*GIÁO TRÌNH*

# SẢN XUẤT TỰ ĐỘNG

*( 45tiết )*

**DÙNG CHO SINH VIÊN NGÀNH CHẾ TẠO MÁY**

***Châu Mạnh Lực***

# I. KHÁI NIỆM VỀ SẢN XUẤT TỰ ĐỘNG

## 1.1. Khái niệm

Ngày nay, để nâng cao năng suất lao động, nâng cao chất lượng và ổn định chất lượng sản phẩm, người ta đã đưa vào các dây chuyền thiết bị sản xuất trong công nghiệp các hệ thống điều khiển tự động từng phần hoặc toàn bộ quá trình sản xuất.

Cùng với việc sử dụng ngày càng nhiều các hệ thống sản xuất tự động, con người đã được cải thiện đáng kể điều kiện lao động như giảm nhẹ được sức lao động, tránh được sự nhàm chán trong công việc, tạo điều kiện cho họ được tiếp cận với các lĩnh vực tiến bộ khoa học kỹ thuật và được làm việc trong môi trường ngày càng văn minh hơn.

Trong nền kinh tế thị trường và điều kiện hội nhập sâu rộng vào nền kinh tế thế giới, vấn đề cạnh tranh càng ngày càng khốc liệt hơn trên nhiều lĩnh vực như chất lượng, mẫu mã và giá thành sản phẩm. Có thể thấy rằng chỉ có thể áp dụng tự động vào quá trình sản xuất mới có thể có cơ hội nâng cao năng suất, tạo tiền đề cho việc giảm giá thành sản phẩm, đồng thời đảm bảo chất lượng ổn định của sản phẩm cũng như có thể thay đổi mẫu mã sản phẩm một cách nhanh chóng.

Ngày nay, hầu hết các dây chuyền sản xuất sử dụng các hệ thống tự động đã cho phép các doanh nghiệp có thể thay đổi công nghệ một cách dễ dàng và thuận lợi với các bộ điều khiển khả trình như trên các máy công cụ điều khiển theo chương trình số CNC, trên các bộ điều khiển logic khả trình PLC. Hơn thế nữa, với sự phát triển nhanh chóng của lĩnh vực công nghệ truyền thông đã cho phép ứng dụng các lĩnh vực tổ chức và điều hành toàn bộ các quá trình sản xuất một cách tối ưu nhất bằng việc sử dụng các công nghệ điều khiển như sử dụng mạng Petri, GRAFCET... trong quá trình sản xuất linh hoạt FSM và trong sản xuất tích hợp CIM.

## 1.2. Mục tiêu của tự động hoá quá trình sản xuất.

Về cơ bản có thể nhận thấy rằng, tự động hoá quá trình sản xuất bên cạnh những mặt tích cực của nó như đã nêu trên thì vẫn có những vấn đề cũng phải cần quan tâm và phải có sự phân tích một cách kỹ lưỡng để có những giải pháp ứng dụng hợp lý, đó là vấn đề xã hội. Vì sự áp dụng tự động hoá nhanh vào các hệ thống thiết bị công nghiệp sẽ kéo theo một lực lượng lớn công nhân thất nghiệp, mặt khác

quá trình chuẩn bị nhân lực có trình độ nghề nghiệp thích ứng chưa đáp ứng được một cách động bộ, đặc biệt là trong quá trình đào tạo. Hầu hết các doanh nghiệp có mức độ tự động hoá cao đều sử dụng các công nhân củ được đào tạo lại để kịp thời phục vụ trước mắt, bởi vậy họ cũng chỉ được nắm bắt những kỹ năng và thao tác hết sức cơ bản mà chưa đạt đến trình độ theo yêu cầu. Do vậy cần phải nhìn nhận vấn đề một cách có khoa học để xác lập được mục tiêu của tự động hoá.

Có thể thấy rằng, tự động hoá quá trình sản xuất là nhằm đạt đến các mục tiêu chủ yếu sau đây:

Giảm giá thành sản phẩm do việc tăng năng suất, giảm các chi phí về vật tư tiêu hao, giảm tỹ lệ phế phẩm.

Loại bỏ hoàn toàn những công việc nặng nhọc, độc hại và nguy hiểm đến tính mạng con người, ví dụ như trong các lò phản ứng hạt nhân, trong công tác thám hiểm vũ trụ và thăm dò đáy biển....

Thay thế hoàn toàn cho con người trong quá trình kiểm tra, giám sát chất lượng sản phẩm, nhờ vậy loại trừ được các yếu tố chủ quan mà làm cho chất lượng sản phẩm ổn định hơn.

Thực hiện các động tác hết sức chính xác và khéo léo mà con người không thể làm được như chế tạo và lắp ráp các chi tiết và linh kiện cực nhỏ như các vi mạch, chíp xử lý, trong công nghệ nanô cũng như trong công tác tìm kiếm và sửa chữa các hư hỏng và sự cố trong quá trình vận hành và khai thác các thiết bị...

### 1.3. Khả năng thích ứng nhanh và tạo nhanh các sản phẩm

Bằng các kỹ thuật hiện đại, ngày nay các nhà sản xuất công nghiệp có thể rút ngắn đến mức tối đa thời gian kể từ khi hình thành ý tưởng đến khi cho ra đời các sản phẩm như sử dụng các công nghệ tích hợp, các phần mần mềm thiết kế, quản lý và chế tạo như công nghệ CAD/CAM.

### 1.4. Cấu trúc chung của một hệ thống sản xuất tự động

Mỗi một hệ thống sản xuất tự động phải bao hàm trong đó các khối điều khiển và khối chấp hành. Chức năng công nghệ cùng với các chương trình hoạt động được chuyển đến cho khối điều khiển dưới dạng các chương trình cứng hay mềm thông qua các thao tác trực tiếp của con người thông qua bàn phím lập chương trình  hay từ các vật mang tin như các loại băng đột lỗ, đĩa mềm, băng từ hoặc cáp truyền số liệu. Khối điều khiển sẽ nhận và xữ lý các thông tin thông qua các hệ thống tính toán số học, logic và nội suy sẽ phát ra các lệnh tương thích tại từng thời điểm thích hợp cho khối chấp hành nhằm thực hiện các chức năng công tác của quá

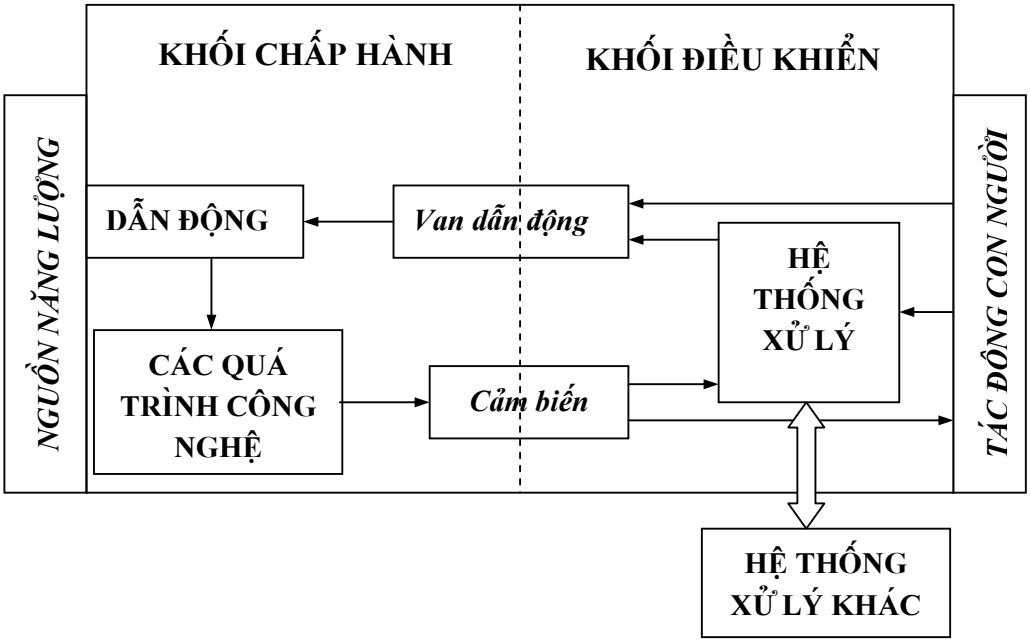
trình công nghệ như dịch chuyển bàn máy, thực hiện công việc ép, cung cấp phôi liệu , nguyên liệu, trộn hoặc tháo sản phẩm, quay bàn máy đến vị trí tiếp theo...

Để khối chấp hành có thể hoạt động được cũng như để giám sát sự hoạt động của các cơ cấu chấp hành, người ta còn bố trí thêm trong hệ thống các cơ cấu trung gian như các van đóng mở các thiết bị dẫn động và các cảm biến để giám sát, theo dõi sự hoạt động của hệ thống nhằm điều chỉnh sự hoạt động một cách nhanh chóng, kịp thời và chuẩn xác.

Để đảm bảo cho sự hoạt động ổn định của hệ thống, cần phải cung cấp nguồn năng lượng như điện, dầu ép hoặc khí nén có chất lượng và cần phải có sự tham gia của con người như lập chương trình, thay đổi chương trình hoạt động, khởi động chu trình cũng như can thiệp trực tiếp một cách kịp thời khi cần thiết như hệ thống có sự cố hoặc có thông báo lỗi...

Ngày nay, với kỹ thuật truyền số liệu đã phát triển ở mức độ cao, các hệ thống sản xuất tự động còn thực hiện được sự giao tiếp với các máy tính và các mạng truyền thông cục bộ trong từng phân xưởng, trong nhà máy (LAN) hoặc cả đối với các hệ thống sản xuất tự động khác ở các địa điểm khác nhau (NET).

Có thể tóm tắt sơ đồ cấu trúc của một hệ tự động nói chung :




*Hình 1.1 : Sơ đồ hệ thống sản xuất tự động*

## 1.5. Khối chấp hành

*1.5.1. Các hệ thống dẫn động*

Mục đích của hệ thống dẫn động trong hệ thống sản xuất tự động là nhằm thực hiện các chức năng công tác của quá trình sản xuất. Yêu cầu  cơ bản đối với các hệ thống dẫn động này là có thể điều khiển tốc độ vô cấp để nhằm thoã mãn với nhiều chế độ công nghệ khác nhau cũng như đảm bảo sự hoạt động của hệ thống một cách ổn định nhằm chống lại các ảnh hưởng do nhiễu tác động. Đối với hệ thống dẫn động, ngoài yêu cầu trên, còn phải đảm bảo có hệ số cứng vững động lực học cao, đáp ứng nhanh và nhạy đối với các tín hiệu điều khiển. Có độ bền cơ học cao và ổn định với sự hoạt động trong thời gian dài và liên tục. Tuỳ theo yêu cầu về chế độ công nghệ của từng hệ thống và các quá trình công nghệ mà có sự phân tích và lựa chọn các hệ thống này một cách thích hợp.

Có thể phân hệ dẫn động theo 2 dạng cơ bản trên cơ sở nguồn năng lượng : điện và thủy khí.

*a. Dẫn động điện*

Dẫn động điện được sử dụng phổ biến nhất hiện nay do nguồn cung cấp năng lượng điện thông dụng và phổ biến nhất. Sử dụng năng lượng điện rất thuận tiện và đơn giản vì hầu hết các doanh nghiệp đều có nguồn năng lượng điên, có thể nguồn năng lượng từ hệ thống điện quốc gia, nguồn năng lượng điện từ các máy phát điện cục bộ hoặc các nguồn năng lượng điện khác như phong điện, thuỷ điện hoặc năng lượng mặt trời. Hơn nữa các tín hiệu điều khiển và các tín hiệu giám sát cùng chung một đại lượng vật lý nên việc điều khiển tự động là thuận lợi nhất. Tuy nhiên, dẫn động điện chỉ có thể tạo ra các chuyển động quay, vì thế muốn tạo ra các chuyển động khác theo yêu cầu như chuyển động tịnh tiến, chuyển động lắc theo chu kỳ hoặc các chuyển động quay trong góc quay giới hạn cần phải có thêm các cơ cấu cơ khí tham gia. Đối với các động cơ chạy điện, người ta thường sử dụng các loại sau đây:

** Động cơ 1 chiều kích từ độc lập.*

Loại động cơ điện 1 chiều đã được sử dụng từ rất sớm. Ưu điểm của loại động cơ này là có dãi công suất rất rộng, có thể từ vài w đến hàng trăm kw. Khoảng điều chỉnh tốc độ cũng rất lớn, mạch điều chỉnh tốc độ đã rất ổn định và có hệ số động lực học cao. Tuy nhiên do sử dụng cổ góp nên có thể gây ra hiện tượng nhiễu hoặc bị mài mòn, đặc biệt khi làm việc trong các môi trường có độ ẩm cao hoặc có

nhiều thành phần ăn mòn như các vùng ven biển, do vậy cần phải thường xuyên bảo dưỡng để đảm bảo chế độ làm việc của động cơ.

** Động cơ bước.*

Động cơ bước là một loại máy điện hoạt động dưới tác dụng của các xung rời rạc và kế tiếp. Nó có thể quay theo cả 2 chiều tùy thuộc vào thứ tự cung cấp điện cho các cực của *stato*. Mỗi lần quay do tác động của 1 xung được 1 bước tương ứng với một góc của trục động cơ và dừng lại chính xác dưới tác dụng của điện - từ trường. Bước là lượng dịch chuyển về góc quay nhỏ nhất được xác định bởi 2 vị trí ổn định kế tiếp nhau. Trị số này thường được xác định theo số bước trong một vòng quay của động cơ. Thông thường là từ 6 ÷ 200 bước / vòng. Đối với loại động cơ bước hỗn hợp làm việc theo nguyên tắc từ trở nhỏ nhất có thể tăng số bước của động cơ một cách đáng kể nhằm tăng khả năng độ dịch chuyển tinh nhằm thỏa mãn theo yêu cầu của quá trình công nghệ.

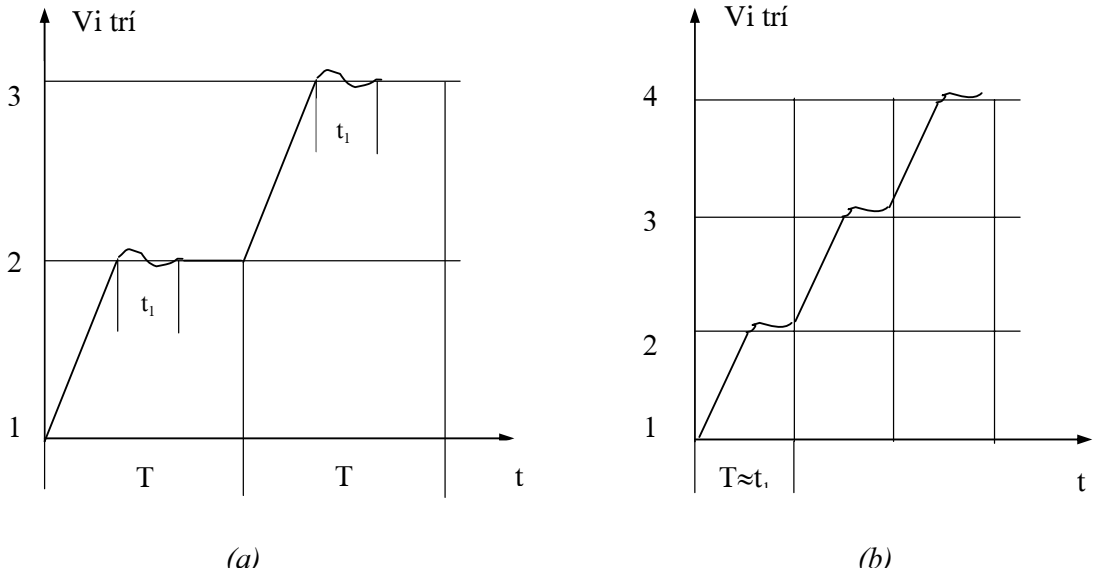
Nguyên lý làm việc của động cơ bước tương tự như nguyên lý làm việc của động cơ không đồng bộ : phần cố định (võ động cơ) là *stato* có các cuộn dây lắp đều trên chu vi có dạng vòng nhẫn đặt vào các rãnh *stato* có dạng chữ U để tạo nên từ lực đẩy nam châm. Phần quay của động cơ là *rotor* được gắn với trục thường được chế tạo từ nam châm vĩnh cửu hay sắt non. Tại mỗi thời điểm, vị trí của chúng được xác định theo giá trị của bước mà tại đó từ trở của chúng là nhỏ nhất trong trường hợp thao tác đầy bước và có thể nằm ở một vị trí thích hợp với từ trở nhỏ nhất trong trường hợp điều khiển vi bước. Như vậy, có thể thấy là quá trình hoạt động của động cơ bước là gián đoạn theo từng bước hay vi bước, trong đó xẩy ra liên tục quá trình khởi động và dừng theo từng xung cung cấp. Tại vị trí dừng, mô men điện từ tác dụng giữ roto ổ vị trí chính xác và người ta còn gọi là mô men hãm.

Tần số xung sẽ quyết định tốc độ quay của động cơ bước, tần số xung càng lớn thì tốc độ quay càng cao và ngược lại. Tuy nhiên không thể tăng tần số lên quá lớn vì khi đó hằng số thời gian điện từ sẽ giảm xuống và thời gian dừng của động cơ bước (để dập tắt các dao động) có thể vượt quá hằng số thời gian điện từ và như vậy sẽ tạo nên sự hoạt đông liên tục và không thỏa mãn yêu cầu của động cơ bước. Hơn nữa, khi đó động cơ cũng không thể đảo chiều được. Đây gọi là hiện tượng bội tốc hay vùng tần số làm việc quá giới hạn.

Sự hoạt động của động cơ bước được biểu diễn qua đặc tính động lực của chúng như hình 1.2. Trong đó hình *(a)* biểu diễn sự hoạt động bình thường của động cơ bước với quá trình khởi động - dừng và lại khởi động tiếp tục bước thứ 2, 3

... theo cùng chiều hoặc ngược chiều mà không có sự sai lệch bước. Thời gian giữa 2 xung đủ đảm bảo làm tắt hết dao động và rôto sẽ dừng để thực hiện tiếp tục xung kế tiếp.

Hình *(b)* biểu diễn vùng bội tốc, tại vùng này, động cơ không đáp ứng tức thời các lệnh khởi động - dừng đúng vị trí và bước bị sai lệch hoặc không thể đảo chiều được. Trong trường hợp này, hằng số thời gian điện từ có giá trị xấp xỉ với thời gian cần thiết để mô mem hãm của động cơ gây ra để dừng *rotor* đúng vị trí bước.




*Hình 1.2: Sự dịch chuyển của động cơ bước*

Mối quan hệ giữa mô men và tần số bước hay tốc độ quay của động cơ bước tương tự như động cơ không đồng bộ 3 pha và được biểu diễn qua đặc tính cơ của chúng. Mỗi hãng sản xuất động cơ bước đều có các đặc tính cơ khác nhau và thường được cung cấp theo khi bán hàng.

Từ đồ thị đặc tính cơ, ta có một số nhận xét liên quan đến điều kiện làm việc và chế độ sử dụng của động cơ bước như sau:

Tần số tới hạn $f_t$ : là tần số bước lớn nhất mà động cơ có thể làm việc mà không bị sai lệch bước quay khi có tải trọng. Tần số bước tới hạn lớn nhất $f_{tmax}$ là tần số mà động cơ bước có thể đạt được khi quay không tải.

Tần số khởi động $f_k$ là tần số bước lớn nhất mà động cơ có thể khởi động khi có tải. Tần số khởi động lớn nhất $f_{kmax}$ là tần số bước lớn nhất mà tại đó động cơ có thể khởi động không tải.

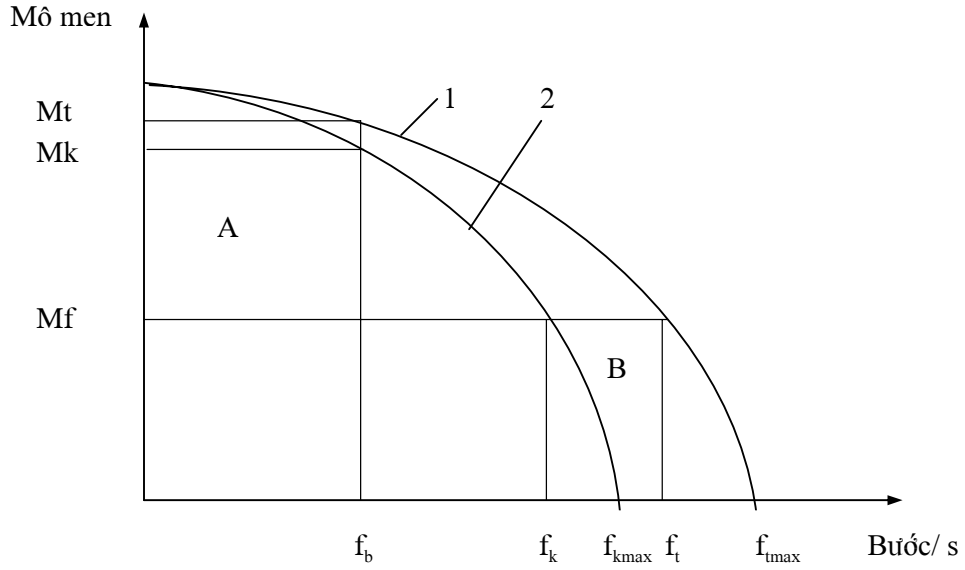
Môмen tới hạn Mt : là mô men lớn nhất tác động lên trục mà động cơ có thể quay tương ứng với tần số tới hạn. Đường *(1)* trên hình vẽ đặc trưng cho mô men tới hạn theo sự thay đổi của tần số bước.

Mô men khởi động Mk : là mô men lớn nhất mà động cơ có thể quay với tần số khởi động. Sự thay đổi của Mk theo tần số được biểu thị bằng đường *(2)*.

Từ hai đường đặc tính đã tạo nên hai vùng làm việc của động cơ bước. Vùng *A* là vùng khởi động - dừng ; vùng *B* là vùng bội tốc và không nên dùng.

Mf và $f_b$ là mô men tải và tần số bước.

Ngoài ra, như đã nói trên, động cơ bước còn có mô men duy trì (mô men tĩnh) do từ trường của cuộn dây *stato* tạo ra để giữ động cơ đứng yên (dừng đúng tại vị trí bước).

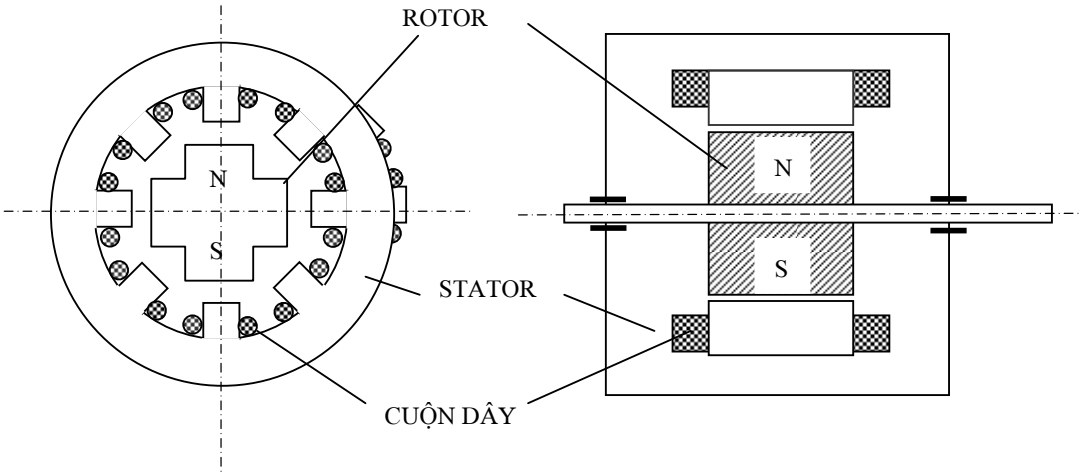



*Hình 1.3: Đặc tính cơ động cơ*

Cấu trúc của động cơ bước như đã mô tả ở trên có thể được biểu diễn như hình 1.4.

Tuy nhiên, để nâng cao công suất của động cơ bước cũng như để tăng số bước của động cơ, người ta ghép đồng trục nhiều động cơ bước được biểu diễn như

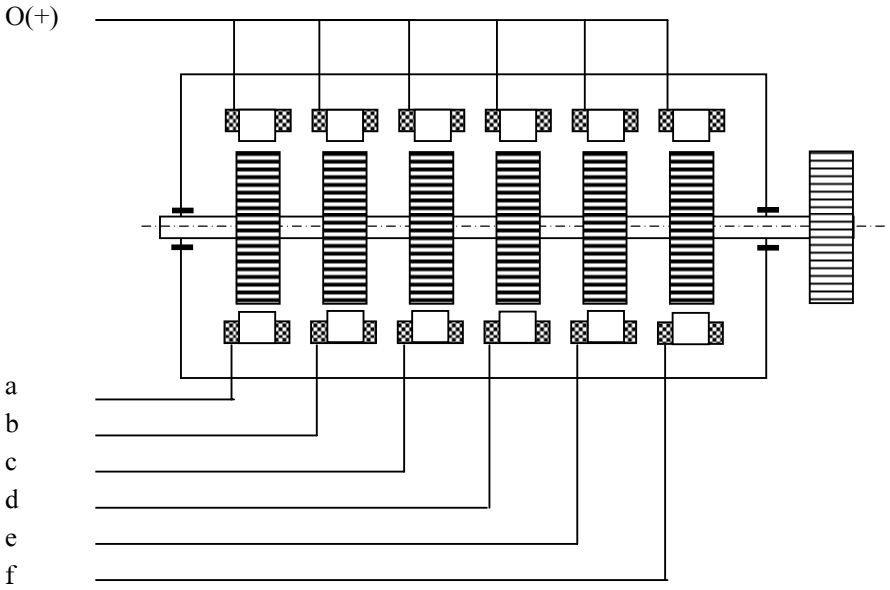
trên để tăng mô men và khi đó các cực (răng) của s*tato* được bố trí lệch đi một góc góc nào đó sao cho phân bố góc răng chia đều cho số s*tato* được ghép.




*Hình 1.4 : cấu tạo của động cơ bước*

Ví dụ có 6 s*tato* ghép lại và số răng (bước s*tato*) của s*tato* là 20, ta sẽ có các cực của s*tato* xếp lệch trên đường sinh 1 góc là :

$$\varphi_l = \frac{360^0}{20 \times 6} = 3^0$$




*Hình 1.5: biểu diễn động cơ bước và sự bố trí bước Stato.*

Việc điều khiển động cơ bước được thực hiện do một thiết bị điện tử thực hiện được gọi là bộ chuyển phát. Tuy vậy trước khi đi đến việc nghiên cứu cấu trúc bộ điều khiển này, ta thiết lập công thức xác định số bước của động cơ để từ đó xây dựng quá trình điều khiển cho thích hợp.

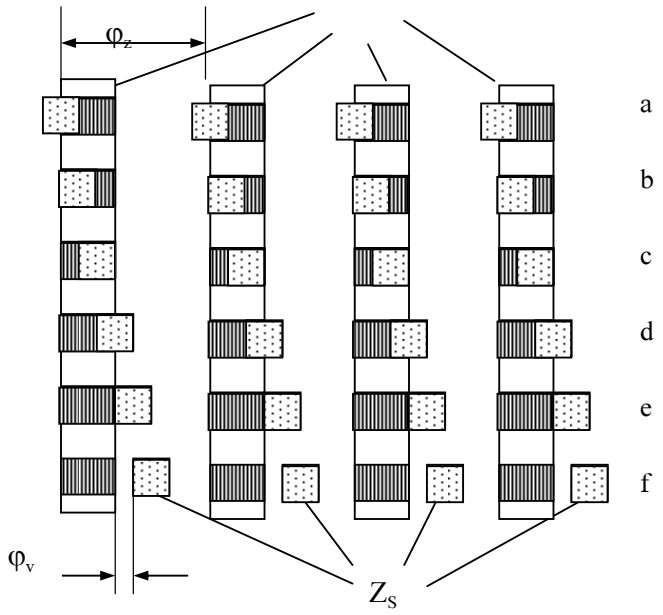
Nếu gọi số răng của s*tato* là Zs và số răng của *roto*r là Zr thì số bước $n_b$ của động cơ bước được tính :

$$n_b = \frac{Z_s \times Z_r}{Z_s - Z_r}$$

Nếu được ghép đồng trục nhiều *rotor* và *stato* theo hình 8 thì số bước của động cơ bước được tính.

$$n_b = \frac{Z_s \times Z_r}{Z_s - Z_r} \times k$$

Trong đó, k là số lượng *rotor* và *stato* được ghép.




*Hình 1.6. Sơ dồ bố trí các cực của động cơ bước*

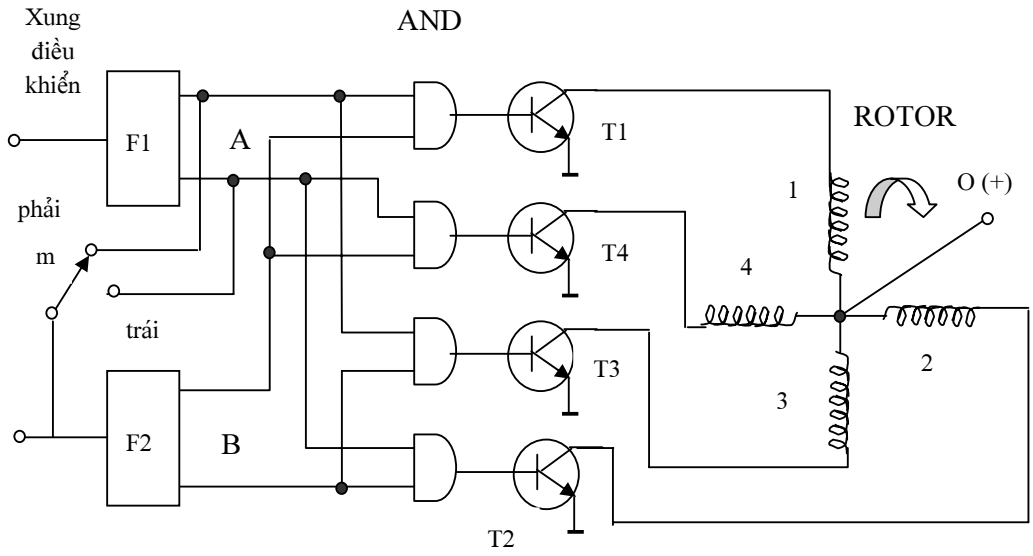
Hệ điều khiển động cơ bước phải đảm bảo được các chức năng sau đây:

Cung cấp đủ số lượng xung cần thiết vào cuộn dây *stato* theo yêu cầu công nghệ. Tạo các xung với các tần số khác nhau và tần số này có thể thay đổi được theo quy luật mong muốn.

- Chuyển các xung điện áp vào các cuộn dây *stato* theo yêu cầu về chiều quay một cách chính xác.

- Làm giảm được các dao động cơ học.

Hệ thống này được thực hiện bằng một sơ đồ *logic* mạch điện tử số được bố trí như trên hình 1.7.




CHIỀU QUAY 1234

| m | A | B | T1 | T2 | T3 | T4 |
|---|---|---|---|---|---|---|
| PHẢI | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

CHIỀU QUAY 1432

| m | A | B | T1 | T2 | T3 | T4 |
|---|---|---|---|---|---|---|
| TRÁI | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

*Hình 1.7. Mạch điều khiển động cơ bước*

Ví dụ trên hình 1.7 là biểu diễn sơ đồ bộ chuyển phát sử dụng mạch *logic* điện tử số thực hiện điều khiển 1 động cơ bước có 4 cuộn s*tato* tương ứng với các cuộn dây được đánh số 1,2,3,4.

Đầu vào được cung cấp bằng các xung do máy phát xung hoặc bộ tạo xung phát ra. Thành phần của mạch là các linh kiện bán dẫn và vi mạch. Việc kích thích các cực của động cơ bước thực hiện theo thứ tự 1 - 2 - 3 - 4 do các *Transitor* công suất thực hiện. (T1, T2, T3, T4) thông qua các cổng *logic* thực hiện. Nhờ sự thay đổi trình tự bộ chuyển phát cung cấp cho *Transitor* qua các cổng logic *AND* mà chiều quay của động cơ có thể thay đổi được. Hai bộ ổn định P1 và P2 tạo các tín hiệu đóng mở cho các cổng điều khiển. Trạng thái của chúng có thể điều khiển động cơ quay theo chiều kim đồng hồ hoặc ngược lại.

**Động cơ không đồng bộ 3 pha điều khiển bằng biến tần.*

Động cơ 3 pha sử dụng biến tần để điều khiển tốc độ mới được nghiên cứu và ứng dụng gần đây, tuy nhiên nó đã được phát triển một cách nhanh chóng do có nhiều ưu điểm nổi trội hơn các loại động cơ trên như mô men mở máy lớn, mô men quá tải lớn, công suất lớn cũng như chất lượng đặc tính cơ tốt. Không cần phải có cổ góp và vì thế có độ bền về cơ học tốt hơn và không cần thiết phải bảo dưỡng định kỳ như động cơ điện 1 chiều.

*b. Dẫn động bằng thuỷ lực và khí nén*

Đối với các loại động cơ thủy - khí và xi lanh - piston thủy - khí, ưu điểm của hệ dẫn động này là có thể tạo ra chuyển động quay và chuyển động thẳng tuỳ theo mục đích và yêu cầu sử dụng. Vì thế chúng có thể sử dụng một cách trực tiếp mà không cần thiết phải qua các bộ chuyển đổi trung gian bằng các cơ cấu cơ khí khác. Tuy nhiên do khác nhau về các đại lượng vật lý của thông số điều khiển với hệ điều khiển nên cần có các bộ chuyển đổi và vì thế có thể gây ra các sai số.

** Dẫn động thuỷ lực.*

Dẫn động thuỷ lực được sử dụng trong trường hợp khi có yêu cầu tải trọng tác dụng lớn, chuyển động chậm và cần thiết phải được điều khiển chính xác. Ưu điểm đáng kể nhất của loại dẫn động này là có thể tạo được một công suất rất lớn trong khi kích thước khuôn khổ và kết cấu là nhỏ gọn nhất. Tuy nhiên nó có nhược điểm là các phần tử trong hệ thống cần phải có độ chính xác khi chế tạo và lắp ráp

rất cao, giá thành đắt. Nhạy cảm với sự thay đổi của nhiệt độ, cần phải bố trí hệ thống dầu hồi nên chỉ sử dụng độc lập trong từng máy riêng rẽ.

Ngày nay, có thể kết hợp giữa dẫn động thuỷ lực với các dẫn động điện như các loại động cơ bước - điện thuỷ lực để khắc phục các nhược điểm trên.

** Dẫn động khí nén.*

Dẫn động khí nén thường được dùng trong các trường hợp đóng mở các van, các dẫn động không yêu cầu về độ chính xác điều khiển tốc độ và vị trí. Thường hay sử dụng trong các hệ thống điều khiển logic như gá lắp, kẹp chặt, lắp ghép các chi tiết, dịch chuyển phôi liệu, nâng hạ hoặc quay bàn...

Ưu điểm của loại dẫn động này là tận dụng các nguồn khí nén có sẵn trong các nhà máy, phân xưởngvà khí thải sau khi sử dụng xong có thể thải ra môi trường mà không cần hệ thống ống dẫn phức tạp.

Nhược điểm: Do đàn tính của khí nén và sự nhạy cảm với nhiệt độ nên chỉ dùng với trường hợp tác đọng 2 vị trí như đóng, mở.

## 1.6.Khối điều khiển

Khối điều khiển là nơi thực hiện các nhiệm vụ như nhận thông tin từ các vật mang tin và chuyển nó thành các lệnh để điều khiển khối chấp hành, đồng thời giám sát sự hoạt động của khối chấp hành thông qua các cảm biến để điều chỉnh hệ thống hoạt động ổn định một cách kịp thời và chuẩn xác.

### *1.6.1. Hệ điều khiển cứng*

Với hệ điều khiển này, việc điều khiển quá trình hoạt động tự động được thực hiện theo công nghệ lập tuyến. Ví dụ như trong các mạch điều khiển logic bằng điện và khí nén hay trong các hệ điều khiển cứng như với các câm lắp trên trục phân phối trong các máy tự động điều khiển bằng cam.

Đặc điểm của hệ điều khiển cứng này là chỉ hoạt động theo các chương trình mà người thiết kế đã lắp đặt. Khi cần thay đổi một số hoạt động của hệ thống thì phải điều chỉnh hoặc thay thế một số phần tử, thậm chí có thể phải thay thế toàn bộ hệ thống khác.

Phạm vi ứng dụng của phương pháp điều khiển này trong những trường hợp hoạt động  theo từng chức năng mà tính chất hoạt động của nó là đã tối ưu, ổn định như các trường hợp gắp phôi, kẹp và tháo chi tiết, các chương trình hoạt động của các máy dập, máy cắt đột, các hệ thống vận chuyển nguyên liệu, sản phẩm, đóng gói, đo kiểm và phân loại sản phẩm...

*1.6.3. Hệ điều khiển lập trình được*

Ngày nay, với sự phát triển của các lĩnh vực vi điện tử và tin học đã cho phép người ta đưa vào các bộ vi xữ lý, các bộ tính toán số học, logíc và nội suy vào trong các khối điều khiển nhằm mục đích điều khiển quá trình hoạt động của các thiết bị một cách linh hoạt bằng việc thay đổi các chương trình hoạt động của các máy móc, thiết bịtheo công nghệ yêu cầu nhờ sự thay đổi chương trình trên cơ sổ các vật chất đã có.

Ngày nay, các hệ điều khiển khả lập trình đã được đưa vào và sử dụng khá rộng rãi như các bộ điều khiển khả trình PLC sử dụng trong công nghiệp, các hệ thống điều khiển số trên các máy công cụ CNC, các trung tâm gia công, Robot và các hệ thống DNC, CIM.

## 1.7. Van dẫn động

Mục đích của các van là sử dụng tín hiệu điều khiển từ khối điều khiển có công suất rất nhỏ để thực hiện các chức năng đóng mở các dẫn động có công suất lớn. Đặc điểm của các van này có nhiều kiểu khác nhăunhng yêu cầu cơ bản là phải có tác động nhanhvà nhạy với các tín hiệu điều khiển. Tuỳ theo hệ dẫn động mà sử dụng các các loại van tương ứng.

Đối với dẫn động điện, nó thường là Rơle, Công tắc tơ hoặc khởi động từ.

Đối với dẫn động thuỷ khí thì thường sử dụng các loại van Solenoid, van trượt hoặc van quay điều khiển.

## 1.8. Cảm biến

Mục đích của cảm biến là giám sátccs hoạt động của các dẫn động hoặc các cơ cấu chấp hành để nhận các đại lượng vật lý thựccủa quá trình hoạt động và biến đổi nó thành các dại lượng điều khiển (đại lượng điện) để cung cấp cho khối điều khiển nhằm điều khiển hệ thống một cách ổn định và chính xác nhất.

Tuỳ thuộc vào các quá trình công nghệ mà lựa chọn các loại cảm biến cho thích hợp với các đại lượng cần xác định. Cảm biến có rất nhiều loại như cảm biến áp suất; cảm biến tốc độ; cảm biến lực và khối lượng; cảm biến vị trí, vận tốc và gia tốc; cảm biên hồng ngoại; cảm biến từ; cảm biến lưu lượng; cảm biến nhiệt độ.... Tuy nhiên trong kỹ thuật, thông thường người ta sử dụng các loại cảm biến số và tương tự. Trong trường hợp sử dụng cảm biến tương tự thì cần thiết phải có thêm bộ chuyển đổi AD (tương tự sang số).

## II: CÁC HỆ THỐNG SỐ ĐẾM VÀ MÃ

Hệ thống số đếm (*Number system*) là một tập hợp có thứ tự các chử số để biểu diễn một số bất kỳ. Trong thực tế ta thường gặp hệ thống đếm thập phân với cơ số của hệ thống đếm này là *10*, ngoài ra ta còn gặp các hệ thống số đếm khác như hệ đếm nhị phân (*Binary*) với cơ số đếm là *2*, hệ đếm cơ số *8* (*Octal*) và hệ đếm cơ số *16* (*Hexadecimal*). Các hệ đếm này rất có ý nghĩa trong các hệ thống số như máy tính, các bộ vi xử lý...

Cá hệ thống số hoạt động với hệ đếm nhị phân với 2 chử số *0* và *1* và mỗi một chử số là *1 bit* (*binary digital*). Một nhóm *8 bit* tạo thành *1 byte* và nhóm *4 bit* tạo nên *1 nibble*. Vì rằng hầu hết hệ thống số hiện nay (*Digital system*) chỉ hiểu các con số *0* và *1* nên bất kỳ thông tin nào , mà thường là dưới dạng chử số, chử cái hay các ký hiệu đặc biệt ... phải được biến đổi thành dạng số nhị phân trước khi nó có thể được xử lý bằng các mạch số.

Thí dụ: Một số 11011101 là một *byte*

Trong đó, 1101 là một *nibble*.

Nói chung, trong bất kỳ một hệ thống số đếm nào, thì một tập có thứ tự các ký hiệu - *gọi là chử số* - cùng với các luật được định nghĩa đều được dùng để thực hiện các phép toán như cộng, trừ, nhân, chia... Một tập hợp các số đó tạo ra một số gồm 2 phần là **phần nguyên** và phần **thập phân** và chúng được ngăn cách nhau bằng dấu phẩy.

$(N)_b = d_{n-1}\, d_{n-2} ... d_1\, d_0, d_{-1}\, d_{-2} ... d_{-m}$

Trong đó:

N là một số.

b là cơ số của hệ thống số đếm

n là số chử số có trong phần nguyên

m là số chử số có trong phần thập phân

$d_{n-1}$ là chử số có nghĩa nhất MSB (*Most sinificant bit*)

$d_{-m}$ là chử số ít nghĩa nhất LSB (*Least sinificant bit*)

Thí dụ: $(1101,10101)_2$ là hệ đếm cơ số 2, số 1 phía cuối cùng bên trái là tương ứng với $d_{n-1}$ (n =4) là MSB và số 1 phía cuối cùng bên phải là ứng với $d_{-m}$ là LSB (m=5).

Các hệ thống số đếm thông thường được sử dụng biểu diễn ở bảng 1-1:

*Bảng 2-1*

| Hệ đếm | Ký hiệu | Cơ số | Những chử số và ký tự được sử dụng | Trọng lượng được gán cho vị trí thứ i | Ví dụ |
|---|---|---|---|---|---|
| Nhị phân | % | 2 | 0 1 | $2^i$ | 10101,00 |
| Cơ số 8 (octal) | @ | 8 | 0 1 2 3 4 5 6 7 | $8^i$ | 4235,37 |
| Thập phân | # | 10 | 0123456789 | $10^i$ | 3987,352 |
| Cơ số 16 (hexa) | $ | 16 | 0123456789ABCDEF | $16^i$ | 3FB9,C6 |

Như đã nói ở trên, để có thể xử lý được các thông tin bằng các mạch số điện tử cần phải biến đổi các ký tự bằng các chử, các dấu, các ký hiệu đặc biệt và các số thành các ký tự ở dạng số nhị phân. Đây thực chất là một quá trình **mã hóa** (*encoder*). Như vậy, mã hóa thông tin là xác định các chử cái và chử số, các dấu bằng cách sử dụng các ký hiệu khác.

Trong một hệ thống số, các mã khác nhau có thể được dùng cho các hoạt động khác nhau và vì lý do cần thiết, người ta phải chuyển đổi từ mã này sang mã khác bằng các mạch *chuyển mã* để thực hiện mục đích đó.

Mã trong thực tế còn được sử dụng vì lý do an toàn, có nghĩa là các thông tin mà người ta chuyển đi cần thiết người khác không thể đọc được. Cũng vì lý do này mà có rất nhiều mã ra đời mà người ta còn gọi là *"mật mã"*. Trong phạm vi nghiên cứu về điều khiển số, chúng ta chỉ quan tâm đến một số mã chủ yếu đã nêu trên.

## 2.1. Hệ đếm nhị phân

Hệ thống đếm số với cơ số *2* gọi là hệ đếm nhị phân. Chỉ có 2 loại ký hiệu được dùng để biểu diễn tất cả các số ở trong hệ thống đếm là *0* và *1*. Mỗi giá trị của chúng trong số được gọi là *1 bít*. Đây là hệ thống đếm có cơ số nhỏ nhất và nó là hệ thống số đếm vị trí, tức là tất cả các vị trí được gán một trọng lượng xác định. Thí dụ: $(11010,101)_2$ có thể được viết lại trong hệ thập phân là: $1.2^4+1.2^3+0.2^2+1.2^1+0.2^0+1.2^{-1}+0.2^{-2}+1.2^{-3}=26,625$

Từ cơ sở trên, ta có thể chuyển đổi 1 số *nhị phân* thành một số *thập phân* tương đương và ngược lại.

Thí dụ: Hãy chuyển một số sau đây $(25,675)_{10}$ ở hệ thập phân sang hệ nhị phân

Để tiến hành thủ tục chuyển số thập phân trên sang số nhị phân, ta phân chúng làm thành 2 phần - *phần nguyên (PN) và phần thập phân (PL)*- . Đối với phần nguyên, ta chia liên tiếp cho 2 và giữ lại các số dư. PN của số nhị phân là dãy số dư được đọc từ dưới lên:

Ví dụ:

| | **Thương** | **Dư** |
|---|---|---|
| 25:2 | 12 | 1 |
| 12:2 | 6 | 0 |
| 6:2 | 3 | 0 |
| 3:2 | 1 | 1 |
| 1:2 | 0 | 1 |

Vậy PN của số nhị phân là:11001

Đối với phần thập phân, ta nhân liên tiếp với 2 và giữ lại các số nguyên được sinh ra, kết quả của PL là các số nguyên được đọc từ trái sang phải:

| 0,675 | 0,35 | 0,70 | 0,40 | 0,80 | 0,60 |
|---|---|---|---|---|---|
| x 2 | x 2 | x 2 | x 2 | x 2 | x 2 |
| 1,35 | 0,70 | 1,40 | 0,80 | 1,60 | 1,20 |
| 1 | 0 | 1 | 0 | 1 | 1 |

Vậy PL của số nhị phân là: 101011

Kết quả cuối cùng: $(25,675)_{10} = (11001,101011)_2$

*Chú ý: Việc chuyển đổi từ số thập phân sang số nhị phân không phải là luôn luôn đúng, ta có thể chấp nhận một giá trị gần đúng bằng cách kết thúc quá trình nhân 2 ở một giá trị mong muốn sao cho sai số do chuyển đổi mã có thể chấp nhận được.*

Cũng như các phép toán trong hệ thập phân, số nhị phân cũng có các phép toán số học cơ bản như cộng, trừ, nhân, chia. Việc thực hiện tính toán nó được thực hiện theo các nguyên tắc sau:

*2.1.1. Phép cộng nhị phân:*

Có thể biểu diễn luật cộng nhị phân theo bảng 2-2

Kết quả trong bảng 2.2 là biểu diễn phép cộng 2 số nhị phân sau:

$$\begin{array}{r} 1100 \\ +\ 1010 \\ \hline 10110 \end{array}$$

Ví dụ: Hãy cộng các số nhị phân sau:

| | |
|---|---|
| 1010011 | 100101,101 |
| +1101101 | +110001,001 |
| 11000000 | 1010110,110 |

*Bảng 2-2*

| Số hạng 1 | Số hạng 2 | Tổng | Nhớ | Kết quả |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| | | | | 1 |

*2.1.2. Phép trừ nhị phân*

Luật trừ nhị phân được trình bày ở bảng 2-3

*Bảng 2-3*

| Số bị trừ | Số trừ | Hiệu số | Mượn | Kết quả |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |

Kết quả trên là của phép trừ sau:

| |
|---|
| 1100 |
| -1010 |
| 0010 |

Ví dụ: Hãy trừ một số số nhị phân sau:

| | |
|---|---|
| 11011 | 1101001,101 |
| -10110 | -110010,001 |
| 00101 | 0110111,100 |

Tuy nhiên,trong phép trừ không phải là luôn luôn thực hiện được, ví dụ khi phép trừ với số bị trừ nhỏ hơn số trừ thì kết quả sẽ khác đi do nó phải mang dấu âm.

Vì thế, để đơn giản trong khi thực hiên phép trừ, người ta xem phép trừ là kết quả của một phép cộng của số bị trừ với số trừ mang dấu âm, khi đó người ta chỉ thực hiện việc cộng bình thường và xác định kết quả của nó dựa vào các nhận xét trên cơ sổ của kết quả của phép cộng đó. Điều này trong thực tế lại càng thuận lợi hơn vì các mạch số được thiết kế chỉ thực hiện các phép cộng nhị phân mà không cần phảithiết kế mạch trừ. Để thuận tiện cho việc sử dụng, người ta đưa ra các định nghĩa bù một và bù hai.

*a. Bù một*

Trong số nhị phân, nếu thế bít *1* bằng bít *0* và ngược lại, ta sẽ có một số nhị phân khác được gọi là *bù một* của số nhị phân thứ nhất.

Ví dụ: 01100110 là *bù một* của số nhị phân 10011001.

*b. Bù hai*

Nếu cộng thêm *1* vào *bù một* của một số nhị phân thì ta nhận được một số nhị phân khác được gọi là *bù hai* của nó.

Ví dụ: 01100110 *là bù một* của số nhị phân 10011001 thì *bù hai* của nó chính là 01100111.

Đối với 01001110 thì *bù một* là : 10110001 và *bù hai* là : 10110010

Đối với 00110101 thì *bù một* là : 11001010 và *bù hai* là : 11001011

Từ đó, ta có kết luận :

- Nếu *bit* có giá trị nhỏ nhất LBS là số *0* thì *bù hai* nhận được bằng cách đổi mỗi *bit 0* thành *1* và mỗi bit *1* thành *0* ngoại trừ LBS và *bit* sát với LBS.
- Nếu LBS là số *1* thì *bù hai* nhận được bằng cách đổi mỗi *bit 0* thành *1* và *bit 1* thành *0* ngoại trừ LBS (không thay đổi).

Trên cơ sổ đó, ta có thể sử dụng quy tắc sau đây để tìm *bù hai* của số nhị phân: *Kiểm tra các bít từ LBS đến MBS (từ phải sang trái), viết các bit như nguyên dạng của chúng nếu là bít 0 cho đến khi gặp bit 1 đầu tiên thì lấy bù một của tất cả các bit còn lại trừ bít 1 đó.*

Áp dụng : Tìm *bù hai* của các số sau :

←-----

01100100 → bù 2 : 10011100

←-----

10010010 → bù 2 : 01101110

←-----

11011000 →bù 2 : 00101000

←-----

01100111 → bù 2 : 10011001

Từ đó, ta cũng có kết luận : *Bù hai* của một số chính là chính số đó.

*c. Phép trừ sử dụng bù hai*

Như trên đã nói, phép trừ nhị phân có thể được thực hiện bằng cách cộng số bị trừ với *bù hai* của số trừ. Nếu có *1 nhớ* cuối cùng là *bit 1* được sinh ra thì hủy bỏ giá trị nhớ đó và kết quả là những *bit* còn lại và đó là số dương (số bị trừ lớn hơn số trừ). Nếu như *1 nhớ* cuối cùng là *bit 0* (không có nhớ) thì kết quả là số âm (số bị trừ nhỏ hơn số trừ) và hiệu của nó là *bù hai* của kết quả phép cộng ở trên.

Ví dụ : Thực hiện phép trừ nhị phân sử dụng phương pháp cộng với *bù hai* của số trừ:

Ví dụ: 7-5 =2 được biểu diễn theo phương pháp trên

0111 + 1011 =**1**0010 là số dương và có giá trị là 0010 (2 trong hệ thập phân), trong đó -5 được biểu diễn bằng *bù hai* của 0101 là 1011(bỏ nhớ cuối cùng là bit **1** và kết quả : 0010 = + 2trong hệ thập phân).

Ví dụ: 5-7=-2 được biểu diễn theo phương pháp trên

0101 + 1001 = 1110 là một số âm và hiệu của nó là 0010 (bù hai của 1110) chính là 2 trong hệ thập phân, trong đó -7 được biểu diễn bằng *bù hai* của 0111 là 1001. Ở đây, nhớ cuối cùng là *bit 0* hay không có nhớ nên kết quả là số âm và ở dưới dạng *bù hai* tức là *bù hai* của 1110 là 0010 tức là (- 2).

*2.1.3. Phép nhân nhị phân :*

Tương tự như phép nhân thập phân. Ví dụ : nhân 1101 với 1010.

Số bị nhân 1101(*muntifplicant*)

Số nhân 1010 (*muntifplier*)

0000 hàng thứ nhất

1101 hàng thứ hai

0000 hàng thứ ba

1101

**10000010**

*2.1.4. Phép chia nhị phân :*

Tương tự như phép chia số thập phân :

Ví dụ : chia 110110 cho 1101

Số bị chia : 110110 | 1101 (số chia)

- 1101 **100,0101** (kết quả)

00001

10

100 (thêm *bit 0* và phía kết quả là dấu phẩy)

1000

10000

-1101

00110 tiếp tục thêm *bít 0* và lại tực hiện tiếp...

Ví dụ : Chia 110110 cho 1001 có kết quả 110. Kết quả của phép chia có thể là chia hết hoặc không chia hết, tuy nhiên theo mức độ chính xác yêu cầu mà ta kết thúc việc thực hiện.

## 2.2. Hệ dếm cơ số 8 *(OCTAL)*

Nó được sử dụng rộng rãi trong các máy tính và máy vi tính để nhập dữ liệu. Mỗi chữ số cơ số *8* là tổ hợp của 3 chữ số nhị phân. Do vậy, tập các số nhị phân *3 bit* có thể được biểu diễn bằng các chữ số cơ số *8* rất thuận lợi khi nhập dữ liệu vào máy tính. Do vì các mạch số chỉ có thể xử lý các tín hiệu *nhị phân 0 và 1* nên cơ số *8* phải được tái tạo lại thành dạng *nhị phân* bằng các mạch chuyển đổi mã.

Các ký hiệu được dùng trong hệ đếm cơ số *8* là : 01234567. Nó cũng bao gồm 2 phần là **phần nguyên** và phần **thập phân** và được ngăn cách nhau bằng dấu phẩy.

Ví dụ : $(6327,4051)_8$ hoặc @ 6327,4051.

Cũng như hệ *nhị phân*, hệ đếm *Octal* cũng có thể chuyển đổi sang hệ đếm *thập phân* và ngược lại, nhưng chỉ khác là thay vì số *2* trong hệ nhị phân bằng số *8* trong hệ *Octal.*

$(6327,4051)_8 = 6.8^3 + 3.8^2 + 2.8^1 + 7.8^0 + 4.8^{-1} + 0.8^{-2} + 5.8^{-3} + 1.8^{-4} = (3287,5100098)_{10}$

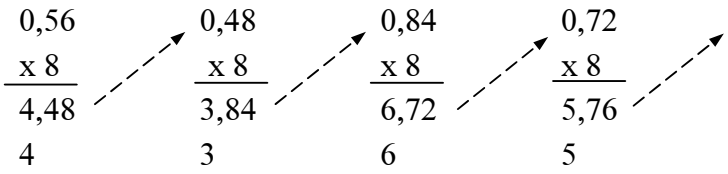
Thí dụ : chuyển đổi $(658,56)_{10}$ sang hệ cơ số *8*.

Phần nguyên :

| | **Thương** | **Dư** |
|---|---|---|
| 658 : 8 | 82 | 2 |
| 82 : 8 | 41 | 0 |
| 41 : 8 | 5 | 1 |
| 5 : 8 | 0 | 5 |

Kết quả phần nguyên: 5102

Phần thập phân :




Phần thập phân : 0,4365

Kết quả cuối cùng : $(5102,4365)_8$.

Cũng như trong hệ nhị phân, *việc chuyển đổi cho phần thập phân có thể là không luôn luôn chính xác mà có thể xác định giới hạn kết thúc theo một yêu cầu cho phép về sai số.*

*2.2.1. Chuyển đổi hệ đếm cơ số 8 sang hệ nhị phân và ngược lại*

Như đã phân tích ở trên, hệ đếm cơ số *8* có thể chuyển sang hệ nhị phân tương đương bằng việc thay thế mỗi chữ số trong hệ cơ số *8* bằng *3 bit* hệ nhị phân.

Thí dụ :

$(475)_8 = (100\ 111\ 101)_2$ và cũng có thể viết @ 475 =% 100 111 101

@ 574,321 = % 101 111 100, 011 010 001

Và ngược lại, chuyển từ hệ nhị phân sang hệ đếm *Octal* được thực hiện như sau: gộp các nhóm *3 bit* bắt đầu từ LSB (ít ý nghĩa nhất) hay sát ngay với bên trái dấu phẩy và chuyển dần về phía MSB (bit nhiều ý nghĩa nhất). Với phần *thập phân*, bắt đầu từ *bit* sát ngay với bên phải dấu phẩy và chuyển dần sang phải.

Thí dụ :

% 10011101 = @ 235

% 10110,10010 = @ 26,44

*2.2.2. Các phép tính số học :*

Các luật số học cơ số *8* cũng tương tự như luật số học *nhị phân* và *thập phân*.Tuy nhiên, các phép tính số học trong hệ cơ số *8* ít mang ý nghĩa thực tế sử dụng nên ta không cần phải xem xét kỹ chúng.

## 2.3. Hệ đếm cơ số 16 *(HEXADECIMAL)*

Hệ đếm này được sử dụng rất nhiều trong máy tính và các bộ vi xử lý. Có *16* tổ hợp của số nhị phân *4 bit* và tập hợp các số nhị phân *4 bit* có thể nhập vào máy tính dưới dạng các chữ số *Hexa*. Số *Hexa* được biến đổi thành dạng nhị phân trước

khi chúng được xử lý bởi các mạch số. Cơ số của hệ đếm này là *16* bao gồm **0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F**. Hệ đếm này còn gọi là hệ đếm ký tự. Bảng 2-4 chỉ ra sự tương đương giữa các số *thập phân - Hexa và Binary*.

Các ký hiệu được dùng trong hệ đếm cơ số *16* là : **0123456789ABCDEF**. Nó cũng bao gồm 2 phần là **phần nguyên** và phần **thập phân** và được ngăn cách nhau bằng dấu phẩy.

Ví dụ : $(3C8,F1)_{16}$ hoặc $ 3C8,F1

Cũng như hệ *nhị phân*, hệ đếm *Hexa* cũng có thể chuyển đổi sang hệ đếm *thập phân* và ngược lại, nhưng chỉ khác là thay vì số *2* trong hệ nhị phân bằng số *16* trong hệ *Hexa.*

Ví dụ: $(3A,2F)_{16} = 3.16^1 + 10.16^0 + 2.16^{-1} + 15.16^{-2} = (58,1836)_{10}$.

*Bảng 2-4*

| Hệ thập phân<br># | Hệ Hexa<br>$ | Hệ Nhị phân<br>% |
|---|---|---|
| 0 | 0 | 0000 |
| 1 | 1 | 0001 |
| 2 | 2 | 0010 |
| 3 | 3 | 0011 |
| 4 | 4 | 0100 |
| 5 | 5 | 0101 |
| 6 | 6 | 0110 |
| 7 | 7 | 0111 |
| 8 | 8 | 1000 |
| 9 | 9 | 1001 |
| 10 | A | 1010 |
| 11 | B | 1011 |
| 12 | C | 1100 |
| 13 | D | 1101 |
| 14 | E | 1110 |
| 15 | F | 1111 |

Ví dụ chuyển đổi số #675,625 $(675,625)_{10}$ thành hệ *Hexa.*

Phần nguyên:

| | Thương | Dư |
|---|---|---|
| 675 : 16 | 42 | 3 |
| 42 : 16 | 2 | 10 |
| 2 : 16 | 0 | 2 |

Phần nguyên là: 2A3

Phần thập phân:

0,625

x 16

10,00

10 Phần thập phân là: 0,A

Kết quả cuối cùng là: $(675,625)_{10}$ = $ 2A3,A

*2.3.1. Phép chuyển đổi từ Hexa sang Binary và ngược lại :*

Như đã nói ở trên, các số *Hexa* có thể chuyển đổi thành dạng *Binary* bằng việc thay thế mỗi chữ số *Hexa* bằng *4 bit Binary* tương đương.

Ví dụ : $(2F9A)_{16}$ = $(001011111010)_2$ hoặc $2F9A = % 0010 1111 1001 1010

Ví dụ : % 1010101110110110 = $ 33DB

$(0,01101101001110)_2$ = % 0, 0110 1101 0011 1000 = $(0,6D38)_{16}$.

*2.3.2. Các phép tính số học :*

Cũng tương tự như hệ *Octal*, các phép tính số học hệ đếm *Hexa* cũng ít được sử dụng trong thực tế vì thế chúng ta không khảo sát kỹ về nó.

## 2.4. Các số có dấu

Đối với mạch số chỉ có 2 ký hiệu *1* và *0* thì việc biểu thị giá trị dương và âm cần phải được giải quyết bằng việc thêm *1 bit* vào bên trái nhất và được xem như *1 bit* có giá trị lớn nhất (MSB), với *bit 0* **ký hiệu số dương** và *bit 1* **biểu diễn số âm**.

Thí dụ: **0**1000100 là 1 số dương có giá trị 68 (bit **0**)

**1**1000100 là 1 số âm có giá trị 68 (bit **1**)

Có 3 loại số nhị phân có dấu được biểu diễn như bảng 2.5.

*Bảng 2.5* :

| Phương pháp | Dương nhất | Âm nhất |
|---|---|---|
| Mã thuận | $+ (2^{n-1}-1)$ | $- (2^{n-1}-1)$ |
| Bù một | $+ (2^{n-1}-1)$ | $- (2^{n-1}-1)$ |

Bù hai $+(2^{n-1}-1)$ $-(2^{n-1}-1)$

Trong *mã thuận* : *Bit* cao nhất được dùng để biểu diễn **dấu** và các *bit* còn lại để biểu diễn giá trị của số.

Đối với phương pháp *bù một, bù hai* : *Bit* cao nhất được dùng để biểu diễn **dấu** (*0 là số dương và 1 là số âm*). Đối với số dương, việc biểu diễn tương tự như mã thuận. Riêng số *âm* thì trước hết cần phải xét *độ lớn* rồi xem xét là chúng là *bù một hay bù hai.*

Ví dụ : 0101 biểu diễn + 5 ; còn 1010 và 1011 biểu diễn - 5 tương ứng trong phương pháp bù một và bù hai.

Thông thường, phương pháp bù hai được sử dụng rộng rãi hơn vì sự tiện lợi khi thiết kế mạch số để thay thế phép trừ nhị phân bằng phép cộng với bù hai (ở phần trên đã nói).

### 2.5. Mã

Máy tính, các mạch số được sử dụng làm việc ở dạng *nhị phân*. Vì thế việc thao tác với các con số, các chữ cái và các ký tự đặc biệt khác phải được tái tạo thành khuôn dạng nhị phân. Đây gọi là quá trình **mã hóa**. Hiện nay có nhiều mã số và chúng được sử dụng để phục vụ các mục đích khác nhau. Bảng 2-6 biểu diễn 1 số mã thường được sử dụng nhất :

*Bảng 2-6*

| Thập phân<br># | Nhị phân<br>% | BCD | Excess-3 | Gray | Hexa<br>$ | Octal<br>@ |
|---|---|---|---|---|---|---|
| | $B_1B_2B_3B_4$ | DCBA | $E_3E_2E_1E_0$ | $G_3G_2G_1G_0$ | | |
| 0 | 0000 | 0000 | 0011 | 0000 | 0 | 0 |
| 1 | 0001 | 0001 | 0100 | 0001 | 1 | 1 |
| 2 | 0010 | 0010 | 0101 | 0011 | 2 | 2 |
| 3 | 0011 | 0011 | 0110 | 0010 | 3 | 3 |
| 4 | 0100 | 0100 | 0111 | 0110 | 4 | 4 |
| 5 | 0101 | 0101 | 1000 | 0111 | 5 | 5 |
| 6 | 0110 | 0110 | 1001 | 0101 | 6 | 6 |
| 7 | 0111 | 0111 | 1010 | 0100 | 7 | 7 |
| 8 | 1000 | 1000 | 1011 | 1100 | 8 | |
| 9 | 1001 | 1001 | 1100 | 1101 | 9 | |
| 10 | 1010 | | | 1111 | A | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 11 | 1011 | | | 1110 | B | |
| 12 | 1100 | | | 1010 | C | |
| 13 | 1101 | | | 1011 | D | |
| 14 | 1110 | | | 1001 | E | |
| 15 | 1111 | | | 1000 | F | |

*2.5.1. Mã BCD: (Binary Codel Decimal)*

Đây là mã nhị phân *tự nhiên* để biểu diễn các số thập phân 0 đến 9 và mỗi số thập phân tương đương *4 bit*. Nó còn được gọi là mã **8 - 4 - 2 - 1** hay là mã **nhị thập phân**. Mã BCD không phải là một hệ đếm số riêng mà là 1 mã quy định (quy ước) để viết cho đơn giản. Ở hệ này chỉ dùng *9* tổ hợp từ 0000 đến 1001, còn các tổ hợp 1010 đến 1111 thì không dùng (với *4 bit* ta có *16 tổ hợp* số nhị phân nhưng ta chỉ dùng 10 tổ hợp mà thôi).

Mã BCD phải dùng nhiều bit hơn mã nhị phân trực tiếp. Tuy vậy ưu điểm của BCD là dễ chuyển đổi sang hệ thập phân vì với mỗi tổ hợp nhị phân *4 bit* vẫn đứng đúng hàng đơn vị, hàng chục, hàng trăm ... Vì thế ta thấy rất thuận lợi là vừa mang tính rõ ràng của hệ *thập phân*, vừa mang tính mật độ cao của hệ *nhị phân*. Ta gọi là hệ **nhị - thập phân**.

*2.5.2. Mã Excess - 3*

Mã này nhận được bằng cách cộng thêm *3* vào mỗi số mã của mã nhị phân tự nhiên. Thí dụ 1000 của mã *Excess-3* biểu diễn số *5* của thập phân trong khi mã nhị phân tự nhiên biểu diễn số *8* của hệ thập phân.

*2.5.3. Mã Gray*

Chỉ cần thay đổi *1 bit* trong khi biểu diễn các số là sẽ thay đổi giữa *2 số liên tiếp*. Thí dụ 0111 biểu diễn số *5* và 0101 là số *6* trong mã *Gray*. Chính nhờ tính chất này mà người ta sử dụng mã *Gray* vào trong bộ mã đo lường hay kiểm tra các sự chuyển động hoặc vị trí khi dịch chuyển, đặc biệt là trên các máy điều khiển theo chương trình số.

*2.5.4 Mã ký tự*

Trong nhiều trường hợp, hệ thống số được dùng để thao tác dữ liệu có thể ở dạng số, chữ cái hay các ký hiệu đặc biệt. Do vậy, một mã *nhị phân* cho các chữ cái là rất cần thiết. Nếu ta sử dụng một mã *nhị phân n bit* thì sẽ có $2^n$ tổ hợp khác nhau có thể biểu diễn được. Ta có số ký tự cần biểu diễn là bao gồm số chữ số từ *0* đến *9*

là *10 ký tự*; *26* chữ cái từ *A* đến *Z* bao gồm cả chữ hoa, chữ thường; các ký hiệu khác như *@, #, $, %...* và các dấu của các phép tính như ∫ , +, -, /.... Tổng số ký tự cần thiết được biểu diễn là *lớn hơn 70*. Như vậy ta cần phải sử dụng mã nhị phân có số *bít* ≥ 6 (vì $2^6$ = 64 tổ hợp) mới đủ để biểu diễn. Hiện nay người ta chọn mã nhị phân *8 bit* để biểu diễn*( n = 8)* mà trong đó có *7 bit* ( có *128 tổ hợp* khác nhau) dùng để biểu diễn các ký tự và một *bit* dùng để kiểm tra.

Thông thường hiện nay người ta sử dụng các mã sau đây:

- Mã trao đổi thông tin BDC mở rộng : EBCDIC
- Mã ASCII (Mã trao đổi thông tin của hội tiêu chuẩn Mỹ - *American Standard Code for Information Interchange*)

*2.5.5. Các mã lỗi*

Khi thông tin số dưới dạng nhị phân được truyền từ mạch hay hệ thống này sang hệ thống khác thì có thể có lỗi phát sinh (tức là có 1 tín hiệu *0* nào đó trở thành *1* hay ngược lại). Với hệ thống số phức tạp thì có hàng triệu *bit* được xử lý trong giây với yêu cầu tính toàn vẹn dữ liệu cao và nếu có sai phạm thì cần phải được phát hiện để xử lý. Người ta dùng phương pháp cộng thêm *1 bit* vào *bit dữ liệu* gọi là bit *parity*. Nó cho phép phát hiện 1 lỗi đơn trong sự truyền dữ liệu, thông thường nó được dùng để kiểm tra tính **chẵn lẽ** của các bít *1* hay bit *0.* Thí dụ trong mã ASCII có *7 bit* ký tự và thêm vào *1 bit* để làm cho số bit *1 là số chẵn hay số lẽ* theo từng hàng nhằm đảm bảo rằng khi máy đọc sẽ đảm bảo tính đúng đắn và chính xác của các thông tin đưa vào.

Thí dụ : 1000011 là biểu diễn ký tự *C* trong mã ASCII có *3 bit 1(số lẽ).* Để đảm bảo yêu cầu chính xác, người ta thêm vao *1 bit 1* là *bit thứ 8* để cho số *bit 1 là 4* (*chẵn).* Như vậy khi đọc máy sẽ xác định tính đúng đắn của dữ liệu là 11000011, chú ý rằng *bit 1* thêm vào này là chỉ dùng để kiểm tra tính đúng đắn của nó mà không tham gia vào mã ký tự.

Thí dụ ký tự *G* trong mã ASCII là 1000111có số *bít 1 là 4 (chẵn*) nên *bit* kiểm tra là *bít thứ 8 có giá trị bit 0*. Vậy khi máy đọc là 01000111.

# III. TOÁN TỬ LOGIC
# VÀ MỘT SỐ KHÁI NIỆM CƠ BẢN CỦA ĐIỆN TỬ SỐ

## 3.1. Toán tử Logic

Các biến số *Boole* có thể được xử lý bằng các phép *toán tử logic* cơ bản KHÔNG ; VÀ ; HOẶC ; KHÔNG HOẶC; KHÔNG VÀ; HOẶC CÓ LOẠI TRỪ . 3 toán tử đầu làm thành 1 hệ thống *logic* độc lập và hoàn chỉnh. Còn các toán tử *không hoặc*, *không và* và *hoặc có loại trừ* là các hàm phụ thuộc.

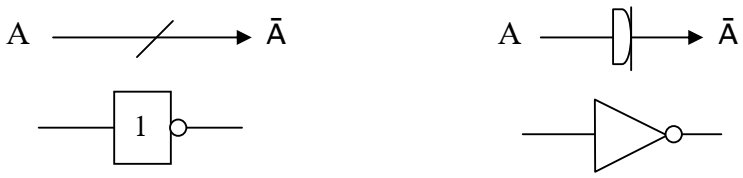
*3.1.1. Toán tử KHÔNG (NOT) : Phủ định.*

A = 1 thì *không A :* $\bar{A} = 0$

A = 0 thì *không A :* $\bar{A} = 1$

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

Ký hiệu : theo tiêu chuẩn của Châu Âu và của Mỹ




Nếu gọi đầu vào A và đầu ra Y, ta có :

| A | Y |
|---|---|
| 0 | 1 |
| 1 | 0 |

*3.1.2. Toán tử VÀ (AND)*

A *AND* B *AND* C = A . B . C = Y.

Y = 1 nếu và chỉ nếu đồng thời A = 1, B = 1 và C = 1

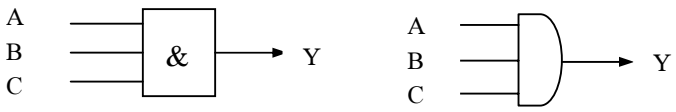
Y = 0 nếu hoặc chỉ A =0 hoặc B=0 hoặc C=0; hoặc A=B=C = 0.

Ta có bảng trạng thái sau:

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

| | | | |
|---|---|---|---|
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

Toán tử VÀ còn có thể viết theo cách toán tập hợp: $A \wedge B \wedge C$.
Ký hiệu:

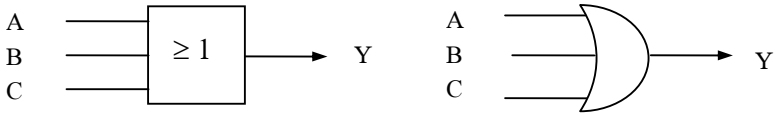



*3.I.3. Toán tử HOẶC (OR)*

A *OR* B *OR* C = A+B+C =Y
Y = 0 nếu khi và chỉ khi đồng thời cả A = 0, B = 0 và C = 0.
Ta có bảng trạng thái sau:

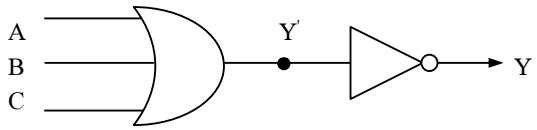
| A | B | C | Y |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

Có thể biểu diễn theo cách toán tập hợp A V B V C = Y
Ký hiệu:




*3.I.4. Toán tử KHÔNG HOẶC (NOR)*

Phép toán NOT - OR được gọi là phép toán KHÔNG HOẶC. Nó được xem là kết hợp của 2 toán tử : OR và NOT.




Ta có: $A + B + C = Y'$

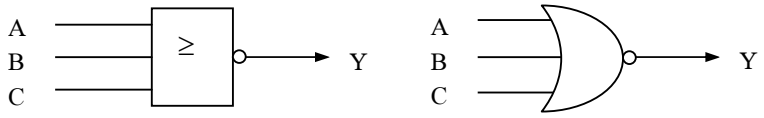
$Y' = \overline{Y}$ hay $Y = \overline{Y'}$

Vậy ta có: $Y = \overline{A+B+C}$

Ta có bảng trạng thái sau:

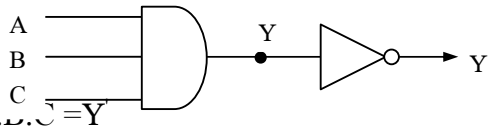
| A | B | C | $Y' = A+B+C$ | $Y = \overline{A+B+C}$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 |

Ký hiệu:




*3.1.5. Toán tử KHÔNG VÀ (NAND) :*

Phép toán NOT - AND được gọi là phép toán *không và*. Nó là sự kết hợp giữa hai toán tử NOT và AND.




Ta có: $A.B.C = Y'$

$Y' = \overline{Y}$ hay $Y = \overline{Y'}$

Vậy: $Y = \overline{A.B.C}$

Ta có bảng trạng thái:

| A | B | C | Y' =A.B.C | Y= $\overline{A.B.C}$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

Ký hiệu:


*3.1.6. Toán tử HOẶC HẠN CHẾ (EXCLUSIVE - OR) :*

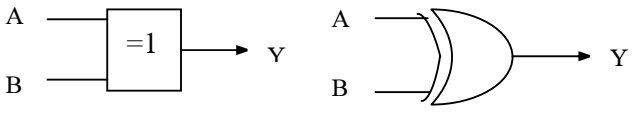
Toán tử này còn gọi là cộng loại trừ (*Exclusive - Or*) = XOR. Đây không phải là 1 phép toán cơ bản và có thể biểu diễn qua các cổng cơ bản AND, OR, NOT hay NAND, NOR.

Mạch này được dùng ở những nơi mà *2* tín hiệu số cần được *so sánh* (Nếu là 2 tín hiệu vào giống nhau đầu ra là 0 còn nếu tín hiệu vào là khác nhau thì tín hiệu ra = 1).

Ta có bảng trạng thái:

| A | B | Y = A ⊕ B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Ký hiệu:



### 3.2. Biểu diễn hình học các toán tử logic

Nguyên tắc: Tập hợp mọi trạng thái của một mệnh đề *X* có thể biểu diễn được bằng hình học. Ta quy định khi biểu diễn trên mặt phẳng, những trường hợp mệnh đề *X* là đúng (*ứng với trạng thái 1*) là diện tích nằm trong vòng tròn. Còn ngoài vòng tròn là diễn tả trạng thái *X = 0.*

- Vùng I là vùng giao của A và B

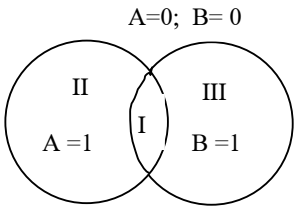
$I = A \wedge B$

Nó tương đương với toán tử VÀ

- Vùng I +I +III là vùng hợp của A và B

$I + II + III = A \vee B$

Nó tương đương với toán tử HOẶC




*3.2.1. Các tính chất cơ bản của toán tử NOT, AND, OR*

*a) Tính hoán vị :*

$A \cdot B = B \cdot A \quad (A \wedge B = B \wedge A)$ (2.1)

$A + B = B + A \quad (A \vee B = B \vee A)$ (2.2)

*b) Tính liên hợp :*

$(A \cdot B).C = A \wedge (B \wedge C) = A \wedge B \wedge C$ (2.3)

$(A + B) + C = A \vee (B \vee C) = A \vee B \vee C$ (2.4)

*c) Tính phân phối :*

$A.(B + C) = (A \wedge B) \vee (A \wedge C)$ (2.5)

$A + (B \cdot C) = (A \vee B) \wedge (A \vee C)$ (2.6)

*d) Tính bù :*

$A + \bar{A} = 1$ (2-7)

$A.\bar{A} = 0$ (2-8)

$A.0 = 0$ (2-9)

$A + 0 = A$ (2-10)

$A.1 = A$ (2-11)

$A + 1 = 1$ (2-12)

*3.2.2 Các định lý của đại số Boole*

*Bảng tổng hợp các định lý của đại số Boole :*

| Số thứ tự | ĐỊNH LÝ | Số thứ tự | ĐỊNH LÝ |
|---|---|---|---|
| 1 | A+0 =A | 12 | A.(A+B) =A |
| 2 | A.1 =A | 13 | A+$\bar{A}$.B =A+B |
| 3 | A+1 =1 | 14 | A.($\bar{A}$+B) =A.B |
| 4 | A.0 =0 | 15 | A.B +A.$\overline{B}$ =A |
| 5 | A+A =A | 16 | (A+B).(A+$\overline{B}$ ) =A |
| 6 | A.A =A | 17 | A.B +$\bar{A}$.C =(A+C).($\bar{A}$+B) |
| 7 | A+$\bar{A}$ =1 | 18 | (A+B).($\bar{A}$ +C) =A.C+$\bar{A}$.B |
| 8 | A.$\bar{A}$ =0 | 19 | A.B +$\bar{A}$.C+B.C =A.B + $\bar{A}$.C |
| 9 | A.(B+C) =A.B +A.C | 20 | (A+B).( $\bar{A}$ +C).(B+C) =(A+B).( $\bar{A}$ +C) |
| 10 | A+B.C = (A+B).(A+C) | 21 | $\overline{A.B.C....} = \overline{A} + \overline{B} + \overline{C} + ...$ |
| 11 | A+A.B =A | 22 | $\overline{A + B + C + ...} = \overline{A}.\overline{B}.\overline{C}....$ |

Định lý 21 và 22 gọi là định lý *De Morgan*.

Ta có thể chứng minh một số định lý sau đây:

*Định lý 10:* A+B.C =(A+B).(A+C)

Từ vế phải,ta khai triển và biến đổi, ta có:

(A+B).(A+C) = A.A +A.C +B.A +B.C = A+A.C +A.B +B.C

=A.(1+C+B) + B.C =A + B.C

Vậy định lý đã chứng minh xong.

*Định lý 12*: A.(A+B) = A

Khai triển vế trái và biến đổi, ta có: A.A +A.B = A+A.B =A.(1 +B) =A

Định lý đã được chứng minh xong.

*Định lý 13*: A + $\bar{A}$.B =A+B

Sử dụng luật phân phối (định lý 10), ta có:

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

*Định lý 19*: A.B + $\bar{A}$ .C +B.C = A.B + $\bar{A}$ .C

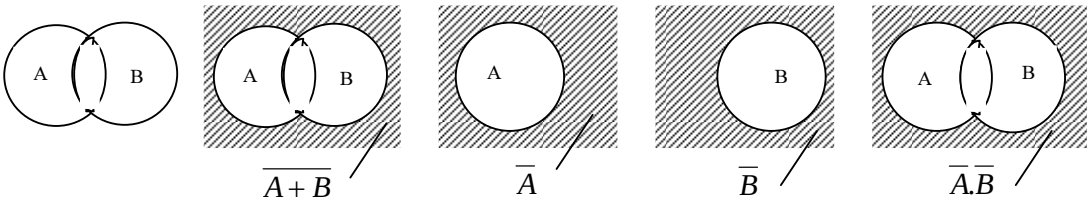
Ta nhân B.C với (A+$\bar{A}$ =1 ), ta có biểu thức sau:

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

Đ*ịnh lý 21 và 22* (*De Morgan).*
Lập bảng

| A | B | A+B | $\overline{A+B}$ | $\overline{A}$ | $\overline{B}$ | $\overline{A}.\overline{B}$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | ***1*** | 1 | 1 | ***1*** |
| 0 | 1 | 1 | ***0*** | 1 | 0 | ***0*** |
| 1 | 0 | 1 | ***0*** | 0 | 1 | ***0*** |
| 1 | 1 | 1 | ***0*** | 0 | 0 | ***0*** |

Ta có kết quả ở bảng trên: $\overline{A+B} = \overline{A}.\overline{B}$
Cũng có thể chứng minh theo phương pháp toán tập hợp:




### 3.3. Mạch logic

Có thể biểu diễn các đại lượng vật lý khác nhau trong thực tế như dòng điện, áp suất chất lỏng hoặc chất khí, tốc độ dịch chuyển, vị trí ... bằng các đại lượng nhị phân. *Mạch logic* là *1 phần tử* có nhiều đầu vào và một số đầu ra. Đầu ra là sự tổ hợp kết quả của các đầu vào. Tùy theo tính chất của phần tử mà có 2 dạng cơ bản:

- Mạch tổ hợp.
- Mạch tuần tự (*sequential circuits- dãy, chuỗi*).

*3.3.1. Các phương pháp biểu diễn hàm logic*

*a. Phương pháp biểu diễn bằng bảng trạng thái:*

Các giá trị của hàm phụ thuộc vào các trạng thái của các biến được trình bày trong một bảng bao gồm *n+1* cột và $2^n$ hàng, trong đó số biến là *n.* Nó thường được gọi là bảng chân lý hoặc bảng trạng thái. Ưu điểm của phương pháp biểu diễn này là ít nhầm lẫn, dễ nhìn và có tính rõ ràng và sáng sủa vì từ bảng đó nhìn thấy được giá trị của *hàm logic* ứng với từng tổ hợp. Tuy vậy nó cồng kềnh và đặc biệt khi mà số biến là khá lớn.

Thí dụ : Một hàm có *3 đầu vào* và có *1 đầu ra* có giá trị bằng *1* khi và chỉ khi chỉ có *1* đầu vào kích thích *( = 1*).

*Bẳng 3.1*

| A | B | C | S |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | **1** |
| 0 | 1 | 0 | **1** |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | **1** |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

Nhìn vào bảng chúng ta thấy hàm S =1 khi trạng thái các biến vào A=0, B=0 và C=1; A=0, B=1 và C=0; A=1, B=0 và C=0.

*b. Biểu diễn bằng phương trình logic*

*Phương trình logic* được viết xuất phát từ các vấn đề của bài toán đặt ra. Nhưng với bài toán phức tạp, cách viết này dễ dẫn đến sai lầm. Người ta đã chứng minh được rằng, một hàm logic có *n biến nhị phân độc lập* bao giờ cũng có thể biểu diễn thành các *hàm tổng của các tích và tích của các tổng*. Đây chính là các dạng chính tắc của *phương trình logic.*

** Dạng tổng của các tích( hay là phương trình chính tắc thứ nhất)*

Có thể lấy một ví dụ để minh họa cách biểu diễn *phương trình logic*.

Cho một *hàm logic* được biểu diễn bằng bảng trạng thái 3-2:

*Bẳng 3-2*

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

| | | | |
|---|---|---|---|
| 1 | 1 | 1 | 1 |

Từ bảng 3-2, giá trị của hàm Y phụ thuộc vào trạng thái của các biến vào mà có thể có giá trị 1, 0 hoặc không xác định ( *don't care- x*), nhưng ta chỉ quan tâm đến các hàng mà giá trị của hàm *Y=1.* Số lần mà hàm *Y=1* chính là số tích của các tổ hợp biến.

Trong mỗi tích đó, các biến có giá trị bằng *1* thì được giữ nguyên, còn các biến có giá trị bằng *0* thì lấy giá trị đảo, có nghĩa là nếu biến *Xi* có giá trị là *1* thì được viết là *Xi*, còn nếu *X*i có giá trị *0* thì được viết là $\overline{Xi}$ .

Cách biểu diễn dạng *tổng của các tích* được hình thành bằng cách viết tổng của các tích trên *(chú ý mỗi tích phải có mặt đầy đủ số biến của hàm).*

Từ bảng 3-2, ta có:

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

Hoặc có thể biểu diễn ở dạng rút gọn:

$$Y = \Sigma 0,5,7 \text{ với } N = 2,3,6$$

Trong đó 0,5,7 là thứ tự của tổ hợp biến tương ứng với các giá trị của hàm Y=1; N = 2,3,6 là thứ tự các tổ hợp biến mà hàm không xác định.

** Dạng tích của các tổng ( hay là phương trình chính tắc thứ hai)*

Từ bảng 3-2, ta chỉ quan tâm đến các hàng mà giá trị của hàm Y=0. Số lần mà hàm Y=0 chính là số tổng của các tổ hợp biến.

Trong mỗi tổng đó, các biến có giá trị bằng *0* thì được giữ nguyên, còn các biến có giá trị bằng *1* thì lấy giá trị đảo, có nghĩa là nếu biến Xi có giá trị là *0* thì được viết là Xi, còn nếu Xi có giá trị *1* thì được viết là $\overline{Xi}$ .

Cách biểu diễn dạng tích của các tổng được hình thành bằng cách viết tích của các tổng trên.

Từ bảng 3-2, ta có:

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

Hay có thể biểu diễn bằng cách viết rút gọn:

$$Y = \Pi\ 1,4 \text{ với } N = 2,3,6$$

Các chỉ số như đã nói ở phần trên.

*c. Biểu diễn hàm logic bằng ma trận Cácno (Karnaugh)*

Cách biểu diễn hàm logic bằng ma trận *Các nô* (hay người ta còn gọi là bảng *Cácno,* bìa *Các nô* hay bìa *K*) là một phương pháp diễn tả tổ hợp của các biến nhị phân cô đọng hơn, đồng thời giúp cho chúng ta một phương pháp hệ thống để đơn

giản hóa và thao tác các biểu thức *Boole*. Kỹ thuật này có lẽ là công cụ sử dụng rộng rãi nhất để tối thiểu các hàm *Boole*. Mặc dù kỹ thuật này có thể sử dụng cho số lượng biến bất kỳ, nhưng thường thì chúng chỉ nên dùng cho số biến tối đa là *6* vì nếu nhiều hơn sẽ gặp nhiều phức tạp, rắc rối và kết quả dễ bị nhầm lẫn.

Để biểu diễn một hàm có *n* biến, ta thiết lập một ma trận bao gồm $2^p$ hàng và $2^q$ cột với mỗi ô tương ứng với mỗi tổ hợp trạng thái các biến

Nếu số biến là chẵn (*n* chẵn) thì *p* và *q* có giá trị như nhau và bằng $\frac{n}{2}$. Trường hợp số biến là lẽ thì*: q = p+1* và đồng thời *p+q =n.*

Sau khi đã thiết lập được bảng ma trận *Cácno*, ta lần lượt ghi các giá trị nhị phân tương ứng của hàm vào trong từng ô.

Ví dụ có hàm *3 biến* A,B,C được biểu diễn như trên bảng 3-1, ta biểu diễn hàm dưới dạng ma trận *Cácno* như sau:

| A,B / C | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 (0) | 1 (2) | 0 (6) | 1 (4) |
| 1 | 1 (1) | 0 (3) | 0 (7) | 0 (5) |

Trên ma trận, các cột của nó được biểu diễn tương ứng trạng thái các biến A và B, còn hàng được biểu diễn theo trạng thái của biến C. Mỗi ô là tương ứng với tổ hợp trạng thái của các biến. Ví dụ ở ô số 2 tương ứng với $\overline{A}.B.\overline{C}$ và có giá trị hàm bằng *1*, còn ở ô thứ 7 tương ứng với A.B.C và gía trị hàm bằng *0*. Các chỉ số nhỏ biểu diễn ở góc của các ô là số thứ tự các tổ hợp hàm tương ứng.

Ví dụ thiết lập ma trận K cho hàm 4 biến:

| C,D / A,B | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 4 | 12 | 8 |
| 01 | 1 | 5 | 13 | 9 |
| 11 | 3 | 7 | 15 | 11 |
| 10 | 2 | 6 | 14 | 10 |

| | | | |
|---|---|---|---|
| | | | |

Mã nhị phân được sử dụng trong ma trận *Cácno* là mã nhị phân đối xứng, do vậy ma trận có tính chu kỳ, khi chuyển từ một ô sang một ô kế tiếp, chỉ có duy nhất một biến thay đổi, chính nhờ tính chất này mà ma trận *Cácno* được sử dụng rất thuận lợi để giải quyết các bài toán mà đặc biệt là trong các mạch *logic tuần tự*.

Ta chú ý rằng, do tính chất của mã nhị phân được sử dụng là đối xứng nên các ô nằm cùng một hàng ( một cột ) ở ngoài cùng bên trái và bên phải (ở trên cùng hay dưới cùng) cũng là các ô kề nhau và cũng chỉ có khác nhau một biến.

Một hàm 6 biến được biểu diễn như sau:

| A,B,C / D,E,F | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
|---|---|---|---|---|---|---|---|---|
| 000 | 0 | 1 | 3 | 2 | 6 | 7 | 5 | 4 |
| 001 | 8 | 9 | 11 | 10 | 14 | 15 | 13 | 12 |
| 011 | 24 | 25 | 27 | 26 | 30 | 31 | 29 | 28 |
| 010 | 16 | 17 | 19 | 18 | 22 | 23 | 21 | 20 |
| 110 | 48 | 49 | 51 | 50 | 54 | 55 | 53 | 52 |
| 111 | 56 | 57 | 59 | 58 | 62 | 63 | 61 | 60 |
| 101 | 40 | 41 | 43 | 42 | 46 | 47 | 45 | 44 |
| 100 | 32 | 33 | 35 | 34 | 38 | 39 | 37 | 36 |

*3.3.2. Các hàm không xác định (don't care)*

Có những trường hợp mà giá trị của *đầu ra* không được thiết lập theo sự tổ hợp của các trạng thái đầu vào ví dụ như khi mã hóa số thập phân từ 0 đến 9 bằng mã BCD gồm *4 bit* thì tương ứng phải có *16* tổ hợp trong khi ta chỉ cóa *10* tổ hợp được sử dụng, như vậy các tổ hợp 1001, 1010, 1011, 1100, 1110 và 1111 là không được sử dụng và những giá trị này người ta định nghĩa là những hàm không xác định.

Có những trường hợp mà tại đó, tổ hợp của các trạng thái các biến là không có ý nghĩa hoặc không thể xẫy ra ví dụ như khi cửa thang máy đóng mà tiếp điểm sàn thang máy chưa đóng xác định là không có người trong thang máy chẳng hạn thì trạng thái nút bấm gọi tầng trong buồng của thang máy có thể sẽ không có ý nghĩa

mà cũng có thể có ý nghĩa tùy theo nhà thiết kế. Chính vì thế mà trạng thái của các đầu ra của hàm cũng sẽ rơi vào trạng thái không xác định.

### *3.3.3. Phương pháp tối thiểu hóa các hàm logic*

Các *hàm logic* cũng tương tự như các hàm số toán học khác, có nghĩa là trong quá trình phân tích và tổng hợp chúng ta cần phải thực hiện một việc làm thường xuyên là tối thiểu hóa các hàm sao cho biểu thức được biểu diễn dưới dạng đơn giản nhất có thể mà vẫn đảm bảo được các chức năng yêu cầu, điều đó sẽ làm cho hàm đơn giản và tường minh hơn, song đối với *hàm logic* thì điều này còn mang một ý nghĩa hơn là sẽ làm giảm đáng kể số cổng cũng như các dây nối cần thiết và kích thước khuôn khổ của chúng cũng vì thế sẽ được giảm đi. Thực chất của việc tối thiểu hóa là tìm dạng biểu diễn đại số đơn giản nhất và thường có hai phương pháp:

- Phương pháp biến đổi trực tiếp sử dụng các định lý của đại số *Boole*.
- Phương pháp sử dụng thuật toán.

*a. Phương pháp biến đổi trực tiếp sử dụng các định lý của đại số Boole*

Căn cứ vào các định lý của đại số *Boole* để rút gọn *hàm logic*, tuy nhiên phương pháp này sẽ gặp rất nhiều khó khăn vì bị hạn chế bởi tính trực quan nên nhiều khi các kết quả đã được rút gọn nhưng vẫn chưa thể khẳng định được nó đã tối thiểu hay chưa. Chính vì thế mà phương pháp này ít được sử dụng, đặc biệt là trong việc tối thiểu hóa các hàm phức tạp.

Ví dụ, ta có một hàm mà có thể được biểu diễn chúng ở dưới dạng phương trình chính tắc thứ nhất -*Tổng của các tích*-.

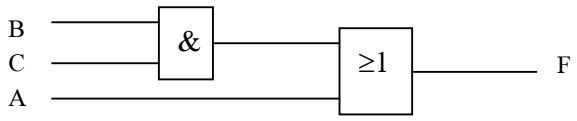
$F = \overline{A}.B.C + A.\overline{B}.\overline{C} + A.\overline{B}.C + A.B.\overline{C} + A.B.C$

Sử dụng các định lý của hàm *Boole*, ta có:

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

$$A.\left(B + \overline{B}\right) + \overline{A}.B.C = A(1 + B.C) = A + B.C$$

Hàm *logic* trên có thể biểu diễn bằng mạch logic số sau:




*b. Phương pháp sử dụng thuật toán*

** Phương pháp sử dụng ma trận Cácno*

Theo nguyên tắc chung của phương pháp tối thiểu sử dụng ma trận *Cácno* thì trước hết, ta lập ma trận như đã trình bày ở trên, sau đó ghi các giá trị của hàm tương ứng với các tổ hợp các trạng thái của các biến vào.

Tiếp theo, ta tiến hành xác định **cách tối thiểu hàm** theo kiểu *tích cực tiểu* hay *tổng cực tiểu*. Nếu xaс định *tổng cực tiểu* thì ta đánh dấu các ô có giá trị **1** và nếu xác định *tích cực tiểu* thì chọn các ô có giá trị **0**.

Chọn hai ô kề nhau có cùng giá trị *1* (hoặc *0*), theo tính chất của ma trận *Cácno* thi chúng chỉ khác nhau một biến, do vậy mà tổng hoặc tích của *hàm logic* có thể được giảm đi một biến.

Ví dụ ta có *hàm logic* có *n* biến, nếu ta viết theo dạng *tổng của các tích* của 2 ô kề nhau có cùng giá trị **1**, ta có:

$$(X_1.X_2....X_j...X_n) + (X_1.X_2....\overline{X_j}...X_n) = (X_1.X_2....X_{j-1}...X_n).(X_j + \overline{X_j})$$

$$= (X_1.X_2....X_{j-1}...X_n) \qquad \{ \text{do } (X_j + \overline{X_j}) = 1 \}$$

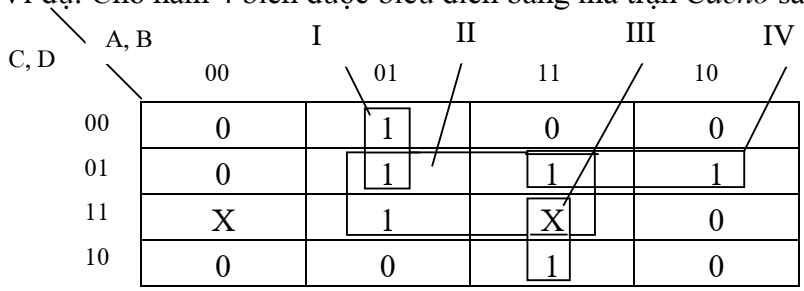
Như vậy biến $X_j$ sẽ không có mặt trong biểu thức trên mà *hàm logic* vẫn không thay đổi, hay nói cách khác tổng của chúng đã giảm đi được một biến.

Tương tự với trường hợp có *4* ô kề nhau có cùng một giá trị của hàm, bằng cách chứng minh như vậy, ta sẽ giảm được *2* biến.

Tổng quát với $2^k$ cặp ô kề nhau có cùng giá trị thì ta sẽ giảm được *k* biến.

Do tính chất của hàm số *Boole* mà giá trị *1* có thể tham gia vào một số nhóm, điều đó cho phép chúng ta có thể lựa chọn các ô kề nhau đến mức tối đa nhằm làm cho hàm càng đơn giản có nghĩa là hàm được tối thiểu. Tính chất đó là $X+X=X$.

Ví dụ: Cho hàm 4 biến được biểu diễn bằng ma trận *Cácno* sau




Nhận xét: Ma trận được biểu diễn như trên gồm có $2^n$ tổ hợp hàm tức là *16* tổ hợp *(n=4)*, tuy nhiên, chỉ có *6* tổ hợp hàm có giá trị **1** và *8* tổ hợp hàm có giá trị **0**, ngoài ra có *2* tổ hợp hàm không xác định được ký hiệu bằng chữ **" x".**

Về nguyên tắc chung thì người ta có thể tối thiểu hàm theo **dạng tổng của các tích** hay là **tích của các tổng**, có nghĩa là sự hoạt động của *hàm logic* sẽ không thay đổi nhưng *mạch logic* sẽ khác nhau và chắc chắn là khi lựa chọn mạch, người ta sẽ chọn mạch nào có số cổng ít nhất và tổng số lần các biến xuất hiện trong mạch là ít nhất (số *literal là ít nhất*).

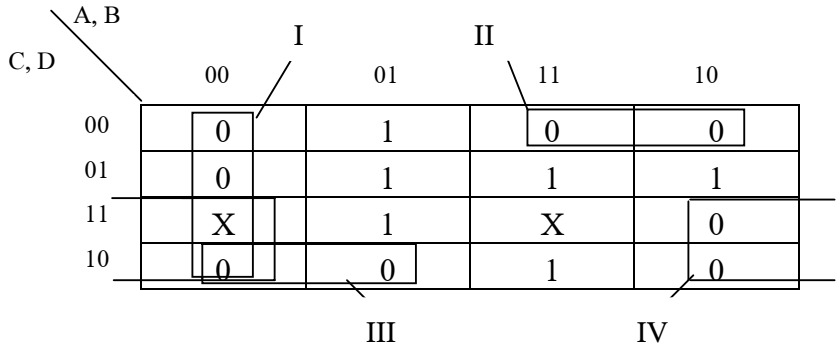
Đối với các tổ hợp có giá trị không xác định, về mặt nguyên tắc người ta coi là các hàm đó lấy các giá trị tùy ý, có nghĩa là hoặc giá trị **1** hoặc giá trị **0.** Tuy nhiên khi áp dụng thực tế thì nên chọn giá trị thích hợp sao cho việc tối thiểu là thuận lợi nhất và đạt được kết quả tối giản nhất có thể.

Trở lại với ví dụ trên, ta có:

Xác định theo *tích cực tiểu*, ta có các nhóm liên kết các ô kề nhau có gía trị **1** như được biểu diễn trên ma trận gồm có *4 nhóm (I, II, III, IV)* trong đó có *1* nhóm 4 ô và *3* nhóm 2 ô. Theo tính chất đã nêu trên, ta có *hàm logic* tối thiểu là:

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

Xác định theo *tổng cực tiểu*, ta có các nhóm liên kết các ô kề nhau có giá trị **0** như được biểu diễn dưới đây, nó gồm có 4 nhóm và theo tính chất của các nhóm có các ô kề nhau, ta có thể tìm thấy được hàm tối thiểu của tích các tổng.




$$F = I.II.III.IV = (A+B).\left(\overline{A}+C+D\right)\left(A+\overline{C}+D\right)\left(B+\overline{C}\right)$$

**. Phương pháp Quin Mc.Cluskey*

***- Một số định nghĩa***:

- *Đỉnh :* Là một tích chứa đầy đủ các biến của hàm xuất phát, nếu hàm có *n* biến thì đỉnh là tích của *n* biến.

Đỉnh *1* là đỉnh mà hàm có giá trị bằng *1* và đỉnh *0* là đỉnh có giá trị bằng *0.* Đỉnh không xác định là đỉnh mà tại đó hàm có thể lấy giá trị *1* hay *0.*

- *Tích cực tiểu:* Là một tích có số biến là cực tiểu để *hàm có giá trị bằng 1 hay không xác định*.
- *Tích quan trọng:* Là tích cực tiểu mà giá trị hàm *chỉ duy nhất* bằng *1*.

***- Phương pháp tiến hành***:

- Bước 1:

+ Tìm các tích cực tiểu bằng cách lập bảng biểu diễn các giá trị hàm bằng *1* và các giá trị *không xác định* rồi ghi vào *bảng a.*

+ Sắp xếp các tổ hợp biến theo *mã nhị phân* theo thứ tự *số các chử số 1* trong tổ hợp tăng dần từ *0,1,2,3*...ghi vào *bảng b.*

+ So sánh mỗi tổ hợp thứ *i* với tổ hợp *i+1*, nếu *2 tổ hợp* khác nhau chỉ ở một cột thì kết hợp chúng thành *1 tổ hợp mới* không có mặt giá trị cột đó (*chú ý là thứ tự các biến là phải được sắp xếp như nhau).* Tiếp theo, thay chổ cột không có mặt giá trị của nó bằng một dấu gạch "-" và đánh dấu V vào 2 tổ hợp củ và ghi vào *bảng c*.

+ Tiếp tục như trên bằng cách chọn tiếp các tổ hợp khác nhau chỉ 1 chử số *1* và có cùng một vị trí dấu gạch "-" ( *tức là có cùng một biến vừa bị giản ước ở bước trước đây*) và ghi vào *bảng d*.

Lại tiếp tục cho đến khi các tổ hợp không còn khả năng liên kết nữa thì đó chính là *tích cực tiểu* của hàm đã cho.

Ví dụ: Tối thiểu hóa hàm f($X_1$, $X_2$, $X_3$, $X_4$) với các đỉnh bằng *1* là L=2,3,7,12,14,15; Các đỉnh không xác định N = 6,13

| Bảng a | | Bảng b | | | Bảng c | | Bảng d | |
|---|---|---|---|---|---|---|---|---|
| Số thập phân | Số nhị phân $X_1$, $X_2$, $X_3$, $X_4$ | Số chử số 1 | Số thập phân | Số cơ số 2 $X_1$, $X_2$, $X_3$, $X_4$ | Liên kết | $X_1$, $X_2$, $X_3$, $X_4$ | Liên kết | $X_1$, $X_2$, $X_3$, $X_4$ |
| 2 | 0010 | 1 | 2 | 0010V | 2,3 | 001-V | 2,3,6,7<br>2,6,3,7 | 0-10 |
| 3 | 0011 | 2 | 3 | 0011V | 2,6 | 0-10V | 6,7,14,15<br>6,14,7,15 | -11- |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 6 | 0110 | | 6 | 0110V | 3,7 | 0-11V | 12,13, 14,15 | 11-- |
| 12 | 1100 | | 12 | 1100V | 6,7 | 011-V | | |
| 7 | 0111 | 3 | 7 | 0111V | 6,14 | -110V | | |
| 13 | 1101 | | 13 | 1101V | 13,13 | 110-V | | |
| 14 | 1110 | | 14 | 1110V | 12,14 | 11-0V | | |
| 15 | 1111 | 4 | 15 | 1111V | 7,15 | -111V | | |
| | | | | | 13,15 | 11-1V | | |
| | | | | | 14,15 | 111-V | | |

- Bước 2:

Việc tìm các *tích quan trọng* cũng được thực hiện theo nhiều giai đoạn. Gọi $L_i$ là tập các đỉnh *1* và *không có các đỉnh có chứa giá trị không* xác định, $Z_i$ *là tập các tích cực tiểu* và Ei *là tập các tích quan trọng* của bước thứ *i*.

+ Với *i = 0* ta có giả sử ta có $L_0 = L = (2,3,7,14,15)$ và $Z_0 = Z = (X_1X_3, X_2X_3, X_1X_2)$.

Xác định các tích quan trọng $E_0$ từ $L_0$ và $Z_0$ như sau:

Lập bảng có *mỗi hàng ứng với một tích cực tiểu thuộc $Z_0$, mỗi cột ứng với một đỉnh thuộc $L_0$*. Đánh dấu *"x"* vào các ô trong bảng ứng với tích cực tiểu bằng *1*.

Xét trên từng cột *nếu có chỉ 1 dấu "x"* thì *tích cực tiểu ứng với nó là tích quan trọng.*

+ Với *i=1*, Tìm $L_1$ từ $L_0$ bằng cách *loại khỏi $L_0$ các đỉnh 1 của $E_0$*; Tìm $Z_1$ từ $Z_0$ bằng cách *loại khỏi $Z_0$ các tích trong $E_0$ và các tích đã nằm trong hàng đã được chọn từ $E_0$* (các tích không cần thiết).

Lập bảng và bằng cacḥ tương tự như trên ta sẽ tìm tích quan trọng $E_1$

Tiếp tục công việc cho đến khi xét hết các cực tiểu.

$$L_{i+1} = L_i - E_i \text{ và } Z_{i+1} = Z_i - E_i$$

Lập bảng $L_{i+1}$, $Z_{i+1}$ để tìm $E_{i+1}$ ... cho đến khi $L_k = 0$

Trở lại với ví dụ, ta lập bảng:

| $Z_0$ \ $L_0$ | 2 | 3 | 7 | 12 | 14 | 15 |
|---|---|---|---|---|---|---|
| $\overline{X_1}.X_3$ | (x) | (x) | x | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| $X_2X_3$ | | | x | | x | x |
| $X_1X_2$ | | | | (x) | x | x |

Kết quả cuối cùng ta có hàm tối thiểu là $f = \overline{X_1}.X_3 + X_1.X_2$

*3.3.4. Mạch tổ hợp*

Nếu *đầu ra* tại bất kỳ một thời điểm nào chỉ phụ thuộc vào trạng thái của *đầu vào* ở tại thời điểm đó, nghĩa là không có phần tử nhớ ở trong mạch thì gọi là mạch *logic tổ hợp*. Theo quan điểm của điều khiển thì *mạch tổ hợp* là mạch hở hay nói cách khác là hệ *không có phản hồi*, tức là trạng thái hoạt động của các phần tử không bị ảnh hưởng của trạng thái tín hiệu *đầu ra.*

*a. Mô hình toán của mạch tổ hợp*

Về mặt toán học, giả thiết một mạch tổ hợp có *n đầu vào* và *m đầu ra* ta có thể biểu diễn nó bằng *m* phương trình đại số *Boole* như sau:

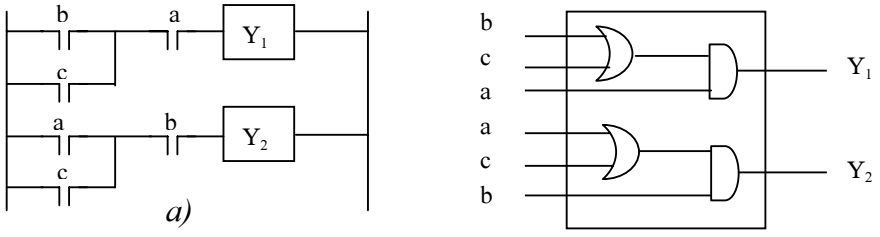
$$Y_j = f_i(X_1, X_2 ..., X_n) \text{ với } J = 1 \div m$$

Và mô hình toán có thể được biểu diễn bằng sơ đồ khối như sau:


*b. Phân tích mạch tổ hợp*

Nhiệm vụ của việc phân tích mạch tổ hợp là từ mạch đã có, mô tả hoạt động và viết các *hàm logic* của các đầu ra theo các biến đầu vào và tối thiểu hóa các mạch có thể.

Ví dụ có mạch tổ hợp được biểu diễn bằng mạch rơ le và bằng mạch số như sau:




*Hình 3-1: Mạch tổ hợp rơ le (a) và mạch số (b)*

Từ sơ đồ *mạch tổ hợp*, chúng ta có thể biểu diễn mạch theo cách lập bảng trạng thái, phương trình *logic* hay ma trận *Cácno*. Sau đó tối thiểu nó theo một trong những cách mà chúng ta đã biết và cuối cùng là thiết lập được một mạch tổ hợp tối giản có thể.

Theo cácphương pháp biểu diễn *hàm logic*, ta có thể biểu diễn nó bằng ma trận *Cácno* hay phương trình *logic* như sau:

Phương trình logic

$Y_1 = a(b+c)$

$Y_2 = b(a+c)$

Matrận *Cácno*

| c \ a,b | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 |

Y

| c \ a,b | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 |

$Y_2$

Để tối thiểu hàm logic này, ta có thể tiến hành như sau:

$Y_1 = a.b + a.c$

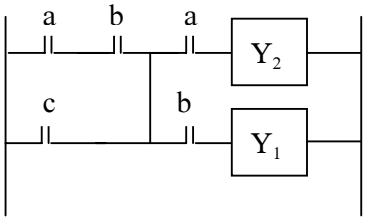
$Y_2 = b.a + b.c$

Từ đây ta có thể rút ra:

$a.b = b.a = a.c - Y_1 = b.c - Y_2$

Vậy ta có: $a.c + Y_2 = b.c + Y_1$

Từ đó ta có thể biểu diễn lại sơ đồ như sau:




So với sơ đồ ban đầu ta đã giảm di được 1 biến *(literal)*

*c. Tổng hợp mạch logic tổ hợp*

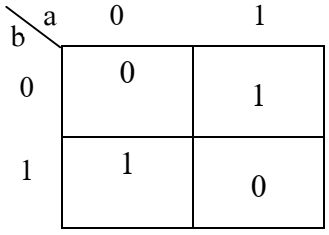
Tổng hợp mạch logic tổ hợp thực chất là thiết kế mạch tổ hợp, nhiệm vụ chính ở đây là từ yêu cầu của bài toán đặt ra chúng ta cần phải xây dựng được một mạch logic có tính chất vừa thỏa mãn các yêu cầu làm việc của hệ thống vừa phải đơn giản nhất có thể cũng như việc lựa chọn các phần tử của hệ thống là phù hợp với các yêu cầu thực tế.

Thông thường người ta thực hiện tổng hợp mạch logic tổ hợp theo kiểu mạch rơ le và mạch số. Mỗi một phương pháp thường mang tính đặc trưng riêng nhưng chức năng hoạt động là không có gì khác nhau.

Ví dụ: Thiết kế một hệ thống đèn cầu thang thỏa mãn các yêu cầu đặt ra là nếu khi tác động lên công tắc *a* hay *b* thì đèn sáng và nếu không một tác động nào lên cả *a* và *b* hay đồng thời tác động cả *a* và *b* thì đèn sẽ tắt.

Đây thực chất là một bài toán hoặc hạn chế (XOR) vì khi có 2 biến vào khác nhau thì đèn sáng hay nói cách khác là giá trị hàm bằng *1*. *(ở đây ta coi là khi đèn sáng tương ứng với giá trị hàm L=1 và khi đèn tắt thì L=0)*

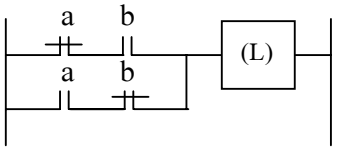
Có thể biểu diễn hàm logic bằng ma trận Cácno có 2 biến vào như sau:



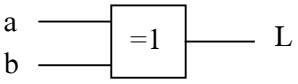

Ta có từ ma trận Cácno:

L= a.b +b.a = a ⊕ b

Sơ đồ mạch rơ le:



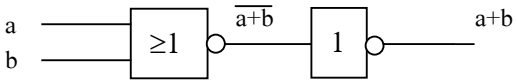

Và sơ đồ mạch số:



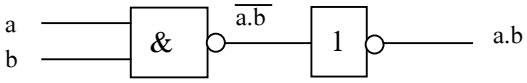

*d. Một số tính chất đặc biệt của mạch logic*

** Một số phép biến đổi thường gặp*

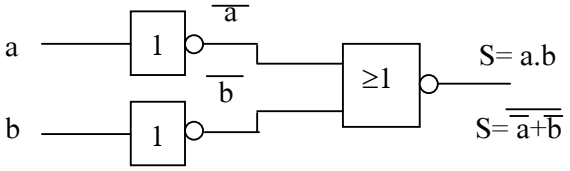
+ Biến đổi NOR thành OR




+ Biến đổi NAND thành AND

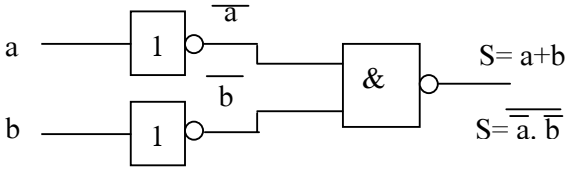



+ Biến đổi NOR thành AND




Theo định lý De Morgan thì $\overline{S= a+b} = \overline{a}.\ b$. Do vậy khi đầu vào là phần bù của a và b ($\overline{a}$, $\overline{b}$) thì ta có $S = \overline{a} + \overline{b} = a.b$.

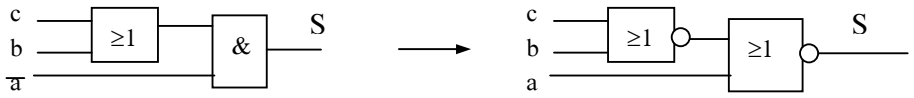
+ Biến đổi NAND thành OR



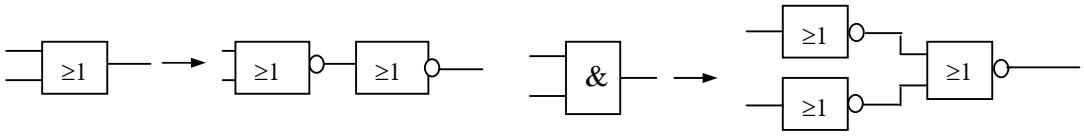
** Phương pháp biến đổi chỉ dùng một loại toán tử logic*

+ Biến đổi bằng sơ đồ TTL

Ví dụ hàm S = $\bar{a}$.(b + c), ta cần biến đổi hàm trên về chỉ hoặc AND hoặc OR




Trong phép biến đổi này, người ta sử dụng phương pháp biến đổi AND thành NOR và $\overline{\overline{a}} = a$.




+ Biến đổi bằng cách sử dụng các định lý đại số Boole

Ví dụ hàm S = $\bar{a}$.(b + c)

Ta bù 2 lần để không làm thay đổi giá trị của S: $S = \overline{\overline{\bar{a}.(b + c)}}$

Áp dụng định lý De Morgan, ta có: $S = \overline{a + \overline{(b + c)}}$ chỉ còn 1 loại TTL *OR*.

Ví dụ chỉ dùng TTL *AND:* $S = c.b + \bar{a}.x$

Ta bù 2 lần để không làm thay đổi giá trị của S: $S = \overline{\overline{c.b + \bar{a}.x}}$

Áp dụng định lý De Morgan, ta có: $S = \overline{\overline{c.b} . \overline{\bar{a}.x}}$ chỉ còn một loại TTL *NAND*

*3.3.5. Mạch tuần tự*

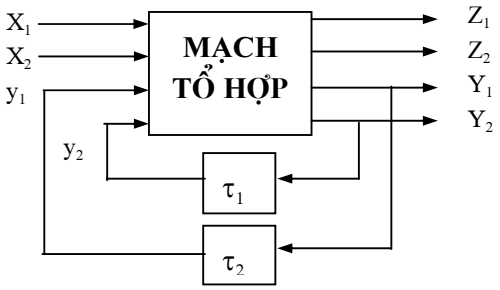
Đối với mạch mà *đầu ra* tại một thời điểm không những chỉ phụ thuộc vào trạng thái hiện tại của *đầu vào* mà còn phụ thuộc vào cả trình tự tác động của trạng thái *đầu vào ở trong quá khứ*, nghĩa là tồn tại các phần tử nhớ các trạng thái trong mạch, mạch đó gọi là mạch logic tuần tự. Như vậy, về mặt thiết bị thì ở mạch tuần tự không những chỉ có các phần tử đóng mở mà còn có cả phần tử nhớ. Một hệ thống logic tuần tự có thể chứa một số hệ thống logic tổ hợp con.

Sơ đồ cấu trúc của mạch tuần tự là mạch có phản hồi.

*a. Mô hình toán học của hàm tuần tự*

Sự hoạt động của mạch được thể hiện trên sơ đồ với các biến vào X và đầu ra Z cùng với các thay đổi của biến nội bộ Y và y. Do sự thay đổi của các biến vào x sẽ làm cho các tín hiệu ra Z thay đổi và cả tín hiệu Y cũng thay đổi. Mọi sự thay đổi của biến nội bộ Y sẽ được phản hồi vào biến vào y sau thời gian τ và nó lại tác động vào hệ làm thay đổi đầu ra Z ...

Nhiệm vụ của hệ là làm sao để mạch hoạt động ổn định có nghĩa là khi có sự thay đổi trạng thái tín hiệu đầu vào thì hệ thống phải chuyển từ một trạng thái ổn định này sang một trạng thái ổn định khác qua trạng thái quá độ.



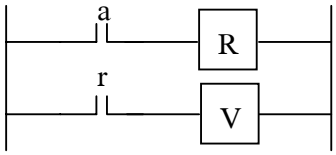

*b. Các trạng thái ổn định và quá độ*

Với logic tuần tự, các phép toán thực hiện theo dãy và trong mỗi một lệnh tiếp theo chỉ có thể được thực hiện khi trạng thái trước đó đã kết thúc.

Vậy vấn đề cơ bản ở đây là ngoài các biến vào đã biết còn có một số các biến thứ cấp mà ở trạng thái ban đầu chúng ta chưa biết.

** Trạng thái ổn định và quá độ trong logic tuần tự*

Ví dụ 1: Có sơ đồ rơ le sau:




| a | R | r | V | Trạng thái |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Ổn định |
| 1 | 1 | 0 | 0 | Quá độ |
| 1 | 1 | 1 | 1 | Ổn định |
| 0 | 0 | 1 | 1 | Quá độ |
| 0 | 0 | 0 | 0 | Ổn định |

Các mạch logic tuần tự đều có sử dụng rơle. Trên sơ đồ biểu diễn rơ le *R* có tiếp điểm thường mở *r, a* là nút bấm và chính là biến vào. Khi tác động *a (a=1)* có dòng điện qua *R (R=1),* tuy nhiên không phải ngay lập tức *R* đạt đến giá trị định mức nên phải sau một thời gian *Δt (Δt* bằng khoảng một vài phần trăm sec) thì tác động hút của cuộn dây mới có thể bắt đầu và tương ứng với trạng thái đó là r mới đóng lại *(r =1).* Khi *r* tác động thì *V* tác động *(V=1).*

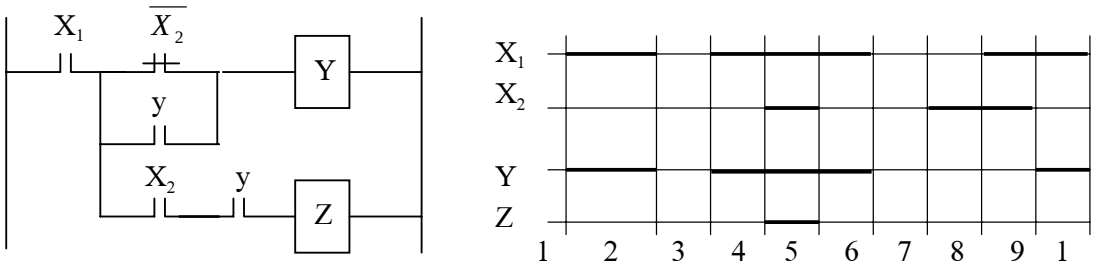
Khi cắt *a (a=0)* thì *R* mất điện *(R=0),* cuộn dây bị mất điện và lò xo sẽ đẩy tiếp điểm *r* trở về trạng thái ban đầu và sau một koảng thời gian $\Delta t_1$ nào đó *(*$\Delta t_1$ bằng khoảng vài phần trăm sec) thì *r* sẽ mở (*r=0).* Khi đó *V* sẽ hết tác động *(V=0).*

Trạng thái của các biến vào và gía trị đầu ra được biểu diễn như ở bẳng trên. Từ bảng chúng ta có nhận xét sau:

- Ở trạng thái *ổn định*, **R** và **r** lấy *cùng một giá trị*
- Ở trạng thái *quá độ*, *R* sẽ lấy *giá trị ổn định tiếp theo* của *r*.

Ví dụ 2: Có một mạch tuần tự được biểu diễn bằng mạch rơle, ta có thể biểu diễn trạng thái hoạt động của của mạch khi thay đổi các trạng thái biến vào như sau

Ta xây biểu đồ biểu thị sự hoạt động của mạch bằng cách sử dụng trục tung biểu thị các đại lượng vào, ra và trục hoành biểu thị thời gian.

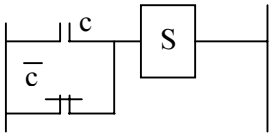



Các vạch đậm  biểu diễn trạng thái tác động (giá trị 1) và các vạch nhạt biểu diễn trạng thái 0 của hệ. Từ sơ đồ ta có trạng thái Z = 1 khi Y =1, $X_1$ và $X_2$ bằng 1 (cột số 5) tức là trình tự  các biến vào là $X_1$ =1 rồi đến $X_2$ =1. Còn nếu cho $X_2$ =1 trước rồi đến $X_1$ =1 sau thì rõ ràng là Y=0 và Z=0( cột thứ 9).

**. Các ngẫu nhiên về công nghệ*

Ngẫu nhiên công nghệ có nghĩa là có 1 sự rắc rối trong hoạt động của hệ thống - nói cách khác là hệ thống hoạt động không tin cậy-.

Ví dụ: Có một mạch logic như hình vẽ, sự hoạt động của hàm S không có gì thay đổi khi ta thay đổi biến vào c vì từ sơ đồ này ta có thể biểu diễn nó bằng phương trình đại số Boole sau:




$$S = c + \bar{c} = 1$$

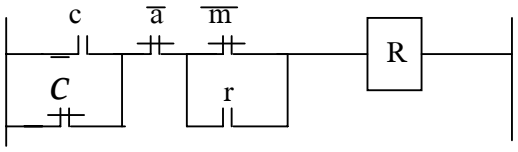
Tuy nhiên ta chú ý rằng, khi trạng thái biến c thay đổi từ *0* đến *1* thì $\bar{c}$ sẽ thay đổi từ *1* đến *0* và do sự thay đổi này của $\bar{c}$ nên có thể có một khoảng thời gian vài phần trăm sec nào đó hàm S có giá trị bằng *0* (khi mà *c* chưa đạt đến giá trị *1* mà $\bar{c}$ đã đạt giá trị *0*). Thời gian mà S = 0 là hoàn toàn phụ thuộc vào phần cứng của thiết bị (các phần tử thiết lập nên hệ thống).

Với trường hợp này thì không ảnh hưởng gì đến sự hoạt động của hệ thống *mạch locgic tổ hợp* vì sự thay đổi trạng thái này nói chung là rất ngắn và khoảng thời gian đó không đủ để gây ra các tác động khác.

Đối với *mạch logic tuần tự* thì điều này gây ra một sự rối loạn đáng kể vì trong mạch có chứa phần tử nhớ và khi trạng thái của hệ chuyển từ trạng thái *1* đến *1* qua trạng thái *0* rất ngắn thì phần tử nhớ ngay lúc đó củng chuyển trạng thái từ *1* đến *0* và dẫn đến kết quả là có thể làm rối loạn chu kỳ hoạt động bình thường.

Để thấy rõ điều này, ta khảo sát sơ đồ mạch rơle sau:

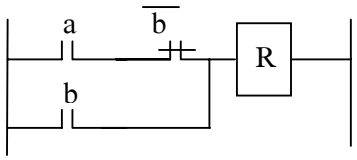
Khi tác động vào *c* và *c* đạt được giá trị *c=1* sau khi $\bar{c}$ qua giá trị *0* và dòng điện sẽ không qua R (R=0) trong khoảng thời gian tương ứng với khoảng từ khi mà $\bar{c} = 0$ đến khi *c=1*. Khi đó tiếp điểm thường mở của R là *r = 0* và nếu *m =0* lúc đó thì R= 0. Nếu sau khoảng thời gian trên và R sẽ bằng *1* và sau đó tiếp điểm thường mở *r =1*, lúc này nếu *m = 0* thì Rvẫn bằng *1*.




Để tránh các sự rắc rối trong hoạt động của hệ thống, người ta tìm biện pháp loại bỏ các ngẫu nhiên về công nghệ mà một trong những biện pháp đó là phải được giải quyết bằng phần cứng của nó. Khi thiết kế và chế tạo cần phải được chú ý và kiểm tra chính xác trước khi đưa vào hệ thống.

Ví dụ có một sơ đồ hoạt động được biểu diễn dưới đây:

Ta có thể biểu diễn phương trình logic bằng:



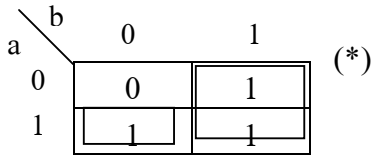

$$R = a.b + b$$

Theo định lý về tính phân phối thì:

$$R = a.b + b = (a+b).(b+b) = a+b \quad (\text{vì } b+b = 1)$$

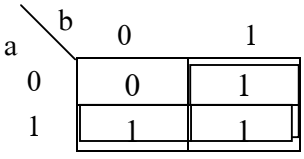
Tuy nhiên vì lý do công nghệ mà không phải chắc chắn là *b+b=1* mà có thể có một khoảng thời gian rất ngắn mà tổng đó bằng không. Điều này tương ứng với sơ đồ được biểu diễn trên ma trận đưới đây(*)

$$R = a.b + b$$




(*)

(**)



Trên sơ đồ (**) biểu diễn khi hệ thống có 1 ô được phủ 2 lần nên không có trường hợp nào mà R = 0.

$$R = a + b$$

*c. Giải bài toán logic tuần tự*

Bài toán mạch logic tuần tự là bài toán khó, hơn nữa từ một yêu cầu được đặt ra lại có nhiều cách giải quyết khác nhau, do vậy vấn đề chính ở đây là cần dựa vào một chỉ tiêu tối ưu nào đó, đồng thời để tìm được lời giải tối ưu thì ngoài các suy luận toán học logic người thiết kế còn phải tận dụng các kinh nghiệm thực tể rất phong phú và đa dạng. Sau đây chỉ giới thiệu hai phương pháp cơ bản là phương pháp *bảng chuyển trạng thái* (hay còn gọi là ma trận) và phương pháp *trực tiếp từ ma trận Cácno.*

** Phương pháp dùng bảng trạng thái*

Các bước tiến hành:

- Thiết lập ma trận sơ khai theo đó ghi tất cả trạng thái đã biết của các biến, ma trận này được thiết lập từ số cột bằng $2^n$ trong đó *n* là số biến vào.
- Việc chuyển từ một giai đoạn hoạt động này sang giai đoạn khác chỉ có thể được tiến hành bằng cách thay đổi trạng thái của chỉ một biến.
- Các trạng thái ổn định và trạng thái quá độ phát sinh luôn luôn cùng nằm trong cùng một cột.
- Mỗi hàng ngang chỉ dành cho một trạng thái ổn định, nhưng cũng có thể có một hay nhiều trạng thái quá độ.
- Ma trận chập nhận được bằng cách ghép các hàng ngang. Người ta cũng chỉ có thể ghép các hàng sao cho không có trạng thái ổn định thuộc cùng một cột. Các hàng có cùng số (là các trạng thái ổn định hay quá độ và cùng các ô trống) thì ưu tiên cho các trạng thái ổn định.
- Nếu với bài toán chỉ có một biến phụ thì ta có thể tiến hành như sau:

** Với các trạng thái ổn định, X và tiếp điểm của nó x có cùng giá trị. Với các trạng thái quá độ X lấy giá trị của trạng thái ổn định ở tiếp ngay sau đó.*

- Với bài toán nhiều biến phụ thì ta thu được các phương trình của các Rơle này như sau:

** Với các trạng thái ổn định, một rơle như X có cùng với giá trị tiếp điểm x của nó.*

** Với các trạng thái quá độ kề sát trạng thái ổn định mà nó hướng đến thì X lấy giá trị sát giá trị của trạng thái ổn định tiếp theo.*

* Với các trạng thái ổn định riêng rẽ, các trạng thái ổn định mà nó hướng tới người ta định các đường hoạt động chỉ bằng mũi tên. Các trạng thái quá độ kề bên

các đường thay đổi trạng thái của rơle lấy giá trị sát giá trị trạng thái ổn định tiếp theo.

-Đối với hàm sơ cấp thì trong các bảng, các đầu ra của các trạng thái ổn định thì lấy đúng giá trị của nó quá độ liên hệ giữa 2 trạng thái ổn định có các giá trị khác nhau có thể lấy giá trị *1 hay 0*, nếu 2 trạng thái ổn định có cùng giá trị thì nó lấy ngay chính giá trị đó

Ví dụ:

Có một thiết bị hoạt động theo nguyên tắc sau: khi tác động lên nút ấn *m* cho phép chạy và một tác động lên *a* cho phép dừng.

Ta có thể biểu diễn quá trình hoạt động của thiết bị theo bảng trạng thái sau:

| m | a | F |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 0 | 1 |
| 0 | 1 | 0 |

Từ bảng trạng thái, ta thấy rõ sự hoạt động của thiết bị F như sau:

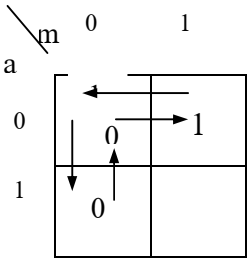
Khi m =0, a =0 thì F = 0 (Trạng thái ổn định)

Khi m =0, a =0 thì F = 1 (Trạng thái ổn định)

Khi m =0 (hết tác động), a =0 thì F =1

Khi m =0, a =1(tác động a) thì F =0

Ta có thể biểu diễn nó bằng ma trận Cácno:



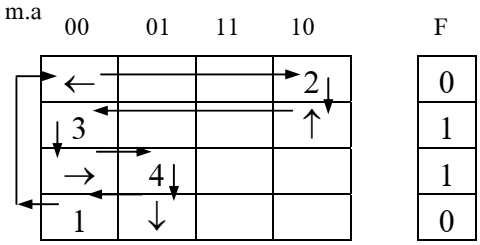

Tại ô $\bar{a}.\bar{m}$: Ta thấy rằng có 2 giá trị trong cùng một ô là F = *0* và F =*1*. Điều này là không thể có được trong sự hoạt động của một thiết bị. Điều này chứng tỏ rằng thiết bị này không tể thể hoạt động theo nguyên tắc của mạch logic được. Vì vậy cần phải có thêm một biến phụ mà trong thực tế đó chính là 1 rơle.

Để giải bài toán này, ta sẽ tiến hành theo từng bước sau đây:

+ Thiết lập ma trận sơ khai:

Đây là bài toán có 2 biến vào là *m* và *a*. Vậy ta thiết lập ma trận có 4 cột và tương ứng với các cột là biểu diễn các trạng thái của biến vào tương ứng với mã nhị phân đối xứng, nghĩa là khi chuyển từ ô này sang một ô kế tiếp chỉ có một biến thay đổi.

Mỗi giai đoạn hoạt động được biểu thị trong 1 cột ứng với trạng thái của các biến vào. Các giai đoạn ổn định của hàm được ghi vào trong *1 vòng tròn* và khi chuyển từ 1 trạng thái ổn định này sang 1 trạng thái ổn định khác thì phải qua một trạng thái quá độ được đánh số bình thường.




Số hàng của ma trận được xác định theo số các trạng thái ổn định của hàm.

Giá trị của hàm F được ghi vào một bảng tương ứng với giá trị của các hàng.

*Nhận xét:*

Từ ma trận trên chúng ta thấy rằng:

- Khi không có một tác động nào lên *a* và *m* thì F = 0 và tương ứng ta có trạng thái ổn định ←.
- Khi có một tác động lên *m* (ấn lên *m*) tức *m =1* thì F =1 sau khi qua trạng thái quá độ 2 để đến trạng thái ổn định ↑ (*a* vẫn bằng 0).
- Nhả nút ấn *m (m =0)* thì giá trị của hàm không thay đổi (thiết bị vẫn hoạt động) F=1 và sau khi chuyển qua trạng thái quá độ 3 để đến trạng thái ổn định →.
- Tác động lên *a (a =1)* thì thiết bị dừng và F=0 sau khi qua trạng thái quá độ 4 để đạt đến trạng thái ổn định ↓.
- Cuối cùng là trở về trạng thái ổn định ← sau khi qua trạng thái quá độ 1.

+ Thiết lập ma trận chập: (ma trận xếp chồng)

*Theo nguyên tắc nhóm nhiều nhất các hàng bằng cách xếp chồng các ô có cùng một số ( có thể ổn định hay quá độ) hoặc 1 ô có số với một ô trống. Khi các ô có cùng một số thì ưu tiên xếp cho ô có trạng thái ổn định.*

Từ ma trận trên, ta có ma trận xếp chồng sau:

Đây là một ma trận có 8 ô tương ứng với một hàm 3 biến là *m, a* và một biến phụ là tiếp điểm *x* của rơ le X, biến này người ta còn gọi là biến thứ cấp và ở hàng trên tương ứng với *x = 0* và hàng thứ 2 tương ứng với *x = 1*.

| m.a | | 00 | 01 | 11 | 10 |
|---|---|---|---|---|---|
| x | 0 | ← | ↓ | | 2 |
| | 1 | → | 4 | | ↑ |

Ở hàng trên có 2 trạng thái ổn định ← và ↓.

Ở hàng thứ 2 có 2 trạng thái ổn định → và ↑.

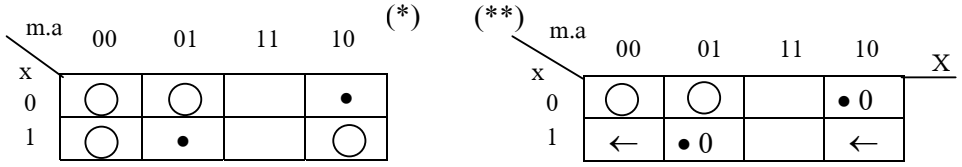
Các ô quá độ biểu diễn bằng các con số, còn các ô trống biểu diễn các trạng thái không xác định.

+ Xác định phương trình của rơ le X:

Chúng ta sẽ thể hiện trên 1 ma trận chập bằng cách đặt vào các ô có trạng thái ổn định 1 vòng tròn và 1 dấu chấm ở ô có trạng thái quá độ (*).

Đặt các giá trị của hàm X theo nguyên tắc sau (**):

- Với trạng thái ổn định thì giá trị của rơ le X và tiếp điểm *x* lấy cùng giá trị. Nếu *x = 0* thì X =0 và *x =1* thì X =1
- Với trạng thái quá độ X lấy giá trị của trạng thái ổn định ngay tiếp theo




+ Tối thiểu hóa mạch logic:

Từ ma trận trên ta có: $X = m.\bar{a} + x.\bar{a} = \bar{a}.(m + x)$

+ Xác định phương trình của đầu ra F:

Tương tự như trên, chúng ta thiết lập một ma trận gồm 8 ô và đặt các vòng tròn vào các ô tương ứng với trạng thái ổn định của hàm và 1 dấu chấm đặc trưng cho trạng thái quá độ. Ma trận được thiết lập từ các giá trị của ma trận sơ khai mà trong đó, các giá trị ổn định được lấy theo giá trị tương ứng của hàm F (ở ma trận sơ khai) và các giá trị quá độ sẽ lấy theo ma trận sơ khai hay có thể theo nguyên tắc là nếu nằm giữa 2 trạng thái ổn định có cùng 1 giá trị thì nó lấy chính giá trị đó. Còn nếu nằm giữa 2 trạng thái ổn định có giá trị khác nhau thì có thể lấy theo giá trị ổn định ngay trước đó hay sát ngay sau đó (tức là có thể lấy giá trị *0* hay*1* tùy ý mà kết quả mạch sẽ không thay đổi).


+ Tối thiểu hàm logic, ta có: F = x

** Phương pháp trực tiếp sử dụng ma trận Cácno*

Phương pháp này thích hợp với các mạch điện tử và thường được sử dụng trong công nghiệp để nghiên cứu các mạch khí nén.

Các bước tiến hành giải bài toán logic tuần tự bằng phương pháp trực tiếp sử dụng ma trận Cácno:

- Thiết lập ma trận Cácno với các biến đã biết bằng việc biểu diễn chu trình hoạt động bằng các đường có mũi tên.
- Ghi các giá trị đầu ra vào các ô. Việc biểu thị được thực hiện từ 1 ô sang một ô kế tiếp.
- Nếu sự hoạt động dẫn chúng ta một lần nữa đến 1 ô đã bị chiếm bằng một biến ra khác với biến mà chúng ta đã đặt vào đó, người ta sẽ tăng gấp đôi bảng ban đầu và tiếp tục biểu thị trong 1 ô đối xứng.
- Bằng cách tăng gấp đôi bảng ban đầu, người ta làm xất hiện một rơ le phụ như rơ le X và các biến thứ cấp của đầu vào mà ta thêm vào sẽ là $x$ và $\bar{x}$.
- Phương trình của X:

+ X = *1* đối với các ô ứng với $x$ ($x$ =*1*) kể cả cá ô điều khiển sự chuyển trạng thái của rơ le từ *0* đến *1* và một số ô điều khiển sự chuyển trạng thái của rơ le từ *1* đến *0*.

+ Các ô điều khiển sự chuyển trạng thái và các ô nằm kề sát đường đối xứng của giản đồ đầy đủ được đánh dấu bằng nét đậm hay nét màu.

+ Tất cả các ô có giá trị X =1 được gạch chéo.

+ Để có phương trình của X người ta hợp nhóm các ô gạch chéo không quan tâm đến nội dung của nó.

- Phương trình của đầu ra:

Nhận xét về các giá trị đầu ra trong các ô điều khiển sự chuyển trạng thái của rơ le ( các ô có nét đậm và các ô ứng với các trạng thái quá độ).

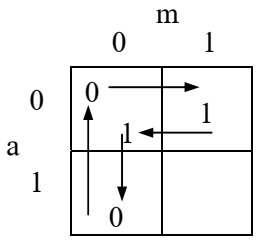
+ Khi các ô này nằm giữa 2 giá trị khác nhau của đầu ra người ta có thể biểu thị bên trong ô bằng *0* hay bằng *1*, người ta đánh dấu bằng 1 dấu chấm "•".

+Phương trình của đầu ra nhận được bằng cáh ghép các ô có đánh dấu 1 và nếu điều đó cho phép ghép nhóm dễ dàng thì ghép các ô có dấu chấm "•" và các ô trống.

Ví dụ :

Có một thiết bị hoạt động theo nguyên tắc sau: khi tác động lên nút ấn *m* cho phép chạy và một tác động lên *a* cho phép dừng.

Ta biểu diễn các giá trị của F trong ma trận Cácno. Sự có mặt của 1 ô mang 2 giá trị *1* và *0* tương ứng với các biến *a=0* và *m=0* có nghĩa rằng đây không phải là một bài toán logic tổ hợp vì thế cần thiết phải thêm 1 rơ le.




Ta xây dựng một ma trận thứ hai bằng cách nhân đôi ma trận thứ nhất để thêm biến *x* của rơ le X..

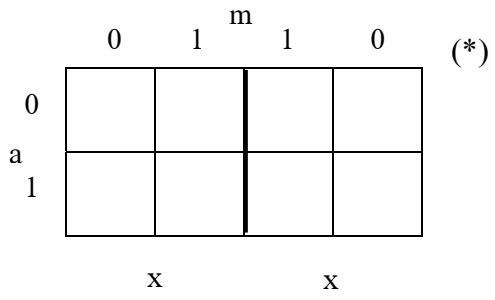
Hai phần của ma trận mới được ngăn cách nhau bằng nét kẽ đậm và ứng với hai trạng thái có thể của rơ le X mà chúng ta vừa mới thêm vào. Các tiếp điểm của rơ le này có thể được biểu diễn như hình dưới đây (*).

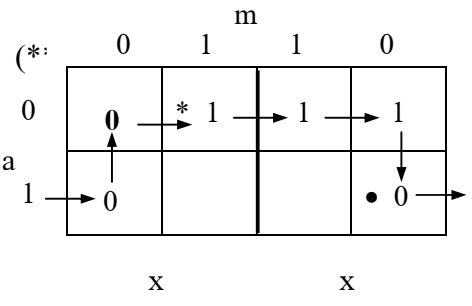
Sự hoạt động tương ứng với bài toán được đặt ra sẽ được biểu thị bằng cách chuyển trạng thái từ một ô sang một ô kế tiếp. Ta thấy rằng, khi tác động 1 xung lên *m (m =1)* thì chu trình hoạt động của thiết bị sẽ bắt đầu chuyển từ ô có trạng thái *0* (*m= 0* và *a=0*) sang ô đánh dấu "∗", rơ le sẽ thay đổi trạng thái và chuyển từ giá trị *0* sang giá trị *1.* Do vậy ô này tương ứng với một trạng thái quá độ rất ngắn (vài phần sec). Ta sẽ đặt giá trị F *=1* trong ô này hoặc F có thể sau vài phần sec có thể lấy giá trị *1*. Sự hoạt động vẫn thỏa mãn nếu ta lấy giá trị của ô này bằng *0.*

Khi tác động vào biến *a* sẽ dẫn chúng ta đến ô đánh dấu "•", rơ le thay đổi trạng thái và chuyển từ giá trị *1* sang gía trị *0*. Đó cũng là một trạng thái quá độ và cũng tương tự như trên, ô này có thể lấy giá trị *0* hoặc *1* (**).

Tuy vậy chúng ta cần chú ý là *khi một trạng thái quá độ nằm giữa 2 trạng thái ổn định có giá trị khác nhau thì có thể lấy các giá trị 0 hoặc 1, còn nếu giữa 2 trạng thái ổn định có cùng giá trị thì nó lấy chính giá trị ổn định đó.*

*Chú ý: Một ô có trạng thái quá độ là ô nằm kề với đường biên hoặc đường kẽ đậm và có mũi tên chỉ chu trình cắt ngang chúng.*



- Xác định phương trình của X:

+ Rơ le X chuyển sang trạng thái *1* khi chu trình làm việc chuyển tới ô có đánh dấu "*". Do vậy ô này và tất cả các ô khác ở phía bên phải phải bằng *1* trừ ô có đánh dấu "•".

+ Gạch chéo các ô có X=*1* và nhóm các ô gạch chéo này cho dù nội dung của nó là bao nhiêu. Ta có (*).

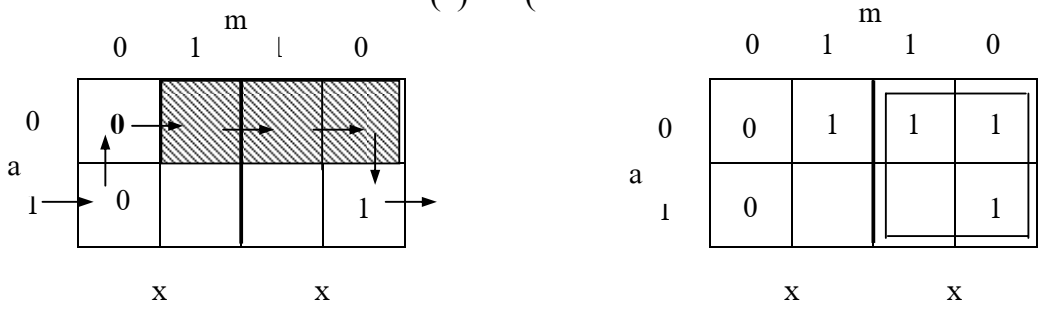
$$X = a.m + a.x = a(m+x).$$

- Xác định phương trình của F:

Với F, ta lấy các ô có giá trị *1* bằng cáh nhóm hoặc không nhóm các ô có giá trị *0* hay các ô trống. Ta có (**).

$$F = x$$

(*) (**

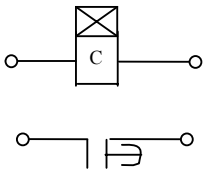



*d. Phần tử nhớ*

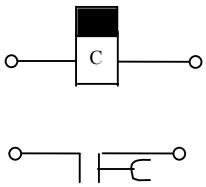
** Rơ le thời gian*

Đây là một phần tử nhớ kiểu 2 trạng thái 0 và 1, tuy nhiên sẽ tồn tại một trạng thái khá dài qúa độ - gồm có loại đóng chậm và ngắt chậm-.

Ký hiệu:




Rơ le đóng chậm




Rơ le ngắt chậm

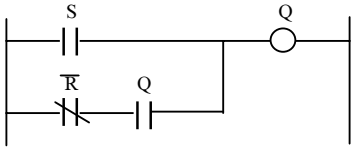
** Các mạch lật(Flip-Flop) Trigơ*

Mạch lật là các phần tử có khả năng nhớ 1 trong 2 trạng thái 0 hoặc 1. Nó gồm có một hoặc một số đầu vào điều khiển và hai đầu ra ở trạng thái ổn định Q và /Q.

- *Mạch lật SR*

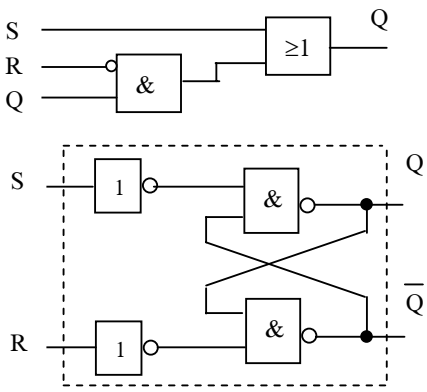
Mạch SR Flip -Flop có thể được thực hiện theo kiểu cổng NAND với phần tử trội là cổng SET hoặc kiểu NOR với phần tử trội là cổng RESET.

Từ sơ đồ mạch điều khiển, ta có:




$Q = S + (\overline{R}. Q)$

Biểu diễn theo mạch số và biến đổi, ta có:



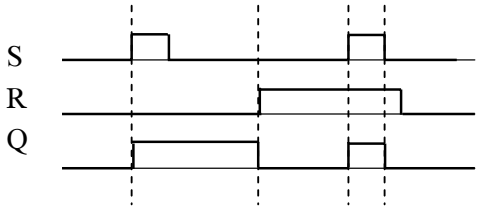

Ký hiệu EU Ký hiệu US

S = SET Đặt giá trị 1(ON)

R = RESET Trở về 0 (OFF)

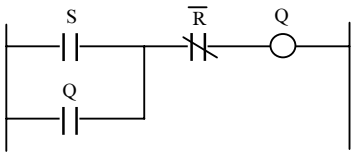
Ta có bảng trạng thái và biểu đồ xung của loại Flip- Flop SR có SET trội:

| S | R | $Q_n$ |
|---|---|---|
| 0 | 0 | $Q_{n-1}$ |
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 1 | 1 |



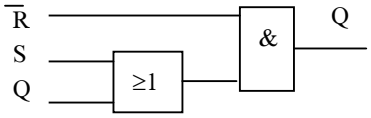

Mạch lật SR theo liểu cổng NOR với RESET trội

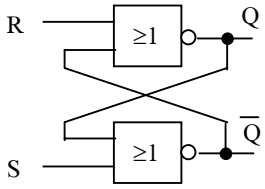
Ta có sơ đồ điều khiển như sau:

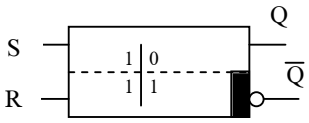



$Q = (S+Q).\overline{R}$

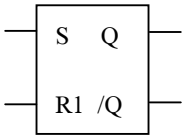
Biểu diễn bằng mạch số và biến đổi, ta có



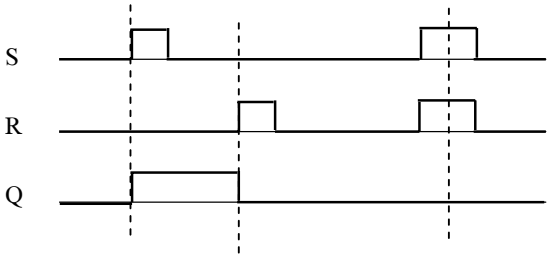

Ký hiệu EU




Ký hiệu US

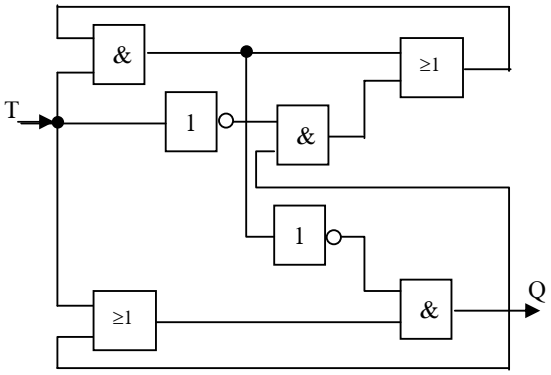
Ta có bảng trạng thái và biểu đồ xung SR Flip- Flop RESET trội như sau:

| S | R | $Q_n$ |
|---|---|---|
| 0 | 0 | $Q_{n-1}$ |
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 1 | 0 |

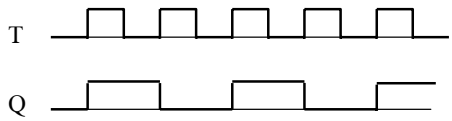



*-. Mạch lật kiểu T:*

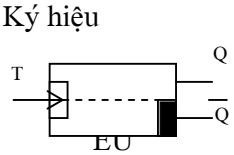
Tín hiệu vào là tín hiệu xung, nó được sử dụng trong các bộ đếm và bộ ghi. Mạch logic của nó được biểu diễn như sau:

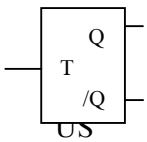



Biểu đồ xung:




Ký hiệu

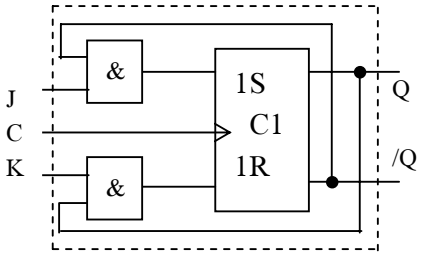


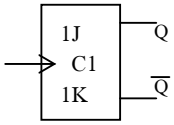
-. *Mạch lật JK*

Flip -Flop kiểu JK gồm có 2 cổng vào J,K tương ứng với SET, RESET và 1 xung C, 2 cổng ra Q và /Q. Điều khác nhau giữa JK và SR là ở chổ có thể kết hợp với mọi tín hiệu ở cổng vào đều cho phép.

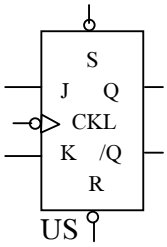
Sơ đồ mạch logic của JK:




Ký hiệu:



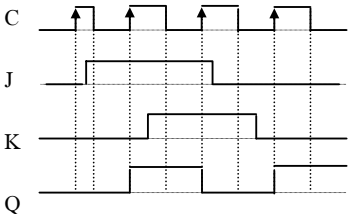

EU

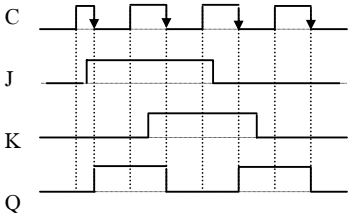



US

Bảng trạng thái và biểu đồ xung của JK:

| C | J | K | $Q_n$ |
|---|---|---|---|
| ↓ | 0 | 0 | $Q_{n-1}$ |
| ↓ | 0 | 1 | 0 |
| ↓ | 1 | 0 | 1 |
| ↓ | 1 | 1 | ⊗ |



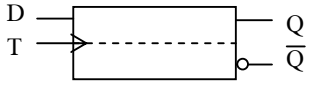

Chú ý: ⊗ *Thay đổi trạng thái* *Thay đổi TT ở sườn tăng*
*Thay đổi TT ở sườn giảm*

- *Mạch lật D*

Mạch lật kiểu D có 1 tín hiệu điều khiển là D, tín hiệu T và 2 đầu ra Q và /Q.

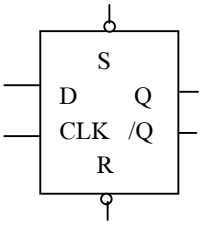
Nó được sử dụng trong các thanh ghi dịch trong kỹ thuật điều khiển.

Ký hiệu:



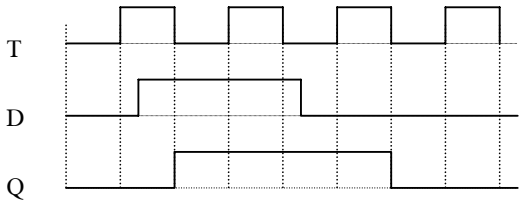

EU

US




Bảng trạng thái và biểu đồ xung mạch lật kiểu D:

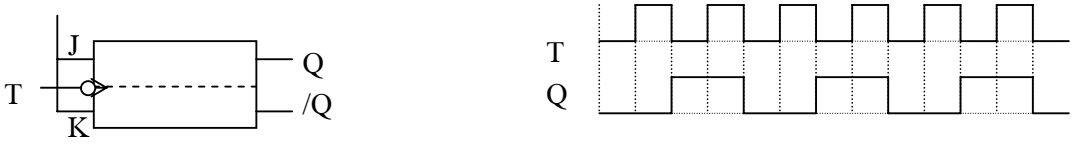
| T | D | $Q_n$ |
|---|---|---|
| ↓ | 1 | 1 |
| ↓ | 0 | 0 |
| | | $Q_{n-1}$ |



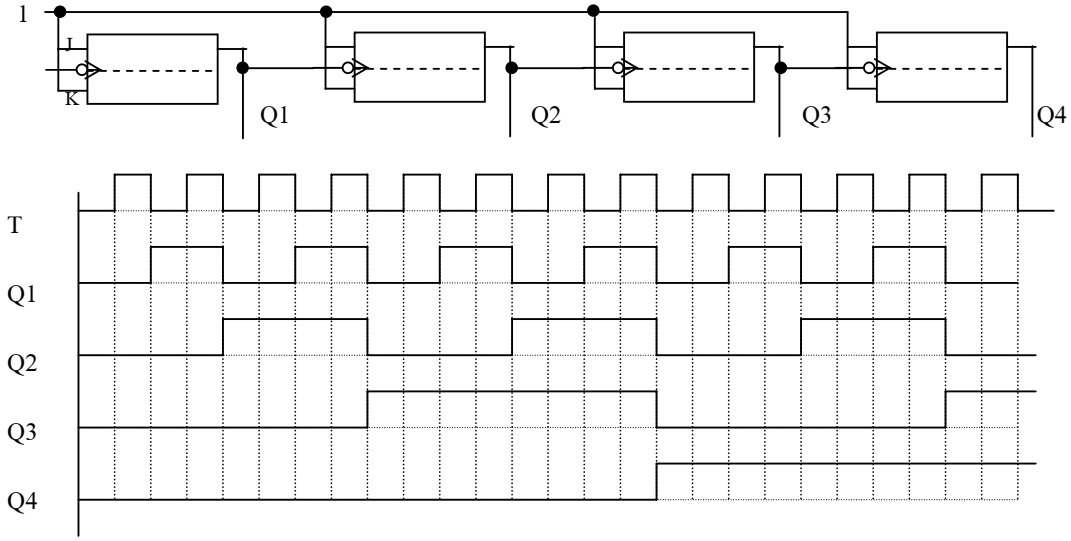
*e. Các toán tử logic ứng dụng vào bộ đếm và bộ ghi*

*- Nguyên tắc đếm:*

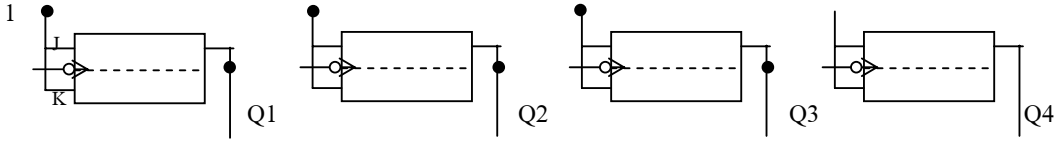
Sử dụng một mạch lật JK trong trường hợp cho J.K =1, Khi đó trạng thái đầu ra sẽ thay đổi theo giá trị của xung T. Như vậy có thể coi mỗi mạch JK là một bộ chia đôi.




*- Bộ đếm không đồng bộ*




*- Bộ đếm đồng bộ*




*- Thanh ghi dịch* (Shift Register)

Thanh ghi dịch là 1 bộ nhớ rất quan trọng trong các hệ chuỗi kế tiếp đồng bộ, bao gồm 1 loạt các bộ nhớ cơ sổ nối tiếp nhau. Nội dung tin chứa trong thanh ghi sẽ được dịch chuyển từng bước 1 sang trái hay phải theo chu trình khi có 1 xung ra lệnh dịch chuyển. Có thể biểu diễn sự làm việc của thanh ghi dịch như sau :

Ta gọi : bi là bộ nhớ thứ i và bi+1 là bộ nhớ kế tiếp.

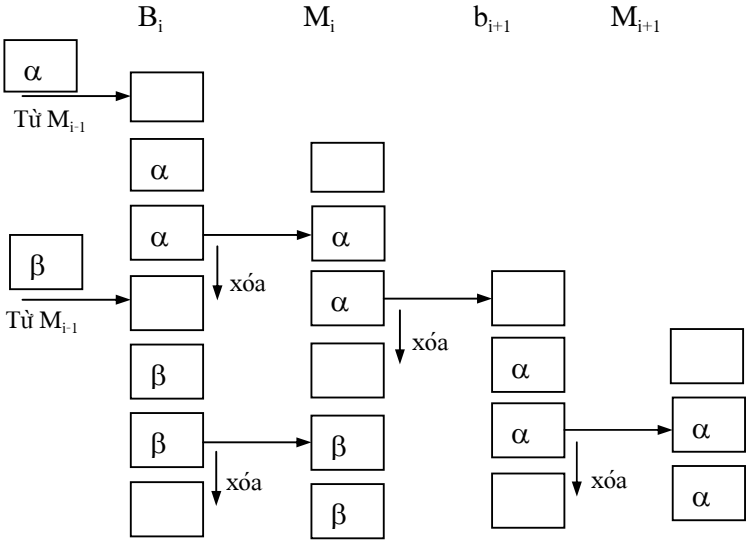
Mi là bộ nhớ phụ giữa 2 bộ nhớ bi và bi+1.

α;β : thông tin nhị phân cần dịch chuyển.

Quá trình xảy ra sau 4 bước :

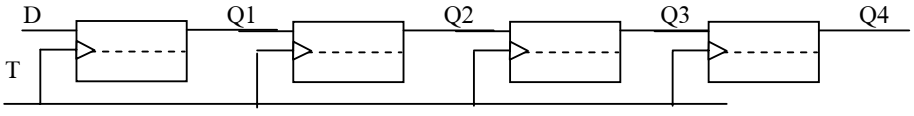
1. Chuyển tin α trong bộ nhớ thứ bi sang Mi.
2. Xóa nội dung tin chứa trong bi
3. Xóa nội dung tin α từ Mi sang Mi+1.
4. Xóa nội dung tin chứa trong Mi.

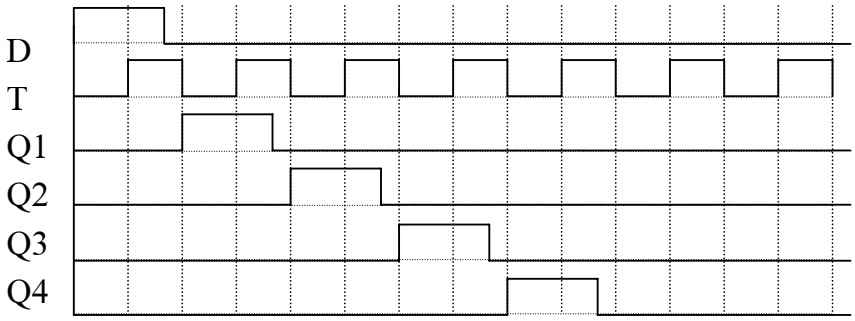
Các chu kỳ trên lại được tiếp tục. Nhờ bộ nhớ phụ Mi mà các thông tin nhị phân thực hiện được việc dịch chuyển đồng thời và không dẫm lên nhau :




*Hình 3.1 : Sơ đồ hoạt động của thanh ghi dịch*

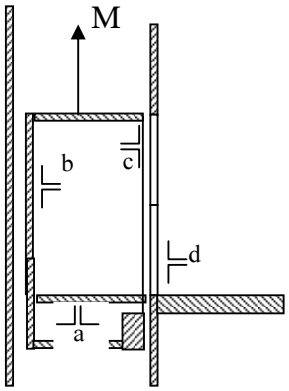
Ví dụ về bộ ghi sử dụng 4 mạch lật kiểu D với 2 trạng thái 4 bit






*3.3.6. Một số ví dụ ứng dụng*

*a. Các bài toán logic tổ hợp*




Bài toán 1. Thiết lập phương trình logic điều khiển thang máy theo nguyên tắc hoạt động như sau:

Khi cửa thang máy đóng kín ***(c =1)*** và không có người trong thang máy ***(a=0)*** thì tác động lên nút gọi thang máy ở bên ngoài ***(d)*** mới có tác dụng. Khi có người vào bên trong thang máy thì tiếp điểm ***(a)*** mới đóng ***(a=1)*** và khi đó nếu tác động vào nút gọi tầng cần đến ***(b)*** và cửa đã đóng kín ***(c=1)*** thì mới có tác dụng. Mọi tác động lên các nút ấn bên ngoài hay bên trong thang máy sẽ không có tác dụng khi cửa chưa được đóng kín.

Xác định phương trình logic trong trường hợp khi chưa có người ở trong thang máy mà nút ấn gọi tầng ***(b)*** vẫn có tác dụng và cả trong trường hợp không có tác dụng.

Ta lập ma trận Cácno

Từ ma trận ta thấy, đây là bài toán logic tổ hợp và việc giải bài toán được tiến hành theo các nguyên tắc tối thiểu hàm.

| bd \ ac | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 1 | 0 | 0 |
| 11 | 0 | 1 | 1 | 0 |
| 10 | 0 | x | 1 | 0 |

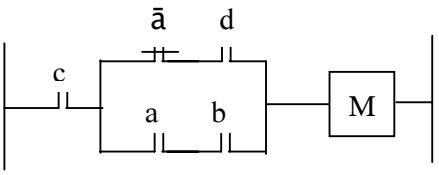
Trường hợp khi không có người trong thang máy thì nút ấn gọi tầng *b* không có tác dụng:

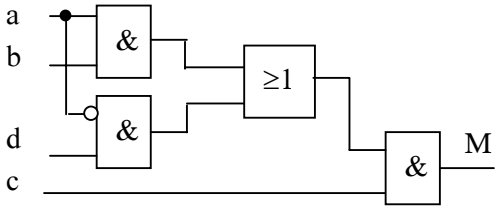
M= a.b.c.d +a.b.c.d + a.b.c.d + a.b.c.d

Tối thiểu mạch logic, ta có:

M= c(ā.d + a.b)

Ta có sơ đồ tiếp điểm và mạch số:




Trường hợp thứ 2 khi không có người trong thang máy ta ấn nút gọi tầng vẫn có tác dụng.

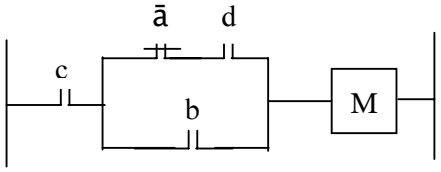
Tacó phương trình logic:

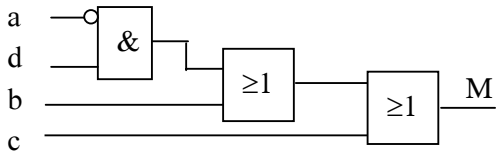
M = a.b.c.d + a.b.c.d + a.b.c.d + a.b.c.d + a.b.c.d

Sử dụng các phương pháp tối thiểu, ta có:

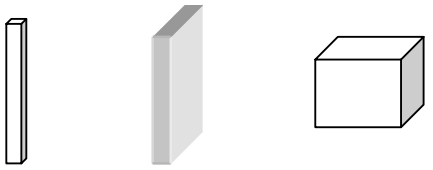
M = c(ā.d +b)

Sơ đồ tiếp điểm và sơ đồ mạch số được biểu diễn như sau:



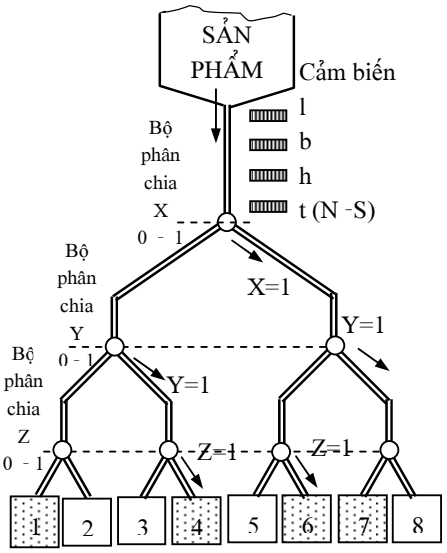
Bài toán 2. Dây chuyền phân loại tự động các sản phẩm có dạng thanh (có 1 cạnh dài và 2 cạnh ngắn), dạng tấm (có 2 cạnh dài và 1 cạnh ngắn) và dạng khối (có 3 cạnh dài). Các cạnh được bố trí các cảm biến để theo giỏi và đồng thời phát hiện các chi tiết dạng tấm có bị từ tính hoặc không.



THANH TẤM KHỐI

Hãy thiết lập các TT logic cho hệ thống.

Ta có thể bố trí hệ thống phân loại theo nguyên tắc sau đây:




| l | b | h | t | Thùng chứa | X | Y | Z |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 7 | 1 | 1 | 0 |
| 0 | 0 | 0 | 1 | 7 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 6 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 7 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 6 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 7 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 6 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 7 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 4 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 4 | 0 | 1 | 1 |

Ta có các ký hiệu sau:

- Chiều dài l: l = 1 cạnh dài l = 0 cạnh ngắn
- Chiều rộng b: b = 1 cạnh dài b = 0 cạnh ngắn
- Chiều cao h: h = 1 cạnh dài h = 0 cạnh ngắn
- Từ tính t(N -S): t =1 có từ tính t = 0 không có từ tính

Các chi tiết sau khi qua bộ cảm biến theo dõi sẽ đi qua bộ phân chia X, Y, Z và đi vào các thùng chứa tương ứng.

Giả sữ các chi tiết dạng thanh (có một cạnh dài và 2 cạnh ngắn) được chứa vào thùng chưa số 1.

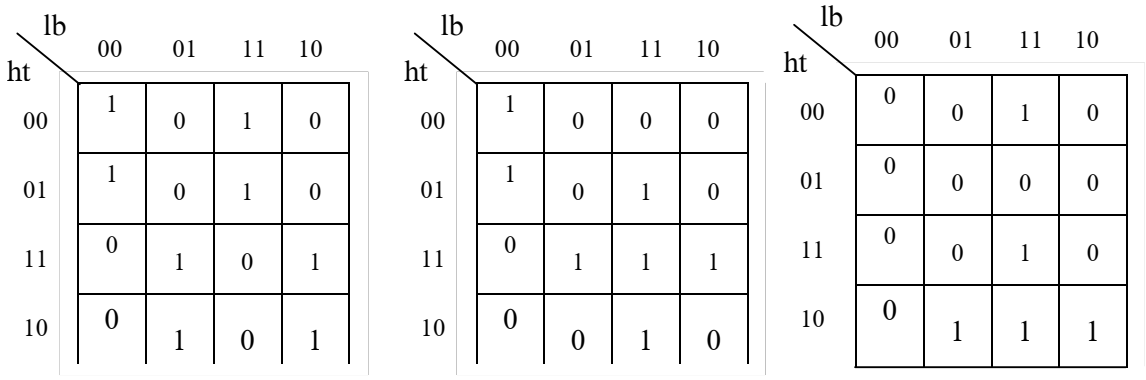
Các chi tiết dạng khối (có 3 cạnh dài) chứa vào thùng chứa số 4.

Các chi tiết dạng thanh (có 2 cạnh dài) và không có từ tính chứa vào thùng 6.

Các chi tiết còn lại sẽ chứa vào thùng số 7

Các thùng 2,3,5,8 để trống

Lập ma trận Cácno cho hàm X, Y, Z:



| ht \ lb | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 0 | 1 | 0 |
| 01 | 1 | 0 | 1 | 0 |
| 11 | 0 | 1 | 0 | 1 |
| 10 | 0 | 1 | 0 | 1 |

X

| ht \ lb | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 0 | 0 | 0 |
| 01 | 1 | 0 | 1 | 0 |
| 11 | 0 | 1 | 1 | 1 |
| 10 | 0 | 0 | 1 | 0 |

Y

| ht \ lb | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 1 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 1 | 0 |
| 10 | 0 | 1 | 1 | 1 |

Z

Tối thiểu hàm logic của X, Y, Z ta có:

$$X = \overline{l}.\overline{b}.\overline{h} + \overline{l}.b.h + l.b.\overline{h} + l.\overline{b}.h$$

$$= \overline{b}(\overline{l}.\overline{h} + l.h) + b(\overline{l}.h + l.\overline{h}) = \overline{b}\overline{(l \oplus h)} + b(l \oplus h) = \overline{b \oplus (l \oplus h)}$$

$$Y = l.b.h + b.h.t + l.b.t + \overline{l}.\overline{b}.\overline{h} + l.b.t$$

$$= l.b\,(t+h) + h.t\,(\overline{b+l}) + \overline{l}.b.h$$

$$Z = l.b.\overline{t} + l.b.h + l.h.\overline{t} + b.h.\overline{t} = l.b(\overline{t}+h) + h.\overline{t}(l+b)$$

Biểu diễn mạch tiếp điểm hay mạch số các hàm X, Y,Z.

*b.Các bài toán logic tuần tự*

Bài toán 1: Có một máy cắt tole điều khiển tự động theo nguyên tắc sau:

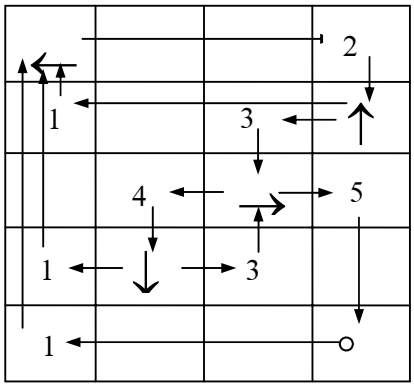
Ở trạng thái ban đầu thì đầu cắt ở trạng thái treo (ở vị trí trên cùng). Khi tác động vào nút ấn m thì ly hợp điện từ E sẽ đóng và truyền chuyển động quay sang trục khuỷu và làm cho đầu mang dao cắt chuyển động đi xuống. Trên trục khuỷu, người ta gắn một cam và nó quay đồng thời với trục. Khi trục mang cam quay được hơn một nửa vòng thì cam trên trục tác động lên công tắc b và trục tiếp tục quay để nâng đầu dao lên trên mặc cho lúc đó m đã trở về 0 (m=0). Khi cam quay đến vị trí ban đầu thì công tắc b trở về 0 (b=0) và trục dừng lại ở điểm trên cùng.

Nếu trong khi trục quay mà cam chưa tác động vào công tắc b và m đã trở về 0 (m=0) thì trục sẽ dừng lại. Để cho trục tiếp tục quay thì cần tác dụng vào m.

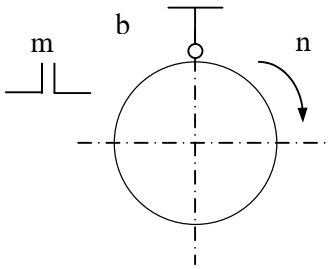
Trục vẫn dừng lại tại ví trí trên cùng khi b chuyển từ 1 sang 0 mặc cho m khi đó vẫn bằng 1.

*Giải theo phương pháp biến trạng thái*

mb 00 01 11 10 E




| E |
|---|
| 0 |
| 1 |
| 1 |
| 1 |
| 0 |




Ma trận rút gọn:

| | | | |
|---|---|---|---|
| ← | 4 | 3 | ↑ |
| 1 | ↓ | → | ○ |

| mb | 00 | 01 | 11 | 10 | X |
|---|---|---|---|---|---|
| x 0 | ◎ | 1 | 1 | ◎ | |
| 1 | 0. | ← | ← | ← | |

| mb | 00 | 01 | 11 | 10 | E |
|---|---|---|---|---|---|
| x 0 | ◎ | 1 | 1 | ← | |
| 1 | 0. | ← | ← | ◎ | |

Tối thiểu, ta có : $X = b + m.x$ $E = b + m.\overline{x}$

*Giải theo phương pháp trực tiếp sử dụng trực tiếp ma trận Karnaugh:*

| m | 0 | 1 |
|---|---|---|
| b 0 | | |
| 1 | | |

| m | 0 | 1 | 1 | 0 | X |
|---|---|---|---|---|---|
| b 0 | | | | | |
| 1 | | | | | |

$X = b + m.x$

| m | 0 | 1 | 1 | 0 | E |
|---|---|---|---|---|---|
| b 0 | | | | | |
| 1 | | | | | |

$E = b + m.\overline{x}$

Trường hợp thứ 2:

| m | 0 | 1 |
|---|---|---|
| b 0 | | |
| 1 | | |

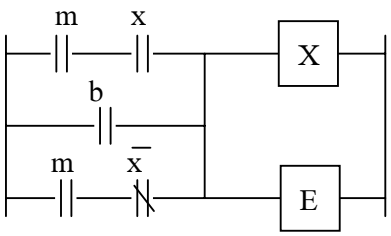
| m | 0 | 1 | 1 | 0 | X |
|---|---|---|---|---|---|
| b 0 | | | | | |
| 1 | | | | | |

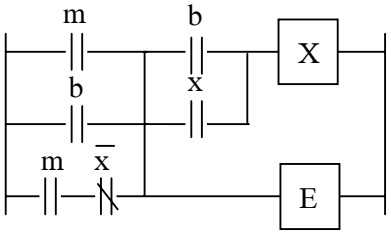
$X = m(b + x)$

| m | 0 | 1 | 1 | 0 | E |
|---|---|---|---|---|---|
| b 0 | | | | | |
| 1 | | | | | |

$E = b + m.\overline{x}$

Sơ đồ tiếp điểm



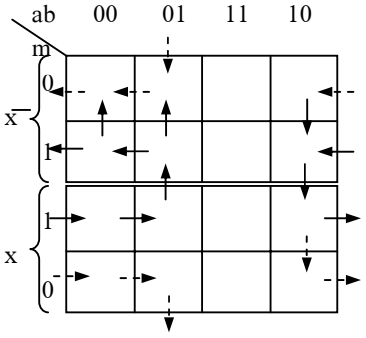

Bài toán 2: Có một xi lanh thủy lực được điều khiển bằng van điện E. Ở trạng thái nghỉ, ta có a =1; b= 0; m=0 và E=0.

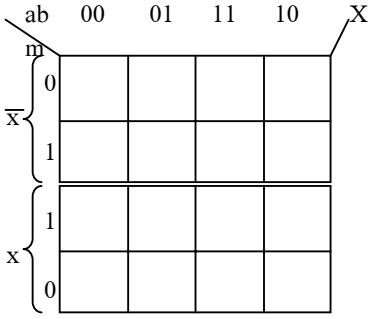
Đóng tiếp điểm m thì E=1 và làm cho Piston chuyển động sang phải (a=0) và cuối cùng sẽ tác động lên b (b=1), khi b=1 thì sẽ tác động đến E vaE=0 sẽ làm cho Piston chuyển động về bên trái và cuối hành trình tác động vào a (a=1). Nếu khi đó m vẫn đang đóng (m=1) thì quá trình chuyển động của Piston lại tiếp tục.

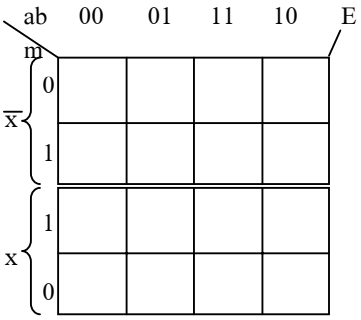
Nếu tiếp điểm m đã mở (m=0) trước khi Piston chuyển động sang trái và tác động lên a thì Piston sẽ dừng lại và kết thúc chu trình.

Chú ý: Tiếp điểm m có thể mở (m=0) trước khi Piston chuyển động sang phải và tác động vào b.

Giải:

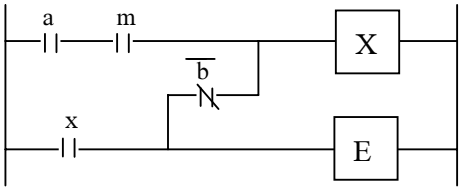



$X = a.m + x.\overline{b}$ $E = x$

Sơ đồ tiếp điểm



# IV. GRAFCET- CÔNG CỤ MÔ TẢ MẠCH LOGÍC TUẦN TỰ

## 4.1. Khái niệm về GRAFCET:

*GRAFCE* viết tắt của các dòng chữ bằng tiếng Pháp : *Graphe Fonctionel de Commande Etapes - Trasition* (chuỗi chức năng điều khiển các giai đoạn - chuyển tiếp) do 2 cơ quan AFCET (liên hiệp Pháp về tin học kinh tế và kỹ thuật) và ADEPA (tổ chức nhà nước về sự phát triển nền sản xuất tự động hóa) hợp tác soạn thảo ra vào tháng 11/1982 và đã đăng ký ở tổ chức tiêu chuẩn hóa Pháp dưới mã hiệu NFCO 3190.

Trên cơ sổ này, một tiêu chuẩn CEI đã được soạn thảo.

*4.1.1. Hoạt động theo trình tự logic của thiết bị công nghiệp*

Khi thiết kế một hệ thống sản xuất tự động hay khi vận hành, khai thác hoặc bảo trì, sữa chữa thì điều quan trọng nhất là cần phải nắm được chức năng hoạt động chung của hệ thống và hiểu kỹ được tính năng của từng phần tử trong hệ thống cũng như một số vấn đề liên quan đến trình tự hoạt động của các phần tử đó.

Có thể có một cách mô tả chúng bằng lời, nói cách khác là người ta sẽ liệt kê sự hoạt động của hệ thống và chú dẫn về sự hoạt động của các phần tử theo trình tự của thời gian. Nếu người viết diễn đạt được một cách rõ ràng và súc tích thì người vận hành, bảo trì ... có thể nắm bắt được một cách dễ dàng đối với những hệ thống không có quá nhiều giai đoạn hay quá nhiều phần tử. Còn nếu hệ thống phức tạp thì rõ ràng điều này sẽ rất khó có thể hình dung được.

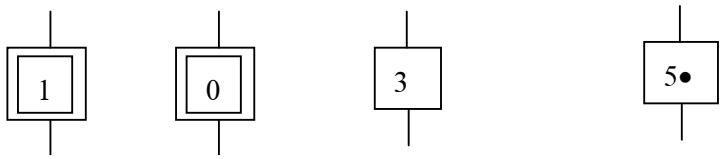
Ngôn ngữ *GRAFCET* đã hỗ trợ cho chúng ta rất nhiều trong việc mô tả sự hoạt động của các thiết bị và các phần tử của hệ thống một cách rõ ràng và chặt chẽ.

*4.1.2. Một số ký hiệu được sử dụng trong GRAFCET*:

*a. Trạng thái* : là biểu thị một sự hoạt động nào đó của phần tử điều khiển và trong 1 trạng thái, các hành vi điều khiển là không thay đổi. Một trạng thái có thể là hoạt động hay không hoạt động và điều khiển là thực hiện các mệnh đề logic chứa các biến vào và các biến ra để hệ thống có được 1 trạng thái ổn định trong hệ.

Trạng thái ban đầu : 2 ô vuông lồng vào nhau và có đánh số 0 hay 1.

Trạng thái : 1 ô vuông và có đánh số. Nếu là trạng thái đang hoạt động có thêm dấu "." bên cạnh.

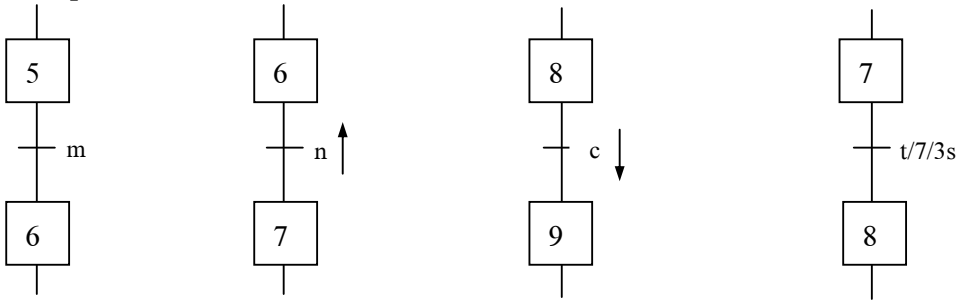



Trạng thái ban đầu  Trạng thái  Trạng thái đang thực hiện

*b. Chuyển tiếp:*

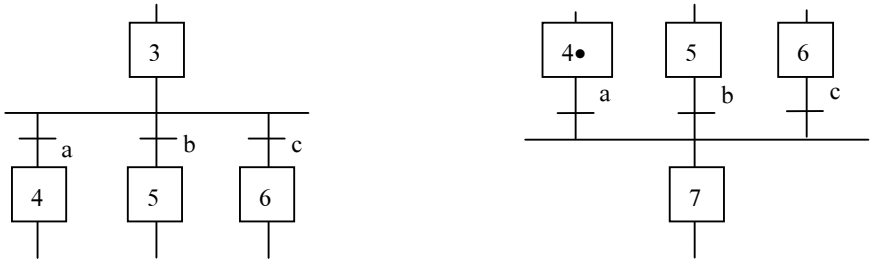
Việc chuyển tiếp từ trạng thái này sang trạng thái khác được thực hiện khi các điều kiện được thỏa mãn. Nó được biểu diễn bằng tuyến thực hiện và 1biến hay một số biến được biểu diễn bằng nét ngang ở giữa 2 trạng thái.

Bình thường việc chuyển từ trạng thái này sang trạng thái khác được thực hiện theo trình tự từ trên xuống qua các trạng thái khi các chuyển tiếp giữa chúng được thõa mãn. Tuy nhiên việc chuyển trạng thài này chỉ được thực hiện khi trạng thái trước đó đã kết thúc và nó sẽ được thực hiện ở phía sườn tăng hay giảm của biến chuyển tiếp hay sau 1 số giây kể từ khi tác động cuối cùng của trạng thái trước đó được thực hiện. Việc lựa chọn chu trình hoạt động trong nhiều chu trình được thực hiện bằng các chuyển tiếp khi phân nhánh. Có thể có loại phân nhánh đơn và phân nhánh kép.



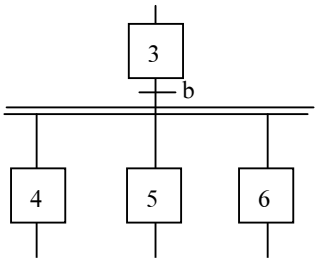

Thay đổi ở sườn tăng  Thay đổi ở sườn giảm  Sau 3s từ khi TT thứ 7 kết thúc

*c. Điều kiện phân nhánh.*

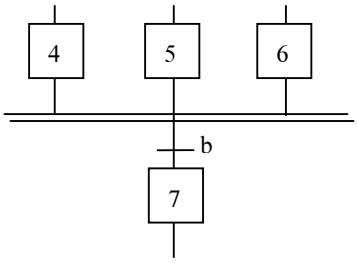



Trạng thái 3 sẽ chuyển sang TT 4 khi điều kiện *a* thỏa mãn

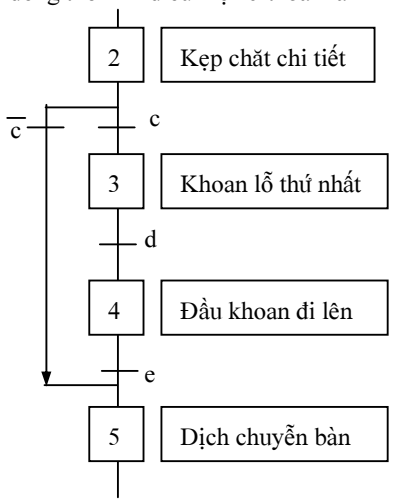
Trạng thái 4 đang thực hiện và sẽ chuyển sang TT 7 khi điều kiện *a* thỏa mãn



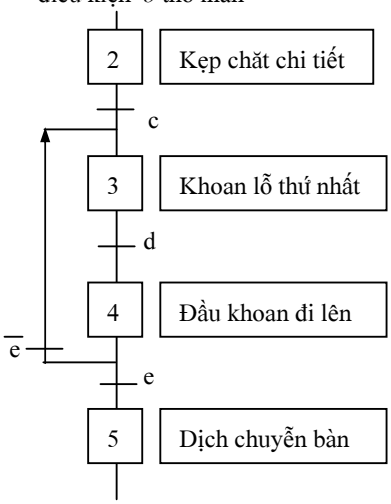

Trạng thái 3 sẽ chuyển sang TT 4,5,6 đồng thời khi điều kiện *b* thỏa mãn




Trạng thái 4,5,6 sẽ chuyển sang TT 7 khi điều kiện *b* thỏ mãn




Nếu $\overline{c}$ thõa mãn thì chu trình sẽ được thực hiện theo 2-5 (bỏ qua các giai đoạn 3,4). Ngược lại nếu *c* thỏa mãn thì chu trình sẽ thực hiện theo 2,3,4,5




Nếu $\overline{e}$ thõa mãn thì chu trình sẽ được thực hiện theo 2,3,4,3,4 (trở lại giai đoạn 3,4.) Ngược lại nếu *e* thỏa mãn thì chu trình sẽ thực hiện theo 2,3,4,5

*4.1.3. Ví dụ về ứng dụng Grafcet.*

Bài toán 1: Khoan 2 lỗ trên chi tiết.

Chi tiết gia công được gá lên đồ gá. Nếu tác đông lên "m" (m=1) thì piston N sẽ tiến hành kẹp chặt chi tiết. Sự kẹp chặt sẽ được kết thúc cho đến khi cảm biến "a" tác động : a=1.

Khi a=1, $M_1$ sẽ quay và đầu khoan đi xuống (piston D đi xuống). Khi đạt được chiều sâu khoan sẽ tác động lên d và d=1. Sau 1 thời gian dừng D để làm bóng lỗ (t=2s) đầu khoan đi lên và kết thúc hành trình khi tác động vào e (e=1).

Khi e=1, P đẩy đồ gá về phía trước cho đến khi chạm b (b=1). Khi b=1, D lại đi xuống và chu trình lại được tiếp tục để khoan lỗ 2 cho đến khi e=1.

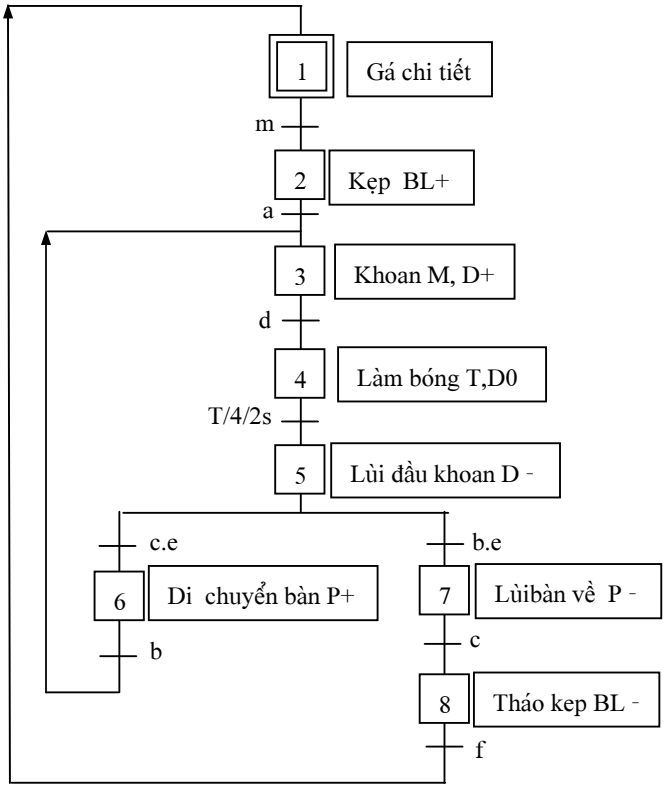
Khi e=1, P lại đẩy đồ gá lùi về cho đến khi chạm c (c=1). Khi c=1, N lùi về để tháo chi tiết cho đến khi tác động f (f=1) thì kết thúc chu trình. $M_1$ sẽ dừng và để tháo lắp chi tiết gia công.

Và chu trình sẽ lại được tiếp tục nếu ta lại tác động vào "m".

Chú ý : Sau giai đoạn 5, chương trình có thể sẽ chuyển sang giai đoạn 6 hay giai đoạn 7 tùy thuộc vào điều kiện chuyển tiếp b, c, e được thõa mãn.

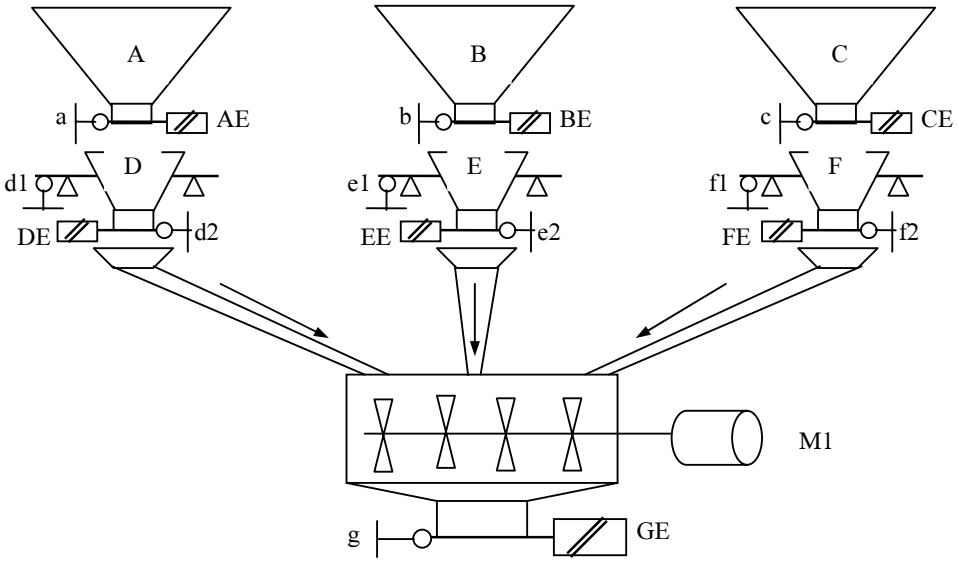
Nếu b. e=1, có nghĩa là b=1 và e=1. Điều đó tương ứng với quá trình khoan xong lỗ thứ 2 và trạng thái 5 sẽ chuyển sang 7 va` 8 rồi kết thúc chu trình.

Nếu c.e=1 thì có nghĩa nó vừa khoan xong lỗ 1 và cần phải dịch chuyển bàn đến vị trí số 2, tức là cho đến khi b=1.

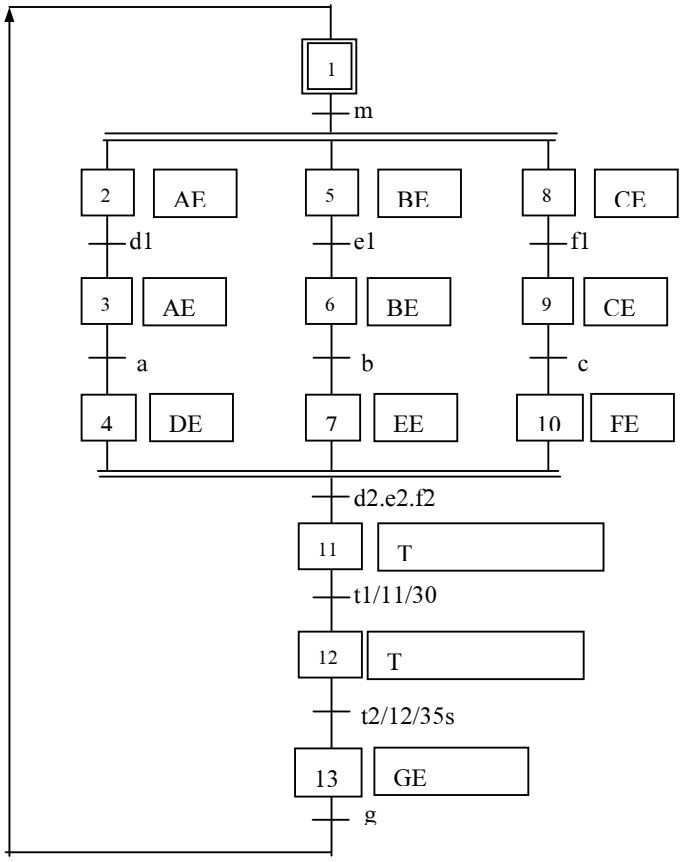



Bài toán 2: Hệ thống định lượng, phối liệu và trộn tự động:

Có 3 phểu chứa 3 loại bột khác nhau A, B, C. Các phểu định lượng tương ứng với 3 loại bột đó. Khi đủ khối lượng theo yêu cầu nó sẽ tác động lên các cảm biến $d_1$, $e_1$, $f_1$




Khi các cảm biến này tác động (=1), tương ứng có các van điện AE, BE, CE đóng cửa phễu chứa tương ứng A, B, C.




Khi van phễu chứa đóng sẽ tác động lên các cảm biến tương ứng a, b, c và các cảm biến này sẽ tác động đến van điện DE, EE và FE tương ứng để mở cửa phểu định lượng D, E, F cho liệu xuống thùng khuấy G. Khi mở cửa các phểu định lượng sẽ tác động đến $d_2$, $e_2$, $d_2$ và sẽ đóng điện động cơ khuấy M (khi cả 3 cảm biến đều tác động). Động cơ M sẽ quay và sau khoảng thời gian $t_1$ để trộn bột đều, sẽ chuyển sang trạng thái ra liệu bằng cách tác động vào van điện GE để mở cửa ra

liệu. Động cơ M tiếp tục quay tiếp trong khoảng thời gian $t_2$ để đẩy hết liệu ra ngoài rồi sau đó van GE sẽ đóng lại và đóng cửa thùng trộn. Khi đóng, nó tác động lên g và  chu trình trộn kết thúc.

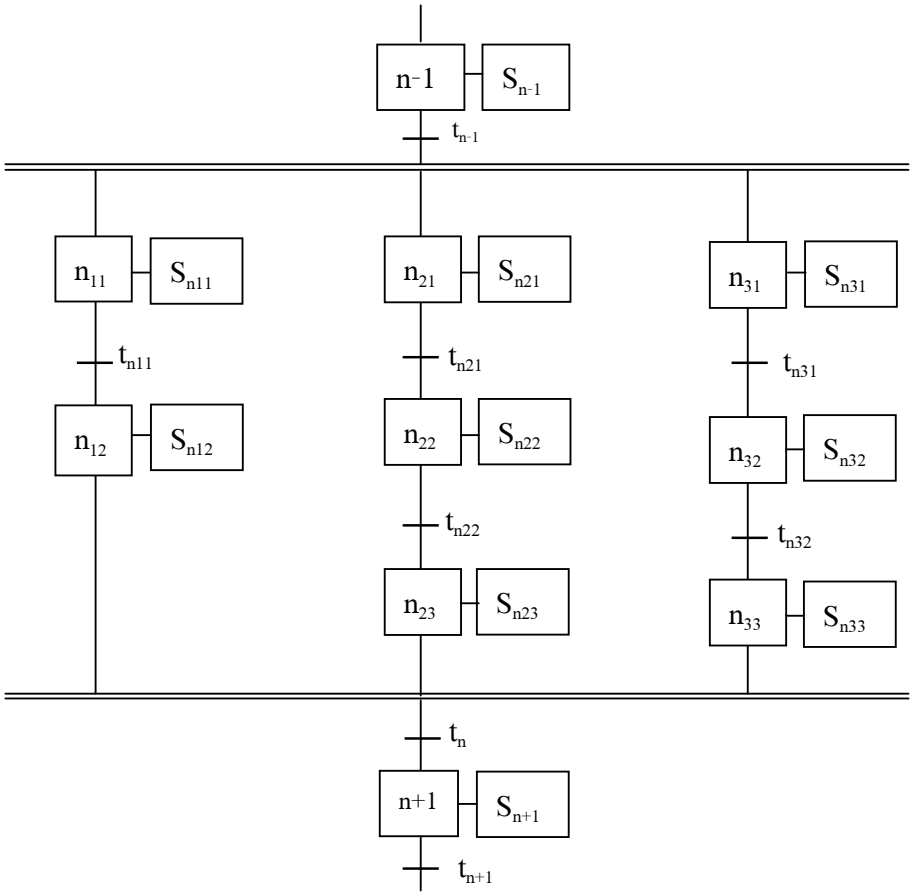
Muốn tiếp tục, ta khởi động lại chu trình bằng cách bấm "m".

*4.1.4. Điều kiện phân nhánh*

*a. Phân nhánh đồng thời (vô điều kiện)*

Cho một đoạn GRAFCET biểu diễn quá trình hoạt động của một hệ thống công nghiệp theo nguyê tắc sau: Trong quá trình hệ thống đang thực hiện trạng thái hoạt động n-1 với lệnh thực hiện là $S_{n-1}$ và nếu có chuyển tiếp $t_{n-1}$ khả tiếp sẽ chuyển sang thực hiện trạng thái $n_{11}$, $n_{21}$,$n_{31}$ đồng thời.

Trên mỗi nhánh độc lập sẽ thực hiện lần lượt các trạng thái trong nhánh đó cho đến trạng thái cuối cùng của từng nhánh. Sau đó nhập về làm một nhờ chuyển tiếp $t_n$ để tới trạng thái chung n+1.



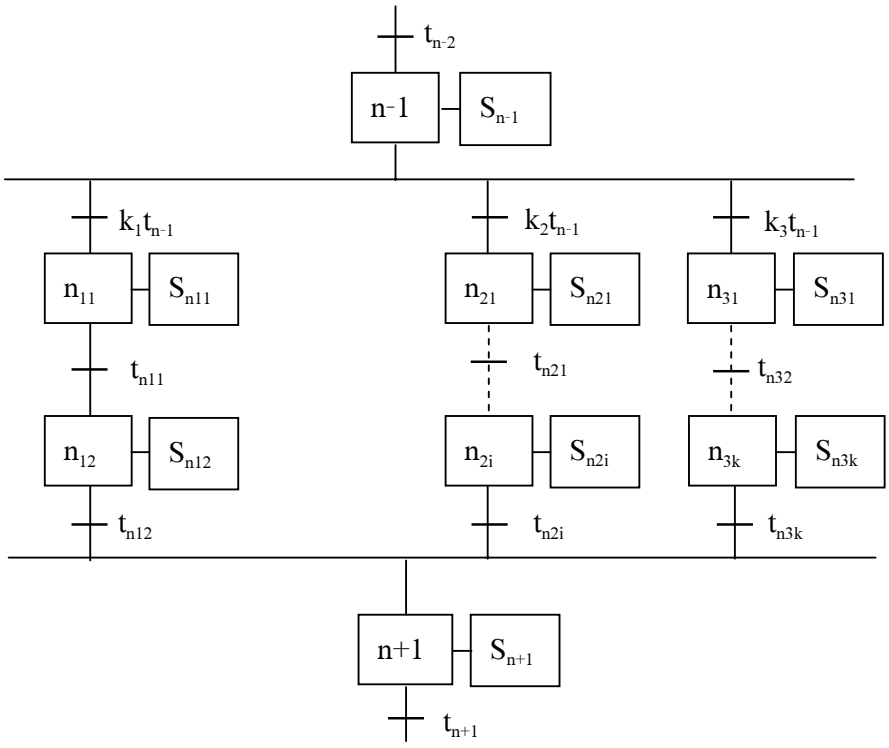
Để mô tả trường hợp trên, chúng ta có sơ đồ GRAFCET trên. Trong sơ đồ này cần chú ý là khi rẽ thành các nhánh đều từ một chuyển tiếp chung là $t_{n-1}$.

Khi nhập về làm một phải có điều kiện là tất cả nhánh đã thực hiện xong trạng thái cuối của mình, cho dù trạng thái cuối của từng nhánh có thể tới không cùng lúc (tùy thuộc vào số lượng các trạng thái của mỗi nhánh) rồi mới qua chuyển tiếp $t_n$ để về chung trạng thái kế là n+1.

Như vậy, $t_{n-1}$ là chuyển tiếp để rẽ các nhánh và $t_n$ là chuyển tiếp để các nhánh nhập về làm một, và đều nằm ở ngoài phạm vi các nhánh (trên sơ đồ, chúng ở ngoài miền giới hạn bởi các đường gạch kép).

*b. Trường hợp rẽ nhánh lựa chọn (có điều kiện)*

Khác với trên, các nhánh được rẽ đồng thời với cùng một chuyển tiếp $t_{n-1}$, còn ở đây việc rẽ vào nhánh nào được chọn lựa theo điều kiện tương ứng bằng các chuyển tiếp riêng rẽ cho từng nhánh.



Sơ đồ cho thấy muốn rẽ vào nhánh nào thì phải có điều kiện tương ứng, thể hiện bằng các chuyển tiếp ở đầu mỗi nhánh. Chúng điều khiển việc rẽ vào nhánh mong muốn. Từ một nhánh, khi trở về dòng chính sau khi đã thực hiện xong trạng thái cuối cùng của mình thì dùng chuyển tiếp ở cuối mỗi nhánh. Như vậy, chuyển tiếp để rẽ vào từng nhánh và trở về dòng chính đều ở trong phạm vi của nhánh. Trên sơ đồ, chúng đều ở trong miền giới hạn bởi các đường gạch đơn. Ở đây, dùng vạch đơn với ý không rẽ song song đồng thời vào các nhánh mà chỉ rẽ vào một trong các nhánh tùy điều kiện chọn lựa, đó là các $k_i$.

*4.1.5. Điều khiển hệ thống*

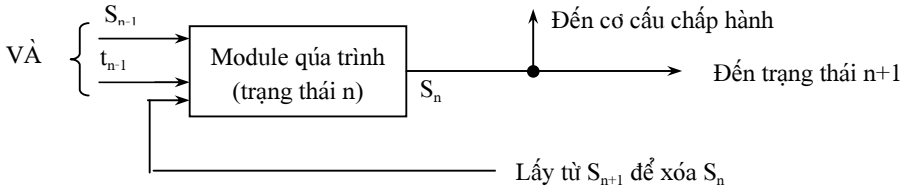
Sau khi mô tả hoạt động của hệ thống bằng sơ đồ GRAFCET, ta thấy rõ việc điều khiển hệ thống là thực hiện lần lượt các trạng thái, mà thực chất là tạo ra các lệnh điều hành si ở từng trạng thái, lần lượt từ trạng thái này đến trạng thái kế tiếp.

Sơ đồ GRAFCET còn cho thấy rõ một trạng thái n muốn thực hiện , xuất ra lệnh điều hành sn, thì phải có 2 điều kiện bắt buộc là :

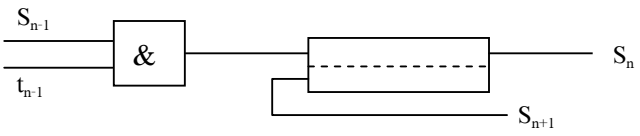
- Có trạng thái n-1 với lệnh sn-1.
- Có chuyển tiếp tn-1 từ trạng thái n-1 sang trạng thái n.

Khi có sn, điều khiển thực hiện trạng thái n, thì tới phiên mình sn sẽ cùng với chuyển tiếp tn, tạo ra sn+1 để thực hiện trạng thái n+1. Lệnh điều hành sn+1 đồng thời còn dùng để xóa lệnh điều hành sn của trạng thái n kế trước.

Hình dưới mô tả cách thức điều khiển đó cho trạng thái n, gọi là một mođuyn quá trình.

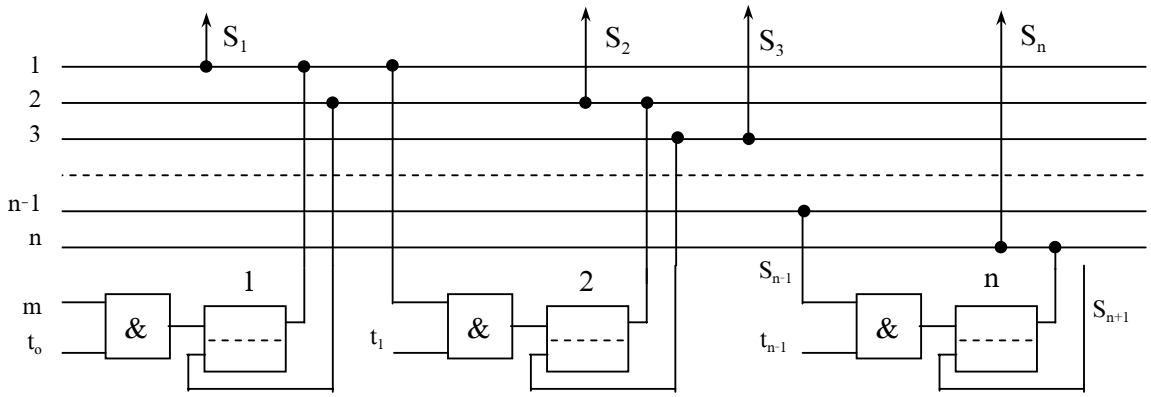



Ứng dụng các phần từ logic cơ bản ta dễ dàng thực hiện một mođuyn như thế.




Module n

Một hệ thống gồm nhiều trạng thái sẽ được điều khiển bởi một dãy nối tiếp các mođuyn qúa trình. Dưới đây là một cách mô tả mạch điều khiển hệ thống n trạng thái theo phương pháp GRAFCET bởi một dãy nối tiếp của các moduyn.




Một hệ thống các đường nằm ngang, đánh số từ 1 đến n, mô tả đường tín hiệu các đầu ra tương ứng S1 đến Sn của các trạng thái 1 đến n. Phía trên các đường này là các lệnh Si hướng về bộ phận chấp hành để thực hiện các trạng thái, còn phía dưới mô tả các mođuyn quá trình, được đánh số từ 1 đến n, ứng với các trạng thái 1 đến n.

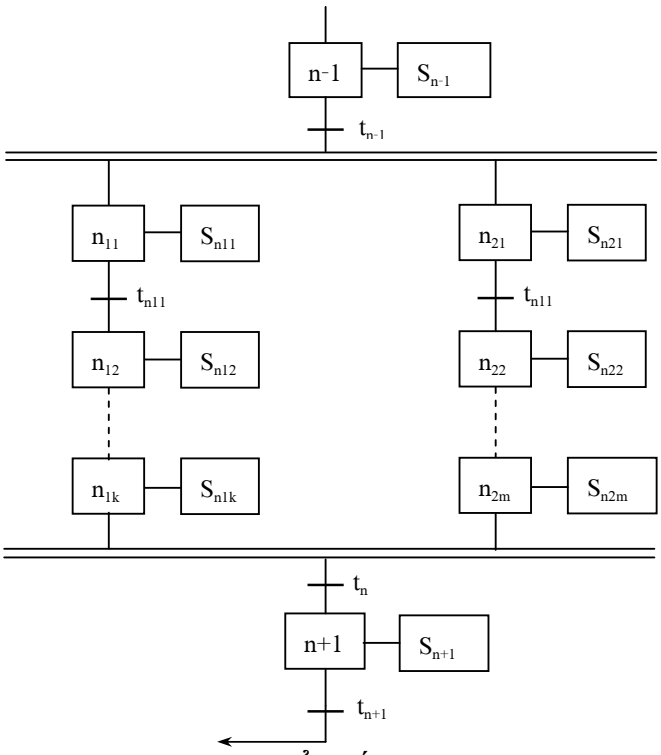
Hoạt động của các mođun như sau :

Tín hiệu khởi động m sẽ cùng với chuyển tiếp to là chuyển tiếp từ trạng thái nghĩ sang trạng thái 1, đi qua phần tử VÀ, kích thích bộ nhớ xuất ra tín hiệu gởi về đường tín hiệu 1 để đưa ra lệnh chấp hành S1.

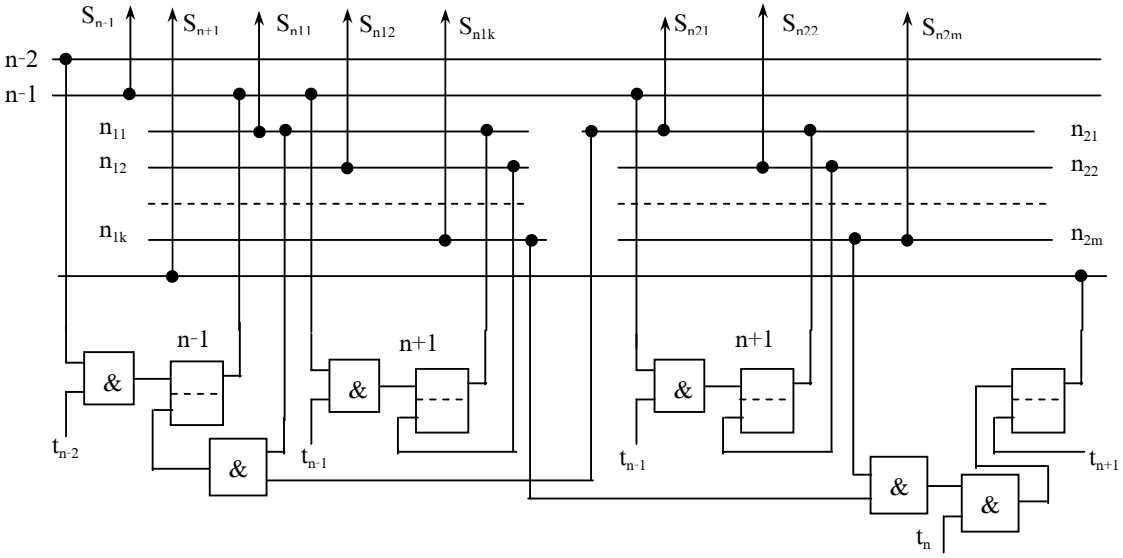
Khi đã có S1, nó sẽ cùng với chuyển tiếp t1 (từ trạng thái 1 --> 2) đi qua phần tử VÀ ở mođuyn thứ 2, xuất ra tín hiệu gởi về đường tín hiệu 2. Từ đó, một mặt đưa ra lệnh chấp hành S2, mặt khác quay lại mođuyn 1 để xóa tín hiệu S1 ở bộ nhớ 1. Tương tự S2 sẽ được xóa bởi S3 đấy từ đường tín hiệu 3.

Tại mođuyn cuối cùng, Sn-1 cùng với tn-1 qua phần tử VÀ kích thích bộ nhớ, xuất ra tín hiệu về đường tín hiệu n để đưa ra lệnh chấp hành Sn. Trạng thái n là trạng thái cuối, do đó không có lệnh Sn+1 để xóa Sn mà dùng trực tiếp tn - là chuyển tiếp từ trạng thái n trở về trạng thái nghĩ.

*a. Trường hợp rẽ nhánh song song*




Từ trạng thái n-1 qua chuyển tiếp $t_{n-1}$ rẽ thành 2 nhánh hoạt động riêng rẽ. Nhánh thứ nhất có k trạng thái từ n(11) đến n(1k), nhánh thứ 2 có m trạng thái từ n(21) đến n(2m). Sau khi mỗi nhánh thực hiện xong trạng thái cuối của mình thì qua chuyển tiếp tn để nhập lại về trạng thái n+1 rồi trở về trạng thái nghĩ qua chuyển



tiếp tn+1. Với trường hợp này ta có mạch điều khiển sau : Điều cần quan tâm ở đây là việc điều khiển khi rẽ nhánh và khi nhập về.

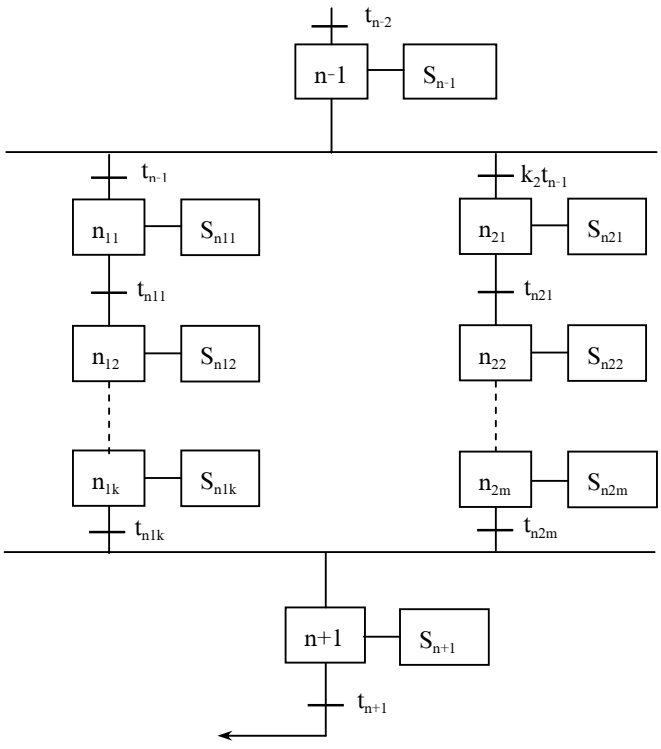
Tại chỗ rẽ nhánh, ta dùng tín hiệu $s_{n-1}$ cùng với chuyển tiếp $t_{n-1}$ để rẽ nhánh, qua phần tử VÀ của hai mođuyn n(11), n(21) rồi vào bộ nhớ tạo ra $S_{n(11)}$ và $S_{n(21)}$. Tiếp ngay sau đó ta dùng chúng qua phần tử VÀ về xóa $S_{n-1}$ tại bộ nhớ thuộc mođuyn n-1. Có nghĩa là sau khi xuất hiện trạng thái đầu của cả hai nhánh mới xóa lệnh $S_{n-1}$ của trạng thái kế trước.

Tại chỗ nhập từ các nhánh n+1, ta cùng đồng thời cả hai trạng thái cuối $S_{n(1K)}$ , $S_{n(2m)}$ của hai nhánh, qua phần tử VÀ rồi cùng với chuyển tiếp tn qua một phần tử VÀ khác, kích thích bộ nhớ xuất ra $S_{n+1}$ ở trạng thái n+1. Nghĩa là chỉ sau khi cả 2 nhánh đã thực hiện xong trạng thái cuối thì mới chuyển tiếp đến n+1 thông qua chuyển tiếp $t_n$.

Để xóa $S_{n+1}$ nhằm trở về trạng thái nghĩ, ta dùng trực tiếp chuyển tiếp $t_{n+1}$.

*b. Trường hợp rẽ nhánh lựa chọn*

Chẳng hạn bắt đầu từ trạng thái n-1 rẽ nhánh : - khi có $k1t_{n-1}$ : sang nhánh 1 : -khi có $k_2t_{n-1}$ sang nhánh 2 : rồi về trạng thái n+1 ở dòng chính và trở lại trạng thái nghĩ qua chuyển tiếp $t_{n+1}$. Mạch điều khiển hệ thống này được trình bày dưới đây, chỉ khác trường hợp trước ở chỗ rẽ nhánh và nhập về. Muốn rẽ sang nhánh nào thì dùng chuyển tiếp $kit_{n-1}$ của nhánh đó bằng cách dùng tín hiệu chọn là $k1.t_{1-n}$ hoặc k2. $t_{n=1}$ (logic VÀ) cùng với lệnh $S_{n-1}$ thông qua phần tử VÀ ở từng nhánh để vào bộ nhớ và xuất ra $S_{n(11)}$ hoặc $S_{n(21)}$ tùy theo điều kiện chọn.

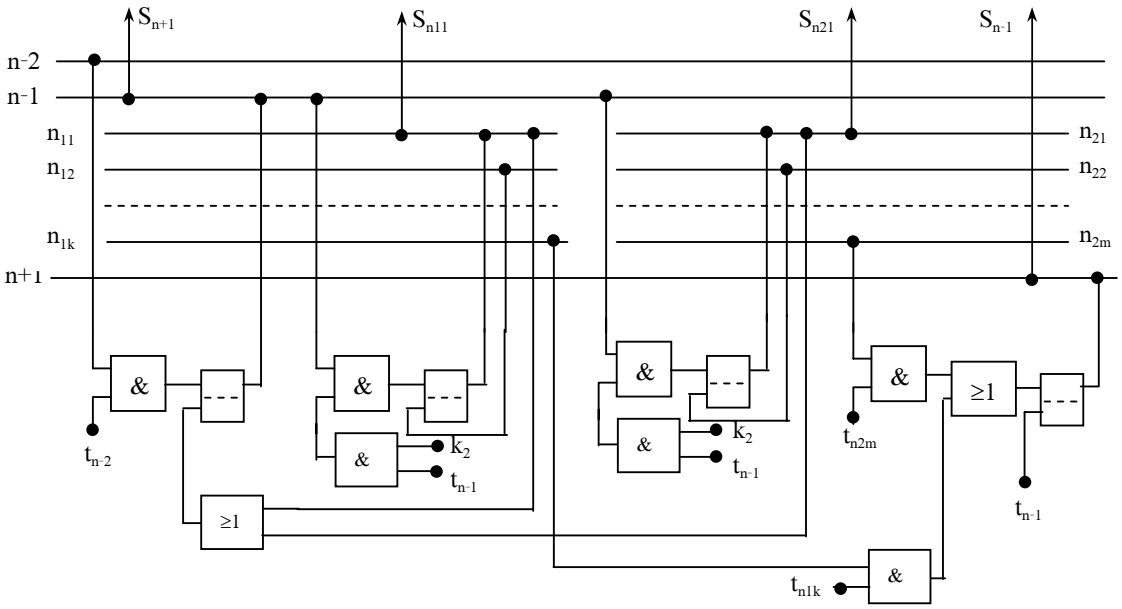



Để xóa lệnh $S_{n-1}$ sau khi rẽ nhánh, mà ở đây có thể là nhánh 1 hoặc nhánh 2, cho nên phải dùng cách đưa $S_{n(11)}$ , $S_{n(21)}$ thông qua phần tử HOẶC để đưa về xóa $S_{n-1}$.

Khi từ nhánh nhập về dòng chính, tức là về trạng thái n+1 phải tính đến khả năng có thể là từ nhánh 1 hoặc từ nhánh 2.

Nếu từ nhánh 1, thì phải

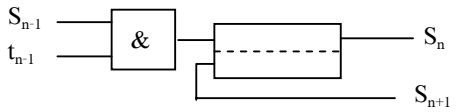
dùng lệnh điều hành $S_{n(1k)}$ của trạng thái cuối nhánh 1, cùng với chuyển tiếp $t_{n(1k)}$ thông qua phẩn tử VÀ . Nếu từ nhánh 2 nhập về thì dùng $S_{n(2m)}$ và $t_{n(2m)}$ thông qua phẩn tử VÀ thứ 2 . Đầu ra của 2 phẩn tử VÀ này đi vào phần tử HOẶC để kích thích bộ nhớ, xuất ra lệnh $S_{n+1}$ trở về dòng chính.




*4.1.6. Điều khiển hệ thống bằng sổ ghi tuần tự*

Theo phương pháp GRAFCET, một hệ thống tự động tuần tự được mô tả bằng một dãy nối tiếp của các trạng thái. Mỗi trạng thái được mô tả bằng một mođuyn và có thể dùng các phần tử logic có bản để điều khiển.

Chẳng hạn để điều khiển một trạng thái n ta dùng một mođuyn có sơ đồ sau :

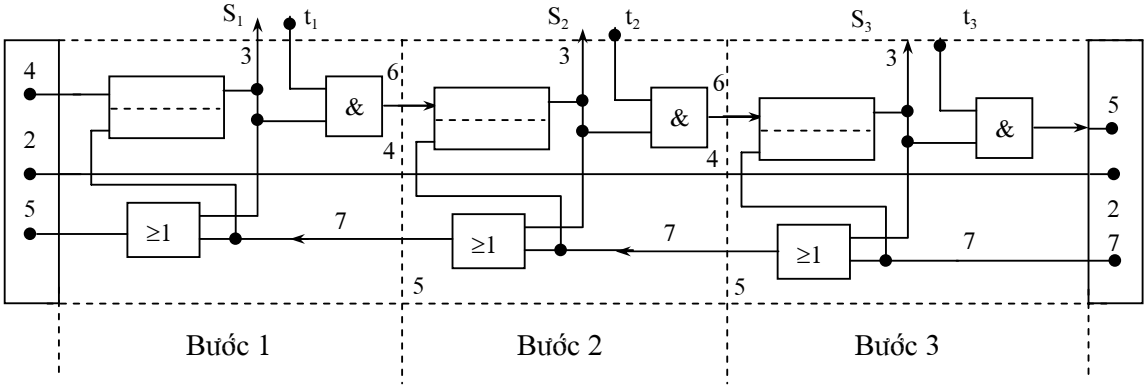



Tạo ra lệnh điều khiển $S_n$ của trạng thái n bằng logic VÀ của $S_{n+1}$ và $t_{n-1}$. Sau đó, khi bước sang trạng thái sau là n+1, thì dùng lệnh $S_{n+1}$ trở về xóa $S_n$.

Bằng cách đó, nhiều hãng trên thế giới đã chế tạo sổ ghi tuần tự thực hiện việc điều khiển các trạng thái của hệ thống.

Dưới đây là ví dụ sơ đồ logic của sổ ghi :

Ở đây trình bày 3 sổ ghi nối tiếp để điều khiển 3 trạng thái theo các bước 1, 2, 3. Ở đầu và cuối của cụm 3 sổ ghi có chỗ nối vào ra giao tiếp với bên ngoài. Mỗi sổ ghi có các phần tử VÀ, HOẶC, NHỚ, (ưu tiên xóa, dù ở cửa 4 có tín hiệu), hoạt động như sau :



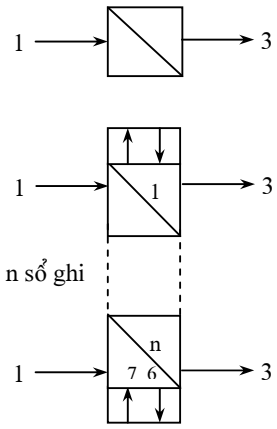

- Lệnh khởi động quá trình đi vào cửa 4, kích thích bộ nhớ của sổ ghi đầu tiên cho ra lệnh điều khiển s1, ở cửa 3 để thực hiện trạng thái 1, đồng thời dẫn đến một trong hai đầu vào của phần tử VÀ rồi chờ ở đó.

- Khi có tín hiệu chuyểne tiếp $t_1$ đi vào cửa 1, phần tử VÀ, do đã có S1 chờ sẵn sẽ xuất tín hiệu ra cửa 6, đi vào cửa 4 của sổ ghi kế sau và kích thích bộ nhớ cho ra lệnh điều khiển $S_2$ và nó được dẫn tới phần tử VÀ để chờ khi có chuyển tiếp $r_2$ sẽ xuất từ hiệu vào cửa 4 của sổ ghi tiếp theo.

- Tín hiệu S2 còn quay về phần tử HOẶC, xuất ra ở cửa 5, đi vào cửa 7 của sổ ghi kế trước rồi về xóa $S_1$ . Để xóa $S_2$ thì dùng $S_3$ theo cách tương tự.

- Để xóa $S_3$ ở sổ ghi cuối cùng, ta dùng chuyển tiếp $t_3$ đi vào phần tử VÀ, xuất ra cửa 6 rồi nối mạch về cửa 7.

- Cửa số 2 dùng để nối với nguồn mới cho các bộ nhớ.




Thời gian đáp ứng của sổ ghi khoảng 6 ở áp làm việc 4 bar. Sổ ghi tuần tự được chế tạo thành từng khối và có thể ghép với nhau thành một dãy kế tiếp mà không cần có đường nối ngoài. Sổ ghi tuần tự được lý hiệu đơn giản như hình sau :

Trường hợp chỉ có 1 sổ ghi thì 1 là ký hiệu cửa vào số 1 của chuyển tiếp, và 3 là cửa ra số 3 của lệnh điều khiển si ở từng sổ ghi.

Trường hợp một sãy sổ ghi thì có thêm số thứ tự cho các sổ ghi 1,2 ... n và hai đầu giao tiếp. Cửa 2 nối nguồn còn cửa 4 là để khởi động quá trình. Khi sử dụng sổ ghi, có 2 cửa quan trọng nhất là cửa số 1 nối với chuyển tiếp ti, và cửa số 3 dẫn lệnh điều khiển si tới bộ chấp hành.

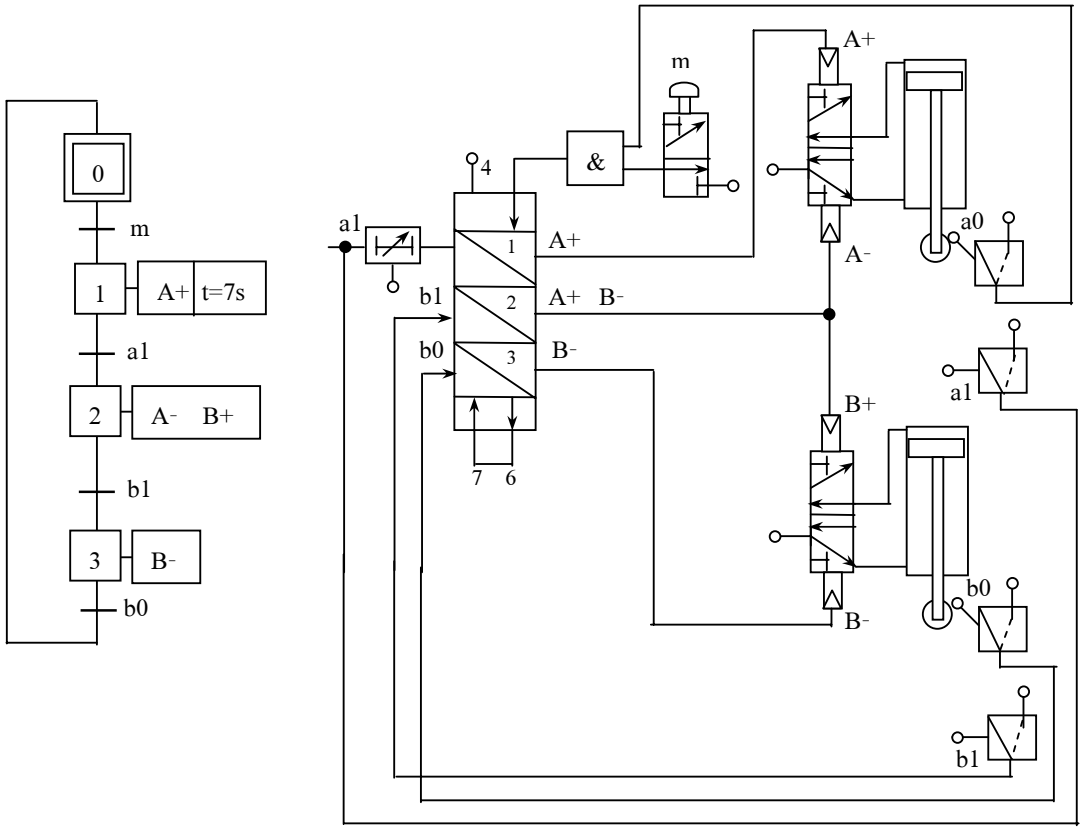
Do đó, quá trình lắp đặt, bảo dưỡng, sữa chữa hệ thống điều khiển rất đơn giản, thuận tiện, khó sai sót.

*.1.7. Ứng dụng sổ ghi tuần tự cho các trường hợp cơ bản.*

Sau đây xem xét các hệ thống tự động cơ bản và việc ứng dụng sổ ghi tuần tự để điều khiển chúng.

*a. Trường hợp không rẽ nhánh (nhánh đơn)*

Một hệ thống có hai pittong khí nén hoạt động theo sơ đồ GRAFCET như hình bên dưới




Có hai pittong A, B hoạt động với 3 trạng thái 1, 2 và 3. Tại điểm đầu và cuối hành trình pittong, đặt các cảm biến $a_0$, $a_1$ và $b_0$, $b_1$ để tạo ra các chuyển tiếp.

Chú ý rằng, ở trạng thái 1 yêu cầu khi kết thúc trạng thái, phải kéo dài một thời gian t = 7s mới chuyển sang trạng thái 2. Ở trạng thái 2, lệnh điều khiển $A^-$ và $B^+$ được thực hiện đồng thời.

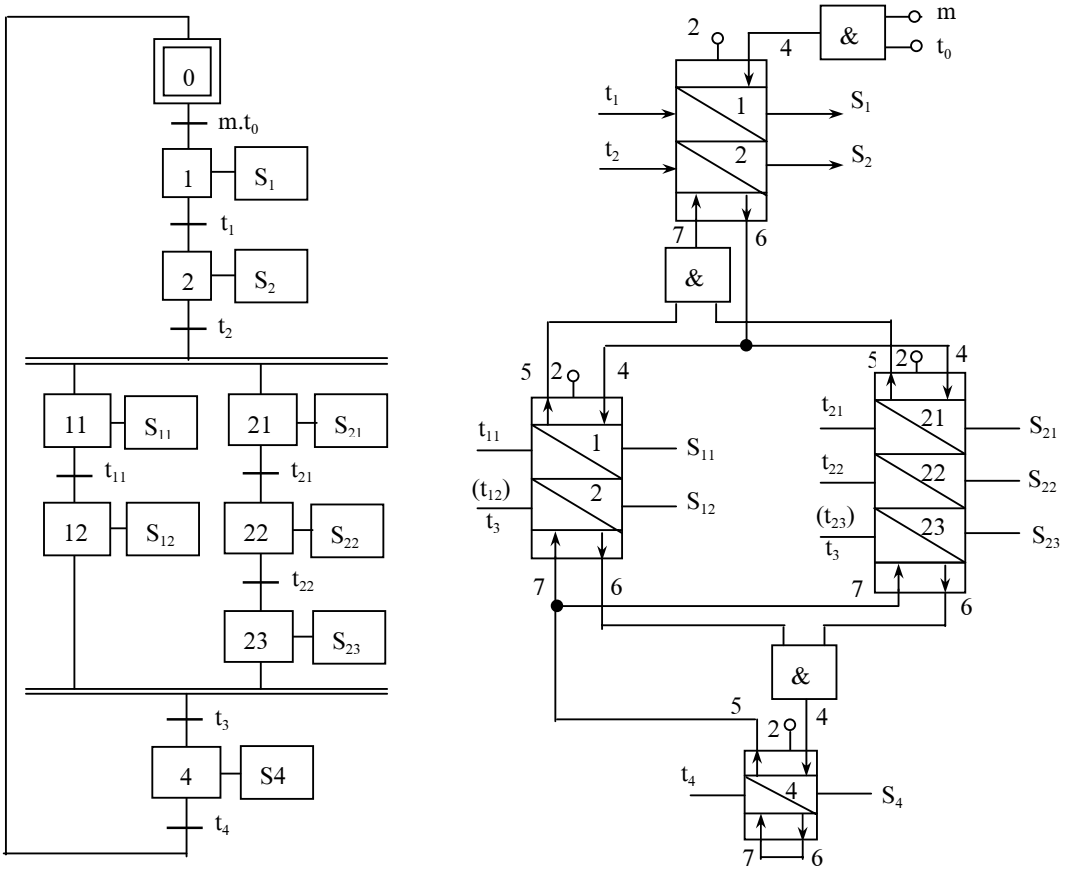
Hình bên phải là mạch điều khiển. Để điều khiển hai pittong A, B hoạt động, dùng 2 van phân phối loại 5/2 có các đầu vào điều khiển A+, $A^-$ và B+, $B^-$.

Để tạo ra các lệnh van phân phối, dùng ba sổ ghi tuần tự 1, 2, 3 tương ứng với 3 trạng thái của hệ. Từ trạng thái nghĩ, nhấn van m cùng với chuyển tiếp $a_0$ qua phần tử VÀ đi vào cửa 4 thì sổ ghi 1 xuất ra lệnh $A^+$ tới van phân phối để pittong A đi xuống. Ở cuối hành trình của nó, chuyển tiếp a1 xuất tín hiệu đi qua phần tử trễ - được điều chỉnh t= 7s theo yêu cầu - rồi mới tác động vào sổ ghi 1 để sổ ghi 2 xuất ra lệnh đồng thời đi đến đầu điều khiển $A^-$, $B^+$ của các van phân phối. Khi pittong B đi xuống, chuyển tiếp b1 sẽ đưa về sổ ghi 2 làm cho sổ ghi 3 xuất ra lệnh điều khiển $B^-$ đưa pittong B về vị trí ban đầu.

Để đưa toàn bộ hệ thống về trạng thái nghỉ, ta dùng cảm biến bo đưa vào sổ ghi 3 để xuất tín hiệu ra cửa 6 rồi nối về cửa 7 để xóa lệnh B- của sổ ghi 3.

*b. Trường hợp rẽ nhánh song song*

Một hệ thống có hoạt động rẽ hai nhánh song song được mô tả trên sơ đồ GRAFCET bên trái, còn bên phải là mạch điều khiển dùng các sổ ghi tuần tự.




Sau khi thực hiện trạng thái 2 với lệnh $S_2$, chuyển tiếp $t_2$ sẽ làm hệ rẽ thành 2 nhánh, còn chuyển tiếp $t_3$ làm 2 nhánh nhập về làm một và thực hiện trạng thái 4.

Cần chú ý chỗ rẽ nhánh và nhập về.

Tại chỗ rẽ, khi có chuyển tiếp $t_2$ đưa vào sổ ghi 2 thì ở cửa ra 6 của nó sẽ đồng thời tác động vào cửa 4 của sổ ghi đầu tiên của cả hai nhánh để xuất ra các lệnh $S_{11}$ và $S_{31}$ thực hiện việc rẽ nhánh.

Để nhập về trạng thái 4, dùng chuyển tiếp t3 tác động vào sổ ghi cuối của cả 2 nhánh để chúng cuất tín hiệu ra cửa 6 rồi cùng dẫn tới phần tử VÀ, từ đó xuất ra lệnh tác động vào cửa 4 của sổ ghi 4 để có lệnh S4. Ta phải dùng phần tử VÀ là để

đảm bảo rằng cả hai nhánh đều thực hiện xong trạng th ái cuối từng nhánh, dù chúng có thể so le về thời gian.

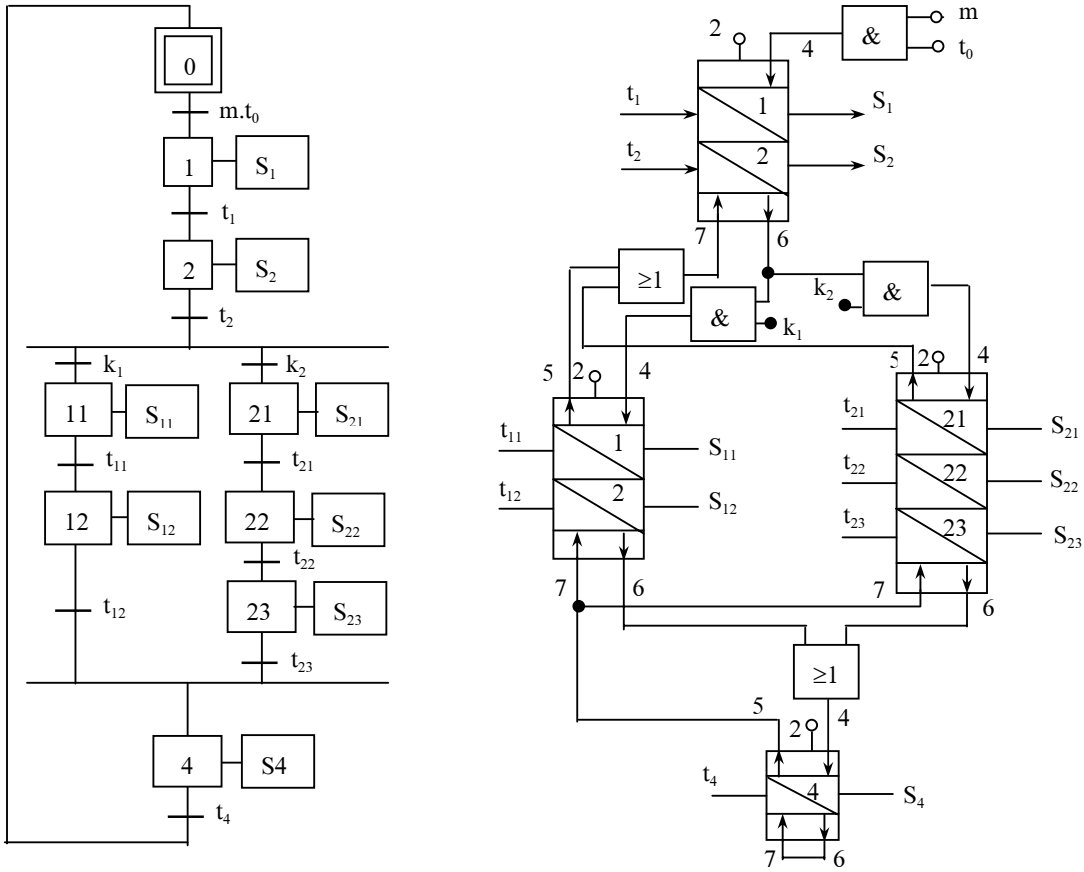
Để xóa lệnh S1 ở sổ ghi 2 sau khi đã thực hiện việc rẽ nhánh, thì dùng tín hiệu xuất ra cửa 5 của 2 sổ ghi đầu của cả 2 nhánh, đưa về phần tử VÀ rồi từ đó xuất tín hiệu vào cửa 7 của sổ ghi 2 để xóa $S_2$. Ở đây dùng phần tử VÀ cùng một mục đích như trên là đảm bảo chỉ khi 2 nhánh cùng thực hiện trạng thái đầu của mình thì mới xóa lệnh $S_2$.

Khi nhập về, có lệnh $S_4$ thì sổ ghi 4 xuất ra cửa 5 tín hiệu đồng thời đi vào cửa 7 của hai sổ ghi cuối từng nhánh để xóa lệnh $S_{12}$ và $S_{23}$.

Để trở về trạng thái nghĩ cho toàn bộ hệ thống, dùng chuyển tiếp $t_4$ kích thích sổ ghi 4 cho ra tín hiệu ở cửa 6 rồi đưa về cửa 7 để xóa $S_4$.

*c. Trường hợp rẽ nhánh có lựa chọn*

Hình bên trái dưới đây mô tả GARFCET hoạt động của một hệ thống tự động có rẽ nhánh lựa chọn. Tùy theo điều kiện lựa chọn $K_1$ hoặc $K_2$ mà từ trạng thái 2 rẽ sang nhánh 1 hoặc 2, rồi sau đó về dòng chính.



Hình bên phải là mạch điều khiển. Khi rẽ nhánh, lấy tín hiệu lấy tín hiệu chuyển tiếp t2 lúc thực hiện xong trạng thái 2 làm tín hiệu kích sổ ghi 2 xuất ra cửa 6 đi về 2 phần tử và ở 2 nhánh. Tại mỗi phần tử và này nếu có $K_1$ thì rẽ vào nhánh 1, nếu có $K_2$ - rẽ vào nhánh 2. Để xóa $S_2$ thì dùng tín hiệu xuất ra ở cửa 5 của hai sổ ghi

# V. MẠNG PETRI (PETRI NETS)

## 5.1. Khái quát chung

Đối với một hệ thống sản xuất tự động CIM (Computer Integrated Manufacturing) thì vấn đề tổ chức sản xuất và quản lí sao cho có hiệu quả là mọt trong những mục tiêu cơ bản nhất. Trong thực tế đã có một số giải pháp cũng như có nhiều nghiên cứu trong lĩnh vực này, trong số đó mạng Petri được xem là một ngôn ngữ mang tính tượng trưng cao và dễ dàng giao tiếp giữa các nhà thiết kế khác nhau trên thế giới cũng như rất thuật lợi và dễ dàng cho người sử dụng. Hơn thế nữa, mạng Petri lại rất được ưa dùng đối với các hệ thống sản xuất cơ khí tự động mà trong đó có robot tham gia vào quá trình cấp phôi liệu cho các máy và thu nhận sản phẩm gia công cùng với một nhà kho kiểu hiện đại.

Mạng petri cho phép người thiết kế được suy nghĩ và lựa chọn trong các khả năng rộng lớn các phương án với một mô hình gần gũi nhất với quá trình công nghiệp. Ngôn ngữ Petri mang bản chất tự nhiên và có thể dễ dàng biểu diễn được tính chất hoạt động của hệ thống sản xuất bằng mô hình hoá.

Trong dây chuyền sản xuất công nghiệp, các thiết bị, máy móc thường hoạt động theo một trình tự logic chặt chẽ nhằm đảm bảo chất lượng sản phẩm và hiệu quả của quá trình sản xuất cũng như an toàn cho người sử dụng và thiết bị. Cấu trúc làm việc tuần tự của dây chuyền đã đưa ra yêu cầu cho điều khiển là điều khiển sự hoạt động thống nhất và đồng bộ của các thiết bị trong dây chuyền. Trạng thái và các hoạt động của các thiét bị trong hệ thống được thực hiện kế tiếp nhau, có nghĩa là trạng thái hoạt động tiếp sau chỉ được thực hiện khi trạng thái hoạt động trước đó đã kết thúc. Việc mô hình hoá và tổng hợp các trạng thái hoạt động của mạch tuần tự thường được sử dụng bằng các hệ thống điều khiển như phương pháp đồ hình trạng thái Mealy hoặc Moore, bảng chuyển trạng thái, Grafcet và mạng Petri. Trong đó Grafcet và mạng Petri được sử dụng rộng rãi nhất trong các hệ thống sản xuất công nghiệp nói chung và lĩnh vực chế tạo cơ khí nói riêng.

Đối với mạng Petri, để biểu diễn sự hoạt động của hệ thống sản xuất công nghiệp, người ta thường sử dụng phương pháp mạng Petri diễn dịch hoặc mạng Petri nhuộm màu để phân tích và điều khiển hệ thống. Các công cụ toán học như ma trận ngẫu nhiên và các phương pháp biểu đồ phân nhánh và các lưu đồ kín có phản hồi ... cũng được sử dụng một cách rộng rãi.

Để có thể nắm bắt được những vấn đề cơ bản mà từ đó có thể tiến hành phân tích, tính toán và thiết kế hệ thống, cần phải đưa ra một số khái niệm và định nghĩa cơ bản nhất cũng như giải pháp tổng quát để phân tích hệ thống một cách toàn diện.

## 5.2. Các định nghĩa và phương pháp phân tích tổng quát

### *5.2.1 Các định nghĩa cơ bản*

Mạng Petri *(Petri Nets)* là một kiểu biểu đồ được phân chia thành 2 tập hợp :

- Vị trí *(Places)* : biểu diễn các trạng thái hoạt động có thể có của một thiết bị trong hệ thống sản xuất. Nó dược kí hiệu bằng một vòng tròn và trong đó có thể chứa một số dấu chấm (•) mà đặc trưng cho sự hoạt động của hệ thống.

- Chuyển tiếp *(Transition*) : biểu diễn các thao tác hay hành động theo trình tự hoạt động của các thiết bị trong hệ thống. Nó được kí hiệu bằng 1 hình chữ nhật hay có thể là 1 hình vạch ngang.

- Các cung nối *(Arc)* : Chỉ hướng hoạt động của hệ thống bằng sự liên kết giữa vị trí và chuyển tiếp cũng như giữa chuyển tiếp và vị trí. Một cung không bao giờ liên kết giữa 2 tập hợp mang cùng bản chất ví dụ như không thể liên kết giữa vị trí và vị trí dược. Trên các cung người ta có thể đánh dấu bằng các chữ số đặc trưng cho khả năng về trọng lượng được chuyển trên các cung theo hướng xác định. Nếu không có chữ số đi kèm theo các cung thì người ta qui định là trọng lượng được truyền là 1.

- Hàm đánh giá của 1 mạng Petri là 1 vectơ hợp thành có giá trị nguyên dương hoặc bằng không (0) với số thành phần bằng số vị trí.

Để có thể khái quát các định nghĩa trên đây chúng ta có thể khảo sát 1 mạng Petri đơn giản như biểu diễn ở hình 1-1

Mạng này chứa 4 vị trí được đánh dấu bằng $p_1$, $p_2$, $p_3$, $p_4$ và 5 chuyển tiếp được đánh dấu $t_1$, $t_2$, $t_3$, $t_4$.

Hàm đánh giá của nó là vectơ $M_o$ = {1,2,1,3}

Mạng Petri trên có thể được biểu diễn bằng 1 hàm PN = (P,T,A,W,$M_o$) Trong đó :

$P=\{p_1,p_2, \dots , p_n\}$ là 1 hữu hạn các vị trí (Places)

$T=\{t_1, t_2, \dots , t_n\}$ là 1 tập hợp hữu hạn các chuyển tiếp (Transition)

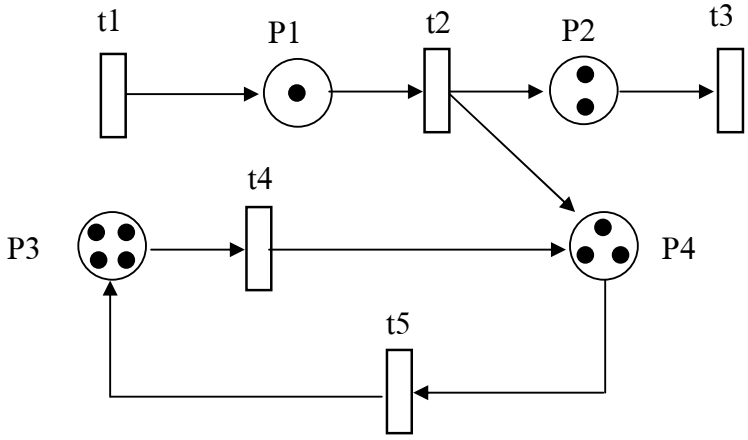
$A \subseteq (PxT) \cup (TxP)$ là 1 tập hợp hữu hạn các cung (Arc)

$W: \rightarrow \{1,2,3, \dots\}$ là hàm trọng lượng được gắn vào các cung (Weigh)

$M_0: \rightarrow \{0,1,2 \dots\}$ là hàm đánh giá ban đầu (Marcking)

Chú ý : $P \cap T = \varnothing$

Nếu 1 mạng Petri không chứa hàm đánh giá ban đầu N = (P,T,A,W) thì người ta gọi là mạng Petri thông thường và khi đó ta xem hàm trọng lượng được gắn với các cung bằng đơn vị, trong các vòng tròn biểu diễn vị trí không có dấu chấm hoặc chỉ có tối đa là 1. Khi đó PN=(N,$M_o$) và được gọi là mạng Petri mở rộng EPNs (Extended Petri Nets).




*Hình 5.1. Mạng Petri*

*5.2.2 Hoạt động của mạng Petri*

Như đã nói ở trên, các chuyển tiếp là khả năng chuyển các trạng thái hoạt động từ các vị trí đến các chuyển tiếp và từ chuyển tiếp sang vị trí.Do vậy sự hoạt động của mạng Petri có thế được biểu diễn như sau :

${}^0t$ là tập hợp các vị trí đi vào chuyển tiếp t, $(p,t) \in A$

$t^0$ là tập hợp các vị trí đi ra khỏi chuyển tiếp t, $(p,t) \in A$

${}^0p$ là tập hợp các chuyển tiếp đi vào vị trí p, $(t,p) \in A$

${}^0t$ là tập hợp các chuyển tiếp đi ra khỏi vị trí t, $(t,p) \in A$

1 chuyển tiếp t được gọi là khả tiếp (có thể vượt qua) Nếu hoặc là $p \in {}^0t$ , $M(p) \geq w(p\ t)$ hoặc theo cách khác, nếu tất cả vị trí di vào p từ t chứa số dấu (•) ít hơn hàm trọng lượng ở cung liên kết từ p tới t. Do đó trong trường hợp của 1 mạng Petri thông thường, chỉ cần tất cả các vị trí đi vào chứa ít hơn 1 dấu (•) thì nó có khả năng chuyển tiếp.

Trong phần này, chúng ta đi sâu vào việc phân tích và thiết kế mạng Petri cho 1 hệ thống SXTĐ CIM bao gồm 1 nhà kho phôi liệu và sản phẩm, 1 robot và

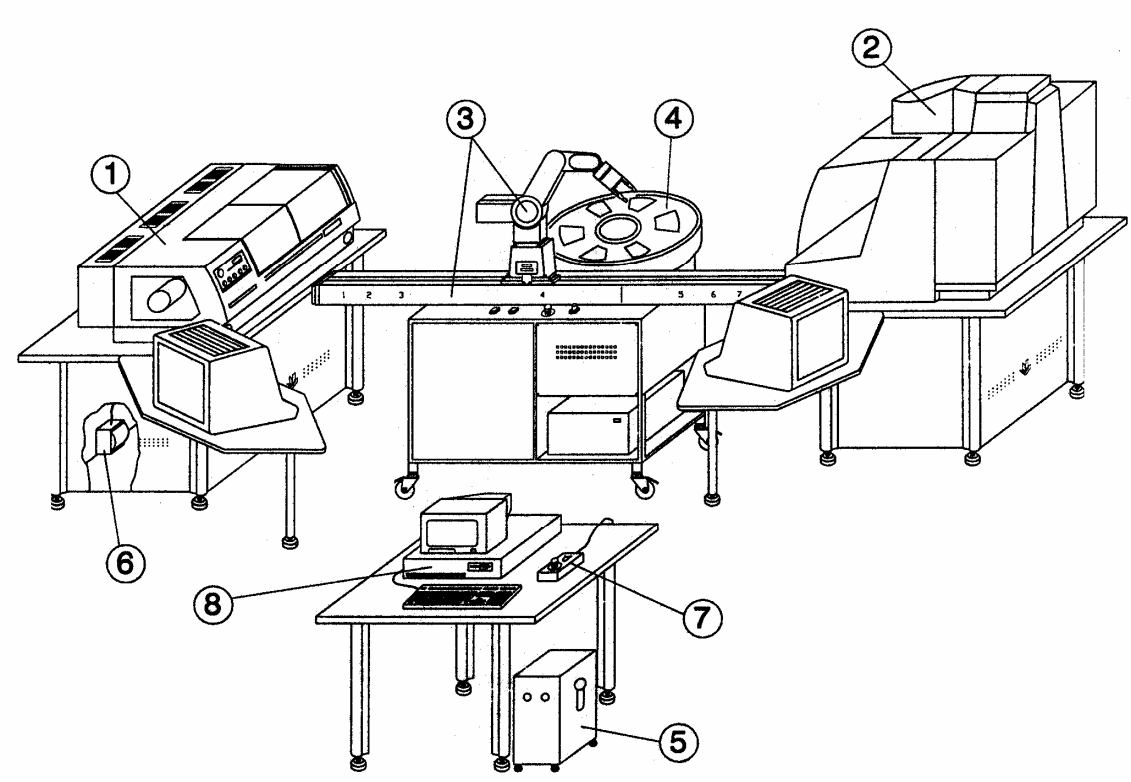
bàn dẫn hướng cung cấp phôi liệu cho máy tiện CNC và máy phay CNC cũng như lấy các chi tiết gia công xong trả lại kho... vì thế chúng ta chỉ giới hạn trong phạm vi

Petri thông thường có nghĩa là các cung chỉ mang hàm trọng lượng đơn vị và các vị trí tối đa có 1 dấu (•).

Giới thiệu hệ thống thiết bị trong mạng điều khiển CIM.

Hệ thống SXTĐ được lắp đặt tại phòng thí nghiệm SXTĐ khoa cơ khí do hãng ALECOP Tây Ban Nha cung cấp theo chương trình dự án với mục đích xây dựng 1 mô hình sản xuất công nghiệp tự động hoá để phục vụ cho công tác đào tạo và nghiên cứu. Mạng Petri được cung cấp bằng phần mềm Edifac cùng với mạng quản lí hệ thống và các phần mềm điều khiển máy tiện, máy phay, robot và nhà kho được kết nối qua máy tính với giao diện RS232 và RS485.

Sơ đồ thiết bị được bố trí như hình dưới.




*Hình 5-2: Sơ đồ bố trí hệ thống tự động CIM*

1. Máy tiện
2.Máy phay
3. Robot 5 bậc tự do và ray dẫn hướng
4. Nhà kho
5. Tủ điện
6. Bộ quét
7. Bộ lựa chọn chạy tay - tự động điều khiển robot
8. Máy vi tính

Giới thiệu chức năng hoạt động của mạng Petri

Các máy CNC tiện và phay được lập trình dưới dạng trực tiếp bằng tay trên cụm CNC hoặc trong phần mềm CAD/CAM VERO INTERNATIONAL. Trước khi hoạt động chính thức, người ta phải qua giai đoạn chạy mô phỏng vừa để kiểm tra sự hoạt động của chương trình được thiết lập, vừa để xác định chu kì thời gian gia công các chi tiết làm cơ sổ dữ liệu cho việc thiết kế mạng Petri sau này.

Bàn chứa phôi và chi tiết gia công (carrusel magazin) là mô hình của 1 nhà kho hiện đại mà trong đó các chủng loại phôi liệu cho từng loại chi tiết và các loại bán thành phẩm và sản phẩm cuối cùng được mã hoá và xếp theo loại dưới sự quản lí của 1 hệ thống máy tính và được hoạt động bằng các băng tải và tay máy để cung cấp, thu nhận hay di chuyển các phôi liệu được gọi hay xếp vào kho các bán thành phẩm hay sản phẩm nói trên.

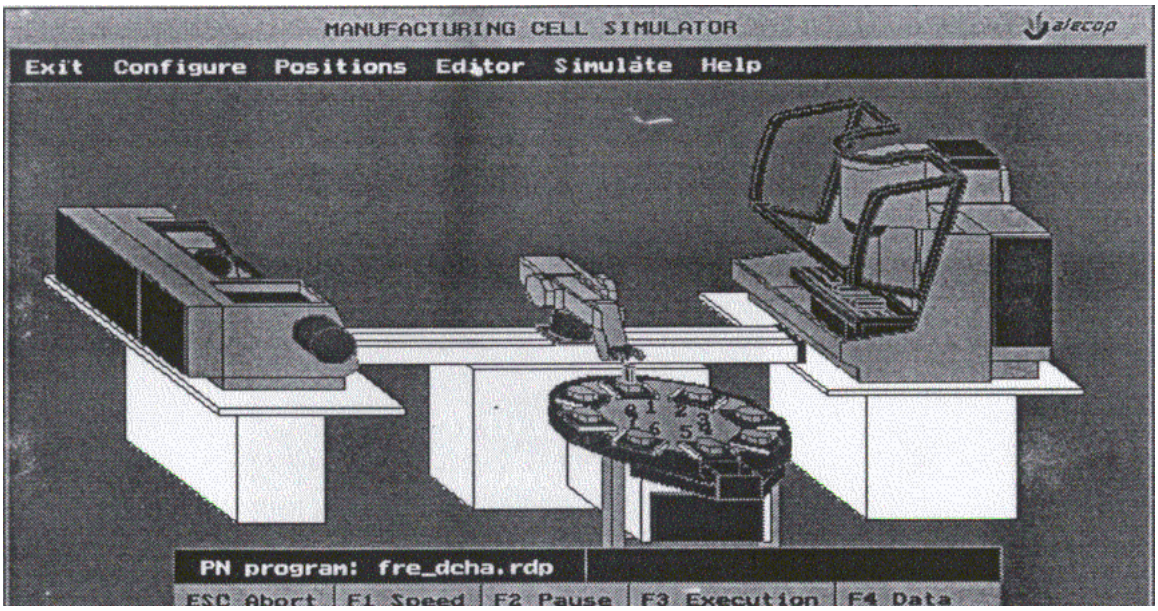
Để thực hiện việc cung cấp các phôi cho máy gia công CNC, người ta sử dụng 1 robot 5 bậc tự do và 1 ray dẫn đến các máy chịu sự cung cấp phôi liệu và lấy sản phẩm. Hoạt động của robot được lập trình bằng ngôn ngữ riêng của nó hay có thể lập trình theo phương pháp dạy học. Sự hoạt động của nó là hoạt toàn phù hợp với kích thước, khuôn khổ và khối lượng, hình dạng phôi liệu cần cung cấp, đảm bảo độ chính xác về định vị trong quá trình dịch chuyển và cấp phôi vào các vấu kẹp của mâm cặp hay các đồ gá trên máy phay cũng như khi nhận nó từ các cơ cấu trên của máy.

Cũng như với các máy CNC, trước khi đưa robot vào hệ thống hoạt động cấp phôi tự động cần thiết phải chạy mô phỏng để kiểm tra tính hợp lí và đúng đắn của chương trình và xác định thời gian chu trình hoạt động của chúng làm cơ sổ dữ liệu khi lập chương trình điều khiển hệ thống.

Vấn đề còn lại là làm thế nào để cho chúng hoạt động trên các thiết bị khác nhau 1 cách độc lập trong khi phải đảm bảo tính đồng bộ trong hệ thống. Điều này có nghĩa là mọi sự hoạt động phải ăn khớp nhau và đồng thời phải để thời gian chết

của các máy là tối thiểu. Về lĩnh vực này thì mạng Petri có thể hoàn toàn đảm bảo 1 cách chắc chắn và chính xác. Về cơ bản, mạng Petri cũng giống như GRAFCET (Graphe Fonctionnel de commande e'tape transition) là công cụ mô tả hoạt động các mạch logic tuần tự như đã giới thiệu ở trên. Do vậy mà vấn đề chồng chéo sự hoạt động của các thiết bị trong thời gian thực của hệ thống là không thể xảy ra. Vì thế vấn đề còn lại là làm thế nào để giảm thiểu thời gian chết (thời gian dừng chờ đợi, không hoạt động) của các máy và thời gian hoạt động của robot là ngắn nhất.

Để có thể hình dung sự hoạt động của hệ thống, chúng ta có thể quan sát sự hoạt động của robot cấp phôi như hình vẽ dưới đây.



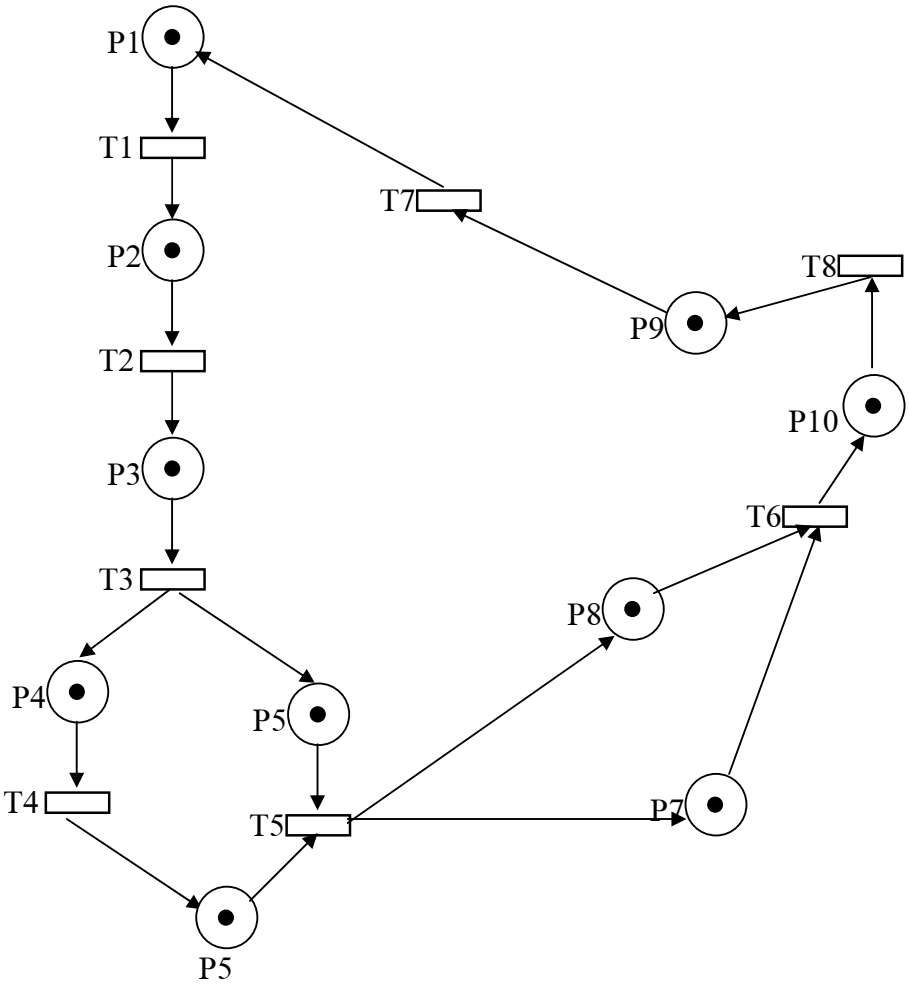

*Hình 5-3 : Sơ đồ hoạt động của hệ thống*

### 5.3. Lập chương trình điều khiển hệ thống điều khiển tự động CIM

Chúng ta sẽ thiết lập chương trình điều khiển sự hoạt động của hệ thống sản xuất tự động CIM bằng phần mềm EDIFAC được cung cấp bởi hãng ALECOP của Tây Ban Nha.

Sau đây là 1 số ví dụ về lập chương trình điều khiển hệ thống CIM bằng mạng Petri.

*5.3.1 Điều khiển hệ thống cấp phôi cho máy tiên và máy phay CNC P-T1.RPG*




*Hình 5-4: Mạng Petri cung điều khiển cấp phôi cho máy phay và máy tiện CNC*

$P_1$ : Trạng thái ban đầu của hệ thống CIM

$P_2$ : Lựa chọn phôi cấp cho máy phay 1.PA1 a001

$P_3$ : Cấp phôi cho máy phay. PR1

$P_4$ : Lựa chọn phôi cấp cho máy tiện 2. PA1 a002

$P_5$ : Máy phay gia công chiết theo chương trình. PT1

$P_6$ : Cấp phôi cho máy tiện. PR1

$P_7$ : Máy tiện gia công chi tiết theo chương trình. PT1

$P_8$ : Lấy chi tiết gia công xong ra khỏi máy phay. PR1

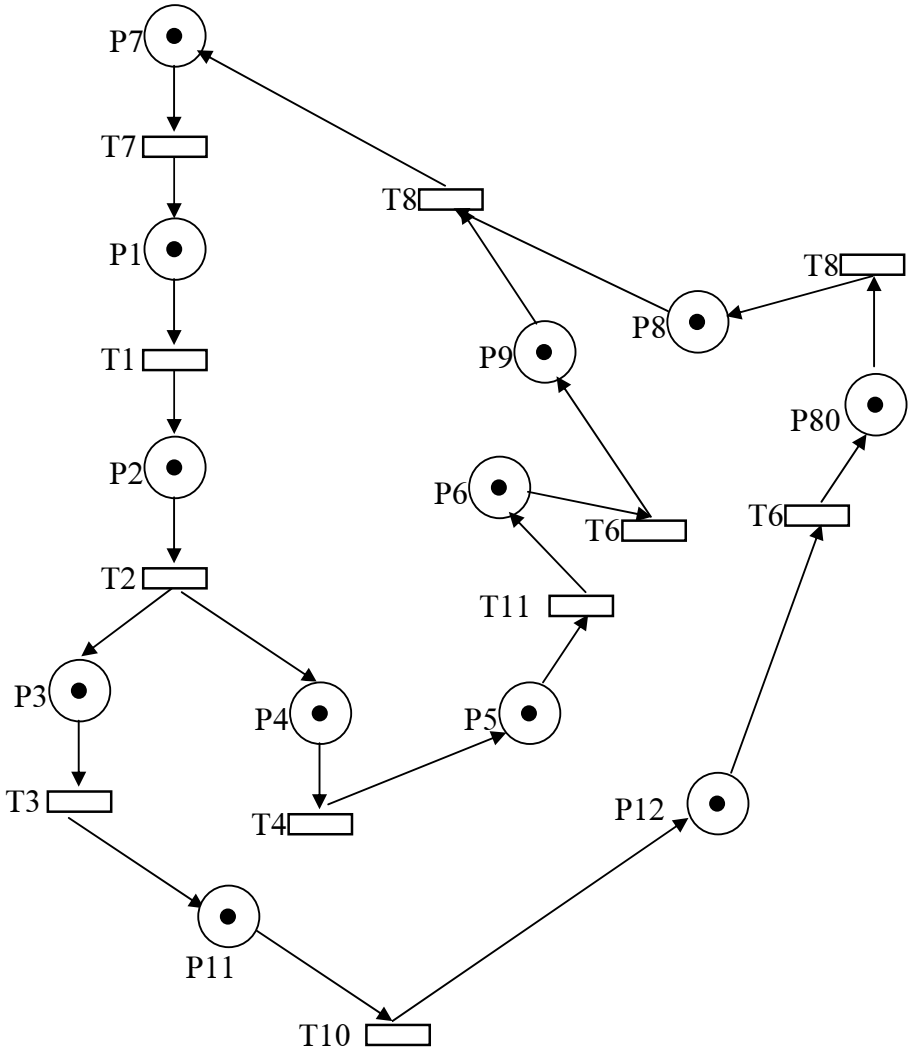
$P_{10}$ : Chọn Pallet phay. PA1 pa001

$P_9$ : Lấy chi tiết tiện. PR1

$T_1$ : Khởi động chương trình
$T_2$ : Dừng Pallet. LA1+
$T_3$ : Dừng robot. LR+
$T_4$ : Dừng Pallet. LA+
$T_5$ : Dừng robot và dừng máy phay. LR1 & LF1
$T_6$ : Dừng Robot và dừng tiện. LR1 & LT1
$T_8$ : Dừng Pallet. LA+
$T_7$ : Dừng Robot. LR+

*5.3.2 Chương trình điều khiển cấp phôi cho áy phay và máy tiện P-T2.RPG*

$P_1$ : Chọn phôi phay. PA1 p2
$P_2$ : Lấp phôi phay. PR1_cap_foi_fay-0002
$P_4$ : Chọn Pallet tiện. PA1 p1
$P_3$ : Chạy máy phay theo chương trình. PF1 0000016-0000
$P_5$ : Cấp phôi tiện. PR1_cap_foi_tien-001
$P_6$ : Chạy máy tiện theo chương trình. PT1 0000011-0000
$P_9$ : Chạy máy tiện theo chương trình. PT1 0000011-0000
$P_{10}$ : Máy tiện gia công chi tiết theo chương trình. PT1
$P_{11}$ : Chọn Pallet phay. PA1 p2
$P_{12}$ : Lấy chi tiết phay. PR1 - 0001
$P_8$ : Chọn Pallet tiện. PA1 p1
$P_7$ : Lấy chi tiết tiện. PR1-0001
$T_1$ : Dừng Pallet. LA1+
$T_2$ : Dừng robot. LR1+
$T_3$ : Dừng máy phay. LF1+
$T_4$ : Dừng Pallet. LA1+
$T_{11}$: Dừng Robot. LR1+
$T_5$ : Dừng máy tiện. LT1+
$T_{10}$ : Dừng Pallet. LA1+
$T_6$ : Dừng Robot. LR1+
$T_8$ : Dừng Pallet và dừng máy tiện LA1+ & LT1+
$T_7$ : Dừng Robot. LR1+




*Hình 5.5 : Mạng Petri P-T2. RPG*

*5.3.3 Chương trình điều khiển hệ thống P-T3.PRG*

$P_1$ : Trạng thái ban đầu

$P_2$ : Chọn Pallet phay. PA1 p2-0000

$P_3$ : Cấp phôi phay. PR1-0001

$P_4$ : Chọn Pallet tiện. PA1 p1

$P_5$ : Chọn máy phay. PF1 000003-0000

$P_6$ : Cấp phôi tiện. PR1-0001

$P_7$ : Chạy máy tiện. PT1 000007-0000
$P_8$ : Chọn Pallet phay. PA1-0000
$P_9$ : Lấy chi tiết phay. PR1-0001
$P_{12}$ : Chạy máy tiện. PT1 000007-0000
$P_{10}$ : Chọn Pallet tiện. PA1 p1
$P_{11}$ : Lấy chi tiết tiện. PR1-0001
$T_2$ : Dừng Pallet. LA1+
$T_3$ : Dừng Robot. LR1+
$T_4$ : Dừng Pallet. LA1+
$T_5$ : Dừng máy phay. LF1+
$T_6$ : Dừng Robot. LR1+
$T_7$ : Dừng máy tiện. LR1+
$T_8$ : Dừng Pallet. LA1+
$T_9$ : Dừng máy tiện. LR1+
$T_{10}$ : Dừng Pallet. LA1+
$T_{11}$ : Dừng Robot. LR1+

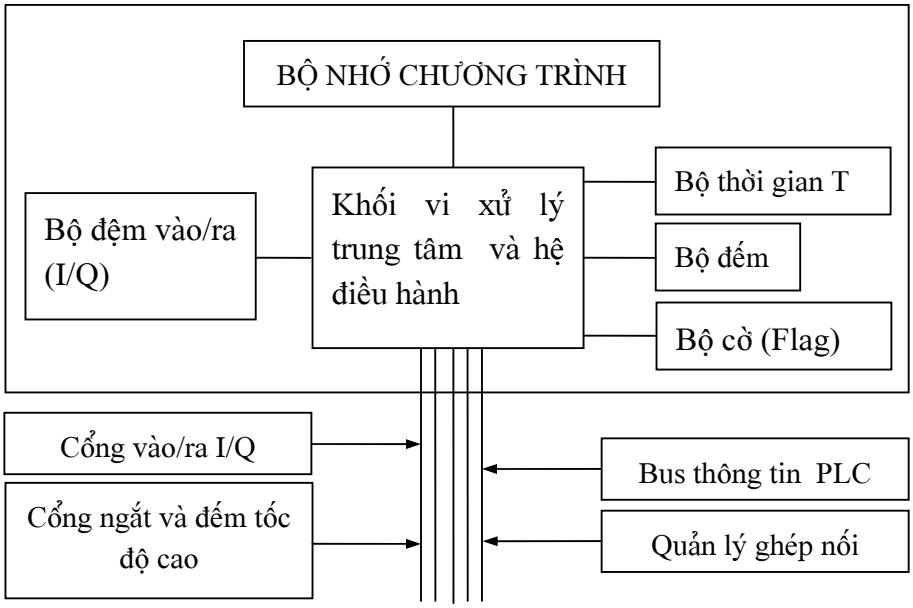
# VI. ĐIỀU KHIỂN LOGIC KHẢ LẬP TRÌNH PLC

*(Programmable Logic Control)*

## 6. 1. Khái niệm

Điều khiển logic khả lập trình cho phép thực hiện linh hoạt các thuật toán điều khiển số thông qua một ngôn ngữ độc lập thay vì việc thể hiện thuật toán đó bằng mạch logic số như đã trình bày trước đây. Như vậy, với chương trình điều khiển trong mình. PLC đã trở thành một bộ điều khiển số NC *(Numeric Control)* nhỏ gọn và có tính linh hoạt cao, dễ thay đổi thuật toán để thích ứng với sự thay đổi công nghệ. Đặc biệt hơn là dễ dàng trao đổi thông tin với môi trương xung quanh (với các PLC khác và máy tính). Toàn bộ chương trình điều khiển được lưu trữ trong bộ nhớ của PLC dưới dạng các khối chương trình (OB, FC hay FB) và được thực hiện lặp theo chu trình theo các vòng quét *(scan)*.

Mô hình tổng quát của một bộ điều khiển PLC được mô tả như hình dưới đây




*Hình 6-1: Sơ đồ cấu trúc bộ PLC*

Để có thể thực hiện được 1 chương trình điều khiển, PLC cần phải có tính năng giống như 1 máy tính, nghĩa là phải có 1 bộ vi xử lý CPU và 1 hệ điều hành, bộ nhớ để lưu chương trình điều khiển, dữ liệu và tất nhiên phải có các cổng vào/ra

để giao tiếp được với đối tượng điều khiển và môi trường xung quanh (ngoại vi). Ngoài ra, để giải quyết được bài toán điều khiển số, PLC còn phải được trang bị thêm các khối chức năng đặc biệt như bộ đếm, bộ thời gian, đồng hồ ...

## 6.2. Các khối chủ yếu trong PLC

### *6.2.1. Khối tổ chức OB (Organization Block)*

Mục đích : Hình thành giao diện *(interface)* giữa hệ điều hành và chương trình điều khiển.

Tùy thuộc từng số hiệu của khối, các khối tổ chức được vận hành theo những cách thức khác nhau.

Ví dụ : khối OB1 được hệ điều hành gọi theo chu kỳ để thực hiện chương trình điều khiển.

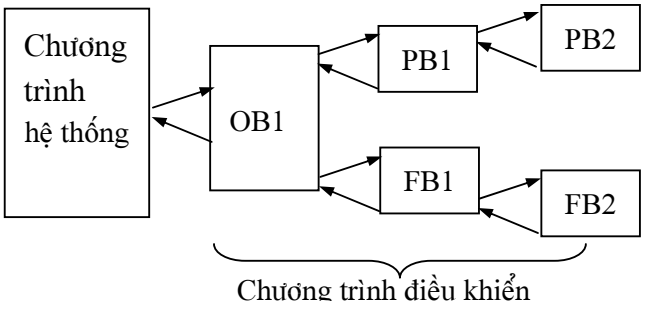
Khối OB2 và OB3 xử lý chương trình đóng ngắt ...

### *6.2.2. Khối chương trình :PB (Program Block).*

Khối chương trình với những chức năng riêng giống như một chương trình con hay 1 hàm. Các chức năng điều khiển có thể được biểu diễn bằng 1 sơ đồ trong các khối chương trình.

### *6.2.3.. Khối chức năng : FB (Function Block)*

Các khối chức năng có thể trao đổi được một lượng dữ liệu lớn với các khối chương trình khác. Nó có thể được gán các tham số vì thông thường khi lập trình, người ta gán 1 tham số hình thức và tương ứng với nó sẽ có 1 tham số thực. Khi khối được gọi lên thì các tham số thực sẽ thay thế các tham số hình thức.




*Hình 6-2 : Ví dụ về các khối xử lý khi lập trình có cấu truc*

Chú ý: Chương trình trong khối chức năng chỉ được viết bằng phương pháp biểu diễn bằng kiểu lệnh STL mà thôi *(Statement list)* .

Tương tự như khối OB, khối FB được chia làm 2 loại là FB được lập trình sẵn trong hệ điều hành và loại FB được lập dưới dạng các phần mềm.

*6.2.4. Khối DB (Data Block) khối dữ liệu:*

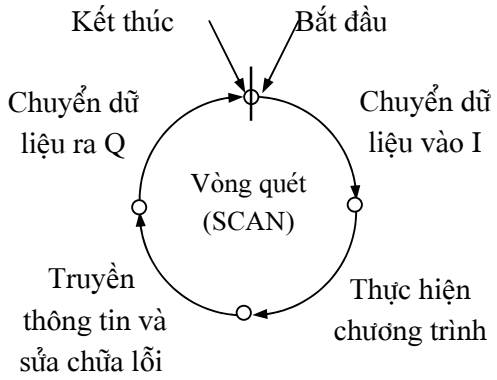
Khối DB chứa các dữ liệu cần thiết để lập chương trình. Các tham số của khối do người lập trình tự đặt và một chương trình ứng dụng có thể có nhiều khối và được phân biệt với nhau bằng các chỉ số nguyên ghi ngay sau khối DB1, DB2 ...

**6.3. Nguyên tắc hoạt đông cơ bản**

Bộ điều khiển PLC hoạt đông theo nguyên tắc chu trình kín và được gọi là chu trình quét *(scan)*. Nó hoạt động theo 4 giai đoạn như sau :

*6.3.1. Mổ đầu chương trình : Nhập dữ liệu.*

Lúc bắt đầu chương trình, PLC nhập dữ liệu từ các cổng vào số tới vùng bộ đệm ảo I.*(Input)*




*6.3.2. Thực hiện chương trình.*

Sau khi đã nhập xong dữ liệu, PLC gọi chương trình điều khiển cất giữ trong bộ nhớ chương trình theo từng lệnh lần lượt để thực hiện theo các thứ tự kế tiếp nhau. Trong quá trình quét, chương trình được thực hiện từ lệnh đầu tiên đến lệnh kết thúc của khối OB1.

*6.3.3. Giai đoạn truyền thông tin ra và tự sữa chữa lỗi.*

Các kết quả xử lý sẽ được kiểm tra và sữa chữa lỗi rồi chuyển sang bộ đệm ảo ở cổng ra số Q.

*6.3.4.Giai đoạn kết thúc chương trình.*

Chuyển các nội dung điều khiển đến đầu ra và chuyển sang cơ cấu chấp hành.

Sau đó lại bắt đầu 1 chu trình mới: Người ta gọi là vòng lặp

**6..4 Ngôn ngữ lập trình**

Các loại PLC nói chung có nhiều ngôn ngữ lập trình nhằm phục vụ cho các đối tượng sử dụng khác nhau. Về cơ bản thì các ngôn ngữ được các nhà chế tạo quy định và theo từng *verson* cụ thể. Vì vậy khi sử dụng cần phải có thời gian tìm hiểu - làm quen.

Về cơ bản, các ngôn ngữ lập trình PLC gồn có 3 loại cơ bản :

*6.4.1. Ngôn ngữ "liệt kê lệnh" STL (Statement List)*

Loại này giống như các ngôn ngữ thường được dùng trên máy tính _ Lập trình bằng các lệnh. Một chương trình được ghép bằng nhiều câu lệnh theo 1 thuật toán nhất định. Mỗi lệnh chiếm 1 hàng và đều có cấu trúc chung là "tên lệnh + toán hạng"

Ví dụ :

```
...
004   A    I 0.2
005   O    I 0.3
...
```

Tên và loại đối tượng lệnh (VÀO), tham số 0.3

Tên lệnh

Nhãn

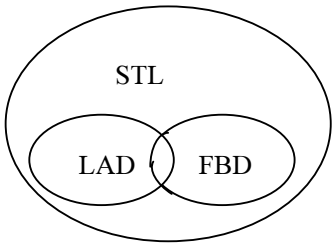
*6.4.2.Ngôn ngữ bậc thang : (Biểu đồ bậc thang : LAD _ Ladder logic)*

Đây là dạng ngôn ngữ kiểu đồ họa và rất thích hợp với mạch logic tiếp điểm. Với những người thiết kế mạch logic tiếp điểm thì ngôn ngữ này rất thích hợp vì tính trực quan và tường minh của nó.

*6.4.3. Ngôn ngữ "Sơ đồ khối" FBD (Function Block Diagram).*

*(Người ta còn gọi là "lưu đồ điều khiển CSF - Control System Flow)*

Đây là dạng ngôn ngữ sơ đồ mạch số và cũng rất thích hợp cho người thiết kế các mạch điện tử số.




*Hình 6-3 : Quan hệ giữa các ngôn ngử lập trình PLC*

Một chương trình được viết trên LAD hay FBD đều có thể chuyển được sang STL. Nhưng chương trình được viết bằng ngôn ngữ STL chưa hẳn đã chuyển sang được kiểu LAD và FBD vì trong STL có chứa nhiều lệnh mà trong LAD hay FBD không có hay không thể biểu diễn được.

Có thể thấy rằng, trong ngôn ngữ lập trình PLC, các lệnh thường được chia thành 3 nhóm sau :

+Nhóm lệnh cơ bản : được dùng cho tất cả các loại khối và có thể biểu diễn được bằng cả 3 phương pháp.

+Nhóm lệnh bổ trợ.

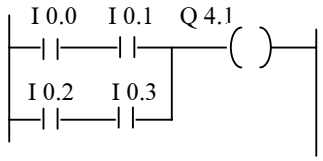
+Nhóm các lệnh hệ thống.

Với 2 nhóm lệnh sau, chúng được dùng trong các khối chức năng và chỉ có thể biểu diễn được bằng ngôn ngữ lập trình STL. Đặc biệt với các lệnh hệ thống thì tính chất của nó là rất ngắn gọn và mạnh nhưng chỉ những người có kiến thức chuyên sâu về PLC, về công nghệ điều khiển và nhiều kinh nghiệm lập trình mới có thể sử dụng được.

Ví dụ các kiểu ngôn ngữ lập trình.

*Ví dụ 1*

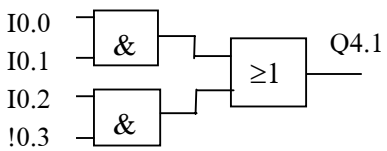
LAD




STL

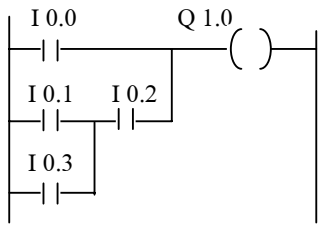
```
A   I 0.0
A   I 0.1
O
A   I 0.2
A   I 0.3
=   Q 4.1
```

CSF(FBD)




*Ví dụ 2*

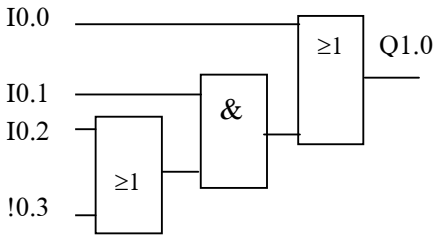
LAD




STL

```
O   I 0.0
O
A   I 0.1
A   (
O   I 0.2
O   I 0.3
)
=   Q 1.0
```

CSF(FBD)




*3.5.5 Một số ví dụ áp dụng :*

Bài toán 1:

Ví dụ về điều khiển cho thùng trộn

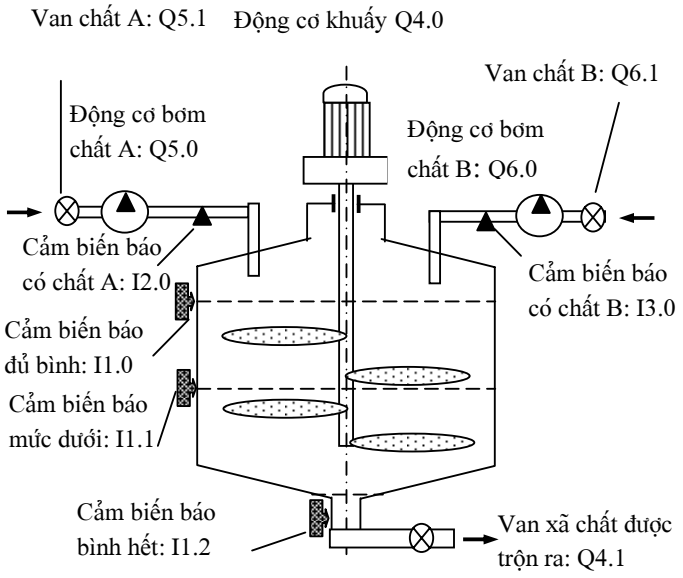
Sơ đồ thùng trộn và bố trí các cảm biến:

Nguyên lý hoạt động:Quá trình trộn chất lỏng chỉ được thực hiện khi chất lỏng trong bình đã hết và ta tác động vào nút ấn khởi động chu trình (I8.0=1) (không biểu diễn trên hình vẽ)

.

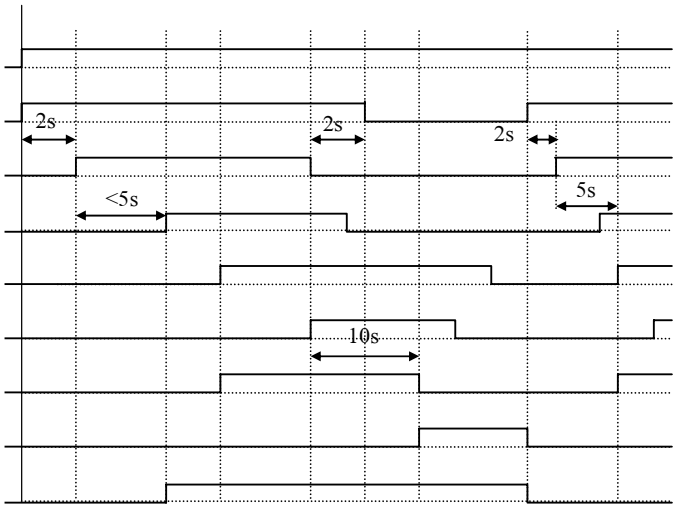
Các chất lỏng sẽ được bơm vào bình sau khi mở van chất lỏng A và B được 2 giây. Q5.0=1 và Q6.0=1.

Nếu sau khi động cơ bơm đã khởi động được 5 giây mà không có tín hiệu thông báo có chất lỏng chảy vào bình thì dừng tất cả các quá trình lại và thông báo ra panel điều khiển có sự cố. I2.0=0, I3.0=0.




*Hình : Sơ đồ bình trộn chất*

| | |
|---|---|
| *Nút khởi động* | *I1.0* |
| *Mở van bơm* | *Q5.1* |
| *Động cơ bơm* | *Q5.0* |
| *Có chất lỏng vào thùng* | *I2.0* |
| *Mức chất lỏng dưới* | *I1.1* |
| *Mức chất lỏng trên* | *I1.0* |
| *Động cơ trộn* | *Q4.0* |
| *Van xã* | *Q4.1* |
| *Kiệt bình* | *I1.2* |




Khi bình đã được bơm đầy : I1.0=1 thì dừng cả 2 động cơ bơm. Quá trình dừng máy bơm được thực hiện theo nguyên tắc sau: Dừng máy bơm trước: Q5.0=0 và Q6.0=0 rồi sau 2 giây sẽ khóa van Q5.1=0 và Q6.1=0.

Động cơ trộn hoạt động Q4.0=1 khi bắt đầu có tín hiệu từ cảm biến báo mức dưới I1.1=1 cho đến khi cảm biến báo đủ bình I1.0=1 và tiếp tục trộn thêm 10 giây rồi kết thúc.

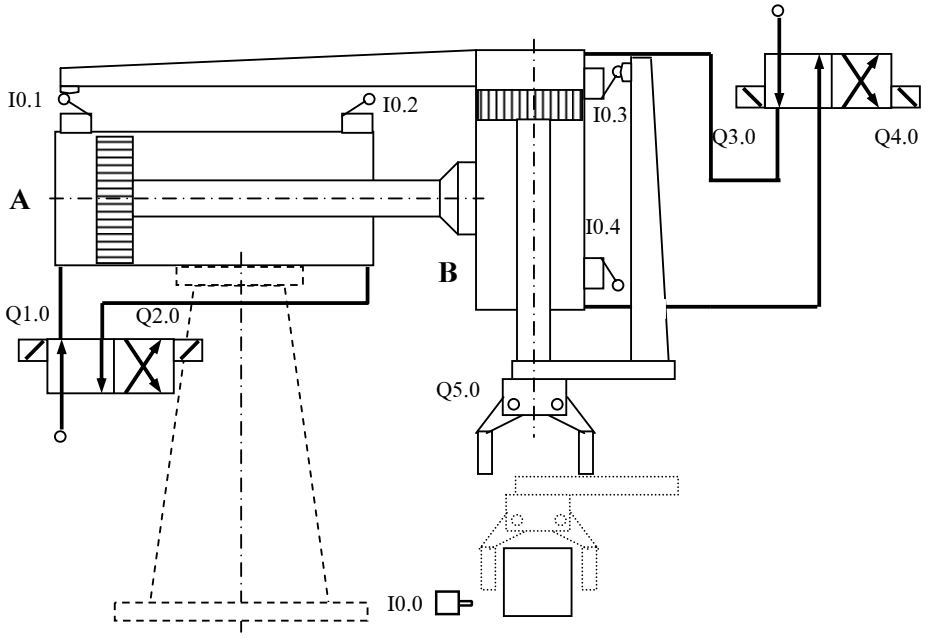
Sau khi đã được trộn đều thì chất lỏng được tháo ra khỏi bình nhờ van xả Q4.1=1 và cho tới khi van báo kiệt bình I1.2=0 thì kết thúc.

Nếu trong lúc này nút khởi động vẫn còn tác dụng (I8.0 =1) thì chu trình lại được tiếp tục cho đến giai đoạn xã kiệt nước trong bình mặc dầu khi đó nút khởi động đã hết tác dụng (I8.0 =0) và sau đó là kết thúc.

Bài toán 2:

Một người máy hoạt động như sau:

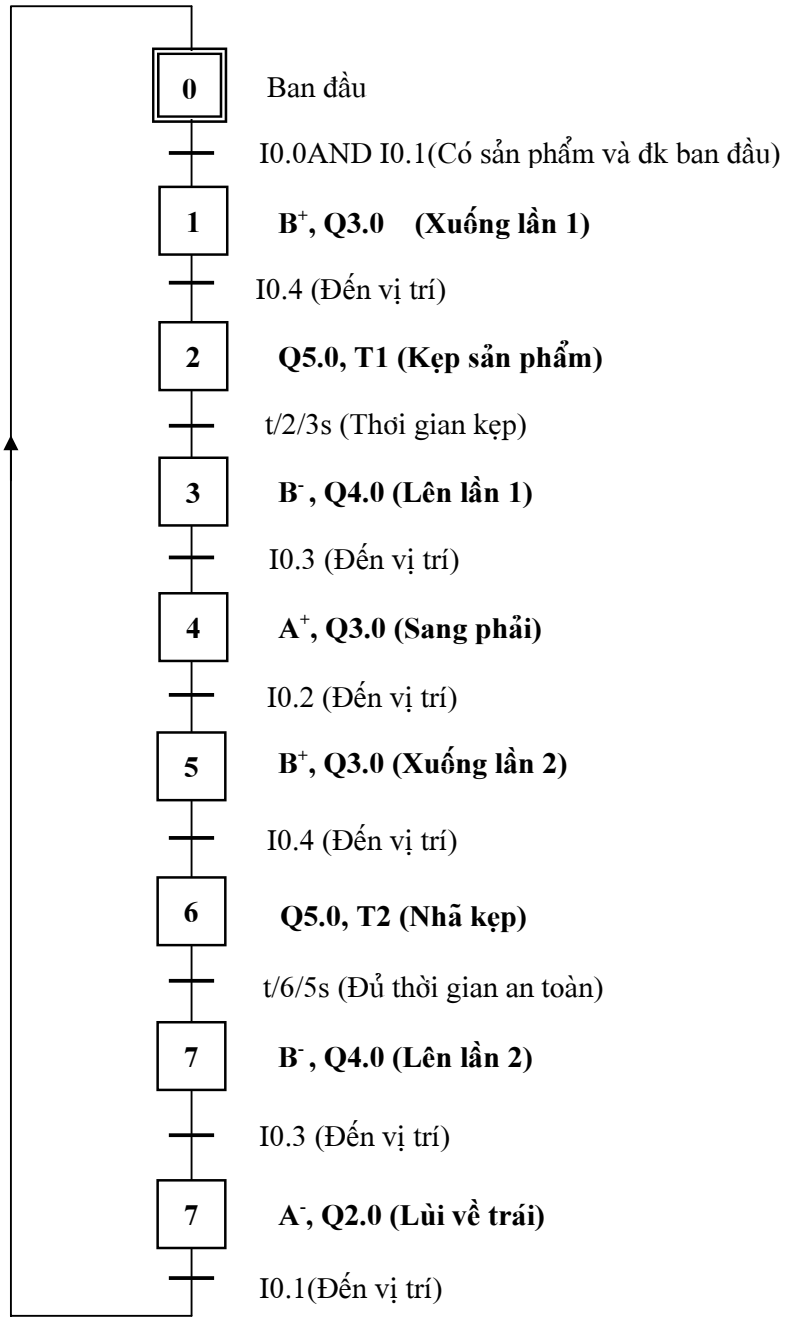
Ban đầu Piston A nằm ở vị trí cuối cùng bên trái (I0.1=1) và Piston B ở vị trí trên cùng (I0.3=1). Khi có sản phẩm đến thì cảm biến quang I0.0 tác động và chu trình bắt đầu.




Piston B đi xuống và khi tác động lên I0.4 thì nam châm điện từ Q5.0 sẽ tác động và kẹp sản phẩm, sau thời gian 3 sec thì Piston B dịch chuyển lên trên và mang sản phẩm đi lên cho đến khi I0.3 tác động thì Piston A sẽ dịch chuyển sang phảicho đến khi tác động lên I2.0 thì Piston B sẽ đi xuống. Khi tác động lên I4.0 thì Nam châm điện từ Q5.0 sẽ mở để nhã sản phẩm. Sau thời gian mở 5sec để đảm bảo là sản phẩm đã hoàn toàn rời khỏi tay kẹp thì Piston B dịch chuyển lên trên, khi tác động lên I0.3 thì Piston A lùi về bên trái cho đến khi tác động lên I1.0 thì kết thúc chu trình.

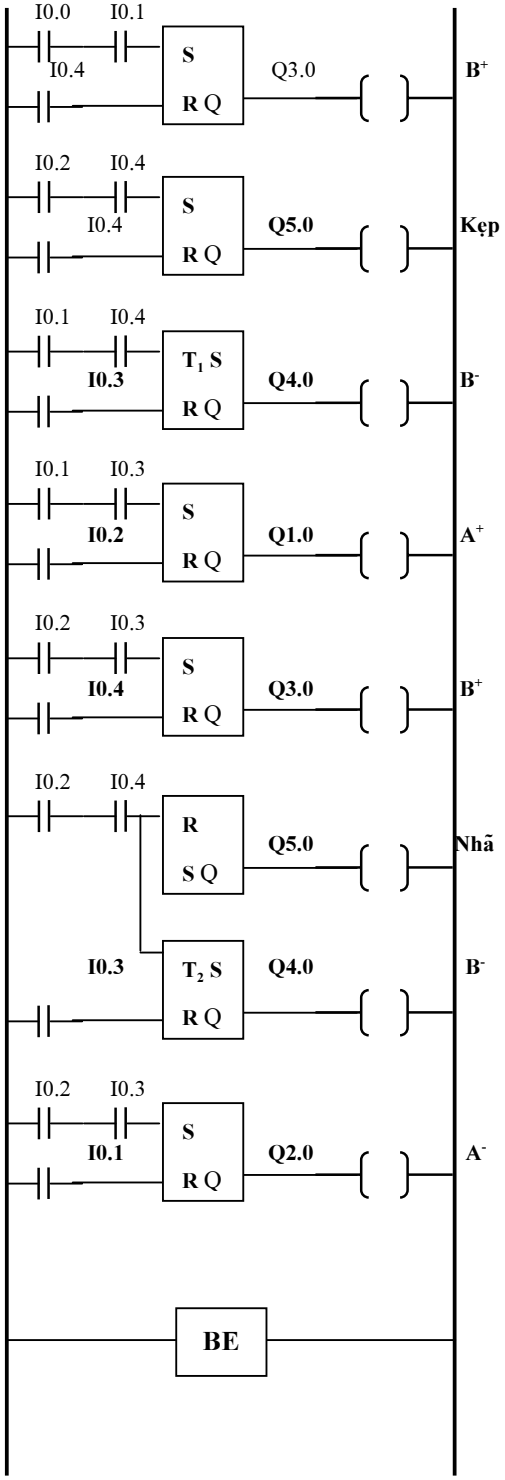
Nếu cảm biến I0.0 phát hiện có sản phẩm đến thì chu trình làm việc lại dược tiếp tục.

*Thiết lập GRAFCET*

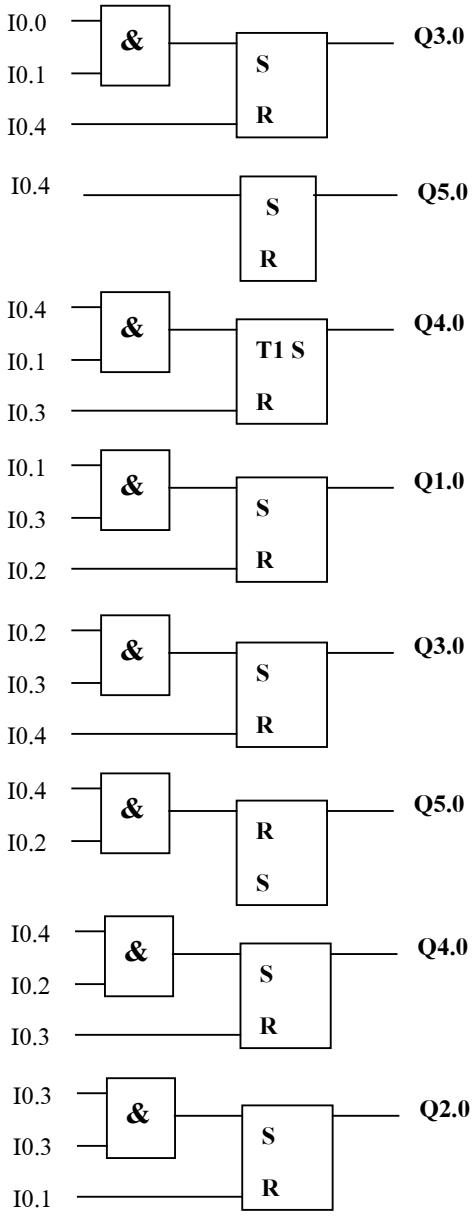



*Chương trình*

**LADDER**




**CSF**




**BE**