BỘ CÔNG THƯƠNG

TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG THÀNH PHỐ HỒ CHÍ MINH

**Khoa Công Nghệ Thông Tin**



**Digital Circuits**

**GVHD: Phùng Thế Bảo**

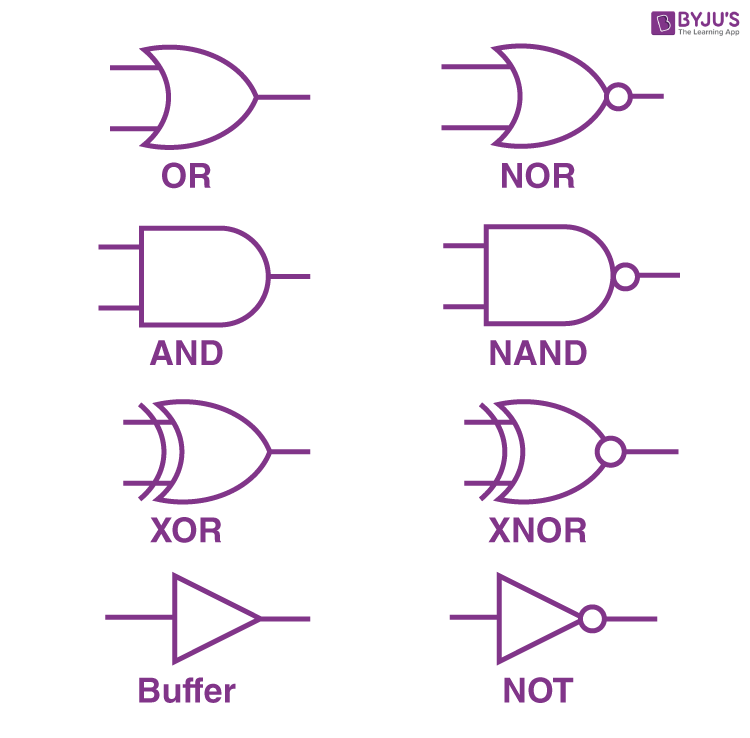
**Nhóm thực hiện: Nhóm 12**

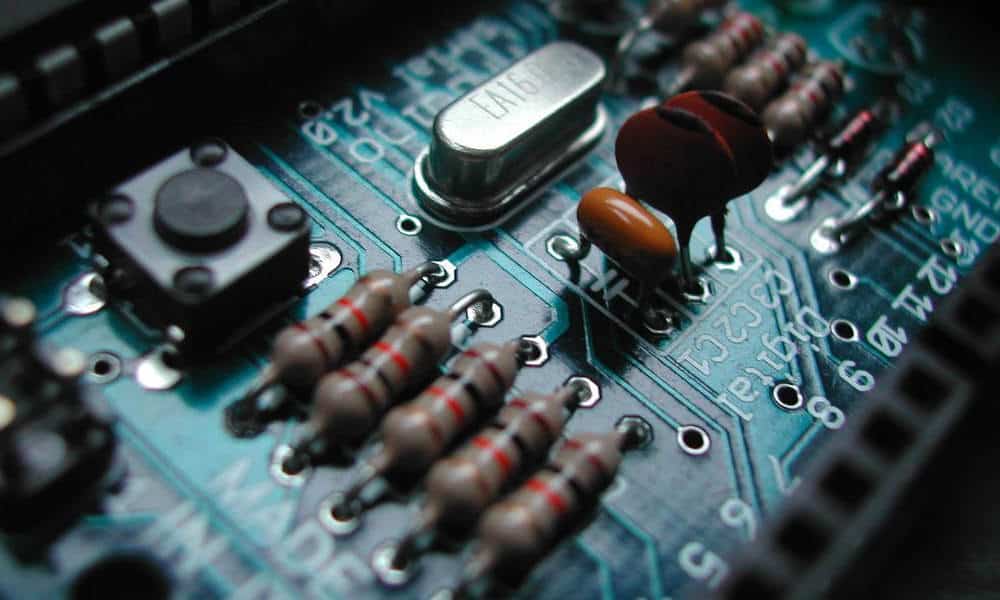
1. Lăng Minh Hải
2. Trần Trọng Quí
3. Nguyễn Minh Sang
4. Nguyễn Văn Thành
5. Nguyễn Hoành Thịnh

**Thành phố Hồ Chí Minh, tháng 10 năm 2023**

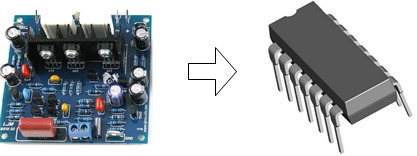
1. Giới thiệu về Digital Circuits
   1. Digital Circuits là gì?

* Digital circuits là các mạch điện tử sử dụng các linh kiện bán dẫn để xử lý hoặc lưu trữ các tín hiệu kỹ thuật số.
* Các tín hiệu kỹ thuật số (digital) chỉ có hai giá trị: 0 hoặc 1, tương ứng với hai trạng thái logic: sai hoặc đúng.
* Các digital circuits có thể được thiết kế để thực hiện các chức năng logic cơ bản như AND, OR, NOT,… hay các chức năng phức tạp hơn như cộng, trừ, nhân, chia, mã hóa, giải mã, v.v.
* Các digital circuits là cơ sở của các thiết bị điện tử hiện đại như máy tính, điện thoại thông minh, máy chơi game, đồng hồ thông minh, v.v.

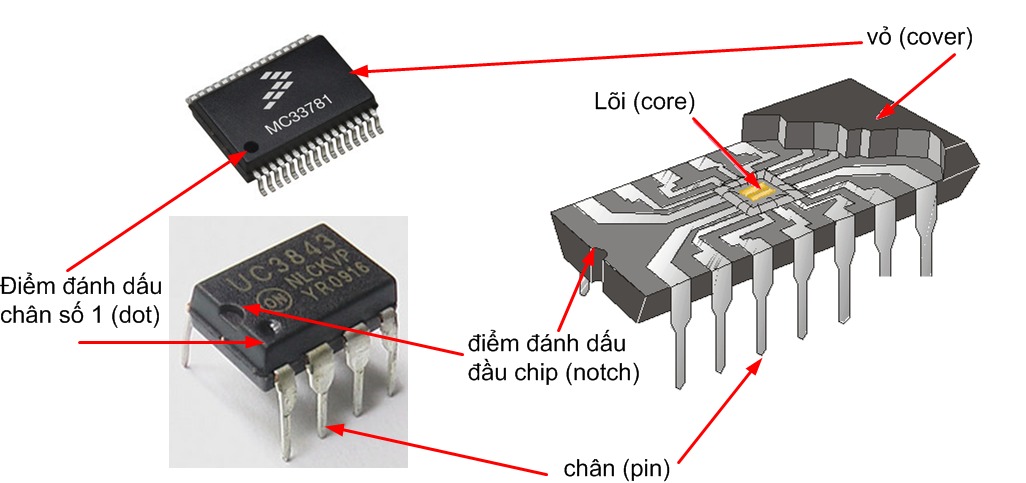




* 1. Vi mạch là gì?

Mạch tích hợp (Integrated circuit), viết tắt là IC, còn được gọi là vi mạch hoặc chip là những linh kiện điện tử có lõi làm bằng vật liệu bán dẫn. Vật liệu bán dẫn có rất nhiều loại nhưng phổ biến là Silicon (Si) và Germanium (Ge). Hiểu đơn giản, vi mạch là mạch điện tử rất nhỏ được đóng gói thành một linh kiện hoàn chỉnh.

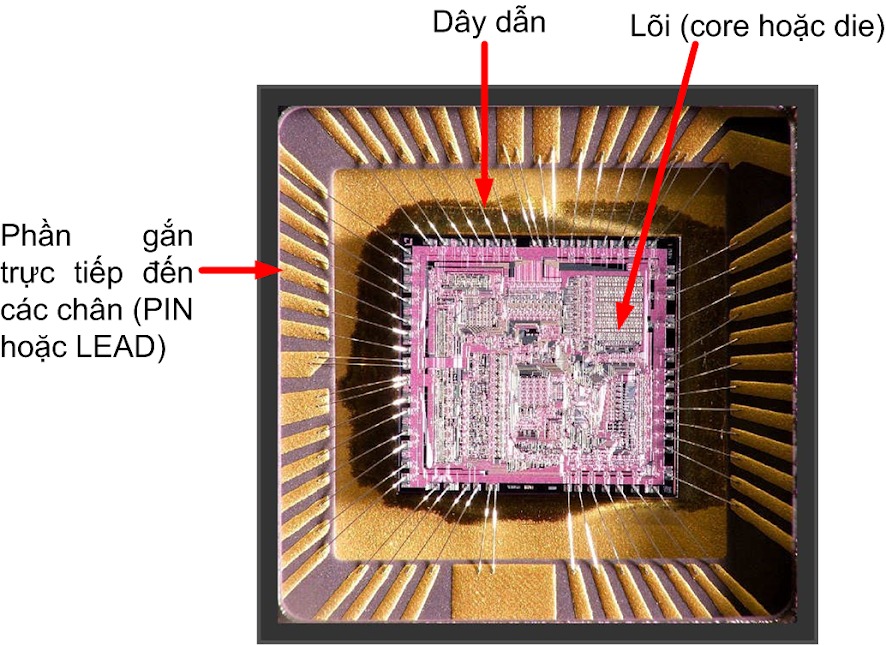
Cấu tạo cơ bản của một vi mạch (chip) gồm có vỏ chip (cover), lõi (core) và chân (pin).



Lõi (CORE hoặc DIE) vi mạch là mạch điện gồm các thành phần thụ động hoặc tích cực hoặc cả hai. Các thành phần thụ động gồm có điện trở, tụ điện, cuộn cảm. Các thành phần tích cực gồm có diodes, transistor, NMOS, PMOS,... Cụm từ "thiết kế vi mạch" được ngầm hiểu là thiết kế phần lõi vi mạch thông qua các công cụ phần mềm chuyên dụng. Việc chế tạo phần lõi và sản xuất ra chip thuộc về nhà máy chế tạo chip.

Chân chip (PIN hoặc LEAD) là vật liệu dẫn điện được kết nối đến lõi thông qua các dây dẫn điện, gọi là BOND WIRE. Các dây này thường làm bằng vàng.

Vỏ là thành phần bao bọc và cố định lõi, dây dẫn và các chân, được làm bằng vật liệu cách điện như nhựa, gốm, thủy tinh,...



* 1. Phân loại vi mạch

Vi mạch có thể được phân làm nhiều loại khác nhau dựa trên các đặc điểm về cấu trúc, chức năng hay mật độ tích hợp.

Một cách phân loại phổ biến dựa trên mật độ tích hợp là dựa vào số lượng transistor hoặc CMOS hoặc số cổng (gate) có trong vi mạch để chia thành các nhóm như sau:

Vi mạch mật độ tích hợp thấp SSI (Small Scale Integration) là những vi mạch chứa khoảng từ vài đến vài chục transistor.

Ví dụ như các chip 74151 (Multiplexer), 7404 (cổng đảo), ...

Vi mạch mật độ tích hợp trung bình MSI (Medium Scale Integration) là những vi mạch chứa khoảng vài trăm transistor

Ví dụ như các chip 54157 (multiplexer) chứa khoảng 100 transistor

Vi mạch mật độ tích hợp cao LSI (Large Scale Integration) là những vi mạch chứa khoảng vài nghìn transistor

Ví dụ như chip Intel 4004 chưa khoảng 2300 transistor, Intel 8008 khoảng 3500 transistor

Vi mạch mật độ tích hợp rất cao VLSI (Very Large Scale Integration) là những vi mạch chứa khoảng vài chục nghìn transistor trở lên

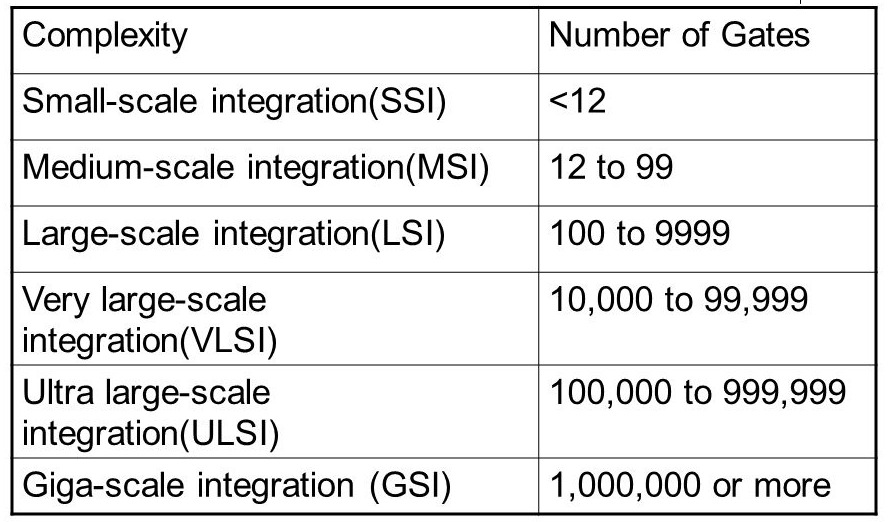
Ví dụ như chip 8088 chưa khoảng 29000 transistor, Motorola 68000 chứa khoảng 68000 transistor, ARM 1 chứa khoảng 25000 transistor

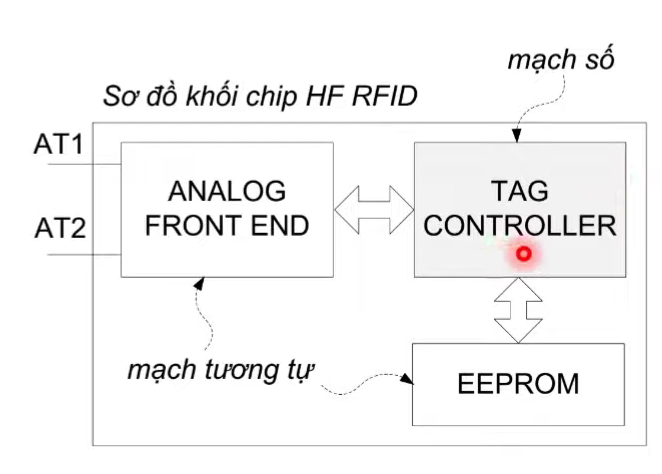
Vi mạch mật độ tích siêu cao ULSI (Ultra Large Scale Integration) là những vi mạch chứa khoảng vài triệu transistor.

Ví dụ như chip Pentium I, II và III đời đầu khoảng vài triệu transistor, ...

Vi mạch mật độ tích hợp cực cao GSI (Giant Scale Integration) là những vi mạch chứa khoảng vài trăm triệu transistor hoặc hơn.

Ví dụ các dòng Dual-core của Intel, AND K8/K10, ... dùng cho máy tính cá nhân PC hoặc các dòng đa nhân dùng cho các điện thoại thông minh như A7 của Apple có số lượng transistor khoảng 1 tỷ với 2 nhân ARM64, ...





* 1. Sơ lược lịch sử vi mạch

William B. Shockley và nhóm nghiên cứu của ông tại Bell Laboratories của Tổng công ty American Telephone and Telegraph đã đóng góp vào sự ra đời của mạch tích hợp.

Đội ngũ của Shockley phát hiện ra rằng, trong các điều kiện đúng đắn, các electron sẽ tạo thành một rào cản tại bề mặt của một số loại tinh thể cụ thể, và họ học cách kiểm soát luồng điện chảy qua tinh thể bằng cách thay đổi rào cản này.

Việc kiểm soát luồng electron qua tinh thể cho phép đội ngũ tạo ra một thiết bị có khả năng thực hiện một số hoạt động điện tử như khuếch đại tín hiệu, công việc trước đây thường được thực hiện bằng bóng chân không.

Họ đặt tên cho thiết bị này là "transistor," từ sự kết hợp của từ "transfer" (chuyển đổi) và "resistor" (điện trở). Nghiên cứu về cách tạo ra các thiết bị điện tử bằng vật liệu rắn trở nên nổi tiếng với cái tên "bóng bán dẫn." Các bóng bán dẫn đã chứng minh rằng chúng rất cứng cáp, dễ làm việc, đáng tin cậy hơn, nhỏ gọn hơn và giá rẻ hơn so với bóng chân không.

Sử dụng các nguyên tắc và vật liệu tương tự, các kỹ sư sớm học cách tạo ra các thành phần điện tử khác như điện trở và tụ điện. Bây giờ khi các thiết bị điện tử có thể được làm rất nhỏ, phần lớn nhất của mạch là dây nối không thoải mái giữa các thiết bị. Đối với các vi mạch tích hợp, từ viết tắt IC - viết tắt của cụm từ "Integrated Circuit" - thường được sử dụng. Cũng được gọi là một vi mạch, hoặc một vi mạch nhỏ.

1. Cổng Logic và Đại số Boolean
   1. **Đại số Boolean**

**Giới thiệu khái quát:**

- Đại số Boolean hay đại số Boole được phát minh bởi nhà toán học Anh George Boole(1815-1864).

- Đại số Boolean là một cấu trúc đại số thao tác với các biến luận lý nhị phân (biến luận lý).

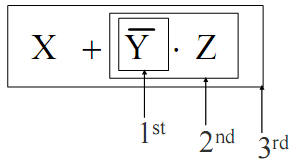
- Biến luận lý chỉ mang giá trị 1 (TRUE) hoặc 0 (FALSE).

**Các phép toán logic cơ bản trong đại số Boolean:**

* Phép cộng: **(+)** hay **OR.**
* Phép nhân: **(.)** hay **AND.**
* Phép bù: **NOT.**

**Độ ưu tiên của các phép toán**

* + Biểu thức được tính từ **trái** sang **phải**.
  + Biểu thức trong **ngoặc đơn** được đánh giá trước.
  + Thứ tự ưu tiên của các phép toán : **NOT > AND > OR**

 **Ví dụ**: Y = A + B.

**Bảng chân trị**

|  |  |  |
| --- | --- | --- |
| A | B | A AND B |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

|  |  |
| --- | --- |
| A | (NOT A) |
| 0 | 1 |
| 1 | 0 |

|  |  |  |
| --- | --- | --- |
| A | B | A OR B |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

**Các tiên đề của đại số boolean**

**Tiên đề 1**: A = 0 khi và chỉ khi A không bằng 1

A = 1 khi và chỉ khi A không bằng 0

**Tiên đề 2**: Phần tử đồng nhất

x + 0 = x

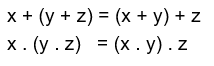
x . 1 = x

**Tiên đề 3**: Tính giao hoán

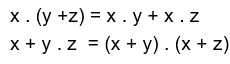
x + y = y + x

x . y = y . x

**Tiên đề 4**: Tính kết hợp



**Tiên đề 5**: Tính phân phối



**Tiên đề 6**: Tính bù



**Tính đối ngẫu**

Đại số Boolean mang tính đối ngẫu .

Đổi **AND** thành **OR** và **OR** thành **AND.**

Đổi **0** thành **1** và **1** thành **0.**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Cột 1** | **Cột 2** | **Cột 3** |
| **Dòng 1** | **1 + 1 = 1** | **1 + 0 = 0 + 1 = 1** | **0 + 0 = 0** |
| **Dòng 2** | **0 . 0 = 0** | **0 . 1 = 1 . 0 = 0** | **1 . 1 = 1** |

**Ví dụ :**

x . (y + z) = x . y + x . z **đối ngẫu** x + yz = (x + y)(x + z)

**Các định lý của đại số boolean**

**Định lí 1**: Luật lũy đẳng



**Định lí 2** : Luật nuốt



**Định lí 3** : Luật hấp thu



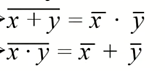
**Định lí 4 :** Luật bù kép



**Định lí 5 :**



**Định lí 6** : De Morgan



**Hàm Boolean**

- Kết hợp các biến, hằng số, toán tử, dấu ngoặc tạo thành 1 **biểu thức boolean.**

**Ví dụ**: x + yz

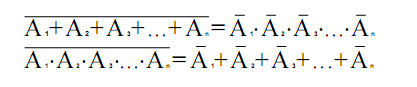
- Kết hợp theo thứ tự 1 tên hàm, 1 dấu bằng và cuối cùng là 1 biểu thức boolean sẽ cho chúng ta được một **hàm boolean** ( hàm boolean dạng chuẩn). Giá trị của hàm Boolean có thể là 0 hoặc 1.

Ví dụ: f(x,y,z) = x + yz

**Phần bù của hàm Boolean**

**Có 2 cách để tìm phần bù của hàm Boolean:**

+ Sử dụng định lí De Morgan.

****

**Ví dụ:**

F = x . y +

= = .

= ( . ( x + y + )

+ Lấy đối ngẫu và lấy bù các biến.

Đổi AND thành OR và OR thành AND

Đổi 0 thành 1 và 1 thành 0

**Ví dụ**:

**F = x . y +**

Lấy đối ngẫu : (x + y) . (

Bù các biến **= ( . ( x + y + )**

* 1. Biểu diễn hàm boole

**Bảng trạng thái**

- Bảng trạng thái gồm các cột, liệt kê giá trị (trạng thái) mỗi biến theo từng cột và giá trị hàm theo một cột riêng (thường là bên phải bảng).

- Bảng trạng thái còn được gọi là bảng sự thật hay bảng chân trị.

- Số hàng của bảng là , n là số các biến nhị phân được sử dụng trong hàm

**Ví dụ** bảng trạng thái của hàm f(x,y,z) = x + yz

|  |  |  |  |
| --- | --- | --- | --- |
| **x** | **y** | **z** | **f** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

- Rõ ràng, trực quan. Sau khi xác định các giá trị biến vào có thể tìm được giá trị  
đầu ra nhờ bảng trạng thái. Để giải quyết bài toán ở dạng logic thì sử dụng bảng trạng thái là hữu ích nhất. Do vậy, trong quá trình thiết kế mạch số việc đầu tiên nên làm là lập bảng trạng thái.  
- **Nhược điểm** chủ yếu của bảng trạng thái là sẽ phức tạp nếu số biến quá nhiều, không thể dùng các công thức và định lý để tính toán.

**Bài tập : Lập bảng chân trị từ biểu thức**

F = C +

|  |  |  |
| --- | --- | --- |
| B | C | F |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

F = (A + B)(AC + A) + A(B + BC)

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | C | F |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

**Dạng chính tắc của hàm Boolean**

Hạng tích (**minterm**): ký hiệu mi , với i = 0 đến 2n -1, là các tổ hợp gồm **tích** các biến, trong đó

* Giá trị ‘1’ được biểu diễn bằng nguyên biến (biến trực tiếp)
* Giá trị ‘0’ được biểu diễn bằng đảo biến (biến phủ định)

Hạng tổng (**Maxterm**): ký hiệu Mi , với i = 0 đến 2n -1, là các tổ hợp gồm **tổng** các biến, trong đó:

* Giá trị ‘0’ được biểu diễn bằng nguyên biến (biến trực tiếp)
* Giá trị ‘1’ được biểu diễn bằng đảo biến (biến phủ định)

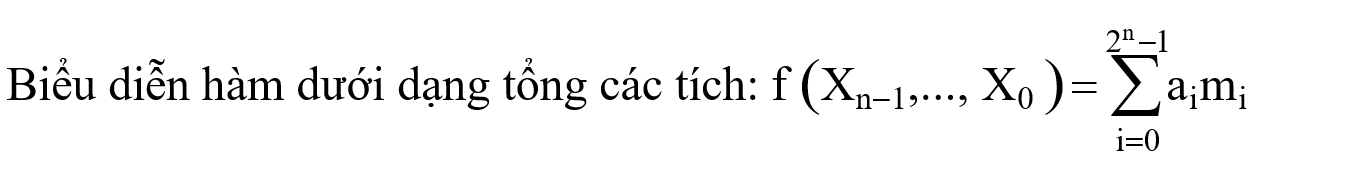
Mối quan hệ giữa minterm và Maxterm:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Biến** | | | **minterm** | | **Maxterm** | |
| **x** | **y** | **z** | **Biểu thức** | **Kí hiệu** | **Biểu thức** | **Kí hiệu** |
| 0 | 0 | 0 |  | m0 | x + y + z |  |
| 0 | 0 | 1 | z |  | x + y+ |  |
| 0 | 1 | 0 |  |  | x + + z |  |
| 0 | 1 | 1 |  |  | x + |  |
| 1 | 0 | 0 |  |  |  |  |
| 1 | 0 | 1 |  |  |  |  |
| 1 | 1 | 0 |  |  |  |  |
| 1 | 1 | 1 | x y z |  |  |  |

**Một biểu thức n biến luôn có thể biểu diện dưới 2 dạng**

+ Dạng tuyển (tổng các tích – **SOP(sum of product)):** Mỗi số hạng là một *hạng tích* hay *minterm* (mi). Mà tại đó tổ hợp hàm Boolean có giá trị là 1.

+ Dạng hội (tích các tổng – **POS(product of sum)):** Mỗi thừa số là một *hạng tổng* hay *Maxterm* (Mi). Mà tại đó tổ hợp hàm Boolean có giá trị là 0.

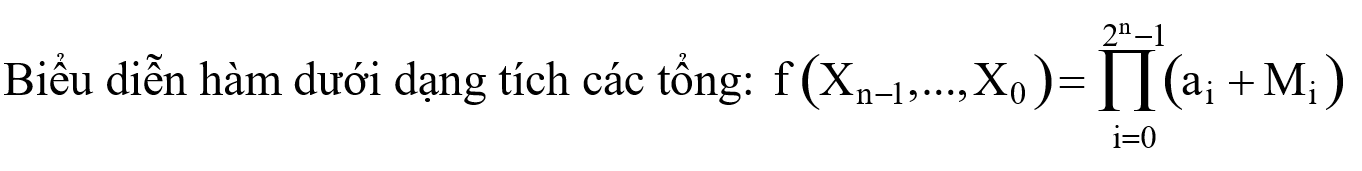


**Các bước để biểu diễn hàm Bool theo dạng tổng của các tích:**

1. Xây dựng một bảng chân trị cho hàm Boolean.

2. Hình thành một minterm cho mỗi sự kết hợp của các biến tạo ra hàm có giá trị là 1.

3. Biểu thức cuối cùng là cộng tất cả các minterm thu được từ bước 2.



**Các bước để biểu diễn hàm Bool theo dạng tích của các tổng**

1. Xây dựng một bảng chân trị cho hàm Boolean.

2. Hình thành một maxterm cho mỗi sự kết hợp của các biến với các biến này thì hàm này có giá trị là 0.

3. Biểu thức cuối cùng là nhân tất cả các maxterm thu được từ bước 2.

**Ví dụ**: Cho bảng chân trị hàm f

|  |  |  |  |
| --- | --- | --- | --- |
| x | y | z | f |
| 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 | 1 |

**Biểu diễn dưới dạng tổng các tích - SOP**

Các **minterm** tương ứng là:

x x.y. x.y.z

Sau đó lấy tổng(OR) của tất cả các **minterm** ta được biểu thức hàm f

f(x,y,z) = + + x. + x.y. + x.y.z

= 001 + 011 + 100 + 110 + 111

=

=

**Biểu diễn dưới dạng tích các tổng - POS**

f(x,y,z) = (x + y + z)(x + (

Các **Maxterm** tương ứng là

(x + y + z) (x + (

Sau đó, lấy tích (AND) của tất cả các maxterm này, ta được biểu thức hàm f

f(x,y,z) = (x + y + z)(x + (

= 000 . 010 . 101

=

=

**Bài tập**: Tính biểu thức hàm f (x,y,z) = x.y + z dưới dạng tổng các tích

*Chuẩn hoá hàm về dạng chuẩn tổng các tích (chuẩn tắc tuyển):*

* Thêm các biến còn thiếu vào các hạng tích mà không làm ảnh hưởng đến kết quả bằng cách nhân hạng tích đó với ‘1’ (tổng của nguyên biến và đảo biến còn thiếu).
* Loại bỏ các hạng tích lặp lại (hạng tích thừa).

f(x,y,z) = x.y + z

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

= x.y. + x.y.z + z + y.z + x.z + x.y.z

**Ghi chú** : Loại bỏ 1 xyz đi vì theo luật lũy đẳng x + x = x

= x.y + x.y.z + z + .y.z + x.z

= 110 + 111 + 001 + 011 + 101

= +   
 =

**Ví dụ**: Tính biểu thức f(x,y,z) = (x + y)(x + ) dưới dạng tích các tổng

*\* Chuẩn hoá hàm về dạng chuẩn tích các tổng (chuẩn tắc hội):*

* Thêm các biến còn thiếu vào các hạng tổng mà không làm ảnh hưởng đến kết quả bằng cách cộng hạng tổng đó với ‘0’ (tích của nguyên biến và đảo biến còn thiếu).
* Loại bỏ các hạng tổng lặp lại (hạng tổng thừa).

f(x,y,z) = (x + y)(x + )

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

= (x + y + z)(x + y + )(x + y + )(x + + )

**Ghi chú:** Loại bỏ 1 x + y + z’ vì theo luật lũy đẳng x.x=x

= (x + y + z)(x + y + )(x + + )

= 000 . 001 . 011

=

* 1. Tối thiểu hóa hàm Boole

- Là đưa hàm Boolean về dạng biểu diễn đơn giản nhất sao cho biếu thức có chứa ít nhất các thừa số và mỗi thừa số chứa ít nhất các biến, mạch logic thực hiện có chứa ít nhất các vi mạch số.

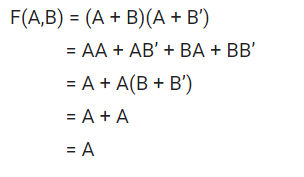
- Phương pháp sử dụng phương pháp đại số. Áp dụng các định lí tiên đề các luật để tối thiểu hàm boolean đến mức thấp nhất.

**Ví dụ**:

f(x,y,z) = x + y + xy

= x(1 + y) + y

= x + y



***Chú*** *ý: nếu trong tổng các tích, xuất hiện một biến và đảo của biến đó trong hai số hạng khác nhau, các thừa số còn lại trong hai số hạng đó tạo thành thừa số của một số hạng thứ ba thì số hạng thứ ba đó là thừa và có thể bỏ đi.*

F = C +

= C + (áp dụng đli de morgan)

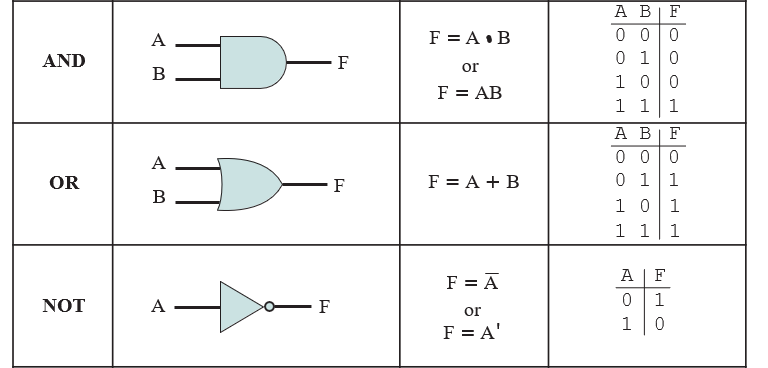
= = 1

* 1. Cổng logic

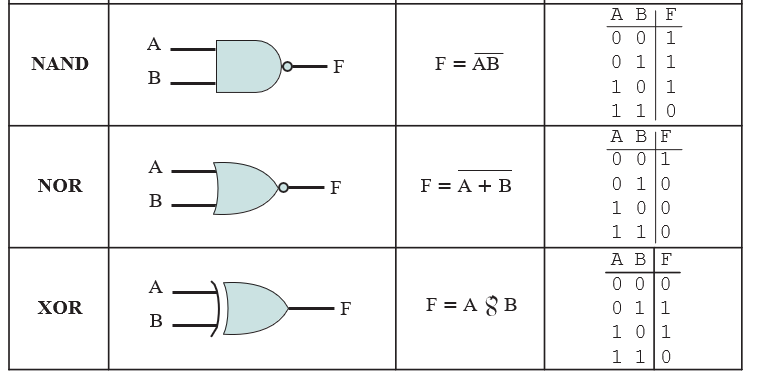
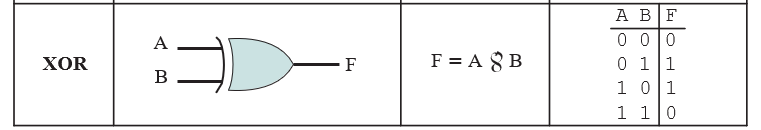
- Là những thiết bị điên tử rất nhỏ bé, có ít nhất 1 ngõ vào nhưng chỉ có duy nhất 1 ngõ ra, các giá trị vào hoặc ra chỉ có thể nhận một trong hai giá trị là 1 hoặc 0.

- Các **transistor** được ghép nối lại để tạo thành các cổng logic có thể thực hiện các phép toán cơ bản của đại số Boolean. Các cổng cơ bản này lại được lắp ghép thành các phần tử chức năng lớn hơn như mạch cộng 1 bit, nhớ 1 bit,… từ đó tạo thành 1 máy tính hoàn chỉnh.

**Các cổng logic cơ bản**

****

**Một số cổng ghép thông dụng như**

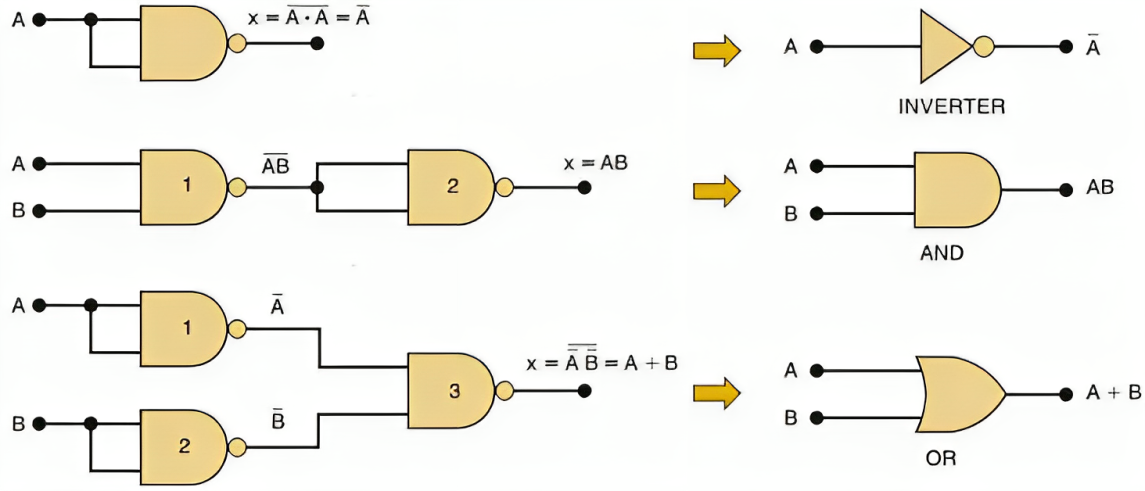
****

****

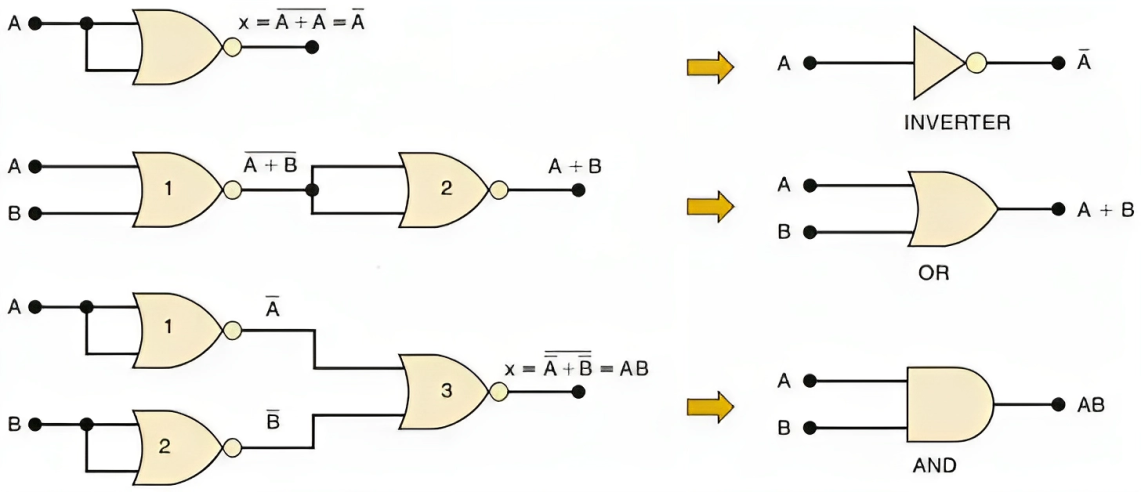


**Sự đa năng của cổng NAND và cổng NOR**

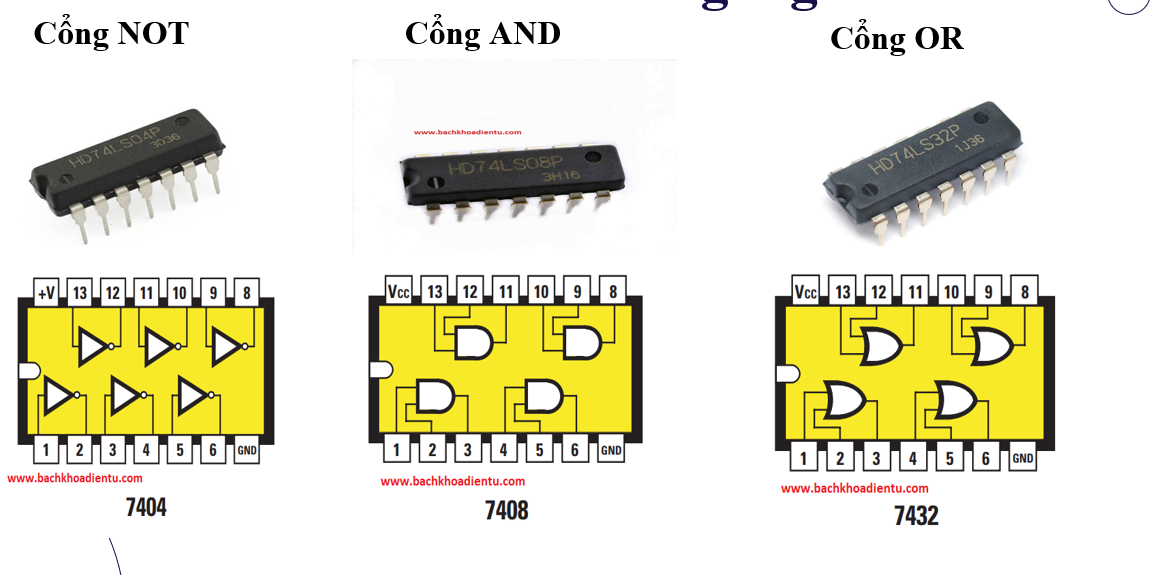
Từ cổng NAND có thể tạo ra các cổng NOT,AND,OR

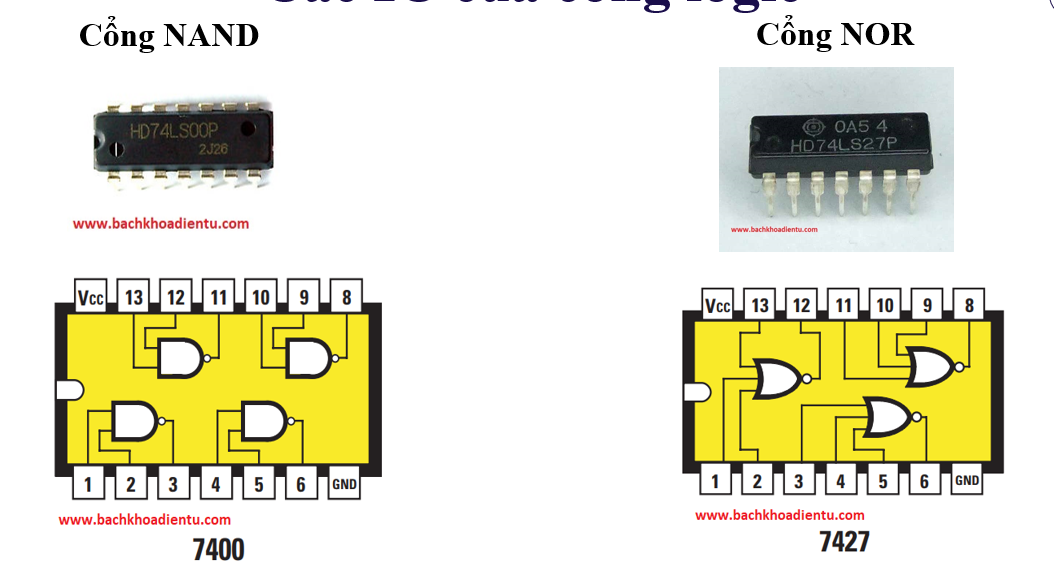


Từ cổng **NOR** có thể tạo ra các cổng NOT,AND,OR



**Các IC cổng logic**





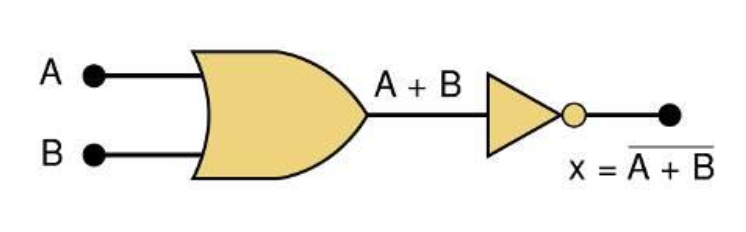
**Bài tập:**

Xây dựng 1 mạch logic từ biểu thức

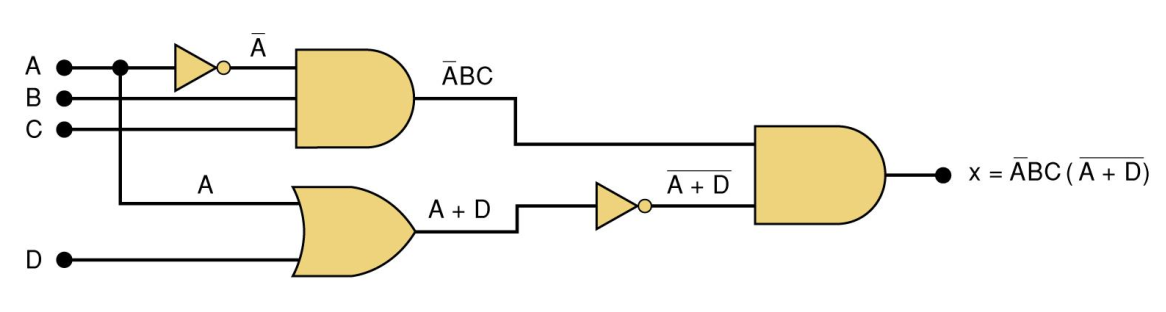
F = + B

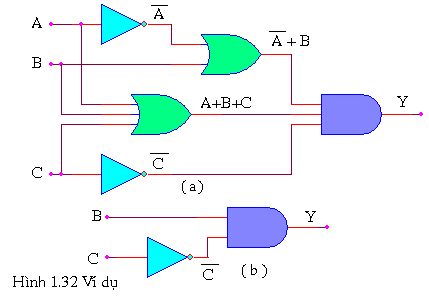


F =

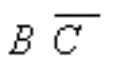


**Tìm biểu thức logic của mạch logic**

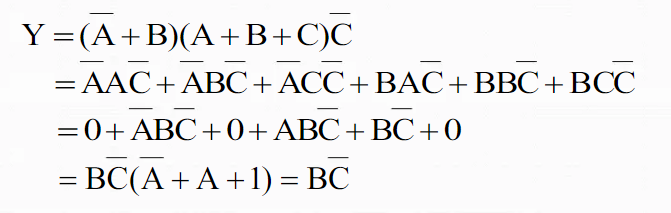


**Đơn giản mạch**





**+ Rút gọn biểu thức**



Sau khi rút gọn được biểu thức ta sẽ thu được mạch logic mới ở hình b

* 1. Bản đồ Karnaugh

Bản đồ Karnaugh (K-map) là một công cụ quan trọng trong lĩnh vực thiết kế mạch điện tử và lý thuyết hệ thống logic số.

Được đặt tên theo nhà toán học người Mỹ Maurice Karnaugh.

Bản đồ Karnaugh có vai trò quan trọng trong việc thiết kế và tối ưu hóa hệ thống logic số vì nó giúp biểu diễn và quản lý các hàm logic Boolean một cách trực quan và hiệu quả.

Một số lý do K-map quan trọng trong thiết kế mạch điện tử:

* *Biểu diễn trực quan:* cho phép biểu diễn hàm logic Boolean bằng biểu đồ 2D với ô vuông đại diện cho các kết hợp giá trị của biến đầu vào.
* *Tối ưu hóa hàm logic:* cho phép dễ dàng tối ưu hóa hàm logic bằng cách nhóm các ô có giá trị 1 lại với nhau. Quá trình này giúp giảm bớt số lượng cổng logic cần thiết để thực hiện hàm logic, từ đó giảm thiểu diện tích và tiêu thụ năng lượng của mạch.
* *Dễ thực hiện:* có thể dễ dàng chuyển đổi sang các mạch cơ bản như cổng AND, OR, NOT, và XOR. Điều này giúp đơn giản hóa quá trình thiết kế và triển khai mạch điện tử.
* *Giảm lỗi và xác thực:* là công cụ hữu ích trong việc kiểm tra tính đúng đắn của hệ thống logic và giảm thiểu sai sót trong quá trình thiết kế.
* *Tiết kiệm thời gian và tài nguyên:* giúp tối ưu hóa hàm logic nhanh chóng và hiệu quả hơn, tiết kiệm thời gian và tài nguyên so với việc tối ưu hóa bằng cách thủ công hoặc sử dụng phần mềm.

Cách biểu diễn hàm logic Boolean trên K-map

Bước 1: Xác định số biến đầu vào và tạo bản đồ Karnaugh phù hợp

* Một hàm logic với n biến sẽ yêu cầu một bản đồ Karnaugh kích thước 2n.

Bước 2: Tạo bản đồ Karnaugh và đánh số các ô

* Vẽ Bản đồ Karnaugh với các ô tương ứng với các kết hợp giá trị của biến đầu vào. Đánh số các ô theo thứ tự từ trái sang phải và từ trên xuống dưới (bằng mã Gray).
* Số lượng bit sẽ tăng theo số biến vd 3 biến là 8 bit, 4 biến là 16 , 5 biến là 32 , 6 biến là 64 Vì vậy Kmap chỉ giải quyết những bài từ 2 tới 6 biến vì lớn hơn thì Kmap rất là lớn nên những bài lớn nên sử dụng đại số bool để thực hiện

Bước 3: Điền giá trị hàm logic vào các ô tương ứng

* Dựa vào hàm logic Boolean, điền giá trị hàm logic vào từng ô trên Bản đồ Karnaugh. Sử dụng 1 để biểu diễn giá trị đúng (TRUE) và 0 để biểu diễn giá trị sai (FALSE).

Bước 4: Nhóm các ô có giá trị 1 lại với nhau

* Bắt đầu nhóm các ô chứa giá trị 1 lại với nhau. Điều này có thể được thực hiện bằng cách nối chúng theo chiều ngang hoặc chiều dọc, tránh nối theo đường chéo. Mỗi nhóm sẽ đại diện cho một sản phẩm trong hàm logic cuối cùng.

Bước 5: Viết hàm logic sau khi tối ưu hóa

* Sau khi đã nhóm, viết lại hàm logic tối ưu hóa bằng cách kết hợp các biến và các nhóm ô đã được xác định. Mỗi nhóm ô sẽ tương ứng với một mệnh đề logic trong hàm cuối cùng.
* ***Ví dụ:***

🡺 3 biến A, B, C 23 = 8

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AB  C | 00 | 01 | 11 | 10 |
| 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | C | Y |
| 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 |

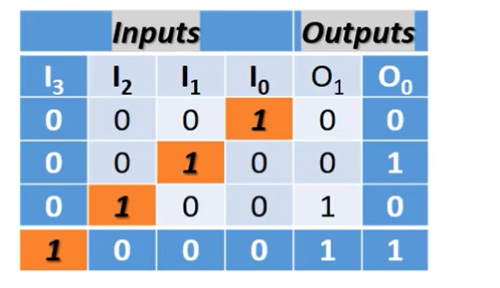
* Y = AB + C

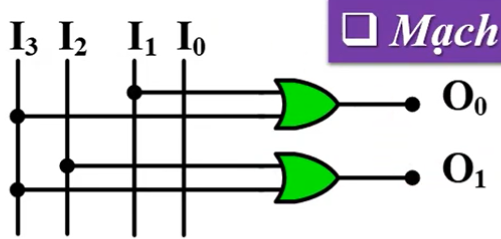
Ứng dụng của bản đồ Karnaugh

* Thiết kế mạch điện tử
  + Tối ưu hóa hàm logic
  + Kiểm tra tính đúng đắn của hệ thống logic
* Một số lĩnh vực cụ thể:
  + Mạch điều khiển (thiết kế mạch điều khiển)
    - Là một công cụ quan trọng trong việc thiết kế mạch điện tử, bao gồm các mạch tổng hợp, bộ xử lý tín hiệu số, và các mạch điều khiển. Nó giúp xác định cách kết nối các cổng logic để thực hiện chức năng mong muốn và đảm bảo tính đúng đắn của mạch.
  + Thiết bị viễn thông số
    - K-map được sử dụng để thiết kế các thiết bị viễn thông như modem, router, thiết bị thu phát sóng, và các thiết bị khác liên quan đến viễn thông và truyền thông
  + Truyền thông không dây
    - Tối ưu hóa việc mã hóa và giải mã tín hiệu, quyết định định tuyến, quản lý tài nguyên truyền thông, và cải thiện hiệu suất mạng không dây.
* Ngoài ra còn một số ứng dụng như là mạch đồng bộ hoá và không đồng bộ hoá, giáo dục nghiên cứu, phát triển phần mềm logic,…

1. Mạch logic cơ bản
   1. Mạch giải mã – mã hoá
      1. Mã hoá và giải mã trong mạch điện tử số là gì?

* Là quá trình biến đổi và trích xuất thông tin từ dạng tín hiệu số một cách hiệu quả
* Mã hoá
  + Là quá trình biến đổi thông tin từ dạng ban đầu thành một dạng tín hiệu số. Mục tiêu của mã hoá có thể là giảm kích thước của dữ liệu, bảo mật thông tin, hoặc chuẩn hóa tín hiệu để dễ dàng truyền tải qua các phương tiện số.
* Giải mã
  + Là quá trình trích xuất thông tin từ tín hiệu số đã được mã hoá để khôi phục thông tin ban đầu. Giải mã là bước quan trọng để hiểu được thông tin và sử dụng nó. Các mạch giải mã được thiết kế để thực hiện quá trình ngược lại so với mã hoá.
* Một số loại mã hoá và giải mã phổ biến bao gồm mã hóa - giải mã mã nguồn (ví dụ: mã Morse cho văn bản), mã hóa kênh (ví dụ: mã Manchester cho truyền tín hiệu qua cáp mạng Ethernet), và nhiều loại mã hóa - giải mã hình ảnh và âm thanh.
  + 1. Mạch mã hoá
* Gồm m ngõ vào và n ngõ ra
* Khi có 1 đường ở ngõ vào ở trạng thái tích cực mức (1) thì sẽ tạo ra 1 số nhị phân tương ứng ở các ngõ ra
* Mối quan hệ giữa m và n : m <= 2n
* m = 3, 4 thì n = 2
* m = 5, 6, 7, 8 thì n = 3
* m = 9, 10, 11, 12, 13, 14, 15, 16 thì n = 4



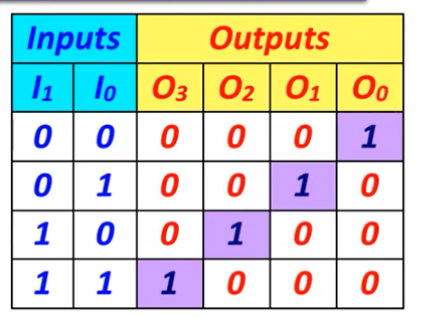
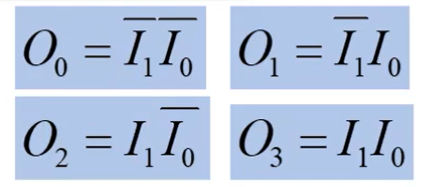
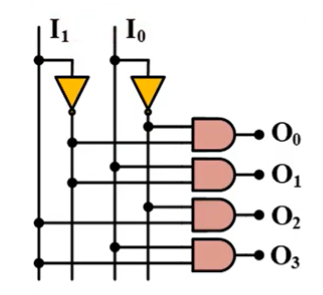


* + 1. Mạch giải mã
* Gồm n ngõ vào và m ngõ ra
* Mối quan hệ giữa m và n : m = 2n

Mạch giải mã 2 đường sang 4 đường

Mạch giải mã 3 đường sang 8 đường

Mạch giải mã 4 đường sang 16 đường

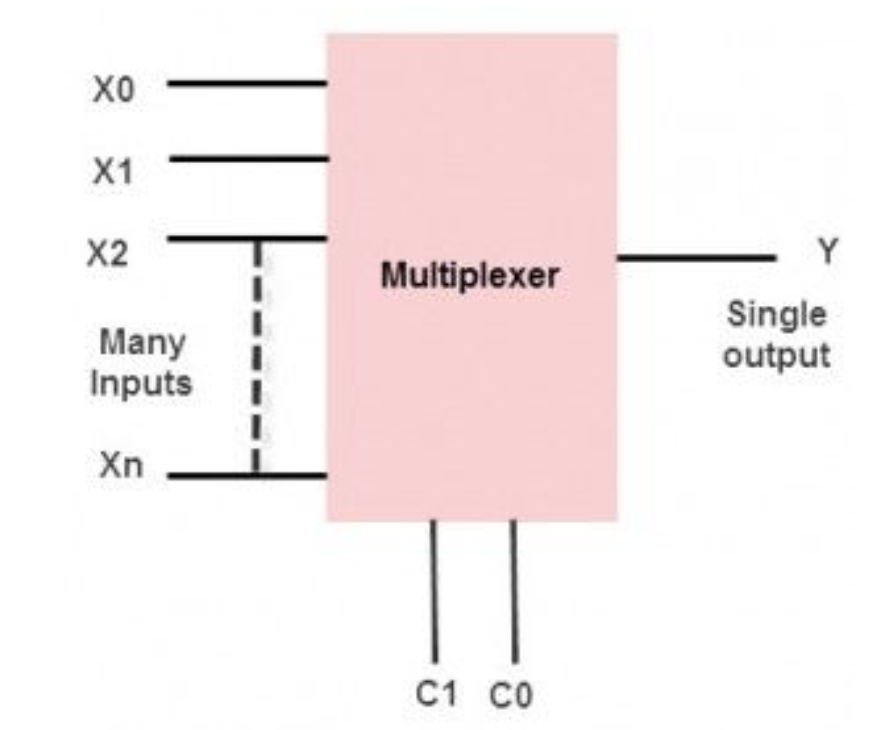


* 1. Bộ ghép kênh, bộ phân kênh
     1. Bộ ghép kênh

**Khái niệm**

Bộ ghép kênh là một thiết bị có nhiều đầu vào và đầu ra một dòng. Các dòng chọn xác định đầu vào nào được kết nối với đầu ra, đồng thời tăng lượng dữ liệu có thể được gửi qua mạng trong một thời gian nhất định. Nó còn được gọi là bộ chọn dữ liệu.

Công tắc đa vị trí một cực là một ví dụ đơn giản về mạch không điện tử của bộ ghép kênh và nó được sử dụng rộng rãi trong nhiều mạch điện . Bộ ghép kênh được sử dụng để thực hiện chuyển mạch tốc độ cao và được xây dựng bởi Linh kiện điện tử .



Bộ ghép kênh có khả năng xử lý cả tương tự và ứng dụng kỹ thuật số . Trong các ứng dụng tương tự, bộ ghép kênh được tạo thành từ rơ le và công tắc bóng bán dẫn, trong khi trong các ứng dụng kỹ thuật số, bộ ghép kênh được xây dựng từ tiêu chuẩn cổng logic . Khi bộ ghép kênh được sử dụng cho các ứng dụng kỹ thuật số, nó được gọi là bộ ghép kênh kỹ thuật số.

**Phân loại:**

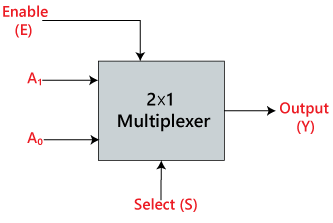
*Bộ ghép kênh được chia làm 4 loại:*

* Bộ ghép kênh 2-1 ( 1 dòng chọn )
* Bộ ghép kênh 4-1 ( 2 dòng chọn )
* Bộ ghép kênh 8-1 ( 3 dòng chọn )
* Bộ ghép kênh 16-1 ( 4 dòng chọn )

***Bộ ghép kênh 2-1***

Trong bộ ghép kênh 2×1, chỉ có hai đầu vào, tức là A 0 và A 1 , 1 đường lựa chọn, tức là S 0 và các đầu ra đơn, tức là Y. Trên cơ sở kết hợp các đầu vào có mặt tại vùng chọn dòng S 0 thì một trong 2 đầu vào này sẽ được nối với đầu ra. Sơ đồ khối và bảng chân lý của bộ ghép kênh 2 × 1 được đưa ra dưới đây.

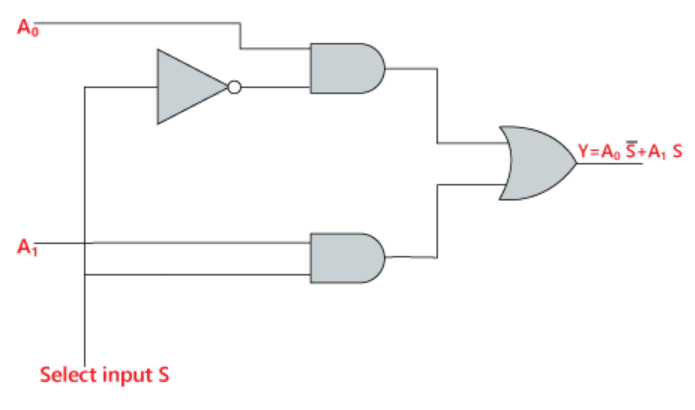
Sơ đồ khối:



Bảng sự thật:



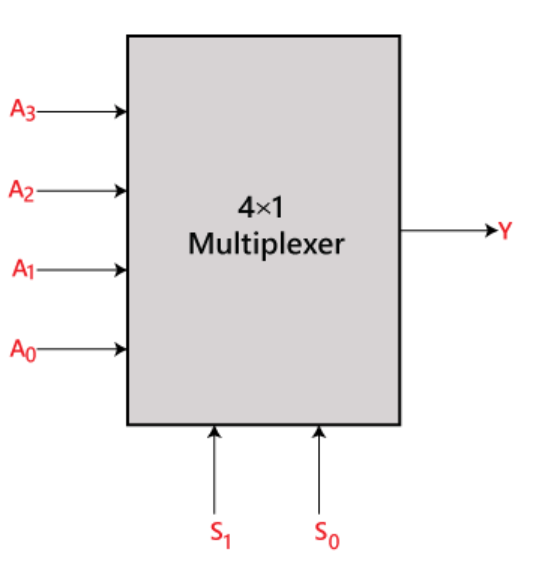
Mạch logic:



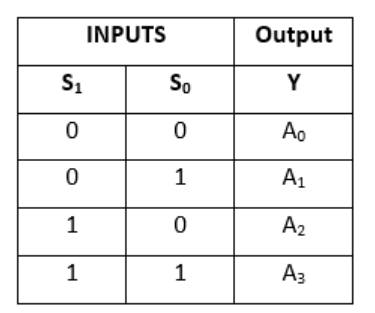
***Bộ ghép kênh 4-1***

Trong bộ ghép kênh 4×1, có tổng cộng bốn đầu vào, tức là A 0 , A 1 , A 2 và A 3 , 2 dòng lựa chọn, tức là S 0 và S 1 và một đầu ra, tức là Y. Bật trên cơ sở kết hợp các đầu vào có mặt tại các dòng lựa chọn S 0 và S 1 , một trong 4 đầu vào này được kết nối với đầu ra. Sơ đồ khối và bảng chân lý của bộ ghép kênh 4 × 1 được đưa ra dưới đây.

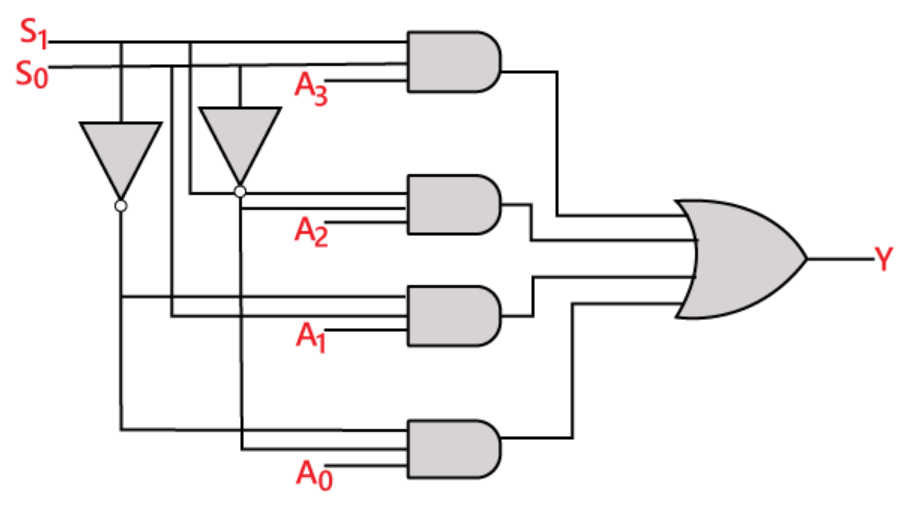
Sơ đồ khối:



Bảng sự thật:



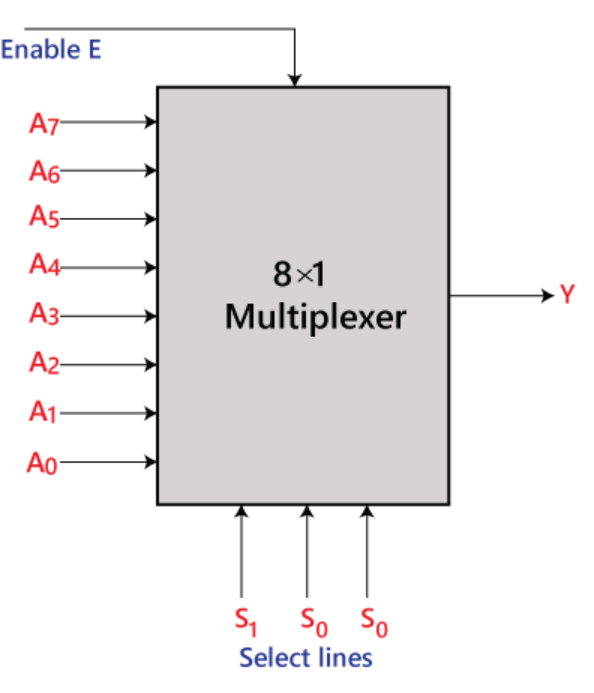
Mạch logic:



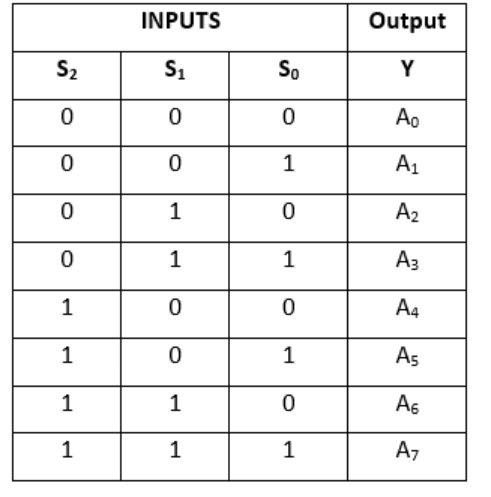
***Bộ ghép kênh 8-1***

Trong bộ ghép kênh 8 đến 1, có tổng cộng tám đầu vào, tức là A 0 , A 1 , A 2 , A 3 , A 4 , A 5 , A 6 và A 7 , 3 dòng lựa chọn, tức là S 0 , S 1 và S 2 và một đầu ra, tức là Y. Trên cơ sở kết hợp các đầu vào có tại các dòng lựa chọn S 0 , S 1 và S 2 , một trong 8 đầu vào này được kết nối với đầu ra. Sơ đồ khối và bảng chân lý của bộ ghép kênh 8 × 1 được đưa ra dưới đây.

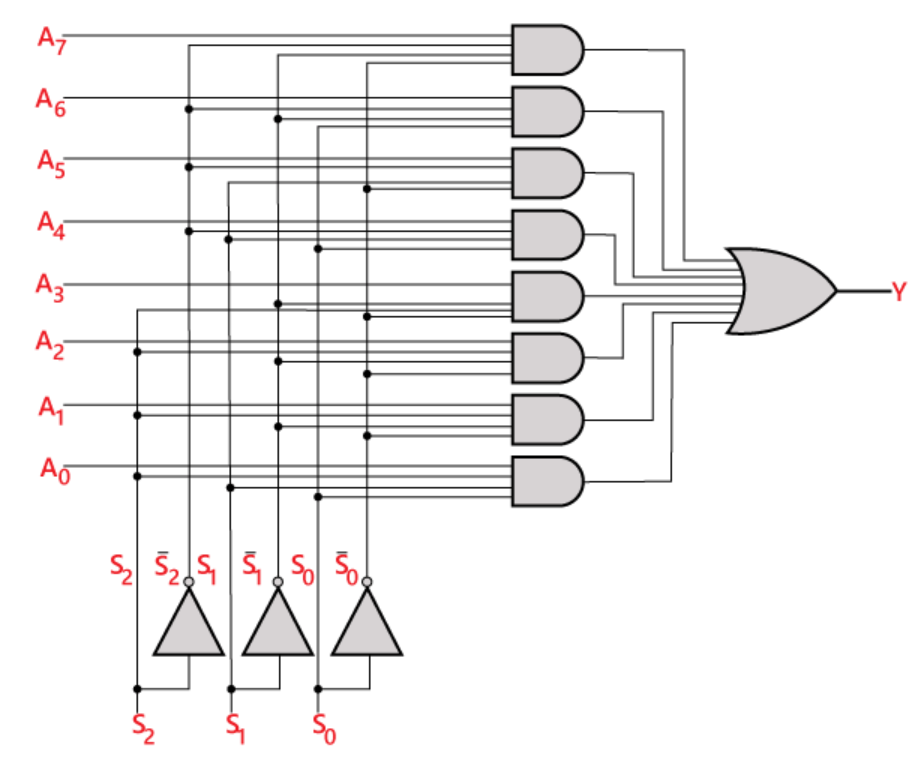
Sơ đồ khối:



Bảng sự thật:



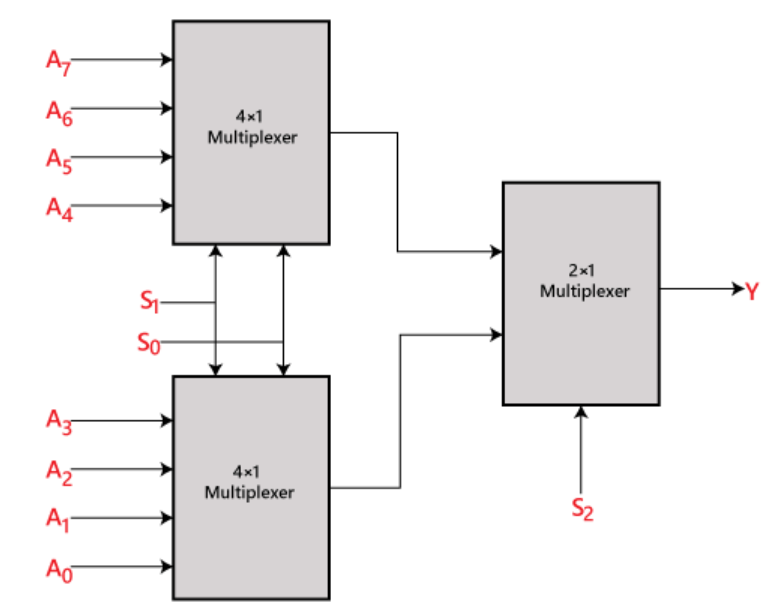
Mạch logic:



***Bộ ghép kênh 8-1 có thể sử dụng bộ ghép kênh 4-1 và 2-1:***

Chúng ta có thể triển khai bộ ghép kênh 8 × 1 bằng cách sử dụng bộ ghép kênh bậc thấp hơn. Để triển khai bộ ghép kênh 8 × 1, chúng ta cần hai bộ ghép kênh 4 × 1 và một bộ ghép kênh 2 × 1. Bộ ghép kênh 4 × 1 có 2 đường lựa chọn, 4 đầu vào và 1 đầu ra. Bộ ghép kênh 2 × 1 chỉ có 1 dòng lựa chọn.

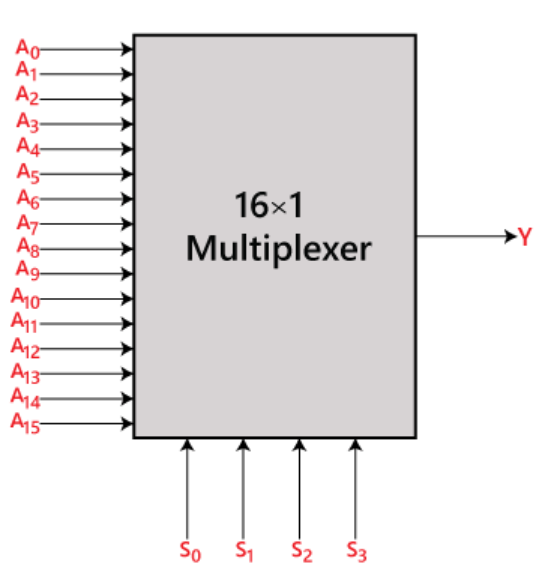
Để nhận được 8 đầu vào dữ liệu, chúng ta cần hai bộ ghép kênh 4 × 1. Bộ ghép kênh 4 × 1 tạo ra một đầu ra. Vì vậy, để có được đầu ra cuối cùng, chúng ta cần bộ ghép kênh 2 × 1. Sơ đồ khối của bộ ghép kênh 8 × 1 sử dụng bộ ghép kênh 4 × 1 và 2 × 1 được đưa ra dưới đây.



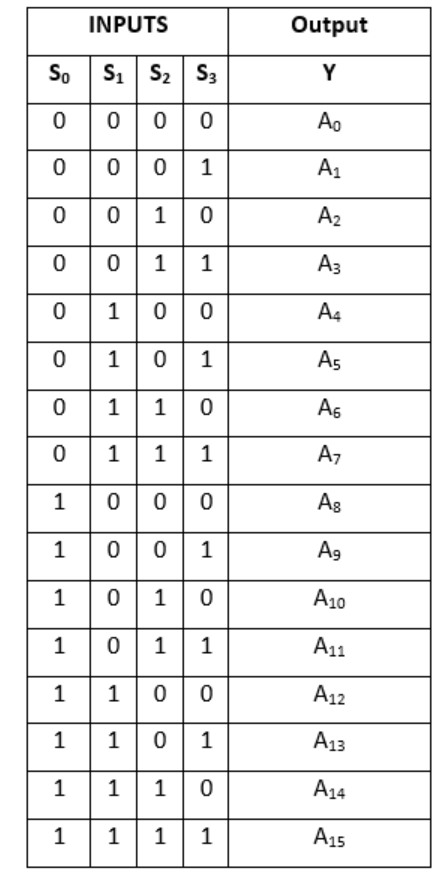
***Bộ ghép kênh 16-1***

Trong bộ ghép kênh 16 đến 1, có tổng cộng 16 đầu vào, tức là A 0 , A 1 , …, A 16 , 4 dòng lựa chọn, tức là S 0 , S 1 , S 2 , và S 3 và một đầu ra, tức là , Y. Trên cơ sở kết hợp các đầu vào có mặt tại các dòng lựa chọn S 0 , S 1 và S 2 , một trong 16 đầu vào này sẽ được kết nối với đầu ra. Sơ đồ khối và bảng chân lý của 16 × 1

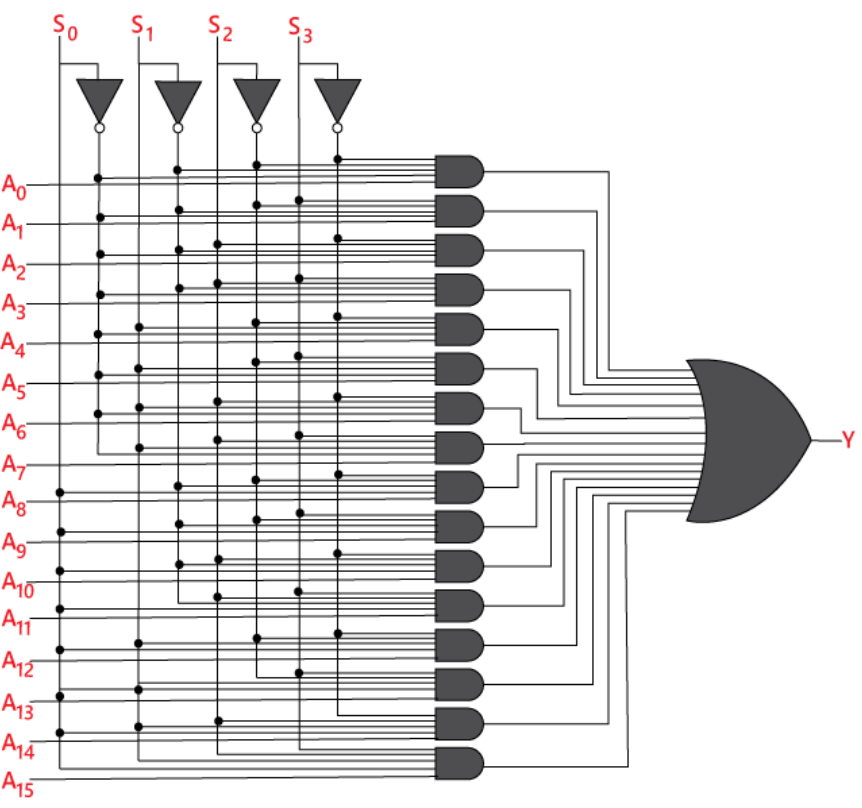
Sơ đồ khối:



Bảng sự thật:



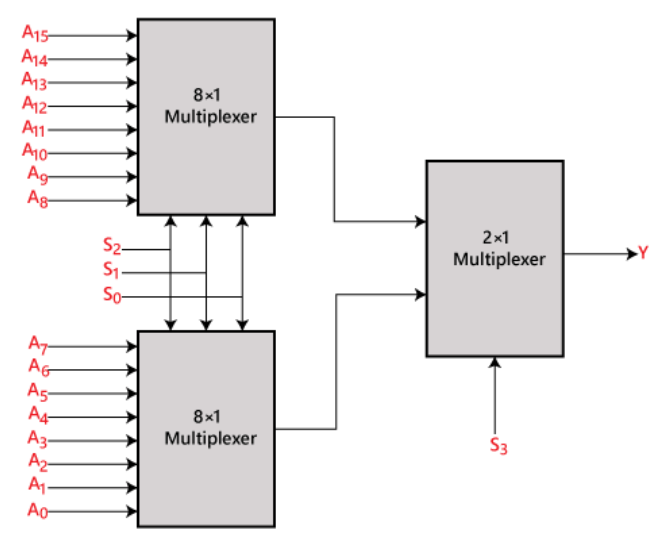
Mạch logic:



***Bộ ghép kênh 16-1 có thể sử dụng bộ ghép kênh 8-1 và 2-1***

Chúng ta có thể triển khai bộ ghép kênh 16 × 1 bằng cách sử dụng bộ ghép kênh bậc thấp hơn. Để triển khai bộ ghép kênh 8 × 1, chúng ta cần hai bộ ghép kênh 8 × 1 và một bộ ghép kênh 2 × 1. Bộ ghép kênh 8 × 1 có 3 đường lựa chọn, 4 đầu vào và 1 đầu ra. Bộ ghép kênh 2 × 1 chỉ có 1 dòng lựa chọn.

Để nhận được 16 đầu vào dữ liệu, chúng ta cần hai bộ ghép kênh 8 × 1. Bộ ghép kênh 8 × 1 tạo ra một đầu ra. Vì vậy, để có được đầu ra cuối cùng, chúng ta cần bộ ghép kênh 2 × 1. Sơ đồ khối của bộ ghép kênh 16 × 1 sử dụng bộ ghép kênh 8 × 1 và 2 × 1 được đưa ra dưới đây.



***Ưu điểm, nhược điểm***

1. Ưu điểm

Làm giảm chi phí cũng như độ phức tạp của mạch

Làm cho mạch truyền ít phức tạp hơn và tiết kiệm hơn

Tản nhiệt ít hơn do dòng chuyển mạch tương tự nằm trong khoảng từ 10mA đến 20mA.

Khả năng ghép kênh có thể được mở rộng để chuyển đổi tín hiệu âm thanh, tín hiệu video, v.v.

Độ tin cậy của hệ thống kỹ thuật số có thể được cải thiện bằng cách sử dụng MUX vì nó giảm số lượng kết nối có dây bên ngoài.

Thiết kế logic có thể được đơn giản hóa thông qua MUX

1. Nhược điểm

Cần có thêm độ trễ trong các cổng chuyển đổi và tín hiệu I / O truyền qua bộ ghép kênh.

Các cổng có thể được sử dụng đồng thời có những hạn chế

Việc chuyển đổi các cổng có thể được xử lý bằng cách thêm độ phức tạp của phần sụn

Việc kiểm soát bộ ghép kênh có thể được thực hiện bằng cách sử dụng các cổng I / O bổ sung.

1. Ứng dụng

**Hệ thống thông tin liên lạc**

Đến hệ thống thông tin liên lạc vừa có mạng truyền thông vừa có hệ thống truyền dẫn. Bằng cách sử dụng một bộ ghép kênh, hiệu quả của hệ thống thông tin liên lạc có thể được tăng lên bằng cách cho phép truyền dữ liệu, chẳng hạn như dữ liệu âm thanh và video từ các kênh khác nhau thông qua các đường dây hoặc cáp đơn.

**Bộ nhớ máy tính**

Bộ ghép kênh được sử dụng trong bộ nhớ máy tính để duy trì một lượng lớn bộ nhớ trong máy tính, và cũng để giảm số lượng đường đồng cần thiết để kết nối bộ nhớ với các bộ phận khác của máy tính.

**Mạng điện thoại**

Trong mạng điện thoại, nhiều tín hiệu âm thanh được tích hợp trên một đường truyền duy nhất với sự trợ giúp của bộ ghép kênh.

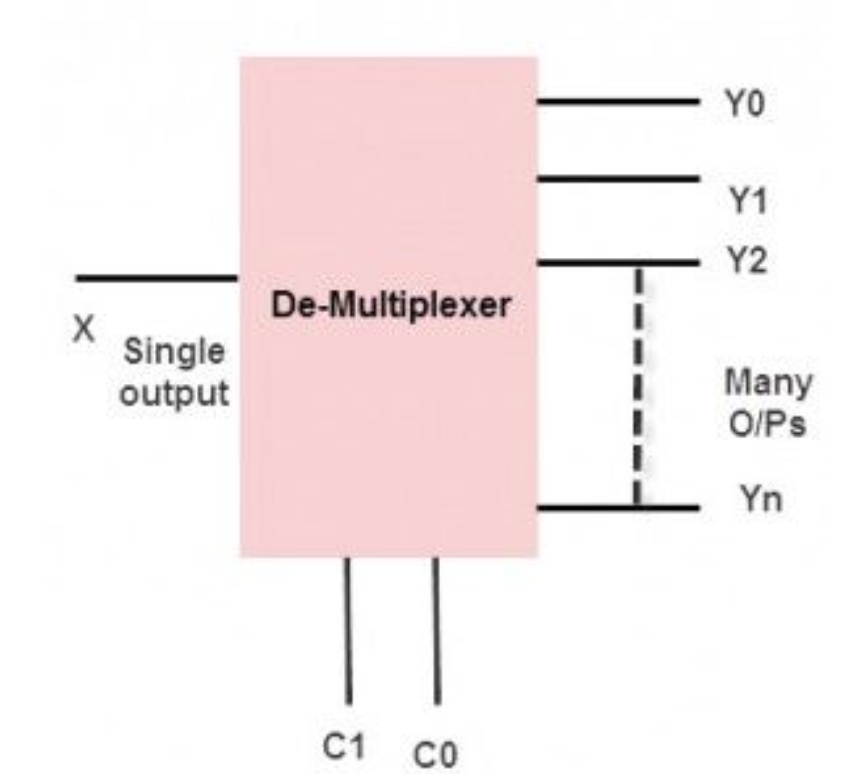
Truyền từ Hệ thống Máy tính của Vệ tinh

Bộ ghép kênh được sử dụng để truyền tín hiệu dữ liệu từ hệ thống máy tính của tàu vũ trụ hoặc vệ tinh tới hệ thống mặt đất bằng sử dụng vệ tinh GSM .

* + 1. Bộ dồn kênh (Demultiplexer)

***Khái niệm***

Bộ khử ghép kênh cũng là thiết bị có một đầu vào và nhiều đường ra. Nó được sử dụng để gửi tín hiệu đến một trong nhiều thiết bị. Sự khác biệt chính giữa bộ ghép kênh và bộ ghép kênh là bộ ghép kênh lấy hai hoặc nhiều tín hiệu và mã hóa chúng trên một sợi dây, trong khi bộ ghép kênh ngược lại với những gì bộ ghép kênh làm.



***Phân loại***

Bộ phân kênh được phân thành bốn loại

1-2 bộ tách kênh (1 dòng chọn)

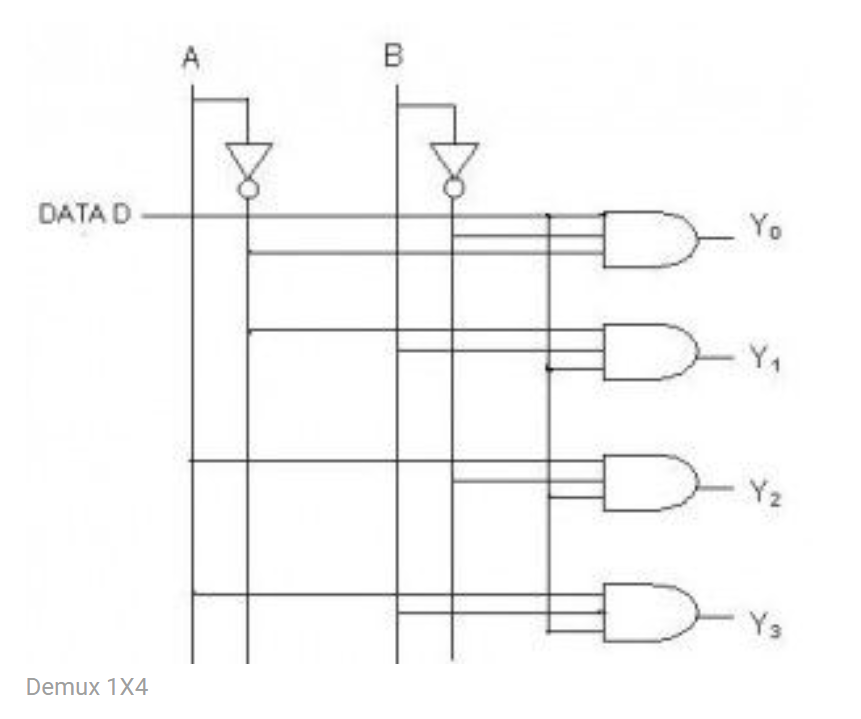
1-4 bộ tách kênh (2 dòng chọn)

1-8 bộ tách kênh (3 dòng chọn)

Bộ tách kênh 1-16 (4 dòng chọn)

***Bộ phân kênh 1-4***

Bộ phân kênh 1 đến 4 bao gồm 1 bit đầu vào, 4 bit đầu ra và các bit điều khiển. Sơ đồ mạch bộ tách kênh 1X4 được hiển thị bên dưới.



Bit i / p được coi là Data D. Bit dữ liệu này được truyền tới bit dữ liệu của các dòng o / p, giá trị này phụ thuộc vào giá trị AB và i / p điều khiển.

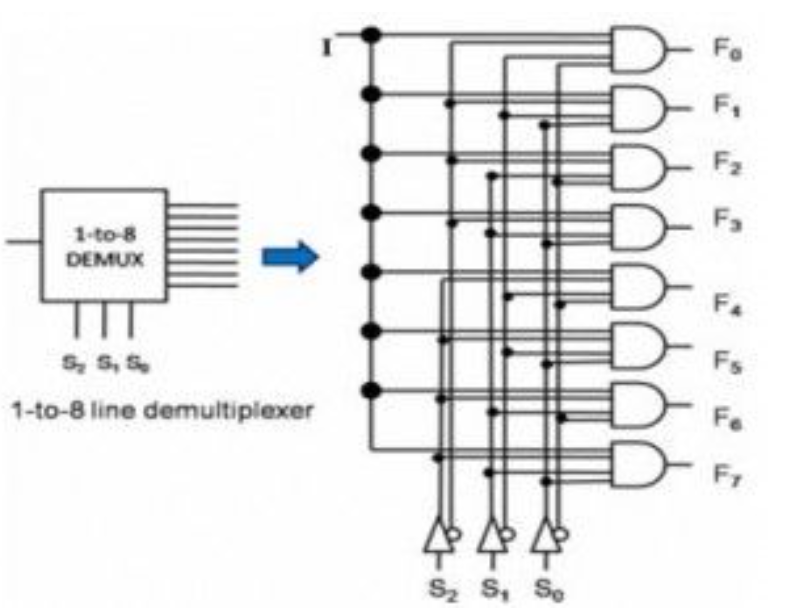
Khi điều khiển i / p AB = 01, cổng AND thứ hai phía trên được phép trong khi các cổng AND còn lại bị hạn chế. Do đó, chỉ có bit dữ liệu D được truyền tới đầu ra, và Y1 = Dữ liệu.

Nếu bit dữ liệu D ở mức thấp, đầu ra Y1 ở mức thấp. IF bit dữ liệu D cao, đầu ra Y1 cao. Giá trị của đầu ra Y1 phụ thuộc vào giá trị của bit dữ liệu D, các đầu ra còn lại ở trạng thái thấp.

Nếu đầu vào điều khiển thay đổi thành AB = 10, thì tất cả các cổng bị hạn chế ngoại trừ cổng AND thứ ba từ trên xuống. Sau đó, bit dữ liệu D chỉ được truyền tới đầu ra Y2 và Y2 = Dữ liệu. . Ví dụ tốt nhất về bộ phân kênh 1X4 là IC 74155.

***Bộ phân kênh 1-8***

Bộ tách kênh còn được gọi là bộ phân phối dữ liệu vì nó yêu cầu một đầu vào, 3 đường được chọn và 8 đầu ra. Bộ khử ghép kênh lấy một dòng dữ liệu đầu vào duy nhất và sau đó chuyển nó sang bất kỳ một trong các dòng đầu ra. Sơ đồ mạch bộ phân kênh 1 đến 8 được hiển thị bên dưới, nó sử dụng 8 cổng AND để đạt được hoạt động.



Bit đầu vào được coi là dữ liệu D và nó được truyền đến các dòng đầu ra. Điều này phụ thuộc vào giá trị đầu vào điều khiển của AB. Khi AB = 01, cổng thứ hai phía trên F1 được bật, trong khi các cổng AND còn lại bị vô hiệu hóa và bit dữ liệu được truyền đến đầu ra cho F1 = dữ liệu. Nếu D thấp thì F1 thấp và nếu D cao thì F1 cao. Vì vậy, giá trị của F1 phụ thuộc vào giá trị của D, và các đầu ra còn lại ở trạng thái thấp.

* 1. Mạch cộng

***Khái niệm***

Mạch cộng là một mạch điện tử thực hiện việc cộng số. Trong máy tính hiện đại phép cộng nằm bên trọng đơn vị xử lý số luận lý (ALU). Mặc dù các mạch cộng có thể được tạo ra cho nhiều hệ đếm, loại mạch cộng thường dùng nhất hoạt động trên hệ nhị phân.

Trong nhiều máy tính và các loại vi xử lý, bộ cộng không chỉ được sử dụng trong đơn vị logic số hoc, mà còn được sử dụng trong những phần khác của vi xử li, chúng được sử dụng để tính toán các địa chỉ, chỉ số bảng, toán tử tăng và giảm, và các toán tử tương đương.

Mặc dù bộ cộng có thể được xây dựng cho nhiều kiểu số, như Số thập phân mã hóa nhị phân hay excess-3, các bộ cộng thông dụng hoạt động trên số nhị phân.

Mạch cộng được chia làm 2 loại:

* Mạch cộng bán phần
* Mạch cộng toàn phần
  + 1. Mạch cộng bán phần

Mạch có khả năng thực hiện các phép cộng đơn giản với sự trợ giúp của các cổng logic. Trước tiên cùng tìm hiểu cộng 1 bit

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 10

Đây là những tổ hợp 1 bit. Tuy nhiên có 1 kết quả của phép cộng 1 + 1 là 10. Mặc dù vấn đề này có thể được giải quyết với sự trợ giúp của Cổng EXOR, nhưng nếu chúng ta quan tâm đến đầu ra thì kết quả tổng phải được viết lại dưới dạng đầu ra 2 bit.

Do đó các phương trình trên có thể được viết là:

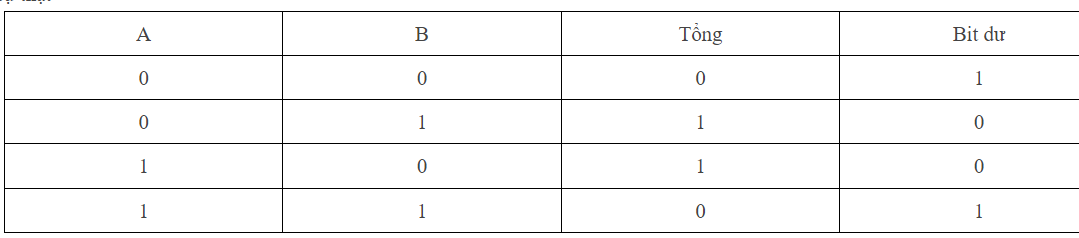
0 + 0 = 00

0 + 1 = 01

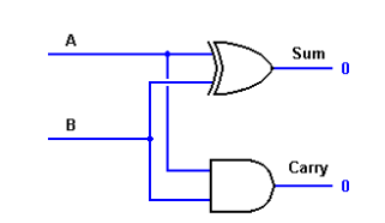
1 + 0 = 01

1 + 1 = 10

Bảng sự thật



Sơ đồ logic



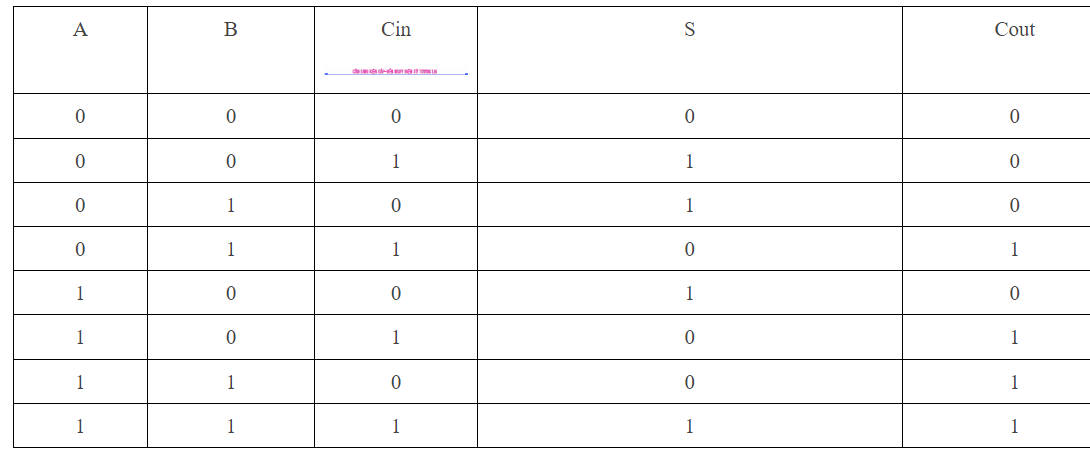
Để cộng 2 số nhiều bit hơn, mạch cộng bán phần không thể thực hiện được đối với những bit lớn hơn. Vì vậy mạch cộng toàn phần sẽ trợ giúp để thực hiện những phép tính phức tạp hơn.

* + 1. Mạch cộng toàn phần

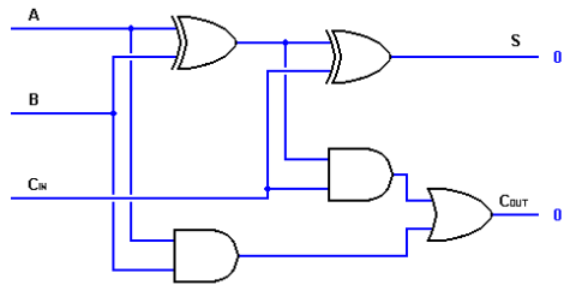
Đây là loại mạch khó thực hiện hơn một chút so với mạch cộng bán phần. Sự khác biệt chính giữa bộ cộng bán phần và bộ cộng toàn phần là bộ cộng toàn phần có ba ngõ vào và hai ngõ ra.

Hai đầu vào đầu tiên là A và B và đầu vào thứ ba là Cin chính là số dư của phép tính trước. Khi thiết kế bộ cộng toàn phần chúng ta sẽ có thể xâu chuỗi tám trong số chúng lại với nhau để tạo ra một bộ cộng rộng và xếp tầng mang bit từ bộ cộng này sang bộ tiếp theo.

Bảng sự thật:

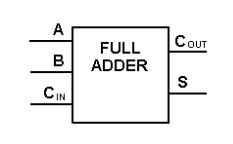


Sơ đồ logic:

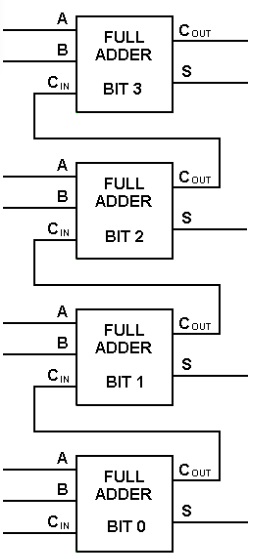


Để đơn giản sơ đồ trên thì người ta đã thiết kế một sơ đồ đơn giản hơn được sử dụng để thể hiện hoạt động của mạch.

Và dưới đây là sơ đồ đơn giản của bộ cộng đầy đủ 1 bit

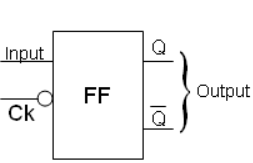


Với sơ đồ này, chúng ta có thẻ thực hiện cộng các bit cao hơn ví dụ như cộng 2 số 4 bit chúng ta sẽ ghép nối nối tiếp 4 bộ cộng 1 bit gồm 4 tầng.



* 1. Tổng quan về flip flop (FF)

FF là mạch có khả năng lật lại trạng thái ngõ ra tuỳ theo sự tác động thích hợp của ngõ vào, điều này có ý nghĩa quan trọng trong việc lưu trữ dữ liệu trong mạch và xuất dữ liệu ra khi cần. Có nhiều loại flip flop khác nhau, chúng được sử dụng rộng rãi trong nhiều ứng dụng. Các mạch FF thường được kí hiệu như sau



Nếu các ngõ vào sẽ quyết định ngõ ra là cái gì thì ngõ đồng hồ ck lại chỉ ra rằng khi nào mới có sự thay đổi đó. Chân Ck có thể tác động mức thấp hay mức cao tuỳ vào cấu trúc bên trong của từng IC FF, do đó với một IC FF cố định thì chỉ có một kiểu tác động và chỉ một mà thối, ví dụ với IC 74112 chỉ có một cách tác động là xung Ck tác động theo cạnh xuống.

Bộ nhớ: flip-flop được sử dụng để lưu trữ dữ liệu tạm thời trong các ứng dụng như bộ nhớ RAM (Random Access Memory).

Đồng bộ hóa dữ liệu: Flip-flops thường được sử dụng trong các mạch điện tử để đồng bộ hóa dữ liệu. Chúng giữ trạng thái cho đến khi một tín hiệu đồng bộ hóa đến, sau đó chuyển đổi trạng thái và truyền tín hiệu tiếp theo.

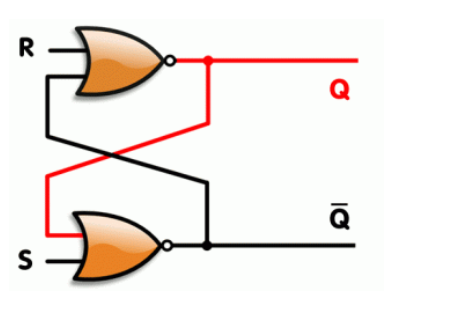
Điều khiển thời gian: Flip-flops cũng được sử dụng trong việc điều khiển thời gian trong mạch điện tử. Chúng có thể được sử dụng để chia nhỏ tín hiệu thời gian thành các chu kỳ nhỏ hơn và điều khiển tần số hoạt động của hệ thống.

Xây dựng các bộ đếm: Trong các ứng dụng đếm, flip-flops thường được sử dụng để tạo ra các bộ đếm đơn giản và phức tạp. Chúng có thể được kết hợp với các cổng logic để tạo ra các bộ đếm với các tính năng và chức năng cụ thể.

Lưu trữ trạng thái hệ thống: Trong hệ thống điều khiển, flip-flops được sử dụng để lưu trữ trạng thái hệ thống, đảm bảo rằng các thông tin và trạng thái hiện tại được lưu trữ và duy trì trong quá trình hoạt động của hệ thống.

* + 1. Các loại flip flop

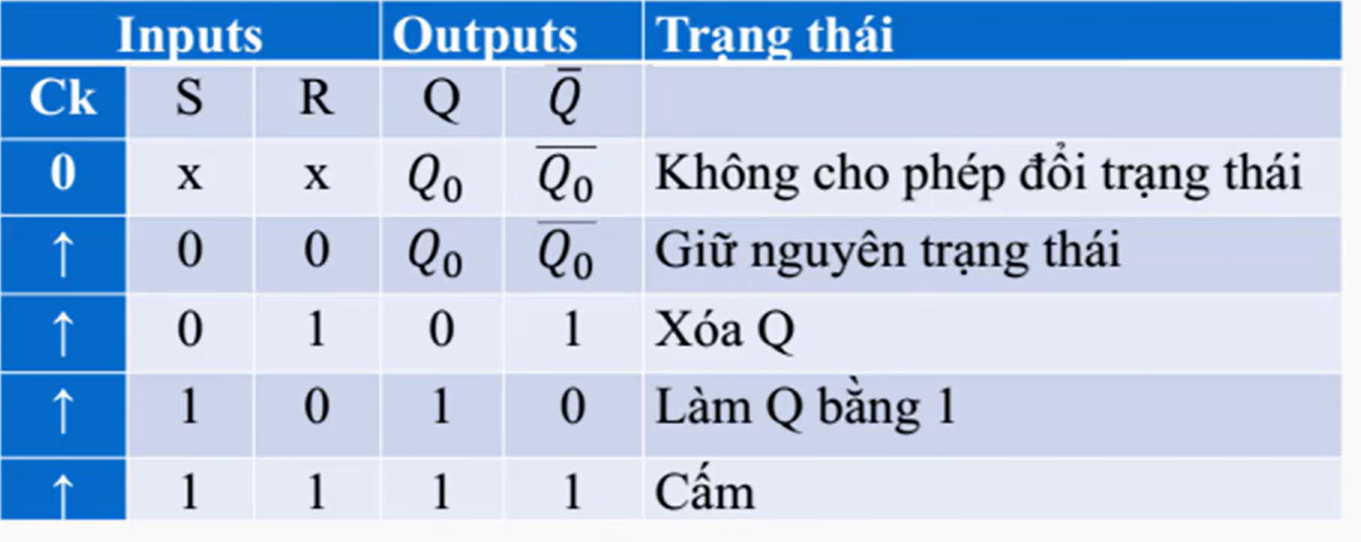
***FF SR(mạch lật lại đặt)***



Mạch lật SR (SR flip-flop) là một loại mạch lưu trữ dữ liệu trong điện tử kỹ thuật số. Mạch lật SR có hai đầu vào chính: Set (đặt) và Reset (xóa), cùng với một đầu ra và một tín hiệu đồng hồ để đồng bộ hóa hoạt động của nó. Mạch này có hai trạng thái đầu ra cơ bản: "0" và "1", và nó thường được sử dụng để lưu trữ một bit dữ liệu.

Mạch lật SR được sử dụng trong nhiều ứng dụng điện tử kỹ thuật số, nhưng có thể dễ dàng dẫn đến trạng thái không ổn định nếu không được sử dụng cẩn thận. Điều này đã dẫn đến sự phát triển của các loại mạch lật khác như mạch lật D, mạch lật JK và mạch lật T, để giải quyết các vấn đề liên quan đến mạch lật SR.

Mạch lật SR (Set-Reset flip-flop . Mạch lật SR có hai đầu vào chính: Set (đặt) và Reset (xóa), cùng với một đầu ra và một tín hiệu đồng hồ để đồng bộ hóa hoạt động của nó. Dưới đây là cách mạch lật SR hoạt động:



Trạng thái ban đầu:

Đầu ra của mạch lật SR được đặt ở một giá trị nào đó, có thể là "0" hoặc "1", tùy thuộc vào điều kiện ban đầu của mạch hoặc tín hiệu đầu vào trước đó.

Đầu vào Set (S):

Khi tín hiệu Set (S) được kích hoạt và ở mức logic "1", đầu ra của mạch lật SR sẽ được đặt thành "1". Điều này có nghĩa rằng nó giữ giá trị "1" cho đến khi có một tín hiệu Reset được kích hoạt.

Đầu vào Reset (R):

Khi tín hiệu Reset (R) được kích hoạt và ở mức logic "1", đầu ra của mạch lật SR sẽ được xóa thành "0". Điều này có nghĩa rằng nó giữ giá trị "0" cho đến khi có một tín hiệu Set được kích hoạt.

Trạng thái không xác định:

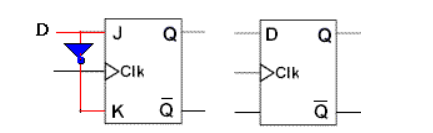
Nếu cả hai đầu vào Set và Reset đều ở mức logic "1" cùng một lúc, thì trạng thái của mạch lật SR không xác định. Điều này gây ra tình trạng không ổn định và không được sử dụng trong các ứng dụng thực tế.

Đồng bộ hóa:

Tín hiệu đồng hồ thường được sử dụng để đồng bộ hóa hoạt động của mạch lật SR, đảm bảo rằng việc đặt và xóa chỉ xảy ra tại những điểm cụ thể trong chu kỳ đồng hồ.

Mạch lật SR thường được sử dụng trong các ứng dụng nơi bạn cần lưu trữ một bit dữ liệu, như trong bộ lưu trữ đệm, các ứng dụng đồng bộ hóa, và các mạch chuyển đổi và điều khiển. Tuy nhiên, cần phải cẩn thận khi sử dụng nó để tránh trạng thái không ổn định do tín hiệu Set và Reset bị kích hoạt cùng một lúc.

***Mạch D***



Mạch lật D (D flip-flop) là một loại mạch lưu trữ dữ liệu trong hệ thống điện tử kỹ thuật số. Mạch lật D hoạt động như một bộ lưu trữ 1-bit dữ liệu và có khả năng lưu trữ trạng thái trạng thái "1" hoặc "0". Điểm mạnh của mạch lật D là nó có khả năng đồng bộ hóa trạng thái lưu trữ với tín hiệu đồng hồ.

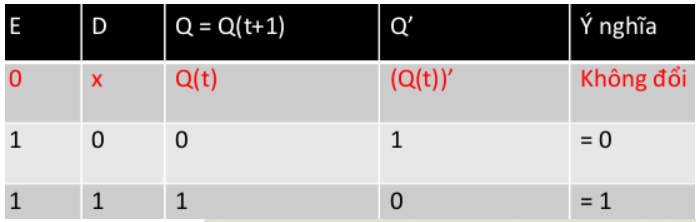
Cấu trúc cơ bản của mạch lật D bao gồm các đầu vào chính sau đây:

Đầu vào D: Đây là tín hiệu dữ liệu đầu vào mà bạn muốn lưu trữ. Khi tín hiệu đồng hồ (clock) được kích hoạt, trạng thái của mạch lật D sẽ được cập nhật dựa trên giá trị của đầu vào D.

Đầu ra Q: Đây là đầu ra của mạch lật D, hiển thị trạng thái lưu trữ của nó sau khi xảy ra sự kiện đồng hồ.

Tín hiệu đồng hồ (Clock): Tín hiệu đồng hồ được sử dụng để đồng bộ hóa hoạt động của mạch lật D. Khi xảy ra cạp đồng hồ (tức là tín hiệu đồng hồ chuyển từ "0" sang "1"), trạng thái của đầu vào D sẽ được sao chép vào đầu ra Q.

Cách hoạt động cơ bản của mạch lật D:

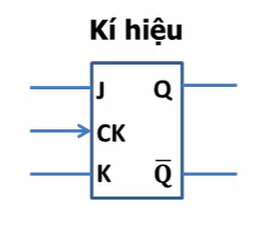


Khi tín hiệu đồng hồ chuyển từ "0" sang "1" (cạp đồng hồ), giá trị của đầu vào D sẽ được sao chép vào đầu ra Q. Nó giữ trạng thái mới cho đến khi cạp đồng hồ tiếp theo.

Khi tín hiệu đồng hồ không hoạt động, trạng thái của đầu ra Q được giữ nguyên, không thay đổi.

Mạch lật D thường được sử dụng rộng rãi trong các ứng dụng điện tử kỹ thuật số, như bộ lưu trữ đệm, bộ đếm, các mạch chuyển đổi và điều khiển, và trong việc đồng bộ hóa các hoạt động của các phần tử điện tử khác trong hệ thống.

***Mạch JK***



Mạch lật JK (JK flip-flop) là một loại mạch lưu trữ dữ liệu trong hệ thống điện tử kỹ thuật số. Nó là một loại flip-flop cung cấp tính năng đa dạng hơn so với mạch lật SR (Set-Reset) thông thường. Mạch lật JK được thiết kế để giải quyết một số vấn đề liên quan đến mạch lật SR, đặc biệt là trạng thái không xác định khi cả hai đầu vào Set và Reset đều ở mức logic "1" cùng một lúc.

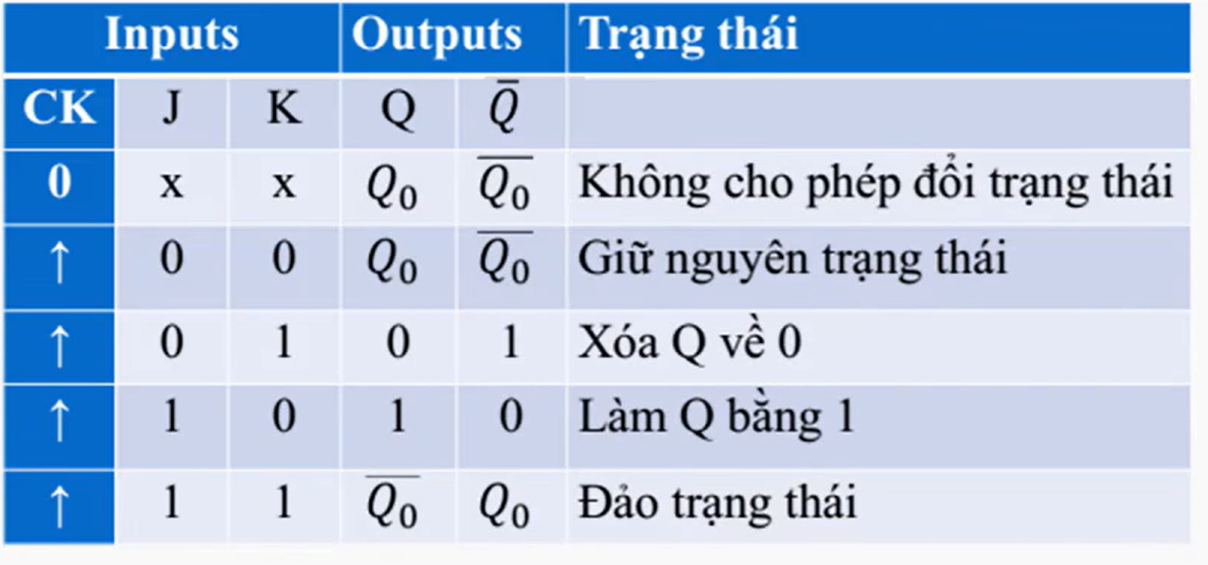
Mạch lật JK có ba đầu vào chính:

J (Set): Đây là đầu vào Set, tương tự như đầu vào Set trong mạch lật SR. Khi tín hiệu J ở mức logic "1," nó gây hiệu ứng đặt trạng thái đầu ra của mạch lật JK thành "1."

K (Reset): Đây là đầu vào Reset, tương tự như đầu vào Reset trong mạch lật SR. Khi tín hiệu K ở mức logic "1," nó gây hiệu ứng xóa trạng thái đầu ra của mạch lật JK thành "0."

Tín hiệu đồng hồ (Clock): Tín hiệu đồng hồ được sử dụng để đồng bộ hóa hoạt động của mạch lật JK. Khi xảy ra cạp đồng hồ (tức là tín hiệu đồng hồ chuyển từ "0" sang "1"), trạng thái của mạch lật JK sẽ được cập nhật dựa trên giá trị của đầu vào J và K.

Cách hoạt động cơ bản của mạch lật JK:



Khi tín hiệu đồng hồ chuyển từ "0" sang "1" (cạp đồng hồ):

Nếu J và K đều ở mức logic "0," trạng thái của mạch lật JK giữ nguyên.

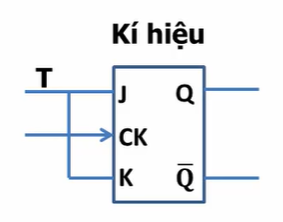
Nếu J ở mức logic "1" và K ở mức logic "0," mạch lật JK sẽ được đặt thành "1" (được đặt).

Nếu J ở mức logic "0" và K ở mức logic "1," mạch lật JK sẽ được xóa thành "0" (được xóa).

Nếu J và K đều ở mức logic "1," trạng thái của mạch lật JK sẽ chuyển đổi (đảo).

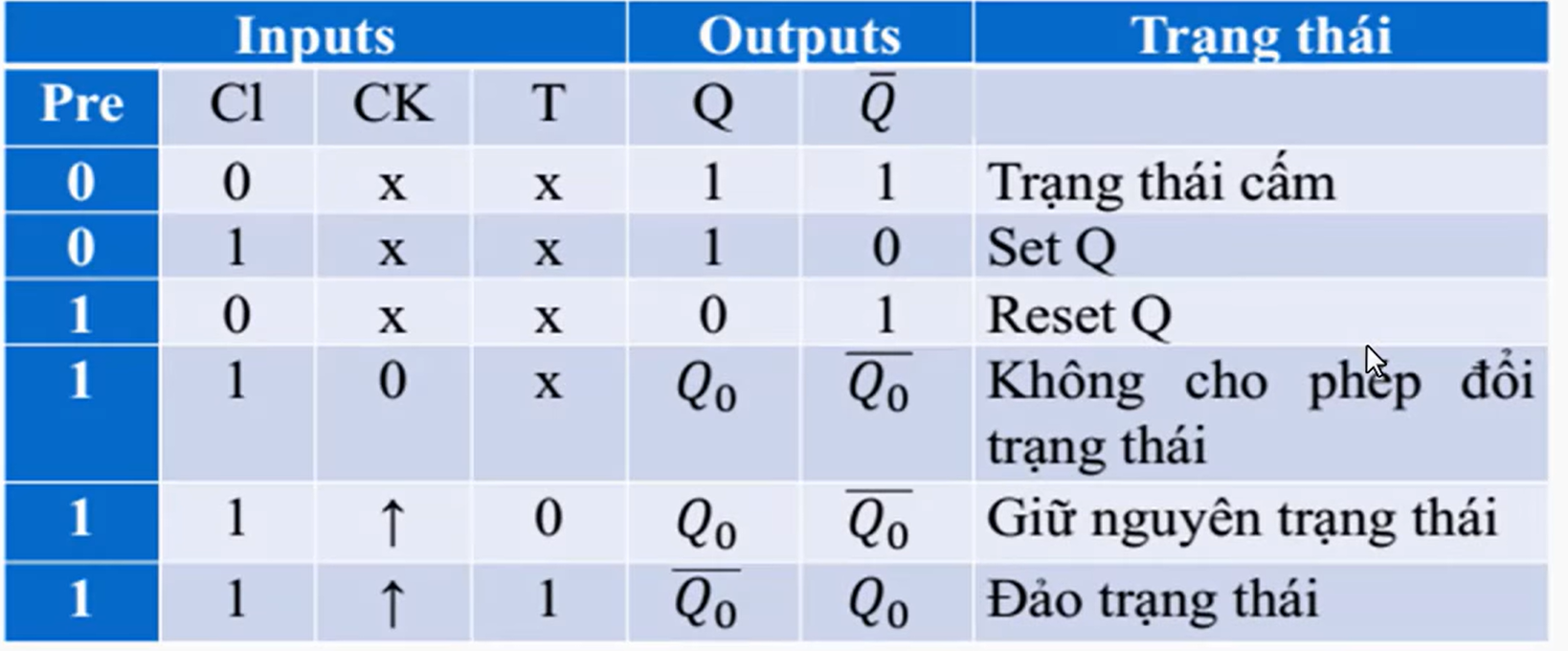
Mạch lật JK được sử dụng rộng rãi trong các ứng dụng điện tử kỹ thuật số, đặc biệt là khi cần đồng bộ hóa và kiểm soát trạng thái trong các mạch chuyển đổi, đếm, và các ứng dụng đồng bộ hóa khác.

***Mạch T***



Mạch T trong flip-flop, còn được gọi là mạch lật T (T flip-flop), là một loại mạch lưu trữ dữ liệu trong hệ thống điện tử kỹ thuật số. Mạch lật T thường được sử dụng để thực hiện hoặc giữ trạng thái đảo ngược của mạch lật (flip-flop) hiện tại. Nó cung cấp tính năng toggle (đảo) trạng thái của đầu ra mà không cần đến các đầu vào Set và Reset như trong mạch lật SR hoặc J-K.

Cấu trúc cơ bản của mạch lật T bao gồm các đầu vào sau:



Đầu vào T (Toggle): Đây là đầu vào chính của mạch lật T. Khi tín hiệu T ở mức logic "1," nó gây hiệu ứng đảo trạng thái của mạch lật T, nghĩa là nếu trạng thái trước đó là "0," thì nó sẽ chuyển thành "1," và nếu trạng thái trước đó là "1," thì nó sẽ chuyển thành "0."

Tín hiệu đồng hồ (Clock): Tín hiệu đồng hồ được sử dụng để đồng bộ hóa hoạt động của mạch lật T. Khi xảy ra cạp đồng hồ (tức là tín hiệu đồng hồ chuyển từ "0" sang "1"), trạng thái của mạch lật T sẽ được cập nhật dựa trên giá trị của đầu vào T.

Cách hoạt động cơ bản của mạch lật T:

Khi tín hiệu đồng hồ chuyển từ "0" sang "1" (cạp đồng hồ):

Nếu đầu vào T ở mức logic "0," trạng thái của mạch lật T giữ nguyên (không đổi).

Nếu đầu vào T ở mức logic "1," trạng thái của mạch lật T đảo ngược (nếu trước đó là "0," thì chuyển thành "1," và ngược lại).

Mạch lật T được sử dụng trong nhiều ứng dụng điện tử kỹ thuật số, đặc biệt là khi bạn cần kiểm soát và đảo trạng thái của các mạch lật một cách định kỳ hoặc theo một lịch trình cụ thể. Nó cũng thường được sử dụng để thiết lập một đầu ra đồng bộ theo tín hiệu đồng hồ hoặc để tạo chu kỳ động cơ trong các ứng dụng điện tử.

* 1. Mạch tuần tự

Mạch tuần tự là một hệ thống hoặc mạng lưới các thiết bị điện tử hoạt động theo trình tự, tức là các thiết bị hoạt động một sau một, thay vì cùng một lúc. Trong mạch tuần tự, thông tin hoặc dữ liệu được xử lý từng phần, từ thiết bị này đến thiết bị khác theo trình tự cố định.

Mạch tuần tự thường có một số lượng lớn các thiết bị hoặc phần tử gọi là "đơn vị xử lý." Mỗi đơn vị xử lý thực hiện một phần cụ thể của quá trình xử lý hoặc tính toán dữ liệu và sau đó chuyển giao dữ liệu đã xử lý đến đơn vị xử lý tiếp theo trong chuỗi.

Mạch tuần tự thường được sử dụng trong các ứng dụng yêu cầu xử lý dữ liệu theo trình tự, chẳng hạn như trình tự hoạt động của máy tính, truyền dữ liệu qua mạng, quy trình sản xuất tự động trong công nghiệp, hoặc bất kỳ ứng dụng nào mà việc thực hiện các tác vụ theo trình tự là cần thiết.

Mạch tuần tự thường đi kèm với một quy trình quản lý trình tự, giúp điều khiển các bước cụ thể và xác định thứ tự của các hoạt động. Trong nhiều trường hợp, mạch tuần tự cũng có thể được lập trình để thực hiện các quy trình khác nhau dựa trên điều kiện và dữ liệu đầu vào cụ thể.

Ví dụ về mạch tuần tự trong cuộc sống hàng ngày có thể là mạch tự động trong một dây chuyền sản xuất, trong đó mỗi máy móc hoặc robot thực hiện một công đoạn cụ thể để lắp ráp sản phẩm cuối cùng.

Nguyên lý hoạt động của mạch tuần tự liên quan đến việc xử lý thông tin hoặc dữ liệu theo trình tự cố định và trong một thứ tự cụ thể. Dưới đây là cách mạch tuần tự hoạt động:

Trình tự cố định: Mạch tuần tự hoạt động theo một trình tự xác định trước, trong đó các bước hoặc hoạt động được thực hiện một sau một. Thứ tự này có thể được quyết định trước hoặc dựa trên logic hoặc điều kiện cụ thể. Trình tự cố định đảm bảo rằng mỗi bước hoặc đơn vị xử lý được kích hoạt theo đúng thứ tự.

Đơn vị xử lý: Mạch tuần tự bao gồm một số đơn vị xử lý (hoặc bước xử lý), mỗi đơn vị có nhiệm vụ thực hiện một phần cụ thể của quá trình. Các đơn vị xử lý có thể là máy móc, thiết bị điện tử, hoặc phần mềm được thiết kế để thực hiện công việc cụ thể.

Điều kiện và logic: Mạch tuần tự có thể được thiết lập để phản ứng theo các điều kiện hoặc tình huống cụ thể. Điều này có nghĩa là việc thực hiện các bước hoặc chuyển đổi trạng thái có thể dựa trên logic hoặc điều kiện cụ thể. Điều này cho phép mạch tuần tự thích ứng với các tình huống biến đổi.

Đồng bộ hóa: Mạch tuần tự thường được điều khiển bằng tín hiệu đồng hồ hoặc tín hiệu đồng bộ khác để đảm bảo rằng mỗi đơn vị xử lý hoặc bước hoạt động chỉ được kích hoạt vào thời điểm cụ thể trong trình tự. Điều này giúp đồng bộ hóa và điều khiển quá trình tuần tự.

Lặp lại: Mạch tuần tự có thể được thiết kế để lặp lại trình tự hoạt động nhiều lần nếu cần thiết. Lặp lại này có thể xảy ra theo số lần cố định hoặc dựa trên điều kiện cụ thể.

Xử lý dữ liệu: Trong quá trình tuần tự, dữ liệu có thể được xử lý, biến đổi hoặc chuyển đổi từ đầu vào thành đầu ra cụ thể, tuân theo các qui tắc và thuật toán được đặt ra.

Mạch tuần tự thường được sử dụng trong một loạt các ứng dụng, từ tự động hóa sản xuất công nghiệp đến điều khiển máy tính và các hệ thống điện tử kỹ thuật số khác, để xử lý dữ liệu theo trình tự cụ thể.

Có nhiều loại mạch tuần tự, và chúng có thể được phân loại dựa trên cách chúng lưu trữ và xử lý dữ liệu cũng như trạng thái làm việc. Dưới đây là một số loại mạch tuần tự phổ biến:

Mạch Đồng Bộ (Synchronous Circuits):

Mạch tuần tự đồng bộ sử dụng một tín hiệu đồng hồ chung để đồng bộ hóa các hoạt động bên trong mạch.

Flip-flops thường được sử dụng trong các mạch đồng bộ để lưu trữ và truy cập dữ liệu.

Mạch Bất Đồng Bộ (Asynchronous Circuits):

Mạch tuần tự bất đồng bộ không sử dụng tín hiệu đồng hồ chung và thay vào đó sử dụng các tín hiệu kiểm soát riêng biệt để đồng bộ hóa các hoạt động.

Các loại mạch như latches bất đồng bộ có thể được sử dụng trong mạch này.

Mạch Tuần Tự Về Thời Gian (Timed Sequential Circuits):

Mạch này hoạt động dựa trên một chuỗi thời gian cố định và có các trạng thái quy định và kiểm soát thời gian.

Mạch Tuần Tự Sự Kiểm Soát (Controlled Sequential Circuits):

Mạch này thay đổi trạng thái dựa trên các tín hiệu kiểm soát bên ngoài, nhưng không phụ thuộc vào đồng hồ.

Mạch Tuần Tự Dựa Trên Trạng Thái (State-Based Sequential Circuits):

Mạch này dựa vào trạng thái hiện tại để quyết định trạng thái tiếp theo và đối ứng với các sự kiện.

Mạch Tuần Tự Dựa Trên Giới Hạn (Limited Sequential Circuits):

Mạch này được giới hạn bởi một số trạng thái cố định và quyết định dựa trên trạng thái hiện tại để chuyển đổi giữa chúng.

Mạch tuần tự thường được sử dụng trong nhiều ứng dụng, từ máy tính và vi xử lý đến hệ thống điều khiển tự động và các loại thiết bị điện tử khác. Sự lựa chọn của loại mạch tuần tự phụ thuộc vào yêu cầu cụ thể của ứng dụng.