A}.B.C + A.\overline{B}.C + A.B.C$

Tóm lại, từ một hàm cho dưới dạng bảng sự thật, ta có thể viết ngay biểu thức của hàm dưới dạng tổng chuẩn như sau:

- Số số hạng của biểu thức bằng số giá trị 1 của hàm thể hiện trên bảng sự thật

- Mỗi số hạng trong tổng chuẩn là tích của tất cả các biến tương ứng với tổ hợp mà hàm có trị riêng bằng 1, biến được giữ nguyên khi có giá trị 1 và được đảo nếu giá trị của nó = 0.

2.2.2. Dạng tích chuẩn

Đây là dạng của hàm logic có được từ triển khai theo định lý Shanon thứ hai:

Tất cả các hàm logic có thể triển khai theo một trong những biến dưới dạng tích của hai tổng như sau:

$$\mathbf{f(A,B,...,Z) = [\overline{A} + f(1,B,...,Z)].[A + f(0,B,...,Z)]} \qquad (2)$$

Cách chứng minh định lý Shanon thứ hai cũng giống như đã chứng minh định lý Shanon thứ nhất.

Với hai biến, hàm f(A,B) có thể triển khai theo biến A

$$f(A,B) = [\overline{A} + f(1,B)].[A + f(0,B)]$$

Mỗi hàm trong hai hàm vừa tìm được lại có thể triển khai theo biến B

$$f(1,B) = [\overline{B} + f(1,1)].[B + f(1,0)] \quad \& \quad f(0,B) = [\overline{B} + f(0,1)].[B + f(0,0)]$$

$$f(A,B) = \{\overline{A} + [\overline{B} + f(1,1)].[B + f(1,0)]\}.\{A + [\overline{B} + f(0,1)].[B + f(0,0)]\}$$

Vậy: $f(A,B) = [\overline{A}+\overline{B} + f(1,1)].[\overline{A}+B + f(1,0)].[A+\overline{B} + f(0,1)].[A+B + f(0,0)]$

Cũng như dạng chuẩn thứ nhất, f(i,j) là giá trị riêng của f(A,B) khi A=i và B=j trong bảng sự thật của hàm.

Với hàm 3 biến:

$f(A,B,C)=[\overline{A}+\overline{B}+\overline{C}+f(1,1,1)].[\overline{A}+\overline{B}+C+f(1,1,0)].[\overline{A}+B+\overline{C}+f(1,0,1)].[\overline{A}+B+C+f(1,0,0)].$
$[A+\overline{B}+\overline{C}+f(0,1,1)].[A+\overline{B}+C+ f(0,1,0)].[A+B+\overline{C}+f(0,0,1)].[A+B+C+f(0,0,0)]$

Số số hạng trong triển khai n biến là 2^n. Mỗi số hạng là tổng (OR) của các biến và trị riêng của hàm.

- Nếu trị riêng bằng 0 số hạng được rút gọn lại chỉ còn các biến (0 là trị trung tính của phép cộng logic)

$$A + B + C + f(0,0,0) = A + B + C \quad \text{nếu} \quad f(0,0,0) = 0$$

- Nếu trị riêng bằng 1, số hạng triển khai = 1

$$A + B + \overline{C} + f(0,0,1) = 1 \quad \text{nếu} \quad f(0,0,1) = 1$$

và biến mất trong biểu thức của tích chuẩn.

Lấy lại thí dụ trên:

Hàng			C	Z=f(A,B,C)
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

Các trị riêng của hàm đã nêu ở trên.

- Hàm Z có giá trị riêng f(0,0,0) = 0 tương ứng với các giá trị của biến ở hàng 0 là A=B=C=0 đồng thời, vậy A+B+C là một số hạng trong tích chuẩn.
- Tương tự với các hàng (4) và (6) ta được các tổ hợp $\overline{A}$+B+C và $\overline{A}+\overline{B}$+C.
- Với các hàng còn lại (hàng 1, 2, 3, 5, 7), trị riêng của f(A,B,C) = 1 nên không xuất hiện trong triển khai.

Tóm lại, ta có: $Z = (A + B + C).(\overline{A} + B + C).(\overline{A}+\overline{B}+C)$

- ***Ý nghĩa của định lý thứ hai:***

Nhắc lại tính chất của các hàm AND và OR: Để $b_1.b_2.... b_n = 0$ chỉ cần ít nhất một biến trong $b_1, b_2,..., b_n = 0$ và $a_1 + a_2 + ... + a_p = 0$ khi các biến $a_1, a_2, ..., a_p$ đồng thời bằng 0.

Như vậy trong thí dụ trên:

Z = (hàng 0).(hàng 4).(hàng 6)

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

Thật vậy, ở hàng 0 tất cả biến = 0: A=0, B=0, C=0 đồng thời nên có thể viết (A+B+C) = 0. Tương tự cho hàng (4) và hàng (6).

Tóm lại,

Biểu thức tích chuẩn gồm các thừa số, mỗi thừa số là tổng các biến tương ứng với tổ hợp có giá trị riêng =0, một biến giữ nguyên nếu nó có giá trị 0 và được đảo nếu có giá trị 1. Số thừa số của biểu thức bằng số số 0 của hàm thể hiện trên bảng sự thật.

2.2.3. Đổi từ dạng chuẩn này sang dạng chuẩn khác:

Nhờ định lý De Morgan, hai định lý trên có thể chuyển đổi qua lại.

Trở lại thí dụ trên, thêm cột $\overline{Z}$ vào bảng sự thật

\

Hàng	A	B	C	Z=f(A,B,C)	$\overline{Z}$

0	0	0	0	0	1
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	1	0
6	1	1	0	0	1
7	1	1	1	1	0

Diễn tả $\overline{Z}$ theo dạng tổng chuẩn:

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

Lấy đảo hai vế:

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

Dùng định lý De Morgan một lần nữa cho từng thừa số trong biểu thức, ta được:

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

Diễn tả $\overline{Z}$ theo dạng tích chuẩn:

$$\overline{Z} = (A + B + \overline{C})(A + \overline{B} + C)(A + \overline{B} + \overline{C})(\overline{A} + B + \overline{C})(\overline{A} + \overline{B} + \overline{C})$$

Lấy đảo hai vế:

$$Z = \overline{(A + B + \overline{C})(A + \overline{B} + C)(A + \overline{B} + \overline{C})(\overline{A} + B + \overline{C})(\overline{A} + \overline{B} + \overline{C})}$$

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

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

2.2.4. Dạng số

Để đơn giản cách viết người ta có thể diễn tả một hàm **Tổng chuẩn** hay **Tích chuẩn** bởi tập hợp các số dưới dấu tổng (Σ) hay tích (Π). Mỗi tổ hợp biến được thay bởi một số thập phân tương đương với trị nhị phân của chúng. Khi sử dụng cách viết này trọng lượng các biến phải được chỉ rõ.

Thí dụ : Cho hàm Z xác định như trên, tương ứng với dạng chuẩn thứ nhất, hàm này lấy giá trị của các hàng 1, 2, 3, 5, 7, ta viết Z=f(A,B,C) = Σ(1,2,3,5,7). Tương tự, nếu dùng dạng chuẩn thứ hai ta có thể viết Z =f(A,B,C)= Π(0,4,6).

Chú ý: *Khi viết các hàm theo dạng số ta phải chỉ rõ trọng số của các bit, thí dụ ta có thể ghi kèm theo hàm Z ở trên 1 trong 3 cách như sau: A=MSB hoặc C=LSB hoặc A=4, B=2, C=1*

2.3. RÚT GỌN HÀM LOGIC

Để thực hiện một hàm logic bằng mạch điện tử, người ta luôn luôn nghĩ đến việc sử dụng lượng linh kiện ít nhất. Muốn vậy, hàm logic phải ở dạng tối giản, nên vấn đề rút gọn hàm logic là **bước đầu tiên phải thực hiện** trong quá trình thiết kế. Có 3 phương pháp rút gọn hàm logic:

- Phương pháp đại số
- Phương pháp dùng bảng Karnaugh
- Phương pháp Quine Mc. Cluskey

2.3.1. Phương pháp đại số

Phương pháp này bao gồm việc áp dụng các tính chất của hàm logic cơ bản. Một số đẳng thức thường được sử dụng được nhóm lại như sau:

(1) $AB + \overline{A}B = B$ $(A+B).(\overline{A}+B) = B$ (1')
(2) $A + AB = A$ $A.(A+B) = A$ (2')
(3) $A + \overline{A}B = A + B$ $A.(\overline{A}+B) = A.B$ (3')

Chứng minh các đẳng thức 1, 2, 3:

(1) $AB + \overline{A}B = B(A+\overline{A}) = B.1 = B$
(2) $A + AB = A(1+B) = A$
(3) $A + \overline{A}B = (A+\overline{A}).(A+B) = A+B$

Các đẳng thức (1'), (2'), (3') là song đối của (1), (2), (3).

Các qui tắc rút gọn:

- Qui tắc 1: Nhờ các đẳng thức trên nhóm các số hạng lại.

Thí dụ: Rút gọn biểu thức $ABC + AB\overline{C} + A\overline{B}CD$

Theo (1) $ABC + AB\overline{C} = AB$

Vậy $ABC + AB\overline{C} + A\overline{B}CD = AB + A\overline{B}CD = A(B+\overline{B}CD)$

Theo (3) $B + \overline{B}CD = B + CD$

Và kết quả cuối cùng: $ABC + AB\overline{C} + A\overline{B}CD = A(B+CD)$

- Qui tắc 2: Ta có thể thêm một số hạng đã có trong biểu thức logic vào biểu thức mà không làm thay đổi biểu thức.

Thí dụ: Rút gọn biểu thức: $ABC + \overline{A}BC + A\overline{B}C + AB\overline{C}$

Thêm ABC vào để được: $(ABC + \overline{A}BC) + (ABC + A\overline{B}C) + (ABC + AB\overline{C})$

Theo (1) các nhóm trong dấu ngoặc rút gọn thành: BC + AC + AB

Vậy: $ABC + \overline{A}BC + A\overline{B}C + AB\overline{C} = BC + AC + AB$

- Qui tắc 3: Có thể bỏ số hạng chứa các biến đã có trong số hạng khác

Thí dụ 1: Rút gọn biểu thức $AB + \overline{B}C + AC$

Biểu thức không đổi nếu ta nhân một số hạng trong biểu thức với 1, ví dụ $(B+\overline{B})$:

$$AB + \overline{B}C + AC = AB + BC + AC(B+\overline{B})$$

Triển khai số hạng cuối cùng của vế phải, ta được:

$$AB + \overline{B}C + ABC + A\overline{B}C$$

Thừa số chung: $AB(1+C) + \overline{B}C(1+A) = AB + \overline{B}C$

Tóm lại: $AB + \overline{B}C + AC = AB + \overline{B}C$.

Trong bài tóan này ta đã đơn giản được số hạng AC.

Thí dụ 2: Rút gọn biểu thức $(A+B).(\overline{B}+C).(A+C)$

Biểu thức không đổi nếu ta thêm vào một thừa số có trị =0, ví dụ B.B

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

Theo (2') $(A+B).(A+B+C) = (A+B)$ và $(\overline{B}+C).(A+\overline{B}+C) = (\overline{B}+C)$

Vậy: $(A+B).(\overline{B}+C).(A+C) = (A+B).(\overline{B}+C)$

Trong bài tóan này ta đã bỏ số hạng A+C

- Qui tắc 4: Có thể đơn giản bằng cách dùng hàm chuẩn tương đương có số hạng ít nhất.

Thí dụ: Hàm f(A,B,C) = Σ(2,3,4,5,6,7) với trọng lượng A=4, B=2, C=1

Hàm đảo của f: $\overline{f(A,B,C)} = \Sigma(0,1) = \overline{A}.\overline{B}.\overline{C} + \overline{A}.\overline{B}.C = \overline{A}.\overline{B} = \overline{A+B}$

Vậy f(A,B,C) = A+B

2.3.2. Dùng bảng Karnaugh

Dùng bảng Karnaugh cho phép rút gọn dễ dàng các hàm logic chứa từ 3 tới 6 biến.

2.3.2.1.Nguyên tắc

Xét hai tổ hợp biến AB và $A\overline{B}$, hai tổ hợp này chỉ khác nhau một bit, ta gọi chúng là hai tổ hợp kề nhau.

Ta có: $AB + A\overline{B} = A$, biến B đã được đơn giản .

Phương pháp của bảng Karnaugh dựa vào việc nhóm các tổ hợp kề nhau trên bảng để đơn giản biến có giá trị khác nhau trong các tổ hợp này.

Công việc rút gọn hàm được thực hiện theo bốn bước:

- Vẽ bảng Karnaugh theo số biến của hàm
- Chuyển hàm cần đơn giản vào bảng Karnaugh
- Gom các ô chứa các tổ hợp kề nhau lại thành các nhóm sao cho có thể rút gọn hàm tới mức tối giản
- Viết kết quả hàm rút gọn từ các nhóm đã gom được.

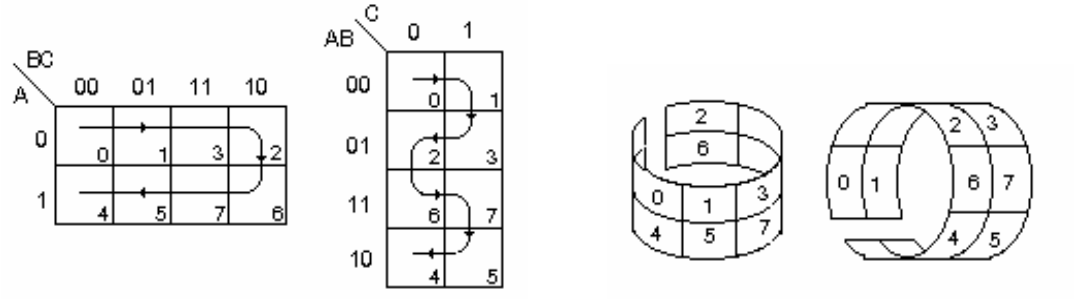
2.3.2.2 Vẽ bảng Karnaugh

- Bảng Karnaugh thực chất là một dạng khác của bảng sự thật, trong đó mỗi ô của bảng tương đương với một hàng trong bảng sự thật.

Để vẽ bảng Karnaugh cho n biến, người ta chia số biến ra làm đôi, phân nửa dùng để tạo $2^{n/2}$ cột, phân nửa còn lại tạo $2^{n/2}$ hàng (nếu n là số lẻ, người ta có thể cho số lượng biến trên cột lớn hơn số lượng biến cho hàng hay ngược lại cũng được). Như vậy, với một hàm có n biến, bảng Karnaugh gồm 2^n ô, mỗi ô tương ứng với tổ hợp biến này. Các ô trong bảng được sắp đặt sao cho hai ô kề nhau chỉ khác nhau một đơn vị nhị phân (khác nhau một bit), điều này cho thấy rất thuận tiện nếu chúng ta dùng mã Gray. Chính sự sắp đặt này cho phép ta đơn giản bằng cách nhóm các ô kề nhau lại.

Với 2 biến AB, sự sắp đặt sẽ theo thứ tự: AB = 00, 01, 11, 10 (đây là thứ tự mã Gray, nhưng để cho dễ ta dùng số nhị phân tương ứng để đọc thứ tự này: 0, 1, 3, 2)

Thí dụ : Bảng Karnaugh cho hàm 3 biến (A = MSB, và C = LSB) (H 2.3)



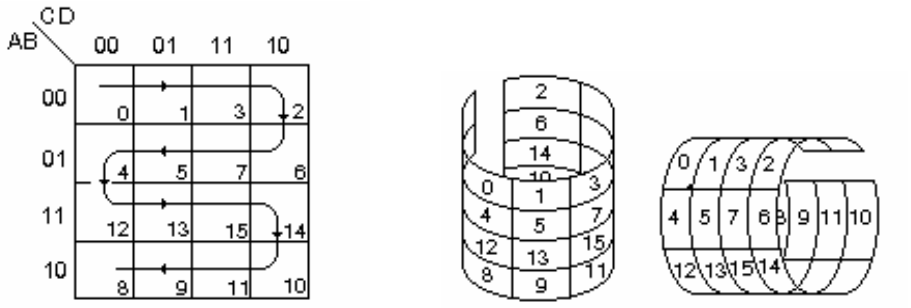
(H 2.3)

Với 3 biến ABC, ta được: ABC = 000, 001, 011, 010, 110, 111, 101, 100 (số nhị phân tương ứng: 0, 1, 3, 2, 6, 7, 5, 4)

Lưu ý là ta có thể thiết lập bảng Karnaugh theo chiều nằm ngang hay theo chiều đứng.

Do các tổ hợp ở các bìa trái và phải kề nhau nên ta có thể coi bảng có dạng hình trụ thẳng đứng và các tổ hợp ở bìa trên và dưới cũng kề nhau nên ta có thể coi bảng có dạng hình trụ trục nằm ngang. Và 4 tổ hợp biến ở 4 góc cũng là các tổ hợp kề nhau.

Hình (H 2.4) là bảng Karnaugh cho 4 biến.



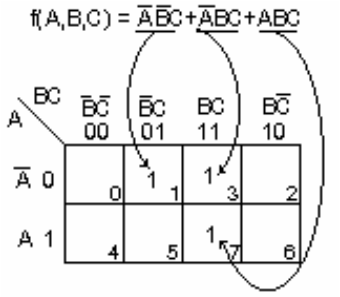
(H 2.4)

2.3.2.3. Chuyển hàm logic vào bảng Karnaugh.

Trong mỗi ô của bảng ta đưa vào giá trị của hàm tương ứng với tổ hợp biến, để đơn giản chúng ta có thể chỉ ghi các trị 1 mà bỏ qua các trị 0 của hàm. Ta có các trường hợp sau:

♦ **Từ hàm viết dưới dạng tổng chuẩn**:

Thí dụ 1 : $f(A,B,C) = \overline{A}.\overline{B}.C + \overline{A}.B.C + A.B.C$



(H 2.5)

♦ **Nếu hàm không phải là dạng chuẩn**, ta phải đưa về dạng chuẩn bằng cách thêm vào các số hạng sao cho hàm vẫn không đổi nhưng các số hạng chứa đủ các biến.

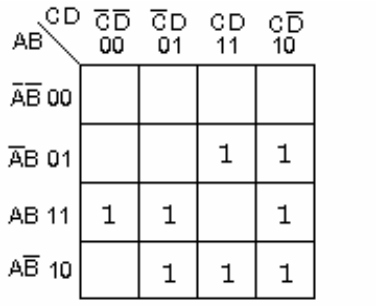
Thí dụ 2 : $Y = \overline{A}BC + AB\overline{D} + A\overline{B}C + A\overline{C}D$

Hàm này gồm 4 biến, nên để đưa về dạng tổng chuẩn ta làm như sau:

$Y = \overline{A}BC(D+\overline{D}) + AB\overline{D}(C+\overline{C}) + A\overline{B}C(D+\overline{D}) + A\overline{C}D(B+\overline{B})$

$Y = \overline{A}BCD + \overline{A}BC\overline{D} + ABC\overline{D} + AB\overline{C}\,\overline{D} + A\overline{B}CD + A\overline{B}C\overline{D} + AB\overline{C}D + A\overline{B}\,\overline{C}D$

Và Hàm Y được đưa vào bảng Karnaugh như sau (H 2.6):



(H 2.6)

♦ **Từ dạng số thứ nhất**, với các trọng lượng tương ứng A=4, B=2, C=1

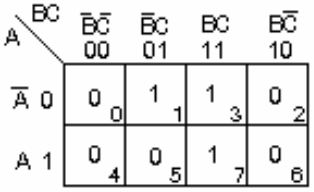
Thí dụ 3 : f(A,B,C) = Σ(1,3,7). Hàm số sẽ lấy giá trị 1 trong các ô 1,3 và 7.

♦ **Từ dạng tích chuẩn**: Ta lấy hàm đảo để có dạng tổng chuẩn và ghi trị 0 vào các ô tương ứng với tổ hợp biến trong tổng chuẩn này. Các ô còn lại chứa số 1.

Thí dụ 4 : $Y = f(A,B,C) = (A+B+C).(A+\overline{B}+C).(\overline{A}+B+C).(\overline{A}+B+\overline{C}).(\overline{A}+\overline{B}\ C)$

$\overline{Y} = \overline{A}.\overline{B}.\overline{C} + \overline{A}.B.\overline{C} + A.\overline{B}.\overline{C} + A.\overline{B}.C + A.B.\overline{C}$

Và bảng Karnaugh tương ứng (H 2.7).



(H 2.7)

♦ **Từ dạng số thứ hai**:

Thí dụ 5 : f(A,B,C) = Π(0,2,4,5,6)

Hàm sẽ lấy các trị 0 ở các ô 0, 2, 4, 5, 6. Dĩ nhiên là ta phải ghi các giá trị 1 trong các ô còn lại (H 2.7).

♦ **Từ bảng sự thật**:

Thí dụ 6 : Hàm f(A,B,C) cho bởi bảng sự thật

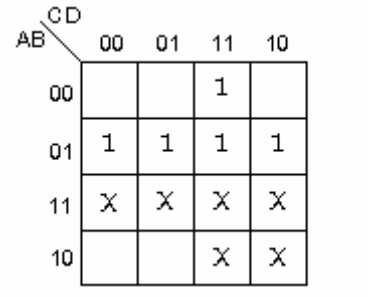
N	A	B	C	f(A,B,C)

0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

Ta ghi 1 vào các ô tương ứng với các tổ hợp biến ở hàng 1, 3 và 7, kết quả giống như ở thí dụ 1.

♦ **Trường hợp có một số tổ hợp cho giá trị hàm không xác định**: nghĩa là ứng với các tổ hợp này hàm có thể có giá trị 1 hoặc 0, do đó, ta ghi dấu X vào các ô tương ứng với các tổ hợp này, lúc gom nhóm ta sử dụng nó như số 1 hay số 0 một cách tùy ý sao cho có được kết quả rút gọn nhất.

Thí dụ 7: f(A,B,C,D) = Σ(3,4,5,6,7) với các tổ hợp từ 10 đến 15 cho hàm có trị bất kỳ (không xác định) (H 2.8).



(H 2.8)

2.3.2.4. Qui tắc gom nhóm

Các tổ hợp biến có trong hàm logic hiện diện trong bảng Karnaugh dưới dạng các số 1 trong các ô, vậy việc gom thành nhóm các tổ hợp kề nhau được thực hiện theo qui tắc sau:

- Gom các số 1 kề nhau thành từng nhóm sao cho số nhóm càng ít càng tốt. Điều này có nghĩa là số số hạng trong kết quả sẽ càng ít đi.

- Tất cả các số 1 phải được gom thành nhóm và một số 1 có thể ở nhiều nhóm.

- Số số 1 trong mỗi nhóm càng nhiều càng tốt nhưng phải là bội của 2^k (mỗi nhóm có thể có 1, 2, 4, 8 ... số 1). Cứ mỗi nhóm chứa 2^k số 1 thì tổ hợp biến tương ứng với nhóm đó giảm đi k số hạng.

- Kiểm tra để bảo đảm số nhóm gom được không thừa.

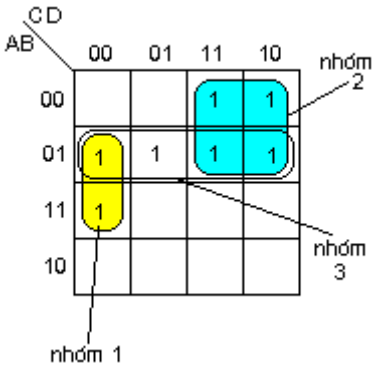
2.3.2.5. Qui tắc rút gọn

- Kết quả cuối cùng được lấy như sau:

Hàm rút gọn là tổng của các tích: Mỗi số hạng của tổng tương ứng với một nhóm các số 1 nói trên và số hạng này là tích của các biến, biến A (hay $\overline{A}$) là thừa số của tích khi tất cả các

số 1 của nhóm chỉ chứa trong phân nửa bảng trong đó biến A có giá trị 1 (hay 0). Nói cách khác nếu các số 1 của nhóm đồng thời nằm trong các ô của biến A và $\overline{A}$ thì biến A sẽ được đơn giản. Hình dưới đây minh họa việc lấy các thừa số trong tích

Thí dụ đối với bảng (H 2.9) ta có kết quả như sau:
- Hàm Y là hàm 4 biến A,B,C,D



(H 2.9)

- Nhóm 1 chứa 2 số 1 (k=1), như vậy nhóm 1 sẽ còn 3 biến, theo hàng, 2 số 1 này ở 2 ô ứng với $\overline{A}B$ và AB, biến A sẽ được đơn giản và theo cột thì 2 ô này ứng với tổ hợp $\overline{C}.\overline{D}$.

Kết quả ứng với nhóm 1 là: $B.\overline{C}.\overline{D}$

- Nhóm 2 chứa 4 số 1 ($4=2^2$, k=2), như vậy nhóm 2 sẽ còn 2 biến, theo hàng, 4 số 1 này ở 2 ô ứng với tổ hợp $\overline{A}\,\overline{B}$ và $\overline{A}B$, biến B sẽ được đơn giản và theo cột thì 4 ô này ứng với tổ hợp CD và C $\overline{D}$, cho phép đơn giản biến D .

Kết quả ứng với nhóm 2 là: $\overline{A}C$.

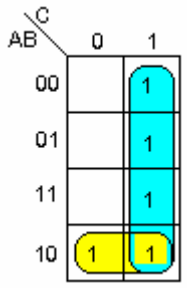
- Nhóm 3 chứa 4 số 1 ($4=2^2$, k=2), như vậy nhóm 2 sẽ còn 2 biến, theo hàng, 4 số 1 này ở ô ứng với tổ hợp $\overline{A}B$, theo cột 4 số 1 này chiếm hết 4 cột nên 2 biến Cvà D được đơn giản.

Kết quả ứng với nhóm 3 là: $\overline{A}B$.

Và hàm Y rút gọn là: **$Y = B\overline{C}\,\overline{D} + \overline{A}C + \overline{A}B$**

Dưới đây là một số thí dụ

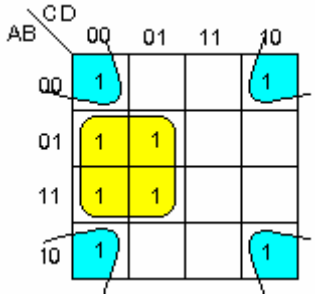
Thí dụ 1 : Rút gọn hàm $Y = f(A,B,C) = \overline{A}.\overline{B}.C + \overline{A}.B.C + A.\overline{B}.\overline{C} + A.\overline{B}.C + A.B.C$



(H 2.10)

(H 2.10) cho $Y = A\overline{B} + C$

Thí dụ 2 : Rút gọn hàm $Y = f(A,B,C,D) = \Sigma(0,2,4,5,8,10,12,13)$ với A=MSB



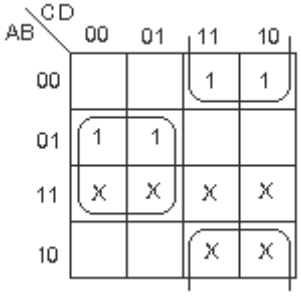
(H 2.11)

(H 2.11) cho $Y = B\overline{C} + \overline{B}\,\overline{D}$

Thí dụ 3 : Rút gọn hàm S cho bởi bảng sự thật:

N	A			D	S
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	0
10→15					x (Không xác định)

Bảng Karnaugh**:** (H 2.12)



(H 2.12)

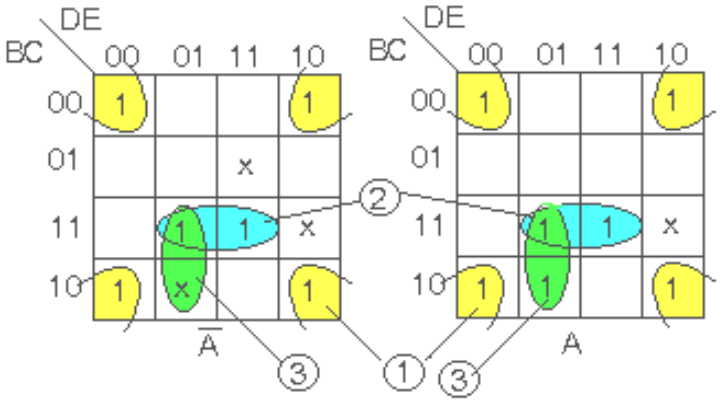
Kết quả : $S = B\overline{C} + \overline{B}C$

2.3.2.6. Rút gọn các hàm nhiều biến bằng cách dùng bảng Karnaugh 4 biến:

Để rút gọn các hàm nhiều biến (5 và 6 biến) người ta có thể dùng bảng Karnaugh 4 biến. Dưới đây là vài thí dụ:

Thí dụ 4 : Rút gọn hàm f(A,B,C,D,E) = ∑ (0,2,8,10,13,15,16,18,24,25,26,29,31) với (7,9,14,30) không xác định

- Trước nhất vẽ 2 bảng Karnaugh cho 4 biến BCDE, một ứng với $\overline{A}$ và một với A
- Bảng ứng với $\overline{A}$ dùng cho các số từ 0 đến 15
- Bảng ứng với A dùng cho các số từ 16 đến 31
- Nhóm các số 1 có cùng vị trí ở hai bảng, kết quả sẽ đơn giản biến A
- Nhóm các số 1 của từng bảng cho đến hết , kết quả được xác định như cách làm thông thường, nhớ A và $\overline{A}$ trong từng nhóm (H 2.13).



(H 2.13)

nhóm (1) cho : $\bar{C}\bar{E}$; (2) cho : BCE ; (3) cho : $B\bar{D}\bar{E}$

Vậy f(A,B,C,D,E) = $\bar{C}\bar{E}$ + BCE + $B\bar{D}\bar{E}$

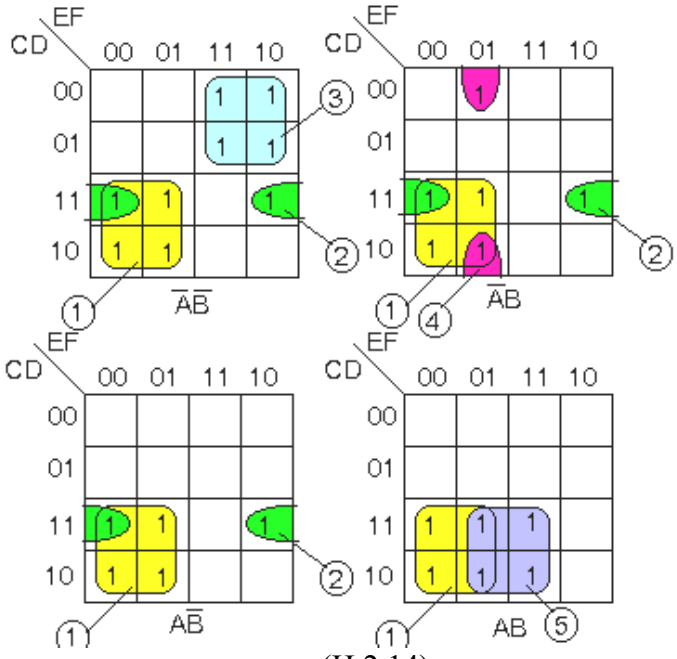
Thí dụ 5 : Rút gọn hàm

f(A,B,C,D,E,F)=∑(2,3,6,7,8,9,12,13,14,17,24,25,28,29,30,40,41,44,45,46,56,57,59,60,61,63)

Tương tự như trên nhưng phải vẽ 4 bảng cho:

$\bar{A}\bar{B}$ cho các số (0-15) ; $\bar{A}B$ cho các số (16-31) ;

AB cho các số (48-63) và $A\bar{B}$ cho các số (32-47).



(H 2.14)

Kết quả: (1) cho $\bar{C}\bar{E}$; (2) $\bar{A}CD\bar{F}$ + $\bar{B}CD\bar{F}$; (3) $\bar{A}\bar{B}\bar{C}\bar{E}$; (4) $\bar{A}BD\bar{E}\bar{F}$; (5) ABCF

Vậy:

f(A,B,C,D,E,F) = $\bar{C}\bar{E}$ + $\bar{A}\bar{B}\bar{C}\bar{E}$ + ABCF + $\bar{A}CD\bar{F}$ + $\bar{B}CD\bar{F}$ + $\bar{A}BD\bar{E}\bar{F}$

2.3.3. Phương pháp Quine-Mc. Cluskey

Phương pháp Quine-Mc. Cluskey cũng dựa trên tính kề của các tổ hợp biến để đơn giản số biến trong các số hạng của biểu thức dạng tổng (minterm). Trong quá trình đơn giản này có thể xuất hiện các số hạng giống nhau mà ta có thể bỏ bớt được.

Phương pháp được thực hiện qua 2 giai đoạn:

Giai đoạn 1: Dựa trên tính kề của các tổ hợp biến để đơn giản số biến trong các số hạng của biểu thức dạng tổng (minterm).

Giai đoạn 2: Kiểm tra và thực hiện việc tối giản .
Thí dụ dưới đây minh họa cho việc thực hiện phương pháp để rút gọn một hàm logic.

Thí dụ 1: Rút gọn hàm f(A,B,C,D) = Σ(1,2,4,5,6,10,12,13,14)

♣ **Giai đoạn 1**

- Các minterm được nhóm lại theo số số 1 có trong tổ hợp và ghi lại trong bảng theo thứ tự số 1 tăng dần:

Trong thí dụ này có 3 nhóm:
Nhóm chứa **một** số 1 gồm các tổ hợp 1, 2, 4
Nhóm chứa **hai** số 1 gồm các tổ hợp 5, 6, 10, 12
Nhóm chứa **ba** số 1 gồm các tổ hợp 13, 14
Bảng 1:

		A	B	C	D
x	1	0	0	0	1
x	2	0	0	1	0
x	4	0			0
x	5	0	1	0	1
x	6	0	1	1	0
x	10	1	0	1	0
x	12	1			0
x	13	1	1	0	1
x	14	1	1	1	0

- Mỗi tổ hợp trong một nhóm sẽ được so sánh với mỗi tổ hợp trong nhóm kế cận. Nếu 2 tổ hợp chỉ khác nhau một biến, ta có thể dùng biểu thức $AB + \overline{A}B = B$ để đơn giản được 1 biến. Biến đã đơn giản được thay bởi dấu -. Đánh dấu x vào các tổ hợp đã xét để tránh sai sót

Như vậy, tổ hợp thứ nhất của nhóm thứ nhất **0001** so sánh với tổ hợp thứ nhất của nhóm thứ hai **0101** vì chúng chỉ khác nhau ở biến B, vậy chúng có thể đơn giản thành **0-01**. Hai số hạng 1 và 5 đã được gom lại thành nhóm (1,5) và được ghi vào bảng 2.

Tiếp tục so sánh tổ hợp **0001** này với các tổ hợp còn lại của nhóm 2 (**0110**, **1010**, **1100**), vì chúng khác nhau nhiều hơn 1 bit nên ta không được kết quả nào khác. Như vậy, ta đã so sánh xong tổ hợp thứ nhất, đánh dấu x trước tổ hợp này để ghi nhớ.

Công việc tiến hành tương tự cho nhóm thứ hai và thứ ba.

Lưu ý: *Nhận xét về việc so sánh các tổ hợp với nhau ta thấy có thể thực hiện nhanh được bằng cách làm bài toán trừ 2 số nhị phân tương ứng của 2 tổ hợp, nếu kết quả là một số có trị* $= 2^k$ *(1, 2, 4,8 ...) thì 2 tổ hợp đó so sánh được và biến được đơn giản chính là biến có trọng*

số =2^k (thí dụ 2 tổ hợp 1 và 5 có hiệu số là 4 nên đơn giản được biến B), nếu hiệu số ≠ 2^k thì 2 tổ hợp đó không so sánh được, tức không có biến được đơn giản.

Kết quả cho bảng thứ hai

- Bảng thứ hai gồm các tổ hợp đã được rút gọn và chỉ còn lại 2 nhóm (giảm một nhóm so với bảng 1).

Bảng 2

		A			D
	1,5	0			1
x	2,6	0			0
x	2,10	-			0
x	4,5	0			-
x	4,6	0			0
x	4,12	-			0
x	5,13	-			1
x	6,14	-			0
x	10,14	1			0
x	12,13	1			-
x	12,14	1	1	-	0

Thực hiện công việc tương tự như trên với hai nhóm trong bảng thứ hai này, các số hạng sẽ được nhóm lại nếu chúng chỉ khác nhau một biến và có vị trí dấu - trùng nhau. Ta được bảng thứ 3.

Bảng 3:

	A			D
2,6 ; 10,14	-	-	1	0
2,10 ; 6,14	-	-	1	0
4,5 ; 12,13	-	1	0	-
4,6 ; 12,14	-	1	-	0
4,12 ; 5,13	-	1	0	-
4,12 ; 6,14	-	1	-	0

Quan sát bảng thứ 3 ta thấy có các tổ hợp giống nhau, như vậy ta có thể loại bỏ bớt các tổ hợp này và chỉ giữ lại một.

Kết quả của hàm rút gọn gồm tổng các số hạng tương ứng với các tổ hợp không gom thành nhóm trong các bảng đầu tiên, đó là tổ hợp (1,5) trong bảng 2, trị tương ứng là $\overline{A}\,\overline{C}D$ với các tổ hợp còn lại trong bảng cuối cùng, đó là các tổ hợp (2,6 ; 10,14) mà trị tương ứng là $C\overline{D}$, (4,5 ; 12,13) cho $B\overline{C}$ và (4,6 ; 12,14) cho $B\overline{D}$ trong bảng 3. Vậy:

$$\mathbf{f(A,B,C,D) = \overline{A}\,\overline{C}D + C\overline{D} + B\overline{C} + B\overline{D}}$$

Đến đây, nếu quan sát các tổ hợp cho các kết quả trên, ta thấy các tổ hợp còn chứa các số hạng giống nhau (số 4 và số 12 chẳng hạn), như vậy kết quả trên có thể là chưa tối giản.

♣ **Giai đoạn 2**:

Để có thể rút gọn hơn nữa ta lập một bảng như sau:

Cột bên trái ghi lại các tổ hợp đã chọn được trong giai đoạn 1, các cột còn lại ghi các trị thập phân có trong hàm ban đầu.

Trên cùng hàng của tổ hợp ta đánh dấu * dưới các cột có số tương ứng (ví dụ hàng chứa tổ hợp 1,5 có các dấu * ở cột 1 và 5). Tương tự cho các tổ hợp khác.

Bảng 4

		1								14
1,5	←	*↓								
2,6 ; 10,14	←									*↓
4,5 ; 12,13	←									
4,6 ; 12,14										*
		x	x	x	x	x	x	x	x	x

Xét các cột chỉ chứa một dấu *, đó là các cột 1,2,10 và 13, các tổ hợp ở cùng hàng với các dấu * này sẽ được chọn, đó là các tổ hợp (1,5), (2,6 ; 10,14), (4,5 ; 12,13), tương ứng với $\overline{A}\,\overline{C}D + C\overline{D} + B\overline{C}$. Đánh dấu X dưới các cột tương ứng với các số có trong các tổ hợp đã chọn. Nếu tất cả các cột đều được đánh dấu thì các tổ hợp đã chọn đủ để diễn tả hàm ban đầu.

Trong trường hợp của bài toán này, sau khi chọn các tổ hợp nói trên thì tất cả cột đã được đánh dấu do đó kết quả cuối cùng là (sau khi loai bỏ tổ hợp $B\overline{D}$):

$$\mathbf{f(A,B,C,D) = \overline{A}\,\overline{C}D + C\overline{D} + B\overline{C}}$$

Thí dụ 2: Rút gọn hàm f(A,B,C,D) = Σ(3,4,6,7,8,11,12,15)

♣ **Giai đoạn 1**

Bảng 1:

		A	B	C	D
x	4	0	1	0	0
x	8	0			0
x	3	0	0	1	1
x	6	0	1	1	0
x	12	1			0
x	7	0			1
x	11	1			1
x	15	1	1	1	1

So sánh các tổ hợp của 2 nhóm gần nhau ta được kết quả cho bảng thứ hai

- Bảng thứ hai gồm các tổ hợp đã được rút gọn và chỉ còn lại 3 nhóm (giảm một nhóm so với bảng 1).

Bảng 2

		A			D
	4,6	0			0
	4,12	-			0
	8,12	1			0
x	3,7	0			1
x	3,11	-			1
	6,7	0			-
x	7,15	-			1
x	11,15	1	-	1	1

Bảng 3:

	A			D
3,7 ; 11,15	-	-	1	1
3,11 ; 7,15	-	-	1	1

Kết quả của hàm rút gọn gồm tổng các số hạng tương ứng với các tổ hợp không gom thành nhóm: (4,6), (4,12), (8,12), (6,7) và (3,7;11,15)

$$f(A,B,C,D) = CD + \overline{A}B\overline{D} + B\overline{C}\,\overline{D} + A\overline{C}\,\overline{D} + \overline{A}BC$$

♣ **Giai đoạn 2**:

Bảng 4

	3							15
3,7;11,15 ←	*↓							*↓
4,6								
4,12								
8,12 ←								
6,7								
	x			x	x	x	x	x

Các cột 3, và 8 chỉ chứa một dấu ***,** các tổ hợp ở cùng hàng với các dấu ***** này sẽ được chọn, đó là các tổ hợp (3,7;11,15) và , (8,12), tương ứng với CD và $A\overline{C}\,\overline{D}$**.**

Đánh dấu X dưới các cột tương ứng với các số có trong các tổ hợp đã chọn.

Đến đây ta thấy còn 2 cột 4 và 6 chưa có dấu X, trong lúc chúng ta còn đến 3 tổ hợp để chọn. Dĩ nhiên trong trường hợp này ta chỉ cần chọn tổ hợp (4,6) ($\overline{A}B\overline{D}$ **)** thay vì chọn (4,12) và (6,7) thì đủ dấu X để lấp đầy các cột.

Tóm lại: $$f(A,B,C,D) = CD + \overline{A}B\overline{D} + A\overline{C}\,\overline{D}$$

Thí dụ về bài toán đầy đủ:

Thí dụ 1:

Cho hàm logic F(A, B, C) thỏa tính chất: F(A,B,C) = 1 nếu có một và chỉ một biến bằng 1

a- Lập bảng sự thật cho hàm F.

b- Rút gọn hàm F.

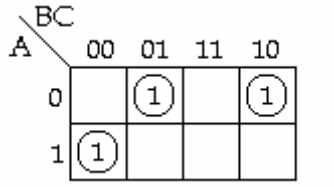
c- Diễn tả hàm F chỉ dùng hàm AND và NOT

Giải

a. Dựa vào điều kiện của bài toán ta có bảng sự thật của hàm F:

A	B	C	F(A,B,C)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

b. Rút gọn hàm F
Bảng Karnaugh



$$F(A,B.C) = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\,\overline{B}\,\overline{C}$$

c. Diễn tả hàm F chỉ dùng hàm AND và NOT
Dùng địnhlý De Morgan, lấy đảo 2 lần hàm F:

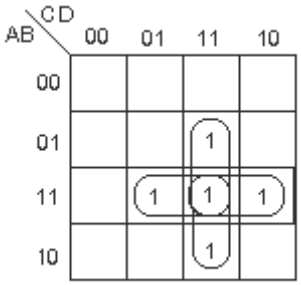
$$F(A,B.C) = \overline{\overline{F(A,B.C)}} = \overline{\overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\,\overline{B}\,\overline{C}} = \overline{\overline{A}\,\overline{B}C}.\overline{\overline{A}B\overline{C}}.\overline{A\,\overline{B}\,\overline{C}}$$

Thí dụ 2:
Cho hàm logic F(A, B, C, D) thỏa tính chất: F(A,B,C,D) = 1 khi có ít nhất 3 biến bằng 1

a- Rút gọn hàm F.
b- Diễn tả hàm F chỉ dùng hàm OR và NOT

Giải

a- Rút gọn hàm F
Ta có thể đưa hàm vô bảng Karnaugh mà không cần vẽ bảng sự thật.
Ta đưa số 1 vào tất cả các ô chứa 3 trị 1 trở lên



Và kết quả của hàm rút gọn là:
F(A,B,C,D) = ABC + ABD + ACD + BCD

b- Diễn tả hàm F chỉ dùng hàm OR và NOT
Dùng định lý De Morgan cho từng số hạng trong tổng
Viết lại hàm F:

$$F(A,B,C,D) = \overline{\overline{ABC}} + \overline{\overline{ABD}} + \overline{\overline{ACD}} + \overline{\overline{BCD}}$$
$$= \overline{\overline{A} + \overline{B} + \overline{C}} + \overline{\overline{A} + \overline{B} + \overline{D}} + \overline{\overline{A} + \overline{C} + \overline{D}} + \overline{\overline{B} + \overline{C} + \overline{D}}$$

BÀI TẬP

1. Diễn tả mỗi mệnh đề dưới đây bằng một biểu thức logic:
a/ Tất cả các biến A,B,C,D đều bằng 1
b/ Tất cả các biến A,B,C,D đều bằng 0

c/ Ít nhất 1 trong các biến X,Y,Z,T bằng 1
d/ Ít nhất 1 trong các biến X,Y,Z,T bằng 0
e/ Các biến A,B,C,D lần lượt có giá trị 0,1,1,0

2. Tính đảo của các hàm sau:

a/ $f_1 = (A + B)(\overline{A} + \overline{B})$
b/ $f_2 = (A + \overline{B} + \overline{C})(B + \overline{C} + D)(\overline{A} + C + D)$
c/ $f_3 = A(C + D) + (\overline{A} + C)(\overline{B} + C + D)$
d/ $f_4 = (AB + C)(BC + D) + \overline{A}BC + \overline{C}D$
e/ $f_5 = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A(BC + \overline{B}\,\overline{C})$

3. Chứng minh bằng đại số các biểu thức sau:

a/ $\overline{A.B + \overline{A}.\overline{B}} = \overline{A}.B + A.\overline{B}$
b/ $A.B + \overline{A}.C = (A + C)(\overline{A} + B)$
c/ $\overline{A.C + B.\overline{C}} = \overline{A}.C + \overline{B}.\overline{C}$
d/ $(A + B)(\overline{A} + C)(B + C) = (A + B)(\overline{A} + \overline{C})$
e/ $\overline{(A + C)(B + \overline{C})} = (\overline{A} + C)(\overline{B} + \overline{C})$

4. Viết dưới dạng tổng chuẩn các hàm xác định bởi:

a/ f(A,B,C) = 1 nếu số nhị phân $(ABC)_2$ là số chẵn
b/ f(A,B,C) = 1 nếu có ít nhất 2 biến số = 1
c/ f(A,B,C) = 1 nếu số nhị phân $(ABC)_2 > 5$
d/ f(A,B,C) = 1 nếu số biến số 1 là số chẵn
e/ f(A,B,C) = 1 nếu có 1 và chỉ 1 biến số =1

5. Viết dưới dạng tích chuẩn các hàm ở bài tập 4
6. Viết dưới dạng số các bài tập 4
7. Viết dưới dạng số các bài tập 5
8. Rút gọn các hàm dưới đây bằng phương pháp đại số (A = MSB)

a/ $f_1 = ABC + A\overline{B}C + AB\overline{C}D$
b/ $f_2 = (A+BC) + \overline{A}\ (\overline{B}+\overline{C})(AD+C)$
c/ $f_3 = (A+B+C)(A+B+\overline{C})(\overline{A}+B+C)(\overline{A}+B+\overline{C})$
d/ $f_4(A,B,C,D) = \Sigma(0,3,4,7,8,9,14,15)$
e/ $f_5 = \overline{A}B + AC + BC$
f/ $f_6 = (A+\overline{C})(B+C)(A+B)$

9. Dùng bảng Karnaugh rút gọn các hàm sau: (A = MSB)

a/ f(A,B,C) = Σ(1,3,4)
b/ f(A,B,C) = Σ(1,3,7)
c/ f(A,B,C) = Σ(0,3,4,6,7)
d/ f(A,B,C) = Σ(1,3,4) . Các tổ hợp biến 6,7 cho hàm không xác định
e/ $f(A,B,C) = \overline{A}.\overline{B}.C + \overline{A}.B.\overline{C} + A.\overline{B}.\overline{C} + A.B.C$
f/ f(A,B,C,D) = Σ(5,7,13,15)
g/ f(A,B,C,D) = Σ(0,4,8,12)
h/ f(A,B,C,D) = Σ(0,2,8,10)
i/ f(A,B,C,D) = Σ(0,2,5,6,9,11,13,14)
j/ f(A,B,C,D) = Π(0,1,5,9,10,15)
k/ f(A,B,C,D) = Π (0,5,9,10) với các tổ hợp biến (2,3,8,15) cho hàm không xác định
l/ f(A,B,C,D,E) = Σ(2,7,9,11,12,13,15,18,22,24,25,27,28,29,31)

m/ f(A,B,C,D.E) = Σ(0,2,8,10,13,15,16,18,24,25,26,29,31) với các tổ hợp biến (7,9,14,30) cho hàm không xác định
n/ f(A,B,C,D,E,F) = Σ(2,3,6,7,8,9,12,13,14,17,24,25,28,29,30,40,41,44,45,46,56,57,59,60,61,63)
o/ f(A,B,C,D,E,F) = Σ(9,11,13,15,16,18,20,22,25,27,29,31,32,34,36,38,41,43,45,47,48,50,52,54)

10. Làm lại các bài tập từ 9f bằng phương pháp Quine-Mc Cluskey.

💻 CHƯƠNG 3 CỔNG LOGIC

❖ **CÁC KHÁI NIỆM LIÊN QUAN**

❖ **CỔNG LOGIC CƠ BẢN**

❖ **THÔNG SỐ KỸ THUẬT**

❖ **Họ TTL**

✧ Cổng cơ bản
✧ Các kiểu ngã ra

❖ **Họ MOS**

✧ NMOS
✧ CMOS

❖ **GIAO TIẾP GIỮA CÁC HỌ IC SỐ**

✧ TTL thúc CMOS
✧ CMOS thúc TTL

Cổng logic là tên gọi chung của các mạch điện tử có chức năng thực hiện các hàm logic. Cổng logic có thể được chế tạo bằng các công nghệ khác nhau (Lưỡng cực, MOS), có thể được tổ hợp bằng các linh kiện rời nhưng thường được chế tạo bởi công nghệ tích hợp IC (Integrated circuit).

Chương này giới thiệu các loại cổng cơ bản, các họ IC số, các tính năng kỹ thuật và sự giao tiếp giữa chúng.

3.1 CÁC KHÁI NIỆM LIÊN QUAN

3.1.1 Tín hiệu tương tự và tín hiệu số

Tín hiệu tương tự là tín hiệu có biên độ biến thiên liên tục theo thời gian. Nó thường do các hiện tượng tự nhiên sinh ra. Thí dụ, tín hiệu đặc trưng cho tiếng nói là tổng hợp của các tín hiệu hình sin trong dải tần số thấp với các họa tần khác nhau.

Tín hiệu số là tín hiệu có dạng xung, gián đoạn về thời gian và biên độ chỉ có 2 mức rõ rệt: mức cao và mức thấp. Tín hiệu số chỉ được phát sinh bởi những mạch điện thích hợp. Để có tín hiệu số người ta phải số hóa tín hiệu tương tự bằng các mạch biến đổi tương tự sang số (ADC)

3.1.2 Mạch tương tự và mạch số

Mạch điện tử xử lý các tín hiệu tương tự được gọi là mạch tương tự và mạch xử lý tín hiệu số được gọi là mạch số.

Một cách tổng quát, mạch số có nhiều ưu điểm so với mạch tương tự:

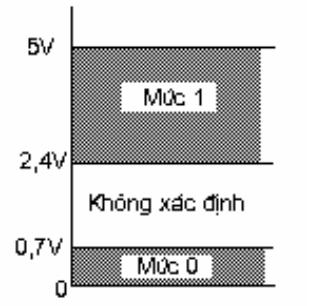
- Dễ thiết kế và phân tích. Vận hành của các cổng logic dựa trên tính chất dẫn điện (bảo hòa) hoặc ngưng dẫn của transistor. Việc phân tích và thiết kế dựa trên chức năng và đặc tính kỹ thuật của các IC và các khối mạch chứ không dựa trên từng linh kiện rời
- Có thể hoạt động theo chương trình lập sẵn nên rất thuận tiện trong điều khiển tự động, tính toán, lưu trữ dữ liệu và liên kết với máy tính.
- Ít bị ảnh hưởng của nhiễu tức có khả năng dung nạp tín hiệu nhiễu với biên độ lớn hơn rất nhiều so với mạch tương tự.
- Dễ chế tạo thành mạch tích hợp và có khả năng tích hợp với mật độ cao.

Dựa vào số cổng trong một chip, người ta phân loại IC số như sau:

- Số cổng < 10: SSI (Small Scale Integrated), mức độ tích hợp nhỏ.
- 10 < Số cổng < 100: MSI (Medium Scale Integrated), mức độ tích hợp trung bình.
- 100 < Số cổng < 1000: LSI (Large Scale Integrated), mức độ tích hợp lớn.
- 1000 < Số cổng < 10000: VLSI (Very Large Scale Integrated), mức độ tích hợp rất lớn
- Số cổng > 10000: ULSI (Ultra Large Scale Integrated), mức độ tích hợp siêu lớn.

3.1.3 Biểu diễn các trạng thái Logic 1 và 0

Trong hệ thống mạch logic, các trạng thái logic được biểu diễn bởi các mức điện thế. Với qui ước **logic dương**, điện thế cao biểu diễn logic 1, điện thế thấp biểu diễn logic 0. Ngược lại ta có qui ước **logic âm**. Trong thực tế, mức 1 và 0 tương ứng với một khoảng điện thế xác định và có một khoảng chuyển tiếp giữa mức cao và thấp, ta gọi là khoảng **không xác định**. Khi điện áp của tín hiệu rơi vào khoảng này, mạch sẽ không nhận ra là mức 0 hay 1. Khoảng này tùy thuộc vào họ IC sử dụng và được cho trong bảng thông số kỹ thuật của linh kiện. (H 3.1) là giản đồ điện thế của các mức logic của một số cổng logic thuộc họ TTL.

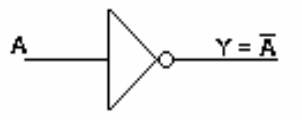


(H 3.1)

3.2 CỔNG LOGIC CƠ BẢN

3.2.1 Cổng NOT

- Còn gọi là cổng đảo (Inverter), dùng để thực hiện hàm đảo $Y = \overline{A}$
- Ký hiệu (H 3.2), mũi tên chỉ chiều di chuyển của tín hiệu và vòng tròn là ký hiệu đảo. Trong những trường hợp không thể nhầm lẫn về chiều này, người ta có thể bỏ mũi tên.



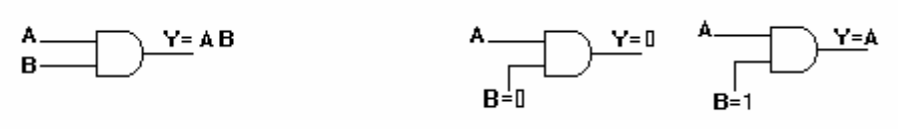
(H 3.2)

A	Y=$\overline{A}$
0	1
1	0

Bảng sự thật

3.2.2 Cổng AND

- Dùng thực hiện hàm AND 2 hay nhiều biến.
- Cổng AND có số ngã vào tùy thuộc số biến và một ngã ra. Ngã ra của cổng là hàm AND của các biến ngã vào.
- Ký hiệu cổng AND 2 ngã vào cho 2 biến (H 3.3a)



(a) (H 3.3) (b)

A	B	Y=A.B
0	0	0
0	1	0
1	0	0
1	1	1

Hoặc

A	B	Y=A.B
x	0	0
x	1	A

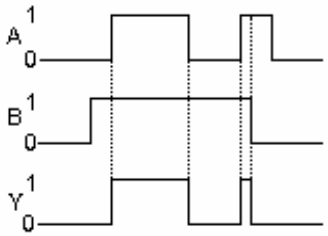
- Nhận xét:

- Ngã ra cổng AND chỉ ở mức cao khi tất cả ngã vào lên cao.
- Khi có một ngã vào = 0, ngã ra = 0 bất chấp các ngã vào còn lại.
- Khi có một ngã vào =1, ngã ra = AND của các ngã vào còn lại.

Vậy với cổng AND 2 ngã vào ta có thể dùng 1 ngã vào làm ngã kiểm soát (H 3.3b), khi ngã kiểm soát = 1, **cổng mở** cho phép tín hiệu logic ở ngã vào còn lại qua cổng và khi ngã kiểm soát = 0, **cổng đóng** , ngã ra luôn bằng 0, bất chấp ngã vào còn lại.

Với cổng AND có nhiều ngã vào hơn, khi có một ngã vào được đưa lên mức cao thì ngã ra bằng AND của các biến ở các ngã vào còn lại.

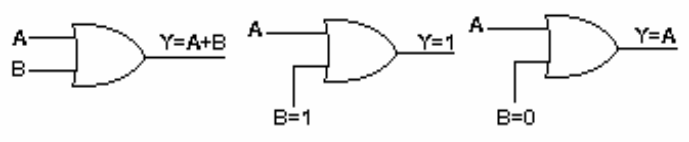
Hình (H 3.4) là giản đồ thời gian của cổng AND hai ngã vào. Trên giản đồ, ngã ra Y chỉ lên mức 1 khi cả A và B đều ở mức 1.



(H 3.4)

3.2.3 Cổng OR

- Dùng để thực hiện hàm OR 2 hay nhiều biến.
- Cổng OR có số ngã vào tùy thuộc số biến và một ngã ra.
- Ký hiệu cổng OR 2 ngã vào



(H 3.5)

- Bảng sự thật

A	B	Y=A+B
0	0	0
0	1	1
1	0	1
1	1	1

Hoặc

A	B	Y=A+B
x	1	1
x	0	A

- Nhận xét: - Ngã ra cổng OR chỉ ở mức thấp khi cả 2 ngã vào xuống thấp.
- Khi có một ngã vào =1, ngã ra = 1 bất chấp ngã vào còn lại.
- Khi có một ngã vào =0, ngã ra = OR các ngã vào còn lại.

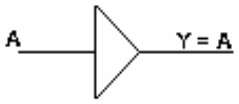
Vậy với cổng OR 2 ngã vào ta có thể dùng 1 ngã vào làm ngã kiểm soát, khi ngã kiểm soát = 0, **cổng mở**, cho phép tín hiệu logic ở ngã vào còn lại qua cổng và khi ngã kiểm soát = 1, **cổng đóng**, ngã ra luôn bằng 1.

Với cổng OR nhiều ngã vào hơn, khi có một ngã vào được đưa xuống mức thấp thì ngã ra bằng OR của các biến ở các ngã vào còn lại.

3.2.4 Cổng BUFFER

Còn gọi là cổng đệm. Tín hiệu số qua cổng BUFFER không đổi trạng thái logic. Cổng BUFFER được dùng với các mục đích sau:

- Sửa dạng tín hiệu.
- Đưa điện thế của tín hiệu về đúng chuẩn của các mức logic.
- Nâng khả năng cấp dòng cho mạch.
- Ký hiệu của cổng BUFFER.



(H 3.6)

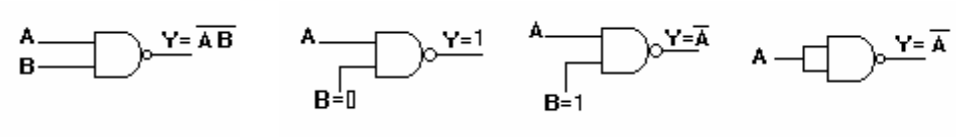
Tuy cổng đệm không làm thay đổi trạng thái logic của tín hiệu vào cổng nhưng nó giữ vai trò rất quan trọng trong các mạch số.

3.2.5 Cổng NAND

- Là kết hợp của cổng AND và cổng NOT, thực hiện hàm $Y = \overline{A.B}$

(Ở đây chỉ xét cổng NAND 2 ngã vào, độc giả tự suy ra trường hợp nhiều ngã vào).

- Ký hiệu của cổng NAND (Gồm AND và NOT, cổng NOT thu gọn lại một vòng tròn)
- Tương tự như cổng AND, ở cổng NAND ta có thể dùng 1 ngã vào làm ngã kiểm soát. Khi ngã kiểm soát = 1, cổng mở cho phép tín hiệu logic ở ngã vào còn lại qua cổng và bị đảo, khi ngã kiểm soát = 0, cổng đóng, ngã ra luôn bằng 1.
- Khi nối tất cả ngã vào của cổng NAND lại với nhau, nó hoạt động như một cổng đảo

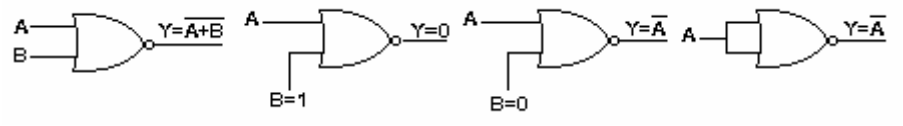


(H 3.7)

3.2.6 Cổng NOR

- Là kết hợp của cổng OR và cổng NOT, thực hiện hàm $\mathbf{Y = \overline{A + B}}$

Ký hiệu của cổng NOR (Gồm cổng OR và NOT, nhưng cổng NOT thu gọn lại một vòng tròn)

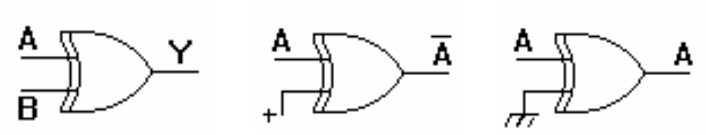


(H 3.8)

Các bảng sự thật và các giản đồ thời gian của các cổng BUFFER, NAND, NOR, sinh viên có thể tự thực hiện lấy

3.2.7 Cổng EX-OR

- Dùng để thực hiện hàm EX-OR. $\mathbf{Y = A \oplus B = \overline{A}B + A\overline{B}}$
- Cổng EX-OR chỉ có 2 ngã vào và 1 ngã ra
- Ký hiệu (H 3.9a)
- Một tính chất rất quan trọng của cổng EX-OR:
  + Tương đương với một cổng đảo khi có một ngã vào nối lên mức cao, (H 3.9b)
  + Tương đương với một cổng đệm khi có một ngã vào nối xuống mức thấp, (H 3.9c)



(a) (b) (c)

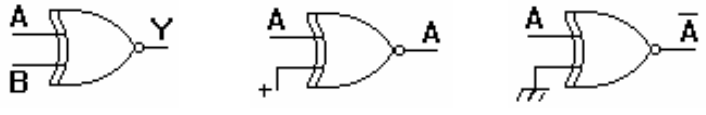
(H 3.9)

3.2.8 Cổng EX-NOR

- Là kết hợp của cổng EX-OR và cổng NOT
- Cổng EX-NOR có 2 ngã vào và một ngã ra
- Hàm logic ứng với cổng EX-NOR là

$$\mathbf{Y = \overline{A \oplus B} = \overline{A}\overline{B} + A.B}$$

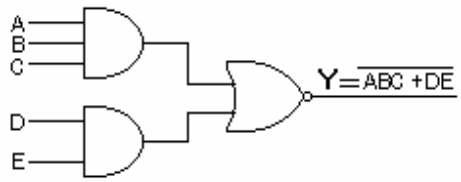
- Ký hiệu (H 3.10)
- Các tính chất của cổng EX-NOR giống cổng EX-OR nhưng có ngã ra đảo lại.



(H 3.10)

3.2.9 Cổng phức AOI (AND-OR-INVERTER)

Ứng dụng các kết quả của Đại số BOOLE, người ta có thể kết nối nhiều cổng khác nhau trên một chip IC để thực hiện một hàm logic phức tạp nào đó. Cổng AOI là một kết hợp của 3 loại cổng AND (A), OR (O) và INVERTER (I). Thí dụ để thực hiện hàm logic $\mathbf{Y = \overline{A.B.C + D.E}}$, ta có cổng phức sau:



(H 3.11)

3.2.10 Biến đổi qua lại giữa các cổng logic

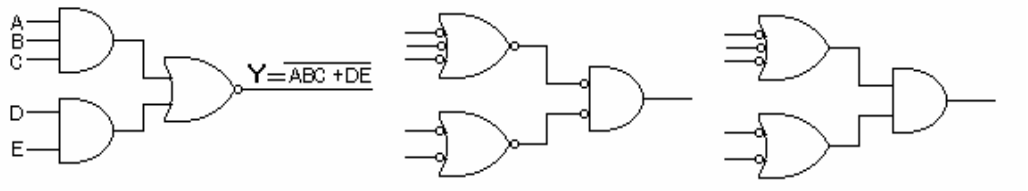
Trong chương Hàm Logic chúng ta đã thấy tất cả các hàm logic có thể được thay thế bởi 2 hàm duy nhất là hàm AND (hoặc OR) kết hợp với hàm NOT. Các cổng logic có chức năng thực hiện hàm logic, như vậy chúng ta chỉ cần dùng 2 cổng AND (hoặc OR) và NOT để thực hiện tất cả các hàm logic. Tuy nhiên, vì cổng NOT cũng có thể tạo ra từ cổng NAND (hoặc NOR). Như vậy, tất cả các hàm logic có thể được thực hiện bởi một cổng duy nhất, đó là cổng NAND (hoặc NOR). Hàm ý này cho phép chúng ta biến đổi qua lại giữa các cổng với nhau.

Quan sát Định lý De Morgan chúng ta rút ra qui tắc biến đổi qua lại giữa các cổng AND, NOT và OR , NOT như sau:

Chỉ cần thêm các cổng đảo ở ngã vào và ngã ra khi biến đổi từ AND sang OR hoặc ngược lại. Dĩ nhiên nếu ở các ngã đã có đảo rồi thì đảo này sẽ mất đi.

Thí dụ 1: Ba mạch dưới đây tương đương nhau:

(H 3.12b) có được bằng cách đổi AND - OR thêm các đảo ở các ngã vào và ra. Từ (H 3.12b) đổi sang (H 3.12c) ta bỏ 2 cổng đảo nối từ ngã ra cổng NOR đến ngã vào cổng AND



(a) (b) (c)

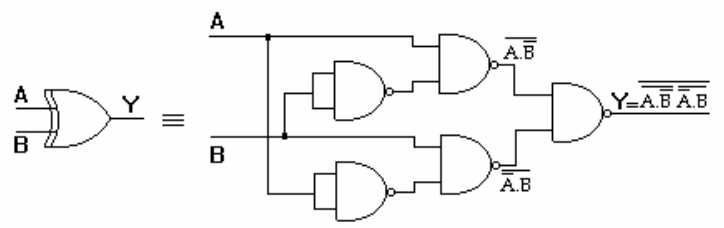
(H 3.12)

Thí dụ 2: Vẽ mạch tương đương của cổng EX-OR dùng toàn cổng NAND

Dùng định lý De-Morgan, biểu thức hàm EX-OR viết lại:

$$Y = A\overline{B} + \overline{A}B = \overline{\overline{A\overline{B}}.\overline{\overline{A}B}}$$

Và mạch tương đương cho ở (H 3.13)



(H 3.13)

3.3 THÔNG SỐ KỸ THUẬT CỦA IC SỐ

Để sử dụng IC số có hiệu quả, ngoài sơ đồ chân và bảng sự thật của chúng, ta nên biết qua một số thuật ngữ chỉ các thông số cho biết các đặc tính của IC.

3.3.1 Các đại lượng điện đặc trưng

- **V_{CC}**: Điện thế nguồn (power supply): khoảng điện thế cho phép cấp cho IC để hoạt động tốt. Thí dụ với IC số họ TTL, V_{CC}=5±0,5 V , họ CMOS V_{DD}=3-15V (Người ta thường dùng ký hiệu V_{DD} và V_{SS} để chỉ nguồn và mass của IC họ MOS)
- **V_{IH}**(min): Điện thế ngã vào mức cao (High level input voltage): Đây là điện thế ngã vào nhỏ nhất còn được xem là mức 1
- **V_{IL}**(max): Điện thế ngã vào mức thấp (Low level input voltage): Điện thế ngã vào lớn nhất còn được xem là mức 0.
- **V_{OH}**(min): Điện thế ngã ra mức cao (High level output voltage): Điện thế nhỏ nhất của ngã ra khi ở mức cao.
- **V_{OL}**(max): Điện thế ngã ra mức thấp (Low level output voltage): Điện thế lớn nhất của ngã ra khi ở mức thấp.
- **I_{IH}**: Dòng điện ngã vào mức cao (High level input current): Dòng điện lớn nhất vào ngã vào IC khi ngã vào này ở mức cao.
- **I_{IL}**: Dòng điện ngã vào mức thấp (Low level input current) : Dòng điện ra khỏi ngã vào IC khi ngã vào này ở mức thấp
- **I_{OH}**: Dòng điện ngã ra mức cao (High level output current): Dòng điện lớn nhất ngã ra có thể cấp cho tải khi nó ở mức cao.
- **I_{OL}**: Dòng điện ngã ra mức thấp (Low level output current): Dòng điện lớn nhất ngã ra có thể nhận khi ở mức thấp.
- **I_{CCH},I_{CCL}**: Dòng điện chạy qua IC khi ngã ra lần lượt ở mức cao và thấp.

Ngoài ra còn một số thông số khác được nêu ra dưới đây

3.3.2 Công suất tiêu tán (Power requirement)

Mỗi IC khi hoạt động sẽ tiêu thụ một công suất từ nguồn cung cấp V_{CC} (hay V_{DD}). Công suất tiêu tán này xác định bởi điện thế nguồn và dòng điện qua IC. Do khi hoạt động dòng qua IC thường xuyên thay đổi giữa hai trạng thái cao và thấp nên công suất tiêu tán sẽ được tính từ dòng trung bình qua IC và công suất tính được là công suất tiêu tán trung bình

$$P_D(avg) = I_{CC}(avg).V_{CC}$$

Trong đó

$$I_{CC}(avg) = \frac{I_{CCH} + I_{CCL}}{2}$$

Đối với các cổng logic họ TTL, công suất tiêu tán ở hàng mW và với họ MOS thì chỉ ở hàng nW.

3.3.3 Fan-Out:

Một cách tổng quát, ngã ra của một mạch logic đòi hỏi phải cấp dòng cho một số ngã vào các mạch logic khác. Fan Out là số ngã vào lớn nhất có thể nối với ngã ra của một IC cùng loại mà vẫn bảo đảm mạch hoạt động bình thường. Nói cách khác Fan Out chỉ khả năng chịu tải của một cổng logic

Ta có hai loại Fan-Out ứng với 2 trạng thái logic của ngã ra:

$$Fan-Out_H = \frac{I_{OH}}{I_{IH}}$$

$$Fan-Out_L = \frac{I_{OL}}{I_{IL}}$$

Thường hai giá trị Fan-Out này khác nhau, khi sử dụng, để an toàn, ta nên dùng trị nhỏ nhất trong hai trị này.

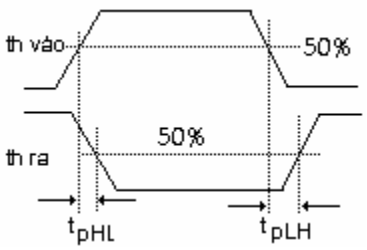
Fan-Out được tính theo đơn vị Unit Load UL (tải đơn vị).

3.3.4 Thời trễ truyền (Propagation delays)

Tín hiệu logic khi truyền qua một cổng luôn luôn có một thời gian trễ.

Có hai loại thời trễ truyền: Thời trễ truyền từ thấp lên cao t_{PLH} và thời trễ truyền từ cao xuống thấp t_{PHL}. Hai giá trị này thường khác nhau. Sự thay đổi trạng thái được xác định ở tín hiệu ra. Thí dụ tín hiệu qua một cổng đảo, thời trễ truyền được xác định như ở (H 3.14)

Tùy theo họ IC, thời trễ truyền thay đổi tử vài ns đến vài trăm ns. Thời trễ truyền càng lớn thì tốc độ làm việc của IC càng nhỏ.



(H 3.14)

3.3.5 Tích số công suất-vận tốc (speed- power product)

Để đánh giá chất lượng IC, người ta dùng đại lượng tích số công suất-vận tốc đó là tích số công suất tiêu tán và thời trễ truyền. Thí dụ họ IC có thời trễ truyền là 10 ns và công suất tiêu tán trung bình là 50 mW thì tích số công suất-vận tốc là:

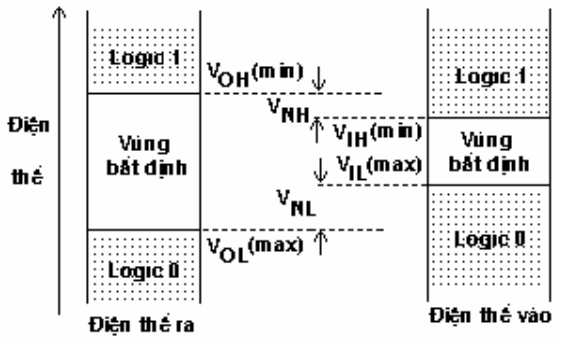
$$10 \text{ ns} \times 5 \text{ mW} = 10.10^{-9} \times 5.10^{-3} = 50 \times 10^{-12} \text{ watt-sec} = 50 \text{ picojoules (pj)}$$

Trong quá trình phát triển của công nghệ chế tạo IC người ta luôn muốn đạt được các IC có công suất tiêu tán và thời trễ truyền càng nhỏ càng tốt. Như vậy một IC có chất lượng càng tốt khi tích số công suất-vận tốc càng nhỏ. Tuy nhiên trên thực tế hai giá trị này thay đổi theo chiều ngược với nhau, nên ta khó mà đạt được các giá trị theo ý muốn, dù sao trong quá trình phát triển của công nghệ chế tạo linh kiện điện tử trị số này luôn được cải thiện .

3.3.6 Tính miễn nhiễu (noise immunity)

Các tín hiệu nhiễu như tia lửa điện, cảm ứng từ có thể làm thay đổi trạng thái logic của tín hiệu do đó ảnh hưởng đến kết quả hoạt động của mạch.

Tính miễn nhiễu của một mạch logic tùy thuộc khả năng dung nạp hiệu thế nhiễu của mạch và được xác định bởi lề nhiễu. Lề nhiễu có được do sự chênh lệch của các điện thế giới hạn (còn được gọi là ngưỡng logic) của mức cao và thấp giữa ngã ra và ngã vào của các cổng (H 3.15).



(H 3.15)

Tín hiệu khi vào mạch logic được xem là mức 1 khi có trị $>V_{IH}(min)$ và là mức 0 khi $<V_{IL}(max)$. Điện thế trong khoảng giữa không ứng với một mức logic nào nên gọi là vùng bất định. Do có sự khác biệt giữa $V_{OH}(min)$ với $V_{IH}(min)$ và $V_{OL}(max)$ với $V_{IL}(max)$ nên ta có 2 giá trị lề nhiễu:

Lề nhiễu mức cao: $V_{NH} = V_{OH}(min) - V_{IH}(min)$

Lề nhiễu mức thấp: $V_{NL} = V_{IL}(max) - V_{OL}(max)$

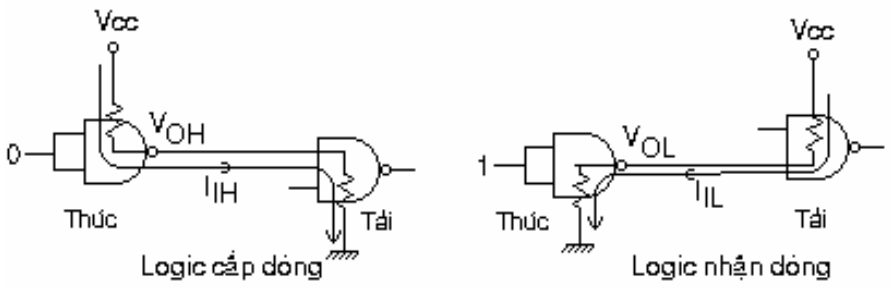
Khi tín hiệu ra ở mức cao đưa vào ngã vào, bất cứ tín hiệu nhiễu nào có giá trị âm và biên độ $>V_{NH}$ đều làm cho điện thế ngã vào rơi vào vùng bất định và mạch không nhận ra được tín hiệu thuộc mức logic nào. Tương tự cho trường hợp ngã ra ở mức thấp tín hiệu nhiễu có trị dương biên độ $>V_{NL}$ sẽ đưa mạch vào trạng thái bất định.

3.3.7 Logic cấp dòng và logic nhận dòng

Một mạch logic thường gồm nhiều tầng kết nối với nhau. Tầng cấp tín hiệu gọi là **tầng thúc** và tầng nhận tín hiệu gọi là **tầng tải**. Sự trao đổi dòng điện giữa hai tầng thúc và tải thể hiện bởi logic cấp dòng và logic nhận dòng.

(H 3.16a) cho thấy hoạt động gọi là cấp dòng: Khi ngã ra mạch logic 1 ở mức cao, nó cấp dòng I_{IH} cho ngã vào của mạch logic 2, vai trò như một tải nối mass. Ngã ra cổng 1 như là một nguồn dòng cấp cho ngã vào cổng 2

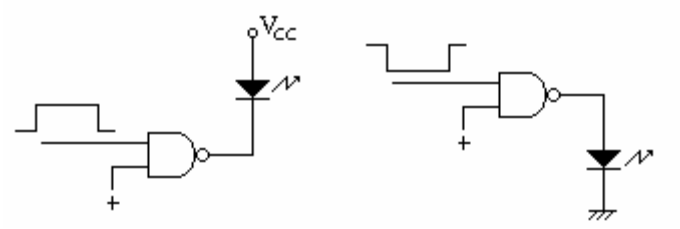
(H 3.16b) cho thấy hoạt động gọi là nhận dòng: Khi ngã ra mạch logic 1 ở mức thấp, nó nhận dòng I_{IL} từ ngã vào của mạch logic 2 xem như nối với nguồn V_{CC}.



(a) (b)

(H 3.16)

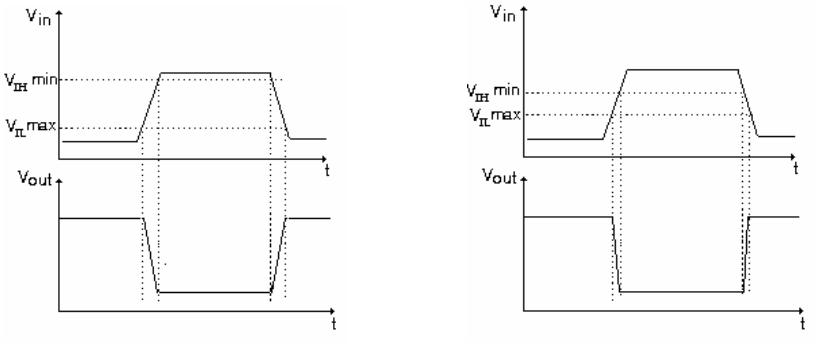
Thường dòng nhận của tầng thúc khi ở mức thấp có trị khá lớn so với dòng cấp của nó khi ở mức cao, nên người ta hay dùng trạng thái này khi cần gánh những tải tương đối nhỏ, ví dụ khi chỉ cần thúc cho một led, người ta có thể dùng mạch (H 3.17a) mà không thể dùng mạch (H 3.17b).



(a) (H 3.17) (b)

3.3.8 Tính Schmitt Trigger

Trong phần giới thiệu lề nhiễu, ta thấy còn một khoảng điện thế nằm giữa các ngưỡng logic, đây chính là khoảng điện thế ứng với transistor làm việc trong vùng tác động. Khoảng cách này xác định lề nhiễu và có tác dụng làm giảm độ rộng sườn xung (tức làm cho đường dốc lên và dốc xuống của tín hiệu ra dốc hơn) khi qua mạch. Lề nhiễu càng lớn khi vùng chuyển tiếp của ngã vào càng nhỏ, tín hiệu ra thay đổi trạng thái trong một khoảng thời gian càng nhỏ nên sườn xung càng dốc. Tuy nhiên vẫn còn một khoảng sườn xung nằm trong vùng chuyển tiếp nên tín hiệu ra không vuông hoàn toàn. (H 3.18a) và (H 3.18b) minh họa điều đó

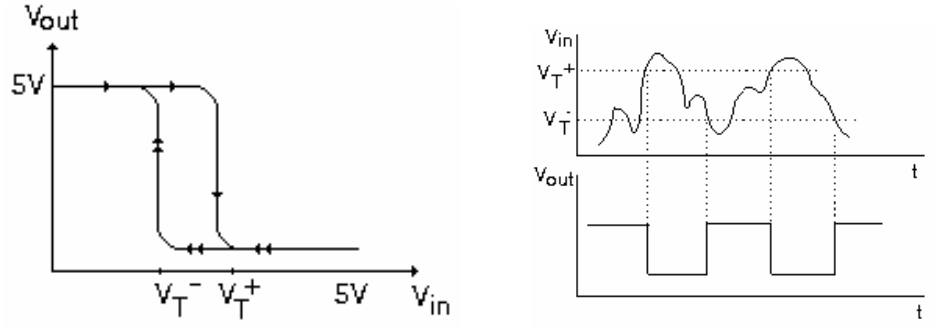


(a) (b)

(H 3.18)

Để cải thiện hơn nữa dạng tín hiệu ngã ra, bảo đảm tính miễn nhiễu cao, người ta chế tạo các cổng có tính trễ điện thế (H 3.19a), được gọi là cổng Schmitt Trigger

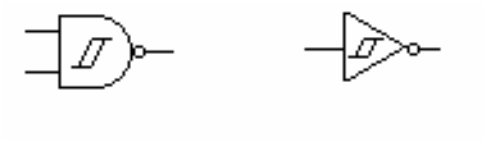
(H 3.19b) mô tả mối quan hệ giữa V_{out} và V_{in} của một cổng đảo Schmitt Trigger.



(a) (b)

(H 3.19)

(H 3.20a&b) là ký hiệu các cổng Schmitt Trigger.



(a) (b)

(H 3.20)

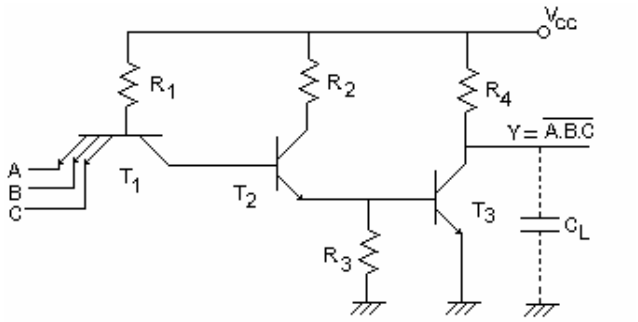
3.4 HỌ TTL

Trong quá trình phát triển của công nghệ chế tạo mạch số ta có các họ: RTL (Resistor-transistor logic), DCTL (Direct couple-transistor logic), RCTL (Resistor-Capacitor-transistor logic), DTL (Diod-transistor logic), ECL (Emitter- couple logic) v.v.... Đến bây giờ tồn tại hai họ có nhiều tính năng kỹ thuật cao như thời trễ truyền nhỏ, tiêu hao công suất ít, đó là họ TTL (transistor-transistor logic) dùng công nghệ chế tạo BJT và họ MOS (Công nghệ chế tạo MOS)

Dưới đây, lần lượt khảo sát các cổng logic của hai họ TTL và MOS

3.4.1 Cổng cơ bản họ TTL

Lấy cổng NAND 3 ngã vào làm thí dụ để thấy cấu tạo và vận hành của một cổng cơ bản



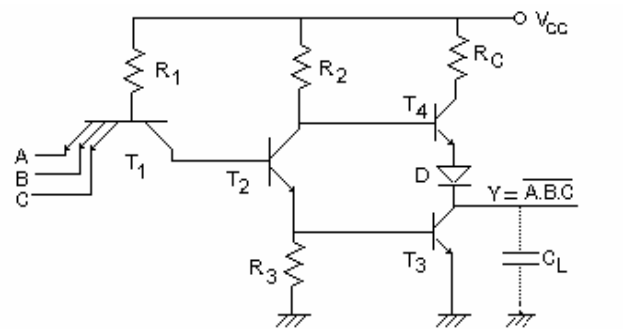
(H 3.21)

Khi một trong các ngã vào A, B, C xuống mức không T_1 dẫn đưa đến T_2 ngưng, T_3 ngưng, ngã ra Y lên cao; khi cả 3 ngã vào lên cao, T_1 ngưng, T_2 dẫn, T_3 dẫn, ngã ra Y xuống thấp. Đó chính là kết quả của cổng NAND.

Tụ C_L trong mạch chính là tụ ký sinh tạo bởi sự kết hợp giữa ngã ra của mạch (tầng thúc) với ngã vào của tầng tải, khi mạch hoạt động tụ sẽ nạp điện qua R_4 (lúc T_3 ngưng) và phóng qua T_3 khi transistor này dẫn do đó thời trễ truyền của mạch quyết định bởi R_4 và C_L, khi R_4 nhỏ mạch hoạt động nhanh nhưng công suất tiêu thụ lúc đó lớn, muốn giảm công suất phải tăng R_4 nhưng như vậy thời trễ truyền sẽ lớn hơn (mạch giao hoán chậm hơn). Để giải quyết khuyết điểm này đồng thời thỏa mãn một số yêu cầu khác , người ta đã chế tạo các cổng logic với các kiểu ngã ra khác nhau.

3.4.2 Các kiểu ngã ra

@ Ngã ra totempole

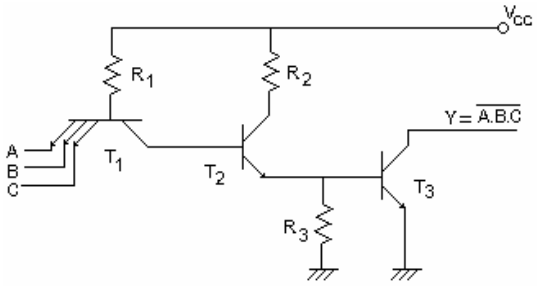


(H 3.22)

R_4 trong mạch cơ bản được thay thế bởi cụm T_4, R_C và Diod D, trong đó R_C có trị rất nhỏ, không đáng kể. T_2 bây giờ giữ vai trò mạch đảo pha: khi T_2 dẫn thì T_3 dẫn và T_4 ngưng, Y xuống thấp, khi T_2 ngưng thì T_3 ngưng và T_4 dẫn, ngã ra Y lên cao. Tụ C_L nạp điện qua T_4 khi T_4 dẫn và phóng qua T_3 (dẫn), thời hằng mạch rất nhỏ và kết quả là thời trễ truyền nhỏ. Ngoài ra do T_3 & T_4 luân phiên ngưng tương ứng với 2 trạng thái của ngã ra nên công suất tiêu thụ giảm đáng kể. Diod D có tác dụng nâng điện thế cực B của T_4 lên để bảo đảm khi T_3 dẫn thì T_4 ngưng**.**

Mạch này có khuyết điểm là không thể nối chung nhiều ngã ra của các cổng khác nhau vì có thể gây hư hỏng khi các trạng thái logic của các cổng này khác nhau.

@ Ngã ra cực thu để hở



(H 3.23)

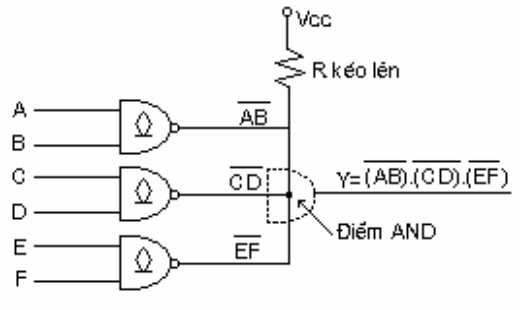
Ngã ra cực thu để hở có một số lợi điểm sau:

- Cho phép kết nối các ngã ra của nhiều cổng khác nhau, nhưng khi sử dụng phải mắc một điện trở từ ngã ra lên nguồn Vcc, gọi là **điện trở kéo lên**, trị số của điện trở này có thể được chọn lớn hay nhỏ tùy theo yêu cầu có lợi về mặt công suất hay tốc độ làm việc.

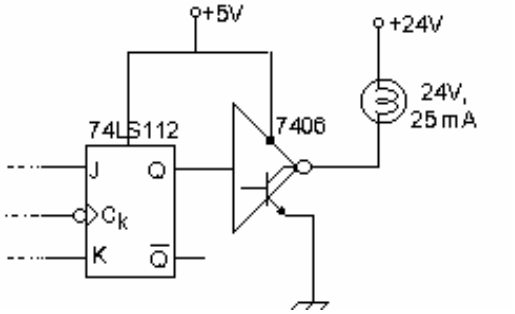
Điểm nối chung của các ngã ra có tác dụng như một cổng AND nên ta gọi là điểm AND (H 3.24)

- Người ta cũng chế tạo các IC ngã ra có cực thu để hở cho phép điện trở kéo lên mắc vào nguồn điện thế cao, dùng cho các tải đặc biệt hoặc dùng tạo sự giao tiếp giữa họ TTL với CMOS dùng nguồn cao.

Thí dụ IC 7406 là loại cổng đảo có ngã ra cực thu để hở có thể mắc lên nguồn 24 V (H 3.25)

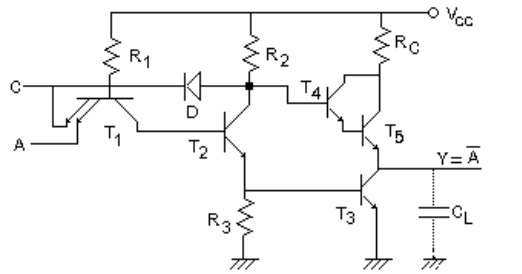


(H 3.24)

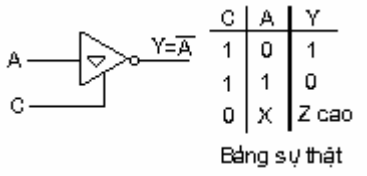


(H 3.25)

@ Ngã ra ba trạng thái



(H 3.26)



(H 3.27)

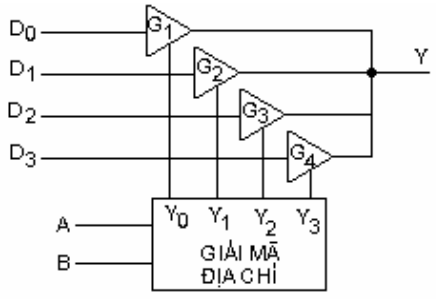
Mạch (H 3.26) là một cổng đảo có ngã ra 3 trạng thái, trong đó T_4 & T_5 được mắc Darlington để cấp dòng ra lớn cho tải. Diod D nối vào ngã vào C để điều khiển. Hoạt động của mạch giải thích như sau:

- Khi C=1, Diod D ngưng dẫn, mạch hoạt động như một cổng đảo
- Khi C=0, Diod D dẫn, cực thu T_2 bị ghim áp ở mức thấp nên T_3, T_4 & T_5 đều ngưng, ngã ra mạch ở trạng thái tổng trở cao.

Ký hiệu của cổng đảo ngã ra 3 trạng thái, có ngã điều khiển C tác động mức cao và bảng sự thật cho ở (H 3.27)

Cũng có các cổng đảo và cổng đệm 3 trạng thái với ngã điều khiển C tác động mức thấp mà SV có thể tự vẽ ký hiệu và bảng sự thật.

(H 3.28) là một ứng dụng của cổng đệm có ngã ra 3 trạng thái: Mạch chọn dữ liệu



(H 3.28)

Vận chuyển: Ứng với một giá trị địa chỉ AB , một ngã ra mạch giải mã địa chỉ được tác động (lên cao) cho phép một cổng mở và dữ liệu ở ngã vào cổng đó được truyền ra ngã ra. Thí dụ khi AB = 00, $Y_0 = 1$ ($Y_1=Y_2=Y_3=0$) G_1 mở, D_0 truyền qua G_1 đến ngã ra, trong lúc G_2, G_3, G_4 đóng, có ngã ra ở trạng thái Z cao, không ảnh hưởng đến hoạt động của mạch.

3.4.3 Đặc tính các loạt TTL

Các IC số họ TTL được sản xuất lần đầu tiên vào năm 1964 bởi hãng Texas Instrument Corporation của Mỹ, lấy số hiệu là 74XXXX & 54XXXX. Sự khác biệt giữa 2 họ 74XXXX và 54 XXXX chỉ ở hai điểm:

74: $V_{CC}=5 \pm 0,5$ V và khoảng nhiệt độ hoạt động từ 0^o C đến 70^o C

54: $V_{CC}=5 \pm 0,25$ V và khoảng nhiệt độ hoạt động từ -55^o C đến 125^o C

Các tính chất khác hoàn toàn giống nhau nếu chúng có cùng số.

Trước số 74 thường có thêm ký hiệu để chỉ hãng sản xuất. Thí dụ SN của hãng Texas, DM của National Semiconductor, S của Signetics

Ngoài ra trong quá trình phát triển, các thông số kỹ thuật (nhất là tích số công suất vận tốc) luôn được cải tiến và ta có các loạt khác nhau: 74 chuẩn, 74L (Low power), 74 H (High speed), 74S (Schottky), 74LS (Low power Schottky), 74AS (Advance Schottky), 74ALS (Advance Low power Schottky), 74F (Fast, Fair Child).

Bảng 3.1 cho thấy một số tính chất của các loạt kể trên:

Thông số kỹ thuật	**74**	**74L**	**74H**	**74S**	**74LS**	**74AS**	**74ALS**	**74F**
Thời trễ truyền (ns)	9	33	6	3	9,5	1,7	4	3
Công suất tiêu tán (mW)	10	1	23	20	2	8	1,2	6
Tích số công suất vận tốc (pJ)	90	33	138	60	19	13,6	4,8	18
Tần số xung C_K max (MHz)	35	3	50	125	45	200	70	100
Fan Out (cùng loạt)	10	20	10	20	20	40	20	33
Điện thế								
V_{OH}(min)	2,4	2,4	2,4	2,7	2,7	2,5	2,5	2,5
V_{OL} (max)	0,4	0,4	0,4	0,5	0,5	0,5	0,4	0,5
V_{IH} (min)	2,0	2,0	2,0	2,0	2,0	2,0	2,0	2,0
V_{IL} (max)	0,8	0,7	0,8	0,8	0,8	0,8	0,8	0,8

Bảng 3.1

- Loạt 74S: Các transistor trong mạch được mắc thêm một Diod Schottky giữa hai cực CB với mục đích giảm thời gian chuyển trạng thái của transistor do đó làm giảm thời trễ truyền.

- Loạt 74AS và 74ALS là cải tiến của 74S để làm giảm hơn nữa giá trị tích số Công suất - Vận tốc.

- Loạt 74F: Dùng kỹ thuật đặc biệt làm giảm diện dung ký sinh do đó cải thiện thời trễ truyền của cổng.

3.5 HO MOS

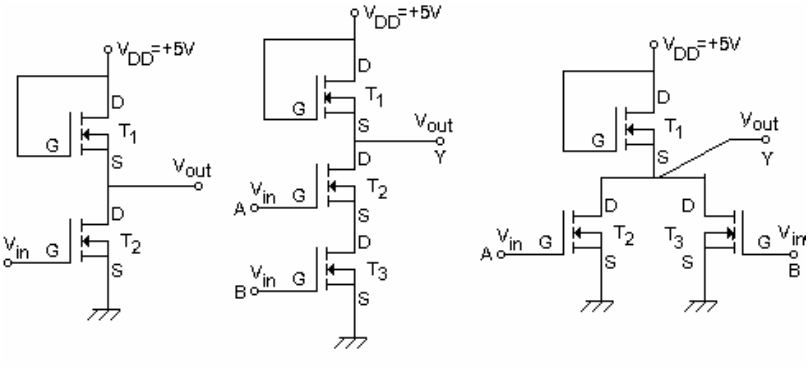
Gồm các IC số dùng công nghệ chế tạo của transistor MOSFET loại tăng, kênh N và kênh P . Với transistor kênh N ta có NMOS, transistor kênh P ta có PMOS và nếu dùng cả hai loại transistor kênh P & N ta có CMOS. Tính năng kỹ thuật của loại NMOS và PMOS có thể

nói là giống nhau, trừ nguồn cấp điện có chiều ngược với nhau do đó ta chỉ xét loại NMOS và CMOS.

Các transistor MOS dùng trong IC số cũng chỉ hoạt động ở một trong 2 trạng thái: dẫn hoặc ngưng.

- Khi dẫn, tùy theo nồng độ pha của chất bán dẫn mà transistor có nội trở rất nhỏ (từ vài chục Ω đến hàng trăm KΩ) tương đương với một khóa đóng.
- Khi ngưng, transistor có nội trở rất lớn (hàng $10^{10}\Omega$), tương đương với một khóa hở.

3.5.1 Cổng cơ bản NMOS



(a) (b) (c)

(H 3.29)

(H 3.29a), (H 3.29b) và (H3.29c) là các cổng NOT, NAND và NOR dùng NMOS Bảng 3.2 cho thấy quan hệ giữa các điện thế của các ngã vào , ra cổng NOT

V_{in}	T_1	T_2	V_{out}
0V (logic 0)	R_{ON} = 100KΩ	$R_{OFF}=10^{10}\Omega$	+5V (logic 1)
+5V (logic1)	R_{ON} = 100KΩ	R_{ON} = 1KΩ	0,05V (logic 0)

Bảng 3.2

Ngoài ra vận hành của cổng NAND và NOR được giải thích như sau:

❖ Cổng NAND:

- Khi 2 ngã vào nối lên mức cao, T_2 và T_3 dẫn, ngã ra xuống thấp.
- Khi có 1 ngã vào nối xuống mức thấp, một trong 2 transistor T_2 hoặc T_3 ngưng, ngã ra lên cao.

Đó chính là kết quả của cổng NAND 2 ngã vào.

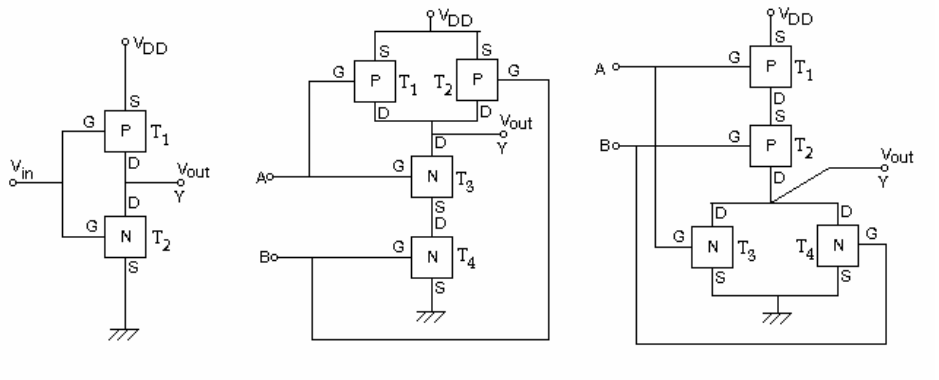
❖ Cổng NOR:

- Khi 2 ngã vào nối xuống mức thấp, T_2 và T_3 ngưng, ngã ra lên cao.
- Khi có 1 ngã vào nối lên mức cao, một trong 2 transistor T_2 hoặc T_3 dẫn, ngã ra xuống thấp.

Đó chính là kết quả của cổng NOR 2 ngã vào.

3.5.2 Cổng cơ bản CMOS

Họ CMOS sử dụng hai loại transistor kênh N và P với mục đích cải thiện tích số công suất vận tốc, mặc dù khả năng tích hợp thấp hơn loại N và P. (H 3.30a), (H 3.30b) và (H 3.30c) là các cổng NOT, NAND và NOR họ CMOS



(a) (b) (c)

(H 3.30)

Bảng 3.3 cho thấy quan hệ điện thế của các ngã vào , ra cổng NOT

V_{in}	T_1	T_2	V_{out}
V_{DD} (logic1)	$R_{OFF}=10^{10}\Omega$	R_{ON} = 1KΩ	0V (logic 0)
0V (logic0)	R_{ON} = 1KΩ	$R_{OFF}=10^{10}\Omega$	V_{DD} (logic 1)

Bảng 3.3

Ngoài ra vận hành của cổng NAND và NOR được giải thích như sau:

❖ Cổng NAND:

- Khi 2 ngã vào nối lên mức cao, T_1 và T_2 ngưng, T_3 và T_4 dẫn, ngã ra xuống thấp.
- Khi có 1 ngã vào nối xuống mức thấp, một trong 2 transistor T_3 hoặc T_4 ngưng, một trong 2 transistor T_1 hoặc T_2 dẫn, ngã ra lên cao.

Đó chính là kết quả của cổng NAND 2 ngã vào.

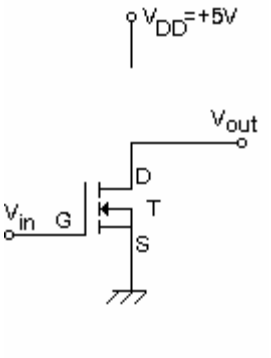
❖ Cổng NOR:

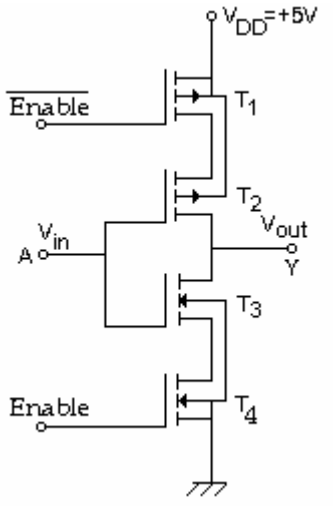
- Khi 2 ngã vào nối xuống mức thấp, T_1và T_2 dẫn, T_3 và T_4 ngưng, ngã ra lên cao.
- Khi có 1 ngã vào nối lên mức cao, một trong 2 transistor T_3 hoặc T_4 dẫn, một trong 2 transistor T_1 hoặc T_2 ngưng, ngã ra xuống thấp.

Đó chính là kết quả của cổng NOR 2 ngã vào.

3.5.3 Các cổng CMOS khác

Người ta cũng sản xuất các cổng CMOS với cực Drain để hở và ngã ra 3 trạng thái để sử dụng trong các trường hợp đặc biệt như họ TTL





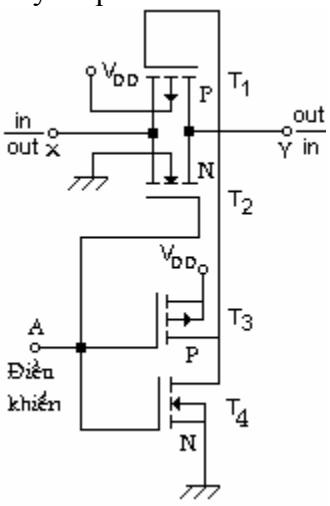
(a) **(H 3.31)** **(b)**

(H 3.31a) là một cổng NOT có cực D để hở, khi sử dụng phải có điện trở kéo lên

(H 3.31b) là một cổng NOT có ngã ra 3 trạng thái:

- Khi ngã vào Enable =1, T_1 và T_4 dẫn, mạch hoạt động như là cổng đảo,
- Khi ngã vào Enable =0, T_1 và T_4 đều ngưng đưa mạch vào trạng thái Z cao.

Ngoài ra lợi dụng tính chất của transistor MOS có nội trở rất nhỏ khi dẫn, người ta cũng chế tạo các mạch có khả năng truyền tín hiệu theo 2 chiều, gọi là khóa 2 chiều. (H 3.32) là một khóa 2 chiều với A là ngã vào điều khiển. Khi A = 0 khóa hở, khi A = 1, khóa đóng cho tín hiệu truyền qua theo 2 chiều



A	X to Y	Y to X
0	OFF	OFF
1	ON	ON

(H 3.32)

Vận hành: T_3 và T_4 vai trò là một cổng đảo

- Khi A = 0, cực G của T_2 ở mức thấp nên T_2 (kênh N) ngưng, cực G của T_1 (kênh P) ở mức cao nên T_1 ngưng, mạch tương đương với khóa hở.
- Khi A =1, cực G của T_2 ở mức cao nên T_2 dẫn, cực G của T_1 ở mức thấp nên T_1 dẫn, mạch tương đương với khóa đóng. Tín hiệu truyền qua một chiều nhờ T_1 (loại P) và theo chiều ngược lại nhờ T_2 (loại N)

Biên độ của tín hiệu V_i truyền qua khóa phải thỏa điều kiện $0 < V_i < V_{DD}$.

Như vậy nếu ta sử dụng nguồn $\pm V_{DD}$ thì khóa cho tín hiệu xoay chiều đi qua.

3.5.3 Đặc tính của họ MOS

Một số tính chất chung của các cổng logic họ MOS (NMOS, PMOS và CMOS) có thể kể ra như sau:

- Nguồn cấp điện : V_{DD} từ 3V đến 15V
- Mức logic: V_{OL} (max) = 0V V_{OH} (min) = V_{DD}
  V_{IL} (max) = 30% V_{DD} V_{IH} (min) = 70%V_{DD}
- Lề nhiễu : V_{NH} = 30%V_{DD} V_{NL} = 30%V_{DD}

Với nguồn 5V, lề nhiễu khỏang 1,5V, rất lớn so với họ TTL.

- Thời trễ truyền tương đối lớn, khỏang vài chục ns, do điện dung ký sinh ở ngã vào và tổng trở ra của transistor khá lớn.
- Công suất tiêu tán tương đối nhỏ, hàng nW, do dòng qua transistor MOS rất nhỏ.
- Số Fan Out: 50 UL

Do tổng trở vào của transistor MOS rất lớn nên dòng tải cho các cổng họ MOS rất nhỏ, do đó số Fan Out của họ MOS rất lớn, tuy nhiên khi mắc nhiều tầng tải vào một tầng thúc thì điện dung ký sinh tăng lên (gồm nhiều tụ mắc song song) ảnh hưởng đến thời gian giao hoán của mạch nên khi dùng ở tần số cao người ta giới hạn số Fan Out là 50, nghĩa là một cổng MOS có thể cấp dòng cho 50 cổng tải cùng loạt.

- Như đã nói ở trên, CMOS có cải thiện thời trễ truyền so với loại NMOS và PMOS, tuy nhiên mật độ tích hợp của CMOS thì nhỏ hơn hai loại này. Dù sao so với họ TTL thì mật độ tích hợp của họ MOS nói chung lớn hơn rất nhiều, do đó họ MOS rất thích hợp để chế tạo dưới dạng LSI và VLSI.

3.5.4 Các loạt CMOS

CMOS có hai ký hiệu: 4XXX do hảng RCA chế tạo và 14XXX của hảng MOTOROLA, có hai loạt 4XXXA (14XXXA) và 4XXXB (14XXXB), loạt B ra đời sau có cải thiện dòng ra.

Ngoài ra còn có các loạt :

- 74C : CMOS có cùng sơ đồ chân và chức năng với IC TTL nếu có cùng số. Thí dụ IC 74C74 là IC gồm 2 FF D tác động bởi cạnh xung đồng hồ giống như IC 7474 của TTL. Hầu hết (nhưng không tất cả) các thông số của loạt 74C giống với 74 TTL nên ta có thể thay thế 2 loại này cho nhau được.
- 74HC (High speed CMOS), 74HCT: Đây là loạt cải tiến của 74C, tốc độ giao hoán có thể so sánh với 74LS, riêng 74HCT thì hoàn toàn tương thích với TTL kể cả các mức logic. Đây là loạt IC CMOS được dùng rộng rãi.
- 74AC và 74ACT (Advance CMOS) cải tiến của 74 HC và HCT về mặt nhiễu bằng cách sắp xếp lại thứ tự các chân, do đó nó không tương thích với TTL về sơ đồ chân.

3.6 GIAO TIẾP GIỮA CÁC HỌ IC SỐ

Giao tiếp là thực hiện việc kết nối ngã ra của một mạch hay hệ thống với ngã vào của mạch hay hệ thống khác. Do tính chất về điện khác nhau giữa hai họ TTL và CMOS nên việc giao tiếp giữa chúng trong nhiều trường hợp không thể nối trực tiếp được mà phải nhờ một mạch trung gian nối giữa tầng thúc và tầng tải sao cho điện thế tín hiệu ra ở tầng thúc phù hợp với tín hiệu vào của tầng tải và dòng điện tầng thúc phải đủ cấp cho tầng tải.

CMOS (V_{DD}= TTL

	5V)						
Thông số	**4000B**	**74HC**	**74HCT**	**74**	**74LS**	**74AS**	**74ALS**
V_{IH}(min)	3,5V	3,5V	2,0V	2,0V	2,0V	2,0V	2,0V
V_{IL}(max)	1,5V	1,0V	0,8V	0,8V	0,8V	0,8V	0,8V
V_{OH}(min)	4,95V	4,9V	4,9V	2,4V	2,7V	2,7V	2,7V
V_{OL}(max)	0,05V	0,1V	0,1V	0,4V	0,5V	0,5V	0,4V
I_{IH}(max)	1µA	1µA	1µA	40µA	20µA	200µA	20µA
I_{IL}(max)	1µA	1µA	1µA	1,6 mA	0,4 mA	2 mA	100µA
I_{OH}(max)	0,4 mA	4 mA	4 mA	0,4 mA	0,4 mA	2 mA	0,4 mA
I_{OL}(max)	0,4 mA	4 mA	4 mA	16 mA	8 mA	20 mA	8 mA

Bảng 3.4

Có thể nói điều kiện để thúc trực tiếp

- Khi dòng điện ra của tầng thúc lớn hơn hoặc bằng dòng điện vào của tầng tải ở cả hai trạng thái thấp và cao.
- Khi hiệu thế ngã ra của tầng thúc ở hai trạng thái thấp và cao phù hợp với điện thế vào của tầng tải.

Như vậy, trước khi xét các trường hợp cụ thể ta xem qua bảng kê các thông số của hai họ IC

3.6.1 TTL thúc CMOS

- TTL thúc CMOS dùng điện thế thấp (V_{DD} = 5V):

Từ bảng 3.4 dòng điện vào của CMOS có trị rất nhỏ so với dòng ra của các loạt TTL, vậy về dòng điện không có vấn đề

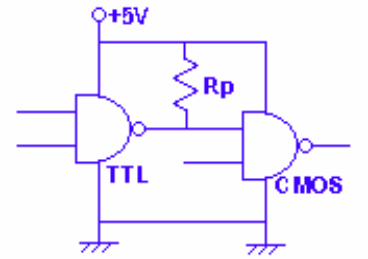
Tuy nhiên khi so sánh hiệu thế ra của TTL với hiệu thế vào của CMOS ta thấy V_{OH}(max) của tất cả các loạt TTL đều khá thấp so với V_{IH}(min) của TTL, như vậy phải có biện pháp nâng hiệu thế ra của TTL lên. Điều này thực hiện được bằng một điện trở kéo lên mắc ở ngã ra của IC TTL (H 3.33)

- TTL thúc 74 HCT:

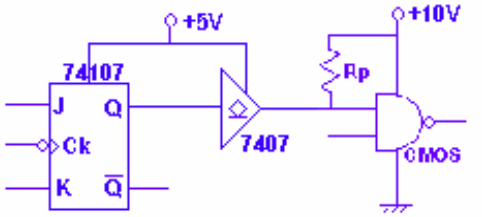
Như đã nói trước đây, riêng loạt 74HCT là loạt CMOS được thiết kế tương thích với TTL nên có thể thực hiện kết nối mà không cần điện trở kéo lên.

- TTL thúc CMOS dùng nguồn cao (V_{DD} = +10V)

Ngay cả khi dùng điện trở kéo lên, điện thế ngã ra mức cao của TTL vẫn không đủ cấp cho ngã vào CMOS, người ta phải dùng một cổng đệm có ngã ra để hở có thể dùng nguồn cao (Thí dụ IC 7407) để thực hiện sự giao tiếp (H 3.34)



(H 3.33)



(H 3.34)

3.6.2 CMOS thúc TTL

- CMOS thúc TTL ở trạng thái cao:

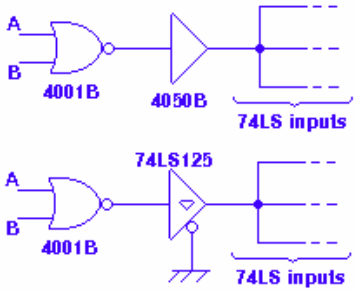
Bảng 3.4 cho thấy điện thế ra và dòng điện ra mức cao của CMOS đủ để cấp cho TTL . Vậy không có vấn đề ở trạng thái cao

- CMOS thúc TTL ở trạng thái thấp:

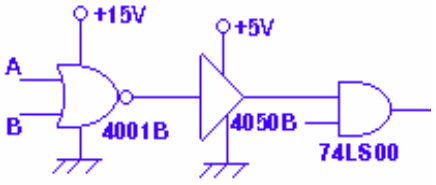
Dòng điện vào ở trạng thái thấp của TTL thay đổi trong khoảng từ 100 µA đến 2 mA. Hai loạt 74HC và 74HCT có thể nhận dòng 4 mA . Vậy hai loạt này có thể giao tiếp với một IC TTL mà không có vấn đề. Tuy nhiên, với loạt 4000B, I_{OL} rất nhỏ không đủ để giao tiếp với ngay cả một IC TTL, người ta phải dùng một cổng đệm để nâng dòng tải của loạt 4000B trước khi thúc vài IC 74LS (H 3.35)

- CMOS dùng nguồn cao thúc TTL:

Có một số IC loạt 74LS được chế tạo đặc biệt có thể nhận điện thế ngã vào cao khoảng 15V có thể được thúc trực tiếp bởi CMOS dùng nguồn cao, tuy nhiên đa số IC TTL không có tính chất này, vậy để có thể giao tiếp với CMOS dùng nguồn cao, người ta phải dùng cổng đệm để hạ điện thế ra xuống cho phù hợp với IC TTL (H 3.36)



(H 3.35)



(H 3.36)

Vài thí dụ dùng cổng thiết kế mạch

1. Dùng cổng NAND 2 ngã vào thiết kế mạch tạo hàm Y = f(A,B,C) =1 khi thỏa các điều kiện sau:

a. A=0, B=1 và C=1

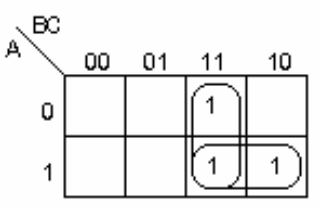
b. A=1, B=1 bất chấp C

Giải

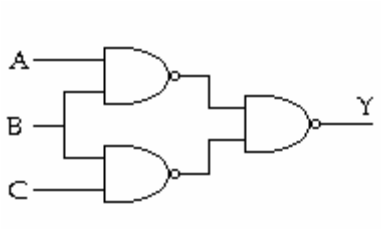
Dự vào điều kiện của bài toán ta có bảng sự thật của hàm Y

A		C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Rút gọn hàm:



Y =AB+BC



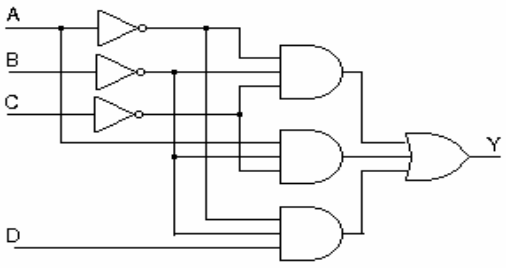
(H 3.37)

Để dùng tòan cổng NAND tạo hàm, ta dùng định lý De Morgan, biến đổi hàm Y:

$$Y = \overline{\overline{Y}} = \overline{\overline{AB + BC}} = \overline{\overline{AB}.\overline{BC}}$$

Và mạch có dạng (H 3.37)

2. Cho mạch



(H P3.38)

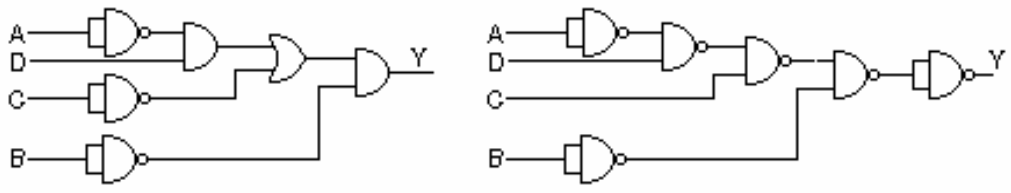
a./ Viết biểu thức hàm Y theo các biến A,B,C.
b./ Rút gọn hàm logic này
c./ Thay thế mạch trên bằng một mạch chỉ gồm cổng NAND 2 ngã vào

Giải

a./ Ta có $Y = \overline{A}.\overline{B}.\overline{C} + A\overline{B}.\overline{C} + \overline{A}.\overline{B}D$
b./ Rút gọn

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

c./ Vẽ mạch thay thế dùng cổng NAND 2 ngã vào
Trước nhất ta vẽ mạch tương ứng hàm rút gọn, sau đó dùng biến đổi cổng



(H P3.39)

❋❋❋

BÀI TẬP

1. Thiết kế mạch thực hiện các hàm sau đây dùng toàn cổng NAND 2 ngã vào:

a./ f(A,B,C) = 1 nếu $(ABC)_2$ là số chẵn.
b./ f(A,B,C) = 1 nếu có ít nhất 2 biến = 1.
c./ f(A,B,C) = 1 nếu số nhị phân $(ABC)_2 > 5$.
d./ f(A,B,C) = 1 nếu số biến có giá trị 1 là số chẵn.
e./ f(A,B,C) = 1 nếu có một và chỉ một biến = 1.

2. Thiết kế mạch gồm 2 ngã vào D, E và 2 ngã ra P, C thỏa các điều kiện sau đây:

- Nếu E = 1 D = 0 $\Rightarrow$ P = 1, C = 0
- Nếu E = 1 D = 1 $\Rightarrow$ P = 0, C = 1
- Nếu E = 0 D bất kỳ $\Rightarrow$ P = 1, C = 1

3. Hàm logic F(A, B, C) thỏa tính chất sau đây :

F(A,B,C) = 1 nếu có một và chỉ một biến bằng 1
a- Lập bảng sự thật cho hàm F.
b- Vẽ mạch logic tạo hàm F.

4. Thiết Kế mạch tạo hàm $Y = \overline{A}.\overline{B}.\overline{C} + A\overline{B}.\overline{C} + \overline{A}.\overline{B}C$ bằng các cổng NAND 2 ngã vào

5. Hàm F(A,B,C) xác đinh bởi bảng sự thật

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

a- Dùng bản đồ Karnaugh rút gọn hàm F.
b- Vẽ sơ đồ mạch logic thực hiện hàm F.
c- Vẽ lại mạch chỉ dùng cổng NOR hai ngã vào.

6. Rút gọn hàm logic :

f(A,B,C,D) = Σ(0,1, 2, 4, 5, 8), A = MSB. Hàm không xác định với các tổ hợp biến (3, 7,10).

Dùng số cổng NOR ít nhất để thực hiện mạch tạo hàm trên.

7. Hàm f(A,B,C) =1 khi số biến = 1 là số chẵn

- Viết biểu thức logic của hàm f(A,B,C) theo tổ hợp biến A,B,C.
- Dùng các cổng EX-OR để thực hiện mạch tạo hàm trên.

8. Một mạch tổ hợp nhận vào một số nhị phân $A=A_3A_2A_1A_0$ (A_0 là LSB) tạo ra ở ngã ra Y ở mức cao khi và chỉ khi 0010<A<1000. Hãy thiết kế mạch với:

a) Cấu trúc NAND-NAND.
b) Toàn cổng NAND 2 ngã vào.

9. Một mạch tổ hợp nhận vào một số BCD, có tên là X. Ngã ra của mạch lên 1 khi thỏa điều kiện $1_{10} \leq X \leq 5_{10}$.
Hãy thiết kế mạch tổ hợp trên, dùng toàn cổng NAND 2 ngã vào.

10. Hàm f(A,B,C,D) =1 khi có ít nhất 3 biến = 1
- Viết biểu thức logic của hàm f(A,B,C,D) theo tổ hợp biến A,B,C,D.
- Dùng các cổng NAND 2 ngã vào (số cổng ít nhất) để thực hiện mạch tạo hàm trên.

🕮 CHƯƠNG 4: MẠCH TỔ HỢP

❋ MẠCH MÃ HÓA
❄ Mạch mã hóa 2^n đường sang n đường
❄ Mạch tạo mã BCD cho số thập phân
❋ MẠCH GIẢI MÃ
❄ Mạch giải mã n đường sang 2^n đường
❄ Mạch giải mã BCD sang 7 đoạn
❋ MẠCH ĐA HỢP VÀ GIẢI ĐA HỢP
❄ Khái niệm
❄ Mạch đa hợp
❄ Ứng dụng của mạch đa hợp
❄ Mạch giải đa hợp
❋ MẠCH SO SÁNH
❄ Mạch so sánh hai số một bit
❄ Mạch so sánh hai số nhiều bit
❋ MẠCH KIỂM / PHÁT CHẴN LẺ
❄ Mạch phát chẵn lẻ
❄ Mạch kiểm chẵn lẻ

Các mạch số được chia ra làm hai loại: Mạch tổ hợp và Mạch tuần tự.

- Mạch tổ hợp: Trạng thái ngã ra chỉ phụ thuộc vào tổ hợp các ngã vào khi tổ hợp này đã ổn định. Ngã ra Q của mạch tổ hợp là hàm logic của các biến ngã vào A, B, C . . ..

$$Q = f(A,B,C\ldots)$$

- Mạch tuần tự : Trạng thái ngã ra không những phụ thuộc vào tổ hợp các ngã vào mà còn phụ thuộc trạng thái ngã ra trước đó. Ta nói mạch tuần tự có tính nhớ. Ngã ra Q_+ của mạch tuần tự là hàm logic của các biến ngã vào A, B, C . . . . và ngã ra Q trước đó.

$$Q_+ = f(Q,A,B,C\ldots)$$

Chương này nghiên cứu một số mạch tổ hợp thông dụng thông qua việc thiết kế một số mạch đơn giản và khảo sát một số IC trên thực tế.

4.1. MẠCH MÃ HÓA

Mã hóa là gán các ký hiệu cho các đối tượng trong một tập hợp để thuận tiện cho việc thực hiện một yêu cầu cụ thể nào đó. Thí dụ mã BCD gán số nhị phân 4 bit cho từng số mã của số thập phân (từ 0 đến 9) để thuận tiện cho máy đọc một số có nhiều số mã; mã Gray dùng tiện lợi trong việc tối giản các hàm logic . . .. Mạch chuyển từ mã này sang mã khác gọi là mạch chuyển mã, cũng được xếp vào loại mạch mã hóa. Thí dụ mạch chuyển số nhị phân 4 bit sang số Gray là một mạch chuyển mã.

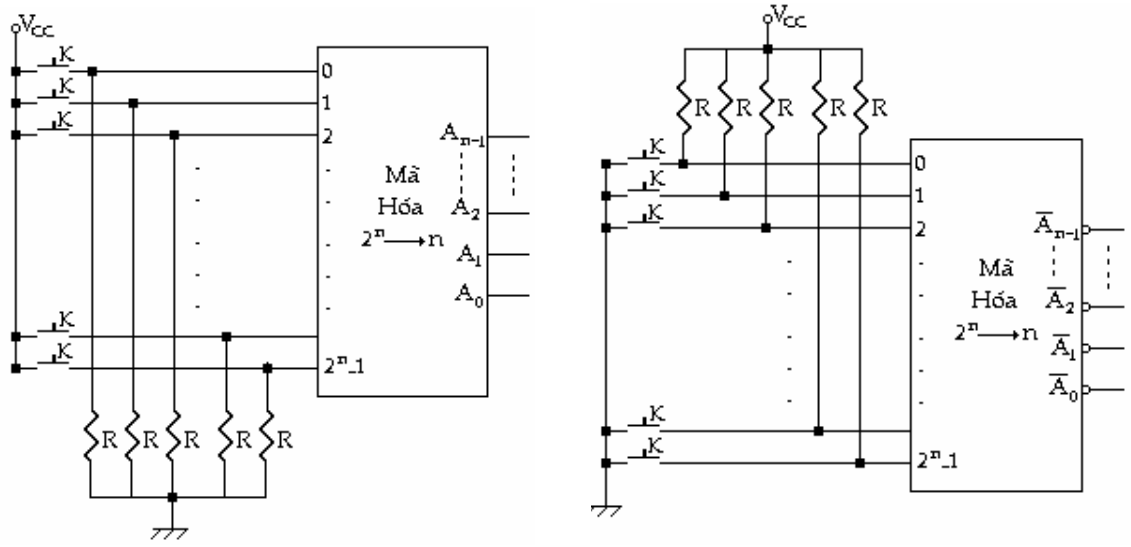
4.1.1 Mạch mã hóa 2^n đường sang n đường

Một số nhị phân n bit cho 2^n tổ hợp số khác nhau. Vậy ta có thể dùng số n bit để mã cho 2^n ngã vào khác nhau, khi có một ngã vào được chọn bằng cách đưa nó lên mức tác động, ở ngã ra sẽ chỉ báo số nhị phân tương ứng. Đó là mạch mã hóa 2^n đường sang n đường.

(H 4.1) là mô hình một mạch mã hóa 2^n đường sang n đường.

- (H 4.1a) là mạch có ngã vào và ra tác động cao : Khi các ngã vào đều ở mức thấp, mạch chưa hoạt động, các ngã ra đều ở mức thấp. Khi có một ngã vào được tác động bằng cách ấn khóa K tương ứng để đưa ngã vào đó lên mức cao, các ngã ra sẽ cho số nhị phân tương ứng.
- (H 4.1b) là mạch có ngã vào và ra tác động thấp. Hoạt động tương tự như mạch trên nhưng có mức tác động ngược lại. (trong mô hình (H 4.1b) ký hiệu dấu o ở ngã ra để chỉ mức tác động thấp, còn ở ngã vào không có dấu o vì là mạch thật)

Trong trường hợp ngã ra có mức tác động thấp, muốn đọc đúng số nhị phân ở ngã ra, ta phải đảo các bit để đọc.



(a) (b)

(H 4.1)

Dĩ nhiên, người ta cũng có thể thiết kế theo kiểu ngã vào tác động thấp và ngã ra tác động cao hay ngược lại. Trên thực tế, ta có thể có bất cứ loại ngã vào hay ra tác động theo bất cứ kiểu nào (mức cao hay thấp).

Ngoài ra, để tránh trường hợp mạch cho ra một mã sai khi người sử dụng vô tình (hay cố ý) tác động đồng thời vào hai hay nhiều ngã vào, người ta thiết kế các mạch mã hóa ưu tiên: là mạch chỉ cho ra một mã duy nhất có tính ưu tiên khi có nhiều ngã vào cùng được tác động.

4.1.1.1 Mã hóa ưu tiên 4 đường sang 2 đường

Thiết kế mạch mã hóa 4 đường sang 2 đường, ưu tiên cho mã có trị cao, ngã vào và ra tác động cao

Bảng sự thật và sơ đồ mạch (H 4.2)

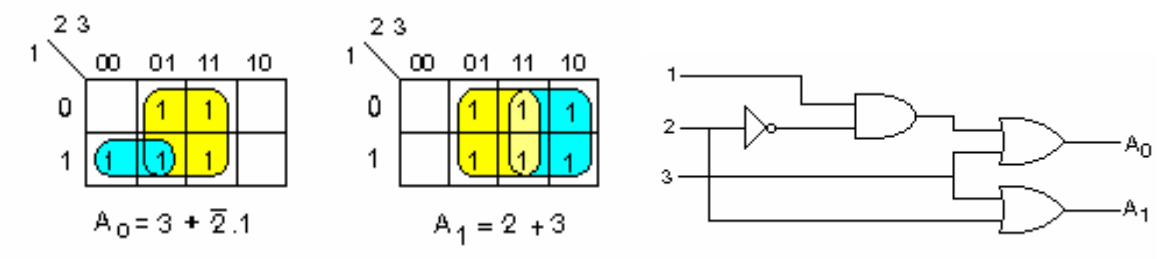
0	1	2	3	A_1	A_0
1	0	0	0	0	0

x	1	0	0	0	1
x	x	1	0	1	0
x	x	x	1	1	1

Bảng 4.1

Nhận thấy biến 0 trong bảng sự thật không ảnh hưởng đến kết quả nên ta chỉ vẽ bảng Karnaugh cho 3 biến 1, 2 và 3. Lưu ý là do trong bảng sự thật có các trường hợp bất chấp của biến nên ứng với một trị riêng của hàm ta có thể có đến 2 hoặc 4 số 1 trong bảng Karnaugh. Thí dụ với trị 1 của cả 2 hàm A_1 và A_0 ở dòng cuối cùng đưa đến 4 số 1 trong các ô 001, 011, 101 và 111 của 3 biến 123.

Từ bảng Karnaugh, ta có kết quả và mạch tương ứng. Trong mạch không có ngã vào 0, điều này được hiểu là mạch sẽ chỉ báo số 0 khi không tác động vào ngã vào nào.



(H 4.2)

4.1.1.2 Mã hóa 8 đường sang 3 đường

Chúng ta sẽ khảo sát một IC mã hóa 8 đường sang 3 đường.

Trên thực tế khi chế tạo một IC, ngoài các ngã vào/ra để thực hiện chức năng chính của nó, người ta thường dự trù thêm các ngã vào và ra cho một số chức năng khác như cho phép, nối mạch để mở rộng hoạt động của IC.

IC 74148 là IC mã hóa ưu tiên 8 đường sang 3 đường, vào/ ra tác động thấp, có các ngã nối mạch để mở rộng mã hóa với số ngã vào nhiều hơn.

Dưới đây là bảng sự thật của IC 74148, trong đó E_i ngã vào nối mạch và cho phép, E_o là ngã ra nối mạch và G_s dùng để mở rộng cho số nhị phân ra.

Dựa vào bảng sự thật, ta thấy IC làm việc theo 10 trạng thái:

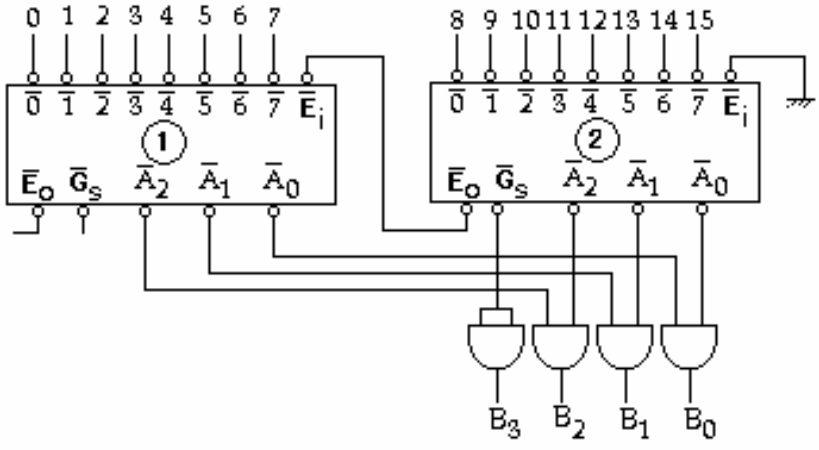
- Các trạng thái từ 0 đến 7: IC mã hóa cho ra số 3 bit
- Các trạng thái 8 và 9: dùng cho việc mở rộng, sẽ giải thích rõ hơn khi nối 2 IC để mở rộng mã hóa cho số 4 bit

Trạng thái	Ngã vào								Ngã ra				
	E_i	0	1	2	3	4	5	6	A_2	A_1	A_0	G_s	E_o
		7											
9	1	x	x	x	x	x	x	x	1	1	1	1	1
8	0	x							1	1	1	1	0
7	0	1	1	1	1	1	1	1	0	0	0	0	1
6	0	1							0	0	1	0	1
5	0	x	x	x	x	x	x	x	0	1	0	0	1
4	0	0							0	1	1	0	1

3	0	x	x	x	x	x	x	0	1	0	0	0	1
2	0	1							1	0	1	0	1
1	0	x	x	x	x	x	0	1	1	1	0	0	1
0	0	1							1	1	1	0	1
		x	x	x	x	0	1	1					
		1											
		x	x	x	0	1	1	1					
		1											
		x	x	0	1	1	1	1					
		1											
		x	0	1	1	1	1	1					
		1											
		0	1	1	1	1	1	1					
		1											

Bảng 4.2

(H 4.3) là cách nối 2 IC để thực hiện mã hóa 16 đường sang 4 đường



(H 4.3)

- IC2 có $E_i = 0$ nên hoạt động theo các trạng thái từ 0 đến 8, nghĩa là mã hóa từ 0 đến 7 cho các ngã ra $A_2A_1A_0$.

- IC1 có E_i nối với E_o của IC2 nên IC1 chỉ hoạt động khi tất cả ngã vào dữ liệu của IC2 lên mức 1 (IC2 hoạt động ở trạng thái 8)

* Để mã hóa các số từ 0 đến 7, cho các ngã vào 8 đến 15 (tức các ngã vào dữ liệu của IC2) lên mức 1, IC2 hoạt động ở trạng thái 8.

Lúc đó $E_{i1} = E_{o2} = 0$: kết quả là IC1 sẽ hoạt động ở trạng thái từ 0 đến 7, cho phép tạo mã các số từ 0 đến 7 (từ 111 đến 000) và IC2 hoạt động ở trạng thái 8 nên các ngã ra $(A_2A_1A_0)_2$= 111, đây là điều kiện mở các cổng AND để cho mã số ra là $B_2B_1B_0 = A_2A_1A_0$ của IC1, trong lúc đó $B_3 = G_{s2} = 1$, ta được kết quả từ 1111 đến 1000, tức từ 0 đến 7 (tác động thấp).

Thí dụ để mã số 4 , đưa ngã vào 4 xuống mức 0, các ngã vào từ 5 đến 15 lên mức 1, bất chấp các ngã vào từ 0 đến 3, mã số ra là $B_3B_2B_1B_0=G_{s2}B_2B_1B_0$=1011, tức số 4

* Để mã hóa các số từ 8 đến 15, cho IC2 hoạt động ở trạng thái từ 0 đến 7 (đưa ngã vào ứng với số muốn mã xuống thấp, các ngã vào cao hơn lên mức 1 và các ngã vào thấp hơn xuống mức 0), bất chấp các ngã vào dữ liệu của IC1 (cho IC1 hoạt động ở trạng thái 9), nên các ngã ra $(A_2A_1A_0)_1$=111, đây là điều kiện mở các cổng AND để cho mã số ra là $B_2B_1B_0$=

$A_2A_1A_0$ của IC2, , trong lúc đó $B_3 = G_{s2} = 0$, ta được kết quả từ 0111 đến 0000, tức từ 8 đến 15.

Thí dụ để mã số 14, đưa ngã vào 14 xuống mức 0, đưa ngã vào 15 lên mức 1, bất chấp các ngã vào từ 0 đến 13, mã số ra là $B_3B_2B_1B_0 = G_{s2}B_2B_1B_0 = 0001$, tức số 14

Muốn có ngã ra chỉ số nhị phân đúng với ngã vào được tác động mà không phải đảo các bit ta có thể thay các cổng AND bằng cổng NAND

4.1.2 Mạch tạo mã BCD cho số thập phân

Mạch gồm 10 ngã vào tượng trưng cho 10 số thập phân và 4 ngã ra là 4 bit của số BCD. Khi một ngã vào (tượng trưng cho một số thập phân) được tác động bằng cách đưa lên mức cao các ngã ra sẽ cho số BCD tương ứng

Bảng sự thật của mạch:

Trạng thái các ngã vào									Mã số ra		
9	8	7	6	5	4	3	2	1	0	A_3	A_2	A_1	A_0
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	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	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

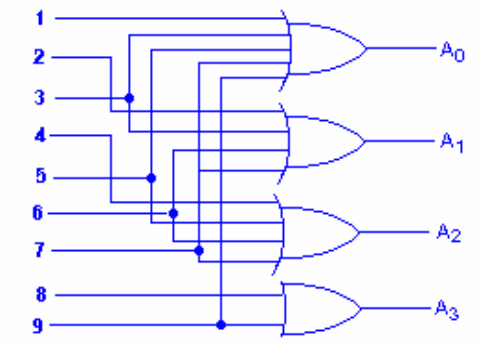
Bảng 4.3

Không cần bảng Karnaugh ta có thể viết ngay các hàm xác định các ngã ra:

$A_0 = 1 + 3 + 5 + 7 + 9$ $A_1 = 2 + 3 + 6 + 7$

$A_2 = 4 + 5 + 6 + 7$ $A_3 = 8 + 9$

Mạch cho ở (H 4.4)



(H 4.4)

Để tạo mã BCD ưu tiên cho số lớn, ta viết lại bảng sự thật và dùng phương pháp đại số để đơn giản các hàm xác định các ngã ra A_3 , A_2 , A_1 , A_0

Trạng thái các ngã vào								Mã số ra		
9	8	7	6	5	4	3	2	A_3	A_2	A_1 A_0
0	0	0	0	0	0	0	0	0	0	0 0
0	0	0	0	0	0	0	0	0	0	0 1
0	0	0	0	0	0	0	1	0	0	1 0
0	0	0	0	0	0	1	x	0	0	1 1
0	0	0	0	0	1	x	x	0	1	0 0
0	0	0	0	1	x	x	x	0	1	0 1
0	0	0	1	x	x	x	x	0	1	1 0
0	0	1	x	x	x	x	x	0	1	1 1
0	1	x	x	x	x	x	x	1	0	0 0
1	x	x	x	x	x	x	x	1	0	0 1

Bảng 4.4

$$A_3 = \bar{9}.8 + 9 = 9 + 8$$

$$A_2 = 7\bar{8}.\bar{9} + 6.7\bar{8}.\bar{9} + 5.6.7\bar{8}.\bar{9} + 4.5.6.7\bar{8}.\bar{9} = (7 + 6.\bar{7} + 5.6.\bar{7} + 4.5.6.7)\bar{8}.\bar{9}$$

$$A_2 = (7 + 6 + 5 + 4)\bar{8}.\bar{9} = (7 + 6 + 5 + 4)\overline{(8 + 9)}$$

$$A_1 = 7\bar{8}.\bar{9} + 6.7\bar{8}.\bar{9} + 3.\bar{4}.5.6.7\bar{8}.\bar{9} + 2.\bar{3}.\bar{4}.5.6.7\bar{8}.\bar{9} = (7 + 6.\bar{7} + 3.\bar{4}.5.6.\bar{7} + 2.\bar{3}.\bar{4}.5.6.7)\bar{8}.\bar{9}$$

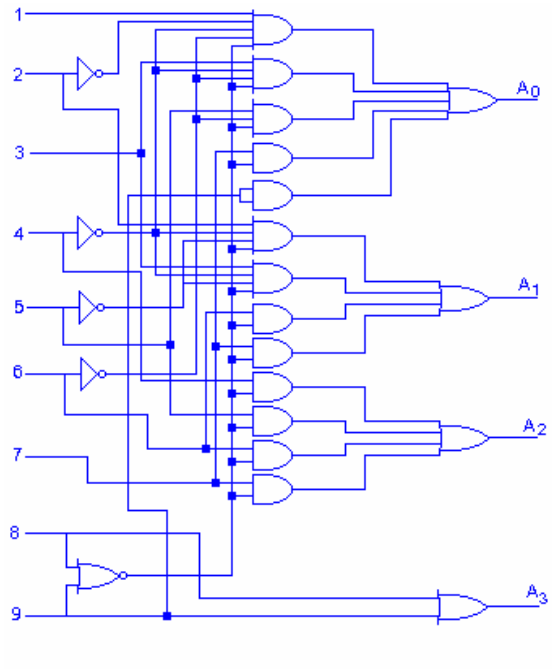
$$A_1 = (7 + 6 + 3.\bar{4}.\bar{5} + 2.\bar{3}.\bar{4}.5)\bar{8}.\bar{9} = (7 + 6 + 3.\bar{4}.\bar{5} + 2.\bar{4}.5)\overline{(8 + 9)}$$

$$A_0 = 9 + 7\bar{8}.\bar{9} + 5.\bar{6}.\bar{7}\bar{8}.\bar{9} + 3.\bar{4}.5.6.\bar{7}\bar{8}.\bar{9} + 1.\bar{2}.\bar{3}.\bar{4}.5.6.\bar{7}\bar{8}.\bar{9}$$

$$= 9 + (7 + 5.\bar{6}.\bar{7} + 3.\bar{4}.\bar{5}.\bar{6}.\bar{7} + 1.\bar{2}.\bar{3}.\bar{4}.\bar{5}.\bar{6}.\bar{7})\bar{8}.\bar{9}$$

$$A_0 = 9 + (7 + 5.\bar{6} + 3.\bar{4}.\bar{6} + 1.\bar{2}.\bar{4}.\bar{6})\bar{8}.\bar{9} = 9 + (7 + 5.\bar{6} + 3.\bar{4}.\bar{6} + 1.\bar{2}.\bar{4}.\bar{6})(\overline{8+9})$$

Mạch cho ở (H 4.5)



(H 4.5)

4.1.3 Mạch chuyển mã

Mạch chuyển từ một mã này sang một mã khác cũng thuộc loại mã hóa.

❁Mạch chuyển mã nhị phân sang Gray

Thử thiết kế mạch chuyển từ mã nhị phân sang mã Gray của số 4 bit.

Trước tiên viết bảng sự thật của số nhị phân và số Gray tương ứng. Các số nhị phân là các biến và các số Gray sẽ là hàm của các biến đó.

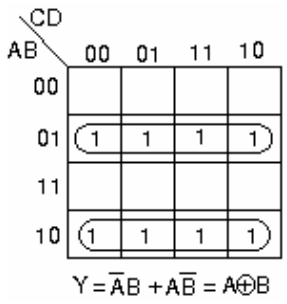
A	B	C	D		X	Y	Z	T
0	0	0	0	→	0	0	0	0
0	0	0	1	→	0	0	0	1
0	0	1	0	→	0	0	1	1
0	0	1	1	→	0	0	1	0
0	1	0	0	→	0	1	1	0
0	1	0	1	→	0	1	1	1
0	1	1	0	→	0	1	0	1
0	1	1	1	→	0	1	0	0
1	0	0	0	→	1	1	0	0
1	0	0	1	→	1	1	0	1
1	0	1	0	→	1	1	1	1
1	0	1	1	→	1	1	1	0
1	1	0	0	→	1	0	1	0
1	1	0	1	→	1	0	1	1
1	1	1	0	→	1	0	0	1
1	1	1	1	→	1	0	0	0
				→				

Bảng 4.5

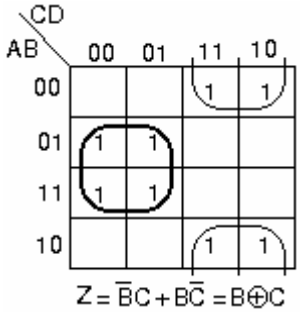
Dùng bảng Karnaugh để xác định X, Y, Z, T theo A, B, C, D

Quan sát bảng sự thật ta thấy ngay: X = A,

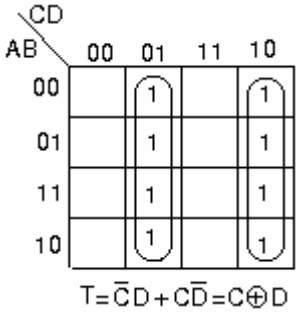
Vậy chỉ cần lập 3 bảng Karnaugh cho các biến Y, Z, T (H 4.6 a,b,c) và kết quả cho ở (H 4.6 d)



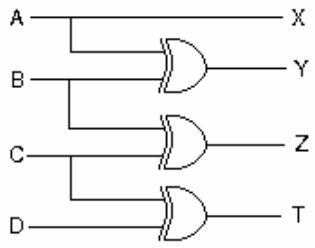
(a)



(b)



(c)



(H 4.6) (d)

4.2 . MẠCH GIẢI MÃ

4.2.1 Giải mã n đường sang 2^n đường

4.2.1.1 Giải mã 2 đường sang 4 đường:

Thiết kế mạch Giải mã 2 đường sang 4 đường có ngã vào cho phép (cũng được dùng để nối mạch)

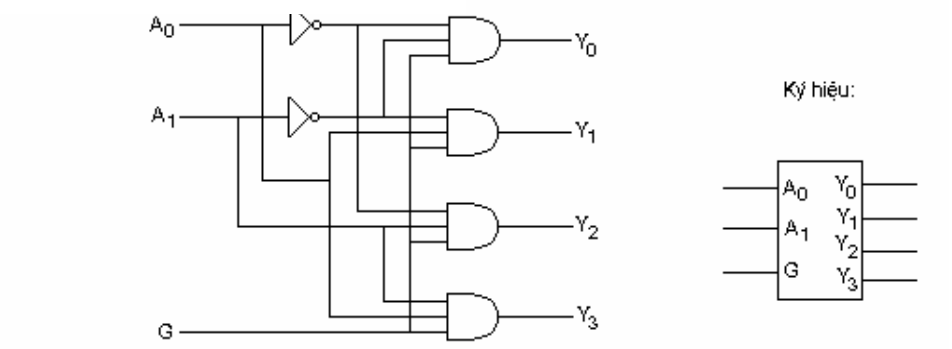
Để đơn giản, ta xét mạch giải mã 2 đường sang 4 đường có các ngã vào và ra đều tác động cao .

Bảng sự thật, các hàm ngã ra và sơ đồ mạch:

Vào			Ra			
G	A_1	A_0	Y_0	Y_1	Y_2	Y_3
0	**x**	**x**	**0**	**0**	**0**	**0**
1	**0**	**0**	**1**	**0**	**0**	**0**
1	**0**	**1**	**0**	**1**	**0**	**0**
1	**1**	**0**	**0**	**0**	**1**	**0**
1	**1**	**1**	**0**	**0**	**0**	**1**

00

$$Y_0 = G.\overline{A}_1\overline{A}_0$$
$$Y_1 = G.\overline{A}_1 A_0$$
$$Y_2 = G.A_1\overline{A}_0$$
$$Y_3 = G.A_1 A_0$$



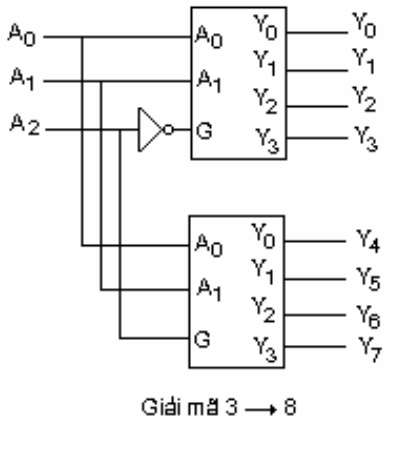
(H 4.7)

4.2.1.2 Giải mã 3 đường sang 8 đường

Dùng 2 mạch giải mã 2 đường sang 4 đường để thực hiện mạch giải mã 3 đường sang 8 đường (H 4.8)

Vào			Ra							
A_2	A_1	A_0	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Quan sát bảng sự thật ta thấy: Trong các tổ hợp số 3 bit có 2 nhóm trong đó các bit thấp A_1A_0 hoàn toàn giống nhau, một nhóm có bit $A_2 = 0$ và nhóm kia có $A_2 = 1$. Như vậy ta có thể dùng ngã vào G cho bit A_2 và mắc mạch như sau.



(H 4.8)

Khi A_2=G=0, IC1 giải mã cho 1 trong 4 ngã ra thấp và khi A_2=G=1, IC2 giải mã cho 1 trong 4 ngã ra cao

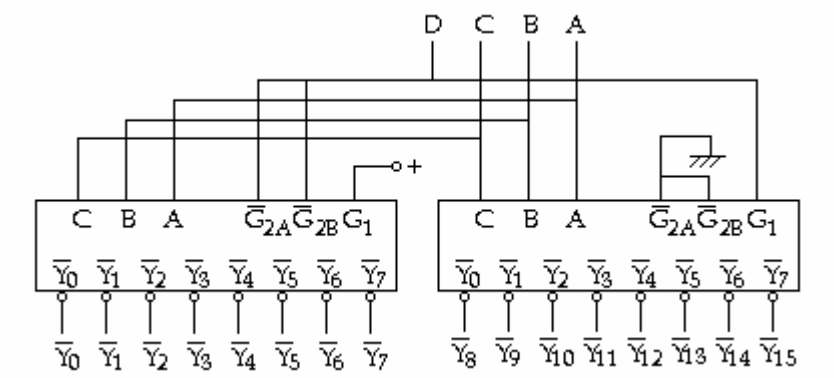
Trên thị trường hiện có các loại IC giải mã như:

- 74139 là IC chứa 2 mạch giải mã 2 đường sang 4 đường, có ngã vào tác động cao, các ngã ra tác động thấp, ngã vào cho phép tác động thấp.
- 74138 là IC giải mã 3 đường sang 8 đường có ngã vào tác động cao, các ngã ra tác động thấp, hai ngã vào cho phép G_{2A} và G_{2B} tác động thấp, G_1 tác động cao.
- 74154 là IC giải mã 4 đường sang 16 đường có ngã vào tác động cao, các ngã ra tác động thấp, 2 ngã vào cho phép E_1 và E_2 tác động thấp

Dưới đây là bảng sự thật của IC 74138 và cách nối 2 IC để mở rộng mạch giải mã lên 4 đường sang 16 đường (H 4.9)

Vào					Ra							
Cho phép		Dữ liệu										
G_1	G_2	C	B	A	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
x	H	x	x	x	H	H	H	H	H	H	H	H
L	x	x	x	x	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

Ghi chú $G_2 = G_{2A}+G_{2B}$, H = 1, L =0, x: bất chấp



(H 4.9)

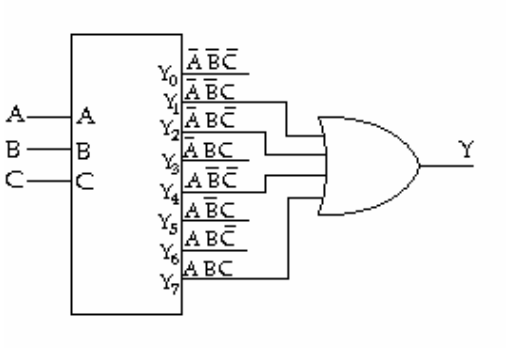
Một ứng dụng quan trọng của mạch giải mã là dùng giải mã địa chỉ cho bộ nhớ bán dẫn.

Ngoài ra, mạch giải mã kết hợp với một cổng OR có thể tạo được hàm logic.

Thí dụ, thiết kế mạch tạo hàm $Y=f(A,B,C)=\overline{A}BC+A\overline{B}\overline{C}+A\overline{B}C+ABC$

Với hàm 3 biến, ta dùng mạch giải mã 3 đường sang 8 đường. 8 ngã ra mạch giải mã tương ứng với 8 tổ hợp biến của 3 biến, các ngã ra tương ứng với các tổ hợp biến có trong hàm sẽ lên mức 1. Với một hàm đã viết dưới dạng tổng chuẩn, ta chỉ cần dùng một cổng OR có số ngã vào bằng với số tổ hợp biến trong hàm nối vào các ngã ra tương ứng của mạch giải mã để cộng các tổ hợp biến có trong hàm lại ta sẽ được hàm cần tạo.

Như vậy, mạch tạo hàm trên có dạng (H 4.10)



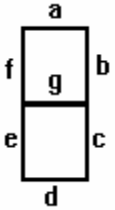
(H 4.10)

Dĩ nhiên, với những hàm chưa phải dạng tổng chuẩn, chúng ta phải chuẩn hóa. Và nếu bài toán có yêu cầu ta phải thực hiện việc đổi cổng, bằng cách dùng định lý De Morgan.

4.2.2 Giải mã BCD sang 7 đọan

4.2.2.1 Đèn 7 đọan

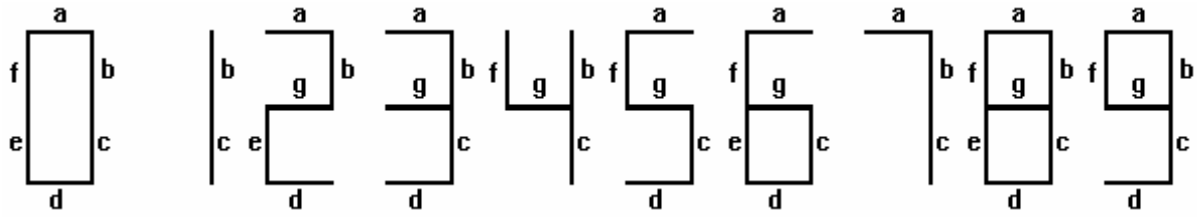
Đây là loại đèn dùng hiển thị các số từ 0 đến 9, đèn gồm 7 đọan a, b, c, d, e, f, g, bên dưới mỗi đọan là một led (đèn nhỏ) hoặc một nhóm led mắc song song (đèn lớn). Qui ước các đọan cho bởi (H 4.11).



(H 4.11)

Khi một tổ hợp các đọan cháy sáng sẽ tạo được một con số thập phân từ 0 - 9.

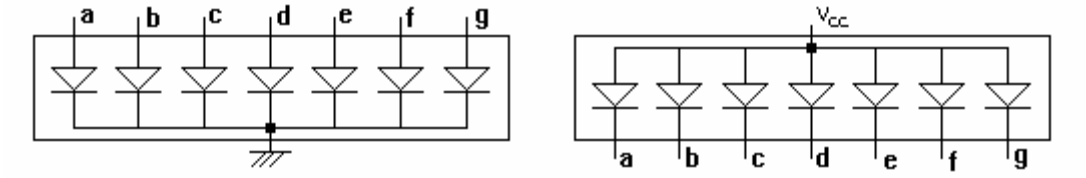
(H 4.12) cho thấy các đoạn nào cháy để thể hiện các số từ 0 đến 9



0 1 2 3 4 5 6 7 8 9

(H 4.12)

Đèn 7 đoạn cũng hiển thị được một số chữ cái và một số ký hiệu đặc biệt.
Có hai loại đèn 7 đoạn:
- Loại catod chung (H 4.13a), dùng cho mạch giải mã có ngã ra tác động cao.
- Loại anod chung (H 4.13b), dùng cho mạch giải mã có ngã ra tác động thấp.



(a) (H 4.13) (b)

4.2.2.2 Mạch giải mã BCD sang 7 đoạn :

Mạch có 4 ngã vào cho số BCD và 7 ngã ra thích ứng với các ngã vào a, b, c, d, e, f, g của led 7 đoạn, sao cho các đoạn cháy sáng tạo được số thập phân đúng với mã BCD ở ngã vào.

Bảng sự thật của mạch giải mã 7 đoạn, có ngã ra tác động thấp:

Số	Ngã vào				Ngã ra						
TP	D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	1	1	0	0	1	1	1	1
2	0	0	1	0	0	0	1	0	0	1	0
3	0	0	1	1	0	0	0	0	1	1	0
4	0	1	0	0	1	0	0	1	1	0	0
5	0	1	0	1	0	1	0	0	1	0	0
6	0	1	1	0	0	1	0	0	0	0	0
7	0	1	1	1	0	0	0	1	1	1	1
8	1	0	0	0	0	0	0	0	0	0	0
9	1	0	0	1	0	0	0	0	1	0	0

Bảng 4.6

Dùng Bảng Karnaugh hoặc có thể đơn giản trực tiếp với các hàm chứa ít tổ hợp, ta có kết quả:

$$a = \overline{D}\,\overline{B}(C\overline{A} + \overline{C}A)$$
$$b = C\overline{B}A + CB\overline{A}$$
$$c = \overline{D}\,\overline{C}B\overline{A}$$
$$d = \overline{D}\,\overline{C}\,\overline{B}A + C\overline{B}\,\overline{A} + CBA$$
$$e = A + C\overline{B}$$
$$f = \overline{C}B + BA + \overline{D}\,\overline{C}A$$
$$g = \overline{D}\,\overline{C}\,\overline{B} + CBA$$

Từ các kết quả ta có thể vẽ mạch giải mã 7 đoạn dùng các cổng logic.
Hai IC thông dụng dùng để giải mã BCD sang 7 đoạn là:
- CD 4511 (loại CMOS, ngã ra tác động cao và có đệm)
- 7447 (loại TTL, ngã ra tác động thấp, cực thu để hở)

Chúng ta khảo sát một IC giải mã BCD sang 7 đoạn : IC 7447
Bảng sự thật của 7447:

Vào								Ra						
Sô / Hàm	LT	RBI	D	C	B	A	BI (1) $\overline{RBO}$	a	b	c	d	e	f	g
0	1	1	0	0	0	0	1	0	0	0	0	0	0	1
1	1	x	0	0	0	1	1	1	0	0	1	1	1	1
2	1	x	0	0	1	0	1	0	0	1	0	0	1	0
3	1	x	0	0	1	1	1	0	0	0	0	1	1	0
4	1	x	0	1	0	0	1	1	0	0	1	1	0	0
5	1	x	0	1	0	1	1	0	1	0	0	1	0	0
6	1	x	0	1	1	0	1	1	1	0	0	0	0	0
7	1	x	0	1	1	1	1	0	0	0	1	1	1	1
8	1	x	1	0	0	0	1	0	0	0	0	0	0	0
9	1	x	1	0	0	1	1	0	0	0	1	1	0	0
10	1	x	1	0	1	0	1	1	1	1	0	0	1	0
11	1	x	1	0	1	1	1	1	1	0	0	1	1	0
12	1	x	1	1	0	0	1	1	0	1	1	1	0	0
13	1	x	1	1	0	1	1	0	1	1	0	1	0	0
14	1	x	1	1	1	0	1	1	1	1	0	0	0	0
15	1	x	1	1	1	1	1	1	1	1	1	1	1	1
(2)	x	x	x	x	x	x	0	1	1	1	1	1	1	1
(3)	1	0	0	0	0	0	0	1	1	1	1	1	1	1
(4)	0	x	x	x	x	x	1	0	0	0	0	0	0	0

Ghi chú:

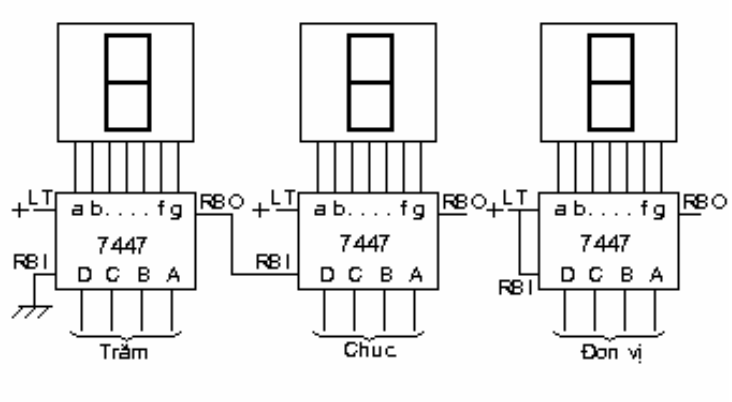
1. BI/RBO được nối theo kiểu điểm AND bên trong IC và được dùng như ngã vào xóa (Blanking Input, BI) và/hoặc ngã ra xóa dợn sóng (Ripple Blanking Output, RBO). Ngã vào BI phải được để hở hay giữ ở mức cao khi cần thực hiện giải mã cho số ra. Ngã vào xóa dợn sóng (Ripple Blanking Input, RBI) phải để hở hay ở mức cao khi muốn đọc số 0.

2. Khi đưa ngã vào BI xuống thấp, ngã ra lên 1 (không tác động) bất chấp các ngã vào còn lại. Ta nói IC làm việc dưới điều kiện bị ép buộc và đây là trường hợp duy nhất BI giữ vai trò ngã vào.

3. Khi ngã vào RBI ở mức 0 và A=B=C=D=0, tất cả các ngã ra kể cả RBO đều xuống 0. Ta nói IC làm việc dưới điều kiện đáp ứng.

4. Khi BI/RBO để hở hay được giữ ở mức 1 và ngã vào thử đèn (Lamp test, LT) xuống 0, tất cả các led đều cháy (ngã ra xuống 0).

Dựa vào bảng sự thật và các ghi chú 7447 là IC giải mã BCD sang 7 đoạn có đầy đủ các chức năng khác như : thử đèn, xóa số 0 khi nó không có nghĩa. Ta có thể hiểu rõ hơn chức năng này với thí dụ mạch hiển thị một kết quả có 3 chữ số sau đây: (H 4.14)



(H 4.14)

Vận hành của mạch có thể giải thích như sau:

- IC hàng đơn vị có ngã vào RBI đưa lên mức cao nên đèn số 0 hàng đơn vị luôn luôn được hiển thị (dòng 0 trong bảng sự thật), điều này là cần thiết để xác nhận rằng mạch vẫn chạy và kết quả giải mã là số 0.
- IC hàng chục có ngã vào RBI nối với ngã ra RBO của IC hàng trăm nên số 0 hàng chục chỉ được hiển thị khi số hàng trăm khác 0 (RBO=1) (dòng 0 đến 15).
- IC hàng trăm có ngã vào RBI đưa xuống mức thấp nên số 0 hàng trăm luôn luôn tắt (dòng ghi chú 3).

4.2.2.3 Hiển thị 7 đoạn bằng tinh thể lỏng (liquid crystal displays, LCD)

LCD gồm 7 đoạn như led thường và có chung một cực nền (backplane). Khi có tín hiệu xoay chiều biên độ khoảng 3 - 15 V_{RMS} và tần số khoảng 25 - 60 Hz áp giữa một đoạn và cực nền, thì đoạn đó được tác động và sáng lên.

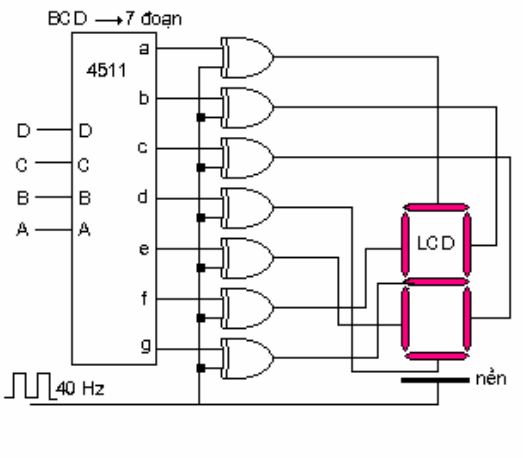
Trên thực tế người ta tạo hai tín hiệu nghịch pha giữa nền và một đoạn để tác động cho đoạn đó cháy.

Để hiểu được cách vận chuyển ta có thể dùng IC 4511 kết hợp với các cổng EX-OR để thúc LCD (H 4.15). Các ngã ra của IC 4511 (Giải mã BCD sang 7 đoạn, tác động cao) nối vào các ngã vào của các cổng EX-OR, ngã vào còn lại nối với tín hiệu hình vuông tần số khoảng 40 Hz (tần số thấp có thể gây ra nhấp nháy), tín hiệu này đồng thời được đưa vào nền. Khi một ngã ra mạch giải mã lên cao, ngã ra cổng EX-OR cho một tín hiệu đảo pha với tín hiệu ở nền, đoạn tương ứng xem như nhận được tín hiệu có biên độ gấp đôi và sẽ sáng lên. Với các ngã ra mạch giải mã ở mức thấp,
ngã ra cổng EX-OR cho một tín hiệu cùng pha với tín hiệu ở nền nên đoạn tương ứng không sáng.

Người ta thường dùng IC CMOS để thúc LCD vì hai lý do:

- CMOS tiêu thụ năng lượng rất thấp phù hợp với việc dùng pin cho các thiết bị dùng LCD.

- Mức thấp của CMOS đạt trị 0 và tín hiệu thúc LCD sẽ không chứa thành phần một chiều, tuổi thọ LCD được kéo dài. (Mức thấp của TTL khoảng 0,4 V, thành phần DC này làm giảm tuổi thọ của LCD).



(H 4.15)

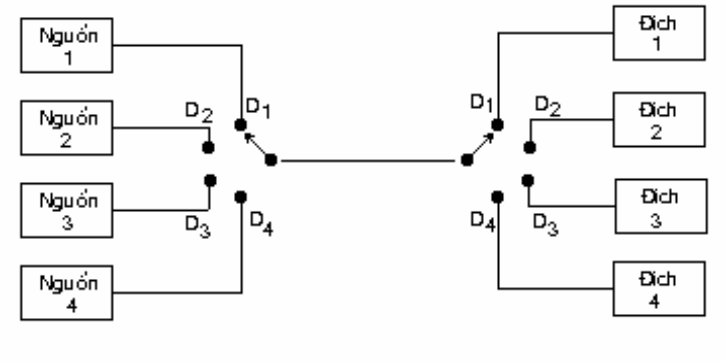
4.3 MẠCH ĐA HỢP VÀ MẠCH GIẢI ĐA HỢP

4.3.1.Khái niệm

Trong truyền dữ liệu, để tiết kiệm đường truyền, người ta dùng một đường dây để truyền nhiều kênh dữ liệu, như vậy phải thực hiện viêc chọn nguồn dữ liệu nào trong các nguồn khác nhau để truyền.

Mạch đa hợp hay còn gọi là mạch chọn dữ liệu sẽ làm công việc này.

Ở nơi thu, dữ liệu nhận được phải được chuyển tới các đích khác nhau, ta cần mạch phân bố dữ liệu hay giải đa hợp (H 4.16).



(H 4.16)

4.3.2 Mạch đa hợp

Còn được gọi là mạch chọn dữ liệu, gồm 2^n ngã vào dữ liệu, n ngã vào địa chỉ (hay điều khiển) và một ngã ra. Khi có một địa chỉ được tác động dữ liệu ở ngã vào tương ứng với địa chỉ đó sẽ được chọn.

- Thiết kế mạch đa hợp 4→1

Mạch có 4 ngã vào dữ liệu D_0 . . . . D_3, 2 ngã vào điều khiển AB và ngã ra Y

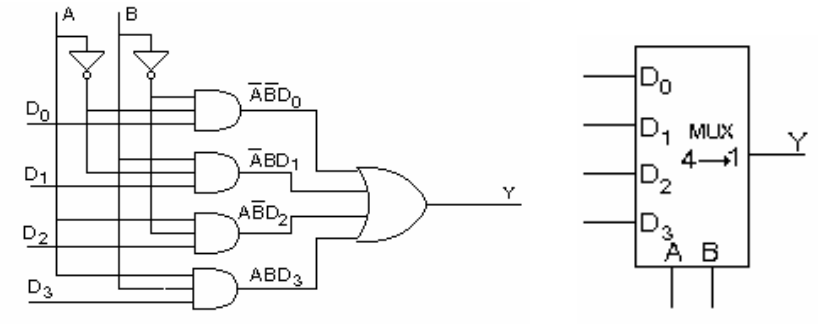
Bảng sự thật:

A	B	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

Tư bảng sự thật ta có hàm Y như sau:

$$Y = \overline{A}.\overline{B}D_0 + \overline{A}BD_1 + A\overline{B}D_2 + ABD_3$$

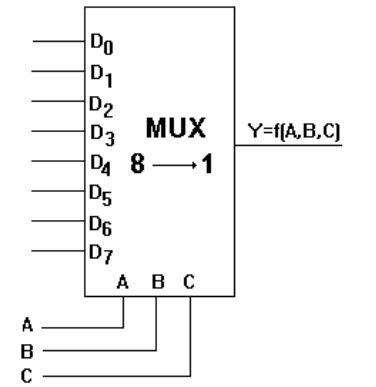
Và mạch có dạng (H 4.17)



(H 4.17)

Nếu chịu khó quan sát ta sẽ thấy mạch đa hợp 4→1 có thể được thiết kế từ mạch giải mã 2 đường sang 4 đường trong đó ngã vào cho phép G đã được tách riêng ra để làm ngã vào dữ liệu (D_0 . . . . D_3) và ngã vào dữ liệu của mạch giải mã đã trở thành ngã vào điều khiển của mạch đa hợp (A, B)

(H 4.18) là ký hiệu một mạch đa hợp với 8 ngã vào dữ liệu, 3 ngã vào điều khiển và 1 ngã ra, ta gọi là đa hợp 8 → 1.



Bảng sự thật:

A	B	C	Y
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

(H 4.18)

Một đa hợp 8 → 1 có ngã ra Y quan hệ với các ngã vào dữ liệu và điều khiển theo hàm :

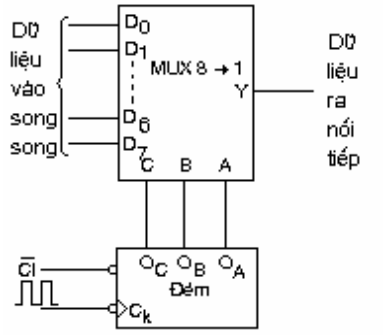
$$Y = \overline{A}.\overline{B}.\overline{C}D_0 + \overline{A}.\overline{B}.CD_1 + \overline{A}B\overline{C}D_2 + \overline{A}BCD_3 + A\overline{B}.\overline{C}D_4 + A\overline{B}CD_5 + AB\overline{C}D_6 + ABCD_7$$

4.3.3 Ứng dung mạch đa hợp

Ngoài chức năng chọn dữ liệu mạch đa hợp còn được dùng để:

4.3.3.1 Biến chuỗi dữ liệu song song thành nối tiếp:

Một mạch đa hợp kết hợp với một mạch đếm sẽ biến chuỗi dữ liệu song song ở ngã vào thành chuỗi dữ liệu nối tiếp ở ngã ra (H 4.19)



(H 4.19)

4.3.3.2 Tạo chuỗi xung tuần hoàn :

Nếu cho dữ liệu vào tuần hoàn, dữ liệu ra nối tiếp cũng tuần hoàn, như vậy chỉ cần đặt trước các ngã vào thay đổi theo một chu kỳ nào đó ta sẽ được chuỗi xung tuần hoàn ở ngã ra.

4.3.3.3 Tạo hàm:

❄ Một đa hợp $2^n \rightarrow 1$ có thể tạo hàm n biến bằng cách cho các biến vào ngã vào điều khiển và cho trị riêng của hàm vào các ngã vào dữ liệu.

Thí dụ: Để tạo hàm 3 biến bằng đa hợp 8→1 ta viết lại biểu thức của đa hợp

$$Y = \overline{A}.\overline{B}.\overline{C}D_0 + \overline{A}.\overline{B}.CD_1 + \overline{A}B\overline{C}D_2 + \overline{A}BCD_3 + A\overline{B}.\overline{C}D_4 + A\overline{B}CD_5 + AB\overline{C}D_6 + ABCD_7$$

So sánh với biểu thức của hàm viết dưới dạng triển khai theo định lý Shanon thứ nhất

$$f(A,B,C) = \overline{A}.\overline{B}.\overline{C}f(0,0,0) + \overline{A}.\overline{B}.Cf(0,0,1) + \overline{A}B\overline{C}f(0,1,0) + \overline{A}BCf(0,1,1) + A\overline{B}.\overline{C}f(1,0,0)$$
$$+ A\overline{B}Cf(1,0,1) + AB\overline{C}f(1,1,0) + ABCf(1,1,1)$$

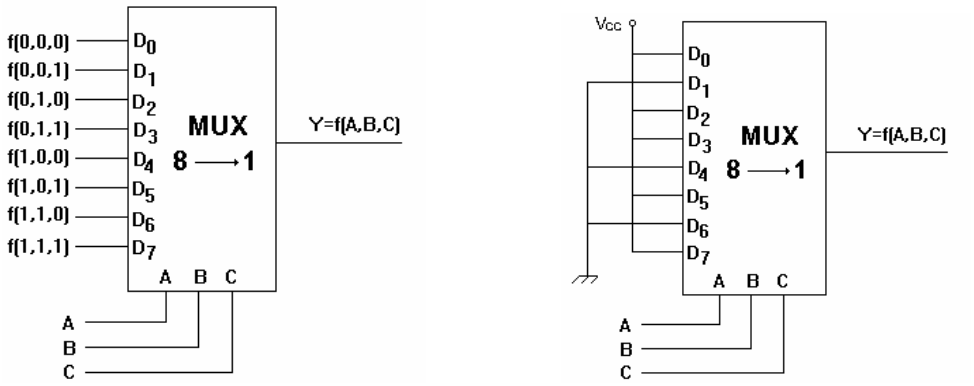
Ta được kết quả:

$D_0 = f(0,0,0)$; $D_1 = f(0,0,1)$, ... $D_6 = f(1,1,0)$ và $D_7 = f(1,1,1)$

Thí dụ: Tạo hàm:

$$Y = f(A, B, C) = \overline{A}.\overline{B}.\overline{C} + \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}C + ABC$$

Ta thấy $D_0=D_2=D_3=D_5=D_7=1$ nên các ngã vào này được nối lên nguồn, các ngã vào còn lại $D_1=D_4=D_6=0$ nên được đưa xuống mass (H 4.20).



(H 4.20)

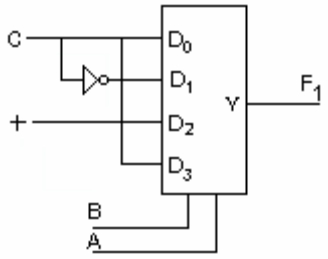
❄ Một đa hợp $2^n \rightarrow 1$ kết hợp với một cổng NOT có thể tạo hàm (n+1) biến. *Thí dụ* : Tạo hàm $F_1 = A\overline{B} + \overline{A}B\overline{C} + \overline{B}C + AC$ dùng đa hợp $4 \rightarrow 1$ và cổng NOT

Giải

Đa hợp 4 sang 1 thực hiện hàm: $Y = \overline{A}\,\overline{B}D_0 + \overline{A}BD_1 + A\overline{B}D_2 + ABD_3$

Chuẩn hóa hàm F_1 : $F_1 = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}C + A\overline{B}\,\overline{C} + ABC$

Để $Y = F_1$ ta phải có: $D_0 = C; D_1 = \overline{C}; D_2 = 1; D_3 = C$



(H 4.21)

Trên thực tế, ta có đủ các loại mạch đa hợp từ $2 \rightarrow 1$ (IC 74157), $4 \rightarrow 1$ (IC 74153), $8 \rightarrow 1$ (IC 74151) và $16 \rightarrow 1$ (74150) . . . .

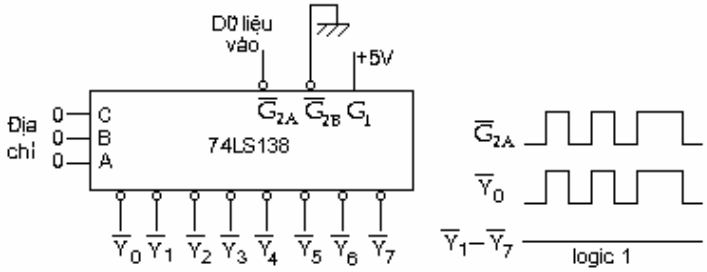
Ngoài ra, để chọn dữ liệu là các nguồn tín hiệu tương tự, ta cũng có các đa hợp tương tự với tên gọi khóa tương tự (analog switch), được chế tạo theo công nghệ MOS như IC 4051 (8 kênh) IC 4053 (2 kênh). . . . Cũng có loại khóa sử dụng được cho cả tín hiệu tương tự và số (bilateral switches) như IC 4016, IC 4066,. . mà sinh viên có thể tìm hiểu, sử dụng dễ dàng khi có bảng tra kỹ thuật.

4.3.4 Mạch giải đa hợp

Mạch giải đa hợp thực chất là mạch giải mã trong đó ngã vào cho phép trở thành ngã vào dữ liệu và ngã vào của tổ hợp số nhị phân trở thành ngã vào địa chỉ.

Trên thị trường, người ta chế tạo mạch giải mã và giải đa hợp chung trong một IC, tùy theo điều kiện mà sử dụng. Thí dụ IC 74138 là IC Giải mã 3 sang 8 đường đồng thời là mạch giải đa hợp $1 \rightarrow 8$.

Khi sử dụng IC 74138 làm mạch giải đa hợp, người ta dùng một ngã vào cho phép làm ngã vào dữ liệu và các ngã vào số nhị phân làm ngã vào địa chỉ. (H 4.22a) là IC 74138 dùng giải đa hợp với ngã vào dữ liệu là $\overline{G_{2A}}$. (H 4.22b) là dạng dữ liệu vào $\overline{G_{2A}}$ và ra ở $\overline{Y_0}$ (vì CBA=000), các ngã ra khác ($\overline{Y_1} - \overline{Y_7}$) ở mức cao.



(a) (H 4.22) (b)

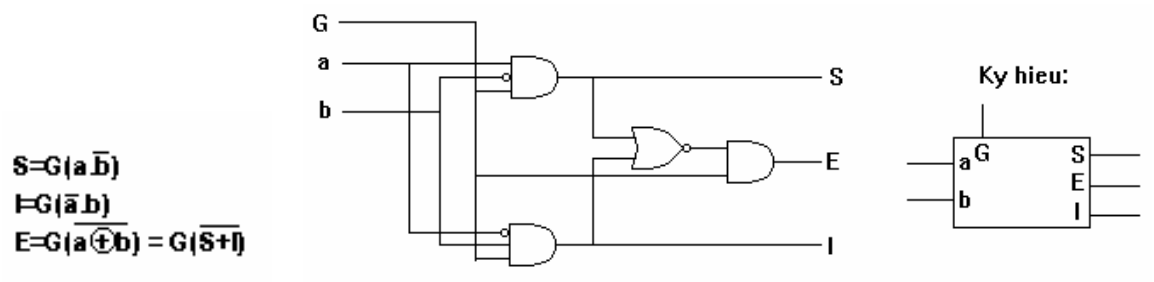
4.4 MẠCH SO SÁNH

4.4.1 Mạch so sánh 2 số 1 bit

Trước tiên ta thiết kế mạch so sánh hai số 1 bit.
Bảng sự thật của mạch so sánh một bit có ngã vào cho phép (nối mạch) G :

G	a	b	S (a>b)	I (a<b)	E (a=b)
0	x	x	0	0	0
1	0	0	0	0	1
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	0	0	1

Bảng 4.7



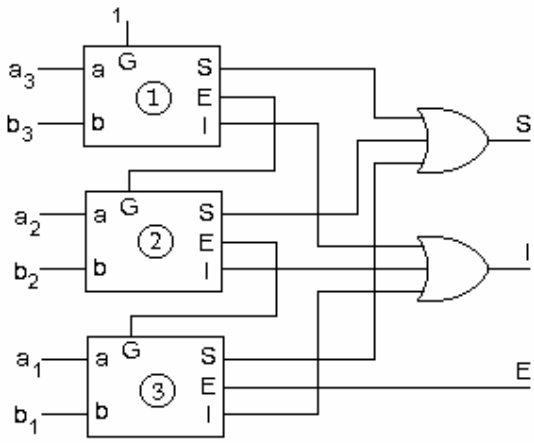
(H 4.23)

Từ mạch so sánh 1 bit ta có thể mở rộng để so sánh nhiều bit.

4.4.2 Mạch so sánh 2 số nhiều bit

Để so sánh 2 số nhiều bit, trước tiên người ta so sánh 2 bit cao nhất (MSB), kết quả lớn hoặc nhỏ hơn do 2 bit này quyết định, nếu 2 bit MSB bằng nhau người ta so sánh 2 bit có trọng số thấp hơn tiếp theo và kết quả được quyết định theo cách tương tự như ở 2 bit MSB. . . . . Sự so sánh được lặp lại cho đến bit LSB để được kết cuối cùng.

Dưới đây là sơ đồ mạch so sánh 3 bit (H 4.24).



(H 4.24)

- IC 1 so sánh 2 bit cao (a_3 & b_3) nên ngã vào cho phép được đưa lên mức cao, nếu kết quả bằng nhau, ngã ra E của nó lên cao, cho phép IC 2 so sánh, nếu kết quả lại bằng nhau, ngã ra E của IC 2 lên cao cho phép IC 3 so sánh, kết quả bằng nhau cuối cùng chỉ bởi ngã ra E của IC 3.

- Các ngã vào cổng OR nhận tín hiệu từ các ngã ra S (hoặc I) sẽ cho kết quả lớn hơn (hoặc nhỏ hơn) tùy vào kết quả so sánh ở bất cứ bit nào. Thật vậy khi có một kết quả lớn hơn (hoặc nhỏ hơn) thì S (hoặc I) ở một IC lên cao, các ngã ra E và I (hoặc S) của các IC khác bằng 0, đây là điều kiện mở cổng OR để cho kết quả so sánh xuất hiện ở một trong các cổng OR này.

Trên thị trường có sẵn loại IC so sánh 4 bit 7485 có ngã nối mạch để mở rộng việc so sánh cho số nhiều bit hơn.

Bảng sự thật của IC 7485

Trạng thái	Ngã A_3,B_3	vào A_2,B_2	so A_1,B_1	sánh A_0,B_0	Vào A'>B'	nối A'<B'	mạch A'=B'	ra A>B	A<B	A=B
1	$A_3>B_3$	x	x	x	x	x	x	1	0	0
2	$A_3<$	x	x	x	x	x	x	0	1	0
3	B_3	$A_2>B_2$	x	x	x	x	x	1	0	0
4	$A_3=$	$A_2<$	x	x	x	x	x	0	1	0
5	B_3	B_2	$A_1>B_1$	x	x	x	x	1	0	0
6	$A_3=$	$A_2=$	$A_1<$	x	x	x	x	0	1	0
7	B_3	B_2	B_1	$A_0>B_0$	x	x	x	1	0	0
8	$A_3=$	$A_2=$	$A_1=$	$A_0<$	x	x	x	0	1	0
9	B_3	B_2	B_1	B_0	0	0	1	0	0	1
10	$A_3=$	$A_2=$	$A_1=$	$A_0=$	1	0	0	1	0	0
11	B_3	B_2	B_1	B_0	0	1	0	0	1	0
	$A_3=$	$A_2=$	$A_1=$	$A_0=$						
	B_3	B_2	B_1	B_0						
	$A_3=$	$A_2=$	$A_1=$	$A_0=$						
	B_3	B_2	B_1	B_0						
	$A_3=$	$A_2=$	$A_1=$							
	B_3	B_2	B_1							
	$A_3=$	$A_2=$								
	B_3	B_2								
	$A_3=$									
	B_3									

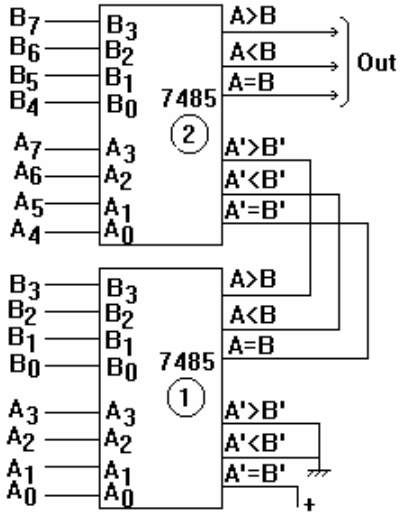
Bảng 4.8

Dựa vào bảng sự thật, ta thấy:

- Khi dùng IC 7485 để so sánh 2 số 4 bit ta phải giữ ngã vào nối mạch A'=B' ở mức cao, hai ngã vào nối mạch còn lại ở mức thấp, như vậy IC mới thể hiện được kết quả của trạng thái 9.

- Khi so sánh 2 số nhiều bit hơn ta phải dùng nhiều IC 7485 và nối ngã ra của IC so sánh bit thấp vào ngã vào nối mạch tương ứng của các IC so sánh các bit cao hơn và IC so sánh các bit thấp nhất có ngã vào nối mạch được mắc như khi dùng riêng lẻ. Để đọc được kết quả so sánh ta phải quan tâm tới các trạng thái 9, 10 và 11 trong bảng sự thật.

(H 4.25) cho ta cách mắc 2 IC 7485 để so sánh 2 số nhị phân 8 bit:



(H 4.25)

Thí dụ :

a. So sánh hai số A7 . . . .A0 = 10101111 và B7 . . . . B0 = 10110001

IC 2 so sánh các bit cao A_7 . . .A_4 = 1010 và B_7 . . .B_4 =1011 có A_7= B_7 , A_6= B_6 , A_5= B_5 và A_4<B_4 cho ngã ra A<B = 1 bất chấp trạng thái của các ngã vào nối mạch (trạng thái 8). Điều này có nghĩa là khi IC so sánh bit cao thấy có kết quả khác nhau giữa 2 số bit cao thì không quan tâm tới kết quả của bit thấp.

b. So sánh hai số A_7 . . . .A_0 = 10101111 và B_7 . . . . B_0 = 10101001

Trong trường hợp này kết quả hai số bit cao bằng nhau nên IC 2 nhìn vào ngã vào nối mạch để xem kết quả so sánh của IC1 (so sánh bit thấp), $A_3A_2A_1A_0$ =1111>$B_3B_2B_1B_0$ = 1001 nên ngã ra A>B = 1 để chỉ kết quả so sánh của 2 số 8 bit (trạng thái 10).

4.5 MẠCH KIỂM / PHÁT CHẴN LẺ

Do yêu cầu kiểm sai trong truyền dữ liệu, người ta có phương pháp kiểm tra chẵn lẻ. Trong phương pháp này, ngoài các bit dữ liệu, người ta thêm vào 1 bit kiểm tra sao cho tổng số bit 1 kể cả bit kiểm tra là số chẵn (KT chẵn) hoặc lẻ (KT lẻ)

1	0	1	1	0	0	1	1

← Bit chẵn lẻ thêm vào (KT lẻ)

1	1	0	0	1	0	1	0

← Bit chẵn lẻ thêm vào (KT chẵn)

Ở nơi thu, mạch kiểm tra chẵn lẻ sẽ kiểm tra lại số số 1 có trên tất cả các bit để biết dòng dữ liệu nhận được đúng hay sai.

Với phương pháp này máy thu sẽ có kết luận đúng khi số bit lỗi là số lẻ. Như vậy phương pháp chỉ cho kết quả đúng với xác suất 50%, tuy nhiên vì xác suất để một lỗi xảy ra là rất nhỏ nên phương pháp vẫn được sử dụng phổ biến trong một số hệ truyền thông.

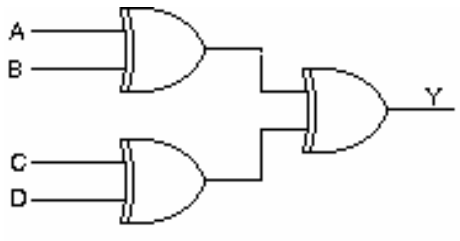
4.5.1 Mạch phát chẵn lẻ (Parity Generator, PG)

Ta sẽ xét trường hợp mạch có 4 bit dữ liệu.

Mạch có 4 ngã vào dữ liệu A, B, C, D và 1 ngã vào chọn chẵn lẻ S

- Giai đoạn 1: Thiết kế mạch ghi nhận số bit 1 là chẵn hay lẻ
Giả sử ta muốn có mạch báo kết quả Y=1 khi số bit 1 là lẻ và Y=0 khi ngược lại.
Lợi dụng tính chất của cổng EX-OR có ngã ra =1 khi số số 1 ở ngã vào là lẻ. Với 4 ngã vào, ta dùng 3 cổng EX-OR để thực hiện mạch ghi nhận này:

$$Y = (A \oplus B) \oplus (C \oplus D)$$



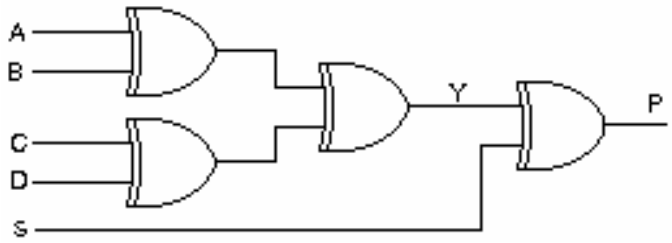
(H 4.26)

- Giai đoạn 2: Thiết kế phần mạch tạo bit chẵn lẻ P theo sự điều khiển của ngã vào S
Giả sử ta muốn có
Tổng số bit 1 của A, B, C, D, P là lẻ khi S = 1 và chẵn khi S = 0

S	Số bít 1 của ABCD	Y	P
0	Lẻ	1	1
0	Chẵn	0	0
1	Lẻ	1	0
1	Chẵn	0	1

Bảng 4.9

Bảng 4.9 cho kết quả: $P = S \oplus Y$
Vậy mạch có dạng



(H 4.27)

4.5.2 Mạch kiểm chẵn lẻ (Parity checker, PC)

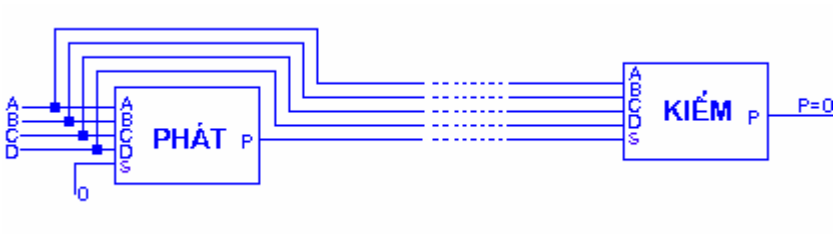
Nếu ta xem mạch phát ở (H 4.27) như là mạch có 5 ngã vào thì ngã ra P quan hệ với số lượng bit 1 ở các ngã vào đó có thể được suy ra từ bảng 4.9

Số bít 1 của ABCDS	P
Lẻ	1
Chẵn	0

Bảng 4.9

Như vậy, ta có thể dùng mạch phát ở trên để làm mạch kiểm tra chẵn lẻ.

Tóm lại, một hệ thống gồm mạch phát và kiểm chẵn lẽ được mắc như (H 4.28) Khi ngã vào S của mạch phát đưa xuống mức 0, nếu bản tin nhận đúng thì ngã ra P ở mạch kiểm cũng xuống 0.



(H 4.28)

Trên thị trường có các IC kiểm/phát chẵn lẻ như 74180 (9 bit) 74280 (9 bit), loại CMOS có 40101 (9 bit), 4531 (13 bit).

BÀI TẬP

1. Thiết kế mạch mã hóa 32 đường sang 5 đường dùng IC 74148 và cổng logic.

2. Thiết kế mạch giải mã 4 đường sang 16 đường từ mạch giải mã 2 đường sang 4 đường có ngã vào cho phép.

3. Thiết kế mạch so sánh 4 bit từ mạch so sánh 1 bit

4. Thiết kế mạch chuyển từ mã Gray sang mã nhị phân

5. Thiết kế mạch chuyển từ mã BCD sang mã Excess-3 của các số từ 0 đến 9.
(Mã Excess-3 của 1 số có được từ trị nhị phân tương ứng cộng thêm 3, thí dụ mã số 0 là 0011, mã số 9 là 1100)

6. Dùng một mạch giải mã 3 sang 8 đường, 2 cổng NAND 3 ngã vào và 1 cổng AND 2 ngã vào thực hiện các hàm sau:

$F_1 = \Sigma(1,2,3)$; $F_2 = \Sigma(4,5,7)$; $F_3 = \Sigma(1,2,3,4,5,7)$

7. Cài đặt các hàm sau dùng bộ dồn kênh (multiplexer) $4 \rightarrow 1$ (Dùng thêm cổng logic nếu cần)

$$F_1 = A\bar{B} + \bar{A}B\bar{C} + \bar{B}C + AC$$

$$F_2 = A \oplus (\bar{B}C)$$

$$F_3 = \prod(1,3,6)$$

8. Thiết kế mạch MUX $4 \rightarrow 1$ từ các MUX $2 \rightarrow 1$

9. Dùng 2 MUX $2 \rightarrow 1$ để thực hiện 1 MUX $3 \rightarrow 1$ như sau:

AB = 00 chọn C
AB = 01 chọn D
AB =1X chọn E (Trường hợp này B không xac định).

10. Thực hiện hàm Z= AB +BC + CA

- Giải mã 3 sang 8 đường (dùng thêm cổng logic nếu cần).
- Đa hợp $4 \rightarrow 1$ (dùng thêm cổng logic nếu cần).
- Hai mạch cộng bán phần và một cổng OR.

☎ CHƯƠNG 5 MẠCH TUẦN TỰ

✸ CHỐT RS

♠ Chốt RS tác động mức cao
♠ Chốt RS tác động mức thấp

✸ FLIPFLOP

♠ FF RS
♠ FF JK
♠ FF T
♠ FF D

✸ MẠCH GHI DỊCH

✸ MẠCH ĐẾM
♠ Đồng bộ
♠ Không đồng bộ
♠ Đếm vòng

Trong chương trước, chúng ta đã khảo sát các loại mạch tổ hợp, đó là các mạch mà ngã ra của nó chỉ phụ thuộc vào các biến ở ngã vào mà không phụ thuộc vào trạng thái trước đó của mạch. Nói cách khác, đây là loại mạch không có khả năng nhớ, một chức năng quan trọng trong các hệ thống logic.

Chương này sẽ bàn về loại mạch thứ hai: mạch tuần tự.

- Mạch tuần tự là mạch có trạng thái ngã ra không những phụ thuộc vào tổ hợp các ngã vào mà còn phụ thuộc trạng thái ngã ra trước đó. Ta nói mạch tuần tự có **tính nhớ**. Ngã ra Q_+ của mạch tuần tự là hàm logic của các biến ngã vào A, B, C . . . . và ngã ra Q trước đó.

$$Q_+ = f(Q,A,B,C\ldots)$$

- Mạch tuần tự vận hành dưới tác động của xung đồng hồ và được chia làm 2 loại: **Đồng bộ** và **Không đồng bộ**. Ở mạch đồng bộ, các phần tử của mạch chịu **tác động đồng thời** của xung đồng hồ (C_K) và ở mạch không đồng bộ thì không có điều kiện này.

Phần tử cơ bản cấu thành mạch tuần tự là các flipflop

5.1 FLIP FLOP

Mạch flipflop (FF) là mạch dao động đa hài lưỡng ổn tức mạch tạo ra sóng vuông và có hai trạng thái ổn định. Trạng thái của FF chỉ thay đổi khi có xung đồng hồ tác động.

Một FF thường có:

- Một hoặc hai ngã vào dữ liệu, một ngã vào xung C_K và có thể có các ngã vào với các chức năng khác.

- Hai ngã ra, thường được ký hiệu là Q (ngã ra chính) và $\overline{Q}$ (ngã ra phụ). Người ta thường dùng trạng thái của ngã ra chính để chỉ trạng thái của FF. Nếu hai ngã ra có trạng thái giống nhau ta nói FF ở **trạng thái cấm.**

Flipflop có thể được tạo nên từ mạch chốt (latch)

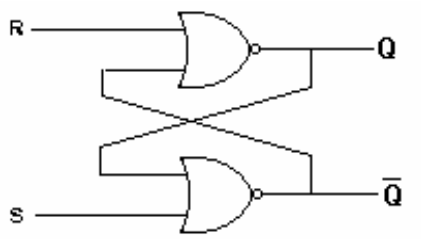
Điểm khác biệt giữa một mạch chốt và một FF là: FF chịu tác động của xung đồng hồ còn mạch chốt thì không.

Người ta gọi tên các FF khác nhau bằng cách dựa vào tên các ngã vào dữ liệu của chúng.

5.1.1 Chốt RS

5.1.1.1. Chốt RS tác động mức cao:

(H 5.1) là chốt RS có các ngã vào R và S tác động mức cao.



(H 5.1)

Các trạng thái logic của mạch cho ở bảng 5.1:

(Đối với mạch chốt vì không có tác động của xung đồng hồ nên ta có thể hiểu trạng thái trước là trạng thái giả sử, còn trạng thái sau là trạng thái khi mạch ổn định).

R	S	Q	Q_+
0	0	0	0⎫ Tác dụng nhớ
0	0	1	1⎭ Q_+= Q
0	1	0	1⎫ Đặt (Set)
0	1	1	1⎭ Q_+=1
1	0	0	0⎫ Đặt lại (Reset)
1	0	1	0⎭ Q_+=0
1	1	0	⎫ $Q_+=\overline{Q}_+=0$ (Cấm)
1	1	1	⎭

Bảng 5.1

R	S	Q_+
0	0	Q
0	1	1
1	0	0
1	1	Cấm

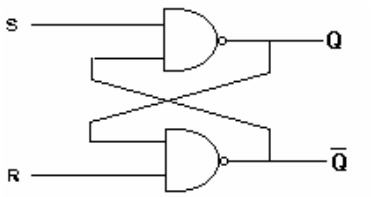
Bảng 5.2

Từ Bảng 5.1 thu gọn lại thành Bảng 5.2 và tính chất của chốt RS tác động mức cao được tóm tắt như sau:

- Khi R=S=0 (cả 2 ngã vào đều không tác động), ngã ra không đổi trạng thái.
- Khi R=0 và S=1 (ngã vào S tác động), chốt được Set (tức đặt Q_+=1).
- Khi R=1 và S=0 (ngã vào R tác động), chốt được Reset (tức đặt lại Q_+=0).
- Khi R=S=1 (cả 2 ngã vào đều tác động), chốt rơi vào trạng thái cấm

5.1.1.2. Chốt RS tác động mức thấp:

(H 5.2) là chốt RS có các ngã vào R và S tác động mức thấp. Các trạng thái logic cho bởi Bảng 5.3

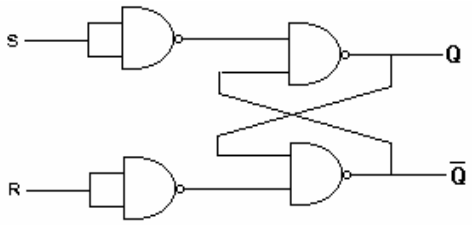


(H 5.2)

S	R	Q_+
0	0	Cấm
0	1	1
1	0	0
1	1	Q

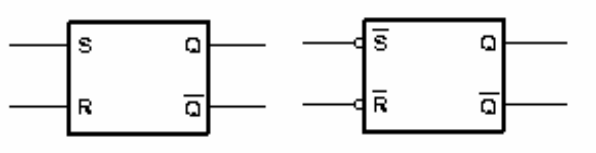
Bảng 5.3

Để có chốt RS tác động mức cao dùng cổng NAND, người ta thêm vào 2 cổng đảo ở các ngã vào của mạch (H 5.2)



(H 5.3)

(H 5.4a) là ký hiệu chốt RS tác động cao và (H 5.4b) là chốt RS tác động thấp.

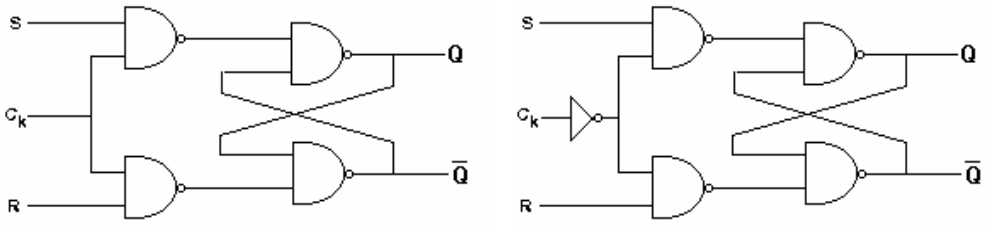


(a) (b)

(H 5.4)

5.1.2 Flip Flop RS

Trong các phần dưới đây, ta luôn sử dụng chốt RS tác động mức cao dùng cổng NAND. Khi thêm ngã vào xung C_K cho chốt RS ta được FF RS . (H 5.5a) là FF RS có các ngã vào R, S và xung đồng hồ C_K đều tác động mức cao.



(a) (H 5.5) (b)

Hoạt động của FF (H 5.5a) cho bởi Bảng sự thật: (Bảng 5.4)

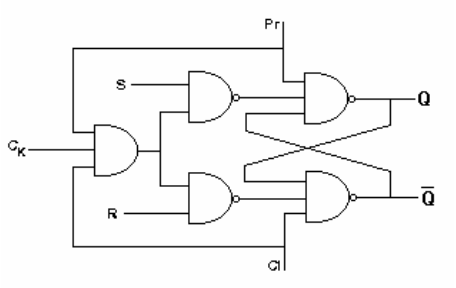
Vào			Ra
C_K	S	R	Q_+
0	x	x	Q
1	0	0	Q
1	0	1	0
1	1	0	1
1	1	1	Cấm

Bảng 5.4

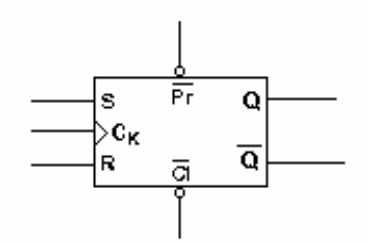
Để có FF RS có xung đồng hồ tác động thấp chỉ cần thêm một cổng đảo cho ngã vào C_K (H 5.5b). Ta có bảng sự thật giống Bảng 5.4, trừ ngã vào C_K phải đảo lại

5.1.2.1. Flipflop RS có ngã vào Preset và Clear:

Tính chất của FF là có trạng thái ngã ra bất kỳ khi mở máy. Trong nhiều trường hợp, có thể cần đặt trước ngã ra Q=1 hoặc Q=0, muốn thế, người ta thêm vào FF các ngã vào Preset (đặt trước Q=1) và Clear (Xóa Q=0), mạch có dạng (H 5.6a) và (H 5.6b) là ký hiệu của FF RS có ngã vào Preset và Clear tác động mức thấp.



(a)



(b)

(H 5.6)

Thay 2 cổng NAND cuối bằng hai cổng NAND 3 ngã vào, ta được FF RS có ngã vào Preset (Pr) và Clear (Cl).

- Khi ngã Pr xuống thấp (tác động) và ngã Cl lên cao ngã ra Q lên cao bất chấp các ngã vào còn lại.
- Khi ngã Cl xuống thấp (tác động) và ngã Pr lên cao ngã ra Q xuống thấp bất chấp các ngã vào còn lại.
- Ngoài ra 2 ngã vào Pr và Cl còn được đưa về 2 ngã vào một cổng AND, nơi đưa tín hiệu C_K vào, mục đích của việc làm này là khi một trong 2 ngã vào Pr hoặc Cl tác động thì mức thấp của tín hiệu này sẽ khóa cổng AND này, vô hiệu hóa tác dụng của xung C_K.

Bảng sự thật của FF RS có Preset và Clear (tác động thấp) cho ở bảng 5.5

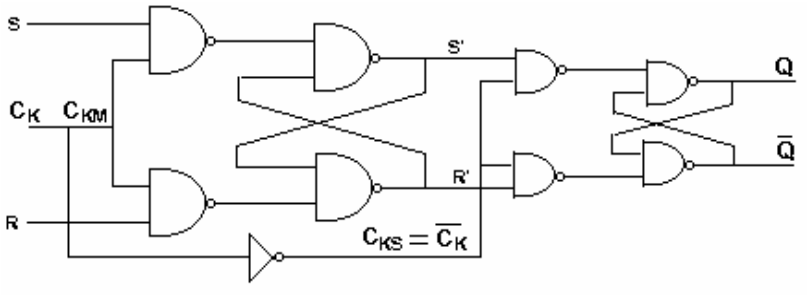
Pr	Cl	C_K	S	R	Q_+
0	0	x	x	x	Cấm
0	1	x	x	x	1
1	0	x	x	x	0
1	1	0	x	x	Q
1	1	1	0	0	Q
1	1	1	0	1	0
1	1	1	1	0	1
1	1	1	1	1	Cấm

Bảng 5.5

Lưu ý: Trên bảng 5.5, dòng thứ nhất tương ứng với trạng thái cấm vì hai ngã vào Pr và Cl đồng thời ở mức tác động, 2 cổng NAND cuối cùng đều đóng, nên Q_+=Q=1.

5.1.2.2. Flipflop RS chủ tớ:

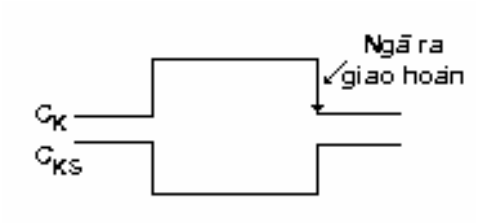
Kết nối thành chuỗi hai FF RS với hai ngã vào xung C_K của hai FF có mức tác động trái ngược nhau, ta được FF chủ tớ (H 5.7).



(H 5.7)

Hoạt động của FF được giải thích như sau:

- Do C_{KS} của tầng tớ là đảo của $C_{KM} = C_K$ của tầng chủ nên khi C_K=1, tầng chủ giao hoán thì tầng tớ ngưng. Trong khoảng thời gian này, dữ liệu từ ngã vào R và S được đưa ra và ổn định ở ngã ra R' và S' của tầng chủ, tại thời điểm xung C_K xuống thấp, R' và S' được truyền đến ngã ra Q và $\overline{Q}$ (H 5.8)



(H 5.8)

- Đối với trường hợp R = S =1 khi C_K=1 thì R'= S' =1, nhưng khi C_K xuống thấp thì một trong hai ngã ra này xuống thấp, do đó mạch thoát khỏi trạng thái cấm, nhưng S' hay R' xuống thấp trước thì không đoán trước được nên mạch rơi vào trạng thái bất định, nghĩa là Q_+ có thể =1 có thể =0, nhưng khác với $\overline{Q}_+$. Ta có bảng sự thật:

S	R	C_K	Q_+
0	0	↓	Q
0	1	↓	0
1	0	↓	1
1	1	↓	Bất định
		↓	

Bảng 5.6

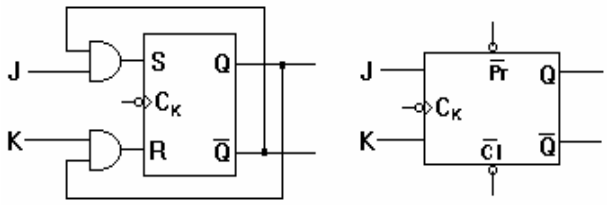
Tóm lại, FF RS chủ tớ đã thoát khỏi trạng thái cấm nhưng vẫn rơi vào trạng thái bất định, đồng thời ta được FF có ngã vào xung đồng hồ tác động bởi cạnh xuống của tín hiệu C_K.

Để có FF RS có ngã vào xung đồng hồ tác động bởi cạnh lên của tín hiệu C_K ta có thể dời cổng NOT đến ngã vào FF chủ và cho tín hiệu C_K vào thẳng FF tớ.

Mặc dù thoát khỏi trạng thái cấm nhưng FF RS chủ tớ vẫn còn trạng thái bất định nên người ta ít sử dụng FF RS trong trường hợp R=S.

5.1.3 Flipflop JK

FF JK được tạo ra từ FF RS theo sơ đồ như (H 5.9a).



(a) (b)
(H 5.9)

(H 5.9b) là ký hiệu FF JK có ngã vào Pr và Cl tác động thấp.
Bảng sự thật 5.7 (Để đơn giản, ta bỏ qua các ngã vào Pr và Cl)

J	K	Q	$\overline{Q}$	S=J$\overline{Q}$	R=KQ	C_K	Q_+
0	0	0	1	0	0	↓	Q
0	0	1	0	0	0	↓	Q
0	1	0	1	0	0	↓	Q=0
0	1	1	0	0	1	↓	0
1	0	0	1	1	0	↓	1
1	0	1	0	0	0	↓	Q=1
1	1	0	1	1	0	↓	1
1	1	1	0	0	1	↓	0

Bảng 5.7

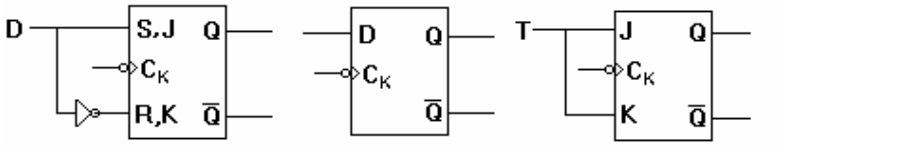
J	K	C_K	Q_+
0	0	↓	Q
0	1	↓	0
1	0	↓	1
1	1	↓	$\overline{Q}$

Bảng 5.8

Bảng 5.8 là bảng rút gọn, suy ra từ bảng 5.7
Kết quả từ bảng 5.8 cho thấy:
FF JK đã thoát khỏi trạng thái cấm và thay vào đó là **trạng thái đảo** (khi J=K=1 thì $Q_+=\overline{Q}$). Người ta lợi dụng trạng thái đảo này để thiết kế mạch đếm

5.1.4 FlipFlop D

Thiết kế từ FF RS (hoặc JK) bằng cách nối một cổng đảo từ S qua R (hoặc từ J qua K). Dữ liệu được đưa vào ngã S (J) mà bây giờ gọi là ngã vào D (H 5.10a&b) và bảng 5.9 cho thấy các trạng thái của FF, cụ thể là mỗi khi có xung C_K tác động dữ liệu từ ngã vào sẽ xuất hiện ở ngã ra.



(a) (b) (c)
(H 5.10)

D	C_K	Q_+
0	↓	0
1	↓	1

Bảng 5.9

T	C_K	Q_+
0	↓	Q
1	↓	$\overline{Q}$

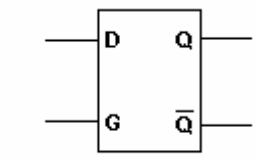
Bảng 5.10

5.1.5 FlipFlop T

Nối chung hai ngã vào J và K của FF JK ta được FF T (H 5.10c). Tính chất của FF T thể hiện trong bảng sự thật 5.10:
- Khi T=0, **FF không đổi trạng thái** dù có tác động của C_K.
- Khi T=1, **FF đổi trạng thái** mỗi lần có xung C_K tác động.

5.1.6 Mạch chốt D

Mạch chốt D hoạt động giống FF D, chỉ khác ở điểm ngã vào xung đồng hồ C_K được thay bằng ngã vào cho phép G, và tác động bằng mức chứ không bằng cạnh (H 5.11) và Bảng 5.11.

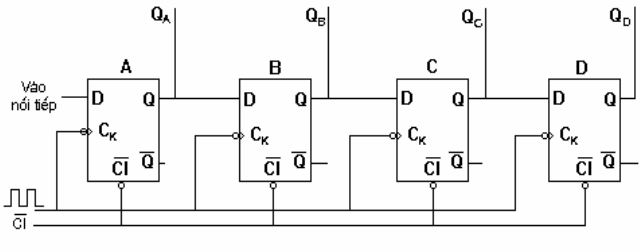


G	D	Q_+
0	x	Q
1	0	0
1	1	1

(H 5.11) Bảng 5.11

5.2 MẠCH GHI DỊCH

5.2.1 Sơ đồ nguyên tắc và vận chuyển (H 5.12)



(H 5.12)

(H 5.12) là sơ đồ một mạch ghi dịch 4 bit đơn giản, mạch gồm 4 FF D nối thành chuỗi (ngã ra Q của FF trước nối vào ngã vào D của FF sau) và các ngã vào C_K được nối chung lại (các FF chịu tác động đồng thời). Mạch ghi dịch này có khả năng dịch phải.

Ngã vào D_A của FF đầu tiên được gọi là **ngã vào dữ liệu nối tiếp**, các ngã ra Q_A, Q_B, Q_C, Q_D là các **ngã ra song song**, ngã ra của FF cuối cùng (FF D) là **ngã ra nối tiếp** .

Trước khi cho mạch hoạt động, tác dụng một xung xóa vào các ngã vào $\overline{Cl}$ (đưa các chân $\overline{Cl}$ đã được nối chung xuống thấp rồi lên cao) để các ngã ra $Q_A = Q_B = Q_C = Q_D = 0$.

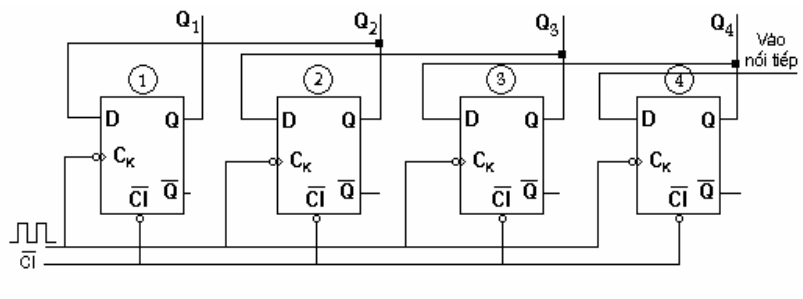
Cho dữ liệu vào D_A, sau mỗi xung đồng hồ, dữ liệu từ tầng trước lần lượt truyền qua tầng sau. (Giả sử D_A là chuỗi dữ liệu gồm 3 bit cao, 2 bit thấp rồi 1 cao và 1 thấp), trạng thái các ngã ra của các FF cho ở Bảng 5.12

Vào			Ra			
Cl	C_K	D_A	Q_A	Q_B	Q_C	Q_D
0	x	x	0	0	0	0
1	↓	1	1	0	0	0
1	↓	1	1	1	0	0
1	↓	1	1	1	1	0
1	↓	0	0	1	1	1
1	↓	0	0	0	1	1
1	↓	1	1	0	0	1
1	↓	0	0	1	0	0
	↓					

Bảng 5.12

Các mạch ghi dịch được phân loại tùy vào số bit (số FF), chiều dịch (phải/trái), các ngã vào/ra (nối tiếp/song song).

Để có mạch dịch trái, dữ liệu nối tiếp đưa vào ngã vào D của FF cuối cùng và các ngã ra của FF sau nối ngược trở lại ngã vào của FF trước (H 5.13)



(H 5.13)

Cho dữ liệu nối tiếp vào ngã vào D của FF 4, sau mỗi xung đồng hồ, dữ liệu truyền từ tầng sau ra tầng trước. Giả sử chuỗi dữ liệu giống như trên, trạng thái các ngã ra của các FF cho ở bảng 5.13

Vào			Ra			
Cl	C_K	D_4	Q_1	Q_2	Q_3	Q_4
0	x	x	0	0	0	0
1	↓	1	0	0	0	1
1	↓	1	0	0	1	1
1	↓	1	0	1	1	1
1	↓	0	1	1	1	0
1	↓	0	1	1	0	0
1	↓	1	1	0	0	1
1	↓	0	0	0	1	0

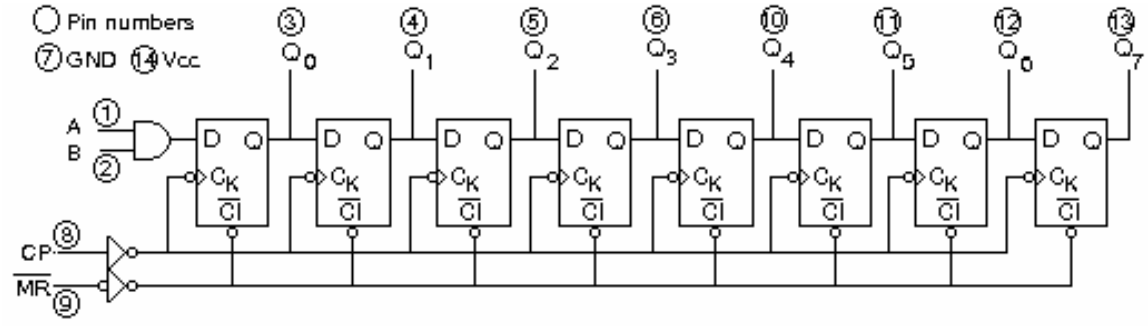
Bảng 5.13

5.2.2 Vài IC ghi dịch tiêu biểu

Trên thị trường hiện có khá nhiều loại IC ghi dịch, có đầy đủ các chức năng dịch phải trái, vào/ra nối tiếp, song song. Sau đây, chúng ta khảo sát 2 IC tiêu biểu:

- IC **74164**: dịch phải 8 bit;
- IC **7495**: 4 bit , dịch phải, trái, vào/ra nối tiếp/song song .

5.2.2.1. IC 74164:

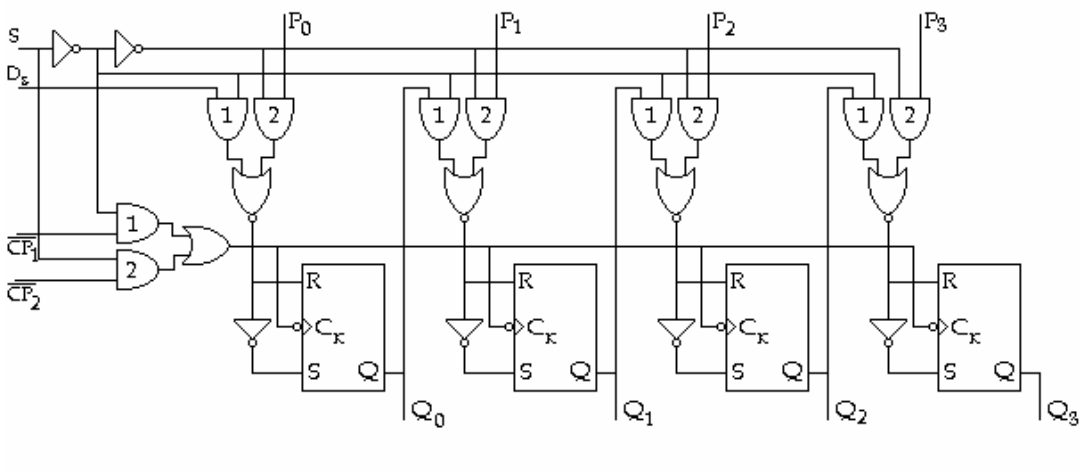


(H 5.14)

$\overline{MR}$: Master Reset, đây cũng là chân Clear của cả mạch, tác động thấp

CP: Clock pulse, ngã vào xung đồng hồ tác động cạnh lên.

5.2.2.2. IC 7495:



(H 5.15)

Ý nghĩa các chân: **S**: Mode control input **Ds**: Serial Data input
P_0 - P_3 : Parrallel data inputs
CP_1 : Serial Clock **CP_2** : Parrallel clock
Q_0 - Q_3 : Parrallel outputs

Dươi đây là các bước thao tác để thực hiện các chức năng của IC

✸ **Nạp dữ liệu song song**

- Chuẩn bị dữ liệu ở các ngã vào P_0 - P_3
- Cho S = 1, dữ liệu được đưa vào các ngã vào của các FF, CP_1 bị khóa, CP_2 là ngã vào C_K, dữ liệu xuất hiện ở ngã ra Q_0 - Q_3 khi có cạnh xuống của C_K

✸ **Dịch phải**

- Sau khi đã nạp dữ liệu song song - Chuẩn bị dữ liệu nối tiếp.
- Cho S = 0
- Đưa dữ liệu nối tiếp vào ngã vào Ds, CP_2 bị khóa, CP_1 là ngã vào C_K, khi C_K tác động, dữ liệu sẽ dịch phải từng bit một trên các ngã ra Q_0 - Q_3

✸ **Dịch trái**

- Nối ngã ra FF sau vào ngã vào song song của FF trước - P_3 là ngã vào nối tiếp
- S = 1 để cách ly ngã ra FF trước với ngã vào FF sau
- CP_2 là ngã vào xung C_K, dữ liệu sẽ dịch trái ứng với cạnh xuống của C_K.

Lưu ý: Mặc dù có 2 ngã vào cho xung C_K nhưng khi sử dụng chúng thường được nối chung lại, lý do là vì ứng với một trạng thái của tín hiệu điều khiển S chỉ có một trong hai cổng AND mở để cho tín hiệu C_K đi qua.

5.2.3. Ứng dụng của ghi dịch

Ghi dịch có khá nhiều ứng dụng:

- Một số nhị phân khi dịch trái 1 bit, giá trị được nhân lên gấp đôi và được chia hai khi dịch phải một bit.

Thí dụ số $1010.00 = 10_{10}$ khi dịch trái thành $10100.0 = 20_{10}$ và khi dịch phải thành $101.000 = 5_{10}$.

- Trong máy tính thanh ghi (tên thường gọi của mạch ghi dịch) là nơi lưu tạm dữ liệu để thực hiện các phép tính, các lệnh cơ bản như quay, dịch ....

- Ngoài ra, mạch ghi dịch còn những ứng dụng khác như: tạo mạch đếm vòng, biến đổi dữ liệu nối tiếp ↔ song song, dùng thiết kế các mạch đèn trang trí, quang báo. . . ..

5.3 MẠCH ĐẾM

Lợi dụng tính đảo trạng thái của FF JK khi J=K=1, người ta thực hiện các mạch đếm.

Chức năng của mạch đếm là đếm số xung C_K đưa vào ngã vào hoặc thể hiện số trạng thái có thể có của các ngã ra.

Nếu xét khía cạnh tần số của tín hiệu thì mạch đếm có chức năng chia tần, nghĩa là tần số của tín hiệu ở ngã ra là kết quả của phép chia tần số của tín hiệu C_K ở ngã vào cho số đếm của mạch.

Ta có các loại: mạch đếm đồng bộ, không đồng bộ và đếm vòng.

5.3.1 Mạch đếm đồng bộ

Trong mạch đếm đồng bộ các FF chịu tác động đồng thời của xung đếm C_K.

5.3.1.1 Mạch đếm đồng bộ n tầng, đếm lên

Để thiết kế mạch đếm đồng bộ n tầng (lấy thí dụ n=4), trước tiên lập bảng trạng thái, quan sát bảng trạng thái suy ra cách mắc các ngã vào JK của các FF sao cho mạch giao hoán tạo các ngã ra đúng như bảng đã lập. Giả sử ta dùng FF tác động bởi cạnh xuống của xung C_K (Thật ra, kết quả thiết kế không phụ thuộc vào chiều tác động của xung C_K, tuy nhiên điều này phải được thể hiện trên mạch nên ta cũng cần lưu ý). Với 4 FF mạch đếm được 2^4=16 trạng thái và số đếm được từ 0 đến 15. Ta có bảng trạng thái:

C_k	Q_D	Q_C	Q_B	Q_A	Số đếm
Xóa	0	0	0	0	0
1↓	0	0	0√	1	1
2↓	0	0	1	0	2
3↓	0	0√	1√	1	3
4↓	0	1	0	0	4
5↓	0	1	0√	1	5
6↓	0	1	1	0	6
7↓	0√	1√	1√	1	7
8↓	1	0	0	0	8
9↓	1	0	0√	1	9
10↓	1	0	1	0	10
11↓	1	0√	1√	1	11
12↓	1	1	0	0	12
13↓	1	1	0√	1	13
14↓	1	1	1	0	14
15↓	1√	1√	1√	1	15
16↓	0	0	0	0	0

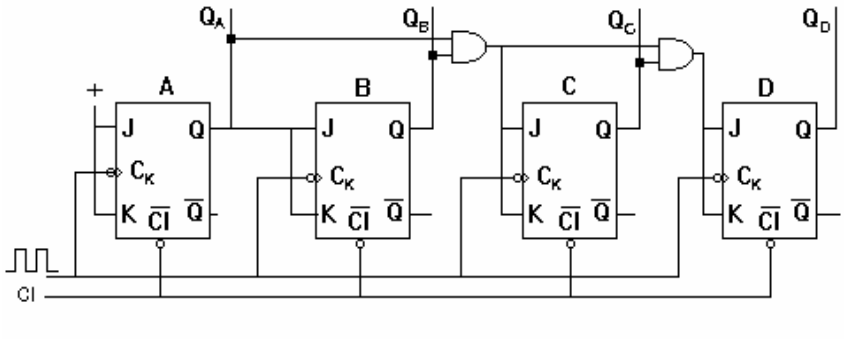
Bảng 5.14

Nhận thấy:

- FF A đổi trạng thái sau từng xung C_K, vậy: $T_A = J_A = K_A = 1$
- FF B đổi trạng thái nếu trước đó $Q_A = 1$, vậy $T_B = J_B = K_B = Q_A$
- FF C đổi trạng thái nếu trước đó $Q_A = Q_B = 1$, vậy: $T_C = J_C = K_C = Q_A.Q_B$
- FF D đổi trạng thái nếu trước đó $Q_A=Q_B=Q_C=1$, vậy:

$$T_D = J_D = K_D = Q_A.Q_B.Q_C = T_C.Q_C$$

Ta được kết quả ở (H 5.16)



(H 5.16)

5.3.1.2 Mạch đếm đồng bộ n tầng, đếm xuống

Bảng trạng thái:

C_k	Q_D	Q_C	Q_B	Q_A	Số đếm
Xóa	0√	0√	0√	0	0
1↓	1	1	1	1	15
2↓	1	1	1√	0	14
3↓	1	1	0	1	13
4↓	1	1√	0√	0	12
5↓	1	0	1	1	11
6↓	1	0	1√	0	10
7↓	1√	0√	0√	0	9
8↓	0	1	1	1	8
9↓	0	1	1√	0	7
10↓	0	1	0	1	6
11↓	0	1√	0√	0	5
12↓	0	0	1	1	4
13↓	0	0	1√	0	3
14↓	0	0	0	1	2
15↓	0	0	0	0	1
16↓					0

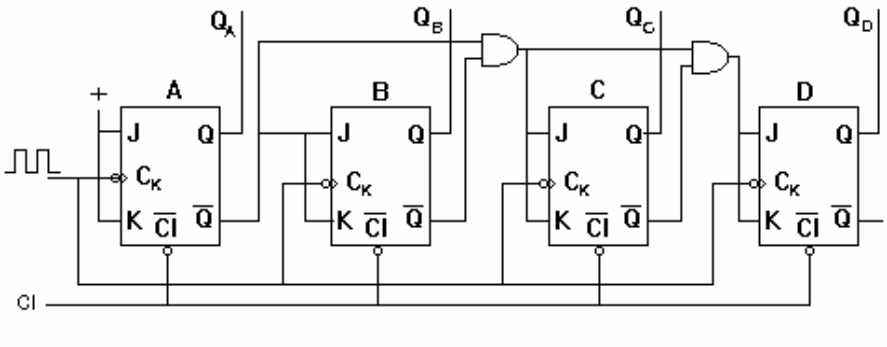
Bảng 5.15

Nhận thấy:
- FF A đổi trạng thái sau từng xung C_K, vậy: $T_A = J_A = K_A = 1$
- FF B đổi trạng thái nếu trước đó $Q_A = 0$, vậy: $T_B = J_B = K_B = \overline{Q_A}$
- FF C đổi trạng thái nếu trước đó $Q_A=Q_B=0$, vậy: $T_C = J_C = K_C = \overline{Q_A}\ \overline{Q_B}$

- FF D đổi trạng thái nếu trước đó $Q_A = Q_B = Q_C = 0$, vậy:

$$T_D = J_D = K_D = \overline{Q_A}\,.\overline{Q_B}\,.\overline{Q_C} = T_C.\,\overline{Q_C}$$

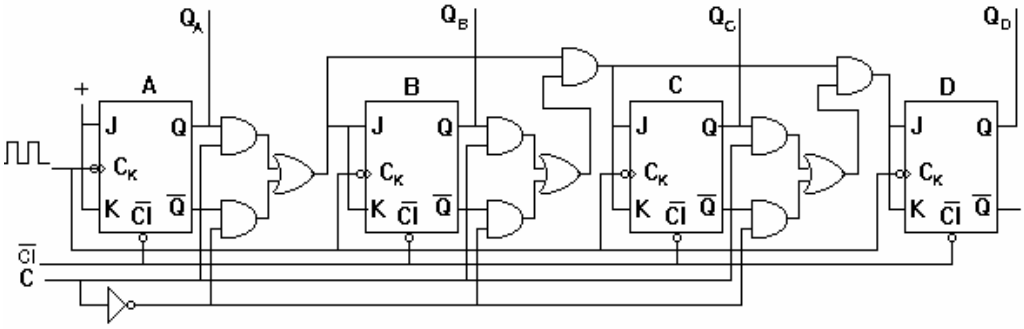
Ta được kết quả ở (H 5.17)



(H 5.17)

5.3.1.3 Mạch đếm đồng bộ n tầng, đếm lên/ xuống

Để có mạch đếm n tầng, đếm lên hoặc xuống ta dùng một đa hợp 2→1 có ngã vào điều khiển C để chọn Q hoặc $\overline{Q}$ đưa vào tầng sau qua các cổng AND. Trong mạch (H 5.18) dưới đây khi C=1 mạch đếm lên và khi C=0 mạch đếm xuống.



(H 5.18)

5.3.1.4 Tần số hoạt động lớn nhất của mạch đếm đồng bộ n tầng:

Trong mạch (H 5.16) ta cần 2 cổng AND. Trong trường hợp tổng quát cho n tầng, số cổng AND là (n-2) như vậy thời gian tối thiểu để tín hiệu truyền qua mạch là:

$$T_{min} = T_{PFF} + T_{P.AND}(n\text{-}2)$$

Tần số cực đại xác định bởi:

$$f_{max} = \frac{1}{T_{min}} = \frac{1}{t_{PFF} + (n-2)T_{PAND}}$$

Để gia tăng tần số làm việc của mạch, thay vì dùng các cổng AND 2 ngã vào ta phải dùng cổng AND nhiều ngã vào và mắc theo kiểu:

$T_A = J_A = K_A = 1$ $\qquad$ $T_B = J_B = K_B = Q_A$

$T_C = J_C = K_C = Q_A.Q_B$ $\qquad$ $T_D = J_D = K_D = Q_A.Q_B.Q_C$

Như vậy tần số làm việc không phụ thuộc vào n và bằng:

$$f_{max} = \frac{1}{T_{PFF} + T_{PAND}}$$

5.3.1.5 Mạch đếm đồng bộ Modulo - N ($N \neq 2^n$)

Để thiết kế mạch đếm modulo - N, trước nhất ta phải chọn số tầng.

Số tầng n phải thỏa điều kiện:

$$2^{n-1} < N < 2^n$$

Thí dụ thiết kế mạch đếm 10 (N = 10).

$$2^{4-1} < 10 < 2^4 .$$

Vậy số tầng là 4

Có nhiều phương pháp thiết kế mạch đếm đồng bộ modulo-N.

Sau đây ta khảo sát hai phương pháp **:** dùng **hàm Chuyển** và **MARCUS**

✸ **Phương pháp dùng hàm Chuyển** (Transfer function)

Hàm Chuyển là hàm cho thấy có sự thay đổi trạng thái của FF. Mỗi loại FF có một hàm Chuyển riêng của nó.

Hàm Chuyển được định nghĩa như sau: hàm có trị 1 khi có sự thay đổi trạng thái của FF ($Q_+ \neq Q$) và trị 0 khi trạng thái FF không đổi ($Q_+ = Q$).

Chúng ta chỉ thiết kế mạch đếm dùng FF JK do đó ta chỉ xác định hàm Chuyển của loại FF này.

Bảng trạng thái của FF JK (Bảng 5.16)

C_K	J	K	Q	Q_+	H
↓	0	0	0	0	0
↓	0	0	1	1	0
↓	0	1	0	0	0
↓	0	1	1	0	1
↓	1	0	0	1	1
↓	1	0	1	1	0
↓	1	1	0	1	1
↓	1	1	1	0	1

Bảng 5.16

Dùng Bảng Karnaugh ta suy ra được biểu thức của H: $H = J\overline{Q} + KQ$

Để thiết kế mạch đếm cụ thể ta sẽ xác định hàm H cho từng FF trong mạch, so sánh với biểu thức của hàm H suy ra J, K của các FF. Dưới đây là một thí dụ.

Thiết kế mạch đếm 10 đồng bộ dùng FF JK

Bảng trạng thái của mạch đếm 10 và giá trị của các hàm H tương ứng:

C_K	Q_D	Q_C	Q_B	Q_A	Q_{D+}	Q_{C+}	Q_{B+}	Q_{A+}	H_D	H_C	H_B	H_A
1↓	0	0	0	0	0	0	0	1	0	0	0	1
2↓	0	0	0	1	0	0	1	0	0	0	1	1
3↓	0	0	1	0	0	0	1	1	0	0	0	1
4↓	0	0	1	1	0	1	0	0	0	1	1	1
5↓	0	1	0	0	0	1	0	1	0	0	0	1
6↓	0	1	0	1	0	1	1	0	0	0	1	1
7↓	0	1	1	0	0	1	1	1	0	0	0	1
8↓	0	1	1	1	1	0	0	0	1	1	1	1

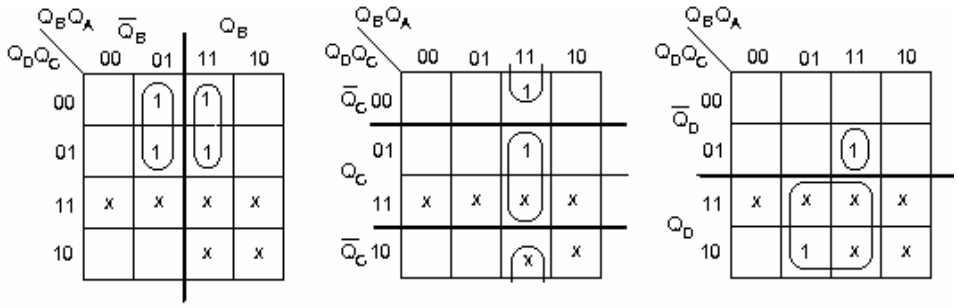
9↓	1	0	0	0	1	0	0	1	0	0	0	1
10↓	1	0	0	1	0	0	0	0	1	0	0	1

Bảng 5.17

Từ bảng 5.17, ta thấy:

$$H_A = 1 = Q_A + \overline{Q_A} \Rightarrow J_A = K_A = 1$$

Để xác định H_B, H_C và H_D ta phải vẽ bảng Karnaugh



$H_B = \overline{Q_D}Q_A\overline{Q_B} + \overline{Q_D}Q_AQ_B$
$\Rightarrow J_B = K_B = \overline{Q_D}Q_A$

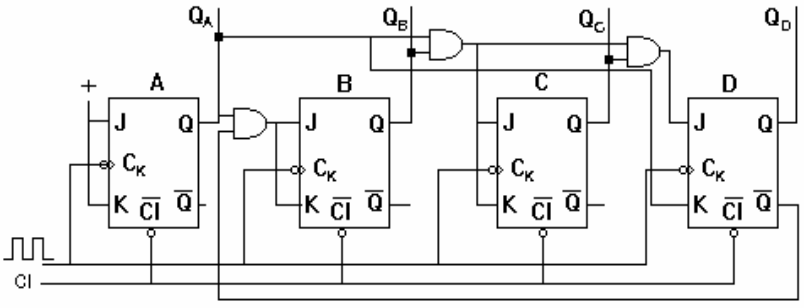
$H_C = Q_BQ_A\overline{Q_C} + Q_BQ_AQ_C$
$\Rightarrow J_C = K_C = Q_BQ_A$

$H_D = Q_CQ_BQ_A\overline{Q_D} + Q_AQ_D$
$\Rightarrow J_D = Q_CQ_BQ_A, K_D = Q_A$

(H 5.19)

Ghi chú: *Trong kết quả của hàm H ta muốn có chứa Q và $\overline{Q}$ tương ứng để suy ra ngay các trị J và K nên ta đã chia bảng Karnaugh ra làm 2 phần chứa Q và $\overline{Q}$ và nhóm riêng từng phần này.*

Từ các kết quả này, ta vẽ được mạch (H 5.20)



(H 5.20)

Bây giờ ta có thể kiểm tra xem nếu như vì một lý do nào đó, số đếm rơi vào các trạng thái không sử dụng (tương ứng với số từ 10 đến 15) thì khi có xung đồng hồ trạng thái tiếp theo sẽ như thế nào ? Mạch có quay về để đếm tiếp ?

Áp dụng các hàm chuyển có được, ứng với mỗi trạng thái Q của từng FF trong các tổ hợp không sử dụng, ta tìm trị H tương ứng rồi suy ra Q_+, ta được bảng kết quả sau:

C_K	Q_D	Q_C	Q_B	Q_A	H_D	H_C	H_B	H_A	Q_{D+}	Q_{C+}	Q_{B+}	Q_{A+}
↓	1	0	1	0	0	0	0	1	1	0	1	1
↓	1	0	1	1	1	1	0	1	0	1	1	0
↓	1	1	0	0	0	0	0	1	1	1	0	1
↓	1	1	0	1	1	0	0	1	0	1	0	0
↓	1	1	1	0	0	0	0	1	1	1	1	1
↓	1	1	1	1	0	1	0	1	0	0	1	0

Bảng 5.18

Từ bảng kết quả ta có kết luận:

- Khi ngã ra rơi vào trạng thái 10_{10} (1010), nó sẽ nhảy tiếp vào trạng thái 11_{10} (1011) rồi sau đó nhảy về 6_{10} (0110) (Dòng 1 và 2)
- Khi ngã ra rơi vào trạng thái 12_{10} (1100), nó sẽ nhảy tiếp vào trạng thái 13_{10} (11 01) rồi sau đó nhảy về 4_{10} (0100) (Dòng 3 và 4)
- Khi ngã ra rơi vào trạng thái 14_{10} (1110), nó sẽ nhảy tiếp vào trạng thái 15_{10} (1111) rồi sau đó nhảy về 2_{10} (0010) (Dòng 5 và 6).

Tóm lại, nếu có một sự cố xảy ra làm cho số đếm rơi vào các trạng thái không sử dụng thì sau 1 hoặc 2 số đếm nó tự động quay về một trong các số đếm từ 0 đến 9 rồi tiếp tục đếm bình thường.

✸ Phương pháp MARCUS

Phương pháp MARCUS cho phép xác định các biểu thức của J và K dựa vào sự thay đổi của Q_+ so với Q

Từ bảng trạng thái của FF JK (Bảng 5.7) ta có thể viết lại Bảng 5.19:

Q	Q_+	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

Bảng 5.19

Để thiết kế mạch, ta so sánh Q_+ và Q để có được bảng sự thật cho J, K của từng FF, sau đó xác định J và K.

Thí dụ thiết kế lại mạch đếm 10 bằng phương pháp MARCUS

Bảng sự thật cho J, K của từng FF

C_K	Q_D	Q_C	Q_B	Q_A	J_D	K_D	J_C	K_C	J_B	K_B	J_A	K_A
1↓	0	0	0	0	0	x	0	x	0	x	1	x
2↓	0	0	0	1	0	x	0	x	1	x	x	1
3↓	0	0	1	0	0	x	0	x	x	0	1	x
4↓	0	0	1	1	0	x	1	x	x	1	x	1
5↓	0	1	0	0	0	x	x	0	0	x	1	x
6↓	0	1	0	1	0	x	x	0	1	x	x	1
7↓	0	1	1	0	0	x	x	0	x	0	1	x
8↓	0	1	1	1	1	x	x	1	x	1	x	1
9↓	1	0	0	0	x	0	0	x	0	x	1	x
10↓	1	0	0	1	x	1	0	x	0	x	x	1

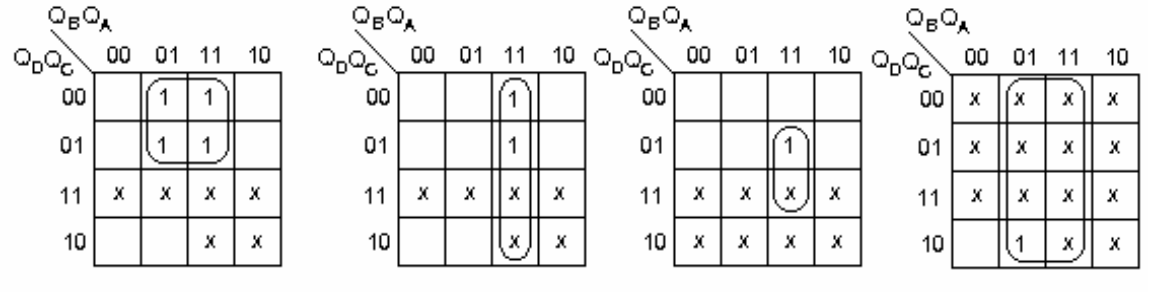
Bảng 5.20

Ghi chú: *Trong bảng 5.20, không có các cột cho Q_+, tuy nhiên ta có thể thấy ngay là dòng bên dưới chính là Q_+ của dòng bên trên, như vậy kết quả có được từ sự so sánh dòng trên và dòng ngay dưới nó.*

Ta thấy ngay $J_A = K_A = 1$

Dùng bảng Karnaugh để xác định các hàm còn lại

Nhận thấy các FF B và C có thể xác định chung cho J và K (cùng vị trí 1 và x), FF D được xác định J và K riêng



$J_B = K_B = \overline{Q_D}Q_A$ $J_C=K_C=Q_B.Q_A$ $J_D=Q_C.Q_B.Q_A$ $K_D=Q_A$

(H 5.21)

Ta được lại kết quả trên.

Trên thị trường có khá nhiều IC đếm:

- 4 bit BCD: 74160, 74162, 74190, 74192, 4192, 4510, 4518. . ..
- 4 bit nhị phân: 74161, 74163, 74191, 74193, 4193, 4516, 4520. . ..
- 8 bit nhị phân: 74269, 74579, 74779. . ..

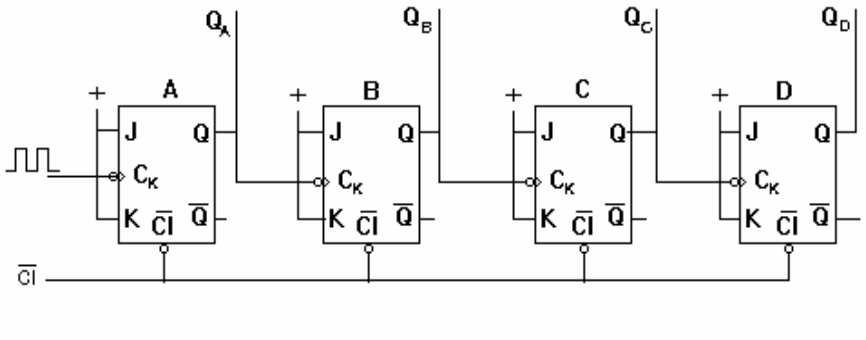
5.3.2 Mạch đếm không đồng bộ

Là các mạch đếm mà các FF không chịu tác động đồng thời của xung C_K.

Khi thiết kế mạch đếm không đồng bộ ta phải quan tâm tới **chiều tác động của xung đồng hồ C_K**.

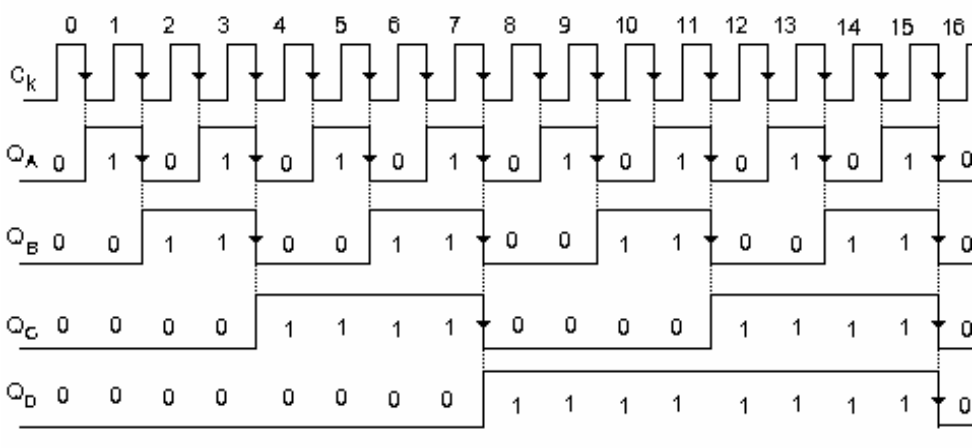
5.3.2.1. Mạch đếm không đồng bộ, n tầng, đếm lên (n=4):

Từ bảng trạng thái 5.14 của mạch đếm 4 bit, ta thấy nếu dùng FF JK tác động bởi **cạnh xuống của xung đồng hồ** thì có thể lấy ngã ra của tầng trước làm xung đồng hồ C_K cho tầng sau, với điều kiện các ngã vào JK của các FF đều được đưa lên mức cao. Ta được mạch đếm không đồng bộ, 4 bít, đếm lên (H 5.22).



(H 5.22)

(H 5.23) là dạng tín hiệu xung C_K và các ngã ra của các FF



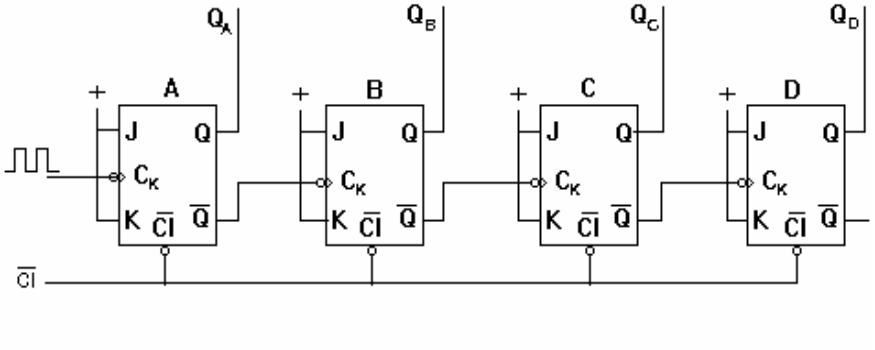
(H 5.23)

Tổ hợp các số tạo bởi các ngã ra các FF D, C, B, A là số nhị phân từ 0 đến 15

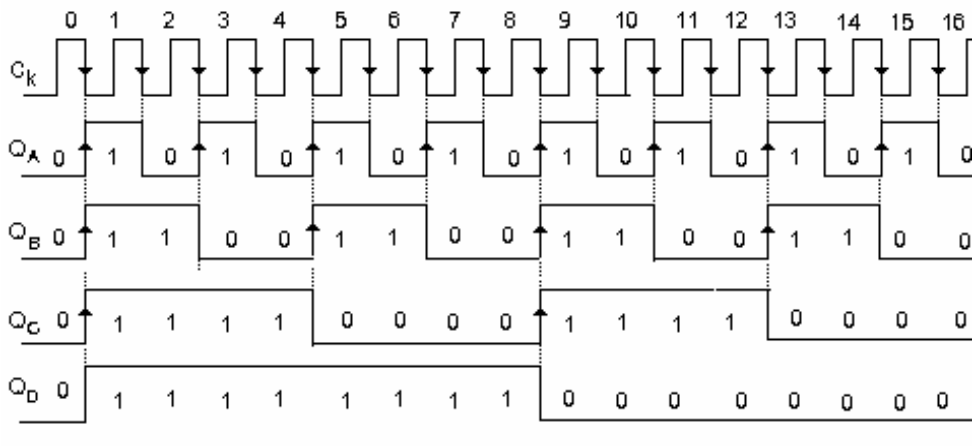
5.3.2.2. Mạch đếm không đồng bộ, n tầng, đếm xuống (n=4):

Để có mạch đếm xuống ta nối $\overline{Q}$ (thay vì Q) của tầng trước vào ngã vào C_K của tầng sau. (H 5.24) là mạch đếm xuống 4 tầng.

Dạng sóng ở ngã ra các FF và số đếm tương ứng cho ở (H 5.25)



(H 5.24)



(H 5.25)

Quan sát tín hiệu ra ở các Flipflop ta thấy sau mỗi FF tần số của tín hiệu ra giảm đi một nửa, nghĩa là:

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

$$f_{Q_B} = \frac{f_{Q_A}}{2} = \frac{f_{CK}}{2^2} = \frac{f_{CK}}{4}$$

$$f_{Q_C} = \frac{f_{Q_A}}{4} = \frac{f_{CK}}{2^3} = \frac{f_{CK}}{8}$$

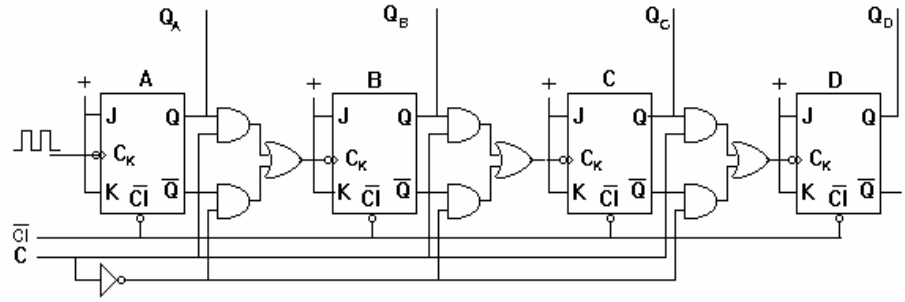
$$f_{Q_D} = \frac{f_{Q_A}}{8} = \frac{f_{CK}}{2^4} = \frac{f_{CK}}{16}$$

Như vậy xét về khía cạnh tần số, ta còn gọi mạch đếm là mạch chia tần.

5.3.2.3. Mạch đếm không đồng bộ, n tầng, đếm lên, xuống (n=4):

Để có mạch đếm lên hoặc đếm xuống người ta dùng các mạch đa hợp 2→1 với ngã vào điều khiển C chung để chọn Q hoặc $\overline{Q}$ của tầng trước nối vào C_K tầng sau tùy theo yêu cầu về cách đếm.

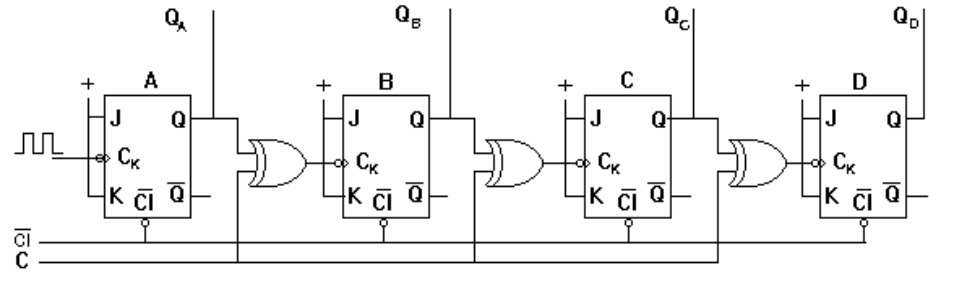
Trong (H 5.26) , khi C =1, Q nối vào C_K , mạch đếm lên và khi C = 0, $\overline{Q}$ nối vào C_K , mạch đếm xuống



c = 0 : đếm xuống c = 1 : đếm lên

(H 5.26)

Trên thực tế , để đơn giản, ta có thể thay đa hợp 2→1 bởi một cổng EX-OR, ngã điều khiển C nối vào một ngã vào cổng EX-OR, ngã vào còn lại nối với ngã ra Q của FF và ngã ra của cổng EX-OR nối vào ngã vào C_K của FF sau, mạch cũng đếm lên/xuống tùy vào C=0 hay C=1.



c = 1 : đếm xuống c = 0 : đếm lên

(H 5.27)

5.3.2.4. Mạch đếm không đồng bộ modulo - N (N=10)

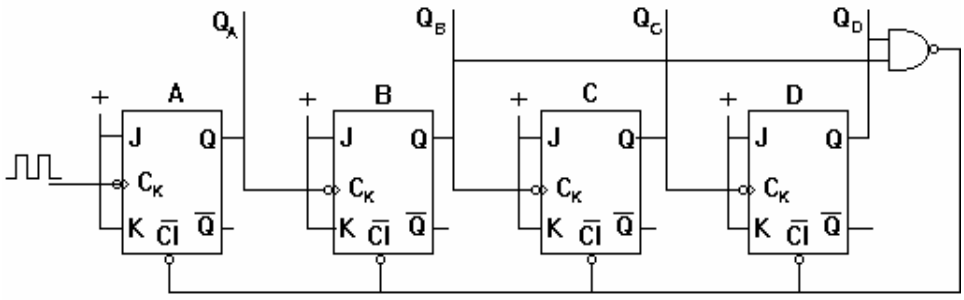
✸ **Kiểu Reset:**

Để thiết kế mạch đếm kiểu Reset, trước nhất người ta lập bảng trạng thái cho số đếm (Bảng 5.21)

Quan sát bảng 5.21 ta thấy ở xung thứ 10, nếu theo cách đếm 4 tầng thì Q_D và Q_B phải lên 1. Lợi dụng hai trạng thái này ta dùng một cổng NAND 2 ngã vào để đưa tín hiệu về xóa các FF, ta được mạch đếm ở (H 5.28).

Số xung C_K vào	Số Q_D	Nhị Q_C	Phân Q_B	Ra Q_A	Số thập phân tương ứng
Xóa	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	2
3	0	0	1	1	3
4	0	1	0	0	4
5	0	1	0	1	5
6	0	1	1	0	6
7	0	1	1	1	7
8	1	0	0	0	8
9	1	0	0	1	9
10	0(1)	0	0(1)	0	10

Bảng 5.21



(H 5.28)

Mạch đếm kiểu Reset có khuyết điểm như:
- Có một trạng thái trung gian trước khi đạt số đếm cuối cùng.
- Ngã vào Cl không được dùng cho chức năng xóa ban đầu.

✸ **Kiểu Preset**:

Trong kiểu Preset các ngã vào của các FF sẽ được đặt trước thế nào để khi mạch đếm đến trạng thái thứ N thì tất cả các FF tự động quay về không.

Để thiết kế mạch đếm không đồng bộ kiểu Preset, thường người ta làm như sau:

- Phân tích số đếm $N = 2^n.N'$ ($N'<N$) rồi kết hợp hai mạch đếm n bit và N'. Việc thiết kế rất đơn giản khi số $N' << N$
- Quan sát bảng trạng thái và kết hợp với phương pháp thiết kế mạch đếm đồng bộ (MARCUS hay hàm chuyển) để xác định JK của các FF.

Thí dụ, để thiết kế mạch đếm 10, ta phân tích 10=2x5 và ta chỉ cần thiết kế mạch đếm 5 rồi kết hợp với một FF (đếm 2)

Bảng trạng thái của mạch đếm 5.

Số xung C_K vào	Số Nhị Q_D	Phân Q_C	Ra Q_B	Số thập phân tương ứng
Xóa	0	0	0	0

1	0	0	1	1
2	0	1	0	2
3	0	1	1	3
4	1	0	0	4
5	0	0	0	0

Bảng 5.22

Giả sử dùng FF JK có xung C_K tác động cạnh xuống.

Từ bảng 5.21, ta thấy có thể dùng tín hiệu ngã ra FF B làm xung đồng hồ cho FF C và đưa J_C và K_C lên mức cao:

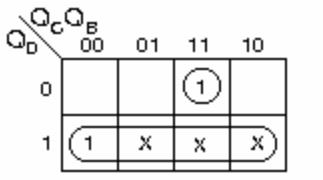
$$C_{KC}= Q_B. \; ; \qquad J_C=K_C=1$$

Các FF B và D sẽ dùng xung C_K của hệ thống và các ngã vào JK được xác định nhờ hàm chuyển:

C_K	Q_D	Q_C	Q_B	H_D	H_B
1↓	0	0	0	0	1
2↓	0	0	1	0	1
3↓	0	1	0	0	1
4↓	0	1	1	1	1
5↓	1	0	0	1	0
	0	0	0		

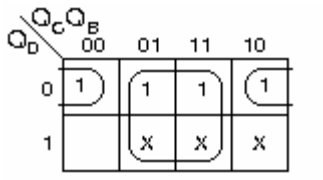
Bảng 5.23

Dùng bảng Karnaugh xác định H_D và H_B rồi suy ra các trị J, K của các FF.



$$H_D = Q_C.Q_B\overline{Q_D} + Q_D$$

$$\Rightarrow \quad J_D = Q_C.Q_B \;\; ; K_D=1$$



$$H_B = \overline{Q_D}\,\overline{Q_B} + Q_B$$

$$\Rightarrow J_B = \overline{Q_D} \; ; K_B=1$$

(H 5.29)

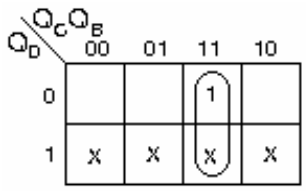
Có thể xác định J, K của các FF B và D bằng phương pháp MARCUS:

C_K	Q_D	Q_C	Q_B	J_D	K_D	J_B	K_B
1↓	0	0	0	0	x	1	x
2↓	0	0	1	0	x	x	1
3↓	0	1	0	0	x	1	x
4↓	0	1	1	1	x	x	1
5↓	1	0	0	x	1	0	x
	0	0	0				

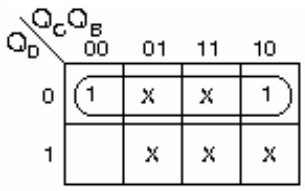
Bảng 5.24

Ta có ngay $K_D=K_B=1$

Dùng bảng Karnaugh xác định J_D và J_B



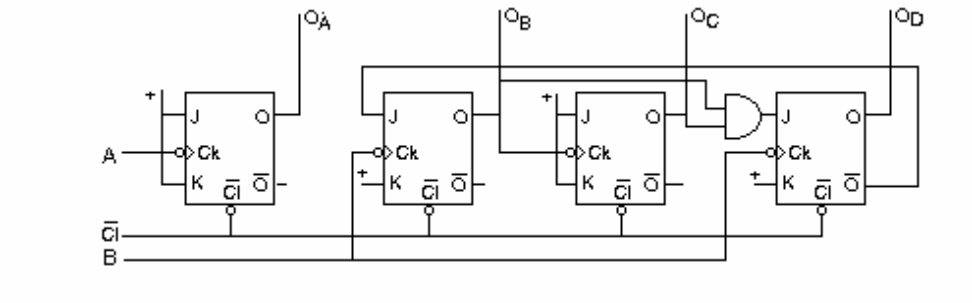
$J_D = Q_C \cdot Q_B$



$J_B = \overline{Q_D}$

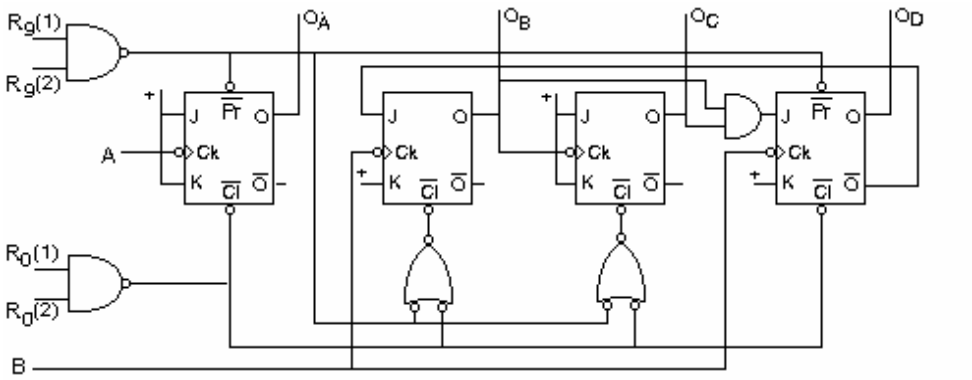
(H 5.30)

(H 5.31) là mạch đếm 10 thiết kế theo kiểu đếm 2x5 với mạch đếm 5 có được từ kết quả trên.



(H 5.31)

IC **7490** là IC đếm 10, có cấu tạo như mạch (H 5.31) thêm các ngã vào Reset 0 và Reset 9 có sơ đồ mạch (H 5.32)



(H 5.32)

Bảng 5.25 là bảng sự thật cho các ngã vào Reset

Reset Inputs				**Outputs**			
$R_0(1)$	$R_0(2)$	$R_9(1)$	$R_9(2)$	Q_D	Q_C	Q_B	Q_A
1	1	0	x	0	0	0	0
1	1	x	0	0	0	0	0
0	x	1	1	1	0	0	1
x	0	1	1	1	0	0	1
x	0	x	0	Đếm	Đếm	Đếm	Đếm
0	x	0	x	nt	nt	nt	nt
0	x	x	0	nt	nt	nt	nt
x	0	0	x	nt	nt	nt	nt

Bảng 5.25

Dùng IC 7490, có thể thực hiện một trong hai cách mắc:

- Mạch đếm 2x5: Nối Q_A vào ngã vào B, xung đếm (C_K) vào ngã vào A

✸ Mạch đếm 5x2: Nối Q_D vào ngã vào A, xung đếm (C_K) vào ngã vào B

Hai cách mắc cho kết quả số đếm khác nhau nhưng cùng một chu kỳ đếm 10. Tần số tín hiệu ở ngã ra sau cùng bằng 1/10 tần số xung C_K (nhưng dạng tín hiệu ra khác nhau).

Dưới đây là hai bảng trạng thái cho hai trường hợp nói trên.

Q_D	Q_C	Q_B	Q_A
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1

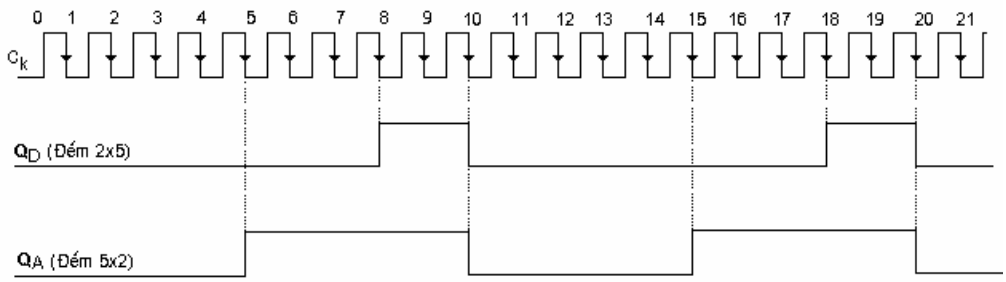
Bảng 5.26 : Đếm 2x5

Q_D	Q_C	Q_B	Q_A
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1

Bảng 5.27 : Đếm 5x2

(H 5.33) cho thấy dạng sóng ở các ngã ra của hai mạch cùng đếm 10 nhưng hai kiểu đếm khác nhau:

- Kiểu đếm 2x5 cho tín hiệu ra ở Q_D không đối xứng
- Kiểu đếm 5x2 cho tín hiệu ra ở Q_A đối xứng



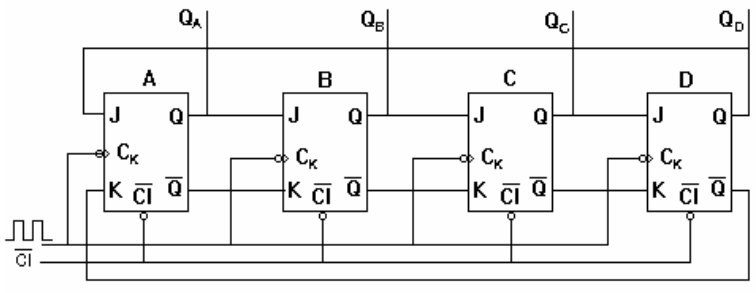
(H 5.33)

5.3.3 Mạch đếm vòng

Thực chất là mạch ghi dịch trong đó ta cho hồi tiếp từ một ngã ra nào đó về ngã vào để thực hiện một chu kỳ đếm. Tùy đường hồi tiếp mà ta có các chu kỳ đếm khác nhau

Sau đây ta khảo sát vài loại mạch đếm vòng phổ biến.

5.3.3.1. Hồi tiếp từ Q_D về J_A và $\overline{Q}_D$ về K_A



(H 5.34)

Đối với mạch này, sự đếm vòng chỉ thấy được khi có đặt trước ngã ra

- Đặt trước Q_A =1, ta được kết quả như bảng 5.28.

C_K	Q_D	Q_C	Q_B	Q_A	Số TP
Preset	0	0	0	1	1
1↓	0	0	1	0	2
2↓	0	1	0	0	4
3↓	1	0	0	0	8
4↓	0	0	0	1	1
:	:	:	:	:	:

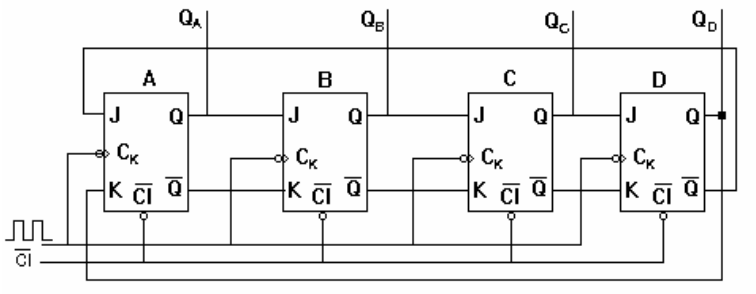
Bảng 5.28

Nếu đặt trước $Q_A = Q_B = 1$ ta có bảng 5.29

C_K	Q_D	Q_C	Q_B	Q_A	Số TP
Preset	0	0	1	1	3
1↓	0	1	1	0	6
2↓	1	1	0	0	12
3↓	1	0	0	1	9
4↓	0	0	1	1	3
:	:	:	:	:	:

Bảng 5.29

5.3.3.2. Hồi tiếp từ $\overline{Q}_D$ về J_A và Q_D về K_A (H 5.35)



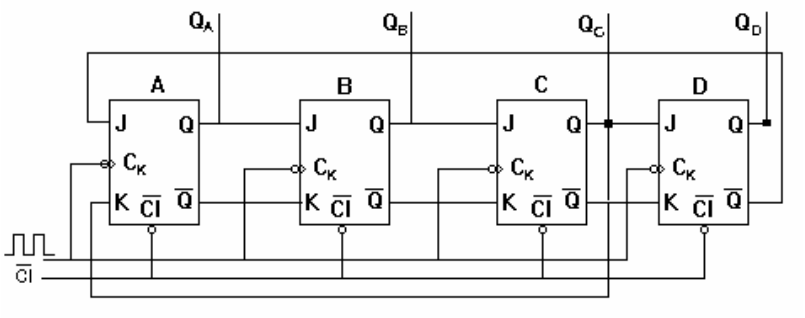
(H 5.35)

Mạch này còn có tên là mạch đếm Johnson. Mạch có một chu kỳ đếm mặc nhiên mà không cần đặt trước và nếu có đặt trước, mạch sẽ cho các chu kỳ khác nhau tùy vào tổ hợp đặt trước đó. Bảng 5.30 là chu kỳ đếm mặc nhiên.

C_K	Q_D	Q_C	Q_B	Q_A	Số TP
Preset	0	0	0	0	0
1↓	0	0	0	1	1
2↓	0	0	1	1	3
3↓	0	1	1	1	7
4↓	1	1	1	1	15
5↓	1	1	1	0	14
6↓	1	1	0	0	12
7↓	1	0	0	0	8
8↓	0	0	0	0	0

Bảng 5.30

5.3.3.3. Hồi tiếp từ $\overline{Q}_D$ về J_A và Q_C về K_A (H 5.36)



(H 5.36)

C_K	Q_D	Q_C	Q_B	Q_A	Số TP
Preset	0	0	0	0	0
1↓	0	0	0	1	1
2↓	0	0	1	1	3
3↓	0	1	1	1	7
4↓	1	1	1	0	14
5↓	1	1	0	0	12
6↓	1	0	0	0	8
7↓	0	0	0	0	0

Bảng 5.31

Vài thí dụ thiết kế mạch đếm

1. Dùng FF JK thiết kế mạch đếm 6, đồng bộ

Bảng trạng thái và hàm chuyển mạch đếm 6:

N	Q_A	Q_B	Q_C	Q_{A+}	Q_{B+}	Q_{C+}	H_A	H_B	H_C
0	0	0	0	0	0	1	0	0	1
1	0	0	1	0	1	0	0	1	1
2	0	1	0	0	1	1	0	0	1
3	0	1	1	1	0	0	1	1	1
4	1	0	0	1	0	1	0	0	1
5	1	0	1	0	0	0	1	0	1

Bảng 5.32

$H_C = 1 \Rightarrow J_C = K_C = 1$

Xác định J_A, K_A, J_B, K_B

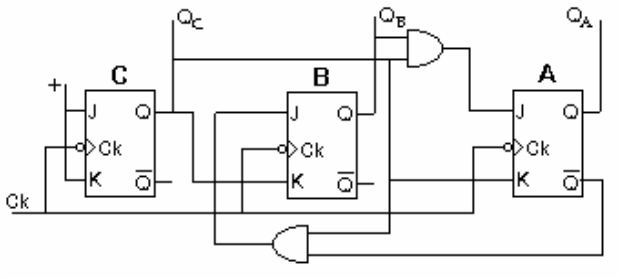
Bảng Karnaugh cho hai hàm chuyển H_A & H_B

(H 5.37)

$H_A = Q_BQ_C\overline{Q}_A + Q_CQ_A$ $\quad$ $H_B = \overline{Q}_A Q_C \overline{Q}_B + Q_CQ_B$

$\Rightarrow J_A = Q_BQ_C$; $K_A = Q_C$ $\quad$ $\Rightarrow J_B = \overline{Q}_A Q_C$; $K_B = Q_C$

Mạch:



(H 5.38)

2. Thiết kế mạch đếm 7 không đồng bộ, dùng FF JK có ngã vào xung đồng hồ tác động bởi cạnh lên của C_K.

Bảng trạng thái

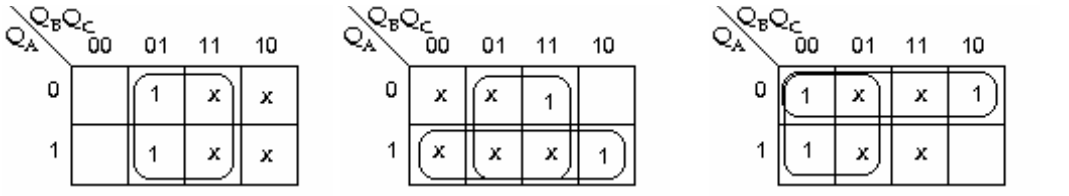
N	Q_A	Q_B	Q_C	J_B	K_B	J_C	K_C
0↑	0	0	0	0	x	1	x
1↑	0	0	1	1	x	x	1
2↑	0	1	0	x	0	1	x
3↑	0	1	1	x	1	x	1
4↑	1	0	0	0	x	1	x
5↑	1	0	1	1	x	x	1
6↑	1	1	0	x	1	0	x
	0	0	0				

Bảng 5.33

Nhận xét bảng trạng thái ta thấy mỗi lần Q_B thay đổi từ 1 xuống 0 thì Q_A đổi trạng thái, mà FF có xung đồng hồ tác động bởi cạnh lên nên ta có thể lấy $\overline{Q}_B$ làm xung đồng hồ cho FFA và $J_A=K_A=1$.

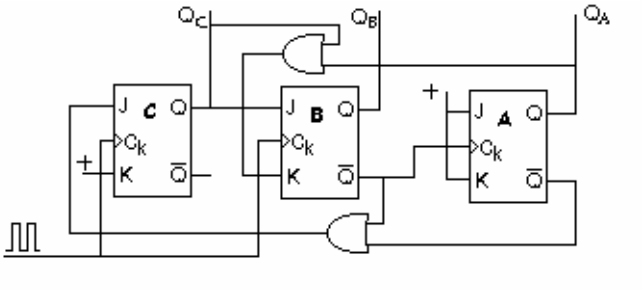
FF B và FFC sẽ dùng xung đồng hồ hệ thống, dùng phương pháp MARCUS để xác định J & K của các FF này.

Ta thấy ngay $K_C=1$



(H 5.39)

$J_B=Q_C$ $K_B=Q_A+Q_C$ $J_C=\overline{Q}_A+\overline{Q}_B$



(H 5.40)

BÀI TẬP

1. Thiết kế bộ đếm đồng bộ có dãy đếm sau: 000, 010, 101, 110 và lặp lại.

2. Làm lại bài 1. Thêm điều kiện các trạng thái không sử dụng 001, 011, 100 và 111 phải luôn luôn nhảy về 000 ở xung đồng hồ kế tiếp.

3. Thiết kế bộ đếm đồng bộ dùng FF-JK với dãy đếm sau: 000, 001, 011, 010, 110,111, 101, 100, 000 . . .

4.

a. Thiết kế một mạch đếm đồng bộ dùng FF-JK tác động cạnh xuống, có dãy đếm như sau: 000, 001, 011, 111, 110, 100, 001. . . Những trạng thái không sử dụng được đưa về trạng thái 000 ở xung đồng hồ kế tiếp. Vẽ sơ đồ mạch.

b. Mắc nối tiếp một bộ đếm 2 (Dùng FF-JK, tác động cạnh xuống) với bộ đếm đã được thiết kế ở câu a. Vẽ dạng sóng ở các ngã ra của bộ đếm giả sử trạng thái ban đầu của các ngã ra đều bằng 0. Xác định dãy đếm của mạch.

5. Thiết kế mạch đếm đồng bộ modulo-12 dùng FF JK.

Dùng ngã ra mạch đếm để điều khiển hệ thống đèn giao thông:

- Đèn xanh cháy trong 40 s
- Đèn vàng cháy trong 20s
- Đèn đỏ cháy trong 40s
- Đèn vàng và đỏ cùng cháy trong 20s. Chu kỳ lặp lại

Cho chu kỳ xung đồng hồ là 10s.

6. Thiết kế mạch đếm đồng bộ dùng FF JK có ngã vào điều khiển X:

- Khi X=0 mạch đếm theo thứ tự 0, 2, 4, 6 rồi trở về 0
- Khi X=1 mạch đếm 0, 6, 4, 2 rồi trở về 0.

Các trạng thái không sử dụng trong hai lần đếm đều trở về 0 khi có xung đồng hồ

CHƯƠNG 6: MẠCH LÀM TOÁN

- SỐ BÙ
- PHÉP TRỪ SỐ NHỊ PHÂN DÙNG SỐ BÙ 1
- PHÉP TRỪ SỐ NHỊ PHÂN DÙNG SỐ BÙ 2
- PHÉP TOÁN VỚI SỐ CÓ DẤU
- MẠCH CỘNG
  - Bán phần
  - Toàn phần
  - Cộng hai số nhiều bít
- MẠCH TRỪ
  - Bán phần
  - Toàn phần
  - Trừ hai số nhiều bit
  - Cộng & trừ hai số nhiều bit trong một mạch
- MẠCH NHÂN
  - Mạch nhân cơ bản
  - Mạch nhân nối tiếp - song song đơn giản
- MẠCH CHIA
  - Mạch chia phục hồi số bị chia
  - Mạch chia không phục hồi số bị chia

6.1 Số bù

Cho số dương N, n bit, các số bù của N được định nghĩa:

Số bù 2: $(N)_2 = 2^n - N$ (số 2^n gồm bit 1 và n bit 0 theo sau)

Số bù 1: $(N)_1 = (N)_2 - 1 = 2^n - N - 1$

Thí dụ 1: N = 1010

Số bù 2 của N là $(N)_2$ = là 10000 - 1010 = 0110

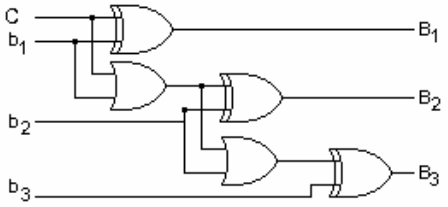
Và số bù 1 của N là $(N)_1$ = 0110 - 1 = 0101

Thí dụ 2: N = 110010101100 $\Rightarrow (N)_2$ = 001101010100 và $(N)_1$ = 001101010011

Nhận xét:

- Để có số bù 2 của một số, bắt đầu từ bit LSB (tận cùng bên phải) đi ngược về bên trái, các bit sẽ giữ nguyên cho đến lúc gặp bit 1 đầu tiên, sau đó đảo tất cả các bit còn lại.
- Để có số bù 1 của một số, ta đảo tất cả các bit của số đó.

Từ các nhận xét trên ta có thể thực hiện một mạch tạo số bù 1 và 2 sau đây:



(H 6.1)

- Khi C=1, B là số bù 1 của b (B_1 và b_1 là bit LSB)

- Khi C=0, B là số bù 2 của b.

Thật vậy, các biểu thức logic của B theo b và C là:

$$B_1 = b_1 \oplus C$$
$$B_2 = b_2 \oplus (C + b_1)$$
$$B_3 = b_3 \oplus (C + b_1 + b_2)$$

- Khi C=1 , các ngã ra cổng OR luôn bằng 1, các cổng EX - OR luôn có một ngã vào bằng 1 nên ngã ra là đảo của ngã vào còn lại, ta được:

$$B_1 = b_1 \oplus 1 = \overline{b_1}$$
$$B_2 = b_2 \oplus (1 + b_1) = b_2 \oplus 1 = \overline{b_2}$$
$$B_3 = b_3 \oplus (1 + b_1 + b_2) = b_3 \oplus 1 = \overline{b_3}$$

- Khi C=0

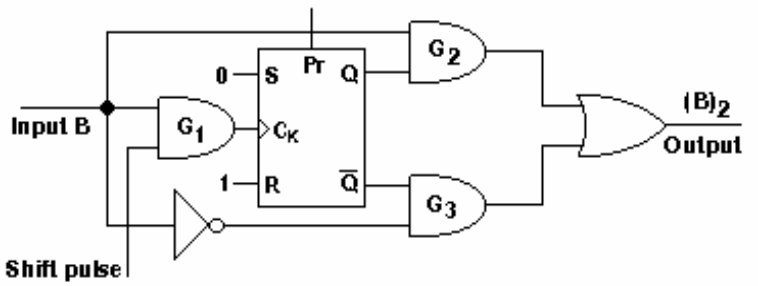
$$B_1 = b_1 \oplus 0 = b_1$$

$$B_2 = b_2 \oplus (0 + b_1) = b_2 \oplus b_1$$
$= b_2$ nếu b_1=0 và $\overline{b_2}$ nếu $b_1 = 1$

$$B_3 = b_3 \oplus (0 + b_1 + b_2) = b_3 \oplus (b_1 + b_2)$$
$= b_3$ nếu b_1 và b_2 đều =0
$= \overline{b_3}$ nếu (b_1 và/hoặc $b_2 = 1$)

Như vậy tất cả các bit sau bit 1 thứ nhất tính từ bit LSB đều bị đảo và B chính là số bù 2 của b

Chúng ta cũng có thể thiết kế mạch tạo số bù hai bằng cách dùng FF RS, có ngã vào R, S tác động mức cao, kết hợp với các cổng logic như (H 6.2). Mạch này dùng khá tiện lợi khi cần thực hiện bài toán cộng và trừ nhiều bit kiểu nối tiếp.



(H 6.2)

Bắt đầu, Preset mạch để ngã ra Q = 1, cổng G_3 đóng, G_2 mở, cho số B đi qua mà không bị đảo cho đến khi có bit 1 đầu tiên đến, cổng G_1 mở cho xung đồng hồ đi qua, FF RS được reset, Q = 0, $\overline{Q}$ = 1, G_2 đóng, G_3 mở, số B đi qua cổng G_2 và bị đảo. Ở ngã ra được số bù 2 của B.

6.2 Phép trừ số nhị phân dùng số bù 1:

Cho hai số dương A và B có n bit (nếu số bit khác nhau, ta thêm số 0 vào , mà không làm thay đổi trị, để cả hai có cùng số bit)

a/ - A≤B

Kết quả A-B là số 0 hoặc âm, phép tính được thực hiện như sau:
Tính A - B:

$$A - B = A-B+2^n-1-2^n+1$$
$$= A+(2^n -B -1) - 2^n+1$$
$$= A+(B)_1 - 2^n+1$$
$$= - \{2^n - [A+(B)_1] -1\}$$
$$= - [A+(B)_1]_1$$

Vậy A-B có được bằng cách cộng số bù 1 của B vào A rồi lấy bù 1 của tổng và thêm dấu trừ . Như vậy để thực hiện phép tính trừ ta chỉ cần dùng phép cộng và phép đảo

Thí dụ 3 : Tính 1001 - 11010 dùng số bù 1
Ta có A = 01001 (thêm số 0 vào để có 5 bit như số B
B = 11010 $\Rightarrow (B)_1 = 00101$

$$A-B = - [A+(B)_1]_1 = - (01001+00101) =- (01110)_1$$
$$= - (10001)$$

Trong hệ thập phân, đây là bài toán $9_{10} - 26_{10} = -17_{10}$
Để thấy dấu trừ được nhận ra như thế nào, ta viết lại phép toán:

	A =	01001
+	$(B)_1$ =	00101
số tràn →	0	01110

Không có số tràn (hay số tràn =0) là dấu hiệu của kết quả âm (hoặc =0) và ta phải lấy bù 1, thêm dấu trừ để đọc kết quả cuối cùng: $(01110)_1 = - 10001$

Thí dụ 4: Tính 10110 - 10110
A = 10110 và B = 10110 $\Rightarrow (B)_1 = 01001$

	A =	10110
+	$(B)_1$ =	01001
số tràn →	0	11111
bù 1 của kq →		00000

Trong phép cộng đầu tiên không có số tràn, kết quả xem như số âm (hoặc =0) lấy bù 1 của kết quả ta được A-B =00000.

b/ - A >B

Kết quả A-B là số dương, phép tính được thực hiện theo qui tắc sau:

Cộng A với $(B)_1$ rồi thêm 1 và không quan tâm tới số nhớ cuối cùng

Thí dụ 5: Tính 110101 - 100110 dùng số bù 1
A = 110101 và B = 100110 $\Rightarrow (B)_1 = 011001$

	A =	110101
+	$(B)_1$ =	011001
	1	001110
		+ 1
số tràn →	1	001111

Bỏ qua số nhớ cuối cùng, ta được kết quả A-B =001111.
Trong hệ thập phân đó là bài toán $53_{10} - 38_{10} = 15_{10}$.

Trong phép tính có số tràn chứng tỏ kết quả là số dương. Số 1 cộng thêm vào xem như lấy từ số nhớ đem qua.

Tóm lại, để thực hiện bài toán trừ, A-B, ta cộng A với bù 1 của B. Dựa vào sự có mặt hay không của số tràn mà có biện pháp xử lý kết quả:

- Nếu số tràn =0, kết quả là số âm (hoặc =0) , ta phải lấy bù 1 của kết quả và thêm dấu - để đọc.
- Nếu số tràn =1, ta cộng thêm 1 vào để có kết quả cuối cùng (bỏ qua bit tràn) là một số dương.

6.3 phép trừ số nhị phân dùng số bù 2:

Phép toán dùng số bù 1 có một bất tiện là ta phải thêm bài toán cộng 1 vào, để tránh việc này ta dùng phép toán với số bù 2

Cho hai số dương A và B có n bit

a/ - A<B

Tính A-B:

$A-B = A-B+2^n-2^n$

$= A+(2^n - B\ ) - 2^n$

$= A+(B)_2 - 2^n$

$= - \{2^n - [A+(B)_2] \}$

$= - [A+(B)_2]_2$

Vậy A-B có được bằng cách cộng số bù 2 của B vào A rồi lấy bù 2 của tổng và thêm dấu trừ. Như vậy ta đã chuyển phép tính trừ thành phép cộng

Thí dụ 6: Tính 1001 - 11010 dùng số bù 2

Ta có A = 01001 (thêm số 0 vào để có 5 bit như số B)

$B = 11010 \Rightarrow (N_2)_2 = 00110$

$A-B = - [A+(B)_2]_2 = - (01001+00110) =- (01111)_2$

$= - (10001)$

Ta được lại kết quả trên

Để thấy dấu trừ được nhận ra như thế nào, ta viết lại phép toán:

	A =	01001
+	$(B)_2$ =	00110
số tràn →	0	01111

Không có số tràn là dấu hiệu của kết quả âm và ta phải lấy bù 2, thêm dấu trừ để đọc kết quả cuối cùng: $(01111)_2 = - 10001$

b/ - A ≥B

Kết quả A-B là 0 hoặc số dương, phép tính được thực hiện theo qui tắc sau:

Cộng A với $(B)_2$ và không quan tâm tới số nhớ ở vị trí 2^n

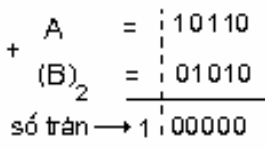
Thí dụ 7 : Tính 110101 - 100110 dùng số bù 2

A = 110101 và B = 100110 $\Rightarrow (B)_2 = 011010$

	A =	110101
+	$(B)_2$ =	011010
số tràn →	1	001111

Có số tràn, kết quả là số dương. Bỏ qua số tràn và đọc ngay kết quả mà không phải biến đổi: $001111 = 15_{10}$

Thí dụ 8 : Tính 10110 - 10110
A = 10110 và B = 10110 $\Rightarrow (B)_2 = 01010$



Bỏ qua số tràn ta được A-B =00000.

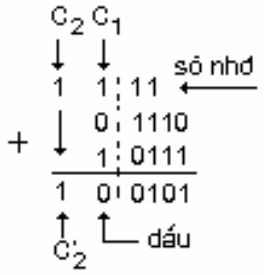
6.4 Phép toán với số có dấu

Cho tới giờ chúng ta thực hiện các phép toán với số không dấu và đôi khi xuất hiện dấu trừ trong kết quả. Trong máy tính, điều này có thể khắc phục được bằng cách dùng số có dấu.

Với qui ước số dương có bit dấu là 0 và số âm có dấu là 1

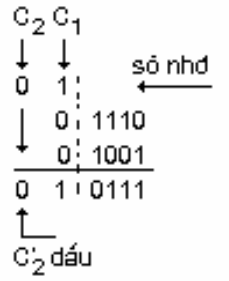
Thí dụ 9: $+10_{10} = 01010$ $+15_{10} = 01111$ $+23_{10} = 010111$
$-10_{10} = 10110$ $- 15_{10} = 10001$ $- 23_{10} = 101001$
Có thể thấy rằng số âm của một số là số bù 2 của nó kể cả bit dấu.
Với cách biểu diễn số có dấu, phép toán trừ trở thành phép toán cộng:
A-B = A+(-B)

Thí dụ 10: Tính A-B =01110 - 01001; B = 01001 = $+9_{10} \Rightarrow - 9_{10} = 10111$

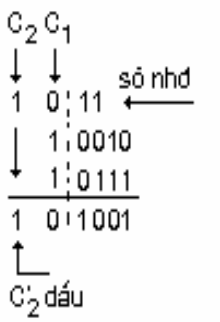


Bit dấu =0 chỉ kết quả dương, bỏ bit tràn C'_2.
Vậy A-B = 00101 $[(+14_{10})-(+9_{10})] = +5_{10}$
Nếu A hoặc B đều dương hoặc âm , kết quả có thể cần thêm một bit do tràn số. Trong trường hợp này bit tràn đầu tiên thuộc kết quả và C'_2 là bit dấu

Thí dụ 11: Tính A+B với A = 01110 $(+14_{10})$ và B = 01001 $(+9_{10})$
Kết quả là 010111 = $+23_{10}$ với $C'_2 = 0$ là bit dấu



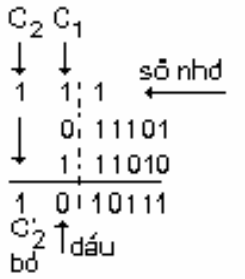
Thí dụ 12: Tính A-B với A =10010 (-14_{10}) và B =01001 ($+9_{10}$)



Một lần nữa C'$_2$ chỉ bit dấu. Kết quả là 101001 = -23_{10} (010111 = 23_{10})

Từ các kết quả trên, ta rút ra qui tắc sau đây:
Nếu $C_1 = C_2$ thì C'$_2$ là bit tràn, bỏ đi và nếu $C_1 \neq C_2$ thì C'$_2$ là bit dấu.

Thí dụ 13: Tính A-B với A =011101 ($+29_{10}$) và B =0110 ($+6_{10}$)
B = 000110 = $+6_{10}$ $\Rightarrow$ - 6_{10} = 111010



Ghi chú: - Trong tất cả trường hợp, ta luôn luôn thực hiện phép cộng do đó có thể bỏ qua phép trừ

- Khi cộng hai số hạng cùng dấu thì có thể xảy ra hiện tượng tràn, lúc đó bit dấu dời về bên trái một bit. Trong các trường hợp khác thì dấu của kết quả ở cùng vị trí với dấu của các số hạng

- Ngoài ra kết quả còn được xử lý tùy vào kết quả so sánh sự khác nhau của hai số nhớ C_1 và C_2 (nhờ một cổng EX-OR).

6.5 Mạch cộng nhị phân:

6.5.1 Mạch cộng bán phần (Half adder, HA):

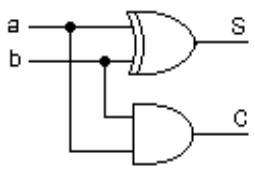
Là mạch cộng hai số 1 bit

vào		ra	
a	b	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

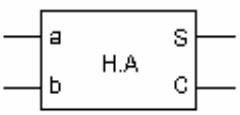
Bảng sự thật

$S = a \oplus b$
$C = a.b$

kết quả



Mạch



Ký hiệu

(H 6.3)

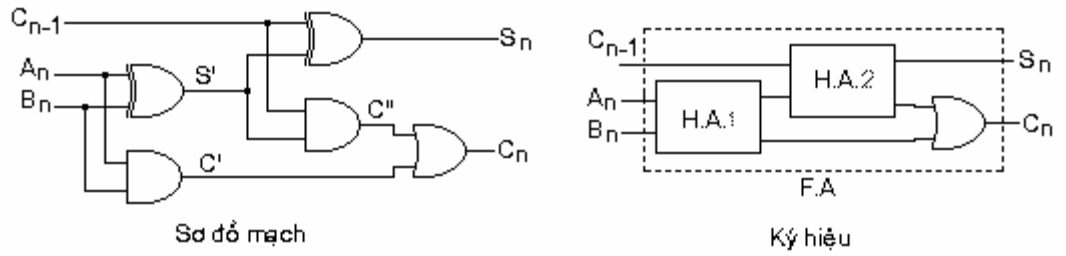
6.5.2 Mạch cộng toàn phần (Full adder,FA) :

Là mạch cộng hai bit ở cùng vị trí trong hai số nhị phân nhiều bit, nói cách khác, đây là mạch cộng hai bit , giả sử thứ n, và bit nhớ có được từ phép cộng hai bit thứ n-1 của hai số nhị phân đó. Ta có bảng sự thật

C_{n-1}		A_n	S_n	C_n
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Dùng bảng Karnaugh ta xác định được S_n và C_n như sau:

$$S_n = C_{n-1} \oplus (A_n \oplus B_n)$$
$$C_n = A_n B_n + C_{n-1}(A_n \oplus B_n)$$



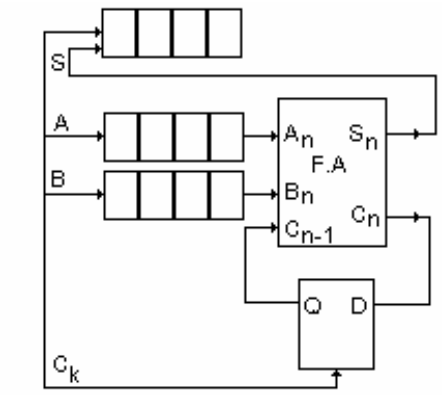
(H 6.4)

Có thể thấy một mạch cộng toàn phần gồm hai mạch cộng bán phần và một cổng OR

6.6 Cộng hai số nhị phân nhiều bit:

6.6.1 Cộng nối tiếp

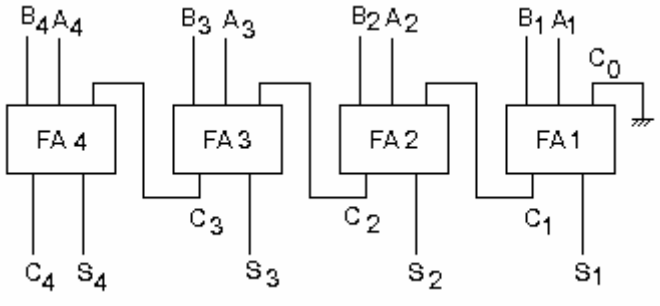
Trong cách cộng nối tiếp, người ta dùng các ghi dịch để chuyển các bit vào một mạch cộng toàn phần duy nhất, số nhớ từ ngã ra C_n được làm trễ một bit nhờ FF D và đưa vào ngã vào C_{n-1}. Như vậy tốc độ của phép cộng tùy thuộc vào tần số xung C_K và số bit phải thực hiện.



(H 6.5)

6.6.2 Cộng song song

Trong cách cộng song song, các bit được đưa đồng thời vào các mạch cộng toàn phần và số nhớ của kết quả ở bit thấp được đưa lên bit cao hơn (H 6.6).



(H 6.6)

Chính vì phải chờ số nhớ mà tốc độ cộng còn hạn chế. Muốn nâng tốc độ cộng lên, người ta thực hiện phép cộng song song định trước số nhớ.

6.6.3 Mạch cộng song song định trước số nhớ

Để tăng tốc độ của mạch cộng song song, người ta tạo trước các số nhớ để đưa đồng thời vào mạch cộng

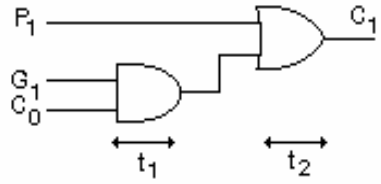
Từ biểu thức xác định số nhớ

$$C_n = A_nB_n + C_{n-1}(A_n \oplus B_n)$$

Đặt $P_n = A_nB_n$ và $G_n = A_n \oplus B_n$

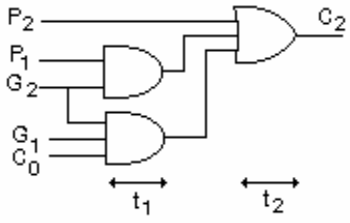
Ta xác định được C_1, C_2, C_3 .... như sau:

$C_1 = P_1 + C_0G_1$



$C_2 = P_2 + C_1G_2$

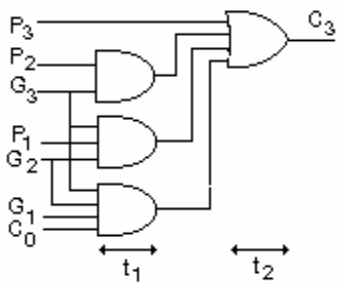
$C_2 = P_2 + P_1G_2 + C_0G_1G_2$



$C_3 = P_3 + C_2G_3$

$C_2 = P_2 + P_1G_2 + C_0G_1G_2$

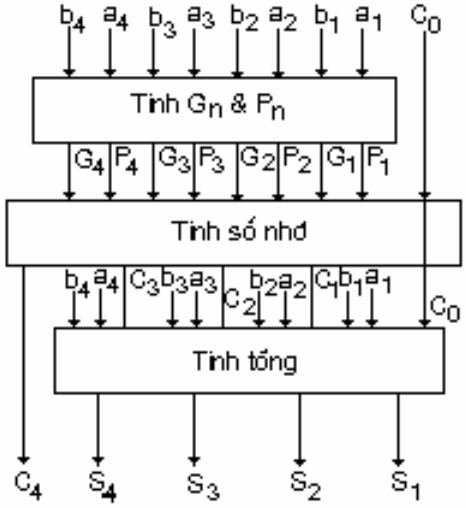
$C_3 = P_3 + P_2G_3 + P_1G_2G_3 + C_0G_1G_2G_3$



(H 6.7)

Nhận thấy thời gian tính số nhớ giống nhau ở các tầng và bằng t_1+t_2 . t_1 là thời gian truyền đồng thời qua các cổng AND và t_2 là thời gian truyền qua cổng OR.

Sơ đồ khối mạch cộng song song định trước số nhớ:



(H 6.8)

Trên thị trường hiện có IC 7483 (tương đương 4008 của CMOS) là IC cộng 4 bit theo kiểu định trước số nhớ.

6.6.4 Cộng hai số BCD

Trên thị trường có các IC cộng số nhị phân, trong lúc trên thực tế nhiều khi chúng ta cần cộng các số BCD để cho kết quả là số BCD.

Chúng ta tìm cách dùng IC 7483 (4008) để cộng hai số BCD

Hai số BCD có trị từ 0_{10} đến 9_{10} khi cộng lại cho kết quả từ 0_{10} đến 18_{10}. Để đọc được kết quả dạng BCD ta phải hiệu chỉnh kết quả có được từ mạch cộng nhị phân.

Dưới đây là kết quả tương đương giữa 3 loại mã: thập phân, nhị phân và BCD

	Phân Nhị					**B C D**					**BCD đọc**
TP	S'=C'$_4$				S'$_1$	S =C$_4$	S$_4$	S$_3$	S$_2$	S$_1$	**theo NP**
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	1	1
2	0	0	0	1	0	0	0	0	1	0	2
3	0	0	0	1	1	0	0	0	1	1	3
4	0	0	1	0	0	0	0	1	0	0	4
5	0	0	1	0	1	0	0	1	0	1	5
6	0	0	1	1	0	0	0	1	1	0	6
7	0	0	1	1	1	0	0	1	1	1	7
8	0	1	0	0	0	0	1	0	0	0	8
9	0	1	0	0	1	0	1	0	0	1	9
10	0	1	0	1	0	1	0	0	0	0	16
11	0	1	0	1	1	1	0	0	0	1	17
12	0	1	1	0	0	1	0	0	1	0	18
13	0	1	1	0	1	1	0	0	1	1	19
14	0	1	1	1	0	1	0	1	0	0	20
15	0	1	1	1	1	1	0	1	0	1	21
16	1	0	0	0	0	1	0	1	1	0	22
17	1	0	0	0	1	1	0	1	1	1	23

18	1	0	0	1	0	1	1	0	0	0	24

Nhận thấy:

- Khi kết quả <10 mã nhị phân và BCD hoàn toàn giống nhau
- Khi kết quả ≥10 để có được mã BCD ta phải cộng thêm 6 cho mã nhị phân

Để giải quyết vấn đề hiệu chỉnh này trước tiên ta sẽ thực hiện một mạch phát hiện kết quả trung gian của mạch cộng hai số nhị phân 4 bit. Mạch này nhận vào kết quả trung gian của phép cộng 2 số nhị phân 4 bit và cho ở ngã ra Y = 1 khi kết quả này ≥10, ngược lại, Y=0.

Bảng sự thật

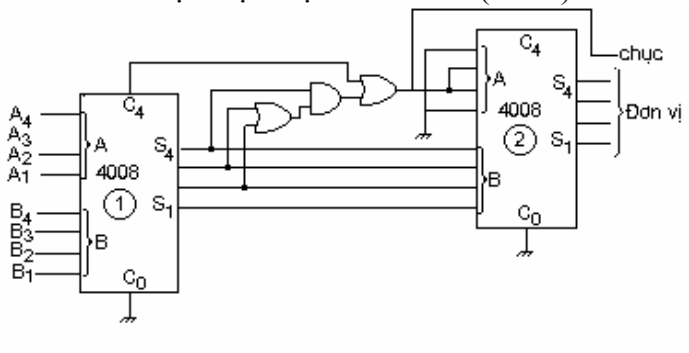
TP	C'_4			S'_2	Y
0-1	0	0	0	0	0
2-3	0	0	0	1	0
4-5	0	0	1	0	0
6-7	0	0	1	1	0
8-9	0	1	0	0	0
10-11	0	1	0	1	1
12-13	0	1	1	0	1
14-15	0	1	1	1	1
16-17	1	0	0	0	1
18	1	0	0	1	1

Ta không dùng ngã vào S'_1 vì từng cặp trị có $C'_4\,S'_4\,S'_3\,S'_2$ giống nhau thì $S'_1 = 0$ và $S'_1 = 1$

Dùng bảng Karnaugh xác định được Y

$$Y = C'_4 + S'_4\,(S'_3 + S'_2)$$

Và mạch cộng hai số BCD được thực hiện theo sơ đồ (H 6.9)



(H 6.9)

Vận hành:

- IC thứ nhất cho kết quả trung gian của phép cộng hai số nhị phân.
- IC thứ hai dùng hiệu chỉnh để có kết quả là số BCD:
- Khi kết quả < 10, IC 2 nhận ở ngã vào B số 0000 (do Y=0) nên kết quả không thay đổi.
- Khi kết quả trung gian ≥ 10, IC 2 nhận ở ngã vào B số $0110_2 = 6_{10}$ (do Y=1) và kết quả được hiệu chỉnh như đã nói trên.

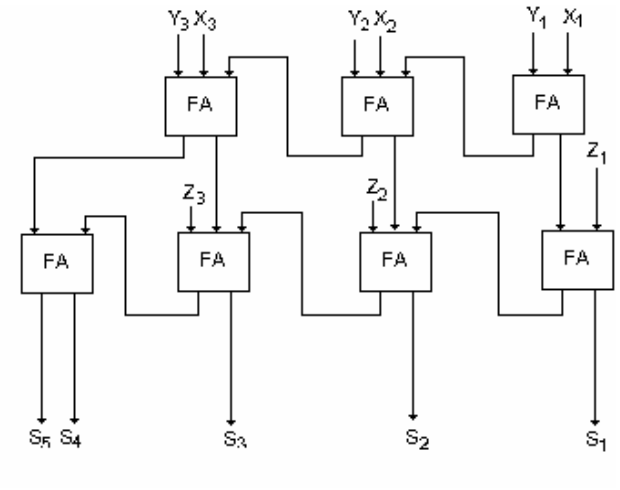
6.6.5 Mạch cộng lưu số nhớ

Nhắc lại, một mạch cộng toàn phần (FA) nhận 3-bit ở ngã vào và cho 2 ngã ra :

- Một là tổng của các bit có cùng trọng số với các bit ở ngã vào
- Một là số nhớ có trọng số gấp đôi trọng số của các bit ở ngã vào

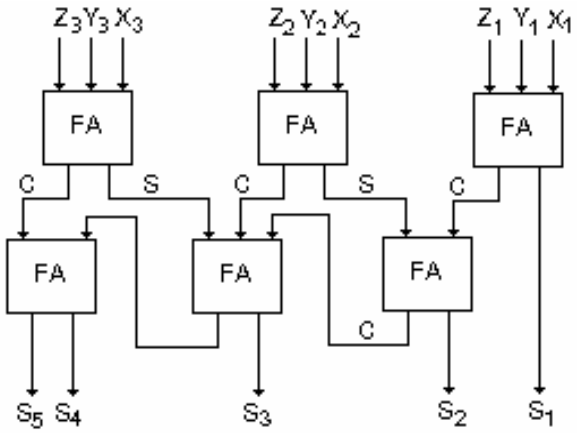
Để cộng một chuỗi số, nhiều mạch cộng toàn phần sẽ được sử dụng, số nhớ được lưu lại để đưa vào mạch cộng bit cao hơn.

Thí dụ 14 : Với 3 số 3-bit X ($X_3X_2X_1$), Y($Y_3Y_2Y_1$), Z ($Z_3Z_2Z_1$) mạch cộng có dạng



(H 6.10)

Người ta dùng mạch cộng loại này để thực hiện bài toán nhân.
Để có kết quả nhanh hơn, có thể dùng mạch (H 6.11)



(H 6.11)

6.7 Mạch trừ nhị phân:

6.7.1 Mạch trừ bán phần

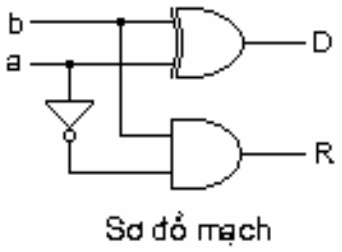
Là mạch trừ hai số 1 bit (H 6.12)

a	b	D	R
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Bảng sự thật

$D = a \oplus b$

$R = \overline{a}.b$



Sơ đồ mạch

(H 6.12)

6.7.2 Mạch trừ có số nhớ (mạch trừ toàn phần)

Là mạch trừ 2 bit có quan tâm tới số nhớ mang từ bit trước

R_{n-1}		B_n	D_n	R_n
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1

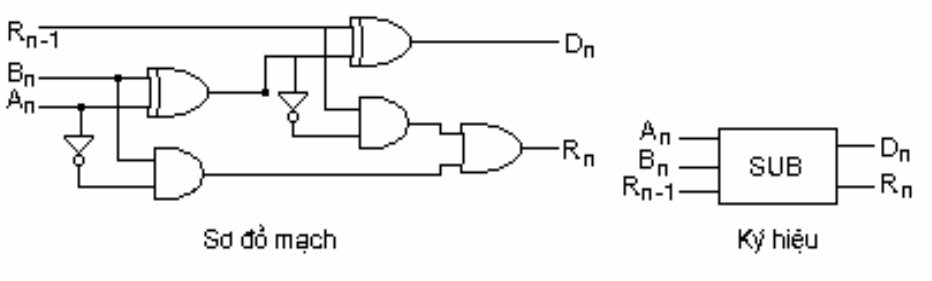
Bảng sự thật

Dùng bảng Karnaugh xác định được các hàm D_n và R_n

$$D_n = R_{n-1} \oplus (A_n \oplus B_n)$$

$$R_n = \overline{A}_n B_n + R_{n-1}\overline{(A_n \oplus B_n)}$$

Và mạch (H 6.13)

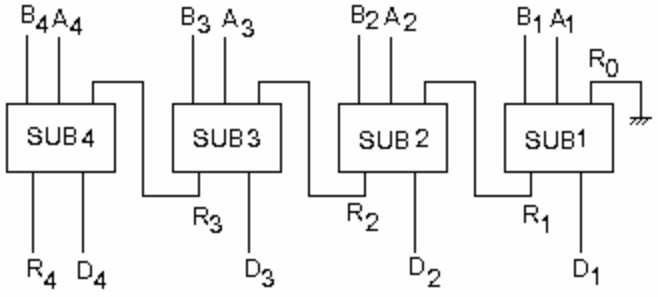


(H 6.13)

Nhận thấy cấu tạo mạch trừ giống như mạch cộng, chỉ khác ở mạch tạo số nhớ

6.7.3 Trừ số nhiều bit

Ta có mạch trừ số nhiều bit bằng cách mắc song song các mạch trừ 1 bit (H 6.14)

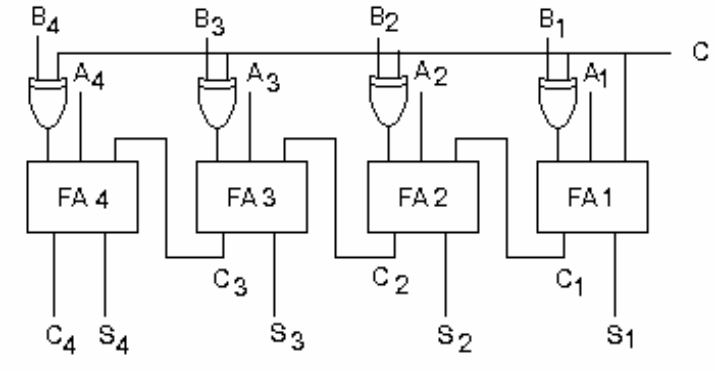


(H 6.14)

6.7.4 Cộng và trừ số nhiều bit trong một mạch

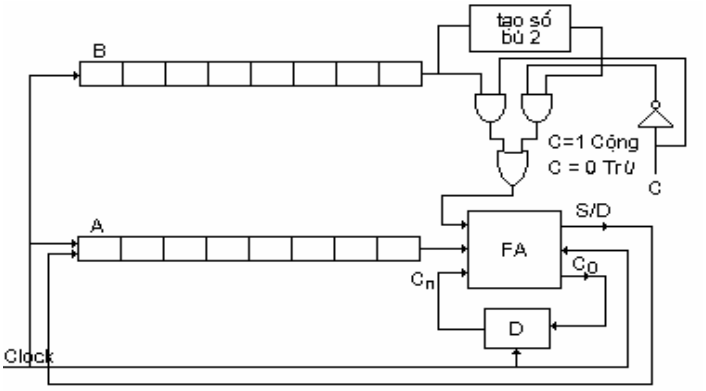
Nhắc lại để thực hiện phép toán trừ, người ta cộng với số bù 1 và cộng thêm 1 (hoặc cộng với số bù 2), như vậy để thực hiện phép trừ A - B ta tính A + $(B)_1$ + 1 . Mạch (H 6.6) được sửa đổi để có thực hiện phép cộng và trừ tùy vào ngã điều khiển C (H 6.15)

- Khi C=0, ta có mạch cộng
- Khi C =1, ta có mạch trừ



(H 6.15)

Ta cũng có thể thực hiện mạch cộng trừ theo kiểu mắc nối tiếp (H 6.16)



(H 6.16)

Nếu hai số A, B là số 8 bit, có dấu, kết quả được xử lý bởi mạch dò số tràn, thiết kế dựa vào biểu thức: $OV = C_7 \oplus C_8$. Khi OV =1 nghĩa là có số tràn (tức $C_7 \neq C_8$), thì số tràn C_8 sẽ là bit dấu, S_8 là một bit của kết quả và khi OV = 0 (tức $C_7 = C_8$), thì S_8 là bit dấu.

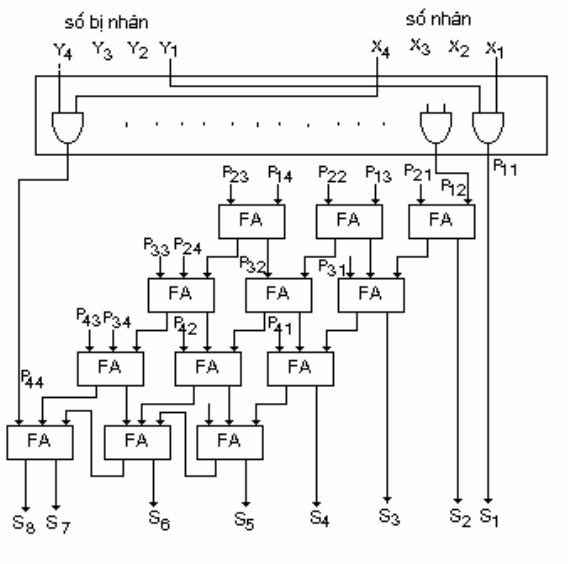
6.8 Mạch nhân

Lấy thí dụ bài toán nhân 2 số 4 bit

				Y_4	Y_3	Y_2	Y_1	Số bị nhân
				X_4	X_3	X_2	X_1	Số nhân
				P_{14}	P_{13}	P_{12}	P_{11}	Tích từng phần
			P_{24}	P_{23}	P_{22}	P_{21}		
		P_{34}	P_{33}	P_{32}	P_{31}			
	P_{44}	P_{43}	P_{42}	P_{41}				
S_8	S_7	S_6	S_5	S_4	S_3	S_2	S_1	Kết quả

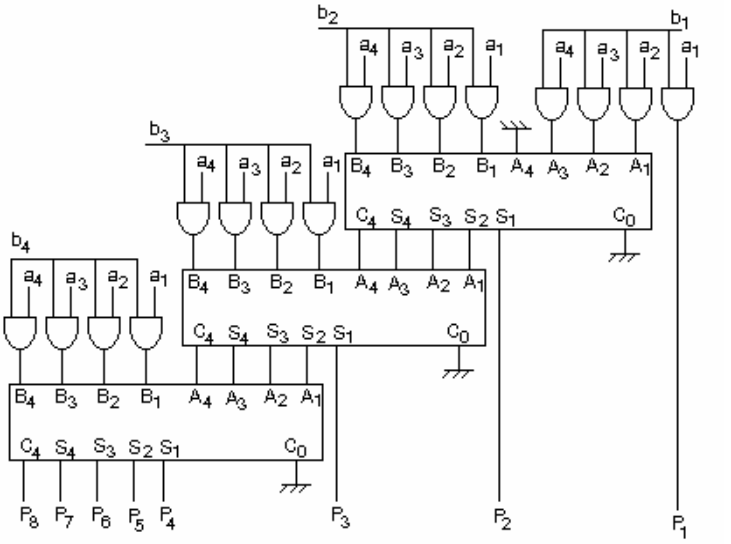
6.8.1. Mạch nhân cơ bản

Việc thực hiện bài toán nhân có thể xem như gồm hai bước:
- Tính các tích từng phần: thực hiện bởi các cổng AND
- Tính tổng của các tích từng phần: Áp dụng bài toán tổng chuỗi số (H 6.17)



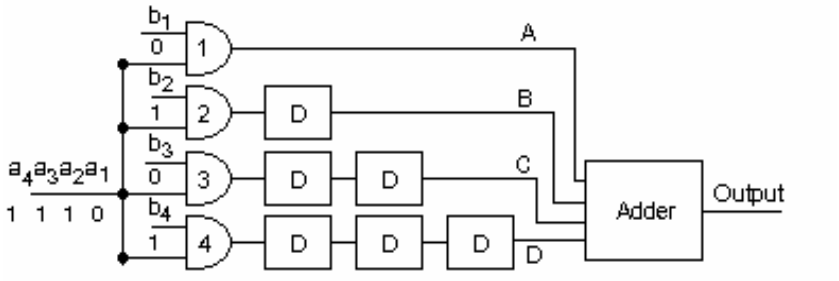
(H 6.17)

Dùng IC cộng 4 bit (7483 hoặc 4008) mạch nhân hai số 4 bit có dạng (H 6.18)



(H 6.18)

6.8.2. Mạch nhân nối tiếp - song song đơn giản (H 6.19)



(H 6.19)

Trong mạch này, một trong hai số được đưa nối tiếp vào mạch (trong trường hợp này là số bị nhân) và số còn lại đưa song song vào mạch.

- Số nhân ($b_4b_3b_2b_1$) đưa song song vào mạch qua các cổng AND đồng thời kiểm soát các cổng này: ứng với bit 1 số bị nhân qua mạch để tới mạch cộng (cổng 2 và 4); ứng với bit 0 ngã ra cổng AND bằng không (cổng 1 và 3)

- Số bị nhân đưa nối tiếp vào mạch theo thứ tự từ bit LSB. Các FF D có tác dụng dịch kết quả của phép nhân (là các tích từng phần) trước khi đưa vào mạch cộng để cộng các tích từng phần này.

Thí dụ 15 : Xem bài toán nhân 10x14. Số nhân là 1010 (10_{10}) và số bị nhân là 1110 (14_{10}). Quá trình nhân giải thích như sau:

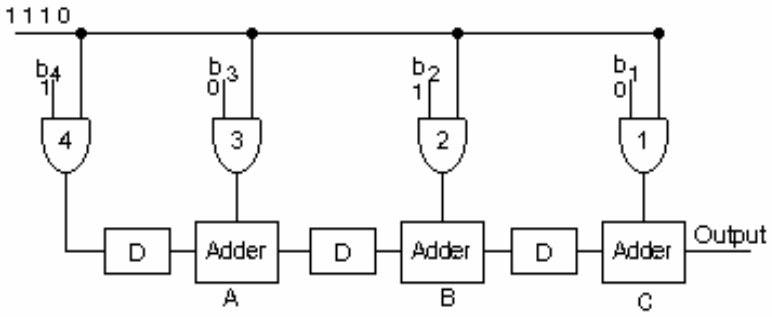
	P_8	P_7	P_6	P_5	P_4	P_3	P_2	P_1
A	0	0	0	0	0	0	0	0
B	0	0	0	1	1	1	0	0
C	0	0	0	0	0	0	0	0
D	0	1	1	1	0	0	0	0
Output	1	0	0	0	1	1	0	0

$10001100_2 = 140_{10}$

Có thể thấy rằng ngã ra A luôn luôn bằng 0 vì bit LSB của số nhân = 0. Ngã ra B có giá trị của số bị nhân được làm trễ 1 bit (1 xung đồng hồ). Ngã ra C được làm trễ 2 bit và luôn bằng 0 (Giống như A). Ngã ra D giống như B nhưng trễ 3 bit. Điều này có thể so sánh với bài toán trên giấy

Số bị nhân					1	1	1	0
Số nhân					1	0	1	0
A					0	0	0	0
B				1	1	1	0	0
C			0	0	0	0	0	0
D		1	1	1	0	0	0	0
Tích	1	0	0	0	1	1	0	0

Muốn không sử dụng mạch cộng số nhiều bit, người ta dùng mạch (H 6.20)



(H 6.20)

Mạch (H 6.20) cần (n-1) mạch cộng và mạch trễ (FF D) cho số nhân n bit. Các cổng AND cho phép các bit của số bị nhân đi qua khi số nhân là 1, số bị nhân (với số bit bất kỳ) được cho vào mạch nối tiếp với bit LSB vào đầu tiên.

Ngã ra cổng 4 sau 4 xung Clock là 1110. Ngã ra cổng 3 luôn luôn bằng 0.

Mạch cộng A cộng số ngã ra 3 và ngã ra 4 bị trễ 1 bit:

	0	0	0	0
1	1	1	0	0
1	1	1	0	0

Tương tự mạch cộng B cộng số bị nhân với kết quả ở A được làm trễ 1 bit

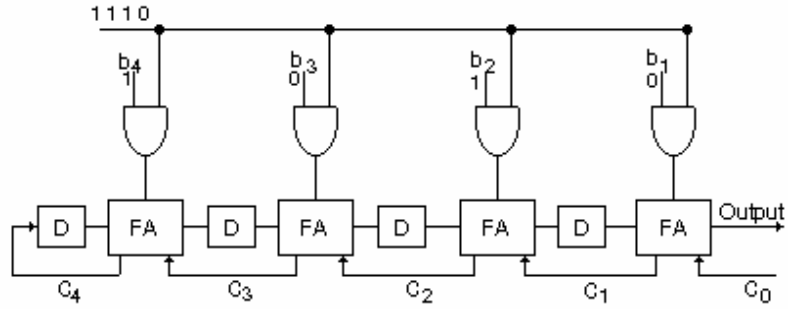
			1	1	1	0
	1	1	1	0	0	0
1	0	0	0	1	1	0

và mạch cộng C

				0	0	0	0
1	0	0	0	1	1	0	0
1	0	0	0	1	1	0	0

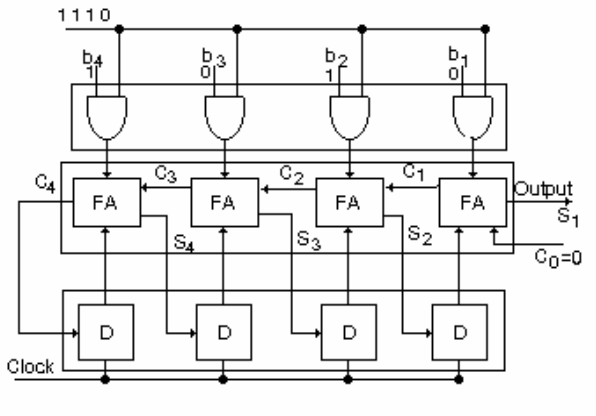
Lưu ý là ở mạch (H 6.20) kết quả cho ở ngã ra mạch cộng C với bit LSB ra đầu tiên, tuy nhiên mạch này chưa quan tâm tới số nhớ.

Mạch (H 6.21) cho kết quả với số nhớ .



(H 6.21)

Và (H 6.22) là một mạch thực tế dùng ghi dịch 4 bit có ngã vào/ra song song, một mạch cộng 4 bit và một chip 4 cổng AND 2 ngã vào để thực hiện bài toán nhân.

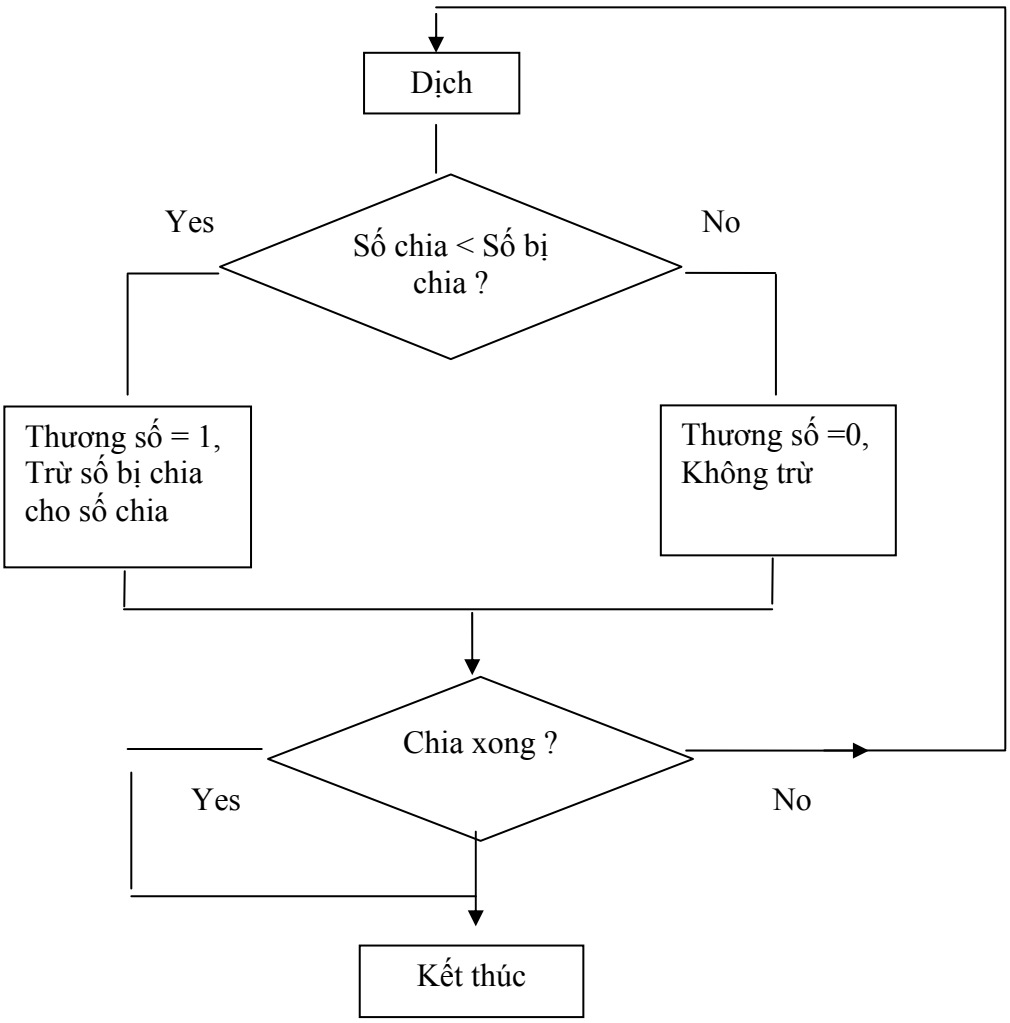


(H 6.22)

6.9 Mạch chia

Nguyên tắc của phép chia số nhị phân là thực hiện phép so sánh một phần của số bị chia (số bit đầu tiên bằng với số bit của số chia) với số chia, nếu số bị chia lớn hơn số chia thì thương số =1, thực hiện phép trừ, ngược lại thì thương số =0, sau đó dịch trái phần còn lại của số bị chia một bit (hoặc dịch phải số chia 1 bit) rồi tiếp tục thực hiện bài toán so sánh giống như trên. Công việc được lặp lại cho đến khi chấm dứt.

Sơ đồ (H 6.23) tóm tắt giải thuật thực hiện bài toán chia

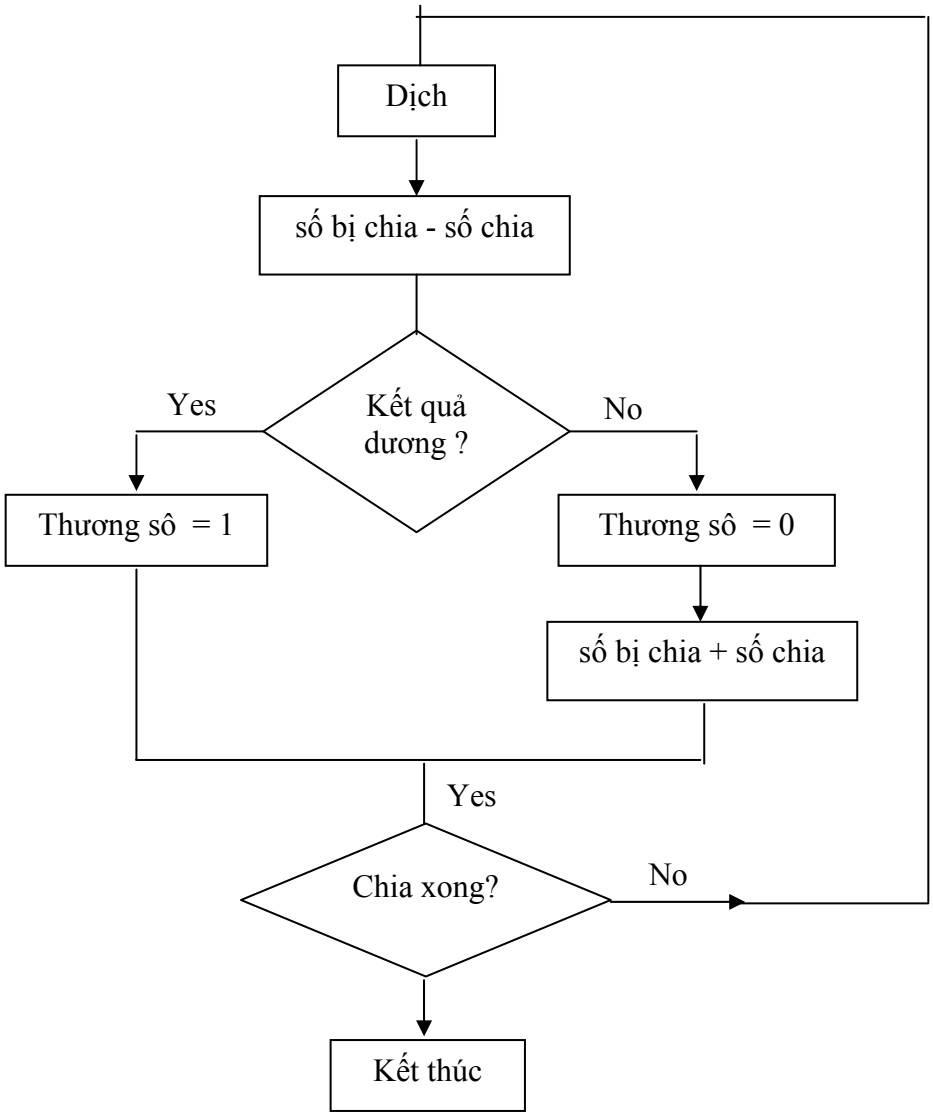


(H 6.23)

6.9.1 Phép chia có phục hồi số bị chia

Thay vì phải thực hiện việc so sánh, người ta làm phép tính trừ một phần số bị chia cho số chia, nếu kết quả dương, thương số là 1, nếu kết quả âm, thương số là 0, trong trường hợp này phải phục hồi lại số bị chia bằng cách cộng số bị chia cho số chia trước khi dịch số bị chia sang trái 1 bit (hoặc số chia sang phải) để tiếp tục lặp lại bài toán cho đến khi kết thúc.

(H 6.24) là sơ đồ giải thuật thực hiện phép chia có phục hồi số bị chia.



(H 6.24)

Để thực hiện phép chia theo sơ đồ trên, ngoài các thanh ghi để chứa các số bị chia, số chia , số thương người ta phải dùng thanh ghi chứa số bị chia được phục hồi.

6.9.2 Phép chia không phục hồi số bị chia

Hệ thống sẽ đơn giản hơn nếu chúng ta dùng phép chia không cần phục hồi số bị chia theo nguyên tắc như dưới đây.

Quan sát giản đồ (H 6.24) ta thấy có 2 trường hợp:

♦ **Số chia lớn hơn số bị chia** (nhánh bên phải)

Lưu ý là dịch số chia về bên phải 1 bit tương đương với chia số đó cho 2

Nhánh bên phải của sơ đồ trên gồm 2 bài toán:

- Cộng số bị chia với số chia.
- Trừ số bị chia cho 1/2 số chia (trừ bị chia cho số chia đã dịch phải)

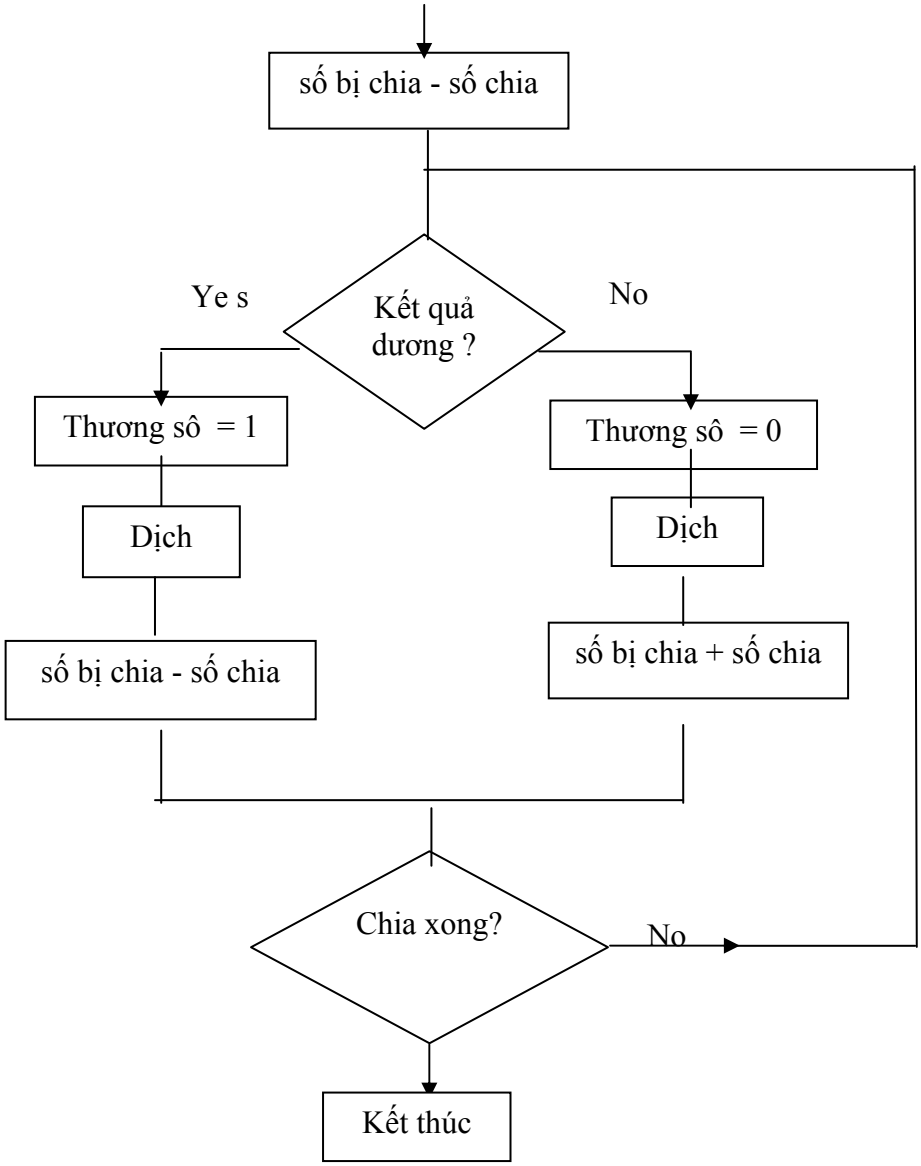
Hai bước này có thể gom lại thành một bước duy nhất như sau:

- Cộng số bị chia với số chia đã dịch phải.

♦ **Số chia nhỏ hơn số bị chia** (nhánh bên trái)

Sau khi lấy kết quả =1, lệnh kế tiếp thực hiện là trừ số chia đã dịch phải.

Từ các kết quả nhận xét trên có thể thay sơ đồ (H 6.24) bởi sơ đồ giải thuật thực hiện phép chia không cần phục hồi số bị chia (H 6.25)



(H 6.25)

Dựa vào sơ đồ (H 6.25), các bước thực hiện bài toán chia như sau:

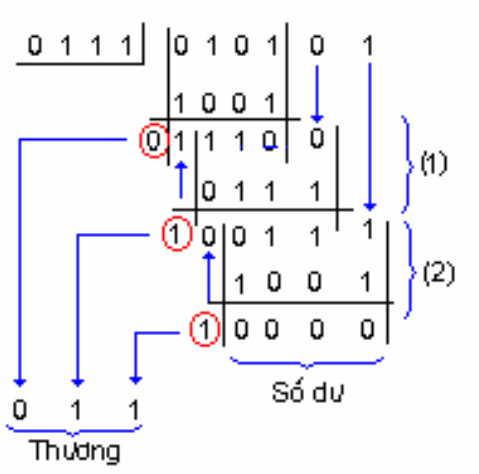
- Số chia (SC) lớn hơn số bị chia (SBC) (SBC - SC < 0), thương số là 0, dịch phải số chia 1 bit (thực tế ta mang thêm 1 bit của số bị chia xuống), thực hiện bài toán cộng số chia và số bị chia

- Số chia nhỏ hơn số bị chia (SBC - SC > 0), thương số là 1, dịch phải số chia 1 bit, thực hiện bài toán trừ (cộng số bù 2) số bị chia cho số chia

Để đơn giản, giả sử số chia và bị chia đều dương (MSB = 0), số bị chia gồm 6 bit và số chia gồm 4 bit.

Thí dụ 1: Thực hiện bài toán chia $21_{10} = 010101_2$ cho $7_{10} = 0111_2$.

Số bù 2 của 0111 là $(0111)_2 = 1001$



Ghi chú:

(1) Số 1 trên mũi tên chỉ rằng kết quả phép toán trừ là số âm, bước kế tiếp là dời và cộng số chia

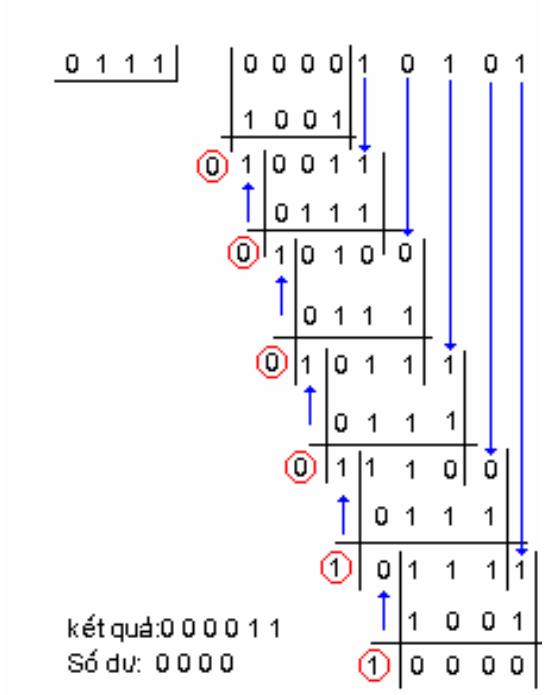
(2) Số 0 trên mủi tên chỉ rằng kết quả phép toán trừ là số dương, bước kế tiếp là dời và trừ số chia (cộng số bù 2)

Thương số có được từ các số tràn mà trên phép tính ta ghi trong vòng tròn.

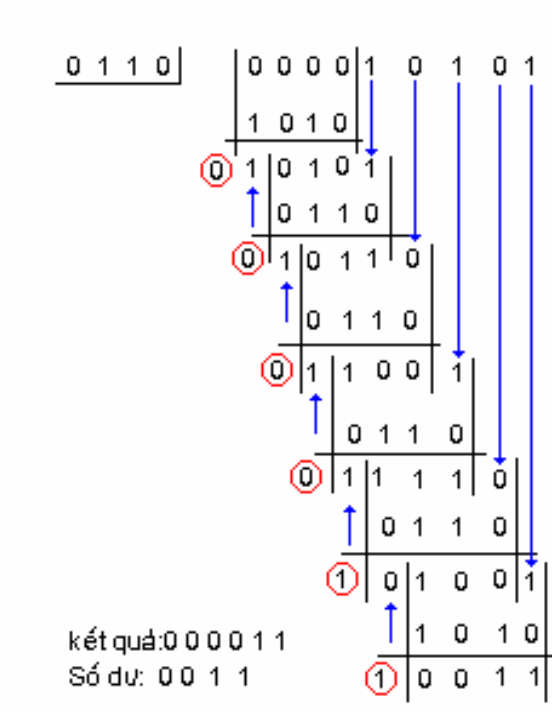
Kết quả: thương là 011(=3) và số dư là 0000(=0)

Bài toán trên cho kết quả với 3 bước cộng/trừ. Tuy nhiên nếu ta chia 21 cho 1 thì cần tới 6 bước cộng trừ để có thương số 6 bit. Một cách tổng quát số bước của bài toán bằng với số bít của số bị chia.

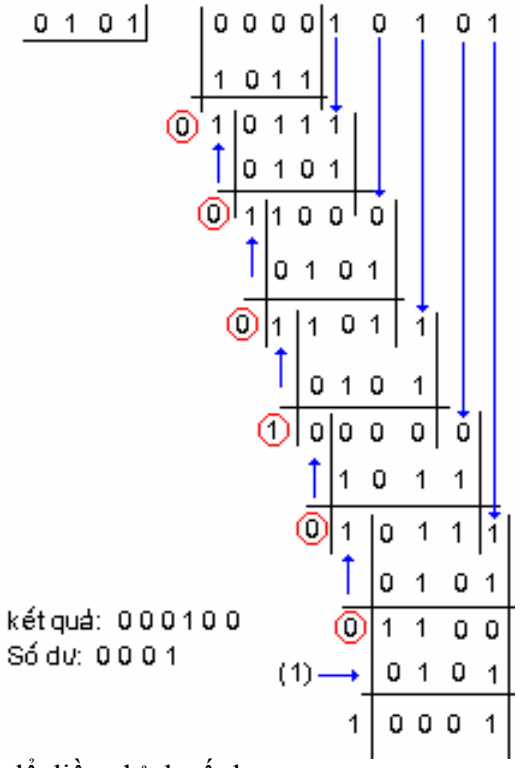
Ta có thể làm lại bài toán với 6 bước cộng/trừ ((thêm 3 bit 0 cho số bị chia)



Thí dụ 2 và 3 dưới đây là bài toán 6 bước
Thí dụ 2 : Chia 21 cho 6 được kết quả 3 và số dư là 3



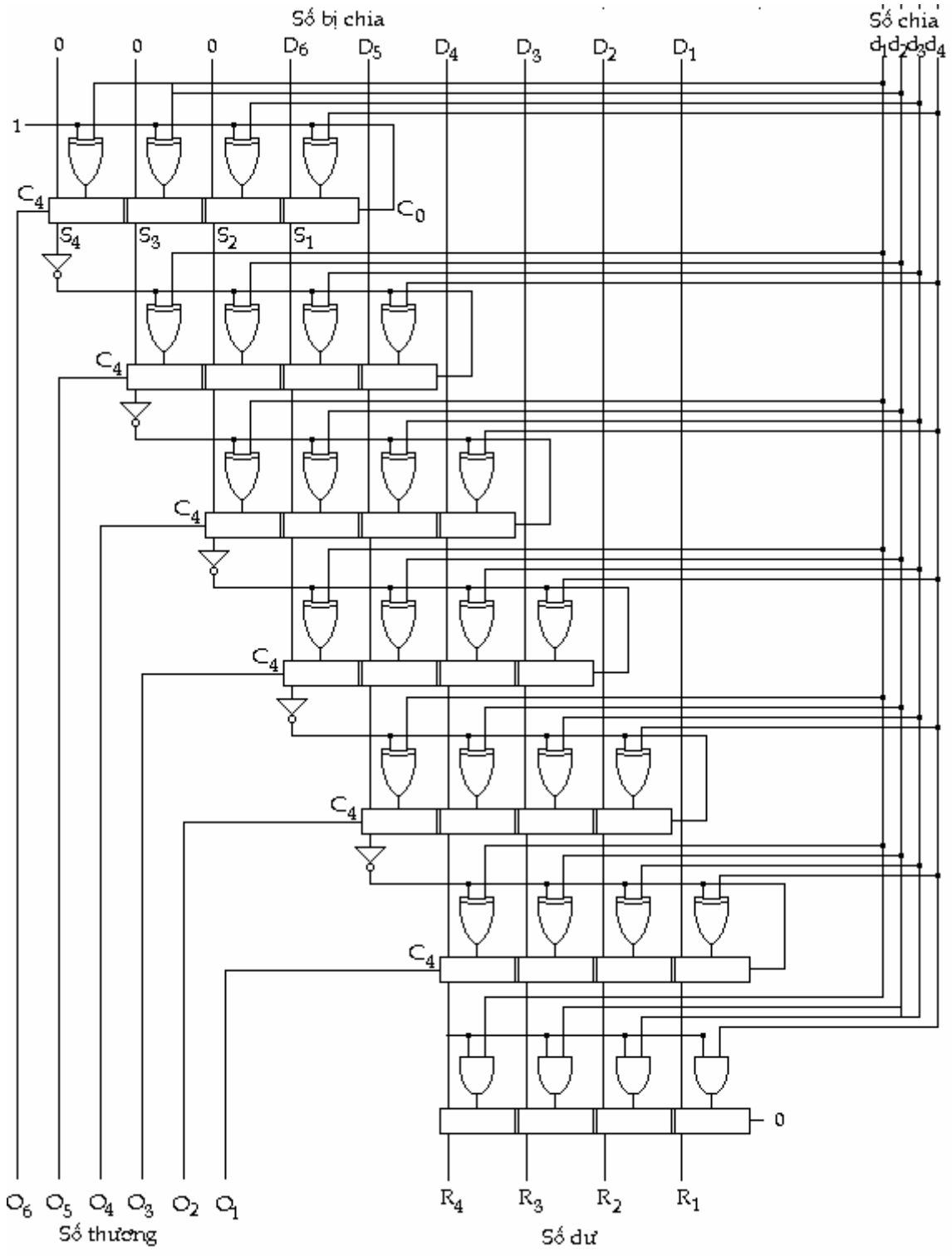
Thí dụ 3 : Chia 21 cho 5, được kết quả 4 và số dư là 1. Tuy nhiên trên phép toán ta thấy phép cộng với số chia cuối cùng cho kết quả âm (số 1100) nên để điều chỉnh số dư ta phải cộng số chia vào và bỏ qua số tràn.



(1) Cộng số chia vào để điều chỉnh số dư

Mạch thực hiện các bài toán này cho ở (H 6.26).

Trong (H 6.26) bước đầu tiên được thực hiện bởi các cổng EX-OR trên cùng có ngã điều khiển = 1 để thực hiện bài toán trừ. Sau bước thứ nhất, bit thứ tư của mạch cộng (S_4) sẽ quyết định phép toán sau đó là cộng (S_4=1) hay trừ (S_4=0) số bị chia với số chia. Số nhớ của bài toán cuối cùng (bước 6) là bit LSB của thương số. Và mạch cộng cuối cùng được thiết kế kết hợp với các cổng AND để xử lý kết quả của số dư như trong hai thí dụ 2 và 3. Nếu kết quả của bài toán ở bước 6 có S_4 = 1 thì cổng AND mở để thực hiện bài toán cộng với số chia để điều chỉnh số dư.


Số bị chia
Số chia
0
0
0
D6
D5
D4
D3
D2
D1
d1 d2 d3 d4
1
C4
C0
S4
S3
S2
S1
C4
C4
C4
C4
C4
0
O6 O5 O4 O3 O2 O1
Số thương
R4 R3 R2 R1
Số dư

(H 6.26)

CHƯƠNG 7: BỘ NHỚ BÁN DẪN

- **THUẬT NGỮ**
- **ĐẠI CƯƠNG VỀ VẬN HÀNH CỦA BỘ NHỚ**
  - Các tác vụ và các nhóm chân của IC nhớ
  - Giao tiếp với CPU
- **CÁC LOẠI BỘ NHỚ BÁN DẪN**
  - ROM
  - PLD
  - RAM
- **MỞ RỘNG BỘ NHỚ**
  - Mở rộng độ dài từ
  - Mở rộng vị trí nhớ
  - Mở rộng dung lượng nhớ

Tính ưu việt chủ yếu của các hệ thống số so với hệ thống tương tự là khả năng lưu trữ một lượng lớn thông tin số và dữ liệu trong những khoảng thời gian nhất định. Khả năng nhớ này là điều làm cho hệ thống số trở thành đa năng và có thể thích hợp với nhiều tình huống. Thí dụ trong một máy tính số, bộ nhớ trong chứa những lệnh mà theo đó máy tính có thể hoàn tất công việc của mình với sự tham gia ít nhất của con người.

Bộ nhớ bán dẫn được sử dụng làm **bộ nhớ chính** trong các máy tính nhờ vào khả năng thỏa mãn tốc độ truy xuất dữ liệu của bộ xử lý trung tâm (CPU).

Chúng ta đã quá quen thuộc với Fliflop, một linh kiện điện tử có tính nhớ. Chúng ta cũng đã thấy một nhóm các FF họp thành thanh ghi để lưu trữ và dịch chuyển thông tin như thế nào. Các FF chính là các phần tử nhớ tốc độ cao được dùng rất nhiều trong việc điều hành bên trong máy tính, nơi mà dữ liệu dịch chuyển liên tục từ nơi này đến nơi khác.

Tiến bộ trong công nghệ chế tạo LSI và VLSI cho phép kết hợp một lượng lớn FF trong một chip tạo thành các bộ nhớ với các dạng khác nhau. Những bộ nhớ bán dẫn với công nghệ chế tạo transistor lưỡng cực (BJT) và MOS là những bộ nhớ nhanh nhất và giá thành của nó liên tục giảm khi các công nghệ LSI và VLSI ngày càng được cải tiến.

Dữ liệu số cũng có thể được lưu trữ dưới dạng điện tích của tụ điện, và một loại phần tử nhớ bán dẫn rất quan trọng đã dùng nguyên tắc này để lưu trữ dữ liệu với mật độ cao nhưng tiêu thụ một nguồn điện năng rất thấp.

Bộ nhớ bán dẫn được dùng như là **bộ nhớ trong** chính của máy tính, nơi mà việc vận hành nhanh được xem như ưu tiên hàng đầu và cũng là nơi mà tất cả dữ liệu của chương trình lưu chuyển liên tục trong quá trình thực hiện một tác vụ do CPU yêu cầu.

Mặc dù bộ nhớ bán dẫn có tốc độ làm việc cao, rất phù hợp cho bộ nhớ trong, nhưng giá thành tính trên mỗi bit lưu trữ cao khiến cho nó không thể là loại thiết bị có tính chất lưu trữ khối (mass storage), là loại thiết bị có khả năng lưu trữ hàng tỉ bit mà không cần cung cấp năng lượng và được dùng như là **bộ nhớ ngoài** (đĩa từ , băng từ , CD ROM . . .). Tốc độ xử lý dữ liệu ở bộ nhớ ngoài tương đối chậm nên khi máy tính làm việc thì dữ liệu từ bộ nhớ ngoài được chuyển vào bộ nhớ trong.

Băng từ và đĩa từ là các thiết bị lưu trữ khối mà giá thành tính trên mỗi bit tương đối thấp. Một loại bộ nhớ khối mới hơn là bộ nhớ **bọt từ** (**magnetic bubble memory**, MBM) là bộ nhớ điện tử dựa trên nguyên tắc từ có khả năng lưu trữ hàng triệu bit trong một chip. Với tốc độ tương đối chậm nó không được dùng như bộ nhớ trong.

Chương này nghiên cứu cấu tạo và tổ chức của các **bộ nhớ bán dẫn**.

7.1 Thuật ngữ liên quan đến bộ nhớ

Để tìm hiểu cấu tạo, hoạt động của bộ nhớ chúng ta bắt đầu với một số thuật ngữ liên quan đến bộ nhớ

- Tế bào nhớ: là linh kiện hay một mạch điện tử dùng để lưu trữ một bit đơn (0 hay 1). Thí dụ của một tế bào nhớ bao gồm: mạch FF, tụ được tích điện, một điểm trên băng từ hay đĩa từ. . . .

- Từ nhớ : là một nhóm các bit (tế bào) trong bộ nhớ dùng biểu diễn các lệnh hay dữ liệu dưới dạng một số nhị phân. Thí dụ một thanh ghi 8 FF là một phần tử nhớ lưu trữ từ 8 bit. Kích thước của từ nhớ trong các máy tính hiện đại có chiều dài từ 4 đến 64 bit.

- Byte : từ 8 bit, đây là kích thước thường dùng của từ nhớ trong các máy vi tính.

- Dung lượng : chỉ số lượng bit có thể lưu trữ trong bộ nhớ. Thí dụ bộ nhớ có khả năng lưu trữ 4.096 từ nhớ 20 bit, dung lượng của nó là 4096 x 20, mỗi 1024 ($=2^{10}$) từ nhớ được gọi là "1K", như vậy 4096 x 20 = 4K x 20. Với dung lượng lớn hơn ta dùng "1M" hay 1meg để chỉ 2^{20} = 1.048.576 từ nhớ.

- Địa chỉ : là số nhị phân dùng xác định vị trí của từ nhớ trong bộ nhớ. Mỗi từ nhớ được lưu trong bộ nhớ tại một địa chỉ duy nhất. Địa chỉ luôn luôn được biểu diễn bởi số nhị phân, tuy nhiên để thuận tiện người ta có thể dùng số hex hay thập phân, bát phân

- Tác vụ đọc : (*Read*, còn gọi là *fetch*), một từ nhớ tại một vị trí nào đó trong bộ nhớ được truy xuất và chuyển sang một thiết bị khác.

- Tác vụ viết : (ghi, *Write*, còn gọi là *store*), một từ mới được đặt vào một vị trí trong bộ nhớ, khi một từ mới được viết vào thì từ cũ mất đi.

- Thời gian truy xuất (access time) : số đo tốc độ hoạt động của bộ nhớ, ký hiệu t_{ACC} Đó là thời gian cần để hoàn tất một tác vụ đọc. Chính xác đó là thời gian từ khi bộ nhớ nhận một địa chỉ mới cho tới lúc dữ liệu khả dụng ở ngã ra bộ nhớ

- Bộ nhớ không vĩnh cữu (volatile) : Bộ nhớ cần nguồn điện để lưu trữ thông tin. Khi ngắt điện, thông tin lưu trữ bị mất. Hầu hết bộ nhớ bán dẫn là loại không vĩnh cữu, trong khi bộ nhớ từ là loại vĩnh cữu (nonvolatile).

- Bộ nhớ truy xuất ngẫu nhiên (Random-Access Memory, **RAM**) : Khi cần truy xuất một địa chỉ ta tới ngay địa chỉ đó. Vậy thời gian đọc hay viết dữ liệu vào các vị trí nhớ khác nhau trong bộ nhớ không tùy thuộc vào vị trí nhớ. Nói cách khác, thời gian truy xuất như nhau đối với mọi vị trí nhớ. Hầu hết bộ nhớ bán dẫn và **nhẫn từ** (bộ nhớ trong của máy tính trước khi bộ nhớ bán dẫn ra đời) là loại truy xuất ngẫu nhiên.

- Bộ nhớ truy xuất tuần tự (Sequential-Access Memory, **SAM**) : Khi cần truy xuất một địa chỉ ta phải lướt qua các địa chỉ trước nó. Như vậy thời gian đọc và viết dữ liệu ở những vị trí khác nhau thì khác nhau. Những thí dụ của bộ nhớ này là băng từ, đĩa từ. Tốc độ làm việc của loại bộ nhớ này thường chậm so với bộ nhớ truy xuất ngẫu nhiên.

- Bộ nhớ đọc/viết (Read/Write Memory, **RWM**) : Bộ nhớ có thể viết vào và đọc ra.

- Bộ nhớ chỉ đọc (Read-Only Memory, **ROM**): là bộ nhớ mà tỉ lệ tác vụ đọc trên tác vụ ghi rất lớn. Về mặt kỹ thuật, một ROM có thể được ghi chỉ một lần ở nơi sản xuất và sau đó thông tin chỉ có thể được đọc ra từ bộ nhớ. Có loại ROM có thể được ghi nhiều lần nhưng tác vụ ghi khá phức tạp hơn là tác vụ đọc. ROM thuộc loại bộ nhớ vĩnh cữu và dữ liệu được lưu giữ khi đã cắt nguồn điện.

- Bộ nhớ tĩnh (Static Memory Devices) : là bộ nhớ bán dẫn trong đó dữ liệu đã lưu trữ được duy trì cho đến khi nào còn nguồn nuôi.

- Bộ nhớ động (Dynamic Memory Devices) : là bộ nhớ bán dẫn trong đó dữ liệu đã lưu trữ muốn tồn tại phải được **ghi lại** theo chu kỳ. Tác vụ ghi lại được gọi là **làm tươi** (refresh).

- Bộ nhớ trong (Internal Memory) : Chỉ bộ nhớ chính của máy tính. Nó lưu trữ các lệnh và dữ liệu mà CPU dùng thường xuyên khi hoạt động.

- Bộ nhớ khối (Mass Memory): Còn gọi là bộ nhớ phụ, nó chứa một lượng thông tin rất lớn ở bên ngoài máy tính. Tốc độ truy xuất trên bộ nhớ này thường chậm và nó thuộc loại vĩnh cữu.

7.2 Đại cương về vận hành của bộ nhớ

7.2.1 Các tác vụ và các nhóm chân của một IC nhớ

Mặc dù mỗi loại bộ nhớ có hoạt động bên trong khác nhau, nhưng chúng có chung một số nguyên tắc vận hành mà chúng ta có thể tìm hiểu sơ lược trước khi đi vào nghiên cứu từng loại bộ nhớ.

Mỗi hệ thống nhớ luôn có một số yêu cầu ở các ngã vào và ra để hoàn thành một số tác vụ:

- Chọn địa chỉ trong bộ nhớ để truy xuất (đọc hoặc viết)
- Chọn tác vụ đọc hoặc viết để thực hiện
- Cung cấp dữ liệu để lưu vào bộ nhớ trong tác vụ viết
- Gửi dữ liệu ra từ bộ nhớ trong tác vụ đọc
- Cho phép (Enable) (hay Không, Disable) bộ nhớ đáp ứng (hay không) đối với lệnh đọc/ghi ở địa chỉ đã gọi đến.

Từ các tác vụ kể trên, ta có thể hình dung mỗi IC nhớ có một số ngã vào ra như sau:

- Ngã vào địa chỉ : mỗi vị trí nhớ xác định bởi một địa chỉ duy nhất, khi cần đọc dữ liệu ra hoặc ghi dữ liệu vào ta phải tác động vào chân địa chỉ của vị trí nhớ đó. Một IC có n chân địa chỉ sẽ có 2^n vị trí nhớ. Ký hiệu các chân địa chỉ là A_0 đến A_{n-1} Một IC có 10 chân địa chỉ sẽ có 1024 (1K) vị trí nhơ.

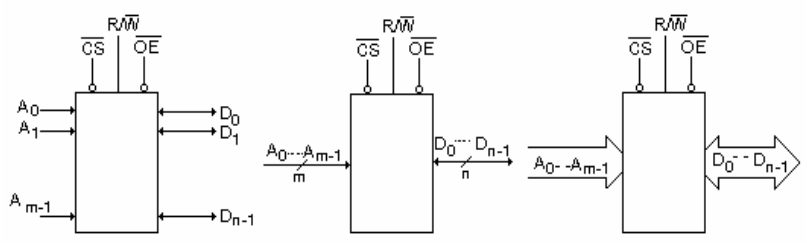
- Ngã vào/ra dữ liệu: Các chân dữ liệu là các ngã vào/ra, nghĩa là dữ liệu luôn được xử lý theo hai chiều. Thường thì dữ liệu vào/ra chung trên một chân nên các ngã này thuộc loại ngã ra 3 trạng thái. Số chân địa chỉ và dữ liệu của một IC xác định dung lượng nhớ của IC đó. Thí dụ một IC nhớ có 10 chân địa chỉ và 8 chân dữ liệu thì dung lượng nhớ của IC đó là 1Kx8 (8K bit hoặc 1K Byte).

- Các ngã vào điều khiển: Mỗi khi IC nhớ được chọn hoặc có yêu cầu xuất nhập dữ liệu các chân tương ứng sẽ được tác động. Ta có thể kể ra một số ngã vào điều khiển:

* $\overline{CS}$: Chip select - Chọn chip - Khi chân này xuống thấp IC được chọn
* $\overline{CE}$: Chip Enable - Cho phép chip - Chức năng như chân $\overline{CS}$
* $\overline{OE}$: Output Enable - Cho phép xuất - Dùng khi đọc dữ liệu
* $R/\overline{W}$: Read/Write - Đọc/Viết - Cho phép **Đọc** dữ liệu ra khi ở mức cao và **Ghi** dữ liệu vào khi ở mức thấp
* $\overline{CAS}$: Column Address Strobe - Chốt địa chỉ cột
* $\overline{RAS}$: Row Address Strobe - Chốt địa chỉ hàng.

Trong trường hợp chip nhớ có dung lượng lớn, để giảm kích thước của mạch giải mã địa chỉ bên trong IC, người ta chia số chân ra làm 2: địa chỉ hàng và địa chỉ cột. Như vậy phải dùng 2 mạch giải mã địa chỉ nhưng mỗi mạch nhỏ hơn rất nhiều. Thí dụ với 10 chân địa chỉ, thay vì dùng 1 mạch giải mã 10 đường sang 1024 đường, người ta dùng 2 mạch giải mã 5 đường sang 32 đường, hai mạch này rất đơn giản so với một mạch kia. Một vị trí nhớ bây giờ có 2 địa chỉ : hàng và cột, dĩ nhiên muốn truy xuất một vị trí nhớ phải có đủ 2 địa chỉ nhờ 2 tín hiệu $\overline{RAS}$ và $\overline{CAS}$.

(H 7.1) cho thấy cách vẽ các nhóm chân của IC nhớ (m chân địa chỉ và n chân dữ liệu). (H 7.1b) và (H 7.1c) vẽ các chân địa chỉ và dữ liệu dưới dạng các Bus. (H 7.1b) được dùng trong các sơ đồ chi tiết và (H 7.1c) được dùng trong các sơ đồ khối.

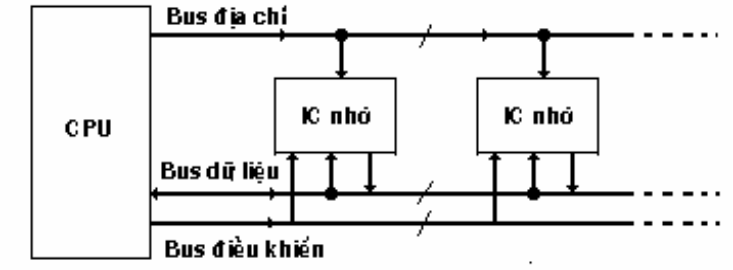


(a) (b) (c)

(H 7.1)

7.2.2 Giao tiếp giữa IC nhớ và bộ xử lý trung tâm (CPU)

Trong hệ thống mọi hoạt động có liên quan đến IC nhớ đều do bộ xử lý trung tâm (Central Processing Unit, CPU) quản lý. Giao tiếp giữa IC nhớ và CPU mô tả ở (H 7.2)



(H 7.2)

Một tác vụ có liên quan đến bộ nhớ được CPU thực hiện theo các bước:

- Đặt địa chỉ quan hệ lên bus địa chỉ.
- Đặt tín hiệu điều khiển lên bus điều khiển.
- Dữ liệu khả dụng xuất hiện trên bus dữ liệu, sẵn sàng để ghi vào hoặc đọc ra.

Để hoạt động của IC đồng bộ, các bước trên phải tuân thủ **giản đồ thời gian** của từng IC nhớ (sẽ đề cập đến khi xét các loại bộ nhớ)

7.3 Các loại bộ nhớ bán dẫn

Có 3 loại bộ nhớ bán dẫn :

- Bộ nhớ bán dẫn chỉ đọc : (Read Only Memory, ROM)
- Bộ nhớ truy xuất ngẫu nhiên : (Random Access Memory, RAM)

Thật ra ROM và RAM đều là loại bộ nhớ truy xuất ngẫu nhiên, nhưng RAM được giữ tên gọi này. Để phân biệt chính xác ROM và RAM ta có thể gọi ROM là **bộ nhớ chết** (nonvolatile, vĩnh cữu) và RAM là **bộ nhớ sống** (volatile, không vĩnh cữu) hoặc nếu coi ROM là **bộ nhớ chỉ đọc** thì RAM là **bộ nhớ đọc được - viết được** (Read-Write Memory)

- Thiết bị logic lập trình được : (Programmable Logic Devices, PLD) có thể nói điểm khác biệt giữa PLD với ROM và RAM là qui mô tích hợp của PLD thường không lớn như ROM và RAM và các tác vụ của PLD thì có phần hạn chế.

7.3.1 ROM (Read Only Memory)

Mặc dù có tên gọi như thế nhưng chúng ta phải hiểu là khi sử dụng ROM, tác vụ đọc được thực hiện rất nhiều lần so với tác vụ ghi. Thậm chí có loại ROM chỉ ghi một lần khi xuất xưởng.

Các tế bào nhớ hoặc từ nhớ trong ROM sắp xếp theo dạng ma trận mà mỗi phần tử chiếm một vị trí xác định bởi một địa chỉ cụ thể và nối với ngã ra một mạch giải mã địa chỉ

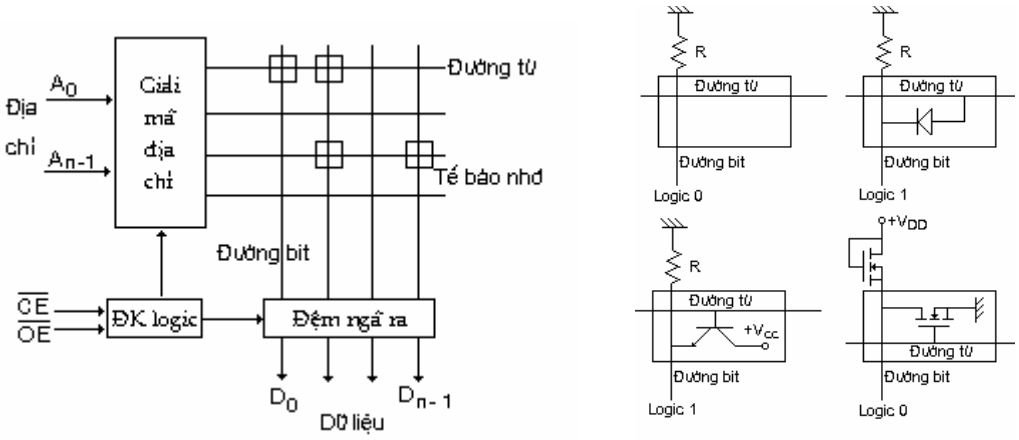
bên trong IC. Nếu mỗi vị trí chứa một tế bào nhớ ta nói ROM có **tổ chức bit** và mỗi vị trí là một từ nhớ ta có **tổ chức từ**.

Ngoài ra, để giảm mức độ cồng kềnh của mạch giải mã, mỗi vị trí nhớ có thể được xác định bởi 2 đường địa chỉ : đường địa chỉ hàng và đường địa chỉ cột và trong bộ nhớ có 2 mạch giải mã nhưng mỗi mạch có số ngã vào bằng 1/2 số đường địa chỉ của cả bộ nhớ.

7.3.1.1 ROM mặt nạ (Mask Programmed ROM, MROM)

Đây là loại ROM được chế tạo để thực hiện một công việc cụ thể như các bảng tính, bảng lượng giác , bảng logarit . . . . ngay sau khi xuất xưởng. Nói cách khác, các tế bào nhớ trong ma trận nhớ đã được tạo ra theo một chương trình đã xác định trước bằng phương pháp mặt nạ: đưa vào các linh kiện điện tử nối từ **đường từ** qua **đường bít** để tạo ra một giá trị bit và để trống cho giá trị bit ngược lại.

- (H 7.3) là mô hình của một MROM trong đó các ô vuông là nơi chứa (hay không) một linh kiện (diod, transistor BJT hay MOSFET) để tạo bit. Mỗi ngã ra của mạch giải mã địa chỉ gọi là đường từ và đường nối tế bào nhớ ra ngoài gọi là đường bit. Khi đường từ lên mức cao thì tế bào nhớ hoặc từ nhớ được chọn.

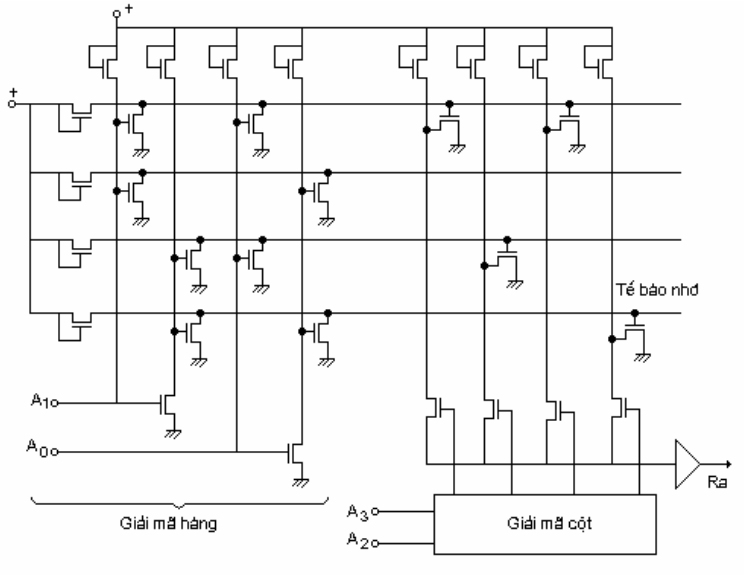


(H 7.3)

Nếu tế bào nhớ là Diod hoặc BJT thì sự hiện diện của linh kiện tương ứng với bit 1 (lúc này đường từ lên cao, Transsisstor hoặc diod dẫn, dòng điện qua điện trở tạo điện thế cao ở hai đầu điện trở) còn vị trí nhớ trống tương ứng với bit 0.

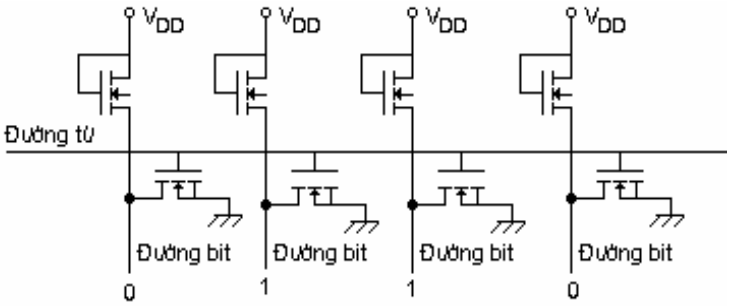
Đối với loại linh kiện MOSFET thì ngược lại, nghĩa là sự hiện diện của linh kiện tương ứng với bit 0 còn vị trí nhớ trống tương ứng với bit 1 (muốn có kết quả như loại BJT thì thêm ở ngã ra các cổng đảo).

(H 7.4) là một thí dụ bộ nhớ MROM có dung lượng 16x1 với các mạch giải mã hàng và cột (các mạch giải mã 2 đường sang 4 đường của hàng và cột đều dùng Transistor MOS và có cùng cấu trúc).



(H 7.4)

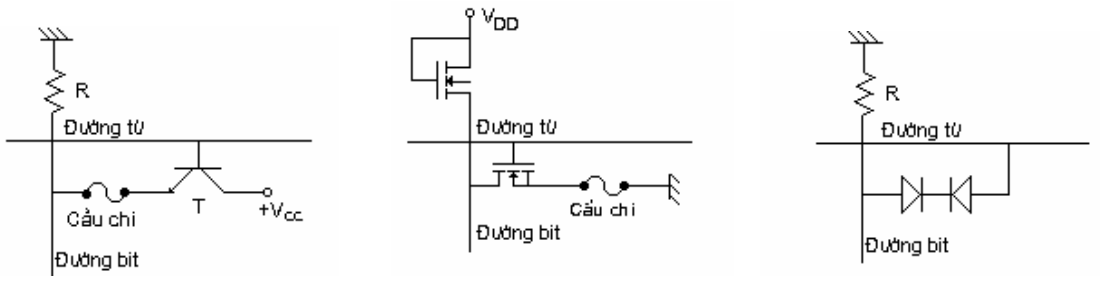
Trong thực tế, để đơn giản cho việc thực hiện, ở mỗi vị trí nhớ người ta đều cho vào một transistor MOS. Nhưng ở những vị trí ứng với bit 1 các transistor MOS được chế tạo với lớp SiO_2 dầy hơn làm tăng điện thế ngưỡng của nó lên, kết quả là transistor MOS này luôn luôn không dẫn điện (H 7.5), Các transistor khác dẫn điện bình thường.



(H 7.5)

7.3.1.2 ROM lập trình được (Programmable ROM, PROM)

Có cấu tạo giống MROM nhưng ở mỗi vị trí nhớ đều có linh kiện nối với cầu chì. Như vậy khi xuất xưởng các ROM này đều chứa cùng một loại bit (gọi là ROM trắng), lúc sử dụng người lập trình thay đổi các bit mong muốn bằng cách phá vỡ cầu chì ở các vị trí tương ứng với bit đó. Một khi cầu chì đã bị phá vỡ thì không thể nối lại được do đó loại ROM này cho phép lập trình một lần duy nhất để sử dụng, nếu bị lỗi không thể sửa chữa được (H 7.6).

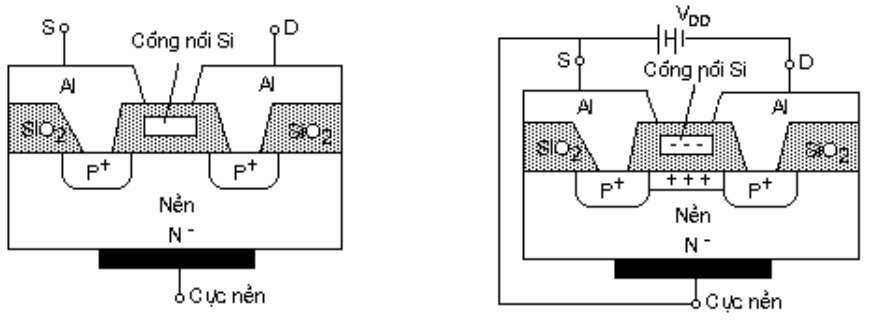


(H 7.6)

Người ta có thể dùng 2 diod mắc ngược chiều nhau, mạch không dẫn điện, để tạo bit 0, khi lập trình thì một diod bị phá hỏng tạo mạch nối tắt, diod còn lại dẫn điện cho bit 1

7.3.1.3 ROM lập trình được, xóa được bằng tia U.V. (Ultra Violet Erasable Programmable ROM, U.V. EPROM)

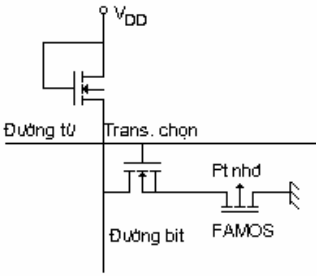
Đây là loại ROM rất tiện cho người sử dụng vì có thể dùng được nhiều lần bằng cách xóa và nạp lại. Cấu tạo của tế bào nhớ của U.V. EPROM dựa vào một transistor MOS có cấu tạo đặc biệt gọi là FAMOS (Floating Gate Avalanche Injection MOS)



(H 7.7)

Trên nền chất bán dẫn N pha loãng, tạo 2 vùng P pha đậm (P^+) nối ra ngoài cho 2 cực S (Source) và D (Drain). Trong lớp cách điện SiO_2 giữa 2 cực người ta cho vào một thỏi Silicon không nối với bên ngoài và được gọi là **cổng nổi**. Khi nguồn V_{DD}, phân cực ngược giữa cực nền và Drain còn nhỏ, transistor không dẫn, nhưng nếu tăng V_{DD} đủ lớn, hiện tượng thác đổ (avalanche) xảy ra, electron đủ năng lượng chui qua lớp cách điện tới bám vào cổng nổi. Do hiện tượng cảm ứng, một điện lộ P hình thành nối hai vùng bán dẫn P^+ , transistor trở nên dẫn điện. Khi cắt nguồn, transistor tiếp tục dẫn điện vì electron không thể trở về để tái hợp với lỗ trống.

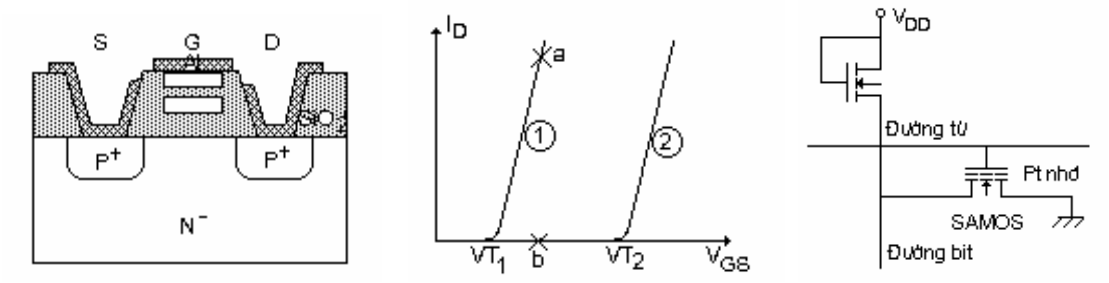
Để xóa EPROM, người ta chiếu tia U.V. vào các tế bào trong một khoảng thời gian xác định để electron trên cổng nổi nhận đủ năng lượng vượt qua lớp cách điện trở về vùng nền tái hợp với lỗ trống xóa điện lộ P và transistor trở về trạng thái không dẫn ban đầu.



(H 7.8)

Mỗi tế bào nhớ EPROM gồm một transistor FAMOS nối tiếp với một transistor MOS khác mà ta gọi là transistor chọn, như vậy vai trò của FAMOS giống như là một cầu chì nhưng có thể phục hồi được.

Để loại bỏ transistor chọn người ta dùng transistor SAMOS (Stacked Gate Avalanche Injection MOS) có cấu tạo tương tự transistor MOS nhưng có đến 2 cổng nằm chồng lên nhau, một được nối ra cực Gate và một để nổi. Khi cổng nổi tích điện sẽ làm gia tăng điện thế thềm khiến transistor trở nên khó dẫn điện hơn. Như vậy nếu ta chọn điện thế V_c ở khoảng giữa VT_1 và VT_2 là 2 giá trị điện thế thềm tương ứng với 2 trạng thái của transistor ($VT_1<V_c<VT_2$) thì các transistor không được lập trình (không có lớp electron ở cổng nổi) sẽ dẫn còn các transistor được lập trình sẽ không dẫn.

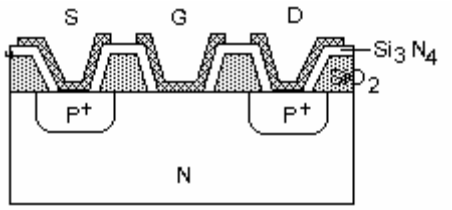


(H 7.9)

Điểm bất tiện của U.V EPROM là cần thiết bị xóa đặc biệt phát tia U.V. và mỗi lần xóa tất cả tế bào nhớ trong một IC nhớ đều bị xóa. Như vậy người sử dụng phải nạp lại toàn bộ chương trình

7.3.1.4 ROM lập trình được và xóa được bằng xung điện (Electrically Erasable PROM, EEPROM hay Electrically Alterable PROM, EAPROM)

Đây là loại ROM lập trình được và xóa được nhờ xung điện và đặc biệt là có thể xóa để sửa trên từng byte. Các tế bào nhớ EEPROM sử dụng transistor MNOS (Metal Nitride Oxide Semiconductor) có cấu tạo như (H 7.10).



(H 7.10)

Giữa lớp kim loại nối ra các cực và lớp SiO_2 là một lớp mỏng chất Nitrua Silic (Si_3N_4) - từ 40nm đến 650nm - Dữ liệu được nạp bằng cách áp một điện thế dương giữa cực G và S (khoảng 20 đến 25V trong 100ms). Do sự khác biệt về độ dẫn điện, electron tích trên bề mặt giữa 2 lớp SiO_2 và Si_3N_4 , các electron này tồn tại khi đã ngắt nguồn và làm thay đổi trạng thái dẫn điện của transistor. Bây giờ nếu áp một điện thế âm giữa cực G và S ta sẽ được một lớp điện tích trái dấu với trường hợp trước. Như vậy hai trạng thái khác nhau của Transistor có thể thiết lập được bởi hai điện thế ngược chiều nhau và như vậy các tế bào nhớ được ghi và xóa với 2 xung điện trái dấu nhau.

7.3.1.5 FLASH ROM

EPROM là loại nonvolatile, có tốc độ truy xuất nhanh (khoảng 120ns), mật độ tích hợp cao, giá thành rẻ tuy nhiên để xóa và nạp lại phải dùng thiết bị đặc biệt và lấy ra khỏi mạch.

EEPROM cũng nonvolatile, cũng có tốc độ truy xuất nhanh, cho phép xóa và nạp lại ngay trong mạch trên từng byte nhưng có mật độ tích hợp thấp và giá thành cao hơn EPROM.

Bộ nhớ FLASH ROM tận dụng được các ưu điểm của hai loại ROM nói trên, nghĩa là có tốc độ truy xuất nhanh, có mật độ tích hợp cao nhưng giá thành thấp.

Hầu hết các FLASH ROM sử dụng cách xóa đồng thời cả khối dữ liệu nhưng rất nhanh (hàng trăm ms so với 20 min của U.V. EPROM). Những FLASH ROM thế hệ mới cho phép xóa từng sector (512 byte) thậm chí từng vị trí nhớ mà không cần lấy IC ra khỏi mạch. FLASH ROM có thời gian ghi khoảng 10μs/byte so với 100 μs đối với EPROM và 5 ms đối với EEPROM

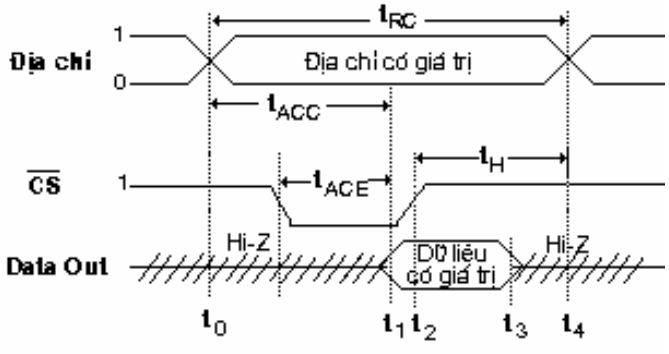
7.3.1.6 Giản đồ thời gian của ROM

Ngoại trừ MROM chỉ dùng ở chế độ đọc, các loại ROM khác đều sử dụng ở hai chế độ đọc và nạp chương trình.

Như vậy ta có hai loại giản đồ thời gian: Giản đồ thời gian đọc và giản đồ thời gian nạp trình.

(H 7.11) là giản đồ thời gian tiêu biểu cho một chu kỳ đọc của ROM.

Các giá trị địa chỉ, các tín hiệu R/$\overline{W}$ và $\overline{CS}$ được cấp từ CPU khi cần thực hiện tác vụ đọc dữ liệu tại một địa chỉ nào đó. Thời gian để thực hiện một tác vụ đọc gọi là chu kỳ đọc t_{RC}. Trong một chu kỳ đọc có thể kể một số thời gian sau:



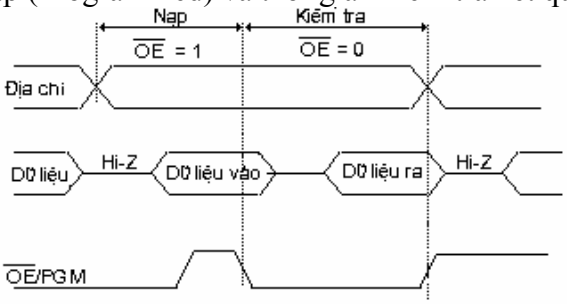
(H 7.11)

- t_{ACC}: Address Access time: Thời gian truy xuất địa chỉ: Thời gian tối đa từ lúc CPU đặt địa chỉ lên bus địa chỉ đến lúc dữ liệu có giá trị trên bus dữ liệu. Đối với ROM dùng BJT thời gian này khoảng từ 30 ns đến 90 ns, còn loại MOS thì từ 200 ns đến 900 ns.

- t_{ACS} (t_{ACE}): Chip select (enable) access time: Thời gian thâm nhập chọn chip: Thời gian tối đa từ lúc tín hiệu $\overline{CS}$ được đặt lên bus điều khiển đến lúc dữ liệu có giá trị trên bus dữ liệu. ROM BJT khoảng 20 ns , MOS 100 ns

- t_H (Hold time): Thời gian dữ liệu còn tồn tại trên bus dữ liệu kê từ lúc tín hiệu $\overline{CS}$ hết hiệu lực

(H 7.12) là giản đồ thời gian của một chu kỳ nạp dữ liệu cho EPROM. Một chu kỳ nạp liệu bao gồm thời gian nạp (Programmed) và thời gian kiểm tra kết quả (Verify)



(H 7.12)

7.3.2 Thiết bị logic lập trình được (Programmable logic devices, PLD)

Là tên gọi chung các thiết bị có tính chất nhớ và có thể lập trình để thực hiện một công việc cụ thể nào đó

Trong công việc thiết kế các hệ thống, đôi khi người ta cần một số mạch tổ hợp để thực hiện một hàm logic nào đó. Việc sử dụng mạch này có thể lặp lại thường xuyên và sự thay đổi một tham số của hàm có thể phải được thực hiện để thỏa mãn yêu cầu của việc thiết kế. Nếu phải thiết kế từ các cổng logic cơ bản thì mạch sẽ rất cồng kềnh, tốn kém mạch in,

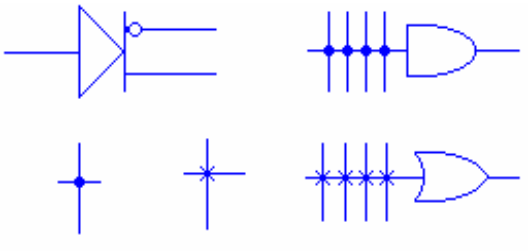
dây nối nhiều, kết quả là độ tin cậy không cao. Như vậy, sẽ rất tiện lợi nếu các mạch này được chế tạo sẵn và người sử dụng có thể chỉ tác động vào để làm thay đổi một phần nào chức năng của mạch bằng cách lập trình. Đó là ý tưởng cơ sở cho sự ra đời của thiết bị logic lập trình được. Các thiết bị này có thể được xếp loại như bộ nhớ và gồm các loại: PROM, PAL (Programmable Array Logic) và PLA (Programmable Logic Array).

Trước nhất, chúng ta xét qua một số qui ước trong cách biểu diễn các phần tử của PLD

Một biến trong các hàm thường xuất hiện ở dạng nguyên và đảo của nó nên chúng ta dùng ký hiêu đệm và đảo chung trong một cổng có 2 ngã ra.

Một **nối chết**, còn gọi là **nối cứng** (không thay đổi được) được vẽ bởi một chấm đậm (**.**) và một **nối sống**, còn gọi là **nối mềm** (dùng lập trình) bởi một dấu (x). Nối sống thực chất là một cầu chì, khi lập trình thì được phá bỏ.

Một cổng nhiều ngã vào thay thế bởi một ngã vào duy nhất với nhiều mối nối (H 7.13).



(H 7.13)

Chúng ta chỉ lấy thí dụ với mạch tương đối đơn giản để thấy được cấu tạo của các PLD, đó là các PLD chỉ thực hiện được 4 hàm mỗi hàm gồm 4 biến, như vậy mạch gồm 4 ngã vào và 4 ngã ra. Trên thực tế số hàm và biến của một PLD rất lớn.

7.3.2.1 PROM

(H 7.14) là cấu tạo PROM có 4 ngã vào và 4 ngã ra.

Có tất cả 16 cổng AND có 4 ngã vào được nối chết với các ngã ra đảo và không đảo của các biến vào, ngã ra các cổng AND là 16 tổ hợp của 4 biến (Gọi là đường tích)
Các cổng OR có 16 ngã vào được nối sống để thực hiện hàm tổng (đường tổng). Như vậy với PROM việc lập trình thực hiện ở các đường tổng.

Thí dụ dùng PROM này để tạo các hàm sau:

$O_1 = A + \overline{D}B + \overline{D}C \quad O_2 = \overline{D}\,\overline{C}BA + DC\overline{B}\,\overline{A} \quad O_3 = C\overline{B}A \quad O_4 = BA + \overline{D}\,\overline{C}$

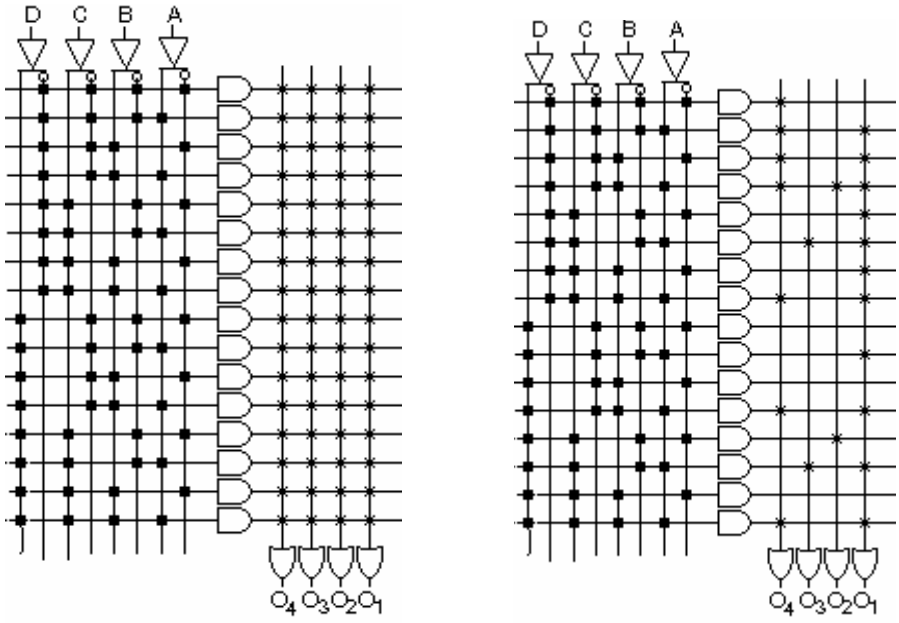
Ta phải chuẩn hóa các hàm chưa chuẩn

$$O_1 = DCBA + \overline{D}CBA + D\overline{C}BA + \overline{D}\,\overline{C}BA + DC\overline{B}A + \overline{D}C\overline{B}A + D\overline{C}\,\overline{B}A + \overline{D}\,\overline{C}\,\overline{B}A + \overline{D}CB\overline{A} + \overline{D}C\overline{B}\,\overline{A} + \overline{D}\,\overline{C}B\overline{A}$$

$$O_3 = C\overline{B}A = DC\overline{B}A + \overline{D}C\overline{B}A$$

$$O_4 = BA + \overline{D}\,\overline{C} = D\overline{C}BA + \overline{D}CBA + D\overline{C}BA + DCBA + \overline{D}\,\overline{C}BA + \overline{D}C\overline{B}\,\overline{A} + D\overline{C}\,\overline{B}\,\overline{A}$$

Mạch cho ở (H 7.14b)



(a) (b)

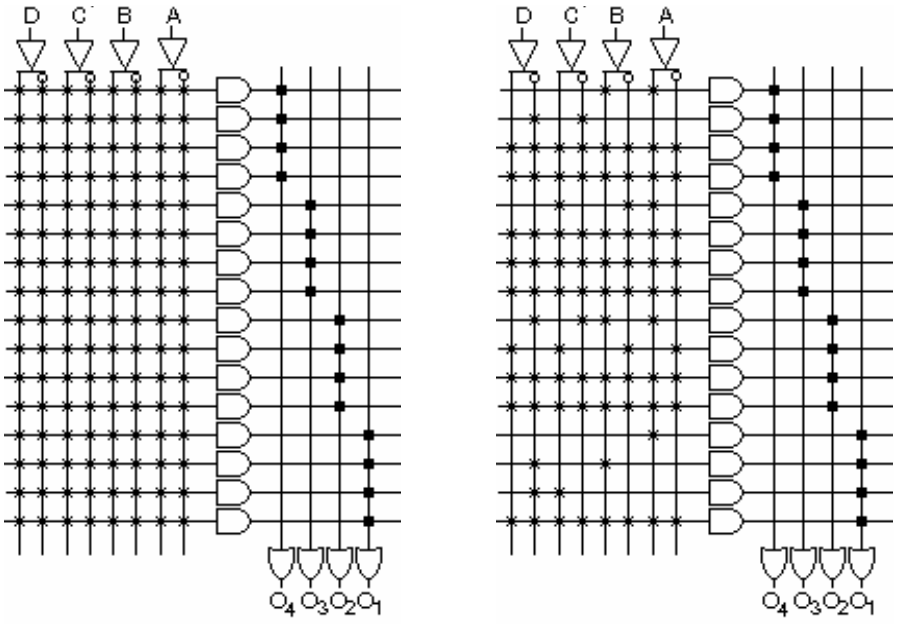
(H 7.14)

7.3.2.2 PAL

Mạch tương tự với IC PROM, PAL có các cổng AND 8 ngã vào được nối sống và 4 cổng OR mỗi cổng có 4 ngã vào nối chết với 4 đường tích. Như vậy việc lập trình được thực hiện trên các đường tích

(H 7.15b) là IC PAL đã được lập trình để thực hiện các hàm trong thí dụ trên:

$$O_1 = A + \overline{D}B + \overline{D}C \qquad O_2 = \overline{D}\,\overline{C}BA + DC\overline{B}\,\overline{A} \qquad O_3 = C\overline{B}A \qquad O_4 = BA + \overline{D}\,\overline{C}$$

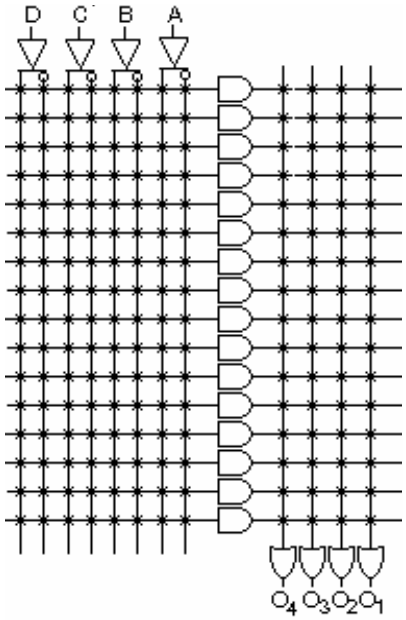


(a) (b)

(H 7.15)

7.3.2.3 PLA

PLA có cấu tạo tương tự PROM và PAL, nhưng các ngã vào của cổng AND và cổng OR đều được nối sống (H 7.16). Như vậy khả năng lập trình của PLA bao gồm cả hai cách lập trình của 2 loại IC kể trên.



(H 7.16)

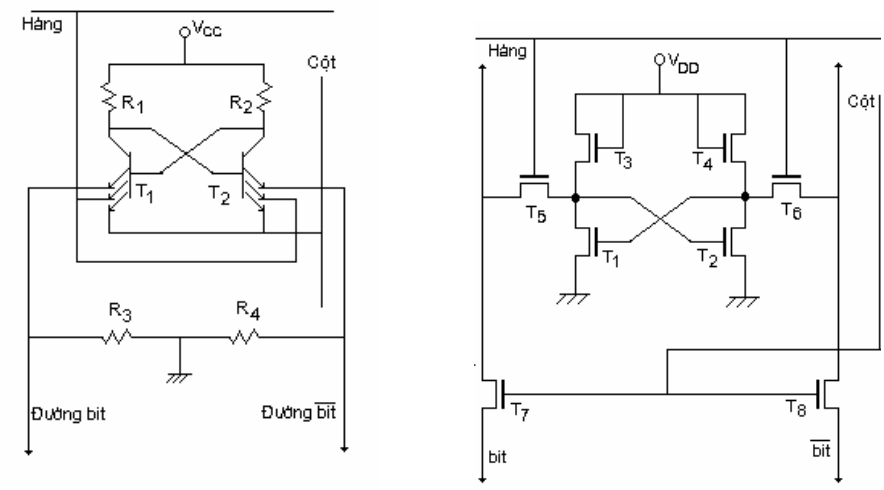
7.3.3 RAM (Random Acess Memory)

Có hai loại RAM : RAM tĩnh và RAM động

RAM tĩnh cấu tạo bởi các tế bào nhớ là các FF, RAM động lợi dụng các điện dung ký sinh giữa các cực của transistor MOS, trạng thái tích điện hay không của tụ tương ứng với hai bit 1 và 0. Do RAM động có mật độ tích hợp cao, dung lượng bộ nhớ thường rất lớn nên để định vị các phần tử nhớ người ta dùng phương pháp đa hợp địa chỉ, mỗi từ nhớ được chọn khi có đủ hai địa chỉ hàng và cột được lần lượt tác động. Phương pháp này cho phép n đường địa chỉ truy xuất được 2^{2n} vị trí nhớ. Như vậy giản đồ thời gian của RAM động thường khác với giản đồ thời gian của RAM tĩnh và ROM.

7.3.3.1 RAM tĩnh (Static RAM, SRAM)

Mỗi tế bào RAM tĩnh là một mạch FlipFlop dùng Transistor BJT hay MOS (H 7.17)



(a) (H 7.17) (b)

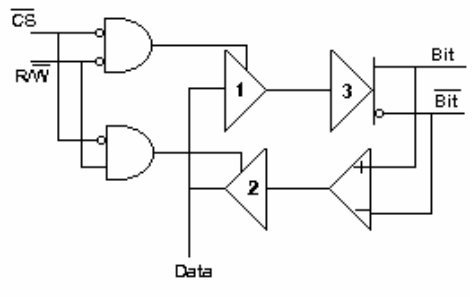
(H 7.17a) là một tế bào nhớ RAM tĩnh dùng transistor BJT với 2 đường địa chỉ hàng và cột.

Khi một trong hai đường địa chỉ hàng hoặc cột ở mức thấp các tế bào không được chọn vì cực E có điện thế thấp hai Transistor đều dẫn, mạch không hoạt động như một FF. Khi cả hai địa chỉ hàng và cột lên cao, mạch hoạt động như FF, hai trạng thái 1 và 0 của tế bào nhớ được đặc trưng bởi hai trạng thái khác nhau của 2 đường bit và $\overline{bit}$.

Giả sử khi T_1 dẫn thì T_2 ngưng, đường bit có dòng điện chạy qua, tạo điện thế cao ở R_3 trong khi đó đường $\overline{bit}$ không có dòng chạy qua nên ở R_4 có điện thế thấp. Nếu ta qui ước trạng thái này tương ứng với bit 1 thì trạng thái ngược lại, là trạng thái T_1 ngưng và T_2 dẫn, hiệu thế ở điện trở R_3 thấp và ở R_4 cao, sẽ là bit 0. R_3 và R_4 có tác dụng biến đổi dòng điện ra điện thế.

Đối với tế bào nhớ dùng MOS, hai đường từ nối với T_5, T_6 và T_7, T_8 nên khi một trong hai đường từ ở mức thấp T_1 và T_2 bị cô lập khỏi mạch, tế bào nhớ không được chọn. Khi cả hai lên cao mạch hoạt động tương tự như trên. Trong mạch này R_1 và R_2 thay bởi T_3 và T_4 và không cần R_3 và R_4 như mạch dùng BJT.

(H 7.18) là mạch điều khiển chọn chip và thực hiện tác vụ đọc/viết vào tế bào nhớ.



(H 7.18)

OPAMP giữ vai trò mạch so sánh điện thế hai đường bit và $\overline{bit}$ cho ở ngã ra mức cao hoặc thấp tùy kết quả so sánh này (tương ứng với 2 trạng thái của tế bào nhớ) và dữ liệu được đọc ra khi cổng đệm thứ 2 mở ($R/\overline{W}$ lên cao).

Khi cổng đệm thứ nhất mở ($R/\overline{W}$ xuống thấp) dữ liệu được ghi vào tế bào nhớ qua cổng đệm 1. Cổng 3 tạo ra hai tín hiệu ngược pha từ dữ liệu vào. Nếu hai tín hiệu này cùng trạng thái với hai đường bit và $\overline{bit}$ của mạch trước đó, mạch sẽ không đổi trạng thái nghĩa là

nếu tế bào nhớ đang lưu bit giống như bit muốn ghi vào thì mạch không thay đổi. Bây giờ, nếu dữ liệu cần ghi khác với dữ liệu đang lưu trữ thì mạch FF sẽ thay đổi trạng thái cho phù hợp với 2 tín hiệu ngược pha được tạo ra từ dữ liệu. Bit mới đã được ghi vào.

- Chu kỳ đọc của SRAM

Giản đồ thời gian một chu kỳ đọc của SRAM tương tự như giản đồ thời gian một chu kỳ đọc của ROM (H 7.11) thêm điều kiện tín hiệu $R/\overline{W}$ lên mức cao.

- Chu kỳ viết của SRAM

(H 7.19) là giản đồ thời gian một chu kỳ viết của SRAM

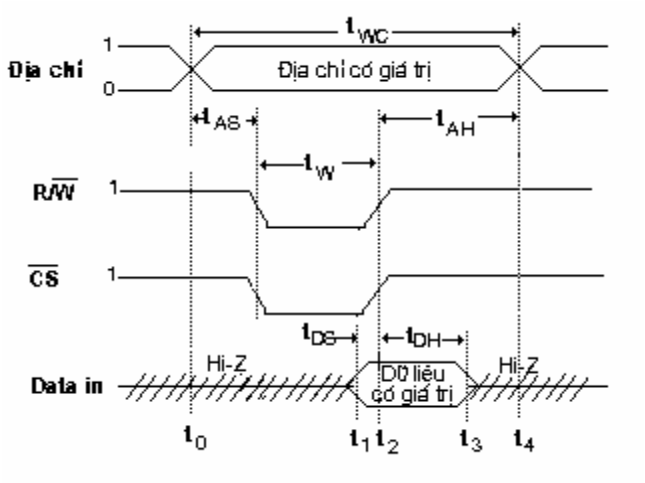
Một chu kỳ viết t_{WC} bao gồm:

- t_{AS} (Address Setup time): Thời gian thiết lập địa chỉ : Thời gian để giá trị địa chỉ ổn định trên bus địa chỉ cho tới lúc tín hiệu $\overline{CS}$ tác động.

- t_W (Write time): Thời gian từ lúc tín hiệu $\overline{CS}$ tác động đến lúc dữ liệu có giá trị trên bus dữ liệu.

- t_{DS} và t_{DH}: Khoảng thời gian dữ liệu tồn tại trên bus dữ liệu bao gồm thời gian trước (t_{DS}) và sau (t_{DH}) khi tín hiệu $\overline{CS}$không còn tác động

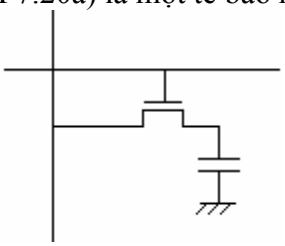
- t_{AH} (Address Hold time): Thời gian giữ địa chỉ: từ lúc tín hiệu $\overline{CS}$không còn tác động đến lúc xuất hiện địa chỉ mới.



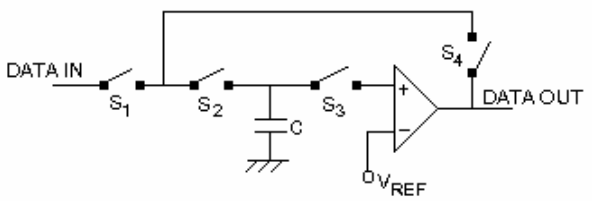
(H 7.19)

7.3.3.2 RAM động (Dynamic RAM, DRAM)

(H 7.20a) là một tế bào nhớ của DRAM



(a)



(H 7.20) (b)

(H 7.20b) là một cách biểu diễn tế bào nhớ DRAM trong đó đơn giản một số chi tiết được dùng để mô tả các tác vụ viết và đọc tế bào nhớ này.

Các khóa từ S_1 đến S_4 là các transistor MOS được điều khiển bởi các tín hiệu ra từ mạch giải mã địa chỉ và tín hiệu $R/\overline{W}$.

Để ghi dữ liệu vào tế bào, các khóa S_1 và S_2 đóng trong khi S_3 và S_4 mở. Bit 1 thực hiện việc nạp điện cho tụ C và bit 0 làm tụ C phóng điện. Sau đó các khóa sẽ mở để cô lập C với phần mạch còn lại. Một cách lý tưởng thì C sẽ duy trì trạng thái của nó vĩnh viễn nhưng thực tế luôn luôn có sự rỉ điện qua các khóa ngay cả khi chúng mở do đó C bị mất dần điện tích .

Để đọc dữ liệu các khóa S_2 , S_3 , S_4 đóng và S_1 mở, tụ C nối với một mạch so sánh với một điện thế tham chiếu để xác định trạng thái logic của nó. Điện thế ra mạch so sánh chính là dữ liệu được đọc ra. Do S_2 và S_4 đóng, dữ liệu ra được nối ngược lại tụ C để làm tươi nó. Nói cách khác, bit dữ liệu trong tế bào nhớ được làm tươi mỗi khi nó được đọc.

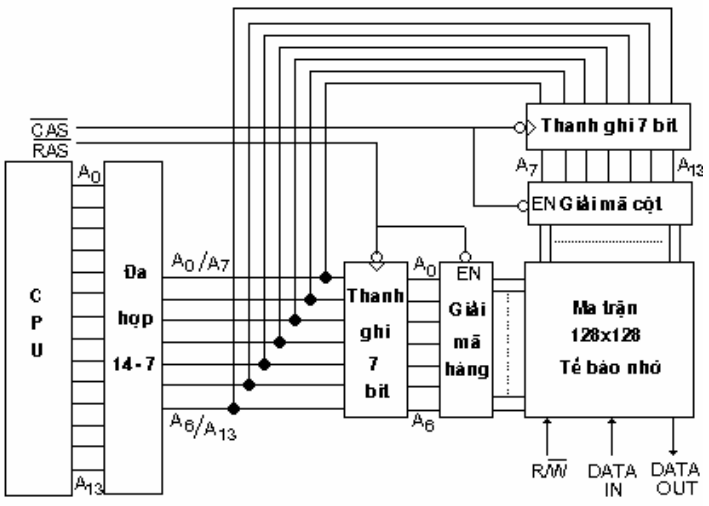
Sử dụng DRAM, được một thuận lợi là dung lượng nhớ khá lớn nhưng phải có một số mạch phụ trợ:

- Mạch đa hợp địa chỉ vì DRAM luôn sử dụng địa chỉ hàng và cột
- Mạch làm tươi để phục hồi dữ liệu có thể bị mất sau một khoảng thời gian ngắn nào đó.

a. Đa hợp địa chỉ

Như đã nói trên, do dung lượng của DRAM rất lớn nên phải dùng phương pháp đa hợp để chọn một vị trí nhớ trong DRAM. Mỗi vị trí nhớ sẽ được chọn bởi 2 địa chỉ hàng và cột lần lượt xuất hiện ở ngã vào địa chỉ.

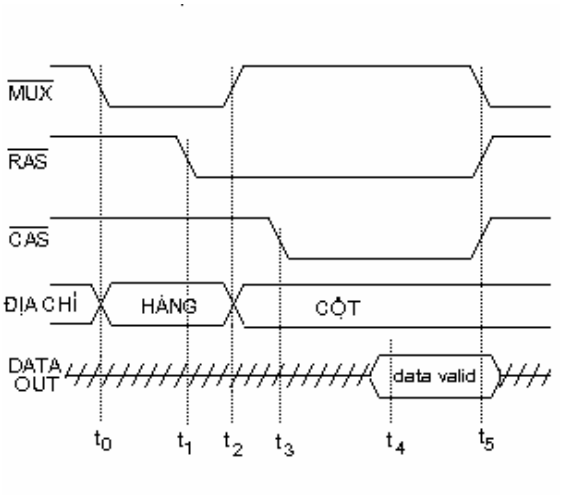
Thí dụ với DRAM có dung lượng 16Kx1, thay vì phải dùng 14 đường địa chỉ ta chỉ cần dùng 7 đường và mạch đa hợp 14 $\rightarrow$ 7 (7 đa hợp 2$\rightarrow$1) để chọn 7 trong 14 đường địa chỉ ra từ CPU (H 7.21). Bộ nhớ có cấu trúc là một ma trận 128x128 tế bào nhớ, sắp xếp thành 128 hàng và 128 cột, có một ngã vào và một ngã ra dữ liệu, một ngã vào $R/\overline{W}$. Hai mạch chốt địa chỉ (hàng và cột) là các thanh ghi 7 bit có ngã vào nối với ngã ra mạch đa hợp và ngã ra nối với các mạch giải mã hàng và cột. Các tín hiệu $\overline{RAS}$ và $\overline{CAS}$ dùng làm xung đồng hồ cho mạch chốt và tín hiệu Enable cho mạch giải mã. Như vậy 14 bit địa chỉ từ CPU sẽ lần lượt được chốt vào các thanh ghi hàng và cột bởi các tín hiệu $\overline{RAS}$ và $\overline{CAS}$ rồi được giải mã để chọn tế bào nhớ. Vận hành của hệ thống sẽ được thấy rõ hơn khi xét các giản đồ thời gian của DRAM.



(H 7.21)

b. Giản đồ thời gian của DRAM

(H 7.22) là giản đồ thời gian đọc và viết tiêu biểu của DRAM (Hai giản đồ này chỉ khác nhau về thời lượng nhưng có chung một dạng nên ta chỉ vẽ một)



(H 7.22)

Giản đồ cho thấy tác động của tín hiệu $\overline{MUX}$ và các tín hiệu $\overline{RAS}$ và $\overline{CAS}$. Khi $\overline{MUX}$ ở mức thấp mạch đa hợp cho ra địa chỉ hàng (A_0 . . . A_6) và được chốt vào thanh ghi khi tín hiệu $\overline{RAS}$ xuống thấp. Khi $\overline{MUX}$ ở mức cao mạch đa hợp cho ra địa chỉ cột (A_7 . . . A_{13}) và được chốt vào thanh ghi khi tín hiệu $\overline{CAS}$ xuống thấp. Khi cả địa chỉ hàng và cột đã được giải mã, dữ liệu tại địa chỉ đó xuất hiện trên bus dữ liệu để đọc ra hoặc ghi vào (khả dụng)

c. Làm tươi DRAM

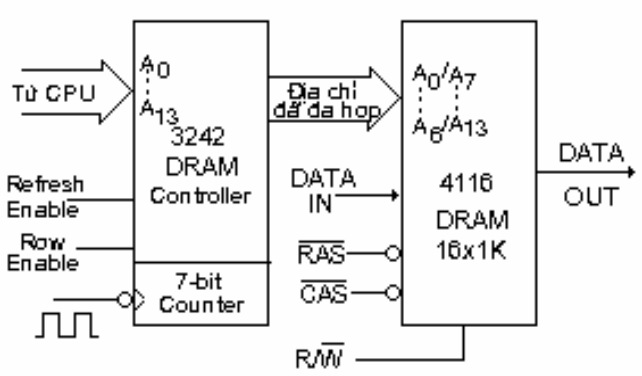
DRAM phải được làm tươi với chu kỳ khoảng 2ms để duy trì dữ liệu.

Trong phần trước ta đã thấy tế bào nhớ DRAM được làm tươi ngay khi tác vụ đọc được thực hiện. Lấy thí dụ với DRAM có dung lượng 16Kx1 (16.384 tế bào) nói trên, chu kỳ làm tươi là 2 ms cho 16.384 tế bào nhớ nên thời gian đọc mỗi tế bào nhớ phải là 2 ms/16.384 = 122 ns. Đây là thời gian rất nhỏ không đủ để đọc một tế bào nhớ trong điều kiện vận hành bình thường. Vì lý do này các hãng chế tạo đã thiết kế các chip DRAM sao cho **mỗi khi tác vụ đọc được thực hiện đối với một tế bào nhớ, tất cả các tế bào nhớ trên cùng một hàng sẽ được làm tươi**. Điều này làm giảm một lượng rất lớn tác vụ đọc phải thực hiện để làm tươi tế bào nhớ. Trở lại thí dụ trên, tác vụ đọc để làm tươi phải thực hiện cho 128 hàng trong 2 ms. Tuy nhiên để vừa vận hành trong điều kiện bình thường vừa phải thực hiện chức năng làm tươi người ta phải dùng thêm mạch phụ trợ, gọi là **điều khiển DRAM** (DRAM controller)

IC 3242 của hảng Intel thiết kế để sử dụng cho DRAM 16K (H 7.23)

Ngã ra 3242 là địa chỉ 7 bit đã được đa hợp và nối vào ngã vào địa chỉ của DRAM. Một mạch đếm 7 bit kích bởi xung đồng hồ riêng để cấp địa chỉ hàng cho DRAM trong suốt thời gian làm tươi. 3242 cũng lấy địa chỉ 14 bit từ CPU đa hợp nó với địa chỉ hàng và cột đã được dùng khi CPU thực hiện tác vụ đọc hay viết. Mức logic áp dụng cho các ngã REFRESH ENABLE và ROW ENABLE xác định 7 bit nào của địa chỉ xuất hiện ở ngã ra mạch controller cho bởi bảng

REFRESH ENABLE	ROW ENABLE	Controller output
HIGH	X	Refresh address (từ mạch đếm)
LOW	HIGH	Địa chỉ hàng (A_0 . . . A_6 từ CPU)
LOW	LOW	Địa chỉ cột (A_7 . . .A_{13} từ CPU)



(H 7.23)

7.4 MỞ RỘNG BỘ NHỚ

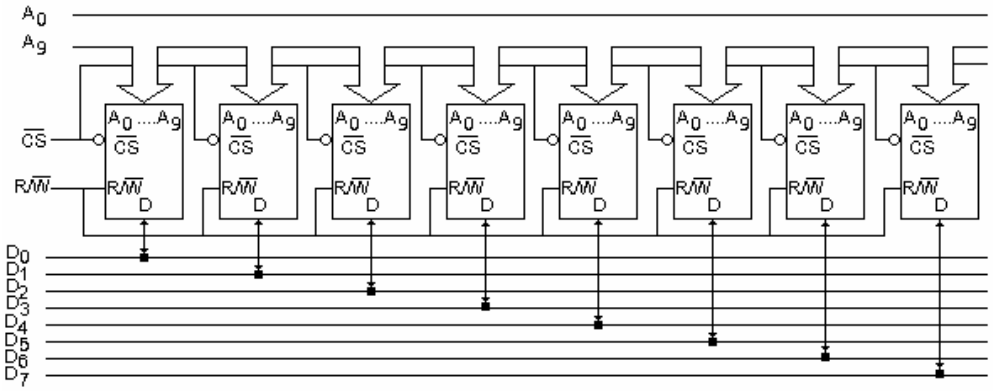
Các IC nhớ thường được chế tạo với dung lượng nhớ có giới hạn, trong nhiều trường hợp không thể thỏa mãn yêu cầu của người thiết kế. Do đó mở rộng bộ nhớ là một việc làm cần thiết. Có 3 trường hợp phải mở rộng bộ nhớ.

7.4.1. Mở rộng độ dài từ

Đây là trường hợp số vị trí nhớ đủ cho yêu cầu nhưng dữ liệu cho mỗi vị trí nhớ thì không đủ. Có thể hiểu được cách mở rộng độ dài từ qua một thí dụ

Thí dụ: Mở rộng bộ nhớ từ 1Kx1 lên 1Kx8 :

Chúng ta phải dùng 8 IC nhớ 1Kx1, các IC nhớ này sẽ được nối chung bus địa chỉ và các đường tín hiệu điều khiển và mỗi IC quản lý một đường bit. 8 IC sẽ vận hành cùng lúc để cho một từ nhớ 8 bit (H 7.24).

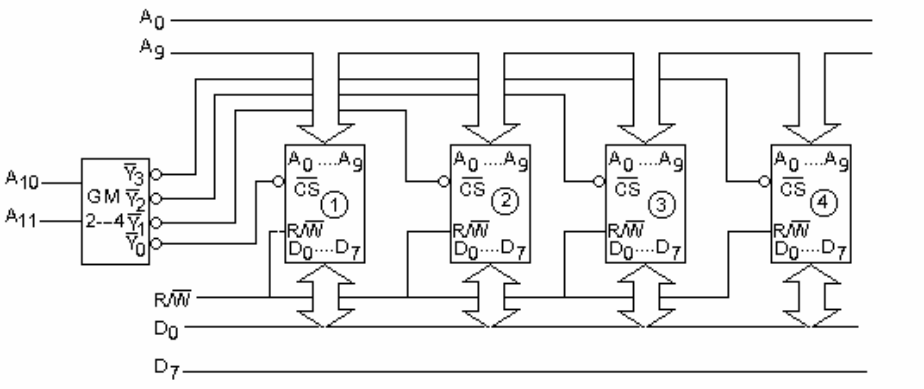


(H 7.24)

7.4.2 Mở rộng vị trí nhớ

Số bit cho mỗi vị trí nhớ đủ theo yêu cầu nhưng số vị trí nhớ không đủ

Thí dụ: Có IC nhớ dung lượng 1Kx8. Mở rộng lên 4Kx8. Cần 4 IC. Để chọn 1 trong 4 IC nhớ cần một mạch giải mã 2 đường sang 4 đường, ngã ra của mạch giải mã lần lượt nối vào các ngã $\overline{CS}$ của các IC nhớ, như vậy địa chỉ của các IC nhớ sẽ khác nhau (H 7.25). Trong thí dụ này IC1 chiếm địa chỉ từ 000H đến 3FFH, IC2 từ 400H đến 7FFH, IC3 từ 800H đến BFFH và IC4 từ C00H đến FFFH

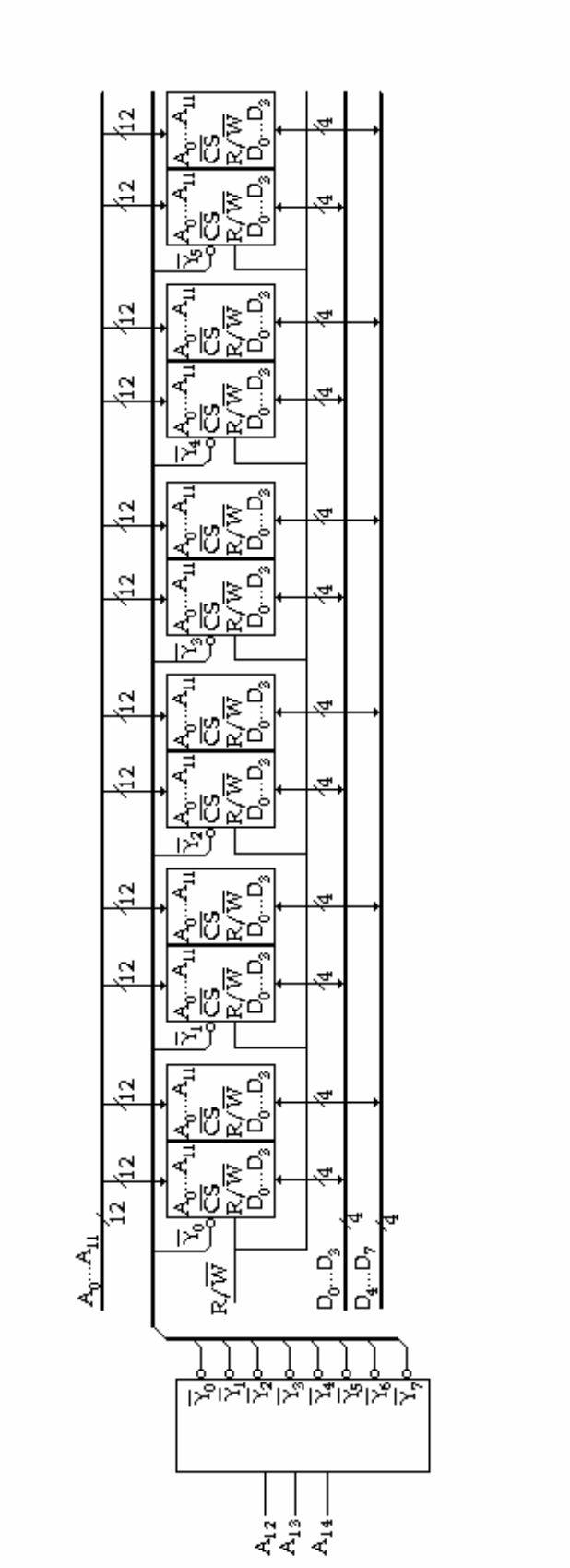


(H 7.25)

7.4.3 Mở rộng dung lượng nhớ

Cả vị trí nhớ và độ dài từ của các IC đều không đủ để thiết kế. Để mở rộng dung lượng nhớ ta phải kết hợp cả hai cách nói trên

Thí dụ: Mở rộng bộ nhớ từ 4Kx4 lên 24Kx8. Cần 6 cặp IC mắc song song, mỗi cặp IC có chung địa chỉ và được chọn bởi một mạch giải mã 3 sang 8 đường (H 7.26). Ta chỉ dùng 6 ngã ra từ Y_0 đến Y_5 của mạch giải mã



(H 7.26)

- Địa chỉ IC (1&2): 0000H - 0FFFH, IC (3&4) : 1000H - 1FFFH, IC (5&6): 2000H - 2FFFH và IC (7&8) : 3000H - 3FFFH IC (9&10): 4000H - 4FFFH và IC (11&12) : 5000H - 5FFFH

BÀI TẬP

1. Dùng IC PROM 4 ngã vào và 4 ngã ra thiết kế mạch chuyển mã từ Gray sang nhị phân của số 4 bit.

2. Dùng IC PAL 4 ngã vào và 4 ngã ra thiết kế mạch chuyển từ mã Excess-3 sang mã Aiken của các số từ 0 đến 9.

Dưới đây là 2 bảng mã

Excess-3

N	A			D
0	0	0	1	1
1	0	1	0	0
2	0	1	0	1
3	0	1	1	0
4	0	1	1	1
5	1	0	0	0
6	1	0	0	1
7	1	0	1	0
8	1	0	1	1
9	1	1	0	0

Aiken

A			D
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

3. Thiết kế mạch để mở rộng bộ nhớ từ 2Kx4 lên 2Kx8

4. Thiết kế mạch để mở rộng bộ nhớ từ 1Kx4 lên 8Kx4.
Cho biết địa chỉ cụ thể của các IC

5. Thiết kế mạch để mở rộng bộ nhớ từ 2Kx4 lên 16Kx8.
Cho biết địa chỉ cụ thể của các IC

✹ CHƯƠNG 8 : BIẾN ĐỔI AD & DA

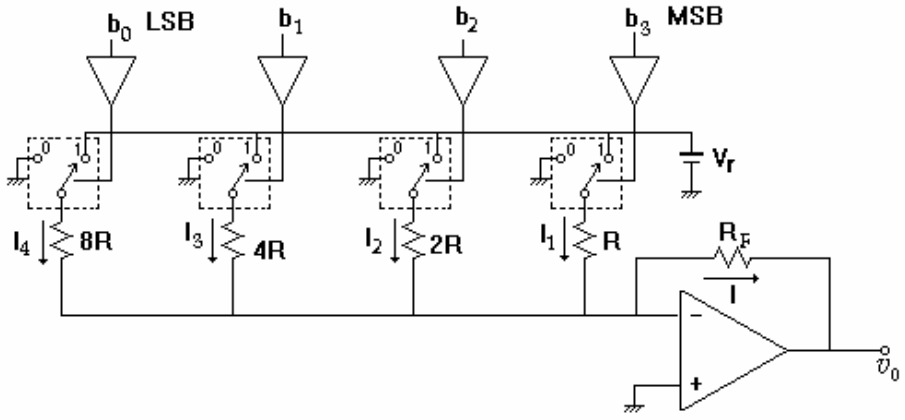
⌖ **BẾN ĐỔI SỐ - TƯƠNG TỰ (DAC)**
♦ DAC dùng mạng điện trở có trọng lượng khác nhau
♦ DAC dùng mạng điện trở hình thang
♦ DAC dùng nguồn dòng có trọng lượng khác nhau
♦ Đặc tính kỹ thuật của DAC
⌖ **BIẾN ĐỔI TƯƠNG TỰ - SỐ (ADC)**
♦ Mạch lấy mẫu và giữ
♦ Nguyên tắc mạch ADC
♦ ADC dùng điện thế tham chiếu nấc thang
♦ ADC gần đúng kế tiếp
♦ ADC dốc đơn
♦ ADC tích phân
♦ ADC lưỡng cực
♦ ADC song song

Có thể nói sự biến đổi qua lại giữa các tín hiệu từ dạng tương tự sang dạng số là cần thiết vì:

- Hệ thống số xử lý tín hiệu số mà tín hiệu trong tự nhiên là tín hiệu tương tự: cần thiết có mạch đổi tương tự sang số.
- Kết quả từ các hệ thống số là các đại lượng số: cần thiết phải đổi thành tín hiệu tương tự để có thể tác động vào các hệ thống vật lý và thể hiện ra bên ngoài (thí dụ tái tạo âm thanh hay hình ảnh) hay dùng vào việc điều khiển sau đó (thí dụ dùng điện thế tương tự để điều khiển vận tốc động cơ)

8.1. Biến đổi số - tương tự (digital to analog converter, ADC)

8.1.1 Mạch biến đổi DAC dùng mạng điện trở có trọng lượng khác nhau (Weighted resistor network)



(H 8.1)

Trong mạch trên, nếu thay OP-AMP bởi một điện trở tải, ta có tín hiệu ra là dòng điện.

Như vậy OP-AMP giữ vai trò biến dòng điện ra thành điện thế ra, đồng thời nó là một mạch cộng

Ta có $v_0 = -R_F.I = -(2^3b_3 + 2^2b_2 + 2b_1+b_0)Vr.R_F/2^3R$

$= -(2^{n-1} b_{n-1} + 2^{n-2} b_{n-2} + ...+ 2b_1 + b_0)Vr.R_F /2^{n-1}.R$

Nếu $R_F = R$ thì:

$v_0 =-(2^{n-1} b_{n-1} + 2^{n-2} b_{n-2} + ...+ 2b_1 + b_0)Vr. /2^{n-1}.$

Thí dụ:

1/ Khi số nhị phân là 0000 thì $v_0 = 0$

1111 thì $v_0 = -15Vr / 8$

2/ Với Vr = 5V ; $R = R_F = 1k\Omega$

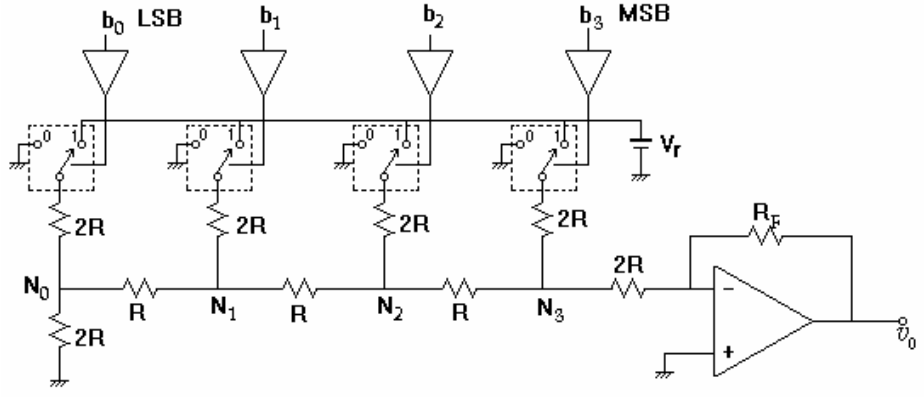
Ta có kết quả chuyển đổi như sau:

b_3	b_2	b_1	b_0	v_0 (V)
0	0	0	0	0
0	0	0	1	-0,625 ← LSB
0	0	1	0	-1,250
0	0	1	1	-1,875
0	1	0	0	-2.500
0	1	0	1	-3,125
0	1	1	0	-3,750
0	1	1	1	-4,375
1	0	0	0	-5,000
1	0	0	1	-5,625
1	0	1	0	-6,250
1	0	1	1	-6,875
1	1	0	0	-7,500
1	1	0	1	-8,125
1	1	1	0	-8,750
1	1	1	1	-9,375 ← Full Scale (V_{FS})

Mạch có một số hạn chế:

- Sự chính xác tùy thuộc vào điện trở và mức độ ổn định của nguồn tham chiếu Vr
- Với số nhị phân nhiều bit thì cần các điện trở có giá trị rất lớn, khó thực hiện.

8.1.2 Mạch đổi DAC dùng mạng điện trở hình thang



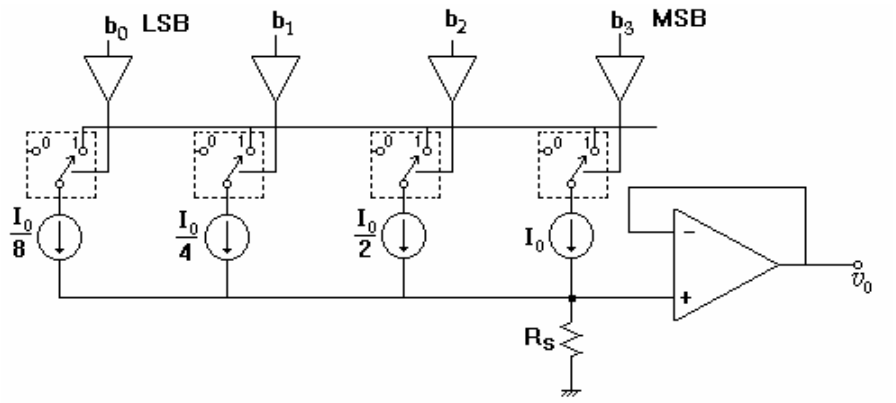
(H 8.2)

Cho $R_F = 2R$ và lần lượt

Cho $b_3 = 1$ các bit khác = 0, ta được: $v_0 = -8(Vr /24)$

Cho $b_2 = 1$ các bit khác = 0, ta được: $v_0 = -4(Vr/24)$
Cho $b_1 = 1$ các bit khác = 0, ta được: $v_0 = -2(Vr/24)$
Cho $b_0 = 1$ các bit khác = 0, ta được: $v_0 = -(Vr/24)$
Ta thấy v_0 tỉ lệ với giá trị B của tổ hợp bit $B = (b_3\ b_2\ b_1\ b_0)_2 \Rightarrow v_0 = -B(Vr/24)$

8.1.3 Mạch đổi DAC dùng nguồn dòng có trọng lượng khác nhau



(H 8.3)

8.1.4 Đặc tính kỹ thuật của mạch đổi DAC

8.1.4.1. Bit có ý nghĩa thấp nhất (LSB) và bit có ý nghĩa cao nhất (MSB)

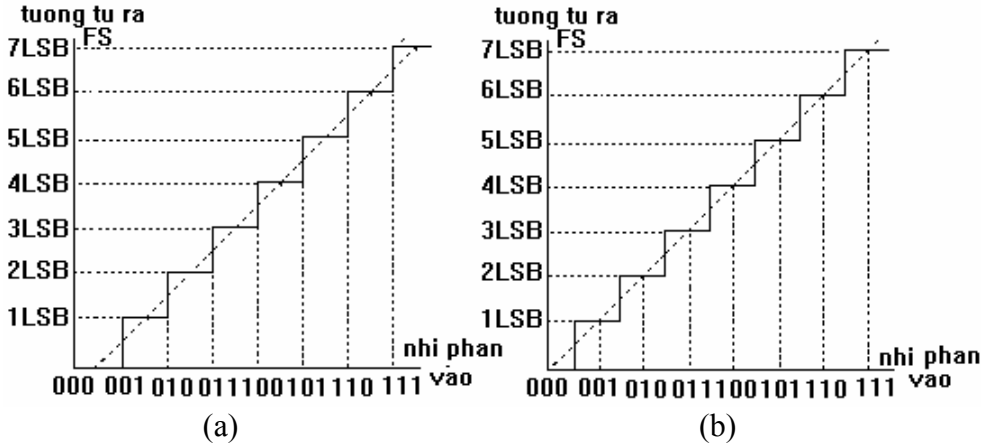
Qua các mạch biến đổi DAC kể trên ta thấy vị trí khác nhau của các bit trong số nhị phân cho giá trị biến đổi khác nhau, nói cách khác trị biến đổi của một bit tùy thuộc vào trọng lượng của bit đó.

Nếu ta gọi trị toàn giai là V_{FS} thì bit LSB có giá trị là: $LSB = V_{FS} / (2^n - 1)$

và bit $MSB = V_{FS} .2^{n-1} / (2^n - 1)$

Điều này được thể hiện trong kết quả của thí dụ 2 ở trên.

(H 8.4) là đặc tuyến chuyển đổi của một số nhị phân 3 bit



(a) (b)

(H 8.4)

(H 8.4a) là đặc tuyến lý tưởng, tuy nhiên, trong thực tế để đường trung bình của đặc tính chuyển đổi đi qua điểm 0 điện thế tương tự ra được làm lệch (1/2)LSB (H 8.4b). Như vậy điện thế tương tự ra được xem như thay đổi ở ngay giữa hai mã số nhị phân vào kế nhau. Thí

dụ khi mã số nhị phân vào là 000 thì điện thế tương tự ra là 0 và điện thế tương tự ra sẽ lên nấc kế 000+(1/2)LSB rồi nấc kế tiếp ở 001+(1/2)LSB.v.v....Trị tương tự ra ứng với 001 gọi tắt là 1LSB và trị toàn giai V_{FS} = 7LSB tương ứng với số 111

8.1.4.2 Sai số nguyên lượng hóa (quantization error)

Trong sự biến đổi, ta thấy ứng với một giá trị nhị phân vào, ta có một khoảng điện thế tương tự ra. Như vậy có một sai số trong biến đổi gọi là sai số nguyên lượng hóa và =(1/2)LSB

8.1.4.3. Độ phân giải (resolution)

Độ phân giải được hiểu là giá trị thay đổi nhỏ nhất của tín hiệu tương tự ra có thể có khi số nhị phân vào thay đổi. Độ phân giải còn được gọi là trị bước (step size) và bằng trọng lượng bit LSB.

Số nhị phân n bit có 2^n giá trị và 2^n - 1 bước

Hiệu thế tương tự ra xác định bởi $v_0 = k.(B)_2$

Trong đó k chính là độ phân giải và $(B)_2$ là số nhị phân

Người ta thường tính phần trăm phân giải:

$$\%res = (k / V_{FS})100\ \%$$

Với số nhị phân n bit

$$\%res = [1 / (2^n - 1)]100\ \%$$

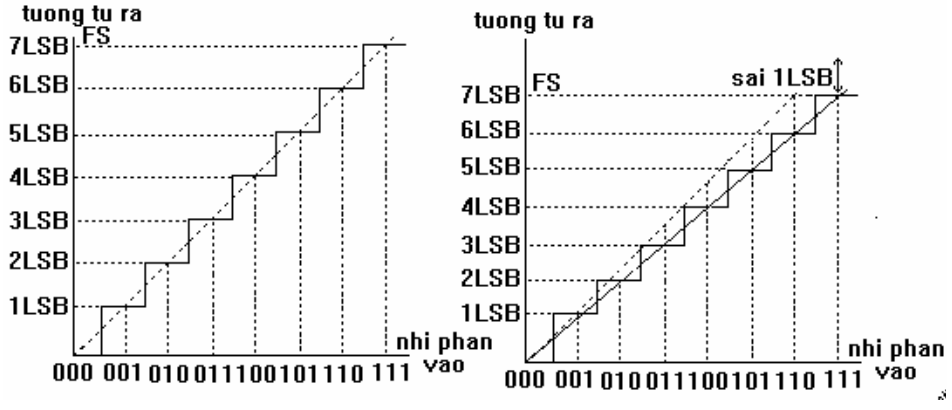
Các nhà sản xuất thường dùng số bit của số nhị phân có thể được biến đổi để chỉ độ phân giải. Số bit càng lớn thì độ phân giải càng cao (finer resolution)

8.1.4.4. Độ tuyến tính (linearity)

Khi điện thế tương tự ra thay đổi đều với số nhị phân vào ta nói mạch biến đổi có tính tuyến tính

8.1.4.5. Độ đúng (accuracy)

Độ đúng (còn gọi là độ chính xác) tuyệt đối của một DAC là hiệu số giữa điện thế tương tự ra và điện thế ra lý thuyết tương ứng với mã số nhị phân vào. Hai số nhị phân kế nhau phải cho ra hai điện thế tương tự khác nhau đúng 1LSB, nếu không mạch có thể tuyến tính nhưng không đúng (H 8.5)



a/ Tuyến tính b/ Tuyến tính nhưng không đúng

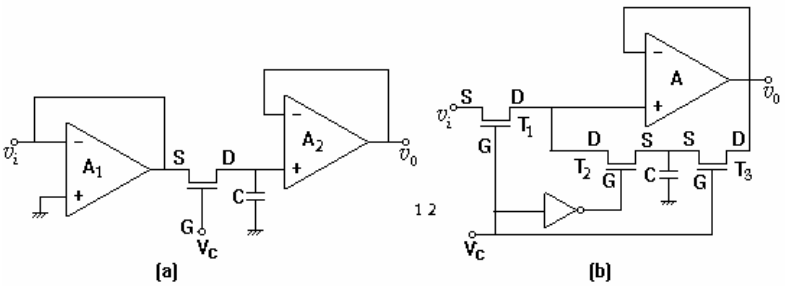
(H 8.5)

8.2. Biến đổi tương tự - số (analog to digital converter, ADC)

8.2.1 Mạch lấy mẫu và giữ (sample anh hold)

Để biến đổi một tín hiệu tương tự sang tín hiệu số, người ta không thể biến đổi mọi giá trị của tín hiệu tương tự mà chỉ có thể biến đổi một số gía trị cụ thể bằng cách **lấy mẫu** tín hiệu đó theo một chu kỳ xác định nhờ một tín hiệu có dạng xung. Ngoài ra, mạch biến đổi cần một khoảng thời gian cụ thể (khoảng 1μs - 1ms) do đó **cần giữ mức tín hiệu** biến đổi trong khoảng thời gian này để mạch có thể thực hiện việc biến đổi chính xác. Đó là nhiệm vụ của mạch lấy mẫu và giữ.

(H 8.6) là dạng mạch lấy mẫu và giữ cơ bản: Điện thế tương tự cần biến đổi được lấy mẫu trong thời gian rất ngắn do tụ nạp điện nhanh qua tổng trở ra thấp của OP-AMP khi các transistor dẫn và giữ giá trị này trong khoảng thời gian transistor ngưng (tụ phóng rất chậm qua tổng trở vào rất lớn của OP-AMP)

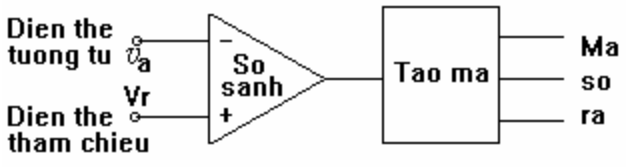


(H 8.6)

8.2.2 Nguyên tắc mạch biến đổi ADC

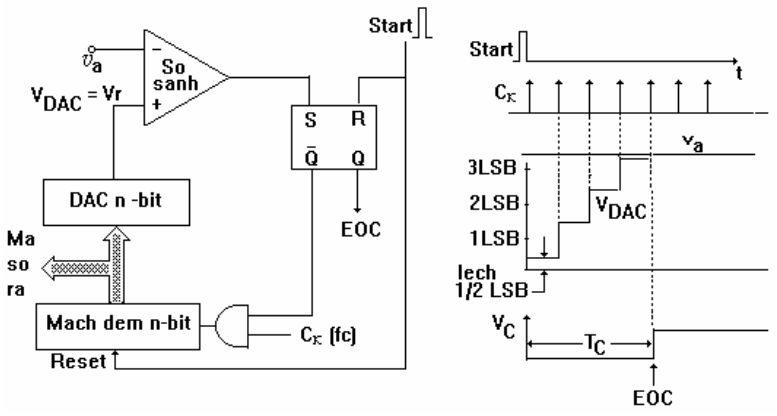
Mạch biến đổi ADC gồm bộ phận trung tâm là một mạch so sánh (H 8.7). Điện thế tương tự chưa biết v_a áp vào một ngã vào của mạch so sánh, còn ngã vào kia nối đến một điện thế tham chiếu thay đổi theo thời gian Vr(t). Khi chuyển đổi điện thế tham chiếu tăng theo thời gian cho đến khi bằng hoặc gần bằng với điện thế tương tự (với một sai số nguyên lượng hóa). Lúc đó mạch tạo mã số ra có giá trị ứng với điện thế vào chưa biết. Vậy nhiệm vụ của mạch tạo mã số là thử một bộ số nhị phân sao cho hiệu số giữa v_a và trị nguyên lượng hóa sau cùng nhỏ hơn 1/2 LSB

$$|v_a - (V_{FS} / 2^n - 1)(B)_2 | < 1/2 \text{ LSB}$$



(H 8.7)

8.2.3 Mạch đổi dùng điện thế tham chiếu nấc thang



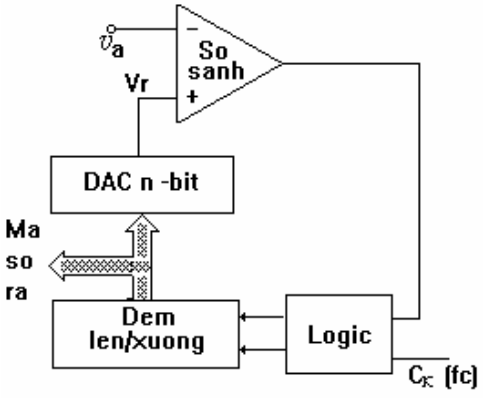
(a) (H 8.8) (b)

Một cách đơn giản để tạo điện thế tham chiếu có dạng nấc thang là dùng một mạch DAC mà số nhị phân vào được lấy từ mạch đếm lên (H 8.8). Khi có xung bắt đầu FlipFlop và mạch đếm được đặt về 0 nên ngã ra $\overline{Q}$ của FF lên 1, mở cổng AND cho xung C_K vào mạch đếm. Ngã ra mạch đếm tăng dần theo dạng nấc thang (V_{DAC}), đây chính là điện thế tham chiếu, khi Vr còn nhỏ hơn v_a, ngã ra mạch so sánh còn ở mức thấp và $\overline{Q}$ vẫn tiếp tục ở mức cao, nhưng khi Vr vùa vượt v_a ngã ra mạch so sánh lên cao khiến $\overline{Q}$ xuống thấp, đóng cổng AND không cho xung C_K qua và mạch đếm ngưng. Đồng thời ngã ra Q lên cao báo kết thúc sự chuyển đổi. Số đếm ở mạch đếm chính là số nhị phân tương ứng với điện thế vào.

Gọi thời gian chuyển đổi là t_c. Thời gian chuyển đổi tùy thuộc điện thế cần chuyển đổi. Thời gian lâu nhất ứng với điện thế vào bằng trị toàn giai:

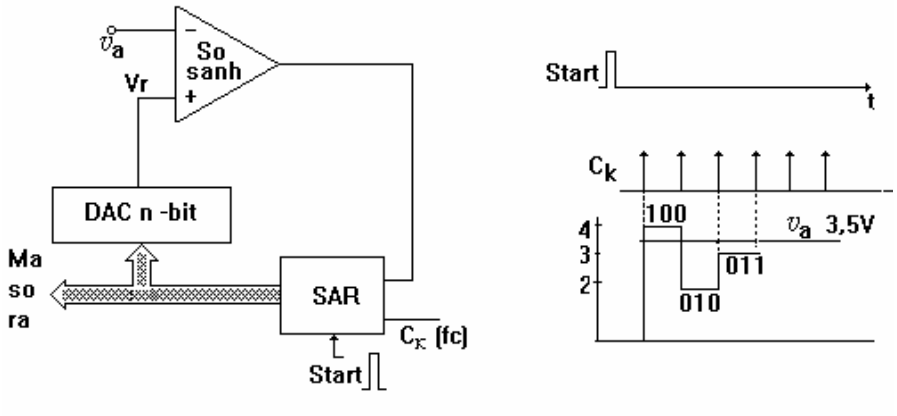
$$t_c(max) = 2^n / f_{CK} = 2^n . T_{CK}$$

Mạch đổi này có tốc độ chậm. Một cách cải tiến là thay mạch đếm lên bởi một mạch đếm lên/xuống (H 8.9). Nếu ngã ra mạch so sánh cho thấy Vr nhỏ hơn v_a, mạch Logic sẽ điều khiển đếm lên và ngược lai thì mạch sẽ đếm xuống. Nếu v_a không đổi Vr sẽ dao động quanh trị v_a với hai trị số khác nhau 1 LSB



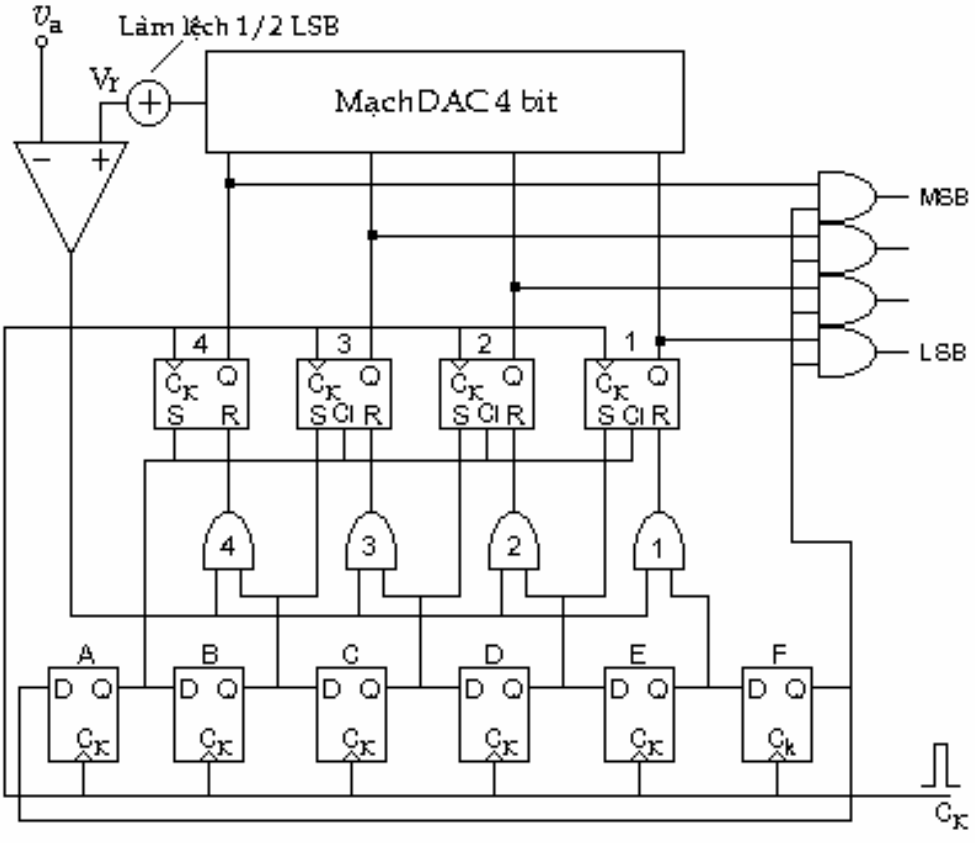
(H 8.9)

8.2.4 Mạch đổi lấy gần đúng kế tiếp (sucessive approximation converter)



(H 8.10)

Mạch đổi lấy gần đúng kế tiếp dùng cách tạo điện thế tham chiếu một cách có hiệu quả hơn khiến việc chuyển đổi ra mã số n bit chỉ tốn n chu kỳ xung C_K. Mạch này bao gồm: một mạch so sánh, một mạch ghi dịch đặc biệt (SAR) và một mạch DAC (H 8.11).



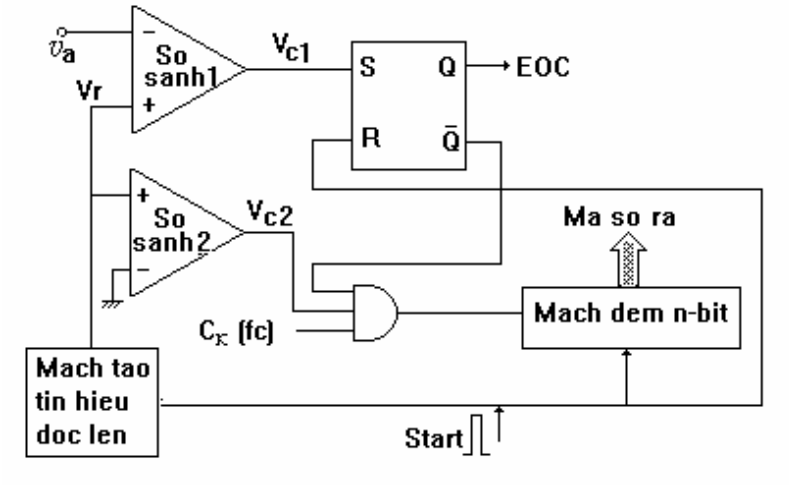
(H 8.11)

Mạch SAR (H 8.11) là mạch ghi dịch có kết hợp điều khiển Logic. Mạch gồm 6 FF D mắc thành chuỗi, ngã ra FF cuối (F) hồi tiếp về FF đầu (A) , khối điều khiển gồm 4 cổng AND và 4 FF RS có ngã vào tác động mức cao, các ngã ra Q của các FF RS được đưa vào mạch DAC để tạo điện thế tương tự Vr (dùng so sánh với điện thế ra từ mạch lấy mẫu và giữ v_a), đồng thời đây cũng là mã số ra khi sự biến đổi đã kết thúc.

Vận hành: Lúc có xung bắt đầu, mạch SAR được đặt về 0. Ngã ra DAC được làm lệch 1/2 LSB để tạo đặc tính chuyển đổi như đã nói trong phần trước, kế đó SAR đưa bit MSB lên cao (bằng cách preset FF A), các bit khác bằng 0, số này được đưa vào mạch DAC để tạo điện thế tham chiếu Vr để so sánh với v_a. Tùy theo kết quả so sánh, nếu Vr > v_a thì ngã ra mạch so sánh ở mức cao khiến SAR bỏ đi bit MSB khi có xung C_K kế tiếp xuất hiện, còn nếu Vr < v_a thì ngã ra mạch so sánh ở mức thấp, khiến SAR giữ bit MSB lại (FF RS 4 giữ nguyên trạng thái) đồng thời đưa bit có nghĩa kế tiếp lên cao (do FF 3 được set từ giá trị 1 ở ngã ra FF B, trị 1 này được chuyển từ FF A sang). Mạch so sánh tiếp tục làm việc và kết quả sẽ được quyết định theo cùng cách thức như đối với bit MSB.... Tiếp tục như vậy cho đến bit cuối cùng của SAR, lúc đó v_a gần Vr nhất và ta được kết quả chuyển đổi trong thời gian tối đa là n chu kỳ xung đồng hồ. Mạch chuyển đổi chấm dứt khi ngã ra FF F lên mức cao cho phép mở các đệm để cho mã số ra.

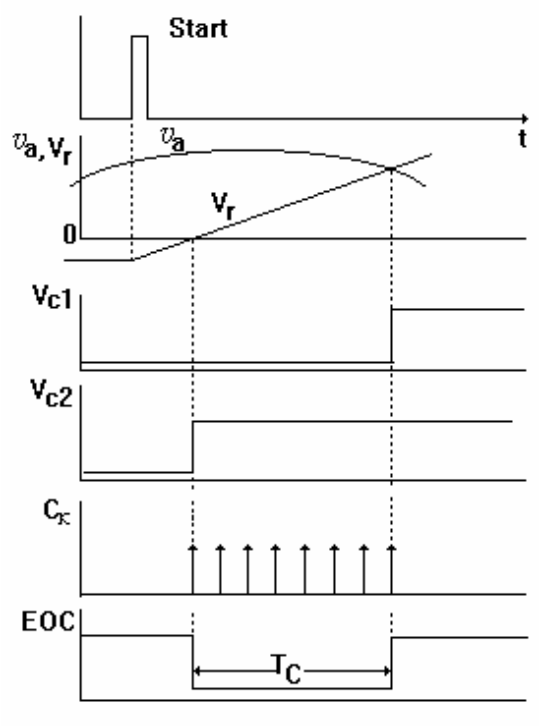
8.2.5 Mạch đổi dùng tín hiệu dốc đơn (single ramp converter)

Điện thế chuẩn từng nấc tạo bởi mạch DAC có thể được thay thế bởi điện thế tham chiếu có dốc lên liên tục tạo bởi mạch tạo tín hiệu dốc lên (thường là mạch tích phân).



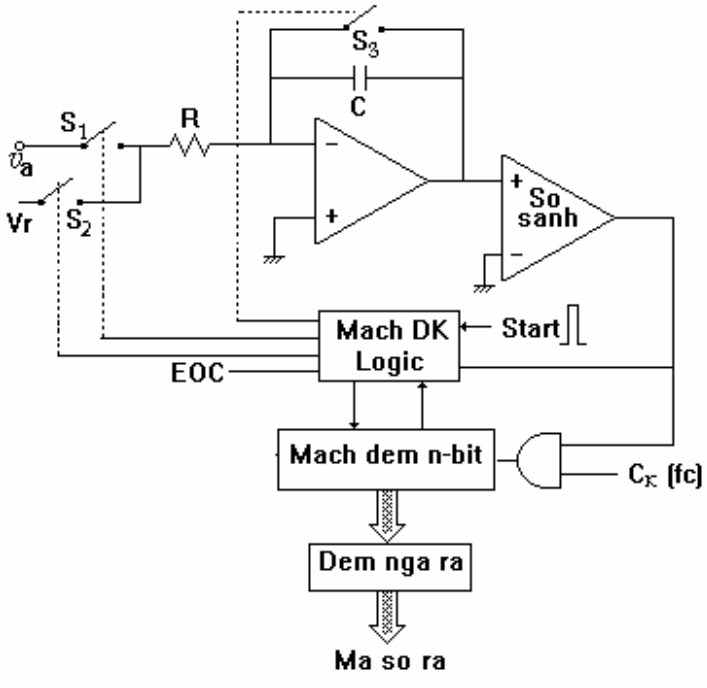
(H 8.12)

Xung bắt đầu đặt mạch đếm n bit về 0 và khởi động mạch tạo dốc lên để tạo Vr, từ một trị hơi âm, khi Vr cắt trục 0 ngã ra mạch so sánh 2 lên cao mở cổng AND cho xung C_K vào mạch đếm. Khi đường dốc đạt trị số bằng trị tương tự cần biến đổi ngã ra mạch so sánh 1 lên cao đưa ngã ra $\overline{Q}$ của FF xuống thấp, cổng AND đóng và kết thúc sự chuyển đổi. Số đếm được ở mạch đếm tỷ lệ với điện thế tương tự vào. Mạch có khuyết điểm là độ dốc của Vr tùy thuộc thông số RC của mạch tích phân nên không chính xác.



(H 8.13)

8.2.6 Mạch đổi lấy tích phân (Integrating Converter)



(H 8.14)

Mạch này giải quyết được khuyết điểm của mạch biến đổi dùng tín hiệu dốc đơn, nghĩa là độ chính xác không tùy thuộc RC.

Xung bắt đầu đưa mạch đếm về 0, mạch điều khiển mở khóa S_3 của mạch tích phân, đóng khóa S_1 để đưa tín hiệu tương tự v_a (giả sử âm) vào mạch tích phân đồng thời mở khóa S_2. Ngã ra mạch tích phân có trị âm nhỏ ban đầu. Tín hiệu tương tự vào được lấy tích phân, độ dốc $-v_a$ /RC. Khi ngã ra mạch tích phân vượt trục 0, ngã ra mạch so sánh lên cao mở cổng

AND đưa xung C_K vào mạch đếm. Không kể lượng lệch âm ban đầu, hiệu thế ngã ra mạch tích phân là:

$$V_I(t) = \int -\frac{v_a}{RC} dt$$

Giả sử v_a không đổi trong thời gian chuyển đổi

$$V_I(t) = -(v_a.t /RC)$$

Nếu v_a âm thì ngã ra mạch tích phân là đường dốc lên đều.

Khi mạch đếm tràn (tức đếm hết dung lượng và tự động quay về 0) mạch Logic điều khiển mở khóa S_1 và đóng khóa S_2 đưa điện thế tham chiếu Vr (dương) đến mạch lấy tích phân. Ngã ra mạch tích phân bây giờ là đường dốc xuống với độ dốc là -Vr /RC. Khi V_I xuống 0, mạch so sánh xuống thấp đóng cổng AND và kết thúc quá trình biến đổi. Số đếm sau cùng của mạch đếm tỷ lệ với điện thế tương tự vào.

Giả sử RC không đổi trong quá trình biến đổi, tích phân trong thời gian t_1 bằng tích phân trong thời gian t_2 nên ta có:

$$| v_a | t_1 = Vr.t_2$$

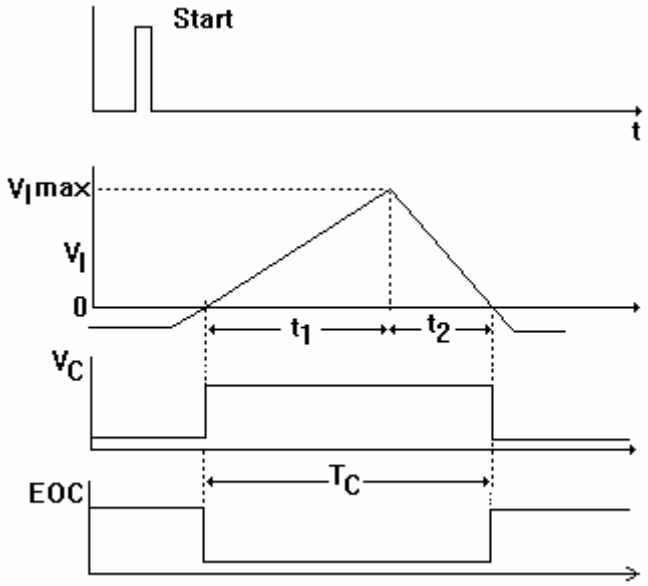
t_1 là thời gian đếm từ 0 cho đến khi tràn nên

$$t_1 = 2^n / f_{CK}$$

và $t_2 = N / f_{CK}$.

N là số đếm sau cùng.

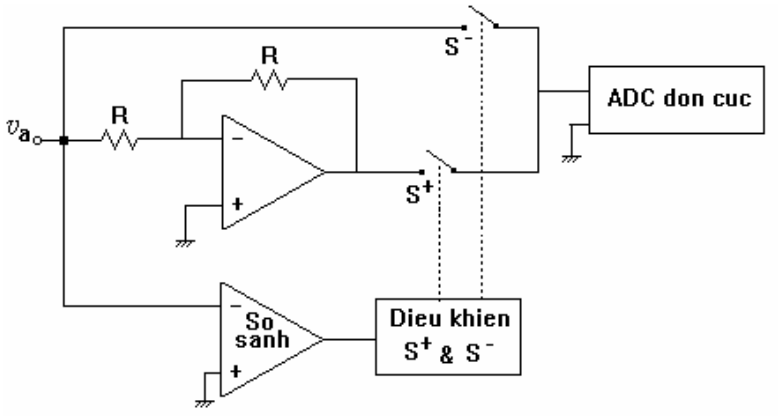
Tóm lại ta thấy số đếm được không phụ thuộc RC



(H 8.15)

8.2.7 Mạch đổi lưỡng cực

Một cách đơn giản để thực hiện chuyển đổi một tín hiệu tương tự lưỡng cực là dùng một mạch đảo tương tự và một mạch so sánh để xác định v_a âm hay dương để đảo hay không trước khi đưa vào mạch ADC đơn cực (H 8.16)



(H 8.16)

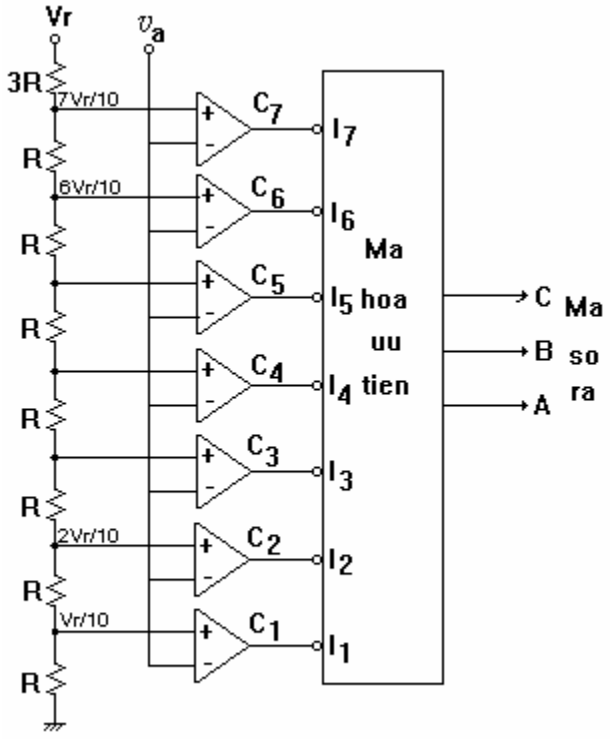
8.2.8 Mạch đổi song song (parallel hay flash conversion)

Đây là mạch đổi có tốc độ chuyển đổi rất nhanh, có thể đạt vài triệu lần trong một giây, áp dụng vào việc chuyển đổi tín hiệu hình trong kỹ thuật video. Thí dụ để có mạch đổi 3 bit, người ta dùng 7 mạch so sánh ở ngã vào và một mạch mã hóa ưu tiên để tạo mã số nhị phân ở ngã ra (H 8.17).

- Khi v_a < Vr /10, các ngã ra mạch so sánh đều lên cao khiến mã số ra là 000
- Khi Vr /10 <v_a < 2Vr /10, ngã ra mạch so sánh 1 xuống thấp khiến mã số ra là 001

- Khi 2Vr /10 <v_a < 3Vr /10, ngã ra mạch so sánh 2 xuống thấp khiến mã số ra là 010

Cứ như thế, ta thấy mã số ra tỷ lệ với điện thế tương tự vào



(H 8.17)