TS. NGUYỄN NAM QUÂN

TOÁN LOGIC & KỸ THUẬT SỐ

NHÀ XUẤT BẢN KHOA HỌC VÀ KỸ THUẬT

TS. NGUYỄN NAM QUÂN

TOÁN LÔGIC VÀ KỸ THUẬT SỐ

NHÀ XUẤT BẢN KHOA HỌC VÀ KỸ THUẬT

HÀ NỘI - 2006

LỜI NÓI ĐẦU

Hiện nay Kỹ thuật điện tử nói chung, trong đó có các hệ thống điện tử số nói riêng có sự phát triển vô cùng mạnh mẽ. Các hệ thống điện tử số đã xâm nhập vào hầu hết các khía cạnh của cuộc sống con người, và trở nên gần gũi tới mức mọi người đều quen thuộc, cũng như thấy bình thường khi nói tới máy tính, máy vi tính, ngôn ngữ lập trình, hệ thống điện thoại số, truyền hình số, internet, rôbot...

Việc số hóa thông tin, xử lý thông tin cũng như số hóa tín hiệu điện tử trong các thiết bị và các hệ thống điện tử số đang là xu thế phát triển của các nước có nền công nghiệp điện tử mạnh. Sự phát triển này đã tạo ra các hệ thống, các thiết bị điện tử cho phép thực hiện được những điều vượt cả trí tưởng tượng của con người, như hệ thống mô phỏng cấu tạo vật lý, sinh học, hóa học, không gian ảo... Sự kết hợp hoàn hảo giữa toán học (toán lôgic) và kỹ thuật công nghệ điện tử ở trình độ cao cho phép tạo ra lý thuyết ôtômat, Kỹ thuật điện tử số và Kỹ thuật xử lý số như hiện nay.

Để giúp sinh viên, các cán bộ nghiên cứu hiểu biết sâu sắc về các hệ thống điện tử số hiện đại, làm cơ sở cho việc làm chủ và có thể sáng tạo ra các hệ thống điện tử số mới, chúng tôi xin giới thiệu quyển sách TOÁN LÔGIC VÀ KỸ THUẬT SỐ. Nội dung của cuốn sách giúp người học hiểu được bản chất của các hệ thống số hiện đại và sự liên hệ hữu cơ giữa toán lôgic với kỹ thuật số, một kiến thức được coi là cơ sở cho ngành kỹ thuật điện tử số, kỹ thuật tính toán cũng như kỹ thuật xử lý tin.

Cuốn sách này được viết dựa trên đề cương môn học Kỹ thuật số đã được thông qua và thống nhất trong cả nước, do tác giả được phân công làm chủ biên, dưới sự phản biện của Hội đồng gồm các giáo sư, tiến sĩ đã tham gia giảng dạy nhiều năm về lĩnh vực khoa học này. Quyển sách được chia làm hai phần chính :

PHẦN I : *Cấu trúc toán lôgic. Phần này chủ yếu nêu lên các vấn đề cơ bản của toán lôgic–toán rời rạc. Bản chất của đại số nói chung và đại số lôgic nói riêng, các hệ cơ số đếm. Đại số Boole và hàm Boole, các phương pháp rút gọn hàm Boole, đạo hàm hàm Boole để đánh giá chất lượng hoạt động của hệ thống số, mã dùng trong máy.*

PHẦN II : *Hệ thống số và mạch số. Phần này đề cập tới các ôtômat không có nhớ và ôtômat có nhớ, các mạch điện của hệ thống số, các đặc điểm hoạt động của hệ thống điện tử số, trong đó đi sâu vào hai bài toán chính đó là bài toán phân tích hệ thống số (dùng để sửa.chữa), và bài toán tổng hợp hệ thống số (dùng để thiết kế) vì đó là hai nhiệm vụ không thể thiếu của người kỹ sư.*

Quyển sách có thể làm tài liệu tham khảo cho kỹ sư ngành Kỹ thuật điện tử Viễn thông, Kỹ thuật điện tử số và kỹ sư Điều khiển tự động cùng các lĩnh vực khoa học

khác. Các sinh viên năm cuối cũng như nghiên cứu sinh của các chuyên ngành liên quan có thể dùng làm tài liệu tham khảo thiết kế tốt nghiệp.

Trong lần xuất bản này, cuốn sách không tránh khỏi còn những sai sót và có những hạn chế nhất định. Chúng tôi mong nhận được các ý kiến đóng góp của bạn đọc để có thể bổ sung và hoàn thiện hơn. Các ý kiến xin gửi về Bộ môn Điện tử–tin học, Khoa Điện tử–Viễn thông, Trường Đại học Bách khoa Hà Nội.

Hà Nội, tháng 5 năm 2005

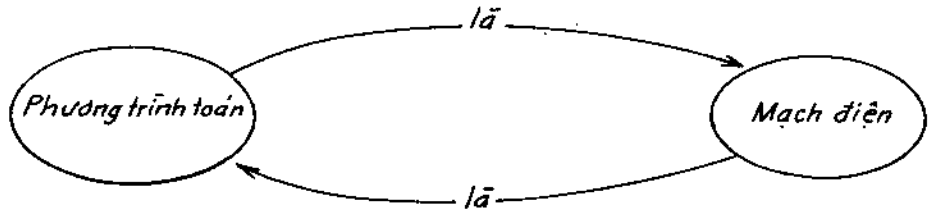
Tác giả

MỤC LỤC

PHẦN I
CẤU TRÚC TOÁN LÔGIC (RỜI RẠC)

GIỚI THIỆU CHUNG

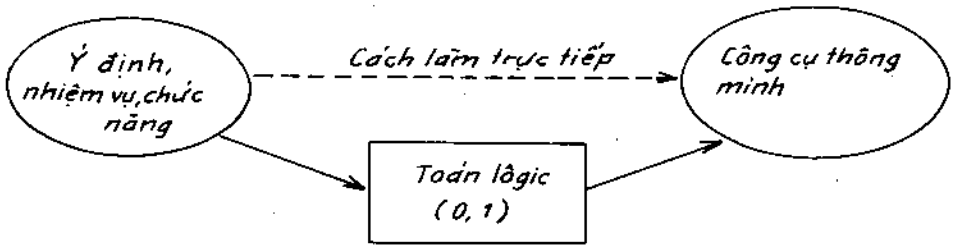
Phần I trình bày toán lôgic hay "toán rời rạc", đây là công cụ, là cơ sở chính để xây dựng nên các hệ thống tự động điều khiển số cũng như các hệ thống tự động số có những chức năng ứng dụng tuyệt vời và rất hiện đại của con người đang được sở hữu. Có thể nói nếu không có toán lôgic–toán rời rạc thì sẽ không có kỹ thuật số, không có các hệ thống số. Toán học lôgic đã thay thế hoàn toàn các nhiệm vụ của toán học cổ điển, là cơ sở để chế tạo ra máy tính điện tử và hàng loạt các chức năng khác, thỏa mãn nhu cầu cuộc sống của con người như : Các hệ thống tự động điều khiển, không gian ba chiều, rôbôt, tổng đài... Vì vậy, để hiểu được cấu trúc cũng như nguồn gốc chế tạo của các hệ thống hiện đại đó bắt buộc chúng ta phải hiểu biết về toán lôgic (0,1). Trong phần này, khi phân tích các quan hệ toán học của toán lôgic chúng ta đứng ở các góc độ khác nhau của nhà toán học, cũng như của nhà kỹ thuật, để từ đó có thể áp dụng toán học vào nhiệm vụ kỹ thuật một cách dễ dàng. Bản chất của các hệ thống toán học về kỹ thuật được chỉ ra trên hình 1.



Hình 1. Quan hệ tổng quát của ôtômat.

Quan hệ phương trình toán là mạch điện và ngược lại mạch điện cũng chính là phương trình toán là hệ thống có sự thống nhất một–một giữa toán và mạch điện, có tên chung là ôtômat. Nhờ có quan hệ đó mà chúng ta có thể xây dựng được những hệ thống điện tử có các chức năng biến hóa và hoạt động vô cùng linh hoạt (bằng phần mềm) mà không cần phải thay đổi linh kiện trong mạch hay trong máy.

Từ khi hình thành xã hội, con người luôn luôn tìm cách chế tạo ra các công cụ, để thực hiện các hoạt động nhằm thỏa mãn các nhu cầu cuộc sống của chính mình. Thông thường các công cụ đó được chế tạo "trực tiếp". Nhưng khi con người cần tới công cụ có các chức năng kỹ thuật, mà các chức năng đó có tính "thông minh", và hoạt động gần giống như hoạt động của con người, như : tính toán, tay máy, rôbôt, tự động hoạt động v.v... thì không thể chế tạo một cách trực tiếp được nữa. Lúc này muốn chế tạo ra các "công cụ thông minh" dùng trong cuộc sống, con người phải làm "gián tiếp" dựa vào một công cụ toán học mới, đó là toán lôgic–toán rời rạc. Cách làm đó được mô tả trên hình 2.



Hình 2. Cách tổng quát chế tạo công cụ "thông minh".

Lúc này vấn đề được đặt ra là phải tìm mọi cách để chuyển ý muốn, nhiệm vụ, chức năng thành quan hệ toán học – các phương trình toán học (toán lôgic). Rồi từ các phương trình toán học dựa vào nó lắp ráp mạch điện thực hiện chức năng hay nhiệm vụ đề ra. Đó là bản chất của vấn đề, và đó cũng chính là tư tưởng xuyên suốt quyển sách này. Như vậy môn học kỹ thuật số cũng chính là : *chỉ ra cách xây dựng cũng như phương pháp chế tạo ra công cụ điện tử thông minh*, dựa trên cơ sở của toán học lôgic. Ở đây toán lôgic là điểm tựa, là bàn đạp, là cầu nối "gián tiếp" để chế ra công cụ thông minh, và chỉ có nắm vững toán lôgic–đại số Boole mới có thể hiểu được bản chất của hệ thống số và phương pháp chế tạo hệ thống số mà thôi.

Mục đích của phần I này là giới thiệu bản chất của cấu trúc đại số để có thể hiểu được về lý thuyết hàm rời rạc. Toán học có nhiều ngành nhưng ta xuất phát từ lý thuyết tập hợp vì qua đó có thể mô tả được các chức năng kỹ thuật, đồng thời từ lý thuyết tập hợp có thể diễn đạt ý muốn của con người và chuyển nó thành quan hệ toán học. Ở đây khi nói đến quan hệ hai ngôi giữa hai tập và trên cùng một tập làm cơ sở để hiểu bản chất cấu tạo ra phép toán cũng như để hiểu một tương tác kỹ thuật, những quan hệ sắp xếp và tập sắp xếp cũng như các tính chất của các quan hệ của chúng mà nhờ đó có quan hệ trật tự đã chọn được các thuộc tính cần thiết trong hệ thống. Một trong những trật tự điển hình là lý thuyết về Dàn cũng sẽ được giới thiệu trong phần này, vì nó là cơ sở lựa chọn chủ yếu trong quá trình thực hiện các hệ thống số. Trên cơ sở hiểu được quan hệ và cấu trúc đại số, từ đó phần này cũng giới thiệu phương pháp tự tạo ra phép toán mới trong cấu trúc đại số để giải quyết nhiệm vụ nào đó, từ đó cũng nêu lên khái niệm hệ hàm đầy đủ và các hệ cơ số đếm.

Chương I

CẤU TRÚC ĐẠI SỐ

Trong chương này ta sẽ nêu lên những định nghĩa và những khái niệm cơ bản của cấu trúc đại số, từ đó có thể hiểu được đại số rời rạc và hàm rời rạc làm cơ sở để hiểu đại số Boole (đại số lôgic). Các phương pháp biểu diễn hàm Boole và ứng dụng của nó, cũng như cấu tạo một số mạch lôgic cơ bản điển hình. Những vấn đề về lý thuyết tập hợp và ứng dụng của nó trong việc mô tả một nhiệm vụ kỹ thuật.

§1.1. NHỮNG KHÁI NIỆM CƠ BẢN VỀ LÝ THUYẾT TẬP HỢP

1.1.1. KHÁI NIỆM VÀ CÁC CÁCH MÔ TẢ

Tập hợp là một khái niệm rất đơn giản của một tổ chức, đó là sự kết hợp hoặc ghép lại của các thực thể nào đó. Tập hợp mặc dù rất đơn giản nhưng nó là một trong những khái niệm rất quan trọng của toán học. Tập hợp đơn giản đến mức người ta không cần dùng định nghĩa mà đưa ngay các tiên đề áp đặt vào nó. Ta biết trong toán học có nhiều ngành, nhánh khác nhau, nhưng ở đây ta dựa trên cơ sở của lý thuyết tập hợp với tính chất cũng như các tiên đề của chúng để mô tả chức năng của một hệ thống kỹ thuật nào đó. Vì tập hợp là một quan hệ không có thứ nguyên nên nó có thể kết hợp giữa các đối tượng khác nhau, kết hợp giữa các chức năng khác nhau để tạo ra một quan hệ một nhiệm vụ nào đó mà ta muốn. Chẳng hạn ta có thể kết hợp giữa tập người và tập bàn ghế để tạo ra lớp học, cũng như chúng ta có thể kết hợp giữa các chức năng (vò, giũ, vắt, sấy khô) để tạo ra cái máy giặt, hoặc kết hợp giữa các chức năng chiếu sáng, quay động cơ, chuyển băng tải, đếm... để tạo ra một dây chuyền sản xuất. Vậy tổng quát ta có thể minh họa một tập bất kỳ như sau :

a) Cách 1 :

$$T = (a, b, c)$$

Trong đó : T là một tập,

a là phần tử của tập T,

b là phần tử của tập T.

Ta có thể viết : $a \subseteq T$ (phần tử a nằm trong tập T, hay tập T bao hàm phần tử a)

$a \in T$ (phần tử a thuộc tập T)

$f \notin T$ (phần tử f không nằm trong tập T)

Cách nói và thể hiện như vậy là theo cách nhìn của nhà toán học khi mô tả một tập T nào đó. Nhưng với nhà kỹ thuật ta lại có thể hiểu một tập như sau :

Máy giặt = (vò, giũ, vắt)

Như vậy ở đây tập T có thể hiểu đó chính là cái máy giặt, do đó có thể lấy tập T đại diện cho một cái máy hoặc một chức năng cụ thể, còn các chức năng vò, giũ, vắt có thể được coi như phần tử của tập T. Từ đó ta có thể nói tập T chính là một cái máy hay một cái rôbôt đồng thời các phần tử (a, b, c) của tập T đó chính là các chức năng của nhiệm vụ kỹ thuật đó.

Ta có thể viết : vò $\subseteq$ máy giặt (chức năng vò nằm trong máy giặt hay máy giặt có chức năng vò)

vò $\in$ máy giặt (chức năng vò thuộc máy giặt)

Chức năng hát $\notin$ máy giặt (hát không thuộc chức năng của máy giặt).

Qua những minh họa đơn giản trên chúng ta đã bước đầu thể hiện được ý định, nhiệm vụ của mình khi xây dựng một hệ thống nào đó dựa trên cơ sở của lý thuyết tập hợp. Đó là bước đầu rất đơn giản ta có thể chuyển ý định thành quan hệ toán học để thực hiện. Ở đây cũng cần lưu ý là còn có những ý định của con người mà chưa có cách chuyển thành quan hệ toán học.

b) Cách 2 :

Để minh họa một tập ta còn có các cách như sau :

T = (2, 4, 6, 8, 10)

Đó là cách mô tả liệt kê tường minh toàn bộ các chức năng tập T, nhưng tập T cũng có thể được mô tả dưới dạng điều kiện như sau :

T = (t | t là số chẵn $\leqslant$ 10)

Nói là : Tập T gồm các phần từ t thỏa mãn điều kiện : t là số chẵn nhỏ hơn hoặc bằng 10. Đây là cách mô tả nhiệm vụ kỹ thuật hoặc giao nhiệm vụ, nó hay được dùng trong thực tế vì ít khi người ta liệt kê cụ thể các chức năng của một hệ thống nào đó.

Ví dụ : T = (t| t thỏa mãn nhiệm vụ là làm sạch quần áo)

Máy giặt = (vò, giũ, vắt, sấy khô)

Tổng quát có thể viết :

T = (thực hiện một nhiệm vụ kỹ thuật nào đó)

Một tập mà không chứa phần tử nào gọi là một tập rỗng :

$$t = ( ) = \phi$$

Điều đó ứng với một hệ thống mà không thực hiện một chức năng nào, hiện tượng đó tương ứng với trường hợp hệ thống bị mất điện, hay một đầu dây lơ lửng không dùng làm việc gì, không nối đi đâu.

Do đặc điểm của tập hợp là có thể gộp tùy ý các thuộc tính khác nhau không phân biệt thứ nguyên, cho nên có thể gộp tập người với tập vật, nhưng điều quan trọng ở đây là kết quả của sự kết hợp lẫn các thứ nguyên đó lại cho chúng ta kết quả là một thuộc tính khác. Qua đó ta cũng thấy các phần tử của tập không có một tính chất chung nào cả, nói cách khác một phần tử của tập này có thể là một phần của một tập khác, ví dụ : tập động cơ quay ở dây chuyền có thể là phần tử ở chức năng động cơ quay trong một hệ thống nào đó.

c) Cách 3 :

Chúng ta còn có một cách khác nữa biểu hiện tập hợp, đó là các tập con như sau :

$$T = \big((a, b), (c), (d, e, f)\big) = (A, B, C)$$

Trong đó : (a, b) = A. Chúng ta có thể nói (a, b) là một tập con, còn A gọi là một khối (tập con = khối).

Từ đó ta có thể viết tập T gồm các khối A, B và C, hay tập T gồm các tập con (a,b), (c) và (d,e,f).

Khi đó có thể viết :

$$a \in A \qquad e \subseteq C \qquad A \subseteq T \qquad K \notin T \qquad d \notin A$$

Qua cách biểu diễn tập con ta có thể thấy là : một tập con có thể là một phần tử của tập lớn hơn, và một tập con có thể có một hay nhiều chức năng. Đó là cách biểu diễn toán học một tập của các tập con. Trong kỹ thuật ta có thể hiểu điều đó như sau :

(a,b) = A. Nếu ký hiệu a là chức năng hình và b là chức năng tiếng, thì A chính là cái TV. Vậy tập con có thể hiểu đó chính là một cái máy nhỏ hoàn chỉnh.

(c) = B. Nếu gọi c là chức năng hát thì B chính là cái cassette.

Còn ký hiệu d là chức năng vò, e là chức năng giũ, f là chức năng vắt thì C chính là cái máy giặt hoàn chỉnh.

Như vậy khi ta biểu diễn tập T dưới dạng tập con điều đó đồng nghĩa với việc xếp (ghép) các chức năng lớn – là các máy con hoàn chỉnh – vào một hệ thống hay vào một chỗ, khi đó tập T được biểu diễn :

$$T = (TV, cassette, máy giặt)$$

Lúc này tập T sẽ thỏa mãn các chức năng nhỏ (a,b,c,d,e,f) chứ ta không đi lắp một hệ thống có chừng đó chức năng. Vậy ý muốn mua các máy về cho một gia đình như trong cuộc sống đời thường có thể được minh họa ở tập T dưới dạng các tập con. Đồng

thời trong thực tế kĩ thuật cũng vậy, không phải lúc nào ta cũng đi đi lắp toàn bộ một hệ thống với đầy đủ mọi chức năng nhỏ bé nhất của nó, mà có thể mua bổ xung vào hệ thống những chức năng hoàn chỉnh đã có sẵn. Với cách biểu diễn tập con sẽ tồn tại các quan hệ tất nhiên sau :

$a \subseteq A$ $\qquad$ $A \subseteq T \rightarrow a \subseteq T$

$\phi \subseteq T$ $\qquad$ (tập rỗng thuộc tập T)

$T \subseteq T$ $\qquad$ (T là tập con của chính nó)

Nếu ta có quan hệ N là tập con của N ($N \subseteq M$) mà M lại là tập con của N ($M \subseteq N$), khi đó ta có thể nới M bằng N ($N = M$). Khi ta kết luận hoặc gọi chính xác A là tập con của T ($A \subseteq T$), điều đó có nghĩa là mọi chức năng của A thì tập T đều có, còn A so với T thì A thiếu ít nhất một phần tử - một chức năng, hay nói cách khác là có ít nhất một phần tử của T không thuộc A, khi đó có thể viết :

$$A \subseteq T$$

Như vậy trong tiết này đã cho ta những bước đầu tiên đơn giản nhất cho phép mô tả ý định, chức năng của mình dưới dạng toán học - toán tập hợp.

1.1.2. TẬP VÀ CÁC QUAN HỆ TOÁN HỌC

Chúng ta đã hiểu tập hợp chính là một chức năng lớn - một hệ thống kỹ thuật nào đó gồm các chức năng nhỏ thành phần. Như vậy sự tác động toán học lên tập cũng như giữa các tập với nhau, sẽ cho ta những hiểu biết về chúng nhờ đó có được các kết luận kỹ thuật kèm theo. Trong tiết này sẽ đề cập đến các phép toán cơ bản của tập hợp.

1.1.2.1. Phép toán hợp của hai tập ký hiệu là ($\cup$)

Kết quả của phép toán hợp giữa hai tập A và B là một tập gồm có các phần tử của tập A hoặc các phần tử của tập B, nó là một tập bao gồm các phần tử của cả hai tập A và B. Điều đó được định nghĩa như sau :

$$A \cup B = \left(x \,\middle|\, \begin{array}{l} x \in A \\ x \in B \text{ hoặc } x \in A \text{ và } B \end{array} \right)$$

Ví dụ : Cho hai tập hợp A và B như sau :

$$A = (a, b, c)$$

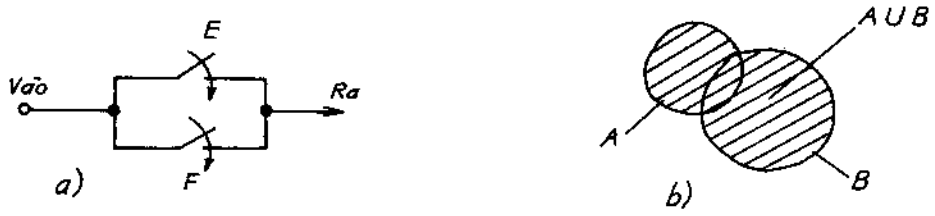
$$B = (e, d, e)$$

Vậy : $\qquad A \cup B = (a, b, c, d, e)$

$$A \cup \varnothing = (a, b, c) \cup ( ) = (a, b, c)$$

Kết quả đó có thể được minh họa bằng đồ thị Viên và mạch điện đơn giản trên hình 2.1 sau :

Có thể nới công tắc E đóng hoặc công tắc F đóng hoặc cả hai công tắc E và F cùng đóng thì tín hiệu được truyền qua, đó chính là mạch điện của phép toán Hợp của hai công tắc E và F. Việc minh họa đó mặc dù rất đơn giản nhưng ta thấy được một



Hình 1.1. Phép toán hợp.

điều quan trọng là : Một phép toán có một mạch điện tương ứng và ngược lại, đó là điều mà không phải quan hệ toán học nào cũng có thể có được.

1.1.2.2. Phép toán giao của hai tập ký hiệu là (∩)

Kết quả của phép toán giao giữa hai tập A và B là một tập gồm có các phần tử của tập A và B, nó là một tập bao gồm các phần tử nằm trong phần giao nhau, phần chung của hai tập đó. Điều đó được định nghĩa như sau :

$$A \cap B = (x \mid x \notin A,\ x \notin B \text{ mà } x \in A \text{ và } B)$$

Ví dụ : Có các tập hợp như sau :

$$A = (a, b, c)$$

$$B = (c, d, e)$$

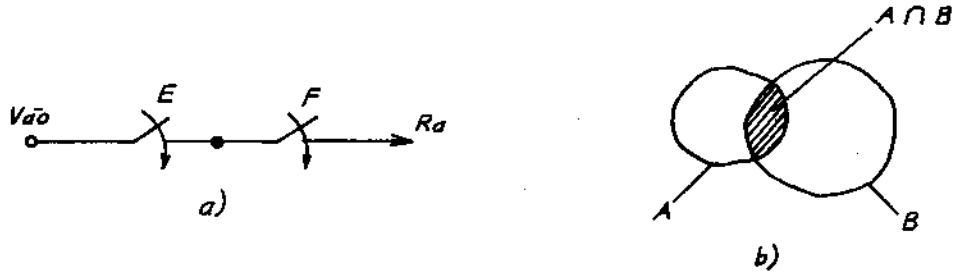
$$D = (r, s, t)$$

Vậy : $A \cap B = (a, b, c) \cap (c, d, e) = (c)$

$A \cap D = (a,b,c) \cap (r, s, t) = \varnothing$

$A \cap \varnothing = (a, b, c) \cap ( ) = \varnothing$

Kết quả đó được minh họa bằng đồ thị Viên và mạch điện đơn giản được biểu diễn trên hình 1.2.



Hình 1.2. Phép toán giao.

Chỉ khi công tắc E và công tắc F cùng đóng thì tín hiệu mới được truyền qua. Đó là mạch điện mắc nối tiếp của hai công tắc E và F thực hiện chức năng của phép toán VÀ (phép toán giao).

1.1.2.3. Phép toán hiệu của hai tập ký hiệu là (\ )

Kết quả của phép toán hiệu giữa hai tập A và B là một tập gồm có các phần tử của tập A mà không chứa các phần tử của tập B. Biểu đồ được định nghĩa như sau :

$$A \setminus B = (x \mid x \in A \text{ nhưng } x \notin B \text{ và } x \notin A \text{ và } B)$$

Ví dụ : Cho các tập sau :

$$A = (a, b, c)$$

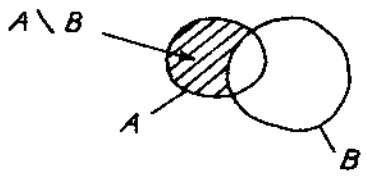
$$B = (c, d, e)$$

$$C = (r, s, t)$$

Vậy : $A \setminus B = (a, b, c) \setminus (c, d, e) = (a, b)$

$A \setminus C = (a, b, c) \setminus (r, s, t) = (a, b, c)$

Kết quả của phép toán hiệu được minh họa bằng đồ thị Viên trên hình 1.3.



Hình 1.3. Phép toán hiệu.

Nhận thấy các phép toán mới gồm có các kí hiệu không quen thuộc đối với chúng ta, điều đó làm ta khó nhận ra các quan hệ hàm số khi biểu diễn trong phương trình. Vì vậy ở đây ta có quy ước là : Dùng lại kí hiệu toán cũ nhưng với nghĩa mới các phép toán, cụ thể là phép toán hợp ứng với phép toán cộng ($\cup$ = +), phép toán giao ứng với phép toán nhân ($\cap$ = ×) và phép toán hiệu tập hợp ứng với phép toán trừ (\ = −). Mỗi một phép toán đều có tính chất riêng đó là các tiên đề của chúng, điều đó được thể hiện trong các quan hệ sau :

1.1.2.4. Các tính chất hay tiên đề của các phép toán

- Tính chất lũy đẳng :

$$A \cup A = A \qquad A + A = A$$

$$A \cap A = A \qquad A \times A = A$$

- Tính chất giao hoán :

$$A \cup B = B \cup A \qquad A + B = B + A$$

$$A \cap B = B \cap A \qquad A . B = B . A$$

- Tính chất kết hợp :

$$A \cup (B \cup C) = (A \cup B) \cup C \qquad A + (B + C) = (A + B) + C$$

$$A \cap (B \cap C) = (A \cap B) \cap C \qquad A . (B . C) = (A . B) . C$$

- Tính chất phân phối :

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C) \qquad A . (B + C) = (A . B) + (A . C)$$

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C) \qquad A + (B . C) = (A + B) . (A + C)$$

- Tính chất loại trừ (tính chất Nuốt) :

$$A \cap (A \cup B) = A \qquad A . (A + B) = A$$

$$A \cup (A \cap B) = A \qquad A + (A . B) = A$$

- Công thức De Morgan :

$$X \setminus (A \cup B) = (X \setminus A) \cap (X \setminus B) \qquad X - (A + B) = (X - A) \cdot (X - B)$$

$$X \setminus (A \cap B) = (X \setminus A) \cup (X \setminus B) \qquad X - (A \cdot B) = (X - A) + (X - B)$$

Các tính chất trên có thể dựa vào các định nghĩa của các phép toán để chứng minh. Sau đây ta sẽ chứng minh một vài tính chất trên.

Ví dụ : Chứng minh tính chất phân phối đối với phép nhân (.)

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

Để chứng minh ta dựa vào quan hệ : Nếu $N \subseteq M$ (N là tập con của M) mà $M \subseteq N$ (M lại là tập con của N) thì có thể viết $N = M$. Vậy ta phải tìm cách chứng minh :

$$A \cdot (B + C) \subseteq (A \cdot B + A \cdot C)$$

$$(A \cdot B + A \cdot C) \subseteq A \cdot (B + C)$$

Thật vậy, giả sử x là một phần tử của tập $(A \cdot (B + C))$ nghĩa là $x \in A$ đồng thời $x \in B$ hoặc $x \in C$. Nếu $x \in B$ có nghĩa là $x \in A.B$

$x \in C$ có nghĩa là $x \in A.C$

Dẫn đến x phải thuộc $A \cdot B$ hoặc $A \cdot C$ tức là thỏa mãn :

$$x \in \big((A \cdot B) + (A \cdot C) \big)$$

Nói cách khác $A \cdot (B + C)$ là tập con của $\big((A \cdot B) + (A \cdot C)\big)$. Vậy ta có :

$$A \cdot (B + C) \in (A \cdot B + A \cdot C)$$

Ngược lại, giả sử x là một phần tử của tập (A.B + A.C), tức là $x \in A.B$ hoặc $x \in A.C$. Nói cách khác như vậy là x có mặt trong hai tập A và tập B hoặc x có mặt trong hai tập A và tập C. Điều đó có nghĩa là $x \in A$ đồng thời x thuộc tập B hoặc tập C, tức là :

$$x \in A \cdot (B + C)$$

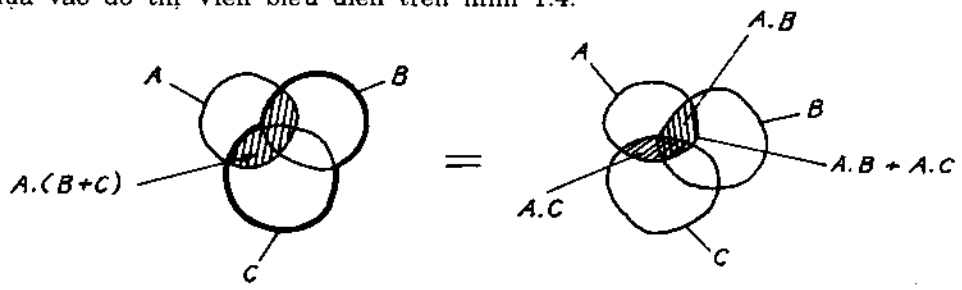
Như vậy có thể nói (A.B + A.C) là tập con của tập A.(B+C), ta có :

$$(A \cdot B + A \cdot C) \subseteq A \cdot (B + C)$$

Từ hai kết quả thu được ở trên ta có kết quả sau :

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

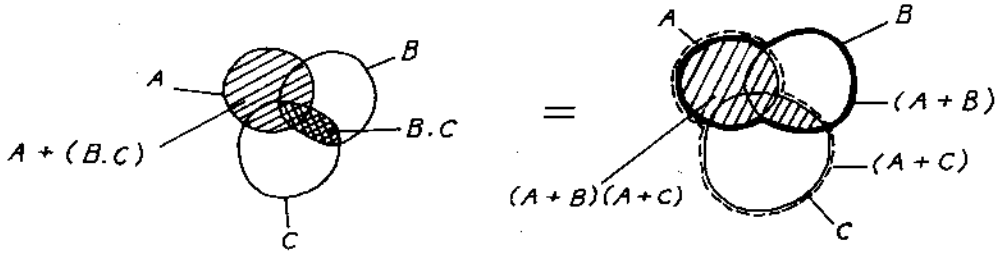
Đó là điều ta phải chứng minh. Bằng cách khác ta cũng có thể nhận được kết quả đó dựa vào đồ thị Viên biểu diễn trên hình 1.4.



Hình 1.4. Tính phân phối của phép nhân.

Ví dụ : Có thể chứng minh tính chất phân phối của phép toán cộng (+) dựa vào đồ thị Viên rất dễ dàng như trên hình 1.5 sau :

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



Hình 1.5. Tính phân phối của phép cộng

1.1.2.5. Phép toán hiệu đối xứng

Phép toán hiệu đối xứng giữa hai tập kí hiệu là $\oplus$.

Kết quả của phép toán hiệu đối xứng giữa hai tập A và B là một tập gồm có các phần tử của tập A và các phần tử của tập B nhưng không chứa các phần tử của tập A và B, điều đó được định nghĩa như sau :

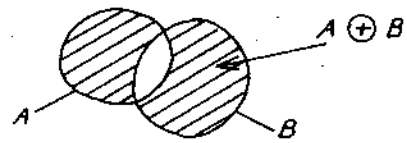
$$A \oplus B = (x \mid x \in A, x \in B \text{ nhưng } x \notin A . B)$$

Ví dụ : Có các tập :

$A = (a, b, c)$

$B = (c, d, e)$

$C = (r, s, t)$



Hình 1.6. Phép toán hiệu đối xứng.

Vậy : $A \oplus B = (a, b, c) \oplus (c, d, e) = (a, b, d, e)$

$A \oplus C = (a, b, c) \oplus (r, s, t) = (a, b, c, r, s, t)$

$A \oplus A = (a, b, c) \oplus (a, b, c) = \Phi$

Ta có thể minh họa kết quả của phép toán hiệu đối xứng ($\oplus$) của hai tập bằng đồ thị Viên trên hình 1.6.

1.1.2.6. Quan hệ phân phối của một tập kí hiệu F(A)

Kết quả của quan hệ phân phối của một tập A là một tập gồm các tập con có thể có của A, nó cho chúng ta biết mọi khả năng phân phối (ghép nối) của các phần tử với nhau thuộc tập A. Điều đó được định nghĩa như sau :

$$F(A) = (X \mid X \subseteq A) \qquad X \text{ là tập con của } A$$

Ví dụ : Có tập $A = (a, b, c)$

Vậy : $P(A) = \big((\Phi), (a), (b), (c), (a,b), (a,c), (b,c), (a,b,c)\big)$

Quan hệ này cho ta một hiểu biết quan trọng về mọi khả năng nhóm họp của các phần tử của tập A với nhau, mà khi biết rõ mọi khả năng nhóm họp của các phần tử

(các chức năng) trong một tập (một hệ thống) chúng ta sẽ chỉ ra được các kết luận kĩ thuật chính xác. Từ đặc điểm của quan hệ phân phối ta có kết quả là : Nếu tập A có n phần tử thì P(A) sẽ có đúng 2^n khả năng nhóm họp.

§1.2. QUAN HỆ VÀ HÀM SỐ

Trong tiết này sẽ nói đến các quan hệ đơn giản nhất cấu tạo nên hàm số, cũng như tương tác đơn giản nhất để cấu tạo các chức năng nhiệm vụ kỹ thuật. Đó là quan hệ hai ngôi của toán học và tương tác hai ngôi trong việc thực hiện nhiệm vụ kỹ thuật.

1.2.1. QUAN HỆ HAI NGÔI TRONG PHÉP TOÁN VỚI TƯƠNG TÁC KỸ THUẬT

Hàm số là một khái niệm mà bất cứ quan hệ toán học nào cũng có. Ở đây ta phải chỉ rõ bản chất của hàm số là gì? Có thể hiểu đơn giản hàm số là sự ràng buộc, sự phụ thuộc tương ứng một một giữa biến số x với hàm số f trong một quan hệ toán học nào đó, và trong kỹ thuật ta cũng có sự ràng buộc, sự phụ thuộc tương ứng để thực hiện một nhiệm vụ hay chế ra một sản phẩm nào đó, phải tuân theo một trật tự thứ tự nhất định mới thực hiện được. Ví dụ như dây chuyền sản xuất, hay trình tự các lệnh trong chương trình... Ở đây nếu ta coi hàm số là một chức năng, một nhiệm vụ kỹ thuật thì hàm số không còn là một quan hệ toán học nữa, mà nó là một trật tự để tạo ra một sản phẩm phục vụ cho con người. Hiểu được như vậy cũng là bước đầu tiến đến tư tưởng chủ đạo của quyển sách này là chuyển một ý muốn, một chức năng hay một nhiệm vụ kỹ thuật thành một quan hệ toán học để đi tới thực hiện nó.

Trước hết ta cần hiểu quan hệ đơn giản nhất tạo nên hàm số, đó chính là quan hệ của một phép toán, là sự ràng buộc lẫn nhau tối thiểu giữa các đối tượng vật lý để tạo ra một nhiệm vụ kỹ thuật đơn giản. Như vậy có thể coi mỗi phép toán tương ứng với một tương tác kỹ thuật đơn giản nhất. Chúng ta đều biết quan hệ của mỗi phép toán cần có hai đối tượng tác động để cho một kết quả. Đó chính là quan hệ hai ngôi, khái niệm đó được minh họa trong ví dụ sau :

Ví dụ : Ta cần thực hiện phép cộng

5 + 3 = 8 (là kết quả ta cần)

Nhưng 5 + = Không có kết quả

+ 3 = Không có kết quả

Ở đây cụ thể là ngôi 5 và ngôi 3 là hai ngôi cần có (hai phần tử) của phép toán cộng (cũng như của một phép toán bất kỳ nào đó) mà chúng không thể thiếu một trong hai ngôi đó.

Trong kỹ thuật tương tác hai ngôi đơn giản nhất ký hiệu là (*) có thể được hiểu như sau :

Công tắc * Đèn điện = Ánh sáng (là kết quả ta muốn).

Ta thấy hai ngôi, ngôi công tắc và ngôi đèn điện không thể thiếu trong quan hệ tương tác của kết quả đó. Điều cần chú ý ở đây là : trong kỹ thuật để ra một kết quả như mong muốn cần có những tương tác nhiều ngôi hơn. Ở đây ta chỉ nói tới tương tác đơn giản nhất : Tương tác hai ngôi.

1.2.2. QUAN HỆ HAI NGÔI GIỮA HAI TẬP

Sau khi hiểu quan hệ hai ngôi giữa hai phần tử ta có thể mở rộng thành quan hệ hai ngôi giữa hai tập. Nói tới quan hệ hai ngôi giữa hai tập cũng chính là tương tác giữa hai tập đó với nhau để cho ra một kết quả thỏa mãn ý đồ cho trước. Tìm hiểu quan hệ hai ngôi giữa hai tập cũng chính là tương tác giữa hai tập đó với nhau để cho ra kết quả thỏa mãn ý đồ cho trước. tìm hiểu quan hệ hai ngôi giữa hai tập cũng chính là tương tác giữa tập điều khiển và tập chấp hành trong một nhiệm vụ kỹ thuật.

Định nghĩa 1.2.1. Cho n tập A1, A2, ..., An, tích trực tiếp của n tập trên được tính như sau :

$$(A_1 \times A_2 \times \ldots \times A_n) = \{(a_1,a_2,\ldots,a_n) \mid a_1 \in A_1,\ a_2 \in A_2,\ldots,\ a_n \in A_n\}$$

Ví dụ : Có các tập A = (a, b)

B = (a, c, d)

Vậy tích trực tiếp (tích chập) của hai tập đó được tính

$$(A \times B) = \{(a,a)\ (a,c)\ (a,d)\ (b,a)\ (b,c)\ (b,d)\}$$

Kết quả cho ta biết mọi khả năng ghép phần tử của hai tập, nhờ đó có được các nhận xét chính xác trong sự lựa chọn.

Định nghĩa 1.2.2. Quan hệ hai ngôi giữa hai tập A và tập B có kết quả ký hiệu R là một tập con của tích trực tiếp (A x B) :

$$R \subseteq (A \times B)$$

Ở đây A là tập điều hành, B là tập chấp nhận còn R là tập thỏa mãn quan hệ tương tác giữa hai tập đó.

Ví dụ : A = (a, b, c, d) – tập các sinh viên tốt nghiệp.

B = (M, N, O, P, Q) – tập các cơ quan cần sinh viên.

Giả sử tập thỏa mãn quan hệ tương tác R cho trước như sau :

$$B = \{(a,N),\ (b,P),\ (c,M),\ (d,Q)\}$$

dễ dàng nhận thấy $B \subset (A \times B)$, quan hệ đó của R có thể được biểu diễn trên bảng ở hình 1.7.

Tọa độ t_{ij} của bảng được xác định như sau :

$t_{ij} = 1$, nếu $(A_i, B_j) \in R$

$t_{ij} = 0$, nếu $(A_i, B_j) \notin R$

Nhìn vào bảng đó ta thấy được quan hệ phân công công tác thỏa mãn R cũng như các quan hệ không thỏa mãn R. Khi nhận được kết quả R = (a,N) ta hiểu rằng sinh viên a được phân công về cơ quan N.

X	M	N	O	P	Q
a	0	1	0	0	0
b	0	0	0	1	0
c	1	0	0	0	0
d	0	0	0	0	1

Hình 1.7. Bảng quan hệ hai ngôi phân phối người.

Nhưng nếu ta coi tập A = (a,b,c,d) là tập các công tắc điều khiển, còn tập B = (M,N,O,P,Q) là tập các động cơ quay hay đèn báo sáng chấp hành thì tập R chính là tập thỏa mãn sự hoạt động giữa tập điều khiển A và tập chấp hành B. Nếu nhận được kết quả R = (a,N) thì biết rằng công tắc a bật thì động cơ N sẽ hoạt động. Vậy tập B chính là tập dự kiến các hoạt động kĩ thuật được cho trước, và quan hệ hai ngôi giữa hai tập có thể dùng để mô tả một nhiệm vụ kĩ thuật nào đó.

1.2.3. QUAN HỆ HAI NGÔI TRÊN MỘT TẬP

Trong cuộc sống đời thường, trong các quan hệ toán học và trong các nhiệm vụ kỹ thuật chúng ta thường gặp quan hệ hai ngôi nhiều hơn cả. Vậy quan hệ hai ngôi trên một tập là gì, tính chất của nó ra sao và ứng dụng của nó để làm gì sẽ được giải đáp trong phần này.

Trong cuộc sống quan hệ hai ngôi trên một tập thấy được thật là đơn giản và dễ hiểu bởi vì lấy hai quan hệ vợ chồng gia đình làm ví dụ, quan hệ đó chính là một quan hệ hai ngôi trên. Trong toán học quan hệ hai ngôi trên một tập có thể chỉ ra dễ dàng bằng ví dụ minh họa sau.

Ví dụ : Ta có N = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ..., n), đó là tập các số nguyên dương.

Vậy : 2 + 6 = 8 Mà 2∈N, 6∈N. Ta có thể nói phép toán cộng của quan hệ hai ngôi 2 và 6 là quan hệ hai ngôi trên một tập N các số nguyên dương.

Định nghĩa 1.2.3. Một quan hệ hai ngôi từ một tập A lên chính bản thân nó được gọi là một quan hệ hai ngôi trên A.

Ví dụ cho A là một tập các số nguyên dương. Chúng ta có thể minh họa một quan hệ hai ngôi R = (x, y) với x∈A, y∈A.

Cho trước quanh hệ (x, y) ∈ R nếu và chỉ nếu (x−y) = 5. Vậy ta có các quan hệ sau đây thỏa mãn (7,2) ∈ R, (9,4) ∈ R, (28,23) ∈ R, ... ta thấy có vô số các quan hệ hai ngôi của (x và y) thỏa mãn R cho trước, đồng thời ta cũng thấy có vô số các quan hệ của (x và y) không thỏa mãn R như (3,5) ∈ R, (7,4) ∈ R,...

Quan hệ (x, y) ∈ R nếu cho x = y, có thể viết (x, x)∈R thì được gọi là quan hệ đồng nhất hay quan hệ đường chéo. Ví dụ cho A = (a, b, c, d, e) quan hệ đường chéo sẽ minh họa trên hình 1.8.

Các quan hệ hai ngôi trên A có những tính chất sau :

a) Nó là quan hệ phản xạ nếu :
(x, y) ∈ R mà (x = y) kéo theo (x, x)∈R với mọi x∈A.

b) Là quan hệ đối xứng nếu :
(x, y)∈R thì có quan hệ (y, x)∈R với mọi x, y thuộc A.

R	a	b	c	d	e
a	1	0	0	0	0
b	0	1	0	0	0
c	0	0	1	0	0
d	0	0	0	1	0
e	0	0	0	0	1

Hình 1.8. Quan hệ hai ngôi trên một tập.

c) Là quan hệ phản xứng (không đối xứng) nếu :
(a, y)∈R thì cũng có quan hệ (m, n) ∈R với mọi x, y, m, n∈A

d) Quan hệ bắc cầu nếu :
(x, y) ∈R mà (y, z)∈R thì kéo theo (x, z)∈R.

Ví dụ cho A là một tập các số nguyên, chúng ta muốn có một quan hệ hai ngôi R trên A thỏa mãn (a, y) ∈R nếu (x−y) = 0, với x∈A và y∈A, rõ ràng quan hệ đó chỉ thỏa mãn khi x = y, vậy có thể nói quan hệ (x, y) ∈R là một quan hệ phản xạ. Trường hợp khác cho (x, y) ∈R nếu và chỉ nếu x > y thì dễ dàng nhận thấy R không phải là quan hệ phản xạ. Qua đó chúng ta nhận thấy là quan hệ hai ngôi trên một tập chỉ xảy ra bốn tính chất đó mà thôi.

Một trong những quan hệ hai ngôi quen thuộc điển hình mà chúng ta ai cũng biết đó là bảng cửu chương, nó là bảng cơ sở cho phép ta tính toán. Đó chính là quan hệ hai ngôi trên một tập B các kí hiệu chữ số từ 0 đến 9.

N = (1, 2, 3, 4, 5, 6, 7, 8, 9)

Quan hệ hai ngôi của tập N chúng ta đều biết như trên hình 1.9.

+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15	16	17	18

Hình 1.9. Quan hệ 2 ngôi trên một tập−Phép cộng.

Điều quan trọng nhất khi tìm hiểu quan hệ hai ngôi trên một tập là ứng dụng vào vấn đề tự tạo ra phép toán, tự định nghĩa lấy phép toán để giải quyết một nhiệm vụ nào đó. Điều này sẽ được minh họa cụ thể trong những phần sau.

1.2.4. CÁC QUAN HỆ TƯƠNG ĐƯƠNG VÀ PHÂN HOẠCH

Trong cuộc sống cũng như trong kỹ thuật chúng ta thường nói nhiệm vụ A tương đương với nhiệm vụ B, hay rôbôt A hoạt động tương đương với hoạt động của rôbôt B,

nhưng nói sao cho chính xác và hiểu sao cho đúng thì cần phải hiểu rõ quan hệ tương đương là gì, và điều đó được mô tả chặt chẽ trong quan hệ toán học của phần này.

Định nghĩa 1.2.4. Một quan hệ hai ngôi trên một tập được gọi là một quan hệ tương đương nếu nó là quan hệ phản xạ, đối xứng và bắc cầu. Ví dụ : Cho A là một tập các dãy số 0 và 1 có độ dài bằng hai :

$$A = (00,\ 01,\ 10,\ 11)$$

Một quan hệ R trên A gọi là tương đương nếu thỏa mãn $(x, y) \in R$ với quan hệ x và y giống nhau. Quan hệ đó dễ dàng được minh họa trên hình 1.10.

Ví dụ : Có một quan hệ hai ngôi trên một tập A được mô tả ở hình 1.11, với A = (a, b, c, d, e, f). Ta có thể dễ dàng kiểm tra thấy quan hệ R = (x, y) là một quan hệ tương đương. Với $x \in A$, $y \in A$.

R	00	01	10	11
00	1	1	1	1
01	1	1	0	0
10	1	0	1	0
11	1	0	0	1

Hình 1.10. Quan hệ tương đương.

R	a	b	c	d	e	f
a	1	1	0	0	0	0
b	1	1	0	0	0	0
c	0	0	1	0	0	0
d	0	0	0	1	1	1
e	0	0	0	1	1	1
f	0	0	0	1	1	1

Hình 1.11. Quan hệ tương đương.

Một trong những khái niệm quan trọng của phần này là phân hoạch Π. Phân hoạch Π thực chất là sự nhóm họp tùy ý các phần tử của một tập A nào đó, chính sự nhóm họp tùy ý đó sẽ cho phép ta chọn được các phân hoạch (các nhóm) thích hợp trong tập A. Nếu ta có một rôbôt có tập chức năng là A thì việc nhóm họp (phân hoạch) tùy ý các chức năng của nó, có thể cho phép ta chọn được các chức năng có cùng thuộc tính vào trong một nhóm, từ đó sẽ cho phép ta xử lí kĩ thuật thích hợp. Ý nghĩa là như vậy trong kĩ thuật nhưng ở đây ta phải nói chính xác bằng quan hệ toán học.

Định nghĩa 1.2.5. Một phân hoạch Π trên một tập hợp A là một tập của các tập con của A, có quan hệ sau :

$$\Pi = (X \mid X \subseteq A)$$

Ví dụ : A = (a, b, c, d, e, f, g, h, i, k) Từ A ta có các phân hoạch

$$\Pi_1 = \big((a, b),\ (c),\ (e, f)\big) = (\overline{a, b},\ \overline{c},\ \overline{e, f}) = (S, T, U)$$

$$\Pi_2 = \big((a, b),\ (c),\ (d, e)\big) = (\overline{a, b, c},\ \overline{d, e}) = (M, N)$$

$$\Pi_3 = \big((g),\ (h, i),\ (k)\big) = (\overline{g},\ \overline{h, i},\ \overline{k}) = (X, Y, Z)$$

Ta thấy Π_1, Π_2, Π_3 đó là những nhóm họp tùy ý của tập A, nhưng quan sát kỹ ta thấy phân hoạch Π_1 và Π_2 có những nhóm họp con giao nhau, còn với các phân hoạch

Π_1 và P_3 thì không thấy có phần chung nào. Trường hợp này ta nói phan hoạch Π_1 và Π_3 là phân hoạch đơn tách biệt, tức là các chức năng trong chúng không lẫn lộn vào với nhau, điều đó phù hợp với thực tế kỹ thuật là chức năng động cơ quay không liên quan gì tới chức năng báo sáng. Vì vậy trong kỹ thuật ta chỉ nói tới các phân hoạch đơn tách biệt, được định nghĩa như sau :

$$A = \bigcup_{i=1}^{n} \Pi_1 \text{ và } \Pi_1.\Pi_j = \varnothing, \text{ với } \Pi_i, \Pi_j \in A$$

Nới một cách khác hợp theo i từ 1 tới n của Π_i bằng A, còn tích của các Π_i thành phần với nhau bằng rỗng, quan hệ đó cũng được minh họa trên hình 1.12.

Hình 1.12. Phân hoạch đơn tách biệt.

1.2.5. QUAN HỆ SẮP XẾP VÀ TẬP SẮP XẾP

Sự phát triển của tự nhiên nói chung cũng như trong kỹ thuật nói riêng đều dựa và trật tự sắp xếp, vì khi có trật tự có sắp xếp thì có sự lựa chọn và phát triển. Nhưng để sắp xếp được một tập lại không đơn giản để chúng ta lựa chọn, mà mỗi một tập lại tương ứng với một rôbôt hay một hệ thống cần phải lựa chọn. Để có thể sắp xếp được một tập ta dựa vào định nghĩa sau.

Định nghĩa 1.2.6. Một quan hệ $\leqslant$ (sắp xếp) trên một tập A gọi là một tập sắp xếp riêng trên trên A nếu nó thỏa mãn các tiên đề sau đây :

- Tính phản xạ nghĩa là :

  Với mọi $x \in A$ ta có trật tự $x \leqslant x$ (x sắp xếp với x)

- Tính đối xứng tức là :

  Nếu $x, y \in A$, mà $x \leqslant y$ và $y \leqslant x$ thì kéo theo $x = y$.

- Tính bắc cầu nghĩa là :

  Nếu $x, y, z \in A$, $x \leqslant y$ và $y \leqslant z$ thì kéo theo $x \leqslant z$.

Với một tập bất kỳ nếu thỏa mãn định nghĩa trên thì gọi là một tập sắp xếp riêng, và quan hệ R = (x,y) gọi là điều kiện thực hiện trật tự sắp xếp riêng trên A.

Trong tập sắp xếp riêng A ta gặp trường hợp đặc biệt gọi là một quan hệ liên thông, khi đó một quan hệ R trên tập A là liên thông nếu $x,y \in A$ kéo theo có quan hệ $x \leqslant y$ hoặc $y \leqslant x$.

Định nghĩa 1.2.7. Một quan hệ $\leqslant$ trên tập A gọi là một sắp xếp toàn phần của a nếu thỏa mãn :

- Nó là một tập sắp xếp riêng A.

- Đồng thời nó thỏa mãn thêm tính liên thông tức là :

Nếu x,y∈A kéo theo trật tự x ≤ y hoặc y ≤ x.

Một tập A trên nó thỏa mãn quan hệ liên thông R mà lại là một quan hệ sắp xếp riêng thì được gọi là một quan hệ sắp xếp toàn phần. Các quan hệ đó được minh họa như sau.

Ví dụ : cho A = (tập các số chia của 36)

A = (1, 2, 3, 4, 6, 9, 12, 18, 36)

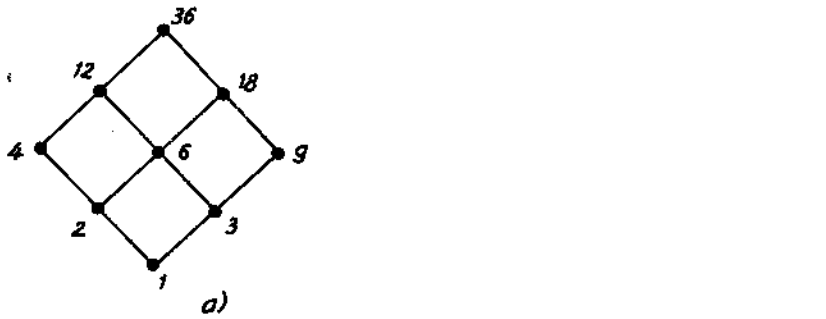
Cho trước quan hệ (x, y)∈R và đồng thời x là số chia của y, thì ta có quan hệ x ≤ y. Để thỏa mãn được quan hệ đó ta có các kết quả như sau :

(2,4) ∈R vậy 2 ≤ 4 x ≤ y

(3,9) ∈R 3 ≤ 9

(6,18) ∈R vậy 6 ≤ 18 ...

và ta còn nhận được nhiều các quan hệ khác nữa. Toàn bộ các quan hệ thỏa mãn sẽ được minh họa trên hình 1.13.



Hình 1.13. Quan hệ sắp xếp.

Nếu cho A = (tập các số chia của 27)

A = (1, 3, 9, 27)

Cũng với quan hệ (x, y) ∈R và x là số chia của y, thì x ≤ y.

Ta nhận được kết quả : (3,9) ∈R vậy 3 ≤ 9

(9,27) ∈R 9 ≤ 27 ...

chúng thỏa mãn quan hệ sắp xếp riêng, nhưng lại có quan hệ cũng thỏa mãn như sau :

(3,3) ∈R vậy 3 ≤ 3 hay x ≤ y và y ≤ x

(9,9) ∈R 9 ≤ 9 x ≤ y y ≤ x,

như vậy cho thấy tập A vừa thỏa mãn quan hệ liên thông vừa là tập sắp xếp riêng, nên ta có thể nói A là một tập có quan hệ sắp xếp toàn phần, nó được minh họa trên hình 1.13b.

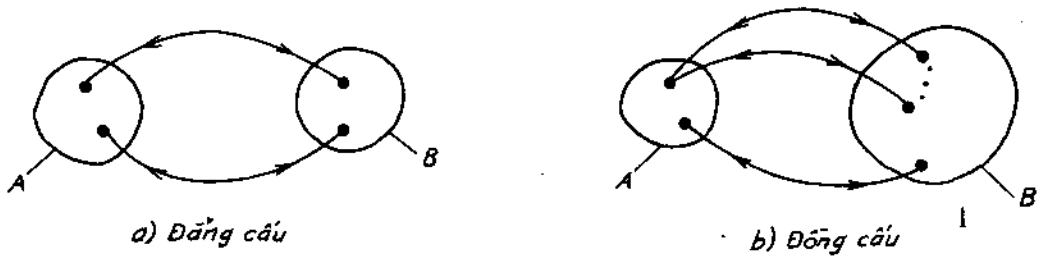
Đã là một tập sắp xếp thì luôn luôn có cực đại cực tiểu, điều đó cho phép chọn lựa dễ dàng. Trong một tập (một dãy) sắp xếp thì khái niệm về phần tử lớn nhất và phần tử cực đại cũng như khái niệm về phần tử bé nhất và phần tử cực tiểu là như nhau.

1.2.6. ĐỒNG CẤU VÀ ĐẲNG CẤU

Khi nói đến một hệ thống chức năng hay một rôbôt, ta thường muốn so sánh xem nó có gì giống nhau, khác nhau hay tương ứng với một hệ thống nào đó. Để thể hiện chính xác điều đó ta phải dùng quan hệ đồng cấu và đẳng cấu (bình đẳng về kết cấu) trong toán học, đó là một khái niệm quan trọng của toán học hiện đại. Trong lý thuyết tập hợp thì hai tập hợp đẳng cấu với nhau là hai mô hình cụ thể của cùng một khái niệm toán học trừu tượng. Hai tập đẳng cấu với nhau thì chúng hoàn toàn giống nhau về mọi phương diện các tính chất được mô tả. Để phân biệt những khái niệm đó ta dựa vào định nghĩa sau.

Định nghĩa 1.2.8. Cho A và B là hai tập hợp (hai hệ thống). Nếu tồn tại tương ứng một–một giữa A và B thì tương ứng đó gọi là đẳng cấu, A và B được gọi là có tính đẳng cấu với nhau. Nếu sự tương ứng đó không phải là một–một mà là nhiều lên từ A sang B hoặc,ngược lại thì tương ứng đó gọi là đồng cấu còn A và B gọi là có tính đồng cấu với nhau.

Để hiểu thêm định nghĩa quan hệ đồng cấu và đẳng cấu được mô tả trên hình 1.14.



Hình 1.14. Đồng cấu và đẳng cấu.

Ví dụ : A = (một dãy số hữu hạn từ 1 đến k) thì đẳng cấu với tập

B = (1, 2, 3, ..., k) vì chúng tương ứng một–một.

Còn cái TV có chức năng hình và chức năng tiếng nó đồng cấu với một hệ thống có chức năng tiếng và có nhiều màn hình. Qua đó chúng ta nhận thấy nếu hai hệ thống đẳng cấu với nhau thì không rút gọn hoặc thay thế được chức năng của chúng, nhưng nếu hai hệ thống chức năng là đồng cấu với nhau thì có thể rút gọn hoặc thay thế các chức năng mà vẫn đảm bảo khả năng hoạt động của hệ thống. Tính chất này sẽ được sử dụng trong việc rút gọn chức năng của ôtômat có nhớ sau này.

Một hệ thống mà không có tính chất đẳng cấu cũng không có tính chất đồng cấu với một hệ thống khác, tức là nó có những chức năng riêng biệt, thì nó là một hệ thống độc lập, loại hệ thống này ta chỉ có thể xây dựng riêng hoặc lấp lấy riêng chức năng cho nó chứ không thể tìm mua hay mượn về được.

1.2.7. CẤU TRÚC ĐẠI SỐ

Chúng ta đều biết đại số có ảnh hưởng rất lớn đến cấu trúc của hệ thống kỹ thuật. Như từ khi xuất hiện đại số lôgic – đại số Boole hai trị (0,1), thì chính nó là cơ sở là nguồn gốc để xây dựng nên các hệ thống số, hệ thống tự động điều khiển hiện đại, hay các rôbôt chức năng... Vậy hệ thống hiện đại số có ảnh hưởng lớn tới các quan hệ kỹ thuật. Nhất là khi tồn tại quan hệ phương trình toán là mạch điện và mạch điện là phương trình toán của lý thuyết ôtômat. Vậy nếu hiểu được chính xác bản chất của cấu trúc đại số, sẽ cho phép ta hiểu được bản chất kỹ thuật của các hệ thống số hiện nay, đồng thời hiểu được khả năng chuyển đổi giữa các loại cấu trúc đại số. Đó là mục đích nghiên cứu ở phần này. Trước hết một cấu trúc đại số được xác định như sau :

Định nghĩa 1.2.9. Cho $N = (n_1, n_2, ..., n_p)$ là một tập hữu hạn các số nguyên không âm và A là một tập nào đó bất kỳ. Nếu ứng với mỗi một phần tử nào đó $n_1 \in N$ ta gán cho một phép toán a_j được định nghĩa trên A ($a_j \in A$), thì chúng ta có một cấu trúc đại số $<A, N>$ – (tập các phép toán, tập các con số).

Như vậy một cấu trúc đại số là một tập có hai phần tử gồm A là một tập bất kì trong đó gồm các phép toán được gán vào, còn N là tập các con số. Với tập N con số ta có thể hiểu được, còn phần tập A khái niệm gán các phép toán có thể được hiểu như sau. Chúng ta đều biết một cấu trúc đại số đơn giản và quen thuộc đó là cấu trúc :

$$<A, N> = < +, -, X, \div ; 0, 1, 2, 3, ..., n>$$

$A = <+, -, X, \div>$ là tập các phép toán

$N = (0, 1, 2, 3, 4, ..., n)$ tập các con số.

Cấu trúc đại số đó là cấu trúc của đại số cổ điển, tập A là tập các phép toán cơ bản gồm (cộng, trừ, nhân và chia) còn N là tập các con số tự nhiên.

Nhìn vào tập các phép toán A ta thấy phép toán được gán rõ nhất là phép toán nhân (x), vì phép toán này hoàn toàn có thể thay thế bằng phép toán cộng (+), bởi vì kết quả của phép toán nhân ta có thể nhận được bằng cách cộng nhiều lần, từ đó ta có thể bỏ phép toán nhân mà thay vào đó là quan hệ cộng nhiều lần, vậy cỡ thể nói phép toán nhân là phép toán được "gán" thêm vào trong tập các phép toán A mà thôi. Đồng thời từ phép toán nhân ta còn có được các phép toán khác như : khai căn, lũy thừa, lôgarit... mà chúng ta quen biết việc tìm ra kết quả của chúng cũng chỉ cần "cộng nhiều lần" là xong.

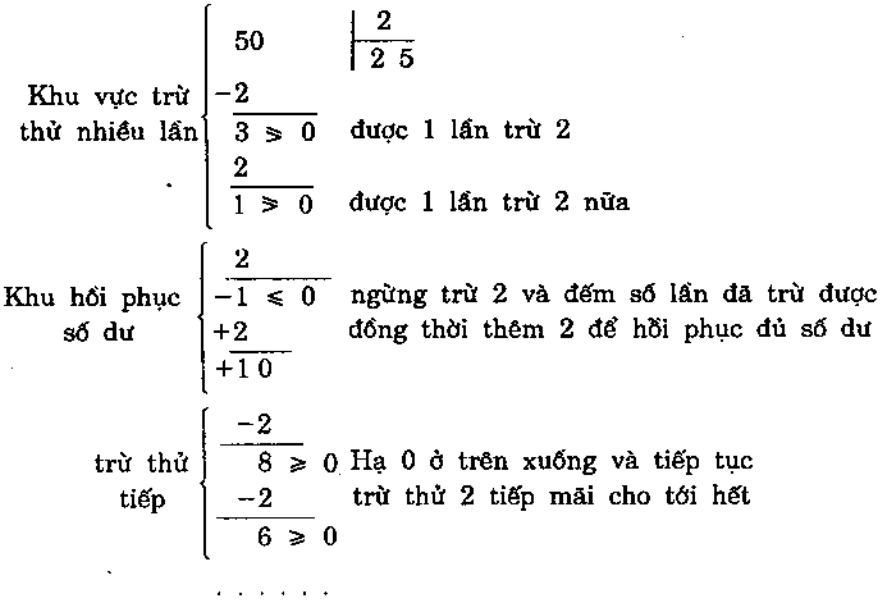
Phép toán chia (÷) cũng là một phép toán được "gán" vào trong tập A, điều đó có thể dễ dàng hiểu được nhờ vào ví dụ sau :

Ví dụ : Tính 50 : 2 = 25

Ta có thể thu được kết quả đó bằng cách làm sau :

Trước hết ta lấy 5 – 2 = 3 kết quả là ≥ 0 ta bảo được 1 lần trừ 2.

Sau đó lấy 3 – 2 = 1 kết quả cho dấu ≥ 0 ta bảo được 1 lần 2 nữa, ở đây ta chỉ cần căn cứ vào dấu ≥ 0 mà kết luận, và cứ trừ mãi cho tới khi xuất hiện dấu ≤ 0.



. . . . . .

Tiếp theo ta lấy 1 − 2 = −1 cho ra kết quả ⩽ 0 thì ngừng trừ 2 và đếm số lần đã trừ được trước đó, rồi ghi vào kết quả (cụ thể là 2 lần trừ). Đồng thời cộng kết quả trừ trước đó với 2 để hồi phục lại số dư trước đó −1 + 2 = +1, tiếp theo hạ 0 xuống thành kết quả 10, và lại tiếp tục trừ thử cho tới hết, số lần trừ thử được ghi ở bên cạnh (cụ thể là 5 lần trừ). Cách làm tích chia như vậy là cơ sở để xây dựng phép tích chia ở bộ số học ALU của máy tính điện tử.

Như vậy phép toán chia chính là quá trình trừ thử nhiều lần, kết quả của phép toán chia của 50 cho 2 nhận được kết quả là 25, nói cách khác 50 chia cho 2 nhận được kết quả là 25 lần trừ 2 chứ không phải là "thương số" như chúng ta vẫn gọi. Vậy phép toán chia có thể được thay thế bằng phép toán trừ mà thực hiện nhiều lần. Do đó ta nói phép toán chia là một phép toán được "gán" thêm vào trong tập A các phép toán trong cấu trúc đại số mà thôi. Phép toán chia còn là nguồn gốc gây ra các quan hệ vi phân, tích phân, sin, cos, artg ... và chúng cũng có thể dùng phép toán trừ để thực hiện được.

Lúc này trong tập A chỉ còn lại hai phép toán cơ bản là cộng và trừ (+, −). Ở đây phép toán cộng cũng là một phép toán được "gán" vào trong A, vì phép toán cộng cũng có thể được thay bằng một quan hệ đơn giản hơn đó là "đếm", và cộng chính là kết quả của quá trình "đếm tăng" (đếm thuận), còn phép trừ là kết quả của quá trình "đếm giảm" (đếm nghịch). Ta có thể nói trong tập A các phép toán của một cấu trúc đại số cổ điển các phép toán của nó đều được "gán" vào, và cả một cấu trúc toán học cổ điển có thể thay thế bằng "quan hệ đếm" − bộ đếm, và đó là mạch điện. Từ đó chúng ta có thể hiểu được tại sao máy tính điện tử lại có thể thực hiện và tính toán đưa ra các kết quả của toán học cổ điển, vì máy tính là một "bộ đếm" mà nó tự "đếm" ra cho ra được các kết quả của các quan hệ toán học đó, sau đó cho qua bộ giải mã sẽ được kết quả toán học theo đúng yêu cầu của con người, đồng thời còn làm được nhiều điều hơn cả tính toán, như trò chơi điện tử, quản lý nhân sự, bảng biểu ... như chúng ta đã biết.

Sau khi nghiên cứu tìm hiểu cấu trúc địa số <A, N>, từ A chúng ta đã biết được là các phép toán nằm trong tập A có thể dùng bộ đếm hay mạch điện thực hiện được. Nhưng với N là tập các con số, vậy chúng có thể dùng mạch điện để thể hiện được hay không?. Muốn trả lời được câu hỏi đó ta cần xem xét tiếp tới phần sau nói về hệ thống cơ số đếm.

Xem xét về con số (N) trong cấu trúc đại số nhẽ ra đó là một phần nằm trong cấu trúc đại số, nhưng đây là một vấn đề khá lớn nên có thể xem xét nó trong một mục riêng tiếp sau đây.

§1.3. CÁC HỆ CƠ SỐ ĐẾM

Đối với mỗi một cấu trúc đại số ngoài tập các phép toán (A) của nó thì tập các con số (N) cũng đóng vai trò rất quan trọng, nó ảnh hưởng trực tiếp tới quan hệ tính toán của đại số đó. Trong phần này ta nghiên cứu tổng quan các loại hệ cơ số đếm và ảnh hưởng của nó trong toán học, và từ đó đi tới hệ cơ số đếm nhị phân (0,1) cũng như ảnh hưởng của nó tới đại số nói chung, và tính toán mô tả con số như thế nào ở trong máy tính.

1.3.1. CÁCH BIỂU DIỄN TỔNG QUÁT MỘT CON SỐ

Chúng ta quen biết cách biểu diễn số trong hệ cơ số đếm mười, mỗi một con số của hệ mười (hệ 10) được viết như sau :

$$1\ 8\ 9\ 7 = 1.10^3 + 8.10^2 + 9.10^1 + 7.10^0$$

Trong đó : (1,8,9,7) là các ký hiệu, ký tự hay còn được gọi là chữ số, (bản chất là ký hiệu)

(0,1,2,3) là số tự nhiên tăng dần.

(10) là hệ cơ số đếm.

Như vậy một con số N bất kỳ, ta có thể biểu diễn dạng tổng quát của nó :

$$N = a_n.q^n + a_{n-1}.q^{n-1} + \ldots + a_2.q^2 + a_1.q^1 + a_0.q^0$$

Trong đó : $(a_n, a_{n-1}, \ldots, a_2, a_1, a_0)$ đó là các ký hiệu chữ số.

(0,1,2, ..., n-1,n) là số tự nhiên tăng dần.

(q) là hệ cơ số đếm.

Qua cách biểu diễn một con số N bất kỳ ta nhận được hệ cơ số đếm q (hệ đếm q), đó là hệ đếm bất kỳ dùng cho cơ số đếm bất kỳ nào đó. Đồng thời do cách biểu diễn $(a_n, a_{n-1}, \ldots, a_2, a_1, a_0)$ là các ký hiệu bất kỳ, cho nên ứng với mỗi một hệ cơ số đếm ta có thể gán cho một bộ ký hiệu riêng nào đó (mỗi một hệ cơ số đếm có tương ứng một bộ ký hiệu chữ số kèm theo), làm như vậy là làm đúng bản chất cách biểu diễn con số nói chung, nhưng lại làm cho phức tạp thêm cách biểu diễn hệ thống số liệu. Trong khi đó ta đã có một bộ ký hiệu chữ số quen thuộc là (0, 1, 2, 3, 4 ,5, 6, 7, 8, 9), ta có thể dùng

bộ ký hiệu đó biểu diễn các con số ở các hệ cơ số đếm mới khác nhau, từ đó ta có quy ước là :

Dùng lại bộ ký hiệu chữ số như cũ của cơ đếm 10 nhưng ở nghĩa mới (viết con số ứng với hệ cơ số đếm mới).

Đồng thời ta cũng biết rằng số lượng ký hiệu chữ số của hệ đếm nào thì tương ứng với hệ cơ số đếm đó, tức là : Hệ đếm mười có 10 ký hiệu, hệ đếm tám thì có 8 ký hiệu, và hệ đếm năm thì chỉ có 5 ký hiệu chữ số mà thôi.

Như vậy một con số được viết ở hệ cơ số đếm khác nhau thì chúng có dạng :

$(25374)_9$ Hệ cơ số 9

$(30651)_7$ Hệ cơ số 7

$(83607)_{10}$ Hệ cơ số 10

Do dùng chung bộ ký hiệu đã có của hệ cơ số đếm 10, cho nên khi ta viết một con số ở bộ cơ số đếm khác cần thêm chỉ số cho con số được viết, đó chính là cơ số đếm của con số đó.

1.3.2. CÁC HỆ CƠ SỐ ĐẾM KHÁC NHAU VÀ CÁCH BIỂU DIỄN CON SỐ

Khi dùng một hệ cơ số đếm bất kỳ để biểu diễn con số, bao giờ cũng đi từ cách biểu diễn con số bé nhất rồi tăng dần lên, khi đó các ký hiệu của hệ cơ số đếm lần lượt được viết ra theo quy ước giá trị tăng dần của các ký hiệu đó, nhưng khi viết tới ký hiệu chữ số cuối cùng của hệ cơ số đếm đó, nếu muốn tăng thêm giá trị thì chuyển thành 10 (ghép ký hiệu 1 với ký hiệu 0), rồi ghép tiếp với các ký hiệu khác tương ứng với con số có giá trị tăng dần. Từ đó ta có cách biểu diễn các con số của các hệ cơ đếm khác nhau như trong ví dụ sau :

Lượng	Hệ 10	Hệ 8	Hệ 5	Hệ 3	Hệ 2	Hệ 16
	0	0	0	0	0	0
.	1	1	1	1	1	1
..	2	2	2	2	10	2
...	3	3	3	10	11	3
....	4	4	4	11	100	4
.....	5	5	10	12	101	5
	6	6	11	20	110	6
.	7	7	12	21	111	7
.	8	10	13	22	1000	8
.	9	11	14	100	1001	9
.	10	12	20	101	1010	A
.	11	13	21	102	1011	B
.	12	14	22	110	1100	C
.	13	15	23	111	1101	D
	14	16	24	112	1110	E
	15	17	30	120	1111	F
	16	20	31	121	10000	10
	...	...	...	...	...	...

Với cách làm tương tự ta có thể viết được các con số ở các hệ cơ đếm khác nhau mà chúng có các giá trị tăng dần mãi. Qua cách viết như vậy, việc biểu diễn một con số với cơ số đếm $\leq$ 10 rất dễ dàng. Nhưng khi cần biểu diễn một con số có cơ số đếm > 10, tức là bị thiếu các ký hiệu chữ số nếu chỉ dùng các ký hiệu chữ số của cơ đếm 10, lúc này ta cần phải bổ sung thêm các ký tự (ký hiệu) chữ số mới. Cụ thể khi ta muốn biểu diễn con số ở hệ cơ đếm 16 (hệ cơ số đếm Hecxa decibel) thì phải bổ xung thêm các ký tự ký hiệu chữ số mới là (A, B, C, D, E, F). Như vậy một con số ở cơ đếm 16 được viết ở dạng sau :

$$(2\ B\ 5\ A)_{16} \text{ hay } (1\ F\ F\ F)_H,\ (A\ 000)_{16}$$

Trong các ký hiệu chữ số mà con người sử dụng thì chữ số 0 là ký hiệu quan trọng nhất và nó được tìm ra sau cùng. Thậm chí có dân tộc chưa có con số 0 mà phải mượn ký tự này như Trung Quốc, vì họ đếm theo quy luật : (− nhất, = nhị, ≡ tam, 四 tứ, 五 ngũ, 六 lục, 七 thất, 八 bát, 九 cửu, + thập) và dùng các ký hiệu đó để viết ra các con số như trong các tờ lịch hàng ngày ta vẫn dùng.

Ví dụ : = + nhị thập là 20 hay

≡ + 六 tam thập lục là 36.

Khi xuất hiện ký hiệu chữ số 0 thì cách biểu diễn con số gặp phải trường hợp khó phân biệt sau :

$$000 = 0 \qquad \text{hoặc} \qquad 00000 = 0$$

Một bên là con số, còn một bên là giá trị của con số đó. Để phân biệt giữa giá trị của con số và con số đó trong kỹ thuật đã đưa ra quy ước :

$$0000 = \varnothing$$

Quy ước này được dùng trong hệ cơ đếm 16, và hay dùng nhất trong cách mô tả dung lượng của bộ nhớ của máy tính.

Ví dụ : Dung lượng bộ nhớ từ $\phi\phi\phi$ đến FFF.

Ở đây (1111) = F.

1.3.3. CÁCH TÍNH TOÁN Ở CÁC HỆ CƠ SỐ ĐẾM KHÁC NHAU

Sau khi nắm được cách biểu diễn trình tự các con số ở các cơ số đếm khác nhau, ta có thể dựa vào đó để thực hiện các phép tính của các con số trong cùng một hệ cơ số đếm.

Ví dụ : thực hiện phép toán cộng (+) hai số ở cơ số 8 như sau :

```
   1 1 1  ← cột số nhớ
   2 5 3 6
+  3 7 4 5 ← (thêm 5)
 ------------------
  (6 5 0 3)8
```

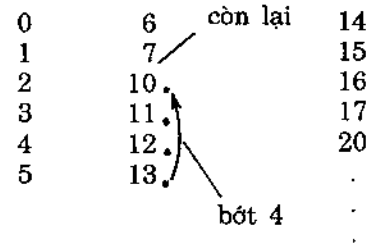
0	6	14
1	7	15
2	10	16
3	11	17
4	12	20
5	13	:

thêm 5

Bắt đầu từ cột đầu tiên bên phải $(6 + 5)_8$, từ 6 ta đếm tăng thêm 5 đơn vị nữa, và nhận được giá trị $(13)_8$, từ đó ta viết kết quả là 3 và nhớ 1 sang cột số bên cạnh giống như ở cơ số 10. Với cách làm tương tự với các cột số còn lại, ta sẽ nhận được toàn bộ kết quả của phép toán cộng giữa hai con số ở cơ số 8.

Phép toán trừ của cơ số 8 được thực hiện qua ví dụ sau :

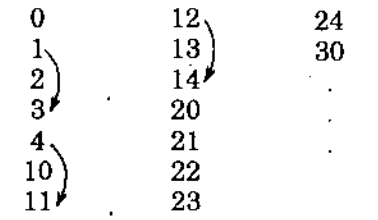
$$\begin{array}{r} 5\ 2\ 4\ 3 \\ -\ 1\ 3\ 6\ 4 \\ \hline (3\ 6\ 5\ 7)_8 \end{array} \leftarrow (\text{bớt } 4)$$



Bắt đầu từ cột đầu tiên bên phải $(3 - 4)_8$, nhưng 3 không trừ được 4 nên ta vay 1 từ cột đứng trước thành $(13)_8$, sau đó ta đếm lùi đi 4 đơn vị sẽ còn dư 7 ghi ở kết quả. Sau đó trả lại 1 đơn vị cho giá trị ở cột đứng trước (như làm phép toán trừ ở cơ 10), và tương tự trừ tiếp ở các cột khác cho tới kết quả cuối cùng.

Cách làm phép toán nhân giữa hai con số ở hệ cơ đếm 5, ví dụ sau :

$$\begin{array}{r} 2\ 1 \leftarrow \text{số nhớ} \\ 2\ 4\ 3 \\ x\ \ 2\ 3 \\ \hline 1\ 3\ 3\ 4 \\ +1\ 0\ 4\ 1 \\ \hline (1\ 2\ 2\ 4\ 4)_5 \end{array}$$

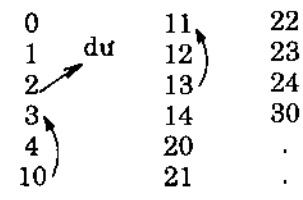


Bắt đầu từ cột đầu tiên $(3 \times 3)_5 = (14)_5$, ta viết 4 và nhớ 1 sang cột bên cạnh, sau đó thực hiện nhân ở cột tiếp là $(3 \times 4)_5 = (2\ 2)_5$ với kết quả nhớ 1 sang ta có kết quả là $(22 + 1)_5 = (23)_5$ từ đó ta viết 3 vào kết quả tính và nhớ 2 sang cột bên cạnh. Cách làm tương tự với các cột còn lại khi nhân 3 lên cho tới cột cuối cùng.

Sau đó nhân 2 lên, kết quả ta viết thụt vào (hay dịch sang trái một cột) cách tính tương tự. Cuối cùng ta cộng kết quả thu được ở cơ số 5, sẽ được đáp số của hai con số khi nhân ở cơ số đếm 5.

Cách làm phép toán chia giữa hai con số ở hệ cơ đếm 5 có thể được thực hiện như sau, ví dụ :

$$\begin{array}{r|l} 1\ 3\ 4,\ 0 & 3 \\ \cdots\ 2\ 4 & \hline \text{dư} \rightarrow \cdots\ 2\ 0 & (2\ 4\ ,\ 3\ \ldots)_5 \\ \cdots\ \ 1 & \end{array}$$



Số đầu tiên là $(13 : 3)_5 = 2$ lần trừ 3 và còn dư 2, ta viết kết quả chia là 2 và số dư là 2, sau đó hạ xuống thành $(24 : 3)_5 = 4$ lần trừ 3 còn dư 2, ta viết 4 vào kết quả

chia và dư 2, sau đó hạ dấu phẩy xuống và tính tiếp cho tới kết quả cuối cùng.

Đối với hệ cơ số 2 và cơ số 16 ta cũng có thể tính toán được :

$$\begin{array}{r} 5\ B\ 6\ 2\ C \\ +\ 2\ 5\ A\ 4\ 2 \\ \hline (8\ 1\ 0\ 6\ E)_{16} \end{array} \qquad \begin{array}{r} 1\ 0\ 1\ 1 \\ +\ 1\ 1\ 1\ 1 \\ \hline (1\ 1\ 0\ 1\ 0)_2 \end{array}$$

Hàng già ⟵ Hàng trẻ ⟶

1 9 8 7

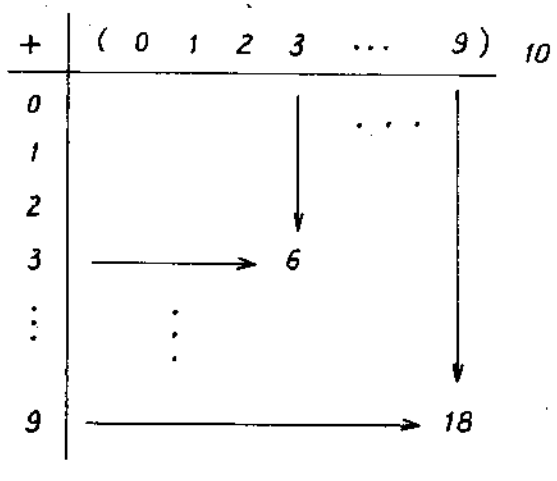
Phép toán nhân hai số nhị phân từ hàng già (từ bên phải) và từ hàng trẻ (từ bên trái):

$$\begin{array}{r} 1\ 0\ 1\ 1 \\ \times\ 1\ 0\ 1 \\ \hline 1\ 0\ 1\ 1 \\ +\ 0\ 0\ 0\ 0 \\ 1\ 0\ 1\ 1 \quad \leftarrow \text{dịch trái} \\ \hline (1\ 1\ 0\ 1\ 1\ 1)_2 \end{array} \qquad \begin{array}{r} 1\ 0\ 1\ 1 \\ \times\ 1\ 0\ 1 \\ \hline 1\ 0\ 1\ 1 \\ +\ 0\ 0\ 0\ 0 \\ \text{dịch phải} \rightarrow\ 1\ 0\ 1\ 1 \\ \hline (1\ 1\ 0\ 1\ 1\ 1)_2 \end{array}$$

Từ các ví dụ đó ta nhận thấy việc tính toán số nhị phân ra các kết quả là dễ dàng và đơn giản. Như ở phép nhân của số nhị phân ta chỉ việc dịch trái số bị nhân (1011) sang trái, (nếu nhân bắt đầu từ hàng trẻ), rồi cộng lại hoặc dịch số bị nhân đó sang phải, (nếu nhân từ hàng già), rồi cộng lại và kết quả của hai cách làm đó cho kết quả là như nhau. Bằng cách dịch số bị nhân sang phải hoặc sang trái rồi cộng kết quả lại sẽ nhận được đáp số, đó cũng chính là cơ sở kỹ thuật để thực hiện phép nhân trong bộ số học ALU của máy vi tính.

Với các cách làm tính như đã nói ở phần trên dùng phương pháp đó tới nay ta đã có thể thực hiện được bốn phép toán (+ cộng, − trừ, . nhân và ÷ chia) đối với bất kỳ con số nào dù chúng được viết ở bất kỳ cơ đếm nào. Đồng thời chúng ta cũng hiểu được rằng bảng cửu chương chỉ giúp ta tính toán ở cơ số đếm 10 thôi, còn đối với các cơ số đếm khác thì ta không thể sử dụng được nữa. Nhưng ở các cơ đếm bất kỳ khác việc tính toán thực hiện rất chậm và không thể nhanh được, điều này có thể giải thích như sau :

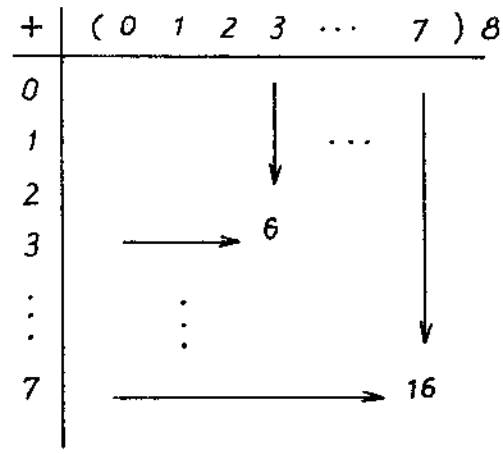
Ở cơ số 10 ta có bảng quan hệ hai ngôi trên một tập của mười ký hiệu chữ số (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) bảng đó ta thường gọi là "bảng cửu chương" ứng với bốn phép toán cơ bản, ví dụ ta có bảng "cửu chương cộng" (phép toán cộng) như sau (hình 1.15).



Hình 1.15. Bảng "cửu chương cộng".

Tương tự bảng cửu chương cộng ta còn có các bảng cửu chương khác là : bảng "cửu chương trừ" bảng "cửu chương nhân" và bảng "cửu chương chia".

Còn ở cơ đếm 8 ta cũng có bảng quan hệ hai ngôi của tám ký hiệu chữ số là (0, 1, 2, 3, 4, 5, 6, 7), bảng này ta gọi tên là "bảng thất chương". Ví dụ ta có bảng "thất chương cộng" (phép toán cộng) ở cơ số 8 trên hình 1.16.



Hình 1.16. Bảng "thất chương cộng".

Từ đó tương tự ta cũng có "thất chương trừ", "thất chương nhân" và "thất chương chia", các bảng thất chương này ta hoàn toàn có thể tính toán ra được khi đã biết cách làm bốn phép tính ở các hệ cơ đếm khác nhau đã được nói tới ở các phần trên.

Tương tự ở cơ đếm 5 ta có năm ký hiệu (0, 1, 2, 3, 4), cũng có thể xây dựng được bảng "tứ chương nhân" (như phép toán nhân ở cơ số 5 đã tính ở phần trên) biểu diễn trên hình 1.17.

•	(0	1	2	3	4)$_5$
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	11	13
3	0	3	11	14	22
4	0	4	13	22	31

Hình 1.17. Bảng "tứ chương nhân".

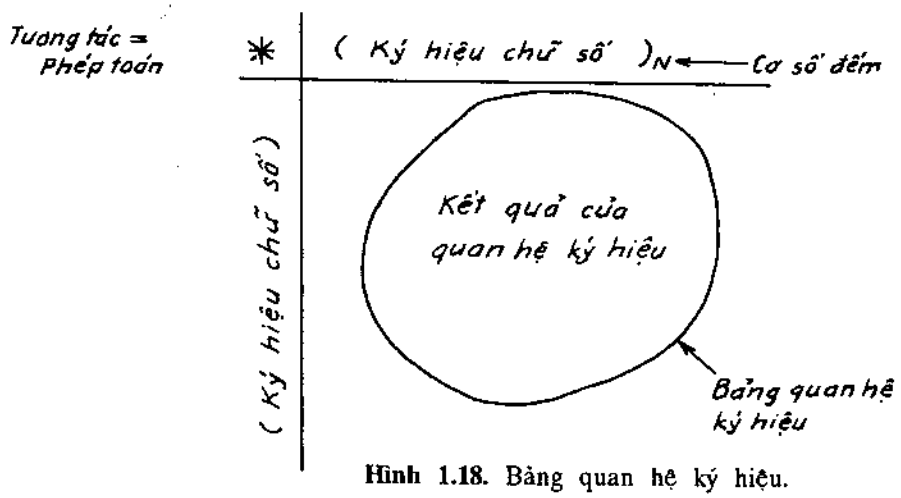
Từ đó ta cũng có được bảng "tứ chương cộng", "tứ chương trừ" và "tứ chương chia" của hệ cơ đếm 5.

Như vậy đối với mỗi hệ cơ số đếm bất kỳ ta đều có bốn bảng "X chương" tương ứng với bốn phép toán của hệ cơ số đếm đó. Với số lượng lớn các bảng như vậy con người không thể thuộc nổi giống như thuộc bảng "cửu chương" của hệ cơ đếm 10 để làm tính.

Từ đó ta đưa ra một bảng "X chương" tổng quát cho mọi hệ cơ số đếm khác nhau, bảng đó có tên là "bảng quan hệ ký hiệu" được mô tả trên hình 1.18.

Trong đó : N là hệ cơ số đếm, nó có ảnh hưởng và quyết định tới kết quả ở bên trong của bảng quan hệ ký hiệu. Từ đó ta có thể gọi nó dưới một cái tên khác là : "N là điều kiện thực hiện quan hệ ký hiệu".

"Bảng quan hệ ký hiệu" cũng chính là tương tác hai ngôi trên một tập các ký hiệu chữ số của phép toán trong hệ cơ đếm đó. Từ bảng quan hệ ký hiệu ta hiểu được rằng ở các hệ cơ số đếm bất kỳ khác cơ đếm 10 ta tính khó và chậm là vì chưa thuộc lòng được bảng quan hệ ký hiệu của hệ cơ đếm đó. Nếu ta thuộc lòng được bảng quan hệ ký hiệu của một cơ đếm bất kỳ như thuộc lòng bảng cửu chương, thì việc tính toán ở hệ cơ đếm mới đó cũng dễ dàng và nhanh chóng.



Hình 1.18. Bảng quan hệ ký hiệu.

Qua đó ta cũng hiểu muốn tính toán nhanh phải thuộc lòng bảng "quan hệ ký hiệu". Đồng thời kết quả đó còn phụ thuộc vào một tham số nữa đó là cơ số đếm N hay còn gọi là điều kiện thực hiện quan hệ ký hiệu. Điều đó được minh họa ở ví dụ sau :

$$\begin{array}{r} 2 \\ +\ 5 \\ \hline (7)_{10} \end{array} = \begin{array}{r} 2 \\ +\ 5 \\ \hline (10)_7 \end{array} = \begin{array}{r} 2 \\ +\ 5 \\ \hline (11)_6 \end{array} = \begin{array}{r} 10 \\ +\ 101 \\ \hline (111)_2 \end{array} = \begin{array}{r} \bullet\bullet \\ +\ \bullet\bullet\bullet\bullet\bullet \\ \hline (\bullet\bullet\bullet\bullet \\ \bullet\bullet\bullet) \end{array} = \text{lượng}$$

Qua ví dụ, ta nhận thấy cùng một tương tác cộng (phép toán cộng), cùng một quan hệ ký hiệu giữa 2 và 5 (2 ; 5) nhưng điều kiện thực hiện quan hệ ký hiệu (N) khác nhau (cơ số đếm khác nhau), thì kết quả ký hiệu của quan hệ ký hiệu sẽ khác nhau. Nhưng dù kết quả đó viết ở dưới dạng của bất kỳ cơ đếm nào thì nó vẫn có chung một giá trị (là kết quả chính xác nhất) đó là "lượng".

Như vậy bản chất của việc tính toán là tìm ra "lượng" nào đó từ kết quả tính toán mà thôi, ví dụ như lượng tiền, lượng vật, lượng người.

Từ đó nẩy ra vấn đề là : làm sao có cách nào đó để tìm ra "lượng" nhanh nhất và dễ nhất, và có cách nào khác thay thế cho bộ óc của con người khi cần tìm lượng không ?

Để tính toán tìm được "lượng" nhanh chóng và dễ dàng ta có thể dùng bảng quan hệ ký hiệu với số lượng ký hiệu chữ số ít nhất, và điều kiện thực hiện quan hệ ký hiệu là đơn giản nhất. Bảng quan hệ ký hiệu đó chính là "bảng nhất chương" của hệ cơ đếm 2, và nó được minh họa trên hình 1.19.

+	$(0\ \ 1)_2$
0	0 1
1	1 10

Bảng nhất chương cộng

•	$(0\ \ 1)_2$
0	0 0
1	0 1

Bảng nhất chương nhân

Hình 1.19. Bảng "nhất chương" của cơ đếm 2.

Trong bảng "nhất chương" của hệ cơ số đếm nhị phân ta thấy có số lượng ký hiệu chữ số khác nhau là ít nhất, chỉ có 0 và 1, đồng thời với điều kiện thực hiện quan hệ ký hiệu là đơn giản nhất, đó là 2 (N = 2), vì nếu N = 1 thì đó là điều kiện tiên quyết và khi đó không còn gì để tính toán được nữa.

"Bảng nhất chương" rất dễ nhớ và dễ thuộc lòng (không khó nhớ khó thuộc như bảng cửu chương) nhưng khả năng tính toán để tìm ra lượng của nó cũng tương đương như "bảng cửu chương", dựa vào bảng nhất chương ta có thể dùng tư duy tính toán tìm lượng như mọi bảng "X chương" khác, nhưng đồng thời bảng nhất chương còn có thể dùng mạch điện để thực hiện được, nói cách khác việc tìm ra kết quả toán học (tìm ra lượng) ta có thể còn dùng mạch điện để đưa ra, tức là có thể dùng mạch điện để thay thế việc tính toán của con người.

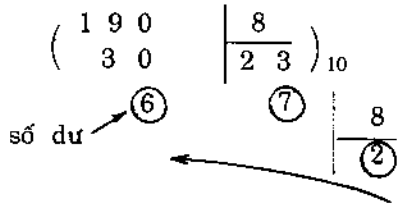
Điều dễ dàng nhận ra bảng "nhất chương nhân" đó chính là mạch VÀ, còn bảng "nhất chương cộng" đó chính là mạch trigơ (mạch Flíp Flốp) mà sau này ta sẽ nói đến. Dùng mạch điện thay thế cho tư duy tính toán của con người đó chính là ưu thế tuyệt đối của hệ cơ số đếm 2 hơn hẳn các hệ cơ đếm khác, nó đã cho phép chúng ta chế ra được các máy tính, các hệ thống tính toán to lớn và mạnh mẽ thay thế cho việc dùng tư duy để tính toán của con người từ xưa tới nay, máy tính ngày nay được coi như đòn bẩy của trí tuệ con người. Đồng thời cũng từ cơ đếm 2, cho phép con người thực hiện được bao nhiêu chức năng kỹ thuật khác bằng các hệ thống kỹ thuật số mà ở các cơ đếm khác không thể có được.

1.3.4. CHUYỂN ĐỔI GIỮA CÁC HỆ CƠ SỐ ĐẾM

Qua cách biểu diễn con số ở các hệ cơ đếm khác nhau ta nhận thấy là cùng một giá trị (cùng một lượng) có thể được biểu diễn bằng các con số khác nhau được viết ở các cơ số đếm khác nhau. Từ đó nảy ra vấn đề là cần có sự chuyển đổi qua lại giữa các con số của các cơ số đếm khác nhau với nhau. Quá trình chuyển đổi con số của các cơ số đếm khác nhau với nhau đó chính là cơ sở để xây dựng "bộ chuyển đổi mã", bộ giải mã mà hay được nhắc đến trong các hệ thống số, cụ thể đó chính là bộ giải mã từ cơ số 2 sang cơ số 10 và ngược lại. Khi có bộ giải mã cho phép giữa người và máy (hệ thống số) hiểu được nhau và trao đổi thông tin qua lại được với nhau.

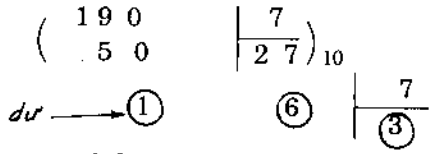
Cách chuyển đổi tổng quát giữa các cơ số đếm khác nhau được thực hiện như sau :

Ví dụ : Cho trước một con số ở hệ cơ đếm 10 là $(190)_{10}$ ta cần chuyển thành một con số có giá trị tương đương ở cơ đếm 8 $(X\ X\ X)_8$, cách làm như sau : Ta chia số cho trước ở hệ cơ số 10 cho 8 tính toán ở cơ số 10, chia mãi để tìm ra số dư



$$(1\ 9\ 0)_{10} = (2\ 7\ 6)_8$$

Nếu muốn chuyển số $(1\ 9\ 0)_{10}$ thành một số tương ứng ở cơ đếm 7, với cách làm tương tự chia số đó cho 7 tính toán ở cơ số 10 để tìm ra các số dư. Ví dụ :



Tất nhiên ta có thể viết kết quả chung là

$$(1\ 9\ 0)_{10} = (2\ 7\ 6)_8 = (3\ 6\ 1)_7$$

Để kiểm tra sự chính xác của các kết quả của sự chuyển đổi ta có thể dựa vào "lượng" (giá trị chính xác) của chúng :

$$\text{Lượng} = \left(\begin{array}{c}\text{cách biểu diễn nguyên thủy}\\ \text{của con số ở cơ số đếm của nó}\end{array}\right)$$

Như vậy :

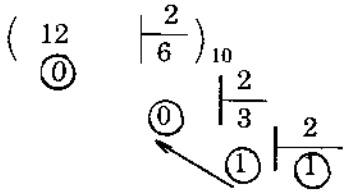
$$\left(190\right)_{10} = 1 . 10^2 + 9 . 10^1 + 0 . 10^0 = 190 \text{ lượng}$$

$$\left(276\right)_{8} = 2 . 8^2 + 7 . 8^1 + 6 . 8^0 = 190 \text{ lượng}$$

$$\left(361\right)_{7} = 3 . 7^2 + 6 . 7^1 + 1 . 7^0 = 190 \text{ lượng}$$

Với cách làm như vậy ta có thể chuyển đổi giữa các con số cơ số 10 sang cơ số 2 và ngược lại để người và máy hiểu nhau. Ví dụ :

- Chuyển một con số của cơ số 10 - $(12)_{10}$ sang cơ số 2 :



Kết quả là : $(12)_{10} = (1100)_2$

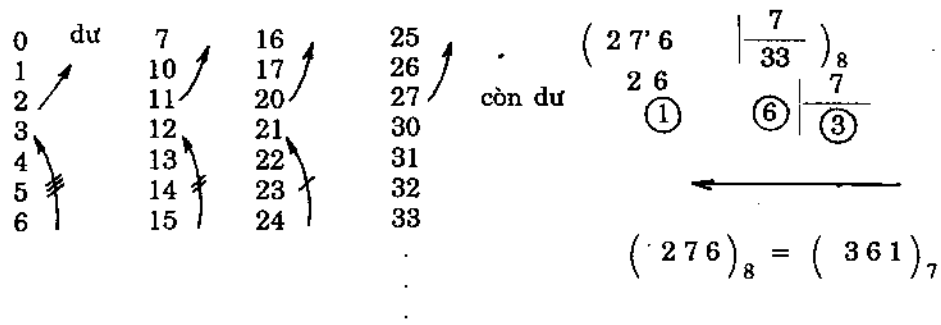
- Chuyển một con số ở cơ số 2 - $(1100)_2$ sang cơ số 10 là :

$$(1100)_2 = 1 . 2^3 + 1 . 2^2 + 0 . 2^1 + 0 . 2^0 = 12 \text{ lượng} = (12)_{10}$$

Ở đây : $2^3 = 8$; $2^2 = 4$; $2^1 = 2$, $2^0 = 1$ còn được gọi là "trọng số" của cơ đếm 2, và cơ đếm hai còn được gọi là có trọng số (8, 4, 2, 1).

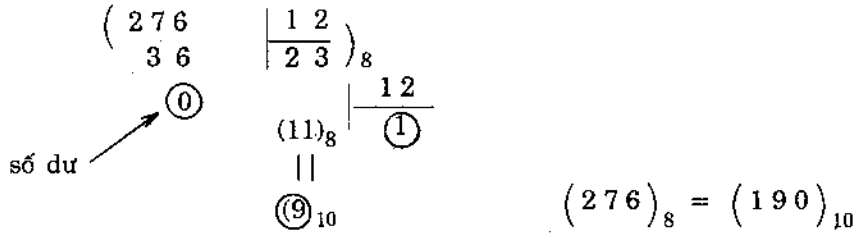
Nay ta phải chuyển đổi một con số của cơ đếm 8 là $(2\ 7\ 6)_8$ thành một con số tương đương nào đó ở cơ đếm 7, cách làm như trên, là thực hiện chia số đó cho 7 tính toán trong cơ số đếm 8 để tìm ra số dư. Ví dụ :

Lần đầu lấy con số $(2\ 7)_8$ sau đó đếm lùi đi 7 (là số chia) sẽ được 3 lần đếm lùi và còn dư là 2. Sau đó hạ 6 xuống thành $(2\ 6)_8$ lại đếm lùi (trừ) đi 7 kèm số dư của nó. Với cách làm tương tự sẽ cho ta kết quả các số dư khi chia 7 tính ở cơ số 8.



Phần trên ta đã thực hiện chuyển đổi giữa các con số của cơ số đếm khác nhau từ cơ số đếm to sang cơ số đếm nhỏ. Nhưng nếu ta muốn chuyển một con số từ cơ đếm nhỏ sang cơ đếm lớn hơn cách làm như sau :

Ví dụ : Chuyển đổi con số $(2\ 7\ 6)_8$ thành một con số tương ứng ở cơ số 10, lúc này ta phải chuyển cơ số 10 thành $(12)_8$ tức là $(10)_{10} = (12)_8$. Sau đó ta chia số phải chuyển đổi với số (12) tính ở cơ số 8. Tổng quát nếu muốn chuyển đổi con số từ cơ số nhỏ sang cơ số lớn hơn thì ta phải đổi cơ số của số lớn hơn (cơ số lớn hơn) thành một số nào đó tương ứng của cơ số nhỏ hơn. Sau đó chia số phải chuyển đổi với con số đó và tính ở cơ số bé có số phải chuyển đổi cụ thể là :



Tới đây ta đã biết được phương pháp tổng quát để chuyển đổi qua lại giữa các con số khi biểu diễn chúng ở hệ cơ số đếm nào đó khác nhau. Nhưng sau này trong kỹ thuật thường chỉ còn quan tâm tới việc chuyển đổi giữa các con số cơ số đếm 10 thành các con số của cơ số đếm 2, và ngược lại là chuyển con số ở cơ đếm 2 của máy thành con số ở cơ đếm 10 – mạch giải mã. Đó cũng là các bộ phận (mạch điện) cơ bản của hệ thống số cũng như của máy tính, chúng sẽ được xây dựng cụ thể ở các phần sau, đó là bộ tạo mã (bàn phím của máy) và bộ giải mã (chỉ thị kết quả) rất quen thuộc với chúng ta.

1.3.5. CÁC CÁCH BIỂU DIỄN CON SỐ KIỂU KHÁC

Ngoài các cách biểu diễn số thông thường, ta còn có các cách biểu diễn con số kiểu khác, mà hiện nay các cách biểu diễn đó vẫn đang được dùng phổ biến trong cuộc sống hàng ngày của chúng ta.

Trước tiên đó là hệ thống đếm và cách biểu diễn số theo "con giáp" rất quen thuộc trong mỗi một tờ lịch ở nhà chúng ta.

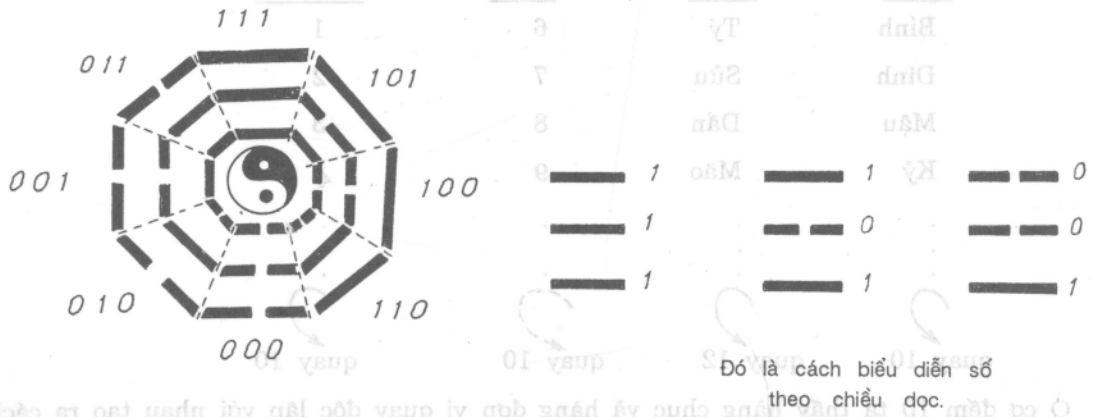
Can	Chi	Chục	Đơn vị
Bính	Tý	6	1
Đinh	Sửu	7	2
Mậu	Dần	8	3
Kỷ	Mão	9	4
.	.	.	.
.	.	.	.
.	.	.	.
quay 10	quay 12	quay 10	quay 10

Ở cơ đếm 10 ta thấy hàng chục và hàng đơn vị quay độc lập với nhau tạo ra cách viết con số. Nhưng ở hệ đếm kiểu "con giáp" thì cột can quay 10 còn cột chi quay 12 và cả hai cột cùng quay tạo ra cách biểu diễn con số của cách biểu diễn số kiểu con giáp.

Ta thấy cách biểu diễn số theo "con giáp" phức tạp hơn và tính toán khó hơn, nhưng thông tin nhận được từ cách biểu diễn như vậy cũng nhiều hơn cách biểu diễn số thông thường cơ số đếm 10. Vì cách biểu diễn số theo con giáp cũng chỉ được "lượng" như ở cơ đếm 10, khi ta hỏi nhau sinh tuổi con giáp nào, và mỗi tờ lịch ta bóc đi cũng chính là con số đếm và luôn có sự khớp nhau về lượng giữa cách biểu diễn số theo con giáp cũng như cách biểu diễn số thông thường. Rồi ở con giáp con số đếm "tý" vừa là 1 cũng vừa là tý, còn ở cơ số đếm 10 thì 1 chỉ là 1 mà thôi. Như vậy chỉ riêng cách biểu diễn ký hiệu chữ số thì đã cho ta thông tin.

Dựa vào cơ số 10 ta tìm được lượng và không có thông tin nào khác nữa, nhưng con số biểu diễn theo con giáp thì ngoài việc cho ta tìm ra lượng (tính tuổi) nó còn kèm theo cho thêm một thông tin khác, đó chính là quy luật của tự nhiên, là điều mà ở cơ đếm 10 không cho được. Mà khi hiểu được quy luật của tự thì con người có thể sống hòa đồng hợp với quy luật đó. Chính vì thế ngoài cơ đếm 10, ở nước ta (vùng văn hóa phương Đông) vẫn tồn tại cách biểu diễn các con số theo con giáp không có ở phương Tây. Cách biểu diễn số kiểu "con giáp" này là đặc trưng cho nền văn hóa của phương Đông nói chung, và đó chính là một cách biểu diễn con số kiểu khác mà con người đang dùng hiện nay.

Một cách biểu diễn số kiểu khác nữa đó là dùng các nét vạch. Ở phương Đông cũng như ở nước ta từ lâu rồi các cụ đã dùng các nét vạch bao gồm vạch liền " ⎯⎯ " hào dương vạch đứt "⎯ ⎯" hào âm, chúng được dùng biểu diễn quẻ trong Kinh dịch của cụ Ngô Tất Tố. Cụ thể hơn cách biểu diễn số bằng các vạch viết theo hàng dọc được thấy rõ trong gương "bát quát" mà nhân dân hay dùng trong đời thường, chúng ta sẽ rất dễ dàng hiểu cách biểu diễn số kiểu vạch khi cho quy ước lôgic vào : Nếu ta quy ước nét vạch liền " ⎯⎯ " là lôgic 1, còn nét vạch đứt " ⎯ ⎯ " là lôgic 0 thì "bát quái" cũng chỉ là một cách biểu diễn số nhị phân mà thôi, điều đó được mô tả trên hình 1.20.



Hình 1.20. Con số ò bát quái.

Cách biểu diễn số theo chiều ngang sẽ là

$$(\text{—} \; \text{= =} \; \text{—}) = (1\ 0\ 1)_2$$

Ngoài ra chúng ta còn quen thuộc với các cơ số đếm khác nhau của nền văn hóa phương Đông như : Ngũ hành (kim, mộc, thủy, hỏa, thổ), ngũ vị, là cơ đếm 5 ; bốn mùa là cơ đếm 4 ; ba tháng một mùa là cơ đếm 3 ; bẩy nốt nhạc là cơ đếm 7 ; lục phủ ngũ tạng là cơ đếm 6 và 5 ; mười hai tháng ứng với một năm là cơ đếm 12... Qua đó ta thấy các cách biểu diễn con số mà con người vẫn dùng hiện nay là rất phong phú, vì đó là cơ sở để con người tìm hiểu và nghiên cứu tự nhiên để phục vụ cho lợi ích của mình.

Ngoài ra ta còn có cách biểu diễn con số theo kiểu La Mã nữa – Chữ số La Mã, ví dụ như V, X, C... Đây là cách biểu diễn con số của nền văn minh thời cổ đại La Mã. Nguyên lý biểu diễn và cách tính toán của loại con số kiểu này hiện nay chúng ta đều chưa biết tới, mặc dù vẫn dùng nó. Bởi vì đây là một nền văn minh cổ đại đã bị huỷ diệt hoàn toàn từ lâu, mà cho tới nay con người vẫn chưa tìm hiểu lại được nó. Như vậy chúng ta đã nghiên cứu khá kỹ về con số, về hệ cơ số đếm và các cách biểu diễn số kiểu khác mà con người vẫn dùng, con số là một phần không thể thiếu được của một cấu trúc đại số.

Sau khi nghiên cứu kỹ về bản chất của tập các phép toán A và tập các con số N của cấu trúc đại số, ta thấy là tập A cũng có hể dùng mạch điện thay thế (bộ đếm), và tập con số N khi chuyển qua nhị phân cũng có thể thực hiện nhờ mạch điện. Như vậy một cấu trúc đại số tổng quát <A, N> hoàn toàn có thể dùng mạch điện để mô tả, minh họa và lắp ráp thay thế được. Điều đó đã chứng minh một cách rõ ràng khả năng xây dựng nên các mạch điện, các máy tính, nó cho phép thay thế việc dùng tư duy, để tính toán của con người trước đây bằng các máy tính điện tử như chúng ta có hiện nay.

§1.4. DÀN VÀ CÁC HỆ THỐNG ĐẠI SỐ

Trong phần này sẽ nói tới một khái niệm cơ bản của đại số hiện đại, đó là khái niệm về dàn. Dàn không chỉ quan trọng trong toán học vì từ khái niệm của nó sẽ tạo

ra các hệ thống đại số, mà quan trọng nhất là đại số lôgic – đại số Boole, mà dàn còn là một công cụ quan trọng dùng để gia công, biến đổi, phân giải ôtômát có nhớ đồng thời dựa vào Dàn ta có thể thực hiện các hệ số cũng như các hệ thống chức năng mong muốn.

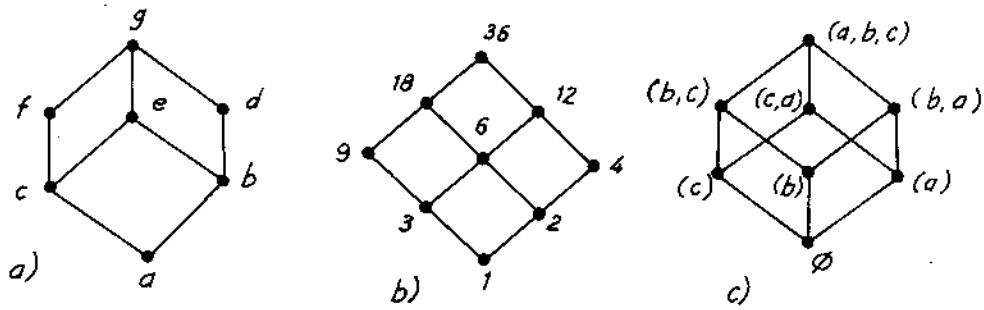
Dựa vào Dàn cho phép ta tìm được toàn bộ các dạng của các tập có cùng một thuộc tính từ một tập cơ bản ban đầu, đó là điều rất cần trong việc nghiên cứu khoa học, vì khi ta tìm được mọi tập có cùng một tính chất, thì điều đó cho phép ta đưa ra các nhận xét hay kết luận chính xác. Dựa vào Dàn ta có thể tối thiểu hóa số lượng trạng thái của ôtômat tổng hợp ôtômat có nhớ ở phần sau.

Dàn cho phép ta tổ chức và biết được các trạng thái có cùng một thuộc tính của ôtômat có nhớ, từ đó tạo ra được các tập tự phụ thuộc rất cần cho quá trình mã hóa của ôtômat có nhớ, điều đó cho phép ta thiết kế lắp ráp hệ thống số một cách mạch lạc gọn gàng, đồng thời điều đó thuận lợi cho quá trình tìm kiếm bằng phần mềm sau này. Như vậy Dàn có liên quan đến thiết kế mạch (phần cứng) cũng như trong tìm kiếm (phần mềm) trong hệ thống kỹ thuật số. Có thể nói Dàn là một khái niệm là một nền tảng quan trọng cho lĩnh vực kỹ thuật số dựa vào, và nếu không có dàn trong toán học thì cũng không có lĩnh vực kỹ thuật số trong kỹ thuật. Chính vì thế mà ta nghiên cứu về dàn trong phần này. Dàn thực chất là một khái niệm lỏng lẻo được hiểu theo định nghĩa sau.

1.4.1. DÀN ĐỊNH NGHĨA VÀ TÍNH CHẤT

Định nghĩa 1.4.1. Dàn là một tập sắp xếp riêng D mà trong đó hai phần tử bất kỳ x và y đều có hai phép toán giao và hợp.

Để hiểu cụ thể hơn nữa về dàn ta có thể thấy được trên hình 1.21.



Hình 1.21. Cách mô tả dàn.

Đó là các dàn mà các phần tử của nó có các quan hệ toán học thỏa mãn định nghĩa trên, ví dụ như :

Trên hình 1.21,a ta có : $e = c + b$

$g = e + d$

$b = e.d$

$a = c.b$

Hay trên hình 1.21,c ta thấy :

phần tử (b,c) = (c) + (b) và (a,c) = (a) + (c)

(b) = (a,b) . (b,c)

(a).(b) = ϕ (c) = (b,c) . (a,c) và ...

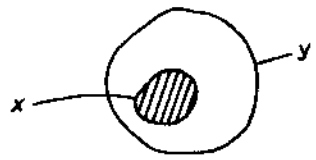
Như vậy mọi phần tử của chúng đều có thể tương tác với phép toán giao và phép toán hợp. Trên hình 1.21,c ta thấy dàn cho biết mọi khả năng nhóm hợp của tập A giống như P(a) = ((ϕ), (a), (b), (c), (a,b), (a,c), (b,c), (a,b,c)), ngoài ra dàn còn cho ta biết trật tự sắp xếp của chúng, có trên có dưới tạo thành một mạng chết cứng không thay đổi, hoán vị các phần tử được. Do đó dàn cho ta nhiều thông tin hơn để dễ dàng quyết định một điều gì đó, hay chọn lựa một nhiệm vụ nào đó thích hợp. Việc ứng dụng cụ thể của dàn như thế nào trong phần sau quyển sách này sẽ đề cập tới.

Dàn là một tập sắp xếp vì vậy trước khi nói tới tính chất của dàn ta phải nói tới tính chất của trật tự của sắp xếp. Tính chất của sắp xếp có liên quan tới các phép toán giao và hợp, từ các định nghĩa của chúng ta có thể nhận thấy là :Trong một tập sắp xếp riêng A nếu tồn tại hai phép toán giao và hợp (+, .) của hai phần tử x, y∈A thì chúng phải thỏa mãn quan hệ sau :

Nếu $x \subseteq y$ thì kéo theo $x . y = x$

$x \leqslant y$ $x + y = y$

Tính chất đó được minh họa trên hình 1.22.



Vậy $x \subseteq y$ thì $\begin{cases} x . y = x \\ x + y = y \end{cases}$
$x \leqslant y$

Hình 1.22. Quan hệ sắp xếp riêng.

Các tính chất cơ bản của hai phép toán giao và hợp có thể được nêu tóm tắt trong định lý sau.

Định lý 1.4.1. Trong một tập sắp xếp riêng A, nếu tồn tại các phép toán giao và phép toán hợp của hai phần tử thuộc A thì chúng sẽ thỏa mãn các tính chất lũy đẳng, giao hoán, kết hợp và loại trừ (tính chất nuốt) sau :

D_1 : Tính chất luỹ đẳng.

$x . x = x$ $x + x = x$

D_2 : Tính chất giao hoán

$x . y = y . x$ $x + y = y + x$

D_3 : Tính chất kết hợp

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

D_4 : Tính chất loại trừ (nuốt)

$$x . (x + y) = x \qquad x + (x . y) = x$$

Các tính chất trên có thể chứng minh một cách dễ dàng và bằng nhiều cách khác nhau như : Dựa vào định nghĩa, dựa vào đồ thị Viên, hay bằng mạch điện đơn giản.

Định lý 1.4.2. Trong một dàn bất kỳ, tất cả các phần tử của nó đều thỏa mãn :

a) Các tính chất từ D_i đến D_4 ở trên.

b) Tính chất bảo toàn, nghĩa là :

Nếu $x \leqslant y$ thì kéo theo $x . z \leqslant y . z$

$$x + z \leqslant y + z$$

c) Tính chất bất đẳng thức môđun, nghĩa là

Nếu $x \leqslant z$ thì $x + (z . y) \leqslant (x + y) . z$

d) Tính chất bất đẳng thức phân bố, nghĩa là

$$x . (y + z) \geqslant (x . y) + (x . z)$$

$$x + (y . z) \leqslant (x + y) . (x + z)$$

Chứng minh : Ví dụ ta chứng minh tính chất c) như sau :

Ta biết : $x \leqslant z$ và $x \leqslant x + y$ nên có $x \leqslant (x + y) . z$

$y \leqslant x + y$ $\qquad z . y \leqslant (x\ y) . z$

Vậy ta có $(x \leqslant (x + y) . z) + (z.y \leqslant (x + y) . z)$ có kết quả là

$$x + (z . y) \leqslant (x + y) . z$$

Tương tự như vậy có thể chứng minh các tính chất khác. Từ các định nghĩa về dàn ta có thể nhận thấy rằng một dàn hữu hạn luôn có một phần tử bé nhất và một phần tử lớn nhất. Phần tử bé nhất nhận được từ kết quả của phép toán giao tất cả các phần tử của dàn với nhau, còn phần tử lớn nhất sẽ thu được bằng phép toán hợp tất cả các phần tử của dàn với nhau.

1.4.2. CÁC LỚP DÀN ĐẶC BIỆT

Dàn có thể được phân chia ra ba loại lớp dàn đặc biệt là : dàn môđun, dàn phân bố và dàn phân bố bù (dàn Boole). Để phân biệt các lớp dàn ta dựa vào định nghĩa toán học của chúng như sau.

1.4.2.1. Dàn môđun

Định nghĩa 1.4.2. Dàn D là một dàn môđun nếu nó thỏa mãn bất đẳng thức môđun sau :

Nếu $x \leqslant z$ thì kéo theo $x + (y . z) = (x + y) . z$

với mọi $x, y, z \in D$

Bất kỳ một dàn nào thỏa mãn định nghĩa đó đều gọi là dàn môđun, và từ định nghĩa về dàn môđun ta có thể nhận được định lý sau.

Định lý 1.4.3. Dàn D là một dàn môđun nếu và chỉ nếu thỏa mãn quan hệ sau :

Nếu $x \leqslant z$ mà có $x . y = x . y$

$x + y = x + y$ thì kéo theo $x = z$

Trước hết ta chứng minh điều kiện cần : Giả sử D là một dàn môđun, đồng thời thỏa mãn các điều kiện của định lý đã nêu thì nhận được kết quả là $x = z$. Với $x,y,z \in D$ ta có :

$x = x + x . y$ (theo tính chất D_4)

$= x + z . y$ (theo định lý)

$= (x + y) . z$ (theo định nghĩa dàn môđun)

$= (z + y) . z$ (theo định lý)

$= z$ (theo tính chất D_4, tính chất nuốt)

Điều kiện đủ : Ta phải chứng minh rằng nếu thỏa mãn định lý thì kéo theo $x = z$, thì dàn D là dàn môđun.

Ta có $x + (y . z) = x + (x . y)$ (theo định lý)

$= x$ (theo tính chất nuốt)

$= z$ (do thỏa mãn định lý)

$= z . (y + z)$ (tính chất nuốt)

$= z . (x + y)$ (thỏa mãn định lý)

Vậy $x + (y . z) = z . (x + y)$ thỏa mãn định nghĩa dàn môđun.

Một lớp dàn quan trọng khác đó là dàn phân bố. Tính chất phân bố không phải phép toán nào cũng có, vậy lớp dàn phân bố thì các phần tử của nó có các phép toán có tính phân bố, cụ thể ta có thể nhận biết qua định nghĩa của nó.

1.4.1.2. Dàn phân bố

Định nghĩa 1.4.3. Dàn D là một dàn phân bố nếu nó thỏa mãn các phương trình phân bố sau đối với mọi $x,y,z \in D$:

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

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

Các phần tử có tính phân bố đối với cả phép toán nhân và phép toán cộng. Từ định nghĩa đó ta có thể nhận được định lý.

Định lý 1.4.4. Dàn D được gọi là dàn phân bố nếu và chỉ nếu đối với mỗi phần tử $x,y,z \in D$ ta có :

$x . y = z . y$ và $x + y = z + y$ thì kéo theo $x = z$

Chứng minh định lý này rất dễ dàng thực hiện như ở trên, chỉ cần vận dụng thêm các quan hệ của định lý. Từ định lý ta suy ra dàn phân bố là một dàn môđun và ngược lại nếu dàn môđun có tính phân bố thì trở thành dàn phân bố. một lớp dàn đặc của dàn phân bố là dàn phân bố bù hay còn gọi là dàn Boole nó là lớp dàn quan trọng nhất mà ta quan tâm.

Cho một dàn D có phần tử bé nhất là 0 và phần tử lớn nhất là 1. Vậy nếu có phần tử x và y thuộc dàn D thì luôn luôn có quan hệ :

$$x \cdot y = 0 \text{ và } x + y = 1$$

Khi đó chúng ta nói : x là phần bù của y và ngược lại

y có phần tử bù là x

Đó cũng chính là điều kiện để tìm phần bù, và hai phần tử bất kỳ nếu chúng thỏa mãn quan hệ trên thì nói chúng là phần bù của nhau. Áp dụng quan hệ đó ta thấy có kết quả sau :

$$0 \cdot 1 = 0 \text{ và } 0 + 1 = 1$$

Vậy có thể nói : 0 có phần bù là 1 và

1 có phần bù là 0

Từ đó ta có quan hệ phủ định (phép toán phủ định) là :

0 phủ định là 1 ký hiệu $\bar{0} = 1$

1 phủ định là 0 ký hiệu $\bar{1} = 0$

Trong cuộc sống ta thường nói phủ định ý kiến của nhau điều đó có thể hiểu là chúng ta lấy phần bù ý kiến của nhau. Cũng từ đây bắt đầu xuất hiện phép toán phủ định đó là phép toán mới phép toán một ngôi. Phép toán phủ định là một trong những phép toán quan trọng nhất của đại số lôgic–đại số Boole, phép toán này sẽ được thực hiện bằng mạch đảo, mạch phủ định trong kỹ thuật.

Đặc điểm của dàn bù có thể nhận thấy như sau.

1.4.1.3. Dàn phân bố bù

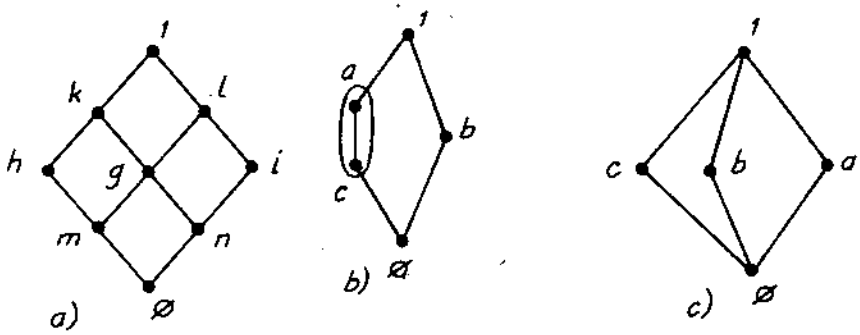
Định nghĩa 1.4.5. Dàn D là một dàn bù nếu như tất cả các phần tử của nó đều có phần tử bù.

Như vậy một cách tổng quát trong một dàn có thể có những phần tử không có phần tử bù, nhưng ngược lại có những phần tử có nhiều phần tử bù. Nhưng đặc điểm đó có thể nhận thấy trên hình sau.

Trong các dàn vẽ ở hình trên ta thấy có những phần tử không có phần tử nào như các phần tử h,g,i,... vì không thỏa mãn điều kiện của phần bù của hình 1.23a. Trong hình 1.23b ta thấy có quan hệ :

$$b \cdot (a, c) = \phi$$

$$b + (a, c) = 1$$



Hình 1.23.

như vậy b có phần tử bù là (a, c) nhưng không có quan hệ :

$$c \,.\, a = ? \text{ và } c + a = ?$$

như vậy dàn D trong hình 1.23b có các phần tử không có phần tử bù là a và c.

Còn trong hình 1.23c ta thấy các phần tử của nó đều có quan hệ :

$a \,.\, b = \Phi$ và $a + b = 1$	vậy có a là phần bù của b.
$b \,.\, c = \Phi$ và $b + c = 1$	vậy có b là phần bù của c.
$c \,.\, a = \Phi$ và $c + a = 1$	vậy có c là phần bù của a.

Kết quả là mọi phần tử của dàn D biểu diễn trên hình 1.23c đều có phần tử bù, vậy đó chính là dàn bù. Nhưng một phần tử không thể có nhiều phần tử bù được vì vậy có định lý sau.

Định lý 1.4.5. Trong một dàn phân bố bù, nếu có phần tử bù thì phần tử bù đó là duy nhất.

Chứng minh : Giả sử a có hai phần tử bù là b và c, như vậy ta có quan hệ :

$$a.b = 0 \qquad a + b = 1$$

$$a.c = 0 \qquad a + c = 1$$

Mặt khác ta có :

$$b = b.1 = b.(a + c) = (b.a) + (b.c) = 0 + (b.c) = (a.c) + (b.c) =$$
$$= (a + b) \,.\, c = c$$

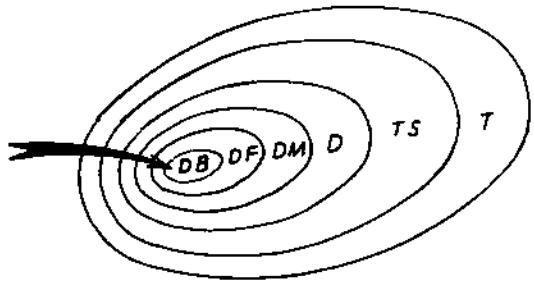
Như vậy định lý đã được chứng minh.

Một dàn phân bố bù còn được gọi là một dàn Boole, nó là dàn hai trị (0,1), giá trị cực đại là 1 và giá trị cực tiểu là 0. Như vậy số lượng phần tử của một dàn Boole là chẵn, tức là nếu có một dàn Boole thì ta sẽ tìm được một số nguyên dương n sao cho số lượng phần tử của dàn Boole nhận được là 2^n.

Tóm lại từ các định nghĩa của dàn ta nhận thấy dàn Boole là tập con của dàn phân bố, nó có đầy đủ các tính chất của dàn phân bố, mà dàn phân bố lại là tập con của dàn

môđun, tất nhiên dàn môđun phải là tập con của dàn, ta đã biết dàn là một tập mà trong đó các phần tử của nó đều có các phép toán giao và phép toán hợp, nói cách khác dàn là một tập sắp xếp mà trong đó hai phần tử bất kỳ x và y có phép toán giao và phép toán hợp, nghĩa là tập hợp của dàn là tập con của tập sắp xếp, còn tập sắp xếp lại là tập con của tập hợp nói chung. Như vậy chúng ta đã đi từ lý thuyết tập (T), đến tập sắp xếp (TS), rồi tới dàn (D), đến dàn môđun (DM), đến dàn phân bố (DF), cuối cùng đi đến dàn phân bố bù chính là đi đến dàn quan trọng nhất dàn Boole (DB).

Như vậy dàn Boole có đầy đủ các tính chất của các lớp dàn khác, và cũng từ đây trở đi ta sẽ chỉ nói tới dàn Boole mà thôi. Quá trình để đi tới dàn Boole của chúng ta được mô tả trên hình 1.24.



Hình 1.24. Tổ chức đến dàn Boole.

§1.5. ĐẠI SỐ BOOLE VÀ HÀM BOOLE

Tới tiết này ta đã có trong những phép toán cơ bản và quan trọng nhất của đại số lôgic hay là đại số Boole đó là những phép toán hợp (+) phép toán hai ngôi, phép toán giao (•) phép toán hai ngôi, và phép toán phủ định (¬) phép toán một ngôi. Những phép toán đó đủ để tạo nên một cấu trúc đại số hoàn chỉnh, dùng chúng ta có thể gia công, biến đổi hàm lôgic dựa vào các tính chất hay tiên đề của chúng. Từ các phép toán đó ta có một cấu trúc đại số mới–đại số lôgic hay là đại số Boole.

1.5.1. ĐẠI SỐ BOOLE VÀ CÁC TÍNH CHẤT CỦA NÓ

Định nghĩa 1.5.1. Đại số Boole là một hệ thống đại số $<+, \cdot, \neg, 0, 1, B>$ trong đó B bao gồm các tập hợp, các phép toán hai ngôi cộng (+) và nhân (.), và phép toán một ngôi phủ định (¬), có giá trị bé nhất là 0 và giá trị lớn nhất là 1, thỏa mãn các tiên đề sau :

$Đ_1$:	$x . x = x$	$x . x = x$
$Đ_2$:	$x . y = y . x$	$x + y = y + x$
$Đ_3$:	$x . (y . z) = (x . y) . z$	$x + (y + z) = (x + y) + z$
$Đ_4$:	$x . (x + y) = x$	$x + (x . y) = x$
$Đ_5$:	$x . (y + z) = (x . y + (x . z)$	$x + (y . z) = (x + y) . (x + z)$

$$Đ_6 : \quad x.\bar{x} = 0 \qquad\qquad x + \bar{x} = 1$$

$$x.1 = x \qquad\qquad x + 1 = 1$$

$$x.0 = 0 \qquad\qquad x + 0 = x$$

$$\bar{\bar{x}} = x$$

Các tiên đề trên là cơ sở để biến đổi đại số Boole, ta nhận thấy đây là một cấu trúc đại số mạnh, vì nó có nhiều phép toán, đồng thời các phép toán chúng hoàn toàn độc lập với nhau. Một trong những thế mạnh nữa của đại số này mà các đại số khác không thể có được đó là các phép toán của nó đều có thể thực hiện được bằng mạch điện. Điều đó cho phép ta lắp mạch điện cho phương trình toán học, và ngược lại một phương trình toán học có một mạch điện tương ứng. Đó là một khái niệm mới mà chỉ có đại số này mới cho được, mới có được. Từ khái niệm đó ta lại có thể nhận được một khái niệm mới nữa là : biến đổi phương trình toán chính là biến đổi mạch điện hay nói một cách khác thay đổi mạch điện ta dùng tiên đề toán học. Đó là những khái nhiệm rất mới đối với các nhà toán học cũng như các nhà kỹ thuật điện tử, nhưng nó sẽ rất quen thuộc với chúng ta sau này.

Quan hệ hai ngôi trên một tập của các phép toán đó được minh hoạ ở ví dụ sau đây.

Ví dụ : Cho đại số Boole 4 phần tử B_4 = (0, a, b, 1), quan hệ hai ngôi của các phép toán thể hiện trên các bảng sau.

•	0	a	b	1
0	0	0	0	0
a	0	a	0	a
b	0	0	b	b
1	0	a	b	1

+	0	a	b	1
0	0	a	b	1
a	a	a	1	1
b	b	1	b	1
1	1	1	1	1

–	1
0	1
a	b
b	a
1	0

Hình 1.25. Các phép toán cơ bản.

Cho tập A = (a, b), xét một tập phân phối của tập A là P(A) có quan hệ với các phép toán trong đại số Boole như thế nào.

$$P(A) = ((\Phi), (a), (b), (a, b))$$

Tác động của ba phép toán lên tập P (A) được minh họa bằng các bảng quan hệ hai ngôi và quan hệ một ngôi trên hình 1.26.

Nhận thấy P (A) đẳng cấu với B_4 có thể minh hoạ trên hình 1.26.

Ví dụ : Đại số Boole với 4 phần tử B_4 = (0, a, b, 1) đẳng cấu với tích trực tiếp của B_1 và B_2 trong đó B_1 = (0,1) và B_2 = (0,1).

$B = B_2 X B_2$ = (00, 01, 10, 11), điều đó được minh họa trên hình 1.27,b.

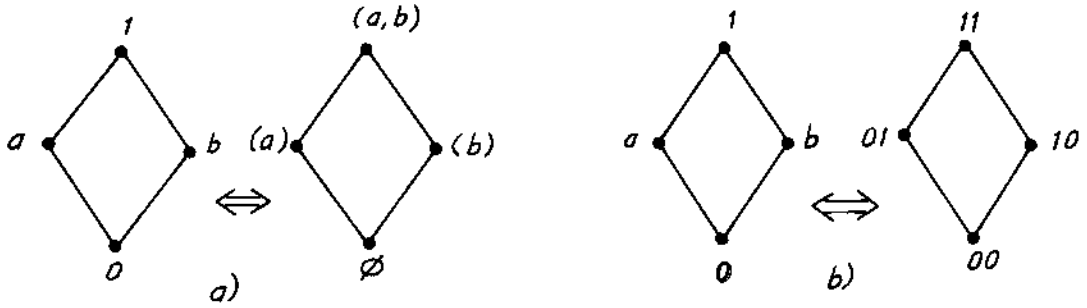
Một trong những định lý quan trọng của đại số Boole là định lý De Morgan, quan hệ của định lý như sau.

•	(ϕ)	(a)	(b)	(a,b)
(ϕ)	ϕ	ϕ	ϕ	ϕ
(a)	ϕ	(a)	ϕ	(a)
(b)	ϕ	ϕ	(b)	(b)
(a,b)	ϕ	(a)	(b)	(a,b)

+	(ϕ)	(a)	(b)	(a,b)
(ϕ)	ϕ	(a)	(b)	(a,b)
(a)	(a)	(a)	(a,b)	(a,b)
(b)	(b)	(a,b)	(b)	(a,b)
(a,b)	(a,b)	(a,b)	(a,b)	(a,b)

−	1
(ϕ)	(a,b)
(a)	(b)
(b)	(a)
(a,b)	(a,b)

Hình 1.26.



Hình 1.27. Quan hệ hai ngôi của phép toán.

Định lý 1.5.1. Trong đại số Boole thì :

$$\overline{(a + b)} = \bar{a} \cdot \bar{b}$$

$$\overline{(a \cdot b)} = \bar{a} + \bar{b}$$

Chứng minh : Ta có quan hệ

$$(a + b) + (\bar{a} \cdot \bar{b}) = \Big((a + b) + \bar{a}\Big) \cdot \Big(a + b) + \bar{b}\Big) \qquad \text{(Tính phân phối)}$$

$$= \Big((a + \bar{a}) + b\Big) \cdot \Big(a + (b + \bar{b})\Big)$$

$$= (1 + b) \cdot (a + 1)$$

$$= 1 + 1 = 1$$

Đồng thời :

$$(a + b) \cdot (\bar{a} \cdot \bar{b}) = \Big((a \cdot (\bar{a} \cdot \bar{b}) + (b \cdot (\bar{a} \cdot \bar{b})\Big)$$

$$= (a \cdot \bar{a}) \cdot \bar{b} + (b \cdot \bar{b}) \cdot \bar{a}$$

$$= 0 + 0 = 0$$

Vậy $(\bar{a} \cdot \bar{b})$ là phần tử bù của $(a + b)$, do đó thỏa mãn :

$$\overline{(a + b)} = \bar{a} \cdot \bar{b}$$

Biểu thức thứ hai của định lý có thể tự chứng minh theo đối ngẫu hoặc dùng bảng lôgic.

Định lý De Morgan là định lý bình thường trong toán học nhưng lại quan trọng trong kỹ thuật, vì nó cho phép chuyển đổi giữa phép toán giao thành phép toán hợp và ngược lại, điều đó tương ứng với việc chuyển đổi giữa mạch VÀ (phép giao) thành mạch HOẶC (phép hợp) và ngược lại, việc đó trong kỹ thuật được ứng dụng để xây dựng, lắp ráp các hệ thống có cùng một loại mạch hay chỉ dùng một loại mạch (toàn dùng mạch

VÀ) hay (toàn là mạch HOẶC), hay thay thế mạch này thành mạch khác. Điều đó cho phép dễ dàng lắp ráp, dễ dàng thay thế, dễ dàng sửa chữa mạch điện trong hệ thống số. Đó là đặc điểm của hệ thống số, mạch số khác hẳn các mạch điện khác, và ta dễ dàng nhận thấy là trong nó có rất nhiều mạch giống nhau, hay rất nhiều IC giống nhau cùng với loại dùng trong mạch số cũng như trong hệ thông số.

1.5.2. HÀM BOOLE VÀ CÁC DẠNG TỔNG QUÁT CỦA NÓ (DẠNG CHÍNH QUY)

Trong phần quan hệ và hàm số mà ta đã nói ở trên, một trong những khái niệm quan trọng để hiểu bản chất của hàm Boole là nghiên cứu các ánh xạ của Boole từ đại số Boole lên chính bản thân nó. Ta đã có cấu trúc của đại số Boole $<+ , \bullet , \neg, 0,1,B>$ vậy hàm của nó được cấu tạo theo định nghĩa sau.

Định nghĩa 1.5.2. Cho $X = (x_1, x_2, ..., x_n)$ là tập các biến của đại số Boole B. Một ánh xạ f của B vào chính bản thân nó gọi là một hàm Boole n biến nếu nó được cấu tạo theo những nguyên tắc sau :

a) Cho a là một hằng số trên B. Hàm hằng số có quan hệ :

$$f(X) = a$$

Cho x_i là một biến của B. Hàm chiếu của hàm Boole được viết :

$$f(X) = x_i$$

b) Nếu f(X) là một hàm Boole thì f(X) cũng là một hàm Boole.

c) Nếu $f_1(X)$ và $f_2(X)$ là các hàm Boole thì $f_1(X) . f_2(X)$ và $f_1(X) + f_2(X)$ cũng là các hàm Boole.

d) Một hàm số được cấu tạo bằng cách áp dụng một số hữu hạn lần các quy luật kể trên đều là hàm Boole. Như vậy hàm Boole là một hàm số được cấu tạo từ các hàm hằng số và hàm chiếu bằng cách áp dụng một số hữu hạn lần các phép toán giao ($\bullet$), phép toán hợp (+) và phép toán phủ định ($\neg$). Tóm lại khi tương tác một phép toán của đại số Boole thì ta vẫn nhận được hàm Boole.

Ta có thể hiểu hàm Boole một cách chính xác theo định nghĩa toán học, nhưng trong kỹ thuật chúng ta cũng cần hiểu bản chất của hàm Boole một cách chính xác để ứng dụng. Hàm số, có thể nói bản chất của nó chính là sự ràng buộc, sự ràng buộc giữa biến x với hàm số f(x), nó cũng chính là sự phụ thuộc, là quan hệ, là sự tương tác lẫn nhau giữa các thành phần để tạo nên một chức năng. Cũng có thể hiểu hàm số chính là trật tự, là trình tự kỹ thuật thực hiện một chức năng nào đó, vậy hàm số đã được hiểu một cách lỏng lẻo hơn. Ở đây khi ta coi quan hệ của hàm số f(x) lỏng lẻo hơn nhưng thực tế hơn thì ta cũng dễ dàng thể hiện ý muốn, chức năng, hay nhiệm vụ của mình thành hàm số.

Khi ta coi hàm số f(x) trong toán học không còn cứng nhắc như trước nữa mà coi đó chính là một chức năng kỹ thuật thì ta đã bước sang một khái niệm quan trọng của quyển sách này chính là chuyển ý muốn, nhiệm vụ, chức năng kỹ thuật thành quan

hệ toán học, thành phương trình toán, thành hàm số f(x) – hàm số trong toán học mới, hàm số của toán học hiện đại toán lôgic.

Trước kia khi nói tới hàm số là nói tới phương trình f(x), nhưng khi ta coi hàm số là một chức năng kỹ thuật, thì chức năng có thể biểu diễn không ở dạng f(x) nữa. Từ đó chúng ta nhận thấy một khái niệm mới nữa là : biểu diễn hàm số không còn ở dạng f(x), mà ở dạng bảng chức năng, bảng chân lý hay là bảng thật. Quyển sách này sẽ cho ta vài cách biểu diễn hàm số dưới dạng bảng chức năng. Lúc này ta hiểu bảng chức năng chính là một dạng mới mô tả hàm số, mô tả quan hệ ràng buộc, ví dụ như sự ràng buộc giữa tín hiệu vào (tín hiệu điều khiển) với đáp ứng ra (tín hiệu chấp hành) của một mạch điện chức năng lôgic nào đó.

Ví dụ : Cho hàm Boole biểu diễn ở bảng chức năng trên hình 1.28 : cho hàm ba biến $f(x_1, x_2, x_3)$ hoạt động theo bảng.

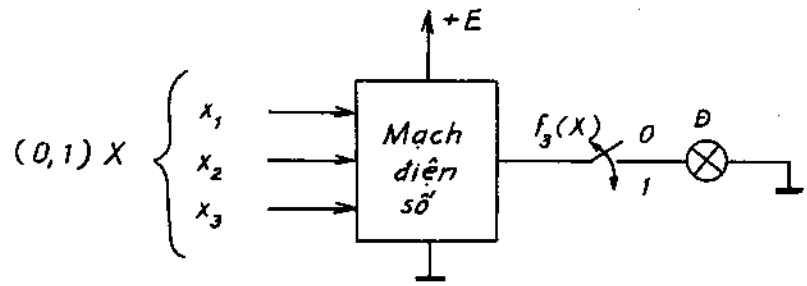
x_1	x_2	x_3	$f_3(X)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$X = (x_1, x_2, x_3)$$

$$f_3(X) = f(x_1, x_2, x_3)$$

Hình 1.28. Bảng chức năng–bảng thuật hàm Boole.

Bảng đó cũng chính là một hàm Boole theo toán học, và nó có một phương trình toán học tương ứng. Nhưng đồng thời nó cũng chính là một chức kỹ thuật của mạch điện trên hình 1.29.



Hình 1.29. Mạch điện–hàm Boole.

Mạch điện đó có hai thông tin chính đó là : tín hiệu vào (X) và đáp ứng ra $f_3(X)$. Tín hiệu được đưa vào trên các đầu vào x_1, x_2, x_3, chúng lấy các giá trị lôgic là 0 hoặc 1 tương ứng với mức điện áp là thấp hay cao, tín hiệu vào đó đồng thời tương ứng với các biến số (x_1, x_2, x_3) của hàm số $f_3(X)$ trong phương trình toán. Còn đầu ra, hay đáp

ứng ra của mạch điện thì tương ứng với kết quả của hàm số $f_3(X)$ ở phương trình toán, nó có giá trị lôgic là 0 hoặc 1 ứng với mức điện áp ra là thấp hoặc cao sẽ làm cho đèn báo sáng Đ tối hay sáng. Nếu đèn báo sáng Đ ta thay bằng một máy công cụ hay một quan hệ chấp hành nào đó thì ta có một hệ thống chức năng đơn giản.

Phương trình toán học của bảng chức năng đó được viết :

$$f_3(X) = (\bar{x}_1 \cdot x_2 \cdot \bar{x}_3 + \bar{x}_1 \cdot x_2 \cdot x_3 + x_1 \cdot \bar{x}_2 \cdot x_3 + x_1 \cdot x_2 \cdot \bar{x}_3 + x_1 \cdot x_2 \cdot x_3)$$

$$f_3(X) = \Big((0\ \ 1\ \ 0) \quad (0\ \ 1\ \ 1) \quad (1\ \ 0\ \ 1) \quad (1\ \ 1\ \ 0) \quad (1\ \ 1\ \ 1)\Big)$$

Từ phương trình toán học đó ta có thể hiểu như sau : Hàm số $f_3(X)$ sẽ có giá trị là 1 nếu cho biến số theo quan hệ $(\bar{x}_1, x_2, \bar{x}_3)$ hoặc $(\bar{x}_1, x_2, x_3)$ hoặc $(x_1, \bar{x}_2, x_3)$ ... còn $f_3(X) = 0$ khi biến số là $(\bar{x}_1, \bar{x}_2, \bar{x}_3)$. Điều đó tương ứng với đáp ứng ra của mạch điện làm cho đèn Đ sáng khi cho tín hiệu vào là (0 1 0) hoặc (0 1 1) hoặc ..., còn đèn Đ sẽ tắt khi gặp tổ hợp tín hiệu (0 0 0)...

Trong mạch điện nói chung khi nói tới "tín hiệu vào" là chỉ nói tới tín hiệu điện thuần túy như tín hiệu vào anten TV, tín hiệu đó hoàn toàn mang tính chất bị động. Nhưng trong mạch lôgic thì "tín hiệu vào" thực chất nó không chỉ là tín hiệu điện mà còn là ý muốn của ta nữa, vì thế trong mạch lôgic tín hiệu vào có tính chủ động. Còn đầu ra của mạch điện điện áp ra chính là kết quả đáp ứng của ý muốn ở đầu vào. Ví dụ từ bảng chức năng trên khi ta chủ động cho tín hiệu vào là (0 0 0) thì đáp ứng ra là 0 đúng theo ý muốn của ta, trong thực tế khi ta bấm lên bàn phím của máy vi tính cũng chính là ta thể hiện ý muốn của mình, và màn hình của máy sẽ hiện ra đáp ứng của ý muốn đó của ta. Tương tự cũng giống như khi ta chơi điện tử bấm tay lên bàn phím của trò chơi thì : ý muốn của ta lúc bấm đó sẽ tạo ra tín hiệu vào đưa vào trong máy tính, và đáp ứng ra của nó chính là kết quả điện tử thể hiện trên màn hình. Vì vậy hệ thống số chính là hệ thống thể hiện ý muốn của con người tới hệ thống chấp hành chức năng thông qua mạch lôgic với tín hiệu khép kín, tức là tín hiệu được xử lý trong nội bộ hệ thống chứ không chuyển qua không gian.

Đặc điểm của loại mạch lôgic là tín hiệu vào rời rạc, đáp ứng ra tín hiệu ở đầu ra của mạch cũng ở dạng rời rạc, chủ yếu dùng điều khiển các chức năng.

Từ bảng chức năng trên hình 1.30 ta nhận thấy bảng đó có hai khu vực, khu tín hiệu vào (x_1, x_2, x_3) và khu đáp ứng ra $f_3(X)$. Trong khu vực tín hiệu vào biểu diễn toàn bộ các khả năng xảy ra ở đầu vào của hệ thống – toàn bộ các khả năng cho tín hiệu vào. Nếu ta có 3 đầu vào tín hiệu thì sẽ có 8 khả năng đưa tín hiệu vào khác nhau, đó cũng chính là tất cả toàn bộ các khả năng có thể có của tín hiệu vào. Vậy nếu ta có n đầu vào tín hiệu thì sẽ có 2^n khả năng cho vào tín hiệu. Khi ta nói tới toàn bộ mọi khả năng của tín hiệu đầu vào thì cũng có nghĩa là những khả năng đó là của chung các mạch điện có cùng số lượng đầu vào, vậy nó luôn luôn cố định với bất kỳ chức năng nào có cùng số đầu vào, nhưng ở khu vực đáp ứng ra $f_3(X)$ thì thay đổi giá trị theo những chức năng khác nhau, tức là mỗi một chức năng thì khu vực đáp ứng ra có giá trị khác nhau, có đáp ứng ra khác nhau. Điều đó được minh họa bằng ví dụ sau.

Ví dụ : Biểu diễn bằng chức năng của nhiệm vụ có một mạch điện có ba đầu vào mà cứ hai đầu vào có cùng giá trị 1 thì báo, bảng đó được biểu diễn trên hình 1.30,a. Một ví dụ khác dùng bảng chức năng mô tả kết quả của bộ tổng có ba đầu vào giá trị, khi kết quả tổng bằng 1 thì báo trên đáp ứng ra, bảng được minh họa trên hình 1.30,b.

x_1	x_2	x_3	$f_3(X)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

a)

x_1	x_2	x_3	$f_3(X)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

b)

Hình 1.30. Minh họa các chức năng của hàm Boole.

Nhận thấy hai hàm số tương đương với nhau nếu chúng có các bảng chức năng hoàn toàn như nhau, nói cách khác hai chức năng tương đương thì bảng thật của chúng giống nhau. Với cách biểu diễn hàm số như vậy ta thấy :

Nếu có một biến sẽ tạo được 4 hàm biểu diễn trên hình 1.31,a.

x		f(x)
0	1	
0	0	$f_1 = 0$
0	1	$f_2 = x$
1	0	$f_3 = \bar{x}$
1	1	$f_4 = 1$

Hình 1.31,a.

Nếu có 2 biến sẽ tạo được 16 hàm thấy được trên hình 1.31,b.

Tổng quát nếu có n biến sẽ nhận được 2^{2^n} hàm số khác nhau.

Ta đã biết hàm Boole có thể được biểu diễn bằng bảng chức năng cũng như bằng quan hệ toán học, mà bảng chức năng (chức năng kỹ thuật) lại có rất nhiều cho nên biểu thức toán học cũng sẽ có nhiều. Nhưng dưới dạng toán học ta có thể biểu diễn bằng một biểu thức tổng quát dùng chu cho mọi chức năng đó là "dạng chính quy" hay "dạng tổng quát" của hàm Boole (đó là điều mà đại số khác không có). Dạng "tổng quát" của hàm Boole có hai dạng tương đương nhau là : Dạng "tuyển chuẩn tắc" (tổng của các tích) và dạng "hội chuẩn tắc" (tích của các tổng), chúng được biểu diễn như sau.

1.5.2.1. Dạng tuyển chuẩn tắc toàn phần (tổng quát tích)

$$f(x_1, \ldots, x_n) = \sum_{11..1}^{00..0} f(e_1, \ldots, e_n) \cdot x_1^{e_1} \cdot x_2^{e_2} \ldots x_n^{e_n}$$

X (x_1, x_2)				$f(x_1, x_2)$	Tên gọi - Chức năng
00	01	10	11		
0	0	0	0	$f_1 = 0$	Hàm 0 (hở mạch)
0	0	0	1	$f_2 = x_1 x_2$	Hàm AND (mạch VÀ)
0	0	1	0	$f_3 = x_1 \not\supset x_2$	Hàm kéo theo không điều kiện
0	0	1	1	$f_4 = x_1$	Hàm chiếu x_1
0	1	0	0	$f_5 = x_1 \not\subset x_2$	Hàm kéo theo không đảo
0	1	0	1	$f_6 = x_2$	Hàm chiếu x_2
0	1	1	0	$f_7 = x_1 \oplus x_2$	Hàm cộng modul 2
0	1	1	1	$f_8 = x_1 + x_2$	Hàm OR (mạch HOẶC)
1	0	0	0	$f_9 = \overline{x_1 + x_2}$	Hàm NOR (mạch HOẶC ĐẢO)
1	0	0	1	$f_{10} = x_1 \equiv x_2$	Hàm tương đương
1	0	1	0	$f_{11} = \overline{x}_2$	Hàm chiếu x_2 phủ định
1	0	1	1	$f_{12} = x_1 \subset x_2$	Hàm kéo theo đảo
1	1	0	0	$f_{13} = \overline{x}_1$	Hàm chiếu x_1 phủ định
1	1	0	1	$f_{14} = x_1 \supset x_2$	Hàm kéo theo có điều kiện
1	1	1	0	$f_{15} = \overline{x_1 \cdot x_2}$	Hàm NAND (mạch VÀ ĐẢO
1	1	1	1	$f_{16} = 1$	Hàm 1 (đóng mạch)

Hình 1.31,b.

Trong đó : $x_i^{e_i} = \begin{cases} x_i & \text{nếu } e_i = 0 \\ \overline{x}_i & \text{nếu } e_i = 1 \end{cases}$ $\qquad (1 \leqslant i \leqslant n)$

$E = (e_1, ..., e_n)$ là số nhị phân n ký hiệu (n biến)

$f(e_1, ..., e_n)$ lấy giá trị bằng 0 hoặc 1 tương ứng với chức năng

Ví dụ : Cho một chức năng kỹ thuật ở bảng trên hình 1.32.

Từ bảng chức năng đó ta có biểu thức toán học tương ứng có dạng :

$$f(x_1, x_2, x_3) = \overline{x}_1 \cdot x_2 \cdot \overline{x}_3 + \overline{x}_1 \cdot x_2 \cdot x_3 + x_1 \cdot \overline{x}_2 \cdot \overline{x}_3 + x_1 \cdot x_2 \cdot \overline{x}_3 + x_1 \cdot x_2 \cdot x_3$$

Ta có thể dùng dạng tuyển chuẩn tắc toàn phần để biểu diễn được chính chức năng kỹ thuật đó như bảng chức năng sau :

x_1	x_2	x_3	$f(x_1,x_2,x_3)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Hình 1.32. Bảng chức năng cụ thể.

$$f(x_1, x_2, x_3) = f(000) \cdot \bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3 + f(001) \cdot \bar{x}_1 \cdot \bar{x}_2 \cdot x_3 + f(010) \cdot \bar{x}_1 \cdot x_2 \cdot \bar{x}_3 +$$
$$+ f(011) \cdot \bar{x}_1 \cdot x_2 \cdot x_3 + f(100) \cdot x_1 \cdot \bar{x}_2 \cdot \bar{x}_3 + f(101) \cdot x_1 \cdot \bar{x}_2 \cdot x_3 +$$
$$+ f(110) \cdot x_1 \cdot x_2 \cdot \bar{x}_2 + f(111) \cdot x_1 \cdot x_2 \cdot x_3$$

Với chức năng của bảng trên ta thấy :

$f(000) = 0$ nên toàn bộ tích $f(000)\ \bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3$ có giá trị bằng 0. Tương tự $f(001) = 0$ và $f(101) = 0$ nên tích số tương ứng giá trị cũng bằng 0, ở đây mỗi tích số còn được gọi là một implicăng hay một mintéc.

$f(010) = 1$ theo giá trị của bảng chức năng nên tích $f(010)\ \bar{x}_1 \cdot x_2 \cdot \bar{x}_3$ không bị mất đi trong biểu thức của hàm số. Tương tự $f(011) = 1$, $f(100) = 1$, $f(110) = 1$ và $f(111) = 1$ cho nên những tích (implicăng) tương ứng với chúng được tồn tại trong biểu thức toán học. Kết quả ta lại nhận được biểu thức dùng dạng tuyển chuẩn tắc toàn phần giống hệt biểu thức đã viết ở trên :

$$f(x_1, x_2, x_3) = \bar{x}_1 \cdot x_2 \cdot \bar{x}_3 + \bar{x}_1 \cdot x_2 \cdot x_3 + x_1 \cdot \bar{x}_2 \cdot \bar{x}_3 + x_1 \cdot x_2 \cdot \bar{x}_3 + x_1 \cdot x_2 \cdot x_3$$

Như vậy khi biểu diễn hàm Boole dưới dạng tuyển chuẩn tắc tổng quát, nếu như muốn có những chức năng khác nhau chúng ta chỉ cần thay đổi giá trị của $f(e_1,...,e_n)$ bằng 0 hoặc bằng 1 tương ứng với chức năng mà ta cần là xong, điều đó chính là cơ sở của phần mềm lập trình trong máy tính sau này.

1.5.2.2. Dạng hội chuẩn tắc toàn phần : (tích của các tổng)

$$f(x_1, x_2, \ldots, x_n) = \prod_{00..0}^{11..1} \left(f(e_1, \ldots, e_n) + x_1^{\bar{e}_1} + x_2^{\bar{e}_2} + \ldots + x_n^{\bar{e}_n} \right)$$

Trong đó : $x_i^{\bar{e}_i} = \begin{cases} x_i & \text{nếu } e_i = 0 \\ \bar{x}_i & \text{nếu } e_i = 1 \end{cases} \qquad (1 \leq i \leq n)$

$E = (e_1, \ldots, e_n)$ là số nhị phân n ký hiệu (n biến)

$f(e_1, e_2, \ldots, e_n)$ có giá trị bằng 0 hoặc 1 tương ứng theo bảng chức năng.

Ví dụ : Vẫn với chức năng cho ở bảng chức năng biểu diễn trên bảng hình 1.32, ta biểu diễn nó dưới dạng hội chuẩn tắc toàn phần như sau :

$$f(x_1, x_2, x_3) = \left(f(000) + x_1 + x_2 + x_3 \right) \cdot \left(f(001) + x_1 + x_2 + \bar{x}_3 \right) \cdot$$
$$\cdot \left(f(010) + x_1 + \bar{x}_2 + x_3 \right) \cdot \left(f(011) + x_1 + \bar{x}_2 + \bar{x}_3 \right) \cdot \left(f(100) + \bar{x}_1 + x_2 + x_3 \right) \cdot$$
$$\cdot \left(f(101) + \bar{x}_1 + x_2 + \bar{x}_3 \right) \cdot \left(f(110) + \bar{x}_1 + \bar{x}_2 + x_3 \right) \cdot \left(f(111) + \bar{x}_1 + \bar{x}_2 + \bar{x}_3 \right)$$

Do vẫn mô tả chức năng cho ví dụ ở phần trên nên ta có quan hệ $f(010) = 1$, mà trong đại số Boole lại có tính chất $1 + x = 1$, cho nên tổng $(1 + x_1 + \bar{x}_2 + x_3) = 1$ do đó nó không cần viết trong quan hệ tích (coi như tổng đó biến mất). Tương tự như thế ta biết $f(011) = 1$, $f(100) = 1$, $f(110) = 1$ và $f(111) = 1$ nên tổng có chứa các giá trị đó có thể bỏ qua trong phương trình.

Vậy chỉ còn các giá trị f(000) = 0, mà 0 + x = x, cho nên tổng có chứa f(000) sẽ tồn tại $(0 + x_1 + x_2 + x_3) = (x_1 + x_2 + x_3)$. Tương tự như vậy các tổng có chứa các giá trị f(001) = 0 và f(101) = 0 thì tồn tại và được viết trong dạng hội chuẩn tắc toàn phần ta nhận được dạng như sau :

$$f(x_1, x_2, x_3) = (x_1 + x_2 + x_3) . (x_1 + x_2 + \bar{x}_3) . (\bar{x}_1 + x_2 + \bar{x}_3)$$

Sau đó ta phá ngoặc và dựa vào quan hệ $x_1 . \bar{x}_1 = 0$ để rút gọn thì sẽ thu được kết quả của hàm số giống hệt dạng tuyển chuẩn tắc quen thuộc của hàm số đó. Mỗi một tổng còn được gọi là maxtéc.

Như vậy một hàm Boole có thể được biểu diễn hoàn toàn như nhau ở hai dạng "dạng tổng quát" là tuyển chuẩn tắc hoàn toàn phần và hội chuẩn tắc toàn phần. Ở dạng tuyển chuẩn tắc toàn phần thì ta chỉ chú ý tới các giá trị 1 của hàm số còn ở dạng hội chuẩn tắc toàn phần thì ta chỉ chú ý tới các giá trị 0 của hàm số.

1.5.3. HÀM BOOLE VÀ CÁC PHƯƠNG PHÁP BIỂU DIỄN

Đối với các loại đại số khác (đại số cổ điển) việc chuyển đổi giữa các dạng hàm số của chúng với nhau rất phức tạp và cần rất nhiều các điều kiện kèm theo dựa trên các định lý toán học. Mặc dù vậy khi chuyển đổi được đôi khi vẫn có sai số kèm theo, nhưng không phải phương trình toán học nào chúng ta cũng có thể tìm được phương trình tương đương của nó biểu diễn ở dạng khác, đó là điều chúng ta đều biết. Nhưng với hàm Boole thì công việc đó lại hoàn khác, chúng ta có thể chuyển đổi dễ dàng một hàm Boole từ dạng này sang dạng khác mà không cần bất kỳ một điều kiện nào, mà kết quả nhận được hoàn toàn chính xác.

Chúng ta có rất nhiều dạng biểu diễn khác nhau của cùng một hàm Boole, điều đó cho phép chúng ta nhìn nhận được tối đa sự hiểu biết một hàm Boole dưới các góc cạnh khác nhau để hiểu nó, và tận dụng ưu thế đó để sử dụng thích hợp cho mình.

1.5.3.1. Dạng bảng chức năng

Bảng chức năng là dạng rất quen thuộc, là dạng cơ bản và là dạng quan trọng nhất. Nó cho phép chúng ta thể hiện rõ nhất và dễ nhất nhiệm vụ kỹ thuật của mình, ví dụ bảng chức năng trên hình 1.33.

x_1	x_2	x_3	f(X)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

$X = (x_1, x_2, x_3)$

Hình 1.33. Bảng chức năng

Với cách biểu diễn bằng bảng cho phép ta dễ dàng diễn đạt các chức năng kỹ thuật theo ý muốn của mình. Từ bảng chức năng ta có thể biểu diễn nhiệm vụ kỹ thuật đó một cách dễ dàng dưới dạng toán học như sau.

1.5.2.2. Dạng biểu thức toán học f(X)

Vẫn từ bảng chức năng biểu diễn trên hình 1.33 thì dạng biểu thức toán học của nó nhận được dễ dàng và quen thuộc như sau :

$$f(x_1, x_2, x_3) = \bar{x}_1 . x_2 . \bar{x}_3 + \bar{x}_1 . x_2 . x_3 + x_1 . \bar{x}_2 . \bar{x}_3 + x_1 . x_2 . \bar{x}_3 + x_1 . x_2 . x_3$$

Khi biểu diễn được một nhiệm vụ kỹ thuật thành một biểu thức toán học tương ứng, thì ta có thể biến đổi biểu thức toán học đó theo các tiên đề, điều đó đồng nghĩa với việc gia công chức năng kỹ thuật, sau này việc làm đó sẽ là thay đổi mạch điện bằng biến đổi toán học.

1.5.2.3. Dạng mã số với các giá trị (0,1)

Tương ứng với ví dụ trên hàm ba biến đó có thể được biểu diễn dưới dạng khác – dạng (0,1) như sau :

$$f(x_1, x_2, x_3) = \{(010)(011)(100)(110)(111)\}$$

Cách biểu diễn này dễ dàng đưa số liệu của hàm số vào máy với số lượng biến lớn để máy tính biến đổi, đồng thời ta cũng hiểu cách đưa tín hiệu vào của hệ thống kỹ thuật.

1.5.3.4. Dạng giá trị con số nhị phân

Ứng với cách biểu diễn chức năng kỹ thuật (hàm số lôgic) ở dạng (0, 1) thì nó có các giá trị của con số kèm theo như sau :

$$f(x_1, x_2, x_3) = (2, 3, 4, 6, 7)$$

Với cách biểu diễn này ta có thể đơn giản hóa cách thể hiện hàm lôgic, thể hiện gọn gàng hơn nhưng vẫn chứa đầy đủ các thông tin cần thiết.

1.5.3.5. Dạng bìa Kácnô (Karnaugh)

Một trong những cách biểu diễn hàm lôgic được ứng dụng nhiều nhất trong việc biến đổi và gia công toán học là bìa Kácnô. Đó là dạng biểu diễn rất thuận lợi cho biến đổi hàm lôgic, điều đó sẽ được nói kỹ trong phần rút gọn hàm lôgic sau này. Bìa Kácnô của hàm lôgic trong ví dụ ở trên có dạng như trên hình 1.34.

Nhìn vào bìa Kácnô thấy có cấu tạo hai trục (trục x_1 và trục x_2 x_3), từ hai trục đó tạo ra các toạ độ trong bảng, mỗi một ô trong bảng tương ứng với một toạ

x_1 \ x_2x_3	00	01	11	10
0			1	1
1	1		1	1

Hình 1.34. Dạng bìa Kácnô.

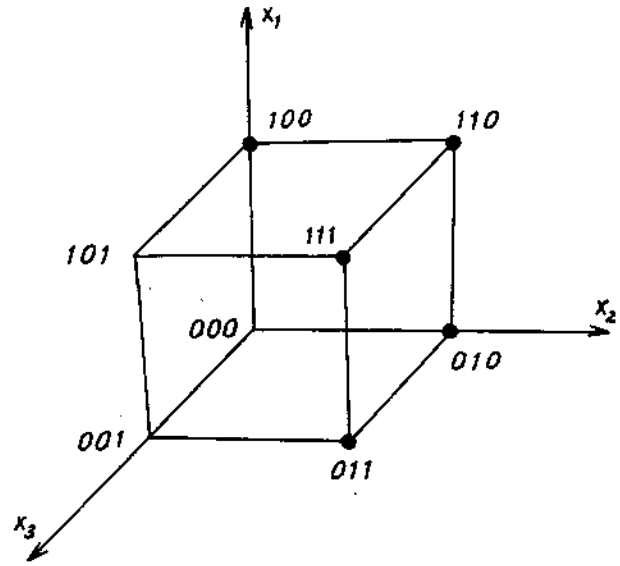
độ nhất định. Trên hai trục có các giá trị (0,1) khi ghép chúng lại thì tạo ra tọa độ của ô đó ($(010) - \bar{x}_1 . x_2 . \bar{x}_3$), đồng thời giá trị (010) đó cũng là tổ hợp của tín hiệu vào của hệ thống – nó cũng ứng với một tích số trong biểu thức toán (hai trục tương ứng với tập tín hiệu vào của chức năng), còn các ô bên trong bìa K lấy giá trị 1 hoặc 0 (thường là ô bỏ trống) tương ứng với $f(x_1, x_2, x_3)$ trong bảng chức năng, đó cũng là đáp ứng ra của hệ thống kỹ thuật (đáp ứng ra của mạch điện).

1.5.3.6. Dạng hình học của hàm lôgic

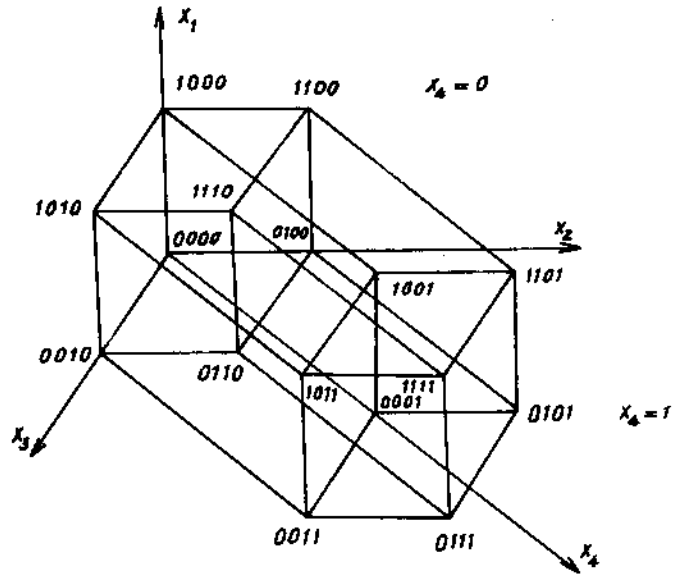
Vẫn với hàm ba biến f(X) trong ví dụ trên, thì hàm đó có thể được biểu diễn trên hình 1.35.

Với cách biểu diễn như vậy ở đỉnh có chấm đậm trên hình tương ứng với đáp ứng ra của hàm có giá trị 1 còn không có chấm ứng với giá trị 0. Ta có thể biểu diễn hàm bốn biến dựa trên hình 1.36.

Bằng cách chấm đậm vào các đỉnh của hình vẽ ứng với đáp ứng của hàm bốn biến là 1. Hình vẽ đó cũng là dạng không gian bốn



Hình 1.35. Dạng hình học hàm Boole cụ thể.



Hình 1.36. Hàm Boole trong không gian 4 chiều.

chiều của hàm lôgic bốn biến. Tương tự bằng cách vẽ như vậy ta có thể biểu diễn không gian n chiều của .hàm lôgic.

§1.6. LÝ THUYẾT NHÓM, VÀNH VÀ TRƯỜNG

Trong tổ chức toán học cũng như trong tổ chức của cấu trúc đại số, không phải lúc nào ta cũng sử dụng hết các phép toán trong tổ chức của nó để thể hiện một quan hệ tương tác tính toán, mà có lúc ta chỉ dùng một hoặc vài phép toán để thể hiện. Từ đó ta có tổ chức toán học nhỏ hơn - cấu trúc đại số nhỏ hơn và có tên gọi cho nó tương ứng, điều đó được thể hiện trong phần này. Đồng thời để khi ta tự định nghĩa lấy phép toán (sẽ nói cách thực hiện ở phần tiết 1.10 tiếp sau) thì biết được kết quả đó thuộc vùng nào của quan hệ toán học, và cũng biết được tổ chức toán được tạo ra đó có là một quan hệ toán học trọn vẹn hay không - một hệ hàm đầy đủ hay không.

1.6.1. NHÓM

Nhóm là một cấu trúc đại số nằm trong hệ thống đại số nhỏ nhất được định nghĩa như sau.

Định nghĩa 1.6.1. Cho một hệ thống đại số (A, *), trong đó A là một tập các con số, còn * là một phép toán hai ngôi được định nghĩa trên A thì tạo ra một hệ thống đại số nhỏ nhất.

Ví dụ : cho N = (0,1,2,3,4,5,...,n) là tập các số nguyên dương.

Ta có các cấu trúc đại số nhỏ nhất đó là : (N, .), (N, +) ...

Định nghĩa 1.6.2. Trong hệ thống đại số (A, *), nếu tồn tại một phần tử e thuộc A (e ∈ A), thì e được gọi là phần tử đơn vị nếu và chỉ nếu đối với một x bất kỳ trong A ta có quan hệ :

$$e * x = x * e = x$$

Ví dụ : Cho N = (0, 1, 2, 3, 4, 5, ..., n)

Ta có một hệ thống đại số là (N, .) thì phần tử đơn vị của nó lă (e = 1) vì thỏa mãn :

$$1 . x = x . 1 = x$$

***Định lý** 1.6.1.* Nếu tồn tại một phần tử đơn vị e đối với một phép toán hai ngôi trên A thì phần tử đó là duy nhất.

Định lý này có thể tự chứng minh.

Định nghĩa 1.6.3. Cho (A, *) là một hệ thống đại số. Trong đó * là phép toán hai ngôi trên A. Hệ thống đại số (A, *) được gọi là nửa nhóm nếu thỏa mãn :

- * là một phép toán đóng trong A

- Phép toán * có tính chất kết hợp, nghĩa là :

$$\forall\ a,b,c \in A \text{ ta có :}$$

$$(a * b) * c = a * (b * c)$$

Ví dụ : Cho A = (2,4,6,8,...,n) là tập số chẵn.

Thì hệ thống đại số (A, +) gọi là nửa nhóm.

Còn hệ thống đại số (A, ÷) không được gọi là nửa nhóm.

Vì phép toán chia ÷ không có tính chất kết hợp.

Định nghĩa 1.6.4. Nửa nhóm mà tồn tại phần tử đơn vị thì được gọi là một vị nhóm.

Ta dễ dàng tự tìm lấy ví dụ về vị nhóm.

Định nghĩa 1.6.5. Cho tập A và một phép toán hai ngôi * trên A với phần tử đơn vị là e. Phần tử x'∈A được gọi là phần tử nghịch đảo của x∈A, nếu chúng có quan hệ sau :

$$x * x' = x' * x = e$$

Định lý 1.6.2. Trong một vị nhóm (A, *, e) nếu tồn tại một phần tử nghịch đảo thì phần tử đó là duy nhất.

Thật vậy, giả sử x có hai phần tử nghịch đảo là x' và x'', theo định nghĩa ta có thể viết :

$$x' = x' * e = x' * (x * x'') = (x' * x) * x'' = e * x'' = x''$$

Định nghĩa 1.6.6. Một vị nhóm (A, *, e) mà trong đó mỗi phần tử của nó có một phần tử nghịch đảo tương ứng thì gọi là một nhóm.

Ví dụ : Cho N là một tập tất cả các số nguyên dương và + là phép toán cộng thông thường các số nguyên. Thì rõ ràng hệ thống đại số (N, +) là một nhóm với phần tử đơn vị là 0 và phần tử nghịch đảo của phần tử x là -x (vì x + (-x) = 0 = e).

Định nghĩa 1.6.7. Nhóm (A, *) gọi là một nhóm giao hoán nếu phép toán * có tính giao hoán.

Ví dụ : Nhóm (A, +) là nhóm giao hoán vì phép toán cộng có tính giao hoán.

Nhóm (A, ÷) không phải là nhóm giao hoán vì phép toán chia không có tính giao hoán.

Định nghĩa 1.6.8. Cho (A, *) là một nhóm, B là một tập con của A. Hệ thống đại số (B, *) gọi là một nhóm con của A nếu bản thân (B, *) cũng là một nhóm.

Sau đây ta sẽ xét đến các hệ thống đại số với hai phép toán hai ngôi, nghĩa là đi đến các khái nhiệm về vành, miền nguyên và trường trong phần tiếp sau đây.

1.6.2. VÀNH

Định nghĩa 1.6.9. Hệ thống đại số (A, +, •) được gọi là một vành V nếu thỏa mãn các điều kiện sau :

- (A, +, 0) là một nhóm
- (A, • , 1) là một nửa nhóm.
- Phép toán nhân có tính phân bố đối với phép cộng tức là :

$$x . (y + z) = x . y + x . z \qquad \forall x, y, z \in A$$

Vành V gọi là vành giao hoán nếu phép nhân có tính giao hoán.

Định lý 1.6.3. Cho x và y là hai phần tử thuộc vành V, hai quan hệ sau đây luôn đúng :

- $x . 0 = 0 . x = 0$
- $x . (-y) = (-x) . y = -x . y$
- $(-x) . (-y) = x . y$

Định nghĩa 1.6.10. Vành (B, +, ., 0, 1) có ít nhất hai phần tử được gọi là vành Boole nếu phép toán nhân có tính chất luỹ đẳng, nghĩa là : $x . x = x \quad x \in B$

1.6.3. MIỀN NGUYÊN VÀ TRƯỜNG

Định nghĩa 1.6.11. Hệ thống đại số với hai phép toán hai ngôi (A, +, •) được gọi là một miền nguyên nếu thoả mãn các điều kiện sau đây :

- (A, +) là một nhóm

- Phép toán nhân có tính giao hoán. Đồng thời nếu $c \neq 0$ mà có $c . a = c . b$ thì có $a = b$, với 0 là phần tử đơn vị.

- Phép toán nhân có tính phân bố đối với phép toán cộng.

Ví dụ : Cho A = (tập hợp các số nguyên dương), còn cộng (+) và nhân (•) là các phép toán thông thường. Thì hệ thống đại số (A, +, •) là một miền nguyên.

Định nghĩa 1.6.12. Hệ thống đại số với hai phép toán hai ngôi (A, +, •) được gọi là một trường nếu thỏa mãn :

- (A, +, •) là một nhóm.

- (A, +, 0, •) là một nhóm.

- Phép toán (•) có tính phân bố đối với phép toán cộng (+).

1.6.4. HỆ HÀM ĐẦY ĐỦ

Bất kỳ một hệ thống đại số nào (A, N) cũng cần có một tập A các phép toán, nó bao gồm rất nhiều các phép toán khác nhau với mục đích là từ các phép toán đó có thể biểu diễn được mọi quan hệ toán học (hàm số) có thể có thuộc cấu trúc đại số đó. Khi đó ta có một quan hệ toán học hoàn chỉnh trọn vẹn, như vậy ta đã có một hệ hàm đầy đủ.

Định nghĩa 1.6.13. Một cấu trúc đại số (A, N) với N là tập các con số còn A là tập các phép toán, mà với các phép toán trong A có thể biểu diễn được mọi quan hệ toán học có thể có của đại số đó thì được gọi là một hệ hàm đầy đủ.

Ví dụ : Cấu trúc đại số $\big((+, -, \bullet, \div), (0,1,2,3, ..., n) - (A, N)\big)$ đó là một hệ hàm đầy đủ, vì với bốn phép toán đó ta có thể biểu diễn mọi quan hệ toán học của đại số đó.

Cấu trúc đại số $\big((+, -, \div), (0,1,2,3, ..., n)\big)$ cũng là một hệ hàm đầy đủ vì phép toán nhân có thể được thay bằng phép toán cộng.

Cấu trúc đại số $\big((-, \cdot, \div), (0,1,2,3, \ldots, n)\big)$ đây không phải là một hệ hàm đầy đủ vì thiếu phép toán cộng nên không thể biểu diễn được hết các quan hệ toán học cần thiết.

Tương tự trong đại số lôgic cũng có thể lấy các ví dụ 1.6.2 sau :

Cấu trúc đại số $\big((+, \cdot, -), (0,1), B)\big)$ đó là một hệ hàm đầy đủ vì biểu diễn được mọi quan hệ toán học cần thiết.

Cấu trúc đại số $\big((+, \cdot), (0,1), B)\big)$ là hệ hàm không đầy đủ vì thiếu phép toán phủ định nên không thể biểu diễn được hết các quan hệ toán học đề ra.

Trong phần này ta đã đưa ra một loạt các định nghĩa để nhận biết và phân biệt các tổ chức đại số nói chung, dựa vào đó ta có thể hiểu biết các hệ thống đại số mới nào đó.

§1.7. ĐẠI SỐ CHUYỂN MẠCH VÉCTƠ VÀ HÀM CHUYỂN MẠCH VÉCTƠ

Đại số Boole hai phần tử (0,1) còn được gọi là hàm chuyển mạch véctơ, vì hai giá trị 0 và 1 của đại số tương ứng với hoạt động của công tắc chuyển mạch "đóng" và "mở", đồng thời nó cũng được ứng dụng trong các hệ thống chuyển mạch điện tử hiện nay. Vậy khi nghiên cứu tới đại số chuyển mạch véctơ chính là xét tới đại số lôgic - đại số Boole hai trị, đồng thời nói tới ứng dụng của chuyển mạch trong hệ thống. Ta biết các ánh xạ Boole từ đại số chuyển mạch véctơ vào chính bản thân nó gọi là các hàm chuyển mạch véctơ, đó cũng chính là mạch điện của chức năng. Hàm chuyển mạch ngoài các phép toán cơ bản của nó là OR, AND và NOT, thêm NOR, NAND, ta còn cho thêm vào các phép toán mới theo yêu cầu kỹ thuật đó là phép toán bù tổng quát X^p và phép toán quay $\tilde{X}$ tạo thành đại số chuyển mạch hoàn hảo, cho phép ta tạo ra những linh kiện điện tử điều khiển theo chương trình. Đó là ứng dụng quan trọng của đại số chuyển mạch véctơ, cũng chính là sự kết hợp giữa toán lôgic với công nghệ điện tử tạo ra CPU, MP... PROM bộ nhớ điều khiển theo chương trình. Để hiểu được điều đó trước hết ta tìm hiểu các định nghĩa và định lý của đại số chuyển mạch.

1.7.1. ĐẠI SỐ CHUYỂN MẠCH

Cũng như đại số Boole hai trị đại số chuyển mạch có dạng (+, ., −, 0, 1, B), trong đó các phép toán hai ngôi (+) được thay bằng mạch OR, phép toán nhân (.) được thay bằng mạch AND, còn phép toán một ngôi phủ định được thay bằng mạch NOT, đồng thời chúng thoả mãn các tiên đề từ $D_1 \div D_6$ của đại số Boole. Đại số chuyển mạch có một số các định lý cơ bản như sau.

Định lý 1.7.1. Cho $X = (x_1, x_2, \ldots, x_n)$ là các biến nhị phân thuộc đại số Boole, chúng ta có :

$$\overline{(x_1 + x_2 + \ldots + x_n)} = \bar{x}_1 . \bar{x}_2 . \ldots . \bar{x}_n$$

$$\overline{(x_1 . x_2 . \ldots . x_n)} = \bar{x}_1 + \bar{x}_2 + \ldots + \bar{x}_n$$

Định lý này được gọi là định lý De Morgan tổng quát, nó có thể chứng minh một cách dễ dàng bằng phương pháp truy chứng.

Định lý 1.7.2. Định lý Shannon cho biết kết quả của phép toán phủ định một hàm chuyển hạch như sau :

$$\overline{\left(f(x_1, x_2, \ldots, x_n), +, \cdot\right)} = f(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n, \cdot, +)$$

Nói cách khác khi phủ định một hàm chuyển mạch kết quả nhận được là phủ định của các biến số, còn phép toán nhân (AND) chuyển thành phép toán cộng (OR) và ngược lại.

Ví dụ : Cho $f(x_1, x_2, x_3) = (\bar{x}_1 + x_2) \cdot x_3 + \bar{x}_1 \cdot x_2 \cdot \bar{x}_3$

Vậy $\overline{f(x_1, x_2, x_3)} = (x_1 \cdot \bar{x}_2 + \bar{x}_3) \cdot (x_1 + \bar{x}_2 + x_3)$

Kết quả đó ta có thể dễ dàng nhận được bằng cách biến đổi toán học dựa vào định lý De Morgan và các tiên đề toán lôgic.

Định lý 1.7.3. Một hàm chuyển mạch bất kỳ có thể được biểu diễn dưới dạng tuyển chính quy :

$$f(x_1, x_2, \ldots, x_n) = \sum_{(00\ldots0)}^{(11\ldots1)} (x_1^{e_1} \cdot x_2^{e_2} \ldots x_n^{e_n})$$

$\forall (e_1, e_2, \ldots, e_n)$ nếu có kết quả $f(e_1, e_2, \ldots, e_n) = 1$

hoặc dưới dạng hội chính quy :

$$f(x_1, x_2, \ldots, x_n) = \prod_{(00..0)}^{(11\ldots1)} (\bar{x}_1^{e_1} + \bar{x}_2^{e_2} + \ldots + \bar{x}_3^{e_3})$$

$\forall e_1, e_2, \ldots, e_n)$ nếu có kết quả $f(e_1, e_2, \ldots, e_n) = 0$

Định lý này rất dễ hiểu vì đối với dạng tuyển chuẩn tắc thì giá trị của $f(e_1, e_2, \ldots, e_n) = 1$ làm cho tích số các biến tương ứng của nó tồn tại, còn đối với dạng hội chuẩn tắc thì giá trị của $f(e_1, e_2, \ldots, e_n) = 0$ làm cho tổng các biến số tương ứng với nó tồn tại trong dạng hội.

1.7.2. PHÉP TOÁN BÙ TỔNG QUÁT VÀ PHÉP TOÁN QUAY

Trong cuộc sống con người sử dụng nhiều phép toán khác nhau theo các nhu cầu khác nhau của toán học, và chúng có các nguồn gốc khác nhau. Ở đây, từ những nhu cầu của thực tế kỹ thuật ta có được những phép toán mới, đó là phép toán bù tổng quát (X^P) và phép toán quay ($\tilde{X}$). Những phép toán mới này kết hợp với các phép toán khác của đại số chuyển mạch tạo nên một hệ thống toán học tuyệt vời làm cơ sở toán chủ yếu cho phép chế ra các IC cỡ lớn và quản lý theo chương trình, chúng được định nghĩa như sau.

1.7.2.1. Phép toán bù tổng quát (A^P)

Định nghĩa 1.7.1. Cho $X = (x_1, x_2, \ldots, x_n)$ là biến thuộc đại số boole, phép toán bù tổng quát (X^P) của X theo điều kiện p được thực hiện bằng cách bù các giá trị thành phần của X theo các giá trị thành phần tương ứng của p, với quy định x_i ($1 \leqslant i \leqslant n$)

sẽ được bù nếu $p_i = 0$ và sẽ không được bù nếu $p_i = 1$, ở đây p_i là thành phần thứ i của p.

Ví dụ : Cho trước p = 101 (ý định bù thường được biết trước)

nếu biết X = 011 → $X^p = 011^{101} = 001$ (vì bù phần tử thứ 2)

X = 100 → $X^p = 100^{101} = 110$ (đó là kết quả tính toán)

Phép toán bù tổng quát là phép toán một ngôi, nếu biết mọi giá trị của X ta sẽ có toàn bộ kết quả tính toán của nó theo p trên bảng ở hình 1.37.

Ta thấy nếu có giá trị của X và một giá trị cho trước của p thì phép toán X^p cho ta giá trị là một con số nhị phân, còn p thường là một nhiệm vụ kỹ thuật phủ định tổng quát đã biết trước.

Nếu cho trước p = (0,0,...,0) thì được kết quả của bù toàn phần (bù toàn bộ các giá trị của X), nói cách khác bù toàn phần là trường hợp riêng của phép toán bù tổng quát.

Cho p = 101

X	X^p
000	010
001	011
010	000
011	001
100	110
101	111
110	100
111	101

Hình 1.37. Phép toán bù tổng hợp.

1.7.2.2. Phép toán quay phải và quay trái

Định nghĩa 1.7.2. Cho $X = (x_1, x_2, ..., x_n)$ là véctơ biến của đại số Boole hai trị. Phép toán quay phải ($\overrightarrow{X}$) và phép toán quay trái ($\overleftarrow{X}$) của X được định nghĩa như sau :

$$\overrightarrow{X} = (x_n, x_1, x_2, \ldots, x_{n-1})$$

$$\overleftarrow{X} = (x_2, x_3, \ldots, x_n, x_1)$$

Ví dụ : Cho X = 110 phép toán quay phải và quay trái của X được tính như sau.

$$\overrightarrow{X} = 011 \text{ và } \overleftarrow{X} = 101$$

Nếu cho biết các giá trị của X thì kết quả của phép toán quay phải và quay trái của X nhận được ở bảng trên hình 1.38 sau.

Phép toán "quay phải" và phép toán "quay trái" chúng là phép toán một ngôi, cho trước một giá trị của X ta sẽ tính được kết quả quay của nó, nhận thấy kết quả cũng cho giá trị như mọi phép toán khác. Phép toán quay phải ($\overrightarrow{X}$) và quay trái ($\overleftarrow{X}$) không chỉ được dùng để tính toán mà còn kết hợp với phép toán bù tổng quát (X^p) tạo ra khả năng quản lý ma trận điểm trong kỹ thuật.

X	$\overrightarrow{X}$	$\overleftarrow{X}$
000	000	000
001	100	010
010	001	100
011	101	110
100	010	001
101	110	011
110	011	101
111	111	111

Hình 1.38. Phép toán quay một ngôi.

Định nghĩa 1.7.3. Cho X (x_1, x_2, ..., x_n) là biến của đại số lôgic, nếu thực hiện quay X nhiều lần (n lần) và không chú ý đến chiều quay được viết như sau :

$$\overset{(n)}{\widetilde{X}} = (\,.\,.((X)).\,.\,) \quad \left.\begin{matrix}\widetilde{\cdot}\\ \cdot\\ \widetilde{\cdot}\end{matrix}\right\} \text{ n lần}$$

Toán tử quay n lần theo định nghĩa trên là quay cùng một chiều quay tất cả theo chiều phải hoặc tất cả theo chiều trái.

Từ khái niệm của phép toán bù tổng quát và phép toán quay, và kết hợp với các phép toán khác của đại số lôgic chúng ta đã tạo ra một cấu trúc đầy đủ và hoàn chỉnh của đại số chuyển mạch véctơ, nó được định nghĩa như sau.

Định nghĩa 1.7.4. Một cấu trúc đại số $(+, ., \neg, \widetilde{X}^p, X, 0, 1, B)$ trong đó có hai phép toán hai ngôi là cộng (+) và nhân (.), và ba phép toán một ngôi là phủ định ($\neg$), bù tổng quát (X^p) và phép toán quay ($\widetilde{X}$) với giá trị lớn nhất là 1, giá trị bé nhất là 0 còn B là tập các biến được gọi là đại số chuyển mạch véctơ.

1.7.2.3. Các tính chất và tiên đề các phép toán

Các tính chất (tiên đề) của phép toán bù tổng quát và phép toán quay như sau :

- $X^{\Phi} = \overline{X}$, $X^{I} = X$ ở đây $\Phi = (0, 0, .., 0)$, $I = (1, 1, .., 1)$
- $p^p = 1$, $\overline{p^p} = p^{\bar{p}} = \bar{p}^p = 0$
- $p^p = X \oplus \bar{p}$
- $p_1 = \bar{p}_2$ nếu và chỉ nếu $X^{p_1} = X^{\bar{p}_2}$
- $X^{p^p} = (X^p)^p = X^{(p^p)} = X$
- $X^{p^{2n-1}} = X^p$
- $p^{\dot{p}^n} = 1$ nếu n lẻ, $p^{p^n} = p$ nếu n chẵn
- $\overline{(X_1^{p_1} + X_2^{p_2})} = \overline{X_1^{p_1}} . \overline{X_2^{p_2}} = X_1^{\bar{p}_1} . X_2^{\bar{p}_2} = \bar{X}_1^{p_1} . \bar{X}_2^{p_2}$

  $\overline{(X_1^{p_1} . X_2^{p_2})} = \overline{X_1^{p_1}} + \overline{X_2^{p_2}} = X_1^{\bar{p}_1} + X_2^{\bar{p}_2} = \bar{X}_1^{p_1} + \bar{X}_2^{p_2}$
- $(X_1 + X_2)^p = (\bar{X}_1 . \bar{X}_2)^{\bar{p}}$
- $(X_1 . X_2)^p = (\bar{X}_1 + \bar{X}_2)^{\bar{p}}$
- $\left(X_1^{p_1} + X_2^{p_2}\right)^{p_3} = \left(X_1^{\bar{p}_1} . X_2^{\bar{p}_2}\right)^{\bar{p}_3}$
- $\left(X_1^{p_1} . X_2^{p_2}\right)^{p_3} = \left(X_1^{\bar{p}_1} + X_2^{\bar{p}_2}\right)^{\bar{p}_3}$

$$\overset{(n)}{\tilde{X}} = X$$

$$- \widetilde{(X_1 . X_2)} = \tilde{X}_1 . \tilde{X}_2$$

$$\widetilde{(X_1 + X_2)} = \tilde{X}_1 + \tilde{X}_2$$

$$- \tilde{X}^{\tilde{p}} = \widetilde{X^{p}}$$

$$- (X^{p}) . \widetilde{(X^{p})} \ldots \overset{(n-1)}{\widetilde{(X^{p})}} = 1 \text{ nếu } X = p, \text{ bằng } 0 \text{ nếu } X \neq p$$

$$(X^{\bar{p}}) + \widetilde{(X^{\bar{p}})} + \ldots + \overset{(n-1)}{\widetilde{(X^{\bar{p}})}} = 0 \text{ nếu } X = p, \text{ bằng } 1 \text{ nếu } X \neq p$$

1.7.3. HÀM CHUYỂN MẠCH VÉCTƠ

Một cấu trúc đại số gồm các phép toán mới được đưa vào kèm theo các tiên đề của chúng cho phép ta tạo ra hàm số của đại số mới. Cũng như các đại số nói chung hàm chuyển mạch véctơ được tạo ra dựa trên đại số chuyển mạch véctơ và có cấu trúc tổng quát theo định nghĩa sau.

Định nghĩa 1.7.5. Cho $(X_1, X_2, \ldots, X_n) = X$ là các biến véctơ, hàm chuyển mạch véctơ được định nghĩa như sau :

– Cho A là hằng số của hàm chuyển mạch. Hàm hằng có cấu tạo :

$$F(X) = A \text{ (hàm hằng)}$$

Cho X_i là tập các biến, hàm chiếu được định nghĩa :

$$F(X) = X_i \text{ trong đó } (1 \leq 1 \leq n) \text{ (hàm chiếu)}$$

– Nếu $F(X_1, X_2, \ldots, X_n)$ là hàm chuyển mạch thì $F^{p}(X_1^{p_1}, X_2^{p_2}, \ldots, X_n^{p_n})$ cũng là hàm chuyển mạch véctơ đối với các giá trị bất kỳ của p_i. n1 n2 nn

– Nếu $F(X_1, X_2, \ldots, X_n)$ là hàm chuyển mạch véctơ thì $F(\tilde{X}_1, \tilde{X}_2, \ldots, \tilde{X}_n)$ cũng là hàm chuyển mạch véctơ.

– Nếu $F_1(X)$ và $F_2(X)$ là hàm chuyển mạch véctơ thì $(F_1 + F_2)$ và $(F_1 . F_2)$ cũng là hàm chuyển mạch véctơ.

– Một hàm số được tạo ra bằng cách áp dụng một số hữu hạn lần các quy luật kể trên thì cũng là hàm chuyển mạch véctơ.

Từ cấu trúc của hàm chuyển mạch ta có một số định lý và các định luật toán học tổng quát áp dụng cho nó, các quan hệ toán học tổng quát đó cũng tương ứng và đúng với các quan hệ của đại số Boole thông thường như sau.

Định lý 1.7.4. Định lý De Morgan tổng quát :

$$(X_1^{p_1} . X_2^{p_2} . \ldots . X_n^{p_n})^{p} = (X_1^{\bar{p}_1} + X_2^{\bar{p}_2} + \ldots + X_n^{\bar{p}_n})^{\bar{p}}$$

$$(X_1^{p_1} + X_2^{p_2} + \ldots + X_n^{p_n}) = (X_1^{\bar{p}_1} . X_2^{\bar{p}_2} . \ldots . X_n^{\bar{p}_n})^{\bar{p}}$$

Định lý 1.7.5. Định lý Shannon tổng quát.

$$F^{p}(X_1^{p1}, X_2^{p2}, \ldots, X_n^{pn} + , \cdot) = \overline{F^{p}}(X_1^{\bar{p}1}, X_2^{\bar{p}2}, \ldots, X_n^{\bar{p}n}, \cdot , +)$$

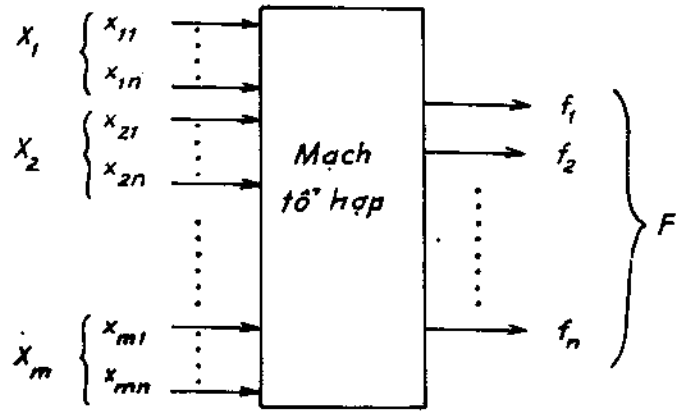
Cũng như hàm Boole hàm chuyển mạch véctơ cũng có dạng chính quy dạng tổng quát của nó. Ở đây hàm chuyển mạch khi có thêm các phép toán mới phép toán bù tổng quát X^p và phép toán quay $\tilde{X}$ đã tạo thêm cho nó có cấu trúc rất lớn, gồm tập các tín hiệu vào

$X = (X_1, X_2, \ldots, X_m)$ trong đó $X_i = (x_1, x_2, \ldots, x_n)$

với $(1 \leqslant i \leqslant n)$

và tập các đáp ứng ra

$F = (f_1, f_2, \ldots, f_n)$. Một tín hiệu vào cụ thể x_{ij} được xác định dựa vào $(1 \leqslant i \leqslant m)$ và $(1 \leqslant j \leqslant n)$. Nếu mỗi x_{ij} lấy giá trị 0 hoặc 1 thì sẽ có 2^{mn} tổ hợp tín hiệu vào khác nhau, cũng sẽ có các đáp ứng ra f_k tương ứng với $(1 \leqslant k \leqslant n)$. Mạch điện tổng quát của đại số chuyển mạch véctơ được biểu diễn trên hình 1.39.



Hình 1.39. Mạch điện của đại số chuyển mạch.

Một chức năng kỹ thuật hay là một hàm lôgic có thể được thực hiện bằng đại số chuyển mạch, và mạch lôgic thực hiện các hàm đó được gọi là mạch tổ hợp. Các chức năng lôgic có thể biểu diễn trên bảng thật hay còn gọi là bảng chân lý thể hiện trên hình 1.40.

X_1	X_2	X_m	F
$\overbrace{x_{11}\ x_{12} \ldots x_{1n}}$	$\overbrace{x_{21}\ x_{22} \ldots x_{2n}}$	$\overbrace{x_{m1}\ x_{m2} \ldots x_{mn}}$	$\overbrace{f_1 \quad f_2 \quad \ldots \quad f_n}$
0 0 ... 0	0 0 ... 0 ... 0 0 ... 0		f_{10} f_{20} ... f_{n0}
0 0 ... 0	0 0 ... 0 ... 0 0 ... 1		f_{11} f_{21} ... f_{n1}
...	... ...		... ...
1 1 ... 1	1 1 ... 1 ... 1 1 ... 1		$f_{12}mn$ $f_{22}mn$ $f_{n2}mn$

Hình 1.40. Bảng tổng quát chức năng của hàm chuyển mạch.

Nhận thấy khi có thêm hai phép toán mới X^p và $\widetilde{X}$ cấu trúc của đại số chuyển mạch lớn hơn lên rất nhiều, và đó là cấu trúc đại số mạch nhất mà con gười có được cho tới nay, vì trong cấu trúc của nó có năm phép toán hoàn toàn độc lập với nhau và không có phép toán nào phụ thuộc hay suy diễn từ phép toán kia. Điều đó cũng có nghĩa là ta có thể thực hiện được mọi chức năng – mọi hàm lôgic từ bộ đại số mạnh mẽ này. Dạng tổng quát của đại số chuyển mạch được thể hiện trong định lý sau.

Định lý 1.7.5. Cho F (X) là hàm chuyển mạch véctơ, trong đó X = (X_1, X_2, . .., X_m) là tập của m biến nhị phân với X_i = (X_1, X_2, ..., X_n) và (1 ≤ 1 ≤ m) được biểu diễn tổng quát như sau :

1.7.3.1. Dạng tuyển chính quy của hàm F(X)

$$F(X) = \sum_{(00\ldots0)}^{(11\ldots1)} F(p_1, p_2, \ldots, p_m) \cdot \left(X_1^{p_1} \cdot (\widetilde{X_1^{p_1}}) \ldots \overbrace{(X_1^{p_1})}^{(n-1)}\right) \ldots \left((X_m^{p_m}) \cdot (\widetilde{X_m^{p_m}}) \ldots \overbrace{(X_m^{p_m})}^{(n-1)}\right)$$

1.7.3.2. Dạng hội chính quy của hàm F(X)

$$F(X) = \prod_{(00\ldots0)}^{(11\ldots1)} \left(F(p_1, p_2, \ldots, p_m) + \left(X_1^{p_1} + (\widetilde{X_1^{p_1}}) + \ldots + \overbrace{(X_1^{p_1})}^{(n-1)}\right) + \ldots + \left(X_m^{p_m} + (\widetilde{X_m^{p_m}}) + \ldots + \overbrace{(X_m^{p_m})}^{(n-1)}\right)\right.$$

Ví dụ : Cho hàm chuyển mạch véctơ biểu diễn trên bảng chức năng cụ thể hình 1.41.

X_1	X_2	F(X)		
		f_1	f_2	f_3
000	001	1	0	1
001	011	0	1	0
100	101	0	0	1
101	110	1	1	1
. . . . .		. . . . .		
Các giá trị khác		0	0	0

Hình 1.41. Minh họa cụ thể một chức năng.

Hàm đó có thể được biểu diễn dưới dạng tuyến chính quy như sau. Do biểu diễn ở dạng tuyển chính quy cho nên ta chỉ chú ý tới các giá trị 1 của hàm số, còn các giá trị khác hàm có giá trị 0 nên ta không cần quan tâm, vì vậy cũng không cần viết hết giá trị các tổ hợp biến làm cho bảng chức năng quá dài. Dạng cụ thể được biểu diễn như sau :

$$\begin{aligned} F(X_1, X_2) = {} & (101)\, X_1^{000} (\widetilde{X_1^{000}})\, (\widetilde{\widetilde{X_1^{000}}})\, X_2^{001}\, (\widetilde{X_2^{001}})\, (\widetilde{\widetilde{X_2^{001}}}) + \\ & + (010)\, X_1^{001}\, (\widetilde{X_1^{001}})\, (\widetilde{\widetilde{X_1^{001}}})\, X_2^{011}\, (\widetilde{X_2^{011}})\, (\widetilde{\widetilde{X_2^{011}}}) + \\ & + (010)\, X_1^{001}\, (\widetilde{X_1^{100}})\, (\widetilde{\widetilde{X_1^{100}}})\, X_2^{101}\, (\widetilde{X_2^{101}})\, (\widetilde{\widetilde{X_2^{101}}}) + \\ & + (111)\, X_1^{101}\, (\widetilde{X_1^{101}})\, (\widetilde{\widetilde{X_1^{100}}})\, X_2^{101}\, (\widetilde{X_2^{101}})\, (\widetilde{\widetilde{X_2^{101}}}) \end{aligned}$$

Qua minh họa biểu diễn cụ thể một chức năng kỹ thuật bất kỳ bằng đại số chuyển mạch véctơ ta thấy được khả năng của nó dùng để thể hiện một nhiệm vụ kỹ thuật nào đó, và về lý thuyết toán học đại số chuyển mạch véctơ khi có phương trình toán học tổng quát của nó thì có nghĩa là có thể biểu diễn được mọi phương trình toán tương ứng với mọi nhiệm vụ kỹ thuật (điều mà các quan hệ toán học khác khó thực hiện được). Bởi vậy nếu ta lắp được một mạch điện mà mạch điện đó thể hiện được phương trình tổng quát của đại số chuyển mạch véctơ thì cũng có nghĩa là ta có thể thực hiện được mọi nhiệm vụ đó một cách cụ thể bằng mạch điện, và điều đó cũng là ứng dụng quan trọng của đại số chuyển mạch véctơ.

1.7.4. ỨNG DỤNG CỦA ĐẠI SỐ CHUYỂN MẠCH VÉCTƠ

Ứng dụng của đại số chuyển mạch trong việc tổng hợp mạch lôgic nhờ sự ra đời và phát triển nhanh chóng của kỹ thuật vi điện tử, nhất là trong việc chế tạo ra các mạch vi điện tử cỡ lớn LSI và cực lớn ELSI, cho phép thay đổi cơ bản nguyên lý thiết kế hệ thống số, đồng thời cho phép thay đổi chức năng của hệ thống số linh hoạt như toán học và đạt được điều đó nhờ lập trình. Người ta cũng dựa theo những thuật toán đã được tiêu chuẩn hóa để chế ra những mạch tổ hợp có sẵn cỡ nhỏ SSI, cỡ trung bình MSI, cỡ lớn LSI...

Để hiểu được điều đó ta xuất phát từ phương trình tổng quát của đại số chuyển mạch, ví dụ từ dạng tuyển chuẩn tắc toàn phần.

$$F(X) = \sum_{(00..0)}^{(11..1)} \left(F(p_1, p_2, \ldots, p_m) \cdot (X_1^{p_1} \ldots \overbrace{(X_1^{p_1})}^{(n-1)}) \ldots \left(X_m^{p_m} \ldots \overbrace{(X_m^{p_m})}^{(n-1)} \right) \right)$$

Kết hợp với tiên đề : $X^p = X \oplus \bar{p}$

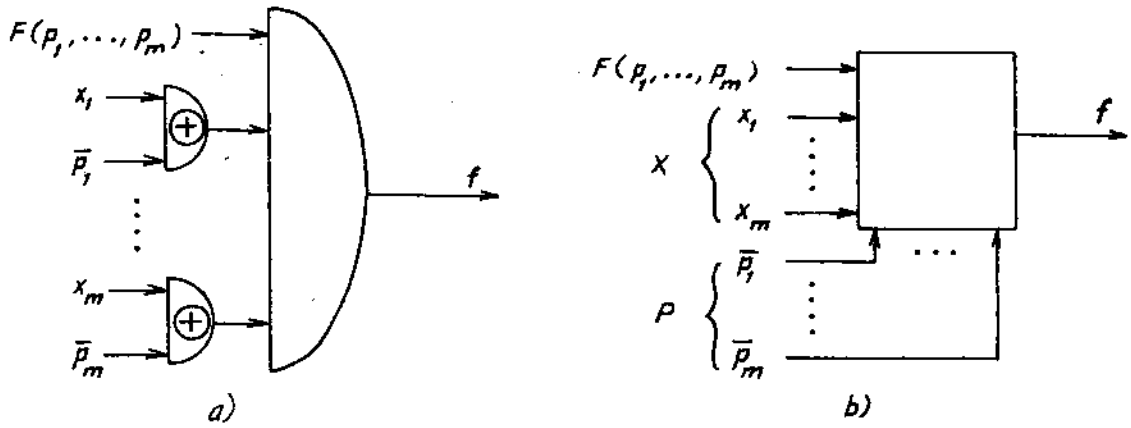
Ta có dạng tuyển chính quy của đại số chuyển mạch như sau :

$$F(X) = \sum_{00..0}^{11..1} \left(F(p_1, p_2, \ldots, p_m) \cdot \left((X_1 \oplus p_1) \ldots \overbrace{(X_1 \oplus p_1)}^{(n-1)} \right) \cdot \right.$$
$$\left. \cdot \left((X_m \oplus p_m) \ldots \overbrace{(X_m \oplus p_m)}^{(n-1)} \right) \right)$$

Với $X_i = (X_1, X_2, \ldots, X_n)$ trong đó $(1 \leq i \leq m)$.

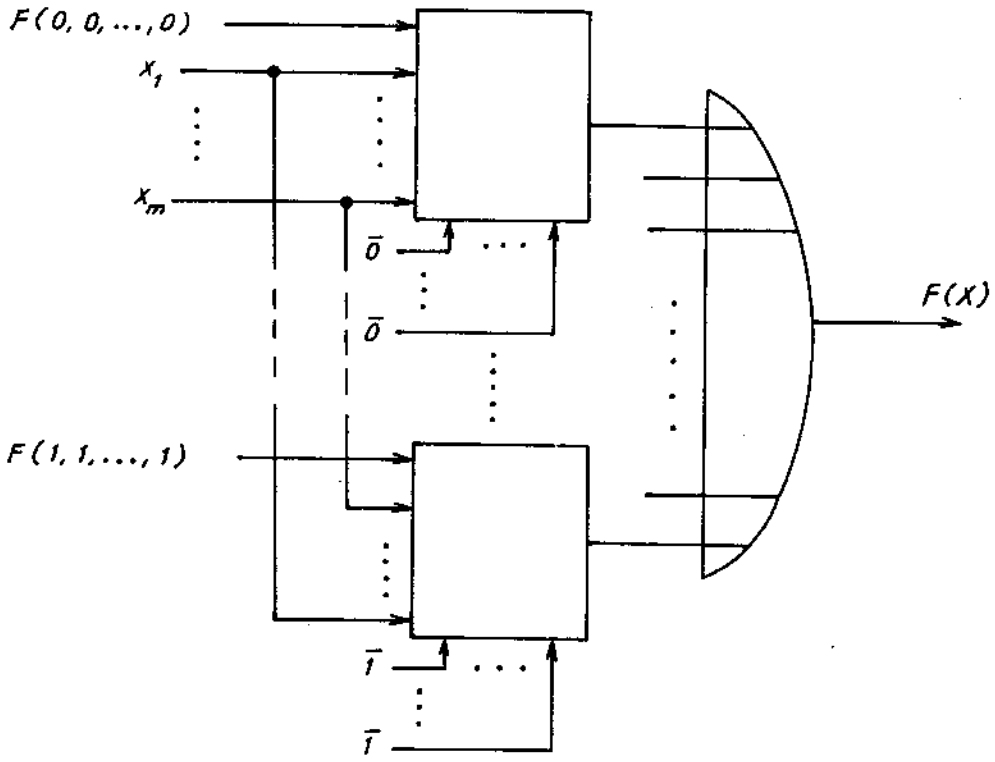
Đó là dạng tổng quát biểu diễn dưới dạng tổng của các tích, mỗi một tích số ta có thể thực hiện bằng sơ đồ vẽ trên hình 1.42,a và chuyển sơ đồ đó thành sơ đồ tương đương bằng cách tách X và p sang một bên được thể hiện trên hình 1.42,b.

Khi ta chuyển tín hiệu vào X (X_1, ..., X_m) sang một bên và tập tín hiệu p = (p_1,..., p_m) sang một bên, thì điều đó tương ứng với việc tạo ra "bus số liệu" và "bus địa chỉ" cho mạch điện, và mạch tích này làm việc hay không được làm việc hoàn toàn phụ thuộc vào tín hiệu điều khiển F (p_1, ..., p_m) lấy giá trị 0 hay 1, điều đó chính là các hệ thống số sau này. Để tạo ra mạch điện tương ứng với phương trình tổng quát vấn đề chỉ còn là tổng các mạch tích đó lại bằng mạch điện (OR) và mạch điện tổng quát cho phép thực hiện được mọi chức năng một cách linh hoạt hoàn toàn tương ứng với phương trình



Hình 1.42. Mạch điện một tích số.

toán học tổng quát được minh họa trên hình 1.43. Sơ đồ mạch điện tổng quát của đại số chuyển mạch véctơ chính là cơ sở để chế tạo ra những mạch IC cỡ lớn hoạt động linh hoạt theo chương trình như bộ nhớ cố định điều khiển theo chương trình PROM, bộ xử lý trung tâm CPU... Đó là những mạch điện cụ thể, hoạt động vạn năng, sử dụng linh hoạt và dễ dàng như toán học, đó cũng chính là ứng dụng rất quan trọng của đại số chuyển mạch véctơ trong kỹ thuật số.



Hình 1.43. Sơ đồ mạch điện tổng quát.

Dựa vào phương trình toán học tổng quát ta xây dựng được một mạch điện tổng quát, và mạch điện tổng quát đó có nguyên lý làm việc như sau : Tín hiệu vào X = (X_1, X_2, ..., X_n) được đưa vào liên tục tùy theo nhu cầu điều khiển hay xử lý, đồng thời do yêu cầu tổng của toán học chạy từ (00...0) tới (11...1), tức là tương ứng với việc quay quét lần lượt từ ô địa chỉ đầu tiên (00...0) ói ô địa chỉ cuối cùng (11...1), và tùy theo các giá trị của F (00...0) tới F (11...1) chúng lấy giá trị 0 hay 1 mà mạch tích tương ứng của chúng làm việc hay không làm việc, từ đó cho tín hiệu điều khiển tới đầu ra F(X). Việc thay đổi giá trị của F (p_1, p_2, ..., p_n) theo yêu cầu chương trình phần mềm cho trước do ta lập ra, vì vậy ở đầu ra F (X) sẽ cho ra tín hiệu điều khiển ứng với chương trình để thực hiện một nhiệm vụ kỹ thuật nào đó. Từ nguyên lý đó cho phép chế tạo ra một mạch điện có kết cấu cố định nhưng thực hiện được nhiều chức năng khác nhau, nhờ thay đổi chương trình điều khiển nó. Theo lý thuyết là có thể làm được vô tận chức năng, tùy thuộc vào giá trị của m, nhưng trong thực tế chế tạo thì m chỉ có một giá trị hữu hạn, cho nên mạch sẽ có một số hữu hạn chức năng theo độ lớn của bus tín hiệu của nó (VD : m = 8 là 8 Bus).

Tóm lại dựa vào phương trình tổng quát, là phương trình cho phép ta mô tả được mọi hàm lôgic - mọi chức năng kỹ thuật, ta lắp ráp ra được một mạch điện tổng quát, nên mạch điện đó có thể thực hiện được mọi chức năng kỹ thuật một cách cụ thể - chỉ cần một mạch điện đó là hàm được mọi chức năng. Từ điều đó người ta đã chế tạo ra các IC cỡ lớn, những mạch số có thể lập trình được như mạch CPU của máy vi tính, và cũng chỉ cần một mạch đó ta có thể làm được rất nhiều chức năng (việc) khác nhau mà không cần thay đổi linh kiện trong máy, điều này chúng ta ai cũng đã thấy rõ, và đó cũng chính là ứng dụng lớn nhất của đại số lôgic hay đại số chuyển mạch véctơ.

Khi có mạch điện tổng quát rồi thì vấn đề chỉ còn là làm thế nào thay đổi được chức năng của nó, làm điều này phải nhờ vào các chương trình phần mềm, mà theo nguyên lý làm việc của mạch điện tổng quát này ta thấy là chỉ cần thay đổi giá trị của hàm F(p_1, p_2 ... p_m) trong phương trình tổng quát là ta sẽ được mọi hàm số (chức năng) khác nhau. Qua đó ta hiểu được rằng chương trình phần mềm của máy tính mà chúng ta dùng, nó chỉ làm thay đổi lần lượt giá trị 0 hay 1 của hàm F(p_1, p_2, ..., p_m) trong mạch điện tổng quát mà thôi. Với mỗi một chương trình phần mềm cho trước, thì mạch điện tổng quát này sẽ đưa ra các tín hiệu điều khiển ở đầu ra F(X) thực hiện chức năng của chương trình phần mềm đó. Từ đó cho phép chúng ta hiểu rất rõ bản chất là tại sao máy tính (máy vi tính) không thay đổi gì về các linh kiện điện tử, mà chỉ cần thay đổi chương trình phần mềm là sẽ thực hiện được nhiều việc nhiều chức năng khác nhau. Cũng từ đó ta đã hiểu được chương trình phần mềm là gì và tác động của nó vào phần cứng như thế nào trong hệ thống số nói chung.

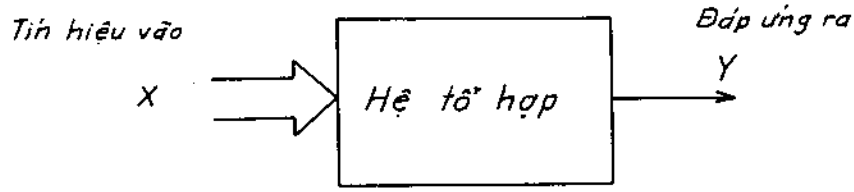
Để xây dựng nên mạch điện tổng quát, người ta dựa vào các mạch điện thực hiện các phép toán cơ bản của đại số Boole, chúng được gọi là các mạch điện lôgic cơ bản, đó là những mạch điện cụ thể, để hiểu được chúng ta nói kỹ trong phần tiếp theo.

1.7.5. CÁC MẠCH LÔGIC CƠ BẢN

Các mạch lôgic cơ bản hay còn được gọi là các phần tử lôgic cơ bản, dựa vào các mạch lôgic này ta có thể xây dựng nên hệ thống ngày nay. Một hệ thống số được xây

dựng toàn bằng các mạch lôgic thì hệ thống đó được gọi là "ôtômat không có nhớ" hay được gọi là "hệ lôgic tổ hợp".

Hệ lôgic tổ hợp là một hệ thống số – mạch lôgic mà tin tức (tín hiệu) ở đầu ra hay là đáp ứng ra của mạch chỉ phụ thuộc vào tin tức (tín hiệu vào) tại thời điểm đang xét. Tức là *đáp ứng ra của mạch phụ thuộc trực tiếp tín hiệu vào*. Nói cách khác đáp ứng ra (tin tức đầu ra) không phụ thuộc vào "lịch sử" của tin tức vào trước đó, nghĩa là nó làm việc và hoạt động "không có nhớ" không lưu giữ thông tin trước đó. Tổng quát của một hệ tổ hợp được mô tả trên hình 1.44.



Hình 1.44. Mạch điện tổng quát

Hệ tổ hợp thực hiện (hoạt động) theo phương trình tổng quát sau :

$$Y = f(X)$$

trong đó : $X = (x_1, x_2, ..., x_n)$: tập tín hiệu vào

$x_i \subseteq X$ $\qquad x_i \in \{0, 1\}$

. Y : là đáp ứng ra $\quad Y \in \{0, 1\}$

Hoạt động cụ thể của hệ tổ hợp được mô tả bằng các bảng chân lý, bảng thật hay bằng các hàm chuyển mạch đặc trưng cho quan hệ giữa tín hiệu vào (X) và đáp ứng ra (Y). Trong mạch tổ hợp (hệ tổ hợp) không chứa một thiết bị hay một phần tử ghi nhớ hay lưu giữ thông tin nào cả.

Định nghĩa 1.7.1.. Các phần tử lôgic cơ bản là các hệ lôgic tổ hợp nhỏ nhất, chúng có hai đầu vào và một đầu ra thực hiện các phép toán lôgic cơ bản là : AND (Và), OR (Hoặc), còn NOT (Phủ định) là mạch có một đầu vào và một đầu ra.

Để thực hiện được yêu cầu đó người ta đã xây dựng nên các mạch lôgic cơ bản như sau. Có nhiều phương pháp vật lý để thực hiện các hàm lôgic cơ bản như dùng cơ học là các công tắc, bằng cơ điện như của hệ thống rơle, bằng thủy lực, bằng ferit, bằng bán dẫn, bằng siêu dẫn v.v. Trong các máy tính điện tử hiện nay các phần tử lôgic cơ bản thường được thực hiện nhờ kỹ thuật bán dẫn, cụ thể thường dùng điot hay tranzitor. Sau đây sẽ đưa ra cấu tạo các mạch điện lôgic cơ bản dùng các linh kiện điện tử.

Trước hết muốn thực hiện được chức năng lôgic cũng như muốn hiểu được nguyên lý làm việc của mạch ta phải có quy ước kỹ thuật như sau :

giá trị lôgic 0 ứng với 0 von

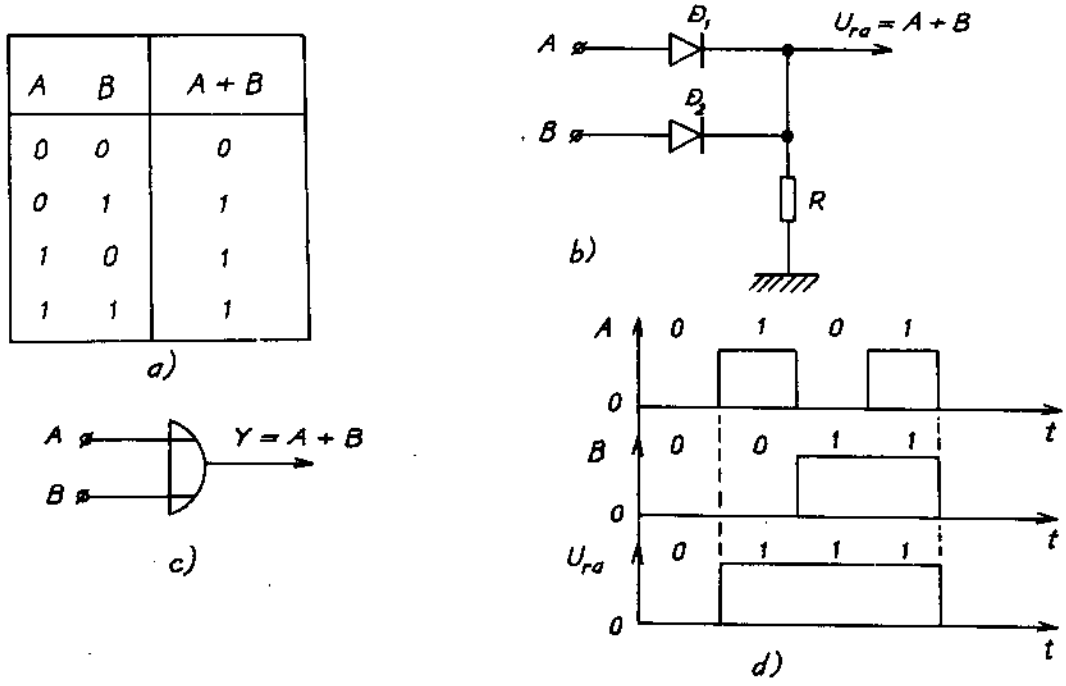
giá trị lôgic 1 ứng với +E (+5 von)

Nếu không có quy ước lôgic kỹ thuật như vậy máy sẽ không hiểu người và ngược

lại người cũng không hiểu máy. Cùng một mạch điện nếu quy ước lôgic khác sẽ có chức năng khác. Mỗi người có quyền đưa ra những mức lôgic khác nhau cho riêng mình, nhưng ở đây ta quy ước mức lôgic như vậy là mức phổ thông và hay gặp nhất trong các hệ thống số hiện nay.

1.7.5.1. Mạch hoặc (OR) dùng điôt điện trở

Bảng chức năng, mạch điện nguyên lý, biểu đồ thời gian và ký hiệu của mạch OR dùng điôt - điện trở được mô tả trên hình 1.45.



Hình 1.45. Mạch Hoặc (OR).

Nguyên lý làm việc được hiểu qua các trường hợp sau :

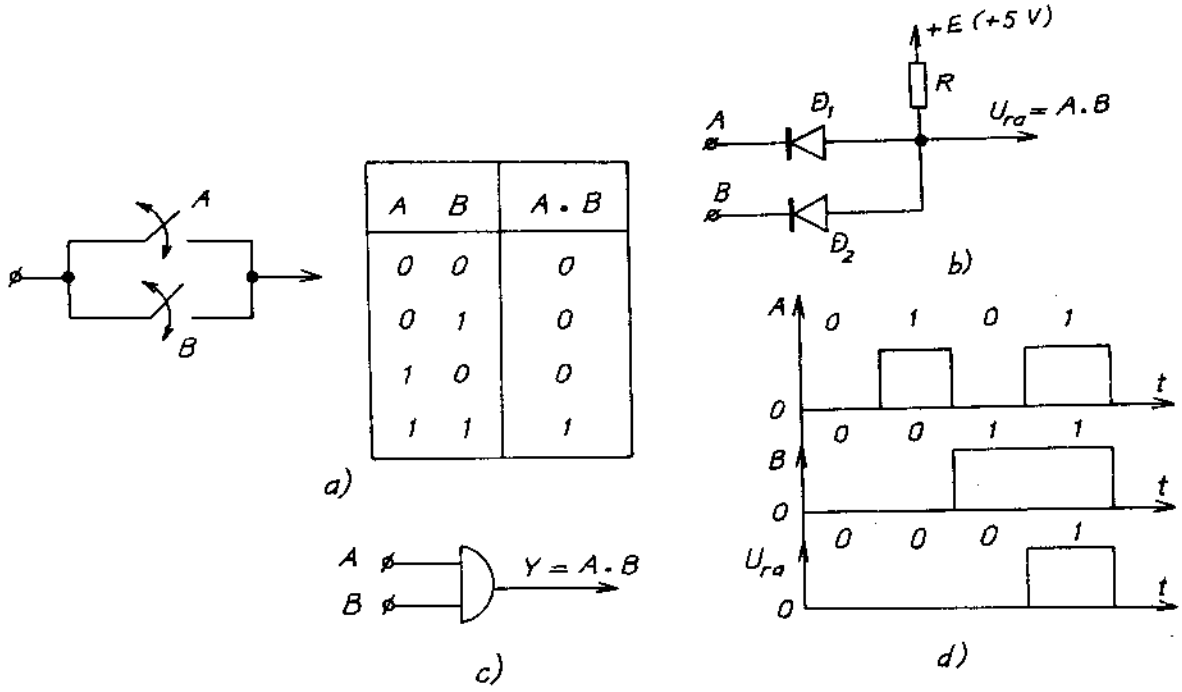
Khi A = 0, B = 0 tức là cho vào đầu A và B của mạch điện điện áp 0 von, khi đó $Đ_1$ và $Đ_2$ không có điện áp nên hở ra ($Đ_1$, $Đ_2$ không thông) làm cho U_{ra} = 0 von ứng với lôgic 0.

Khi A = 1, B = 0 tức là cho vào đầu A điện áp +E (+5 von) còn cho vào đầu B của mạch điện áp 0 von, khi đó $Đ_1$ sẽ thông (đóng mạch) vì được phân cực thuận, sẽ đưa điện áp +E tới đầu ra làm cho U_{ra} = +E ứng với lôgic 1 còn điôt $Đ_2$ sẽ tắt và hở ra, nó không làm ảnh hưởng tới kết quả tại (U_{ra}) đầu ra.

Với cách làm tương tự ta sẽ hiểu được nguyên lý của mạch ở hình 1.44,b. Đồng thời ta thấy mạch điện đó thoả mãn bảng chức năng và biểu đồ thời gian của nó. Đây là một trong những mạch lôgic có cấu tạo cơ bản nhất và đơn giản nhất.

1.7.5.2. Mạch và (AND) dùng điôt - điện trở

Bảng chức năng, mạch điện nguyên lý, biểu đồ thời gian và ký hiệu của mạch AND dùng điôt - điện trở được mô tả trên hình 1.46.

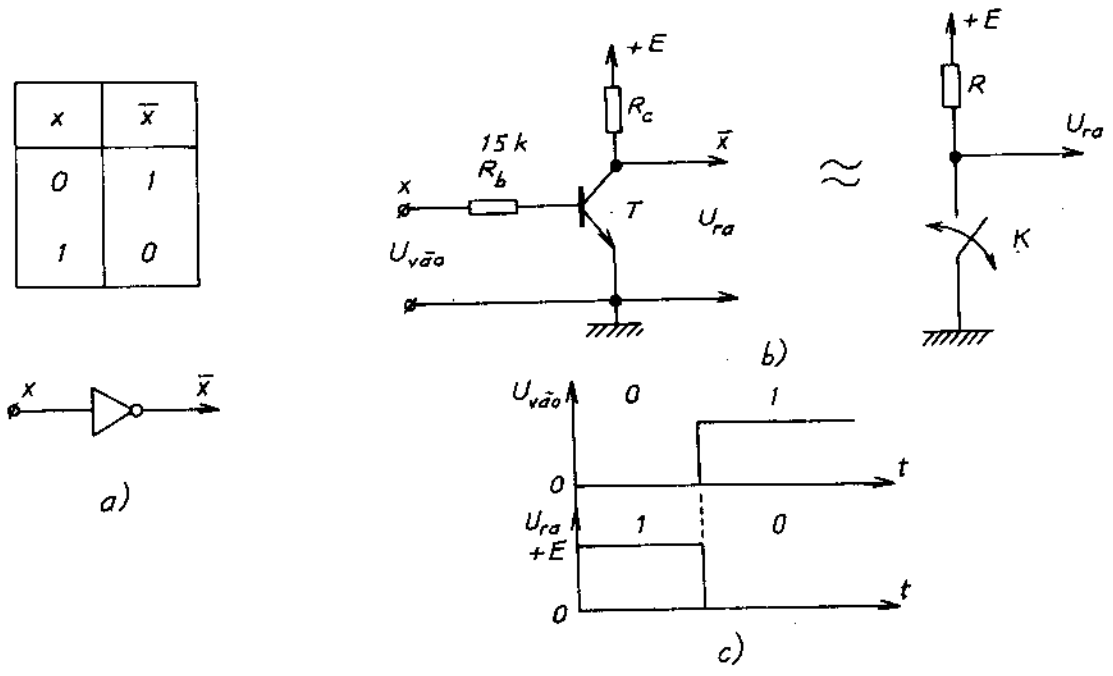


Hình 1.46. Mạch Và (AND)

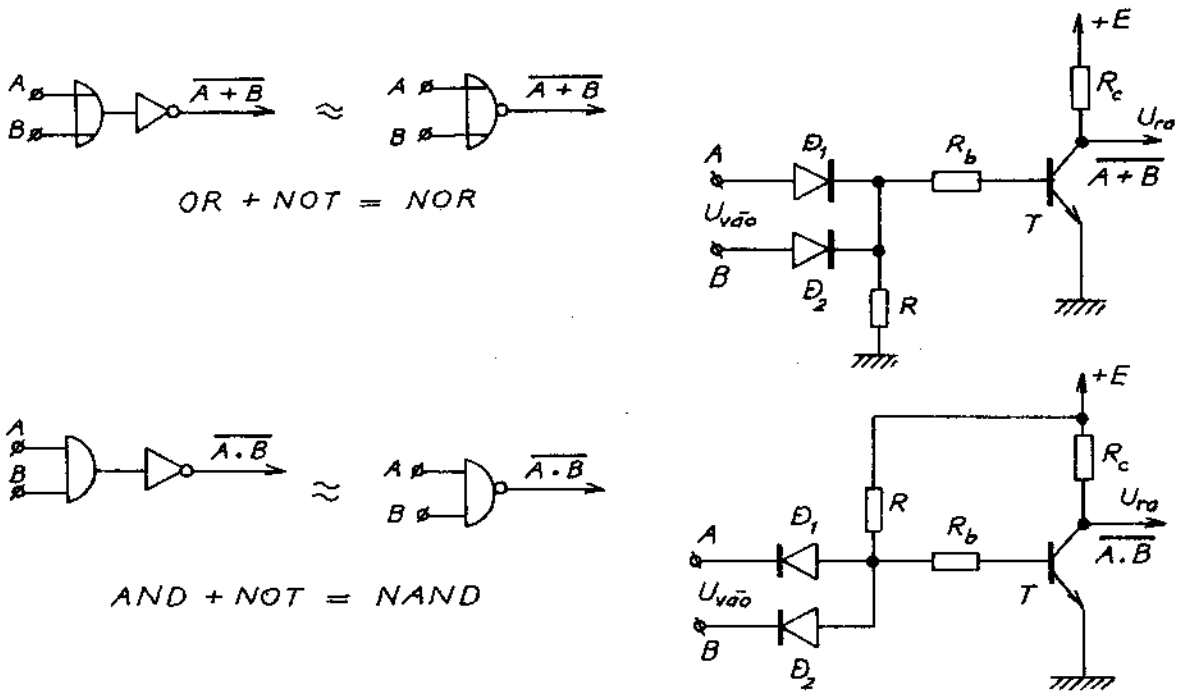
Nguyên lý của mạch Và rất đơn giản, cách hiểu nguyên lý làm việc tương tự ở phần trên.

1.7.5.3. Mạch phủ định (NOT) dùng tranzitor

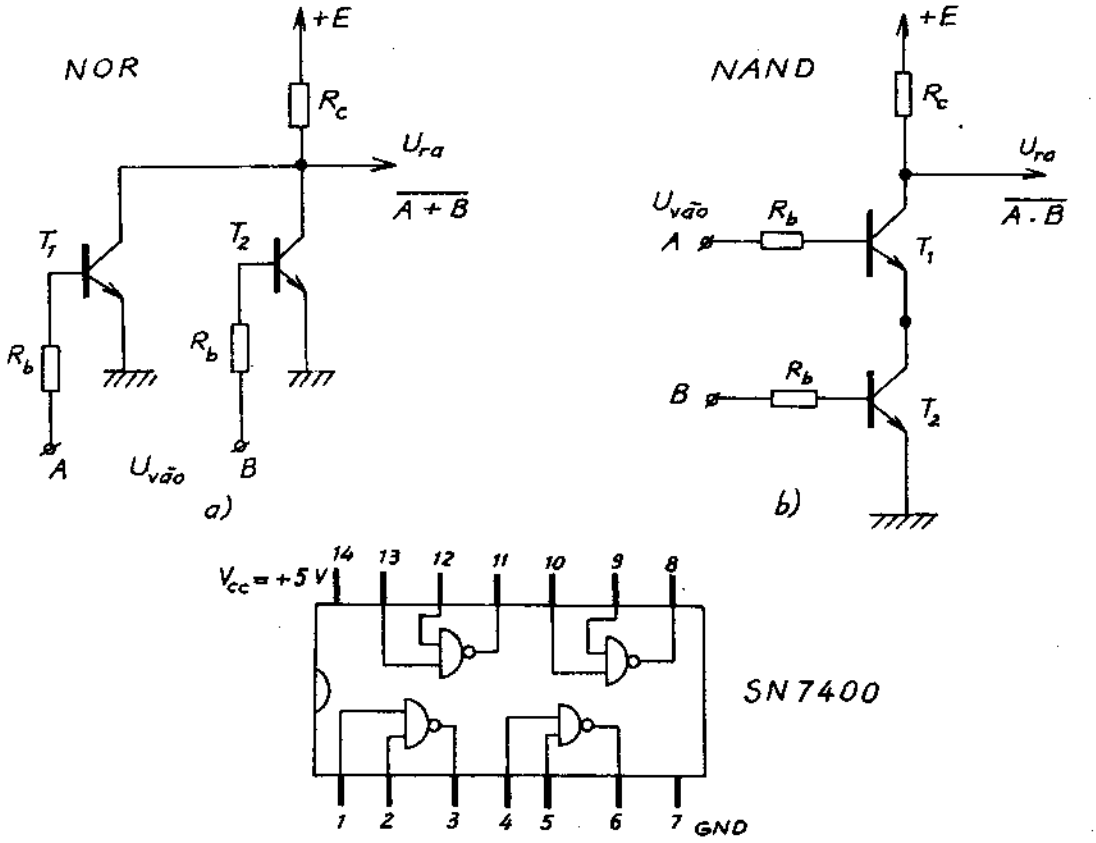
Chức năng và mạch điện phủ định (NOT) được mô tả ở hình 1.47.



Hình 1.47. Mạch Phủ định (NOT)



Hình 1.48. Mạch lôgic kết hợp.



Hình 1.49. Mạch lôgic dùng tranzitor.

Nguyên lý làm việc của mạch phủ định (NOT) cũng chính là một "khóa tranzitor" làm việc ở chế độ bão hòa. Tranzitor T đóng vai trò như một công tắc (khóa) K đóng hay mở trên hình 1.46,b.

Khi ta cho điện áp vào $U_{vào}$ = 0 von lưu ý lôgic 0 thì tranzitor T sẽ tắt (K hở ra) khi đó U_{ra} = +E ứng lôgic 1 còn khi ta cho $U_{vào}$ = +E ứng với lôgic 1 thì làm cho tranzitor T thông (K đóng lại) khi đó U_{ra} = 0 von ứng với lôgic 0.

1.7.5.4. Các mạch lôgic cơ bản kết hợp

Đây là những mạch thực tế hay gặp hơn cả trong các hệ thống số, chúng được tạo ra bằng cách kết hợp giữa mạch hoặc OR, mạch và AND với mạch phủ định NOT. Chức năng, ký hiệu, mạch điện của chúng được mô tả trên hình 1.48.

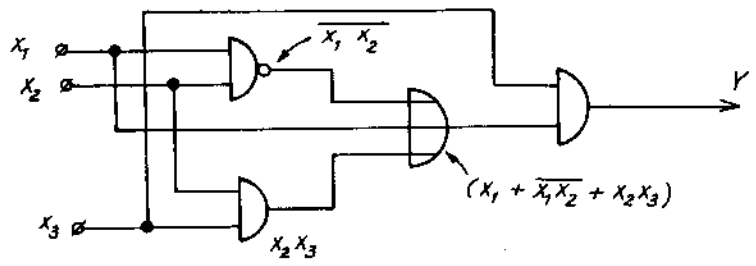
Những mạch NAND và NOR còn có thể được xây dựng bằng tranzitor, chúng còn được chế tạo thành các IC tiêu chuẩn được mô tả trên hình 1.49. (VD : IC SN7400 mạch NAND).

§1.8. NGUYÊN LÝ TỔNG HỢP VÀ PHÂN TÍCH HÀM BOOLE

Tổng hợp (thiết kế) và phân tích (sửa chữa) đó là hai nhiệm vụ chính không thể thiếu được của một kỹ sư khi làm việc với các hệ thống số. Trong phần này sẽ trình bày những nguyên lý cơ bản để thực hiện hai nhiệm vụ đó.

1.8.1. PHÂN TÍCH HÀM BOOLE

Phân tích hàm Boole là một nhiệm vụ khi phải sửa chữa một hệ lôgic tổ hợp, nó là một bài toán ngược, tức là từ sơ đồ mạch điện cho trước ta phải tìm lại được chức năng của mạch điện đó. Ta phải tìm được phương trình toán học cũng như bảng chức năng – là hoạt động chính xác của mạch, từ đó ta mới sửa chữa được hệ thống. Để hiểu được cách làm ta có ví dụ sau : cho trước một mạch lôgic tổ hợp như trên hình 1.50.



Hình 1.50. Mạch điện cụ thể.

Từ mạch điện trên hình 1.50 ta có phương trình chức năng của mạch như sau :

$$Y = x_3 . (x_1 + \overline{x_1 x_2} + x_2 x_3)$$

Bằng cách tìm phương trình toán qua từng mạch lôgic cơ bản cho tới kết quả cuối cùng. Sau khi có phương trình chức năng ta có thể dựa vào nó lập lại bảng chức năng cho mạch điện đó.

1.8.2. TỔNG HỢP HÀM BOOLE

Một hệ lôgic tổ hợp được xây dựng theo yêu cầu của một nhiệm vụ kỹ thuật nào đó. Để xây dựng hay tổng hợp nên một hệ tổ hợp lôgic phức tạp có thể nhận được bằng cách liên kết các phần tử (mạch) lôgic cơ bản lại với nhau theo khả năng là : Đầu ra của một phần tử lôgic cơ bản có thể được nối với một hay nhiều phần tử lôgic cơ bản khác, và các đầu ra của các mạch lôgic cơ bản không được nối trực tiếp với nhau vì sẽ làm mất chức năng lôgic.

Quá trình thực hiện tổng hợp một hệ lôgic tổ hợp có thể phân chia làm ba bước : Bước thứ nhất là xây dựng bảng chân lý – bảng thật mô tả hoạt động của chức năng mà hệ lôgic tổ hợp phải thực hiện. Sau đó bước thứ hai viết chức năng của bảng thật dưới dạng hàm f(X), sau đó dựa vào tiên đề của đại số Boole rút gọn hàm lôgic đó.

Bước cuối cùng là bước thứ ba dựa vào phương trình đã rút gọn chọn các phần tử lôgic cơ bản nối chúng lại với nhau thực hiện hàm lôgic đó. Sau đây chúng ta sẽ xét các bước đã nêu qua một ví dụ cụ thể.

Ví dụ : Ta có một chức năng kỹ thuật hoạt động theo bảng chân lý cho ở bảng trên hình 1.51.

x_1	x_2	x_3	$f(x_1,x_2,x_3)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Hình 1.51. Bảng chức năng cụ thể.

Từ bảng chức đó ta có biểu thức toán học tương ứng có dạng :

$$f(X) = f(x_1, x_2, x_3) = \bar{x}_1 x_2 \bar{x}_3 + \bar{x}_1 x_2 x_3 + x_1 \bar{x}_2 \bar{x}_3 + x_1 x_2 \bar{x}_3 + x_1 x_2 x_3$$

Đến bước tiếp theo ta biến đổi toán học rút gọn hàm số đó như sau :

$$f_3(X) = \bar{x}_1 x_2 (\bar{x}_3 + x_3) + (\bar{x}_1 + x_1) x_2 x_3 + x_1 (\bar{x}_2 + x_2) \bar{x}_3 + x_1 x_2 (\bar{x}_3 + x_3) =$$

$$= \bar{x}_1 x_2 + x_2 x_3 + x_1 \bar{x}_3 + x_1 x_2$$

Ở đây khi nhóm họp để rút gọn ta chú ý tới một khả năng là một tích số – một implicăng có thể được nhóm họp nhiều lần vì : $x + x = x$ hay (nó + nó) = nó hay $\bar{x}_1 x_2 x_3 + \bar{x}_1 x_2 x_3 = \bar{x}_1 x_2 x_3$ đồng thời tính chất giao hoán thì luôn tồn tại (luôn có), nhưng trong phương trình toán lôgic ta không dùng tính chất giao hoán mà viết các biến số theo trật tự tăng dần.

Theo tiêu đề của đại số Boole : $(x + \bar{x}) = 1$ cho nên ta có quan hệ $\bar{x}_1 x_2 (\bar{x}_3 + x_3) = \bar{x}_1 x_2$ đây gọi là quan hệ Dán. Từ đó ta có kết quả :

$$f_3(X) = \bar{x}_1 x_2 + x_2 x_3 + x_1 \bar{x}_3 + x_1 x_2 =$$

$$= (\bar{x}_1 + x_1) x_2 + x_2 x_3 + x_1 \bar{x}_3 =$$

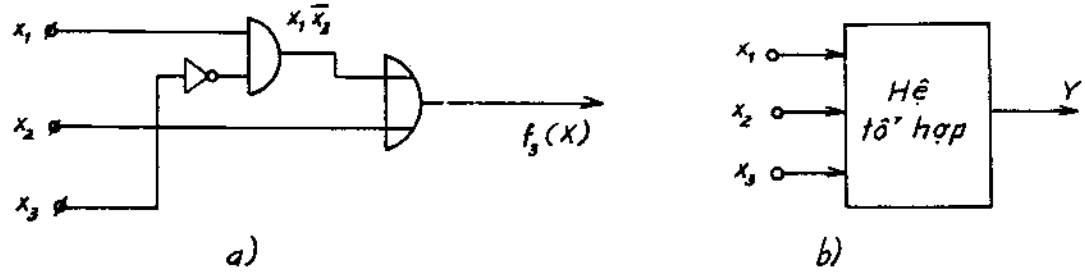
$$= x_2 + x_2 x_3 + x_1 \bar{x}_3 =$$

$$= x_2 (1 + x_3) + x_1 \bar{x}_3 = x_2 + x_1 \bar{x}_3$$

Ở đây theo tiên đề là $(1 + x) = 1$ hay có thể viết (1 + bất kỳ) = 1, cho nên $(1 + x_3) = 1$. Đây gọi là phép toán Nuốt. Sau khi rút gọn ta có hàm tối thiểu của bảng chức năng có dạng :

$$f_3(X)_{min} = x_2 + x_1 \bar{x}_3$$

Sau khi có $f_3(X)_{min}$ ta đi đến bước cuối cùng là dựa vào hàm số được rút gọn đó, xây dựng hay tổng hợp ra mạch lôgic hay là hệ lôgic tổ hợp có dạng minh họa trên hình 1.52.



Hình 1.52. Mạch điện của bảng chức năng đã cho.

Để thực hiện hay tổng hợp ra một OTOMAT không có nhớ – một hệ tổ hợp, một trong những bước quan trọng là phải rút gọn OTOMAT mạch thực hiện sẽ đơn giản hơn rất nhiều. Qua đó ta thấy rõ được tầm quan trọng của việc rút gọn hàm lôgic. Để có phương pháp rút gọn được nhanh chóng và dễ dàng ta đi tới phần tiếp theo đó là phần rút gọn hàm lôgic.

§1.9. RÚT GỌN HÀM LÔGIC (HÀM BOOLE)

Trước đây khi lắp ráp xây dựng các hệ tổ hợp dùng linh kiện rời thì việc tối thiểu hóa hay rút gọn mạch tổ hợp là rất quan trọng và cần thiết, nó là một trong những vấn đề căn bản của lý thuyết tổng hợp ôtômat không có nhớ. Nhưng từ khi kỹ thuật vi điện tử ra đời phát triển, khi con người có thể chế tạo ra các mạch vi điện tử cỡ lớn LSI và siêu lớn VLSI thì vấn đề tối thiểu hóa hàm Boole không còn có ý nghĩa lớn và quan trọng như xưa nữa, vì việc tổng hợp các mạch lôgic dựa trên cơ sở của các IC môđul có sẵn, đồng thời giá thành của các IC cũng không lớn nên mạch có phức tạp hơn chút ít cũng không thành vấn đề lớn cần phải quan tâm.

Nhưng trong phần này khi chúng ta quan tâm tới việc rút gọn hàm Boole, thì vấn đề không chỉ dừng ở các suy nghĩ đó, mà nó còn liên quan tới những vấn đề khác cần thiết đối với một kỹ sư điện tử số như sau :

Việc rút gọn hàm lôgic – hàm Boole là điều của một kỹ sư điện tử số phải biết, vì khi tìm hiểu về việc rút gọn hàm Boole thì họ nhận được những hiểu biết cơ bản dùng

để trao đổi thông tin với đồng nghiệp, nhất là khi phải thiết kế xây dựng những mạch điện chuyên dụng và riêng biệt. Đồng thời trong quá trình phân tích và tổng hợp ôtômat chúng ta luôn dùng đến khái niệm rút gọn hàm lôgic hay tối thiểu hóa các biểu thức của hàm Boole.

Ở đây tối thiểu hóa hàm chuyển mạch là một phương pháp nhằm làm đơn giản hóa hàm chuyển mạch, từ đó có thể thực hiện lắp ráp hàm chuyển mạch – hàm lôgic với số lượng phần tử mạch điện ít nhất. Thực chất là làm đơn giản hóa cách biểu diễn hàm số của hàm chuyển mạch trong một chức năng kỹ thuật cụ thể. Có nhiều phương pháp để rút gọn hàm lôgic như có thể dựa vào các tiên đề, các định lý các định nghĩa và các tính chất của đại số Boole như đã tỉnh bầy ở các phần trên ($x + x = x$; $x + 1 = 1$; $x . \bar{x} = 0$; $x . 0 = 0$; $x + x . y = x$...). Phương pháp dùng biến đổi toán học thường được gọi là phương pháp đại số, dùng phương pháp đại số để rút gọn chỉ tiện lợi đối với hàm ít biến, khi số lượng biến số (tín hiệu vào) tăng quá lớn thì không thể dùng phương pháp này được, vì thời gian thực hiện lâu.

Để thực hiện việc rút gọn hàm Boole khi có nhiều biến người ta dùng phương pháp thuật toán, cho phép thực hiện rút gọn hàm Boole dễ dàng, đồng thời tạo ra các thuật toán cho phép dễ dàng lập trình trên máy tính để tự động hóa quá trình rút gọn làm Boole. Các phương pháp thuật toán hiện nay gồm : phương pháp rút gọn bằng bìa Kácnô (Karnaugh), phương pháp Quine–Mc.Cluskey, phương pháp Roth...

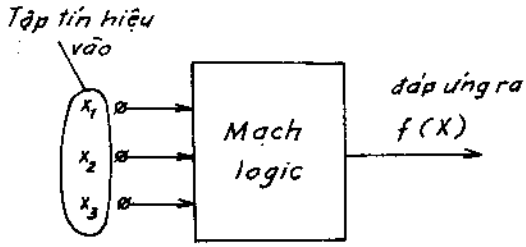
1.9.1. RÚT GỌN HÀM BOOLE THEO PHƯƠNG PHÁP BÌA KÁCNÔ

1.9.1.1. Rút gọn bài toán lôgic hoàn toàn xác định

Phương pháp dùng bìa Kácnô là một phương pháp dựa trên cách biểu diễn hàm lôgic dưới dạng bảng chân lý. Khi nhìn vào bảng và dựa vào nhận xét và so sánh ta sẽ có được dạng tối thiểu của hàm Boole một cách dễ dàng. Để hiểu được cụ thể phương pháp này ta xuất phát từ một nhiệm vụ kỹ thuật được biểu diễn ở bảng chân lý (bảng thật) biểu diễn trên hình 1.53.

x_1	x_2	x_3	f(X)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

a)



b)

Trong đó : $X = (x_1, x_2, x_3)$ là tập tín hiệu vào còn f(X) là đáp ứng ra của mạch, chúng lấy giá trị là 0 hoặc 1.

Hình 1.53. Chức năng kỹ thuật

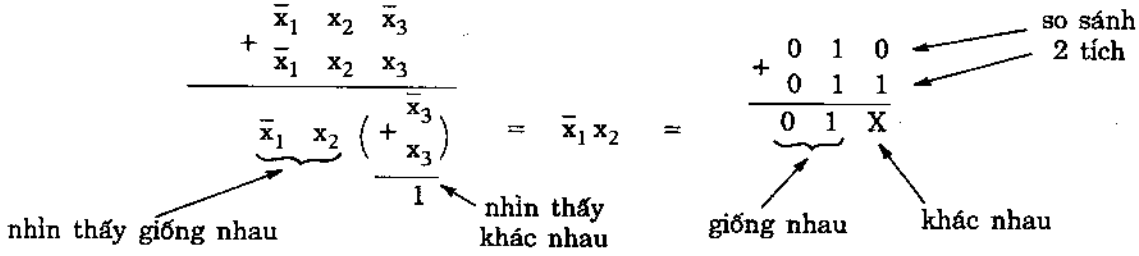
Từ bảng chân lý hàm lôgic của nó có dạng :

$$f(x) = f_3 = f(x_1, x_2, x_3) = \bar{x}_1\, x_2\, \bar{x}_3 + \bar{x}_1\, x_2\, x_3 + x_1\, \bar{x}_2\, \bar{x}_3 + x_1\, x_2\, \bar{x}_3 + x_1\, x_2\, x_3$$

Ta có thể nhóm họp (được gọi là phép toán Dán) hai tích số của hàm số đó như sau :

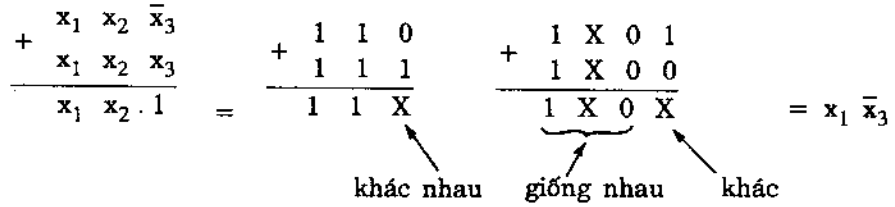
$$(\bar{x}_1\, x_2\, \bar{x}_3 + \bar{x}_1\, x_2\, x_3) = \bar{x}_1\, x_2\, (\bar{x}_3 + x_3) = \bar{x}_1 \cdot x_2$$

Dựa vào tiên đề $(x + \bar{x}) = 1$ ta đã thực hiện được việc rút gọn hai tích số đó theo phương pháp đại số. Nhưng nếu ta biểu diễn theo cách khác, viết theo hàng dọc hai tích đó ta có :



Khi ta nhìn so sánh lần lượt các biến của hai tích số đó, với nhau ta thấy biến số đầu tiên ($\bar{x}_1$) và biến số thứ hai (x_2) của chúng giống nhau, còn biến số thứ ba là (x_3 với $\bar{x}_3$) thì khác nhau vì một giá trị có phủ định còn quá trình kia thì không phủ định. Qua đó ta có nhận xét : nếu so sánh các biến số của hai tích số, *nếu chúng giống nhau thì kết quả của chúng còn ở đáp số, còn nếu chúng khác nhau thì ở đáp số biến đó không còn* (bị mất). Như vậy chúng ta đã chuyển quá trình tính toán theo tiên đề thành việc nhìn và so sánh nhận xét giữa hai tích số, kết quả thu được là như nhau.

Với cách làm dựa vào nhận xét tương tự ta có thể "tính" được kết quả rút gọn của các tích số sau :



Chú ý ở một số sách nước ngoài ngoài khi viết về đại số Boole người ta có thể viết kết quả tính toán hay hàm Boole dưới dạng (0 1 X) = (0 1 2) ; (1 1 X) = (1 1 2) = $x_1\, x_2$; (1 x 0 x) = (1 2 0 2) = $x_1\, \bar{x}_3$)

Nhưng khi so sánh và nhận xét giữa hai tích số của hàm Boole để rút gọn chúng, thì ta gặp phải trường hợp sau :

$$
\begin{array}{r} x_1\ \bar{x}_2\ \bar{x}_3 \\ +\ x_1\ x_2\ x_3 \\ \hline x_1\ ?\ ? \end{array}
\qquad
\begin{array}{r} 1\ 0\ 0 \\ +\ 1\ 1\ 1 \\ \hline 1\ ?\ ? \end{array}
\quad ;
\qquad
\begin{array}{r} 1\ x\ 0\ 1 \\ +\ 1\ 0\ 0\ 0 \\ \hline 1\ ?\ 0\ ? \end{array}
\qquad
\begin{array}{r} x_1 . 1 . \bar{x}_3\ x_4 \\ +\ x_1\ \bar{x}_2\ \bar{x}_3\ \bar{x}_4 \\ \hline x_1\ ?\ \bar{x}_3\ ? \end{array}
$$

Nhận thấy khi so sánh hai tích số (hai implicăng) để thực hiện rút gọn chúng, trong trường hợp này thì số lượng vị trí khác nhau của chúng lớn hơn hoặc bằng hai (≥ 2). Khi đó hai tích số mà ta nhóm họp và muốn rút gọn sẽ không thể rút gọn được. Gặp trường hợp này ta phải để nguyên hai tích số đó. Hay ta có thể nói một cách khác là : nếu hai tích số (hai implicăng) khi so sánh với nhau thấy có số lượng vị trí khác nhau lớn hơn 2, thì chúng không thể rút gọn được. Vậy muốn rút gọn được hai tích số (hai implicăng) thì chúng phải khác nhau chỉ một vị trí.

Định nghĩa 1.9.1 : Hai implicăng (tích số) được gọi là "kế cận, cạnh nhau, hàng xóm" nếu chúng khác nhau chỉ một vị trí, khi đó chúng có thể nhóm họp và rút gọn được.

Từ định nghĩa đó ta nhận thấy nếu có một cách nào đó sắp xếp được các tích số (implicăng) của một hàm lôgic bất kỳ vừa "cạnh nhau" theo hình thức vừa "cạnh nhau" theo định nghĩa vào cùng một khu vực, thì việc rút gọn hàm Boole đó sẽ thực hiện được một cách dễ dàng và nhanh chóng.

Cũng từ suy nghĩ đó Kácnô đã đưa ra một phương pháp sắp xếp các implicăng được gọi tên là bìa Kácnô (bìa K), bìa Kácnô cho phép sắp xếp các tích số của hàm lôgic vừa "cạnh nhau" theo hình thức vừa "cạnh nhau" theo định nghĩa vào cùng một khu vực, để có thể dễ dàng thực hiện việc nhóm họp (rút gọn). Điều đó được minh họa trên hình 1.53 như sau đối với hàm 3 biến :

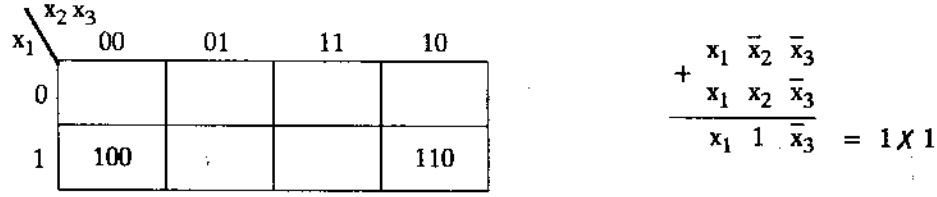
x_1 \ x_2x_3	00	01	11	10
0	000	001	011	010
1	100	101	111	110

x_1 \ x_2x_3	00	01	11	10
0	0	1	3	2
1	4	5	7	6

Hình 1.54. Bìa Kácnô hàm 3 biến.

Chú ý nhất ở đây là cách biểu diễn trục số của bìa Kácnô, với cách biểu diễn đó cho phép ta nhận thấy là : mỗi một ô của bìa ứng với một tọa độ (viết dưới dạng số khi phân), các ô đó vừa "cạnh nhau" theo hình thức ô nọ cạnh ô kia, đồng thời tọa độ của các ô luôn khác nhau chỉ một vị trí – thỏa mãn "cạnh nhau" theo định nghĩa. Như vậy bìa Kácnô đã thỏa mãn được yêu cầu là giữa các ô vừa "cạnh nhau" theo hình thức vừa "cạnh nhau" theo định nghĩa, mà mỗi một ô lại tương ứng với một tích số của hàm lôgic (ví dụ ô có tọa độ (0 1 0) = $\bar{x}_1\ x_2\ \bar{x}_3$), nói cách khác là bìa Kácnô (bìa K) cho phép sắp xếp được các tích số của nhau lôgic vừa "cạnh nhau" theo hình thức vừa "cạnh nhau" theo định nghĩa vào cùng một khu vực. Vấn đề chỉ còn lại là ứng dụng bìa K thế nào để thực hiện việc tối thiểu hóa hàm Boole được nhanh chóng và dễ dàng.

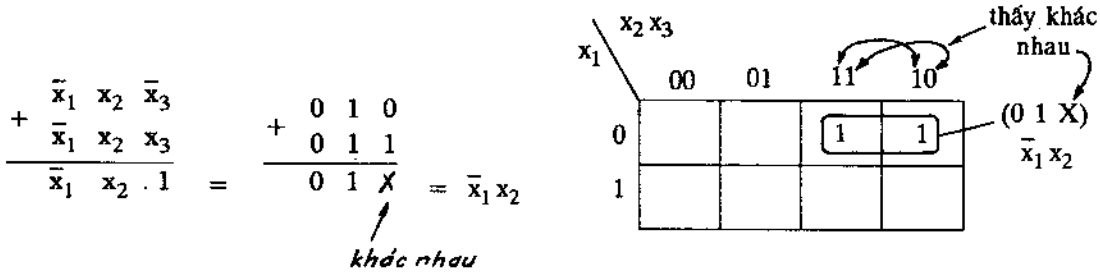
Với bìa Kácnô còn cần phải lưu ý thêm tới trường hợp hai implicăng không cạnh nhau theo hình thức nhưng vẫn rút gọn được với nhau, vì chúng chỉ "cạnh nhau" theo định nghĩa do chúng khác nhau chỉ một vị trí. Hiện tượng đó được minh họa trên hình 1.55.



Hình 1.55. Các ô cạnh nhau theo định nghĩa.

Từ đó ta có thể hình dung ra là hai mép cuối của bìa K có thể được "cuộn lại" dính với nhau. Trên bìa Kácnô hai tích số có thể "không cạnh nhau" theo hình thức mà chỉ "cạnh nhau" theo định nghĩa, nhưng ta vẫn quản lý được chúng, nên vẫn có thể thực hiện việc nhóm họp hay rút gọn chúng một cách dễ dàng trên bìa Kácnô.

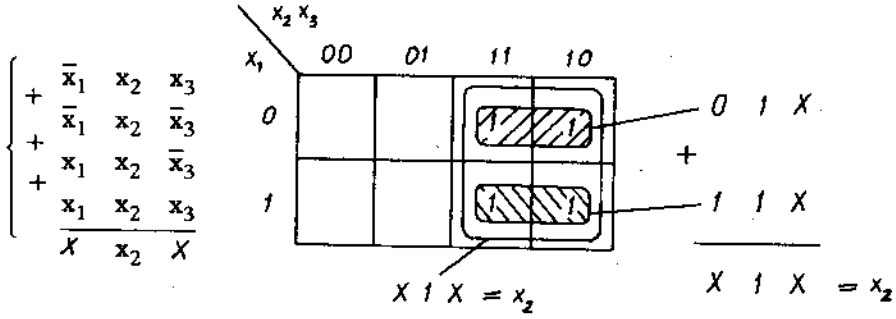
Ứng với bìa Kácnô thực hiện rút gọn hàm Boole được minh họa trên hình 1.56, khi ta ánh xạ hai tích số (đã nêu ở phần trên) vào bìa.



Hình 1.56. Rút gọn trên bìa K

Giá trị lôgic 1 ta nhìn thấy ở trong một ô của bìa Kácnô nó đại diện cho sự tồn tại của tích số trong phương trình tương ứng với tọa độ của ô đó trong bìa Kácnô, đồng thời nó chính là đáp ứng ra của mạch điện ứng với tập tín hiệu vào là tích số đó. Hai tích số có thể rút gọn được nay tương ứng với hai con số 1 lôgic ở hai ô cạnh nhau trên bìa K, chúng có thể được khoanh dán lại, gộp lại hoặc rút gọn với nhau. Từ đó việc so sánh nhận xét giữa các biến của hai tích số giờ đây tương ứng với việc *so sánh tọa độ của hai ô được khoanh dán trên bìa.* Kết quả của việc so sánh tọa độ của hai ô tương ứng với kết quả biến đổi toán học. Như vậy từ nay việc rút gọn hàm Boole chỉ còn là việc só ánh tọa độ của các ô có giá trị 1 cạnh nhau trong bìa Kácnô và cứ tọa độ nào giống nhau thì tồn tại ở kết quả việc rút gọn, còn nếu tọa độ ứng với từng biến số mà khác nhau thì bị mất (viết là dấu X) trong kết quả rút gọn thu được.

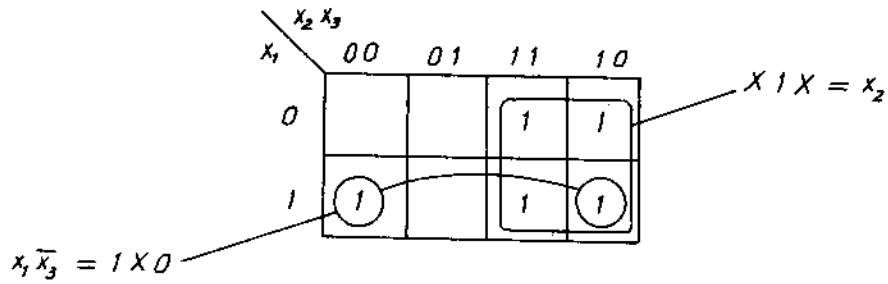
Khi giá trị 1 trong bìa K xuất hiện nhiều hơn như trên hình 1.57.



Hình 1.57. Hàm Boole nhiều tích số.

Do bìa Kácnô cho phép sắp xếp các implicăng "cạnh nhau theo định nghĩa" vào trong cùng một khi vực cho nên ta có thể so sánh tọa độ của cả bốn ô hay khoanh dán cả bốn ô đó liền một lúc. kết quả cho thấy là chỉ có giá trị x_2 là giống nhau ở tất cả bốn ô đó (so sánh từ trên xuống dưới) nên giá trị x_2 sẽ xuất hiện ở kết quả (X 1 X = x_2) cuối cùng. Việc so sánh tọa độ của bốn ô đó thực hiện rất dễ dàng khi các giá trị 1 của các ô nằm cạnh nhau ở một khu vực, từ đó ra kết quả rút gọn thu được rất nhanh và đó là ưu việt đầu tiên của bìa Kácnô của ta bắt đầu nhận thấy khi thực hiện rút gọn hàm Boole. Như vậy mỗi một ô trong bìa K ứng với một tọa độ, ứng với một tích số, ứng với một số nhị phân, ứng với một giá trị cơ số mười và ứng với một tập tín hiệu vào mạch lôgic.

Ánh xạ bài toán rút gọn hàm Boole đã cho ở phần đầu vào bìa Kacnô sẽ có dạng được biểu diễn trên hình 1.58.



Hình 1.58. Rút gọn bài toán cụ thể.

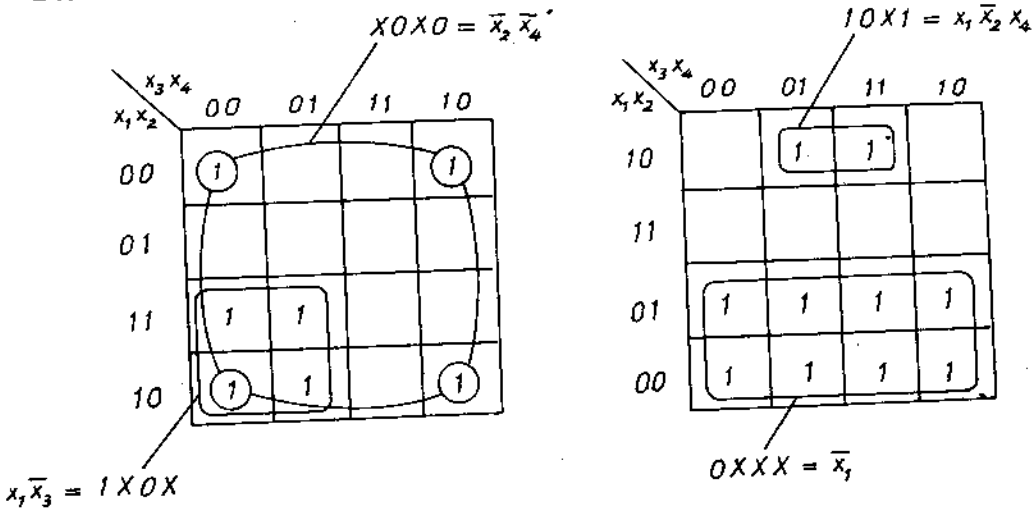
Dựa vào khả năng khoanh dán các ô "cạnh nhau" trên bìa K ta thu được kết quả của mỗi một khoanh dán, mỗi một khoanh dán ứng với một mạch VÀ (AND), mạch VÀ đó cho tín hiệu ra ứng với tín hiệu vào là tọa độ trong khoanh đó. Như vậy muốn thực hiện được chức năng (hàm lôgic) đã đề ra thì cần phải cộng các mạch và đó lại, từ đó hàm rút gọn mà ta cần tìm có dạng :

$$f_{min}\ (x_1,\ x_2,\ x_3) = x_2 + x_1\ \bar{x}_3$$

Đó cũng chính là kết quả rút gọn hàm Boole đã được tính toán dựa vào các tiên đề, hay rút gọn hàm Boole theo phương pháp đại số mà ta thu được ở trên. Nhưng dựa vào cách rút gọn hàm Boole dùng bìa K ngoài kết quả $f_{min}(X)$ ta thu được, còn có thể

nhận được thêm các thông tin khác nữa như biết tích số ($x_1\ x_2\ \overline{x}_3$ = (110)) có thể được nhóm họp ba lần theo tính chất (x + x = x hay nó + nó = nó).

Đối với hàm bốn biến bìa Kácnô có dạng trên hình 1.59.



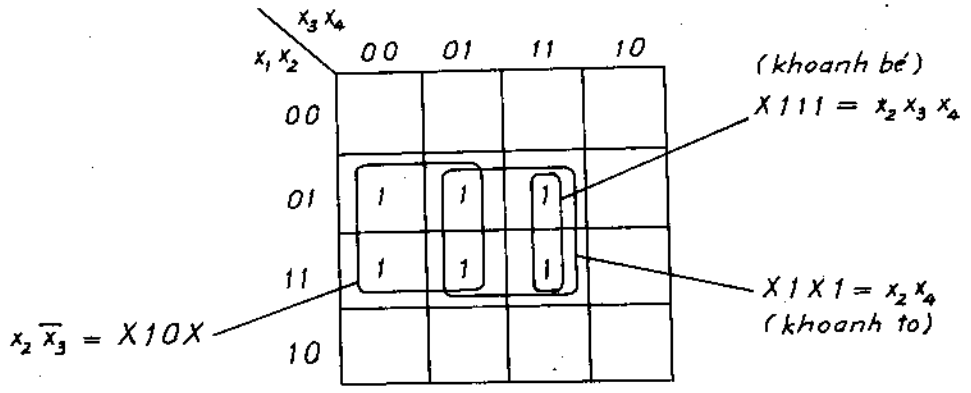
Hình 1.59. Hàm 4 biến.

Dựa vào những kết quả từ việc khoanh dán thu được rút ra nhận xét :

- Kết hợp (khoanh dán) hai ô cạnh nhau sẽ bớt được một biến.
- Kết hợp (khoanh dán) 4 ô cạnh nhau sẽ bớt được 2 biến.
- Tổng quát khoanh dán 2^n ô cạnh nhau sẽ bớt được n biến.

Nói cách khác là chỉ có 2^n ô cạnh nhau, tức là chỉ có (2, 4, 8, 16, 32...) ô cạnh nhau mà thôi.

Có ví dụ minh họa biểu diễn trên hình 1.60.



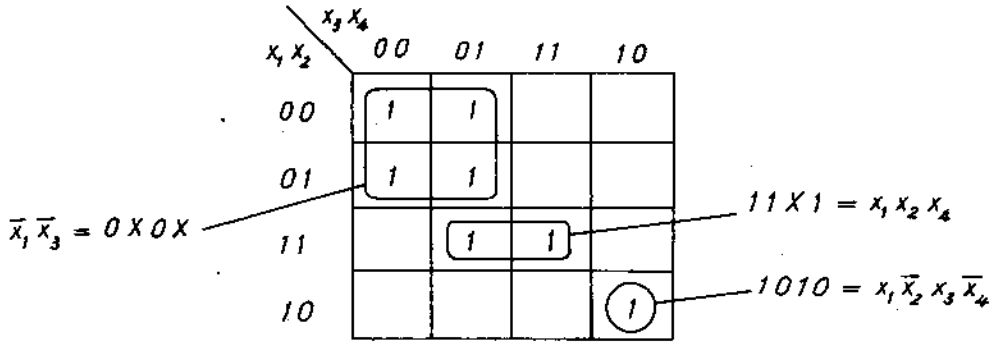
Hình 1.60. Trường hợp khoanh bé.

Ở đây ta có thể thực hiện khoanh 4 ô lại với nhau, nhưng nếu ta chỉ khoanh 2 ô (trong khi có thể khoanh 4) thì được gọi là "khoanh bé", nếu "khoanh bé" thì số lượng biến được bớt đi sẽ ít và mạch lớp ráp sẽ chưa đơn giản, hay còn được gọi là hàm Boole

chưa hoàn toàn tối tiểu hóa. Từ đó ta rút ra kết luận quan trọng là : *Khoanh dán càng to thì mạch điện sẽ càng đơn giản.* Khi rút gọn hàm Boole dùng bìa K thì khoanh càng to càng tốt, điều này làm được dựa vào tính chất là : một implicăng (tích) có thể được nhóm họp nhiều lần vì x + x = x hay (nó + nó) = nó.

Ta có một ví dụ tổng quát về việc rút gọn hàm Boole trên bìa Kácnô như sau : giả sử có một chức năng kỹ thuật được đề ra, ta chuyển chức năng đó thành hàm Boole ở dạng bảng thật (bảng chân lý), từ bảng thật ta ánh xạ hàm đó (chức năng đó) vào bìa Kácnô, ví dụ hàm đó được biểu diễn trên hình 1.61. Từ đó có thể thực hiện được quá trình rút gọn hàm Boole và kết quả thu được sẽ là :

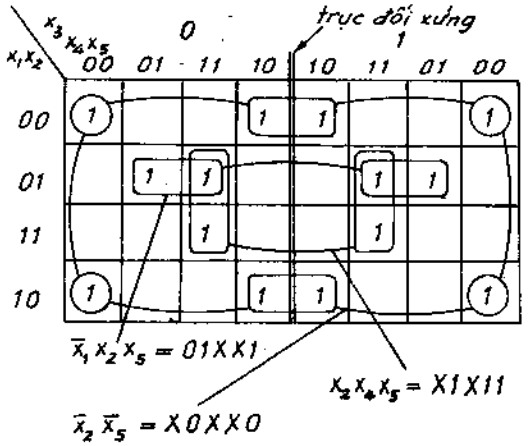
$$f_{min}(x_1, x_2, x_3, x_4) = \bar{x}_1\,\bar{x}_3 + x_1\,x_2\,x_4 + x_1\,\bar{x}_2\,x_3\,\bar{x}_4$$



Hình 1.61. Một ví dụ cụ thể.

Sau khi nhận được hàm lôgic (của chức năng đề ra) đã rút gọn, ta dùng mạch lôgic lắp ráp thành mạch điện thực hiện được chức năng, đó là mạch điện đơn giản nhất.

Cách biểu diễn bìa K ở dạng năm biến được biểu diễn trên hình 1.62.



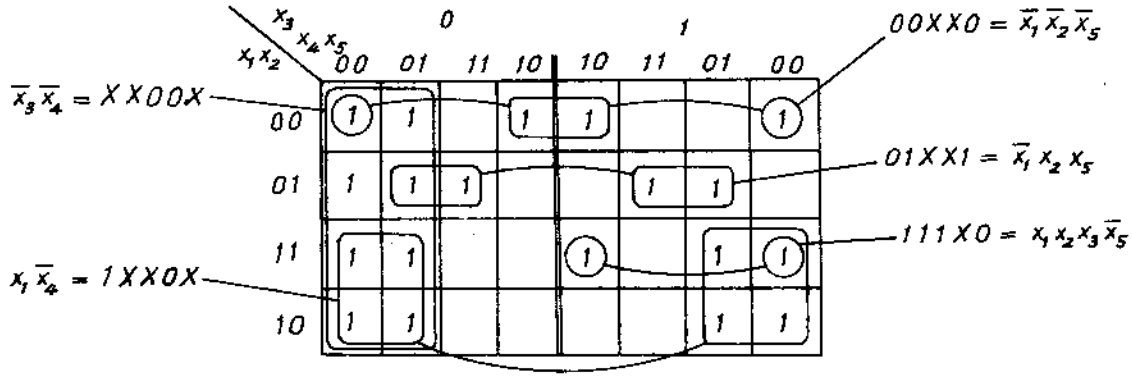
x_1 \ x_3 x_4x_5	00	01	11	10	10	11	01	00
00	0	1	3	2	6	7	5	4
01	8	9	11	10	14	15	13	12
11	24	25	27	26	30	31	29	28
10	16	17	19	18	22	23	21	20

trục đối xứng

Hình 1.62. Hàm Boole năm biến.

Có một ví dụ rút gọn một hàm Boole năm biến bất kỳ được mô tả trên hình 1.63.

$$f_{5min} = f_{min}(x_1, x_2, x_3, x_4, x_5) = \bar{x}_3\,\bar{x}_4 + \bar{x}_1\,\bar{x}_2\,\bar{x}_5 + x_1\,x_2\,\bar{x}_5 + x_1\,x_2\,x_3\,\bar{x}_5 + x_1\,\bar{x}_4$$



Hình 1.63. Minh họa rút gọn một hàm Boole năm biến.

Kết quả $f_{5min}(X)$ dựa vào khoanh dán trên bìa K ta thu được nhanh chóng, chính xác và dễ dàng. Nếu vẫn bài toán này mà áp dụng theo phương pháp đại số, sử dụng tiên đề để biến đổi và rút gọn sẽ không thể dễ dàng cho ta một kết quả như vậy.

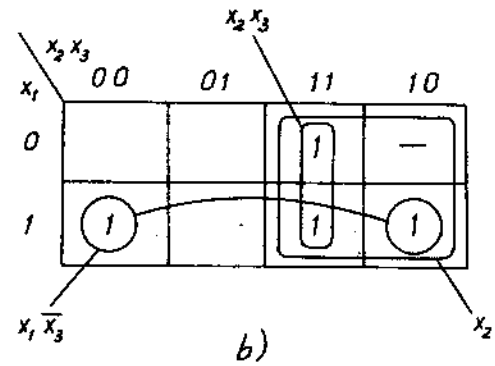
1.9.2. RÚT GỌN BÀI TOÁN LÔGIC KHÔNG HOÀN TOÀN XÁC ĐỊNH

Những bài toán đã xét cho ta thấy : các vị trí ô trong bìa K đều có giá trị 1 hoặc 0 rõ ràng, chúng được gọi một tên chung là "bài toán lôgic hoàn toàn xác định". Nhưng trong thực tế kỹ thuật thì không phải như vậy, không phải lúc nào các vị trí ô (đáp ứng ra) của hàm lôgic trong bìa K đều có giá trị lôgic xác định, khi mà đáp ứng ra đó ta có thể không dùng để điều khiển gì, hay bộ tín hiệu vào cụ thể ứng với ô có tọa độ đó trên bìa không bao giờ xuất hiện trong quá trình làm việc của hệ thống điện tử. Loại bài toán lôgic này có tên gọi là : "bài toán không hoàn toàn xác định", nó được ví dụ minh họa trên hình 4.64.

x_1	x_2	x_3	f(X)
0	0	0	0
0	0	1	0
0	1	0	–
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

vị trí không xác định

a)



Hình 1.64. Bài toán không hoàn toàn xác định

Khi bài toán lôgic có vị trí "không xác định" (có thể ký hiệu là "–" hay "X") thì ta có thể toàn quyền sử dụng vị trí đó sao cho có lợi nhất cho ta thì làm, có thể cho nó

là 0 hay 1 là tùy ta. Thông thường ở vị trí không xác định ta sẽ *chọn sao cho mạch thực hiện hàm lôgic đó được đơn giản nhất*. Để thấy rõ hiệu ứng của cách làm ta có thể giả sử sau :

– Giả sử lấy vị trí không xác định là "1", thì bài toán trở thành bài toán quen thuộc và đã có kết quả là :

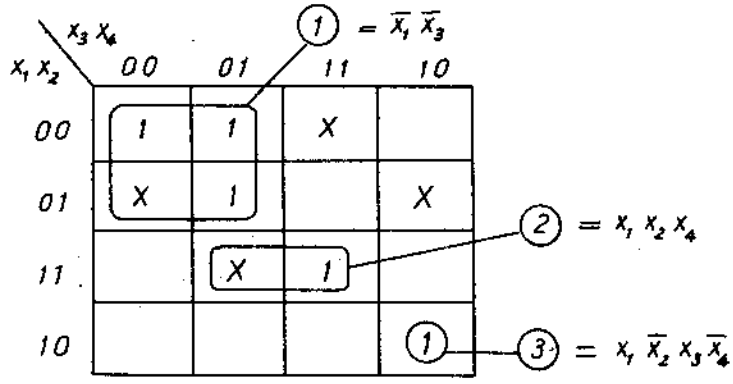
$$f_{3\ min\ "1"} = x_2 + x_1 \bar{x}_3$$

– Giả sử lấy vị trí không xác định là "0", thì kết quả rút gọn của bài toán đó sẽ là.

$$f_{3\ min\ "0"} = x_2\ x_3 + x_1 \bar{x}_3$$

Từ hai kết quả rút gọn thu được của hàm số, ta thấy hai kết quả đó chúng đều thực hiện được chức năng lôgic đã đề ra vì chúng cùng cho tín hiệu ra ở 4 đỉnh (ô) có giá trị là 1. Nhưng độ phức tạp của hàm số cũng như cả mạch điện khi lắp ráp là khác nhau. Từ đó ta biết nên chọn vị trí không xác định như thế nào, khi chọn vị trí không xác định sao cho có lợi nhất cho ta chọn. Cụ thể ở đây ta chọn $f_{3min"1"}$

Từ việc giả sử đó ta nhận thấy : khi có một vị trí không xác định ta sẽ phải giải hai bài toán lôgic khác nhau do phải giả sử để hai lần. Nếu có hai vị trí không xác định ta sẽ phải giải 4 bài toán ứng với 4 tổ hợp lôgic để tìm được mọi khả năng, vì từ đó mới tìm được mạch điện đơn giản nhất. Tổng quát lên nếu có n vị trí không xác định thì ta sẽ phải giải 2^n bài toán để tìm mạch điện đơn giản nhất có lợi nhất. Nhận thấy số lượng bài toán sẽ tăng lên rất nhanh theo hàm mũ và phụ thuộc vào số lượng của vị trí không xác định. Điều đó có thể vượt quá khả năng tính toán của con người. Nhưng nếu dùng bìa Kácnô để giải bài toán tìm hàm rút gọn, ta chỉ cần làm một lần là xong, kết quả luôn cho ta mạch điện là đơn giản nhất hay đáp số là hàm luôn tối tiểu hóa. Điều đó được minh họa ở vị trí chỉ ra trên hình 1.65.



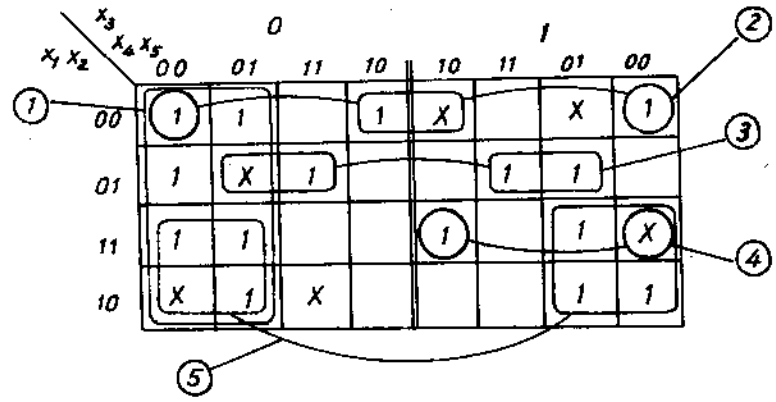
Hình 1.65. Rút gọn bài toán không xác định.

Bài toán không xác định này ta chỉ cần làm một lần, với ba khoanh dán, cho kết quả là :

$$f_{3\ min} = \bar{x}_1 \bar{x}_3 + x_1\ x_2\ x_4 + x_1 \bar{x}_2\ x_3\ \bar{x}_4$$

Đó cũng là mạch đơn giản nhất dễ dàng nhận thấy. Trong khi bài toán này có 4 vị trí không xác định tức là có $2^4 = 16$ khả năng lựa chọn, hay 16 bài toán khác nhau để giải nếu không dùng bài Kácnô. Qua ví dụ này cho ta hiểu hơn và thấy thêm ưu việt của bìa K khi rút gọn hàm Boole.

Dạng tổng quát (thực tế) của bài toán rút gọn hàm Boole 5 biến, mô tả trên bìa K, được minh họa trên hình 1.66.



Hình 1.66. Minh họa rút gọn một hàm lôgic tổng quát.

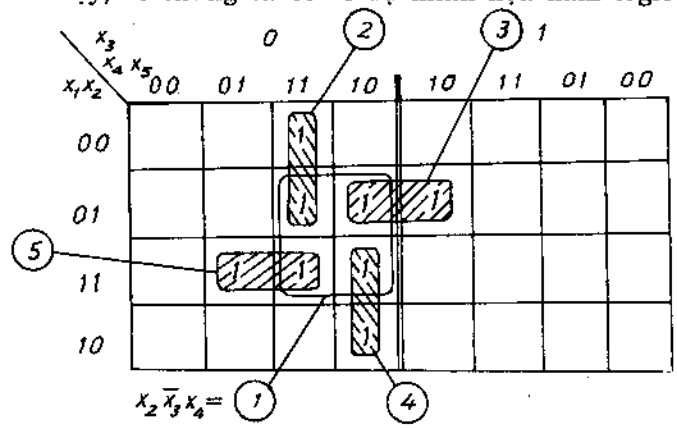
Qua cách khoanh dán trên bìa K ta thấy có những vị trí không xác định (X) ta lại sử dụng trong khoanh dán, và có những vị trí không xác định ta bỏ qua.

Mục đích khoanh dán sao cho ta thu được mạch điện đơn giản nhất, sao cho các vị trí (1) của hàm luôn luôn được thực hiện khoanh dán hết.

1.9.3. ĐỈNH ĐẦU MÚT VÀ CÁC KHỐI ĐẦU MÚT

Rút gọn hàm lôgic dùng bìa K cho ta thấy rằng thực hiện quá trình rút gọn hàm lôgic được thực hiện rất đơn giản và dễ dàng, chỉ cần khoanh dán các ô "cạnh nhau" lại với nhau, và làm sao khoanh dán được càng nhiều ô với nhau – khoanh dán càng to càng tốt, vì khi khoanh dán càng to thì mạch điện lắp ráp hàm lôgic đó càng đơn giản.

Nhưng trong thực tế thì việc rút gọn hàm Boole dùng bìa Kácnô (bìa K) không chỉ đơn giản như vậy, vì chúng ta có ví dụ minh họa hàm lôgic đó có dạng như trên hình 1.67.



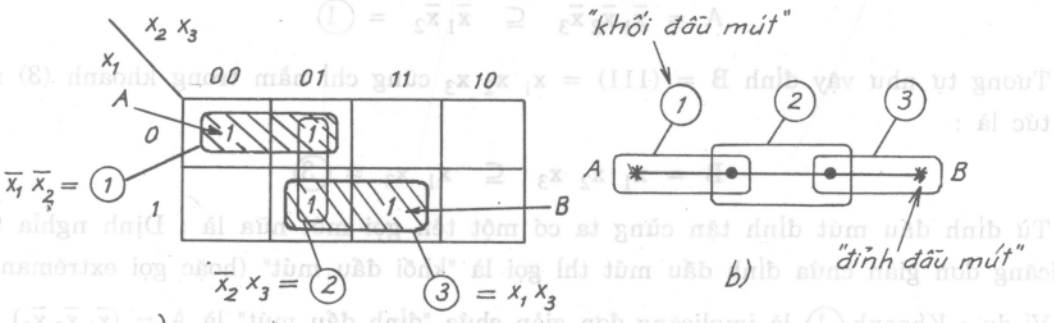
Hình 1.67. Có một chức năng đặc biệt.

Trong quá trình thực hiện khoanh dán để rút gọn hàm lôgic này (nhiệm vụ kỹ thuật này) ta thấy khoanh ① = $x_2 \bar{x}_3 x_4$ là khoanh to nhất theo lý thuyết, nhưng việc thực hiện chức năng lôgic này ta chỉ cần lắp các mạch VÀ ứng với các khoanh (②, ③, ④, ⑤), điều này nhìn trên bìa K rất rõ, còn với khoanh ① khoanh to nhất ta không cần quan tâm không cần lắp ráp mạch VÀ cho nó. Như vậy rõ ràng khoanh dán ① là *khoanh to nhất thì trở thành "khoanh thừa" hay "mạch VÀ thừa"*.

Nếu ta rút gọn hàm lôgic này theo phương pháp đại số dùng tiên đề để biến đổi và rút gọn, thì kết quả rút gọn ① sẽ không thể nhận ra được đó là mạch thừa, vì kết quả ① là từ tiên đề mà ra, từ biến đổi toán học mà xuất hiện ra nó. Đồng thời "mạch thừa" ① không chỉ có một mà nó có thể xuất hiện rất nhiều trong quá trình rút gọn hàm Boole. Nó là những mạch có hay không có nó không ảnh hưởng gì tới chức năng của mạch. Nhưng những "mạch thừa" đó vì có nó sẽ làm tính kinh tế của mạch bị giảm, làm kích thước mạch to lên không cần thiết, đồng thời làm giảm độ tin cậy hoạt động của hệ thống khi các "mạch thừa" này hỏng hóc.

Từ đó xuất hiện vấn đề là : làm sao loại bỏ được các "mạch thừa" đó. Nhưng việc loại bỏ những mạch thừa (như mạch ①) trong quá trình rút gọn hàm lôgic là không dễ dàng, vì các mạch thừa này đều xuất phát từ chính tiên đề, từ chính các cơ sở toán lôgic (từ đại số Boole) mà ra. Nhưng ta bắt buộc phải tìm cách loại bỏ những mạch thừa đó khỏi mạch thiết kế. Để loại bỏ được "mạch thừa" ở đây ta phải đưa ra một khái niệm toán học mới đó là : tự tạo ra, tự định nghĩa ra một phép toán mới để giải quyết nhiệm vụ kỹ thuật. Cụ thể ở đây nó là phép toán có tên là "hiệu tọa độ" ký hiệu (#).

Trước hết muốn loại bỏ được mạch thừa thì ta phải biết được vì sao thừa? và thừa vì cái gì theo toán học? chứ không thể chỉ nhìn trên bìa K và nhận xét là nó thừa là chưa đủ. Để trả lời được câu hỏi : các mạch đó thừa vì cài gì? thì ta có thể dựa vào một ví dụ đơn giản được chỉ ra trên hình 1.68.



Hình 1.68. Một hàm lôgic đơn giản.

Từ bìa K ta có phương trình của hàm lôgic đó là :

$$f_3 (x_1, x_2, x_3) = \bar{x}_1 \bar{x}_2 \bar{x}_3 + \bar{x}_1 \bar{x}_2 x_3 + x_1 \bar{x}_2 x_3 + x_1 x_2 x_3$$

$$= \bar{x}_1 \bar{x}_2 \ (\bar{x}_3 + x3) + (\bar{x}_1 + x_1)\ \bar{x}_2\ x_3 + x_1\ (\bar{x}_2 + x_2)\ x_3$$

$$= \bar{x}_1 \bar{x}_2 \qquad + \bar{x}_2\ x_3 \qquad + x_1\ x_3$$

$$= \textcircled{1} + \qquad \textcircled{2} \qquad + \qquad \textcircled{3} \quad .$$

Hàm số có 4 đỉnh ứng với 4 tích số, nhóm họp các tính số đó từng đôi một theo tiên đề (theo phép toán Dán) ta thu được các kết quả đơn giản hơn, gọi là các implicăng đơn giản (ví dụ : $\bar{x}_1 \bar{x}_2$; $\bar{x}_2\ x_3$; $x_1\ x_3$). Các implicăng đơn giản được rút gọn theo tiên đề chính lại tương ứng với từng khoanh dán trên bìa K (ví dụ : khoanh ① $= \bar{x}_1 \bar{x}_2$ hay khoanh ② $= \bar{x}_2\ x_3$ và khoanh ③ $= x_1\ x_3$).

Từ các khoanh dán thu được ta có thể minh họa hàm lôgic đó thành một đoạn thẳng mô tả trên hình 1.68b, "Đoạn thẳng" mô tả hàm số đó có hai đầu là A và B. Ta có thể gọi là hai đỉnh A và B, chúng có giá trị cụ thể là : A = (000) = $\bar{x}_1 \bar{x}_2 \bar{x}_3$ và B = (111) = $x_1\ x_2\ x_3$. Từ đây ta có một định nghĩa hay một tên gọi mới là :

Định nghĩa 1.9.2 : "Đỉnh đầu mút", "đỉnh tận cùng" "đỉnh đặc biệt", "đỉnh đánh dấu" là đỉnh chỉ nằm trong một implicăng đơn giản.

Ví dụ : Đỉnh (101) = $x_1\ \bar{x}_2\ x_3$ không phải là đỉnh tận cùng, vì đỉnh đó vừa nằm trong khoanh ② vừa nằm trong khoanh ③, tức là chúng có thể được viết :

$$x_1\ \bar{x}_2\ x_3 \ \subseteq \ \bar{x}_2\ x_3 \quad = \textcircled{2}$$

$$x_1\ \bar{x}_2\ x_3 \ \subseteq \ x_1\ x_3 = \textcircled{3}$$

Như vậy đỉnh ($x_1\ \bar{x}_2\ x_3$) nằm đồng thời ở hai implicăng (tích) đơn giản là ② và ③.

Nhưng đỉnh A = (000) = $\bar{x}_1 \bar{x}_2 \bar{x}_3$ là "đỉnh đầu mút" hay "đỉnh tận cùng", vì đỉnh đó chỉ nằm trong khoanh ① mà thôi nó không nằm trong bất cứ một khoanh hay một imphicăng đơn giản nào khác, tức là :

$$A = \bar{x}_1 \bar{x}_2 \bar{x}_3 \ \subseteq \ \bar{x}_1 \bar{x}_2 \ = \textcircled{1}$$

Tương tự như vậy đỉnh B = (111) = $x_1\ x_2\ x_3$ cũng chỉ nằm trong khoanh (3) mà thôi tức là :

$$B = x_1\ x_2\ x_3 \ \subseteq \ x_1\ x_3 = \textcircled{3}$$

Từ đỉnh đầu mút đỉnh tận cùng ta có một tên gọi mới nữa là : Định nghĩa 9.3 implicăng đơn giản chứa đỉnh đầu mút thì gọi là "khối đầu mút" (hoặc gọi extrêman)

Ví dụ : Khoanh ① là implicăng đơn giản chứa "đỉnh đầu mút" là A = ($\bar{x}_1 \bar{x}_2 \bar{x}_3$) thì nó còn được gọi là "khối đầu mút". Tương tự như vậy khoanh ③ = $x_1\ x_3$ có tên gọi là khối đầu mút là implicăng đơn giản thực sự (extrêman).

Từ kết quả có "khối đầu mút" chúng ta dễ dàng nhận thấy rằng : khi lắp mạch điện (hay tìm hàm lôgic tối thiểu hóa) nếu mạch điện là khối đầu mút là implicăng đơn giản thực sự không thể thiếu được, thì bắt buộc phải lắp ráp implicăng đơn giản đó

trong mạch (hay phải giữ lại nó trong hàm tối thiểu), vì nếu bỏ không dùng mạch điện (mạch và) là khối đầu mút thì sẽ bị mất chức năng đầu mút.

Còn nếu implicăng đơn giản như khoanh ② thì có thể không cần lắp mạch VÀ ứng với khoanh số ② = $\bar{x}_2 x_3$, mà mạch vẫn không bị thiếu chức năng hay thiếu đỉnh nào ô nào trong bìa K, vì các đỉnh của khoanh ② đã được các khoanh khác là ① và ③ (đó là các khối đầu mút các mạch VÀ ①, ③ làm thay cho cả. Như vậy implicăng đơn giản ② là một "mạch thừa" và có thể bỏ nó đi không cần lắp, nhưng chức năng của hàm lôgic vẫn đảm bảo, hay nói cách khác là chức năng đó vẫn được thực hiện đầy đủ các giá trị 1 của hàm số ở trên bìa.

Từ đó ta có thể hiểu khoanh ② là *"mạch thừa" vì nó không chứa "đỉnh đầu mút"* hay không chứa "đỉnh tận cùng". Như vậy các implicăng đơn giản không chứa đỉnh đầu mút đều là các "mạch thừa", ta có thể loại bỏ khỏi mạch chức năng thực hiện hàm lôgic.

Đến đây ta đã có những khái niệm và các tên gọi mới như sau :

- Tập các implicăng đơn giản không thể nhóm họp hay rút gọn hơn (tương ứng với các khoanh dán), được gọi là "không gian Z" các implicăng đơn giản không rút gọn.

$$Z = \bigcup_{i=1}^{n} Z_i$$

Trong đó : Z_i là implicăng đơn giản không thể rút gọn.

- Từ tập các implicăng đơn giản hay từ không gian Z, ta tìm được tập các đỉnh đầu mút hay còn được gọi là "không gian L" các đỉnh đầu mút.

$$L = \bigcup_{i=1}^{n} L_i$$

Trong đó : L_i là "đỉnh đầu mút", hay "đỉnh tận cùng".

- Từ hai tập Z các implicăng đơn giản không rút gọn được và tập L các đỉnh đầu mút, ta tìm ra được tập các khối đầu mút hay còn được gọi là "không gian E" các "khối đầu mút" (các extrêman)

$$E = \bigcup_{i=1}^{n} E_i$$

Trong đó : E_i là khối đầu mút là implicăng đơn giản thực sự.

Ví dụ trong bài toán đã chỉ ra trên hình 1.68 ta có các "không gian Z" các implicăng đơn giản, "không gian L" các đỉnh đầu mút và "không gian E" các khối đầu mút như sau :

$$Z = (①, ②, ③) = (\bar{x}_1 \bar{x}_2, \bar{x}_2 x_3, x_1 x_3)$$

$$L = (A, B) = (\bar{x}_1 \bar{x}_2 \bar{x}_3, x_1 x_2 x_3)$$

$$E = (①, ③) = (\bar{x}_1 . \bar{x}_2, x_1 . x_3)$$

Để có mạch điện đơn giản nhất thực hiện được chức năng đề ra ta phải tìm được

các "khối đầu mút" (implicăng đơn giản chứa đỉnh đầu mút), nói cách khác chính là ta phải tìm ra tập các "khối đầu mút E" của hàm lôgic đó. Từ đây ta có một danh từ mới là : Khi ta tìm mạch điện đơn giản nhất chính là tìm "phủ tối thiểu" C_{min} của hàm số (có lúc lại gọi là tìm phủ tối đại của hàm số khi xét về khía cạnh phủ được nhiều giá trị 1 trong bìa Kácnô), như vậy ta có thể nói C_{min} chính là "không gian các khối đầu mút E", hay viết :

$$C_{min} = E$$

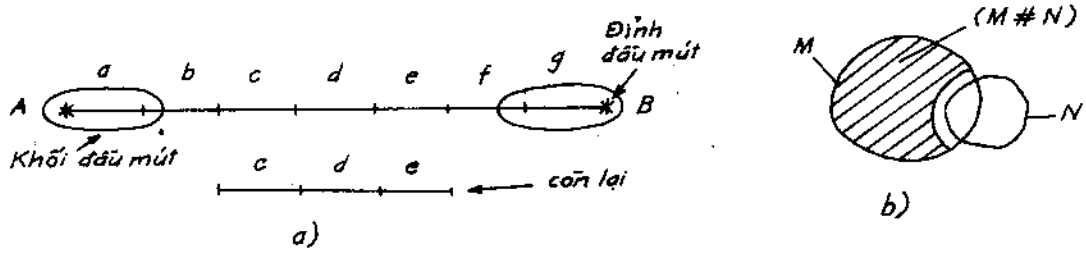
Để tìm được mạch điện đơn giản hay phủ C_{min} của hàm lôgic ta phải dựa vào không gian Z, chúng có quan hệ như sau :

$$C_{min} \subseteq Z$$

Tới đây ta chỉ còn tồn tại một vấn đề là làm sao và bằng cách nào loại trừ được các mạch thừa hay các implicăng đơn giản không có đỉnh đánh dấu (không chứa đỉnh đầu mút). Muốn loại trừ được các mạch thừa, bằng toán học đó là một điều không đơn giản, vì các mạch thừa ② (trong ví dụ trên) xuất hiệu do chính tiên đề toán học hay bản chất của đại số Boole mà ra.

1.9.4. LOẠI BỎ CÁC MẠCH THỪA DÙNG PHÉP TOÁN HIỆU TỌA ĐỘ (#)

Để loại bỏ được mạch thừa hay các implicăng đơn giản không chứa đỉnh đầu mút, ở đây ta phải tự đưa ra hay chế ra một phép toán mới có tên là phép "hiệu tọa độ" có ký hiệu là (#). Định nghĩa của phép toán hiệu tọa độ (#) được chỉ ra ở trên hình 1.69. Còn cách chế tạo cụ thể phép toán hiệu tọa độ (#) nay sẽ được chỉ ra ở phần sau trong phương pháp Roth.



Hình 1.69. Phép hiệu tọa độ (#).

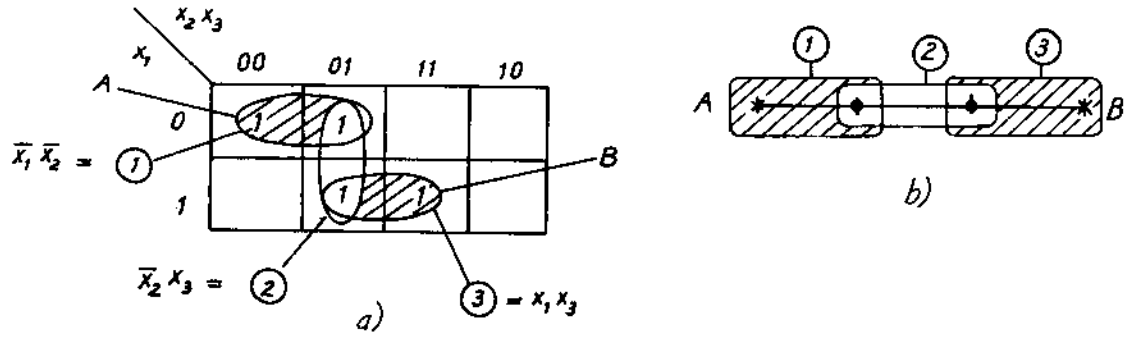
Ta có : (a, b, c, d, e, f, g) # (a, g) = (c, d, e)

Trong đó : (a, b, c, d, e, f, g) là các implicăng đơn giản không rút gọn được $\in Z$

(a, g) là các khối đầu mút $\in E$

Ở đây giả sử chúng ta công nhận có sự tồn tại của phép toán hiệu tọa độ (#) theo định nghĩa để hiểu cách dùng phép toán hiệu tọa độ (#) loại trừ hay bỏ đi các mạch thừa. Việc xây dựng nên phép toán hiệu tọa độ một cách cụ thể hơn sẽ được chỉ ra ở phần sau. Từ đây ta có một khái niệm mới quan trọng là : để giải quyết một nhiệm vụ kỹ thuật cụ thể ta có thể tự đưa ra hay tự định nghĩa ra một phép toán mới, phép toán mới này được chế tạo ra chỉ có thể giải quyết nhiệm vụ kỹ thuật theo toán học.

Việc ứng dụng phép toán hiệu tọa độ (#) để loại bỏ đi các implicăng đơn giản thừa có thể được thực hiện như sau. Trở lại bài toán quen thuộc đã đưa ra ở phần trên ở hình 1.70.



Hình 1.70. Bài toán đơn giản.

Ta đã có kết quả :

$$Z = (①, ②, ③) = (\bar{x}_1\bar{x}_2,\ \bar{x}_2 x_3,\ x_1x_3)$$

$$E = (①, ③) = (\bar{x}_1\bar{x}_2,\ x_1x_3)$$

Áp dụng phép toán hiệu tọa độ ta có kết quả :

$$Z\ \#\ E = (①, ②, ③)\ \#\ (①, ③) = \Phi$$

$$= (\bar{x}_1\bar{x}_2\ x_3 + \bar{x}_2 x_3 + x_1\ x_3)\ \#\ (\bar{x}_1\bar{x}_2 + x_1\ x_3) = \Phi$$

Khi xuất hiện kết quả là rỗng (Φ) thì ta có phủ tối thiểu C_{min} hay hàm lôgic rút gọn là :

$$f_{min}(x_1,x_2,x_3) = C_{min} = \bar{x}_1\bar{x}_2 + x_1x_3$$

Như vậy có thể nói hàm lôgic rút gọn hay phủ tối thiểu C_{min} được tìm ra từ không gian các implicăng đơn giản Z. Đồng thời qua đó cũng hiểu được ứng dụng của phép toán hiệu tọa độ (#).

Trong thực tế nếu hàm lôgic phức tạp hơn, việc tìm được không gian các khối đầu mút E sẽ không thể đơn giản và chỉ làm một lần như vậy. Nhờ có phép toán hiệu tọa độ (#), quá trình tìm ra tập E của một hàm lôgic phức tạp, hay còn gọi thuật toán (các bước) tìm không gian E các implicăng đơn giản không thể thiếu được, có thể thực hiện như sau trên hình 1.71.

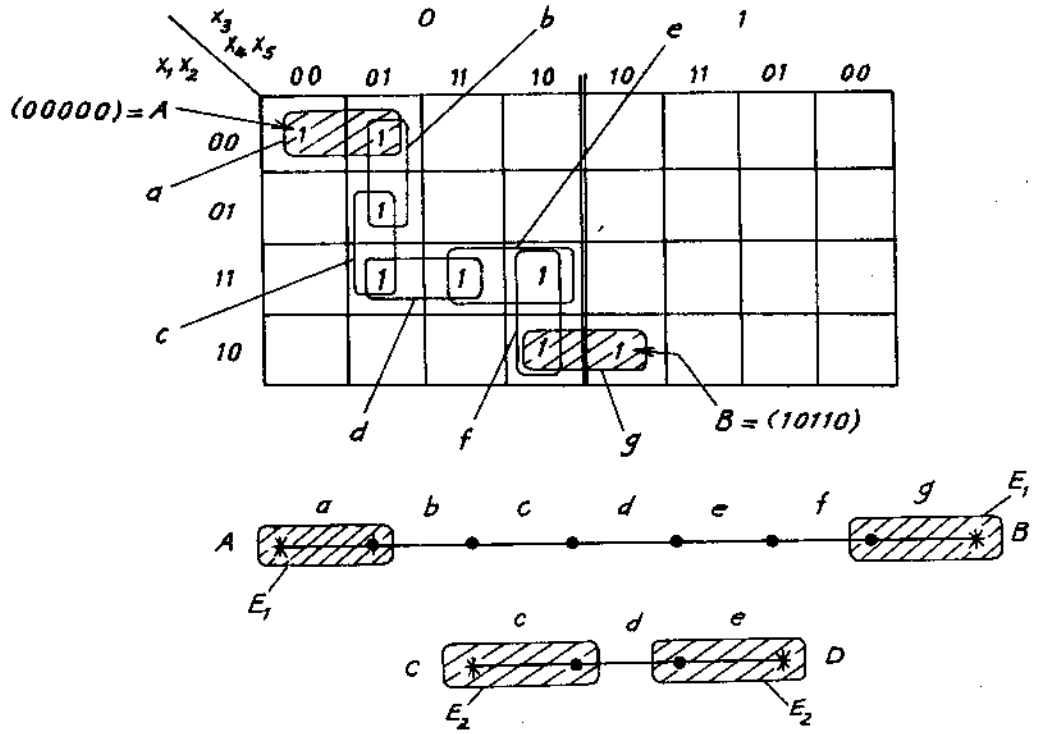
Từ bài toán hàm lôgic 5 biến đơn giản được đưa ra làm ví dụ ta có các kết quả sau :

$$Z = (a,\ b,\ c,\ d,\ e,\ f,\ g)$$

là tập các implicăng đơn giản không rút gọn được. Đồng thời cũng có tập các đỉnh đầu mút L_1 là :

$$L_1 = (A,\ B) \qquad A = (00000)$$

$$B = (10110)$$



Hình 1.71. Các bước tìm không gian E.

Từ tập Z và tập L_1 ta tìm được tập các khối đầu mút đợt một là E_1 có kết quả là :

$$E_1 = (a, g)$$

Sau khi có E_1 để dễ dàng ký hiệu ta coi tập $Z = Z_1$ là tập ban đầu. Từ Z_1 và E_1 dùng phép hiệu tọa độ (#) tìm được không gian mới

$$Z_2 = Z_1 \# E_1$$

$$= (a, b, c, d, e, f, g) \# (a, g) = (a, d, c)$$

Từ tập $Z_2 = (c, d, e)$ sẽ tìm được tập các đỉnh đầu mút L_2 là :

$$L_2 = (C, D) \qquad C = (01001)$$

$$D = (11010)$$

Từ hai tập Z_2 và L_2 ta lại tìm được tập các khối đầu mút lần hai là E_2 có kết quả là

$$E_2 = (c, e)$$

Sau đó lại thực hiện tìm không gian các implicăng đơn giản mới Z_3

$$Z_3 = Z_2 \# E_2$$

ở đây : $Z_3 = (c, d, e) \# (c, e) = \Phi$ - rỗng

Chú ý : Quá trình tìm các không gian các implicăng đơn giản mới cứ làm tương tự mãi cho đến khi $Z_i \# E_i = \Phi$ thì dừng quá trình tìm khối đầu mút, kết quả nhận được sẽ là

$$C_{min} = E = E_1 + E_2 = \bigcup_{i}^{n} E_i$$

$$= (a, g) + (c, e)$$

Kết quả thu được sẽ là tập các khối đầu mút

$$f_{5min} = C_{min} = E = (a, g, c, e)$$

Đó chính là phủ tối thiểu hay là mạch điện đơn giản nhất của hàm lôgic đã cho.

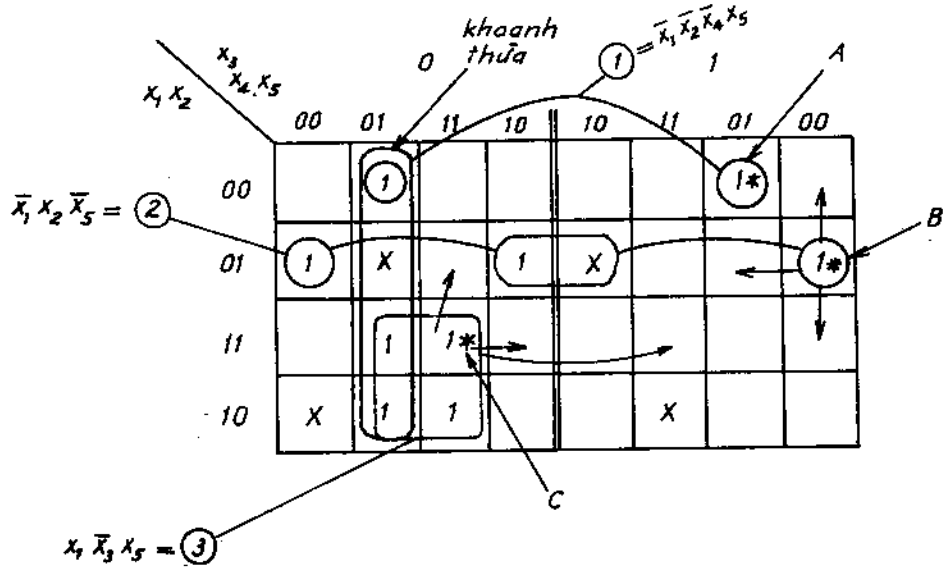
Tổng quát quá trình tìm các khối đầu mút hay các implicăng đơn giản không thể thiếu được E :

$$E = \bigcup_{i=1}^{n} E_i$$

Trong đó : E_i là tập các khối đầu mút trong quá trình tìm.

Tới đây trong quá trình rút gọn hàm Boole trên bìa Kácnô ta cần phải có hai kiến thức quan trọng đó là : đỉnh tận cùng hay đỉnh đánh dấu (nó cũng chính là chức năng không thể thiếu) và khoanh dán to nhất để mạch thực hiện được đơn giản nhất. Phải đầy đủ cả hai điều kiện trên thì một khoanh dán mới có giá trị hay được công nhận (được tồn tại ở $f_{min}(X)$), vì nếu khoanh to nhất ứng với implicăng đơn giản không thể rút gọn được, mà trong khoanh đó không chứa đỉnh đầu mút thì đó chỉ là "khoanh thừa" hay "mạch thừa" mà thôi.

Vấn đề còn lại ở đây là cách tìm ra được "đỉnh đầu mút" hay đỉnh tận cùng trong một bài toán cụ thể rút gọn trên bìa Kácnô. Để hiểu được điều đó ta làm ví dụ rút gọn một hàm lôgic năm biến biểu diễn trên hình 1.72.



Hình 1.72. Cách tìm đỉnh đầu mút.

Cách tìm "đỉnh đầu mút" hay "đỉnh đánh dấu" cụ thể như sau :

Ta "khoanh thử" một khoanh to nhất có thể được trên bìa Kácnô sau đó đi kiểm tra lần lượt từng đỉnh (ô) có giá trị 1 ở trong "khoanh thử" đó xem nó có "cạnh" hay "kế cận" một ô có giá trị 1 ở ngoài "khoanh thử" đó hay không. Nếu tồn tại chỉ 1 ô có giá trị 1 ở ngoài "khoanh thử" kế cận ô có giá trị 1 ở trong "khoanh thử" thì ô có giá trị là 1 ở trong "khoanh thử" không phải là đỉnh đầu mút hay đỉnh đánh dấu.

Ví dụ : ô có giá trị 1 ứng với tọa độ (00001) trong "khoanh thừa", ô đó nằm trong "khoanh thừa" lại "cạnh" ô (00101) nằm bên ngoài "khoanh thừa" có giá trị 1. Vậy ta kết luận ô (00001) không phải là đỉnh đầu mút. Nếu các đỉnh 1 ở trong khoanh đó đều không phải là đỉnh đầu mút thì khoanh đó là khoanh thừa.

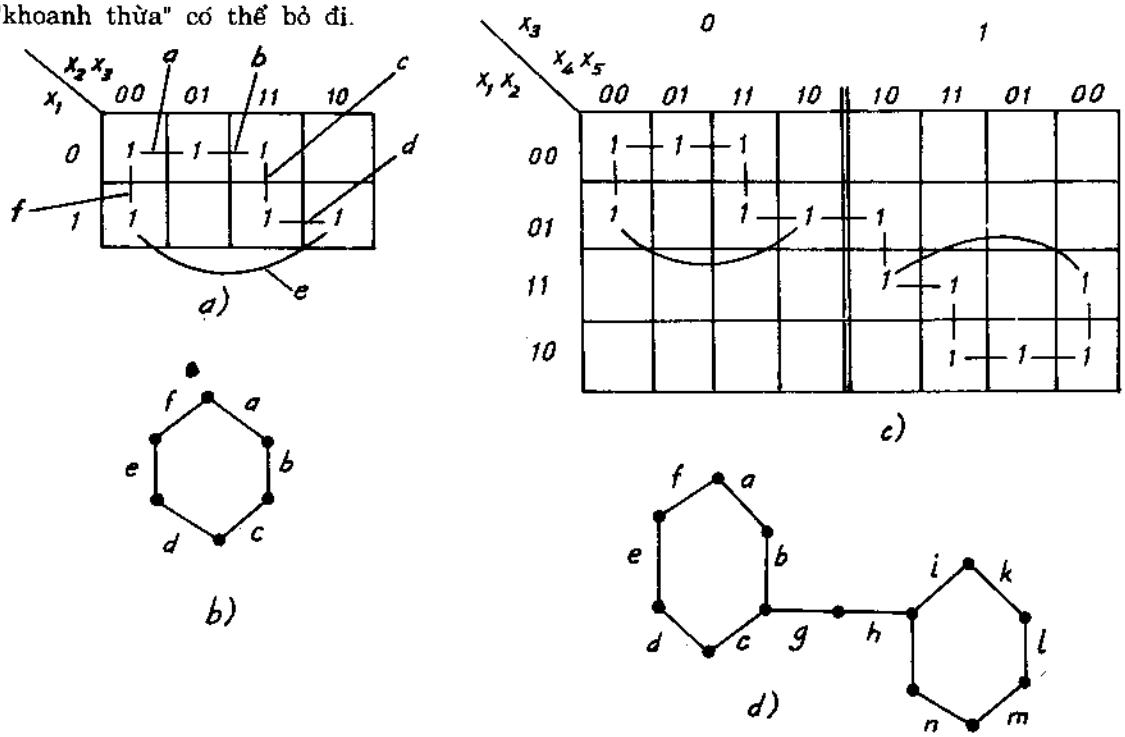
Nhưng nếu có giá trị 1 ở trong khoanh thử khi ta kiểm tra thấy không "kế cận" bất kỳ một ô có giá trị 1 nằm ngoài khoanh thử, thì ô có giá trị 1 ở trong "khoanh thử' ta kiểm tra đó được gọi là "đỉnh đầu mút", đồng thời khi có đỉnh đầu mút thì "khoanh thử" đó trở thành "khối đầu mút" mà ta phải giữ lại khoanh đó.

Ví dụ : khoanh thử ② = $\bar{x}_1\ x_2\ \bar{x}_5$ ta kiểm tra đỉnh B = (01100) nằm trong "khoanh thử" ②. Ta thấy đỉnh B đó nó không "cạnh" hay "kế cận" với bất kỳ một ô có giá trị 1 nào nằm ngoài khoanh thử ②, vì các ô kế cận đỉnh B theo mũi tên kiểm tra đều có giá trị là 0. Từ sự kiểm tra đó ta kết luận đỉnh B là đỉnh "đầu mút", "đỉnh đặc biệt" "đỉnh đánh dấu" và ta đánh dấu (*) vào đỉnh B đó, đồng thời giữ lại khoanh thử ② vì nó trở thành "khối đầu mút" thuộc không gian E.

Với cách làm tương tự cho các khoanh dán khác mà ta có thể khoanh trên bìa Kácnô. Với ví dụ đã cho ta tìm được ba "đỉnh đầu mút" hay ba "đỉnh đánh dấu" là các đỉnh (A, B, C). Từ ba đỉnh đầu mút đó kèm theo ta có ba khối đầu mút đó là các khoanh (①, ②, ③). Từ đó ta có kết quả rút gọn của hàm Boole đã cho tìm bìa K là.

$$C_{min} = f_{min}(x_1, x_2, x_3, x_4, x_5) = \bar{x}_1\bar{x}_2\bar{x}_4x_5 + \bar{x}_1x_2\bar{x}_5 + x_1\bar{x}_3x_5$$

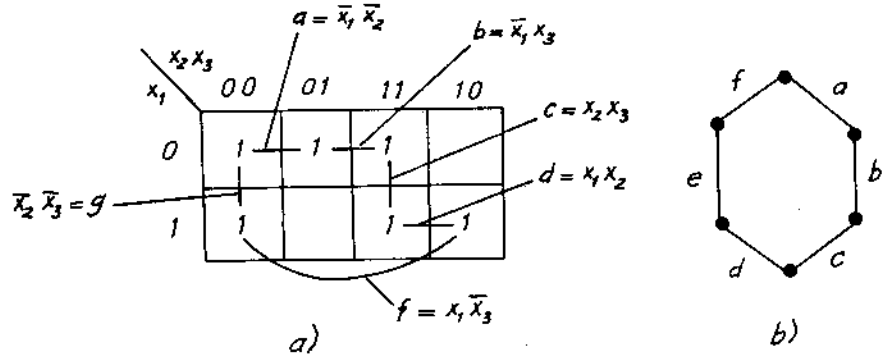
Tóm lại quá trình thực hiện rút gọn hàm Boole dùng bìa Kácnô là ; khoanh to nhất, sau đó kiểm tra xem khoanh đó có "đỉnh đánh dấu" "đỉnh đầu mút" hay không. Nếu có đỉnh đầu mút thì giữ lại khoanh đó còn nếu không có đỉnh đầu mút thì đó là "khoanh thừa" có thể bỏ đi.



Hình 1.73. Loại bài toán lôgic không có đỉnh đầu mút.

Nhưng trong thực tế việc rút gọn hàm Boole không chỉ đơn giản như vậy, vì có thể xuất hiện loại hàm lôgic, hàm Boole không có 'đỉnh tận cùng" hay không có "đỉnh đầu mút". Đây là loại bài toán đặc biệt của hàm Boole, nó liên quan tới cấu trúc của tự nhiên như hàm lôgic có cấu tạo kiểu mạng tổ ong hay mạng giao thông, mạng thần kinh con người... Với loại mạch lôgic có cấu trúc tổ chức như vậy, chỉ có thể dựa vào bìa Kácnô ta mới hình dung được cấu hình của nó, nếu chỉ biến đổi hay rút gọn theo toán học - theo đại số, thì ngoài kết quả rút gọn phủ C_{min} ra, ta không có được thêm thông tin gì khác, và không hình dung ra được cấu hình của bài toán là gì. Có thể lấy ví dụ minh họa bài toán lôgic "không có đỉnh đầu mút" chỉ ra trên hình 1.73.

Để giải loại bài toán lôgic "không có đỉnh đầu mút' cần phải dùng tới phép toán hiệu tọa độ (#), cách giải loại bài toán loại này ta có thể minh họa bằng ví dụ chỉ ra trên hình 1.74, tìm phủ C_{min} hay rút gọn hàm số sau.



Hình 1.74. Một bài toán cụ thể.

Từ bìa Cácnâu ta có tập các implicăng đơn giản không rút gọn được Z là

$$Z = (a, b, c, d, e, f) = Z_1$$

$$L = ( ) = \Phi - \text{rỗng} = L_1$$

Nhận thấy các implicăng đơn giản đồng đều nhau về mức độ rút gọn (chúng giống hệt nhau), từ đó ta có thể chọn bất kỳ trong chúng ra một implicăng đơn giản làm khối đầu mút E_1, ví dụ ta chọn :

$$E_1 = (b)$$

Sau khi có E_1 ta tìm tiếp không gian Z_2 các ienphicăng đơn giản :

$$Z_2 = Z_1 \# E_1$$

$$= (a, b, c, d, e, f) \# (b) = (f, e, d)$$

Có minh họa quá trình Z_2 như sau trên hình 1.75.

Sau khi dùng phép toán hiệu tọa độ (#) "chặt đứt" vòng, ta lại thấy xuất hiện "đỉnh đầu mút" (A, B), và tất nhiên ta lại có khối đầu mút và cách giải bài toán lại trở lại cách làm ta đã nêu ở phần trên.

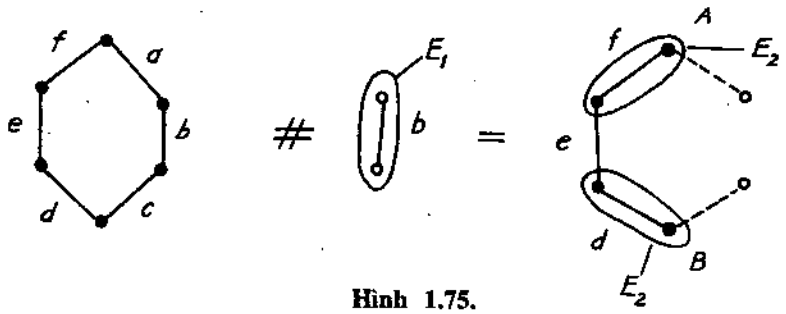
Tới đây ta có kết quả

$$Z_2 = (f, e, d)$$

$$L_2 = (A, B)$$

Từ hai tập Z_2 và L_2 là đỉnh đầu mút ta tìm được khối đầu mút E_2 có kết quả là :

$$E_2 = (f, d)$$



Hình 1.75.

Ta tính tiếp thấy :

$$Z_3 = Z_2 \# E_2$$

$$= (f, e, d) \# (f, d) = \Phi$$

Khi xuất hiện kết quả $Z_{i+1} = Z_i \# E_i = \Phi$ thì ta dừng kết quả tính toán, và cộng các kết quả các khối đầu mút lại :

$$C_{min} = E = E_1 + E_2 = (b) + (f, d)$$

Đó chính là kết quả rút gọn của hàm lôgic không có đỉnh đầu mút đã cho.

$$f_{min}(x_1, x_2, x_3) = \bar{x}_2 x_3 + x_1 x_2 + \bar{x}_2 \bar{x}_3.$$

Ở đây chỉ đưa ra cách sử dụng phép toán hiệu tọa độ (#) để giải một bài toán lôgic đơn giản không có đỉnh tận cùng (đỉnh đầu mút), còn trong thực tế của hàm lôgic khi liên quan tới "cấu trúc của tự nhiên" thì bài toán thực hiện sẽ phức tạp hơn rất nhiều , sự phức tạp không phải vì bài toán to lớn mà vì nhát "chặt" hay "cắt" đầu tiên của phép toán hiệu tọa độ (#) sẽ được đặt vào đâu? Ở điểm nào để sau này ta sẽ nhận được phủ tối thiểu hay hàm lôgic đơn giản nhất của bài toán đó. Trong quyển sách này không nêu hết được các vấn đề phức tạp xảy ra trong quá trình rút gọn hàm Boole. Đồng thời ngoài phủ tối thiểu C_{min}, nếu có được các thông tin khác nữa của hàm lôgic (của mạch điện), thì ta có thể đánh giá được độ tin cậy cũng như chất lượng của nó. Tới đây cũng kết thúc quá trình rút gọn hàm Boole dùng bìa Kácnô.

Dùng bìa K rút gọn hàm lôgic nhanh, dễ dàng và thuận tiện, đồng thời cho ta hình dung được hình dáng của hàm lôgic đó cũng như độ phức tạp của nó. Nhưng bìa Kácnô cũng có những hạn chế rất lớn, đó là không thể giải được các bài toán lôgic có nhiều biến số, vì khi quá "nhiều biến số" hay "nhiều tín hiệu vào" làm cho bìa Kácnô sẽ rất lớn và không thể vẽ nổi. Thông thường chỉ dùng bìa Kácnô giải bài toán lôgic có số biến $\leqslant 7$. Như vậy khi cần rút gọn những bài toán có số lượng biến rất lớn, khoảng

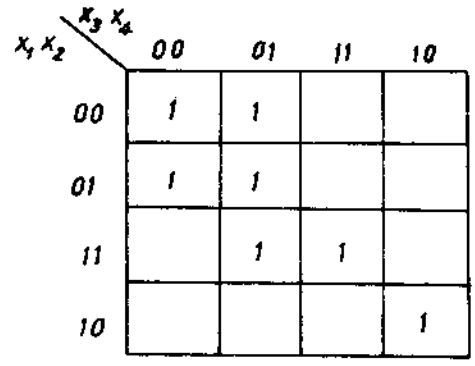
hàng trăm biến hàng trăm tín hiệu vào thì người ta phải dùng phương pháp khác, để có thể lập trình cho máy tính để giải, người ta gọi đó là phương pháp thuật toán hay phương pháp Mc.Cluskey, phương pháp này được trình bầy ở phần tiếp theo.

1.9.5. RÚT GỌN HÀM BOOLE THEO PHƯƠNG PHÁP QUINÉA - Mc.CLUSKEY

Đây là một phương pháp rút gọn hàm Boole dùng thuật toán, nhờ đó ta có thể lập trình cho máy tính để giải được bài toán với số lượng biến số rất lớn tới hàng trăm biến. Ngày nay bằng máy tính người ta đã giải được những bài toán lôgic có số lượng biến số tới trên năm trăm biến. Người ta đã xây dựng được một chương trình tự động rút gọn hàm Boole được viết bằng C++, chương trình này của Mỹ, bắt đầu xây dựng từ những năm 1980, tại trường đại học Califonia của Berkeley, nó có tên là ESPRESSO. Đến nay chương trình ESPRESSO dùng để tự động rút gọn hàm Boole đã được xây dựng hoàn chỉnh, đã được sử dụng rất phổ biến ở nhiều nước, chương trình ESPRESSO có hiệu quả tính toán rất lớn dựa trên khả năng sử dụng hàng loạt các thuật toán cũng như phương pháp tính toán thông minh. Chương trình này trước hết để sử dụng tối thiểu hóa cấu trúc của PLA (Program Logic Array) trong máy tính, đồng thời có thể thực hiện rút gọn các hàm lôgic bất kỳ.

Phương pháp Quinéa–Mc.Cluskey gồm hai quá trình : trước tiên là quá trình tìm "không gian Z" là tập các implicăng đơn giản không rút gọn được, tiếp theo là quá trình xác định tìm các implicăng thật sự không thể thiếu được - không gian các khối đầu mút, đồng thời loại bỏ đi các implicăng đơn giản thừa. Cả hai quá trình tìm kiếm đó có thể dùng bảng để thực hiện như sau :

Ví dụ : Dùng phương pháp Quinéa–Mc.Cluskey rút gọn hàm Boole cho trên bìa Kácnô ở hình 1.76.



Hình 1.76. Một hàm lôgic bất kỳ.

Ta có một số ký hiệu và tên gọi :

K^0 Gọi là (khối - 0) là những implicăng căng đầy đủ
Ví dụ (00110) ; (10100)... là các (khối - 0)

K^1 Gọi là (khối - 1) là những tích số có một (X).
Ví dụ (101X0) ; (11X00)... là các (khối - 1)

K^2 Gọi là (khối - 2) là những tích số có hai (X).
Ví dụ (00X0X) ; (1X10X)... là các (khối - 2).

Tổng quát K^r là (khối - r) trong đó xuất hiện r ký hiệu (X)

Cách thực hiện rút gọn hàm lôgic theo Mc.Cluskey trước hết là quá trình tìm không

gian Z các implicăng đơn giản. Ta kẻ một bảng được chỉ ra trên hình 1.77 trong đó có các cột ứng với các khối, ta kẻ từ cột K^0, K^1... tới K^r. Trong cột đầu tiên K^0 ta ghi tất cả các (khối − 0) của hàm lôgic cần rút gọn vào đó (các ô hay các đích của hàm lôgic), tập các (khối − 0) còn được gọi là "phủ ban đầu" C_0. Từ bài toán đã cho có

$$C_0 = ((0000)\ (0001)\ (0100)\ (0101)\ (1101)\ (1111)\ (1010))$$

K^0	K^1	K^2	K^3
0 0 0 0 V	0 0 0 X V	0 X 0 X	
0 0 0 1 V	0 X 0 0 V		
0 1 0 0 V	0 X 0 1 V		
0 1 0 1 V	0 1 0 X V		
1 1 0 1 V	X 1 0 1		
1 0 1 0	1 1 X 1		

Hình 1.77. Bảng tìm các khối.

1.9.5.1. Quá trình tìm không gian Z các implicăng đơn giản

Sau khi ghi hết K^0, "phủ ban đầu" hay số liệu ban đầu của hàm lôgic cần rút gọn. Thực hiện phép gom giới hạn giữa các (khối −0) với nhau, chính là phép so sánh giữa các (khối −0) với nhau, nếu giữa hai (khối −0) chúng khác nhau chỉ ở một vị trí thì ghi kết quả thành (khối −1) vào cột bên cạnh và đánh dấu (V) ở hai (khối −0) đó, chứng tỏ hai khối này đã được thay thế bởi một (khối −1) ở cột bên cạnh. Còn nếu giữa hai (khối −0) thấy vị trí khác nhau của chúng có số lượng ⩾ 2 thì bỏ qua.

Ví dụ : Lấy (0000) so với (0001) thì kết quả được (khối −1) là (000X) và ta đánh dấu dấu (V) vào đằng sau hai (khối −0) đó. Tương tự ta so sánh tiếp :

(0000) so với (0100) được K^1 là (0X00)

(0000) so với (0101)

thấy vị trí khác nhau ⩾ 2 ta bỏ qua và lại gom giới hạn tiếp cho đến hết dãy K^0.

Tiếp theo lấy (0001) so sánh lần lượt tới hết các (khối −0) ở cột K^o nếu có kết quả thì đánh dấu (V) và ghi sang K^1 ở cột bên cạnh.

Ta cứ làm như vậy cho tới hết cột K^0 sao cho không bỏ sót một cặp so sánh (khối −0) nào.

Sau khi tuần tự gom hết (so sánh) các (khối −0) trong cột K^0, và cột K^1 cũng ghi đầy đủ kết quả của quá trình so sánh, kiểm tra lại toàn bộ cột K^0, những (khối −0) nào không gom được (không có dấu V ở bên cạnh) tức là đỉnh đó độc lập không nhóm họp được với các đỉnh khác, thì đó là những implicăng đơn giản (khối −0) thuộc không gian Z^0 .

$$Z^0 = ((1010))$$

Khi đã có kết quả của cột K^1 ta lại coi cột K^1 đó là số liệu đầu vào để gom giới hạn hay so sánh các cặp (khối -1) với nhau. Nếu so sánh thấy khác nhau chỉ một vị trí thì ghi kết quả vào cột K^2 và đánh dấu (V) vào vị trí các (khối -1) đã được gom. Còn nếu số lượng vị trí khi so sánh ≥ 2 thì bỏ qua.

Ví dụ : Lấy (000 X) so sánh với (0 1 0 X) cho kết quả (khối -2) là (0 X 0 X) ghi vào cột K^2.

Sau khi tuần tự so sánh các (khối -1) trong cột K^1 và cột K^2 cũng ghi đầy đủ kết quả quá trình so sánh. Ta kiểm tra lại toàn bộ cột K^1, những (khối -1) nào không có dấu (V) ở bên cạnh chứng tỏ chúng không được gom (nhóm họp thành khối lớn hơn) thành K^2, thì đó là các implicăng đơn giản không rút gọn được thuộc không gian Z^1

$$Z^2 = ((X\ 101)\ (11\ X\ 1))$$

Khi có kết quả của cột K^2 ta lại gom tiếp, cách làm tương tự mãi cho đến khi gặp cột $K^r = \Phi$. Thì dừng quá trình tìm không Z. Với bài toán đã cho ta thấy $K^3 = \Phi$ vậy implicăng đơn giản thuộc K^2 không thể nhóm thành khối lớn hơn vậy nó thuộc không gian Z^2.

$$Z^2 = ((0\ X\ 0\ X))$$

Kết thúc quá trình tìm không gian các implicăng đơn giản Z ta sẽ có kết quả là tập hợp tất cả các khối không có dấu (V) trong các cột của bảng

$$A = \bigcup_{i}^{n} Z^i$$

Trong đó Z^i là tập hợp các implicăng đơn giản (khối $-i$) những khối không có dấu (V).

Với bài toán đã cho ta có không gian Z là :

$$Z = Z^0 + Z^1 + Z^2$$

$$= ((1010)) + ((X101)\ \ (11X1)) + ((0\ X\ 0\ X))$$

$$Z = ((1010)\ (X101)\ (11\ X\ 1)\ (0\ X\ 0\ X))$$

Nhận thấy quá trình tìm không gian Z các implicăng đơn giản không rút gọn được dựa trên thuật toán so sánh (gom) giữa các implicăng với nhau lần lượt từ tập các (khối -0), tập (khối -1) ... cho tới hết, điều đó rất thuận lợi cho việc lập trình cho máy tính, cho phép tự động hóa quá trình tìm không gian Z.

1.9.5.2. Quá trình tìm không gian các khối đầu mút E

Sau khi đã có tập implicăng đơn giản Z, ta coi đó là số liệu đầu vào để từ nó tìm ra tập các "khối đầu mút E" các implicăng đơn giản không thể thiếu được như sau :

Từ bài toán đề ra ta có các số liệu :

$$Z_1 = ((1010)\ (X\ 101)\ (11\ X\ 1)\ (0\ X\ 0\ X))$$

$$L_1 = C_0 = ((0000)\ (0001)\ (0100)\ (0101)\ (1101)\ (1111)\ (1010))$$

Kẻ một bảng có hàng ứng với không gian Z_1 còn cột thì cứng với các đỉnh L_1 biểu diễn trên hình 1.78.

Z_1 \ L_1	0000	0001	0100	0101	1101	1111	1010
1010							*
X101				*	*		
11X1					*	*	
0X0X	*	*	*	*			

Hình 1.78. Bảng tìm khối đầu mút E.

Sau đó so sánh giữa hàng và cột (i, j) nếu và chỉ nếu implicăng hàng thứ i phủ (chứa) đỉnh của cột thứ j thì đánh dấu (*) vào trục tọa độ đó, còn không chứa thì thôi.

Ví dụ so sánh tọa độ ((X101) ; (0101)) ta thấy (0101) $\subseteq$ (X101) như vậy có thể nói implicăng đơn giản (X101) chứa đỉnh (0101), kết quả là đánh dấu (*) vào tọa độ đó.

Cách làm tương tự với các tọa độ khác của bảng khi đánh dấu các tọa độ xong chúng ta có nhận xét là có cột (đỉnh) chỉ có một ký hiệu (*) và có cột có hai (có thể có số lượng nhiều hơn) ký hiệu (*). qua nhận xét đó có thể hiểu được là : cột có hai dấu (*) nghĩa là đỉnh ứng với cột đó đồng thời nằm trong hai implicăng đơn giản, kết luận đỉnh đó không phải là đỉnh đầu mút (là đỉnh chỉ nằm trong một implicăng đơn giản)

Như vậy những đỉnh chỉ có một dấu (*) là những đỉnh đầu mút. Ví dụ đỉnh (1111) hay (1010) hoặc (0000) là các đỉnh "đầu mút". Vậy những implicăng đơn giản mà có chứa "đỉnh đầu mút" thì đó chính là các "khối đầu mút", chúng thuộc không gian E_1 là không gian các implicăng đơn giản không thể thiếu được. Từ đó ta có kết quả :

$$E_1 = ((1010)\ (11X1)\ (0X0X))$$

Sau khi có E_1 ta tiếp tục tìm không gian Z_2 :

$$Z_2 = Z_1 \ \# \ E_1$$

$$L_2 = L_1 \ \# \ E_1$$

Từ kết quả Z_2 và L_2 ta lại kẻ tiếp một bảng với hàng là tập không gian Z_2 và cột là tập các đỉnh L_2. Tiếp tục so sánh tọa độ giữa hàng và cột của bảng này để tìm ra E_2.

Quá trình cứ tiếp tục mãi cho đến khi :

$$Z_{i+1} = Z_i \ \# \ E_i = \Phi$$

$$L_{i+1} = L_i \ \# \ E_i = \Phi$$

thì dừng kết quả tính toán, và ta thu được kết quả cuối cùng phủ tối thiểu C_{min} là :

$$C_{min} = E = \bigcup_{i=1}^{n} E_i$$

trong đó : E_i là tập các khối đầu mút của quá trình tìm kiếm thứ i.

Với bài toán đề ra ta tính tiếp như sau :

$$Z_2 = Z_1 \# E_1$$

$$= ((1010)(X\,101)(11\,X\,1)(0\,X\,0\,X)) \# (1010)(11\,X\,1)(0\,X\,0\,X)) = \Phi$$

Vậy $Z_2 = \Phi$ ta dừng kết quả tính toán, và thu được kết quả cuối cùng của phủ tối thiểu C_{min} là :

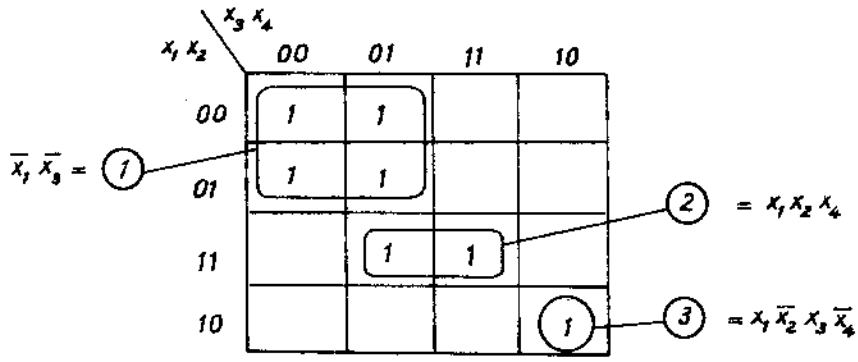
$$C_{min} = E = E_1 + \Phi$$

$$= ((1010)\ (11\,X\,1)\ (0\,X\,0\,X))$$

Vậy

$$f_{min}(x_1, x_2, x_3) = x_1\,\bar{x}_2\,x_3\,\bar{x}_4 + x_1\,x_2\,x_4 + \bar{x}_1\,\bar{x}_3$$

Theo phương pháp gom so sánh của Quinéa–Mc.Cluskey ta cũng thu được kết quả rút gọn của hàm lôgic đã đề ra. Kết quả phủ C_{min} đó dễ dàng nhận thấy trên bìa Kácnô ở hình 1.79 sau.



Hình 1.79. Minh họa kết quả rút gọn.

Thuật toán tìm phủ C_{min} của Mc.Cluskey rất đơn giản, và dễ dàng lập trình trên máy tính. Cho phép ta tự động hóa quá trình rút gọn hàm Boole, và cho khả năng giải hay rút gọn được những hàm lôgic có rất nhiều biến trong thực tế, làm cho các hệ thống chức năng kỹ thuật thực hiện được đơn giản, và có độ tin cậy cao trong quá trình hoạt động. Trong quyển sách này chỉ nêu sơ qua phương pháp rút gọn hàm Boole của Quinéa–Mc.Cluskey.

1.9.6. RÚT GỌN HÀM BOOLE THEO PHƯƠNG PHÁP ROTH

Phần này sẽ đưa ra một phương pháp nữa dùng để rút gọn hàm Boole, hay tìm phủ tối thiểu C_{min} cho hàm lôgic do Roth đưa ra, được gọi là phương pháp Roth. Nhưng phần này ngoài việc đưa ra thuật toán của Roth để tìm phủ tối thiểu cho hàm lôgic ra, thì mục đích chính của phần này là đưa ra khái niệm cũng như phương pháp tự định nghĩa lấy phép toán, hay tự chế tạo ra phép toán mới, nhằm giải quyết một nhiệm vụ kỹ

thuật được đề ra. Đồng thời dựa vào phương pháp của Roth để rút gọn hàm Boole, ta cũng sẽ hiểu được sự ứng dụng trong tính toán của một phép toán mới khi được tạo ra như thế nào. Phương pháp Roth dùng để rút gọn hàm Boole cũng phải qua các bước là tìm không gian các implicăng đơn giản Z, sau đó là tìm không gian các khối đầu mút E là các implicăng không thể thiếu được

Bước 1 : Tìm không gian các implicăng đơn giản Z thông thường hàm lôgic thường cho dưới dạng tuyển chuẩn tắc tức là tổng của các tích, thường dưới dạng phủ C_0 ban đầu gồm các (khối −0). Từ tập các (khối −0) là số liệu đầu vào C_0 cho trước ta phải gom giới hạn tất cả các cặp (khối −0) có thể tạo ra các (khối −1), từ đó lập ra tập không gian các (khối −1). Còn những đỉnh (khối −0) không gom (rút gọn) được thành (khối −1) thì được gọi là các implicăng đơn giản (khối −0) (các đỉnh độc lập) tạo ra tập Z^0 các implicăng đơn giản (khối −0).

Sau đó từ tập các (khối −1) gọi là phủ C_1 ta lại gom các cặp (khối −1) lại để tạo thành (khối −2). Còn những (khối −1) không đơn giản được thì được gọi là các implicăng đơn giản (khối −1) (các (khối −1) không rút gọn được) tạo ra tập Z^1 các implicăng đơn giản (khối −1).

Rồi lại tiếp tục từ tập các(khối −2)gọi là phủ C_2 lại gom các (khối −2) lại với nhau thành (khối −3) và tìm được tập Z_2 các implicăng đơn giản (khối −2).

Quá trình cứ tiếp tục mãi cho đến được tập X^r các implicăng đơn giản và tiếp theo tập $Z^{r+1} = \Phi$ thì dừng quá trình tìm không gian Z và kết quả:

$$Z = \bigcup_{i=0}^{r} Z^i$$

Trong đó Z^i là tập các implicăng đơn giản(khối −i.)

Nhận thấy quá trình tìm không gian Z là một quá trình biến đổi từ các khối nhỏ (khối −0) tạo dần thành các khối lớn hơn (khối −1) rồi từ (khối −1) lại tạo thành các khối lớn hơn (khối −2), rồi từ (khối −2) tạo thành (khối −3) v.v... cứ tạo ra các khối lớn mãi từ tập số liệu ban đầu.

Để tạo ra được những khối lớn hơn từ những khối nhỏ hơn, người ta đã xây dựng nên phép toán mới đầu tiên có tên gọi là "phép toán sao tọa độ" có ký hiệu là (*), được định nghĩa (hay tạo ra) như sau.

Định nghĩa 1.9.3: Phép toán sao tọa độ (*)

Cho A và b là hai implicăng bất kỳ có ký hiệu là :

$$A = (a_1, a_2, ..., a_n)$$

$$B = (b_1, b_2, ..., b_n)$$

Với một bảng quan hệ ký hiệu được biểu diễn trên hình 1.80.

Còn điều kiện thực hiện quan hệ ký hiệu được quy định như sau :

$$A * B = \begin{cases} \Phi \text{ nếu } (a_i * b_i) = Y \text{ với số lượng Y xuất hiện} \geq 2 \text{ lần.} \\ \left(m(a_1 * b_1),\ m(a_2 * b_2),\ \ldots,\ m(a_n * b_n)\right) \text{ nếu không có Y, hay chỉ có một Y xuất hiện} \\ \text{với : } m(0) = 0\ ;\ m(1) = 1 \\ \quad m(Y) = m(X) = X \end{cases}$$

a_i \ b_i	*	0	1	X
0	0	0	Y	0
1	Y	1	1	1
X	0	1	X	X

Hình 1.80. Phép toán sao tọa độ.

Có các ví dụ tính toán :

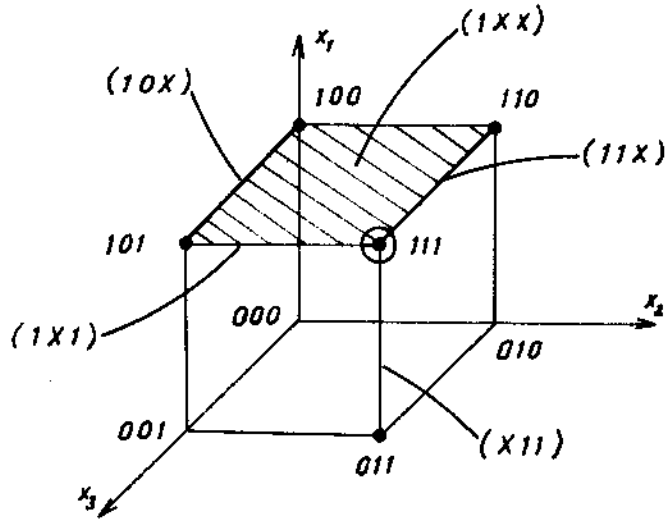
$$\begin{array}{r} 1\ 0\ 0\ 1 \\ *\ \underline{1\ 1\ 0\ 0} \\ 1\ Y\ 0\ Y = \Phi \end{array} \quad ; \quad \begin{array}{r} 1\ X\ 0\ 1\ 0 \\ *\ \underline{1\ 1\ 1\ 1\ 1} \\ 1\ 1\ Y\ 1\ Y \end{array} = \Phi$$

Kết quả (Φ) vì ta thấy Y xuất hiện hai lần (≥ 2)

Khi nhận được kết quả bằng (Φ) thì chứng tỏ là hai implicăng đó cách xa nhau và không thể nhóm họp (rút gọn) được với nhau.

$$\begin{array}{r} 1\ X\ 0\ 1 \\ *\ \underline{1\ 1\ 0\ X} \\ 1\ 0\ 0\ 1 \end{array} \quad ; \quad \begin{array}{r} 1\ 0\ 1\ 0 \\ *\ \underline{1\ 1\ 1\ 0} \\ 1\ Y\ 1\ 0 = 1X10 \end{array} \quad ; \quad \begin{array}{r} 1\ X\ 0\ 0 \\ *\ \underline{1\ 0\ 0\ 1} \\ 1\ 0\ 0\ X \end{array}$$

Có kết quả nhận được vì Y không xuất hiện, hay Y chỉ xuất hiện một lần. Cách tính toán ra kết quả rất đơn giản, nhưng ta thu được kết quả là cái gì khi sử dụng phép toán mới tích "sao tọa độ (*)", muốn biết ta có minh họa trên hình 1.81.



Hình 1.81. Minh họa hoạt động của phép toán sao tọa độ.

Dựa vào hình vẽ ta tính với phép toán sao tọa độ (*) :

$$\begin{array}{r} 1\ 0\ X \\ *\ \underline{1\ 1\ 1} \\ 1\ Y\ X = 1\ X\ X \end{array} \quad ; \quad \begin{array}{r} 1\ X\ X \\ *\ \underline{0\ 1\ 1} \\ X\ 1\ 1 \end{array} \quad ; \quad \begin{array}{r} X\ 1\ 1 \\ *\ \underline{1\ X\ X} \\ 1\ 1\ 1 \end{array} \quad ; \quad \begin{array}{r} 1\ 0\ X \\ *\ \underline{0\ 1\ 1} \\ Y\ Y\ 1 \end{array}$$

$$\begin{array}{r} 1\ 1\ X \\ *\ X\ 1\ 1 \\ \hline 1\ 1\ 1 \end{array} \quad ; \quad \begin{array}{r} 1\ X\ X \\ *\ 1\ 1\ X \\ \hline 1\ 1\ X \end{array} \quad ; \quad \begin{array}{r} X\ 1\ 1 \\ *\ 1\ 0\ X \\ \hline 1\ X\ 1 \end{array}$$

Từ những kết quả thu được ta có thể rút ra kết luận là : phép toán sao tọa độ (*) cho ta kết quả là phần chung lớn nhất có thể có giữa hai implicăng.

Điều quan trọng nhất mà ta thu được từ kết quả tính của phép sao tọa độ (*) là có thể tạo ra khối lớn hơn từ hai khối nhỏ hơn. Điều này cho phép ta tìm kiếm không gian Z từ tập số liệu ban đầu C_0.

Các tính chất hay tiên đề của phép toán (*) :

- Tính chất lũy đẳng :

$$A * A = A$$

- Tính chất giao hoán :

$$A * B = B * A$$

- Tính chất không kết hợp :

$$A * (B * C) \neq (A * B) * C$$

- Nếu $A \subseteq B$ thì $A * B = A$

Thuật toán tìm không gian Z các implicăng đơn giản. Dựa vào phép toán sao tọa độ (*) ta có thể tìm được không gian Z dựa vào ví dụ cụ thể cho trước được chỉ ra trên hình 1.82.

Ví dụ : Rút gọn hàm Boole 4 biến sau.

$x_1 x_2$ \ $x_3 x_4$	00	01	11	10
00	1	1		
01	1	1		
11		1	1	
10				1

Hình 1.82. Bài toán lôgic minh họa.

Từ bài toán lôgic đề ra đó ta có số liệu đầu vào hay phủ ban đầu C_0. Thông thường một bài toán lôgic bất kỳ ta chỉ có phủ C_0 ban đầu người ta cho trước, nhất là đối với những bài toán có số lượng biến số rất lớn không thể vẽ trên bìa K được. Nhiệm vụ đề ra là chỉ từ phủ C_0 ban đầu cùng với sự giúp đỡ của phép toán sao tọa độ (*) ta phải tìm ra được không gian Z các implicăng đơn giản không rút gọn được.

Ở đây ta có :

$$C_o = \left\{ c_1^0, c_2^0, \ldots, c_i^0, \ldots, c_n^0 \right\}$$

$$C_o = ((0000)\ (0001)\ (0100)\ (0101)\ (1101)\ (1111)\ (1010))$$

Trước hết tìm không gian các implicăng đơn giản (khối −0) (Z_0) ta dựa trên bổ đề là : tập không gian các implicăng đơn giản (khối −0) là tập hợp các (khối −0) (c_i^0) sao cho (c_i^0 * C_0) không sinh ra (chứa) một (khối 1) bất kỳ nào.

$$c_i^0 \subseteq C_0 \qquad 1 \leqslant i \leqslant n$$

tức là :

$$Z^0 = \left\{ c_i^0 \mid (c_i^0 * C_0) \not\subseteq \text{(khối } -1\text{) bất kỳ} \right\}$$

Ví dụ ta có :

$$c_1^0 = (0000)$$

$$(c_1^0 * C_0) = \begin{cases} (0000) * (0000) = (0000) \\ (0000) * (0001) = (000X) = (\text{Khối } -1) \\ (0000) * (0100) = (0X00) = (\text{Khối } -1) \\ \quad . \\ \quad . \\ \quad . \\ (0000) * (1010) = \Phi \end{cases}$$

Từ kết quả thu được ta nhận thấy rằng (c_1^0 * C_0) đã sinh ra (khối -1) bất kỳ, có nghĩa là (khối −0) (c_1^0) đó đã được chứa trong một (khối −1) hay nó đã được rút gọn và nhóm họp, chứ không phải là khối độc lập, cho nên có thể viết : $c_1^0 \not\subseteq Z^0$.

Tương tự tiếp theo có c_2^0 = (0001)

$$(c_2^0 * C_0) = \begin{cases} (0001) * (0000) = (000X) = (\text{Khối } -1) \\ (0001) * (0001) = (0001) \\ (0001) * (0100) = \Phi \\ (0001) * (0101) = (0X01) = (\text{Khối } -1) \\ \quad . \\ \quad . \\ \quad . \\ (0001) * (1010) = \Phi \end{cases}$$

Từ kết quả thu được ta thấy rằng (c_2^0 * C_0) đã sinh ra một (khối −1) bất kỳ, vậy kết luận là c_2^0 = (0001) không phải là implicăng đơn giản (khối −0) và có thể viết $c_2^0 \not\subseteq Z^0$.

Cứ áp dụng tính như vậy với c_3^0 = (0100) rồi c_4^0 = (0101) v.v... mãi cho tới c_7^0 = (1010) ta có:

$$(c_7^0 * C_0) = \begin{cases} (1010) * (0000) = \Phi \\ (1010) * (0001) = \Phi \\ (1010) * (0100) = \Phi \\ (1010) * (0101) = \Phi \\ (1010) * (1101) = \Phi \\ (1010) * (1111) = \Phi \\ (1010) * (1010) = (1010) \end{cases}$$

Nhận thấy $c_7^0 = (1010)$ khi tính $(c_7^0 * C_0)$ không sinh ra (tạo ra) được một (khối −1) bất kỳ nào, chứng tỏ tích số (implicăng) đó $c_7^0 = (1010) = x_1 \bar{x}_2 x_3 \bar{x}_4$ Không thể nhóm hợp được với bất kỳ một implicăng khác trong tập C_0 ban đầu, vì vậy nó là implicăng độc lập không rút gọn được, vậy ta có thể viết $c_7^0 \subseteq Z^0$ hay có kết quả :

$$Z^0 = ((1010))$$

Bước 2 : Lập phủ mới không chứa (khối −0), gọi là phủ C_1.

Bản thân phủ mới C_1 không chứa được hàm lôgic ban đầu, nhưng $(C_1 \cup Z^0)$ phải phủ được hàm lôgic đã cho. Trước hết xác định C_1 bằng cách

$$\hat{C}_1 = C_0 \cup (C_0 * C_0)$$

Kết quả tính toán $(C_0 * C_0)$ (chính là "tích" Đêcác của C_0 với chính nó), sẽ cho tất cả các (khối −1) có thể có, đồng thời hợp với C_0 trong đó cũng có thể có các (khối −1) khác, vì phủ ban đầu C_0 biểu diễn không phải chỉ toàn (khối −0) mà còn có thể có các khối khác, nó là một phủ bất kỳ của hàm lôgic bất kỳ. Sau đó rút gọn $\hat{C}_1$ bằng cách loại bỏ khỏi $\hat{C}_1$ những (khối −0), và những khối chứa trong khối khác, khi đó ta sẽ có phủ C_1 rút gọn là :

$$C_1 = \hat{C}_1 - \hat{C}_1^0 - \hat{C}_1^*$$

trong đó : $\hat{C}_1^0$ là các (khối −0)

$\hat{C}_1^*$ là các khối nằm trong khối khác.

Trước hết ta phải loại bỏ các (khối −0), vì ta đã có trong tay các implicăng đơn giản (khối −0) nằm trong không gian Z^0.

Đồng thời ta phải loại bỏ đi các khối nằm trong các khối khác, vì trong kết quả tính toán $(C_0 * C_0)$ là "tích" chập của C_0 theo phép toán sao tọa độ (*) với chính nó, có thể xuất hiện cả những khối lớn hơn so với các khối trước đó, nhưng cũng đồng thời cho ra kết quả có thể ra khối nhỏ hơn so với các khối trước đó, nhưng khi kết quả là nhỏ hơn thì kết quả đó luôn nằm trong các khối tạo ra nó (A ⩽ B thì A * B = A).

Ví dụ : (1 X X) * (X 11) = (111). Ở đây (111) là kết quả tính toán, nó là khối nhỏ hơn các khối ban đầu hay nằm trong các khối ban đầu, tức là (111) ⊆ (X11).

Như vậy những khối nằm trong các khối khác kiểu như (111) được sinh ra trong quá trình tính $(C_0 * C_0)$ sẽ đều nằm trong khối $\hat{C}_1^*$ là các khối nằm trong khối khác, chúng sẽ bị loại bỏ khỏi $\hat{C}_1$.

Ví dụ theo bài toán đã cho ta có.

$$C_0 = ((0000)\ (0001)\ (0100)\ (0101)\ (1101)\ (1111)\ (10101))$$

Ở đây kết quả tính

$$(C_0 * C_0) = \left((c_1^0 * C_0) + (c_2^0 * C_0) + \ldots + (c_7^0 * C_0)\right)$$

$$c_1^0 = (0000)$$

$$(c_1^0 * C_0) = \begin{cases} (0000) * (0000) = (0000) \\ (0000) * (0001) = (000X) \\ (0000) * (0100) = (0X00) = (\text{Khối } -1) \\ (0000) * (0101) = \Phi \\ (0000) * (1101) = \Phi \\ (0000) * (1111) = \Phi \\ (0000) * (1010) = \Phi \end{cases}$$

Tiếp theo tích :

$$c_2^0 = (0001)$$

$$(c_2^0 * C_0) = \begin{cases} (0001) * (0000) = (000X) - (\text{Khối } -1) \\ (0001) * (0001) = (0001) \\ (0001) * (0100) = \Phi \\ (0001) * (0101) = (0X01) - (\text{Khối } -1) \\ (0001) * (1101) = \Phi \\ (0001) * (1111) = \Phi \\ (0001) * (1010) = \Phi \end{cases}$$

. . . . . .

Cách làm tương tự tới :

$$c_7^0 = (1010)$$

$$(c_7^0 * C_0) = \begin{cases} (1010) * (0000) = \Phi \\ (1010) * (0001) = \Phi \\ (1010) * (0100) = \Phi \\ (1010) * (0101) = \Phi \\ (1010) * (1101) = \Phi \\ (1010) * (1111) = \Phi \\ (1010) * (1010) = (1010) \end{cases}$$

Kết quả cuối cùng thu được sau quá trình tính

$$(C_0 * C_0) = (\ (0000)\ (000X)\ (0X00)\ (000X)\ (0001)\ (0X01)\ (010X)\ (X101)\ (11X1)\ (1111)\ (1010)\ )$$

Sau đó tính tiếp ta có

$$\hat{C}_1 = C_0 \cup (C_0 * C_0) = (\ (0000)\ (0001)\ (0100)\ (0101)\ (1101)\ (1111)\ (1010)\ ) \cup (\ (0000)\ (000X)\ (0X00)\ (000X)\ (0001)\ (0X01)\ (010X)\ (X101)\ (11X1)\ (1111)\ (1010)\ )$$

$$\hat{C}_1 = (\ (0000)\ (0001)\ (0100)\ (0100)\ (0101)\ (1101)\ (1111)\ (1010)\ (0000)\ (000X)\ (0X00)\ (000X)\ (0001)\ (0X01)\ (010X)\ (X101)\ (11X1)\ (1111)\ (1010)\ )$$

Tiếp theo ta thực hiện rút gọn $\hat{C}_1$ bằng cách loại bỏ trong nó các (khối −0) ($\hat{C}^0$), loại bỏ đi trong nó các khối nằm trong khối khác $\hat{C}_1^*$. Kết quả thu được lại là :

$$C_1 = \hat{C}_1 - \hat{C}_1^0 - \hat{C}_1^* = ((000X)\ (0X00)\ (0X01)\ (010X)\ (X101)\ (11X1))$$

Bước 3. Từ phủ mới C_1 tìm được không gian các implicăng đơn giản (khối −1) là Z^1 (bước này làm giống như tìm Z^0)

$$Z^1 = \left\{ c_i^1 \mid (c_i^1 * C_1) \not\subseteq \text{(Khối −2) bất kỳ} \right\}$$

Tức là c_i^1 sẽ là implicăng đơn giản (khối −1) (Z^1) nếu như ($c_i^1 * C_1$) không sinh ra một (khối −2) (không nằm trong) bất kỳ nào.

Theo dõi tiếp từ ví dụ đã cho ta có :

$$C_1 = ((000X)\ (0X00)\ (0X00)\ (0X01)\ (010X)\ (X101)\ (11X1))$$

Ta lấy $c_1^1 = (000X)$

$$(c_1^1 * C_1) = \begin{cases} (000X) * (000X) = (000X) \\ (000X) * (0X00) = (0000) \\ (000X) * (0X01) = (0001) \\ (000X) * (010X) = (0X0X) - \text{(Khối −2)} \\ (000X) * (X101) = (0X01) \\ (000X) * (11X1) = \Phi \end{cases}$$

Từ kết quả nhận thấy ($c_1^1 * C_1$) sinh ra một (khối −2) (0X0X) vậy $c_2^1 = (000X)$ không nằm trong Z^1. ($c_2^1 \not\subseteq Z^1$).

lấy $c_2^1 = (0X00)$

$$(c_2^1 * C_1) = \begin{cases} (0X00) * (000X) = (0000) \\ (0X00) * (0X00) = (0X00) \\ (0X00) * (0X01) = (0X0X) \\ (0X00) * (010X) = (0X0X) - \text{(Khối −2)} \\ (0X00) * (010X) = (0100) \\ (0X00) * (X101) = (010X) \\ (0X00) * (11X1) = \Phi \end{cases}$$

Trong kết quả thu được ta thấy xuất hiện một (khối −2) (0X0X) nếu $C_2^1 = (0X00)$ không phải là implicăng đơn giản (khối −1), vì nó bị chứa trong một (khối −2) (0X0X) $\subseteq$ (0X0X).

Cách làm tương tự với $c_3^1 = (0X01)$ và $c_4^1 = (010X)$ thấy chúng đều không nằm trong Z^1 và không phải implicăng đơn giản (khối −1). Ta tính tiếp :

lấy $c_5^1 = (X101)$

$$(c_5^1 * C_1) = \begin{cases} (X101) * (000X) = (0X01) \\ (X101) * (0X00) = (010X) \\ (X101) * (010X) = (0101) \\ (X101) * (010X) = (0101) \\ (X101) * (X101) = (X101) \\ (X101) * (11X1) = (1101) \end{cases}$$

Kết quả tích đã không thấy xuất hiện bất kỳ một (khối −2) nào vậy kết luận c_5^1 = (X101) là implicăng đơn giản (khối −1) hay $c_5^1 \in Z^1$.

Cách làm tương tự với c_6^1 = (11X1), khi ta tính tích sao tọa độ của c_6^1 với tập C_1 (c_6^1 * C_1) cũng không thấy có một (khối −2) nào được sinh ra, vậy c_6^1 là implicăng đơn giản (khối −1) hay $c_6^1 \in Z^1$.

Từ đó ta có tập các implicăng đơn giản (khối −1) Z^1 là :

$$Z^1 = ((X101)\ (11X1))$$

Bước 4 : Lập phủ mới không chứa (khối −1), gọi là phủ C_2 (cách làm tương tự như bước 2)

Chú ý bản thân phủ mới C_2 không chứa được hàm lôgic đã cho, nhưng ($C_2 \cup Z^0 \cup Z^1$) phải phủ được hàm lôgic ban đầu đưa ra. Để xác định được C_2 ta chỉ dựa vào tập số liệu đầu vào là C_1, trước hết ta tìm phủ $\hat{C}_2$ dựa theo công thức :

$$\hat{C}_2 = C_1 \cup (C_1 * C_1)$$

Kết quả tính toán (C_1 * C_1) chính là tích Đề các của C_1 với chính nó (còn được gọi là tích chập của C_1) sẽ cho tất cả các (khối −2) có thể có, đây chính là các khối lớn hơn được tạo ra từ C_1 nhờ có phép toán sao tọa độ (*), đó là kết quả quan trọng nhất khi đưa ra phép toán này. Tất nhiên phép toán sao tọa độ (*) cũng cho ra các kết quả là các khối nhỏ hơn các khối ban đầu, nhưng các khối nhỏ hơn này được chắc chắn sẽ nằm trong các khối đã cho trước đó nên có thể dễ dàng loại bỏ khỏi $\hat{C}_2$. Khi đó ta có phủ C_2 được rút gọn là :

$$C_2 = \hat{C}_2 - \hat{C}_2^1 - \hat{C}_2^*$$

trong đó : $\hat{C}_2^1$ là các (khối −1)

$\hat{C}_2^*$ là các khối nằm trong khối khác.

Ta phải bỏ các (khối −1), vì những implicăng đơn giản (khối −1) đã được giữ lại ở Z^1.

Tiếp tục từ kết quả bài toán đã cho ta có :

$$C_1 = ((000X)\ (0X00)\ (0X01)\ (010X)\ (X101)\ (11X1))$$

Trước hết tính tích chập (C_1 * C_1) ta có :

$$(C_1 * C_1) = (c_1^1 * C_1) + (c_2^1 * C_1) + \ldots + (c_6^1 * C_1)$$

$$c_1^1 = (000X)$$

$$(c_1^1 * C_1) = \begin{cases} (000X) * (000X) = (000X) \\ (000X) * (0X00) = (0000) \\ (000X) * (0X01) = (0001) \\ (000X) * (010X) = (0X0X) - (\text{Khối } -2) \\ (000X) * (X101) = (0X01) \\ (000X) * (11X1) = \Phi \end{cases}$$

Tính tiếp theo

$$c_2^1 = (0X00)$$

$$(c_2^1 * C_1) = \begin{cases} (0X00) * (000X) = (0000) \\ (0X00) * (0X00) = (0X00) \\ (0X00) * (0X01) = (0X0X) \\ (0X00) * (010X) = (0100) \\ (0X00) * (X101) = (010X) \\ (0X00) * (11X1) = \Phi \\ \cdots\cdots\cdots\cdots\cdots \end{cases}$$

Tính tiếp theo cho tới hết :

$$c_6^1 = (11X1)$$

$$(c_6^1 * C_1) = \begin{cases} (11X1) * (000X) = \Phi \\ (11X1) * (0X00) = \Phi \\ (11X1) * (0X01) = (X101) \\ (11X1) * (010X) = (X101) \\ (11X1) * (X101) = (1101) \\ (11X1) * (11X1) = (11X1) \end{cases}$$

Kết quả cuối cùng thu được sau quá trình tính toán :

$$(C_1 * C_1) = (\,(000X)\ (0000)\ (0001)\ (0X0X)\ (0X01)$$
$$(0000)\ (0X00)\ (0X0X)\ (0100)\ (010X)\ (0001)\ (0X01)$$
$$(0101)\ (X101)\ (X101)\ (1101)\ (11X1)\,)$$

Tính được kết quả $\hat{C}_2$ từ bài toán đề ra là :

$$\hat{C}_2 = C_1 \cup (C_1 * C_1) = (\,(000X)\ (0X00)\ (0X01)\ (010X)\ (X101)$$
$$(11X1)\ (000X)\ (0000)\ (0001)\ (0X0X)\ (0000)\ (0X00)\ (0X0X)\ (0100)$$
$$(010X)\ (0001)\ (0X01)\ (0101)\ (X101)\ (X101)\ (11)1)\ (11X1)\,)$$

Tiếp theo rút gọn $\hat{C}_2$ thành C_2 dựa theo công thức ta có kết quả là :

$$C_2 = \hat{C}_2 - \hat{C}_2^1 - \hat{C}_2^* = (\,(0X0X)\,)$$

Bước 5. Từ phủ mới C_2 tìm được không gian các implicăng đơn giản (khối −2) là Z^2 (bước này giống các bước tìm Z^0, Z^1 ở trên)

$$Z^2 = \left\{ c_i^2 \mid (c_i^2 * C_2) \not\subseteq \text{(Khối } -3\text{) bất kỳ} \right\}$$

Từ kết quả của ví dụ đã cho ta có

$$C_2 = (\,(0X0X)\,)$$

Vậy có :

$$c_1^2 = (0X0X)$$

$$c_1^2 * C_2 = (0X0X) * (0X0X) = ((0X0X))$$

Kết quả không tạo ra được (khối −3) nào, vậy kết luận $c_1^2 = (0X0X)$ là implicăng đơn giản (khối −2), thu được không gian các implicăng đơn giản Z^2 là

$$Z^2 = (\,(0X0X)\,)$$

Bước 6. Lại lập phủ mới không chứa (khối −2), gọi là phủ C_3

Từ C_2 xác định $\hat{C}_3$ theo công thức :

$$\hat{C}_3 = C_2 \cup (C_2 * C_2)$$

Sau đó lại rút gọn $\hat{C}_3$ theo quan hệ

$$C_3 = \hat{C}_3 - \hat{C}_3^2 - \hat{C}_3^*$$

trong đó : $\hat{C}_3^2$ là các (khối −2)

$\hat{C}_3^*$ là các khối nằm trong khối khác

Tính mãi cho tới khi $C_{n+1} = \Phi$ thì dừng quá trình tìm không gian Z các implicăng đơn giản, không gian Z có kết quả là :

$$Z = \bigcup_{i=0}^{n} Z^i$$

Với ví dụ từ bài toán đã cho ta có C_2 = (0X0X) vậy :

$$\hat{C}_3 = C_2 \cup (C_2 * C_2) = ((0X0X)\ (0X0X))$$

Rút gọn $\hat{C}_3$ ta được :

$$C_3 = \hat{C}_3 - \hat{C}_3^2 - \hat{C}_3^* = \Phi$$

Đến đây dừng kết quả tính toán , và kết quả thu được không gian các implicăng đơn giản Z của bài toán đã cho là :

$$Z = Z^0 + Z^1 + Z^2$$

$$Z = ((1010)\ (X101)\ (11X1)\ (0X0X))$$

$$Z = x_1 \bar{x}_2 x_3 \bar{x}_4 + x_2 \bar{x}_3 x_4 + x_1 x_2 x_4 + \bar{x}_1 \bar{x}_3$$

Như vậy đã thực hiện xong và kết thúc quá trình tìm không gian Z các implicăng đơn giản không rút gọn được dựa trên thuật toán của Roth. Với việc định nghĩa ra phép toán sao tọa độ (*) Roth đã đưa ra một thuật toán để tìm được không gian Z, từ đó dựa theo thuật toán có thể lập trình cho máy tính để tự động hóa quá trình tìm không gian Z các implicăng đơn giản của bài toán lôgic (hay chức năng kỹ thuật) bất kỳ khi số biến của bài toán (hay số lượng tín hiệu vào) quá lớn. Đến đây vấn đề còn lại chỉ là từ không gian Z thu được, phải tìm ra được không gian các khối đầu mút E các implicăng đơn giản không thể thiếu được của hàm lôgic, tiến tới tìm phủ tối thiểu C_{min} của hàm lôgic (hay của chức năng) đề ra.

Tìm các khối đầu mút không gian các "khối đầu mút E", chúng ta đều biết các khối đầu mút (các extreman) của một hàm lôgic hay một chức năng là những implicăng đơn giản thật sự, những thành phần tối thiểu cần thiết không thể bỏ trong quá trình tìm phủ C_{min}, hay nói cách khác đó là những mạch điện đơn giản không thể thiếu khi lớp ráp mạch điện thực hiện chức năng của hàm (theo hàm) lôgic đã cho. Những implicăng đơn giản khác còn có thể bị loại bỏ nếu nó không chứa (không có) đỉnh đầu mút, còn

nếu implicăng đơn giản và chứa đỉnh đầu mút (được gọi là khối đầu mút) thì bắt buộc phải giữ lại trong phủ C_{min}, vì nếu bỏ đi thì sẽ bị mất chức năng đầu mút, đó là khái niệm rất cơ bản đã được nêu lên ở phần trên.

Như vậy muốn tìm được các khối đầu mút (các extreman) cần phải dựa vào các ""đỉnh đầu mút"" hay còn gọi là "đỉnh tận cùng", "đỉnh đặc biệt". Tập các đỉnh đỉnh đầu mút ta gọi là "không gian L các đỉnh đầu mút". Vấn đề cụ thể ở đây là phải tìm được không gian các đỉnh đầu mút L từ tập Z các implicăng đơn giản đã tìm được, tập Z chính là số liệu đầu vào để từ đó tìm ra tập L.

$$Z = \{ z_1, z_2, ..., z_m\}$$

Cách tìm không gian L các đỉnh đầu mút :

Chúng ta đều biết mỗi một implicăng đơn giản $z_i \subseteq Z$ đều phủ ở bên trong đó các đỉnh ứng với quá trình nhóm hợp các tích số để tạo ra nó, nói cách khác mỗi một ienphicăng đơn giản đều chứa trong nó các đỉnh tạo ra nó.

Ví dụ : Giả sử ta có (010X) là implicăng đơn giản thì (010X) bao gồm (0101) và (0100) hay có thể viết $(\bar{x}_1 x_2 \bar{x}_3) = (\bar{x}_1 x_2 \bar{x}_3 x_4 + \bar{x}_1 x_2 \bar{x}_3 \bar{x}_4)$

Giả sử (0X0X) là implicăng đơn giản thì nó sẽ bao gồm các đỉnh ((0000) (0001) (0100) (0101)). Quá trình này ta gọi là khai triển hay tìm các đỉnh của implicăng đơn giản (implicăng đã được rút gọn).

Từ đó có thể khai triển hay tìm các đỉnh của không gian Z các implicăng đơn giản, từ đó ta có thể tìm được các đỉnh đỉnh đầu mút L cụ thể là :

Tiếp tục từ bài toán đã đề ra, và qua quá trình tìm không gian Z đã thu được kết quả là :

$$Z = ((1010)\ (X1010\ (11X1)\ (0X0X))$$

Khai triển Z sẽ tìm được tập các đỉnh của hàm lôgic $\hat{L}$ là :

$$\hat{L} = ((1010)\ (0101)\ (1101)\ (1101)\ (1111)\ (0000)\ (0001)\ (0100)\ (0101))$$

Sau đó rút gọn $\hat{L}$ theo công thức :

$$L = \hat{L} - \hat{L}^*$$

Trong đó : $\hat{L}^*$ là tập các đỉnh xuất hiện $\geq$ 2 lần trong $\hat{L}$.

Ví dụ : (1101) đã xuất hiện $\geq$ 2 lần trong $\hat{L}$.

Ta loại bỏ các đỉnh có số lần xuất hiện $\geq$ 2 lần, vì đỉnh đầu mút là đỉnh chỉ nằm trong một implicăng đơn giản mà không nằm trong các implicăng khác, như vậy nếu một đỉnh sau khi khai triển lại xuất hiện $\geq$ 2 lần, tức là đỉnh đó nằm trong (thuộc) ít nhất hai implicăng đơn giản của không gian Z, từ đó có thể kết luận là các đỉnh đó không phải là đỉnh đỉnh đầu mút hay đỉnh tận cùng, vì thế có thể loại bỏ chúng. Như vậy sau khi loại bỏ các đỉnh giống nhau (hay xuất hiện $\geq$ 2 lần) trong $\hat{L}$, thì kết quả còn lại chắc chắn đó là tập các đỉnh đầu mút mà ta cần tìm.

Từ ví dụ cụ thể ở trên sau khi loại bỏ đi các đỉnh giống nhau trong $\hat{L}$, ta thu được không gian L các đỉnh đỉnh đầu mút của bài toán lôgic đề ra là :

$$L = ((1010)\ (1111)\ (0000)\ (0001)\ (0100))$$

Để tìm được các khối đầu mút, đó là các implicăng đơn giản thực sự không thể thiếu được trong phủ C_{min} của hàm lôgic, ta cần phải có hai tập số liệu quan trọng đó là : Tập không gian Z các implicăng đơn giản không rút gọn được và tập không gian L các đỉnh đầu mút hay đỉnh đặc biệt, đỉnh tận cùng. Bởi vì một khối đầu mút hay một implicăng đơn giản không thể thiếu được (mạch điện không thể thiếu được) của hàm lôgic nó cần phải thỏa mãn điều kiện trước hết phải là một implicăng đơn giản sau đó đồng thời phải chứa (có) đỉnh đầu mút.

Cho đến đây ta đã có trong tay kết quả của tập không gian Z các implicăng đơn giản và tập các đỉnh đầu mút L từ bài toán đề ra là :

$$Z = ((1010)\ (X001)\ (11X1)\ (0X0X))$$

$$L = ((1010)\ (1111)\ (0000)\ (0001)\ (0100))$$

Từ hai tập số liệu ban đầu Z và L đó ta sẽ tìm được không gian E các khối đầu mút của hàm lôgic đã cho. Muốn tìm được E từ hai tập Z và L ta cần phải chế ra hay tự định nghĩa ra một phép toán mới có tên là "phép toán giao tọa độ" ($\cap$), ở đây chữ "tọa độ" có nghĩa là : nếu viết phép toán theo hàng dọc thì việc tính toán kết quả ở từng cột hoàn toàn độc lập với nhau, nghĩa là kết quả tính toán ở cột số liệu này không phụ thuộc vào kết quả tính toán ở cột khác, nhưng kết quả chung của cả phép toán vẫn đúng. Phép toán giao tọa độ được định nghĩa như sau :

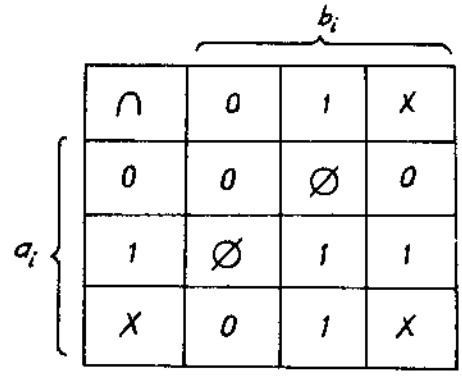
Định nghĩa 1.9.4: Phép toán giao tọa độ ($\cap$)

Cho A và B là hai implicăng (tích số) bất kỳ có ký hiệu là :

$$A = a_1, a_2, \ldots, a_n)$$

$$B = b_1, b_2, \ldots, b_n)$$

Với một bảng quan hệ ký hiệu được biểu diễn trên hình 1.83.



Hình 1.83. Phép toán giao tọa độ.

Với điều kiện thực hiện quan hệ ký hiệu được quy định như sau :

$$A \cap B = \begin{cases} \Phi \text{ nếu } (a_i \cap b_i) = \Phi \text{ với i bất kỳ.} \\ ((a_1 \cap b_1)\ (a_2 \cap b_2) \ldots (a_n \cap b_n)) \\ \text{trong các trường hợp khác.} \end{cases}$$

Cho vài ví dụ tính toán với phép giao tọa độ (∩)

$$\begin{array}{r} 1\ 1\ 0\ 1 \\ \cap\ 1\ 0\ 0\ 0 \\ \hline 1\ \Phi\ 0\ \Phi \end{array} = \Phi \quad ; \quad \begin{array}{r} 1\ X\ 0\ X \\ \cap\ 1\ X\ 1\ 0 \\ \hline 1\ X\ \Phi\ 0 \end{array} = \Phi \quad ; \quad \begin{array}{r} X\ 1\ X\ 0 \\ \cap\ 1\ 1\ X\ X \\ \hline 1\ 1\ X\ 0 \end{array}$$

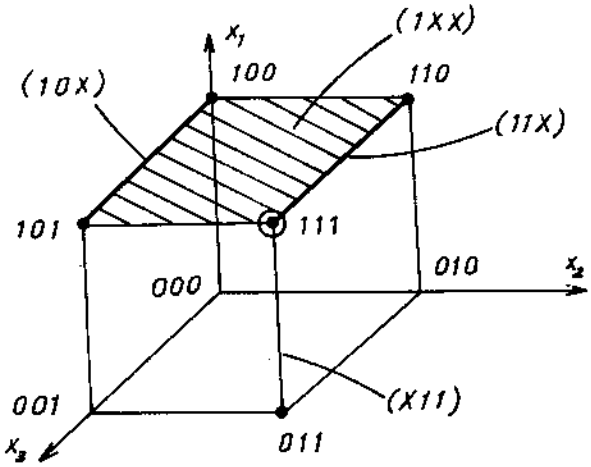
$$\begin{array}{r} X\ 1\ X\ 0 \\ \cap\ 0\ X\ 0\ 0 \\ \hline 0\ 1\ 0\ 0 \end{array} \quad ; \quad \begin{array}{r} X\ 0\ X\ 1 \\ \cap\ 0\ 0\ 1\ 1 \\ \hline 0\ 0\ 1\ 1 \end{array} \quad ; \quad \begin{array}{r} 0\ X\ 1\ X \\ \cap\ 0\ 1\ 0\ X \\ \hline 0\ 1\ \Phi\ X \end{array} = \Phi$$

Có thể viết phép toán theo hàng ngang để tính nhưng nhận biết được kết quả sẽ khó khăn hơn.

$$(1\,X\,0\,0) \cap (1\,X\,1\,0) = \Phi$$

$$(X\,1\,X\,0) \cap (1\,1\,X\,X) = (1\,1\,X\,0)$$

Cách tính toán với phép toán giao tọa độ (∩) giữa hai implicăng bất kỳ cũng rất đơn giản và cho ra đáp số dễ dàng. Nhưng kết quả thu được sau khi tính là gì hay ý nghĩa của phép giao tọa độ (∩) như thế nào, muốn biết có thể dựa vào minh họa trên hình 1.84.



Hình 1.84. Minh họa hoạt động của phép toán giao tọa độ.

Dựa vào hình vẽ và kết quả tính ta biết được ý nghĩa của phép toán mới giao tọa độ (∩) :

$$\begin{array}{r} 1\ 0\ X \\ \cap\ 1\ 1\ X \\ \hline 1\ \Phi\ X \end{array} = \Phi \quad ; \quad \begin{array}{r} 1\ X\ X \\ \cap\ 0\ 1\ 1 \\ \hline \Phi\ 1\ 1 \end{array} = \Phi \quad ; \quad \begin{array}{r} X\ 1\ 1 \\ \cap\ 1\ 0\ X \\ \hline 1\ \Phi\ 1 \end{array} = \Phi$$

$$\begin{array}{r} 1\ X\ X \\ \cap\ X\ 1\ 1 \\ \hline 1\ 1\ 1 \end{array} \quad ; \quad \begin{array}{r} 1\ 1\ X \\ \cap\ X\ 1\ 1 \\ \hline 1\ 1\ 1 \end{array} \quad ; \quad \begin{array}{r} 1\ X\ X \\ \cap\ 1\ 1\ 1 \\ \hline 1\ 1\ 1 \end{array} \quad ; \quad \begin{array}{r} 0\ 1\ 1 \\ \cap\ 1\ 0\ X \\ \hline \Phi\ \Phi\ 1 \end{array} = \Phi$$

Từ kết quả tính toán và hình 1.84 có thể rút ra kết luận là : Phép toán giao tọa độ ($\cap$) cho ta kết quả là phần chung bé nhất có thể có giữa hai implicăng.

Nói cách khác kết quả của phép giao tọa độ ($\cap$) cho ta phần liên quan hay "dính dáng" ít nhất có thể có giữa hai implicăng bất kỳ, và khi kết quả nhận được là (Φ) thì chứng tỏ hai implicăng (hai tích số đó) hoàn toàn không có tí gì liên quan hay "dính dáng" đến nhau, chúng nó hoàn toàn độc lập với nhau. Đó là những nhận xét và hiểu biết rất cần và quan trọng nhận được từ kết quả của phép toán giao tọa độ ($\cap$), vì từ tính chất đó (của phép giao tọa độ) cho phép ta tìm được khối đầu mút E từ tập không gian các implicăng đơn giản Z và không gian các đỉnh đầu mút L.

Các tính chất hay tiên đề của phép toán giao tọa độ ($\cap$) :

- Tính chất lũy đẳng :

$$A \cap A = A$$

- Tính chất giao hoán :

$$A \cap B = B \cap A$$

- Tính chất kết hợp :

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

- Nếu $A \subseteq B$ thì $A \cap B = A$

Thuật toán tìm không gian E các khối đầu mút từ hai tập Z các implicăng đơn giản và L các đỉnh đầu mút dựa trên phép toán giao tọa độ được thực hiện như sau :

Từ hai tập Z và L là số liệu đầu vào

$$Z = \{z_1, z_2, ..., z_i, ... z_m\}$$

$$L = \{l_1, l_2, ..., l_j, ..., l_K\}$$

Định lý 1.9.1. Một implicăng đơn giản z_i nằm trong Z ($z_i \subseteq Z$) sẽ thuộc không gian các khối đầu mút E nếu thỏa mãn quan hệ (tích Đêcác) :

$z_i \cap L$ tồn tại $z_i \cap l_j \neq \Phi$ thì $z_i \in E$ với $l_j \in L$.

Để chứng minh định lý này rất đơn giản, vì z_i là implicăng đơn giản, mà L là tập các đỉnh đầu mút, đồng thời tính chất của phép giao tọa độ là tìm ra phần chung bé nhất giữa hai implicăng, cho nên $z_i \cap L \neq \Phi$ tức là (có nghĩa là) implicăng đơn giản z_i có chứa (hay "dính dáng") đến một đỉnh đầu mút nào đó l_i thuộc L. Như vậy z_i vừa là implicăng đơn giản vừa có chứa đỉnh đầu mút thì chắc chắn nó sẽ là khối đầu mút ($z_i \in E$).

Để hiểu kỹ hơn định lý cũng như hiểu được thuật toán tìm không gian E các khối đầu mút các implicăng đơn giản thật sự không thể thiếu được trong quá trình tìm phủ C_{min} của hàm lôgic, ta có thể dựa vào ví dụ đã nêu trên, khi có tập Z và L như sau :

Đặt $Z_1 = Z$ và $L_1 = L$ ứng với quá trình tìm E_1 đầu tiên.

$$Z_1 = Z = ((1\,0\,1\,0)\ (X\,1\,0\,1)\ (1\,1\,X\,1)\ (0\,X\,0\,X))$$

$$L_1 = L = ((1\,0\,1\,0)\ (1\,1\,1\,1)\ (0\,0\,0\,0)\ (0\,0\,0\,1)\ (0\,1\,0\,0))$$

Dựa vào định lý ta có :

$$z_1 = (1\,0\,1\,0)$$

$$(z_1 \cap L_1) = \begin{cases} (1010) \cap (1010) = (1010) \neq \Phi \\ (1\,010) \cap (1111) = \Phi \\ (1010) \cap (0000) = \Phi \\ (1010) \cap (0001) = \Phi \\ (1010) \cap (0100) = \Phi \end{cases}$$

Từ kết quả thu được ta kết luận $z_1 = (1010)$ là khối đầu mút, vì có kết quả tích $(z_1 \cap L_1) \neq \Phi$ $(z_1 \in E_1)$

$$z_2 = (X101)$$

$$(z_1 \cap L_1) = \begin{cases} (X101) \cap (1010) = \Phi \\ (X101) \cap (1111) = \Phi \\ (X101) \cap (0000) = \Phi \\ (X101) \cap (0001) = \Phi \\ (X101) \cap (0100) = \Phi \end{cases}$$

Từ kết quả tính toán ta thấy $(z_2 \cap L_1) = \Phi$, từ đó có thể nói $z_2 = (X101)$ không phải là khối đầu mút $(z_2 \notin E_1)$.

$$z_3 = (11X1)$$

$$(z_1 \cap L_1) = \begin{cases} (11X1) \cap (1010) = \Phi \\ (11X1) \cap (1111) = (1111) \neq \Phi \\ (11X1) \cap (0000) = \Phi \\ (11X1) \cap (0001) = \Phi \\ (11X1) \cap (0100) = \Phi \end{cases}$$

Như vậy $(z_3 \cap L_1) \neq \Phi$ cho nên $z_3 = (11X1)$ là khối đầu mút $(z_3 \in E_1)$

$$z_4 = (0X0X)$$

$$(z_1 \cap L_1) = \begin{cases} (0X0X) \cap (1010) = \Phi \\ (0X0X) \cap (1111) = \Phi \\ (0X0X) \cap (0000) = (0000) \neq \Phi \\ (0X0X) \cap (0001) = (0001) \neq \Phi \\ (0X0X) \cap (0100) = (0100) \neq \Phi \end{cases}$$

Ta có kết quả $(z_4 \cap L_1) \neq \Phi$ như vậy $z_4 = (0X0X)$ là khối đầu mút $(z_4 \in E_1)$

Đến đây ta kết thúc quá trình tìm các khối đầu mút lần thứ nhất E_1 (trong đó z_i là các khối đầu mút).

$$E_1 = z_1 + z_3 + z_4 = \bigcup_{i=1}^{n} z_i \qquad \text{với } z_i \in Z_1$$

$$E_1 = ((1010)\ (11X1)\ (0X0X))$$

Để tìm được E_2 tiếp theo ta phải tìm phủ mới các implicăng đơn giản Z_2 dựa vào quan hệ :

$$Z_2 = Z_1 \ \# \ E_1$$

Từ ví dụ đã nêu trên và kết quả thu được sau khi tính toán tìm E_1 ta có :

$$Z_2 = ((1010)\ (X1010)\ (11X1)\ (0X0X)) \ \#$$

$$((1010)\ (11X1)\ (0X0X)) = \Phi$$

Khi có kết quả $Z_2 = \Phi$ ta dừng quá trình tìm không gian các khối đầu mút E và thu được phủ tối thiểu C_{min} là :

$$C_{min} = E = \bigcup_{i=1}^{n} E_i$$

Với bài toán đã đề ra ta có :

$$C_{min} = E_1 = ((1010)\ (11X1)\ (0X0X))$$

$$f_{min}\ (x_1, x_2, x_3) = x_1 \bar{x}_2 x_3 \bar{x}_4 + x_1 x_2 x_4 + \bar{x}_1 \bar{x}_3$$

Dây là kết quả rút gọn của hàm Boole quen thuộc, vì bằng các phương pháp rút gọn khác ca Kácnô hay của Mc.Cluskey đã thu được ở phần trên. Nhưng trong thực tế bài toán tổng hợp hay thực hiện một chức năng bằng mạch điện không chỉ đơn giản và dễ dàng như vậy, quá trình tìm phủ tối thiểu C_{min} (hay mạch điện đơn giản nhất) của một hàm lôgíc trong thực tế còn phức tạp hơn nhiều, để hiểu trọn vẹn hơn quá trình tìm C_{min} theo phương pháp Roth ta cần phải tạo ra hay định nghĩa xây dựng phép toán mới có tên là phép toán "hiệu tọa độ" (#). Dây là phép toán ta đã công nhận và sử dụng ở các phần trên khi dùng nó để loại bỏ "mạch thừa" hay "khoanh thừa", ở đây phép toán hiệu tọa độ (#) sẽ được biết đến một cách cụ thể và chính xác như sau.

Định nghĩa 1.9.5: Phép toán hiệu tọa độ **(#)**

Cho A và B là hai implicăng bất kỳ có ký hiệu là :

$$A = (a_1, a_2, ..., a_n)$$

$$B = (b_1, b_2, ..., b_n$$

Với bảng quan hệ ký hiệu được biểu diễn trên hình 1.85.

Với điều kiện thực hiện quan hệ ký hiệu được quy định như sau : (chú ý phép # không có tính chất giao hoán nên vị trí a_i và b_i ở bảng quan hệ ký hiệu phải chính xác).

a_i \ b_i	#	0	1	X
0	Z	Y	Z	
1	Y	Z	Z	
X	1	0	Z	

Hình 1.85. Phép toán hiệu tọa độ.

$$A \# B = \begin{cases} A \text{ nếu } (a_i \# b_i) = Y \text{ với i bất kỳ} \\ \Phi \text{ nếu } (a_i \# b_i) = Z \text{ với mọi i} \\ \bigcup_i^n (a_1,..., a_{i-1}, \alpha_i, a_{i+1},..., a_n) \\ \text{nếu } (a_i \# b_i) = \alpha_i = 0 \text{ hoặc } 1 \end{cases}$$

Có vài ví dụ tính toán sử dụng phép hiệu tọa độ (#) :

Trường hợp :

$$\begin{array}{r} 1\ X\ 0\ 1 \\ \#\ X\ 1\ 1\ X \\ \hline Z\ 0\ Y\ Z \end{array} = 1X01 \qquad \begin{array}{r} X\ 1\ 0\ 0 \\ \#\ X\ 0\ X\ 1 \\ \hline Z\ Y\ Z\ Y \end{array} = X100$$

Kết quả nhận được bằng số bị trừ (hiệu) A vì có Y xuất hiện ở một cột bất kỳ. Điều này chứng tỏ số bị trừ là A (ví dụ A = (1X01)) và số trừ là B (B = X11X) chúng không "dính dáng" liên hệ gì với nhau, cho nên khi trừ (hiệu tọa độ #) đi cho nhau thì số bị trừ là A vẫn còn nguyên là A, mà không bị mất tí gì cả.

Trường hợp 1:

$$\begin{array}{r} 1\ 0\ X\ 0 \\ \#\ X\ 0\ X\ X \\ \hline Z\ Z\ Z\ Z \end{array} = \Phi \qquad \begin{array}{r} 0\ 1\ 1\ X \\ \#\ X\ X\ 1\ X \\ \hline Z\ Z\ Z\ Z \end{array} = \Phi \qquad \begin{array}{r} 1\ X\ 0\ 0 \\ \#\ X\ X\ 0\ X \\ \hline Z\ Z\ Z\ Z \end{array} = \Phi$$

Kết quả của phép hiệu tọa độ (#) giữa số bị trừ A và số trừ B, với B = (X0XX) cho ta kết quả là bằng (Φ), chứng tỏ khi trừ từ đầu đến cuối đều không còn gì cả toàn là (Z) vậy kết quả là rỗng (Φ), cũng là một trong các trường hợp xảy ra khi từ (hiệu tọa độ #)

Trường hợp 2 :

$$\begin{array}{r} 1\ X\ 0\ X \\ \#\ X\ 1\ X\ X \\ \hline Z\ 0\ Z\ Z \end{array} = 100X \qquad \begin{array}{r} 1\ 0\ X\ 1 \\ \#\ 1\ 0\ 0\ X \\ \hline Z\ Z\ 1\ Z \end{array} = 1011$$

Kết quả tính toán theo như định nghĩa của phép (#) và theo điều kiện thực hiện quan hệ ký hiệu, cách thực hiện ở đây là khi có $\alpha_i = 0$ hoặc 1 thì thực hiện gép các số của A (từ a_1 tới a_{i-1}) sau đó ghi giá trị của α_i, tiếp theo lại gép tiếp các số (giá trị) của A (từ a_{i+1} tới a_n) thì sẽ nhận được kết quả của phép toán hiệu tọa độ (#). Chú ý : đây là một phép toán thể hiện sự "gép" ký hiệu chữ số khá điển hình, cách làm giống như khi làm tính cộng (9 + 9) = 18, thì kết quả 18 chính là "sự gép" của hai ký hiệu là ký hiệu 1 và ký hiệu 8 lại với nhau, vấn đề còn lại chỉ là việc gép các ký hiệu thành kết quả (đáp số) của phép toán (#) có ý nghĩa gì là được. Chính là hiểu được kết quả của phép toán.

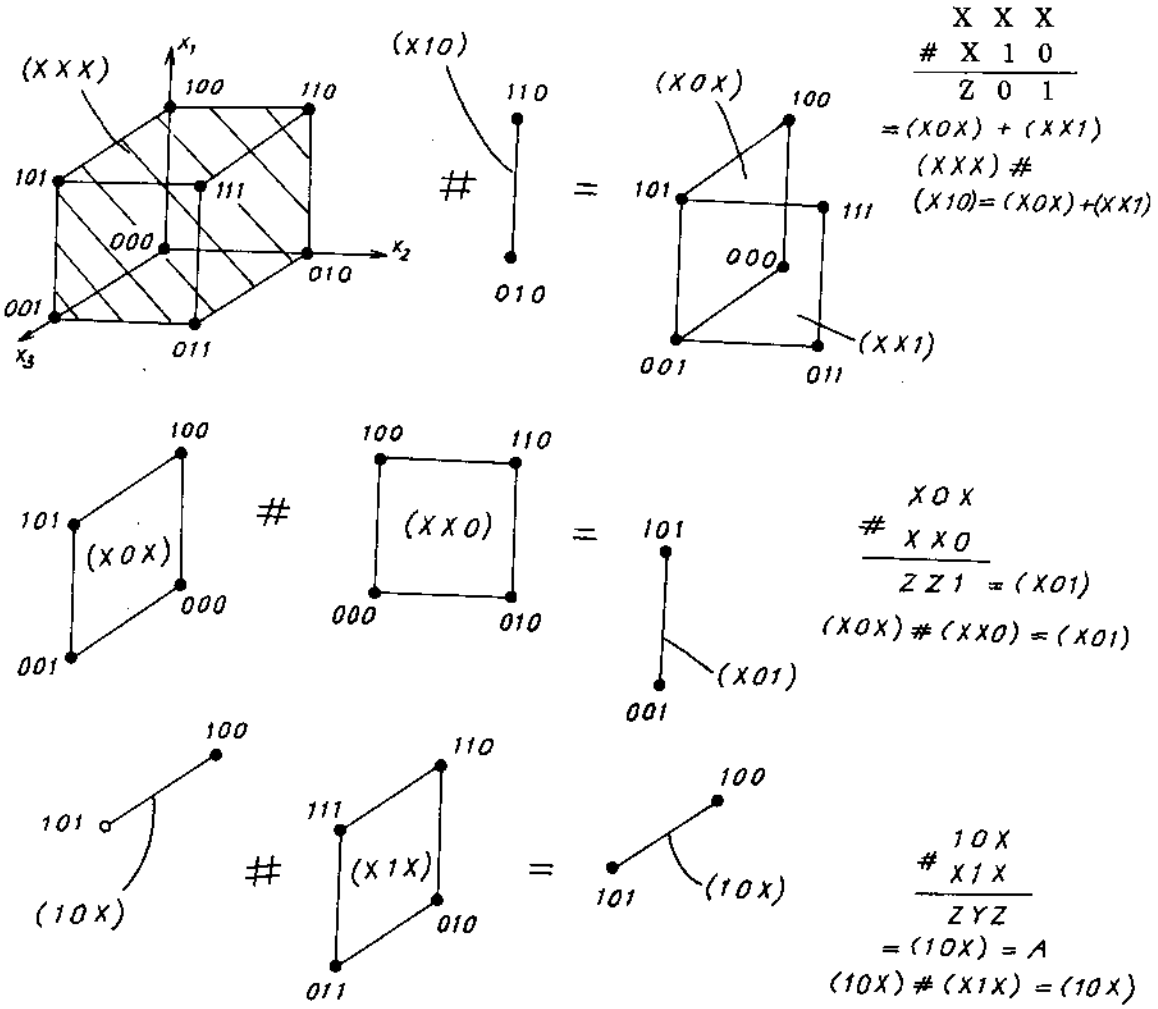
Có thể tính tiếp ở các ví dụ sau khi α_i xuất hiện giá trị 0 hoặc 1 có số lần nhiều hơn.

$$\begin{array}{r} 1\ X\ X\ 0 \\ \#\ X\ 1\ 0\ 0 \\ \hline Z\ 0\ 1\ Z \end{array} = (1\ 0\ X\ 0) \cup (1\ X\ 1\ 0) = (10X0) + (1X10)$$

Đáp số của phép toán ta cũng có thể "tính" hay nhận được rất nhanh theo cách làm sau.

$$\begin{array}{r} 1\ X\ X\ 0 \\ \#\ X\ 1\ 0\ 0 \\ \hline Z\ 0\ 1\ Z \end{array} \quad = (1\,0\,X\,0) + (1\,X\,1\,0)$$

Để hiểu được ý nghĩa và công dụng của một phép toán mới định nghĩa có tên là hiệu tọa độ (#) cũng như việc "ghép" các ký hiệu cho ta kết quả của phép toán này như thế nào, có thể dựa vào hình 1.86 và các kết quả tính toán của nó như sau :



Hình 1.86. Minh họa hoạt động của phép toán hiệu tọa độ.

Nếu không có cách biểu diễn hàm Boole dưới dạng hình học, và nếu kết quả của phép toán hiệu tọa độ (#) không biểu diễn được dưới dạng hình vẽ, thì ta rất khó có thể hình dung để hiểu được bản chất cũng như ý nghĩa của phép toán (#) là gì. Hình vẽ được mô tả trên hình 1.86 giúp ta dễ dàng nhận thấy bản chất của phép toán hiệu

tọa độ (#) giữa hai implicăng A và B, đó là : loại bỏ khỏi số bị trừ A đi giá trị của số trừ B, đồng thời những implicăng (hay những quan hệ toán học nào đó) hơi "dính dáng" hay có chút liên hệ với B đều bị loại trừ theo. Còn nếu giữa số bị trừ A không có liên quan gì tới số trừ B, thì khi trừ đi nhau kết quả vẫn là A.

Các tính chất của phép hiệu tọa độ (#) :

- Không có tính chất giao hoán :

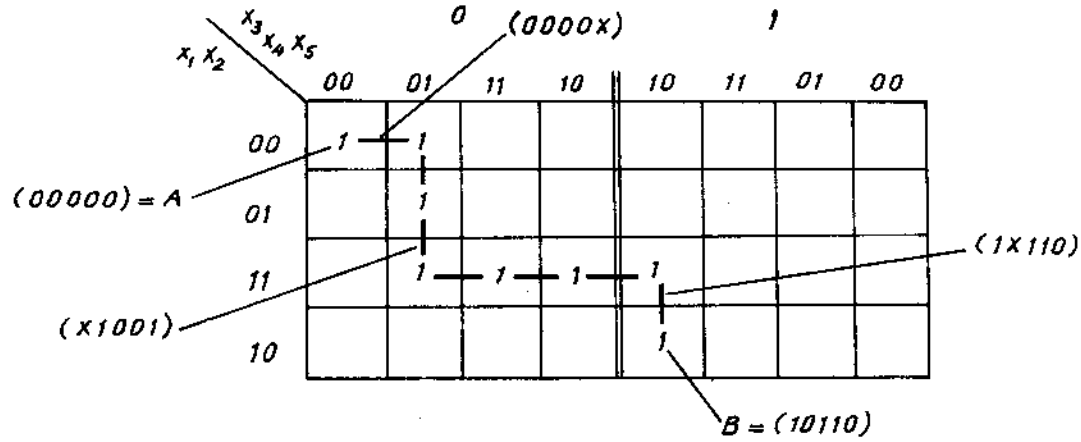
$$A \# B \neq B \# A$$

- Nếu $A \subseteq B$ thì $A \# B = \Phi$

Phép toán hiệu tọa độ (#) đã được nói đến ở các phần trên, ở các phần đó ta chỉ công nhận có phép toán (#) để mà dùng trong các thuật toán tìm phủ C_{min}, đến đây chúng ta đã đưa ra được định nghĩa của phép hiệu tọa độ (#), đồng thời đã xây dựng được cụ thể phép toán mới này. Phép toán (#) được định nghĩa ra nó có tính chất cũng như hiệu ứng riêng khi biến đổi toán học, khi đã hiểu rõ phép hiệu tọa độ (#) ta có thể dùng nó vào mục đích biến đổi toán học riêng của mình.

Đến đây ta đã xét định nghĩa ba phép toán mới theo yêu cầu của kỹ thuật, đó là phép toán "sao tọa độ" (*), phép toán "giao tọa độ" ($\cap$) và phép toán "hiệu tọa độ" (#). Đó là các phép toán mà Roth đã định nghĩa ra để áp dụng vào lý thuyết hay thuật toán tìm phủ C_{min} của hàm lôgic, mỗi một phép toán đó có hiệu ứng riêng của mình, khi hiểu chúng ta hoàn toàn có thể dùng chúng để rút gọn hàm Boole. Ở đây ngoài phương pháp hay cách tìm phủ C_{min} của hàm lôgic, ta còn hiểu được cách thức để tạo ta hay định nghĩa ra một phép toán mới, đó là khái niệm quan trọng. Muốn tạo ra một phép toán mới ta chỉ cần thay đổi bảng quan hệ ký hiệu và điều kiện thực hiện quan hệ ký hiệu của nó, vì vậy để tạo ra một phép toán mới là không khó. Vấn đề còn lại là trả lời câu hỏi phép toán đó dùng vào đâu nữa mà thôi.

Đến đây phép toán hiệu tọa độ (#) đã có, vậy để có thể hiểu trọn vẹn thuật toán của Roth và công dụng quan trọng của phép (#) trong thuật toán tìm phủ C_{min} ta có thể dựa vào ví dụ cho một hàm lôgic năm biến minh họa trên hình 1.87.



Hình 1.87. Bài bài toán lôgic minh họa.

Từ bài toán đã cho ta có số liệu đầu vào là phủ C_0 của hàm số là :

$$C_0 = ((00000)\ (00001)\ (01001)\ (11001)\ (11011)\ (11010)\ (11110)\ (10110)).$$

Dựa vào phủ C_0 của hàm lôgic đã cho có thể tìm được không gian các implicăng đơn giản Z nhờ sự giúp đỡ của phép toán "sao tọa độ" (*), bước tìm "không gian Z" các implicăng đơn giản đã được trình bầy kỹ ở phần trên, ở đây không nói kỹ tới cách làm cụ thể mà chỉ đưa ra kết quả tìm được không gian Z của bài toán đã cho là :

$$Z = ((000X)\ (0X001)\ (X1001)\ (110X1)\ (1101X)\ (11X10)\ (1X110))$$

Từ Z nhờ khai triển các implicăng đơn giản và loại bỏ đi các đỉnh xuất hiện ít nhất là hai lần (có hai đỉnh giống nhau từ hai implicăng đơn giản khác nhau) ta tìm được các đỉnh đầu mút hay không gian L của bài toán đề ra là :

$$L = ((00000)\ (10110)) = (A,\ B)$$

Đến đây đã có hai tập số liệu vào quan trọng đó là tập không gian Z các implicăng đơn giản và tập L các đỉnh đầu mút của bài toán.

$$Z_1 = Z = ((0000X)\ (0X001)\ (X1001)\ (110X1)\ (1101X)$$
$$(11X10)\ (1X110)) = (z_1,\ z_2,\ldots,\ z_i,\ldots,\ z_n)$$
$$L_1 = L = ((00000)\ (10110)) = (l_1,\ l_2,\ldots,\ l_n)$$

Ta đặt ký hiệu Z_1 và l_1 ứng với quá trình tìm không gian các khối đầu mút (extrêman) lần thứ nhất (E_1). Dựa vào định lý 91 đã nêu ở phần trên $z_i \in Z$ là implicăng đơn giản nếu thỏa mãn quan hệ :

$$\text{nếu } (z_i \cap L) \text{ tồn tại } (z_i \cap l_j) \neq \Phi \text{ thì } z_i \in E$$

Ta có kết quả từ ví dụ đề ra như sau

$z_1 = (0000X)$

$$(z_1 \cap L_1) = \begin{cases} (0000X) \cap (00000) = (00000) \neq \Phi \\ (0000X) \cap (10110) = \Phi \end{cases}$$

Kết luận $z_1 = (000X0)$ là khối đầu mút nên $z_1 \in E_1$

Cách làm tương tự có :

$z_2 \cap L_1 = \varnothing$; $z_3 \cap L_1 = \varnothing$...

$z_7 = (1X110)$

$$(z_7 \cap L_1) = \begin{cases} (1X110) \cap (00000) = \varnothing \\ (1X110) \cap (10110) = (10110) \neq \varnothing \end{cases}$$

Từ kết quả thu được ta có $z_7 = (1X110)$ là khối đầu mút $z_7 \in E_1$.

Quá trình tính toán tìm không gian các khối đầu mút lần thứ nhất (E_1) cho ta kết quả :

$$E_1 = ((0000X)\ (1X110))$$

Sau khi tìm được không gian các khối đầu mút (E_1) là các implicăng đơn giản thực sự - các implicăng đơn giản không thể thiếu được trong phủ C_{min} của hàm lôgíc trong bước thứ nhất của quá trình tìm không gian các khối đầu mút. Bước thứ 2 tiếp theo ta phải tìm phủ mới Z_2 để từ đó tìm tiếp không gian các khối đầu mút E_2. Để tìm được Z_2 ta dùng phép toán hiệu tọa độ (#) theo quan hệ sau :

$$Z_2 = Z_1 \# E_1$$

Bước tìm Z_2 chính là ứng dụng quan trọng của phép toán hiệu tọa độ (#) trong thuật toán tìm phủ C_{min} của Roth. Để hiểu kỹ thuật ứng của (#) trong quá trình tìm Z_2 có thể căn cứ vào ví dụ đã cho khi có hai tập :

Z_1 = ((000X) (0X001) (X1001) (110X1) (1101X) (11X10) (1X110))

E_1 = ((0000X) (1X110))

Quan hệ Z_1 hiệu tọa độ (#) đi E_1 ($Z_1 \# E_1$) thực hiện theo "tích Đề các" vì là hai tập tương tác với nhau như sau :

z_1 = (0000X)

$$(z_1 \# E_1) = \begin{cases} (0000X) \# (0000X) = \varnothing \\ (0000X) \# (1X110) = (0000X) \end{cases}$$

do có kết quả ($z_1 \# E_1$) = $\varnothing$ (ở e_i^1 bất kỳ $\in E_1$) nên $z_1 \notin Z_2$ (z_1 không thuộc Z_2)

z_2 = (0X001)

$$z_2 \# E_1 = \begin{cases} (0X001) \# (0000X) = (01001) \\ (0X001) \# (1X110) = (0X001) \end{cases}$$

z_3 = (X1001)

$$z_3 \# E_1 = \begin{cases} (X1001) \# (0000X0 = (X1001) \\ (X1001) \# (1X110) = (X1001) \end{cases}$$

z_4 = (110X1)

$$z_4 \# E_1 = \begin{cases} (110X1) \# (0000X) = (110X1) \\ (110X1) \# (1X110) = (110X1) \end{cases}$$

z_5 = (1101X)

$$z_5 \# E_1 = \begin{cases} (1101X) \# (0000X) = (1101X) \\ (1101X) \# (1X110) = (1101X) \end{cases}$$

z_6 = (11X10)

$$z_6 \# E_1 = \begin{cases} (1X10) \# (0000X) = (11X10) \\ (11X10) \# (1X110) = (11010) \end{cases}$$

z_7 = (1X110)

$$z_7 \# E_1 = \begin{cases} (1X110) \# (0000X) = (1X110) \\ (X110) \# (1X110) = \varnothing \end{cases}$$

Như vậy (z_7 # E_1) xuất hiện kết quả ∅ chứng tỏ là $z_7 \notin Z_2$ (z_7 không thuộc Z_2)

Sau khi tính toán xong sẽ thu được kết quả $\hat{Z}_2$ là phủ chưa rút gọn như sau :

$$\hat{Z}_2 = ((01001)\ (X1001)\ (110X1)\ (1101X)\ (11010)$$

Tiếp theo ta rút gọn $\hat{Z}_2$ bằng cách loại khỏi $\hat{Z}_2$ các implicăng nằm trong implicăng khác (khối khác) sẽ thu được Z_2 :

$$Z_2 = \hat{Z}_2 - Z_2^*$$

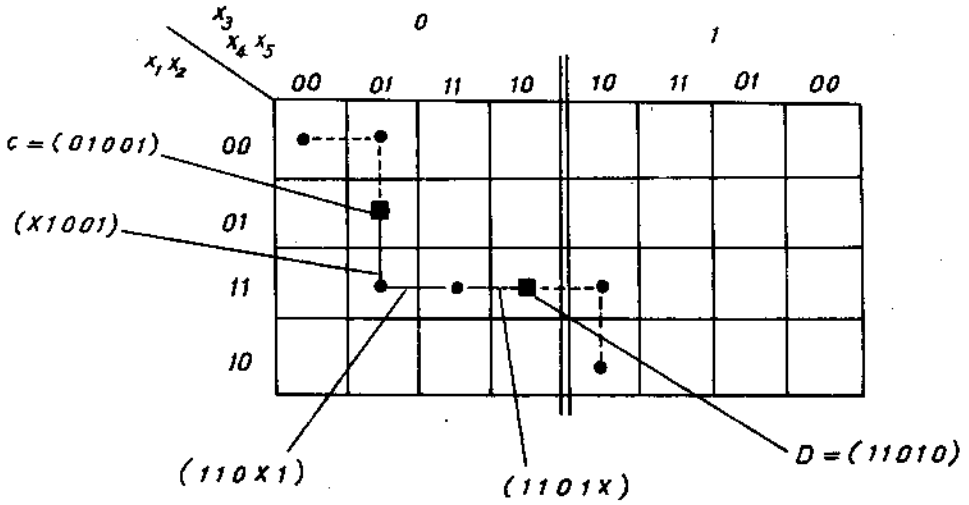
trong đó : Z_2^* là các implicăng nằm trong các implicăng khác.

(ví dụ (01001) ⊆ (X1001))

Kết quả ta thu được phủ mới Z_2 :

$$Z_2 = ((X1001)\ (110X1)\ (1101X))$$

Điều đó được minh họa trên hình 1.88.



Hình 1.88. Kết quả tìm không gian Z_2.

Từ phủ mới Z_2 dễ dàng tìm được L_2 là tập các đỉnh đầu mút lần thứ hai, hiệu ứng của phép toán hiệu tọa độ (#) giống như một nhát "chặt" cho phép tạo ra được đỉnh đầu mút mới từ đó tìm được các khối đầu mút mới (các extrêman) các implicăng đơn giản không thể thiếu trong phủ C_{min}.

Ta có tập số liệu mới :

$$Z_2 = ((X1001)\ (110X1)\ (1101X))$$

$$L_2 = ((01001)\ (11010) = (C,\ D)$$

Từ hai tập số liệu Z_2 và L_2 kết hợp với phép toán giao tọa độ (∩) sẽ dễ dàng tìm

được tập các extrêman E_2 hay các implicăng đơn giản thực sự - các khối đầu mút lần thứ hai của hàm lôgic đã cho là :

$$E_2 = ((X1001)\ (1101X))$$

Tiếp theo tìm tiếp phủ mới Z_3 các implicăng đơn giản, để từ đó tìm được không gian các khối đầu mút lần thứ ba E_3, thép quan hệ sau :

$$Z_3 = Z_2 \ \# \ E_2$$

Tới đây ta có tập số liệu :

$$Z_2 = ((X1001)\ (110X1)\ (1101X))$$

$$E_2 = ((X1001)\ (1101X))$$

Phủ mới Z_3 được xác định :

$$Z_3 = Z_2 \ \# \ E_2 = \varnothing$$

Tới đây kết thúc quá trình tìm không gian các khối đầu mút. Như vậy quá trình tìm phủ mới Z_{n+1} thực hiện mãi cho đến khi :

$$Z_{n+1} = Z_n \ \# \ E_n = \varnothing$$

thì dừng quá trình tìm E, kết quả cuối cùng thu được sẽ là :

$$C_{min} = E = \bigcup_{i}^{n} E_i$$

trong đó E_i các khối đầu mút của lần thứ i trong quá trình tìm phủ C_{min}

Với bài toán đã cho ta có kết quả :

$$E = E_1 + E_2$$

$$C_{min} = E = ((0000X)\ (1X110)\ (X1001)\ (1101X))$$

$$f_{min}(x_1, x_2, x_3, x_4, x_5) = \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 + x_1 x_3 x_4 \bar{x}_5 + x_2 \bar{x}_3 \bar{x}_4 x_5 + x_1 x_2 \bar{x}_3 x_4$$

Kết quả thu được phu C_{min} theo phương pháp của Roth từ bài toán lôgic đề ra, ta cũng sẽ dễ dàng nhận được kết quả đó dựa theo các phương pháp rút gọn hàm Boole của Mc.Cluskey hay từ bìa Kácnô.

Đến đây ta cũng kết thúc thuật toán tìm phủ C_{min} của Roth. Từ thuật toán đưa ra của Roth có thể dễ dàng lập trình cho máy tính để tự động quá trình tối tiểu hoá (rút gọn) hàm lôgic số lượng biến lớn. Để tự động rút gọn hàm Boole theo phương pháp của Rooth, tác giả đã xây dựng được một chương trình viết bằng ngôn ngữ LISP, chương trình này có thể thực hiện việc rút gọn hàm Boole có số lượng biến vào tùy ý kết quả cũng giống như chương trình rút gọn hàm lôgic XPRESO của Berkerley viết bằng ngôn ngữ C++ theo phương pháp thuật toán.

Theo phương pháp của Roth có thể lập trình để tự động tìm ra phủ C_{min} của một hàm lôgic bất kỳ, nhưng kết quả rút gọn hàm Boole của Rôth cũng không quan trọng bằng từ lý thuyết của Roth, hiểu được cách (phương pháp) tự tạo ra hay tự định nghĩa ra một phép toán mới, cho phép giải quyết một nhiệm vụ kỹ thuật nào đó, cũng như có

thể tạo nên một lý thuyết giải quyết một nhiệm vụ được đề ra. Phương pháp Roth là một phương pháp lý thuyết (không phải phương pháp thuật toán) cho phép ta ngoài kết quả là phủ C_{min} thu được, còn có thể thu thêm được các thông tin khác nữa, như đánh giá được chất lượng hoạt động của một chức năng kỹ thuật (hàm lôgic) hay độ tin cậy của mạch lôgic. Khi tự định nghĩa được một phép toán (như phép toán hiệu tọa độ #) cho phép thay đổi cấu trúc của một hàm số, điều đó sẽ mở ra một trong những khả năng thực hiện giải song song bài toán lôgic.

Chương II

ĐẠO HÀM VÀ VI PHÂN HÀM BOOLE

Chúng ta đã xem xét việc thực hiện rút gọn hàm Boole, sau khi đã thu được kết quả hàm Boole ở dạng tối thiểu hóa ta sẽ tiến hành lắp ráp mạch điện dựa trên dạng tối thiểu hóa thu được. Mạch điện được xây dựng sẽ ở dạng đơn giản nhất thực hiện được chức năng của hàm lôgic đã đề ra. Đó chính là quá trình thiết kế hay tổng hợp tạo ra một chức năng lôgic nói chung. Nhưng khi chức năng lôgic đã được lắp ráp thành mạch điện và đi vào hoạt động thì ta lại cần đánh giá được chất lượng hay "độ tin cậy" trong quá trình hoạt động của nó. Đánh giá chất lượng hoạt động của mạch lôgic là một công việc phức tạp và không dễ thực hiện vì hàm lôgic thường có nhiều biến vào, và phụ thuộc vào chất lượng các linh kiện lắp ráp mạch điện. Một trong những công cụ có thể sử dụng để đánh giá chất lượng cũng như độ tin cậy của hệ thống số là dựa vào đạo hàm và vi phân hàm Boole.

Khái niệm về đạo hàm và vi phân hàm Boole ra đời vào cuối thập kỷ 50 của thế kỷ trước. Cũng như đối với hàm liên tục, khi khảo sát đạo hàm của hàm liên tục người ta tìm ra được những điểm đặc trưng hay những điểm đặc biệt của hàm số đó như điểm uốn, cực trị, bước nhảy v.v..., thì ở hàm rời rạc (làm Boole) kết quả của đạo hàm cũng thông báo cho ta một số tính chất của hàm số lôgic đó hay các giá trị đặc trưng của nó.

Những tính chất hay các giá trị đặc trưng của hàm lôgic, cũng chính là các tính chất hay các giá trị đặc trưng của mạch điện. Do đó ta có thể áp dụng đạo hàm vào kỹ thuật, trong lĩnh vực chẩn đoán dự báo sai nhầm, đánh giá tin cậy của hệ thống điều khiển lôgic, dùng trong các hệ thống tự động, phát hiện sai và tự sửa sai, dùng trong lý thuyết nhận dạng v.v...

§2.1. ĐẠO HÀM RIÊNG CỦA HÀM BOOLE

Để hiểu đạo hàm của hàm Boole cho ta khả năng đánh giá được độ tin cậy hay chất lượng hoạt động của mạch lôgic thế nào thì cần phải nắm vững bản chất của đạo hàm. Một trong những khái niệm cơ bản và quan trọng nhất của hàm liên tục là đạo hàm của hàm số mà chúng ta rất quen thuộc là :

$$\frac{df(x)}{dx} = \lim_{\Delta x \to 0} \frac{\Delta f(x)}{\Delta x}$$

Ta có thể hiểu một cách đơn giản hơn là đạo hàm chính là sự "biến động" hay "biến cố" của một hàm số, của một quan hệ, mà sự biến động hay biến cố cho ta các thông tin để có các kết luận cần thiết. Vấn đề ở đây là có thể đưa khái niệm đạo hàm vào hàm rời rạc (hàm Bole) được không? Vì việc đạo hàm hàm rời rạc chính là đạo hàm một mạch điện.

Đối với hàm Boole sự biến động hay biến đổi của hàm số nó cũng chính là sự biến động hay thay đổi của tín hiệu ở đầu vào của mạch điện, khi có sự thay đổi (đạo hàm) của tín hiệu vào thì mạch lôgíc sẽ phải có đáp ứng tương ứng ra của nó, mà đáp ứng ở đầu ra lại chính là kết quả thay đổi của tín hiệu vào và đó cũng chính là thông tin mà ta có thể dựa vào nó để đánh giá được chất lượng hoạt động của mạch.

2.1.1. CÁC ĐỊNH NGHĨA CƠ BẢN

Định nghĩa 2.1.1 : Cho f(x) là một hàm Boole của một biến số Boole x, đạo hàm của hàm Boole f(x) một biến đối với biến x được định nghĩa như sau :

$$\frac{df(x)}{dx} = f(x) \oplus f(\bar{x})$$

Gia số của biến số Δx chính là sự thay đổi hay biến động của biến x từ x đến $\bar{x}$, có nghĩa là

$$dx = \Delta x = x \oplus \bar{x} = 1$$

chính là bước nhẩy của x từ 0 đến 1 và ngược lại, nên gia số của giá trị biến đổi đó là 1.

Từ sự biến đổi của biến số hay gia số của biến số sẽ xuất hiện sự thay đổi của hàm số hay gia số của hàm số là:

$$df(x) = \Delta f(x) = f(x) \oplus f(\bar{x})$$

Ở đây phép toán "hiệu đối xứng" chính là phép toán "cộng môđun" ($\oplus$), dựa vào phép toán này ta tìm ra được gia số hàm số $\Delta f(x)$, và gia số biến số Δx. Ta chú ý rằng trong toán học rời rạc - toán học hai trị trong đại số Boole, thì khái niệm về giới hạn hay tiệm cận không tồn tại, do đó dx và Δx có chung một giá trị. Khi sử dụng phép toán "cộng môđun" ($\oplus$) hay "hiệu đối xứng" trong đại số Boole giữa hai biến số a và b ta có quan hệ : $(a \oplus b) = (\bar{a}b + a\bar{b})$. Áp dụng quan hệ đó để tính đạo hàm cho hàm số trong ví dụ sau :

Ví dụ : Cho $f(x) = ax + b\bar{x}$, trong đó $a, b \in B = \{ 0, a, b, 1\}$ ở đây vì là đại số hai trị nên $a = \bar{b}$.

Vậy ta có :

$$\frac{df(x)}{dx} = (ax + b\bar{x}) \oplus (a\bar{x} + bx) = 1$$

Áp dụng theo định nghĩa và phép toán cộng môđun sẽ cho ta kết quả của phép tính đạo hàm hàm số đã cho trên hình 2.1

x	f(x)	$f(\bar{x})$	df(x)/dx
0	b	a	1
a	1	0	1
b	0	1	1
1	a	b	1

$1 = \bar{0}$
$a = \bar{b}$

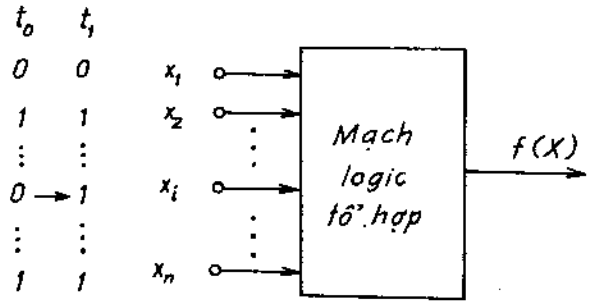
Hình 2.1. Kết quả phép tính đạo hàm.

Trong thực tế hàm Boole một biến hay chỉ có một tín hiệu vào (giống như một công tắc đóng mở) ít gặp, mà thông thường một mạch lôgíc sẽ có số lượng đầu vào nhiều hơn. Từ đó chúng ta đưa ra đưa ra định nghĩa đạo hàm Boole nhiều biến hay đạo hàm riêng hàm Boole trong trường hợp tổng quát như sau.

Định nghĩa 2.1.2 : Cho f (X) là hàm Boole n biến [$X = (x_1, x_2, ..., x_n)$]. Đạo hàm riêng của hàm f (X) đối với biến x_i ($1 \leq i \leq n$) được định nghĩa như sau :

$$\frac{\partial f(X)}{\partial x_i} = f(x_1, \ldots, x_i, \ldots, x_n) \oplus f(x_1, \ldots, \bar{x}_i, \ldots, x_n)$$

Với phương trình toán học nghĩa đạo hàm riêng theo biến x_i ta có thể hiểu theo quan điểm của mạch điện là : chỉ có tín hiệu vào ở chân thứ i thay đổi giá trị từ 0 sang 1 ($0 \rightarrow 1$) hoặc ngược lại, còn các chân tín hiệu vào khác của mạch điện vẫn giữ nguyên giá trị lôgíc trước đó. Khi có một tín hiệu vào thay đổi giá trị thì mạch điện sẽ có đáp ứng ra trên đầu ra của nó, mà dựa vào đó ta có thông tin để đánh giá ngược lại chất lượng của mạch. Như vậy đạo hàm riêng của hàm Boole nhiều biến chính là thay đổi một tín hiệu vào trên một đầu vào bất kỳ của mạch điện, ta có thể minh họa trên hình 2.2.



Hình 2.2. Đạo hàm ứng với thay đổi tín hiệu vào.

2.1.2. CÁC CÁCH TÍNH ĐẠO HÀM RIÊNG HÀM BOOLE

2.1.2.1. Phương pháp tính theo đại số

Ví dụ : Cho hàm số $f(X) = f(x_1, x_2, x_3) = x_1 x_2 + x_3$, tính đạo hàm riêng của $f(X)$ theo biến x_1.

Phương pháp tính theo đại số là áp dụng theo đúng định nghĩa của đạo hàm (chú ý dùng quan hệ cộng môđun ($\oplus$) và định lý Demorgan.

$$\frac{\partial f(X)}{\partial x_1} = (x_1 x_2 + x_3) \oplus (\bar{x}_1 x_2 + x_3) = A \oplus B = \bar{A}.B + A.\bar{B}$$

$$= \overline{(x_1 x_2 + x_3)}\,(\bar{x}_1 x_2 + x_3) + (x_1 x_2 + x_3)\,\overline{(\bar{x}_1 x_2 + x_3)} =$$

$$= \overline{(x_1 x_2 . \bar{x}_3)}\,(\bar{x}_1 x_2 + x_3) + (x_1 x_2 + x_3)\,\overline{(\bar{x}_1 x_2 . \bar{x}_3)} =$$

$$= \left((\bar{x}_1 + \bar{x}_2) . \bar{x}_3\right)(\bar{x}_1 x_2 + x_3) + (x_1 x_2 + x_3)\left((x_1 + \bar{x}_2) . \bar{x}_3\right) =$$

$$= (\bar{x}_1 \bar{x}_3 + \bar{x}_2 \bar{x}_3)(\bar{x}_1 x_2 + x_3) + (x_1 x_2 + x_3)(x_1 \bar{x}_3 + \bar{x}_2 \bar{x}_3) =$$

$$= \bar{x}_1 x_2 \bar{x}_3 + \bar{x}_1 x_3 + \bar{x}_2 x_3 + x_1 x_2 \bar{x}_3 + x_1 x_3 + \bar{x}_2 x_3 =$$

$$= \bar{x}_1 x_2 \bar{x}_3 + x_1 x_2 \bar{x}_3 = (\bar{x}_1 + x_1) x_2 \bar{x}_3 = x_2 \bar{x}_3$$

Kết quả :
$$\frac{\partial (x_1 x_2 + x_3)}{\partial x_1} = x_2 \bar{x}_3$$

Kết quả tính toán này cũng có thể nhận được bằng phương pháp khác có tên là phương pháp đồ thị hay dùng bìa Kácnô.

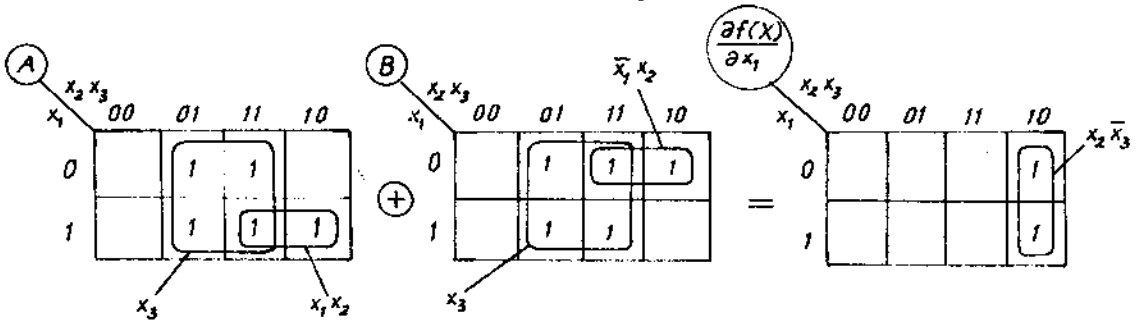
2.1.2.2. Tính đạo hàm riêng của hàm f(X) bằng phương pháp bìa Kácnô

Phương pháp này được minh họa trên hình 2.3

Vẫn với ví dụ hàm $f(X) = x_1 x_2 + x_3$ ta có :

$$\frac{\partial f(X)}{\partial x_1} = (x_1 x_2 + x_3) \oplus (\bar{x}_1 x_2 + x_3) = A \oplus B = x_2 \bar{x}_3$$

Ta ánh xạ A và B vào bìa Kácnô được kết quả như trên hình 2.3.



Hình 2.3. Đạo hàm theo phương pháp đồ thị.

Sau khi ánh xạ A = (x_1 x_2 + x_3) và B = ($\bar{x}_1$ x_2 + x_3) vào bìa Kácnô, ta thực hiện cộng môđun (⊕) hai bìa đó lại với nhau (chồng khít hai bìa K đó lại với nhau) sẽ thu được kết quả như ở bìa K đáp số. Kết quả thu được giống như kết quả tính toán theo phương pháp đại số.

Để tính được đạo hàm riêng của hàm f(X) theo biến x_i ta còn có thể dựa vào định lý, định lý này xuất phát từ đặc điểm của hàm Boole chỉ có hai giá trị là 0 và 1, cho nên khi tính đạo hàm riêng của f (X) theo x_i ta có thể thay gái trị của x_i là 0 hoặc 1 vào trong công thức như sau :

2.1.2.3. Tính đạo hàm riêng của hàm f(X) theo định lý

Định lý 2.1.1 : Đạo hàm riêng của hàm f (X) trong đó X = (x_1, x_2, ..., x_4) theo biến x_i với (1 ≤ i ≤ n) có thể được tính theo công thức sau :

$$\frac{\gamma f(X)}{\gamma x_i} = f(x_1, \ldots, \underset{i}{1}, \ldots, x_n) \oplus f(x_1, \ldots, \underset{i}{0}, \ldots, x_n)$$

Định lý này dựa trên đặc điểm của hàm Boole có hai trị của một biến x_i (nên x_i = 1 thì $\bar{x}_i$ = 0), vậy có thể thay giá trị 0 và 1 của biến x_i vào định nghĩa tính đạo hàm riêng của hàm f (X) theo x_i sẽ thu được công thức như công thức của định lý. Kết quả thu được của định nghĩa cũng giống như của định lý là điều đơn giản và dễ hiểu. Trong thực tế thì khi đạo hàm theo biến x_i là chân đầu vào thứ i của mạch điện sẽ có giá trị lôgic là 1 khi nối lên nguồn + E và là 0 khi nối xuống đất.

Áp dụng công thức tính đạo hàm riêng của hàm f (X) theo biến x_i cho ví dụ đã nêu ở trên ta có :

$$f(X) = f(x_1, x_2, x_3) = x_1 x_2 + x_3$$

$$\frac{\partial f(X)}{\partial x_i} = (1 . x_2 + x_3) \oplus (0 . x_2 + x_3) = x_2 \bar{x}_3$$

$$= (x_2 + x_3) \oplus x_3$$

$$= \overline{(x_2 + x_3)} . x_3 + (x_2 + x_3) . \bar{x}_3 =$$

$$= (\bar{x}_2 . \bar{x}_3) . x_3 + x_2 \bar{x}_3 = x_2 \bar{x}_3 .$$

Đây là một định lý quan trọng vì có thể dựa vào định lý để lập trình cho máy tính theo phương pháp thuật toán cho phép tìm đạo hàm riêng hàm Boole, đó sẽ là một trong những cơ sở quan trọng để đánh giá được chất lượng của mạch cũng như đánh giá được độ tin cậy của hệ thống lôgic một cách tự động.

Đến đây chúng đã có ba phương pháp tính đạo hàm riêng của hàm f (X) theo biến x_i, đó cũng chính là hiện tượng xảy ra khi một mạch lôgic có nhiều đầu vào (từ đầu vào x_1, x_2,..., đến đầu vào x_n) mà giá trị lôgic ở chân thứ i ứng với đầu vào x_i thay đổi giá trị (từ 1 về 0 hoặc ngược lại từ 0 về 1). Như vậy ta chỉ thay đổi giá trị của đầu vào x_i

có một lần. Nhưng trong thực tế không có mạch lôgíc nào chỉ thay đổi tín hiệu vào có một lần (ứng với đạo hàm riêng), mà tín hiệu vào của một mạch điện có thể được thay đổi kế tiếp nhau nhiều lần hết từ đầu vào này đến đầu vào khác (hết chân đầu vào tín hiệu này sang chân đầu vào tín hiệu kia), đồng thời cũng có thể thay đổi tín hiệu lôgic vào ở nhiều đầu vào (nhiều chân) cùng một lúc, theo yêu cầu của chức năng kỹ thuật đề ra. Từ thực tế thay đổi tín hiệu vào của mạch lôgíc như vậy ta có khái niệm hay đưa ra khái niệm đạo hàm bậc cao của hàm Boole. Do cách thay đổi tín hiệu vào của mạch điện có thể phân ra làm hai loại là : lần lượt các tín hiệu vào thay đổi giá trị lôgic ta đưa ra khái niệm "đạo hàm riêng bậc cao loại I", còn trường hợp nhiều đầu vào (nhiều tín hiệu vào) cùng thay đổi giá trị lôgíc một lần thì ta có khái niệm 'đạo hàm riêng bậc cao loại II".

2.1.3. ĐẠO HÀM RIÊNG BẬC CAO CỦA HÀM BOOLE

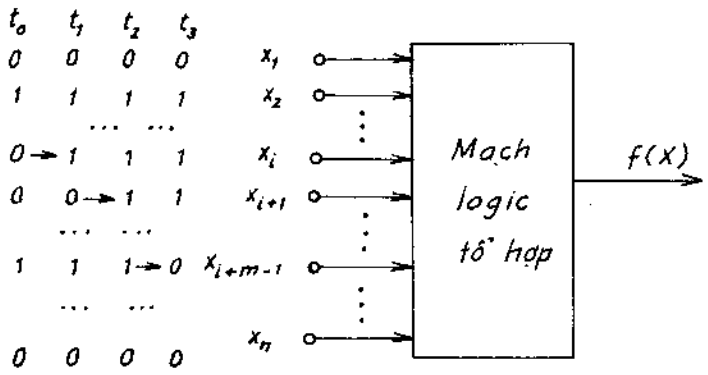
2.1.3.1. Đạo hàm riêng bậc cao loại I

Định nghĩa 2.1.3 : Đạo hàm riêng bậc cao loại I của hàm f(X) với $X = (x_1, x_2,..., x_n)$ đối với m biến từ biến $X_i = (x_i, x_{i+1},..., x_{i+m-1})$được định nghĩa như sau : $(0 \leq m \leq n)$

$$\frac{\partial^m f(X)}{\partial^m X_i} = \frac{\partial^m f(X)}{\partial x_i \, \partial x_{i+1} \cdots \partial x_{i+m-1}} = \frac{\partial}{\partial x_i}\left(\frac{\partial}{\partial x_{i+1}}\left(\cdots\frac{\partial f(X)}{\partial x_{i+m-1}}\right)\cdots\right)$$

Ở đây m là bậc của đạo hàm hay còn được hiểu là số lần thay đổi của tín hiệu vào trên tập đầu vào (X).

Đạo hàm bậc cao loại I có thể hiểu là nhiều tín hiệu vào thay đổi kế tiếp nhau, điều đó được minh họa trên hình 2.4.



Hình 2.4. Đạo hàm bậc cao trên mạch điện.

Qua công thức tính đạo hàm bậc cao loại I có thể hiểu được là : kết quả của đạo hàm riêng theo biến này (x_i) sẽ trở thành số liệu đầu vào (hàm ban đầu) cho đạo hàm riêng của biến tiếp theo (x_{i+1}). Quan điểm đó cứ được thực hiện cho đến hết ứng với bậc của đạo hàm là m (thực hiện m lần).

2.1.3.2. Đạo hàm riêng bậc cao loại II

Đây là một khái niệm không có và không tồn tại ở toán học cổ điển là đạo hàm nhiều biến một lúc, nhưng ở mạch lôgic tổ hợp hay hàm Boole việc thay đổi đồng thời tín hiệu ở trên nhiều đầu vào là hoàn toàn có thể được, và cách thay đổi (đạo hàm) nhiều tín hiệu vào (nhiều biến) một lúc là điều có thể xảy ra theo một nhu cầu kỹ thuật điều khiển nào đó. Ta xem xét khái niệm đạo hàm riêng bậc cao loại II của hàm Boole.

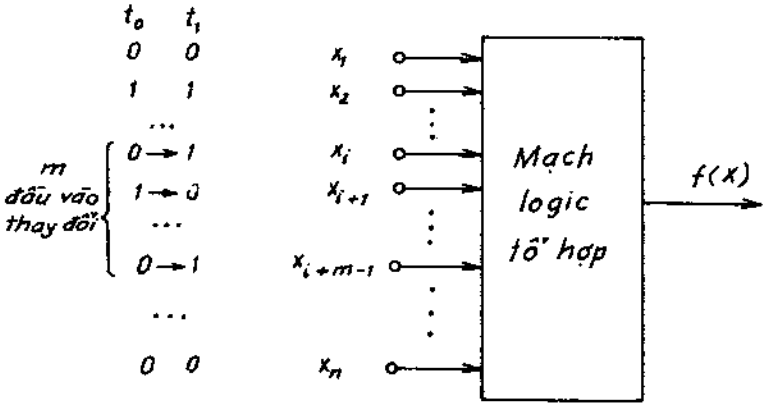
Định nghĩa 2.1.4. Đạo hàm riêng bậc cao loại II của hàm $f(X)$ với $X = (x_1, x_2, \ldots, x_n)$ theo m biến từ biến $X_i = (x_i, x_{i+1} \ldots, x_{i+m-1})$ được định nghĩa như sau :

$$(0 \leqslant m \leqslant n)$$

$$\frac{\partial^m f(X)}{\partial^m X_i} = \frac{\partial^m f(X)}{\partial(x_i, x_{i+1}, \ldots, x_{i+m-1})} = f(x_1, \ldots, x_i, x_{i+1}, \ldots, x_{i+m-1}, \ldots, x_n) \oplus$$

$$f(x_1, \ldots, \bar{x}_i, \bar{x}_{i+1}, \ldots, \bar{x}_{i+m-1}, \ldots, x_n)$$

Ở đây ta chỉ đạo hàm một lần liền một lúc với m biến số, hay chính là phủ định của các biến số đó. Đạo hàm bậc cao loại II có thể hiểu bằng mạch điện mô tả trên hình 2.5.



Hình 2.5. Đạo hàm bậc cao loại II trên mạch điện.

Có thể áp dụng cách tính đạo hàm riêng bậc cao loại I và loại II để tính đạo hàm cho hàm số lôgic ở ví dụ trên.

Ví dụ : Cho $f_3(X) = f(x_1, x_2, x_3) = x_1 x_2 + x_3$

a) Tính đạo hàm riêng bậc cao loại I cho hai biến số là x_1 và x_2. Tức là cho tín hiệu vào ở x_1 và x_2 lần lượt thay đổi.

$$\frac{\partial^2 f_3(X)}{\partial x_1 \partial x_2} = \frac{\partial}{\partial x_2}\left(\frac{\partial f_3(X)}{\partial x_1}\right) = \frac{\partial}{\partial x_2}(x_2 \bar{x}_3) = \bar{x}_3$$

Ở trên ta đã tính được $\dfrac{\partial f_3(X)}{\partial x_1} = x_2 \bar{x}_3$

$$\frac{\partial}{\partial x_2}(x_2\bar{x}_3) = x_2\bar{x}_3 \oplus \bar{x}_2\bar{x}_3 = A \oplus B = \bar{A}B + A\bar{B}$$

$$= \overline{(x_2\bar{x}_3)}(\bar{x}_2\bar{x}_3) + (x_2\bar{x}_3)\overline{(\bar{x}_2\bar{x}_3)} =$$

$$= (\bar{x}_2 + x_3)(\bar{x}_2\bar{x}_3) + (x_2\bar{x}_3)(x_2 + x_3) =$$

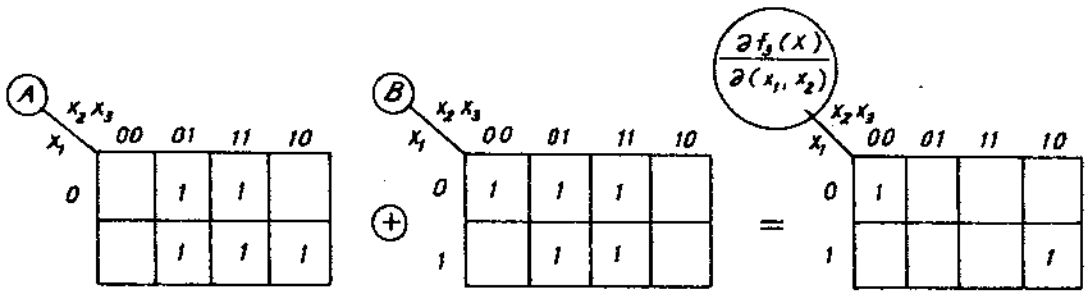
$$= \bar{x}_2\bar{x}_3 + x_2\bar{x}_3 = \bar{x}_3(\bar{x}_2 + x_2) = \bar{x}_3$$

Với kết quả thu được khí tính đạo hàm riêng bậc cao loại I cho hàm số $f_3(X)$ theo hai biến x_1 và x_2, ta hiểu được đáp ứng ra của mạch điện khi thay đổi giá trị lôgic của hai đầu vào tín hiệu x_1 và x_2 như thế nào.

b) Tính đạo hàm riêng bậc cao loại II theo hai biến số (x_1, x_2) như sau. Tức là cho tín hiệu vào ở x_1 và x_2 cùng thay đổi một lúc.

$$\frac{\partial f_3(X)}{\partial(x_1, x_2)} = (x_1x_2 + x_3) \oplus (\bar{x}_1\bar{x}_2 + x_3) = A \oplus B = \bar{x}_1\bar{x}_2\bar{x}_3 + x_1x_2\bar{x}_3$$

Ta tính theo phương pháp đồ thị ánh xạ A và B vào bìa Kácnô trên hình 2.6.



Hình 2.6. Đạo hàm bằng bìa Kácnô.

Kết quả : $$\frac{\partial f_3(X)}{\partial x_1, x_2)} = \bar{x}_1\bar{x}_2\bar{x}_3 + x_1x_2\bar{x}_3$$

Đạo hàm riêng bậc cao loại II của hàm Boole (mạch điện) theo hai biến (x_1, x_2), cho chúng ta biết đáp ứng ra của hàm số (phản ứng của mạch điện) khi ta thay đổi đồng thời giá trị lôgic của hai tín hiệu vào (x_1, x_2) trong hàm số. Dựa vào đáp ứng ra đó của mạch cũng chính là của hàm lôgic, ta có thể đánh giá được chất lượng hay độ tin cậy của chúng.

Từ ví dụ tính toán đạo hàm riêng bậc cao loại I và loại II ta có kết quả là :

$$\frac{\partial f_3(X)}{\partial x_1 \partial x_2} = \bar{x}_3$$

$$\frac{\partial f_3(X)}{\partial(x_1, x_2)} = \bar{x}_1\bar{x}_2\bar{x}_3 + x_1x_2\bar{x}_3$$

Từ hai kết quả đạo hàm nhận được đó là có thể có nhận xét đánh giá hoạt động của mạch (hàm Boole) là :

Khi đạo hàm riêng bậc cao loại I tức là ta thay đổi lần lượt giá trị lôgíc của x_1, sau đó mới thay đổi giá trị lôgíc của x_2, thì kết quả cho ta ($\bar{x}_3$) tương đương với một mạch điện đơn giản hơn mạch ban đầu (hàm số cho trước $f_3(X)$). Vậy có thể kết luận là nếu hoạt động lần lượt thay đổi giá trị của tín hiệu vào thì độ tin cậy khi hoạt động của mạch sẽ tăng lên, hay mạch sẽ hoạt động tốt hơn và an toàn hơn.

Còn khi đạo hàm riêng bậc cao loại II của hàm số $f_3(X)$ đã cho (chính là mạch điện có sẵn), tức là ta thay đổi đồng thời một lúc giá trị lôgíc của hai tín hiệu vào (x_1, x_2), cũng chính là ta thay đổi (đạo hàm) đồng thời hai biến số (x_1, x_2) của hàm lôgíc, kết quả ta thu được ($\bar{x}_1 \bar{x}_2 \bar{x}_3 + x_1 x_2 \bar{x}_3$), kết quả đó tương đương với mạch điện phức tạp hơn (ra một hàm lôgíc phức tạp hơn) mạch ban đầu đã cho (hàm $f_3(X)$. Từ đó ta có thể kết luận là nếu ta thay đổi đồng thời giá trị lôgíc của hai tín hiệu vào (x_1, x_2) của mạch điện đã cho thì độ tin cậy khi hoạt động của mạch sẽ giảm xuống, hay mạch sẽ hoạt động xấu hơn và kém an toàn.

Chú ý là ta chỉ có thể kết luận độ tin cậy hoạt động của mạch tăng lên hay độ tin cậy hoạt động của mạch giảm đi khi có kết quả cụ thể của đạo hàm hay chỉ có thể dựa vào kết quả của đạo hàm của mạch cụ thể khi đó ta mới được kết luận được là tốt hay xấu. Đồng thời kết quả của đạo hàm Boole cũng chỉ được coi là một trong những cơ sở hay dữ kiện mà dựa vào đó có thể đánh giá được chất lượng của mạch điện cũng như độ tin cậy của hệ thống lôgic điều khiển mà thôi.

Trên thế giới người ta đã xây dựng nên những máy tính chuyên dụng, nó như là một cái "cân", để đánh giá độ tin cậy của mạch điện. Quá trình đánh giá độ tin cậy hoàn toàn tự động và máy luôn đưa ra số liệu đánh giá về độ phức tạp của một bài toán lôgic. Khi có cái "cân" để đánh giá độ tin cậy của máy hay độ phức tạp của hàm số, ta có thể đánh giá được chất lượng hoạt động hay độ tin cậy của một hệ thống lôgic nói chung (ví dụ cụ thể là máy tính) khi lắp ráp chế tạo hay khi chúng được sản xuất hàng loạt và hoàn toàn giống nhau về hình thức, nhưng chất lượng và độ tin cậy lại khác nhau.

Để dễ dàng đánh giá định lượng được độ tin cậy hay độ phức tạp của kết quả đạo hàm, ta đưa ra khái niệm về "giá" của một hàm lôgic là : giá (N) của một hàm lôgic bằng số lượng biến số đang dùng cộng với số lượng phép toán đang có ở hàm số.

Ví dụ : $$f_6(X) = \bar{x}_1 . x_3 . \bar{x}_4 + \bar{x}_2 . x_3$$

$$N = 5 + 7 = 12$$

Kết quả đạo hàm nào có "giá N" lớn thì độ tin cậy sẽ kém.

2.1.3.3. Các tính chất cơ bản của đạo hàm riêng hàm Boole

Đạo hàm riêng của hàm Boole có một số các tính chất cơ bản như sau. Giả sử f(X) và g(X) là các hàm Boole nhiều biến với $X = (x_1, x_2, ..., x_n)$.

1. $\frac{\partial \bar{f}(X)}{\partial X_i} = \frac{\partial f(X)}{\partial X_i}$

2. $\frac{\partial^2 f(X)}{\partial X_i \partial X_i} = 0$

3. $\frac{\partial^2 f(X)}{\partial X_i \partial X_j} = \frac{\partial^2 f(X)}{\partial X_j \partial X_i}$

4. $\frac{\partial (f(X) \cdot g(X))}{\partial X_i} = f(X) \frac{\partial g(X)}{\partial X_i} \oplus g(X) \frac{\partial f(X)}{\partial X_i} \oplus \frac{\partial f(X)}{\partial X_i} \cdot \frac{\partial g(X)}{\partial X_i}$

5. $\frac{\partial (f(X) + g(X))}{\partial X_i} = \overline{f(X)} \frac{\partial g(X)}{\partial X_i} \oplus \overline{g(X)} \frac{\partial f(X)}{\partial X_i} \oplus \frac{\partial f(X)}{\partial X_i} \cdot \frac{\partial g(X)}{\partial X_i}$

6. $\frac{\partial (f(X) \oplus g(X))}{\partial X_i} = \frac{\partial f(X)}{\partial X_i} \oplus \frac{\partial g(X)}{\partial X_i}$

Ta có thể dựa vào các định nghĩa của đạo hàm dễ dàng chứng minh được các tính chất nêu ra.

§2.2. ĐẠO HÀM TOÀN PHẦN VÀ VI PHÂN TOÀN PHẦN HÀM BOOLE

Sau khi đã nắm được khái niệm về đạo hàm hàm Boole ta có thể dễ dàng hiểu được về đạo hàm toàn phần và vi phân toàn phần của hàm Boole.

2.2.1. VI PHÂN TOÀN PHẦN CỦA HÀM BOOLE

Trước khi nói tới vi phân toàn phần ta xét tới vi phân riêng phần hàm Boole.

Định nghĩa 2.2.1 : Vi phân riêng phần $dx_i . f(X)$ của hàm $f(X)$ trong đó $X = (x_1, x_2, ..., x_n)$ đối với biến $x_i \in X$ $(1 \leq i \leq n)$ là số gia của $f(X)$ theo giá trị của x_i là :

$$dx_i\ f(X) = f(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n) \oplus$$

$$f(x_1, \ldots, x_{i-1}, x_i \oplus dx_i, x_{i+1}, \ldots, x_n)$$

Từ định nghĩa của vi phân riêng phần ta có định lý sau

Định lý 2.1 :

$$dx_i\ f(X) = \frac{\partial f(X)}{\partial x_i} dx_i$$

Ta có thể chứng minh định lý này bằng một ví dụ cụ thể.

Ví dụ : cho $f_3(X) = x_1x_2 + x_3$

Tính $dx_1 f_3(X) = f(x_1, x_2, x_3) \oplus f((x_1 \oplus dx_1), x_2, x_3)$

$$= (x_1 x_2 + x_3) \oplus \left((x_1 \oplus dx_1) x_2 + x_3\right) =$$

$$= (x_1 x_2 + x_3) \oplus \left((\bar{x}_1 dx_1 + x_1 \overline{dx}_1) x_2 + x_3 \right) =$$

$$= (x_1 x_2 + x_3) \oplus (\bar{x}_1 x_2 dx_1 + x_1 x_2 \overline{dx}_1 + x_3) =$$

$$= x_2 \bar{x}_3 dx_1$$

$$= \frac{\partial f_3(X)}{\partial x_1} \cdot dx_1$$

Định nghĩa 2.2.2 : Vi phân toàn phần df(X) của hàm f(X) là số gia của hàm số theo số gia dX của biến số (X) được tính :

$$df(X) = f(X) \oplus f(X \oplus dX)$$

Từ định nghĩa vi phân toàn phần ta có định lý sau :

Định lý 2.2 :

$$df(X) = \sum_{e} \frac{\partial f(X)}{\partial X^e} \cdot (dX)^e$$

$$= \sum_{e} dX^e \cdot f(X)$$

ở đây : $0 \leqslant e \leqslant (2^n - 1)$ và $\sum$ là tổng xích ma môđun. Một số tính chất cơ bản của vi phân toàn phần.

1. $da = 0$
2. $d\left(af(X)\right) = adf(X)$
3. $d\bar{f}(X) = df(X)$
4. $d\left(df(X)\right) = 0$
5. $d\left(f(X)\,g(X)\right) = f(X)\,dg(X) \oplus g(X)\,df(X) \oplus df(X)\,dg(X)$
6. $d\left(f(X)\,g(X)\right) = \bar{f}(X)\,dg(X) \oplus \overline{g(X)}\,df(X) \oplus df(X)\,dg(X)$
7. $d\left(f(X) \oplus g(X)\right) = df(X) \oplus dg(X)$

2.2.2. ĐẠO HÀM TOÀN PHẦN

Định nghĩa 2.2.3. Cho $f\left[X, y(X)\right]$ là hàm chuyển mạch véctơ, ở đây X và y là các véctơ n và m thứ nguyên tương ứng. Đạo hàm toàn phần của hàm f đối với biến x_i được định nghĩa như sau :

$$\frac{df}{dx_i} = f\left[x_i, y(x_i)\right] \oplus f\left[\bar{x}_i, y(\bar{x}_i)\right]$$

Đạo hàm toàn phần bậc cao của một hàm Boole được định nghĩa như sau.

Định nghĩa 2.2.4. Cho $X = (X_1, X_2)$, Đạo hàm toàn phần bậc cao của hàm f(X) đối với m biến trong X_1 được định nghĩa như sau :

$$\frac{d\,f(X)}{d\,X_1} = \frac{d}{dx_1}\left(\frac{d}{d\,x_2}\left(\cdots\left(\frac{d\,f(X)}{d\,x_m}\right)\cdots\right)\right)$$

ở đây : $X_1 = (x_1, x_2, ..., x_m)$

Khi đã hiểu được đạo hàm hàm Boole thì cũng dễ dàng hiểu được đạo hàm toàn phần của hàm Boole

§2.3. PHƯƠNG PHÁP ĐỒ THỊ HAY PHƯƠNG PHÁP THUẬT TOÁN ĐỂ TÍNH ĐẠO HÀM HÀM BOOLE.

Qua các phần trên chúng ta đã biết nhiều phương pháp khác nhau dùng để tính đạo hàm Boole, từ phương pháp tính đạo hàm theo định nghĩa, tính đạo hàm hàm Boole theo định lý và tính đạo hàm hàm Boole dựa vào bìa Kácnô nhờ ánh xạ vào bìa K. Các phương pháp dùng tính toán đạo hàm hàm Boole nói trên chủ yếu là làm theo định nghĩa tốn thời gian. Ở đây sẽ đưa ra một phương pháp "mẹo" hay thuật toán có thể thực hiện tính đạo hàm hàm Boole được dễ dàng cho kết quả nhanh hơn và thuận tiện hơn. Cơ sở thực hiện của phương pháp thuật toán là dựa vào bìa Kácnô, khi dùng bìa K thông thường chỉ được áp dụng trong trường hợp số lượng biến ít. Để thực hiện phương pháp thuật toán tính đạo hàm hàm Boole người ta đưa ra các bước thực hiện như sau :

* *Bước 1* : Ánh xạ hàm Boole cần phải đạo hàm vào bìa K.

* *Bước 2* : Xác định trục quay là trục qua đó giá trị lôgic của biến x_i thay đổi giá trị (từ 0 thành 1 và ngược lại).

* *Bước 3* : Quay bìa K cùng với hàm f(X) theo trục x_i ứng với biến số phải đạo hàm.

* *Bước 4* : Tính toán kết quả bằng cách cộng môđun bìa K của hàm f(X) với bìa K của hàm f(X) sau khi đã quay theo trục x_i (trục quay ứng với biến số phải đạo hàm).

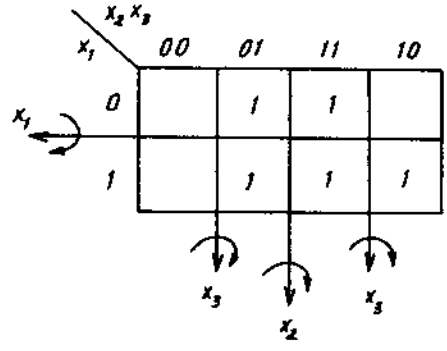
Để hiểu cụ thể các bước thực hiện trong quá trình tích đạo hàm hàm Boole, ta có thể dựa vào hàm số quen biết trong ví dụ ở trên :

Ví dụ : Cho trước $f_3(x_1, x_2, x_3) = x_1 x_2 + x_3$

tính $\frac{\partial f_3(X)}{\partial x_1}$ dùng phương pháp thuật toán.

Bước 1 ánh xạ hàm $f_3(X)$ vào bìa điều này đã được làm ở phần trên, bước 2 xác định trục quay là trục qua đó giá trị của biến số x_i thay đổi giá trị. hai bước đó được mô tả trên hình 3.1 sau.

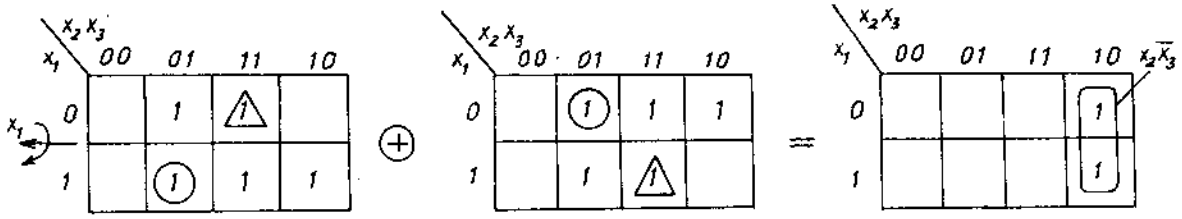
Sau khi ánh xạ $f_3(X)$ vào bìa K, và dựa theo quy ước trục quay là trục qua đó biến số x_i thay đổi giá trị lôgíc ($0 \longleftrightarrow 1$) ta xác định luôn các trục quay cho hàm số $f_3(X)$



Hình 2.7. Xác định trục quay của hàm f (X).

trên bìa K. Từ kết quả đó vấn đề còn lại là sẽ quay hàm $f_3(X)$ theo trục x_i ứng với biến số phải đạo hàm sẽ rất thuận tiện.

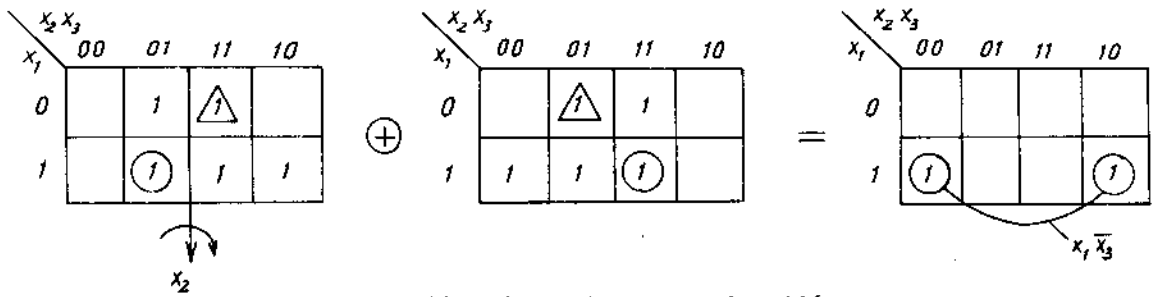
Bước tiếp theo là bước 3 : Quay bìa K cùng với hàm $f_3(X)$ đã được ánh xạ theo trục x_i ứng với biến số phải đạo hàm x_i, cụ thể ở đây ta sẽ quay bìa K của $f_3(X)$ theo trục x_1 là trục ứng với biến số phải đạo hàm là x_1 theo như nhiệm vụ đề ra. Sau đó thực hiện tiếp bước thứ 4 là tính toán kết quả bằng cách cộng mô đan hai bìa K của hàm $f_3(X)$ và của hàm $f_3(X)$ đã được quay theo x_1, cụ thể là "chồng khít" hai bìa đó vào nhau, và tính theo nguyên lý chức năng của phép cộng môđun ($0 \oplus 0 = 0$ và $1 \oplus 1 = 0$) ta sẽ thu được kết quả của phép tính đạo hàm $\frac{\partial f_3(X)}{\partial x_1}$ ở bìa K chứa kết quả. Minh họa các quan hệ đó được biểu diễn ở hình 2.8.



Hình 2.8. Kết quả đạo hàm quay theo biến x_1.

Qua các bước thực hiện ta cũng thu được kết quả cuối cùng $\frac{\partial f_3(X)}{\partial x_1} = x_2 \bar{x}_3$ như khi tính đạo hàm hàm $f_3(X)$ theo biến x_1 bằng các phương pháp khác.

Tương tự ta cũng có thể tính đạo hàm $f_3(X)$ theo biến số x_2 được tích dựa theo các bước thực hiện được mô tả trên hình 2.9.

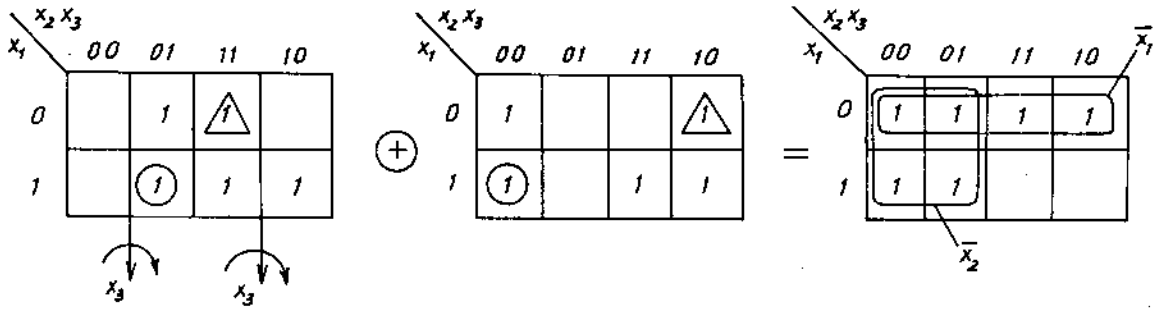


Hình 2.9. Kết quả đạo hàm quay theo biến x_2.

Ta sẽ thu được kết quả $\frac{\partial f_3(X)}{\partial x_2} = x_1 \bar{x}_3$. Kết quả này hoàn toàn có thể thu được theo các phương pháp khác.

Tương tự cách làm đó ta cũng có thể tính toán đạo hàm hàm $f_3(X)$ theo biến x_3,

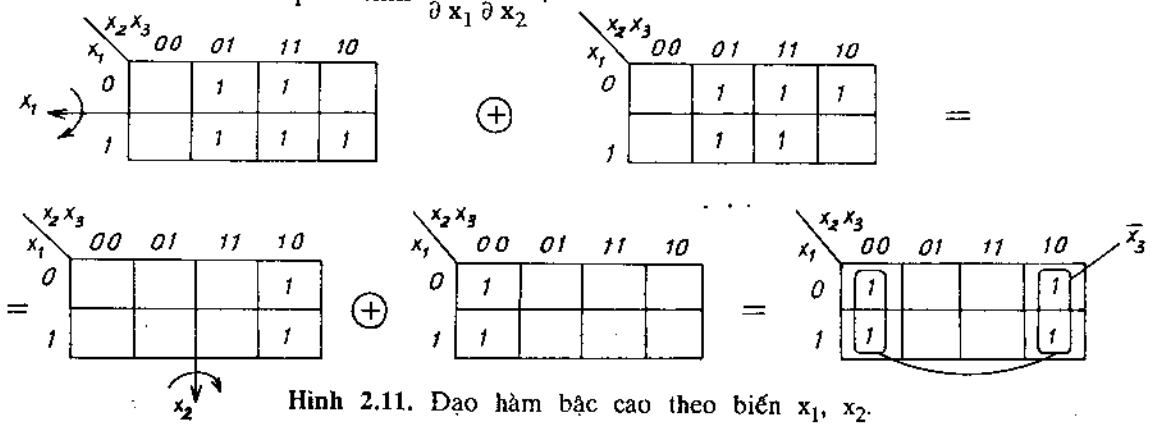
chú ý là khi đạo hàm theo biến x_3 ta phải quay theo hai trục, vì x_3 có hai trục x_3 như đã về ở trên, cách tính được minh họa ở 2.10.



Hình 2.10. kết quả đạo hàm quay theo biến x_3.

Ta thu được kết quả tính toán $\frac{\partial f_3(X)}{\partial x_3} = (\bar{x}_2 + \bar{x}_3)$, kết quả này cũng dễ dàng kiểm tra lại bằng cách tính khác.

Bằng phương pháp thuật toán ta cũng có thể áp dụng để tính toán đạo hàm bậc cao loại I đối với hàm Boole, vì theo định nghĩa của đạo hàm bậc cao loại I thì ta thấy nó là kết quả của đạo hàm của biến x_i sẽ là số liệu đầu vào để tính đạo hàm của biến x_{i+1}, cho nên có thể dựa vào kết quả đạo hàm của một biến này (x_i) để tính tiếp đạo hàm cho biến tiếp theo (x_{i+1}). Điều đó được minh họa tính toán trên hình 2.11, đối với hàm Boole đã cho ta phải tính $\frac{\partial^2 f_3(X)}{\partial x_1 \partial x_2}$.



Hình 2.11. Đạo hàm bậc cao theo biến x_1, x_2.

Từ quan hệ $\frac{\partial^2 f_3(X)}{\partial x_1 \partial x_2} = \frac{\partial}{\partial x_2}\left(\frac{\partial f_3(X)}{\partial x_1}\right)$ như vậy ta tính $\frac{\partial f_3(X)}{\partial x_1}$ trước bằng cách quay bìa K theo trục x_1, kết quả và cách làm đã trình bầy ở phần trên, kết quả ta có $\frac{\partial}{\partial x_2}(x_2 \bar{x}_3)$, từ đây do phải đạo hàm tiếp theo biến x_2 ta lại quay tiếp kết quả của đạo hàm $\frac{\partial f_3(X)}{\partial x_1}$ theo trục x_2 ứng với biến x_2, sau đó tính kết quả thu được $\frac{\partial^2 f_3(X)}{\partial x_1 \partial x_2} = \bar{x}_3$,

kết quả này đã được tính ở phần trên. Như vậy có thể dùng phương pháp thuật toán để tính đạo hàm bậc cao loại I cho hàm Boole, quá trình tính toán theo phương pháp thuật hoàn toàn đơn giản và dễ dàng. Chú ý rằng phương pháp tính toán chỉ có thể áp dụng để tính toán cho đạo hàm riêng và đạo hàm bậc cao loại I của hàm Boole, phương pháp thuật toán không dùng để tính toán đạo hàm bậc cao loại II (có nhiều biến số biến đổi một lúc). Muốn tính toán đạo hàm hàm Boole bậc cao loại II ta chỉ có thể tính toán theo định nghĩa mà thôi.

Sau đây ta thấy một ví dụ đạo hàm cho một hàm Boole bốn biến để thấy rõ tiện ích của phương pháp đạo hàm hàm Boole theo phương pháp thuật toán.

Ví dụ : cho hàm số bốn biến

$$f_4(x_1, x_2, x_3, x_4) = \bar{x}_1 \bar{x}_3 + \bar{x}_1 x_2 \bar{x}_4 + x_2 \bar{x}_3 x_4 + x_1 \bar{x}_2 x_3 \bar{x}_4$$

Tính : $\dfrac{\partial f_4(X)}{\partial x_1}$; $\dfrac{\partial^2 f_3(X)}{\partial x_1 \partial x_2}$.

Trước hết ta thử tính $\dfrac{\partial f_4(X)}{\partial x_1}$ theo định nghĩa

$$\frac{\partial f_4(X)}{\partial x_1} = (\bar{x}_1 \bar{x}_3 + \bar{x}_1 x_2 \bar{x}_4 + x_2 \bar{x}_3 x_4 + x_1 \bar{x}_2 \bar{x}_4) \oplus$$

$$(x_1 \bar{x}_3 + x_1 x_2 \bar{x}_4 + \bar{x}_1 \bar{x}_2 x_3 \bar{x}_4) = A \oplus B$$

Tiếp theo áp dụng công thức $A \oplus B = \bar{A} . B + A . \bar{B}$ ta có :

$$\frac{\partial f_4(X)}{\partial x_1} = \overline{(\bar{x}_1 \bar{x}_3 + \bar{x}_1 x_2 \bar{x}_4 + x_2 \bar{x}_3 x_4 + x_1 \bar{x}_2 x_3 \bar{x}_4)} .$$

$$. (x_1 \bar{x}_3 + x_1 x_2 \bar{x}_4 + x_2 \bar{x}_3 x_4 + \bar{x}_1 x_2 \bar{x}_3 \bar{x}_4)$$

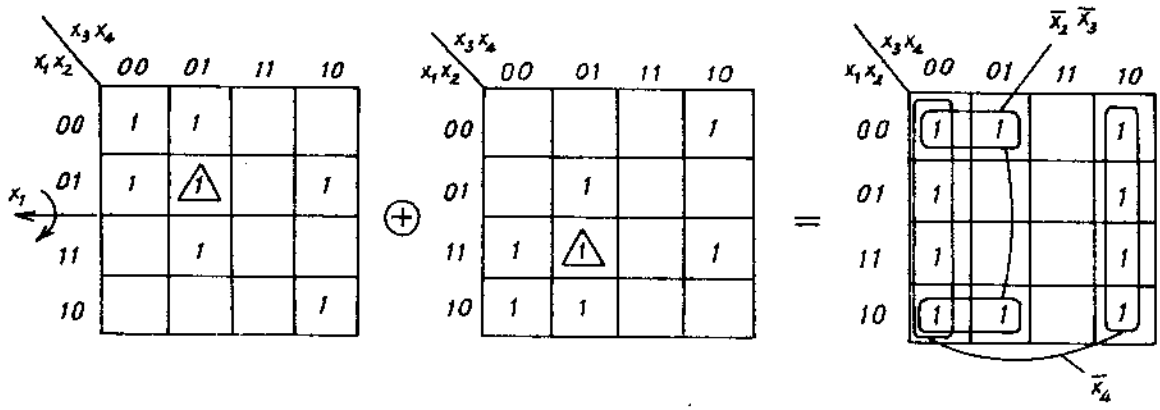
$$+ (\bar{x}_1 \bar{x}_3 + \bar{x}_1 x_2 \bar{x}_4 + x_2 \bar{x}_3 x_4 + x_1 \bar{x}_2 x_3 \bar{x}_4) .$$

$$. \overline{(x_1 \bar{x}_3 + x_1 x_2 \bar{x}_4 + x_2 \bar{x}_3 x_4 + \bar{x}_1 \bar{x}_2 x_3 \bar{x}_4)} = \ldots$$

Sau đó lại phải áp dụng định lý Demorgan tính tiếp, từ đó ta dễ dàng nhận thấy bài toán sẽ lớn lên rất nhanh, và với bài toán đơn giản bốn biến đó chỉ mới tính đạo hàm riêng của nó theo một biến x_1 $\left(\dfrac{\partial f_4(X)}{\partial x_1}\right)$ ta đã khó có thể, thậm chí không thể tính toán được đến kết quả cuối cùng. Đó là chưa kể phải tính $\dfrac{\partial^2 f_4(X)}{\partial x_1 \partial x_4}$ lại càng khó tiến tới đáp số nếu chỉ tính theo định nghĩa mà không dùng phương pháp thuật toán. Nhất là với bài toán có số lượng biến lớn hơn và các tích số có số lượng nhiều hơn, thì việc tính đạo hàm hàm Boole lại càng khó khăn, đó là điều dễ dàng nhận thấy.

Nhưng nếu áp dụng phương pháp thuật toán để tính đạo hàm chúng ta sẽ thu được kết quả tính toán đạo hàm rất nhanh chóng và dễ dàng, điều đó có thể minh họa trên hình 2.12 đối với bài toán bốn biến đã cho như sau :

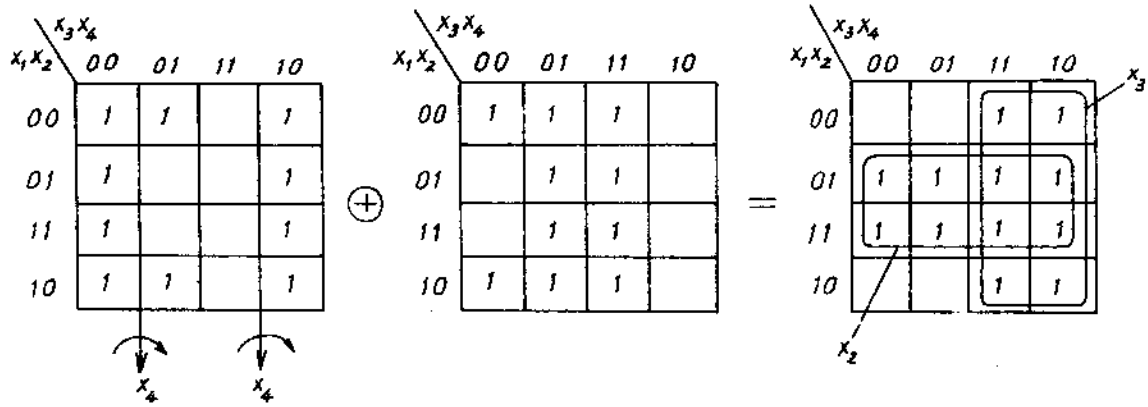
Bước 1 ánh xạ hàm f_4 (X) vào bìa K, sau đó xác định trục quay x_1, rồi quay bìa K của f_4 (X) theo trục quay x_1 ứng với biến số phải đạo hàm, cuối cùng ta tính kết quả.



Hình 2.12. Đạo hàm hàm 4 biến.

Ta nhận được kết quả : $\dfrac{\partial f_4(X)}{\partial x_1} = \bar{x}_4 + \bar{x}_2\bar{x}_3$.

Tính nốt $\dfrac{\partial^2 f_4(X)}{\partial x_1 \partial x_4} = \dfrac{\partial}{\partial x_4}\left(\dfrac{\partial f_4(X)}{\partial x_1}\right)$, như vậy ở đây ta chỉ cần quay tiếp kết quả của đạo hàm $\left(\dfrac{\partial f_4(X)}{\partial x_1}\right)$ theo trục x_4 sau đó tính toán kết quả là xong. Điều đó được minh họa trên hình 2.13.



Hình 2.13. Bước tiếp theo tính đạo hàm bậc cao.

Ta nhận được kết quả : $\dfrac{\partial^2 f_4(X)}{\partial x_1 \partial x_4} = x_2 + x_3$.

Đến đây với bài toán bốn biến đề ra, áp dụng tính toán đạo hàm theo phương pháp thuật toán, đã cho ta kết quả tính toán đạo hàm của hàm số đó một cách dễ dàng và

nhanh chóng, đó chính là sức mạnh cũng như tiện ích của phương pháp thuật toán dùng tính đạo hàm Boole. Với cách làm tương tự ta có thể tính toán đạo hàm với các biến khác.

Phương pháp thuật toán có nhiều điểm thuận tiện để tính đạo hàm hàm Boole, nhưng phương pháp này cũng có những hạn chế ví như chỉ có thể tích toán với số lượng biến số không nhiều, điều này cũng chính là hạn chế của bìa Kácnô. Muốn tính toán được đạo hàm hàm Boole có số lượng biến lớn hơn thì ta phải sử dụng đến máy tính, đó là phương pháp đạo hàm hàm Boole một cách tự động bằng cách lập trình cho máy tính, khi lập trình cho máy tính cũng dựa tiêu các định nghĩa hay định lý tính toán đạo hàm hàm Boole làm thuật toán viết chương trình :

Ý nghĩa và ứng dụng của đạo hàm và vi phân hàm Boole

- Dùng đạo hàm hàm Boole có thể phát hiện sai, đồng thời xây dựng được các "test" thử dùng để đánh giá chất lượng cũng như độ tin cậy trong khả năng điều khiển của hệ thống điện tử.

- Có thể phát hiện được sai số động, phát hiện sai trong quá trình điều khiển cũng như phát hiện sai trong thời gian thực.

- Làm cơ sở để xây dựng các ôtômat tự động sửa sai.

- Tổng hợp và phân tích đánh giá độ tin cậy của hệ thống, cũng như dự báo chuẩn đoán sai nhầm.

- Áp dụng trong lý thuyết nhận dạng, cũng như trong lý thuyết phân giải các hàm Boole.

Mục đích chính của chúng ta khi nghiên cứu về hệ tổ hợp là xây dựng nên các mạch điện điều khiển, như các bộ nhớ cố định, các bộ điều khiển chương trình địa phương, các chương trình lôgic tiêu chuẩn (môđun) như PAL (Programmable Array Logic), PLA (Programmable Logic Array) hay PIA (Programmable Interconnect Array)... Sau khi đã có được mạch điện, nhiệm vụ tiếp theo là phải đánh giá được chất lượng hoạt động của mạch lôgic điều khiển đó, độ tin cậy của nó khi nó hoạt động và có thể phát hiện sai cùng với sửa sai nếu có cho mạch đó, nhiệm vụ này có thể thực hiện dựa vào đạo hàm hàm Boole. Nhưng như vậy vẫn chưa đủ vì người kỹ sư ngoài việc thiết kế ra mạch điện, đánh giá chất lượng của mạch điện đó, còn cần phải biết cách sửa chữa mạch điện khi nó hỏng hóc. Đối với mạch lôgic tổ hợp ngoài những hiện tượng hỏng hóc thông thường như mọi mạch điện khác (như hỏng IC, hỏng linh kiện...), nó còn có một loại "hỏng" đặc biệt, loại "hỏng" này chỉ có riêng ở hệ lôgic tổ hợp, việc "hỏng hóc" đặc biệt này nó có tên gọi là Hazard. Đối với mạch lôgic tổ hợp thường được dùng làm các hệ thống điều khiển, các chương trình điều hành, như vậy nếu hệ thống điều khiển bị "hỏng" bị Hazard thì đây là một sự hỏng hóc rất nguy hiểm, cần phải được khắc phục và sửa chữa ngay. Hazard là hiện tượng "hỏng" của mạch điện trong khi các linh kiện và mạch điện của hệ thống vẫn tốt nguyên. Khái niệm Hazard được trình bày ở phần sau.

§2.4. MÃ DÙNG TRONG KỸ THUẬT (MÃ MÁY)

Mã hay còn được gọi là "mã máy", nó được đưa ra do nhu cầu "hội thoại giữa người và máy", với mục đích làm sao cho máy tính "hiểu" được người và người cũng "hiểu" được máy tính, làm cho giữa máy tính và con người có thể trao đổi thông tin trực tiếp được với nhau. Đây là điều rất quan trọng mà các loại máy công cụ khác không thể có được. Từ khi có mã, có quy định về mã máy tức là có quy định về "ngữ nghĩa" giữa người và máy tính, thì giữa người và máy tính bắt đầu có "ngôn ngữ" để trao đổi với nhau và để "hiểu nhau". Cho tới nay việc "hội thoại" giữa người và máy được thấy qua bàn phím hay con chuột rất dễ dàng và đơn giản, ai cũng có thể làm được, nhờ một khái niệm đơn giản nhưng rất quan trọng đó là mã.

Quy ước về mã của con người luôn luôn kèm theo sau là ngữ nghĩa, con người có "mã ngữ" là ngôn ngữ được quy ước hay mã hóa giữa người với người trao đổi được thông tin với nhau và hiểu nhau. Vậy giữa người và máy tính có quy ước "ngữ nghĩa" như thế nào để hiểu và trao đổi được thông tin với nhau.

2.4.1. MÃ TÍN HIỆU TRONG MÁY

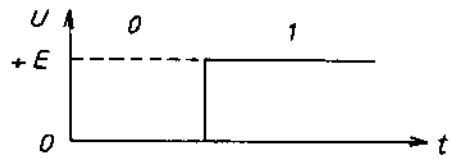
Đối với con người khi nói về lôgic về các giá trị 0 hay 1 thì rất dễ hiểu, nhưng với máy tính nó "không hiểu" được về lôgic về giá trị là 0 hay 1 là gì cả, mà máy chỉ có thể nhận biết và xử lý thông tin thông qua tín hiệu điện. Do đó con người dựa vào tín hiệu điện để có quy ước mã lôgic theo tín hiệu điện.

2.4.1.1 Quy ước mã với dạng tín hiệu kiểu điện thế

Hiện nay trong kỹ thuật, người ta thường quy ước :

- Điện áp 0 von tương ứng với lôgic 0
- Điện áp +E (+5 Von) ứng với lôgic 1

Dạng tín hiệu có thể được minh họa trên hình 1.1.



Hình 2.14. Mã tín hiệu điện thế.

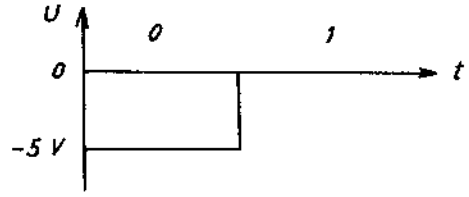
Ta cũng có thể có quy ước mã tín hiệu điện thế kiểu khác như :

- Điện áp 0 von ứng với lôgic 1
- Điện áp -E (-5 von) ứng với lôgic 0

Dạng tín hiệu được mô tả trên hình 1.2.

Tóm lại ta có thể sử dụng điện thế cao ứng với giá trị lôgic 1, còn điện áp thấp ứng với giá trị lôgic 0. Nhưng chú ý là quy ước mức điện áp ứng với giá trị lôgic khác

nhau thì với cùng một mạch điện chức năng sẽ khác nhau, có nghĩa là ta quy ước mã điện áp như thế nào thì phải có mạch điện lôgic tương ứng với mã đó. Quy ước mã kiểu điện thế ứng với việc bật tắt công tắc.



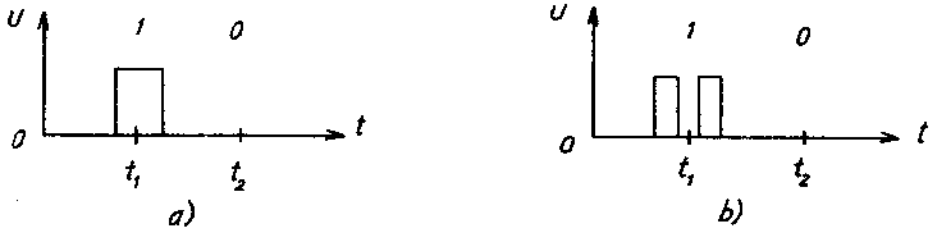
Hình 2.15. Mã tín hiệu điện thế âm.

2.4.1.2. Quy ước mã với dạng tín hiệu kiểu xung

Thông thường người ta quy ước mã tín hiệu là :

- Có xung tương ứng với lôgic 1
- Không có xung tương ứng với lôgic 0

Dạng tín hiệu theo quy ước mã được minh họa trên hình 2.16.



Hình 2.16. Mã tín hiệu kiểu xung điện.

Ở đây xung có thể là một xung hay một loạt xung, thông thường là một xung riêng biệt. Khi dùng xung để làm quy ước mã lôgic 0 hay 1 thì bắt buộc phải có mốc thời gian hay là có xung nhịp làm thời gian chuẩn, vì có xung ứng với giá trị lôgic 1 thì dễ phân biệt, nhưng giá trị lôgic 0 thì phải trùng với mốc thời gian hay xung nhịp khi mới được tính là lôgic 0.

Loại tín hiệu lôgic dùng xung thường tương ứng với tín hiệu điều khiển từ bên ngoài, như các công tắc hay phím bấm trên bàn phím điều khiển. Sự sử dụng xung kích thường liên quan tới hệ thống số đồng bộ theo thời gian.

Đối với tín hiệu điện tử thì có rất nhiều dạng tín hiệu khác nhau, nhưng các tín hiệu điện khi dùng trong hệ thống số (trong ôtômat) thông thường chỉ là các xung vuông, hay các dạng tín hiệu quy ước mẫu như đã nêu trên. Chúng là những tín hiệu điện đơn giản và có thể tạo ra được rất dễ dàng, đó chính là thế mạnh và sự ứng dụng sử dụng dễ dàng của các thiết bị của hệ thống số, ví dụ như tín hiệu lôgic (0,1) kiểu điện áp chỉ cần bật hay tắt công tắc như là bật đèn điện, tín hiệu lôgic (0,1) kiểu xung thì chỉ cần tạo ra bằng phím bấm hay dùng công tắc bấm nhả như khi ta bấm tay lên bàn phím của máy vi tính, hay "nháy chuột" hoặc bấm tay lên bàn phím của điện thoại di động v.v...

Chính vì cách tạo ra tín hiệu "điện lôgic" (0,1) đối với hệ thống số quá đơn giản, nêu việc trao đổi tín hiệu giữa người và "máy số" được thực hiện dễ dàng, hay thực hiện việc hội thoại qua lại giữa người và máy không còn khó khăn như các loại hệ thống tín hiệu điện tử khác, điều đó cho phép sự ứng dụng của hệ thống số hay các ôtômat ngày càng phong phú, đồng thời làm cho "quan hệ" giữa người và máy gần gũi hơn. Nhưng cũng chính vì các tạo ra tín hiệu "điện lôgic" rất đơn giản cho nên con người đã có thể dùng các hiện tượng vật lý phi điện để điều khiển tự động hệ thống số thông qua các loại bộ cảm biến (xensơ) khác nhau, điều này lại càng làm tăng thêm khả năng ứng dụng của hệ thống số, ví dụ như dùng ánh sáng thông qua các bộ cảm biến bằng phôtô điôd hay phôtotranzitor để đóng mở cửa tự động, dùng micrô để điều khiển bằng âm thanh, dùng biến trở nhiệt độ để đo lường chỉ thị số, dùng bộ cảm biến áp suất để tìm được ảnh hưởng của tải trọng lên cầu v.v...

Trong quyển sách này sẽ đề cập việc thiết kế cụ thể ra các hệ thống số các ôtômat chỉ dùng các giá trị lôgic 0 và 1. Tuy nhiên cần phải hiểu rộng hơn là : các giá trị lôgic đó có thể là tín hiệu điện, nhưng cũng có thể lấy từ các bộ cảm biến (như xúc giác con người) theo các hiện tượng vật lý xảy ra xung quanh chúng ta, làm tác nhân để điều khiển các hệ thống số được thiết kế ra đó. Sau khi có quy ước về mã tín hiệu điện, bắt đầu từ nay khi ta nói tới giá trị lôgic 0 hay 1 thì đó là một tín hiệu điện cụ thể và ngược lại một tín hiệu cụ thể sẽ ứng với một giá trị lôgic. Tiếp theo ta sẽ đưa ra các quy ước mã cao hơn về quy ước "ngữ nghĩa" để trao đổi thông tin giữa máy và người, đó là các quy ước về con số dùng để trao đổi về tính toán.

2.4.2. MÃ NHỊ PHÂN (BCD - BINARY CODE DECIBEN - 8.4.2.1)

Mã nhị phân là mã máy đơn giản nhất nhưng cũng là mã hay được sử dụng nhất để "hiểu nhau" trong quan hệ giữa người và máy tính. Mã nhị phân hay mã BCD tuân theo quy luật của con số nhị phân, theo quan hệ 2^n trong đó n là số tự nhiên tăng dần ; n cũng là số lượng tín hiệu vào hay số lượng bít thông tin của máy, với các giá trị của n tạo ra trọng số của một số nhị phân $(2^3, 2^2, 2^1, 2^0) = (8 - 4 - 2 - 1)$. Mã nhị phân quy ước (tương ứng) với các ký hiệu chữ số ở cơ số 10, mà cơ số 10 là cơ số rất quen thuộc của con người, cho nên dùng mã nhị phân giữa người và máy sẽ dễ dàng hiểu nhau, quy ước mã nhị phân được minh họa trên bảng ở hình 2.17.

Nhị phân (BCD)				Cơ 10
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	-
1	0	1	1	-
1	1	0	0	-
1	1	0	1	-
1	1	1	0	-
1	1	1	1	-

Hình 2.17. Mã nhị phân.

Tất nhiên để có thể mã hóa cho 10 ký hiệu chữ số của hệ cơ số 10, phải cần có tối thiểu 4 bit thông tin thì mới không bị trùng mã, tức là có $2^4 = 16$ khả năng mã hóa (trước đây ở hàm Boole 2^4 là khả năng

của tín hiệu vào), như vậy còn thừa 6 khả năng tổ hợp. Với mã nhị phân người và máy dễ dàng trao đổi thông tin với nhau nhưng đó là mã nhị phân tăng dần theo số thứ tự của cơ số 10. Còn nếu chỉ để mã hóa, thì chỉ cần "quy ước" giữa một tín hiệu nhị phân bất kỳ tương ứng với một ký hiệu chữ số của cơ số mười miễn chúng không trùng quy ước là được, từ suy nghĩ đó nẩy ra các loại quy ước mã khác nhau. Một trong những loại mã thông dụng hay được dùng trong kỹ thuật sẽ được nêu tiếp theo.

2.4.3. MÃ GRÂY

Mã Grây (mã theo bìa Kácnô) là mã được viết theo quy luật quen thuộc của bìa Kácnô, mã này còn có tên gọi là mã "chống nhiễu", thực chất là mã chống chạy đua tín hiệu vào hay là mã "chống hazard" thì ý nghĩa sát hơn cả, vì bản chất của của Grây là : bộ tín hiệu nhị phân chỉ có một vị trí lôgic là được thay đổi tương ứng với số thứ tự tăng dần của các ký hiệu (ký tự) chữ số của cơ đếm 10. Quy luật quy ước mã hay quy ước "ngữ nghĩa" của mã Grây sang cơ số 10 được biểu diễn ở bảng được chỉ ra trên hình 2.18.

Mã Grây				Cơ 10
0	0	0	0	0
0	0	0	1	1
0	0	1	1	2
0	0	1	0	3
0	1	1	0	4
0	1	1	1	5
0	1	0	1	6
0	1	0	0	7

Mã Grây				Cơ 10
1	1	0	0	8
1	1	0	1	9
1	1	1	1	-
1	1	1	0	-
1	0	1	0	-
1	0	1	1	-
1	0	0	1	-
1	0	0	0	

Hình 2.18. Mã Grây.

Bảng mã Grây là bảng quy ước "ngữ nghĩa" giữa người và máy có trọng số, vì vị trí của các bit lôgic 0 hay 1 tương ứng với giá trị của trọng số nhị phân khác nhau tương ứng với vị trí của bit thông tin đó. Mã chẳng qua là quy ước "ngữ nghĩa" cho nên ta có thể biến đổi hay có các quy ước mã khác nữa như sau.

2.4.4. MÃ NHỊ PHÂN THỪA BA

Đây cũng là loại mã quen thuộc, mã này phức tạp hơn các bộ mã trước, làm cho việc tìm hiểu hoạt động (chức năng) kỹ thuật của hệ thống khó tìm hiểu hơn, nhưng do có quy luật là thừa 3 nên việc giải mã cũng dễ dàng hay dễ dàng hiểu máy. Bộ mã thừa 3 được quy ước theo bảng ở hình 2.19.

Bắt đầu từ số ba nhị phân (0011) thì ứng với số đầu tiên là chữ số 0 của mã thừa 3. Với cách làm tương tự ta có mã "thừa 3 - Grây".

Mã nhị phân (BCD)				Mã thừa 3
0	0	0	0	-
0	0	0	1	-
0	0	1	0	-
0	0	1	1	0
0	1	0	0	1
0	1	0	1	2
0	1	1	0	3
0	1	1	1	4

Mã nhị phân (BCD)				Mã thừa 3
1	0	0	0	5
1	0	0	1	6
1	0	1	0	7
1	0	1	1	8
1	1	0	0	9
1	1	0	1	-
1	1	1	0	-
1	1	1	1	-

Hình 2.19. Mã thừa 3.

2.4.5. MÃ THỪA 3 - GRÂY

Mã này có các tính chất như mã Grây nhưng do thừa 3 nên bắt đầu từ số ba Grây (0010) sẽ ứng với chữ số 0 đầu tiên của bộ mã thừa 3 Grây. Quy ước ngữ nghĩa của mã thừa 3 - Grây được chỉ ra ở bảng trên hình 2.20.

Mã Grây				Thừa Grây 3
0	0	0	0	-
0	0	0	1	-
0	0	1	1	-
0	0	1	0	0
0	1	1	0	1
0	1	1	1	2
0	1	0	1	3
0	1	0	0	4

Mã Grây				Thừa Grây 3
1	1	0	0	5
1	1	0	1	6
1	1	1	1	7
1	1	1	0	8
1	0	1	0	9
1	0	1	1	-
1	0	0	1	-
1	0	0	0	-

Hình 2.20. Mã Grây thừa 3.

Các bộ mã đã được quy ước ở trên đều có đặc điểm chung là có trọng số cố định, mỗi một bít thông tin ứng với một cột số đều có giá trị của trọng số tương ứng, và giá trị của trọng số được giảm dần từ trái sang phải. Tiếp theo ta sẽ đưa ra khái niệm về mã có trọng số thay đổi.

2.4.6. MÃ CÓ TRỌNG SỐ THAY ĐỔI

Loại mã có trọng số không thay đổi ví dụ như mã nhị phân (BCD) có quy luật của trọng số không thay đổi, giảm dần từ trái qua phải là (8 - 4 - 2 - 1). Đối với loại mã có trọng số "thay đổi" có nghĩa là chúng ta sẽ đưa ra quy luật trọng số khác đi, không theo quy luật của mã nhị phân bình thường đó nữa.

Ví dụ : ta có bộ trọng số quy ước khác đi như sau (2 - 4 - 2 - 1), khi có quy ước

trọng số mới thì mã số đối với máy vẫn có hình thức là số nhị phân thông thường, nhưng giá trị của con số nhị phân đó sẽ khác đi, ví dụ như ta có bộ mã trọng số thay đổi là :

$$(1\,0\,1\,1) = 1.2 + 0.4 + 1.2 + 1.1 = 2 + 0 + 2 + 1 = 5$$

Bảng quy ước mã có trọng số thay đổi minh họa trên hình 2.21.

Mã nhị phân (BCD)				Mã 2-4-2-1
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	-
0	1	1	0	-
0	1	1	1	-

Mã nhị phân (BCD)				Mã 2-4-2-1
1	0	0	0	-
1	0	0	1	-
1	0	1	0	-
1	0	1	1	5
1	1	0	0	6
1	1	0	1	7
1	1	1	0	8
1	1	1	1	9

Hình 2.21. Mã có trọng số thay đổi.

Đặc điểm của mã có trọng số thay đổi là có thể có cùng một giá trị (lượng) nhưng có thể xuất hiện hai bộ mã khác nhau như.

$$(1\,0\,1\,1) = 5 = (0\,1\,0\,1) = 5$$

Khi gặp phải trường hợp cùng một giá trị có hai bộ mã khác nhau thì ta bắt buộc phải loại trừ đi một bộ mã để tránh nhầm lẫn xảy ra ở trong máy, điều này mạch phần cứng phải thực hiện để loại bỏ đi một bộ mã. Ở đây thường người ta loại đi bộ mã quen thuộc của số nhị phân thông thương, ví dụ cụ thể ở đây người ta loại bỏ bộ mã (0101). Với bộ mã hình thức là con số nhị phân nhưng trọng số bị thay đổi sẽ khó hiểu và khó phân tích chức năng của mạch hơn, từ những khái niệm rất sơ giản này đó là mầm mống để tạo ra các bộ mật mã, các bộ mã bí hiểm dùng trong lĩnh vực thông tin khác. Các mã được nêu ở trên đều liên quan tới con số mà chủ yếu là cơ số 10, tức là liên quan tới việc tính toán ở trong máy tính, khi máy tính tính toán đại số theo con số nhị phân. Sau đó đưa ra kết quả tính toán cũng ở dạng số nhị phân, ta chỉ cần giải mã sang con số có đếm 10 là hiểu ngay kết quả tính toán. Ngoài ra liên quan tới con số người ta còn đưa ra các quy ước mã số kiểu khác nữa như sau.

2.4.7. MÃ KHÔNG CÓ TRỌNG SỐ, MÃ (2–8) HAY MÃ (2–10)

Loại mã không có trọng số quy ước mã của nó có thể minh họa ở bảng của hình 2.22.

Một trong những yếu điểm lớn của số nhị phân là con số được viết rất dài, nhiều khi không thể để nguyên như vậy mà đọc được, đồng thời rất khó nhớ đối với con

người. Từ đó người ta đưa ra quy ước con số ở dạng mã (2-8) tức là quy ước từng ba bít thông tin một thành một số đi với nhau, hay quy ước mã (2-10) là từng bộ bốn thông tin làm thành một nhóm với nhau, và mỗi một nhóm bit thông tin đó người ta nhận được một giá trị con số tương ứng.

Mã không trọng số					Số thập phân
0	0	0	0	0	0
1	0	0	0	0	1
1	1	0	0	0	2
1	1	1	0	0	3
1	1	1	1	0	4
1	1	1	1	1	5
0	1	1	1	1	6
0	0	1	1	1	7
0	0	0	1	1	8
0	0	0	0	1	9

Hình 2.22. Mã không có trọng số.

Ví dụ quy ước mã số (2 - 3) là gộp 3 bít lại :

$(0\,1\,0\,1\,1\,0\,1\,0\,0\,1\,0\,1) = (2\ 6\ 4\ 5)_{(2\text{-}8)}$

$\underbrace{0\,1\,0}\ \underbrace{1\,1\,0}\ \underbrace{1\,0\,0}\ \underbrace{1\,0\,1} = (2\,6\,4\,5)_{(2\text{-}8)}$

2 6 4 5

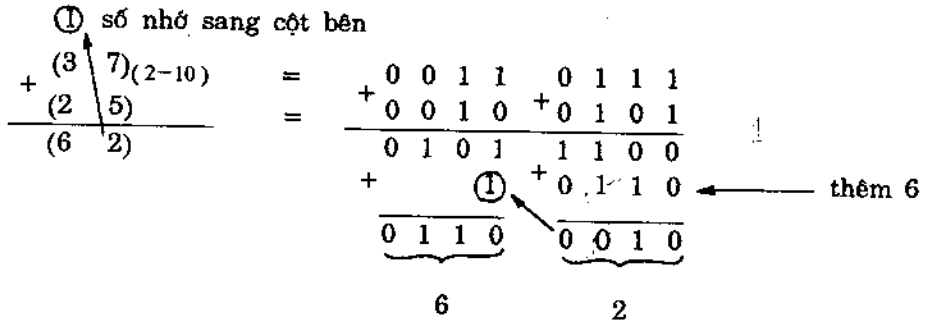
Ví dụ quy ước mã số (2 - 10) là gộp 4 bít lại.

$(0\,1\,0\,1\,0\,0\,1\,1\,1\,0\,0\,1\,0\,1\,1\,1) = (5\,3\,9\,7)_{(2\text{-}10)}$

$\underbrace{0\,1\,0\,1}\ \underbrace{0\,0\,1\,1}\ \underbrace{1\,0\,0\,1}\ \underbrace{0\,1\,1\,1} = (5\,3\,9\,7)_{(2\text{-}10)}$

5 3 9 7

Ta thấy số (2 - 8) chỉ có thể biểu diễn chữ số lớn nhất là $(111) = (7)_{(2\text{-}8)}$. Còn khi ta biểu diễn con số theo mã (2 - 10) thì có thể biểu diễn được đầy đủ 10 ký hiệu (ký tự) chữ số của cơ đếm 10, thậm chí vẫn còn thừa 6 bộ mã không dùng đến. Khi biểu diễn con số ở mã (2 - 10) ta vẫn có thể tính toán số học dễ dàng như ví dụ sau :



Cho thêm 6 để tạo ra số nhớ ① sang cột bên.

Mã nhị phân (2-10) liên quan tới các bộ tạo mã như bàn phím telêfôn hay bàn phím máy công cụ, phần sau sẽ nói cách tạo ra loại mã này.

Các mã chúng ta đã nêu ở trên đều liên quan tới con số cơ đếm 10, tức là chỉ đưa ra được quy ước số nhị phân (mã máy) tương ứng với chữ số của cơ đếm 10, điều này chỉ phục vụ riêng cho quá trình tính toán học trong máy tính hay trong hệ thống số nói chung. Nhưng con người không chỉ có tính toán đơn thuần mà còn muốn thông qua máy tính viết thư, viết báo... trao đổi thông tin với máy quá các chữ viết. Từ đó con người đưa ra bộ mã cho các ký tự chữ viết, bộ mã này hiện nay đã phổ cập thông dụng

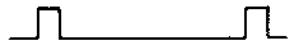
trên khắp thế giới, đó chính là bộ mã của bàn phím máy vi tính. Khi ta có quy ước mã cho chữ viết thì quan hệ giữa người và máy tính được nâng lên ở mức cao hơn đó chính là "ngữ nghĩa", và từ đó quan hệ giữa người và máy được phong phú hơn, và máy tính hay các hệ thống số sẽ phục vụ được đắc lực hơn cho con người. Ta đi tiếp tới bộ quy ước mã đó !

2.4.8. MÃ HÓA KÝ TỰ CHỮ VIẾT HAY MÃ ASCII

Mã ASCII (American National Code for Information Interchange) có thể được gọi là bộ quốc tế mã kỹ thuật, vì bộ mã này đã được các nhà kỹ thuật số trên trái đất chấp nhận và ứng dụng rất rộng rãi, đến mức trở thành bộ mã quen thuộc không thể thiếu được trong máy tính cũng như trong các hệ thống số khác... Đó là bộ mã quy ước giữa mã nhị phân là của máy với chữ viết của con người. Đây là một quy ước "mã nghĩa" giữa máy và người vô cùng quan trọng, khi xuất hiện bộ mã ASCII làm cho hệ thống điện tử số (máy tính) vượt xa các hệ thống điện tử khác như TV hay radio chính là ở chỗ giữa người và máy được trao đổi về ngữ nghĩa với nhau, thể hiện sự "thông minh" hơn của máy tính điện tử so với các hệ thống điện tử khác.

Mã ASCII là bộ mã nhị phân, quy ước giữa mã máy và các ký tự chữ cái, được viết theo quy luật của mã (2 – 10) có ví dụ minh họa cho vài ký tự chữ cái.

$$A = (4\ \ 1)_{(2-10)} = 0\ 1\ 0\ 0\ \ \ 0\ 0\ 0\ 1$$



dạng xung điện của chữ A

Khi bấm phím có chữ A qua bộ tạo mã sẽ có được bộ của tương ứng (0100 0001) đồng thời cũng tạo ra một tín hiệu tương ứng chạy trong máy, nhờ có tín hiệu điện đó mà ký tự chữ A được hiển thị lên màn hình của máy. Như vậy với mỗi một ký tự của bàn phím chúng ta có được bộ mã nhị phân tương ứng, đồng thời có được tín hiệu điện của ký tự đó ở trong máy, và cho phép ký tự đó được hiển thị lên màn hình. Bộ mã nhị phân dùng cho bàn phím chính là mã ASCII, nó quy ước cho các chữ từ chữ A đến chữ Z, thậm chí cả những ký tự cần thiết khác giống như các ký hiệu bàn phím của máy chữ cũng được mã hóa.

Bộ của ASCII cho bàn phím của máy tính được biểu diễn trên bảng chỉ ra ở trên hình 2.23.

Cách tìm mã ứng với một ký tự

$$\text{Ký tự} = (b_8\ b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1)$$

Ví dụ : A = 1 0 0 0 0 0 0 1

B = 1 0 0 0 0 0 1 0

3 = 0 1 1 0 0 1 1

$b_4b_3b_2b_1$ \ $b_7b_6b_5$	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	\	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	,	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	.	<	L	\	l	?
1101	CR	GS	−	=	M	]	m	}
1110	SO	RS	.	>	N	↑	n	~
1111	SI	US	/	?	O	↓	o	DEL

Hình 2.23. Bảng mã ASCII.

SO = 0 0 0 1 1 1 0

a = 1 1 0 0 0 0 1

Mã ASCII là của 8 bit nhị phân, nhưng để mã hóa được toàn bộ ký tự trên bàn phím của máy tính ta chỉ cần dùng 7 bít ($b_1 \div b_7$) là đủ, còn bít thứ 8 là bít parity (kiểm tra chẵn lẻ) dùng để phát hiện lỗi trên ký tự. Ngoài các ký tự chữ số và chữ viết của bàn phím, trong bảng mã ASCII còn có các ký hiệu "chữ ghép" dùng để thực hiện các chức năng. Ý nghĩa của các "chữ ghép" của mã ASCII được chỉ ra trên hình 2.24.

Bảng ý nghĩa các ký hiệu chữ của mã ASCII

Ký hiệu	Ý nghĩa	Ký hiệu	Ý nghĩa
NUL	Số không, không, vô hiệu	BS	Lùi một khoảng ký tự
SOH	Bắt đầu của tiêu đề	HT	Kẻ bảng hướng ngang
STX	Bắt đầu của hành văn	LF	Chuyển dòng
ETX	Kết thúc của hành văn	VT	Kẻ bảng hướng dọc
EOT	Kết thúc truyền tin	FF	Điều khiển chạy giấy
ENQ	Hỏi	CR	Quay về đầu dòng
ACK	Thừa nhận	SO	Dịch ra (shift out)
BEL	Chuông	SI	Dịch vào (Shiff in)
DLE	Chuyển mã (datakinK escape)	EM	Hết giấy
DC1	Điều khiển thiết bị 1	SUB	Trừ
DC2	Điều khiển thiết bị 2	ESC	Chuyển mã
DC3	Điều khiển thiết bị 3	FS	Dấu phân cách (file separator)
DC4	Điều khiển thiết bị 4	GS	Dấu phân cách gói
NAK	Phủ định	RS	Dấu phân cách ghi
SYN	Đồng bộ	US	Dấu phân cách đơn vị
ETB	Kết thúc truyền gói tin	SP	Khoảng trống ký tự
CAN	Hủy bỏ	DEL	Hủy bỏ lùi

Hình 2.24. Bảng ý nghĩa ký hiệu "chữ ghép"

Cũng chính nhờ có bộ chữ cái (ASCII), làm cho việc điều khiển máy tính sẽ tiện lợi hơn thông qua các ngôn ngữ lập trình như FOTRAN, PASCAL... thay vì việc lập trình điều khiển máy tính bằng mã máy hay ngôn ngữ máy ASSEMBL. Nhưng dù là viết theo ngôn ngữ gì, thì máy vẫn chỉ hoạt động trực tiếp theo ngôn ngữ máy (theo mã nhị phân) hoạt động theo phần cứng của máy mà thôi, nói cách khác phần cứng sẽ quyết định chính cho phần mềm cách viết chương trình như thế nào. Như vậy nếu ta trực tiếp lắp ráp ra phần cứng của máy tính thì có thể trực tiếp hay tự đưa ra được ngôn ngữ điều hành cho cái máy tính của mình. hay nói một cách khác mỗi nhà kỹ thuật về phần cứng của máy tính điện tử có thể tự đưa ra một bộ quy ước "mã chữ cái" cho riêng mình, và tự có ngôn ngữ chương trình cho riêng cái máy tính mà mình tạo ra, điều này thuộc bản chất. Nhưng máy tính "mới" đó khó trao đổi thông tin vì với người dùng nó lại phải học lại ngôn ngữ chương trình mới gây phiền phức, cho nên mã ASCII là bộ mã đã được phổ cập và quen thuộc trên trái đất, vì thế các nhà kỹ thuật máy tính dù thiết kế ra máy mới với của mác hoạt động tốt hơn, nhưng vẫn phải có mã "tương thích" với các máy đã được dùng quen trước đây, hay bắt buộc phải dùng bộ mã ASCII khi xây dựng thế hệ máy mới đó.

Đến đây chúng ta khảo sát song một phần quan trọng của kiến thức về hệ thống số, đó là toán lôgic, đại số Boole và hàm Boole, về hệ cơ số đếm nhị phân, về một cách biến đổi toán học kiểu khác dùng bìa Kácnô, hay nói rộng hơn là đã đề cập về hệ lôgíc tổ hợp (Combinatinal Ciruits) chuyên dùng để xây dựng nên các hệ thống điều khiển của hệ thống số. Đây là một hệ thống tín hiệu ra phụ thuộc trực tiếp vào tín hiệu vào, nó không lưu trữ thông tin trước đó, hay còn được gọi là hệ lôgic tổ hợp không có nhớ. Nhưng hệ lôgic tổ hợp còn có một loại mạch hoạt động nhờ lưu giữ thông tin trước đó, gọi là "hệ lôgic có nhớ" sẽ được trình bầy ở phần tiếp theo.

BÀI TẬP PHẦN I

Phần lý thuyết tập hợp

1) Các biểu thức sau viết đúng hay sai?

- $\Phi \subseteq \Phi$ — $\Phi \subseteq \{ \Phi \}$
- $\Phi \in \Phi$ — $\Phi \in \{ \Phi \}$
- $a \subseteq (a, b, c)$ $a \in (a, b, c)$
- $(a, b) \subseteq (a, b, c, (a, b, c))$
- $(a, b) \in (a, b, c, (a, b, c))$

2) Có phải mỗi mệnh đề sau là đúng hay sai với các tập A, B, C bất kỳ?

- Nếu $A \in B$ mà $B \subseteq C$ thì $A \in C$
- Nếu $A \in B$ mà $B \subseteq C$ thì $A \subseteq C$
- Nếu $A \subseteq B$ mà $B \in C$ thì $A \in C$
- Nếu $A \subseteq B$ mà $B \in C$ thì $A \subseteq C$

3) Chứng minh rằng với 3 mệnh đề sau đây là tương đương đối với các tập A, B bất kỳ.

$$A \subseteq B \qquad AUB = B \qquad A \cap B = A$$

4) Nếu có quan hệ

$$(A \cap C) \subseteq (B \cap C) \text{ và } (A \cap C) \subseteq (B \cap \overline{C})$$

thì thỏa mãn $A \subseteq B$

5) Trong lớp có 40 học sinh, trong đó có 28 người thích chơi bóng đá, 15 người thích chơi bóng chuyền mà trong số đó có 8 người vừa thích chơi bóng chuyền vừa thích chơi bóng đá. Hỏi số học sinh chỉ thích chơi bóng đá và số người vừa không thích chơi bóng đá vừa không thích chơi bóng chuyền.

Phần hệ cơ số đếm

1) Các cách viết các con số ở cơ số đếm bất kỳ như sau đúng hay sai? Vì sao?

$(5\ 3\ 7\ 4)_8$	$(2\ 6\ 4\ 5)$	$(3\ 6\ 8\ 4\ 2)_7$
$(2\ 5\ A\ 6)_{15}$	$(4\ 7\ D\ 9)_{10}$	$(1\ 0\ 2\ 1)_3$
$(6\ E\ 5\ A)_B$	$(2\ 0\ A\ C)_{16}$	$(2\ 4\ 3\ 0)_5$

2) Tính toán cộng các con số sau

$$\begin{array}{r} (6\ 3\ 0\ 5\ 2)_7 \\ +\ (3\ 5\ 4\ 6\ 5)_7 \\ \hline (\qquad\qquad)_7 \end{array} \qquad \begin{array}{r} (5\ 2\ 4\ 7\ 3)_8 \\ +\ (6\ 4\ 5\ 6\ 5)_8 \\ \hline (\qquad\qquad)_8 \end{array} \qquad \begin{array}{r} (2\ 5\ A\ 6\ 6)_{16} \\ +\ (6\ B\ 2\ 5\ D)_{16} \\ \hline (\qquad\qquad)_{16} \end{array}$$

3) Làm tính trừ các con số sau :

$(5\,2\,0\,3)_6 - (2\,5\,3\,4)_4 =$

$(7\,2\,4\,6)_9 - (3\,8\,6\,7)_9 =$

$(D\,2\,5\,6)_{16} - (9\,3\,A\,8)_{16} =$

4) Tính toán các phép nhân các số sau

$(2\,4\,3\,5)_6 \times (2\,4\,3)_6 =$

$(3\,2\,4\,5)_7 \times (3\,4\,5)_7 =$

$(2\,6\,4\,7)_8 \times (3\,1\,2)_8 =$

5) Làm các phép toán chia sau

$(2\,5\,3\,1)_8 : (7)_8 =$

$(4\,3\,2\,5)_7 : (5)_7 =$

$(2\,7\,3\,6)_9 : (1\,2)_9 =$

6) Các phép toán sau tính đúng hay sai? tại sao?

$$\begin{array}{r} 2\ 5\ 3\ 7\ 6 \\ +\ 4\ 7\ 6\ 8\ 4 \\ \hline (7\ 4\ 2\ 7\ 1)_9 \end{array} \qquad \begin{array}{r} 6\ 4\ 2\ 7\ 1 \\ -\ 3\ 5\ 4\ 3\ 5 \\ \hline (2\ 6\ 7\ 3\ 5)_9 \end{array} \qquad \begin{array}{r} 6\ 5\ 3\ 2\ 4 \\ +\ 2\ 4\ 5\ 6\ 3 \\ \hline (1\ 2\ 3\ 4\ 2\ 0)_7 \end{array} \qquad \begin{array}{r} 5\ 4\ 6\ 3\ 1 \\ +\ 2\ 5\ 4\ 3\ 5 \\ \hline (2\ 7\ 2\ 5\ 6)_8 \end{array}$$

7) Chuyển đổi giữa các cơ số đếm

$(2\ 5\ 4)_{10} = (\quad)_{10}$; $(5\ 3\ 1)_6 = (\quad)_5$

$(3\ 7\ 5)_8 = (\quad)_7$; $(6\ 3\ 4)_8 = (\quad)_6$

8) Chuyển đổi giữa cơ số 10 sang cơ số 2

$(1\ 5)_{10} = (\quad)_2$; $(2\ 3)_{10} = (\quad)_2$

$(4\ 6)_{10} = (\quad)_2$; $(7\ 5)_{10} = (\quad)_2$

9) Chuyển đổi cơ số 2 thành cơ số 10

$(1\ 0\ 1\ 1)_2 = (\quad)_{10}$; $(1\ 1\ 0\ 1\ 0)_2 = (\quad)_{10}$

$(1\ 1\ 0\ 1\ 1\ 0)_2 = (\quad)_{10}$; $(1\ 0\ 1\ 1\ 0\ 1)_2 = (\quad)_{10}$

10) Tính toán kết quả sau :

$(4\ 3\ 2)_7 \times (3\ 4)_7 + (4\ 5\ 6\ 0)_7 = (\quad)_7$

$\big((6\ 3\ 2)_8 + (4\ 5\ 3)_8\big) + (6)_8 = (\quad)_8$

$$\Big((7\ 3\ 5)_9 - (6\ 4\ 3)_9\Big) \times (3\ 2)_9 = (\quad)_9$$

11) Trong khu vườn có $(26)_8$ con bò, số lượng gà nhiều gấp $(5)_8$ lần số lượng bò. Hỏi trong khu vườn có bao nhiêu cái chân của bò và của gà.

12) Một cửa hàng có $(3\,2)_9$ Kg đường, sau khi bán hết $(6\,4\,3)_9$ gam đường. Số đường còn lại chia làm $(6)_9$ túi. Hỏi mỗi túi có bao nhiêu Kg đường và chia xong còn dư bao nhiêu gam đường. Với $(1000)_9$ g đường = 1 Kg đường ở cơ số 9.

13) Khôi phục lại phép toán sau? nó được tính ở cơ số nào?

$$\begin{array}{r} 6\square\ 7\ 6\ 5\ 3\ \square \\ +\ \ 7\ 4\ \square\ \square\ 6\ 7\ 8 \\ \hline (1\ 4\ 7\ 8\ \square\ 3\ 2\ 6)? \end{array} \qquad \begin{array}{r} 4\ 6\ 5\ \square\ 0\ 2 \\ -\ \square\ \square\ 6\ 5\ 3\ 4 \\ \hline (3\ 0\ 5\ \square\ 3\ \square)? \end{array}$$

Phần dàn và hệ thống đại số

1) Cho a, b, c là các phần tử của dàn D. Chứng minh rằng nếu $a \leqslant b$ thì

$$a + (b\,.\,c) \leqslant b\,.\,(a + c).$$

2) Chứng minh rằng dàn D là một dàn phân bố nếu và chỉ nếu đối với các phần tử a, b, c bất kỳ thuộc dàn ta có :

$$(a + b)\,.\,c \leqslant a + (b\,.\,c)$$

3) Chứng minh rằng dàn D là một dàn môđun nếu và chỉ nếu :

$$a + \Big(b\,.\,(a + c)\Big) = (a + b)\,.\,(a + c)$$

4) Cho x, y, z là là các phần xử của dàn D. Chứng minh rằng :

$$x\,.\,y + y\,.\,z + z\,.\,x \leqslant (x + y)\,.\,(y + z)\,.\,(z + x)$$

5) Chứng minh rằng dàn D là dàn phân bố nếu và chỉ nếu đối với mỗi một tập hợp các phần tử x, y, z thuộc dàn D, có quan hệ sau :

$$x\,.\,z \leqslant y \leqslant x + z$$

Kéo theo :

$$x\,.\,y + y\,.\,z = y = (x + y)\,.\,(y + z)$$

Phần đại số Boole

1) Một căn nhà có ba cửa ra vào. Lập bảng chân lý bảng thật cho chức năng nếu cứ có hai cửa ra vào bị mở đồng thời thì báo hiệu.

2) Lập bảng chức năng hay bảng chân lý cho hàm số sau :

$$f(x_1, x_2, x_3, x_4) = x_1\ x_2\ \bar{x}_3 + \bar{x}_2\ x_3\ x_4 + x_3\ \bar{x}_4$$

3) Xác định hàm $\bar{f}$ (A, B, C, D) nếu biết :

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

4) Trong đại số Boole chứng minh rằng :

Nếu a $\subseteq$ b thì thỏa mãn :

$$a(\bar{b} + c) + a\bar{c} = a(b + \bar{b}c)(\bar{a} + b)$$

5) Trong đại số Boole chứng minh rằng

Nếu c $\leqslant$ b thì có quan hệ

$$(a + c)(\bar{a} + b) = ab + c$$

6) Chứng minh rằng hai hàm Boole sau đây tương đương

$$f_1(A, B, C) = (A + C)(B + \bar{C})$$

$$f_2(A, B, C) = (A + B)(A + \bar{B} + C)(\bar{A} + B + \bar{C})$$

7) Hai hàm Boole sau đây có tương đương không? Vì sao?

$$f_1(A, B, C, D) = BC + A\bar{B}D + A\bar{B}\bar{C}\bar{D}$$

$$f_2(A, B, C, D) = A\bar{B}\bar{C} + ADC + BD$$

8) Cho R là một quan hệ phản xạ trên tập A. Chứng minh rằng R là một quan hệ tương đương nếu và chỉ nếu :

(a, b) $\in$ R và (a, c) $\in$ R thì có (b, c) $\in$ R

9) Dùng các hàm lôgíc cơ bản AND, OR, NOT viết lại hàm cộng môđun sau.

$$F_3 = (A \oplus B) \oplus C$$

$$F_4 = (A \oplus B) \oplus (C \oplus D)$$

$$f_3 = \overline{(A \oplus C)} \oplus B$$

$$f_4 = (A \oplus B) \oplus \overline{(C \oplus D)}$$

10) Hãy chứng minh các quan hệ sau :

$$A + \bar{A}.B = A + B$$

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

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

11) Rút gọn các biểu thức sau theo tiên đề

$$Y = \bar{A} + AB\bar{C} + (\bar{A} + AB\bar{C})(A + \bar{A}\bar{B}C)$$

$$Y = (AB + AB\bar{C} + CD)(A.\bar{C} + B.\bar{D})$$

$$Y = A\bar{C}D + \bar{A}CB + \bar{A}\bar{B}C + CD$$

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

12) Với các giá trị nào làm cho $\bar{A}.B + C.\bar{D} = 0$ thì thỏa mãn quan hệ :

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

13) Lập bảng chân lý cho các hàm Boole sau :

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

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

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

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

14) Chứng minh các đẳng thức sau :

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

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

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

$$\overline{A}.\overline{B} + \overline{A}B + A\overline{B} + A.B = 1$$

15) Chứng minh các đẳng thức sau :

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

$$A\overline{B} + BD + DCE + D\overline{A} = A\overline{B} + D$$

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

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

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

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

$$ABC + A\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C = A \oplus B \oplus C$$

16) Chứng minh các đẳng thức sau :

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

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

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

$$A.(A \oplus B) = A.\overline{B}$$

$$A.B(A \oplus B \oplus C) = ABC$$

17) Hãy lập bảng chân lý cho các hàm dưới đây. Sau đó giải thích quan hệ của F_1 và F_2.

$$\begin{cases} F_1 = A\overline{B} + B\overline{C} + C\overline{A} \\ F_2 = \overline{A}B + \overline{B}C + \overline{C}A \end{cases}$$

$$\begin{cases} F_1 = ABC + \overline{A}\,\overline{B}\,\overline{C} \\ F_2 = \overline{(A\overline{B} + B\overline{C} + C\overline{A})} \end{cases}$$

$$\begin{cases} F_1 = \overline{A \oplus B \oplus C} \\ F_2 = ABC + A\overline{B}\,\overline{C} + \overline{A}B\overline{C} + \overline{A}\,\overline{B}C \end{cases}$$

$$\begin{cases} F_1 = A\overline{C}D + \overline{A}\,\overline{B} + BC \\ F_2 = A\overline{B}C + A\overline{B}\,\overline{D} + B\overline{C}\,\overline{D} \end{cases}$$

18) Dùng bìa Kácnô giải thích quan hệ trong bài tập 17.

19) Hãy viết biểu thức toán học ANO - OR cho bảng chân lý của các hàm lôgic $F_1 \div F_5$ dưới đây hình 1.1

A	B	C	F_1	F2	F3	F_4	F5
0	0	0	0	0	0	0	1
0	0	1	1	0	1	1	0
0	1	0	1	0	1	1	0
0	1	1	0	1	0	1	1
1	0	0	1	0	1	0	0
1	0	1	0	1	0	0	0
1	1	0	0	1	0	0	1
1	1	1	1	1	0	1	0

Hình 1.1. Cho hàm số.

20) Hãy viết các hàm lôgic sau thành biểu thức có số hạng nhỏ nhất.

$$Z = AB + BC + CA$$

$$Z = S + \overline{R}y$$

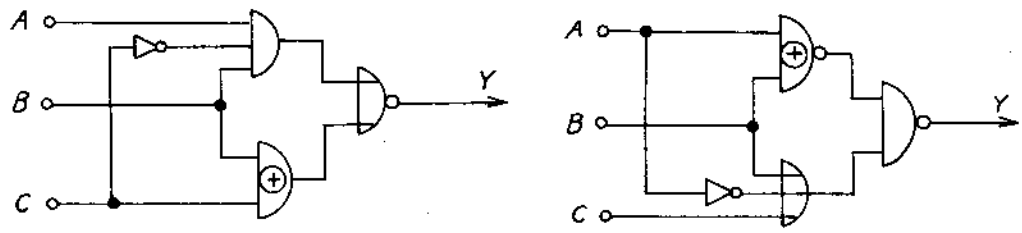
$$Z = j\bar{y} + \bar{k}y$$

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

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

21) Bìa Kácnô là gì? Quy luật sắp xếp biến số trên bìa Kácnô như thế nào và với mục đích gì?

22) Hãy viết biểu thức toán học, bảng chân lý và bìa Kácnô cho mạch điện sau ở hình 1.2.



Hình 1.2. Cho mạch

23) Dùng các quan hệ toán học gì để rút gọn hàm Boole.

24) dùng cách biến đổi toán học hay dựa vào tiên đề để rút gọn hàm Boole sau.

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

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

$AB + A\overline{B} + \overline{A}B + \overline{A}\,\overline{B}$

$(A + AB + ABC)(A + B + C)$

$(A\overline{B} + \overline{A}B)(AB + \overline{A}\,\overline{B})$

$(A\overline{B} + D)(A + \overline{B})D$

$A\overline{B} + C + \overline{A}\,\overline{C}D + B\overline{C}D$

25) Dựa vào tiên đề rút gọn hàm lôgic sau dưới dạng hàm lôgic cơ bản AND - OR và OR - AND.

$\overline{AB + \overline{A}\,\overline{B}}$

$\overline{(A \oplus B)}.\overline{(B \oplus C)}$

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

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

$\overline{(A\overline{C} + BD + \overline{A}BC)}$

26) Tìm hàm phủ định (NOT) của các hàm lôgic sau

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

$(\overline{A}\,\overline{B} + \overline{B}\,\overline{D}).(AC + BD)$

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

$AB + B\overline{D} + \overline{B}C + \overline{C}D$

$D\left[\overline{C} + (AB + D).E\right]$

$A \oplus B \oplus C$

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

Rút gọn hàm lôgic dùng bìa Kácnô

1) Hãy viết dạng OR - AND ở dạng rút gọn cho hàm lôgic trong các bìa Kácnô sau trên hình 1.3.

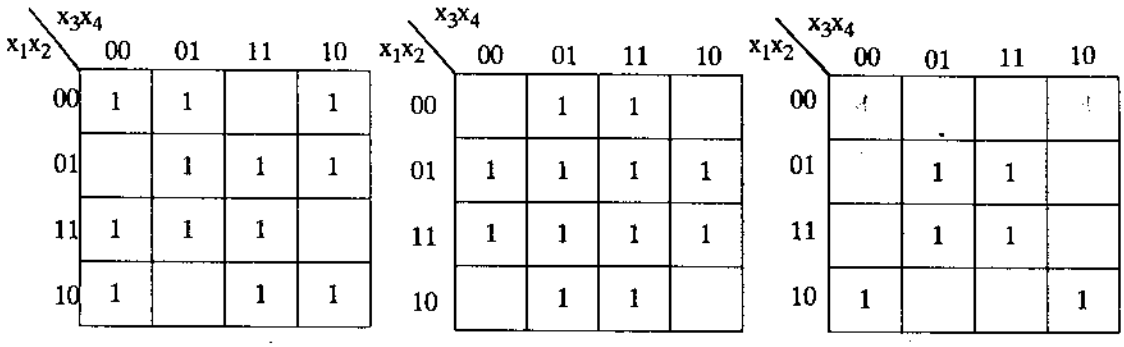
A \ BC	00	01	11	10
0	1			1
1	1		1	1

A \ BC	00	01	11	10
0	1	1	1	1
1		1	1	

A \ BC	00	01	11	10
0	1	1		
1		1		

Hình 1.3. Có hàm.

2) Rút gọn hàm Boole sau ở hình 1.4.



Hình 1.4. Cho hàm Boole

3) Dùng bìa Kácnô rút gọn hàm Boole sau ở dạng OR-AND

$F(A,B,C) = (0,1,2,5)$

$F(A,B,C) = (0,2,4,6,7)$

$F(A,B,C) = (0,1,2,3,4,5)$

$F(A,B,C,D) = (0,1,8,9,10)$

4) Dùng khái niệm đỉnh đầu mút và khoanh to nhất (loại bỏ mạch thừa) tối tiểu hóa hàm Boole sau trên bìa Kácnô.

$F(x_1,x_2,x_3) = (1,2,4,5)$

$F(x_1,x_2,x_3) = (2,3,4,5,7)$

$F(x_1,x_2,x_3,x_4) = (0,1,9,11,13,14,15)$

$F(x_1,x_2,x_3,x_4) = (2,3,8,10,11,13,15)$

$F(x_1,x_2,x_3,x_4) = (0,1,2,3,4,9,10,12,13,14,15)$

$F(x_1,x_2,x_3,x_4) = (1,3,,8,9,10,11,14,15)$

$F(x_1,x_2,x_3,x_4) = (3,5,8,9,11,13,14,15)$

$F(x_1,x_2,x_3,x_4) = (0,1,2,3,4,9,10,11,12,13,14,15)$

5) Tối tiểu hóa hàm Boole sau theo phương pháp Quine–Mc.Cluskey :

$F(x_1,x_2,x_3) = (0,1,2,3,7)$

$F(x_1,x_2,x_3) = (2,3,4,6,7)$

$F(x_1,x_2,x_3,x_4) = (2,3,5,8,10,11,13,15)$

$F(x_1,x_2,x_3,x_4) = (0,2,3,4,5,7,8,9,10,11,12,13,15)$

6) Dùng phương pháp bìa Kácnô rút gọn hàm lôgic có vị trí không xác định sau :

$F(x_1,x_2,x_3) = (0,1,2,7)$

$$X = (3,6) \leftarrow \text{không xác định}$$

$$F(x_1, x_2, x_3, x_4) = (0,1,8,9,13,15)$$

$$X = (2,5,14)$$

$$F(x_1, x_2, x_3, x_4) = (2,3,8,10,11,13,15)$$

$$X = (0,4,7,9)$$

$$F(x_1, x_2, x_3, x_4) = (0,2,3,4,10,11,12,14,15)$$

$$X = (1,9,13)$$

7) Tìm đỉnh đầu mút và tối tiểu hóa các hàm Boole sau trên bìa Kácnô

$$F(x_1, x_2, x_3, x_4, x_5) = (0,1,2,4,6,8,12,14,20,22,25,27,29,31)$$

$$F_5(X) = (0,2,3,7,9,11,12,13,15,16,18,19,23,26)$$

$$F_5(X) = (1,3,5,7,9,11,13,15,16,18,20,22,25,26,29,31)$$

8) Dùng bìa Kácnô tối tiểu hóa hàm Boole năm biến có vị trí không xác định sau.

$$F_5(X) = (1,3,5,8,10,14,17,20,21,27,30,31)$$

$$X = (7,12,18,24,26)$$

$$F_5(X) = (0,1,3,4,5,9,12,13,17,21,22,24,28,30)$$

$$X = (4,7,8,17,26)$$

$$F_5(X) = (0,1,2,4,6,8,12,14,19,20,22,27,29,31)$$

$$X = (7,11,25,28)$$

Phần đạo hàm và vi phân hàm Boole

1) Chứng minh rằng điều kiện cần và đủ để một hàm Boole $f(x_1, x_2, ..., x_n)$ không phụ thuộc vào biến x_i $(1 \leq i \leq n)$ là $\dfrac{\partial f(X)}{\partial x_i} = 0$ $\qquad X = (x_1, x_2, .., x_n)$

2) Dựa vào định nghĩa tính đạo hàm riêng của hàm Boole sau

Cho : $f(x_1, x_2, x_3) = \bar{x}_1 \bar{x}_2 + x_2 x_3 + x_1 x_3 \qquad X = (x_1, x_2, x_3)$

Tính : $\dfrac{\partial f(X)}{\partial x_1}$; $\dfrac{\partial f(X)}{\partial x_2}$; $\dfrac{\partial f(X)}{\partial x_3}$

3) Dùng định lý 2.1 tính đạo hàm riêng của hàm Boole sau.

Cho :

x_1 \ x_2x_3	00	01	11	10
0		1		1
1	1	1	1	

Tính : $\frac{\partial f(X)}{\partial x_1}$; $\frac{\partial f(X)}{\partial x_2}$; $\frac{\partial f(X)}{\partial x_3}$

Cho hàm số : $f(x_1, x_2, x_3) = \bar{x}_3 + \bar{x}_1 x_2$

Tính : $\frac{\partial f(X)}{\partial x_1}$; $\frac{\partial f(X)}{\partial x_2}$; $\frac{\partial f(X)}{\partial x_3}$

4) Dùng phương pháp bìa Kácnô (dạng đồ thị) để tính đạo hàm riêng của hàm Boole sau.

Cho : $f(x_1, x_2, x_3) = \bar{x}_2 + x_1 \bar{x}_3$

Tính : $\frac{\partial f(X)}{\partial x_1}$; $\frac{\partial f(X)}{\partial x_2}$; $\frac{\partial f(X)}{\partial x_3}$

Cho hàm : $f(x_1, x_2, x_3, x_4) = \bar{x}_1 \bar{x}_2 + x_1 x_2 x_3 + x_1 \bar{x}_2 x_4 + \bar{x}_2 \bar{x}_3 x_4$

Tính : $\frac{\partial f(X)}{\partial x_1}$; $\frac{\partial f(X)}{\partial x_2}$; $\frac{\partial f(X)}{\partial x_3}$

5) Tính đạo hàm bậc cao loại I cho hàm Boole sau dựa vào định nghĩa.

Cho hàm : $f(x_1, x_2, x_3) = x_1 \bar{x}_2 + x_1 x_3 + x_2 x_3$

Tính : $\frac{\partial^2 f(X)}{\partial x_1 \partial x_2}$; $\frac{\partial^2 f(X)}{\partial x_2 \partial x_3}$; $\frac{\partial^2 f(X)}{\partial x_1 \partial x_3}$

6) Dựa vào định nghĩa tính đạo hàm bậc cao loại II cho hàm Boole sau.

Cho hàm : $f(x_1, x_2, x_3) = x_1 x_3 + x_2 x_3 + x_1 x_2$

Tính : $\frac{\partial f(X)}{\partial (x_1, x_2)}$; $\frac{\partial f(X)}{\partial (x_2, x_3)}$; $\frac{\partial f(X)}{\partial (x_1, x_3)}$

7) Tính đạo hàm bậc cao loại I dựa theo định nghĩa cho hàm số sau. Rồi dùng phương pháp đồ thị kiểm tra lại kết quả.

Cho hàm : $f_4(X) = x_1 \bar{x}_3 + x_1 x_2 x_4 + x_1 \bar{x}_2 \bar{x}_3 x_4$

Tính : $\frac{\partial^2 f_4(X)}{\partial x_1 \partial x_2}$; $\frac{\partial^2 f_4(X)}{\partial x_2 \partial x_3}$; $\frac{\partial^2 f_4(X)}{\partial x_1 \partial x_3}$

8) Tính đạo hàm bậc cao loại II dựa theo định nghĩa cho hàm Boole sau. Rồi dùng phương pháp đồ thị kiểm tra lại kết quả

Cho hàm : $f_4(X) = \bar{x}_1 x_4 + x_1 x_2 \bar{x}_3 + x_1 \bar{x}_2 x_3 \bar{x}_4$

Tính : $\frac{\partial f_4(X)}{\partial (x_1, x_2)}$; $\frac{\partial f_4(X)}{\partial (x_2, x_3)}$; $\frac{\partial f_4(X)}{\partial (x_1, x_3)}$

9) Dùng phương pháp thuật toán để tính đạo hàm riêng của hàm Boole sau.

Cho hàm : $f_4(X) = \bar{x}_1\bar{x}_3 + x_1\bar{x}_2x_3 + \bar{x}_2\bar{x}_3x_4 + x_1x_3x_4$

Tính : $\frac{\partial f_4(X)}{\partial x_1}$; $\frac{\partial f_4(X)}{\partial x_2}$; $\frac{\partial f_4(X)}{\partial x_3}$; $\frac{\partial f_4(X)}{\partial x_4}$

10) Dùng phương pháp thuật toán để tính toán đạo hàm riêng hàm Boole không xác định sau.

Cho : $f_4(X) = (3, 4, 6, 7, 9, 13)$

$X = (2, 12, 14)$ – Không xác định

Tính : $\frac{\partial f_4(X)}{\partial x_1}$; $\frac{\partial f_4(X)}{\partial x_2}$; $\frac{\partial f_4(X)}{\partial x_3}$; $\frac{\partial f_4(X)}{\partial x_4}$

11) Dùng phương pháp thuật toán để tính đạo hàm bậc cao loại I cho hàm Boole sau.

Cho : $f_4(X) = x_1\bar{x}_3 + x_1x_2\bar{x}_4 + \bar{x}_1x_3x_4 + \bar{x}_1\bar{x}_2x_3 + \bar{x}_1\bar{x}_2\bar{x}_3x_4$

Tính : $\frac{\partial^2 f_4(X)}{\partial x_1 \partial x_4}$; $\frac{\partial^2 f_4(X)}{\partial x_2 \partial x_3}$; $\frac{\partial^2 f_4(X)}{\partial x_1 \partial x_4}$; $\frac{\partial^3 f_4(X)}{\partial x_2 \partial x_3 \partial x_4}$

PHẦN II
HỆ THỐNG SỐ VÀ MẠCH SỐ

GIỚI THIỆU CHUNG

Phần II đề cập đến các mạch điện cụ thể thực hiện các chức năng khác nhau của hệ thống số. Các mạch điện này được xây dựng và lắp ráp dựa trên cơ sở của toán lôgic – toán rời rạc và các phương trình toán học.

Trong phần này sẽ trình bầy chi tiết về hai bài toán kỹ thuật, đó là bài toán phân tích ôtômat dùng cho việc sửa chữa và bài toán thiết kế hay thực hiện mạch điện cho một ôtômat cụ thể.

Đối với bài toán phân tích, đây là một bài toán ngược, tức là từ sơ đồ mạch điện đã có ta phải tìm lại bảng chức năng hay tìm lại hoạt động chính xác hoạt động của mạch điện. Dựa trên cơ sở hiểu biết chính xác hoạt động của mạch điện thì ta mới phát hiện được chỗ sai để sửa chữa. Bởi vì hệ thống số là một hệ thống điện tử, nhưng cách sửa chữa nó không giống cách sửa chữa các mạch điện tử khác thông thường, không giống như chữa đài, ampli hay TV... Do đài hay TV khi bị hỏng thì vị trí hỏng của chúng hoàn toàn không thay đổi, nên ta chỉ cần phát hiện ra chỗ hỏng là sửa chữa thay thế được. Còn đối với hệ thống số, nó là một hệ thống điện tử tự động vận động, nên khi bị hỏng một chỗ hay sai một bit, thì vị trí "hỏng" đó sẽ không nằm im một chỗ mà vận động theo hoạt động chung của hệ thống, nó cứ theo quá trình tự vận động đó cho tới khi xảy ra hoạt động vô lý, lúc đó hệ thống sẽ bị kẹt cứng lại không làm việc được nữa. Khi đó người ta thường gọi là hệ thống bị hỏng hay treo máy. Do đặc điểm "hỏng hóc" của hệ thống số như vậy nếu muốn sửa chữa nó, ta chỉ còn cách dựa vào hiểu biết chính xác chức năng hoạt động của nó (bảng chức năng) thì mới phát hiện ra chỗ hoạt động sai để sửa chữa. Đây là một vấn đề khó khi tiếp xúc tới các thiết bị điện tử số, cũng như máy công cụ có liên quan tới kỹ thuật số.

Với bài toán thiết kế tổng hợp hệ thống số thì nhiệm vụ quan trọng nhất là làm sao chuyển ý muốn, một nhiệm vụ, một chức năng thành các phương trình toán, rồi dựa vào phương trình toán để lắp ráp ra mạch điện cụ thể. Đây cũng là một đặc trưng, một khái niệm mới mẻ khi ta muốn chế ra một công cụ mới "thông minh". Khi thiết kế một

hệ thống số cần phải dựa vào khái niệm về dàn, về phân hoạch, mã hóa trạng thái cũng như khái niệm về phân giải ôtômat... nhằm đảm bảo cho mạch điện lắp ráp gọn gàng, mạch lạc, dễ thay thế sửa chữa, tiện lợi cho việc quản lý bằng phần cứng và cả bằng phần mềm sau này. Đồng thời phải đưa ra được các biện pháp chống hazard, chống chạy đua tín hiệu vào X và chạy đua trạng thái Y, chống hoạt động sai nhầm trong hệ thống. Để đảm bảo tăng độ hoạt động tin cậy cho hệ thống, các vấn đề phức tạp đó sẽ được trình bầy cụ thể kèm các ví dụ minh họa nhằm cho ta có được một cách nhìn tổng quát để chế ra mạch điện của hệ thống số hoạt động đúng chức năng, nhưng đạt được mạch điện lắp ráp gọn gàng đẹp đẽ, đồng thời độ tin cậy cao khi làm việc.

Ngoài ra trong phần này còn giới thiệu về mạch điện cũng như các ôtômat đặc biệt hay gặp trong các hệ thống số. Sự có mặt của chúng cùng với phối hợp hài hòa của chúng, làm cho hệ thống số hoạt động phong phú hơn, đa dạng hơn, đồng thời phục vụ cho con người tiện lợi và đắc lực hơn.

Hệ thống số được chia ra làm hai loại có tính chất riêng đó là ôtômat không có nhớ còn được gọi là "hệ tổ hợp" (Combinational Circuits) và ôtômat có nhớ còn được gọi là "hệ dãy" (Sequential Circuits).

Chương III

ÔTÔMAT KHÔNG CÓ NHỚ

§3.1. KHÁI NIỆM CHUNG VÀ MÔ HÌNH TOÁN HỌC

3.1.1. KHÁI NIỆM CHUNG

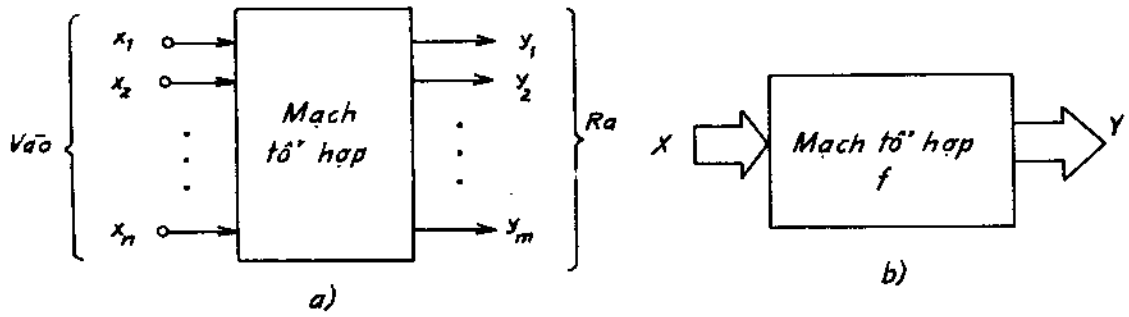
Ôtômát không có nhớ hay hệ tổ hợp đã được đề cập trong phần toán lôgic, vì giữa toán lôgic và mạch tổ hợp có quan hệ trực tiếp với nhau trong quá trình thực hiện ôtômat.

Hệ lôgic tổ hợp là một hệ thống điện tử số mà tin tức (tín hiệu) ở đầu ra của nó chỉ phụ thuộc trực tiếp vào tin tức ở đầu vào của hệ thống tại thời điểm đang xét. Nói cách khác đáp ứng ra của mạch số chỉ phụ thuộc trực tiếp tín hiệu tác động vào mà thôi, không phụ thuộc vào lịch sử của thông tin trước đó, vì hệ thống mạch điện loại này không lưu giữ trạng thái của thông tin trước đó hay không ghi nhớ thông tin trước đó *– hệ thống không có nhớ.*

Chức năng hoạt động của ôtômat không có nhớ được mô tả bằng bảng chức năng, bảng chân lý hay bảng thật, và mạch điện của nó chỉ bao gồm các phần tử, các mạch lôgic cơ bản như (OR, AND, NOT, NAND, NOR) mà không chứa bất kỳ một phần tử ghi giữ thông tin nào. Mạch điện của hệ thống hoạt động theo nguyên tắc có tín hiệu vào thì có đáp ứng tín hiệu điều khiển ở đầu ra, nếu mất tín hiệu vào thì không có đáp ứng ra. Đó là quan hệ ràng buộc một–một (không giống ôtômat có nhớ), hoạt động đó của hệ tổ hợp cũng giống như hoạt động của các mạch điện dân dụng như đài, hay TV... Khi mất tín hiệu vào anten thì cũng mất đáp ứng ra, màn hình sẽ mất hình ảnh. Quan hệ phụ thuộc giữa tín hiệu vào và đáp ứng ra như vậy thì gọi là hệ thống điện tử không có nhớ.

3.1.2. MÔ HÌNH TOÁN HỌC CỦA HỆ TỔ HỢP

Mô hình toán học hay mạch điện tổng quát của hệ tổ hợp được mô tả trên hình 3.1.



Hình 3.1. Mô hình toán học tổng quát hệ tổ hợp.

Hệ thống có n đầu vào và m đầu ra, trong đó :

$X = (x_1, x_2, ..., x_n)$ là tập tín hiệu vào có giá trị (0,1)

$Y = (y_1, y_2, ..., y_m)$ là tập các đáp ứng ra có giá trị (0,1).

Từ đó ta có quan hệ toán học tổng quát của hệ tổ hợp là :

$$Y = f(X)$$

Tổng quát hơn có thể nói một ôtômat không có nhớ M có các quan hệ phụ thuộc là :

$$M = (X, Y, f)$$

Trong đó f là hàm số hay là cấu tạo mạch điện.

§3.2. NGUYÊN LÝ TỔNG HỢP VÀ PHÂN TÍCH HỆ TỔ HỢP

3.2.1. BÀI TOÁN PHÂN TÍCH ÔTÔMAT KHÔNG CÓ NHỚ

Bài toán phân tích là một bài toán ngược, từ mạch điện lôgic có trước, thông thường là một mạch điện bị hỏng, ta phải tìm lại chức năng hoạt động tức là bảng chân lý hay bảng thật của mạch điện đó. Dựa vào đó có thể biết mạch điện đó bắt đầu hoạt động sai từ khâu nào. Việc phân tích ôtômat không có nhớ được tiến hành qua bước sau :

- Đặt biến số (X) cho các đầu vào của mạch điện.

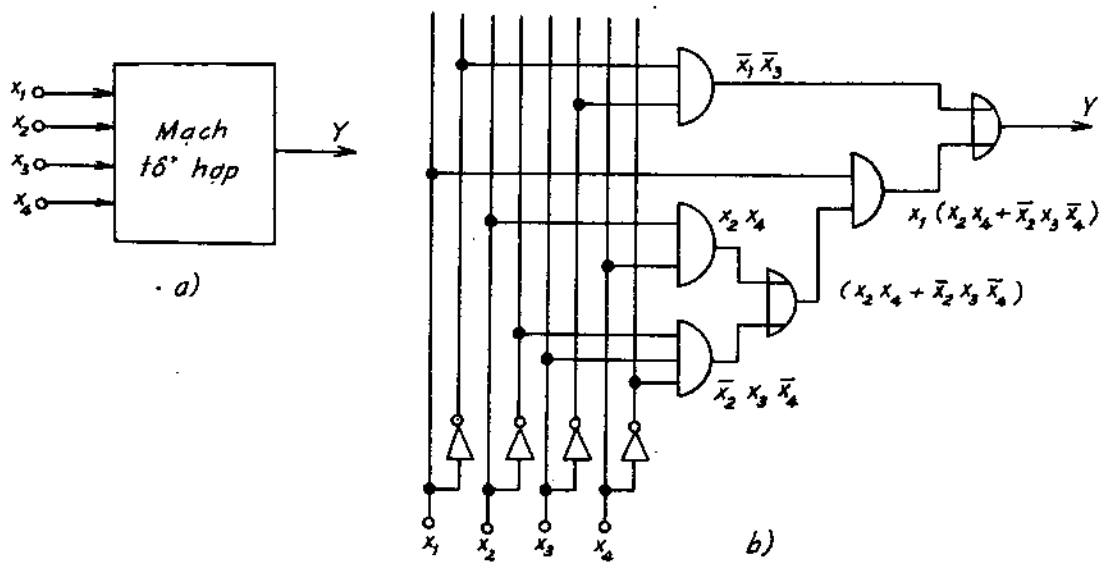
- Viết phương trình toán ở đầu ra của mỗi mạch lôgic theo các biến tín hiệu vào. Từ đó tìm ra phương trình toán học chức năng của mạch điện đó.

- Ánh xạ phương trình chức năng vào bìa Kácnô, vì bìa Kácnô cho ta mọi tín hiệu vào và mọi đáp ứng ra của mạch điện cũng như của một hàm số.

- Sau đó từ bìa Kácnô chuyển thành bảng thật.

Minh họa các bước phân tích một mạch điện lôgic không có nhớ qua ví dụ sau.

Ví dụ : Thực hiện phân tích chức năng mạch tổ hợp 4 biến cho ở hình 3.2.



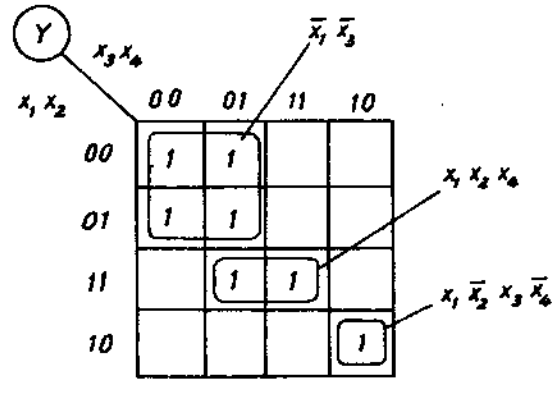
Hình 3.2. Sơ đồ mạch tổ hợp cho trước.

Từ sơ đồ mạch điện cho trước, sau khi đã ký hiệu biến số cho các tín hiệu vào ta thu được phương trình toán học của mạch tổ hợp dưới dạng :

$$Y = \bar{x}_1 . \bar{x}_3 + x_1 . (x_2 . x_4 + \bar{x}_2 . x_3 . \bar{x}_4)$$

$$Y = \bar{x}_1 . \bar{x}_3 + x_1 . x_2 . x_4 + x_1 . \bar{x}_2 . x_3 . \bar{x}_4$$

Sau khi có được phương trình toán học của mạch điện ta ánh xạ phương trình toán học đó vào bìa Kácnô và thu được bảng chân lý hay bảng thật của mạch tổ hợp đó được chỉ ra trên hình 3.3.



Bảng chức năng

x_1	x_2	x_3	x_4	Y
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

b)

Hình 3.3. Bảng chức năng của mạch.

Với cách làm tương tự ta có thể phân tích hay tìm lại chức năng hay bảng thật của bất kỳ một mạch lôgic tổ hợp nào trên bất kỳ đầu ra nào của hệ thống số.

3.2.2. BÀI TOÁN TỔNG HỢP HỆ TỔ HỢP

Bài toán thiết kế ôtômat không có nhớ là thực hiện từ yêu cầu chức năng hay từ bảng chân lý, xây dựng mạch điện tối thiểu hay mạch đơn giản nhất thực hiện chức năng đề ra. Cách thiết kế hệ tổ hợp đã đề cập trong phần toán lôgic. Muốn thiết kế hay tổng hợp một ôtômat không có nhớ cần qua các bước thực hiện như sau :

- Chuyển ý muốn chức năng thành bảng chân lý.

- Rút gọn bảng chân lý (hàm lôgic). Bước này có hai cách : biến đổi hàm số theo toán học hoặc ánh xạ bảng chân lý vào bìa Kácnô rồi khoanh dán rút gọn trên đó.

- Từ phương trình toán học đã được rút gọn, dựa vào đó vẽ ra mạch điện của ôtômat cần thực hiện.

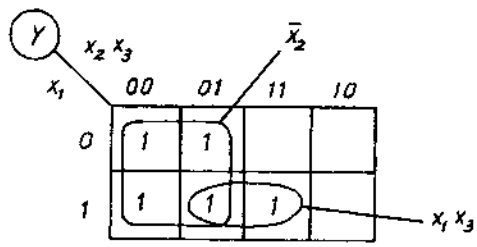
Các bước thiết kế mạch điện được minh họa trong ví dụ sau.

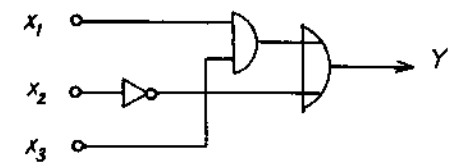
Ví dụ : Thực hiện ôtômat không có nhớ với 3 biến vào có bảng chân lý hay bảng thật cho trước chỉ ra trên hình 3.4.

Bảng thật

x_1	x_2	x_3	Y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

a)





c) $Y = \bar{x}_2 + x_1 x_3$

Hình 3.4. Mạch điện thực hiện bảng chức năng.

Với cách làm đơn giản như vậy ta có thể tổng hợp được các hệ tổ hợp với chức năng lôgic bất kỳ. Sau khi có sơ đồ nguyên lý trên hình 3.4c, tiếp theo tìm linh kiện và mạch lôgic lắp ráp ra mạch điện cụ thể của ôtômat.

Ví dụ trên cho bài toán thiết kế ôtômat là một ví dụ đơn giản. Trong thực tế sẽ có những chức năng phức tạp hơn, có nhiều đầu vào (X) và nhiều đầu ra (Y) hơn, sau khi rút gọn hàm số ta có thể nhóm kết hợp hàm số thêm một lần nữa (không phải là thực hiện rút gọn), khi đó sẽ tạo ra hàm hay mạch lôgic có nhiều tầng mạch.

Sau khi có sơ đồ mạch điện nguyên lý ta chỉ cần lắp ráp mạch điện cho ôtômat là xong. Trong thực tế, khi lắp ráp mạch điện ta sẽ gặp một hiện tượng rất đặc biệt mà chỉ có trong mạch điện của hệ tổ hợp. Đó là hiện tượng hazard, sai nhầm hay còn gọi là "dỏng đảnh", là hiện tượng mạch tốt, linh kiện không hỏng nhưng hoạt động chức năng lúc được lúc không. Đây là một hiện tượng rất nguy hiểm chỉ xảy ra trong hệ tổ

hợp, vì mạch tổ hợp thường được dùng làm mạch điều khiển. Mục tiếp theo ta nghiên cứu bản chất của hazard và tìm biện pháp khắc phục.

§3.3. HAZARD TRONG HỆ LÔGIC TỔ HỢP

3.3.1. KHÁI NIỆM

Việc tổng hợp xây dựng nên mạch lôgic tổ hợp nhìn chung không có gì phức tạp, vì chỉ cần có phương trình toán học ta dựa vào phương trình toán học đó sẽ vẽ ra được mạch điện, và lắp ráp thành hệ thống điều khiển. Nhưng trong thực tế không phải mạch tổ hợp nào xây dựng lên cũng có thể hoạt động tốt được, nguyên nhân do kết cấu về điện của hệ thống lôgic tổ hợp gây ra, hiện tượng hoạt động không ổn định xẩy ra trong hệ tổ hợp được gọi là *hazard*.

Hazard còn được gọi là "sai nhầm", nhưng cũng có thể được gọi chính xác hơn là "đỏng đảnh", hoạt động lúc được lúc không của hệ lôgic. Sự đỏng đảnh sai nhầm này xẩy ra trong một mạch điện hoàn toàn không có hỏng hóc linh kiện, khác với hỏng thông thường như ở TV hay đài, ở đây mạch hoàn toàn tốt về linh kiện, nhưng điều khiển chức năng lúc được lúc không. Hiện tượng của hazard trong mạch lôgic tổ hợp có thể gặp là :

- Hazard chỉ xuất hiện một lần và không bao giờ gặp lại nữa.

- Hazard có thể xuất hiện nhiều lần, có thể xuất hiện theo chu kỳ và cũng có thể xuất hiện phi chu kỳ.

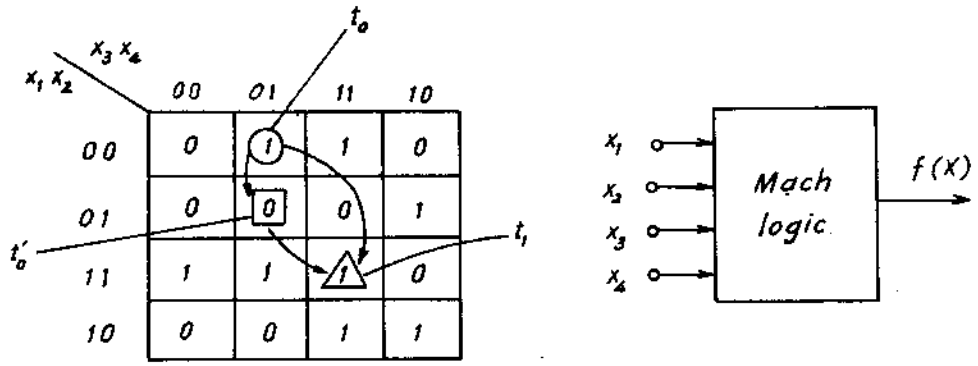
- Hazard có thể do chính chức năng của mạch điện gây ra. Đây là trường hợp khó giải quyết nhất khi thiết kế. Muốn tìm cách khắc phục hay "sửa chữa" đối với loại "hỏng" (hazard) trong hệ lôgic tổ hợp, trước tiên phải tìm hiểu bản chất của hazard là gì? Nguyên nhân hazard xuất hiện từ đâu? Vì sao mạch lôgic lại hoạt động "đỏng đảnh"?

Như ta đã biết, một trong những đặc tính quan trọng của mạch điện khi làm việc đó là quán tính, độ linh động hay còn được gọi là sự chậm trễ của mạch. Chính sự chậm trễ (quán tính) của mạch làm cho tín hiệu từ đầu vào không thể được truyền tức khắc tới đầu ra của mạch điện, điều đó làm cho các thiết bị bị điều khiển không thể có phản ứng tức khắc đối với tín hiệu đưa vào (X). Do tất cả các mạch điện đều có quán tính hay thời gian trễ nhất định, ngay cả ở các mạch vi điện tử cũng có thời gian trễ (τ). Điều này ta có thể thấy ngay trong các sổ tay (catalog) sản xuất IC của từng hãng, ví dụ hãng sản xuất IC nổi tiếng "Texas Instruments" thời gian trễ của một khóa IC đóng mở τ = 5 đến 20 ns, đó là quán tính mạch do chế tạo, khi dùng các IC lôgic đó xây dựng nên hệ thống điện tử sẽ làm cho hệ thống có quán tính hay thời gian trễ của hệ thống điện tử. Ngay cả sự thay đổi nhiệt độ môi trường cũng làm cho thời gian trễ hay quán tính mạch điện thay đổi. Khi thời gian trễ thay đổi sẽ làm cho hoạt động của hệ thống lôgic thay đổi gây ra sai lệch khi điều khiển, đó chính là hazard.

Bản chất của hazard trong hệ thống lôgic là do sự chậm trễ của tín hiệu hay quán tính của mạch điện gây ra. Việc tìm hiểu bản chất của hazard, nguyên nhân xuất hiện hazard, phân tích và nhận dạng hazard có một ý nghĩa rất quan trọng không những trong tổ hợp xây dựng các hệ thống lôgic, mà còn trong việc chuẩn đoán sai nhầm hay dự báo độ tin cậy trạng thái làm việc của hệ thống, cũng như tìm được cách khắc phục được hiện tượng hazard trong hệ lôgic tổ hợp.

3.3.2. BẢN CHẤT CỦA HAZARD

Để hiểu được nguyên nhân xuất hiện hazard trong hệ lôgic tổ hợp, hazard chỉ xuất hiện trong hệ tổ hợp mà không xuất hiện ở bất cứ một hệ thống điện tử nào khác, ta khảo sát một ví dụ cho một mạch tổ hợp mà hoạt động của nó được mô tả trên bìa K như hình 3.5.



Hình 3.5. Mạch chức năng lôgic.

Giả sử tín hiệu vào X = (x_1, x_2, x_3, x_4) thay đổi giá trị từ : (0 0 0 1) đến 1 1 1 1) hay có thể nói tổng quát là (X) thay đổi giá trị từ Q sang P (Q → P). Nhìn vào bìa K của hàm lôgic (mạch tổ hợp) đã cho, ta thấy đáp ứng ra của mạch lôgic tổ hợp khi tín hiệu vào bị thay đổi có giá trị :

$$f(Q) = f(0001) = ① \rightarrow f(P) = f(1111) = ⚠$$

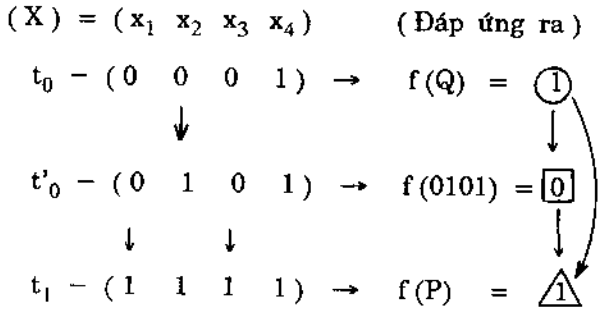
Như vậy tín hiệu vào (X) thay đổi giá trị từ Q = (0001) đến P = (1111) ((0001) → (1111)) làm cho đáp ứng ra của mạch sẽ thay đổi giá trị từ ① sang ⚠(① → ⚠), sự thay đổi giá trị điều khiển ở đầu ra của mạch theo sự thay đổi của tín hiệu vào (X) như vậy là hoàn toàn đúng và chính xác, khi đó hazard hoàn toàn không xuất hiện và không xảy ra điều khiển bị sai nhầm.

Nhưng thực tế xảy ra có thể không như vậy vì : nhìn vào các giá trị lôgic của tín hiệu vào khi thay đổi từ Q = (0001) sang P = (1111) ta thấy các giá trị x_1, x_2 và x_3 bị thay đổi, còn giá trị x_4 (đầu tín hiệu vào x_4) vẫn giữ nguyên. Chúng ta đã biết là mạch điện nào cũng có quán tính có thời gian trễ là (τ) và sự thay đổi giá trị (từ 0 sang 1)

hay ngược lại (từ 1 về 0) của tín hiệu vào đều có thời gian trễ nhất định trước khi tới đầu ra.

Ở đây ta thấy các tín hiệu vào (x_1, x_2, x_3) các giá trị lôgic của chúng đều bị thay đổi, khi ta thay đổi bộ tín hiệu vào (đưa vào một bộ tín hiệu vào mới), tất nhiên chúng sẽ có thời gian chậm trễ khi tới được đầu ra của mạch điện (mặc dù thời gian trễ rất nhỏ có thể hàng μs hay ns). Đồng thời thời gian trễ của mỗi đường tín hiệu vào (biến vào x_i) lại hoàn toàn khác nhau, vì mạch điện IC không thể chế tạo hoàn toàn giống nhau về thời gian trễ hay quán tính của nó, dù cùng một chủng loại mạch IC. Từ đó dẫn tới mỗi một đường tín hiệu vào có thời gian trễ hoàn toàn khác nhau, cụ thể ta có (x_1, x_2, x_3) được thay đổi giá trị đồng thời, cụ thể là cùng thay đổi từ (0 sang 1) (ba tín hiệu vào cùng thay đổi một lần) và chúng có thời gian trễ khác nhau khi tới đầu ra, nói một cách khác là có sự "chạy đua" của tín hiệu vào đó tới đầu ra của mạch điện.

Vì có sự "chạy đua" giữa ba tín hiệu vào (x_1, x_2, x_3) (vì x_4 không thay đổi giá trị nên coi như x_4 không đua) ở đây ta giả sử x_2 chạy nhanh hơn (có thời gian trễ nhỏ hơn) x_1 và x_3 (coi như thời gian trễ của x_1 và x_3 giống nhau và trễ lâu hơn so với x_2). Quan hệ đó của các tín hiệu vào ta có thể viết :



Do ta giả thiết x_2 "chạy" nhanh hơn x_1 và x_3 cho nên giá trị của x_2 từ 0 sang 1 ($0 \rightarrow 1$) sẽ xong trước trong khi giá trị của (x_1, x_3) vẫn giữ nguyên chưa kịp thay đổi vì "chạy chậm" hơn. Sau một thời gian thì (x_1, x_2) mới từ 0 thành 1 ($0 \rightarrow 1$) theo như yêu cầu nhiệm vụ thay đổi tín hiệu vào (X) đã đưa ra.

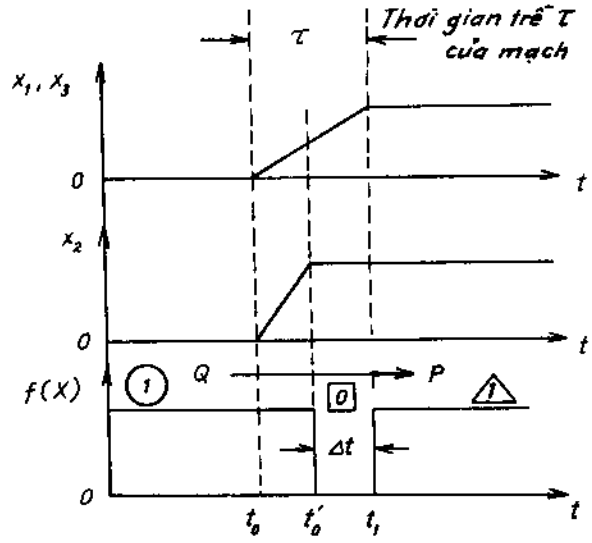
Quan hệ "chạy đua" giữa ba tín hiệu vào có thể được minh họa trên biểu đồ thời gian ở hình 3.6, với chú ý khi vẽ là : do chỉ quan tâm tới thời gian trễ của mạch là τ, cho nên quá trình thay đổi giá trị lôgic $\big((0 \rightarrow 1)$ và $(1 \rightarrow 0)\big)$ là như nhau và không cần xét đến khi vẽ.

Do giả thiết x_2 thời gian trễ nhỏ nên "đến trước" làm cho bộ tín hiệu vào bị thay đổi :

$$t_0 \qquad t_0' \qquad t_1$$

$$(0001) \rightarrow (0101) \rightarrow (1111)$$

Làm cho đáp ứng ra hay hàm f(X) thay đổi : $1 \rightarrow 0 \rightarrow 1$, trong khi nhẽ ra giá trị hàm f(X) không được thay đổi trong quá trình chuyển đổi từ Q sang P $\big(f(Q) = f(P) = 1\big)$. Tức là tín hiệu điều khiển ở đầu ra là cố định không thay đổi mặc dù bộ tín hiệu vào thay đổi giá trị theo yêu cầu của hàm lôgic (chức năng của mạch điện).



Hình 3.6. Hiện tượng hazard.

Do x_2 "chạy nhanh" hơn (x_1, x_3) đã làm xuất hiện trong khoảng thời gian Δt một xung zêrô nhất thời $\left(f(0101) = \boxed{0}\right)$. Như vậy trong thời gian quán tính hay thời gian trễ τ của mạch tín hiệu ra đã thay đổi từ 1 đến 0 rồi đến 1 (mà nhẽ ra không được thay đổi), tạo ra một xung kim nhất thời. Hiện tượng xuất hiện một xung zêrô ở đầu ra của mạch được gọi là hiện tượng hazard, và đó chính là *hazard nhất thời*. Hazard loại này chỉ xuất hiện trong thời gian trễ τ sau đó lại mất đi ngay. Như vậy ta có thể nói rằng sự chạy đua của tín hiệu vào gây ra hazard, hay thời gian trễ của mạch điện sẽ làm xuất hiện hazard, là tín hiệu xung điều khiển không mong muốn ở đầu ra.

Xung hazard là một xung rất bé xuất hiện ở đầu ra mạch lôgic tổ hợp vì ($\Delta t < \tau$), mà τ chỉ là thời gian trễ của mạch, chính vì vậy xung hazard có thể xuất hiện nhưng không nguy hiểm, không gây ra điều khiển sai nhầm được, vì xung hazard quá hẹp nên năng lượng của nó không đủ lớn để có thể kích nhầm hay kích nối mạch điện tiếp theo để gây ra điều khiển sai, do đó dù có xung hazard nhưng mạch điện vẫn hoạt động tốt. Xung hazard chỉ thật sự nguy hiểm khi độ rộng Δt của nó được lớn lên, khi đó xung hazard sẽ có thể có năng lượng đủ lớn kích lật chuyển mạch điện tiếp theo gây ra hiện tượng điều khiển lôgic sai nhầm.

Như vậy có thể thấy với bộ tín hiệu vào thay đổi kiểu khác, với tổ hợp khác sẽ không xuất hiện được xung hazard trong ví dụ trên. Hay với một chức năng khác dù cho có chạy đua tín hiệu vào giữa (x_1, x_2 và x_3) như ví dụ trên, nhưng nếu f(0101) = 1 (vì là chức năng khác) thì hazard cũng không thể xuất hiện do xung zêrô nhất thời không có. Qua đó chúng ta thấy hazard hay là sự "đỏng đảnh" của mạch lôgic điều khiển xuất hiện rất ngẫu nhiên cho dù với một mạch điện có các linh kiện hoàn toàn tốt.

3.3.3. CÁC LOẠI HAZARD TRONG HỆ LÔGIC TỔ HỢP

Trước hết ta có một số định nghĩa tên gọi khi nói về hazard như sau

$$Q = (x_1, x_2, \ldots, x_i, x_{i+1}, \ldots, x_n)$$

$$P = (x_1, x_2, \ldots, \bar{x}_i, \bar{x}_{i+1}, \ldots, x_4)$$

Ở đây P và Q là tập tín hiệu vào của mạch, nhưng yêu cầu giữa P và Q cần có số lượng vị trí thay đổi giá trị lôgic ≥ 2, vì chỉ khi tập tín hiệu vào thay đổi giá trị lôgic đồng thời với ít nhất ở 2 vị trí (2 biến số) thì mới xuất hiện 'chạy đua" tín hiệu vào, và khi đó hazard mới có khả năng xuất hiện. Còn nếu tín hiệu vào chỉ thay đổi giá trị lần lượt trên từng đầu vào một (giống như đạo hàm riêng bậc cao loại I) thì sẽ không có chạy đua tín hiệu và hazard không thể xuất hiện được, vì bản chất của hazard xuất hiện do sự chạy đua của tín hiệu vào.

Định nghĩa 3.1 : Nếu tập tín hiệu vào (X) thay đổi từ Q sang P thì được gọi là có sự chuyển đổi từ Q sang P (Q → P).

Định nghĩa 3.2 : Hazard nhất thời xuất hiện trong lôgic tổ hợp là hiện tượng tín hiệu ra (đáp ứng ra) của mạch xuất hiện giá trị lôgic (tín hiệu điều khiển cụ thể) khác với các giá trị lôgic quy định cho chúng theo hàm Boole, trong thời gian chuyển đổi từ (Q → P).

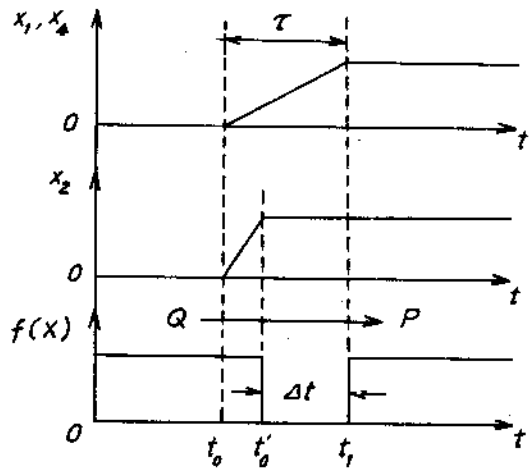
3.3.3.1. Hazard tĩnh trong mạch lôgic

Do có sự "chạy đua" giữa các tín hiệu vào với nhau trong thời gian chuyển đổi từ (Q→P) mà xuất hiện hazard. Nếu f (Q) = f (P) tức là có sự thay đổi của tín hiệu vào nhưng sự điều khiển ở đầu ra của mạch lôgic vẫn không đổi (vẫn được giữ nguyên) dù là 0 hay 1 theo chức năng đề ra, nhưng xuất hiện hazard, khi số lượng tín hiệu chạy đua không nhiều, thì đó là hazard tĩnh.

Định nghĩa 3.3 : Hazard nhất thời xuất hiện trong mạch lôgic tổ hợp trong thời gian chuyển đổi (Q → P) được gọi là hazard tĩnh nếu và chỉ nếu f (Q) = f (P).

Theo định nghĩa này, hazard nhất thời cũng chính là hazard tĩnh, tức là loại hazard chỉ xuất hiện như một xung không theo quy định của hàm lôgic mà thôi. Hiện tượng của hazard tĩnh cũng giống như khi xem xét bản chất của hazard nhất thời đã nêu ở phần trên. Nó không nguy hiểm, vì độ rộng của xung hazard tĩnh Δt luôn nhỏ hơn τ quán tính của mạch ($\Delta t < \tau$) hệ thống lôgic hoạt động bình thường dù có xuất hiện hazard.

Nhưng hazard tĩnh nguy hiểm ở chỗ, nó có thể gây ra "sai nhầm" cho điều khiển của hệ thống lôgic khi giá trị độ rộng hazard (Δt) lớn lên, điều này sẽ xảy ra khi sự "chạy đua" của tín hiệu vào quá "khốc liệt" tức là có tín hiệu vào "chạy" quá nhanh còn tín hiệu khác lại "chạy" quá chậm, hiện tượng đó được minh họa trên hình 3.7.



Hình 3.7. Chạy đua ở hazard tĩnh.

Ta thấy x_2 trong quá trình "chạy đua" (thay đổi giá trị lôgic) đã "chạy" nhanh hơn tốc độ quán tính - trễ của (x_1, x_4), thể hiện ở hình vẽ độ dốc xung x_2 dốc hơn, điều đó làm cho Δt của xung hazard tăng lên. Khi độ rộng Δt của xung hazard lớn lên, năng lượng của xung tăng theo, khi đó xung hazard trở nên "nguy hiểm" hơn vì nó có thể kích lật chuyển một mạch điện ở tiếp sau hệ thống mạch lôgic, gây ra hiện tượng điều khiển "sai nhầm" trong hệ thống tổ hợp lôgic.

3.3.3.2. Hazard động trong hệ lôgic tổ hợp

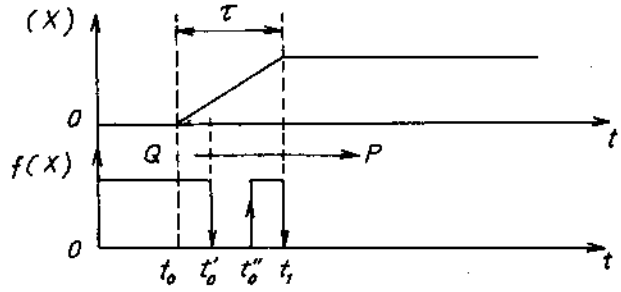
Trong thực tế khi thay đổi bộ tín hiệu vào của mạch lôgic ứng với quá trình chuyển đổi (Q → P) có thể có rất nhiều tín hiệu vào cùng thay đổi, giống như hiện tượng ở đạo hàm bậc cao loại II có thể đạo hàm một tập biến (nhiều biến thay đổi giá trị lôgic) một lúc. Khi có sự chạy đua của các tín hiệu vào tới đầu ra của mạch. Ví dụ trường hợp

Q = (00000)

P = (01101) dễ dàng nhận thấy có đua (X)

$$
\begin{array}{lccccccccc}
(X) - & (x_1 & x_2 & x_3 & x_4 & x_5) & & & & \\
t_0 - & 0 & 0 & 0 & 0 & 0 & \rightarrow & f(Q) & = & 1 \\
 & & & \downarrow & & & & & & \downarrow \\
t_0' - & 0 & 0 & 1 & 0 & 0 & \rightarrow & f(X') & = & 0 \\
 & & \downarrow & & & & & & & \downarrow \\
t_0'' - & 0 & 1 & 1 & 0 & 0 & \rightarrow & f(X'') & = & 1 \\
 & & & & & \downarrow & & & & \downarrow \\
t_1 - & 0 & 1 & 1 & 0 & 1 & \rightarrow & f(P) & = & 0
\end{array}
$$

Hiện tượng đó được minh họa trên hình 3.8.



Hình 3.8. Hazard động.

Do có nhiều tín hiệu vào đồng thời thay đổi giá trị lôgic từ (0 sang 1 và từ (1 về 0), mà mỗi một tín hiệu vào (một biến x_i) có tốc độ "chạy" khác nhau hay quán tính khác nhau vô tình làm cho giá trị hàm f(X) ở đầu ra xuất hiện thay đổi như hình 3.8. Chú ý ở đây do nhiều tín hiệu vào có thời gian "chạy" khác nhau nên ta chỉ biểu thị quá trình biến đổi tín hiệu (X) với thời gian trễ của mạch là τ. Hiện tượng tín hiệu ra f(X) thay đổi giá trị (1 → 0 → 1 → 0) được gọi là hazard động, tức là xuất hiện nhiều xung không cần thiết trong khoảng thời gian trễ của mạch (τ). Từ hiện tượng đó ta có :

Định nghĩa 3.4 : Hazard nhất thời xuất hiện trong mạch lôgic trong thời gian

chuyển đổi (Q → P) được gọi là hazard động nếu và chỉ nếu f (Q) = f̄ (P), nhưng khi xuất hiện hazard nhất thời động, thì tín hiệu xung hazard phải thay đổi ít nhất ba lần, chẳng hạn từ (1 → 0 → 1 → 0).

Khi có nhiều tín hiệu vào cùng thay đổi một lần và giữa các tín hiệu vào đó có thời gian quán tính khác nhau nên có thể gây ra hazard động. Mà hazard động trong thời gian chuyển đổi (Q → P) có thể xuất hiện rất nhiều xung không cần thiết. Như vậy trong thời gian chuyển mạch τ đã rất nhỏ rồi, mà lại xuất hiện nhiều xung hazard nhỏ hơn trong khoảng thời gian τ đó, từ đó ta dễ dàng hiểu rằng xung hazard động không có gì nguy hiểm cả, vì một xung bị chia thành nhiều xung con thì năng lượng cũng nhỏ và độ rộng xung quá bé không đủ kích được mạch khác. Hiện tượng hazard động ta có thể tưởng tượng như là khi đèn đang sáng ta cho tín hiệu vào thay đổi (Q → P) để tắt đèn (f (Q) = f̄ (P)) thì đèn tắt xong rồi lại hơi sáng lên sau đó mới tắt hẳn. Hazard động ít có khả năng gây ra điều khiển "sai nhầm" trong mạch lôgic tổ hợp.

3.3.3.3. Hazard hàm số trong mạch tổ hợp

Ta đã biết hazard có thể xuất hiện do chính chức năng của mạch trong cả hai trường hợp là hàm f (X) lấy giá trị lôgic là 0 hoặc 1.

Định nghĩa 3.5 : Hazard nhất thời được gọi là hazard hàm số trong thời gian chuyển đổi tín hiệu vào (Q → P) nếu thỏa mãn.

- f (Q) = f (P)

- Hàm f(X) lấy cả hai giá trị 1 và 0 trong thời gian chuyển đổi từ (Q → P).

Ta có thể hiểu là : trong thời gian chuyển đổi (Q → P) thì hàm lôgic không thay đổi giá trị (f (Q) = f (P)), nhưng nếu lấy f (Q) = f (P) = 0 của hàm lôgic đó thì hazard vẫn xuất hiện, hay lấy f (Q) = f (P) = 1 thì hazard cũng vẫn xảy ra, tức là hàm f (X) lấy cả hai giá trị 1 và 0 trong thời gian chuyển đổi (Q → P) thì hazard nhất thời vẫn xuất hiện. Hiện tượng đó được gọi là hazard hàm số. Trong thực tế có những hàm số hazard nhất thời chỉ xuất hiện khi điều khiển lôgic là 1 (f (X) = 1) còn điều khiển lôgic ở đầu ra là 0 thì lại không có hazard nhất thời xuất hiện, và ngược lại có thể điều khiển ra thì không bị hazard.

Độ nguy hiểm của hazard hàm số cũng giống như đối với hazard tĩnh, nhưng nó nguy hiểm hơn một mức nữa là vì bất kỳ quá trình điều khiển nào (0 hay 1) thì đều có khả năng xuất hiện hazard, hay đều có khả năng gây ra "sai nhầm" khi điều khiển trong mạch.

3.3.3.4. Hazard lôgic trong mạch tổ hợp

Còn một hiện tượng hazard nữa xuất hiện trong mạch tổ hợp, đây là loại hazard nguy hiểm nhất, hay gây ra điều khiển "sai nhầm" nhiều nhất trong các hệ thống mạch tổ hợp điều khiển, nó có tên gọi là "hazard lôgic". Bản chất xuất hiện hazard lôgic ta có thể hiểu như sau.

Khi tập tín hiệu vào hay tập biến vào của hàm lôgic thay đổi đồng thời nhiều biến số hay nhiều tín hiệu vào trong thời gian chuyển đổi (Q → P), mà mỗi một tín hiệu vào có thời gian trễ hay tốc độ "chạy đua" khác nhau, trong quá trình chạy đua đó, ví dụ gặp phải trường hợp là : ở đây ta lấy Q = (00000)

P = (01101)

(X)	x_1	x_2	x_3	x_4	x_5		
t_0	0	0	0	0	0	→	f(Q) = 1
			↓				↓
t_0'	0	0	1	0	0	→	f(X') = 0
		↓					↓
t_0''	0	1	1	0	0	→	f(X'') = 0
					↓		↓
t_0'''	0	1	1	0	1	→	f(X''') = 0
	↓						↓
t_1	1	1	1	0	1	→	f(P) = 1

Hiện tượng hazard lôgic được mô tả trên hình 3.9.

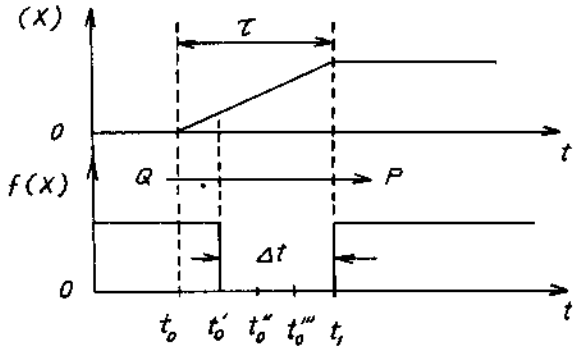
Từ hiện tượng đó của hazard ta có định nghĩa

Định nghĩa 3.6 : Hazard nhất thời được gọi là hazard lôgic trong thời gian chuyển đổi từ (Q → P) nếu :

- f (Q) = f (P)

- Hàm f (X) chỉ lấy một giá trị (hoặc 1 hoặc 0)

- Trong thời gian chuyển đổi từ (Q → P) xuất hiện một xung hazard có độ rộng Δt lớn ở đầu ra, khi quá trình chạy đua ngẫu nhiên của các tín hiệu vào tạo ra hàm f(X) có cùng một giá trị lôgic.

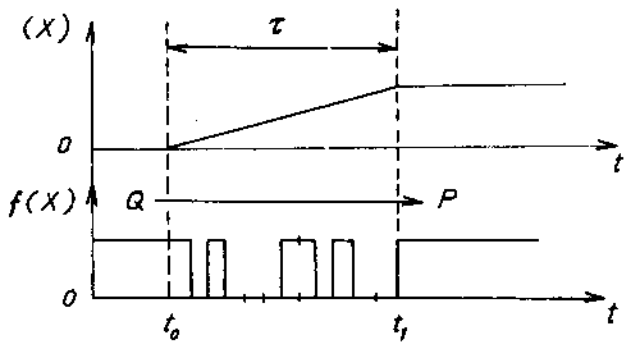


Hình 3.9. Hazard lôgic.

Như vậy trong quá trình chuyển đổi từ (Q → P) của tập tín hiệu vào, có nhiều tín hiệu cùng thay đổi giá trị và hàm lôgic vô tình hay ngẫu nhiên xảy ra trường hợp có cùng một giá trị lôgic hazard ở đầu ra f(X) của mạch. Điều đó tạo nên một xung hazard ở đầu ra của mạch có độ rộng Δt lớn lên rất nhiều, khi Δt lớn làm cho xung hazard có năng lượng lớn đủ khả năng kích lật chuyển một mạch tiếp theo sau mạch điều khiển, điều đó gây ra hiện tượng điều khiển "sai nhầm" trong hệ thống lôgic tổ hợp. *Đó là điều vô cùng nguy hiểm đối với các hệ thống tổ hợp cỡ lớn có nhiều đầu*

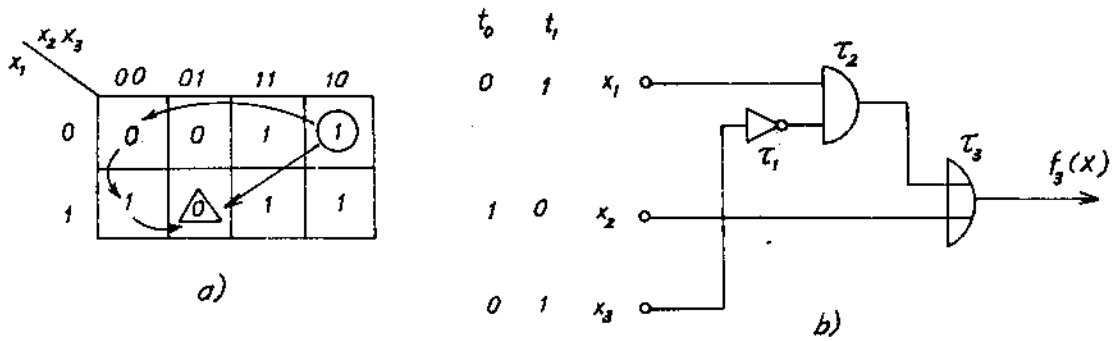
vào. Ta thấy rõ là hazard lôgic là hazard nguy hiểm nhất trong "họ nhà hazard" xuất hiện trong hệ tổ hợp.

Phần trên chúng ta đã biết được bản chất của hazard xuất hiện là do sự chạy đua của tín hiệu vào trong mạch lôgic trong thời gian chuyển đổi từ (Q → P), và hiện tượng hazard chỉ có ở trong mạch lôgic tổ hợp, mà không xuất hiện trong bất kỳ một mạch điện tử cũng như hệ thống điện tử nào khác. Đồng thời chúng ta cũng nên lên các định nghĩa hay các tên gọi của từng hazard xuất hiện tương ứng với các hiện tượng chạy đua của tín hiệu vào, nhưng trong thực tế hoạt động của quá trình chuyển đổi từ (Q → P) trong mạch lôgic tổ hợp ta rất phức tạp, rất ít khi ta gặp từng loại hazard riêng biệt. Trong thực tế quá trình chuyển đổi từ (Q → P) ta thường gặp sự tổ hợp hỗn loạn các loại hazard đó, khi thường gặp số lượng tín hiệu vào thay đổi rất lớn hay thay đổi nhiều. Hiện tượng đó được minh họa trên hình 3.10.



Hình 3.10. Hiện tượng tổng quát xuất hiện hazard.

Trên hình 3.11 là một mạch điện lôgic có thể xuất hiện hazard, hàm lôgic trên bìa K và mạch điện của nó.

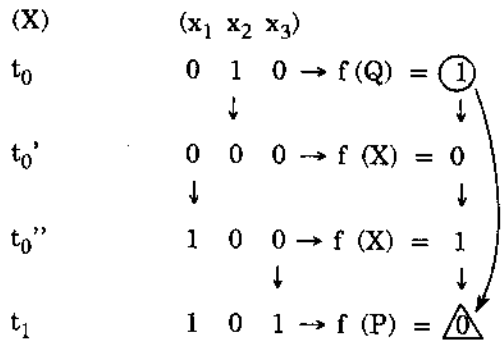


Hình 3.11. Mạch lôgic có chạy đua tín hiệu vào.

Nhìn vào sơ đồ mạch điện ta dễ dàng nhận thấy tín hiệu vào x_2 sẽ "chạy" nhanh nhất vì nó được nối trực tiếp tới đầu ra, tiếp theo sẽ là tín hiệu x_1 vì chỉ phải vượt qua trễ của mạch VÀ là τ_2, còn x_3 sẽ chạy chậm hơn cả vì phải qua trễ τ_1 của mạch đảo sau đó qua τ_2 của mạch VÀ. Ta giả sử có bộ tín hiệu vào làm xuất hiện quá trình chuyển đổi từ (Q → P)

$$Q = (0\ 1\ 0)$$
$$\downarrow\ \downarrow\ \downarrow$$
$$P = (1\ 0\ 1)$$

Quá trình chuyển đổi (Q → P) đó cho phép xuất hiện chạy đua đồng thời các tín hiệu x_1, x_2 và x_3 có khả năng chậm trễ như đã biết, hiện tượng xảy ra là :



Do f (Q) = 1 và f (P) = 0 tức là f (Q) = $\overline{f}$ (P) mà xuất hiện hazard rất rõ cho nên đã xuất hiện trong mạch hazard nhất thời động. Như vậy xét về mặt kỹ thuật có thể thấy ngay được sự xuất hiện của hazard đồng thời biết ngay được mạch có thể có hazard hay không chưa cần cho nó hoạt động. Người ta cũng có những phương pháp khác như phương pháp thuật toán dùng đạo hàm để phát hiện khả năng xuất hiện hazard. Nhưng bằng phương pháp toán học chỉ phát hiện có khả năng xẩy ra hazard mà thôi, còn từ khi phát hiện được có hazard cho tới khi tìm được cách khắc phục được nó lại không dễ dàng. Mỗi một mạch điều khiển có thể có sự xuất hiện hazard khác nhau, thậm chí có mạch lôgic cho bộ tín hiệu vào (Q → P) có số lượng biến số "chạy đua" rất lớn nhưng hazard lại không xuất hiện, nhưng có mạch rất đơn giản và chỉ vài tín hiệu vào chạy đua thôi thì hazard lại xuất hiện và gây ra "sai nhầm". Vì vậy muốn khắc phục được hazard thì trước hết phải căn cứ vào mạch điện cụ thể của nó, sau đó dùng kỹ thuật phân tích phát hiện khả năng xuất hiện hazard, rồi mới tìm cách khắc phục được hazard. Đây là một quá trình cần tỉ mỉ và hiểu rõ mạch điện. Dưới đây là một vài phương pháp nhằm khắc phục và hạn chế sự xuất hiện hazard trong hệ thống lôgic điều khiển.

3.3.3.5. Các biện pháp khắc phục hazard

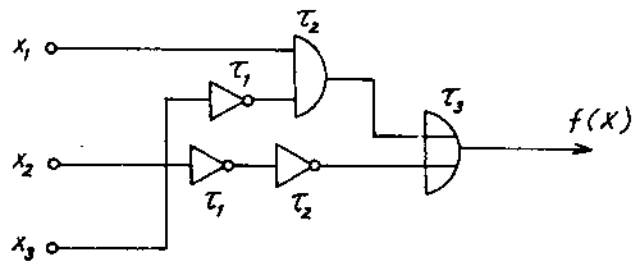
Ta đã biết hazard hay "sai lầm" xuất hiện do có sự chạy đua của tín hiệu vào trong hệ lôgic tổ hợp, nói cách khác hazard xuất hiện là do sự khác nhau về thời gian của đường truyền tín hiệu từ đầu vào đến đầu ra của mạch, từ đó ta có những cách khắc phục hazard như sau :

- Biện pháp trước hết và đơn giản nhất làm biến mất hazard là không để xuất hiện quá trình chạy đua của các tín hiệu vào trong mạch lôgic, có nghĩa là cho tín hiệu vào thay đổi giá trị lôgic chỉ trên một đầu vào tín hiệu, hoạt động giống như đạo hàm riêng bậc cao loại I. Khi chỉ có một tín hiệu vào "chạy" ở trong mạch lôgic thì sẽ không còn "đua" tín hiệu nữa và chắc chắn hazard không thể xuất hiện. Nhưng từng tín hiệu vào

thay đổi giá trị lôgic sẽ làm cho mạch hoạt động chậm chạp, hơn nữa không phải quá trình điều khiển nào cũng cho phép làm như vậy, thông thường có sự thay đổi nhiều tín hiệu vào một lúc.

- Tiếp theo khi phải chấp nhận quá trình chuyển đổi (Q → P) có nhiều tín hiệu thay đổi hay có nhiều biến (X) chạy đua. Cách khắc phục là chọn giá trị linh kiện hay IC có thời gian trễ τ rất nhỏ, vì ta biết hazard chỉ xuất hiện trong thời gian quán tính τ của mạch, khi τ càng nhỏ có nghĩa là xung hazard có độ rộng Δt nhỏ, khi đó nó không đủ năng lượng kích lật mạnh tiếp theo hay không đủ "sức" gây ra điều khiển "sai nhầm". Nhưng khi chọn linh kiện lắp ráp hệ thống hay chọn IC có τ rất nhỏ đồng nghĩa với việc chọn linh kiện IC có chất lượng cao, mà chất lượng linh kiện cao sẽ làm cho giá thành của hệ điều hành tăng, đây cũng là một vấn đề cần phải quan tâm khi thiết kế. Để giá thành hệ điều khiển không cao mà hazard không xuất hiện, đảm bảo chất lượng điều khiển của mạch cao, ta sẽ phải chọn linh kiện IC chất lượng cao ở những vị trí quan trọng và cần thiết nhất ở trong mạch điện điều hành, điều này liên quan tới mạch có chức năng cụ thể.

- Nhưng nếu ta chấp nhận có sự chạy đua tín hiệu vào (X) trong quá trình chuyển đổi (Q → P), đồng thời không dùng linh kiện có chất lượng cao để tiết kiệm kinh phí mà vẫn đảm bảo hệ thống lôgic điều khiển hoạt động tốt mà hazard không xuất hiện, có thể dùng phương pháp khắc phục hazard bằng cách thêm các mạch trễ trên đường truyền tín hiệu, để đảm bảo cho thời gian chạy đua của tín hiệu ở trong mạch từ đầu vào đến đầu ra ở các đường tín hiệu truyền qua là như nhau hay xấp xỉ nhau. Phương pháp được minh họa ở trên hình 3.12.



Hình 3.12. Mạch chống chạy đua tín hiệu vào.

Ta đã biết x_2 là đường tín hiệu vào chạy nhanh tới đầu ra, vậy trên đường truyền của x_2 ta cho thêm hai mạch đảo có thời gian trễ τ_1 và τ_2 làm cho tín hiệu trên x_2 xuất hiện cùng một lúc ở đầu ra với x_3, khi đó hazard sẽ không xuất hiện hay giảm bớt hazard. phương pháp này có thể gây ra hazard nếu đường trễ thêm vào lại làm cho x_2 chạy quá chậm và lại phát sinh hiện tượng đua tín hiệu vào.

Để tránh xảy ra hiện tượng đua tín hiệu vào, cần biết chính xác thời gian trễ của τ_1, τ_2, sau đó phải chế ra được mạch đảo có thời gian trễ đúng bằng giá trị của τ_1 và τ_2 đã biết, tiếp theo thêm các mạch đảo (NOT) hay mạch trễ vào trong mạch thiết kế.

- Ở mức cao hơn khi ta phải chấp nhận có sự chạy đua tín hiệu vào trong quá trình chuyển đổi (Q → P), không muốn dùng linh kiện có chất lượng cao (τ <<), đồng thời đã thêm các mạch trễ (không ảnh hưởng tới chức năng lôgic của mạch) nhưng vẫn không thể khắc phục được hết hazard, khi đó ta có thêm một cách nữa khắc phục hazard là dùng xung đồng bộ, tức là ta bất chấp có sự chạy đua của tín hiệu vào, và giữa các đường truyền tín hiệu từ đầu vào tới đầu ra có thời gian trễ hoàn toàn khác nhau. Nhưng tín hiệu truyền lan trong hệ lôgic dù nhanh dù chậm, dù đến trước hay đến sau thì chúng chỉ được lan truyền tiếp khi có sự cho phép của xung đồng bộ. Xung đồng bộ thông thường "chờ" theo đường tín hiệu "chạy" chậm nhất, khi đó các tín hiệu truyền đến sớm phải "chờ" cho đầy đủ các tín hiệu khác khi đó xung nhịp hay xung đồng bộ mới cho phép được truyền tiếp. Nếu cho vào mạch điều khiển thêm xung đồng bộ, cũng có thể làm giảm đáng kể ảnh hưởng tác hại của hazard.

- Trong trường hợp các phương pháp đã nêu ở trên đều được áp dụng, nhưng hiện tượng hazard vẫn xuất hiện trong hệ thống lôgic điều khiển, thì buộc phải thay đổi chức năng điều khiển, tức là thay đổi chức năng của hàm lôgic của hệ thống điều khiển hay phải xây dựng lại một mạch điện khác. Cách khắc phục hazard này xuất phát từ hazard là do chính chức năng của hệ thống lôgic gây ra, cho nên chỉ thay đổi chức năng mới có thể khắc phục được hiện tượng hazard.

Sau khi tìm hiểu được bản chất của hazard hay nguyên nhân gây ra điều khiển "sai nhầm" trong hệ thống điều khiển lôgic tổ hợp, ta đã đề cập các phương pháp hạn chế hay cách khắc phục đồng thời làm giảm bớt khả năng xuất hiện của hazard, từ đó làm tăng thêm chất lượng cũng như độ tin cậy của hệ thống điều khiển của mạch lôgic. Qua đó chúng ta cũng thấy để có được một mạch điện điều khiển tốt, cho chất lượng cao, thì phần cứng xây dựng nên mạch điện mang tính quyết định. Người thiết kế phải hiểu rất kỹ và sâu sắc hệ thống kỹ thuật mình thiết kế thì mới có thể khắc phục được hazard trong mạch điện, cũng như phải thêm hay bớt các mạch điện phụ như thế nào, mà không làm thay đổi chức năng của hệ thống. Từ đó làm cho chất lượng của mạch sẽ cao hơn, giá trị kinh tế của mạch sẽ cao hơn. Điều này cũng dễ dàng nhận thấy trong thực tế là ; vẫn là những mạch điện có cùng một chức năng điều khiển, nhưng mỗi hãng sản xuất đưa ra mạch khác nhau và giá trị kinh tế của các mạch đó cũng khác nhau, tùy thuộc vào trình độ và sự quan tâm tới việc tăng độ tin cậy, tăng chất lượng điều khiển mạch của hãng. Nhưng bản chất vẫn chỉ là làm giảm tối đa khả năng xuất hiện hazard trong mạch.

Chương IV

ÔTÔMÁT CÓ NHỚ

§4.1. KHÁI NIỆM CHUNG VÀ MÔ HÌNH TOÁN HỌC

4.1.1. KHÁI NIỆM CHUNG

Trong chương này chúng ta sẽ nói đến một hệ thống số mà hoạt động của hệ thống này còn quan trọng hơn hoạt động điều khiển của hệ lôgic tổ hợp (Combinational Circuits) đã được trình bầy ở phần trên. Hệ thống này có tên gọi là ôtômat có nhớ (Sequential Circuits) hay là "*hệ dãy*". Hoạt động của hệ thống mạch điện này có tính chất kế tiếp nhau, tức là trạng thái hoạt động của mạch điện ngoài việc chịu ảnh hưởng trực tiếp của tín hiệu tác động vào (X) nó còn phụ thuộc vào sự lưu trữ thông tin trước đó của mạch hay trạng thái bên trong trước đó (S) của chính nó. Nói cách khác các hệ kế tiếp làm việc theo nguyên lý có nhớ, tức là trạng thái tiếp theo của hệ thống ở một thời điểm nào đó có tính kế tiếp các trạng thái của hệ thống ở các thời điểm trước đó được lưu giữ hay ghi nhớ lại.

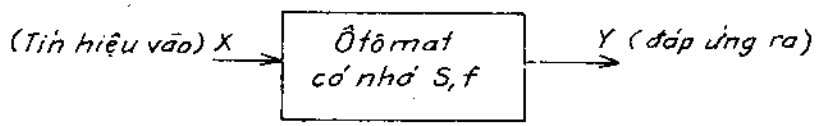
Đối với cuộc sống của con người thì các hoạt động kế tiếp đều tồn tại sự phát triển và còn cho ta "ngữ nghĩa". Đơn giản nhất như cuộc sống tự nhiên xung quanh ta khi ta thấy cái cây mọc lên – trạng thái tiếp theo, thì cái cây đó phải phụ thuộc vào trạng thái trước đó là cái mầm, đồng thời chịu tác động của tín hiệu vào là nước, không khí, ánh sáng cùng khoáng chất .v.v. Như vậy một hoạt động trên cơ sở kế tiếp là kế thừa những kết quả (thông tin) trước đó, rồi từ đó mới sinh ra kết quả tiếp theo (trạng thái tiếp theo) hay thông tin tiếp theo. Ngoài ra bất cứ một hoạt động có tính chất kế tiếp nào đó đều cho ta "nghĩa lý" điều ta muốn có, cũng như "vô nghĩa" điều ta không muốn có. Giống như tiếng nói hay chữ viết đều căn cứ vào kết quả trước đó mới có (cho ra) kết quả tiếp theo, ví dụ chữ HANOI khi ta đánh máy chữ thì chữ H đánh xong và đó là trạng thái trước đó, dựa vào nó tiếp theo ta đánh chữ A, sau chữ A là chữ N... rồi kế tiếp đến các chữ khác. Từ đó cho ta ngữ nghĩa là Hà Nội. Nhưng nếu ta đánh máy có các chữ kế tiếp nhau là STDQA lại cho ta một nội dung vô nghĩa đối với con người. Còn đối với máy thì bộ chữ HANOI hay STDQA là như nhau. Chính vì dựa vào đặc điểm đó của máy chúng ta sẽ xây dựng nên một ôtômat có nhớ hay một hệ thống số hoạt động

kế tiếp nhau theo quy ước "có nghĩa" của con người. Hệ thống đó hoạt động giống như hoạt động của con người là làm theo một "lời dặn dò" hay theo trình tự của chương trình định sẵn do con người đưa ra. Đó chính là hệ dãy mà ta sẽ nói đến cũng như thiết kế xây dựng ra nó trong phần này.

Để xây dựng được ôtômat có nhớ trước hết ta phải tìm cách chuyển các chức năng của hệ thống thành các quan hệ toán học, vì nếu một chức năng chưa mô tả được hay không mô tả được dưới dạng toán học thì sẽ không thực hiện được. Để mô tả được "hệ dãy" dưới dạng toán học, người ta dùng các mô hình toán học hay các bảng chức năng - bảng chuyển biến trạng thái. Đó là cách biểu diễn mới của hàm số mô tả "sự phụ thuộc" hay "sự ràng buộc", vì hàm số trước tiên đó là sự phụ thuộc ràng buộc lẫn nhau.

4.1.2. MÔ HÌNH TOÁN HỌC CỦA ÔTÔMAT CÓ NHỚ

Đối với ôtômat có nhớ, đáp ứng ra của hệ thống mạch điện (Y) không chỉ phụ thuộc trực tiếp vào tín hiệu vào (X) mà còn phụ thuộc vào trạng thái bên trong trước đó (S) của hệ thống. Có thể mô tả sơ đồ khối tổng quát của ôtômat có nhớ như trên hình 4.1.



Hình 4.1. Sơ đồ khối của ôtômat có nhớ

Ở đây : X - tập tín hiệu vào
S - các trạng thái trong trước đó của mạch
Y - đáp ứng ra của hệ thống
f - sự phụ thuộc hay mạch điện.

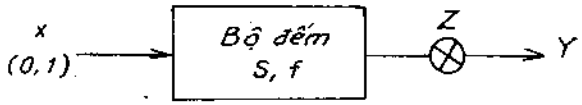
Hoạt động của ôtômat có nhớ được minh họa bằng quan hệ toán học như sau :

$$Y = f(S, X)$$

Quan hệ toán học trên, cho ta hiểu rõ được bản chất hoạt động rất chính xác của ôtômat có nhớ, bản chất hoạt động "kế tiếp" rất rõ của hệ thống số kiểu này.

Để hiểu rõ hơn nữa giữa quan hệ toán học và hoạt động thật sự của ôtômat có nhớ, ta có thể dựa vào bộ đếm làm ví dụ minh họa, vì bộ đếm chính là một ôtômat có nhớ rất điển hình về hoạt động kế tiếp, đồng thời rất gần gũi và dễ hiểu đối với chúng ta.

Ví dụ : Có một bộ đếm tổng quát mô tả trên hình 4.2



Hình 4.2. Sơ đồ khối tổng quát của bộ đếm.

Bộ đếm luôn luôn chỉ có một đầu vào là (x), đầu vào (x) có thể có các giá trị :

$x = 0$ ứng với không cho gì vào

$x = 1$ ứng với cho vật đếm vào.

Ví dụ máy đếm tiền, nếu bộ đếm ta đã đếm được 5 tờ tiền, tức là tại thời điểm t_0 ta có trạng thái bên trong trước đó của bộ đếm là $S(t_0) = 5$.

Nếu tại thời điểm tiếp theo $t_1 > t_0$ ta cho thêm một tờ tiền nữa ($x = 1$) thì trạng thái tiếp theo của bộ đếm sẽ là 6 ($S(t_1) = 6$), đồng thời theo quan hệ toán học $Y = f(S, X)$ ta có :

$$S(t_1) = f(5,1) = 6 \text{ - cho thêm 1 vào}$$

Nếu : ($x = 0$) ta có :

$$S(t_1) = f(5,0) = 5 \text{ - không cho gì vào}$$

Qua đó ta thấy rất rõ trạng thái tiếp theo của bộ đếm ($S(t_1)$) sẽ phụ thuộc vào trạng thái trước đó ($S(t_0)$) và tín hiệu vào (X) là 0 hay là 1 đưa vào bộ đếm. Nói cách khác trạng thái tiếp theo (phản ứng) của ôtômat có nhớ sẽ phụ thuộc vào trạng thái trong trước đó (S) và tín hiệu vào (X) của nó. Nhưng mỗi một trạng thái của ôtômat lại có một đáo ứng ra của nó (tín hiệu ra của mạch). Như vậy ta cũng có thể nói : đáp ứng ra của ôtômat có nhớ cũng phụ thuộc vào trạng thái trước đó của ôtômat (S) và tín hiệu tác động vào nó.

Trong phương trình toán học của ôtômat có nhớ cho chúng ta hai thông tin. Đó là thông tin về trạng thái tiếp theo của ôtômat và thông tin về đáp ứng ra hay tín hiệu ra của ôtômat. Hai thông tin đó chúng cùng bị phụ thuộc đồng thời vào trạng thái bên trong trước đó của ôtômat (S) và tín hiệu tác động vào (X) của nó. Trong phương trình hoạt động tổng quát ta chưa phân biệt và tách riêng hai thông tin đó. Muốn hiểu rõ sự tách bạch của riêng từng thông tin đó trong hoạt động của ôtômat, ta có thể viết chúng riêng thành hai quan hệ toán học như sau :

$$Z = f\left(S(t_0),\ X\right)$$

$$S(t_1) = f\left(S(t_0),\ X\right)$$

Trong đó : Z : Chính là đáp ứng ra của mạch.

$S(t_1)$: Là trạng thái tiếp theo của ôtômat.

$S(t_0)$: Là trạng thái trong trước đó.

X : Tín hiệu tác động vào.

Đến đây chúng ta có mô hình toán học tổng quát của ôtômat có nhớ.

§4.2. CÁC PHƯƠNG PHÁP MÔ TẢ CƠ BẢN CỦA ÔTÔMAT CÓ NHỚ

Các mô hình toán học cũng chính là cách mô tả hoạt động của ôtômat có nhớ, hay ta cũng có thể gọi đó là "hàm số" của nó. Ta đã có các cách biểu diễn hàm số, đó là :

biểu diễn ở dạng f(X) quen thuộc, sau đó hàm số được biểu diễn dưới dạng bảng thật, bảng chân lý của hàm lôgic, tiếp theo là một cách biểu diễn "hàm số' kiểu khác nữa, đó chính là bảng chức năng của ôtômat có nhớ sẽ nêu ra trong phần này.

4.2.1 DẠNG TỔNG QUÁT CỦA ÔTÔMAT CÓ NHỚ

Định nghĩa 4.1. Bất kỳ một ôtômat có nhớ nào cũng có thể được biểu diễn dưới dạng tổng quát sau :

$$M = (X, S, Z, f, g)$$

Trong đó : M : Là một ký hiệu tượng trưng cho ôtômat M nào đó.

X : Là tập không trống hữu hạn các tín hiệu vào hay các biến vào :

$$X = (x_1, x_2, ..., x_n)$$

S : Là một tập không trống hữu hạn các trạng thái trong của ôtômat có nhớ :

$$S = (s_1, s_2, ..., s_p)$$

Z : Là một tập không trống hữu hạn các đáp ứng ra, nó chỉ lấy hai giá trị là 0 và 1 :

$$Z = (z_1, z_2, ..., z_m)$$

f : Là hàm chuyển biến trạng thái (là một ký hiệu tọa độ) cho biết sự phụ thuộc của trạng thái tiếp theo với trạng thái trước đó và tín hiệu vào (biến vào) :

$$s\ (t + 1) = f\ (s\ (t),\ x)$$

g : Là hàm ra (ký hiệu tọa độ) cho biết đáp ứng ra của mạch theo trạng thái trước đó và tín hiệu vào. Lấy hai giá trị 0 và 1 :

$$z\ (t) = g\ (s\ (t),\ x)$$

Ở đây ta ký hiệu s(t) là trạng thái trước đó, s(t+ 1) là trạng thái tiếp theo, x là tín hiệu vào và z (t) là đáp ứng ra của mạch điện.

Ở đây khi ta dùng hai hàm chuyển biến trạng thái f và hàm ra g là ta đã thực hiện tách nhỏ thông tin xảy ra trong hệ thống hoạt động kế tiếp thành hai thông tin riêng rẽ nhỏ hơn, đó là trạng thái tiếp theo phụ thuộc vào trạng thái trước đó s (t) và tín hiệu tác động vào X như thế nào dựa vào hàm f. Đáp ứng ra của hệ thống dựa vào hàm g tuân theo quan hệ tổng quát Y = f (S, X) cụ thể là :

$$s\ (t + 1) = f\ (s\ (t),\ x)$$

$$z\ (t) = g\ (s\ (t),\ x)$$

Các quan hệ của hàm chuyển biến trạng thái f và hàm ra g của ôtômat có nhớ là rất quan trọng, vì chúng cho ta biết hành vi hay ứng xử của hệ thống số cụ thể ra sao cùng với các thông tin khác như (X) tập tín hiệu vào, (S) tập trạng thái trong và (Z) tập các đáp ứng ra của ôtômat.

4.2.2. ÔTÔMÁT MEALY (BẢNG CHỨC NĂNG M_1)

Mô hình Mealy hay ôtômat Mealy là cách mô tả cụ thể hoạt động của một ôtômat có nhớ tại mọi thời điểm, nó cho phép ta hiểu chính xác hoạt động của hệ thống số theo từng tín hiệu tác động vào ôtômat Mealy hay còn được gọi là bảng Mealy. Đây là một cách biểu diễn hàm số kiểu khác, nó còn được coi là một bảng chức năng mô tả hoạt động cụ thể của ôtômat có nhớ. Ôtômát Mealy được biểu diễn tổng quát như sau :

$$M_1 = (X_1, S_1, Z_1, f_1, g_1)$$

Trong đó :

$$s(t+1) = f_1(s(t), x)$$

$$z(t) = g_1(s(t), x)$$

Ở đây hàm chuyển biến trạng thái f_1 và hàm ra g_1 phụ thuộc vào trạng thái trong s(t) và tín hiệu vào x ứng với thời điểm tác động của tín hiệu vào.

Bảng trạng thái M_1 hay ôtômat Mealy mô tả hoạt động cụ thể ôtômat có nhớ được mô tả trên hình 4.3.

Bảng M_1 :

S_1 \ X_1	$s(t+1)$				Z_1		
	x_1	x_2	...	x_n	x_1	...	x_n
s_1	$f_1(s_1, x_1)$	$f_1(s_1, x_2)$	.....	$f_1(s_1, x_n)$	$g_1(s_1, x_1)$	.....	$g_1(s_1, x_n)$
s_2	$f_1(s_2, x_1)$	.....	.....	$f_1(s_2, x_n)$	$g_1(s_2, x_1)$	.....	.....
.	.....	.....	.....	.....	.....	(0, 1)	.....
.	.....	.....	.....	.....	.....		.....
s_p	$f_1(s_p, x_1)$	.....	.....	$f_1(s_p, x_n)$	$g_1(s_p, x_1)$	.....	$g_1(s_p, x_n)$

Khu vực toàn bộ tất cả các trạng thái của ôtômat

Khu vực trạng thái trước đó

Khu vực trạng thái tiếp theo hay trạng thái mới được sinh ra. Khu hàm f_1.

Khu vực đáp ứng ra. Khu vực hàm g_1.

Hình 4.3. Bảng chức năng tổng quát của ôtômat có nhớ.

Nhìn vào bảng chức năng M_1 ta thấy : (X) là các tín hiệu vào hay tập biến vào. S là tập các trạng thái của ôtômat, đó cũng chính là cột biểu diễn cụ thể các trạng thái trước đó của nó từ ($s_1 \div s_p$), trong đó p là toàn bộ các trạng thái của ôtômat, đồng thời ta cũng có được toàn bộ các trạng thái của ôtômat có nhớ trong quá trình hoạt động của nó. Khu vực hàm f_1 mô tả các trạng thái tiếp theo mà được sinh ra phụ thuộc theo trạng thái trước đó và tín hiệu vào. Khu vực hàm ra g_1 cũng mô tả đáp ứng ra Z phụ thuộc vào trạng thái trước đó và tín hiệu vào thế nào, ở đây nó chỉ lấy hai giá trị 0 và 1.

Trạng thái trước đó S luôn mô tả toàn bộ số lượng trạng thái có thể có của ôtômat (không được thiếu trạng thái nào), cho nên ở khu vực trạng thái tiếp theo (khu vực hàm f) không được có trạng thái ngoài tập S trạng thái ban đầu. Nói cách khác là : *trạng thái tiếp theo hay ký hiệu mới được sinh ra không được vượt quá các trạng thái trước đó của ôtômat.* Đây là một điều rất quan trọng cần phải biết khi thiết kế một ôtômat số có nhớ, vì nếu có trạng thái mới được tạo ra mà vượt quá các trạng thái các ký hiệu ban đầu đã cho của ôtômat, thì chính là đã thiết kế sai ngay từ đầu.

* *Nguyên lý làm việc của bảng M_1 như sau* :

– Nếu trạng thái trước đó là s_1 dưới tác động của tín hiệu vào là x_1 thì trạng thái tiếp theo sẽ là $\boxed{f_1(s_1, x_1)}$.

– Nếu trạng thái trước đó là s_2 dưới tác động của tín hiệu vào x_2 thì trạng thái tiếp theo sẽ là f_1 (s_2, x_2).

– Nếu trạng thái trước đó là s_p dưới tác động của tín hiệu vào là x_n thì trạng thái tiếp theo sẽ là $f_1(\triangle s_p, x_n)$.

Đáp ứng ra :

– Nếu trạng thái trước đó là s_1 dưới tác động của tín hiệu vào là x_1 thì đáp ứng ra của mạch sẽ là $\boxed{g_1(s_1, x_1)}$. Như vậy mỗi một trạng thái mới (trạng thái tiếp theo) sẽ có một đáp ứng ra tương ứng với nó.

– Nếu trạng thái đó là s_p dưới tác động của tín hiệu vào là x_n thì đáp ứng ra của mạch sẽ là $g_1(\triangle s_p, x_n)$.

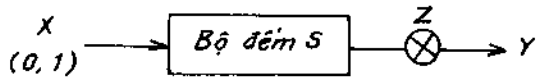
Như vậy hoạt động của bảng M_1 cho phép ta tách riêng làm hai khu vực, đó là khu trạng thái tiếp theo (hàm f_1) và khu đáp ứng ra của mạch (hàm g_1). Trong bảng chức năng đó đã mô tả tất cả các trạng thái tiếp theo có thể sinh ra, và toàn bộ các đáp ứng ra của ôtômat ứng với từng trạng thái riêng ở từng thời điểm. Điều đó cho phép ta dễ dàng hiểu được hoạt động của ôtômat có nhớ và dễ dàng nghiên cứu từng hoạt động chuyển đổi trạng thái của nó.

Một điều quan trọng nữa khi biểu diễn được hoạt động cụ thể của ôtômat có nhớ trên bảng Mealy (M_1) là : trạng thái tiếp theo của ôtômat có nhớ sẽ là (trở thành) trạng thái trước đó của trạng thái tiếp theo của nó. *Điều đó cho phép ta hiểu được hoạt động của hệ thống số có nhớ tại mọi thời điểm,* điều này rất cần và rất quan trọng khi lắp ráp thiết kế cụ thể ôtômat có nhớ cũng như sửa chữa hệ thống khi bị hỏng hóc sau này. Đồng thời chúng ta cũng nhận ra là sự hoạt động theo hàm f và hàm g của ôtômat căn cứ theo từng tọa độ trong bảng M_1 (bảng chức năng) và mỗi một tọa độ trong bảng hoạt động hoàn toàn độc lập với nhau.

Để hiểu hơn nữa công dụng của bảng Mealy dùng để mô tả cụ thể các hoạt động của ôtômat có nhớ, ta khảo sát ví dụ sau.

Ví dụ : Dùng ôtômat Mealy (bảng M_1) mô tả hoạt động của bộ đếm 5 – có 5 trạng thái, mà cứ đếm đến 3 thì báo (cho tín hiệu đèn sáng ở đầu ra Z).

Sơ đồ khối của bộ đếm và bảng chức năng được biểu diễn trên hình 4.4.



a)

X / S	s (t + 1)		Z	
	0	1	0	1
0	0	1	0	0
1	1	2	0	0
2	2	3	0	1
3	3	4	1	0
4	4	0	0	0

b)

Hình 4.4. Bảng chức năng M_1 của bộ đếm 5 mà 3 thì báo.

Hoạt động của bộ đếm được mô tả cụ thể trên bảng M_1 hoàn toàn tuân theo phương trình toán học tổng quát :

$$Y = f(S, X)$$

Kết quả cụ thể ở từng tọa độ như sau.

Vì là độ đếm nên chỉ có một đầu vào X và lấy hai giá trị 0 và 1.

Giả sử nếu trạng thái trước đó S = 3, thì khi cho X = 1 ta có :

$$s(t+1) = f(3, 1) = 4$$

$$z(t) = g(3, 1) = 0 \text{ không báo sáng}$$

Khi cho X = 0 ta có :

$$s(t+1) = f(3, 0) = 3$$

$$z(t) = g(3, 0) = 1 \text{ đầu ra Z đèn sẽ sáng (sẽ báo).}$$

Tiến hành tương tự ta sẽ khảo sát được hoạt động của bộ đếm 5 ở tất cả các khả năng cũng như ở tất cả các thời điểm hoạt động của nó. Nhìn vào bảng chức năng ta cũng thấy vì là bộ đếm nên nếu không cho gì vào (X = 0), thì trạng thái tiếp theo (khu hàm f) sẽ giống hệt trạng thái trước đó ở toàn bộ các khả năng, còn nếu cho giá trị đếm vào (X = 1) thì trạng thái tiếp sẽ lớn hơn trạng thái trước đó một đơn vị.

Do yêu cầu cứ đếm đến 3 thì báo (Z = 1), ta thấy khu đáp ứng ra Z (khu hàm g) có giá trị 1, tương ứng với tọa độ của trạng thái tiếp theo là 3, còn các trạng thái khác thì không báo gì cả nên giá trị của Z = 0. Nói cách khác : ôtômat Mealy (bảng M_1) dùng trạng thái tiếp theo để điều hành (cho Z = 1, đèn sáng hoặc làm một thao tác kỹ thuật nào đó). Đồng thời ta nhận thấy là khu vực hàm f ứng với các trạng thái mới được sinh ra (trạng thái tiếp theo), đã không xuất hiện các trạng thái lạ hay không có ký hiệu mới được xuất hiện mà chỉ lặp đi lặp lại các trạng thái (ký hiệu) trước đó ở S mà thôi.

Ví dụ trên cho biểu diễn hay cách mô tả hoạt động cụ thể một ôtômat có nhớ hay một hệ thống số hoạt động kế tiếp, khi đã mô tả được thành bảng chức năng (chính là

một dạng "hàm số" toán học) thì từ đó ta có thể chuyển thành toán lôgic để lắp ráp thực hiện được chức năng đó. Các phần sau sẽ trình bày đến kết quả cuối cùng là lắp ráp thành mạch cụ thể thực hiện chức năng hay nhiệm vụ đã đề ra.

4.2.3. ÔTÔMÁT MOORE (BẢNG CHỨC NĂNG M_2)

Để mô tả hoạt động của một ôtômat có nhớ, ta có thể biểu diễn bằng bảng chức năng M_2 hay còn gọi là ôtômat Moore.

Ôtômát Moore có quan hệ tổng quát :

$$M_2 = (X_2, S_2, Z_2, g_2)$$

Trong đó :

$$s(t+1) = f_2(s(t), x)$$

$$z(t) = g_2(s(t))$$

So sánh giữa ôtômat Mealy (M_1) và ôtômat. Moore (M_2) ta thấy quan hệ của trạng thái tiếp theo s (t + 1) (theo khu vực hàm f) của chúng hoàn toàn giống nhau. Như vậy bảng chức năng của ôtômat Moore sẽ giống hệt ở ôtômat Mealy ở khu vực hàm f. Bảng M_1 khác bảng M_2 ở đáp ứng ra (Z) hay ở khu vực hàm ra (g).

Bảng trạng thái M_2 hay ôtômat Moore được mô tả trên hình 4.5.

Bảng M_2 :

X_2 / S_2	s (t + 1)				Z_2
	x_1	x_2		x_n	
s_1	$f_2(s_1, x_1)$	$f_2(s_1, x_2)$	. . . . .	$f_2(s_1, x_n)$	$g_2(s_1)$
s_2	$f_2(s_2, x_1)$	. . . . .	. . . . .	. . . . .	$g_2(s_2)$
.	. . . . .	. . . . .	. . . . .	. . . . .	.
.	. . . . .	. . . . .	. . . . .	. . . . .	.
.	. . . . .	. . . . .	. . . . .	. . . . .	.
s_p	$f_2(s_p, x_1)$	. . . . .	. . . . .	$f_2(s_p, x_n)$	$g_2(s_p)$

Hình 4.5. Bảng chức năng ôtômat Moore.

Nhìn vào bảng M_2 ta thấy hàm ra (g_2) hay đáp ứng ra (Z_2) không phụ thuộc trực tiếp vào tín hiệu vào (X_2).

Chức năng của bảng (Moore) trong khu vực hàm f giống như ở bảng M_1 (Mealy), đồng thời các tính chất của nó cũng như các yêu cầu của bảng M_2 cũng giống như bảng M_1, cụ thể là : trạng thái tiếp theo tương ứng với từng tọa độ ở trong bảng, và trạng thái mới được sinh ra (trạng thái tiếp theo) không được vượt quá các trạng thái trước đó của ôtômat.

Để hiểu được hoạt động của ôtômat Moore (bảng M_2) ta có thể khảo sát ví dụ minh họa sau.

Ví dụ : Dùng ôtômat Moore mô tả hoạt động của bộ đếm 5, mà cứ đếm tới 3 thì báo.

Bảng chức năng của bộ đếm được biểu diễn trên hình 4.6 (chú ý là sau này khi đã quen ta cũng không cần viết ở khu trạng thái tiếp theo có s (t + 1) nữa).

Bảng M_2 :

S \ X	s (t + 1)		Z
	0	1	
0	0	1	0
1	1	2	0
2	2	3	0
3	3	4	1
4	4	0	0

Hình 4.6. Bảng chức năng bộ đếm 5 mà đến 3 thì báo.

Chức năng hoạt động của bảng Moore (M_2) cũng giống như chức năng bộ đếm 5 ở trên là : Giả sử trạng thái đã có S = 3.

Nếu cho X = 1 :

$$s(t + 1) = f(3, 1) = 4$$

$$z(t + 1) = g(3) = 1$$

Nếu cho X = 0 :

$$s(t + 1) = f(3, 0) = 3$$

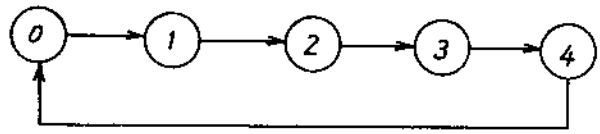
$$z(t) = g(3) = 1$$

Qua bảng M_2 của bộ đếm 5 ta thấy khu vực hàm f (khu trạng thái tiếp theo) không có gì thay đổi, giống khu vực hàm (f) của bảng M_1 đã được trình bầy ở phần trên.

Ta thấy hàm (g) là bảng M_2 khác với bảng M_1, vì khu vực hàm g (đáp ứng ra) không phụ thuộc trực tiếp vào tín hiệu vào (X) ((X) không thấy có giá trị ở cột Z), giá trị lôgic của hàm (g) phụ thuộc trực tiếp vào trạng thái trước đó của bộ đếm cụ thể là dùng trạng thái 3 ở khu vực trạng thái trước đó điều khiển cho Z = 1 dùng để báo sáng. Nói cách khác : ôtômat Moore (M_2) dùng trạng thái trước đó để điều khiển (khi Z = 1).

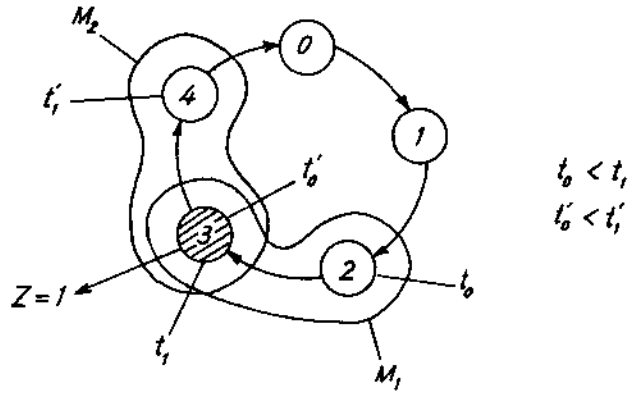
Như vậy vẫn là bộ đếm 5 mà cứ 3 thì báo, ta có được hai phương pháp biểu diễn đó là bảng M_1 (Mealy) và bảng M_2 (Moore), hai cách biểu diễn đó đồng khả năng và tương đương với nhau khi thiết kế. Hai phương pháp tương đương nhau, có thể được giải thích như sau :

Vì ôtômat có nhớ cũng như bộ đếm luôn hoạt động theo nguyên lý "vòng quanh", cụ thể ở đây bộ đếm 5 là một bộ đếm vòng, nó làm việc theo nguyên lý mô tả trên hình 4.7.



Hình 4.7. Bộ đếm vòng quanh.

Một hoạt động quay vòng thì việc phân biệt "quá khứ" và "tương lai" hay trước đó và tiếp theo là không xác định vì nó phụ thuộc điểm chọn làm gốc. Chính vì giữa Mealy và Moore có sự phân biệt về quan điểm "trước" và "sau" khác nhau về trạng thái trước và trạng thái tiếp theo. Điều đó có thể được minh họa hoạt động của bộ đếm 5 (bộ đếm vòng quanh) trên hình 4.8.



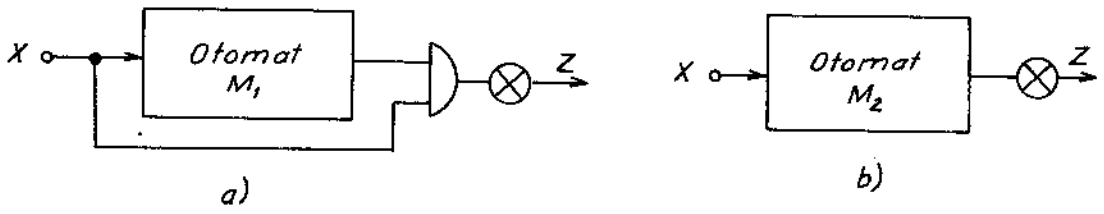
Hình 4.8. Sự giống nhau trong cùng một chức năng của M_1 và M_2.

Quan điểm của Mealy quá khứ (trạng thái trước đó) ứng thời điểm t_0 và tương lai (trạng thái tiếp theo) là thời điểm t_1 mà ($t_0 < t_1$). Trong khi đó với Moore lại có quan điểm quá khứ (trạng thái trước đó) là tại t'_0 còn tương lai (trạng thái tiếp theo) ở thời điểm t'_1 mà ($t'_0 < t'_1$). Nhưng nếu ôtômat có nhớ hoạt động không theo kiểu "vòng quanh", thì hai cách biểu diễn theo bảng chức năng M_1 của Mealy và theo bảng chức năng M_2 của Moore là không thống nhất và không tương đương với nhau nữa.

Đến đây ta có được hai cách (hai bảng chức năng M_1 và M_2) để mô tả chức năng hoạt động của ôtômat có nhớ. Đây là một việc làm hết sức quan trọng vì có mô tả được chức năng hoạt động của ôtômat có nhớ (theo toán học) thì mới có khả năng thiết kế và lắp ráp ra được hệ thống chức năng đó. Đồng thời các kết luận sau này mà chúng ta thu được.trong quá trình nghiên cứu về kỹ thuật số nói chung chỉ dựa trên nền tảng và bản chất hoạt động của hai bảng chức năng (M_1, M_2). Nhưng do bảng M_1 và bảng M_2 chỉ khác nhau ở khu vực hàm ra (Z) lại là tín hiệu điều khiển cụ thể một đối tượng bị điều khiển nào đó (như đèn, quạt, động cơ...). Nên sơ đồ khối mạch điện tổng quát của hai cách biểu diễn ôtômat có nhớ sẽ có dạng biểu diễn trên hình 4.9.

Nói một cách khác là : khi ta nhìn vào một mạch điện thực tế và xem cách điều khiển ở đầu ra của mạch điện (hệ thống số) đó, có thể biết ngay được người thiết kế ra mạch điện từ lúc khởi đầu họ dùng ôtômat Mealy hay ôtômat Moore để biểu diễn hoạt

động của mạch điện. Chú ý là : việc lắp ráp thiết kế dùng theo cách biểu diễn bảng M_1 hay dùng bảng M_2 là tùy ý, và theo ý thích của người thiết kế mà thôi, vì giữa hai cách biểu diễn ôtômat có nhớ theo Mealy và theo Moore không có cách biểu diễn nào "lợi" cách biểu diễn nào, và khi thiết kế cụ thể mạch điện thì độ "khó" của chúng là như nhau.



Hình 4.9. Mạch tổng quát của M_1 và M_2.

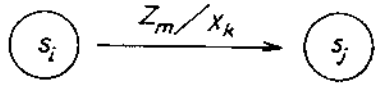
4.2.4. BIỂU DIỄN ÔTÔMAT CÓ NHỚ BẰNG ĐỒ HÌNH (GRAF)

Ta đã có hai cách mô tả hoạt động của ôtômat có nhớ bằng dùng bảng M_1 và bảng M_2. Dưới đây sẽ đề cập các cách biểu diễn (mô tả) ôtômat có nhớ dùng phương pháp đồ hình.

4.2.4.1. Đồ hình Mealy (M_1) - graf mealy

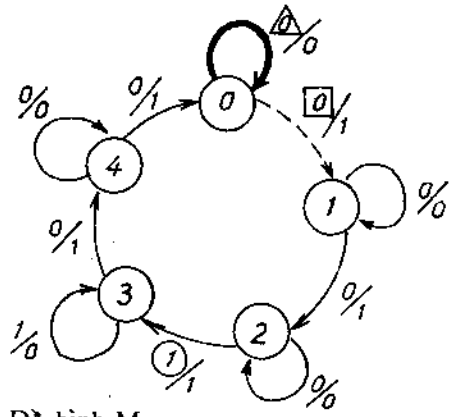
Định nghĩa 4.2 : Đồ hình Mealy bao gồm các nút và nhánh có hướng. Trong các nút có ghi trạng thái của ôtômat, còn nhánh có hướng chỉ hướng chuyển biến trạng thái của ôtômat, trên đó có ghi đáp ứng ra và tín hiệu vào tương ứng.

Trên hình mô tả đồ hình M_1



Hình 4.10. Đồ hình Mealy.

Ví dụ : Dùng đồ hình Mealy mô tả hoạt động của bộ đếm 5, cứ đếm đến 3 thì báo. Cách biểu diễn được mô tả trên hình 4.11.



b) Đồ hình M_1

S \ X	0	1	Z: 0	Z: 1
0	0	1	0	0
1	1	2	0	0
2	2	3	0	1
3	3	4	1	0
4	4	0	0	0

a) Bảng chức năng

Hình 4.11. Dùng đồ hình Mealy mô tả bộ đếm.

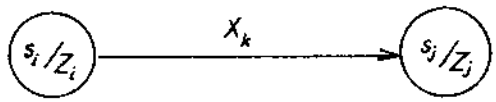
Cách vẽ đồ hình M_1 ứng với quá trình chuyển biến trạng thái từ 0 chuyển sang 0 (trạng thái trước đó là 0 trạng thái tiếp theo là 0) ứng với nét đậm (0 → 0), thì ta có đáp ứng ra Z = 0 , với tín hiệu vào X = 0 từ đó vẽ được hướng chuyển biến trạng thái. Với cách làm tương tự của quá trình chuyển biến trạng thái từ 0 chuyển sang 1 (0 → 1) nhìn vào bảng M_1 ta thấy đáp ứng ra Z = $\boxed{0}$ với tín hiệu vào X = 1, từ đó ta vẽ được hướng chuyển biến trạng thái đó vào đồ hình và ghi kết quả của đáp ứng ra và tín hiệu vào hướng chuyển biến trạng thái đó.

Với cách làm tương tự ta vẽ được đồ hình ứng với các quá trình chuyển biến trạng thái khác của ôtômat hay của bộ đếm 5 được đề ra ở trên. Dễ dàng nhận thấy rằng đồ hình chuyển biến trạng thái của ôtômat có nhớ và bảng chức năng hay bảng chuyển biến trạng thái của nó là tương ứng với nhau.

4.2.4.2. Đồ hình Moore (M_2) - graf Moore

Định nghĩa 4.3 : Đồ hình Moore bao gồm các nút và nhánh có hướng. Trong các nút có ghi trạng thái của ôtômat và đáp ứng ra tương ứng của trạng thái đó. Nhánh có hướng chỉ hướng chuyển biến trạng thái, trên đó có ghi tín hiệu tác động vào (hay biến vào).

Đồ hình M_2 được mô tả trên hình 4.12.

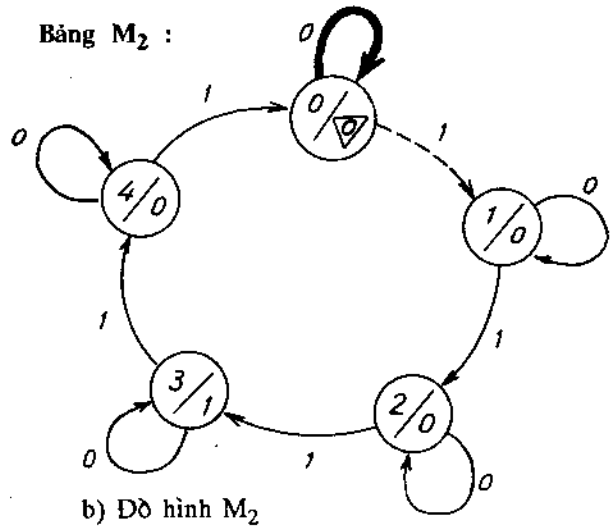


Hình 4.12. Đồ hình Moore.

Ví dụ : Dùng đồ hình Moore mô tả hoạt động của bộ đếm 5, mà cứ đếm tới 3 thì báo.

Cách biểu diễn được mô tả trên hình 4.14.

Bảng M_2 :



b) Đồ hình M_2

S \ X	0	1	Z
0	0	1	0
1	1	2	0
2	2	3	0
3	3	4	1
4	4	0	0

a) Bảng chức năng

Hình 4.13. Dùng đồ hình Moore mô tả bộ đếm.

Cách vẽ đồ hình M_2 : trong nút ta chỉ cần ghi trạng thái với đáp ứng ra với mỗi nút tương ứng với một trạng thái của ôtômat. Sau đó vẽ quá trình chuyển biến trạng thái. Với quá trình (0 → 0) có tín hiệu kích vào tương ứng là 0 (X = 0), còn tiếp theo với quá trình chuyển biến (0 → 1) thì có tín hiệu kích vào (có cái đếm cho vào) X = 1, ta ghi trên mũi tên chỉ hướng chuyển biến đó giá trị. Với cách vẽ như vậy ta dùng đồ hình Moore đã mô ta được toàn bộ hoạt động có nhớ của bộ đếm 5 cứ đến 3 thì báo được đưa ra.

Đến đây ta đã có bốn cách hay bốn phương pháp dùng để mô tả hoạt động của ôtômat có nhớ. Việc mô tả được hoạt động của ôtômat cho phép ta lắp ráp xây dựng mạch điện thực hiện ôtômat có nhớ. dùng bảng chức năng hay còn được gọi là cách mô tả hoạt động của ôtômat có nhớ, dựa vào đó có các bước tiếp theo để xây dựng nên mạch điện cụ thể, nhưng dùng đồ hình để mô tả hoạt động của ôtômat có nhớ thì chưa có phương pháp để làm ra mạch điện được. Nhưng ưu việt của phương pháp đồ hình là ta dễ dàng mô tả được hoạt động của ôtômat, đồng thời dễ trao đổi thảo luận về hoạt động của hệ thống số có số lượng trạng thái rất lớn. Chính vì vậy khi bắt đầu thiết kế một hệ thống số người ta thường vẽ ra hoạt động (mô tả bằng đồ hình) của nó trước, sau đó mới chuyển từ đồ hình thành bảng chức năng, và từ bảng chức năng mới đi các bước tiếp theo để có được mạch điện của hệ thống số. Đó cũng chính là các bước thiết kế cụ thể ôtômat có nhớ. Phương pháp "mô tả" trực tiếp hay dùng bảng chức năng (M_1, M_2) biểu diễn hoạt động của ôtômat có nhớ, chỉ hay được dùng với các ôtômat số có số lượng trạng thái ít.

Chúng ta đều đã biết một ôtômat có nhớ có thể được mô tả bằng bảng Mealy (M_1) hay bảng Moore (M_2), tức là một chức năng kỹ thuật, ta có thể biểu diễn bởi hai cách là dùng ôtômat Mealy (M_1) hay dùng ôtômat Moore (M_2), như vậy giữa hai ôtômat M_1 và ôtômat M_2 có sự liên hệ với nhau, từ đó xuất hiện vấn đề là chuyển đổi qua lại giữa hai cách biểu diễn ôtômat có nhớ.

4.2.5. CHUYỂN ĐỔI GIỮA ÔTÔMAT MEALY VÀ ÔTÔMAT MOORE

Việc biểu diễn hoạt động ôtômat có nhớ dùng bảng M_1 hay bảng M_2 có những tính chất khác nhau về nguyên lý mạch điện, hay nói một cách khác là giữa hai ôtômat Mealy và ôtômat Moore vẫn có các tính chất về mạch khác nhau. Vấn đề là chúng ta cần có phương pháp để chuyển đổi giữa chúng mà không cần phải mô tả lại từ đầu.

4.2.5.1. Chuyển đổi từ bảng Mealy sang bảng Moore (M_1 → M_2)

Số liệu ban đầu là ôtômat Mealy, nó có quan hệ :

$$M_1 = (X_1, S_1, Z_1, f_1, g_1)$$

Từ số liệu ban đầu của ôtômat Mealy có trước đó, ta phải tìm được các giá trị của ôtômat Moore tương ứng là :

$$M_2 = (X_2, S_2, Z_2, f_2, g_2)$$

Do quá trình chuyển đổi từ ($M_1 \rightarrow M_2$) là tương ứng một một ($1 \longleftrightarrow 1$) với cùng một bài toán hay cùng một chức năng điều khiển, cho nên chúng ta có thể viết :

$$X_1 = X_2$$

$$Z_1 = Z_2$$

Tức là dù có chuyển đổi thế nào thì cũng phải thỏa mãn là : tín hiệu vào như nhau sẽ cho đáp ứng ra giống nhau. Như vậy quá trình chuyển đổi ($M_1 \rightarrow M_2$) ta đã biết được X_2 và Z_2 theo số liệu ban đầu X_1 và Z_1 cho trước. Vấn đề còn lại là ta phải tìm (f_2, g_2) theo số liệu ban đầu là (f_1, g_1) cho trước của M_1 mà thôi.

Người ta đã chứng minh :

Nếu tồn tại một trạng thái $s_i \in M_1$, dưới tác động của tín hiệu vào X_k, sẽ sinh ra trạng thái tiếp theo và đáp ứng ra của M_1 như sau :

$$f_1 (s_i, X_k) = s_j - \textit{ứng tọa độ trong bảng } M_1$$

$$g_1 (s_i, X_k) = z_r$$

Thì sẽ tồn tại một trạng thái q_j của M_2 mà chúng có quan hệ :

$$s_j = q_j .$$

Trong đó : s_j là trạng thái tiếp theo của M_1 cho trước.

q_j là trạng thái trước đó của M_2 sẽ tương ứng phải tìm.

Nói cách khác là một trạng thái tiếp theo (s_i) của bảng M_1 cho trước (số liệu ban đầu) thì chắc chắn ta sẽ có được một trạng thái tương ứng q_j của bảng M_2 phải tìm, và cũng vì M_2 phải tìm nên luôn luôn có thể xác định được trạng thái của nó cần tìm dựa theo bảng M_1 vì đó là quá trình chuyển đổi ($M_1 \rightarrow M_2$).

Do q_j là trạng thái của bảng M_2 ($q_j \in M_2$) cho nên ta có quan hệ :

$$s (t + 1) = f_2 (q_j , X_k) = q_j^*$$

$$z (t) = g_2 (q_j) = z_j$$

Từ nhiệm vụ phải chuyển đổi ($M_1 \rightarrow M_2$) nên việc chuyển biến trạng thái (trạng thái chuyển biến tiếp theo) của hai ôtômat M_1 và M_2 phải tương ứng với nhau (do khu vực hàm f của M_1 và M_2 là giống nhau), đồng thời đáp ứng ra của chúng cũng phải là như nhau, từ đó ta có quan hệ :

$$q_j^* = f_2 (q_j , X_k) = f_1 (s_j , X_k)$$

$$z_j = g_2 (q_j) = g_2 (s_i , X_k) = z_r$$

Từ các quan hệ ở trên ta cũng có thể hiểu là trạng thái tiếp theo của bảng M_1 sẽ tương ứng với trạng thái trước đó của M_2, đây là quan hệ ($1 \longleftrightarrow 1$) do ta chủ động lựa chọn. Vấn đề còn lại là trạng thái tiếp theo của M_2 có giá trị phải tương ứng với hoạt động của M_1, vì khi ta chuyển đổi ($M_1 \rightarrow M_2$) thì chức năng của chúng là không đổi, không được khác nhau.

Xét ví dụ : Giả sử có một ôtômat, hoạt động của nó được biểu diễn ở bảng Mealy (M_1) như sau :

Bảng M_1 (số liệu ban đầu cho trước) ở hình 4.14. Ta phải chuyển thành bảng M_2 tương ứng.

S_1 \ X_1	x_1	x_2	Z_1 x_1	Z_1 x_2
s_1	$s_2\ (q_1)$	$s_4\ (q_4)$	z_2	z_1
s_2	$s_3\ (q_2)$	–	–	–
s_3	$s_2\ (q_3)$	$s_1\ (q_5)$	z_1	–
s_4	–	$s_2\ (q_6)$	–	z_2

Hình 4.14. Cho một ôtômat theo M_1.

Trong bảng ta thấy có các vị trí ký hiệu "–" đó là các trạng thái không xác định của ôtômat đã cho.

Từ bảng M_1 đã cho ta có :

$$X_1 = (x_1, x_2)$$
$$Z_1 = (z_1, z_2)$$
$$S_1 = (s_1, s_2, s_3, s_4)$$

Từ nhiệm vụ chuyển biến ($M_1 \rightarrow M_2$) chúng phải có chức năng như nhau cho nên ta có :

$$X_2 = X_1 = (x_1, x_2)$$
$$Z_2 = Z_1$$

Đảm bảo thỏa mãn quan hệ tín hiệu vào như nhau ($X_2 = X_1$), thì sẽ có đáp ứng ra giống nhau ($Z_2 = Z_1$). Vấn đề là phải tìm hàm chuyển biến trạng thái $f_2 \in M_2$ chuyển biến như thế nào (căn cứ theo bảng M_1) và đáp ứng ra hàm ra $g_2 \in M_2$ có giá trị thế nào từ bảng M_1.

Do ta đã chọn trạng thái tiếp theo của M_1 tương ứng với trạng thái trước đó của M_2 ($s_j = q_j$) ta sẽ có cấu tạo của bảng M_2 như trên hình 4.15.

Bảng M_2 (bảng phải tìm) :

S_2 \ X_2	x_1	x_2	Z_2
q_1	q_2	△	z_2
q_2	q_3	q_5	–
q_3	q_2	–	z_1
q_4	–	q_6	z_1
q_5	q_1	q_4	–
q_6	q_2	–	z_2

Hình 4.15. Kết quả chuyển thành bảng M_2.

Khi ta có các trạng thái của bảng M_2 là S_2 được ký hiệu là (q_j) tương ứng với các trạng thái tiếp theo của bảng M_1 (đã cho trước) theo quan hệ ký hiệu tương ứng như viết trên bảng M_1 ở hình 4.15.

Cách điền bảng M_2 tương ứng theo hoạt động (cho trước) của M_1, tức là điền trạng thái tiếp theo (hoạt động của M_2) theo bảng M_1 cho trước. Ta sẽ điền bảng M_2 theo từng tọa độ.

Dựa vào bảng M_1 ta có tọa độ đầu $(s_1, x_1) \in M_1$:

$$f_1(s_1, x_1) = s_2 = q_1 \qquad (q_1 \in M_2)$$

tương ứng ở bảng M_2 ta có :

$$f_2(q_1, x_1) = f_1(s_2, x_1) = s_3 \text{ (theo bảng } M_1)$$

mà : $s_3 = q_2$ theo ký hiệu đặt ra.

Tóm lại có kết quả :

$$f_2(q_1, x_1) = (q_2)$$

Từ kết quả thu được ta điền trạng thái (q_2) vào bảng M_2 theo tọa độ của nó. Như vậy dựa vào hoạt động của bảng M_1 mà ta đã có trạng thái tương ứng ở M_2.

Đáp ứng ra tương ứng M_2 theo M_1 được xác định ở tọa độ (s_1, x_1) là :

$$g_1(s_1, x_1) = g_2(q_1) = (z_2) \rightarrow \text{điền } (z_2) \text{ vào bảng } M_2.$$

Với cách làm tương tự với tọa độ khác $(s_1, x_2) \in M_1$:

$$f_1(s_1, x_2) = s_4 = q_4 \qquad \text{(ký hiệu)}$$

Vậy ta có :

$$f_2(q_4, x_2) = f_1(s_4, x_2) = s_2 = q_6 \text{ (ký hiệu)}$$

$$f_2(q_4, x_2) = q_6$$

Ta điền giá trị q_6 vào bảng M_2 theo tọa độ của nó ở trong bảng.

Đáp ứng ra của M_2 tính theo M_1 được xác định theo tọa độ (s_1, x_2) là :

$$g_1(s_1, x_2) = g_2(q_4) = \boxed{z_1} \rightarrow \text{điền nó vào } M_2$$

Tiếp tục với tọa độ khác như $(s_2, x_2) \in M_1$:

$$f_2(q_1, x_2) = f_1(s_2, x_2) = \text{"–" (không xác định)}$$

$$f_2(q_1, x_2) = \triangle$$

Ta điền dấu không xác định $\triangle$ vào bảng M_2 với tọa độ tương ứng.

Tiếp theo ở tọa độ mới $(s_2, x_1) \in M_1$:

$$f_1(s_2, x_1) = s_3 = q_2$$

$$f_2(q_2, x_1) = f_1(s_3, x_1) = s_2 = q_3$$

$$f_2(q_2, x_1) = q_3$$

Điền q_3 vào bảng M_2 theo tọa độ của nó

$$g_2(q_3) = g_1(s_2, x_1) = z_1 \rightarrow \text{điền vào } M_2$$

Với cách làm tương tự ta sẽ có tọa độ tương ứng ở M_2. Như vậy đã thực hiện quá trình chuyển đổi từ ôtômat Mealy thành ôtômat Moore ($M_1 \rightarrow M_2$) với chức năng của chúng là hoàn toàn giống nhau.

4.2.5.2. Chuyển đổi từ ôtômat Moore sang ôtômat Mealy ($M_2 \rightarrow M_1$)

Ta có nhận xét ban đầu là : ôtômat Moore giá trị hàm ra (Z) chỉ phụ thuộc vào trạng thái trước đó của nó :

$$z(t) = g(s(t))$$

Đồng thời số lượng trạng thái của ôtômat Moore luôn luôn lớn hơn hoặc bằng số lượng trạng thái của ôtômat Mealy ($S_2 \geq S_1$) tương ứng. Bài toán đặt ra là :

Cho trước ôtômat Moore : $M_2 = (X_2, S_2, Z_2, f_2, g_2)$

Cần phải tìm ôtômat Mealy tương ứng cùng chức năng:

$$M_1 = (X_1, S_1, Z_1, f_1, g_1).$$

Do cùng là một chức năng (cùng một bài toán) cần tìm tương ứng cho nên có :

$$X_1 = X_2$$

$$Z_1 = Z_2$$

Tín hiệu vào như nhau ($X_1 = X_2$) thì có đáp ứng ra phải giống nhau ($Z_1 = Z_2$).

Từ các nhận xét trên và khu vực hàm (f) của bảng M_1 và bảng M_2 giống nhau cho nên ta có thể chọn :

$$S_1 = S_2 \qquad \text{và cho } f_1 = f_2 = f$$

Vấn đề còn lại ở đây là phải tìm hàm ra g_1 tương ứng của bảng M_1 theo các số liệu ban đầu cho trước ở bảng M_2. Nếu ta lấy quan hệ trạng thái :

$$s_i = s_i^*$$

Trong đó : $s_i^* \in M_2$ cho trước

$s_i \in M_1$ phải tìm.

Thì ta sẽ có quan hệ :

$f_2(s_i^*, X_k) = s_j$ nào đó của M_2 cho trước

$s_j = f_1(s_i, X_k)$ của M_1 (phải tìm)

Hay hiểu cách khác do khu vực hàm f của bảng M_1 và bảng M_2 là giống nhau cho nên :

$$f_2(s_i^*, X_k) = f_1(s_i, X_k) = s_j.$$

Từ đó ta có quan hệ của hàm ra (g_1) trong bảng M_1 theo số liệu ban đầu từ bảng M_2 như sau :

$$g_1 (s_i, X_k) = g_2 (s_i^*) .$$

Khảo sát ví dụ về hoạt động của một ôtômat được minh họa trên bảng Moore (M_2) như ở hình 4.16.

Bảng M_2 :

S_2 \ X_2	x_1	x_2	Z_2
s_1^*	s_2	–	–
s_2^*	s_3	s_2	z_1
s_3^*	s_3	s_2	z_2

Hình 4.16. Cho một ôtômat theo bảng M_2.

Tìm ôtômat Mealy tương ứng.

Từ bảng M_2 đã cho trước ta cần phải chuyển nó thành bảng M_1 tương ứng :

$$X_1 = X_2 = (x_1, x_2)$$

$$Z_1 = Z_2 = (z_1, z_2)$$

$$S_1 = S_2 = (s_1^*, s_2^*, s_3^*)$$

$f_1 = f_2$ (khu vực hàm f của chúng giống nhau)

Vấn đề là tìm ra hàm (g_1) của bảng M_1 hay khu vực đáp ứng ra (Z_1 của bảng M_1.

Bảng M_1 phải tìm sẽ có cấu tạo là : Khu vực hàm chuyển biến trạng thái (f_1) sẽ giống hệt khu vực chuyển biến trạng thái của bảng M_2 do ($f_1 = f_2$).

Cấu tạo bảng M_1 ở trên hình 4.17.

Bảng M_1 :

S_1 \ X_1			Z_1	
	x_1	x_2	x_1	x_2
s_1	s_2	–	z_1	–
s_2	s_3	s_2	z_2	z_1
s_3	s_3	s_2	z_2	z_1

Hình 4.17. Bảng M_1 cần tìm.

Khu vực (S_1) và (f_1) của bảng M_1 phải tìm giống bảng M_2 vì có quan hệ :

$$S_1 = S_2$$

$$f_1 = f_2$$

Chỉ còn phải điền nốt khu vực (Z_1) của bảng M_1 theo bảng M_2 đã cho ở trên. Ta cũng tính theo từng tọa độ.

Từ bảng M_2 xét tọa độ (s_1, x_1) ta có :

$f_2(s_1^*, x_1) = s_2 \in$ bảng M_2, vậy cũng chính là s_2^*. Tương ứng tọa độ đó ở bảng M_1 ta có :

$$f_1(s_i, x_1) = s_2 \in \text{bảng } M_1.$$

Mà đáp ứng ra của bảng M_2 phải giống như đáp ứng ra của bảng M_1, vì $(Z_1 = Z_2)$ từ đó ta có quan hệ :

$$g_1(s_1, x_1) = g_2(s_2^*) = z_1 \text{ (từ bảng } M_2)$$

$g_1(s_1, x_1) = \textcircled{z_1} \rightarrow$ điều giá trị $\textcircled{z_1}$ vào bảng M_1.

Tiếp theo ta có tọa độ khác ở bảng M_2 - (s_2^*, x_1) :

$f_2(s_2^*, x) = s_3 \in M_2$, vậy đó cũng chính là s_3^* :

Đáp ứng ra tương ứng sẽ là : $g_2(s_3^*) = z_2$

Tương ứng với tọa độ đó ở M_1 ta có :

$f_1(s_2, x_2) = s_3 \rightarrow$ có đáp ứng ra của bảng M_1 giống như ssáp ứng ra của bảng M_2 vậy ta có :

$$g_1(s_2, x_2) = g_2(s_3^*) = z_2$$

$$g_1(s_2, x_2) = z_2$$

Ta điền z_2 vào bảng M_1 ứng với tọa độ (s_2, x_1) của nó ở vị trí đáp ứng ra.

Tiếp tục với các tọa độ khác $(s_1^*, x_2) \in M_2$:

$$f_2(s_1^*, x_2) = \text{"}-\text{"} \text{ không xác định trạng thái}$$

Vậy :

$$g_1(s_1, x_2) = g_2(-) = \text{"}-\text{"} \text{ không xác định}$$

$$g_1(s_1, x_2) = \text{"}-\text{"}$$

Ta điền vào tọa độ (s_1, x_2) ở khu vực đáp ứng ra (Z_1) của bảng M_1 dấu không xác định $(-)$.

Với cách làm tương tự như vậy đối với các tọa độ khác của bảng M_2, ta sẽ có được đáp ứng ra của bảng M_1 tương ứng với đáp ứng ra của bảng M_2.

Như vậy chúng ta đã thực hiện được việc chuyển đổi mạch điện : từ mạch điện đáp ứng ra (Z) phụ thuộc trực tiếp tín hiệu vào (tín hiệu vào nối đến tận đầu ra) đó là ôtômat Mealy thành mạch điện đáp ứng ra (Z) (chỉ phụ thuộc gián tiếp vào tín hiệu vào đó là ôtômat Moore. Đồng thời cũng có thể chuyển đổi ngược lại mạch điện lắp ráp kiểu Moore (M_2) thành mạch điện lắp ráp kiểu Mealy (M_1). Quá trình chuyển đổi đó rất linh hoạt, đối với các kỹ sư phần cứng khi muốn thay đổi kết cấu mạch chuyển đổi ôtômat để lựa chọn các mạch điện theo yêu cầu kỹ thuật cần thiết.

Trước khi đề cập quá trình thiết kế lắp ráp ôtômat có nhớ ta cần nên thêm một dạng biểu diễn phụ khác cho cách biểu diễn của bảng Mealy (M_1) được mô tả trên hình 4.18.

Ví dụ : Dùng ôtômat Mealy mô tả bộ đếm 5 mà cứ đếm đến 3 thì báo sau.

S \ X	0	1	Z 0	Z 1
0	0	1	0	0
1	1	2	0	0
2	2	3	0	1
3	3	4	1	0
4	4	0	0	0

S \ X	0	1
0	0/0	1/0
1	1/0	2/0
2	2/0	3/1
3	3/1	4/0
4	4/0	0/0

Hình 4.18. Dạng khác của ôtômat Mealy.

Hai bảng chức năng đó cùng biểu diễn bộ đếm 5 mà cứ 3 thì báo theo Mealy (M_1), hai bảng chức năng đó hoàn toàn như nhau và tương đương nhau, bảng ở hình 4.18a thì tách ra thành hai khu vực riêng rẽ là : Khu trạng thái tiếp theo và khu đáp ứng ra, điều này cho phép nghiên cứu xem xét riêng hoạt động của ôtômat được dễ dàng. Còn trên hình 4.18b thì người ta gộp luôn trạng thái ôtômat kèm theo đáp ứng ra.

Bộ đếm chính là một ôtômat có nhớ rất điển hình. Đồng thời bộ đếm nói chung cũng như bộ đếm vòng được ứng dụng rất nhiều trong thực tế, khi cần chỉ thị số các kết quả (như đồng hồ số, cần chỉ thị số...), đồng thời bộ đếm cũng tham gia trực tiếp tới quá trình tạo các tín hiệu điều khiển trong các hệ thống số các ôtômat số nói chung, hay cụ thể hơn là dùng để đọc địa chỉ các ô nhớ trong máy tính, những ứng dụng đó của bộ đếm sẽ được lần lượt trình bầy cụ thể trong các phần sau.

Muốn thực hiện hay thiết kế một ôtômat có nhớ với chức năng cụ thể của nó chứ không phải là thiết kế ra bộ đếm, vì bộ đếm chỉ là trường hợp riêng của ôtômat có nhớ mà thôi. Vì thế cách biểu diễn ôtômat có nhớ trong bảng M_1 (cũng như trong bảng M_2) khu vực trạng thái tiếp theo khu vực hàm (f) sẽ hoàn toàn không theo quy luật "đếm" nữa. Khi đó khu vực trạng thái tiếp theo của ôtômat có nhớ sẽ có các trạng thái tiếp theo xuất hiện (tùy ý) theo từng yêu cầu chức năng cụ thể. Đồng thời khu vực đáp ứng ra hay khu vực hàm ra (g) của ôtômat có nhớ, các giá trị lôgic điều khiển cụ thể (0 hay 1) đối tượng bị điều khiển cũng xuất hiện (tùy ý) không theo một quy luật nào, mà hoàn toàn phụ thuộc vào một nhiệm vụ điều hành cụ thể. Vấn đề còn lại ở là từ những cách mô tả ôtômat có nhớ (hệ dãy) ta thực hiện thiết kế lắp ráp ôtômat có nhớ như thế nào. Như vậy chúng ta có thể hiểu là : một ôtômat có nhớ thì khu vực hàm chuyển biến trạng thái (f), khu vực đáp ứng ra hay hàm ra (g) hoàn toàn có thể được thay đổi tùy ý (không theo quy luật đếm nữa), theo một chức năng cụ thể nào đó được đưa ra. Việc ôtômat có khả năng thay đổi cả ở khu vực hàm f và cả ở khu vực hàm g

đã tạo nên sự linh hoạt đến mức độ tuyệt vời của hệ thống số (ví dụ : như máy dệt hoa văn tự động, hàn chân IC tự động, cắm linh kiện điện tử tự động, hay thêu thùa tự động...). Đồng thời cũng chính việc cho phép thay đổi trạng thái ở khu vực hàm (f) lẫn đáp ứng ra ở khu vực hàm (g). Khi biểu diễn ôtômat có nhớ lại làm nẩy sinh ra các vấn đề kỹ thuật cần phải giải quyết. Điều "nẩy sinh" mới này trong hệ thống số sẽ được đề cập đến ở phần tiếp theo sau đây.

§4.3. PHÂN HOẠCH THẾ VÀ DÀN PHÂN HOẠCH THẾ D_M

Ở trong phần đầu quyển sách chúng ta đã nói tới phân hoạch, đã hiểu thế nào là phân hoạch và các đặc điểm của nó. Có thể nhắc lại là : phân hoạch là sự nhóm họp tùy tiện các phần tử (thông thường là các trạng thái của ôtômat) của một tập nào đó cho trước. Dàn là một tập sắp xếp mà các phần tử của nó đều có hai phép toán giao và phép toán hợp ($\cap$, $\cup$). Dàn cho chúng ta biết tất cả các tập có cùng một thuộc tính nếu phần tử nguồn của dàn có cùng chung một thuộc tính.

Dựa vào tính chất và đặc điểm của phân hoạch cũng như tính chất của dàn, ở đây chúng ta sẽ đưa ra một khái niệm quan trọng, một bước không thể thiếu được trong quá trình thực hiện thiết kế ôtômat có nhớ, đó là vấn đề rút gọn ôtômat có nhớ. Trước khi hiểu được trọn vẹn quá trình cực tiểu hóa ôtômat có nhớ, chúng ta phải nắm được một khái niệm quan trọng trong lý thuyết đại số đó là phân hoạch thế. Một phân hoạch mà phân hoạch hy vọng có tính chất thay thế thì được gọi tắt là "phân hoạch thế" (PHT).

Phân hoạch thế (PHT) là một sự nhóm họp tùy tiện các phần tử (thông thường là các trạng thái của ôtômat) mà hy vọng là chúng có thể thay thế được cho nhau. Ở đây chúng ta chỉ cần có "hy vọng" thay thế trong sự nhóm họp tùy tiện đó mà thôi.

Bài toán tìm phân hoạch thế (PHT) chủ yếu dựa trên tập trạng thái (S) của ôtômat (M), thông thường nó được viết :

$$S = (s_1, s_2, ..., s_p)$$

Trong đó : p là số lượng trạng thái của ôtômat M.

Để cho đơn giản từ nay trở đi chúng ta sẽ viết tập trạng thái S dưới dạng

$$S = (1, 2, 3, ..., p)$$

4.3.1. MỘT SỐ KÝ HIỆU MỚI

Trong phần I khi nghiên cứu về cấu trúc đại số chúng ta đã đề cập tới phân hoạch và các phép toán cơ bản của đại số phân hoạch. Một số quy ước cách viết các ký hiệu mới về phân hoạch và dàn như sau :

Ví dụ : Cho trước tập trạng thái ôtômat :

$$S = (1, 2, 3, 4, 5, 6, 7)$$

Từ tập S cho trước đó ta có thể có các cách nhóm họp hay phân hoạch Π_1 và Π_2 như sau :

$$\Pi_1 = \big((1,2,3),\ (4),\ (5,6)\big)$$

$$\Pi_2 = \big((2,3),\ (4),\ (5),\ (6,7)\big)$$

Hay có thể viết dưới dạng là :

$$\Pi_1 = (\overline{1,2,3}\ ,\ \overline{4}\ ,\ \overline{5,6})$$

$$\Pi_2 = (\overline{2,3}\ ,\ \overline{4}\ ,\ \overline{5}\ ,\ \overline{6,7})$$

Với quy ước : các khối có một phần tử thì không cần viết nhưng phải coi như nó tồn tại vì trong quá trình biến đổi toán học của ôtômat có nhớ các khối có một phần tử sẽ giữ nguyên giá trị và hình dáng của nó cho tới kết quả cuối cùng không thay đổi, ta lại có thể viết các phân hoạch đó dưới dạng.

$$\Pi_1 = (\overline{1,2,3)}\ ,\ \overline{5,6}) = (A,B)$$

$$\Pi_2 = (\overline{2,3)}\ ,\ \overline{6,7}) = (C,D)$$

* Ký hiệu $B_{\Pi}(s)$ sẽ được hiểu là : phần tử (s) nằm trong khối (B) nào đó của phân hoạch Π.

Ví dụ : $A_{\Pi_1}(2)$ nghĩa là phần tử 2 nằm trong khối A của phân hoạch Π_1.

$D_{\Pi_2}(7)$ là phần tử 7 nằm trong khối D của phân hoạch Π_2.

* Ký hiệu $a \equiv b\ (\Pi)$ có nghĩa là : phần tử a và phần tử b nằm trong cùng một khối nào đó của phân hoạch Π. (Ví dụ theo nhóm họp Π_1, Π_2 ở trên).

Ví dụ : $1 \equiv 2\ (\Pi_1)$ có nghĩa là phần tử 1 và phần tử 2 nằm trong cùng một khối của phân hoạch Π_1.

$6 \equiv 7\ (\Pi_2)$ có nghĩa là phần tử 6 và phần tử 7 nằm trong cùng một khối của phân hoạch Π_2.

Với ký hiệu mới $a \equiv b\ (\Pi)$ cho ta thông tin là : Phần tử a và phần tử b nằm trong cùng một khối của phân hoạch Π là đủ thông tin, còn muốn biết cụ thể hơn chúng nằm trong khối nào của phân hoạch Π, thì dùng thêm ký hiệu $B_{\Pi}(s)$ nữa, sẽ chỉ được chính xác các phần tử a và b nằm trong cùng một khối và chúng nằm cụ thể trong khối nào (A, B hay D) của phân hoạch Π nào đó.

Dựa vào các ký hiệu mới đó chúng ta sẽ đưa ra được quan hệ toán học hay điều kiện để tìm PHT của ôtômat có nhớ. Đồng thời sẽ rất dễ dàng hiểu được khái niệm quan trọng trong phần này, đó là khái niệm tìm phân hoạch thế (PHT) của ôtômat có nhớ.

Cũng từ khái niệm về phân hoạch là sự nhóm họp tùy tiện, chúng ta sẽ có được các phân hoạch đặc biệt hay còn gọi là các phân hoạch (nhóm họp) tầm thường, từ ví dụ ở trên ta có phân hoạch tầm thường :

$$\Pi = (\overline{1},\overline{2},\overline{3},\overline{4},\overline{5},\overline{6},\overline{7}) = \Phi\ .\text{rỗng}$$

$$\Pi = \overline{(1,2,3,4,5,6,7)} = I\ .\text{đầy}$$

Chúng ta có hai phân hoạch tầm thường đó là phân hoạch rỗng Φ phân hoạch đầy I, chúng là những phân hoạch đặc biệt.

4.3.2. PHÂN HOẠCH THẾ CỦA ÔTÔMAT

Muốn tìm được phân hoạch thế (PHT) của ôtômat có nhớ ta có thể dựa vào định nghĩa hay điều kiện tìm PHT sau :

Định nghĩa 4.3 : Một phân hoạch Π trên tập trạng thái S của ôtômat M, được gọi là một phân hoạch thế (có tính chất thay thế) nếu và chỉ nếu ứng với mỗi bộ chữ vào (tập tín hiệu vào), hàm chuyển biến trạng thái f ánh xạ các khối của Π vào chính các khối của nó.

Nói cách khác : phân hoạch Π được gọi là một phân hoạch thế (PHT) nếu nó thỏa mãn quan hệ sau :

$$\Pi = \overline{(s_i, s_j)} \qquad \text{với } t_1 > t_0$$

$$s_i \equiv s_j (\Pi) - t_0$$

$$f(s_i, x) \equiv f(s_j, x) - t_1$$

Trong đó : $s_i, s_j \in S$ và $x \in X$

Ta có thể hiểu "công thức" (quan hệ) hay định nghĩa tìm (PHT) như sau : Từ tập số hiệu vào S ta chọn ra một phân hoạch bất kỳ Π với hai phần tử là (s_i và s_j), sau đó gộp chúng vào thành một khối hay một nhóm họp Π. Khi ta nhóm họp hai trạng thái s_i, $s_j \in S$ tạo ra một phân hoạch ($\Pi = \overline{(s_i, s_j)}$ theo ký hiệu ta có thể viết :

$$s_i \equiv s_j \ (\Pi) - t_0$$

Có nghĩa là s_i và s_j nằm trong cùng một khối (tập con) của phân hoạch Π ở thời điểm ban đầu t_0.

Tại thời điểm t_0 ta có thể viết ký hiệu quan hệ $s_i \equiv s_j$ (Π) có nghĩa là : ở trạng thái ban đầu hai trạng thái s_i và s_j nằm trong cùng một khối (một bọc) của phân hoạch Π.

Tại thời điểm tiếp theo t_1 ($t_1 > t_0$) dưới tác động của tín hiệu vào x ($x \in X$) sẽ tạo ra trạng thái mới tiếp theo đó là : $f(s_i, x)$ và $f(s_j, x)$ tương ứng với hai trạng thái ban đầu là s_i và s_j. Nếu tại thời điểm mới t_1 mà có quan hệ hay có thể viết :

$$f(s_i , x) \equiv f(s_j , x) \ (\Pi) - t_1$$

Nghĩa là dưới tác động của tín hiệu vào (x) tạo ra trạng thái mới, mà các trạng thái mới này chúng vẫn nằm trong cùng một khối (nào đó) của phân hoạch Π. Điều đó cho phép ta kết luận phân hoạch $\Pi = \overline{(s_i, s_j)}$ là phân hoạch thế (PHT). Còn nếu dưới tác động của tín hiệu vào (x) sinh ra trạng thái mới không thỏa mãn điều kiện của PHT hay không viết được theo quan hệ trên, thì chứng tỏ phân hoạch Π đó không phải là phân hoạch thế.

Sau khi có điều kiện tìm phân hoạch thế (PHT), ta sẽ áp dụng nó để tìm phân hoạch thế cho một ôtômat có nhớ như sau.

Ví dụ : Tìm phân hoạch thế cho ôtômat hoạt động theo bảng chức năng trên hình 4.19.

S \ X	0	1	Z
1	2	3	0
2	①	③	0
3	1	4	0
4	1	5	1
5	1	4	1

Hình 4.19. Cho ôtômat bất kỳ.

Từ tập trạng thái S của ôtômat đã cho, ta nhóm họp tùy tiện hay lấy ra một phân hoạch Π_1 nào đó như sau :

$$\Pi_1 = \overline{(1, 2)}$$

Theo định nghĩa có thể viết : $1 \equiv 2\ (\Pi_1) - t_0$

$$\begin{cases} f(1, 0) = 2 \equiv f(2, 0) = ① \ (\Pi_1) - t_1 \\ f(1, 1) = 3 \equiv f(2, 1) = ③ \ (\Pi_1) - t_1 \end{cases}$$

Ta thực hiện nhóm họp ngẫu nhiên trạng thái (1) và trạng thái (2) tạo ra phân hoạch $\Pi_1 = \overline{(1, 2)}$, do chúng cùng nằm trong một khối của phân hoạch Π nên ta có thể viết :

$$1 \equiv 2\ (\Pi_1) - t_0$$

Như vậy tại thời điểm ban đầu t_0 hay trạng thái trước đó hai trạng thái (1) và (2) là ở trong cùng một khối. Dưới tác động của tín hiệu vào (x) cụ thể có giá trị là (x = 0 và x = 1) theo bảng chức năng, tại thời điểm t_1 dưới tác động của tín hiệu vào x = 0 ta có trạng thái tiếp theo là :

$$f(1, 0) = 2 \quad \text{và} \quad f(2, 0) = ① - t_1$$

Như vậy trạng thái tiếp theo tại t_1 theo bảng hoạt động của ôtômat có nhớ đã cho ta có trạng thái tiếp theo là 2 và ① đó là các trạng thái mới sinh ra, so sánh chúng với phân hoạch $\Pi_1 = \overline{(1, 2)}$ lúc đầu thấy trạng thái mới sinh ra là (2 và ①) vẫn nằm trong cùng một khối là $\overline{(1, 2)}$ của Π_1, từ đó ta có thể viết :

$$f(1, 0) = 2 \equiv f(2, 0) = ① \ (\Pi_1) - t_1$$

Dựa vào kết quả thu được là 2 và ① ta điền ký hiệu (≡) thể hiện chúng vẫn nằm trong cùng một khối của Π_1 đã chọn. Như vậy dưới tác động của tín hiệu vào (x = 0) trạng thái tiếp theo của ôtômat đã cho vẫn nằm trong cùng một khối $\Pi_1 = \overline{(1, 2)}$ chọn

trước đó. Đó là với tín hiệu vào (x = 0), còn trường hợp với tín hiệu vào (x = 1) ta có trạng thái tiếp theo :

$$f(1,\ 1) = 3 \text{ và } f(2,\ 1) = \textcircled{3} - t_1$$

Dưới tác động của tín hiệu vào (x = 1) chúng ta thu được các trạng thái tiếp theo là (3 và ③). Theo quy ước từ dấu khối có một phần tử thì ta không cần viết nhưng phải coi như nó tồn tại nên ta có thể viết phân hoạch Π_1 dưới dạng :

$$\Pi_1 = \overline{(1,2)} = \overline{(1,2)}\ ,\ \overline{3}) = (A, B)$$

Vậy khi ta thu được trạng thái tiếp theo là (3 và ③) dưới tác động của tín hiệu vào (x = 1), thì các trạng thái mới được tạo ra đó vẫn nằm trong khối ($\overline{3}$ = B) của phân hoạch Π_1, nghĩa là chúng vẫn nằm trong cùng một khối của Π_1 nên chúng ta có thể viết :

$$f(1,1) = 3 \equiv f(2,1) = \textcircled{3}\ (\Pi_1) - t_1$$

Chú ý : một cách tổng quát từ trường hợp trạng thái tiếp theo đã nêu (3 và ③) ta có thể áp dụng coi như chúng vẫn nằm trong cùng một khối khi các trạng thái tiếp theo của chúng giống nhau, hay như nhau.

Vậy chúng ta có thể kết luận phân hoạch $\Pi_1 = \overline{(1,2)}$ là phân hoạch thế (PHT) (có tính chất thay thế).

Với cách làm tương tự ta nhóm họp một phân hoạch Π_1^* mới có quan hệ :

$$\Pi_1^* = \overline{(1,3)} \qquad t_2 > t_1 > t_0$$

$$1 \equiv 3\ (\Pi_1^*) - t_0$$

$$\begin{cases} f(1,0) = 2\ ?\ f(3,0) = 1 - t_1 \\ f(1,1) = 3\ ?\ f(3,1) = 4 - t_1 \end{cases}$$

Ta kích tiếp (kiểm tra tiếp) vào các trạng thái tiếp theo mà chúng tạo thành cặp (1,2) và (3,4). Cặp (1,2) kích tiếp tín hiệu (x = 1 và 0) kết quả ta đã biết ở trên, chỉ còn cặp (3,4) tiếp theo ta kiểm tra (kích) tiếp.

$$f(3\ ,\ 0) = 1 \equiv f(4\ ,\ 1) = 1 - t_2$$

$$f(3\ ,\ 1) = 4\ ?\ f(4\ ,\ 1) = 5 - t_2$$

Ở đây ta kích tiếp tục được vì ôtômat hoạt động liên tục theo tác động của tín hiệu vào, đồng thời trong "điều kiện" tìm hay theo định nghĩa tìm phân hoạch thế không nới ánh xạ bao nhiêu lần, vì vậy ta có thể kích tiếp tín hiệu (x) vào các trạng thái tiếp theo đã được sinh ra, với hy vọng trạng thái tiếp đó nó lại trở lại với trạng thái trong cùng một khối của Π_1^* .

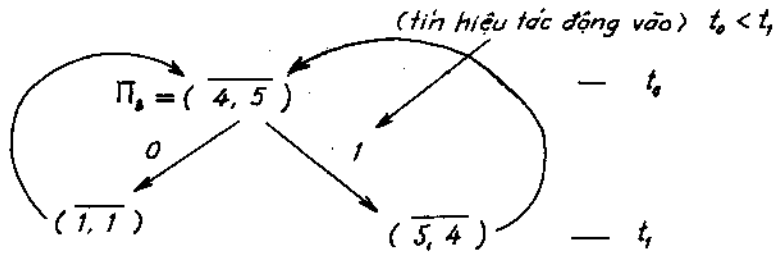
Đến đây các trạng thái mới sinh ra là (4, 5) vậy chúng ta có thể kích tiếp (kiểm tra tiếp), nhưng đến đây chúng ta không cần kiểm tra tiếp nữa (kích tiếp nữa) vì

ôtômat hoạt động kiểu vòng quanh, kích tiếp lại trở lại như các trạng thái trước đó tại trạng thái ban đầu t_0, đồng thời đến đây ta cũng thấy trong quá trình kích tiếp theo (kiểm tra các cặp tiếp theo) ta thấy toàn bộ trạng thái của ôtômat trong ví dụ nêu ra đều đã được xuất hiện, điều này tương đương với phân hoạch đầy I tầm thường ($I = \overline{(1,2,3,4,5)}$) đó phân hoạch $\Pi_1^* = \overline{(1,3)}$ tương đương một phân hoạch đầy I nên có thể kết luận : phân hoạch $\Pi_1^* = \overline{(1,3)}$ không phải là phân hoạch thế.

Như vậy ta có thể nhận thấy là dựa vào điều kiện tìm phân hoạch thế (PHT) có thể tìm được phân hoạch thế hoặc không tìm được phân hoạch thế trong bảng trạng thái hoạt động của ôtômat có nhớ cho trước. Cách tìm PHT như trên thực hiện rất chậm.

Sau đây chúng ta xét một phương pháp nhanh hơn để tìm phân hoạch thế :

Ví dụ : Ta có một phân hoạch mới $\Pi_1 = \overline{(4,5)}$, với hai trạng thái 4 và 5 trong phân hoạch Π_1, ta thấy tín hiệu kích vào x cũng chỉ có hai giá trị là 0 và 1. Vậy chúng ta có thể kích 0 và kích 1 vào phân hoạch $\Pi_1 = \overline{(4,5)}$, sẽ thu được kết quả chỉ ra trên hình 4.20.

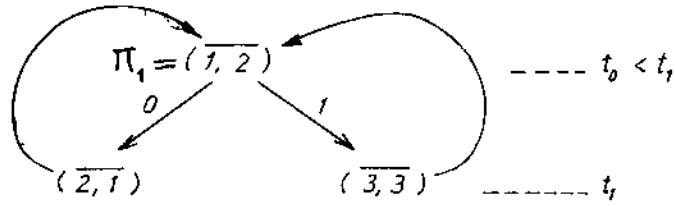


Hình 4.20. Lưu đồ chuyển biến trạng thái.

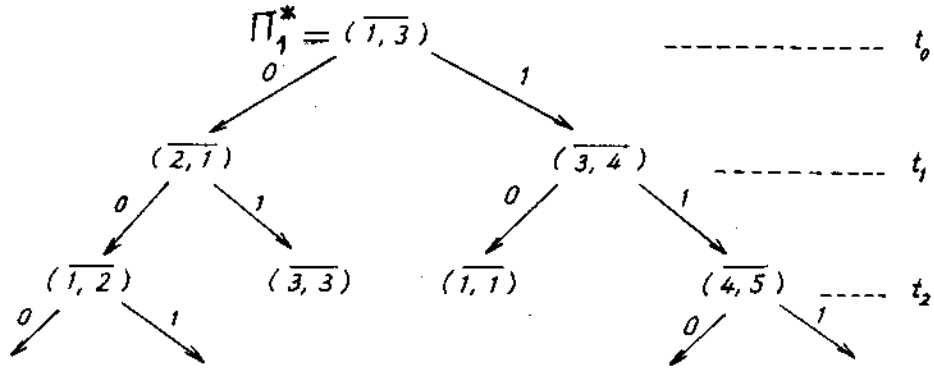
Qua hình 4.20 kết quả thu được sau khi kích, ta thấy trạng thái tiếp theo của chúng dưới tác động của tín hiệu kích vào là 0 và 1, tại thời điểm t_1 lại giống hệt hay nói cách khác là lại quay trở lại trạng thái trước đó tại t_0. Khi thu được kết quả trạng thái trước đó dưới tác động kích từ tín hiệu bên ngoài (x), tạo ra trạng thái tiếp theo lại quay lại như trạng thái trước đó có nghĩa là : kích mãi thì kết quả vẫn không thay đổi. Đây là một hiện tượng rất đặc biệt của quá trình chuyển biến trạng thái của ôtômat có nhớ, khi mà hoạt động của nó là bất kỳ, không còn theo quy luật của bộ đếm nữa. Hiện tượng hoạt động thành từng cặp trạng thái, dưới tác động của tín hiệu vào sinh ra trạng thái tiếp theo giống hệt cặp trạng thái trước đó, ta có thể gọi là phân hoạch thế hay có tính chất thay thế.

Ta có thể dùng cách vẽ đồ hình để kiểm tra lại phân hoạch thế $\Pi_1 = \overline{(1,2)}$ đã tìm thấy ở trên như trên 4.21.

Cũng với cách làm này ta có thể kiểm tra lại $\Pi_1^* = \overline{(1,3)}$ đã biết là l0 phải là phân hoạch thế, như trên trên hình 4.22.



Hình 4.21. Kiểm tra lại PHT theo lưu đồ trạng thái.

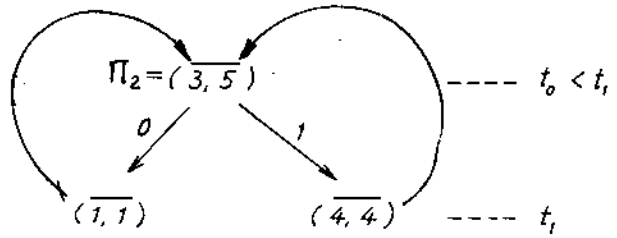


Hình 4.22. Kiểm tra không phải PHT.

Ta thấy trạng thái tiếp theo của chúng không quay lại (giống) trạng thái trước đó và cứ phát triển mãi, vậy $\Pi_1^* = \overline{(1, 3)}$ không phải là phân hoạch thế. Trong thực tế quá trình nhóm họp thành từng cặp phân hoạch các trạng thái của ôtômat có nhớ là các cặp không phải là phân hoạch thế, còn nếu tìm được phân hoạch thế (PHT) chỉ là trường hợp đặc biệt.

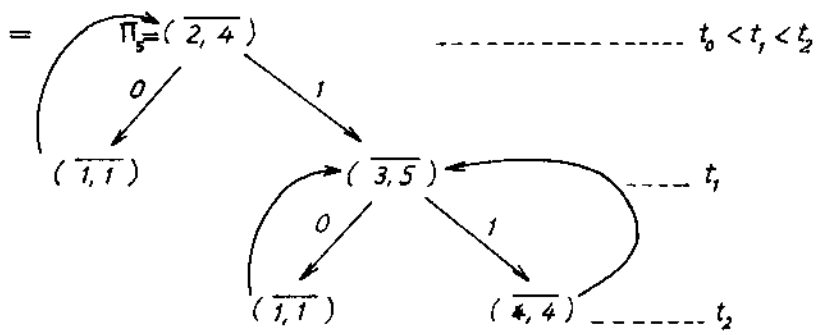
Để không bỏ sót các phân hoạch không được kiểm tra chúng ta có thể nhóm họp các trạng thái của ôtômat có nhờ lần lượt theo mọi khả năng có thể có trong tập S của nó.

Các phân hoạch thế khác của ôtômat đã cho có thể tìm được thêm như trên hình 4.23.



Hình 4.23. Thêm một PHT.

Kết quả $\Pi_2 = \overline{(3, 5)}$ là phân hoạch thế. Ở đây ta có một phân hoạch thế đặc biệt, đó là trường hợp phân hoạch đặc biệt tìm được trên hình 4.24.



Hình 4.24. Một PHT đặc biệt.

Nếu ngay từ đầu ta nhóm họp $\Pi_5 = \overline{(2,4}\ ,\ \overline{3,5)}$ thì kết quả cho ta thấy dưới tác động của tín hiệu vào (x) thì nhóm trạng thái $\overline{(2,4)}$ luôn luôn chuyển về nhóm trạng thái (hay là khối) $\overline{(3,5)}$, như vậy nếu ta nhóm họp chúng ngay từ đầu tạo ra một phân hoạch của chúng, thì luôn thỏa mãn là trạng thái tiếp theo luôn được ánh xạ vào một khối nào đó của Π_4. Từ đó ta có thể kết luận là phân hoạch $\{\Pi_5 = \overline{(2,4}\ ,\ \overline{3,5)}$ là một phân hoạch thế.

Đến đây từ nhiệm vụ đặt ra là tìm phân hoạch thế (PHT) cho một ôtômat cho trước trong ví dụ ở trên, chúng ta đã tìm được một tập các phân hoạch thế của ôtômat đó là :

$$\Pi_1 = \overline{(1,2)} \qquad \Pi_3 = \overline{(4,5)}$$

$$\Pi_2 = \overline{(3,5)} \qquad \Pi_5 = \overline{(2,4}\ ,\ \overline{3,5)}$$

Kết quả ta đã tìm được 4 phân hoạch thế của ôtômat đã cho ($\Pi_1, \Pi_2, \Pi_3, \Pi_5$) còn tất cả các nhóm họp khác có thể có còn lại của ôtômat đều không phải là PHT. Tuy nhiên các PHT đó chỉ là các phân hoạch thế có 2 phần tử được nhóm họp, trong khi đó nhiệm vụ là phải tìm ra tất cả các dạng, các loại phân hoạch thế của ôtômat đó, tức là còn phải tìm được các PHT có số lượng trạng thái nhóm họp lớn hơn 2 (VD : $\Pi_i = \overline{(1,2,4)}$ nhóm họp bao gồm 3 trạng thái). Vậy các nhóm họp có số lượng trạng thái nhiều hơn có phải là PHT hay không? Muốn trả lời được câu hỏi đó ta phải là dựa vào dàn phân hoạch thế (D_M). Từ đó có thể tìm ra được tất cả các loại PHT của ôtômat có nhớ.

4.3.3. DÀN PHÂN HOẠCH THẾ - D_M

Phần trên khi chúng ta tìm phân hoạch thế (PHT) theo định nghĩa, người ta không hạn chế số phần tử trong một khối của một phân hoạch. Điều đó có nghĩa là các khối của một phân hoạch được xem xét hay không có thể có 2 phần tử, 3 phần tử, 4 phần tử v.v..., miễn là chúng thỏa mãn điều kiện của phân hoạch thế. Nhưng nếu chúng ta có một phân hoạch bất kỳ, trong đó một khối của nó có một số lượng lớn các phần tử thì

số lượng các phân hoạch thế (PHT) là rất lớn tới mức không thể tìm hết được. Cho nên người ta đưa ra cách tìm tập tất cả các phân hoạch thế nhỏ nhất, rồi từ chúng có thể xây dựng và tạo ra được các phân hoạch thế có số lượng phần tử trong một khối lớn hơn (nhiều hơn 2 phần tử) mà chắc chắn là PHT, cách làm này sẽ dễ dàng thực hiện hơn để tạo ra các PHT có số lượng phần tử nhiều hơn. Phương pháp đó dựa vào việc xây dựng dàn phân hoạch thế D_M.

Dàn phân hoạch thế được xây dựng dựa trên tập tất cả các PHT nhỏ nhất, tức một phân hoạch chỉ có 2 phần tử. Phân hoạch có hai phần tử mà có tính chất là một PHT, thì phân hoạch đó có một tên mới được gọi theo định nghĩa sau.

Định nghĩa 4.4 : Một cặp trạng thái được gọi là một cặp nhận dạng (p, q) nếu dưới tác động của tín hiệu vào tạo ra trạng thái tiếp theo, thì trạng thái tiếp theo đó chúng vẫn chỉ nằm trong cùng một khối của phân hoạch ban đầu (tức là thỏa mãn điều kiện của PHT với phân hoạch có 2 trạng thái).

Một phân hoạch thế nhỏ nhất ta có thể viết $\Pi^m_{\equiv}(\overline{p,q})$ trong đó (p, q) là một cặp trạng thái nhận dạng. Từ những cặp trạng thái nhận dạng hay các cặp PHT nhỏ nhất, ta có thể tạo ra các phân hoạch thế lớn hơn bằng cách lấy tổng của chúng. Vậy bước đầu tiên phải tìm được tập của tất cả các cặp nhận dạng, hay tập tất cả các PHT nhỏ nhất, sau đó dựa vào định lý sau để xây dựng được các PHT lớn hơn.

Định lý 4.1 : Nếu phân hoạch Π_1 và Π_2 là các phân hoạch thế trên tập trạng thái S của ôtômat M, thì $(\Pi_1 + \Pi_2)$ và $(\Pi_1 . \Pi_2)$ cũng là phân hoạch thế.

Chứng minh : Ta chứng minh $(\Pi_1 + \Pi_2)$ là PHT giả sử $\Pi_1 = (\overline{p,q})$ là PHT và $\Pi_2 = (\overline{m,n})$ là PHT như vậy tại thời điểm ban đầu t_0 ta có quan hệ theo toán học :

$$t_0 \rightarrow (\Pi_1 + \Pi_2) = (\overline{p,q}\ ,\ \overline{m,n})$$

Do Π_1 và Π_2 là phân hoạch thế (PHT) có nghĩa là dưới tác động của tín hiệu vào trạng thái tiếp theo vẫn trở lại trạng thái trước đó (giống hệt trạng thái trước đó) nên ta có thể viết :

$$\overline{(p,q)} \xrightarrow{x} \overline{(p,q)} \quad \text{và} \quad \overline{(m,n)} \xrightarrow{x} \overline{(m,n)}$$

t_0 t_1 t_0 t_1

Như vậy dưới tác động của tín hiệu vào (x) tại t_1 tổng các PHT $(\Pi_1 + \Pi_2)$ các khối của nó vẫn không thay đổi, tức là có thể viết :

$$t_1 \rightarrow (\Pi_1 + \Pi_2) = (\overline{p,q}\ ,\ \overline{m,n})$$

Như vậy khi Π_1 và Π_2 là các phân hoạch thế thì tổng của chúng $(\Pi_1 + \Pi_2)$ cũng là phân hoạch thế.

Trở lại với ví dụ trước đây để minh họa kết quả của định lý như sau :

Phân hoạch $\Pi_1 = (\overline{1,2})$ và $\Pi_3 = (\overline{3,4})$ là các PHT

Vậy :

$$\Pi^* = (\Pi_1 + \Pi_2) = (\overline{1,2}) + (\overline{3,4}) = (\overline{1,2}\ ,\ \overline{3,4}) = (\overline{1,2,3,4})$$

Từ đó ta có thể kết luận luôn :

$\Pi^* = (\Pi_1 + \Pi_2) = (\overline{1,2,3,4})$ là PHT mà không cần kiểm tra Π^* nữa.

Tương tự ta có :

$$\Pi_3 = (\overline{3,4}) \text{ và } \Pi_2 = (\overline{4,5}) \text{ là các PHT}$$

Vậy :

$$\Pi^{**} = (\Pi_2 + \Pi_3) = (\overline{4,5}) + (\overline{3,4}) = (\overline{4,5}\ ,\ \overline{3,4}) = (\overline{3,4,5})$$

Ta cũng có kết luận :

$$\Pi^{**} = (\Pi_2 + \Pi_3) = (\overline{3,4,5}) \text{ là PHT.}$$

Từ định lý 4.1 cho phép ta thực hiện quan hệ tổng của các phân hoạch thế nhỏ nhất, thu được các phân hoạch thế lớn hơn đó là :

$$\Pi^* = (\overline{1,2,3,4}) \text{ - có 4 phần tử}$$

$$\Pi^{**} = (\overline{3,4,5}) \text{ - có 3 phần tử.}$$

Từ đó nếu muốn có những phân hoạch thế có số phần tử trong một khối lớn hơn 2, hay muốn có các loại, các dạng phân hoạch thế của một ôtômat có nhớ, ta chỉ cần tìm đầy đủ tập các phân hoạch thế nhỏ nhất với hai phần tử, sau đó cộng chúng lại với nhau sẽ được các loại PHT của ôtômat cho trước. Cách đó cũng chính là cơ sở để xây dựng nên dàn PHT D_m của ôtômat có nhớ.

Cách vẽ dàn phân hoạch thế (PHT) của ôtômat có nhớ

Trong phần này trình bày phương pháp vẽ dàn phân hoạch thế của ôtômat có nhớ D_M, và một phương pháp có thể tìm được mọi tập có cùng một thuộc tính từ một nguồn tập con có trước. Ở phần lý thuyết về dàn đã nói ở phần đầu ta đã biết dàn là một tập sắp sếp mà trong đó mọi phần tử của nó đều có (chứa) hai phép toán là "hợp" và "giao". Để xây dựng nên dàn phân hoạch thế D_M ta dựa vào định lý sau.

Định lý 4.2 : Tập hợp tất cả các phân hoạch thế (PHT) trên tập trạng thái S của ôtômat M tạo thành dàn D_M với phần tử lớn nhất và bé nhất tương ứng là I và $\varnothing$ và chúng tuân theo sự sắp xếp tự nhiên của các phân hoạch.

Chứng minh : Do dàn là tập sắp xếp của các phần tử chứa các phép toán (+) và (•), cho nên nếu các phần tử của nó là PHT thì việc thực hiện "cộng" của các PHT (chính là cộng véctơ) sẽ tạo ra tập mới (phần tử mới) là một PHT nên phần tử mới đó chắc chắn thuộc (nằm trong) dàn D_M là dàn phân hoạch thế.

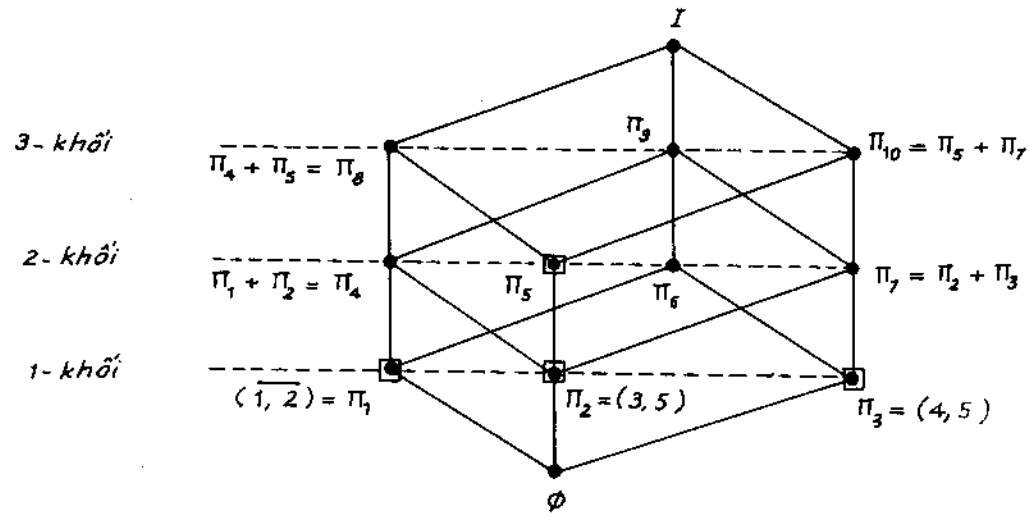
Ta có ví dụ minh họa việc vẽ dàn phân hoạch thế D_M cho ôtômat đã được nêu ra trong ví dụ trên.

Ta có tập của toàn bộ các phân hoạch thế, nhỏ nhất có hai phần tử là :

$$\Pi_1 = (\overline{1,2}) \qquad \Pi_3 = (\overline{4,5})$$

$$\Pi_2 = (\overline{3,5}) \qquad \Pi_5 = (\overline{2,4}\ ,\ \overline{3,5})$$

Từ các phân hoạch thế nhỏ nhất tìm được đó, ta có thể tìm được các phân hoạch thế lớn hơn, hay các dạng khác nữa của PHT của ôtômat đã cho. Dàn phân hoạch thế D_M, được chỉ ra trên hình 4.25.



Hình 4.25. Dàn phân hoạch thế.

Nhìn vào hình 4.25, ta thấy các phần tử của dàn đều được tạo ra bởi tổng của hai phần tử như cách cộng vectơ. Do là tổng của các phân hoạch thế nên kết quả các phần tử của dàn hay các phân hoạch được tạo ra chắc chắn đó là một phân hoạch thế.

Ví dụ như :

$$\Pi_1 + \Pi_2 = (\overline{1,2}) + (\overline{3,5}) = (\overline{1,2}\ ,\ \overline{3,5}) = \Pi_4$$

$$\Pi_1 + \Pi_3 = (\overline{1,2}) + (\overline{4,5}) = (\overline{1,2}\ ,\ \overline{4,5}) = \Pi_6$$

$$\Pi_2 + \Pi_3 = (\overline{3,5}) + (\overline{4,5}) = (\overline{3,5}\ ,\ \overline{4,5}) = \Pi_7$$

Từ cách làm cộng các PHT (như cách cộng vectơ) ta còn nhận được các thông tin khác nữa là :

$$\Pi_1 + \Pi_2 = (\overline{1,2}\ ,\ \overline{3,5}) = (\overline{1,2,3,5}) = \Pi_4$$

$$\Pi_2 + \Pi_3 = \left((\overline{3,5})\ ,\ (\overline{4,5})\right) = (\overline{3,4,5}) = \Pi_7$$

Kết quả thu được $\Pi_4 = (\overline{1,2,3,5})$ và $\Pi_7 = (\overline{3,4,5})$ là các phân hoạch thế có số lượng phần tử trong một khối lớn hơn. Đồng thời chúng được sắp xếp tuân theo quy luật tăng dần, có nghĩa là chúng nằm ở hàng có 2 khối (ở đây ta phải hiểu là hai khối, mà mỗi khối có hai phần tử, hay là 2 khối bé nhất).

Định nghĩa 4.5 : Các phân hoạch mà từ đó chúng ta có thể xây dựng được *dàn nguyên* bằng cách lấy tổng của chúng gọi là các phần tử sinh.

Ví dụ : $\Pi_1 + \Pi_2 = \Pi_4$ thì chúng ta gọi Π_1 và Π_2 là phần tử sinh của Π_4 hay tạo ra phaan hoạch thế Π_4.

$\Pi_5 + \Pi_7 = \Pi_{10}$ thì Π_5 và Π_7 là phần tử sinh của Π_{10}.

Định nghĩa 4.6 : Các phân hoạch không được tạo thành từ cách lấy tổng của các phân hoạch khác được gọi là *các phần tử sinh cơ bản*.

Ví dụ : Π_1, Π_2, Π_3 và Π_5 ở trong dàn phân hoạch thế D_M là các phần tử sinh cơ bản, vì chúng không được tạo thành bằng cách lấy tổng của các phân hoạch thế hay không được tạo ra từ các phân hoạch thế khác. Chúng là các PHT được tìm ra theo định nghĩa. Chúng cũng phải tuân theo trật tự sắp xếp của tự nhiên là : (Π_1, Π_2, Π_3) là các phân hoạch thế có 1−khối, mỗi khối có 2 phần tử (khối bé nhất) cho nên chúng phải nằm ở hàng 1 khối, còn $\{\Pi_5 = (\overline{2,4}, \overline{3,5})$ có hai khối bé nhất nên nằm ở hàng 2−khối.

Định lý 4.3 : Tập của các phần tử sinh cơ bản là tập tối thiểu cần thiết để xây dựng nên dàn D_M bằng cách lấy tổng của chúng.

Đây là điều tất nhiên khi chúng ta muốn vẽ dàn hay xây dựng nên dàn. Đồng thời chúng ta có thể hiểu là tập các phần tử sinh cơ bản là tập hợp của các phân hoạch thế nhỏ nhất. Từ tập toàn bộ các phân hoạch thế nhỏ nhất đó ta vẽ hay xây dựng nên dàn phân hoạch thế D_M, thì dàn phân hoạch thế D_M sẽ cho chúng ta các dạng, các loại phân hoạch thế lớn bé của ôtômat đó, nói cách khác dàn D_M cho chúng ta toàn bộ "dòng họ" phân hoạch thế của ôtômat đã cho. Tóm lại khi có dàn phân hoạch thế D_M hay khi biết được toàn bộ "dòng họ" của phân hoạch thế, thì chúng ta sẽ có được kết luận chính xác cho công việc tiếp theo của mình.

Định nghĩa 4.7 : Phần tử sinh cơ bản không thuộc (nằm) hàng thứ nhất (1 − khối) thì được gọi là phần tử sinh cơ bản tự do.

Ví dụ : $\Pi_5 = (\overline{2,4}, \overline{3,5})$ là phần tử sinh cơ bản tự do, ngay từ đầu nó đã được nằm ở hàng thứ hai (2−khối), cụ thể ở đây Π_5 có thể được sắp xếp ở vị trí bất kỳ thuộc hàng thứ hai, nhưng nó bị nối với Π_2 vì $\Pi_2 = (\overline{3,5})$ mà trong Π_5 có khối $(\overline{3,5})$ $\left((\overline{3,5}) \in \Pi_5\right)$. Sau khi sắp xếp xong các phần tử sinh cơ bản ta chỉ cần nối chúng lại (cộng vectơ) là có thể vẽ xong dàn phân hoạch thế D_M.

Một điều cần lưu ý nữa khi vẽ dàn là : khi cộng các phân hoạch thế lại để tạo ra các phần tử mới, khi có các phần tử mới có cùng giá trị thì ta chắp chúng lại với nhau, điều này làm cho dàn D_M luôn được chụm về I. Điều này được minh họa như sau.

Ta có :

$$\Pi_4 = (\overline{1,2}, \overline{3,5})$$

$$\Pi_6 = (\overline{1,2}\ ,\ \overline{4,5})$$
$$\Pi_7 = (\overline{3,5}\ ,\ \overline{4,5})$$

Từ ba phần tử sinh đó ta tạo ra phân hoạch thế mới :

$$\Pi_9' = \Pi_4 + \Pi_6 = (\overline{1,2}\ ,\ \overline{3,5}) + (\overline{1,2}\ ,\ \overline{4,5}) = (\overline{1,2}\ ,\ \overline{3,5}\ ,\ \overline{4,5})$$

$$\Pi_9'' = \Pi_6 + \Pi_7 = (\overline{1,2}\ ,\ \overline{4,5}) + (\overline{3,5}\ ,\ \overline{4,5}) = (\overline{1,2}\ ,\ \overline{3,5}\ ,\ \overline{4,5})$$

Kết quả giá trị của Π_9' và Π_9'' hoàn toàn giống nhau nên chúng sẽ được vẽ chập lại với nhau thành Π_9.

$$\Pi_9 = \Pi_9' = \Pi_9'' = (\overline{1,2}\ ,\ \overline{3,5}\ ,\ \overline{4,5}) \text{ - thuộc 3 - khối}$$

Có những phân hoạch không thuộc dàn phân hoạch thế hay nằm ngoài dàn D_M.

Ví dụ : $\Pi_1^* = (\overline{1,3})$ không phải là một phân hoạch thế theo kết luận ở phần trước, những phân hoạch không phải là phân hoạch thế như Π_1^* của một ôtômat có nhớ có rất nhiều, thậm chí có thể nhiều hơn các phân hoạch không phải là PHT theo cách vẽ dàn :

$$\Pi^* = \Pi_1 + \Pi_1^* = (\overline{1,2}) + (\overline{1,3}) = (\overline{1,2,3})$$

Ta có thể kết luận chính xác là $\Pi^* = (\overline{1,2,3})$ không phải là PHT, vì Π_1^* không phải là phân hoạch thế, đây là một phân hoạch lớn hơn nhưng không phải là PHT, hay nói cách khác là nó nằm ngoài dàn PHT.

Từ dàn phân hoạch thế D_M ta tìm được tập các phân hoạch có khả năng thay thế nhau, hay còn được gọi là "dòng họ" nhà phân hoạch thế của ôtômat có nhớ đã cho như sau :

$$\Pi_1 = (\overline{1,2})\ ;\ \Pi_2 = (\overline{3,5})\ ;\ \Pi_3 = (\overline{4,5})$$

$$\Pi_4 = (\overline{1,2}\ ,\ \overline{3,5}) = (\overline{1,2,3,5})$$

$$\Pi_5 = (\overline{2,4}\ ,\ \overline{3,5}) = (\overline{2,3,4,5})$$

$$\Pi_6 = (\overline{1,2}\ ,\ \overline{4,5}) = (\overline{1,2,4,5})$$

$$\Pi_7 = (\overline{3,5}\ ,\ \overline{4,5}) = (\overline{3,4,5})$$

$$\Pi_8 = (\overline{1,2}\ ,\ \overline{3,5}\ ,\ \overline{2,4}\ ,\ \overline{3,5}) = (\overline{1,2}\ ,\ \overline{2,4}\ ,\ \overline{3,5}) = (\overline{1,2,3,4,5})$$

$$\Pi_9 = (\overline{1,2}\ ,\ \overline{3,5}\ ,\ \overline{1,2}\ ,\ \overline{4,5}) = (\overline{1,2}\ ,\ \overline{3,5}\ ,\ \overline{4,5}) = (\overline{1,2,3,4,5})$$

$$\Pi_{10} = (\overline{2,4}\ ,\ \overline{3,5}\ ,\ \overline{3,5}\ ,\ \overline{4,5}) = (\overline{2,4}\ ,\ \overline{3,5}\ ,\ \overline{4,5}) = (\overline{2,3,4,5})$$

Đó là toàn bộ các phân hoạch có thể có của ôtômat có nhớ trong ví dụ ở trên, còn những phân hoạch hay các nhóm họp khác của các trạng thái của ôtômat đã cho mà không giống kết quả đó, thì chắc chắn đó không phải là phân hoạch thế, hay chúng nằm ngoài dàn phân hoạch thế D_M.

Để hiểu được ứng dụng quan trọng của dàn phân hoạch thế D_M trong quá trình thực hiện ôtômat có nhớ, chúng ta đi tới phần tiếp theo.

§4.4. CỰC TIỂU HÓA SỐ LƯỢNG TRẠNG THÁI CỦA ÔTÔMAT CÓ NHỚ

Thực hiện việc rút gọn ôtômat có nhớ có nhiều phương pháp khác nhau, trong đó có cả những phương pháp thuật toán để thực hiện sự rút gọn ôtômat có nhớ. Trong phần này chúng ta sẽ đề cập đến một phương pháp có tính chất lý thuyết để rút gọn ôtômat có nhớ. Đây là một cách rút gọn đơn giản và dễ thực hiện.

Ở phần trên đối với việc mô tả một ôtômat có nhớ cho trước, chúng ta đã dựa chủ yếu vào hàm chuyển biến trạng thái f để tìm ra được các phân hoạch thế và đã xây dựng được dàn phân hoạch thế D_M. Nhưng trong bảng chức năng dùng để mô tả hoạt động của ôtômat có nhớ còn có hàm ra hay đáp ứng ra, là hàm g mà chúng ta chưa dùng tới. Để thực hiện việc rút gọn ôtômat có nhớ ta có thể sử dụng hàm g để tìm ra phân hoạch tương đương (PHTĐ) của nó.

4.4.1. PHÂN HOẠCH TƯƠNG ĐƯƠNG (PHTĐ) CỦA ÔTÔMAT CÓ NHỚ

Một trong các khái niệm quan trọng khi nghiên cứu cấu trúc của ôtômat có nhớ là khái niệm về tương đương hay còn được gọi là "cùng chức năng". Để tìm được phân hoạch tương đương (PHTĐ) của một ôtômat ta dựa vào định nghĩa sau.

Định nghĩa 4.7 : Phân hoạch Π trên tập trạng thái S của ôtômat M được gọi là một phân hoạch tương đương (PHTĐ) nếu và chỉ nếu ứng với mỗi bộ chữ vào (tập tín hiệu vào), các bộ chữ ra (đáp ứng ra) đối với các trạng thái trong cùng một khối của Π đều như nhau.

Nói cách khác phân hoạch Π trên tập trạng thái S của ôtômat M được gọi là phân hoạch tương đương nếu thỏa mãn :

* *Với ôtômat Mealy* M_1 phân hoạch tương đương là :

$$\Pi = (\overline{s_i , s_j}) \qquad t_0 < t_1$$

$$s_i \equiv s_j (\Pi) \qquad - t_0$$

$$g(s_i, x) = g(s_j , x) \qquad - t_1$$

Trong đó : $s_i, s_j \in S \quad x \in X$

Từ tập S ta chọn ra bất kỳ một phân hoạch Π có hai phần tử là s_i và s_j và gộp chúng thành một khối (một nhóm họp). Từ đó tại thời điểm t_0 ta có thể viết :

$$s_i \equiv s_j (\Pi) \qquad - t_0$$

Tại thời điểm t_1 ($t_1 > t_0$ dưới tác động của tín hiệu vào x ($x \in X$) sẽ tạo ra đáp ứng ra của trạng thái s_i là $g(s_i, x)$ và đáp ứng ra của s_j là $g(s_j, x)$, thì hai đáp ứng ra đó của chúng phải như nhau (tức là cùng làm một việc giống nhau) nên có thể viết

$$g(s_i, x) = g(s_j, x) \quad - t_1$$

Điều đó có nghĩa là : Trước đó tại t_0 thì hai trạng thái s_i và s_j nằm trong cùng một khối, nhưng dưới tác động của tín hiệu vào thì đáp ứng ra của chúng vẫn như nhau, giống nhau hay cùng làm một việc.

* *Với ôtômat Moore* M_2 phân hoạch tương đương phải thỏa mãn quan hệ :

$$\Pi = (\overline{s_i, s_j}) \qquad t_0 < t_1$$

$$s_i \equiv s_j\ (\Pi) \qquad - t_0$$

$$g(s_i) = g(s_j) \qquad - t_1$$

Trong đó : $s_i, s_j \in S$ và $x \in X$

Ở đây do cách biểu diễn ôtômat có nhớ của Moore đáp ứng ra Z không phụ thuộc vào tín hiệu vào x một cách trực tiếp, cho nên hàm g không có tín hiệu x tác động vào khi biểu diễn tại t_1.

Trở lại ví dụ của ôtômat có nhớ đã nêu ở phần trên để tìm phân hoạch tương đương của nó.

Ví dụ : Tìm phân hoạch tương đương cho ôtômat cho trước trên hình 4.26.

Do hoạt động của ôtômat biểu diễn theo bảng Moore nên muốn tìm phân hoạch tương đương (PHTĐ) cho ôtômat đã cho ta dựa vào điều kiện tìm theo quan hệ của Moore :

Ta chọn một phân hoạch bất kỳ $\Pi_1 = (\overline{1,2})$.

$$\Pi_1 = (\overline{1,2}) \qquad t_0 < t_1$$

$$1 \equiv 2\ (\Pi_1) \qquad - t_0$$

$$g(1) = \triangle 0 = g(2) = \boxed{0} \quad - t_1$$

Kết quả thu được đáp ứng ra của chúng là như nhau, đều bằng 0.

Với cách làm tương tự chúng ta tìm ra được các phân hoạch tương đương (PHTĐ) của ôtômat đã cho như sau :

$\Pi_2 = (\overline{1,3})$; $\Pi_3 = (\overline{2,3})$;

$\Pi_4 = (\overline{4,5})$

Còn với phân hoạch $\Pi_5 = (\overline{1,4})$ sẽ không phải là phân hoạch thế tương đương vì :

$$\Pi_5 = (\overline{1,4}) \qquad t_0 < t_1$$

$$1 \equiv 4\ (\Pi_4) - t_0$$

$$g(1) = 0 \neq g(4) = 1 - t_1$$

S \ X	0	1	Z
1	2	3	△0
2	1	3	[0]
3	1	4	0
4	1	5	1
5	1	4	1
	f		g

Hình 4.26. Một ôtômat cho trước.

Kết quả cuối cùng, tập các phân hoạch tương đương nhỏ nhất (với hai phần tử) của ôtômat đã cho là :

$$\Pi_1 = (\overline{1,2}) \qquad \Pi_2 = (\overline{1,3})$$

$$\Pi_3 = (\overline{2,3}) \qquad \Pi_4 = (\overline{4,5})$$

Từ tập các phân hoạch tương đương (PHTĐ) nhỏ nhất đã tìm được, chúng ta có thể xây dựng được dàn phân hoạch tương đương của ôtômat đã cho theo cách vẽ dàn như cách vẽ dàn phân hoạch thế D_M, từ đó tìm được "dòng họ" các phân hoạch tương đương của ôtômat đã cho.

Như vậy dựa vào hàm chuyển biến trạng thái f và hàm đáp ứng ra g chúng ta đã tìm được tập các phân hoạch thế (PHT) và tập các phân hoạch tương đương (PHTĐ) của ôtômat có nhớ kết quả thu được đó như sau :

Tập các phân hoạch thế (PHT)	Tập các phân hoạch tương đương (PHTĐ)
$\Pi_1 = (\overline{1,2})$	$\Pi_1 = (\overline{1,2})$
$\Pi_2 = (\overline{3,5})$	$\Pi_2 = (\overline{1,3})$
$\Pi_3 = (\overline{4,5})$	$\Pi_3 = (\overline{2,3})$
$\Pi_4 = (\overline{1,2}, \overline{3,5})$	$\Pi_4 = (\overline{4,5})$
$\Pi_5 = (\overline{2,4}, \overline{3,5})$	
$\Pi_6 = (\overline{1,2}, \overline{4,5})$	
$\Pi_7 = (\overline{3,5}, \overline{4,5})$	
Π_8 , Π_9 , Π_{10}	

Từ kết quả thu được của các tập phân hoạch thế và tập các phân hoạch tương đương chúng ta nhận thấy : có phân hoạch chỉ là phân hoạch thế (PHT) mà không phải là phân hoạch tương đương (PHTĐ).

Ví dụ : $\Pi_2 = (\overline{3,5})$ - ở tập các PHT.

Có phân hoạch lại chỉ là phân hoạch tương đương (PHTĐ) mà không phải là phân hoạch thế (PHT).

Ví dụ : $\Pi_2 = (\overline{1,3})$ - ở tập các PHTĐ.

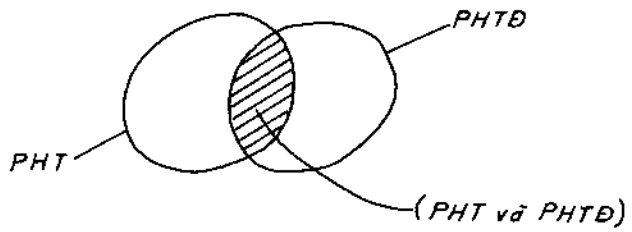
Có phân hoạch lại vừa là phân hoạch thế vừa là phân hoạch tương đương như.

Ví dụ : $\Pi_1 = (\overline{1,2})$ hay $\Pi_3 = (\overline{4,5})$

Có thể minh họa kết quả như ở hình 4.27.

Định lý 4.4 : Một phân hoạch Π vừa là một phân hoạch thế vừa là một phân hoạch tương đương thì được gọi là một phân hoạch thế tương đương (PHTTĐ). Một phân

hoạch thế tương đương bao gồm tập các khối và trong mỗi khối bao gồm tập của các trạng thái tương đương thay thế được cho nhau.



Hình 4.27. Quan hệ giữa PHT và PHTĐ.

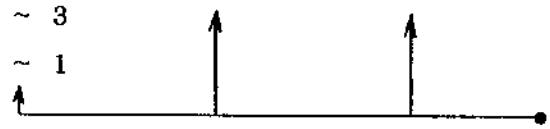
Ví dụ : Ta có một phân hoạch thế tương đương Π như sau (ký hiệu dấu ~ thể hiện sự tương đương thay thế được cho nhau)

$$\Pi = \left((\overline{1,2,3}) , (\overline{4,5}) , (\overline{6,7}) \right) = (A, B, C)$$

Thì ta có : $1 \sim 2$ $\quad 4 \sim 5$ $\quad 6 \sim 7$

$2 \sim 3$

$3 \sim 1$



Khi ta có được một phân hoạch (một nhóm họp) là phân hoạch thế tương đương (PHTTĐ) thì có thể khẳng định là các trạng thái trong cùng một khối của nó là có thể thay thế được cho nhau vì :

Trước hết nó là một phân hoạch thế (PHT) tức là các trạng thái của phân hoạch có khả năng thay thế, do chúng luôn hoạt động theo từng cặp, hay có thể nói trong từng khối của PHT dưới tác động của tín hiệu tác động vào (x), khi có tín hiệu (x) tác động thì trạng thái tiếp theo luôn trở lại trạng thái trước đó, trở về lại khối của nó nên có khả năng thay thế được.

Đồng thời nó lại là một phân hoạch tương đương (PHTĐ) tức là các trạng thái trong phân hoạch hoàn toàn giống nhau về chức năng hay tương đương nhau về chức năng.

Như vậy một phân hoạch thế tương đương (PHTTĐ) là một sự nhóm họp đặc biệt của ôtômat có nhớ, là sự nhóm họp của các trạng thái có thể thay thế được cho nhau, mà các trạng thái đó lại tương đương nhau về chức năng, cho nên chúng ta có thể khẳng định chính xác là : hai hay các trạng thái trong một khối của phân hoạch thế tương đương là hoàn toàn có thể thay thế cho nhau.

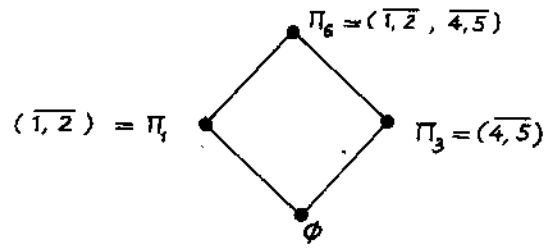
Đến đây từ ví dụ ôtômat đã nêu ở trên ta có được tập các phân hoạch thế tương đương (PHTTĐ) như sau :

$$\Pi_1 = (\overline{1,2}) \quad ; \quad \Pi_3 = (\overline{4,5})$$

Định lý 4.5 : Tích hay tổng của các phân hoạch thế tương đương là một phân hoạch thế tương đương.

Định lý cho phép chúng ta vẽ được dàn phân hoạch thế tương đương D_M^*. Cách vẽ dàn phân hoạch thế tương đương giống hệt cách vẽ dàn phân hoạch thế D_M.

Dàn phân hoạch thế tương đương được biểu diễn trên hình 4.28.



Hình 4.28. Dàn phân hoạch thế tương đương.

Định lý 4.6 : Tập hợp tất cả các phân hoạch thế tương đương trên trạng thái S của ôtômat M tạo ra dàn con D_M^* của dàn phân hoạch thế D_M.

$$D_M^* \subseteq D_M$$

Định lý 4.7 : Mỗi một ôtômat hoàn toàn xác định M có một phân hoạch duy nhất Π ứng với ôtômat tối thiểu (ôtômat có tập trạng thái S ít nhất). Phân hoạch đó nằm ở mức cao nhất của dàn phân hoạch thế tương đương D_M^*.

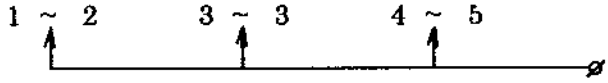
Định lý 4.7 cho chúng ta thấy phương pháp cực tiểu hóa hay rút gọn ôtômat có nhớ hoàn toàn xác định dựa vào việc tìm kiếm phân hoạch thế tương đương (PHTTĐ) là phương pháp làm rất đơn giản và dễ dàng. Sau khi tìm được các phân hoạch thế tương đương nhỏ nhất gồm có hai trạng thái, chúng ta chỉ cần vẽ dàn phân hoạch thế tương đương D_M^* cho nó, sau đó tìm được phân hoạch cao nhất của dàn phân hoạch thế tương đương thì đó chính là ôtômat tối thiểu $\text{ôtômat}_{\min}$. Theo ví dụ đã cho ta có ôtômat tối thiểu là :

$$\text{ôtômat}_{\min} = \Pi_6 = (\overline{1,2}\ ,\ \overline{4,5})$$

Nhưng do từ đầu chúng ta đã có quy ước là khối có một phần tử thì không cần viết nhưng phải coi như nó tồn tại. Nay ta đã có kết quả cuối cùng của quá trình tìm hiểu ôtômat tối thiểu, cho nên kết quả đầy đủ ta phải viết :

$$\text{ôtômat}_{\min} = \Pi_6 = (\overline{1,2}\ ,\ \overline{3}\ ,\ \overline{4,5})$$

Do Π_6 là phân hoạch thế tương đương nên ta có thể khẳng định được các trạng thái thay thế cho nhau là



Từ sự khẳng định thay thế được đó chúng ta có kết quả minh họa trên hình 4.29.

Ở đây ta chú ý là có khối có một phần tử (khối $\overline{3}$) nghĩa là trạng thái 3 là trạng thái mà nó chỉ có thể thay thế cho chính nó, nói cách khác là trạng thái duy nhất quan trọng mà không thể thay thế được bằng trạng thái khác, đó là trạng thái không thể thiếu của ôtômat có nhớ là trạng thái không thể rút gọn được.

Bảng chức năng

S \ X	0	1	Z
~~1~~	~~1~~	~~3~~	~~0~~
1	1	3	0
3	1	4	0
4	1	4	1
~~4~~	~~1~~	~~4~~	~~1~~

Hình 4.29. Cách thay thế rút gọn trạng thái ôtômat.

Sau khi thay thế các trạng thái vào bảng chức năng của ôtômat có nhớ chúng ta thấy có các hàng của ôtômat có các trạng thái hoàn toàn giống nhau, nên ta hoàn toàn có thể xóa đi được một hàng, hay rút gọn được một hàng, sau khi xóa xong chúng ta thu được ôtômat tối thiểu có số lượng trạng thái ít nhất như hình 4.30.

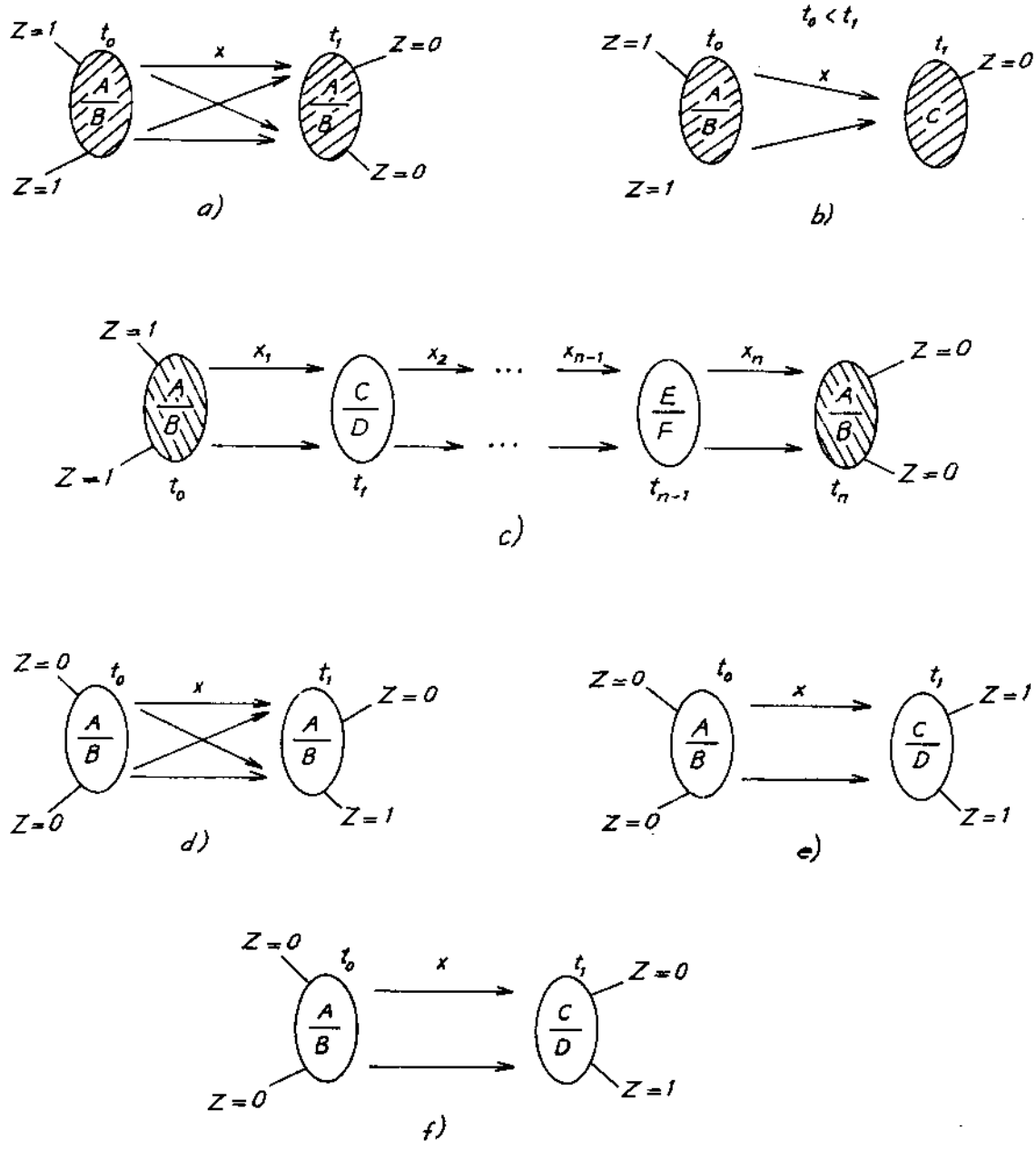
S \ X	0	1	Z
1	1	3	0
3	1	4	0
4	1	4	1

Hình 4.30. Ôtômat tối thiểu.

Sau khi thực hiện quá trình rút gọn ôtômat có nhớ, ôtômat có năm trạng thái, nay còn có ba trạng thái, nhưng chức năng của ôtômat vẫn hoàn toàn không thay đổi, hay chức năng vẫn tương đương ôtômat đã cho. Nói một cách khác chúng ta đã có được một ôtômat tối thiểu với số lượng trạng thái là ít nhất với cùng chức năng. Toàn bộ lý thuyết tìm phân hoạch thế (PHT), phân hoạch tương đương (PHTĐ) trong quá trình rút gọn ôtômat có nhớ hoàn toàn xác định được minh họa trên hình 4.31.

Ở đây chúng ta đưa ra hai trạng thái tổng quát là A và B, hình vẽ minh họa trường hợp nào thì hai trạng thái A và B có thể thay thế được cho nhau hay khẳng định là rút gọn được dưới tác động của tín hiệu vào (x) và trường hợp nào chúng không thể thay thế rút gọn được hay không thỏa mãn là phân hoạch thế tương đương.

Từ hình vẽ ta thấy các (hình 4.31 a, b, c) hai trạng thái (A, B) nhóm họp lại chúng có thể rút gọn (chập lại) chúng lại làm một được vì : dưới tác động của tín hiệu vào x chúng sinh ra trạng thái tiếp theo lại trở lại trạng thái trước đó là (A, B) tức là thỏa mãn quan hệ nhóm họp của chúng là một phân hoạch thế (PHT), đồng thời đáp ứng ra



Hình 4.31. Minh họa quá trình rút gọn ôtômat.

Z của chúng lại giống nhau (cùng là 0 hoặc cùng là 1) nên chúng lại thỏa mãn thêm điều kiện của phân hoạch tương đương (PHTĐ), cho nên nhóm họp (A, B) vừa có tính chất là (PHT) vừa có tính chất là (PHTĐ) nên chúng có thể thay thế được cho nhau hay nói cách khác là chúng có thể chập lại làm một hay chỉ tương đương với một trạng thái.

Hình 4.3,d : quan hệ nhóm họp của (A, B) dưới tác động của tín hiệu vào sinh ra trạng thái mới vẫn là (A, B) vậy chúng thỏa mãn điều kiện là phân hoạch thế (PHT), nhưng do đó đáp ứng ra Z của chúng khác nhau nên không thỏa mãn là phân hoạch tương đương (PMTĐ), cho nên trong trường hợp này ta không thể thay thế hay rút gọn chúng được.

Hình 4.31,e : nhóm họp (A, B) dưới tác động của tín hiệu vào lại sinh ra trạng thái (C, D) như vậy chúng không phải là phân hoạch thế (PHT) mặc dù điều kiện đầu ra Z của chúng cùng giống nhau (cùng là Z = 1) thỏa mãn là PHTĐ. Trường hợp này sự nhóm họp đó cũng không thể rút gọn được.

Hình 4.31,f : sự nhóm họp của (A, B) dưới tác động của tín hiệu vào (x) tạo ra quan hệ vừa không phải là phân hoạch thế (vì kết quả là (C, D), vừa không phải là phân hoạch tương đương vì điều khiển đầu ra Z của chúng khác nhau, nên trong trường hợp này không thể rút gọn được.

Việc rút gọn ôtômat có nhớ trong quá trình thực hiện thiết kế hay tổng hợp ôtômat có nhớ là điều bắt buộc phải làm và không thể thiếu được trong các bước hay trong quá trình xây dựng lắp ráp ôtômat có nhớ, vì việc rút gọn ôtômat sẽ tránh cho chúng ta gặp phải trường hợp bất khả kháng khi xây dựng ôtômat, đồng thời đã là một hệ thống số dùng để thực hiện hay điều khiển một hệ thống kỹ thuật thì các trạng thái dùng để điều hành phải là duy nhất để tránh sự nhầm lẫn nhằm đảm bảo độ tin cậy hoạt động của hệ thống điều khiển trong quá trình vận hành.

4.4.2. PHƯƠNG PHÁP THUẬT TOÁN DÙNG ĐỂ RÚT GỌN ÔTÔMAT CÓ NHỚ

Có một phương pháp rút gọn trạng thái của ôtômat có nhớ là "phương pháp thuật toán", hay phương pháp "dùng mẹo" và không có một lý thuyết rõ rệt như phương pháp rút gọn đã được đề cập.

Để thực hiện việc rút gọn ôtômat có nhớ chúng ta đã xây dựng nên một lý thuyết dựa vào khái niệm tìm ra phân hoạch thế (PHT) và phân hoạch tương đương (PHTĐ), sau đó vẽ dàn phân hoạch thế D_M hay chính xác hơn là vẽ dàn phân hoạch thế tương đương D_M^*. Từ dàn D_M^* ở vị trí cao nhất của nó đó chính là ôtômat min (ôtômat có số lượng trạng thái nhỏ nhất). Khi ta vẽ được dàn đồng nghĩa với việc ta tìm được những khả năng thay thế cao nhất của các trạng thái của ôtômat có nhớ cho trước.

Phương pháp rút gọn ôtômat có nhớ dùng thuật toán dựa trên việc xây dựng nên một ma trận các trạng thái của hệ thống. Ma trận này có dạng là một tam giác vuông cân, gồm các hàng và cột được kẻ song song với các cạnh của góc vuông của tam giác này. Số lượng hàng và số lượng cột của ma trận là bằng nhau, và bằng chính số lượng trạng thái (S) của bảng chức năng của ôtômat có nhớ cho trước. Mục đích của việc xây dựng ma trận tam giác vuông cân này là tìm được các khả năng tương tác có thể có của ôtômat đã cho với tác động của tín hiệu vào x của nó. Thứ tự của hàng ta sẽ ghi lần lượt từ trên xuống các trạng thái của ôtômat cần phải rút gọn trạng thái vào cột ở cực trái của ma trận tam giác vuông. Thứ tự của cột ghi lần lượt từ trái sang phải từ cột thứ hai lần lượt các trạng thái của ôtômat đã cho.

Từ quan hệ giữa hàng và cột của ma trận tam giác vuông cân, ta sẽ kẻ các đường thẳng song song giữa chúng tạo ra các ô vuông tương ứng với tọa độ giữa hàng và cột hay là tọa độ của các trạng thái của ôtômat tương ứng với nhau. Với cách làm này chúng ta thu được mọi khả năng tác động của ôtômat có nhớ với tín hiệu vào của nó và hoàn toàn không thiếu hay bỏ sót một trường hợp nào.

Sau khi có được một ma trận tam giác vuông cân và kẻ xong các ô vuông tương ứng với tọa độ của hàng và cột của tam giac vuông cân, tiếp theo từng ô vuông ứng với tọa độ của hàng và cột là các trạng thái của ôtômat, ở đây tọa độ của ô vuông ứng với trạng thái của ôtômat có nhớ, thì (tọa độ) hay một cặp trạng thái đó sẽ được coi là cặp trạng thái ban đầu của ôtômat ứng với thời điểm t_0. Điều này giống như một cặp trạng thái nhận dạng (p, q) hay là một phân hoạch nhỏ nhất gồm hai trạng thái ta đã nói ở phần trên.

Sau đó ta sẽ điền vào ô vuông (ứng với tọa độ của hai trạng thái đó) các cặp trạng thái tiếp theo tương ứng với các tín hiệu kích vào x ứng với thời điểm t_1 ($t_0 < t_1$). Sau khi điền xong trạng thái tiếp theo vào ô vuông ứng với tọa độ của cặp trạng thái trước đó, ta sẽ kiểm tra xem cặp trạng thái tiếp theo đó có phải là cặp hoạt động tương đương hay không? Nếu các cặp trạng thái tiếp theo có đáp ứng ra Z giống nhau hay tương đương nhau thì ta đánh dấu (√) vào ô vuông đó. Còn nếu các trạng thái tiếp theo trong ô vuông không tương đương nhau hay có đáp ứng ra Z khác nhau, thì ta đánh dấu (X) vào ô vuông đó.

Ví dụ : Thực hiện rút gọn ôtômat có nhớ cho trong bảng chức năng sau trên hình 4.32.

Bảng chức năng

S \ x	0	1	Z
1	2	3	0
2	1	3	0
3	1	4	0
4	1	5	1
5	1	4	1

Hình 4.32. Ôtômat làm ví dụ.

Ma trận tam giác vuông cân của ôtômat đã cho được xây dựng như hình 4.33.

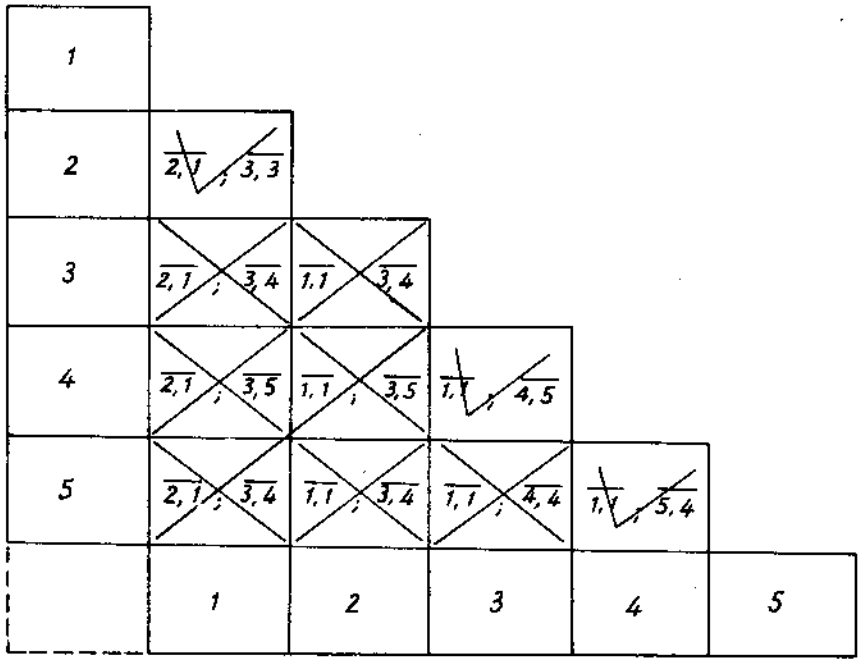
Tọa độ của các ô vuông trong bảng được xác định như sau :

Ứng với tọa độ đầu tiên (2, 1), đó cũng chính là một cặp trạng thái đầu trước đó. Dưới tác động của tín hiệu vào (x = 0) sẽ tạo ra trạng thái tiếp theo là $(\overline{2,1})$ như thấy ở trên bảng chức năng.

Tương tự cặp trạng thái đầu (2, 1) dưới tác động của tín hiệu vào x = 1 sẽ tạo ra cặp trạng thái tiếp theo là $(\overline{3,3})$. Như vậy trong ô vuông đầu tiên ứng với tọa độ đầu tiên (2,1) ta sẽ ghi vào trong ô vuông đã cặp các trạng thái tiếp theo xuất hiện dưới tác động của tín hiệu vào hay xuất hiện ở trên bảng trạng thái của ôtômat đã cho là $\boxed{\overline{2,1};\overline{3,3}}$. Sau đó ta kiểm tra tiếp xem trạng thái tiếp theo đó có tương đương nhau về đáp ứng ra hay có tương đương về hoạt động điều khiển hay không cụ thể như sau

$$g(2) = g(1) = 0$$

$$g(3) = 0$$



Hình 4.33. Ma trận tam giác vuông cân.

Ta thấy hoạt động đáp ứng ra của chúng tương đương nhau cho nên ô vuông ứng với tọa độ đầu tiên (2,1) đó ta ký hiệu vào ô đó dấu (√).

Tương tự tiếp theo ta có tọa độ của ô vuông khác là (3, 1), 3 và 1 đó là trạng thái ban đầu trước đó tạo thành một cặp hay một phân hoạch nhỏ nhất. Dưới tác động của tín hiệu vào x (là 0 hoặc 1) sẽ tạo ra cặp trạng thái tiếp theo tương ứng theo bảng trạng thái đã cho là ($\overline{2,1}$ và $\overline{3,4}$), các cặp trạng thái tiếp theo đó ta sẽ ghi vào trong ô vuông của ma trận tam giác vuông cân như trên hình 4.33. Sau đó chúng ta kiểm tra xem các trạng thái tiếp theo mới được sinh ra đó có tương đương hay cùng nhau về đáp ứng ra hay không, cụ thể ta có :

$$g(2) = g(1) = 0$$

$$g(3) = 0 \neq g(4) = 1$$

Từ kết quả đó chúng ta thấy ($\overline{3,4}$) chúng không tương đương nhau, nên kết quả là ô vuông ứng với tọa độ (1, 3) là không tương đương nhau, nên chúng ta ký hiệu dấu (X) vào ô vuông của tọa độ đó.

Sau khi kiểm tra đầy đủ các ô trong ma trận tam giác đó ta sẽ chọn ra các ô có dấu (√) đó là các ô vuông có các trạng thái tiếp theo hoạt động đáp ứng ra là tương đương nhau, chúng tạo thành một tập P các cặp trạng thái hoạt động tương đương nhau.

$$P = (\overline{2,1} ; \overline{3,3} ; \overline{1,1} ; \overline{4,5} ; \overline{1,1} ; \overline{5,4}) =$$

$$P = (\overline{2,1} ; \overline{3,3} ; \overline{4,5})$$

Từ tập P kết quả cuối cùng đó chúng ta có được kết luận là :

Trạng thái 2 có thể thay thế vào trạng thái 1.

Trạng thái 3 thì thay vào cho chính nó.

Trạng thái 4 thì có thể thay cho trạng thái 5.

Kết quả rút gọn ôtômat có nhớ theo phương pháp thuật toán hay ma trận tam giác vuông cân hoàn toàn giống như ôtômat tối thiểu mà chúng ta đã tìm được ở phần trên là có ôtômat min ở trên hình 4.34.

Bảng chức năng ôtômat$_{min}$

S \ x	0	1	Z
1	1	3	0
3	1	4	0
4	1	4	1

Hình 4.34. Ôtômat min.

4.4.3. TỐI THIỂU HÓA TRẠNG THÁI CỦA ÔTÔMAT KHÔNG HOÀN TOÀN XÁC ĐỊNH

Ôtômát không hoàn toàn xác định là ôtômat mà trong bảng chức năng có các vị trí hay các ô bỏ trống, không xác định rõ giá trị cụ thể, các ô bỏ trống này có thể nằm trong khu vực hàm f (các trạng thái tiếp theo) cũng có thể thuộc khu vực hàm g (các đáp ứng ra của ôtômat).

Bản chất của các ô trống trong khu vực hàm f đó chính là : các trạng thái tiếp theo sẽ không xuất hiện nếu tín hiệu vào bị "cấm", tức là tín hiệu vào nào đó của ôtômat bị cản không cho xuất hiện hay không bao giờ tác động vào trong quá trình hoạt động của ôtômat. Trong trường hợp trạng thái tiếp theo dù được sinh ra dưới tác động của tín hiệu vào nào đó nhưng trạng thái tiếp theo đó cũng không được dùng để điều khiển hay không có tác dụng gì trong quá trình hoạt động của ôtômat. Trong trường hợp trạng thái tiếp theo sẽ được coi là một ô trống không xác định trong bảng chức năng hay bảng chuyển biến trạng thái của ôtômat.

Bản chất của ôtômat xuất hiện trong khu vực hàm g là : trong trường hợp có bộ tín hiệu vào nào đó bị cấm không được xuất hiện, làm cho trạng thái tiếp theo tương ứng cũng không được xuất hiện, khi đó đáp ứng ra sẽ là ô trống với giá trị không xác định. Hoặc ở trong trường hợp trạng thái tiếp theo được xuất hiện nhưng nó không dùng để điều khiển gì, thì khi đó đáp ứng ra của nó cũng được coi là một ô trống có giá trị là không xác định.

Trong bảng chức năng hay bảng chuyển biến trạng thái của ôtômat nếu thấy có ô trống có giá trị là không xác định, ta có thể chủ động "gán" vào các ô trống đó các giá trị khác nhau của hàm chuyển biến trạng thái f hay hàm ra g của ôtômat (tùy theo trong từng điều kiện cụ thể) sao cho có lợi nhất. Thông thường ta sẽ gán vào các ô trống các giá trị sao cho sẽ nhận được ôtômat tối thiểu tức là ôtômat có mạch điện đơn giản nhất hay số lượng trạng thái là ít nhất.

Khi ta "gán" các giá trị cụ thể của ôtômat vào các ô trống có giá trị không xác định, tức là ta đã chuyển một ôtômat không xác định thành một ôtômat xác định, nói

cách khác là khi ta gán vào ô trống các giá trị cụ thể là ta đã chuyển quá trình rút gọn một ôtômat không xác định thành quá trình rút gọn một ôtômat hoàn toàn xác định.

Cách gán các giá trị cụ thể vào các ô trống không xác định là cách duy nhất cho phép ta rút gọn được ôtômat không hoàn toàn xác định, vì đối với ôtômat không hoàn toàn xác định thì quy luật bắc cầu có thể không còn đúng, điều đó dẫn đến tìm ra phân hoạch thế (PHT) và phân hoạch tương đương (PHTĐ) sẽ không nhất quán, khi đó tìm ra được tập phân hoạch thế tương đương (PHTTĐ) lại càng khó và việc vẽ dàn PHTTĐ (D_M^*) là không thực hiện được cho nên thuật toán tối tiểu hóa ôtômat có nhớ với ôtômat hoàn toàn xác định không còn có thể áp dụng để rút gọn ôtômat có nhớ không xác định được nữa bởi vì : thuật toán tối thiểu hóa trạng thái của ôtômat chủ yếu xây dựng trên cơ sở phải tìm ra tập phân hoạch thế tương đương nằm ở mức cao nhất của dàn phân hoạch thế tương đương (D_M^*). Như vậy muốn rút gọn được ôtômat có nhớ không xác định ta bắt buộc phải "gán" các giá trị theo mọi khả năng để chuyển nó về dạng ôtômat hoàn toàn xác định, từ đó mới thực hiện được quá trình rút gọn. Sau khi tìm được ôtômat nhỏ nhất ứng với mọi khả năng "gán" các giá trị khác nhau của ôtômat, nếu thấy ôtômat nào có số lượng trạng thái ít nhất trong tất cả thì ta chọn đó là ôtômat tối thiểu của ôtômat có nhớ không xác định đã cho. Ở đây ta chỉ chú ý một điều là : ôtômat tối thiểu sau cùng, khi ta so sánh kết quả với các ôtômat rút gọn thu được trong quá trình gán các giá trị, thì ôtômat rút gọn đó luôn luôn nằm ở mức cao nhất của dàn phân hoạch thế tương đương D_M^*.

Bảng chức năng

S \ X	0	1	Z
1	2	3	0
2	1	3	0
3	1	4	0
4	1	5	1
5	1	4	–

Hình 4.35. Cho ôtômat không xác định.

Ví dụ rút gọn ôtômat không xác định chỉ ra ở hình 4.35.

Ta nhận thấy bảng chức năng của ôtômat có nhớ không hoàn toàn xác định đã cho có một ô trống (một vị trí không xác định) thuộc khu vực hàm g là khu vực đáp ứng ra, còn khu vực hàm f thì các trạng thái tiếp theo đều có giá trị cụ thể xác định. Vị trí không xác định của hàm g chúng ta có thể gán cho nó hai giá trị là 0 hoặc là 1, quá trình "gán" giá trị cho ô trống không xác định như sau :

Giả sử chúng ta lấy giá trị ô trống là 1. Khi ta cho giá trị của ô trống là 1 thì ôtômat không xác định đã cho sẽ trở thành ôtômat hoàn toàn xác định rất quen thuộc của chúng ta, vì nó đã được xem xét nghiên cứu và rút gọn ở phần trên. Kết quả chúng ta đã thu được hai tập các phân hoạch thế (PHT) và phân hoạch tương đương (PHTĐ) như sau :

Tập các PHT	Tập các PHTĐ
$\Pi_1 = (\overline{1,2})$	$\Pi_1 = (\overline{1,2})$
$\Pi_2 = (\overline{3,5})$	$\Pi_2 = (\overline{1,3})$
$\Pi_3 = (\overline{4,5})$	$\Pi_3 = (\overline{2,3})$
$\Pi_5 = (\overline{2,4}, \overline{3,5})$	$\Pi_4 = (\overline{4,5})$

Từ tập các phân hoạch thế và phân hoạch tương đương nhỏ nhất đó, chúng ta đã tìm ra được tập phân hoạch thế tương đương nhỏ nhất (PHTTĐ) là :

$$\Pi_1 = (\overline{1,2}) \quad \text{và} \quad \Pi_3 = (\overline{4,5})$$

Từ tập các phân hoạch thế tương đương nhỏ nhất đó ta sẽ vẽ được dàn PHTTĐ (D_M^*) tương ứng với việc ta gán giá trị 1.

Giả sử bây giờ ta lại gán giá trị 0 vào ô trống. Khi cho giá trị ô trống là 0 ta nhận thấy khu vực hàm f của ôtômat đã cho không có gì thay đổi, điều đó có nghĩa là tập các phân hoạch thế (PHT) trong trường hợp này sẽ hoàn toàn giống ở phần trên (khi gán kết quả ô trống là 1), chỉ có tập các phân hoạch tương đương là có sự thay đổi, kết quả cụ thể chúng ta sẽ thu được tập các PHT và tập các PHTĐ nhỏ nhất là :

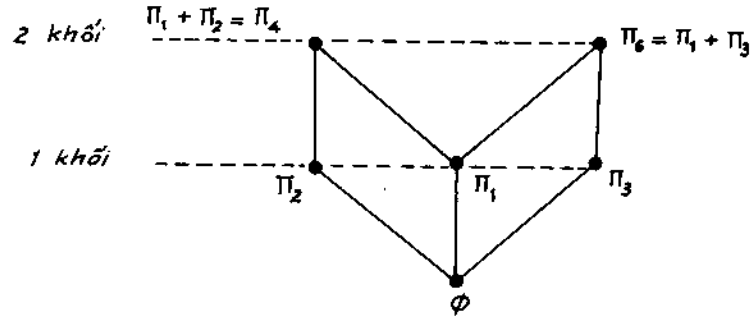
Tập các PHT	Tập các PHTĐ
$\Pi_1 = (\overline{1,2})$	$\Pi_1 = (\overline{1,2})$
$\Pi_2 = (\overline{3,5})$	$\Pi_2 = (\overline{1,3})$
$\Pi_3 = (\overline{4,5})$	$\Pi_3 = (\overline{1,5})$
$\Pi_5 = (\overline{2,4}, \overline{3,5})$	$\Pi_5 = (\overline{2,5})$
	$\Pi_6 = (\overline{3,5})$

Từ tập các phân hoạch thế và tập các phân hoạch tương đương nhỏ nhất thu được đó, ta tìm ra được tập phân hoạch thế tương đương (PHTTĐ) nhỏ nhất là :

$$\Pi_1 = (\overline{1,2}) \quad \text{và} \quad \Pi_2 = (\overline{3,5})$$

Từ tập các phân hoạch thế tương đương nhỏ nhất thu được đó sẽ vẽ được dàn PHTTĐ (D_M^*) tương ứng với việc ta gán giá trị 0 vào ô trống. Như vậy ta sẽ có "hai dàn" PHTTĐ (D_M^*), tổng quát lên nếu dàn phân hoạch thế tương đương nào cao hơn sẽ ứng với ôtômat được rút gọn hơn. Ở đây ta vẽ chung cả hai dàn PHTTĐ ứng với việc gán giá trị 0 và 1 vào ô trống của ôtômat có nhớ đã cho được chỉ ra ở trên hình 4.36.

Sau khi vẽ xong dàn PHTTĐ chung cho cả hai trường hợp chúng ta thấy : Π_6 là ôtômat min ứng với trường hợp ta gán cho ô trống giá trị 1, còn Π_4 là ôtômat min ứng



Hình 4.36. Dàn PHTTĐ của ôtômat không xác định.

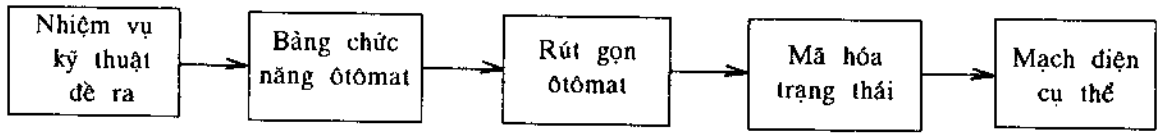
với trường hợp gán giá trị cho ô trống là 0. Hai ôtômat min cho cả hai trường hợp gán 0 và gán là 1 của ôtômat có nhớ không hoàn toàn xác định (Π_4 và Π_6) đều ở vị trí cao nhất của dàn PHTTĐ (D^*_M) nhưng độ cao của chúng bằng nhau, cho nên chúng ta có thể chọn Π_4 hoặc Π_6 làm ôtômat tối thiểu cho ôtômat không hoàn toàn xác định đã cho đều được, vì đây là một ví dụ đặc biệt không điển hình. Trong trường hợp tổng quát nếu gặp trường hợp giả sử Π_4 cao hơn Π_6 thì ôtômat min của ví dụ đã cho sẽ là Π_4.

§4.5. MÃ HÓA TRẠNG THÁI CỦA ÔTÔMAT CÓ NHỚ

4.5.1. KHÁI NIỆM CƠ BẢN CỦA MÃ HÓA TRẠNG THÁI

Bước quan trọng tiếp theo sau quá trình tối thiểu hóa (rút gọn) trạng thái của ôtômat có nhớ là việc thực hiện mã hóa trạng thái của ôtômat đã được rút gọn đó. Việc mã hóa trạng thái cho ôtômat chỉ tiến hành đối với ôtômat đã được rút gọn.

Mã hóa trạng thái của một ôtômat có nhớ nằm trong thuật toán thực hiện hay lắp ráp chức năng của ôtômat có nhớ thành mạch điện cụ thể. Quá trình thực hiện thiết kế ôtômat có nhớ được minh họa ở trên hình 4.37.



Hình 4.37. Quá trình tổng hợp ôtômat có nhớ.

Quá trình tổng hợp ôtômat có nhớ không thể bỏ qua bước mã hóa trạng thái của ôtômat.

Để hiểu rõ mã hóa, ta có thể sử dụng ví dụ đơn giản và quen thuộc, đó là từ bảng chức năng của bộ đếm 5 chỉ ra trên hình 4.38.

Ở hình 4.38a cho ta biết kết quả cụ thể và các giá trị tiếp theo của bộ đếm 5 dưới tác động của tín hiệu vào (x) là 0 hoặc là 1. Trên hình 4.38,b cho ta biết trạng thái của bộ đếm 5 nếu dùng để điều khiển một hệ thống có tính chất lần lượt kiểu bộ đếm.

S \ X	0	1
0	0	1
1	1	2
2	2	3
3	3	4
4	4	0

a)

=

S \ X	0	1
A	A	B
B	B	C
C	C	D
D	D	E
E	E	A

b)

=

$y_1y_2y_3$ \ X	0	1
0 0 0	0 0 0	0 0 1
0 0 1	0 0 1	0 1 0
0 1 0	0 1 0	0 1 1
0 1 1	0 1 1	1 0 0
1 0 0	1 0 0	0 0 0

c)

=

$y_1y_2y_3$ \ X	0	1
0 0 0	0 0 0	0 0 1
0 0 1	0 0 1	0 1 1
0 1 1	0 1 1	0 1 0
0 1 0	0 1 0	1 1 0
1 1 0	1 1 0	0 0 0

d)

Hình 4.38. Các dạng của bộ đếm 5.

Trên hình 4.38,c là bộ đếm được viết dưới dạng số nhị phân ở mã (B C D) tăng dần dưới tác động của tín hiệu vào (X). Khi bộ đếm được biểu diễn dưới dạng của các con số (0, 1) thì ở dạng đó người ta bảo là bộ đếm đã được mã hóa.

Như vậy mã hóa trạng thái là việc gán cho mỗi một trạng thái của ôtômat một tổ hợp của các con số (0, 1) hay là một tổ hợp của số nhị phân, với mục đích phân biệt các trạng thái đó so với nhau. Đảm bảo duy trì hoạt động. Nói một cách khác chính xác hơn *mã hóa trạng thái của ôtômat có nhớ là "đặt tên" các trạng thái của ôtômat dưới dạng nhị phân.*

Trên hình 4.38,d chúng ta đã mã hóa trạng thái hay "đặt tên" các trạng thái của bộ đếm 5 theo mã Grây hay quy luật nhị phân của bìa Kácnô. Với cách mã hóa bộ đếm 5 theo quy luật của bìa Kácnô ta cũng có thể lắp ráp ra được bộ đếm 5, mà trạng thái trong của nó không tăng theo quy luật của bộ đếm nhị phân thông thường, mà thay đổi theo quy luật của mã bìa Kácnô. Như vậy cũng là bộ đếm 5 nếu mã hóa theo nhị phân (B C D) sẽ có một cấu trúc mạch điện, còn nếu mã hóa theo bìa Kácnô sẽ có dạng mạch điện khác. Như vậy *cùng một chức năng hay cùng một ôtômat, nếu mã hóa khác nhau hay "đặt tên" các trạng thái khác nhau, sẽ có mạch điện khác nhau.*

Quá trình mã hóa trạng thái của ôtômat có nhớ có thể khái quát như sau :

Xuất phát từ phương trình cơ bản của ôtômat có nhớ :

$$Y = f(S, X)$$

Trong đó : $X = x_1, x_2, ..., x_n)$ là tập các tín hiệu vào của ôtômat có nhớ.

$S = (s_1\ s_2,...,s_p)$ là tập các trạng thái trong của ôtômat có nhớ.

Nhìn vào hình 4.38 biểu diễn bộ đếm 5 ta thấy trạng thái S của bộ đếm 5 mã hóa đã được chuyển thành tập các bít thông tin (0, 1) ở dạng (y_1, y_2, y_3). Tổng quát, ta có "tập các biến cần dùng để mã hóa" trạng thái S của ôtômat có nhớ là :

$$Y = (y_1, y_2, ..., y_k)$$

Trong đó : k là số lượng biến trạng thái cần dùng để mã hóa cho tập trạng thái S của ôtômat. Chúng có quan hệ là :

$$k = \left[\log_2 p\right]$$

Trong đó : k là số lượng biến trạng thái cần để mã hóa.

p là số lượng trạng thái của ôtômat có nhớ.

$[\log_2 p]$ là lấy theo phần nguyên tăng.

Ví dụ : Cho mặt ôtômat có nhớ có 5 trạng thái (p = 5), vậy số lượng biến trạng thái cần thiết để mã hóa cho các trạng thái của ôtômat đó là :

$$k = \left[\log_2 5\right] = 2{,}138 = 3$$

Như vậy k lấy theo phần nguyên tăng (k = 3) để đảm bảo chắc chắn mã hóa không bị trùng tên, và ta cần 3 biến trạng thái để mã hóa cho ôtômat đó, hay cần ba bít thông tin của số nhị phân, sau này đó chính là 3 trigơ dùng để xây dựng nên ôtômat này.

Sau khi đã có tập biến trạng thái cần dùng để mã hóa, tức là trạng thái S của ôtômat đã được thay thế bởi tập các trạng thái cần dùng để mã hóa. Như vậy ta có được phương trình cơ bản của ôtômat có nhớ ở dạng tổng quát là :

$$Y = f\left((y_1, y_2, \ldots, y_k), (x_1, x_2, \ldots, x_n)\right) = f(S, X)$$

Trong đó $(x_1, x_2, \ldots, x_n)$ là các tín hiệu vào của mạch điện, còn $(y_1, y_2, \ldots, y_k)$ là các tín hiệu của trạng thái trước đó của ôtômat. Khi tất cả các tham số của phương trình đều là "tín hiệu" điện thì phương trình toán học tổng quát đó chính là một mạch điện.

Ví dụ : Thực hiện hay tìm hệ phương trình trạng thái tiếp theo bộ đếm 5 mà cứ 3 thì báo, mã hóa theo quy luật nhị phân tăng dần và dùng ôtômat Moore thể hiện ở hình 4.39.

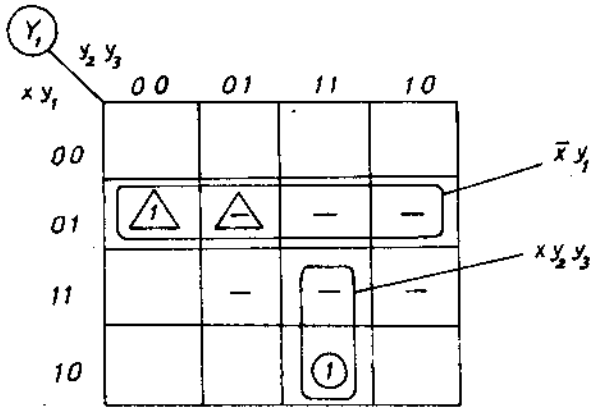
Trong phương trình tổng quát cơ bản của một ôtômat có nhớ thì Y là trạng thái tiếp theo, còn y_i là trạng thái trước đó và các x_j là tín hiệu vào. Vậy ở đây chúng ta sẽ

	$y_1y_2y_3$ \ x	0	1	Z
0	0 0 0	0 0 0	0 0 1	0
1	0 0 1	0 0 1	0 1 0	0
2	0 1 0	0 [1] 0	0 1 1	0
3	0 1 1	0 1 1	①0 0	1
4	1 0 0	△1 0 0	0 0 0	0
Bộ mã thừa	1 0 1	△– –	– – –	–
	1 1 0	– – –	– – –	–
	1 1 1	– – –	– – –	–

$Y_1\ Y_2\ Y_3$ $Y_1\ Y_2\ Y_3$

Hình 4.39. Bảng mã hóa của bộ đếm 5.

tìm hiểu hệ phương trình trạng thái tiếp theo phụ thuộc vào trạng thái trước đó và tín hiệu vào như thế nào bằng cách : Ánh xạ Y_1 vào bìa Kácnô được mô tả trên hình 4.40.



Hình 4.40. Ánh xạ Y_1 vào bìa K.

Khi thực hiện việc ánh xạ Y_1 vào bìa Kácnô thì ta thực hiện ánh xạ cả vị trí không xác định (△), vị trí có giá trị lôgic là 1 (1 và (1)) còn lại là các giá trị lôgic 0. Mỗi một giá trị của Y_1 trên bảng mã hóa có một tọa độ của nó, tọa độ đó tương ứng với một tọa độ trên bìa Kácnô (ví dụ : tọa độ ($x\ y_1\ y_2\ y_3$) = (0101) cũng chính là tọa độ của (△) trên bìa Kácnô). Với cách làm tương tự chúng ta sẽ có được các giá trị lôgic trên bảng mã hóa tương ứng với các tọa độ khác của nó trên bìa Kácnô.

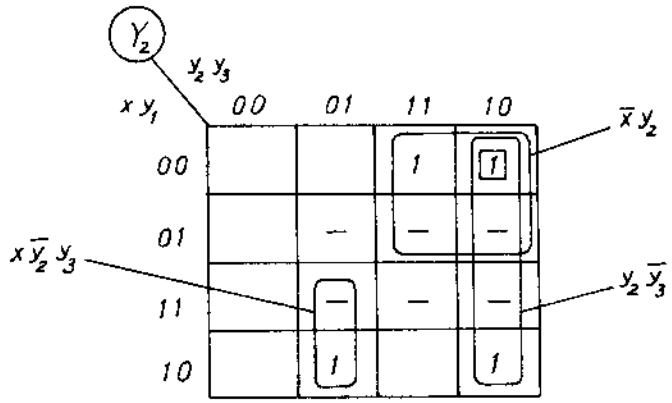
Sau khi ánh xạ xong mọi giá trị từ vị trí không xác định, vị trí xác định lôgic 1 và vị trí có giá trị lôgic 0 của Y_1 lên bìa Kácnô, ta thực hiện việc rút gọn và có được kết quả :

$$Y_1 = \bar{x}\,y_1 + x\,y_2\,y_3$$

Với cách làm tương tự ta ánh xạ Y_2 vào bìa K, mô tả trên hình 4.41.

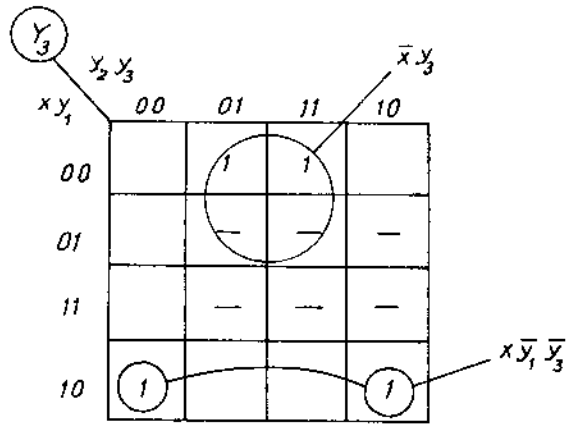
Sau khi ánh xạ Y_2 vào bài chúng ta có được kết quả :

$$Y_2 = \bar{x}\,y_2 + y_2\,\bar{y}_3 + x\,\bar{y}_2\,y_3$$



Hình 4.41. Ánh xạ Y_2 vào bìa K.

Sau đó ta ánh xạ nốt Y_3 vào bìa Kácnô được kết quả trên hình 4.42.



Hình 4.42. Ánh xạ Y_3 vào bìa K.

Kết quả thu được sau rút gọn là :

$$Y_3 = \bar{x}\, y_3 + x\, \bar{y}_1\, \bar{y}_3$$

Từ các kết quả đã rút gọn chúng ta thu được hệ các phương trình các trạng thái tiếp theo Y_i phụ thuộc vào trạng thái trước đó (y_i) và tín hiệu vào x của bộ đếm 5 là :

$$Y_1 = \bar{x}\, y_1 + x\, \bar{y}_2\, \bar{y}_3$$

$$Y_2 = \bar{x}\, y_2 + y_2\, \bar{y}_3 + x\, \bar{y}_2\, y_3$$

$$Y_3 = \bar{x}\, y_3 + x\, \bar{y}_1\, \bar{y}_3$$

$$Z = \bar{y}_1\, y_2\, y_3$$

Hệ phương trình chuyển biến trạng thái của bộ đếm 5 mã hóa theo quy luật nhị phân tăng dần mà ta thu được cũng chính là mạch điện cụ thể mà ta sẽ lắp ráp ra bộ đếm 5 mà cứ đến 3 thì báo sau này.

Một điều cần chú ý nữa ở đây là : Trong hệ phương trình trạng thái tiếp theo, ta có mô tả thêm đáp ứng ra Z ($Z = \bar{y}_2\, y_2\, y_3 = 011$), đó là đáp ứng ra của bộ đếm phụ thuộc vào trạng thái trước đó (theo y_i) và tín hiệu vào, x. Do bộ đếm 5 nhưng cứ đếm đến 3 thì báo được mô tả theo ôtômat Moore, nên Z không phụ thuộc trực tiếp vào tín hiệu vào x, vì thế trong phương trình mô tả đáp ứng ra $Z = \bar{y}_2\, y_2\, y_3$ không thấy xuất hiện biến vào x. Tín hiệu vào x chỉ tác động gián tiếp khi tạo ra đáp ứng ra Z ở bộ đếm 5.

Bộ đếm 5 cứ đếm đến 3 thì báo nếu chúng ta mã hóa hay "đặt tên" các trạng thái của nó cách khác, thì tất nhiên sẽ nhận được hệ phương trình chuyển biến trạng thái của nó có dạng khác ứng với mạch điện khác.

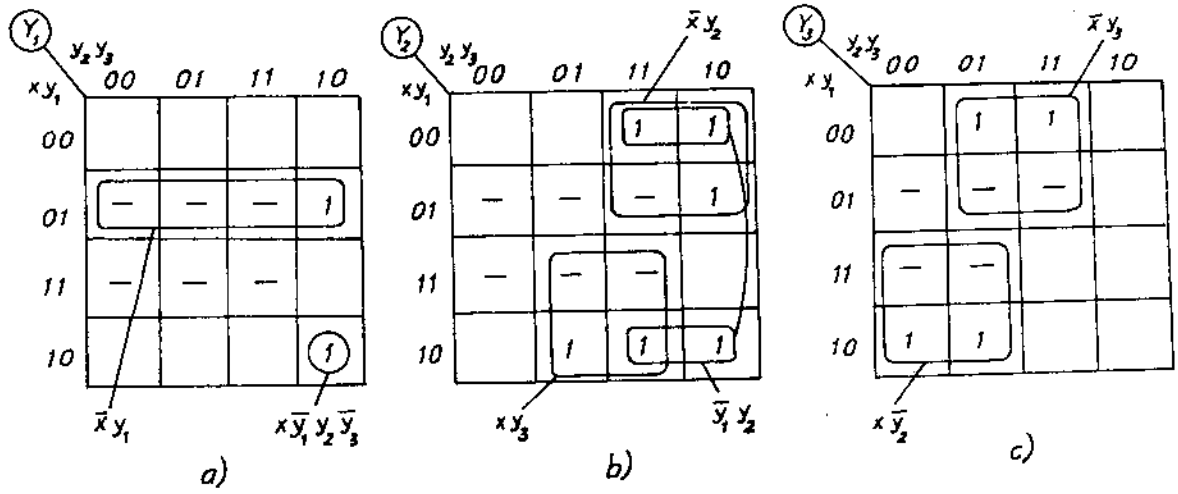
Ví dụ : Tìm hệ phương trình các trạng thái tiếp theo của bộ đếm 5 mà cứ đếm đến

3 thì báo, mã hóa theo quy luật của bìa Kácnô (mã hóa theo mã chống nhiễu Grây) dùng ôtômat Moore. Bộ đếm 5 sẽ được mô tả trên hình 4.43.

	$y_1y_2y_3$ \ x	0	1	Z
0	0 0 0	0 0 0	0 0 1	0
1	0 0 1	0 0 1	0 1 1	0
2	0 1 1	0 1 1	0 1 0	0
3	0 1 0	0 1 0	1 1 0	1
4	1 1 0	1 1 0	0 0 0	0
Bộ mã thừa	1 1 1	– – –	– – –	–
	1 0 1	– – –	– – –	–
	1 0 0	– – –	– – –	–
		$Y_1\ Y_2\ Y_3$	$Y_1\ Y_2\ Y_3$	

Hình 4.43. Bộ đếm 5 mã hóa theo bìa K.

Để tìm được hệ phương trình chuyển biến trạng thái của bộ đếm 5 mã hóa theo bìa Kácnô, chúng ta cũng thực hiện ánh xạ (Y_1, Y_2, Y_3) vào bìa K như phần trên đã làm. Kết quả có được trên hình 4.44.



Hình 4.44. Ánh xạ Y_i vào bìa Kácnô.

Sau khi thực hiện ánh xạ chúng ta rút gọn các Y_i và thu được hệ phương trình chuyển biến trạng thái có dạng :

$$Y_1 = \bar{x}\, y_1 + x\, \bar{y}_1\, y_2\, \bar{y}_3$$

$$Y_2 = \bar{x}\, y_2 + x\, y_3 + \bar{y}_1\, y_2$$

$$Y_3 = \bar{x}\, y_3 + x\, \bar{y}_2$$

$$Z = \bar{y}_1\, y_2\, y_3$$

Từ kết quả thu được nhận thấy : vẫn cùng là bộ đếm 5 cứ đếm 3 thì báo nhưng mã hóa trạng thái khác nhau (theo nhị phân tăng dần, hay mã theo bìa Kácnô) thì ta thu được hệ phương trình chuyển biến trạng thái có dạng hoàn toàn khác nhau. Ta cũng biết hệ phương trình chuyển biến trạng thái sẽ là mạch điện lắp ráp cụ thể. Như vậy ta có thể nói là : cùng một chức năng (bộ đếm) nhưng mã hóa (đặt tên) trạng thái khác nhau thì mạch điện thực hiện sẽ khác nhau.

Thông thường người ta sẽ tìm cách mã hóa sao cho có được mạch điện đơn giản nhất, điều này về lý thuyết thì dễ hiểu, nhưng thực hiện sẽ không đơn giản vì : mã hóa (đặt tên) sẽ có muôn vàn cách, nếu ôtômat có số trạng thái lớn thì đây sẽ là bài toán khó có thể tìm ra kết quả cuối cùng, vì ta phải thực hiện mọi khả năng mã hóa rồi mới đưa ra được kết luận.

Chính vì đặc điểm đó mà ở đây chúng ta sẽ nói tới một cách mã hóa sao cho có mạch "tốt" nhất (chưa chắc đã là gọn nhất), mạch tốt nhất chính là loại mạch điện dễ lắp ráp và dễ dàng tìm kiếm sửa chữa cũng như thay thế khi hỏng hóc.

4.5.2. CÁC PHƯƠNG PHÁP MÃ HÓA CHO ÔTÔMAT CÓ NHỚ

Khi chúng ta mở một máy vi tính hay một hệ thống số ra để nhìn vào mạch điện của nó, thì ta sẽ thấy mạch điện và các con IC được lắp ráp rất ngăn nắp và có trật tự. Không giống như mạch điện của TV hay của radio (các mạch tương tự) trông rất rắc rối và phức tạp. Điều này có liên quan tới cách mã hóa cho ôtômat. Thực hiện mã hóa trạng thái cho ôtômat không chỉ cho phép ta có được mạch điện trật tự gọn gàng, mà còn liên quan tới cả cấu trúc ở bên trong con IC khi người ta chế tạo ra nó. Nói cách khác là : mã hóa trạng thái của ôtômat sẽ cho chúng ta có được sự thống nhất về mạch điện từ trong lòng con IC cho tới các mạch điện bên ngoài con IC (mạch in), nó cho phép lắp ráp chế tạo một hệ thống số hiện nay được thực hiện bằng toán học trước khi nối mạch cụ thể.

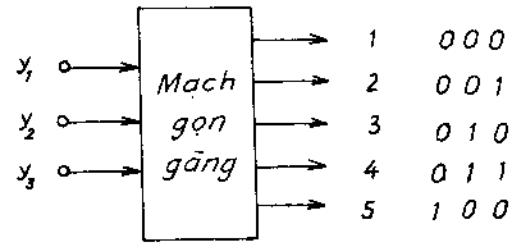
Ví dụ : Cần lắp ráp một ôtômat có 5 trạng thái dùng để điều khiển các thiết bị khác nhau.

Do có 5 trạng thái khác nhau tức là trạng thái S của ôtômat có số lượng trạng thái là 5 (p = 5). Vậy số biến trạng thái cần dùng để mã hóa là :

$$k = \left[\log_2 p\right] = \left[\log_2 5\right] = 2{,}138 = 3$$

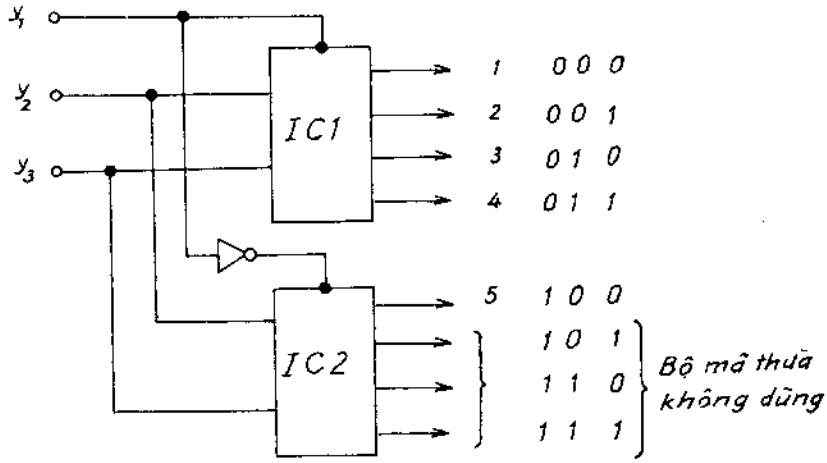
Như vậy cần 3 biến trạng thái (y_1, y_2, y_3) để mã hóa trạng thái cho ôtômat đó.

Nếu ta mã hóa sao cho có mạch đơn giản nhất thì sơ đồ mạch điện tổng quát sẽ có dạng biểu diễn ở trên hình 4.45. Mạch có đủ 5 đầu ra nhưng khó sửa chữa và thay thế. Kiểu mạch giống như của radio hay TV.



Hình 4.45. Mạch điện gọn nhất.

Do ta dùng k = 3 biến để mã hóa trạng thái cho ôtômat cho phép mã hóa $2^3 = 8$ khả năng vì chỉ cần mã hóa phân biệt có 5 trạng thái, cho nên còn 3 bộ mã bị thừa không được sử dụng. Nếu với quan điểm mã hóa sao cho có được mạch điện tốt (chấp nhận bị thừa) để quản lý và sửa chữa thay thế được dễ dàng, thì chúng ta có cách tổ chức mạch như được biểu diễn ở trên hình 4.46 nó vẫn có đầy đủ 5 đầu ra ứng với 5 trạng thái cần có để điều khiển, nhưng quản lý dễ hơn, chữa dễ hơn.



Hình 4.46. Mạch đẹp dễ thay thế sửa chữa.

Mạch điện trên hình 4.46 vẫn dùng 3 biến (y_1, y_2, y_3) để mã hóa các trạng thái, và mạch vẫn có 5 đầu ra hoàn toàn độc lập dùng để điều khiển như ở mạch điện được rút gọn ở hình 4.45.

Khi viết giá trị mã hóa (y_1 y_2 y_3) ta vẫn hiểu rõ và chính xác là biến mã hóa đầu tiên (y_1) dùng để mã hóa (hay quản lý) các IC, nó chỉ có một việc là quản lý các (IC_1 và IC_2) mà thôi. Hai biến còn lại (y_2 y_3) dùng để quản lý từng chân ra của IC, và hai biến đó chỉ quản lý chân của IC mà không liên quan gì đến biến đầu tiên (y_1). Cách bố trí mã hóa như vậy cho phép ta dễ dàng khi lập chương trình phần mềm cho máy vi tính sau này. Thông thường phần mềm tìm kiếm là tìm lần lượt từng thuộc tính riêng biệt, điều này tương ứng với việc chỉ xét riêng biến (y_1) để tìm ra IC_1 hay IC_2, sau khi tìm xong thuộc tính đầu thì phần mềm sẽ đi sâu tìm tiếp các thuộc tính tiếp theo.... Từ đó chúng ta có thể nói việc mã hóa trạng thái để đi đến lắp ráp mạch điện của hệ thống số có liên quan hữu cơ tới phần mềm lập trình trong máy vi tính.

Khi chúng ta có biến (y_1) chỉ tìm kiếm quản lý IC không liên quan tới các biến (y_2

y_3) chỉ quản lý chân IC, ngược lại các biến của IC (y_2 y_3) chỉ quản lý chân mà không liên quan gì tới biến (y_1), dùng quản lý con IC, giống như một địa chỉ gồm có khối nhà và số phòng. Ta thấy khối nhà không liên quan gì với số phòng và ngược lại số phòng không liên quan gì tới khối nhà. Nhưng chúng nó kết hợp lại với nhau thì cho ta một địa chỉ chính xác nhà nào và phòng số mấy.

Từ đó chúng ta có khái niệm về tập biến tự phụ thuộc tức là tập biến mã hóa cho một thuộc tính mà không liên quan gì tới phần mã khác quản lý một thuộc tính khác. Như vậy muốn mã hóa trạng thái của ôtômat để đạt được mạch điện "tốt" dễ quản lý, dễ lắp ráp, dễ sửa chữa và thay thế, thì chúng ta phải mã hóa sao cho tập biến trạng thái cần dùng để mã hóa (y_1, y_2, ...,y_k) phải tạo ra được các tập biến tự phụ thuộc. Một tập biến tự phụ thuộc được định nghĩa như sau.

Định nghĩa 4.6 : Tập tự phụ thuộc của các biến trạng thái cần dùng để mã hóa (y_1, y_2, ..., y_k) là tập hợp của các biến trạng thái (y_1, y_2, ..., y_i) sao cho các trạng thái tiếp theo ($1 \leqslant i \leqslant k$) chỉ phụ thuộc vào các biến trạng thái trước đó (y_1, y_2,..., y_i) và các biến vào (tín hiệu vào).

Ví dụ : Địa chỉ lớp học sinh viên nhà C12 phòng 306. Ta có thể viết C12306 từ đó ta cần có 6 bít thông tin để mã hóa địa chỉ phòng học là (y_1 y_2 y_3 y_4 y_5 y_6). Như vậy khối nhà C12 cần 3 bít thông tin để mã hóa là (y_1 y_2 y_3) còn (y_4 y_5 y_6) dùng mã hóa số phòng. Nếu ta muốn tìm khối nhà bằng cách tác động của tín hiệu vào X. Để thực hiện việc tìm kiếm thuộc tính khối nhà, trạng thái tiếp theo Y_i sẽ có giá trị mới (ví dụ là C16 còn số phòng ta không để ý). Ta thấy C16 cũng chỉ liên quan tới tổ hợp của 3 bít (y_1 y_2 y_3) như đối với C12 (trạng thái trước đó) mà thôi, còn các biến thông tin khác (y_4 y_5 y_6) quản lý về số phòng thì không có liên quan. Như vậy ta có thể nói : các biến (y_1 y_2 y_3) là một tập biến trạng thái tự phụ thuộc tập biến trạng thái đó chỉ thay đổi với chính bản thân nó theo tín hiệu vào. Tập biến trạng thái đó chỉ liên quan tới tìm ra thuộc tính khối nhà. Cũng như tập biến (y_4 y_5 y_6) cũng là một tập tự phụ thuộc liên quan tới việc tìm kiếm thuộc tính là số phòng.

Thực tế không phải bất kỳ một ôtômat nào cũng có thể tạo ra các tập tự phụ thuộc.

Đối với máy tính hay một hệ thống số có tập tự phụ thuộc (có thuộc tính riêng) ta sẽ có mạch điện lắp ráp ra mạch lọc và gọn gàng của nó, vì máy vi tính có các thuộc tính riêng như ROM, RAM, CPU... chúng hoàn toàn độc lập nhau trong một hệ thống thống nhất, nên chắc chắn tồn tại tập tự phụ thuộc.

Đến đây vấn đề chỉ còn là : làm sao biết được ôtômat có nhớ nào có tập biến mã hóa trạng thái là tập tự phụ thuộc, còn ôtômat nào thì tập biến trạng thái của nó không tạo ra được tập tự phụ thuộc. Đối với một ôtômat không tạo ra được tập tự phụ thuộc cách mã hóa chỉ là làm theo phương pháp mã hóa ngẫu nhiên theo ý của người thiết kế mà thôi. Để tìm được hay khẳng định được ngay từ đầu ôtômat cần phải thiết kế có tồn tại tập tự phụ thuộc hay không chúng ta dựa vào định lý sau :

Định lý 4.4 : Cho một ôtômat M có p trạng thái với số lượng biến trạng thái cần dùng để mã hóa là $k = [\log_2 p]$. Sẽ tồn tại một kiểu mã hóa trạng thái cho M với k biến nhị phân sao cho i biến trạng thái tiếp theo ($1 \leq i \leq k$) chỉ phụ thuộc vào i biến trạng thái hiện tại tương ứng, nếu và chỉ nếu có một phân hoạch thế nhận dạng tất cả các trạng thái có cùng mã nhị phân từ i bít đó.

Nói một cách khác : ôtômat M có p trạng thái với số lượng biến trạng thái cần dùng để mã hóa là $k = [\log_2 p]$. Sẽ tồn tại một kiểu mã hóa trạng thái cho M đủ để tạo ra một tập tự phụ thuộc nếu và chỉ nếu tồn tại một phân hoạch thế trong tập trạng thái S của ôtômat M. Như vậy sự tồn tại một phân hoạch thế (PHT) là điều kiện tiên quyết khẳng định ôtômat đó có tồn tại tập tự phụ thuộc hay không?

Ví dụ : ta có một phân hoạch thế ở dạng :

$$\Pi = (\overline{1,2} ; \overline{3,4} ; \overline{5,6})$$

Ta thấy ôtômat có 6 trạng thái nên số lượng biến trạng thái cần dùng để mã hóa là :

$$k = \left[\log_2 6\right] = 3$$

Ta cần có ba biến (y_1, y_2, y_3) dùng để mã hóa, từ đó ta chọn i = 2 (y_1 y_2) dùng để mã hóa cho các khối của phân hoạch thế, chẳng hạn ta mã hóa khối như sau :

Khối 1 mã là 00

2 01

3 10

Còn biến cuối cùng (y_3) ta dùng để mã hóa cho các phần tử của khối, vì trong mỗi khối có 2 phần tử.

Vậy ta có bảng mã hóa trạng thái cho từng trạng thái của ôtômat là :

y_1 y_2 y_3		
1 = 0 0 0	3 = 0 1 0	5 = 1 0 0
2 = 0 0 1	4 = 0 1 1	6 = 1 0 1

Ví dụ : Có một phân hoạch thế có dạng

$$\Pi = (\overline{1,2,3} ; \overline{4,5,6})$$

Trong trường hợp này do phân hoạch thế đã cho có hai khối, cho nên ta có thể dùng một biến (y_1) để mã hóa cho khối. Còn hai biến còn lại (y_2 y_3) dùng để mã hóa cho các phần tử trong 1 khối, vì mỗi khối chỉ có 3 phần tử. Chẳng hạn ta mã hóa cho phần tử như sau : phần tử thứ 1 mã hóa là 00

2 01

3 10

Vậy ta sẽ có bảng mã hóa (đặt tên) trạng thái cho từng trạng thái của phân hoạch thế đã cho là

	y_1	y_2	y_3					
1	=	0	0	0	4	=	1	0	0
2	=	0	0	1	5	=	1	0	1
3	=	0	1	0	6	=	1	1	0

Qua cách mã hóa cho một phân hoạch thế (PHT) như vậy ta thấy có thể dùng một tập biến để mã hóa cho các khối, và một tập biến khác sẽ được dùng để mã hóa cho các phần tử trong một khối. Khi đó tập mã hóa cho các khối sẽ tạo ra một tập tự phụ thuộc được dùng để quản lý khối, và tập biến này hoàn toàn không có liên quan gì tới tập biến trạng thái dùng để mã hóa cho phần tử của khối. Tương tự như vậy tập biến trạng thái dùng để mã hóa cho phần tử của khối cũng tạo ra được một tập tự phụ thuộc dùng để quản lý các phần tử của khối, và tập biến này cũng hoàn toàn không liên quan gì tới tập biến dùng để quản lý các khối. Như vậy chúng ta có hai tập biến tự phụ thuộc dùng quản lý các khối và tập biến trạng thái dùng để quản lý các phần tử của khối, hai tập biến trạng thái đó hoàn toàn không có liên quan gì đến nhau. Nhưng khi gép chúng nó lại với nhau ta sẽ có tên chính xác của các trạng thái của ôtômat M cũng như các trạng thái có mặt trong tập phân hoạch thế.

4.5.3. PHƯƠNG PHÁP MÃ CHO ÔTÔMAT DÙNG MỘT PHÂN HOẠCH THẾ

Như ta đã biết là một ôtômat có nhớ có thể có cả một dàn phân hoạch thế (D_M), tức là sẽ có rất nhiều các (PHT). Nhiệm vụ của phương pháp mã hóa này là : tìm kiếm sao cho chỉ dùng một phân hoạch thế đủ để mã hóa cho toàn bộ các trạng thái của ôtômat có nhớ, đồng thời phải tạo ra được tập tự phụ thuộc.

Định lý 4.5. Nếu tồn tại một phân hoạch thế (PHT) không tầm thường của ôtômat M sao cho thỏa mãn được quan hệ :

$$\left[\log_2 |\Pi|\right] + \left[\log_2 m(\Pi)\right] = \left[\log_2 p\right] = k$$

Thì tồn tại một cách mã hóa trạng thái cho ôtômat M với số lượng tối thiểu các biến trạng thái cần dùng để mã hóa đủ để tạo ra một tập tự phụ thuộc.

Trong đó : $|\Pi|$ là số lượng các khối của phân hoạch Π

$m(\Pi)$ là số lượng trạng thái của khối lớn nhất của phân hoạch Π.

p là số lượng trạng thái của ôtômat M

$k = [\log_2 p]$ là số biến trạng thái cần dùng để mã hóa cho ôtômat.

Ngoài ra ta có các nhận xét thêm là : $\left[\log_2 |\Pi|\right]$ là số biến trạng thái cần thiết dùng để mã hóa cho các khối của phân hoạch Π, còn $\left[\log_2 m(\Pi)\right]$ là số lượng biến trạng thái cần dùng để mã hóa cho các phần tử của khối lớn nhất trong phân hoạch

Π. Như vậy số lượng biến cần thiết để mã hóa cho khối cộng (ghép) với số lượng biến cần thiết để mã hóa cho các phần tử của khối sẽ trở thành số lượng biến trạng thái cần thiết dùng để mã hóa cho cả ôtômat M.

Chứng minh : Do phân hoạch thế Π có $|\Pi|$ các khối nên số biến trạng thái cần thiết để mã hóa cho các khối sẽ là

$$\left\lceil \log_2 |\Pi| \right\rceil \text{ với } r < k$$

Như vậy ta có r biến để mã hóa cho số lượng khối của phân hoạch Π $(y_1, y_2, ..., y_r)$. Đồng thời vì phân hoạch Π là phân hoạch thế (PHT) cho nên nó thỏa mãn tính chất của một phân hoạch thế là

Tại t_0 : $s_i \equiv s_j$ (Π) thì trạng thái tiếp theo

tại t_1 : $f(s_i, x) \equiv (s_j, x)$ (Π) $\forall x \in X$

Cho nên trạng thái tiếp theo vẫn nằm trong cùng một khối của trạng thái trước đó, thỏa mãn quan hệ tạo ra tập tự phụ thuộc với tập biến dùng để mã hóa :

$$Y = (y_1, y_2, ..., y_r, y_{r+1}, ..., y_k)$$

Đồng thời ta có số lượng biến trạng thái cần dùng để mã hóa cho các phần tử trong khối lớn nhất (đủ để mã hóa cho các phần tử của khối bé hơn) tính theo quan hệ :

$$s = \left\lceil \log_2 m(\Pi) \right\rceil$$

Do đó giữa các biến cần dùng để mã hóa chúng có quan hệ

$$r + s = k$$

Tức là số lượng biến cần dùng để mã hóa cho các khối cộng với số lượng biến cần dùng để mã hóa cho phần tử của khối bằng số lượng biến cần thiết dùng để mã hóa cho cả ôtômat M. Tập số lượng biến cần dùng để mã hóa là :

$$Y = (y_1, y_2, \cdots, y_r, y_{r+1}, y_{r+2}, \cdots, y_{r+s})$$

$$Y = (y_1, y_2, \cdots, y_k)$$

Do dàn phân hoạch thế D_M sẽ có nhiều các phân hoạch thế khác nhau cho nên bất cứ phân hoạch thế nào thỏa mãn được định lý 4.5, đều có thể dùng phân hoạch thế đó để mã hóa được cho ôtômat, đồng thời cho phép tạo ra được tập tự phụ thuộc. Ngược lại nếu trong dàn phân hoạch thế D_M ta không tìm được bất kỳ một phân hoạch thế nào thỏa mãn yêu cầu của định lý 4.5, thì chắc chắn việc mã hóa cho ôtômat đó không thể tạo ra tập tự phụ thuộc. Sau đây chúng ta sẽ đưa ra ví dụ minh họa cho định lý 4.5.

Ví dụ : Cho ôtômat có nhớ hoạt động theo bảng chức năng như sau. Hãy mã hóa trạng thái cho ôtômat đó theo phương pháp dùng một phân hoạch thế đủ để tạo ra một tập tự phụ thuộc (mạch gọn đẹp), ôtômat hoạt động theo bảng chức năng chỉ ra trên hình 4.47.

S \ x_1x_2	00	01	11	10	Z 00	Z 01	Z 11	Z 10
1	1	4	3	1	1	0	1	1
2	1	4	4	1	1	1	0	1
3	1	4	5	8	0	0	0	1
4	1	4	6	8	0	0	1	0
5	8	8	7	8	0	1	0	0
6	8	8	8	8	0	0	1	1
7	1	1	1	1	1	0	0	0
8	1	1	1	1	1	1	0	0

Hình 4.47. Bảng chức năng một ôtômat.

Trước hết nhìn vào bảng chức năng hình 4.47 ta thấy ôtômat này không có phân hoạch thế tương đương (PHTTĐ) vì đáp ứng ra Z của nó có các giá trị theo các hàng không giống nhau. Từ đó chúng ta có thể kết luận bảng chức năng của ví dụ đưa ra đó là một ôtômat đã được rút gọn (ôtômat min). Nhiệm vụ của chúng ta lúc này là mã hóa trạng thái cho ôtômat đó.

Trình tự thực hiện mã hóa :

- Tìm phân hoạch thế cho ôtômat (dàn PHT D_M)
- Từ tập các phân hoạch thế tìm ra một phân hoạch thế thỏa mãn yêu cầu của định lý 4.5.

Trước hết là tìm tất cả các phân hoạch thế nhỏ nhất của ôtômat, sau đó dựa vào các phân hoạch thế nhỏ nhất (các phần tử sinh cơ bản) ta vẽ được dàn phân hoạch thế D_M. Từ dàn phân hoạch thế D_M ta thu được các phân hoạch thế của ôtômat đã cho là :

$$\Pi_1 = (\overline{1,4,5,8} \ ; \ \overline{2,3,6,7})$$

$$\Pi_2 = (\overline{1,2} \ ; \ \overline{3,4} \ ; \ \overline{5,6} \ ; \ \overline{7,8})$$

Đó là hai phân hoạch thế của ôtômat đã cho có thể dùng chúng mã hóa cho các trạng thái của ôtômat nếu như chúng thỏa mãn được các quan hệ của định lý.

Chúng ta chọn $\Pi_1 = (\overline{1,4,5,8} \ ; \ \overline{2,3,6,7})$ rồi kiểm tra theo định lý 4.5 xem nó có thể hóa được cho ôtômat hay không.

Trước tiên do số lượng trạng thái của ôtômat là 8 (p = 8) cho nên số biến tối thiểu cần thiết để mã hóa cho ôtômat này là :

$$k = \left[\log_2 p\right] = \left[\log_2 8\right] = 3 \rightarrow (y_1 \ y_2 \ y_3)$$

Số lượng các khối của phân hoạch thế Π_1 là 2 $\left(|\Pi_1| = 2\right)$ vậy số lượng biến cần để mã hóa cho số khối của phân hoạch thế Π_1 sẽ là :

$$r = \left[\log_2 |\Pi_1|\right] = \left[\log_2 2\right] = 1$$

Trong mỗi một khối (hay là khối lớn nhất) có số lượng các phần tử của khối là 4 ($m(\Pi_1) = 4$), vậy số lượng biến trạng thái cần thiết để mã hóa cho số phần tử của khối sẽ là :

$$s = \left[\log_2 m(\Pi_1)\right] = \left[\log_2 4\right] = 2$$

Vậy phân hoạch thế Π_1 thỏa mãn được yêu cầu của định lý là :

$$r = \left[\log_2 |\Pi_1|\right] + \left[\log_2 m(\Pi_1) = \left[\log_2 p\right] = k\right.$$

thay : $r + s = k$

$1 + 2 = 3$

Như vậy ta có thể dùng biến trạng thái (y_1) để mã hóa cho số khối của phân hoạch Π_1, còn các biến trạng thái là (y_2 y_3) được dùng để mã hóa cho các phần tử trong từng khối của phân hoạch thế Π_1, và chúng tạo được tập tự phụ thuộc của các biến dùng để mã hóa các trạng thái S của ôtômat để ra. bảng mã hóa (đặt tên) cụ thể cho từng trạng thái của ôtômat đã cho được chỉ ra ở bảng của hình 4.48.

Bảng mã hóa dùng một phân hoạch thế.

y_2 y_3 \ y_1	0	1
0 0	1	2
0 1	4	3
1 0	5	6
1 1	8	7

Hình 4.48. Bảng mã hoá cụ thể trạng thái.

Nhìn vào bảng hình 4.48 cột y_1 của bảng lấy hai giá trị là (0 và 1) để mã hóa cho hai khối riêng biệt, còn các hàng (y_2 y_3) thì mã hóa cho từng phần tử trong một khối của phân hoạch Π_1. Từ đó chúng ta có được tọa độ hay giá trị mã hóa (tên gọi) cụ thể của từng trạng thái của ôtômat M đã cho là :

1 - 0 0 0 5 - 0 1 0

2 - 1 0 0 6 - 1 1 0

3 - 1 0 1 7 - 1 1 1

4 - 0 0 1 8 - 0 1 1

Với mỗi một trạng thái của ôtômat ta có được một quan hệ mã hóa (một tên gọi), nhưng đặc điểm chính và quan trọng ở đây là cách mã hóa này tạo được tập tự phụ thuộc.

Với cách làm tương tự chúng ta cũng có thể kiểm tra theo điều kiện của định lý cho phân hoạch thế $\Pi_2 = (\overline{1,2}; \overline{3,4}; \overline{5,6}, \overline{7,8})$ xem phân hoạch đó có thể dùng để

mã hóa cho ôtômat đưa ra hay không? Trong trường hợp cụ thể này chúng ta cũng thấy phân hoạch thế Π_2 cũng thỏa mãn được yêu cầu của định lý 4.5, cho nên có thể dùng phân hoạch đó để mã hóa cho ôtômat M đề ra, với hai biến (y_1 y_2) dùng để mã hóa cho các khối, còn biến (y_3) dùng để mã hóa cho các phần tử trong một khối.

Đặc điểm của cách mã hóa dùng một phân hoạch thế là mã hóa trạng thái cho ôtômat chặt chẽ hơn mã hóa tùy ý bất kỳ ở chỗ : trạng thái giữa các cột (hay giữa các khối) không thể trao đổi hay hoán vị sang nhau, nhưng trong từng khối từng cột thì việc thay đổi vị trí của các trạng thái vẫn thực hiện được, điều này sẽ lại tạo ra cách mã hóa trạng thái kiểu khác, hay có mạch điện kiểu khác (vẫn là tập tự phụ thuộc). Để khắc phục điều đó dẫn ta tới một cách mã hóa chặt chẽ hơn, đó là mã hóa tối ưu hay còn được gọi là mã hóa trạng thái cho ôtômat dùng một tập các phân hoạch thế (PHT).

4.5.4. PHƯƠNG PHÁP MÃ HÓA CHO ÔTÔMAT DÙNG MỘT TẬP PHÂN HOẠCH THẾ

Phần này chúng ta sẽ xét tới điều kiện để hóa trạng thái với số lượng biến tối thiểu dùng một tập các phân hoạch thế. Khi chúng ta dùng một tập phân hoạch thế để mã hóa trạng thái cho ôtômat sẽ dễ dàng tìm được tập tối thiểu các biến trạng thái ít phụ thuộc lẫn nhau hơn, nên việc thực hiện ôtômat của dễ dàng và gọn gàng hơn. Tuy nhiên khi ta thực hiện mã hóa dùng một tập phân hoạch thế sẽ xuất hiện một trường hợp nếu không chú ý sẽ dẫn tới tình trạng số biến trạng thái sử dụng sẽ bị lớn hơn số lượng biến tối thiểu cần thiết để mã hóa cho ôtômat theo cách thông thường.

Ví dụ cho ôtômat có các phân hoạch thế là

$$\Pi_1 = (\overline{1,2,3}\ ;\ \overline{4,5}\ ;\ \overline{6,7})$$

$$\Pi_2 = (\overline{1,2,3,4}\ ;\ \overline{5,6,7})$$

Đây là một ôtômat có 7 trạng thái, như vậy số lượng biến trạng thái tối thiểu cần thiết theo cách thông thường sẽ là 3 $\left(\left[\log_2 7\right] = 3\right)$, như vậy chỉ cần các biến (y_1 y_2 y_3) là để mã hóa trạng thái cho ôtômat.

Như do Π_1 là phân hoạch thế nên muốn tạo ra tập con tự phụ thuộc (mã hóa cho khối), ở đây Π_1 có 3 khối cho nên để mã hóa cho các khối của Π_1 ta cần có hai biến để mã hóa (y_1 y_2). Đồng thời ở đây khối lớn nhất của Π_1 có 3 trạng thái hay có 3 phần tử, vậy muốn mã hóa được cho 3 phần tử của khối lớn nhất ta cần phải có 2 biến là (y_3 y_4). Như vậy (y_1 y_2) dùng để mã hoá cho khối của Π_1 còn (y_3 y_4) sẽ dùng để mã hóa cho các phần tử của khối lớn nhất của phân hoạch thế Π_1, vậy tổng lại chúng ta cần có 4 biến để mã hóa (y_1 y_2 y_3 y_4). Trong khi nếu mã hóa như bình thường thì ta chỉ cần có 3 biến để mã hóa. Ta biết mỗi một biến dùng để mã hóa sẽ là một bít thông tin hay một mạch trigơ, cho nên mã hóa cho ôtômat theo phân hoạch thế Π_1 có thể làm cho mạch lớn hơn so với mạch của ôtômat đó mà mã hóa theo cách thông thường. Từ đó để thực hiện được việc mã hóa dùng phân hoạch thế sao cho có số lượng biến là tối thiểu chúng ta đưa ra phương pháp dùng một tập phân hoạch thế, đảm bảo giảm nhỏ được sự

phụ thuộc qua lại giữa các biến trạng thái đồng thời vẫn dùng với số lượng biến ít nhất để mã hóa. Điều kiện để tìm ra được tập phân hoạch thế mã hóa cho ôtômat được chỉ ra ở định lý sau.

Định lý 4.6 : Cho một ôtômat có nhớ đã được rút gọn M với p trạng thái, và một tập A không trống hữu hạn của m phân hoạch thế. Tồn tại một cách mã hóa với k biến $\left(k = \left\lceil \log_2 p \right\rceil\right)$ trạng thái mà tập con của các biến trạng thái mã hóa khối cho từng phân hoạch của A là tự phụ thuộc nếu và chỉ nếu thỏa mãn :

1. $K_{\Pi_i} + \mu_{\Pi_i} = k \qquad \forall \Pi_i \in A$

2. $\sum_{i=1}^{m} K_{\Pi_i} = K_{\Pi^m}$

3. $K_{\Pi^m} + \mu_{\Pi^m} = k$

Ở đây : $k = \left\lceil \log_2 p \right\rceil$: là tập biến tối thiểu cần để mã hóa cho ôtômat.

$K_{\Pi_i} = \left\lceil \log_2 |\Pi_i| \right\rceil$: là tập biến tối thiểu cần để mã hóa cho khối của phân hoạch thứ Π_i. $\quad 1 \leqslant i \leqslant m$

$\mu_{\Pi_i} = \left\lceil \log_2 m(\Pi_i) \right\rceil$: là tập biến tối thiểu cần thiết để mã hóa cho phần tử thuộc khối lớn nhất của phân hoạch thứ Π_i.

$\Pi^m = \prod_{i=1}^{m} \Pi_i$: là phân hoạch tích (kết quả tích của m phân hoạch của tập A)

$K_{\Pi^m} = \left\lceil \log_2 |\Pi^m| \right\rceil$: là tập biến cần thiết để mã hóa cho các khối của phân hoạch tích

$\mu_{\Pi^m} = \left\lceil \log_2 m(\Pi^m) \right\rceil$: là tập biến cần thiết để mã hóa cho các phần tử của khối của phân hoạch tích.

Nhìn vào từng điều kiện của định lý 4.6 ta có nhận xét :

- *Điều kiện 1* đảm bảo cho mỗi một phân hoạch $\Pi_1 \in A$ chỉ cần dùng k biến trạng thái để mã hóa là đủ để tạo ra mã nhị phân độc lập cho từng trạng thái của ôtômat trang Π_i.

- *Điều kiện 2* xác định là số lượng biến dùng để mã hóa các phân hoạch của tập A các phân hoạch thế đúng bằng số lượng biến tối thiểu để mã hóa cho các trạng thái của phân hoạch tích Π^m.

Ví dụ cho : $\Pi_1 = (\overline{1,2} ; \overline{3} ; \overline{4})$

$\Pi_2 = (\overline{1} ; \overline{2} ; \overline{3,4})$

Phân hoạch tích của chúng được tính như sau :

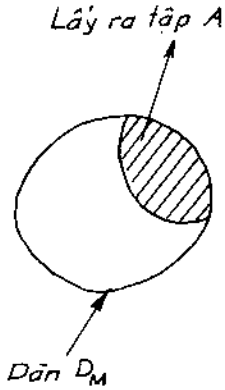
$$\Pi^2 = \Pi_1 . \Pi_2 = (\overline{1} ; \overline{2} ; \overline{3} ; \overline{4})$$

Ta tính kết quả theo cách tính của tính Đêcác.

Điều đó có nghĩa là không cần phải yêu cầu thêm các biến khác (ngoài số lượng tối thiểu biến k) để mã hóa cho tập A.

- *Điều kiện 3* đảm bảo không cần dùng thêm các biến trạng thái để tạo nên mã cho các trạng thái của khối lớn nhất của phân hoạch tích Π^m. Nhận thấy số lượng biến trạng thái cần dùng để mã hóa cho phân hoạch tích chính bằng số lượng biến tối thiểu dùng để mã hóa cho ôtômat M.

Ví dụ : Cho ôtômat hoạt động theo bảng chức năng được chỉ ra trên hình 4.49. Dùng một tập phân hoạch thế để mã hóa cho ôtômat với số lượng biến cần dùng để mã hóa là tối thiểu đủ để tạo ra một tập tự phụ thuộc.



Z \ X	0	1	Z
A	D	C	0
B	F	C	0
C	E	B	0
D	B	E	1
E	A	D	0
F	C	D	0

Hình 4.49. Cho trước một ôtômat.

Chúng ta thực hiện mã hóa dùng một tập phân hoạch thế theo các bước sau :

- Vẽ dàn phân hoạch thế D_M để biết ôtômat có tồn tại tập phân hoạch thế hay không.

- Từ dàn phân hoạch thế chọn ra một tập A các phân hoạch thế.

- Kiểm tra xem tập phân hoạch thế A có thỏa mãn các điều kiện của định lý (4.6) hay không.

- Nếu thỏa mãn định lý thì mã hóa cho tập A theo các biến cho từng khối trong từng phân hoạch thế của tập A.

Qua các bước ở trên chúng ta sẽ tìm được hai phân hoạch thế của ôtômat đã cho là :

$$\Pi_1 = (\overline{A,B,C} ; \overline{D,E,F})$$

$$\Pi_2 = (\overline{A,F} ; \overline{B,E} ; \overline{C,D})$$

Từ hai tập phân hoạch thế Π_1, Π_2 tìm được từ ôtômat đã cho, ta chọn tập phân hoạch thế A sẽ được dùng để mã hóa cho ôtômat là :

$$A = (\Pi_1 , \Pi_2)$$

Do có hai phân hoạch thế trong A cho nên m = 2. Bước tiếp theo ta kiểm tra tập A các phân hoạch thế đó xem có thỏa măn định lý hay thỏa măn điều kiện để mã hóa hay không.

Từ ôtômat đã cho ta thấy số lượng trạng thái của ôtômat p = 6 (ôtômat có 6 trạng thái).

Ta tính được các tham số của tập A theo định lý như sau :

$$k = \left[\log_2 p\right] = \left[\log_2 6\right] = 3$$

$$K_{\Pi_1} = \left[\log_2 |\Pi_1|\right] = \left[\log_2 2\right] = 1$$

$$K_{\Pi_2} = \left[\log_2 |\Pi_2|\right] = \left[\log_2 3\right] = 2$$

$$\mu_{\Pi_1} = \left[\log_2 m(\Pi_1)\right] = \left[\log_2 3\right] = 2$$

$$\mu_{\Pi_2} = \left[\log_2 m(\Pi_2)\right] = \left[\log_2 2\right] = 1$$

$$\Pi^2 = \Pi_1 . \Pi_2 = (\overline{A,B,C} ; \overline{D,E,F}) . (\overline{A,B} ; \overline{C,D} ; \overline{E,F}) =$$

$$= (\overline{A} ; \overline{B} ; \overline{C} ; \overline{D} ; \overline{E} ; \overline{F}) - \text{có 6 khối mà mỗi khối có 1 phần tử.}$$

$$K_{\Pi^2} = \left[\log_2 |\Pi^2|\right] = \left[\log_2 6\right] = 3$$

$$\mu_{\Pi^2} = \left[\log_2 m(\Pi^2)\right] = \left[\log_2 1\right] = 0$$

Thay các giá trị tính toán được vào các điều kiện của định lý :

1. $K\Pi_1 + \mu\Pi_1 = 1 + 2 = 3 = k$

   $K\Pi_2 + \mu\Pi_2 = 1 + 2 = 3 = k$

2. $K\Pi_2 + \mu\Pi_1 = 1 + 2 = 3 = K\Pi^2$

3. $K\Pi^2 + \mu\Pi^2 = 3 + 0 = 3 = k.$

Như vậy chúng ta thấy cả 3 điều kiện của định lý đều đã được thỏa măn, từ đó có thể nói là dùng tập A các phân hoạch thế có thể mã hóa cho ôtômat, với số lượng biến dùng để mã hóa là ít nhất, đồng thời đủ để tạo ra tập tự phụ thuộc.

Tiếp theo ta có cách mã hóa (đặt tên) cụ thể cho từng trạng thái của ôtômat như sau : Dùng một biến trạng thái (y_1) để mã hóa cho các khối của Π_1 và hai biến trạng thái (y_2 y_3) dùng để mã hóa cho các khối của phân hoạch Π_2. Từ đó ta có bảng mã hóa (đặt tên) cụ thể chỉ ra trên hình 4.50.

Bảng đặt tên cụ thể.

y_2y_3 \ y_1	0	1	
0 0	A	F	(A , F)
0 1	B	E	($\overline{B}$, $\overline{E}$)
1 0	C	D	($\overline{C}$, $\overline{D}$)
1 1	–	–	

Hình 4.50. Bảng đặt tên trạng thái.

$\Pi_1 = (\overline{A, B, C}\ ;\ \overline{D, E, F})$ ta dùng (y_1) để mã hóa cho các khối của Π_1, cụ thể thấy từng cột trong bảng chính là các trạng thái trong từng khối của Π_1 ($y_1 = 0$ có các khối (A, B, C), $y_1 = 1$ ứng với các khối (D, E, F))

Tương tự với hai biến mã hóa $(y_2\ y_3)$ ta mã hóa cho từng khối của phân hoạch thế $\Pi_2 = (\overline{A, F}\ ;\overline{B, E}\ ;\overline{C, D})$. Như vậy chúng ta dùng các biến trạng thái chỉ mã hóa cho từng khối của các phân hoạch thế (Π_1, Π_2). Từ đó chắc chắn tạo ra được tập tự phụ thuộc với số lượng biến dùng để mã hó là ít nhất, nhưng vẫn tìm được mã hay tên gọi của từng trạng thái của ôtômat là :

A = 000 B = 001

C = 010 D = 110

E = 101 F = 100

Ứng với một tập các phân hoạch thế A ta luôn có duy nhất một cách mã hóa cho các trạng thái của ôtômat, vì giữa hai cột trạng thái không hoán vị được, đồng thời giữa các hàng trạng thái cũng không thể hoán vị được do chúng phải nằm trong từng khối của Π_1 và Π_2. Cách mã hóa ta thu được là duy nhất nên được gọi là cách mã hóa tối ưu. Với cách mã hóa như vậy dùng một tập phân hoạch thế đủ để đảm bảo tạo ra khối tự phụ thuộc với số lượng biến mã hóa là nhỏ nhất hay ít nhất.

Nếu ta không tìm được tập A nào đó thỏa mãn điều kiện của định lý (4.6) thì chỉ còn cách là dùng mã hóa (đặt tên) ngẫu nhiên cho ôtômat được mà thôi.

Tóm lại với một ôtômat có nhớ chúng ta có thể có ba cách để mã hóa hay đặt tên cho các trạng thái là : Mã hóa ngẫu nhiên, mã hóa dùng một phân hoạch thế và mã hóa dùng một tập phân hoạch thế. Tùy theo từng trường hợp cụ thể khi thiết kế xây dựng ôtômat có nhớ ta có thể áp dụng một trong ba cách mã hóa đó.

§4.6 PHÂN GIẢI ÔTÔMAT CÓ NHỚ

4.6.1. KHÁI NIỆM

Trước hết chúng ta cần hiểu phân giải ôtômat có nhớ là gì và phân giải ôtômat có nhớ khi nào và để làm gì? Việc thực hiện phân giải ôtômat có nhớ thông thường chỉ xảy ra khi chúng ta gặp phải một ôtômat quá lớn,. nó có số lượng trạng thái rất lớn tới mức chúng ta không thể thực hiện tổng hợp trực tiếp nổi, vì việc thực hiện ôtômat sẽ đòi hỏi một khối lượng tính toán rất lớn cũng như cần một hệ thống mạch điện rất lớn. Trong trường hợp đó chúng ta phải tìm cách phân giải hay phân chia nó ra thành các ôtômat con thành phần nhỏ hơn, (tạo thành các môđun), sau đó nối chúng lại hay ghép chúng lại thành ôtômat cần phải xây dựng. Khi có được các ôtômat con thành phần nhỏ hơn sẽ cho phép ta thực hiện lắp ráp nó một cách dễ dàng hay còn có thể sử dụng kế thừa các ôtômat đã được nó sẵn, đồng thời có thể tạo ra các môđun từ đó cho phép xây dựng nhà máy để sản xuất hàng loạt ôtômat cơ bản.

4.6.2. PHÂN GIẢI NỐI TIẾP ÔTÔMAT CÓ NHỚ

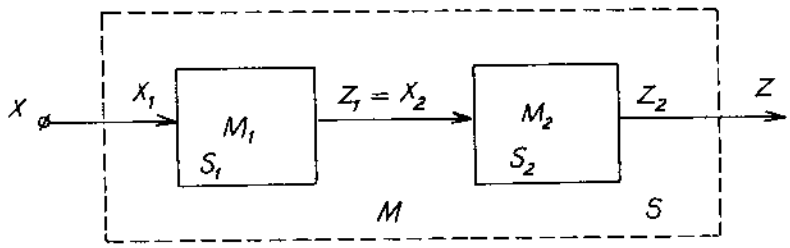
Việc phân giải nối tiếp ôtômat có nhớ liên quan tới hai bài toán : Bài toán thuận là cho trước hai ôtômat M_1 và M_2 ta nối tiếp chúng lại sẽ tạo ra một ôtômat M lớn hơn ($M_1 \ominus M_2 = M$). Bài toán thứ hai là bài toán ngược cho trước một ôtômat M ta phải phân chia ôtômat đó ra làm hai ôtômat con thành phần M_1 và M_2, sao cho khi nối tiếp M_1 và M_2 sẽ tạo ra ôtômat M ($M = M_1 \ominus M_2$).

4.6.2.1. Bài toán thuận của phân giải nối tiếp

Nhiệm vụ của bài toán là :

Cho trước hai ôtômat con M_1 và M_2, nối tiếp chúng lại với nhau sẽ tạo ra một ôtômat M lớn hơn, ta phải tìm ra chức năng của ôtômat M lớn hơn được tạo ra này. Sơ đồ biểu diễn nối liên tiếp hai ôtômat được mô tả trên hình 4.51.

Định nghĩa 4.7 : Cho trước hai ôtômat có nhớ $M_1 = (X_1, S_1, Z_1, f1, g_1)$ và $M_2 = (X_2, S_2, Z_2, f_2, g_2)$ nối tiếp M_1 và M_2 tạo ra ôtômat M như sau :



Hình 4.51. Nối tiếp hai ôtômat.

$$M = (X, S, Z, f, g)$$

$$M_1 \ominus M_2 = M = (X_1, (S_1 \times S_2), Z_2, f, g)$$

trong đó : $X = X_1$

$S = (S_1 \times S_2)$- Tích Đêcác S_1 và S_2

$Z = Z_2$

$Z_1 = X_2$ đầu ra M_1 là đầu vào của M_2

$$s(t+1) = f(s(t), x) = f(s(t), x_1) =$$

$$= \{f_1(s(t), x_1 ; f_2(s(t), x_2)\} =$$

$$= \{f_1(s(t), x_1) ; f_2(s(t), g_1(s(t), x_1))\}$$

$$= \{s_1(t+1) ; s_2(t+1)\}$$

$$z(t) = g(s(t), x) = g_2(s(t), x_2) =$$

$$= g_2(s(t), g_1(s(t), x_2)) = z_2(t)$$

Từ các quan hệ tính được từ trong định nghĩa, ta thấy mọi tham số của ôtômat M đều được xác định từ các tham số cho trước của hai ôtômat ban đầu M_1 và M_2. Đồng thời việc nối tiếp hai ôtômat thành phần cho trước thành một ôtômat M lớn hơn không cần có thêm điều kiện giàng buộc nào cả.

4.5.2.2. Bài toán ngược của phân giải nối tiếp

Nhiệm vụ đặt ra ở đây là từ một ôtômat M cho trước chúng ta tìm cách phân giải nó thành các ôtômat con thành phần là M_1 và M_2 sao cho khi nối tiếp chúng lại với nhau thì thực hiện được chức năng của ôtômat M. Phần này chúng ta phải đưa ra điều kiện cho phép ta biết được một ôtômat khi nào thì có thể thực hiện được phân giải nối tiếp khi nào không thực hiện được phân giải nối tiếp, vì không phải cứ bất kỳ một ôtômat có nhớ nào cũng thực hiện được phân giải nối tiếp.

Định nghĩa 4.8 : Ôtômát M_1 nối tiếp với M_2 được gọi là phân giải nối tiếp của ôtômat M nếu và chỉ nếu M_1 nối tiếp với M_2 thực hiện ôtômat M ($M_1 \textcircled{\rightarrow} M_2 = M$). Phân giải đó là không tầm thường nếu M_1 và M_2 có số lượng trạng thái ít hơn trạng thái của M.

Điều kiện cần và đủ để có thể thực hiện được phân giải nối tiếp một ôtômat có nhớ thành hai ôtômat con thành phần nếu thỏa mãn định lý sau :

Định lý 4.7 : Ôtômát M có phân giải nối tiếp không tầm thường nếu và chỉ nếu tồn tại một phân hoạch thế không tầm thường Π trên tập trạng thái S của ôtômat M.

Ta thấy nội dung của định lý phát biểu có liên quan tới phân hoạch thế của ôtômat có nhớ, đây là một sự nhóm họp hay hoạt động đặc biệt của ôtômat có nhớ. Do là hoạt động đặc biệt nên có thể có ôtômat có nhớ hoạt động của nó không tạo ra được phân hoạch thế (PHT), nhưng cũng có ôtômat sự hoạt động hay bảng chức năng của nó cho phép tạo ra tập phân hoạch thế. Định lý (4.7) cho phép ta có thể thực hiện phân giải nối tiếp ôtômat có nhớ cũng như cho biết ngay từ đầu ôtômat đó có phân giải được nối tiếp hay không? Sự tồn tại của tập phân hoạch thế chính là điều kiện tiên quyết để thực hiện được phân hoạch nối tiếp một ôtômat có nhớ.

Do đặc điểm hoạt động của phân hoạch thế là trạng thái tiếp theo dưới tác động của tín hiệu vào lại quay về trạng thái trước đó, cho nên ta có thể hiểu hoạt động của phân hoạch thế là ở trong từng bộ từng cặp riêng biệt. Điều đó cho phép ta có thể dựa vào tập phân hoạch thế để xây dựng nên một ôtômat riêng biệt M_1, sau đó tìm cách, xây dựng (tìm ra) được thêm một ôtômat M_2 nào đó nối tiếp với M_1, tạo ra hoạt động của ôtômat M.

Các bước thực hiện phân giải nối tiếp ôtômat có nhớ như sau :

1. Vẽ dàn phân hoạch thế D_M, từ đó tìm ra một phân hoạch thế không tầm thường Π.

2. Từ phân hoạch Π tìm được ta tìm ra một phân hoạch mới τ sao cho thỏa mãn điều kiện $\Pi . \tau = \Phi$.

3. Từ phân hoạch thế Π tạo nên ôtômat M_1, và từ phân hoạch τ tạo ra ôtômat M_2.

Các bước trên có thể tiến hành theo trình tự :

Cho trước :

$$M = (X, S, Z, f, g)$$

Từ ôtômat M đã cho ta tìm ra được một phân hoạch thế Π. Từ phân hoạch Π ta tạo ra ôtômat M_1 như sau :

$$M_1 = (X_1, S_1, Z_1, f_1, g_1)$$

$$M_1 = \left(X, \left\{B_\Pi\right\}, \left\{B_\Pi\right\} \times \left\{X\right\}, f_\Pi, e\right)$$

Trong đó :

$S_1 = \left\{B_\Pi\right\}$: Tập các khối của phân hoạch thế Π

$X = X_1$: Vì là tín hiệu vào của mạch mắc nối tiếp.

$\left\{B_\Pi\right\} \times \left\{X\right\}$: Trạng thái S_1 tích Đêcác với tín hiệu vào X tạo ra đáp ứng ra Z_1

f_Π : Hàm ánh xạ hay hàm chuyển biến trạng thái của M_1, nó có quan hệ

$$f_\Pi\left(\left\{B_\Pi\right\}, X_1\right) = f\left(\left\{B_\Pi\right\}, X\right) \text{ của M đã cho}$$

e : Là hàm ẩn, ánh xạ từ tập tích trực tiếp (tích Đêcác) của $\left\{B_\Pi\right\} \times \left\{X\right\}$ vào chính nó.

$$e = \left(\left\{B_\Pi\right\}, X\right) = \left\{B_\Pi\right\} \times \left\{X\right\}.$$

Cũng từ phân hoạch thế Π ta chọn được ở trên, ta tìm được phân hoạch τ thỏa mãn quan hệ $\tau . \Pi = \Phi$. Sau khi tìm được phân hoạch τ, dựa vào phân hoạch τ tìm được ôtômat M_2 có cấu tạo :

$$M_2 = \left(X_2, S_2, Z_2, f_2, g_2\right)$$

$$M_2 = \left(Z_1, \left\{B_\tau\right\}, Z, f_2, g_2\right)$$

$$M_2 = \left(\left\{B_\Pi\right\} \times \left\{X\right\}, \left\{B_\tau\right\}, Z, f_2, g_2\right)$$

Trong đó :

$S_2 = \left\{B_\tau\right\}$: Là tập các khối của phân hoạch τ

$Z_2 = Z$: Là tập đáp ứng ra của ôtômat M đã cho

f_2 : Là hàm ánh xạ thỏa mãn quan hệ

$$f_2\left(\left\{B_\tau\right\}, \left\{B_\Pi\right\} \times \left\{X\right\}\right) = f\left(B_\tau . B_\Pi, X\right)$$

g_2 : Là hàm ra ánh xạ thỏa mãn quan hệ

$$g_2\left(\left\{B_\tau\right\}, \left\{B_\Pi\right\} \times \left\{X\right\}\right) = g\left(B_\tau . B_\Pi, X\right)$$

$B_\Pi . B_\tau$: Là phân hoạch tích.

Như vậy là mọi tham số của ôtômat M_1 và M_2 đều được tìm ra từ ôtômat M cho trước. Ta có thể kiểm tra lại kết quả đó khi nối tiếp M_1 và M_2 ($M_1 \textcircled{\rightarrow} M_2$) lại với

nhau (như coi M_1, M_2 cho trước ở phần trên) sẽ lại được ôtômat M đã cho ban đầu.

$$M_1 \textcircled{\rightarrow} M_2 = (X_1, B_\Pi \times B_\tau, Z_2, g_2, f_2) =$$

$$= (X, S, Z, f, g) = M$$

Ở đây do điều kiện $\tau \cdot \Pi = \varnothing$ cho nên $B_\Pi \times B_\tau = S$.

Ví dụ : Thực hiện phân giải nối tiếp ôtômat cho ở bảng chức năng trên hình 4.52.

Bảng chức năng

			Z	
	0	1	0	1
X / S				
1	3	1	1	0
2	3	1	0	1
3	2	1	0	1
4	1	3	0	0
5	6	4	1	0
6	5	4	0	1

Hình 4.52. Cho trước một ôtômat.

Bước 1 : Tìm dàn phân hoạch thế của ôtômat đã cho, trên cơ sở tìm ra các phân hoạch thế nhỏ nhất của nó, kết quả thu được :

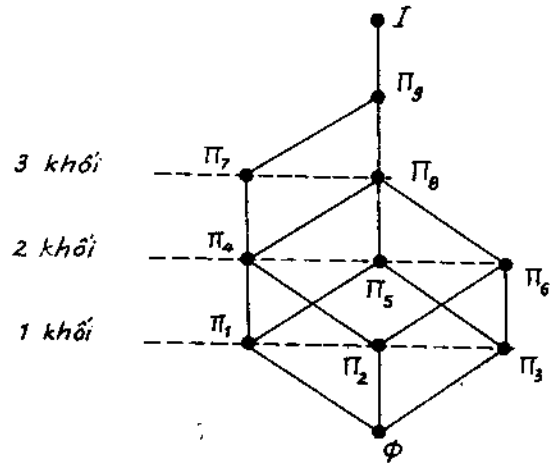
$$\Pi_1 = (\overline{1,2}), \quad \Pi_2 = (\overline{2,3}), \quad \Pi_3 = (\overline{5,6})$$

$$\Pi_7 = (\overline{1,2,3,4})$$

Dựa vào các phân hoạch thế tìm được chúng ta vẽ được dàn phân hoạch thế D_M của ôtômat được chỉ ra trên hình 4.53.

Trong dàn D_M ta có được các phân hoạch thế.

$$\Pi_4 = \Pi_1 + \Pi_2 = (\overline{1,2}\ ;\ \overline{2,3}) = (\overline{1,2,3})$$



Hình 4.53. Dàn phân hoạch thế D_M.

$$\Pi_5 = \Pi_1 + \Pi_3 = (\overline{1,2}\ ;\ \overline{5,6})$$

$$\Pi_6 = \Pi_2 + \Pi_3 = (\overline{2,3}\ ;\ \overline{5,6})$$

$$\Pi_8 = \Pi_4 + \Pi_6 = (\overline{1,2,3}\ ;\ \overline{5,6})$$

$$\Pi_9 = \Pi_7 + \Pi_8 = (\overline{1,2,3,4}\ ;\ \overline{5,6})$$

Dựa vào dàn phân hoạch thế ta chọn phân hoạch $\Pi = \Pi_9$. Do tồn tại phân hoạch thế nên thỏa mãn điều kiện của định lý (4.7), cho phép ta thực hiện được phân giải nối tiếp ôtômat đã cho. Nói cách khác là vì tồn tại phân hoạch thế cho nên phân giải nối tiếp ôtômat cho là thực hiện được.

Ta có phân hoạch thế ban đầu lựa chọn là :

$$\Pi = \Pi_9 = (\overline{1,2,3,4}\ ;\ \overline{5,6}) = \{A,B\} = \{B_\Pi\}$$

Do phân hoạch Π có số lượng phần tử trong một khối lớn nhất là 4, cho nên phân hoạch τ phải có ít nhất là 4 khối thì mới thỏa mãn được quan hệ $\tau . \Pi = \Phi$, bằng cách các phần tử của khối lớn nhất của Π sẽ được phân chia ra thành từng phần tử nằm trong từng khối riêng biệt của τ.

Ta có : $\Pi = (\overline{1,2,3,4}\ ;\ \overline{5,6})$ từ đó suy ra τ

$$\tau = (\overline{1,5}\ ;\ \overline{2,6}\ ;\ \overline{3}\ ;\ \overline{4})$$

Từ hai phân hoạch Π và τ thu được đó chắc chắn chúng sẽ thỏa mãn $\Pi . \tau = \Phi$

$$\Pi . \tau = (\overline{1}\ ;\ \overline{2}\ ;\ \overline{3}\ ;\ \overline{4}\ ;\ \overline{5}\ ;\ \overline{6}) = \Phi$$

Cũng từ hai phân hoạch Π và τ đó, dựa vào chúng có thể tạo ra hai ôtômat M_1 và M_2. Nếu thực hiện mắc nối tiếp chúng lại với nhau sẽ tạo ra ôtômat M ban đầu đã cho, thực hiện được chức năng của ôtômat M. Đồng thời số lượng trạng thái của hai ôtômat M_1 và M_2 là ít hơn ôtômat M ban đầu.

$$\tau = (\overline{1,5}\ ;\ \overline{2,6}\ ;\ \overline{3}\ ;\ \overline{4}) = (a,b,c,d) = \{B_\tau\}$$

Trước hết dựa vào phân hoạch Π thực hiện M_1 từ ôtômat M đã cho như sau :

$$M_1 = \left(X, \{B_\Pi\}, \{B_\Pi\} \times \{X\}, f_\Pi, e\right)$$

$$\Pi = (\overline{1,2,3,4}\ ;\ \overline{5,6}) = \{A,B\} = \{B_\Pi\}$$

Hay có thể viết :

$$M_1 = \left(\{0,1\}, \{A,B\}, \{(A,0),(A,1),(B,0),(B,1)\}, f_\Pi, e\right)$$

Chúng ta có cấu tạo của bảng chức năng của ôtômat M_1 như trên hình 4.54.

Vấn đề ở đây là dựa vào ôtômat M cho trước (là số liệu ban đầu) ta sẽ điền vào bảng trạng thái các trạng thái tiếp theo của M_1 như thế nào.

Bảng chức năng M_1

S_1 \ X_1			Z_1	
	0	1	0	1
A	ⓐ	Ⓐ	α	β
B	B	A	γ	η

Hình 4.54. Ôtômát M_1.

Cách điền bảng M_1 :

Chúng ta sẽ điền theo từng tọa độ trên bảng chức năng M_1 tọa độ đầu tiên (A, 0), ta dựa vào quan hệ :

$$f_\Pi\left(\left\{B_\Pi\right\}, X\right) = f\left(\left\{B_\Pi\right\}, X\right\}$$

$$f_\Pi(A,0) = f(\overline{1,2,3,4}, 0) = \overline{3,3,2,1} \text{ theo M cho trước}$$

$$= \overline{3,3,2,1} = ⓐ \text{ do ký hiệu.}$$

Tọa độ tiếp theo (A, 1), cách làm tương tự trên :

$$f_\Pi(A,1) = f(\overline{1,2,3,4}, 1) = \overline{1,1,1,3} \text{ theo M cho trước}$$

$$= \overline{1,1,1,3} = Ⓐ$$

Với tọa độ tiếp theo (B, 0) cách làm tương tự

$$f_\Pi(B,0) = f(\overline{5,6}, 0) = \overline{4,4} = A$$

Với tọa độ cuối cùng (B, 1)

$$f_\Pi(B, 1) = f(\overline{5,6}, 1) = \overline{6,5} = B$$

Do phân hoạch Π là phân hoạch thế, nên trạng thái tiếp theo dưới tác động của tín hiệu vào (X) luôn luôn hoạt động theo từng bộ từng cặp trạng thái, cho nên đã tạo ra các trạng thái tiếp theo M_1 là A và B dễ dàng điền vào bảng M_1 ở khu vực trạng thái tiếp theo (hàm f_Π).

Cách điền khu vực hàm ra (e) của M_1 : ta dựa vào quan hệ

$$e\left(\left\{B_\Pi\right\}, X\right) = \left\{B_\Pi\right\} \times \left\{X\right\}$$

$$e\left(A, X\right) = \left(A, X\right) = \left(A, 0\right), \left(A, 1\right) = \alpha, \beta$$

$$e\left(B, X\right) = \left(B, X\right) = \left(B, 0\right), \left(B, 1\right) = \gamma, \eta$$

Do đầu ra Z_1 của ôtômat M_1 ta không biết chính xác (cho nên gọi e là hàm ẩn số), mà ta chỉ biết chính xác là đầu ra của M_1 là đầu vào của ôtômat M_2 ($Z_1 = X_2$) do mắc nối tiếp, cho nên tọa độ ở đầu ra Z_1 hay đáp ứng ra của ôtômat M_1 chúng ta cho các ẩn số là (A, 0) = α, (A, 1) = β, (B, 0) = γ, (B, 1) = η. Như vậy coi như điền xong hay tạo xong ôtômat M_1, bảng chức năng của ôtômat M_1 được tạo ra từ phân hoạch Π và ôtômat M cho trước.

Tiếp theo chúng ta dựa vào phân hoạch τ để xây dựng nên ôtômat M_2, ở đây ta chỉ cần có lưu ý là các giá trị (α, β, γ, η) đáp ứng ra của M_1 sẽ trở thành tín hiệu vào (X_2) của ôtômat M_2.

$$M_2 = \left(X_2, \left\{B_\tau\right\}, Z_2, f_2, g_2\right) \qquad S_2 = \left\{B_\tau\right\}$$

$$\tau = (\overline{1,5}, \overline{2,6}, \overline{3}, \overline{4}) = (a, b, c, d) = \left\{B_\tau\right\}$$

Từ đó có thể có ôtômat M_2 là : $g_2 = g$

$$M_2 = \left(X_2, S_2, Z_2, f_2, g_2\right)$$

$$M_2 = \left(\left\{\alpha, \beta, \gamma, \eta\right\}, \left\{B_\tau\right\}, Z_2, f_2, g_2\right)$$

$$M_2 = \left(\left\{\alpha, \beta, \gamma, \eta\right\}, \left\{a, b, c, d\right\}, Z, f_2, g\right) \qquad Z_2 = Z$$

Tới đây ta có được cấu tạo bảng chức năng của ôtômat M_2 như chỉ ra trên hình 4.55.

Bảng chức năng M_2

X_2 / S_2					Z_2			
	α	β	γ	η	α	β	γ	η
a	c	a	b	d	1	0	1	0
b	c	a	a	d	0	1	-	1
c	c	b	-	-	0	1	-	-
d	d	a	-	-	0	0	-	-

Hình 4.55. Ôtômát M_2.

Cách điền bảng chức năng ôtômat M_2 :

Ta sẽ điền trạng thái tiếp theo cho ôtômat M_2 theo từng tọa độ của nó ở trong bảng, để điền được trạng thái vào bảng ta dựa vào quan hệ.

$$f_2\left(\left\{B_\tau\right\}, \left\{B_\Pi\right\} \times \left\{X\right\}\right) = f\left(B_\tau . B_\Pi, X\right)$$

Với tọa độ đầu tiên (a, α) :

$$f_2\left(a, \alpha\right) = f_2\left(a, (A, 0)\right) = f\left(a . A, 0\right) =$$

$f\left(\overline{1,5} . \overline{1,2,3,4}, 0\right) = f\left(\overline{1}, 0\right) = 3$ trong bảng chức năng M, mà 3 = c ký hiệu trạng thái M_2.

Vậy : $f_2(a, \alpha) =$ ⓒ

Tọa độ tiếp theo (a, β) với cách làm tương tự trên :

$$f_2\left(a, \beta\right) = f_2\left(a, (A, 1)\right) = f\left(a . A, 1\right) =$$

$$f\left(\overline{1,5}\ .\ \overline{1,2,3,4}\ ,\ 1\right) = f\left(\overline{1}\ ,\ 1\right) = 1 = \textcircled{a}$$

Với cách làm tương tự ta sẽ lần lượt tính toán tìm ra các trạng thái cần điền ứng với từng tọa độ ở trong bảng M_2. Sau đây chúng ta xem xét tới một tọa độ đặc biệt.

Tọa độ (c, γ) chúng ta có :

$$f_2(c,\ \gamma) = f_2\ \left(c,\ (B,\ 0)\right) = f\ (c\ .\ B,\ 0) =$$

$$f(\overline{4}\ .\ \overline{5,6}\ ,\ 0) = f(\Phi\ ,\ 0) = \textcircled{-}\ \text{không xác định.}$$

$$f_2(c,\gamma) = \textcircled{-}$$

Tiếp theo ta sẽ điền nốt khu vực hàm ra (g_2) của M_2 cách làm cũng điền theo từng tọa độ, và dựa vào quan hệ sau :

$$g_2\left(\left\{B_\tau\right\}\ ,\ \left\{B_\Pi\right\}\ \times\ \left\{X\right\}\right) = g\left(B_\tau\ .\ B_\Pi\ ,\ X\right)$$

Với tọa độ đầu tiên trong khu vực hàm ra (a, α) :

$$g_2(a,\alpha) = g_2\left(a,(A,0)\right) = g(a\ .\ A,\ 0)$$

$$g(\overline{1,5}\ .\ \overline{1,2,3,4}\ ,\ 0) = g(\overline{1}\ ,\ 0) = 1$$

$$g_2(a\ ,\ \alpha) = \textcircled{1}$$

Tọa độ (a, β) :

$$g_2(a,\beta) = g_2\left(a,(A,1)\right) = g(a\ .\ A,\ 1)$$

$$g(\overline{1,5}\ .\ \overline{1,2,3,4}\ ,\ 1) = g(\overline{1}\ ,\ 1) = 0$$

$$g_2(a\ ,\ \beta) = \textcircled{0}$$

Với cách làm tương tự chúng ta sẽ thu được giá trị của đáp ứng ra của M_2 theo chức năng của M đã cho trước. Sau đây ta xét thêm tọa độ đặc biệt :

Tọa độ (c, γ) ta có :

$$g_2(c,\ \gamma) = g_2\ \left(c,\ (B,\ 0)\right) = g\ (c\ .\ B,\ 0) =$$

$$g(\overline{4}\ .\ \overline{5,6}\ ,\ 0) = g(\Phi\ ,\ 0) = -\ \text{vì không có trạng thái nên không điều khiển, không xác định}$$

$$g_2\ (c,\ \gamma) = \textcircled{-}$$

Như vậy từ phân hoạch τ tìm ra được từ phân hoạch Π chúng ta đã xây dựng được ôtômat M_2, mà hoạt động hay từng trạng thái tiếp theo của nó đã tìm được đều dựa vào ôtômat M cho trước (số liệu ban đầu). Sau khi xây dựng được mạch điện cho hai ôtômat M_1 và M_2 ta chỉ cần nối tiếp chúng lại sẽ tạo ra một ôtômat thực hiện chức năng của ôtômat M ban đầu ($M_1\ \textcircled{\rightarrow}\ M_2 = M$).

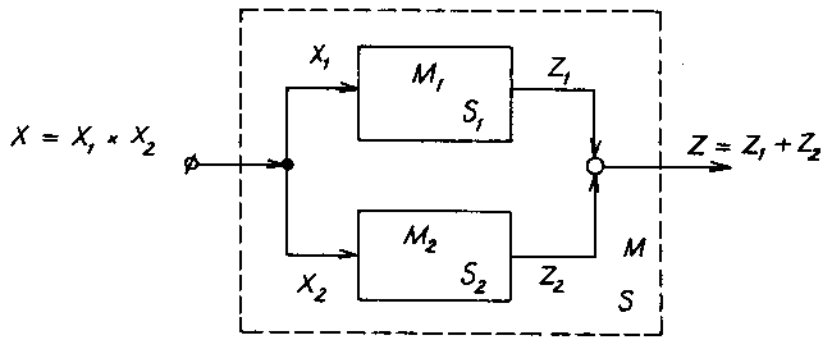
4.6.3. PHÂN GIẢI SONG SONG ÔTÔMAT CÓ NHỚ

Việc phân giải song song ôtômat có nhớ cũng có hai bài toán là : bài toán thuận và bài toán ngược. Bài toán thuận là cho trước hai ôtômat M_1 và M_2, sau đó chỉ việc nối

song song chúng lại sẽ tạo ra một ôtômat M nào đó. Thực hiện việc mắc song song đó chúng ta không cần có điều kiện nào cả. Bài toán ngược là cho trước ôtômat M, ta phải chia nó ra thành hai ôtômat con thành phần là M_1 và M_2, sao cho khi thực hiện nối song song chúng lại với nhau thì thực hiện được chức năng của ôtômat M ban đầu. (M_1 // M_2 = M).

4.6.3.1. Bài toán thuận của phân giải song song ôtômat

Nhiệm vụ của bài toán này là cho trước hai ôtômat M_1 và M_2, thực hiện nối song song chúng lại sẽ tạo ra một ôtômat M lớn hơn. Sơ đồ biểu diễn nối song song hai ôtômat được mô tả trên hình 4.56.



Hình 4.56. Nối song song hai ôtômat.

Định nghĩa 4.9 : Cho trước hai ôtômat có nhớ

$$M_1 = (X_1, S_1, f_1, g_1) \text{ và } M_2 = (X_2, S_2, Z_2, f_2, g_2)$$

Nối song song M_1 và M_2 tạo ra ôtômat M như sau :

$$M = (X, S, Z, f, g)$$

$$M_1 \,//\, M_2 = M = (X_1 \times X_2, S_1 \times S_2, Z_1 \times Z_2, f, g)$$

Trong đó :

$$X = (X_1 \times X_2)$$

$$S = (S_1 \times S_2)$$

$$Z = (Z_1 \times Z_2)$$

$$f\Big((s_1, s_2), (x_1, x_2)\Big) = \Big(f_1(s_1, x_1), f_2(s_2, x_2)\Big)$$

$$g\Big((s_1, s_2), (x_1, x_2)\Big) = \Big(g_1(s_1, x_1), g_2(s_2, x_2)\Big)$$

Như vậy các tham số của ôtômat M đều đã được tính toán hay xác định dựa vào các tham số của hai ôtômat M_1 và M_2 đã cho. Việc nối song song hai ôtômat có sẵn ta không cần một điều kiện hay yêu cầu gì cả.

4.5.3.2. Bài toán ngược của phân giải song song ôtômat

Nhiệm vụ của phần này là một ôtômat M cho trước chúng ta phải tìm cách phân giải nó ra thành hai ôtômat con thành phần là M_1 và M_2 sao cho sau khi nối song song,

chúng phải thực hiện được ôtômat M. Không phải bất kỳ ôtômat có nhớ nào cũng thực hiện được phân giải song song. Cho nên ở đây chúng ta phải đưa ra điều kiện kiểm tra hay cho phép phân giải song song một ôtômat có nhớ. Điều kiện thực hiện phân giải song sóng ôtômat được nêu lên ở trong định lý (4.8).

Định nghĩa 4.10 : Hai ôtômat M_1 và M_2 được gọi là phân giải song song của ôtômat M nếu và chỉ nếu M_1 nối song song với M_2 thực hiện ôtômat M ($M_1 // M_2 = M$). Phân giải đó là không tầm thường nếu M_1 và M_2 có số lượng trạng thái ít hơn trạng thái của M.

Điều kiện cần và đủ để có thể thực hiện được phân giải song song của ôtômat có nhớ thành hai ôtômat con thành phần nếu thỏa mãn định lý sau.

Định lý 4.8 : Ôtômát M có phân giải song song không tầm thường nếu và chỉ nếu tồn tại hai phân hoạch thế không tầm thường Π_1 và Π_2 trên tập trạng thái S của ôtômat M sao cho $\Pi_1 . \Pi_2 = \Phi$.

Dựa vào định lý (4.8) cho phép ta biết được ngay từ đầu một ôtômat có nhớ có thể phân giải song song được hay không. Để thực hiện phân giải song song một ôtômat có nhớ M cho trước chúng ta phải qua các bước sau :

1. Vẽ dàn phân hoạch thế D_M của ôtômat M
2. Từ dàn phân hoạch thế D_M ta tìm ra hai phân hoạch thế Π_1 và Π_2 sao cho thỏa mãn $\Pi_1 . \Pi_2 = \Phi$.
3. Từ phân hoạch Π_1 tạo ra ôtômat M_1 và từ phân hoạch Π_2 xây dựng nên ôtômat M_2.

Dựa vào các bước đã nêu trên chúng ta thực hiện phân giải song song ôtômat M theo trình tự sau.

Cho trước :

$$M = (X, S, Z, f, g)$$

Từ ôtômat M đã cho chúng ta tìm ra được một phân hoạch thế Π_1. Từ phân hoạch Π_1 ta tạo ra ôtômat M_1 như sau :

$$M_1 = (X_1, S_1, Z_1, f_1, g_1)$$

$$M_1 = \left(X\left\{ B_{\Pi_1} \right\}, \left\{ B_{\Pi_1} \right\} \times \left\{ X \right\} , f_{\Pi_1} , e \right)$$

Trong đó :

$X_1 = X$ vì đầu vào ta có thể nối lại với nhau.

$S_1 = \left\{ B_{\Pi_1} \right\}$ Tập các khối của phân hoạch thế Π_1

$Z_1 = \left\{ B_{\Pi_1} \right\} \times \left\{ X \right\}$ Trạng thái S_1 tích Đêcác với X tạo ra đáp ứng ra Z_1 theo Mealy.

$f_1 = f_{\Pi_1}$ Hàm chuyển biến f_1 là hoạt động của hàm f theo phân hoạch Π_1.

g_2 = e là hàm đồng nhất vì ta không thể nối đầu ra của 2 ôtômat với nhau.

Cũng như vậy từ phân hoạch thế Π_2 tạo ra được ôtômat M_2 như sau :

$$M_2 = (X_2, S_2, Z_2, f_2, g_2)$$

$$M_2 = \left(\left\{B_{\Pi_2}\right\}, \left\{B_{\Pi_2}\right\} \times \left\{X\right\}, f_{\Pi_2}, e\right)$$

Trong đó :

X_2 = X Vì đầu vào ta có thể hàn nối lại.

$S_2 = \left\{B_{\Pi_2}\right\}$ Tập các khối của phân hoạch Π_2

$Z_1 = \left\{B_{\Pi_2}\right\} \times \left\{X\right\}$ Trạng thái S_2 tích Đêcác với X tạo ra đáp ứng ra Z_1 theo Mealy.

$f_2 = f_{\Pi_1}$ Hàm f_2 hoạt động theo hàm f theo phân hoạch Π_2

g_2 = e là hàm đồng nhất.

Ví dụ : Thực hiện phân giải song song ôtômat có nhớ hoạt động theo bảng chức năng ở trên hình 4.57 sau :

Bảng chức năng

S \ X	0	1	Z
1	4	3	0
2	6	3	0
3	5	2	1
4	2	5	0
5	1	4	0
6	3	4	0

Hình 4.57. Cho một ôtômat.

Sau khi vẽ dàn phân hoạch thế cho ôtômat M người ta tìm được hai phân hoạch thế không tầm thường sau :

$$\Pi_1 = (\overline{1,2,3}\ ;\ \overline{4,5,6}) = (\overline{A,B}) = \left\{B_{\Pi_1}\right\}$$

$$\Pi_2 = (\overline{1,6}\ ;\ \overline{2,5}\ ;\ \overline{4,3}) = (\overline{C,D,E}) = \left\{B_{\Pi_2}\right\}$$

Ta thực hiện $\Pi_1 . \Pi_2 = (\overline{1}\,;\overline{2}\,;\overline{3}\,;\overline{6}\,;\overline{5}\,;\overline{4}) = \Phi$

Từ kết quả thu được chúng ta thấy hai phân hoạch Π_1 và Π_2 thỏa mãn điều kiện của định lý.

+ Từ phân hoạch thế Π_1 ta xây dựng được ôtômat M_1 dựa theo ôtômat M đã cho như sau ở trên hình 4.58.

Bảng chức năng M_1

S_1 \ X_1	0	1	Z_1
A	(B)	(A)	1
B	A	B	0

Hình 4.58. Ôtômát M_1.

$$\Pi_1 = (\overline{1,2,3}\ ;\ \overline{4,5,6}) = (A,B) = \left\{B_{\Pi_1}\right\}$$

$$M_1 = \left((0,1)\ ,\ (A,B)\ ,\ (A,B)\times(0,1)\ ,\ f_{\Pi_1}\ e\right)$$

Cách điền bảng M_1 như sau :

Ta sẽ điền vào bảng M_1 theo từng tọa độ của nó, để điền được chúng ta dựa vào quan hệ

$$f_1 = f_{\Pi_1}$$

Với tọa độ đầu tiên (A, 0) của bảng M_1 :

f_1 (A, 0) = $f(\overline{1,2,3}\ ,\ 0) = \overline{4,6,5}$ theo bảng M.

mà $\overline{4,5,6}$ = B theo ký hiệu của bảng M_1. Vậy f_1(A, 0) = (B)

Tọa độ tiếp theo (A, 1) :

$f_1(A,\ 1) = f(\overline{1,2,3}\ ,\ 1) = \overline{3,3,2} = A$

$f_1(A,\ 1) =$ (A)

Theo cách làm như trên ta có :

$f_1(B,\ 0) = A$ $\qquad$ $f_1(B,\ 1) = B$

Ở đây đầu ra hay đáp ứng ra Z_1 ta chưa thể điền ngay được giá trị. Sau khi tìm được bảng chức năng cho ôtômat M_2 từ Π_2, chúng ta sẽ thống nhất cách điền cho Z_1 và Z_2 theo hàm e nên được gọi là hàm đồng nhất sẽ nêu cụ thể ở phần sau.

+ Từ phân hoạch thế Π_2 ta xây dựng được ôtômat M_2 từ ôtômat M đã cho như sau ở trên hình 4.59.

Bảng chức năng M_2

S_2 \ X_2	0	1	Z_1
C	(E)	(E)	0
D	C	E	0
E	D	D	1

Hình 4.59. Ôtômát M_2.

$$\Pi_2 = (\overline{1,6}\ ;\ \overline{2,5}\ ;\ \overline{4,3}) = (\overline{C,D,E}) = \left\{B_{\Pi_2}\right\}$$

$$M_2 = \left((0,1) , (C,D,E) , Z_2 , f_{\Pi_2} , e \right)$$

Cách điền bảng M_2 như sau :

Điền vào bảng M_2 theo từng tọa độ của nó, chúng ta dựa vào quan hệ :

$$f_2 = f\Pi_2$$

Với tọa độ đầu tiên (C, 0) của bảng M_2 :

$$f_2(C, 0) = f(C, 0) = \overline{4,3} \text{ theo bảng M.}$$

Mà $\overline{4,3}$ = E theo ký hiệu trạng thái của M_2 vậy

$$f_2(C, 0) = (E)$$

Tọa độ tiếp theo (C, 1) :

$$f_2(C, 1) = f(\overline{1,6} , 1) = \overline{3,4} = E$$

$$f_2 (C, 1) = \triangle E$$

Với cách làm tương tự thu được các kết quả :

$$f_2(D, 0) = C \qquad f_2(D, 1) = E$$

$$f_2(E, 0) = D \qquad f_2(C, 1) = D$$

Từ các kết quả thu được chúng ta điền đầy được khu vực trạng thái tiếp theo của ôtômat M_2 (khu f_2).

Sau khi điền đầy đủ các trạng thái tiếp theo cho ôtômat M_1 và ôtômat M_2, dựa vào đáp ứng ra Z của ôtômat M cho trước, ta điền nốt đáp ứng ra cho M_1 và M_2 (khu vực hàm đồng nhất e) như sau :

Từ ôtômat M cho trước, do ôtômat biểu diễn theo Moore ở đáp ứng ra nên ta có thể viết :

$$g(3) = 1$$

Do điều kiện $\Pi_1 . \Pi_2 = \Phi$ cho nên ta tìm hai trạng thái bất kỳ của Π_1 và Π_2 nhân với nhau sao cho kết quả ra khối có 1 phần tử là 3 (ứng với ôtômat M). Ví dụ ở ôtômat M_1 lấy trạng thái A = $(\overline{1,2,3})$ còn ở ôtômat M_2 lấy trạng thái

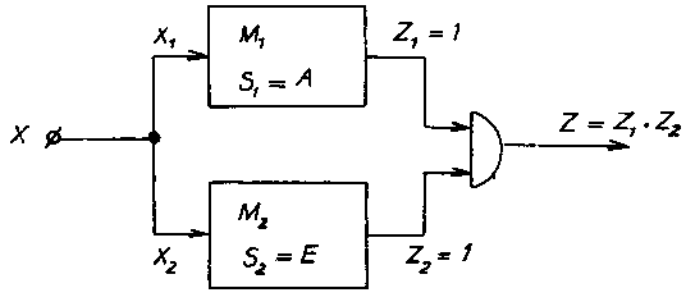
E = $(\overline{4,3})$, từ hai trạng thái đó ta có :

$$A . E = (\overline{1,2,3}) . (\overline{4,3}) = \overline{3} \text{ ra phần tử 3}$$

Như vậy chỉ có hai trạng thái A và E của hai ôtômat M_1 và M_2 tạo ra trạng thái 3 của ôtômat M bằng cách nhân chung lại với nhau. Tức là chỉ ở ôtômat M_1 xuất hiện trạng thái A có giá trị 1 ở đầu ra, và ôtômat M_2 xuất hiện trạng thái E có giá trị 1 ở đầu ra, thì (M_1 // M_2) mới cho đáp ứng ra là giá trị 1 tương đương với trạng thái 3 ở ôtômat M. Vậy hàm đồng nhất ở đây ta sẽ dùng mạch và cho hai ôtômat M_1 và M_2.

$$g_1(A) = 1 \text{ và } g_2(E) = 1$$

chỉ khi g_1 (A) = 1 và g_2 (E) = 1 thì đầu ra nối song song của hai ôtômat (M_1 // M_2) sẽ có giá trị 1 như ở ôtômat M cho trước, còn các trạng thái khác thì đáp ứng ra của M_1 và M_2 đều bằng 0. Sơ đồ cách nối song song ôtômat M_1 và ôtômat M_2 để tạo ra ôtômat M như biểu diễn ở trên hình 4.60.

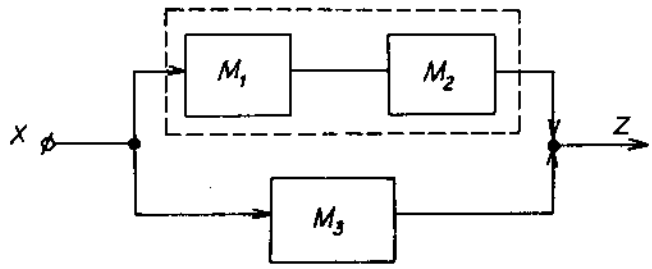


Hình 4.60. Cách tổ hợp song song hai ôtômat.

Ở đây ta chú ý do đầu ra của hai ôtômat M_1 và M_2 không thể hàn nối lại với nhau, vì nếu nối chúng lại sẽ bị chập mạch, cho nên việc nối song song hai ôtômat ta chỉ có thể nối hàn trực tiếp ở đầu vào. Còn ở đầu ra chúng ta luôn phải dùng mạch lôgíc để "hàn nối song song" chúng lại với nhau trên cơ sở đồng nhất. Nếu ở đáp ứng ra của ôtômat M cho trước có nhiều giá trị 1 (Z có nhiều vị trí 1), lúc này chúng ta không thể chỉ dùng một mạch lôgic để "hàn nối" song song cho hai ôtômat M_1 và M_2 được nữa, mà ta phải dùng tổ hợp của các mạch lôgíc lại với nhau sao cho thỏa mãn được đáp ứng ra của ôtômat M ban đầu, lúc đó ở đầu ra ta mới coi như nối "song song" được các ôtômat.

Sau khi đã phân giải song song và tìm được hai ôtômat con thành phần là M_1 và M_2, ta sẽ mã hóa cho chúng và sẽ thực hiện lắp ráp mạch điện cho hai ôtômat con đó. Khi đã có mạch điện cụ thể của hai ôtômat M_1 và M_2 ta chỉ cần nối song song chúng lại sẽ ra một ôtômat thực hiện chức năng của ôtômat M ban đầu đưa ra. Hai ôtômat con thành phần đó ta thấy số lượng trạng thái của chúng ít hơn ở ôtômat M vậy thỏa mãn mục đích yêu cầu đề ra cho nhiệm vụ phân giải ôtômat.

Ta chỉ cần thực hiện được hai nhiệm vụ là phân giải nối tiếp và phân giải song song cho ôtômat có nhớ. Từ hai cách phân giải đó chúng ta có thể tạo ra những ôtômat hay những mạch điện có khả năng phân giải hỗn hợp, điều đó được minh họa trên hình 4.61.



Hình 4.61. Phân giải hỗn hợp ôtômat.

Chương V

THỰC HIỆN ÔTÔMAT CÓ NHỚ

Như đã biết một hệ thống số có thể được xây dựng dựa vào các mạch logic cơ bản, đó là các ôtômat không có nhớ. Tuy nhiên khi thực hiện ôtômat có nhớ (hệ dãy) không thể chỉ dựa vào các mạch lôgic mà còn cần đến các mạch điện ghi giữ thông tin hay ghi giữ trạng thái trước đó, bởi vì hoạt động của một ôtômat có nhớ phải thỏa mãn quan hệ toán học là :

$$Y = f(S, X)$$

Trong đó (Y) là trạng thái tiếp theo phải phụ thuộc vào trạng thái trước đó hay tin tức trước đó (S) cùng với tín hiệu tác động vào là (X). Đối với tín hiệu vào của ôtômat có nhớ thông thường là một xung điện kích vào, đơn giản nhất là một xung được tạo ra khi bấm tay lên phím bấm của một bàn phím nào đó. Nhưng để có được trạng thái trước đó hay thông tin trước đó của ôtômat có nhớ bắt buộc phải dùng tới các phần tử ghi giữ thông tin, chúng có tên là "trigơ" hay "Flíp–Flốp", và còn được gọi là các "mạch lật".

Phần tiếp theo sẽ đề cập đến chức năng của các loại trigơ đang được sử dụng khi xây dựng một ôtômat có nhớ.

Ngoài ra việc thiết kế hay thực hiện một ôtômat có nhớ còn phụ thuộc vào cấu trúc của loại ôtômat cần được xây dựng, bởi vì ôtômat có nhớ được chia ra làm hai loại. Loại thứ nhất là ôtômat đồng bộ, hoạt động của ôtômat loại này được thống nhất theo một xung nhịp điều khiển hay hoạt động được chuẩn hóa theo thời gian, loại ôtômat đồng bộ tín hiệu kích vào thường là các xung theo thời gian. Loại thứ hai là ôtômat không đồng bộ, hoạt động của ôtômat loại này không theo xung nhịp hay không được chuẩn hóa theo thời gian, đối với ôtômat loại này tín hiệu tác động vào có thể là xung điện hay là một điện thế được đưa vào.

§5.1. CÁC PHẦN TỬ GHI NHỚ THÔNG TIN - TRIGƠ

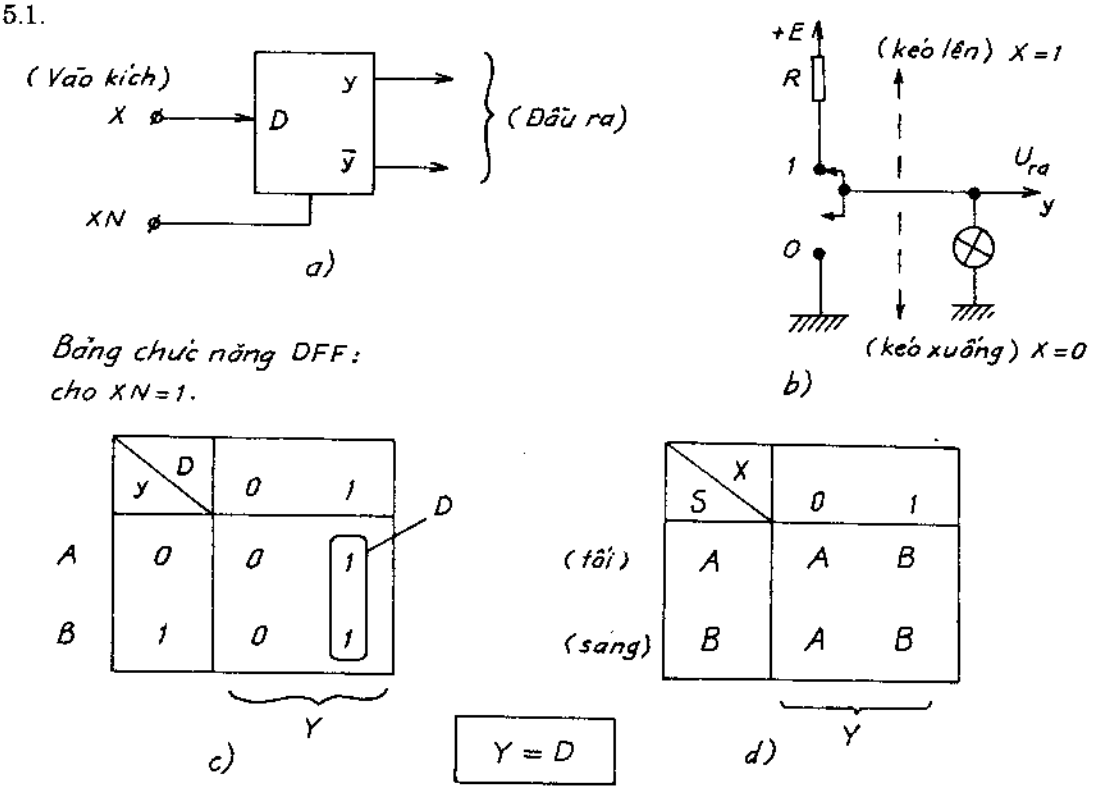
Các phần tử nhớ cơ bản - trigơ (Flíp–Flốp) thường được chế tạo sẵn trong các chíp điện tử IC. Nguyên lý làm việc và mạch điện cụ thể của chúng trình bày trong giáo

trình mạch điện tử hay kỹ thuật xung, nên ở đây chỉ đề cập tới các chức năng hoạt động của chúng để ứng dụng chúng vào việc xây dựng tổ hợp ôtômat có nhớ.

Các trigơ thực chất là một ôtômat có nhớ đơn giản nhất, nó "đóng-mở", "bật-tắt" hay 1 và 0. Do trigơ là một ôtômat có nhớ bé nhất cho nên *hoạt động của nó vẫn phải tuân theo cách biểu diễn của ôtômat có nhớ,* tức là chức năng của nó phải tuân theo cách mô tả bảng chức năng của Mealy hay Moore.

5.1.1. TRIGƠ D (D FLÍP FLỐP–DFF)

Đây là trigơ đầu tiên và đơn giản nhất, nó có hai đầu kích phụ thuộc lẫn nhau là đầu vào (D) vào đầu vào kích xung nhịp (XN). Nó có hai đầu ra là y và $\bar{y}$ còn được gọi là hai vai của trigơ. Ký hiệu của DFF và bảng chức năng của nó được chỉ ra ở trên hình 5.1.

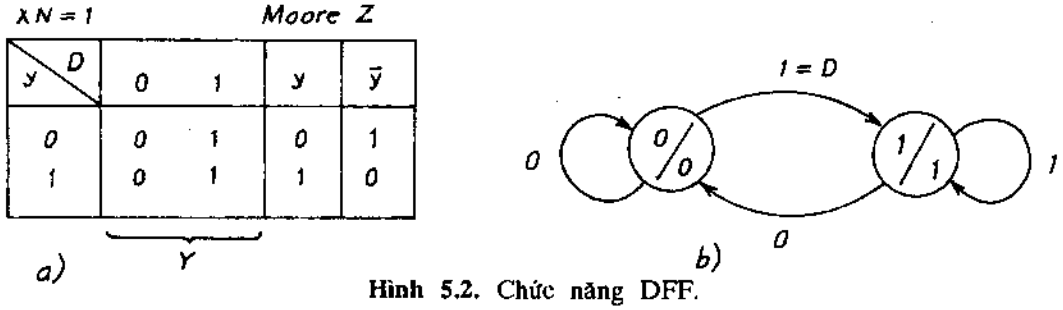


Hình 5.1. Chức năng của DFF.

Do DFF là ôtômat có nhớ cho nên khu vực hàm f hay khu trạng thái tiếp theo của nó được phép điền tùy ý, ví dụ khu trạng thái tiếp theo Y của DFF ta điền theo hoạt động của trigơ D hay của chuyển mạch trên hình 5.1b, hoạt động của chuyển mạch đó được mô tả trên bảng ở hình 5.1d, đó cũng chính là bảng chức năng của DFF. Từ bảng chức năng có bảng mã hóa của trigơ D ở hình 5.1c ta có được phương trình chuyển biến trạng thái của nó là :

$$Y = D$$

Do trigơ D có hai đầu ra là y và $\bar{y}$ ứng với đầu ra Z của ôtômat có nhớ, ta có thể hiểu ($Z = y$, $\bar{Z} = \bar{y}$) vậy bảng chức năng đầy đủ của DFF sẽ được biểu diễn trên hình 5.2a, còn ở hình 5.2b là đồ hình của nó.

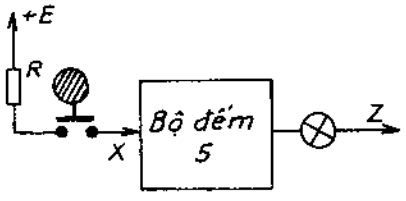


Hình 5.2. Chức năng DFF.

Các ví dụ sau minh hoạ việc ứng dụng các chức năng của trigơ D.

Ví dụ : Thực hiện bộ đếm 5, cứ đếm đến 3 thì báo dùng trigơ D (DFF), mã hóa theo nhị phân tăng dần và mô tả bằng ôtômat Moore.

Chúng ta có thể viết bảng mã hóa theo nhị phân tăng dần của bộ đếm 5 trên bảng ở hình 5.3.



a)

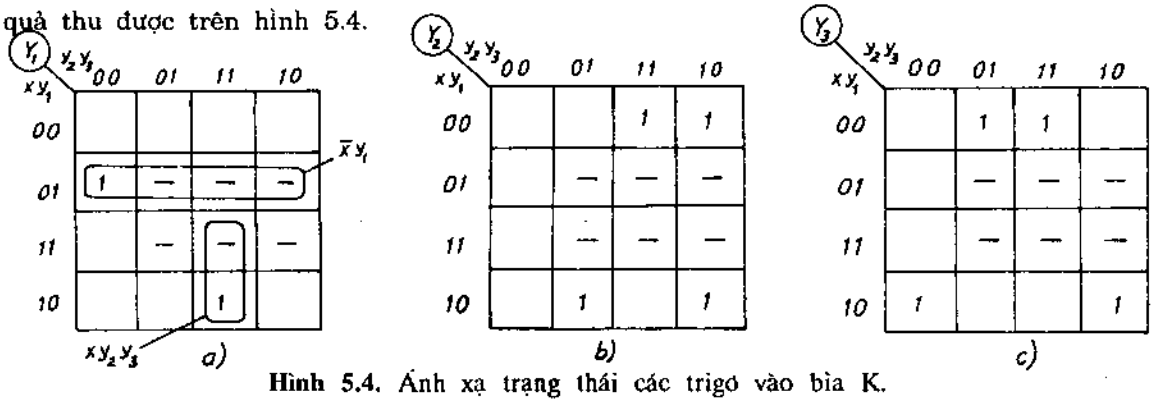
Bảng mã hóa của bộ đếm :

	$y_1y_2y_3$ \ X	0	1	Z
0	000	000	001	0
1	001	001	010	0
2	010	010	011	0
3	011	011	100	1
4	100	100	000	0
	101	– – –	– – –	–
	110	– – –	– – –	–
	111	– – –	– – –	–
		$Y_1\,Y_2\,Y_3$	$Y_1\,Y_2\,Y_3$	

b)

Hình 5.3. Bảng mã hóa của bộ đếm.

Phương pháp thiết kế thực hiện bằng cách ánh xạ Y_1, Y_2 và Y_3 vào bìa Kácnô, kết quả thu được trên hình 5.4.



Hình 5.4. Ánh xạ trạng thái các trigơ vào bìa K.

Sau đó từ các bìa Kácnô chúng ta rút gọn và thu được hệ phương trình chuyển biến trạng thái của Y_1, Y_2 và Y_3. Kết quả chúng ta thu được hệ phương trình chuyển biến trạng thái có dạng :

$$Y_1 = \bar{x}\, y_1 + x\, y_2\, y_3$$

$$Y_2 = \bar{x}\, y_2 + y_2\, \bar{y}_3 + x\, \bar{y}_2\, y_3$$

$$Y_3 = \bar{x}\, y_3 + x\, \bar{y}_1\, \bar{y}_3$$

$$Z = \bar{y}_1\, y_2\, y_3$$

Do chức năng của trigơ D hoạt động theo phương trình chuyển biến trạng thái (Y = D), vì thế ta có thể thay D vào hệ phương trình chuyển biến trạng thái ở trên, mà D lại là đầu vào kích của trigơ D, cho nên ta thu được hệ phương trình hàm kích có dạng :

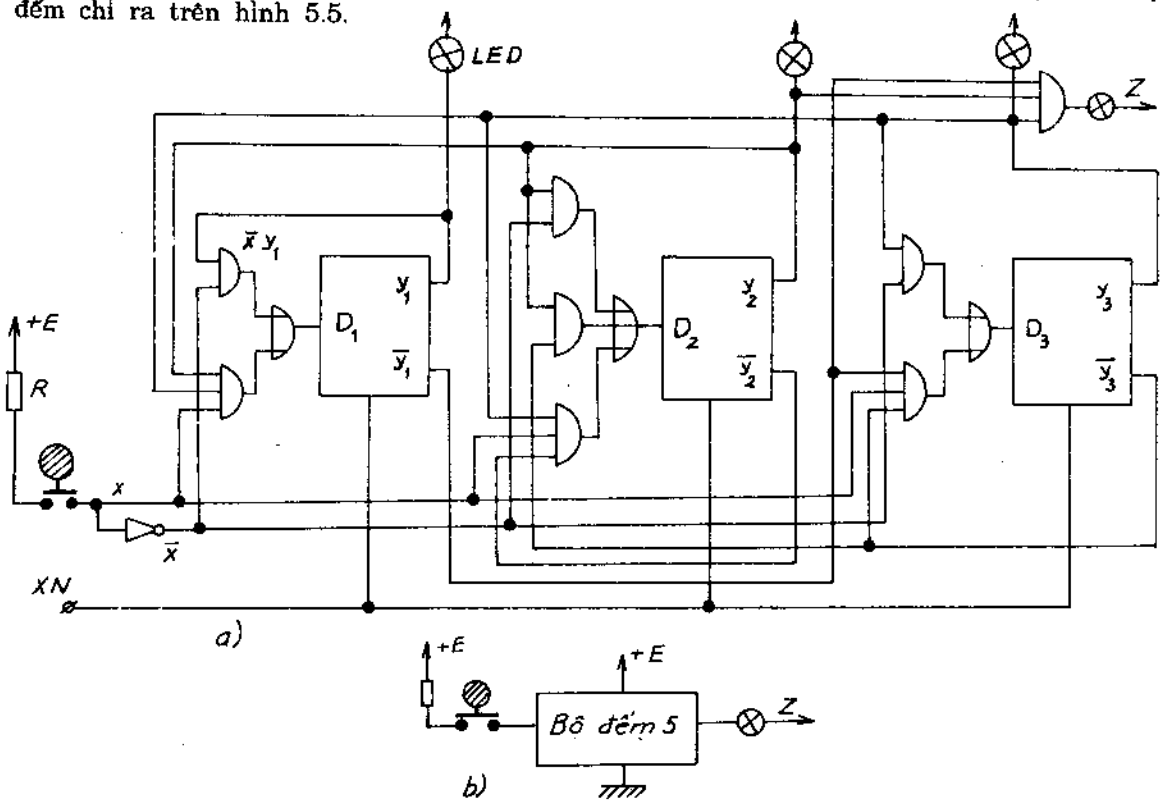
$$D_1 = \bar{x}\, y_1 + x\, y_2\, y_3$$

$$D_2 = \bar{x}\, y_2 + y_2\, \bar{y}_3 + x\, \bar{y}_2\, y_3$$

$$D_3 = \bar{x}\, y_3 + x\, \bar{y}_1\, x\, \bar{y}_1\, \bar{y}_3$$

$$Z = \bar{y}_1\, y_2\, y_3$$

Từ hệ phương trình hàm kích đó chúng ta có được mạch điện cụ thể của bộ đếm 5 mà cứ đến 3 thì báo, trạng thái trong của bộ đếm thay đổi theo quy luật của nhị phân tăng dần. Dựa vào hệ phương trình hàm kích chúng ta vẽ ra được mạch điện của bộ đếm chỉ ra trên hình 5.5.



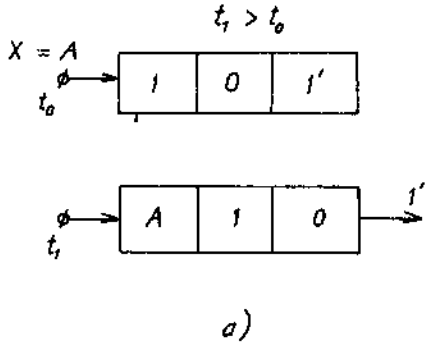
Hình 5.5. Bộ đếm 5 dùng DFF.

Sau khi vẽ mạch điện chức năng của bộ đếm 5 dùng DFF, ta có hể dựa theo sơ đồ đó tìm kiếm các linh kiện và tiến tới lắp ráp được mạch cụ thể. Có một lưu ý là đầu ra (y) của các DFF nếu chúng ta lắp thêm đèn LED để chỉ thị, ta gọi đó là các đèn LED chỉ thị trạng thái trong hay trạng thái nội bộ của bộ đếm, các đèn chỉ thị đó sẽ hoạt động (sáng tối) theo đúng quy luật của bảng mã hóa của bộ đếm 5. Điều đó cho phép chúng ta kiểm tra được hoạt động của bộ đếm (hay của ôtômat) trong từng xung bấm một, khi xây dựng cũng như khi sửa chữa một ôtômat có nhớ.

Từ ví dụ xây dựng bộ đếm 5 dùng DFF chúng ta nhận thấy *muốn có được mạch điện cho ôtômat có nhớ bắt buộc chúng ta phải tìm được hệ phương trình hàm kích của ôtômat đó.* Trong trường hợp xây dựng bộ đếm 5 dùng DFF, việc có được hệ phương trình kích của ôtômat có nhớ rất dễ dàng, vì hoạt động của trigơ D theo quan hệ (Y = D) nếu là loại trigơ khác (như RSFF hay JKFF...) thì việc tìm ra hệ phương trình kích sẽ phức tạp hơn. Ví dụ cho thấy, khi xây dựng một ôtômat có nhớ nếu không bị ràng buộc bởi bất kỳ điều kiện nào nên sử dụng DFF.

Với cách làm như trên, chúng ta có thể chế tạo ôtômat có chức năng bất kỳ nếu chúng ta có được bảng mã hóa chức năng của ôtômat có nhớ, hay chức năng nào đó.

Sau đây chúng ta khảo sát một ví dụ khác về xây dựng một ôtômat có nhớ dùng DFF, khi xây dựng bộ ghi dịch phải 3 bit, dùng trigơ D. Hoạt động của bộ ghi dịch phải và bảng chức năng của nó được chỉ ra ở trên hình 5.6.



Bảng mã hóa của bộ ghi dịch :

$y_1 y_2 y_3$ \ X	0	1
0 0 0	0 0 0	1 0 0
0 0 1	0 0 0	1 0 0
0 1 0	0 0 1	1 0 1
0 1 1	0 0 1	1 0 1
1 0 0	0 1 0	1 1 0
1 0 1	0 1 0	1 1 0
1 1 0	0 1 1	1 1 1
1 1 1	0 1 1	1 1 1
b)	$Y_1 Y_2 Y_3$	$Y_1 Y_2 Y_3$

Hình 5.6. Bộ ghi dịch phải và chức năng.

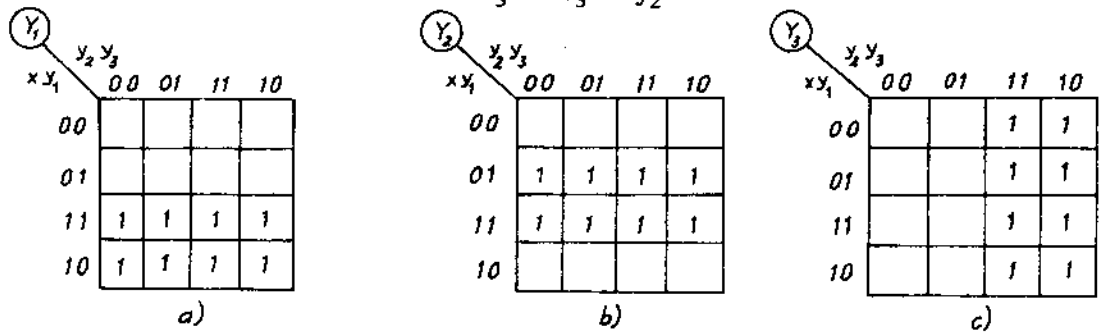
Bảng mã hóa ánh xạ Y_1, Y_2 và Y_3 vào bìa K được chỉ ra trên hình 5.7.

Từ các bìa Kácnô ta rút gọn bằng phương pháp khoanh dán ta thu được hệ phương trình chuyển biến trạng thái của bộ ghi dịch phải ba bít, hệ phương trình chuyển biến trạng thái đó cũng chính là hệ phương trình hàm kích của bộ ghi dịch phải 3 bit :

$$D_1 = Y_1 = X$$

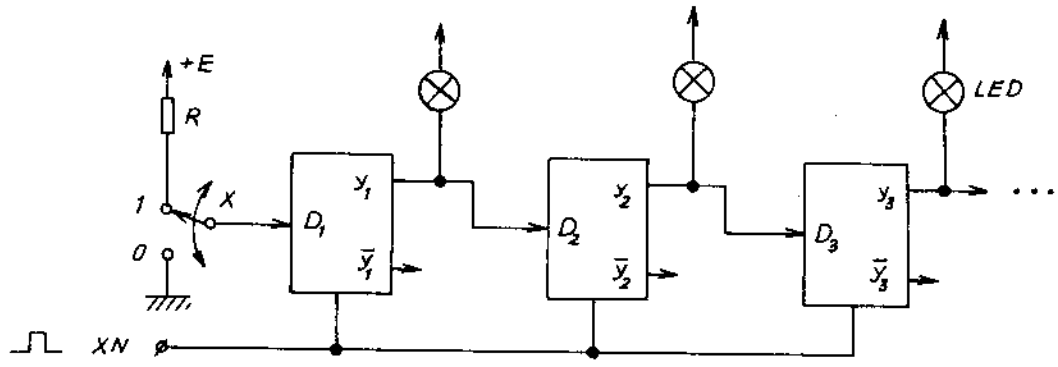
$$D_2 = Y_2 = y_1$$

$$D_3 = Y_3 = y_2$$



Hình 5.7.

Từ hệ phương trình kích thu được, ta vẽ mạch điện chức năng của bộ ghi dịch phải 3 bit dùng DFF như trên hình 5.8.



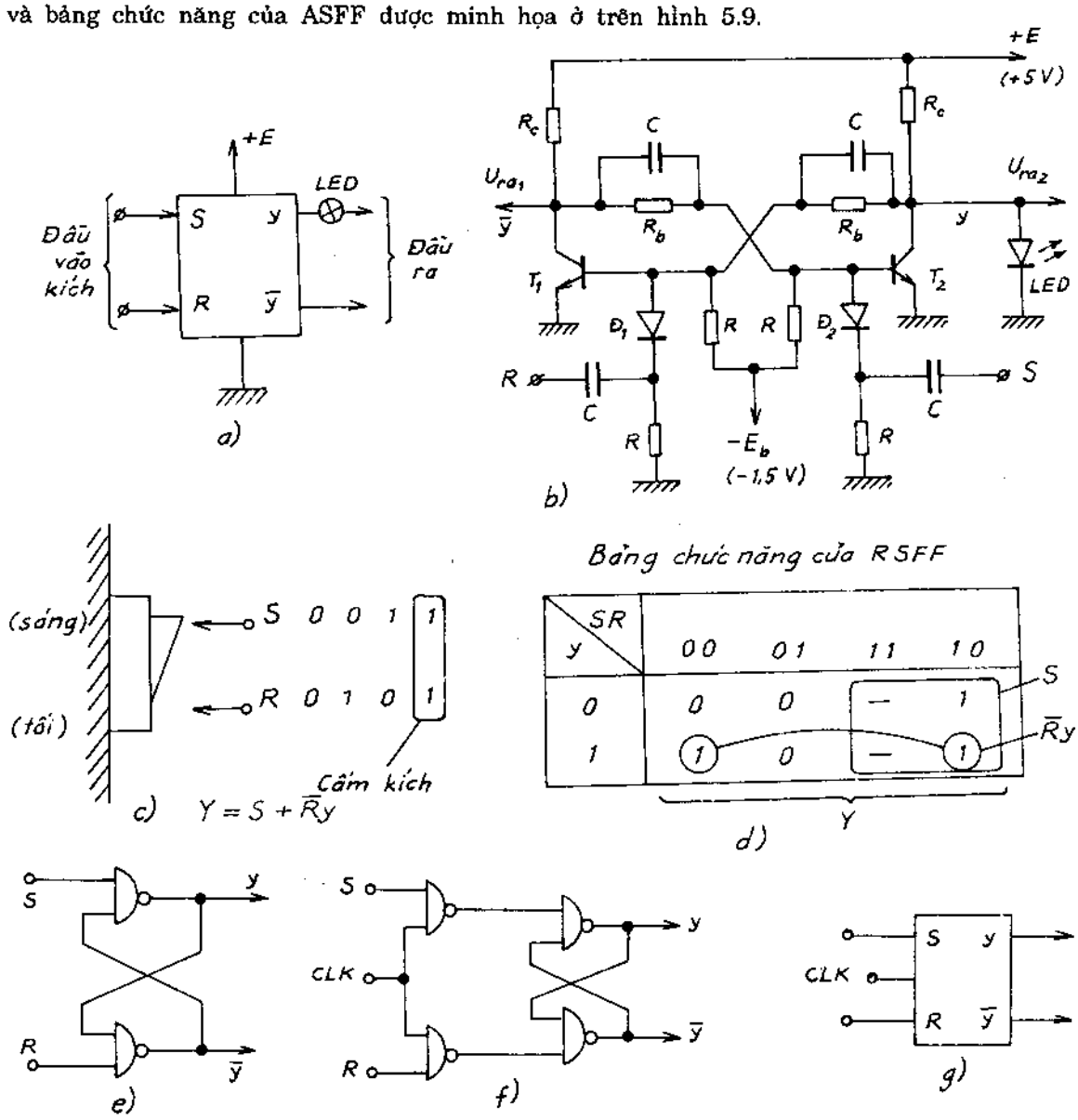
Hình 5.8. Bộ ghi dịch phải.

Nguyên lý làm việc của mạch phụ thuộc vào chuyển mạch X và xung nhịp (XN) theo phương pháp : gạt chuyển mạch X đầu vào xong thì cho (bấm) xung nhịp. Khi XN kích vào đồng thời các trigơ D, làm cho trigơ D chuyển trạng thái theo tín hiệu đang có trên đầu kích D của từng trigơ, các trigơ D theo hoạt động của XN, đầu ra trạng thái trước đó của trigơ này lại là đầu vào kích của trigơ tiếp theo tạo nên quan hệ dịch trạng thái của các trigơ. Với cách nối như vậy chúng ta có thể xây dựng bộ ghi dịch phải có độ dài tùy ý. Tương tự ta có thể xây dựng được bộ ghi dịch trái dùng trigơ D.

Bộ ghi dịch thường được sử dụng trong quảng cáo : ví dụ như chúng ta thay các đèn LED chỉ thị ở đầu ra các trigơ bằng các chữ phát sáng thì ta sẽ được hình ảnh các chữ chạy sáng lần lượt, nếu thay LED bằng các bóng đèn bố trí thích hợp, khi dịch chuyển quá trình sáng của chúng sẽ tạo nên pháo hoa điện tử. Bộ ghi dịch hay dùng nhất để tạo bảng đèn, ma trận các điểm sáng, mỗi điểm sáng được điều khiển bằng một trigơ D, do tính chất ghi dịch của nó nêu trên bảng đèn sáng sẽ tạo ra được các dòng chữ chạy.

5.1.2. TRIGƠ RS (RS FLÍP FLỐP - RSFF)

Trigơ RS là một ôtômat có nhớ nhỏ nhất, nó có hai trạng thái ổn định, muốn chuyển đổi được trạng thái của trigơ ta cần có hai đầu vào kích là S và R. Trigơ RS có hai đầu ra là y và $\bar{y}$ luôn ngược pha với nhau, được lấy ra bằng điện áp trên "hai vai" của trigơ, dựa vào mức điện áp ra trên đầu ra y ta biết được trigơ đang ở trạng thái nào (ví dụ nếu đầu ra y = +E thì trigơ đang ở trạng thái 1). Ký hiệu sơ đồ mạch điện và bảng chức năng của ASFF được minh họa ở trên hình 5.9.



Hình 5.9. Chức năng của trigơ RS.

Khi ta kích vào SR của trigơ để lật chuyển hay điều khiển nó, thì ta phải kích bằng cả hai tín hiệu kích là S và R. Nếu kích vào với S = 0 (hoặc R = 0) thì được coi là có kích nhưng kích "nhẹ nhàng", còn khi cho S = 1 (hay cho R = 1) thì là "kích mạch" bấm mạch vào. Quy ước về "kích" như vậy hợp với quy định lôgic là : lôgic 0 là

điều khiển 0 (như ngồi xuống) lôgic 1 là điều khiển 1 (đứng lên), còn không điều khiển gì là X (X có cả 0 và 1) tức là muốn đứng lên hay ngồi xuống tùy ý. Nguyên lý làm việc hay chức năng của RSFF là :

- Kích (0 0) là (S = 0 và R = 0) tức là "kích nhẹ" vào trigơ (công tắc đèn) cả trên lẫn dưới, nên đèn giữ nguyên trạng thái cũ, tức là trước đó y = 0 thì sau khi kích (0 0) tiếp theo Y = 0 (vẫn là 0), còn trước đó y = 1 thì sau khi kích (0 0) tiếp theo Y = 1 (vẫn là 1) điều đó được viết trong bảng chức năng, qua đó khi kích (SR = 00) được gọi là "giữ nguyên" trạng thái của RSFF.

- Kích (0 1) tức là (S = 0 còn R = 1) như vậy S ta "kích nhẹ" còn R ta "bấm mạnh". Kết quả là đèn tắt thì vẫn tắt (y = 0 thì Y = 0), còn nếu đèn đang sáng thì sẽ thành tắt (y = 1 sau khi kích (01) thì Y = 0). Do đó có thể gọi kích (SR = 01) là "xóa" hay thiết lập 0 cho RSFF.

-Tương tự như vậy khi ta kích (SR = 10) làm cho đèn đang sáng vẫn sáng (y = 1 thì Y = 1), còn nếu đèn đang tắt thì kích 10 sẽ sáng lên (y = 0 thì sau khi kích Y = 1). Hoạt động như vậy của trigơ ta có thể gọi kích (SR = 10) là "thiết lập 1" cho trạng thái của RSFF.

- Trường hợp kích (11) tức là (SR = 11) trường hợp này giống như ta "bấm mạnh" cả trên lẫn dưới của công tắc (trigơ) sẽ làm hỏng công tắc, đối với trigơ RS thì trạng thái của nó ta không thể biết chính xác sẽ như thế nào?, điều này thuộc bản chất mạch điện của RSFF, nếu không biết chính xác trạng thái của trigơRS sẽ như thế nào mà dùng RSFF để điều khiển chức năng thì rất nguy hiểm, trong thực tế người ta "cấm kích" tổ hợp (11) đối với loại trigơ này. Thể hiện điều đó ở khu trạng thái tiếp theo là các dấu không xác định.

Từ bảng chức năng của trigơ RS ta tìm được phương trình chuyển biến trạng thái của nó, hay trạng thái tiếp theo Y phụ thuộc vào trạng thái trước đó (y) dưới tác động của tín hiệu vào (SR) như thế nào :

$$Y = S + \overline{R}y$$

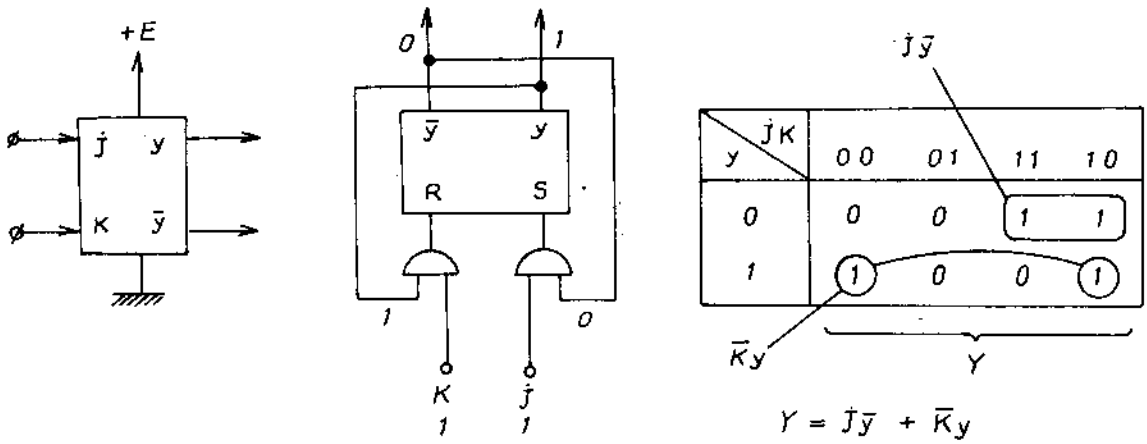
5.1.3. TRIGƠ JK (JK FLÍP FLỐP - JKFF)

Trigơ JK là sự cải tiến trigơ RS để dùng nốt bộ tín hiệu kích vào (11).

Cũng như các trigơ khác trigơ JK cũng là một ôtômat có nhớ nhỏ nhất, nó có hai trạng thái ổn định, muốn chuyển đổi trạng thái của JKFF hay điều khiển hoạt động của nó chúng ta cần có hai đầu kích vào đó là đầu J và K. Trigơ JK cũng có hai đầu ra y và $\overline{y}$, đầu ra y dùng để xác định trạng thái của trigơ nếu y = +E (y = 1) thì trigơ ở trạng thái "1". Ký hiệu, sơ đồ mạch điện vào bảng chức năng của JKFF được mô tả trên hình 5.10.

Nguyên lý làm việc của JKFF cũng tương tự như ở trigơ RS.

- Kích vào (J K = 0 0) làm hai mạch VÀ cùng không làm việc nên trạng thái của trigơ là không thay đổi.



Hình 5.10. Trigơ JK.

- Khi kích vào (J = 0 và K = 1), nếu JKFF đang ở 1 tức là (y = 1), mức lôgic 1 đó được đưa về nằm chờ ở đầu vào mạch VÀ khi có K = 1 (S = 0), thì mạch VÀ sẽ thông đưa tín hiệu 1 vào đầu R làm cho trigơ về 0 (Y = 0) như vậy (y = 1 kích SR = 01 thì được Y = 0). Còn nếu y = 0 thì lôgic 0 đó đưa về đầu vào nằm chờ khi có K = 1 (S = 0) thì mạch VÀ đó không làm việc nên trạng thái của trigơ vẫn giữ nguyên tức là (y = 0 kích SR = 0 1 thì kết quả Y = 0), cũng giống ở RSFF khi kích (J K = 0 1) thì là "xóa" hay "thiết lập" 0 cho JKFF.

- Tương tự như vậy khi ta kích vào (JK = 10) thì là "thiết lập" 1 cho trigơ JK.

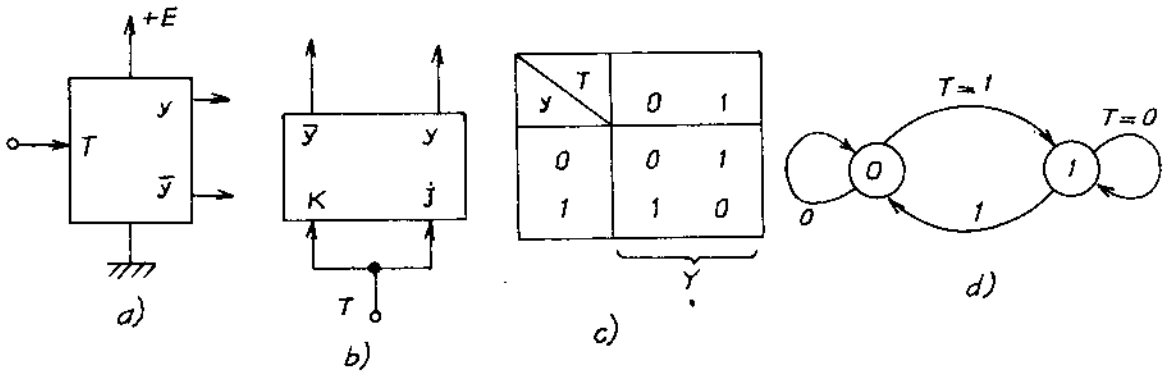
- Trường hợp kích (JK = 11) giả sử trigơ ở trạng thái "1" tức là (y = 1), do cách nối dây tín hiệu 1 đó đưa về nằm chờ ở đầu vào mạch VÀ. Khi có K = 1 (S = 1) thì mạch VÀ thông đưa tín hiệu kích vào đầu R làm cho trigơ JK lại về "0". Khi trigơ JK về "0" tức là (y = 0 còn $\bar{y}$ = 1), giá trị lôgic 1 ở đầu $\bar{y}$ lại được đưa về nằm chờ sẵn ở đầu mạch VÀ. Khi ta kích tiếp (J = 1 và K = 1) thì khi J = 1 lại làm mạch VÀ thông kích tín hiệu vào đầu S làm trigơ lại về "1". Nếu ta kích tiếp (JK = 11) thì trigơ lại về "0"..., như vậy khi ta kích tổ hợp 11 (JK = 11) thì trạng thái của trigơ luôn "đảo lại" trạng thái trước đó. Hiện tượng luôn đảo lại trạng thái trước đó giống như khi ta bấm vào phím công tắc nguồn của TV, nó luôn đảo lại trạng thái giữa bật và tắt của TV. Như vậy tổ hợp kích vào 11 (JK = 11) đã được sử dụng ở JKFF và hoạt động đó được biểu diễn ở bảng chức năng của trigơ JK.

Từ bảng chức năng hay hoạt động thật sự của trigơ JK chúng ta tìm được phương trình chuyển biến trạng thái của nó là :

$$Y = J\,\bar{y} + \bar{K}\,y$$

5.1.4. TRIGƠ T (T FLÍP FLỐP - TFF)

Trigơ cuối cùng được sử dụng trong các hệ thống số hiện nay là trigơ T, loại trigơ này cũng có hai trạng thái ổn định, muốn chuyển đổi trạng thái hay điều khiển hoạt động của TFF chúng ta dùng 1 đầu kích T, đáp ứng ra hay đầu ra của trigơ là y và $\bar{y}$. Trigơ T được minh họa trên hình 5.11.



Hình 5.11. Trigơ T.

Nguyên lý của trigơ T trình bày qua bảng trạng thái trên hình 5.11,c.

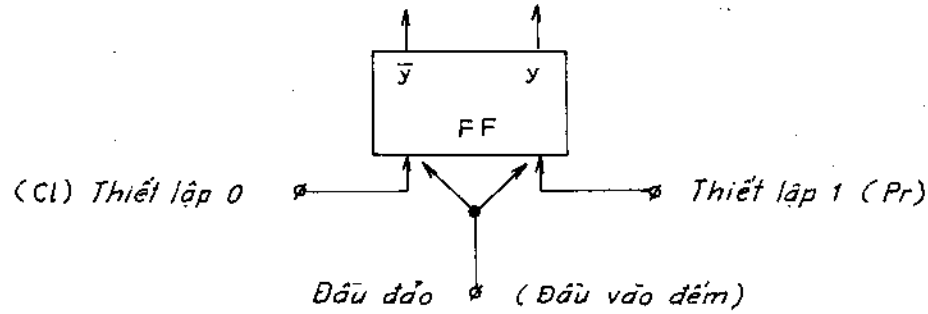
- Khi kích (T = 0) thì trạng thái tiếp theo Y giữ trạng thái đó y.

- Khi ta kích (T = 1) thì trạng thái tiếp theo Y luôn đảo lại so với trạng thái trước đó. Như vậy khi T = 1 thì tác động của nó giống như ở trigơ JK khi ta kích (JK = 11). Vì vậy nếu không có trigơ T thì ta có thể dùng trigơ JK khi chập đầu J vào đầu K hàn chúng lại với nhau.

Từ bảng chức năng của TFF ta tìm được phương trình chuyển biến trạng thái của nó là :

$$Y = T\,\bar{y} + \bar{T}\,y$$

Tới đây ta đã tìm hiểu xong nguyên lý và chức năng hoạt động thực tế (chính xác) của bốn loại trigơ (DFF, RSFF, JKFF và TFF). Tổng quát đối với một trigơ chúng ta có thể kích vào và lấy thông tin ra trên các đầu ra như chỉ ra ở trên hình 5.12.

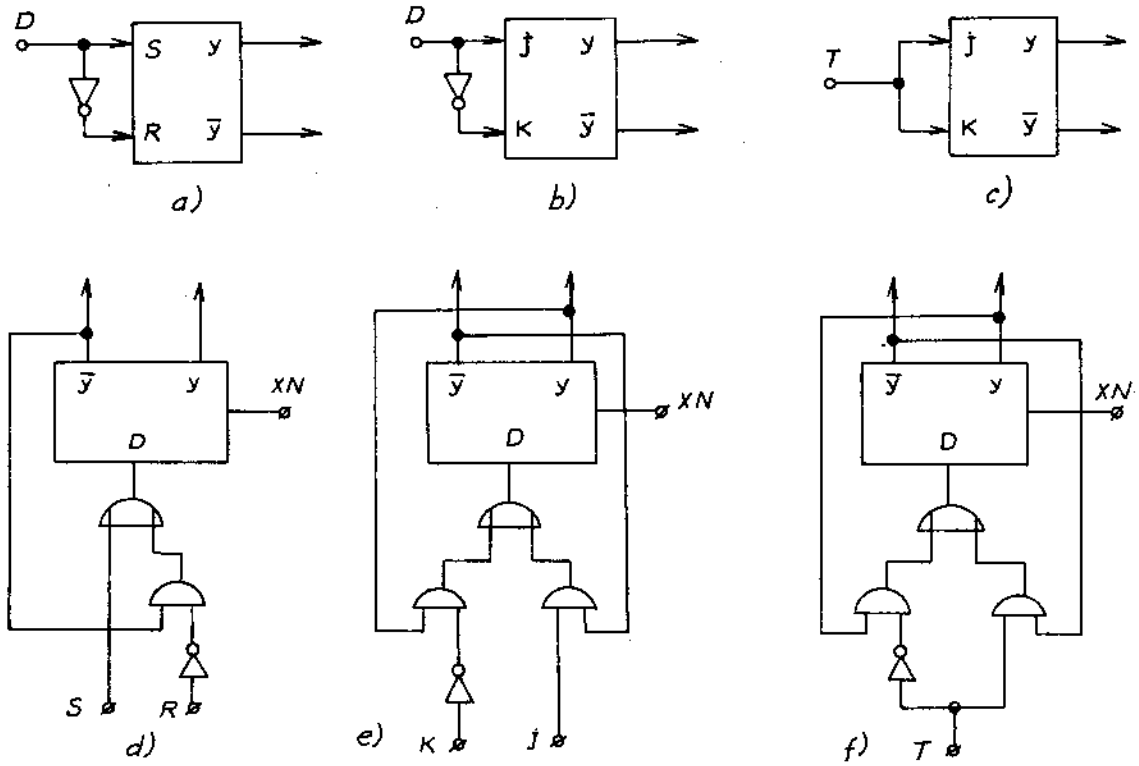


Hình 5.12. Trigơ tổng quát.

Một trigơ ngoài hai đầu vào kích để lật chuyển thiết lập trạng thái cho nó, trigơ còn có thêm đầu vào đảo (như đầu T của TFF) để đảo lại trạng thái trước đó của nó, đầu vào đảo còn có tên gọi là "đầu vào đếm", vì muốn xây dựng hay tổng hợp bộ đếm nhị phân tăng dần, thì ta phải dùng tới đầu kích đảo đó.

5.1.5. CHUYỂN ĐỔI GIỮA CÁC LOẠI TRIGƠ

Có một số cách chuyển đổi giữa các loại trigơ như trên hình 5.13.



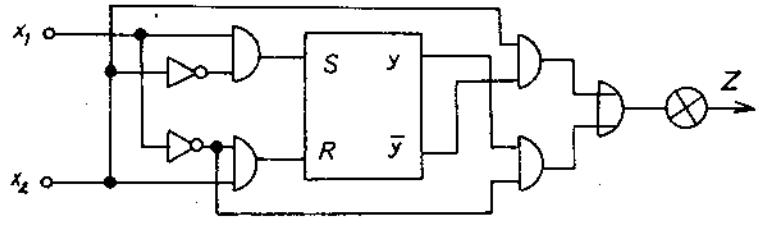
Hình 5.13. Chuyển đổi trigơ.

§5.2. PHÂN TÍCH ÔTÔMAT CÓ NHỚ

Phân tích chức năng ôtômat có nhớ là bài toán ngược, tức là từ một sơ đồ mạch điện của ôtômat có nhớ cho trước, chúng ta phải tìm lại bảng chức năng ban đầu của ôtômat đó hay của mạch điện đó. Từ trước tới nay việc sửa chữa một hệ thống điện tử bị hỏng thì thông thường chúng ta dựa vào các hiện tượng hỏng của hệ thống đó, rồi từ hiện tượng hỏng chúng ta suy đoán dần để dẫn tới phát hiện được chỗ sai mà sửa. Nhưng đối với hệ thống số thì hiện tượng hỏng hầu như không thấy (mặc dù nó đang hỏng thật sự), vì chỉ có đóng mở (sáng hay tối) là 0 hay 1 ở đầu ra Z của hệ thống, vì vậy một hệ thống số (một ôtômat có nhớ) bị hỏng chúng ta rất khó phát hiện ra chỗ sai hỏng để sửa chữa.

Để hiểu được trình tự hay các bước trong quá trình phân tích một ôtômat với mạch điện cho trước, chúng ta dựa vào ví dụ sau.

Ví dụ : Phân tích hay tìm lại bảng chức năng của ôtômat có nhớ theo sơ đồ mạch điện chỉ ra trên hình 5.14.



Hình 5.14. Cho một sơ đồ mạch điện.

Bước 1 : Từ sơ đồ tìm ra hệ phương trình hàm kích :

$$S = x_1 . \bar{x}_2$$

$$R = \bar{x}_1 . x_2$$

$$Z = \bar{x}_1 y + x_2 \bar{y}$$

Bước 2 : Từ hệ phương trình kích chúng ta tính được hay lập ra bảng kích, nó được chỉ ra ở hình 5.15. Do đáp ứng ra Z trong phương trình thấy phụ thuộc trực tiếp vào tín hiệu vào (x_1 và x_2) cho nên bảng chức năng sẽ có dạng của ôtômat Mealy.

Bảng kích : (bảng kích thế)

x_1x_2 / y	00		01	11	10	Z			
	S	R	S R	S R	S R	00	01	11	10
0	(0)	[0]	0 1	0 0	1 0	(0)	1	1	0
1	0	0	0 1	0 0	1 0	1	1	0	0

Hình 5.15. Bảng kích của ôtômat.

Cách tính hay cách điền bảng kích :

Chúng ta tính toán theo từng toạ độ (x_1 x_2, y) một :

- Với toạ độ đầu tiên (00, 0) ta có :

$x_1 = 0$, $x_2 = 0$ và $y = 0$ ta thay chúng vào hệ phương trình kích sẽ tính được kết quả :

$$S = x_1 . \bar{x}_2 = 0 . \bar{0} = (0)$$

$$R = \bar{x}_1 x_2 = \bar{0} . 0 = [0]$$

$$Z = \bar{x}_1 y + x_2 \bar{y} = \bar{0} . 0 + 0 . \bar{0} = (0)$$

Với cách làm tương tự ta tính cho các toạ độ khác và điền đầy kết quả vào bảng kích.

Ta có bảng chuyển biến trạng thái hay bảng chức năng của trigơ RS trên hình 5.16.

Bảng chức năng RSFF.

y / SR	00	01	11	10	
0	0	0	–	1	← kích thế
1	1	0	–	1	← thì được

Hình 5.16. Bảng RSFF.

Bước 3 : Tìm bảng mã hóa của ôtômat bằng cách kết hợp giữa bảng kích với bảng trạng thái của RSFF, kết quả bảng mã hóa được chỉ ra trên hình 5.17.

Bảng mã hóa : (bảng thì được)

	x_1x_2 / y	00	01	11	10	Z 00	Z 01	Z 11	Z 10
A	0	⓪	0	0	1	0	1	1	0
B	1	①	0	1	1	1	1	0	0

Hình 5.17. Bảng mã hóa thu được.

Sự kết hợp giữa bảng kích với bảng chức năng của RSFF tạo ra bảng mã hóa điều đó sẽ thấy ở cách điền bảng mã hóa sau.

Cách điền bảng mã hóa :

- Ở cột đầu tiên của bảng kích ta cho tín hiệu vào $x_1\ x_2 = 00$, từ tín hiệu vào (X) đó tạo ra tín hiệu kích trực tiếp vào trigơ là : SR = 00, đối với trigơ RS theo bảng chức năng của nó khi kích SR = 00 nghĩa là giữ nguyên trạng thái, cho nên trạng thái trước đó y = 0 thì trạng thái tiếp theo thu được Y = ⓪ ở trên bảng mã hóa (bảng "thì được"), còn khi y = 1 do kích SR = 00 thì thu được Y = ①.

- Với cách làm tương tự ta có ở cột thứ hai khi $x_1\ x_2 = 01$ cho vào, nó sẽ tạo ra tín hiệu kích vào RSFF là SR = 01, đối với RSFF kích vào SR = 01 nghĩa là thiết lập "0" cho nên kết quả thu được Y = 0.

- Trường hợp ở cột tiếp theo $x_1\ x_2 = 11$ ta có SR = 00 là giữ nguyên trạng thái nên y = 0 thì Y = 0 còn y = 1 thì trạng thái tiếp theo Y = 1.

- Đối với cột cuối cùng khi $x_1x_2 = 10$, tín hiệu vào đó đã tạo ra tín hiệu kích SR = 10 tức là thiết lập "1". Cho nên kết quả trạng thái tiếp theo trên bảng mã hóa Y = 1 cho dù bất kỳ trạng thái trước đó là gì.

Bảng mã hóa là bảng đặt tên trạng thái cho ôtômat, cho nên muốn tìm lại bảng chức năng của ôtômat đã cho thì ta phải quy ước của mã thành trạng thái. Do bảng mã hóa chỉ có một giá trị mã y = 0 và y = 1, cho nên ta quy ước tên trạng thái là 0 = A và 1 = B (trạng thái B).

Bước 4 : Bảng chức năng của ôtômat đã cho hay mạch điện đã cho, hay là hoạt động chính xác của sơ đồ ban đầu cho ở hình 5.14 của ví dụ đã đề ra thu được ở trên hình 5.18.

Như vậy để thực hiện phân tích một ôtômat có nhớ hay tìm lại bảng chức năng

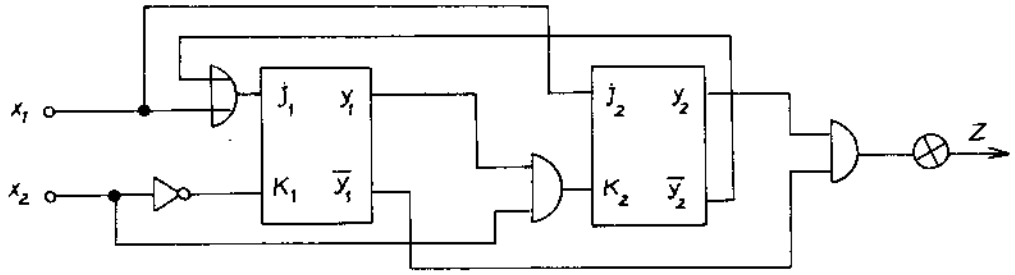
hoạt động của một sơ đồ mạch điện đã cho, chúng ta cần phải trải qua 4 bước làm tuần tự.

S \ x_1x_2	00	01	11	10	Z 00	Z 01	Z 11	Z 10
A	A	A	A	B	0	1	1	0
B	B	A	B	B	1	1	0	0

Hình 5.18. Bảng chức năng tìm được.

Tiếp sau đây chúng ta sẽ xét một ví dụ với một ôtômat có nhớ phức tạp hơn.

Ví dụ : Phân tích ôtômat có nhớ được biểu diễn trên hình 5.19.



Hình 5.19. Có mạch điện của ôtômat.

Để thực hiện phân tích ôtômat đã cho ta cũng lần lượt qua các bước sau :

Bước 1 : Từ sơ đồ mạch điện chúng ta tìm được hệ phương trình hàm kích có dạng :

$$J_1 = x_1 + \bar{y}_2 \qquad J_2 = x_1$$

$$K_1 = \bar{x}_2 \qquad K_2 = x_2 \cdot y_1$$

$$Z = \bar{y}_1\, y_2$$

Bước 2 : Từ hệ phương trình kích ta tính toán để tìm ra bảng kích như trên hình 5.20. Nhìn vào hệ phương trình thu được ta thấy đáp ứng ra Z không phụ thuộc trực tiếp vào tín hiệu vào (x_1, x_2) nên bảng chức năng sẽ mô tả theo ôtômat Moore.

Bảng kích (kích thế)

a)

y_1y_2 \ x_1x_2	00 J_1 K_1	00 J_2 K_2	01 J_1 K_1	01 J_2 K_2	11 J_1 K_1	11 J_2 K_2	10 J_1 K_1	10 J_2 K_2	Z
0 0	⚠①	0 0	1 0	0 0	1 0	1 0	1 1	1 0	0
0 1	0 1	0 0	0 0	0 0	1 0	1 0	1 1	1 0	1
1 0	1 1	0 0	1 0	0 1	1 0	1 0	1 1	1 0	0
1 1	0 1	0 0	0 0	0 1	1 0	1 0	1 1	1 0	0
	Y_1	Y_2							

Bảng chức năng của JKFF

b)

y \ JK	00	01	11	10	
0	0	0	1	1	kích thế
1	1	0	0	1	thì được
		Y			

Hình 5.20. Bảng kích (a) và bảng JKFF (b).

Cách tính tìm ra bảng kích :

Ta cũng tính theo từng tọa độ của bảng kích (x_1 x_2, y_1 y_2)

- Tọa độ đầu tiên (00, 00) ta có :

$$x_1 = 0 \qquad x_2 = 0 \qquad y_1 = 0 \qquad y_2 = 0$$

Với các giá trị đó chúng ta thay vào hệ phương trình kích và tính được các kết quả :

$$J_1 = x_1 + \bar{y}_2 = 0 + \bar{0} = 0 + 1 = 1$$

$$K_1 = \bar{x}_2 = \bar{0} = (1)$$

$$J_2 = x_1 = 0$$

$$K_2 = x_2 . y_1 = 0 . 0 = 0$$

$$Z = \bar{y}_1 . y_2 \text{ vậy } Z = 1 \text{ ứng với } \bar{y}_1 y_2 = 0\ 1$$

Ta điền các kết quả thu được vào bảng kích ứng với tọa độ đầu tiên đó.

- Các tọa độ còn lại của bảng kích chúng ta tính toán tương tự.

Với cách làm như vậy việc tính toán điền đầy bảng kích sẽ rất lâu, ta có cách tính toán nhanh hơn như sau :

Từ phương trình $J_2 = x_1$ nhận thấy giá trị của biến x_1 là như thế nào thì giá trị của J_2 sẽ giống như vậy, mà không phụ thuộc vào giá trị của các biến khác, nếu $x_1 = 0$ ở cột nào thì toàn bộ cột số của J_2 sẽ bằng 0 ($J_2 = 0$) ở cột đó, còn nếu $x_1 = 1$ thì cả cột số của J_2 bằng 1 ($J_2 = 1$), kết quả sẽ điền ngay được các giá trị của J_2 ở các cột của bảng kích.

Với cột $K_1 = \bar{x}_2$ cách làm cũng tương tự trên, chỉ chú ý ở đây là giá trị của cột K_1 được tính theo phủ định của biến x_2.

Còn từ phương trình kích $J_1 = x_1 + \bar{y}_2$ việc tính giá trị của J_1 bằng cách cho luôn giá trị của $x_1 = 1$ hoặc $x_1 = 0$, ta giả sử :

Nếu giá trị của $x_1 = 1$ thì $J_1 = x_1 + \bar{y}_2 = 1 + \bar{y}_2 = 1$ mà không phụ thuộc vào các biến khác còn lại (x_1, y_1), nên toàn bộ cột của J_1 chúng ta điền giá trị là 1.

Nếu giá trị của $x_1 = 0$ thì $J_1 = x_1 + \bar{y}_2 = 0 + \bar{y}_2 = \bar{y}_2$ vậy toàn bộ cột số của j_1 sẽ lấy giá trị của cột số $\bar{y}_2$ tương ứng với cột có $x_1 = 0$.

Với cách làm tương tự của cực kích $K_2 = x_2 \cdot y_1$, ở đây chúng ta cũng cho giá trị của $x_2 = 0$ và $x_2 = 1$ vào để tính ra K_2, và điền kết quả tính được của K_2 vào cột tương ứng.

Như vậy, nếu ta kích tín hiệu cho các trigơ như bảng kích, thì thu được trạng thái của các trigơ lật chuyển như bảng mã hóa. Tiếp theo là tìm bảng mã hóa.

Bước 3 : Bảng mã hóa (bảng thì được) các giá trị của nó sẽ thu được từ bảng kích kết hợp với bảng chuyển biến trạng thái của trigơ JK. Kết quả của bảng mã hóa được chỉ ra trên hình 5.21.

	y_1y_2 \ x_1x_2	00	01	11	10	Z
A	0 0	[1] 0	1 0	1 1	1 1	0
B	0 1	0 1	0 1	1 1	1 1	1
C	1 0	0 0	1 0	1 1	0 1	0
D	1 1	0 1	1 0	1 0	0 1	0

Hình 5.21. Bảng mã hóa thu được.

Trước khi tính hay điền giá trị vào bảng mã hóa ta cần lưu ý là trigơ JK có hai đầu vào kích là (JK), vì vậy muốn cần chuyển trạng thái của nó ta cần phải có hai tín hiệu kích, như vậy khi có hai tín hiệu kích vào ta sẽ được một trạng thái của một trigơ JK.

Cách điền bảng mã hóa :

Cách điền bảng mã hóa được thực hiện theo từng tọa độ của bảng, với từng cặp tín hiệu kích (JK) tương ứng, đồng thời các cặp tín hiệu kích hoạt động và cho ra kết quả hoàn toàn độc lập với nhau :

- Với tọa độ đầu tiên (x_1 x_2, y_1 y_2) = (00, 00) tức là với tín hiệu vào (x_1, x_2) và trạng thái trước đó (y_1, y_2) ở trong mạch của ôtômat, chúng đã tạo ra tín hiệu kích đối với trigơ JK đầu tiên là (J_1 K_1 = ⚠ (1)). Đồng thời trạng thái trước đó của trigơ đầu tiên là $y_1 = 0$, dưới tác động của tín hiệu vào J_1K_1 = ⚠ (1) là đảo lại, thì ta sẽ thu được trạng thái tiếp theo của trigơ đầu tiên ở trang bảng mã hóa là Y_1 = [1] .

Tương tự với toạ độ đầu tiên (00, 00) ta có bộ tín hiệu kích $J_2K_2 = 00$, như vậy trạng thái trước đó của trigơ JK thứ hai là $y_2 = 0$, dưới tác động của tín hiệu vào $J_2K_2 = 00$ tức là giữ nguyên trạng thái, thì kết quả thu được ở trên bảng mã hóa là $Y_2 = 0$.

- Bằng cách làm tương tự như trên ta tính toán và xác định trạng thái tiếp theo (kết quả thu được) của các trigơ ở các toạ độ khác trên bảng mã hóa, và cũng bằng

cách đó ta đã điền đầy được bảng mã hóa, hay thu được bảng mã hóa từ bảng kích kết hợp với bảng chuyển biến trạng thái của trigơ JK.

Tiếp theo từ bảng mã hóa chúng ta phải tìm lại bảng chức năng hay hoạt động thật sự của ôtômat đã cho, vì bảng mã hóa chính là bảng đặt tên cho các trạng thái của ôtômat, cho nên để có được bảng chức năng của ôtômat chúng ta gán tên hay gán mã cho ôtômat như sau :

$$00 = A \; ; \quad 01 = B \; ; \quad 10 = C \quad \text{và} \quad 11 = D$$

Như vậy ta thấy cứ hai trạng thái (có thể nhiều hơn) của hai trigơ JK sẽ cho ta một trạng thái của ôtômat.

Bước 4 : Bảng chức năng của ôtômat đã cho hay của sơ đồ mạch điện đã cho trong ví dụ đưa ra ở trên được biểu diễn ở trên hình 5.22.

Từ bảng chức năng của ôtômat chúng ta có thể phát hiện ra sự sai hỏng của mạch điện.

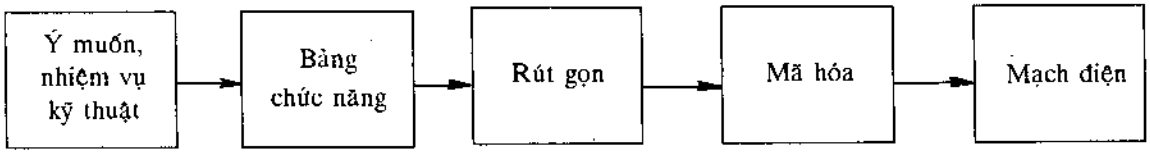
S \ x_1x_2	00	01	11	10	Z
A	C	C	D	D	0
B	B	B	D	D	1
C	A	C	D	B	0
C	B	C	C	B	0

Hình 5.22. Bảng chức năng.

Với cách làm như trên để tìm ra được bảng chức năng của một ôtômat, chúng ta có thể phát triển thành bài toán lớn hơn khi tăng số lượng các trigơ ở trong mạch, cũng như có thể tăng số lượng tín hiệu vào (đầu vào) của sơ đồ, đồng thời cũng có thể thay đổi cách mắc mạch tùy theo hoạt động của một ôtômat bất kỳ, hay trong mạch điện sử dụng hỗn hợp các loại trigơ khác nhau.

§5.3. TỔNG HỢP ÔTÔMAT CÓ NHỚ

Tổng hợp ôtômat có nhớ là bài toán thiết kế ôtômat hay còn gọi là bài toán thuận, tức là từ một nhiệm vụ hay chức năng đề ra, ta phải xây dựng được mạch điện thực hiện chức năng đó. Quy trình tổng hợp một ôtômat có nhớ được chỉ ra ở trên hình 5.23.



Hình 5.23. Quy trình tổng hợp ôtômat có nhớ.

5.2.1. BẢNG CHỨC NĂNG CỦA CÁC TRIGƠ (CHUYÊN DÙNG ĐỂ TỔNG HỢP)

Tổng hợp ôtômat có nhớ là quá trình ngược lại của bài toán phân tích, giữa bài toán tổng hợp ôtômat có nhớ và bài toán phân tích ôtômat có nhớ chúng có liên hệ hữu

cơ mật thiết với nhau. Để phân tích ôtômat có nhớ chúng ta đã sử dụng đến trigơ cụ thể là dùng bảng chuyển biến trạng thái hay là bảng chức năng của chúng – bảng chức năng của các trigơ chuyên dùng để phân tích. Để tổng hợp được một ôtômat có nhớ chúng ta cũng lại phải sử dụng tới trigơ, và phải sử dụng bảng chức năng của trigơ nhưng chuyên dùng để tổng hợp. Các bảng chức năng của các trigơ chuyên dùng để tổng hợp được biểu diễn trên hình 5.24.

$t_0 < t_1$

y	Y	D
0	0	0
0	1	1
1	0	0
1	1	1

a)

Muốn chuyển Thì kích

y	Y	S	R
0	0	0	–
0	1	1	0
1	0	0	1
1	1	–	0

b)

t_0 t_1

y	Y	J	K
0	0	0	–
0	1	1	–
1	0	–	1
1	1	–	0

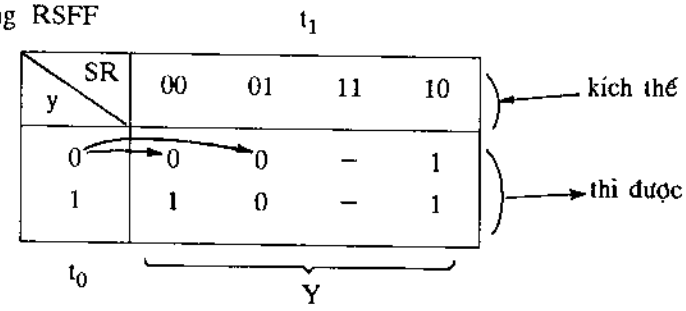
c)

t_0 t_1

y	Y	T
0	0	0
0	1	1
1	0	1
1	1	0

d)

Bảng RSFF



e)

Hình 5.24. Bảng chức năng của trigơ.

Để hiểu được bản chất hoạt động của bảng chuyển biến trạng thái (bảng chức năng) của các trigơ chuyên dùng để tổng hợp ôtômat có nhớ, chúng ta lấy ví dụ hoạt động của trigơ RS (RSFF) như sau :

Từ bảng chức năng hoạt động của RSFF trên hình 5.24,e chúng ta thấy : muốn chuyển trạng thái của nó từ trạng thái trước đó t_0 y = 0 sang trạng thái tiếp theo t_1 Y = 0, thì đầu kích vào cho trigơ RS sẽ là SR = 00 và SR = 01 (kích như vậy đều đảm bảo y = 0 sẽ chuyển thành Y = 0), từ đó ta thấy để có được (y = 0 chuyển thành Y = 0) thì S bắt buộc phải bằng 0 (S = 0) còn R thì có thể là 0 hoặc 1 (R = 0 hoặc 1), từ đó ta có thể nói : muốn chuyển từ trạng thái trước đó t_0 y = 0 thành trạng thái tiếp theo t_1 là Y = 0 thì khi ta kích S phải bằng 0 còn đầu R thì kích tùy ý (là 0 hay 1 cũng được) hay ký hiệu (R = –). Như vậy ở bảng chức năng trên hình 5.24,b của trigơ RS nếu ta muốn chuyển trạng thái từ (y = 0 sang Y = 0) thì ta phải kích vào (SR = 0 –).

Tương tự nếu ta muốn chuyển trạng thái của trigơ RS từ trạng thái trước đó y = 0 thành trạng thái tiếp theo Y = 1, thì ta cần phải kích vào đầu kích của nó (SR = 10).

Với cách hiểu như vậy ở các quá trình lật chuyển trạng thái khác của RSFF ta sẽ có các tín hiệu kích cần thiết thích hợp để lật chuyển trạng thái cho nó, như chỉ ra ở bảng chức năng chuyên dùng để tổng hợp trên hình 5.24,b.

Cũng giống như vậy, với các trigơ khác (DFF, JKFF hay TFF) chúng ta cũng có được quan hệ giữa tín hiệu kích của chúng để chủ động tạo ra được trạng thái tiếp theo thích hợp cho từng loại trigơ đó, gọi là bảng chuyển biến trạng thái của các trigơ chuyên dùng để thực hiện trong quá trình tổng hợp (hay thiết kế) ôtômat có nhớ. Các bảng chức năng của từng trigơ chuyên dùng để tổng hợp chúng ta đã mô tả đầy đủ trên hình 5.24. Như vậy vẫn là bảng chức năng của trigơ, nhưng do nhiệm vụ là dùng để phân tích hay dùng để tổng hợp mà chúng có dạng khác nhau mà thôi.

Sau khi đã hiểu được bản chất hoạt động của bảng trạng thái của các trigơ sẽ được dùng khi thiết kế ôtômat có nhớ, đến đây vấn đề còn lại là sẽ áp dụng cụ thể các bảng chức năng đó của các trigơ như thế nào trong việc tổng hợp ôtômat có nhớ, cũng như trong việc tạo ra bảng kích cho ôtômat có nhớ sau này.

5.3.2. THỦ TỤC THỰC HIỆN TỔNG HỢP ÔTÔMAT CÓ NHỚ

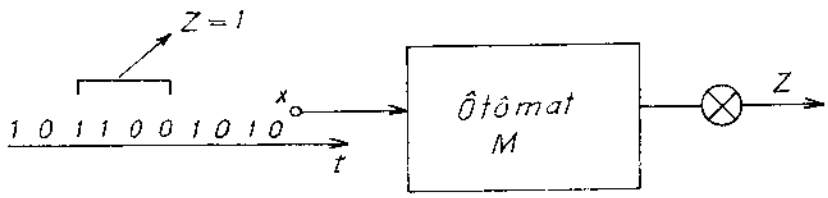
Muốn thực hiện tổng hợp một ôtômat có nhớ sẽ phải trải qua nhiều bước, một trong những bước quan trọng nhất là thực hiện chuyển từ một ý muốn của ta hay một nhiệm vụ kỹ thuật được đề ra trở thành bảng chứ năng. Tiếp theo sau khi có bảng chứ năng ta phải rút gọn trạng thái cho nó, rồi mã hóa trạng thái cho ôtômat đã được rút gọn và kết hợp với bảng chức năng của trigơ để tạo ra bảng kích. Từ bảng kích ta sẽ tìm được hệ phương trình kích (chuyển một ý muốn nhiệm vụ thành toán 0, 1) rồi lắp ráp sơ đồ mạch điện thực hiện nhiệm vụ đề ra.

Để minh họa các bước phải thực hiện trong quá trình xây dựng một ôtômat có nhớ ta khảo sát một ví dụ sau.

Ví dụ : Tổng hợp một ôtômat có nhớ có một đầu vào x và một đầu ra Z hoạt động theo yêu cầu sau :

- Tín hiệu vào là 0 hoặc 1 xuất hiện ngẫu nhiên bất kỳ, liên tục và kế tiếp nhau.

- Đầu ra Z = 1 (có đèn báo) nếu như gặp dãy số ở đầu vào x là 0 0 1 1 hoặc nếu gặp dãy số vào x là 1 1 0 0.



Hình 5.25. Chức năng ôtômat phải thực hiện.

- Đầu ra Z = 0 trong mọi trường hợp gặp khác đối với dãy số đầu vào x.

- Dùng trigơ JK để thực hiện, mô tả ôtômat theo cách biểu diễn của ôtômat Mealy.

Sơ đồ khối của ôtômat cần phải được xây dựng được biểu diễn trên hình 5.25.

Bước 1 : Chuyển ý muốn hay một nhiệm vụ kỹ thuật thành bảng chức năng. Bảng chức năng phải được mô tả theo cách biểu diễn của Mealy (M_1) do nhiệm vụ yêu cầu.

Trước khi tìm cách chuyển nhiệm vụ thành bảng chức năng chúng ta cần phải lưu ý là : các trạng thái của ôtômat (hay ở bảng chức năng) cần phải ký hiệu trạng thái sao cho dễ dàng nhận biết hoạt động, và chỉ cần nhìn vào nó (vào ký hiệu chức năng) là đã có thể biết được hoạt động của ôtômat đã đến đâu, từ đó dễ dàng theo dõi hoạt động của nó trong quá trình điền bảng trạng thái sau này.

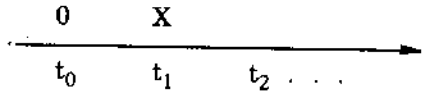
Do nhiệm vụ đề ra là : tín hiệu vào là 0 hoặc 1 xuất hiện ngẫu nhiên bất kỳ, cho nên ta có thể gặp được dãy số vào là 0011 (hoặc 1100) là hoàn toàn ngẫu nhiên, có thể gặp được dãy số đó cũng như có thể không bao giờ gặp được chúng. Muốn chuyển chức năng cho trước thành bảng ta phải tìm được hết các khả năng có thể gặp của ôtômat đó, không được bỏ sót trạng thái nào.

Trước hết cần phải phân biệt được đang cho 0 vào hay đang cho 1 vào đầu vào x, ta quy ước trạng thái là :

q_0 : là ta đang cho 0 vào đầu x ;

q_1 : là trạng thái ứng với việc cho 1 vào đầu vào x.

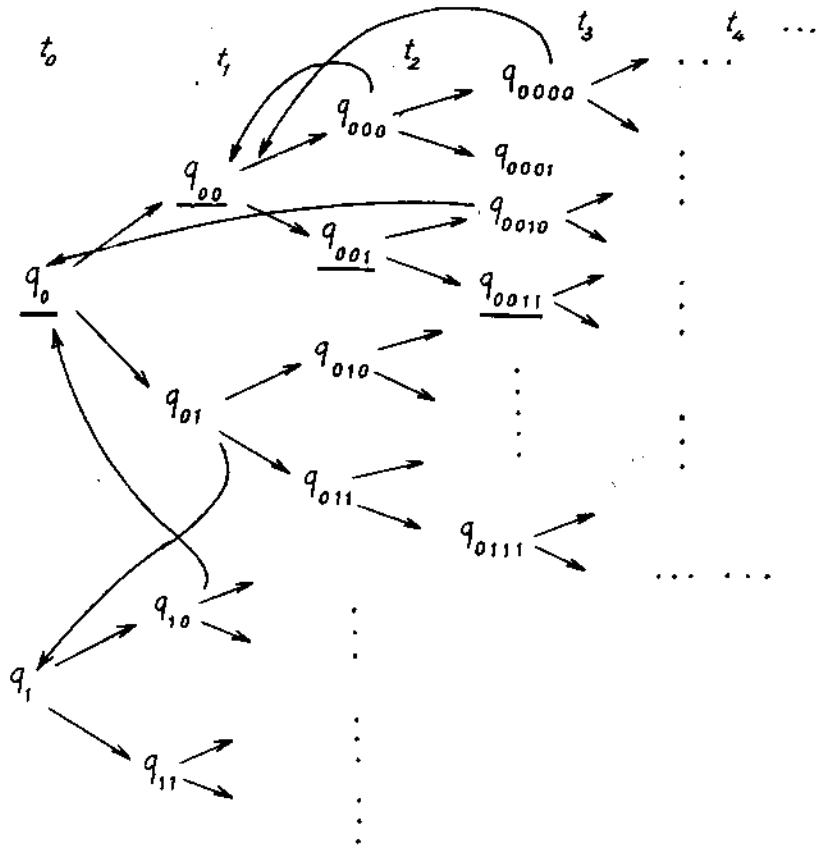
Giả sử ta đã cho 0 vào tại thời điểm t_0, thì tại thời điểm t_1 sẽ có thể xuất hiện giá trị 0 hoặc giá trị 1.



Nếu tại t_0 ta đã có trạng thái q_0, thì tại thời điểm tiếp theo t_1 ta có thể có các trạng thái là q_{00} hoặc q_{01}. Tương tự như vậy tại thời điểm t_2 cũng có thể có hai giá trị hoặc 0 hoặc 1 xuất hiện, và có được trạng thái mới được tạo ra như trên hình 5.26 sau.

Với cách tìm ra trạng thái mới theo các khả năng xuất hiện ngẫu nhiên của hai giá trị 0 hoặc 1 ở đầu vào x, ta thấy số lượng trạng thái khác nhau sẽ rất lớn và khó có thể thực hiện.

Nhưng nhiệm vụ đề ra chỉ yêu cầu (0 0 1 1) tức là chỉ cần có hai giá trị (00) để còn chờ đợi hai ẩn số chưa biết (0 0 X X), như vậy nếu xuất hiện tại thời điểm t_2 trạng thái q_{000} thì số 0 đầu tiên ta có thể bỏ qua và chỉ xét q_{00} ở hai giá trị 00 sau cùng. Như vậy ta có thể hiểu : trạng thái tiếp theo q_{000} được quy về trạng thái là q_{00}, tức là trạng thái tiếp theo được quy về trạng thái trước đó, điều đó dẫn đến các trạng thái tiếp theo sau nữa của q_{000} như q_{0000}, q_{0001} ... sẽ hoàn toàn không thể xuất hiện được nữa, nên số lượng trạng thái sẽ giảm bớt, mà không bỏ sót khả năng nào.



Hình 5.26. Các trạng thái của ôtômat.

Trường hợp quy về tiếp theo là tại thời điểm t_3 nếu xuất hiện ở đầu x, giá trị 0, trong khi trạng thái trước đó đã là q_{001} (rất gần tới yêu cầu 0011) sẽ tạo ra trạng thái là q_{0010}, thì khi có trạng thái mới này coi như việc tìm kiếm không còn tác dụng, nên q_{0010} sẽ được quy về trạng thái ban đầu q_0, để rồi ta lại kiểm tra tiếp các giá trị 0 hoặc 1 ngẫu nhiên khác được đưa vào. Từ đó các trạng thái tiếp theo sau của q_{0010} cũng sẽ không bao giờ xuất hiện được nữa.

Một trường hợp ta cần quan tâm nữa là : tại thời điểm t_0 giả sử ta đã có trạng thái q_0, nhưng trạng thái tiếp theo tại t_1 lại có giá trị đầu vào x là 1 (x = 1) vì xuất hiện ngẫu nhiên, làm cho kết quả tạo ra trạng thái q_{01}, trong trường hợp này do yêu cầu đề ra là cần hai số 0 (0 0 X X) mà lại xuất hiện q_{01} nên trạng thái này ta có thể quy về q_1 để xét tiếp (1 X X X) với hy vọng tìm được dãy số (1 1 0 0) như yêu cầu đề ra cho ôtômat. Khi q_{01} được quy về trạng thái trước đó q_1 thì các trạng thái tiếp theo nữa của q_{01} như q_{010}, q_{011} ... cũng không được xuất hiện, nên số lượng trạng thái của ôtômat sẽ giảm đáng kể.

Tóm lại từ nhiệm vụ đề ra và các trạng thái tiếp theo sẽ được quy về các trạng thái trước đó (các trạng thái tiến gần đến yêu cầu), với lượng thông tin cho mỗi một trạng thái sẽ tăng dần lên, cho nên trạng thái của ôtômat chỉ còn lại là :

x = (0 0 1 1) ta có các trạng thái q_0, q_{00}, q_{001}, q_{0011}

x = (1 1 0 0) ta có các trạng thái q_1, q_{11}, q_{110} và q_{1100}

Như vậy sau khi xem xét và phân tích thì số lượng trạng thái của ôtômat ta phải xây dựng thấy nó chỉ còn có 8 trạng thái đó mà thôi, và 8 trạng thái đó đại diện cho mọi trạng thái có thể có (có thể xảy ra khi đầu vào x giá trị 0 hay 1 xuất hiện ngẫu nhiên của ôtômat). Đến đây vấn đề còn lại chỉ là điền vào bảng chức năng các trạng thái đó trong từng trường hợp cụ thể, ta sẽ tìm ra được bảng chức năng của ôtômat đề ra theo yêu cầu ở trên.

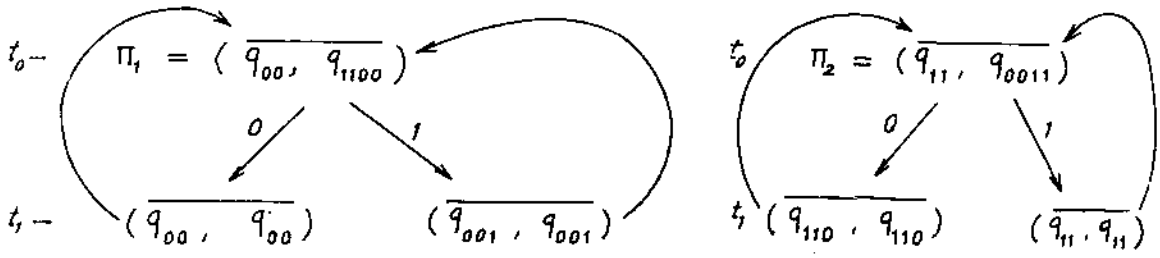
Chú ý là bảng chức năng phải được mô tả theo ôtômat Mealy, bảng chức năng đó được chỉ ra ở trên hình 5.27.

S \ x	0	1	Z: 0	Z: 1
q_0	q_{00}	q_1	0	0
q_1	q_0	q_{11}	0	0
q_{00}	q_{00}	q_{001}	0	0
q_{11}	q_{110}	q_{11}	0	0
q_{001}	q_0	q_{0011}	0	1
q_{110}	q_{1100}	q_1	1	0
q_{0011}	q_{110}	q_{11}	0	0
q_{1100}	q_{00}	q_{001}	0	0

Hình 5.27. Chức năng của ôtômat.

Bước 2 : Từ bảng chức năng của ôtômat cần tổng hợp ta tìm được, bước tiếp theo là thực hiện rút gọn nó theo phương pháp tìm phân hoạch thế (PHT) và phân hoạch tương đương (PHTĐ).

Cặp phân hoạch thế chúng ta tìm được theo hình 5.28.



Hình 5.28. Tìm các phân hoạch thế.

Qua hoạt động thật sự của ôtômat ta thấy : dưới tác động của tín hiệu vào là 0 và là 1 thì trạng thái tiếp theo lại trở lại với trạng thái trước đó, nên ta có các phân hoạch thế là :

$$\Pi_1 = \overline{(q_{00}\ ,\ q_{1100})}$$

$$\Pi_2 = \overline{(q_{11}\ ,\ q_{0011})}$$

Tiếp theo ta kiểm tra ngay các phân hoạch thế tìm được đó xem có phải là phân hoạch tương đương (PHTĐ) hay không.

Ví dụ với $\Pi_1 = \overline{(q_{00}\ ,\ q_{1100})}$ ta kiểm tra điều kiện của (PHTĐ) theo Mealy như sau :

$$g\ (q_{00},\ 0) = 0 = g\ (q_{1100},\ 0) = 0$$

$$g\ (q_{00},\ 1) = 0 = g\ (q_{1100},\ 1) = 0$$

Vậy ta có thể kết luận $\Pi_1 = \overline{(q_{11}\ ,\ q_{0011})}$ là phân hoạch tương đương (PHTĐ).

Tương tự với $\Pi_2 = \overline{(q_{11}\ ,\ q_{0011})}$ ta kiểm tra tiếp :

$$g\ (q_{11},\ 0) = 0 = g\ (q_{0011},\ 0) = 0$$

$$g\ (q_{11},\ 1) = 0 = g\ (q_{0011},\ 1) = 0$$

Như vậy ta cũng thấy $\Pi_2 = \overline{(q_{11}\ ,\ q_{0011})}$ cũng là phân hoạch tương đương (PHTĐ).

Tóm lại phân hoạch Π_1 và Π_2 vừa là phân hoạch thế lại là phân hoạch tương đương, nên chúng là phân hoạc thế tương đương (PHTTĐ), hay nói cách khác là ta có thể khẳng định là chúng có thể thay thế trạng thái cho nhau, cụ thể là :

$$q_{00} \sim q_{1100} \quad \text{và} \quad q_{11} \sim q_{0011}$$

Tức là q_{00} có thể thay thế cho q_{1100}, và cứ chỗ nào trong bảng ghi trạng thái là q_{1100} thì vị trí trạng thái đó ta sẽ thay bằng q_{00}. Tương tự như vậy q_{11} có thể được thay vào vị trí của trạng thái q_{0011} trong bảng chức năng.

Sau khi thay thế các trạng thái vào bảng chức năng chúng ta sẽ xóa đi được các dòng giống hệt nhau trong bảng chức năng, ta sẽ có được ôtômat có nhớ với số lượng trạng thái tối thiểu, hay còn gọi là ôtômat min. Ôtômat đã rút gọn được chỉ ra trên hình 5.29.

S \ x			Z	
	0	1	0	1
q_0	q_{00}	q_1	0	0
q_1	q_0	q_{11}	0	0
q_{00}	q_{00}	q_{001}	0	0
q_{11}	q_{110}	q_{11}	0	0
q_{001}	q_0	q_{11}	0	1
q_{110}	q_{00}	q_1	1	0

Hình 5.29. Bảng chức năng rút gọn.

Bước 3 : Mã hóa trạng thái cho ôtômat min hay "đặt tên" cho các trạng thái của bảng chức năng. Như chúng ta đã biết việc đặt tên hay mã hóa trạng thái cho ôtômat có nhiều cách khác nhau. Trong phần này sẽ giới thiệu cách mã hóa trạng thái cho ôtômat min dùng một phân hoạch thế, khi dùng phân hoạch thế để mã hóa cho phép tạo ra được tập tự phụ thuộc, điều đó cho phép ta có thể thực hiện ôtômat bằng mạch đẹp và gọn gàng.

a) Từ bảng ôtômat min đã thu được người ta tìm được một phân hoạch thế sau :

$$\Pi = \overline{(q_0, q_{00}, q_{11}}\ ;\ \overline{q_1, q_{11}, q_{001})}$$

Do có 6 trạng thái ôtômat cho nên ta cần có 3 biến để mã hóa trạng thái (y_1, y_2, y_3). Giả sử ta dùng biến (y_1) dùng để mã hóa khối, còn hai biến (y_2 y_3) dùng để mã hóa cho các phần tử của khối. Ta có bảng mã hóa hay bảng đặt tên cụ thể trạng thái như sau ở trên hình 5.30.

y_2y_3 \ y_1	0	1
0 0	q_0	q_1
0 1	q_{00}	q_{11}
1 0	q_{110}	q_{001}
1 1	–	–

q_0 – 000
q_1 – 100
q_{00} – 001
q_{11} – 101
q_{110} – 010
q_{001} – 110

Hình 5.30. Bảng mã trạng thái.

Sau khi có mã (tên) cho từng trạng thái của ôtômat rất cụ thể, ta có được bảng mã hóa của ôtômat như chỉ ra trên hình 5.31.

Bảng mã hóa của ôtômat (viết theo nhị phân tăng dần)

	x / $y_1y_2y_3$			Z	
		0	1	0	1
q_0	0 0 0	0 0 1	1 0 0	0	0
q_{00}	0 0 1	0 0 1	1 1 0	0	0
q_{110}	0 1 0	0 0 1	1 0 0	1	0
–	0 1 1	–	–	–	–
q_1	1 0 0	0 0 0	1 0 1	0	0
q_{11}	1 0 1	0 1 0	1 0 1	0	0
q_{001}	1 1 0	0 0 0	1 0 1	0	1
–	1 1 1	–	–	–	–
		$Y_1Y_2Y_3$	$Y_1Y_2Y_3$	a)	

y	Y	J	K
0	0	0	–
0	1	1	–
1	0	–	1
1	1	–	0

b)

Hình 5.31. Bảng mã hóa của ôtômat.

Sau khi mã hóa xong các trạng thái của ôtômat và xây dựng xong bảng mã hóa chức năng cho ôtômat đó, tức là chúng ta có bảng "muốn thế", muốn các trigơ lật chuyển như bảng mã hóa. Để thực hiện được việc lật chuyển trạng thái các trigơ như bảng mã hóa yêu cầu thì bước tiếp theo chúng ta phải xây dựng bảng kích.

Bước 4 : Tìm bảng kích, muốn tìm được bảng kích là bảng "thì kích" chúng ta phải kết hợp giữa hai bảng : bảng mã hóa trạng thái chức năng của ôtômat ở hình 1.6a với bảng chức năng chuyên dùng để tổng hợp ôtômat của trigơ JK ở trên hình 5.31,b, vì bảng mã hóa trạng thái của ôtômat chính là bảng yêu cầu lật chuyển trạng thái của các trigơ, mà muốn lật chuyển được trạng thái của trigơ JK theo ý ta, thì phải căn cứ vào bảng chuyển biến trạng thái của trigơ theo tín hiệu kích vào, hay đó chính là bảng lật chuyển trạng thái thực sự của nó, từ bản chất nhân quả đó chúng ta tìm được bảng kích theo yêu cầu. Bảng kích được chỉ ra ở hình 5.32. Trong quá trình tìm bảng kích chúng ta cũng sẽ biết được ứng dụng cụ thể của bảng chức năng của trigơ chuyên dùng để tổng hợp như thế nào.

Bảng kích :

	x	0						1						Z	
	$y_1y_2y_3$	J_1	K_1	J_2	K_2	J_3	K_3	J_1	K_1	J_2	K_2	J_3	K_3	0	1
q_0	0 0 0	0	–	0	–	1	–	1	–	0	–	0	–	0	0
q_{00}	0 0 1	0	–	0	–	–	0	1	–	1	–	–	1	0	0
q_{110}	0 1 0	0	–	–	1	1	–	1	–	–	1	0	–	1	0
–	0 1 1	–	–	–	–	–	–	–	–	–	–	–	–	–	–
q_1	1 0 0	–	1	0	–	0	–	–	0	0	–	1	–	0	0
q_{11}	1 0 1	–	1	1	–	–	1	–	0	0	–	–	0	0	0
q_{001}	1 1 0	–	1	–	1	0	–	–	0	–	1	1	–	0	1
–	1 1 1	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Hình 5.32. Bảng kích của ôtômat.

Cách tìm ra bảng kích, tức là tìm ra bảng của các tín hiệu kích cụ thể, để đảm bảo cho các trigơ lật chuyển trạng thái như bảng mã hóa yêu cầu như sau :

Trước hết nhìn vào bảng mã hóa ở hình 5.31,a theo toạ độ đầu tiên ta thấy trigơ đầu tiên (y_1) quá trình chuyển biến trạng thái của nó từ trạng thái trước đó $y_1 = 0$ sang trạng thái tiếp theo $Y_1 = 0$, như vậy là chuyển trạng thái từ 0 sang 0 ($y_1 = 0$ thành $Y_1 = 0$). Dựa theo bảng chuyển biến trạng thái của trigơ JK trên hình 5.31,b chúng ta thấy muốn chuyển được trigơ JK từ 0 sang 0 (y = 0 sang Y = 0) thì tín hiệu kích vào phải thỏa mãn (JK = 0 –), từ đó ta suy ra với trigơ JK đầu tiên ($y_1 = 0$ thành $Y_1 = 0$) cần phải có tín hiệu kích vào đầu vào của trigơ đầu tiên đó là (J_1 K_1 = 0 –) tín hiệu kích đó được ghi vào bảng kích ở hình 5.32.

Với cách làm tương tự : từ bảng mã hóa ta thấy ở toạ độ đầu tiên $y_2 = 0$ chuyển thành $Y_2 = 0$, thì ở bảng kích ta phải kích tín hiệu ($J_2K_2 = 0 -$) sẽ đạt được yêu cầu chuyển đổi trạng thái của trigơ thứ 2.

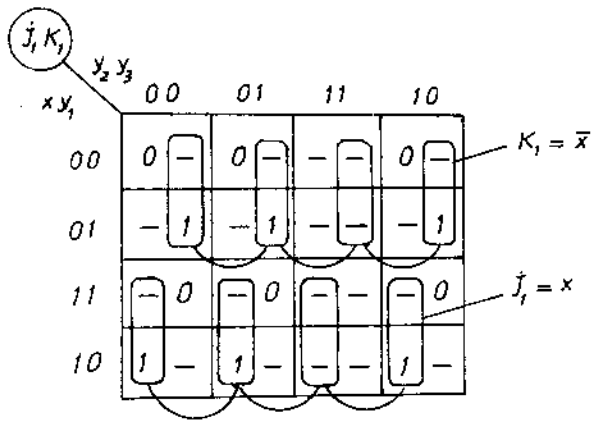
Tương tự ở trigơ thứ 3 trong toạ độ đầu tiên ở bảng mã hóa ta thấy $y_3 = 0$ phải chuyển thành $Y_3 = 1$, vậy tín hiệu kích ở bảng kích sẽ phải là ($J_3K_3 = 1 -$) thì trigơ thứ 3 sẽ thỏa mãn việc chuyển trạng thái.

Chúng ta cứ làm lần lượt như vậy đối với từng toạ độ của bảng mã hóa, cho từng trigơ cụ thể một cách lần lượt, ta sẽ điền đầy được các tín hiệu kích rất cụ thể vào bảng kích.

Tóm lại để yêu cầu các trigơ phải lật chuyển trạng thái như bảng mã hóa, thì ta phải kích tất cả các tín hiệu kích như bảng kích. Muốn có được các tín hiệu kích một cách cụ thể theo như bảng kích yêu cầu, thì ta cần phải có mạch điện cụ thể hoạt động theo hệ phương trình hàm kích. Tức là từ bảng kích chúng ta phải tìm ra hệ phương trình hàm kích và ta bước sang thủ tục tiếp theo.

Bước 5 : Từ bảng kích ta tìm ra hệ phương trình hàm kích như sau : ánh xạ vào bìa các tín hiệu của từng cực kích một.

Ví dụ : Ánh xạ J_1K_1 vào bìa Kácnô, kết quả thu được ở trên hình 5.33.



Hình 5.33. Ánh xạ J_1K_1 vào bìa.

Sau khi ánh xạ vào bìa Kácnô rút gọn ta được

$$J_1 = x \qquad K_1 = \bar{x}$$

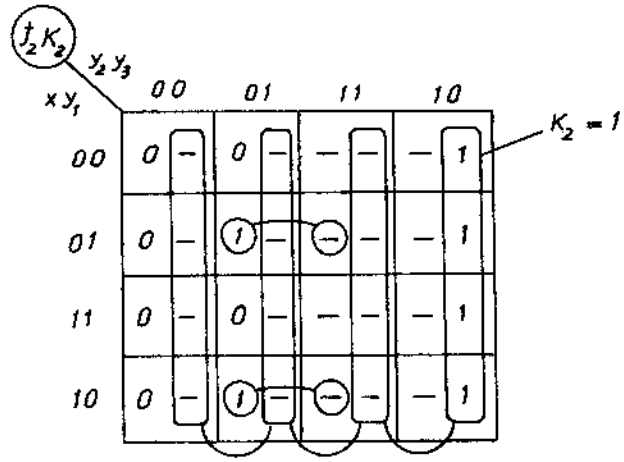
Tiếp theo ánh xạ J_2K_2 vào bìa Kácnô ở trên hình 3.34.

Rút gọn ta thu được phương trình kích :

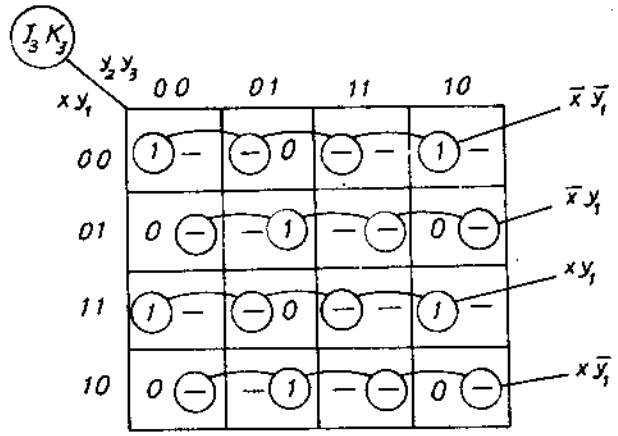
$$J_2 = \bar{x}\, y_1\, y_3 + x\, \bar{y}_1\, y_3 \qquad K_2 = 1 \text{ (nối } K_1 \text{ lên } +E).$$

$$J_2 = y_3\, (\bar{x}\, y_1 + x\, \bar{y}_1) = y_3 K_3$$

Ánh xạ J_3 K_3 vào bìa Kácnô ở trên hình 5.35.



Hình 5.34. Ánh xạ J_2K_2 vào bìa K.



Hình 5.35. Ánh xạ J_3K_3 vào bìa K.

Rút gọn ta thu được :

$$J_3 = \bar{x}\,\bar{y}_1 + x\,y_1 = \overline{\overline{K}_3} \qquad K_3 = \bar{x}\,y_1 + x\,\bar{y}_1$$

Sau khi rút gọn chúng ta có hệ phương trình kích là :

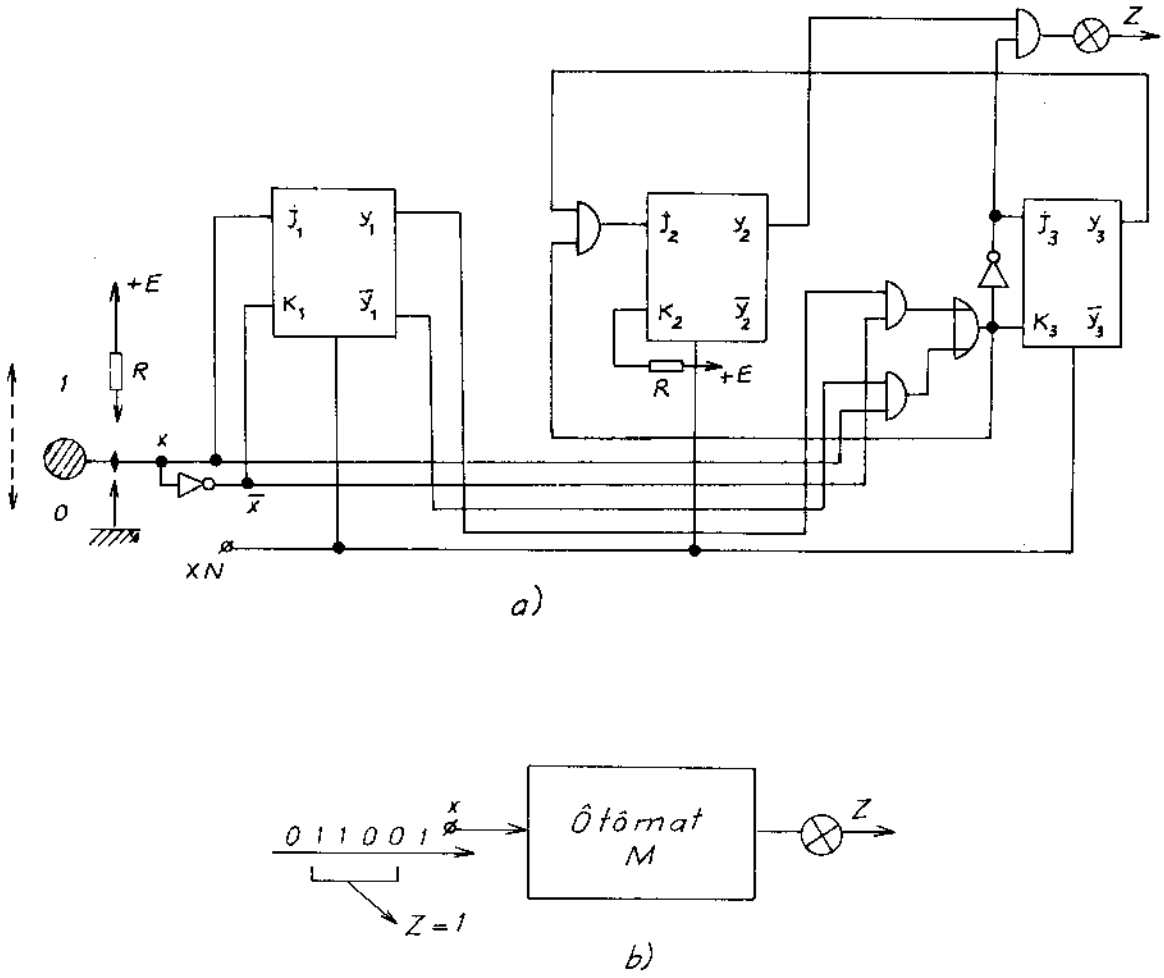
$$J_1 = x \qquad J_2 = y_3\,K_3 \qquad J_3 = \overline{K}_3$$

$$K_1 = \bar{x} \qquad K_2 = 1 \qquad K_3 = \bar{x}\,y_1 + x\,\bar{y}_1$$

$$Z = \bar{x}\,\bar{y}_1 y_2 + x\,y_1 y_2 = y_2(\bar{x}\,\bar{y}_1 + x\,y_1) = y_2\,J_3$$

Từ hệ phương trình kích đã được rút gọn chúng ta vẽ ra mạch điện của ôtômat có nhớ hoạt động theo yêu cầu đề ra ở trên hình 5.36.

Đến đây chung ta đã có mạch điện cụ thể thực hiện nhiệm vụ đã đề ra của ôtômat có nhớ, với yêu cầu Z = 1 hay đèn sáng lên chỉ trong trường hợp nếu gặp dãy số đầu vào là 0011 hoặc 1100, đồng thời cũng kết thúc các bước thực hiện tổng hợp ôtômat có nhớ theo một ý muốn hay một nhiệm vụ kỹ thuật được đưa ra.



Hình 5.36. Sơ đồ mạch điện của ôtômat phải thực hiện.

Điều đáng chú ý ở đây là mạch điện đã được thực hiện theo cách mã hóa dùng một phân hoạch thế, và chúng ta có cách tìm hệ phương trình kích bằng cách ánh xạ cực kích vào bìa Kácnô. Nhưng nếu chúng ta mã hóa biểu khác cho ôtômat, thì mạch điện sẽ khác đi mặc dù vẫn cùng một chức năng đề ra.

b) Phần tiếp theo đây sẽ vẫn thực hiện ôtômat có nhớ đã cho, nhưng mã hóa cách khác, đồng thời cho chúng ta biết thêm một phương pháp khác có thể tìm được hệ phương trình kích. Chúng ta xuất phát từ ôtômat min ở trên hình 5.37.

Bước 3 : Mã hóa trạng thái cho ôtômat theo quy luật của bìa Kácnô như trên hình (ví dụ q_0 = 000, q_1 = 001, ..., q_{110} = 111), từ đó chúng ta thu được bảng mã hóa biểu diễn ở hình 5.38 như sau :

Bước 4 : Từ bảng mã hóa của ôtômat và bảng chuyển biến trạng thái của JKFF chuyên dùng để tổng hợp biểu diễn ở hình 5.38,b, kết hợp chúng lại, chung ta có bảng "thì kích" bảng kích được biểu diễn trên hình 5.39 như sau.

Ôtômat min :

	x / 3	0	1	Z 0	Z 1
000	q_0	q_{00}	q_1	0	0
001	q_1	q_0	q_{11}	0	0
011	q_{00}	q_{00}	q_{001}	0	0
010	q_{11}	q_{110}	q_{11}	0	0
110	q_{001}	q_0	q_{11}	0	1
111	q_{110}	q_{00}	q_1	1	0

Hình 5.37. Ôtômat rút gọn.

Bảng mã hóa biểu diễn theo bìa Kácnô, bảng "muốn thế"

	x / $y_1y_2y_3$	0	0	Z 0	Z 1
q_0	0 0 0	0 1 1	0 0 1	0	0
q_1	0 0 1	0 0 0	0 1 0	0	0
q_{00}	0 1 1	0 1 1	1 1 0	0	0
q_{11}	0 1 0	1 1 1	0 1 0	0	0
q_{001}	1 1 0	0 0 0	0 1 0	0	1
q_{110}	1 1 1	0 1 1	0 0 1	1	0
–	1 0 1	——	——	–	–
–	1 0 0	——	——	–	–

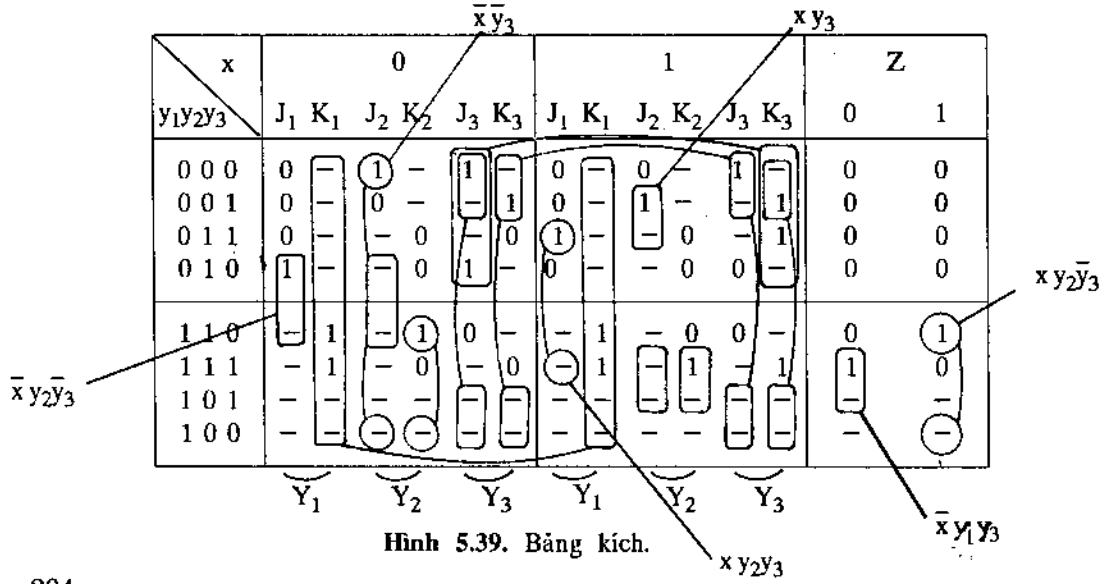
a)

y	Y	J	K
0	0	0	–
0	1	1	–
1	0	–	1
1	1	–	0

b)

Hình 5.38. Bảng mã hóa của ôtômat.

Bảng kích của ôtômat biểu diễn theo bìa Kácnô trên hình 5.39.



Hình 5.39. Bảng kích.

Do bảng kích đã được biểu diễn dưới dạng trật tự của bìa Kácnô nên ta có thể "khoanh dán" hay rút gọn trực tiếp trên bảng kích đó (còn nếu bảng kích chưa ở dạng bìa Kácnô thì ta dùng phương pháp ánh xạ vào bìa Kácnô cho từng cực kích một), kết quả ta sẽ thu được hệ phương trình hàm kích như sau :

$$J_1 = \bar{x}\,\bar{y}_2\,\bar{y}_3 + x\,y_2\,y_3 = y_2(\bar{x}\,\bar{y}_3 + x\,y_3) = y_2\,J_2$$

$$K_1 = 1 \text{ sẽ nối lên } +E$$

$$J_2 = \bar{x}\,\bar{y}_3 + x\,y_3$$

$$K_2 = \bar{x}\,y_1\,\bar{y}_3 + x\,y_1\,y_3 = y_1(\bar{x}\,\bar{y}_3 + x\,y_3) = y_1\,J_2$$

$$J_3 = \bar{x}\,\bar{y}_1 + \bar{y}_2$$

$$K_3 = x\,\bar{y}_1 + \bar{y}_2$$

$$Z = \bar{x}\,y_1\,y_3 + x\,y_1\,\bar{y}_3 = y_1(\bar{x}\,y_3 + x\,\bar{y}_3) = y_1\,\bar{J}_2$$

Sau khi có được hệ phương trình kích ta có thể biến đổi thêm để rút gọn nó nếu có thể làm được, điều này chỉ là làm "thử thêm" mà thôi, và không phải với ôtômat bất kỳ nào cũng có thể làm được, từ đó chúng ta có được hệ phương trình kích đơn giản hơn và gọn hơn như sau :

$$J_1 = y_2\,J_2$$

$$K_1 = 1$$

$$J_2 = \bar{x}\,\bar{y}_3 + x\,y_3$$

$$K_2 = y_1\,J_2$$

$$J_3 = \bar{x}\,\bar{y}_1 + \bar{y}_2$$

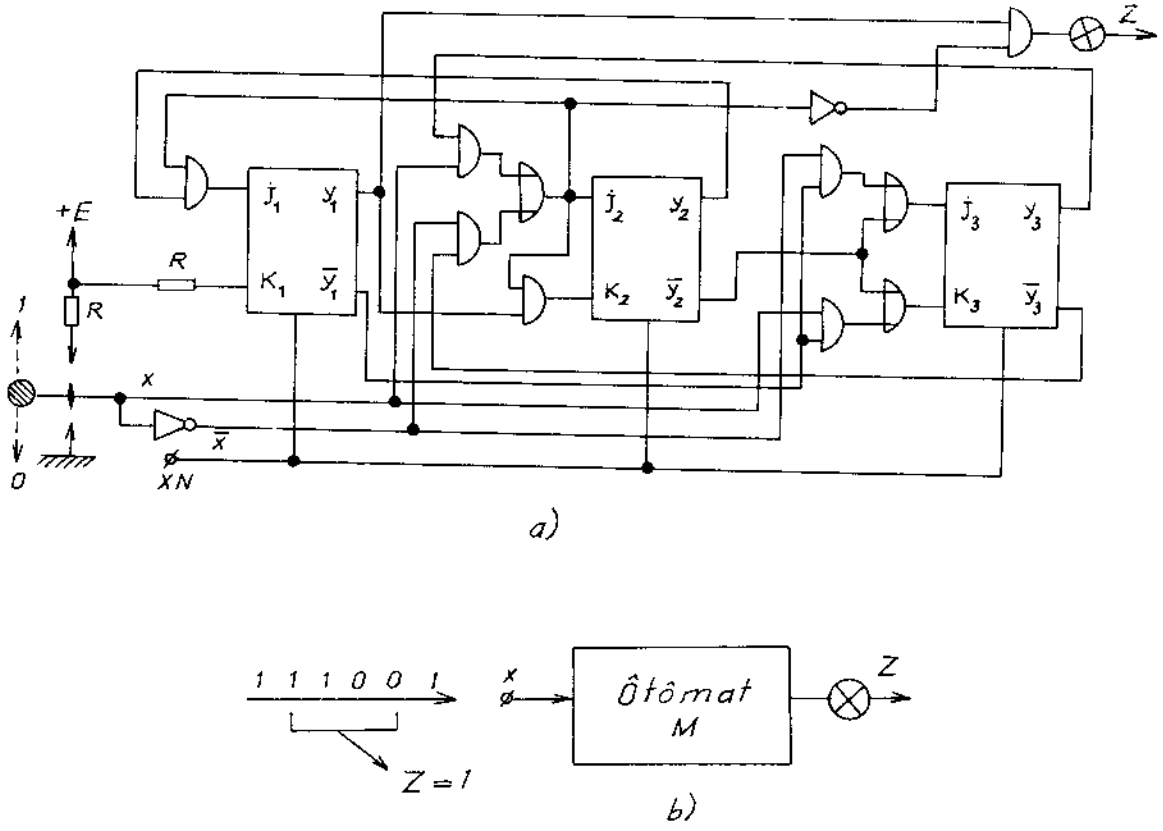
$$K_3 = x\,\bar{y}_1 + \bar{y}_2$$

$$Z = y_1\,\bar{y}_2$$

Dựa vào hệ phương trình kích gọn nhất thu được, chúng ta vẽ ra sơ đồ nguyên lý của ôtômat có nhớ cần phải tổng hợp trên hình 5.40.

Đến đây chúng ta đã kết thúc các bước hay quá trình tổng hợp một ôtômat có nhớ có một đầu vào là x và một đầu ra là Z kiểu đồng bộ. Sau khi có sơ đồ nguyên lý thực hiện chức năng của ôtômat có nhớ ta có thể tìm kiếm các linh kiện và lắp ráp ra một mạch điện, và mạch điện này khi ta tác động tín hiệu vào là (0 0 1 1) hoặc (1 1 0 0) thì đèn LED báo sáng ở đầu ra Z sẽ sáng lên, còn các trường hợp khác thì nó không sáng.

Ví dụ trên đã cho phép ta hiểu rõ cách tổng hợp và thiết kế ra một ôtômat có nhớ đồng bộ với một đầu vào x và một đầu ra Z theo một mã nhị phân (tín hiệu vào) cho trước. Bằng cách làm tương tự, chúng ta có thể tổng hợp ra các ôtômat bất kỳ với tổ



Hình 5.40. Sơ đồ mạch điện nguyên lý của ôtômat.

hợp mã hóa ở đầu vào khác đi (không phải là (0011) hay (1100) nữa), cũng như có thể kéo dài thêm tăng số lượng bit thông tin của từ mã ở đầu vào ôtômat. Nhưng cho dù tổ hợp mã ở đầu vào bất kỳ có dạng tùy ý như thế nào, hay từ mã đó có dài đến bao nhiêu, thì chúng ta vẫn tìm ra được các trạng thái cần thiết của ôtômat đó như đã làm trong ví dụ, bằng cách tăng dần lượng thông tin trong từng trạng thái của ôtômat, sau đó từ các trạng thái tìm được đó chúng ta lập ra bảng trạng thái của ôtômat bằng cách điền chúng theo từng toạ độ thích hợp trong bảng chức năng. Sau khi có bảng chức năng của một ôtômat bất kỳ từ đó chúng ta sẽ tổng hợp ra được ôtômat đó.

Ví dụ : Thực hiện ôtômat có nhớ với một đầu vào x và một đầu ra Z, hoạt động theo yêu cầu : Z = 1 nếu như gặp dãy số vào (bộ mã tín hiệu đưa vào) là 0 0 1 0 hoặc 1 1 0 1, Z = 0 trong các trường hợp khác. Từ mã tín hiệu vào đó ta tìm ngay được các trạng thái cần thiết của ôtômat đó là : (q_0, q_{00}, q_{001}, q_{0010} và q_1, q_{11}, q_{110}, q_{1101}). Sau đó lập bảng chức năng của ôtômat đó và thực hiện nó theo các bước đã được nêu trong quá trình tổng hợp ôtômat có nhớ ở phần trên.

Trên đây là các bước thực hiện ôtômat có nhớ chỉ có một đầu vào (x). Trong thực tế kỹ thuật đầu vào kích cho ôtômat có thể có nhiều đầu vào, để thực hiện ôtômat có nhớ có nhiều đầu vào phải thực hiện ôtômat có nhớ kiểu xung.

§5.4. THỰC HIỆN ÔTÔMAT CÓ NHỚ BẰNG MẠCH KIỂU XUNG

Tín hiệu đưa vào đầu vào của một ôtômat có nhớ thông thường là một tín hiệu điện, tín hiệu điện lại có nhiều dạng khác nhau, ở đây chúng ta xét tới việc xây dựng một ôtômat có nhớ có tín hiệu vào là các xung điện.

Định nghĩa 5.1. Ôtômat có nhớ không được chuẩn hóa thời gian (không có xung đồng bộ) làm việc theo kiểu xung nếu thỏa mãn các điều kiện sau :

* Các tín hiệu vào là các xung có độ rộng đủ lớn (đủ năng lượng) để có thể kích lật chuyển được các trigơ ở trong mạch.

* Tại một thời điểm xung kích chỉ được xuất hiện trên một đầu vào (nếu ôtômat có 2 đầu vào kích x_1x_2 thì ta cần phải cấm kích tổ hợp $x_1x_2 = 11$, giống như ở mạch trigơ RS cấm kích tổ hợp 11).

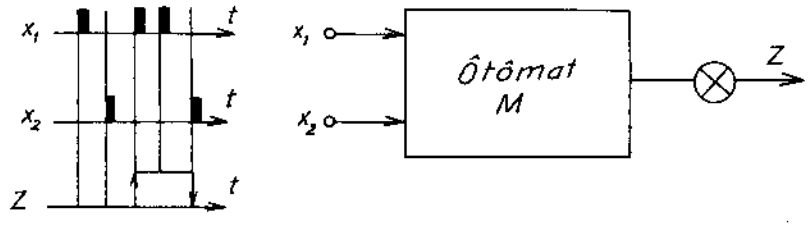
* Trạng thái bên trong của ôtômat chỉ thay đổi một lần khi một trong các đầu vào của ôtômat có xung kích vào.

Do điều kiện xung kích vào không được cùng đồng thời xuất hiện và mỗi khi trên một đầu vào có xung xuất hiện thì ôtômat chỉ thay đổi trạng thái một lần, cho nên số lượng tổ hợp của các tín hiệu vào khác nhau sẽ chính bằng số lượng trạng thái của ôtômat. Tín hiệu ra của ôtômat có thể là xung hay điện thế.

Ví dụ : Tổng hợp một ôtômat có nhớ kiểu xung có hai đầu vào là (x_1, x_2) và một đầu ra Z, hoạt động theo yêu cầu sau :

- Tín hiệu vào là các xung xuất hiện ngẫu nhiên và bất kỳ kế tiếp nhau.
- Trên các đầu vào x_1 và x_2 xung không được cùng đồng thời xuất hiện (cấm kích $x_1\ x_2 = 11$).
- Khi có xung xuất hiện trên đầu vào x_1 thì đầu ra Z = 1 nếu như đã có đúng một xung x_2 xuất hiện sau xung x_1 xuất hiện sau cùng.
- Đầu ra Z = 0 trong các trường hợp khác.
- Khi Z = 1 thì Z giữ nguyên giá trị cho đến khi xuất hiện xung x_2 thì Z trở về 0.
- Dùng trigơ JK thực hiện mô tả theo bảng chức năng Moore.

Để có thể xây dựng được bảng chức năng cho ôtômat đã đưa ra chúng ta có sơ đồ khối của nó được chỉ ra trên hình 5.41.

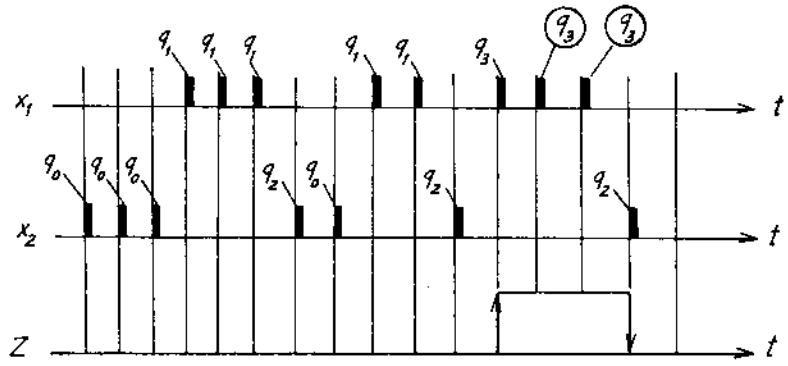


Hình 5.41. Sơ đồ nguyên lý hoạt động.

Bước 1 : Chuyển nhiệm vụ đề ra thành bảng chức năng.

Để thực hiện được bước quan trọng là tìm bảng chức năng của ôtômat có nhớ chúng ta cần xem xét kỹ và phân tích hoạt động của nó với đầy đủ mọi khả năng xuất hiện của các trạng thái có thể có của ôtômat.

Muốn tìm được các trạng thái cần thiết nhất của ôtômat cần phải xây dựng ta có thể dựa vào biểu đồ thời gian chỉ ra trên hình 5.42.



Hình 5.42. Biểu đồ thời gian kích xung.

Biểu đồ cho chúng ta biết mọi khả năng xuất hiện của các xung một cách ngẫu nhiên trên các đầu vào (x_1, x_2). Từ đó ta có các trạng thái cần thiết của ôtômat như sau :

q_0 : là trạng thái chỉ có xung xuất hiện trên đầu vào x_2 mà không cho xung vào ở đầu vào x_1.

q_1 : là trạng thái có xung xuất hiện ở đầu vào x_1 mà không có xung ở đầu vào x_2.

q_2 : là trạng thái có xung x_2 xuất hiện sau xung x_1 xuất hiện sau cùng.

q_3 : trạng thái khi có xung ngẫu nhiên xuất hiện trên đầu vào x_1 sau đúng một xung x_2 xuất hiện sau xung x_1 xuất hiện sau cùng.

Từ đó chúng ta có thể có bảng chức năng hay bảng chức năng hay bảng chuyển biến trạng thái của ôtômat cần phải tổng hợp như ở trên hình 5.45. Bảng chức năng ở dạng ôtômat Moore.

S \ X	x_1	x_2	Z
q_0	q_1	q_0	0
q_1	q_1	q_2	0
q_2	q_3	q_0	0
q_3	(q_3)	q_2	1

Hình 5.43. Bảng chức năng.

Bảng chức năng có bằng cách chúng ta điền lần lượt từng toạ độ trong bảng các trạng thái thích hợp, như đã chỉ ra trong biểu đồ thời gian cũng như đã quy ước ký hiệu của chúng. Chú ý rằng do nhiệm vụ đề ra cho ôtômat là khi Z = 1 thì giá trị của Z không thay đổi cho tới khi có xung x_2 xuất hiện thì Z mới trở về 0. Như vậy khi chưa có xung x_2 xuất hiện mà chỉ có xung x_1 ta cho vào, lúc này trạng thái của ôtômat vẫn thỏa mãn yêu cầu đề ra, nếu ta gọi trạng thái đó là (q_3).

Với bảng chức năng thu được trên hình 5.43 chúng ta vẫn chưa có thể thực hiện được ôtômat vì bảng đó vẫn ở dạng lý thuyết là có xung vào (hay kích x_1 hoặc x_2 vào) mà chưa mô tả được ở dạng bảng chức năng ở dạng lôgic, vậy muốn tổng hợp thực hiện được ôtômat này chúng ta phải chuyển bảng chức năng ở hình 5.43 thành bảng chức năng lôgic chỉ ra trên hình 5.44.

Bảng chức năng :

S \ x_1x_2	00	01	11	10	Z
q_0	q_0	q_0	–	q1	0
q_1	q_1	q_2	–	q_1	0
q_2	q_2	q_0	–	q_3	0
q_3	q_3	q_2	–	q_3	1

Hình 5.44. Bảng chức năng.

Trên bảng chức năng lôgic hình 5.44, ta chú ý :

- Do nhiệm vụ đề ra là xung không được cùng đồng thời xuất hiện nên tổ hợp x_1 x_2 = 1 1 bị cấm, hay không bao giờ có tín hiệu kích vào như vậy, cho nên trạng thái tiếp theo của ôtômat sẽ là các trạng thái không xác định.

- Khi có tổ hợp x_1 x_2 = 0 0 có nghĩa là khe giữa các xung lúc này không có xung kích vào ở đầu vào, vì vậy trạng thái của ôtômat không được thay đổi cho tới khi có xung kích tiếp theo ở đầu vào x_1 hoặc x_2, vì vậy trạng thái tiếp theo như trạng thái trước đó.

- Với tổ hợp x_1 x_2 = 0 1 tức là ở đầu vào x_1 không có xung, còn xung kích vào đầu vào x_2, vậy toàn bộ cột trạng thái của xung x_1 được điền vào bảng chức năng lôgic.

Sau khi có bảng chức năng hay bảng chuyển biến trạng thái lôgic của ôtômat có nhớ chúng ta đi tiếp tới bước tiếp theo trong quá trình tổng hợp ôtômat là.

Bước 2 : Rút gọn ôtômat có nhớ.

Với bảng chức năng của ôtômat ta tìm được thì đây đã là ôtômat min (ôtômat tối thiểu) vì : nó có phân hoạch thế (PMT) là $\Pi_1 = (\overline{q_1 , q_3})$ nhưng lại không có phân hoạch tương đương (PHTĐ). Ôtômat rút gọn được chỉ ra trên hình 5.45.

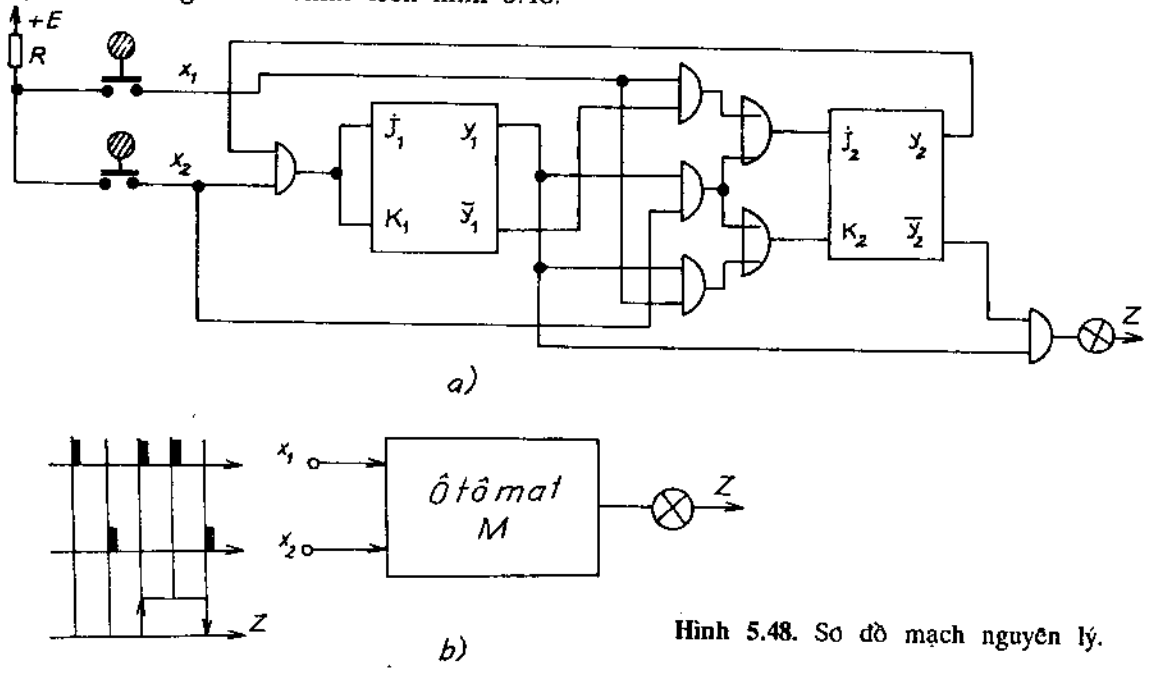
Do bảng kích đã được biểu diễn theo dạng của bìa Kácnô nên có thể khoanh dán hay rút gọn trực tiếp trên bảng kích. Kết quả ta sẽ thu được hệ phương trình như sau :

$$J_1 = x_2 y_2 \qquad J_2 = x_2 y_1 + x_1 \bar{y}_1$$

$$K_1 = x_2 y_2 \qquad K_2 = x_2 y_1 + x_1 y_1$$

$$Z = y_1 \bar{y}_2$$

Dựa vào hệ phương trình kích chúng ta vẽ ra sơ đồ nguyên lý của mạch điện thực hiện chức năng của ôtômat trên hình 5.48.



Hình 5.48. Sơ đồ mạch nguyên lý.

Với cách làm tương tự chúng ta có thể tăng số lượng đầu vào cho ôtômat.

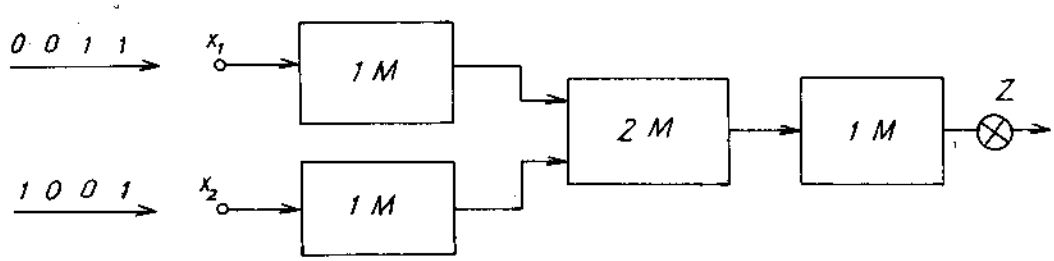
Tiếp theo từ nhiệm vụ có xung x_1 xuất hiện thì Z = 1 nếu như đã có đúng một xung x_2 sau xung x_1 sau cùng, ta có thể tăng thêm chức năng là có đúng m xung x_2 sau xung x_1 sau cùng, thì Z = 1 nếu như có xung x_1 xuất hiện. Lúc này ta phải thêm trạng thái cho ôtômat để đếm số lượng xung x_2 xuất hiện sau xung x_1 xuất hiện sau cùng.

Bài toán cũng có thể được mở rộng và phát triển dựa trên chức năng : Khi Z = 1 thì Z giữ nguyên giá trị cho đến khi có xung x_2 thì Z trở về 0. Từ chức năng này ta có thể tăng thêm điều kiện là khi Z = 1 thì Z giữ nguyên giá trị cho đến khi có n xung x_2 thì Z trở về 0. Trong trường hợp này chúng ta cũng phải cho thêm trạng thái của ôtômat để đếm số lượng xung x_2 cần thiết để đưa giá trị Z về 0.

Hoạt động của ôtômat có nhớ giống như một khóa điện tử có hai phím bấm, và phải bấm trên hai phím bấm x_1 và x_2 sao cho đúng yêu cầu đề ra cho số lần bấm thì khóa mới mở (Z = 1), và cũng phải bấm đúng yêu cầu thì cửa mới đóng (Z = 0). Như vậy thiết kế ôtômat có nhớ đơn giản với hai đầu vào (x_1, x_2) và một đầu ra Z có thể tạo

ra một ôtômat có nhớ bất kỳ với p đầu vào và các xung kích vào trên các đầu vào đó có thể xuất hiện bất kỳ.

Nếu chúng ta tổ hợp lẫn lộn chồng chất hai ôtômat có nhớ ở hai ví dụ đã nêu chúng ta sẽ có được một ôtômat có nhớ có độ phức tạp hay chức năng phong phú hơn rất nhiều. Sơ đồ khối tổng quát có thể được chỉ ra ở trên hình 5.49. Với 1M là ôtômat 1 đầu vào kích và 2M là ôtômat có 2 đầu vào kích.



Hình 5.49. Sơ đồ khối tổng quát.

§5.5. THỰC HIỆN ÔTÔMAT CÓ NHỚ DÙNG MẠCH KIỂU ĐIỆN THẾ

Một ôtômat có nhớ được gọi là mạch kiểu điện thế nếu hoạt động của nó tuân theo định nghĩa sau.

Định nghĩa 5.2. Mạch có nhớ không được chuẩn hóa thời gian được gọi là mạch kiểu điện thế nếu và chỉ nếu :

– Các tín hiệu vào ở dạng điện thế (không phải là xung).

– Các tín hiệu vào giữ nguyên không thay đổi nếu bên trong mạch trạng thái chưa ổn định.

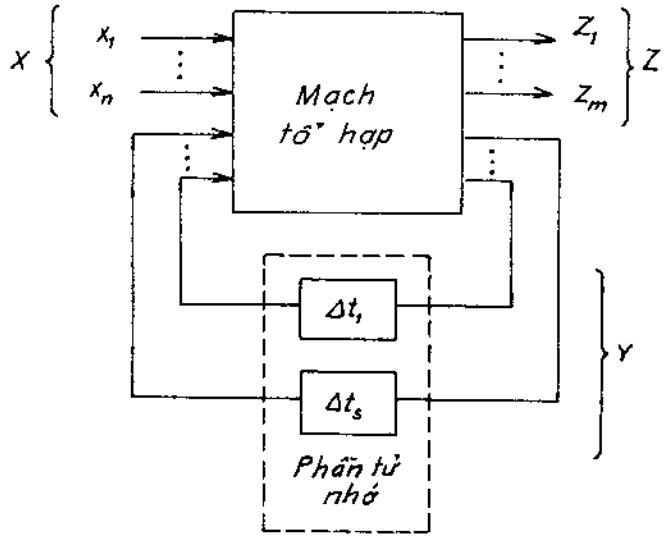
Như vậy ôtômat có nhớ hoạt động theo kiểu điện thế có tín hiệu vào ở dạng điện thế, tức là tín hiệu vào sẽ tồn tại lâu, khi đó sẽ xảy ra trường hợp ôtômat đã chuyển sang trạng thái mới rồi mà tín hiệu vào (cũ) vẫn còn tồn tại, cho nên tín hiệu vào cũ đó lại tác động tiếp vào trạng thái mới đó của ôtômat, lại làm ôtômat chuyển tiếp sang trạng thái mới nữa. Vì trạng thái tiếp theo sẽ trở thành trạng thái trước đó của trạng thái tiếp theo của nó. Do đó ôtômat có nhớ hoạt động theo kiểu điện thế còn được gọi là mạch không đồng bộ, hoạt động của loại mạch này khác với mạch đồng bộ ở các điểm sau :

a) Ôtômat có nhớ không đồng bộ vừa có các trạng thái ổn định vừa có các trạng thái không ổn định hay còn được gọi là trạng thái nhất thời. Thông thường ôtômat chuyển từ trạng thái ổn định này sang trạng thái ổn định khác phải qua một số trạng thái không ổn định (trạng thái nhất thời) trước khi đạt được đến trạng thái ổn định cuối cùng.

b) Do có các trạng thái nhất thời là các trạng thái không ổn định nên rất khó xác

định được cũng như dự đoán trước được trạng thái tiếp theo của ôtômat. Nếu xảy ra trường hợp không xác định được trạng thái ổn định cho ôtômat thì phải thiết kế lại mạch khác để loại bỏ trạng thái không ổn định. Điều này làm cho ôtômat có nhớ không đồng bộ hoạt động theo kiểu điện thế khó thiết kế hơn nhưng hoạt động nhanh hơn ôtômat có nhớ đồng bộ hay mạch kiểu có xung nhịp.

Để mô tả hoạt động của ôtômat không đồng bộ người ta có nhiều mô hình khác nhau, nhưng nói chung rất khó mô tả được hoạt động chính xác của mạch trong trường hợp tổng quát. Mô hình tổng quát của ôtômat không đồng bộ chỉ ra trên hình 5.50.



Hình 5.50. Mô hình tổng quát.

Trong đó : $X = (x_1, x_2, \ldots, x_n)$ là tập các tín hiệu vào.

$Z = (z_1, z_2, \ldots, z)$ là tập các đáp ứng ra.

$Y = (y_1, y_2, \ldots, y_s)$ tập các trạng thái trong.

5.5.1. PHÂN TÍCH MẠCH KHÔNG ĐỒNG BỘ

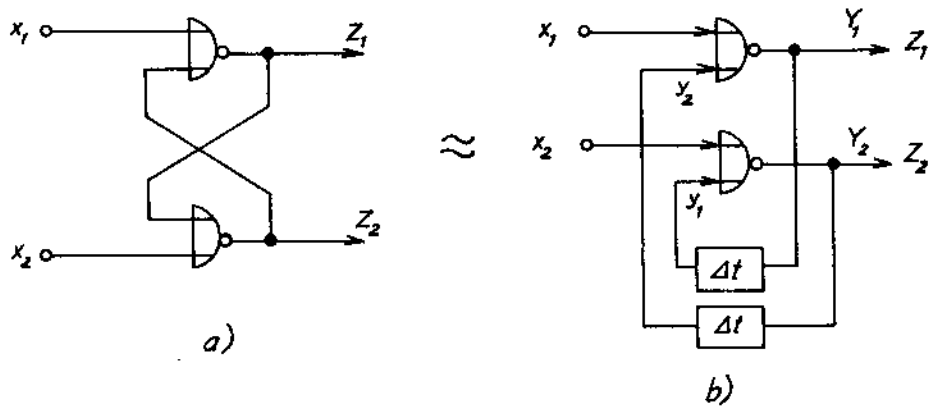
Trong mô hình tổng quát của ôtômat không đồng bộ (hệ lơi) trên hình 5.50 có đường phản hồi từ đầu ra về đầu vào, tín hiệu đưa về đầu vào chính là tập các trạng thái trong của ôtômat hay cụ thể hơn đó là tập tín hiệu lấy ra từ các trạng thái của các trigơ của hệ thống đó. Như vậy một mạch tổ hợp bất kỳ không có xung nhịp hay chuẩn hóa thời gian mà có quan hệ phản hồi thì đều có thể xem nó là một mạch có nhớ kế tiếp kiểu điện thế. Chúng ta xét mạch không đồng bộ đơn giản được chỉ ra trên hình 5.51.

Từ sơ đồ mạch điện tổ hợp có phản hồi hình 5.51,a chúng ta chuyển nó tương đương thành dạng của mô hình ôtômat không đồng bộ hình 5.51,b.

Từ sơ đồ tổng quát của mô hình hệ lơi hay ôtômat không đồng bộ chúng ta có hàm chuyển biến trạng thái tiếp theo như sau :

$$Y_1 = \bar{x}_1 . \bar{y}_2$$

$$Y_2 = \bar{x}_2 . \bar{y}_1$$



Hình 5.51. Mạch đơn giản nhất.

Trên cơ sở các phương trình chuyển biến trạng thái đó ta thành lập được bảng chuyển tiếp trạng thái biểu diễn trên hình 5.52 bằng cách tính toán ra theo toạ độ.

y_1y_2 \ x_1x_2	00	01	11	10
0 0	1 1	1 0	0 0	0 1
0 1	0 1	0 0	0 0	0 1
1 1	0 0	0 0	0 0	0 0
1 0	1 0	1 0	0 0	0 0
	$Y_1 Y_2$			$Y_1 Y_2$

Hình 5.52. Bảng chuyển tiếp trạng thái.

Cách tính hay cách điền bảng chuyển tiếp :

Toạ độ đầu tiên :

$$\left.\begin{matrix} y_1 y_2 = 0\,0 \\ x_1 x_2 = 0\,0 \end{matrix}\right\} \rightarrow \text{thay vào hàm } \left.\begin{matrix} Y_1 = \bar{x}_1 . \bar{y}_2 = \bar{0} . \bar{0} = 1 \\ Y_2 = \bar{x}_2 . \bar{y}_1 = \bar{0} . \bar{0} = 1 \end{matrix}\right\} \rightarrow Y_1 Y_2 = 11$$

$$\left.\begin{matrix} y_1 y_2 = 0\,0 \\ x_1 x_2 = 0\,1 \end{matrix}\right\} \rightarrow \left.\begin{matrix} Y_1 = \bar{x}_1 . \bar{y}_2 = \bar{0} . \bar{0} = 1 \\ Y_2 = \bar{x}_2 . \bar{y}_1 = \bar{1} . \bar{0} = 0 \end{matrix}\right\} \rightarrow Y_1 Y_2 = 10$$

$$\left.\begin{matrix} y_1 y_2 = 0\,0 \\ x_1 x_2 = 1\,1 \end{matrix}\right\} \rightarrow \left.\begin{matrix} Y_1 = \bar{x}_1 . \bar{y}_2 = \bar{1} . \bar{0} = 0 \\ Y_2 = \bar{x}_2 . \bar{y}_1 = \bar{1} . \bar{0} = 0 \end{matrix}\right\} \rightarrow Y_1 Y_2 = 00$$

Với cách làm tương tự chúng ta sẽ tính ra và điền đầy được bảng chuyển tiếp trên hình 5.52.

Sau khi có bảng chuyển tiếp trạng thái ta thấy do mạch không đồng bộ nên có hai trạng thái tiếp theo là :

- Trạng thái ổn định.

- Trạng thái không ổn định (trạng thái nhất thời).

Hai loại trạng thái xuất hiện trong ôtômat kiểu điện thế do bản chất hoạt động của nó khi tín hiệu kích vào dưới dạng điện thế, có nghĩa là tín hiệu vào tồn tại lâu, như vậy khi ôtômat chuyển sang trạng thái mới rồi thì tín hiệu ở đầu vào vẫn còn tác dụng và kích tiếp (đó là trạng thái nhất thời) để làm cho ôtômat lại chuyển đến trạng thái tiếp theo sau nữa..., vậy trạng thái ổn định được định nghĩa như sau.

Định nghĩa 5.3. Trạng thái tiếp theo (Y_1 Y_2) được gọi là ổn định nếu nó thỏa mãn :

$$Y_1\ Y_2 = y_1\ y_2$$

Trong đó : (y_1 y_2) là trạng thái trước đó.

Như vậy một trạng thái ổn định của một ôtômat có nhớ không đồng bộ hoạt động kiểu điện thế tức là : trạng thái tiếp theo giống trạng thái trước đó, trở lại trạng thái trước đó. Khi trạng thái tiếp theo trở lại trạng thái trước đó thì có nghĩa là kích mãi tín hiệu vào thì trạng thái của ôtômat vẫn thế, hay có duy trì mãi tín hiệu vào dưới dạng điện thế thì trạng thái của ôtômat vẫn không thay đổi, khi đó ta có để tín hiệu vào hay kết thúc tín hiệu vào, trạng thái của ôtômat vẫn thế không thay đổi. Có thể nói ở hệ lơi trạng thái của nó ổn định trong sự vận động.

Từ định nghĩa trạng thái "ổn định" cũng như bản chất của sự ổn định trong ôtômat không đồng bộ, ta nhìn vào bảng chuyển tiếp trạng thái ở hình 5.52 ta thấy các trạng thái ổn định được tô đậm vì trạng thái tiếp theo giống trạng thái trước đó, còn lại các trạng thái khác đó là trạng thái không ổn định (trạng thái tồn tại nhất thời).

Dựa vào bảng chuyển tiếp trạng thái dưới tác động của tín hiệu vào (x_1 x_2) ở dạng điện thế hay tồn tại lâu, chúng ta có thể xác định được dãy trạng thái hoạt động của ôtômat từ trạng thái ban đầu cho đến trạng thái ổn định cuối cùng được chỉ ra trong các ví dụ sau.

Ví dụ : Xem xét sự thay đổi của trạng thái ban đầu $y_1\ y_2 = 10$ dưới tác động của tín hiệu vào (x_1 x_2) dưới dạng điện thế trong các khả năng khác nhau là :

+ Cho $x_1\ x_2 = 0\ 0$ hoặc $0\ 1$ ở dạng điện thế tồn tại lâu, thì ta thấy trạng thái tiếp theo $Y_1\ Y_2 = 10$ vẫn giống trạng thái trước đó $y_1\ y_2 = 10$ ($10 = y_1\ y_2 = Y_1\ Y_2 = 10$). Vậy đây là trạng thái ổn định của ôtômat dù ta có kích mãi $x_1\ x_2 = 0\ 0$ thì trạng thái đó vẫn không thay đổi.

+ Nhưng nếu với tín hiệu vào $x_1\ x_2 = 1\ 1$ thì sự thay đổi trạng thái của mạch trải qua các quá trình sau :

$$y_1\ y_2 = 1\ 0 \rightarrow 0\ 0 \rightarrow 00 = Y_1\ Y_2 \quad \text{(ổn định)}$$

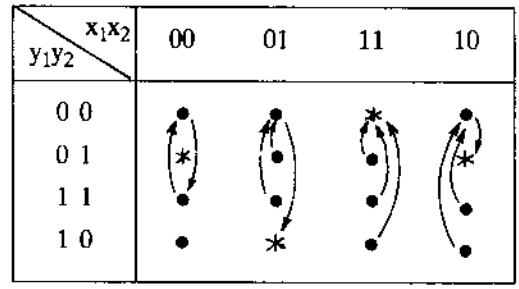
Như vậy dưới tác động của tín hiệu vào $x_1x_2 = 11$ ôtômat sẽ chuyển từ trạng thái ban đầu $y_1y_2 = 10$ đến trạng thái ổn định $Y_1y_2 = 00$. Như vậy ôtômat phải qua một trạng thái không ổn định hay trạng thái nhất thời là (00).

+ Với trường hợp kích tín hiệu vào điện thế theo quan hệ $x_1\ x_2 = 1\ 0$ thì sự thay đổi trạng thái của mạch sẽ đi qua các trạng thái sau :

$$y_1\ y_2 = 10 \rightarrow 0\ 0 \rightarrow 0\ 1 \rightarrow \mathbf{0\ 1} = Y_1\ Y_2 \qquad \text{(ổn định)}$$

Từ các quá trình chuyển biến trạng thái được minh hoạ ở trên, chúng ta có thể mô tả tất cả các quá trình chuyển biến trạng thái của ôtômat không đồng bộ trên bảng đồ thị chuyển tiếp trên hình 5.53.

Trong bảng đồ thị chuyển biến trạng thái các trạng thái ổn định được đánh dấu bằng (*), còn các trạng thái không ổn định hay trạng thái nhất thời được đánh dấu bằng (•). Từ bảng đồ thị chuyển biến trạng thái chúng ta nhận thấy có quá trình chuyển biến trạng thái của mạch tạo thành chu trình kín, tức là không tồn tại trạng thái ổn định của ôtômat. Ví dụ ứng với cột $x_1x_2 = 00$ khi ta kích vào với tín hiệu như vậy thì ôtômat sẽ không có trạng thái ổn định và trạng thái cuối cùng không thể xác định được.



Hình 5.53. Đồ thị chuyển biến trạng thái.

Đồng thời ta cũng thấy được các trạng thái ổn định của ôtômat và cả quá trình chuyển biến trạng thái của nó từ trạng thái ban đầu tới trạng thái ổn định, và theo định nghĩa thì ôtômat chỉ có ba trạng thái ổn định là có thể xác định được sau khi cho tín hiệu kích vào mà thôi.

5.5.2. CHẠY ĐUA TRẠNG THÁI (CHẠY ĐUA Y) VÀ BIỆN PHÁP LOẠI TRỪ

Ở phần I cuốn sách, chúng ta đã nói đến hazard và chạy đua (x) hay chạy đua của các tín hiệu vào, do hiện tượng quá độ hay thời gian trễ (quán tính) của tín hiệu vào gây ra. Hiện tượng hazard sai nhầm do tín hiệu vào có quán tính gây ra hay có thời gian trễ của tín hiệu vào gây ra chỉ xảy ra ở hệ tổ hợp. Khi xây dựng hệ dãy hay ôtômat có nhớ phải dùng tới trigơ hay phần tử ghi giữ thông tin, mà các trigơ lại có quá trình lật chuyển trạng thái khác nhau hay thời gian quán tính của trigơ là khác nhau nên xảy ra *quá trình "chạy đua" lật chuyển trạng thái của các trigơ* khi có nhiều trigơ cùng lật chuyển trạng thái cùng một lúc. Đó chính là quá trình "chạy đua y" hay chạy đua trạng thái của ôtômat có nhớ. Khi các trigơ có thời gian lật chuyển khác nhau hay là "chạy đua y", làm cho hoạt động của ôtômat có nhớ không còn chính xác nữa mà sẽ xảy ra sai nhầm hazard. Để có thể khắc phục hiện tượng hazard do chạy đua y ở ôtômat có nhớ, chúng ta phải tìm hiểu bản chất của chạy đua y bằng cách trở lại ôtômat không đồng bộ đã nêu ở phần trên.

Khi phân tích mạch không đồng bộ ở trên chúng ta đã coi giá trị của các biến trạng thái y thay đổi đồng thời. Nhưng như đã nói, do thời gian lật chuyển của các biến

trạng thái hay các trigơ là khác nhau, nên việc các biến trạng thái thay đổi đồng thời trong thực tế là khó có thể xảy ra, mà sẽ gây ra trạng thái phụ làm giảm độ tin cậy hoạt động của ôtômat.

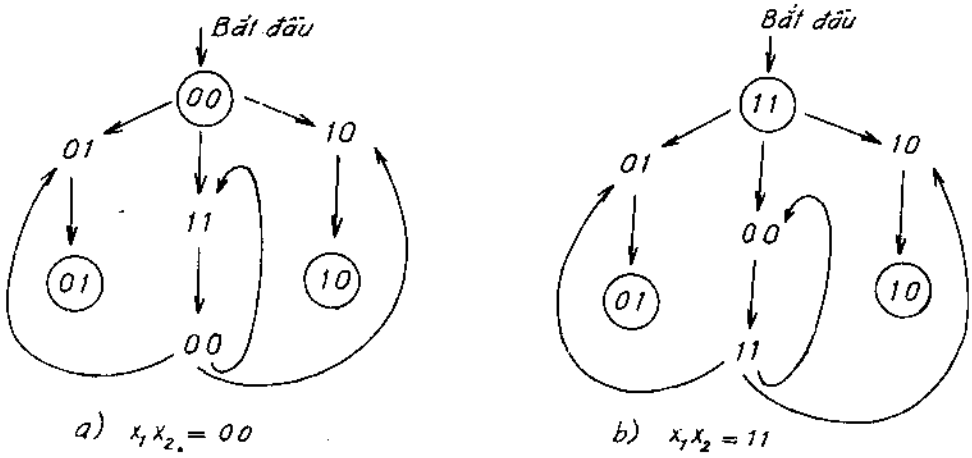
Ví dụ : Theo bảng trạng thái của ôtômat đã chỉ ra ở trên hình 5.52 chúng ta thấy có hai biến trạng thái là (y_1 và y_2), chúng có quan hệ chuyển tiếp là :

$$Y_1 = \bar{x}_1 . \bar{y}_2$$

$$Y_2 = \bar{x}_2 . \bar{y}_1$$

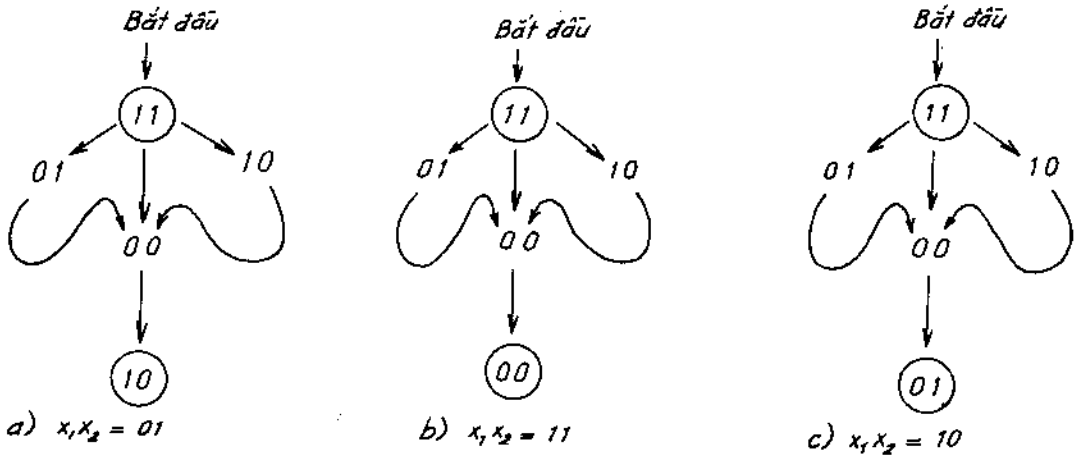
+ Dưới tác động của bộ tín hiệu vào dưới dạng điện thế $x_1\ x_2$ = 0 0 làm cho trạng thái của ôtômat chuyển từ trạng thái trước đó ($y_1\ y_2$ = 0 0) sang trạng thái tiếp theo ($Y_1\ Y_1$ = 1 1). Như vậy từng biến trạng thái y_1 và y_2 cùng chuyển từ 0 sang 1 đồng thời, gây ra hiện tượng chạy đua y. Nhưng do tốc độ lật chuyển hay thời gian trễ (Δt) của các biến y_1 và y_2 là khác nhau, từ đó gây ra các trạng thái phụ làm hoạt động của ôtômat không còn chính xác nữa (xuất hiện hazard), để hiểu rõ chúng ta xem xét kỹ hơn như sau.

Do tác động $x_1\ x_2$ = 0 0 làm cho ôtômat chuyển đổi ($y_1\ y_2$ = 0 0 → 1 1 = $Y_1\ Y_2$). Nhưng do tốc độ thay đổi hay trễ của y_1 và y_2 khác nhau nên : Giả sử y_1 chạy nhanh hơn y_2 hay quán tính quá độ của y_1 nhỏ hơn y_2 nên sẽ xảy ra kết quả ($y_1\ y_2$ = 0 0 → 1 0 đây là trạng thái phụ trước khi đi đến trạng thái ổn định). Tương tự ta cũng có thể giả sử y_1 chạy chậm hơn y_2 khi đó sẽ xảy ra trường hợp ($y_1\ y_2$ = 00 → 01). Nhưng nếu y_1 và y_2 tốc độ chạy như nhau và cứ chạy đều mãi cho kết quả là ($y_1\ y_2$ = 00 → 11) và cứ chạy đều mãi cho tới khi xuất hiện sự chạy đua giữa y_1 và y_2. Các hiện tượng đó được minh họa trên hình 5.54.



Hình 5.54. Hiện tượng chạy đua y.

+ Dưới tác động của tín hiệu vào khác như x_1x_2 = 01 hay x_1x_2 = 10 mà ôtômat có xuất hiện chạy đua y tức là (y_1y_2 = 00 → 11 = $Y_1\ Y_2$) thì các hiện tượng chạy đua đó của biến y sẽ được minh họa trên hình 5.55.



Hình 5.55. Chạy đua y khi tín hiệu vào khác.

Ở đây ta chú ý do tín hiệu vào là ở dạng điện thế tồn tại lâu cho nên dù có chạy đua y thế nào thì cuối cùng ôtômat cũng sẽ chuyển tới trạng thái ổn định là trạng thái tiếp theo giống trạng thái trước đó ($Y_1\ Y_2 = y_1\ y_2$) và không thể thoát ra được nữa (ổn định tại đó) dù cho tín hiệu tác động vào $x_1\ x_2$ có còn tồn tại hay đã kết thúc. Các trạng thái ổn định trên các hình 5.55 đều được vẽ cùng với khoanh tròn (ví dụ : (01) hay (10) v.v...).

Qua đây chúng ta cũng thấy điều kiện để xuất hiện chạy đua trạng thái hay chạy đua y là có hai hay nhiều biến trạng thái phải thay đổi đồng thời dưới tác động của tín hiệu vào.

Nhìn vào hình 5.55 mô tả quá trình chạy đua y nếu được xảy ra dưới tác động của tín hiệu vào ta thấy quá trình chuyển biến tốt nhất bất chấp có chạy đua giữa y_1 và y_2 là ở hình 5.55,b, vì dù cho có chạy đua hay không thì kết quả ổn định cuối cùng vẫn là (y_1y_2 = (11) → (00) = Y_1Y_2) giống như mong muốn của ta khi cho tín hiệu vào là x_1x_2 = 11. Trong trường hợp này ôtômat chuyển biến trạng thái đúng và hoàn toàn chính xác.

Trường hợp 5.55,a và c chúng ta thấy dưới tác động của tín hiệu vào x_1x_2 = 01 hay x_1x_2 = 10 đều làm cho ôtômat có xuất hiện chạy đua y. Nhưng sau khi chạy đua thì ôtômat chuyển trạng thái về trạng thái ổn định (10) hoặc (01). Như vậy kết quả thu được sau khi kích tín hiệu vào là :

x_1x_2 = 01 làm cho (y_1y_2 = (11) → (10) = Y_1Y_2)

x_1x_2 = 10 làm cho (y_1y_2 = (11) → (01) = Y_1Y_2)

Kết quả là dưới tác động của tín hiệu vào (x_1x_2) làm cho trạng thái của ôtômat chuyển tới trạng thái "ổn định sai" nhưng sau khi có chạy đua y ôtômat chỉ chuyển về duy nhất có một trạng thái ổn định sai là (10) hoặc (01) cho nên ta có thể nói đó là các trạng thái ổn định sai nhưng quản lý được. Trong trường hợp này có thể coi như ôtômat chuyển trạng thái đúng. Hiện tượng chạy đua như vậy còn được gọi là chạy đua tới hạn tức là chạy đua tới một kết quả hạn định trước được.

Còn trường hợp cuối cùng xảy ra khi có chạy đua y dưới tác động của tín hiệu vào $x_1x_2 = 00$ hay $x_1x_2 = 11$ chỉ ra trên hình 5.54. Nhìn vào hình vẽ ta thấy nếu giả sử y_1 nhanh hơn y_2 thì $y_1y_2 = 10$ (trước đó $y_1y_2 = 00$) sau đó chuyển đến trạng thái ổn định là $Y_1Y_2 = (10)$, còn nếu như y_1 chậm hơn y_2 dưới tác động của tín hiệu vào thì $y_1y_2 = 01$ (trạng thái trung gian hay trạng thái phụ) sau đó chuyển về trạng thái ổn định là $Y_1Y_2 = (01)$. Trong thực tế ta không thể biết được y_1 chạy nhanh hơn y_2 hay y_2 quán tính nhỏ hơn y_1, như vậy trạng thái ổn định sai là (10) hay (01) ta không biết chính xác được hay không thể quản lý được các trạng thái đó. Trong trường hợp kích tín hiệu vào mà không thể biết chính xác hay chắc chắn trạng thái của ôtômat sẽ chuyển về đâu, là một điều không cho phép khi thiết kế một hệ thông số. Ở đây ta có hai biến (y_1y_2) làm xuất hiện hai trạng thái ổn định sai không quản lý được, nếu ta có số lượng biến y nhiều hơn thì số lượng trạng thái ổn định sai không quản lý hay biết trước được sẽ nhiều hơn, làm cho hoạt động của ôtômat sẽ cùng sai nhiều và nguy hiểm hơn. Hiện tượng này chúng ta gọi là chạy đua trạng thái không tới hạn hay không đến một giới hạn chắc chắn nào cả.

Các biện pháp loại trừ chạy đua trạng thái không tới hạn.

a) Phương pháp giới hạn hay ngăn chặn các biến vào

Để không cho xuất hiện chạy đua trạng thái hay giảm chạy đua y, ta chủ động hạn chế hay cấm không cho các tín hiệu vào ôtômat mà gây ra hiện tượng chạy đua không tới hạn. (Ví dụ : cấm kích tổ hợp $x_1x_2 = 00$ đối với ôtômat nêu ở phần trên). Hiện tượng này xẩy ra giống như hoạt động của trigơ RS (RSFF) khi ta cấm kích tổ hợp 11 (SR = 11).

b) Phương pháp mã hóa

Dùng các trigơ khi chúng có thời gian lật chuyển không giống nhau (không đồng thời) mà khác nhau, để tránh hiện tượng chạy đua y có thể dùng cách đặt tên hay mã hóa trạng thái cho ôtômat thích hợp.

Có nhiều phương pháp mã hóa để không xuất hiện quá trình chạy đua y hay làm cho các trigơ trong ôtômat lật chuyển không đồng thời. Người ta có thể căn cứ vào sự lật chuyển trạng thái cụ thể của ôtômat để mã hóa, với mục đích làm sao cho : chuyển từ trạng thái này sang trạng thái khác của ôtômat chỉ có một biến trạng thái y là thay đổi (chỉ có một trigơ là chuyển trạng thái), hay làm cho số lượng biến trạng thái phải lật chuyển đồng thời là ít nhất, khi đó sẽ tránh được chạy đua y hay giảm thiểu được chạy đua trạng thái của ôtômat. Tất nhiên việc làm này không phải lúc nào ta cũng thực hiện được, nhưng đó là điều ta phải biết khi thiết kế ôtômat.

Thông thường người ta hay dùng mã kề (mã Grây hay mã bìa Kácnô), nó được định nghĩa : hai trạng thái được gọi là kề nhau nếu chúng chỉ khác nhau ở giá trị của một biến trạng thái trong "tên gọi" của nó.

Ví dụ : Trạng thái A = 011

B = 001

Trạng thái A là kế với trạng thái B vì mã tên của chúng chỉ khác nhau ở giá trị của một biến trạng thái.

Dùng phương pháp mã kế có thể khắc phục được chạy đua trạng thái, điều đó được minh họa ở ví dụ sau :

Ví dụ : Xét bảng mã hóa của bộ đếm 8 là bộ đếm vòng được chỉ ra trên hình 5.56.

S \ x	1
0	1
1	2
2	3
3	4
4	5
5	6
6	7
7	0

a)

$y_1y_2y_3$ \ x	1
0 0 0	0 0 1
0 0 1	0 1 0
0 1 0	0 1 1
0 1 1	1 0 0
1 0 0	1 0 1
1 0 1	1 1 0
1 1 0	1 1 1
1 1 1	0 0 0

b) $Y_1Y_2Y_3$

$y_1y_2y_3$ \ x	1
0 0 0	0 0 1
0 0 1	0 1 1
0 1 1	0 1 0
0 1 0	1 1 0
1 1 0	1 1 1
1 1 1	1 0 1
1 0 1	1 0 0
1 0 0	0 0 0

c) $Y_1Y_2Y_3$

Hình 5.56. Bộ đếm 8 xét chạy đua y.

Trước hết xem xét mã hóa bộ đếm 8 theo quy luật của nhị phân tăng dần trên hình 5.56b, chúng ta thấy dưới tác động của tín hiệu vào (x = 1) làm cho giữa trạng thái trước đó với trạng thái tiếp theo có biến khả năng chạy đua như sau.

Ví dụ : $y_1\,y_2y_3 = 0\,0\,1 \rightarrow Y_1\,Y_2\,Y_3 = 0\,1\,0$

hay $y_1\,y_2y_3 = 0\,1\,1 \rightarrow Y_1\,Y_2\,Y_3 = 1\,0\,0$

Như vậy bộ đếm vòng cơ đếm 8 nếu ta mã hóa theo nhị phân tăng dần thông thường sẽ chắc chắn xảy ra chạy đua y do có nhiều trigơ (bít) cùng thay đổi giá trị hay chạy đua trạng thái của ôtômat, do thời gian lật chuyển của các trigơ là khác nhau.

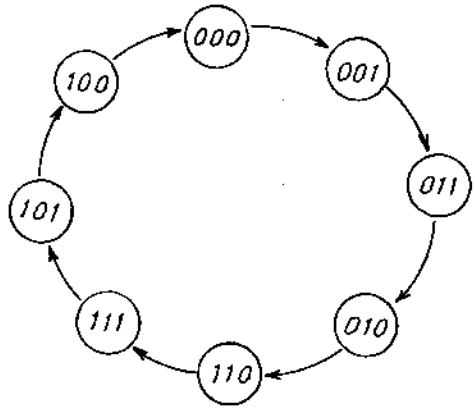
Còn vẫn là bộ đếm 8 chúng ta mã hóa theo quy luật của mã kế hay mã Grây như chỉ ra trên hình 5.56,c chúng ta thấy dưới tác động của tín hiệu vào (x = 1) làm cho ôtômat (bộ đếm) lật chuyển trạng thái từ trạng thái trước đó thành trạng thái tiếp theo là không có chạy đua y.

Ví dụ : $y_1\,y_2y_3 = 0\,0\,1 \rightarrow Y_1\,Y_2\,Y_3 = 0\,1\,1$

hay $y_1\,y_2y_3 = 0\,1\,1 \rightarrow Y_1\,Y_2\,Y_3 = 0\,1\,0$

Với cách mã hóa cho bộ điếm 8 dùng mã kế thì việc chuyển biến trạng thái tuần tự của ôtômat (bộ đếm) sẽ không thể có chạy đua y, vì việc chuyển đổi giữa trạng thái trước đó ($y_1y_2y_3$) thành trạng thái tiếp theo ($Y_1Y_2Y_3$) luôn chỉ có một biến y hay chỉ có một trigơ lật chuyển trạng thái. Khi chỉ có một biến trạng thái được lật chuyển thì không thể có hiện tượng chạy đua trạng thái.

Đồ thị chuyển biến trạng thái của bộ đếm 8 mã hóa theo mã Grây chống chạy đua trạng thái được mô tả trên hình 5.57.



y_1 y_2 y_3 \ x	1
0 0 0	
0 0 1	
0 1 1	
0 1 0	
1 1 0	
1 1 1	
1 0 1	
1 0 0	

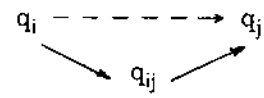
Hình 5.57. Mã Grây chống chạy đua y.

Nhìn vào hình vẽ chúng ta thấy trong trường hợp này mạch không có trạng thái ổn định nào cả.

c) Mã hóa trạng thái ôtômat theo phương pháp mã đệm để tránh chạy đua y

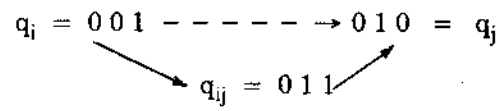
Trong thực tế trạng thái của ôtômat luôn được rút gọn sao cho có số lượng trạng thái là ít nhất, vì thế khi ta thực hiện mã hóa cho ôtômat min luôn xảy ra việc tồn tại các bộ mã thừa (do $2^k \geq p$). Chúng ta có thể dùng các bộ mã thừa đó làm mã đệm cho ôtômat để tránh quá trình chạy đua trạng thái cho ôtômat. Khái niệm mã đệm được hiểu như sau :

Giả sử có ôtômat phải chuyển đổi trạng thái từ trạng thái q_i tới trạng thái tiếp theo là q_j dưới tác động của tín hiệu vào. Trong quá trình chuyển đổi trực tiếp q_i sang q_j ($q_i \rightarrow q_j$) thì xảy ra hiện tượng chạy đua trạng thái. Muốn tránh được hiện tượng này chúng ta phải tìm ra một tổ hợp (cách tìm là không có lý thuyết) các trạng thái đệm (q_i q_j) là (q_{ij}) sao cho quá trình chuyển đổi ($q_i \rightarrow q_{ij} \rightarrow q_j$) là không xuất hiện chạy đua trạng thái. Ta có thể minh họa quá trình tạo mã đệm như sau :



Có ví dụ minh họa cụ thể quá trình tìm mã đệm như sau :

Giả sử $q_i = 001$ và $q_j = 010$ ta có mã đệm của chúng là $q_{ij} = 011$.



Thông thường mã đệm q_{ij} được tìm và lấy ra từ bộ mã thừa hay không dùng đến của ôtômat đã được rút gọn. Nhưng trong thực tế kỹ thuật không phải ôtômat nào cũng có thể tìm được mã đệm nằm trong khu vực mã thừa không dùng đến của nó. Trong

trường hợp này để chống chạy đua trạng thái trong ôtômat không đồng bộ người ta phải dùng phương pháp thêm biến phụ được nêu tiếp sau.

d) Phương pháp thêm biến phụ để tránh chạy đua trạng thái trong ôtômat có nhớ kiểu điện thế

Thông thường khi mã hóa trạng thái cho một ôtômat min có số lượng trạng thái ít nhất chúng ta thường dùng số lượng trạng thái cần dùng để mã hóa là ít nhất, làm như vậy sẽ được mạch điện thực hiện chức năng của ôtômat là đơn giản nhất nhưng khó tránh khỏi việc chạy đua trạng thái xuất hiện trong ôtômat đó.

Nhưng trong phương pháp thêm biến phụ tức là chúng ta cho thêm biến cần dùng để mã hóa vào mạch, tức là chấp nhận mạch điện sẽ phức tạp hơn kinh phí cao hơn nhưng chống được chạy đua trạng thái và hệ thống kỹ thuật hoạt động tốt hơn. Để hiểu rõ bản chất của việc cho thêm biến phụ vào để mã hóa trạng thái cho ôtômat ta xét một ví dụ sau.

Ví dụ : Cho ôtômat hoạt động theo bảng chức năng và được mã hóa theo nhị phân tăng dần chỉ ra trên hình 5.58.

Bảng chức năng

S \ x_1x_2	00	01	10	11
00	q_0	q_0	q_0	q_1	–
01	q_1	q_1	q_2	q_1	–
10	q_2	q_2	q_0	q_3	–
11	q_3	q_3	q_2	q_3	–

a)

Bảng mã hóa

y_1y_2 \ x_1x_2	00	01	10	11
0 0	00	00	01	–
0 1	01	10	01	–
1 0	10	00	11	–
1 1	11	10	11	–

b)

Hình 5.58. Cho bảng chức năng ôtômat.

Nhìn vào bảng mã hóa theo cách mã hóa nhị phân thông thường thì chúng ta thấy có một vị trí xuất hiện hiện tượng chạy đua y, tức là có hai biến trạng thái phải biến đổi đồng thời dưới tác động của tín hiệu vào (x_1x_2 = 01), kết quả cụ thể là (y_1y_2 = 01 → 10 = Y_1Y_2) xuất hiện chạy đua y.

Dùng phương pháp thêm biến phụ để khắc phục hiện tượng chạy đua trạng thái trong ôtômat đó, phương pháp thực hiện cụ thể như sau :

Thông thường, ôtômat có 4 trạng thái, nên chỉ cần có hai biến trạng thái cần dùng để mã hóa (y_1y_2) là đủ, để mã hóa trạng thái cho ôtômat mà không bị trùng tên nữa (nhưng xuất hiện chạy đua y), vấn đề ở đây chúng ta dùng thêm biến phụ, tức là tăng thêm số lượng trạng thái cần dùng để mã hóa, chấp nhận mạch phức tạp hơn để chống chạy đua y. Giả sử chúng ta tăng thêm hai biến cần dùng để mã hóa, (y_1 y_2 y_3 y_4). Bảng đặt tên của trạng thái chỉ ra trên hình 5.59.

	Trạng thái	y_1 y_2 y_3 y_4
Bộ mã của các trạng thái cơ bản	q_0	0 0 0 1
	q_1	0 0 1 0
	q_2	0 1 0 0
	q_3	1 0 0 0
Tên của các trạng thái đệm	q_0q_1	0 0 1 1
	q_0q_2	0 1 0 1
	q_1q_2	0 1 1 0
	q_2q_3	1 1 0 0

Hình 5.59. Cho thêm biến phụ chống đua y.

Khi ta cho thêm biến phụ thì khu vực mã thừa sẽ nhiều lên và trong đó chúng ta có thể tìm ra được các mã đệm thích hợp. Trong bảng "đặt tên" hay mã hóa trạng thái cụ thể cho ôtômat ta có khu vực mã đệm, khu vực mã đệm này ta có được dựa vào hoạt động cụ thể của ôtômat có nhớ chỉ ra ở trên hình 3.58,a.

Ví dụ : Kích tín hiệu vào x_1x_2 = 10 thì xuất hiện quá trình chuyển biến trạng thái từ trạng thái trước đó (q_0) sang trạng thái tiếp theo là (q_1) vậy ($q_0 \rightarrow q_1$), từ đó ta có trạng thái đệm (q_0q_1 hay q_{01}).

Tương tự như vậy chúng ta tìm ra các bộ mã đệm khác của ôtômat cần phải mã hóa.

Mà ta đã có tên mã với trạng thái q_0 = 0001 và tên mã của q_1 = 0010 vậy kết hợp hai tên mã đó lại ta sẽ thu được tên của mã đệm là q_0q_1 = 0011. Với cách làm tương tự chúng ta sẽ có tên cần thiết cho từng bộ mã đệm của ôtômat.

Sau khi có đầy đủ tên mã của các trạng thái cần thiết ta có bảng mã hóa cho ôtômat có nhớ ở trên hình 5.58,a, theo phương pháp tăng thêm biến phụ và dùng mã đệm để chống chạy đua trạng thái như chỉ ra trên hình 5.60.

Bảng mã hóa

	$y_1y_2y_3y_4$ \ x_1x_2	00	01	10	11
q_0	0 0 0 1	0 0 0 1	0 0 0 1	0 0 1 1	
q_1	0 0 1 0	0 0 1 0	0 1 1 0	0 0 1 0	
q_2	0 1 0 0	0 1 0 0	0 1 0 1	1 1 0 0	
q_3	1 0 0 0	1 0 0 0	1 1 0 0	1 0 0 0	
q_{01}	0 0 1 1			0 0 1 0	
q_{02}	0 1 0 1		0 0 0 1		
q_{12}	0 1 1 0		0 1 0 0		
q_{23}	1 1 0 0		0 1 0 0	1 0 0 0	

Hình 5.60. Bảng mã hóa ôtômat thêm biến phụ chống đua y.

Sau khi mã hóa bằng phương pháp thêm biến phụ và dùng mã đệm, nhìn vào bảng mã hóa chúng ta thấy không xuất hiện chạy đua trạng thái, tức là không gặp trường hợp khi thay đổi trạng thái của nó có hai hay nhiều biến trạng thái thay đổi đồng thời.

Bước tiếp theo ta sẽ lập bảng kích, ở đây cần chú ý là phải lập bảng kích cho toàn bộ bảng mã hóa kể cả khu vực mã đệm, khi đó hệ phương trình kích sẽ phức tạp hơn và mạch điện lắp ráp sẽ lớn hơn khi ta tăng thêm biến phụ khi mã hóa, thay vào đó mạch sẽ chống được quá trình chạy đua y và hoạt động của ôtômat tin cậy hơn. Nhưng nếu tăng số lượng biến trạng thái lên quá lớn, làm cho ôtômat sẽ hoạt động kém đi, như vậy phương pháp tăng biến trạng thái để chống đua y cho ôtômat chỉ nên áp dụng cho từng ôtômat cụ thể và chắc chắn có lợi thì ta mới dùng.

5.5.3. HAZARD TRONG ÔTÔMAT CÓ NHỚ

Trong phần hệ tổ hợp hay ôtômat không có nhớ, chúng ta đã đề cập khái niệm hazard hay hoạt động "đỏng đảnh" trong mạch điện khi xuất hiện chạy đua của các tín hiệu vào (X). Các loại hazard trong mạch tổ hợp chỉ xuất hiện nhất thời khi ta cho tín hiệu tác động vào mà xảy ra hiện tượng chạy đua (nhiều tín hiệu vào cùng biến đổi một lần). Trong hệ dãy hay ôtômat có nhớ cũng xuất hiện hiện tượng hazard nhưng hiện tượng hazard trong mạch có nhớ phức tạp hơn và đa dạng hơn.

Trong ôtômat có nhớ ngoài các loại hazard tương tự như trong mạch tổ hợp, thì trong mạch có nhớ còn có nhiều loại hazard khác nhau như hazard trạng thái ổn định, hazard chu trình v.v... Do tính chất phức tạp của hiện tượng hazard xuất hiện trong hệ dãy nên phần này chúng ta chỉ nêu lên những hiện tượng cơ bản nhất, từ đó sẽ có biện pháp khắc phục khi ta phải tổng hợp hai thiết kế một ôtômat có nhớ. Sau đây là các định nghĩa tên gọi cho một vài loại hazard như sau.

Định nghĩa 5.4. Sau khi có tín hiệu tác động vào nếu có sự chuyển đổi trạng thái từ Q sang P ($Q \rightarrow P$), và sau khi đã xác lập trạng thái cuối cùng mà ôtômat chuyển đến một trạng thái ổn định sai, thì hiện tượng đó gọi là hazard trạng thái ổn định.

Định nghĩa 5.5. Nếu sau khi có sự chuyển đổi trạng thái từ Q sang P ($Q \rightarrow P$), và sau khi đã xác lập trạng thái cuối cùng mà ôtômat chuyển đến một trạng thái không xác định (chu trình), thì hiện tượng đó gọi là hazard chu trình.

Định nghĩa 5.6. "Hazard cốt yếu" là sự chạy đau tới hạn giữa sự thay đổi của một (hoặc nhiều) biến vào có bù (x thành $\bar{x}$ hoặc ngược lại) và biến trạng thái y của ôtômat trên đường phản hồi.

Để minh họa cho khái niệm "hazard cốt yếu" chúng ta xét một vài ví dụ sau :

Ví dụ : Cho một ôtômat không đồng bộ hoạt động theo bảng chức năng chỉ ra trên hình 5.61.

y_1y_2 \ x	0	1
0 0	0 1	0 0
0 1	0 1	1 1
1 1	1 0	1 1
1 0	1 0	0 0

Hình 5.61. Bảng chức năng mã hóa.

Từ bảng mã hóa của ôtômat chúng ta có được hệ phương trình chuyển biến trạng thái là :

$$Y_1 = \bar{x} \cdot y_1 + x \cdot y_2$$

$$Y_2 = \bar{x} \cdot \bar{y}_1 + x \cdot y_2$$

Để tránh hazard tĩnh chúng ta thêm phần dư là các hàm số phụ, với yêu cầu các phần dư thêm vào đó không được làm thay đổi hay ảnh hưởng tới chức năng của mạch. Ví dụ chúng ta thêm vào như sau :

$$Y_1 = \bar{x} \cdot y_1 + x \cdot y_2 + y_1 \cdot y_2$$

$$Y_2 = \bar{x} \cdot \bar{y}_1 + x \cdot y_2 + \bar{y}_1 \cdot y_2$$

Từ hệ phương trình đó chúng ta sẽ tính được trạng thái tiếp theo (Y_1 Y_2) theo tọa độ của bảng mã hóa của ôtômat. Giả sử xét quá trình chuyển biến trạng thái của ôtômat ứng với tọa độ (x, y_1 y_2) = (1, 11). Như vậy trạng thái trước đó (y_1y_2 = 11) dưới tác động của tín hiệu vào (x = 1) thì trạng thái tiếp theo của ôtômat theo bảng chức năng sẽ là (Y_1 Y_2 = 11).

Như vậy khi tác động vào x = 1 thì ta có (y_1y_2 = 11 → 11 = Y_1 Y_2) ta thấy trạng thái tiếp theo giống trạng thái trước đó vậy đây là một trạng thái ổn định.

Từ tọa độ (x, y_1 y_2) = (1, 11) ta có x = 1, y_1 = 1 và y_2 = 1. Ta thay các giá trị đó vào hệ phương trình có thêm phần phụ để kiểm tra xem phần phụ thêm vào có làm ảnh hưởng tới chức năng của ôtômat hay không. Sau khi thay các giá trị vào và tính ra ta được kết quả Y_1 = 1 và Y_2 = 1 như vậy (Y_1 Y_2 = 11) đúng như yêu cầu ở bảng mã hóa. Nói một cách khác là phần phụ thêm vào không ảnh hưởng tới chức năng của ôtômat.

Sau đây ta xem xét hiện tượng "hazard cốt yếu" khi có sự thay đổi giữa x và $\bar{x}$, giữa chúng có sự thay đổi giá trị ngẫu nhiên nên có các trường hợp sau :

+ *Trường hợp thứ nhất* : x và $\bar{x}$ có giá trị thay đổi đồng thời, hiện tượng xảy ra được mô tả trên bảng ở hình 5.62.

	t_0	t_1	t_2
x	1	0	0
$\bar{x}$	0	1	1
y_1	1	1	1
y_2	1	1	0
Y_1	1	1	1
Y_2	1	0	0
Y_1Y_2	11 →	10 →	10

$t_0 < t_1 < t_2$

Hình 5.62. Hiện tượng "hazard cốt yếu".

Tại thời điểm t_0 ta có các giá trị đầu vào là (x, y_1y_2) =(1, 11) ứng tọa độ ở bảng mã hóa :

$$x = 1 \; ; \; \bar{x}_1 = 0 \; ; \; y_1 = 1 \text{ và } y_2 = 1$$

Thay các giá trị đó vào hệ phương trình chuyển biến trạng thái ta tính ra được $Y_1 = 1$ và $Y_2 = 1$ tức là ($Y_1Y_2 = 11$). Các giá trị đó ta viết được cột đầu tiên của bảng là 5.62.

Tại thời điểm t_1 lúc này giá trị của x và $\bar{x}$ thay đổi đồng thời nên :

$x = 0$; $\bar{x} = 1$; lúc này $Y_1\ Y_2 = 11$ tại thời điểm t_0 trở thành ($y_1\ y_2 = 11$) tại thời điểm t_1 nên ta có $y_1 = 1$ và $y_2 = 1$. Các giá trị đó ta lại thay vào hệ phương trình chuyển biến trạng thái tính ra ta thu được $Y_1 = 1$ và $Y_2 = 0$ tức là ($Y_1Y_2 = 10$). Những giá trị và kết quả thu được ta viết vào cột t_1 trong bảng trên hình 5.62.

Tại thời điểm t_2 lúc này là quá trình chuyển biến nốt từ t_1 sang t_2 và giá trị của x và $\bar{x}$ vẫn giữ nguyên :

$$x = 0 \; ; \; \bar{x} = 1 \; ; \; y_1 = 1 \text{ và } y_2 = 0$$

Thay các giá trị đó vào hệ phương trình và ta tính ra được $Y_1 = 1$ và $Y_2 = 0$ tức là ($Y_1Y_2 = 10$). Đây là trạng thái ổn định vì có tăng thêm thời gian nữa thì kết quả tính ra vẫn như thế. Những kết quả tính được chúng ta ghi tương ứng vào cột t_2 ở trong bảng. Từ kết quả ở bảng ta có kết luận : Khi giá trị x và $\bar{x}$ thay đổi đồng thời thì trạng thái của ôtômat sẽ chuyển đến trạng thái ổn định là ($Y_1Y_2 = 10$).

+ *Trường hợp thứ hai* : x và $\bar{x}$ không thay đổi giá trị đồng thời (tức là khi x đã thay đổi giá trị $\bar{x}$ vẫn còn giữ giá trị trước đó hoặc ngược lại). Hiện tượng đó được mô tả ở bảng trên hình 5.63.

$t_0 < t_1 < t_2 < t_3 < t_4$

	t_0	t_1	t_2	t_3	t_4
x	1	0	0	0	0
$\bar{x}$	0	0	0	1	1
y_1	1	1	1	0	0
y_2	1	1	0	0	1
Y_1	1	1	0	0	0
Y_2	1	0	0	1	1
Y_1Y_2	11 →	10 →	00 →	01 →	01

Hình 5.63. Hiện tượng "hazard cốt yếu".

Tại thời điểm ban đầu t_0 chúng ta có giá trị các biến :

$$x = 1 \; ; \; \bar{x} = 0 \; ; \; y_1 = 1 \text{ và } y_2 = 1$$

Từ đó ta tính ra được $Y_1 = 1$ và $Y_2 = 1$ nên ($Y_1\ Y_2 = 11$) các giá trị thu được chúng ta ghi ở cột t_0 trong bảng trên hình 5.63.

Tại thời điểm t_1 lúc này giá trị của ($x = 0$) tức là nó đã thay đổi giá trị so với trước đó, nhưng do giả thiết không thay đổi đồng thời nên ($\bar{x} = 0$) là vẫn ở giá trị 0 trước đó. Ta có các giá trị :

$$x = 0\ ;\ \bar{x} = 0\ ;\ y_1 = 1\ ;\ y_2 = 1$$

Thay vào ta tính ra được là $Y_1 = 1$ và $Y_2 = 0$ tức là ($Y_1\ Y_2 = 10$). Điền các giá trị vào cột t_1 của bảng.

Tại thời điểm t_2 là quá trình biến đổi nốt khi $\bar{x}$ vẫn chưa kịp thay đổi giá trị và ta có :

$$x = 0\ ;\ \bar{x} = 0\ ;\ y_1 = 1\ ;\ y_2 = 0$$

Thay vào hệ phương trình ta tính ra được $Y_1 = 0$ và $Y_2 = 0$ hay ($Y_1\ Y_2 = 00$). Kết quả ta có cột giá trị tại t_2 ở bảng.

Tại thời điểm t_3 lúc này biến $\bar{x}$ mới thay đổi sang giá trị ($\bar{x} = 1$) ta có các giá trị :

$$x = 0\ ;\ \bar{x} = 1\ ;\ y_1 = 0 \text{ và } y_2 = 0.$$

Thay các giá trị đó vào phương trình ta tính được $Y_1 = 0$ và $Y_2 = 1$ hay ($Y_1\ Y_2 = 01$). Kết quả ghi ở cột t_3.

Tại thời điểm t_4 là quá trình chuyển biến nốt sau khi $\bar{x}$ thay đổi giá trị trong ôtômat, ta có :

$$x = 0\ ;\ \bar{x} = 1\ ;\ y_1 = 0\ ;\ y_2 = 1$$

Thay các giá trị vào hàm chuyển biến trạng thái ta tính được trạng thái tiếp theo $Y_1 = 0$ và $Y_2 = 1$ hay ($Y_1\ Y_2 = 01$). Kết quả ta có cột t_3 trong bảng. Ở đây trạng thái tiếp theo ($Y_1\ Y_2 = 01$) chính là trạng thái trước đó ($y_1\ y_3 = 01$) tạo ra, cho nên nó là trạng thái ổn định mãi dù có kéo dài thêm thời gian.

Qua đó chúng ta thu được quá trình chuyển đổi trạng thái của ôtômat từ trạng thái ban đầu ($y_1\ y_2 = 11$) dưới tác động của tín hiệu vào x nhưng giá trị của x và $\bar{x}$ thay đổi không đồng thời là :

$$11 \rightarrow 10 \rightarrow 00 \rightarrow 01 \rightarrow 01$$

Ở đây sau quá trình biến đổi trạng thái chúng ta nhận được một trạng thái ổn định là ($Y_1\ Y_2 = 01$). Đó chính là trạng thái ổn định sai (nhẽ ra phải là trạng thái 10) xảy ra do x và $\bar{x}$ không thay đổi giá trị đồng thời, hiện tượng xảy ra như vậy chúng ta gọi là hazard cốt yếu.

Sau khi biết bản chất của hazard cốt yếu, muốn khắc phục được nó chúng ta chỉ cần thêm các phần tử làm chậm (trễ) trên đường phản hồi về của cac biến trạng thái ($y_1\ y_2$) tức là phá vỡ quá trình biến đổi trạng thái của ôtômat trong khoảng thời gian từ t_0 đến t_3. Có nghĩa thực chất là làm sao trễ tín hiệu ($y_1\ y_2$) để chờ cho $\bar{x}$ thay đổi xong trạng thái của nó thì hazard cốt yếu sẽ bị loại trừ.

Việc phân tích và nhận dạng được bản chất của hazard trong lý thuyết ôtômat là rất quan trọng, nó rất cần thiết cho quá trình tổng hợp và thiết kế ôtômat có nhớ. Khi hiểu rõ về hazard và bản chất của nó chúng ta sẽ có cách khắc phục sự ảnh hưởng của hazard trong hoạt động của hệ thống số, điều đó cho phép làm tăng chất lượng hoạt động cũng như tăng được độ tin cậy cho các hệ thống số cũng như đối với ôtômat có nhớ. Để nhận dạng được hazard ngoài việc phân tích cụ thể hoạt động của mạch điện, người ta còn áp dụng lý thuyết đạo hàm hàm Boole bậc cao hay lý thuyết mô phỏng để phân tích. Đó là những vấn đề rất lớn và khó, nhiều khi phải căn cứ vào một bài toán cụ thể hay một ôtômat cụ thể để xem xét, chính vì vậy không đề cập đến trong quyển sách này. Nhưng qua đó chúng ta cũng hiểu được là cùng một chức năng kỹ thuật cùng một nguyên lý mạch điện cơ bản, nhưng mỗi hãng sản xuất khác nhau lại cho ra kết quả về chất lượng hoạt động cũng như độ an toàn tin cậy hoạt động của nó rất khác nhau, quyết định tới tính kinh tế của mạch điện đó do từng hãng sản xuất là khác nhau. Tất cả các điều đó phần lớn đều liên quan tới việc khắc phục hazard của từng hãng đối với mạch điện chức năng đó như thế nào, vì khắc phục được hazard có nghĩa là làm tăng độ tin cậy hoạt động của mạch điện, mà mạch điện nào càng hoạt động tin cậy thì người sử dụng mạch điện đó sẽ càng nhiều.

Việc phân tích bản chất và nhận dạng hazard trong hệ thống số là một vấn đề rất lớn và rất quan trọng, ở đây chúng ta sẽ nêu ra một vài nhận xét chung để lưu ý khi phân tích một hiện tượng hazard nếu gặp.

* Hazard tĩnh thường chỉ xảy ra trong hệ tổ hợp và chúng ta có thể khắc phục nó bằng cách thêm các phần tử làm trễ trên đường truyền tín hiệu. Mục đích là làm sao tín hiệu vào (dù có chạy đua) cũng đến được đầu ra tại cùng một thời điểm.

* Hazard cốt yếu thường hay gặp trong mạch ôtômat có nhớ không đồng bộ, có thể khắc phục bằng cách thêm các phần tử làm chậm trên đường phản hồi của các biến trạng thái y.

* Hazard trạng thái ổn định và hazard chu trình thường xảy ra trong các ôtômat không đồng bộ, loại hazard này ta có thể dùng phương pháp mã hóa thích hợp cho các trạng thái của ôtômat, thực chất là mã hóa sao cho không để xảy ra đua trạng thái ở trong mạch.

* Hazard động thường cũng chỉ xảy ra trong các mạch tổ hợp, nó không xảy ra ở các mạch lôgic hai lớp nối tiếp nhau. Loại hazard này nếu có cũng không thật sự nguy hiểm như các loại hazard khác.

Nếu trường hợp cuối cùng khi thiết kế một ôtômat mà ta không thể khắc phục hết hazard hay hoạt động sai nhầm trong nó, thì ta chỉ còn có thể khắc phục bằng cách dùng xung đồng bộ hay thậm chí phải thiết kế lại chức năng cho ôtômat đó.

5.5.4. TỔNG HỢP ÔTÔMAT CÓ NHỚ KHÔNG ĐỒNG BỘ

Phần trên đã nói tất cả các hiện tượng có thể xảy ra trong ôtômat không đồng bộ, từ chạy đua trạng thái đến hazard có thể xuất hiện trong ôtômat, từ mã hóa trạng thái

cũng như mạch phải đơn giản nhất v.v... Trong phần này trình bày việc tổng hợp hay thiết kế cụ thể một ôtômat có nhớ không đồng bộ, sao cho nó hoạt động tốt nhất về kỹ thuật, tức là phải chống hazard do chạy đua tín hiệu vào (x), đồng thời phải mã hóa sao cho trong hoạt động của ôtômat chống được chạy đua (y), đồng thời lại phải rất gọn. Những hoạt động kỹ thuật quan trọng nhất để đảm bảo cho ôtômat hoạt động được hoàn toàn tin cậy, sẽ được nêu trong ví dụ sau.

Để tổng hợp được một ôtômat có nhớ chúng ta phải trải qua các bước hay trình tự sau :

+ Từ nhiệm vụ của ôtômat có nhớ chúng ta phải lập thành bảng chuyển biến trạng thái hay bảng chức năng của ôtômat đó.

+ Tiếp theo phải thực hiện rút gọn hay tối thiểu hóa số lượng trạng thái cho ôtômat.

+ Sau đó thực hiện mã hóa trạng thái cho ôtômat sao cho tránh được hazard cũng như hiện tượng chạy đua của tín hiệu vào x cũng như trạng thái của ôtômat y.

+ Rút gọn và tìm hệ phương trình hàm kích cũng như hàm ra của ôtômat ở dạng đơn giản gọn gàng nhất.

+ Xây dựng và vẽ sơ đồ nguyên lý thực hiện ôtômat.

Ví dụ : Tổng hợp ôtômat có nhớ không đồng bộ có hai đầu vào x_1 và x_2 với một đầu ra Z. Hoạt động theo yêu cầu sau :

+ Tín hiệu vào ở đầu vào (x_1, x_2) là 0 hoặc 1 xuất hiện ngẫu nhiên bất kỳ.

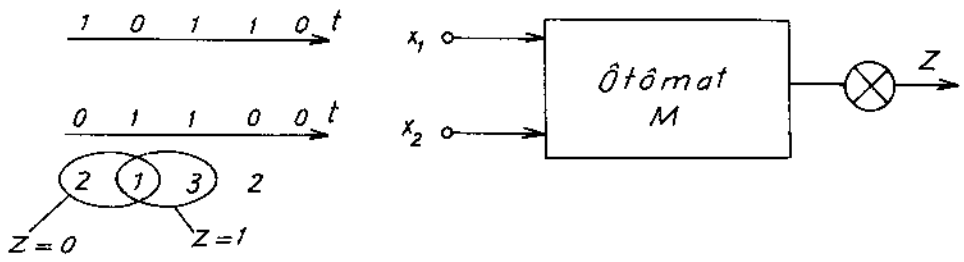
+ Đầu ra Z = 1 nếu như gặp tổ hợp các giá trị biến vào tiếp theo (coi như là một số nhị phân) lớn hơn tổ hợp của giá trị hiện tại (hay trước đó). Ví dụ tại thời điểm ban đầu t_0 có (x_1 x_2 = 01) tổ hợp đó có giá trị là 1, còn ở thời điểm tiếp theo t_1 tổ hợp của tín hiệu vào giả sử gặp lại là (x_1 x_2 = 11) có giá trị là 3, ta thấy (3 > 1) về giá trị thì Z = 1.

+ Z = 0 trong các trường hợp khác (nhỏ hơn hay bằng).

+ Mô tả hoạt động của ôtômat theo ôtômat Moore và dùng trigơ JK (JKFF) thực hiện.

+ Phải dùng và áp dụng các yêu cầu kỹ thuật tốt nhất để chống hazard do chạy đua tín hiệu vào (x_1, x_2), và chạy đua trạng thái (y) nếu gặp khi xây dựng ôtômat.

Sơ đồ khối của ôtômat được minh họa ở hình 5.64.



Hình 5.64. Chức năng của ôtômat.

Các loại ôtômat đồng bộ cũng như ôtômat kiểu xung mà chúng ta đã tổng hợp, và thực hiện chúng qua các ví dụ đã được nêu ở các phần trước chúng ta thấy là các tín hiệu kích vào cho các ôtômat đó là các giá trị lôgic (0 hoặc 1) hay các xung điện thuần tuý, các tín hiệu kích vào đó chúng không được kèm theo "nghĩa lý" hay ý nghĩa gì ở trong các tín hiệu vào đó. Nhưng đối với ôtômat không đồng bộ thì tín hiệu vào của ôtômat ở một mức độ cao hơn (dù nó vẫn là tín hiệu lôgic hay tín hiệu điện đưa vào), tức là kèm theo tín hiệu đưa vào ôtômat ta còn gắn thêm vào đó ý nghĩa (ngữ nghĩa) cho tín hiệu đó. Có nghĩa là một tín hiệu điện kích vào ôtômat có "nghĩa lý" kèm theo đối với tín hiệu vào đó. Trong ví dụ này ý nghĩa của tín hiệu vào ở dạng rất đơn giản là (với $x_1x_2 = 01$ có nghĩa giá trị là 1, còn với tín hiệu vào $x_1x_2 = 10$ lại có nghĩa giá trị là 2, hay với bộ tín hiệu $x_1x_2 = 11$ nó lại có giá trị là 3). Dựa trên các quy ước "ngữ nghĩa" đó chúng ta phải thực hiện "so sánh" các ý nghĩa đó mà quyết định tới đáp ứng ra của ôtômat là 0 hay 1 (đầu ra Z = 0 hay Z = 1). Điều đó cho phép chúng ta xây dựng nên một ôtômat hoạt động gần với hoạt động của con người hơn một bước, vì đáp ứng hay xử thế của con người có thể được dựa trên "ý nghĩa" của tín hiệu vào hay thông tin vào mà nó nhận được, mà các đáp ứng hay ứng xử đó của con người không chỉ phụ thuộc vào quan hệ lôgic (0 hay 1) hay tác động trực tiếp vào con người kiểu như một tín hiệu (một xung) đơn thuần.

Ở đây lưu ý thêm một lần nữa là cần phải áp dụng tất cả những yêu cầu kỹ thuật tốt nhất, để đảm bảo sự hoạt động cho ôtômat là tin cậy nhất, khi chúng ta xây dựng và thực hiện thiết kế ôtômat được đưa ra.

Để thực hiện được ôtômat bước đầu tiên ta phải nhận xét và phân tích hoạt động của nó trong mọi khả năng, để từ đó tìm được những trạng thái cần thiết nhất, đại diện cho tất cả các khả năng có thể xảy ra cho ôtômat. Từ những trạng thái đó sẽ xây dựng được bảng chức năng hay bảng chuyển biến trạng thái của ôtômat.

Các khả năng xảy ra cho ôtômat khi cho tín hiệu vào được nêu ra như sau : Ở đây chúng ta lần lượt giả sử tại trạng thái t_0 các tổ hợp tín hiệu vào được xuất hiện, sau đó kèm các tổ hợp tại t_1 của các tín hiệu vào tiếp theo từ đó ta có các quan hệ sau :

	giá trị			
	t_0	t_1	t_0	t_1
x_1	0	0	(0 − 0)	
x_2	0	0		
x_1	0	0	(0 − 1) = q_1	
x_2	0	1		
x_1	0	1	(0 − 2) = q_2	
x_2	0	0		
x_1	0	1	(0 − 3)	
x_2	0	1		

	giá trị			
	t_0	t_1	t_0	t_1
x_1	0	0	(1 − 0) = q_3	
x_2	1	0		
x_1	0	0	(1 − 1)	
x_2	1	1		
x_1	0	1	(1 − 2)	
x_2	1	0		
x_1	0	1	(1 − 3) = q_4	
x_2	1	1		

	giá trị					giá trị			
	t_0	t_1	t_0	t_1		t_0	t_1	t_0	t_1
x_1	1	0	(2 – 0) = q_5		x_1	1	0	(3 – 0)	
x_2	0	0			x_2	1	0		
x_1	1	0	(2 – 1)		x_1	1	0		
x_2	0	1			x_2	1	1	(3 – 1) = q_7	
x_1	1	1			x_1	1	1		
x_2	0	0	(2 – 2)		x_2	1	0	(3 – 2) = q_8	
x_1	1	1			x_1	1	1		
x_2	0	1	(2 – 3) = q_6		x_2	1	1	(3 – 3)	

Giả sử : tại thời điểm t_0 chúng ta cho tín hiệu vào trên đầu vào $x_1\ x_2 = 00$, do tín hiệu vào xuất hiện ngẫu nhiên nên có bốn khả năng xảy ra tiếp theo của tín hiệu vào tại thời điểm tiếp theo t_1, chúng là các tổ hợp giá trị là : $x_1\ x_2 = 00$, $x_1x_2 = 01$, $x_1x_2 = 10$ và $x_1x_2 = 11$. Các khả năng có thể xảy ra đó của tín hiệu vào tại thời điểm t_1 chúng ta viết tiếp theo sau giá trị $x_1x_2 = 00$ ở thời điểm t_0. Đồng thời các tổ hợp tín hiệu vào đó lại được coi như là một "giá trị" của con số nhị phân, cho nên chúng ta cũng có được giá trị của tín hiệu vào tại thời điểm t_0 trước đó đối với giá trị của bộ tín hiệu vào tại thời điểm t_1 được ghi tương ứng ở bên cạnh. Với cách làm tương tự đối với tổ hợp của tín hiệu vào tại thời điểm t^o như các trường hợp $x_1x_2 = 01$, $x_1x_2 = 10$ và $x_1x_2 = 11$.

Sau khi tìm được các khả năng xảy ra đối với bộ tín hiệu vào (x_1, x_2) tại thời điểm t_0 và t_1 chúng ta có nhận xét kỹ hơn như sau : tại thời điểm t_0 nếu tín hiệu vào $x_1x_2 = 00$ (ứng với giá trị bằng 0), thì tại thời điểm tiếp theo t_1 chúng ta có các quan hệ giá trị của số nhị phân giữa thời điểm trước đó và thời điểm tiếp theo là : (0 – 0), (0 – 1), (0 – 2) và (0 – 3) dựa vào quan hệ giá trị đó ta sẽ dễ dàng nhận ra bộ tín hiệu vào tiếp theo (ứng với một số nhị phân) có giá trị lớn hơn hay bé hơn hoặc bằng nhau so với giá trị của tín hiệu vào trước đó để điền giá trị cho đáp ứng ra Z sau này ở bảng chức năng.

Trong các tổ hợp giá trị của số nhị phân ứng với bộ tín hiệu (x_1, x_2) cho vào cho ta thấy ở trường hợp gặp quan hệ (0 – 3) sẽ xuất hiện chạy đua của tín hiệu vào x_1 với x_2 vì cả hai giá trị của chúng đều cùng thay đổi một lần vì tại t_0 ta có $x_1\ x_2 = 00$ còn tại thời điểm tiếp theo t_1 ta có bộ tín hiệu $x_1\ x_2 = 11$. Do nhiệm vụ đề ra là phải chống hazard do chạy đua của tín hiệu vào, nên trong bảng trạng thái sau này khi gặp hiện tượng chạy đua của tín hiệu vào khi kích tín hiệu vào tạo ra trạng thái tiếp theo, thì trạng thái mới sinh ra ta sẽ coi như đó là trạng thái không xác định (–) tức là không quan lý trạng thái đó nếu xuất hiện chạy đua x, có nghĩa là dù tín hiệu vào đó có gây ra chạy đua (xuất hiện hazard) hay không ta cũng không để ý, làm như vậy coi như ta đã chống được hazard do chạy đua của tín hiệu vào. Tương tự như vậy

chúng ta cũng sẽ gặp các trường hợp chạy đua của tín hiệu vào là : $\boxed{(1-2)}$ $\boxed{(2-1)}$, và $\boxed{(3-0)}$, đó là những vị trí sẽ không quản lý trạng thái trong bảng chức năng hay bảng chuyển biến trạng thái sau này. Một trong những trường hợp đặc biệt nữa mà chúng ta gặp khi cho tổ hợp của tín hiệu vào là : giữa thời điểm trước đó t_0 và thời điểm tiếp theo t_1 chúng tạo ra các giá trị bằng nhau, ví dụ như (0 - 0), (1 - 1), (2 - 2) và (3 - 3). Khi gặp trường hợp như vậy chúng ta có thể coi như kéo dài tín hiệu vào từ t_0 đến tận t_1 mà không thay đổi, nếu chúng ta coi tín hiệu vào ở trong trường hợp đó (bằng nhau) là tồn tại lâu và kéo dài tới tận thời điểm mới t_1, điều đó có nghĩa là chúng ta coi như trạng thái của ôtômat là giữ nguyên không thay đổi từ thời điểm t_0 tới trạng thái tiếp theo là thời điểm t_1. Hiện tượng đó cũng được thể hiện trên bảng chuyển tiếp hay bảng chức năng của ôtômat khi ta thực hiện sau đây.

Sau khi phân tích và xem xét các trường hợp đặc biệt xảy ra khi cho bộ tín hiệu vào (x_1, x_2) thì phần còn lại sẽ là các trường hợp bình thường xảy ra đối với ôtômat, do đó chúng ta chỉ cần phân biệt các trạng thái tổ hợp bình thường đó của tín hiệu vào bằng trạng thái cụ thể của ôtômat nữa là xong, nghĩa là dựa vào trạng thái của ôtômat chúng ta cũng sẽ biết được hoạt động ở đầu vào tác động của nó. Cụ thể các trạng thái cần phân biệt hay là các trạng thái thực thể cần thiết của ôtômat cần xây dựng là :

Nếu gặp trường hợp (0 - 1) của các giá trị tín hiệu vào chúng ta cho nó là trạng thái q_1 của ôtômat. Trường hợp (0 - 2) ứng với trạng thái q_2, tổ hợp (1 - 0) ứng với trạng thái q_3 và v.v... cho đến tổ hợp (3 - 2) sẽ ứng với trạng thái q_8. Như vậy số lượng trạng thái cần thiết của ôtômat phải xây dựng sẽ là tám.

Từ những hiểu biết trên chúng ta tìm được bảng chức năng cũng như bảng chuyển biến trạng thái của ôtômat được chỉ ra ở trên hình 5.65.

$t_0 < t_1 < t_2$

S \ x_1x_2	0 (00)	1 (01)	3 (11)	2 (10)	Z
t_0 t_1					
q_1 (0−1)	q_3	q_1	q_4	−	1
q_2 (0−2)	q_5	−	q_6	q_2	1
q_3 (1−0)	q_3	q_1	−	q_2	0
q_4 (1−3)	−	q_7	q_4	q_8	1
q_5 (2−0)	q_5	q_1	−	q_2	0
q_6 (2−3)	−	q_7	q_6	q_8	1
q_7 (3−1)	q_3	q_7	q_4	−	0
q_8 (3−2)	q_5	−	q_6	q_8	0

Hình 5.65. Bảng trạng thái của ôtômat phải thực hiện.

Cách điền bảng trạng thái hay cách xây dựng bảng trạng thái cũng rất đơn giản, ta dựa theo trật tự thời gian của các giá trị ($t_0 < t_1 < t_2$) mà ghi trạng thái tương ứng trong khu vực trạng thái tiếp theo hay là khu vực hàm f. Còn khu vực hàm g là khu đáp ứng ra ta điền giá trị 0 hay 1 theo yêu cầu của nhiệm vụ là : giá trị của bộ tín hiệu vào tiếp theo lớn hơn giá trị của bộ tín hiệu vào trước đó thì Z = 1 nên :

$$t_0 \quad t_1$$

Ví dụ : ứng với $q_1 (0 - 1)$ ta thấy 0 < 1 nên thỏa mãn điều kiện giá trị tiếp theo lớn hơn giá trị trước đó vậy đầu ra Z = 1.

$q_2 (0 - 2)$ đầu ra Z = 1 vì 0 < 2

$q_3 (1 - 0)$ đầu ra Z = 0 vì 1 > 0

Với cách làm tương tự chúng ta sẽ điền đầy giá trị cho cột Z các đáp ứng ra của ôtômat.

Sau khi tìm được bảng chức năng cho ôtômat không đồng bộ cần phải xây dựng chỉ ra trên hình 5.65, đây là bước quan trọng nhất trong quá trình thực hiện ôtômat có nhớ. Đồng thời trong quá trình xây dựng bảng chức năng chúng ta cũng đã chống được hiện tượng hazard do chạy đua của tín hiệu vào (x_1, x_2).

Bước tiếp theo chúng ta thực hiện rút gọn bảng chức năng của ôtômat, để có được số lượng trạng thái ít nhất hay trạng thái tối thiểu cần thiết. Ta lại qua các bước tìm phân hoạch thế (PHT) và phân hoạch tương đương (PHTĐ) của ôtômat đó.

Nhìn vào bảng chức năng của ôtômat chúng ta dễ dàng tìm được các phân hoạch thế sau :

$$\Pi_1 = (\overline{q_3, q_5}) \qquad \text{và} \qquad \Pi_2 = (\overline{q_4, q_6})$$

Đồng thời chúng ta cũng thấy các (PHT) tìm được có tính chất của phân hoạch tương đương :

$$g\ (q_3) = g\ (q_5) = 0 \text{ và}$$

$$g\ (q_4) = g\ (q_6) = 1$$

Như vậy $\Pi_1 = (\overline{q_3, q_5})$ và $\Pi_1 = (\overline{q_4, q_6})$ vừa là phân hoạch thế vừa là phân hoạch tương đương nên chúng tạo nên tập phân hoạch thế tương đương, hay nói một cách khác là : chúng ta có thể khẳng định từng phần tử trong các khối của từng phân hoạch có thể thay thế được cho nhau, tức là :

$$q_3 \sim q_5 \qquad \text{và} \qquad q_4 \sim q_6$$

Cụ thể là cứ vị trí q_5 ở trong bảng chức năng ta viết thành q_3, và vị trí q_6 ta thay bằng trạng thái q_4, sau khi thay thế xong chúng ta xóa đi các dòng (hàng) giống nhau và thu được ôtômat rút gọn (ôtômat min) được mô tả trên hlnh 5.66.

Sau khi thực hiện xong việc rút gọn ôtômat bước tiếp theo là thực hiện mã hóa (đặt tên) trạng thái cho nó. Nhiệm vụ đặt ra cho bước mã hóa này là do yêu cầu đảm bảo các nhiệm vụ kỹ thuật tốt nhất cho ôtômat, nên mã hóa sao cho lắp mạch đẹp và

Ôtômat min

S \ x_1x_2	00	01	11	10	Z
q_1	q_3	q_1	q_4	–	1
q_2	q_3	–	q_4	q2	
q_3	q_3	q_1	–	q_2	0
q_4	–	q_7	q_4	q_8	1
q_7	q_3	q_7	q_4	–	0
q_8	q_3	–	q_4	q_8	0

Hình 5.66. Ôtômat min thu được.

gọn gàng đồng thời phải chống được chạy đua trạng thái (chạy đua y do thời gian lật chuyển của các trigơ là khác nhau) để chống hazard chạy đua trạng thái cho ôtômat được xây dựng, cụ thể cách làm như sau :

Trước hết chúng ta có nhận xét : Khi nhìn vào bảng mô tả hoạt động của ôtômat tối thiểu ta thấy dưới tác động vào của tín hiệu x_1 x_2 = 00 và x_1x_2 = 11 sẽ không thể xảy ra chạy đua trạng thái (chạy đua y) được vì dưới tác động của hai cặp tín hiệu vào đó tạo ra trạng thái tiếp theo là hai cột là các trạng thái q_3 và cột chỉ toàn trạng thái q_4. Vậy ta có thể tạo đường thiết lập trạng thái riêng cho ôtômat để tạo ra trạng thái q_3 hay q_4 nếu gặp tổ hợp các tín hiệu vào là x_1x_2 = 00 hay x_1x_2 = 11, hiện tượng này gọi là "thiết lập trạng thái cứng" nên không thể xảy ra chạy đua trạng thái hay chạy đua y được. Sau này trên mạch chúng ta sẽ thấy cụ thể hơn.

Như vậy để mã hóa tránh được chạy đua y cho ôtômat chúng ta chỉ còn phải xem xét mã hóa cho hai cột còn lại của ôtômat ứng với bộ tín hiệu kích vào là x_1x_2 = 01 và x_1x_2 = 10.

Để đảm bảo lắp ráp tạo nên mạch đẹp, để quản lý, thay thế và sửa chữa cho ôtômat, chúng ta sẽ mã hóa cho ôtômat dùng một phân hoạch thế. Phân hoạch thế dùng để mã hóa trạng thái cho ôtômat chúng ta tìm được có dạng :

$$\Pi = (\overline{q_1, q_2, q_3} \ ; \ \overline{q_4, q_7, q_8})$$

Do ôtômat rút gọn chỉ còn có sáu trạng thái (là các trạng thái q_1, q_2, q_3, a_4, q_7 và q_8) cho nên số lượng biến cần dùng để mã hóa là :

$$k = \left[\log \frac{6}{2}\right] = 3$$

Ta cần có 3 biến trạng thái để mã hóa (đặt tên) cho sáu trạng thái của ôtômat. Đến đây nhiệm vụ cụ thể để mã hóa trạng thái cho ôtômat chỉ còn là : mã hóa trạng thái sao cho tránh chạy đua y, tức là mã hóa sao cho không để xảy ra hiện tượng có hai biến trạng thái cùng thay đổi đồng thời (⩾ 2 biến trạng thái) khi ôtômat chuyển

biến trạng thái, dưới tác động của tập tín hiệu vào (x_1, x_2). Bảng mã hóa (đặt tên) cụ thể cho các trạng thái của ôtômat chỉ ra trên hình 5.67.

y_2y_3 \ y_1	0	1
0 0	q_3	q_4
0 1	q_1	q_7
1 0	q_2	q_8
1 1	–	–

Hình 5.67. Bảng mã hóa trạng thái.

Từ bảng mã hóa (đặt tên) cụ thể cho từng trạng thái của ôtômat ta có kết quả tên (mã) như sau :

$$q_3 = 0\ 0\ 0 \qquad q_4 = 1\ 0\ 0$$

$$q_1 = 0\ 0\ 1 \qquad q_7 = 1\ 0\ 1$$

$$q_2 = 0\ 1\ 0 \qquad q_8 = 1\ 1\ 0$$

Với cách đặt trên mã hóa như vậy cho các trạng thái của ôtômat sẽ chống được hazard trạng thái, chống được hiện tượng chạy đua y cho ôtômat, hiện tượng này sẽ được minh họa và chỉ ra như sau :

Nhìn vào bảng chuyển biến trạng thái của ôtômat trên hình 5.66 chúng ta thấy : dưới tác động của tín hiệu vào là $x_1x_2 = 01$ làm cho ôtômat chuyển từ trạng thái trước đó t_0 là q_3 sang trạng thái tiếp theo là q_1

$$x_1x_2 = 01 \rightarrow (q_3 - q_1) \text{ tức là } (000 \rightarrow 001)$$

như vậy chỉ có một trigơ (y_3) là thay đổi trạng thái khi trạng thái của ôtômat chuyển từ q_3 sang q_1 $(q_3 \rightarrow q_1)$. Như vậy không xảy ra đua trạng thái :

Tương tự như vậy ta có tín hiệu vào $x_1x_2 = 10$:

$$x_1x_2 = 10 \rightarrow (q_4 \rightarrow q_8) \text{ tức là } (100 \rightarrow 110)$$

ta thấy cũng chỉ có một trigơ (y_2) một biến trạng thái là y_2 lật chuyển trạng thái, khi ôtômat chuyển trạng thái từ q_4 sang q_8 $(q_4 \rightarrow q_8)$ dưới tác động của bộ tín hiệu vào $x_1x_2 = 10$. Khi chỉ có một biến trạng thái (một trigơ) bị lật chuyển thì không thể xuất hiện hiện tượng chạy đua y, tức là chống được hazard trạng thái cho ôtômat. Kết quả ta thu được như vậy nhờ cách mã hóa trạng thái chống đua y cho ôtômat mà có.

Với cách kiểm tra tương tự cho các quá trình chuyển biến trạng thái có thể xảy ra đối với ôtômat, dưới tác động của tín hiệu vào, ta thấy mã hóa trạng thái như vậy cho ôtômat hoàn toàn có thể chống được chạy đua y đảm bảo hoạt động rất tốt cho ôtômat.

Bảng mã hóa trạng thái của ôtômat cần phải thiết kế được chỉ ra trên hình 5.68. Bảng mã hóa này chúng ta viết theo bìa Kácnô để rút gọn dễ dàng sau này.

	$y_1y_2y_3$ \ x_1x_2	00	01	11	10	Z
q_3	0 0 0	0 0 0	0 0 1	——	0 1 0	0
q_1	0 0 1	0 0 0	0 0 1	1 0 0	——	1
–	0 1 1	——	——	——	——	–
q_2	0 1 0	0 0 0	——	1 0 0	0 1 0	1
q_8	1 1 0	0 0 0	——	1 0 0	1 1 0	0
–	1 1 1	——	——	——	——	–
q_7	1 0 1	0 0 0	1 0 1	1 0 0	——	0
q_4	1 0 0	——	1 0 1	1 0 0	1 1 0	1

a)

y	Y	J	K
0	0	0	–
0	1	1	–
1	0	–	1
1	1	–	0

b)

Hình 5.68. Bảng mã hóa không có chạy đua y.

Ở đây ta còn có một lưu ý nhỏ nữa là : nếu trạng thái trước đó là q_1 = (001) dưới tác động của tín hiệu vào x_1x_2 = 11 sẽ chuyển sang trạng thái tiếp theo q_4 = (100), nếu trong quá trình chuyển biến trạng thái bình thường của ôtômat thì sẽ xảy ra hiện tượng chạy đua trạng thái hay chạy đua y, do xuất hiện hai biến trạng thái (y_1 và y_3) hay hai trigơ cùng lật chuyển một lần, suy nghĩ như vậy là chính xác. Nhưng do toàn bộ cột trạng thái của ôtômat dưới tác động của tín hiệu vào x_1x_2 = 11 là q_4 = (100) vậy chúng thực hiện "thiết lập cứng", có nghĩa là lắp mạch sao cho cứ gặp tín hiệu x_1x_2 = 11 thì thiết lập trạng thái của ôtômat về q_4 = (100). Khi đã thực hiện thiết lập cứng (giống như xóa trạng thái của ôtômat) thì hiện tượng chạy đua trạng thái dù có theo hình thức nhưng cũng sẽ không bao giờ xảy ra được, hay ảnh hưởng của chạy đua trạng thái sẽ không xuất hiện trong quá trình hoạt động của ôtômat.

Với cách làm "thiết lập cứng" trạng thái của ôtômat, điều đó cũng sẽ được áp dụng khi gặp bộ tín hiệu kích vào là x_1x_2 = 00. Khi gặp bộ tín hiệu vào như vậy mặc dù trạng thái trước đó của ôtômat là gì thì trạng thái tiếp theo của nó đều phải chuyển về q_3 = (000), hiện tượng này giống như xóa trạng thái của ôtômat. Điều này chỉ có thể thực hiện được (thiết lập cứng trạng thái cho ôtômat) khi toàn bộ trạng thái của ôtômat trên một cột của bảng chuyển biến trạng thái phải giống hệt nhau (tới cùng một trạng thái). Điều này cũng sẽ được thấy rõ trên mạch điện cụ thể của ôtômat.

Tiếp theo chúng ta tìm bảng kích cho ôtômat bằng cách kết hợp bảng mã hóa với bảng chuyển biến trạng thái của ôtômat chuyên dùng để tổng hợp. Bảng kích được tìm thấy trên hình 5.69.

Bảng kích của ôtômat phải thực hiện.

x_1x_2 / $y_1y_2y_3$	00			01			11			10			Z
	J_1K_1	J_2K_2	J_3K_3	J_1K_1	J_2K_2	J_3K_3	J_1K_1	J_2K_2	J_3K_3	J_1K_1	J_2K_2	J_3K_3	
0 0 0	0 –	0 –	0 –	0 –	0 –	1 –	—	—	—	0 –	1 –	0 –	0
0 0 1	0 –	0 –	– 1	0 –	0 –	– 0	1 –	0 –	– 1	—	—	—	1
0 1 1	—	—	—	—	—	—	—	—	—	—	—	—	–
0 1 0	0 –	– 1	0 –	—	—	—	1 –	– 1	0 –	0 –	– 0	0 –	1
1 1 0	– 1	– 1	0 –	—	—	—	– 0	– 1	0 –	– 0	– 0	0 –	0
1 1 1	—	—	—	—	—	—	—	—	—	—	—	—	–
1 0 1	– 1	0 –	– 1	– 0 0	–	– 0	– 0 0	–	– 1	—	—	—	0
1 0 0	—	—	—	– 0 0	–	1 –	– 0 0	–	0 –	0 –	1 –	0 –	1

Hình 5.69. Bảng kích của ôtômat.

Từ bảng kích chúng ta tìm được hệ phương trình kích như sau :

$$J_1 = x_1 x_2$$

$$K_1 = \bar{x}_1 \bar{x}_2$$

$$J_2 = x_1 \bar{x}_2$$

$$K_2 = \bar{x}_1 + x_2$$

$$J_3 = \bar{x}_1 x_2$$

$$K_3 = x_1 + \bar{x}_2$$

$$Z = \bar{y}_1 y_3 + \bar{y}_1 y_2 + y_1 \bar{y}_2 \bar{y}_3$$

Từ hệ phương trình kích chúng ta vẽ mạch điện thực hiện chức năng của ôtômat dựa theo hệ phương trình kích đó. Đồng thời lưu ý phần tín hiệu "thiết lập cứng" cho mạch điện. Quan hệ giữa hệ phương trình kích và "thiết lập cứng" theo quan hệ mạch điện như sau.

Thiết lập cứng xảy ra với bộ tín hiệu vào $x_1x_2 = 00$ và $x_1x_2 = 11$, cụ thể là bộ tín hiệu vào ($\bar{x}_1 \bar{x}_2$) lật chuyển các trigơ ($y_1\ y_2\ y_3 = 000 = q_3$) trong mọi trường hợp hoạt động của ôtômat còn bộ tín hiệu ($x_1\ x_2$) lật chuyển các trigơ về trạng thái ($y_1y_2y_3 = 100 = q_4$).

Quan hệ cụ thể của các tín hiệu kích là :

$$J_1 = x_1 x_2 + \text{Thiết lập cứng} = x_1 x_2 + x_1 x_2 = x_1 x_2$$

$$K_1 = \bar{x}_1 \bar{x}_2 + \text{TLC} = \bar{x}_1 \bar{x}_2 + \bar{x}_1 \bar{x}_2 = \bar{x}_1 \bar{x}_2$$

$J_2 = x_1 \bar{x}_2$

$K_2 = \bar{x}_1 + x_2$ + TLC $= \bar{x}_1 + x_2 + x_1 x_2 + \bar{x}_1 \bar{x}_2 = \bar{x}_1 + x_2$

$J_3 = \bar{x}_1 x_2$

$K_3 = x_1 + \bar{x}_2$ + TLC $= x_1 + \bar{x}_2 + x_1 x_2 + \bar{x}_1 \bar{x}_2 = x_1 + \bar{x}_2$

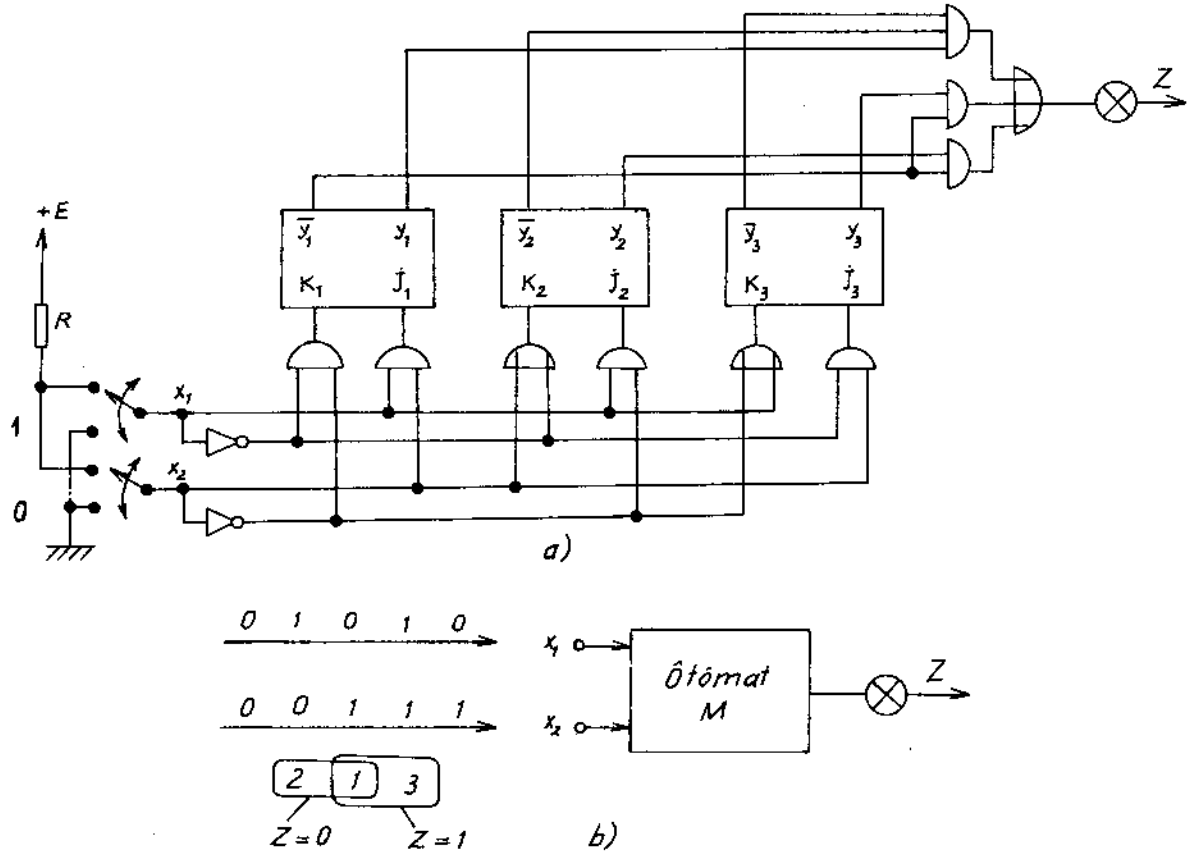
Tóm lại hệ phương trình kích kể cả kèm theo quan hệ của tín hiệu "thiết lập cứng" (TLC) của ôtômat ta thu được là :

$$J_1 = x_1 x_2 \qquad J_2 = x_1 \bar{x}_2 \qquad J_3 = \bar{x}_1 x_2$$

$$K_1 = \bar{x}_1 \bar{x}_2 \qquad K_2 = \bar{x}_1 + x_2 \qquad K_3 = x_1 + \bar{x}_2$$

$$Z = \bar{y}_1 y_3 + \bar{y}_1 y_2 + y_1 \bar{y}_2 \bar{y}_3$$

Sơ đồ mạch điện cụ thể của ôtômat cần phải thực hiện dựa theo hệ phương trình kích tổng hợp được biểu diễn trên hình 5.70.



Hình 5.70. Sơ đồ mạch điện của ôtômat cần phải thực hiện.

Với mạch điện đó, khi chúng ta cho tổ hợp tín hiệu vào trên đầu vào (x_1, x_2), nếu giá trị của tập tín hiệu vào tiếp theo lớn hơn tổ hợp giá trị của tín hiệu vào trước đó (hay hiệu đã có) thì đầu ra đèn sẽ sáng (Z = 1), còn các tổ hợp khác của tập tín hiệu vào (nếu không thỏa mãn lớn hơn giá trị trước đó) thì Z = 0.

Ngoài việc thực hiện được đúng chức năng đã yêu cầu, ở mạch điện đó đã thực hiện

chống chạy đua tín hiệu vào x, đảm bảo không xảy ra hazard do chạy đua tín hiệu vào, cũng như mã hóa trạng thái cho ôtômat không để xuất hiện chạy đua trạng thái hay chạy đua y, đảm bảo hazard trạng thái cũng không thể xuất hiện. Việc mã hóa cho ôtômat dùng một phân hoạch thế nên sơ đồ mạch điện đơn giản.

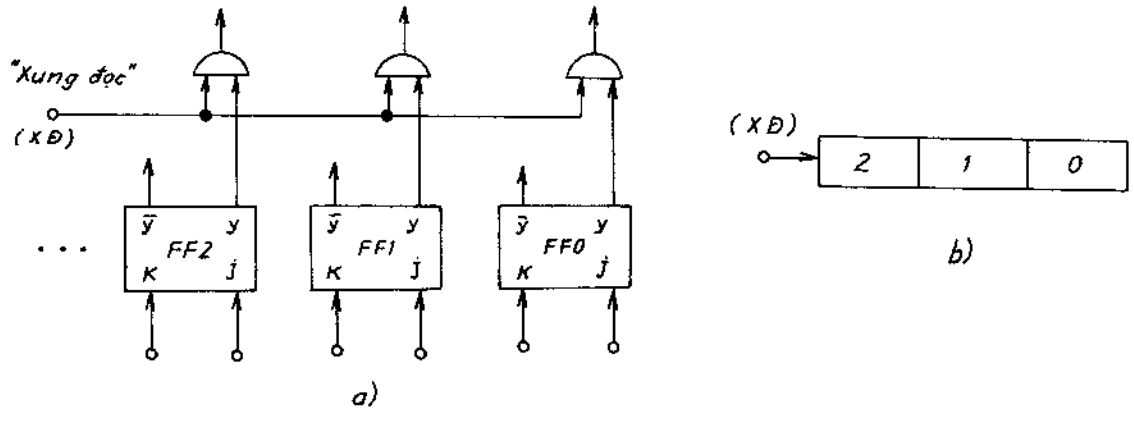
Đến đây chúng ta đã đề cập đến các loại ôtômat có nhớ khác nhau như : ôtômat đồng bộ hoạt động theo xung nhịp, ôtômat kiểu xung là tín hiệu vào ở dạng xung điện và ôtômat kiểu điện thế là tín hiệu vào là điện thế hay công tắc bật. Khi đã nghiên cứu các phương pháp thiết kế ôtômat có nhớ với chức năng bất kỳ, chúng ta có thể thiết kế các chức năng của ôtômat với bất kỳ một loại trigơ nào (DFF, RSFF, JKFF hay TFF).

§5.6. MỘT VÀI LOẠI ÔTÔMAT CÓ NHỚ ĐIỂN HÌNH

Phần trên chúng ta chỉ nói tới "ôtômat có nhớ" một cách tổng quát, ứng với hệ thống số thực hiện một chức năng kỹ thuật được đề ra, các hệ thống như vậy không có được một tên gọi cụ thể mà chỉ có tên chung là ôtômat. Ở đây trong phần này khi chúng ta nói tới vài loại ôtômat có nhớ điển hình, tức là đó là các ôtômat có nhớ có tên gọi và rất hay gặp trong hệ thống số, sau đây chúng ta sẽ nêu ra các ôtômat có nhớ đặc biệt này và tên gọi chức năng và cả ứng dụng thực tế của chúng trong hệ thống số nói chung.

5.6.1. THANH GHI TĨNH

Đây là một cấu trúc ôtômat có nhớ đơn giản nhất bao gồm các trigơ sắp xếp cạnh nhau. Nhiệm vụ của thanh ghi tĩnh là ghi giữ thông tin vào không được làm mất thông tin, đồng thời có thể đọc thông tin nhiều lần mà thông tin vẫn được duy trì. Thanh ghi tĩnh chỉ mất thông tin khi bị "xóa" hay bị mất nguồn cung cấp. Sơ đồ nguyên lý của một thanh ghi tĩnh ba bit được minh họa trên hình 5.71.

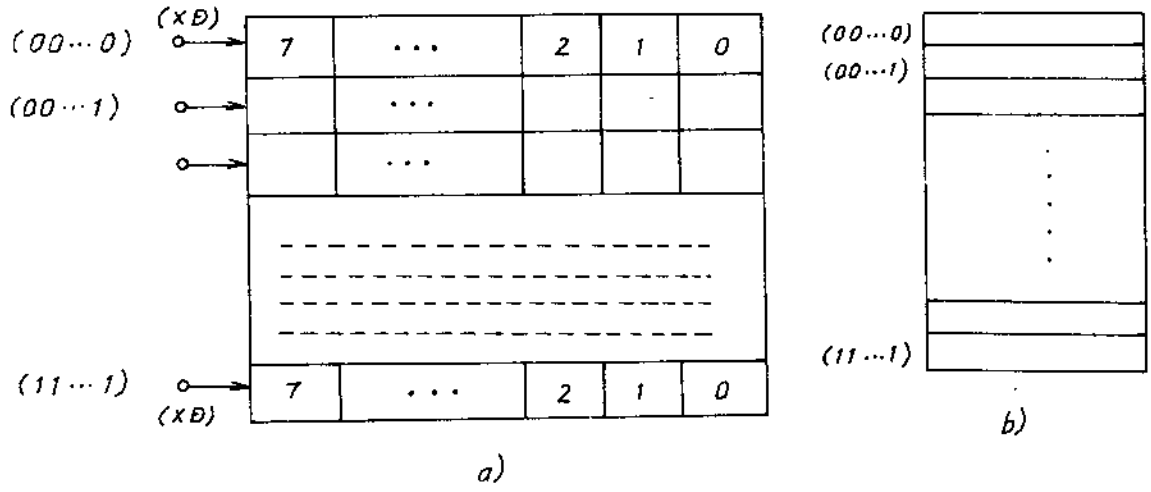


Hình 5.71. Thanh ghi 3 bit và ký hiệu.

Trên hình vẽ mô tả mạch điện của thanh ghi ba bit dùng trigơ JK (JKFF), chúng ta có thể dùng các loại trigơ khác (DFF, RSFF hay TFF) để xây dựng thanh ghi. Nếu ta muốn ghi giữ được số liệu lớn hơn hay xây dựng thanh ghi có số lượng bit lớn hơn chỉ cần thêm các trigơ.

Để ghi nhận các số liệu vào thanh ghi ta dùng các đầu kích (thông thường hay dùng các đầu thiết lập 1 và thiết lập 0) để ghi thông tin vào thanh ghi. Sau khi ghi xong thì nội dung số liệu được ghi giữ trong thanh ghi, muốn đọc nội dung đó ra thì ta dùng "xung đọc", mỗi khi có xung đọc thì nội dung của thanh ghi được đọc ra dưới dạng bit song song qua các mạch VÀ. Hoạt động của thanh ghi thường là ghi vào song song và lấy ra thông tin song song. Nhờ có "xung đọc" nên nội dung của thanh ghi có thể đọc đi đọc lại nhiều lần mà không bị mất thông tin sau khi đọc. Trên sơ đồ mạch điện ta đọc nội dung của thanh ghi trên các đầu ra y nhờ có các mạch VÀ kết hợp với xung đọc, cho chúng ta mã thuận của nội dung thanh ghi. Còn nếu chúng ta dùng các đầu ra $\bar{y}$ để lấy thông tin ra, thì sẽ thu được dưới dạng mã ngược của nội dung thông tin đã được ghi.

Nếu chúng ta sắp xếp thật nhiều thanh ghi tĩnh vào một khu vực, thì sẽ tạo ra bộ nhớ tĩnh (RAM) được chỉ ra ở trên hình 5.72.



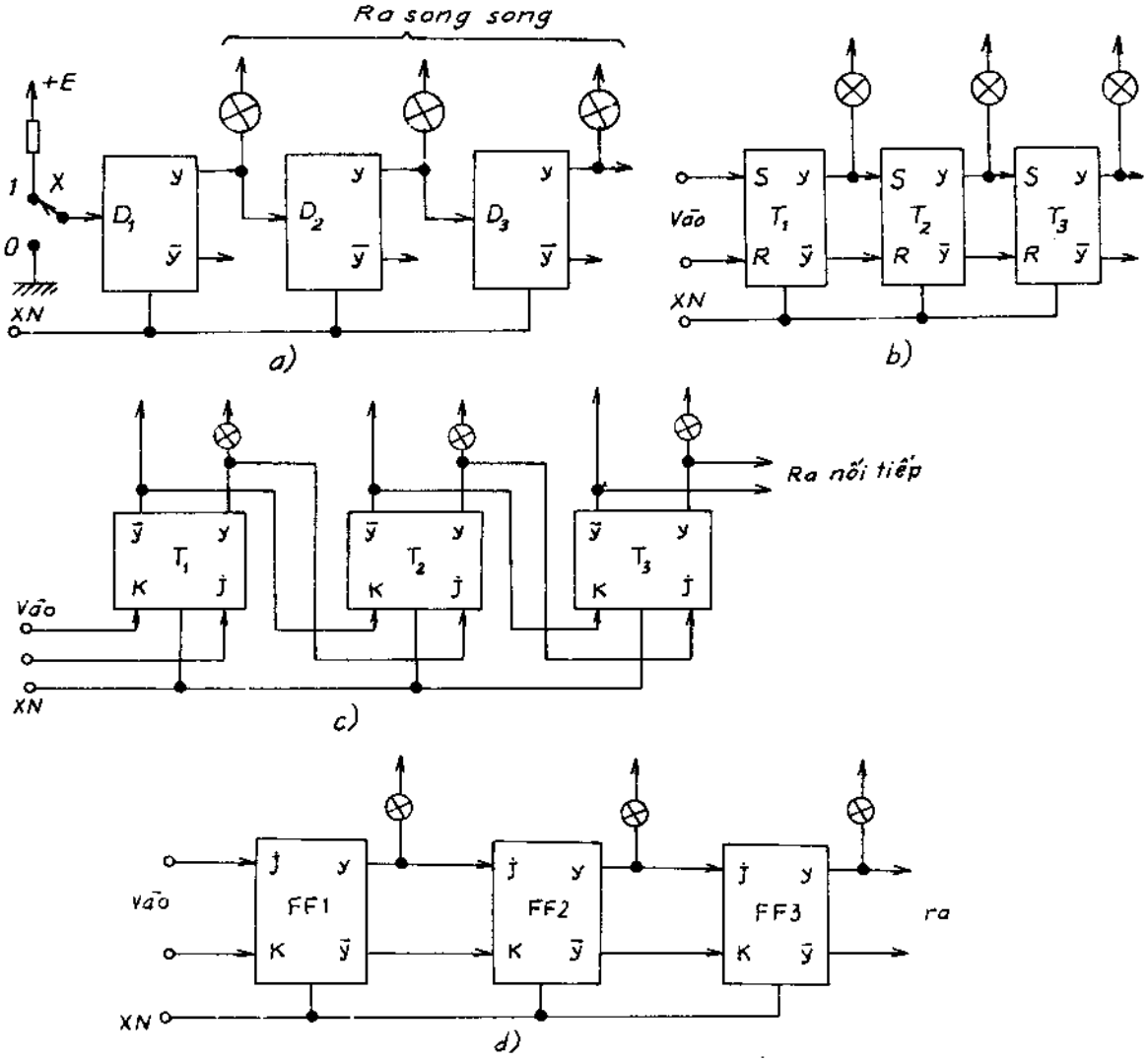
Hình 5.72. Bộ nhớ tĩnh của máy tính.

Thanh ghi tĩnh dùng để ghi giữ thông tin, nên bộ nhớ tĩnh cũng dùng để ghi giữ thông tin không thay đổi, như các hằng số, các địa chỉ, cũng như các chương trình điều hành (ví dụ : DOS) hay một chương trình phần mềm nào đó. Nội dung sẽ được đọc ra từ bộ nhớ nhờ xung đọc, đọc lần lượt từ (00...0) tới (11...1) sẽ tạo ra các tín hiệu điều khiển từ bộ nhớ, cũng như có thể đọc một ô địa chỉ nào đó nếu cần.

Thanh ghi tĩnh dùng để ghi giữ thông tin, nên ngoài việc ghi giữ thông tin nó còn được dùng làm các mạch chốt (latch) ghim thông tin lại, hay lưu giữ thông tin chờ một thời gian nếu cần. Ngoài ra còn dùng để lưu giữ một thông tin cần thiết cho quá trình nhận dạng như dạng vân tay, khuôn mặt, hình ảnh cũng như âm thanh...

5.6.2. THANH GHI ĐỘNG (BỘ GHI DỊCH)

Thanh ghi động còn được gọi là thanh ghi dịch, dùng để ghi giữ thông tin đồng thời thông tin đó có thể được dịch sang phải hoặc dịch sang trái nếu cần. Thanh ghi động cũng như thanh ghi tĩnh có thể được xây dựng và thực hiện bằng các loại trigơ khác nhau. Bộ ghi dịch phải 3 bit dùng một vài loại trigơ khác nhau được mô tả trên hình 5.73 như sau.



Hình 5.73. Bộ ghi dịch phải ba bit.

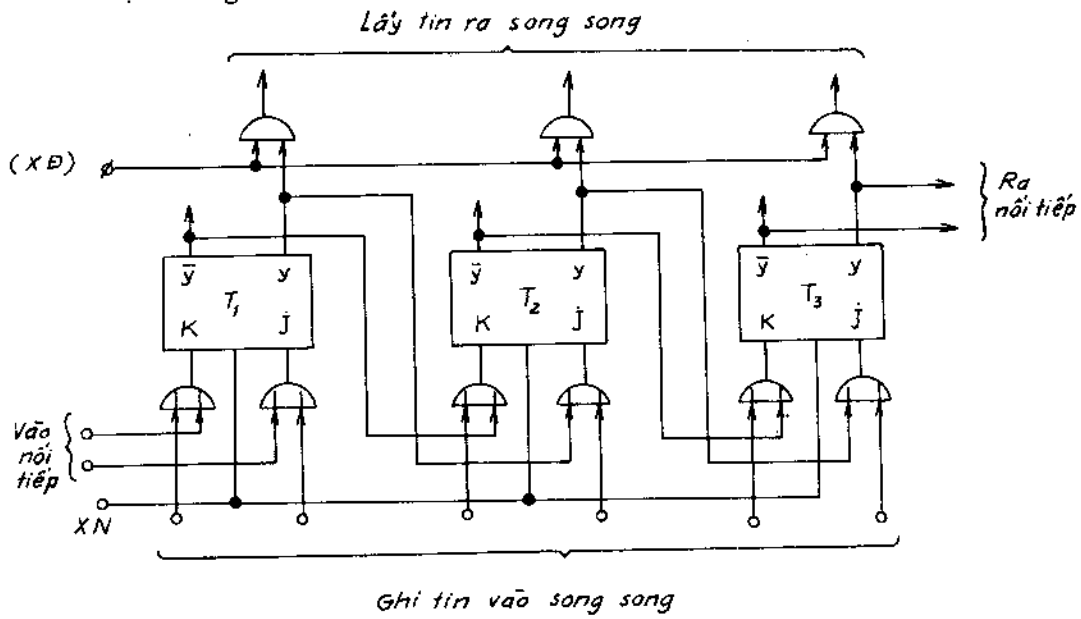
Bộ ghi dịch có thể dùng vài loại trigơ khác nhau để xây dựng, muốn có số lượng bit nhiều hơn ta có thể góp thêm các trigơ lại. Tùy theo cấu tạo của các trigơ sử dụng ta có thể sử dụng xung nhịp (XN) hay không đều được, và chỉ cần nhìn vào chiều nối dây từ đầu ra trigơ này kích vào đầu vào của trigơ tiếp theo là chúng ta có thể hiểu ngay đó là bộ ghi dịch phải hay ghi dịch trái. Mỗi một trigơ trong bộ ghi dịch điều khiển một đèn chỉ thị (LED), nếu chúng ta có một bảng đèn hay một ma trận điểm sáng, thì

ta có thể thực hiện được bảng đèn chữ chạy (chữ là ký hiệu ghi trước, còn chạy chính là dịch nó đi), cũng như dùng bộ ghi dịch làm các bảng quảng cáo, hay làm pháo hoa điện tử v.v...

Ứng dụng của bộ ghi dịch trong thực tế là rất phong phú, nhưng tổng quát lại khi có bộ ghi dịch chúng ta có những chức năng chính sau :

- Ghi vào nối tiếp → lấy ra nối tiếp
- Ghi vào nối tiếp → lấy ra song song
- Ghi vào song song → lấy ra nối tiếp
- Ghi vào song song → lấy ra song song.

Để thỏa mãn các chức năng tổng quát đó của bộ ghi dịch, ta vừa phải ghi số liệu vào vừa phải có tín hiệu dịch từ trigơ bên cạnh sang, nên chúng ta phải có mạch lôgic để kết hợp được các tín hiệu đó khi kích vào cực kích của trigơ. Sự kết hợp các tín hiệu được minh họa đơn giản như sau ở trên hình 5.74.



Hình 5.74. Bộ ghi dịch phải 3 bit tổng quát.

Ví dụ : Xây dựng bộ ghi dịch phải 3 bit dùng trigơ JK (JKFF) có khả năng ghi vào song song như sau.

Ngoài các chức năng chính của bộ ghi dịch như đã nêu bộ ghi dịch còn có thêm các chức năng phụ như xóa, đồng bộ theo xung nhịp (XN), vừa dịch sang phải vừa có thể dịch sang trái bằng các tín hiệu cho phép dịch phải hay cho phép dịch trái v.v... Chúng ta chỉ cần dùng thêm các mạch tổ hợp (mạch Và (AND), mạch hoặc (OR)) kết hợp các tín hiệu đó vào là xong. Trong thực tế người ta đã xây dựng bộ ghi dịch tổng quát bằng các IC chuẩn có sẵn, sơ đồ nguyên lý của chúng đều được vẽ và chỉ ra cụ thể trong các quyển sổ tay tra cứu IC (ví dụ : bộ ghi dịch tổng quát 4 bit là con IC SN 7495). Bộ ghi

dịch hay thanh ghi dịch được sản xuất hay chế tạo rất nhiều loại khác nhau, tùy theo nhu cầu cụ thể chúng ta có thể tự thiết kế xây dựng hay tìm các IC đã có sẵn mà sử dụng.

Một số vi mạch (IC) ghi dịch thông dụng.

- Bộ ghi dịch 4 bit, vào song song hoặc nối tiếp dùng JKFF, có đầu xóa :

  54/74 195 ; 54/74 S195.

- Bộ ghi dịch 4 bit, vào song song hoặc nối tiếp dùng JKFF, không có đầu vào xóa :

  54/74 L99.

- Bộ ghi dịch 8 bit vào nối tiếp, ra nối tiếp :

  54/74 91A ; 54/74 LS91.

- Bộ ghi dịch 5 bit, ghi vào song song hoặc nối tiếp, có đầu xóa :

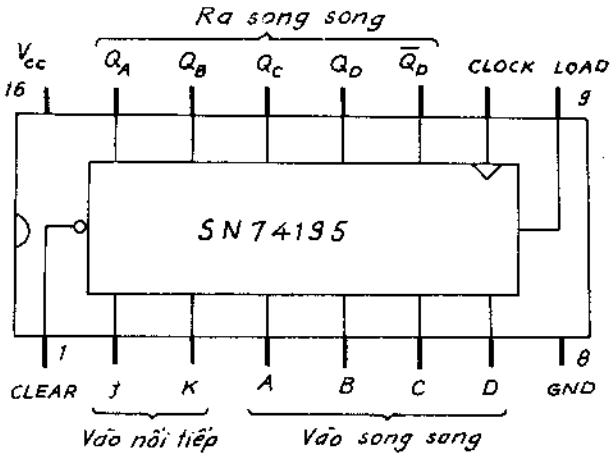
  54/74 96 ; S4/74 LS96.

- Bộ ghi dịch 8 bit, ghi vào nối tiếp, ra song song, dùng RSFF, có đầu xóa :

  54/74 164 ; 54/74 LS 164.

- Bộ ghi dịch 8 bit, vào song song hoặc nối tiếp, ra song song, dùng JKFF, có đầu xóa : 54/74 199.

Ví dụ minh họa cấu tạo IC 74195 là bộ ghi dịch 4 bit được chỉ ra trên hình 3.75.



Hình 5.75. Minh họa một IC là bộ ghi dịch.

Từ thanh ghi động hay bộ ghi dịch, chúng ta có thể sắp xếp chúng lại để xây dựng nên bộ nhớ động, có được bộ nhớ động và chúng ta quản lý được nó, đọc thông tin ra từ bộ nhớ động rồi đưa lên màn hình của máy vi tính, sẽ tạo ra được các hình ảnh "động" hay chuyển dịch trên màn hình theo ý muốn của ta. Đó chính là cơ sở để tạo nên các trò chơi điện tử, cũng như các kỹ xảo hình ảnh, các hình ảnh sinh động trên TV cũng như trên màn hình của máy vi tính v.v...

5.6.3. BỘ ĐẾM

Bộ đếm là một ôtômat có nhớ điển hình, cho chúng ta biết rất rõ trạng thái tiếp theo phụ thuộc vào trạng thái trước đó dưới tác động của tín hiệu vào như thế nào. Quan hệ đó cũng chính là cơ sở để chúng ta có thể thiết kế ra các bộ đếm khác nhau với các cơ số đếm (Kđ) bất kỳ. Cũng giống như ôtômat có nhớ muốn xây dựng được thì phải dùng tới các phần tử nhớ (các trigơ) để có được bộ đếm chúng ta cũng cần đếm các trigơ. Nhưng ở đây để xây dựng nên bộ đếm (Kđ = 2^n) chúng ta dùng đầu đảo (đầu vào đếm) để thực hiện.

Bộ đếm là một mạch điện rất quan trọng trong hệ thống số, chúng được dùng để đếm thời gian dùng trong hẹn giờ, dùng chia tần hay phân tần, dùng điều khiển các mạch điện khác như bộ giải mã v.v... Bộ đếm có thể đếm tăng dần hay đếm cộng, cũng như có thể đếm giảm dần hay gọi là đếm trừ, có thể đếm thuận nghịch vừa tăng và vừa giảm, có thể đếm theo các quy ước mã khác nhau như mã Gray (mã bìa Kácnô), mã vòng, mã BCD v.v...

Trong phần này chúng ta sẽ nói tới tất cả các phương pháp xây dựng bộ đếm cũng như nguyên lý làm việc của các bộ đếm. Để hiểu rõ chúng ta sẽ lần lượt xét tới các bộ đếm sau, các bộ đếm chúng chủ yếu là các bộ đếm vòng quanh sau một số lần đếm nhất định trạng thái của bộ đếm lại trở lại trạng thái ban đầu. Bộ đếm là một ôtômat có nhớ với một đầu vào x (0 hoặc 1) và một đầu ra Z, số trạng thái của bộ đếm chính bằng cơ số đếm (Kđ) của nó. Tiếp theo chúng ta xét lần lượt các loại bộ đếm khác nhau.

5.6.3.1. Bộ đếm nhị phân tăng dần - bộ đếm cộng

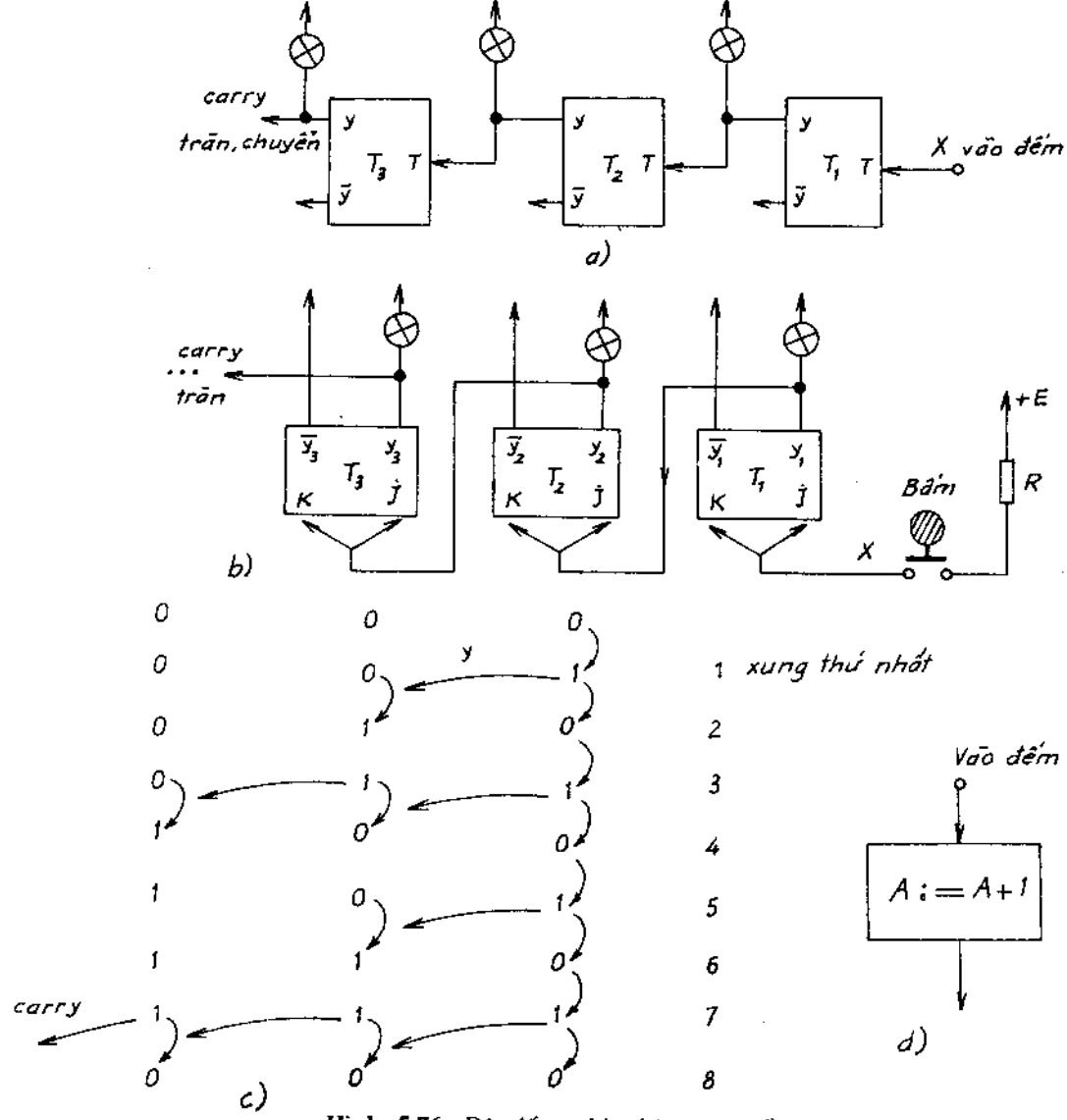
Bộ đếm nhị phân (BCD) tăng dần dùng đầu vào đếm (đầu đảo) tổ hợp ta dùng đầu ra y. Trạng thái trong của bộ đếm thay đổi lần lượt theo quy luật của mã nhị phân tăng dần, tổ hợp nên bộ đếm chúng ta chỉ dùng đầu ra thuận y của các trigơ kết hợp với đầu vào đếm. Sơ đồ của bộ đếm dùng 3 trigơ JKFF (TFF) có cơ số đếm Kđ = 8 (Kđ = 2^n) được chỉ ra trên hình 5.76.

Trước hết chúng ta có quy ước là nếu tổ hợp :

- Dùng đầu ra y thì trigơ chuyển từ (1 → 0) sẽ cho ra xung chuyển sang cột bên cạnh, có tác dụng kích lật được trigơ đó.

- Dùng đầu ra $\bar{y}$ thì khi trigơ chuyển từ (0 → 1) sẽ có xung chuyển có tác dụng kích sang trigơ bên cạnh.

Với quy ước như vậy khi nối thực tế mạch điện chúng ta phải nối sao cho thỏa mãn. Quy ước đó cũng xuất phát từ chiều của điôt ở các đầu vào kích của trigơ, mà chiều của điôt kích ($Đ_1$, $Đ_2$)) đã được chỉ ra cụ thể trong sơ đồ mạch điện của trigơ RS ở hình 5.9b, chiều của điôt ($Đ_1$, $Đ_2$) sẽ quyết định trigơ được kích bằng xung âm (sườn đột biến âm) hay kích bằng xung dương (ứng với sườn xung kích có đột biến dương). Nguyên lý hoạt động của mạch trigơ đã được nói rất kỹ trong giáo trình kỹ thuật xung mà ở đây chúng ta không đi sâu. Từ những kiến thức cơ bản đó chúng ta phải nối mạch điện cụ thể thỏa mãn được quy ước nêu trên.



Hình 5.76. Bộ đếm nhị phân tăng dần.

Từ quy ước và sơ đồ mạch điện trên hình 5.76,b chúng ta sẽ nêu được nguyên lý làm việc của bộ đếm như chỉ ra trên hình 5.76,c.

Trạng thái ban đầu của bộ đếm (0 0 0), khi chúng ta "bấm" hay cho xung thứ nhất, xung đó được đưa vào đầu vào đảo, làm cho T_1 chuyển từ (0 → 1), do tổ hợp dùng đầu ra y nên không có xung chuyển sang hàng bên cạnh, vì thế trigơ T_2 vẫn ở 0, cũng như trigơ T_3 càng vẫn ở 0. Sau khi cho xung thứ nhất trạng thái của bộ đếm là (0 0 1).

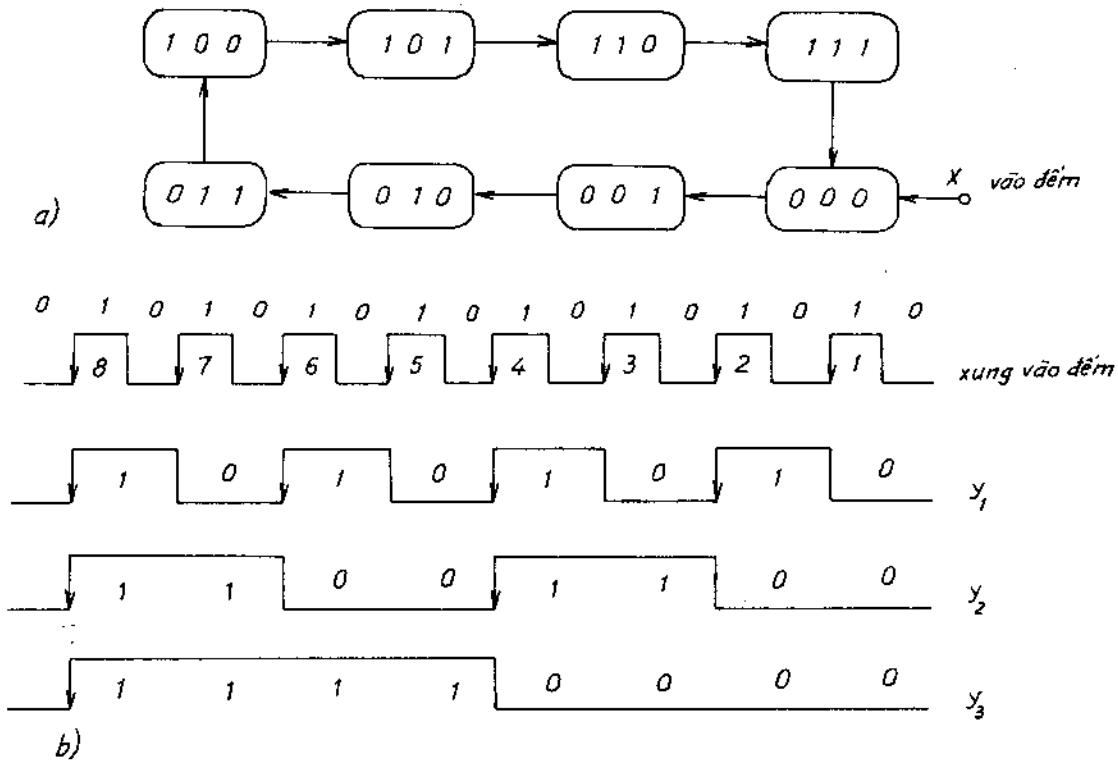
Ta cho xung thứ hai làm T_1 chuyển (1 → 0), do tổ hợp dùng đầu y nên xuất hiện xung chuyển làm cho T_2 lật từ (0 → 1), khi T_2 chuyển từ (0 → 1) sẽ không có xung chuyển sang cột bên cạnh nên T_3 vẫn ở 0. Sau xung thứ 2 trạng thái của bộ đếm sẽ là (0 1 0).

Ta cho tiếp xung thứ 3 làm cho T_1 chuyển (0 → 1), không có xung chuyển sang cột bên cạnh, nên trigơ T_2 vẫn ở 1, còn T_3 vẫn ở 0. Sau xung thứ 3 trạng thái của bộ đếm lúc này là (0 1 1).

Cho tiếp xung thứ 4 làm cho T_1 (1 → 0), cho ra xung chuyển làm T_2 (1 → 0), lại cho ra xung chuyển làm T_3 (0 → 1). Sau xung thứ 4 trạng thái trong của bộ đếm là (1 0 0).

Với cách làm tương tự chúng ta tiếp tục lần lượt cho các xung đếm vào, thì sau 8 xung bộ đếm lại trở lại trạng thái ban đầu là (000) vì vậy cho Kđ = 8, đồng thời cho ra một xung chuyển (carry) kích sang cột tiếp theo. Đây là bộ đếm vòng quanh có số đếm là 8, và trạng thái trong của bộ đếm biến đổi theo quy luật nhị phân tăng dần.

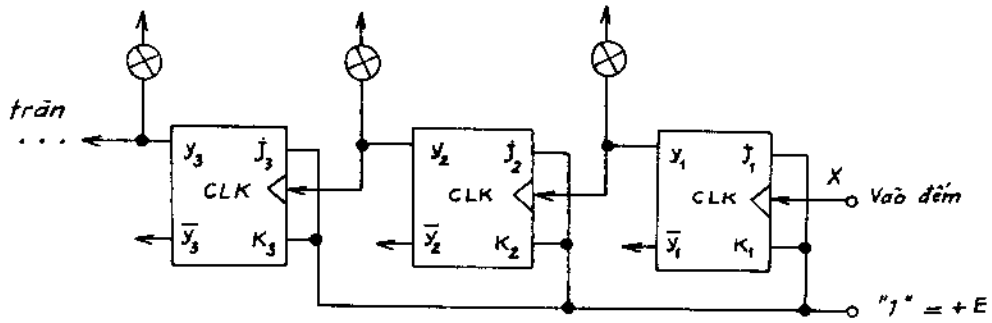
Qua nguyên lý làm việc của bộ đếm chúng ta thấy các mũi tên (↶) đó chính là nguyên lý lật chuyển và hoạt động của bộ đếm. Qua nguyên lý làm việc của bộ đếm chúng ta có thể nói rằng : bộ đếm lật chuyển trạng thái của các trigơ theo nguyên lý lật chuyển kiểu "đôminô", tức là trigơ này lật chuyển sẽ tạo ra xung tràn, xung chuyển (carry) làm lật chuyển sang trigơ ở cột tiếp theo. Đồ hình trạng thái và biểu đồ thời gian của bộ đếm 8 với ba trigơ được mô tả trên hình 5.77.



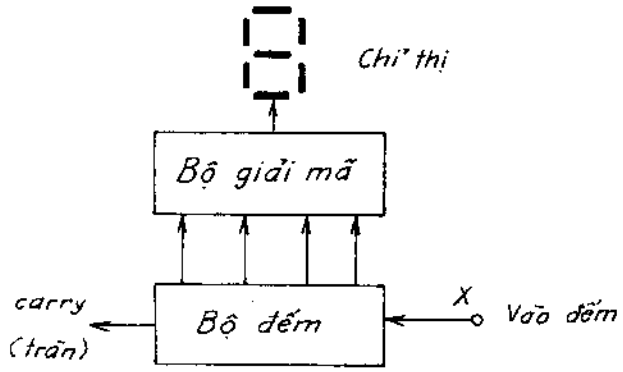
Hình 5.77. Đồ hình trạng thái của bộ đếm.

Bộ đếm nhị phân tăng tổ hợp chủ yếu dùng đầu ra y, nhưng người ta vẫn có thể tổ hợp bộ đếm nhị phân tăng theo "cách khác" điều đó được mô tả trên hình 5.78.

Bộ đếm nhị phân tăng dần có trạng thái trong biến đổi theo mã nhị phân (BCD), cho nên đầu ra của bộ đếm chúng ta có thể nối vào đầu vào bộ giải mã (thông thường là giải mã nhánh), từ đó tạo ra một cấu trúc chỉ thị số trên một cột số như trên hình 5.79.



Hình 5.78. Bộ đếm nhị phân tăng.



Hình 5.79. Một cột số chỉ thị số.

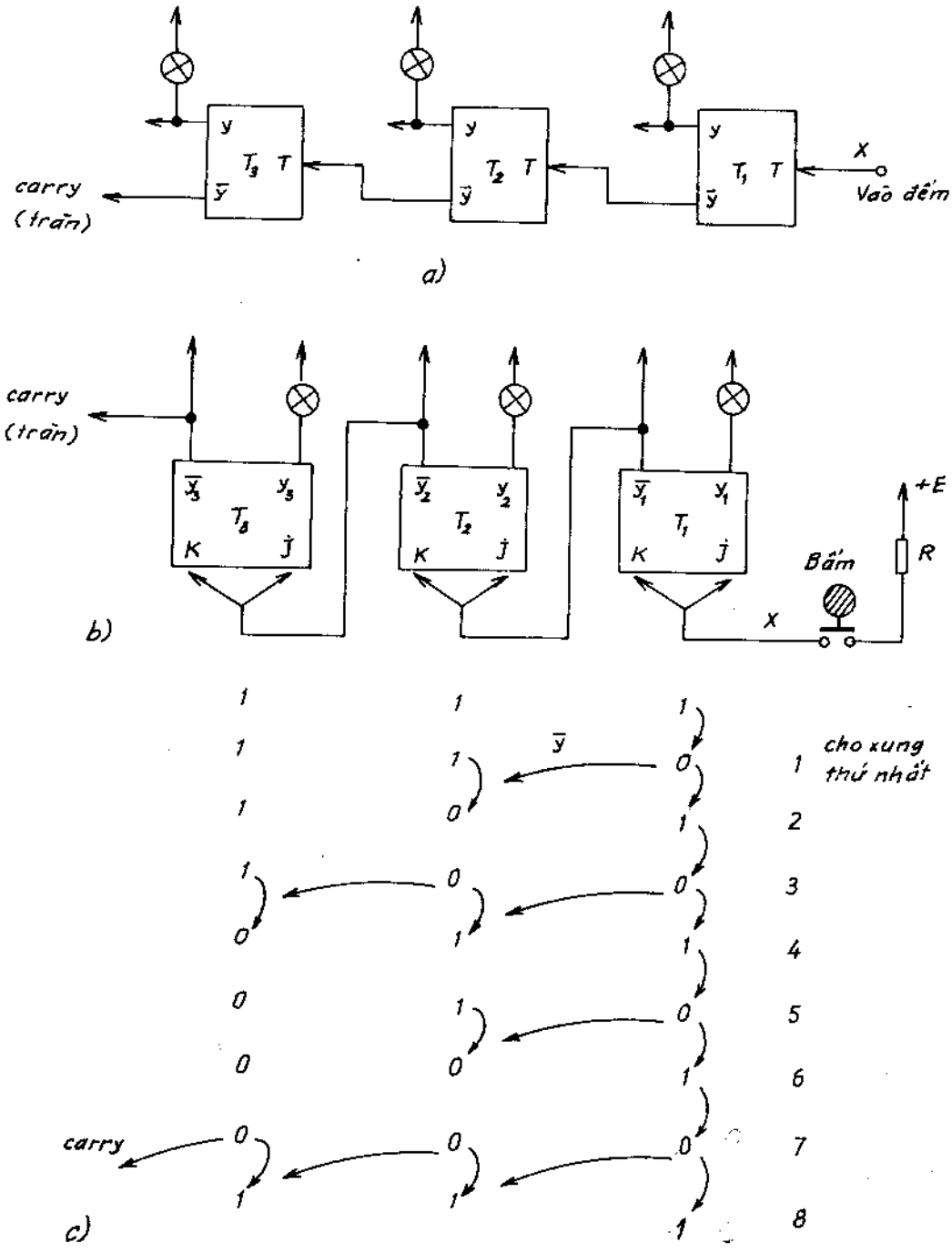
5.6.3.2. Bộ đếm nhị phân giảm dần – bộ đếm trừ

Bộ đếm nhị phân (BCD) giảm dần tổ hợp dùng đầu ra $\bar{y}$. Trạng thái của bộ đếm sẽ thay đổi lần lượt theo quy luật của mã nhị phân nhưng có kết quả giảm dần. Sơ đồ của bộ đếm ngược (đếm trừ) dùng ba trigơ JK (có cơ số đếm Kđ = 8) được chỉ ra ở trên hình 5.80.

Khi tổ hợp dùng đầu ra $\bar{y}$, thì khi trigơ lật chuyển từ (0 → 1), ở đầu ra $\bar{y}$ sẽ có một xung chuyển (carry) sang cột (trigơ) bên cạnh, xung đó có khả năng lật chuyển trạng thái của trigơ. Sơ đồ nguyên lý của bộ đếm trình bày trên hình 5.80,b.

Trạng thái ban đầu của bộ đếm là (1 1 1), khi chúng ta "bấm" tức là cho vào bộ đếm xung thứ nhất, xung đó được đưa vào đầu vào đảo của trigơ T_1 làm cho T_1 chuyển từ (1 → 0), do tổ hợp dùng đầu ra $\bar{y}$ nên quá trình lật chuyển đó của T_1 không có xung chuyển sang trigơ (cột) bên cạnh, vì thế trigơ T_2 vẫn ở 1, cũng như trigơ T_3 lại càng vẫn ở 1. Như vậy sau khi ta cho xung thứ nhất vào trạng thái của bộ đếm trừ sẽ là (1 1 0).

Ta tiếp tục cho xung thứ 2 làm cho T_1 chuyển (0 → 1) do dùng đầu ra $\bar{y}$ nên quá trình lật chuyển đó của T_1 tạo ra xung chuyển (carry) làm cho T_2 lật chuyển (1 → 0), khi T_2 chuyển từ (1 → 0) sẽ không có xung chuyển sang cột bên cạnh nên trigơ T_3 vẫn ở 1. Như vậy sau 2 xung vào (sau khi cho xung thứ 2 vào) trạng thái của bộ đếm sẽ là (1 0 1).



Hình 5.80. Bộ đếm nhị phân giảm.

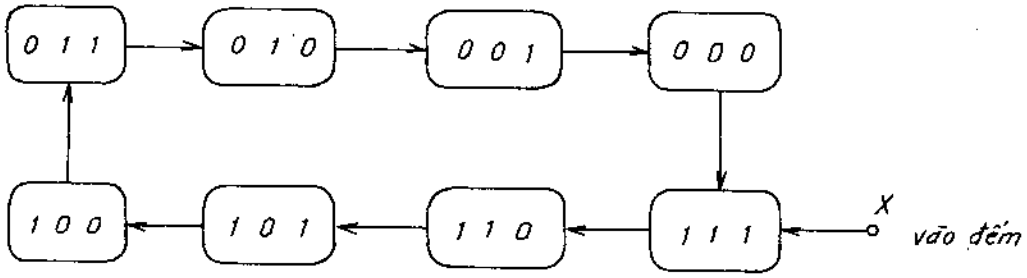
Ta cho tiếp xung thứ 3 làm cho T_1 chuyển (1 → 0), không có xung chuyển sang trigơ bên cạnh, nên trigơ T_2 vẫn ở 0 còn trigơ T_3 vẫn ở 1. Sau xung thứ 3 trạng thái của bộ đếm sẽ là (1 0 0).

Cho tiếp xung (trừ) thứ tư, làm cho T_1 chuyển (0 → 1) đầu ra $\bar{y}_1$, cho ra một xung

chuyển làm cho T_2 chuyển (0 → 1), lại cho ra xung chuyển làm cho T_3 chuyển từ (1 → 0). Sau 4 xung vào trạng thái của bộ đếm sẽ là (0 1 1).

Với cách làm tương tự chúng ta tiếp tục cho lần lượt các xung đếm vào, thì sau 8 xung bộ đếm sẽ lại trở lại trạng thái ban đầu là (1 1 1) vậy cho ta có cơ số đếm Kđ = 8, đồng thời ở đầu ra $\bar{y}_3$ tạo ra một xung chuyển (tràn) sang cột tiếp theo. Đây là bộ đếm trừ, vì các trạng thái của bộ đếm sau khi ta cho xung đếm vào sẽ giảm dần giá trị theo quy luật của mã nhị phân. Đồng thời đây cũng là bộ đếm trừ đếm vòng quanh. Do trạng thái của bộ đếm giảm dần theo quy luật của mã nhị phân nên đầu ra của bộ đếm có thể nối với bộ giải mã để đưa ra chỉ thị số.

Đồ hình trạng thái của bộ đếm ngược (trừ) được mô tả trên hình 5.81.



Hình 5.81. Đồ hình trạng thái bộ đếm trừ.

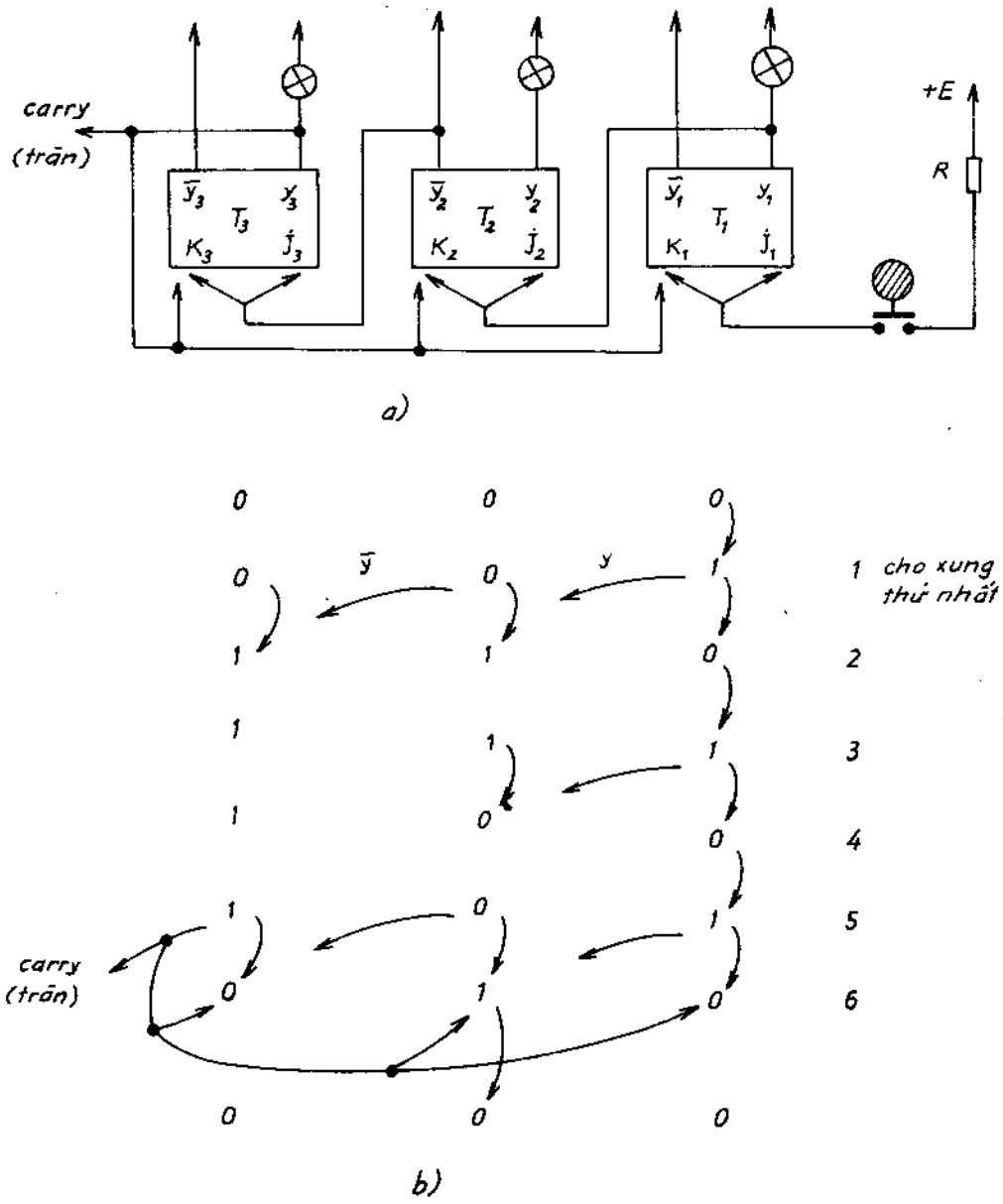
5.6.3.3. Bộ đếm tổ hợp kết hợp đầu ra y và $\bar{y}$ cho Kđ (cơ số đếm) bất kỳ

Giả sử dụng ta có bộ đếm trong đó tổ hợp chung cả đầu ra y và $\bar{y}$ như chỉ ra ở trên hình 5.82.

Nguyên lý làm việc của bộ đếm tổ hợp chung đầu ra y và $\bar{y}$ dựa trên điều kiện : Tổ hợp dùng đầu ra y thì khi trigơ lật chuyển từ (1 → 0) thì đầu ra y sẽ xuất hiện một xung chuyển (carry) sang cột bên cạnh, còn nếu ta dùng đầu ra $\bar{y}$ thì khi trigơ chuyển từ (0 → 1) sẽ có xung chuyển kích sang trigơ bên cạnh.

Nguyên lý làm việc của bộ đếm này được chỉ ra trên hình 5.82,b. Trạng thái ban đầu của bộ đếm là (0 0 0), khi chúng ta cho xung thứ nhất vào (bấm một xung) làm cho trigơ T_1 chuyển từ (0 → 1), do tổ hợp dùng đầu ra y nên quá trình lật chuyển trạng thái đó của T_1 không có xung chuyển sang trigơ bên cạnh, vì thế trigơ T_2 vẫn ở trạng thái 0, cũng như trigơ T_3 vẫn ở 0. Như vậy sau khi chúng ta cho xung thứ nhất vào trạng thái của bộ đếm là (0 0 1).

Tiếp theo chúng ta cho xung vào thứ 2 làm cho trigơ T_1 chuyển từ (1 → 0), do tổ hợp dùng đầu ra y nên quá trình lật chuyển đó của T_1 sẽ cho ra xung chuyển kích sang trigơ bên cạnh làm cho trigơ T_2 chuyển từ (0 → 1), do trigơ T_2 tổ hợp dùng đầu ra $\bar{y}$ nên quá trình lật chuyển đó của T_2 sẽ cho ra một xung chuyển kích tiếp theo làm cho trigơ T_3 chuyển từ (0 → 1). Như vậy sau khi kết thúc xung kích thứ 2 trạng thái của bộ đếm sẽ là (1 1 0).



Hình 5.82. Bộ đếm tổ hợp kết hợp y và $\bar{y}$.

Cho tiếp xung kích thứ 3 vào làm cho T_1 chuyển từ (0 → 1), không có xung chuyển sang cột bên cạnh nên trạng thái của T_2 vẫn ở 1, còn với trigơ T_3 vẫn ở 1. Như vậy sau khi cho xung thứ 3 trạng thái của bộ đếm sẽ là (1 1 1).

Ta cho tiếp xung thứ 4 làm cho T_1 chuyển từ (1 → 0), sẽ cho ra xung chuyển làm cho T_2 sẽ chuyển tử (1 → 0), không có xung chuyển sang cột bên cạnh nên T_3 vẫn ở 1. Như vậy sau xung thứ 4 trạng thái của bộ đếm là (1 0 0).

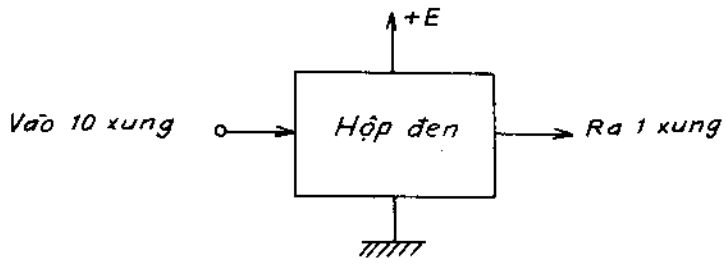
Cho tiếp xung thứ 5 vào làm cho trạng thái của trigơ T_1 chuyển từ (0 → 1), sẽ

không có xung chuyển sang cột tiếp theo nên trigơ T_2 vẫn ở 0, còn trigơ T_3 vẫn ở 1. Vậy sau 5 xung trạng thái của bộ đếm sẽ là (1 0 1).

Khi chúng ta cho xung thứ 6 vào làm cho trigơ T_1 chuyển từ (1 → 0), sẽ cho ra xung làm cho trigơ T_2 chuyển từ (0 → 1), lại cho ra xung chuyển tiếp kích sang cột bên cạnh làm cho T_3 chuyển từ (1 → 0), lại tạo ra xung chuyển tiếp (xung carry) kích chuyển sang cột tiếp theo, nhưng xung chuyển từ đầu ra y_3 của trigơ T_3 lại được đấu nối đưa về kích vào các đầu thiết lập 0 (đầu kích K) của các trigơ làm cho T_1 ở 0 lại xóa 0 nên T_1 vẫn ở 0, còn trigơ T_2 đang ở 1 kích vào K_2 xóa nó đi làm cho T_2 chuyển từ (1 → 0), quá trình chuyển trạng thái này không tạo ra xung chuyển nên T_2 trở về 0, với trigơ T_3 đang ở 0 lại kích xung xóa vào K_3 nên nó vẫn ở 0. Đây là quá trình chuyển biến trạng thái của các trigơ trong bộ đếm xảy ra do có vòng hội tiếp về "xóa" các trạng thái của các trigơ. Như vậy ta thấy sau khi kích xung thứ 6 thì trạng thái của bộ đếm là (0 0 0) trở lại trạng thái ban đầu.

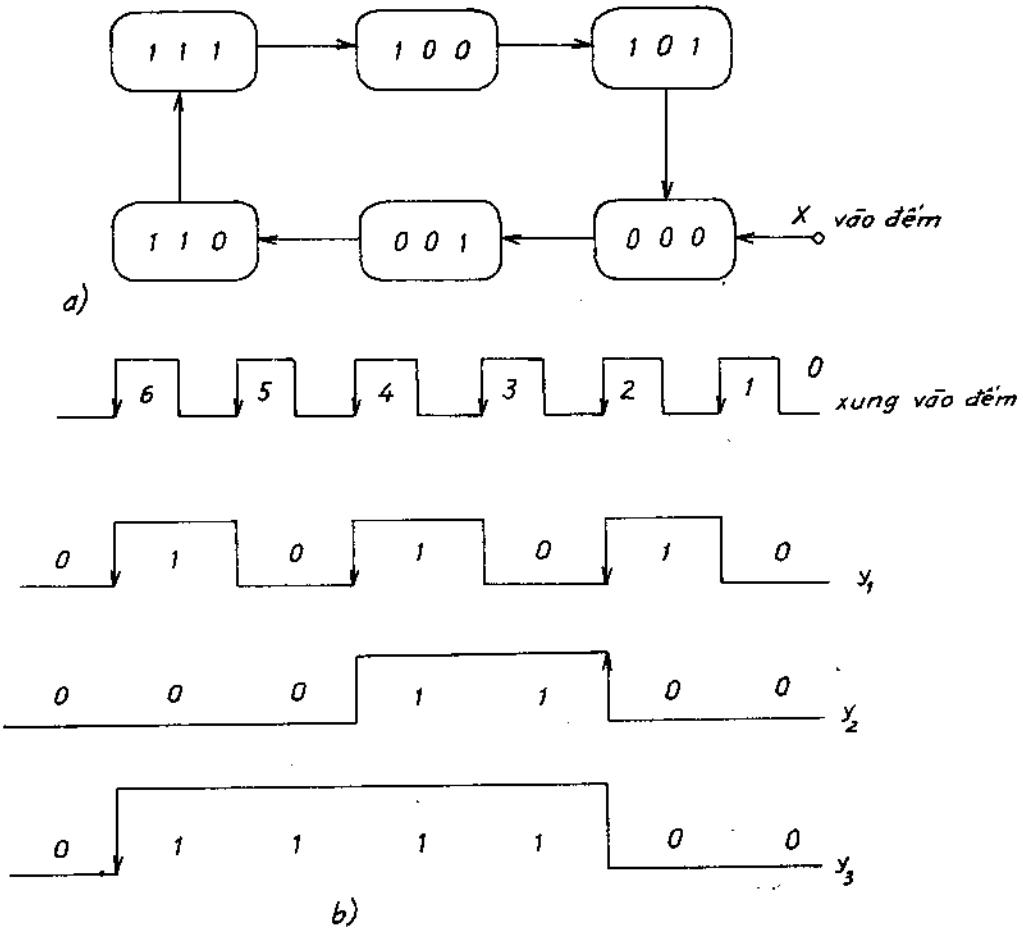
Sau khi xem xét nguyên lý làm việc của bộ đếm tổ hợp lẫn lộn đầu ra y và $\bar{y}$, chúng ta thấy với cách làm đó sẽ cho chúng ta tạo ra được bộ đếm có cơ số đếm bất kỳ không theo quy luật 2^n. Tạo được bộ đếm có cơ số đếm bất kỳ là rất quan trọng trong ứng dụng của hệ thống số đối với cuộc sống, vì không phải lúc nào chúng ta cũng cần bộ đếm có cơ số đếm nhị phân (Kđ = 2^n), mà chúng ta cần có các bộ đếm có các cơ đếm khác nhau như bộ đếm cơ số đếm Kđ = 5 (năm ngón tay), cần có cơ số đếm Kđ = 6 rồi Kđ = 10 để tạo ra cơ số đếm 60 (Kđ = 60) dùng trong đồng hồ điện tử chỉ thị số, nhất là với cơ số đếm 10 (Kđ = 10) rất quen thuộc và cần thiết trong cuộc sống của chúng ta.

Ta thấy với bộ đếm tổ hợp hỗn hợp đầu ra y và $\bar{y}$ cho chúng ta Kđ bất kỳ, nhưng bộ đếm kiểu này nó có một nhược điểm rất lớn là : trạng thái bên trong của các trigơ của bộ đếm không thay đổi quy luật của mã nhị phân thông thường (mã BCD), cho nên đầu ra của bộ đếm không thể phối ghép trực tiếp được với bộ giải mã và không thể chỉ thị số được. Chính vì vậy trong thực tế nếu chúng ta chỉ quan tâm tới kết quả đếm mà không quan tâm tới chỉ thị trạng thái nội bộ của các trigơ, thì người ta vẫn sản xuất các con IC đếm như một cái hộp đen dùng để phân chia tần số, ví dụ : cho vào 6 xung đầu ra cho ra 1 xung vậy có Kđ = 6, hay cho vào 10 xung đầu ra cho ra 1 xung, đó là bộ chia tần có Kđ = 10 (hệ số phân tần). Loại bộ đếm đó được minh họa trên hình 5.83.



Hình 5.83. Bộ đếm Kđ bất kỳ.

Bộ đếm có cơ số đếm bất kỳ tổ hợp hỗn hợp đầu ra y và $\bar{y}$ mô tả trên hình 5.84 có biểu đồ thời gian và đồ hình trạng thái mô tả trên hình 5.84.



Hình 5.84. Đồ hình trạng thái và biểu đồ thời gian.

Phương pháp tổ hợp hỗn hợp đầu ra y và $\bar{y}$ của các trigơ cho chúng ta bộ đếm với cơ số đếm bất kỳ. Trong trường hợp tổ hợp hay thực hiện bộ đếm có cơ số đếm (Kđ) bất kỳ cho trước tổ hợp dùng đầu ra y và $\bar{y}$ dựa theo công thức :

2^n = Kđ = con số = số nhị phân → nếu = 0 nối y
→ nếu = 1 nối $\bar{y}$

Trong đó : n : là số trigơ cần dùng.

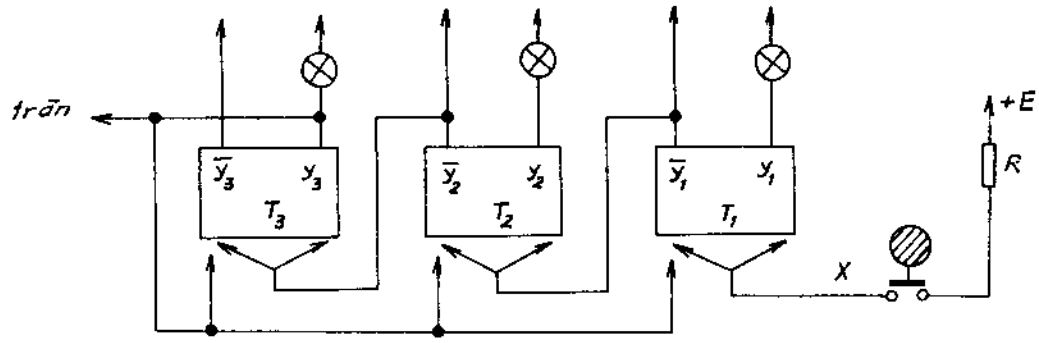
Kđ : là cơ số đếm cho trước.

Ví dụ : Cho trước Kđ = 5.

Do Kđ = 5 nên ta cần 3 trigơ, áp dụng công thức

$$2^3 - 5 = 3 = 0\ 1\ 1 \rightarrow \text{tổ hợp } (y_3\ \bar{y}_2\ \bar{y}_1)$$

Mạch điện của bộ đếm 5 (Kđ = 5) cho trước cơ số đếm, tổ hợp dùng đầu ra y và $\bar{y}$ được mô tả trên hình 5.85.



Hình 5.85. Bộ đếm cơ số đếm 5 tổ hợp y và $\bar{y}$.

Sau khi ta thu được bộ đếm cho cơ số đếm bất kỳ tổ hợp dùng đầu ra y và $\bar{y}$, nhưng trạng thái trong của bộ đếm không theo quy luật của số nhị phân hay mã nhị phân (Binary Codes Decimal) thông thường, nên không nối được với bộ giải mã và không chỉ thị số được.

5.6.3.4. Bộ đếm với cơ số đếm bất kỳ biến đổi theo quy luật mã nhị phân BCD

Cách thiết kế hay tổng hợp loại bộ đếm kiểu này giống như cách thiết kế bộ đếm 5 cứ đếm đến 3 thì báo sáng ở đầu ra Z, dùng trigơ D (DFF). Đó là bộ đếm có cơ số đếm bất kỳ nhưng trạng thái bên trong của nó biến đổi theo quy luật của mã nhị phân thông thường. Nhưng vẫn bộ đếm có cơ số đếm bất kỳ trạng thái biến đổi theo quy luật nhị phân dùng loại trigơ khác (JKFF hay RSFF) thì cách thiết kế cụ thể thế nào sẽ được minh họa cụ thể ở ví dụ sau. Cách thiết kế giống như ở ôtômat thông thường.

Ví dụ : Thực hiện bộ đếm 5 (Kđ = 5) có trạng thái trong biến đổi theo nhị phân thông thường, mà cứ đếm đến 3 thì báo sáng trên đầu ra Z, dùng trigơ JK (JKFF) thực hiện theo mô tả của ôtômat Moore.

Theo đúng lý luận hay phương pháp thiết kế một ôtômat thông thường trước tiên chúng ta phải lập bảng trạng thái mô tả hoạt động của bộ đếm này (Kđ = 5). Bảng trạng thái của bộ đếm 5 dùng ôtômat Moore mô tả được chỉ ra trên hình 5.86.

S \ x	0	1	Z
0	0	1	0
1	1	2	0
2	2	3	0
3	3	4	1
4	4	0	0

Hình 5.86. Bảng chức năng của bộ đếm 5.

Bước tiếp theo chúng ta phải thực hiện rút gọn ôtômat có nhớ dựa vào bảng trạng thái hoạt động của nó. Ở đây do bảng chức năng là hoạt động của một bộ đếm nên không thể rút gọn được vì không có phân hoạch thế không tồn tại PHT , mà không có không tìm được PHT thì không có phân hoạch thế tương đương (dù có tồn tại các phân hoạch tương đương), khi không có phân hoạch thế tương đương (không tồn tại PHTTĐ) thì đó chính là ôtômat tối thiểu rồi và không thể rút gọn hơn được nữa. Như vậy bảng chức năng của bộ đếm 5 trên hình 5.85 chính là ôtômat tối thiểu đã được rút gọn.

Từ nhận xét trên, tổng quát lại chúng ta có thể nói bộ đếm luôn là một ôtômat tối thiểu, hay khi thực hiện bộ đếm chúng ta không cần quan tâm tới việc rút gọn nó nữa (không giống như đối với việc tổ hợp một ôtômat bất kỳ).

Sau khi có được bảng chức năng của bộ đếm 5 là tối thiểu, bước tiếp theo chúng ta thực hiện quá trình mã hóa trạng thái cho nó, hay nói cách khác là "đặt tên" cho các trạng thái của bộ đếm dưới dạng nhị phân. Việc đặt tên hay mã hóa trạng thái cho ôtômat (bộ đếm) theo yêu cầu đề ra cho bộ đếm là trạng thái trong (trạng thái các trigơ của bộ đếm) của nó phải theo quy luật của mã nhị phân thông thường nên bảng đặt tên cũng như bảng mã hóa chức năng của bộ đếm được chỉ ra trên hình 5.87.

0	0 0 0
1	0 0 1
2	0 1 0
3	0 1 0
4	1 0 0

a)

$y_1y_2y_3$ \ x	0	1	Z
0 0 0	000	001	0
0 0 1	001	010	0
0 1 0	010	011	0
0 1 1	011	100	1
1 0 0	100	000	0

b)

y	Y	J	K
0	0	0	–
0	1	1	–
1	0	–	1
1	1	–	0

c)

Hình 5.87. Bảng mã hóa bộ đếm 5.

Có bảng mã hóa, bước tiếp theo ta lập bảng kích cho bộ đếm, vì muốn trạng thái của trigơ của bộ đếm lật chuyển như bảng mã hóa, thì chúng ta cần có các tín hiệu kích cho từng trigơ như bảng kích. Muốn có bảng kích cho ôtômat, ta lại kết hợp giữa bảng mã hóa và bảng chuyển biến trạng thái của trigơ JK, vì nhiệm vụ là dùng JKFF thiết kế bộ đếm 5. Bảng kích của bộ đếm chỉ ra trên hình 5.88.

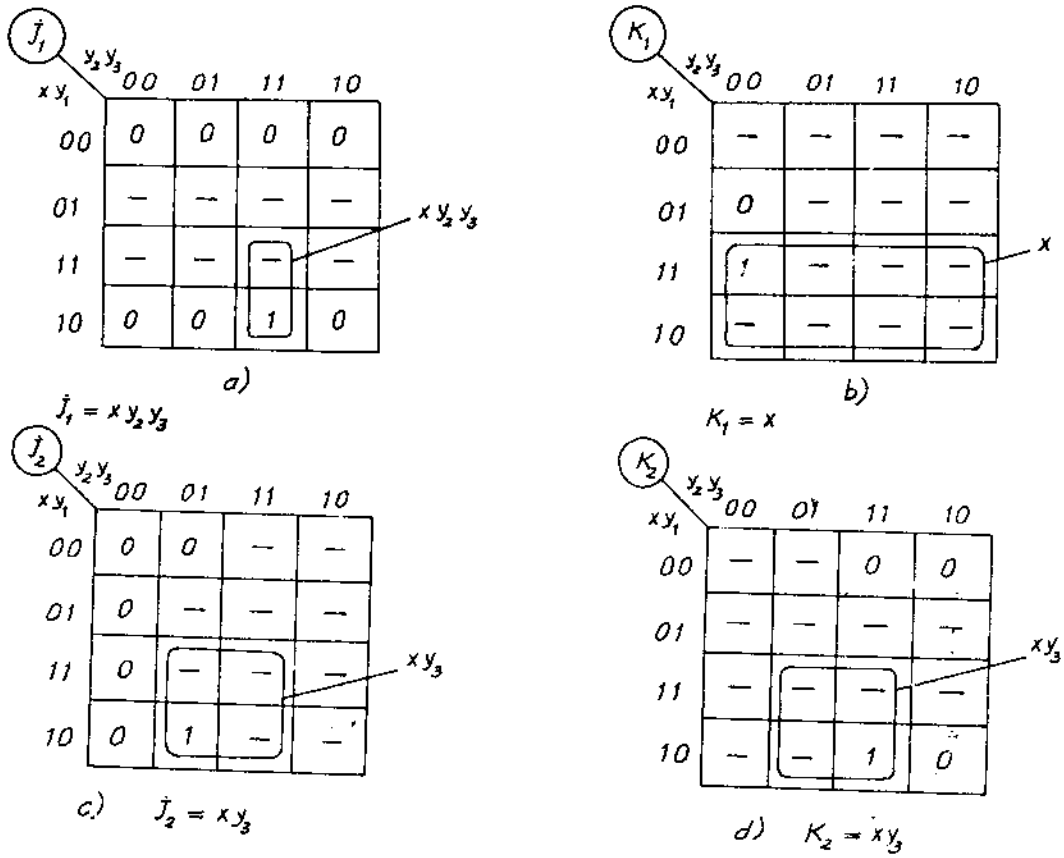
Bộ đếm 5 là mạch điện cụ thể, mạch điện đó phải tạo ra được tín hiệu kích giống như tín hiệu kích của bảng kích, mạch điện cụ thể đó chính là hệ phương trình kích của bộ đếm này. Chúng ta nhận thấy bảng kích viết không theo quy luật của Vla Kácnô, vì thế ta không thể nhóm họp hay tìm được phương trình kích trực tiếp từ bảng kích. Ở

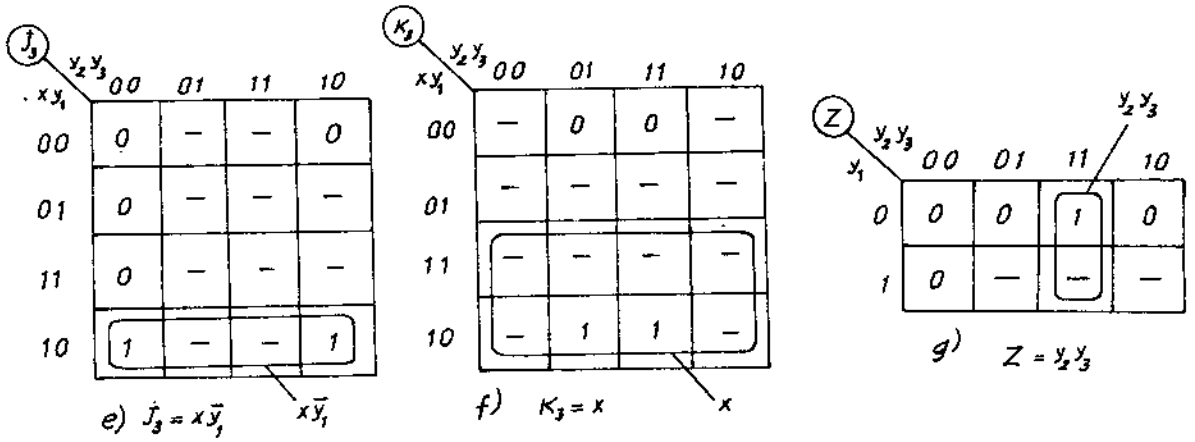
	x / $y_1y_2y_3$	0						1						Z
		J_1	K_1	J_2	K_2	J_3	K_3	J_1	K_1	J_2	K_2	J_3	K_3	
0	0 0 0	0	–	0	–	0	–	0	–	0	–	1	–	0
1	0 0 1	0	–	0	–	–	0	0	–	1	–	–	1	0
2	0 1 0	0	–	–	0	0	–	0	–	–	0	1	–	0
3	0 1 1	0	–	–	0	–	0	1	–	–	1	–	1	1
4	1 0 1	–	0	0	–	0	–	–	1	0	–	0	–	0

Hình 5.88. Bảng kích dùng JKFF của bộ đếm.

đây muốn tìm được hệ phương trình kích chúng ta phải ánh xạ vào bìa Kácnô đối với từng cực kích hay từng đầu vào kích của từng trigơ một. Trước khi ánh xạ cụ thể vào bìa Kácnô các cực kích của bộ đếm ta còn cần biết thêm là : bộ đếm 5 là một bộ đếm vòng quanh cho nên các trạng thái thừa của nó (còn 3 trạng thái hay 3 bộ mã thừa) không được xuất hiện trong bộ đếm, điều đó tạo thêm cho chúng ta những vị trí không xác định mà ta có thể chọn lựa để tạo ra những mạch điện đơn giản nhất.

Từ bảng kích đã biết ở hình 5.88 ánh xạ vào bìa K cho từng cực kích của từng trigơ được chỉ ra trên hình 5.89.





Hình 5.89. Tìm hệ phương trình kích của bộ đếm 5.

Hệ phương trình kích hay mạch điện cụ thể của bộ đếm 5 là :

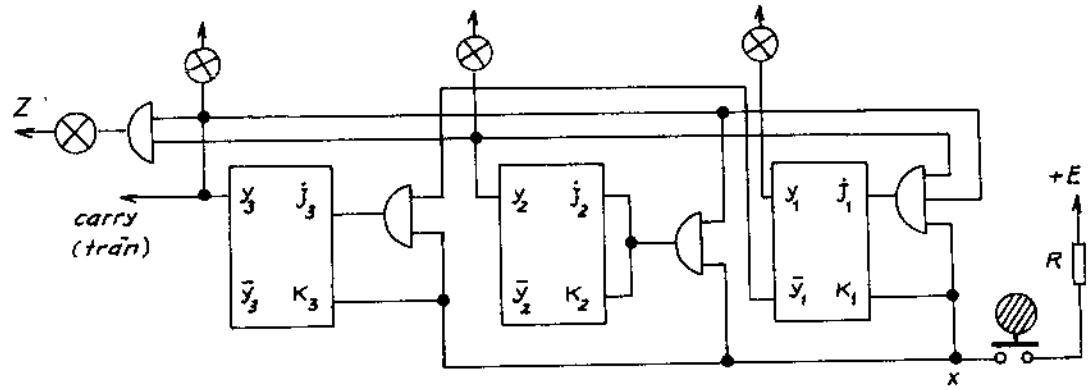
$$J_1 = x\,y_2\,y_3 \qquad K_1 = x$$

$$J_2 = x\,y_3 \qquad K_2 = x\,y_3$$

$$J_3 = x\,\bar{y}_1 \qquad K_3 = x$$

$$Z = y_2\,y_3$$

Từ hệ phương trình kích ta có mạch điện cụ thể của bộ đếm 5, trạng thái trong biến đổi theo quy luật nhị phân thông thường, mà cứ đếm đến 3 thì sẽ có báo sáng ở đầu ra Z, thiết kế dùng trigơ JK (JKFF) hoạt động theo ôtômat Moore, tức là đầu ra Z hoạt động không phụ thuộc trực tiếp vào tín hiệu vào x, mạch điện đó được mô tả trên hình 5.90.



Hình 5.90. Bộ đếm 5 trạng thái trong theo BCD.

Với cách làm tương tự ta có thể xây dựng bộ đếm 5 với cách biểu diễn theo bảng ôtômat Mealy.

Ví dụ này đã trình bày phương pháp thiết kế một bộ đếm có hệ cơ số đếm bất kỳ, nhưng trạng thái trong của bộ đếm sẽ biến đổi lần lượt theo quy luật nhị phân thông thường, đây là một yêu cầu rất quan trọng khi ứng dụng để thiết kế bộ đếm dùng để chỉ thị số các kết quả đo lường trong thực tế như : đo tốc độ chỉ thị số, đo trọng lượng

chỉ thị số, đo nhiệt độ chỉ thị số v.v... Khi trạng thái trong của bộ đếm thay đổi theo quy luật nhị phân thông thường thì ta hoàn toàn có thể ghép bộ đếm với bộ giải mã nhánh (7 thanh) chỉ thị số.

Trong ví dụ nêu lên cách thiết kế bộ đếm dùng JKFF, nhưng với cách làm tương tự ta có thể thiết kế dùng các loại trigơ khác nữa, như dùng DFF, TFF hay RSFF, cũng như có thể dùng với các loại trigơ hoạt động tự do hay theo một xung nhịp đồng bộ, tùy theo chức năng của một trigơ cụ thể khi ta dùng nó để thiết kế bộ đếm.

Như vậy qua ví dụ thiết kế bộ đếm 5 (cơ số đếm bất kỳ) chúng ta thấy trạng thái trong của bộ đếm được quyết định bởi quá trình mã hóa – quá trình đặt tên các trạng thái của bộ đếm. Nếu chúng ta đặt tên (mã hóa) trạng thái của từng trạng thái của bộ đếm theo quy luật mã nhị phân thông thường, thì trạng thái trong của bộ đếm sẽ thay đổi theo đúng quy luật của mã nhị phân BCD thông thường. Còn nếu chúng ta mã hóa trạng thái của bộ đếm theo quy luật khác (quy luật nào đó), thì trong quá trình mạch điện của bộ đếm (cơ số đếm bất kỳ) hoạt động, trạng thái trong (trạng thái của các trigơ của bộ đếm) của nó sẽ biến đổi theo quy luật của mã trạng thái đó.

5.6.3.5. Bộ đếm dùng các loại mã khác

Bộ đếm cơ số đếm 5 (cơ số đếm bất kỳ) có trạng thái trong thay đổi theo mã Gray (mã chống nhiễu), với khoảng cách Hamning giữa hai từ mã (hai trạng thái) cạnh nhau của bộ đếm chỉ khác nhau một bit (biến) thông tin được biểu diễn trên hình 5.91.

S \ x	0	1	Z
0	0	1	0
1	1	2	0
2	2	3	0
3	3	4	1
4	4	0	0

	$y_1y_2y_3$ \ x	0	1	Z
0	000	000	001	0
1	001	001	011	0
2	011	011	010	0
3	010	010	110	1
4	110	110	000	0
		$Y_1Y_2Y_3$	$Y_1Y_2Y_3$	

Hình 5.91. Bộ đếm 5 dùng mã Grây.

Dựa vào bảng mã hóa của bộ đếm 5 theo mã Grây ta có thể lập được bảng kích và thực hiện được bộ đếm đó dùng trigơ JK nếu ta muốn dựa vào hệ phương trình kích thu được từ bảng kích.

Với cách làm tương tự chúng ta có thể xây dựng bộ đếm 6 (cơ số đếm bất kỳ) với mã Johnson, hay trạng thái của bộ đếm thay đổi theo quy luật của mã Johnson bằng cách đặt tên (mã hóa trạng thái) của bộ đếm theo quy luật của mã Johnson. Hoạt động của bộ đếm 6 cứ 4 thì báo dùng mã Johnson được chỉ ra trên hình 5.92.

S \ x	0	1	Z
0	0	1	0
1	1	2	0
2	2	3	0
3	3	4	0
4	4	5	1
5	5	0	0

	$y_1y_2y_3$ \ x	0	1	Z
0	000	000	100	0
1	100	100	110	0
2	110	110	111	0
3	111	111	011	0
4	011	011	001	1
5	001	001	000	0
		$Y_1Y_2Y_3$	$Y_1Y_2Y_3$	

Hình 5.92. Bộ đếm 6 dùng mã Johnson.

Cũng từ bảng mã hóa của bộ đếm 6 với mã Johnson chúng ta có thể tìm được bảng kích vào thực hiện được bộ đếm đó bằng trigơ JK nếu ta muốn.

Nếu dùng mã vòng chúng ta cũng có thể thực hiện được bộ đếm 6 với mã vòng chỉ ra trên hình 5.93.

	$y_1\ y_2\ y_3\ y_4\ y_5\ y_6$ \ x	0	1	Z
0	1 0 0 0 0 0	1 0 0 0 0 0	0 1 0 0 0 0	0
1	0 1 0 0 0 0	0 1 0 0 0 0	0 0 1 0 0 0	0
2	0 0 1 0 0 0	0 0 1 0 0 0	0 0 0 1 0 0	0
3	0 0 0 1 0 0	0 0 0 1 0 0	0 0 0 0 1 0	0
4	0 0 0 0 1 0	0 0 0 0 1 0	0 0 0 0 0 1	1
5	0 0 0 0 0 1	0 0 0 0 0 1	1 0 0 0 0 0	0
		$Y_1\ Y_2\ Y_3\ Y_4\ Y_5\ Y_6$		

Hình 5.93. Bộ đếm 6 dùng mã vòng 6 bit.

Khi có bảng mã hóa của bộ đếm 6 dùng mã vòng, từ đó chúng ta lập ra bảng kích vào dùng JKFF thực hiện cụ thể bộ đếm đó nếu chúng ta muốn có, với việc thực hiện là đơn giản và quen thuộc như đã trình bày ở phần trên.

Từ các cách làm nêu trên chúng ta có thể xây dựng cụ thể được các bộ đếm có cơ số đếm lớn hơn tùy theo ý muốn của mình (bộ đếm cơ số đếm bất kỳ), mà bộ đếm đó có thể hoạt động theo quy luật của mã nhị phân BCD, mã Grây – mã bìa Kácnô, mã Johnson hay mã vòng v.v... một cách dễ dàng.

Đến đây ta có thể nói là cách thực hiện hay thiết kế một bộ đếm hoàn toàn giống như cách thực hiện một ôtômat bất kỳ. Bộ đếm và ôtômat bất kỳ giống nhau ở chỗ có thể mã hóa (đặt tên) trạng thái của chúng là tùy ý. Nhưng giữa bộ đếm và ôtômat có

nhớ có sự khác nhau cơ bản là ở chỗ : bộ đếm chỉ có một đường tín hiệu kích vào, trong khi việc kích vào ôtômat có nhớ có thể được thực hiện bằng nhiều đường tín hiệu một lúc (tập các tín hiệu vào X). Khi thay đổi trạng thái dưới tác động của tín hiệu vào thì bộ đếm thay đổi lần lượt trạng thái của nó theo trình tự đếm, trong khi ở ôtômat có nhớ khi thay đổi trạng thái dưới tác động của tín hiệu vào là không thay đổi trạng thái theo trình tự, mà theo trạng thái nào đó của ôtômat tùy theo nhiệm vụ đề ra. Điều khác nhau cơ bản thứ ba giữa chúng là trong quá trình thay đổi trạng thái theo quan hệ vòng quanh (vì bất kỳ ôtômat có nhớ nào cũng hoạt động vòng quanh) thì trong quá trình hoạt động vòng quanh đó các trạng thái của bộ đếm chỉ được xuất hiện có một lần, còn với ôtômat có nhớ ở trong quá trình hoạt động vòng quanh của nó, thì một trạng thái của nó có thể được xuất hiện nhiều lần tùy theo công việc được đề ra.

5.6.3.6. Bộ đếm thuận nghịch theo mã nhị phân

Bộ đếm thuận nghịch (cộng – trừ) là một bộ đếm có thể thực hiện được hai chức năng xen kẽ nhau, đếm thuận – *đếm tăng* và đếm nghịch – *đếm trừ*. Để điều khiển được bộ đếm nay ta chỉ cần dùng một đường tín hiệu vào là x, giá trị của tín hiệu vào x có thể là 0 hoặc 1 xuất hiện tùy ý nhưng ta có quy ước :

x = 1 là đếm thuận đếm tăng – đếm cộng.

x = 0 là đếm nghịch đếm giảm – đếm trừ.

Bảng chức năng của bộ đếm thuận nghịch với cơ số đếm là 8 (Kđ = 8) cơ số đếm 8 theo mã nhị phân thông thường (BCD) được chỉ ra trên hình 5.94.

S \ x	0	1
0	7	1
1	0	2
2	1	3
3	2	4
4	3	5
5	4	6
6	5	7
7	6	0

	$y_1y_2y_3$ \ x	0	1
0	000	111	001
1	001	000	010
2	010	001	011
3	011	010	100
4	100	011	101
5	101	100	110
6	110	101	111
7	111	110	000

Hình 5.94. Bộ đếm thuận nghịch Kđ = 8.

Dựa vào bảng chức năng trên hình 5.94, bước tiếp theo là lập ra bảng kích cho bộ đếm đó, để tìm ra được hệ phương trình kích và thực hiện mạch điện của bộ đếm.

Trên cơ sở của bộ đếm thuận nghịch cơ số đếm 8 với mã nhị phân thường chúng ta có thể thực hiện được bộ đếm thuận nghịch với cơ số đếm bất kỳ (Kđ = 6, Kđ = 10 v.v...) với mã nhị phân BCD, cũng như có thể thiết kế được bộ đếm thuận nghịch cơ số đếm bất kỳ với các loại mã khác như mã Grây, mã Johnson hay mã vòng v.v...

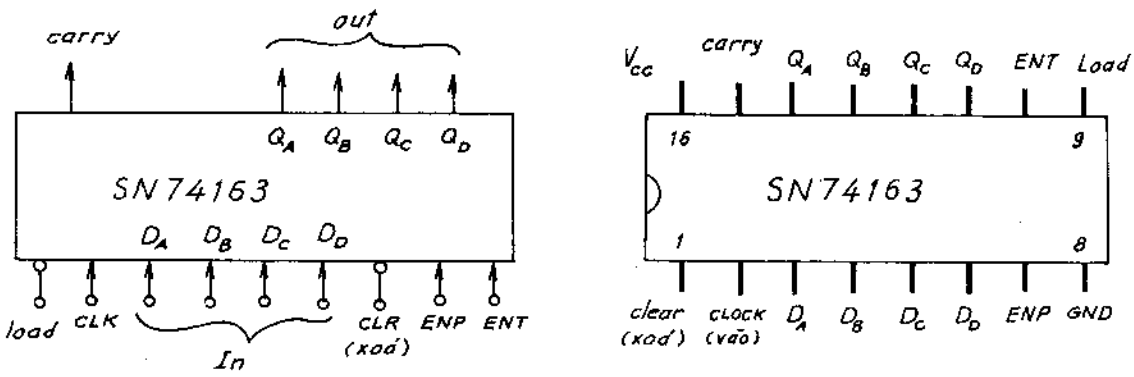
Bộ đếm thuận nghịch trong thực tế thường hay được sử dụng để đếm số lượng ôtô vào ra liên tục ngẫu nhiên trong bãi để xe, hay đo tốc độ tức thời của ôtô đi trên đường v.v...

5.6.2.7. Bộ đếm vạn năng hay bộ đếm điều khiển trạng thái

Bộ đếm vạn năng là một bộ đếm đã được chế tạo hoàn chỉnh trong một IC tiêu chuẩn. Từ IC đếm đã có sẵn đó, chúng ta chỉ cần một phép tổ hợp đơn giản là có thể thực hiện được một bộ đếm với cơ số đếm bất kỳ (Kđ theo ý của ta muốn và $\neq 2^n$), đồng thời trạng thái trong của nó biến đổi theo quy luật của mã nhị phân (BCD), điều này cho phép ta nối trực tiếp với bộ giải mã và chỉ thị được ngay kết quả đếm. Loại bộ đếm vạn năng bằng các IC được sản xuất sẵn theo tiêu chuẩn có rất nhiều loại khác nhau và của nhiều hãng sản xuất khác nhau, nhưng cách sử dụng chúng để thực hiện đếm (với cơ đếm Kđ $\neq 2^n$) có cơ số đếm bất kỳ là giống nhau và tổ hợp để chỉ thị kết quả cũng như nhau.

Ví dụ : Sử dụng IC đếm vạn năng SN74163.

Sơ đồ chức năng và sơ đồ chân của SN74163 là bộ đếm điều khiển trạng thái, được chỉ ra trên hình 5.95.



Hình 5.95. Cấu tạo IC đếm SN74163.

Muốn cho IC làm việc ngoài việc cung cấp nguồn V_{cc} đồng thời phải cho đầu vào nạp (load) ở điện thế thấp (0 von).

Tín hiệu vào dùng để đếm (x) được đưa vào đầu Clock là các xung đơn, x cũng có thể là xung nhịp khi dùng để chia tần (phân tần) cũng có thể là xung đơn của từng kết quả cần đếm.

Đầu ra ở chân Carry là tín hiệu báo tràn bộ đếm, tín hiệu chuyển hay tín hiệu nhớ sang cột số bên cạnh. Đầu ra Carry của bộ đếm này được nối tiếp vào đầu vào Clok của bộ đếm tiếp theo, cho phép chúng ta nối được nhiều bộ đếm với nhau để tạo ra được bộ đếm có cơ đếm lớn hơn

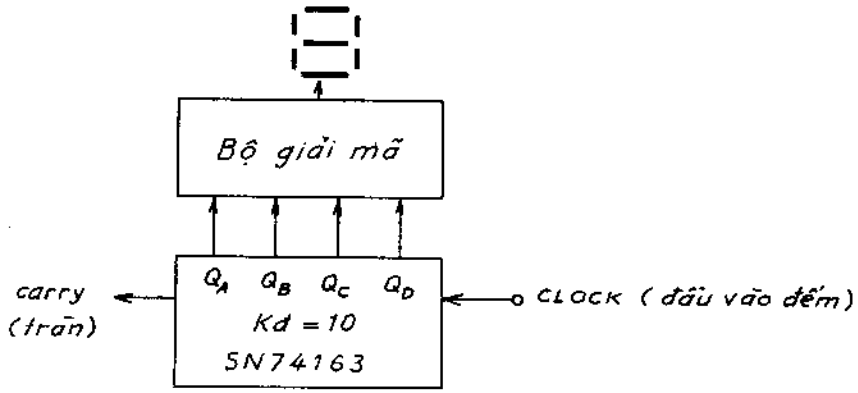
$$Kđ = Kđ_1 . Kđ_2 .... Kđ_n = \prod_{i=1}^{n} Kđ_i$$

Các đầu vào điều khiển thêm là ENP và ENT, chúng là đầu tín hiệu cho phép

đếm, chỉ khi cả hai đầu điều khiển ENP và ENT cùng có tín hiệu thì bộ đếm trong IC mới thực hiện đếm. Điều đó cho phép chúng ta dùng các chân điều khiển phụ ENP và ENT để thực hiện quá trình đếm được đồng bộ. Nếu ta không cần đồng bộ hay không dùng các cực ENP và ENT thì nối chúng lên (+E) nguồn cung cấp V_{cc}. Trong trường hợp này bộ đếm SN74163 hoạt động hoàn toàn độc lập.

Đầu vào Clear (CLR) là đầu vào xóa nội dung của bộ đếm, đây là đầu vào kích không đồng bộ. Khi ta cho đầu CLR ở mức điện áp thấp (0 von) thì được kết quả là xóa toàn bộ bộ đếm bất kể các đầu vào điều khiển khác như thế nào (không phụ thuộc vào các đầu điều khiển). Đầu Clear (xóa) cho phép xóa bộ đếm để thực hiện quá trình đếm từ đầu hay đồng bộ quá trình đếm. Khi đếm ta chỉ cần cho xung vào đầu vào Clock, thì bộ đếm thấy đổi một trạng thái theo quy luật nhị phân tăng dần.

Để chỉ thị trạng thái trong của bộ đếm chúng ta dùng các đầu ra (Q_A, Q_B, Q_C, Q_D,), tổ hợp trạng thái của các đầu ra theo kết quả của mã nhị phân (BCD), đồng thời kết quả chỉ thị đó cũng chính là kết quả đếm của bộ đếm thu được. Do các đầu ra Q hoạt động theo quy luật của nhị phân, nên thường được nối trực tiếp với đầu vào của bộ giải mã, để chỉ thị luôn con số mà bộ đếm đã nhận được. Điều đó cho phép chúng ta tổ chức một cột chỉ thị chữ số tiêu chuẩn như hình 5.96.



Hình 5.96. Một cột chỉ thị số.

Khi ta tổ hợp bộ đếm có cơ số lớn, cần phải kết hợp nhiều bộ đếm (ví dụ Kđ = 10) lại với nhau - Kết hợp nhiều cột số đếm lại với nhau, điều quan trọng nhất khi tổ hợp các cột số đếm lại là : xung vào đếm (xung CLOK) luôn phải cho vào đầu vào đếm của cột số ứng với hàng số trẻ nhất, thông thường phải là cột số ở hàng đơn vị. Nếu chúng ta không chú ý tới điều đó thì có thể gặp trường hợp con số đếm bị đọc (hay sắp xếp) ngược. Khi lắp mạch điện thật cần phải lưu ý tới điều này.

Các đầu kích vào điều khiển D - (D_A D_B D_C D_D) là các đầu vào điều khiển tổ hợp, cách tổ hợp các đầu vào D sẽ quyết định tới hệ cơ số đếm của bộ đếm này.

Ví dụ : Tổ hợp các đầu điều khiển để có bộ đếm cơ số 10 (Kđ = 10).

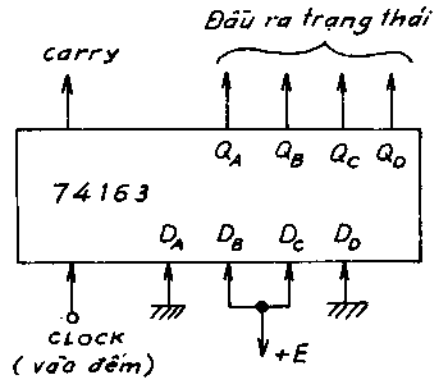
Cách làm là : 2^n - Kđ = số nhị phân → 0 nối tiếp 0 von

1 nối +E

Vậy ta có : $2^4 = 16$

$$16 - 10 = 6 = 0 \quad 1 \quad 1 \quad 0$$
$$D_A \; D_B \; D_C \; D_D$$

Cách nối cụ thể sẽ là : các đầu vào D_A và D_B ta nối 0 von còn các đầu vào D_B và D_D ta nối lên nguồn cung cấp +E. Mạch điện cụ thể của bộ đếm 10 (Kđ = 10) dùng mạch IC SN74163 sẽ được biểu diễn trên hình 5.98.



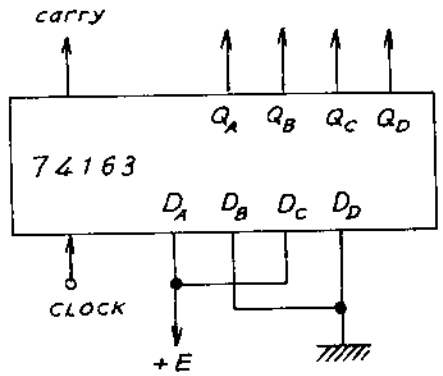
Hình 5.97. Bộ đếm có số 10 dùng SN74163.

Sau khi tổ hợp đấu nối xong các đầu điều khiển D, đồng thời các đầu vào điều khiển khác thoả mãn cho phép IC hoạt động, ta chỉ cần cho 10 xung vào đầu vào CLOK (đầu vào đếm) sẽ nhận được ở đầu ra Carry (đầu ra nhớ sang cột số bên cạnh) là 1 xung. Trong quá trình đếm các xung vào (CLOCK) thì trạng thái trong của bộ đếm sẽ biến đổi theo quy luật của nhị phân và được chỉ thị ra trên đầu ra Q (Q_A Q_B Q_C Q_D) đầu ra chỉ thị trạng thái trong của bộ đếm.

Với cách làm tương tự chúng ta có thể dùng SN74163 thực hiện bộ đếm 6 (Kđ = 6) và mạch điện của nó được chỉ ra trên hình 5.98.

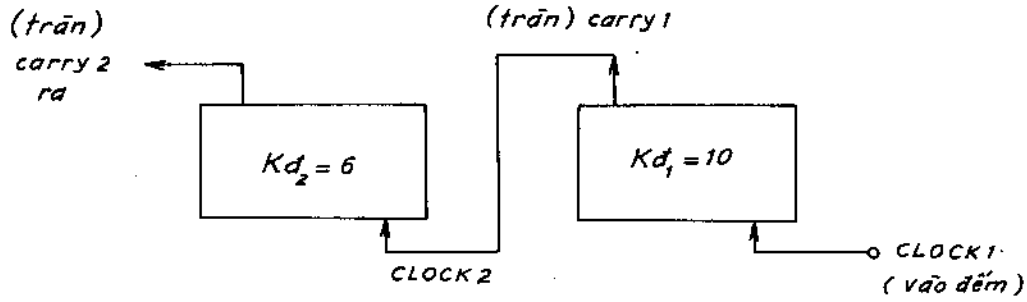
$$16 - 6 = 10 = \begin{matrix} 1 & 0 & 1 & 0 \\ D_A & D_B & D_C & D_D \end{matrix}$$

Vậy các đầu điều khiển D_A và D_C nối +E, còn các đầu vào D_B và D_D ta sẽ nối đất (0 von).



Hình 5.98. Bộ đếm có số 6 dùng SN74163.

Từ bộ đếm Kđ = 10 và bộ đếm Kđ = 6 ta có thể tổ hợp nên bộ đếm có cơ đếm 60 (Kđ = 60) như trên hình 5.99.

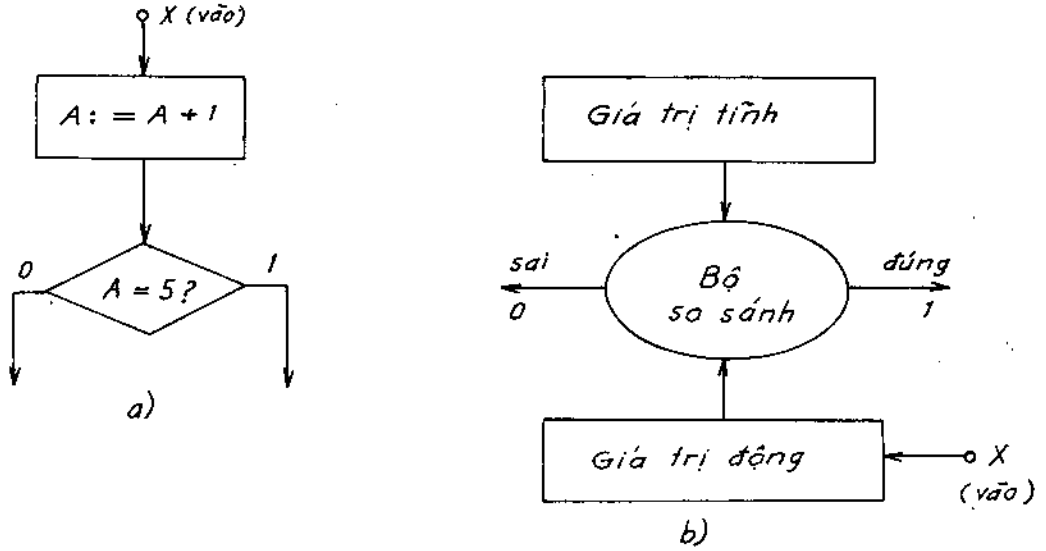


Hình 5.99. Bộ đếm cơ số đếm 60.

Dùng IC SN74163 chúng ta có thể tổ hợp và thực hiện được các bộ đếm khác nhau có cơ số đếm từ (0 cho tới 16). Cơ số đếm tối đa của con IC này là 16 sau đó nó lại vòng trở lại trạng thái ban đầu. Loại IC đếm SN74163 có rất nhiều loại khác nhau, nhưng cách tổ hợp ra các cơ số đếm bất kỳ cũng như nguyên lý làm việc là không có gì khác nhau. Từ đó chúng ta có thể tùy ý tìm chọn các con IC thích hợp để dùng.

5.6.4. BỘ SO SÁNH BẰNG NHAU (GIỐNG NHAU)

Bộ so sánh không phải là một ôtômat có nhớ, khi kết hợp với ôtômat có nhớ cho phép tạo ra các hệ thống số hoạt động linh hoạt hơn, hiệu quả hơn và tạo ra được nhiều chức năng hơn. Chức năng tổng quát của bộ so sánh được chỉ ra trên hình 5.100.



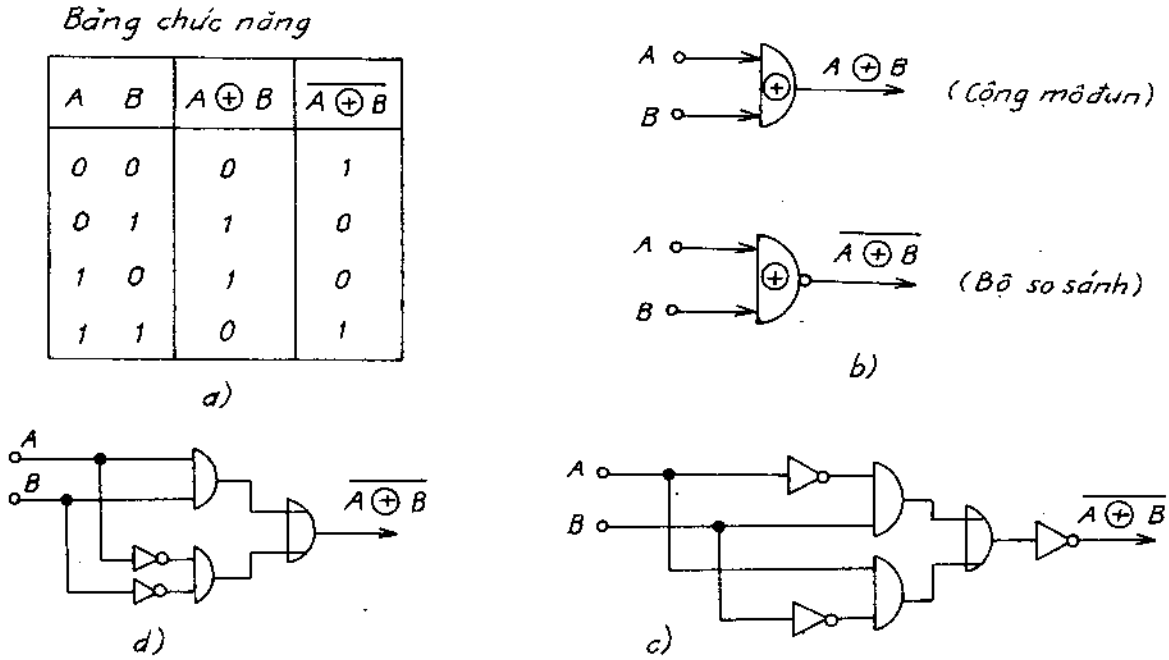
Hình 5.100. Bộ so sánh tổng quát.

Bộ so sánh luôn luôn phải thực hiện nhiệm vụ so sánh "giá trị tĩnh" (cho trước) với một "giá trị động" sẽ xuất hiện, nếu chúng có cùng giá trị (giống nhau) thì báo là đúng (1) còn nếu sai thì bộ so sánh báo sai (0). Thông thường "giá trị tĩnh" ứng với một mã số nhị phân được cho trước, giá trị cho trước này thường được ghi cố định vào một

thanh ghi tĩnh. Còn "giá trị động" thông thường hay gặp là một bộ đếm nhị phân tăng dần (cũng có thể giảm dần), nhưng giá trị động cũng có thể được lấy từ bộ tổng hay bộ ghi dịch v.v... Bộ so sánh làm nhiệm vụ so sánh mã số của thanh ghi tĩnh với kết quả thu được ở thanh ghi động với nhau. Để hiểu cụ thể hơn chức năng của bộ so sánh ta tìm hiểu bộ so sánh một bit thông tin.

5.6.4.1. Bộ so sánh một bit thông tin

Bảng chức năng cũng như mạch điện nguyên lý của bộ so sánh một bit được chỉ ra trên hình 5.101.



Hình 5.101. Bộ so sánh một bit.

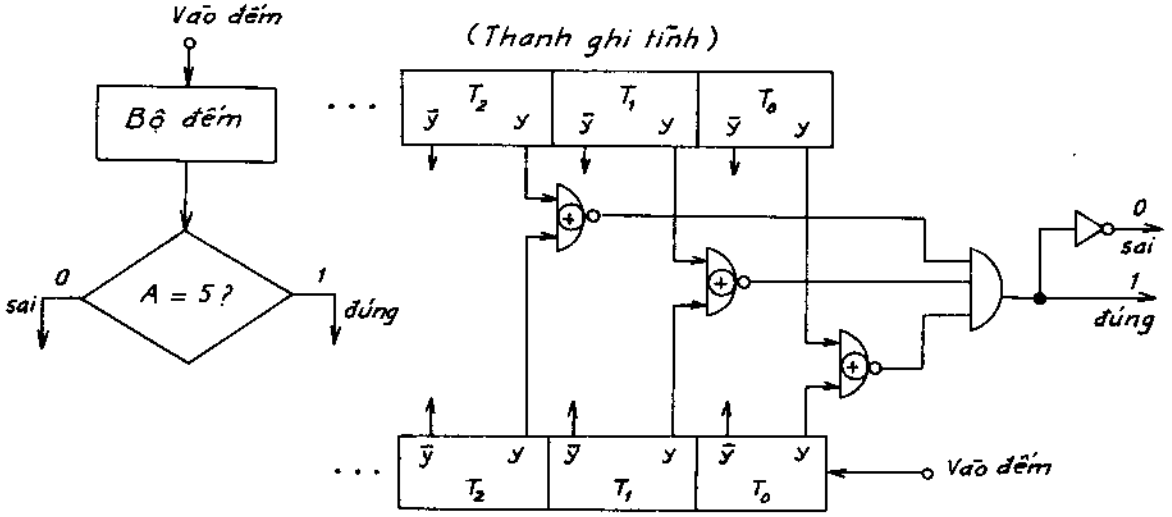
Bộ so sánh một bit thực chất là so sánh giữa một bit A và một bit B. Nếu đầu vào bit A và bit B giống nhau (cùng có giá trị 1 hay cùng có giá trị 0) thì đầu ra bộ so sánh có giá trị là 1, còn các trường hợp khác giữa hai bit A và B thì đầu ra báo là 0. Từ bảng chức năng ta cũng có được mạch điện cụ thể của bộ so sánh một bit (hình 5.101,c). Trên cơ sở của bộ so sánh một bit chúng ta có thể xây dựng được bộ so sánh với số lượng bit thông tin lớn hơn.

5.6.4.2. Bộ so sánh bằng nhau 3 bit

Bộ so sánh 3 bit dùng so sánh 3 bit thông tin của một thanh ghi tĩnh với 3 bit thông tin, ví dụ của một bộ đếm nhị phân tăng dần. Sơ đồ nguyên lý của bộ so sánh ba bit được chỉ ra trên hình 5.103.

Bộ so sánh ba bit là bộ so sánh giữa 3 bit của thanh ghi tĩnh với 3 bit của thanh ghi động (bộ đếm), bằng cách so sánh từng bit của hai thanh ghi đó với nhau trên cơ sở nguyên lý của bộ so sánh một bit. Đầu ra của bộ so sánh chỉ báo đúng khi tất cả các bit

thông tin từ T_0 đến T_2 so sánh thấy hoàn toàn giống nhau. Trên nguyên lý như vậy chúng ta có thể xây dựng nên bộ so sánh với số lượng bit lớn hơn.



Hình 5.102. Bộ so sánh ba bit.

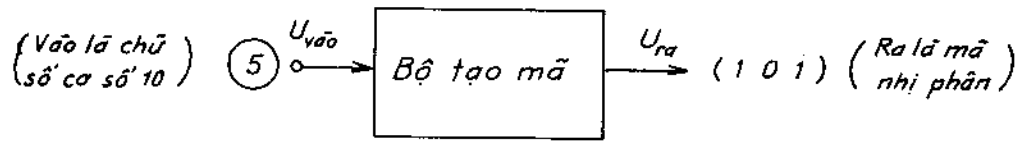
Ứng dụng của bộ so sánh trong thực tế rất phong phú. Ta có thể dùng để xây dựng khóa điện tử với một khóa mã cho trước, nếu cần khóa mã này có thể thay đổi mã khóa hàng ngày ; có thể áp dụng vào việc hẹn giờ hay cho trước giá trị cần dừng (ví dụ bơm xăng, đếm lượng sản phẩm v.v...) ; cũng có thể dùng điều khiển cầu thang máy khi so sánh thấy giống mã tầng thì thang máy dừng ; cũng như dùng trong việc nhận dạng vân tay, nhận dạng hình ảnh hay tiếng nói v.v...

Để xây dựng bộ so sánh nhiều bit người ta có thể dùng một cách khác mà không cần phải sử dụng nhiều mạch so sánh một bit, bằng cách dùng bộ ghi dịch có điều khiển. Bộ so sánh này chỉ cần dùng một bộ so sánh một bit, sau đó lần lượt dịch từng bit thông tin lên để so sánh, nếu thấy đúng thì lại dịch tiếp để so sánh bit tiếp theo và cứ làm lần lượt như vậy mãi cho đến hết thanh ghi. Nếu phát hiện thấy sai ở bit nào đó thì báo sai. Bộ so sánh lại này yêu cầu đồng bộ và có mạch điều khiển, mạch thực hiện sẽ gọn hơn.

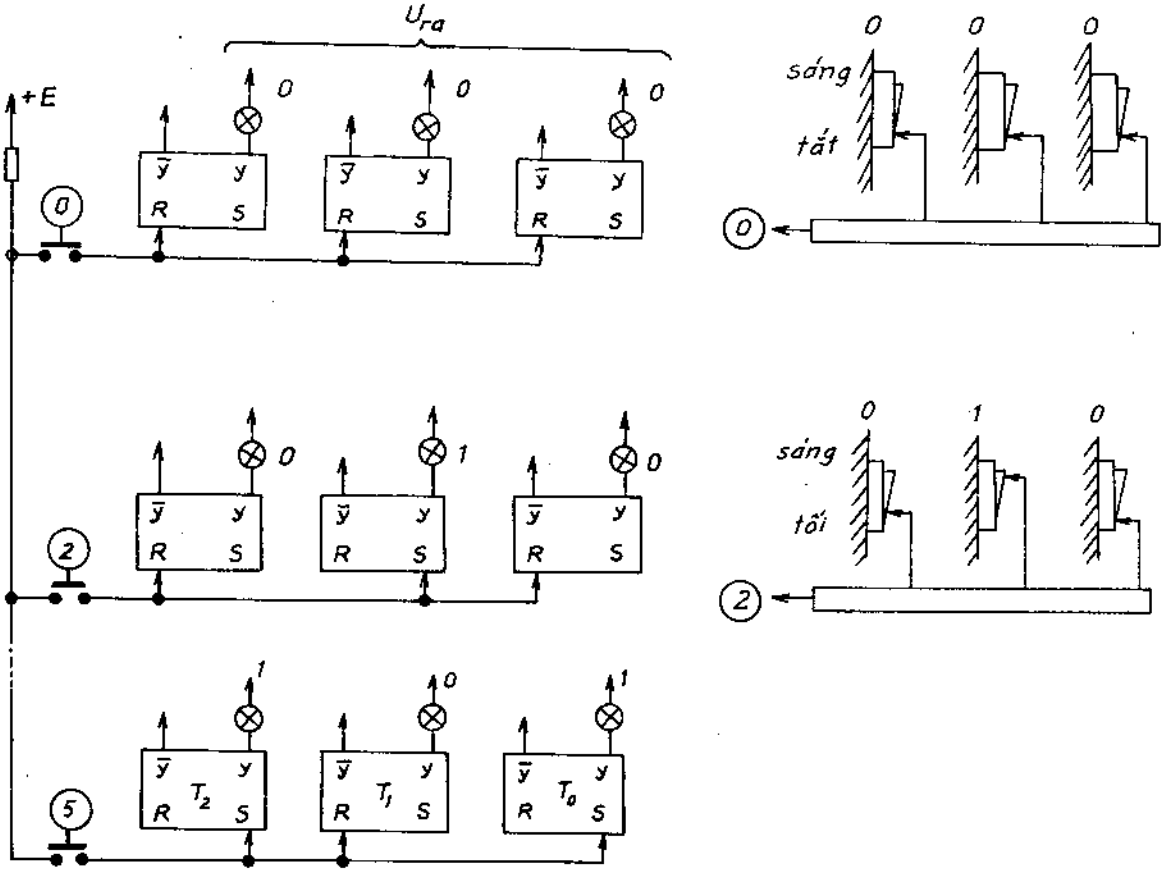
5.6.5. BỘ TẠO MÃ

Việc trao đổi thông tin giữa người và máy cần có bộ tạo mã để máy có thể "hiểu" được người khi trao đổi thông tin với nhau. Vì vậy để trao đổi thông tin với máy bắt buộc ta phải xây dựng ra "bộ tạo mã" để chuyển ý muốn của con người thành mã máy rồi truyền vào trong máy trao đổi với máy. Nhiệm vụ hay chức năng của bộ tạo mã được minh họa trên hình 5.103.

Chức năng của bộ tạo mã là khi ta cho vào một chữ số (một ký tự) ở cơ số đếm mười thì ở đầu ra của bộ tạo mã ta nhận được mã nhị phân tương ứng. Mạch điện để thực hiện chức năng đó, chính là bộ tạo mã. Mạch điện của bộ tạo mã được chỉ ra trên hình 5.104.



Hình 5.103. Chức năng bộ tạo mã.

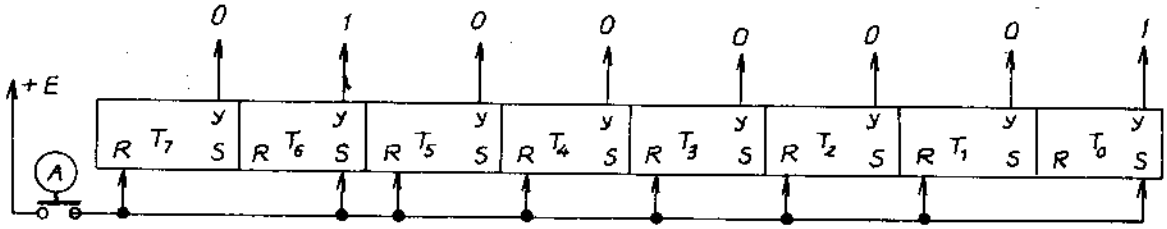


Hình 5.104. Bộ tạo mã nhị phân ba bit.

Nguyên lý làm việc của bộ tạo mã nhị phân ba bit rất đơn giản, khi ta bấm vào một phím (đầu vào bộ tạo mã) thì dù cho trước đó trạng thái của nó trigơ từ T_0 đến T_2 ở trạng thái bất kỳ nào đó, sau khi bấm phím thì các trigơ sẽ về trạng thái tương ứng với mã nhị phân của phím bấm đó. Các đầu ra y của các trigơ sẽ cho ra (đầu ra bộ tạo mã) mã nhị phân tương ứng.

Đầu ra của bộ tạo mã ta nhận được mã nhị phân (BCD) ở dạng song song, nếu lấy ra trên các đầu ra y của các trigơ chúng ta sẽ thu được mã thuận, còn nếu ta lấy ra trên các đầu ra $\bar{y}$ thì ta sẽ thu được mã ngược. Đầu ra của bộ tạo mã chúng ta có thể ghi song song vào một thanh ghi tĩnh để lưu giữ lại mã đó, đó chính là cơ sở để ghi nhận chương trình từ bàn phím vào bộ nhớ RAM của máy vi tính.

Sau khi xây dựng được bộ tạo mã từ cơ đếm 10 sang cơ đếm 2, ta có thể xây dựng được bộ tạo mã từ ký tự chữ viết sang mã nhị phân như sau. Bằng cách dựa vào bộ mã ASCII ta có mỗi một ký tự chữ viết sẽ có một mã nhị phân tương ứng. Ví dụ tạo mã cho phím (A), sơ đồ bộ tạo mã chữ A được chỉ ra ở trên hình 5.105.

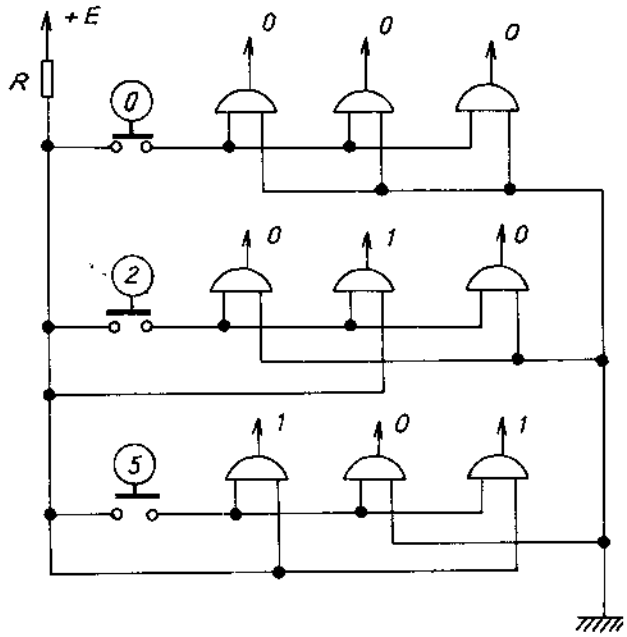


Hình 5.105. Bộ tạo mã cho chữ A.

Với cách làm tương tự chúng ta có thể tạo ra mã nhị phân của chữ B hay C v.v... cho tới chữ Z.

Để tạo ra mã máy từ bàn phím hay các phím bấm người ta có thể rút gọn hay nối mạch cho gọn hơn hay để bớt phải dùng nhiều các trigơ. Điều đó được minh họa trên hình 5.106.

Ứng dụng của bộ tạo mã là rất phong phú, nó có thể dùng xây dựng nên khóa điện tử mà chỉ bấm đúng số thì cửa mới mở còn không thì sẽ báo động, cũng như dùng trong tổng đài, trong hệ thống điện thoại hiện nay, rồi ở bàn phím của máy vi tính v.v... Bộ tạo mã phím bấm còn được dùng rộng rãi trong các máy dân dụng cũng như các máy công cụ. Ví dụ như máy giặt, máy dệt tự động v.v...

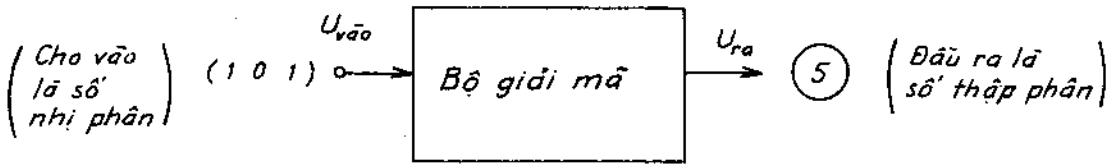


Hình 5.106.
Bộ tạo mã đơn giản.

Một điều cần lưu ý khi xây dựng cũng như xem xét một bộ tạo mã là : mã là do con người tự quy ước ra, cho nên mỗi người có thể có quyền quy ước những "mã riêng" và tạo ra mã riêng để phục vụ cho việc riêng của mình. Chính vì lẽ đó không phải lúc nào bộ tạo mã cũng cho ra bộ mã nhị phân thông thường (BCD), mà có thể là một mã bất kỳ nào đó. Vì thế muốn hiểu được hoạt động của hệ thống thông qua bộ tạo mã, thì trước hết ta phải biết được quy ước mã của người thiết kế ra nó như thế nào, rồi từ đó mới hiểu được hoạt động của nó cũng như sửa chữa nó nếu cần.

5.6.6. BỘ GIẢI MÃ

Bộ giải mã là một thuật ngữ rất quen thuộc hay được dùng trong các hệ thống số cũng như trong máy tính. Nhiệm vụ của bộ giải mã là chuyển đổi từ mã máy dưới dạng mã nhị phân thành "nghĩa" ở cơ số mười để con người có thể "hiểu" máy trong quá trình thực hiện việc trao đổi thông tin với máy. Bộ giải mã có thể chuyển đổi một mã nhị phân của máy thành ra chữ viết. Muốn xây dựng được những bộ giải mã như vậy trước hết ta phải hiểu được cách xây dựng bộ giải mã từ mã nhị phân thành ra cơ số đếm mười. Nhiệm vụ của bộ giải mã đó được chỉ ra trên hình 5.107.



Hình 5.107. Sơ đồ khối chức năng bộ giải mã.

Chức năng của bộ giải mã là nếu ta cho vào một mã nhị phân bất kỳ thì ở đầu ra của nó ta sẽ nhận được một chữ số của cơ đếm mười tương ứng. Để thực hiện được nhiệm vụ đó của bộ giải mã ta có hai phương pháp :

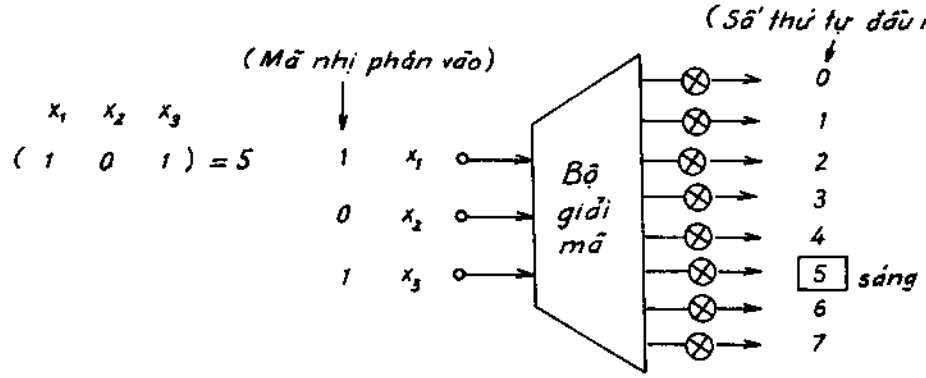
- Mã nhị phân đưa vào tương ứng với số thứ tự ở đầu ra của bộ giải mã.

- Mã nhị phân cho vào qua bộ giải mã sẽ cho ra chữ số tương ứng với mã đó. Bộ giải mã kiểu này còn có tên gọi là bộ giải mã nhánh hay "bộ giải mã 7 thanh".

5.6.6.1. Bộ giải mã với mã nhị phân vào tương ứng với số thứ tự đầu ra

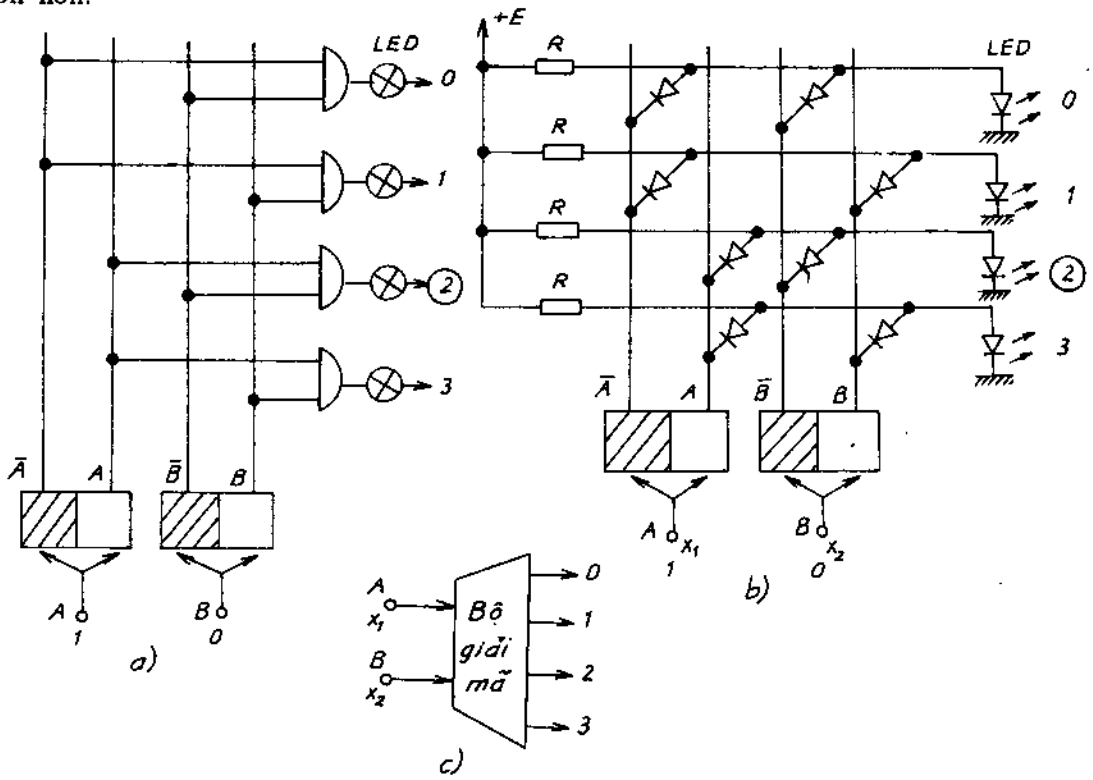
Chức năng của bộ giải mã kiểu này được minh họa bằng bộ giải mã ba bit vào (x_1 x_2 x_3) chỉ ra trên hình 5.108.

Số nhị phân đưa vào bộ giải mã phải cho vào ở dạng song song, ví dụ ta đưa mã song song (x_1 x_2 x_3 = 1 0 1) vào thì đầu ra ứng với số thứ tự là 5 thì đèn sẽ sáng lên tương ứng, tương tự nếu ta cho vào mã nhị phân (x_1 x_2 x_3 = 0 1 1) thì đèn tương ứng với số thứ tự là 3 sẽ sáng lên. Như vậy với mỗi một mã nhị phân cho vào sẽ tương ứng với một số thứ tự ở đầu ra, điều đó cho ta kết quả là một số nhị phân đưa vào sẽ tương ứng với một chữ số ở cơ số mười.



Hình 5.108. Bộ giải mã nhị phân 3 bit vào.

Mạch điện nguyên lý bộ giải mã với hai đầu vào (x_1 x_2) và tương ứng sẽ có 4 đầu ra được chỉ ra ở trên hình 5.109. Với nguyên lý xây dựng bộ giải mã 2 đầu vào bằng cách làm tương tự ta có thể xây dựng được các bộ giải mã khác với số lượng đầu vào lớn hơn.



Hình 5.109. Sơ đồ mạch điện bộ giải mã 2 đầu vào.

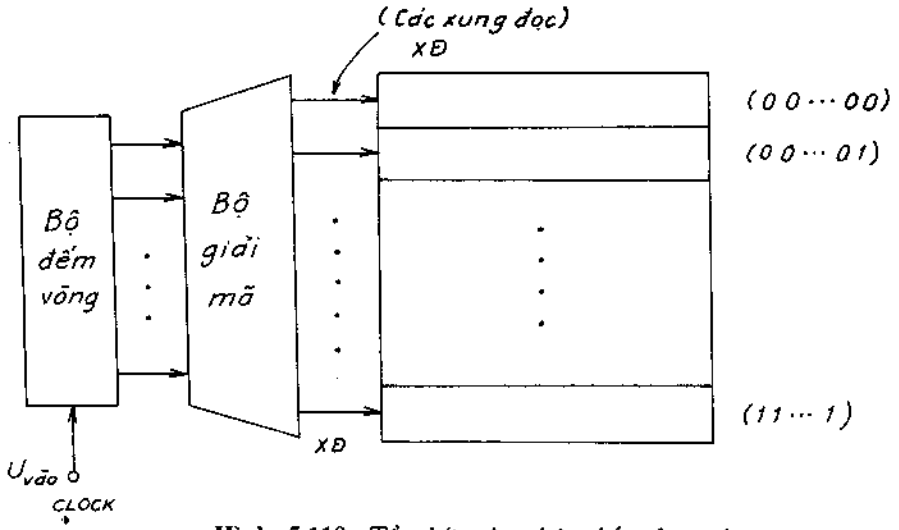
Nguyên lý hoạt động của bộ giải mã như sau : cho vào mã nhị phân (x_1 x_2 = AB = 10) như vậy trigơ A ở trạng thái 1 cho đầu ra A thế cao (ứng với lôgic 1), còn trigơ B ở trạng thái 0 có nghĩa là đầu ra $\overline{B}$ có thế cao (ứng với lôgic 1). Như vậy chỉ có mạch *VÀ* ứng với đầu ra 2 là làm việc vì đầu vào của nó có hai điện thế cao (ứng với mã 1), làm cho đầu ra ứng với mạch và số 2 sẽ có thế cao (lôgic 1) dẫn đến đèn LED tương

ứng của nó sẽ sáng. như vậy cho vào AB = 10 thì ứng với số 2 (đèn 2 sáng). Thực hiện tương tự với các bộ mã vào khác ta sẽ thu được đèn ở đầu ra sáng tương ứng với bộ mã đó.

Ở đây ta sử dụng mạch và hai đầu vào dùng điôt – điện trở, cho nên người ta thường gọi đây là bộ giải mã ma trận điôt – điện trở như trên hình 5.109,b. Các trigơ A và B ứng với đầu vào bộ giải mã ta có thể nhận được từ thanh ghi tĩnh, bộ tổng hay từ bộ đếm v.v...

Ứng dụng của bộ giải mã với mã nhị phân đưa vào tương ứng với số thứ tự ở đầu ra là rất phong phú, nếu ta cho mã vào bộ giải mã một cách lần lượt (thường được tạo ra từ bộ đếm) thì đầu ra bộ giải mã đèn sẽ sáng lần lượt, điều đó cho phép ta thực hiện việc quảng cáo với các chữ mỗi một chữ ứng với một đèn ở đầu ra bộ giải mã hay các hình lần lượt sáng tắt theo trình tự hay theo một quy luật mà ta dự kiến trước.

Nhưng ứng dụng quan trọng nhất của bộ giải mã kiểu này là bộ giải mã kết hợp với bộ đếm để lần lượt đọc các ô nhớ từ bộ nhớ của máy tính. Sơ đồ tổng quát ứng dụng bộ giải mã vào việc đọc lần lượt (đọc quét) bộ nhớ của máy tính được chỉ ra trên hình 5.110.



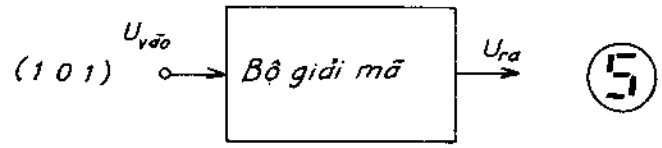
Hình 5.110. Tổ chức đọc bộ nhớ của máy.

Dung lượng của bộ nhớ từ ô địa chỉ (00...0) đến ô địa chỉ (11...1), bộ đếm có số đếm lần lượt từ số (00...0) đến giá trị đếm đầy là (11...1). Đầu ra của bộ đếm ta đưa vào đầu vào bộ giải mã, đầu ra của bộ giải mã là các đầu được đánh số thứ tự lần lượt từ đầu ra (00...0) đến đầu ra (11...1) tương ứng với các địa chỉ của các ô nhớ trong bộ nhớ. Khi ta cho xung vào (xung CLOCK) đầu vào bộ đếm làm cho giá trị của bộ đếm tăng dần, đây là bộ đếm vòng nó sẽ đếm từ giá trị (00...0) đến giá trị (11...1), làm cho đầu ra bộ giải mã xuất hiện lần lượt các xung ra tương ứng, xung đó là xung đọc đưa vào các đầu đọc của các ô nhớ một cách lần lượt, làm cho nội dung của cả bộ nhớ được lấy ra (đọc ra) một cách lần lượt. Do đây là bộ đếm vòng quanh nên giá trị của bộ đếm sau khi đầy lại trở lại giá trị ban đầu là (00...0) và nội dung bộ nhớ lại được đọc lại lần khác một

cách lần lượt. Nếu ta thu lại lần lượt các nội dung được đọc ra từ bộ nhớ, rồi sau đó đưa tín hiệu đó lên màn hình máy tính thì ta có thể hiển thị được toàn bộ nội dung của bộ nhớ lên màn hình của máy tính. Còn nếu muốn đọc nội dung của một ô nhớ bất kỳ nào đó mà ta cần, ta chỉ việc đưa con số đếm (ứng với số địa chỉ của ô nhớ cần đọc) vào bộ đếm, thì đầu ra bộ giải mã tương ứng xuất hiện xung đọc để đọc nội dung của ô nhớ đó. Ứng dụng của bộ giải mã loại này thì có nhiều và thuận tiện, nhưng nhược điểm của loại bộ giải mã mã nhị phân vào tương ứng với số thứ tự đầu ra là ta phải đếm và đánh số thứ tự của đầu ra mà không hiển thị được ngay giá trị của mã nhị phận vào thanh chữ số cơ đếm mười ở đầu ra. Chính vì vậy để thuận lợi hơn cho con người khi sử dụng người ta xây dựng bộ giải mã nhánh hay bộ giải mã 7 thanh.

5.6.6.2. Bộ giải mã 7 thanh

Nhiệm vụ của bộ giải mã 7 thanh là chỉ thị ngay kết quả của một mã nhị phân đưa vào thành một chữ số (một ký tự) của cơ số đếm 10 ở đầu ra, được chỉ ra trên hình 5.111.



Hình 5.111. Chức năng bộ giải mã 7 thanh.

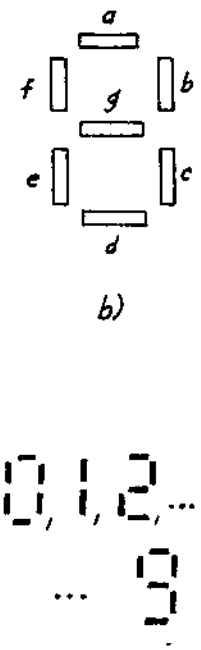
Yêu cầu mã nhị phân đưa vào bộ giải mã phải cho vào ở dạng song song (viết theo hàng dọc). Cách thiết kế hay cách xây dựng bộ giải mã nhánh được thực hiện qua các bước như sau :

Trước hết ta phải xây dựng bảng mô tả quan hệ giữa ký hiệu chữ số (ký tự) tương ứng với việc hiển thị trên bảng LED 7 thanh. Bảng đó được chỉ ra trên hình 5.113.

Do yêu cầu phải chỉ thị mười chữ số của cơ đếm 10 nên bộ đếm ta phải dùng 4 trigơ (A B C D), trong đó trigơ cuối cùng D là trigơ ứng với hàng trẻ nhất của bộ đếm, vậy tín hiệu đếm phải đưa vào đầu vào đếm của trigơ này. Đồng thời sẽ còn thừa ra 6 bộ mã ứng với các vị trí không xác định, vì bộ đếm mười là bộ đếm vòng quanh từ 0 đến 9 (từ 0000 đến 1001) sau đó lại quay trở lại. Đầu ra của bộ đếm cơ số 10 sẽ được đưa vào đầu vào bộ giải mã 7 thanh.

Theo ký hiệu của các chữ số cần phải chỉ thị ta sẽ cho các nhánh của bộ đèn LED 7 thanh sáng tối thích hợp, muốn thực hiện được điều đó trước hết phải ký hiệu "tên" cho từng nhánh của bộ đèn chỉ thị 7 thanh (ví dụ như trên hình 5.112,b). Ví dụ khi ta muốn chỉ thị số "0" thì nhánh g phải tắt (mã 0) còn các nhánh khác sẽ được sáng lên (mã 1) ; nếu muốn chỉ thị số "1" thì ta cho nhánh a và b sáng lên (mã 1) còn các nhánh khác thì tắt. Với cách làm tương tự chúng ta sẽ dựa vào 7 thanh mà chỉ thị được các chữ số từ (0 đến 9) của bộ đếm cơ 10, đồng thời điền đầy vào bảng điều khiển nhánh của bộ giải mã như trên hình 5.112,a.

	Bộ đếm				7 thanh						
Chữ số	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	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	–	–	–	–	–	–	–

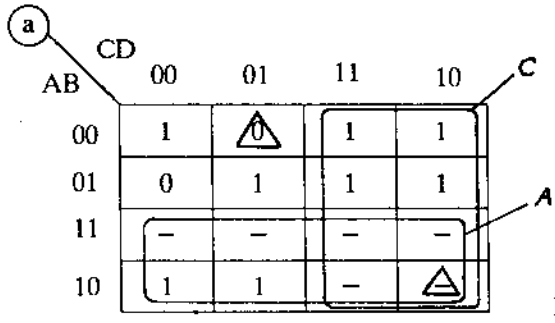


Hình 5.112. Bảng quan hệ giữa chữ số và bảng LED 7 thanh.

Trong bộ đèn chỉ thị 7 nhánh thì các nhánh từ (a ÷ g) hoạt động sáng tối hoàn toàn độc lập so với nhau, nhưng chúng sáng hay tối theo cùng một tín hiệu điều khiển được lấy ra từ bộ đếm. Nhờ hoạt động sáng tối của các nhánh cùng theo một lệnh điều khiển cùng một "khẩu lệnh", nên sự kết hợp sáng tối của từng nhánh lại với nhau sẽ cho ta các hình ảnh của một ký hiệu nào đó. Từ đó ta có thể hiểu rằng nếu các chữ số do con người ký hiệu ra mà viết kiểu khác thì sẽ không thể dùng 7 thanh (với số lượng thanh ít nhất) để chỉ thị được nữa. Chúng ta cần biết việc điều khiển sáng tối của một thanh theo tín hiệu điều khiển lấy ra từ bộ đếm (A B C D) như thế nào. Hay nói một cách khác phương trình toán học điều khiển một thanh (do hoạt động độc lập từng thanh) theo tín hiệu chung đưa vào từ bộ đếm có quan hệ thế nào ?

Bước tiếp theo là ánh xạ hoạt động của từng nhánh hay từng thanh vào bìa Kácnô theo tín hiệu điều khiển là (A B C D) đầu ra của bộ đếm, kết quả cụ thể như sau :

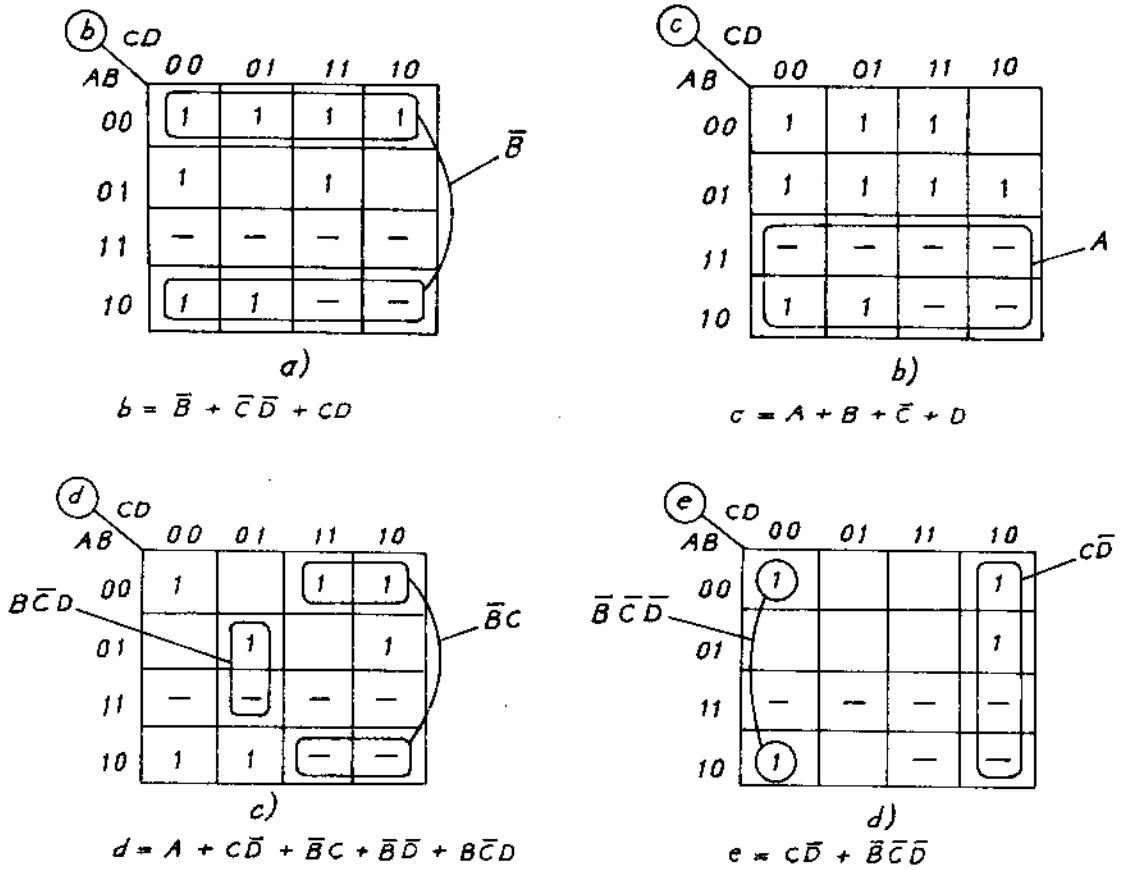
Đối với nhánh a ánh xạ vào bìa Kácnô ta thu được bảng chỉ ra trên hình 5.113.



$$a = A + C + BD + \overline{B}\,\overline{D}$$

Hình 5.113. Ánh xạ nhánh a vào bìa Kácnô.

Bằng cách làm tương tự chúng ta ánh xạ nhánh b, nhánh c, nhánh d, nhánh e vào bìa Kácnô kết quả chỉ ra trên hình 5.114.



Hình 5.114. Ánh xạ các nhánh vào bìa Kácnô.

Dựa vào kết quả ánh xạ các nhánh của bộ chỉ thị 7 nhánh vào bìa Kácnô ta thu được phương trình toán học của từng nhánh, đó chính là sự phụ thuộc (độc lập so với nhau) của từng nhánh theo tín hiệu điều khiển chung nhận được từ bộ đếm (A B C D). Với cách làm tương tự ta ánh xạ nốt nhánh f và nhánh g vào bìa và tìm được phương trình điều khiển nhánh của chúng.

Kết quả cuối cùng thu được hệ phương trình điều khiển từng nhánh theo tín hiệu điều khiển (A B C D) như sau :

$$a = A + C + B\,D + \overline{B}\,\overline{D}$$

$$b = \overline{B} + \overline{C}\,\overline{D} + C\,D$$

$$c = A + B + \overline{C} + D$$

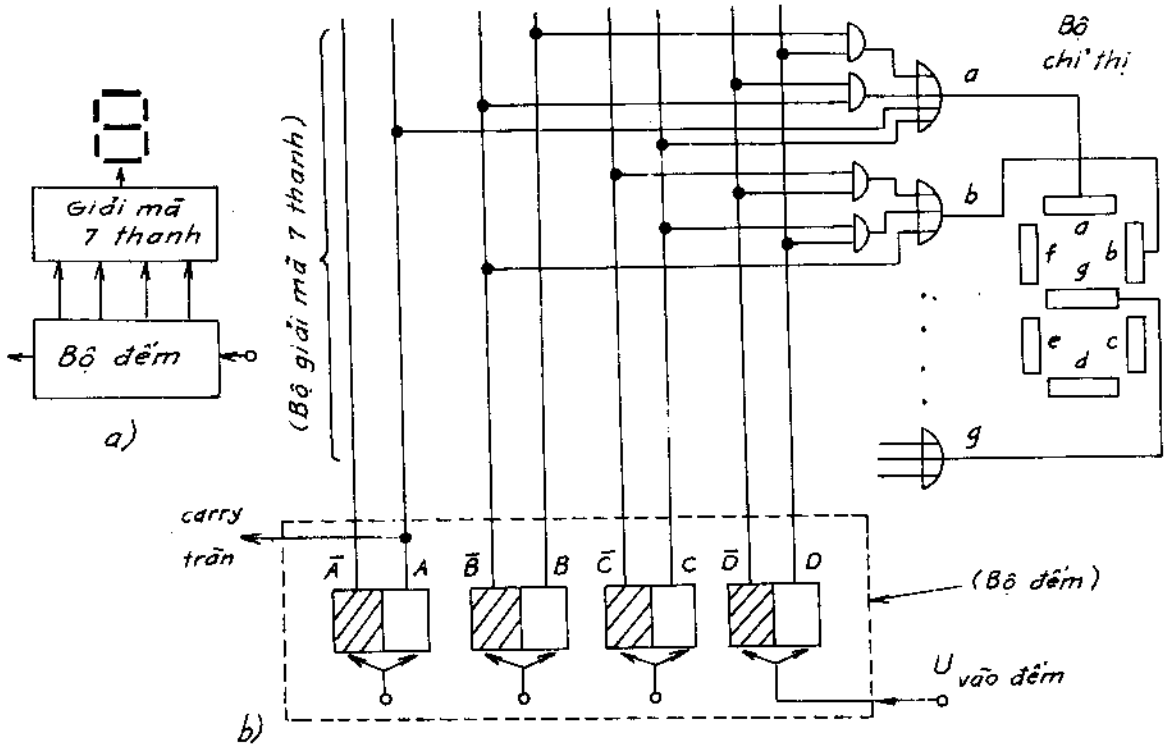
$$d = A + C\,\overline{D} + \overline{B}\,C + \overline{B}\,\overline{D} + B\,\overline{C}\,D$$

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

$$e = C\,\overline{D} + \overline{B}\,\overline{C}\,\overline{D}$$

$$g = A + C\,\overline{D} + \overline{B}\,C + B\,\overline{C}$$

Bước cuối cùng là dựa vào hệ phương trình điều khiển từng nhánh đã tìm được ta vẽ ra mạch điện của bộ giải mã 7 nhánh, mạch được chỉ ra trên hình 5.115.



Hình 5.115. Sơ đồ bộ giải mã 7 thanh.

Bộ giải mã nhánh chúng ta có thể lắp lấy được nó, nhưng trong thực tế bộ giải mã 7 nhánh đã được lắp ráp thành một con IC chuẩn có sẵn, đó là con SN7447.

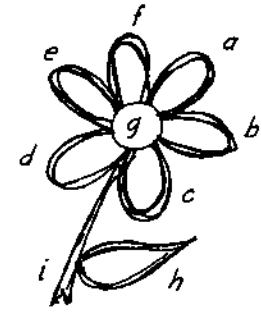
Dưới đây là ví dụ ứng dụng cách thiết kế bộ giải mã 7 thanh để thực hiện bộ giải mã điều khiển chỉ thị sáng tối cho một bông hoa với 9 thanh như sau. Ở đây mỗi thanh không còn là một khe sáng hình chữ nhật nữa, mà mỗi thanh có thể có hình dáng bất kỳ như cánh hoa, lá hay cuống hoa v.v... Đồng thời dựa vào đặc điểm cơ bản của bộ giải mã 7 thanh là có tính chất kế thừa các nét hay các nhánh có trước đó để tạo ra hình dáng thích hợp. Từ đó ta thiết kế ra được bộ giải mã điều khiển sự chỉ thị sáng tối cho bông hoa, bảng mô tả quan hệ giữa chỉ thị sáng tối của từng "nhánh" của bông hoa theo tín hiệu điều khiển có được từ bộ đếm được chỉ ra trên hình 5.116.

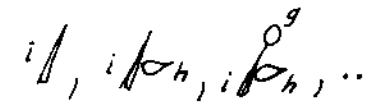
Sau khi có được bảng chỉ thị điều khiển một bông hoa sáng tối, bước tiếp theo ta ánh xạ từng nét vẽ (nhánh) vào bìa Kácnô để tìm ra được hệ phương trình điều khiển cho từng nét vẽ đó. Sau cùng là từ hệ phương trình điều khiển của từng nét vẽ ta lắp ráp ra "bộ giải mã chỉ thị hoa". Tiếp theo ta lắp ráp cụ thể mạch điện sẽ thực hiện được việc điều khiển sáng tối của một bông hoa theo ý ta. Các bước thiết kế ở đây hoàn toàn dựa theo các bước thiết kế "bộ giải mã 7 thanh" mà ta đã biết.

Bộ giải mã không chỉ giải mã số nhị phân thành cơ số đếm 10, không chỉ là bộ giải mã 7 thanh mà nó còn có thể giải mã ra các hình ảnh rồi giải mã ra cả tiếng nói nữa.

Phần trên chúng ta đã nói tới việc xây dựng bộ giải mã 7 thanh dùng để chỉ thị chữ số của cơ số đếm 10 cũng như các bộ giải mã dùng điều khiển các nét các hình dùng trong quảng cáo. Nhưng chưa nói tới bộ giải mã chỉ thị ra chữ viết từ chữ A đến chữ Z, mà cách thực hiện ra "bộ giải mã chữ" cũng dựa trên cách thiết kế của bộ giải mã 7 thanh. Cách thực hiện được chỉ ra trong phần tiếp theo.

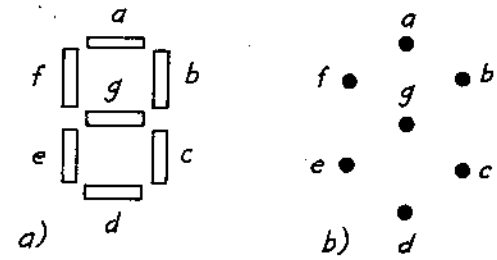
Hình chỉ thị	A	B	C	D	a	b	c	d	e	f	g	h	i
	Bộ đếm				9 nét vẽ								
	0	0	0	0	0	0	0	0	0	0	0	0	1
	0	0	0	1	0	0	0	0	0	0	0	1	1
	0	0	1	0	0	0	0	0	0	0	1	1	1
	0	0	1	1	1	0	0	0	0	0	1	1	1
	0	1	0	0	1	1	0	0	0	0	1	1	1
	0	1	0	1	1	1	1	0	0	0	1	1	1
	0	1	1	0	1	1	1	1	0	0	1	1	1
	1	0	0	0	1	1	1	1	1	0	1	1	1
	1	0	0	1	1	1	1	1	1	1	1	1	1





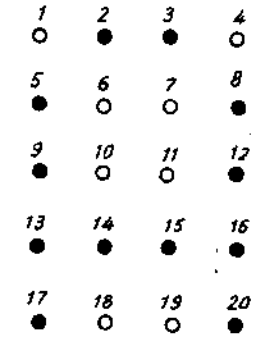
Hình 5.116. Bảng mô tả quan hệ điều khiển bông hoa.

"Bộ giải mã 7 thanh" thực chất phải nói đó chính là "bộ giải mã 7 điểm sáng", vì với mỗi một điểm sáng chúng ta phải nhìn qua một cái khe hẹp dài hình chữ nhật mà thôi, điều này được minh họa trên hình 5.117.



Hình 5.117. Bộ chỉ thị 7 thanh - 7 điểm sáng.

Vì bộ chỉ thị 7 thanh là chỉ thị 7 điểm sáng nên chúng ta có thể sắp xếp nhiều điểm sáng lại với nhau tạo ra một ma trận điểm sáng. Ở đây mỗi điểm sáng cũng giống như một nhánh của bộ giải mã nhánh nó sáng tối hoàn toàn độc lập với nhau. Chính vì là ma trận điểm sáng sắp xếp theo trật tự, đồng thời mỗi điểm lại sáng tối độc lập so với nhau, từ đó tạo cho ta khả năng mô tả ra chữ viết ra các ký tự của chữ viết. Điều đó được minh họa trên hình 5.118 với ma trận điểm đơn giản (5 hàng và 4 cột).



Hình 5.118. Ma trận điểm sáng mô tả chữ A.

Từ cách chỉ thị như hình vẽ ta sẽ có được bảng quan hệ điều khiển điểm sáng thành chữ A như trên hình 5.119.

Các chữ	A	B	C	D	E	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A	0	0	0	0	0	0	1	1	0	1	0	0	1	1	0	0	1	1	1	1	1	1	0	0	1
B	0	0	0	0	1																				
.			.																						
.			.																						
.			.																						
Z	1	1	0	0	0																				

Hình 5.119. Bảng quan hệ điều khiển chữ.

Với cách làm tương tự ta có thể tìm được quan hệ điều khiển từng điểm sáng để tạo ra các chữ khác trong bảng chữ cái. Sau đó tìm hệ phương trình điều khiển cho từng điểm sáng rồi lắp mạch xây dựng "bộ giải mã chữ".

Ở đây với một ma trận điểm sáng đơn giản (5 x 4) sẽ có những ký tự chữ viết khó mô tả cũng như không thể mô tả được trên ma trận đó. Muốn chỉ thị được các ký tự chữ viết cũng như các chữ được viết đẹp và tự nhiên hơn, thì số điểm sáng của ma trận điểm sáng chúng ta phải tăng thêm lên. Nếu điểm sáng càng nhiều thì khối lượng thiết kế "bộ giải mã chữ" sẽ nhiều lên theo, nhưng ta thu được chữ viết sẽ được chỉ thị đẹp hơn. Điều cần lưu ý ở đây là khi số điểm sáng (số nhánh) tăng lên thì số lượng các trigơ (Flíp Flốp) trong bộ đếm ta cũng phải tăng theo tương ứng. Ví dụ với 20 điểm sáng thì ta phải dùng bộ đếm với 5 trigơ (A B C D E) thì mới điều khiển không bị trùng tín hiệu.

Khi nói tới bộ giải mã chữ viết với các điểm sáng trên ma trận điểm sáng, từ ma trận điểm sáng ta liên hệ sang ma trận kim, tức là mỗi một điểm sáng sẽ ứng với một kim được đâm ra hay thụt vào. Khi nói tới ma trận kim mà các kim sẽ được điều khiển đâm ra hay thụt vào, điều đó cho phép ta xây dựng được đầu in kim hay các máy in kim. Lúc các kim ở ma trận kim được đâm ra đập vào mặt giấy theo tín hiệu điều khiển sẽ tạo ra các nét của ký tự chữ viết trên mặt giấy. Từ bộ giải mã chữ cho tới bộ giải mã in kim nguyên lý hoạt động của chúng hoàn toàn giống nhau, cách thiết kế ra chúng cũng trên cơ sở của bộ giải mã 7 thanh.

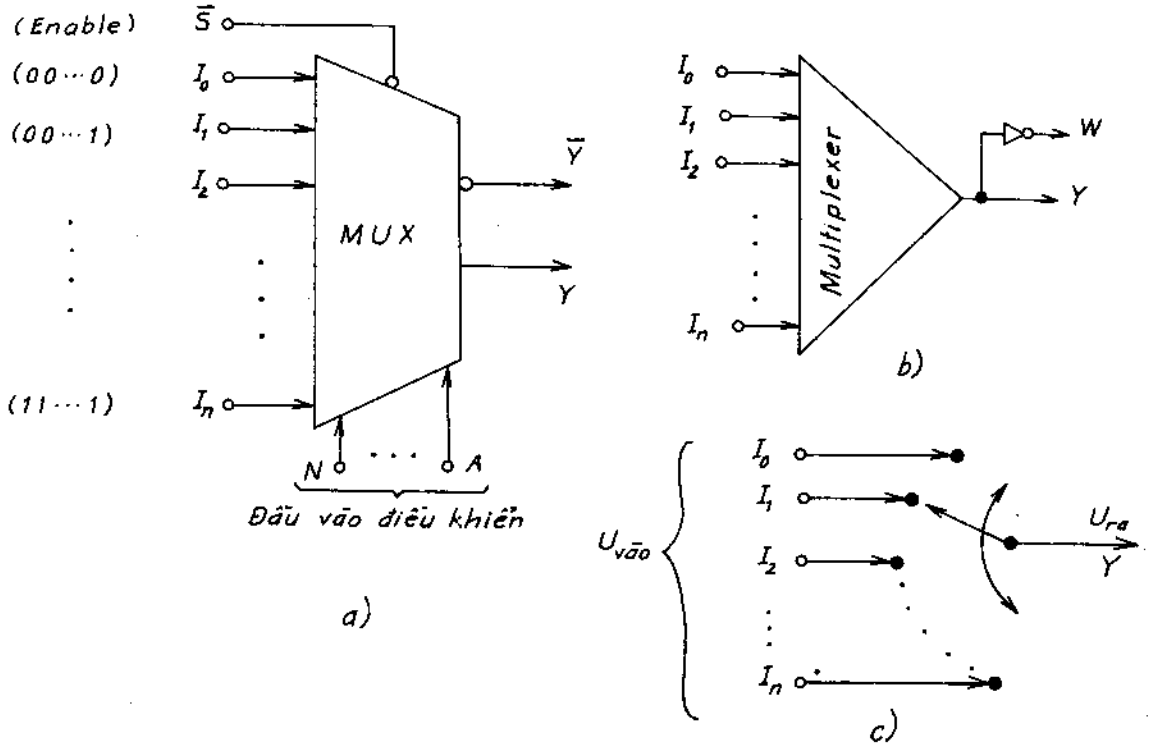
Sau khi đã có bộ giải mã ký tự chữ viết, rồi ma trận kim cho phép in ra chữ viết, thì việc trao đổi thông tin giữa người và máy chuyển sang một bước mới là hoạt động của máy có "nghĩa lý" hơn, và nhờ có bộ giải mã con người "dễ hiểu" máy hơn, điều khiển máy móc của hệ thống số dễ dàng hơn, hiệu quả hơn, đồng thời máy móc cũng thực hiện được nhiều chức năng hơn như trong vẽ hình hay in ấn. Đó chính là vai trò quan trọng của bộ giải mã dùng trong hệ thống số hiện nay.

5.7. BỘ CHỌN SỐ LIỆU (DATA SELECTOR) HAY BỘ DỒN KÊNH (MULTIPLEXER-MUX)

Đây là một mạch điện đặc biệt, nó liên quan tới bộ đếm và bộ giải mã, thực chất MUX chính là sự cải tiến của bộ giải mã và nó có thể thực hiện được nhiều chức năng khác nhau trên cùng một mạch (MUX). Nhiệm vụ của mạch MUX có thể thực hiện được là :

- Tạo hàm lôgic ở hệ tổ hợp.
- Dồn kênh để tạo ra một kênh thông tin mới (ghép kênh).
- Phân đường, cho phép từng luồng thông tin đi qua.

Nhiệm vụ chính của mạch MUX có thể được chỉ ra trên hình 5.120.



Hình 5.120. Nhiệm vụ của mạch MUX.

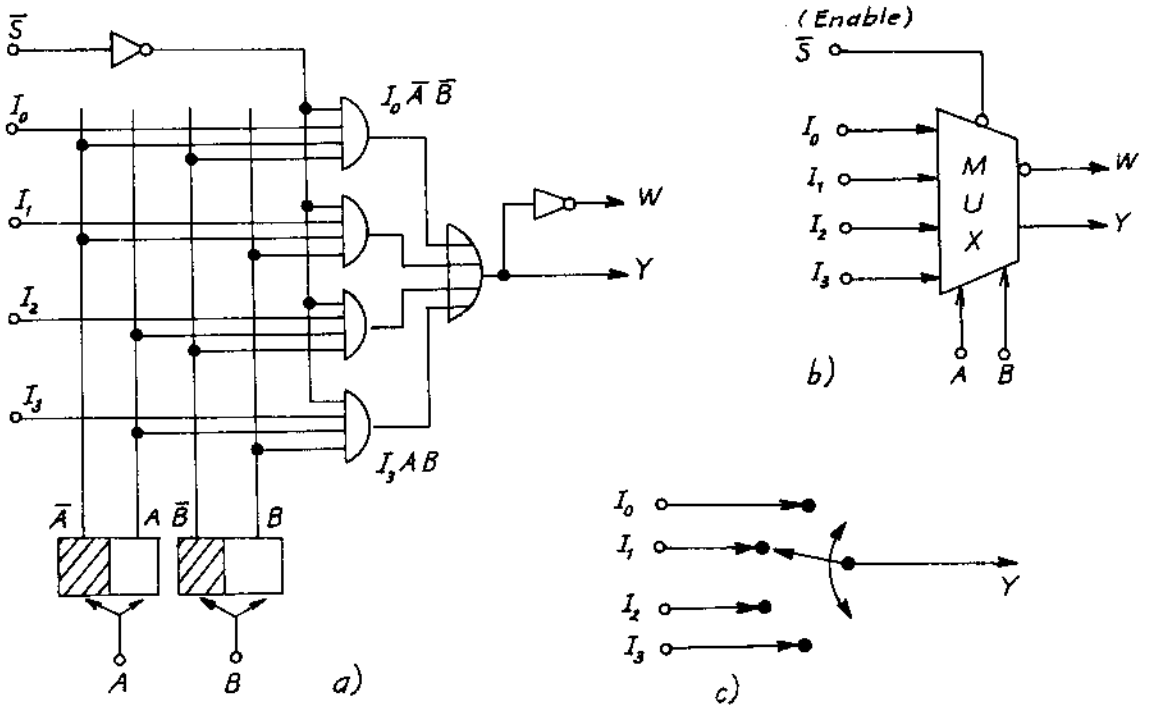
Mạch MUX hoạt động giống như một chuyển mạch cơ khí như trên hình 5.120,c. Khi chuyển mạch quay và tiếp xúc vào các đầu từ ($I_0 \div I_n$) thì tín hiệu tương ứng với đầu vào đó sẽ được truyền tới đầu ra Y. Người ta thay quá trình chuyển mạch cơ khí bằng một chuyển mạch điện tử MUX (multiplexer).

Mạch MUX bao gồm các đầu vào tín hiệu là ($I_0 \div I_n$) các đầu vào điều khiển từ (A ÷ N), với một đầu vào điều khiển chung cho phép là E (Enable) hay là đầu chọn vỏ S (Chip Selector), và hai đầu ra Y và đầu ra đảo $\overline{Y}$ (Y hay W). Ở đây đầu vào ($I_0 \div I_n$) và đầu điều khiển (A ÷ N) chúng có quan hệ tương đối so với nhau, tức là nếu chúng ta coi ($I_0 \div I_n$) là đầu vào thì đầu (A ÷ N) sẽ là đầu điều khiển và ngược lại.

Thông thường đầu vào (A ÷ N) là đầu điều khiển, các đầu điều khiển đó thường được lấy ra trên các đầu ra của các trigơ trong bộ tổng, bộ ghi dịch, thanh ghi tĩnh v.v... nhưng phổ thông nhất chúng thường được lấy ra trên các đầu ra của các trigơ của bộ đếm (đầu điều khiển thường là bộ đếm).

Việc quay chuyển mạch cơ khí lần lượt từ I_0 tới I_n sẽ tương ứng với việc bộ đếm sẽ đếm lần lượt từ giá trị (00...0) đến (11...1), quá trình đếm lần lượt sẽ "quét" hay cho phép tín hiệu vào từ I_0 đến I_n lần lượt sẽ được truyền ra đầu ra Y. Để hiểu được bản chất hoạt động của "chuyển mạch điện tử" này chúng ta sẽ xem xét nguyên lý làm việc của mạch MUX đơn giản nhất đó là mạch MUX 4 đầu vào (I_0, I_1, I_2, I_3) và hai đầu vào điều khiển (A, B), có tên là SN74150.

Sơ đồ nguyên lý của nó được chỉ ra trên hình 5.121.



Hình 5.121. Sơ đồ nguyên lý MUX 4 đầu vào.

Phương trình chức năng của mạch MUX 4 đầu vào là :

$$Y = \overline{S} . (I_0 \overline{A}\ \overline{B} + I_1 \overline{A}\ B + I_2 A \overline{B} + I_3 A B)$$

Nguyên lý làm việc của mạch MUX là : Nếu S = 1 thì mạch hoàn toàn không làm việc, dù có tín hiệu vào đầu (I_0 ÷ I_3) cũng như có tín hiệu vào điều khiển tại đầu điều khiển (A, B). Từ đó đầu S chính là đầu vào cho phép làm việc (Enable) của mạch.

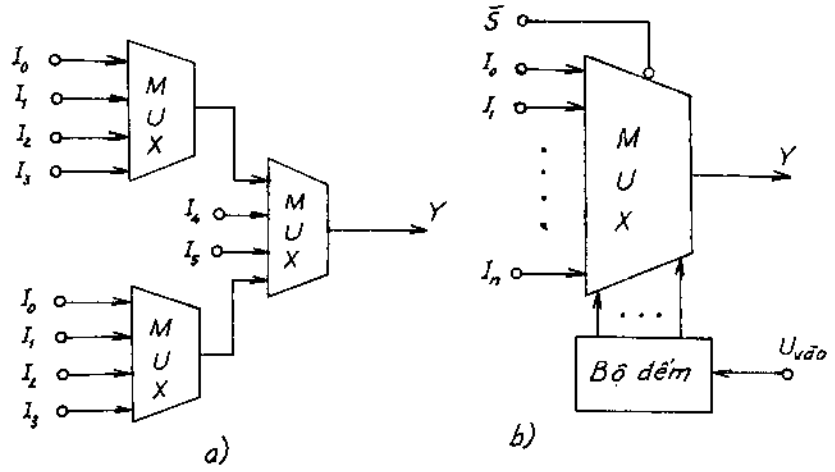
Khi ta cho S = 0 là đầu vào S được nối đất thì mạch MUX hoạt động bình thường. Giả sử đầu điều khiển ta cho (A B = 1 0). Khi A = 1 thì đầu ra A sẽ có thế cao (mức lôgic là 1), còn B = 0 thì đầu ra $\overline{B}$ sẽ có thế cao (mức lôgic $\overline{B}$ là 1). Từ kết quả đầu ra A và $\overline{B}$ có thế cao (mức lôgic 1) ta thấy chỉ có mạch VÀ ứng với đầu vào I_2 làm việc,

còn các mạch VÀ khác đều tắt, khi đó toàn bộ tín hiệu ta đưa vào đầu vào I_2 sẽ được truyền tới đầu ra Y. Qua nguyên lý làm việc nêu trên ta thấy hoạt động điều khiển của mạch MUX hoàn toàn giống nguyên lý làm việc của bộ giải mã loại số thứ tự đầu ra tương ứng với mã nhị phân đưa vào.

Cách làm tương tự khi ta cho đầu vào điều khiển (A, B) bằng các tổ hợp khác, thì đầu vào tín hiệu I tương ứng sẽ được truyền tới đầu ra Y theo tín hiệu điều khiển của (A, B).

Nếu ta tổ hợp các trigơ (A, B) của đầu điều khiển thành một bộ đếm nhị phân tăng dần, sau đó ta cho tín hiệu vào bộ đếm đó, thì bộ đếm sẽ lần lượt đếm từ tổ hợp đầu tiên (AB = 00) tới tổ hợp cuối cùng là (AB = 11), làm cho các tín hiệu vào lần lượt từ đầu vào I_0 tới đầu vào I_3 sẽ lần lượt được truyền ra tới đầu ra Y, kết quả là hoạt động chuyển mạch điều khiển như vậy ở (A, B) hoàn toàn tương ứng như chuyển mạch cơ khí được vặn để dịch chuyển lần lượt tiếp xúc từ đầu vào I_0 tới đầu vào I_3 của chuyển mạch cơ khí trên hình 5.121,c. Mạch MUX chính là chuyển mạch điện tử, nó được ứng dụng phổ biến nhất trong hệ thống tổng đài hiện nay.

Với cách làm tương tự ta có thể xây dựng được các mạch MUX với số lượng đầu vào lớn hơn, như MUX 8 đầu vào có tên là SN74151, rồi mạch MUX 16 đầu vào SN74153. Từ các mạch MUX cơ bản đó chúng ta có thể xây dựng được các mạch MUX có số đầu vào rất lớn bằng cách tổ hợp chúng lại. Điều đó được minh họa trên hình 5.122.



Hình 5.122. Cách tạo ra MUX với số đầu vào lớn.

Sau khi nắm được nguyên lý làm việc của mạch MUX ta đi tới phần ứng dụng trong thực tế của loại mạch này hay nói tới các chức năng mà mạch MUX có thể thực hiện được.

5.7.1. CHỨC NĂNG TẠO HÀM LÔGIC DÙNG MUX

Việc tạo hàm lôgic trong hệ tổ hợp (hệ không có nhớ), để có được một mạch điện thực hiện một hàm lôgic hay một chức năng cho trước, người ta dựa theo phương trình

toán học của hàm số đó, rồi dùng các mạch lôgic (AND, OR, NOT) lắp ráp mạch tạo ra hàm lôgic đó. Thay vì làm công việc như vậy ta có thể dùng MUX để tạo hàm mạch điện cho một chức năng lôgic hay một hàm lôgic bất kỳ, cách thực hiện sẽ được hiểu qua ví dụ sau.

Ví dụ : Thực hiện chức năng lôgic cho hàm số sau

$$f(x_1, x_2) = \bar{x}_1 x_2 + x_1 \bar{x}_2 = x_1 \oplus x_2$$

Đây là một hàm lôgic là hàm cộng môđun, hàm lôgic này ta có thể hoàn toàn dùng các mạch lôgic (AND, OR, NOT). Nhưng có thể dùng MUX với 2 đầu vào điều khiển để thực hiện (hay tạo ra hàm số cho trước) chức năng cũng như hàm số lôgic cho mạch điện này.

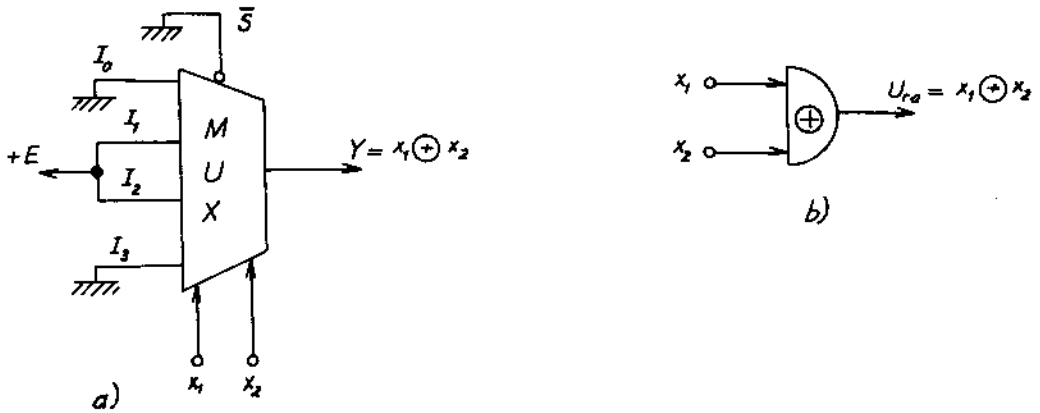
Muốn thực hiện mạch điện cho hàm lôgic ta dùng MUX 4 đầu vào ($I_0 \div I_3$) với hai đầu vào điều khiển là (A, B). Để thực hiện chức năng tạo hàm, lúc này ta coi hai đầu vào (A, B) là đầu vào tín hiệu, còn các đầu vào ($I_0 \div I_3$) là các đầu tổ hợp điều khiển. Dựa vào phương trình tổng quát chức năng của MUX 4 đầu vào là :

$$Y = \bar{S} . (I_0 \bar{A}\, \bar{B} + I_1 \bar{A}\, B + I_2 A \bar{B} + I_3 A B)$$

Hàm số ta cần có cần tạo ra (cần có mạch điện) là :

$$f(x_1, x_2) = \bar{x}_1 x_2 + x_1 \bar{x}_2$$

Nếu ta cho S = 0 để mạch MUX hoạt động bình thường và ta coi $A = x_1$ và $B = x_2$ là các tín hiệu vào, thì giá trị của hàm số $Y = f(x_1, x_2)$ chỉ còn phụ thuộc vào các đầu điều khiển ($I_0 \div I_3$). Để có được hàm số cần thiết cho trước (tạo hàm) ta chỉ cần cho $I_0 = I_3 = 0$ (nối đất) còn $I_1 = I_2 = 1$ (nối lên +E) là xong. Vậy mạch điện của bộ cộng môđun dùng MUX được chỉ ra ở trên hình 5.123.



Hình 5.123. Dùng MUX thực hiện mạch cộng môđun.

Với cách làm tương tự có thể thực hiện được các hàm lôgic khác nhau với hai đầu vào (x_1, x_2). Nếu muốn có hàm lôgic có số đầu vào lớn hơn là ba (x_1, x_2, x_3) hay bốn (x_1, x_2, x_3, x_4) ta phải dùng các mạch MUX có đầu vào điều khiển là 3 (A, B,C) hay

là 4 (A, B, C, D), chúng đều là các mạch MUX đã được chế tạo sẵn, rất dễ dàng sử dụng theo phương pháp thực hiện đã được chỉ ra ở trên.

Nhưng nếu với yêu cầu tạo ra hàm lôgic với lượng đầu vào rất lớn và quan hệ phức tạp bất kỳ, thì cách thực hiện không còn đơn giản như vậy nữa, mà phải thực hiện theo phương pháp tổ hợp các mạch MUX cơ bản có sẵn lại với nhau. Cách làm cụ thể dùng MUX 4 đầu tạo ra hàm lôgic bất kỳ có số lượng đầu vào lớn hơn với quan hệ toán học bất kỳ được minh họa như sau.

Ví dụ : Thực hiện hàm lôgic với chức năng theo hàm 4 biến sau :

$$f(x_1, x_2, x_3, x_4) = \bar{x}_1 x_2 \bar{x}_3 \bar{x}_4 + \bar{x}_1 x_2 x_3 x_4 + x_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 + x_1 \bar{x}_2 x_3 x_4$$

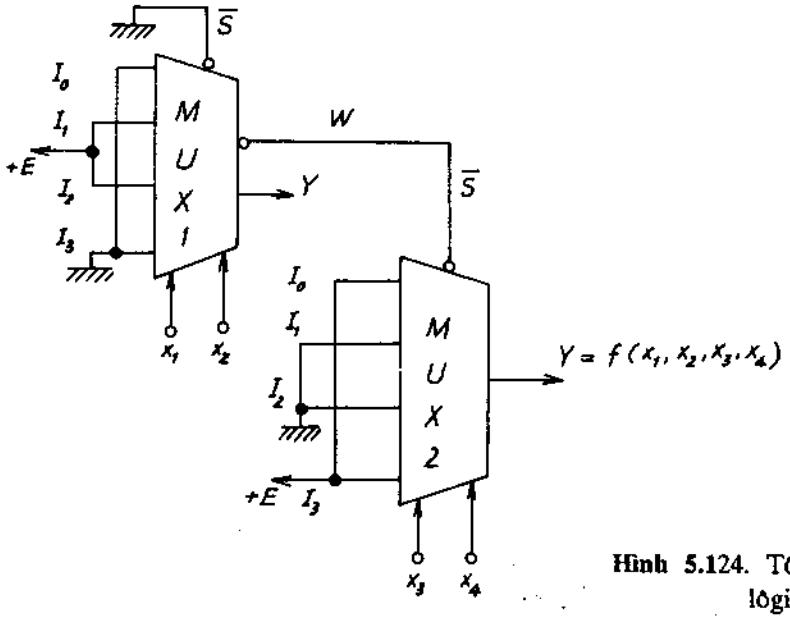
Hàm 4 biến này ta có thể hoàn toàn có thể dùng các mạch lôgic cơ bản hay dùng MUX 4 đầu điều khiển (A, B, C, D) thực hiện dễ dàng. Nhưng nếu ta thực hiện hàm số đó dùng MUX 2 đầu vào điều khiển (A, B) theo phương pháp tổ hợp, thì mạch điện của nó được tạo ra theo cách như sau :

Trước hết ta biến đổi hàm số cần tạo ra như sau :

$$f(x_1, x_2, x_3, x_4) = \bar{x}_1 x_2 \bar{x}_3 \bar{x}_4 + \bar{x}_1 x_2 x_3 x_4 + x_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 + x_1 \bar{x}_2 x_3 x_4 =$$

$$= (\bar{x}_1 x_2 + x_1 \bar{x}_2) . (\bar{x}_3 \bar{x}_4 + x_3 x_4) = Y$$

Mỗi một ngoặc đơn của hàm số ta dùng một mạch MUX 4 đầu vào ($I_0 \div I_3$) thực hiện, sau đó tổ hợp chúng lại sẽ tạo ra được hàm lôgic đã cho. Mạch điện cụ thể được minh họa trên hình 5.124.

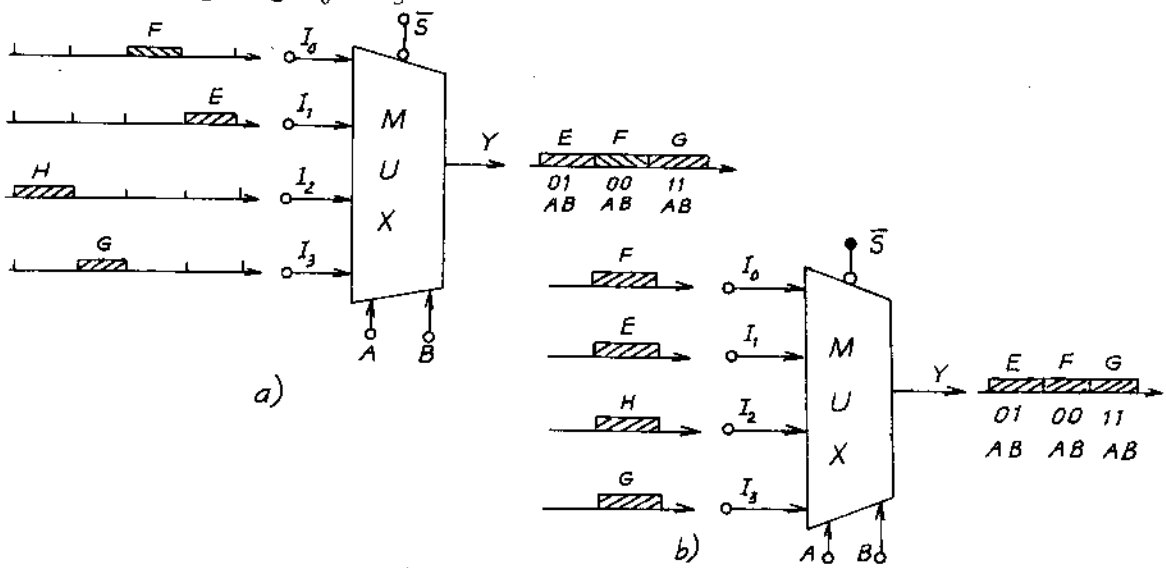


Hình 5.124. Tổ hợp MUX tạo hàm lôgic lớn hơn.

Với cách làm tương tự ta có thể tổ hợp (thực hiện) được các hàm lôgic bất kỳ bằng tổ hợp các mạch MUX đã được chế tạo sẵn.

5.7.2. CHỨC NĂNG DỒN KÊNH HAY PHÂN ĐƯỜNG DÙNG MUX

Đây là một trong những chức năng quan trọng mà mạch MUX có thể thực hiện được. Chức năng dồn kênh, ghép kênh hay chức năng phân đường (phân luồng) tín hiệu thực chất chỉ là một, vì mạch phân đường chính là mạch dồn kênh đặc biệt, khi ta dồn kênh từ đầu đến cuối (toàn bộ tín hiệu của một kênh) được truyền đến cuối (toàn bộ tín hiệu của một kênh) được truyền đến đầu ra Y. Đối với chức năng dồn kênh hay phân đường thì các đầu vào ($I_0 \div I_n$) là đường tín hiệu vào, thông thường nó có thể lấy từ độ ghi dịch, tín hiệu hình hay tín hiệu thoại ở dạng (0, 1) đó chính là các thuê bao được truyền đến, hay bất kỳ một tín hiệu dịch truyền theo thời gian nào được đưa vào các đầu vào ($I_0 \div I_n$). Còn đầu vào điều khiển (A ÷ N) đó là các đầu chọn, nó cho phép chọn bit thông tin nào ở các đầu vào I được truyền tới đầu ra Y, từ đó cho phép ta tạo được một kênh thông tin mới, đó chính là chức năng dồn kênh. Cũng từ chức năng này cho phép ta chuyển một tín hiệu song song thành một tín hiệu nối tiếp, đó là điều rất quan trọng trong truyền thông tin số. Cho phép thực hiện chuyển tất cả các kênh thông tin vào song song thành một kênh thông tin nối tiếp truyền vào một đường truyền thông tin chung đó là cáp quang. Quá trình thực hiện dồn kênh tạo ra một kênh thông tin riêng nhờ đầu vào điều khiển (A ÷ N) được minh họa trên hình 5.125 bằng mạch MUX 4 đầu vào song song ($I_0 \div I_3$).



Hình 5.125. Ứng dụng MUX để tạo một kênh tín hiệu.

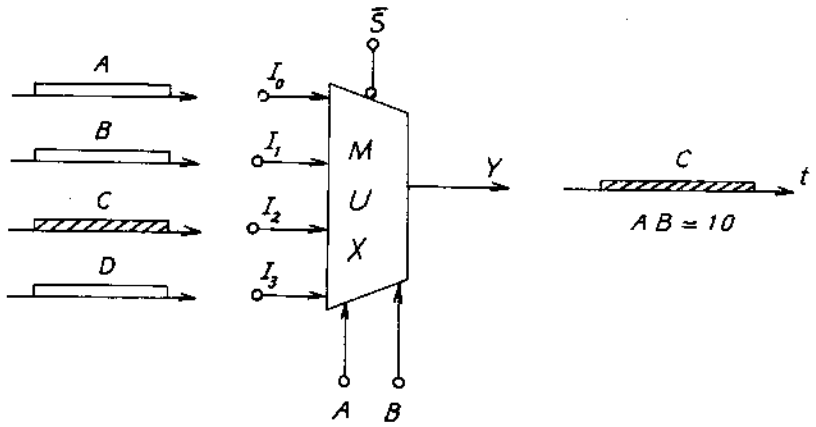
Kênh thông tin riêng (EFG) thường được chúng ta dự kiến trước, dựa vào nguyên lý của mạch MUX có thể dùng đầu vào điều khiển (A, B) để chọn lấy tín hiệu thích hợp từ các đầu vào song song ($I_0 \div I_3$) truyền tới đầu ra Y để tạo nên một kênh truyền hay một tín hiệu truyền riêng cần thiết theo ý của ta. Thông thường tín hiệu đến các đầu vào ($I_0 \div I_3$) có thể xuất hiện ngẫu nhiên theo thời gian (hình 5.125,a) và ta có thể dùng đầu vào điều khiển (A, B) để tạo ra tín hiệu kênh truyền thích hợp (E F G), còn trường hợp tín hiệu vào trên các đầu vào ($I_0 \div I_3$) xuất hiện đồng thời, khi đó cũng có

thể dựa vào đầu vào điều khiển (A, B) để cho tín hiệu vào được lần lượt truyền tới đầu ra Y theo yêu cầu (E F G) mà ta dự kiến trước (hình 5.125,b). Cách thực hiện điều khiển như vậy cho phép ta tạo ra tín hiệu nối tiếp từ các tín hiệu vào song song, cũng từ khả năng điều khiển mạch MUX như vậy cho chúng ta khả năng ứng dụng của nó rất phong phú trong thực tế.

Ứng dụng nhiều nhất của MUX là trong máy vi tính. Khi đọc quét bộ nhớ, để truyền và hiển thị bộ nhớ lên màn hình máy vi tính cũng có thể dùng MUX. Trong truyền hình số khi ta muốn tạo ra các kỹ xảo hình ảnh, hay thực hiện ghép kênh dòng truyền tải của truyền hình số qua mạng cáp. Khi tín hiệu hình ta thu được từ nhiều nguồn khác nhau. Chúng cần phải tạo thành một đường tín hiệu, thành một dòng truyền tải để truyền vào đường cáp quang ta phải sử dụng mạch MUX.

Trong tổng đài mạch MUX cũng được sử dụng rất nhiều, thậm chí không thể thiếu được khi xây dựng tổng đài. Dùng MUX cho phép ta thực hiện việc ghép kênh để tăng tốc độ truyền, tăng khả năng hoạt động của tuyến cáp quang. Tất cả các thuê bao các loại tín hiệu được đưa vào tổng đài đều được dồn kênh, tạo ra một tín hiệu duy nhất truyền vào sợi cáp quang nối giữa các thành phố hay các quốc gia. Qua đó chúng ta thấy ứng dụng của mạch MUX là rất lớn và rất quan trọng trong cuộc sống của con người.

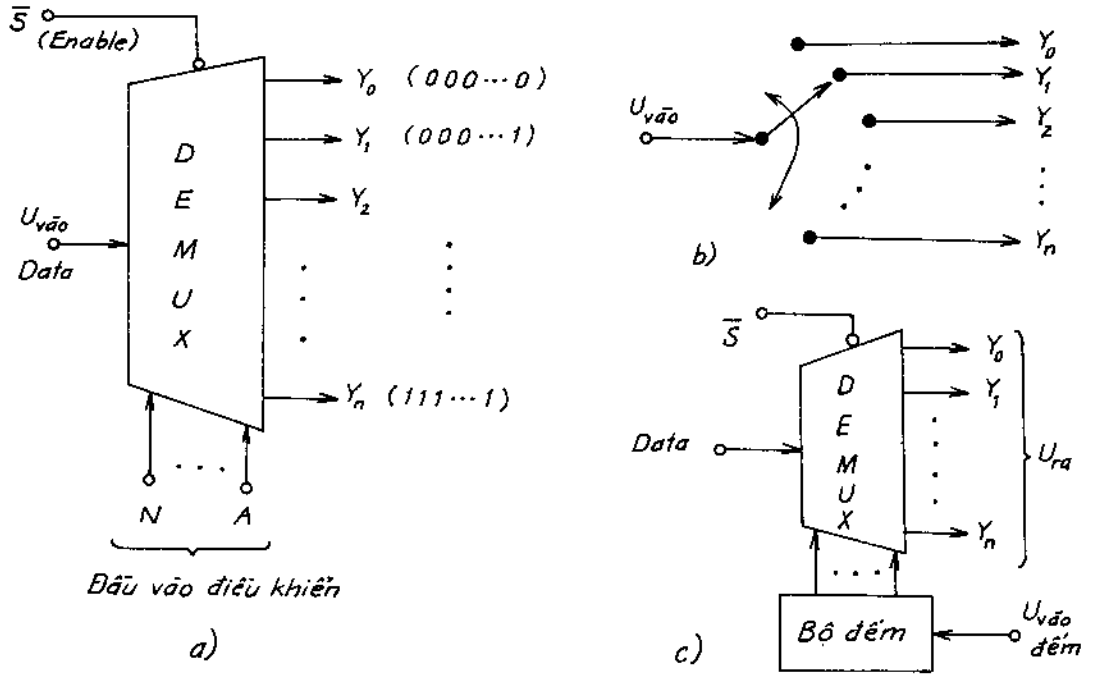
Chức năng phân đường dùng MUX, đây là chức năng đơn giản nhất và dễ hiểu nhất của mạch MUX. Chức năng phân đường cũng chỉ là một dạng đặc biệt của chức năng dồn kênh mà thôi, vì phân đường chính là chức năng dồn kênh nhưng dồn kênh từ đầu đến cuối hay cho đi qua toàn bộ thông tin xuất hiện trên một đầu vào I_i dưới sự điều hành cho qua của tín hiệu điều khiển (A ÷ N). Minh họa hoạt động của MUX trong chức năng phân đường được chỉ ra trên hình 5.126.



Hình 5.126. Dùng MUX cho một đường thông tin truyền qua.

§5.8. BỘ PHÂN KÊNH DEMUX

Mạch DEMUX là một mạch đặc biệt được tạo ra từ sự kết hợp giữa bộ giải mã, bộ đếm. Nhiệm vụ chính của mạch là phân kênh (tách kênh), hay từ một tín hiệu chung phân kênh truyền tới một đầu ra theo yêu cầu dựa vào tín hiệu điều khiển. Chức năng của mạch DEMUX được chỉ ra trên hình 5.127.



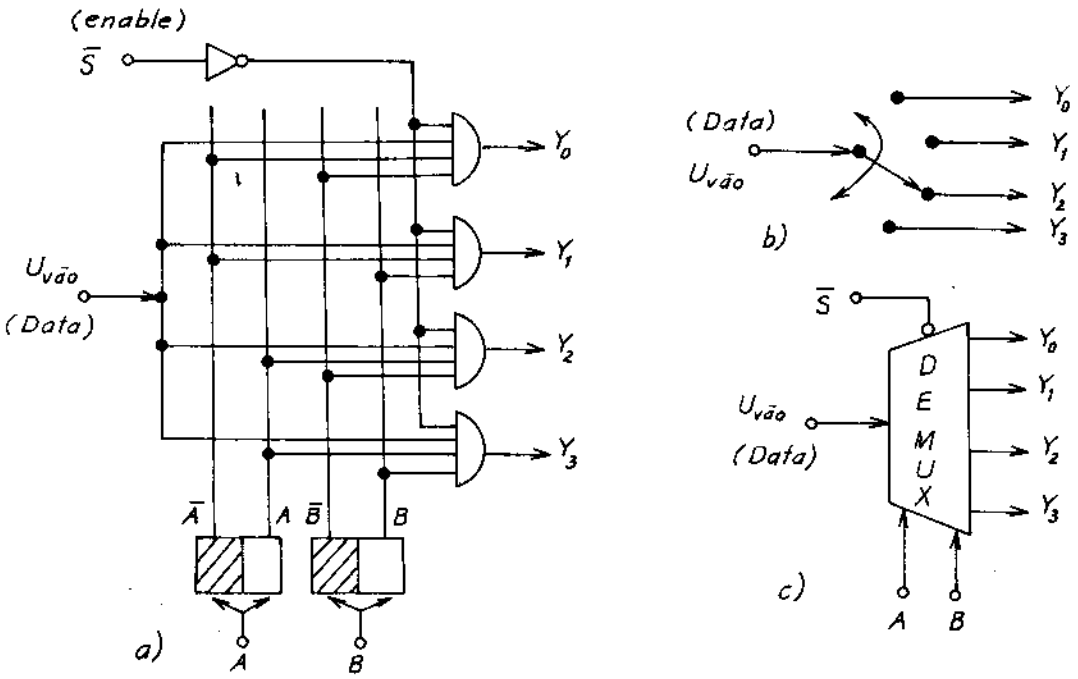
Hình 5.127. Chức năng của bộ phân kênh DEMUX.

Mạch MEMUX có một đầu vào $U_{vào}$ (số liệu) hay các tín hiệu truyền đến, có n đầu vào điều khiển (A ÷ N) tương ứng sẽ có 2^n đầu ra (Y_0, Y_1, ..., Y_n).

Mạch DEMUX hoạt động giống như một chuyển mạch cơ khí hình 5.127,b, tiếp điểm lần lượt tiếp xúc với các đầu ra Y_i. Khi tiếp điểm chạm vào đầu ra nào thì tín hiệu vào ($V_{vào}$ hay Data) sẽ được truyền tới đầu ra đó. Trên hình 5.128 là mạch điện nguyên lý của DEMUX.

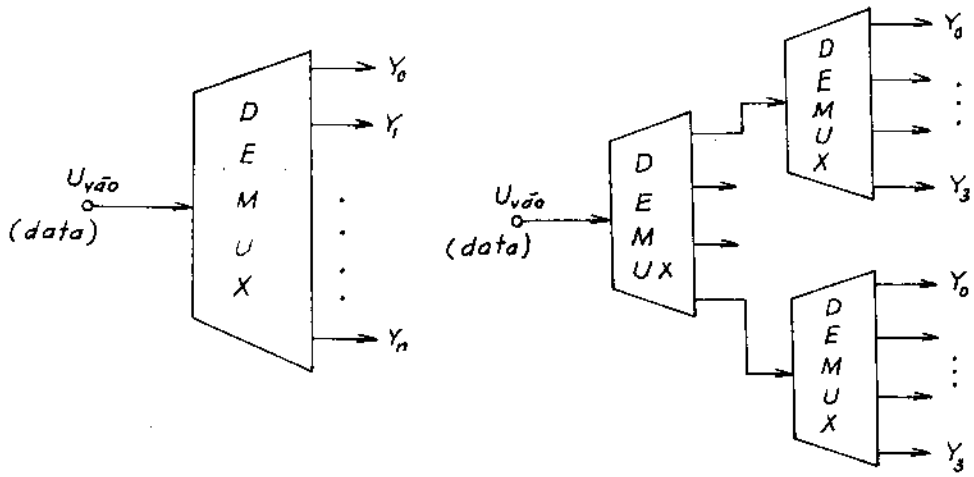
Nguyên lý làm việc của DEMUX giống như của bộ giải mã. Khi cho $\overline{S}$ = 0 thì mạch hoạt động bình thường nếu ta cho đầu điều khiển (A B = 1 0) thì đầu ra A và $\overline{B}$ có thể cao ứng với mức lôgic 1, khi đó tín hiệu vào ($V_{vào}$) sẽ được truyền tới đầu ra Y_2 tương ứng, còn các đầu ra khác các mạch Và sẽ không làm việc nên không có tín hiệu ra.

Như vậy chỉ cần cho tín hiệu điều khiển vào đầu vào (A, B) sẽ cho phép truyền tín hiệu ra ở đầu ra tương ứng, điều này rất tiện lợi khi điều khiển tín hiệu nhận được truyền tới từng thuê bao theo sự điều khiển. Ở đây (A, B) là các trigơ có thể lấy từ các mạch điện như thanh ghi, bộ ghi dịch, bộ tổng v.v... Nhưng nếu các trigơ (A, B) ta tổ



Hình 5.128. Sơ đồ nguyên lý của mạch DEMUX 4 đầu ra.

hợp thành một bộ đếm nhị phân tăng dần, sau đó cho thực hiện đếm từ (AB = 00) đến AB = 11), tín hiệu điều khiển đó sẽ làm cho tín hiệu vào xuất hiện ở đầu ra từ Y_0 đến Y_3 một cách lần lượt. Hoạt động đó của mạch DEMUX hoàn toàn giống hoạt động của chuyển mạch cơ khí hình 5.128,b. Từ đó từ có thể gọi mạch DEMUX là chuyển mạch điện tử. Chuyển mạch điện tử hơn hẳn chuyển mạch cơ khí là có thể thực hiện chuyển mạch với tốc độ rất cao hàng triệu lần chuyển đổi trong một giây, điều mà chuyển mạch cơ khí hoàn toàn không thể làm được. Đồng thời chuyển mạch điện tử không có tiếp xúc cơ khí nên không tạo ra tia lửa điện gây nhiễu cho mạch, và có thể chuyển mạch với hoạt động ngẫu nhiên tiếp xúc theo điều khiển, mà không cần phải lần lượt như chuyển mạch cơ khí.



Hình 5.129. Tăng số lượng đầu ra.

Ứng dụng của mạch DEMUX cũng rất phong phú, nhưng chủ yếu dùng trong đầu ra của tổng đài. Chuyển mạch điện tử MUX và DEMUX là hai chuyển mạch cơ bản thiết yếu của tổng đài, từ khâu nhận tín hiệu vào cho tới phân phối tín hiệu ở đầu ra, mà không một tổng đài nào không có chúng, đó là ứng dụng quan trọng trong cuộc sống của hai mạch này.

Mạch DEMUX cũng được sản xuất theo tiêu chuẩn với số lượng đầu ra hữu hạn. Vậy muốn tăng số lượng đầu ra của mạch DEMUX chúng ta có thể thực hiện theo hình 5.129.

§5.9. BỘ TỔNG

Bộ tổng là một mạch điện thực hiện phép toán cộng nhị phân của hai con số nhị phân A và B, nó là một mạch điện thực hiện một phép toán số học thông thường, đó là phép toán cộng (cũng như phép toán trừ), mà từ đó có thể thực hiện được các phép toán nhân và chia. Điều đó cho phép chúng ta xây dựng được bộ số học ALU (Arithmetic Logic Unit) trong máy tính số. Bộ tổng cũng liên hệ rất chặt chẽ với các hệ thống số và là thành phần không thể thiếu trong cấu tạo của các bộ vi xử lý μP.

Muốn xây dựng được mạch điện chúng ta nhắc lại thao tác chính của một phép cộng nhị phân có hai con số A và B :

$$A = (a_n, a_{n-1}, \ldots, a_2, a_1)$$

$$B = (b_n, b_{n-1}, \ldots, b_2, b_1)$$

Phép toán cộng của chúng được thực hiện là :

$$\begin{array}{r} A \\ +\ B \\ \hline C \end{array} = \begin{array}{r} (a_n, a_{n-1}, \ldots, a_2, a_1) \\ +\ (b_n, b_{n-1}, \ldots, b_2, b_1) \\ \hline (C_n, C_{n-1}, \ldots, C_2, C_1) = \end{array} \begin{array}{r} a_n \\ +\ b_n \\ \hline C_n \end{array} \begin{array}{r} a_{n-1} \cdots \\ +\ b_{n-1} \cdots \\ \hline C_{n-1} \cdots \end{array} \begin{array}{r} a_2 \\ +\ b_2 \\ \hline C_2 \end{array} \begin{array}{r} a_1 \\ +\ b_1 \\ \hline C_1 \end{array} \begin{array}{r} 1\ 1\ 1 \\ 1\ 0\ 1\ 1 \\ +\ 1\ 1\ 1\ 1 \\ \hline 1\ 1\ 0\ 1\ 0 \end{array}$$

Do A và B là các số nhị phân nên $a_i \in (0, 1)$ và $b_i \in (0, 1)$, khi cộng là cộng từng cột số kèm theo số nhớ.

Trước hết ta xét tới cột tổng đầu tiên $(a_1 + b_1)$. Đây là một cột cộng không bình thường, và nó là một cột cộng đặc biệt, vì đó là cột tổng đầu tiên của hai số nhị phân. Do là cột cộng đầu tiên của hai số nhị phân nên đó là một cột tổng chưa "hoàn chỉnh", vì nó không có số nhớ từ cột bên cạnh đưa sang. Chính vì vậy nó là cột cộng hai số nhị phân không có nhớ, là một quan hệ cộng chưa hoàn chỉnh nên được gọi là "bán tổng". Hiện tượng này chỉ có ở cột cộng đầu tiên.

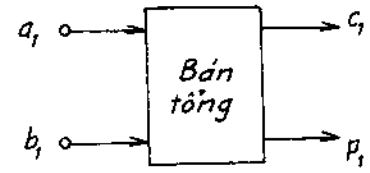
5.9.1. BỘ BÁN TỔNG

Bộ bán tổng có bảng chức năng và mạch điện được chỉ ra trên hình 5.130.

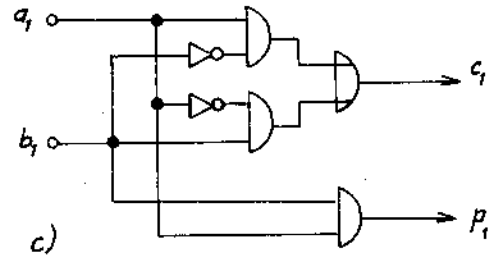
a_1	b_1	c_1	p_1
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

a)

$$c_1 = \bar{a}_1 b_1 + a_1 \bar{b}_1$$
$$p_1 = a_1 b_1$$



b)



c)

Hình 5.130. Bộ bán tổng.

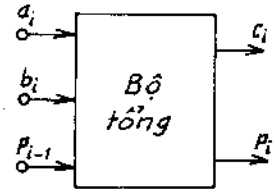
Bộ bán tổng có hai đầu vào (a_1, b_1) đó là giá trị nhị phân ở cột đầu tiên của phép toán cộng, cùng với hai đầu ra (c_1 và p_1), ở đây c_1 là đầu ra cho kết quả tổng của cột đầu tiên và p_1 là đầu ra nhớ (carry) dùng để chuyển giá trị nhớ (chuyển) của cột đầu tiên sang cột cộng bên cạnh (a_2 và b_2).

Bắt đầu từ cột tổng hai số (a_2, b_2) cho tới cột cộng của cặp số cuối cùng (a_n, b_n), các cột số của bộ tổng còn được cộng thêm giá trị của số nhớ có được từ cột bên trước đó chuyển sang. Đến lúc này bộ tổng có đầy đủ các giá trị đưa vào (các tín hiệu vào) cho nên được gọi là bộ tổng hay toàn tổng.

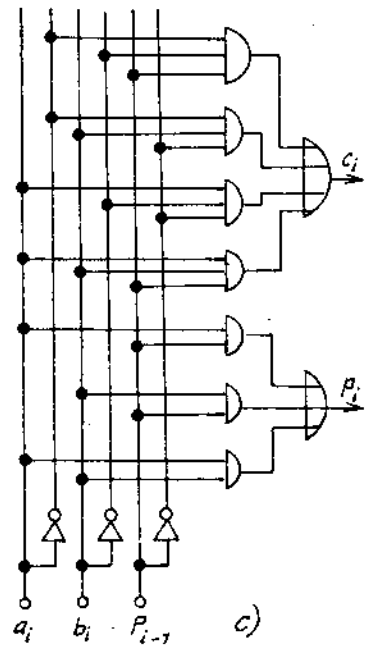
5.9.2. BỘ TỔNG MỘT CỘT SỐ BẤT KỲ

a_i	b_i	p_{i-1}	c_i	p_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

a)



b)



c)

$$C_i = \bar{a}_i \bar{b}_i P_{i-1} + \bar{a}_i b_i \bar{P}_{i-1} + a_i \bar{b}_i \bar{P}_{i-1} + a_i b_i P_{i-1}$$

$$P_i = a_i P_{i-1} + b_i P_{i-1} + a_i b_i$$

Hình 5.131. Bộ tổng một cột số.

Bộ tổng ở cột số bất kỳ đầy đủ bao gồm có ba đầu tín hiệu vào đó là : các giá trị số liệu cần cộng (a_i và b_i) và một đầu tín hiệu vào ứng với giá trị nhớ của cột số trước đó chuyển sang (p_{i-1}). Có hai đầu ra c_i cho kết quả tổng và p_i là số nhớ của cột đó chuyển tới cột tiếp theo. Bảng chức năng và sơ đồ nguyên lý của bộ tổng được chỉ ra trên hình 5.131.

Sau khi có được mạch điện của bộ tổng và bộ bán tổng chúng ta có thể xây dựng mạch điện cho một bộ tổng của hai con số nhị phân (A, B) với độ dài của con số hay số lượng cột số là bất kỳ. Do mạch điện của bộ tổng dùng toàn mạch tổ hợp nếu có thể gọi là bộ tổng liên hợp hay bộ cộng.

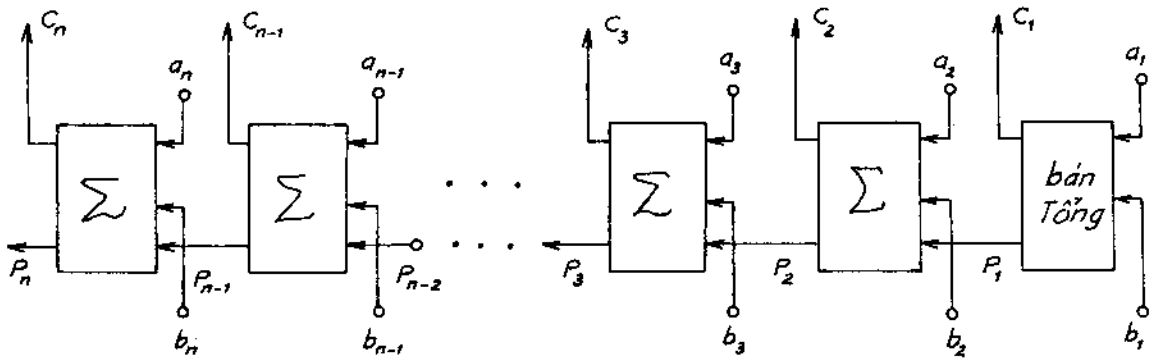
5.9.3. BỘ CỘNG SONG SONG NHIỀU BIT

Dựa vào mạch điện của bộ tổng và bán tổng có thể xây dựng được mạch điện thực hiện chức năng tổng và cho ra kết quả tổng của hai con số nhị phân (A, B).

$$A = (a_n, a_{n-1}, \cdots, a_2, a_1)$$

$$B = (b_n, b_{n-1}, \cdots, b_2, b_1)$$

Mạch điện của bộ cộng song song nhiều bit của hai con số nhị phân (A, B) được mô tả trên hình 5.132.



Hình 5.132. Mạch điện bộ cộng nhiều bit.

Sau khi có mạch điện của bộ tổng việc tính toán kết quả cộng của hai số nhị phân bất kỳ (A và B) rất đơn giản, vì ta chỉ cần cho số nhị phân của con số A vào các đầu vào a_i tương ứng và cho số nhị phân của con số B vào các đầu b_i ($1 \leqslant i \leqslant n$) tương ứng trong bộ cộng, ngay sau đó ta sẽ nhận được kết quả tổng của hai số (A, B) trên các đầu ra từ (C_1 tới C_n) :

$$\begin{array}{r} A \\ + \; B \\ \hline C \end{array} \quad = \quad \begin{array}{r} (a_n, a_{n-1}, \cdots, a_2, a_1) \\ + \; (b_n, b_{n-1}, \cdots, b_2, b_1) \\ \hline (C_n, C_{n-1}, \cdots, C_2, C_1) \end{array}$$

Có một điều cần lưu ý là ở cột cuối cùng ($a_n + b_n$) nếu có số nhớ hay kết quả tổng tràn ô, khi đó trên đầu nhớ tương ứng (p_n) sẽ xuất hiện tín hiệu nhớ sang cột bên cạnh. Ở cột bên cạnh đó sẽ có một trigơ ghi giữ lại kết quả tràn ô cuối cùng ứng với cột tổng

thứ n. Trigơ cuối cùng đó có thể là trigơ dấu của kết quả tổng hay ghi giữ bit cuối của kết quả tổng. Độ dài của con số nhị phân A và B tương ứng với các mạch tổng và bán tổng được sử dụng.

Trong thực tế giá trị của con số nhị phân A và con số nhị phân B ta có thể lấy ra từ một ô nhớ hay một thanh ghi số liệu. Đơn giản hơn là ta có thể tự tạo ra giá trị số nhị phân A cũng như B qua bộ phím bấm. Khi ta bấm phím một số, qua bộ tạo mã, đầu ra của bộ tạo mã sẽ cho ra số nhị phân tương ứng của số A cũng như của con số B cần thực hiện cộng, tín hiệu đó sẽ được đưa vào các cột số a_i và b_i của bộ tổng và đầu ra c_i sẽ cho ra kết quả cuối cùng. Như vậy các đầu ra từ (c_1 đến c_n) ta cho chúng đi tiếp qua bộ giải mã 7 thanh, thì kết quả tổng đó sẽ được cho ra dưới dạng con số của cơ đếm 10. Kết quả là ta cho vào hai con số A và B từ bàn phím, sẽ nhận được kết quả tổng của chúng ngay sau đó bằng con số được chỉ thị dưới dạng cơ đếm 10. Đó là thao tác đơn giản cộng hai số ở máy tính nhỏ bấm bằng tay mà ta vẫn sử dụng. Muốn thực hiện cộng được con số lớn có nhiều cột số, ta có thể dùng mã (2 – 10) để lắp ráp ra mạch thực hiện, nguyên lý cách thực hiện đã được nêu ra ở phần mã.

Để thu được kết quả cộng của hai số nhị phân A và B ngoài phương pháp dùng mạch tổ hợp, chúng ta còn có cách khác nữa là sử dụng phần tử nhớ (các trigơ) để xây dựng bộ tổng. Cách thực hiện cũng rất đơn giản, vì ta biết số quá trình đếm với kết quả cộng là một. Muốn thực hiện được kết quả tổng của hai số nhị phân A và B, ta chỉ cần cho vào "bộ đếm A" giá trị của con số A cần phải cộng, rồi cho vào "bộ đếm B" giá trị của con số B. Sau đó "bộ đếm A" thì ta thực hiện đếm tăng (đếm cộng) còn "bộ đếm B" thì thực hiện đếm giảm (đếm trừ). Mỗi lần "bộ đếm B" giảm đi một giá trị thì "bộ đếm A" lại đếm tăng (cộng thêm) một giá trị. Quá trình đó cứ thực hiện mãi cho đến khi "bộ đếm B" cho kết quả là zero (không còn gì) thì dừng quá trình đếm. Khi đó kết quả đếm được lưu lại trên 'bộ đếm A" sẽ là kết quả tổng của hai con số nhị phân A và B. Đây là bộ tổng tích lũy.

Bộ tổng có thể thực hiện bằng nhiều cách và có thể lắp ráp bằng mạch điện để thực hiện.

§5.10. BỘ TRỪ

Bộ trừ hay mạch điện thực hiện phép toán trừ giữa hai số nhị phân thực hiện cũng đơn giản, có nhiều cách làm và cách thực hiện khác nhau. Có thể dùng bộ tổng để tạo ra bộ trừ thông qua mã bù, cũng có thể thực hiện phép toán hiệu thông qua các bộ đếm. Cũng như có thể lắp ráp mạch điện thực hiện chức năng hiện theo bảng chân lý chỉ ra trên hình 5.133.

Cách xây dựng mạch của bộ trừ hoàn toàn giống như cách xây dựng mạch cho bộ tổng, chỉ có một điều lưu ý là tín hiệu P_i ở bộ tổng đó là tín hiệu nhớ 1 sang cột số bên cạnh, còn trong bộ trừ thì tín hiệu P_i ở chính là tín hiệu trả lại 1 cho cột bên cạnh nếu số bị trừ nhỏ quá mà ta phải "vay" theo nguyên lý làm phép toán trừ. Từ đó ta có thể lắp ráp ra mạch điện thực hiện phép toán trừ với độ dài bit số là bất kỳ.

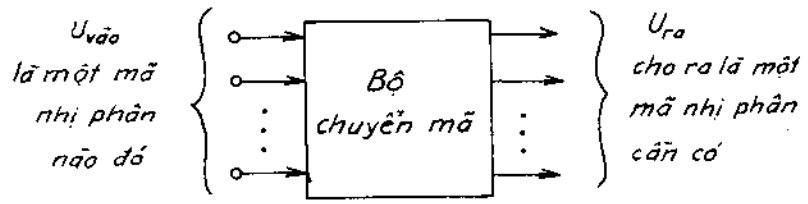
a_1	b_1	C_1	P_1
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

a_i	b_i	P_{i-1}	C_1	p_i
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Hình 5.133. Bảng thật của phép hiệu.

§5.11. BỘ CHUYỂN MÃ

Bộ chuyển mã là một mạch điện được sử dụng không nhiều và ít phổ biến trong hệ thống số : chúng được dùng khi phải chuyển đổi mã máy hay mã nhị phân bất kỳ thành mã nhị phân quen thuộc, điều này cho phép các nhà kỹ thuật tạo ra các mạch thực hiện quan hệ tương thích trong các hệ thống máy tính cũng như trong hệ thống số. Nhiệm vụ của bộ chuyển mã là chuyển đổi từ một mã nhị phân này thành một mã nhị phân khác. Nhiệm vụ của bộ chuyển đổi mã được chỉ ra trên hình 5.134.



Hình 5.134. Bộ chuyển mã nhị phân.

Ta cần lưu ý là bộ chuyển mã hoàn toàn khác bộ giải mã, ở chỗ là bộ chuyển mã thực hiện chuyển đổi từ bộ mã nhị phân này thành một bộ mã nhị phân khác. Đầu vào là một tín hiệu quan hệ nhị phân và đầu ra cũng là một tín hiệu quan hệ nhị phân.

Trong phần mã hóa chúng ta cũng đã nói tới các loại mã như mã BCD, mã Grây, mã thừa 3, mã thừa 3 Grây v.v... Đó là các bộ mã dùng trong máy mà do con người tạo ra quy ước ngay từ đầu. Do có nhiều loại mã dùng trong máy tính cũng như trong hệ thống số như vậy, từ đó nẩy ra nhiệm vụ chuyển đổi giữa các loại mã đó sang nhau như : mã BCD thành mã Grây

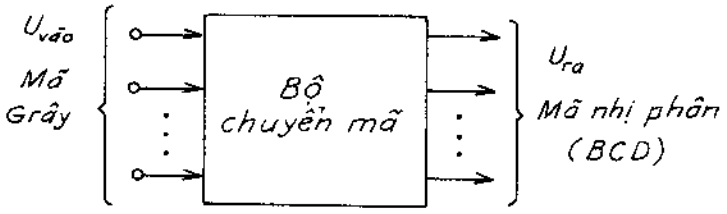
mã BCD thành mã thừa 3

mã Grây thành mã BCD

mã thừa 3 thành mã BCD.

Từ đó có thể hiểu bộ chuyển mã có thể chuyển đổi rất nhiều các loại mã khác nhau thành các loại mã cần thiết. Đây là một chức năng có thể dùng trong nhiều lĩnh vực kỹ thuật khác nhau.

Ví dụ : Xây dựng bộ chuyển mã từ mã Grây ba bit thành mã BCD thông thường. Chức năng của bộ chuyển đổi đó được minh họa trên hình 5.135.



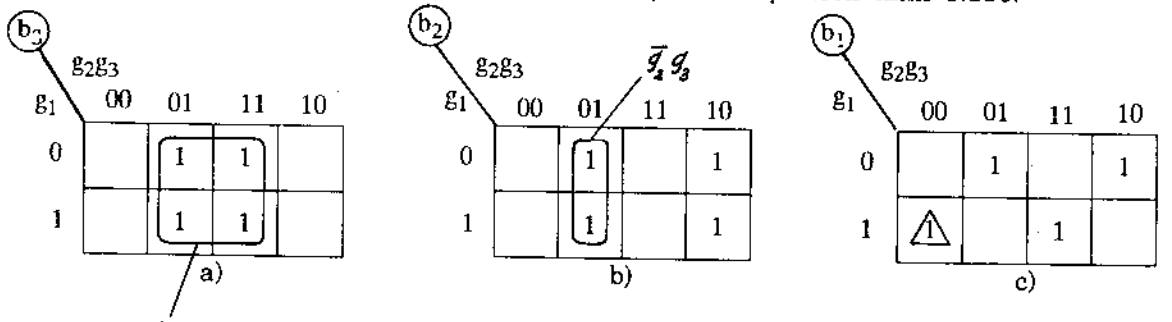
a)

Mã Grây			Mã nhị phân		
g_3	g_2	g_1	b_3	b_2	b_1
0	0	0	0	0	0
0	0	1	0	0	1
0	1	1	0	1	0
0	1	0	0	1	1
1	1	0	1	0	0
1	1	1	1	0	1
1	0	1	1	1	0
1	0	0	1	1	1

b)

Hình 5.135. Bộ chuyển đổi mã Grây thành mã BCD.

Cách thiết kế bộ chuyển mã từ mã Grây 3 bit thành mã nhị phân (BCD) như sau. Từ bảng chân lý thể hiện quan hệ chuyển đổi giữa hai loại mã nhị phân ta ánh xạ vào bìa Kácnô để tìm quan hệ của từng đầu ra (b_3, b_2, b_1) theo các tín hiệu vào là (g_3, g_2, g_1), kết quả của bước ánh xạ vào bìa Kácnô được thể hiện trên hình 5.136.



$b_3 = g_3$

$b_2 = \bar{g}_2 g_3 + g_2 \bar{g}_3$

$b_1 = \bar{g}_1 \bar{g}_2 g_3 + \bar{g}_1 g_2 \bar{g}_3 + g_1 \bar{g}_2 \bar{g}_3 + g_1 g_2 g_3$

Hình 5.136. Kết quả ánh xạ vào bìa Kácnô.

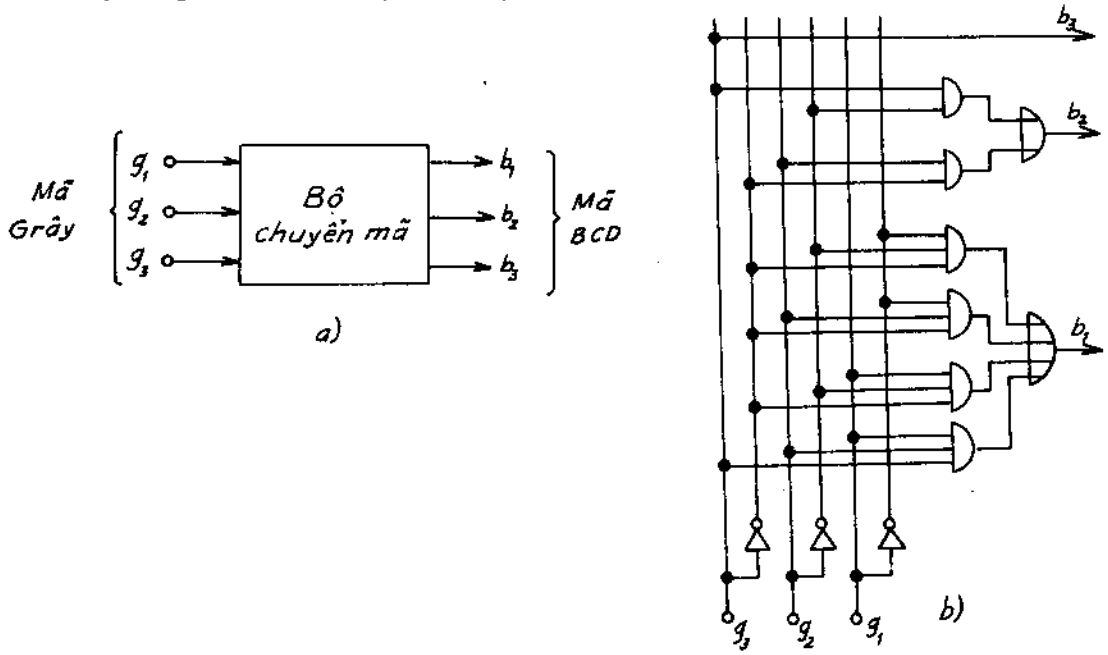
Sau khi ánh xạ bảng chân lý bộ chuyển mã ba bit vào bìa Kácnô, thu được hệ phương trình đáp ứng ra (b_3, b_2, b_1) theo các tín hiệu vào (g_3, g_2, g_1) là :

$$b_3 = g_3$$

$$b_2 = \bar{g}_2 g_3 + g_2 \bar{g}_3$$

$$b_1 = \bar{g}_1 \bar{g}_2 g_3 + \bar{g}_1 g_2 \bar{g}_3 + g_1 \bar{g}_2 \bar{g}_3 + g_1 g_2 g_3$$

Dựa vào hệ phương trình thu được ta có mạch điện nguyên lý của bộ chuyển mã ba bit của Grây sang mã (ABC) được thể hiện trên hình 5.137.



Hình 5.137. Bộ chuyển mã 3 bit mã Grây thành mã BCD.

Với cách làm tương tự có thể xây dựng được bộ chuyển mã từ mã Grây thành mã (BCD) với số lượng bit thông tin lớn hơn. Đồng thời với cách làm như vậy ta có thể xây dựng và tạo nên bộ chuyển đổi mã từ một mã bất kỳ này thành một mã cần thiết khác theo yêu cầu kỹ thuật.

Chương VI

TỔNG HỢP HỆ THỐNG SỐ DÙNG VI ĐIỆN TỬ

Xây dựng một hệ thống số bao gồm hai bước chính, bước thứ nhất là tổng hợp lôgic, bước này chỉ liên quan đến chức năng và khía cạnh lôgic của hệ thống số, tiếp theo đến bước thứ hai là tổng hợp mạch lắp mạch điện, liên quan tới thực hiện điện tử chức năng lôgic đã đưa ra. Các chương trình bày ở phần trên đã phân tích kỹ khía cạnh lôgic của bài toán (của chức năng), và sơ đồ nguyên lý của một ôtômat có nhớ, trên cơ sở các mạch trigơ cũng như các mạch lôgic kết nối chúng lại để thực hiện chức năng. Bước tiếp theo trong quá trình tổng hợp hệ thống số là lắp ráp mạch điện. Tín hiệu điện, thời gian trễ điện tử khi hoạt động, công suất, tải, tần số hoạt động là các tham số cơ bản, đó là chỉ tiêu kỹ thuật hoạt động của hệ thống số trong thực tế. Khi lắp mạch trong thực tế dùng IC để thực hiện ôtômat có nhớ, phải thận trọng để không xảy ra sự cố hỏng IC.

§6.1. GIỚI THIỆU CHUNG VỀ MẠCH VI ĐIỆN TỬ SỐ

Các chức năng lôgic cũng như các ôtômat có nhớ đều có thể lắp ráp mạch bằng linh kiện rời. Nhưng do sự phát triển rất mạnh của kỹ thuật vi điện tử cũng như công nghệ chế tạo IC nên quá trình tổng hợp lắp ráp các mạch số và các hệ thống số có nhiều thay đổi cơ bản. Đến nay người thiết kế mạch không phải tự xây dựng lắp ráp các mạch lôgic cơ bản nữa bởi vì chúng đã được xây dựng và chế tạo sẵn dưới dạng mạch IC hay các mạch điện modul. Thực hiện mạch điện cho hệ thống số dùng IC dễ dàng hơn, đơn giản hơn, độ tin cậy cao hơn, kích thước nhỏ hơn và được những chức năng lớn hơn hay rất lớn với giá thành hạ. Trong chương này giới thiệu một số mạch vi điện tử cơ bản.

6.1.1. MẠCH VI ĐIỆN TỬ VÀ PHÂN LOẠI

Việc phân loại IC là cần thiết vì cho ta biết các tính chất sơ bộ của một mạch IC cũng như tên gọi của nó. Do kỹ thuật sản xuất IC ngày càng phát triển nên người ta có xu hướng xây dựng luôn cả loại vi điện tử và mạch tổ hợp hay các khối chức năng trong

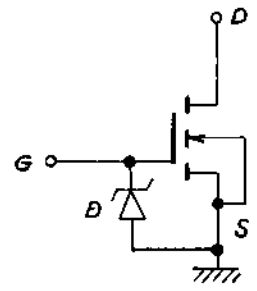
một vỏ (chip). Cho nên có thể coi phần chủ yếu của mạch vi điện tử là IC. Mạch tổ hợp IC được xây dựng bằng cách trên một đế tinh thể bán dẫn cỡ vài mm^2 tạo ra rất nhiều các phần tử mạch điện khác nhau. Để phân loại mạch IC người ta có nhiều tiêu chí hay quan điểm khác nhau.

6.1.1.1. Phân loại IC số theo công nghệ chế tạo

Để chế tạo ra một mạch IC hiện nay người ta có hai loại chính là công nghệ bán dẫn chuyển tiếp PN và công nghệ bán dẫn trường được chế tạo theo công nghệ MOS (Metal – Oxide – Semiconductor). Dùng công nghệ bán dẫn chuyển tiếp PN người ta chế tạo ra được các IC bán dẫn thông thường, với loại IC này có đặc điểm là điện áp nguồn cung cấp thường nhỏ (V_{cc} = +5 von), nếu điện áp cung cấp lớn sẽ dễ bị đánh thủng. Do nguồn cung cấp nhỏ nên IC loại này khả năng tải không cao.

Công nghệ MOS chế tạo ra các IC số có thể chịu được điện áp cung cấp cao tới hàng chục von, trở kháng vào của mạch điện cực kỳ cao cho nên khả năng tải của loại IC số dùng công nghệ MOS là rất lớn (do dòng điện vào của mạch rất nhỏ). Nhưng do đầu vào (cực G) của mạch có chất điện môi SiO_2 dày chưa đến 1 μm cách điện, tạo ra một điện dung ký sinh. Điện tích trên điện dung vào có thể lưu trữ rất lâu (không tiêu tán được) do điện trở vào quá lớn. Điều đó có lợi khi dùng hiện tượng đó để lưu giữ thông tin, nhưng do không tiêu tán được điện tích nên điện tích ở trên điện dung cực cửa ngày càng lớn, tích tụ cả điện tích không gian vào, từ đó sinh ra điện áp cao ở cực cổng, nó có thể đánh thủng lớp điện môi SiO_2 và làm hỏng tranzitor trường. Như vậy tranzitor MOS hoàn toàn có thể tự hỏng mặc dù không sử dụng. Qua đó phải chú ý khi bảo quản IC loại MOS, phải có đế cắm trung hòa điện tích ở các chân IC.

Để tránh sự cố tự đánh thủng của tranzitor trường do hiện tượng tích điện ở cực cửa (G) người ta chế tạo thêm mạch bảo vệ nối vào chân vi mạch, hình 6.1 minh họa cách bảo vệ đó.



Hình 6.1. Mạch bảo vệ cực cổng G.

Ở đầu vào người ta mắc thêm một điôt Zene Đ. Nếu điện áp tĩnh điện ở cực G tích lại lớn quá thì điôt Zene Đ sẽ bị đánh thủng, tạo đường thoát cho các điện tích ở cực cửa G, làm cho lớp cách điện silic SiO_2 không bị đánh thủng.

6.1.1.2. Phân loại theo mức độ tập trung các phần tử trong một vỏ

Như đã biết mỗi một con IC khi sản xuất có chứa trong nó số lượng phần tử khác nhau, từ đó dựa vào số lượng các phần tử chứa trong một vỏ IC (chip) người ta phân chia chúng thành các lớp khác nhau.

Mạch vi điện tử cỡ nhỏ SSI (Small Scale Integration) cỡ hàng trăm phần tử, mạch cỡ lớn LSI (Large Scale Integration) có cỡ hàng ngàn phần tử, mạch cỡ cực lớn VLSI

(Very Large Scale Integration) cỡ hàng chục ngàn phần tử, mạch cỡ siêu lớn VVLSI (Very Very Large Scale Integration) hay ELSI (Extra Large Scale Integration) cỡ vài trăm ngàn phần tử trong một vỏ (chip) IC.

6.1.1.3. Phân loại theo chức năng sử dụng

Loại DIC (digital) là loại IC số, làm việc với hai mức tín hiệu điện, quan hệ giữa đầu vào và đầu ra là quan hệ lôgic rời rạc. Chúng thường được dùng để lắp ráp xây dựng nên các hệ thống số.

Loại AIC (Analog) là loại IC tương tự, làm việc với tín hiệu vào biến đổi liên tục theo thời gian. Quan hệ giữa đầu vào và đầu ra là quan hệ liên tục của điện áp. Loại IC tương tự thường được dùng trong đài, trong bộ khuếch đại ampli, trong mạch TV ...

Loại IC biến đổi tương tự sang số (A/D), hay chuyển từ tín hiệu số thành tín hiệu tương tự (D/A). Loại mạch này được dùng để phối ghép giữa hai hệ thống tín hiệu điện tử giữa số (Digital) và tương tự (Analog).

Trong phần này chủ yếu đi sâu vào cách sử dụng loại DIC (Digital) để xây dựng nên các hệ thống số, đây là mạch IC được chế tạo theo loại TTL (tranzitor, tranzitor lôgic).

6.1.1.4. Phân loại theo hãng sản xuất

Một mạch IC có cùng chức năng nhưng có thể do nhiều hãng cùng sản xuất. Do đó cũng có thể dựa vào hãng sản xuất để phân loại IC, cũng như để đánh giá chất lượng của một hệ thống số khi ta tiếp xúc. Trên thế giới có những hãng lớn chuyên sản xuất IC như sau, mỗi một hãng có ký hiệu chữ riêng cho con IC của mình sản xuất.

Texas Instruments – SN	SN 7400
	SN 54/74
Motorola – MTRL, MC	MC 7400
National – IPC	IPC 7400
	IPC 8200
Siemens – FL	FL 7400
Philips – FJ	FJ 7400
CEMI (Balan) – UCY	UCY 7400

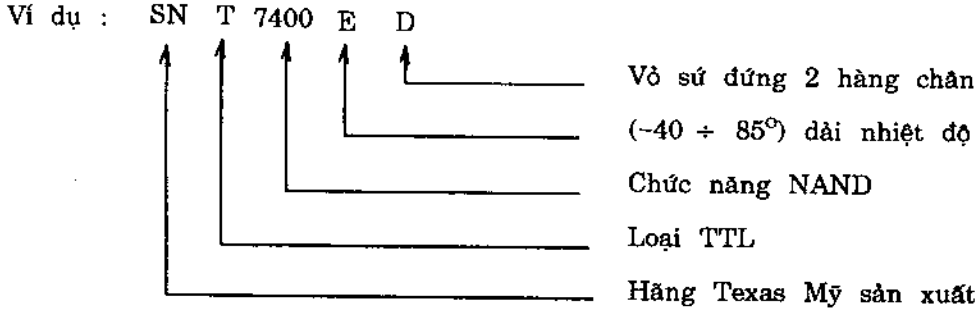
Qua đó ta hiểu là ký hiệu chữ được in trên con IC là ký hiệu của hãng sản xuất, còn con số đằng sau là quy ước quốc tế chức năng của con IC đó ví dụ :

SN 7400 là mạch NAND do hãng Texas Instruments của Mỹ chế tạo

FL 7400 là mạch NAND do hãng Siemens của Đức chế tạo.

Ngoài các ký hiệu chính là hãng sản xuất và chức năng của con IC đó, người ta

còn có thể có thêm các ký hiệu khác để chỉ công nghệ chế tạo, dải nhiệt độ cho phép, cách đóng vỏ con IC... được minh họa như sau :



Tất cả các ký hiệu về tên hãng sản xuất, chỉ tiêu kỹ thuật của các con IC đã sản xuất luôn được ghi rõ trong các sổ tay tra cứu. Trên cơ sở các ký hiệu đã nêu trên chúng ta có thể tự tìm hiểu để sử dụng chúng khi xây dựng các hệ thống số cho thích hợp.

6.1.2. CÁC CHỈ TIÊU KỸ THUẬT CỦA IC LOẠI TTL

Phần này đưa ra các chỉ tiêu kỹ thuật của mạch IC loại TTL (tranzitor tranzitor lôgic) là loại IC hiện đang được sử dụng rộng rãi nhất trong việc xây dựng các hệ thống số. Do đó ta cần tìm hiểu và biết chỉ tiêu kỹ thuật của chúng và đây là loại IC có tốc độ hoạt động rất cao và giá thành rẻ so với các loại IC khác.

6.1.2.1. Nguồn cung cấp $V_{cc} = +5V \pm 5\%$

Mức điện áp cung cấp yêu cầu + 5 von là mức điện áp không cao nhưng yêu cầu độ ổn định rất cao. Đó là yêu cầu chế tạo rất cao khi thiết kế bộ nguồn ổn áp cung cấp cho máy vi tính. Độ ổn định của mạch nguồn cung cấp phải giống hệt như điện áp của pin hay của acquy, vì không được phép có nhiễu công nghiệp theo đường nguồn cung cấp xuyên vào trong máy vi tính. Nếu có nhiễu công nghiệp lọt theo đường mạch nguồn vào máy tính, nó sẽ làm hỏng các thông tin của máy tính và làm hỏng các dữ liệu được ghi vào máy (không giống như TV và đài có nhiễu cũng không ảnh hưởng gì nhiều) đó là điều tai hại cho người sử dụng máy tính, cũng như đối với hệ thống số dùng để điều khiển các chức năng khác nhau của hệ thống tự động lớn. Do đó chỉ tiêu ổn định của nguồn cung cấp ở đây được đưa lên hàng đầu.

6.1.2.2. Mức lôgic

Mức lôgic 1 điện áp (2,4 ÷ 5) von

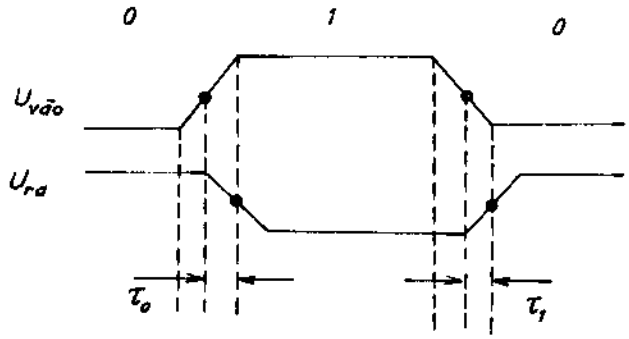
Mức lôgic 0 điện áp (0 ÷ 0,4) von.

Điều kiện tốt nhất của mức lôgic 1 điện áp là + 5 von còn mức lôgic 0 là 0 von, với mức điện áp như vậy chức năng các mạch điện lôgic hoạt động rất tin cậy. Nhưng do có điện áp dư khi tranzitor thông cũng như điện áp sụt trên tải ở đầu ra, làm cho mức lôgic tối thiểu cho phép là : mức lôgic 1 là + 2,4 von, còn mức lôgic 0 là + 0,4 von, với

các mức lôgic như vậy thì mạch lôgic còn hoạt động được. Trường hợp các mức lôgic không thỏa mãn giá trị điện áp như vậy thì mạch lôgic không thể hoạt động tin cậy được nữa. Với giải điện áp của mức lôgic như vậy ta có thể đo đạc kiểm tra chất lượng hoạt động của mạch.

6.1.2.3. Tác động nhanh τ của mạch TTL

Tác động nhanh của mạch TTL hay là quán tính của mạch được xác định theo biểu đồ thời gian chỉ ra trên hình 6.2.



Hình 6.2. Biểu đồ thời gian trễ trong mạch TTL.

Trong đó τ_0 là thời gian chuyển từ (0 sang 1) còn thời gian τ_1 là thời gian chuyển từ (1 về 0). Tác động nhanh τ của mạch được xác định :

$$\tau = \frac{\tau_0 + \tau_1}{2}$$

Giá trị thời gian trễ τ của mạch được cho trong sổ tay tra cứu. Ta có thể dựa chủ yếu vào thời gian trễ τ để đánh giá chất lượng của mạch IC như sau. Với loại IC hoạt động cực nhanh $\tau \leqslant 5$ ns, loại nhanh $\tau \leqslant 10$ ns, loại hoạt động trung bình τ khoảng (10 ÷ 100) ns, loại IC tác động nhanh chậm τ cỡ μs.

6.1.2.4. Công suất tổn hao P_A

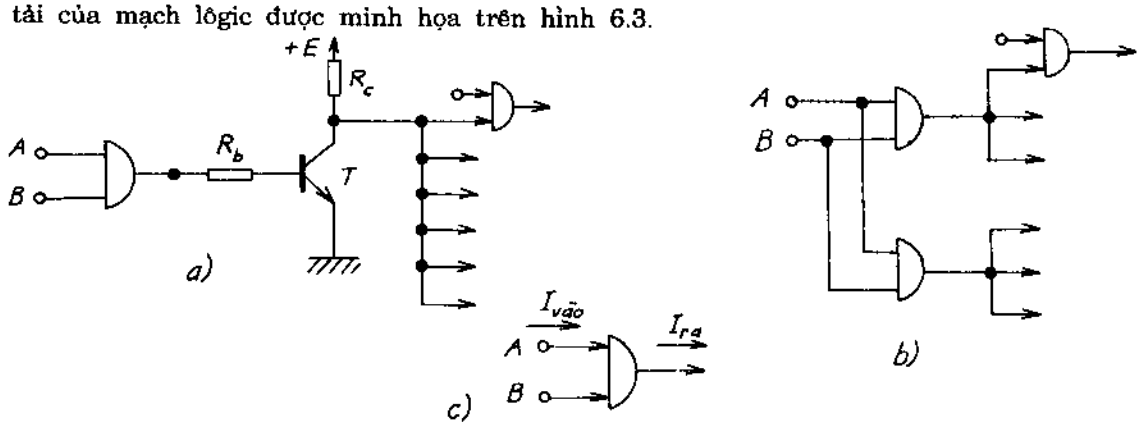
Một con IC loại TTL công suất tổn hao khoảng P_A = (10 ÷ 20) mW. Với một con IC thì công suất tổn hao không đáng kể, nhưng nếu xây dựng các mạch điện lớn có mật độ tập trung của các IC cao sẽ làm cho mạch bị nóng lên. Chính vì có công suất tổn hao P_A làm nóng máy nên các máy tính cần có hệ thống quạt thông gió và làm nguội máy. Đó cũng là một trong những tham số kỹ thuật đáng lưu tâm khi xây dựng các hệ thống số dùng mạch IC.

6.1.2.5. Khả năng tải N của mạch

Mỗi một mạch lôgic loại TTL đều có một dòng điện đầu vào ($I_{vào}$) và một dòng điện cung cấp ở đầu ra (I_{ra}). Đầu ra mạch lôgic này lại kích vào đầu vào mạch lôgic tiếp theo, khả năng kích được bao nhiêu mạch tiếp theo chính là khả năng tải của mạch lôgic loại TTL. Khả năng tải của mạch được xác định :

$$N = \frac{I_{ra\ min}}{I_{vào\ max}}$$

Thông thường một mạch lôgic loại TTL trong thực tế chỉ có thể có khả năng tải hay kích được từ 2 đến 3 mạch lôgic tiếp theo là cùng. Các phương pháp tăng khả năng tải của mạch lôgic được minh họa trên hình 6.3.



Hình 6.3. Các biện pháp tăng khả năng tải.

6.1.3. CÁC MẠCH ĐIỆN CƠ BẢN TRONG IC LOẠI TTL

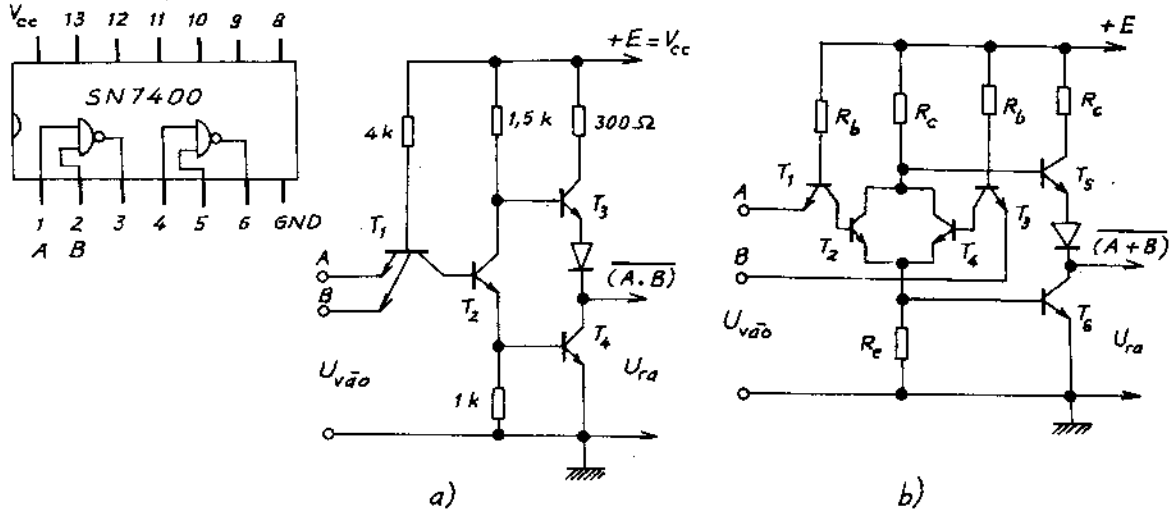
Muốn dùng mạch điện IC để xây dựng nên các hệ thống số sử dụng trong thực tế ta phải nối mạch điện cho IC, phải cho tín hiệu điện vào và nối mạch tải lấy tín hiệu ra từ IC. Trong quá trình lắp ráp mạch điện IC với các linh kiện khác ở trong mạch nếu ta không nắm vững mạch điện cụ thể (vùng đầu vào và khu vực đầu ra) bên trong của con IC, thì có thể nối sai hay làm hỏng IC mà ta hoàn toàn không biết vì sao. Để tránh làm hỏng mạch IC, ta cần biết thêm về cấu tạo vài mạch điện cơ bản thường hay được dùng khi chế tạo IC. Khi biết được cách mắc mạch bên trong con IC thì ta có thể chủ động đưa tín hiệu vào và lấy tín hiệu ra mà không còn lo IC bị hỏng nữa, vì khi lắp ráp mạch điện có lúc phải dùng các con IC đắt tiền.

6.1.3.1. Mạch điện cơ bản hay được dùng trong con IC loại TTL

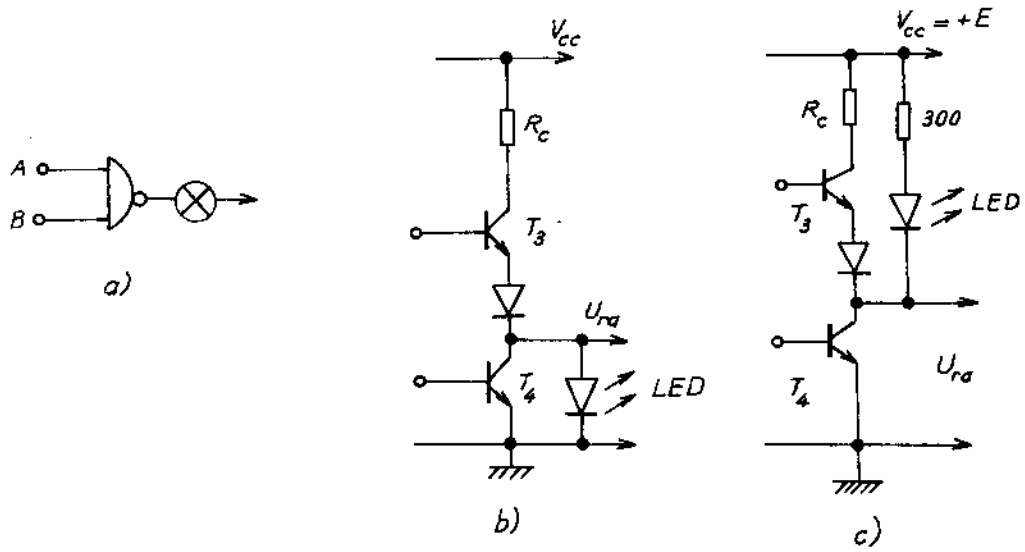
Mạch điện cơ bản trong IC loại TTL được vẽ trên hình 6.4.

Từ sơ đồ mạch điện dễ dàng nhận thấy tín hiệu vào ($V_{vào}$) đưa vào cực emitơ của tranzito T_1 (không đưa vào trigơ như trong mạch khuếch đại) cho nên nếu đưa vào điện áp cao khi T_1 tắt (hở ra) toàn bộ IC không làm việc nên IC không hỏng được. Trường hợp khác U_v chập đất thì T_1 lại hoạt động bình thường vì emitơ nối đất. Qua đó ta thấy tín hiệu đưa vào ở đầu vào mạch IC khó có thể làm hỏng nó (không giống như ở mạch khuếch đại có thể làm hỏng mạch khi cho tín hiệu vào không đúng).

Ta để ý tới khu vực đầu ra của mạch, thường đầu ra (V_{ra}) được nối với tải hay bộ chỉ thị. Ở đây ta phải chú ý vì việc nối tải tiếp theo của IC có thể làm hỏng nó. Hiện tượng như sau. Ta có mạch chỉ thị lôgic 0 và chỉ thị lôgic 1 ở đầu ra của mạch được vẽ trên hình 6.5.



Hình 6.4. Mạch điện cơ bản NAND và NOR trong IC.



Hình 6.5. Mạch chỉ thị đầu ra của IC.

Trường hợp chỉ thị mức lôgic 1 mạch điện như hình 6.5,a thì việc chỉ thị dùng điôt LED mắc mạch như vậy mạch sẽ hoạt động bình thường, khi $U_{ra} = +E$ sẽ làm cho điôt chỉ thị sáng lên. Trường hợp này không gây hỏng IC.

Trường hợp chỉ thị lôgic 0 mạch điện mắc như hình 6.5,b, chỉ thị lôgic 0 là khi U_{ra} = 0 von thì đèn chỉ thị sáng lên. Khi T_4 thông (đóng mạch) làm cho U_{ra} = 0 von, nếu ở mạch chỉ thị dùng đèn điod LED ta không mắc nối tiếp thêm một điện trở khoảng 300 Ω thì có thể làm nóng IC hay gây hỏng IC, vì khi T_4 thông U_{ra} = 0 làm cho điôt LED phân cực thuận cũng thông (đóng mạch) nếu không có điện trở hạn dòng 300 Ω thì trở kháng ở mạch chỉ thị T_4 thông và điôt LED thông sẽ là xấp xỉ 0 Ω, làm cho

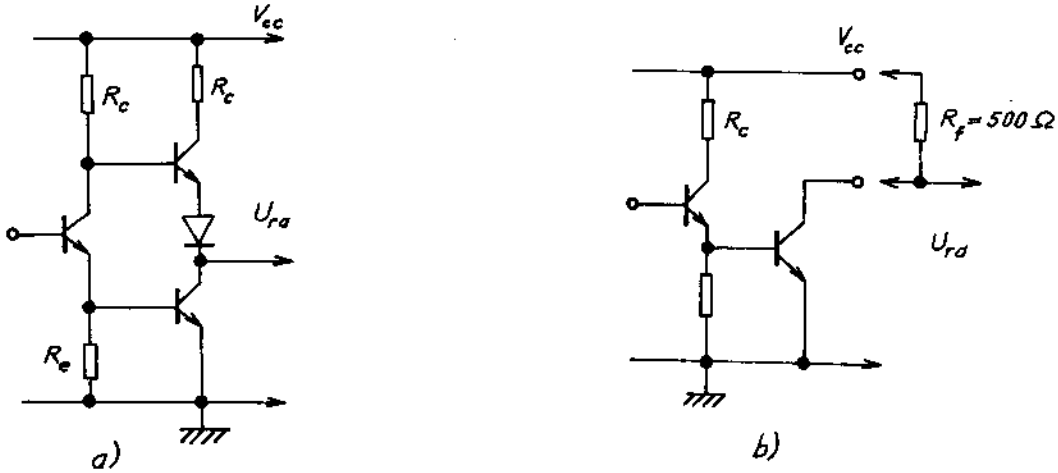
dòng điện chảy qua chúng sẽ rất lớn, hoặc làm cháy điôt LED hoặc làm hỏng tranzito T_4. Nếu tranzito T_4 hỏng tức là IC sẽ bị hỏng. Hiện tượng như vậy còn có thể được gọi là chập tải gây hỏng IC, vì vậy khi lắp mạch điện cụ thể ta cần phải lưu ý tới trường hợp này. Nếu phải chỉ thị mức lôgic 0 thì phải mắc thêm một điện trở 300 Ω để hạn chế dòng điện tránh làm hỏng IC.

Thông thường đối với IC loại TTL mạch vào và mạch điện đầu ra của nó được mắc mạch như vậy. Nhưng cũng có thể có con IC đầu vào và đầu ra của nó người ta tổ chức theo kiểu khác. Trong trường hợp này để tránh làm hỏng IC ta vẫn nên tìm và biết qua về các mạch vào ra của nó, thì nối mạch sẽ an toàn và chủ động hơn và không sợ bị chập tải gây hỏng IC. Các mạch điện cơ sở cho các loại IC khác nhau thông thường đều được chỉ ra và vẽ sẵn trong sổ tay tra cứu, các mạch điện vào và ra đó sẽ đại diện tổ chức mạch cho toàn bộ dòng họ IC được đưa ra trong sổ tay tra cứu, trước khi lắp mạch cụ thể ta cần biết và nghiên cứu kỹ từ trước các mạch này.

Như vậy để thiết kế và lắp ráp mạch điện thực tế cho một hệ thống số, thì ngoài việc tìm hiểu chức năng lôgic của mạch IC sẽ dụng, người ta còn phải biết chính xác tổ chức của mạch điện đầu vào và đầu ra của con IC đó để không làm hỏng IC khi lắp mạch thực tế.

6.1.3.2. Các kiểu mạch ra của IC loại TTL

Mạch ra thông thường như đã biết của con IC loại TTL được chỉ ra trên hình 6.6,a.

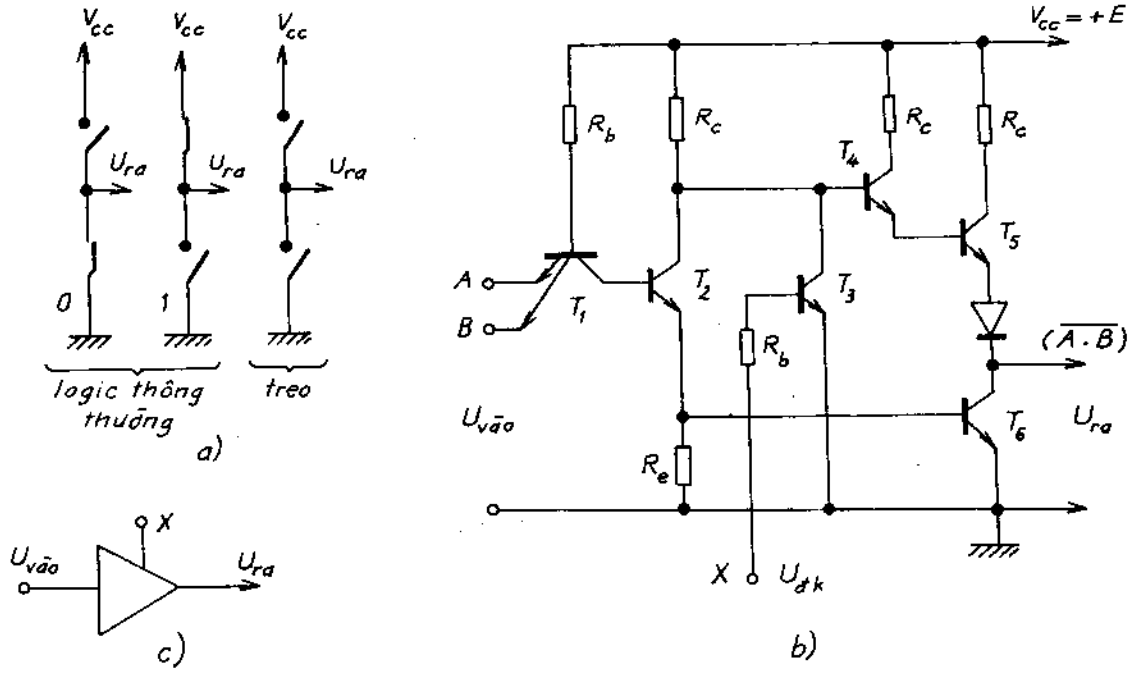


Hình 6.6. Một vài kiểu mạch ra của IC loại TTL.

Loại mạch ra bình thường có trở kháng ra thấp, nếu bị ngắn mạch tải có thể gây hỏng IC. Để đảm bảo an toàn hơn cho IC khi ta sử dụng nó, người ta chế ra loại IC mạch ra dạng colectơ hở như trên hình 6.6b. Muốn sử dụng được loại IC này ta phải mắc thêm ở ngoài một điện trở phụ $R_f = 500\ \Omega$, nếu không có điện trở mắc thêm vào thì IC không làm việc. Khi có thêm điện trở phụ R_f nối vào thì dù có bị chập tải mạch IC vẫn không bị hỏng.

6.1.4. MẠCH BA TRẠNG THÁI HAY MẠCH TRỞ KHÁNG CAO (TREO)

Mạch ba trạng thái, mạch trở kháng cao hay mạch treo là một mạch điện không phải là mạch có chức năng lôgic, nhưng nó là mạch điện không thể thiếu được trong các hệ thống số, vì nó quyết định tới cấu hình cũng như tổ chức nối mạch của hệ thống số. Có mạch ba trạng thái việc nối mạch điện trong các hệ thống số trở nên rõ ràng và đơn giản hơn rất nhiều. Điển hình là việc sử dụng mạch điện trở kháng cao (treo) trong cấu hình của máy vi tính. Để hiểu được vai trò quan trọng của mạch ba trạng thái, ta đi vào nghiên cứu chức năng hoạt động cũng như ứng dụng của nó trong việc thực hiện lắp ráp hệ thống số. Chức năng của mạch ba trạng thái được minh họa trên hình 6.7.



Hình 6.7. Mạch ba trạng thái U_{ra} trở kháng cao.

Đối với mạch lôgic thông thường đầu ra (U_{ra}) của mạch chỉ có hai giá trị là 0 hoặc 1 tương ứng với các công tắc hoặc đóng hoặc mở. Còn trong trường hợp trở kháng cao ở đầu ra thì hai công tắc cùng mở, làm cho đầu ra (U_{ra}) bị "treo" lơ lửng như mô tả trên hình 6.7,a.

Trong trường hợp đầu ra bị treo lơ lửng, cho phép chúng ta có thể cho tín hiệu vào đầu ra (U_{ra}) mà không sợ bị "chập mạch" (ngắn mạch) như trong trường hợp ở mạch lôgic thông thường. Đây là một ưu thế tuyệt đối và là một ưu việt chỉ có được ở mạch ba trạng thái mà ta có thể lợi dụng trong việc truyền thông tin trong mạch.

Để có được hoạt động của mạch ba trạng thái (hai trạng thái lôgic là 0 và 1, và một trạng thái treo) như chỉ ra trên hình 6.7,a, người ta xây dựng mạch điện cụ thể

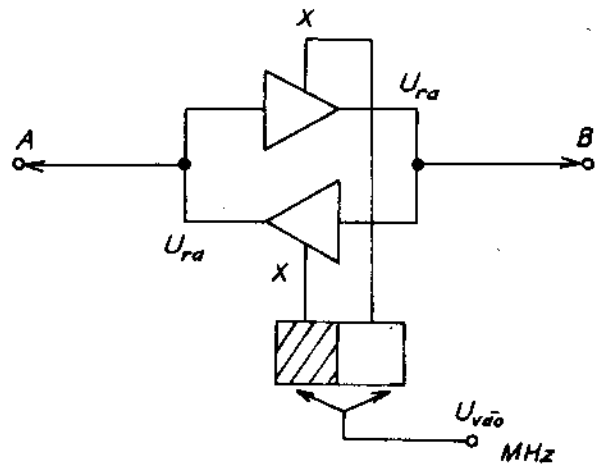
thực hiện được ba trạng thái mô tả ở trên hình 6.7,b. Dựa vào sơ đồ mạch điện ta có thể nêu nguyên lý làm việc của nó hoạt động theo yêu cầu của mạch ba trạng thái.

Nguyên lý làm việc của mạch điện :

Mạch có một đầu vào điều khiển U_{dk} là X. Khi ta cho X = 0 (U_{dk} = 0 von) thì tranzito T_3 sẽ tắt (hở ra) vì đầu vào bazơ của nó nối đất. Lúc này mạch hoạt động như một mạch lôgic NAND thông thường. Ở đầu ra của mạch sẽ cho các giá trị lôgic là 0 (U_{ra} = 0 von) hay giá trị lôgic 1 (U_{ra} = V_{cc}). Ở đây ta không nói sâu tới chức năng lôgic.

Khi ta cho X = 1 ($U_{kđ}$ + E) làm cho tranzito T_3 sẽ thông (đóng lại) và xuất hiện một dòng điện chảy qua nó xuống đất. Khi T_3 thông làm cho điện áp tại điểm A (U_A = 0 von) nối đất, dẫn đến $U_{ce\ T2}$ = 0 von như vậy tranzito T_2 không có nguồn cung cấp nên T_2 sẽ tắt (hở ra), không có dòng điện chảy qua T_2 nên U_{Re} = 0 (điện áp trên điện trở Re). Mà U_{Re} lại là điện áp vào của tranzito T_6 (U_{Re} = $U_{be\ T6}$ = 0 von) nên làm cho T_6 tắt (hở ra). Đồng thời với việc U_A = 0 von làm cho điện áp vào bazơ của T_4 (U_A = $U_{be\ Tc}$) nối đất, điều đó làm cho tranzito T_4 không làm việc (hở ra). Khi T_4 hở ra làm cho tranzito T_5 không có điện trở thiên áp (bazơ treo) nên tranzito T_5 tắt (hở ra). Kết quả cuối cùng khi cho X = 1 là T_5 tắt (hở ra) và T_6 tắt (hở ra) điều đó giống như hai công tắc cùng mở, làm cho đầu ra (U_{ra}) của mạch bị treo lơ lửng, gọi là ở mức trở kháng cao. Như vậy với đầu vào điều khiển X mạch điện đã cho thực hiện được chức năng của mạch ba trạng thái trở kháng cao.

Ứng dụng cơ bản và cũng là ứng dụng cơ sở quan trọng nhất của mạch ba trạng thái là : dùng mạch ba trạng thái để thực hiện được thông tin hai chiều hay truyền tín hiệu điện hai chiều trên một đường dây hay trên một sợi dây điện. Cách thực hiện chỉ ra trên hình 6.8.

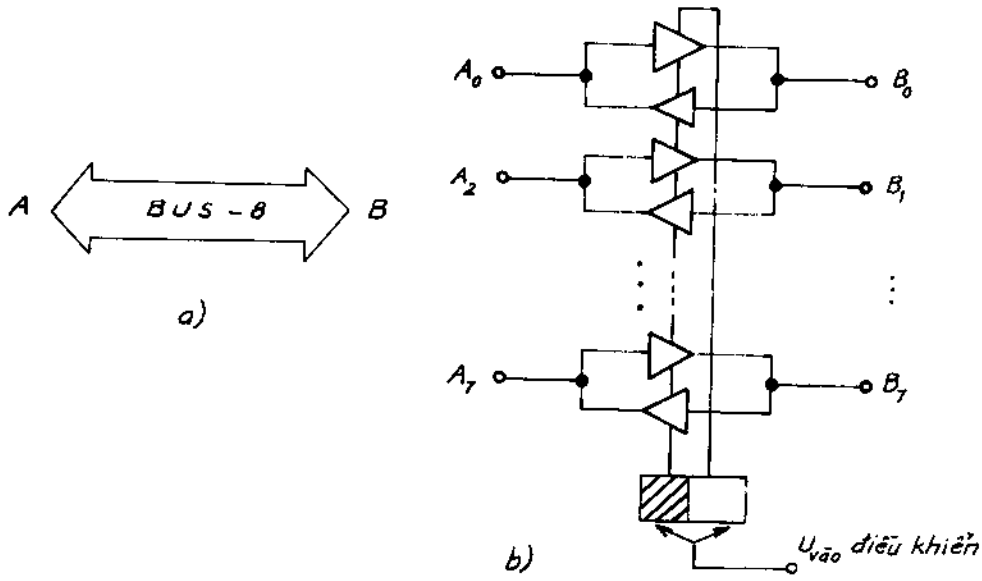


Hình 6.8. Thông tin hai chiều trên một sợi dây khi sử dụng mạch ba trạng thái.

Trong mạch để truyền được thông tin hai chiều đoạn dây điện AB người ta dùng hai mạch ba trạng thái (treo) và một trigơ để điều khiển chiều thông tin không trùng nhau khi hoạt động. Do đầu ra U_{ra} bị treo nên tín hiệu truyền qua lại không bị ảnh

hướng gì. Tùy theo yêu cầu truyền thông tin mà ta đưa xung điều khiển vào $U_{vào}$. Nói cách khác đầu vào kích $U_{vào}$ quyết định hướng truyền hay chiều truyền thông tin trên đoạn dây AB.

Từ một sợi dây có thể tổ chức được truyền thông tin theo hai chiều trên nó, ta có thể tổ chức hay ghép nhiều sợi dây có được cách truyền thông tin hai chiều, tạo thành một BUS thông tin hai chiều. Một BUS thông tin hai chiều sử dụng mạch ba trạng thái (treo) được minh họa trên hình 6.9.



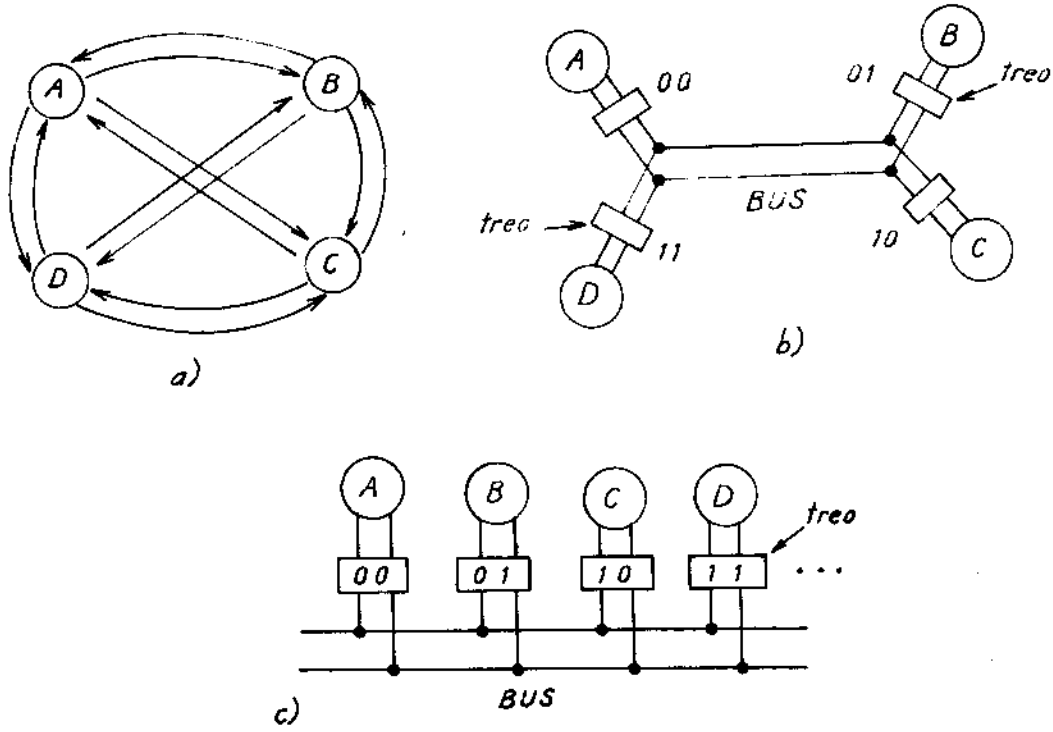
Hình 6.9. Cách tổ chức BUS thông tin hai chiều.

Một trong những ứng dụng quan trọng nữa của mạch ba trạng thái là làm thay đổi tổ chức kết cấu của mạng thông tin liên lạc hai chiều trên toàn mạng. Ứng dụng đó được minh họa trong ví dụ sau.

Ví dụ : Có 4 điểm thông tin (A, B, C, D) cần phải có sự liên lạc thông tin hai chiều giữa các điểm đó. Để đáp ứng được yêu cầu thông tin đề ra cần phải có cách nối mạng mô tả trên hình 6.10,a.

Nhưng cách tổ chức nối mạng thông tin theo hình 6.10,a cần có rất nhiều đường dây nối mạng, nhất là khi số điểm thông tin cần kết nối liên lạc nhiều hơn nữa. Để tiết kiệm dây nối và có kết cấu tổ chức mạng thông tin hai chiều đơn giản hơn, nhưng chức năng liên lạc là tương đương và không thay đổi, ta có thể dùng mạch treo để thực hiện. Mạng thông tin dùng mạch trở kháng cao (treo) được chỉ ra trên hình 6.10,b. Ở đây dùng một đường truyền BUS chung và có các cổng là mạch trở kháng cao (treo). Chức năng của mạng là tương đương nhau về kết nối, ví dụ muốn liên lạc giữa hai điểm (A - B) hai chiều ta chỉ cần treo (C, D) lên. Còn nếu muốn liên lạc (A - C) thì chỉ cần treo (B, D) lên là xong. Việc điều khiển treo tự động các cổng thông tin, người ta dùng

các bộ mã được truyền vào đường truyền trước khi trao đổi thông tin cụ thể. Có thể hiểu đơn giản đó là mã vùng trong tổng đài chẳng hạn.



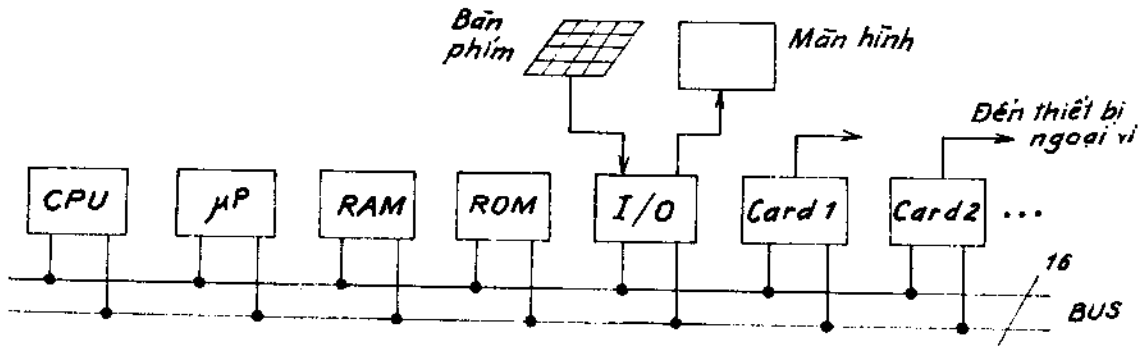
Hình 6.10. Mạng thông tin hai chiều cho 4 điểm.

Như vậy khi xuất hiện mạch trở kháng cao ba trạng thái làm cho việc kết nối thông tin trở nên thuận lợi hơn, tiết kiệm hơn và dễ dàng hơn. Vừa tiết kiệm được đường dây vì thông tin hai chiều chỉ trên một sợi dây, đồng thời tổ chức mạng thông tin cũng rõ ràng mạch lạc hơn phù hợp với thực tế hơn. Đó là ưu việt tuyệt đối mà chỉ có được ở mạch trở kháng cao.

Những mạng thông tin kết nối theo hình 6.10a có ưu việt là nối thông tin được nhanh và không phải chờ đợi, tổ chức như vậy trong thực tế vẫn được sử dụng trong quân sự khi số lượng điểm nối thông tin ít và yêu cầu thông tin phải có ngay tức thời.

Còn cách tổ chức như hình 6.10,b có ưu điểm là đơn giản, tiết kiệm và có thể kết nối được rất nhiều điểm thông tin với nhau. Nhưng phải dùng đường dây BUS chung, nên tốc độ truyền tin bị chậm, thậm chí có thể xảy ra tắc nghẽn đường truyền, đồng thời cần có tổ chức điều khiển đảm bảo sự đồng bộ hoạt động rất cao thì mới phát huy được hiệu quả của tổ chức mạng. Loại tổ chức như vậy đang được sử dụng rộng rãi trong hệ thống thông tin dân dụng. Ở đây BUS thông tin thông thường là một tuyến cáp quang. Loại kết nối thông tin kiểu này thông thường chỉ cho phép kết nối được hai điểm với nhau.

Từ sơ đồ ở hình 6.10b, ta có thể chuyển đổi tương đương về dạng kết nối chỉ ra như ở hình 6.10c. Ở đây cấu hình mạch điện như hình 6.10c là một cấu hình tổ chức liên kết thông tin rất cơ bản và rất quan trọng. Cấu hình đó có thể dùng trong các mạng thông tin nói chung, đó chính là tổ chức cấu hình của máy tính hay của máy vi tính hiện nay. Đó là cấu hình chuẩn có được nhờ có mạch ba trạng thái. Chính vì dùng mạch trở kháng cao trong việc xây dựng mạch điện cho máy vi tính nên các mạch điện (đường dây BUS) bên trong máy vi tính rất gọn, mạch lạc và rõ ràng. Cấu hình đó được minh họa trên hình 6.11.



Hình 6.11. Cấu hình của máy vi tính.

Ở đây dùng chung một BUS thông tin là đường dây gồm 8 bit thông tin hoặc 16 bit thông tin tùy theo. Trên đầu vào mỗi khối chức năng điều có các mạch ba trạng thái hay trở kháng cao, cho phép kết nối thông tin giữa hai khối cụ thể còn các khối chức năng khác thì "treo" lên. Thực hiện điều đó người ta dùng các lệnh đầu tiên của phần mềm trước khi có lệnh thực hiện cụ thể việc trao đổi thông tin giữa các khối.

Đây cũng là một trong các ứng dụng rất quan trọng của mạch ba trạng thái dùng trong các hệ thống số, mặc dù mạch trở kháng cao chỉ là một mạch điện có nguyên lý đặc biệt, mà không phải là mạch điện có chức năng lôgic. Đến đây các vấn đề khái quát chung về mạch vi điện tử IC, các tham số cơ bản của nó và các mạch điện chính cơ bản cần biết, cần lưu ý khi lắp ráp mạch số dùng vi điện tử đều đã được trình bày. Dưới đây là việc tổng hợp cụ thể hơn chức năng lôgic dùng IC.

§6.2. TỔNG HỢP MẠCH HỆ THỐNG SỐ DÙNG IC LOẠI TTL

Tổng hợp mạch cụ thể hệ thống số bằng IC số, thực chất là bước tiếp theo hay bước cuối cùng trong quá trình xây dựng hệ thống số. Sau khi ta đã tổng kết ra hệ thống số và có được sơ đồ nguyên lý của ôtômat có nhớ, đến đây chỉ cần chọn các con IC thích hợp, cho nguồn cung cấp và nối chúng lại bằng mạch theo sơ đồ nguyên lý đã có là xong. Sau đó có thể thử cho hệ thống hoạt động theo các tín hiệu vào khác nhau, hay đo đạc kiểm tra chức năng của mạch. Khi ta đã thiết kế chính xác, thì chắc chắn

mạch điện sẽ thực hiện đúng theo yêu cầu đề ra. Đó là một điều kiện thuận lợi ở mạch số khi xây dựng mạch điện cụ thể so với xây dựng mạch điện cho hệ thống tương tự. Ngoài các mạch điện cụ thể cho từng ôtômat khi thiết kế, người ta còn chế tạo sẵn các IC thực hiện một chức năng nào đó để ta tiện sử dụng.

6.2.1. BỘ NHỚ CỐ ĐỊNH CHỈ ĐỌC - ROM (READ ONLY MEMORY)

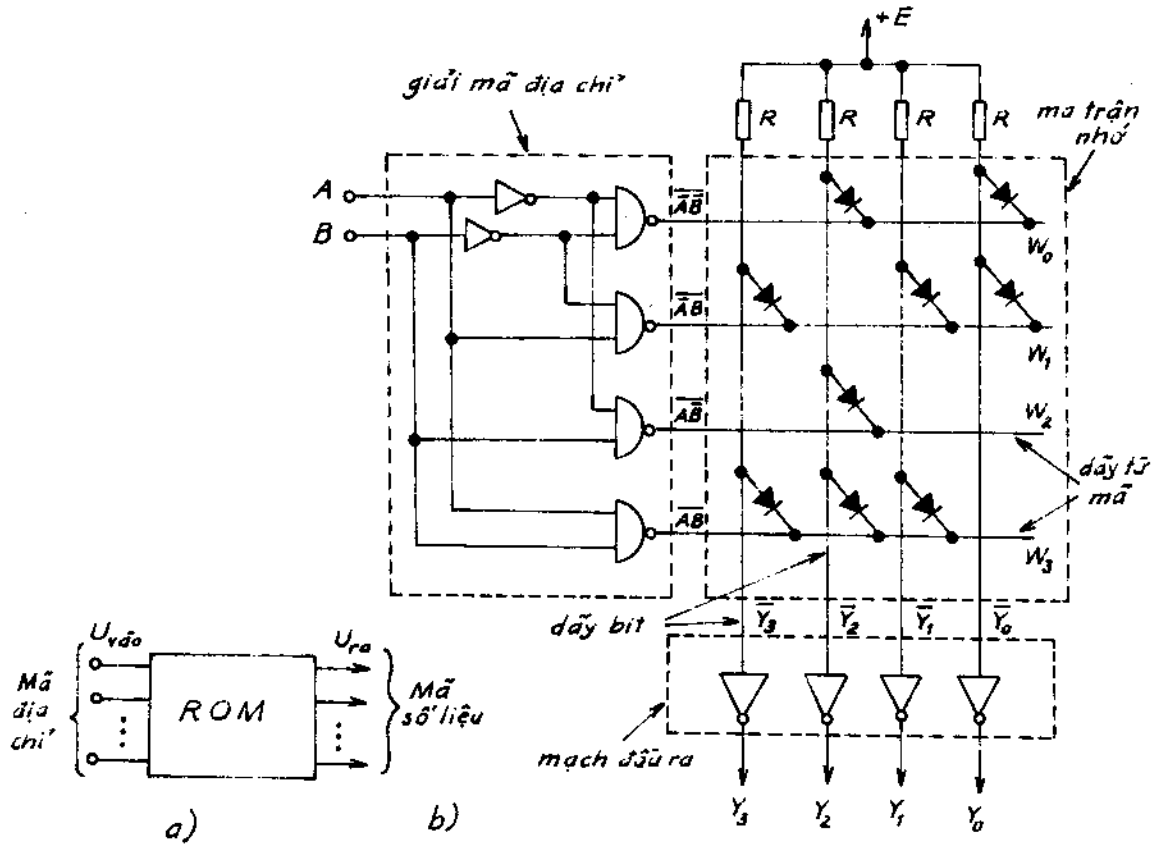
Bộ nhớ ROM là bộ nhớ chỉ có đọc nội dung đã được ghi sẵn ở bên trong của nó ra, nghĩa là thông tin đã ghi trong bộ nhớ là cố định biết trước, thông tin này không thay đổi ngay cả khi mất nguồn điện cung cấp và nó được ghi sẵn vào bộ nhớ ROM từ nhà máy sản xuất. Những nội dung thông tin biết trước cố định thông thường có ví dụ như : Bộ mã ASCII mã cho bàn phím của máy vi tính, đó là thông tin biết trước và mãi mãi không thay đổi nên được ghi luôn vào bộ nhớ cố định, nó gọi là bộ nhớ bàn phím ROMBIOD. Hay như một chương trình riêng (địa phương) dùng để điều khiển máy tính mà chương trình đó không thay đổi, ví dụ như chương trình kiểm tra (Test) máy tính ban đầu khi ta bật máy, chương trình này cũng được ghi sẵn vào một con IC nhớ ROM, tạo ra một bộ nhớ cố định, đọc nó sẽ cho ra chương trình riêng để thử máy tính. Hay ở trong một rôbôt, trong một máy công cụ có các nhiệm vụ không thay đổi, thì các chức năng đó cũng sẽ được ghi sẵn cố định vào trong ROM. Khi ta muốn thay đổi chức năng làm một việc cố định khác, thì ta chỉ cần nhổ con ROM trước đó ra và thay vào đó con IC có bộ nhớ ROM mới, thì hệ thống đó lại hoạt động theo nội dung của bộ nhớ cố định mới. Ví dụ như hiện nay máy vi tính có bộ nhớ ROM ghi ký tự chữ viết tiếng Anh, nếu ta thay con ROM tiếng Anh đó bằng một con IC nhớ ROM tiếng Việt, thì ta sẽ tạo ra bàn phím tiếng Việt. Khi tạo ra bàn phím tiếng Việt được, thì trên cơ sở đó ta có thể tạo ra bộ nhớ bàn phím cho tiếng Trung Quốc hay Hàn Quốc... đều được. Như vậy về phần cứng ta có thể làm thay đổi bộ mã cho bàn phím của máy tính, hay thay đổi trình tự điều khiển của một chức năng hay máy công cụ bằng cách thay đổi ROM của nó.

Để hiểu rõ bộ nhớ chỉ đọc ROM ta đi sâu vào cấu tạo của nó. ROM cố định có ba phần cơ bản : bộ giải mã địa chỉ, ma trận phần tử nhớ, và mạch điện ở đầu ra.

Bộ giải mã địa chỉ dùng để tìm địa chỉ của một ô nhớ cần đọc ra trong ROM. Điều đó cho phép ta đọc ngẫu nhiên nội dung của từng ô nhớ đã được ghi cố định, cũng như ta có thể đọc lần lượt từ ô (00 ... 0) đến ô cuối cùng (11 ... 1) của ROM. Nội dung được đọc ra là tín hiệu điều khiển cụ thể cho hệ thống số.

Ma trận phần tử nhớ dùng ghi giữ thông tin cố định cần được nhớ, nó có thể được xây dựng bằng điôt hay tranzitor kích cỡ của nó phụ thuộc vào nội dung cần ghi giữ và độ dài từ mã các bit thông tin. Mạch điện đầu ra dùng để đưa tín hiệu (nội dung thông tin) được đọc ra từ bộ nhớ tới thiết bị chấp hành. Người ta có thể dùng ROM thiết kế ra mạch tổ hợp cũng như mạch dãy.

Mạch ROM đơn giản nhất dùng điôt được chỉ ra trên hình 6.12.



Hình 6.12. Cấu tạo của ROM dùng điôt.

Từ sơ đồ mạch điện của ROM đơn giản nhất với hai đầu vào (A, B) ta có quan hệ :

$$W_0 = \overline{\overline{A} . \overline{B}} \quad ; \quad W_1 = \overline{\overline{A} . B} \quad ; \quad W_2 = \overline{A . \overline{B}} \quad ; \quad W_3 = \overline{A . B}$$

Ma trận nhớ là ma trận điôt điện trở, nó bao gồm các mạch VÀ dùng điôt – điện trở, nó hoạt động giống như bộ giải mã thông thường dùng điôt (bộ giải mã số thứ tự đầu ra tương ứng mã nhị phân đưa vào). Nguyên lý làm việc của ROM giống như bộ giải mã nhưng nó không phải là bộ giải mã, vì ở đây các điôt lắp trong mạch sắp xếp không theo một trật tự nào cả, mà các điôt đó sắp xếp (hàn mạch) theo một nội dung chủ định trước. Nội dung đó có thể là một mã nhị phân luôn cố định (ví dụ có nội dung ghi là (0101) hay (1011)), có thể là một câu lệnh điều khiển hay một hàm lôgic nào đó cần có trước và sẽ dùng cố định.

Các mã nhị phân (A, B) cho trước đưa vào là một hàm số cố định hay một địa chỉ không thay đổi ta sẽ nhận được và lấy nội dung ra ở trên đầu ra của bộ nhớ là (Y_3, Y_2, Y_1, Y_0). Với các cách nối cụ thể các điôt như vậy ta có thể tìm được bộ mã người ta đã ghi vào ROM như sau :

$$Y_3 = \overline{W_1 . W_3} = \overline{\overline{A}\, B . A\overline{B}} = \overline{A} . B + A . B = B$$

$$Y_2 = \overline{W_0 . W_2 . W_3} = \overline{\overline{A} . \overline{B} . A\overline{B} . A\overline{B}} = \overline{A} . \overline{B} + A . \overline{B} + A . B = A + \overline{B}$$

$$Y_1 = \overline{W_1 . W_3} = B$$

$$Y_0 = \overline{W_0 . W_1} = \overline{A} . \overline{B} \; . \; \overline{\overline{A} . \overline{B}} = \overline{A} . \overline{B} + \overline{A} . B = \overline{A}$$

Dưới tác động của tín hiệu vào (A, B) hay còn được gọi là mã địa chỉ, đầu ra của bộ nhớ ROM (Y_3, Y_2, Y_1, Y_0) được xác định ở bảng chân lý hay bảng chức năng chỉ ra trên hình 6.13.

A	B	Y_3	Y_2	Y_1	Y_0
0	0	0	1	0	1
0	1	1	0	1	1
1	0	0	1	0	0
1	1	1	1	1	0

Hình 6.13. Nội dung ghi trước của ROM.

Từ bảng thật của bộ nhớ cố định ROM chỉ ra trên hình 6.13 thực chất là bộ chuyển mã từ mã đầu vào (A, B) thành mã đầu ra (Y_3, Y_2, Y_1, Y_0), với một đặc điểm quan trọng là chuyển đổi một mã ít bit thành một mã nhị phân nhiều bit hơn mà thôi.

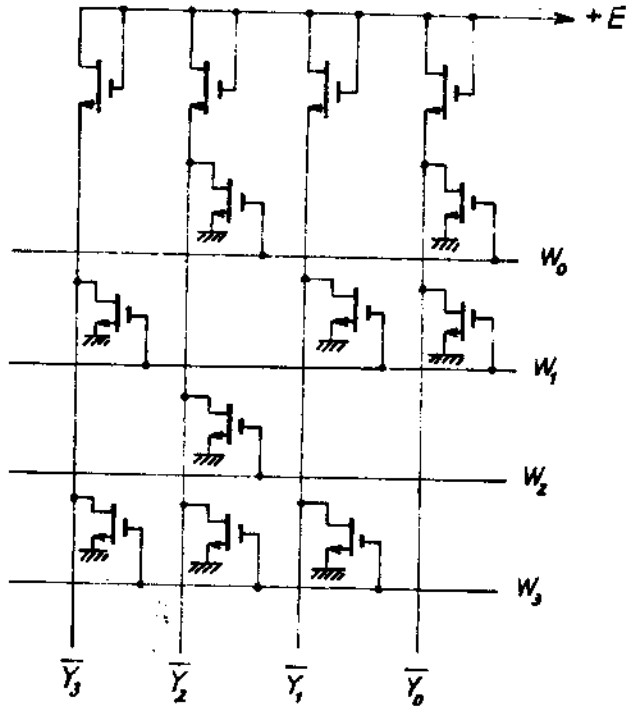
Nguyên lý làm việc của ROM là : từ bộ giải mã địa chỉ (bộ giải mã thông thường) thực hiện chuyển mã nhị phân đưa vào (A, B) thành ra bốn cổng NAND tạo ra các đường dây từ mã còn được gọi là đường dây từ, cũng là tín hiệu vào cho ma trận nhớ điôt - điện trở. Ma trận nhớ điôt - điện trở thực chất đó là các mạch VÀ điôt - điện trở được nối theo yêu cầu cho trước, đầu ra các mạch VÀ đó là các đường dây bit, mỗi một đường dây bit tương ứng với một đầu ra của một mạch VÀ điôt - điện trở, nó được ký hiệu tương ứng là Y_i. Các đầu ra Y_i - (Y_3, Y_2, Y_1, Y_0) tạo ra tập các tín hiệu ra tương ứng với nội dung của bộ nhớ cần được đọc ra, theo yêu cầu của tín hiệu vào (A, B).

Quan hệ giữa đầu vào và đầu ra của bộ nhớ ROM là quan hệ tương ứng một một. Vì vậy, người ta thường coi như mã đầu vào (A, B) là mã địa chỉ của ô nhớ, mà ô nhớ đó có nội dung (Y_3, Y_2, Y_1, Y_0) được ghi nhớ cố định (bằng nối điôt sẵn) từ trước. Như vậy muốn có nội dung ô nhớ cố định ROM khác đi thì ta cần phải nối các điôt đó khác đi, còn khi đọc thì ta chỉ cần cho tín hiệu tương ứng ở đầu vào (A, B). Với nguyên lý đơn giản như vậy của một bộ nhớ ROM ta có thể xây dựng được các bộ nhớ cố định có số lượng địa chỉ (tín hiệu vào) lớn hơn, và dung lượng của bộ nhớ cũng như nội dung của từng ô nhớ nhiều hơn.

Trong ma trận phần tử nhớ, đường ngang dùng để chọn từ mã nên gọi tắt là dây từ, còn đường dọc để chọn bit thông tin nên gọi là đường dây bit. Chỗ giao nhau giữa dây từ và dây bit gọi là phần tử nhớ. Tại phần tử nhớ, nếu có điôt thì dữ liệu được ghi là lôgic 1, còn nếu không có điôt thì số liệu lưu giữ được nhận là lôgic 0. Nội dung dữ liệu lưu giữ trong ma trận nhớ là không thể thay đổi sau khi các điôt được "làm cố định", hay được hoàn thành chế tạo ra ROM từ nhà máy. Điều đó làm cho bộ nhớ cố định ROM chỉ có thể đọc nội dung ra mà thôi.

Phần trên hình 6.12 ma trận phần tử nhớ người dùng điôt để thực hiện (chế tạo), nhưng ma trận phần tử nhớ chỉ ra ở phần trên đó có thể được thực hiện bằng tranzito trường MOS được mô tả trên hình 6.13.

Tại vị trí giao nhau giữa dây từ và dây bit người ta dùng phần tử nhớ là tranzitor MOS. Phần tử nhớ là nơi giao nhau đó sẽ có nội dung lôgic là 1 nếu có tranzitor MOS, còn nếu không có tranzitor thì nhận nội dung lôgic 0. Chỉ lưu ý các đường dây thông tin ở mức điện thế cao ứng với lôgic 1.



Hình 6.13. Ma trận phần tử nhớ dùng tranzitor MOS.

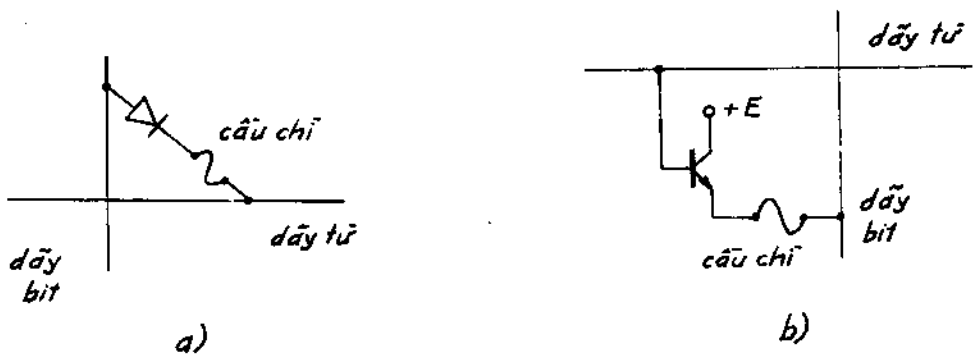
Muốn chế tạo ra một ma trận nhớ của ROM người ta căn cứ vào nội dung cần lưu giữ thông tin (biết trước) mà thiết kế ra "mặt nạ" dùng trong phương pháp quang khắc để chế tạo ra IC. Sau khi có mặt nạ người ta chế tạo ra bộ nhớ ROM hàng loạt (cùng một nội dung ghi giữ) cho phép giảm giá thành khi sản xuất với số lượng lớn.

Nhưng nếu số lượng sản xuất không lớn người ta dùng bộ nhớ PROM sẽ kinh tế hơn. Bộ nhớ PROM là bộ nhớ chỉ đọc, nhưng có thể "ghi nội dung" bằng cách đặc biệt, mà người sử dụng có thể tự ghi lấy theo một nội dung riêng. Nó hay được dùng trong máy công cụ tự động hay máy chuyên dụng. Để hiểu và sử dụng PROM ta tìm hiểu cấu tạo của nó.

6.2.2. BỘ NHỚ CHỈ ĐỌC CÓ THỂ TỰ GHI (PROM)

Phần quan trọng nhất của bộ nhớ PROM là cấu tạo của phần tử "nhớ", là chỗ giao nhau của đường dây từ và đường dây bit. Khi nắm được cấu tạo của phần tử nhớ thì có thể dễ dàng sử dụng (ghi vào) bộ nhớ PROM. Phần tử nhớ của PROM

có thể chế tạo bằng điôt hay dùng tranzitor loại TTL được chỉ ra trên hình 6.14.



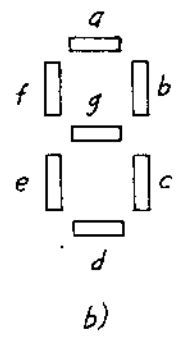
Hình 6.14. Cấu tạo phần tử nhớ PROM.

IC PROM trắng là con IC mà tất cả các phần tử nhớ (chỗ giao nhau của hai đường dây tọa độ) của nó đều có điôt hoặc tranzitor, mà chúng được nối tiếp với một dây sợi "cầu chì". Nếu ta muốn ghi nội dung cần nhớ cố định của mình vào bộ nhớ PROM trắng, thì vấn đề chỉ còn là : loại bỏ hay "đốt" (làm đứt) phần tử nhớ tương ứng với tọa độ của đường dây từ giao với đường dây bit là xong. Quá trình loại bỏ, làm đứt hay "đốt" phần tử nhớ thực hiện dễ dàng bằng cách cho dòng điện vào đủ lớn làm đứt dây "cầu chì" của phần tử nhớ thích hợp. Do có tọa độ đường dây từ và đường dây bit rất chính xác nên ta có thể đốt bỏ được phần tử nhớ chủ động theo yêu cầu. Khi đã bị đốt bỏ hay làm đứt "cầu chì" thì tại vị trí nhớ đó phần tử nhớ có giá trị lôgic là 0 (còn lại không bị đốt bỏ lôgic là 1). Khi đã đốt bỏ phần tử nhớ ở trong lòng con IC thì ta không thể hồi phục lại được nữa. Cho nên PROM là bộ nhớ cố định chỉ đọc, chỉ thực hiện được ghi thông tin vào một lần. Mỗi một IC nhớ PROM cần có một dòng điện tử lớn theo

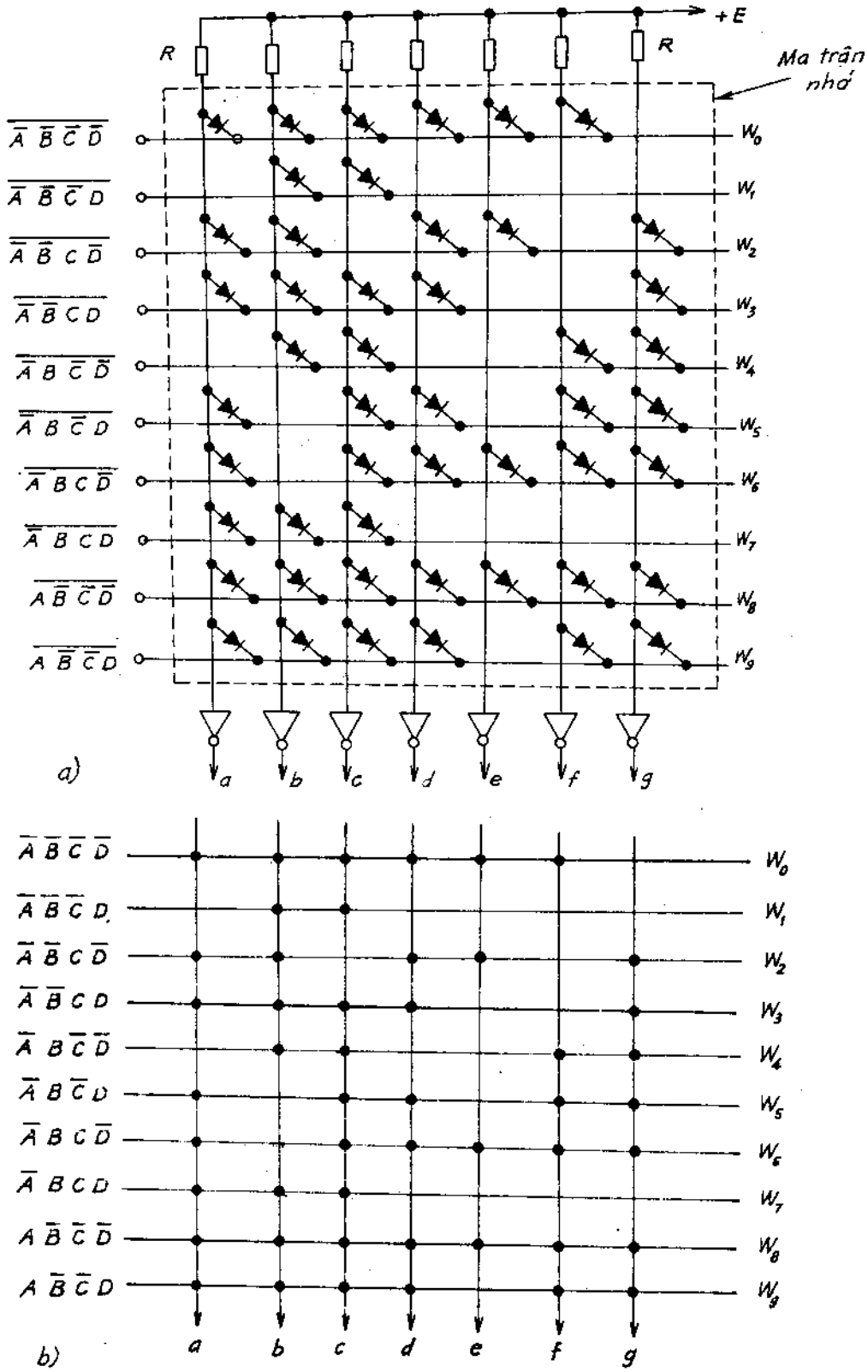
Vào nhị phân | Ra 7 thanh

Chữ số	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

a)



Hình 6.15. Chức năng chỉ thị.



Hình 6.16. Sơ đồ bộ giải mã 7 thanh dùng ROM.

quy định để làm đứt "cầu chì", dòng điện đó cũng như chỉ tiêu kỹ thuật của một con IC nhớ PROM được cho sẵn trong sổ tay tra cứu.

Ngoài cách thức dùng "cầu chì" để loại bỏ phần tử nhớ của PROM, người ta còn dùng hiệu ứng Sốtky để thay thế "cầu chì" bằng cách dùng các điôt Sốtky. Khi xuất xưởng ta có PROM trắng các điôt Sốtky đều ngắt và hở ra tương ứng với bit 0. Muốn tạo ra phần tử nhớ ở giá trị lôgic 1, người ta phải cho vào IC hay cho vào điôt Sốtky một điện áp ngược đủ lớn, để đánh thủng mặt ghép của nó tạo thành chập cực của điôt vĩnh viễn, gây nối mạch cố định mãi. Loại PROM này cũng chỉ ghi được một lần.

Ứng dụng của ROM hay PROM rất phong phú, nhưng chủ yếu tạo ra bộ nhớ ROM trong máy vi tính. Ngoài ra chúng còn dùng để thiết kế ra mạch tổ hợp. Sau đây sẽ nói tới một ứng dụng cụ thể của bộ nhớ chỉ đọc ROM hay PROM.

Ví dụ dùng ROM hay PROM xây dựng bộ giải mã 7 thanh chỉ thị các chữ số của cơ số 10. Bảng chức năng chỉ thị được chỉ ra trên hình 6.15.

Thực hiện bảng chức năng đó người ta dùng bộ nhớ ROM có ma trận nhớ được chỉ ra ở trên hình 6.16.

Sơ đồ trên hình 6.16,b đó là sơ đồ ký hiệu vẽ cho đơn giản của bộ nhớ ROM dùng xây dựng bộ giải mã nhánh (7 thanh) chỉ thị các chữ số của cơ số mười. Còn sơ đồ chỉ ra ở hình 6.16,a là sơ đồ cấu tạo hay mạch điện cụ thể chế tạo của bộ nhớ ROM dùng điôt - điện trở thực hiện bộ giải mã 7 thanh. Nguyên lý làm việc của sơ đồ mạch điện này giống hệt như nguyên lý làm việc của sơ đồ mạch điện của bộ nhớ ROM đơn giản nhất, với hai đầu vào (A, B) đã được nêu ở phần trên.

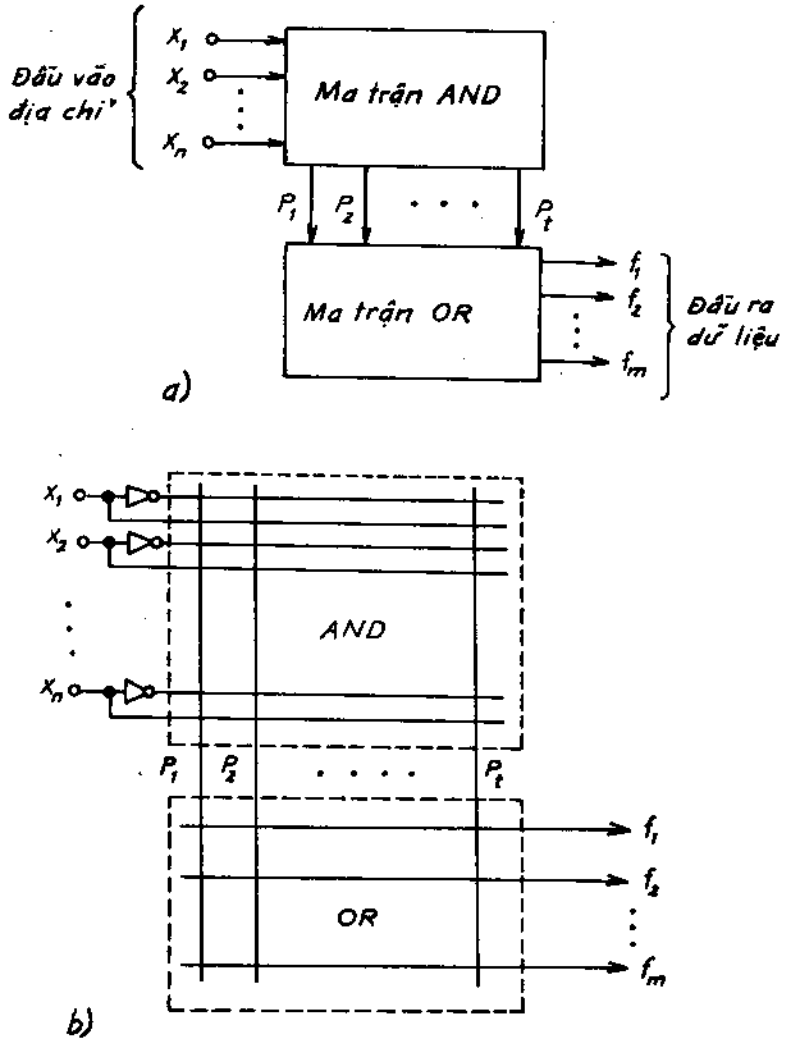
Với cách làm như vậy có thể dùng bộ nhớ ROM hay thiết kế PROM thực hiện các bộ chỉ thị khác nhau bất kỳ. Như có thể dùng PROM chế ta bộ chỉ thị ký tự chữ viết, chế ta các bảng quảng cáo hiển thị theo ý ta. Nếu dùng số lượng lớn các bộ nhớ chỉ đọc PROM có thể hiển thị cả một nội dung câu chữ hay bài khóa, nếu nối với bảng chỉ thị điểm sáng có thể tạo ra các bảng chữ chạy có nội dung của bài khóa. Tất nhiên do ROM cũng như PROM là bộ nhớ chỉ đọc nội dung ra, cho nên các ký tự cũng như các hiển thị hay bộ chỉ thị sẽ có nội dung không thay đổi mà được lặp đi lặp lại mãi mà thôi. Nếu muốn chỉ thị nội dung khác phải thiết kế lại bộ ROM hay PROM mới mà thay vào.

6.2.3. MẢNG LÔGIC LẬP TRÌNH PLA (PROGRAMMABLE LOGIC ARRAY)

Mảng lôgic lập trình PLA về mặt cấu tạo cũng giống như bộ nhớ chỉ đọc ROM gồm có hai ma trận là : ma trận các mạch AND, ma trận các mạch OR. Nhưng chúng khác nhau ở chỗ là : đầu ra ma trận AND của PLA là các tích lôgic của n biến số ứng với n tín hiệu vào từ (X_1 ÷ X_n), còn đầu ra của ROM có thể là tích số, nhưng tích số đó có thể không cần đầy đủ n biến số của tín hiệu vào.

Việc chế tạo ra PLA dựa trên phương trình tuyển chuẩn tắc toàn phần đầy đủ, bao gồm có các tích toàn phần dựa vào ma trận các mạch AND. Còn ma trận các mạch OR

chính là dạng tuyển hay tổng của các tích đó. Từ đó sơ đồ tổng quát của mạch PLA được thấy trên hình 6.17. Mảng lôgic PLA cho phép lập trình với cả hai ma trận AND và ma trận OR, trong khi đó ở ROM ta chỉ được lập trình với ma trận OR.



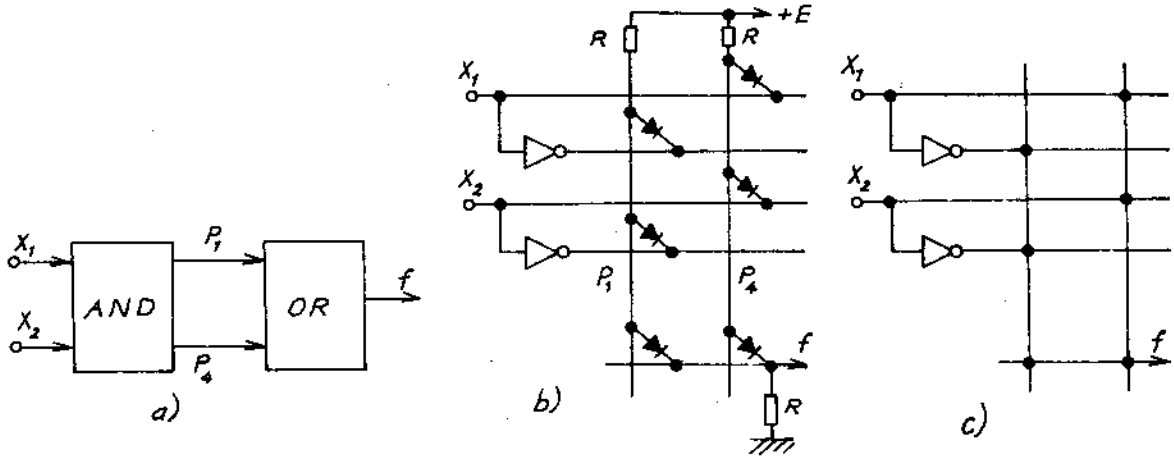
Hình 6.17. Sơ đồ tổng quát của mảng lôgic PLA.

Cấu tạo một mạch PLA đơn giản nhất gồm có 2 đầu vào (X_1, X_2) dùng điôt – điện trở được chỉ ra trên hình 6.18.

Từ sơ đồ mạch điện PLA ta thấy ma trận AND chỉ cho ra hai tích số $P_1 = \overline{X}_1 . \overline{X}_2$ và $P_4 = X_1 . X_2$. Còn trong ma trận OR chỉ có một tổng $f = P_1 + P_4$ (trong khi đó có thể có những tổng khác). Như vậy mạch PLA đó thực hiện hàm số :

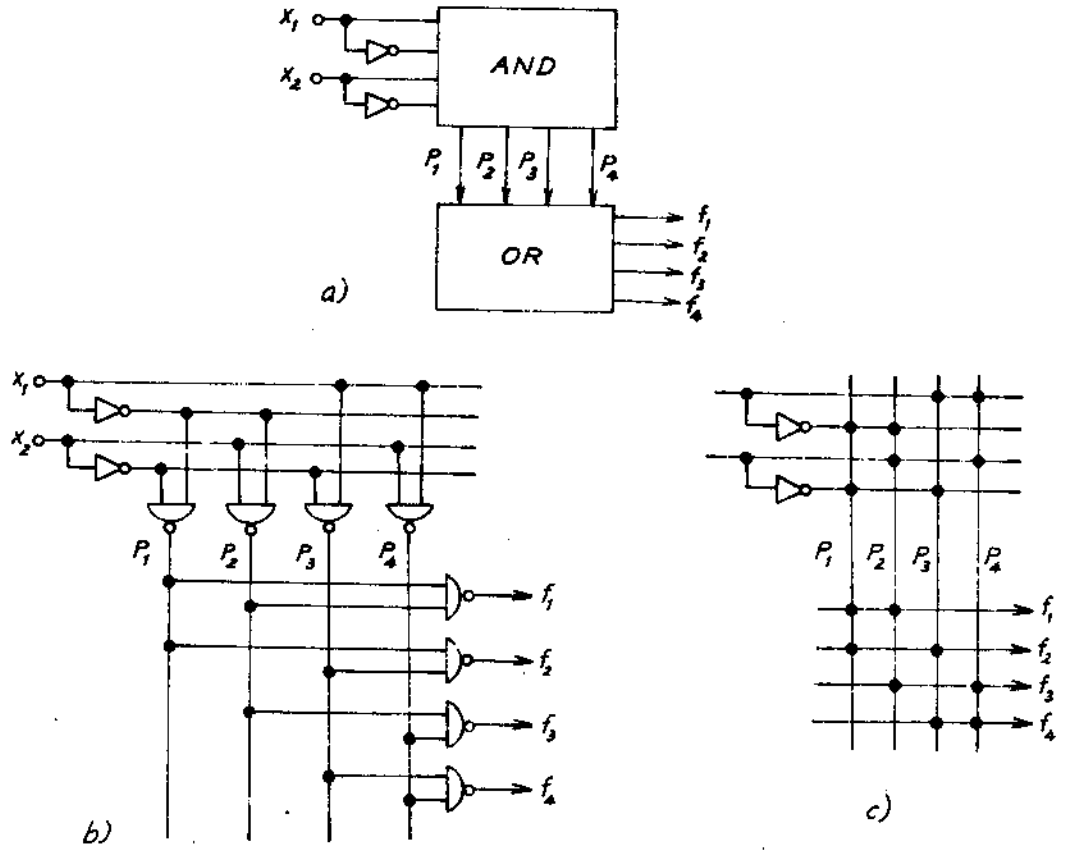
$$f = \overline{X}_1 . \overline{X}_2 + X_1 . X_2$$

Phương trình ta nhận được đó chính là bộ so sánh một bit, hay có thể dùng PLA thực hiện chức năng so sánh.



Hình 6.18. Mạch PLA đơn giản nhất.

Sơ đồ của một mạch PLA hai đầu vào (X_1, X_2) đầy đủ dùng loại TTL được chỉ ra trên hình 6.19.



Hình 6.19. Mạch PLA tạo hàm lôgic.

Từ sơ đồ của mạch điện nguyên lý có quan hệ :

$$P_1 = \overline{\overline{X}_1 . \overline{X}_2} \qquad P_2 = \overline{\overline{X}_1 . X_2} \qquad P_3 = \overline{X_1 . \overline{X}_2} \qquad P_4 = \overline{X_1 . X_2}$$

Đầu ra của PLA ta có :

$$f_1 = \overline{P_1 . P_2} = \overline{(\overline{\overline{X}_1 . \overline{X}_2}) . (\overline{\overline{X}_1 . X_2})} = \overline{X}_1 . \overline{X}_2 + \overline{X}_1 . X_2$$

$$f_2 = \overline{P_1 . P_3} = \overline{(\overline{X_1} . \overline{X_2}) . (X_1 . \overline{X_2})} = \overline{X}_1 . \overline{X}_2 + X_1 . \overline{X}_2$$

$$f_3 = \overline{P_2 . P_4} = \overline{(\overline{X_1} . X_2) . (\overline{X_1} . X_2)} = \overline{X}_1 . X_2 + X_1 . X_2$$

$$f_4 = \overline{P_3 . P_4} = \overline{(X_1 . \overline{X_2}) . (\overline{X_1} . X_2)} = X_1 . \overline{X_2} + X_1 . X_2$$

Dầu ra f_i của PLA nhận được kết quả là tổng của các tích số, từ quan hệ đó ta lập bảng chức năng cho các hàm được tạo ra như sau, ở trên hình 6.20.

X_1	Y_2	f_1	f_2	f_3	f_4
0	0	1	1	0	0
0	1	1	0	1	0
1	0	0	1	0	1
1	1	0	0	1	1

Hình 6.20. Bảng chức năng các hàm được tạo ra.

Sau khi đã có mảng lập trình lôgic PLA lắp ráp hay nối mạch theo yêu cầu tạo hàm, nếu muốn có hàm lôgic nào ta chỉ cần sử dụng đầu ra f_i ($1 \leqslant i \leqslant m$) tương ứng của hàm lôgic đó. Như vậy một ma trận PLA ta có thể có nhiều hàm lôgic khác nhau hay có thể điều khiển một lúc nhiều chức năng khác nhau là bộ PLA với cùng một địa chỉ đầu vào (một tập tín hiệu vào) X sẽ cho ra tín hiệu điều khiển thích hợp trên các đầu ra F.

Với : $X = (X_1, X_2, \ldots, X_n)$ - Tập tín hiệu vào

$F = (f_1, f_2, \ldots, f_m)$ - Tập đáp ứng ra.

Tập tín hiệu đáp ứng ra f_i có thể là tập tín hiệu điều khiển chức năng cho các chức năng khác nhau, nhưng vẫn tập tín hiệu ra đó lại có thể được coi là các bộ mã được "đọc ra" theo địa chỉ của tín hiệu vào.

Ví dụ : Với địa chỉ (X_1, X_2) là :

(0 0) ta có bộ mã (1 1 0 0)

(1 0) ta có bộ mã (0 1 0 1)

Như vậy mảng lôgic lập trình PLA ngoài khả năng tạo hàm lôgic điều khiển chức năng, thì nó còn có thể thực hiện nhiệm vụ tạo mã theo yêu cầu sử dụng của ta. Qua đó thấy việc sử dụng mạch PLA trong thực tế kỹ thuật là rất phong phú và có hiệu quả.

Phần trên đã trình bày phương pháp tạo hàm dùng bộ PLA đơn giản với hai đầu vào (X_1, X_2). Bằng cách làm tương tự có thể tạo ra các hàm số phức tạp hơn, điều khiển các chức năng lớn hơn với số lượng tín hiệu vào điều khiển nhiều hơn. Thông thường PLA hay được dùng làm mạch điều khiển lôgic chương trình, do đó độ tin cậy hoạt động của PLA là cần thiết. Nói cách khác độ tin cậy hoạt động cũng như chất

lượng của hàm lôgic được tạo ra ở đây là rất quan trọng nhưng còn ít được quan tâm đến. Vì cùng một chức năng cùng một nhiệm vụ điều khiển có thể thực hiện được bằng nhiều mạch điện khác nhau, hay tổ hợp trong PLA khác nhau, vậy cách tổ hợp nào tốt hơn hoạt động có độ tin cậy cao hơn (nhất là khi số lượng tập biến vào – tín hiệu vào rất lớn) là điều ta cần phải nghĩ đến. Nói cách khác là cần phải đánh giá được chất lượng làm việc của mạch PLA hoạt động trong hệ thống số, đây là một vấn đề không đơn giản vì nó chỉ được nói đến khi có một mạch PLA với một chức năng cụ thể.

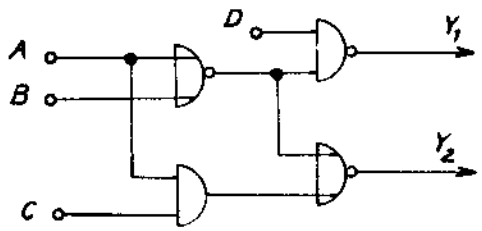
Các mạch PLA ta có thể tìm mua trên thị trường, chỉ tiêu kỹ thuật của mạch có thể tra cứu trong sổ tay (catalog). Dựa vào đó ta có thể sử dụng và điều khiển hoạt động của mạch PLA theo nhiệm vụ đề ra. Vài mạch tổ hợp của PLA là TMS 2000, TMS 2200.

Kỹ thuật số là một lĩnh vực khoa học mới và đang ngày càng phát triển mạnh mẽ, vì vậy quyển sách này cũng không thể nêu hết được mọi vấn đề. Hy vọng thời gian tiếp theo quyển "Toán logic và kỹ thuật số" sẽ được bổ sung thêm các kiến thức mới để nội dung ngày càng phong phú hơn.

PHẦN BÀI TẬP

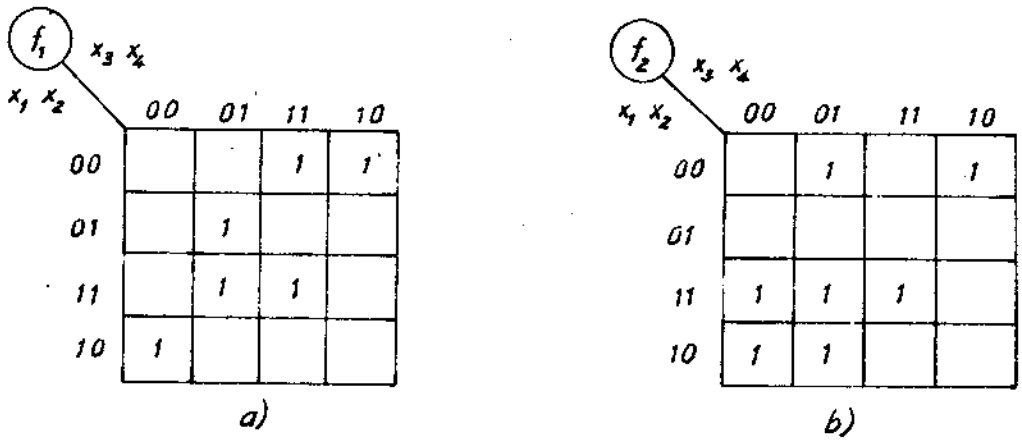
Phần ôtômat không có nhớ

1) Phân tích mạch tổ hợp cho ở hình B.1 sau :



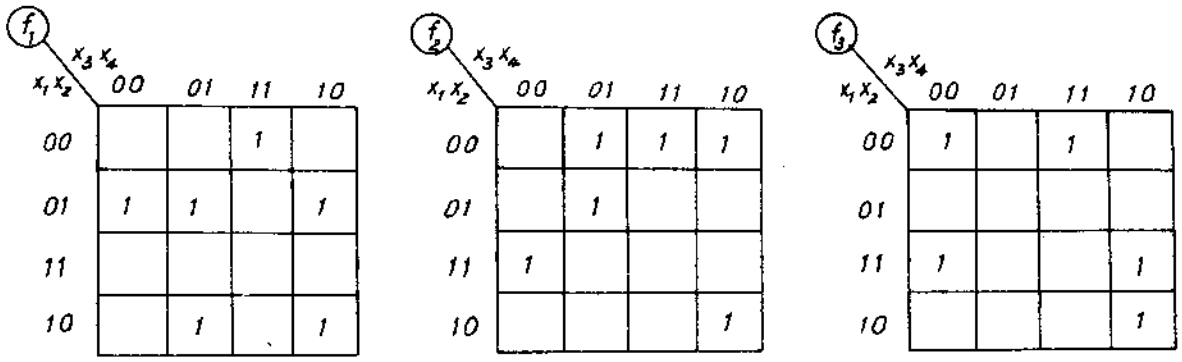
Hình B.1. Một ôtômat không có nhớ.

2) Tổng hợp mạch tổ hợp dùng NAND có chức năng cho ở bìa Kácnô trên hình 3.2 sau.



Hình B.2. Cho hàm số.

3) Thiết kế hệ tổ hợp thực hiện ba mạch tổ hợp có chức năng chỉ ra trên hình B.3 sau.



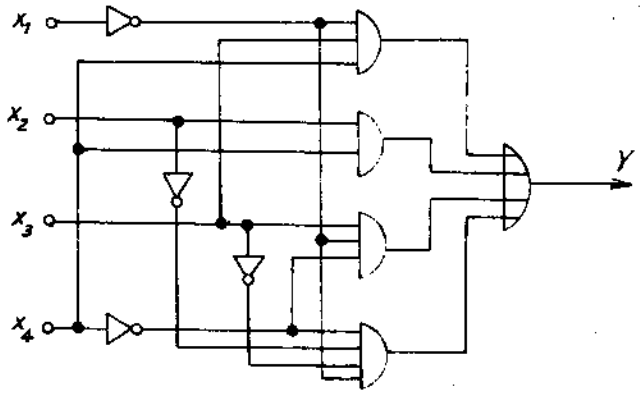
Hình B.3. Cho hệ tổ hợp.

5) Rút gọn hàm số và dùng mạch NAND thực hiện chức năng sau :

$$f_1(x_1, x_2, x_3) = (0, 2, 4, 6, 7)$$

$$f_2(x_1, x_2, x_3) = (2, 3, 4, 6, 7)$$

6) Đơn giản hóa mạch điện lôgic trên hình B.4 sau :



Hình B.4. Có một mạch lôgic.

7) Tìm mạch tổ hợp mô tả hàm số sau sao cho khi lắp mạch, số linh kiện phải sử dụng là ít nhất.

$$f_4(X) = \bar{x}_1\bar{x}_2\bar{x}_3x_4 + x_1x_2\bar{x}_3x_4 + x_1\bar{x}_2\bar{x}_3\bar{x}_4 + \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4 +$$

$$+ \bar{x}_1x_2x_3x_4 + x_1x_2x_3\bar{x}_4 + \bar{x}_1x_2\bar{x}_3\bar{x}_4 + x_1\bar{x}_2x_3x_4$$

8) Thực hiện chức năng lôgic sau bằng các loại mạch khác nhau.

$$f_4(X) = (1, 3, 4, 5, 6, 7, 12, 13)$$

9) Xây dựng mạch tổ hợp 4 đầu vào và 3 đầu ra bằng mạch NAND của ba hàm số sau :

$$f_1(x_1, x_2, x_3, x_4) = (3, 4, 5, 7, 13, 15)$$

$$f_2(x_1, x_2, x_3, x_4) = (3, 8, 9, 11, 13, 15)$$

$$f_3(x_1, x_2, x_3, x_4) = (2, 8, 9, 11, 13, 14)$$

10) Thực hiện mạch tổ hợp báo chẵn bit cho một mạch điện có 5 tín hiệu vào. Và một mạch điện báo bit chẵn là 2 bit trong 5 đầu tín hiệu vào.

Phần ôtômat có nhớ

11) Cho hai ví dụ mô tả hoạt động của ôtômat có nhớ dùng bảng Mealy và hai ví dụ dùng bảng Moore.

12) Chứng minh rằng mỗi ôtômat Mealy có thể chuyển đổi thành ôtômat Moore và ngược lại.

13) Tìm tập phân hoạch thế của các ôtômat M_1 và M_2 cho ở bảng chức năng trên hình B.5 sau, đồng thời vẽ dàn phân hoạch thế cho các ôtômat đó.

Bảng M_1

S \ X	a	b	Z: a	Z: b
A	C	B	0	1
B	A	D	1	0
C	A	A	1	0
D	B	B	0	1

a)

Bảng M_2

S \ X	a	b	c	Z: a	Z: b	Z: c
A	D	F	B	0	1	0
B	C	E	A	0	1	0
C	E	A	D	0	1	0
D	F	B	C	1	1	0
E	A	D	F	1	1	0
F	B	C	E	1	1	0

b)

Hình B.5. Cho hoạt động của hai ôtômat.

14) Tìm các phân hoạch thế tương đương (PHTTĐ) của hai ôtômat M_1 và M_2 có bảng chức năng trên hình B.6 sau.

Tìm các phần tử sinh cơ bản và vẽ dàn phân hoạch thế tương đương D_M^* cho các ôtômat đó. Tối tiểu hóa trạng thái cho hai ôtômat đó.

Bảng M_1

S \ X	0	1	Z: 0	Z: 1
A	C	D	1	1
B	B	C	0	1
C	C	A	1	0
D	D	C	0	0
E	E	C	0	0
F	F	C	0	1

a)

Bảng M_2

S \ X	0	1	Z: 0	Z: 1
A	A	G	0	1
B	B	D	0	0
C	D	E	1	0
D	G	E	1	1
E	E	G	0	1
F	F	D	0	0
G	C	F	0	1

b)

Hình B.6. Có hai ôtômat.

15) Tối tiểu hóa trạng thái của các ôtômat không hoàn toàn xác định cho ở trên hình B.7 sau.

Bảng chức năng

S \ X	a	b	c
A	A,1	F,0	C,0
B	-	G,0	C,0
C	A,0	G,0	B,0
D	A,0	H,1	E,1
E	B,0	H,1	D,1
F	C,1	I,0	D,0
G	C,1	I,0	E,0
H	D,0	H,1	A,1
I	D,0	I,1	-
J	E,0	J,1	C,0

a)

Bảng chức năng

S \ X	a	b	c	d	e	f	g
A	-	E,1	A,-	A,0	D,0	B,0	-
B	D,1	-	A,-	B,0	A,-	A,-	-
C	D,1	-	-	B,0	A,1	-	G,0
D	E,-	B,-	-	-	-	B,0	A,-
E	E,1	-	E,-	B,-	A,-	B,-	A,1
F	C,-	H,1	G,0	B,0	-,1	F,1	-
G	C,1	E,1	G,0	-	-	-	F,0
H	E,0	B,0	E,-	A,1	D,1	B,-	A,1

b)

Hình B.7. Cho các ôtômat không xác định.

16) Chuyển đổi từ ôtômat Mealy thành bảng chức năng của ôtômat Moore cho các ôtômat cho ở hình B.8 sau.

S \ X	0	1	Z 0	Z 1
a	b	a	0	0
b	b	c	0	0
c	b	a	1	0

a)

S \ X	0	1	Z 0	Z 1
A	C	B	0	1
A	A	D	1	0
C	A	A	1	0
D	B	B	0	1

b)

Hình B.8. Cho hai ôtômat.

17) Chuyển đổi từ bảng chức năng ôtômat Moore thành bảng chức năng của ôtômat Mealy cho các ôtômat trên hình B.9.

S \ X	0	1	Z
a	a	c	0
b	a	b	0
c	d	b	0
d	a	c	1

a)

S \ X	0	1	Z
A	C	B	0
B	A	D	0
C	A	A	0
D	B	B	1
E	C	A	1

b)

Hình B.9. Cho các ôtômat.

18) Chứng minh rằng tồn tại một phân hoạch thế của ôtômat cho trên hình B.10 thỏa mãn điều kiện dùng một phân hoạch thế để mã hóa, đồng thời đưa ra một cách mã hóa cho ôtômat dùng phân hoạch đó.

Bảng chức năng

S \ X	a	b	c	Z
A	F	E	D	0
B	E	F	D	0
C	D	D	B	0
D	C	C	F	1
E	A	B	C	0
F	B	A	C	0

Hình B.10. Cho trước ôtômat.

19) Tối tiểu hóa các trạng thái của ôtômat cho trên hình B.11. Mã hóa trạng thái theo phương pháp mã hóa tối ưu.

Bảng chức năng

S \ X	a	b	c	Z a	Z b	Z c
A	A	F	C	1	0	0
B	-	G	B	-	0	0
C	A	G	B	0	0	0
D	A	H	E	0	1	1
E	B	H	D	0	1	1
F	C	I	D	1	0	0
G	C	J	E	1	0	0
H	D	H	A	0	1	1
I	D	I	-	0	1	-
J	E	J	C	0	1	0

Hình B.11. Có ôtômat.

20) Phân giải song song cho ôtômat chỉ ra trên hình B12 sau và xây dựng dàn phân hoạch thế cho nó.

Bảng chức năng

S \ X	0	1	Z
1	5	2	0
2	6	1	0
3	1	6	0
4	2	5	0
5	3	4	0
6	4	3	1

Hình B.12. Phân giải song song cho ôtômat.

21) Phân giải song song cho ôtômat chỉ ra trên hình B.13 sau và tìm các phần tử sinh cho nó. Vẽ đồ hình Moore cho ôtômat.

Bảng chức năng

S \ X	0	1	Z
1	4	3	0
2	6	3	0
3	5	2	0
4	2	5	0
5	1	4	0
6	3	4	1

Hình B.13. Cho trước ôtômat.

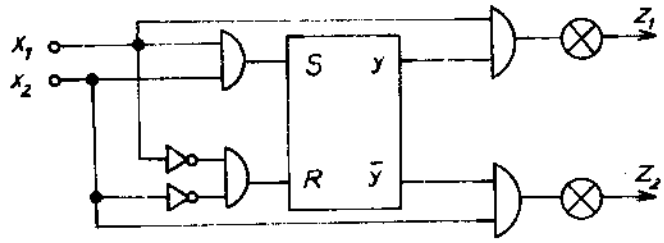
22) Phân giải nối tiếp cho ôtômat chỉ ra trên hình B.14 và vẽ dàn phân hoạch thế cho nó. Vẽ đồ hình Mealy cho ôtômat.

S \ X			Z	
	0	1	0	1
1	3	2	1	0
2	3	2	1	0
3	2	1	0	1
4	1	3	0	0
5	6	4	1	0
6	5	4	0	1

Hình B.14. Cho bảng nhiệm vụ ôtômat.

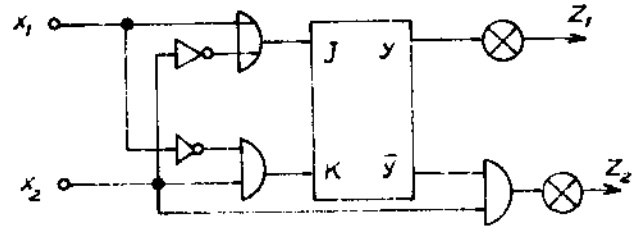
Phần thực hiện ôtômat có nhớ

23) Phân tích ôtômat có nhớ được chỉ ra trên sơ đồ ở hình B.15*a* sau:



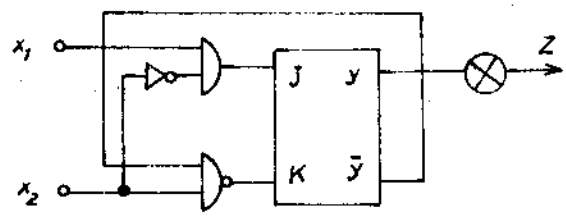
Hình B.15,*a*.Cho một mạch điện.

24) Tìm bảng chức năng hoạt động của ôtômat cho sơ đồ ở hình B.15,*b*:



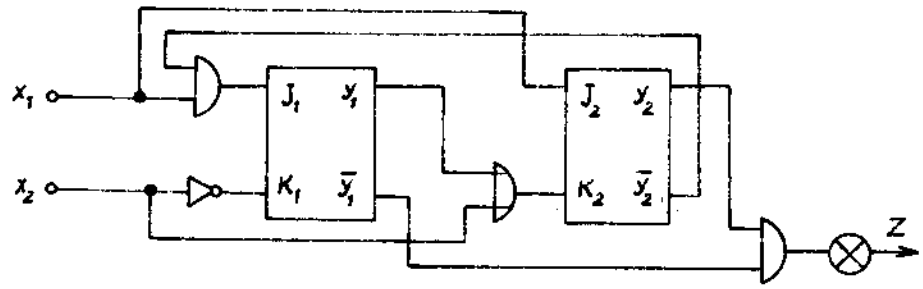
Hình B.15,*b*.Có ôtômat.

25) Phân tích hoạt động của sơ đồ mạch điện ôtômat có nhớ cho trên hình B.16 sau.



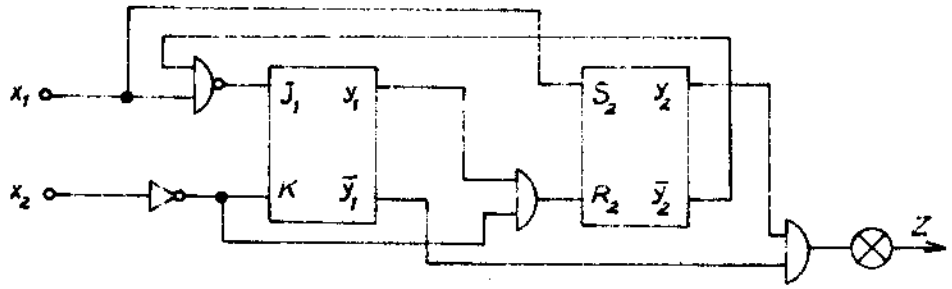
Hình B.16. Có sơ đồ mạch điện.

26) Phân tích ôtômat có nhớ ở hình B.17 sau.



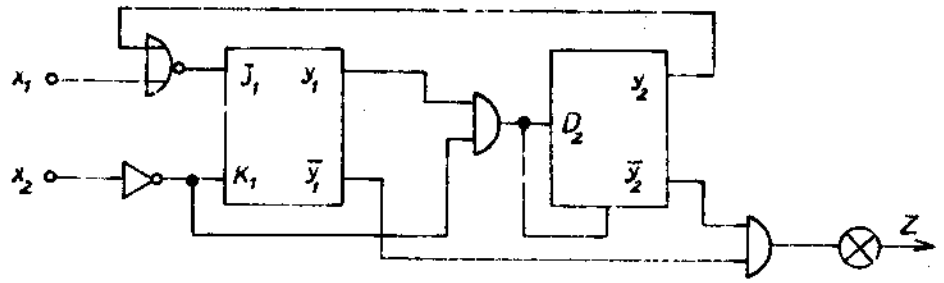
Hình B.17. Có ôtômat.

27) Tìm bảng chức năng cho ôtômat có nhớ ở hình B.18 sau.



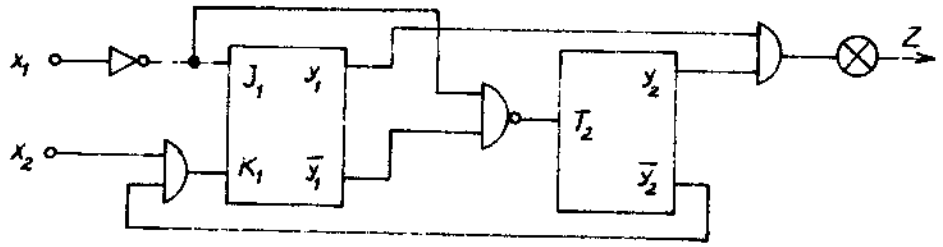
Hình B.18. Cho một mạch điện.

28) Tìm hoạt động chính xác của ôtômat cho trên hình B.19



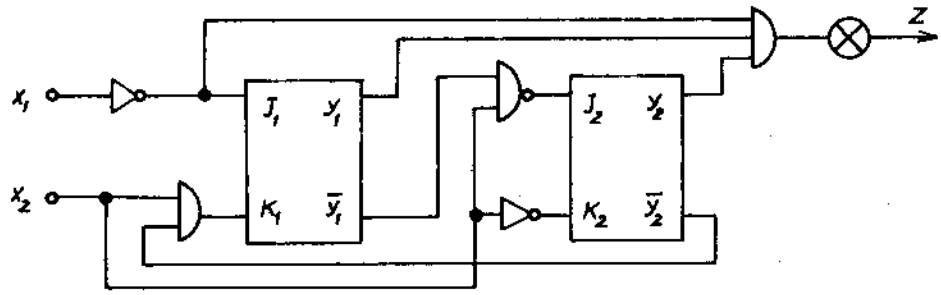
Hình B.19. Có mạch điện của ôtômat.

29) Phân tích ôtômat có nhớ cho ở hình B.20 sau.



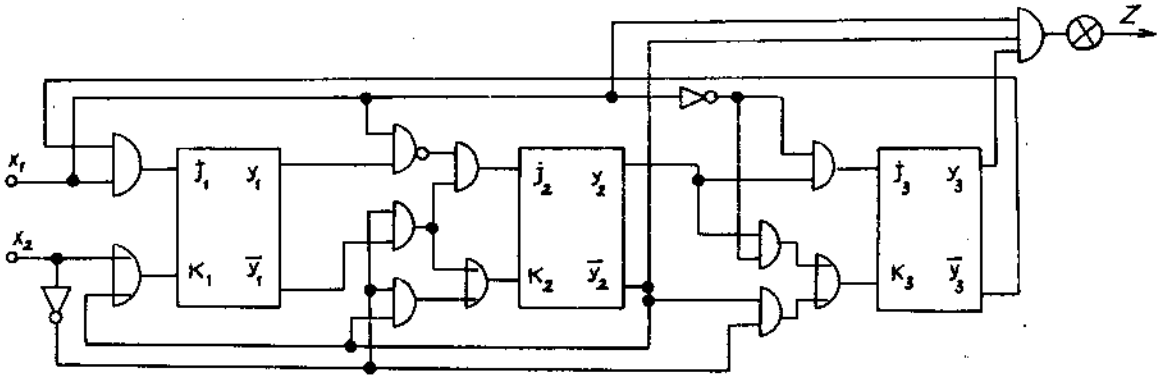
Hình B.20. Mạch điện ôtômat có nhớ.

30) Tìm bảng chức năng cho ôtômat có nhớ ở hình B.21 sau.



Hình B.21. Có mạch điện.

31) Phân tích chức năng hoạt động của ôtômat cho ở trên hình B.22 sau.



Hình B.22. Có một ôtômat.

32) Thực hiện ôtômat có nhớ chức năng hoạt động theo bảng chỉ ra trên hình B.23, dùng JKFF lắp ráp.

Bảng chức năng

S \ X			Z	
	0	1	0	1
A	C	B	0	1
B	A	D	1	0
C	A	A	1	0
D	B	B	1	0

Hình B.23. Cho ôtômat

33) Dùng trigơ D (DFF) thực hiện ôtômat có nhớ chức năng cho trên hình B.24 sau.

Bảng chức năng

S \ X	0	1	Z
A	C	B	0
B	A	D	1
C	A	A	0
D	B	C	0

Hình B.24. Chức năng của ôtômat.

34) Thực hiện ôtômat có nhớ theo bảng chức năng cho trên hình B.25. Dùng JKFF thực hiện.

S \ x_1x_2	00	01	11	10	Z
a	a	c	d	b	1
b	a	d	c	a	1
c	a	a	a	d	0
d	a	b	b	c	0

Hình B25. Cho một ôtômat.

35) Tổng hợp ôtômat có nhớ hoạt động theo bảng chức năng cho trên hình B.26. Mã hóa dùng một phân hoạch thế và dùng JKFF thực hiện.

Bảng chức năng

S \ X	0	1	Z
1	4	3	0
2	6	3	0
3	5	2	1
4	2	5	0
5	1	4	0
6	3	4	0

Hình B.26. Thực hiện ôtômat.

36) Thực hiện ôtômat cho ở hình B.26. Mã hóa theo nhị phân tăng dần và dùng trigơ D thực hiện mạch.

37) Cho ôtômat có nhớ hoạt động theo bảng chuyển biến trạng thái trên hình B.27 sau.
a) Dùng phương pháp phân hoạch thế phân giải song song ôtômat đã cho thành hai ôtômat con thành phần.
b) Thực hiện ôtômat sau khi đã phân giải bằng trigơ JK (JKFF).

Bảng chức năng

S \ x_1x_2	00	01	11	10	Z
a	a	e	j	d	0
b	b	f	h	i	0
c	a	g	e	b	-
d	b	h	f	a	0
e	b	a	b	h	0
f	b	b	a	j	1
g	a	c	d	f	0
h	a	d	i	e	1
i	a	g	e	b	0
j	a	i	d	f	-

Hình B.27. Tổng hợp ôtômat đã cho.

38) Tổng hợp so sánh liên tiếp 1 bit có hai đầu vào.

a) Tìm các trạng thái làm việc của ôtômat cần có bằng cách tổng hợp từ bộ so sánh đã cho.

b) Thành lập bảng chuyển biến trạng thái. Ôtômat đã tổng hợp là ôtômat kiểu gì ?

c) Vẽ đồ hình (graf) trạng thái.

39) Cho ôtômat hoạt động theo bảng chức năng hình B.28.

Bảng chức năng

S \ X			Z	
	0	1	0	1
A	C	B	1	1
B	G	-	0	-
C	F	A	0	0
D	D	E	0	0
E	K	I	0	0
F	D	E	1	1
G	F	J	1	1
H	G	H	1	1
I	G	H	0	0
J	C	I	1	1
K	F	J	0	0

Hình B.28. Cho ôtômat.

a) Dùng phương pháp phân hoạch thế để mã hóa trạng thái tối ưu cho ôtômat.

b) Thực hiện ôtômat đã cho dùng trigơ D (DFF).

40) Tổng hợp bộ tổng 1 bit có hai đầu vào (x_1, x_2) và một đầu ra Z. Các số nhị phân đưa vào liên tiếp và bắt đầu bằng chữ số có giá trị bé nhất.

a) Phân tích tìm các trạng thái làm việc của ôtômat cần tổng hợp từ bộ tổng đã cho. Đây là ôtômat kiểu gì ?

b) Lập bảng chuyển biến trạng thái cho ôtômat.

c) Vẽ đồ hình trạng thái (graf) của ôtômat.

41) Tổng hợp ôtômat có nhớ đồng bộ, có một đầu vào x và một đầu ra Z. Hoạt động theo yêu cầu sau.

- Tín hiệu vào là 0 hoặc 1 xuất hiện ngẫu nhiên liên tục kế tiếp nhau.
- Đầu ra Z = 1 nếu như gặp đúng dãy số vào kế tiếp nhau là (0010) hoặc (1101).
- Đầu ra Z = 0 trong mọi trường hợp khác.
- Dùng ôtômat Mealy mô tả bảng chức năng cho ôtômat. Thực hiện ôtômat bằng trigơ JK (JKFF).
- Mã hóa cho ôtômat theo bìa Kácnô.

42) Thực hiện ôtômat có nhớ đồng bộ, có một đầu vào x và một đầu ra Z. Hoạt động theo yêu cầu sau :

- Tín hiệu vào là 0 hoặc 1 xuất hiện ngẫu nhiên bất kỳ liên tục kế tiếp nhau.
- Đầu ra Z = 1 nếu như gặp được dãy số vào kế tiếp nhau là (0101) hoặc (1010).
- Đầu ra Z = 0 trong mọi trường hợp khác.
- Dùng ôtômat Moore mô tả chức năng cho ôtômat.
- Dùng một phân hoạch thế mã hóa cho ôtômat, sau đó thực hiện ôtômat bằng trigơ D (DFF).

43) Thực hiện ôtômat có nhớ đồng bộ với chức năng như của bài tập số 19 và bài tập số 20. Nhưng thực hiện mã hóa cho ôtômat theo mã nhị phân tăng dần.

44) Thực hiện ôtômat có nhớ có một đầu vào là x và một đầu ra Z. Hoạt động theo yêu cầu sau.

- Tín hiệu vào là 0 hoặc 1 xuất hiện ngẫu nhiên bất kỳ liên tục kế tiếp nhau.
- Đầu ra Z = 1 nếu như gặp dãy số vào kế tiếp nhau là (0 0 1 1 0).
- Đầu ra Z = 0 trong mọi trường hợp khác.
- Dùng ôtômat Mealy mô tả bảng chức năng.
- Mã hóa theo bìa Kácnô và dùng JKFF thực hiện.

45) Tổng hợp ôtômat có nhớ đồng bộ có một đầu vào x và một đầu ra Z. Hoạt động theo yêu cầu sau :

- Tín hiệu vào là 0 hoặc 1 xuất hiện ngẫu nhiên liên tục kế tiếp nhau.
- Đầu ra Z = 1 có tín hiệu báo nếu chỉ gặp được dãy số vào là (1 0 0 1 0).
- Đầu ra Z = 0 trong mọi tổ hợp khác.
- Dùng ôtômat Moore mô tả hoạt động.

- Mã hóa theo mã nhị phân thường và dùng JKFF lắp mạch.

46) Thiết kế một khóa của điện tử có một đầu vào x và một đầu ra Z mở chốt cửa. Hoạt động theo yêu cầu :

- Ở ngoài cửa có hai phím mầu xanh (0) và đỏ (1) được bấm vào ngẫu nhiên bất kỳ kế tiếp nhau.
- Chốt cửa (đầu ra Z = 1) sẽ mở nếu như bấm đúng tổ hợp tín hiệu là : "xanh đỏ đỏ đỏ xanh" - tức là (0 1 1 1 0).
- Chốt cửa sẽ không mở (Z = 0) và có thể báo động nếu bấm không đúng mã mở của khóa.
- Dùng ôtômat Moore để mô tả chức năng hoạt động của khóa điện tử.
- Mã hóa cho ôtômat theo phương pháp dùng một phân hoạch thế. Sau đó dùng trigơ JK (JKFF) lắp mạch.

47) Thực hiện ôtômat có nhớ đồng bộ có một đầu vào x và một đầu ra Z. Hoạt động theo yêu cầu sau.

- Tín hiệu vào là 0 hoặc 1 xuất hiện ngẫu nhiên bất kỳ kế tiếp nhau.
- Đầu ra Z = 1 sẽ báo nếu như gặp được tổ hợp tín hiệu vào là : (0 1 1 0 1) hoặc (1 0 0 1 0).
- Đầu ra Z = 0 nếu gặp các tổ hợp tín hiệu khác.
- Dùng ôtômat Moore mô tả hoạt động của ôtômat.
- Dùng cách mã hóa tối ưu để mã hóa cho ôtômat.

Sau đó thiết kế mạch điện dùng trigơ D (DFF).

48) Thực hiện một ôtômat có nhớ đồng bộ có một đầu vào x và một đầu ra Z. Hoạt động theo yêu cầu sau :

- Tín hiệu vào là 0 hoặc 1 xuất hiện ngẫu nhiên liên tục kế tiếp nhau.
- Đầu ra Z = 1 cho tín hiệu báo nếu như gặp dãy số vào là (0 1 0 0 1) hoặc (1 0 1 1 0).
- Đầu ra Z = 0 nếu gặp các tổ hợp khác.
- Dùng ôtômat Mealy để mô tả chức năng cho ôtômat.
- Dùng phương pháp mã hóa chống chạy đua y (chạy đua trạng thái) để cho ôtômat hoạt động được tin cậy.
- Sau đó thực hiện mạch điện bằng trigơ JK (JKFF).

49) Thực hiện ôtômat có nhớ kiểu xung với hai đầu vào tín hiệu là (x_1, x_2) và một đầu ra Z hoạt động theo yêu cầu sau.

- Tín hiệu vào là các xung, xuất hiện ngẫu nhiên liên tục kế tiếp nhau trên hai đầu vào x_1 và x_2. Nhưng các xung vào không được phép xuất hiện đồng thời (nếu kích tổ hợp x_1 x_2 = 11).
- Khi xung trên đầu vào x_1 được xuất hiện thì đầu ra Z = 1 cho tín hiệu báo,

nếu như đã có đúng một xung vào đầu vào x_2 xuất hiện sau xung x_1 xuất hiện sau cùng.

- Đầu ra Z = 0 trong mọi trường hợp khác.
- Khi đầu ra z = 1 thì nó sẽ giữ nguyên giá trị cho tới khi có xung vào tiếp theo trên các đầu vào thì Z trở về 0.
- Dùng ôtômat Mealy mô tả hoạt động.
- Mã hóa trạng thái theo mã Grây.
- Dùng trigơ JK (JKFF) thực hiện lắp ráp.

50) Thiết kế một khóa điện tử mở và đóng, có hai phím bấm xanh (x_1) và đỏ (x_2), với một chốt cửa (đầu ra Z) hoạt động theo nhiệm vụ sau :

- Tín hiệu mở cửa là các xung bấm được tạo ra từ hai phím bấm xanh và phím bấm đỏ. Chúng lần lượt được bấm ngẫu nhiên bất kỳ kế tiếp nhau bằng một tay. Nhưng không được bấm hai tay cùng một lúc (hai xung không được phép cùng đồng thời xuất hiện).
- Khi bấm vào phím bấm xanh (x_1) thì chốt cửa sẽ được mở ra (Z = 1) nếu như trước đó đã có đúng hai lần bấm vào phím đỏ (x_2) sau khi trước đó đã bấm vào phím bấm xanh (x_1) sau cùng.
- Cửa sẽ không được mở (Z = 0) trong các trường hợp khác. Khi cửa được mở (Z =1) thì nó vẫn cứ mở mãi cho tới khi ta bấm đúng vào phím bấm đỏ (x_2) thì cánh cửa mới đóng lại. Tức là khi cửa đã mở nếu bấm vào phím xanh (x_1) thì cửa vẫn cứ mở và có thể báo động.
- Dùng ôtômat Moore để mô tả hoạt động.
- Mã hóa trạng thái theo mã nhị phân.
- Dùng trigơ D (DFF) thực hiện mạch.

51) Tổng hợp một ôtômat kiểu xung có hai đầu vào là (x_1 và x_2) với một đầu ra Z. Hoạt động theo yêu cầu :

- Tín hiệu vào là xung, xuất hiện ngẫu nhiên kế tiếp nhau trên hai đầu vào là x_1 và x_2. Nhưng các xung không được cùng đồng thời xuất hiện (cấm kích x_1 x_2 = 11).
- Khi có xung xuất hiện trên đầu vào x_1 thì đầu ra Z = 1 cho tín hiệu báo, nếu như đã có đúng một xung x_2 xuất hiện sau xung x_1 xuất hiện sau cùng.
- Đầu ra Z = 0 trong mọi trường hợp khác.
- Khi đầu ra Z = 1 thì sẽ giữ nguyên trạng thái đó mãi cho đến khi có đúng hai xung x_2 xuất hiện thì đầu ra Z trở về 0.
- Dùng ôtômat Moore mô tả hoạt động của ôtômat.
- Mã hóa dùng một phân hoạch thế.
- Thực hiện mạch điện dùng trigơ JK (JKFF).

52) Thiết kế một hệ thống số hoạt động bằng các xung có hai đầu vào (x_1, x_2) và một đầu ra Z. Hoạt động theo yêu cầu sau.

- Tín hiệu vào là các xung xuất hiện ngẫu nhiên bất kỳ kế tiếp nhau trên các đầu vào x_1 và x_2. Nhưng chúng không được cùng đồng thời xuất hiện.
- Khi có xung x_1 xuất hiện thì đầu ra Z = 1 nếu như đã có đúng 2 xung x_2 xuất hiện sau xung x_1 xuất hiện sau cùng.
- Đầu ra Z = 0 trong mọi trường hợp khác.
- Khi đầu ra Z = 1 thì nó sẽ giữ nguyên giá trị cho tới khi có đúng 2 xung x_2 xuất hiện liên tiếp thì Z trở về 0.
- Dùng ôtômat Mealy để mô tả hoạt động.
- Mã hóa chống chạy đua y để ôtômat hoạt động tốt.
- Lắp mạch dùng trigơ JK (JKFF).

53) Tổng hợp ôtômat có nhớ kiểu điện thế (mạch không đồng bộ) có hai đầu vào x_1 và x_2, với một đầu ra Z. Đầu ra Z = 1 nếu tổ hợp giá trị các biến vào tiếp theo (coi như một số nhị phân có giá trị) lớn hơn và bằng giá trị hiện tại trước đó. Z = 0 trong mọi trường hợp khác. Cùng với các yêu cầu kỹ thuật sau :

- Thiết kế ôtômat với các yêu cầu kỹ thuật tốt nhất là : chống được chạy đua x, chống được chạy đua y. Dùng một phân hoạch thế để mã hóa cho phép lắp ráp mạch được đẹp và gọn gàng.
- Dùng ôtômat Moore để mô tả chức năng.
- Lắp ráp mạch dùng trigơ JK (JKFF).

54) Tổng hợp ôtômat không đồng bộ có hai đầu vào (x_1, x_2) và một đầu ra Z. Hoạt động theo nhiệm vụ sau :

- Tín hiệu vào đầu vào (x_1, x_2) là dãy số 0 hoặc 1 xuất hiện ngẫu nhiên kế tiếp nhau.
- Đầu ra Z = 1 nếu như tổng các con số 1 nhận được kể từ thời điểm ban đầu (trước đó) là một số lẻ. Còn Z = 0 trong mọi trường hợp khác.
- Dùng ôtômat Mealy để mô tả hoạt động.
- Thực hiện ôtômat đã cho bằng RSFF.

55) Cho ôtômat có bảng chức năng mô tả trên hình B.29. Mã hóa trạng thái ôtômat sao cho không xuất hiện hiện tượng chạy đua tới hạn, đảm bảo không có hazard tĩnh.

S \ x_1x_2	00	01	11	10	Z
A	A	B	A	D	1
B	A	B	B	B	0
C	A	C	B	C	0
D	A	C	A	C	0

Hình B.29. Cho một ôtômat.

56) Tổng hợp mạch không đồng bộ có hai đầu vào (x_1, x_2) và một đầu ra X. Hoạt động theo yêu cầu sau :

- Tín hiệu vào là 0 hoặc 1 xuất hiện ngẫu nhiên kế tiếp nhau trên đầu vào (x_1, x_2), chúng được coi như một số nhị phân.
- Đầu ra Z = 1 nếu như tổng các giá trị của biến vào trước đó với giá trị của biến vào tiếp theo là một số lẻ. Ví dụ : tại thời điểm t_0 có x_1 x_2 = 01 có giá trị là 1, tại thời điểm t_1 ($t_0 < t_1$) ta có x_1x_2 = 10 có giá trị là 2. Vậy tổng của chúng 1 + 2 = 3 là số lẻ nên Z = 1.
- Đầu ra Z = 0 trong các trường hợp khác.
- Dùng ôtômat Mooer mô tả chức năng.
- Thực hiện ôtômat sao cho không xuất hiện chạy đua tín hiệu vào x, và mã hóa trạng thái sao cho không có chạy đua y. Mạch lắp ráp đẹp đẽ gọn gàng.
- Lắp ráp mạch dùng trigơ D (DFF).

57) Thiết kế hệ thống điều khiển đèn giao thông dùng ôtômat có nhớ, cho một ngã tư có lưu lượng xe cân đối. Hệ thống có thể điều khiển bằng tay cũng như hoạt động tự động.

- Mô tả hoạt động dùng ôtômat Moore.
- Mã hóa tùy ý và dùng trigơ JK (JKFF) lắp ráp.

58) Dùng bộ ghi dịch điều khiển sáng tối lần lượt các chữ Đ H B K được dùng làm bảng chỉ thị.

- Xây dựng bảng mã hóa nhị phân cho bộ ghi dịch.
- Thực hiện bộ ghi dịch chỉ thị này bằng trigơ D hoặc bằng trigơ JK (JKFF).

59) Thực hiện bộ đếm nhị phân tăng dần có cơ số đếm Kđ = 16 bằng trigơ T (TFF) hoặc dùng trigơ JK (JKFF).

60) Thực hiện bộ đếm nhị phân giảm dần có cơ số đếm Kđ = 16 dùng trigơ T (TFF) hoặc bằng trigơ JK (JKFF).

61) Tổng hợp bộ đếm có cơ số đếm Kđ = 12 theo phương pháp tổ hợp lần lộn y và $\bar{y}$ bằng trigơ JK (JKFF).

62) Thực hiện bộ phân tần với hệ số chia tần là 10 theo phương pháp tổ hợp lần lộn y và $\bar{y}$ bằng trigơ JK.

63) Thiết kế bộ đếm có hệ số đếm Kđ = 60 dùng để đếm xung giây theo phương pháp tổ hợp lần lộn y và $\bar{y}$, dùng cách nối tiếp hai bộ đếm. Sử dụng trigơ JK lắp mạch.

64) Tổng hợp bộ đếm có cơ số đếm Kđ = 10, trạng thái trong của nó thay đổi theo quy luật nhị phân tăng dần, dùng trigơ JK thực hiện.

65) Thực hiện bộ đếm có cơ số đếm Kđ = 12, trạng thái bên trong của nó biến đổi theo quy luật của nhị phân tăng dần. Bộ đếm được dùng để đếm giờ trong đồng hồ. Dùng trigơ JK lắp mạch điện cụ thể.

66) Thực hiện bộ đếm có cơ số đếm Kđ = 10, có trạng thái bên trong biến đổi theo quy luật nhị phân giảm dần. Dùng trigơ JK lắp ráp.

67) Thực hiện bộ đếm có cơ số đếm Kđ = 10, với trạng thái bên trong biến đổi theo quy luật mã Grây. Sử dụng trigơ JK lắp ráp mạch.

68) Tổng hợp bộ đếm có cơ số đếm Kđ = 10, có trạng thái bên trong biến đổi theo quy luật mã Johnson. Dùng trigơ JK thực hiện.

69) Tổng hợp bộ đếm có cơ số đếm Kđ = 10, mà trạng thái bên trong biến đổi theo quy luật mã bất kỳ tự ta đưa ra. Dùng trigơ JK hoặc trigơ RS thực hiện.

70) Thực hiện bộ đếm thuận nghịch với Kđ = 8, có trạng thái bên trong thay đổi theo quy luật của mã nhị phân. Dùng trigơ JK thực hiện.

71) Thực hiện bộ đếm thuận nghịch với Kđ = 10, với trạng thái bên trong biến đổi theo quy luật của mã nhị phân. Dùng trigơ JK hoặc trigơ RS lắp mạch.

72) Tổng hợp bộ đếm giờ với cơ số đếm Kđ = 12 sử dụng bộ đếm vạn năng điều khiển trạng thái SN74163.

73) Xây dựng bộ đếm xung giây có cơ số đếm Kđ = 60 dùng bộ đếm vạn năng điều khiển trạng thái SN74163.

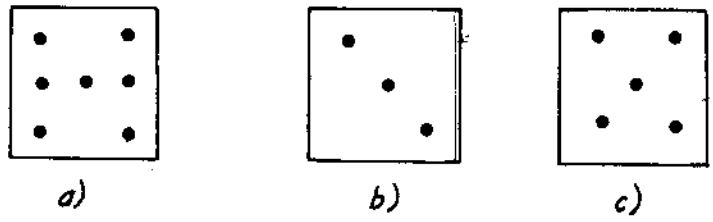
74) Tổng hợp bộ tạo mã Grây ba bit dùng trigơ RS (RSFF).

75) Thực hiện bộ tạo mã Johnson ba bit bằng trigơ JK.

76) Thiết kế bộ tạo mã vòng bốn bit bằng trigơ RS.

77) Thực hiện bộ tạo mã nhị phân thừa 3 dùng trigơ RS hoặc trigơ JK. 78) Tổng hợp bộ tạo mã bất kỳ theo ý ta dùng trigơ RS hoặc trigơ JK.

79) Xây dựng bộ giải mã chỉ thị điểm sáng cho con xúc xắc điện tử mô tả trên hình B.30 sau, cho vào là mã nhị phân (BCD).



Hình B.30. Bộ chỉ thị của xúc xắc điện tử.

80) Dùng bộ đếm có trạng thái trong biến đổi theo mã nhị phân (BCD), kết hợp với bộ giải mã chỉ thị điểm sáng của xúc xắc điện tử, tổng hợp và lắp mạch cho một con xúc xắc điện tử. Khi bấm phím chơi sẽ cho chỉ thị kết quả ngẫu nhiên như ở con xúc xắc.

81) Thiết kế bộ giải mã ma trận điểm sáng có kích thước (3 x 4), với mã đưa vào là nhị phân (BCD), chỉ thị cho các chữ cái A O C T H P Q và D. Vẽ sơ đồ nguyên lý của bộ giải mã.

82) Thực hiện bộ giải mã ma trận điểm sáng có kích thước là (3 x 5), có mã nhị phân đưa vào, chỉ thị cho các chữ số của cơ đếm 10. Xây dựng sơ đồ nguyên lý và lắp mạch thực hiện.

83) Thiết kế bộ giải mã 7 thanh chỉ thị các chữ số của cơ đếm 10, với mã nhị phân đưa vào là mã Grây. Vẽ mạch điện của bộ giải mã.

84) Xây dựng bộ chuyển mã từ của Grây thành mã nhị phân thường (BCD). Vẽ mạch điện nguyên lý.

86) Thiết kế bộ chuyển mã từ mã nhị phân (BCD) thành mã Johnson, có mạch điện thực hiện.

87) Thực hiện bộ chuyển mã từ mã thừa 3 nhị phân thành mã Grây, cùng với mạch điện thực hiện.

88) Thiết kế bộ tạo mã từ mã nhị phân thông thường (BCD) thành mã bất kỳ theo ý ta, kèm theo mạch thực hiện cụ thể.

89) Dùng MUX 4 đầu vào tạo hàm cho hàm Boole sau

$$f_5(X) = (0,1,2,3,4,8,9,11,13,14,18,19,20,21,25,26,29,30,31)$$

$$f_5(X) = (0,1,2,4,5,6,8,9,10,12,14,16,18,22,30)$$

90) Thực hiện hàm Boole 6 biến.

$$f_5(X) = (0,1,2,3,5,7,12,14,16,18,20,22,26,28,30,32$$

$$= 34,37,39,40,43,45,50,51,53,60,61,62,63)$$

a) Dùng mạch MUX 4 đầu vào và mạch lôgic.

b) Bằng mạch MUX 4 đầu vào và mạch MUX 8 đầu vào.

91) Xây dựng ROM thực hiện bộ chuyển mã bốn bít sau :

a) Mã nhị phân (BCD) thành mã thừa 3.

b) Mã nhị phân (BCD) thành mã Grây.

c) Mã nhị phân (BCD) thành mã có trọng số (4-4-2-1).

d) Mã Grây thành mã Johnson.

e) Mã thừa 3 Grây thành mã nhị phân (BCD).

92) Dùng PLA tạo hàm các hàm Boole sáu biến sau.

$$f_1(X) = AB\overline{C}\,\overline{D}EF + AE\overline{F} + ABD + \overline{B}\,\overline{E}$$

$$f_2(X) = \overline{A}\,BE\overline{F} + A\overline{C}\,\overline{D} + \overline{B}D + \overline{E}\,F$$

$$f_3(X) = ACD\overline{E}F + BD\,\overline{E} + \overline{C}\,\overline{D}E + \overline{A}\,F$$

$$f_4(X) = AB\overline{C}D\overline{E} + A\,\overline{D}\,E + \overline{B}\,D + EF + \overline{A}\,F$$

$$f_5(X) = \overline{B}CDE\overline{F} + A\,\overline{C}\,D + \overline{E}\,\overline{A} + B\overline{D}$$

93) Dùng PLA thiết kế bộ chuyển mã 4 bit sau :

a) Mã nhị phân (BCD) thành mã Grây.

b) Mã nhị phân (BCD) thành mã thừa 3.

c) Mã Johnson thành mã Grây.

d) Mã thừa 3 Grây thành mã nhị phân (BCD).

TÀI LIỆU THAM KHẢO

1) *Almaini A. E. A.* Electronic Logic Systems 1986

2) *Bolton M.* Digital systems design with Programmable logic 1990

3) *Bùi Minh Tiêu* – Kỹ thuật số I, I – Nhà xuất bản Đại học 1997

4) *Brown F. M.* Boolean Reasoning the logic of Boolean equations 1990

5) *Brzozowski J. A.* Decompositon of Boolean Fuctions specified by cubes 1997

6) *Kalisz D.* Podstawy elektroniki cyfrowej 1998

7) *Kania D.* Algorytmiczna metoda wyboru ztozonych decompozycji funkeji Boolowskich 1999

8) *Luba T. Nowicka M.* Nowoczesna systeza logiczna PW 1998

9) *Luba T.* Podstawy uklado'w logicznych PW 1998

10) *Majewski W.* Uklady logiczne Wybrane Zagadnienia PW 1998

11) *Nguyễn Nam Quân.* Các phần tử và các khâu của máy tính điện tử ĐHTC – 1976

12) *Nguyễn Nam Quân.* Automatyczne Przeksztalcamie i minimalizacja fumkcji Przelaczajacych za pomoca maszyny cyfrowey 1985.

13) *Nguyễn Thúy Vân* – Kỹ thuật số 1999

14) *Nguyễn Xuân Quỳnh* – Lý thuyết mạch lôgic và kỹ thuật số 1984

15) *Sasao T.* Logic Synthesis and Optimization 1993

16) *Systo M. M.* Algorytmy optymalizacji diskretnej 1993

17) *Tadeusz luba.* Synteza Uklado'w Logicznych PW 2000

18) *Traczyk W.* Uklady cyfrowe Podstawy Teoretyczne i metody syntezy WNT – 1982.

19) *Vũ Đức Thọ (biên dịch)* – Cơ sở kỹ thuật điện tử số – Bộ môn điện tử – Đại học Thanh Hoa Bắc Kinh – NXBGD, 2003.

TOÁN LOGIC VÀ KỸ THUẬT SỐ

Tác giả: Nguyễn Nam Quân

Chịu trách nhiệm xuất bản	:	PGS, TS. TÔ ĐĂNG HẢI
Biên tập	:	NGUYỄN NGỌC KHUÊ, NGUYỄN ĐĂNG
Thiết kế bìa	:	TRẦN THẮNG

NHÀ XUẤT BẢN KHOA HỌC VÀ KỸ THUẬT
70 TRẦN HƯNG ĐẠO – HÀ NỘI

In 800 cuốn, khổ 19 x 27cm tại cơ sở số 2 Nhà in Đại học quốc gia.
Giấy phép xuất bản số: 546–3–12/9/2005
In xong và nộp lưu chiểu tháng 11 năm 2005

205327

Giá: 62.000đ