Trường CĐSP Bến Tre

Tổ Tin Học

Giáo trình

KỸ THUẬT SỐ

Chủ biên **Võ Thanh Ân**

Lưu hành nội bộ

Bến Tre, Năm 2004

CHƯƠNG 1: CÁC HỆ THỐNG SỐ

- ✓ NGUYÊN LÝ CỦA VIỆC VIẾT SỐ
- ✓ CÁC HỆ THỐNG SỐ
- ✓ BIẾN ĐỔI QUA LẠI GIỮA CÁC HỆ THỐNG SỐ
- ✓ CÁC PHÉP TOÁN SỐ NHỊ PHÂN
- ✓ MÃ HOÁ
  - Mã BCD
  - Mã Gray

I. GIỚI THIỆU

Nhu cầu về định lượng nhất là trong những trao đổi thương mại, đã có từ khi xã hội hình thành. Đã có 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…

Việc sử dụng các hệ thống số hằng ngày quá quen thuộc, khiến chúng ta quên đi sự hình thành và các qui tắc viết các con số.

Phần này nhắc lại một cách sơ lượt về nguyên lý của việc viết số và giới thiệu các hệ thống số khác ngoài hệ thống thập phân quen thuộc. Chúng ta sẽ đặt biệt chú ý đến hệ thống nhị phân là hệ thống được dùng trong lĩnh vực tin học – điện tử.

II. NGUYÊN LÝ CỦA VIỆC VIẾT SỐ

Một số được viết bằng cách đặt kề nhau các ký tự được chọn trong một tập hợp. Mỗi ký hiệu trong mỗi số được gọi là một số mã (số hạng – digit).

Ví dụ, trong hệ thống thập phân, 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 số mã tuỳ thuộc vị trí của nó trong số đó. Giá trị này được gọi là trọng số của số mã. Ví dụ, số 1998 trong hệ thập phân, số 9 đầu sau số 1 có trọng số là 900 trong khi số 9 thứ hai chỉ là 90.

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

S_b = {S_0, S_1, S_2, … S_{b-1} }

Một số n trong hệ b được viết dưới dạng:

N = ($a_n a_{n-1} a_{n-2} \ldots a_i \ldots a_1 a_0, a_{-1} a_{-2} \ldots a_{-m}$) với $a_i \in S$.

Sẽ có giá trị:

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

III. CÁC HỆ THỐNG SỐ

1. Hệ thập phân – Decimal system – Cơ số 10

Hệ thập phân dùng 10 chữ số: 0 1 2 3 4 5 6 7 8 9 để biểu diễn các số.

Ví dụ: Tính giá trị của **1 234 567** trong hệ thập phân.

Biểu diễn theo công thức tổng quát:

$1\,234\,567 = 1*10^6 + 2*10^5 + 3*10^4 + 4*10^3 + 5*10^2 + 6*10^1 + 7*10^0$

1 234 567 = 1 000 000 + 200 000 + 30 000 + 4 000 + 500 + 60 + 7

2. Hệ nhị phân – Binary system – Cơ số 2

Hệ nhị phân dùng 2 chữ số : **0 1** để biểu diễn các số.

Ví dụ: Tính giá trị của số **100 111** trong hệ nhị phân.

Biểu diễn theo công thức tổng quát:

$100\ 111_{Bin} = 1*2^5 + 0*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 1*2^0$

$100\ 111_{Bin} = 100\ 000_{Bin} + 00\ 000_{Bin} + 0\ 000_{Bin} + 100_{Bin} + 10_{Bin} + 1$

Nếu đổi sang cơ số 10 ta được:

$100\ 111_{Bin} \Leftrightarrow 32_{Dec} + 0_{Dec} + 0_{Dec} + 4_{Dec} + 2_{Dec} + 1_{Dec}$

$100\ 111_{Bin} \Leftrightarrow 39_{Dec}$

3. Hệ bát phân – Octal system – Cơ số 8

Hệ bát phân dùng 8 chữ số: **0 1 2 3 4 5 6 7** để biểu diễn các số.

Ví dụ: Tính giá trị của số **123 456** trong hệ bát phân.

Biểu diễn theo công thức tổng quát:

$123\ 456_{Oct} = 1*8^5 + 2*8^4 + 3*8^3 + 4*8^2 + 5*8^1 + 6*8^0$

$123\ 456_{Oct} = 100\ 000_{Oct} + 20\ 000_{Oct} + 3\ 000_{Oct} + 400_{Oct} + 50_{Oct} + 6_{Oct}$

Nếu đổi sang cơ số 10 ta được:

$123\ 456_{Oct} \Leftrightarrow 32768_{Dec} + 8192_{Dec} + 1536_{Dec} + 256_{Dec} + 40_{Dec} + 6_{Dec}$

$123\ 456_{Oct} \Leftrightarrow 42\ 798_{Dec}$

4. Hệ thập lục phân – Hexadecimal system – Cơ số 16

Hệ thập lục phân dùng 16 chữ số: **0 1 2 3 4 5 6 7 8 9 A B C D E F** để biểu diễn các số.

Ví dụ: Tính giá trị của số 4B trong hệ thập lục phân.

Biểu diễn theo công thức tổng quát:

$4B_{Hex} = 4*16^1 + B*16^0$

$4B_{Hex} = 40_{Hex} + B_{Hex}$

Nếu theo cơ số 10 ta có:

$4B_{Hex} \Leftrightarrow 64_{Dec} + 11_{Dec}$

$4B_{Hex} \Leftrightarrow 75_{Dec}$

IV.BIẾN ĐỔI QUA LẠI GIỮA CÁC CƠ SỐ

1. Đổi một cơ số từ hệ b sang hệ 10

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

Một số N trong hệ b được viết:

$N_b = a_n a_{n-1} ... a_i ... a_0 a_{-1} a_{-2} ... a_{-m}$ với $a_i \in S_b$

Có giá trị tương ứng với hệ cơ số 10 là:

$$N_{10} = a_n b^n + a_{n-1} b^{n-1} + ... + a_i b^i + ... + a_0 b^0 + a_{-1} b^{-1} + a_{-2} b^{-2} + ... + a_{-m} b^{-m} = \sum_{i=-m}^{n} a_i b^i$$

Ví dụ 1: Đổi số 1010,11 ở cơ số 2 sang cơ số 10 ta làm như sau:

$1011{,}11_2 \Leftrightarrow 1.2^3 + 0.2^2 + 1.2^1 + 1.2^0 + 1.2^{-1} + 1.2^{-2}$

$1011{,}11_2 \Leftrightarrow 8 + 0 + 4 + 1 + 0{,}5 + 0{,}25$

$1011{,}11_2 \Leftrightarrow 13{,}75_{10}$

Ví dụ 2: Đổi giá trị của số 4B,8F trong hệ thập lục phân sang hệ thập phân.

$4B,8F_{16} \Leftrightarrow 4*16^{1} + B*16^{0} + 8*16^{-1} + 15*16^{-2}$

$4B,8F_{16} \Leftrightarrow 64 + 11 + 0,5 + 0.05859375$

$4B,8F_{16} \Leftrightarrow 75,55859375_{10}$

2. Đổi một cơ số từ hệ 10 sang hệ b

Đây là bài toán tìm một dãy các ký hiệu cho số N viết trong hệ b. Một số N viết trong dạng cơ số 10 và viết trong cơ số b có dạng như sau:

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

Trong đó:

$(a_n a_{n-1} \ldots a_0)_b$ = PE(N) là phần nguyên của N.

$(0, a_{-1} a_{-2} \ldots a_{-m})_b$ = PF(N) là phần thập phân của N.

Có 2 cách biến đổi khác nhau cho phần nguyên và phần thập phân.

- Phần nguyên – PE(N)

Phần nguyên có thể viết lại như sau:

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

Ta thấy rằng, nếu lấy PE(N) chia cho b thì ta sẽ có số dư là a_0, được thương là $PE'(N) = (a_n b^{n-1} + a_{n-1} b^{n-2} + \ldots + a_1) b$. Vậy số dư của lần thứ nhất này chính là bit có trọng số nhỏ nhất (bit LSB).

Tiếp tục cho đến khi được phép chia cuối cùng, đó chính là bit lớn nhất (MSB).

- Phần thập phân – PF(N)

Phần thập phân có thể được viết lại như sau:

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

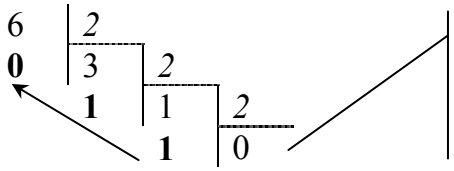
Ta thấy rằng nếu nhân PF(N) với b ta được $a_{-1} + a_{-2} b^{-1} + \ldots + a_{-m} b^{-m+1} = a_{-1} + PF'(N)$. Vậy a_{-1} chính là bit lẽ đầu tiên của phần thập phân.

Tiếp tục lặp lại bài toán nhân phần lẽ của kết quả có được của phép nhân trước đó với b cho tới khi kết quả phần lẽ bằng 0, ta tìm được dãy số $(a_{-1} a_{-2} a_{-3} \ldots a_{-m})$.

Chú ý: Phần thập phân của số N khi đổi sang hệ b có thể gồm vô số số hạng (do kết quả phần thập phân có được luôn khác 0), vậy tuỳ theo yêu cầu về độ chính xác của kết quả mà ta lấy một số số hạng nhất định.

Ví dụ: Đổi số 6,3 sang hệ nhị phân.

Phần nguyên ta thực hiện như sau:



Kết quả phép chia bằng không (kết thúc). Lấy ngược phần dư ta được: $110_{Bin} \Leftrightarrow 6_{Dec}$

Phần thập phân ta thực hiện như sau:

0,3*2 = 0,6	➔ $a_{-1} = 0$	Lấy phần chẵn là 0
0,6*2 = 1,2	➔ $a_{-2} = 1$	Lấy phần chẵn là 1
0,2*2 = 0,4	➔ $a_{-3} = 0$	
0,4*3 = 0,8	➔ $a_{-4} = 0$	
0,8*2 = 1,6	➔ $a_{-5} = 1$	
0,6*2 = 1,2	➔ $a_{-6} = 1$	
0,2*2 = 0,4	➔ $a_{-7} = 0$	(tiếp tục…)

Như vậy kết quả bài toán nhân luôn luôn khác 0, nếu kết quả bài toán chỉ cần 5 số lẽ thì ta lấy PF(N) = 0,01001.

Kết quả cuối cùng là: $6,3_{10} \Leftrightarrow 110,01111_2$

3. Đổi một cơ số từ hệ b sang hệ b^k

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 2 phía và đặt thành thừa số chung.

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

Giả sử k =3 số N được viết lại như sau:

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

Phần chứa trong mỗi dấu ngoặc luôn nhỏ hơn b^k (k=3), vậy số này chính là một số trong hệ b^k và được biểu diễn bởi các ký hiệu tương ứng trong hệ này.

Ví dụ 1: Đổi số 10011101010,10011 từ hệ cơ số 2 sang hệ cơ số 8 (k=3 vì $8 = 2^3$)

Từ dấu phẩy gom từng 3 số, ta có thể thêm số 0 vào bên trái của số hoặc bên phải sau dấu phẩy cho đủ nhóm 3 (k=3) số, ta được như sau:

010 **011** 101 **010**, 100 **110**$_{(2)} \Leftrightarrow 2352,46_{(8)}$

Ví dụ 2: Đổi số 10011101010,10011 từ hệ cơ số 2 sang hệ cơ số 16 (k=4 vì $16 = 2^4$)

Từ dấu phẩy gom từng 4 số, ta có thể thêm số 0 vào bên trái của số hoặc bên phải sau dấu phẩy cho đủ nhóm 4 (k=4) số, ta được như sau:

0100 1110 **1010**, 1001 **1000**$_{(2)} \Leftrightarrow 4EA,98_{(16)}$

Ngoài ra, ta cũng có thể biến đổi một số từ b^k sang b^p thực hiện trung gian qua hệ b. Điều này dễ dàng suy ra từ 2 ví dụ trên, đọc giả tự nghiên cứu.

Dưới đây là bảng kê các số đầu tiên trong 4 hệ số thường gặp:

Thập phân	Nhị phân	Bát phân	Thập lục phân
0	00000	0	0
1	00001	1	1
2	00010	2	2
3	00011	3	3
4	00100	4	4
5	00101	5	5
6	00110	6	6
7	00111	7	7
8	01000	10	8
9	01001	11	9
10	01010	12	A

Thập phân	Nhị phân	Bát phân	Thập lục phân
11	01011	13	B
12	01100	14	C
13	01101	15	D
14	01110	16	E
15	01111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
21	10101	25	15

V. CÁC PHÉP TÍNH TRONG HỆ NHỊ PHÂN

1. Giới thiệu

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

2. Phép cộng

Là phép tính làm cơ sở cho các phép tính khác. Ta có các chú ý sau:

$0 + 0 = 0$

$0 + 1 = 1 + 0 = 1$

$1 + 1 = 0$, nhớ 1 (đem qua bit cao hơn).

Ngoài ra để thực hiện bài toán cộng nhiều số ta nên nhớ:

- Nếu số bit số 1 chẵn thì kết quả bằng 0.
- Nếu số bit số 1 lẽ thì kết quả bằng 1.
- Cứ 1 cặp số 1, cho 1 số nhớ.

Ví dụ: Tính 011 + 101 + 011 + 011

```
      11     ← số nhớ
     111     ← số nhớ
    ----
     011
+    101
     011
     011
    ----
    1110
```

3. Phép trừ

Ta có các chú ý sau:

$0 - 0 = 0$

$1 - 1 = 0$

$1 - 0 = 1$

$0 - 1 = 1$, nhớ 1 cho bit cao hơn.

Ví dụ: Tính 1011 – 0101

```
      1      ← số nhớ
    ----
     1011
–    0101
    ----
     0110
```

4. Phép nhân

Ta có các chú ý sau:

$0 \times 0 = 0$

$0 \times 1 = 0$

$1 \times 1 = 1$

Ví dụ: Tính 110×101

```
        110
  ×     101
  ------------
        110
  +   000
      110
  ------------
      11110
```

5. Phép chia

Tương tự như phép chia trong hệ cơ số 10.

Ví dụ: Tính 1001100100 : 11000

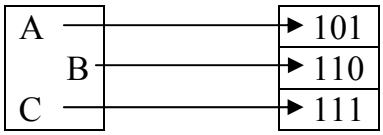
```
1001100100 | 11000
–11000     | 11001,1
-----------
0011100
 –11000
 -------
  00100100
   –11000
   -------
    001100 0   ← thêm 0 vào để
     –1100 0   chia lấy phần lẽ.
     -------
      0000 0
```

VI.MÃ HOÁ

1. Tổng quát

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

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



Tập nguồn có thể là tập hợp các số, các ký tự, dấu, các lệnh dùng trong truyền dữ liệu… và tập 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 một từ mã. Tập hợp các từ mã tạo ra theo cùng một qui luật cho ta bộ mã. Việc chọn mã tuỳ vào mục đích sử dụng.

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

Công việc ngược lại mã hoá là giải mã.

Cách biểu diễn các số trong trong các hệ khác nhau cũng được xem là một hình thức mã hoá, như vậy, ta có mã thập phân, nhị phân, thập lục phân… và việc chuyển từ mã này sang mã khác cũng thuộc bài toán mã hoá.

Trong kỹ thuật số ta thường sử dụng mã BCD và mã Gray. Ta sẽ xét chúng ở phần ngay sau đây.

2. Mã BCD (Binary Coded Decimal)

Mã BCD dùng số 4 bit nhị phân thay thế cho từng số hạng trong số thập phân.

Ví dụ: Số $729_{(10)}$ có mã BCD là **0111** 0010 **1001**$_{(BCD)}$

Mã BCD rất thuận lợi để mạch điện tử đọc các giá trị thập phân và hiển thị bằng các đèn bảy đoạn (led 7 đoạn) và các thiết bị sử dụng kỹ thuật số khác.

Hình: Led 7 đoạn.

3. Mã Gray

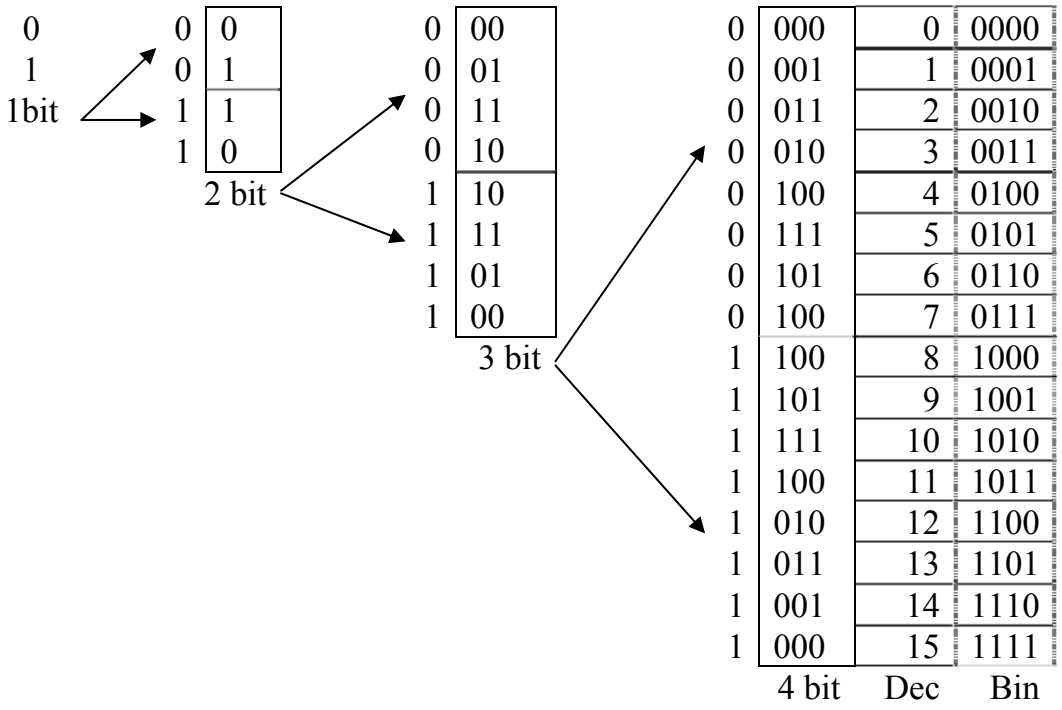
Mã Gray hay còn họi là **mã cách khoảng đơn vị**.

Nếu quan sát thông tin từ máy đếm, đang đếm sự kiện tăng dần từng đơn vị của một số nhị phân. Ta sẽ được các số nhị phân dần dần thay đổi. Tại thời điểm quan sát, có thể có những lỗi rất quan trọng, ví dụ từ số 7 (0111) và số 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 các trạng thái liên tiếp sau chẳn hạn:

0111 → 0101 → 0100 → 1100 → 1001

Trong một quan sát ngắn, kết quả thấy được khác nhau. Để tránh hiện tượng này, người ta cần mã hoá mỗi số hạng sau cho 2 số liên tiếp chỉ khác nhau một phần tử nhị phân (1 bit) gọi là **mã cách khoảng đơn** vị hay mã **Gray** và 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 thành lập mã Gray dựa vào tính chất đối xứng của nó. Để thực hiện mã Gray nhiều bit, ta thực hiện từ tập mã Gray 1 bit. Ta làm như sau:



Ta có một cách khác để xác định một số mã Gray tương ứng với mã nhị phân như sau:

- Xác định số nhị phân tương ứng với Gray cần tìm.

- Dịch trái số nhị phân 1 bit sau đó cộng không số nhớ với số nhị phân đó, bỏ bit cuối.

Ví dụ: Xác định số 14 của mã Gray ta làm như sau:
Xác định số nhị phân tương ứng: $14_{(10)} \Leftrightarrow 1110_{(2)}$
Dịch trái 1 bit số $1110_{(2)}$ ta được số $11100_{(2)}$, sau đó cộng bỏ bít cuối như sau:

$$\begin{array}{rll} & 111\cancel{0} & \leftarrow \text{Số nhị phân tương ứng } 14_{(10)} \\ + & 1110\cancel{0} & \leftarrow \text{Số nhị phân tương ứng } 14_{(10)} \text{ dịch trái 1 bít.} \\ \hline & 1001 & \leftarrow \text{Số mã Gray (cộng hai số trên không số nhớ và bỏ bít cuối).} \end{array}$$

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

I. HÀM LOGIC CƠ BẢN

1. Một số định nghĩa

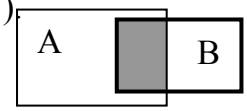
- Trạng thái logic được biểu diễn bằng số 0 hoặc 1.
- Biến logic là đại lượng biểu diễn bởi một ký hiệu (chữ hay dấu) chỉ gồm các giá trị 0 hay 1 tuỳ theo điều kiện nào đó.
- Hàm logic diễn tả một nhóm biến logic liên hệ với nhau bởi các phép toán logic. Cũng như biến logic, hàm logic chỉ nhận 1 giá trị 0 hoặc 1.

2. Biểu diễn biến và hàm logic

a. Giản đồ Venn

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

Ví dụ: Phần giao nhau của 2 tập hợp A và B (màu xám) biểu diễn tập hợp trong đó A và B đúng (A and B = 1).



b. 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 những tổ hợp của n biến, có cả thảy 2^n tổ hợp có thể có. Các cột ghi tên biến, cột cuối cùng ghi tên hàm và giá trị của hàm tương ứng với các tổ hợp biến trên cùng hàng.

Ví dụ: Hàm **F(A,B) = A OR B** có bảng sự thật như sau:

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

c. Bảng Karnaugh

Đây là cách biểu diễn khác của bảng sự thật trong đó mỗi hàng của bảng sự thật được thay thế bởi 1 ô mà tọa độ hàng và cột có giá trị xác định bởi tổ hợp đã cho của biến.

Bảng Karnaugh của hàm có n biến gồm 2^n ô. Bảng Karnaugh rất thuận tiện để đơn giản hàm logic bằng cách nhóm các ô lại với nhau.

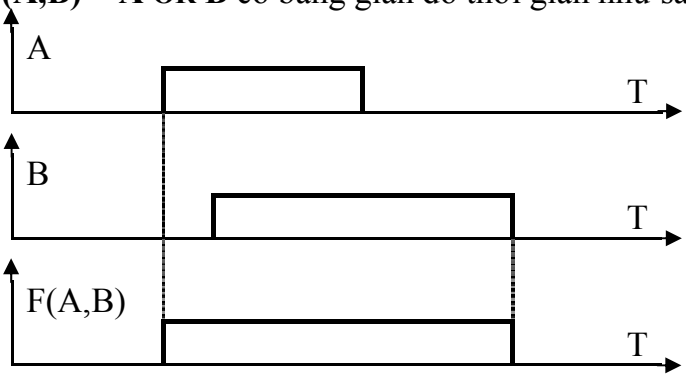
Ví dụ: Hàm **F(A,B) = A OR B** có bảng Karnaugh như sau:

A \ B	0	1
0	0	1
1	1	1

d. Giản đồ thời gian

Dùng để diễn tả quan hệ giữa hàm và biến theo thời gian.

Ví dụ: Hàm **F(A,B) = A OR B** có bảng giản đồ thời gian như sau:



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.

Ví dụ: Trong một hệ thống số có 2 giá trị điện áp 0V (thấp) và 5V (cao), ta có thể chọn một trong hai qui ước sau:

Điện áp	Logic dương	Logic âm
0V	1	0
5V	0	1

4. Các hàm logic cơ bản

a. Hàm NOT (đảo, bù)

Phép toán (gạch trên): $\overline{\ }$

Bảng sự thật dưới đây: $Y = \overline{A}$

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

b. Hàm OR (hoặc)

Phép toán: + (cộng).
Bảng sự thật dưới đây.

A	B	F(A,B) = A + B
0	0	0
0	1	1
1	0	1
1	1	1

c. Hàm AND (và)

Phép toán: • (nhân – dấu chấm).
Bảng sự thật dưới đây.

A	B	F(A,B) = A.B
0	0	0
0	1	0
1	0	0
1	1	1

d. Hàm EX–OR (OR loại trừ)

Phép toán: ⊕ (exor).
Bảng sự thật dưới đây.

A	B	F(A,B) = A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

5. Tính chất của các hàm logic cơ bản

a. Tính chất cơ bản

- Có một phần tử trung tính duy nhất cho mỗi toán tử + (cộng) và . (nhân).
  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 chất 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.
  Phép nhân: A . (B + C) = A . B + A . C
  Phép cộng: A + (B . C) = (A + B) . (A + C)
- Không có phép tính lũy thừa và thừa số.

A + A + … + A = A

A . A . … . A = A

- Tính bù.

$\overline{\overline{A}} = A$

$A + \overline{A} = 1$

$A.\overline{A} = 0$

b. Tính song đối (duality)

Tất cả các biểu thức logic vẫn đúng khi ta thay phép toán + (cộng) bởi phép toán • (nhân), 0 bởi 1 hay ngược lại.

Ta hãy xét các ví dụ sau:

$A + B = B + A \quad \Leftrightarrow \quad A . B = B . A$

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

$A + 1 = 1 \quad \Leftrightarrow \quad A . 0 = 0$

c. Định lý De Morgan

Định lý De Morgan được phát biểu bởi 2 biểu thức sau:

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

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

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

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

Định lý De Morgan cho ta thấy các hàm logic không độc lập với nhau. Chúng có thể biến đổi qua lại do đó chúng ta có thể dùng hàm [AND và NOT] hoặc [OR và NOT] để biểu diễn tất cả các hàm.

Ví dụ: Chỉ dùng hàm AND và NOT biểu diễn hàm: $Y = AB + BC + \overline{A}C$

Chỉ việc đảo Y hai lần ta được kết quả: $Y = \overline{\overline{Y}} = \overline{\overline{AB + BC + \overline{A}C}} = \overline{\overline{AB}.\overline{BC}.\overline{\overline{A}C}}$

II. CÁC DẠNG CHUẨN CỦA HÀM LOGIC

1. Giới thiệu

Hàm logic được biểu diễn bởi tổ hợp của những tổng và tích logic.

Nếu là tổng của những tích ta có dạng: $f(X,Y,Z) = XY + XZ + \overline{Y}Z$

Nếu là tích của những tổng ta có dạng: $f(X,Y,Z) = (X + Y)(X + Z)(\overline{Y} + Z)$

Một hàm logic được gọi là hàm chuẩn nếu mỗi số hạng chứa đầy đủ các biến. Ta hãy xem hàm sau: $f(X,Y,Z) = XYZ + X\overline{Y}Z + X\overline{Y}\overline{Z}$ là một tổng chuẩn. Mỗi số hạng của tổng chuẩn gọi là **minterm**.

Ta hãy xem hàm sau: $f(X,Y,Z) = (X + Y + Z)(X + \overline{Y} + Z)(X + \overline{Y} + \overline{Z})$ là một tích chuẩn. Mỗi số hạng của tích chuẩn gọi là **maxterm**.

2. Dạng tổng chuẩn

Để có hàm logic dưới dạng chuẩn ta áp dụng định lý triển khai của Shanon. Dạng tổng chuẩn có thể 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 2 tích như sau:

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

Ví dụ: Ta triển khai với hàm 2 biến f(A, B) như sau:

Khai triển theo biến A: $f(A,B) = A.f(1,B) + \overline{A}.f(0,B)$

Mỗi hàm trong 2 hàm vừa tìm được, tiếp tục khai triển theo biến B:

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

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

Nhân vào ta được: $f(A,B) = AB.f(1,1) + A\overline{B}.f(1,0) + \overline{A}B.f(0,1) + \overline{A}\,\overline{B}.f(0,0)$

Với mỗi cặp *i, j* ta có lượng giá trị *f(i, j)* biểu diễn một giá trị riêng của *f(A, B)* trong bài toán phải giải.

Với hàm 3 biến, khai triển ta được:

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

Khai triển hàm n biến, ta được 2^n số hạng.

Mỗi số hạng trong triển khai là tích của một tổ hợp biến và một trị riêng của hàm. Có hai trường hợp có thể xảy ra:

- Giá trị riêng bằng 1, số hạng thu gọn chỉ còn biến.
  $\overline{A}\,\overline{B}C.f(0,0,1) = \overline{A}\,\overline{B}C$ nếu *f(0,0,1)* = 1.
- Giá trị riêng bằng 0, số hạng nhân hàm bằng 0. Số hạng này biến mất trong biểu thức tổng (theo qui tắc X + 0 = X).
  $\overline{A}\,\overline{B}C.f(0,0,1) = 0$ nếu *f(0,0,1)* = 0 (theo qui tắc X.0 = 0).

Ví dụ: Cho hàm 3 biến A, B, C xác định bởi bảng sự thật sau, viết dạng hàm tổng chuẩn cho hàm:

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

- Hàm Z có trị riêng *f(0,0,1)* = 1 tương ứng với giá trị của tổ hợp biến ở "Hàng 1" là A = 0, B = 0, C = 1. Tổ hợp này là $\overline{A}\,\overline{B}C.f(0,0,1) = \overline{A}\,\overline{B}C.1 = \overline{A}\,\overline{B}C$ là một số hạng trong tổng chuẩn
- Tương tự các tổ hợp (2), (3), (5), (7) cũng là các số hạng của tổng chuẩn.
- Cuối cùng ta có: $Z = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}C + ABC$

3. Dạng tích chuẩn

Để có hàm logic dưới dạng chuẩn ta áp dụng định lý triển khai của Shanon. Dạng tích chuẩn có thể 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 2 tổng như sau:

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

Ví dụ: Ta triển khai với hàm 2 biến f(A, B) như sau:

Khai triển theo biến A: $f(A,B)=[A+f(0,B)].[\overline{A}+f(1,B)]$

Mỗi hàm trong 2 hàm vừa tìm được, tiếp tục khai triển theo biến B:

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

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

Áp dụng tính chất phân bố của phép cộng ta được:

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

Với mỗi cặp *i, j* ta có lượng giá trị *f(i, j)* biểu diễn một giá trị riêng của *f(A, B)* trong bài toán phải giải.

Với hàm 3 biến, khai triển ta được:

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

Khai triển hàm n biến, ta được 2^n số hạng.

Mỗi số hạng trong triển khai là tổng của một tổ hợp biến và một trị riêng của hàm. Có hai trường hợp có thể xảy ra:

- Giá trị riêng bằng 0, số hạng thu gọn chỉ còn biến.
  $[\overline{A}+\overline{B}+C+f(1,1,0)]=\overline{A}+\overline{B}+C$ nếu *f(1,1,0)* = 0 (theo qui tắc X + 0 = X).
- Giá trị riêng bằng 1, số hạng hàm bằng 1. Số hạng này biến mất trong biểu thức tích (theo qui tắc X.1 = X).
  $[\overline{A}+\overline{B}+C+f(1,1,0)]=1$ nếu *f(1,1,0)* = 1 (theo qui tắc X+1 = 1).

Ví dụ: Cho hàm 3 biến A, B, C xác định bởi bảng sự thật sau, viết dạng hàm tích chuẩn cho hàm:

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

- Hàm Z có trị riêng *f(1,1,1)* = 0 tương ứng với giá trị của tổ hợp biến ở "Hàng 0" là A = 0, B = 0, C = 0. Tổ hợp này là:

$[A+B+C+f(1,1,1)]=[A+B+C+0]=A+B+C$ là một số hạng trong tích chuẩn

- Tương tự các tổ hợp (4), (6) cũng là các số hạng của tích chuẩn.
- Cuối cùng ta có: $Z=(A+B+C)(\overline{A}+B+C)(\overline{A}+\overline{B}+C)$

4. 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 ví dụ trên, ta 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 hàm $\overline{Z}$ theo dạng chuẩn thứ nhất, ta được: $\overline{Z}=\overline{A}B\overline{C}+A\overline{B}\overline{C}+AB\overline{C}$
  Lấy bù 2 vế ta được dạng tích chuẩn (tổng chuẩn → tích chuẩn):
  $Z=\overline{\overline{Z}}=\overline{\overline{A}B\overline{C}+A\overline{B}\overline{C}+AB\overline{C}}=(A+B+C)(\overline{A}+B+C)(\overline{A}+\overline{B}+C)$
- Diễn tả hàm $\overline{Z}$ theo dạng chuẩn thứ hai, ta được:
  $\overline{Z}=(A+B+\overline{C})(A+\overline{B}+C)(A+\overline{B}+\overline{C})(\overline{A}+B+\overline{C})(\overline{A}+\overline{B}+\overline{C})$
  Lấy bù 2 vế ta được dạng tổng chuẩn (tích chuẩn → tổng chuẩn):
  $Z=\overline{\overline{Z}}=\overline{(A+B+\overline{C})(A+\overline{B}+C)(A+\overline{B}+\overline{C})(\overline{A}+B+\overline{C})(\overline{A}+\overline{B}+\overline{C})}$
  $=\overline{A}\overline{B}C+\overline{A}B\overline{C}+\overline{A}BC+A\overline{B}C+ABC$

5. 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 của biến được thay bởi một số thập phân tương đương với giá trị nhị phân của chúng. Khi sử dụng cách viết này qui ước trọng lượng của biến phải không được thay đổi.

Ví dụ: Cho hàm Z xác định như trên, tương ứng với dạng chuẩn thứ nhất, hàm lấy giá trị các hàng 1, 2, 3, 5, 7 ta viết Z = Σ(1,2,3,5,7). Tương tự nếu dùng dạng chuẩn thứ 2 ta viết Z = Π(0,4,6).

III. RÚT GỌN HÀM LOGIC

1. Giới thiệu

Để thực hiện một hàm logic bằng mạch điện tử, người ta luôn nghĩ đến việc sử dụng linh kiện một cách 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 ba phương pháp rút gọn hàm logic chủ yếu như sau:

- Phương pháp đại số.

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

2. Phương pháp đại số

Phương pháp này bao gồm việc sử dụng các tính chất của đại số Boole. Người ta thường dùng các đẳng thức (các qui tắc) dưới đây để đơn giản hàm logic.

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

a. Qui tắc 1

Dùng các đẳng thức logic để rút gọn hàm.

Ví dụ: Rút gọn hàm $Z = ABC + AB\overline{C} + A\overline{B}CD$

$$Z = ABC + AB\overline{C} + A\overline{B}CD = \underbrace{ABC + AB\overline{C}}_{(1)} + A\overline{B}CD =$$

$$AB + A\overline{B}CD = A(\underbrace{B + \overline{B}CD}_{(3)}) = A(B + CD)$$

b. 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.

Ví dụ: Rút gọn hàm $Z = ABC + \overline{A}BC + A\overline{B}C + AB\overline{C}$

Thêm ABC vào ta được:

$$Z = \underbrace{ABC + \overline{A}BC}_{BC} + \underbrace{ABC + A\overline{B}C}_{AC} + \underbrace{ABC + AB\overline{C}}_{AB} = BC + AC + AB$$

c. Qui tắc 3

Có thể bỏ số hạng chứa các biến đã có trong số hạng khác.

Ví dụ 1: Rút gọn $Z = AB + \overline{B}C + AC$

Biểu thức không đổi khi ta nhân một số hạng với 1 $(1 = B + \overline{B})$

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

$$= AB(1 + C) + \overline{B}C(1 + A) = AB + \overline{B}C$$

Ví dụ 2: Rút gọn $Z = (A + B)(\overline{B} + C)(A + C)$

Biểu thức không đổi khi ta cộng một số hạng với 0 $(0 = B.\overline{B})$

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

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

$$= \underbrace{(A + B)(A + B + C)}_{(2')}\underbrace{(\overline{B} + C)(A + \overline{B} + C)}_{(2')} = (A + B)(\overline{B} + C)$$

d. Qui tắc 4

Có thể đơn giản bằng cách dùng hàm tổng chuẩn tương đương có số hạng ít nhất.

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

Hàm đảo $\overline{Z} = \overline{f(A,B,C)} = \sum(0,1) = \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C = \overline{A}\overline{B} = \overline{A+B}$

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

3. Dùng bảng Karnaugh

a. Nguyên tắc

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

Xét 2 tổ hợp AB và $A\overline{B}$, hai tổ hợp này chỉ khác nhau một bit gọi là hai tổ hợp kề nhau. Ta có $AB + A\overline{B} = A$, biến B được đơn giản.

Phương pháp 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 thực hiện theo ba bước.

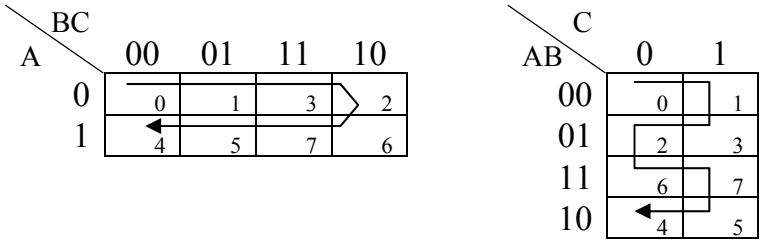
- Thiết lập bảng Karnaugh.
- Chuyển các hàm cần đơn giản vào bảng.
- Nhóm các ô chứa tổ hợp kề nhau sau cho có thể rút gọn hàm tới mức tối giản.

b. Thiết lập bảng Karnaugh

Bảng Karnaugh thực chất là một dạng khác của bảng sự thật, trong đó mỗi ô của bảng tương đương với 1 hàm trong bảng sự thật.

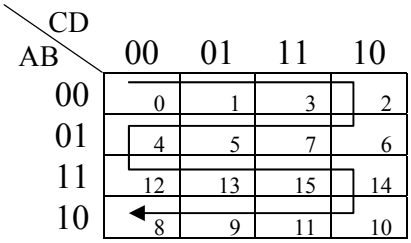
Để thiết lập bảng Karnaugh, người ta chia biến ra làm đôi, phân nữa dùng để tạo $2^{n/2}$ cột, phân nữa còn lại tạo $2^{n/2}$ dòng (nếu n là số lẻ, ta có thể chọn số lượng biến làm cột lớn hơn số lượng biến làm dòng hay ngược lại). Như vậy, nếu hàm có n biến, bảng Karnaugh là bảng có 2^n ô, mỗi ô tương ứng với một tổ hợp của biến. Các ô trong bảng được sắp đặt kề nhau chỉ khác nhau một đơn vị nhị phân (khác nhau 1 bit). Điều này rất thuận tiện khi chúng ta dùng mã Gray. Chính sự sắp đặt này giúp ta đơn giản bằng cách nhóm các ô lại.

Ví dụ: Bảng Karnaugh 3 biến với A ở vị trí MSB và C ở vị trí LSB. Dấu mũi tên tăng theo chiều số thứ tự của mã Gray.



Do các tổ hợp bìa trái và bìa phải kề nhau nên có thể coi bảng dạng hình trụ thẳng đứng. Tương tự, bìa trên và bìa dưới kề nhau nên cũng có thể coi bảng như hình trụ nằm ngang. Bốn tổ hợp biến ở 4 góc là kề nhau.

Bảng Karnaugh cho hàm 4 biến được biểu diễn như sau – chiều theo mũi tên là chiều tăng theo mã Gray:

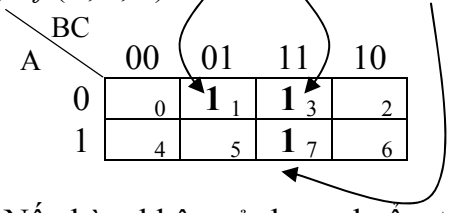


c. Biểu diễn hàm logic trong bảng Karnaugh

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

- Từ hàm viết dưới dạng tổng chuẩn:

Ví dụ: $f(A,B,C) = \overline{A}\overline{B}C + \overline{A}BC + ABC$



- Nếu hàm không ở 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 đầy đủ các biến.

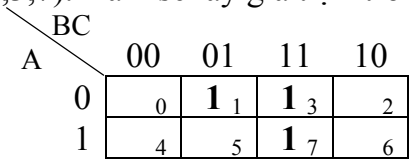
Ví dụ: $f(A,B,C,D) = ABC + AB\overline{D} + A\overline{B}C + A\overline{B}D$, hàm 4 biến ta đưa về dạng tổng chuẩn như sau (loại bỏ các số hạng lặp lại):

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

$$= ABCD + ABC\overline{D} + AB\overline{C}\,\overline{D} + A\overline{B}CD + A\overline{B}C\overline{D} + A\overline{B}\overline{C}D$$

- Từ dạng số Σ (tổng), hàm sẽ có giá trị 1 trong những ô là số tương ứng.

Ví dụ: f(A, B, C) = Σ(1,3,7). Hàm sẽ lấy giá trị 1 trong những ô 1, 3, 7.

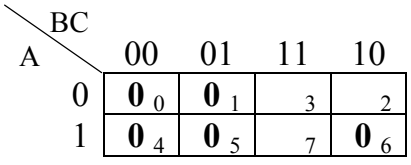


- Từ dạng tích chuẩn, ta lấy hàm đảo để có dạng tổng chuẩn và ghi giá trị 0 vào các ô tương ứng với tổ hợp biến trong tổng chuẩn này.

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

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

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



- Từ dạng số Π (tích), ta đưa 0 vào các ô số trong biểu thức tích, dĩ nhiên các ô khác còn lại ghi 1.

Ví dụ: f(A, B, C) = Π(0,2,3,7). Hàm sẽ lấy giá trị 0 trong những ô 0, 2, 3, 7.

- Từ bảng sự thật ghi 1 vào các ô tương ứng với tổ hợp biến mà hàm cho giá trị riêng là 1.

Ví dụ: Cho bảng sự thật sau:

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

Ta sẽ ghi 1 vào các ô: 1, 2, 3, 5, 7.

- Trường hợp một số tổ hợp cho giá trị hàm không xác định, nghĩa là ứng với tổ hợp này hàm có giá trị 0 hoặc 1 tuỳ ý. Do đó, ta ghi dấu × vào các ô tương ứng trong với tổ hợp này, lúc gom nhóm, ta cho các giá trị × này là 0 hay 1 tuỳ ý sao có kết quả có lợi cho ta (kết quả đơn giản nhất).

d. Qui tắc rút gọn

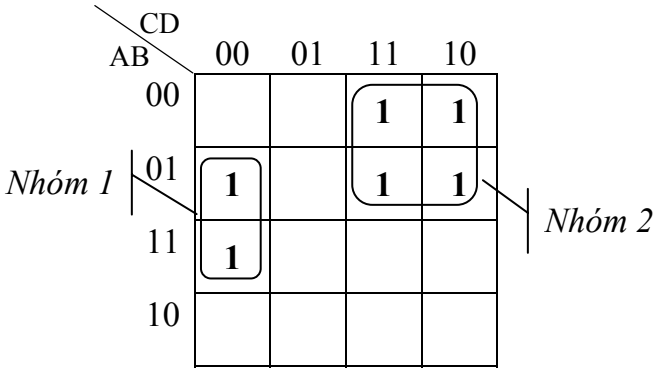
Để rút gọn hàm, ta gom các số 1 kề nhau thành từng nhóm sao cho số nhóm càng ít càng tốt, điều này có nghĩa là số hạng trong kết quả đích càng ít đi.

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

Số 1 trong mỗi nhóm phải là bội của 2^k. Cứ mỗi nhóm 2^k số 1, thì tổ hợp biến tương ứng ta đơn giản được k số hạng.

Kết quả cuối cùng được lấy như sau: Hàm rút gọn là tổng của các tích. Mỗi số hạng của tổng tương ứng với 1 nhóm các số 1 nói trên và số hạng này là tích của các biến, biến 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. Nếu các số 1 đồng thời nằm trong ô A và $\overline{A}$ thì biến A sẽ được đơn giản.

Ví dụ: Ta xem cách chọn nhóm và rút gọn bảng dưới đây.

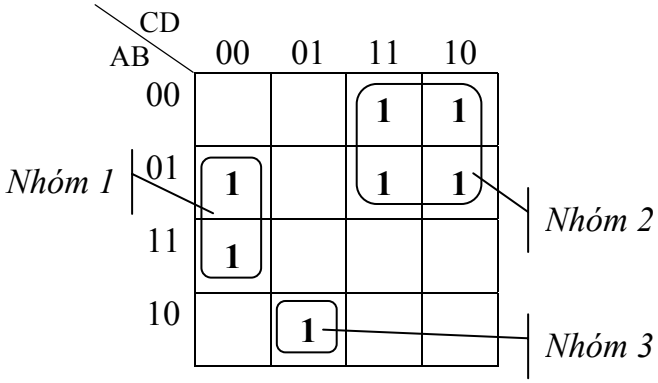


- Nhóm 1 chứa 2 số 1 ($2 = 2^1$, k = 1), như vậy nhóm 1 sẽ còn 3 biến. Theo hàng, 2 số 1 ở 2 ô đó là $\overline{A}B$ (01) và AB (11) nên biến A sẽ được đơn giản, còn lại B. Theo cột thì 2 ô này ứng với tổ hợp $\overline{C}\overline{D}$ (00) ➔ Kết quả nhóm 1 là: $B\overline{C}\overline{D}$.

- Nhóm 2 chứa 4 số 1 ($4 = 2^2$, k = 2), như vậy nhóm 2 sẽ còn 2 biến. Theo hàng, 4 số 1 ở 2 hàng đó là $\overline{A}\overline{B}$ (00) và $\overline{A}B$ (01) nên biến B sẽ được đơn giản, còn lại $\overline{A}$. Theo cột thì 2 cột này ứng với tổ hợp CD (11) và $C\overline{D}$ (10) nên biến D được đơn giản, còn lại C ➔ Kết quả nhóm 2 là: $\overline{A}C$.

Ví dụ 1: Đơn giản hàm:

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

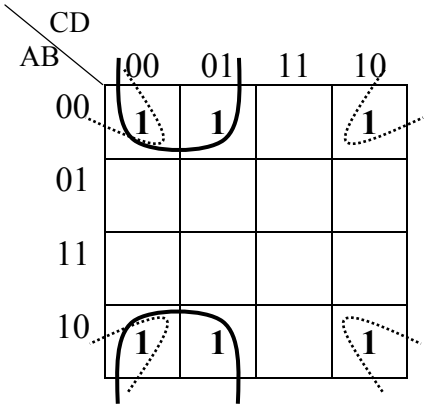


Nhóm 1: $B\overline{C}\overline{D}$. Nhóm 2: $\overline{A}C$. Nhóm 3: $A\overline{B}\overline{C}D$.

Vậy hàm rút gọn $Y = f(A,B,C,D) = B\overline{C}\overline{D} + \overline{A}C + A\overline{B}\overline{C}D$

Ví dụ 2: Đơn giản hàm:

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



—— Nhóm 1 - 4 số 1, gồm 2 số 1 trên và 2 số 1 dưới: $\overline{B}\overline{C}$

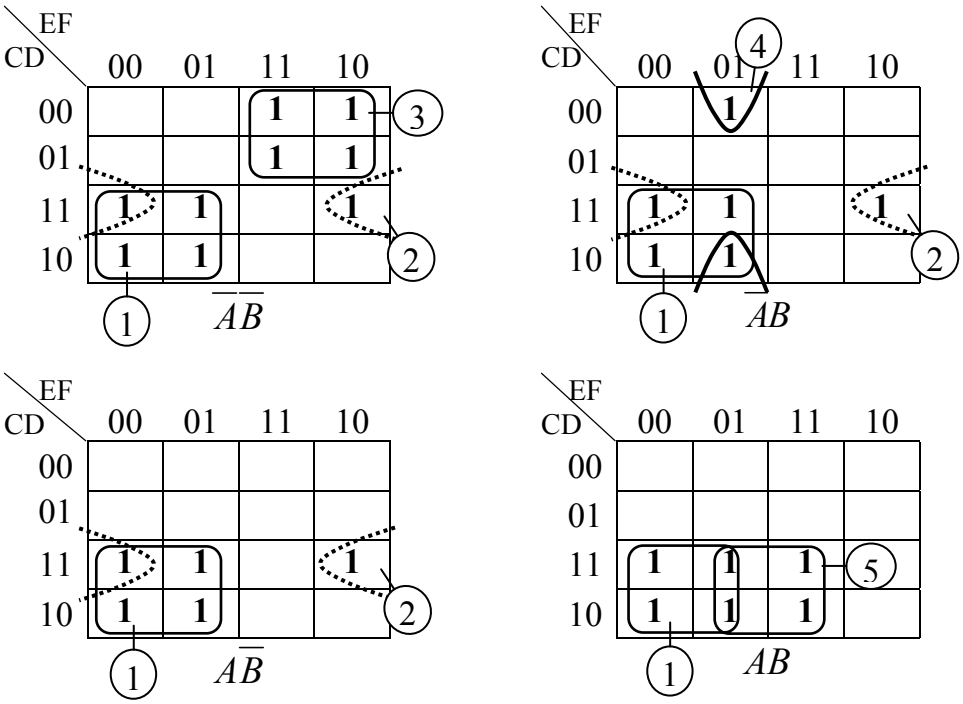
····· Nhóm 2 - 4 số 1, gồm 4 số ở 4 góc: $\overline{B}\overline{D}$

Vậy $Y = \overline{B}\overline{C} + \overline{B}\overline{D}$

Ví dụ 3: Rút gọn hàm 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). Tương tự như trên, nhưng ta phải vẽ 4 bảng ứng với 4 tổ hợp của AB là:

$\overline{A}\overline{B}$ cho các số từ 0 đến 15. $\overline{A}B$ cho các số từ 16 đến 31.

$A\overline{B}$ cho các số từ 32 đến 47. AB cho các số từ 48 đến 63.



(1) $\Leftrightarrow C\overline{E}$; (2) $\Leftrightarrow \overline{A}CD\overline{F} + \overline{B}CD\overline{F}$; (3) $\Leftrightarrow \overline{A}\overline{B}C\overline{E}$; (4) $\Leftrightarrow \overline{A}\overline{B}\overline{D}\overline{E}F$; (5) $\Leftrightarrow ABCF$.

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

4. Phương pháp Quine–Mc. Cluskey

a. Nguyên tắc

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

Phương pháp chia làm hai giai đoạn.

- Giai đoạn 1: Xác định các tích thứ nhất là minterm có được trong quá trình đơn giản nói trên.
- Giai đoạn 2: Tối giản các tích thứ nhất.

b. Ví dụ minh họa

Rút gọn hàm $f(A,B,C,D) = \sum(1,2,4,5,6,10,12,13,14)$

- **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 ví dụ này ta có 3 nhóm.

- Nhóm chứa 1 số 1 gồm: 1, 2, 4 – (0010, 0010, 0100).
- Nhóm chứa 2 số 1 gồm: 5, 6, 10, 12 – (0101, 0110, 1010, 1100).
- Nhóm chứa 3 số 1 gồm: 13, 14 – (1101, 1110).

Thiết lập bảng 1 như sau:

Chọn	Hàng	A	B	C	D
×	1	0	0	0	1
×	2	0	0	1	0
×	4	0	1	0	0
×	5	0	1	0	1
×	6	0	1	1	0
×	10	1	0	1	0
×	12	1	1	0	0
×	13	1	1	0	1
×	14	1	1	1	0

Mỗi tổ hợp trong nhóm sẽ được so sánh với tổ hợp trong nhóm kế cận. Nếu 2 tổ hợp chỉ khác nhau 1 biến, ta dùng biểu thức $AB + \overline{A}B = B$ để đơn giản 1 biến. Biến đã được đơn giản sẽ được thay bởi dấu – (gạch ngang). Đánh × vào 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 2: 0101, chúng chỉ khác nhau 1 biến B, vậy ta có thể đơn giản thành 0–01. Hai số hạng 1 và 5 đã được gom lại.

Thiết lập bảng 2 như sau:

Chọn	Hàng	A	B	C	D
	1,5	0	–	0	1
×	2,6	0	–	1	0
×	2,10	–	0	1	0
×	4,5	0	1	0	–
×	4,6	0	1	–	0
×	4,12	–	1	0	0
×	5,13	–	1	0	1
×	6,14	–	1	1	0
×	10,14	1	–	1	0
×	12,13	1	1	0	–
×	12,14	1	1	–	0

Tiếp tục thực hiện công việc tương tự như trên với 2 nhóm trong bảng thứ 2 này, các số hạng sẽ được gom lại nếu chúng chỉ khác nhau 1 biến và có cùng vị trí dấu – (dấu gạch trùng nhau).

Thiết lập bảng 3 như sau:

Chọn	Hàng	A	B	C	D
	2,6; 10,14	–	–	1	0
	2,10; 6,14	–	–	1	0
	4,5; 12,13	–	0	1	–
	4,6; 12,14	–	1	–	0
	4,12; 5,13	–	1	0	–
	4,12; 6,14	–	1	–	0

Quan sát bảng 3 ta thấy không thể rút gọn được nữa, đồng thời có các tổ hợp giống nhau, ta 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 trước và các tổ hợp trong bảng cuối.Ta có các tổ hợp sau:

- (1,5) ⇔ $\overline{A}CD$ ở bảng 2.
- (2,6; 10,14) = (2,10; 6,14) ⇔ $C\overline{D}$ ở bảng cuối.
- (4,5; 12,13) = (4,12; 5,13) ⇔ $B\overline{C}$ ở bảng cuối.
- (4,6; 12,14) = (4,12; 6,14) ⇔ $B\overline{D}$ ở bảng cuối.

Vậy kết thúc bước 1 ta được: $f(A,B,C,D) = \overline{A}CD + C\overline{D} + B\overline{C} + B\overline{D}$

Đến đây, quan sát các tổ hợp cho kết quả trên, ta thấy các tổ hợp còn chứa các số hạng giống nhau (số 4,12). Như vậy kết quả có thể chưa tối giản. Ta tiếp tục chuyển sang bước 2.

- **Giai đoạn 2**: Để rút gọn hơn nữa ta lập một bảng với cách thiết lập như sau:

- Cột bên trái ghi các tổ hợp đã được chọn trong giai đoạn 1, các cột còn lại ghi giá trị thập phân trong hàm ban đầu.
- Trên cùng hàng của tổ hợp ta đánh dấu * với các ô tương ứng của cột. Ví dụ hàng chứa tổ hợp (1,5) ta đánh dấu * vào ô tương ứng cột 1 và 5, trên dòng (1,5). Tương tự cho các tổ hợp khác. Sau đó, ta sẽ lần lượt dò các dòng từ trên xuống, đánh × vào dòng cuối tương ứng với dấu * trên các dòng này, đến khi nào tất cả các ô của dòng cuối đều được đánh dấu × thì ta ngưng, lúc đó tổ hợp các dòng được chọn là kết quả hàm. Như trên, ta được bảng sau:

Tổ hợp	1	2	4	5	6	10	12	13	14
1,5 ←	*↓			*↓					
2,6; 10,14 ←		*↓			*↓	*↓			*↓
4,5; 12,13 ←			*↓	*			*↓	*↓	
4,6; 12,14			*		*		*		*
Chọn đủ →	×	×	×	×	×	×	×	×	×

Kết quả ta được: $f(A,B,C,D) = \overline{A}CD + C\overline{D} + B\overline{C}$. Ta đã loại được giá trị $B\overline{D}$.

CHƯƠNG 3: CỔNG LOGIC

- ✓ CÁC KHÁI NIỆM LIÊN QUAN
- ✓ CÁC CỔNG LOGIC CƠ BẢN
- ✓ CÁC 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

I. KHÁI NIỆM LIÊN QUAN

1. Giới thiệu

Cổng logic là tên chung của các mạch điện tử 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 các công nghệ tích hợp IC (Intergrated circuit).

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

2. Tính hiệu tương tự và tính hiệu số

Tính 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.

Tín hiệu số là tín hiệu có dạng xung, gián đoạn về thời gian và biên độ, chỉ có 2 mức rõ rệt là mức cao và mức thấp. Tín hiệu số chỉ được phát sinh bởi các mạch điện tích hợp.

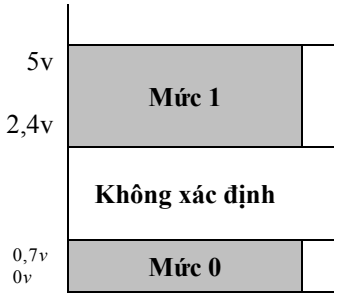
3. Mạch tương tự và mạch số

Mạch điện tử xử lý các tín hiệu tương tự gọi là mạch tương tự, xử lý các tín hiệu số gọi là mạch số. Một cách tổng quát, mạch số có nhiều ưu điểm hơn mạch tương tự:

- Ít bị ảnh hưởng của nhiễu.
- Dễ chế tạo thành mạch tích hợp.
- Dễ thiết kế và phân tích. Việc phân tích thiết kế dựa trên tính năng của IC và khối mạch chứ không dựa trên từng linh kiện rời.
- Thuận tiện trong điều khiển tự động, tính toán, lưu trữ dữ liệu và liên kết với máy tính.

4. Biểu diễn trạng thái logic 0 và 1

Trong hệ thống mạch logic, các trạng thái logic được biểu diễn bởi các mức điện thế. Với qui ước logic dương là điện thế cao (mức logic 1), điện thế thấp là biểu diễn mức logic thấp (lgoic 0). Việc qui ước này có thể được đặt ngược lại. Trong thực tế, mức logic 1 và logic 0 tương ứng với một khoảng điện thế xác định và có một khoảng chuyển tiếp giữa mức cao và mức thấp, ta gọi là khoảng không xác định.

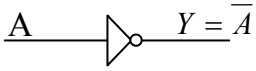


II. CÁC CỔNG LOGIC CƠ BẢN

1. Cổng NOT

Còn gọi là cổng đảo (Inventer), dùng để thực hiện hàm: $Y = \overline{A}$

Ký hiệu mũi tên là chiều dòng điện, vòng trong là ký hiệu đảo. Trong những trường hợp không gây nhầm lẫn, ta bỏ dấu mũi tên.



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

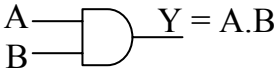
Bảng sự thật

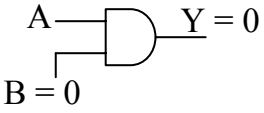
2. Cổng AND

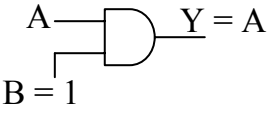
Dùng thực hiện hàm AND của 2 hay nhiều biến.

Cổng AND có số ngã vào tuỳ thuộc vào số biến và có một ngã ra. Ngã ra cổng là hàm AND của các biến ngã vào.

Dưới đây là ký hiệu và bảng sự thật của cổng AND 2 ngã vào.







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

Hoặc

A	B	Y=A.B
×	0	0
0	1	0
1	1	1

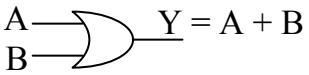
Đọc giả tự cho nhận xét về cổng AND.

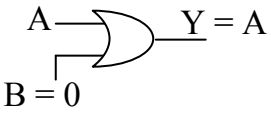
3. Cổng OR

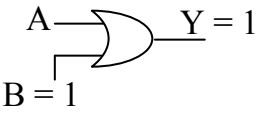
Dùng thực hiện hàm OR của 2 hay nhiều biến.

Cổng OR có số ngã vào tuỳ thuộc vào số biến và có một ngã ra. Ngã ra cổng là hàm OR của các biến ngã vào.

Dưới đây là ký hiệu và bảng sự thật của cổng OR 2 ngã vào.







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

Hoặc

A	B	Y=A + B
×	1	1
0	0	0
1	0	1

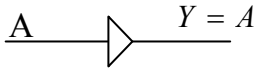
Đọc giả tự cho nhận xét về cổng OR.

4. Cổng BUFFER

Còn gọi là cổng đệm, có 1 ngã vào và 1 ngã ra. Tính hiệu số qua cổng BUFFER không đổi trạng thái logic. Cổng BUFFER dùng trong các mục đích sau:

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

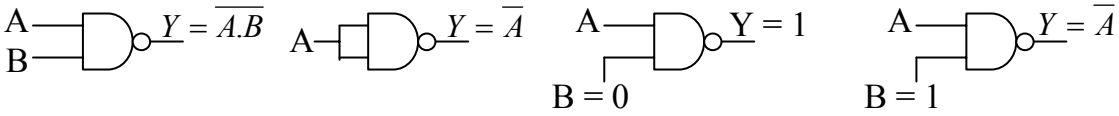
Ký hiệu cổng BUFFER như sau:



5. Cổng NAND

Là kết hợp giữa cổng AND và cổng NOT, thực hiện hàm $Y = \overline{A.B}$ (đây là trường hợp 2 ngã vào, trường hợp nhiều ngã vào, đọc giả tự suy ra).

Dưới đây là ký hiệu và bảng sự thật của cổng NAND 2 ngã vào.

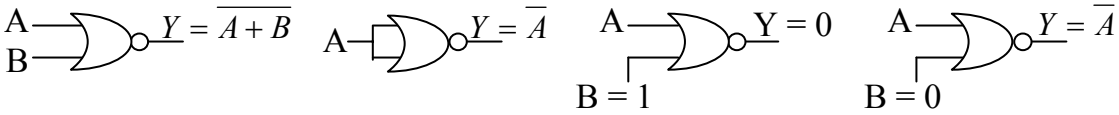


A	B	$Y = \overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

6. Cổng NOR

Là kết hợp giữa cổng OR và cổng NOT, thực hiện hàm $Y = \overline{A + B}$ (đây là trường hợp 2 ngã vào, trường hợp nhiều ngã vào, đọc giả tự suy ra).

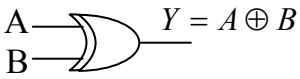
Dưới đây là ký hiệu và bảng sự thật của cổng NOR 2 ngã vào.



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

7. Cổng EX-OR

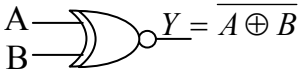
Dùng để thực hiện hàm EX-OR. $Y = A \oplus B = \overline{A}B + A\overline{B}$.
Cổng EX-OR 2 ngã vào và 1 ngã ra.



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

8. Cổng EX-NOR

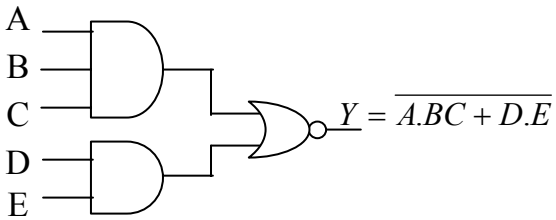
Dùng để thực hiện hàm EX-NOR. $Y = \overline{A \oplus B}$.
Cổng EX-NOR 2 ngã vào và 1 ngã ra.



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

9. Cổng phức AOI (And – Or – Inverter)

Ứng dụng các kết quả của Đại số Boole, người ta có thể nối nhiều cổng khác nhau trên 1 chip IC để thực hiện một hàm logic phức tạp nào đó. Cổng AOI là một loại cổng kết hợp 3 loại cổng AND, OR và NOT. Ví dụ, để thực hiện hàm logic $Y = \overline{A.B.C + D.E}$, ta có cổng phức sau:

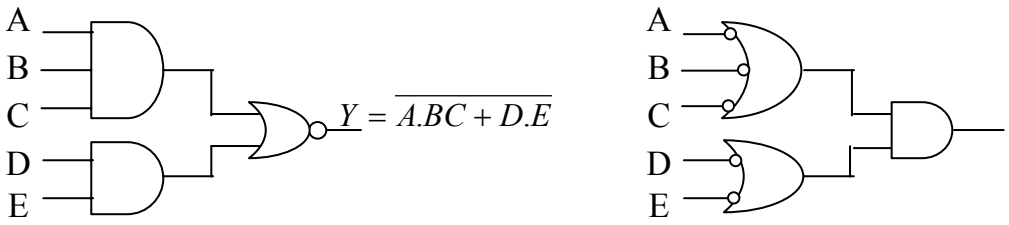


10. Biến đổi qua lại giữa các cổng logic

Trong chương Hàm Logic chúng ta đã thấy tất cả các hàm logic có thể được thay thế bởi 2 hàm logic là AND (hoặc OR) và NOT. Các cổng logic có chức năng thực hiện hàm logic, ta chỉ cần dùng hàm AND (hoặc OR) và NOT để thực hiện các hàm logic này. Tuy nhiên, vì cổng NOT cũng có thể thực hiện bằng cổng NAND (hoặc NOR). Như vậy tất cả các hàm logic đều có thể được thực hiện bởi 1 cổng duy nhất đó là cổng NAND (hoặc NOR). Hàm ý này, cũng 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 từ ngã vào và ngã ra khi biến đổi từ AND sang OR và ngược lại, nếu ở các ngã này đã có cổng đảo rồi thì cổng đảo này sẽ biến mất.

Ví dụ: Hai mạch dưới đây là tương đương nhau.



III. THÔNG SỐ KỸ THUẬT CỦA IC SỐ

1. Các đại lượng điện đặc trưng

Để sử dụng tốt IC, ta nên biết các thuật ngữ, đặt tính của IC, sơ đồ chân của IC.

- V_{cc}: Điện thế nguồn (power supply). Đây là khoảng điện thế cấp cho IC để nó hoạt động tốt.
- V_{IH}: Điện thế ngã vào mức cao (high level input voltage). Đây là điện thế ngã vào nhỏ nhất được xem là ở mức 1.
- V_{IL}: Điện thế ngã vào mức thấp (low level input voltage). Đây là điện thế ngã vào lớn nhất được xem là ở mức 0.
- V_{OH}: Điện thế ngã ra mức cao (high level output voltage). Đây là điện thế ngã ra nhỏ nhất được xem là ở mức cao.
- V_{OL}: Điện thế ngã ra mức thấp (low level output voltage). Đây là điện thế ngã ra lớn nhất được xem là ở mức thấp.
- I_{IH}: Dòng điện ngã vào mức cao (high level input current). Đây là dòng điện lớn nhất vào ngã vào của IC ở mức cao.
- I_{IL}: Dòng điện ngã vào mức thấp (low level input current). Đây là dòng điện ra khỏi IC khi ở mức thấp.
- I_{OH}: Dòng điện ngã ra mức cao (high level output current). Đây là 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). Đây là dòng điện lớn nhất ở 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, IC còn một số thuật ngữ khác, chúng ta sẽ đề cập khi nói về tính chất của IC.

2. Công suất tiêu tán (power requirement)

Mỗi khi IC hoạt động, sẽ tiêu thụ một công suất từ nguồn cung cấp V_{CC} (hoặc V_{DD}). Công suất tiêu tán này xác định bởi hiệu điện thế nguồn và dòng điện qua IC. Do khi hoạt động, dòng điện trong IC thường thay đổi ở mức cao và mức thấp, nên công suất được tính từ dòng điện trung bình qua IC.

$P_{D(avg)} = I_{CC(avg)}.V_{CC}$

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

3. Fan-Out

Một cách tổng quát, ngã ra của một mạch logic đòi hỏi phải thúc mạch vào của một số mạch logic khác. Fan-Out là số ngã vào lớn nhất có thể nối với ngã ra của một IC cùng loại mà vẫn đảm bảo cho mạch hoạt động bình thường. Ta có 2 loại Fan-Out ứng với 2 trạng thái logic ngã ra.

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

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

Thường 2 giá trị Fan-Out này khác nhau. Để an toàn, ta dùng giá trị nhỏ trong 2 giá trị này. Fan-Out tính theo đơn vị Unit Load (UL - tải đơn vị).

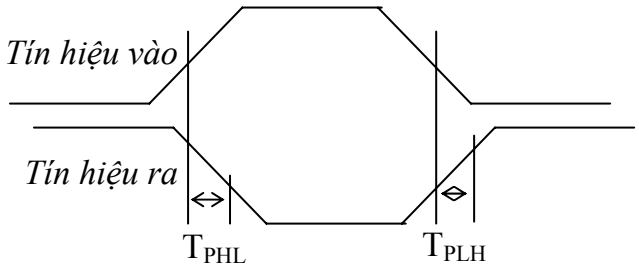
4. Thời trễ truyền (propagation delay)

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

Có 2 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.

Tuỳ theo họ IC, thời trễ truyền có thể từ vài *ns* đến vài trăm *ns*. Thời trễ truyền càng lớn thì tốc độ IC càng nhỏ.

Ví dụ: Tín hiệu qua cổng đảo, thời trễ truyền xác định như sau:



5. Tích số công suất – vận tốc (speed – power product)

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

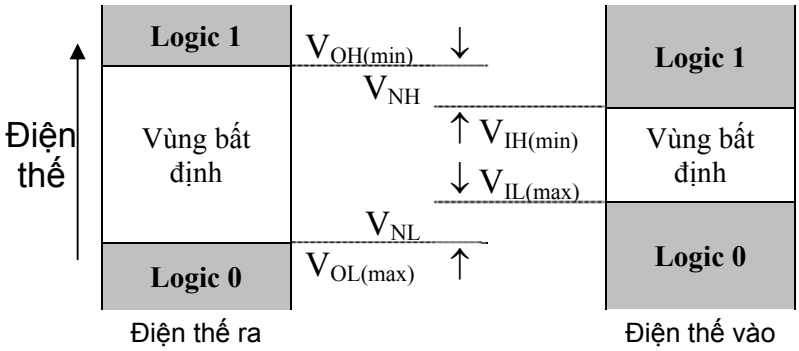
$10ns \times 50mW = 10.\ 10^{-9} \times 50.\ 10^{-3} = 500.\ 10^{-12}$ walt-sec = 500 picojoules (pj).

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

6. Tính miễn nhiễu (noise immunity)

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

Tín miễn nhiễu của một mạch logic tuỳ thuộc vào khả năng dung nạp hiệu thế nhiễu của mạch và xác định bởi lề nhiễu, cho bởi:



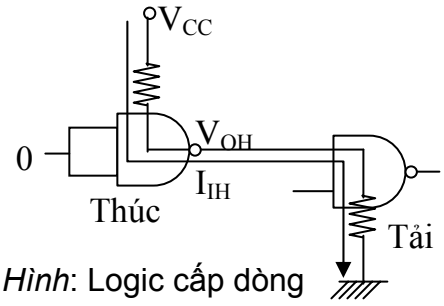
Tín hiệu khi vào mạch logic được xem là mức 1 khi có trị lớn hơn $V_{IH(min)}$, được xem là mức 0 khi có trị nhỏ hơn $V_{IL(max)}$. Điện thế trong khoảng giữa không ứng với một mức logic nào, nên gọi là vùng bất định. Do có sự khác biệt giữa $V_{OH(min)}$ với $V_{IH(min)}$, $V_{OL(max)}$ với $V_{IL(max)}$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

- Lề nhiễu mức cao: $V_{NH} = V_{OH(min)} - V_{IH(min)}$
- Lề nhiễu mức thấp: $V_{NL} = V_{OL(max)} - V_{IL(max)}$

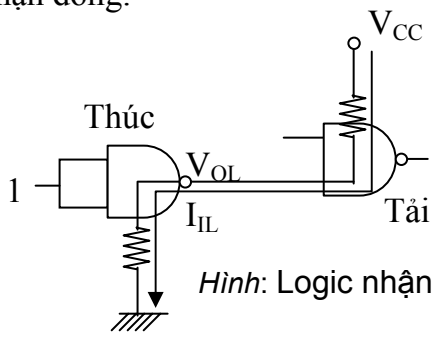
Khi tín hiệu ra ở mức cao đưa vào ngã vào, bất cứ tín hiệu nào có giá trị âm và biên độ lớn hơn V_{NH} đều làm cho điện thế ngã vào rơi vào vùng bất định và mạch không biết tín hiệu ở mức logic nào. Tương tự cho trường hợp ngã ra ở mức thấp, tín hiệu nhiễu có trị dương biên độ > V_{NL} sẽ đưa mạch vào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

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, tầng nhận tín hiệu gọi là tầng tải. Sự trao đổi dòng điện giữa hai tầng thúc và tầng tải thể hiện bởi logic cấp dòng và logic nhận dòng.



Hình: Logic cấp dòng



Hình: Logic nhận dòng

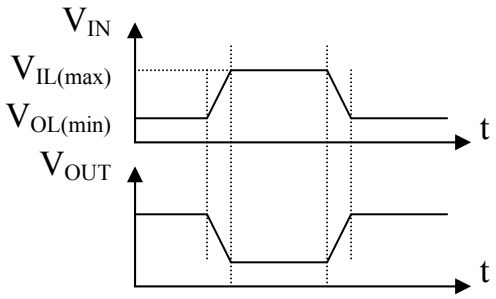
Logic cấp dòng: Khi ngã ra của cổng NAND thứ nhất ở mức cao nó cấp dòng I_{IH} cho ngã vào của cổng NAND thứ hai. Ngã ra của cổng một như nguồn cấp cho ngã vào cổng hai.

Logic nhận dòng: Khi ngã ra của cổng NAND thứ nhất ở mức thấp nó nhận dòng I_{IL} từ ngã vào của cổng NAND thứ hai.

Thường dòng nhận của tầng thúc khi ở mức thấp có giá trị khá lớn so với dòng cấp của nó khi ở mức cao. Người ta hay dùng trạng thái này để gánh những tải tương đối nhỏ, ví dụ như một đèn led.

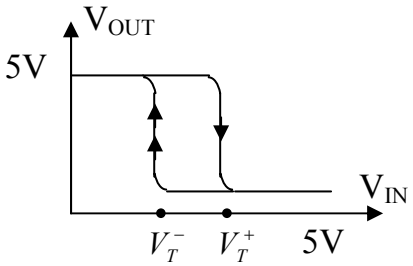
8. Tính Schmitt Trigger

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 tác dụng làm giảm độ rộng sườn xung của tín hiệu qua mạch. Tuy nhiên vẫn còn một khoảng sườn xung nằm trong vùng chuyển tiếp nên tín hiệu ra không vuông hoàn toàn. Hình dưới đây minh họa tính chất đó.



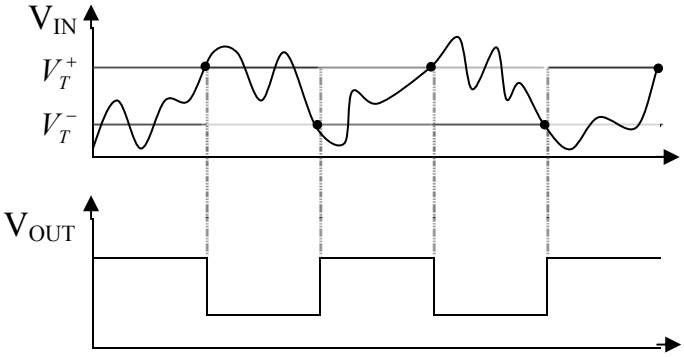
Hình: Tín hiệu vào/ra của cổng logic (không vuông).

Để cải thiện hơn nữa tín hiệu ngã và, bảo đảm tính miễn nhiễu cao, người ta chế ra các cổng có tính điện trở thế, được gọi là cổng Schmitt Trigger. Hình dưới đây minh họa tín hiệu điện vào và sự thay đổi logic của cổng Schmitt Trigger.

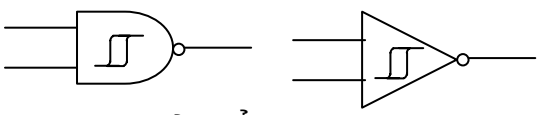


Hình: Tín hiệu vào/ra của cổng Schmitt Trigger.

Hình dưới đây, mô tả V_{IN} và V_{OUT} của cổng Schmitt Trigger. Nếu ngã vào đang là điện thế thấp thì chỉ khi nào điện thế ngã vào vượt qua V_T^+ ngã ra mới đổi trạng thái. Ngược lại, nếu điện thế ngã vào đang ở mức cao thì chỉ khi nào điện thế ngã vào nhỏ hơn V_T^- ngã ra mới đổi trạng thái.



Hình: *Mô* tả V_{IN} và V_{OUT} của cổng Schmitt Trigger.



Hình: Ký hiệu của cổng Schmitt Trigger.

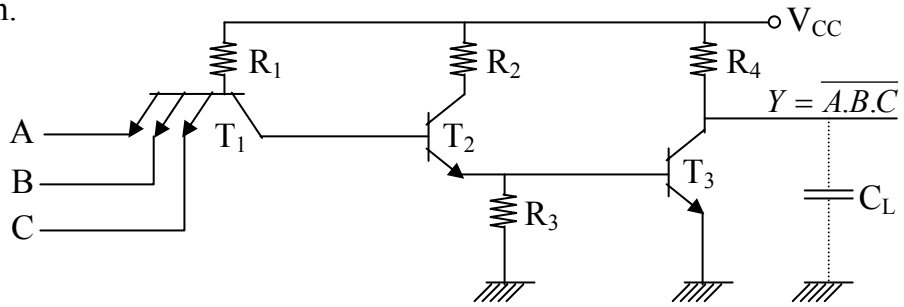
IV.HỌ TTL

1. Giới thiệu

Trong quá trình phát triển của công nghệ chế tạo mạch số, ta có họ: RTL (Resistor – transistor logic), DCTL (Direct couple – transistor logic), RCTL (Resistor-capacitor transistor logic), DTL (Diod – transistor logic), ECL (Emitter – couple transistor logic),… Hiện nay, tồn tại nhiều họ có tính năng kỹ thuật cao như: Thời trễ truyền nhỏ, tiêu hao công suất ít. Ta sẽ xét một vài họ có tính năng kỹ thuật này, đầu tiên ta xét họ TTL (Transistor – transistor logic).

2. Cổng cơ bản họ TTL

Ta lấy cổng NAND 3 ngã vào làm ví dụ để thấy cấu tạo và vận hành của một cổng cơ bản.



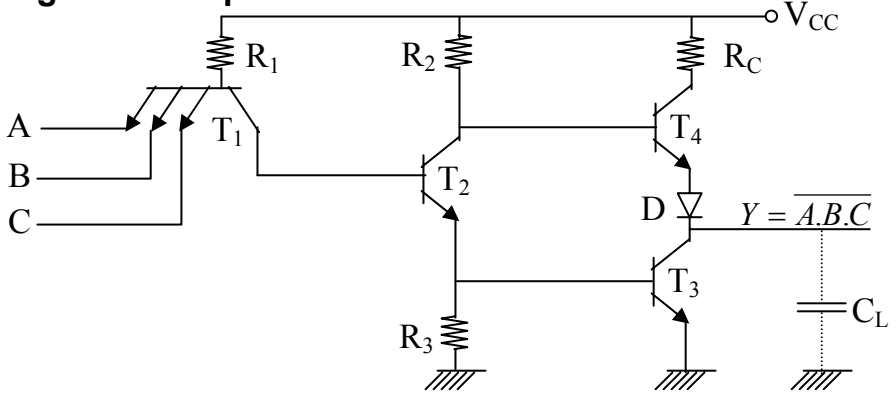
Hình: Mô phỏng cổng NAND ba ngã vào họ TTL.

Khi một trong các cổng A, B, C xuống mức 0; T_1 dẫn, đưa đến T_2 ngưng, T_3 ngưng, ngã ra Y lên cao. Khi cả 3 ngã vào A, B, C lên cao; T_1 ngưng, đưa đến T_2 dẫn, T_3 dẫn, ngã ra Y xuống thấp. Đó chính là kết quả của cổng NAND 3 ngã vào.

Tụ C_L trong mạch chính là tụ ký sinh ngã ra của mạch kết hợp với ngã vào của tầng tải, khi mạch hoạt động tụ sẽ nạp điện qua R_4 (lúc T_3 ngưng) và phóng điện qua transistor T_3 khi nó bắt đầu dẫn điện, 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 lớn, muốn giảm công suất phải tăng R_4 nhưng như vậy, thời trễ truyền sẽ lớn hơn (tỉ lệ nghịch giữa thời trễ truyền và công suất). Để giải quyết vấn đề này và để thoả mãn một số yêu cầu khác, người ta đã chế tạo các cổng logic với các ngã khác nhau. Ta sẽ xét sau đây.

3. Cổng cơ bản họ TTL

a. Ngã ra totempole

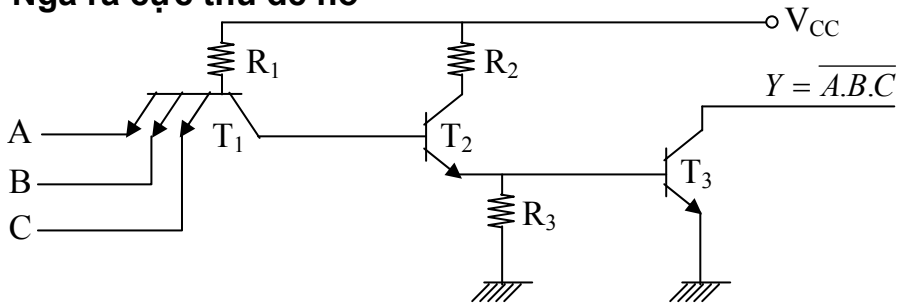


Hình: Ngã ra totempole.

Ta thấy, R_4 trong mạch được thay thế bởi cụm T_4, R_C và Diod D, trong đó R_C có giá trị rất nhỏ, không đáng kể. Tương tự như mạch trên, khi cả 3 ngã vào A, B, C lên cao; T_1 ngưng, đưa đến T_2 dẫn, T_3 dẫn, T_4 ngưng, ngã ra Y xuống thấp. Khi một trong các cổng A, B, C xuống mức 0; T_1 dẫn, đưa đến T_2 ngưng, T_3 ngưng, T_4 dẫn ngã ra Y lên cao. Tụ C_L nạp điện khi T_4 dẫn và phóng điện qua T_3 khi T_3 dẫn, thời hằng mạch rất nhỏ nên thời trễ truyền nhỏ. Ngoài ra, T_3 và T_4 luân phiên ngưng tương ứng với 2 trạng thái của ngã ra nên công suất giảm đáng kể. Diod D có tác dụng nâng điện thế cực B của T_4 lên, đảm bảo 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 các trạng thái logic của các cổng khác nhau này.

b. Ngã ra cực thu để hở

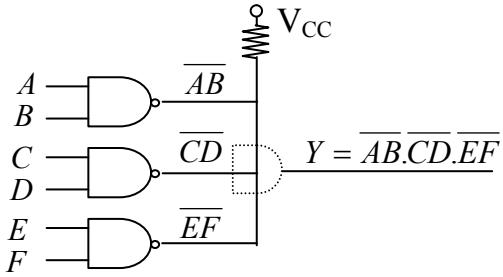


Hình: Ngã ra cực thu để hở.

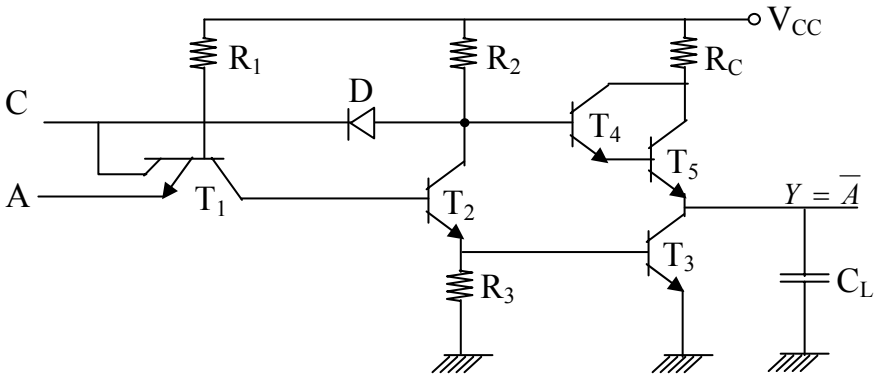
Ngã ra cực thu để hở có một số lợi điểm sau:

- Cho phép kết nối các ngõ ra của nhiều cổng khác nhau, nhưng khi sử dụng phải mắc một điện trở từ ngã ra lên nguồn V_{CC}, gọi là điện trở kéo lên, trị số của điện trở có thể được chọn lớn hay nhỏ tuỳ 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 gọi điểm AND.
- Người ta cũng chế tạo các IC ngã ra cực thu để hở, cho phép điện trở kéo lên mắc vào nguồn điện thế cao, dùng cho các tải đặc biệt hoặc hoặc dùng tạo sự giao tiếp giữa họ TTL với CMOS dùng nguồn cao.

Ví dụ: IC 7406 là loại cổng đảo có ngã ra cực thu để hở có thể mắc lên nguồn 24V.



c. Ngã ra ba trạng thái

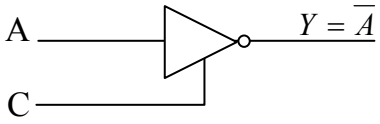


Hình: Ngã ra cổng đảo ba trạng thái.

Mạch trên là một cổng đảo có ngã ra 3 trạng thái, trong đó T_4 và T_5 được mắc để cấp dòng cho tải. Diod D nối vào ngã vào C để điều khiển. Hoạt động của mạch được 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.

Dưới đây là ký hiệu cổng đảo ba trạng thái, có ngã điều khiển C tác động mức cao và bảng sự thật của nó.

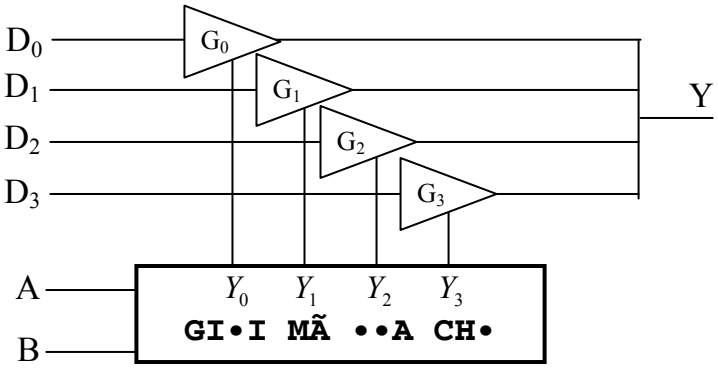


C	A	Y
1	0	1
1	1	0
0	×	Z cao

Bảng sự thật

Các cổng đảo và các cổng đệm ba trạng thái, với ngã điều khiển C tác động ở mức thấp, đọc giả tự vẽ ký hiệu và bảng sự thật.

Một số ứng dụng của cổng đệm ba trạng thái, như để chọn dữ liệu mô tả ở hình dưới đây.



Hoạt động: Ứng với một giá trị nhị phân của địa chỉ AB, một ngã ra được tác động, cho phép một cổng mở và dữ liệu tương ứng được truyền qua. Ví dụ AB = 00, $Y_0 = 1$ ($Y_1 = Y_2 = Y_3 = 0$), G_0 mở, dữ liệu D_0 truyền qua G_0, lúc này G_1,G_2,G_3 đóng và có trạng thái Z cao nên không ảnh hưởng đến hoạt động của mạch.

4. Đặc tính các loạt TTL

Các IC số họ TTL được sản xuất đầu tiên vào năm 1964 bởi hãng Texas Instructment Corporation của Mỹ, lấy số hiệu là 74×××× và 54××××. Sự khác nhau của hai họ này như sau:

74××××: $V_{CC} = 5 \pm 0.5V$ và nhiệt độ hoạt động từ 0^0C đến 70^0C.

54××××: $V_{CC} = 5 \pm 0.25V$ và nhiệt độ hoạt động từ -55^0C đến 125^0C.

Các đặc tính khác hoàn toàn giống nhau nên chúng có cùng số. Một số tính chất của các loạt trên được thể hiện trong bảng sau:

Tham 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
Tham số điện thế	**74**	**74L**	**74H**	**74S**	**74LS**	**74AS**	**74ALS**	**74F**
$V_{OH\ (min)}$	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

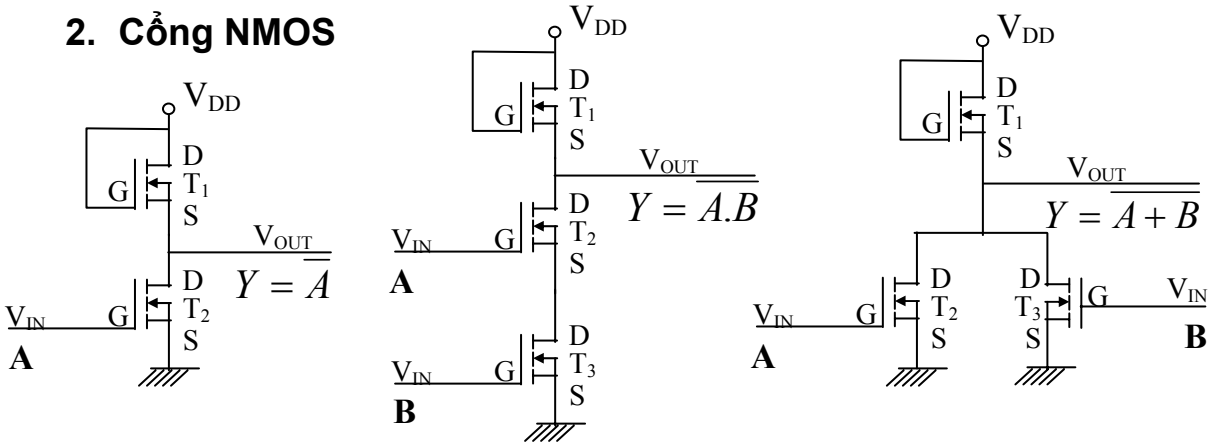
Hình: Bảng một số tính năng kỹ thuật.

V. HỌ MOS

1. Giới thiệu

Họ MOS gồm các IC số dùng công nghệ chế tạo của transitor MOSFET loại tăng, kênh N và kênh P. Với loại N ta có NMOS, loại P ta có PMOS, nếu dùng cả P và N ta có CMOS. Tính năng kỹ thuật của NMOS và PMOS là giống nhau trừ nguồn cấp điện có chiều ngược với nhau, do đó, ta chỉ xét loại NMOS và CMOS.

2. Cổng NMOS



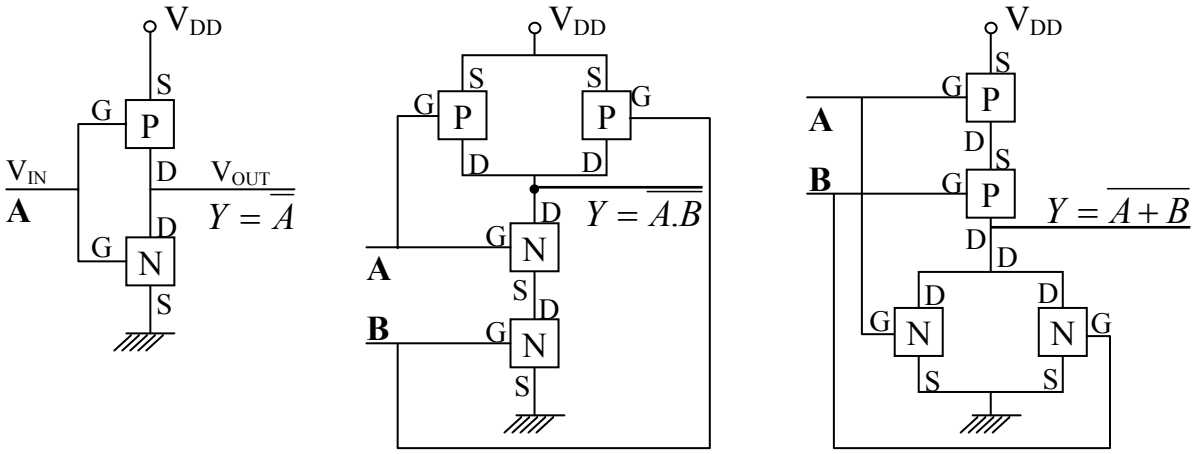
Hình: Cổng NOT, cổng NAND và cổng NOR dùng NMOS.

Bảng dưới đây cho thấy điện thế vào và ra của cổng NOT.

V_{IN}	T_1	T_2	V_{OUT}
0V (logic 0)	$R_{ON} = 100\ K\Omega$	$R_{OFF} = 10^{10}\ \Omega$	+5V (logic 1)
+5V (logic 1)	$R_{ON} = 100\ K\Omega$	$R_{ON} = 1K\Omega$	0.05V (logic 0)

3. Họ CMOS

Họ CMOS sử dụng 2 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.



Hình: Cổng NOT, cổng NAND và cổng NOR dùng CMOS.

Bảng dưới đây cho thấy điện thế vào và ra của cổng NOT.

V_{IN}	T_P	T_N	V_{OUT}
V_{DD} (logic 1)	$R_{OFF} = 10^{10}\ \Omega$	$R_{ON} = 1\ K\Omega$	0V (logic 0)
0V (logic 0)	$R_{ON} = 1\ K\Omega$	$R_{OFF} = 10^{10}\ \Omega$	V_{DD} (logic 0)

4. Đặc tính của họ MOS

Một số tính chất chung của họ MOS 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 ..... $V_{NH} = 30\%V_{DD}$ $V_{NL} = 30\%V_{DD}$
- Fan-Out ..... 50 UL

Do tổng trở của transistor MOS rất lớn nên số Fan-Out của họ MOS rất lớn. Tuy nhiên khi dùng ở tần số cao, người ta giới hạn ở số 50. Nghĩa là một cổng MOS có thể cấp dòng cho 50 cổng tải cùng loạt.

5. Các loạt CMOS

CMOS có 2 ký hiệu: 4××× do hãng RCA chế tạo và 14××× do hãng MOTOROLA chế tạo, có hai loạt 4×××A (14×××A), 4×××B (14×××B), loạt B ra đời có cải thiện dòng ra.

Ngoài ra, còn có các loạt 74C (CMOS có cùng sơ đồ chân với TTL nếu có cùng số), 74HC (High speed CMOS), 74HCT (Hoàn toàn tương thích với TTL kể cả các mức logic), 74AC và 74ACT (Advance CMOS) cải tiến của 74HC và 74HCT về mặt nhiễu.

VI.GIAO TIẾP GIỮA CÁC HỌ IC SỐ

1. Giới thiệu

Giao tiếp là thực hiện 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 2 họ IC TLL và CMOS nên việc giao tiếp trong nhiều trường hợp không thể nối trực tiếp được mà phải nhờ một mạch trung gian nối giữa tầng thúc và tầng tải sau cho tín hiệu ra của tầng thúc phù hợp với tín hiệu vào của tầng tải và dòng điện tầng thúc phải đủ thúc cho tầng tải.

Dưới đây là 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 2 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 tham số của 2 họ IC.

Tham số	CMOS (V_{DD} = 5V)			TTL			
	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
$I_{IH(max)}$	1 µA	1 µA	1 µA	40 µA	20 µA	200 µA	20 µA
$I_{IL(max)}$	1 µA	1 µA	1 µA	1.6 mA	0.4 mA	2 mA	100 µA
$I_{OH(max)}$	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

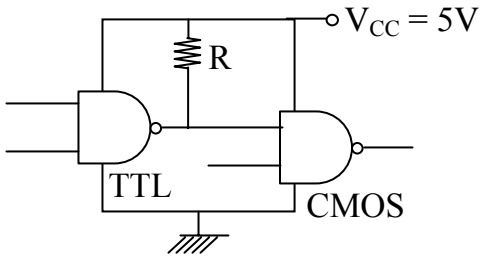
Hình: Bảng một số tính năng kỹ thuật của CMOS và TTL.

2. Dùng TTL thúc CMOS

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

Từ bảng tính năng kỹ thuật trên, ta thấy dòng điện của CMOS có trị rất nhỏ so với dòng của các TTL, vậy dòng điện không có vấn đề.

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



Hình: Điện trở kéo lên.

- **TTL thúc 74HCT.**

Như đã nói trên đây, loạt 75HCT 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 của CMOS, người ta phải dùng một cổng đệm có ngã ra để hở có thể dùng nguồn cao (IC 7407 chẳn hạn) để thực hiện giao tiếp.

3. Dùng CMOS thúc TTL

- **CMOS thúc TTL ở trạng thái cao.**

Từ bảng tính năng kỹ thuật, ta thấy dòng điện ra mức cao của CMOS đủ cấp cho TTL, vậy dòng điện không có vấn đề.

- **CMOS thúc TTL ở trạng thái thấp.**

Dòng điện vào ở trạng thái thấp của TLL thay đổi trong khoảng từ 100μA đến 2*mA*. Vậy hai loạt này có thể giao tiếp với IC TTL mà không có vấn đề gì. Tuy nhiên, với loạt 4000B, I_{OL} rất nhỏ không đủ 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.

- **CMOS dùng nguồn cao thúc TTL.**

Có một số IC loạt 74LS, được chế tạo đặc biệt, có thể nhận điện thế 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 giao tiếp với CMOS dùng nguồn cao, người ta phải dùng cổng đệm hạ điện thế ra thấp xuống cho phù hợp với IC TTL.

CHƯƠNG 4: MẠCH TỔ HỢP

- ✓ **MẠCH MÃ HOÁ**
  - **Mạch mã hoá từ 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**
  - **Mạch đa hợp**
  - **Ứng dụng của mạch đa hợp**
  - **Mạch giải đa hợp**
- ✓ **MẠCH SO SÁNH**
  - **Mạch so sánh 2 số 1 bit**
  - **Mạch so sánh 2 số nhiều bit**
- ✓ **MẠCH KIỂM PHÁT CHẴN LẼ**
  - **Mạch phát chẵn lẽ**
  - **Mạch kiểm chẵn lẽ**

I. GIỚI THIỆU

Các mạch số được chia thành 2 loại mạch: Mạch tổ hợp và mạch tuần tự.

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

  Nghĩa là: Q = f(A, B, C,…).

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

  Nghĩa là: $Q_+ = f(Q,A, B, C,…)$.

II. MẠCH MÃ HÓA

1. Giới thiệu

Mã hóa là gán một ký hiệu cho một đối tượng để thực hiện một yêu cầu cụ thể nào đó. Ví dụ, mã BCD gán số nhị phân cho từng số mã của số thập phân để thuận tiện cho việc đọc một số có nhiều số mã. Mã Gray dùng thuận tiện trong việc tối giản các hàm logic,… Mạch dùng để chuyển mã từ mã này sang mã kia gọi là mạch chuyển mã, cũng là một loại mạch mã hoá.

2. Mạch mã hoá từ 2^n đường sang n đường

a. Giới thiệu mạch mã hoá và mạch mã hoá ưu tiên

Một số nhị phân n bit cho 2^n tổ hợp khác nhau. Vậy có thể dùng số n bit để mã cho 2^n ngã vào khác nhau. Khi có một ngã vào được tác động, ở ngã ra chỉ báo số nhị phân tương ứng. Đó là mạch mã hoá 2^n đường sang n đường.

Để tránh trường hợp mạch cho một mã sai khi người sử dụng vô tình (hay cố ý) tác động đồng thời vào 2 hay nhiều ngã vào, người ta thiết kế mạch mã hoá ưu tiên: ***Chỉ cho một mã duy nhất có tính ưu tiên khi nhiều ngã vào cùng được tác động***.

b. Mã hoá ưu tiên từ 4 đường sang 2 đường

Thiết kế mạch mã hoá ưu tiên từ 4 đường sang 2 đường, ưu tiên cho mã có trị cao và vào/ra tác động cao.

Dưới đây là bảng sự thật và sơ đồ mạch. Do các ngã ra A_1 và A_0 không phụ thuộc vào cột 0, nên trong bảng đồ Karnaugh ta chỉ dùng các cột 1, 2, 3 (Dĩ nhiên nếu dùng 4 cột 0, 1, 2, 3 kết quả cũng vậy). Do A_0 bằng 1 tại 100 (4), ××1 (1, 3, 5, 7), tương tự cho A_1. Ta có bảng sự thật cho A_0 và A_1 như sau:

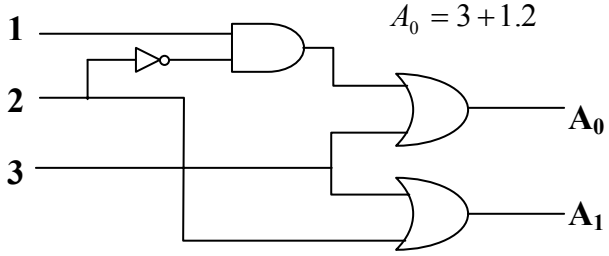
0	1	2	3	A_1	A_0
1	0	0	0	0	0
×	1	0	0	0	1
×	×	1	0	1	0
×	×	×	1	1	1

1,2 \ 3	0	1
00		1
01		1
11		1
10	1	1

$A_0 = 3 + 1.\overline{2}$

1,2 \ 3	0	1
00		1
01		1
11	1	1
10	1	1

$A_0 = 3 + 2$



Hình: Bảng sự thật, bảng Karnaugh, sơ đồ mạch của mạch mã hoá ưu tiên từ 4 đường sang 2 đường.

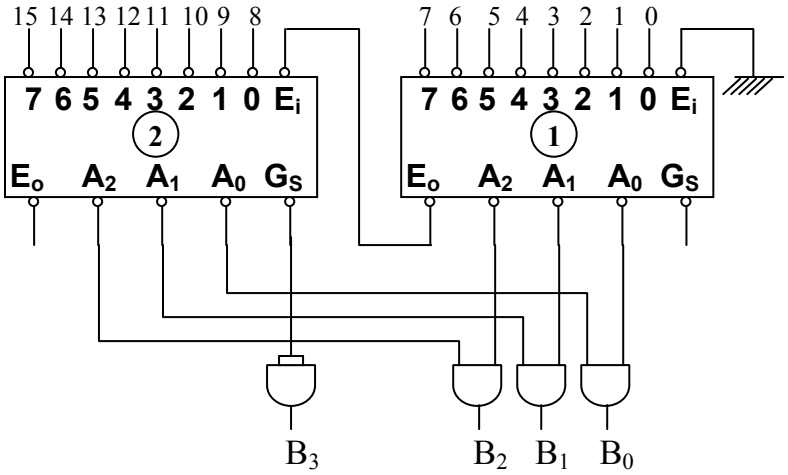
c. Mã hoá ưu tiên từ 8 đường sang 3 đường

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

Dưới đây là bảng sự thật của IC 74148.

Trạng thái	Ngã vào									Ngã ra				
	E_i	0	1	2	3	4	5	6	7	A_2	A_1	A_0	G_S	E_O
9	1	×	×	×	×	×	×	×	×	1	1	1	1	1
8	0	1	1	1	1	1	1	1	1	1	1	1	1	0
7	0	×	×	×	×	×	×	×	0	0	0	0	0	1
6	0	×	×	×	×	×	×	0	1	0	0	1	0	1
5	0	×	×	×	×	×	0	1	1	0	1	0	0	1
4	0	×	×	×	×	0	1	1	1	0	1	1	0	1
3	0	×	×	×	0	1	1	1	1	1	0	0	0	1
2	0	×	×	0	1	1	1	1	1	1	0	1	0	1
1	0	×	0	1	1	1	1	1	1	1	1	0	0	1
0	0	0	1	1	1	1	1	1	1	1	1	1	0	1

Dưới đây là cách ghép 2 IC mã hoá ưu tiên từ 8 đường sang 3 đường thành 16 đường sang 4 đường.



Hình: Cách ghép 2 IC từ 8 đường sang 3 đường thành 16 đường sang 4 đường.

Hoạt động của mạch như sau:

- IC1 có $E_i = 0$ nên hoạt động các **trạng thái từ 0 đến 8** nghĩa là mã hóa từ 0 đến 7 cho các ngã ra $A_2A_1A_0$.
- IC2 có E_i nối với E_o của IC1 nên:
  - Khi các ngõ vào của IC1 có giá trị từ 0 đến 7 thì $E_{i2} = E_{o1} = 1$, vậy IC2 sẽ hoạt động ở "**trạng thái 9"** (trong bảng sự thật của IC74148), nghĩa là bất chất các ngã vào, các ngã ra luôn bằng 1, đây là điều kiện mở cổng AND cho ra các số $B_2B_1B_0$. Lúc này B_3 chính là G_{S2} ($B_3 = G_{S2} = 1$). Ta được kết quả từ 0 đến 7 (tác động ở trạng thái thấp).
  - Khi các ngõ vào của IC1 có giá trị 1 "**trạng thái 8"** thì $E_{i2}=E_{o1}=0$, vậy IC2 sẽ hoạt động, các cổng ra của IC1 = 1 nên nó sẽ mở cổng AND để IC2 hoạt động cho các số từ 8 đến 15, do chân $G_{S2}=B_3=0$ (tác động ở trạng thái thấp).

d. Mạch tạo mã BCD sang 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 được tác động lên mức cao, ngã ra sẽ cho số BCD tương ứng.

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

Từ bảng sự thật, ta có phương trình các ngã ra như sau:

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

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

e. Mạch chuyển mã nhị phân sang Gray

Chuyển mã này sang mã khác cũng thuộc bài toán mã hóa.

Ta thử thiết kế mạch chuyển từ mã nhị phân sang mã Gray của số nhị phân 4 bit. Trước tiên, ta viết bảng sự thật của mã nhị phân và mã Gray tương ứng. Các số nhị phân là các biến, các số Gray sẽ là các hàm của các biến đó.

Dùng bảng Karnaugh để tối giản hàm trước khi thực hiện mạch.

Mã nhị phân					Mã Gray			
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

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

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

Vậy cần lập 3 bảng Karnaugh cho Y, Z, T.

AB \ CD	00	01	11	10
00				
01	1	1	1	1
11				
10	1	1	1	1

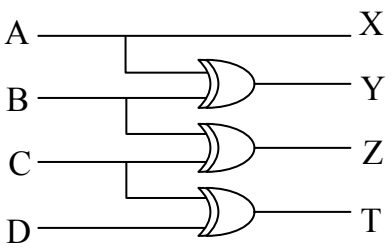
$Y = \overline{A}B + A\overline{B} = A \oplus B$

AB \ CD	00	01	11	10
00			1	1
01	1	1		
11	1	1		
10			1	1

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

AB \ CD	00	01	11	10
00		1		1
01		1		1
11		1		1
10		1		1

$Z = \overline{C}D + C\overline{D} = C \oplus D$



III. MẠCH GIẢI MÃ

1. Giải mã n đường sang 2^n đường

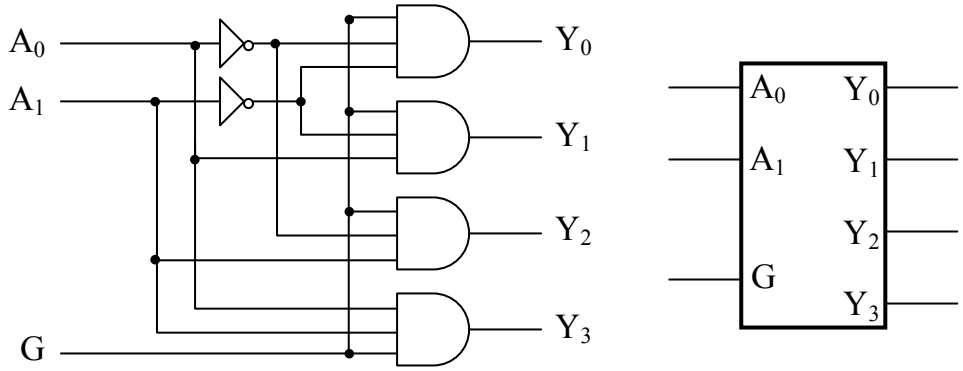
a. Giải mã 2 đường sang 4 đường

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

Bảng sự thật và sơ đồ mạch:

Vào			Ra			
G	A_1	A_0	Y_0	Y_1	Y_2	Y_3
0	×	×	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

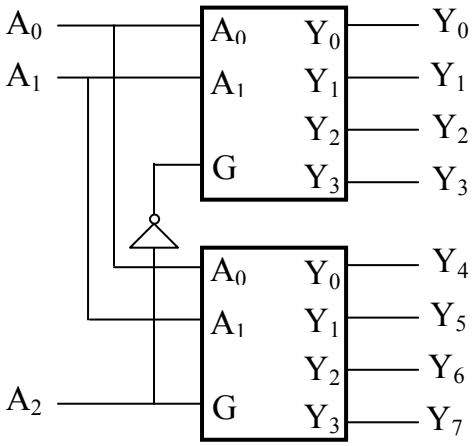
$$Y_0 = G.\overline{A_1}.\overline{A_0}$$
$$Y_1 = G.\overline{A_1}.A_0$$
$$Y_2 = G.A_1.\overline{A_0}$$
$$Y_3 = G.A_1.A_0$$



Hình: Sơ đồ mạch và ký hiệu của IC giải mã từ 2 đường sang 4 đường.

b. Giải mã 3 đường sang 8 đường

Dùng 2 IC giải mã từ 2 đường sang 4 đường để thực hiện mạch giải mã 3 đường sang 8 đường.



Hình: Giải mã từ 3 đường sang 8 đường.

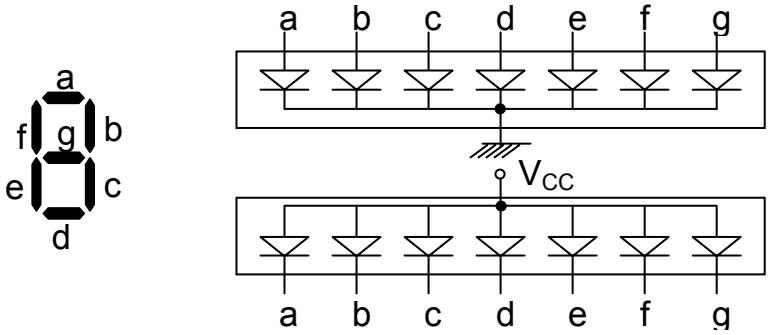
Trên thị trường có bán các loại IC sau:

- 74139 là IC chứa 2 mạch giải mã từ 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 từ 3 đường sang 8 đường có ngã vào tác động cao, các ngã ra tác động thấp, hai ngã E_1 và E_2 tác động thấp, E_3 tác động cao.
- 74154 là IC giải mã 4 đường sang 16 đường có ngã vào tác đọng cao, các ngã ra tác động thấp, hai ngã vào cho phép E_1 và E_2 tác động thấp.

c. Giải mã BCD sang 7 đoạn

- ***Đèn 7 đoạn***: Đây là loại đèn hiển thị các số từ 0 đến 9, đèn gồm 7 đoạn a, b, c, d, e, f, g, bên dưới mỗi đoạn là 1 led (đèn nhỏ) hoặc một nhóm led mắc song song. Qui ước các đoạn qui định bởi hình dưới đây.



Khi một tổ hợp, các đoạn cháy sáng sẽ tạo thành một con số thập phân từ 0 đến 9. Đèn 7 đoạn còn hiển thị được một số chữ cái và một số ký tự đặc biệt.

Có 2 loại đèn 7 đoạn: Loại catod chung và loại anod chung.

- ***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.

Khi led 7 đoạn thuộc loại catod chung thì thì mạch giải mã có ngã ra tác động ở mức cao (và ngược lại cho anod chung).

Bảng sự thật của mạch 7 đoạn, ngã ra tác động thấp.

Số TP	Ngã vào				Ngã ra						
	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

Dùng bảng Karnaugh hoặc có thể đơn giản các hàm có ít tổ hợp ta đươc:

$a = \overline{D}\overline{B}(C\overline{A} + \overline{C}A)$ $\quad c = \overline{D}\overline{C}B\overline{A}$ $\quad d = \overline{D}\overline{C}\overline{B}A + C\overline{B}\overline{A} + CBA$

$b = C\overline{B}A + CB\overline{A}$ $\quad e = A + C\overline{B}$ $\quad f = \overline{C}B + BA + \overline{D}\overline{C}A$ $\quad g = \overline{D}\overline{C}\overline{B} + CBA$

Từ các kết quả trên, ta có thể vẽ mạch giải mã 7 đoạn dùng các cổng logic.

Hai IC thông dụng dùng giải mã BCD sang 7 đoạn là: CD4511(loại CMOS ngõ ra tác động cao và có cổng đệm) và 7447 (loại TTL, ngã ra tác động thấp, cực thu để hở).

Ngoài ra, IC còn có một số ngã vào ra điều khiển khác như: LT (thử đèn), RBI (vào xóa dợn sóng), RBO (ra xóa dợn sóng), các chân RBI và RBO kết hợp để thực hiện việc cho phép hiển thị số 0 có nghĩa và không hiển thị nó khi không có nghĩa, LE (cho phép chốt).

Ghi chú: Sinh viên nên tìm tài liệu nghiên cứu thêm về IC 7447.

IV.MẠCH ĐA HỢP VÀ MẠCH GIẢI ĐA HỢP

1. Khái niệm

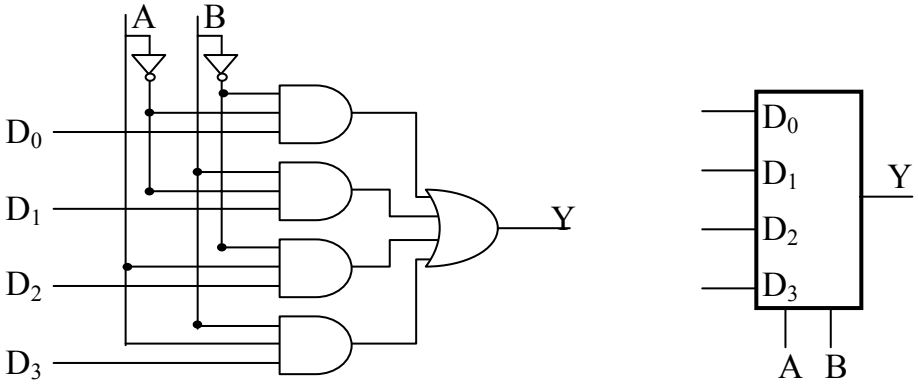
Trong truyền dữ liệu, để tiết kiệm đường truyền, người ta dùng một đường dây để truyền nhiều kênh dữ liệu, như vậy phải thực hiện chọn nguồn dữ liệu nào trong các nguồn khác nhau để truyền. Mạch đa hợp hay còn gọi là mạch chọn dữ liệu sẽ làm công việc này. Ở nơi thu, dữ liệu phải được phân bố cho các đích khác nhau, ta cần mạch phân bố dữ liệu hay mạch giải đa hợp.

Hình: Mô hình dùng mạch đa hợp, mạch giải đa hợp truyền dữ liệu.

2. Mạch đa hợp

Còn gọi là mạch chọn dữ liệu, gồm 2^n ngã vào dữ liệu, n ngã vào địa chỉ (hay điều khiển) và 1 ngã ra. Khi một ngã vào địa chỉ được tác động, dữ liệu ngã vào tương ứng với địa chỉ đó sẽ được chọn.

Mạch đa hợp được thiết kế dựa trên mạch giải mã. Dưới đây, là mạch đa hợp 4 sang 1. Mạch có 4 ngã vào dữ liệu D_0, D_1, D_2, D_3, hai ngã vào địa chỉ A, B và một ngã ra Y.



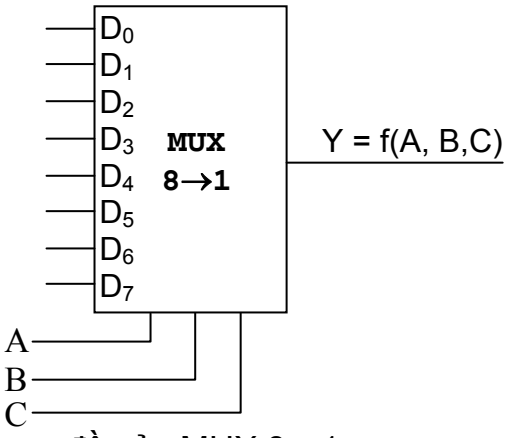
Hình: Mạch đa hợp 4 → 1.

Ngã ra của đa hợp xem như là hàm của biến ngã vào:

$$Y = \overline{A}\overline{B}.D_0 + \overline{A}B.D_1 + A\overline{B}.D_2 + AB.D_3$$

Mạch đa hợp từ 8 → 1, có 8 ngã vào dữ liệu, 3 ngã vào điều khiển, một ngã ra, được thiết kế như sau:

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



Hình: Bảng sự thật và sơ đồ của MUX 8→1.

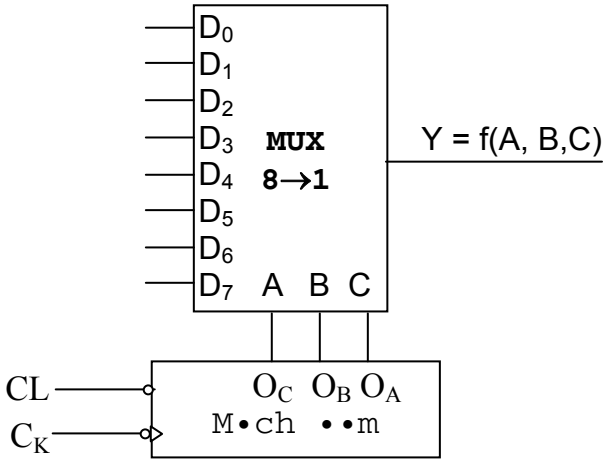
3. Ứng dụng mạch đa hợp

a. Chọn dữ liệu

Đây là chức năng ta đã xét ở phần trên của mạch đa hợp. Khi một ngã vào địa chỉ được tác động, dữ liệu ngã vào tương ứng với địa chỉ đó sẽ được chọn, mạch đa hợp đóng vai trò là SWITCH của các ngã vào dữ liệu.

b. Biến chuỗi dữ liệu song song thành nối tiếp

Mạch đa hợp kết hợp với mạch đếm sẽ biến chuỗi dữ liệu song song ở ngã vào thành chuỗi dữ liệu nối tiếp ở ngã ra.



Hình: Biến chuỗi dữ liệu song song thành nối tiếp.

c. Tạo chuỗi xung tuần hoàn

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

d. Tạo hàm

- Một đa hợp $2^n \rightarrow 1$ có thể tạo hàm n biến bằng cách cho các ngã vào điều khiển và cho trị riêng của hàm vào các ngã vào dữ liệu (đưa xuống mass nếu logic 0, đưa lên nguồn V_{CC} nếu logic 1 chẳn hạn).
- Một đa hợp $2^n \rightarrow 1$ kết hợp với cổng NOT có thể tạo hàm (n + 1) biến. Nếu kết hợp nhiều đa hợp, người ta có thể thực hiện hàm nhiều biến hơn.
- Ví dụ: Cài đặt hàm sau dùng đa hợp $4 \rightarrow 1$ (Dùng thêm cổng logic nếu cần).

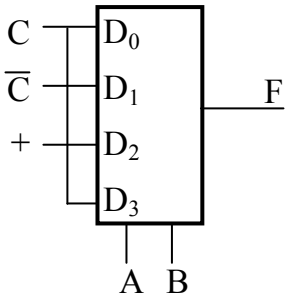
$$F = A\overline{B} + \overline{A}B\overline{C} + \overline{B}C + AC$$

Giải

Đa hợp $4 \rightarrow 1$ thực hiện hàm: $Y = \overline{A}\overline{B}.D_0 + \overline{A}B.D_1 + A\overline{B}.D_2 + AB.D_3$

Chuẩn hóa hàm F ta được: $F = \underbrace{\overline{A}\overline{B}.C}_{\overline{A}\overline{B}.D_0} + \underbrace{\overline{A}B.\overline{C}}_{\overline{A}B.D_1} + \underbrace{A\overline{B}.C + A\overline{B}.\overline{C}}_{A\overline{B}.D_2} + \underbrace{AB.C}_{AB.D_3}$

So sánh Y và F ta được: $D_0 = C;\ D_1 = \overline{C};\ D_2 = (C + \overline{C}) = 1;\ D_3 = C$



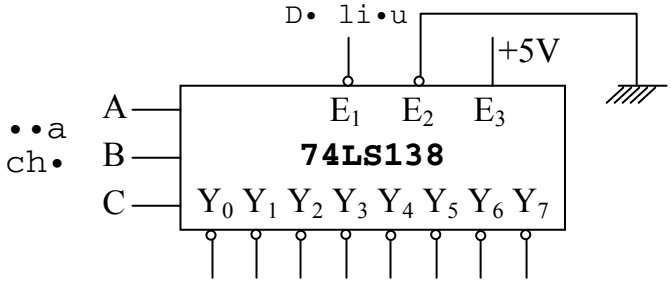
Hình: Mạch đa hợp thực hiện hàm logic.

Trên thực tế, ta có đủ các loại mạch đa hợp từ 2→1 (IC74157), 4→1 (IC74153), 8→1 (IC74151), 16→1 (IC74150),…

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 trên 1 IC, tuỳ theo điều kiện mà sử dụng. Ví dụ: IC 74138 là IC giải mã 3 đường 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. Hình dưới đây là IC 74138 dùng giải đa hợp cho dữ liệu vào ở E_1.



Hình: IC giải đa hợp.

V. MẠCH SO SÁNH

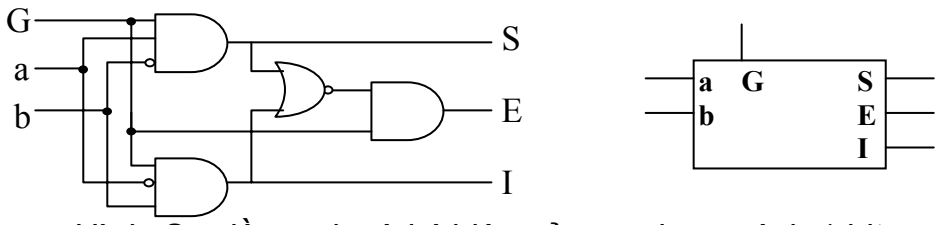
1. Mạch so sánh 2 số 1 bit

Bảng sự thật của mạch so sánh 1 bit có ngã vào nối mạch G.

G	A	b	S (a>b)	I (a<b)	E (a=b)
0	×	×	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

Từ bảng trên ta được kết quả sau:

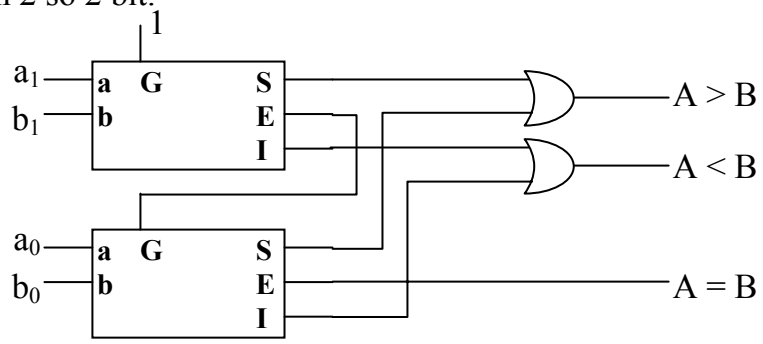
$S = G.a\bar{b}$ $\quad I = G.\bar{a}b$ $\quad E = G.(\overline{a \oplus b}) = G.(\overline{S + I})$



Hình: Sơ đồ mạch và ký hiệu của mạch so sánh 1 bit.

2. Mạch so sánh 2 số nhiều bit

Từ mạch so sánh 2 số 1 bit, ta có thể mở rộng so sánh nhiều bit. Dưới đây là sơ đồ mạch so sánh 2 số 2 bit.



Hình: Mạch so sánh số 2 bít A (a_1a_0) và B (b_1b_0).

Ngoài ra, 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.

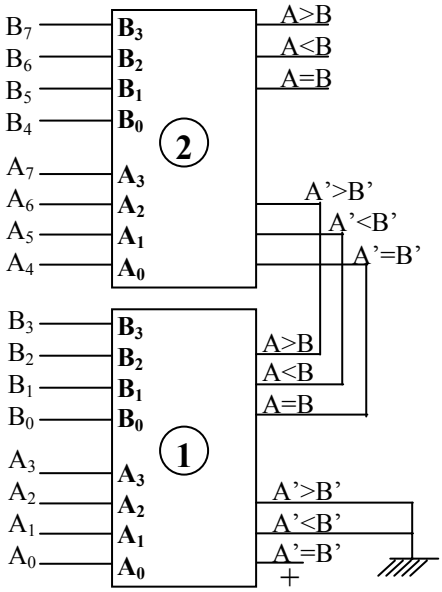
Bảng sự thật của IC 7485.

Trạng thái	Ngã vào so sánh				Ngã vào nối mạch			Ngã ra		
	A_3,B_3	A_2,B_2	A_1,B_1	A_0,B_0	A'>B'	A'<B'	A'=B'	A>B	A<B	A=B
1	$A_3>B_3$	×	×	×	×	×	×	1	0	0
2	$A_3<B_3$	×	×	×	×	×	×	0	1	0
3	$A_3=B_3$	$A_2>B_2$	×	×	×	×	×	1	0	0
4	$A_3=B_3$	$A_2<B_2$	×	×	×	×	×	0	1	0
5	$A_3=B_3$	$A_2=B_2$	$A_1>B_1$	×	×	×	×	1	0	0
6	$A_3=B_3$	$A_2=B_2$	$A_1<B_1$	×	×	×	×	0	1	0
7	$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0>B_0$	×	×	×	1	0	0
8	$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0<B_0$	×	×	×	0	1	0
9	$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	0	0	1	0	0	1
10	$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	1	0	0	1	0	0
11	$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	0	1	0	0	1	0

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 kết quả ở 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 của các IC so sánh bit cao hơn và IC so sánh các bit thấp nhất có ngã vào nối mạch được mắc như khi dùng riêng lẽ.

Ví dụ: Mắc IC 7485 để thực hện số so sánh 2 số 8 bit $A_7…A_0$ và $B_7…B_0$.



- So sánh 2 số $A_7…A_0$ = 10101111 và $B_7…B_0$ = 10110001. IC2 so sánh các bit cao $A_7A_6A_5A_4$ = 1010 và $B_7B_6B_5B_4$ = 1011, cho ngã ra A<B bất chấp các trạng thái của ngã vào nối mạch. Điều này có nghĩa nếu IC so sánh các bit cao khác nhau thì không quan tâm tới kết quả các bit thấp.
- So sánh 2 số $A_7…A_0$ = 10101111 và $B_7…B_0$ = 10101001. IC2 so sánh các bit cao $A_7A_6A_5A_4$ = 1010 và $B_7B_6B_5B_4$ = 1010 là bằng nhau, vậy kết quả tuỳ thuộc vào ngã vào nối mạch được nối với IC1. Kết quả so sánh của IC1 là $A_3A_2A_1A_0$ = 1111 và $B_3B_2B_1B_0$ = 1001 cho kết quả A>B, vậy chân A'>B' của IC2 lên mức logic cao nên IC2 cho kết quả A>B (trạng thái 10).

VI.MẠCH KIỂM PHÁT CHẴN LẺ

1. Giới thiệu

Do yêu cầu kiểm sai trong truyền dữ liệu, người ta có phương pháp kiểm tra chẳn lẽ. Trong phương pháp này, ngoài các bit dữ liệu, người ta thêm 1 bit kiểm tra sao cho tổng số bit 1 kể cả bit kiểm tra là số chẳn (kiểm tra chẳn) hoặc lẻ (kiểm tra lẻ).

1	0	1	1	0	0	1

1 bit chẳn lẻ thêm vào – KT lẻ.

1	1	0	0	1	0	1

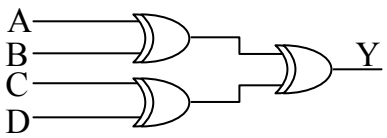
0 bit chẳn lẻ thêm vào – KT chẳn.

Ở nơi thu, mạch sẽ kiểm tra lại số số 1 trên tất cả các bit để biết dòng dữ liệu là đúng hay sai.

2. Mạch phát chẳn lẻ (Parity Generator)

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 chẳn lẻ.

- ***Giai đoạn 1***: Thiết kế mạch ghi nhận số số 1 là chẳn hay lẻ. Giả sử ta muốn có mạch báo kết quả Y = 1 khi số số 1 là lẻ, Y = 0 khi số số 1 là chẳn. Lợi dụng tính chất của hàm EX-OR có ngã ra bằng 1 khi số số 1 lẻ, với 4 ngã vào, ta dùng 3 cổng EX-OR để thực hiện mạch này. $Y = (A \oplus B) \oplus (C \oplus D)$.

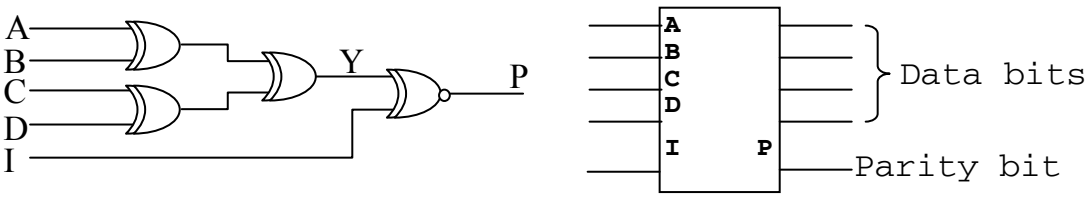


Hình: Ngã ra bằng 1 khi số số 1 vào lẽ.

- ***Giai đoạn 2***: Thiết kế mạch tạo bit chẳn lẻ P theo sự điều khiển của ngã vào I. Giả sử ta muốn có tổng số bit 1 của A, B, C, D, P là lẻ khi I = 0 và chẳn khi I = 1.

I	Số bit 1 của ABCD	Y	P
0	Lẻ	1	0
0	Chẳn	0	1
1	Lẻ	1	1
1	Chẳn	0	0

Từ bảng trên ta thấy: $P = \overline{I \oplus Y}$
Vậy mạch có dạng:



Hình: Sơ đồ mạch của bit P trong kiểm tra chẳn lẻ.

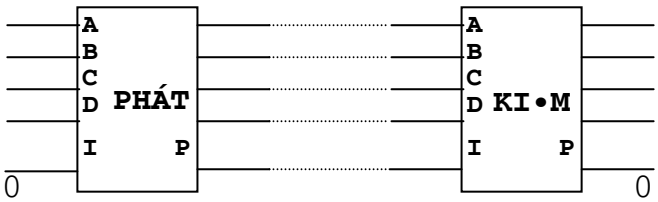
3. Mạch kiểm chẵn lẻ (Parity Checker)

Nếu ta xem mạch phát như là mạch có 5 ngã vào thì ngã ra P quan hệ với số lượng bit 1 ở các ngã vào có thể suy ra từ bảng sự thật trên.

Số bit 1 của ABCDI	P
Lẻ	0
Chẵn	1

Như vậy ta có thể dùng mạch phát trên để làm mạch kiểm tra chẵn lẻ.

Tóm lại, một hệ thống gồm mạch phát kiểm tra chẵn lẽ và mạch thu kiểm tra chẵn lẻ ta mắc chúng với nhau theo hình dưới đây.



Hình: Sơ đồ phát – thu của mạch kiểm tra chẵn lẻ.

Khi ngã vào I của mạch phát đưa xuống mức 0, nếu bản tin nhận đúng thì ngã ra P của mạch kiểm cũng xuống mức 0.

Trên thị trường có bán các IC kiểm phát chẵn lẻ như: 74180 (9bit), 74280 (9 bit), loại CMOS có 40101 (9 bit), 4531 (13 bit).

Dưới đây là bảng sự thật của IC 74180.

Ngã vào			Ngã ra	
Tổng số 1 bit dữ liệu	Chẵn	Lẻ	Tổng chẵn	Tổng lẻ
Chẵn	1	0	1	0
Lẻ	1	0	0	1
Chẵn	0	1	0	1
Lẽ	0	1	1	0
×	1	1	0	0
×	0	0	1	1

CHƯƠNG 5: MẠCH TUẦN TỰ

- ✓ **FLIPFLOP**
  - **FF RS**
  - **FF JK**
  - **FF T**
  - **FF D**
- ✓ **MẠCH GHI DỊCH**
- ✓ **MẠCH ĐẾM**
  - **Đồng bộ**
  - **Không đồng bộ**
  - **Đếm vòng**

I. GIỚI THIỆU

Trong chương trước, chúng ta đã khảo sát các loại mạch tổ hợp, đó là các mạch mà ngã ra của nó không phụ thuộc vào trạng thái trước đó của mạch. Nói cách khác, nó là loại mạch không có khả năng nhớ, một chức năng quan trọng của hệ thống logic.

Trong chương này, ta sẽ xét loại mạch thứ 2 là mạch tuần tự.

- Mạch tuần tự là mạch có ngã ra không những phụ thuộc vào các trạng thái ngã vào mà còn phụ thuộc vào 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.

  Nghĩa là: $Q_+ = f(Q,A,B,C,\ldots)$

- 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 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.

II. FLIPFLOP

1. Giới thiệu

Mạch flipflop (FF) là mạch đa hài lưỡng ổn tức mạch tạo ra sóng vuông và có 2 trạng thái cân bằng.

Trạng thái cân bằng của FF chỉ thay đổi khi có xung đồng hồ tác động.

Một FF thường có một hoặc nhiều ngã vào, và hai ngã ra. Tính nhớ của FF được thể hiện ở điểm: Trạng thái của FF vẫn được giữ nguyên mặc dù sự tác động ngã vào đã chấm dứt.

Hai ngã ra của FF thường được ký hiệu là Q (ngã ra chính) và $\overline{Q}$ (ngã ra phụ). Người ta thường chỉ trạng thái của FF bởi ngã ra chính của nó. Nếu hai ngã ra có trạng thái giống nhau ta nói FF ở trạng thái cấm.

FF có thể tạo nên từ các mạch chốt (latch).

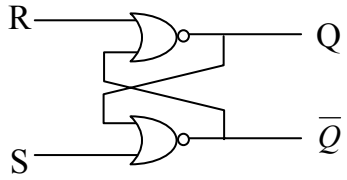
Điểm khác biệt giữa một mạch chốt và một FF là: FF chịu tác động của xung đồng hồ còn mạch chốt thì không.

Người ta gọi tên các FF khác nhau bằng cách dựa vào tên các ngã vào của chúng.

2. Chốt RS

a. Chốt RS tác động cao

Dưới đây là chốt RS có ngã vào R và S tác động ở mức cao.



Hình: Chốt RS tác động mức cao.

Các trạng thái logic của mạch được biểu diễn trong bảng dưới đây.

R	S	Q	Q_+	
0	0	0	0	Tác dụng nhớ
0	0	1	1	
0	1	0	1	Đặt (Set)
0	1	1	1	
1	0	0	0	Đặt lại (Reset)
1	0	1	0	
1	1	0	0	$Q_+=\overline{Q}_+$ (Cấm)
1	1	1	1	

R	S	Q_+
0	0	Q
0	1	1
1	0	0
1	1	Cấm

Từ bảng bên, ta tóm tắt lại hoạt động của chốt RS trong bảng trên.

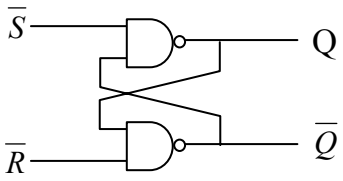
Từ bảng trên, ta tóm tắt hoạt động của RS như sau:

- Khi R = S = 0, ngã ra không đổi trạng thái.
- Khi R = 0 và S = 1, chốt được Set (tức đặt $Q_+ = 1$).
- Khi R = 1 và S = 0, chốt được Reset (tức đặt $Q_+ = 0$).
- Khi R = S = 1, chốt rơi vào trạng thái cấm.

b. Chốt RS tác động thấp

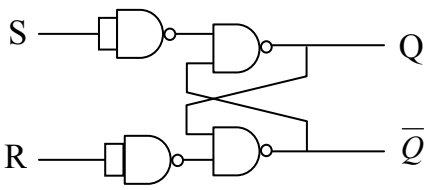
Dưới đây là chốt RS có ngã vào R và S tác động ở mức thấp.

S	R	Q_+
0	0	Cấm
0	1	1
1	0	0
1	1	Q

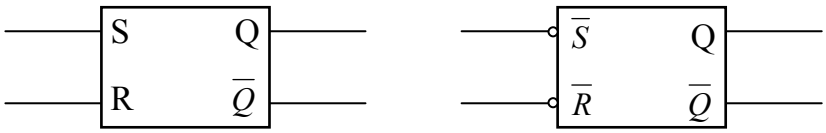


Hình: Chốt RS tác động mức thấp.

Để có chốt RS tác động mức cao dùng cổng NAND, người ta thêm hai cổng đảo ở các ngõ vào của mạch.



Hình: Chốt RS tác động mức cao.



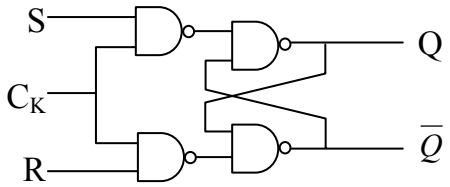
Hình: Ký hiệu chốt RS tác động mức cao và RS tác động thấp.

3. FlipFlop RS

a. Cấu trúc tổng quát FlipFlop RS

Trong các phần dưới đây, ta 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. Dưới đây là bảng sự thật FF RS có các ngã vào R, S và xung đồng hồ C_K đều tác động mức cao.

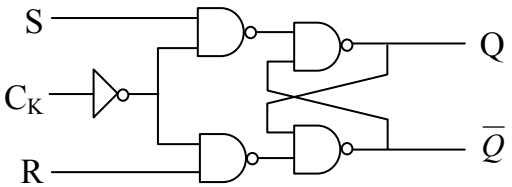
Vào			Ra
C_K	S	R	Q_+
0	×	×	Q
1	0	0	Q
1	0	1	0
1	1	0	1
1	1	1	Cấm



Hình: FF RS tác động mức cao.

Để có FF xung đồng hồ tác động mức thấp, ta thêm một cổng đảo cho ngã vào C_K. Ta được bảng sự thật giống như trên, ngoại trừ ngã vào C_K đảo ngược lại.

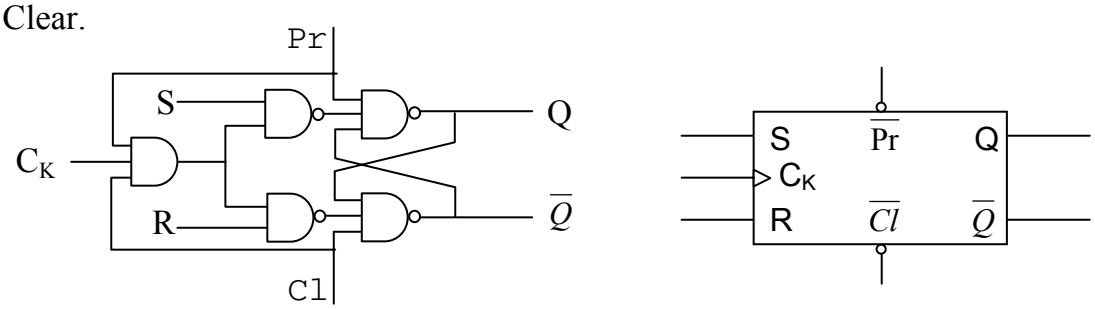
Vào			Ra
C_K	S	R	Q_+
1	×	×	Q
0	0	0	Q
0	0	1	0
0	1	0	1
0	1	1	Cấm



Hình: FF RS có C_K tác động mức thấp.

b. FlipFlop RS có ngã vào Preset và Clear

Tính chất của FF là có ngã ra bất kỳ khi mở máy. Trong nhiều trường hợp ta cần đặt trước ngã ra Q=1 hoặc Q=0, muốn thế, người ta thêm vào FF các ngã vào Preset (Q=1) và Clear (Q=0). Dưới đây là dạng mạch và ký hiệu của FF RS có ngã vào Preset và Clear.



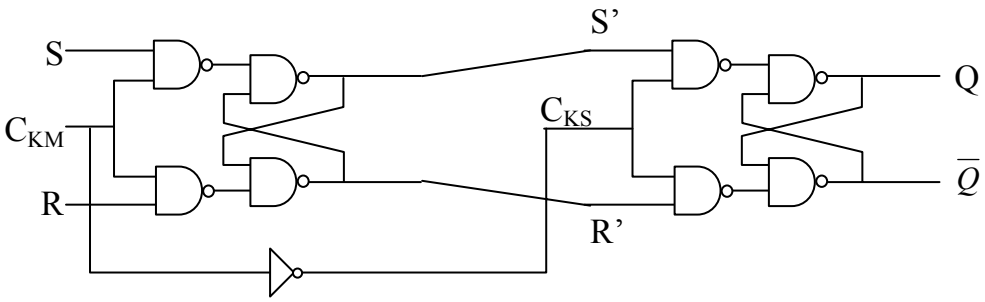
Hình: FlipFlop RS có ngã vào Preset và Clear.

Bảng sự thật của FF RS có Preset và Clear tác động thấp.

Pr	Cl	C_K	S	R	Q_+
0	0	×	×	×	Cấm
0	1	×	×	×	1
1	0	×	×	×	0
1	1	0	×	×	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

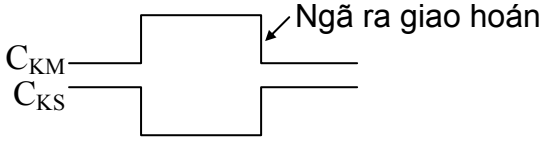
c. FlipFlop RS chủ tớ

Kết nối thành chuỗi hai FF RS với các ngã vào xung C_K của 2 FF có mức tác động ngược nhau, ta được FF chủ tớ. Với cách mắc này, mạch thoát khỏi trạng thái cấm (nhưng vẫn rơi vào trạng thái bất định) đồng thời có xung C_K tác động bằng cạnh.



Hình: Sơ đồ FF RS chủ tớ.

Hoạt động của FF được giải thích như sau: Do C_{KS} của tầng tớ là đảo của C_{KM} của tầng chủ, nên khi C_{KM} = 1, tầng chủ giao hoán và tầng tớ ngưng. Trong khoảng thời gian này, dữ liệu ngã vào R và S được đưa ra và đã ổn định ở ngã ra R' và S', tại thời điểm xung C_K xuống thấp, R' và S' được truyền đến ngã ra Q và $\overline{Q}$.



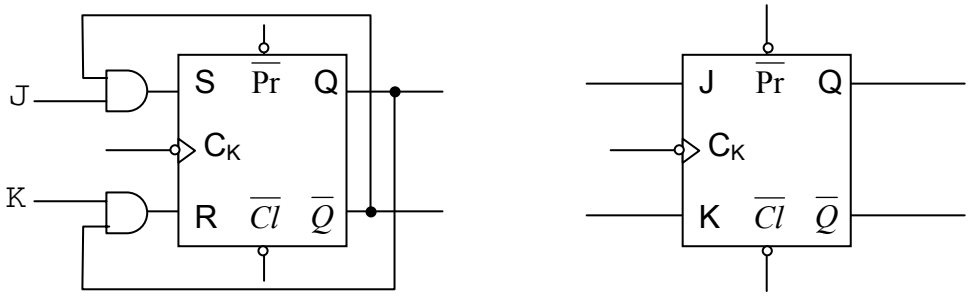
Hình: Vị trí xảy ra giao hoán.

Đối với trường hợp R = S = 1 khi C_{KM} = 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 bằng 1 có thể bằng 0, nhưng khác $\overline{Q}_+$. Ta có bảng sự thật như sau:

S	R	C_K	Q_+
0	0	↓	Q
0	1	↓	0
1	0	↓	1
1	1	↓	Bất định

4. FlipFlop JK

FF JK được tạo từ FF RS theo sơ đồ sau:



Hình: Cấu tạo FF JK có ngã vào Pr và Cl tác động thấp.

Bảng sự thật của FF JK.

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

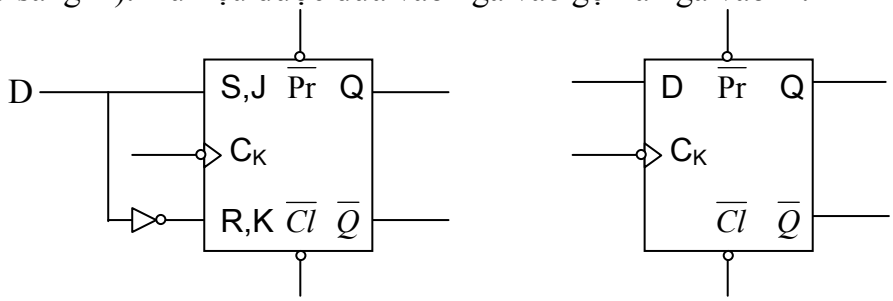
Từ bảng trên, ta có thể rút gọn thành bảng sau:

J	K	C_K	Q_+
0	0	↓	Q
0	1	↓	0
1	0	↓	1
1	1	↓	Đảo Q

Kết quả trên cho ta thấy: FF JK đã thoát khỏi trạng thái cấm và thay vào đó là trạng thái đảo (khi J=K=1). Người ta lợi dụng trạng thái này để thiết kế mạch đếm.

5. FlipFlop D

Thiết kế FF D từ FF RS (hoặc FF JK) bằng cách nối một cổng đảo từ S qua R (hoặc từ J sang K). Dữ liệu được đưa vào ngã vào gọi là ngã vào D.



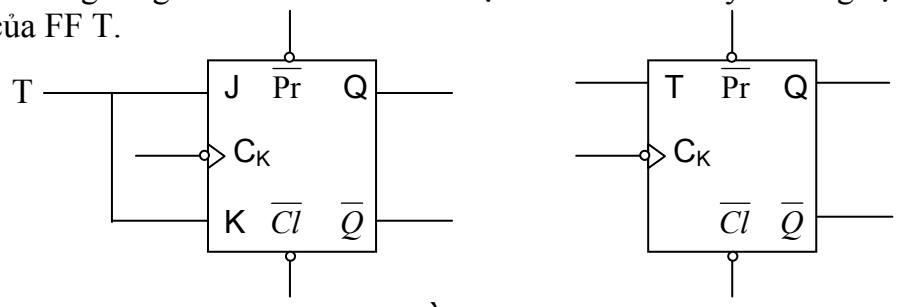
Hình: Sơ đồ và ký hiệu FF D.

Bảng sự thật của FF D được biểu diễn như sau:

D	C_K	Q_+
0	↓	0
1	↓	1

6. FlipFlop T

Nối chung 2 ngã vào của FF JK ta được FF T. Dưới đây là bảng sự thật và sơ đồ ký hiệu của FF T.



Hình: Sơ đồ và ký hiệu FF T.

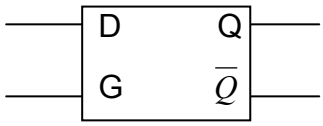
Bảng sự thật của FF T được biểu diễn như sau:

T	C_K	Q_+
0	↓	Q
1	↓	$\overline{Q}$

7. Mạch chốt D

Mạch chốt D hoạt động giống như FF D, chỉ khác nhau 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.

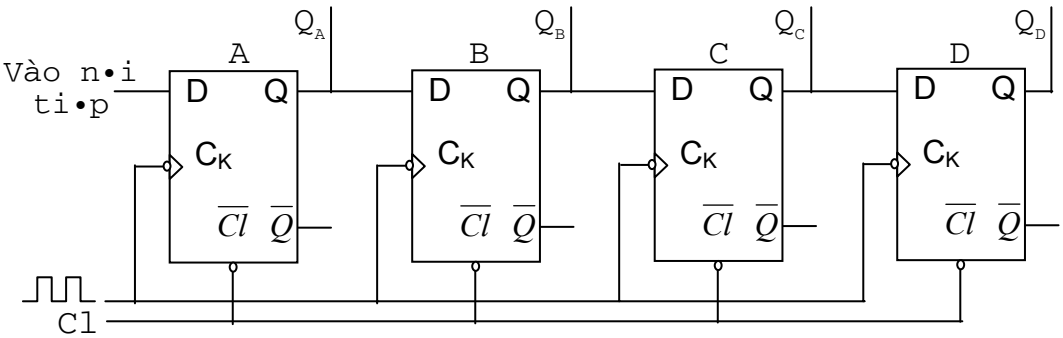
D	G	Q_+
×	0	Q
0	1	0
1	1	1



Hình: Ký hiệu mạch chốt D.

III. MẠCH GHI DỊCH

1. Sơ đồ nguyên tắc và vận chuyển



Hình: Sơ đồ mạch ghi dịch đơn giản.

Các FF D nối chung ngã vào C_K để được tác động đồng thời, ngã ra Q của FF trước nối với ngã vào D của FF sau. Ngã vào D_A của FF đầu tiên gọi là ngã vào của 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 mạch hoạt động, tác dụng một xung xóa các ngã vào $\overline{Cl}$ (đưa chân $\overline{Cl}$ xuống thấp rồi đưa lên cao như cũ) 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 của tầng trước lần lượt truyền qua tầng sau. Giả sử D_A có dữ liệu lần lượt vào như sau: 3 bit cao, 2 bit thấp, 1 cao, 1 thấp. Ta có bảng sự thật của sơ đồ mạch như sau:

Vào			Ra			
Cl	C_K	D_A	Q_A	Q_B	Q_C	Q_D
0	×	×	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

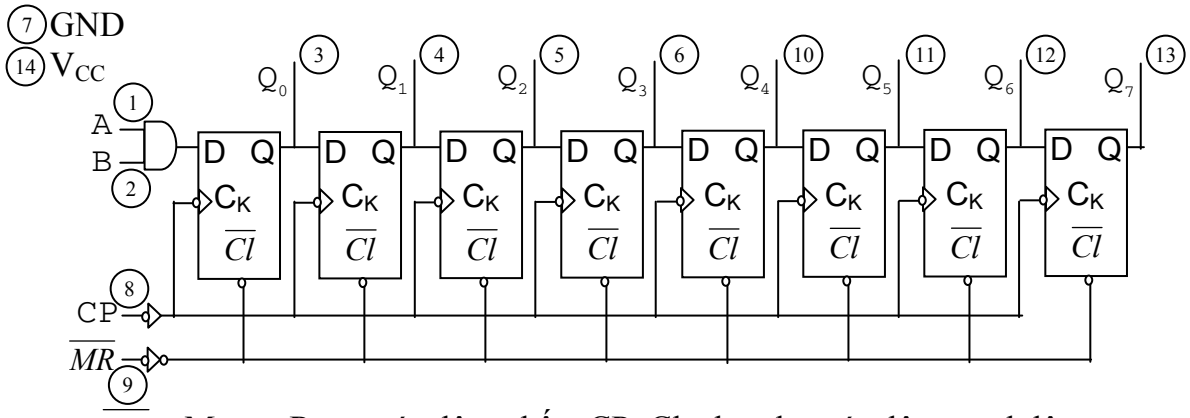
Các mạch ghi dịch được phân loại tuỳ vào số bit (số FF), chiều dịch (trái/phải), các ngã vào ra (nối tiếp/song song).

2. Vài IC ghi dịch tiêu biểu

a. Giới thiệu

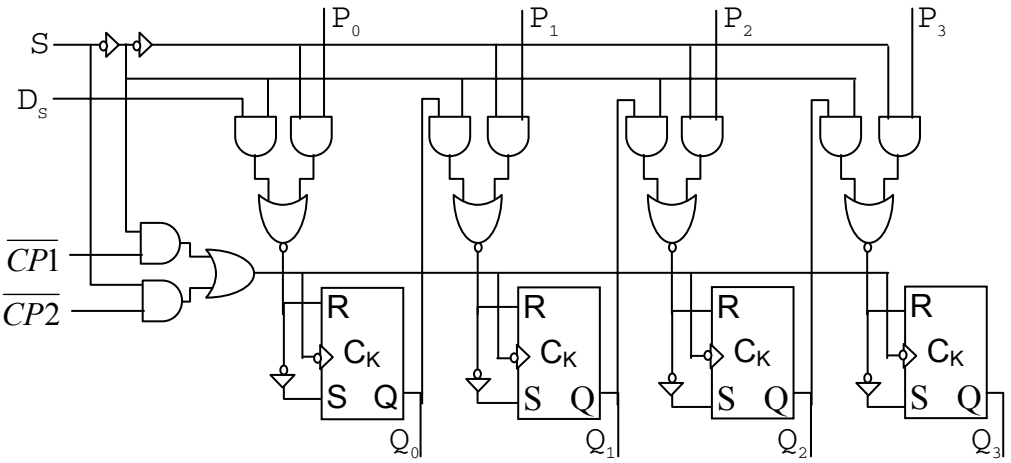
Trên thị trường hiện có hiện có khá nhiều loại IC ghi dịch có đầy đủ chức năng dịch trái, dịch phải, vào ra nối tiếp/song song. Sau đây, chúng ta khảo sát 2 IC tiêu biểu: IC74164 là IC dịch phải 8 bit, IC 7495 là IC 4 bit, dịch phải, trái, vào ra nối tiếp/song song.

b. IC 74164



$\overline{MR}$: Master Reset, tác động thấp. CP: Clock pulse, tác động cạnh lên.

c. IC 7495



Hình: Sơ đồ mạch IC 7495.

Ý nghĩa các chân

S: Mode control input.
D_S: Serial data input.
$P_0 \rightarrow P_3$: Parrallel data inputs.
CP1: Serial clock.
CP2: Parrallel clock.
$Q_0 \rightarrow Q_3$: Parrallel data outputs.

Nạp dữ liệu song song

- Chuẩn bị dữ liệu ngã vào P_0 đến P_3.
- Cho S = 1, dữ liệu được đưa vào các ngã vào của các FF, CP_1 bị khoá, CP_2 là ngã vào C_K, dữ liệu xuất hiện ở ngã ra Q_0 đến Q_3 khi có cạnh xuống của C_K.

Nạp dữ liệu nối tiếp

- Cho S = 0.
- Đưa dữ liệu nối tiếp vào D_S, CP_2 bị khoá, CP_1 là ngã vào C_K, khi có cạnh xuống của C_K dữ liệu dịch từng bit trên các ngã ra Q_0 đến Q_3.

Dịch phải

- Nạp dữ liệu song song.
- Đưa dữ liệu nối tiếp ở D_S và cho C_K tác động.

Dịch trái

- Nối ngã ra của FF sau vào ngã vào song song của FF trước.
- P_3 là ngã vào nối tiếp.
- Cho S = 1 để cách ly FF trước với FF sau.
- CP_2 là ngã vào xung C_K, dữ liệu sẽ được dịch trái ứng với cạnh xuống của xung C_K.

d. Ứng dụng của mạch ghi dịch

Mạch ghi dịch có nhiều ứng dụng.

- Một số nhị phân khi dịch trái một bit, giá trị nhị phân sẽ được nhân lên gấp đôi. Khi dịch phải 1 bit, giá trị nhị phân được chia 2 (lấy phần nguyên).

- Trong máy tính, thanh ghi là nơi lưu tạm dữ liệu để thực hiện các phép tính, các lệnh cơ bản như: quay, dịch phải, dịch trái,…
- Ngoài ra, mạch ghi dịch còn những ứng dụng khác như: tạo mạch đếm vòng, biến đổi nối tiếp ↔ song song.

IV.MẠCH ĐẾM

1. Giới thiệu

Lợi dụng tính đảo trạng thái của FF JK, người ta thực hiện 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 của ngã ra và nếu xét khía cạnh tần số của tín hiệu thì mạch đếm có chức năng của mạch chia tần, nghĩa là tần số tín hiệu ngã ra là kết quả của phép chia tần số của tín hiệu ngã vào cho một số nào đó.

2. Mạch đếm đồng bộ

a. Mạch đếm đồng bộ n tầng đếm lên

Trong các mạch đếm đồng bộ, các FF chịu tác động đồng thời của xung C_K.

Để thiết kế mạch đếm đồng bộ n tầng (ví dụ n = 4), trước tiên, ta lập bảng trạng thái, quan sát bảng trạng thái suy ra cách mắc ngã vào JK của các FF sao cho mạch giao hoán tạo trạng thái ngã ra đúng với bảng đã lập. Giả sử FF có xung C_K tác động ở cạnh xuống, với 4 FF mạch đếm được $2^4 = 16$ trạng thái và số xung đếm được từ 0 đến 15, với mạch đếm lên, ta có bảng trạng thái dưới đây.

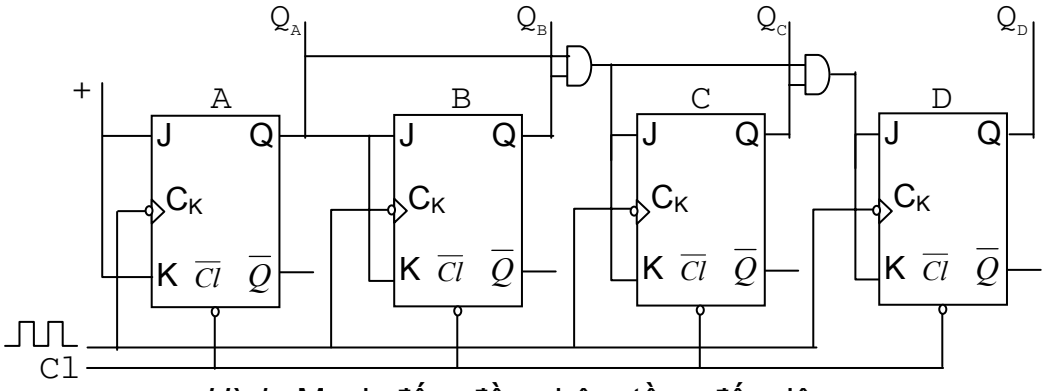
C_K	Q_D	Q_C	Q_B	Q_A	Số xung đếm
Xóa	0	0	0	0	0
↓	0	0	0✓	1	1
↓	0	0	1	0	2
↓	0	0✓	1✓	1	3
↓	0	1	0	0	4
↓	0	1	0✓	1	5
↓	0	1	1	0	6
↓	0✓	1✓	1✓	1	7
↓	1	0	0	0	8
↓	1	0	0✓	1	9
↓	1	0	1	0	10
↓	1	0✓	1✓	1	11
↓	1	1	0	0	12
↓	1	1	0✓	1	13
↓	1	1	1	0	14
↓	1✓	1✓	1✓	1	15
↓	0	0	0	0	0

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: $T_D=J_D=K_D=T_C.Q_C$.
Ta được kết quả như hình sau:



Hình: Mạch đếm đồng bộ n tầng đếm lên.

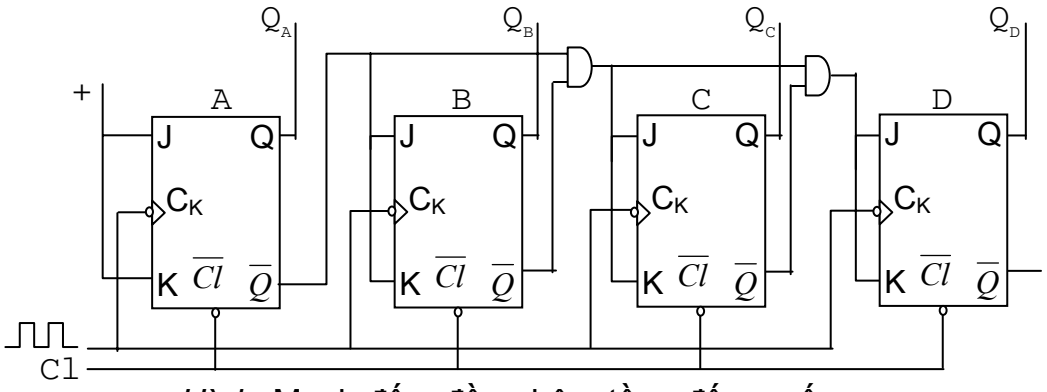
b. Mạch đếm đồng bộ n tầng đếm xuống

Giả sử FF có xung C_K tác động ở cạnh xuống, với 4 FF mạch đếm được $2^4 = 16$ trạng thái và số xung đếm được từ 0 đến 15, với mạch đếm xuống, ta có bảng trạng thái dưới đây.

C_K	Q_D	Q_C	Q_B	Q_A	Số đếm
Xóa	0	0✓	0✓	0	0
↓	1	1	1	1	15
↓	1	1	1✓	0	14
↓	1	1	0	1	13
↓	1	1✓	0✓	0	12
↓	1	0	1	1	11
↓	1	0	1✓	0	10
↓	1	0	0	1	9
↓	1✓	0✓	0✓	0	8
↓	0	1	1	1	7
↓	0	1	1✓	0	6
↓	0	1	0	1	5
↓	0	1✓	0✓	0	4
↓	0	0	1	1	3
↓	0	0	1✓	0	2
↓	0	0	0	1	1
↓	0	0	0	0	0

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 $Q_A = Q_B = Q_C = 1$, vậy: $T_D=J_D=K_D= T_C.\overline{Q}_C$.

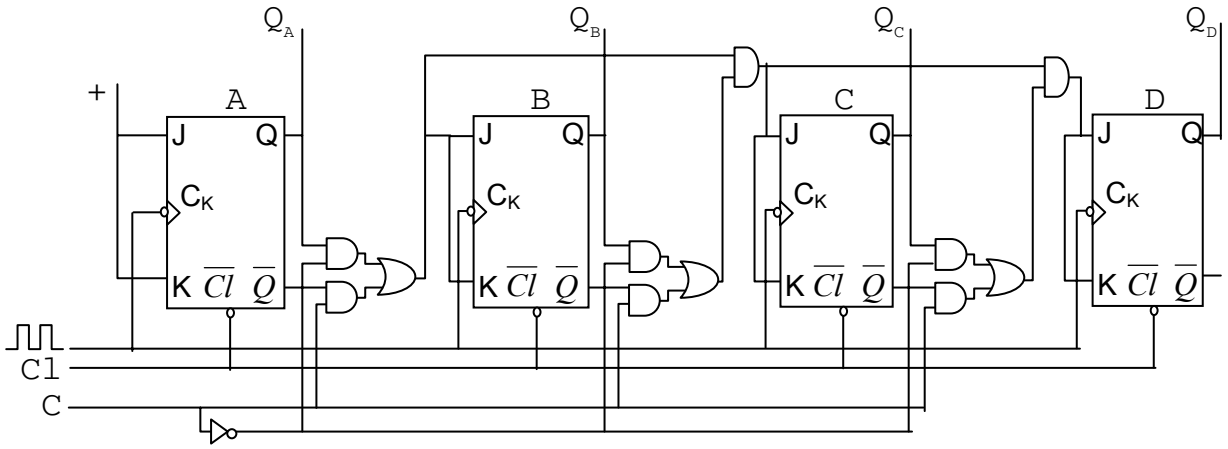
Ta được kết quả như hình sau:



Hình: Mạch đếm đồng bộ n tầng đếm xuống.

c. Mạch đếm đồng bộ n tầng đếm lên, xuống

Để có mạch n tầng đếm lên hoặc xuống, ta dùng một mạch 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 dưới đây, C = 0 mạch đếm lên, C = 1 mạch đếm xuống.



Hình: Mạch đếm đồng bộ n tầng đếm lên, xuống.

d. Tần số hoạt động lớn nhất của mạch đếm đồng bộ n tầng

Ta xét mạch đếm đồng bộ n tầng đếm lên, ta cần dùng 2 cổng AND. Trong trường hợp tổng quát cho n tầng, số cổng AND dùng là ***n – 2*** như vậy thời gian tối thiểu để tín hiệu truyền qua mạch là:

$$T_{min} = T_{P\,FF} + (n-2).T_{P\,AND}$$

Tầng số cực đại xác định bởi:

$$f_{max} = \frac{1}{T_{min}} = \frac{1}{T_{P\,FF} + (n-2)T_{P\,AND}}$$

Để gia tăng tần số làm việc của mạch, thay vì dùng cổng AND 2 ngã vào, ta phải dùng cổng AND nhiều ngã vào và mắc theo kiểu:

$$T_A = J_A = KA = 1.$$

$$TB = JB = KB = Q_A$$

$$TC = JC = KC = Q_A.Q_B$$

$$T_D = J_D = K_D = Q_A.Q_B.Q_C$$

Như vậy tần số làm việc không phụ thuộc vào n và bằng:

$$f_{\max} = \frac{1}{T_{\min}} = \frac{1}{T_{P\,FF} + T_{P\,AND}}$$

e. 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 là ***n*** phải thoả điều kiện: $2^{n-1} < N < 2^n$.

Ví dụ: Thiết kế mạch đếm 10 (N = 10).

Ta thấy $2^{4-1} = 2^3 < 10 < 2^4$, vậy số tần là 4.

Có nhiều phương pháp thiết kế mạch đếm đồng bộ modulo N. Sau đây, ta khảo sát hai phương pháp: Phương pháp dùng hàm chuyển và phương pháp MARCUS.

i. Ph••ng pháp dùng hàm chuy•n (Transfer function)

Hàm chuyển được định nghĩa như sau: Hàm có giá trị 1 khi có sự thay đổi trạng thái của FF và hàm có giá trị 0 khi FF không đổi trạng thái.

Ta sẽ xác định hàm chuyển của FF JK. Dưới đây là bảng trạng thái của FF JK và hàm chuyển H.

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

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.

Ví dụ: Thiết kế mạch đếm 10 đồng bộ dùng FF JK.

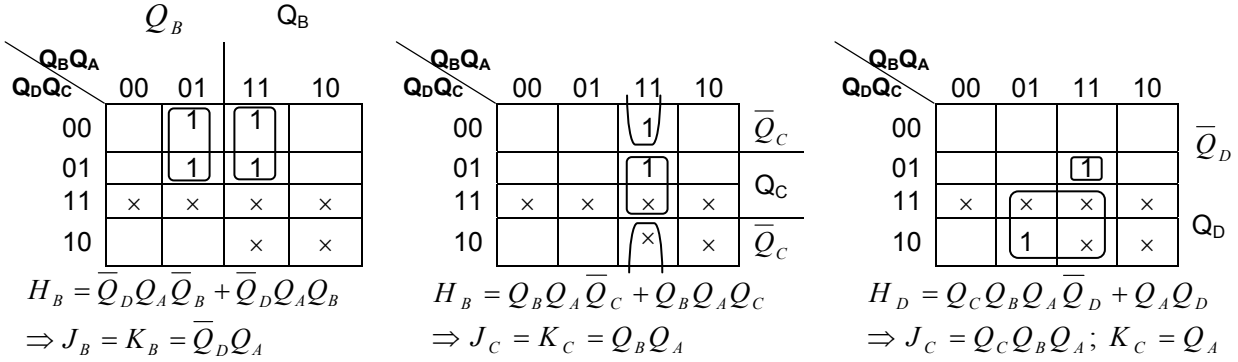
Bảng trạng thái của mạch đếm 10 và giá trị của hàm H tương ứng.

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	1	1

Từ bảng sự thật trên, ta thấy:

$$H_A = 1 = Q_A + \overline{Q}_A \Rightarrow J_A = K_A = 1$$

Để xác định H_B, H_C, H_D ta phải vẽ bảng đồ Karnaugh.



$H_B = \overline{Q}_D Q_A \overline{Q}_B + \overline{Q}_D Q_A Q_B$

$\Rightarrow J_B = K_B = \overline{Q}_D Q_A$

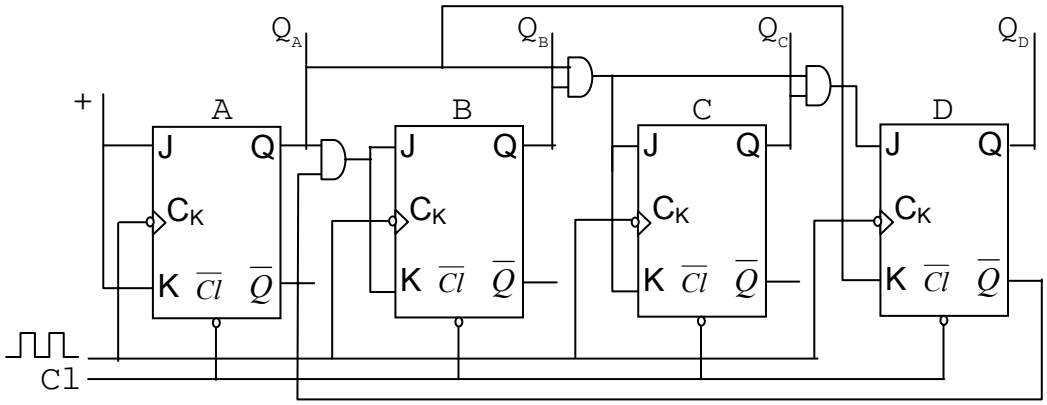
$H_B = Q_B Q_A \overline{Q}_C + Q_B Q_A Q_C$

$\Rightarrow J_C = K_C = Q_B Q_A$

$H_D = Q_C Q_B Q_A \overline{Q}_D + Q_A Q_D$

$\Rightarrow J_C = Q_C Q_B Q_A;\ K_C = Q_A$

Từ kết quả trên ta vẽ được mạch:



Hình: Mạch đếm 10.

ii. Ph••ng pháp MARCUS

Phương pháp MARCUS cho phép xác định các biểu thức của J, K dựa vào sự khác nhau của Q_+ so với Q sau mỗi lần tác động của xung C_K.

Từ bảng trạng thái của FF JK ta có thể rút gọn lại bảng sau:

Q	Q_+	J	K
0	0	0	×
0	1	1	×
1	0	×	1
1	1	×	0

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.

Ví dụ: Thiết kế mạch đếm 10 bằng phương pháp MARCUS.

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	×	0	×	0	×	1	×
2↓	0	0	0	1	0	×	0	×	1	×	×	1
3↓	0	0	1	0	0	×	0	×	×	0	1	×
4↓	0	0	1	1	0	×	1	×	×	1	×	1
5↓	0	1	0	0	0	×	×	0	0	×	1	×
6↓	0	1	0	1	0	×	×	0	1	×	×	1
7↓	0	1	1	0	0	×	×	0	×	0	1	×
8↓	0	1	1	1	1	×	×	1	×	1	×	1
9↓	1	0	0	0	×	0	0	×	0	×	1	×
10↓	1	0	0	1	×	1	0	×	0	×	×	1

Từ bảng sự thật trên, ta thấy:

$$J_A = K_A = 1$$

Dùng bảng Karnaugh xác định các hàm còn lại. Ta thấy rằng, FF B và FF C có thể xác định chung cho J và K vì chúng có cùng vị trí 1 và vị trí ×. FF D được xác định J và K riêng.

Q_DQ_C \ Q_BQ_A	00	01	11	10
00		1	1	
01		1	1	
11	×	×	×	×
10			×	×

$J_B = K_B = \overline{Q}_D Q_A$

Q_DQ_C \ Q_BQ_A	00	01	11	10
00			1	
01			1	
11	×	×	×	×
10			×	×

$J_C = K_C = Q_B Q_A$

Q_DQ_C \ Q_BQ_A	00	01	11	10
00				
01			1	
11	×	×	×	×
10	×	×	×	×

$J_D = Q_C Q_B Q_A$

Q_DQ_C \ Q_BQ_A	00	01	11	10
00	×	×	×	×
01	×	×	×	×
11	×	×	×	×
10		1	×	×

$K_D = Q_A$

Ta được kết quả như trên.

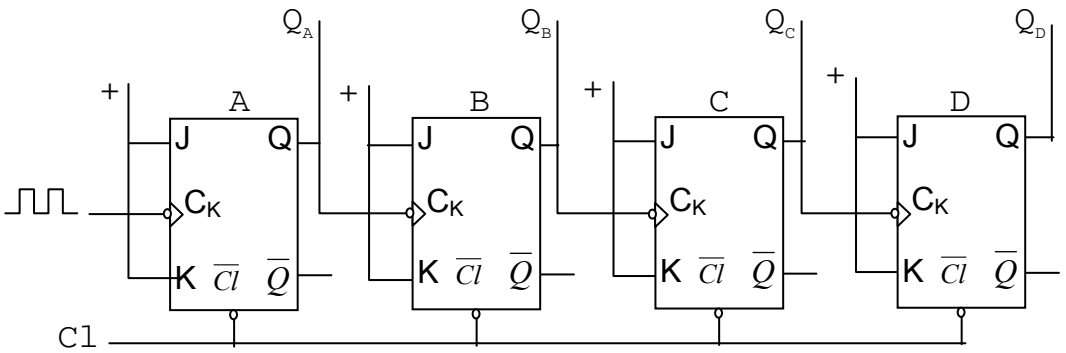
3. Mạch đếm không đồng bộ

a. Mạch đếm không đồng bộ n tầng đếm lên (n=4)

Trong các mạch đếm đồng bộ, xung C_K không tác động đồng thời lên các FF.

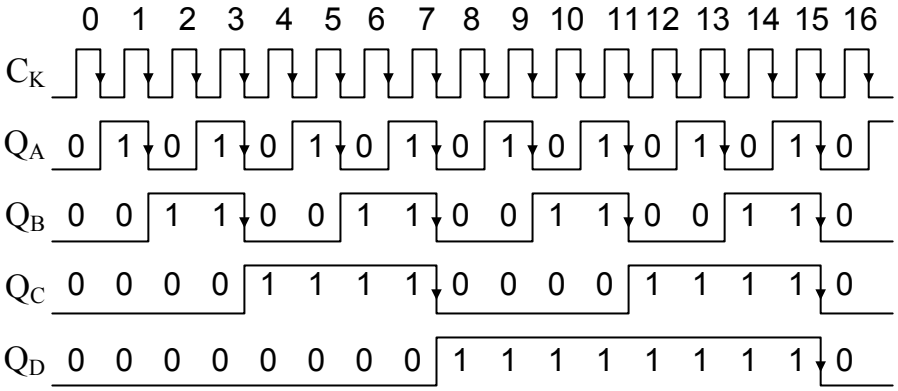
Từ bảng trạng thái của mạch đếm đồng bộ n tầng đếm lên (trình bày ở trên), ta thấy nếu dùng FF JK với xung đồng hồ tác động cạnh xuống thì có thể lấy ngã ra của tầng trước làm xung đồng hồ C_K cho tầng sau, với điều kiện các ngõ vào JK đều được đưa lên mức cao. Ta được mạch đếm không đồng bộ 4 bit, đếm lên.

Ta được kết quả như hình sau:



Hình: Mạch đếm không đồng bộ n tầng đếm lên.

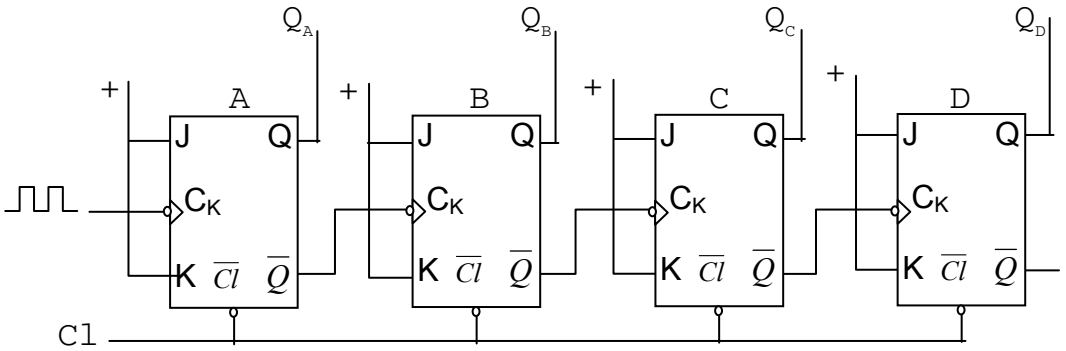
Dưới đây là tín hiệu của xung C_K và ngã ra của các FF.



Tổ hợp các số tạo bởi các ngã ra các FF D, C, B, A là số nhị phân từ 0 đến 15.

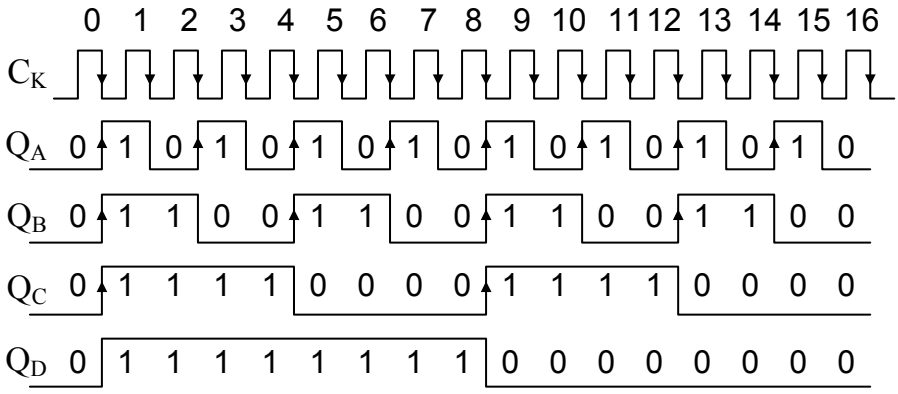
b. Mạch đếm không đồng bộ n tầng đếm xuống (n=4)

Để có mạch đếm không đồng bộ n tầng đếm xuống, ta nối $\overline{Q}$ của tầng trước vào ngã vào C_K của tầng sau. Dưới đây là sơ đồ mạch, sơ đồ xung C_K và sơ đồ các ngã ra của các FF.



Hình: Mạch đếm không đồng bộ n tầng đếm xuống.

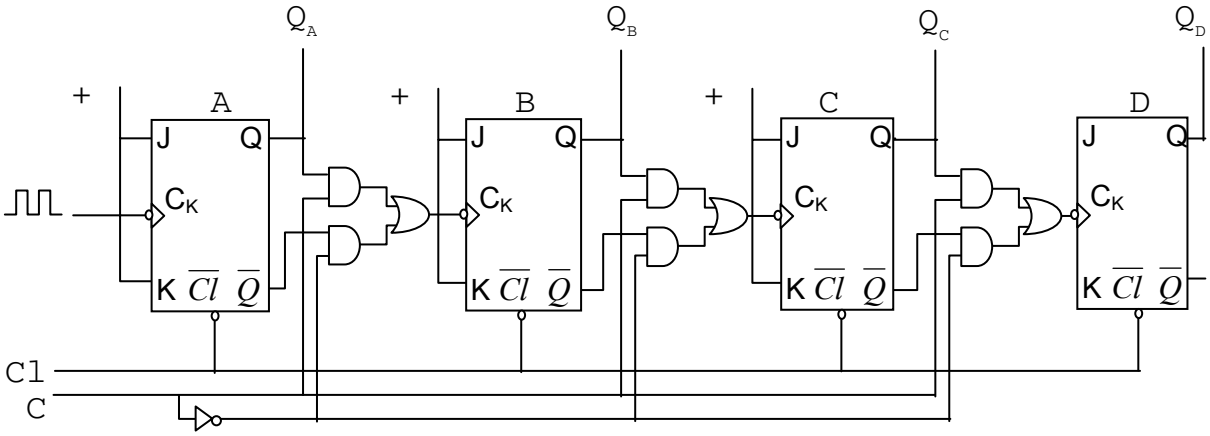
Dưới đây là tín hiệu của xung C_K và ngã ra của các FF.



Tổ hợp các số tạo bởi các ngã ra các FF D, C, B, A là số nhị phân từ 15 xuống 0.

c. Mạch đếm không đồng bộ n tầng đếm lên, xuống

Để có mạch n tầng đếm lên hoặc xuống, ta dùng một mạch 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 dưới đây, C = 0 mạch đếm xuống, C = 1 mạch đếm lên.



Hình: Mạch đếm không đồng bộ n tầng đếm lên, xuống.

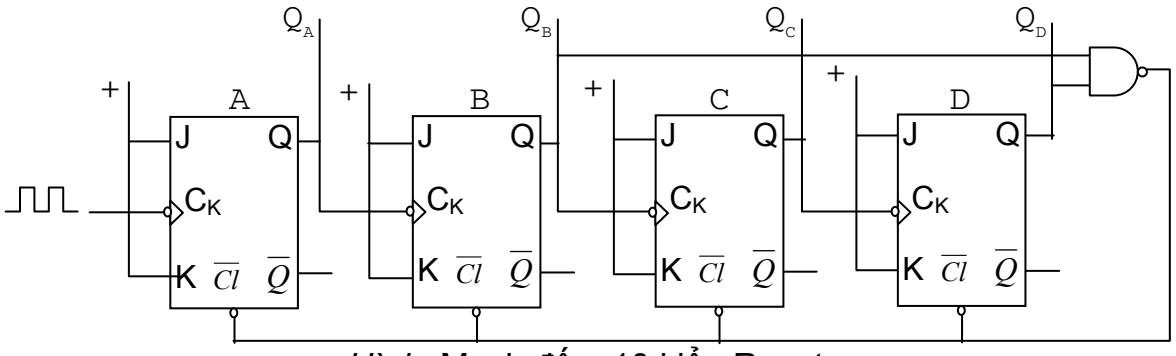
d. Mạch đếm không đồng bộ modulo – N (N = 10)

i. Kiểu Reset

Để thiết kế mạch đếm kiểu Reset, trước nhất người ta lập bảng trạng thái cho số đếm.

C_K	Q_D	Q_C	Q_B	Q_A	Dec
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

Quan sát bảng trên ta thấy, ở xung thứ 10, nếu theo cách đếm 4 tầng thì Q_D và Q_B phải lên 1 (số trong ngoặc). Lợi dụng 2 trạng thái này ta dùng một cổng NAND 2 ngã vào để đưa các tín hiệu về xoá các FF, ta được mạch đếm như dưới đây.



Hình: Mạch đếm 10 kiểu Reset.

ii. Kiểu Preset

Trong kiểu Preset các ngã vào của các FF sẽ được đặt trước hoặc nối với một ngã ra nào đó hoặc một mạch tổ hợp có ngã vào nối với các ngã ra của các FF để khi mạch đếm đến trạng thái thứ N thì tất cả các FF tự động quay về 0.

Thường người ta sẽ quan sát bảng trạng thái và kết hợp với phương pháp MARCUS để xác định JK của các FF và để dễ thiết kế, người ta phân $N = 2^n$. N' (N'<N) rồi kết hợp hai mạch đếm n bit và N'.

Ví dụ: Để thiết kế mạch đếm 10, ta chỉ cần thiết kế mạch đếm 5 rồi kết hợp với 1 FF (mạch đếm 2).

Dưới đây là bảng trạng thái của mạch đếm 5.

Số xung C_K vào	Số nhị phân			Số thập phân
	Q_D	Q_C	Q_B	
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

Giả sử FF JK có xung C_K tác động cạnh xuống. Từ bảng trên ta thấy, có thể dùng tín hiệu ngã ra của FF B làm xung đồng hồ cho FF C.

$$C_{KC} = Q_B$$

FF C, FF D dùng xung C_K của hệ thống và ngã vào JK được xác định theo phương pháp MARCUS.

C_K	Q_D	Q_C	Q_B	J_D	K_D	J_C	K_C
Xóa	0	0	0	0	×	1	×
1↓	0	0	1	0	×	×	1
2↓	0	1	0	0	×	1	×
3↓	0	1	1	1	×	×	1
4↓	1	0	0	×	1	0	×
5↓	0	0	0				

Ta thấy ngay $K_D = K_B = 1$.

Dùng bảng đồ Karnaugh xác định J_D và J_B.

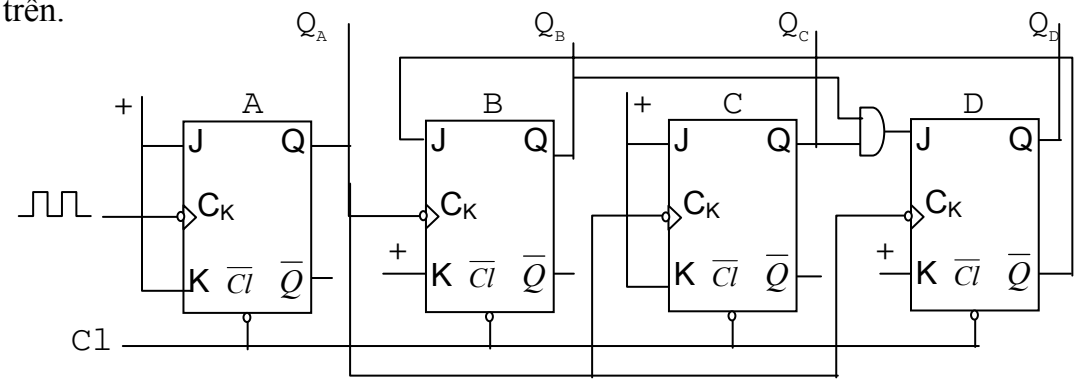
Q_D \ Q_CQ_B	00	01	11	10
0			**1**	
1	×	×	×	×

$J_D = Q_C.Q_B$

Q_D \ Q_CQ_B	00	01	11	10
0	**1**	×	×	**1**
1		×	×	×

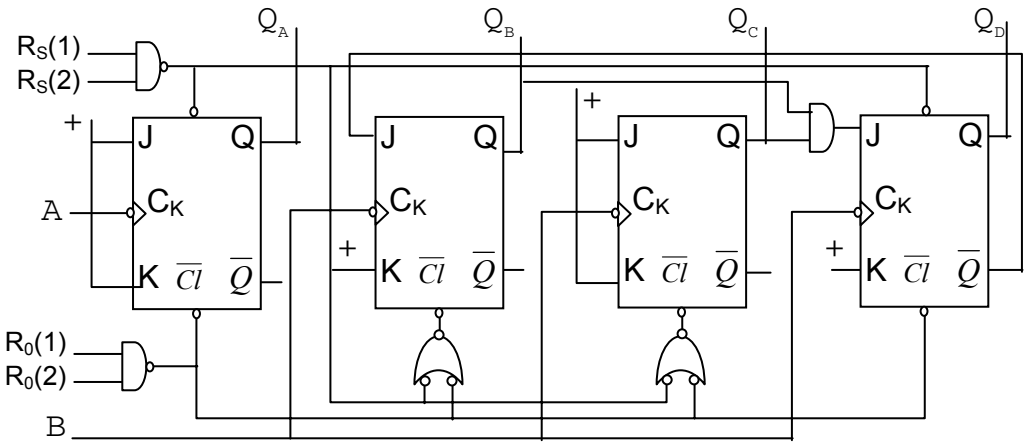
$J_B = \overline{Q}_D$

Dưới đây là mạch đếm 10 thiết kế theo kiểu 2×5, với mạch đếm 5 có được từ kết quả trên.



Hình: Mạch đếm 10 kiểu 2×5.

IC 7490 là IC đếm 10, có sơ đồ mạch với các ngã vào Reset như dưới đây.



Hình: Sơ đồ mạch IC 7490.

Dưới đây là bảng sự thật cho các ngã vào Reset.

Reset Input				Output			
$R_0(1)$	$R_0(2)$	$R_S(1)$	$R_S(1)$	Q_D	Q_C	Q_B	Q_A
1	1	0	×	0	0	0	0
1	1	×	0	0	0	0	0
×	×	1	1	1	0	0	1
×	0	×	0	Đếm	Đếm	Đếm	Đếm
0	×	0	×	Đếm	Đếm	Đếm	Đếm
0	×	×	0	Đếm	Đếm	Đếm	Đếm
×	0	0	×	Đếm	Đếm	Đếm	Đếm

IC 7490 có thể thực hiện một trong 2 cách mắc sau:

- Mạch đếm 2×5. Nối Q_A vào ngã vào B, xung C_K vào ngã vào A.
- Mạch đếm 5×2. Nối Q_D vào ngã vào A, xung C_K vào ngã vào B.

Hai cách mắc trên cho kết quả số đếm khác nhau nhưng cùng một chu kỳ 10. Tần số tín hiệu của ngã ra sau bằng 1/10 tần số xung C_K.

Dưới đây là bảng trạng thái cho 2 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	1	1

Đếm 2×5

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

Đếm 5×2

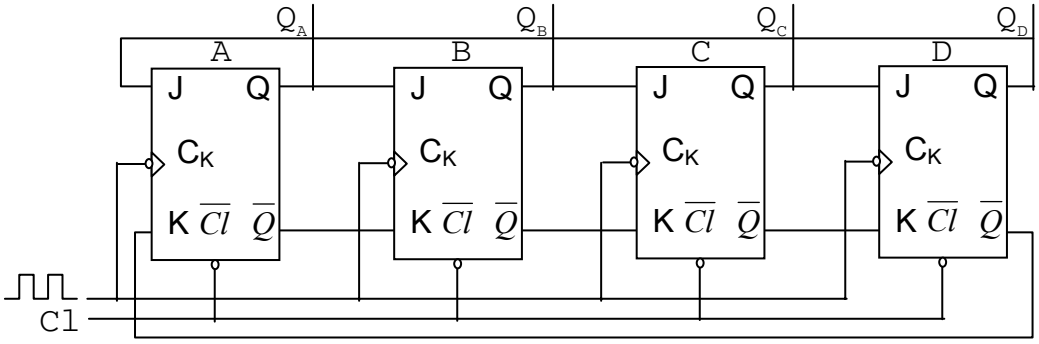
4. Mạch đếm vòng

a. Giới thiệu

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. Tuỳ đường hồi tiếp mà ta có chu kỳ đếm khác nhau.

Sau đây, ta khảo sát một vài loại mạch đếm vòng phổ biến.

b. Hồi tiếp từ Q_D về J_A và $\overline{Q}_D$ về K_A



Hình: Mạch hồi tiếp từ Q_D về J_A và $\overline{Q}$ về K_A.

Đối với mạch này, sự đếm vòng chỉ thấy khi có đặt trước ngã ra. Ta xét các ví dụ đặt trước $Q_A = 1$ và đặt trước $Q_A = Q_B = 1$, ta được bảng dưới đây.

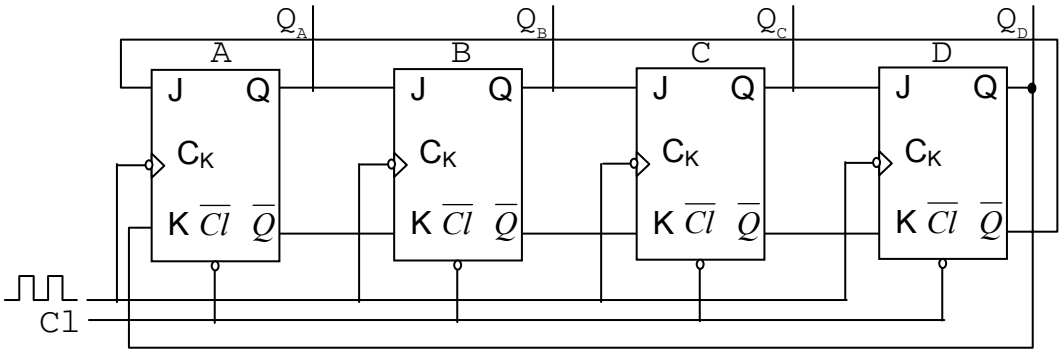
C_K	Q_D	Q_C	Q_B	Q_A	Dec
Preset	0	0	0	1	1
1↓	0	0	1	0	2
2↓	0	1	0	0	4
3↓	1	0	0	0	8
4↓	0	0	0	1	1
:	:	:	:	:	:

Đặt trước $Q_A = 1$

C_K	Q_D	Q_C	Q_B	Q_A	Dec
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
:	:	:	:	:	:

Đặt trước $Q_A = Q_B = 1$

c. Hồi tiếp từ $\overline{Q}_D$ về J_A và Q_D về K_A



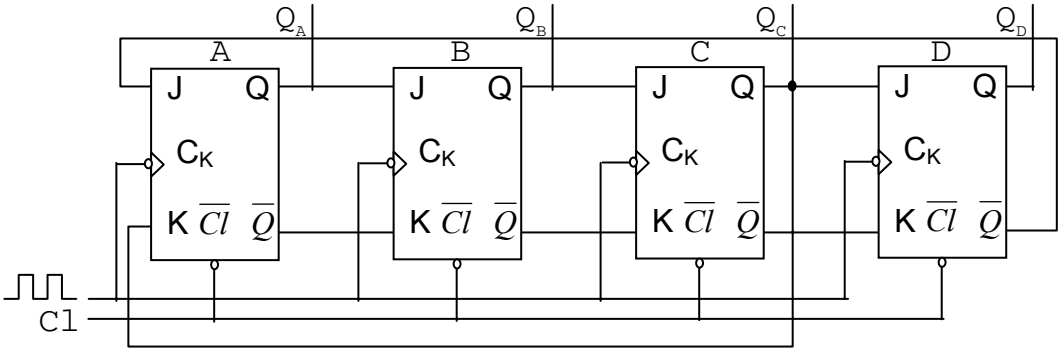
Hình: Hồi tiếp từ $\overline{Q}_D$ về J_A và Q_D về K_A.

Mạch này còn có tên là mạch Johnson. Mạc có chu kỳ đếm mặc nhiên mà không cần đặt trước. Nếu đặt trước, mạch sẽ cho các chu kỳ khác nhau, tùy vào tổ hợp đặt trước. Bảng dưới đây là chu kỳ đếm mặc nhiên.

C_K	Q_D	Q_C	Q_B	Q_A	Dec
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	1	12
7↓	1	0	0	0	8
8↓	0	0	0	0	0

Không đặt trước

d. Hồi tiếp từ $\overline{Q}_D$ về J_A và Q_C về K_A



Hình: Hồi tiếp từ $\overline{Q}_D$ về J_A và Q_C về K_A.

C_K	Q_D	Q_C	Q_B	Q_A	Dec
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	1	12
6↓	1	0	0	0	8
7↓	0	0	0	0	0

Không đặt trước

CHƯƠNG 6: MẠCH LÀM TOÁN

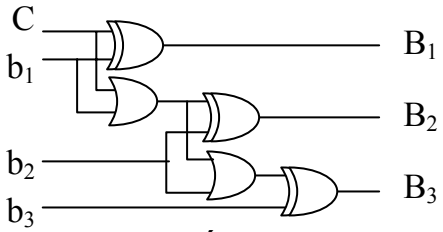
- ✓ SỐ BÙ
- ✓ PHÉP TOÁN VỚI SỐ BÙ 1
- ✓ PHÉP TOÁN VỚI SỐ BÙ 2
- ✓ PHÉP TOÁN VỚI SỐ BÙ 2 KỂ CẢ BIT DẤU
- ✓ MẠCH CỘNG
  - Bán phần
  - Toàn phần
  - Cộng nhiều bit
- ✓ MẠCH TRỪ
  - Bán phần
  - Toàn phần
  - Cộng trừ trong một mạch
- ✓ MẠCH NHÂN
- ✓ MẠCH CHIA

I. SỐ BÙ

Cho số dương N, n bit, các số bù của N được định nghĩa như sau:
Số bù 2: $(N)_2 = 2^n - N$.
Số bù 1: $(N)_1 = (N)_2 - 1 = 2^n - N - 1$.
Ví dụ 1: Ta cho N = 1010.
Số bù 2 của N là $(N)_2 = 10000 - 1010 = 0110$.
Và số bù 1 của N là $(N)_1 = 0110 - 1 = 0101$.
Ví dụ 2: Ta cho N = 1100 1010 1100.
Số bù 2 của N là $(N)_2 = 0011\ 0101\ 0100$.
Và số bù 1 của N là $(N)_1 = 0011\ 0101\ 0011$.
Nhận xét:

- Để có số bù 2 của một số, bắt đầu từ bit LSB (bit tận cùng bên phải), đi ngược về bên trái, các số 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 ta đảo tất cả các bit của số đó.

Từ nhận xét trên, ta có thể tạo mạch với số bù 1 và bù 2 (hình dưới).



Hình: Mạch tạo số bù 1 và bù 2 (3 bit).

Khi C = 1, mạch tạo ngã ra là số nhị phân bù 1 (của số ngã vào).
Khi C = 0, mạch tạo ngã ra là số nhị phân bù 2 (của số ngã vào).
Ta xét biểu thức ngã ra theo các ngã vào như sau:

$$B_1 = b_1 \oplus C$$
$$B_2 = b_2 \oplus (C + b_1)$$
$$B_3 = b_3 \oplus (C + b_1 + b_2)$$

Khi C = 1, các ngã ra của cổng OR luôn bằng 1, các cổng EX-OR luôn có 1 ngã vào bằng 1 nên ngã ra là đảo của ngã vào còn lại.

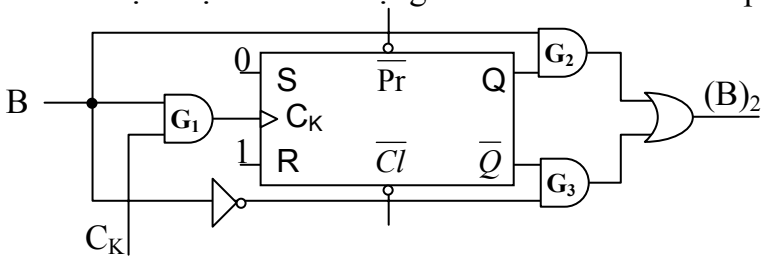
$$B_1 = b_1 \oplus 1 = \bar{b}_1$$
$$B_2 = b_2 \oplus (1 + b_1) = b_2 \oplus 1 = \bar{b}_2$$
$$B_3 = b_3 \oplus (1 + b_1 + b_2) = b_3 \oplus 1 = \bar{b}_3$$

Khi C = 0.

$$B_1 = b_1 \oplus 0 = b_1$$
$$B_2 = b_2 \oplus (0 + b_1) = b_2 \oplus b_1$$
$$B_3 = b_3 \oplus (0 + b_1 + b_2) = b_2 \oplus (b_1 + b_2)$$

Vậy tất cả các bít sau bit đầu tiên bằng 1 (tính từ bít trọng số nhỏ nhất - LSB) đều bị đảo trạng thái. Đây chính là số bù 2 của b.

Chúng ta có thể thiết kế mạch tạo số **bù 2** bằng cách dùng FF RS. Mạch này dùng thuận tiện khi cần thực hiện bài toán cộng và trừ nhiều bit nối tiếp.



Hình: Mạch tạo số bù 2 dùng FF RS.

Bắt đầu, Preset mạch để ngã ra Q = 1, các cổng G_2 mở, G_3 đóng 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 và $\bar{Q}$ =1, G_3 mở, G_2 đóng, số B đi qua cổng G_2 và bị đảo. Ở ngã ra được số bù 2 của B.

II. CÁC PHÉP TOÁN NHỊ PHÂN TRÊN SỐ BÙ 1

1. Trường hợp $N_1 < N_2$

Cho số 2 số dương N_1 và N_2 có n bit (nếu số bit khác nhau ta phải thêm 0 vào, mà không làm thay đổi giá trị, để cả hai có cùng số bit).

Ta tính:

$$\begin{aligned} N_1 - N_2 &= N_1 - N_2 + 2^n - 1 - 2^n + 1 \\ &= N_1 + (2^n - N_2 - 1) - 2^n + 1 \\ &= N_1 + (N_2)_1 - 2^n + 1 \\ &= -\{2^n - [N_1 + (N_2)_1] - 1\} \\ &= -[N_1 + (N_2)_1]_1 \end{aligned}$$

Vậy $N_1 - N_2$ có được bằng cách cộng số bù 1 của N_2 vào N_1 rồi lấy bù 1 của tổng và thêm dấu trừ. Như vậy, ta có thể thực hiện phép trừ chỉ cần dùng phép cộng và phép đảo.

Ví dụ: Tính 1001 – 11010 dùng số bù 1.

Ta có: $N_1 = 01001$ (thêm vào số 0 để có 5 bit như N_2).

$N_2 = 11010 \rightarrow (N_2)_1 = 00101$

$N_1 - N_2 = -[N_1 + (N_2)_1]_1 = -(01001 + 00101) = -(01110)_1 = -(10001)$

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.

+	N_1	01001
	N_2	00101
Số tràn→	0	01110

Không có số tràn là dấu hiệu của số âm, ta phải lấy bù 1 và thêm dấu trừ để đọc kết quả cuối cùng: $-(01110)_1 = -10001$.

2. Trường hợp $N_1 \geq N_2$

Kết quả $N_1 - N_2$ là số 0 hoặc số dương, phép tính được thực hiện theo qui tắc sau: Cộng N_1 với $(N_2)_1$ rồi cộng thêm 1 mà không quan tâm đến số nhớ.

Ví dụ 1: Tính 110101 – 100110.
$N_1 = 110101$ và $(N_2)_1 = 011001$.

+	N_1	110101
	$(N_2)_1$	011001
	1	001110
		+ 1
Số tràn→	1	001111

Bỏ qua số nhớ cuối cùng ta được kết quả $N_1 - N_2 = 001111$.
Trong hệ thập phân đây là bài toán: $53_{10} - 38_{10} = 15_{10}$.
Trong phép tính trên có số tràn chứng tỏ kết quả là số dương. Số 1 cộng thêm vào xem như lấy từ số nhớ đem qua.

Ví dụ 2: Tính 10110 – 10110.
$N_1 = 10110$ và $(N_2)_1 = 01001$.

+	N_1	10110
	$(N_2)_1$	01001
		11111
		+ 1
Số tràn→	1	00000

Trong phép cộng đầu tiên, không có số tràn, kết quả xem như số âm của số bù và khi cộng thêm 1 thì xuất hiện số tràn mà ta đã bỏ qua. Vậy $N_1 - N_2 = 00000$.

III. CÁC PHÉP TOÁN NHỊ PHÂN TRÊN SỐ BÙ 2

1. Trường hợp $N_1 < N_2$

Các toán dùng số bù 1 bất tiện vì ta phải cộng 1 vào, để tránh việc này, ta dùng phép toán dùng số bù 2.
Tương tự, cho 2 số nhị phân dương N_1 và N_2 có n bit.
Ta tính:

$$\begin{aligned} N_1 - N_2 &= N_1 - N_2 + 2^n - 2^n \\ &= N_1 + (2^n - N_2) - 2^n \\ &= N_1 + (N_2)_2 - 2^n \\ &= -\{2^n - [N_1 + (N_2)_2]\} \\ &= -[N_1 + (N_2)_2]_2 \end{aligned}$$

Vậy $N_1 - N_2$ có được bằng cách cộng số bù 2 của N_2 vào N_1 rồi lấy bù 2 của tổng và thêm dấu trừ. Như vậy, ta đã chuyển phép tính trừ thành phép tính cộng.

Ví dụ: Tính 1001 – 11010 dùng số bù 2.
$N_1 = 01001$ và $(N_2)_2 = 00110$.

Vậy $N_1 - N_2 = -[N_1 + (N_2)_2]_2 = -[01001+00110]_2 = -(01111)_2 = -(10001)$.

Tương tự như trên, để thấy trừ được nhận ra như thế nào, ta viết lại phép toán.

+	N_1	01001
	$(N_2)_2$	00110
Số tràn→0		01111

Không có số tràn là dấu hiệu của số âm. Ta phải lấy bù 2 và thêm dấu trừ để có kết quả cuối cùng.

2. Trường hợp $N_1 \geq N_2$

Kết quả $N_1 - N_2$ là số 0 hoặc số dương, phép tính được thực hiện theo qui tắc sau: Cộng N_1 với $(N_2)_2$ mà không quan tâm đến số nhớ ở vị trí 2^n.

Ví dụ 1: Tính 110101 – 100110.

$N_1 = 110101$ và $(N_2)_2 = 011010$.

+	N_1	110101
	$(N_2)_2$	011010
Số tràn→1		001111

Có số tràn, đây là kết quả số dương. Bỏ qua số nhớ cuối cùng, không cần biến đổi ta được kết quả $N_1 - N_2$ =001111.

Trong hệ thập phân đây là bài toán: $53_{10} - 38_{10} = 15_{10}$.

Ví dụ 2: Tính 10110 – 10110.

$N_1 = 10110$ và $(N_2)_1 = 01010$.

+	N_1	10110
	$(N_2)_2$	01010
Số tràn→1		00000

Bỏ qua số tràn, ta được $N_1 - N_2 = 00000$.

IV.CÁC PHÉP TOÁN DÙNG SỐ BÙ 2 KỂ CẢ BIT 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, đều này có thể khắc phục được bằng cách dùng số có dấu.

Với qui ước chung là: Số dương bit dấu là 0, số âm bit dấu là 1.

Ví dụ 1: Ta lấy một số số âm và dương đối nhau như dưới đây (lưu ý là hai số đối nhau cộng lại phải bằng 0).

+10 = 01010 +15 = 01111 +23 = 010111

–10 = 10110 –15 = 10001 –23 = 101001

Có thể thấy rằng, số âm của một số là bù 2 của nó kể cả bit dấu.

Với cách biểu diễn số có dấu, phép toán trừ trở thành phép toán cộng.

$$\mathbf{N_1 - N_2 = N_1 + (-N_2)}$$

Ví dụ 2: Tính $N_1 - N_2 = 01110 - 01001$.

$N_2 = 01001 = +9_{10}$ ➔ $-9_{10} = 10111$.

```
C2↓   C1↓
 1      1  11      ← Số nhớ
      + 0 1110
        1 0111
 ---------------
 1      0 0101
C'2↑  Dấu↑
```

Bit dấu bằng 0 chỉ kết quả là số dương, bỏ số tràn C'_2. Vậy $N_1 - N_2 = 00101$. Trong thập phân đây là bài toán $[14 + (-9)] = 5$.

Nếu N_1, N_2 đều dương hoặc âm, kết quả có thể cần thêm 1 bit do tràn số. Trong trường hợp này bit tràn đầu tiên thuộc kết quả C'_2 là bit dấu.

Ví dụ 3: Tính $N_1 + N_2 = 01110 + 01001$ (Bài toán: $14_{10} + 9_{10}$).
Kết quả là: 010111 (23_{10}). Với $C'_2 = 0$ là bit dấu.

```
C2↓  C1↓
 0     1        ← Số nhớ
     + 0 1110
       0 1001
 --------------
 0     1 0111
Dấu = C'2↑
```

Ví dụ 4: Tính $N_1 - N_2 = 10010 - 01001$ (Bài toán: $-14_{10} - 9_{10}$).
Tương tự như trên: $(N_2)_2 = 10111$.

```
C2↓  C1↓
 1     0  11     ← Số nhớ
     + 1 0010
       1 0111
 --------------
 1     0 1001
Dấu = C'2↑
```

Một lần nữa C'_2 chỉ bit dấu. Kết quả là: 101001 ⇔-23_{10} (010111 ⇔$+23_{10}$).
Từ các kết quả trên, ta rut ra qui tắc sau:
Nếu $C_1 = C_2$ thì C'_2 là bit tràn bỏ đi. Nếu $C_1 \neq C_2$ thì C'_2 là bit dấu.

Ví dụ 5: Tính $N_1 - N_2 = 011101 - 0110$ (Bài toán: $29_{10} - 6_{10}$).
Tương tự như trên, N_2 phải có số bit bằng N_1: $N_2 = 000110$ ➔ $(N_2)_2 = 111010$.

```
C2↓   C1↓
 1      1 1       ← Số nhớ
      + 0 11101
        1 11010
 ---------------
 1      0 10111
C'2↑  Dấu↑
```

Trường hợp này $C_1 = C_2$ nên C'_2 là bit tràn, ta bỏ đi.

Ghi chú:

- Trong tất cả trường hợp, ta luôn thực hiện phép cộng do đó có thể bỏ qua phép trừ.
- Khi hai số hạng cùng dấu thì có thể xảy ra hiện tượng tràn, lúc đó bít dấu dời về bên trái 1 bit. Trong các trường hợp khác thì dấu của kết quả cùng vị trí với dấu của số hạng.
- Ngoài ra, kết quả còn được xử lý bằng cách so sánh hai số nhớ C_1 và C_2 như nói trên.

V. MẠCH CỘNG NHỊ PHÂN

1. Mạch cộng nhị phân bán phần (Half adder, HA)

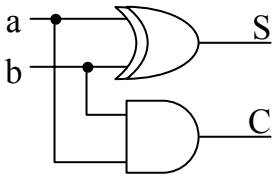
Là mạch cộng 2 số 1 bit.

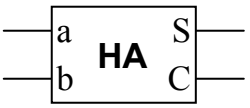
Vào		Ra	
a	b	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

Bảng sự thật.

$S = a \oplus b$
$C = a.b$

Kết quả





2. Mạch cộng nhị phân toàn phần (Full adder, FA)

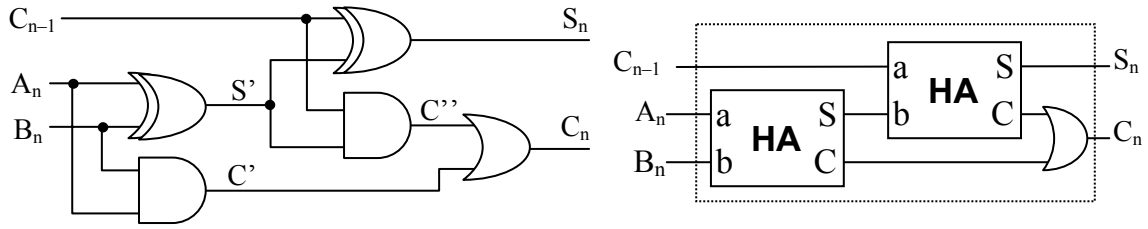
Là mạch cộng 2 số 1 bit ở cùng vị trí trong 2 số nhị phân nhiều bit, nói cách khác, đây là mạch cộng 2 bit (giả sữ thứ n) và bít nhớ từ phép cộng 2 bit thứ n–1 của 2 số nhị phân đó. Ta có bảng sự thật như dưới đây.

C_{n-1}	B_n	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 đồ Karnaugh ta xác định S_n và C_n, ta được:

$$S_n = C_{n-1} \oplus (A_n \oplus B_n)$$

$$C_n = A_nB_n + C_{n-1}(A_n \oplus B_n)$$



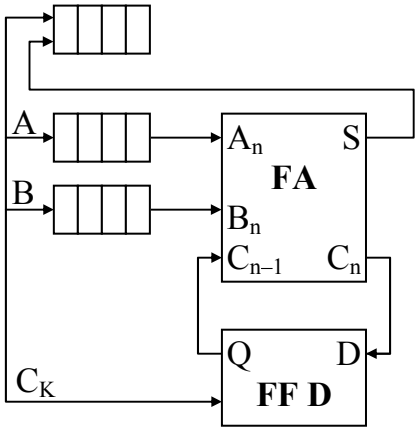
Hình: Sơ đồ mạch và ký hiệu của mạch cộng toàn phần.

Có thể thấy, mạch cộng toàn phần gồm 2 mạch cộng bán phần và một cổng OR.

VI.CỘNG HAI SỐ NHỊ PHÂN NHIỀU BIT

1. Cộng nối tiếp

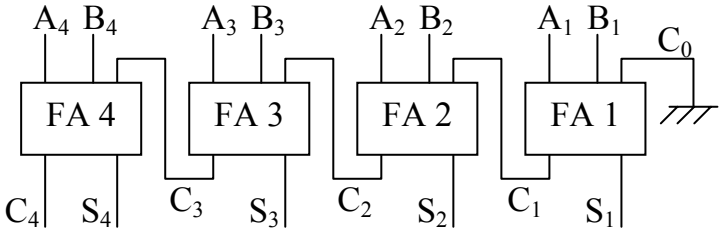
Trong cách cộng nối tiếp, người ta dùng các ghi dịch để chuyển các bit vào một mạch cộng duy nhất, số nhớ từ ngã ra C_n được làm trễ 1 bit nhờ FF D và đưa vào ngã vào C_{n-1}. Vậy tốc độ của phép cộng tùy thuộc vào các xung C_K và số bit phải thực hiện.



Hình: Sơ đồ mạch cộng nối tiếp.

2. Cộng song song

Trong cách cộng song song, mỗi mạch cộng toàn phần dùng cho 1 bit, số nhớ của bit trước sẽ được mang qua bit sau, chính vì lý do này mà tốc độ cộng còn hạn chế. Muốn nâng tốc độ cộng, người ta thực hiện phép cộng song song định trước số nhớ.



Hình: Sơ đồ mạch cộng song song.

3. Mạch cộng song song định trước số nhớ

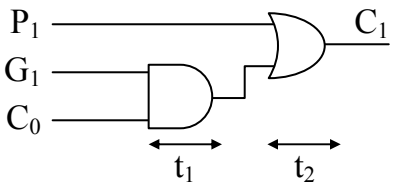
Để tăng tốc độ của mạch cộng song song, người ta tạo trước các số nhớ để đưa đồng thời vào mạch cộng.

Từ biểu thức xác định số nhớ: $C_n = A_n.B_n + C_{n-1}(A_n \oplus B_n)$

Ta đặt: $P_n = A_n.B_n$ và $G_n = A_n \oplus B_n$

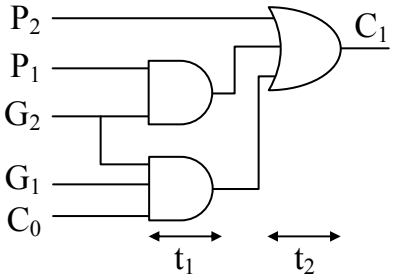
Ta xác định C_1, C_2, C_3,… như sau:

$C_1 = P_1 + C_0G_1$



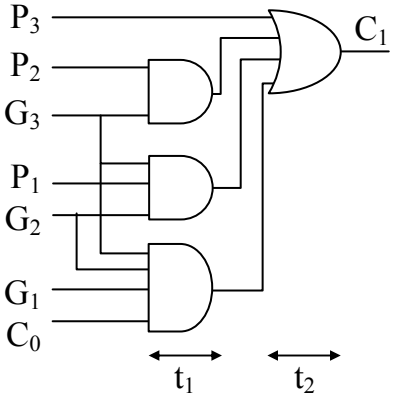
$C_2 = P_2 + C_1G_2$

$C_2 = P_2 + P_1G_2 + C_0G_1G_2$

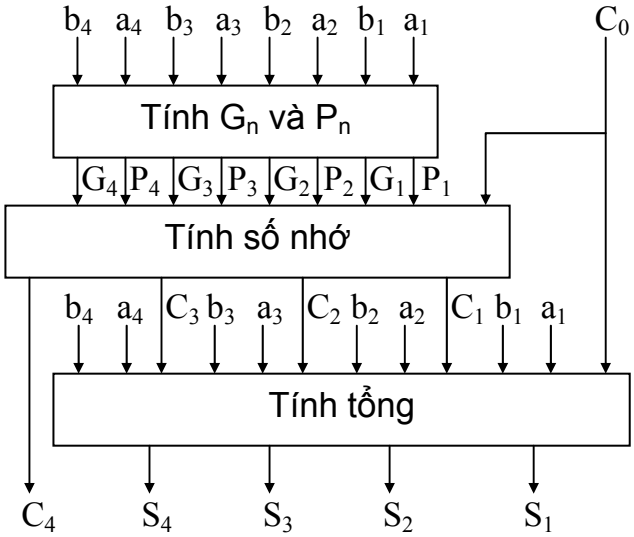


$C_3 = P_3 + C_2G_3$

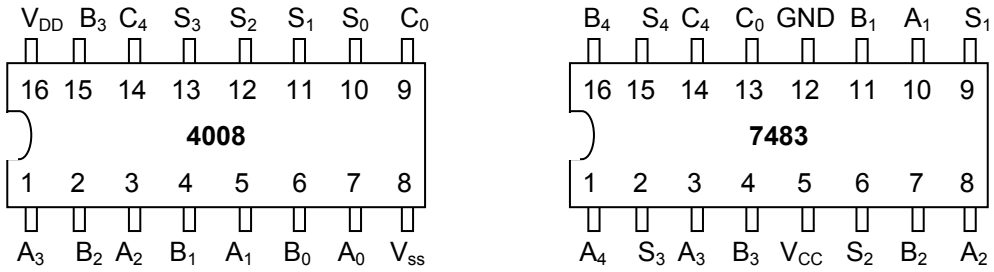
$C_3 = P_3 + P_2G_3 + P_1G_2G_3 + C_0G_1G_2G_3$



Ta nhận thấy thời gian tính số nhớ bằng nhau ở tất cả các tầng và bằng $t_1 + t_2$ là thời gian truyền qua hai cổng AND và OR. Dưới đây là sơ đồ mạch cộng song song định trước số nhớ.



Trên thị trường hiện có IC 7483 (tương đương với 4008 của CMOS) là IC cộng 4 bit theo kiểu định trước số nhớ.



Hình: Sơ đồ chân IC 4008 và IC 7483.

4. Mạch cộng hai số BCD

Dùng IC 7483 (hoặc 4008) để cộng 2 số BCD.

Hai số BCD có trị từ 0 đến 9 khi cộng lại cho kết quả từ 0 đến 18. Để đọc được kết quả dưới dạng BCD, ta phải hiệu chỉnh kết quả có được từ mạch cộng nhị phân.

Dưới đây là bảng tương đương của 3 mã: thập phân, nhị phân, BCD.

TP	Số nhị phân					Số BCD					BCD đọc theo NP
	S'=C'4	S'4	S'3	S'2	S'1	S=C4	S4	S3	S2	S1	
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

Ta nhận thấy:

- Khi kết quả <10 mã nhị phân và BCD trùng nhau.
- Khi kết quả ≥ 10, để có được mã BCD ta phải cộng thêm 6 cho mã nhị phân.

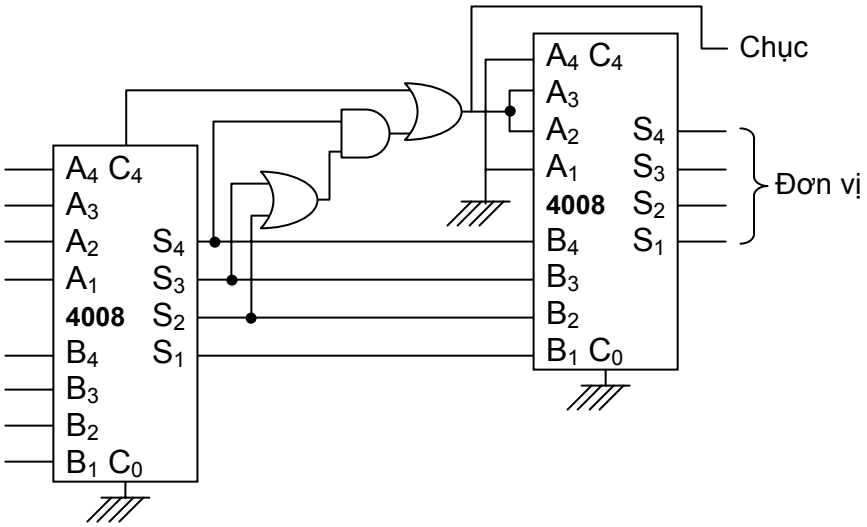
Như vậy, ta sẽ thực hiện một mạch có ngã ra Y = 1 khi phát hiện kết quả phép cộng ≥ 10.

TP	C'_4	S'_4	S'_3	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 ứng với từng cặp trị số $C'_4S'_4S'_3S'_2$ giống nhau thì $S'_1 = 0$ và $S'_1 = 1$ (Y không phụ thuộc vào S'_1).

Dùng bảng đồ Karnaugh xác định Y ta được như sau:

$$Y = C'_4 + S'_4(S'_3 + S'_2)$$



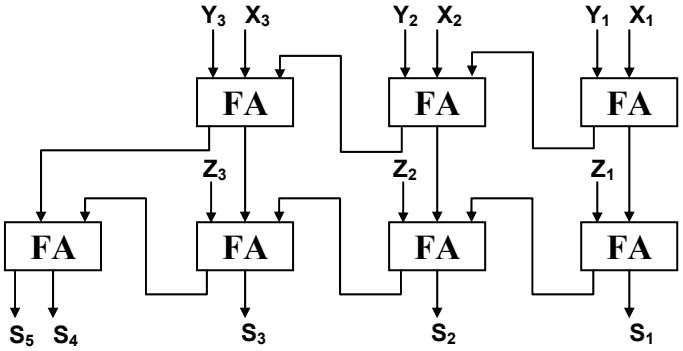
Hình: Sơ đồ mạch cộng hai số BCD dùng IC 4008.

5. Mạch cộng lưu số nhớ

Nhắc lại mạch cộng toàn phần FA, nhận 3 bit ở ngã vào và 2 bit ở ngã ra.

Để cộng một chuỗi số, nhiều mạch cộng toàn phần được sử dụng, số nhớ được lưu lại để đưa vào mạch cộng bit cao hơn.

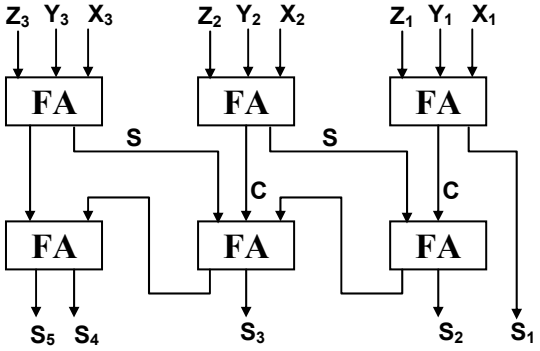
Ví dụ: Tính X + Y + Z, với X, Y, Z là số nhị phân 3 bit.



Hình: Mạch cộng 3 số 3 bit dùng FA.

Người ta dùng mạch cộng này để thực hiện một bài toán nhân.

Để có kết cộng quả nhanh hơn ta có thể dùng mạch như hình dưới đây.



Hình: Mạch cộng 3 số 3 bit dùng FA (cải tiến).

VII. MẠCH TRỪ NHỊ PHÂN

1. Mạch trừ nhị phân bán phần

Là mạch trừ 2 số 1 bit.

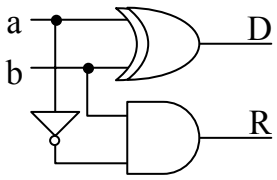
Vào		Ra	
A	b	D	R
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Bảng sự thật.

$D = a \oplus b$

$R = \bar{a}.b$

Kết quả



2. Mạch trừ nhị phân có số nhớ (mạch trừ toàn phần)

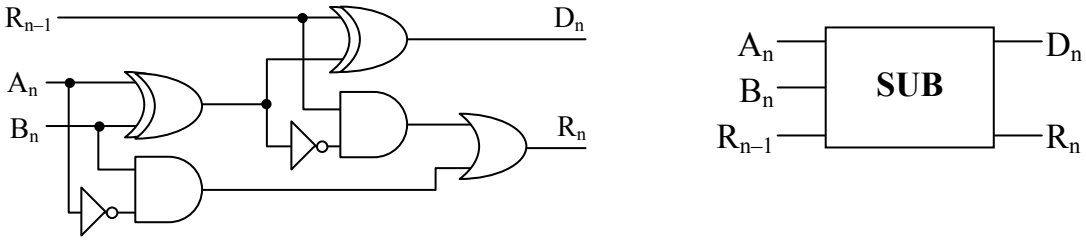
Là mạch trừ 2 bit có quan tâm đến số nhớ mang từ bit trước.

R_{n-1}	B_n	A_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

Dùng bảng đồ Karnaugh ta xác định D_n và R_n, ta được:

$$D_n = R_{n-1} \oplus (A_n \oplus B_n)$$

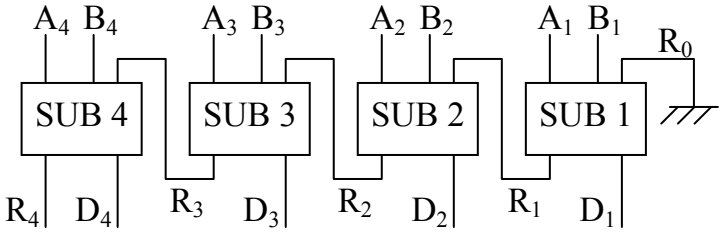
$$R_n = \overline{A}_n B_n + R_{n-1}(\overline{A_n \oplus B_n})$$



Hình: Sơ đồ và ký hiệu của mạch trừ toàn phần.

3. Trừ nhiều bit

Ta có mạch trừ nhiều bit bằng cách mắc song song các mạch trừ 1 bit.

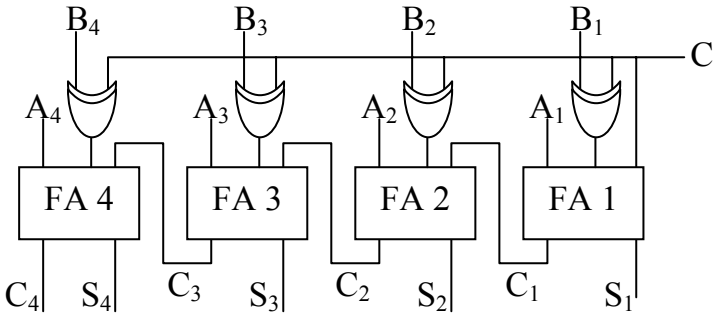


Hình: Sơ đồ mạch trừ nhiều bit.

4. Cộng và trừ nhiều bit trong một mạch

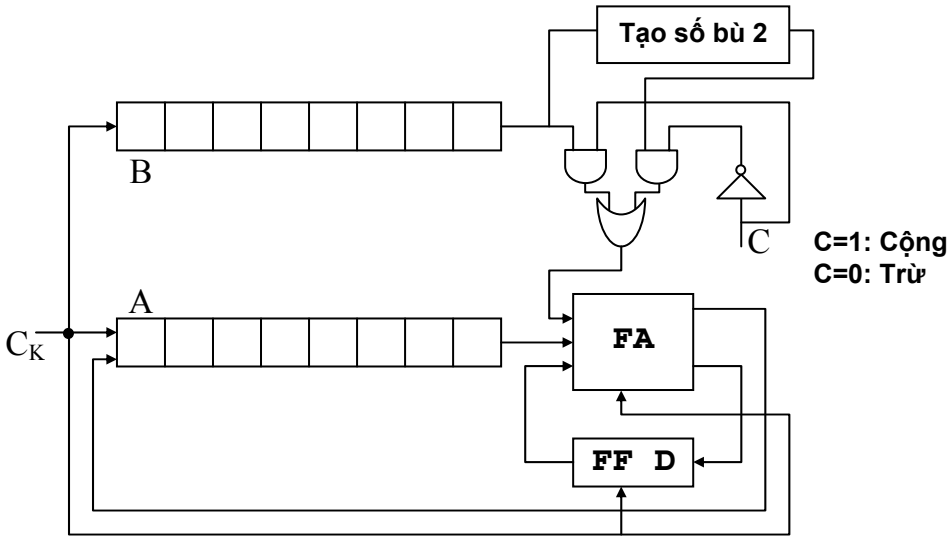
Để thực hiện phép toán trừ, người ta cộng với số bù 1 và cộng thêm 1 (cộng với số bù 2). Như vậy, để thực hiện phép tính A – B ta tính A + $(B)_1$ + 1. Mạch cộng toàn phần được sửa đổi để có thể thực hiện phép toán cộng và trừ tuỳ vào ngã vào điều khiển C.

- C = 0, ta có mạch cộng.
- C = 1, ta có mạch trừ.



Hình: Sơ đồ cộng và trừ chung một mạch.

Ta cũng có thể thực hiện mạch cộng trừ theo kiểu mắc nối tiếp.



Hình: Mạch cộng hoặc trừ nối tiếp (chung mạch).

Nếu A, B là số 8 bit, kết quả được xử lý bởi mạch dò số tràn, thiết kế dựa vào biểu thức: $\boldsymbol{OV = C_7 \oplus C_8}$ hoặc $OV = A_8 B_8 \overline{S}_8 + \overline{A}_8 \overline{B}_8 S_8$. Khi $\boldsymbol{OV = 1}$ nghĩa là có số tràn, thì C_8 là bit dấu, S_8 là 1 bit kết quả; khi $\boldsymbol{OV = 0}$ thì S_8 là bit dấu.

VIII. MẠCH NHÂN

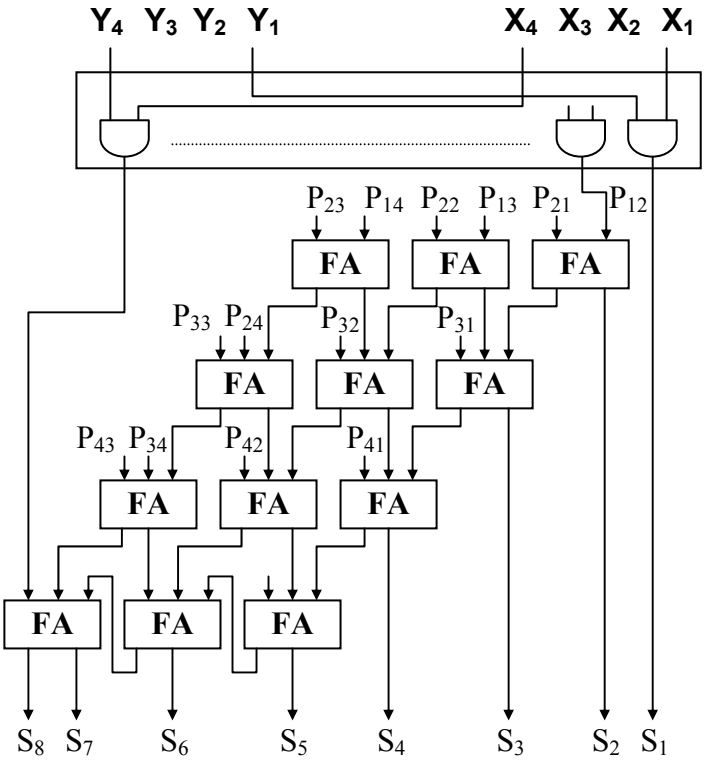
1. Mạch cơ bản

Lấy ví dụ nhân 2 số 4 bit như sau:

				Y_4	Y_3	Y_2	Y_1	Thừa số 1
				X_4	X_3	X_2	X_1	Thừa số 2
				P_{14}	P_{13}	P_{12}	P_{11}	
			P_{24}	P_{23}	P_{22}	P_{21}		Các tích
		P_{34}	P_{33}	P_{32}	P_{31}			từng phần
	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

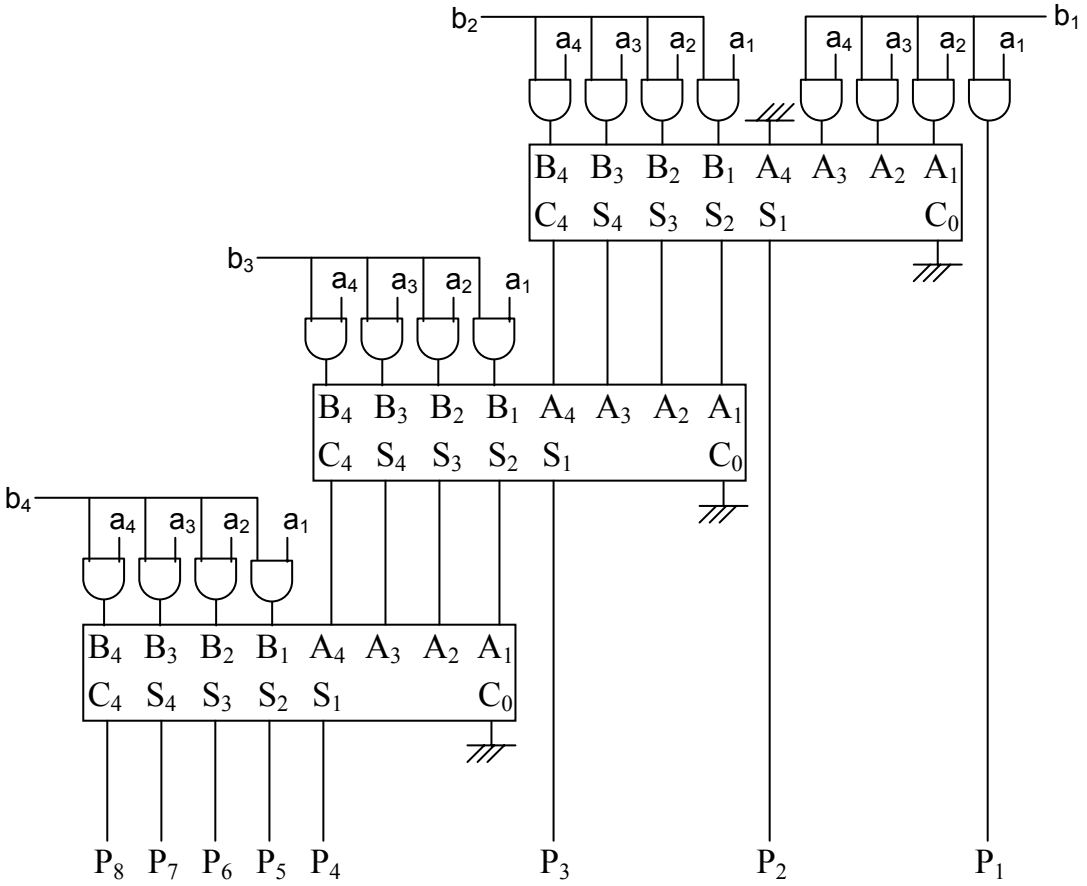
Việc thực hiện phép nhân có thể chia làm 2 bước.

- Tính các tích từng phần, được thực hiện bởi các cổng AND.
- Tính tổng các tích từng phần: Áp dụng bài toán tổng chuỗi số.



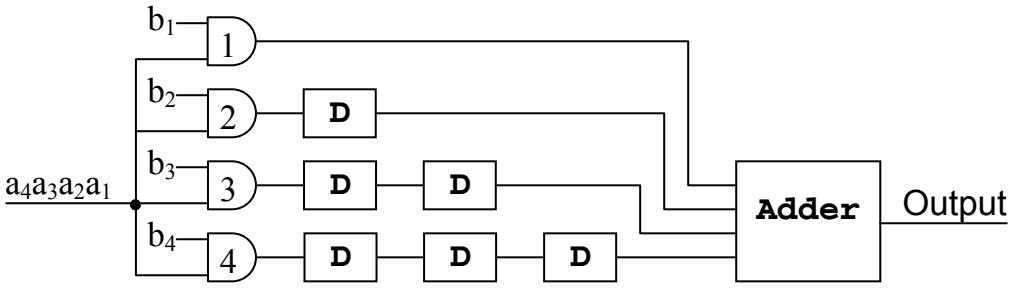
Hình: Sơ đồ mạch nhân số 4 bit.

Dùng IC cộng 4 bit (7483 hoặc 4008) mạch nhân 2 số 4 bit có dạng như dưới đây.



Hình: Sơ đồ mạch nhân dùng IC cộng 4 bit.

2. Mạch nhân nối tiếp – song song đơn giản



Hình: Mạch nhân nối tiếp – song song đơn giản.

Trong loại mạch này, 1 trong 2 thừa số được đưa nối tiếp vào mạch, thừa số còn lại được đưa song song vào mạch.

Thừa số $a_4a_3a_2a_1$ được đưa nối tiếp vào mạch, bắt đầu từ bit LSB (a_1). Các FF D có tác dụng dịch kết quả của phép nhân trước khi đưa vào mạch cộng để tính các tích từng phần này.

Thừa số $b_4b_3b_2b_1$ được đưa song song vào mạch như hình trên.

Ví dụ: Xét bài toán 10×14. Thừa số thứ nhất là 1010, thừa số thứ 2 là 1110. Quá trình nhân được giải thích như sau:

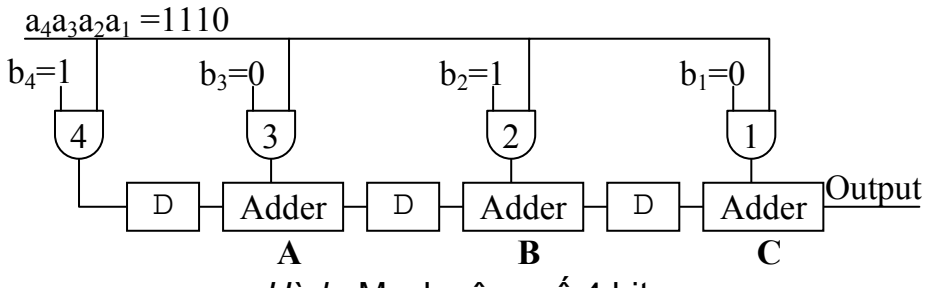
	P_8	P_7	P_6	P_5	P_4	P_3	P_2	P_1
A	0	0	0	0	0	0	0	0
B	0	0	0	1	1	1	0	0
C	0	0	0	0	0	0	0	0
D	0	1	1	1	0	0	0	0
Output	1	0	0	0	1	1	0	0

Ta thấy rằng: $10001100_2 = 140_{10}$.

Ngã ra A luôn bằng 0 vì LSB của số nhân bằng 0. Ngã ra B có giá trị nhân được làm trễ 1 bit (1 xung đồng hồ). Ngã ra C được làm trễ 2 bit (luôn bằng 0 – giống A). Ngã ra D được làm trễ 3 bit. Điều này có thể so sánh với bài toán trên giấy như dưới đây.

				1	1	1	0	Thừa số 1
				1	0	1	0	Thừa số 2
				0	0	0	0	
			1	1	1	0		
		0	0	0	0			
	1	1	1	0				
1	0	0	0	1	1	0	0	Tích

Muốn không sử dụng mạch cộng số nhiều bit, người ta sử dụng mạch theo sơ đồ sau:



Hình: Mạch cộng số 4 bit.

Mạch trên cần (n–1) mạch cộng và mạch trễ cho số nhân n bit. Các cổng AND cho phép các bit của thừa số thứ nhất đi qua khi bit của thừa số 2 là 1. Thừa số thứ nhất với số bit bất kỳ cho vào mạch nối tiếp với bit LSB đầu tiên.

Ngã ra cổng AND thứ tư sau 4 xung Clock có giá trị là 1110. Ngã ra cổng 3 là 0000 (do $b_3 = 0$).

Vậy Adder A sẽ tính giá trị ngã ra cổng AND 4 bị trễ 1 bit và ngã ra cổng AND 3 là: 11100 + 0000.

	0	0	0	0
1	1	1	0	0
1	1	1	0	0

Tương tự, mạch cộng B cộng số nhị phân ở ngỏ ra cổng AND 2 với kết quả A được làm trễ 1 bit.

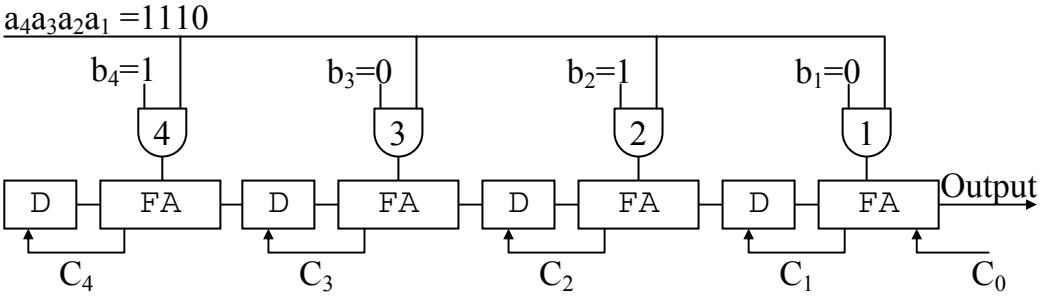
		1	1	1	0	
	1	1	1	0	0	
1	0	0	0	1	1	0

Và cộng tiếp với mạch cộng C, kết quả ở mạch cộng A trễ 1 bit. Ta được kết quả ở ngã ra của mạch cộng C.

			0	0	0	0	0
1	0	0	0	1	1	0	0
1	0	0	0	1	1	0	0

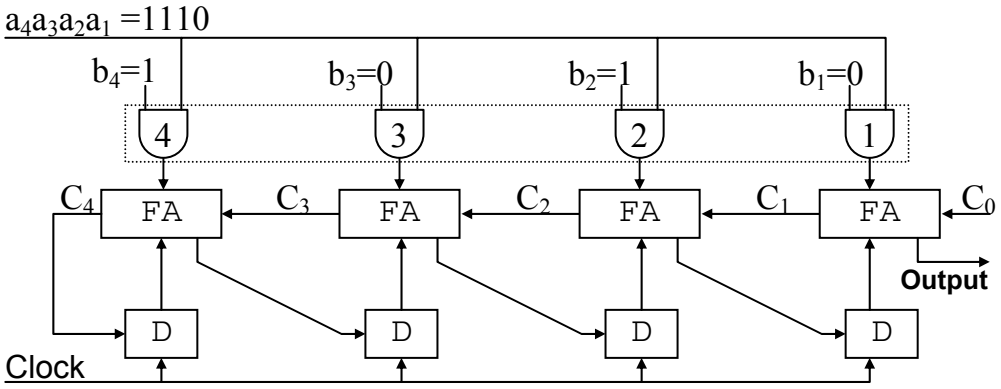
Lưu ý: **Các mạch trên là mạch cộng chưa quan tâm đến số nhớ.**

Để thiết kế mạch cho kết quả với số nhớ ta thiết kế như dưới đây.



Hình: Mạch nhân số 4 bit có quan tâm số nhớ.

Để thiết kế mạch thực tế dùng ghi dịch 4 bit có ngã vào/ra song song và một chíp 4 cổng AND (IC 4 cổng AND) để thực hiện bài toán nhân, có sơ đồ như dưới đây.



Hình: Mạch nhân 4 bit dùng tổ hợp các IC.

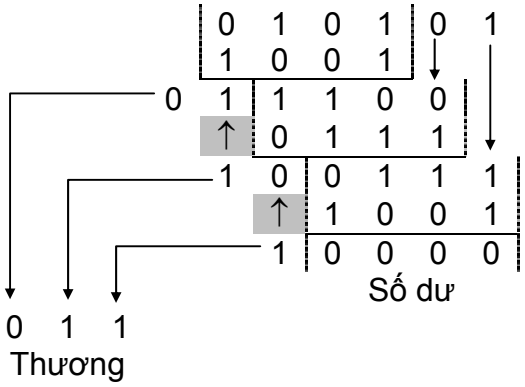
IX.MẠCH CHIA

Tương tự như mạch nhân, mạch chia có thể xem như một chuỗi mạch trừ liên tiếp của số bị chia và số chia. Tuy nhiên, tuỳ theo kết quả của mạch trừ dương hay âm mà có cách xử lý khác nhau. Dưới đây là qui tắc:

- Số chia (SC) lớn hơn số bị chia (SBC) (SBC–SC<0), thương số là 0, dịch phải số chia 1 bit, thực hiện bài toán cộng số chia và số bị chia.
- SC<SBC (SC–SBC>0), thương số là 1, dịch phải số chia 1 bit, thực hiện bài toán trừ số chia và số bị chia (cộng với số bù 2).

Để đơn giản, ta xét số bị chia và số chia đều dương (MSB = 0), số bị chia gồm 6 bit và số chia gồm 4 bit.

Ví dụ 1: Thực hiện bài toán $21_{10} = 010101_2$ chia $7_{10} = 0111_2$.
Số bù 2 của $7_{10} = 0111_2$ là $(0111)_2 = 1001$.



Dấu ↑ trong ô chỉ lên số 1, tức là kết quả phép trừ là số âm, bước kế tiếp là dời và cộng với số chia – cộng với 0111.

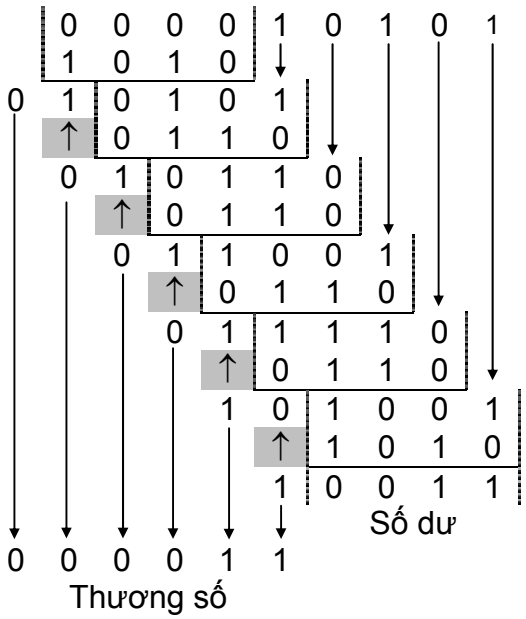
Dấu ↑ trong ô chỉ lên số 0, tức là kết quả phép trừ là số dương, bước kế tiếp là dời và trừ với số chia (cộng với số bù 2) – cộng với 1001.

Thương số là số có được từ các số tràn, như hình trên.

Kết quả thương là 011 (3) và số dư là 0000 (0).

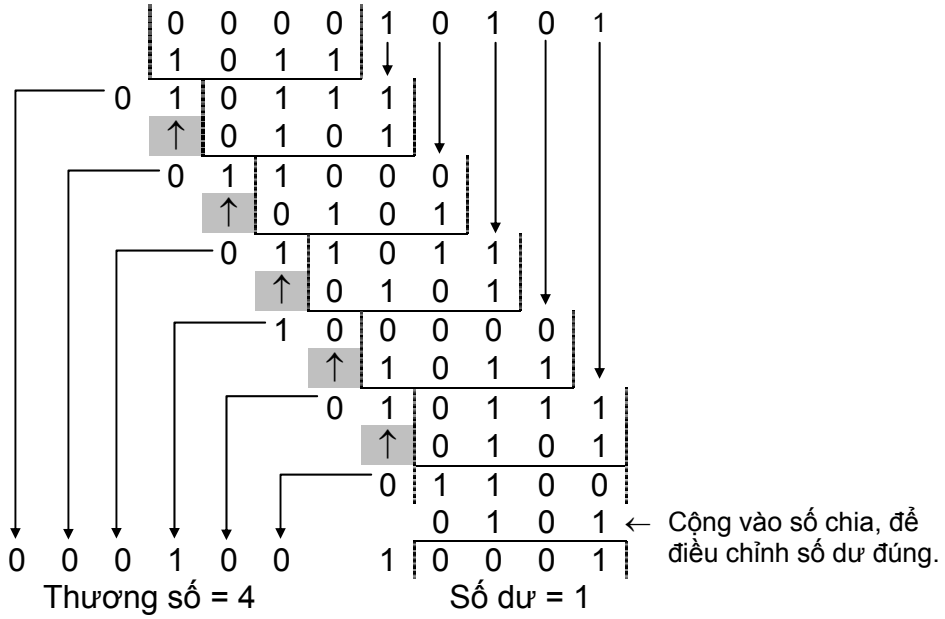
Bài toán trên cho kết quả với 3 mức cộng/trừ. Tuy nhiên, nếu chia 21 cho 1 ta cần 6 phép cộng trừ để có thương số 6 bit. Một cách tổng quát, số mức của bài toán bằng với số bit của số bị chia (ta có thể thêm vào phép chia trên 3 số 0 trước số bị chia để có được thương số là 6 bit).

Ví dụ 2: Thực hiện bài toán $21_{10} = 010101_2$ chia $6_{10} = 0110_2$.
Số bù 2 của $6_{10} = 0111_2$ là $(0111)_2 = 1010$.

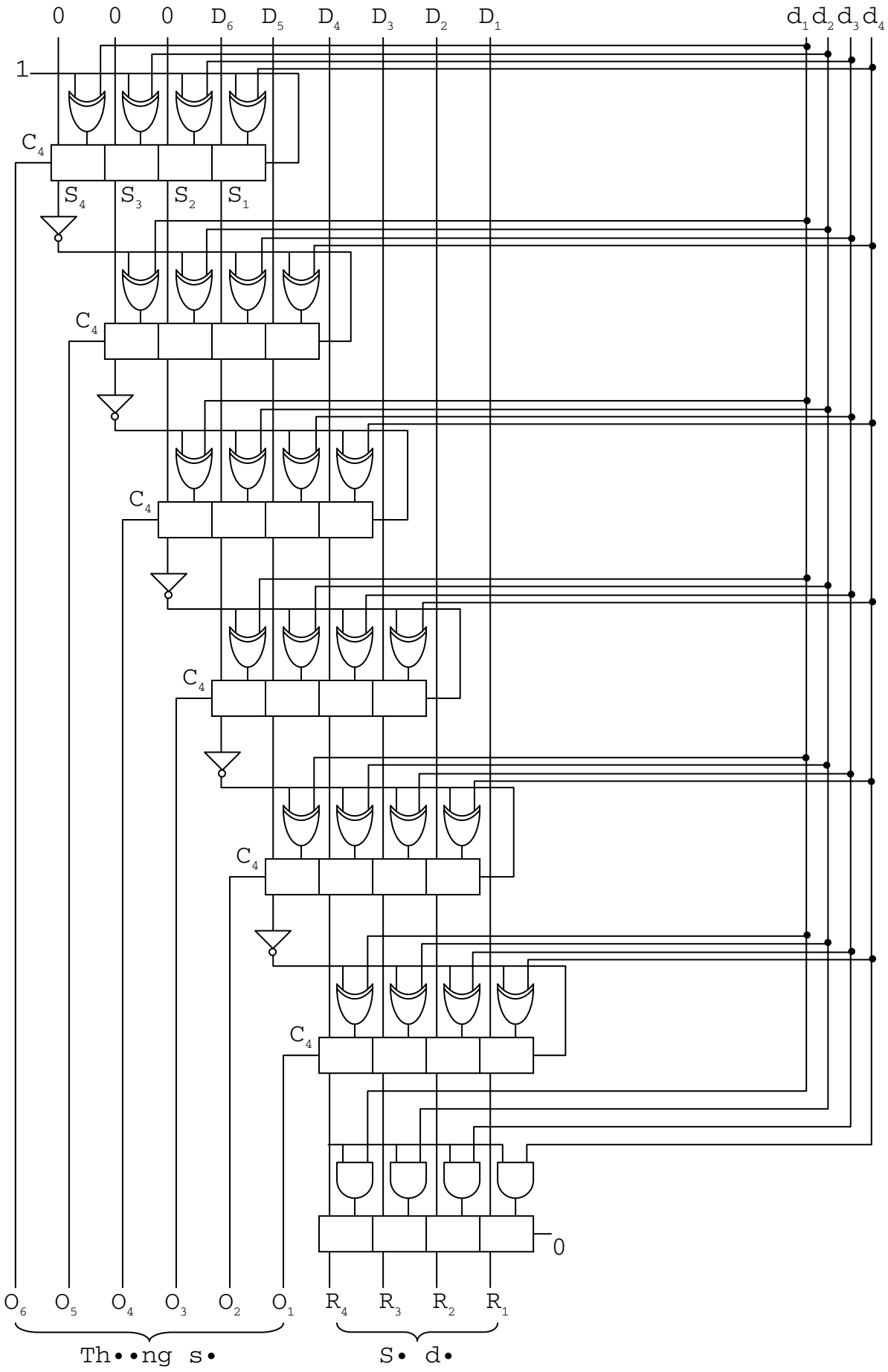


Ví dụ 3: Thực hiện bài toán $21_{10} = 010101_2$ chia $5_{10} = 0110_2$, được kết quả là 4 dư 1. Tuy nhiên, trên phép toán ta thấy phép cộng với phép chia cuối cùng cho kết quả âm (1100), để điều chỉnh ta phải cộng vào số chia và bỏ qua số tràn.

Số bù 2 của $5_{10} = 0101_2$ là $(0111)_2 = 1011$.



Trong mạch chia hình dưới đây, bước đầu tiên được thực hiện bởi các cổng EXOR trên cùng, có ngã vào điều khiển là 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. Mạch cộng cuối cùng được thiết kế kết hợp các cổng AND để xử lý kết quả của số dư như trong hai ví dụ 2 và 3. Nếu kết quả của bài toán ở bước 6 có S_4=1 thì cổng AND được mở để thực hiện bài toán cộng với số chia để điều chỉnh số dư.



CHƯƠNG 7: BỘ NHỚ BÁN DẪN

- ✓ THUẬT NGỮ
- ✓ VẬN HÀNH TỔNG QUÁT
- ✓ GIAO TIẾP VỚI CPU
- ✓ CÁC LOẠI BỘ NHỚ BÁN DẪN
  - ROM
  - PLD
  - RAM
- ✓ MỞ RỘNG BỘ NHỚ BÁN DẪN
  - Mở rộng độ dài từ
  - Mở rộng vị trí nhớ
  - Mở rộng dung lượng nhớ

I. GIỚI THIỆU

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 dài hay ngắn. Khả năng nhớ này là điều làm cho hệ thống số trở nên đa năng và có thể thích hợp với nhiều tình huống. Ví dụ, 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 quen thuộc với các FlipFlop, đó là một thiết bị nhớ điện tử. Chúng ta 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.

Dữ liệu cũng có thể được lưu trữ dưới dạng điện tích của tụ điện, và một loại phần tử nhớ rất quan trọng đã dùng nguyên tắc này để lưu trử dữ liệu với mật độ cao nhưng tiêu thụ 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 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 của CPU.

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 thiết bị có tính chất lưu trữ khối (mass storage) – là loại 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 của 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 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.

Trong chương này, chúng ta nghiên cứu cấu tạo và tổ chức của các bộ nhớ bán dẫn.

II. 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.

Ngoài ra để thực hiện bài toán cộng nhiều số ta nên nhớ:

- ***Tế bào nhớ***: là linh kiện hay một mạch điện tử dùng để lưu trữ một bit đơn (0 hay 1). Ví dụ tế bào nhớ là một FF, tụ được tính 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 số nhị phân. Ví dụ một thanh ghi 8 FF là một phần tử nhớ lưu trữ từ 8 bit. Kích thước của từ nhớ trong máy tính hiện đại có độ 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. Ví dụ bộ nhớ có khả năng lưu trữ 4096 từ nhớ 20 bit, dung lượng của nó là 4096×20, mỗi 1024 ($1024 = 2^{10}$) từ nhớ được gọi là **1K**, như vậy 4096×20 = 4K×20. Với dung lượng lớn hơn ta dùng **1M** ($1M=2^{10}K$) để chỉ 1048576 từ nhớ…
- ***Địa chỉ***: là số nhị phân dùng xác định vị trí của từ nhớ trong bộ nhớ. Mỗi từ nhớ được lưu trữ trong bộ nhớ tại một địa chỉ duy nhất. Địa chỉ luôn luôn được biểu diễn bởi số nhị phân. Tuy nhiên để dễ hiểu người ta dùng số hex, số bát phân, số thập phân.
- ***Tác vụ đọc***: Read hay còn gọi là felch, 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 hay còn gọi là store, một từ mới được đặt vào một vị trí trong bộ nhớ, khi từ mới được viết thì từ cũ mất đi.
- ***Thời gian truy xuất (access time)***: số đo tốc độ hoạt động của bộ nhớ, ký hiệu là 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 bộ nhớ vĩnh cữu (nonvolatile).
- ***Bộ nhớ truy xuất ngẫu nhiên (Random Access Memory, RAM)***: đó là bộ nhớ mà vị trí của tế bào nhớ trong nó không ảnh hưởng đến thời gian đọc hay viết dữ liệu vào. Nói cách khác, thời gian truy xuất như nhau đối với mọi vị trí nhớ. Hầu hết các loại bộ nhớ bán dẫn là loại truy xuất ngẫu nhiên.
- ***Bộ nhớ truy xuất tuần tự (Sequential Access Memory, SAM)***: đó là bộ nhớ mà thời gian đọc hay viết dữ liệu ở các vị trí khác nhau thì khác nhau. Những ví dụ của loại bộ nhớ này là băng từ, đĩa từ, CD–ROM,… Tốc độ của các 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 chỉ được ghi một lần ở nơi sản xuất và sau đó thông tin chỉ có thể đọc ra từ bộ nhớ. Có loại nhớ ROM có thể được ghi nhiều lần nhưng tác vụ ghi khá phức tạp hơn tác

vụ đọc. ROM thuộc loại bộ nhớ vĩnh cữu và dữ liệu được lưu trữ 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 của bộ nhớ này thường chậm và nó thuộc loại vĩnh cữu.

III. ĐẠI CƯƠNG VỀ VẬN HÀNH CỦA BỘ NHỚ

Mặc dù mỗi loại bộ nhớ có hoạt động bên trong khác nhau, nhưng chúng có chung một nguyên tắc vận hành mà chúng ta có thể tìm hiểu sơ lượt trước khi đi vào nghiên cứu từng loại bộ nhớ.

Mỗi hệ thống nhớ luôn có một số yêu cầu ở các ngã vào/ra để hoàn thành một số tác vụ, đó là:

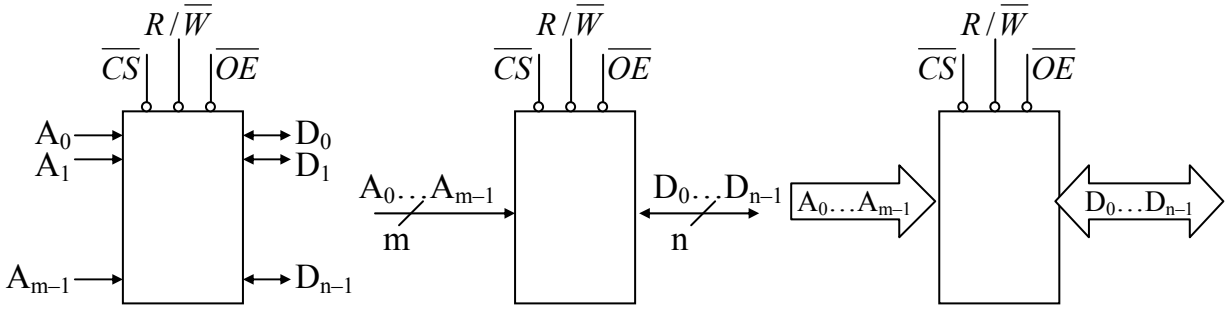
- Chọn địa chỉ trong bộ nhớ để thực hiện việc đọc ra hoặc viết vào.
- Chọn tác vụ đọc hoặc viết để thực hiện.
- Cung cấp dữ liệu vào để lưu trữ vào bộ nhớ trong trong tác vụ viết.
- Gửi dữ liệu ra từ bộ nhớ trong tác vụ đọc.
- Cho phép (Enable) hay không cho phép (Disable) bộ nhớ đáp ứng đối với lệnh 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, với nhiệm vụ tương ứng 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 từ A_0 đến A_{n-1}. Ví dụ, IC có 10 chân địa chỉ sẽ có 1K = 1024 (2^{10}) vị trí nhớ.
- ***Ngã vào/ra dữ liệu***: Các chân dữ liệu là các ngã vào ra, nghĩa là dữ liệu luôn được xử lý 2 chiều. Thường là dữ liệu vào ra chung một chân nên các ngã này thuộc loại ngã ra 3 trạng thái. Số chân địa chỉ và chân dữ liệu của IC xác định dung lượng nhớ của IC đó. Ví dụ, IC có 10 chân địa chỉ và 8 chân dữ liệu thì dung lượng nhớ của IC đó là 1K×8 = 8K bit = 1KB.
- ***Các ngã vào điều khiển***: Mỗi 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 như sau:
    - $\overline{CS}$: 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 khi chân này ở mức cao, ghi dữ liệu khi chân này ở mức thấp.

- $\overline{CAS} - \overline{RAS}$: Column Address Strobe – Row Address Strobe – Chốt địa chỉ cột – Chốt địa chỉ hàng. Chỉ những IC nhớ có địa chỉ hàng và cột mới có chân này.

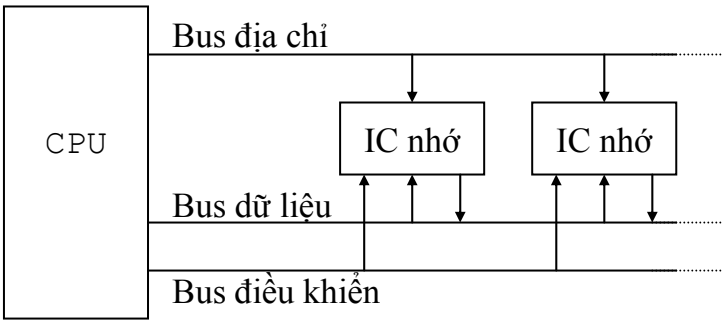
Hình dưới đây cho thấy cách vẽ nhóm chân các IC nhớ, *m* chân địa chỉ, *n* chân dữ liệu. Cách vẽ các chân địa chỉ và dữ liệu dưới dạng BUS.



Hình: *Cách vẽ nhóm chân IC nhớ.*

IV.GIAO TIẾP GIỮA IC NHỚ VÀ BỘ XỬ LÝ TRUNG TÂM

Trong mọi hoạt động có liên quan đến IC nhớ đều do bộ xử lý trung tâm (Central Proccessing Unit, CPU) quản lý. Giao tiếp giữa IC nhớ và CPU mô tả như hình dưới đây.



Hình: *Giao tiếp giữa IC nhớ và CPU.*

Một tác vụ 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 phục vụ.

Dĩ nhiên, các bước trên phải tuân thủ giản đồ thời gian của từng IC nhớ (sẽ đề cập đến khi xét các loại bộ nhớ).

V. CÁC LOẠI BỘ NHỚ BÁN DẪN

1. Giới thiệu

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 có lẽ vì lý do lịch sử ra đời của các chủng loại khác nhau. Để phân biệt chính xác ROM và RAM ta có thể gọi ROM là bộ nhớ chết (nonvolatile) và RAM là bộ nhớ sống (volatile) 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.

2. ROM

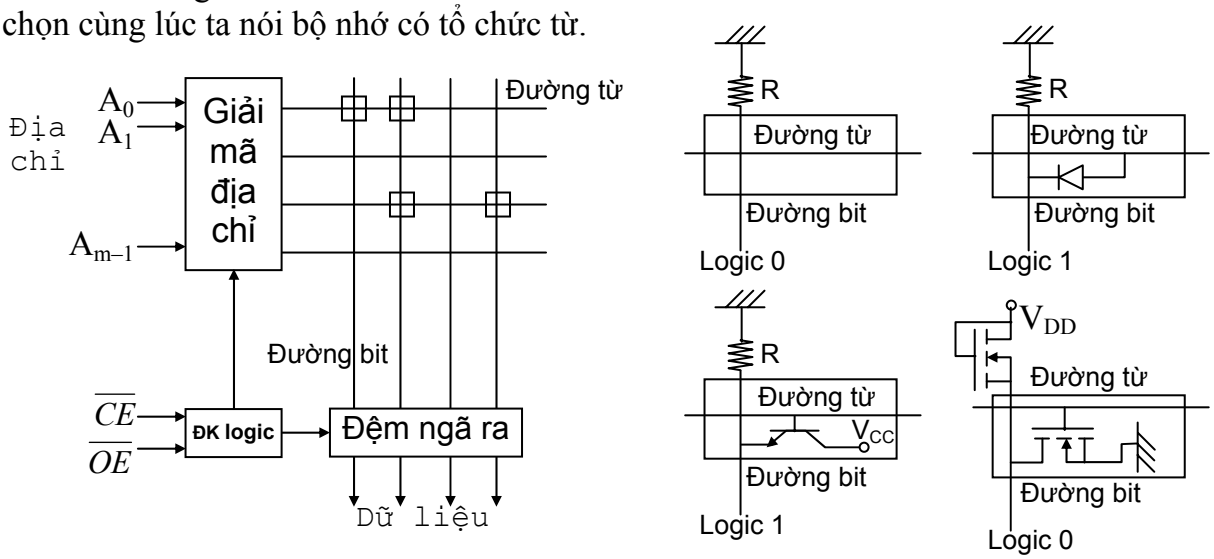
a. Giới thiệu

Các tế bào nhớ hoặc từ nhớ trong ROM được 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 của 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 ½ số đường địa chỉ của cả bộ nhớ.

b. 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 xác định trước bằng phương pháp mặt nạ: đưa vào các linh kiện điện tử nối từ **đường từ** qua **đường bit** để tạo ra một giá trị bit và để trống cho giá trị bit ngược lại.

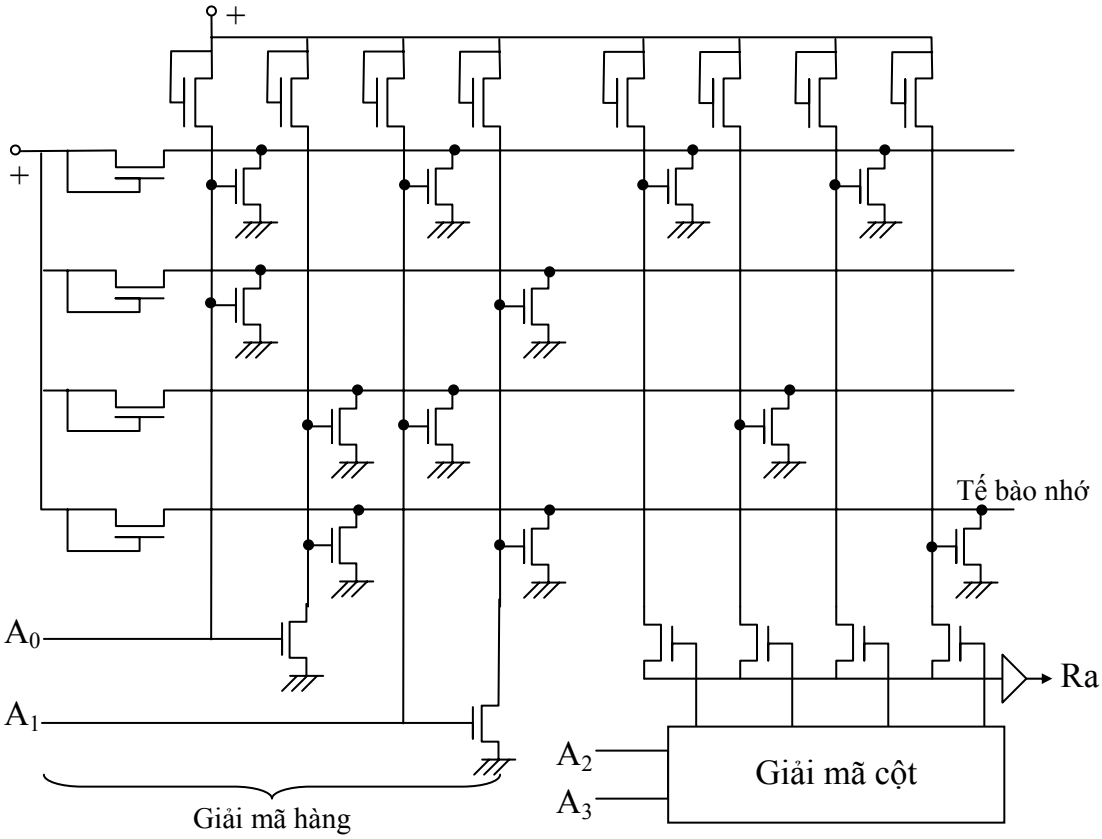
Dưới đây là mô hình của một MROM trong đó các ô vuông là nơi chứa (hay không chứa) một linh kiện (diod, transistor BJT, MOSFET) để tạo bit. Mỗi ngã ra của mạch giải mã địa chỉ gọi là đường từ và đường nối tế bào nhớ ra ngoài gọi là đường bit. Khi đường từ lên mức cao thì tế bào nhớ được chọn. Khi nhiều tế bào nhớ được chọn cùng lúc ta nói bộ nhớ có tổ chức từ.



Hình: *Mô hình của MROM.*

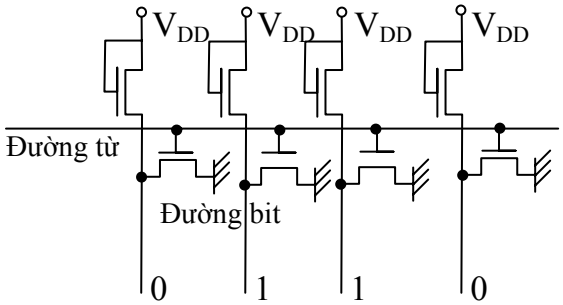
Nếu tế bào nhớ là Diod hay BJT thì sự hiện diện của linh kiện tương ứng với bit 1 còn vị trí trống tương ứng với bit 0. Đối với linh kiện MOSFET thì ngược lại (muốn có kết quả như BJT thì thêm ở ngã ra các cổng đệm đảo).

Dưới đây là ví dụ bộ nhớ MROM có dung lượng 16×1 với các mạch giải mã hàng và cột (các mạch giải mã 2 ra 4 của hàng và cột đều dùng Transistor MOS và có cùng cấu trúc).



Hình: *Ví dụ về bộ nhớ MROM.*

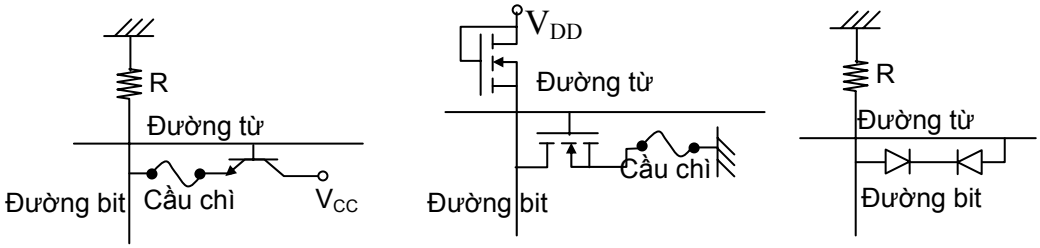
Trên thực tế để đơn giản cho việc thực hiện, ở mỗi vị trí nhớ, người ta đều cho vào một transistor MOS được chế tạo lớp SiO_2 dày hơn làm tăng hiệu thế ngưỡng của nó lên, kết quả là transistor MOS này luôn không dẫn điện, các transistor khác dẫn điện bình thường.



Hình: *Transistor MOS có SiO_2 dày hơn (cho mức 1).*

c. ROM lập trình được

Có cấu tạo giống như MROM nhưng ở mỗi vị trí đều có các linh kiên nối với các 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 tại 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.



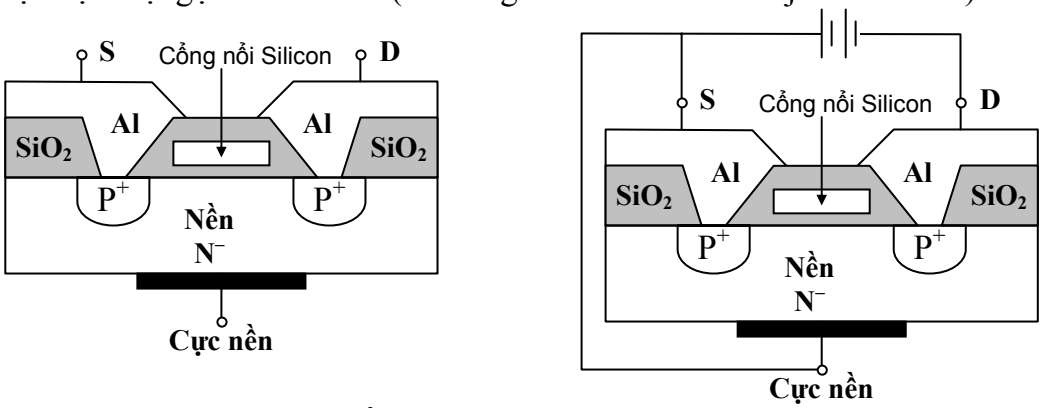
Hình: *Tổ chức bit của ROM lập trình được.*

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 được phá hỏng tạo mạch nối tắt, diod còn lại dẫn điện cho bit 1.

d. ROM lập trình được, xóa được bằng tia U.V.

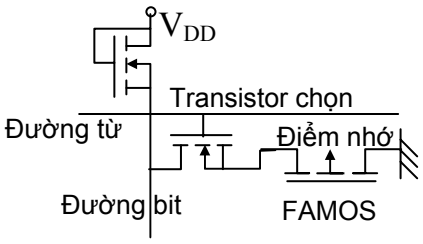
Ultra Violet Programmable ROM, U.V. EPROM.

Đây là loại ROM rất thuận tiện cho người sử dụng vì có thể dùng được nhiều lần bằng cách xóa và nạp lại. Cấu tạo của tế bào nhớ U.V. EPROM là một transistor MOS có cấu tạo đặc biệt gọi là FAMOS (Floating Gate Avalanche Injection MOS).



Hình: *Cấu tạo bên trong U.V. EPROM.*

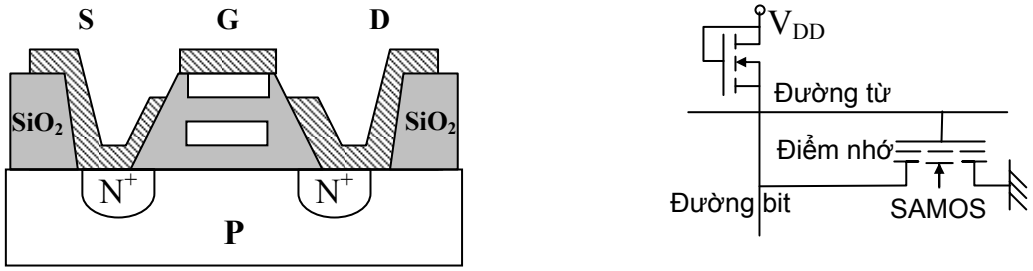
Trên nền chất bán dẫn N pha loãng, tạo 2 vùng P pha đậm (P^+) nối ra ngoài cho 2 cực S (Source) và D (Drain). Trong lớp cách điện SiO_2 giữa 2 cực người ta cho vào một thỏi Silicon không nối với bên ngoài và được gọi là **cổng nổi**. Khi nguồn V_{DD} phân cực ngược giữa cực nền và cực Drain còn nhỏ, 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 2 vùng bán dẫn P^+, transistor trở nên dẫn điện. Khi cắt nguồn, transistor vẫn tiếp tục dẫn điện vì electron không thể trở về tái hợp với lỗ trống. Để xóa EPROM, người ta chiếu tia U.V. vào các tế bào trong khoảng thời gian để electron trên cổng nổi nhận đủ năng lượng vượt qua lớp điện trở về vùng nền tái hợp với lỗ trống xóa điện lộ P và transistor trở về trạng thái không dẫn ban đầu.



Hình: *Cấu tạo tế bào nhớ.*

Mỗi tế bào nhớ của EPROM gồm transiator FAMOS nối tiếp với một transitor MOS khác mà ta gọi là transistor chọn.

Để loại bỏ transistor chọn, người ta dùng transistor SAMOS (Stacked Gate Avalanche Injection MOS có cấu tạo tương tự như 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 khó dẫn điện hơn. Như vậy, nếu ta chọn V_C ở khoảng giữa VT_1 và VT_2 ($VT_1 < V_C < VT_2$) thì các transistor không lập trình (không có lớp electron ở cổng nổi) sẽ dẫn còn các transistor được lập trình sẽ không dẫn.



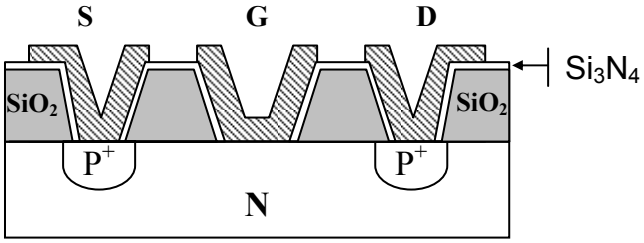
Hình: *Cấu tạo của tế bào nhớ SAMOS.*

Điểm bất lợi của U. V. EPROM là cần thiết bị xóa đặt biệt phát tia U. V. và mỗi lần xóa tất cả tế bào nhớ trong một IC đều bị xóa. Như vậy, người sử dụng phải nạp lại toàn bộ chương trình.

e. ROM lập trình được, xóa được bằng xung điện

Electrically Erasable PROM – EEPROM, Electrically Alterable PROM – EAPROM.

Đây là loại ROM lập trình được và xóa được nhờ xung điện và đặc biệt có thể xóa để sửa chữa trên từng byte. Các tế bào nhớ EEPROM sử dụng transistor MNOS (Metal Nitride Oxide Semiconductor), có cấu tạo như dưới đây.



Hình: *Cấu tạo của tế bào nhớ EEPROM.*

Giữa lớp kim loại nối ra các cực và lớp SiO_2 là một lớp mỏng chất Si_3N_4 – dày từ 40*nm* đến 650*nm*. Dữ liệu được nạp bằng cách áp một điện thế dương giữa cực G và S

(khoảng 20V 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 ngược chiều giữa 2 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 điện thế ngược chiều nhau.

f. 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 là loại nonvolatile, có tốc độ truy xuất nhanh, cho phép xóa và ghi lại ngay trong mạch trên từng byte, mật độ tích hợp thấp, giá thành cao hơn EPROM.

Bộ nhớ FLASH tận dụng được ưu điểm của 2 loại ROM nói trên, nghĩa là tốc độ truy xuất nhanh, mật độ tích hợp cao và giá thành thấp.

Hầu hết FLASH ROM sử dụng cách xóa đồng thời cả khối dữ liệu nhưng rất nhanh (vài trăm ms so với 20min của U.V. EPROM). Những FLASH ROM thế hệ mới cho phép xóa từng sector (512 bytes). FLASH ROM có thời gian ghi khoảng 10μs so với 100μs đối với EPROM và 5ms đối với EEPROM.

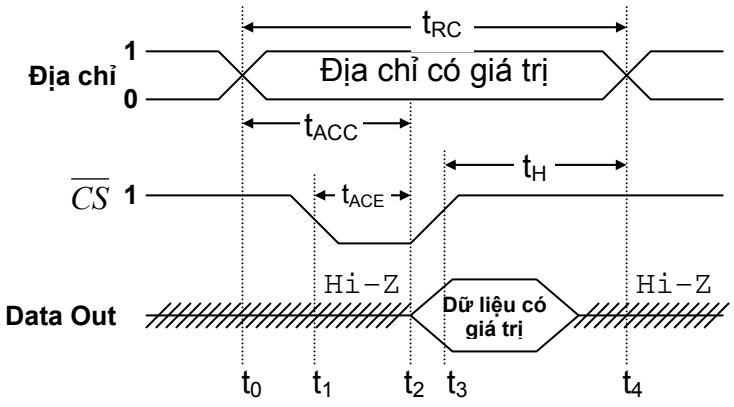
g. Giản đồ thời gian của ROM

Ngoại trừ MROM chỉ dùng chế độ đọc, các loại ROM khác đều sử dụng 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 chương trình.

Chu k• ••c c•a ROM

Các địa chỉ, các tính 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 có một số thời gian như sau:



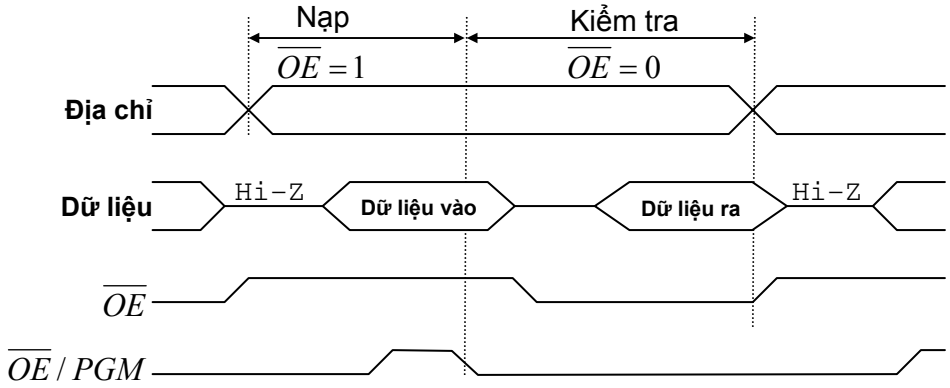
Hình: *Giản đồ thời gian cho một chu kỳ đọc của ROM.*

- t_{ACC} (Address access time): Thời gian truy xuất địa chỉ. Đây là 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 30ns đến 90ns, còn loại MOS khoảng 200ns đến 900ns.

- $\mathbf{t_{ACS}}$ ($\mathbf{t_{ACE}}$) (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 khoảng 100*ns*.
- $\mathbf{t_H}$ (Hold time): Thời gian dữ liệu còn tồn tại trên BUS dữ liệu kể từ khi tín hiệu $\overline{CS}$ hết hiệu lực.

Chu k• vi•t c•a ROM

Giản đồ thời gian cho một chu kỳ nạp dữ liệu cho EPROM gồm thời gian nạp (Programmed) và thời gian kiểm tra kết quả (Verify).



Hình: *Giản đồ thời gian cho một chu kỳ nạp dữ liệu của EPROM.*

h. Thiết bị logic lập trình được (Programmable Logic Devices, PLD)

i. Gi•i thi•u

Là tên chung của các thiết bị có tính chất nhớ và có thể lập trình để thực hiện một công việc cụ thể nào đó.

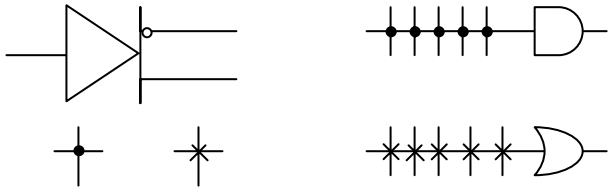
Trong công việc thiết kế các hệ thống, đôi khi người ta cần một số mạch tổ hợp để thực hiện một hàm logic nào đó, mà công việc này lặp lại thường xuyên và sự thay đổi một tham số của hàm có thể phải thực hiện để thỏa mãn yêu cầu của việc thiết kế. Nếu phải thiết kế các mạch logic cơ bản thì mạch sẽ rất 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 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), PLA (Programmable Array Array).

Trước nhất, chúng ta nói qua một số qui ước trong cách biểu diễn các phần tử của PLD.

Một biến trong hàm thường xuất hiện với dạng nguyên và dạng đảo của nó nên chúng ta dùng 2 cổng có hai ngã ra đảo và không đảo.

Một **nối chết** được gọi là **nối cứng** (không thay đổi được) được vẽ bởi một dấu • (chấm đậm). Một **nối sống** được gọi là **nối mềm** (dùng lập trình) được vẽ bởi một dấu ×, nối sống thực chất là một cầu chì, khi lập trình có thể bị phá bỏ.

Một cổng có nhiều ngã vào (khi vẽ) thay thế bởi một ngã vào duy nhất với nhiều mối nối.



Hình: *Biểu diễn các mối nối.*

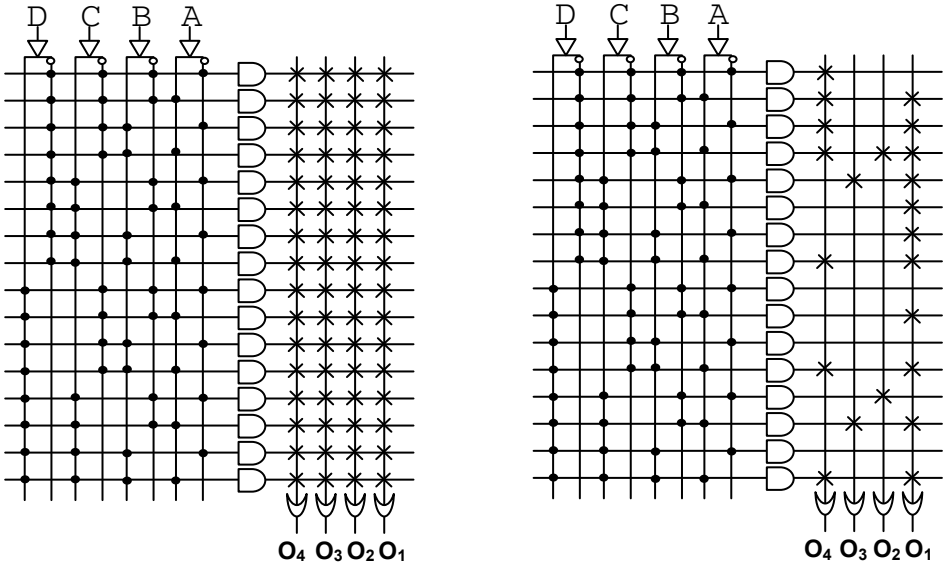
ii. PROM

Để thực hiện 4 hàm 4 biến, mạch có 4 ngã vào và 4 ngã ra.

Có tất cả 16 cổng AND 4 ngã vào và 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 tích 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, đối với PROM việc lập trình thực hiện ở cổng OR.

Dưới đây là sơ đồ của PROM (trái) và PROM đã được lập trình (phải).



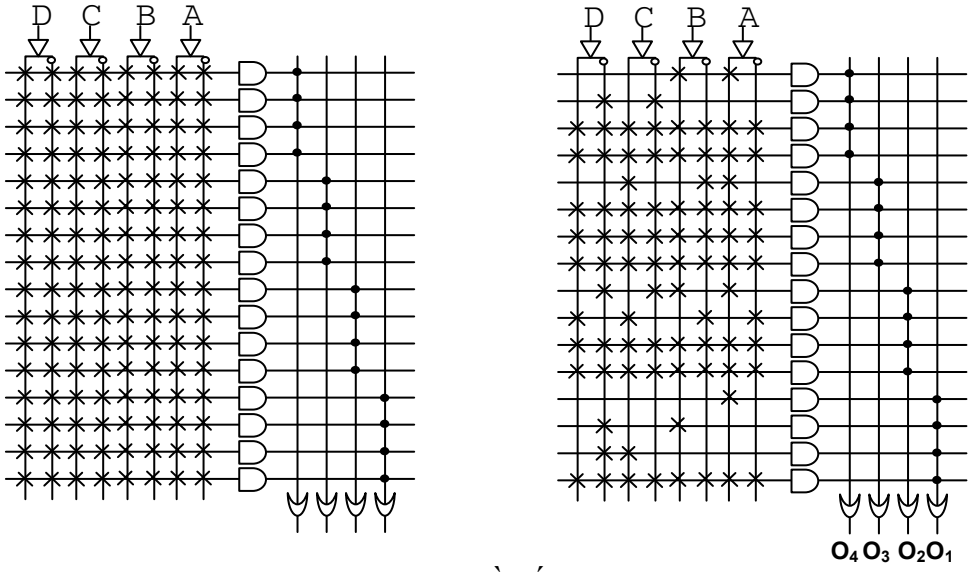
Hình: *Sơ đồ cấu tạo của PROM.*

iii. PAL

Mạch tương tự với IC PROM, PAL có các cổng AND 8 ngã vào được nối sống và 4 cổng OR mỗi cổng OR có 4 ngã vào được nối chết với 4 đường tích. Như vậy việc lập trình được thực hiện trên các đường tích.

$O_1 = A + \overline{D}B + \overline{D}C$ $\qquad$ $O_2 = \overline{D}\overline{C}BA + DC\overline{B}\overline{A}$

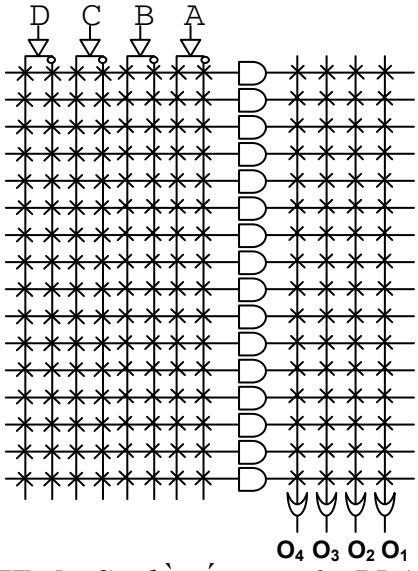
$O_3 = C\overline{B}A$ $\qquad$ $O_4 = BA + \overline{D}\overline{C}$



Hình: *Sơ đồ cấu tạo của PAL.*

iv. PLA

PLA tương tự như hai loại trên nhưng các ngã vào của cổng AND và các cổng OR đều được nối sống. Như vậy khả năng lập trình của PLA bao gồm cả hai cách lập trình của 2 loại kể trên.



Hình: *Sơ đồ cấu tạo của PLA.*

3. RAM (Random Access Memory)

a. Giới thiệu

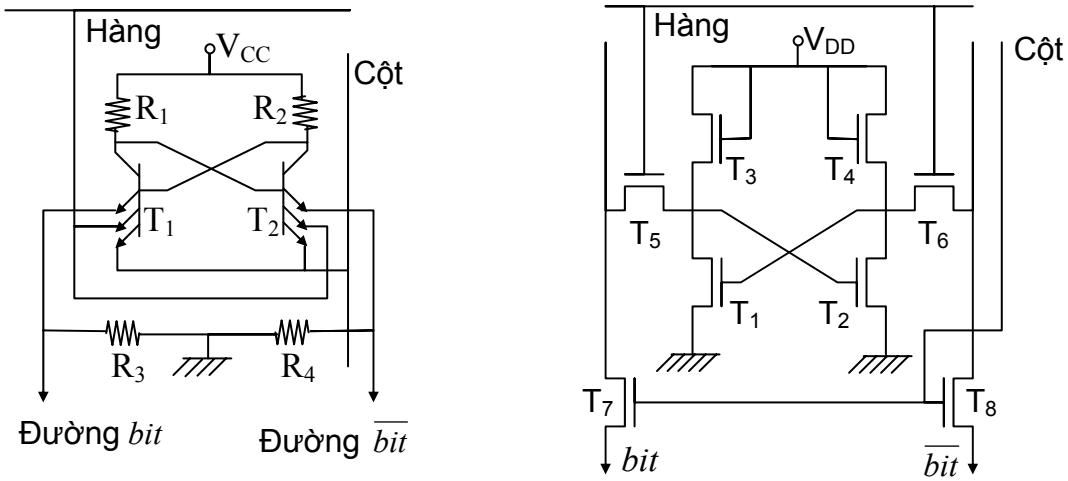
Có hai loại RAM: RAM tĩnh và RAM động.

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 trạng thái của bit 1 và 0. Do RAM động có mật độ tích điện 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 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.

b. RAM tĩnh (Static RAM, SRAM)

Mỗi tế bào RAM tĩnh là một mạch FlipFlop dùng Transitor BJT hay MOS.



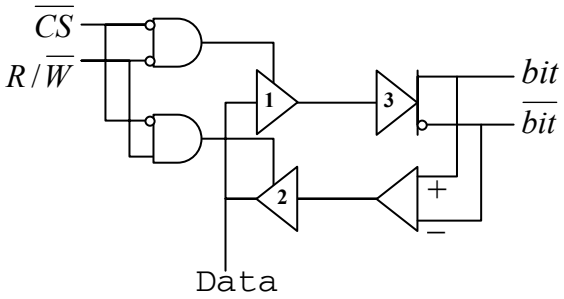
Hình: *Sơ đồ cấu tạo RAM tĩnh*.

Xét SRAM dùng Transistor BJT với hai đường địa chỉ hàng và cột ta thấy: Khi một trong 2 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 2 đường địa chỉ hàng và cột lên cao, mạch hoạt động như một FF, hai trạng thái 1 và 0 của tế bào nhớ được đặt trưng bởi 2 trạng thái khác nhau của 2 đường *bit* và $\overline{bit}$.

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 điện 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 T_2 dẫn hiệu điện thế ở 2 điện trở R_3 và R_4 ngược lại sẽ là bit 0. R_3 và R_4 có tác dụng biến đổi dòng điện qua điện thế.

Đối với tế bào nhớ dùng MOS, hai đường từ nối 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 không được chọn. Khi cả 2 lên cao, mạch hoạt động tương tụ 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.

Dưới đây là mạch điều khiển chọn chip và thực hiện tác vụ đọc/viết vào tế bào nhớ.



Hình: *Mạch điều khiển chọn chip*.

OPAMP giữ vai trò so sánh điện thế hai đường bit và $\overline{bit}$ cho ở ngã ra mức cao hoặc thấp tuỳ kết quả so sánh này (tương ứng với 2 trạng thái của tế bào nhớ) và dữ liệu đọc ra khi cổng đệm thứ 2 mở ($R/\overline{W}$ lên cao).

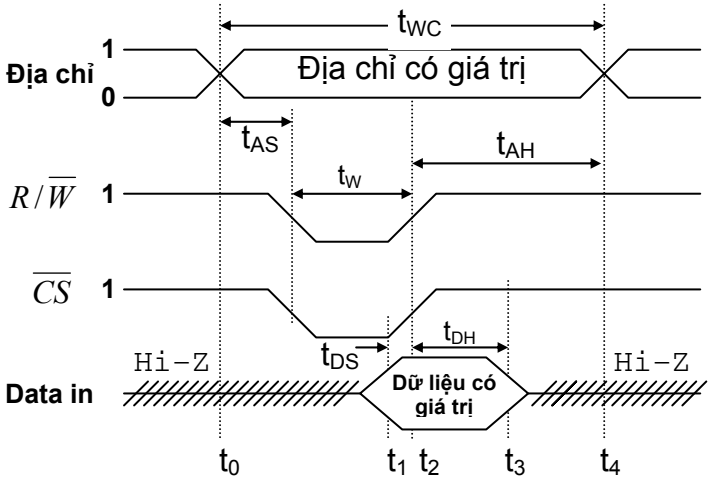
Khi cổng đệm thứ nhất mở, ($R/\overline{W}$ xuống thấp) dữ liệu được ghi vào tế bào nhớ qua cổng đệm 1, cổng 3 tạo ra 2 tín hiệu ngược pha từ dữ liệu vào. Nếu 2 tín hiệu này cùng trạng thái với 2 đường *bit* và $\overline{bit}$ của mạch trước đó, mạch sẽ không đổi trạng thái, nghĩa là tế bào nhớ đang lưu bit giống như bit muốn ghi vào thì mạch không thay đổi. Nếu dữ liệu cần ghi khác với dữ liệu đang lưu trữ thì 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 k• ••c c•a SRAM

Giản đồ thời gian của một chu kỳ đọc SRAM tương tự như giản đồ thời gian của một chu kỳ đọc ROM, thêm điều kiện tín hiệu $R/\overline{W}$ lên cao.

Chu k• vi•t c•a SRAM – t_{wc}

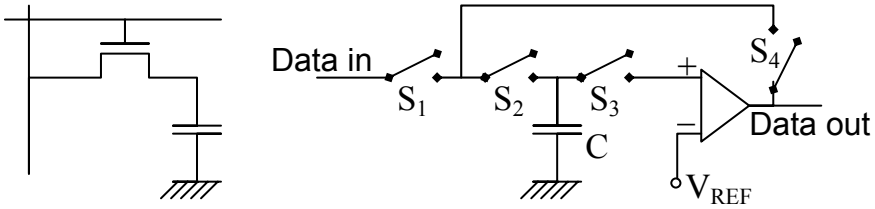
- $\mathbf{t_{AS}}$ (Address setup time): Thời gian thiết lập địa chỉ. Đây là thời gian để giá trị địa chỉ ổn định trên BUS địa chỉ cho tới lúc tín hiệu $\overline{CS}$ tác động.
- $\mathbf{t_W}$ (Write time): Thời gian từ lúc $\overline{CS}$ tác động đến lúc dữ liệu có giá trị trên BUS dữ liệu.
- $\mathbf{t_{DS}}$ và $\mathbf{t_{DH}}$: Khoảng thời gian dữ liệu còn tồn tại trên BUS dữ liệu kể từ khi tín hiệu $\overline{CS}$ hết hiệu lực, bao gồm thời gian trước ($\mathbf{t_{DS}}$) và thời gian sau ($\mathbf{t_{DH}}$).
- $\mathbf{t_{AH}}$ (Address hold time): Thời gian giữ địa chỉ là thời gian từ lúc tín hiệu $\overline{CS}$ không còn tác động đến lúc xuất hiện địa chỉ mới.



Hình: *Giản đồ thời gian cho một chu kỳ viết của RAM.*

c. RAM động (Dynamic RAM, DRAM)

Dưới đây là cấu tạo của một tế bào DRAM.



Hình: *Tế bào nhớ của DRAM.*

Hình trên là một cách biểu diễn tế bào nhớ của 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, S_2 đóng trong khi S_3, S_4 mở bit 1 thực hiện việc nạp cho tụ C và bit 0 làm cho 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 rĩ 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 được nối với một mạch so sánh với một điện thế chuẩn 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, điện thế 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.

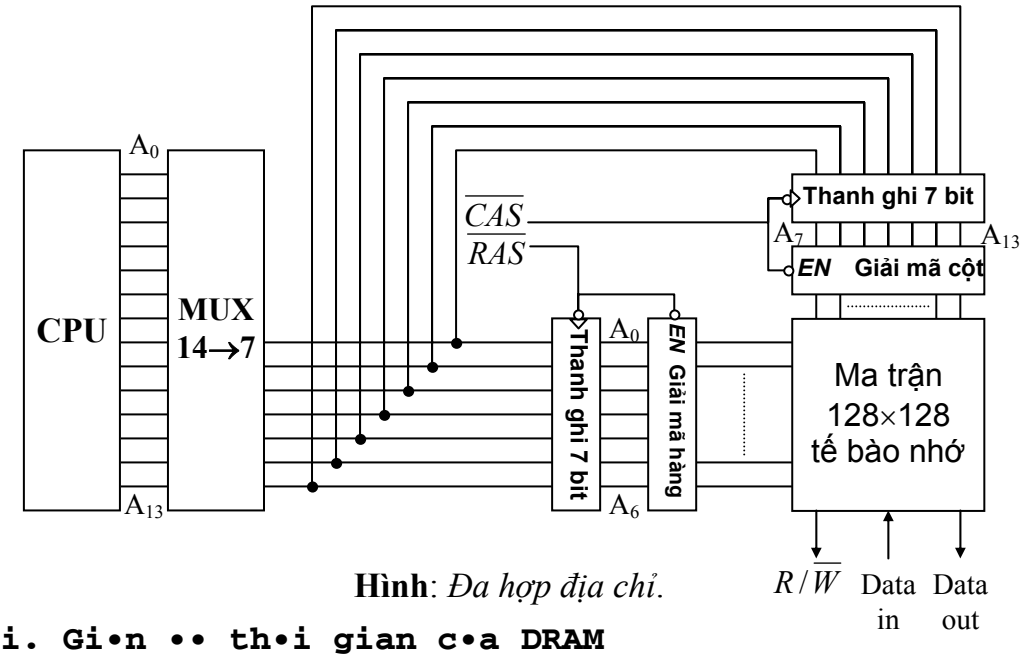
Sử dụng DRAM, được một thuận lợi là dung lượng nhớ khá lớn nhưng phải có một số mạch phụ trợ như:

- 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.

i. •a h•p ••a ch•

Như đã nói trên, do dung lượng 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.

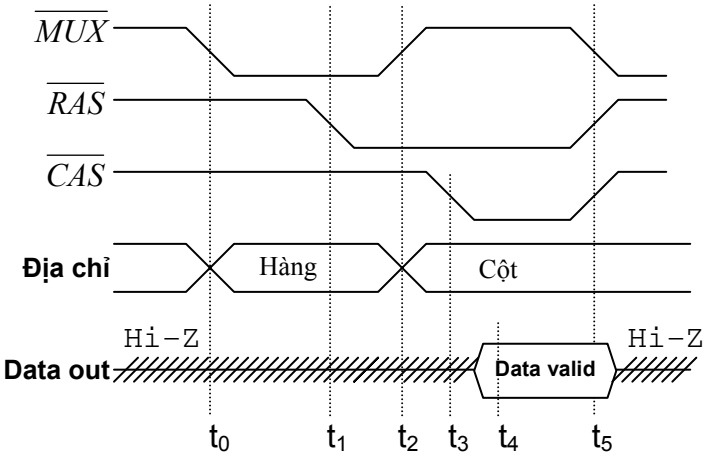
Ví dụ với DRAM có dung lượng 16K×1, thay vì phải dùng 14 đường địa chỉ, ta chỉ cần dùng 7 đường và mạch đa hợp 14→7 (7 đa hợp 2→1) để chọn 7 trong 14 đường địa chỉ ra từ CPU. Bộ nhớ có cấu trúc là một ma trận 128×128 tế bào nhớ, sắp xếp thành 128 hàng và 128 cột, có một ngã vào và một ngã ra dữ liệu, 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 rỏ hơn khi xét các giản đồ thời gian của DRAM.



Hình: *Đa hợp địa chỉ.*

ii. Giản đồ thời gian của DRAM

Giản đồ thời gian đọc và viết tiêu biểu của DRAM chỉ khác nhau về thời lượng nhưng có chung một dạng, nên ta chỉ vẽ một giản đồ.



Hình: *Giản đồ thời gian của DRAM.*

Giản đồ cho thấy tác động của tín hiệu $\overline{MUX}$ và các tín hiệu $\overline{RAS}$, $\overline{CAS}$. Khi $\overline{MUX}$ ở mức thấp mạch đa hợp cho ra địa chỉ hàng ($A_6 \ldots A_0$) 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 ($A_{13} \ldots A_7$) và được chốt vào thanh ghi khi tín hiệu $\overline{CAS}$ xuống thấp. Khi cả hai địa chỉ hàng và cột đều được giản mã, dữ liệu tại địa chỉ đó xuất hiện trên BUS dữ liệu để đọc ra hoặc ghi vào.

iii. Làm tươi DRAM

DRAM phải được làm tươi với chu kỳ khoảng 2*ms* để duy trì dữ liệu.

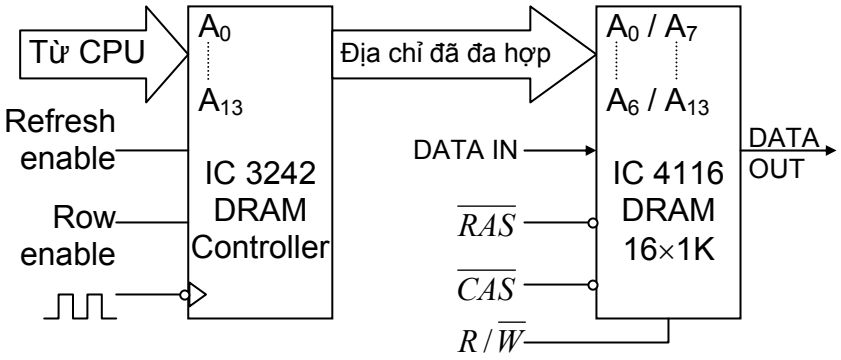
Trong phần trước ta đã thấy tế bào nhớ của DRAM được làm tươi ngay khi tác vụ đọc được thực hiện. Lấy ví dụ với DRAM với dung lượng 16K×1 (16×1024×1=16 3 84 tế bào nhớ) nói trên, chu kỳ làm tươi là 2*ms* cho 16 3 84 tế bào nhớ nên thời gian đọc mỗi tế bào phải là $2ms/16384 \approx 122ns$. Đây là khoảng thời gian rất nhỏ không đủ để đọc tế bào nhớ trong điều kiện vận hành bình thường. Chính vì lý

do nầy 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 ví 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 hãng Intel thiết kế để sử dụng cho DRAM 16K (hình dưới).

Ngã ra IC 3242 là địa chỉ 7bit đã được đa hợp và nối vào ngã vào địa chỉ của DRAM. Một mạch đếm 7bit kích bởi xung đồng hồ riêng để cấp địa chỉ hàng cho DRAM trong suốt thời gian làm tươi. IC 3242 cũng lấy địa chỉ 14bit 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 7bit nào của địa chỉ xuất hiện ở ngã ra controller cho bởi bảng dưới đây.

REFRESH ENABLE	ROW ENABLE	CONTROLLER OUTPUT
High	×	Refresh address từ mạch đếm
Low	High	Địa chỉ hàng (A_6…A_0 từ CPU)
Low	Low	Địa chỉ cột (A_{13}…A_7 từ CPU)



Hình: *Sơ đồ mắc IC 3242 vào DRAM.*

VI.MỞ RỘNG BỘ NHỚ

1. Giới thiệu

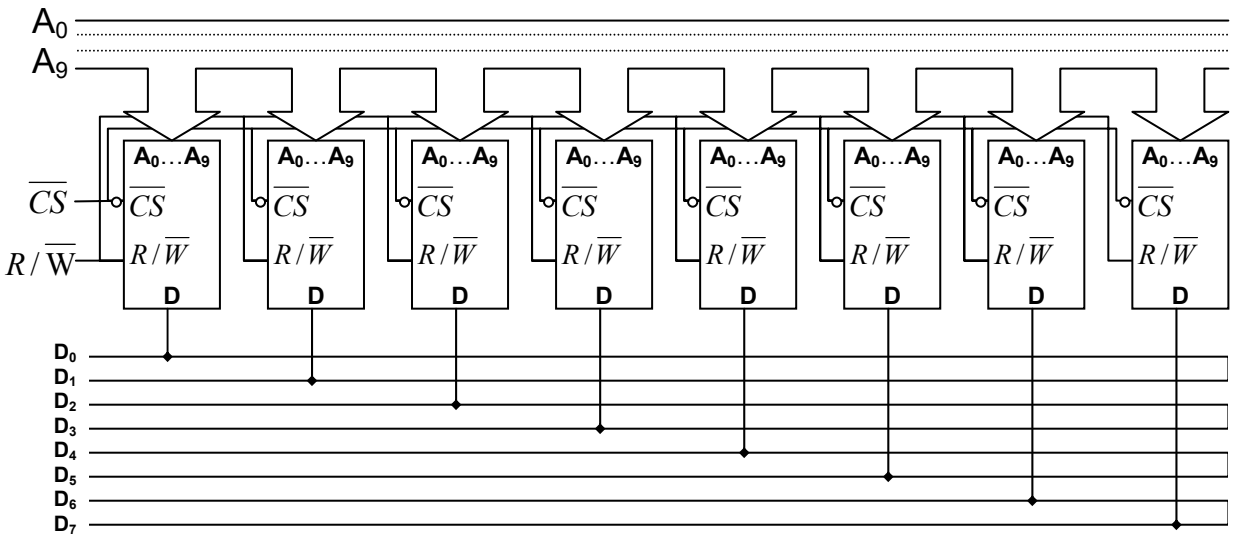
Các IC nhớ thường được chế tạo với dung lượng nhớ có gới hạn, trong nhiều trường hợp không thể thỏa mãn yêu cầu của việc thiết kế. Do đó mở rộng bộ nhớ là một việc làm cần thiết. Có 3 trường hợp phải mở rộng bộ nhớ, ta sẽ xét dưới đây.

2. Mở rộng độ dài từ

Đây là trường hợp số vị trí nhớ đủ cho yêu cầu nhưng dữ liệu tại mỗi vị trí nhớ không đủ. Có thể hiểu được cách mở rộng độ dài từ qua một ví dụ.

Ví dụ: Mở rộng bộ nhớ từ 1K×1 lên 1K×8.

Chúng ta cần dùng 8 IC 1K×1, các IC nhớ này sẽ được nối chung BUS địa chỉ và các tín hiệu điều khiển và mỗi IC quản lý một đường bit. 8 IC sẽ vận hành cùng lúc để cho một từ nhớ 8bit.

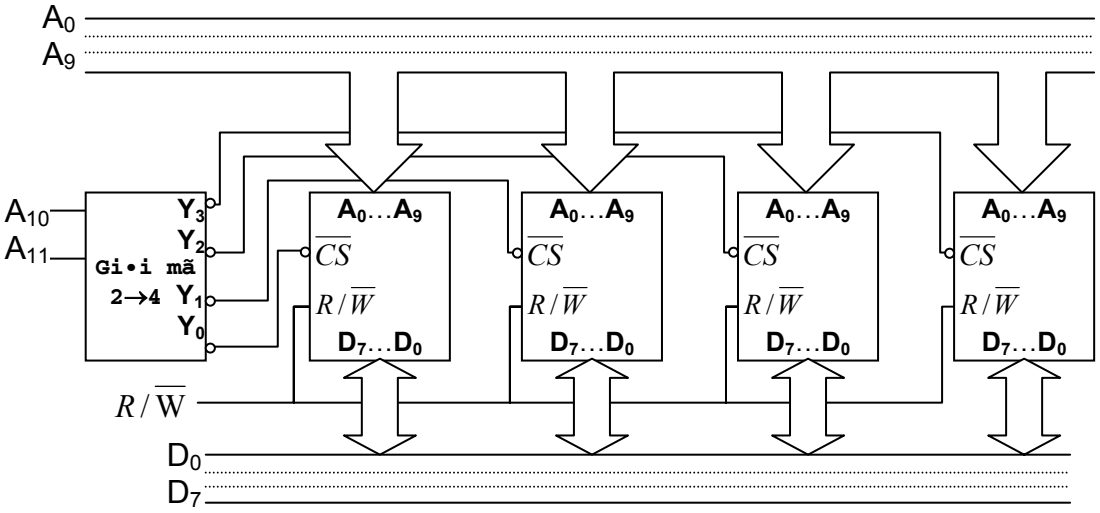


Hình: *Mở rộng độ dài từ 1K×1→1K×8.*

3. Mở rộng vị trí nhớ

Đây là trường hợp số bit cho mỗi vị trí nhớ đủ theo yêu cầu nhưng số vị trí nhớ không đủ.

Ví dụ: Có IC nhớ dung lượng 1K×8. Mở rộng lên 4K×8, vậy cần 4 IC này. Để chọn 1 trong 4 IC nhớ, ta 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 IC nhớ, như vậy địa chỉ của IC nhớ sẽ khác nhau. Trong ví dụ này, IC1 chiếm địa chỉ từ 000H đến 3FFH, IC 2 chiếm địa chỉ từ 400H đến 7FFH, IC 3 chiếm địa chỉ từ 800H đến BFFH, IC 4 chiếm địa chỉ từ C00H đến FFFH.



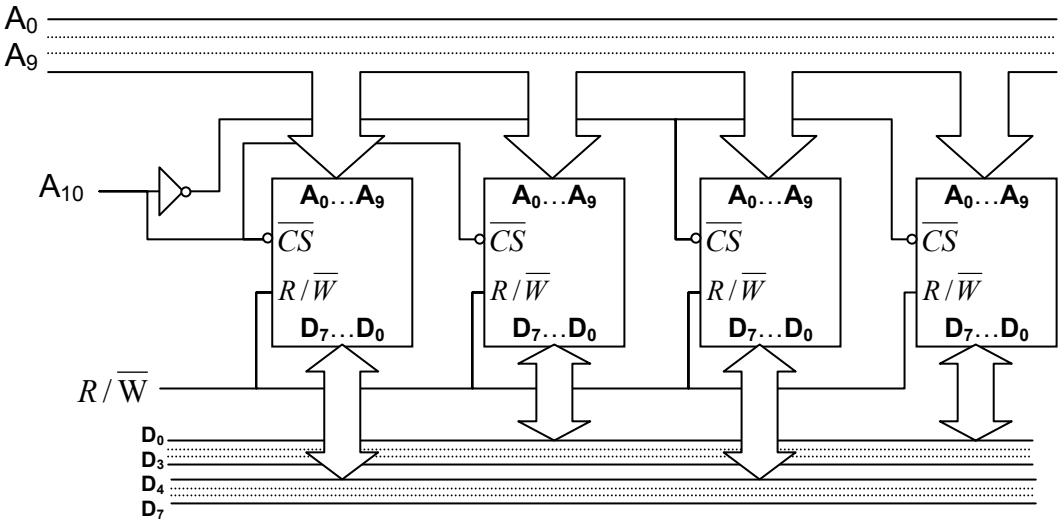
Hình: *Mở rộng vị trí nhớ 1K×8→4K×8.*

4. Mở rộng dung lượng nhớ

Đây là trường hợp cả vị trí nhớ và độ dài của IC đều không đủ để thiết kế. Để mở rộng dung lượng nhớ ta phải kết hợp cả 2 cách trên.

Ví dụ:Mở rộng bộ nhớ từ 1K×4 lên 2K×8. Cần 2 cặp IC mắc song song, mỗi IC có chung địa chỉ và được chọn bởi một mạch giải mã 1 đường sang 2 đường.

Địa chỉ của các IC như sau: IC 1(&2): 000H đến 3FFH, IC (3&4): 400H đến 7FFH.



Hình: *Mở rộng dung lượng nhớ 1K×4→2K×8.*

CHƯƠNG 8: BIẾN ĐỔI AD & DA

- ✓ **BIẾN 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**
  - **Đặt tính kỹ thuật của DAC**
- ✓ **BIẾN ĐỔI TƯƠNG TỰ – SỐ (ADC)**
  - **Mạch lấy mẫu và giữ**
  - **Nguyên tắc của 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**

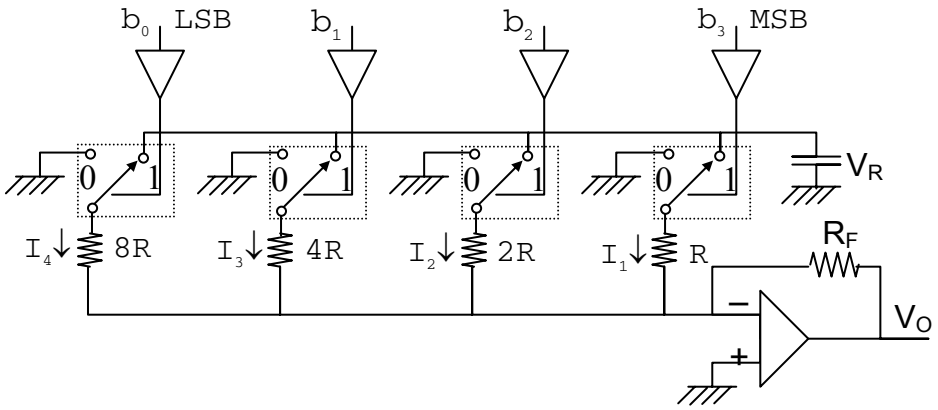
I. GIỚI THIỆU

Có thể nói sự biến đổi qua lại giữa các tín hiệu từ dạng tương tự sang dạng số (và ngược lại) là cần thiết vì:

- Hệ thống 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 (ví dụ tái tạo âm thanh, hình ảnh,…) hay dùng vào việc điều khiển sau đó (ví dụ dùng điện thế tương tự để điều khiển các vận tốc động cơ).

II. BIẾN ĐỔI SỐ – TƯƠNG TỰ (DIGITAL TO ANALOG CONVERTER, DAC)

1. Mạch biến đổi DAC dùng mạng điện trở có trọng lượng khác nhau



Hình: *Mạch biến đổi DAC dùng mạng điện trở có trọng lượng khác nhau.*

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 điện thế, đồng thời nó là một mạch cộng.

Ta có: $$V_O = -R_F.I = \frac{-(2^3 b_3 + 2^2 b_2 + 2^1 b_1 + b_0)V_R.R_F}{2^3 R}$$
$$= \frac{-(2^{n-1} b_{n-1} + 2^{n-2} b_{n-2} + ... + 2b_1 + b_0)V_R.R_F}{2^{n-1} R}$$

Nếu $R_F = R$ thì: $V_O = \frac{-(2^{n-1} b_{n-1} + 2^{n-2} b_{n-2} + ... + 2b_1 + b_0)V_R}{2^{n-1}}$

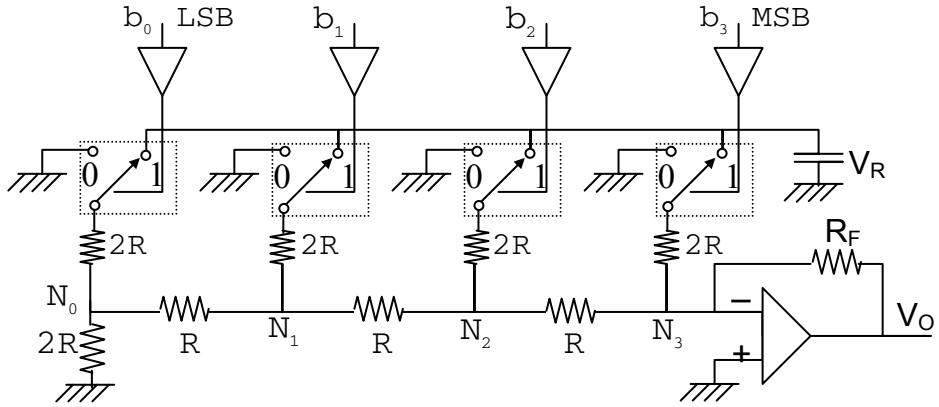
Ví dụ: Khi số nhị phân là 0000 thì $v_o = 0V$, 1111 thì $v_o = -15V$.
Với $V_R = 5V$; $R = R_F = 1K\Omega$, ta có kết quả như sau:

b_3	b_2	b_1	b_0	V_O (*V*)
0	0	0	0	0
0	0	0	1	– 0.625
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

Mạch có một số hạn chế như sau:

- Sự chính xác tùy thuộc vào điện trở và mức độ ổn định của nguồn tham chiếu V_R.
- Với số nhị phân nhiều bit, thì cần các điện trở có giá trị rất lớn, khó thực hiện.

2. Mạch biến đổi DAC dùng mạng điện trở hình thang



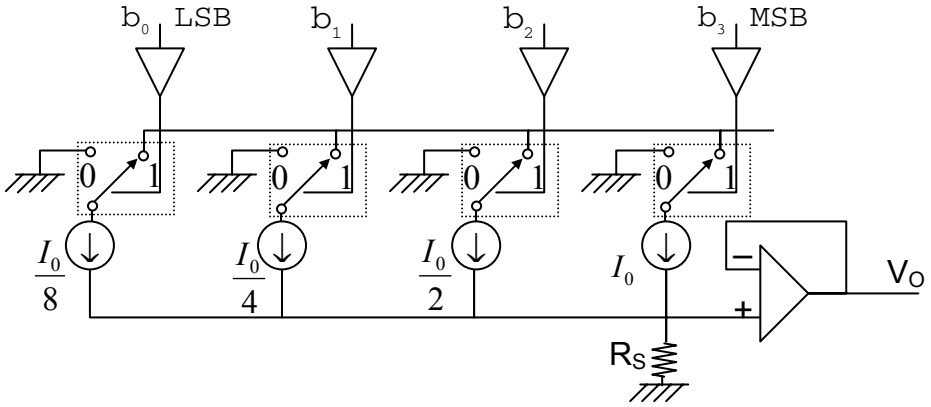
Hình: *Mạch biến đổi DAC dùng mạng điện trở hình thang.*

Cho $R_F = 2R$ và lần lượt cho các bit có giá trị như dưới đây.

- Cho $b_3 = 1$, các bit khác = 0, ta được $v_o = -8\ (V_r / 24)$.
- Cho $b_2 = 1$, các bit khác = 0, ta được $v_o = -4\ (V_r / 24)$.
- Cho $b_1 = 1$, các bit khác = 0, ta được $v_o = -2\ (V_r / 24)$.
- Cho $b_0 = 1$, các bit khác = 0, ta được $v_o = -(V_r / 24)$.

Ta thấy, v_o tỉ lệ với giá trị B của tổ hợp $B = b_3b_2b_1b_0 \Rightarrow v_o = -B(V_R / 24)$.

3. Mạch biến đổi DAC dùng nguồn có trọng lượng khác nhau



Hình: *Mạch biến đổi DAC dùng nguồn có trọng lượng khác nhau.*

4. Đặt tính kỹ thuật của mạch biến đổi DAC

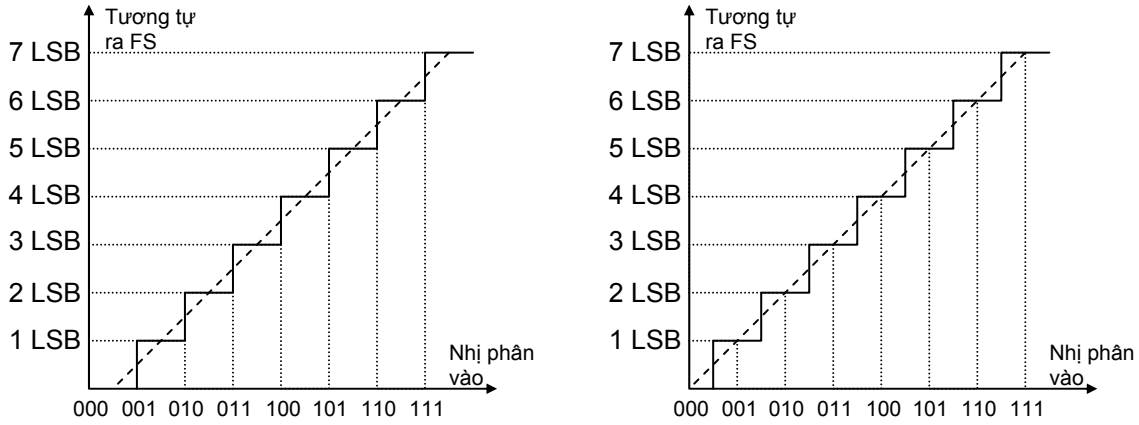
a. Bit có ý nghĩa thấp nhất và bit có ý nghĩa cao nhất

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 dổi khac 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 và bit MSB có giá trị là:

- $LSB = V_{FS} / (2^n - 1)$.
- $MSB = V_{FS}.2^{n-1} / (2^n - 1)$.

Điều này được thể hiện trong kết quả của 2 ví dụ trên. Hình dưới đây là đặc tuyến chuyển đổi của một số nhị phân 3 bit.



Hình: *Đặc tuyến chuyển đổi của một số nhị phân 3 bit.*

Đặc tuyến chuyển đổi của một số nhị phân 3 bit là hai hình trên, (a) là đặc tuyến lý tưởng, tuy nhiên trong thực tế để đường trung bình của đặc tuyến chuyển đổi đi qua

điểm 0 điện thế tương tự ra lệch ½ LSB (b). Như vậy điện thế tương tự ra được xem như thay đổi ở ngay giữa hai mã số nhị phân kế tiếp nhau. Ví dụ, khi mã số nhị phân vào là 000 thì điện thế tương tự ra là 0 và điện thế tương tự sẽ lên nấc kế 000 + ½LSB, rồi lên nấc kế nữa ở 001 + ½LSB… Trị tương tự ra ứng với 001 gọi tắt là 1LSB và trị toàn giai V_{FS} = 7LSB tương ứng với số 111.

b. 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 bằng ½ LSB.

c. 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 phânn n bit có 2^n giá trị và $2^n - 1$ bước.

Hiệu thế tương tự xác định bởi $v_o = k.(B)_2$. Trong đó k là độ phân giải và B_2 là số nhị phân. Người ta thường tính % độ phân giải:

$$\%res = (k / V_{FS}).100\%$$

Với số nhị phân n bit ta có:

$$\%res = \frac{1}{2^n - 1}.100\%$$

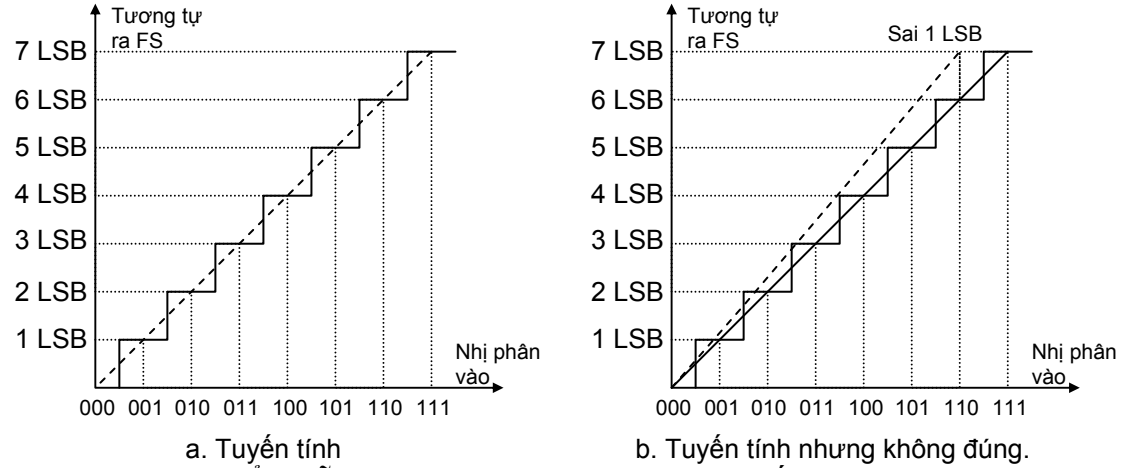
Các nhà sản xuất thường dùng số bit của số nhị phân có thể được biến đổi để chỉ độ phân giải. Số bit càng lớn thì độ phân giải càng cao (finer solution).

d. Độ tuyến tính (linearity)

Khi điện thế tương tự ra thay đổi đều với số nhị phân vào ta nói mạch biến đổi có tuyến tính.

e. 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.



a. Tuyến tính b. Tuyến tính nhưng không đúng.

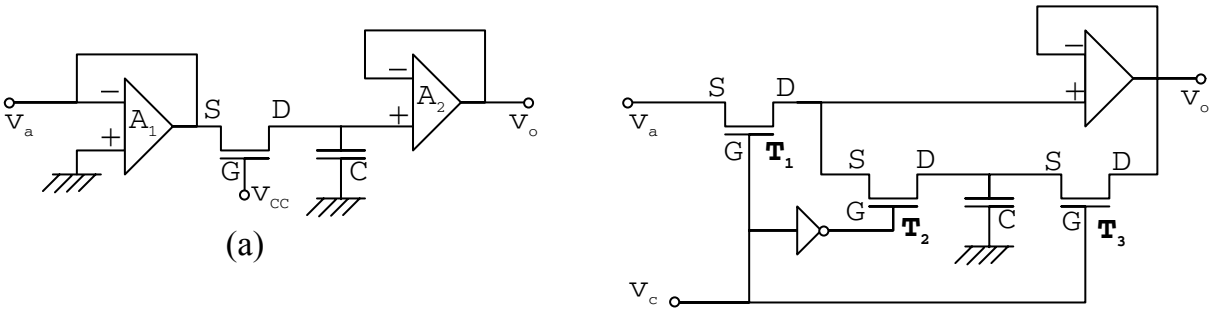
Hình: *Biểu diễn độ đúng (accuracy) của một số nhị phân 3 bit.*

III. BIẾN ĐỔI TƯƠNG TỰ – SỐ (ANALOG TO DIGITAL CONVERTER, ADC)

1. Mạch lấy mẫu và giữ

Để biến đổi một tín hiệu tương tự sang số, mạch biến đổi cần một khoảng thời gian cụ thể (khoảng $1\mu s$ đến $1ms$) 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.

Hình dưới đây là dạng mạch lấy mẫu và giữ cơ bản: Điện thế tương tự cần biến đổi được lấy mẫu trong khoảng thời gian rất ngắn do tụ nạp điện nhanh qua tổng trở ra thấp của OP–AMP khi qua các transistor dẫn và giữ giá trị này trong khoảng thời gian transitor ngưng (tụ phóng rất chậm qua tổng trở vào rất lớn của OP–AMP).

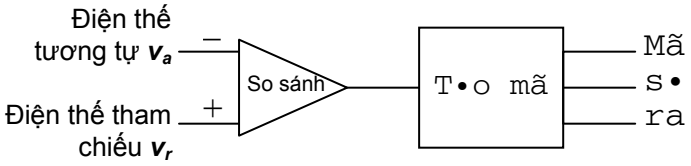


Hình: *Mạch lấy mẫu và giữ cơ bản.*

2. Nguyên tắc mạch biến đổi ADC

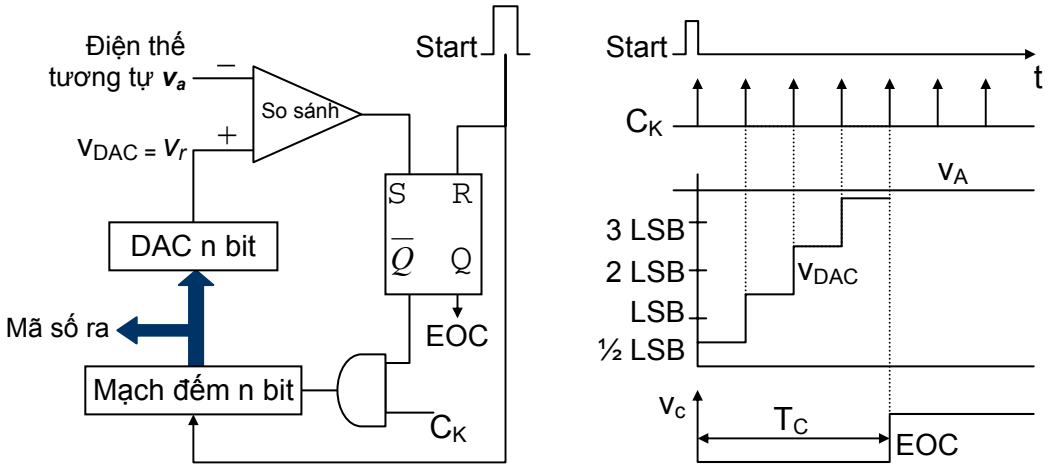
Mạch biến đổi ADC gồm bộ phận trung tâm là một mạch so sánh, còn ngã vào kia nối đến một điện thế tham chiếu thay đổi theo thời gian $V_r(t)$. Khi chuyển đổi điện thế tham chiếu tăng theo thời gian cho đến khi bằng hoặc gần bằng với điện thế tương tự (với một sai số nguyên lượng hóa). Lúc đó mạch tạo 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 thế giữa v_a và trị nguyên lượng hóa sau cùng nhỏ hơn ½ LSB.

$$\left| v_a - \frac{V_{FS}}{2^n - 1}.(B)_2 \right| < \frac{1}{2} LSB$$



Hình: *Mạch mô tả nguyên tắc biến đổi ADC.*

3. Mạch đổi dùng điện thế tham chiếu nấc thang

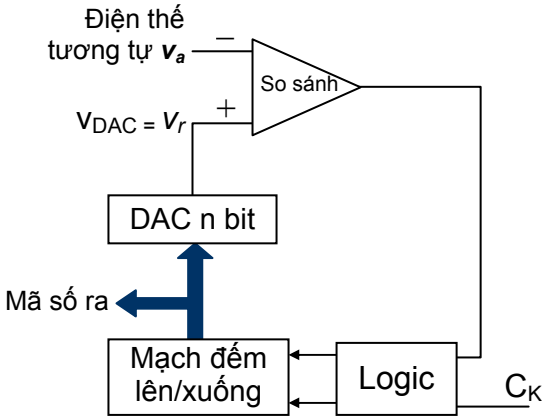


Hình: *Mạch đổi dùng điện thế tham chiếu nấc thang.*

Một cách đơn giản để tạo điện thế tham chiếu có dạng nấc thang là dùng một mạch DAC mà số nhị phân vào lấy được từ mạch đếm lên (hình trên). Khi có xung bắ đầu FF 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 V_r 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 V_r vượt qua V_a ngã ra mạch so sánh lên cao khiến $\overline{Q}$ ở mức thấp, đóng cổng AND không cho xung C_K qua và mạch đếm ngưng. Đồng thời 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.

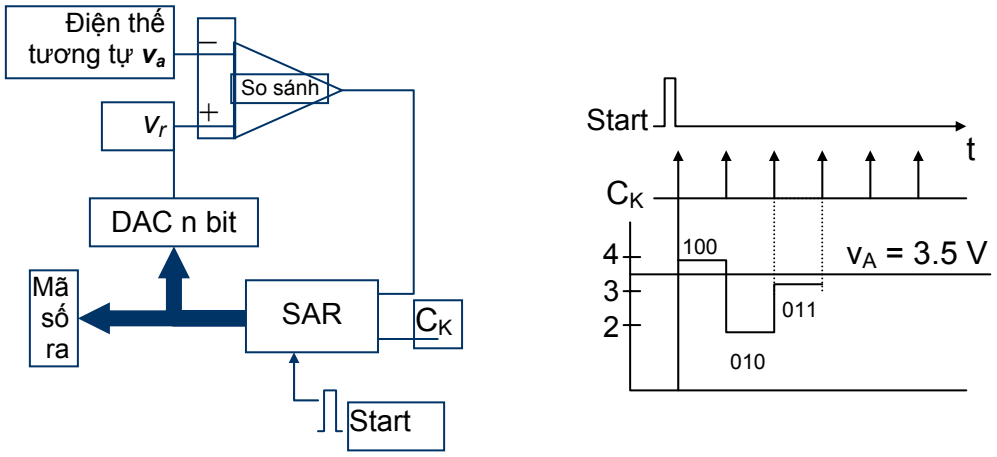
Gọi thời gian chuyển đổi là t_C. Thời gian chuyển đổi tùy thuộc vào điện thế cần chuyển đổi. Thời gian lâu nhất ứng với điện thế vào bằng trị toàn giai: $t_{c(\max)} = \dfrac{2^n}{f_{CK}}$.

Mạch đổi này có tốc độ chậm. Một cách cải tiến là thay mạch đếm lên bởi một mạch đếm lên xuống (hình dưới). Nếu ngã ra mạch so sánh cho thấy v_r nhỏ hơn v_a, mạch logic sẽ điều khiển việc đếm lên và ngược lại mạch sẽ đếm xuống. Nếu v_a không đổi v_r sẽ dao động quanh trị v_a với hai trị số khác nhau 1 LSB.



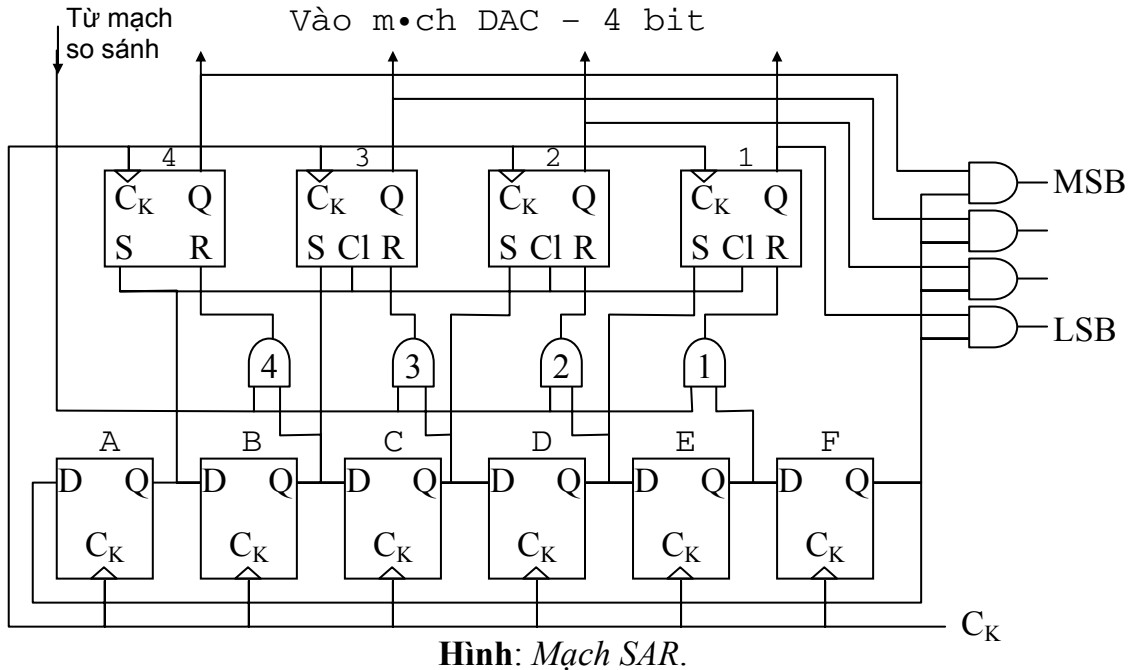
Hình: *Mạch đổi dùng điện thế tham chiếu nấc thang cải tiến.*

4. Mạch đổi lấy gần đúng kế tiếp



Hình: *Mạch đổi lấy gần đúng kế tiếp.*

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 gồm: Mạch so sánh, mạch ghi chuyển đặc biệt (SAR) và mạch DAC (hình dưới đây).

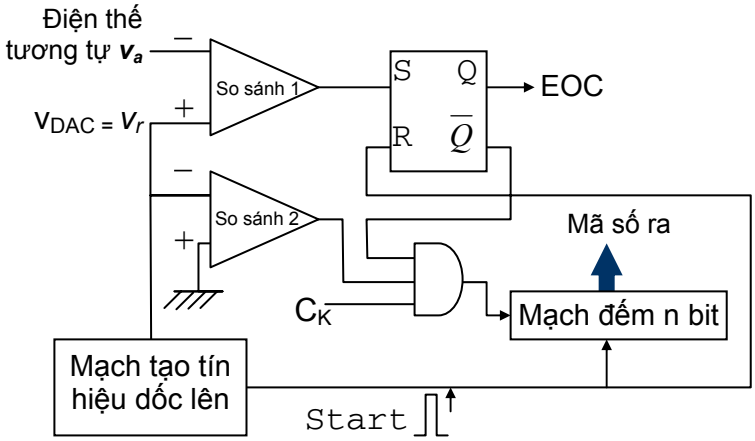


Hình: *Mạch SAR.*

Mạch SAR (hình trên) là mạch ghi dịch có kết hợp điều khiển logic. Lúc có xung bắt đầu, mạch SAR được đặt về 0, ngã ra DAC được làm lệch ½ LSB để tạo đặc tính chuyển đổi như đã nói trong phần trước, kế đó SAR đưa bit MSB lên cao, các bit khác bằng 0, số này được đưa vào DAC để tạo điện thế tham chiếu v_r để so sánh với v_a. Tùy theo kết quả so sánh, nếu $v_r > v_a$ thì ngã ra mạch so sánh ở mức thấp khiến SAR bỏ đi bit MSB, nếu $v_r < v_a$ thì ngã ra mạch so sánh ở mức cao, khiến SAR giữ bit MSB lại. Tiếp tục như vậy cho đến khi đến bit cuối cùng của SAR, lúc đó v_a gần với v_r nhất và ta được kết quả chuyển đổi trong thời gian tối đa là ***n*** chu kỳ xung đồng hồ.

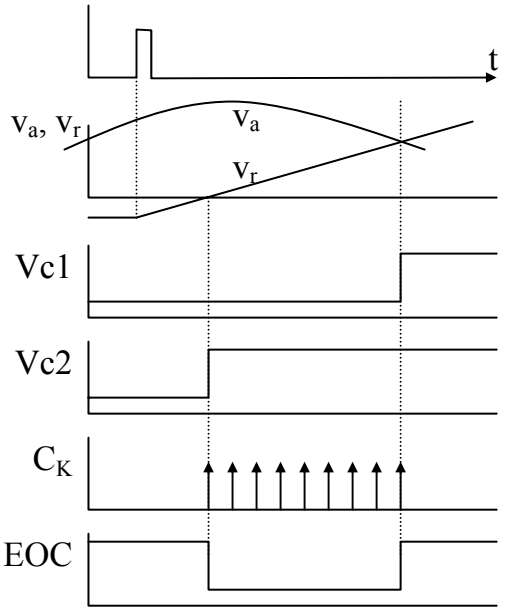
5. Mạch đổi dùng tín hiệu dốc đơn

Điện thế chuẩn từng nấc tạo bởi mạch DAC có thể được thay thế bởi một điện thế tham chiếu có dốc liên tục tạo bởi mạch tín hiệu dốc lên (thường là mạch tích phân).



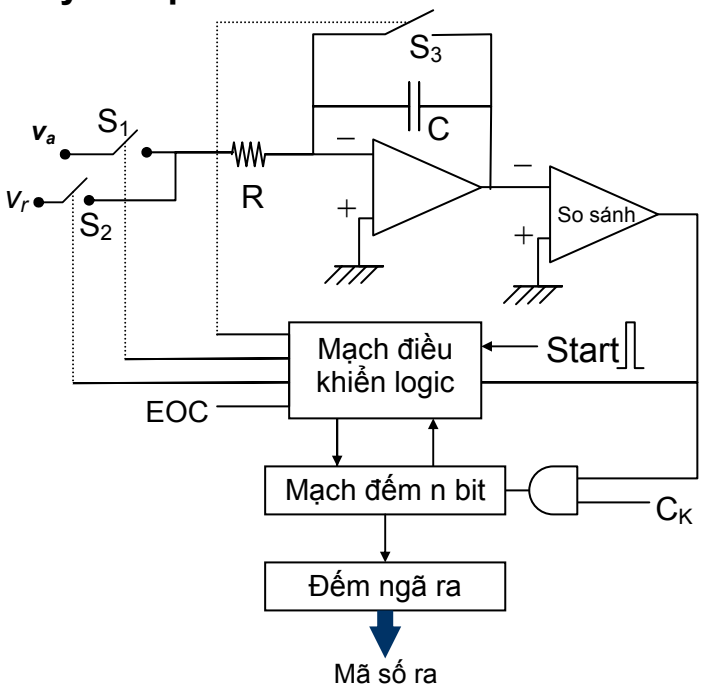
Hình: *Mạch đổi dùng tín hiệu dốc đơn.*

Xung bắt đầu đặt mạch đếm n bit về 0 và khởi động tạo mạch dốc lên để tạo v_r, từ một trị hơi âm, khi v_r 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 Q của FF xuống thấp, cổng AND đóng và kết thúc sự chuyển đổi. Số đếm được ở mạch tỷ lệ với điện thế tương tự vào. Mạch có khuyết điểm là độ dốc của v_r tùy thuộc thông số RC của mạch tích phân nên không chính xác.



Hình: *Đồ thị của mạch đổi dùng tín hiệu dốc đơn.*

6. Mạch đổi lấy tích phân



Hình: *Mạch đổi lấy tích phân.*

Mạch này giải quyết được khuyết điểm của mạch biến đổi dùng tín hiệu dốc đơn, nghĩa là độ chính xác không tùy thuộc vào 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 $-\frac{v_a}{RC}$. Khi ngã ra 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 điện thế ngã ra mạch tích phân là:

$$V_I(t) = \int -\frac{v_a}{RC} dt$$

Giả sử v_a không đổi trong khoảng thời gian chuyển đổi, ta được:

$$V_I(t) = -\frac{v_a . t}{RC}$$

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 v_r (dương) đến mạch lấy tích phân. Ngã ra mạch tích phân bây giờ là đường dốc xuống với độ dốc là $-\frac{v_r}{RC}$. 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:

$$|v_a| t_1 = v_r t_2$$

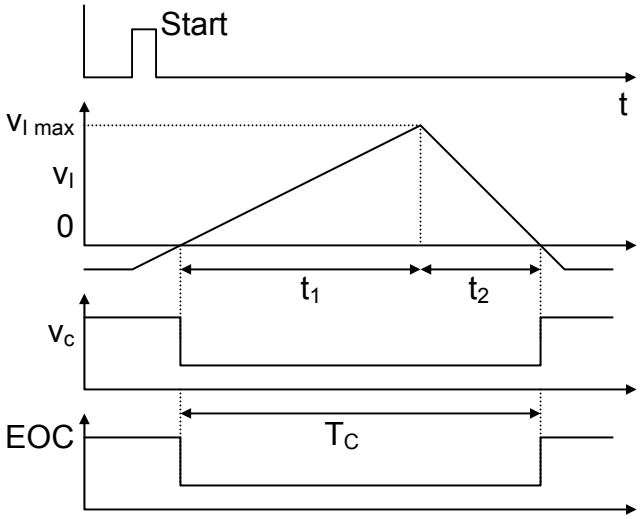
t_1 là thời gian đếm từ 0 cho đến khi tràn nên:

$$t_1 = \frac{2^n}{f_{C_K}}$$

$$t_2 = \frac{N}{f_{C_K}}$$

N là số đếm sau cùng.

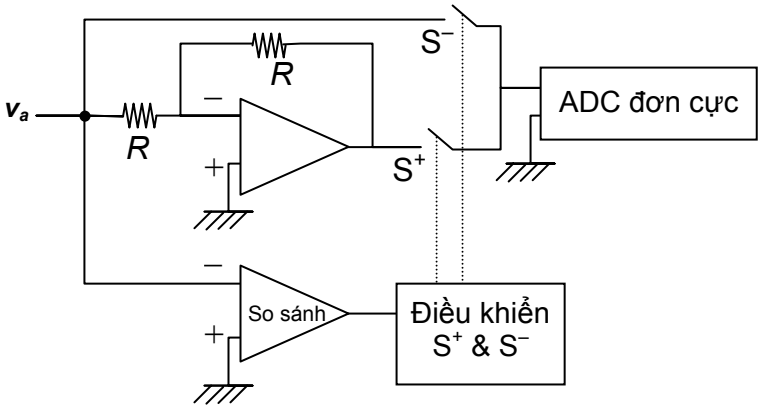
Tóm lại ta thấy số đếm không phụ thuộc vào RC.



Hình: *Đồ thị mạch đổi tích phân.*

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 (hình dưới).



Hình: *Mạch đổi lưỡng cực.*

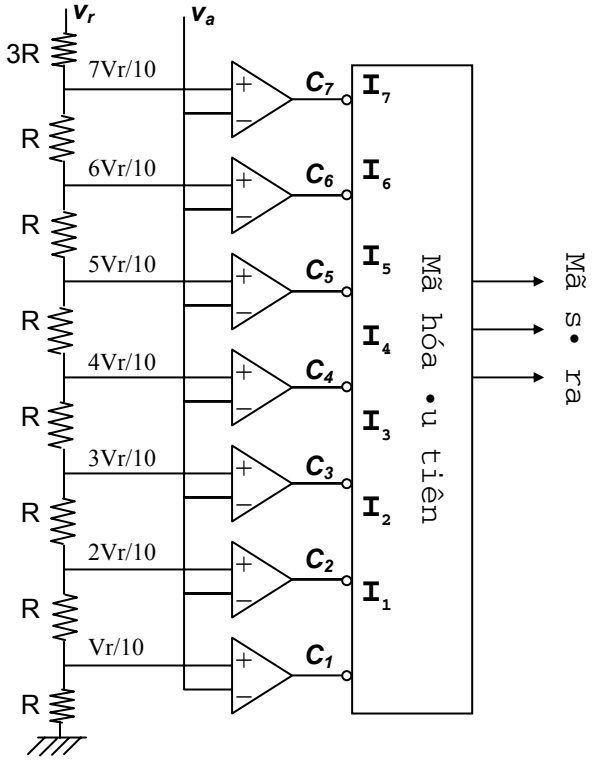
8. Mạch đổi song song

Đây là mạch đổi có tốc độ chuyển đổi rất nhanh, có thể đạt vài triệu lần trong 1 giây, áp dụng vào việc chuyển đổi tín hiệu trong kỹ thuật video. Ví dụ để có mạch 3 đổi 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 (hình dưới).

- Khi $v_a < \frac{v_r}{10}$, các ngã ra mạch so sánh đều lên cao khiến mã số ra là: ***000***.

- Khi $\frac{v_r}{10} < v_a < \frac{2v_r}{10}$, ngã ra mạch so sánh 1 xuống thấp, mã số ra là: ***001***.
- Khi $\frac{2v_r}{10} < v_a < \frac{3v_r}{10}$, ngã ra mạch so sánh 2 xuống thấp, mã số ra là: ***010***.

Cứ như thế, ta thấy mã số ra tỷ lệ với điện thế tương tự vào.



Hình: *Mạch đổi song song*.