

ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
KHOA CÔNG NGHỆ ĐIỆN TỬ VÀ TRUYỀN THÔNG

BÀI GIẢNG :
THIẾT KẾ MẠCH LOGIC VÀ ANALOG
(Tài liệu lưu hành nội bộ)

Thái nguyên, tháng 10 năm 2012

PHẦN I: THIẾT KẾ MẠCH LOGIC

Chương I: Đại số boole và các linh kiện điện tử số

1.1. Một số khái niệm cơ bản

- Biến logic: Đại lượng biểu diễn bằng ký hiệu nào đó chỉ lấy giá trị "1" hoặc "0".
- Hàm logic: Biểu diễn nhóm các biến logic liên hệ với nhau thông qua các phép toán logic, một hàm logic cho dù là đơn giản hay phức tạp cũng chỉ nhận giá trị hoặc là "1" hoặc là "0".
- Các phép toán logic: có 3 phép toán cơ bản.

Phép nhân (và) - kí hiệu là AND.

Phép cộng (hoặc) - kí hiệu là OR.

Phép phủ định (đảo) - kí hiệu là NOT

1.1.1. Biểu diễn biến và hàm logic

b. Bảng thật, bảng trạng thái:

*Bảng thật : Quan hệ hàm ra với biến vào ở thời điểm hiện tại.

*Bảng trạng thái: Hàm ra không những phụ thuộc vào biến vào ở thời điểm hiện tại mà còn phụ thuộc vào (trạng thái) quá khứ của nó.

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

Bảng thật $f(A,B) = A+B$

C	D_n	Q_{n+1}
0	0	Q_n
0	1	Q_n
1	0	0
1	1	1

Bảng trạng thái

b. Bìa Karnaugh (Bìa các nô).

Biểu diễn tổng đ-ong đ-ong bảng thật. Mỗi dòng của bảng thật ứng với một ô của bìa các nô. Toạ độ của ô đ-ợc quy định bởi giá trị tổ hợp biến, giá trị của hàm tổng đ-ong ứng với tổ hợp biến đ-ợc ghi trong ô.

A \ B	0	1
0	0	1
1	1	1

Bìa các nô
 $f(A,B) = A + B$

1.1.2. Một số tính chất của hàm nhân, cộng, phủ định:

- Tồn tại phần tử trung tính duy nhất cho phép "nhân", phép "cộng".

$A + 0 = A$; 0 - Phần tử trung tính cho phép tính "cộng".

$A \cdot 1 = A$; 1 - Phần tử trung tính cho phép "nhân".

- Hoán vị: $A + B = B + A$; $A \cdot B = B \cdot A$.

- Kết hợp $(A + B) + C = A + (B + C) = (A + C) + B$

$(A \cdot B) \cdot C = A \cdot (B \cdot C) = (A \cdot C) \cdot B$

- Phân phối: $A \cdot (B + C) = A \cdot B + A \cdot C$

- Không có số mũ, không có hệ số.

$A + A + \dots + A = A$; $A \cdot A \dots A = A$.

- Bù: $\overline{\overline{A}} = A$; $A + \overline{A} = 1$; $A \cdot \overline{A} = 0$

* Định lý Demorgan:

Tr-ờng hợp tổng quát: $\overline{f[x_i, \bullet, +]} = f[\overline{x_i}, +, \bullet]$

Thí dụ: $\overline{X + Y} = \overline{X} \cdot \overline{Y}$; $\overline{X \cdot Y} = \overline{X} + \overline{Y}$

(Đảo của một tổng bằng tích các đảo, đảo của một tích bằng tổng các đảo)

1.1.3. Biểu diễn giải tích các hàm logic

Với các kí hiệu hàm, biến và các phép tính giữa chúng. Có hai dạng giải tích đ-ợc sử dụng là.

+ Dạng tuyến: Hàm đ-ợc cho d-ới dạng tổng của tích các biến.

+ Dạng hội: Hàm đ-ợc cho d-ới dạng tích của tổng các biến.

+ Dạng tuyến chính quy: Nếu mỗi số hạng chứa đầy đủ mặt các biến.

+ Dạng tuyến không chính quy: Chỉ cần ít nhất một số hạng chứa không đầy đủ mặt các biến.

+ Hội chính quy: Nếu mỗi thừa số chứa đầy đủ mặt các biến.

+ Hội không chính quy: chỉ cần ít nhất một thừa số không chứa đầy đủ mặt các biến.

Thí dụ: $f(X,Y,Z) = \bar{X}.\bar{Y}.\bar{Z} + \bar{X}YZ + \bar{X}YZ + XYZ$ (tuyến chính quy)

$f(X,Y,Z) = \bar{X}.\bar{Y} + \bar{X}\bar{Y}Z + \bar{X}YZ + XZ$ (tuyến không chính quy)

$f(x,y,z) = (X + Y + Z).(X + \bar{Y} + Z).(X + \bar{Y} + \bar{Z})$. (hội chính quy).

$f(x,y,z) = (X + Y + Z).(Y + Z).(Z + \bar{Y} + \bar{X})$. (hội không chính quy).

a. Biểu diễn hàm dạng tuyến chính quy

Nguyên tắc :

- Giá trị của hàm thành phần chỉ nhận giá trị một.
- Số hạng là tổng của tích các biến. $Z = A.B.C + \bar{A}.\bar{B}.\bar{C}$
- Nếu giá trị của hàm thành phần bằng không ta loại số hạng đó.
- Chỉ quan tâm đến các tổ hợp biến tại đó hàm thành phần nhận trị "1".
- Số số hạng bằng số lần hàm thành phần nhận trị "1".
- Trong biểu thức logic các biến nhận trị "1" giữ nguyên, biến nhận trị "0" ta lấy phủ định.

Thí dụ : Cho hàm logic dạng tuyến nh- sau:

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

Tại các tổ hợp biến 1, 2, 3, 5, 7 của biến vào hàm nhận trị "1")

STT	A	B	C	Z = F(A,B,C)
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)
5	1	0	1	1 F(1,0,1)
7	1	1	1	1 F(1,1,1)

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

b. Biểu diễn hàm dạng hội chính quy

Nguyên tắc:

- Giá trị của hàm thành phần chỉ nhận giá trị không.
- Số hạng là tích của tổng các biến tổng các biến. $Z = (A + B + \bar{C}).(A + \bar{B} + C)$
- Nếu giá trị của hàm thành phần bằng giá một, thì thừa số đó bị loại bỏ.
- Hàm chỉ quan tâm đến các tổ hợp biến tại đó hàm thành phần nhận trị "0".
- Số thừa số bằng số lần hàm thành phần nhận trị "0" .

- Trong biểu thức logic các biến nhận trị "0" giữ nguyên, các biến nhận trị "1" ta lấy phủ định.

Thí dụ : Cho hàm logic dạng hội nh- sau:

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

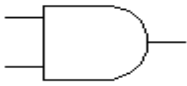
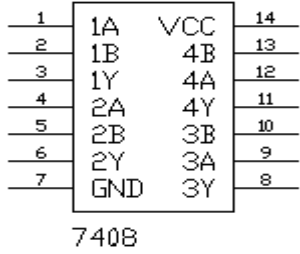
Tại các tổ hợp biến 0, 4, 6 hàm logic nhận trị "0"

STT	A	B	C	Z = F(A,B,C)
0	0	0	0	0 F(0,0,0)
4	1	0	0	0 F(1,0,0)
6	1	1	0	0 F(1,1,0)

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

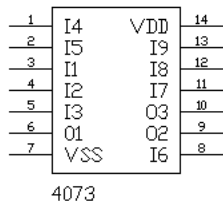
1.2. Các hàm logic cơ bản

1.2.1 Hàm VÀ - AND

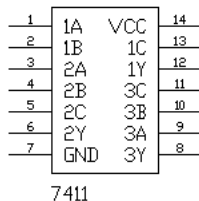
Phương trình	Bảng chân lý	Ký hiệu và sơ đồ chân															
$Y=A.B$	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>Y</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	A	B	Y	0	0	0	0	1	0	1	0	0	1	1	1	  <p>7408</p>
A	B	Y															
0	0	0															
0	1	0															
1	0	0															
1	1	1															

Đối với hàm VÀ giá trị của hàm chỉ bằng 1 khi các biến của nó đều bằng 1; hay chỉ cần có một biến bằng 0 hàm sẽ có giá trị bằng 0

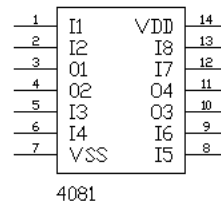
Các IC AND thông dụng



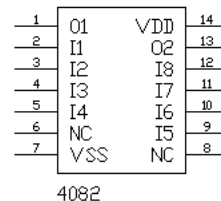
AND 3 lối vào



AND 3 lối vào

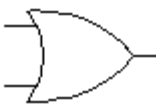
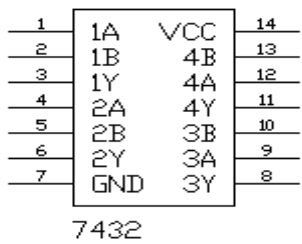


AND 2 lối vào



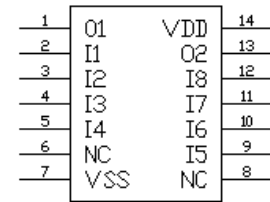
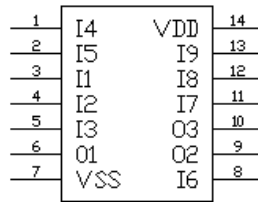
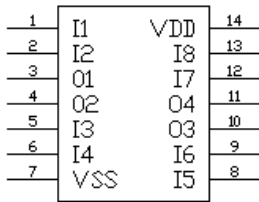
AND 4 lối vào

1.2.2 Hàm HOẶC – OR

Phương trình	Bảng chân lý	Ký hiệu và sơ đồ chân															
$Y=A+B$	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>Y</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	A	B	Y	0	0	0	0	1	1	1	0	1	1	1	1	  <p>7432</p>
A	B	Y															
0	0	0															
0	1	1															
1	0	1															
1	1	1															

Đối với hàm HOẶC giá trị của hàm chỉ bằng 0 khi các biến của nó đều bằng 0; hay chỉ cần có một biến bằng 1 hàm sẽ có giá trị bằng 1

Các IC OR thông dụng khác

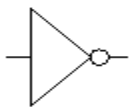
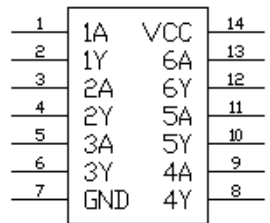


AND 2 lối vào

AND 3 lối vào


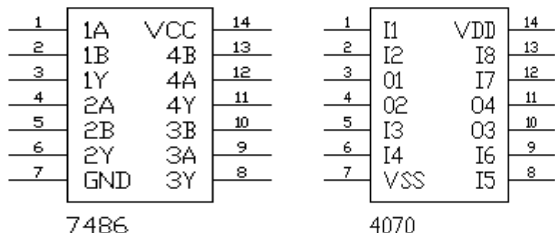
AND 4 lối vào

1.2.3 Hàm ĐẢO - NOT

Phương trình	Bảng chân lý	Ký hiệu và sơ đồ chân						
$Y=\bar{A}$	<table border="1"> <thead> <tr> <th>A</th> <th>Y</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	Y	0	1	1	0	  <p>7404</p>
A	Y							
0	1							
1	0							


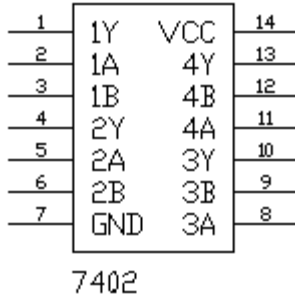
Đối với hàm NOT giá trị của hàm sẽ là đảo của giá trị biến. Khi biến có giá trị bằng 0 thì hàm bằng 1 ngược lại khi biến bằng 1 thì hàm có giá trị bằng 0.

1.2.4. Hàm Hoặc tuyệt đối - XOR

Phương trình	Bảng chân lý	Ký hiệu và sơ đồ chân															
$Y = \bar{A}B + A\bar{B}$	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>Y</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	B	Y	0	0	0	0	1	1	1	0	1	1	1	0	  <p>The diagram shows two XOR gate ICs. The 7486 has pins 1A, 1B, 1Y, 2A, 2B, 2Y, GND, 4A, 4B, 4Y, 3A, 3B, 3Y, VCC, and 8. The 4070 has pins I1, I2, O1, O2, I3, I4, VSS, VDD, I8, I7, O4, O3, I6, I5, 14, 13, 12, 11, 10, 9, and 8.</p>
A	B	Y															
0	0	0															
0	1	1															
1	0	1															
1	1	0															

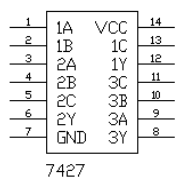
Ta thấy giá trị của hàm sẽ bằng 1 khi các biến có giá trị khác nhau. Ngược lại giá trị của hàm có giá trị bằng 0 khi giá trị của các biến là bằng nhau (cùng bằng 0 hay 1)

1.2.5 Hàm hoặc đảo - NOR

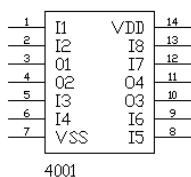
Phương trình	Bảng chân lý	Ký hiệu và sơ đồ chân															
$Y = \overline{A + B}$	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>Y</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	B	Y	0	0	1	0	1	0	1	0	0	1	1	0	  <p>The diagram shows a 7402 NOR gate IC with pins 1Y, 1A, 1B, 2Y, 2A, 2B, GND, 4Y, 4A, 4B, 3A, 3B, 3A, VCC, and 14, 13, 12, 11, 10, 9, 8.</p>
A	B	Y															
0	0	1															
0	1	0															
1	0	0															
1	1	0															

Đối với hàm NOR giá trị của hàm sẽ bằng 1 khi toàn bộ giá trị của biến bằng 0. Ngược lại, một trong các giá trị của biến bằng 1 giá trị của hàm có giá trị bằng 0. Hay nói khác đi nó là hàm đảo của hàm OR.

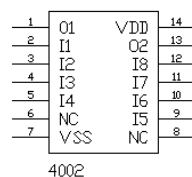
Một số IC NOR khác



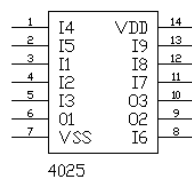
7427



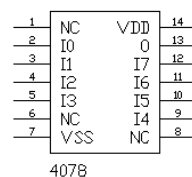
4001



4002



4025



4078

NOR 3 lối vào

NOR 2 lối vào

NOR 4 lối vào

NOR 3 lối vào

NOR 8 lối vào

1.2.6 Hàm Và đảo - NAND

Phương trình	Bảng chân lý	Ký hiệu và sơ đồ chân															
$Y = \overline{A \cdot B}$	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>Y</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	B	Y	0	0	1	0	1	1	1	0	1	1	1	0	<p>Pinout for 7400: 1: 1A, 2: 1B, 3: 1Y, 4: 2A, 5: 2B, 6: 2Y, 7: GND, 8: 3Y, 9: 3A, 10: 3B, 11: 4Y, 12: 4A, 13: 4B, 14: VCC.</p>
A	B	Y															
0	0	1															
0	1	1															
1	0	1															
1	1	0															

Đối với hàm NAND giá trị của hàm sẽ bằng 0 khi toàn bộ giá trị của biến bằng 1. Ngược lại, một trong các giá trị của biến bằng 0 giá trị của hàm có giá trị bằng 1. Hay nói khác đi nó là hàm đảo của hàm AND

1.2.7 Hàm XNOR

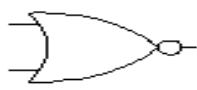
phương trình	Bảng chân lý	Ký hiệu và sơ đồ chân															
$Y = A \oplus B = \overline{A}B + A\overline{B}$	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>Y</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	A	B	Y	0	0	1	0	1	0	1	0	0	1	1	1	<p>Pinout for 74266: 1: 1A, 2: 1B, 3: 1Y, 4: 2Y, 5: 2A, 6: 2B, 7: GND, 8: 3A, 9: 3B, 10: 3Y, 11: 4Y, 12: 4A, 13: 4B, 14: VCC.</p>
A	B	Y															
0	0	1															
0	1	0															
1	0	0															
1	1	1															

Đối với hàm XNOR nếu các giá trị của biến là bằng nhau (đều bằng 1 hay bằng 0) thì giá trị của hàm sẽ là 1 ngược lại hàm có giá trị bằng 0.

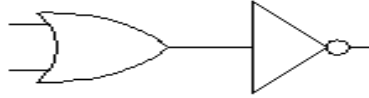
Thực chất 7 hàm trên chỉ có 3 hàm đầu tiên là các hàm cơ bản, 4 hàm còn lại có thể xây dựng từ 3 hàm trên.

Ví dụ:

+ Hàm NOR là sự kết hợp của hàm NOR và hàm NOT.

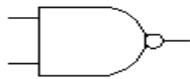


Hàm NOR

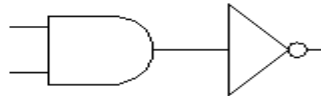


Sự kết hợp của hàm NOR và NOT

+ Hàm NAND là sự kết hợp của hàm AND và hàm NOT



Hàm NAND

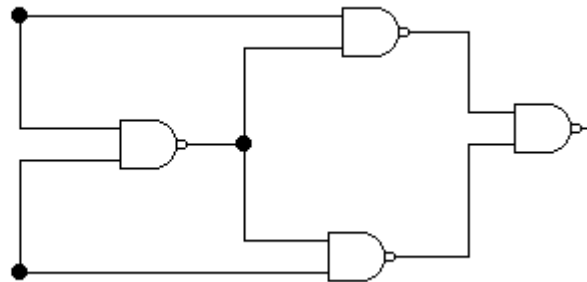


Sự kết hợp của hàm AND và NOT

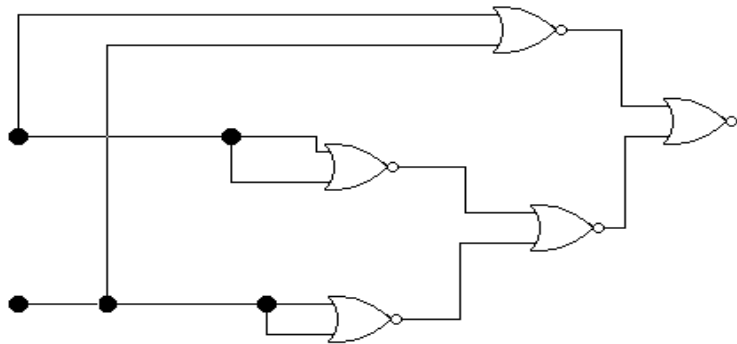
+ Hàm XOR là sự kết hợp của các hàm NAND hoặc hàm NOR



Hàm XOR



Sự kết hợp của hàm NAND



Hàm XOR Sự kết hợp của hàm NOR

Tuy nhiên việc tích hợp các mạch cơ bản để tạo ra các hàm khác sẽ rất hữu ích trong việc thiết kế mạch. Nó sẽ làm giảm đi số lượng IC trên một bo mạch, dẫn đến làm giảm chi phí cho mạch vì một IC XOR (74LS86) có chứa 4 phần tử XOR cũng có giá thành như một IC NAND hay IC NOR.

1.3. Tối thiểu hóa các hàm logic

Một hàm logic có thể có vô số cách biểu diễn giải tích t-ơng đ-ơng. Tuy nhiên chỉ tồn tại 1 cách gọn nhất tối -u về số biến, số số hạng hay thừa số và đ-ợc gọi là tối giản. việc tối giản hàm logic mang ý nghĩa quan trọng về ph-ơng diện kinh tế, kỹ thuật. Để tối thiểu hoá các hàm logic ng-ời ta th-ờng dùng ph-ơng pháp đại số và ph-ơng pháp bìa các nô.

1.3.1. Ph-ơng pháp đại số:

Biến đổi biểu thức logic dựa vào các tính chất của đại số **Boole**.

Thí dụ : $A.B + \bar{A}.B = B$; $A+A.B = A$; $A + \bar{A}.B = A + B$.

Ta chứng minh các đẳng thức trên, theo tính chất đối ngẫu:

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

$$A + A.B = A \Leftrightarrow A.(A + B) = A.$$

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

Quy tắc 1:

Nhóm các số hạng có thừa số chung.

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

Quy tắc 2:

Đ- a số hạng đã có vào biểu thức logic.

$$\begin{aligned} & A.B.C + \bar{A}.B.C + A.\bar{B}.C + A.B.\bar{C} = \\ & = A.B.C + \bar{A}.B.C + A.\bar{B}.C + A.B.C + A.B.\bar{C} + A.B.C \\ & = B.C.(A + \bar{A}) + A.C.(B + \bar{B}) + A.B.(C + \bar{C}) = B.C + A.C + A.B \end{aligned}$$

Quy tắc 3:

Có thể loại các số hạng thừa.

$$\begin{aligned} & A.B + \bar{B}.C + A.C = A.B + \bar{B}.C + A.C (B + \bar{B}). \\ & = A.B + \bar{B}.C + A.B.C + A.\bar{B}.C \\ & = A.B + \bar{B}.C \text{ (loại } A.C) \end{aligned}$$

Ví dụ : Hãy tối giản hàm sau bằng phương pháp đại số:

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

Giải:

Từ yêu cầu của bài ta có bảng chân lý nh- sau

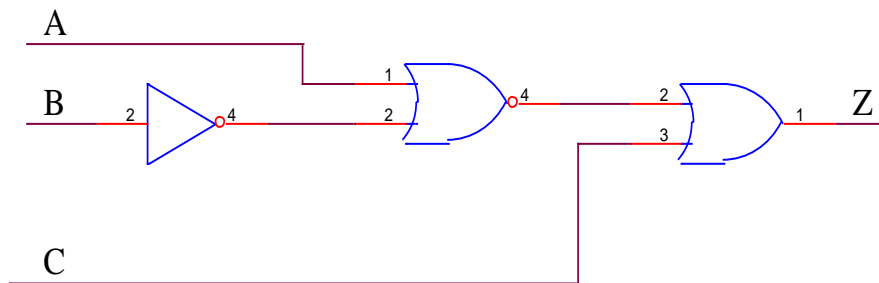
STT	A	B	C	Z = F(A,B,C)
1	0	0	1	1 F(0,0,1)
2	0	1	0	1 F(0,1,0)
3	0	1	1	1 F(0,1,1)
5	1	0	1	1 F(1,0,1)
7	1	1	1	1 F(1,1,1)

Từ bảng chân lý ta có phương trình trạng thái như sau:

$$Z = \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}C + ABC = \bar{A}C(\bar{B} + B) + \bar{A}B\bar{C} + AC(\bar{B} + B)$$

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

Mạch logic thực hiện:



1.3.2. Phương pháp bảng Karnaugh (biểu các nô)

a. Cấu tạo:

- Gồm 1 đồ hình các ô vuông, hàm có n biến bảng có 2^n ô (1 biến - 2 ô, 2 biến - 4 ô, 3 biến - 8 ô)
- Thứ tự của các ô do giá trị tổ hợp biến quy định
- Hai ô kề nhau, hoặc đối xứng chỉ khác nhau 1 giá trị của biến.
- Giá trị của hàm tương ứng với tổ hợp biến được ghi ngay trong ô đó.
- Các ô tại đó giá trị của hàm không xác định được đánh bằng dấu "X".

b. Nguyên tắc tối giản hàm logic trên biểu các nô

- Thực hiện nhóm các ô tại đó hàm nhận trị "1" hoặc "0" kề nhau hoặc đối xứng, số ô trong một nhóm dán phải là số lũy thừa của 2 (khi viết hàm dạng tuyển ta nhóm các ô có giá trị "1", dạng hội nhóm các ô có giá trị "0").

- Trong một nhóm dán các biến có trị thay đổi ta loại, các biến có trị không đổi giữ nguyên, điều này có nghĩa là số ô trong nhóm dán càng nhiều thì số biến bị loại càng tăng (2 ô - loại 1 biến, 4 ô - loại 2 biến ... 2^m ô - loại m biến).

- Biểu thức logic có số số hạng hay thừa số chính bằng số nhóm dán. Khi viết hàm logic d- ối dạng tuyển các biến còn lại nhận trị "1" ta giữ nguyên, nhận trị "0" ta lấy phủ định, khi viết hàm logic d- ối dạng hội thì ngược lại.

- Một ô có thể tham gia vào nhiều nhóm dán.

- Các ô tại đó giá trị hàm không xác định ta coi tại ô đó hàm có thể lấy giá trị "1" hoặc "0" tùy từng trường hợp cụ thể.

* Chú ý: Phương pháp tối giản hàm logic trên dựa vào các nhóm chỉ thích hợp với hàm có số biến ≤ 6 . Trường hợp hàm có số biến lớn hơn 6, bảng các nhóm rất phức tạp.

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

4 cột 2 hàng (3 hàm biến)
biến

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

2 cột 4 hàng 3 hàm

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

4 hàng 4 cột (3 biến)

Ví dụ 1:

Cho hàm số : $Y(A, B, C, D) = \sum 0,1,2,4,5,6,8,9,10,14$

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

Giải:

Để thiết kế được mạch logic đầu tiên chúng ta phải lập được bảng chân lý của hàm.

STT	A	B	C	D	F(A,B,C,D)
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	1
15	1	1	1	1	0

Lập bìa các nô tối giản hàm

F		CD			
		00	01	11	10
AB	00	1	1	0	1
	01	1	1	0	1
	11	0	0	0	1
	10	1	1	0	1

Phương trình trạng thái của hàm như sau:

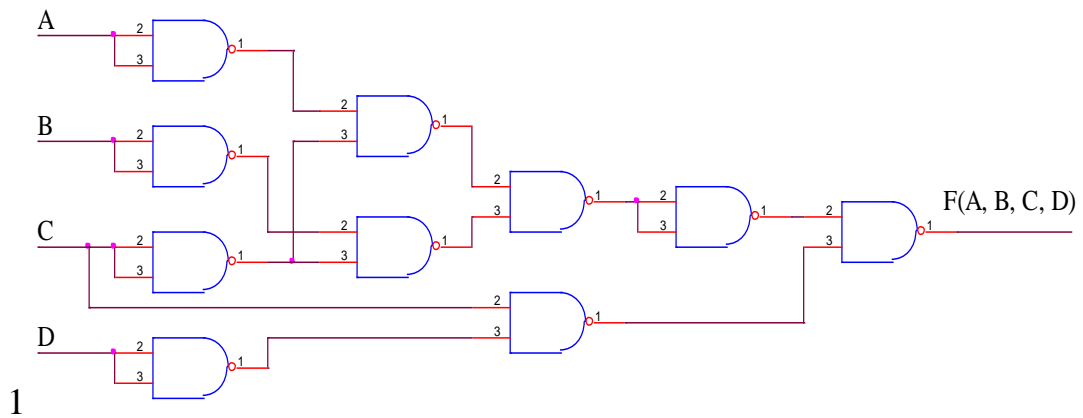
$$F(A, B, C, D) = \bar{A}\bar{C} + \bar{B}\bar{C} + C\bar{D}$$

Xây dựng mạch logic từ phần tử NAND 2 đầu vào

$$F(A, B, C, D) = \bar{A}\bar{C} + \bar{B}\bar{C} + C\bar{D} = \overline{\overline{\bar{A}\bar{C}} + \overline{\bar{B}\bar{C}} + \overline{C\bar{D}}} = \overline{\overline{\bar{A}\bar{C}} \cdot \overline{\bar{B}\bar{C}} \cdot \overline{C\bar{D}}}$$

$$F(A, B, C, D) = \overline{\overline{\bar{A}\bar{C}} \cdot \overline{\bar{B}\bar{C}} \cdot \overline{C\bar{D}}} = \overline{\overline{\bar{A}\bar{C}} \cdot \overline{\bar{B}\bar{C}} \cdot C \cdot D}$$

Sơ đồ mạch logic



Hình 1.25: Sơ đồ mạch logic chỉ dùng phần tử NAND hai đầu vào

Ví dụ 2:

Cho hàm số:

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

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

Giải:

Bảng chân lý của hàm như sau :

STT	A	B	C	D	F(A,B,C,D)
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	1
15	1	1	1	1	0

Lập bìa các nô tối giản hàm:

		F			
		CD			
AB		00	01	11	10
	00	0	0	0	1
	01	1	1	0	1
11		0	0	0	1
10		0	0	0	1

Phương trình trạng thái của hàm:

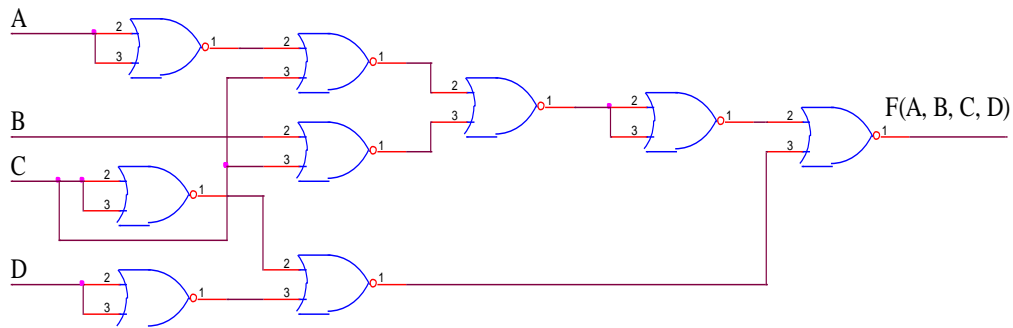
$$F(A, B, C, D) = (\bar{A} + C)(B + C)(\bar{C} + \bar{D})$$

Xây dựng mạch logic từ các phần tử NOR hai đầu vào.

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

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

Sơ đồ mạch logic như sau:



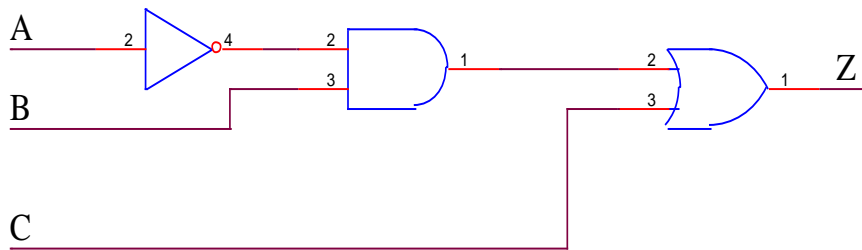
Hình 1.26: Sơ đồ mạch logic chỉ dùng phần tử NOR hai đầu vào

Chương 2: Thiết kế mạch logic tổ hợp

2.1. Mạch logic là gì

Mạch logic là mạch gồm các phân tử logic AND, OR, NOR, NOT, NAND, XOR, XNOR để thực hiện các yêu cầu của bài toán đưa ra. Một mạch logic dù đơn giản hay phức tạp thì kết quả đầu ra của mạch cũng chỉ nhận một trong hai mức logic là “ 0 ” hoặc “ 1 ”.

Vi dụ : Cho mạch logic sau :



Hình 2.1: Mạch logic

2.2. Quy trình thiết kế

Quy trình thiết kế mạch logic như sau:

- + Xây dựng phương trình logic sử dụng các phương trình theo CTT, hay CTH hoặc có thể sử dụng bảng chân lý để biểu diễn
- + Sử dụng bảng karnaugh hoặc các phương pháp đại số để tối thiểu hóa hàm logic hoặc đưa hàm logic về dạng mà dễ thiết kế mạch
- + Thiết kế mạch cho chạy thử
- + Đánh giá tính ổn định của mạch

Thí dụ:

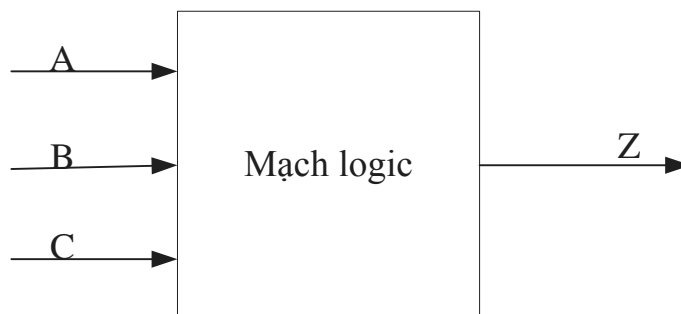
Thiết kế mạch logic thực hiện phép toán sau, dùng các phân tử logic cơ bản

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

Giải:

Phân tích yêu cầu

Mạch của chúng ta gồm có 3 biến đầu vào là A, B, và D và một hàm đầu ra là Z . Ta có sơ đồ tổng quát nh- sau



Hình 2.3: Sơ đồ mô phỏng

Từ yêu cầu của bài ta có bảng trạng thái nh- sau

STT	A	B	C	Z = F(A,B,C)
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)
5	1	0	1	1 F(1,0,1)
7	1	1	1	1 F(1,1,1)

Tối giản hàm để đ- a về hàm tối giản nhất

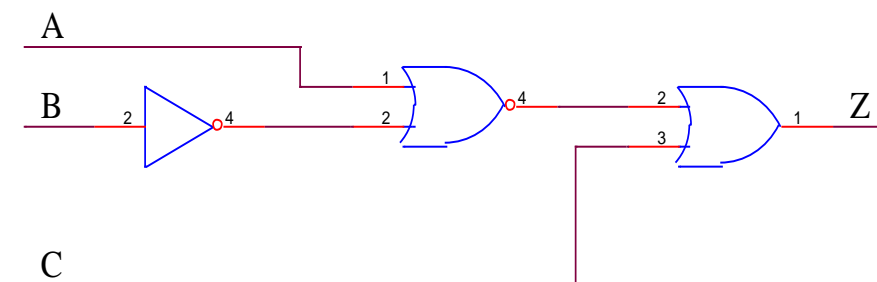
$$Z = \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}C + ABC = \bar{A}C(\bar{B} + B) + \bar{A}B\bar{C} + AC(\bar{B} + B)$$

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

B- óc 4: Vẽ sơ đồ mạch logic thực hiện bài toán

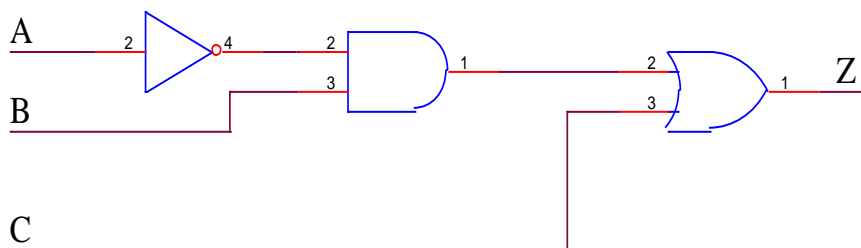
- Xây dựng mạch logic dùng phần tử NOR và OR

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



- Xây dựng mạch từ phần tử OR và AND

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



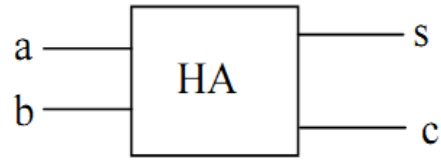
2.3. Thiết kế mạch số học

2.3.1. Thiết kế bộ cộng bán tổng (HA-Half Adder)

Bộ cộng bán tổng thực hiện cộng hai số nhị phân một bit

Quy tắc cộng như sau:

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



Hình 2.4: Sơ đồ mô phỏng

Trong đó:

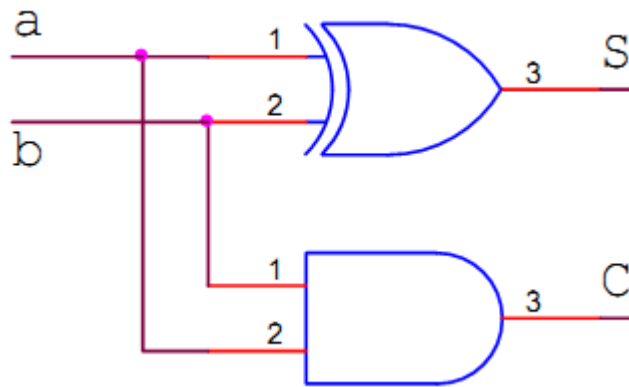
a, b là số cộng, s là tổng của phép cộng, c là số nhớ

Bảng chân lý mô tả hoạt động của mạch và phương trình logic như sau

a	b	s	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

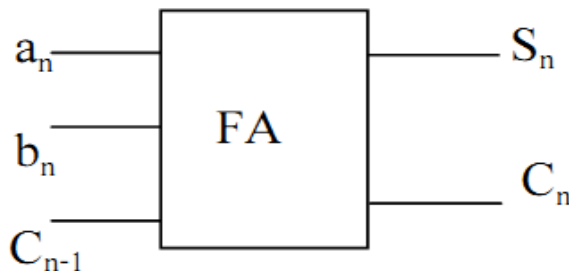
$$s = \bar{a}b + a\bar{b} = a \oplus b \quad c = ab$$

Mạch cộng này chỉ cho phép cộng hai số nhị phân một bit mà không thực hiện cộng hai số nhị phân nhiều bit.



Hình 2.5: Sơ đồ mạch logic cộng hai số nhị phân một bit

2.3.2. Thiết kế mạch cộng toàn phần (FA- Full adder)



Hình 2.6: Sơ đồ mô phỏng mạch

Trong đó

C_{n-1} : Số nhớ của lần cộng trước đó

C_n : Số nhớ của lần cộng hiện tại

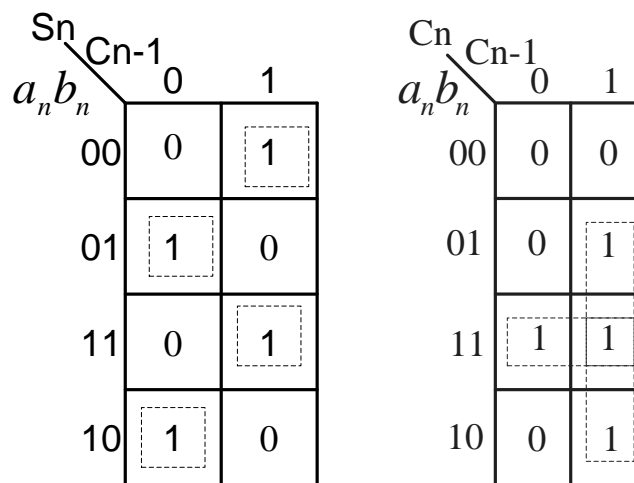
S_n : Tổng hiện tại

Bảng chân lý của mạch cộng toàn phần

a_n	b_n	C_{n-1}	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

Bảng trạng thái

Tối giản hàm đầu ra bằng phương pháp bìa các nô

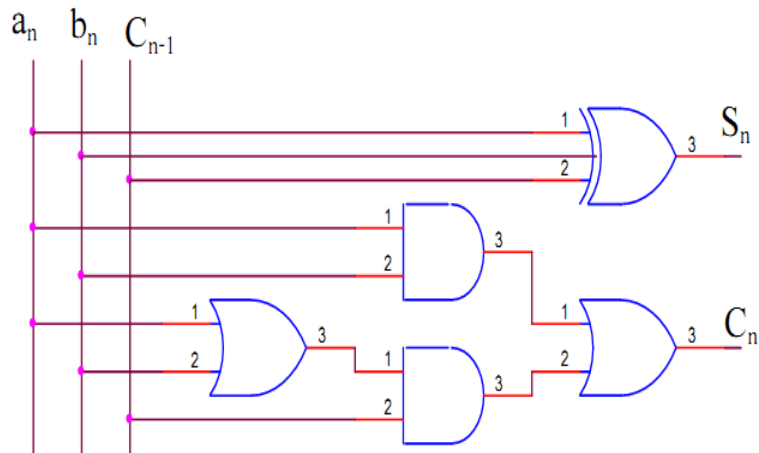


Phương trình trạng thái hàm S_n và C_n

$$S_n = \bar{a}_n \bar{b}_n C_{n-1} + \bar{a}_n b_n \bar{C}_{n-1} + a_n b_n C_{n-1} + a_n \bar{b}_n \bar{C}_{n-1} = a_n \oplus b_n \oplus C_{n-1}$$

$$C_n = C_{n-1} b_n + C_{n-1} a_n + a_n b_n = a_n b_n + C_{n-1} (a_n + b_n)$$

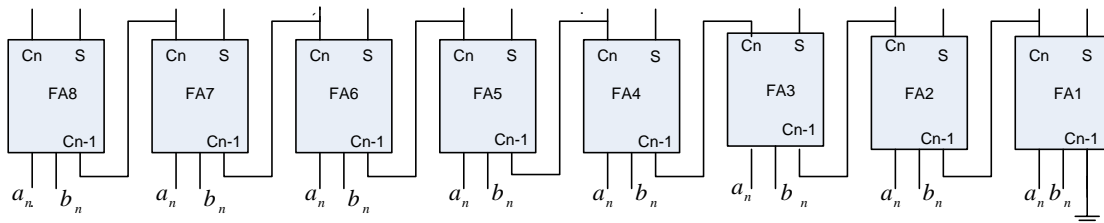
Sơ đồ mạch cộng toàn phần



Hình 2.7: Sơ đồ mạch cộng toàn phần

2.3.3. Mạch công hai số nhị phân 8 bit

Để thực hiện phép cộng hai số nhị phân 8 bit ta sử dụng 8 bộ FA nối tiếp với nhau như sơ đồ dưới đây



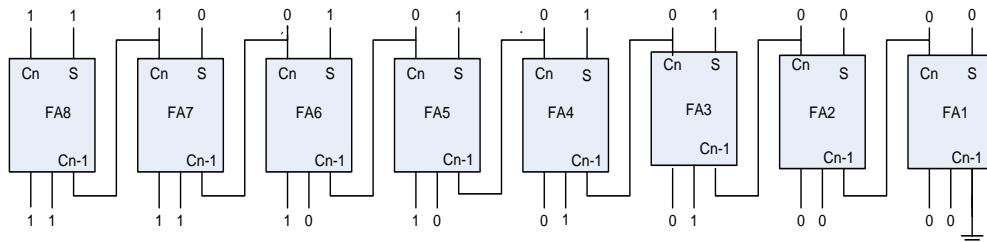
Hình 2.8: Sơ đồ khối mạch cộng hai số nhị phân 8 bit

Theo sơ đồ thiết kế như trên thì chân C_{n-1} của FA đầu tiên (FA có trọng số thấp nhất) được nối với đất vì hai bit thấp nhất khi cộng với nhau sẽ không có bit nhớ của phép cộng trước đó. Trong khi các bit C_{n-1} của FA sau phải được nối với bit tràn C_n (bit nhớ) của các FA trước đó, như vậy kết quả của FA sau không chỉ phụ thuộc vào hai bit đầu vào a_n, b_n mà còn phụ thuộc vào kết quả của FA trước đó, điều này là logic với phép cộng toàn phần hai số nhiều bit.

Ví dụ : Cộng hai số nhị phân 8 bit sau:

$$a_n = 11110000$$

$$b_n = 11001100$$

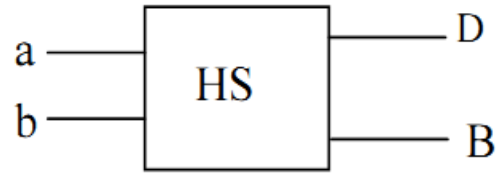


Kết quả phép cộng là: $S_n = 10111100$

2.3.3. Thiết kế bộ bán trừ (bộ trừ bán phần -HS)

Bộ bán trừ thực hiện trừ hai số nhị phân một bit

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



Hình 2.9: Sơ đồ mô phỏng

Trong đó a số bị trừ, b số trừ, D là hiệu, B là số mượn

Bảng chân lý mô tả hoạt động và sơ đồ mạch :

a	b	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

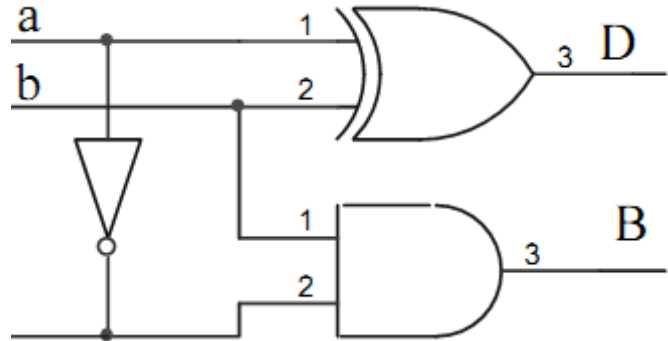
Bảng trạng thái

Phương trình trạng thái

$$D = a \cdot \bar{b} + \bar{a} \cdot b = a \oplus b$$

$$B = \bar{a} \cdot b$$

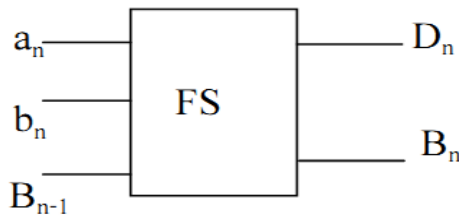
Mạch trừ này chỉ cho phép trừ hai số nhị phân một bit mà không thực hiện trừ hai số nhị phân nhiều bit.



Hình 2.10: Sơ đồ mạch bán trừ

2.3.4. Thiết kế bộ trừ toàn phần (FS- Full Subtractor)

Sơ đồ mô phỏng :



Hình 2.11: Sơ đồ mô phỏng

Trong đó :

B_{n-1} : Số mượn của lần trừ trước đó

B_n : Số mượn của lần trừ hiện tại

D_n : Hiệu số hiện tại

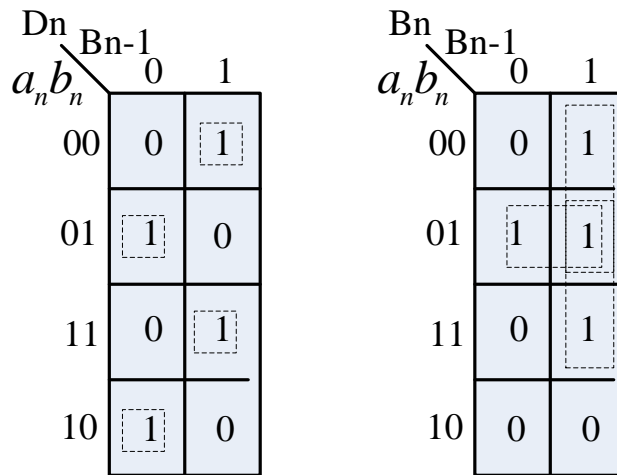
a_n : Số bị trừ

b_n : Số trừ

Bảng chân lý mô tả hoạt động của mạch:

a_n	b_n	B_{n-1}	D_n	B_n
0	0	0	0	0
0	0	1	1	1

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

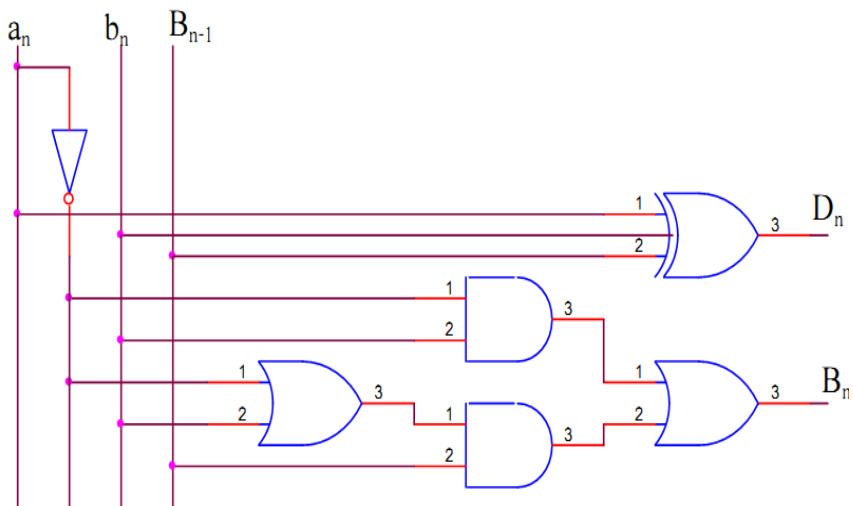


Ta có phương trình trạng thái các hàm đầu ra như sau :

$$D_n = \bar{a}_n \bar{b}_n B_{n-1} + a_n b_n B_{n-1} + \bar{a}_n b_n \bar{B}_{n-1} + a_n \bar{b}_n \bar{B}_{n-1} = a_n \oplus b_n \oplus B_{n-1}$$

$$B_n = \bar{a}_n B_{n-1} + b_n B_{n-1} + \bar{a}_n b_n = \bar{a}_n b_n + B_{n-1}(\bar{a}_n + b_n)$$

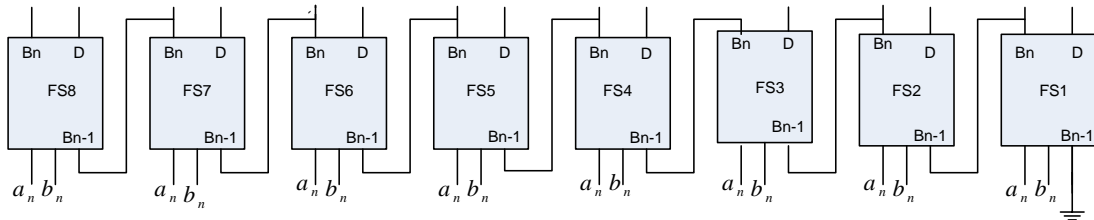
Sơ đồ mạch logic như sau:



Hình 2.12: Sơ đồ mạch trừ toàn phần

2.3.5. Mạch trừ hai số nhị phân 8 bit

Để trừ hai số nhị phân 8 bit ta ghép 8 bộ trừ đầy đủ với nhau ta được sơ đồ như sau:



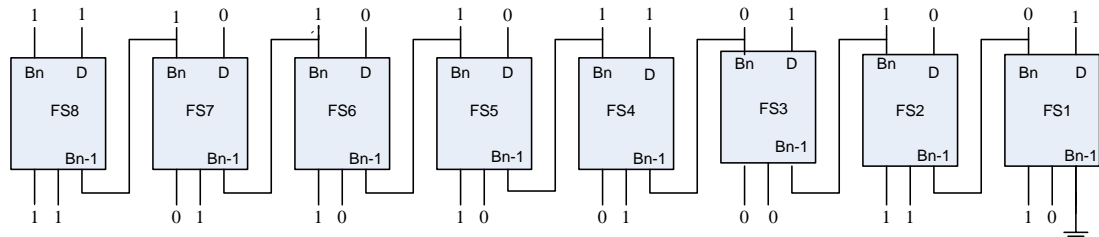
Hình 2.13: Sơ đồ khối mạch trừ hai số nhị phân 8 bit

Theo sơ đồ thiết kế như trên thì chân B_{n-1} của FS đầu tiên (FS có trọng số thấp nhất) được nối với đất vì hai bit thấp nhất khi cộng với nhau sẽ không có bit nhớ của phép cộng trước đó. Trong khi các bit B_{n-1} của FA sau phải được nối với bit tràn B_n (bit nhớ) của các FA trước đó, như vậy kết quả của FS sau không chỉ phụ thuộc vào hai bit đầu vào a_n, b_n mà còn phụ thuộc vào kết quả của FS trước đó, điều này là logic với phép trừ toàn phần hai số nhiều bit.

Ví dụ : trừ hai số nhị phân 8 bit sau:

$$a_n = 10110011$$

$$b_n = 11001010$$



Kết quả phép trừ là : $D_n = 10001101$

2.4. Thiết kế mạch so sánh

2.4.1. Mạch so sánh 1 bit

Là mạch thực hiện chức năng so sánh hai số nhị phân 1 bit .

Xét hai số nhị phân 1 bit a và b. Có các trường hợp sau đây:

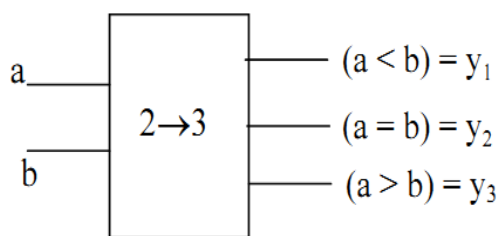
$$+ a = 0, b = 0 \Rightarrow a = b.$$

$$+ a = 1, b = 1 \Rightarrow a = b.$$

$$+ a = 0, b = 1 \Rightarrow a < b.$$

$$+ a = 1, b = 0 \Rightarrow a > b.$$

Về phương diện mạch điện, mạch so sánh 1 bit có hai ngõ vào và 3 ngõ ra. Các ngõ vào a và b là các bit cần so sánh. Các ngõ ra thể hiện kết quả so sánh: $y_1(a < b)$, $y_2(a = b)$, $y_3(a > b)$ sơ đồ khối và bảng chân lý mạch so sánh như sau:



Hình 2.14: Sơ đồ mô phỏng

a	b	y ₁	y ₂	y ₃
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

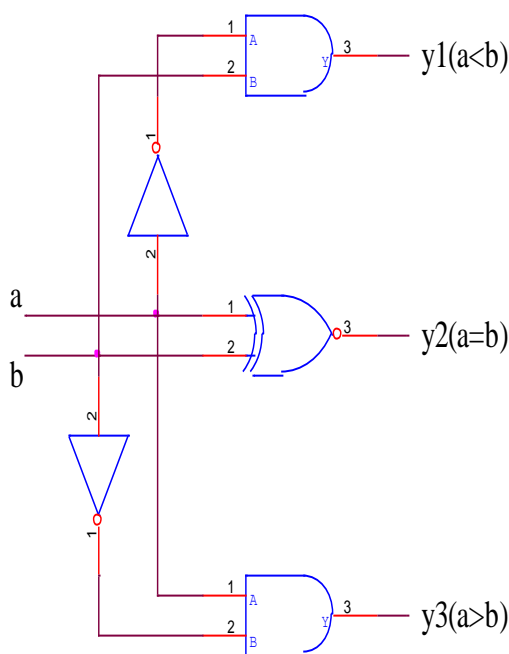
Bảng chân lý

Từ bảng trạng thái ta có phương trình trạng thái và sơ đồ mạch logic như sau:

$$y_1 = \bar{a} \cdot b$$

$$y_2 = \bar{a} \cdot \bar{b} + a \cdot b = \overline{a \oplus b}$$

$$y_3 = a \cdot \bar{b}$$

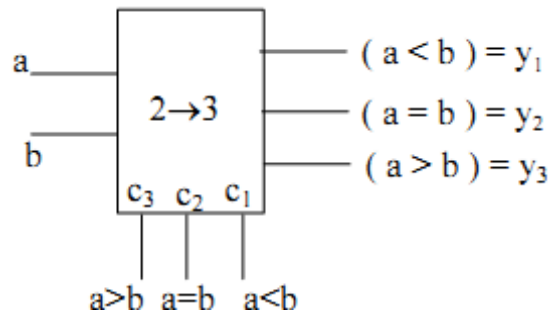


Hình 2.15: Sơ đồ mạch so sánh 1 bit

2.4.2. Mạch so sánh hai số 8 bit

Để thiết kế mạch so sánh hai số 8 bit ta sẽ thiết kế mạch so sánh hai số 1 bit. Dùng các phần tử logic ta dễ dàng thiết kế được mạch so sánh 1 bit như trên. Tuy nhiên mạch so sánh trên không thể phát triển để so sánh nhiều bit được. Muốn so sánh hai số nhiều bit ta phải tuân theo trình tự so sánh từ bit cao nhất trước (bit có nhiều ý nghĩa nhất). Nếu số nào có bit cao hơn thì số đó sẽ lớn hơn và kết thúc việc so sánh, nếu hai bit có trong số cao nhất bằng nhau thì sẽ so sánh hai số có trọng số thấp hơn, cứ như vậy cho đến bit thấp nhất, hai số bằng nhau nếu tất cả các

bít tung ứng của hai số đều bằng nhau. Để so sánh hai số 8 bit ta phải thêm các bit điều khiển vào mạch so sánh hai số một bit, gọi là mạch so sánh 1 bit đầy đủ. Ta có sơ đồ khối như sau:



Hình 2.16: Sơ đồ mô phỏng bộ so sánh hai số 1bit đầy đủ

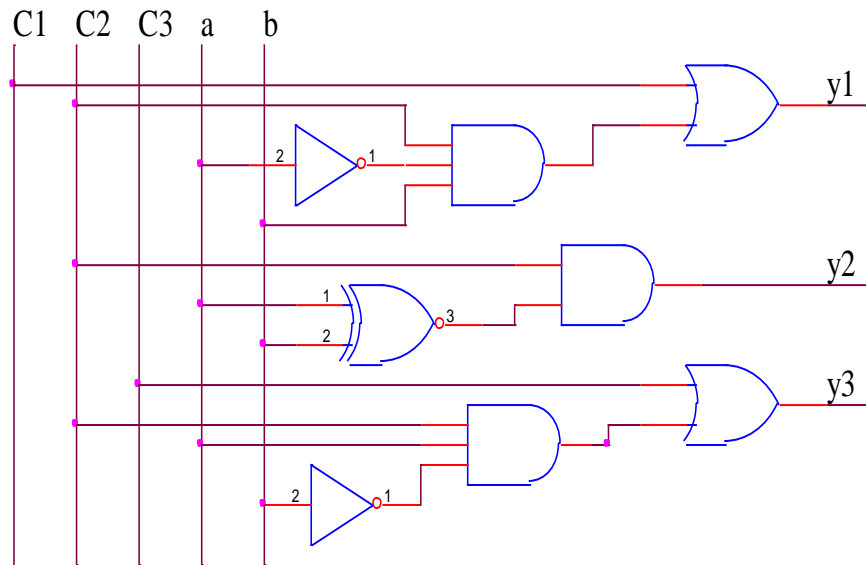
Bảng trạng thái mô tả hoạt động như sau:

Lối vào điều khiển			Lối vào dữ liệu		Lối ra		
c_3	c_2	c_1	a	b	y_3	y_2	y_1
a>b	a=b	a<b			a>b	a=b	a<b
1	0	0	x	x	1	0	0
0	0	1	x	x	0	0	1
0	1	0	0	0	0	1	0
0	1	0	0	1	0	0	1
0	1	0	1	0	1	0	0
0	1	0	1	1	0	1	0

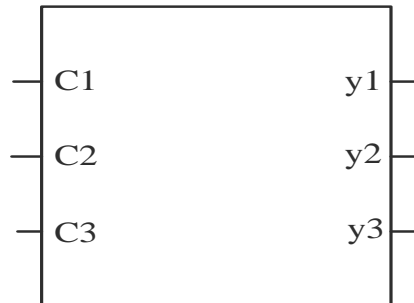
Phương trình trạng thái bộ so sánh hai bit đầy đủ như sau:

$$y_1 = c_1 + c_2 \bar{a}b; \quad y_2 = c_2 \bar{a}\bar{b} + c_2 ab = c_2 (a \oplus b); \quad y_3 = c_3 + c_2 a\bar{b}$$

Sơ đồ mạch logic bộ so sánh hai bit đầy đủ:

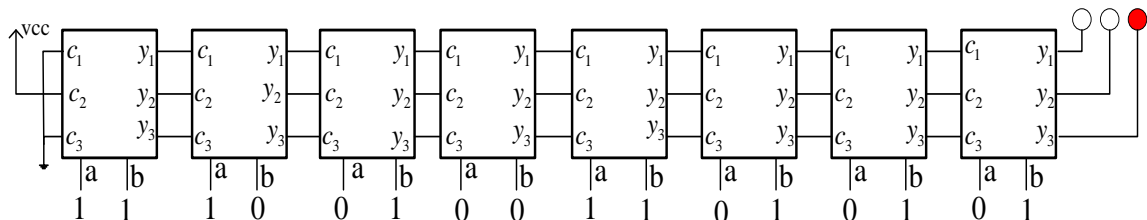


Hình 2.17: Sơ đồ mạch logic bộ so sánh hai bit đầy đủ
 Từ sơ đồ mạch logic trên ta có sơ đồ khối bộ so sánh hai số một bit như sau:



Hình 2.18: Sơ đồ khối bộ so sánh 1 bit đầy đủ

Để có bộ so sánh 2 số 8 bit ta phải ghép 8 bộ so sánh 1 bit đầy đủ lại với nhau ta có sơ đồ như sau :



Hình 2.19: Sơ đồ bộ so sánh hai số nhị phân 8 bit

So sánh hai số : $a=11001000$, $b=10101111$, ta thấy $a > b$ nếu đèn nối với y_3 sáng chứng tỏ mạch ta thiết kế là đúng.

2.5. Thiết kế mạch dồn kênh

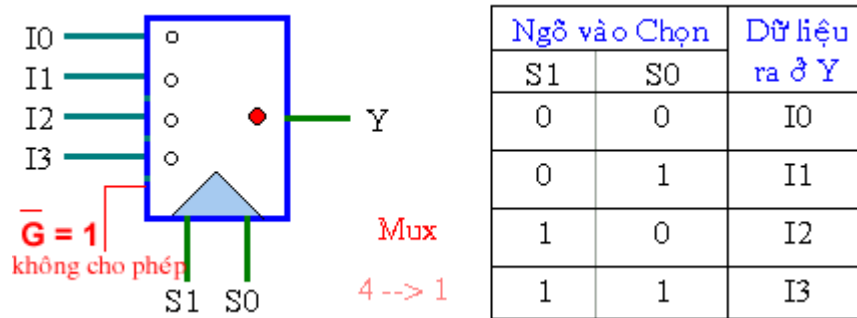
2.5.1. Khái niệm

Mạch dồn kênh hay còn gọi là mạch ghép kênh, đa hợp (Multiplexer-MUX) là 1 dạng mạch tổ hợp cho phép chọn 1 trong nhiều đường đường vào song song (các kênh vào) để đưa tới 1 đường ra (gọi là kênh truyền nối tiếp). Việc chọn đường nào trong các đường đường vào do các đường chọn quyết định. Ta thấy

MUX hoạt động như 1 công tắc nhiều vị trí được điều khiển bởi mã số. Mã số này là dạng số nhị phân, tùy tổ hợp số nhị phân này mà ở bất kì thời điểm nào chỉ có 1 đường vào được chọn và cho phép đưa tới đường ra.

Các mạch dồn kênh thường gặp là 2 sang 1, 4 sang 1, 8 sang 1, ...Nói chung là từ 2^n sang 1. Mục tiếp theo sẽ phân tích và thiết kế mạch dồn kênh 4 sang 1

2.5.2. Mạch dồn kênh 4 sang 1



Hình 2.20: Mạch dồn kênh 4 sang 1 và bảng hoạt động

Mạch trên có 2 đường điều khiển chọn là S0 và S1 nên chúng tạo ra 4 trạng thái logic. Mỗi một trạng thái sẽ cho phép 1 đường vào I nào đó qua để truyền tới đường ra Y. Như vậy tổng quát nếu có 2^n đường vào song song thì phải cần n đường điều khiển chọn.

Cũng nói thêm rằng, ngoài những đường như ở trên, mạch thường còn có thêm đường G: được gọi là đường vào cho phép (enable) hay xung đánh dấu (strobe). Mạch tổ hợp có thể có 1 hay nhiều đường vào cho phép và nó có thể tác động mức cao hay mức thấp. Như mạch dồn kênh ở trên, nếu có thêm 1 đường cho phép G tác động ở mức thấp, tức là chỉ khi $G = 0$ thì hoạt động dồn kênh mới diễn ra còn khi $G = 1$ thì bất chấp các đường vào song song và các đường chọn, đường ra vẫn giữ cố định mức thấp (có thể mức cao tùy dạng mạch)

Như vậy khi $G = 0$

$S_1S_0 = 00$, dữ liệu ở I0 sẽ đưa ra ở Y

$S_1S_0 = 01$, dữ liệu ở I1 sẽ đưa ra ở Y

$S_1S_0 = 10$, dữ liệu ở I2 sẽ đưa ra ở Y

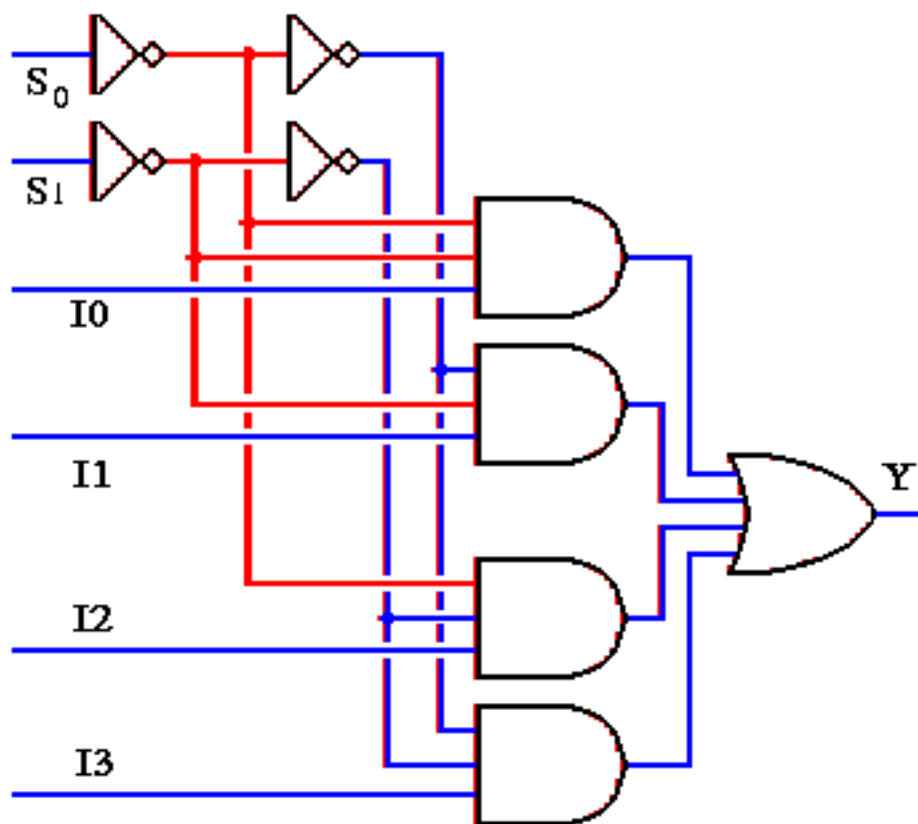
$S_1S_0 = 11$, dữ liệu ở I3 sẽ đưa ra ở Y

do đó biểu thức logic của mạch khi có thêm đường G là

$$Y = G\bar{S}_1\bar{S}_0I_0 + G\bar{S}_1S_0I_1 + GS_1\bar{S}_0I_2 + GS_1S_0I_3$$

Ta có thể kiểm chứng lại biểu thức trên bằng cách: từ bảng trạng thái ở trên, viết biểu thức logic rồi rút gọn (có thể dùng phương pháp rút gọn dùng bìa Karnaugh. Nhận thấy rằng tổ hợp 4 cổng NOT để đưa 2 đường điều khiển chọn S_0, S_1 vào các cổng AND chính là 1 mạch mã hoá 2 sang 4, các đường ra mạch mã hoá như là xung mở cổng AND cho 1 trong các đường I ra ngoài. Vậy mạch trên cũng có thể vẽ lại như sau:

Sơ đồ mạch logic của mạch

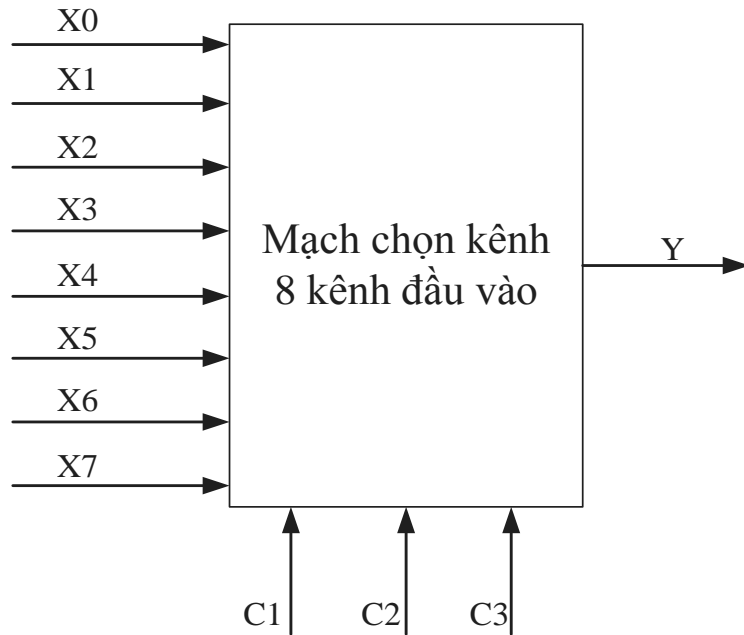


Hình 2.21: Sơ đồ mạch hợp kênh 4 đầu vào một đầu ra

Cũng nói thêm rằng, ngoài những đường như ở trên, mạch thường còn có thêm đường G : được gọi là đường vào cho phép (enable) hay xung đánh dấu (strobe). Mạch tổ hợp có thể có 1 hay nhiều đường vào cho phép và nó có thể tác động mức cao hay mức thấp. Như mạch dồn kênh ở trên, nếu có thêm 1 đường cho phép G tác động ở mức thấp, tức là chỉ khi $G = 0$ thì hoạt động dồn kênh mới diễn ra còn khi $G = 1$ thì bất chấp các đường vào song song và các đường chọn, đường ra vẫn giữ cố định mức thấp (có thể mức cao tùy dạng mạch)

2.5.3. Thiết kế mạch dồn kênh 8 sang 1

Sơ đồ nguyên lý



Hình 2.22: Sơ đồ khối mạch chọn kênh 8 đầu vào 1 đầu ra

Mạch gồm có 8 ngõ vào và một ngõ ra :

- X0, X1, X2, X3, X4, X5, X6, X7 : Các kênh dữ liệu vào
- Y : Kênh dữ liệu đầu ra
- C1, C2, C3 : Các ngõ vào điều khiển

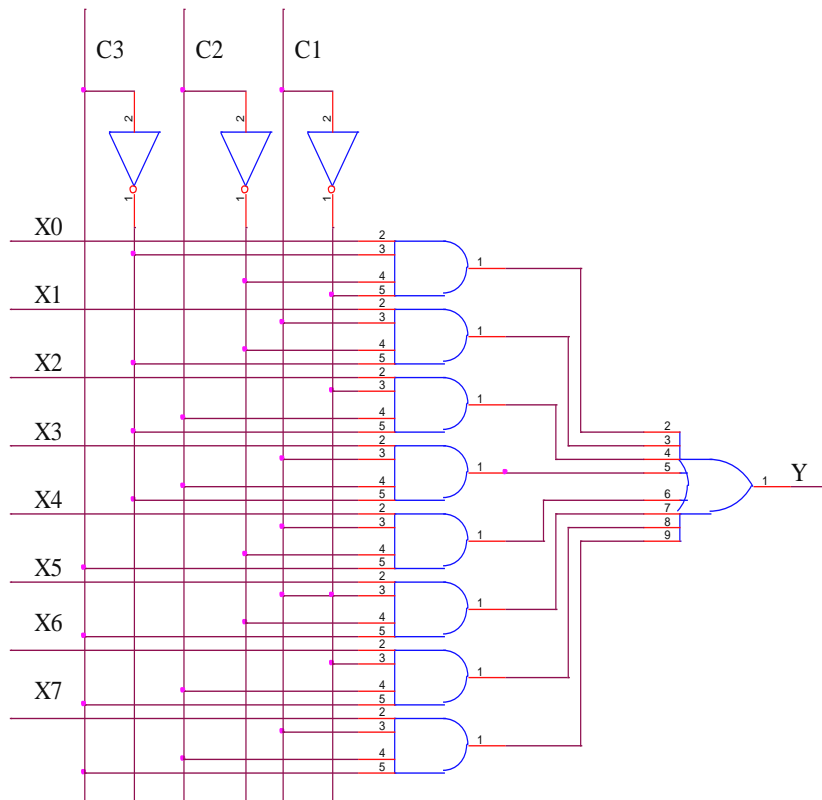
Bảng trạng thái hoạt động

C3	C2	C1	Y
0	0	0	X0
0	0	1	X1
0	1	0	X2
0	1	1	X3
1	0	0	X4
1	0	1	X5
1	1	0	X6
1	1	1	X7

Phương trình logic mô tả hoạt động của mạch

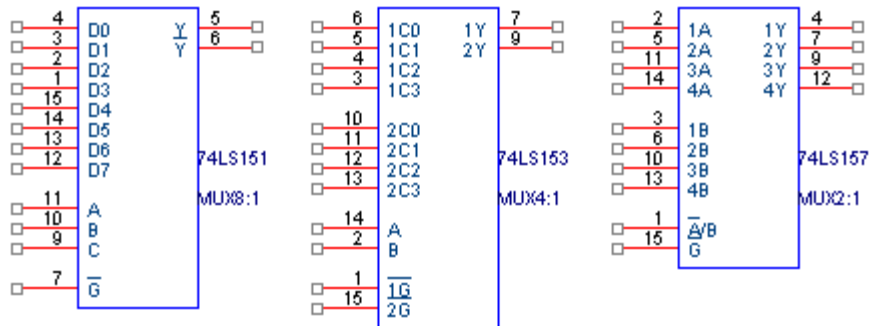
$$Y = X_0 \cdot \bar{C}_1 \cdot \bar{C}_2 \cdot \bar{C}_3 + X_1 \cdot C_1 \cdot \bar{C}_2 \cdot \bar{C}_3 + X_2 \cdot \bar{C}_1 \cdot C_2 \cdot \bar{C}_3 + X_3 \cdot C_1 \cdot C_2 \cdot \bar{C}_3 + X_4 \cdot \bar{C}_1 \cdot \bar{C}_2 \cdot C_3 + X_5 \cdot C_1 \cdot \bar{C}_2 \cdot C_3 + X_6 \cdot \bar{C}_1 \cdot C_2 \cdot C_3 + X_7 \cdot C_1 \cdot C_2 \cdot C_3$$

Sơ đồ mạch logic



Hình 2.23: Sơ đồ mạch chọn kênh 8-1

2.5.4. Một số IC dồn kênh hay dùng



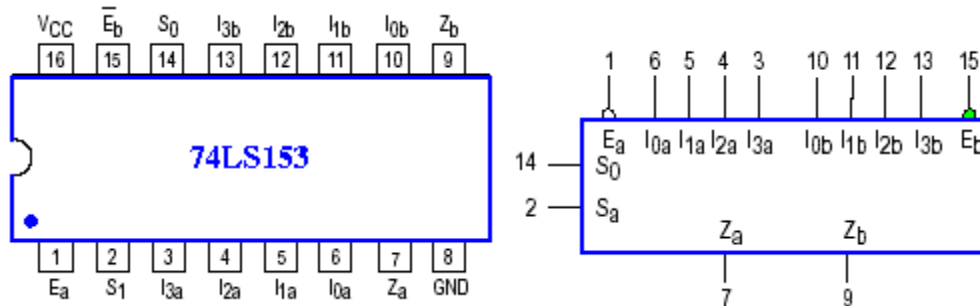
Hình 2.24: Kí hiệu khối của một số IC dồn kênh hay dùng

74LS151 có 8 đường vào dữ liệu, 1 đường vào cho phép G tác động ở mức thấp, 3 đường vào chọn C B A, đường ra Y còn có đường đảo của nó: Khi G ở mức thấp nó cho phép hoạt động ghép kênh mã chọn CBA sẽ quyết định 1 trong 8 đường dữ liệu được đưa ra đường Y. Ngược lại khi G ở mức cao, mạch không được phép nên $Y = 0$ bất chấp các đường chọn và đường vào dữ liệu.

74LS153 gồm 2 bộ ghép kênh 4:1 có 2 đường vào chọn chung BA mỗi bộ có đường cho phép riêng, đường vào và đường ra riêng. Tương tự chỉ khi G ở mức 0 đường Y mới giống 1 trong các đường vào tùy mã chọn.

74LS157 gồm 4 bộ ghép kênh 2:1 có chung đường vào cho phép G tác động ở mức thấp, chung đường chọn A. Đường vào dữ liệu 1I0, 1I1 có đường ra tương ứng là 1Y, đường vào dữ liệu 2I0, 2I1 có đường ra tương ứng là 2Y, ... Khi G ở thấp và A ở thấp sẽ cho dữ liệu vào ở đường nI0 ra ở nY (n = 1,2,3,4) còn khi A ở cao sẽ cho dữ liệu vào ở nI1 ra ở nY. Khi $\bar{G} = 1$ thì Y = 0

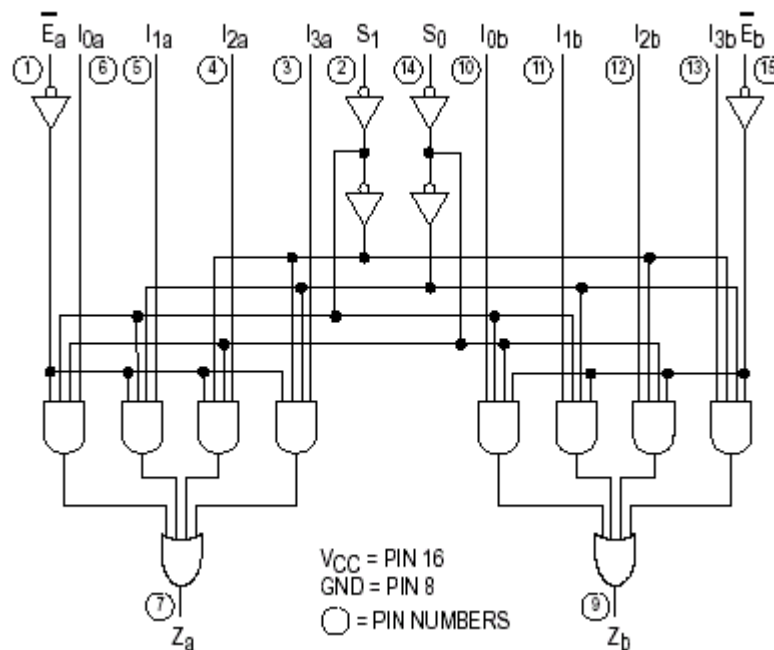
Chẳng hạn với 74LS153, kí hiệu khối, chân ra, bảng trạng thái và cấu tạo logic được minh hoạ ở những hình dưới, với những IC khác cũng tương tự



Hình 2.25: Kí hiệu khối và chân ra của 74LS153

Bảng sự thật của 74LS53

Ngõ chọn		Cho phép	Ngõ vào dẫn kênh				Ngõ ra Z
S1	S0		I0	I1	I2	I3	
X	X	1	X	X	X	X	0
0	0	0	0	X	X	X	0
0	0	0	1	X	X	X	1
0	1	0	X	0	X	X	0
0	1	0	X	1	X	X	1
1	0	0	X	X	0	X	0
1	0	0	X	X	1	X	1
1	1	0	X	X	X	0	0
1	1	0	X	X	X	1	1



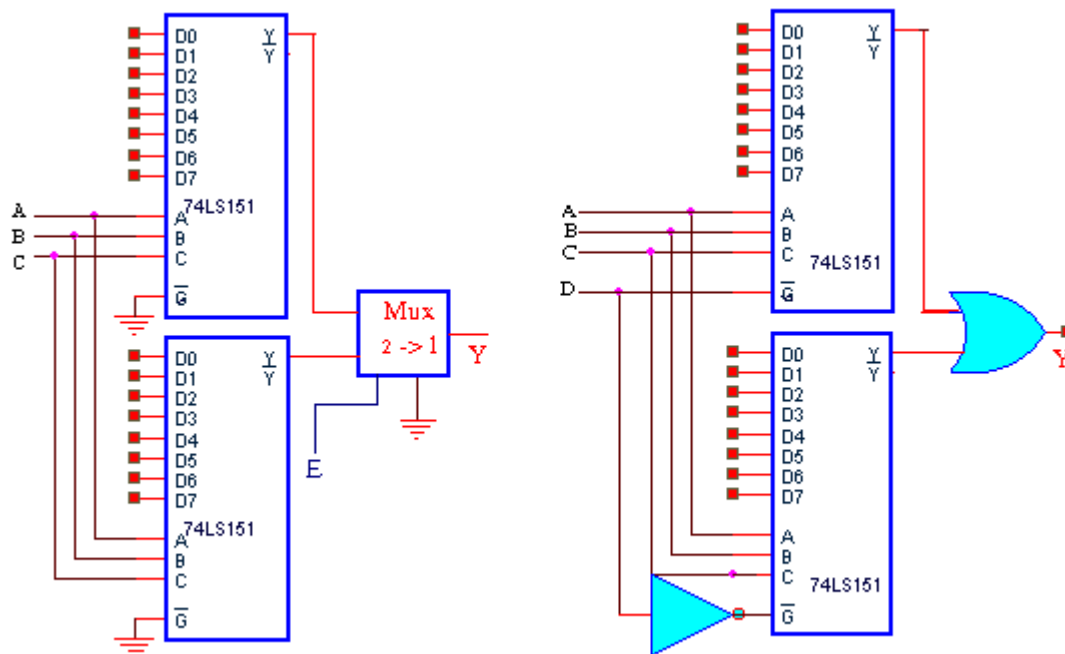
Hình 2.26: Cấu tạo bên trong của 74LS153

2.5.4. Ứng dụng

a) Mở rộng kênh ghép

Các mạch ghép kênh ít đường vào có thể được kết hợp với nhau để tạo mạch ghép kênh nhiều đường vào. Ví dụ để tạo mạch ghép kênh 16:1 ta có thể dùng IC 74LS150 hoặc các IC tương tự, nhưng có 1 cách khác là ghép 2 IC 74LS151

Sơ đồ ghép như sau:



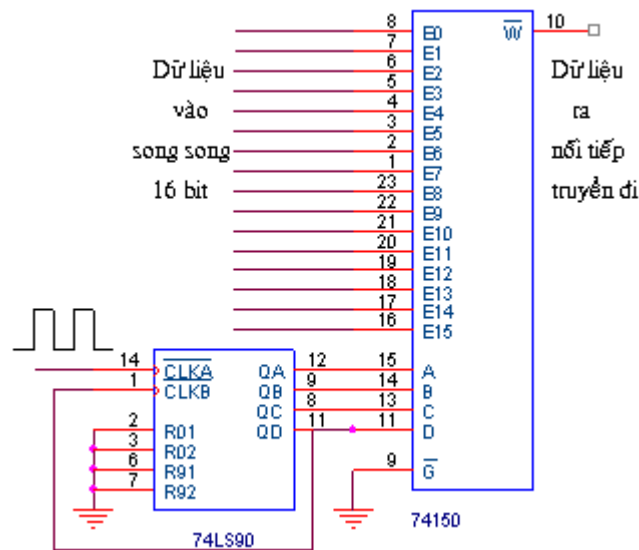
Hình 2.27: Hai cách mở rộng kênh ghép 16 sang 1 từ IC74LS151

(74LS151 là IC dồn kênh 8 sang 1)

b) Chuyển đổi song song sang nối tiếp:

Các dữ liệu nhị phân nhiều bit, chẳng hạn mã ASCII, word,... thường được xử lý song song, tức là tất cả chúng được làm 1 lúc. Trong máy tính, dữ liệu được di chuyển từ nơi này đến nơi khác cùng 1 lúc trên các đường dẫn điện song song gọi là các bus. Khi dữ liệu được truyền đi qua khoảng cách dài chẳng hạn hàng chục mét thì cách truyền song song không còn thích hợp vì tốn nhiều đường dây, nhiễu, Lúc này mạch dồn kênh có thể dùng như mạch chuyển đổi song song sang nối tiếp.

Cách nối



Hình 2.28: Chuyển đổi dữ liệu truyền từ song song sang nối tiếp

Mạch ở hình trên cho phép truyền dữ liệu 16 bit trên đường truyền nối tiếp thông qua IC dồn kênh 74LS150. Tất nhiên cần 1 mạch đếm để tạo mã số nhị phân 4 bit cho 4 đường chọn của mạch dồn kênh (chẳng hạn 74LS93). Mạch đếm hoạt động khiến mã chọn thay đổi từ 0000 rồi 0001, rồi đến 1111 và lại vòng trở lại 0000 đếm lên tiếp khiến dữ liệu vào song song được chuyển đổi liên tiếp sang nối tiếp. Cũng cần phải có một mạch dao động để tạo xung kích cho mạch đếm, nếu tần số dao động tạo xung kích cho mạch đếm rất lớn thì dữ liệu được luân chuyển nhanh, và với tốc độ lớn như vậy với cảm nhận của con người thì dữ liệu dường như được truyền đồng thời. Nguyên lý này được áp dụng cho ghép kênh điện thoại và nhiều ứng dụng khác

c) Dùng dồn kênh để thiết kế tổ hợp:

Các mạch đơn kênh với hoạt động logic như đã xét ở trước ngoài cách dùng để ghép nhiều đường vào còn có thể dùng để thiết kế mạch tổ hợp đôi khi rất dễ dàng vì:

Không cần phải đơn giản biểu thức nhiều

Thường dùng ít IC

Dễ thiết kế

Bài toán thiết kế mạch tổ hợp như bảng dưới đây cho thấy rõ hơn điều này

Ví dụ: Thiết kế mạch tổ hợp thoả bảng sự thật sau

Vào			Ra
C	B	A	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Từ bảng sự thật ta có biểu thức logic là:

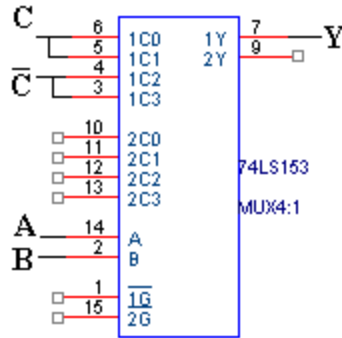
$$Y = ABC + \bar{A}BC + A\bar{B}C + \bar{A}\bar{B}C$$

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

Hay có thể viết

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

Đây là biểu thức thuộc dạng tổng của các tích. Như cách thiết kế ở trước ta sẽ sử dụng các cổng logic gồm 3 cổng NOT, 4 cổng NAND, 1 cổng OR, còn nếu chuyển sang dùng toàn cổng NAND không thì phải cần tới 3 cổng NAND 2 đường vào, 4 cổng NAND 3 đường vào và 1 cổng NAND 4 đường vào chưa kể là phải đơn giản biểu thức nếu có thể trước khi thực hiện.



Hình 2.29: Thiết kế tổ hợp dùng mạch dồn kênh

Tương tự sử dụng MUX 2-1, MUX 4-1 MUX 8-1 thiết kế các hàm sau:

$$Y1(A, B, C, D) = \sum 1,2,3,5,7,9,10,12,14,15$$

$$Y2(A, B, C, D, E) = \sum 1,4,5,7,10,13,14,16,18,19,21,24,27,28,30$$

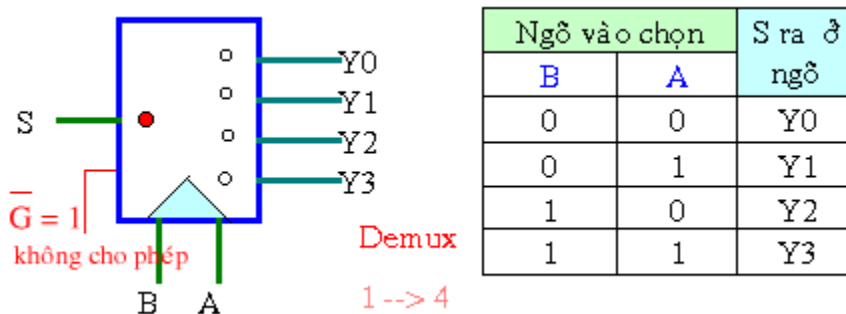
2.6. Mạch tách kênh

2.6.1. Khái niệm

Bộ chuyển mạch phân kênh hay còn gọi là tách kênh, giải đa hợp (demultiplexer) có chức năng ngược lại với mạch dồn kênh tức là: tách kênh truyền thành 1 trong các kênh dữ liệu song song tùy vào mã chọn đường vào. Có thể xem mạch tách kênh giống như 1 công tắc cơ khí được điều khiển chuyển mạch bởi mã số. Tùy theo mã số được áp vào đường chọn mà dữ liệu từ 1 đường sẽ được đưa ra đường nào trong số các đường song song.

Các mạch tách kênh thường gặp là 1 sang 2, 1 sang 4, 1 sang 8, ...Nói chung từ 1 đường có thể đưa ra 2^n đường, và số đường để chọn sẽ phải là n. Mục dưới sẽ nói đến mạch tách kênh 1 sang 4.

2.6.2. Mạch tách kênh 1 sang 4



Hình 2.30: Mạch tách kênh 1 sang 4

Mạch tách kênh từ 1 đường sang 4 đường nên số đường chọn phải là 2

Khi đường cho phép G ở mức 1 thì nó cấm không cho phép dữ liệu vào được truyền ra ở bất kì đường nào nên tất cả các đường ra đều ở mức 0

Như vậy khi G = 0 BA = 00 dữ liệu S được đưa ra đường Y0, nếu S = 0 thì Y0 cũng bằng 0 và nếu S = 1 thì Y0 cũng bằng 1, tức là S được đưa tới Y0; các đường khác không đổi

Tương tự với các tổ hợp BA khác thì lần lượt ra ở S sẽ là Y1, Y2, Y3

Biểu thức logic của các đường ra sẽ là:

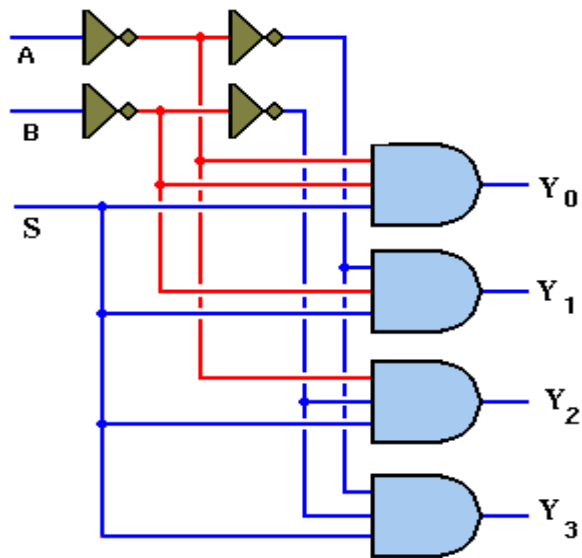
$$Y_0 = G \cdot \bar{B} \cdot \bar{A} \cdot S$$

$$Y_1 = G \cdot \bar{B} \cdot A \cdot S$$

$$Y_2 = G \cdot B \cdot \bar{A} \cdot S$$

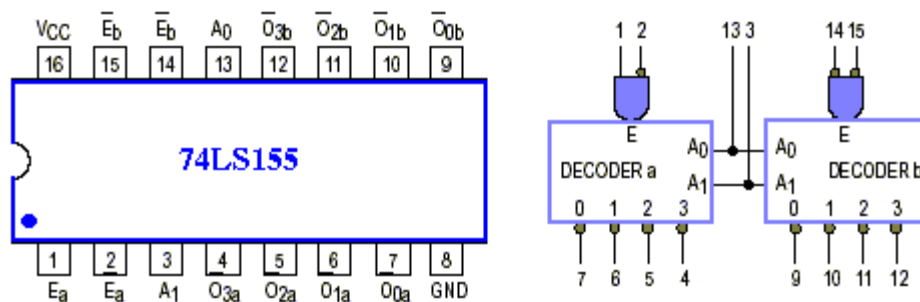
$$Y_3 = G \cdot B \cdot A \cdot S$$

Từ đây có thể dùng cổng logic để thiết kế mạch tách kênh



Hình 2.31: Cấu trúc của mạch tách kênh 1 sang 4

Ví dụ: Khảo sát IC 74LS155



Hình 2.32: Kí hiệu khối và chân ra của 74LS155

Trong cấu trúc của nó gồm 2 bộ tách kênh 1 sang 4, chúng có 2 đường chọn A0A1 chung, đường cho phép cũng có thể chung khi nối chân 2 nối với chân 15). Một lưu ý khác là bộ tách kênh đầu có đường ra đảo so với đường vào (dữ liệu vào chân 1 không đảo) còn bộ tách kênh thứ 2 thì đường vào và đường ra như nhau khi được tác động (dữ liệu vào chân 14 đảo).

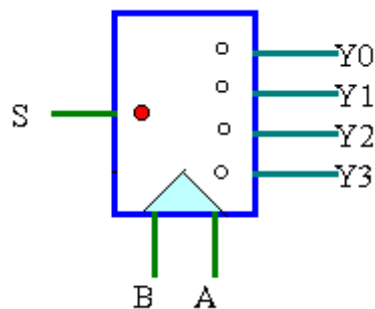
Cấu trúc logic của mạch không khác gì so với mạch đã xét ở trên ngoài trừ mạch có thêm đường cho phép

Bảng sự thật của 74LS155

Ngõ chọn		Cho phép	Kênh vào	Kênh ra "a"				Cho phép	Kênh vào	Kênh ra "b"			
A ₁	A ₀	Ea	Ea	O ₀	O ₁	O ₂	O ₃	Eb	Eb	O ₀	O ₁	O ₂	O ₃
X	X	1	X	1	1	1	1	1	X	1	1	1	1
X	X	X	0	1	1	1	1	X	1	1	1	1	1
0	0	0	1	0	1	1	1	0	0	0	1	1	1
0	1	0	1	1	0	1	1	0	0	1	0	1	1
1	0	0	1	1	1	0	1	0	0	1	1	0	1
1	1	0	1	1	1	1	0	0	0	1	1	1	0

Mạch tách kênh hoạt động như mạch giải mã. Nhiều mạch tách kênh còn có chức năng như 1 mạch giải mã. Thật vậy, vào dữ liệu S không được dùng như 1 đường vào dữ liệu nối tiếp mà lại dùng như đường vào cho phép còn các đường vào chọn CBA khi này lại được dùng như các đường vào dữ liệu và các đường ra vẫn giữ nguyên chức năng thì mạch đa hợp lại hoạt động như 1 mạch giải mã.

Tùy thuộc mã dữ liệu áp vào đường C B A mà một trong các đường ra sẽ lên cao hay xuống thấp tùy cấu trúc mạch. Như vậy mạch tách kênh 1:4 như ở trên đã trở thành mạch giải mã 2 sang 4. Thực tế ngoài đường S khi này trở thành đường cho phép giải mã, mạch trên sẽ phải cần một số đường điều khiển khác để cho phép mạch hoạt động giải mã hay tách kênh; còn cấu tạo logic của chúng hoàn toàn tương thích nhau. Hình sau cho phép dùng mạch tách kênh 1 sang 4 để giải mã 2 sang 4.

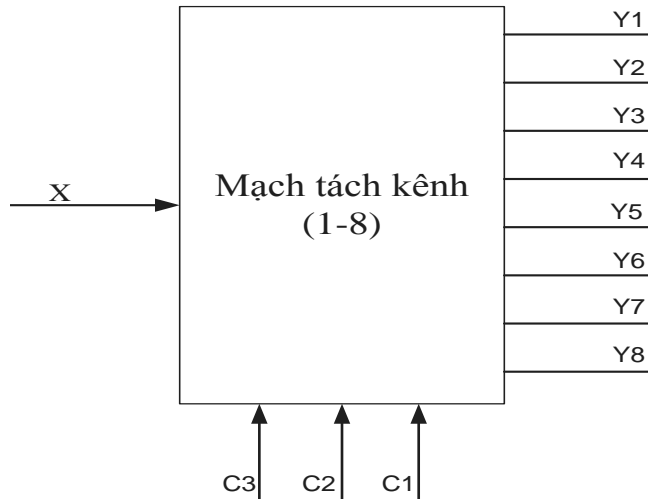


Hình 2.33: Mạch tách kênh hoạt động như mạch giải mã

Tương tự ta cũng có các loại mạch khác như vừa tách kênh 1:8 vừa giải mã 3:8, tách kênh 1:16/giải mã 4:16...

2.6.3. Thiết kế mạch phân kênh 1 ngõ vào 8 ngõ ra

Sơ đồ nguyên lý



Hình 2.34: Sơ đồ nguyên lý mạch phân kênh 1-8

Trong đó:

X: Kênh dữ liệu vào

Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8: là các kênh đầu ra

C1, C2, C3 : là tín hiệu điều khiển

Tại một thời điểm chỉ có một đầu ra được kết nối tới kênh đầu vào, tùy theo giá trị của tổ hợp biến điều khiển để chúng ta xác định được kênh nào được phép kết nối với đầu vào.

Bản trạng thái mô tả quá trình hoạt động mạch phân kênh

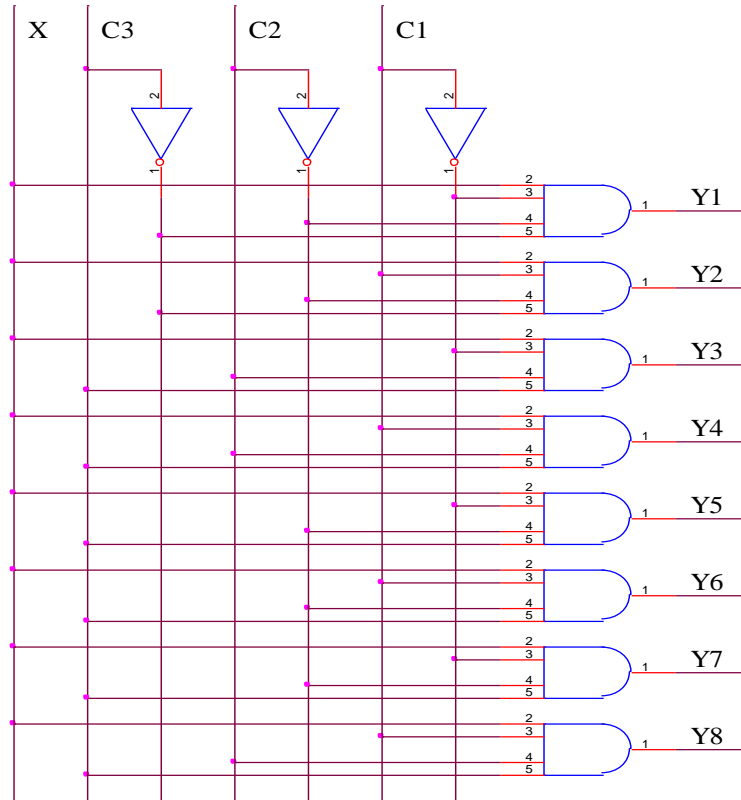
C3	C2	C1	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8
0	0	0	x	0	0	0	0	0	0	0
0	0	1	0	x	0	0	0	0	0	0
0	1	0	0	0	x	0	0	0	0	0
0	1	1	0	0	0	x	0	0	0	0
1	0	0	0	0	0	0	x	0	0	0
1	0	1	0	0	0	0	0	x	0	0
1	1	0	0	0	0	0	0	0	x	0
1	1	1	0	0	0	0	0	0	0	x

Phương trình logic các ngõ ra

$$Y_1 = X \cdot \overline{C_1} \cdot \overline{C_2} \cdot \overline{C_3} \quad Y_2 = X \cdot C_1 \cdot \overline{C_2} \cdot \overline{C_3} \quad Y_3 = X \cdot \overline{C_1} \cdot C_2 \cdot \overline{C_3} \quad Y_4 = X \cdot C_1 \cdot C_2 \cdot \overline{C_3}$$

$$Y_5 = X \cdot \overline{C_1} \cdot \overline{C_2} \cdot C_3 \quad Y_6 = X \cdot C_1 \cdot \overline{C_2} \cdot C_3 \quad Y_7 = X \cdot \overline{C_1} \cdot C_2 \cdot C_3 \quad Y_8 = X \cdot C_1 \cdot C_2 \cdot C_3$$

Sơ đồ mạch logic



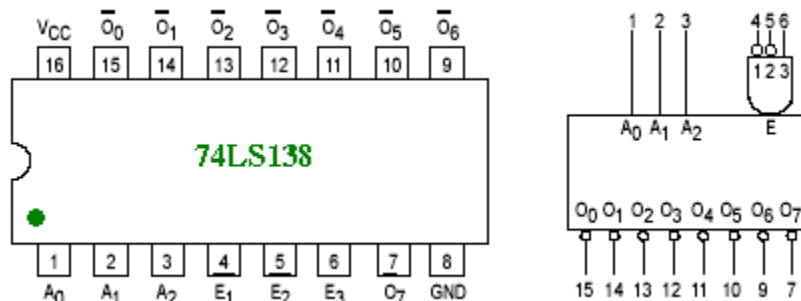
Hình 2.35: Sơ đồ mạch phân kênh 1 đầu vào 8 đầu ra

2.6.4. Một số IC giải mã tách kênh hay dùng

Khảo sát IC tách kênh/giải mã tiêu biểu 74LS138

74LS138 là IC MSI giải mã 3 đường sang 8 đường hay tách kênh 1 đường sang 8 đường thường dùng và có hoạt động logic tiêu biểu, nó còn thường được dùng như mạch giải mã địa chỉ trong các mạch điều khiển và trong máy tính.

Sơ đồ chân và kí hiệu logic như hình dưới đây:



Hình 2.36: Kí hiệu khối và chân ra của 74LS138

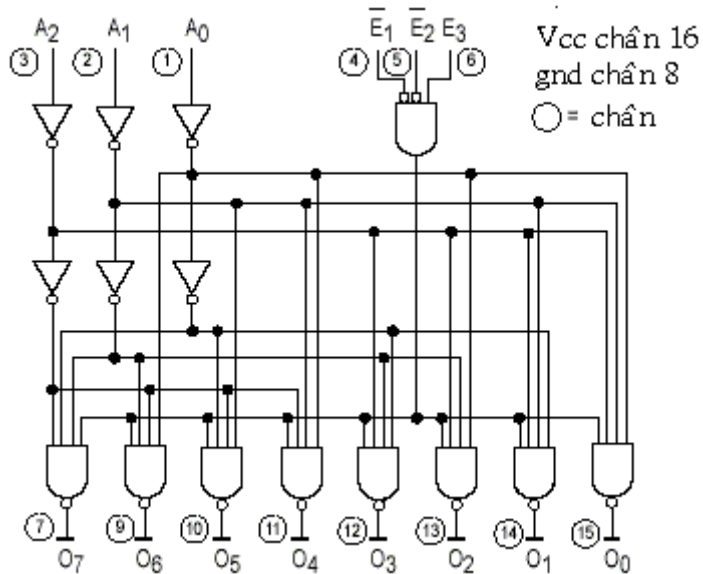
Trong đó

A0, A1, A2 là 3 đường địa chỉ đường vào

E1, E2 là các đường vào cho phép (tác động mức thấp)

E3 là đường vào cho phép tác động mức cao

O0 đến O7 là 8 đường ra (tác động ở mức thấp)



Hình 2. 37: Cấu trúc bên trong 74LS138

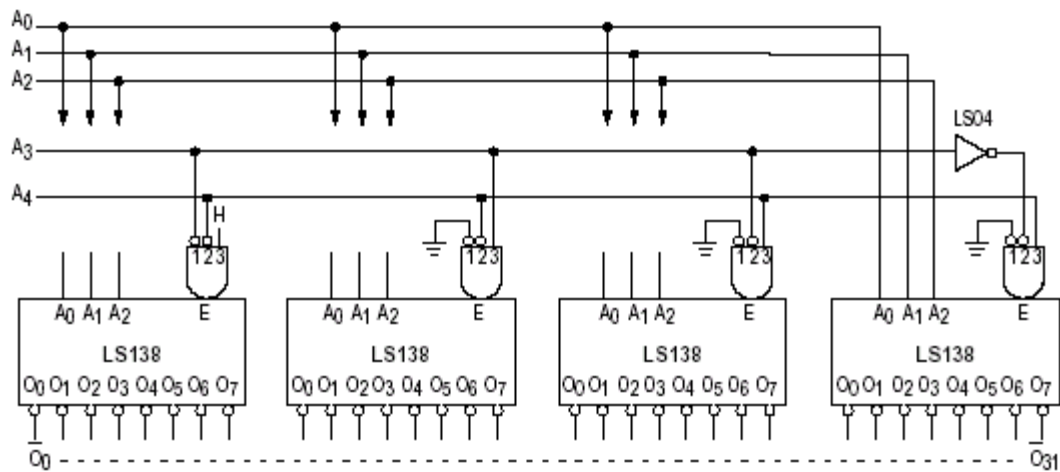
Hoạt động giải mã như sau:

Đưa dữ liệu nhị phân 3bit vào ở C, B, A(LSB), lấy dữ liệu ra ở các đường O0 đến O7; đường cho phép E2 và E3 đặt mức thấp, đường cho phép E1 đặt ở mức cao. Chẳng hạn khi CBA là 001 thì đường O1 xuống thấp còn các đường ra khác đều ở cao.

Hoạt động tách kênh:

Dữ liệu vào nối tiếp vào đường E2, hay E3 (với đường còn lại đặt ở thấp). Đặt G = 1 để cho phép tách kênh. Như vậy dữ liệu ra song song vẫn lấy ra ở các đường O0 đến O7. Chẳng hạn nếu mã chọn là 001 thì dữ liệu nối tiếp S sẽ ra ở đường O1 và không bị đảo.

Mở rộng đường giải mã: 74LS138 dùng thêm 1 cổng đảo còn cho phép giải mã địa chỉ từ 5 sang 32 đường (đủ dùng trong giải mã địa chỉ của máy vi tính). Hình ghép nối như sau:



Hình 2.38: Ghép 4 IC 74LS138 để có mạch giải mã 5 đường sang 32 đường

2.6.5: Ứng dụng

a) Dùng mạch tách kênh thiết kế mạch logic

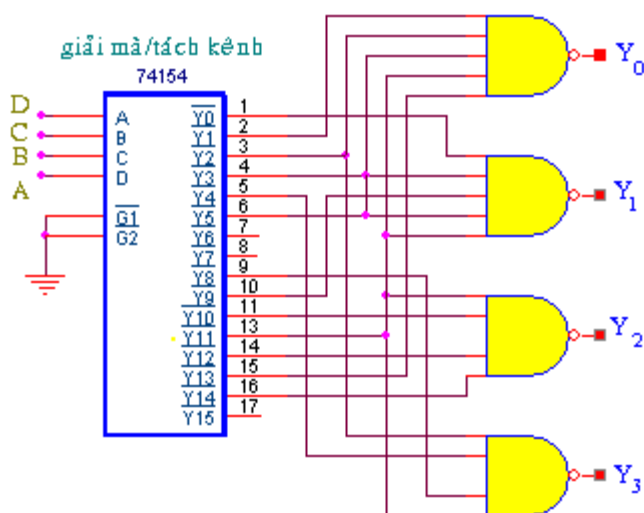
Cũng giống như mạch dồn kênh, mạch tách kênh hay giải mã còn có thể dùng để thiết kế mạch logic tổ hợp. Nếu như việc thiết kế tổ hợp dùng mạch dồn kênh không dùng thêm các cổng logic thì với mạch tách kênh hay giải mã tổ hợp lại phải thêm vào một số cổng logic mới đạt được logic tổ hợp mong muốn. Nhưng bù lại, mạch tách kênh/giải mã cho phép thiết kế tổ hợp nhiều đường ra một cách dễ dàng. Ví dụ sau sẽ minh họa rõ hơn vấn đề này:

Cho bảng sự thật như hình sau:

Thứ tự	Dữ liệu vào				Ra			
	D	C	B	A	Y ₀	Y ₁	Y ₂	Y ₃
0	0	0	0	0	0	1	0	0
1	0	0	0	1	1	0	0	0
2	0	0	1	0	1	0	0	1
3	0	0	1	1	0	1	0	0
4	0	1	0	0	0	0	0	1
5	0	1	0	1	0	1	0	0
6	0	1	1	0	0	0	0	0
7	0	1	1	1	1	0	0	0
8	1	0	0	0	0	0	0	1
9	1	0	0	1	0	1	0	0
10	1	0	1	0	0	0	1	0
11	1	0	1	1	1	1	0	1
12	1	1	0	0	0	0	1	0
13	1	1	0	1	1	0	1	0
14	1	1	1	0	0	0	1	0
15	1	1	1	1	0	0	0	0

Nếu sử dụng cách cũ, ta sẽ xây dựng bìa K, rồi rút gọn, với 4 đường vào, 4 đường ra, xem ra việc rút gọn khá dài và phức tạp. Dùng IC 74154 (giải mã 4 sang 16, tách kênh 1 sang 16), thì bài toán sẽ đơn giản hơn.

Thật vậy, trước hết cần nối 4 đường vào A, B, C, D tới 4 đường chọn của IC tách kênh, rồi dựa vào bảng sự thật ở trên, ta xác định các vị trí tổ hợp làm Y0 lên 1. Bên mạch giải mã/tách kênh ta sẽ nối các đường ra tương ứng với vị trí tổ hợp tới Y0. Vì có tất cả 5 đường ra lên 1 nên cuối cùng Y0 sẽ là NAND của 5 đường ra ấy. Tương tự với các đường ra Y1, Y2, Y3. Cách nối mạch như hình dưới đây

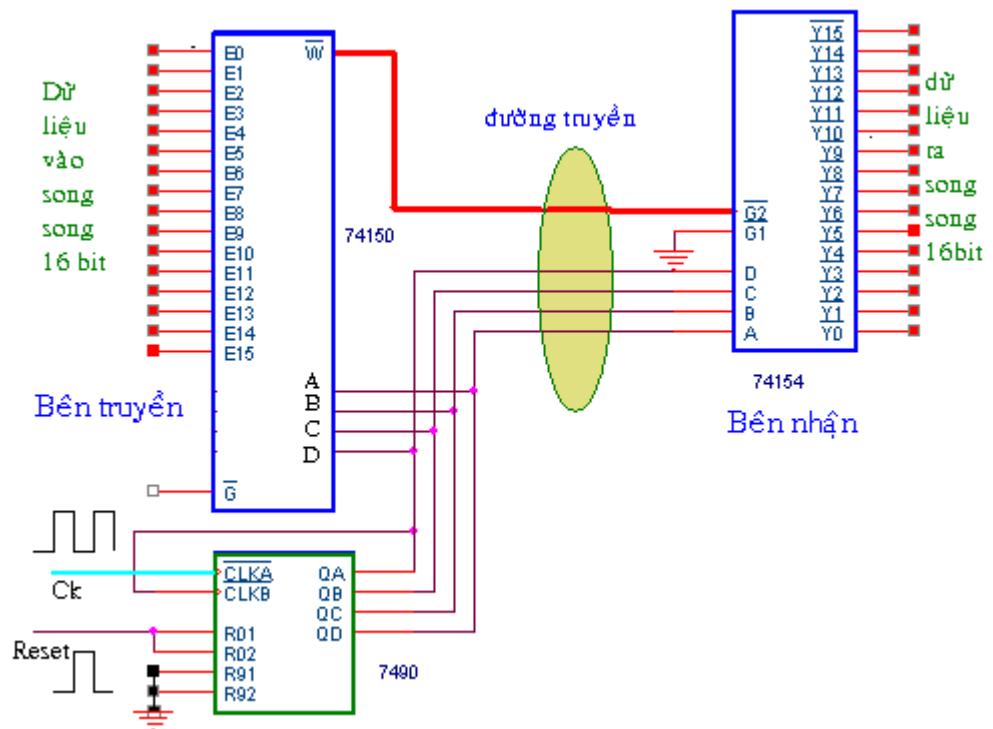


Hình 2.40: Ứng dụng mạch tách kênh thiết kế tổ hợp

Nếu trong 1 cột đường ra mà số bit 0 nhiều hơn số bit 1 thì ta sẽ dùng cổng NOR gom tất cả các đường bit 0 ấy.

b) Chia sẻ đường truyền

Ở phần trước ta đã nói đến ứng dụng của mạch dồn kênh cho phép chuyển đổi dữ liệu từ song song sang nối tiếp và truyền đi. Khi dữ liệu đến nơi cần nhận, chẳng hạn máy tính khác thì nó cũng xử lý dữ liệu ở dạng song song. Do đó lại phải cần 1 mạch chuyển đổi từ dữ liệu nối tiếp đến thành dữ liệu song song và ở đây mạch giải mã/tách kênh được dùng



Hình 2.41: Truyền dữ liệu nối tiếp

Đề ý là mạch giải mã/tách kênh ở bên nhận cũng phải cần mã chọn áp vào các đường DCBA, mã này được lấy từ mạch đếm bên truyền, do đó dữ liệu bên truyền đi và bên nhận lại mới đồng bộ nhau. Như vậy ta sẽ cần 5 đường dây gồm 1 đường truyền dữ liệu nối tiếp, 1 đường mass chung và 4 đường mã số chọn. Ngoài ra do mạch đếm tự động reset khi đếm hết mã (lên 1111) làm dữ liệu được truyền liên tục nên ta cần phải có 1 mạch chốt ở đường ra song song để chặn dữ liệu lại khi đủ 16 bit truyền mới cho ra một lượt

Thực ra thì cách truyền này vẫn chưa hiệu quả lắm, chỉ dùng ở khoảng cách gần, ta vẫn có thể giảm bớt số dây chuyền đi nữa (thay vì 6 đường dây như ở trên). Thật vậy, thay vì truyền đi tới 4 đường cho mã số chọn từ mạch đếm ta sẽ chỉ truyền đi 1 đường xung đồng hồ chung tức là bên nhận sẽ đặt thêm 1 mạch đếm nữa để tạo mã số chọn cho bộ giải mã/tách kênh và mạch đếm này được cấp xung ck giống như mạch đếm của bên truyền. Cách hay hơn nữa là truyền xung ck ngay trên đường truyền nối tiếp, tất nhiên ta phải mã hoá xung ck lại để nó không lẫn lộn với dữ liệu truyền và bên nhận cũng phải có 1 mạch phát hiện và tách xung ck ra khỏi dữ liệu nhận. Đây được gọi là cách truyền tin (dữ liệu) nối tiếp đồng bộ (synchronous data transmission). Ngoài ra còn có cách truyền tin nối tiếp không đồng bộ tức là bên truyền và bên nhận không dùng xung đồng hồ như nhau, hay nói

cách khác dữ liệu truyền và nhận không đồng bộ nhau. 2 cách truyền này ta sẽ được tìm hiểu rõ hơn nhiều ở môn học “truyền số liệu”, “giao tiếp máy tính”

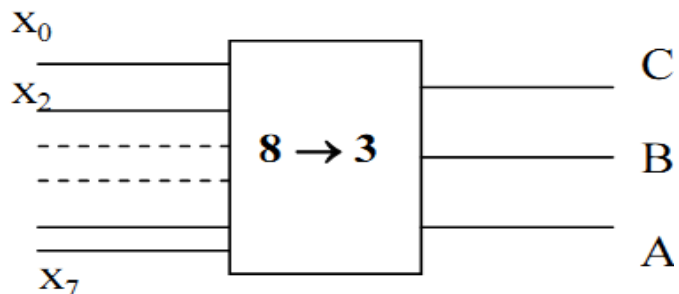
Cũng cần nói thêm rằng các đường vào của mạch dồn kênh không chỉ là 1 byte, 1 word dữ liệu song song cần truyền mà có thể là các đường tín hiệu riêng lẻ, chẳng hạn một số đường lấy từ cảm biến nhiệt độ của lò nhiệt, của các gian phòng chống cháy nổ; một số khác là từ cảm biến dò mực chất lỏng, một số khác lại từ các công tắc tiếp điểm khi bị tác động sẽ tạo mức tín hiệu logic phản hồi, Tất cả đều được thu thập chuyển đổi và dồn lại để truyền về nơi cần thiết chẳng hạn phòng giám sát điều khiển. Tại đây các thông tin được tách trở lại và xử lí, hiển thị về tình trạng của nơi đang giám sát thu thập chẳng hạn có kẻ đột nhập cửa, có khói có thể xảy ra cháy, mực nước, nhiệt độ vượt quá mức cho phép hay tất cả vẫn bình thường. Như vậy đây có thể được sử dụng cho hệ thống theo dõi an ninh từ xa.

2.7. Thiết kế mạch mã hóa

Mạch mã hóa (Encoder) là mạch có nhiệm vụ biến đổi những ký hiệu quen thuộc với con người, sang những ký hiệu không quen thuộc với con người.

2.7.1. Thiết kế mạch mã hóa nhị phân từ 8 sang 3

Sơ đồ khối mạch như sau



Hình 2.42: Sơ đồ khối mạch mã hóa nhị phân từ 8 sang 3

Trong đó

- x_0, x_1, \dots, x_7 là các ngõ vào tín hiệu
- A, B, C là các ngõ ra

Mạch mã hóa nhị phân thực hiện biến đổi tín hiệu ngõ vào thành một từ mã nhị phân tương ứng ở ngõ ra cụ thể như sau:

$0 \rightarrow 000$ $1 \rightarrow 001$ $2 \rightarrow 010$ $3 \rightarrow 011$
 $4 \rightarrow 100$ $5 \rightarrow 101$ $6 \rightarrow 110$ $7 \rightarrow 111$

Chọn mức tác động tích cực ở ngõ vào là mức logic 1, ta có bản trạng thái mô tả hoạt động của mạch như sau:

x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	C	B	A
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

Khi một ngõ vào ở trạng thái tích cực (mức logic 1) và các ngõ vào không được tích cực nhận mức logic 0. Thì ngõ ra xuất hiện từ mã tương ứng . Cụ thể khi $x_0=1$, các đầu ra còn lại $x_1= x_2= x_3 = x_4 = x_5 = x_6 = x_7 = 0$, thì từ mã ngõ ra là 000. khi $x_1=1$ các đầu ra còn lại $x_0= x_2= x_3 = x_4 = x_5 = x_6 = x_7 = 0$, thì từ mã ngõ ra nhận giá trị 001,..vv...

Từ bảng trạng thái ta có phương trình trạng thái ngõ ra như sau:

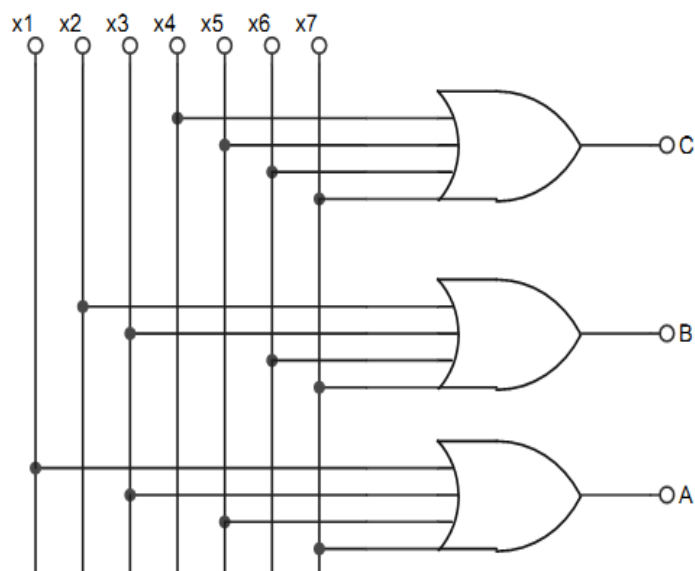
$$A = x_1 + x_3 + x_5 + x_7$$

$$B = x_2 + x_3 + x_6 + x_7$$

$$C = x_4 + x_5 + x_6 + x_7$$

Từ phương trình trạng thái ngõ ra ta có sơ đồ mạch logic thực hiện quá trình mã hóa như sa

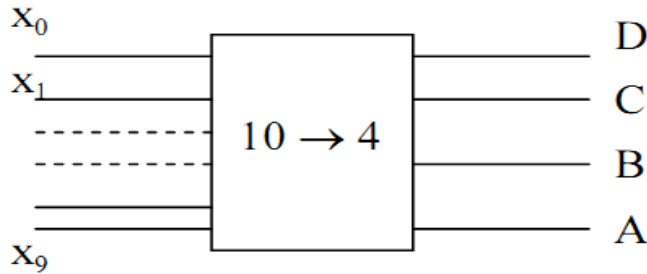
Mạch logic dùng phần tử OR



Hình 2.43: Mạch mã hóa nhị phân 8 bit sang 3

2.7.2. Thiết kế mạch mã hóa thập phân 10-4

Sơ đồ khối



Hình 2.44: Sơ đồ khối mạch mã hóa thập phân

Trong đó

- x_0, x_1, \dots, x_9 là các ngõ vào tín hiệu
- A, B, C, D là các ngõ ra

Mạch mã hóa nhị phân thực hiện biến đổi tín hiệu ngõ vào thành một từ mã nhị phân tương ứng ở ngõ ra cụ thể như sau:

$0 \rightarrow 0000$ $1 \rightarrow 0001$ $2 \rightarrow 0010$ $3 \rightarrow 0011$ $4 \rightarrow 0100$
 $5 \rightarrow 0101$ $6 \rightarrow 0110$ $7 \rightarrow 0111$ $8 \rightarrow 1000$ $9 \rightarrow 1001$

Chọn mức tác động tích cực ở ngõ vào là mức logic 1, ta có bản trạng thái mô tả hoạt động của mạch như sau:

x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	D	C	B	A
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	1	1	1
0	0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	1	0	0	1

Từ bảng trạng thái ta có phương trình trạng thái ngõ ra như sau:

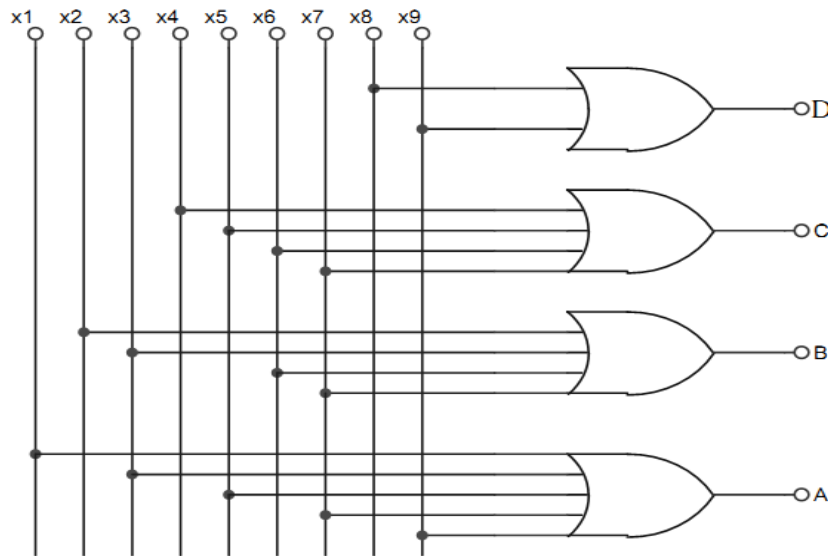
$$A = x_1 + x_3 + x_5 + x_7 + x_9$$

$$B = x_2 + x_3 + x_6 + x_7$$

$$C = x_4 + x_5 + x_6 + x_7$$

$$D = x_8 + x_9$$

Từ phương trình trạng thái tối giản ta có sơ đồ mạch logic dùng phần tử OR như sau:



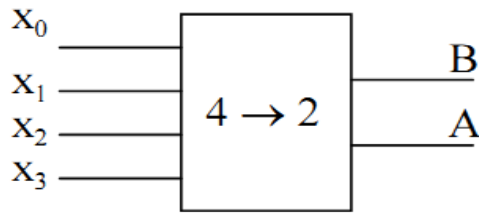
Hình 2.45: Sơ đồ mạch mã hóa thập phân dùng OR

2.8. Thiết kế mạch mã hóa ưu tiên

Với mạch mã hoá được cấu tạo bởi các cổng logic như ở hình trên ta có nhận xét rằng trong trường hợp nhiều phím được nhấn cùng 1 lúc thì sẽ không thể biết được mã số sẽ ra là bao nhiêu. Do đó để đảm bảo rằng khi 2 hay nhiều phím hơn được nhấn, mã số ra chỉ tương ứng với đường vào có số cao nhất được nhấn, người ta đã sử dụng mạch mã hoá ưu tiên.

Vấn đề ưu tiên: Khi có nhiều tín hiệu đồng thời tác động, tín hiệu nào có mức ưu tiên cao hơn ở thời điểm đang xét sẽ tác động, tức là nếu ngõ vào có độ ưu tiên cao hơn bằng 1 trong khi nếu các ngõ vào có độ ưu tiên thấp hơn nếu bằng 1 thì mạch sẽ tạo ra từ mã nhị phân ứng với ngõ vào có mức độ ưu tiên cao nhất.

2.8.1. Thiết kế mạch mã hóa ưu tiên 4-2



Hình 2.46: Sơ đồ mô phỏng

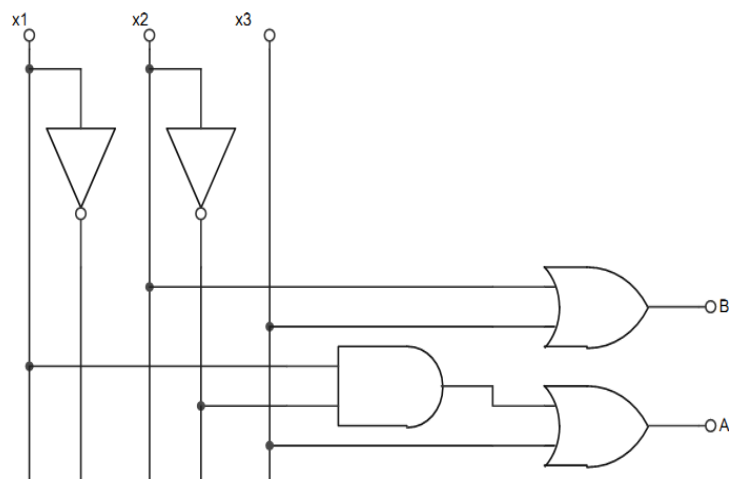
x_0	x_1	x_2	x_3	B	A
1	0	0	0	0	0
x	1	0	0	0	1
x	x	1	0	1	0
x	x	x	1	1	1

Bảng chân lý

Phương trình tối giản:

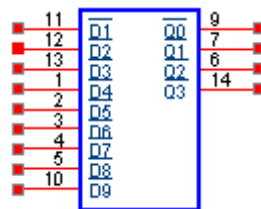
$$A = x_1 \cdot \overline{x_2} \cdot \overline{x_3} + x_3 = x_1 \cdot \overline{x_2} + x_3$$

$$B = x_2 \cdot \overline{x_3} + x_3 = x_2 + x_3$$



Hình 2.47: Sơ đồ mạch mã hóa ưu tiên 4-2

IC 74LS147 là mạch mã hoá ưu tiên 10 đường sang 4 đường, nó đã được tích hợp sẵn tất cả các cổng logic trong nó. Kí hiệu khối của 74LS147 như hình 2.1.5 ở bên dưới:



Hình 2.48: IC74LS147

Bảng sự thật của 74LS147

Các ngõ vào thập phân tác động ở mức thấp									Các ngõ ra BCD tác động ở thấp			
1	2	3	4	5	6	7	8	9	Q ₃	Q ₂	Q ₁	Q ₀
1	1	1	1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	0
X	0	1	1	1	1	1	1	1	1	1	0	1
X	X	0	1	1	1	1	1	1	1	1	0	0
X	X	X	0	1	1	1	1	1	1	1	0	1
X	X	X	X	0	1	1	1	1	1	1	0	0
X	X	X	X	X	0	1	1	1	1	1	0	1
X	X	X	X	X	X	0	1	1	1	1	0	0
X	X	X	X	X	X	X	0	1	0	1	1	1
X	X	X	X	X	X	X	X	0	0	1	1	0

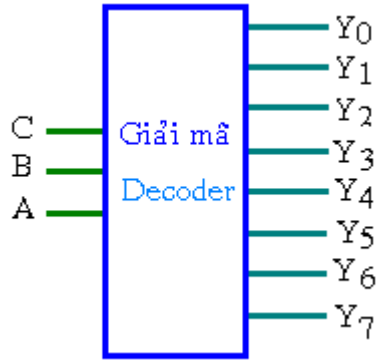
Nhìn vào bảng sự thật ta thấy thứ tự ưu tiên giảm từ đường vào 9 xuống đường vào 0. Chẳng hạn khi đường vào 9 đang là 0 thì bất chấp các đường khác (X) số BCD ra vẫn là 1001 (qua công đảo nữa). Chỉ khi đường vào 9 ở mức 1 (mức không tích cực) thì các đường vào khác mới có thể được chấp nhận, cụ thể là đường vào 8 sẽ ưu tiên trước nếu nó ở mức thấp. Với mạch mã hoá ưu tiên 8 đường sang 3 đường, cũng có IC tương ứng là 74LS148.

2.9. Thiết kế mạch giải mã

Mạch giải mã là mạch có chức năng ngược lại với mạch mã hoá tức là nếu có 1 mã số áp vào đường vào thì tương ứng sẽ có 1 đường ra được tác động, mã đường vào thường ít hơn mã đường ra. Tất nhiên đường vào cho phép phải được bật lên cho chức năng giải mã. Mạch giải mã được ứng dụng chính trong ghép kênh dữ liệu, hiển thị led 7 đoạn, giải mã địa chỉ bộ nhớ. Hình dưới là sơ đồ khối của mạch giải mã

2.9.1. Giải mã 3 sang 8

Mạch giải mã 3 đường sang 8 đường bao gồm 3 đường vào tạo nên 8 tổ hợp trạng thái, ứng với mỗi tổ hợp trạng thái được áp vào sẽ có 1 đường ra được tác động.

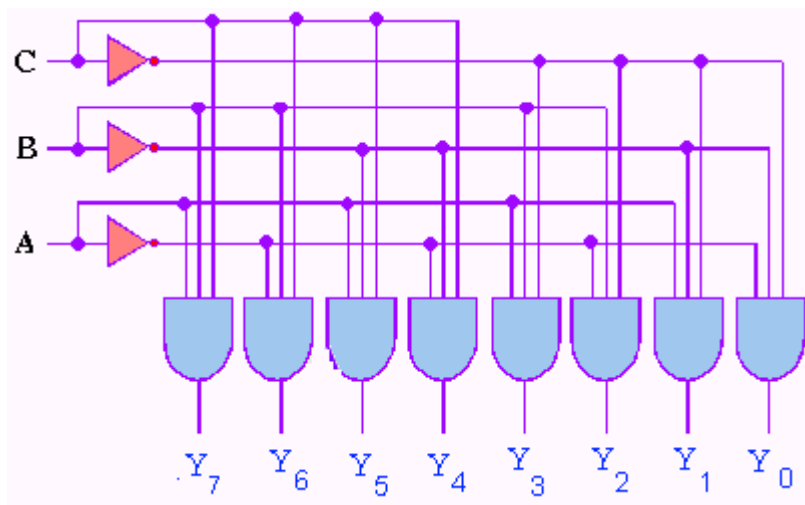


Hình 2.49: Khối giải mã 3 sang 8

Bảng sự thật mạch giải mã 3 sang 8

C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Từ bảng sự thật ta có thể vẽ được sơ đồ mạch logic của mạch giải mã trên



Hình 50: Cấu trúc mạch giải mã 3 sang 8

• **Ứng dụng**

a). rút gọn hàm logic sử dụng mạch giải mã

Nhiều hàm logic có đường ra là tổ hợp của nhiều đường vào có thể được xây dựng từ mạch giải mã kết hợp với một số cổng logic ở đường ra (mạch giải mã chính là 1 mạch tổ hợp nhiều cổng logic cỡ MSI). Mạch giải mã đặc biệt hiệu quả hơn so với việc sử dụng các cổng logic rời trong trường hợp có nhiều tổ hợp đường ra.

Ví dụ: Sau thực hiện mạch cộng 3 số X, Y, Z cho tổng là S và số nhớ là C thực hiện bằng mạch giải mã:

Giả sử mạch cộng thực hiện chức năng

logic như bảng sau:

X	Y	Z	S	C
0	0	0	0	0
0	0	1	1	0

Từ bảng cho phép ta xác định được các tổ hợp logic đường vào để S rồi C ở mức cao

$$S(x, y, z) = \sum 1,2,4,7$$

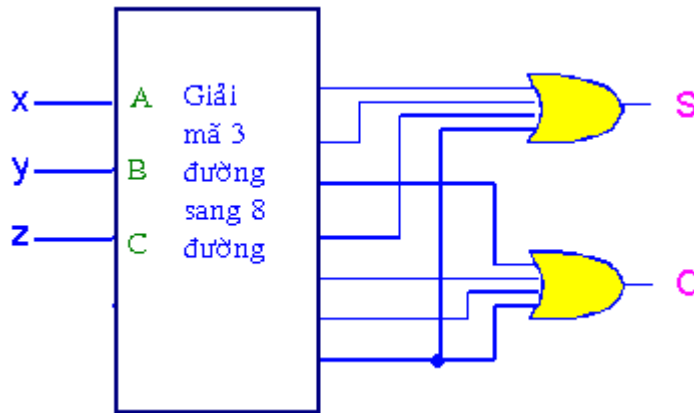
$$C(x, y, z) = \sum 3,5,6,7$$

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

Như vậy sẽ cần 1 cổng OR để nối chung các tổ hợp logic thứ 1, 2, 4, 7 để đưa ra đường S

Tương tự đường ra C cũng cần 1 cổng OR với đường vào là tổ hợp logic thứ 2, 5, 6, 7

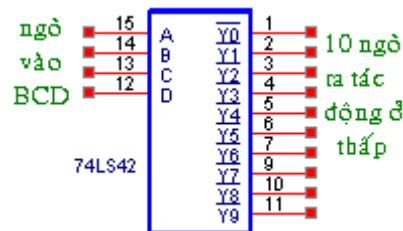
Vậy mạch giải mã thực hiện bảng logic trên sẽ được mắc như sau:



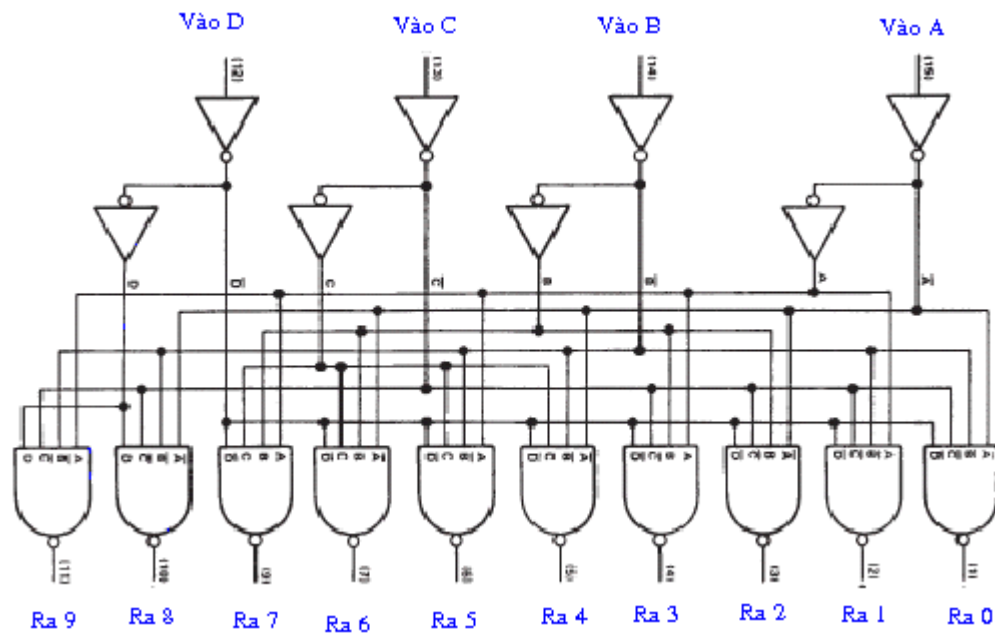
Hình 2.51: Ứng dụng mạch giải mã làm mạch cộng

2.9.2. Mạch giải mã BCD sang thập phân

74LS42 là IC làm nhiệm vụ giải mã 4 đường sang 10 đường. Cấu tạo logic và bảng hoạt động của nó sẽ minh họa rõ hơn cho mạch giải mã này:



Hình 2.52: Sơ đồ chân IC giải mã BCD sang thập phân



Hình 2.53: Cấu trúc mạch của 74LS42, giải mã 4 sang 10

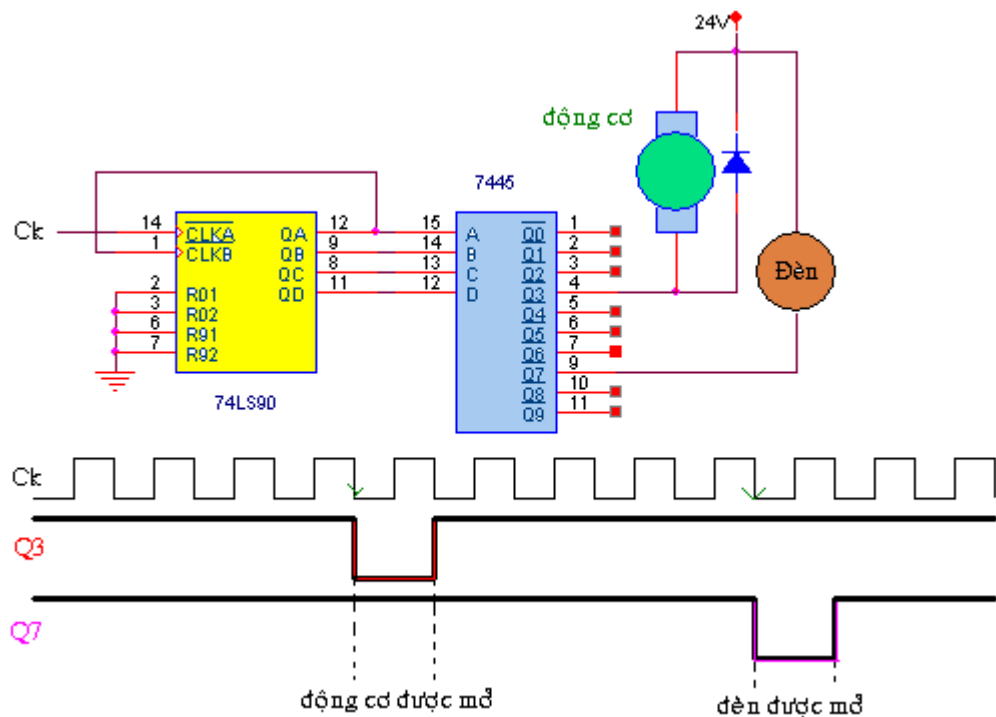
Bảng sự thật của 74LS42

Vào BCD				Ra tác động ở thấp
D	C	B	A	
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

Đề ý là vì có 4 đường vào nên sẽ có 16 trạng thái logic đường ra. Ở đây chỉ sử dụng 10 trạng thái logic đầu, 6 trạng thái sau không dùng. Với mạch giải mã 4 sang 16 thì sẽ tận dụng hết số trạng thái ra. Một điểm nữa là các đường ra của 7442 tác động ở mức thấp

Về nguyên tắc ta có thể mã hoá từ n đường sang m đường và ngược lại giải mã từ m đường sang n đường, chức năng giữa mã hoá và giải mã không rõ rệt lắm, chúng đều làm nhiệm vụ chuyển đổi từ

mã này sang mã khác (những mạch ở trên đều nói đến mã hệ 2, thực ra còn nhiều loại mã khác). Cũng chỉ có một số chúng được tích hợp sẵn trong IC như 7441, 7442 là giải mã BCD sang thập phân, 7443 là giải mã thừa 3 sang thập phân, ... Mạch sau minh hoạ cách kết hợp mạch đếm sẽ học ở chương sau với mạch giải mã để cung cấp các hoạt động định thời và định thứ tự, IC giải mã 7445 được dùng vì tải là động cơ có áp lớn dòng lớn ngoài sức cung cấp của các IC giải mã thường



Hình 2.54: Ứng dụng 74LS45

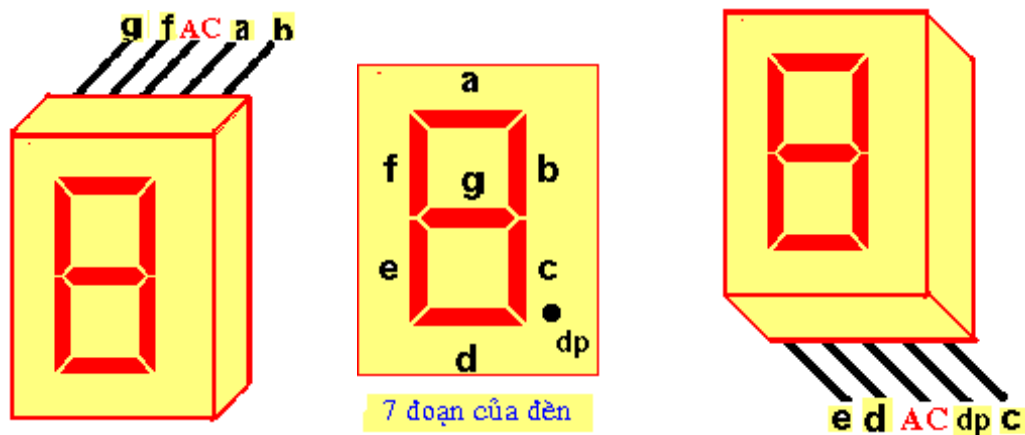
Hình trên cho thấy, mạch đếm tạo ra 16 tổ hợp trạng thái cho mạch mã hoá. Phải 4 chu kì xung ck thì Q3 mới xuống thấp, cho phép động cơ được cấp nguồn; còn đèn được mở chỉ sau 8 chu kì xung ck. Thời gian mở của tải là 1 chu kì xung ck. Ta có thể điều chỉnh thời gian này từ mạch dao động tạo xung ck. Về nguyên tắc hoạt động của mạch đếm 74LS90 ta sẽ tìm hiểu ở chương sau.

2.9.3. Thiết kế mạch giải mã BCD sang led 7 đoạn

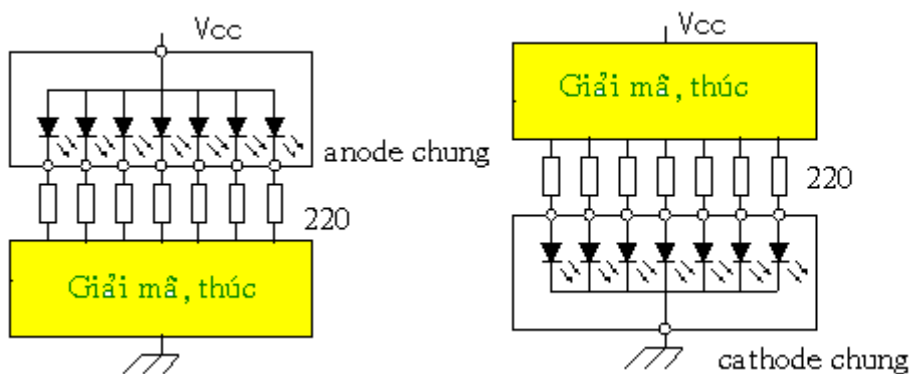
Một dạng mạch giải mã khác rất hay sử dụng trong hiển thị led 7 đoạn đó là mạch giải mã BCD sang led 7 đoạn. Mạch này phức tạp hơn nhiều so với mạch giải mã BCD sang thập phân đã nói ở phần trước bởi vì mạch khi này phải cho ra tổ hợp có nhiều đường ra lên cao xuống thấp hơn (tùy loại đèn led anode chung hay cathode chung) để làm các đoạn led cần thiết sáng tạo nên các số hay kí tự.

Led 7 đoạn

Trước hết hãy xem qua cấu trúc và loại đèn led 7 đoạn của một số đèn được cấu tạo bởi 7 đoạn led có chung anode (AC) hay cathode (KC); được sắp xếp hình số 8 vuông (như hình trên) ngoài ra còn có 1 led con được đặt làm dấu phẩy thập phân cho số hiển thị; nó được điều khiển riêng biệt không qua mạch giải mã. Các chân ra của led được sắp xếp thành 2 hàng chân ở giữa mỗi hàng chân là A chung hay K chung. Thứ tự sắp xếp cho 2 loại như trình bày ở dưới đây.



Hình 2.55: Cấu trúc và chân ra của 1 dạng led 7 đoạn

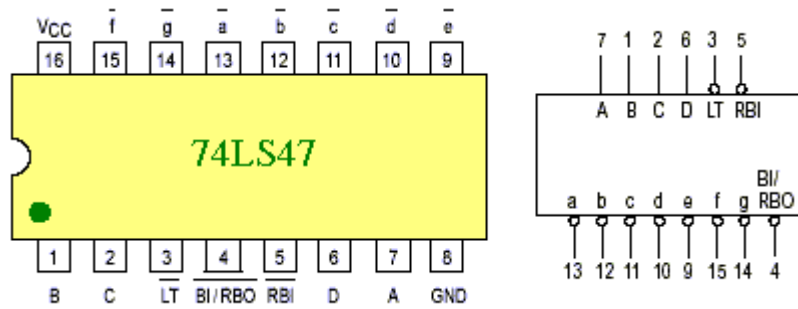


Hình 2.56: Led 7 đoạn loại anode chung và cathod chung cùng với mạch thúc giải mã

Để đèn led hiển thị 1 số nào thì các thanh led tương ứng phải sáng lên, do đó, các thanh led đều phải được phân cực bởi các điện trở khoảng 180 đến 390 ohm với nguồn cấp chuẩn thường là 5V. IC giải mã sẽ có nhiệm vụ nối các chân a, b,.. g của led xuống mass hay lên nguồn (tùy A chung hay K chung)

Khảo sát 74LS47

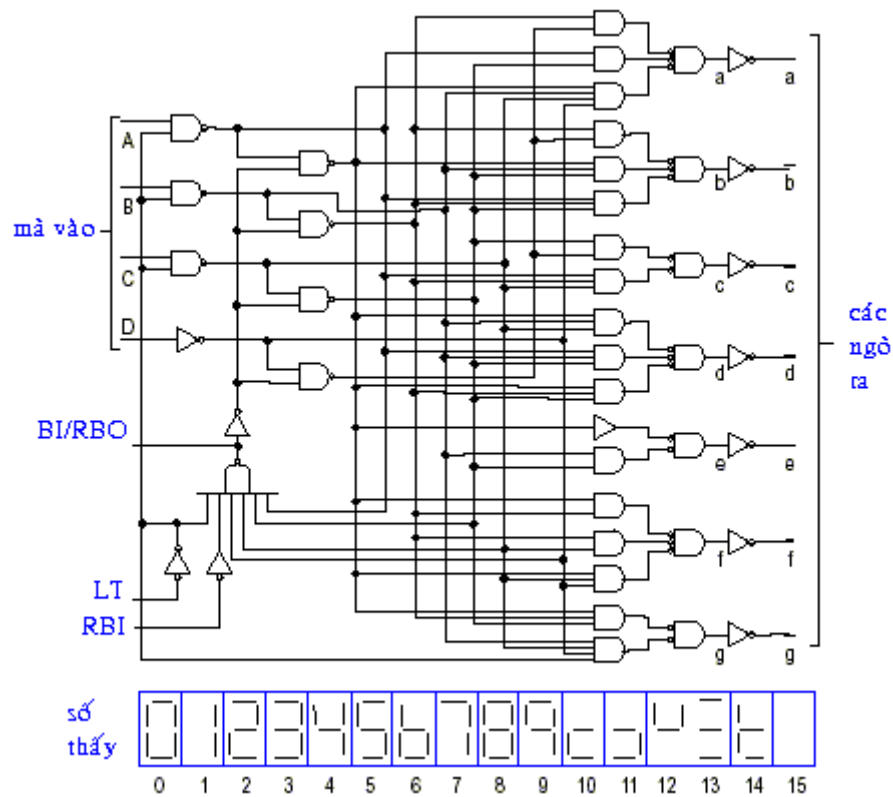
Với mạch giải mã ở trên ta có thể dùng 74LS47. Đây là IC giải mã đồng thời thúc trực tiếp led 7 đoạn loại Anode chung luôn vì nó có các đường ra cực thu để hở và khả năng nhận dòng đủ lớn. Sơ đồ chân của IC như sau:



Hình 2.57: Kí hiệu khối và chân ra 74LS47

Trong đó

- A, B, C, D là các đường vào mã BCD
- RBI là đường vào xoá dọn sóng
- LT là đường thử đèn
- BI/RBO là đường vào xoá hay đường ra xoá rợn
- a tới g là các đường ra (cực thu đèn)



Hình 2.58: Cấu trúc bên trong của 74LS47 và dạng số hiển thị

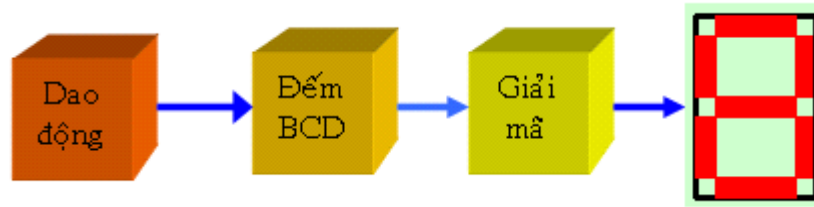
Hoạt động của IC được tóm tắt theo bảng dưới đây

Số thấy	\overline{LT}	\overline{RBI}	D	C	B	A	$\overline{BI/RBO}$	\overline{a}	\overline{b}	\overline{c}	\overline{d}	\overline{e}	\overline{f}	\overline{g}	Ghi chú
0	1	X	0	0	0	0	1	0	0	0	0	0	0	1	1
1	1	X	0	0	0	1	1	1	0	0	1	1	1	1	
2	1	X	0	0	1	0	1	0	0	1	0	0	1	0	
3	1	X	0	0	1	1	1	0	0	0	0	1	1	0	
4	1	X	0	1	0	0	1	1	0	0	1	1	0	0	
5	1	X	0	1	0	1	1	0	1	0	0	1	0	0	
6	1	X	0	1	1	0	1	1	1	0	0	0	0	0	
7	1	X	1	1	1	1	1	0	0	0	1	1	1	1	
8	1	X	1	0	0	0	1	0	0	0	0	0	0	0	
9	1	X	1	0	0	1	1	0	0	0	1	1	0	0	
10	1	X	1	0	1	0	1	1	1	1	0	0	1	0	2
11	1	X	1	0	1	1	1	1	1	0	0	1	1	0	
12	1	X	1	1	0	0	1	1	0	1	1	1	0	0	
13	1	X	1	1	0	1	1	0	1	1	0	1	0	0	
14	1	X	1	1	1	0	1	1	1	1	0	0	0	0	
15	1	X	1	1	1	1	1	1	1	1	1	1	1	1	
BI	X	X	X	X	X	X	0	1	1	1	1	1	1	1	3
RBI	1	0	0	0	0	0	0	1	1	1	1	1	1	1	4
LT	0	X	X	X	X	X	1	0	0	0	0	0	0	0	5

- Nhận thấy các đường ra mạch giải mã tác động ở mức thấp (0) thì led tương ứng sáng
- Ngoài 10 số từ 0 đến 9 được giải mã, mạch cũng còn giải mã được 6 trạng thái khác, ở đây không dùng đến (ghi chú 2)
- Để hoạt động giải mã xảy ra bình thường thì chân LT và BI/RBO phải ở mức cao
- Muốn thử đèn led để các led đều sáng hết thì kéo chân LT xuống thấp (ghi chú 5)
- Muốn xoá các số (tắt hết led) thì kéo chân BI xuống thấp (ghi chú 3)

Khi cần giải mã nhiều led 7 đoạn ta cũng có thể ghép nhiều tầng IC, muốn xoá số 0 vô nghĩa ở trước thì nối chân RBI của tầng đầu xuống thấp, khi này chân ra RBO cũng xuống thấp và được nối tới tầng sau nếu muốn xoá tiếp số 0 vô nghĩa của tầng đó (ghi chú 4). Riêng tầng cuối cũng thì RBI để trống hay để mức cao để vẫn hiển thị số 0 cuối cùng

Ví dụ: Hãy xem một ứng dụng của mạch giải mã led 7 đoạn:



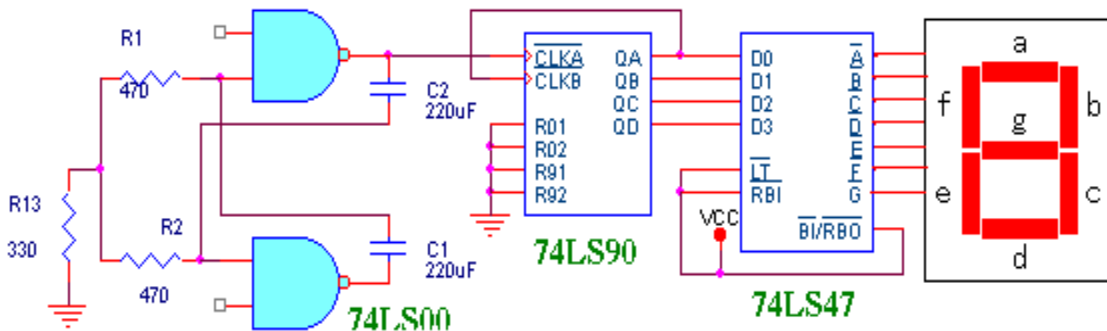
Hình 2.60. Ứng dụng mạch giải mã 74LS47

- Mạch dao động tạo ra xung kích cho mạch đếm, ta có thể điều chỉnh chu kỳ xung để mạch đếm nhanh hay chậm
- Mạch đếm tạo ra mã số đếm BCD một cách tự động đưa tới mạch giải mã có thể là cho đếm lên hay đếm xuống
- Mạch giải mã sẽ giải mã BCD sang led 7 đoạn để hiển thị số đếm thập phân

Bây giờ ta có thể thay mạch dao động bằng 1 bộ cảm biến chẳng hạn dùng bộ thu phát led đặt ở cửa vào nếu mỗi lần có 1 người vào thì bộ cảm biến sẽ tạo 1 xung kích cho mạch đếm. Lưu ý rằng IC 7490 là IC đếm chia 10 không đồng bộ mà ta sẽ học ở chương sau

Như vậy với ứng dụng này ta đã có hệ thống đếm số người vào cổng cũng có thể đếm sản phẩm qua băng truyền,... tất nhiên chỉ hạn chế ở số người vào nhiều nhất là 9.

Khi này hình trên được trình bày ở dạng mạch cụ thể như sau:

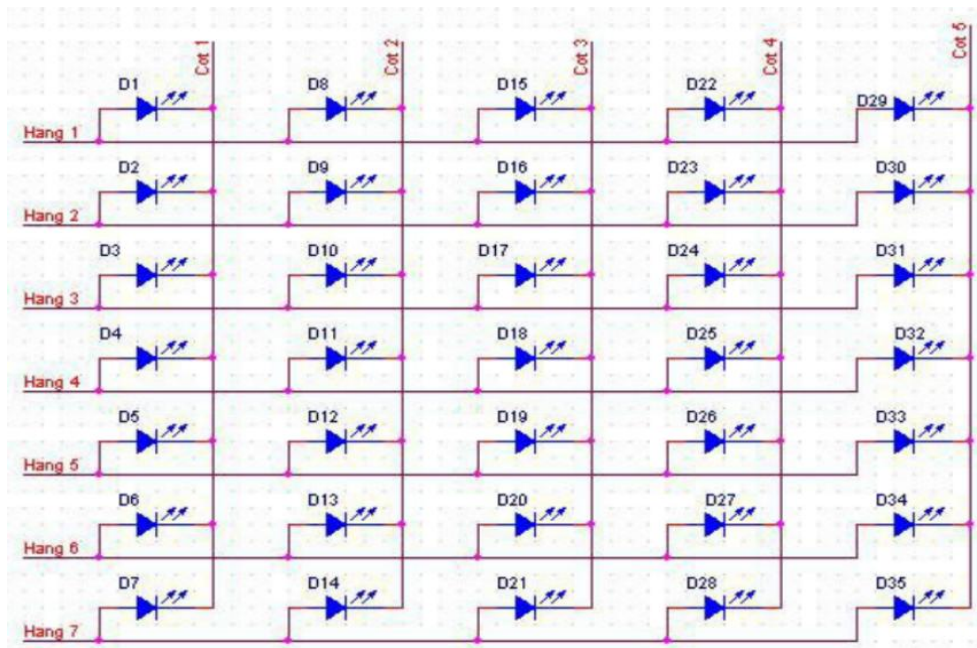


Hình 2.61: Minh họa ứng dụng 74LS47 trong mạch hiển thị led 7 đoạn

2.9. Thiết kế mạch điều khiển ma trận LED 5x7

2.9.1. Cấu tạo ma trận LED 5x7

Ma trận LED 5x3 là một ma trận gồm 35 đèn LED được sắp xếp thành 7 hàng và 5 cột, các đèn LED này được nối chung với nhau bởi Anot hoặc Katot. Tại mỗi giao điểm của hàng với cột là một LED đơn



Hình 2. 44: Sơ đồ cấu trúc bên trong ma trận LED 5x7 Catốt chung

2.9.2. Điều khiển ma trận LED 5x7

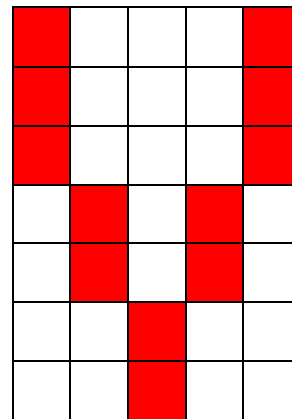
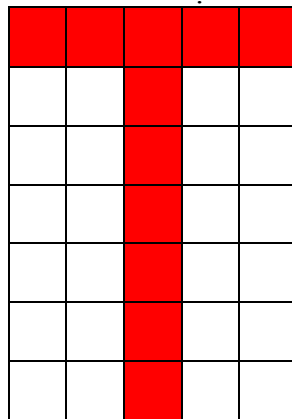
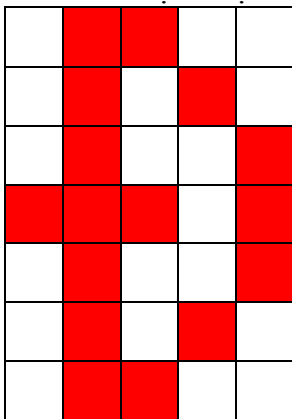
Trong phần này chúng ta chỉ đi tìm hiểu cách thức điều khiển ma trận led để hiển thị các chữ cái, các con số hoặc các ký hiệu tĩnh.

Trước tiên ta xác định tất cả những nội dung có thể được hiển thị trên ma trận để xác định với từng nội dung hiển thị, đèn LED nào trong ma trận sẽ được sáng và đèn nào không được sáng. Từ những kết quả giải mã đó ta xác định được nhóm các đèn LED sẽ sáng cho một hoặc một số nội dung được hiển thị hay nói khác đi ta xem có các đèn nào hoạt động giống nhau thì nhóm thành một nhóm. Như vậy thay vì phải điều khiển $7 \times 5 = 35$ đèn LED đơn ta có thể chỉ phải điều khiển một số lượng nhóm đèn nào đó ít hơn nhiều.

Để điều khiển được ma trận LED điều đầu tiên cần tính được mã của LED. Tủy theo mỗi loại khác nhau để chung ta vào hàng và cột là các mức logic 0 hay 1. Nguyên lý quét dựa vào hiện tượng lưu ảnh trên võng mạc và con người chỉ nhìn được 24h/s. Tại mỗi thời điểm chỉ có một điểm sáng, do tần số quét nhanh nên mắt con người cảm nhận được LED quét đồng thời. Khi muốn LED nào sang chúng ta cần phải đưa tín hiệu vào điều khiển cho LED đó.

2.3.7.3. Ví dụ

Thiết kế mạch hiển thị chữ ĐTVT trên ma trận LED



VT	ĐT	ĐT	T	VT
V	Đ	T	Đ	V
V	Đ	T		ĐV
Đ	ĐV	ĐT	V	Đ
	ĐV	T	V	Đ
	Đ	VT	Đ	
	Đ	ĐTV		

ĐTV: Y1
 ĐT: Y2
 ĐV: Y3
 VT: Y4
 Đ: Y5
 T: Y6
 V: Y7

Ta có bảng chân lý sau:

	A	B	Y1	Y2	Y3	Y4	Y5	Y6	Y7
Đ	0	0	1	1	1	0	1	0	0
T	0	1	1	1	0	1	0	1	0
V	1	0	1	0	1	1	0	0	1
T	1	1	1	1	0	1	0	1	0

Từ bảng chân lý ta có được các phương trình của lối ra Y và có thể vẽ được sơ đồ mạch.

Chương 3. Mạch tuần tự

3.1. Khái niệm chung

Trigơ (Flip - Flop) là phân tử cơ bản nhất để từ đó chế tạo ra các mạch dãy (mạch logic có nhớ). Mạch Trigơ thuộc loại mạch không đồng bộ có hai trạng thái ổn định bền theo thời gian ứng với hai mức logic "1" và "0". Trạng thái của Trigơ có thể thay đổi khi tác động xung lên các đầu vào. Trạng thái tương lai của Trigơ không những phụ thuộc vào các biến vào mà còn phụ thuộc vào trạng thái hiện tại. Khi ngừng tác động xung lên các đầu vào, trạng thái Trigơ giữ nguyên, với đặc điểm này các mạch Trigơ được dùng để lưu trữ thông tin dưới dạng mã nhị phân.

3.2. Các loại Trigơ

3.2.1 Trigơ R-S không đồng bộ

Là loại Trigơ cơ bản nhất để từ đó tạo ra các loại Trigơ khác gồm có 2 đầu vào là R, S và hai đầu ra Q, \bar{Q} với:

- Q: Đầu ra chính thông dụng sử dụng.
- \bar{Q} : Đầu ra phụ, luôn thoả mãn $Q + \bar{Q} = 1$
- R (Reset): Đầu vào xoá.
- S (Set): Đầu vào thiết lập.

Ý tưởng thiết kế trigơ R-S không đồng bộ theo các điều kiện sau:

$$+ R_n = S_n = 0, \text{ trạng thái của trigơ giữ nguyên } \rightarrow Q_{n+1} = Q_n.$$

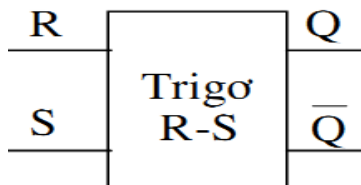
$$+ R_n = 0; S_n = 1 \text{ đầu ra trigơ nhận giá trị "1"} \rightarrow Q_{n+1} = 1.$$

$$+ R_n = 1; S_n = 0 \text{ đầu ra trigơ nhận giá trị "0"} \rightarrow Q_{n+1} = 0.$$

+ $R_n = 1; S_n = 1$ đây là trạng thái cấm, trạng thái Trigơ là không xác định, trong bảng trạng thái được đánh dấu bằng dấu "x".

Hoạt động của trigơ R-S tuân theo bảng trạng thái nh hình vẽ.

- n: Trạng thái hiện tại
- n + 1: Trạng thái tương lai.
- "-": Giá trị tùy chọn - có thể lấy giá trị "1" hoặc "0".
- x: Trạng thái cấm tại đó giá trị của hàm ra là không xác định.



Hình 3.1: Sơ đồ mô phỏng

R_n	S_n	Q_{n+1}
0	0	Q_n
0	1	1
1	0	0
1	1	x

Bảng trạng thái

Q_{n+1}	R_n, S_n			
Q_n	00	01	11	10
0	0	1	x	0
1	1	1	x	0

Bảng chuyển tiếp

Q_n	Q_{n+1}	R_n	S_n
0	0	-	0
0	1	0	1
1	0	1	0
1	1	0	-

Bảng đầu vào kích

Thực hiện nhóm các ô có giá trị “1” trong bảng trạng thái (dạng tuyển) ta có:

$$Q_{n+1} = S_n + \overline{R_n} \cdot Q_n \quad (1)$$

Nhóm các ô có giá trị “0” trong bảng trạng thái (dạng hội) ta có:

$$Q_{n+1} = \overline{R_n} \cdot (Q_n + S_n) \quad (2)$$

$$\text{Từ (1)} \rightarrow Q_{n+1} = \overline{\overline{S_n + Q_n \cdot \overline{R_n}}} = \overline{\overline{S_n} \cdot \overline{Q_n} \cdot \overline{R_n}} \quad (3)$$

$$\text{Từ (2)} \rightarrow Q_{n+1} = \overline{\overline{R_n} \cdot (Q_n + S_n)}} = \overline{\overline{R_n} \cdot \overline{Q_n} \cdot \overline{S_n}}$$

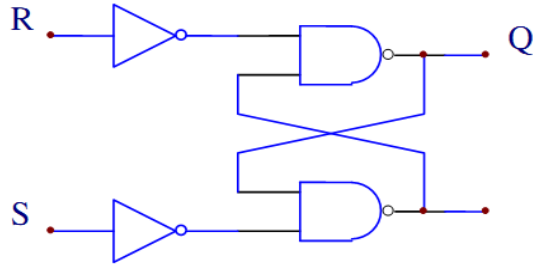
$$\rightarrow \overline{Q_{n+1}} = \overline{\overline{R_n} \cdot (\overline{Q_n} \cdot \overline{S_n})} \quad (4)$$

$$\text{Từ (2)} \rightarrow Q_{n+1} = \overline{\overline{R_n} \cdot (Q_n + S_n)}} = \overline{\overline{R_n} + \overline{(Q_n + S_n)}} \quad (5)$$

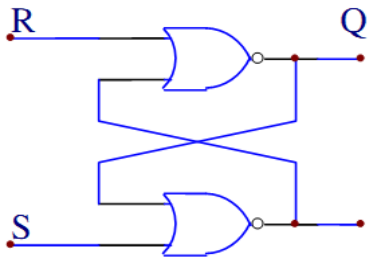
$$\text{Từ (1)} \rightarrow Q_{n+1} = \overline{\overline{S_n + R_n \cdot Q_n}}} = \overline{\overline{S_n} + \overline{Q_n} + \overline{R_n}}$$

$$\overline{Q_{n+1}} = \overline{\overline{\overline{S_n} + \overline{Q_n} + \overline{R_n}}}} \quad (6)$$

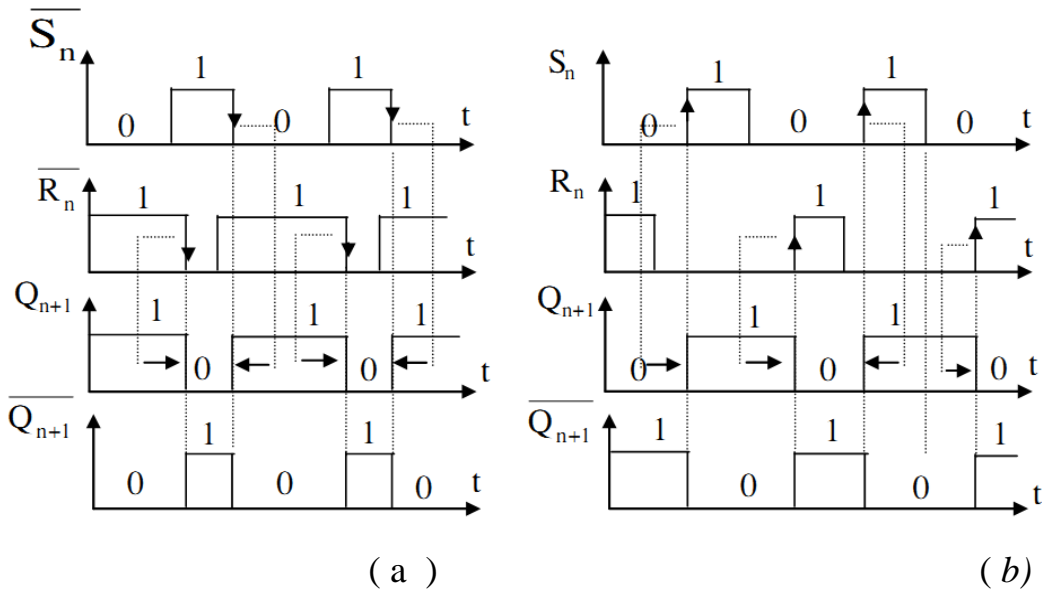
Từ (3) và (4), (5) và (6) cho phép ta xây dựng Trigơ RS không đồng bộ từ các phân tử NAND, các phân tử NOR hai lối vào.



Hình 3.2: Sơ đồ Trơ R-S dùng phân tử NAND



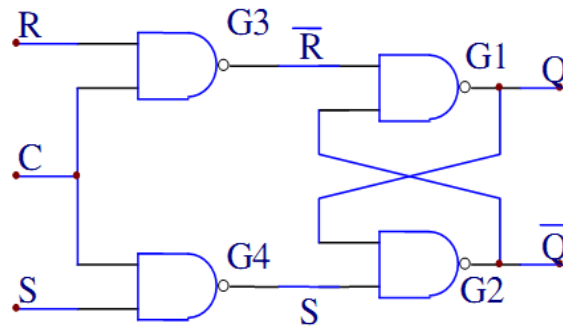
Hình 3.3: Sơ đồ Trơ R-S dùng phân tử NOR



Hình 3.4: Giải đồ điện áp làm việc của Trơ R-S tử NAND (a), tử NOR (b)

3.2.2. Trơ R-S đồng bộ.

Ng-ời ta muốn Trơ chỉ phản ứng vào những thời điểm xác định, điều này đ-ợc thực hiện bằng cách đ- a thêm tới đầu vào tín hiệu phụ C đ-ợc gọi là tín hiệu đồng bộ. Khi $C = "0"$ thì $\bar{R} = \bar{S} = 1$ trạng thái Trơ giữ nguyên còn $C = "1"$ hoạt động của sơ đồ giống Trơ R-S không đồng bộ nh- đã phân tích ở phần trên.



Hình 3.5: Trơ R-S đồng bộ

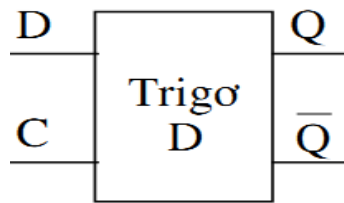
3.2.3. Trơ D (Delay)

Gồm có hai đầu vào C, D_n và hai đầu ra Q_n, Q̄_n với:

- C: Biến điều khiển (xung nhịp - xung đồng bộ)
- D_n: Dữ liệu vào.

ý t-ởng thiết kế trơ D tuân theo các điều kiện sau:

- + Khi C = 0, trạng thái của trơ giữ nguyên → Q_{n+1} = Q_n.
- + Khi C = 1, giá trị đầu ra trơ nhận giá trị đã đến đầu vào D → Q_{n+1} = D_n.



Hình 3.6: Sơ đồ mô phỏng

C	D _n	Q _{n+1}
0	0	Q _n
0	1	Q _n
1	0	0
1	1	1

Bảng trạng thái

Q _{n+1} C, D _n Q _n	Q _n			
	00	01	11	10
0	0	0	1	0
1	1	1	1	0

Bảng chuyển tiếp

Q _n	Q _{n+1}	C	D
0	0	0	-
0	1	1	1
1	0	1	0
1	1	0	-

Bảng đầu vào kích

Thực hiện nhóm các ô có giá trị “1” trong bảng trạng thái (dạng tuyến) ta có:

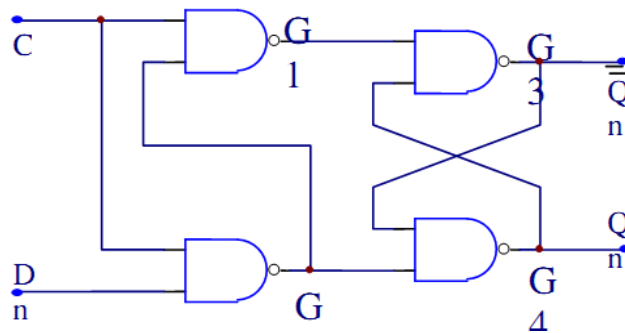
$$Q_{n+1} = C.D_n + Q_n.\bar{C} + Q_n.D_n = C.D_n + Q_n[\overline{\bar{C} + D_n}] = C.D_n + Q_n.\bar{C}.\bar{D}_n$$

$$= \overline{\overline{C.D_n + Q_n.C.C.D_n}} = \overline{\overline{C.D_n} . \overline{Q_n.C.C.D_n}} \quad (1)$$

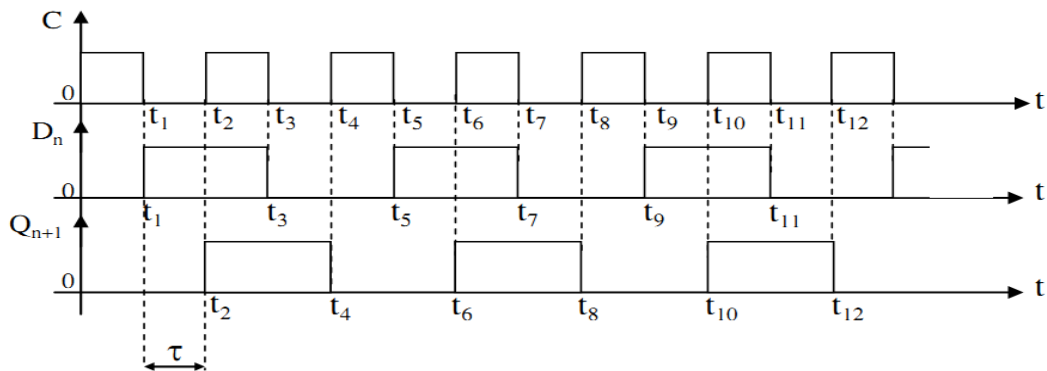
Thực hiện nhóm các ô có giá trị “0” trong bảng trạng thái (dạng hội) ta có:

$$\begin{aligned} Q_{n+1} &= [\overline{C + D_n}][Q_n + C][Q_n + D_n] = [\overline{C + D_n}][Q_n + Q_n.D_n + Q_n.C + C.D_n] \\ &= [\overline{C + D_n}][Q_n(1 + D_n + C) + C.D_n] = \overline{(\overline{C + D_n})(Q_n + C.D_n)} = \overline{C.D_n} . \overline{Q_n.C.D_n} = \overline{C.C.D_n} . \overline{Q_n.C.D_n} \\ &\Rightarrow \overline{Q_{n+1}} = \overline{C.C.D_n} . \overline{Q_n.C.D_n}. \quad (2) \end{aligned}$$

Từ các biểu thức (1) và (2) ta có mạch điện của trigơ D được xây dựng từ các phần tử NAND hai lối vào.



Hình 3.7: Trigơ D xây dựng từ phần tử NAND



Hình 3.8: Giản đồ xung mô tả quá trình hoạt động Trigơ D

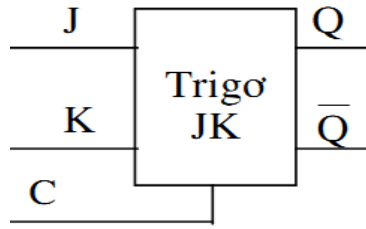
3.2.4. Trigơ vạn năng J-K

Gồm có 3 đầu vào C, J_n, K_n và hai đầu ra Q_n, $\overline{Q_n}$ với:

- C: Xung đồng bộ.
- J_n, K_n: Các đầu vào điều khiển.

ý t- ởng thiết kế trigơ vạn năng J, K theo các điều kiện sau:

- + K_n = J_n = 0, trạng thái của trigơ giữ nguyên → Q_{n+1} = Q_n.
- + K_n = 0; J_n = 1 đầu ra trigơ nhận giá trị "1" → Q_{n+1} = 1.
- + K_n = 1; J_n = 0 đầu ra trigơ nhận giá trị "0" → Q_{n+1} = 0.
- + K_n = 1; J_n = 1 trigơ lật trạng thái → Q_{n+1} = $\overline{Q_n}$.



Hình 3.9: Sơ đồ mô phỏng

K_n	J_n	Q_{n+1}
0	0	Q_n
0	1	1
1	0	0
1	1	$\overline{Q_n}$

Bảng trạng thái

$Q_{n+1} \backslash Q_n$	$K_n J_n$	00	01	11	10
0	0	0	1	1	0
1	1	1	1	0	0

Bảng chuyển tiếp

Q_n	Q_{n+1}	K_n	J_n
0	0	-	0
0	1	-	1
1	0	1	-
1	1	0	-

Bảng đầu vào kích

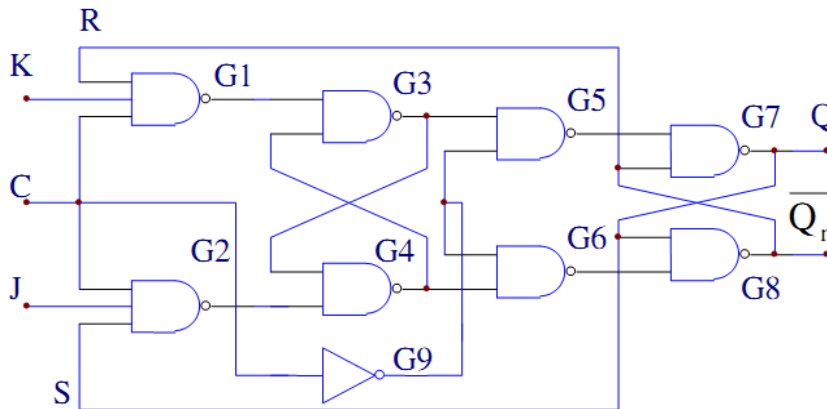
Thực hiện nhóm các ô có giá trị “1” trong bảng trạng thái (dạng tuyển) ta có:

$$Q_{n+1} = \overline{K_n} \cdot Q_n + J_n \cdot \overline{Q_n} \quad (1)$$

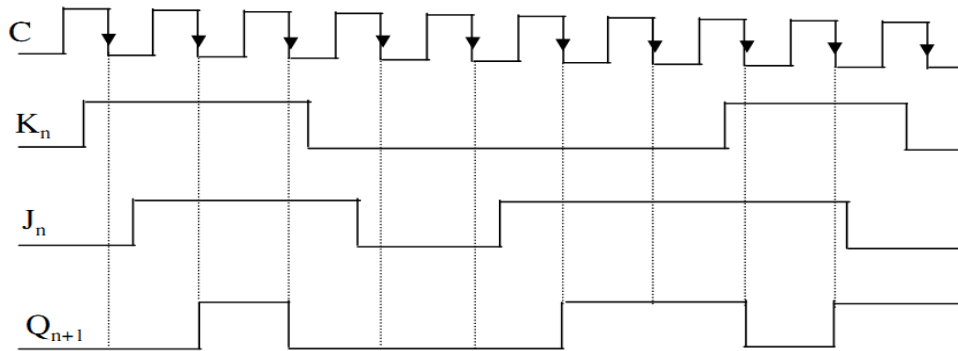
Nhóm các ô có giá trị “0” trong bảng trạng thái (dạng hội) ta có:

$$Q_{n+1} = (\overline{K_n} + \overline{Q_n}) \cdot (J_n + Q_n) \quad (2)$$

Các biểu thức (1) hoặc (2) đ- ợc gọi là ph- ơng trình đặc tính của Trigơ vạn năng J-K



Hình 3.10: Sơ đồ mạch logic Trigơ J-K



Hình 3.11: Sơ đồ xung mô tả quá trình làm việc Trigo J-K

3.2.5. Trigo đếm T

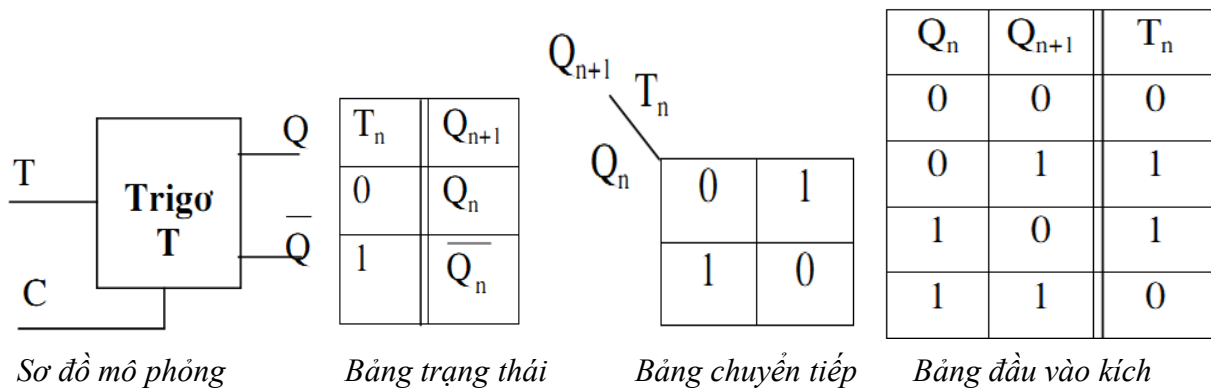
Có hai đầu vào T, C, hai đầu ra Q, \bar{Q} .

-C: Xung đồng bộ (xung nhịp).

-T: Biến điều khiển, thoả mãn yêu cầu sau:

+ T = "0" trạng thái Trigo giữ nguyên $\rightarrow Q_{n+1} = Q_n$.

+ T = "1" Trigo lật trạng thái $\rightarrow Q_{n+1} = \bar{Q}_n$.



Sơ đồ mô phỏng

Bảng trạng thái

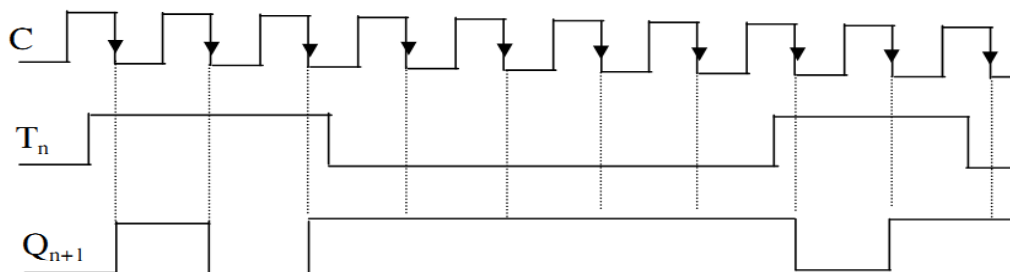
Bảng chuyển tiếp

Bảng đầu vào kích

Từ bảng chuyển tiếp ta có: $Q_{n+1} = T_n \cdot \bar{Q}_n + \bar{T}_n \cdot Q_n$ (1)

$$Q_{n+1} = (T_n + Q_n) \cdot (\bar{T}_n + \bar{Q}_n) \quad (2)$$

Biểu thức (1) và (2) đ- ọc gọi là ph- ong trình đặc tính của Trigo đếm T.



Hình 3.12: Sơ đồ xung mô tả quá trình làm việc Trigo T

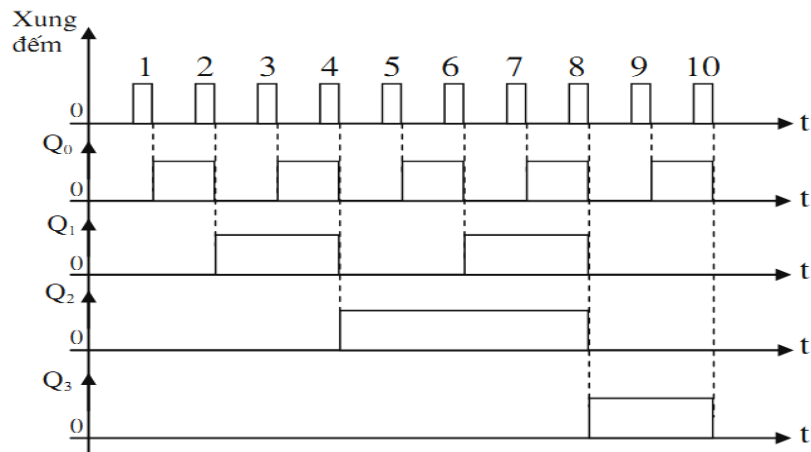
3.3. Thiết kế bộ đếm đồng bộ

3.3.1. Các bước thiết kế bộ đếm đồng bộ

- Phân tích yêu cầu thiết kế, vẽ giản đồ xung mô tả
- Xác định loại trigơ, số lượng trigơ
- Lập bảng trạng thái của bộ đếm
- Lập phương trình trạng thái
- Vẽ sơ đồ logic

3.3.2. Thiết kế bộ đếm tiến thập phân đồng bộ

- Bước 1: Vẽ giản đồ xung cho bộ đếm



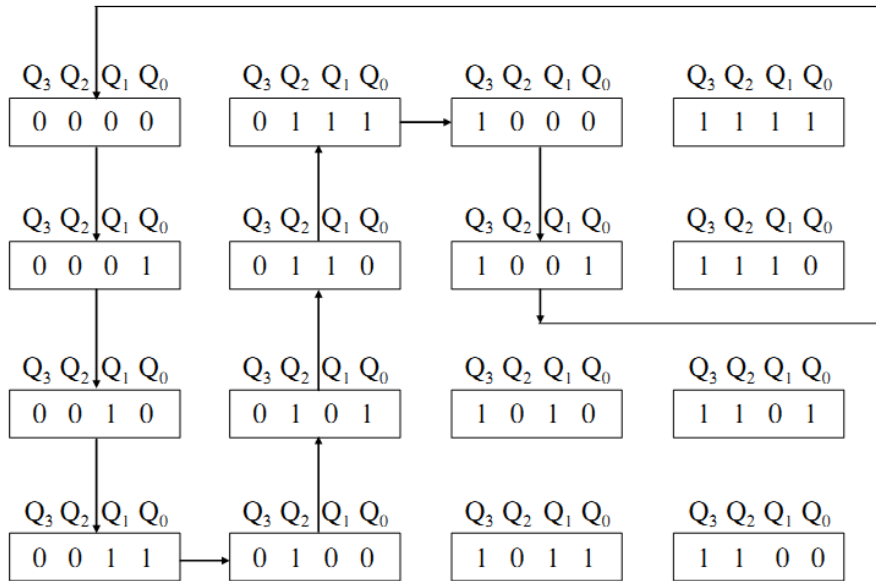
Hình 3.18: Giản đồ xung của bộ đếm module 10

- Bước 2: Lập bảng trạng thái cho bộ đếm

Xung vào	Trạng thái			
	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	0	0	0	0

Hình 3.19: Bảng trạng thái của bộ đếm module 10

- Bước 3: Lập đồ hình chuyển đổi trạng thái



Hình 3.20: Đồ hình chuyển đổi trạng thái của bộ đếm

- Bước 4 : Lập mối quan hệ đầu vào theo đầu ra

Xung đếm	Trạng thái các trigơ đếm								Trạng thái các hàm đầu vào kích của trigơ							
	Hiện tại				Tiếp theo											
	Q_3	Q_2	Q_1	Q_0	Q_3'	Q_2'	Q_1'	Q_0'	J_3	K_3	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	0	0	0	1	0	-	0	-	0	-	1	-
1	0	0	0	1	0	0	1	0	0	-	0	-	1	-	-	1
2	0	0	1	0	0	0	1	1	0	-	0	-	-	0	1	-
3	0	0	1	1	0	1	0	0	1	-	1	-	-	1	-	1
4	0	1	0	0	0	1	0	1	0	-	-	0	0	-	1	-
5	0	1	0	1	0	1	1	0	0	-	-	0	1	-	-	1
6	0	1	1	0	0	1	1	1	0	-	-	0	-	0	1	-
7	0	1	1	1	1	0	0	0	1	-	-	1	-	1	-	1
8	1	0	0	0	1	0	0	1	-	0	0	-	0	-	1	-
9	1	0	0	1	0	0	0	0	-	1	0	-	0	-	-	1

Hình 3.21: Bảng trạng thái minh họa quá trình làm việc của bộ đếm module 10

- Bước 5: Tối giản các hàm đầu vào

J_0		Q_1Q_0			
		Q_3Q_2	00	01	11
00	01	11	10		
00	01	11	10		
11	10				
10					

$$J_0 = 1$$

K_0		Q_1Q_0			
		Q_3Q_2	00	01	11
00	01	11	10		
00	01	11	10		
11	10				
10					

$$K_0 = 1$$

J_1		Q_1Q_0			
		Q_3Q_2	00	01	11
00	01	11	10		
00	01	11	10		
11	10				
10					

$$J_1 = \overline{Q_3} \cdot Q_0$$

K_1		Q_1Q_0			
		Q_3Q_2	00	01	11
00	01	11	10		
00	01	11	10		
11	10				
10					

$$K_1 = Q_0$$

J_2		Q_1Q_0			
		Q_3Q_2	00	01	11
00	01	11	10		
00	01	11	10		
11	10				
10					

$$J_2 = Q_1 \cdot Q_0$$

K_2		Q_1Q_0			
		Q_3Q_2	00	01	11
00	01	11	10		
00	01	11	10		
11	10				
10					

$$K_2 = Q_1 \cdot Q_0$$

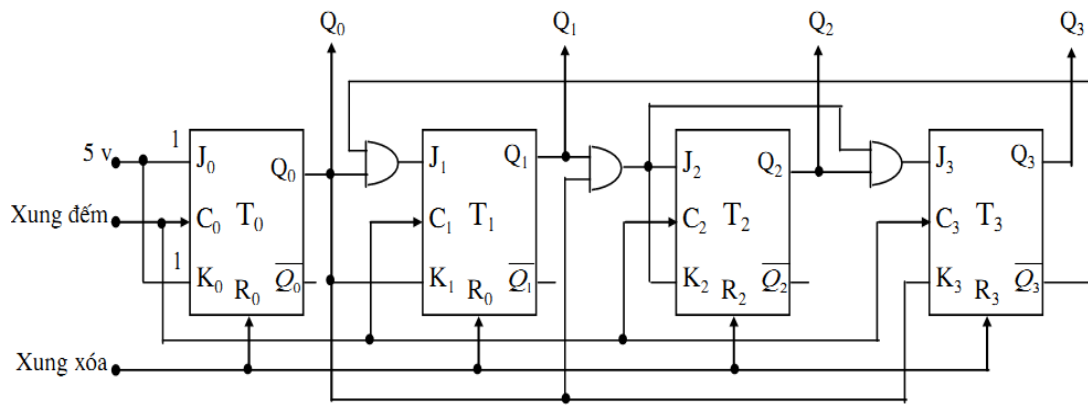
J_3		Q_1Q_0			
		Q_3Q_2	00	01	11
00	01	11	10		
00	01	11	10		
11	10				
10					

$$J_3 = Q_1 \cdot Q_0 \cdot Q_2$$

K_3		Q_1Q_0			
		Q_3Q_2	00	01	11
00	01	11	10		
00	01	11	10		
11	10				
10					

$$K_3 = Q_0$$

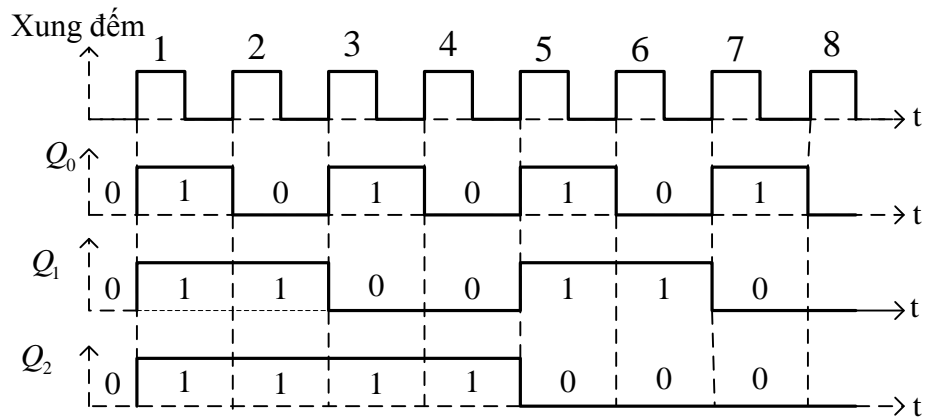
- Bước 6 : Vẽ sơ đồ mạch của bộ đếm



Hình 3.22: Sơ đồ mạch bộ đếm module 10

3.3.3. Thiết kế bộ đếm lùi module 8

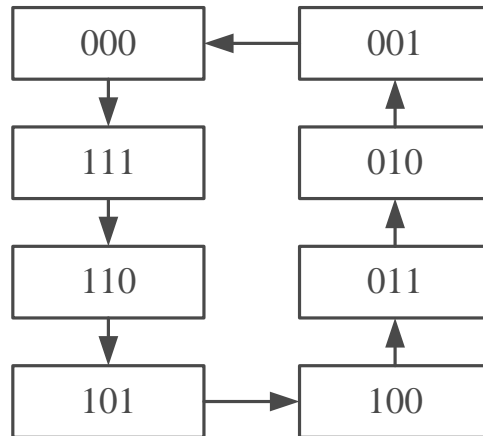
Giải đồ xung của bộ đếm



Bảng trạng thái của bộ đếm

Xung	Q_2	Q_1	Q_0
0	0	0	0
1	1	1	1
2	1	1	0
3	1	0	1
4	1	0	0
5	0	1	1
6	0	1	0
7	0	0	1

Đồ hình chuyển đổi trạng thái của bộ đếm



Để thiết kế bộ đếm này ta dùng trigơ JK.

Bảng trạng thái hoạt động của bộ đếm

Xung Đếm	Trạng thái của trigơ đếm						Trạng thái đầu vào kích của trigơ đếm					
	Hiện tại			Tiếp theo			K_2	J_2	K_1	J_1	K_0	J_0
	Q_2	Q_1	Q_0	Q_2'	Q_1'	Q_0'						
0	0	0	0	1	1	1	-	1	-	1	-	1
7	1	1	1	1	1	0	0	-	0	-	1	-
6	1	1	0	1	0	1	0	-	1	-	-	1
5	1	0	1	1	0	0	0	-	-	0	1	-
4	1	0	0	0	1	1	1	-	-	1	-	1
3	0	1	1	0	1	0	-	0	0	-	1	-
2	0	1	0	0	0	1	-	0	1	-	-	1
1	0	0	1	0	0	0	-	0	-	0	1	

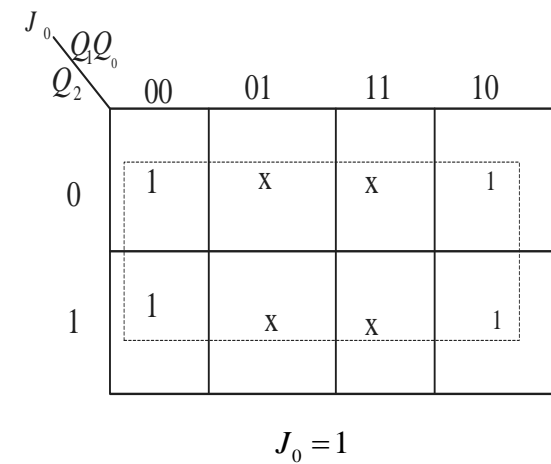
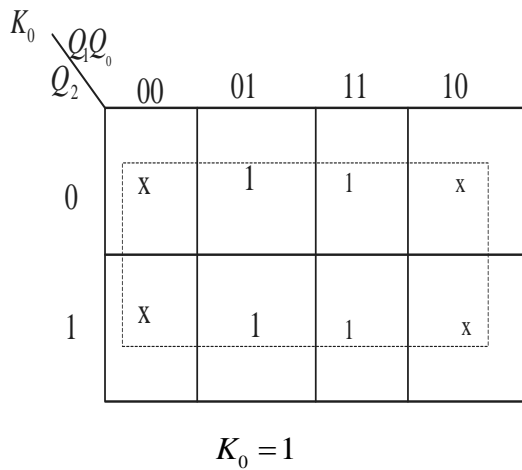
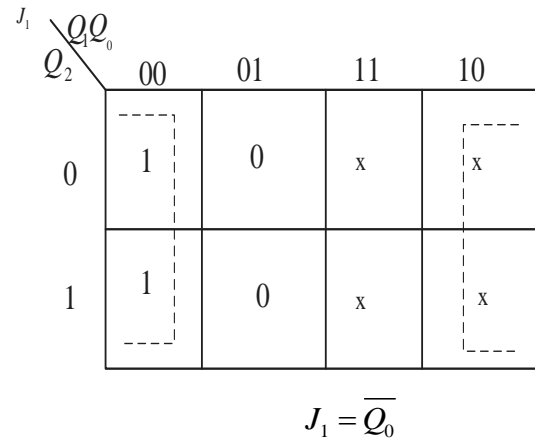
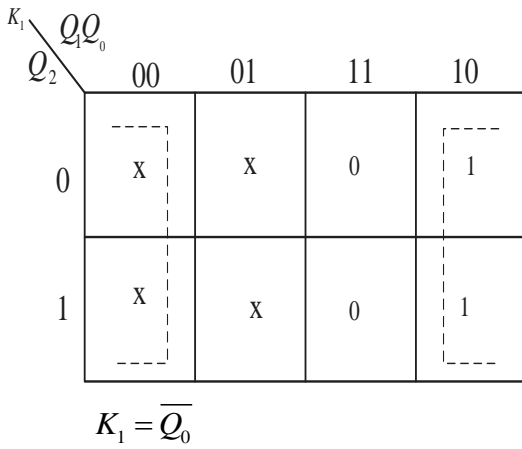
Tối giản hàm

K_2	$Q_1 Q_0$				
Q_2		00	01	11	10
0		X	x	x	X
1		1	0	0	0

$$K_2 = \overline{Q_1} \cdot \overline{Q_0}$$

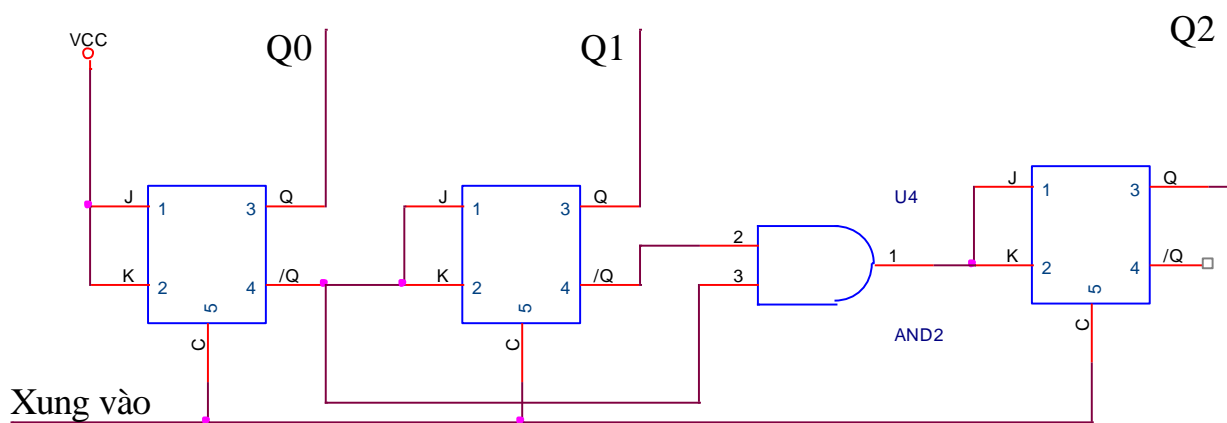
J_2	$Q_1 Q_0$				
Q_2		00	01	11	10
0		1	0	0	0
1		x	x	x	x

$$J_2 = \overline{Q_1} \cdot \overline{Q_0}$$



Ta có phương trình trạng thái bộ đếm nghịch module 8 như sau:

$$K_0 = 1 ; J_0 = 1 ; K_1 = \overline{Q_0} ; J_1 = \overline{Q_0} ; K_2 = \overline{Q_1} \cdot \overline{Q_0} ; J_2 = \overline{Q_1} \cdot \overline{Q_0}$$



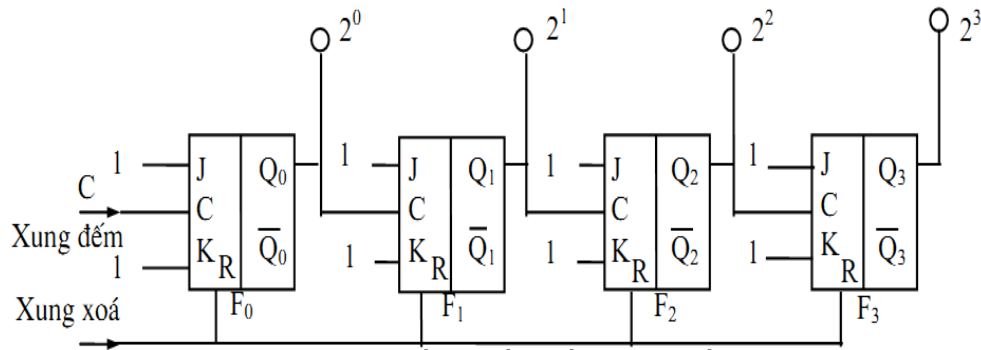
Sơ đồ mạch logic bộ lùi đồng bộ

3.4. Bộ đếm không đồng bộ

3.4.1. Bộ đếm tiến

Bộ đếm tiến không đồng bộ là bộ đếm mà ta ghép nối tiếp các trigơ với nhau, xung cần đếm được đưa vào một cách tuần tự tại lối vào động bộ (cửa C) của

Trigơ đầu tiên, đầu ra của trigơ trước được nối với đầu vào đồng bộ C của trigơ tiếp theo cấp cao hơn (Q_i được nối với C_{i+1}).

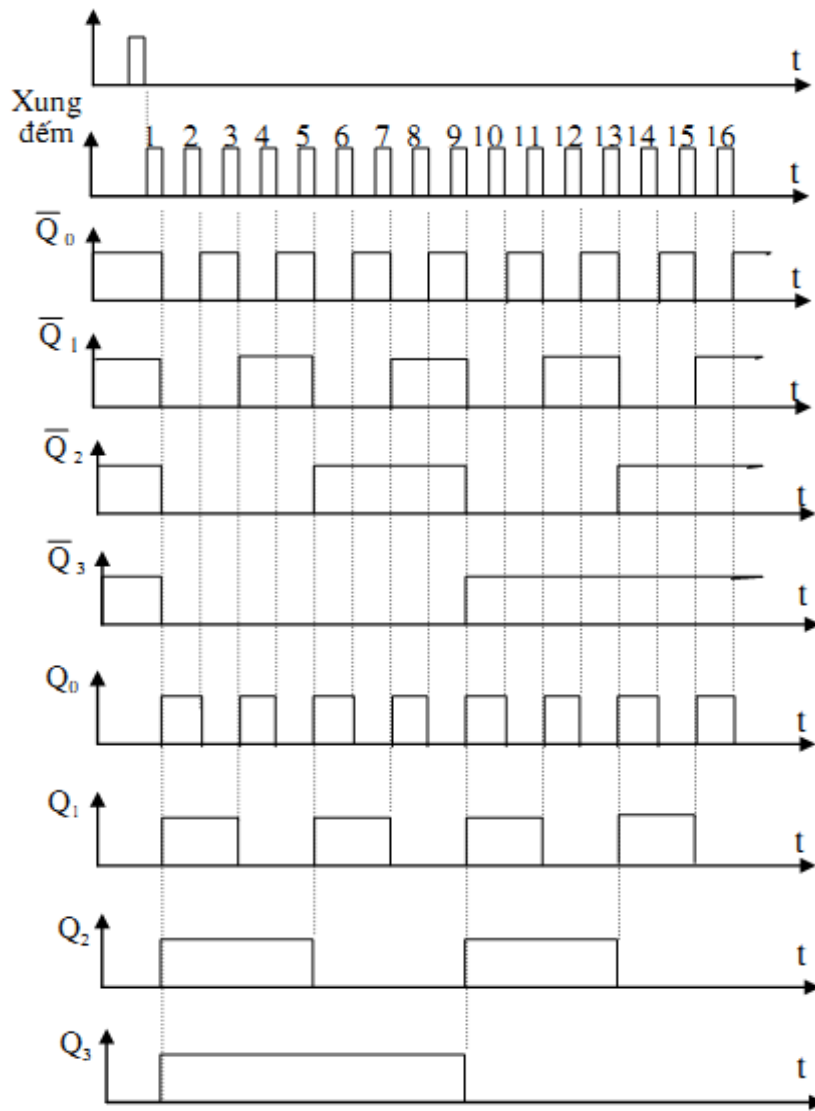


Hình 3.13: Sơ đồ bộ đếm tiến không đồng bộ

- Xung xoá phải xuất hiện trước dãy xung đếm để thiết lập trạng thái ban đầu $Q_0 = Q_1 = Q_2 = Q_3 = "0"$.
- Để trạng thái của trigơ bất kỳ chỉ lật khi đầu ra Q của trigơ cấp thấp hơn kề nó chuyển đổi từ "1" về "0" thì các đầu vào điều khiển của các trigơ phải cùng nhận trị "1" ($J=K=1$).
- Qua mỗi trigơ F_i thực hiện chia đôi tần số của dãy xung vào.
- Để tạo ra bộ đếm có dung lượng lớn ta cần tăng số trigơ (số bit) khi đó do có hiện tượng trễ tích lũy giữa dãy xung vào và dãy xung ra làm giảm khả năng đếm nhanh khi số bit tăng dần, độ trễ tích lũy chung bằng tổng độ trễ do các trigơ tạo nên. Đây cũng chính là nhược điểm chính của bộ đếm nhị phân nối tiếp (không đồng bộ).

3.4.2. Thiết kế bộ đếm lùi

Giản đồ xung của bộ đếm:



Hình 3.14: Giải đồ xung bộ đếm lùi

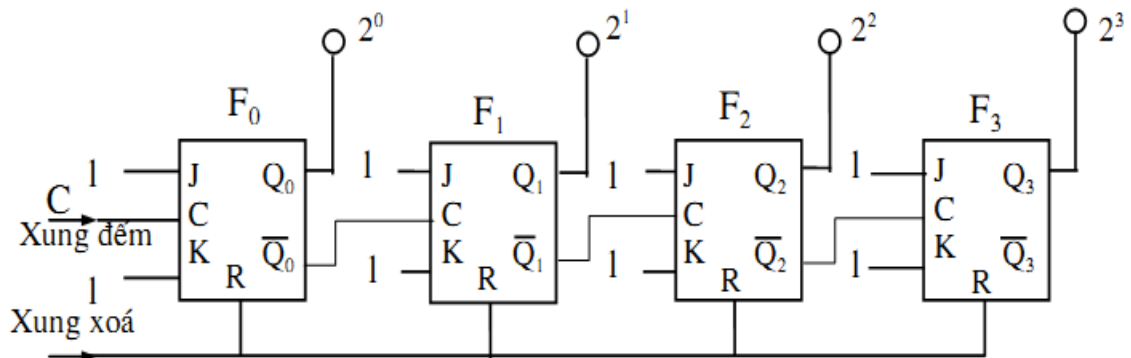
Nguyên lý làm việc t-ong tự nh- bộ đếm nhị phân thuận, giá trị nhị phân của bộ đếm giảm dần khi có xung đ-a tới, ở bộ đếm nhị phân ng-ợc nối tiếp mà các trigơ đ-ợc xây dựng từ các phần tử NAND ng-ời ta thực hiện nối \bar{Q}_i với C_{i+1}

Bảng trạng thái của bộ đếm ngược:

Số xung vào	Trạng thái trigơ đếm			
	F_3	F_2	F_1	F_0
0	0	0	0	0
1	1	1	1	1
2	1	1	1	0
3	1	1	0	1

4	1	1	0	0
5	1	0	1	1
6	1	0	1	0
7	1	0	0	1
8	1	0	0	0
9	0	1	1	1
10	0	1	1	0
11	0	1	0	1
12	0	1	0	0
13	0	0	1	1
14	0	0	1	0
15	0	0	0	1
16	0	0	0	0

Hình 3.15: Bảng trạng thái bộ đếm

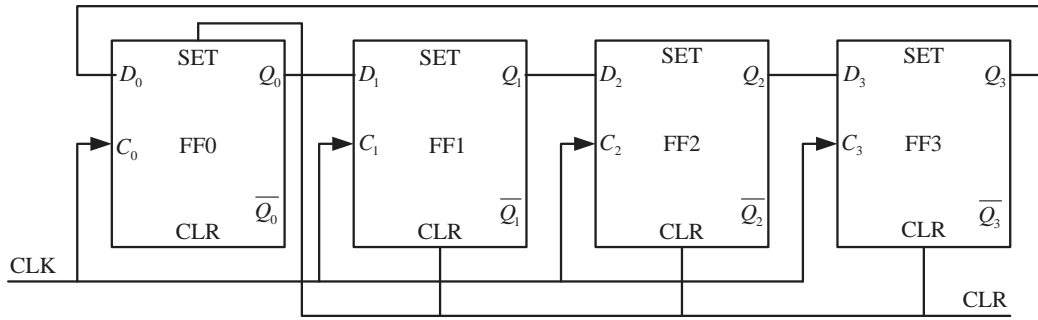


Hình 3.16: Sơ đồ mạch đếm lùi

3.5. Mạch đếm vòng

3.5.1 Đếm vòng

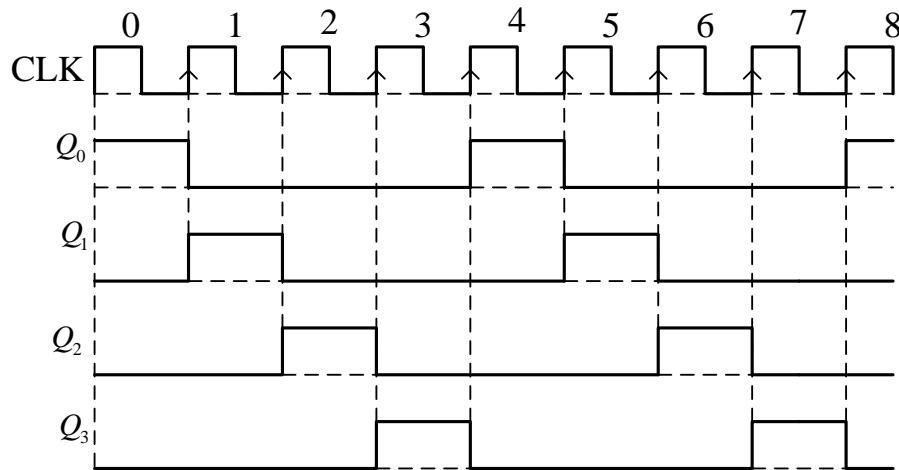
Mạch đếm vòng có cấu trúc cơ bản là thanh ghi dịch với đường ra tầng sau cùng được đưa về đường vào tầng đầu. Hình dưới là mạch đếm vòng 4 bit dùng FF D.



Hình 3.23: Mạch đếm vòng 4 bit

Nhưng để ý rằng, khi mới bật nguồn cho mạch đếm chạy, ta không biết bit 1 nằm ở đường ra của tầng nào. Do đó, cần phải xác lập dữ liệu dịch chuyển ban đầu cho bộ đếm. Ta có thể dùng đường Pr và Cl để làm, như là đã từng dùng để đặt số đếm cho các mạch đếm khác đã nói ở trước, giả sử trạng thái ban đầu là 1000 vậy ta có thể reset tầng FF 3 để đặt Q₃ mức 1, các tầng khác thì xóa bằng clear.

Giả sử ban đầu chỉ cho D₀ = 1, các đường vào tầng FF khác là 0. Bây giờ cấp xung ck đồng bộ khi ck lên cao, dữ liệu 1000 được dịch sang phải 1 tầng do đó Q₀ = 1, các đường ra khác là 0. Tiếp tục cho ck xuống thấp lần nữa, Q₁ sẽ lên 1, các đường ra khác là 0. Như vậy sau 4 nhịp xung ck thì Q₃ lên 1 và đưa về làm D₀ = 1. mạch đã thực hiện xong 1 chu trình. Trạng thái các đường ra của mạch như hình sau:



Hình 3.24. Dạng sóng minh họa mạch đếm vòng

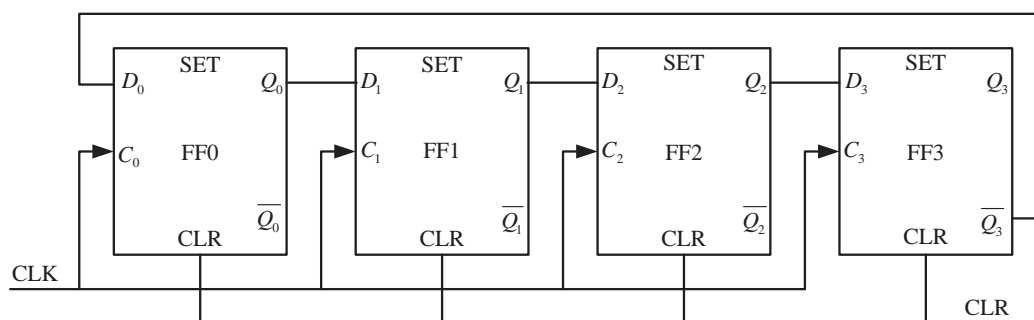
Hình trên cho thấy rằng, dạng sóng các đường ra là sóng vuông, dịch vòng quanh, chu kì như nhau nhưng lệch nhau đúng 1 chu kì xung vào Ck. Số đếm ra là 1, 2, 4, 8 không phải là số xung vào (như bảng trạng thái đếm phía dưới)

Với 4 số đếm ra từ 4 tầng FF ta có mạch đếm mod 4. Chỉ 4 trạng thái ra trong tổng số 16 trạng thái có thể, điều này làm giảm hiệu quả sử dụng của mạch đếm vòng. Nhưng nó cũng có ưu điểm nổi bật so với mạch đếm chia hệ 2 là không cần mạch giải mã trong cấu trúc mạch (vì thường trong trạng thái của số đếm ra chỉ có 1 bit 1) .

Clock Pulse	Q3	Q2	Q1	Q0
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	1	0	0	0

CLEAR	FF0	FF1	FF2	FF3
	1	0	0	0

3.5.2. Đếm Johnson (đếm vòng xoắn)



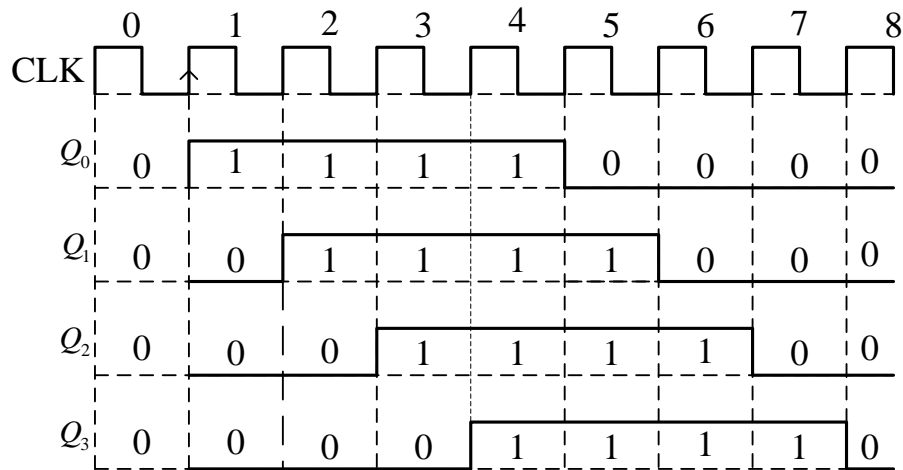
Hình 3.25 Mạch đếm vòng xoắn

Clock Pulse	Q3	Q2	Q1	Q0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	1	1	1
4	1	1	1	1
5	1	1	1	0
6	1	1	0	0
7	1	0	0	0

CLEAR	FF0	FF1	FF2	FF3
	0	0	0	0

Mạch đếm Johnson có một chút thay đổi so với đếm vòng ở chỗ đường ra đảo tầng cuối được đưa về đường vào tầng đầu. Hoạt động của mạch cũng giải thích tương tự. Với n tầng FF thì đếm vòng xoắn cho ra $2n$ số đếm do đó nó còn được coi là mạch đếm mod $2n$ (đếm nhị phân cho phép đếm với chu kỳ đếm đến 2^n). Như vậy ở trên là mạch đếm vòng xoắn 4 bit. Bảng bên cho thấy 8 trạng thái đường ra và hình dưới sẽ minh họa cho số đếm.

Ta có thể nạp trạng thái ban đầu cho mạch là 1000 bằng cách sử dụng đường Pr và Cl giống như ở trên. Dạng sóng các đường ra cũng giống như trên, hơn thế nữa, nó còn đối xứng giữa mức thấp với mức cao trong từng chu kỳ



Hình 3.26: Dạng sóng mạch đếm vòng xoắn

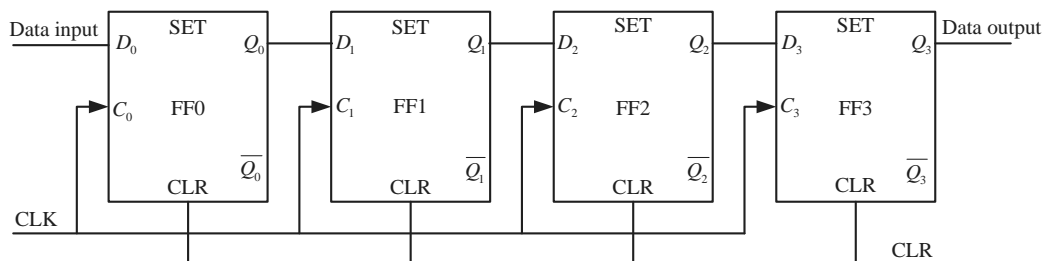
3.6. Bộ ghi dịch

3.6.1. Giới thiệu

Ở phần trước ta đã được biết đến các loại FF. Chúng đều có thể lưu trữ (nhớ 1 bit) và chỉ khi có xung đồng bộ thì bit đó mới truyền tới đường ra (đảo hay không đảo). Bây giờ nếu ta mắc nhiều FF nối tiếp lại với nhau thì sẽ nhớ được nhiều bit. Các đường ra sẽ phân hoạt động theo xung nhịp Clock đưa đến đầu vào. Có thể lấy đường ra ở từng tầng FF (gọi là các đường ra song song) hay ở tầng cuối (đường ra nối tiếp). Như vậy mạch có thể ghi lại dữ liệu (nhớ) và dịch chuyển nó (truyền) nên mạch được gọi là ghi dịch. Ghi dịch cũng có rất nhiều ứng dụng đặc biệt trong máy tính, như chính cái tên của nó: lưu trữ dữ liệu và dịch chuyển dữ liệu chỉ là ứng dụng nổi bật nhất.

3.6.2. Cấu tạo

Ghi dịch có thể được xây dựng từ các FF khác nhau và cách mắc cũng khác nhau nhưng thường dùng FF D, chúng được tích hợp sẵn trong 1 IC gồm nhiều FF (tạo nên ghi dịch n bit). Hãy xem cấu tạo của 1 ghi dịch cơ bản 4 bit dùng FF D



Hình 3.27: Ghi dịch 4 bit cơ bản

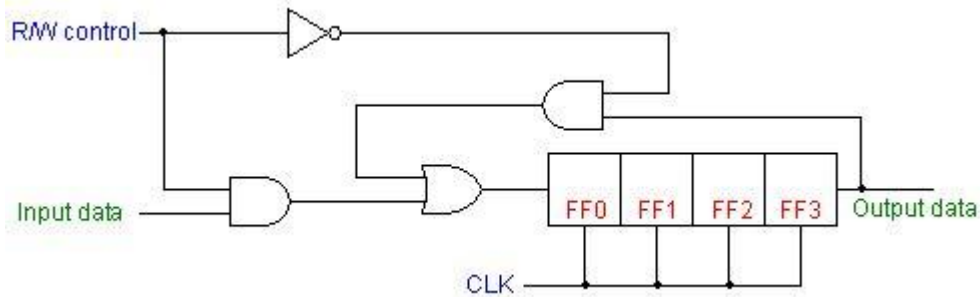
	FF0	FF1	FF2	FF3
CLEAR	0	0	0	0

3.6.2. Hoạt động

Thanh ghi, trước hết được xoá (áp xung CLEAR) để đặt các đường ra về 0. Dữ liệu cần dịch chuyển được đưa vào đường D của tầng FF đầu tiên (FF0). Ở mỗi xung kích lên của xung clock, sẽ có 1 bit được dịch chuyển từ trái sang phải, nối tiếp từ tầng này qua tầng khác và đưa ra ở đường Q của tầng sau cùng (FF3). Giả sử dữ liệu đưa vào là 1001, sau 4 xung clock thì ta lấy ra bit LSB, sau 7 xung clock ta lấy ra bit MSB.

	FF0	FF1	FF2	FF3	
0000	1	0	0	1	0000

Nếu tiếp tục có xung clock và không đưa thêm dữ liệu vào thì đường ra chỉ còn là 0 (các FF đã reset: đặt lại về 0 hết. Do đó ta phải ghim dữ liệu lại. Một cách làm là sử dụng 2 cổng AND, 1 cổng OR và 1 cổng NOT như hình dưới đây.



Hình 3.28: Cho phép chốt dữ liệu trước khi dịch ra ngoài

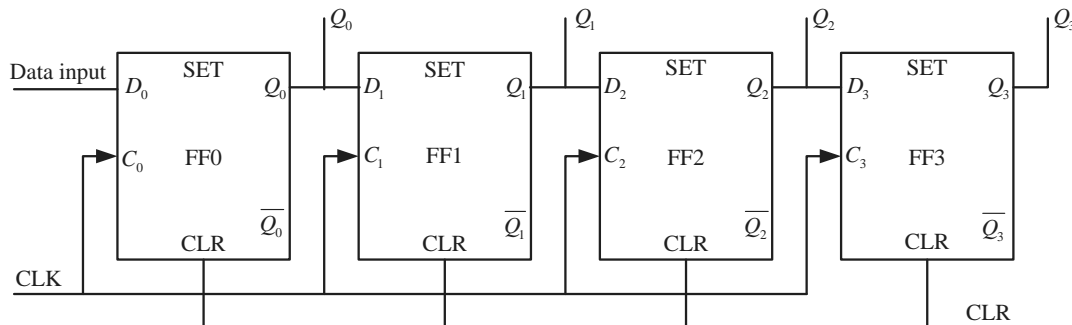
Dữ liệu được đưa vào thanh ghi khi đường điều khiển R/W control ở mức cao (Write). Dữ liệu chỉ được đưa ra ngoài khi đường điều khiển ở mức thấp (Read).

WRITE	FF0	FF1	FF2	FF3
1001	0	0	0	0

3.6.3. Một số bộ ghi dịch thông dụng

3.6.3.1. Bộ ghi dịch vào nối tiếp ra song song

Dữ liệu sẽ được lấy ra ở 4 đường Q của 4 tầng FF, vì chung nhịp clock nên dữ liệu được lấy ra cùng lúc.



Hình 3.29: Mạch ghi dịch vào nối tiếp ra song song

Bảng dưới đây cho thấy làm như thế nào dữ liệu được đưa tới đường ra 4 tầng FF

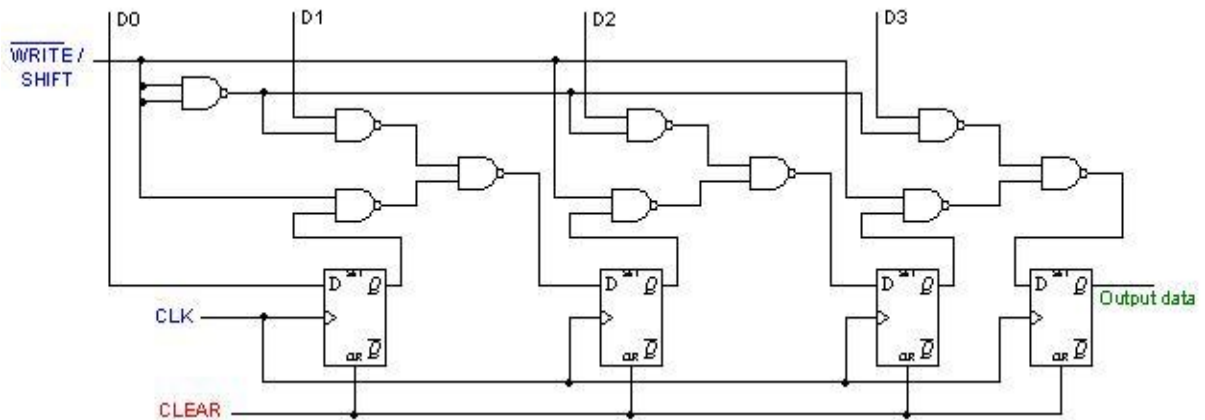
CLEAR	Q0	Q1	Q2	Q3
	0	0	0	0

3.6.3.2. Bộ ghi dịch vào song song ra nối tiếp

Bây giờ muốn đưa dữ liệu vào song song (còn gọi là nạp song song) ta có thể tận dụng đường vào không đồng bộ Pr và Cl của các FF để nạp dữ liệu cùng một lúc vào các F. Mạch hoạt động bình thường khi nạp song song ở thấp như đã nói. Khi nạp song song WRITE = 1 cho phép nạp

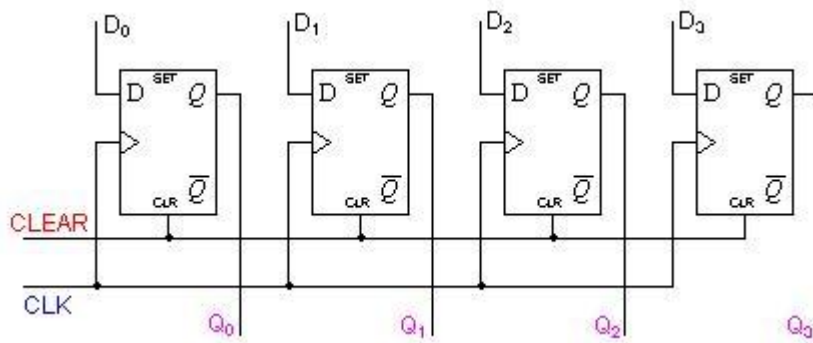
CLEAR	Q0	Q1	Q2	Q3
	0	0	0	0

ABCD được đưa vào Pr và Cl đặt và xoá để $Q_0 = A$, $Q_1 = B$, ... Xung ck và đường vào nối tiếp không có tác dụng (vì sử dụng đường không đồng bộ Pr và Cl) Một cách khác không sử dụng chân Pr và Cl được minh họa như hình dưới đây. Các cổng NAND được thêm vào để nạp các bit thấp D1, D2, D3. Đường WRITE/SHIFT dùng để cho phép nạp (ở mức thấp) và cho phép dịch (ở mức cao). Dữ liệu nạp và dịch vẫn được thực hiện đồng bộ như các mạch trước.



H3.2.4b Mạch ghi dịch nạp song song ra nối tiếp

Với mạch hình 3.2.4b đường ra dữ liệu là nối tiếp, ta cũng có thể lấy ra dữ liệu song song như ở hình 3.2.5, Cấu trúc mạch không khác so với ở trên. Dữ liệu được đưa vào cùng lúc và cũng lấy ra cùng lúc (mạch như là tầng đệm và hoạt động khi có xung ck tác động lên).

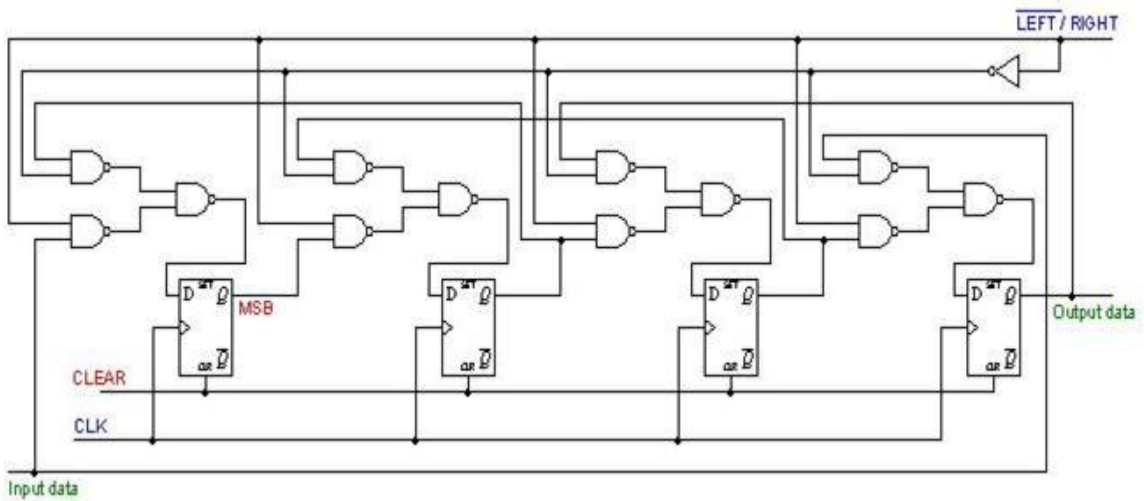


Hình 3.2.5 Mạch ghi dịch vào song song ra song song

Ghi dịch 2 chiều

Như đã thấy, các mạch ghi dịch nói ở những phần trên đều đưa dữ liệu ra bên phải nên chúng thuộc loại ghi dịch phải. Để có thể dịch chuyển dữ liệu ngược trở lại (dịch trái) ta chỉ việc cho dữ liệu vào đường D của tầng cuối cùng, đường ra Q được đưa tới tầng kế tiếp, Dữ liệu lấy ra ở tầng đầu.

Để dịch chuyển cả 2 chiều, có thể nối mạch như hình dưới đây:



Hình 3.2.6 Mạch ghi dịch cho phép dịch chuyển cả 2 chiều

Với mạch trên, các cổng NAND và đường cho phép dịch chuyển dữ liệu trái hay phải. Bảng dưới đây minh họa cho mạch trên: dữ liệu sẽ dịch phải 4 lần rồi dịch trái 4 lần. Để ý là thứ tự 4 bit ra bị đảo ngược lại so với chúng ở trên.



Hình 3.22: Sơ đồ bộ ghi dịch đầu vào nối tiếp Trigo JK nối kiểu Trigo D

- Khi lệnh ghi nhận trị "1" thông tin nhị phân $D_0 \div D_7$ đ- ọc ghi vào các trigo D ($F_0 \div F_7$), kết thúc lệnh ghi (nhận trị "0") thông tin nhị phân đ- ọc l- u trữ trong đó.

Khi có lệnh đ- ọc (G nhận trị "1") các cổng 3 trạng thái đ- ọc mở, thông tin nhị phân đ- ọc gửi tới địa chỉ cần nhận

Các thao tác ghi - đ- ọc đ- ọc thực hiện đồng thời với cả 8 bit thông tin.

Ngoài ra ng- ời ta còn kết hợp ph- ơng pháp nối tiếp và song song trong một bộ ghi dịch để sử dụng linh hoạt các - u thế của mỗi cách đồng thời tạo khả năng chuyển từ một dãy thông tin nối tiếp thành dạng song song hoặc ng- ợc lại. Hình 4 đ- a ra cấu trúc một bộ ghi dịch 4 bit kiểu này, sử dụng 4 trigo D kết hợp với các cổng logic phụ.