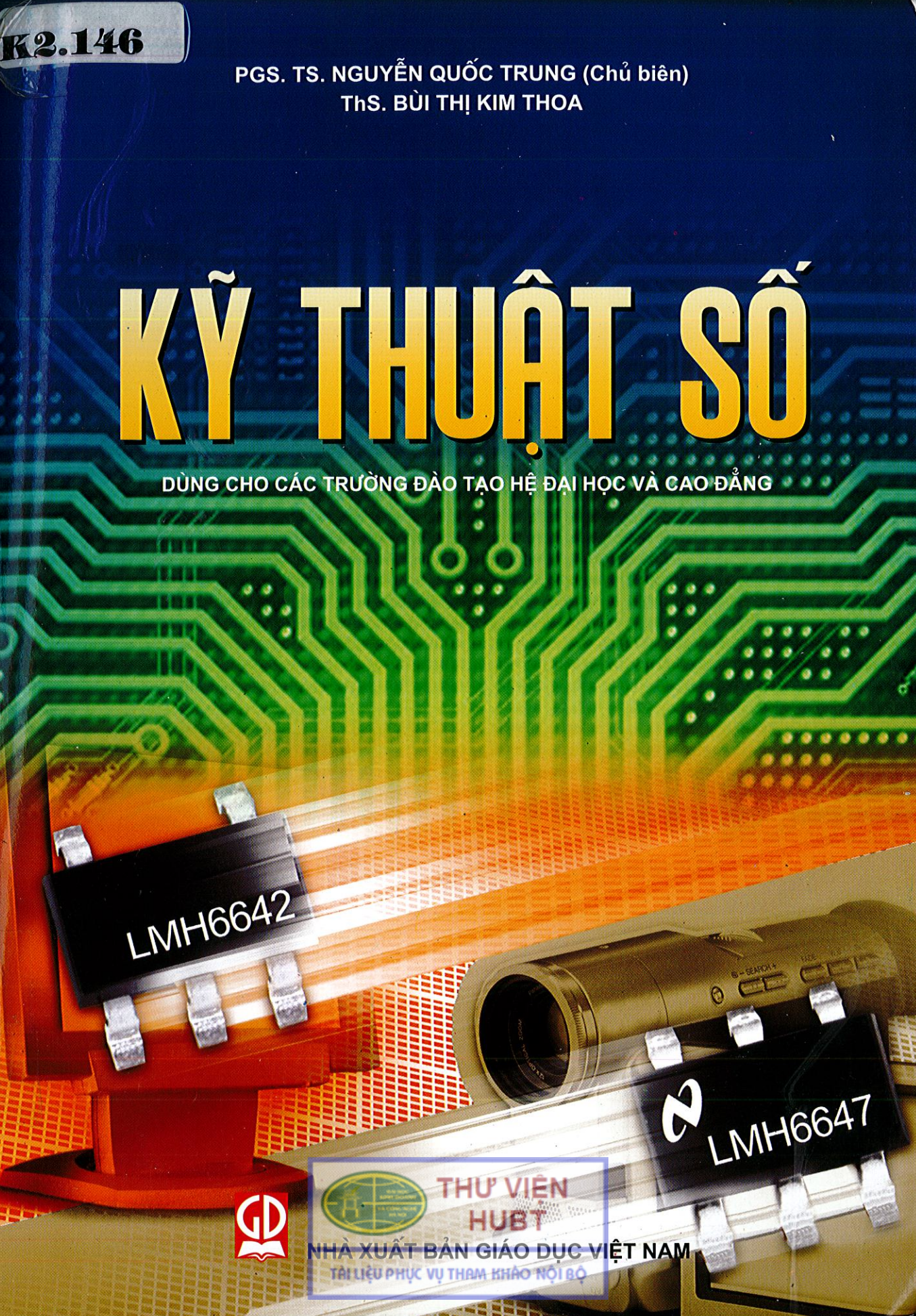
PGS. TS. NGUYỄN QUỐC TRUNG (Chủ biên)
ThS. BÙI THỊ KIM THOA

KỸ THUẬT SỐ

DÙNG CHO CÁC TRƯỜNG ĐÀO TẠO HỆ ĐẠI HỌC VÀ CAO ĐẲNG



NHÀ XUẤT BẢN GIÁO DỤC VIỆT NAM

PGS.TS. NGUYỄN QUỐC TRUNG (Chủ biên)
ThS. BÙI THỊ KIM THOA

KỸ THUẬT SỐ

(Dùng cho các trường đào tạo hệ Đại học và Cao đẳng)

(Tái bản lần thứ nhất)

NHÀ XUẤT BẢN GIÁO DỤC VIỆT NAM

LỜI NÓI ĐẦU

Chúng ta đang sống trong kỷ nguyên mà các sản phẩm công nghệ cao đã thâm nhập vào mọi lĩnh vực đời sống – xã hội. Công nghệ nano – vật liệu mới; công nghệ thông tin số, công nghệ robot thông minh; công nghệ sinh học; công nghệ nghiên cứu về sự sống và nhiều công nghệ tiên tiến khác nữa đã thực sự làm thay đổi thế giới trong những thập kỷ qua.

Kỷ nguyên công nghệ số đã và đang tiếp tục phát triển không ngừng nhằm thông minh hóa và hiện đại hóa hệ thống, nâng cao chất lượng các hệ thống truyền thông và điều khiển. Sự bùng nổ của các sản phẩm công nghệ cao ngày nay phần nhiều dựa trên sự ra đời và phát triển của công nghệ kỹ thuật số và hệ thống thông tin số. Có thể kể đến rất nhiều ứng dụng của kỹ thuật số trong cuộc sống như: hệ thống truyền hình số; thông tin di động số; máy ảnh kỹ thuật số; các thiết bị MP3/MP4; các thiết bị điều khiển số, vv...

Trường Đại học Sư phạm Kỹ thuật Hưng Yên là trường đại học sư phạm kỹ thuật lớn nhất của khu vực phía Bắc. Mục tiêu của nhà trường là trở thành trường đại học định hướng thực hành/ứng dụng hàng đầu trong khu vực với nhiệm vụ đào tạo kỹ sư công nghệ cho các công ty, doanh nghiệp và cử nhân sư phạm kỹ thuật cho các trường chuyên nghiệp và dạy nghề góp phần vào công cuộc công nghiệp hoá, hiện đại hoá đất nước. Chính vì thế, ngoài việc tập trung đào tạo kiến thức lý thuyết chuyên môn, sinh viên cần phải đạt được các năng lực nghề nghiệp cần thiết. Các năng lực này được hình thành thông qua quá trình học tập lý thuyết, thực hành/thí nghiệm, thông qua các đồ án môn học, đồ án tốt nghiệp và thực tập tại các công ty, doanh nghiệp.

Với sự giúp đỡ của Dự án giáo dục đại học Việt Nam – Hà Lan, cộng với kinh nghiệm nhiều năm nghiên cứu và giảng dạy tại Trường đại học Sư phạm Kỹ thuật Hưng Yên, tập thể các giảng viên Khoa Điện – Điện tử đã cho ra đời bộ sách Kỹ thuật Điện – Điện tử. Cuốn Kỹ thuật số là một trong những cuốn sách thuộc bộ sách này.

Mục tiêu của cuốn sách nhằm cung cấp các kiến thức lý thuyết tổng quan và đầy đủ nhất về phương pháp phân tích và tổng hợp mạch số cho sinh viên, giảng viên và các kỹ sư, kỹ thuật viên chuyên ngành Điện – Điện tử. Bên cạnh đó, các bài tập luyện tập và các ví dụ ứng dụng thực tiễn, đã được tác giả kiểm nghiệm thông qua các bài thực hành/thí nghiệm, sẽ được bổ sung vào trong nội dung của cuốn sách. Với cấu trúc như vậy, người đọc có thể dễ dàng đọc và tự học. Kinh nghiệm cho thấy, người đọc sẽ nhớ rất tốt và hứng thú với những gì mà họ được đọc và thực hành, hơn là họ được nghe.

Cuốn sách cũng là tài liệu tham khảo rất có ích cho tất cả các sinh viên đang theo học các ngành Kỹ thuật điện tử, Viễn thông, Kiến trúc máy tính, Tin học, Tự động hoá, Đo lường, Điều khiển,.... ở các trường Cao đẳng và Đại học.

Nội dung cuốn sách gồm có 6 chương:

Chương 1: Trình bày các khái niệm cơ bản của hệ thống số, các hệ thống số đếm, các phép tính số học và các loại mã.

Chương 2: Trình bày cơ sở toán logic, các phương pháp biểu diễn và tối thiểu hoá hàm logic.

Chương 3: Giới thiệu về các thông số, cấu tạo và hoạt động của các vi mạch logic.

Chương 4: Trình bày các phương pháp phân tích và thiết kế mạch tổ hợp, các mạch logic tổ hợp thường gặp, các ứng dụng.

Chương 5: Trình bày các phương pháp phân tích và thiết kế mạch dãy, các mạch logic dãy thường gặp, các ứng dụng.

Chương 6: Trình bày các bộ chuyển đổi số – tương tự và tương tự – số.

Trong mỗi chương, sau mỗi phần lý thuyết đều có các ví dụ cụ thể và các ứng dụng được giải thích rõ ràng giúp người đọc hiểu rõ vấn đề, các hệ thống bài tập cuối mỗi chương giúp người học nắm vững các kiến thức đã học.

Các tác giả xin chân thành cảm ơn các đồng nghiệp trong Bộ môn Kỹ thuật Điện tử của trường Đại học Sư phạm Kỹ thuật Hưng Yên đã đóng góp nhiều ý kiến bổ ích cho việc soạn thảo.

Nội dung cuốn sách đã được dùng làm tài liệu giảng dạy cho các lớp Đại học và Cao đẳng trong nhiều năm, tuy nhiên không thể tránh khỏi những thiếu sót. Chúng tôi rất mong nhận được ý kiến đóng góp của các bạn đồng nghiệp, bạn đọc để kịp thời chỉnh sửa, bổ sung cho lần tái bản sau.

Mọi ý kiến đóng góp xin gửi về: Khoa Điện – Điện tử, trường Đại học Sư phạm Kỹ thuật Hưng Yên, huyện Khoái Châu, Hưng Yên hoặc Công ty Cổ phần sách Đại học và Dạy nghề – Nhà xuất bản Giáo dục Việt Nam, 25 Hàn Thuyên, Hà Nội.

Các tác giả

Chương 1
KHÁI NIỆM CƠ BẢN CỦA HỆ THỐNG SỐ

1.1. KHÁI NIỆM TÍN HIỆU SỐ

Về cơ bản có hai cách biểu diễn giá trị của đại lượng, đó là tương tự (analog) và số (digital).

– Biểu diễn dạng tương tự: trong cách biểu diễn dạng tương tự, một đại lượng được biểu diễn bằng hiệu điện thế, cường độ dòng điện, hay số đo chuyển động tương quan với giá trị của đại lượng đó.

Ví dụ, đồng hồ đo vận tốc trong xe ôtô, kim đo phải lệch tương ứng với tốc độ hiện tại của xe và độ lệch này phải thay đổi tức thì khi vận tốc xe tăng hay giảm.

Một ví dụ khác về đại lượng tương tự là chiếc micrô. Trong thiết bị này, biên độ hiệu điện thế đầu ra luôn tỷ lệ với cường độ sóng âm tác động vào màng rung của micrô ở đầu vào.

Các đại lượng tương tự có một đặc điểm rất quan trọng đó là: Đại lượng tương tự có thể thay đổi theo một khoảng giá trị liên tục.

– Biểu diễn dạng số: Trong cách biểu diễn dạng số, đại lượng được biểu diễn bằng các biểu tượng gọi là ký số (digit).

Ví dụ, đồng hồ hiện số, hiển thị thời gian trong ngày như giờ, phút, giây dưới dạng số thập phân. Tuy thời gian trong ngày thay đổi liên tục, nhưng số hiện của đồng hồ số lại thay đổi từng bước, mỗi bước là một phút hay một giây.

Nói cách khác, các đại lượng số có đặc điểm là giá trị của nó thay đổi theo từng bước rời rạc.

Vì tính rời rạc trong biểu diễn dạng số nên khi đọc giá trị của đại lượng số, không hề có sự mơ hồ.

a) Ưu điểm của kỹ thuật số so với kỹ thuật tương tự

– Do sử dụng chuyển mạch nên nhìn chung thiết bị số dễ thiết kế hơn.

– Thông tin được lưu trữ dễ dàng.

– Tính chính xác và độ tin cậy cao hơn.

– Có thể lập trình để điều khiển hệ thống số.

– Ít ảnh hưởng bởi nhiễu.

– Nhiều mạch số có thể được tích hợp trên một chíp IC.

b) Giới hạn của kỹ thuật số

Mặc dù hệ thống số có rất nhiều ưu điểm, nhưng bên cạnh đó vẫn có một số hạn

chế. Do hầu hết các đại lượng vật lý đều có bản chất là tương tự, nên muốn tận dụng được hệ thống kỹ thuật số thì chúng ta phải thực hiện các bước sau:

– Biến đổi đầu vào dạng tương tự thành dạng số (A/D).

– Xử lý tín hiệu số.

– Biến đổi đầu ra dạng số thành dạng tương tự (D/A).

Tuy nhiên, quá trình trên được coi là quá trình tất yếu đối với hệ thống số.

Ở một số hệ thống, để tận dụng cả ưu điểm của kỹ thuật số và kỹ thuật tương tự người ta dùng cả hai hệ thống. Trong các hệ thống lai ghép này thì việc quan trọng là phải xác định được phần nào của hệ thống nên sử dụng kỹ thuật số và phần nào nên sử dụng kỹ thuật tương tự.

1.2. TRẠNG THÁI NHỊ PHÂN VÀ MỨC LOGIC

Trong hệ thống kỹ thuật số, thông tin được xử lý đều biểu diễn dưới dạng nhị phân. Bất kỳ thiết bị nào chỉ có hai trạng thái hoạt động đều có thể biểu diễn được các đại lượng dưới dạng nhị phân.

Ví dụ, một công tắc chỉ có hai trạng thái hoạt động là đóng hoặc mở. Ta có thể quy ước công tắc mở biểu diễn nhị phân 0 và công tắc đóng biểu diễn nhị phân 1. Với quy ước này ta có thể biểu diễn số nhị phân bất kỳ.

Có vô số thiết bị chỉ có hai trạng thái hoạt động hay vận hành ở hai điều kiện đối lập nhau như: bóng đèn (sáng/tối), điôt (dẫn/không dẫn), rơle (ngắt/đóng),...

Trong thiết bị điện tử số, thông tin nhị phân được biểu diễn bằng hiệu điện thế (hay dòng điện) tại đầu vào hay đầu ra của mạch. Thông thường, số nhị phân 0 và 1 được biểu diễn bằng hai mức điện thế danh định. Ví dụ: 0V có thể biểu diễn bằng nhị phân 0 và +5V biểu diễn bằng nhị phân 1. Trên thực tế, các số 0 hoặc 1 được biểu diễn bằng một khoảng điện thế quy định nào đó. Ví dụ, khoảng điện thế từ 0V đến 0,8V được quy định là mức logic 0 và từ 3V đến 5V được quy định là mức logic 1.

Đối với hệ thống kỹ thuật số giá trị chính xác của hiệu điện thế hay dòng điện là không quan trọng, chỉ cần nó nằm trong khoảng quy định mức logic 0 hay 1.

1.3. KHÁI NIỆM BIT, BYTE, WORD

– Bit (binary digit – số nhị phân): Là một trong hai số 0 và 1 dùng trong các thiết bị số để biểu thị các số, các ký tự và các lệnh máy. Bit là đơn vị nhỏ nhất của thông tin.

– Byte: Hầu hết máy tính đều thao tác và lưu trữ thông tin, dữ liệu nhị phân theo từng nhóm 8 bit, chính vì vậy chuỗi 8 bit này có tên là byte.

– Word – từ: Thông tin dữ liệu được tạo thành từ một đơn vị cơ bản gọi là từ (word). Tuỳ theo từng loại máy, 1 từ có thể là 8 bit, 16 bit, 32 bit,...Các thiết bị chỉ truyền đi hay nhận vào nguyên 1 từ hay nhiều từ chứ không phải chỉ vài bit của từ. Tuy nhiên đơn vị nhớ cơ bản là bit.

1.4. CÁC HỆ THỐNG SỐ ĐẾM

1.4.1. Các hệ thống số đếm sử dụng trong kỹ thuật số

Để biểu diễn các số đo, các đại lượng vật lý ta cần các hệ thống số đếm. Trong một hệ thống số đếm bất kỳ, một con số được biểu diễn dưới dạng một dãy các chữ số liên tiếp. Như vậy, ứng với mỗi tập hợp các chữ số dùng để biểu diễn các con số chúng ta sẽ được một hệ thống đếm khác nhau. Người ta gọi cơ số của hệ đếm là số chữ số khác nhau dùng để biểu diễn các con số trong hệ đếm đó.

Trong kỹ thuật số, có bốn hệ thống số đếm quan trọng là:

– Hệ đếm thập phân (Decimal): còn được gọi là hệ cơ số 10, nó sử dụng 10 chữ số để biểu diễn các con số, 10 chữ số đó là: 0, 1, 2,..., 9.

– Hệ nhị phân (Binary): còn được gọi là hệ cơ số 2, nó sử dụng hai chữ số để biểu diễn tất cả các con số, hai chữ số đó là 0 và 1.

– Hệ bát phân (Octal): còn được gọi là hệ cơ số 8, nó sử dụng 8 chữ số để biểu diễn tất cả các con số, 8 chữ số đó là: 0, 1, 2,..., 7.

– Hệ thập lục phân (Hexa): còn được gọi là hệ cơ số 16, nó sử dụng 16 ký hiệu để biểu diễn tất cả các con số, 16 ký hiệu đó là: 0, 1, 2, .. , 9, A, B, C, D, E, F.

Các hệ thống số đếm chúng ta nói ở trên là các hệ thống đếm theo vị trí, tức là giá trị của các chữ số phụ thuộc vào vị trí của nó trong con số. Vì thế trong các con số, chữ số đầu tiên được gọi là chữ số (bit) có ý nghĩa nhất (MSD – Most Significant Digit), tức có trọng số lớn nhất và chữ số cuối cùng là chữ số (bit) ít ý nghĩa nhất (LSD – Least Significant Digit), tức có trọng số bé nhất.

1.4.2. Chuyển đổi giữa các hệ thống số đếm

a) Chuyển đổi từ các hệ thống số đếm khác sang hệ thập phân

Nếu có số A trong hệ thống đếm B thì ta có thể chuyển đổi sang hệ thập phân theo công thức sau:

$$(A)_B = a_{n-1}B^{n-1} + a_{n-2}B^{n-2} + \ldots + a_0B^0 + a_{-1}B^{-1} + \ldots + a_{-m}B^{-m} \qquad (1.4.2.1)$$

Trong đó:

A là một số ($A = a_{n-1}\, a_{n-2} \ldots a_0, a_{-1}\, a_{-2}.\, a_{-m}$)

B là cơ số của hệ đếm; $0 \le a_k \le B - 1$

n là số chữ số trong phần nguyên

m là số chữ số trong phần thập phân

a_{n-1} là chữ số có ý nghĩa nhất

a_{-m} là chữ số ít ý nghĩa nhất

B^k là trọng số của chữ số ở vị trí k; với $k = -m \div n - 1$.

Ví dụ 1.4.2.1:

$(1101,01)_2 = 1.2^3 + 1.2^2 + 0.2^1 + 1.2^0 + 0.2^{-1} + 1.2^{-2} = (13,25)_{10}$

$(12,4)_8 = 1.8^1 + 2.8^0 + 4.8^{-1}$

$(13,8)_{16} = 1.16^1 + 3.16^0 + 8.16^{-1}$

b) Chuyển đổi từ hệ thập phân sang các hệ thống số đếm khác

Với phần nguyên, ta thực hiện chia liên tiếp số thập phân cho cơ số của hệ đếm cho đến khi thương bằng 0 và thực hiện lấy số dư theo thứ tự số dư cuối cùng là chữ số có ý nghĩa nhất và số dư đầu tiên là chữ số ít ý nghĩa nhất.

Với phần lẻ sau dấu phẩy, sự chuyển đổi được thực hiện bằng cách nhân liên tiếp cơ số của hệ đếm và giữ lại phần nguyên được sinh ra từ trái qua phải.

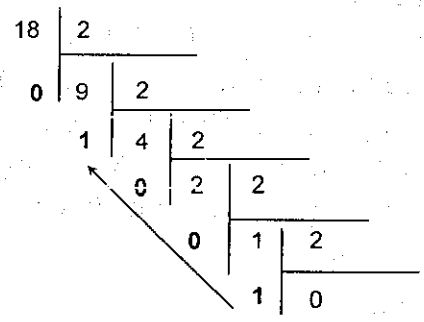
Ví dụ 1.4.2.2. Chuyển $(18,25)_{10}$ sang hệ nhị phân.

Giải:

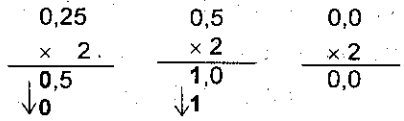
Phần nguyên thực hiện chia liên tiếp cho 2 cho đến khi thương bằng 0 như trên hình 1.4.2.1:

Vậy $(18)_{10} = (10010)_2$

Với phần lẻ thực hiện nhân liên tiếp với 2 như trên hình 1.4.2.2:



Hình 1.4.2.1



Hình 1.4.2.2

Vậy: $(0,25)_{10} = (0,01)_2$

Ta có: $(18,25)_{10} = (10010,01)_2$

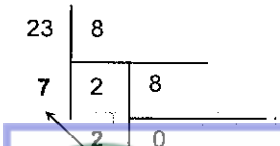
(Kiểm tra lại kết quả bằng cách chuyển từ hệ nhị phân sang hệ thập phân như đã học ở mục trước).

Lưu ý, sự chuyển đổi không phải luôn luôn chính xác, nói chung một lượng gần tương đương có thể được xác định bằng sự kết thúc quá trình nhân tại điểm mong muốn.

Ví dụ 1.4.2.3. Chuyển đổi $(23,15)_{10}$ sang hệ bát phân.

Giải:

Phần nguyên:



Vậy: $(23)_{10} = (27)_8$

Phần lẻ:

0,15	0,2	0,6	0,8
× 8	× 8	× 8	...
↓1,2	↓1,6	↓4,8	...
1	1	4	

Vậy: $(0,15)_{10} \approx (0,114)_8$

Ta có: $(23,15)_{10} \approx (27,114)_8$

(Kiểm tra lại kết quả bằng cách chuyển từ hệ bát phân sang hệ thập phân như đã học ở mục trước).

Tương tự, lấy ví dụ chuyển từ hệ thập phân sang thập lục phân.

c) Chuyển đổi từ hệ nhị phân sang hệ bát phân và ngược lại

Với 3 bit nhị phân có thể tạo ra được $(2^3 = 8)$ 8 tổ hợp số nhị phân 3 bit khác nhau. Như vậy, mỗi ký số bát phân có thể được biểu diễn bằng nhóm mã nhị phân ba bit khác nhau. Khi nhập dữ liệu vào máy tính thì ba bit nhị phân có thể được biểu diễn bằng một ký số bát phân là rất thuận tiện. Trước khi dữ liệu được xử lý thì nó được tái tạo thành dạng nhị phân bằng các mạch chuyển đổi.

Để chuyển từ hệ nhị phân sang hệ bát phân ta thực hiện nhóm số nhị phân thành từng nhóm ba bit và chuyển sang ký số bát phân tương ứng.

Đối với phần nguyên thực hiện nhóm từ phải sang trái, đối với phần lẻ thực hiện nhóm từ trái sang phải. Nếu nhóm cuối cùng không đủ 3 bit thì thêm bit 0 vào.

Ngược lại, chuyển từ bát phân sang nhị phân đổi từng ký số bát phân thành từng nhóm nhị phân 3 bit.

Bảng chuyển đổi:

Số hệ 8	0	1	2	3	4	5	6	7
Số hệ 2	000	001	010	011	100	101	110	111

Từ bảng chuyển đổi trên ta có thể đổi bất kỳ số hệ 2 nào sang hệ 8 hoặc ngược lại.

Ví dụ 1.4.2.4:

$(001\ 011\ 001\ 010\ 101,101\ 010\ 100)_2 = (13125,524)_8$

$(713,26)_8 = (111\ 001\ 011,010\ 110)_2$

d) Chuyển từ hệ nhị phân sang hệ thập lục phân và ngược lại

Tương tự như mục (c) ở đây ta nhóm từng nhóm 4 bit. Bảng chuyển đổi:

Số Hexa	0	1	2	3	4	5	6	7
Số nhị phân	0000	0001	0010	0011	0100	0101	0110	0111
Số Hexa	8	9	A	B	C	D	E	F
Số nhị phân	1000	1001	1010	1011	1100	1101	1110	1111

Có bốn bit nhị phân có thể tạo được (2^4) 16 tổ hợp số nhị phân 4 bit khác nhau.

Mỗi tổ hợp của bốn bit nhị phân có thể biểu diễn bằng một ký số thập lục phân. Như vậy, khi nhập dữ liệu vào máy tính thì bốn bit nhị phân được biểu diễn dưới dạng các ký số Hexa rất thuận tiện. Số Hexa được biến đổi thành dạng nhị phân trước khi chúng được xử lý bởi mạch số.

Ví dụ 1.4.2.5:

$(0101\ 0010\ 0111\ 1011\ 1001,1001\ 1011)_2 = (527B9,9B)_{16}$

$(5AC,9E)_{16} = (10110101100,1001111)_2$

e) Chuyển từ hệ bát phân sang hệ thập lục phân và ngược lại

Do chuyển đổi qua lại giữa hệ 2 và hệ 8, giữa hệ 2 và hệ 16 rất nhanh chóng nên khi chuyển từ hệ 8 sang hệ 16 hoặc ngược lại ta dùng hệ 2 làm trung gian.

Ví dụ 1.4.2.6:

$(723)_8 = (111010011)_2 = (1D3)_{16}$

$(C4)_{16} = (11000100)_2 = (304)_8$

1.4.3. Phép đếm trong các hệ thống số đếm

Cũng tương tự như hệ 10, đối với hệ 8 hoặc hệ 16 do nó gồm 8 hoặc 16 ký tự nên khi đếm đến 7 hoặc F nó quay trở về 0 và ký số trước nó tăng thêm 1 đơn vị.

Ví dụ 1.4.3.1:

Hệ 8: 0, 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 16, 17, 20, …

Hệ 16: 0,1, 2,…, E, F,10,11,…,1E,1F, 20,….

Quy tắc đếm trong hệ 2: Sau mỗi lần đếm bit có trọng số là 1 (2^0) sẽ đảo bit (1 sang 0 hoặc 0 sang1), còn đối với các bit khác sẽ đảo bit khi bit ngay sau nó chuyển từ 1 sang 0.

Ví dụ: 0,1,10,11,100,101,110,111,1000,….

Số tiếp theo của số nhị phân gồm n bit 1 là số gồm (n + 1) bit với bit 1 đầu tiên và n bit 0 tiếp theo.

1.5. CÁC PHÉP TÍNH SỐ HỌC TRONG HỆ NHỊ PHÂN

1.5.1. Cộng nhị phân

Phép cộng hai số nhị phân được tiến hành giống như cộng số thập phân. Tuy nhiên, chỉ có bốn trường hợp có thể xảy ra trong phép cộng 2 bit nhị phân tại vị trí bất kỳ đó là:

1) 0 + 0 = 0
2) 1 + 0 = 1
3) 1 + 1 = 10 (bằng 0 nhớ 1)
4) 1 + 1 + 1 = 11 (bằng 1 nhớ 1)

Trường hợp cuối cùng xảy ra khi hai bit ở vị trí nào đó đều là 1 và có nhớ từ một vị trí trước đó.

Phép cộng là phép tính quan trọng nhất, sau này ta sẽ thấy các phép trừ, nhân, chia đều được thực hiện thông qua phép cộng.

Ví dụ 1.5.1.1:

$$\begin{array}{r} 1011\ (11)_{10} \\ +1000\ (\ 8)_{10} \\ \hline 10011\ (19)_{10} \end{array} \qquad \begin{array}{r} 11{,}011\ (3{,}375)_{10} \\ +\ 10{,}110\ (2{,}75)_{10} \\ \hline 110{,}001\ (6{,}125)_{10} \end{array}$$

Không cần xét phép cộng hơn hai số nhị phân cùng một lúc, bởi vì ở tất cả các hệ thống kỹ thuật số, hệ mạch thực sự thực hiện phép cộng chỉ có thể cộng mỗi lần hai số. Khi phải cộng hơn hai số, nó sẽ cộng hai số đầu tiên trước, rồi cộng tiếp kết quả với số thứ ba, và cứ thế. Đó không phải là một khuyết điểm nghiêm trọng, vì máy tính hiện đại có khả năng thực hiện một phép cộng trong vài ns.

Phép cộng là phép tính số học quan trọng nhất trong hệ thống kỹ thuật số. Ta sẽ thấy, các phép trừ, nhân, chia được thực hiện ở hầu hết máy vi tính và máy tính bấm hiện đại nhất thực ra chỉ dùng phép cộng làm phép toán cơ bản của chúng.

1.5.2. Biểu diễn các số có dấu

Do đa số máy tính xử lý cả số âm lẫn số dương, nên cần có dấu hiệu nào đó để biểu thị dấu của số (+ hay −). Thường thì người ta thêm vào một bit gọi là bit dấu, thông thường chấp nhận bit 0 là bit dấu biểu thị số dương và bit 1 là bit dấu biểu thị số âm. Bit dấu này được thêm vào ở vị trí ngoài cùng bên trái. Hệ thống phổ biến nhất để biểu diễn số nhị phân có dấu là hệ bù hai. Trước khi xem xét điều này được thực hiện ra sao, ta phải tìm hiểu cách thành lập số bù 1 và số bù 2 của một số nhị phân:

– Số bù 1: Để có số bù 1 của một số nhị phân, ta thay mỗi bit 0 thành bit 1 và bit 1 thành bit 0. Nói cách khác là đảo tất cả các bit của số đó.

Ví dụ 1.5.2.1: Số nhị phân ban đầu: 1001011

Đảo mỗi bit để thành lập dạng bù 1: 0110100

Sở dĩ người ta gọi là số bù 1 vì tổng 2 bit có trọng số tương ứng trong hai số nói trên luôn bằng 1.

– Số bù 2: Bù hai của một số nhị phân được hình thành bằng cách lấy bù 1 của số đó và cộng thêm 1 đơn vị.

Ví dụ 1.5.2.2:

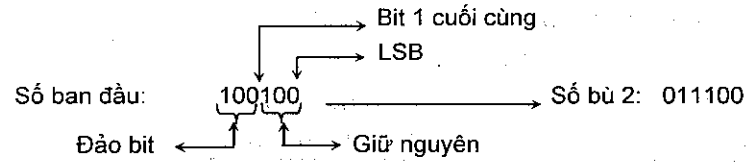
Số nhị phân ban đầu:	1001101
Số bù 1:	0110010
Cộng 1 để hình thành dạng bù hai:	+ 1
Số bù hai của số nhị phân ban đầu:	0110011

Bù 2 của số bù 2 chính là số nhị phân ban đầu.

**** Quy tắc tìm số bù 2:***

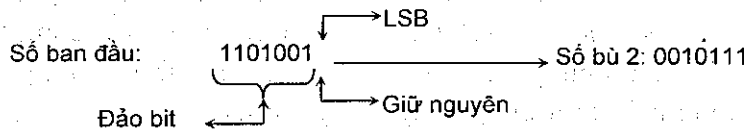
– Nếu bit ít ý nghĩa nhất (LSB) là 0 thì giữ nguyên các bit từ LSB đến bit 1 cuối cùng, các bit còn lại thực hiện đảo bit.

Ví dụ 1.5.2.3:



– Nếu LSB là 1 thì giữ nguyên LSB, các bit còn lại thực hiện đảo bit.

Ví dụ 1.5.2.4:



**** Biểu diễn số có dấu trong hệ bù 2:***

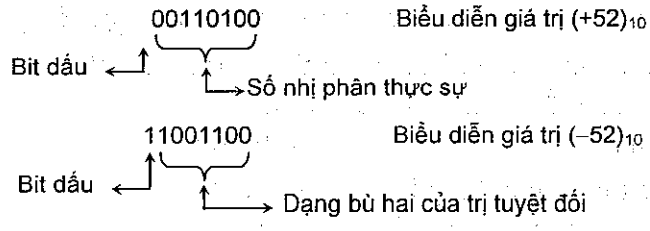
Hệ bù hai biểu diễn những số có dấu theo cách sau đây:

+ Nếu là số dương, trị tuyệt đối được biểu diễn theo dạng nhị phân thực sự của nó và bit dấu là 0 được đặt vào trước MSB (bit có ý nghĩa nhất).

+ Nếu là số âm, trị tuyệt đối được biểu diễn ở dạng bù 2 và bit dấu là 1 được đặt trước MSB.

Ví dụ 1.5.2.5. Biểu diễn $(+52)_{10}$ và $(52)_{10}$ bằng 8 bit trong hệ bù 2.

Giải: Ta có bit tận cùng bên trái là bit dấu, bảy bit còn lại biểu diễn cho trị tuyệt đối như sau:



Sở dĩ hệ bù 2 được dùng để biểu diễn những số có dấu bởi vì như ta sẽ thấy, nó cho phép thực hiện phép trừ nhưng thực ra là phép cộng.

Khi chuyển sang dạng bù hai ta thực hiện đối với cả bit dấu, thì số bù hai của một số biểu diễn số âm của số đó.

Ví dụ 1.5.2.6.

$(+52)_{10} = 00110100 \xrightarrow{\text{Bù 2}} 11001100 = (-52)_{10}$

$(-52)_{10} = 11001100 \xrightarrow{\text{Bù 2}} 00110100 = (+52)_{10}$

**** Giá trị thập phân của số nhị phân trong hệ bù 2:***

Nếu ta có $b_{n-1}b_{n-2}...b_0$ là số nhị phân có dấu được biểu diễn trong hệ bù 2 thì ta có thể chuyển đổi sang hệ thập phân theo công thức sau:

$$b_{n-1}b_{n-2}...b_0 = -b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + ... + b_0 2^0$$

Trong đó: b_{n-1} là bit dấu.

Ví dụ 1.5.2.7. Tìm giá trị thập phân tương ứng của các số nhị phân có dấu được biểu diễn trong hệ bù 2 như sau:

a) 01100101

b) 10110101

Giải:

a) $01100101 = -0.2^7 + 1.2^6 + 1.2^5 + 0.2^4 + 0.2^3 + 1.2^2 + 0.2^1 + 1.2^0 = (101)_{10}$

b) $10110101 = -1.2^7 + 0.2^6 + 1.2^5 + 1.2^4 + 0.2^3 + 1.2^2 + 0.2^1 + 1.2^0 = (-75)_{10}$

**** Trường hợp đặc biệt ở dạng biểu diễn bù 2:***

Số nhị phân có (n + 1) bit, trong đó bit dấu là 1 và n bit trong trị tuyệt đối đều là bit 0 thì số thập phân tương đương là -2^n.

Ví dụ 1.5.2.8.

$$1000 = (-2^3)_{10} = (-8)_{10}; \qquad (100000) = (-2^5)_{10} = (-32)_{10}$$

Do đó, ta có thể phát biểu rằng toàn bộ khoảng giá trị mà (n + 1) bit biểu diễn được ở hệ bù 2 có dấu là từ -2^n đến $-(2^n - 1)$. Tổng cộng có 2^{n+1} giá trị khác nhau kể cả số 0.

1.5.3. Cộng trong hệ bù 2

Ở đây bit dấu của mỗi số được thao tác theo cùng cách thức với các bit trị tuyệt đối. Đối với hệ bù 2 thì yêu cầu số bị cộng và số cộng phải có cùng số bit.

Thực hiện cộng các bit có trọng số tương ứng với nhau, nếu kết quả có số nhớ thì cộng vào bit có trọng số cao hơn kế tiếp đó.

Khi thực hiện phép cộng, số bit quy định cho số bị cộng, số cộng và kết quả là như nhau. Vị trí của bit dấu và các bit trị tuyệt đối là tương ứng, nếu kết quả có bit nhớ cuối cùng là 1 được sinh ra thì bit này được bỏ đi và kết quả là những bit còn lại.

Ví dụ 1.5.3.1. Với số bit quy định cho trị tuyệt đối là 7, thực hiện cộng $(+9)_{10}$ với $(-4)_{10}$ trong hệ bù 2.

Giải:

```
  0 0001001   (+9)10
+ 1 1111100   (-4)10
-------------------
1 0 0000101   (+5)10
```

Bit dấu

Bit nhớ này được bỏ qua, kết quả là 0 0000101= $(+5)_{10}$

Sự tràn số: Khi số bit quy định cho biểu diễn trị tuyệt đối của kết quả không đủ thì sẽ gây ra sự tràn số vào vị trí bit dấu làm cho kết quả bị sai.

Ví dụ 1.5.3.2. Quy định số bit biểu diễn trị tuyệt đối là 7, thực hiện phép cộng $(+120)_{10}$ với $(+23)_{10}$ trong hệ bù 2.

Giải:

$$\begin{array}{r} 0\ 1111000 \quad (+120)_{10} \\ +0\ 0010111 \quad (+\ 23)_{10} \\ \hline 1\ 0001111 \end{array}$$

Bit dấu sai ← → Trị tuyệt đối sai

Hiện tượng tràn số này chỉ xảy ra khi cộng hai số dương hoặc hai số âm. Muốn phát hiện hiện tượng tràn số, kiểm tra bit dấu của số bị cộng và số cộng. Nếu số bị cộng và số cộng có bit dấu khác nhau thì không cần kiểm tra tiếp, nếu số bị cộng và số cộng cùng dấu thì phải kiểm tra bit dấu của kết quả. Nếu kiểm tra thấy kết quả có bit dấu ngược với số bị cộng và số cộng thì có nghĩa là đã xảy ra hiện tượng tràn số, phải dành số bit nhiều hơn cho phép tính. Máy vi tính dùng một mạch đặc biệt để phát hiện mọi trường hợp tràn số và báo kết quả sai.

Trong trường hợp muốn tăng số bit để không có số nhớ hoặc không xảy ra hiện tượng tràn số thì chỉ cần lặp lại bit có ý nghĩa nhất (bit dấu).

Ví dụ 1.5.3.3. Ta có số nhị phân có dấu trong hệ bù 2 được biểu diễn bằng 4 bit là 1001 và 0110. Khi ta muốn biểu diễn nó bằng 8 bit chỉ cần lặp lại 4 lần bit đầu tiên thì giá trị của các số là không thay đổi.

$1001 = 11111001 \qquad (-2^3 + 2^0 = -2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^0)$

$0110 = 00000110$

1.5.4. Trừ trong hệ bù hai

Phép trừ dùng trong hệ bù 2 thật ra liên quan đến phép cộng và không khác với trường hợp áp dụng cho phép cộng đã xét ở mục trước.

Phép trừ được thực hiện bằng cách cộng số bị trừ với số bù hai của số trừ.

Ví dụ 1.5.4.1. Với số bit quy định cho trị tuyệt đối là 7, thực hiện phép trừ $(+19)_{10}$ cho $(+14)_{10}$ trong hệ bù 2.

Giải:

Ta thực hiện lấy bù 2 của 00001110 để có 11110010 rồi thực hiện phép cộng:

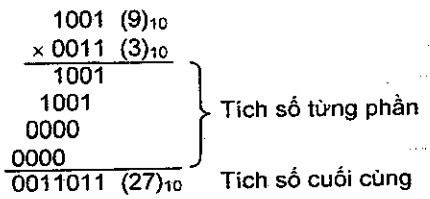
$$\begin{array}{rl} 0\ 0010011 & (+19)_{10} \\ +\ 1\ 1110010 & (-14)_{10} \\ \hline 10\ 0000101 & (+\ 5)_{10} \end{array}$$

Kết quả là $(0\ 0000101)_2 = (+5)_{10}$

1.5.5. Nhân nhị phân

Được thực hiện hệt như nhân số thập phân. Quá trình này thật ra còn đơn giản hơn, do ký số của số nhân chỉ là 0 hoặc 1, vì vậy ta luôn chỉ nhân cho 0 hay 1.

Ví dụ sau minh hoạ nhân hai số nhị phân không dấu.



Trong ví dụ này, số bị nhân và số nhân ở dạng nhị phân đích thực và không sử dụng bit dấu.

Phép nhân trong hệ bù hai:

Trong những máy tính có sử dụng dạng biểu diễn bù 2, phép nhân được thực hiện theo cách thức vừa mô tả trên, với điều kiện cả số nhân lẫn số bị nhân đều ở dạng nhị phân thực sự.

Đầu tiên xác định bit dấu của hai số nhân để xác định bit dấu của kết quả theo quy tắc:

– Nếu bit dấu là như nhau thì tích là số dương.

– Nếu bit dấu là khác nhau thì tích là số âm.

Các bước tiếp theo có các khả năng như sau:

+ Nếu hai số cần nhân đều dương, thực hiện bỏ bit dấu, trị tuyệt đối của chúng đã ở dạng nhị phân thực sự và được nhân ở chính đạng đó. Tích số kết quả, dĩ nhiên là dương và được gán bít dấu là 0.

+ Nếu hai số là âm, thực hiện bù hai của mỗi số để chuyển thành số dương. Bỏ bit dấu, thực hiện nhân trị tuyệt đối. Tích số kết quả vẫn là dương và được gán bit dấu 0.

+ Nếu là một số âm và một số dương, thực hiện lấy bù 2 của số âm để chuyển thành dương. Bỏ bit dấu, thực hiện nhân trị tuyệt đối. Tuy nhiên, kết quả phải âm do các số ban đầu trái dấu. Vì vậy, tích số sau đó được chuyển sang dạng bù hai và được gán bit dấu là 1.

Ví dụ 1.5.5.1. Nhân hai số nhị phân có dấu sau: 01001100 và 11010110.

Giải:

Số thứ nhất có bit dấu là 0 nên nó là số dương.

Số thứ hai có bit dấu là 1 nên nó là số âm, phải lấy bù hai để chuyển thành số dương: 11010110 → 00101010

Bit dấu của kết quả là 1.

Thực hiện phép nhân hai trị tuyệt đối như trên hình 1.5.5.1.

Do kết quả là một số âm nên kết quả sau khi nhân được lấy bù hai và thêm bit dấu là 1: 1100110001000 → 0011001111000

Kết quả cuối cùng: 1 0011001111000

```
      1001100
    × 1010110
  ----------
      0000000
    + 1001100
  ----------
     10011000
   + 1001100
  ----------
    111001000
  + 0000000
  ----------
    111001000
  + 1001100
  ----------
   11010001000
  + 0000000
  ----------
   11010001000
  + 1001100
  ----------
  1100110001000        Kết quả sau khi nhân
```

Hình 1.5.5.1. Thực hiện phép nhân

1.5.6. Chia nhị phân

Ví dụ, chia hai số nhị phân không dấu:

```
a)  1001 | 11          b)  1010 | 100
  - 011  | 0011          - 100  | 0010
  ------                 ------
    0011                   010
  - 0011
  ------
       0
```

Phép trừ trong các phép chia thường được thực hiện bằng cách cộng với bù hai của số trừ.

Chia trong hệ bù 2:

Đầu tiên phải xác định bit dấu của kết quả theo bit dấu của số bị chia và số chia theo quy tắc như đối với phép nhân. Lấy bù hai số âm, sau khi ta đã có cả hai số là dương. Quá trình thực hiện phép chia đối với hai số dương được tiến hành như sau:

– Ban đầu thương số là 0.

– Lấy số bị chia cộng với bù hai của số chia ta nhận được phần số dư đầu tiên. Nếu số dư đó là dương hoặc 0 thì cộng thêm 1 vào thương số và chuyển sang bước tiếp theo. Nếu số dư bằng 0 hoặc âm thì quá trình chia kết thúc.

– Lấy số dư đã nhận được ở bước trên cộng với bù hai của số chia ta nhận phần số dư thứ hai. Nếu số dư đó là dương hoặc 0 thì lặp lại quá trình đối với số dư vừa nhận được. Thực hiện cho đến khi số dư bằng 0 hoặc âm thì quá trình chia kết thúc.

Như vậy, thương số có giá trị bằng số lần cộng với bù 2 của số chia.

Sau khi nhận được thương số ta sẽ tuỳ từng trường hợp để đưa ra kết quả cuối

cùng. Nếu bit dấu của kết quả là 0 thì kết quả cuối cùng chính là thương số nhận được, nếu bit dấu của kết quả là 1 thì kết quả cuối cùng là bù hai của thương số.

Ví dụ 1.5.6.1. Thực hiện chia hai số nhị phân có dấu: số bị chia 01100100 và số chia 00011001.

Giải:

+ Bit dấu của cả hai số là dương nên kết quả cũng là số dương.

+ Đầu tiên thương số là 0: 0000000

+ Lấy số bị chia cộng với bù hai của số chia:

```
  01100100
+ 11100111
----------
  01001011   Số dư đầu tiên là một số dương
```

Cộng 1 vào thương: 00000000 + 1 = 00000001.

+ Lấy số dư vừa nhận được cộng với bù hai của số chia:

```
  01001011
+ 11100111
----------
  00110010   Số dư thứ 2 là một số dương
```

Cộng 1 vào thương: 00000001 + 1 = 00000010

+ Lấy số dư vừa nhận được cộng với bù hai của số chia:

```
  00110010
+ 11100111
----------
  00011001   Số dư thứ 3 là một số dương
```

Cộng 1 vào thương: 00000010 + 1 = 00000011

+ Lấy số dư vừa nhận được cộng với bù hai của số chia:

```
  00011001
+ 11100111
----------
  00000000   Số dư thứ 4 là 0
```

Cộng 1 vào thương: 00000011 + 1 = 00000100

Số dư thứ 4 là 0 nên quá trình chia kết thúc, thương nhận được là: **00000100.**

Ví dụ 1.5.6.2. Thực hiện chia hai số nhị phân có dấu: số bị chia 01100100 và số chia 11001110.

Giải:

+ Bit dấu của hai số là ngược nhau nên kết quả là số âm.

+ Lấy bù hai của số âm để chuyển thành dương: 11001110 → 00110010 (số chia mới).

+ Đầu tiên thương số là 0: 0000000

+ Lấy số bị chia cộng với bù hai của số chia mới:

```
  01100100
+ 11001110
----------
  00110010    Số dư đầu tiên là một số dương
```

Cộng 1 vào thương: 00000000 + 1 = 00000001

+ Lấy số dư vừa nhận được cộng với bù hai của số chia mới:

```
  00110010
+ 11001110
----------
  00000000    Số dư thứ 2 là 0
```

Cộng 1 vào thương: 00000001 + 1 = 00000010

Số dư thứ 2 là 0 nên quá trình chia kết thúc, thương nhận được là: **00000010.**

Kết quả cuối cùng là bù hai của thương nhận được: **11111110.**

1.6. CỘNG TRỪ TRONG HỆ THẬP LỤC PHÂN

1.6.1. Cộng trong hệ thập lục phân

Khi cộng trong hệ thập lục phân sử dụng các quy tắc sau:

– Cộng các ký số có trọng số tương ứng với nhau, giá trị theo hệ thập phân.

– Nếu tổng của hai ký số nhỏ hơn hoặc bằng 15_{10} thì kết quả được biểu diễn ở dạng thập lục phân (0 đến F).

– Nếu tổng của hai số lớn hơn 15_{10} thì lấy giá trị đó trừ đi 16_{10} và nhớ 1 sang ký số có trọng số cao hơn liền kề.

Ví dụ 1.6.1.1. Thực hiện cộng các số hệ thập lục phân sau:

a) $23_{16} + 16_{16}$; b) $2B_{16} + 84_{16}$; c) $DF_{16} + AC_{16}$

Giải:

a)
$$\begin{array}{r} 23_{16} \\ +\ 16_{16} \\ \hline 39_{16} \end{array}$$
Cột phải: $3_{16} + 6_{16} = 3_{10} + 6_{10} = 9_{10} = 9_{16}$
Cột trái: $2_{16} + 1_{16} = 2_{10} + 1_{10} = 3_{10} = 3_{16}$

b)
$$\begin{array}{r} 2B_{16} \\ +\ 84_{16} \\ \hline AF_{16} \end{array}$$
Cột phải: $B_{16} + 4_{16} = 11_{10} + 4_{10} = 15_{10} = F_{16}$
Cột trái: $2_{16} + 8_{16} = 2_{10} + 8_{10} = 10_{10} = A_{16}$

c)
$$\begin{array}{r} DF_{16} \\ +\ AC_{16} \\ \hline 18B_{16} \end{array}$$
Cột phải: $F_{16} + C_{16} = 15_{10} + 12_{10} = 27_{10}$
$27_{10} - 16_{10} = 11_{10} = B_{16}$ nhớ 1
Cột trái: $D_{16} + A_{16} + 1_{16} = 13_{10} + 10_{10} + 1_{10} = 24_{10}$
$24_{10} - 16_{10} = 8_{10} = 8_{16}$ nhớ 1

1.6.2. Trừ trong hệ thập lục phân

Phương pháp trừ trong hệ thập lục phân như sau:

Chuyển số trừ trong hệ thập lục phân sang nhị phân. Lấy bù hai của số nhị phân đó và chuyển số bù hai đó sang hệ thập lục phân. Thực hiện cộng số bị trừ với số thập lục phân mới tạo thành.

Ví dụ 1.6.2.1. Thực hiện trừ các số hệ thập lục phân sau:

a) $84_{16} - 2A_{16}$

b) $C3_{16} - 0B_{16}$

Giải:

a) $2A_{16} = 00101010$

Lấy bù 2 của $2A_{16} = 11010110 = D6_{16}$

Thực hiện cộng:

$$\begin{array}{r} 84_{16} \\ +\ D6_{16} \\ \hline \not{1}5A_{16} \end{array}$$

Bit nhớ sinh ra ở vị trí đầu tiên được bỏ đi, như trong hệ bù 2

Kết quả cuối cùng: $5A_{16}$

b) $0B_{16} = 00001011$

Lấy bù 2 của $0B_{16} = 11110101 = F5_{16}$

Thực hiện cộng:

$$\begin{array}{r} C3_{16} \\ +\ F5_{16} \\ \hline \not{1}B8_{16} \end{array}$$

Bit nhớ sinh ra ở vị trí đầu tiên được bỏ đi, như trong hệ bù 2

Kết quả cuối cùng: $B8_{16}$.

1.7. MÃ HÓA SỐ CỦA HỆ THẬP PHÂN

Máy tính và các mạch số được dùng để thao tác dữ liệu có thể là số, chữ cái hay các ký tự đặc biệt. Vì các mạch số làm việc ở dạng nhị phân nên các số, các chữ cái và các ký tự đặc biệt khác phải được cải tạo thành khuôn dạng nhị phân. Có nhiều cách để làm việc này và quá trình này gọi là mã hoá. Tồn tại nhiều mã số và các mã khác nhau phục vụ những mục đích khác nhau. Các mã còn được sử dụng để dò và sửa lỗi.

1.7.1. Mã BCD

Trong kỹ thuật số để chuyển đổi các con số giữa hai hệ đếm cơ số 2 và cơ số 10 một cách tự động người ta dùng phương pháp biểu diễn nhị phân. Người ta dùng một nhóm bốn bit nhị phân để biểu diễn mười chữ số của hệ đếm thập phân. Phương pháp biểu diễn này được gọi là phương pháp mã hoá các con số trong hệ đếm 10 bằng nhóm mã hệ nhị phân (Binary Coded Decimal BCD). Thực ra đó là số thập phân được

viết kiểu nhị phân. Các chữ số của hệ 10 từ 0,1,...,9 đều được biểu diễn bằng một số nhị phân có 4 bit. Tuỳ theo cách sử dụng 10 trên 16 tổ hợp mã nhị phân 4 bit mà ta có các loại mã BCD khác nhau. Số nhị phân 4 bit có trọng số 8 – 4 – 2 – 1 được gọi là mã BCD 8421 (hoặc mã BCD có trọng số tự nhiên).

Ví dụ 1.7.1.1:

Số hệ thập phân Biểu diễn bằng mã BCD 8421 Biểu diễn bằng số nhị phân

16 $(0001\ 0110)_{BCD}$ $(10000)_2$

Mã BCD 8421 được dùng để chuyển các con số từ hệ 10 sang hệ 2 và ngược lại. Nhìn một con số lớn viết ở hệ nhị phân ta khó hình dung độ lớn của nó ở hệ 10. Nhưng viết ở mã BCD ta dễ hình dung ra độ lớn của nó.

Trong thực tế, đôi khi mã BCD 8421 dùng không thuận lợi, lúc đó người ta dùng các mã BCD có trọng số 2421, 5121, 7421.

– Bảng các loại mã BCD có trọng số khác nhau như trên hình 1.7.1.1.

– Cộng các số mã BCD:

Cộng 2 số mã BCD được thực hiện như sau:

+ Phép cộng được thực hiện theo quy tắc cộng nhị phân bình thường.

+ Nếu tổng bốn bit có giá trị nhỏ hơn hoặc bằng 9, đó chính là số mã BCD.

+ Nếu tổng bốn bit có giá trị lớn hơn 9 (hoặc ngoài nhóm 4 bit biểu diễn mã BCD) thì phải cộng 6_{10} (0110) vào tổng đó, bit nhớ được mang sang bốn bit liền kề trước đó.

Số hệ 10	Mã BCD			
	Trọng số 8421	Trọng số 7421	Trọng số 2421	Trọng số 5121
0	0000	0000	0000	0000
1	0001	0001	0001	0001
2	0010	0010	0010	0010
3	0011	0011	0011	0011
4	0100	0100	0100	0111
5	0101	0101	0101	1000
6	0110	0110	0110	1001
7	0111	0111	0111	1010
8	1000	1001	1110	1011
9	1001	1010	1111	1111

Hình 1.7.1.1. Bảng các loại mã BCD có trọng số khác nhau

Ví dụ 1.7.1.2. Cộng các số BCD sau:

a) 0011 + 0101; b) 010001100010 + 010100100011

c) 1001 + 0100; d) 1001 + 1000; e) 01100011 + 01010011

Giải:

Các số thập phân được cộng tương ứng để so sánh.

a)
```
  0011      3
+ 0101    + 5
------    ---
  1000      8
```

b)
```
  0100 0110 0010      462
+ 0101 0010 0011    + 523
-----------------   -----
  1001 1000 0101      985
```

c)
```
  1001
+ 0100
------
  1101    không phải số BCD (>9)
+ 0110    +6
------
0001 0011 Số BCD
```

```
  9
+ 4
---
 13
```

d)
```
  1001
+ 1000
------
 1 0001   không phải số BCD (có số nhớ)
+ 0110    + 6
------
0001 0111 Số BCD
```

```
  9
+ 8
---
 17
```

e)
```
  0110 0111
+ 0101 0011
-----------
  1011 1010   cả hai nhóm không phải mã BCD (>9)
+ 0110 0110   + 6 vào cả hai nhóm
-----------
0001 0010 0000  Số BCD
```

```
  67
+ 53
----
 120
```

– Chuyển đổi mã BCD sang nhị phân

Thực hiện bằng cách: "Tính tổng nhị phân của các số nhị phân tương đương với giá trị thập phân của các bit 1 ở dạng mã BCD".

Đối với mã nhị phân trọng số tương ứng của các bit là: ...64, 32, 16, 8, 4, 2, 1. Còn đối với mã BCD thì trọng số tương ứng của các bit 1 là: ...80, 40, 20, 10, 8, 4, 2, 1 vì 4 bit cuối là hàng đơn vị, 4 bit tiếp theo là hàng chục,...trong đó trọng số của từng 4 bit một vẫn là 8,4,2,1.

Ví dụ 1.7.1.3. Chuyển đổi số 01010011 (53_{10}) dạng mã BCD sang mã nhị phân.

Giải: Viết ra mã nhị phân tương đương cho mọi bit 1 trong mã BCD, sau đó cộng tất cả theo hệ nhị phân.

Như vậy, mã nhị phân tương ứng là 0110101.

Ngoài mã BCD nói trên là những mã có trọng số ra, ta còn gặp một số mã thông dụng khác không có trọng số được nêu ra trong bảng trên hình 1.7.1.2.

– ***Mã dư 3 (XS – 3) :*** Mã này cũng dùng 4 bit nhị phân để mã hoá từng chữ số trong hệ thập phân. Từ mã nhị phân 4 bit đó được tạo thành bằng cách cộng thêm 3 đơn vị vào mã BCD 8421. Mã này dùng trong các thiết bị tính toán số học và xử lý tín hiệu số.

– ***Mã Gray*:** Đặc điểm của mã này là hai số kế tiếp nhau chỉ khác nhau 1 bit. Vì vậy tốc độ đếm của mã Gray trong máy tính nhanh hơn so với mã nhị phân.

+ Phương pháp chuyển từ mã nhị phân sang Gray:

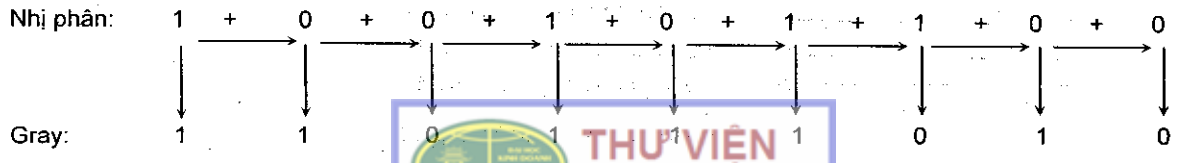
1. Bit có trọng số lớn nhất trong mã nhị phân được giữ nguyên khi chuyển sang mã Gray.

2. Từ trái sang phải cộng modul (cộng không nhớ) hai bit nhị phân liền kề nhau để tạo ra bit tiếp theo trong mã Gray.

Số hệ 10	Dư 3	Gray	Johnson
0	0011	0000	00000
1	0100	0001	00001
2	0101	0011	00011
3	0110	0010	00111
4	0111	0110	01111
5	1000	0111	11111
6	1001	0101	11110
7	1010	0100	11100
8	1011	1100	11000
9	1100	1101	10000
10		1111	
11		1110	
12		1010	
13		1011	
14		1001	
15		1000	

Hình 1.7.1.2. Một số mã thường dùng

Ví dụ 1.7.1.4. Số nhị phân: 100101100 chuyển sang mã Gray như sau:

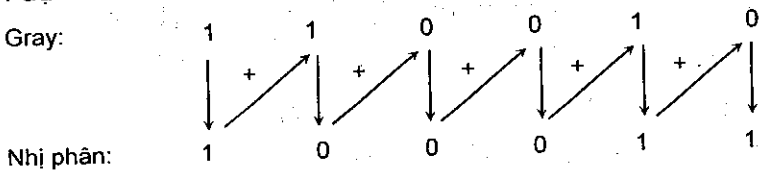


+ Phương pháp chuyển từ mã Gray sang nhị phân:

1. Bit có trọng số lớn nhất trong mã Gray được giữ nguyên khi chuyển sang mã nhị phân.

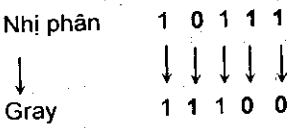
2. Từ trái sang phải cộng từng bit vừa tạo thành trong mã nhị phân với bit liền kề trong mã Gray để tạo thành bit tiếp theo trong mã nhị phân.

Ví dụ 1.7.1.5. Số 110010 biểu diễn bằng mã Gray chuyển sang nhị phân như sau:



Mã Gray có thể được suy ra từ mã nhị phân bằng cách đảo bit đứng bên phải bit 1 của mã nhị phân.

Ví dụ 1.7.1.6:



– ***Mã Johnson***: Cũng sử dụng 5 chữ số hệ 2 để biểu diễn các số hệ 10. Đặc điểm là khi chuyển sang số tiếp theo sẽ thay chữ số "0" bằng chữ số "1" bắt đầu từ phải sang trái cho đến khi đạt đến 11111 ứng với số 5 trong hệ 10 thì lại thay thế dần chữ số "1" bằng chữ số "0" cũng theo chiều từ phải sang trái.

Ví dụ 1.7.1.7. Biểu diễn bằng mã Jonhson số $(257)_{10}$.

Giải:

$$(257)_{10} = (000111111111100)_{Jonhson}$$

Ví dụ 1.7.1.8. Biểu diễn bằng mã dư 3 số $(257)_{10}$.

Giải:

$$(257)_{10} = (010110001010)_{dư\ 3}$$

BÀI TẬP CHƯƠNG 1

Bài 1.1. Chuyển đổi giữa các hệ thống số đếm:

a) $(132)_{10} \rightarrow (\ )_2 \rightarrow (\ )_8 \rightarrow (\ )_{16}$

b) $(1000111,101)_2 \rightarrow (\ )_{10} \rightarrow (\ )_{16} \rightarrow (\ )_8$

c) $(1A,2C)_{16} \rightarrow (\ )_2 \rightarrow (\ )_8 \rightarrow (\ )_{10}$

Bài 1.2. Viết bốn số tiếp theo của số nhị phân: 100011011

Bài 1.3 Cho số nhị phân có dấu trong hệ bù hai, xác định giá trị thập phân trong mỗi trường hợp:

a) 1 001010

b) 0 10101001

Bài 1.4. Thực hiện các phép tính sau trong hệ bù hai:

a) $(+9)_{10} + (+22)_{10}$; $(+21)_{10} + (-27)_{10}$ với số bit quy định cho trị tuyệt đối là 7.

b) $(+15)_{10} - (+29)_{10}$; $(-11)_{10} - (-30)_{10}$ với số bit quy định cho trị tuyệt đối là 7.

c) $(+25)_{10} \times (+9)_{10}$; $(-11)_{10} \times (-13)_{10}$; $(+36)_{10} \times (-17)_{10}$

d) $(+25)_{10} : (+5)_{10}$; $(-55)_{10} : (+11)_{10}$; $(-36)_{10} : (-6)_{10}$

Kiểm tra lại kết quả bằng cách thực hiện phép tính trong hệ thập phân.

Bài 1.5. Xác định khoảng giá trị thập phân có dấu có thể biểu diễn được bằng 2 byte ở trong hệ bù 2.

Bài 1.6. Với 2 byte có thể biểu diễn giá trị thập phân lớn nhất bằng mã nhị phân thông thường và bằng mã BCD8421 là bao nhiêu?

Bài 1.7. Chuyển đổi $(1001100001110110)_{BCD8421}$ sang mã nhị phân thông thường.

Bài 1.8. Thực hiện các phép tính sau trong hệ thập lục phân:

a) $79_{16} + 43_{16}$; $C7_{16} + BF_{16}$

b) $E9_{16} - 25_{16}$; $AB_{16} - 8D_{16}$

Bài 1.9. Cộng các số mã BCD sau:

a) 01111000 + 00010000; b) 10000111 + 10010100

Bài 1.10. Chuyển đổi các số thập phân sau sang mã Gray:

a) 79_{10} ; b) 136_{10}

Chương 2

ĐẠI SỐ LOGIC

Đại số logic còn được gọi là đại số Boole. Lý thuyết này do George Boole – nhà toán học người Anh đưa ra năm 1847.

2.1. CƠ SỞ ĐẠI SỐ LOGIC

Ta đã biết mạch số hoạt động ở chế độ nhị phân, nơi mỗi điện thế vào và ra sẽ có giá trị 0 hoặc 1; việc chỉ định giá trị 0 và 1 biểu thị khoảng điện thế định sẵn. Đặc điểm này của mạch logic cho phép sử dụng đại số logic làm công cụ phân tích và thiết kế các hệ thống kỹ thuật số.

Đại số logic dùng để phân tích hay thiết kế những mạch điện có quan hệ giữa biến và hàm. Trong đó biến và hàm chỉ nhận một trong hai giá trị là 0 và 1, hai giá trị này không biểu thị số lượng to nhỏ cụ thể mà chủ yếu là để biểu thị hai trạng thái logic khác nhau (đúng và sai, cao và thấp, mở và đóng, ...).

Đại số logic là phương tiện biểu diễn mối quan hệ giữa đầu ra và đầu vào của mạch logic dưới dạng phương trình đại số. Đầu vào sẽ được xem là các biến logic có mức logic quyết định mức logic của đầu ra (hàm logic) tại thời điểm bất kỳ. Biến logic và hàm logic thường được ký hiệu bằng chữ cái.

Tóm lại ta có:

x_i là biến logic khi x_i chỉ lấy một trong hai giá trị là 0 và 1 ($x_i \in \{0,1\}$).

Tập hợp n biến logic có 2^n tổ hợp giá trị khác nhau. Giá trị thập phân tương ứng biểu diễn các tổ hợp này là: $0 \div 2^n - 1$.

$F(x_1, x_2, \ldots, x_n)$ là hàm logic khi các biến của hàm là biến logic và F chỉ lấy một trong hai giá trị 0 hoặc 1.

Trong thực tế, đại số logic chỉ có ba phép toán cơ bản: OR. AND và NOT. Các phép toán cơ bản này được gọi là phép toán logic.

2.2. CÁC PHÉP TOÁN LOGIC VÀ CÁC CỔNG LOGIC CƠ BẢN

2.2.1. Phép toán OR và cổng OR

a) Phép toán OR hay còn được gọi là phép cộng logic

– Hàm OR (hàm hoặc): $y = x_1 + x_2$

– Bảng trạng thái như trên hình 2.2.1.1a.

– Mạch điện minh hoạ quan hệ logic OR (hình 2.2.1.1b):

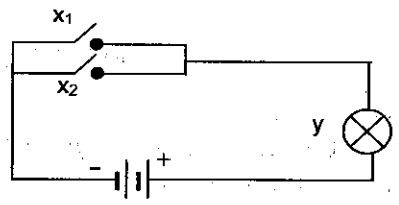
– Mở rộng cho trường hợp tổng quát có n biến:

$$y = x_1 + x_2 + \ldots + x_n$$

Mạch điện thực hiện quan hệ logic OR được gọi là cổng OR.

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

a)



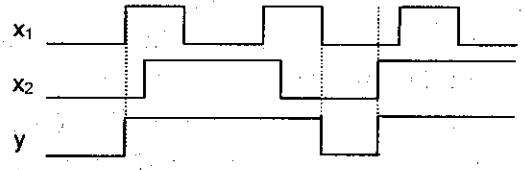
b)

Hình 2.2.1.1. a) Bảng trạng thái; b) Mạch điện minh hoạ quan hệ logic

b) Cổng OR

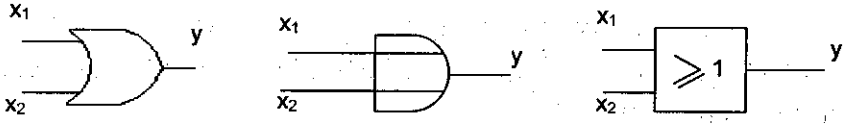
– Định nghĩa: Là mạch có từ hai đầu vào trở lên và có đầu ra bằng tổ hợp OR các biến đầu vào.

– Giản đồ thời gian hình 2.2.1.2:



Hình 2.2.1.2. Giản đồ thời gian của cổng OR

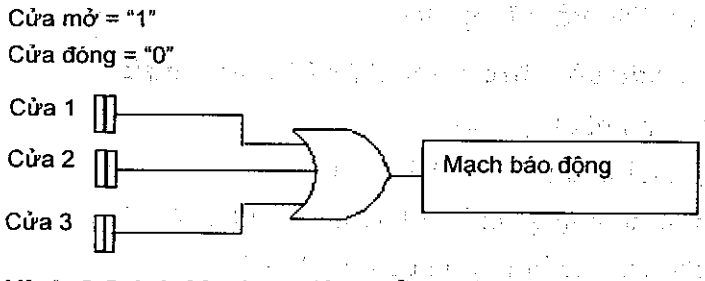
– Ký hiệu logic hình 2.2.1.3:



Hình 2.2.1.3. Ký hiệu logic của cổng OR

Ví dụ 2.2.1.1. Xét ứng dụng của cổng OR trong mạch tự động cảnh báo trạng thái của các cửa.

Hệ thống có thể sử dụng cho một ngôi nhà có 2 cửa sổ và 1 cửa chính, mạch sẽ báo động khi có ít nhất một cửa bị mở. Trên mỗi cửa được gắn một cảm biến, cảm biến tạo ra mức logic 1 khi cửa mở và mức logic 0 khi cửa đóng, hình 2.2.1.4.



Hình 2.2.1.4. Mạch tự động cảnh báo trạng thái các cửa

2.2.2. Phép toán AND và cổng AND

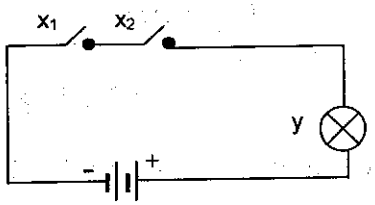
a) Phép toán AND hay còn được gọi là phép nhân logic

– Hàm AND (hàm và): $y = x_1.x_2$

– Bảng trạng thái như trên hình 2.2.2.1a.

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

a)



b)

Hình 2.2.2.1. a) Bảng trạng thái b) Mạch điện minh hoạ quan hệ logic AND

– Mạch điện minh hoạ quan hệ logic AND (hình 2.2.2.1b):

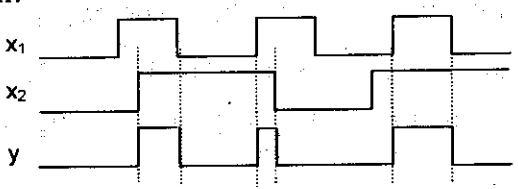
– Mở rộng cho trường hợp tổng quát có n biến: $y = x_1 . x_2 .... . x_n$.

Mạch điện thực hiện quan hệ logic AND được gọi là cổng AND.

b) Cổng AND

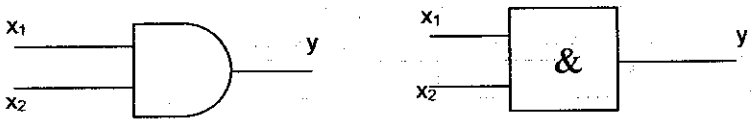
+ Định nghĩa: là mạch có từ hai đầu vào trở lên và một đầu ra bằng tổ hợp AND các biến đầu vào.

+ Giản đồ thời gian:



Hình 2.2.2.2. Giản đồ thời gian của cổng AND

+ Ký hiệu logic:

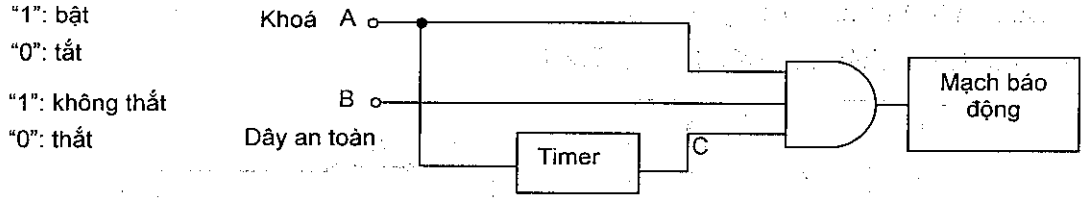


Hình 2.2.2.3. Ký hiệu logic của cổng AND

Ví dụ 2.2.2.1. Xét ứng dụng của cổng AND trong hệ thống tự động cảnh báo khóa dây an toàn cho lái xe.

Hình 2.2.2.4, mạch sẽ cảnh báo khi khoá xe được bật và dây an toàn chưa thắt. Nếu khoá xe được bật, điện áp tương ứng mức logic 1 được cung cấp cho đầu vào A của cổng AND. Nếu dây an toàn chưa thắt, điện áp tương ứng mức logic 1 được cung

cấp cho đầu vào B của cổng AND. Đồng thời khi khoá xe bật, điện áp tương ứng mức logic 1 được cung cấp cho đầu vào C của cổng AND trong 30 giây. Như vậy, khi khoá xe bật mà dây an toàn chưa thắt, mạch báo động sẽ báo trong 30 giây.



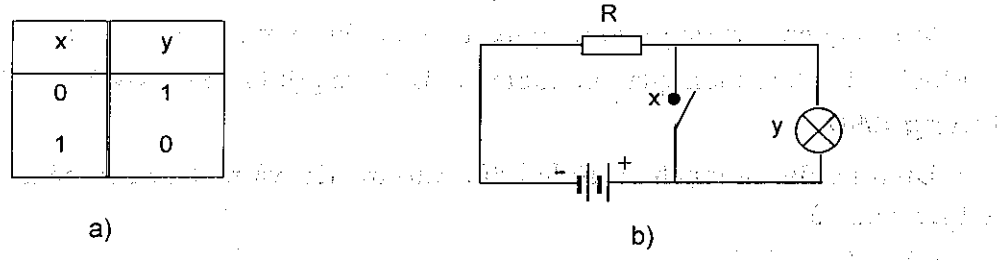
Hình 2.2.2.4. Một ứng dụng của cổng AND

2.2.3. Phép toán NOT và cổng NOT

a) Phép toán NOT hay còn được gọi phép đảo hay phép phủ định

– Hàm NOT (hàm đảo): $y = \bar{x}$

– Bảng trạng thái như trên hình 2.2.3.1a.



x	y
0	1
1	0

a)

b)

Hình 2.2.3.1. a) Bảng trạng thái ; b) Mạch điện minh hoạ quan hệ logic NOT

– Mạch điện minh hoạ quan hệ logic NOT trên hình 2.2.3.1b.

Mạch điện thực hiện quan hệ logic NOT được gọi là cổng NOT.

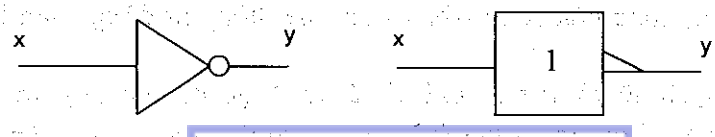
b) Cổng NOT

– Định nghĩa: Là mạch có duy nhất một đầu vào và mức logic ở đầu ra luôn ngược với mức logic ở đầu vào.

– Giản đồ thời gian trên hình 2.2.3.2:

Hình 2.2.3.2. Giản đồ thời gian của cổng NOT

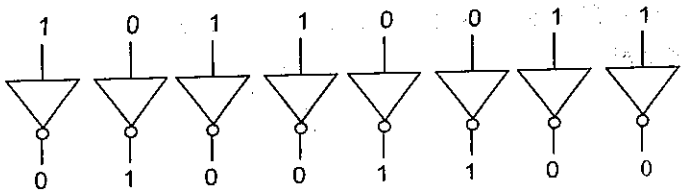
– Ký hiệu logic trên hình 2.2.3.3:



Hình 2.2.3.3. Ký hiệu logic của cổng NOT

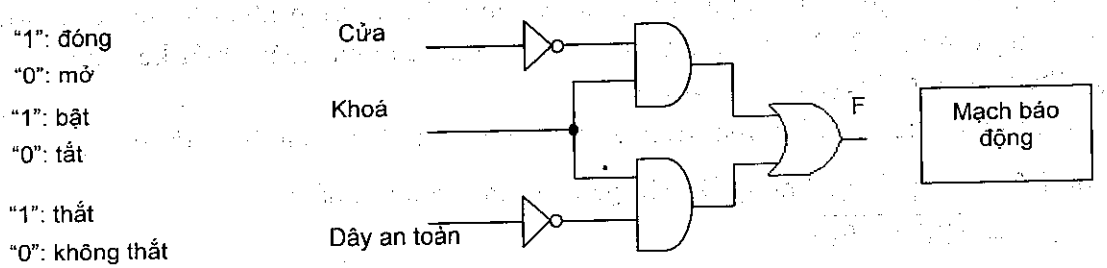
Ví dụ 2.2.3.1. Sử dụng cổng đảo để tạo ra số bù 1.

Hình 2.2.3.4 minh họa mạch tạo ra số bù 1 của số nhị phân 8 bit. Đầu vào là số nhị phân, đầu ra là bù 1 của số nhị phân đó.



Hình 2.2.3.4. Mạch tạo số bù 1 sử dụng cổng NOT

Ví dụ 2.2.3.2. Hình 2.2.3.5 minh họa một mạch logic dùng để kích hoạt một tín hiệu báo động khi đầu ra F của nó ở mức logic cao. Mạch báo động cảnh báo cho lái xe biết cửa mở hoặc dây an toàn chưa khóa khi xe nổ máy.



Hình 2.2.3.5. Một ứng dụng của các cổng cơ bản

2.3. CÁC ĐỊNH LUẬT CƠ BẢN CỦA ĐẠI SỐ LOGIC

1. Các mệnh đề cơ sở

$x + 0 = x$ $\quad$ $x + 1 = 1$ $\quad$ $x + \bar{x} = 1$

$x \cdot 0 = 0$ $\quad$ $x \cdot 1 = x$ $\quad$ $x \cdot \bar{x} = 0$

2. Định luật đồng nhất

$x + x = x$

$x \cdot x = x$

3. Định luật phủ định của phủ định

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

4. Định luật kết hợp

$x_1 + (x_2 + x_3) = (x_1 + x_2) + x_3$

$x_1 \cdot (x_2 \cdot x_3) = (x_1 \cdot x_2) \cdot x_3$

5. Định luật giao hoán

$x_1 + x_2 = x_2 + x_1$

$x_1 \cdot x_2 = x_2 \cdot x_1$

6. Định luật phân phối

$$x_1(x_2 + x_3) = x_1.x_2 + x_1.x_3$$

$$(x_1 + x_2)(x_1 + x_3) = x_1.x_1 + x_1.x_3 + x_2.x_1 + x_2.x_3$$
$$= x_1(1 + x_2 + x_3) + x_2.x_3 = x_1 + x_2.x_3$$

7. Định lý DE MORGAN

$$\overline{x_1.x_2} = \overline{x_1} + \overline{x_2}$$

$$\overline{x_1 + x_2} = \overline{x_1}.\overline{x_2}$$

Định lý này có thể mở rộng cho hàm nhiều biến:

$$\overline{x_1.x_2...x_n} = \overline{x_1} + \overline{x_2} + ... + \overline{x_n}$$

$$\overline{x_1 + x_2 + ... + x_n} = \overline{x_1}.\overline{x_2}...\overline{x_n}$$

Định lý này giúp ta chuyển phép cộng logic thành phép nhân logic và ngược lại. Vận dụng định lý De Morgan chúng ta có thể giải các bài toán thiết kế mạch logic tổ hợp theo các cửa logic cơ bản cho sẵn.

Chú ý: Trong các định luật trên, x_i có thể là biến đơn hoặc biểu thức.

Ví dụ 2.3.1.1. Áp dụng định lý De Morgan cho các hàm sau đây:

a) $F = \overline{ABC + DEF}$

b) $F = \overline{A\overline{B} + \overline{C}D + EF}$

c) $F = \overline{\overline{A + B + C}}$

d) $F = \overline{(A + B)\overline{CD} + E + \overline{F}}$

e) $F = \overline{\overline{A + B\overline{C} + D(E + \overline{F})}}$

Giải:

a) $F = \overline{ABC + DEF} = \overline{ABC}.\overline{DEF} = (\overline{A} + \overline{B} + \overline{C})(\overline{D} + \overline{E} + \overline{F})$

b) $F = \overline{A\overline{B} + \overline{C}D + EF} = \overline{A\overline{B}}.\overline{\overline{C}D}.\overline{EF} = (\overline{A} + B)(C + \overline{D})(\overline{E} + \overline{F})$

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

d) $F = \overline{(A + B)\overline{CD} + E + \overline{F}} = \overline{(A + B)\overline{CD}}.\overline{(E + \overline{F})}$

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

e) $F = \overline{\overline{A + B\overline{C} + D(E + \overline{F})}} = \overline{\overline{A + B\overline{C}}}.\overline{D(E + \overline{F})}$

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

2.4. CÁC PHƯƠNG PHÁP BIỂU DIỄN HÀM LOGIC

Trước hết ta xét khái niệm hàm xác định đầy đủ và không xác định đầy đủ.

Hàm xác định đầy đủ là hàm có trị số xác định với mọi tổ hợp biến. Hàm không thoả mãn điều kiện trên là hàm không xác định đầy đủ. Tại những tổ hợp biến mà trị số của hàm không xác định (có thể là "0" hoặc "1") giá trị của hàm sẽ được ký hiệu bằng dấu "x". Những tổ hợp biến này cũng có thể không bao giờ xảy ra.

2.4.1. Biểu diễn hàm bằng bảng trạng thái

Tương tự như trong đại số thông thường, một hàm logic có thể được biểu diễn bởi bảng giá trị của hàm số đó. Là bảng miêu tả quan hệ giữa các giá trị của hàm số tương ứng với mọi giá trị có thể của biến số.

Một hàm có n biến, bảng sẽ có (n + 1) cột (trong đó n cột là giá trị của biến và một cột là giá trị của hàm) và 2^n hàng tương ứng với 2^n tổ hợp giá trị khác nhau của n biến vào. Ứng với mỗi tổ hợp giá trị biến ghi giá trị hàm tương ứng. Để khỏi bỏ sót hoặc trùng lặp ta nên sắp xếp các tổ hợp biến lối vào tuần tự theo số đếm nhị phân.

Ví dụ 2.4.1.1. Đèn báo hiệu của một hội đồng giám khảo gồm 3 thành viên sẽ sáng nếu đa số trong các thành viên đều đóng công tắc bỏ phiếu thuận. Lập bảng trạng thái của hàm số logic đó.

Giải: Gọi A, B, C là ba công tắc, công tắc đóng thì các biến A, B, C lấy giá trị 1, công tắc ngắt thì các biến lấy giá trị 0. Gọi F là trạng thái của đèn được điều khiển, đèn sáng F = 1, đèn tắt F = 0. Ta được bảng trạng thái như trên hình 4.2.1.1.

Lối vào			Lối ra
A	B	C	F
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	1

Hình 2.4.1.1. Bảng trạng thái của hàm 3 biến

Dùng bảng trạng thái để biểu diễn hàm tuy có ưu điểm là rõ ràng trực quan, nhưng có nhược điểm là cách biểu diễn này sẽ trở nên rối rắm khi hàm có nhiều biến.

2.4.2. Biểu diễn hàm bằng phương trình logic

Trước hết ta xét khái niệm về minterm (số hạng tối thiểu) và maxterm (số hạng tối đa):

Một hàm logic có n biến, mỗi biến có thể nhận một trong hai giá trị 0 hoặc1, vậy ta sẽ có 2^n tổ hợp biến. Mỗi tổ hợp biến ta có thể tạo thành một số hạng là tích tất cả các biến có trong cùng một tổ hợp biến. Trong các số hạng đó biến bằng 1 được viết giữ nguyên biến còn biến bằng 0 được viết đảo biến, các số hạng này được gọi là

minterm (số hạng tối thiểu). Gọi là số hạng tối thiểu vì minterm là tích các biến có trong một tổ hợp biến, tích này chỉ bằng 1 khi tất cả các biến đều bằng 1. Như vậy, ứng với mỗi một minterm ta chỉ tìm được một tổ hợp giá trị biến tương ứng để nó bằng 1 và chỉ có một tổ hợp biến mà thôi.

Mỗi tổ hợp biến ta cũng có thể tạo thành một số hạng là tổng tất cả các biến có trong cùng một tổ hợp biến. Trong các số hạng đó biến bằng 0 được viết giữ nguyên biến còn biến bằng 1 được viết đảo biến, các số hạng này được gọi là maxterm (số hạng tối đa). Maxterm là tổng tất cả các biến có trong tổ hợp biến nên chỉ cần trong các biến bằng 1 thì maxterm bằng 1, maxterm bằng 0 chỉ trong một trường hợp duy nhất ứng với tất cả các biến trong tổ hợp biến đều bằng 0. Như vậy, các trường hợp maxterm bằng 1 là tối đa, trường hợp minterm bằng 1 là tối thiểu. Một hàm có n biến ta có 2^n maxterm và 2^n minterm.

Ví dụ 2.4.2.1. Một hàm F(A,B,C) có ba biến là A, B, C ta có 8 tổ hợp biến được sắp xếp một cách trình tự theo nhị phân là: 000, 001, 010, 011, 100, 101, 110, 111. Tương ứng với 8 tổ hợp biến này ta có 8 số hạng tối thiểu minterm ký hiệu là m_0, m_1, …, m_7, và 8 số hạng tối đa maxterm ký hiệu là M_0, M_1, … , M_7 như trên hình 2.4.2.1.

Biến			Minterm	Maxterm
A	B	C		
0	0	0	$m_0 = \bar{A}\bar{B}\bar{C}$	$M_0 = A + B + C$
0	0	1	$m_1 = \bar{A}\bar{B}C$	$M_1 = A + B + \bar{C}$
0	1	0	$m_2 = \bar{A}B\bar{C}$	$M_2 = A + \bar{B} + C$
0	1	1	$m_3 = \bar{A}BC$	$M_3 = A + \bar{B} + \bar{C}$
1	0	0	$m_4 = A\bar{B}\bar{C}$	$M_4 = \bar{A} + B + C$
1	0	1	$m_5 = A\bar{B}C$	$M_5 = \bar{A} + B + \bar{C}$
1	1	0	$m_6 = AB\bar{C}$	$M_6 = \bar{A} + \bar{B} + C$
1	1	1	$m_7 = ABC$	$M_7 = \bar{A} + \bar{B} + \bar{C}$

Hình 2.4.2.1. Minterm và maxterm của hàm 3 biến

Trong một minterm và maxterm có mặt tất cả các biến số có trong tổ hợp biến của hàm, các biến số này chỉ xuất hiện một lần dưới dạng trực tiếp hoặc dạng đảo. Hàm logic có thể được biểu diễn dưới dạng là tổng các minterm hoặc tích các maxterm.

– ***Các tính chất của maxterm và minterm:***

- Hai minterm và maxterm của số hạng có cùng chỉ số là phủ định của nhau.

Ví dụ: $m_0 = \overline{M_0}$

- Tổng logic của tất cả các minterm = 1.
- Tích logic của tất cả các maxterm = 0.
- Tích hai minterm khác nhau bất kỳ = 0.
- Tổng hai maxterm khác nhau bất kỳ = 1.

– Phương pháp biểu diễn:

Biểu diễn hàm logic bằng các phương trình logic cho thấy rõ mối quan hệ giữa hàm và biến thông qua các phép toán logic cơ bản là phương pháp biểu diễn thích hợp trong mọi trường hợp kể cả các quan hệ logic phức tạp, hàm có nhiều biến. Dùng phương trình logic biểu diễn hàm sẽ đơn giản gọn ghẽ hơn là dùng bảng chân lý và rất tiện để thực hiện các phép toán logic và tối thiểu hoá hàm bằng phương pháp đại số.

Phương trình logic có thể được xác lập theo các cách sau:

Cách 1: Biểu diễn hàm dưới dạng tổng các tích:

Phương trình: $F = \sum_{0}^{2^n-1} f_i m_i$ (f_i là giá trị của hàm tương ứng với tổ hợp thứ i)

Như vậy, ta chỉ lấy tổng các minterm tương ứng với $f_i = 1$.

Cách 2: Biểu diễn hàm dưới dạng tích các tổng:

Phương trình: $F = \Pi(f_i + M_i)$ (f_i là giá trị của hàm tương ứng với tổ hợp thứ i)

Như vậy, ta chỉ lấy tích của các maxterm tương ứng với $f_i = 0$.

Ví dụ 2.4.2.2. Một hàm ba biến có bảng trạng thái như trên hình 2.4.2.2:

i	A	B	C	F	Minterm	Maxterm
0	0	0	0	1	m_0	M_0
1	0	0	1	0	m_1	M_1
2	0	1	0	0	m_2	M_2
3	0	1	1	1	m_3	M_3
4	1	0	0	1	m_4	M_4
5	1	0	1	0	m_5	M_5
6	1	1	0	0	m_6	M_6
7	1	1	1	1	m_7	M_7

Hình 2.4.2.2. Bảng trạng thái

Ta có thể xác định hàm logic theo hai cách nói trên:

Cách 1: Lấy tổng các minterm ứng với $f_i = 1$ ta được:

$F = m_0 + m_3 + m_4 + m_7 = \overline{A}\overline{B}\overline{C} + \overline{A}BC + A\overline{B}\overline{C} + ABC$

Ký hiệu: $F(A,B,C) = \sum(0,3,4,7)$, trong đó: 0,3,4,7 là các giá trị thập phân tương ứng với các tổ hợp nhị phân mà hàm có giá trị bằng 1.

Cách 2: Lấy tích các maxterm ứng với $f_i = 0$ ta được:

$F = M_1.M_2.M_5.M6 = (A + B + C)(A + \overline{B} + C)(\overline{A} + B + \overline{C})(\overline{A} + \overline{B} + C)$

Ký hiệu: $F(A,B,C) = \Pi(1,2,5,6)$, trong đó: 1,2,5,6 là các giá trị thập phân tương ứng với các tổ hợp nhị phân mà hàm có giá trị bằng 0.

Hàm logic F xác định theo hai cách trên là như nhau.

$F(A,B,C) = \sum(0,3,4,7) = \Pi(1,2,5,6)$

Hay: $\overline{A}\overline{B}\overline{C} + \overline{A}BC + A\overline{B}\overline{C} + ABC = (A + B + C)(A + \overline{B} + C)(\overline{A} + B + \overline{C})(\overline{A} + \overline{B} + C)$

Ví dụ 2.4.2.3. Cho bảng trạng thái của hàm không xác định đầy đủ có ba biến như trên hình 2.4.2.3.

i	A	B	C	F	Minterm	Maxterm
0	0	0	0	X	m_0	M_0
1	0	0	1	1	m_1	M_1
2	0	1	0	0	m_2	M_2
3	0	1	1	1	m_3	M_3
4	1	0	0	1	m_4	M_4
5	1	0	1	0	m_5	M_5
6	1	1	0	0	m_6	M_6
7	1	1	1	x	m_7	M_7

Hình 2.4.2.3. Bảng trạng thái

Ta có: $F(A,B,C) = \sum_m(1,3,4)$ với N = 0,7 hoặc $F(A,B,C) = \Pi(2,5,6)$ với N = 0,7.

Ở đây, N = 0,7 để chỉ rằng các tổ hợp ứng với các giá trị thập phân đó hàm có giá trị không xác định.

2.4.3. Biểu diễn bằng bảng Karnaugh

Khi một hàm logic có số lượng biến tương đối nhỏ ($k \le 6$) người ta thường biểu diễn chúng dưới dạng một bảng gọi là bảng Karnaugh. Theo phương pháp này một hàm có n biến được biểu diễn trên một bảng gồm 2^n ô vuông. Mỗi ô vuông tương ứng với 1 hàng trong bảng trạng thái. Lưu ý rằng các tổ hợp biến ở đây được xếp theo thứ tự của mã Gray tức là hai ô liền kề các minterm chỉ khác nhau có một bit.

Trong các ô của bảng Karnaugh ghi giá trị của hàm tương ứng.

Lưu ý: các tổ hợp biến hàm có giá trị 0 thì có thể bỏ trống hoặc ghi 0.

Ví dụ 2.4.3.1. Trên bảng 2.4.3.1 là bảng Karnaugh của một số hàm logic có 2,3,4,5,6 biến.

a) $F(A,B) = \overline{A}B + A\overline{B}$

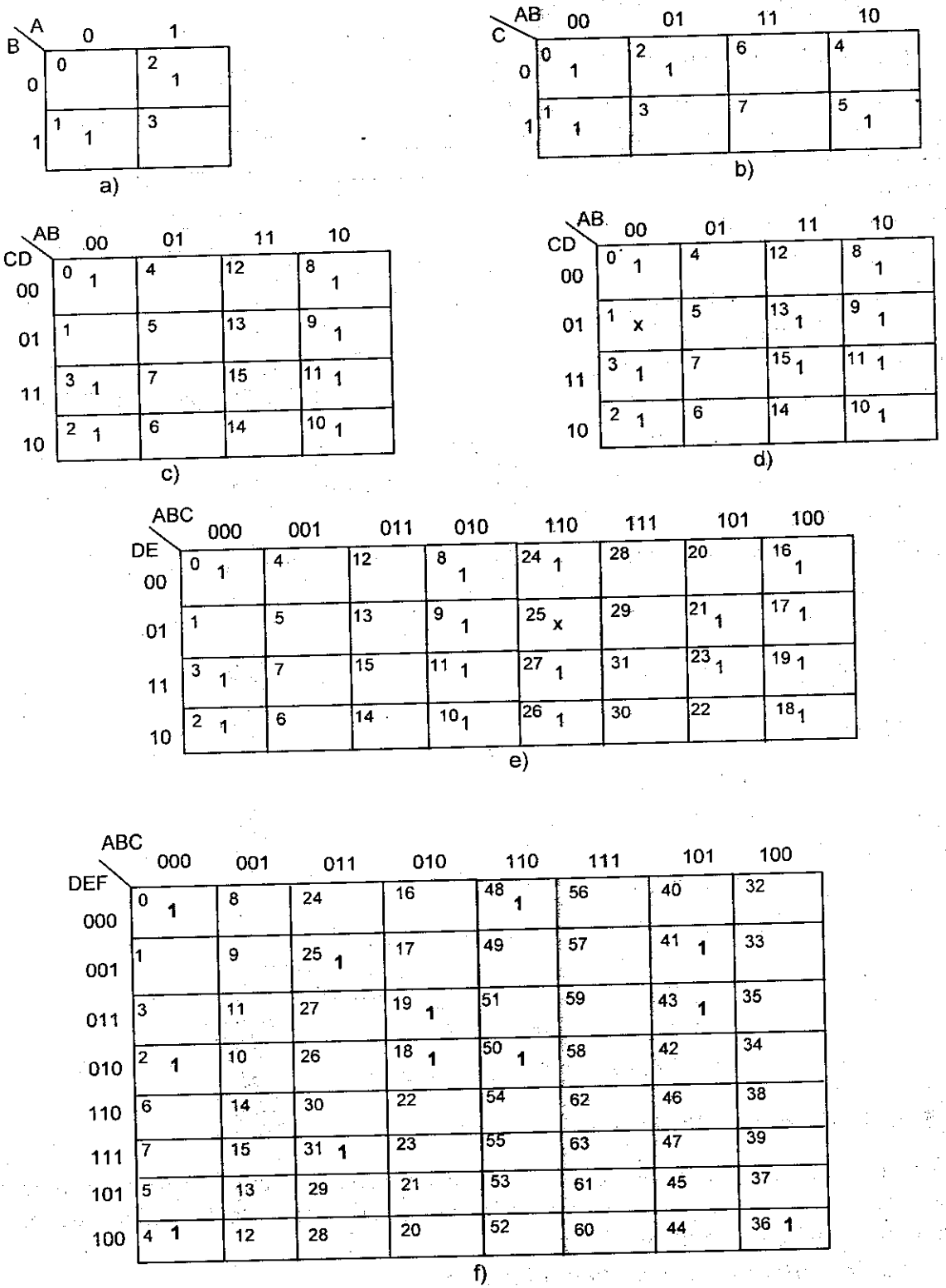
b) $F(A,B,C) = \sum(0,1,2,5)$

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

d) $F(A,B,C,D) = \sum(0,2,3,8,9,10,11,13,15)$ với N = 1

e) $F(A,B,C,D,E) = \sum_m(0,2,3,8,9,10,11,16,17,18,19,21,23,24,26,27)$ với N = 25

f) $F(A,B,C,D,E,F) = \sum_m(0,2,4,18,19,25,31,36,41,43,48,50)$



Hình 2.4.3.1. Bảng Karnaugh của một số hàm có 2, 3, 4, 5, 6 biến

2.4.4. Biểu diễn bằng sơ đồ logic

– Cách vẽ sơ đồ logic của hàm logic:

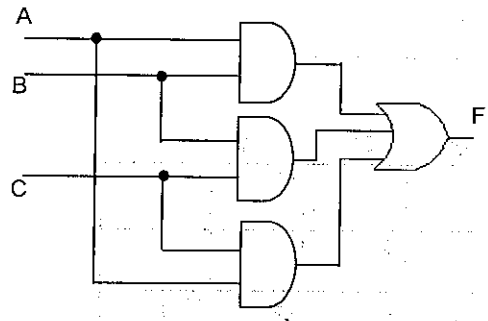
Ta dùng ký hiệu logic của mạch điện tử thay thế phép tính logic có trong biểu thức hàm logic thì được sơ đồ logic của hàm.

Ví dụ 2.4.4.1. Cho hàm

$F = AB + BC + AC$

Vẽ sơ đồ logic của hàm.

Giải: Sơ đồ logic như hình 2.4.4.1. Thay phép toán OR bằng ký hiệu OR và phép toán AND bằng ký hiệu AND.



Hình 2.4.4.1. Sơ đồ logic (ví dụ 2.4.4.1)

– Cách xác định biểu thức từ sơ đồ logic:

Trên sơ đồ logic, từ đầu vào đến đầu ra, viết biểu thức hàm đầu ra của từng cấp, cuối cùng được biểu thức hàm logic toàn sơ đồ.

Ví dụ 2.4.4.2. Cho sơ đồ logic như hình 2.4.4.2, hãy viết biểu thức hàm logic của sơ đồ.

Giải: Ta có:

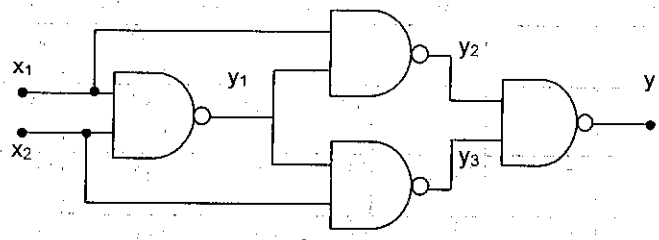
$$y_1 = \overline{x_1.x_2} = (\overline{x_1} + \overline{x_2})$$

$$y_2 = \overline{x_1.y_1}$$

$$y_3 = \overline{x_2.y_1}$$

$$y = \overline{y_2.y_3} = \overline{\overline{x_1.y_1}.\overline{x_2.y_1}} = x_1.y_1 + x_2y_1 = y_1(x_1 + x_2)$$

$$= (\overline{x_1} + \overline{x_2})(x_1 + x_2) = \overline{x_1}x_2 + x_1\overline{x_2}$$



Hình 2.4.4.2. Sơ đồ logic (ví dụ 2.4.4.2)

2.4.5. Biểu diễn hàm logic dưới dạng khối

Trong phần trước chúng ta đã nghiên cứu cách biểu diễn hình học các hàm logic n biến bằng bảng Karnaugh, trong phần này chúng ta sẽ tiến hành nghiên cứu một phương pháp biểu diễn hình học khác.

a) Biểu diễn hình học dưới dạng khối n chiều

Nội dung của phương pháp này là nhờ phép ánh xạ các hàm logic n biến lên trên một khối n chiều (ký hiệu là khối n).

Để làm phép ánh xạ của hàm logic n biến $f(x_1, x_2,\ldots, x_n)$ lên khối n chiều chúng ta thực hiện phép tương ứng giữa các tổ hợp biến (tích các biến) của hàm logic dưới dạng chuẩn tắc tuyển (tổng các tích) và các đỉnh của siêu khối n chiều.

Trên khối n chiều, chúng ta định nghĩa hệ tọa độ với các tọa độ $(e_1, e_2,\ldots, e_n)$, ở đây $e_i = 1$ hoặc 0.

Sau đó chúng ta thực hiện phép tương ứng giữa các tổ hợp biến là:

$x_1^{e_1} x_2^{e_2} \ldots x_n^{e_n}$ với các đỉnh của khối n chiều $(e_1, e_2,\ldots, e_n)$.

Ở đây:

Nếu $e_i = 0$ thì $x_i^{e_i} = \overline{x_i}$

Nếu $e_i = 1$ thì $x_i^{e_i} = x_i$

Hàm logic $f(x_1, x_2,\ldots,x_n)$ sẽ lấy các giá trị 1 hoặc 0 trên các tổ hợp biến của nó.

Nếu lấy giá trị 1 ta ký hiệu là f^1.

Nếu lấy giá trị 0 ta ký hiệu là f^0.

Ví dụ 2.4.5.1. Cho hàm logic 3 biến x_1, x_2, x_3 sau đây:

$$F(x_1, x_2, x_3) = \overline{x_1} x_2 x_3 + x_1 \overline{x_2} x_3 + \overline{x_1}\ \overline{x_2}\ \overline{x_3} + x_1 x_2 x_3$$

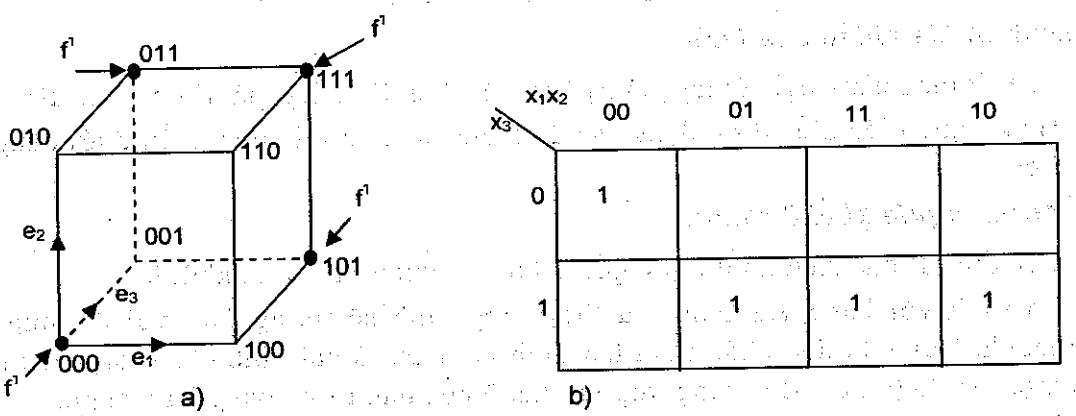
Hãy biểu diễn hàm logic này ở dạng khối 3 chiều và dạng bảng Karnaugh.

Giải:

– Biểu diễn hàm logic dưới dạng khối 3 chiều, hình 2.4.5.1a.

– Biểu diễn hàm bằng bảng Karnaugh, hình 2.4.5.1b.

Quy ước: Tại các đỉnh nếu ta đánh dấu "•" thì sẽ tương ứng với f^1.



Hình 2.4.5.1. Dạng khối 3 chiều và bảng Karnaugh

Ví dụ 2.4.5.2. Cho hàm logic 1 biến x_1: $F(x_1) = \overline{x_1}$

Hãy biểu diễn hàm ở dạng bảng Karnaugh và dạng siêu khối.

Giải:

+ Biểu diễn hàm bằng bảng Karnaugh, hình 2.4.5.2a:

+ Biểu diễn hàm dạng siêu khối, hình 2.4.5.2b:

Hình 2.4.5.2. Bảng Karnaugh và dạng khối của hàm 1 biến

Ví dụ 2.4.5.3. Cho hàm logic 2 biến x_1, x_2:

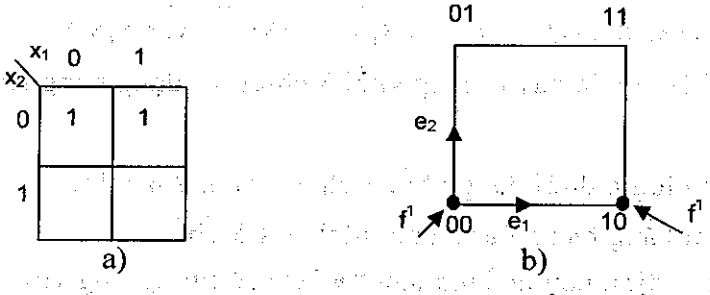
$$F(x_1,x_2)=\overline{x_1}\,\overline{x_2}+x_1\overline{x_2}$$

Hãy biểu diễn hàm ở dạng bảng Karnaugh và dạng siêu khối.

Giải:

– Biểu diễn hàm bằng bảng Karnaugh, hình 2.4.5.3a:

– Biểu diễn hàm dạng siêu khối, hình 2.4.5.3b:



Hình 2.4.5.3. Bảng Karnaugh và dạng khối 2 chiều

b) Định nghĩa chiều của khối

Ta biết rằng hàm logic sẽ lấy giá trị 1 tại các đỉnh f^1 và lấy giá trị 0 tại các đỉnh f^0.

Hàm logic biểu diễn dưới dạng các đỉnh của siêu khối là biểu diễn dưới dạng tổng các tích.

****Định nghĩa khối 0 chiều:***

Các đỉnh f^1 của siêu khối được gọi là khối 0 chiều, ta gọi là khối 0.

So sánh với bảng Karnaugh ta thấy rằng đỉnh sẽ tương ứng với ô trong bảng Karnaugh. Vậy dẫn đến khái niệm hai đỉnh kề nhau là hai đỉnh chỉ khác nhau giá trị của một tọa độ (0 hoặc 1) tương ứng với hai ô kề nhau trong bảng Karnaugh.

+ Hai đỉnh (khối 0) kề nhau ghép với nhau sẽ tạo ra một tọa độ ký hiệu x, tại đó các đỉnh có giá trị khác nhau (0 hoặc 1).

****Định nghĩa khối 1 chiều:***

Khối một chiều là khối có một tọa độ mang giá trị x, ta gọi là khối 1. Vậy hai khối 0 kề nhau sẽ tạo thành một khối 1 chiều, ta gọi là khối 1.

Về mặt biểu diễn hình học khối 1 chính là cạnh (hai điểm xác định một đường thẳng).

Hai khối 1 kề nhau là hai khối chỉ khác nhau giá trị của một tọa độ (0 hoặc 1). Nếu ghép hai khối 1 kề nhau sẽ tạo ra một tọa độ x nữa, tại tọa độ đó các khối 1 có giá trị khác nhau (0 hoặc 1).

Định nghĩa khối 2 chiều:

Khối hai chiều là khối có hai tọa độ mang giá trị x, ta gọi là khối 2. Vậy hai khối 1 kề nhau sẽ tạo thành một khối 2.

Vậy hai khối (n – 1) chiều (có (n – 1) tọa độ mang giá trị x) kề nhau sẽ tạo ra một tọa độ có giá trị x nữa.

Định nghĩa khối n chiều:

Khối n chiều là khối có n tọa độ mang giá trị x, ta gọi là khối n. Vậy hai khối (n – 1) kề nhau sẽ tạo thành một khối n.

Ví dụ 2.4.5.4. Cho hàm logic 3 biến x_1, x_2, x_3 sau đây:

$$F(x_1,x_2,x_3)=\overline{x_1}x_2x_3+x_1\overline{x_2}x_3+\overline{x_1}\,\overline{x_2}\,\overline{x_3}+x_1\overline{x_2}x_3+x_1x_2x_3+x_1\overline{x_2}\,\overline{x_3}$$

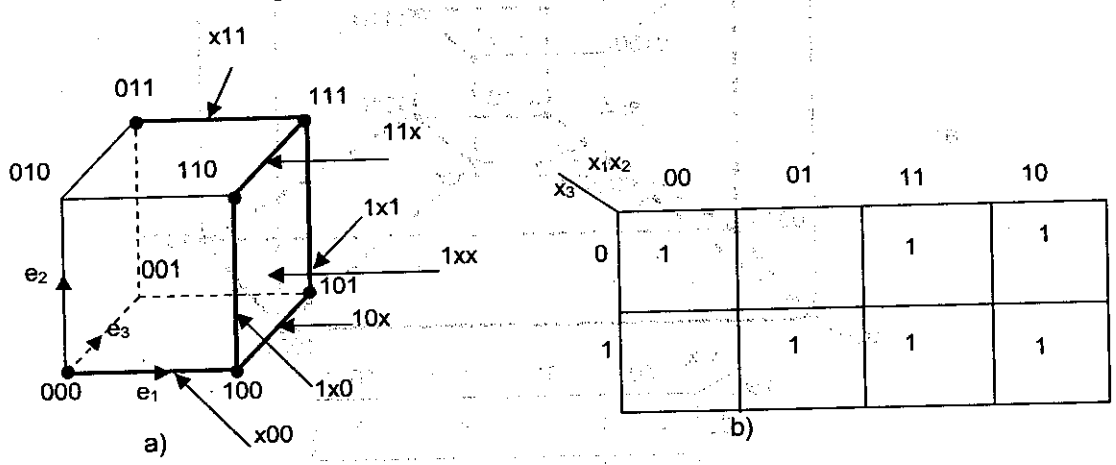
– Hãy biểu diễn hàm logic này ở dạng hình học?

– Hãy tìm các khối 0, khối 1, khối 2?

– Hãy biểu diễn bằng bảng Karnaugh?

Giải:

– Biểu diễn hàm logic dưới dạng hình học, hình 2.4.5.4a:



Hình 2.4.5.4. Dạng khối 3 chiều và bảng Karnaugh

– Các khối: Khối 0: {000, 100, 101, 111, 110, 011}

Khối 1: {x00, 1x0, 10x, 1x1, 11x, x11}

Khối 2: {1xx}

– Biểu diễn bằng bảng Karnaugh, hình 2.4.5.4b.

Nhận xét: Nhìn vào hình khối ta thấy các khối 1: 1x0 và 1x1, 11x và 10x được gọi là hai khối 1 kế cận nhau.

Nhìn vào bảng Karnaugh chúng ta có thể thấy rõ các khối:

– Khối 0 là các tổ hợp tương ứng với 6 ô hàm bằng 1.

– Khối 1 là 6 khả năng có thể kết hợp hai ô kế cận hoặc đối xứng nhau (các ô chỉ khác nhau một biến).

– Khối 2 là khả năng kết hợp 4 ô kế cận với nhau.

Ví dụ 2.4.5.5. Cho hàm logic 4 biến x_1, x_2, x_3, x_4 sau đây:

$$F(x_1,x_2,x_3,x_4) = \bar{x}_1x_2x_3x_4 + x_1\bar{x}_2\bar{x}_3\bar{x}_4 + x_1\bar{x}_2x_3\bar{x}_4 + x_1\bar{x}_2x_3x_4 + x_1x_2\bar{x}_3\bar{x}_4 + x_1x_2\bar{x}_3x_4$$
$$+ x_1\bar{x}_2\bar{x}_3\bar{x}_4 + x_1\bar{x}_2\bar{x}_3x_4 + x_1x_2\bar{x}_3\bar{x}_4 + x_1x_2x_3\bar{x}_4 + x_1\bar{x}_2\bar{x}_3\bar{x}_4$$

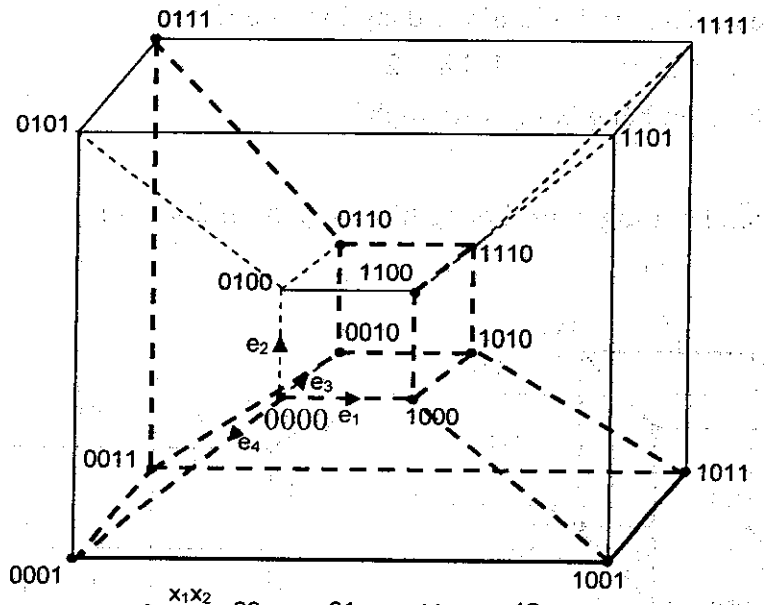
– Hãy biểu diễn hàm logic này ở dạng hình học siêu khối?

– Hãy tìm các khối 0, khối 1, khối 2?

– Hãy biểu diễn bằng bảng Karnaugh?

Giải: Biểu diễn hàm dưới dạng hình học siêu khối, hình 2.4.5.5a:

a)



b)

x_3x_4 \ x_1x_2	00	01	11	10
00	1		1	1
01	1			1
11	1	1	1	1
10	1	1		1

Hình 2.4.5.5. Dạng khối 4 chiều và bảng

– Khối 0: {0000, 1000, 1010, 1110, 1100, 0110, 0001, 1001, 1011, 0111, 0011, 0010}

– Khối 1: {x000, 1x00, 10x0, 1x10, 11x0, x110, x001, 10x1, 000x, 100x, 101x, 011x, 00x1, 001x, 0x10, x010, 00x0, x011, 0x11}

– Khối 2: {1xx0, 10xx, x00x, xx10, x0x0, 00xx, 0x1x}

– Khối 3: {x0xx}

– Biểu diễn bằng bảng Karnaugh, hình 2.4.5.5b.

Nhận xét: Nhìn vào bảng Karnaugh chúng ta có thể thấy rõ các khối:

– Khối 0 là các tổ hợp tương ứng với 12 ô hàm bằng 1.

– Khối 1 là 19 khả năng có thể kết hợp hai ô kế cận hoặc đối xứng nhau (các ô chỉ khác nhau một biến).

– Khối 2 là 7 khả năng kết hợp 4 ô kế cận hoặc đối xứng với nhau.

– Khối 3 là khả năng kết hợp 8 ô đối xứng nhau ở cột đầu và cột cuối (đó là những ô chỉ khác nhau 1 biến một cách liên tục).

c) Biểu diễn hàm logic dưới dạng phức hợp khối

****Tích trực tiếp (tích Descartes)***

Giả sử tập hợp Z_2 là tập hai phần tử của đại số logic (đại số Boole) 0 và 1.

Vậy tập hợp các phần tử n chiều sẽ là tích trực tiếp của n tập hợp Z_2, ta có thể viết như sau: $\underbrace{Z_2.Z_2...Z_2}_{n} = Z_2^n$

Ví dụ 2.4.5.6. Cho tập hợp Z_2 : $Z_2 = \{0,1\}$

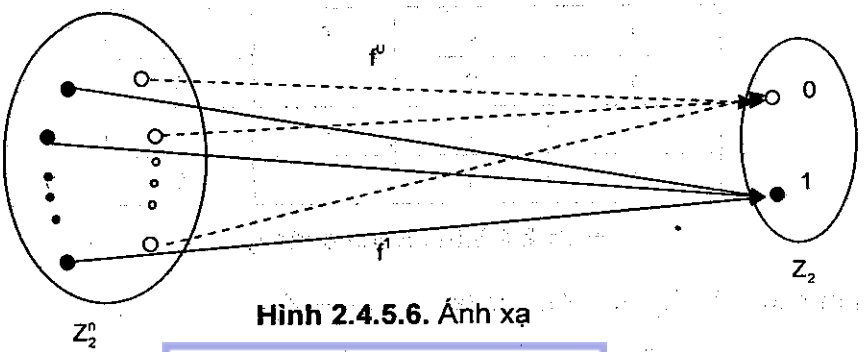
Hãy tìm $Z_2^2 = ?$ và $Z_2^3 = ?$

Giải:

$Z_2^2 = Z_2.Z_2 = \{(00),(01),(10),(11)\}$

$Z_2^3 = Z_2.Z_2.Z_2 = \{(000),(010),(100),(110),(001),(011),(101),(111)\}$

Nhận xét: Vậy ta có thể viết theo phép ánh xạ như sau: $Z_2^n \xrightarrow{f} Z_2$, hình 2.4.5.6



Hình 2.4.5.6. Ánh xạ

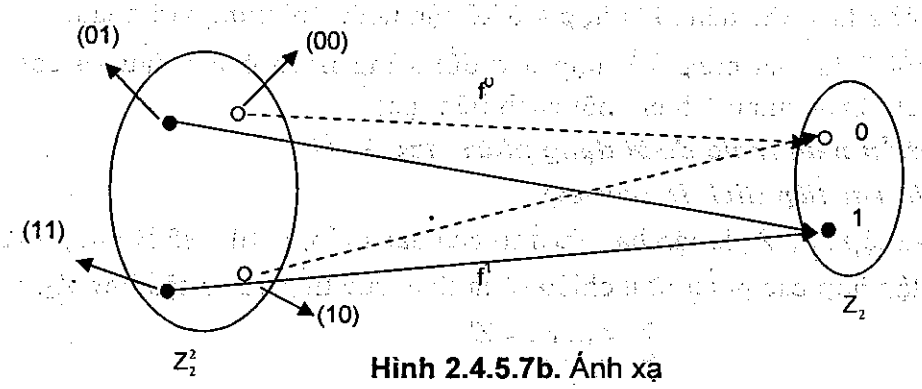
Ví dụ 2.4.5.7. Cho hàm hai biến $F(x_1, x_2)$ có bảng chân lý như trên hình:

X_1	X_2	F
0	0	0
0	1	1
1	0	0
1	1	1

Hình 2.4.5.7a. Bảng chân lý

Hãy tìm ánh xạ của Z_2^2 vào Z_2, tức là: $Z_2^2 \xrightarrow{f} Z_2$

Giải: Ánh xạ của Z_2^2 vào Z_2 biểu diễn trên hình 2.4.5.7b.



Hình 2.4.5.7b. Ánh xạ

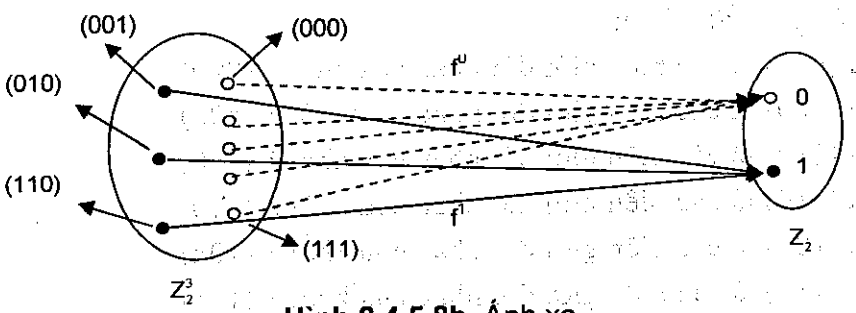
Ví dụ 2.4.5.8. Cho hàm ba biến $F(x_1, x_2, x_3)$ có bảng chân lý cho trên hình 2.4.5.8a:

X_1	X_2	X_3	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Hình 2.4.5.8a. Bảng chân lý

Hãy tìm ánh xạ của Z_2^2 vào Z_2, tức là: $Z_2^2 \xrightarrow{f} Z_2$

Giải: Ánh xạ của Z_2^2 vào Z_2 biểu diễn trên hình 2.4.5.8b.



Hình 2.4.5.8b. Ánh xạ

Nhận xét: Ta có thể dùng đại số tập hợp để nghiên cứu đại số logic.

Định nghĩa biên giới và tọa độ:

Ta nói rằng hai khối 0 chiều kề nhau sẽ tạo ra một khối 1, vậy ta gọi hai khối 0 này là các biên giới đối nhau của khối 1 đó. Tọa độ lấy giá trị x gọi là tọa độ tự do, các tọa độ lấy giá trị 0 hoặc 1 gọi là các tọa độ ràng buộc.

Hai khối 1 chiều kề nhau sẽ tạo thành một khối 2 chiều và ta nói rằng hai khối 1 chiều này là hai biên giới đối nhau của khối 2 chiều.

Chú ý: Hai khối 1 chiều có 1 đỉnh chung không phải là biên giới đối nhau của khối 2 chiều.

Hai khối (r – 1) chiều kề nhau sẽ tạo thành 1 khối r chiều và ta nói rằng 2 khối (r – 1) chiều này là các biên giới đối nhau của khối r chiều.

Hay nói cách khác một khối r chiều sẽ tạo ra hai khối (r – 1) chiều kề nhau tức là tạo ra hai biên giới đối nhau.

Ví dụ 2.4.5.9. Cho hàm logic 3 biến x_1, x_2, x_3 sau đây:

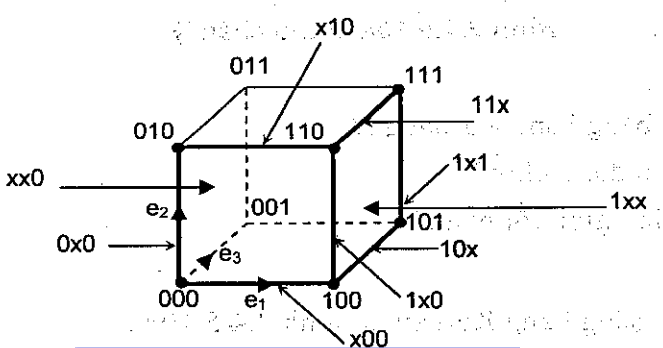
$$F(x_1,x_2,x_3)=\overline{x_1}x_2\overline{x_3}+x_1\overline{x_2}x_3+\overline{x_1}\overline{x_2}x_3+x_1\overline{x_2}x_3+x_1x_2x_3+x_1x_2\overline{x_3}$$

– Hãy biểu diễn hàm logic này ở dạng khối?

– Hãy tìm các biên giới đối nhau?

– Hãy biểu diễn bằng bảng Karnaugh?

Giải: Biểu diễn hàm logic dưới dạng khối, hình 2.4.5.9a:



Hình 2.4.5.9a. Biểu diễn hàm dưới dạng khối

– Các biến giới đối nhau:

(000) và (010) là các biên giới đối nhau của khối 1 (0x0)

(000) và (100) là các biên giới đối nhau của khối 1 (x00)

(100) và (110) là các biên giới đối nhau của khối 1 (1x0)

(110) và (010) là các biên giới đối nhau của khối 1 (x10)

(100) và (101) là các biên giới đối nhau của khối 1 (10x)

(101) và (111) là các biên giới đối nhau của khối 1 (1x1)

(111) và (110) là các biên giới đối nhau của khối 1 (11x)

(x00) và (x10) là các biên giới đối nhau của khối 2 (xx0)

(0x0) và (1x0) là các biên giới đối nhau của khối 2 (xx0)

(1x0) và (1x1) là các biên giới đối nhau của khối 2 (1xx)

(10x) và (11x) là các biên giới đối nhau của khối 2 (1xx)

– Biểu diễn bằng bảng Karnaugh, hình 2.4.5.9b:

x_3 \ x_1x_2	00	01	11	10
0	1	1	1	1
1			1	1

Hình 2.4.5.9b. Bảng Karnaugh

Ví dụ 2.4.5.10. Cho hàm hai biến $F(x_1, x_2)$ có bảng chân lý cho trên hình 2.4.5.10a:

X_1	X_2	F
0	0	1
0	1	1
1	0	1
1	1	0

Hình 2.4.5.10a. Bảng chân lý

– Hãy biểu diễn bằng bảng Karnaugh?

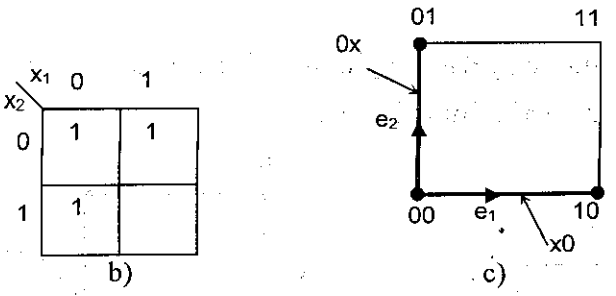
– Hãy biểu diễn ở dạng khối?

– Hãy tìm các biên giới đối nhau?

Giải:

– Biểu diễn hàm bằng bảng Karnaugh, hình 2.4.5.10b :

– Biểu diễn hàm dạng khối, hình 2.4.5.10c:



Hình 2.4.5.10.b,c Bảng Karnaugh và dạng khối

– Các biên giới đối nhau:

(00) và (10) là các biên giới đối nhau của khối 1 (x0)

(00) và (01) là các biên giới đối nhau của khối 1 (0x)

Định nghĩa không gian các khối:

– Định nghĩa không gian các khối 0:

Tập hợp tất cả các khối 0 của hàm logic $f(x_1, x_2,\ldots,x_n)$ được gọi là không gian các khối 0.

Ký hiệu: K^0.

– Định nghĩa không gian các khối 1:

Tập hợp tất cả các khối 1 của hàm logic $f(x_1, x_2,\ldots,x_n)$ được gọi là không gian các khối 1.

Ký hiệu: K^1.

– Định nghĩa không gian các khối r:

Tập hợp tất cả các khối r của hàm logic $f(x_1, x_2,\ldots,x_n)$ được gọi là không gian các khối r.

Ký hiệu: K^r.

Ở đây: $0 \le r \le n$.

Ví dụ 2.4.5.11. Cho hàm hai biến $F(x_1, x_2)$ có bảng chân lý như trên hình 2.4.5.11a:

X_1	X_2	F
0	0	1
0	1	0
1	0	1
1	1	1

Hình 2.4.5.11a. Bảng chân lý

– Hãy biểu diễn bằng bảng Karnaugh?

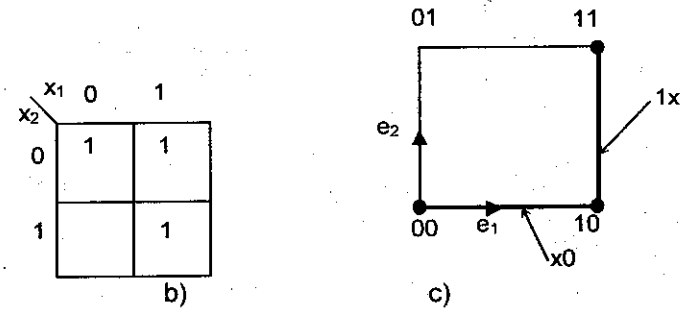
– Hãy biểu diễn ở dạng khối?

– Hãy tìm K^0, K^1, K^2?

Giải:

– Biểu diễn hàm bằng bảng Karnaugh, hình 2.4.5.11b:

– Biểu diễn hàm dạng khối, hình 2.4.5.11c:



Hình 2.4.5.11.b,c Bảng Karnaugh và dạng khối

$$K^0 = \begin{Bmatrix} 00 \\ 10 \\ 11 \end{Bmatrix}, \quad K^1 = \begin{Bmatrix} x0 \\ 1x \end{Bmatrix}, \quad K^2 = \varnothing .$$

Ví dụ 2.4.5.12. Cho hàm logic 3 biến x_1, x_2, x_3 sau đây:

$$F(x_1, x_2, x_3) = x_1\overline{x_2}\,\overline{x_3} + x_1\overline{x_2}x_3 + x_1x_2x_3 + x_1x_2\overline{x_3} + \overline{x_1}x_2x_3$$

– Hãy biểu diễn bằng bảng Karnaugh?

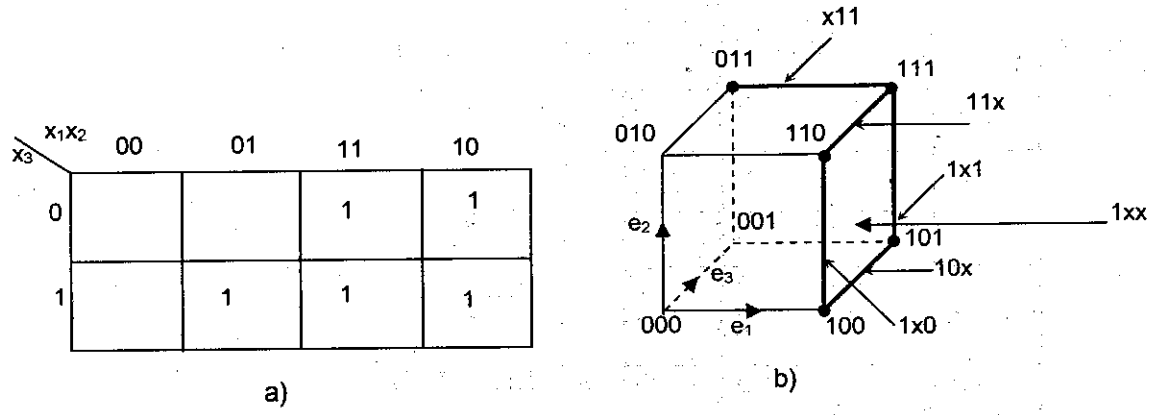
– Hãy biểu diễn hàm logic này ở dạng khối?

– Hãy tìm K^0, K^1, K^2?

Giải:

– Biểu diễn bằng bảng Karnaugh, hình 2.4.5.12a.

– Biểu diễn hàm logic dưới dạng khối, hình 2.4.5.12b.



Hình 2.4.5.12. Bảng Karnaugh và dạng khối

$$K^0 = \begin{Bmatrix} 100 \\ 101 \\ 111 \\ 110 \\ 011 \end{Bmatrix}, \quad K^1 = \begin{Bmatrix} 1x0 \\ 10x \\ 1x1 \\ 11x \\ x11 \end{Bmatrix}, \quad K^2 = \{1xx\}, \quad K^3 = \varnothing.$$

Định nghĩa phức hợp khối:

– Ký hiệu phức hợp khối của hàm logic n biến $f(x_1, x_2,\ldots,x_n)$ được ký hiệu là K(f).

– Định nghĩa: Phức hợp khối K(f) của hàm logic n biến là hợp của không gian các khối $K^0, K^1,\ldots,K^n$ được viết như sau:

$$K(f) = K^0 \cup K^1 \cup \ldots \cup K^n = \bigcup_{i=0}^{n} K^i$$

Ví dụ 2.4.5.13. Cho hàm logic 2 biến x_1, x_2: $F(x_1, x_2) = \overline{x_1}\ \overline{x_2} + \overline{x_1}x_2 + x_1x_2$

– Hãy biểu diễn hàm bằng bảng Karnaugh.

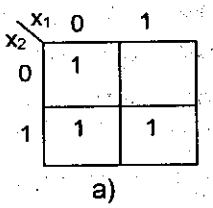
– Hãy biểu diễn hàm ở dạng khối.

– Hãy tìm K(f).

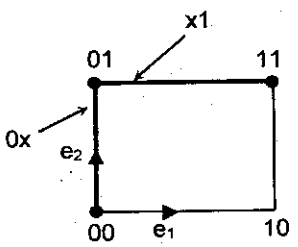
Giải:

– Biểu diễn hàm bằng bảng Karnaugh, hình 2.4.5.13a.

– Biểu diễn hàm dạng khối hình 2.4.5.13b.



a)



b)

Hình 2.4.5.13. Bảng Karnaugh và dạng khối

$$K^0 = \begin{Bmatrix} 00 \\ 01 \\ 11 \end{Bmatrix}, \quad K^1 = \begin{Bmatrix} 0x \\ x1 \end{Bmatrix}, \quad K^2 = \varnothing.$$

Vậy: $K(f) = K^0 \cup K^1 = \begin{Bmatrix} 00 \\ 01 \\ 11 \\ 0x \\ x1 \end{Bmatrix}$

Ví dụ 2.4.5.14. Cho hàm logic 3 biến x_1, x_2, x_3 sau đây:

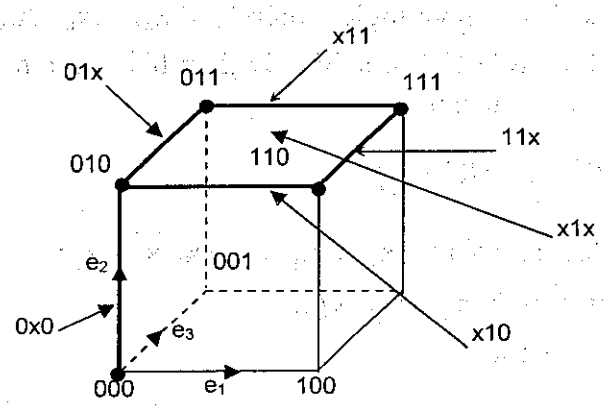
$$F(x_1,x_2,x_3)=\overline{x_1}\,\overline{x_2}\,\overline{x_3}+\overline{x_1}x_2x_3+x_1\overline{x_2}\,\overline{x_3}+x_1x_2\overline{x_3}+x_1x_2x_3$$

– Hãy biểu diễn hàm logic này ở dạng khối?

– Hãy tìm K^0, K^1, K^2 và K(f)?

Giải:

– Biểu diễn hàm logic dưới dạng khối, hình 2.4.5.14:



Hình 2.4.5.14. Biểu diễn hàm dưới dạng khối

– $K^0=\begin{Bmatrix}000\\010\\011\\110\\111\end{Bmatrix}$, $K^1=\begin{Bmatrix}0x0\\01x\\x11\\11x\\x10\end{Bmatrix}$, $K^2=\{x1x\}$, $K^3=\varnothing$.

Vậy: $K(f)=K^0\cup K^1\cup K2=\begin{Bmatrix}000 & 0x0 & x1x\\ 010 & 01x & \\ 011 & x11 & \\ 110 & 11x & \\ 111 & x10 & \end{Bmatrix}$

d) Các thuật toán tách biên giới và hợp biên giới

****Thuật toán tách biên giới:***

– Mục đích: Chúng ta sử dụng thuật toán tách biên giới để tìm hai khối kề nhau tức là hai biên giới đối nhau từ một khối có kích thước lớn hơn.

Giả sử ta ký hiệu một khối r chiều như sau:

$C^r=(a_1, a_2,\ldots, a_n)$

a_i là tọa độ, a_i bằng 0, 1 hoặc x.

Khối r chiều sẽ có r tọa độ lấy giá trị x.

Vậy hai biên giới đối nhau của khối r chiều C^r sẽ được tìm nhờ toán tử tách biên giới thứ i.

– Định nghĩa: Toán tử tách biên giới thứ i được định nghĩa như sau

$$\partial_i^p(a_1,a_2,...,a_n) = \begin{cases}(a_1, a_2,...,a_{i-1}, p, a_{i+1},...,a_n) & a_i = x \\ \varnothing & a_i \neq x\end{cases}$$

Ở đây p = 0 hoặc 1.

Ví dụ 2.4.5.15. Cho khối 1 sau đây: $C^1 = (x1)$

Hãy dùng thuật toán tách biên giới ∂_i^p để tìm các biên giới đối nhau của khối C^1 này.

Giải: Ở đây có hai tọa độ a_1 và a_2: $a_1 = x$ và $a_2 = 1$

Xét i = 1: $\partial_1^p(x1) = (p1)$ do $a_1 = a_i = x$

Tức là: $\partial_1^0(x1) = (01)$

$\partial_1^1(x1) = (11)$

Xét i = 2: $\partial_2^p(x1) = \varnothing$ do $a_2 = a_i = 1 \neq x$

Ví dụ 2.4.5.16. Cho khối 2 sau đây: $C^2 = (1x0x)$

Hãy dùng thuật toán tách biên giới ∂_i^p để tìm các biên giới đối nhau của khối C^2 này.

Giải: Ở đây có 4 tọa độ: $a_1 = 1$, $a_2 = x$, $a_3 = 0$, $a_4 = x$.

$\partial_1^p(1x0x) = \varnothing$ do $a_1 = 1 \neq x$.

$\partial_2^p(1x0x) = (1p0x)$ do $a_2 = x$.

$\Rightarrow \partial_2^p(1x0x) = (100x)$

$\partial_2^p(1x0x) = (110x)$

$\partial_3^p(1x0x) = \varnothing$ do $a_3 = 0 \neq x$.

$\partial_4^p(1x0x) = (1x0p)$ do $a_4 = x$.

$\Rightarrow \partial_4^p(1x0x) = (1x00)$

$\partial_4^p(1x0p) = (1x01)$

Ví dụ 2.4.5.17. Cho khối 3 sau đây: $C^3 = (1xx01x)$

Hãy dùng thuật toán tách biên giới ∂_i^p để tìm các biên giới đối nhau của khối C^3 này.

Giải: Ở đây có 6 tọa độ:$a_1 = 1$, $a_2 = x$, $a_3 = x$, $a_4 = 0$, $a_5 = 1$, $a_6 = x$.

$\partial_1^p(1xx01x) = \varnothing$ do $a_1 = 1 \neq x$.

$\partial_2^p(1xx01x) = (1px01x)$ do $a_2 = x$.

$\Rightarrow \partial_2^p(1xx01x) = (10x01x)$

$\partial_2^p(1xx01x) = (11x01x)$

$\partial_3^p(1xx01x) = (1xp01x)$ do $a_3 = x$.

$\Rightarrow \partial_3^p(1xx01x) = (1x001x)$

$\partial_3^p(1xx01x) = (1x101x)$

$\partial_4^p(1xx01x) = \varnothing$ do $a_4 = 0 \neq x$.

$\partial_5^p(1xx01x) = \varnothing$ do $a_5 = 1 \neq x$.

$C_3^1(1xx01x) = (1xx01p)$ do $a_6 = x$.

$\Rightarrow \partial_6^p(1xx01x) = (1xx010)$

$\partial_6^p(1xx01x) = (1xx011)$

Nhận xét: Từ khối r ta dùng thuật toán tách biên giới ∂_i^p để nhận được các khối (r − 1) và cứ tiếp tục dùng thuật toán cho đến khi nhận được các khối 0.

Thuật toán hợp biên giới:

– Mục đích: Chúng ta có thể sử dụng thuật toán hợp biên giới để tìm khối (r + 1) chiều từ các khối r chiều.

Giả sử ta ký hiệu một khối r chiều như sau:

$C^r = (a_1, a_2, \ldots, a_n)$

a_i là tọa độ, a_i bằng 0, 1 hoặc x.

Khối r chiều sẽ có r tọa độ lấy giá trị x.

Vậy khối (r + 1) chiều C^{r+1} có thể tìm được nhờ thuật toán hợp biên giới thứ i.

– *Định nghĩa:* Thuật toán hợp biên giới thứ i được định nghĩa như sau:

$$\delta_i(a_1, a_2, \ldots, a_n) = \begin{cases} (a_1, \ldots, a_{i-1}, x, a_{i+1}, \ldots, a_n) = C^{r+1} & \text{nếu } a_i \neq x \text{ và } C^{r+1} \subset K(f) \\ \varnothing & \text{nếu } a_i = x \text{ hoặc } C^{r+1} \not\subset K(f) \end{cases}$$

Ví dụ 2.4.5.18. Cho hàm logic 2 biến x_1, x_2: $F(x_1, x_2) = \overline{x_1 x_2} + \overline{x_1} x_2 + x_1 x_2$

– Hãy biểu diễn hàm ở dạng khối.

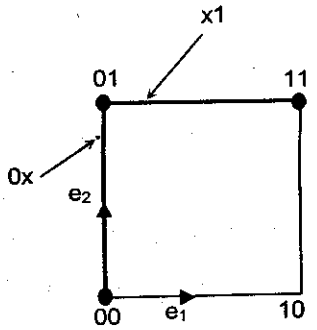
– Hãy tìm K(f).

– Hãy tác động toán tử δ_i vào các khối của K(f) để tìm các khối có kích thước lớn hơn.

Giải:

– Biểu diễn hàm dạng khối, hình 2.6.5.18:

$$K^0 = \begin{Bmatrix} 00 \\ 01 \\ 11 \end{Bmatrix}, \quad K^1 = \begin{Bmatrix} 0x \\ x1 \end{Bmatrix}, \quad K^2 = \varnothing, \quad K(f) = \begin{Bmatrix} 00 \\ 01 \\ 11 \\ 0x \\ x1 \end{Bmatrix}$$



Hình 2.4.5.15. Biểu diễn hàm dưới dạng khối

– Ở đây ta có i = 1,2.

Xét: $\delta_i(00)$

– $\delta_1(00) = \varnothing$ vì $a_1 = 0 \neq x$ nhưng $(x0) \not\subset K(f)$.

– $\delta_2(00) = (0x)$ vì $a_2 = 0 \neq x$ và $(0x) \subset K(f)$.

Xét: $\delta_i(01)$

– $\delta_1(01) = (x1)$ vì $a_1 = 0 \neq x$ và $(x1) \subset K(f)$.

– $\delta_2(01) = (0x)$ vì $a_2 = 1 \neq x$ và $(0x) \subset K(f)$.

Xét: $\delta_i(11)$

– $\delta_1(11) = (x1)$ vì $a_1 = 1 \neq x$ và $(x1) \subset K(f)$.

– $\delta_2(11) = \varnothing$ vì $a_2 = 1 \neq x$ nhưng $(x1) \not\subset K(f)$.

Xét: $\delta_i(0x)$

– $\delta_1(0x) = \varnothing$ vì $a_1 = 0 \neq x$ nhưng $(xx) \not\subset K(f)$.

– $\delta_2(0x) = \varnothing$ vì $a_2 = x$

Xét: $\delta_i(x1)$

– $\delta_1(x1) = \varnothing$ vì $a_1 = x$

– $\delta_2(x1) = \varnothing$ vì $a_2 = 1 \neq x$ nhưng $(xx) \not\subset K(f)$.

Nhận xét: Từ khối K^0 ta dùng thuật toán hợp biên giới để tìm được các khối K^1, K^2,...

e) Các phép toán trên phức hợp khối

Nhận xét: Chúng ta biết rằng mỗi một hàm logic sẽ tương ứng với một phức hợp khối vì vậy ta có thể nói rằng:

Hai hàm logic là tương đương nhau nếu và chỉ nếu hai phức hợp khối của chúng là hoàn toàn giống nhau.

Giả sử ta có hai hàm logic sau:

$f(x_1, x_2,\ldots,x_n)$

$g(x_1, x_2,\ldots,x_n)$

Đối với hai hàm logic này tồn tại 3 phép toán logic cơ bản là:

Nhân logic: f.g

Cộng logic: f + g

Phủ định logic: $\bar{f}$ và $\bar{g}$.

Vậy chúng ta cũng có sự tương ứng giữa các phép toán logic trên các hàm logic và các phép toán tập hợp (logic) trên các phức hợp khối của chúng như sau:

+ Phép nhân logic f.g tương ứng với phép giao tập hợp như sau:

$f.g \rightarrow K(f.g) = k(f) \cap k(g)$

+ Phép cộng logic f + g tương ứng với phép hợp tập hợp như sau:

$K(f + g) = K(f) \cup K(g)$

+ Phép phủ định logic $\bar{f}$ và $\bar{g}$ tương ứng với phép bù tập hợp như sau:

$K(\bar{f}) = \overline{K(f)}$

$K(\bar{g}) = \overline{K(g)}$

Ví dụ 2.4.5.19. Cho 2 hàm logic 2 biến x_1, x_2 như sau:

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

$g(x_1, x_2) = \bar{x}_1 x_2 + x_1 x_2$

a) Hãy tính: f.g

f + g

$\bar{f}$ và $\bar{g}$.

b) Hãy tính: K(f) và K(g)

$k(f) \cap k(g)$

$K(f) \cup K(g)$

$K(\bar{f})$, $\overline{K(f)}$

$K(\bar{g})$, $\overline{K(g)}$

Giải:

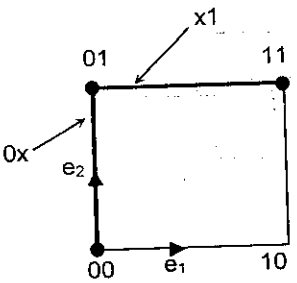
a) $f.g = x_1.x_2$

$f + g = 1$

$\bar{f} = x_1 \bar{x}_2$

$\bar{g} = \bar{x}_1 \bar{x}_2 + \bar{x}_1 x_2$

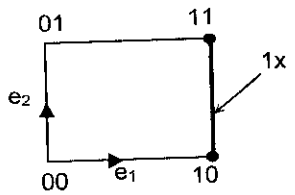
b) Ta biểu diễn hàm f dưới dạng khối như trên hình 2.4.5.16a



$$K^0 = \begin{Bmatrix} 00 \\ 01 \\ 11 \end{Bmatrix}, K^1 = \begin{Bmatrix} 0x \\ x1 \end{Bmatrix}, K(f) = \begin{Bmatrix} 00 \\ 01 \\ 11 \\ 0x \\ x1 \end{Bmatrix}$$

Hình 2.4.5.16a. Dạng khối của hàm f

Ta biểu diễn hàm g dưới dạng khối như trên hình 2.4.5.16b:



$$K^0 = \begin{Bmatrix} 10 \\ 11 \end{Bmatrix}, K^1 = \{1x\},$$

$$K(g) = \begin{Bmatrix} 10 \\ 11 \\ 1x \end{Bmatrix}; \quad k(f) \cap k(g) = \{11\}$$

Hình 2.4.5.16b. Dạng khối của hàm g

$$K(f) \cup K(g) = \begin{Bmatrix} 00 & 0x \\ 01 & x1 \\ 10 & 1x \\ 11 & xx \end{Bmatrix}, K(\bar{f}) = \{10\}, \quad \overline{K(f)} = \{10\},$$

$$K(\bar{g}) = \begin{Bmatrix} 00 \\ 01 \\ 0x \end{Bmatrix}, \quad \overline{K(g)} = \begin{Bmatrix} 00 \\ 01 \\ 0x \end{Bmatrix}$$

2.5. HÀM NOR VÀ HÀM NAND

2.5.1. Hàm NOR (không hoặc: NOT – OR)

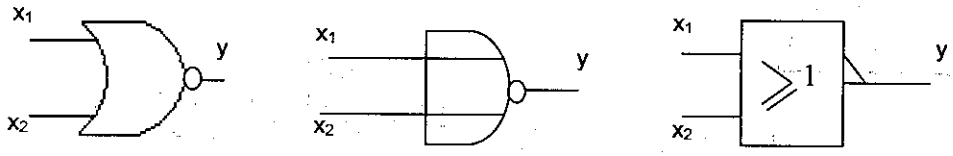
– Hàm logic: $y = \overline{x_1 + x_2}$

– Bảng chân lý cho trên hình 2.5.1.1.

– Ký hiệu logic cho trên hình 2.5.1.2.

x_1	x_2	y
0	0	1
0	1	0
1	0	0
1	1	0

Hình 2.5.1.1. Bảng chân lý của cổng NOR



Hình 2.5.1.2. Ký hiệu logic của cổng NOR

– Trong trường hợp tổng quát nếu n biến ta cũng có:

$$y = \overline{x_1 + x_2 + ... + x_n}$$

2.5.2. Hàm NAND (không và: NOT – AND)

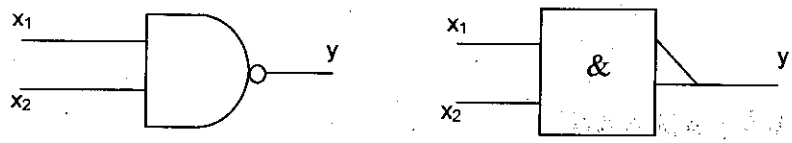
– Hàm logic: $y = \overline{x_1.x_2}$

– Bảng chân lý cho trên hình 2.5.2.1 :

x_1	x_2	y
0	0	1
0	1	1
1	0	1
1	1	0

Hình 2.5.2.1. Bảng chân lý của cổng NAND

– Ký hiệu logic trên hình 2.5.2.2:



Hình 2.5.2.2. Ký hiệu logic của cổng NAND

Tổng quát nếu có n biến ta cũng có:

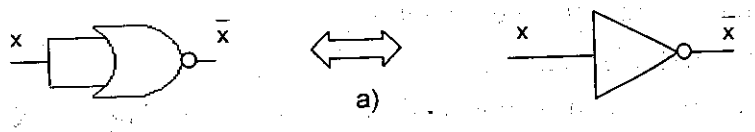
$$y = \overline{x_1.x_2...x_n}$$

2.5.3. Tính đa dụng của cổng NOR và cổng NAND

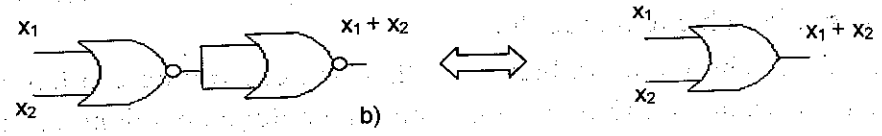
Tất cả các biểu thức Boole đều kết hợp 3 phép toán cơ bản OR, AND và NOT. Do đó, bất kỳ biểu thức nào cũng đều có thể được thực hiện bằng cách dùng cổng OR, AND và NOT. Tuy nhiên, có thể thực hiện biểu thức logic bất kỳ chỉ dùng cổng NOR hoặc NAND mà không cần thêm loại cổng nào khác.

a) Dùng cổng NOR thay cho ba cổng logic cơ bản, hình 2.5.3.1

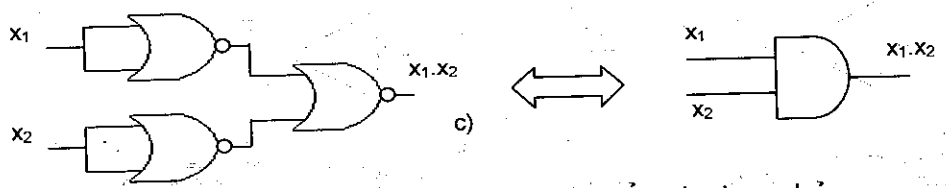
+ $\overline{x + x} = \bar{x}$:



+ $\overline{x_1 + x_2} = x_1 + x_2$:



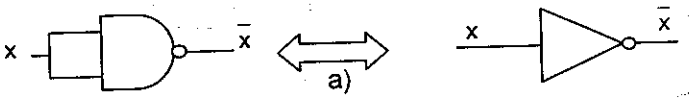
+ $\overline{\bar{x}_1 + \bar{x}_2} = x_1.x_2$:



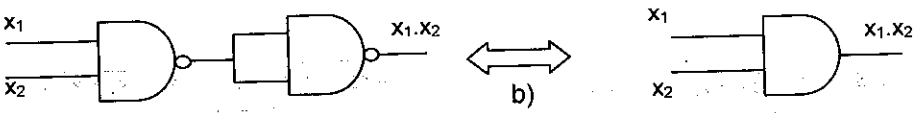
Hình 2.5.3.1. Dùng cổng NOR thay cho ba cổng logic cơ bản
a) Cổng NOT ; b) Cổng OR ; c) Cổng AND

b) Dùng cổng NAND thay cho ba cổng logic cơ bản, hình 2.5.3.2

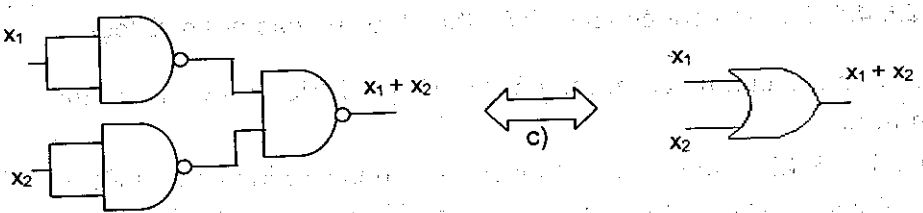
+ $\overline{x.x} = x$:



+ $\overline{x_1.x_2} = x_1 x_2$:



+ $\overline{\bar{x}_1.x_2} = x_1 + x_2$:



Hình 2.5.3.2. Dùng cổng NAND thay cho ba cổng logic cơ bản
a) Cổng NOT ; b) Cổng AND ; c) Cổng OR

2.5.4. Ký hiệu logic thay thế

Trong các phần trước chúng ta đã tìm hiểu 5 loại cổng logic cơ bản và các ký hiệu chuẩn để biểu diễn chúng trên sơ đồ mạch logic.

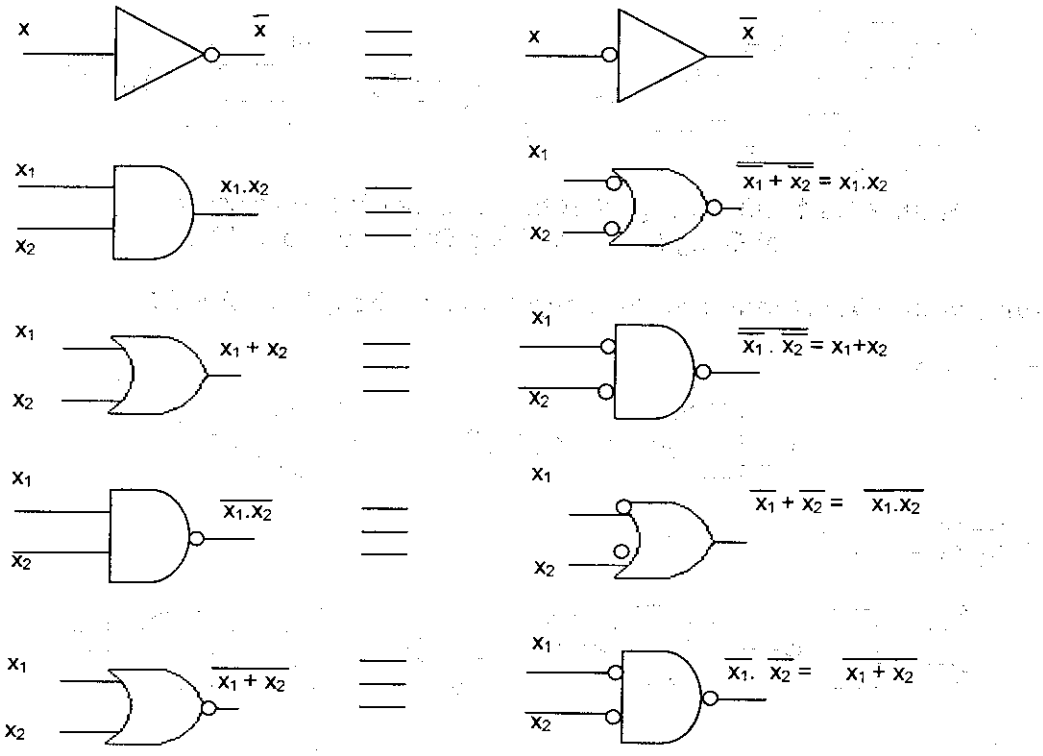
– Khái niệm mức logic tích cực:

+ Nếu không có vòng tròn nhỏ ở đường vào hay đường ra trên ký hiệu mạch logic, đường đó được gọi là tích cực ở mức cao.

+ Nếu có vòng tròn nhỏ ở đường vào hay đường ra trên ký hiệu mạch logic, đường đó được gọi là tích cực ở mức thấp.

Từ ký hiệu đó sẽ cho chúng ta biết trạng thái tích cực ở mức cao hay tích cực ở mức thấp ở đầu vào và đầu ra, từ đó chúng ta giải thích được hoạt động của mạch.

– Các ký hiệu logic thay thế, hình 2.5.4.1:

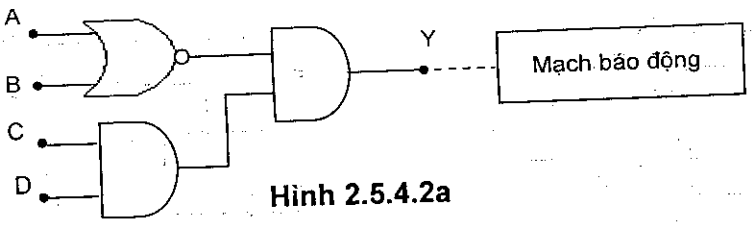


Hình 2.5.4.1. Ký hiệu chuẩn và ký hiệu thay thế của các phần tử logic

Tùy theo chức năng của mạch mà ta có thể dùng ký hiệu thay thế để thấy rõ hoạt động của mạch hơn.

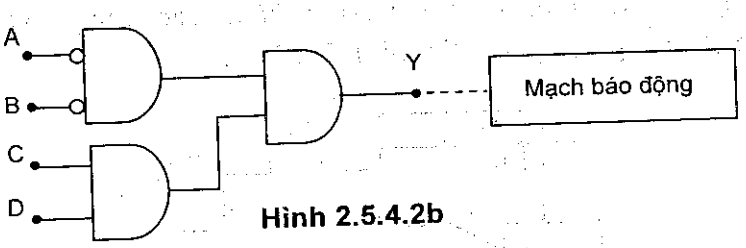
Ví dụ 2.5.4.1. Hình 2.5.4.2a minh họa một mạch logic dùng để kích hoạt một tín hiệu báo động khi đầu ra Y của nó ở mức logic cao. Hãy sửa đổi sơ đồ mạch để nó biểu diễn hoạt động của mạch hiệu quả hơn?

Giải:



Hình 2.5.4.2a

Sơ đồ hình 2.5.4.2a được vẽ lại trên hình 2.5.4.2b. Từ hình 2.5.4.2b ta thấy ngay được đầu ra Y sẽ ở mức cao khi A = B = 0 và C = D = 1.



Hình 2.5.4.2b

2.6. HÀM XOR VÀ HÀM XNOR

2.6.1. Hàm XOR (Exclusive – OR)

Hàm hoặc loại trừ hay còn được gọi là hàm hoặc tuyệt đối, hàm cộng modul 2, hàm không tương đương, hàm khác dấu,...

– Hàm logic: $y = x_1.\overline{x_2} + \overline{x_1}.x_2$

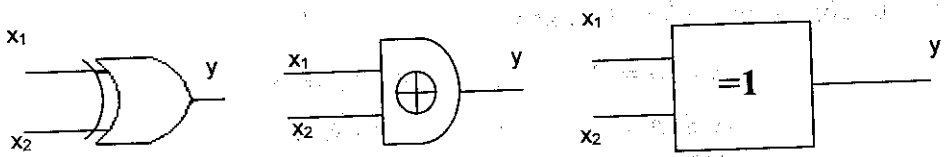
Được viết là: $y = x_1 \oplus x_2$

– Bảng chân lý cho trên hình 2.6.1.1:

– Ký hiệu logic, hình 2.6.1.2:

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

Hình 2.6.1.1. Bảng chân lý của cổng XOR



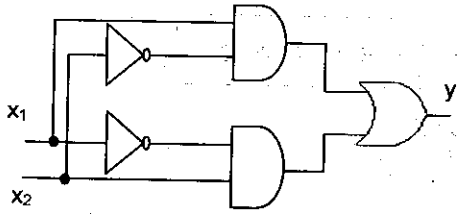
Hình 2.6.1.2. Ký hiệu logic của cổng XOR

– Sơ đồ logic tương ứng được trình bày trên hình 2.6.1.3.

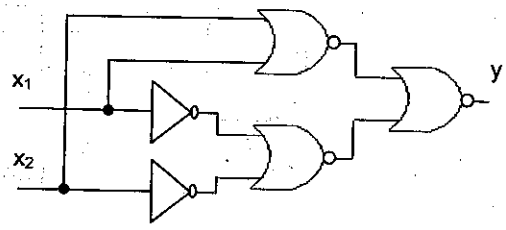
– Ta có thể thiết kế sơ đồ mạch XOR bằng NOT và NOR:

$$y = \overline{x_1.\overline{x_2} + \overline{x_1}.x_2} = \overline{\overline{x_1} + x_2} + \overline{x_1 + \overline{x_2}}$$

Sơ đồ logic của mạch tương ứng với phương trình này được trình bày trên hình 2.6.1.4.

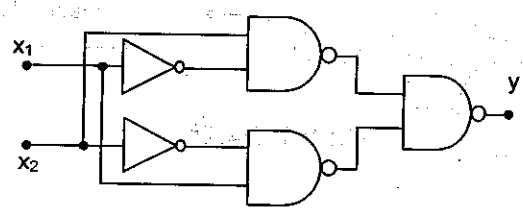


Hình 2.6.1.3. Mạch XOR dùng NOT,AND,OR

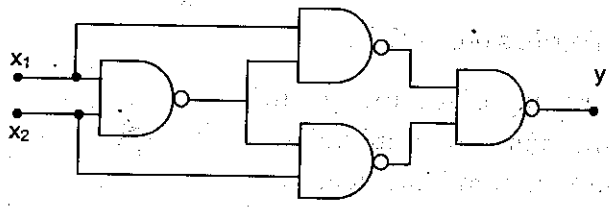


Hình 2.6.1.4. Mạch XOR dùng NOT, NOR

– Ta cũng có thể thiết kế mạch XOR bằng các cổng NOT và NAND (hình 2.6.1.5), hoặc chỉ bằng cổng NAND (hình 2.6.1.6).



Hình 2.6.1.5. Mạch XOR dùng NOT, NAND



Hình 2.6.1.6. Mạch XOR dùng NAND

– Các tính chất của hàm XOR:

+ Tính giao hoán: $A \oplus B = B \oplus A$

+ Tính kết hợp: $A \oplus (B \oplus C) = (A \oplus B) \oplus C$

+ Tính phân phối: $A(B \oplus C) = AB \oplus AC$

Ngoài ra ta còn có một số tính chất sau:

$$A \oplus 0 = A \quad ; \quad A \oplus 1 = \overline{A}$$

$$A \oplus A = 0 \quad ; \quad A \oplus \overline{A} = 1$$

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

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

$$A \oplus B = C \Leftrightarrow A \oplus C = B \Leftrightarrow B \oplus C = A$$

– Hàm XOR nhiều biến:

Dùng tính chất kết hợp ta cũng có thể xây dựng được các mạch XOR nhiều lối vào từ mạch XOR hai lối vào này.

Lấy mạch XOR 3 lối vào làm ví dụ:

+ Hàm logic: $y = x_1 \oplus x_2 \oplus x_3$

+ Bảng chân lý cho trên hình 2.6.1.7:

x_1	0	0	0	0	1	1	1	1
x_2	0	0	1	1	0	0	1	1
x_3	0	1	0	1	0	1	0	1
y	0	1	1	0	1	0	0	1

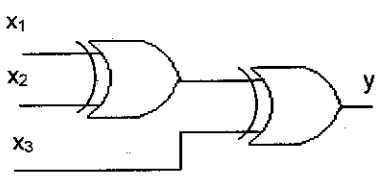
Hình 2.6.1.7. Bảng chân lý của hàm XOR 3 biến

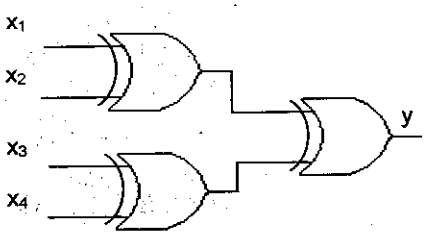
+ Nhận xét bảng chân lý:

- Nếu số mức logic 1 ở lối vào là lẻ → lối ra là mức logic 1.
- Nếu số mức logic 1 ở lối vào là chẵn → lối ra là mức logic 0.

Nguyên tắc này áp dụng cho các mạch XOR nhiều lối vào (n lối vào).

- Ta có thể xây dựng cổng XOR 3 lối vào hoặc 4 lối vào từ các cổng XOR 2 lối vào như hình 2.6.1.8.





Hình 2.6.1.8. Sơ đồ logic của hàm XOR 3 biến

2.6.2. Hàm XNOR (Exclusive – NOR)

Hàm không hoặc loại trừ hay còn được gọi là hàm không hoặc tuyệt đối, hàm tương đương, hàm cùng dấu,...

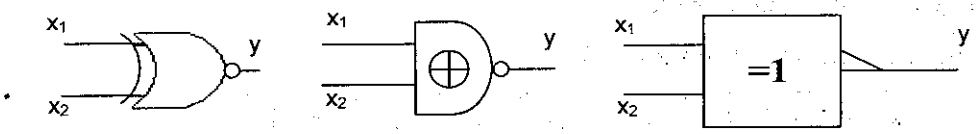
– Hàm logic: $y = x_1x_2 + \overline{x}_1\overline{x}_2 = \overline{x_1 \oplus x_2} = x_1 \sim x_2$

– Bảng chân lý cho trên hình 2.6.2.1:

x_1	x_2	y
0	0	1
0	1	0
1	0	0
1	1	1

Hình 2.6.2.1. Bảng chân lý của hàm XNOR

– Ký hiệu logic, hình 2.6.2.2:



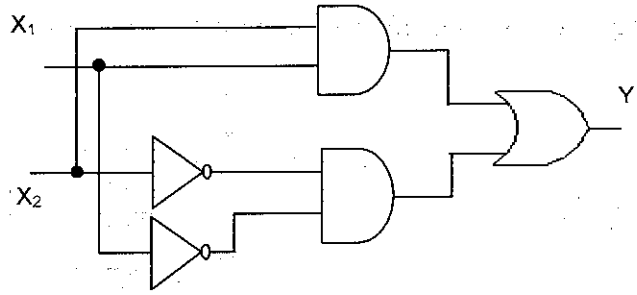
Hình 2.6.2.2. Ký hiệu logic của hàm XNOR

Nhận xét: Lối ra của XNOR 2 lối vào là đảo của XOR 2 lối vào.

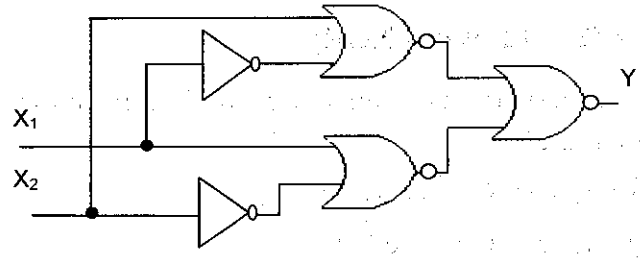
Tạ cũng có thể xây dựng được các cổng XNOR nhiều lối vào bằng cách tương tự như xây dựng XOR nhiều lối vào. Từ các phần tử logic cơ bản AND, OR, NOT hoặc NAND và NOT hay NOR và NOT hoặc chỉ bằng NAND và chỉ bằng NOR ta có thể tạo được các cổng XNOR.

Trên hình 2.6.2.3 giới thiệu sơ đồ logic mạch XNOR hai lối vào được xây dựng từ các phần tử logic cơ bản NOT, AND và OR.

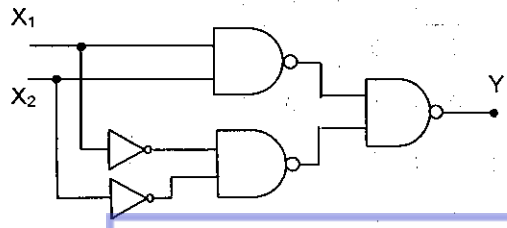
Trên hình 2.6.2.4 là sơ đồ logic mạch XNOR hai lối vào được xây dựng từ các phần tử logic NOT và NOR.



Hình 2.6.2.3. Mạch XNOR xây dựng từ NOT, AND, OR



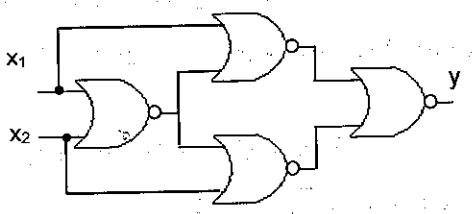
Hình 2.6.2.4. Mạch XNOR xây dựng từ NOT, NOR



Hình 2.6.2.5. Mạch XNOR xây dựng từ NOT, NAND

Trên hình 2.6.2.5 vẽ sơ đồ logic của mạch XNOR 2 lối vào được tạo nên từ cổng NOT và NAND.

Trên hình 2.6.2.6 vẽ sơ đồ logic mạch XNOR 2 lối vào chỉ dùng các cổng NOR.



Hình 2.6.2.6. Mạch XNOR xây dựng từ OR

– Các tính chất của hàm XNOR:

$A \sim 0 = \overline{A}$	$A \sim \overline{A} = 0$	$\overline{A \sim B} = \overline{A} \sim B = A \sim \overline{B}$
$A \sim A = 1$	$A \sim B = B \sim A$	$A \sim B = \overline{A} \sim \overline{B}$
$A \sim 1 = A$	$A \sim (B \sim C) = (A \sim B) \sim C$	$A \sim B = C \Leftrightarrow A \sim C = B \Leftrightarrow B \sim C = A$

2.7. CÁC PHƯƠNG PHÁP TỐI THIỂU HÓA HÀM LOGIC

Trong việc thiết kế các khối chức năng logic, tìm ra được một sơ đồ logic đơn giản đáp ứng đầy đủ các yêu cầu của khối chức năng cần thiết kế, thì yêu cầu hàng đầu của công tác thiết kế các mạch điện tử là tính kinh tế và mạch phải có tính ổn định độ tin cậy cao. Để đảm bảo các yêu cầu này thì sơ đồ logic phải bao gồm số các phần tử logic cơ bản ít nhất, các sơ đồ càng đơn giản càng có độ tin cậy và ổn định cao. Để xây dựng được một sơ đồ như vậy chúng ta phải tìm ra được một phương trình logic tối giản mô tả đúng chức năng logic của mạch điện tử cần thiết kế. Các hàm logic mà ta thường gặp thường không phải là dạng tối giản, nếu ta xây dựng mạch dựa trên phương trình này thì sẽ tốn kém vì phải dùng nhiều phần tử linh kiện logic, sơ đồ càng phức tạp độ ổn định, độ tin cậy càng kém, xác suất hư hỏng càng tăng. Vì thế trước khi xây dựng mạch bao giờ cũng phải tìm cách rút gọn hàm đưa phương trình biểu diễn về dạng tối giản. Phương trình ở dạng tối giản khi các số hạng phải là ít nhất và số biến trong mỗi số hạng cũng phải là ít nhất.

2.7.1. Tối thiểu hoá hàm logic bằng phương pháp đại số

Áp dụng các định luật của đại số logic để đơn giản hàm logic sao cho hàm cuối cùng là tối giản. Vì trong thực tế các biểu thức logic rất đa dạng, từ một hàm logic cũng có thể biểu diễn theo nhiều cách khác nhau nên với phương pháp này khó có thể tìm ra một quy trình tối ưu để tìm ra được một biểu thức logic tối giản một cách nhanh nhất. Tuy nhiên, nếu nắm chắc các định luật của đại số Boole và có kinh nghiệm chúng ta có thể thu được kết quả tốt.

Một số công thức thường dùng:

1) $AB + A\bar{B} = A$; 2) $A + AB = A$

3) $A + \bar{A}B = A + B$; 4) $AB + \bar{A}C + BC = AB + \bar{A}C$

Từ công thức (4) ta có hệ quả: $AB + \bar{A}C + BCD = AB + \bar{A}C$

Chứng minh:

1) $AB + A\bar{B} = A(B + \bar{B}) = A$; 2) $A + AB = A(1 + B) = A$

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

4) $AB + \bar{A}C + BC = AB + \bar{A}C + (A + \bar{A})BC = AB(1 + C) + \bar{A}C(1 + B) = AB + \bar{A}C$

Ví dụ 2.7.1.1. Tối thiểu hoá các hàm logic sau:

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

b) $F = A(BC + \bar{B}\bar{C}) + A(\bar{B}C + B\bar{C}) = A(BC + \bar{B}\bar{C} + \overline{BC + \bar{B}\bar{C}}) = A$

c) $F = AB + \bar{A}C + \bar{B}C = AB + \bar{A}C + BC + \bar{B}C = AB + C(\bar{A} + B + \bar{B}) = AB + C$

Hoặc: $F = AB + \bar{A}C + \bar{B}C = AB + (\bar{A} + \bar{B})C = AB + \overline{AB}C = AB + C$

d) $F = A\bar{B} + B\bar{C} + \bar{B}C + \bar{A}B = A\bar{B} + B\bar{C} + A\bar{C} + \bar{B}C + \bar{A}B$

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

e) $F(A,B,C) = \sum(0,1,5,6,7) = \bar{A}\,\bar{B}\,\bar{C} + \bar{A}\,\bar{B}C + A\bar{B}C + AB\bar{C} + ABC$

$= \bar{A}\,\bar{B}\,(\bar{C} + C) + AC(\bar{B} + B) + AB(\bar{C} + C) = \bar{A}\,\bar{B} + AC + AB$

f) $F(A,B,C,D) = \sum(1,3,4,6,8,9,10)$

$= \bar{A}\,\bar{B}\,\bar{C}D + \bar{A}\,\bar{B}CD + \bar{A}B\bar{C}\,\bar{D} + \bar{A}BCD + A\bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C}D + A\bar{B}C\bar{D}$

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

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

g) $F(A,B,C,D,E) = \Pi(0,1,2,19,20)$

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

$.(\bar{A} + B + C + \bar{D} + \bar{E})(\bar{A} + B + \bar{C} + D + E)$

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

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

$= (A + B + C + DE)\left[\bar{A} + B + (C + \bar{D} + \bar{E})(\bar{C} + D + E)\right]$

$= (A + B + C + DE)\left[\bar{A} + B + CD + CE + \bar{C}\bar{D} + \bar{D}\bar{E} + \bar{C}\bar{E} + D\bar{E}\right]$

$= (A + B + C + DE)\left[\bar{A} + B + (CE + \bar{C}\bar{D} + \bar{D}E) + (CD + \bar{C}\bar{E} + D\bar{E})\right]$

$= (A + B + C + DE)(\bar{A} + B + CE + \bar{C}\bar{D} + CD + \bar{C}\bar{E})$

$= B + (A + C + DE)(\bar{A} + CE + \bar{C}\bar{D} + CD + \bar{C}\bar{E})$

$= B + ACE + A\overline{C}\overline{D} + ACD + A\overline{C}\overline{E} + \overline{A}C + CE + CD + \overline{A}DE + CDE$

$= B + A\overline{C}\overline{D} + A\overline{C}\overline{E} + (ACE + \overline{A}C + CE) + CD + \overline{A}DE$

$= B + A\overline{C}\overline{D} + A\overline{C}\overline{E} + ACE + \overline{A}C + CD + \overline{A}DE$

2.7.2. Tối thiểu hoá hàm logic bằng bảng Karnaugh

– Cho hàm dạng tổng các tích:

Phương pháp này được tiến hành theo các bước sau:

1. Biểu diễn hàm đã cho trên bảng Karnaugh.

2. Kết hợp thành từng nhóm 2^n ô gồm các ô có giá trị bằng "1" hoặc "x" chỉ khác nhau 1 bit liên tiếp nhau trên bảng Karnaugh. Khi kết hợp các ô cần tuân theo quy tắc sau:

– Số ô chứa trong 1 nhóm phải là tối đa (2^n ô với n là tối đa).

– Trong mỗi nhóm phải có ít nhất một ô chứa giá trị "1" không nằm trong nhóm khác, mặt khác mỗi ô chứa giá trị "1" có thể được sử dụng để kết hợp nhiều lần.

– Phải đảm bảo tất cả các ô chứa giá trị "1" đều được kết hợp và số nhóm kết hợp phải là tối thiểu.

3. Nhóm 2^n ô sẽ bỏ đi được n biến đó là những biến vừa xuất hiện ở cả dạng trực tiếp lẫn dạng đảo, số hạng tạo thành là tịch các biến còn lại được gọi là tích cực tiểu. Hàm tối giản là tổng các tích cực tiểu.

4. Trong một số trường hợp, có thể có nhiều cách kết hợp, nghĩa là có thể có nhiều hàm tối giản. Những hàm tối giản này cần được so sánh, kiểm tra để chọn ra hàm tối giản thực sự.

Ví dụ 2.7.2.1. Cho hàm $F(A,B,C) = \sum_m(0,1,2,5)$ hãy tối thiểu hoá hàm bằng bảng Karnaugh.

Giải: Kết hợp các ô như hình 2.7.2.1 ta được ít nhất 2 nhóm phủ hết các ô chứa giá trị "1" của hàm, các ô trong nhóm chỉ khác nhau một bit.

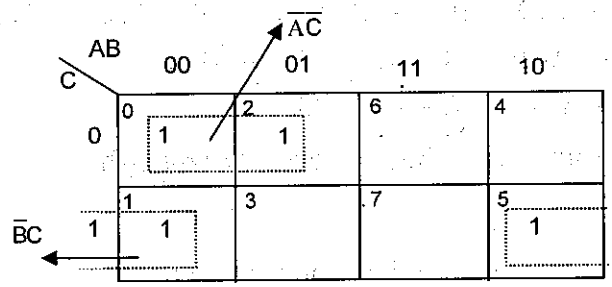
Nhóm 1 gồm 2 ô (0, 2), trong đó A và C luôn giữ nguyên giá trị "0" còn B vừa có giá trị "0" vừa có giá trị "1" nên B triệt tiêu còn lại tích $\overline{A}\overline{C}$ (nhóm này tương đương với $\overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} = \overline{A}\overline{C}(\overline{B} + B)$).

Nhóm 2 gồm 2 ô (1, 5), trong đó B luôn giữ nguyên giá trị "0" và C luôn giữ nguyên giá trị "1", còn A vừa có giá trị "0" vừa có giá trị "1" nên A triệt tiêu còn lại tích $\overline{B}\,C$ (nhóm này tương đương với $\overline{A}\overline{B}C + A\overline{B}C = \overline{B}C(\overline{A} + A)$).

Không kết hợp ô 0 và 1 vì cả hai ô đó đều đã thuộc 2 nhóm trên. Ta được:

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

Khi kết hợp các ô ta cần xét đối với các ô ít có khả năng kết hợp trước, nếu không có thể tạo ra các nhóm thừa. Như ở ví dụ 2.7.2.1, nếu ban đầu ta quan sát thấy ô 0 và 1 là kế cận nhau và ta kết hợp luôn thì nhóm đó sẽ thừa.



Hình 2.7.2.1. Tối thiểu hoá hàm

Ví dụ 2.7.2.2. Cho hàm $F(A,B,C,D)=\sum(0,1,2,5,6,7,10,14)$ hãy tối thiểu hoá hàm bằng bảng Karnaugh.

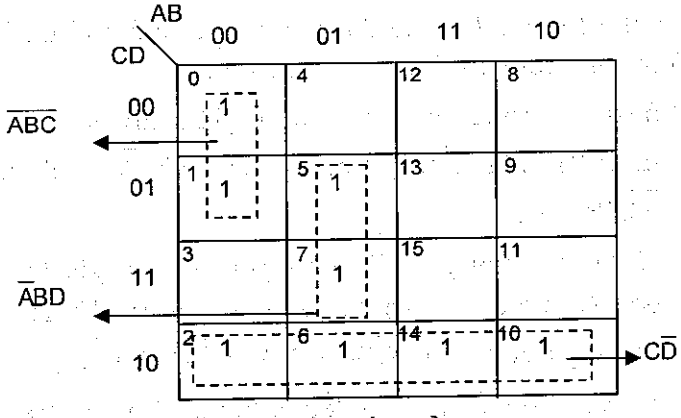
Giải: Kết hợp các ô như hình 2.7.2.2 ta được ít nhất 3 nhóm phủ hết các ô chứa giá trị "1" của hàm, các ô trong nhóm khác nhau một bit liên tiếp.

Nhóm 1 gồm 2 ô (0, 1), trong đó A, B, C luôn giữ nguyên giá trị "0" còn D vừa có giá trị "0" vừa có giá trị "1" nên D triệt tiêu còn lại tích $\overline{ABC}$ (nhóm này tương đương với $\overline{ABCD}+\overline{ABCD}=\overline{ABC}(\overline{D}+D)$).

Nhóm 2 gồm 2 ô (5, 7), trong đó A luôn giữ nguyên giá trị "0" và B,D luôn giữ nguyên giá trị "1" còn C vừa có giá trị "0" vừa có giá trị "1" nên C triệt tiêu còn lại tích $\overline{A}BD$ (nhóm này tương đương với $\overline{ABC}D+\overline{A}BCD=\overline{A}BD(\overline{C}+C)$).

Nhóm 3 gồm 4 ô (2,6,10,14), trong đó C luôn giữ nguyên giá trị "1" và D luôn giữ nguyên giá trị "0" còn A và B vừa có giá trị "0' vừa có giá trị "1" nên bị triệt tiêu còn lại tích $C\overline{D}$ (nhóm này tương đương với $\overline{A}\overline{B}CD+\overline{A}BC\overline{D}+AB\overline{C}\overline{D}+A\overline{B}C\overline{D}=C\overline{D}$)

Ta được: $F=\overline{ABC}+\overline{A}BD+C\overline{D}$



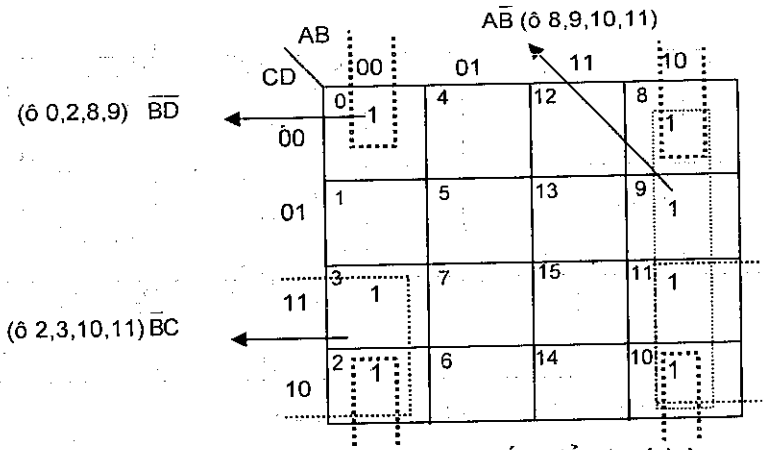
Hình 2.7.2.2. Tối thiểu hoá hàm

Trong ví dụ 2.7.2.2 ta thấy nếu kết hợp các ô (0,1), (1,5), (6,7), (2,6,10,14) hoặc (0,2), (1,5), (5,7), (2,6,10,14) hoặc (0,2), (1,5), (6,7), (2,6,10,14) thì trong mỗi nhóm đều thỏa mãn các quy tắc nhưng số nhóm chưa tối thiểu.

Ví dụ 2.7.2.3. Cho hàm $F = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}CD + \bar{A}\bar{B}C\bar{D} + A\bar{B}$

Hãy tối thiểu hoá hàm bằng bảng Karnaugh.

Giải: Kết hợp các ô như hình 2.7.2.3 ta được ít nhất 3 nhóm phủ hết các ô chứa giá trị "1" của hàm, với số ô trong nhóm là tối đa. Ta có: $F = \bar{B}\bar{D} + A\bar{B} + \bar{B}C$

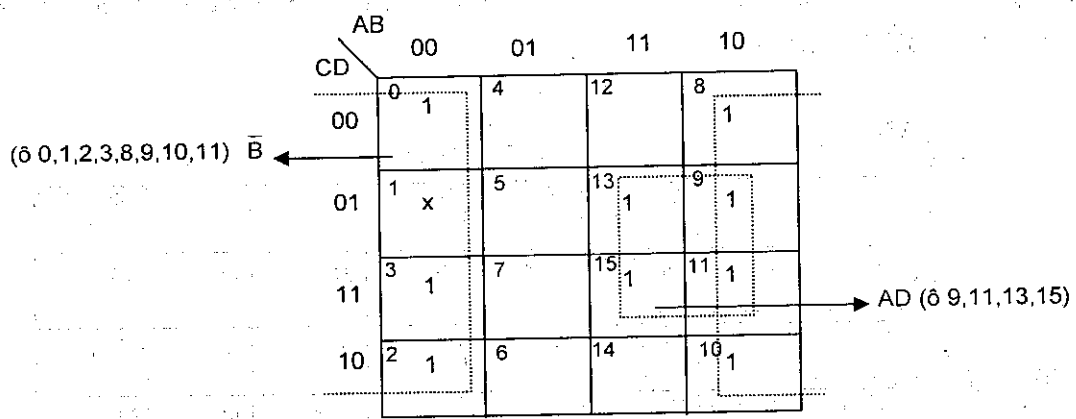


Hình 2.7.2.3. Tối thiểu hoá hàm

Ví dụ 2.7.2.4. Cho hàm $F(A,B,C,D) = \sum(0,2,3,8,9,10,11,13,15)$ với N = 1.

Hãy tối thiểu hoá hàm bằng bảng Karnaugh.

Giải: Kết hợp các ô như hình 2.7.2.4 ta được ít nhất hai nhóm phủ hết các ô "1" của hàm, số ô trong nhóm là tối đa. Ta được $F = \bar{B} + AD$



Hình 2.7.2.4. Tối thiểu hoá hàm

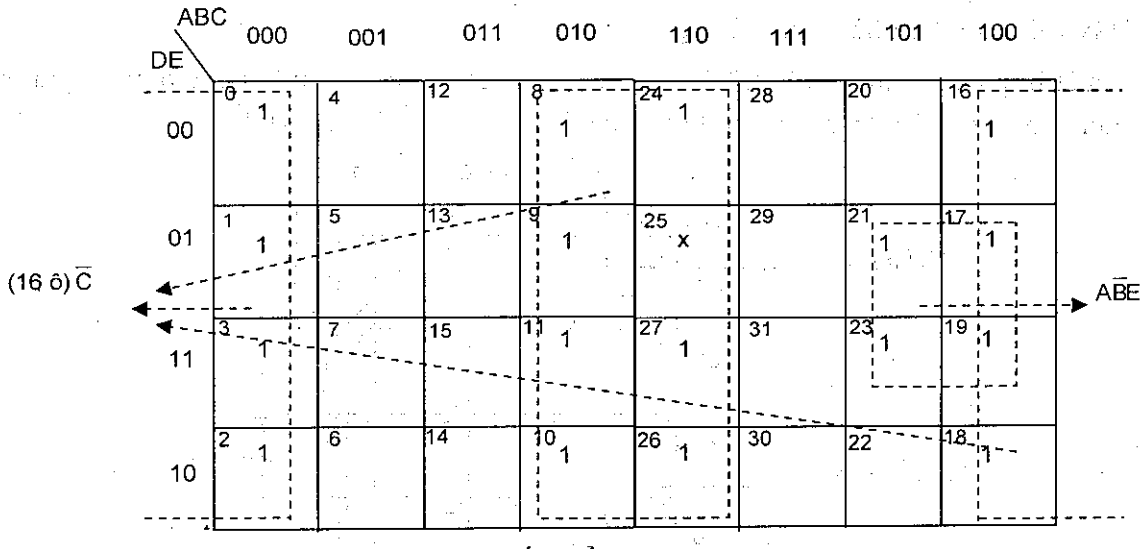
Ví dụ 2.7.2.5. Cho hàm logic:

$F(A,B,C,D,E) = \sum_m(0,1,2,3,8,9,10,11,16,17,18,19,21,23,24,26,27)$ với N = 25.

Tối thiểu hóa hàm bằng bảng Karnaugh?

Giải: Kết hợp các ô như hình 2.7.2.5 ta được ít nhất hai nhóm phủ hết các ô "1" của hàm, số ô trong nhóm là tối đa. Nhóm 1 gồm 16 ô và nhóm hai gồm 4 ô, ta có:

$F = \bar{C} + A\bar{B}E$



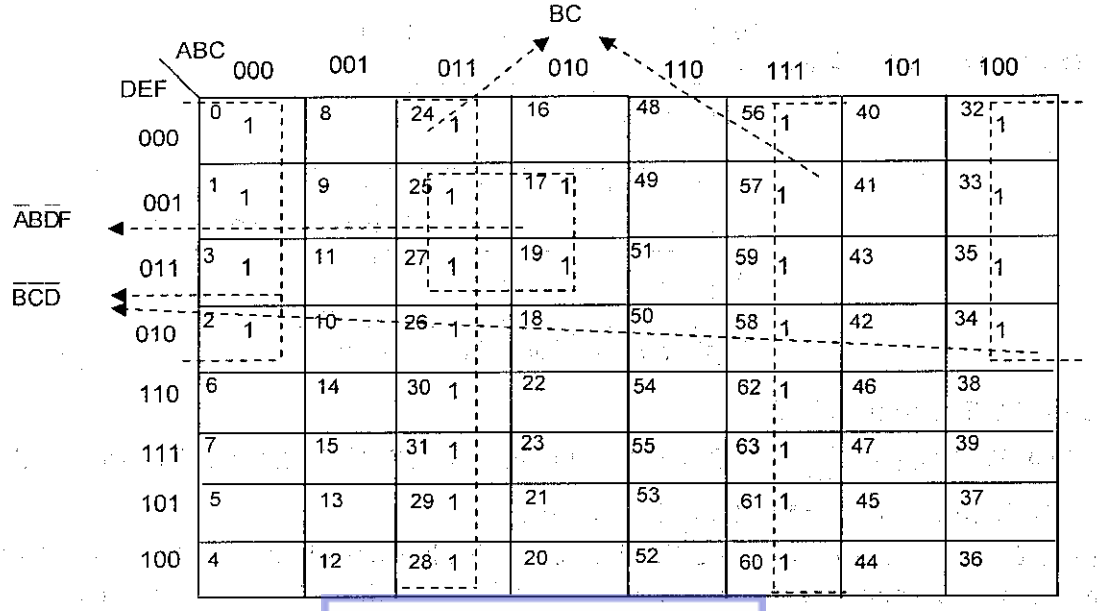
Hình 2.7.2.5. Tối thiểu hoá hàm

Ví dụ 2.7.2.6. Cho hàm logic:

$$F(A,B,C,D,E,F) = \sum_m (0,1,2,3,17,19,24,25,26,27,28,29,30,31,32,33,34,35,$$
$$56,57,58,59,60,61,62,63)$$

Tối thiểu hóa hàm bằng bảng Karnaugh?

Giải: Kết hợp các ô như hình 2.7.2.6 ta được ít nhất 3 nhóm phủ hết các ô "1" của hàm, số ô trong nhóm là tối đa. Nhóm 1 gồm 16 ô, nhóm 2 gồm 8 ô và nhóm 3 gồm 4 ô, ta có: $F = BC + \overline{B}\overline{C}\overline{D} + \overline{A}B\overline{D}F$



Hình 2.7.2.6. Tối thiểu hoá hàm

– Cho hàm dạng tích các tổng:

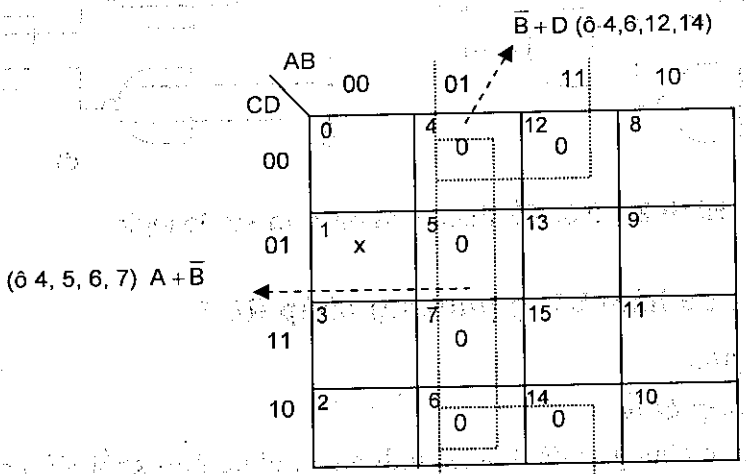
Phương pháp tương tự như hàm ở dạng tổng các tích, chỉ khác là thay các ô chứa giá trị "1" bằng các ô chứa giá trị "0" và thay tổng các tích bằng tích các tổng khi biểu diễn hàm.

Ví dụ 2.7.2.7. Cho hàm $F(A,B,C,D) = \Pi(4,5,6,7,12,14)$ với N = 1.

Tối thiểu hoá hàm bằng bảng Karnaugh.

Giải: Kết hợp các ô như hình 2.7.2.7 ta được ít nhất 2 nhóm phủ hết các ô "0" của hàm, số ô trong nhóm là tối đa. Ta có: $F = (A + \overline{B})(\overline{B} + D)$

Tuỳ thuộc vào yêu cầu thiết kế (loại phần tử sử dụng) mà ta lựa chọn phương pháp tối thiểu cho phù hợp. Chẳng hạn, chỉ dùng cổng NAND thì phương trình viết dưới dạng tổng các tích, chỉ dùng cổng NOR thì phương trình viết dưới dạng tích các tổng sau đó đảo 2 lần và áp dụng định lý Morgan để biến đổi phương trình.



Hình 2.7.2.7. Tối thiểu hoá hàm

Ví dụ 2.7.2.8. Cho hàm logic 4 biến A, B, C, D: $F(A,B,C,D) = \sum(4,5,12,13,14,15)$

a) Vẽ sơ đồ logic chỉ dùng cổng NAND.

b) Vẽ sơ đồ logic chỉ dùng cổng NOR.

Giải:

a) Tối thiểu hóa hàm:

Kết hợp các ô có giá trị 1 trong bảng Karnaugh như hình 2.7.2.8a, ta được:

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

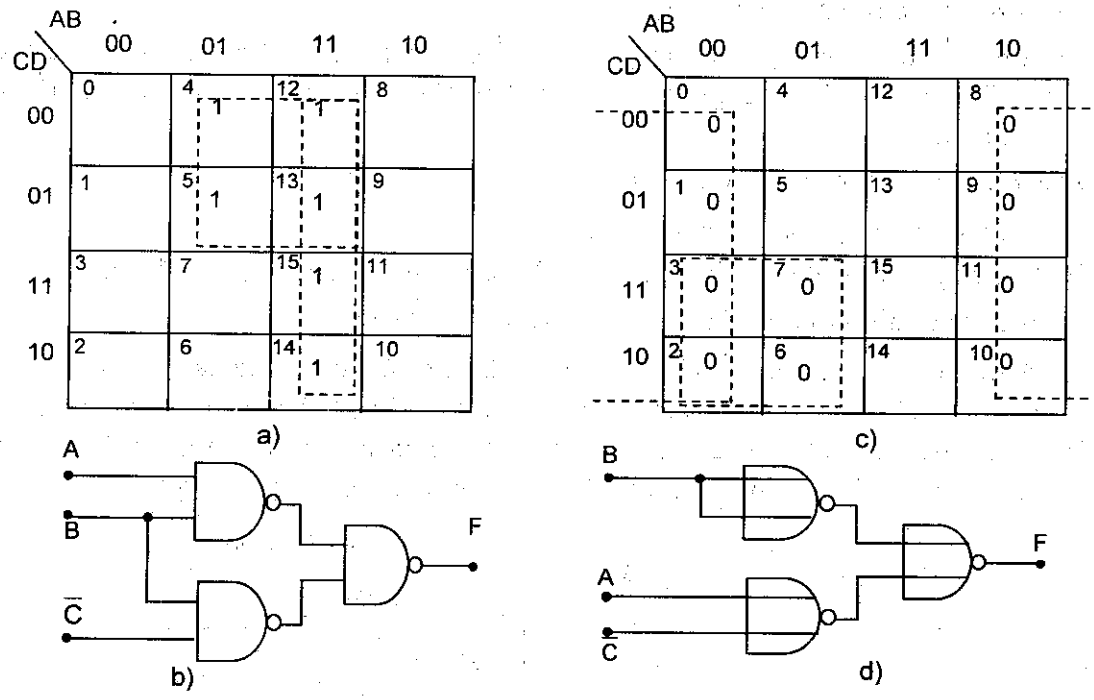
Sơ đồ logic như hình 2.7.2.8b.

b) Tối thiểu hóa hàm:

Kết hợp các ô có giá trị 0 trong bảng Karnaugh như hình 2.7.2.8c, ta được:

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

Sơ đồ logic như hình 2.7.2.8d.



Hình 2.7.2.8. Tối thiểu hoá hàm và sơ đồ logic

2.7.3. Tối thiểu hóa hàm bằng phương pháp ROT

2.7.3.1. Bài toán phủ

a) Phủ của phức hợp khối

Trong phần trước chúng ta đã đưa ra định nghĩa phức hợp khối K(f) của hàm logic n biến $f(x_1, x_2,\ldots, x_n)$ và đưa ra sự tương ứng giữa hàm logic f và phức hợp khối K(f).

* *Định nghĩa phủ của phức hợp khối*: Phủ C của phức hợp khối K(f) là tập hợp C các khối của phức hợp khối K(f) (đã cho) sao cho mỗi đỉnh của K^0 phải nằm ít nhất trong một khối nào đó của tập hợp C. Vậy: $C \subseteq K(f)$

Ví dụ 2.7.3.1. Cho hàm logic 2 biến x_1, x_2 như sau:

$$f(x_1,x_2) = \overline{x_1}\,\overline{x_2} + x_1\overline{x_2} + x_1x_2$$

– Hãy biểu diễn hàm ở dạng khối?

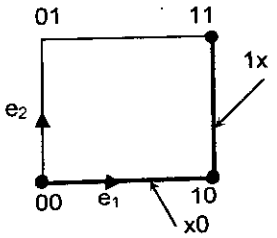
– Hãy tìm K(f)?

– Hãy tìm các phủ C của K(f)?

Giải: Hàm được biểu diễn ở dạng khối trên hình 2.7.3.1:

Vậy:

$$C_1 = K(f) = \begin{Bmatrix} 00 & x0 \\ 10 & 1x \\ 11 & \end{Bmatrix},\ C_2 = \begin{Bmatrix} 00 \\ 10 \\ 11 \end{Bmatrix},\ C_3 = \begin{Bmatrix} 11 \\ x0 \end{Bmatrix},\ C_4 = \begin{Bmatrix} 00 \\ 10 \\ 1x \end{Bmatrix},\ldots.$$



$$K(f)=\begin{Bmatrix}00 & x0\\ 10 & 1x\\ 11 & \end{Bmatrix}$$

Hình 2.7.3.1. Dạng khối

b) Giá của phủ

** Giá của khối:*

Định nghĩa: Giả sử ta có khối r chiều C^r của hàm n biến, giá của khối C^r này được định nghĩa như sau: $G^a[C^r]=n-r$.

Ví dụ 2.7.3.2. Cho các khối sau đây của hàm hai biến:

$C^0(01)$, $C^1(1x)$, $C^1(x0)$

Hãy tìm giá G^a của các khối này?

Giải:

$G^a[C^0]=G^a[01]=2-0=2$

$G^a[C^1]=G^a[1x]=2-1=1$

$G^a[C^1]=G^a[x0]=2-1=1$

Ví dụ 2.7.3.3. Cho các khối sau đây của hàm 3 biến:

$C^0(001)$, $C^1(1x0)$, $C^2(xx1)$

Hãy tìm giá G^a của các khối này?

Giải:

$G^a[C^0]=G^a[001]=3-0=3$

$G^a[C^1]=G^a[1x0]=3-1=2$

$G^a[C^2]=G^a[xx1]=3-2=1$

** Giá của phủ:*

– Giá G^a của phủ C:

Định nghĩa: Giá G^a của phủ C là tổng các giá G^a của các khối trong phủ này:

$$G^a[C]=\sum_{r=0}^{n} q_r(n-r)$$

q_r: tổng các khối r chiều.

Ví dụ 2.7.3.4. Cho hàm hai biến có phủ như sau:

$C=\begin{Bmatrix}00\\ 0x\\ 11\end{Bmatrix}$ Hãy tìm $G^a[C]$?

Giải: n = 2, $q_0 = 2$, $q_1 = 1$, $q_2 = 0$

Vậy: $G^a[C] = .2.(2 - 0) + 1.(2 - 1) = 5$.

– *Giá G^b của phủ C:*

Định nghĩa: Giá G^b của phủ C là giá G^a của phủ cộng với số lượng các khối p chứa trong phủ này:

$$G^b[C] = G^a + p = \sum_{r=0}^{n} q_r(n-r) + \sum_{r=0}^{n} q_r = \sum_{r=0}^{n} q_r(n-r+1)$$

Ví dụ 2.7.3.5. Cho hàm hai biến có phủ C:

$$C = \begin{Bmatrix} 00 \\ x0 \\ 11 \end{Bmatrix}$$ Hãy tìm $G^b[C]$?

Giải: n = 2, $q_0 = 2$, $q_1 = 1$, $p = q_0 + q_1 = 3$.

Vậy: $G^b[C] = 2.(2 - 0) + 1.(2 - 1) + 3 = 8$.

** Nhận xét:*

– Giá G^a chính là số đầu vào của mạch AND.

– p là số đầu vào của mạch OR.

Vậy giá G^b chính là số lượng đầu vào của mạch AND và mạch OR để thực hiện hàm $f(x_1, x_2, \ldots, x_n)$.

Ví dụ 2.7.3.6. Cho hàm 6 biến có phủ như sau:

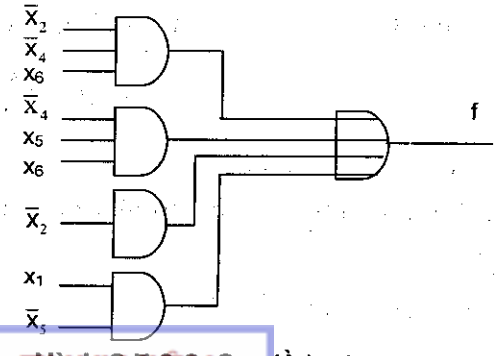
$$C = \begin{Bmatrix} x0x0x1 \\ xxx011 \\ x0xxxx \\ 1xxx0x \end{Bmatrix}$$ Hãy tìm $G^b[C]$?

Hãy vẽ mạch logic thực hiện phủ này?

Hãy tìm $G^b[C]$?

Giải: Ta có hàm f tương ứng với phủ C: $f = \overline{x_2 x_4} x_6 + \overline{x_4} x_5 x_6 + \overline{x_2} + x_1 \overline{x_5}$

+ Sơ đồ trên hình 2.7.3.2:



Hình 2.7.3.2. Sơ đồ logic

+ Ta có n = 6, $q_0 = 0$, $q_1 = 0$, $q_2 = 0$, $q_3 = 2$, $q_4 = 1$, $q_5 = 1$, $q_6 = 0$.

$$G^b[C] = \sum_{r=0}^{n} q_r(n-r+1) = 2(6-3+1)+1(6-4+1)+1(6-5+1) = 13$$

c) Phủ tối thiểu

– *Định nghĩa:* Phủ C của phức hợp khối K(f) được gọi là phủ tối thiểu nếu phủ này có giá G^b tối thiểu.

– Ký hiệu phủ tối thiểu là C_{min}.

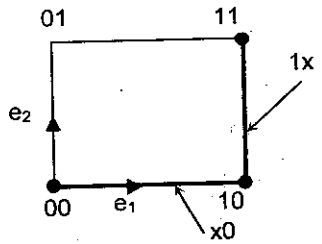
Ví dụ 2.7.3.7. Cho hàm logic 2 biến x_1, x_2 như sau:

$$f(x_1, x_2) = \overline{x_1 x_2} + x_1\overline{x_2} + \overline{x_1}x_2$$

– Hãy biểu diễn hàm ở dạng khối và tìm K(f)?

– Hãy tìm phủ tối thiểu của K(f), $G^b[C_{min}]$?

Giải: Hàm được biểu diễn ở dạng khối trên hình 2.7.3.3



Hình 2.7.3.3. Dạng khối

$$K(f) = \begin{Bmatrix} 00 & x0 \\ 10 & 1x \\ 11 & \end{Bmatrix} ; \quad C_{min} = \begin{Bmatrix} x0 \\ 1x \end{Bmatrix}$$

Vậy: $G^b[C_{min}] = 2(2 - 1 + 1) = 4$

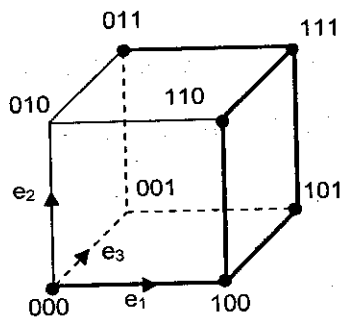
Ví dụ 2.7.3.8. Cho hàm logic 3 biến x_1, x_2, x_3 sau đây:

$$F(x_1, x_2, x_3) = \overline{x_1 x_2 x_3} + x_1 x_2 \overline{x_3} + \overline{x_1} x_2 x_3 + x_1 \overline{x_2 x_3} + x_1 x_2 x_3 + x_1 \overline{x_2} x_3$$

– Hãy biểu diễn hàm ở dạng khối và tìm K(f)?

– Hãy tìm phủ tối thiểu C_{min} và $G^b[C_{min}]$?

Giải: Hàm được biểu diễn dưới dạng khối trên hình 2.7.3.4



Hình 2.7.3.4. Dạng khối

$$K(f) = \begin{Bmatrix} 000 & x00 & 1xx \\ 100 & 10x & \\ 101 & 1x1 & \\ 111 & x11 & \\ 110 & 11x & \\ 011 & 1x0 & \end{Bmatrix} ; \quad C_{min} = \begin{Bmatrix} x11 \\ x00 \\ 1xx \end{Bmatrix}$$

$G^b[C_{min}] = 2(3 - 1 + 1) + 1(3 - 2 + 1) = 8.$

Nhận xét:

Trong ví dụ này ta thấy C_{min} chính là K^2 và các tổ hợp của K^1 mà không chứa trong K^2.

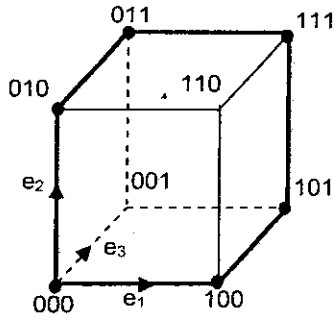
Ví dụ 2.7.3.9. Cho hàm logic 3 biến x_1, x_2, x_3 sau đây:

$$F(x_1,x_2,x_3) = \overline{x_1}\overline{x_2}\overline{x_3} + \overline{x_1}x_2\overline{x_3} + \overline{x_1}x_2x_3 + x_1\overline{x_2}\overline{x_3} + x_1x_2x_3 + x_1\overline{x_2}x_3$$

– Hãy biểu diễn hàm ở dạng khối và tìm K(f)?

– Hãy tìm phủ tối thiểu C_{min} và $G^b[C_{min}]$?

Giải: Hàm được biểu diễn dưới dạng khối trên hình 2.7.3.5.



Hình 2.7.3.5. Dạng khối

$$K(f) = \left\{\begin{matrix} 000 & x00 \\ 100 & 10x \\ 101 & 1x1 \\ 111 & x11 \\ 010 & 01x \\ 011 & 0x0 \end{matrix}\right\};$$

$$C_{min1} = \left\{\begin{matrix} x00 \\ 1x1 \\ 01x \end{matrix}\right\} \text{ hoặc } C_{min2} = \left\{\begin{matrix} 10x \\ x11 \\ 0x0 \end{matrix}\right\}$$

$G^b[C_{min}] = 3(3 - 1 + 1) = 9.$

d) Bài toán phủ

Trong thực tế có thể hàm logic có các tổ hợp biến không xác định như trong phần trước đã trình bày, và ta có các ký hiệu như sau:

– L: Là phức hợp khối của các tổ hợp biến xác định (hàm lấy giá trị bằng 1).

– N: Là phức hợp khối của các tổ hợp biến không xác định (hàm lấy giá trị x).

Khi tối thiểu hóa chúng ta có thể sử dụng các tổ hợp biến không xác định như đã trình bày trong phần tối thiểu hóa bằng bảng Karnaugh.

Khi biểu diễn hàm ở dạng khối ta ký hiệu đỉnh của L bằng dấu "•" và đỉnh của N bằng dấu "X".

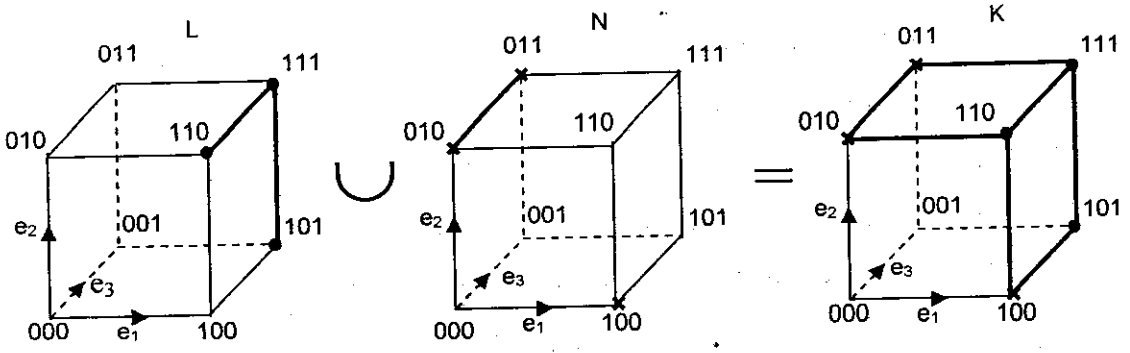
Gọi K là phức hợp khối của các biến xác định và không xác định: $K = L \cup N$

Ví dụ 2.7.3.10. Cho hàm logic 3 biến sau: $F(x_1,x_2,x_3) = \sum(5,6,7)$ với N = 2,3,4.

– Hãy biểu diễn hàm ở dạng khối?

– Hãy tìm L, N và K?

Giải: Hàm được biểu diễn dưới dạng khối trên hình 2.7.3.6



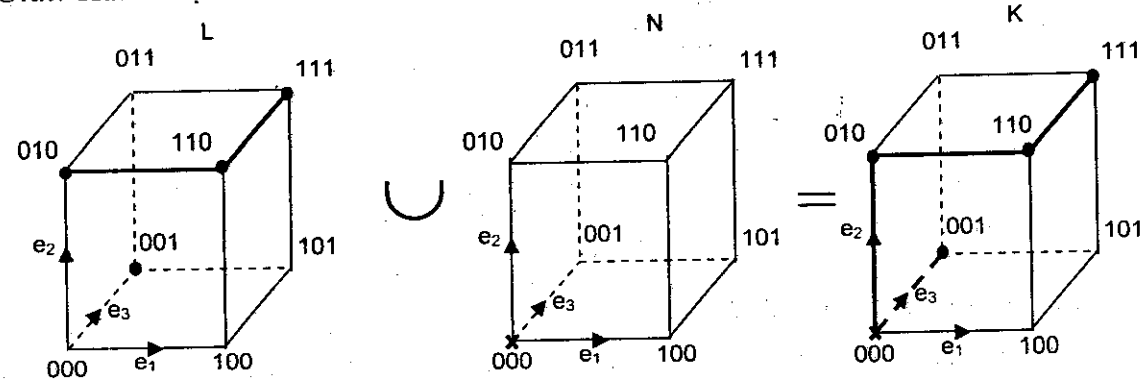
Hình 2.7.3.6. Biểu diễn dưới dạng khối

$$L=\begin{Bmatrix}101\\111\\110\\1x1\\11x\end{Bmatrix},\ N=\begin{Bmatrix}010\\100\\011\\01x\end{Bmatrix},\ K=\begin{Bmatrix}101 & 010 & x11 & x1x\\111 & 100 & x10 & 1xx\\110 & 011 & 1x0 & \\1x1 & 01x & 10x & \\11x & & & \end{Bmatrix}$$

Ví dụ 2.7.3.11. Cho hàm logic 3 biến sau: $F(x_1,x_2,x_3)=\sum(1,2,6,7)$ với N = 0.

– Hãy biểu diễn hàm ở dạng khối?
– Hãy tìm L, N và K?
– Hãy tìm các C^L_{min}, $G^b\left[C^L_{min}\right]$?

Giải: Hàm được biểu diễn dưới dạng khối như trên hình 2.7.3.7



Hình 2.7.3.7. Biểu diễn hàm dưới dạng khối

$$L=\begin{Bmatrix}001\\010\\110\\111\\x10\\11x\end{Bmatrix},\ N=\{000\},\ K=\begin{Bmatrix}001 & x10\\010 & 11x\\110 & 0x0\\111 & 00x\\000 & \end{Bmatrix},\ C^L_{min}=\begin{Bmatrix}001\\11x\\x10\end{Bmatrix},\ G^b[C^L_{min}]=4+6=10$$

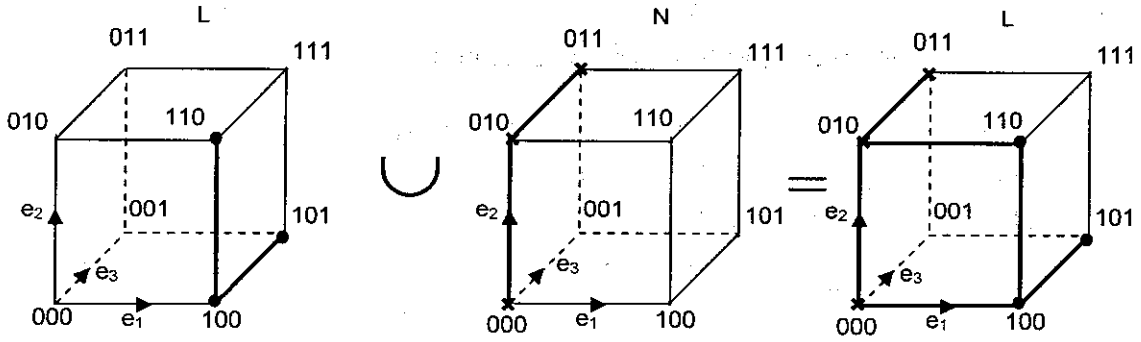
Ví dụ 2.7.3.12. Cho hàm logic 3 biến sau: $F(x_1, x_2, x_3) = \sum(4,5,6)$ với N = 0,2,3.

– Hãy biểu diễn hàm ở dạng khối?

– Hãy tìm L, N và K?

– Hãy tìm các C^L_{min}, $G^b\left[C^L_{min}\right]$, C^K_{min} và $G^b[C^K_{min}]$?

Giải: Hàm được biểu diễn dưới dạng khối trên hình 2.7.3.8



Hình 2.7.3.8. Biểu diễn hàm dưới dạng khối

$$L = \begin{Bmatrix} 100 \\ 101 \\ 110 \\ 10x \\ 1x0 \end{Bmatrix},\ N = \begin{Bmatrix} 000 \\ 010 \\ 011 \end{Bmatrix},\ K = \begin{Bmatrix} 000 & 01x & xx0 \\ 010 & x10 & \\ 110 & 0x0 & \\ 101 & x00 & \\ 100 & 1x0 & \\ 011 & 10x & \end{Bmatrix}$$

$$C^L_{min} = \begin{Bmatrix} 1x0 \\ 10x \end{Bmatrix},\ G^b[C^L_{min}] = 6,\ C^K_{min} = \begin{Bmatrix} 01x \\ 10x \\ xx0 \end{Bmatrix},\ G^b[C^K_{min}] = 8$$

Phủ k của phức hợp khối L:

– *Định nghĩa:* Phủ k của phức hợp khối L là tập hợp C^{KL} các khối sao cho các khối của C^{KL} phải nằm trong K(f) và các khối của C^{KL} phải chứa tất cả các đỉnh của L và một số đỉnh của N nếu cần thiết.

– *Nhận xét:* Phức hợp của phủ C^{KL} ta ký hiệu là: $K(C^{KL})$.

Vậy ta có quan hệ sau đây: $L \subseteq K(C^{KL}) \subseteq K(f)$

Mục đích của chúng ta là tìm được phủ K của phức hợp L có giá trị tối thiểu và đấy gọi là bài toán phủ.

Ta có: $K = L \cup N$ hay $L = K \setminus N$.

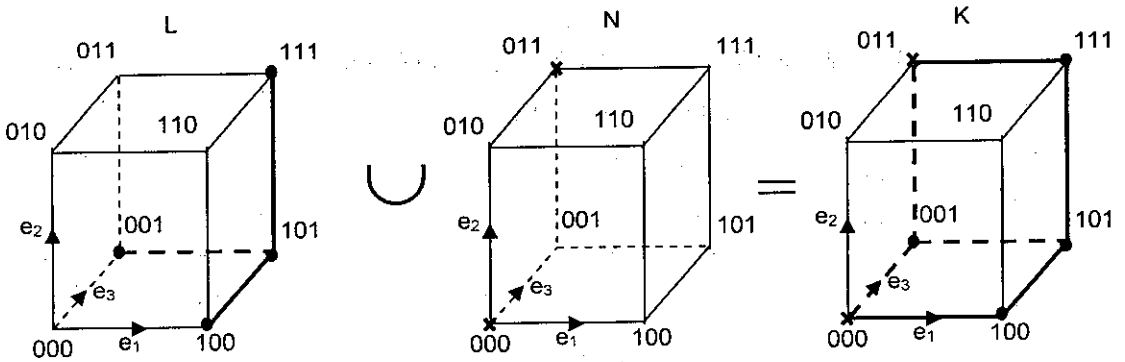
Ví dụ 2.7.3.13. Cho hàm logic 3 biến sau: $F(x_1, x_2, x_3) = \sum(1,4,5,7)$ với N = 0,3.

– Hãy biểu diễn hàm ở dạng khối?

– Hãy tìm L, N và K?

– Hãy tìm các C^L_{min}, $G^b\left[C^L_{min}\right]$, C^K_{min}, $G^b[C^K_{min}]$, C^{KL}_{min}, $G^b[C^{KL}_{min}]$?

Giải: Hàm được biểu diễn dưới dạng khối trên hình 2.7.3.9



Hình 2.7.3.9. Biểu diễn hàm dưới dạng khối

$$L = \begin{Bmatrix} 001 \\ 100 \\ 101 \\ 111 \\ 10x \\ 1x1 \\ x01 \end{Bmatrix},\ N = \begin{Bmatrix} 000 \\ 011 \end{Bmatrix},\ K = \begin{Bmatrix} 001 & 000 & x0x \\ 100 & 011 & xx1 \\ 101 & x00 & \\ 111 & 00x & \\ 10x & 0x1 & \\ 1x1 & x11 & \\ x01 & & \end{Bmatrix}$$

$$C^L_{min} = \begin{Bmatrix} 10x \\ 1x1 \\ x01 \end{Bmatrix}, G^b[C^L_{min}] = 9, C^K_{min} = \begin{Bmatrix} x0x \\ xx1 \end{Bmatrix},\ G^b[C^K_{min}] = 4, C^{KL}_{min} = \begin{Bmatrix} x0x \\ xx1 \end{Bmatrix},\ G^b[C^{KL}_{min}] = 4$$

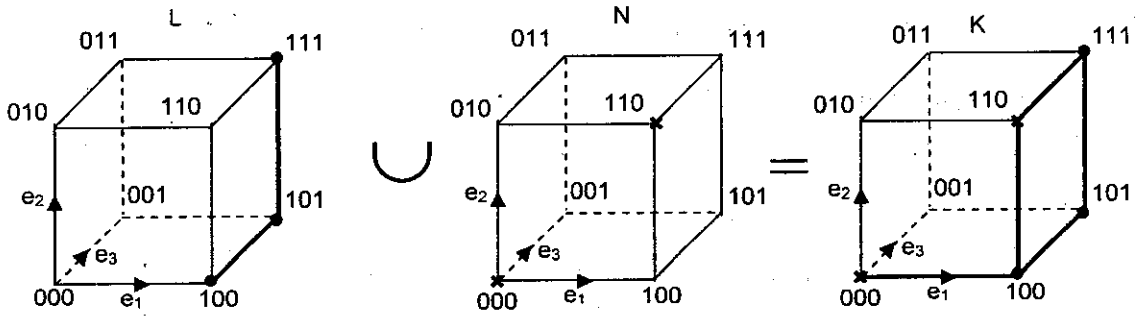
Ví dụ 2.7.3.14. Cho hàm logic 3 biến sau: $F(x_1, x_2, x_3) = \sum(4,5,7)$ với N = 0,6.

– Hãy biểu diễn hàm ở dạng khối?

– Hãy tìm L, N và K?

– Hãy tìm các C^L_{min}, $G^b\left[C^L_{min}\right]$, C^K_{min}, $G^b[C^K_{min}]$, C^{KL}_{min}, $G^b[C^{KL}_{min}]$?

Giải: Hàm được biểu diễn dưới dạng khối trên hình 2.7.3.10



Hình 2.7.3.10. Biểu diễn hàm dưới dạng khối

$$L=\begin{Bmatrix}100\\101\\111\\10x\\1x1\end{Bmatrix},\ N=\begin{Bmatrix}000\\110\end{Bmatrix},\ K=\begin{Bmatrix}100 & 000 & 1xx\\101 & 110 & \\111 & x00 & \\10x & 1x0 & \\1x1 & 11x & \end{Bmatrix}$$

$$C^{L}_{min}=\begin{Bmatrix}10x\\1x1\end{Bmatrix},\ G^{b}[C^{L}_{min}]=6,\ C^{K}_{min}=\begin{Bmatrix}x00\\1xx\end{Bmatrix},\ G^{b}\left[C^{K}_{min}\right], C^{KL}_{min}=\{1xx\},\ G^{b}[C^{KL}_{min}]=2$$

2.7.3.2. Implicant đơn giản

a) Implicant đơn giản là gì

– Định nghĩa: Khối z nào đó của phức hợp khối K(f) là implicant đơn giản nếu nó tuân theo điều kiện sau đây: $\delta_i(z) = \emptyset$ với mọi i.

Tức là: Khối z là implicant đơn giản của phức hợp khối K(f) nếu trong K(f) không có bất kỳ một khối nào khác có chứa z.

Nói cách khác: Implicant đơn giản z không thuộc bất kỳ một khối nào khác của phức hợp khối K(f).

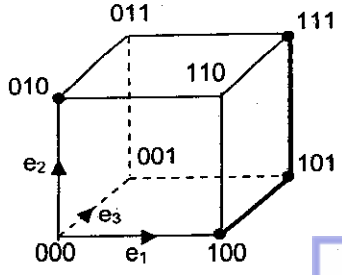
Ví dụ 2.7.3.15. Cho hàm logic 3 biến x_1, x_2, x_3 sau đây:

$$F(x_1,x_2,x_3)=\overline{x_1}x_2\overline{x_3}+x_1\overline{x_2x_3}+x_1x_2x_3+x_1\overline{x_2}x_3$$

– Hãy biểu diễn hàm ở dạng khối và tìm K(f)?

– Hãy tìm các implicant đơn giản của phức hợp khối K(f)?

Giải: Hàm được biểu diễn dưới dạng khối trên hình 2.7.3.11:



Hình 2.7.3.11. Dạng khối của hàm

$$K(f)=\begin{Bmatrix}010 & 10x\\100 & 1x1\\101 & \\111 & \end{Bmatrix}$$

Xét implicant đơn giản là khối 0: z^0

– Với: $z_1^0 = (010)$ ta có: $\delta_1(010) = Ø$, $\delta_2(010) = Ø$, $\delta_3(010) = Ø$

Vậy $\delta_i(010) = Ø$ với mọi i, khối 0 (010) là implicant đơn giản 0 chiều.

– Với: $z_2^0 = (100)$ ta có: $\delta_1(100) = Ø$, $\delta_2(010) = Ø$, $\delta_3(010) = 10x$

Vậy khối 0 (100) không phải là implicant đơn giản.

– Với: $z_3^0 = (101)$ ta có: $\delta_1(101) = Ø$, $\delta_2(101) = 1x1$, $\delta_3(101) = 10x$

Vậy khối 0 (101) không phải là implicant đơn giản.

– Với: $z_4^0 = (111)$ ta có: $\delta_1(111) = Ø$, $\delta_2(111) = 1x1$, $\delta_3(111) = Ø$.

Vậy khối 0 (111) không phải là implicant đơn giản.

Ta có $z_1^0 = (010)$ được gọi là implicant đơn giản 0 chiều.

Xét implicant đơn giản là khối 1: z^1

– Với: $z_1^1 = (10x)$ ta có: $\delta_1(10x) = Ø$, $\delta_2(10x) = Ø$, $\delta_3(10x) = Ø$

Vậy $\delta_i(10x) = Ø$ với mọi i, khối 1 (10x) là implicant đơn giản 1 chiều.

– Với: $z_2^1 = (1x1)$ ta có: $\delta_1(1x1) = Ø$, $\delta_2(1x1) = Ø$, $\delta_3(1x1) = Ø$

Vậy $\delta_i(1x1) = Ø$ với mọi i, khối 1 (1x1) là implicant đơn giản 1 chiều.

Tổng kết: Ta có 3 Implicant đơn giản là z_1^0, z_1^1, z_2^1.

b) Tập hợp implicant đơn giản

* Tập hợp các implicant đơn giản r chiều ký hiệu là: Z^r.

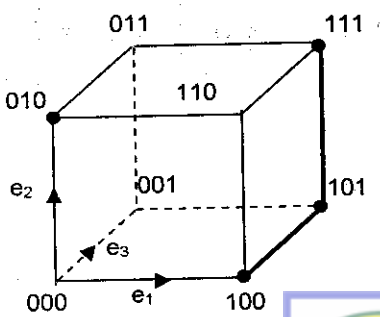
* Tập hợp các implicant đơn giản r chiều Z^r sẽ tạo thành tập hợp các implicant đơn giản Z: $Z = \bigcup_{r=0}^{n} Z^r$

Ví dụ 2.7.3.16. Cho hàm logic 3 biến x_1, x_2, x_3 sau đây:

$$F(x_1, x_2, x_3) = \overline{x_1} x_2 \overline{x_3} + x_1 \overline{x_2}\ \overline{x_3} + x_1 x_2 x_3 + x_1 \overline{x_2} x_3$$

Hãy tìm Z^0, Z^1, Z^2 và Z?

Giải: Từ dạng khối của hàm trên hình 2.7.3.12, ta có:



Hình 2.7.3.12. Dạng khối của hàm

$$Z^0 = \{010\},\ Z^1 = \begin{Bmatrix} 1x1 \\ 10x \end{Bmatrix},\ Z^2 = Ø$$

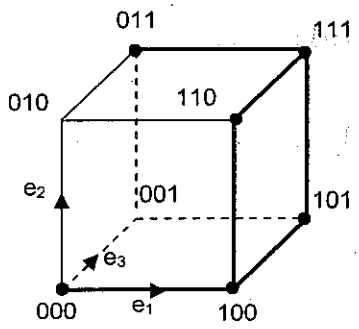
Vậy: $Z = Z^0 \cup Z^1$

Ví dụ 2.7.3.17. Cho hàm logic 3 biến x_1, x_2, x_3 sau đây:

$$F(x_1,x_2,x_3)=\overline{x_1}x_2x_3+\overline{x_1x_2x_3}+x_1x_2x_3+x_1\overline{x_2x_3}+x_1\overline{x_2}x_3+x_1x_2\overline{x_3}$$

Hãy tìm Z^0, Z^1, Z^2 và Z?

Giải: Từ dạng khối của hàm trên hình 2.7.3.13, ta có:



Hình 2.7.3.13. Dạng khối của hàm

$$Z^0=\varnothing,\ Z^1=\begin{Bmatrix}x00\\x11\end{Bmatrix},\ Z^2=\{1xx\}$$

$$\text{Vậy: } Z=Z^1\cup Z^2=\begin{Bmatrix}x00\\x11\\1xx\end{Bmatrix}$$

c) Định lý

**Định lý:* Nếu C_{min} là phủ tối thiểu thì $C_{min}\subseteq Z$

**Chứng minh:* Giả sử khối $C_k\in C_{min}$ và C_k không phải là implicant đơn giản, thì theo định nghĩa implicant đơn giản ta thấy rằng với mọi i nào đó ta có: $\delta_i(C_k)\neq\varnothing$.

Vì thế ta thay khối C_k bằng khối $\delta_i(C_k)$ để được phủ C' mới. Phủ C' mới này có giá (G^a) thấp hơn giá của phủ C_{min}, bởi vì $\delta_i(C_k)$ có giá thấp hơn khối C_k.

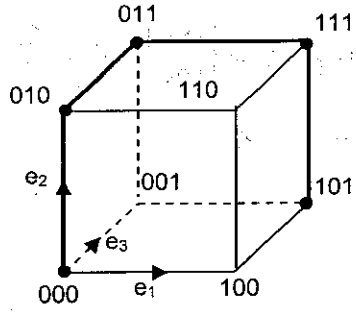
Ví dụ 2.7.3.18. Cho hàm logic 3 biến x_1, x_2, x_3 sau đây:

$$F(x_1,x_2,x_3)=\overline{x_1}x_2\overline{x_3}+\overline{x_1}x_2x_3+x_1x_2x_3+x_1\overline{x_2}x_3+\overline{x_1x_2x_3}$$

– Hãy tìm Z?

– Hãy tìm C_{min}, $G^b[C_{min}]$?

Giải: Từ dạng khối của hàm trên hình 2.7.3.14, ta có:



Hình 2.7.3.14. Dạng khối của hàm

$$Z=\begin{Bmatrix}0x0\\01x\\x11\\1x1\end{Bmatrix},C_{min1}=\begin{Bmatrix}0x0\\x11\\1x1\end{Bmatrix},\ C_{min2}=\begin{Bmatrix}0x0\\01x\\1x1\end{Bmatrix}$$

$$G^b[C_{min}]=9.$$

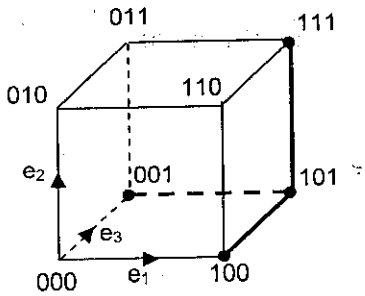
Ví dụ 2.7.3.19. Cho hàm logic 3 biến x_1, x_2, x_3 sau đây:

$$F(x_1, x_2, x_3) = \overline{x_1}\overline{x_2}x_3 + x_1x_2x_3 + x_1\overline{x_2}x_3 + x_1\overline{x_2}\,\overline{x_3}$$

+ Hãy tìm Z?

+ Hãy tìm C_{min}, $G^b[C_{min}]$?

Giải: Từ dạng khối của hàm trên hình 2.7.3.15, ta có:



$$Z = \begin{Bmatrix} x01 \\ 10x \\ 1x1 \end{Bmatrix}, \quad C_{min} = \begin{Bmatrix} x01 \\ 10x \\ 1x1 \end{Bmatrix}$$

$G^b[C_{min}] = 9.$

Hình 2.7.3.15. Dạng khối của hàm

2.7.3.3. Thuật toán (*)

Trong phần định nghĩa implicant đơn giản ta thấy rằng để tìm Z chúng ta phải xác định tất cả các khối của phức hợp khối K(f) và phải sử dụng các toán tử hợp biên giới và tách biên giới để tìm các khối của K(f) rồi phải kiểm tra mỗi một khối của K(f) để xem nó có phải là implicant đơn giản hay không.

Nếu K(f) rất nhiều khối thì khối lượng tính toán sẽ rất lớn. Để tránh nhược điểm này chúng ta sẽ xét thuật toán (*).

a) Tích tọa độ (*)

– Định nghĩa:

Tích tọa độ * được định nghĩa bằng bảng cho trên hình 2.7.3.16:

*	0	1	x
0	0	y	0
1	y	1	1
x	0	1	x

Hình 2.7.3.16. Tích toạ độ *

Tức là: $0*0 = 0$ $\quad 1*0 = y$ $\quad x*0 = 0$

$0*1 = y$ $\quad 1*1 = 1$ $\quad x*1 = 1$

$0*x = 0$ $\quad 1*x = 1$ $\quad x*x = x$

– Các tính chất của tích tọa độ (*):

Tích tọa độ (*) có các tính chất sau đây:

+ $C^r*C^s = C^s*C^r$ (tính giao hoán).

+ $C^r*(C^s*C^t) \neq (C^r*C^s)*C^t$ (không có tính chất kết hợp).

+ Nếu $C^r \subseteq C^s$ thì $C^r * C^s = C^r$.

+ Nếu $C^r * C^s = C^t$ và $C^r \subset C^t$ thì đối với i nào đó ta có: $\delta_i(C^r) \neq \varnothing$ và $\delta_i^p(C^t) \neq \varnothing$.

Và nếu chúng ta tác động một dãy các phép toán hợp biên giới δ_i vào khối C^r thì chúng ta sẽ thu được khối C^t.

Và nếu chúng ta tác động một dãy các phép toán tách biên giới δ_i^p vào khối C^t thì chúng ta sẽ thu được khối C^r.

+ Nếu C_1^r và C_2^r là các biên giới đối nhau của khối C^{r+1} nào đấy thì ta có:

$C_1^r * C_2^r = C^{r+1} = \delta_i(C_1^r) = \delta_i(C_2^r)$.

Ở đây i là tọa độ mà ở đó C_1^r và C_2^r có giá trị khác nhau.

b) Tích tọa độ trên hai khối

– Định nghĩa: Giả sử ta có hai khối C^r và C^s như sau:

$C^r = (a_1, a_2, \ldots, a_i, \ldots, a_n)$

$C^s = (b_1, b_2, \ldots, b_i, \ldots, b_n)$

a_i, b_i: các tọa độ.

C^r, C^s: các khối nằm trong phức hợp khối K(f).

$C^r * C^s$ được định nghĩa như sau:

$$C^r * C^s = \begin{cases} \text{– Ø nếu } a_i * b_i = y \text{ và số lượng y lớn hơn 1 (tức là từ 2 trở lên)} \\ \text{– } [m(a_1*b_1), m(a_2*b_2), \ldots, m(a_i*b_i), \ldots, m(a_n*b_n)] \text{ Nếu } a_i*b_i = y \text{ nhưng số lượng } y \leq 1 \\ \text{Ở đây } m(a_i*b_i) \text{ được xác định như sau: } \begin{cases} m(0) = 0 \\ m(1) = 1 \\ m(y) = m(x) = x \end{cases} \end{cases}$$

Ví dụ 2.7.3.20. Cho hàm logic 2 biến x_1, x_2 như sau:

$f(x_1, x_2) = \overline{x_1}\,\overline{x_2} + \overline{x_1}x_2 + x_1x_2$

– Hãy biểu diễn hàm ở dạng hình học?

– Hãy tìm phức hợp khối K(f)?

– Hãy tìm phủ C (theo đầu bài đã cho)?

– Hãy tìm tích tọa độ (*) giữa các khối trong phủ C?

Giải: Hàm được biểu diễn dưới dạng hình học trên hình 2.7.3.17

Ta ký hiệu: $C_1^0 = (00)$, $C_2^0 = (01)$, $\Rightarrow C_2^0 * (11)$

$C_1^0 * C_2^0 = (00)*(01) = [m(0*0), m(0*1)] = [m(0), m(y)]$

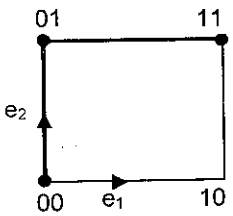
Ở đây chỉ có một y xuất hiện vậy ta có: $C_1^0 * C_2^0 = 0x$.

$C_1^0 * C_3^0 = (00)*(11) = [m(0*1), m(0*1)] = [m(y), m(y)]$

Ở đây chỉ có 2 y xuất hiện vậy ta có: $C_1^0 * C_3^0 = Ø$.

$C_2^0 * C_3^0 = (01)*(11) = [m(0*1), m(1*1)] = [m(y), m(1)]$

Ở đây chỉ có một y xuất hiện vậy ta có: $C_2^0 * C_3^0 = x1$.



Hình 2.7.3.17. Dạng khối của hàm

$$K(f) = \begin{Bmatrix} 00 & 0x \\ 01 & x1 \\ 11 & \end{Bmatrix};$$

$$C = K^0 = \begin{Bmatrix} 00 \\ 01 \\ 11 \end{Bmatrix} \text{(phủ theo đầu bài đã cho)}$$

Ví dụ 2.7.3.21. Cho hàm logic 3 biến x_1, x_2, x_3 như sau:

$$f(x_1, x_2, x_3) = \overline{x_1}\,\overline{x_2}\,\overline{x_3} + \overline{x_1}x_2x_3 + x_1x_2 + x_1\overline{x_2}$$

– Hãy biểu diễn hàm ở dạng hình học?

– Hãy tìm phủ C (theo đầu bài đã cho)?

– Hãy tìm tích tọa độ (*) giữa các khối trong phủ C?

Giải: Hàm được biểu diễn dưới dạng hình học trên hình 2.7.3.18

Ta ký hiệu: $C_1^0 = (011)$, $C_2^0 = (000)$, $C_1^1 = (11x)$, $C_2^1 = (10x)$

$C_1^0 * C_2^0 = (011)*(000) = [m(0*0), m(1*0), m(1*0)] = [m(0), m(y), m(y)] = Ø$.

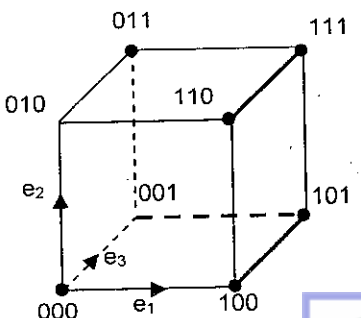
$C_1^0 * C_1^1 = (011)*(11x) = [m(0*1), m(1*1), m(1*x)] = [m(y), m(1), m(1)] = x11$.

$C_1^0 * C_2^1 = (011)*(10x) = [m(0*1), m(1*0), m(1*x)] = [m(y), m(y), m(1)] = Ø$.

$C_2^0 * C_1^1 = (000)*(11x) = [m(0*1), m(0*1), m(0*x)] = [m(y), m(y), m(0)] = Ø$.

$C_2^0 * C_2^1 = (000)*(10x) = [m(0*1), m(0*0), m(0*x)] = [m(y), m(0), m(0)] = x00$.

$C_1^1 * C_2^1 = (11x)*(10x) = [m(1*1), m(1*0), m(x*x)] = [m(1), m(y), m(x)] = 1xx$.



Hình 2.7.3.18. Dạng khối của hàm

$$C = \begin{Bmatrix} 011 \\ 000 \\ 11x \\ 10x \end{Bmatrix} \text{(phủ theo đầu bài đã cho)}$$

Ví dụ 2.7.3.22. Cho hàm logic 5 biến x_1, x_2, x_3, x_4, x_5 như sau:

$$f(x_1,x_2,x_3,x_4,x_5) = \overline{x_1x_2x_3x_4x_5} + \overline{x_1}\,\overline{x_2x_3x_4x_5} + x_2x_4 + x_1\overline{x_2x_3x_4}$$

– Hãy tìm phủ C (theo đầu bài đã cho)?

– Hãy tìm tích tọa độ * giữa các khối trong phủ C?

Giải:

$$C = \begin{Bmatrix} 00000 \\ 00001 \\ x1x1x \\ 1000x \end{Bmatrix} \text{(phủ theo đầu bài đã cho)}$$

Ta ký hiệu: $C_1^0 = (00000)$, $C_2^0 = (00001)$, $C_1^1 = (1000x)$, $C_3^1 = (x1x1x)$

$C_1^0 * C_2^0 = (00000)*(00001) = 0000x.$

$C_1^0 * C_1^1 = (00000)*(1000x) = x0000.$

$C_1^0 * C_3^1 = (00000)*(x1x1x) = Ø.$

$C_2^0 * C_1^1 = (00001)*(1000x) = x0001.$

$C_2^0 * C_3^1 = (00001)*(x1x1x) = Ø$

$C_1^1 * C_3^1 = (1000x)*(x1x1x) = Ø.$

c) Thuật toán (*)

Mục đích: Thuật toán (*) cho phép chúng ta tìm các tập hợp các implicant đơn giản Z xuất phát từ phủ ban đầu nào đấy của hàm logic $f(x_1, x_2,\ldots, x_n)$.

Các giai đoạn như sau:

– Giai đoạn 1:

+ Bắt đầu từ phủ ban đầu $\hat{C}_0$.

+ Chúng ta xây dựng phủ C_0 bằng cách loại ra các khối nằm trong các khối khác của phủ $\hat{C}_0$.

+ Chúng ta định nghĩa tập hợp $\hat{C}_0^*$ như sau:

$$\hat{C}_0^* = \left\{c \middle| c \subset d; c,d \subset \hat{C}_0\right\}$$

$\hat{C}_0^*$ là tập hợp tất cả các khối nằm trong các khối khác của $\hat{C}_0$.

+ Loại bỏ tập hợp $\hat{C}_0^*$ ra khỏi $\hat{C}_0$ để được phủ C_0.

Vậy: $C_0 = \hat{C}_0 \setminus \hat{C}_0^*$.

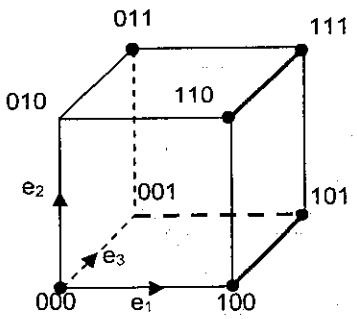
Ví dụ 2.7.3.23. Cho phủ ban đầu như sau:

$$\hat{C}_0 = \begin{Bmatrix} 011 \\ 000 \\ 111 \\ 100 \\ 11x \\ 10x \end{Bmatrix}$$

– Hãy biểu diễn hàm logic ở dạng hình học?

– Hãy tìm $\hat{C}_0^*$ và C_0?

Giải: Hàm được biểu diễn dưới dạng hình học trên hình 2.7.3.19.



Hình 2.7.3.19. Dạng khối của hàm

Ta có:

$111 \subset 11x$: 111 là biên giới của khối 11x.

$100 \subset 10x$: 100 là biên giới của khối 10x $\Rightarrow \hat{C}_0^* = \begin{Bmatrix} 111 \\ 100 \end{Bmatrix}$

$$C_0 = \hat{C}_0 \setminus \hat{C}_0^* = \begin{Bmatrix} 011 \\ 000 \\ 100 \\ 111 \\ 10x \\ 11x \end{Bmatrix} \setminus \begin{Bmatrix} 111 \\ 100 \end{Bmatrix} = \begin{Bmatrix} 011 \\ 000 \\ 10x \\ 11x \end{Bmatrix}$$

– *Giai đoạn 2:*

Tìm tập hợp các implicant đơn giản là các khối 0: Z^0 (tức là implicant đơn giản 0 chiều), chúng ta có thể tìm Z^0 nhờ quy tắc sau đây:

Bổ đề: Tập hợp các implicant đơn giản 0 chiều là tập hợp các khối 0 C^0 sao cho $C^0 * C_0$ không chứa bất kỳ khối 1 nào của phức hợp khối K(f).

Tức là: $Z^0 = \{C^0 \mid C^0 * C_0$ không chứa bất kỳ khối 1 nào của K$\}$.

Chứng minh: C^0 là khối 0 vậy nếu C^0 là implicant đơn giản thì $\delta_i(C^0) = \emptyset$, nếu $C^0 * C_0 \supseteq$ một khối nào đấy, tức là $\delta_i(C^0) \neq \emptyset$ (hoặc $\delta_i(C^0)$ = khối 1 nào đấy), tức là C^0 chứa trong khối 1 nào đấy.

Vậy nếu C^0 là implicant đơn giản thì $C^0 * C_0$ không chứa bất kỳ khối 1 nào.

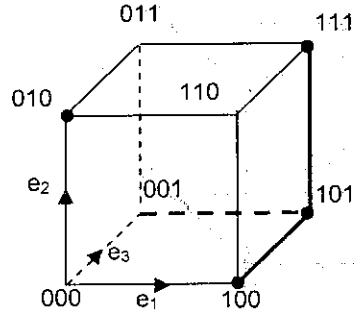
Ví dụ 2.7.3.24. Cho phủ ban đầu như sau:

$$\hat{C}_0 = \begin{Bmatrix} 010 \\ 111 \\ 10x \end{Bmatrix}$$

– Hãy biểu diễn hàm logic ở dạng hình học?

– Hãy tìm Z^0?

Giải: Hàm được biểu diễn dưới dạng hình học như trên hình 2.7.3.20.



Hình 2.7.3.20. Dạng khối của hàm

Ở đây: $\hat{C}_0^* = \varnothing \Rightarrow C_0 = \hat{C}_0$

Ta ký hiệu: $C_1^0 = (010)$, $C_2^0 = (111)$.

$$\Rightarrow C_1^0 * C_0 = (010) * \begin{Bmatrix} 010 \\ 111 \\ 10x \end{Bmatrix} = \begin{Bmatrix} (010)*(010) \\ (010)*(111) \\ (010)*(10x) \end{Bmatrix} = \begin{Bmatrix} 010 \\ y1y \\ yy0 \end{Bmatrix} = \begin{Bmatrix} 010 \\ \varnothing \\ \varnothing \end{Bmatrix}$$

$\Rightarrow C_1^0 * C_0$ không chứa bất kỳ khối 1 nào.

$\Rightarrow C_1^0$ là implicant đơn giản 0 chiều.

$$\Rightarrow C_2^0 * C_0 = (111) * \begin{Bmatrix} 010 \\ 111 \\ 10x \end{Bmatrix} = \begin{Bmatrix} (111)*(010) \\ (111)*(111) \\ (111)*(10x) \end{Bmatrix} = \begin{Bmatrix} y1y \\ 111 \\ 1x1 \end{Bmatrix} = \begin{Bmatrix} \varnothing \\ 111 \\ 1x1 \end{Bmatrix}$$

$\Rightarrow C_2^0 * C_0$ chứa một khối 1 là (1x1)

$\Rightarrow C_2^0$ không phải implicant đơn giản.

$\Rightarrow Z^0 = \{C_1^0\} = \{010\}$

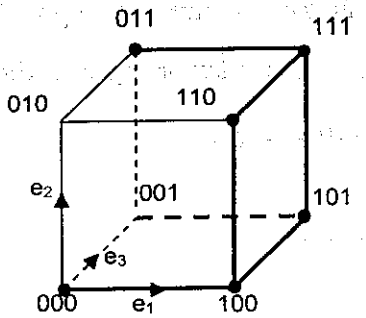
Ví dụ 2.7.3.25. Cho phủ ban đầu như sau:

$$\hat{C}_0 = \begin{Bmatrix} 011 \\ 000 \\ 11x \\ 10x \end{Bmatrix}$$

– Hãy biểu diễn hàm logic ở dạng hình học?

– Hãy tìm Z^0?

Giải: Hàm được biểu diễn dưới dạng hình học như trên hình 2.7.3.21.



Hình 2.7.3.21. Dạng khối của hàm

Ở đây: $\hat{C}_0^* = \varnothing \Rightarrow C_0 = \hat{C}_0$

Ta ký hiệu: $C_1^0 = (011)$, $C_2^0 = (000)$.

$$\Rightarrow C_1^0 * C_0 = (011) * \begin{Bmatrix} 011 \\ 000 \\ 10x \\ 11x \end{Bmatrix} = \begin{Bmatrix} (011)*(011) \\ (011)*(000) \\ (011)*(10x) \\ (011)*(11x) \end{Bmatrix} = \begin{Bmatrix} 011 \\ 0yy \\ yy1 \\ y11 \end{Bmatrix} = \begin{Bmatrix} 011 \\ \varnothing \\ \varnothing \\ x11 \end{Bmatrix}$$

$\Rightarrow C_1^0 * C_0$ chứa một khối 1: (x11).

$\Rightarrow C_1^0$ không phải là implicant đơn giản 0 chiều.

$$\Rightarrow C_2^0 * C_0 = (000) * \begin{Bmatrix} 011 \\ 000 \\ 10x \\ 11x \end{Bmatrix} = \begin{Bmatrix} (000)*(011) \\ (000)*(000) \\ (000)*(10x) \\ (000)*(11x) \end{Bmatrix} = \begin{Bmatrix} 0yy \\ 000 \\ y00 \\ yy0 \end{Bmatrix} = \begin{Bmatrix} \varnothing \\ 000 \\ x00 \\ \varnothing \end{Bmatrix}$$

$\Rightarrow C_2^0 * C_0$ chứa một khối 1 là (x00)

$\Rightarrow C_2^0$ không phải implicant đơn giản không chiều.

$\Rightarrow Z^0 = \varnothing$

– *Giai đoạn 3:*

+ Chúng ta xây dựng phủ mới C_1 không chứa các khối 0 (phủ mới C_1 này có thể không phải là phủ của K bởi vì nó không chứa các khối 0 của K nhưng $C_1 \cup Z^0$ là phủ của K).

+ Trước hết ta xây dựng phủ $\hat{C}_1$ bằng cách sau đây: $\hat{C}_1 = C_0 \cup (C_0 * C_0)$

Ở đây $C_0 * C_0$ chứa tất cả các khối ngoài khối chứa trong C_0.

+ Sau đó ta tìm C_1 bằng cách loại bỏ các khối nằm trong các khối khác của phủ $\hat{C}_1$ và loại bỏ các khối 0.

+ Ký hiệu $\hat{C}_1^0$ là tập hợp tất cả các khối 0 nằm trong phủ $\hat{C}_1$.

+ Ký hiệu $\hat{C}_1^*$ là tập hợp tất cả các khối nằm trong các khối khác:

$$\hat{C}_1^* = \{c \mid c \subset d; c, d \subset \hat{C}_1\}$$

Vậy: $C_1 = \hat{C}_1 \setminus \hat{C}_1^0 \setminus \hat{C}_1^*$.

Ví dụ 2.7.3.26. Cho phủ ban đầu như sau:

$$\hat{C}_0 = \begin{Bmatrix} 011 \\ 000 \\ 11x \\ 10x \end{Bmatrix}$$

Hãy tìm C_1?

Giải: Ở đây: $\hat{C}_0^* = \varnothing \Rightarrow C_0 = \hat{C}_0$

$$C_0 * C_0 = \begin{Bmatrix} 011 \\ 000 \\ 11x \\ 10x \end{Bmatrix} * \begin{Bmatrix} 011 \\ 000 \\ 11x \\ 10x \end{Bmatrix}$$

Để tiện quan sát ta lập bảng như trên hình 2.7.3.22:

$C_0 * C_0$	011	000	11x
011	\	\	\
000	0yy (Ø)	\	\
11x	y11 (x11)	yy0 (Ø)	\
10x	yy1 (Ø)	y00 (x00)	1yx (1xx)

Hình 2.7.3.22

Vậy: $C_0 * C_0 = \begin{Bmatrix} x11 \\ x00 \\ 1xx \end{Bmatrix}$

$$\hat{C}_1 = C_0 \cup (C_0 * C_0) = \begin{Bmatrix} 011 & x11 \\ 000 & x00 \\ 11x & 1xx \\ 10x & \end{Bmatrix}, \quad \hat{C}_1^0 = \begin{Bmatrix} 011 \\ 000 \end{Bmatrix},$$

$$\hat{C}_1^* = \begin{Bmatrix} 11x \\ 10x \end{Bmatrix}, \quad C_1 = \hat{C}_1 \setminus \hat{C}_1^0 \setminus \hat{C}_1^* = \begin{Bmatrix} x11 \\ x00 \\ 1xx \end{Bmatrix}$$

– Giai đoạn 4:

Tìm tập hợp các implicant đơn giản 1 chiều là tập hợp các khối 1: C^1 sao cho $C^1 * C_1$ không chứa bất kỳ khối 2 nào của K(f).

Tức là: $Z^1 = \{C^1 | C^1 * C_1$ không chứa khối 2 nào của K$\}$

Chứng minh giống trường hợp Z^0.

Ví dụ 2.7.3.27. Cho phủ C_1 như sau:

$$C_1 = \begin{Bmatrix} x11 \\ x00 \\ 1xx \end{Bmatrix}$$

Hãy tìm Z^1?

Giải: Ta ký hiệu: $C_1^1 = (x11), \; C_2^1 = (x00)$

$$\Rightarrow C_1^1 * C_1 = (x11) * \begin{Bmatrix} x11 \\ x00 \\ 1xx \end{Bmatrix} = \begin{Bmatrix} \varnothing \\ 111 \end{Bmatrix}$$

Ta thấy $C_1^1 * C_1$ không chứa bất kỳ khối 2 nào.

$\Rightarrow C_1^1$ là implicant đơn giản một chiều.

$$\Rightarrow C_2^1 * C_1 = (x00) * \begin{Bmatrix} x11 \\ 1xx \end{Bmatrix} = \begin{Bmatrix} \varnothing \\ 100 \end{Bmatrix}$$

Ta thấy $C_2^1 * C_1$ không chứa bất kỳ khối 2 nào.

$\Rightarrow C_2^1$ là implicant đơn giản một chiều.

Vậy cuối cùng ta có:

$$Z^1 = \begin{Bmatrix} x11 \\ x00 \end{Bmatrix}$$

– Giai đoạn 5:

Chúng ta xây dựng phủ mới C_2 không chứa các khối 0 và khối 1 và các khối nằm trong các khối khác.

+ Trước hết ta xây dựng phủ $\hat{C}_2$ như sau: $\hat{C}_2 = C_1 \cup (C_1 * C_1)$

Ở đây $C_1 * C_1$ chứa tất cả các khối không nằm trong C_1.

+ Sau đó xây dựng C_2 bằng cách loại bỏ các khối chứa trong khối khác của phủ $\hat{C}_2$, loại bỏ các khối 0 và khối 1.

Ký hiệu:

$\hat{C}_2^0$: tập hợp tất cả các khối 0 của $\hat{C}_2$.

$\hat{C}_2^1$: tập hợp tất cả các khối 1 của $\hat{C}_2$.

$\hat{C}_2^*$: tập hợp tất cả các khối chứa trong khối khác.

$$\hat{C}_2^* = \{c \mid c \subset d; c, d \subset \hat{C}_2\}$$

Vậy: $C_2 = \hat{C}_2 \setminus \hat{C}_2^0 \setminus \hat{C}_2^1 \setminus \hat{C}_2^*$.

Ví dụ 2.7.3.28. Cho phủ C_1 như sau:

$$C_1 = \begin{Bmatrix} x11 \\ x00 \\ 1xx \end{Bmatrix}$$

Hãy tìm C_2?

Giải: Để tìm $C_1 * C_1$ ta dùng bảng như trên hình 2.7.3.23:

$C_1 * C_1$	x11	x00
x11	\	\
x00	xyy (Ø)	\
1xx	111	100

Hình 2.7.3.23

Vậy: $C_1 * C_1 = \begin{Bmatrix} 111 \\ 100 \end{Bmatrix}$; $\hat{C}_2 = C_1 \cup (C_1 * C_1) = \begin{Bmatrix} x11 \\ x00 \\ 1xx \\ 111 \\ 100 \end{Bmatrix}$

Ở đây: $\hat{C}_2^0 = \begin{Bmatrix} 111 \\ 100 \end{Bmatrix}$, $\hat{C}_2^1 = \begin{Bmatrix} x11 \\ x00 \end{Bmatrix}$, $\hat{C}_2^* = Ø$.

$C_2 = \hat{C}_2 \setminus \hat{C}_2^0 \setminus \hat{C}_2^1 \setminus \hat{C}_2^* = \{1xx\}$.

– Giai đoạn 6:

Tìm tập hợp các implicant đơn giản 2 chiều Z^2. Chúng ta có thể tìm Z^2 nhờ bổ đề sau:

Bổ đề: Tập hợp các implicant đơn giản hai chiều là tập hợp các khối 2: C^2, sao cho C^2*C_2 không chứa bất kỳ khối 3 nào của K.

Tức là: $Z^2 = \{C^2 | C^2*C_2$ không chứa khối 3 nào của K$\}$

Chứng minh giống trường hợp Z^0.

Ví dụ 2.7.3.29. Cho phủ C_2 như sau:

$$C_2 = \{1xx\}$$

Hãy tìm Z^2?

Giải: Ta có: $C^2 = \{1xx\}$

$C^2*C_2 = \{1xx\}*\{1xx\} = \{1xx\}$

C^2*C_2 không chứa bất kỳ khối 3 nào nên C^2 là implicant đơn giản hai chiều.

Vậy: $Z^2 = \{1xx\}$.

– Giai đoạn 7:

Chúng ta xây dựng phủ mới C_3 không chứa các khối 0, khối 1, khối 2 và các khối nằm trong các khối khác.

+ Trước hết xây dựng $\hat{C}_3$ như sau: $\hat{C}_3 = C_2 \cup (C_2*C_2)$

Ở đây C_2*C_2 chứa tất cả các khối trừ những khối nằm trong C_2.

+ Sau đó xây dựng C_3 bằng cách loại bỏ các khối 0, khối 1, khối 2 và các khối nằm trong các khối khác của $\hat{C}_3$.

Ký hiệu:

$\hat{C}_3^0$: tập hợp tất cả các khối 0 của $\hat{C}_3$.

$\hat{C}_3^1$: tập hợp tất cả các khối 1 của $\hat{C}_3$.

$\hat{C}_3^*$: tập hợp tất cả các khối chứa trong khối khác.

$$\hat{C}_3^* = \{c \mid c \subset d; c, d \subset \hat{C}_3\}$$

Vậy: $C_3 = \hat{C}_3 \setminus \hat{C}_3^0 \setminus \hat{C}_3^1 \setminus \hat{C}_3^2 \setminus \hat{C}_3^*$.

Ví dụ 2.7.3.30. Cho phủ C_2 như sau:

$$C_2 = \{1xx\}$$

Hãy tìm C_3?

Giải: $C_2*C_2 = \{1xx\}*\{1xx\} = \{1xx\}$

$\Rightarrow \hat{C}_3 = \{1xx\}$, $\hat{C}_3^0 = \varnothing$, $\hat{C}_3^1 = \varnothing$, $\hat{C}_3^2 = \{1xx\}$, $\hat{C}_3^* = \varnothing$

Vậy: $C_3 = \hat{C}_3 \setminus \hat{C}_3^0 \setminus \hat{C}_3^1 \setminus \hat{C}_3^2 \setminus \hat{C}_3^* = \varnothing$.

Tương tự Z^1 và Z^2 tiếp tục tìm Z^3 ta có $Z^3 = \varnothing$.

Nhận xét:

– Quá trình tìm Z kết thúc ở đây và tập hợp các ví dụ trên ta có:

$$Z = Z_1 \cup Z_2 = \begin{Bmatrix} x11 \\ x00 \\ 1xx \end{Bmatrix}$$

– Tổng quát, chúng ta sẽ tìm các giai đoạn cho đến khi $C_{r+1} = Ø$ và ta có:

$$Z = \bigcup_{i=0}^{r} Z^i$$

2.7.3.4. Thái cực

a) Định nghĩa

– Implicant đơn giản z của phức hợp khối K được gọi là thái cực L nếu nó chứa đỉnh d của L và đỉnh d không nằm trong implicant đơn giản khác nào đó của K (tức là d chỉ nằm trong z).

Đỉnh d được gọi là đỉnh đánh dấu.

– Tập hợp tất cả các thái cực L được ký hiệu là: E(K,L).

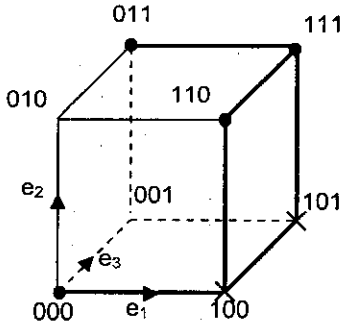
Nhận xét: $E(K,L) \subseteq Z$.

Ví dụ 2.7.3.31. Cho phủ ban đầu C_0 như sau:

$$C_0 = \begin{Bmatrix} 011 \\ 000 \\ 11x \\ 10x \end{Bmatrix}, \text{ trong đó: } L = \begin{Bmatrix} 011 \\ 000 \\ 11x \end{Bmatrix}, \; N = \{10x\}$$

Hãy tìm Z = ?, E(K,L) = ?

Giải:



Hình 2.7.3.24. Dạng khối

– Trong phần trước chúng ta đã tìm được:

$$Z = \begin{Bmatrix} x11 \\ x00 \\ 1xx \end{Bmatrix}$$

Ta ký hiệu: $z_1^1 = x11$, $z_2^1 = x00$, $z_1^2 = 1xx$.

– Bây giờ chúng ta kiểm tra xem các implicant đơn giản này có chứa các đỉnh đánh dấu hay không.

– Trước hết chúng ta kiểm tra các đỉnh (khối 0) trên hình 2.7.3.24 xem có phải là các đỉnh đánh dấu hay không:

+ Đỉnh $011 \subset L$ và đỉnh 011 chỉ nằm trong implicant đơn giản x11 chứ không nằm trong implicant đơn giản khác. Vậy đỉnh 011 là đỉnh đánh dấu và implicant đơn giản $z_1^1 = x11$ là thái cực L.

+ Đỉnh 000 $\subset$ L và đỉnh 000 chỉ nằm trong implicant đơn giản x00 chứ không nằm trong implicant đơn giản khác. Vậy đỉnh 000 là đỉnh đánh dấu và implicant đơn giản $z_2^1 = x00$ là thái cực L.

+ Đỉnh 110 $\subset$ L và đỉnh 110 chỉ nằm trong implicant đơn giản 1xx chứ không nằm trong implicant đơn giản khác. Vậy đỉnh 110 là đỉnh đánh dấu và implicant đơn giản $z_1^2 = 1xx$ là thái cực L.

+ Đỉnh 111 $\subset$ L nhưng nằm trong 2 implicant đơn giản x11 và 1xx nên đỉnh 111 không phải là đỉnh đánh dấu.

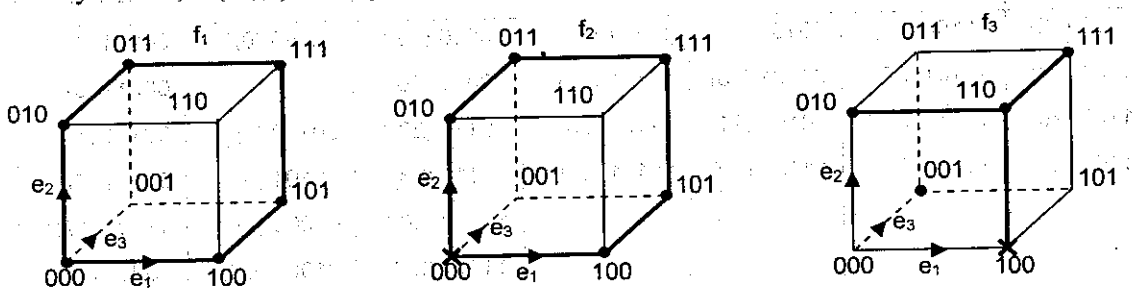
+ Đỉnh 100 không thuộc L và nằm trong 2 implicant đơn giản x00 và 1xx nên đỉnh 100 không phải là đỉnh đánh dấu.

+ Đỉnh 101 không thuộc L và nằm trong implicant đơn giản 1xx nên đỉnh 101 không phải là đỉnh đánh dấu.

Vậy: E(K,L) ≡ Z.

Ví dụ 2.7.3.32. Cho các hàm 3 biến f_1, f_2 và f_3 được biểu diễn dạng hình học như trên hình 2.7.3.25:

Hãy tìm Z, E(K,L) tương ứng với các hàm?



Hình 2.7.3.25. Dạng khối của các hàm f_1, f_2, f_3

Giải:

+ Với f_1:

$$Z = \begin{Bmatrix} x11 \\ 1x1 \\ 10x \\ x00 \\ 0x0 \\ 01x \end{Bmatrix}, \quad E(KL) = \emptyset;$$

+ Với f_2:

$$Z = \begin{Bmatrix} x11 \\ 1x1 \\ 10x \\ x00 \\ 0x0 \\ 01x \end{Bmatrix}, \quad E(KL) = \emptyset;$$

+ Với f_3:

$$Z = \begin{Bmatrix} 001 \\ 1x0 \\ x10 \\ 11x \end{Bmatrix}, \quad E(KL) = \begin{Bmatrix} 001 \\ x10 \\ 11x \end{Bmatrix}$$

Ví dụ 2.7.3.33. Cho K = L, N = Ø

$$Z = \begin{Bmatrix} x110x & 111xx & xx101 & 1x1x1 \\ 1xx1x & 0xx0x & x0xx1 & \end{Bmatrix}$$

Hãy tìm đỉnh đánh dấu và tập E(K, L)?

Giải: Để tìm các đỉnh đánh dấu chúng ta phải viết các implicant đơn giản thành các khối 0 theo bảng như trên hình 2.7.3.26:

Trong bảng trên hình 2.7.3.26 để dễ quan sát ta ghi các giá trị thập phân tương ứng với các tổ hợp nhị phân. Ta nhận thấy có các tổ hợp chỉ nằm trong một implicant đơn giản, trong bảng trên đó là các tổ hợp được đánh dấu "*".

Như vậy, ta có:

Implicant đơn giản $z_1^3 = 1xx1x$ chứa bốn đỉnh đánh dấu: 10010, 10110, 11010, 11011.

Implicant đơn giản $z_2^3 = 0xx0x$ chứa bốn đỉnh đánh dấu: 00000, 00100, 01000, 01001.

Implicant đơn giản $z_3^3 = x0xx1$ chứa ba đỉnh đánh dấu: 00011, 00111, 10001.

Vậy các implicant đơn giản z_1^3, z_2^3, z_3^3 là các thái cực L.

$$E(K,L) = \begin{Bmatrix} 1xx1x \\ 0xx0x \\ x0xx1 \end{Bmatrix}$$

x110x	111xx	xx101	1x1x1	1xx1x	0xx0x	x0xx1
01100 (12)	11100 (28)	00101 (5)	10101 (21)	10010 (18)*	00000 (0)*	00001 (1)
01101 (13)	11101 (29)	01101 (13)	10111 (23)	10011 (19)	00001 (1)	00011 (3)*
11100 (28)	11110 (30)	10101 (21)	11101 (29)	10110 (22)*	00100 (4)*	00101 (5)
11101 (29)	11111 (31)	11101 (29)	11111 (31)	10111 (23)	00101 (5)	00111 (7)*
				11010 (26)*	01000 (8)*	10001 (17)*
				11011 (27)*	01001 (9)*	10011 (19)
				11110 (30)	01100 (12)	10101 (21)
				11111 (31)	01101 (13)	10111 (23)

Hình 2.7.3.26

b) Định lý

Bất kỳ phủ tối thiểu K của phức hợp L nào cũng chứa tập E(K,L): $E(K,L) \subseteq C_{min}^{K,L}$

Chứng minh:

– Theo ví dụ 2.7.3.24 ta có: $C_{min}^{K,L} \subseteq Z$

– Theo định nghĩa ta thấy rằng: $E(K,L) \subseteq Z$

Và bất kỳ thái cực L nào của E(K,L) cũng phải chứa đỉnh đánh dấu d, mà đỉnh đánh dấu là đỉnh không nằm trong các khối khác của Z mà chỉ nằm trong e.

Vậy điều cần thiết là phủ K của phức hợp L phải chứa các đỉnh đánh dấu và $C_{min}^{K,L} \subseteq Z$ mà $E(K,L) \subseteq Z$.

Vậy: $E(K,L) \subseteq C_{min}^{K,L} \subseteq Z$

Nhận xét:

– Theo định lý trên, nếu E(K,L) là phủ, tức là nó chứa tất cả các đỉnh của L, thì E(K,L) sẽ là phủ tối thiểu duy nhất.

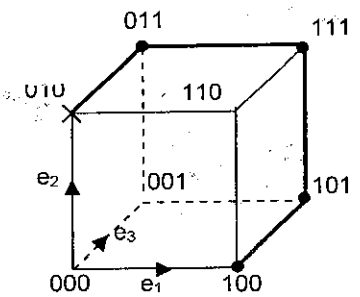
– Nếu phủ ban đầu C_0 được cho dưới dạng tập hợp các đỉnh thì E(K,L) sẽ được tìm một cách dễ dàng, nhưng nếu số đỉnh rất nhiều thì việc kiểm tra các đỉnh sẽ gặp nhiều khó khăn phức tạp và dễ nhầm lẫn. Vì vậy, để tìm E(K,L) chúng ta sử dụng thuật toán giao tọa độ.

Ví dụ 2.7.3.34. Cho hàm logic ba biến sau: $f(x_1,x_2,x_3)=\sum(3,4,5,7)$ với N = 2

– Hãy biểu diễn hàm ở dạng khối?

– Hãy tìm Z, E(K,L), $C_{min}^{K,L}$?

Giải: Hàm được biểu diễn dưới dạng khối trên hình 2.7.3.27.



$$L^0=\begin{Bmatrix}011\\111\\101\\100\end{Bmatrix},\ N^0=\{010\},\ K^0=\begin{Bmatrix}011\\111\\101\\100\\010\end{Bmatrix}$$

Hình 2.7.3.27. Dạng khối của hàm

$$Z=\begin{Bmatrix}01x\\x11\\1x1\\10x\end{Bmatrix},\ E(K,L)=\{10x\}$$ do có chứa đỉnh đánh dấu là (100), $C_{min}^{K,L}=\{x11,10x\}$

2.7.3.5. Phép toán giao tọa độ

Mục đích: Để xác định tập E(K,L) bằng đại số, chúng ta dùng phép toán giao tọa độ.

a) Định nghĩa

Phép toán giao tọa độ được định nghĩa bằng bảng cho trên hình 2.7.3.28:

∩	0	1	x
0	0	∅	0
1	∅	1	1
x	0	1	x

Hình 2.7.3.28. Phép giao toạ độ

Tức là: $0\cap 0=0$ $1\cap 0=\varnothing$ $0\cap 1=\varnothing$

$1\cap x=1$ $0\cap x=0$ $1\cap 1=1$ $x\cap x=x$

b) Phép giao tọa độ của hai khối

– Định nghĩa: Giả sử có hai khối C^r và C^s

$C^r = (a_1 a_2 \ldots a_i \ldots a_n)$

$C^s = (b_1 b_2 \ldots b_i \ldots b_n)$

Phép giao tọa độ giữa hai khối:

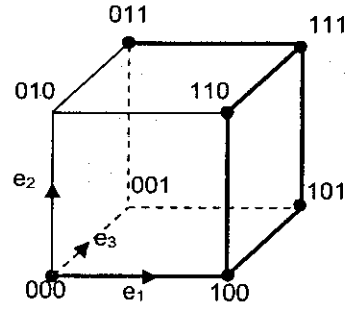
$$C^r * C^s = \begin{cases} - \text{ Ø nếu có dù chỉ một tọa độ } a_i \cap b_i = \varnothing \\ - [(a_1 \cap b_1), (a_2 \cap b_2), \ldots, (a_i \cap b_i), \ldots, (a_n \cap b_n)] \text{ trong trường hợp ngược lại.} \end{cases}$$

Ví dụ 2.7.3.35. Cho hàm logic ba biến như sau: $f(x_1, x_2, x_3) = \sum(0,3,4,5,6,7)$

– Hãy tìm Z?

– Hãy tìm $z_i \cap z_j$?

Giải: Từ dạng khối của hàm được biểu diễn trên hình 2.7.3.29, ta có:



$$Z = \begin{Bmatrix} x00 \\ x11 \\ 1xx \end{Bmatrix}$$

Hình 2.7.3.29. Dạng khối của hàm

Ta ký hiệu: $z_1^1 = (x00); z_2^1 = (x11), z_1^2 = (1xx)$

Vậy: $z_1^1 \cap z_2^1 = (x00) \cap (x11) = (x\varnothing\varnothing) = \varnothing$

$z_1^1 \cap z_1^2 = (x00) \cap (1xx) = (100)$

$z_2^1 \cap z_1^2 = (x11) \cap (1xx) = (111)$

– Các tính chất của phép giao:

+ $C^r \cap C^s = C^r$ nếu $C^r \subset C^s$

+ $C^r \cap C^s = C^s \cap C^r$

+ $C^r \cap C^s = \begin{cases} \subseteq C^r \\ \subseteq C^s \end{cases}$

+ $C^r \cap C^s \cap C^t = C^r \cap C^t \cap C^s$

c) Lân cận z

**Định nghĩa:*

– Ký hiệu lân cận z: U(z,Z)

– Định nghĩa: Lân cận z được định nghĩa bởi quan hệ sau đây:

$U(z,Z) = \{z' | z' \subset Z$ và $z' \cap z \neq \varnothing\}$ (có nghĩa là z' gần với z).

Ví dụ 2.7.3.36.

Cho $Z=\begin{Bmatrix} x00 \\ x11 \\ 1xx \end{Bmatrix}$ hãy tìm $U(z_i,Z)$?

Giải: Ta ký hiệu: $z_1^1=(x00); z_2^1=(x11), z_1^2=(1xx)$

+ Tìm $U(z_1^1,Z)$:

$z_2^1 \cap z_1^1 = \emptyset \Rightarrow z_2^1 \not\subset U(z_1^1,Z)$

$z_1^2 \cap z_1^1 = 100 \neq \emptyset \Rightarrow z_1^2 \subset U(z_1^1,Z) \Rightarrow U(z_1^1,Z)=\begin{Bmatrix} x00 \\ 1xx \end{Bmatrix}$

+ Tìm $U(z_2^1,Z)$:

$z_1^1 \cap z_2^1 = \emptyset \Rightarrow z_1^1 \not\subset U(z_2^1,Z)$

$z_1^2 \cap z_2^1 = 111 \neq \emptyset \Rightarrow z_1^2 \subset U(z_2^1,Z) \Rightarrow U(z_2^1,Z)=\begin{Bmatrix} x11 \\ 1xx \end{Bmatrix}$

+ Tìm $U(z_1^2,Z)$:

$z_1^1 \cap z_1^2 = 100 \neq \emptyset \Rightarrow z_1^1 \subset U(z_1^2,Z)$

$z_2^1 \cap z_1^2 = 111 \neq \emptyset \Rightarrow z_1^2 \subset U(z_2^1,Z)$

$\Rightarrow U(z_1^2,Z)=\begin{Bmatrix} x11 \\ x00 \\ 1xx \end{Bmatrix}$

*Định nghĩa lân cận z bỏ z:

– Ký hiệu lân cận z bỏ z: $U^{'}(z,Z)$

– Định nghĩa: $U^{'}(z,Z) = U(z,Z) \setminus z$.

Ví dụ 2.7.3.37.

Cho $Z=\begin{Bmatrix} x00 \\ x11 \\ 1xx \end{Bmatrix}$ hãy tìm $U^{'}(z_i,Z)$?

Giải: Ta ký hiệu: $z_1^1=(x00); z_2^1=(x11), z_1^2=(1xx)$

$U^{'}(z_1^1,Z) = U(z_1^1,Z) \setminus z_1^1 = \{1xx\}$

$U^{'}(z_2^1,Z) = U(z_2^1,Z) \setminus z_2^1 = \{1xx\}$

$U^{'}(z_1^2,Z) = U(z_1^2,Z) \setminus z_1^2 = \begin{Bmatrix} x00 \\ x11 \end{Bmatrix}$

d) Định lý

Để tìm tập hợp các thái cực, chúng ta sử dụng định lý sau đây:

Định lý: "Implicant đơn giản e là thái cực L nếu và chỉ nếu nó thoả mãn hai điều kiện sau đây:

$K[e \cap L] \neq \varnothing$ và $K[e \cap L] \neq K[e \cap L \cap U'(e,Z)]$"

Chứng minh:

– Giả sử e là thái cực L, thì e có ít nhất một đỉnh đánh dấu d và $K[e \cap L] \neq \varnothing$.

– Theo định nghĩa d không nằm trong một khối nào khác của Z, vì thế $K[e \cap U'(e,Z) \cap Z]$ không chứa đỉnh d, nhưng $K[e \cap L]$ lại chứa đỉnh d vậy $K[e \cap L] \neq K[e \cap U'(e,Z) \cap Z]$.

– Bây giờ chúng ta giả sử rằng: $K[e \cap L] \neq K[e \cap U'(e,Z) \cap Z]$ và $K[e \cap L] \neq \varnothing$ thì $K[e \cap L] \neq K[U'(e,Z)]$

Bởi vì nếu $K[e \cap L] \subseteq K[U'(e,Z)]$ thì $K[e \cap L] = K[e \cap U'(e,Z) \cap L]$ vì $K[e \cap L] \neq K[U'(e,Z)]$.

Có nghĩa là có tối thiểu một đỉnh d chứa trong $K[e \cap L]$ và không chứa trong $K[e \cap U'(e,Z) \cap L]$, đỉnh d này không nằm trong bất kỳ implicant đơn giản nào khác, vậy e là thái cực.

Nhận xét:

– Định lý trên cho phép chúng ta tìm E(K,L) đối với hàm logic cho ở bất kỳ dạng nào mà không cần phải đưa về dạng chuẩn, tức là không cần đưa về khối 0.

– Nếu E(K,L) là $C^{K,L}$, thì E(K,L) là $C^{K,L}_{min}$ và bài toán phủ đã giải xong.

– Nhưng E(K,L) không phải luôn luôn là $C^{K,L}_{min}$ mà chỉ là một phần của $C^{K,L}_{min}$ ($E(K,L) \subset C^{K,L}_{min}$) thì lúc đó chúng ta sẽ tìm tiếp tục theo quá trình lặp, chúng ta đánh số quá trình lặp theo các bước lặp.

– Để tìm $C^{K,L}_{min}$ ta đánh số tập K, L, Z, E(K,L) theo thứ tự các bước sau:

Bước 1: $K_1, L_1, Z_1, E_1(K_1,L_1)$

Bước 2: $K_2, L_2, Z_2, E_2(K_1,L_1)$

.....

Ví dụ 2.7.3.38.

Cho $Z = \begin{Bmatrix} x110x \\ 111xx \\ xx101 \\ 1x1x1 \\ 1xx1x \\ 0xx0x \\ x0xx1 \end{Bmatrix} = Z_1$, $L = \begin{Bmatrix} x1101 \\ 00x11 \\ 00000 \\ 10010 \\ 10x01 \end{Bmatrix} = L_1$. Hãy tìm $U'(z,Z)$ và E(K,L)?

Giải:

+ Để tính U (z,Z) với mọi z ta dùng bảng trên hình 2.7.3.30:

Từ bảng trên hình 2.7.3.30, ta có:

$U(z_1^2,Z) = \{ z_2^2, z_3^2, z_4^2, z_2^3 \}$

$U(z_2^2,Z) = \{ z_1^2, z_3^2, z_4^2, z_1^3 \}$

$U(z_3^2,Z) = \{ z_1^2, z_2^2, z_4^2, z_2^3, z_3^3 \}$

$U(z_4^2,Z) = \{ z_1^2, z_2^2, z_2^3, z_1^3, z_3^3 \}$

$U(z_1^3,Z) = \{ z_2^2, z_2^3, z_3^3 \}$

$U(z_2^3,Z) = \{ z_1^2, z_2^3, z_3^3 \}$

$U(z_3^3,Z) = \{ z_3^2, z_4^2, z_1^3, z_2^3 \}$

$z \cap z$	z_2^2 =111xx	z_3^2 = xx101	z_4^2 = 1x1x1	z_1^3 = 1xx1x	z_2^3 = 0xx0x	z_3^3 = x0xx1
z_1^2 = x110x	1110x	x1101	11101	Ø	0110x	Ø
z_2^2 = 111xx	\	11101	111x1	1111x	Ø	Ø
z_3^2 = xx101	11101	\	1x101	Ø	0x101	x0101
z_4^2 = 1x1x1	111x1	1x101	\	1x111	Ø	101x1
z_1^3 = 1xx1x	1111x	Ø	1x111	\	Ø	10x11
z_2^3 = 0xx0x	Ø	0x101	Ø	Ø	\	00x01

Hình 2.7.3.30

+ Để tìm K[e ∩ L] ta dùng bảng như trên hình 2.7.3.31:

e ∩ L	x1101	00x11	00000	10010	10x01
z_1^2 = x110x	x1101	Ø	Ø	Ø	Ø
z_2^2 = 111xx	11101	Ø	Ø	Ø	Ø
z_3^2 = xx101	x1101	Ø	Ø	Ø	10101
z_4^2 = 1x1x1	11101	Ø	Ø	Ø	10101
z_1^3 = 1xx1x	Ø	Ø	Ø	10010	Ø
z_2^3 = 0xx0x	01101	Ø	00000	Ø	Ø
z_3^3 = x0xx1	Ø	00x11	Ø	Ø	10x01

Hình 2.7.3.31

Từ bảng trên hình 2.7.3.31, ta có:

$K[z_1^2 \cap L] = \{x1101\}$; $K[z_2^2 \cap L] = \{11101\}$; $K[z_3^2 \cap L] = \{x1101, 10101\}$

$K[z_4^2 \cap L] = \{11101, 101011\}$; $K[z_1^3 \cap L] = \{10010\}$; $K[z_2^3 \cap L] = \{00000\}$

$K[z_3^3 \cap L] = \{00x11, 10x01\}$

+ Bây giờ ta tìm $K[z \cap L \cap U'(z, Z)]$

$K[z_1^2 \cap L \cap U'(z_1^2, Z)] = K[x1101] = K[z_1^2 \cap L] \Rightarrow z_1^2 \not\subset E_1(K_1, L_1)$

$K[z_2^2 \cap L \cap U'(z_2^2, Z)] = K[11101] = K[z_2^2 \cap L] \Rightarrow z_2^2 \not\subset E_1(K_1, L_1)$

$K[z_3^2 \cap L \cap U'(z_3^2, Z)] = K[x1101, 10101] = K[z_3^2 \cap L] \Rightarrow z_3^2 \not\subset E_1(K_1, L_1)$

$K[z_3^2 \cap L \cap U'(z_3^2, Z)] = K[x1101, 10101] = K[z_3^2 \cap L] \Rightarrow z_3^2 \not\subset E_1(K_1, L_1)$

$K[z_4^2 \cap L \cap U'(z_4^2, Z)] = K[11101, 10101] = K[z_4^2 \cap L] \Rightarrow z_4^2 \not\subset E_1(K_1, L_1)$

$K[z_1^3 \cap L \cap U'(z_1^3, Z)] = \emptyset \neq K[z_1^3 \cap L] \Rightarrow z_1^3 \subset E_1(K_1, L_1)$

$K[z_2^3 \cap L \cap U'(z_2^3, Z)] = K[01101] \neq K[z_2^3 \cap L] \Rightarrow z_2^3 \subset E_1(K_1, L_1)$

$K[z_3^3 \cap L \cap U'(z_3^3, Z)] = K[10101] \neq K[z_3^3 \cap L] \Rightarrow z_3^3 \subset E_1(K_1, L_1)$

$\Rightarrow E_1(K_1, L_1) = \{z_1^3, z_2^3, z_3^3\} = \{1xx1x, 0xx0x, x0xx1\}$

Nhận xét:

Theo ví dụ trên $E_1(K_1, L_1)$ không phải là phủ của L_1 bởi vì E_1 không phủ tất cả các đỉnh của L_1 và chúng ta phải tìm $E_2(K_2, L_2)$. Trước khi tìm $E_2(K_2, L_2)$ ta phải tìm L_2, tìm L_2 bằng phép hiện tọa độ.

2.7.3.6. Thủ tục tìm E_r và phép hiện tọa độ #

Phần trên chúng ta đã nói về vấn đề tìm E_r bởi vì ta phải làm phép lặp để giải bài toán phủ. Để tìm E_r trước hết ta phải tìm L_r, phép hiện tọa độ sẽ giúp ta giải quyết vấn đề này.

a) Phép hiện tọa độ

**Định nghĩa:*

– Phép hiện tọa độ được ký hiệu là #.

– Phép hiện tọa độ được định nghĩa bằng bảng trên hình 2.7.3.32:

#	0	1	x
0	z	y	z
1	y	z	z
x	1	0	z

Hình 2.7.3.32. Phép hiện tọa độ

Tức là:

0 # 0 = z; 0 # 1 = y; 0 # x = z; 1 # 0 = y; 1 # 1 = z; 1 # x = z;

x # 0 = 1; x # 1 = 0; x # x = z;

b) Định nghĩa hiện tọa độ giữa hai khối

– Giả sử có hai khối C^r và C^s như sau:

$C^r = (a_1, a_2, \ldots, a_i, \ldots, a_n)$

$C^s = (b_1, b_2, \ldots, b_i, \ldots, b_n)$

$C^r \# C^s$: là hiện tọa độ giữa hai khối C^r và C^s.

$$C^r \# C^s = \begin{cases} - C^r \text{ nếu } a_i \# b_i = y \text{ đối với bất kỳ i nào} \\ - \emptyset \text{ nếu } a_i \# b_i = z \text{ với mọi i} \\ - \bigcup_{\alpha_i} (a_1, a_2, \ldots, a_{i-1}, \alpha_i, a_i + 1, \ldots, a_n) \text{ ở đây } a_i \# b_i = \alpha_i = 0 \text{ hoặc } 1. \end{cases}$$

Ví dụ 2.7.3.39. Giả sử cho hai khối C^r và C^s như sau:

$C^r = xxx$ và $C^s = 1x1$

– Hãy tính $C^r \# C^s$?

– Hãy minh họa bằng biểu diễn ở dạng khối?

Giải:

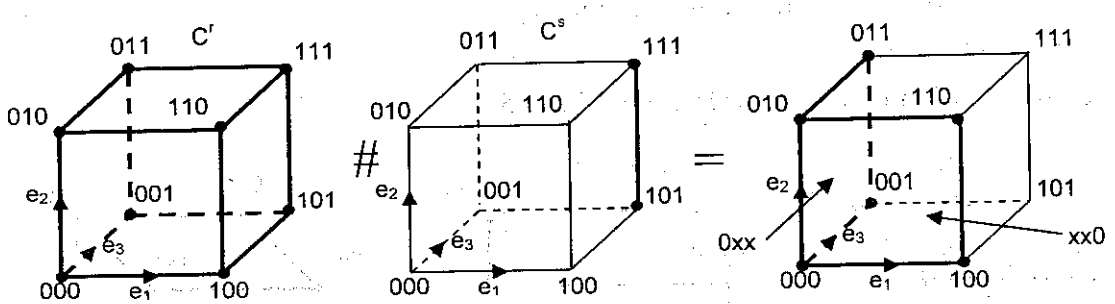
$C^r = xxx \Rightarrow a_1 = x, a_2 = x, a_3 = x$

$C^s = 1x1 \Rightarrow b_1 = 1, b_2 = x, b_3 = 1$

$C^r \# C^s = xxx \# 1x1 = 0z0 = \alpha_1 z\, \alpha_3.$

$\Rightarrow C^r \# C^s = 0a_1a_2 \cup a_1a_20 = 0xx \cup xx0.$

Ta có biểu diễn dạng khối trên hình 2.7.3.33.



Hình 2.7.3.33. Dạng khối

Ví dụ 2.7.3.40. Giả sử cho hai khối C^r và C^s như sau:

$C^r = xx0$ và $C^s = 1xx$

– Hãy tính $C^r \# C^s$?

– Hãy minh họa bằng biểu diễn ở dạng khối?

Giải:

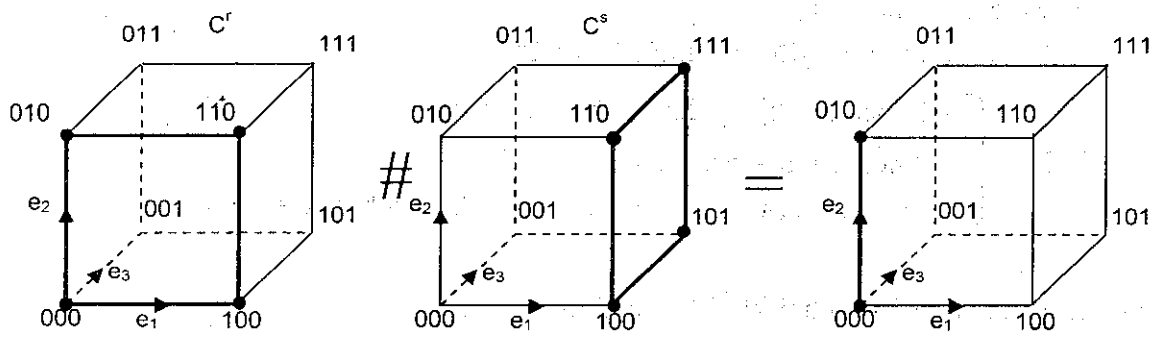
$C^r = xx0 \Rightarrow a_1 = x, a_2 = x, a_3 = 0$

$C^s = 1xx \Rightarrow b_1 = 1, b_2 = x, b_3 = x$

$C^r \# C^s = xx0 \# 1x0 = 0zz = \alpha_1 z\ z$

$\Rightarrow C^r \# C^s = 0a_2a_3 = 0x0.$

Ta có biểu diễn dạng khối trên hình 2.7.3.34.



Hình 2.7.3.34. Dạng khối

Ví dụ 2.7.3.41. Giả sử cho hai khối C^r và C^s như sau:

$C^r = x10$ và $C^s = 1x1$

– Hãy tính $C^r \# C^s$?

– Hãy minh họa bằng biểu diễn ở dạng khối?

Giải:

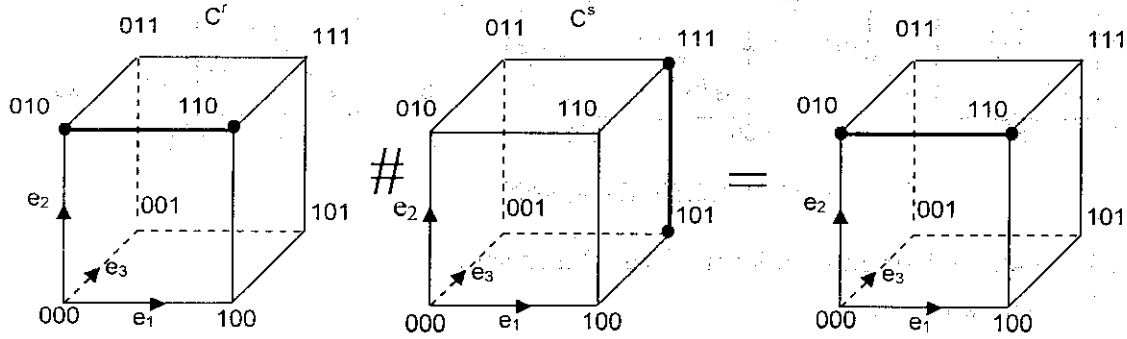
$C^r = x10 \Rightarrow a_1 = x, a_2 = 1, a_3 = 0$

$C^s = 1x1 \Rightarrow b_1 = 1, b_2 = x, b_3 = 1$

$C^r \# C^s = x10 \# 1x1 = 0zy$ (ở đây có một y)

$\Rightarrow C^r \# C^s = x10.$

Ta có biểu diễn dạng khối trên hình 2.7.3.35.



Hình 2.7.3.35. Dạng khối

khối C^r và C^s như sau:

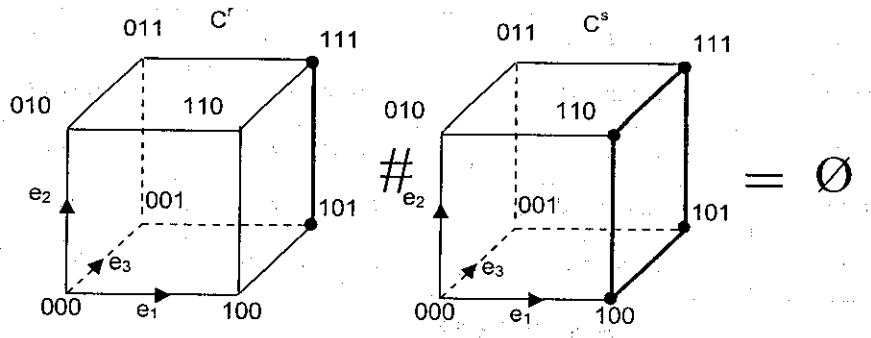
diễn ở dạng khối?

$a_2 = x, a_3 = 1$

$1, b_2 = x, b_3 = x$

$\# 1xx = zzz \Rightarrow C^r \# C^s = \emptyset$

diễn dạng khối trên hình 2.7.3.36.



Hình 2.7.3.36. Dạng khối

– Các tính chất của #

+ $C^r \# C^s = C^r$ nếu $C^r \cap C^s = \emptyset$

+ $C^r \# C^s \subseteq C^r$ và $C^r \# C^s \not\subset C^s$

+ $(C^r \# C^s) \# C^t \neq C^r \# (C^s \# C^t)$ (không có tính chất kết hợp)

+ $C^r \# C^s \neq C^s \# C^r$ (không có tính chất giao hoán)

+ $(C^r \cup C^s) \# C^t = (C^r \# C^t) \cup (C^s \# C^t)$ (tính phân phối đối với phép hợp)

+ $(C^r \cap C^s) \# C^t = (C^r \# C^t) \cap (C^s \# C^t)$ (tính phân phối đối với phép giao)

+ $(C^r \# C^s) \# C^t = (C^r \# C^t) \# C^s$

c) Các bước tìm $E_r(K_r, L_r)$

Xác định L_r thông qua L_{r-1} và E_{r-1}

Để tìm L_r thông qua L_{r-1} và E_{r-1} chúng ta sử dụng định lý sau:

Định lý: Giả sử L_{r-1} là tập hợp các khối $\{C_1, C_2, \ldots C_p\}$

Giả sử E_{r-1} là tập hợp các khối $\{e_1, e_2, \ldots e_q\}$

Thì L_r được xác định bằng tập hợp các khối sau đây:

$\{[(C_1 \# e_1) \# e_2] \# \ldots \# e_q\} \cup \{[(C_2 \# e_1) \# e_2] \# \ldots \# e_q\} \cup \ldots \cup \{[(C_p \# e_1) \# e_2] \# \ldots \# e_q\}$

$\Rightarrow L_r = \bigcup_{i=1}^{p} (C_i \# E_{r-1})$ hoặc $L_r = L_{r-1} \# E_{r-1}$

Ví dụ 2.7.3.43.

Cho $L_1 = \begin{Bmatrix} x1101 \\ 00x11 \\ 00000 \\ 10010 \\ 10x01 \end{Bmatrix}$, $E_1 = \begin{Bmatrix} 1xx1x \\ 0xx0x \\ x0xx1 \end{Bmatrix}$

Hãy tính $L_2 = L_1 \# E_1$?

Giải: Ta tìm L_2 nhờ bảng trên hình 2.7.3.37:

$L_1 \# E_1$	e_1 = 1xx1x	e_2 = 0xx0x	e_3 = x0xx1
C_1 = x1101	0zzyz = C_1	1zzzz = 11101	zyzzz = x1101
C_2 = x1101	yzzzz = C_2	zzzyz = C_2	zzzzz = Ø
C_3 = x1101	yzzyz = C_3	zzzzz = Ø	zzzzz = Ø
C_4 = x1101	zzzzz = Ø	zzzzz = Ø	zzzzz = Ø
C_5 = x1101	zzzyz = C_5	yzzzz = 10x01	zzzzz = Ø

Hình 2.7.3.37

Ta có: $C_1 \# E_1 = 11101$; $C_2 \# E_1 = Ø$; $C_3 \# E_1 = Ø$; $C_4 \# E_1 = Ø$; $C_5 \# E_1 = Ø$

Vậy: $L_2 = L_1 \# E_1 = \{11101\}$.

d) Tìm E_2

*Xác định Z_2:

Trước hết xây dựng tập $\hat{Z}_2$ là tập hợp các implicant đơn giản không chứa E_1:

$\hat{Z}_2 = Z_1 \setminus E_1$.

Sau đó từ $\hat{Z}_2$ chúng ta loại ra những khối không cực đại của $\hat{Z}_2$.

*Định nghĩa khối không cực đại

Giả sử ta có hai khối u và v; u và v $\subset \hat{Z}_2$.

Ta nói rằng u < v nếu giá của u không thấp hơn giá của v và $u \cap L_2 \subseteq v \cap L_2$, tức là: u < v nếu $G^a[u] \geq G^a[v]$ và $u \cap L_2 \subseteq v \cap L_2$

Nếu u < v và u,v $\subset \hat{Z}_2$ thì ta loại u ra khỏi $\hat{Z}_2$.

*Định lý:

Nếu u và v $\subset Z$ và u < v thì phủ tối thiểu sẽ không chứa u.

Và chúng ta có: $Z_2 = \{v | u < v; u,v \subset \hat{Z}_2\}$

Ví dụ 2.7.3.42. Giả sử cho hai khối C^r và C^s như sau:

C^r = 1x1 và C^s = 1xx

– Hãy tính $C^r \# C^s$?

– Hãy minh họa bằng biểu diễn ở dạng khối?

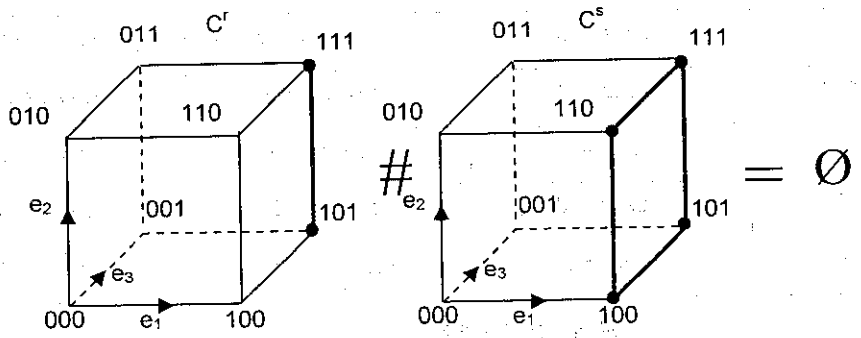
Giải:

C^r = 1x1 $\Rightarrow a_1 = 1, a_2 = x, a_3 = 1$

C^s = 1xx $\Rightarrow b_1 = 1, b_2 = x, b_3 = x$

$C^r \# C^s$ = 1x1 # 1xx = zzz $\Rightarrow C^r \# C^s = \emptyset$

Ta có biểu diễn dạng khối trên hình 2.7.3.36.



Hình 2.7.3.36. Dạng khối

– Các tính chất của #

+ $C^r \# C^s = C^r$ nếu $C^r \cap C^s = \emptyset$

+ $C^r \# C^s \subseteq C^r$ và $C^r \# C^s \not\subset C^s$

+ $(C^r \# C^s) \# C^t \neq C^r \# (C^s \# C^t)$ (không có tính chất kết hợp)

+ $C^r \# C^s \neq C^s \# C^r$ (không có tính chất giao hoán)

+ $(C^r \cup C^s) \# C^t = (C^r \# C^t) \cup (C^s \# C^t)$ (tính phân phối đối với phép hợp)

+ $(C^r \cap C^s) \# C^t = (C^r \# C^t) \cap (C^s \# C^t)$ (tính phân phối đối với phép giao)

+ $(C^r \# C^s) \# C^t = (C^r \# C^t) \# C^s$

c) Các bước tìm $E_r(K_r, L_r)$

Xác định L_r thông qua L_{r-1} và E_{r-1}

Để tìm L_r thông qua L_{r-1} và E_{r-1} chúng ta sử dụng định lý sau:

Định lý: Giả sử L_{r-1} là tập hợp các khối $\{C_1, C_2, \ldots, C_p\}$

Giả sử E_{r-1} là tập hợp các khối $\{e_1, e_2, \ldots, e_q\}$

Thì L_r được xác định bằng tập hợp các khối sau đây:

$\{[(C_1 \# e_1) \# e_2] \# \ldots \# e_q\} \cup \{[(C_2 \# e_1) \# e_2] \# \ldots \# e_q\} \cup \ldots \cup \{[(C_p \# e_1) \# e_2] \# \ldots \# e_q\}$

$\Rightarrow L_r = \bigcup_{i=1}^{p} (C_i \# E_{r-1})$ hoặc $L_r = L_{r-1} \# E_{r-1}$.

Ví dụ 2.7.3.43.

Cho $L_1 = \begin{Bmatrix} x1101 \\ 00x11 \\ 00000 \\ 10010 \\ 10x01 \end{Bmatrix}$, $E_1 = \begin{Bmatrix} 1xx1x \\ 0xx0x \\ x0xx1 \end{Bmatrix}$

Hãy tính $L_2 = L_1 \# E_1$?

Giải: Ta tìm L_2 nhờ bảng trên hình 2.7.3.37:

$L_1 \# E_1$	e_1 = 1xx1x	e_2 = 0xx0x	e_3 = x0xx1
C_1 = x1101	0zzyz = C_1	1zzzz = 11101	zyzzz = x1101
C_2 = x1101	yzzzz = C_2	zzzyz = C_2	zzzzz = Ø
C_3 = x1101	yzzyz = C_3	zzzzz = Ø	zzzzz = Ø
C_4 = x1101	zzzzz = Ø	zzzzz = Ø	zzzzz = Ø
C_5 = x1101	zzzyz = C_5	yzzzz = 10x01	zzzzz = Ø

Hình 2.7.3.37

Ta có: $C_1 \# E_1$ = 11101; $C_2 \# E_1 = Ø$; $C_3 \# E_1 = Ø$; $C_4 \# E_1 = Ø$; $C_5 \# E_1 = Ø$

Vậy: $L_2 = L_1 \# E_1 = \{11101\}$.

d) Tìm E_2

*Xác định Z_2:

Trước hết xây dựng tập $\hat{Z}_2$ là tập hợp các implicant đơn giản không chứa E_1:

$\hat{Z}_2 = Z_1 \setminus E_1$.

Sau đó từ $\hat{Z}_2$ chúng ta loại ra những khối không cực đại của $\hat{Z}_2$.

*Định nghĩa khối không cực đại

Giả sử ta có hai khối u và v; u và v $\subset \hat{Z}_2$.

Ta nói rằng u < v nếu giá của u không thấp hơn giá của v và $u \cap L_2 \subseteq v \cap L_2$, tức là: u < v nếu $G^a[u] \geq G^a[v]$ và $u \cap L_2 \subseteq v \cap L_2$

Nếu u < v và u,v $\subset \hat{Z}_2$ thì ta loại u ra khỏi $\hat{Z}_2$.

*Định lý:

Nếu u và v $\subset Z$ và u < v thì phủ tối thiểu sẽ không chứa u.

Và chúng ta có: $Z_2 = \{v | u < v; u,v \subset \hat{Z}_2\}$

Ví dụ 2.7.3.44.

Cho $Z_1 = \begin{Bmatrix} x110x \\ 111xx \\ xx101 \\ 1x1x1 \\ 1xx1x \\ 0xx0x \\ x0xx1 \end{Bmatrix}$, $E_1 = \begin{Bmatrix} 1xx1x \\ 0xx0x \\ x0xx1 \end{Bmatrix}$, $L_2 = \{11101\}$

Hãy tính Z_2?

Giải:

$$\hat{Z}_2 = Z_1 \setminus E_1 = \begin{Bmatrix} x110x \\ 111xx \\ xx101 \\ 1x1x1 \end{Bmatrix}$$

Ta ký hiệu: $Z_1^2 = x110x$; $Z_2^2 = 111xx$; $Z_3^2 = xx101$; $Z_4^2 = 1x1x1$

$G^a[Z_1^2] = G^a[Z_2^2] = G^a[Z_3^2] = G^a[Z_4^2] = 3$

$Z_1^2 \cap L_2 = Z_2^2 \cap L_2 = Z_3^2 \cap L_2 = Z_4^2 \cap L_2 = 11101$

Vậy: $Z_2 = \{Z_1^2, Z_2^2, Z_3^2, Z_4^2\}$

* Xác định $E_2(K_2,L_2)$

$K[Z_1^2 \cap L_2] = K[Z_2^2 \cap L_2] = K[Z_3^2 \cap L_2] = K[Z_4^2 \cap L_2] = 11101$

Vậy ta chọn một trong 4 implicant đơn giản này để lập $E_2(K_2,L_2)$

Giả sử ta lấy Z_1^2, ta có $E_2(K_2,L_2) = \{Z_1^2\}$

Ta tìm E_3:

$L_3 = L_2 \# E_2 = \{Ø\}$

Vậy: $E(K,L) = E_1 \cup E_2 = \begin{Bmatrix} 1xx1x \\ 0xx0x \\ x0xx1 \\ x110x \end{Bmatrix}$, $C_{min}^{K,L} = E(K,L)$

Tổng kết phương pháp Rot

1.Tìm $Z = Z_1$: Dùng phép tích tọa độ: $Z_1 = \bigcup_{r=0}^{n} Z^r$

2.Tìm $E_1 = E_1(K_1,L_1)$: Dùng phép giao tọa độ.

3.Tìm $\hat{Z}_2$: $\hat{Z}_2 = Z_1 \setminus E_1$

4. Tìm L_2: $L_2 = L_1 \# E_1$

5. Tìm Z_2: $Z_2 = \{v | u < v;\ u,v \subset \hat{Z}_2\}$ và $u < v$ nếu $G^a[u] \geq G^a[v]$ và

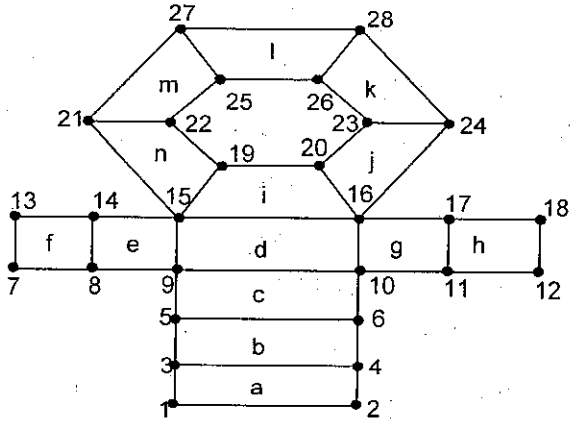
$u \cap L_2 \subseteq v \cap L_2$

6. Tìm E_2 tiếp tục từ bước 2 cho đến khi $L_r = \emptyset$ thì dừng lại.

Ta được: $C_{min}^{K,L} = E_1 \cup E_2 \cup ... \cup E_{r-1}$

Ví dụ 2.7.3.45. Cho $K_1 = L_1 = K\{a,b,c,...,m,n\}$, $N = \emptyset$ được minh họa bằng hình vẽ 2.7.3.38:

Hãy tính $C_{min}^{K,L}$?



Hình 2.7.3.38

Giải:

– $Z = \{a,b,c,...,m,n\} = Z_1$

– Tìm E_1:

a chứa 2 đỉnh đánh dấu: 1,2

f chứa 2 đỉnh đánh dấu: 7,13

h chứa 2 đỉnh đánh dấu: 12,18

Vậy: $E_1 = \{a,f,h\}$

– Tìm $\hat{Z}_2$:

$\hat{Z}_2 = Z_1 \setminus E_1 = \{b,c,d,e,g,i,j,k,l,m,n\}$

– Tìm L_2: $L_2 = L_1 \# E_1 =$

$= \{5,6,9,10,15,16,19,20,21,22,23,24,25,26,27,28\}$

– Tìm Z_2: G^a của các khối $\subset \hat{Z}_2$ đều bằng nhau.

+ $b \cap L_2 = \{5,6\}$

$c \cap L_2 = \{5,6,9,10\}$

Ta thấy: $b \cap L_2 \subset c \cap L_2$ tức $b < c$ vậy ta bỏ b lấy c.

+ $e \cap L_2 = \{9,15\}$

$d \cap L_2 = \{9,15,10,16\}$

Ta thấy: $e \cap L_2 \subset d \cap L_2$ tức $e < d$ vậy ta bỏ e lấy d.

+ $g \cap L_2 = \{10,16\}$

$d \cap L_2 = \{9,15,10,16\}$

Ta thấy: $g \cap L_2 \subset d \cap L_2$ tức $g < d$ vậy ta bỏ g lấy d.

Ta được: $Z_2 = \{c,d,i,j,k,l,m,n\}$

– Tìm E_2: Trong Z_2 c chứa hai đỉnh đánh dấu 5,6 vậy ta có $E_2 = \{c\}$

– Tìm $\hat{Z}_3$: $\hat{Z}_3 = Z_2 \setminus E_2 = \{d,i,j,k,l,m,n\}$

– Tìm L_3: $L_3 = L_2 \# E_2 = \{15,16,19,20,21,22,23,24,25,26,27,28\}$

– Tìm Z_3: G^a của các khối $\subset \hat{Z}_3$ đều bằng nhau.

$d \cap L_3 = \{15,16\}$

$i \cap L_3 = \{15,16,19,20\}$

Ta thấy: d $\cap L_3 \subset$ i $\cap L_3$ tức d < i vậy ta bỏ d lấy i.

Ta được: Z_3 = {i,j,k,l,m,n}

– Tìm E_3: Trong Z_3 ta thấy các khối không chứa đỉnh đánh dấu, vậy để lập tập E_3 chúng ta lấy bất kỳ khối nào của Z_3. Giả sử chúng ta lấy khối i ta có: E_3 = {i}

– Tìm $\hat{Z}_4$: $\hat{Z}_4 = Z_3 \setminus E_3$ = {j,k,l,m,n}

– Tìm L_4: $L_4 = L_3 \# E_3$ = {21,22,23,24,25,26,27,28}

– Tìm Z_4: G^a của các khối $\subset \hat{Z}_4$ đều bằng nhau.

j $\cap L_4$ = {23,24}

k $\cap L_4$ = {23,24,26,28}

Ta thấy: j $\cap L_4 \subset$ k $\cap L_4$ tức j < k vậy ta bỏ j lấy k.

n $\cap L_4$ = {21,22}

m $\cap L_4$ = {21,22,25,27}

Ta thấy: n $\cap L_4 \subset$ m $\cap L_4$ tức n < m vậy ta bỏ n lấy m.

Ta được: Z_4 = {k,l,m}

– Tìm E_4: Trong Z_4 ta thấy k chứa hai đỉnh đánh dấu là 23,24 và m chứa hai đỉnh đánh dấu 21,22.

Vậy: E_4 = {k,m}

– Tìm $\hat{Z}_5$: $\hat{Z}_5 = Z_4 \setminus E_4$ = {l}

– Tìm L_5: $L_5 = L_4 \# E_4$ = {Ø}

Ta được: $C_{min} = E_1 \cup E_2 \cup E_3 \cup E_4 = \{a,f,h,c,i,k,m\}$

2.7.4. Tối thiểu hóa bằng phương pháp Quine – Mc.Cluskey

Phương pháp Quine – Mc.Cluskey phát triển giữa những năm 1950 tìm ra cách tối thiểu hóa của mọi biểu thức Boole.

Quá trình tối thiểu hóa gồm các bước minh họa trên hình 2.7.4.1.

1. Các bước tìm các implicant đơn giản

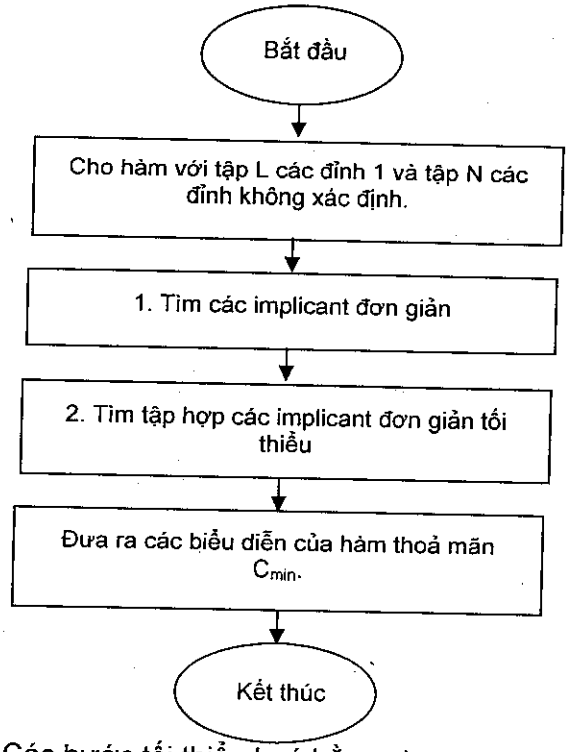
– Biểu diễn các đỉnh 1 và đỉnh không xác định của hàm dưới dạng mã nhị phân (cột 1).

– Sắp xếp các tổ hợp mã trên thành các nhóm theo số lượng chữ số 1 có trong chúng (cột 2).

– So sánh các tổ hợp thuộc nhóm thứ i với từng tổ hợp thuộc nhóm thứ i + 1 (trong cột 2), nếu chúng khác nhau chỉ một bit thì kết hợp 2 tổ hợp đó thành một tổ hợp mới (cột 3), trong đó thay bit khác nhau của 2 tổ hợp đó bằng một gạch ngang (–) đồng thời đánh dấu kiểm soát "γ" vào 2 tổ hợp cũ để xác định đó không phải là implicant đơn giản. Nếu một tổ hợp nào đó không thể kết hợp với tổ hợp khác thì có nghĩa đó là implicant đơn giản và ta đánh dấu kiểm soát "*".

– Loại bớt các tổ hợp giống nhau trong cột 3 và lặp lại bước 3 cho đến khi hết

khả năng kết hợp các tổ hợp với nhau thì thôi, trong các bước này các phần tử có thể tổ hợp với nhau phải khác nhau 1 bit và đồng thời các dấu "–" của chúng phải cùng vị trí. Các tổ hợp cuối cùng được tạo ra là các implicant đơn giản và ta đánh dấu "*". Tập hợp tất cả các tổ hợp được đánh dấu "*" chính là các implicant đơn giản và bước thứ nhất của thuật toán kết thúc ở đây.



Hình 2.7.4.1. Các bước tối thiểu hoá bằng phương pháp Quine – Mc.Cluskey

2. Các bước tìm tập hợp các implicant đơn giản tối thiểu

Trong bước này, ta tìm tập hợp nhỏ nhất các implicant đơn giản sao cho nó bao phủ toàn bộ các đỉnh 1 của hàm. Điều này được thực hiện qua một bảng tổ chức như sau:

– Mỗi cột tương ứng với một đỉnh 1 (các đỉnh không xác định không được dùng đến trong bước này). Mỗi hàng tương ứng với một implicant đơn giản mà ta đã tìm được ở bước thứ nhất của thuật toán. Đánh dấu "x" vào ô (m,n) nếu implicant đơn giản ở hàng thứ m phủ đỉnh 1 ở cột thứ n.

– Xét từng cột, cột nào chỉ có một dấu "x" thì thay bằng dấu "⊗" có nghĩa là implicant đơn giản tương ứng với hàng đó là implicant đơn giản tối thiểu sẽ có mặt trong kết quả cuối cùng. Chúng ta gạch một đường qua cột và hàng mà qua đó ta đã tìm thấy implicant đơn giản tối thiểu (hai đường đó giao nhau tại ô "⊗") . Các implicant đơn giản tối thiểu thường bao phủ các đỉnh 1 khác nữa, chúng ta gạch dọc tất cả các cột mà có một dấu "x" đã bị gạch ngang. Đến đây có thể còn các đỉnh chưa được phủ đó là các đỉnh tương ứng với các dấu "x" chưa được gạch, vì thế chúng ta phải tìm càng ít các implicant đơn giản càng tốt sao cho phủ hết các đỉnh còn lại, các implicant đơn giản đó là các implicant đơn giản tối thiểu.

– Kết quả cuối cùng sẽ là tổng của các implicant đơn giản tối thiểu ta đã tìm được ở trên.

Ví dụ 2.7.4.1. Cho hàm logic bốn biến x_1, x_2, x_3, x_4 như sau:

$$F(x_1, x_2, x_3, x_4) = \sum(1,2,4,5,6,7,8,9) \text{ với } N = 14,15$$

Tối thiểu hóa hàm bằng phương pháp Quine – Mc.Cluskey.

Giải:

1. Quá trình tìm các implicant đơn giản minh họa trong bảng trên hình 2.7.4.2

Cột thứ nhất là các tổ hợp mã nhị phân tương ứng với các đỉnh 1 và không xác định.

Cột 2 sắp xếp các tổ hợp mã nhị phân thành các nhóm có số chữ số 1 giống nhau, sao cho hai nhóm kề nhau có số bit khác nhau ít nhất. Nhóm thứ nhất có 1 chữ số 1 trong các vị trí, nhóm thứ hai có 2 chữ số 1 trong các vị trí,....

Tiếp theo ta so sánh các phần tử trong nhóm thứ nhất với nhóm thứ 2, nhóm thứ 2 với nhóm thứ 3,...các nhóm có thể kết hợp với nhau được ghi rõ trong cột 3, vị trí bit khác nhau được ký hiệu " – ".

Trong cột 3 là các implicant đơn giản được tạo thành, ta tiếp tục so sánh các nhóm trong cột 3 với nhau. Nếu 2 nhóm chỉ khác nhau một bit và cùng vị trí "–" thì được kết hợp với nhau tạo ra nhóm mới, các nhóm có thể kết hợp với nhau được ghi rõ trong cột 4. Các nhóm trong cột 3 không thể kết hợp với nhóm khác và các nhóm ở cột 4 là các implicant đơn giản đều được đánh dấu " * ".

Cột 1		Số chữ số 1	Cột 2		Cột 3		Cột 4	
1	0001	1	1	0001 y	1,5	0 – 01*	4,5,6,7	01––*
2	0010		2	0010 y	1,9	–001*	6,7,14,15	–11–*
4	0100		4	0100 y	2,6	0 – 10*		
5	0101		8	1000 y	4,5	010 – y		
6	0110	2	5	0101 y	4,6	01– 0 y		
7	0111		6	0110 y	8,9	100 –*		
8	1000		9	1001 y	5,7	01–1 y		
9	1001	3	7	0111 y	6,7	011– y		
14	1110		14	1110 y	6,14	– 110 y		
15	1111	4	15	1111 y	7,15	– 111 y		
					14,15	111 – y		

Hình 2.7.4.2. Tìm các implicant đơn giản

Vậy ta nhận được các implicant đơn giản sau:

$0-01$: $\overline{x}_1\overline{x}_3x_4$

-001: $\overline{x}_2\overline{x}_3x_4$

$0-10$: $\overline{x}_1x_3\overline{x}_4$

$100-$: $x_1\overline{x}_2\overline{x}_3$

$01--$: $\overline{x}_1x_2$

$-11-$: x_2x_3

Trong phần này các đỉnh không xác định được coi như các đỉnh 1.

2. Quá trình tìm các implicant đơn giản tối thiểu được minh họa trong bảng hình 2.7.4.3

		1	2	4	5	6	7	8	9
(1,5)	0–01	x			x				
(1,9)	–001	x							x
(2,6)	0–10		⊗			x			
(8,9)	100–							⊗	x
(4,5,6,7)	01––			⊗	x	x	x		
(6,7,14,15)	–11–					x	x		

Hình 2.7.4.3. Tìm các implicant đơn giản tối thiểu

Trong bảng hình 2.7.4.3 những cột chỉ có một dấu "x" được thay bằng "⊗" đó là những implicant đơn giản tối thiểu, những ô có dấu "x" cùng hàng với ô có dấu "⊗" được bỏ đi và những cột có ô "x" được bỏ đi thì tất cả các ô trong cột đều được bỏ đi. Ta nhận thấy còn hai ô "x" tương ứng với đỉnh 1 và tương ứng với 2 implicant đơn giản $0-01$ và -001, hai implicant đơn giản đó chứa đỉnh 1 vì thế ta có thể chọn một trong hai implicant đơn giản đó làm implicant đơn giản tối thiểu (ta thấy khi một implicant được gạch đi thì kéo theo implicant kia cũng được gạch đi).

Kết quả cuối cùng gồm các implicant đơn giản tối thiểu sau:

$0-01$, $0-10$, $100-$, $01--$, hoặc -001, $0-10$, $100-$, $01--$

Vậy: $F = \overline{x}_1\overline{x}_3x_4 + \overline{x}_1x_3\overline{x}_4 + x_1\overline{x}_2\overline{x}_3 + \overline{x}_1x_2$ hoặc $F = \overline{x}_2\overline{x}_3x_4 + \overline{x}_1x_3\overline{x}_4 + x_1\overline{x}_2\overline{x}_3 + \overline{x}_1x_2$

Ví dụ 2.7.4.2. Cho hàm logic bốn biến x_1, x_2, x_3, x_4 như sau:

$F(x_1, x_2, x_3, x_4) = \sum(4,5,6,8,9,10,13)$ với N = 0,7,15

Tối thiểu hóa hàm bằng phương pháp Quine – Mc.Cluskey.

Giải:

1. Quá trình tìm các implicant đơn giản minh họa trong bảng trên hình 2.7.4.4

Cột 1		Số chữ số 1	Cột 2		Cột 3		Cột 4	
0	0000	0	0	0000 γ	0,4	0–00*	4,5,6,7	01––*
4	0100	1	4	0100 γ	0,8	–000*	5,7,13,15	–1–1*
5	0101		8	1000 γ	4,5	010– γ		
6	0110	2	5	0101 γ	4,6	01–0 γ		
7	0111		6	0110 γ	8,9	100–*		
8	1000		9	1001 γ	8,10	10–0*		
9	1001		10	1010 γ	5,7	01–1 γ		
10	1010	3	7	0111 γ	5,13	–101 γ		
13	1101		13	1101 γ	6,7	011– γ		
15	1111	4	15	1111 γ	9,13	1–01*		
					7,15	–111 γ		
					13,15	11–1 γ		

Hình 2.7.4.4. Tìm các implicant đơn giản

2. Quá trình tìm các implicant đơn giản tối thiểu được minh họa trong bảng hình 2.7.4.5

		4	5	6	8	9	10	13
(0,4)	0–00	x						
(0,8)	–000				x			
(8,9)	100–				x	x		
(8,10)	10–0				x		⊗	
(9,13)	1–01					x		x
(4,5,6,7)	01––	x	x	⊗				
(5,7,13,15)	–1–1		x					x

Hình 2.7.4.5. Tìm các implicant đơn giản tối thiểu

Trong bảng trên hình 2.7.4.5 còn hai đỉnh 1 chưa được phủ đó là cột 9 và 13 và ta chọn implicant đơn giản 1 – 01 sẽ phủ hết cả hai đỉnh đó.

Vậy: $F = x_1\overline{x}_2\overline{x}_4 + x_1\overline{x}_3x_4 + \overline{x}_1x_2$

Ví dụ 2.7.4.3. Cho hàm logic 5 biến x_1, x_2, x_3, x_4, x_5 như sau:

$$F(x_1,x_2,x_3,x_4,x_5) = \sum(0,2,8,9,10,11,16,17,18,19)$$

Tối thiểu hóa hàm bằng phương pháp Quine – Mc.Cluskey.

Giải:

1. Quá trình tìm các implicant đơn giản minh họa trong bảng trên hình 2.7.4.6

Cột 1		Số chữ số 1	Cột 2		Cột 3		Cột 4	
0	00000	0	0	00000 γ	0,2	000–0 γ	0,2,16,18	–00–0*
2	00010	1	2	00010 γ	0,8	0–000 γ	0,2,8,10	0–0–0*
8	01000		8	01000 γ	0,16	–0000 γ	8,9,10,11	010––*
9	01001		16	10000 γ	2,10	0–010 γ	16,17,18,19	100––*
10	01010	2	9	01001 γ	2,18	–0010 γ		
11	01011		10	01010 γ	8,9	0100– γ		
16	10000		17	10001 γ	16,17	1000– γ		
17	10001		18	10010 γ	16,18	100–0 γ		
18	10010	3	11	01011 γ	9,11	010–1 *		
19	10011		19	10011 γ	10,11	0101– γ		
					17,19	100–1 γ		
					18,19	1001– γ		

Hình 2.7.4.6

		0	2	8	9	10	11	16	17	18	19
(9,11)	010–1						x				x
(0,2,16,18)	–00–0	x	x					x		x	
(0,2,8,10)	0–0–0	x	x	x		x					
(8,9,10,11)	010––			x	⊗	x	x				
(16,17,18,19)	100––							x	⊗	x	x

Hình 2.7.4.7

2. Quá trình tìm các implicant đơn giản tối thiểu được minh họa trong bảng hình 2.7.4.7

Ta có: $F=\bar{x}_2\bar{x}_3\bar{x}_5+\bar{x}_1x_2\bar{x}_3+x_1\bar{x}_2\bar{x}_3$ hoặc $F=\bar{x}_1\bar{x}_3\bar{x}_5+\bar{x}_1x_2\bar{x}_3+x_1\bar{x}_2\bar{x}_3$

BÀI TẬP CHƯƠNG 2

Bài 2.1. Hãy tìm hàm đảo của các hàm logic dưới đây (dùng định lý De Morgan và các định luật):

a) $F = (\overline{AB} + \overline{BD})(AC + BD)$; b) $F = AB + B\overline{D} + \overline{B}C + C\overline{D}$

c) $F = \overline{AB + \overline{C} + \overline{A}D}$; d) $F = \overline{\overline{A + B + CD} + \overline{C + D} + AB}$

Bài 2.2. Cho hàm F có ba biến A, B, C; ba biến này không bao giờ cùng ở mức cao hay cùng ở mức thấp. Hàm có mức logic cao khi có hai đầu vào có mức logic cao, trong trường hợp còn lại hàm có mức logic thấp. Hãy lập bảng trạng thái biểu diễn hàm.

Bài 2.3. Một bóng đèn đường cần đóng, ngắt độc lập ở 4 nơi khác nhau. Lập bảng trạng thái của hàm logic đó.

Bài 2.4. Cho hàm F có bảng trạng thái như trên hình 2.4BT.

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Hình 2.4BT

Biểu diễn hàm bằng phương trình logic dưới dạng tổng các tích và tích các tổng.

Bài 2.5. Thành lập bảng trạng thái cho hàm số sau: $F = AB\overline{D} + \overline{B}CD + \overline{AC}$

Bài 2.6. Cho các hàm logic có phương trình như sau:

a) $F_1(A, B, C) = \Sigma(0, 2, 4, 6)$ với N = 1, 3

b) $F_2(A, B, C, D) = \Pi(1, 2, 3, 6, 8, 9, 11, 12)$

Biểu diễn hàm bằng bảng Karnaugh.

Bài 2.7. Cho hàm logic có phương trình sau:

$F = AB\overline{C} + \overline{A}BC + \overline{B}C + ABCD + \overline{C}\overline{D}$

Biểu diễn hàm bằng bảng Karnaugh.

Bài 2.18. Cho hàm logic có phương trình sau: $F = (A+B)(\overline{B}+C+D)(\overline{A}+\overline{C})$

Biểu diễn hàm bằng bảng Karnaugh.

Bài 2.9. Cho các hàm logic 3 biến x_1, x_2, x_3 sau đây:

$F_1(x_1, x_2, x_3) = x_1x_2\overline{x_3} + x_1\overline{x_2}x_3 + x_1x_2x_3 + \overline{x_1}x_2x_3$

$F_2(x_1, x_2, x_3) = \overline{x_1}x_2\overline{x_3} + x_1\overline{x_2}\,\overline{x_3} + x_1x_2\overline{x_3} + x_1x_2x_3$

– Hãy biểu diễn bằng bảng Karnaugh?

– Hãy biểu diễn hàm logic này ở dạng khối?

– Hãy tìm các khối 0, khối 1, khối 2?

– Hãy tìm các biên giới đối nhau?

Bài 2.10. Cho hàm logic 3 biến x_1, x_2, x_3 sau đây:

$$F(x_1,x_2,x_3) = \overline{x_1}\overline{x_2}\overline{x_3} + x_1\overline{x_2}\overline{x_3} + x_1x_2\overline{x_3} + x_1x_2x_3 + \overline{x_1}x_2x_3$$

– Hãy biểu diễn bằng bảng Karnaugh?

– Hãy biểu diễn hàm logic này ở dạng khối?

– Hãy tìm K^0, K^1, K^2?

Bài 2.11. Cho hàm logic 4 biến như sau:

$$F(x_1,x_2,x_3,x_4) = \overline{x_1}x_2x_3\overline{x_4} + x_1\overline{x_2}\overline{x_3}x_4 + \overline{x_1}\overline{x_2}x_3x_4 + x_1\overline{x_2}x_3x_4 + x_1x_2\overline{x_3}\overline{x_4}$$
$$+ x_1\overline{x_2}x_3\overline{x_4} + x_1x_2\overline{x_3}x_4 + \overline{x_1}x_2\overline{x_3}\overline{x_4} + \overline{x_1}x_2x_3\overline{x_4} + x_1\overline{x_2}\overline{x_3}x_4 + \overline{x_1}x_2x_3\overline{x_4}$$

Hãy tìm K^0, K^1, K^2 và K(f)?

Bài 2.12. Cho hàm logic 3 biến x_1, x_2, x_3 sau đây:

$$F(x_1,x_2,x_3) = \overline{x_1}\overline{x_2}\overline{x_3} + \overline{x_1}x_2x_3 + x_1\overline{x_2}\overline{x_3} + x_1\overline{x_2}x_3 + x_1x_2x_3$$

– Hãy biểu diễn hàm logic này ở dạng khối?

– Hãy tìm K(f).

– Hãy tác động toán tử δ_i vào các khối của K(f) để tìm các khối có kích thước lớn hơn.

Bài 2.13. Cho hàm logic 4 biến như sau:

$$F(x_1,x_2,x_3,x_4) = \overline{x_1}x_2x_3\overline{x_4} + x_1\overline{x_2}\overline{x_3}x_4 + \overline{x_1}\,\overline{x_2}\,\overline{x_3}x_4 + x_1\overline{x_2}x_3x_4 + x_1x_2\overline{x_3}\,\overline{x_4} + \overline{x_1}\overline{x_2}\overline{x_3}\overline{x_4}$$
$$+ x_1\overline{x_2}x_3\overline{x_4} + x_1\overline{x_2}\overline{x_3}\overline{x_4} + \overline{x_1}x_2\overline{x_3}\overline{x_4} + \overline{x_1}x_2x_3\overline{x_4} + x_1x_2\overline{x_3}\overline{x_4} + \overline{x_1}x_2\overline{x_3}\overline{x_4}$$

– Hãy biểu diễn hàm logic này ở dạng khối?

– Hãy tìm K(f).

– Hãy tác động toán tử δ_i vào các khối của K(f) để tìm các khối có kích thước lớn hơn.

Bài 2.14. Cho 2 hàm logic 3 biến x_1, x_2, x_3 như sau:

$$f(x_1,x_2,x_3) = \overline{x_1x_2} + x_2x_3$$

$$g(x_1,x_2,x_3) = x_2\overline{x_3} + \overline{x_1}x_3$$

– Hãy tính: f.g

f + g

$\overline{f}$ và $\overline{g}$.

– Hãy tính: K(f) và K(g) ; $k(f) \cap k(g)$; $K(f) \cup K(g)$;

$K(\overline{f})$, $\overline{K(f)}$; $K(\overline{g})$, $\overline{K(g)}$

Bài 2.15. Tối thiểu hoá các hàm sau bằng phương pháp đại số:

a) $F(A,B,C) = \sum_m 0,2,3,4,6$

b) $F(A,B,C) = \Pi 1,2,3,6,7$

c) $F(A,B,C,D,E,F) = \Pi 5,7,21$

d) $F(A,B,C,D,E,F) = \sum_m 10,40,42$

Bài 2.16. Tối thiểu hoá các hàm sau bằng phương pháp đại số:

a) $A(\overline{A}C + BD) + B(C + DE) + B\overline{C}$

b) $\overline{A}\overline{B} + AC + \overline{C}D + \overline{B}\overline{C}\overline{D} + B\overline{C}E + \overline{B}\overline{C}G + \overline{B}CF$

c) $\overline{\overline{A}BC\overline{A}\overline{B} + \overline{B}\overline{C} + \overline{C}\overline{A}}$

Bài 2.17. Chứng minh các đẳng thức sau:

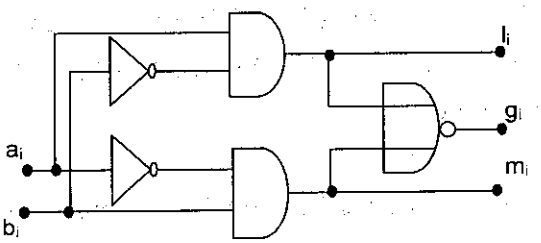
a) $A \oplus (B \sim C) = (A \oplus B) \sim C$; b) $A \oplus B \oplus C = A \sim B \sim C$;

c) $A \oplus B \oplus C \oplus D = \overline{A \sim B \sim C \sim D}$

Bài 2.18. Xây dựng sơ đồ mạch tổ hợp dùng các cổng XOR 2 đầu vào cho bởi hàm sau:

$F = \overline{A}\overline{B}\overline{C}D + A\overline{B}\overline{C}D + A\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}\overline{C}D + \overline{A}BCD + ABC\overline{D} + \overline{A}\overline{B}\overline{C}\overline{D} + A\overline{B}CD$

Bài 2.19. Cho sơ đồ logic như hình 2.19BT, viết phương trình hàm logic F.



Hình 2.19BT

Bài 2.20. Thực hiện các hàm logic sau:

$F_1 = AB + \overline{B}\overline{C} + BC$; $F_2 = A(\overline{B} + CD)$

a) Bằng cổng NOR hai đầu vào

b) Bằng cổng NOR có số đầu vào tuỳ ý.

Bài 2.21. Thực hiện các hàm logic sau:

$F_1 = AB + \overline{B}\overline{C} + BC$; $F_2 = A(\overline{B} + CD)$

a) Bằng cổng NAND hai đầu vào

b) Bằng cổng NAND có số đầu vào tuỳ ý.

Bài 2.22. Hãy dùng các phép tính NOT, AND, OR để viết các hàm XOR và hàm XNOR sau:

$F_3 = A \oplus B \oplus C$; $T_3 = A \sim B \sim C$; $F_4 = A \oplus B \oplus C \oplus D$; $T_4 = A \sim B \sim C \sim D$

a) Hãy khái quát sự phụ thuộc của kết quả vào số lượng các giá trị 0 và 1 của các biến số trong trường hợp hàm n biến.

b) Tìm mối quan hệ giữa hàm XOR và hàm XNOR trong trường hợp hàm n biến.

Bài 2.23. Hãy áp dụng tính chất của hàm XOR đơn giản các hàm sau:

a) $\overline{A}B \oplus BC \oplus AB \oplus \overline{BC}$; b) $\overline{A}B \oplus BC \oplus AB \oplus B\overline{C}$; c) $\overline{AB} \oplus BC \oplus AB \oplus \overline{BC}$

Bài 2.24. Hãy dùng cổng AND và OR hai đầu vào để thực hiện hàm:

$$F(A,B,C) = \Sigma\,(3, 5, 6)$$

Bài 2.25. Với các giá trị như thế nào của các biến số thì cả ba đẳng thức sau cùng đúng:

$$\overline{A} + AB = 0$$

$$AB = AC$$

$$AB + A\overline{C} + CD = \overline{C}D$$

Bài 2.26. Chứng minh rằng:

Nếu $\overline{A}B + C\overline{D} = 0$ thì $AB + \overline{C}(\overline{A} + \overline{D}) = AB + BD + \overline{BD} + \overline{AC}D$

Bài 2.27. Tối thiểu hoá các hàm sau bằng bảng Karnaugh:

$F(A,B,C) = \sum_m 0,1,2,3,4,5,6$; $F(A,B,C,D) = \sum_m 0,1,2,3,4,9,10,12,13,14,15$

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

Bài 2.28. Hãy vẽ sơ đồ của hàm xác định không hoàn toàn:

$F(A, B, C, D) = \Pi\,(5, 9, 10, 11, 13, 15)$ với $N = 0, 7, 8, 12$

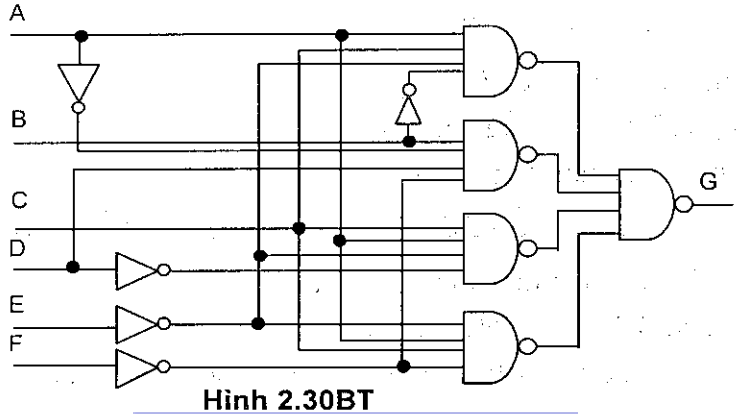
a) Chỉ dùng các mạch NOR.

b) Chỉ dùng các mạch NAND.

Bài 2.29. Vẽ sơ đồ mạch cho biểu thức sau với số cổng sử dụng là ít nhất:

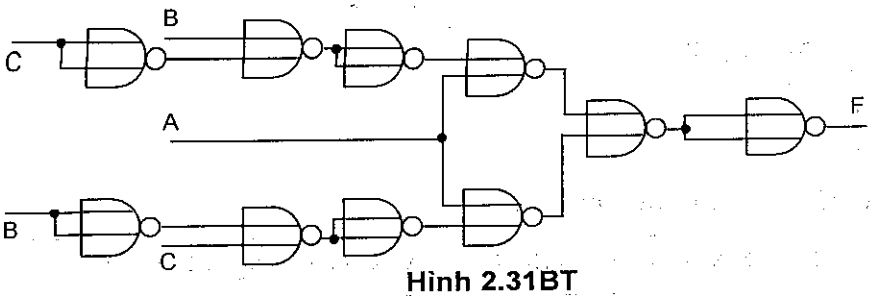
$$F = ABCD + A\overline{BC}D + \overline{AD}$$

Bài 2.30. Hãy đơn giản hoá mạch logic cho ở hình 2.30BT.



Hình 2.30BT

Bài 2.31. Cho sơ đồ như ở hình 2.31BT, xác định khả năng có thể đơn giản hoá mạch mà vẫn dùng loại cổng đó.



Hình 2.31BT

Bài 2.32. Vẽ sơ đồ mạch thực hiện hàm logic sau: $F = A \oplus B \oplus C$

a) Chỉ dùng mạch NAND 7400 *(Gợi ý: Sơ đồ tối ưu dùng 8 phần tử).*

b) Chỉ dùng mạch NOR 7402 *(Gợi ý: Sơ đồ tối ưu dùng 8 phần tử).*

Bài 2.33. Thiết kế mạch logic tổ hợp có 4 đầu vào A, B, C, D và 2 đầu ra F_1, F_2 như sau:

$$F_1(A,B,C,D) = \sum_m (3,4,5,7,13,15)$$

$$F_2(A,B,C,D) = \sum_m (3,8,9,11,13,15)$$

a) Chỉ dùng các cổng logic cơ bản.

b) Chỉ dùng các cổng NAND.

c) Chỉ dùng các cổng NOR.

Bài 2.34. Thiết kế mạch logic tổ hợp có 4 đầu vào A, B, C, D và 3 đầu ra F_1, F_2, F_3 như sau:

$$F_1(A,B,C,D) = \sum_m (10,12,14,15) \quad \text{với N = 7,8.}$$

$$F_2(A,B,C,D) = \sum_m (0,7,12) \qquad \text{với N = 2,14,15}$$

$$F_3(A,B,C,D) = \sum_m (2,8,14) \qquad \text{với N = 0,10,12}$$

a) Chỉ dùng các cổng NAND.

b) Chỉ dùng các cổng NOR.

Bài 2.35. Cho các khối sau đây của hàm 6 biến:

$C^1(0x1100)$, $C^3(1x01xx)$, $C^5(xx1xxx)$

Hãy tìm giá G^a của các khối này?

Bài 2.36. Cho hàm ba biến có phủ như sau:

$$C = \left\{ \begin{matrix} 000 \\ 0x0 \\ x11 \\ xx0 \end{matrix} \right\}$$ Hãy tìm $G^a[C]$?

Bài 2.37. Cho hàm ba biến có phủ như sau:

$$C = \begin{Bmatrix} 000 \\ 010 \\ x00 \\ 0x1 \\ xx0 \end{Bmatrix}$$ Hãy tìm $G^b[C]$?

Bài 2.38. Cho hàm logic 4 biến như sau:

$$F(x_1,x_2,x_3,x_4) = \bar{x}_1x_2x_3\bar{x}_4 + x_1\bar{x}_2\bar{x}_3\bar{x}_4 + x_1\bar{x}_2x_3x_4 + x_1x_2x_3x_4 + \bar{x}_1x_2\bar{x}_3x_4 + x_1x_2\bar{x}_3\bar{x}_4$$
$$+ x_1\bar{x}_2x_3\bar{x}_4 + x_1x_2\bar{x}_3\bar{x}_4 + x_1x_2x_3\bar{x}_4 + \bar{x}_1\bar{x}_2x_3x_4 + \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4$$

– Hãy biểu diễn hàm ở dạng khối và tìm K(f)?

– Hãy tìm phủ tối thiểu của K(f), $G^b[C_{min}]$?

Bài 2.39. Cho hàm logic 3 biến sau: $F(x_1,x_2,x_3) = \sum(0,6,7)$ với N = 2,3,5.

– Hãy biểu diễn hàm ở dạng khối?

– Hãy tìm L, N và K?

* Phủ của phức hợp khối L:

Định nghĩa: Phủ C^L của phức hợp khối L là tập hợp C^L các khối sao cho các khối của C^L phải nằm trong K(f) và các khối của C^L chỉ chứa các đỉnh của L (không chứa các đỉnh của N).

Bài 2.40. Cho hàm logic 3 biến sau: $F(x_1,x_2,x_3) = \sum(0,2,4,6,7)$ với N = 1,3.

– Hãy biểu diễn hàm ở dạng khối?

– Hãy tìm L, N và K?

– Hãy tìm các C^L_{min}, $G^b\left[C^L_{min}\right]$?

*Phủ của phức hợp khối K:

Định nghĩa: Phủ C^K của phức hợp khối K là tập hợp C^K các khối sao cho các khối của C^K phải nằm trong K(f) và các khối của C^K phải chứa tất cả các đỉnh của L và tất cả các đỉnh của N (tức là tất cả các đỉnh của K).

Bài 2.41. Cho hàm logic 3 biến sau: $F(x_1,x_2,x_3) = \sum(0,2,4,7)$ với N = 5,6.

– Hãy biểu diễn hàm ở dạng khối?

– Hãy tìm L, N và K?

– Hãy tìm các C^L_{min}, $G^b\left[C^L_{min}\right]$, C^K_{min} và $G^b[C^K_{min}]$?

Bài 2.42. Cho hàm logic 3 biến sau: $F(x_1,x_2,x_3)=\sum(0,4,5)$ với N = 3.

– Hãy biểu diễn hàm ở dạng khối?

– Hãy tìm L, N và K?

– Hãy tìm các C^L_{min}, $G^b\left[C^L_{min}\right]$, C^K_{min}, $G^b[C^K_{min}]$, C^{KL}_{min}, $G^b[C^{KL}_{min}]$?

Bài 2.43. Cho hàm logic 3 biến sau: $F(x_1,x_2,x_3)=\sum(3,5,6,7)$ với N = 2,4.

– Hãy biểu diễn hàm ở dạng khối?

– Hãy tìm L, N và K?

– Hãy tìm các C^L_{min}, $G^b\left[C^L_{min}\right]$, C^K_{min}, $G^b[C^K_{min}]$, C^{KL}_{min}, $G^b[C^{KL}_{min}]$?

Bài 2.44. Cho hàm logic 4 biến sau: $F(x_1,x_2,x_3,x_4)=\sum(1,7,12,13,14)$ với N = 5,9,15.

– Hãy biểu diễn hàm ở dạng khối?

– Hãy tìm L, N và K?

– Hãy tìm các C^L_{min}, $G^b\left[C^L_{min}\right]$, C^K_{min}, $G^b[C^K_{min}]$, C^{KL}_{min}, $G^b[C^{KL}_{min}]$?

Bài 2.45. Cho hàm logic 4 biến x_1, x_2, x_3, x_4 sau đây:

$$F(x_1,x_2,x_3,x_4)=\sum(0,1,4,5,6,8,10,14)$$

Hãy tìm Z^0, Z^1, Z^2, Z^3 và Z?

Bài 2.46. Cho hàm logic 4 biến x_1, x_2, x_3, x_4 như sau:

$$f(x_1,x_2,x_3,x_4)=\overline{x_1}\,\overline{x_2}\,\overline{x_3}\,\overline{x_4}+\overline{x_1}\,\overline{x_2}x_3\overline{x_4}+\overline{x_1}\,\overline{x_3}x_4+x_2x_3$$

– Hãy biểu diễn hàm ở dạng hình học?

– Hãy tìm phủ C (theo đầu bài đã cho)?

– Hãy tìm tích tọa độ * giữa các khối trong phủ C?

Bài 2.47. Cho hàm logic 6 biến x_1, x_2, x_3,x_4, x_5,x_6 như sau:

$$f(x_1,x_2,x_3,x_4,x_5,x_5)=\overline{x_1}\,\overline{x_2x_3x_4x_5x_6}+x_1\overline{x_2x_3x_4x_5x_6}+x_2x_4x_6+x_1\overline{x_2x_3x_4}$$

– Hãy tìm phủ C (theo đầu bài đã cho)?

– Hãy tìm tích tọa độ * giữa các khối trong phủ C?

Bài 2.48. Tối thiểu hóa hàm sau bằng phương pháp Quine – Cluskey

$$F(A,B,C,D,E)=\sum(0,1,2,8,9,10,30,31)$$

Chương 3

CÁC HỌ VI MẠCH LOGIC CƠ BẢN

3.1. MỞ ĐẦU

Trong chương 2 chúng ta đã biết về các cổng logic, chương này giới thiệu về công nghệ vi mạch được sử dụng để thực hiện các cổng logic này cũng như các loại thiết bị vi mạch.

Trong chương này trình bày hai công nghệ vi mạch chính là TTL và CMOS và định nghĩa các thông số hoạt động của chúng. Ngoài ra, các đặc điểm hoạt động của các xê – ri khác nhau trong các công nghệ vi mạch này được đưa ra so sánh. Về bảng chân lý của một cổng logic nhất định thì giống nhau đối với các công nghệ vi mạch khác nhau, nó chỉ khác nhau về các đặc điểm điện như là công suất, tần số, khả năng kháng nhiễu,...

3.2. ĐẶC ĐIỂM HOẠT ĐỘNG VÀ CÁC THÔNG SỐ

3.2.1. Đặc điểm của các vi mạch logic

– Lối vào và lối ra của các vi mạch logic chỉ có hai mức điện áp V_L và V_H tương ứng với mức logic 0 và 1 (có thể viết tắt là L và H).

– Các mạch logic phải được nuôi bằng nguồn nuôi có một điện áp chuẩn đã được quy định.

– Cùng một chức năng logic nhưng kỹ thuật điện tử có thể thực hiện theo những sơ đồ nguyên lý khác nhau.

– Những vi mạch được xây dựng trên cùng một kiểu sơ đồ nguyên lý được xếp vào một họ logic. Các vi mạch logic trong cùng một họ logic phải được nuôi bằng nguồn điện có điện áp bằng điện áp nuôi chuẩn cho họ logic đó. Các mức logic của các vi mạch này phải như nhau. Các vi mạch logic có mức logic phù hợp chúng có thể ghép nối trực tiếp với nhau.

3.2.2. Các thông số cơ bản của vi mạch logic

Các IC số được phân loại theo độ phức tạp của mạch như số lượng cổng riêng biệt cần thiết để xây dựng mạch thực hiện cùng chức năng logic.

Sự phân lớp mạch	Các cổng logic riêng biệt cần thiết
Mạch tích hợp cỡ nhỏ (SSI)	< 12
IC cỡ trung bình (MSI)	≥ 12 và < 100
IC cỡ lớn (LSI)	≥ 100 và < 1000
IC cực lớn (VLSI)	≥ 1000

Những đặc tính của IC số được dùng để so sánh chúng là:

1. Trở kháng ra (output impedance)

Thay đổi theo trạng thái đầu ra cao hay thấp. Nếu mạch ra dùng một tranzito với trở tải ở colectơ thường có trở kháng ra $Z_{out} \approx 2000\Omega$ cao hơn mạch dùng hai tranzito ($Z_{out} \approx 70\Omega$).

2. Hệ số mắc tải (Fan out)

Cho biết lối ra có thể điều khiển đồng thời được bao nhiêu lối vào song song của các mạch khác.

3. Hệ số hợp lối vào (Fan in)

Cho biết có thể mắc song song bao nhiêu lối vào vẫn đảm bảo hợp thông số.

4. Thời gian trễ (Propagation delay per gate)

Thời gian trễ trên một cửa T_p là thời gian từ lúc lối vào nhận được tín hiệu đến lúc lối ra bắt đầu thay đổi trạng thái. Thời gian trễ T_p được xác định theo công thức:

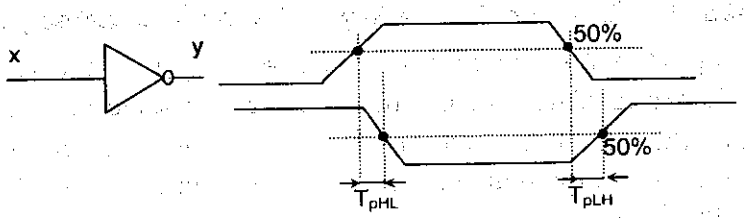
Trong đó: $T_p = \dfrac{T_{pHL} + T_{pLH}}{2}$

T_{pHL} là thời gian trễ khi tín hiệu chuyển từ mức cao sang mức thấp.

T_{pLH} là thời gian trễ khi tín hiệu chuyển từ mức thấp sang mức cao.

Thời gian trễ tại các mức được xác định giữa hai thời điểm tín hiệu vào và tín hiệu ra chuyển đổi mức một nửa thời gian như trên hình 3.2.2.1, T_{pHL} và T_{pLH} thường không bằng nhau trong các mạch.

T_p càng nhỏ tốc độ làm việc càng cao.



Hình 3.2.2.1. Cửa đảo và thời gian trễ qua cửa đảo

5. Các tham số dòng điện và điện áp

– V_{IH}: điện áp vào mức cao, là điện áp vào được cổng nhận là mức logic 1.

– V_{IL}: điện áp vào mức thấp, là điện áp vào được cổng nhân ra là logic 0.

– V_{OH}: điện áp ra mức cao, là điện áp tại đầu ra tương ứng với mức logic 1.

– V_{OL}: điện áp ra ở mức thấp, là điện áp tại đầu ra tương ứng với mức logic 0.

– I_{IH}: dòng điện vào mức cao, là dòng điện tối thiểu được cung cấp bởi một nguồn tương ứng với mức logic 1.

– I_{IL}: dòng điện vào mức thấp, là dòng điện tối đa được cung cấp bởi 1 nguồn tương ứng với mức logic 0.

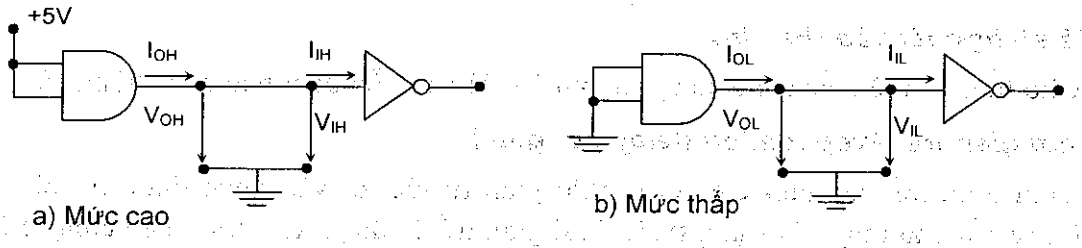
+ I_{OH}: dòng điện ra mức cao, là dòng điện cổng có thể đưa ra tại mức cao.

+ I_{OL}: dòng điện ra mức thấp, là dòng điện cổng có thể đưa ra tại mức thấp.

+ I_{CCH}: dòng điện cung cấp mức cao, là dòng điện cung cấp khi đầu ra của cổng ở logic 1.

+ I_{CCL}: dòng điện cung cấp mức thấp, là dòng điện cung cấp khi đầu ra của cổng ở logic 0.

Các tham số dòng điện và điện áp được minh họa trên hình 3.2.2.2.



Hình 3.2.2.2. Các tham số dòng điện và điện áp

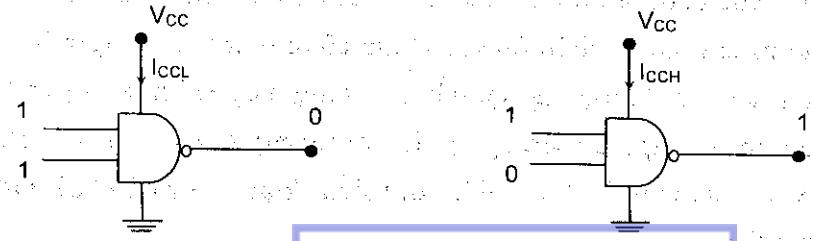
6. Nguồn nuôi (Power Supply)

Nguồn nuôi cho các mạch logic phải là nguồn ổn áp có điện áp ra đúng với điện áp nuôi quy định riêng cho từng họ mạch logic. Khi lối vào, lối ra thay đổi trạng thái làm cho cường độ dòng điện trong toàn bộ mạch thay đổi đột ngột, sự thay đổi này có thể làm rối loạn hoạt động của các mạch khác. Để khắc phục hiện tượng này giữa chân nguồn và đất của các vi mạch người ta thường mắc thêm tụ lọc để loại bỏ nhiễu, các tụ này phải dùng tụ gốm có điện dung cỡ chừng 0,1μF đến 1μF.

7. Công suất tiêu thụ đối với một cửa logic (Power dissipated per gate)

Công suất càng lớn khi mạch có nhiều điện trở có giá trị nhỏ và tranzito làm việc ở chế độ bão hoà. Trong cùng một họ logic các sê – ri khác nhau công suất tiêu thụ trên một cửa cũng khác nhau. Ví dụ họ logic TTL sê – ri 74 có công suất tiêu thụ trên một cửa là 10mW; 74L công suất tiêu thụ trên một cửa là 1mW; 74H công suất tiêu thụ trên một cửa là 22mW.

Cùng một loại vi mạch, cùng một nguồn cung cấp nhưng khi tín hiệu ở đầu ra khác nhau thì dòng điện tiêu hao do nguồn cung cấp khác nhau (tương ứng là I_{CCH} và I_{CCL} như hình 3.2.2.3) .



Hình 3.2.2.3. Dòng điện tiêu hao tương ứng với mức logic ở đầu ra

Để ước tính dòng điện cần cho một nhóm cổng, phải đặt giả thiết một nửa đầu ra sinh ra 1 và một nửa đầu ra sinh ra 0, dòng trung bình: $I_{CC} = \frac{I_{CCL} + I_{CCH}}{2}$

Công suất tiêu hao được tính bằng tích của điện áp nguồn cung cấp và dòng điện tiêu hao.

Vậy, công suất tiêu hao trung bình: $P_D = V_{CC}.I_{CC}$.

Họ PMOS, NMOS và CMOS tiêu thụ công suất rất nhỏ so với các họ logic khác.

8. Mức độ chống tạp âm (Noise Immunity Level)

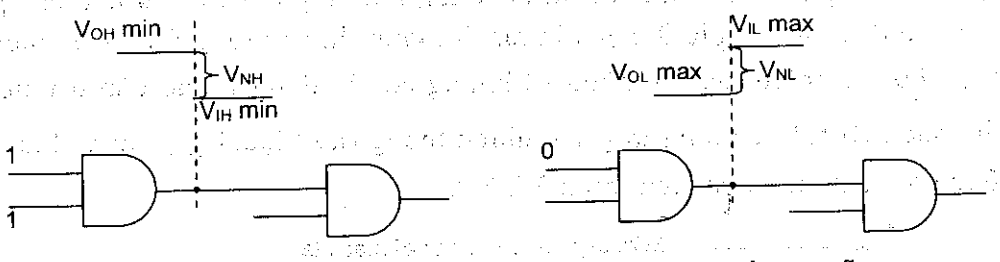
Là biên độ tạp âm lớn nhất có thể vào mạch mà không làm thay đổi trạng thái lối ra.

Khoảng điện thế đầu vào và khoảng điện thế đầu ra quy định mức logic trong một họ vi mạch nào đó thường không bằng nhau, sự khác nhau đó đặc trưng cho khả năng chống nhiễu của mạch.

Khoảng điện thế chênh lệch đó được gọi là khoảng dự trữ chống nhiễu và được tính như sau:

$$V_{NH} = V_{OH}(min) - V_{IH}(min); \qquad V_{NL} = V_{IL}(max) - V_{OL}(max)$$

Khoảng dự trữ chống nhiễu được minh họa trên hình 3.2.2.4.



Hình 3.2.2.4. Minh họa khoảng dự trữ chống nhiễu

Ta thấy rằng nếu có nhiễu làm cho điện áp đầu ra tại mức cao giảm xuống dưới giá trị V_{OH} min nhưng vẫn lớn hơn V_{IH} min thì đầu vào vẫn nhận được là mức logic 1. Tương tự như thế, nếu có nhiễu làm cho điện áp đầu ra tại mức thấp tăng lên trên giá trị V_{OL} max nhưng vẫn nhỏ hơn V_{IL} max thì đầu vào vẫn nhận được là mức logic 0. Như vậy, họ IC nào có khoảng dự trữ chống nhiễu càng lớn thì khả năng chống nhiễu càng cao.

Ví dụ 3.2.2.1. Cho một họ logic có $V_{OH(min)} = 2,4V$, $V_{IH(min)} = 2V$, $V_{OL(max)} = 0,4V$ và $V_{IL(max)} = 0,8V$. Hãy xác định:

a) Đỉnh nhiễu có biên độ lớn nhất chấp nhận được khi đầu ra ở mức cao đang kích thích một đầu vào.

b) Đỉnh nhiễu có biên độ lớn nhất có thể chấp nhận được khi đầu ra ở mức thấp đang kích thích một đầu vào.

Giải:

a) Đỉnh nhiễu có biên độ lớn nhất bằng khoảng dự trữ chống nhiễu ở mức cao:

$$V_{NH} = V_{OH(min)} - V_{IH(min)} = 2,4V - 2V = 0,4V.$$

b) Đỉnh nhiễu có biên độ lớn nhất bằng khoảng dự trữ chống nhiễu ở mức thấp:

$$V_{NL} = V_{IL(max)} - V_{OL(max)} = 0,8V - 0,4V = 0,4V.$$

9. Tần số xung nhịp cực đại (Maximum Clock Rate)

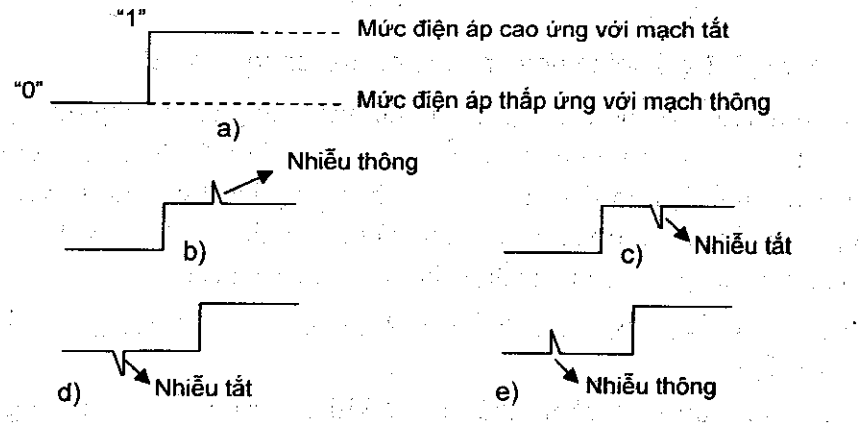
Khi các cửa logic dùng làm trigơ thì loại cửa nào có T_p nhỏ sẽ làm việc được với những xung nhịp có tần số cao, tốc độ chuyển mạch nhanh.

10. Độ ổn định nhiễu tĩnh

Độ ổn định nhiễu tĩnh là tham số đặc trưng cho sự làm việc tin cậy của mạch khi có tác động nhiễu.

* *Định nghĩa nhiễu tĩnh:* Nhiễu tĩnh là nhiễu mà các giá trị giữ nguyên không đổi trong khoảng thời gian lớn hơn đáng kể thời gian của giai đoạn quá độ của mạch.

Mạch logic ở chế độ tĩnh sẽ nằm ở một trong hai trạng thái thông hoặc tắt, tương ứng với hai mức logic (tùy ta quy ước cực tính logic). Trạng thái thông là ứng với điện áp ra thấp, khi đó dòng colectơ của tranzito đảo lớn, tức là dòng điện ra cung cấp cho các tầng tiếp theo sẽ rất nhỏ. Trạng thái tắt tương ứng với mức điện áp ra cao, khi đó dòng colectơ của tranzito đảo nhỏ, tức dòng điện ra cung cấp cho các tầng tiếp theo sẽ rất lớn. Vì vậy, ta phải phân biệt hai loại nhiễu (thường biểu diễn ở dạng điện áp) là nhiễu thông và nhiễu tắt. Giả sử ta có hai mức logic 0 và 1 (0 là điện áp thấp, 1 là điện áp cao), ở mức logic 0 mạch là mạch thông thì sẽ tương ứng với nhiễu tắt (ký hiệu: U_{nh}^{-}) bởi vì nhiễu lúc đó thông sẽ không có tác dụng gì cả, còn ở mức logic 1 mạch là mạch tắt thì sẽ tương ứng với nhiễu thông (ký hiệu: U_{nh}^{+}) bởi vì lúc đó nhiễu tắt không có tác dụng gì cả, xem hình 3.2.2.4.



Hình 3.2.2.4 . Nhiễu thông và nhiễu tắt tác động vào mạch thông và tắt

Nhiễu có thể làm cho mạch chuyển biến nhầm. Trên hình 3.2.2.4 ta thấy ở hình b nhiễu thông tác dụng vào mạch thông nên không có tác dụng, ở hình c nhiễu tắt tác dụng vào mạch thông nên có tác dụng, ở hình d nhiễu tắt tác dụng vào mạch tắt nên không có tác dụng và ở hình e nhiễu thông tác dụng vào mạch tắt nên có tác dụng.

Vì vậy, để tăng độ ổn nhiễu ta dùng các đại lượng nhiễu này.

** Định nghĩa độ ổn định nhiễu:* Độ ổn định nhiễu tĩnh của mạch là biên độ cực đại cho phép của nhiễu thông và tác động lên đầu vào của mạch mà vẫn đảm bảo mạch không làm việc nhầm.

Có khi người ta không sử dụng giá trị tuyệt đối của điện áp nhiễu tĩnh cho phép cực đại làm độ ổn định nhiễu mà bằng hệ số ổn định nhiễu tĩnh: là tỷ số của điện áp nhiễu tĩnh cho phép cực đại ở đầu trên chênh lệch mức logic cho phép cực tiểu ($\Delta U_{min} = U^1_{min} - U^0_{max}$)

$$K^{\pm}_{nh} = \frac{U^{\pm}_{nh}}{\Delta U_{min}}$$

Ở đây: U^1_{min} : điện áp cho phép cực tiểu của mức logic 1;

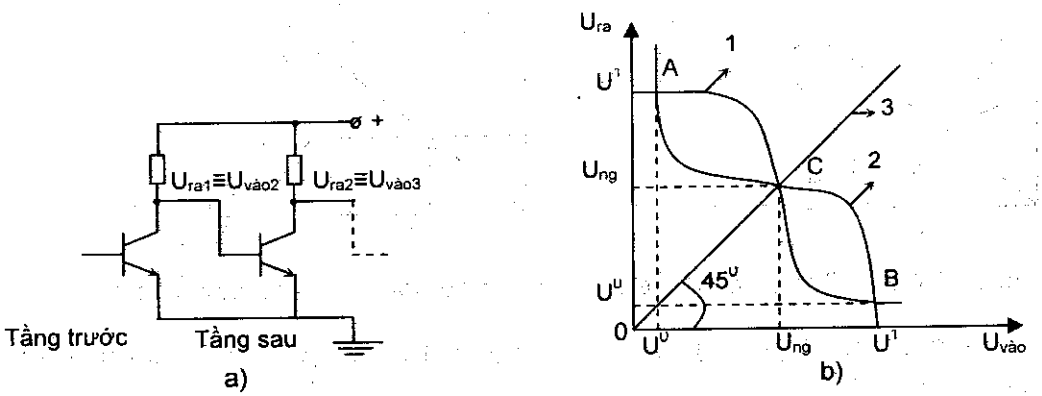
U^0_{max} : điện áp cho phép cực đại của mức logic 0;

$U^+_{nh} = (U^0_{ng\,min} - U^0_{max})$: độ ổn định nhiễu tuyệt đối;

$U^-_{nh} = (U^1_{ng\,max} - U^1_{max})$: độ ổn định nhiễu tuyệt đối;

$U^0_{ng\,min}$: điện áp ngưỡng cực tiểu của mức logic 0;

$U^1_{ng\,max}$: điện áp ngưỡng cực đại của mức logic 1;



Hình 3.2.2.5 . Đặc tuyến truyền đạt của mạch logic loại đảo

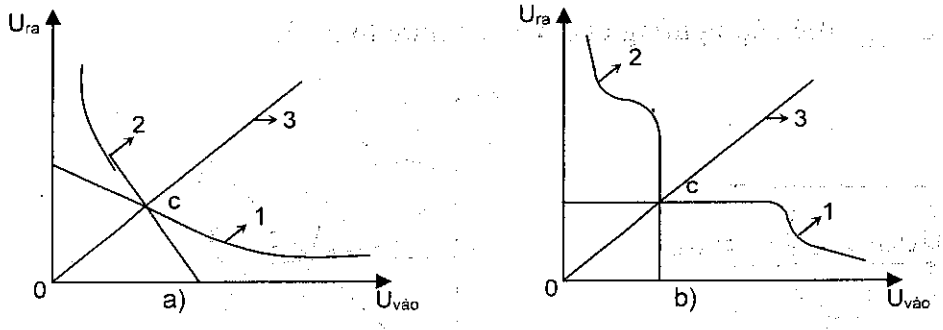
a) Sự ghép giữa các tầng đảo; b) Đặc tuyến truyền đạt K: đường 1 – đặc tuyến truyền đạt xuôi dương; đường 2 – đặc tuyến truyền đạt ngược dương; đường 3 – đường phân giác chia đôi góc vuông thứ nhất (trên đường này $U_{ra} = U_{vào}$).

Để xác định các đại lượng này ta thường dùng một đặc tuyến rất quan trọng của mạch logic, đó là đặc tuyến truyền đạt $U_{ra} = f(U_{vào})$. Thường các mạch logic được thực hiện trên các mạch thuộc loại phủ định như NAND, NOR tín hiệu vào ở mức logic 1 (điện áp cao) thì tín hiệu ra ở mức logic 0 (điện áp thấp) và ngược lại. Từ đó ta có đặc tuyến truyền đạt ở hình 3.2.2.5b là đường cong 1. Trong các thiết bị logic đầu ra của tầng trước sẽ nối vào đầu ra của tầng sau, tức là U_{ra} tầng trước sẽ chính là $U_{vào}$ tầng sau. Vì vậy, ta sẽ dùng đặc tuyến truyền đạt ngược để xét khi ta coi tất cả các tầng là hoàn toàn giống nhau (hình 3.2.2.5a).

Đặc tuyến truyền đạt ngược là sự phụ thuộc nghịch của $U_{ra} = f(U_{vào})$, đặc tuyến truyền đạt ngược rút ra trực tiếp từ đặc tuyến truyền đạt bằng cách thay U_{ra} bằng $U_{vào}$ và $U_{vào}$ bằng U_{ra}, nghĩa là đổi trục tọa độ trên đồ thị, ta sẽ có đường cong 2 trên hình 3.2.2.5b.

Trên hình 3.2.2.5b ta thấy đặc tuyến truyền đạt xuôi và ngược cắt nhau tại 3 điểm: A, B, C. Điểm C nằm trên đường phân giác (đường 3) là điểm mà tại đó $U_{ra} = U_{vào}$, điện áp vào ở điểm C ta gọi là điện áp ngưỡng (U_{ng}). Ta thấy rằng điểm A và điểm B là hai điểm làm việc ổn định của mạch logic, bởi vì nếu điện áp vào nằm trong đoạn mạch từ C đến B (trừ hai đầu mút) thì sau một vài lần đảo (qua các bộ đảo) điểm làm việc sẽ trở về điểm B, nếu điện áp vào đúng bằng điện áp ở điểm B thì nó sẽ ổn định ở đó. Còn điện áp vào nằm đoạn từ C đến A, cũng như trên, sau một vài lần đảo điểm làm việc sẽ trở về điểm A và ổn định ở đó.

Như vậy, muốn mạch logic của chúng ta làm việc được thì đặc tuyến truyền đạt xuôi và ngược phải cắt nhau ở ba điểm. Nếu số điểm cắt nhau ít hơn ba thì mạch sẽ không làm việc được (hình 3.2.2.6). Nói cách khác, nếu không có hai điểm làm việc ổn định A và B thì mạch sẽ không làm việc được, lúc đó nếu tín hiệu vào thay đổi mức logic thì đầu ra vẫn không thay đổi mức logic.



Hình 3.2.2.6. Đặc tuyến truyền đạt không có hai điểm làm việc ổn định

Muốn cho đặc tuyến truyền đạt xuôi và ngược cắt nhau tại ba điểm thì phải thực hiện được điều kiện sau:

$$\left|\frac{dU_{ra}}{dU_{vào}}\right|_{U_{ra}=U_{vào}} > 1$$

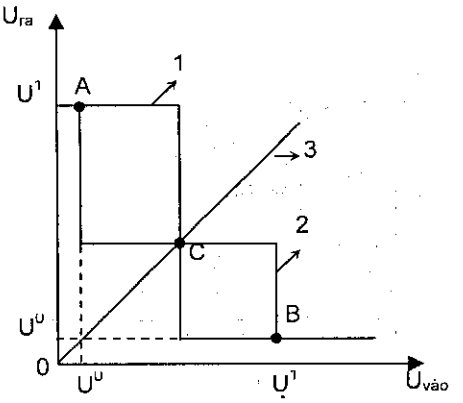
Dựa vào mối quan hệ toán học giữa điện áp ra và vào ta có thể xác định được điều kiện làm việc ổn định của một mạch logic theo các tham số của nó.

Một mạch logic làm việc lý tưởng là phải có các điều kiện sau:

$$\left|\frac{dU_{ra}}{dU_{vào}}\right|_{U_{ra}=U_{vào}} > \infty \qquad (3.2.2.1)$$

Điều kiện (3.2.2.1) là điều kiện để mạch làm việc tốt nhất, nhưng thực tế không bao giờ đạt được điều kiện lý tưởng (3.2.2.1). Điều kiện lý tưởng này được vẽ trên hình 3.2.2.7.

Tọa độ các điểm cắt của đặc tuyến xuôi và ngược (A, B, C) cho ta xác định được các tham số quan trọng nhất của mạch logic là U^0 (điện áp của mức logic 0), U^1 (điện áp của mức logic 1), U_{ng} (điện áp của mức ngưỡng). Dựa vào đó ta xác định được chênh lệch mức logic $U_L = U^1 - U^0 \equiv \Delta U$.

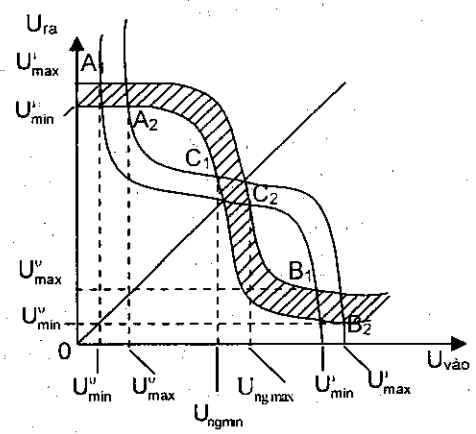


Hình 3.2.2.7. Điều kiện lý tưởng để mạch làm việc tốt nhất

Trong các mạch điện thực tế ta thấy rằng do các tham số của các linh kiện khác với giá trị danh định, số tải mà ở đầu ra của mạch khác nhau và các yếu tố khác, nên đặc tuyến truyền đạt của chúng không phải là đường mảnh (không có bề rộng), mà là một dải với bề rộng xác định nào đó (hình 3.2.2.8). Do đó, các tham số của mạch chúng ta vừa nói ở trên sẽ cùng nằm trong một dải:

$$U^0_{min} \le U^0 \le U^0_{max}; \quad U^1_{min} \le U^1 \le U^1_{max}; \quad U_{ng\,min} \le U_{ng} \le U_{ng\,max}$$

Chúng ta có thể dùng đặc tuyến truyền đạt này để xác định độ ổn định nhiễu tuyệt đối và tương đối (hệ số ổn định nhiễu K^1_{nh}). Quá trình xác định như sau: trên hình 3.2.2.9a là đặc tuyến trường hợp mạch điện đồng nhất (đặc tuyến là đường chứ không phải là dải), chúng ta sẽ dịch đặc tuyến truyền đạt ngược lên phía trên cho đến khi nó tiếp xúc với đặc tuyến truyền đạt xuôi ở điểm C_0, sau đó ta dịch đặc tuyến truyền đạt ngược xuống phía dưới cho đến khi nó tiếp xúc với đặc tuyến truyền đạt xuôi ở điểm C_1. Hoành độ của hai điểm C_0 và C_1 sẽ là U^0_{ng} và U^1_{ng}, từ đó ta xác định được độ ổn định nhiễu tuyệt đối:



Hình 3.2.2.8. Đặc tuyến truyền đạt của mạch điện thực tế khi có sai lệch các tham số của mạch.

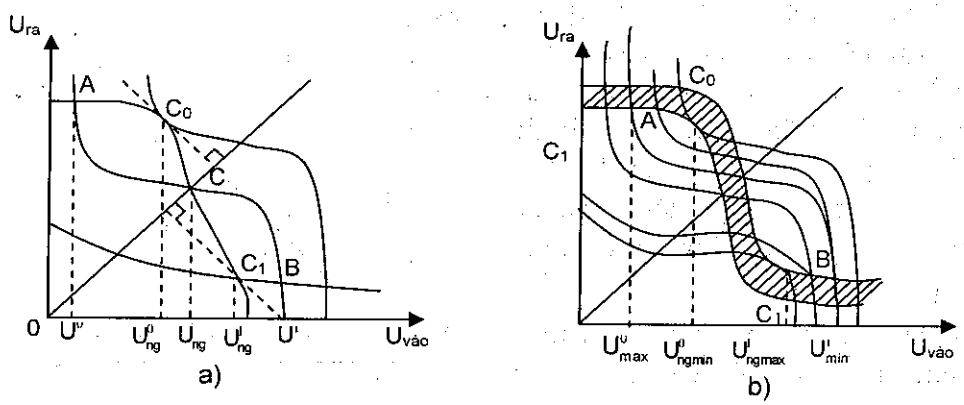
$$U_{nh} = U^0_{ng} - U^0 = U^1 - U^1_{ng}$$

Như thế tức là ta có: $U_{nh} = U^+_{nh} = U^-_{nh}$

Nếu mạch là không đồng nhất ta cũng xác định tương tự theo hình 3.2.2.9b, như vậy ta có: $U_{nh} = U^0_{ng\,min} - U^0_{max} = U^1_{min} - U^1_{ng\,max}$

Trong trường hợp nhiễu xuất hiện ngay ở đầu vào có thể làm đặc tuyến truyền đạt xuôi dịch chuyển, trong trường hợp đó ta cũng có thể làm tương tự.

Ví dụ về sự xuất hiện nhiễu trong mạch logic: Giả sử ta có mạch điện như hình 3.2.2.10, mạch gồm nhiều khâu logic mắc với nhau.

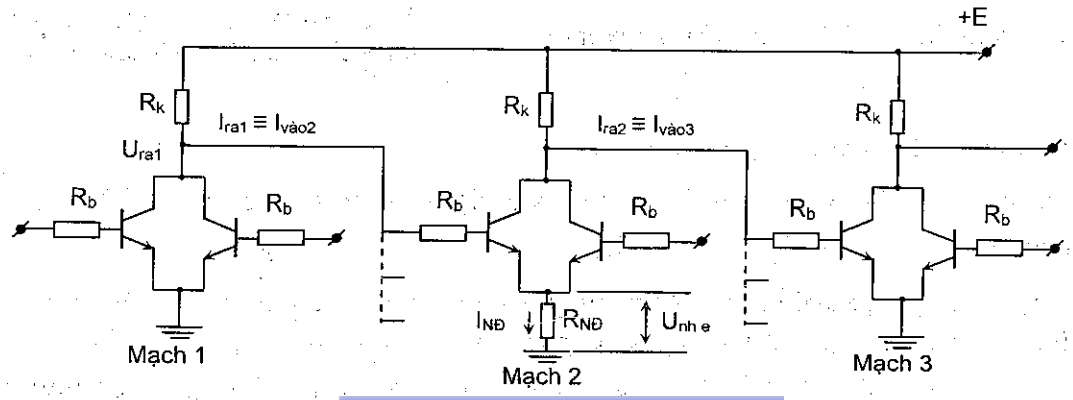


Hình 3.2.2.9. Đặc tuyến để xác định độ ổn định

Trong trường hợp này nhiễu tĩnh sẽ xuất hiện trên đường dây nối đất, vì trên đường dây đó sẽ có dòng lớn chảy qua nên sẽ gây sụt áp lớn trên điện trở của đường đất. Trên hình 3.2.2.10 ta có ba mạch logic NOR nối liên tiếp với nhau, giả sử ở emitơ của các tranzito trong mạch 2 có tồn tại một điện trở R_{ND} nào đó (điện trở của đường nối đất). R_D có giá trị nhỏ nên sụt áp trên nó chỉ đáng kể khi có dòng lớn chạy qua, gọi dòng điện cho qua R_{ND} là I_{ND}, như thế nó sẽ sinh ra một điện áp trên R_{ND}, điện áp này chính là điện áp nhiễu đạt ở emitơ, ta gọi là $U_{nh\ e}$, điện áp nhiễu này bằng:

$$U_{nh\ e} = R_{ND} I_{ND}$$

Điện áp này gây tác hại lớn tùy trường hợp cụ thể. Giả sử mạch 1 tắt làm I_{k1} rất nhỏ, nên điện áp ra của mạch 1 U_{ra1} tăng lớn gần bằng E, điện áp U_{ra1} này sẽ đưa vào mạch 2 làm dòng điện ra của mạch 1 I_{ra1} lớn, tức là dòng điện vào của mạch 2 $I_{vào2}$ lớn, làm cho mạch 2 thông, dòng qua nó I_{k2} sẽ rất lớn, tức là dòng I_{ND} lớn, kết quả $U_{nh\ e}$ lớn. Khi xuất hiện $U_{nh\ e}$ sẽ làm cho I_b của mạch 2 giảm, sẽ làm giảm mức bão hòa của mạch 2 (mạch 2 lúc này ở trạng thái thông) và điện ra của mạch 2 U_{ra2} tăng lên sẽ làm cho I vào 3 tăng lên. Đáng lẽ mạch 2 thông sẽ làm cho mạch 3 tắt, nhưng vì I vào 3 tăng lên có thể làm cho mạch 3 chuyển biến nhầm, tức là có thể làm cho mạch 3 thông. Nhiễu này là nhiễu thông tác động lên mạch tắt (mạch 3).



Hình 3.2.2.10. Nhiễu xuất hiện do sự sụt áp trên đường đất

Ngoài ra ta còn phải chú ý đến nhiễu xuất hiện trên điện trở của mạch nguồn cung cấp, vì ở đó có dòng lớn chạy qua nên sẽ gây sụt áp lớn.

Tác dụng của các điện áp nhiễu này lớn hay bé phụ thuộc vào từng trường hợp cụ thể, dạng mạch thế nào và số nguồn cung cấp bằng bao nhiêu.

3.3. HỌ LOGIC RTL (Resistor – Transistor – Logic)

3.3.1. Mạch NOT

Mạch logic điện trở – tranzito (RTL) thuật toán phủ định chỉ thực hiện trên một tranzito. Các IC họ RTL được nuôi bằng nguồn V_{CC} = +3,6V, điện áp vào và ra của mạch chỉ có hai mức:

Mức thấp: L = 0V

Mức cao: H > 1,5V.

Hình 3.3.1.1 trình bày sơ đồ nguyên lý của mạch đảo họ RTL.

Điện áp ra của mạch là điện thế trên colecto của tranzito.

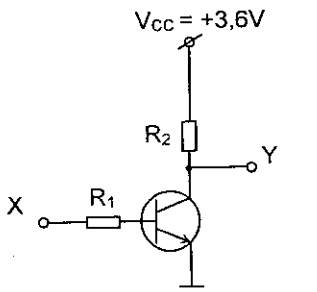
Điện thế lối ra: $V_Y = V_{CC} - I_C R_2$

Khi X = 0: $I_B = 0$ nên $I_C = 0$.

Khi đó: $V_Y = V_{CC}$ = +3,6V, đầu ra ở mức logic cao (Y = 1).

Khi X = 1: I_B tăng nên I_C tăng

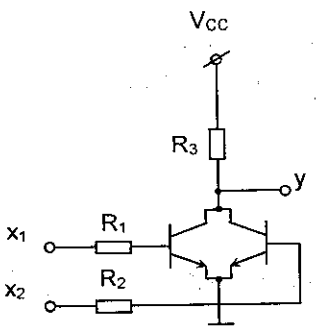
Khi đó: $V_Y = V_{CC} - I_2R_2$, đầu ra ở mức logic thấp (Y = 0)



Hình 3.3.1.1. Mạch đảo họ RTL

Điện áp ra của mạch phụ thuộc vào điện áp vào tuân theo hàm logic: $Y = \overline{X}$.

3.3.2. Mạch NOR



a)

x1	x2	y
L	L	H
H	L	L
L	H	L
H	H	L

b)

Hình 3.3.2.1

a) Mạch NOR họ RTL ; b) Bảng trạng thái.

Sơ đồ mạch như hình 3.3.2.1a.

Nguyên lý hoạt động của mạch như sau:

Khi hai lối vào ở mức "L" thì cả hai tranzito cấm, lối ra ở mức "H".

Khi một trong hai lối vào ở mức "H" thì một trong hai tranzito thông, lối ra ở mức "L".

Khi hai lối vào ở mức "H" thì hai tranzito thông bão hoà, lối ra ở mức "L".

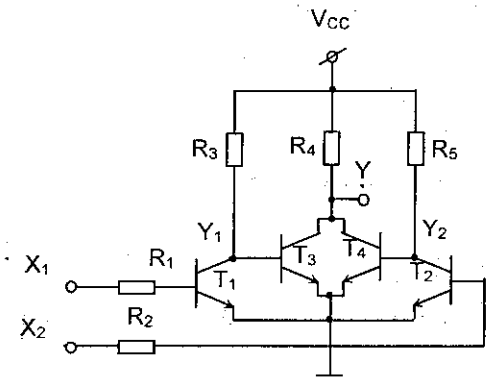
Bảng trạng thái của mạch cho trên hình 3.3.2.1b.

Điện áp ra của mạch phụ thuộc vào điện áp vào tuân theo hàm logic: $Y = \overline{X_1 + X_2}$

3.3.3. Mạch AND

Từ sơ đồ hình 3.3.3.1 ta thấy T_1, T_2 là hai mạch NOT; T_3 và T_4 tạo thành mạch NOR. Như vậy, ta có: $Y = \overline{Y_1 + Y_2} = \overline{\overline{X_1} + \overline{X_2}} = X_1.X_2$

Họ logic RTL hiện nay không được sản xuất nữa, tuy nhiên nó vẫn được dùng trong một số mạch điều khiển.



Hình 3.3.3.1. Mạch AND RTL

3.4. HỌ LOGIC DTL (Diode Transistor Logic)

Mạch logic tổ hợp diode – tranzito (DTL) là loại mạch được dùng rất phổ biến trong kỹ thuật vi điện tử, nhất là trong loại mạch lai màng mỏng.

3.4.1. Mạch NOT

Sơ đồ mạch cho trên hình 3.4.1.1.

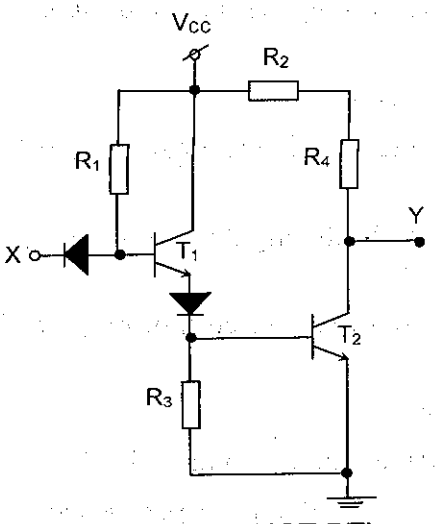
Nguyên lý hoạt động của mạch như sau:

Bình thường nếu lối vào để hở, V_{B1} luôn ở mức cao nên T_1 thông, T_2 thông và Y ở mức logic thấp.

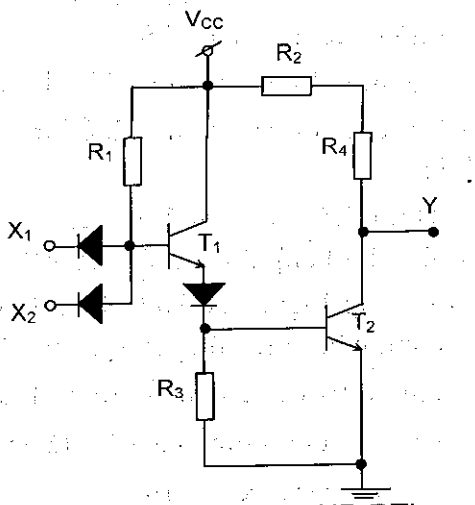
Nếu X = 1, diode không dẫn và V_{B1} ở mức cao dẫn đến T_1, T_2 thông và Y ở mức logic thấp.

Nếu X = 0, diode dẫn và V_{B1} ở mức thấp dẫn đến T_1 cấm, T_2 cấm và Y ở mức cao.

Điện áp ra của mạch phụ thuộc vào điện áp vào tuân theo hàm logic: $Y = \overline{X}$.



Hình 3.4.1.1. Mạch NOT DTL



Hình 3.4.2.1. Mạch NAND DTL

3.4.2. Mạch NAND

Sơ đồ mạch đảo hình 3.4.1.1 nếu ta mắc song song 2 diode với hai lối vào X_1, X_2 như hình 3.4.2.1 ta sẽ được mạch NAND 2 lối vào.

Phân tích tương tự mạch NOT ta thấy chỉ khi cả hai đầu vào X_1 và X_2 cùng ở mức logic cao để cả hai diode cùng cấm thì đầu ra Y mới ở mức logic thấp, còn các trường hợp khác đầu ra Y ở mức logic cao.

Điện áp ra của mạch phụ thuộc vào điện áp vào tuân theo hàm logic: $Y = \overline{X_1 X_2}$.

Nếu ở lối vào ta mắc song song n diode ta sẽ được mạch NAND n lối vào theo hàm logic: $Y = \overline{X_1 X_2 ... X_n}$.

Đặc điểm của họ logic DTL là tốc độ làm việc chậm (f < 1MHz), được dùng trong điện tử công nghiệp, điện tử y tế. Ngày nay, DTL không được sản xuất nữa, trong các thiết bị điện tử có dùng loại vi mạch thuộc họ logic này nếu vi mạch bị hỏng cần phải thay thế ta có thể dùng các vi mạch logic họ TTL có chức năng tương tự.

3.5. HỌ TTL CHUẨN (74XX)

3.5.1. Các đặc điểm của họ TTL

– Các họ 54 /74TTL là các họ IC số thông dụng. Họ 74 được dùng trong thương mại còn họ 54 được dùng trong quân sự. Khác nhau duy nhất của 2 loại này là giới hạn nhiệt độ và nguồn cung cấp.

	Họ 74	Họ 54
Giới hạn nhiệt độ	$0^0C \div 70^0C$	$-55^0C \div 125^0C$
Nguồn cung cấp	$5 \pm 0.25V$	$5 \pm 0.5V$

Nếu V_{CC} vượt quá 7V, chip TTL chắc chắn sẽ bị hỏng, vì thế 7V được gọi là định mức nguồn cực đại tuyệt đối.

– Các mức logic của họ TTL như trên hình 3.5.1.1.

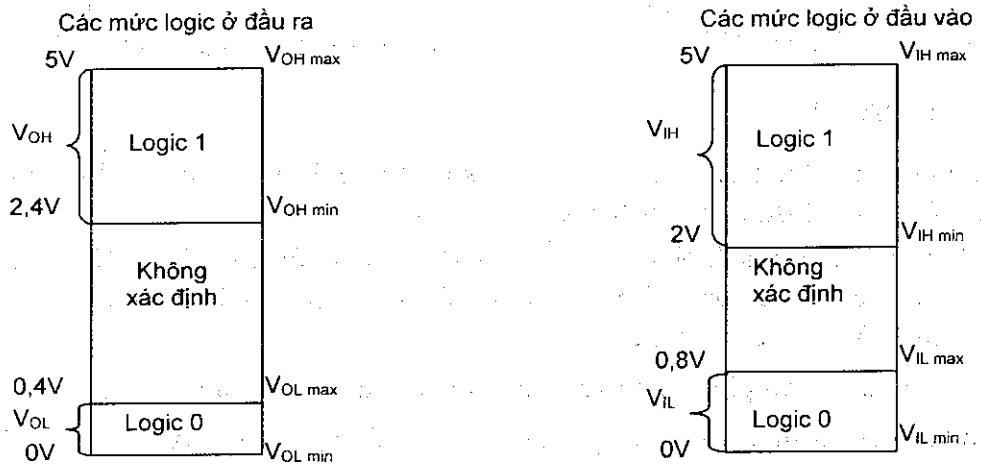
– Khoảng dự trữ chống nhiễu mức thấp và mức cao là như nhau bằng 0,4V (hình 3.5.1.1).

– Dòng DC trung bình I_{CC} của cổng NAND TTL khoảng 2mA, như vậy công suất trung bình P_D bằng 2mA.5V = 10mW.

Mỗi cổng logic tiêu hao rất ít dòng và công suất, tuy nhiên một chip có thể gồm nhiều cổng nên tổng công suất sẽ rất đáng kể.

– Trễ do truyền của phần tử TTL khoảng 10ns.

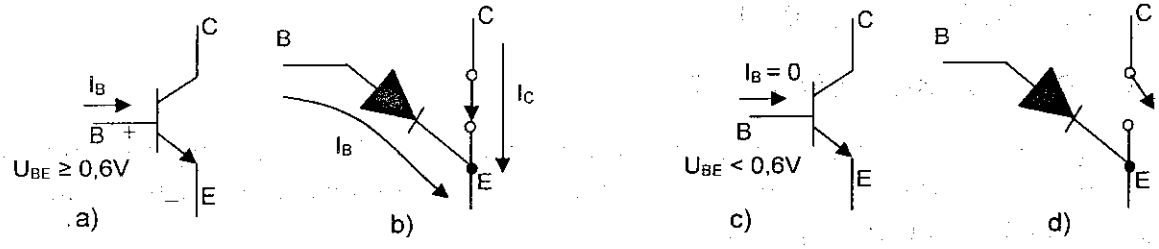
– Tần số xung nhịp của TTL là 35MHz (35 triệu chu kỳ trong mỗi giây).



Hình 3.5.1.1. Các mức logic TTL

3.5.2. Mạch TTL chuẩn

– ***Hoạt động của Tranzito:***



Hình 3.5.2.1. Các trạng thái hoạt động của tranzito ứng với U_{be}

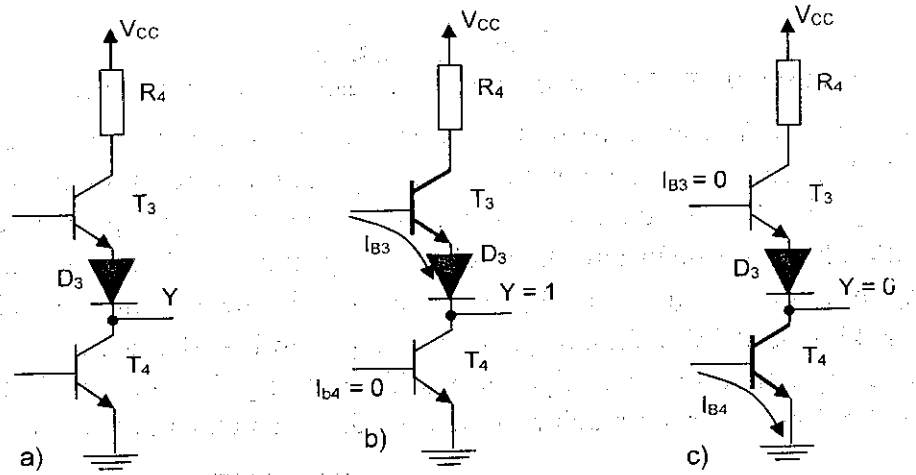
Khi áp điện thế dương lớn hơn 0,6V giữa cực gốc và cực phát của tranzito lưỡng cực npn, dòng cực gốc sẽ chạy. Lúc dòng cực gốc chạy, cực góp của tranzito dẫn

điện, giữa cực góp và cực phát vận hành giống chuyển mạch đóng (hình 3.5.2.1a,b).

Khi áp điện thế nhỏ hơn 0,6V giữa cực gốc và cực phát của tranzito lưỡng cực npn, dòng cực gốc không chạy. Lúc này giữa cực góp và cực phát vận hành giống chuyển mạch mở (hình 3.5.2.1c,d).

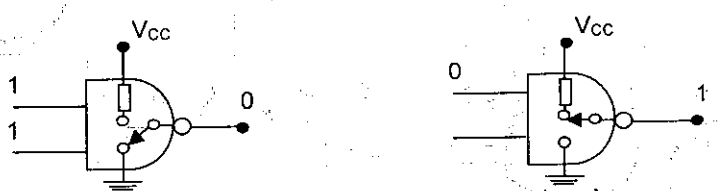
– Mạch đầu ra của phần tử TTL:

Mạch đầu ra của phần tử TTL gồm hai tranzito, một điện trở và một diode (hình 3.5.2.2a), mỗi thời điểm chỉ có một tranzito dẫn điện. Trường hợp T_3 dẫn T_4 khóa, đầu ra Y nối với V_{CC} qua D_3, T_3, R_4, điện thế tại điểm Y so với đất khoảng 3,4V (hình 3.5.2.2b). Trường hợp T_3 khóa T_4 dẫn, đầu ra Y nối đất thông qua T_4 và điện thế tại điểm Y so với đất khoảng 0,2V (hình 3.5.2.2c). Trên các hình vẽ tranzito được vẽ đậm minh họa đang ở trạng thái dẫn.



Hình 3.5.2.2. Mạch đầu ra của phần tử TTL và các trạng thái ra

Như vậy trong mạch đầu ra này, T_3 và T_4 hoạt động như chuyển mạch đóng, nối đầu ra Y với V_{CC} hoặc với đất (hình 3.5.2.3).

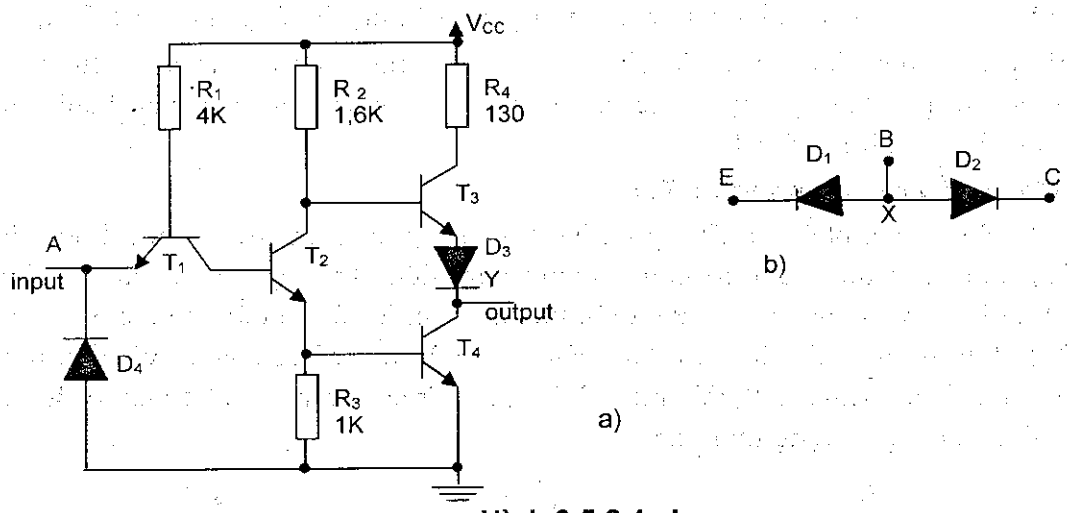


Hình 3.5.2.3. Các trạng thái của đầu ra

Điện trở R_4 giới hạn dòng ra khi Y = 1, nếu không có R_4 có thể xảy ra dòng lớn quá mức. Diode D_3 là một phần của mạch phân cực T_3. Các tranzito T_3 và T_4 của mạch đầu ra này được gọi sắp xếp theo kiểu totem – pole.

– Hoạt động của cổng NOT (TTL):

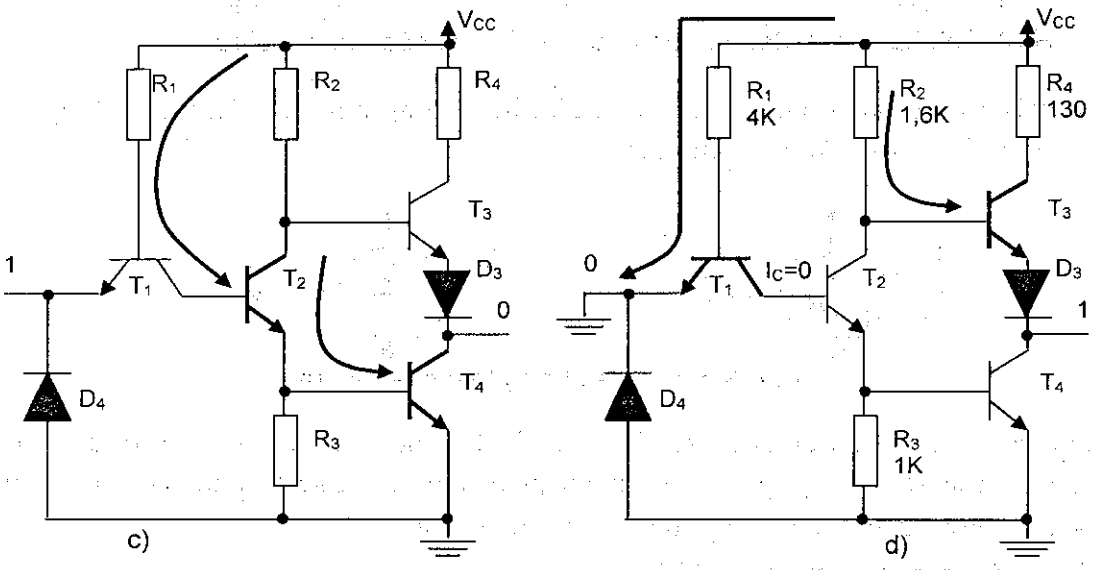
Hình 3.5.2.4a minh họa sơ đồ chi tiết của mạch NOT, hình 3.5.2.4b minh họa sơ đồ tương đương diode của T_1. Trong mạch, tranzito T_1 là tranzito nối với đầu vào, tranzito T_2 được gọi là mạch tách pha.



Hình 3.5.2.4a,b

a) Mạch NOT (TTL) chuẩn ; b) Sơ đồ diode tương đương của T_1.

Khi đầu vào ở mức cao, tranzito T_1 có tiếp giáp B – E của phân cực ngược và tiếp giáp B – C phân cực thuận. Nguồn V_{CC} cung cấp dòng qua R_1, D_2 đến cực B của T_2 kích thích T_2 dẫn. Khi T_2 dẫn, sụt áp trên R_2 khiến cho điện áp tại cực C của T_2 khoảng 0,8V không đủ phân cực thuận cho cả tiếp giáp E – B của T_3 và D_3 nên T_3 khóa; đồng thời dòng từ cực E của T_2 truyền đến cực B của T_4, điện thế tại cực E của T_2 có giá trị khoảng 0,7V phân cực thuận cho tiếp giáp E – B của T_4, T_4 dẫn nên đầu ra Y có mức logic thấp khoảng 0,2V, do trở kháng trạng thái mở của T_4 thấp (1 đến 25Ω), hình 3.5.2.4c. D_3 trong mạch có nhiệm vụ đảm bảo cho T_3 khóa khi T_2 dẫn.



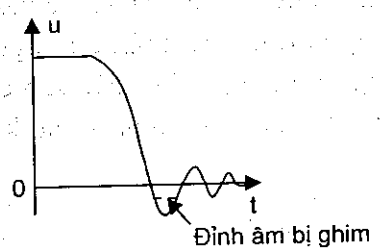
Hình 3.5.2.4c,d. Hoạt động của một cổng NOT (TTL)

Khi đầu vào ở mức thấp, tranzito T_1 có tiếp giáp B – E của phân cực thuận. Nguồn V_{CC} cung cấp dòng qua R_1, qua D_1 qua đầu cuối của đầu vào A nối đất. Điện

thế thuận qua D_1 giữ điện thế tại điểm X ở mức xấp xỉ 0,7V, không đủ để phân cực thuận cho D_2 và tiếp giáp E – B của T_2. Không có dòng cung cấp cho cực B của T_2 nên T_2 khóa, do không có dòng cực góp của T_2 nên điện thế tại cực C của T_2 đủ lớn để phân cực thuận cho tiếp giáp E – B của T_3 và D_3 nên T_3 dẫn, đồng thời không có dòng cung cấp cho cực B của T_4 nên T_4 khóa. Vì vậy, đầu ra Y được nối với V_{CC} qua D_3, T_3 và R_3 và có mức logic cao, hình 3.5.2.4d.

Diode D_4 trong mạch được gọi là diode ghim trên đầu vào TTL. Nhiệm vụ của diode ghim là ngăn không cho điện thế đầu vào xuống mức âm.

Dưới điều kiện bình thường, A không xuống mức âm (logic 1 = +3,4V; logic 0 = +0,2V), các diode ghim vận hành như mạch hở và không tham gia vào hoạt động của mạch. Nhưng ở mạch tốc độ cao, khi mạch chuyển từ 1 sang 0 thường xảy ra hiện tượng gọi chuông (ringing), hình 3.5.2.5. Đây là một dao động không mong muốn, vì nó làm chậm hoạt động của mạch.

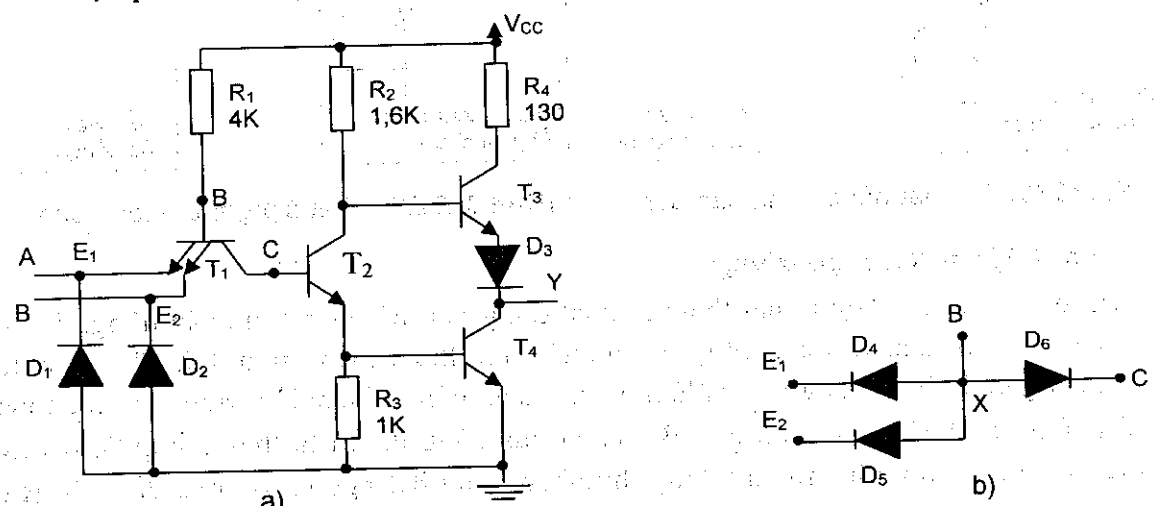


Hình 3.5.2.5. Hiện tượng gọi chuông

Tại đỉnh âm của xung gọi chuông, diode ghim dẫn điện. Diode này sẽ ghim đỉnh âm, giúp giảm bớt hiệu ứng gọi chuông và làm tăng tốc độ của mạch.

– ***Hoạt động của cổng NAND (TTL):***

Hình 3.5.2.6a minh họa sơ đồ chi tiết của mạch NAND, hình 3.5.2.6b minh họa sơ đồ diode tương đương của T_1. Ta thấy tranzito T_1 có hai cực phát, do vậy nó có hai tiếp giáp E – B dùng để mở tranzito T_1. Tranzito đầu vào nhiều cực phát có thể có đến 8 cực phát cho mỗi cổng NAND.

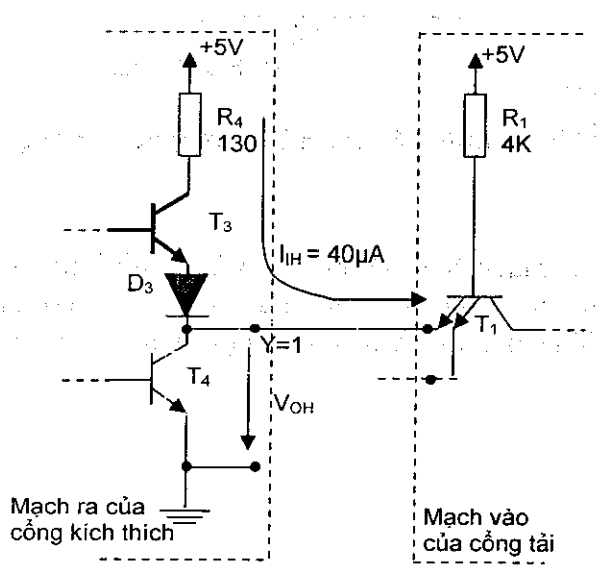


Hình 3.5.2.6.
a) Mạch NAND (TTL) chuẩn ; b) Sơ đồ diode tương đương của T_1.

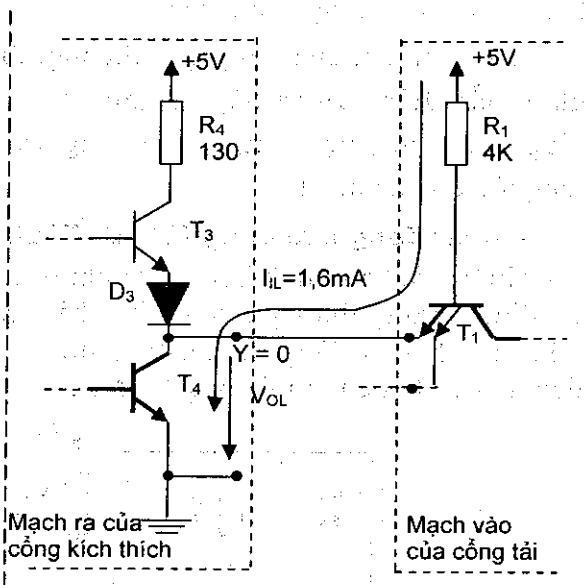
Phân tích mạch NAND tương tự như mạch NOT ta đã phân tích ở trên. Trong mạch này, chỉ khi cả hai đầu vào A và B đều ở mức cao thì cả hai diode D_4, D_5 phân cực ngược và D_6 phân cực thuận nên T_2 dẫn và đầu ra ở mức thấp; Khi có ít nhất một đầu vào ở mức thấp thì sẽ có ít nhất một diode phân cực thuận, dẫn đến điện thế thuận của diode tại điểm X không đủ phân cực thuận cho D_6 và tiếp giáp E – B của T_2 nên T_2 khóa dẫn đến đầu ra ở mức cao. Nếu không có tải nối từ điểm Y xuống đất, V_{OH} sẽ nằm trong khoảng từ 3,4V đến 3,8V do V_{CC} trừ đi hai lần sụt áp 0,7V (tiếp giáp E – B của T_3 và D_3), điện thế này sẽ giảm nếu có tải.

– Hoạt động cung cấp dòng:

Khi đầu ra TTL ở trạng thái cao nó hoạt động như bộ phận cung cấp dòng, do nó cung cấp dòng từ đầu ra của cổng kích thích cho đầu vào của cổng tải, hình 3.2.5.7. Dòng này là dòng rò phân cực ngược rất nhỏ (khoảng 40μA). T_3 thực hiện hành động cung cấp dòng (I_{IH}) cho đầu vào của cổng tải, vì thế T_3 được gọi là tranzito cung cấp dòng hay tranzito kéo lên, do nó kéo điện thế đầu ra lên trạng thái cao.



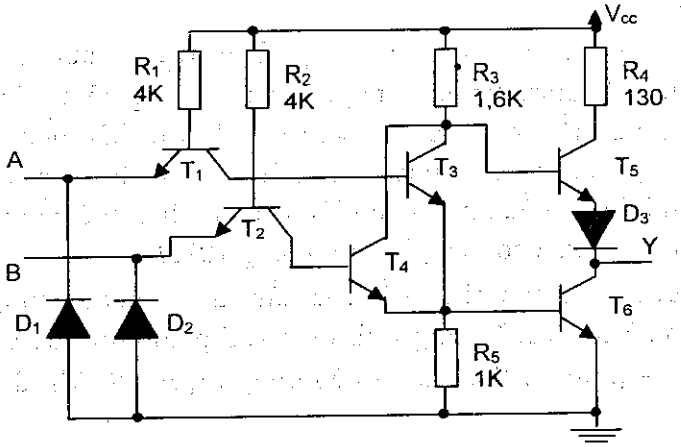
Hình 3.5.2.7. Hoạt động cung cấp dòng



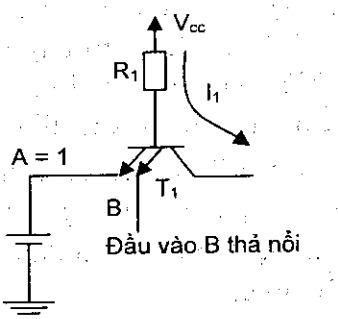
Hình 3.5.2.8. Hoạt động thu nhận dòng

– Hoạt động thu nhận dòng:

Khi đầu ra TTL ở trạng thái thấp nó hoạt động như bộ phận thu nhận dòng, do nó thu nhận dòng từ đầu vào của cổng tải mà nó đang kích thích, hình 3.5.2.8. Tranzito T_4 của cổng kích thích đang mở và làm ngắn mạch điểm Y đến đất. Mức điện thế thấp tại Y phân cực thuận tiếp giáp E – B của T_1 của cổng tải và ta thấy dòng điện chạy ngược qua T_4. T_4 thực hiện hành động thu nhận dòng bắt nguồn từ dòng đầu vào (I_{IL} khoảng 1,6mA) của cổng tải, vì thế T_4 được gọi là tranzito thu nhận dòng hay tranzito kéo xuống, do nó kéo điện thế đầu ra xuống trạng thái thấp.



Hình 3.5.2.9. Mạch NOR (TTL) chuẩn



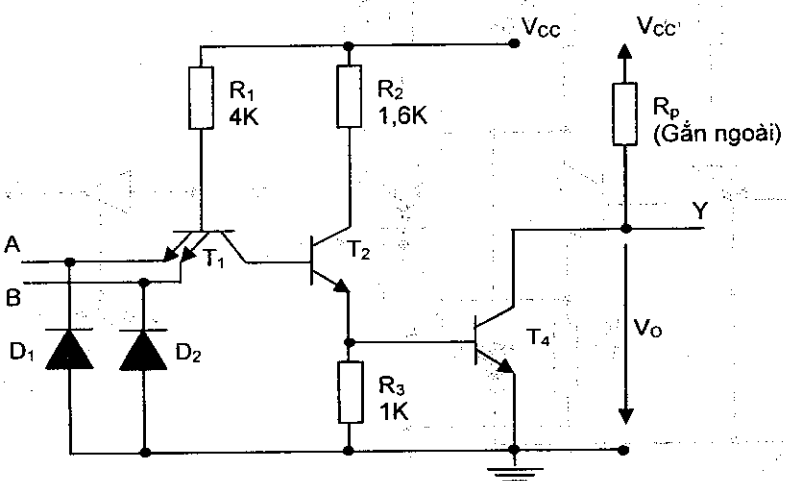
Hình 3.5.2.10. Đầu vào TTL thả nổi

– Hoạt động của cổng NOR (TTL):

Hình 3.5.2.9 minh họa mạch bên trong của cổng NOR TTL. Mạch NOR không sử dụng tranzito nhiều cực phát, từng đầu vào được áp đến cực E của một tranzito riêng biệt. Trong mạch, hai tranzito T_3 và T_4 (mạch tách pha) có cực C được nối chung với nhau và nối với cực B của T_5, cực E được nối chung với nhau và nối với cực B của T_6. Vì thế, chỉ cần có ít nhất một tranzito dẫn thì dẫn đến đầu ra Y ở mức thấp.

– Đầu vào thả nổi:

Đối với mạch TTL, đầu vào thả nổi vận hành như logic 1. Ta thấy rằng, để có đầu vào bằng 0 thì phải có đường dẫn đến đất cho dòng I_1 nhưng đầu vào thả nổi không cung cấp đường dẫn như thế, vì vậy chúng vận hành như đang có mức logic 1 tại đầu vào thả nổi, hình 3.5.2.10. Hiệu điện thế giữa đầu vào thả nổi và đất xấp xỉ bằng 1,7V, mặc dù đầu vào thả nổi được xem như mức logic 1, nhưng hiệu điện thế lại nằm trong khoảng không xác định do vậy nên tránh để các đầu vào thả nổi. Chúng ta có thể xử lý các đầu vào không sử dụng đến bằng cách nối chúng với đầu vào được sử dụng, hoặc nối với V_{CC} hoặc nối với đất tùy thuộc vào chức năng của phần tử.



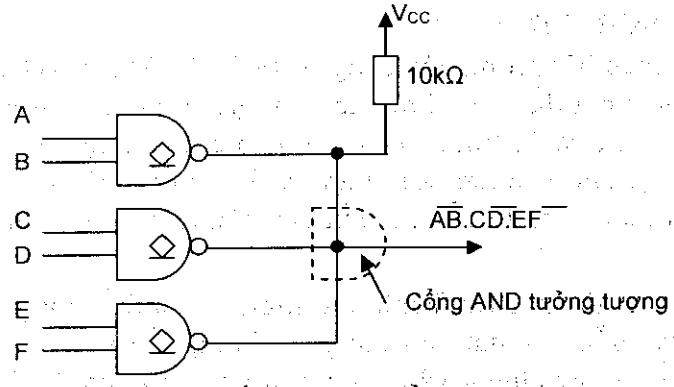
Hình 3.5.2.11. Mạch TTL hở cực góp với điện trở ngoài kéo lên.

– Các đầu ra hở cực góp (OC):

Một số mạch TTL được thiết kế với các đầu ra hở cực góp. Cấu trúc hở cực góp loại bỏ T_3, D_3 và R_4 của mạch ra, đầu ra Y được lấy tại cực góp của T_4 đang để hở. Để mạch hoạt động được chuẩn xác thì phải mắc một điện trở kéo lên (R_p) từ bên ngoài, hình 3.5.2.11. Điện trở này không thuộc mạch bên trong của thiết bị TTL, là điện trở riêng biệt mà ta phải nối với đầu ra linh kiện, nếu không có điện trở này điện thế đầu ra có thể ở mức không xác định. R_p thường được chọn có giá trị khoảng 10kΩ.

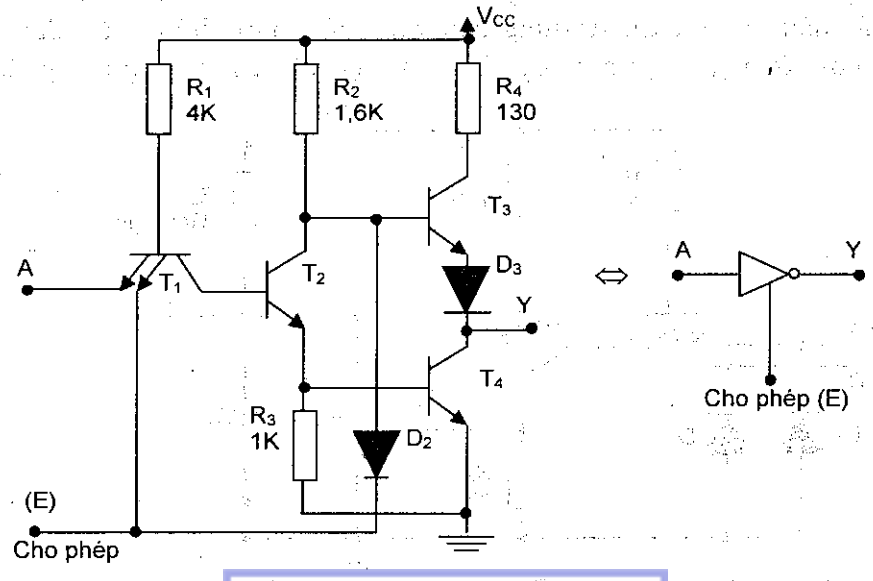
Với các linh kiện có đầu ra hở cực góp, cho phép nối các đầu ra với nhau tương đương với phép toán logic AND, hình 3.5.2.12. Những linh kiện OC phải chịu tốc độ chuyển mạch chậm hơn đầu ra totem – pole, vì thế không nên dùng mạch OC khi đòi hỏi tốc độ cao.

Ký hiệu đầu ra hở cực góp là hình thoi có gạch dưới.



Hình 3.5.2.12. Nối dây theo cổng AND dùng các cổng hở cực góp

– TTL ba trạng thái (Tristate):



Hình 3.5.2.13. Bộ đảo TTL ba trạng thái

Cấu hình ba trạng thái là dạng cấu hình thứ ba của đầu ra TTL. Nó cho phép nối các đầu ra với nhau của cấu hình totem – pole. Gọi là cấu hình ba trạng thái vì cho phép có ba trạng thái đầu ra: cao, thấp và trở kháng cao (Hi – Z). Trạng thái Hi –Z là điều kiện trong đó cả hai tranzito trong cấu hình totem – pole đều đóng, do vậy đầu ra có trở kháng cao so với đất và với V_{CC} (đầu cuối bị hở hay thả nổi).

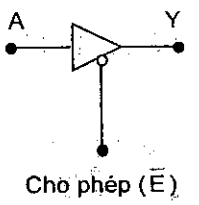
Hình 3.5.2.13 mô tả mạch cho bộ đảo ba trạng thái. Mạch có hai đầu vào: A là đầu vào logic thông thường, E là đầu vào cho phép có thể tạo trạng thái Hi – Z.

+ Với E = 1, mạch hoạt động như một bộ đảo bình thường, do tiếp giáp E_2 – B của T_1 và D_2 phân cực ngược.

+ Với E = 0, mạch chuyển sang trạng thái Hi – Z, do mức thấp tại E phân cực thuận cho tiếp giáp E_2 – B của T_1 làm cho T_2 khóa và dẫn đến T_4 khóa. Đồng thời mức thấp tại E cũng phân cực thuận cho D_2, dẫn dòng tại cực gốc của T_3 nối đất nên T_3 cũng đóng.

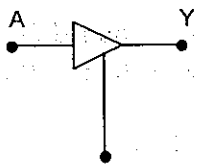
Với cả hai tranzito totem – pole đều ở trạng thái không dẫn dòng, đầu ra là mạch hở. Các IC ba trạng thái cho phép nối các đầu ra với nhau mà vẫn đảm bảo tốc độ chuyển mạch, tuy nhiên mỗi lần chỉ có một đầu vào E được cho phép.

– ***Bộ đệm ba trạng thái (Tristate Buffer):***



Cho phép ($\bar{E}$)

$\bar{E}$	Y
0	A
1	Hi – Z



Cho phép (E)

E	Y
1	A
0	Hi – Z

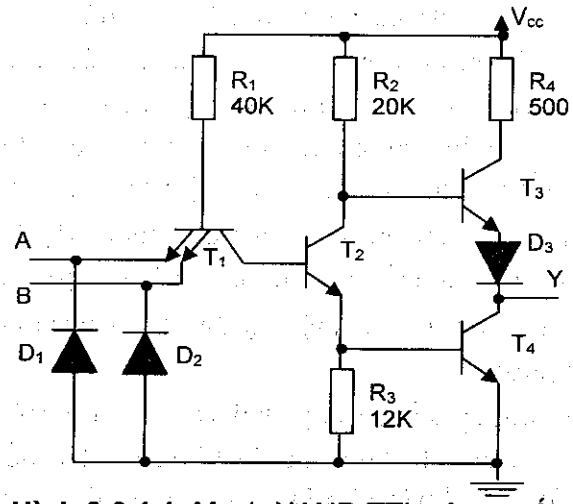
Hình 3.5.2.14. Bộ đệm không đảo ba trạng thái

Bộ đệm ba trạng thái là mạch dùng để kiểm soát đường đi của tín hiệu logic từ đầu vào đến đầu ra. Hình 3.5.2.14 mô tả ký hiệu và chức năng của các bộ đệm ba trạng thái.

3.6. HỌ TTL CẢI TIẾN

3.6.1. TTL công suất thấp (74LXX)

Với điện thế nguồn cố định +5V, nên có thể giảm công suất bằng cách hạ thấp dòng điện. Điện trở trong mạch 74LXX lớn gấp 5 đến 10 lần điện trở trong mạch 74XX, hình 3.6.1.1.



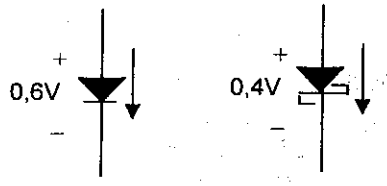
Hình 3.6.1.1. Mạch NAND TTL công suất thấp

3.6.2. TTL Schottky (74SXX)

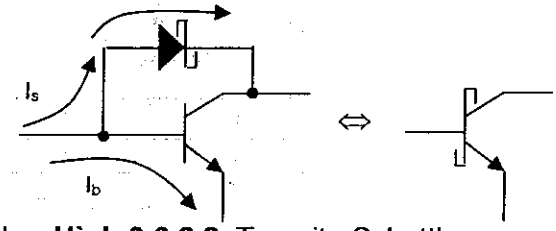
Với các họ 74, 74L đều hoạt động theo nguyên tắc chuyển mạch bão hòa, hoạt động này gây nên một khoảng trễ trong thời gian các tranzito chuyển từ trạng thái dẫn sang khóa, và giới hạn tốc độ chuyển mạch.

– Diode Schottky:

Diode thông thường sẽ dẫn điện khi áp điện thế thuận khoảng 0,6V, còn đối với diode Schottky chỉ khoảng 0,4V, hình 3.6.2.1.



Hình 3.6.2.1. Diode thường và diode Schottky



Hình 3.6.2.2. Tranzito Schottky

– Tranzito Schottky:

Họ 74SXX giảm bớt khoảng thời gian trễ này khi không cho tranzito tiến sâu vào trạng thái bão hòa, bằng cách dùng tranzito Schottky (SBD) nối giữa cực gốc và cực góp của mỗi tranzito, hình 3.6.2.2. Khi đó chỉ dòng điện cần thiết để kích hoạt tranzito mới chạy vào cực gốc, phần còn lại (dòng gây hiện tượng bão hòa) chạy qua diode. Như vậy, tranzito ngừng dẫn điện nhanh hơn tranzito không có diode.

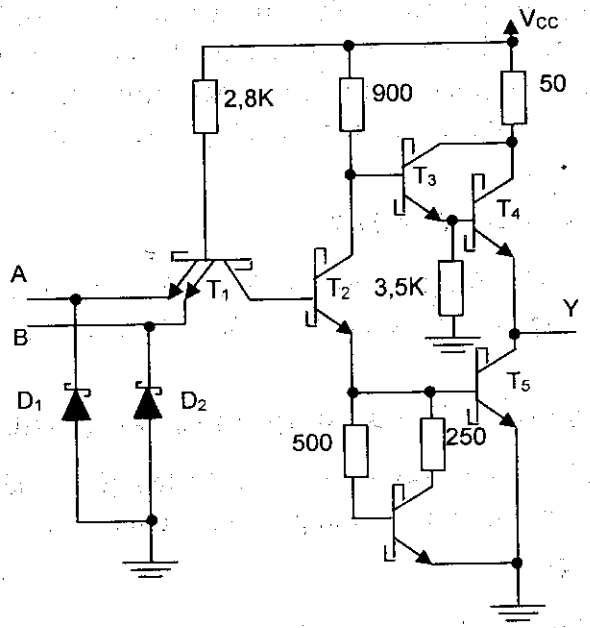
– Mạch Schottky:

Hình 3.6.2.3 minh họa một cổng NAND Schottky. Các yếu tố giúp cải thiện tốc độ, đó là:

+ Sử dụng tranzito Schottky: Do tranzito hoạt động không bão hòa nên tốc độ được tăng lên so với TTL chuẩn, tuy nhiên công suất tiêu hao lại lớn hơn.

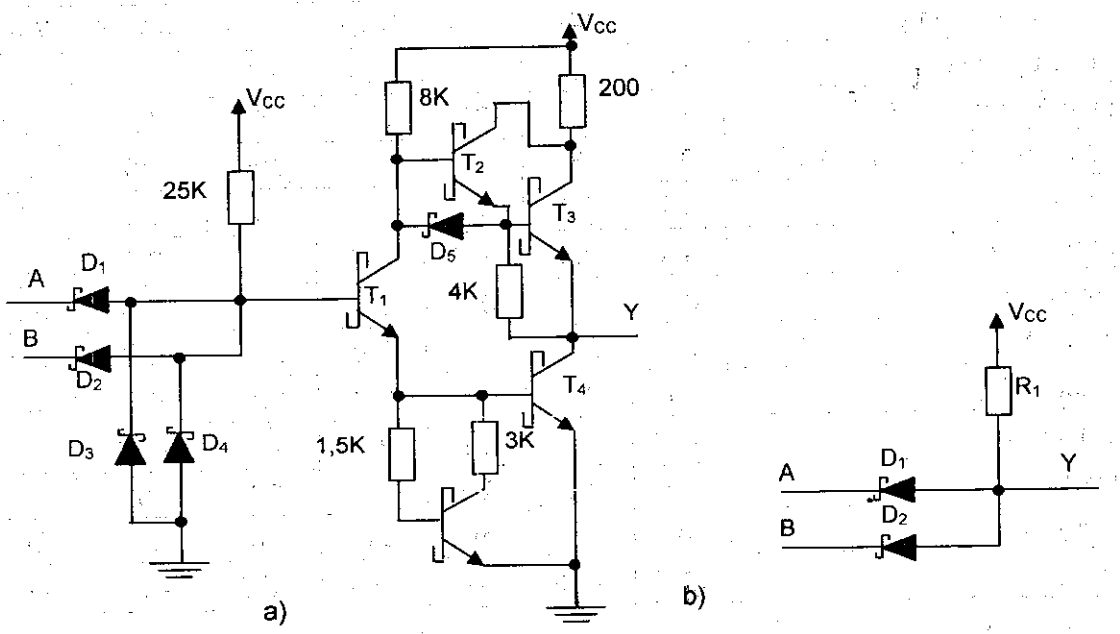
+ Cấu hình Darlington: Trước khi dòng điện từ mạch tách pha T_2 đến được T_4, nó được T_3 khuếch đại, kích thích T_4 mạnh hơn và mở/khóa T_4 nhanh hơn.

+ Điện trở thấp: Các giá trị điện trở trong mạch Schottky thấp hơn giá trị điện trở trong mạch TTL chuẩn. Tuy nhiên, điện trở nhỏ thì dòng cao dẫn đến mức tiêu hao công suất gia tăng.



Hình 3.6.2.3. Mạch NAND sử dụng tranzito Schottky.

3.6.3. TTL Schottky công suất thấp (74LSXX)



Hình 3.6.3.1.
a) Mạch Schottky công suất thấp; b) Mạch AND dùng.

Trong họ 74LSXX, ta có được tốc độ gần đạt đến tốc độ của họ 74XX, nhưng định mức công suất lại xấp xỉ với 74LXX.

– Mạch Schottky công suất thấp:

Logic diode được dùng trong phần quyết định của mạch Schottky công suất thấp, thay cho tranzito đa cực phát (hình 3.6.3.1a). Diode D_1, D_2 và điện trở R_1 vận hành như cổng AND (hình 3.6.3.1b), phần đầu ra của mạch tương tự họ 74SXX. 74LSXX có công suất tiêu hao thấp hơn do dùng điện trở có giá trị lớn hơn 74SXX nhưng tốc độ chậm hơn 74SXX.

3.6.4. TTL Schottky nâng cao (74ASXX)

74ASXX có tốc độ tăng lên rõ rệt so với 74SXX, với công suất tiêu hao thấp hơn nhiều. 74ASXX còn có những cải tiến khác, bao gồm yêu cầu dòng vào thấp (I_{IL}, I_{IH}) do đó hệ số tải lớn hơn 74SXX.

3.6.5. TT Schottky công suất tiêu hao thấp nâng cao (74ALSXX)

74ALS là phiên bản cải tiến của 74LS về cả tốc độ lẫn công suất tiêu hao. Do giá thành khá cao nên 74ALS thường được sử dụng trong các mạch đòi hỏi tốc độ cao.

3.6.6. TTL nhanh (74FXX)

Đây là họ TTL mới, vận dụng kỹ thuật chế tạo mạch tích hợp kiểu mới làm giảm bớt điện dung giữa các linh kiện để rút ngắn thời gian trễ do truyền.

3.6.7. So sánh đặc điểm của các họ TTL

Các thông số	74	74L	74S	74LS	74AS	74ALS	74F
Trễ do truyền (ns)	10	33	3	9,5	1,7	4	3
Công suất tiêu hao (mW)	10	1	20	2	8	1,2	6
Tốc độ xung nhịp max (MHz)	35	3	125	45	200	70	100
Hệ số tải	10	20	10	20	40	20	33
V_{OH} min (V)	2,4	2,4	2,7	2,7	2,5	2,5	2,5
V_{OL} max (V)	0,4	0,4	0,5	0,5	0,5	0,4	0,5
V_{IH} min (V)	2,0	2,0	2,0	2,0	2,0	2,0	2,0
V_{IL} max (V)	0,8	0,8	0,8	0,8	0,8	0,8	0,8
I_{OH} (mA)	–0,4		–1	–0,4	–2	–0,4	–1
I_{OL} (mA)	16	3,6	20	8	20	8	20
I_{IH} (μA)	40		50	20	20	20	20
I_{IL} (mA)	–1,6	0,18	–2	–0,4	–0,5	–0,1	–0,6

Ví dụ 3.6.7.1. Xê – ri TTL nào có thể kích thích nhiều đầu vào linh kiện nhất thuộc cùng xê – ri?

Giải: Dựa vào bảng 3.6.7.1 ta thấy xê – ri 74AS có hệ số tải cao nhất.

3.6.8. Xác định hệ số tải

Để xác định một đầu ra IC có khả năng kích thích được bao nhiêu đầu vào khác nhau, ta cần biết khả năng kích thích dòng của đầu ra (I_{OH}, I_{OL}) và yêu cầu dòng của mỗi đầu vào (I_{IL}, I_{IH}). Thông tin này luôn được giới thiệu trong bảng tra cứu dữ liệu các IC.

Hệ số tải được xác định theo công thức:

Hệ số tải mức thấp = I_{OL}(max)/ I_{IL} (max).

Hệ số tải mức cao = I_{OH}(max)/ I_{IH} (max).

Nếu hệ số tải mức thấp và mức cao là như nhau thì đó chính là hệ số tải của phần tử, nếu hệ số tải ở hai mức không bằng nhau thì hệ số tải được chọn sẽ là giá trị nhỏ hơn trong hai giá trị đó.

Trong các hệ thống kỹ thuật số có khuynh hướng kết hợp giữa nhiều họ logic khác nhau, khi đó cách xác định hệ số tải như sau:

Bước 1: Cộng gộp I_{IH} của tất cả các đầu vào nối đến một đầu ra. Tổng này phải thấp hơn giá trị chỉ định I_{OH} của đầu ra.

Bước 2: Cộng gộp I_{IL} của tất cả các đầu vào nối đến một đầu ra. Tổng này phải thấp hơn giá trị chỉ định I_{OL} của đầu ra.

Ví dụ 3.6.8.1. Một đầu ra cổng NAND 74LS00 đang kích thích 3 đầu vào 74SXX và một đầu vào 74XX. Kiểm tra xem có vấn đề gì về tải không?

Giải:

1.Cộng tất cả các giá trị I_{IH} của các cổng tải: 3.(50μA)+1.(40μA) = 190 μA .

I_{OH} của đầu ra 74LS là 0,4mA, lớn hơn tổng các I_{IH} của tải nên không gây ra vấn đề gì khi đầu ra cao.

2. Cộng tất cả các giá trị I_{IL} của các cổng tải: 3.(2mA)+1.(1,6mA) = 7,6 mA .

I_{OL} của đầu ra 74LS là 8mA, lớn hơn tổng các I_{IL} của tải nên cũng không gây ra vấn đề gì khi đầu ra thấp.

3.6.9. Ký hiệu mạch logic họ TTL

– Ký hiệu: SN5402

SN7402

– Cách đọc:

+ SN: hãng sản xuất Texas Intruments

+ Hai số đầu chỉ nhiệt độ làm việc:

74: $0^0C \div 70^0C$

54: $-55^0C \div 125^0C$

+ Hai số sau chỉ chức năng logic:

02: Mạch NOR (có 4 cổng NOR 2 lối vào V_{CC} = +5V).

– Hoặc: SN7400

00: mạch NAND 2 lối vào (4 NAND, V_{CC} = +5V)

7404/74LS04: gồm 6 mạch NOT, V_{CC} = +5V.

7408/74LS08: gồm 4 AND, V_{CC} = +5V.

7432: gồm 4 cổng OR.

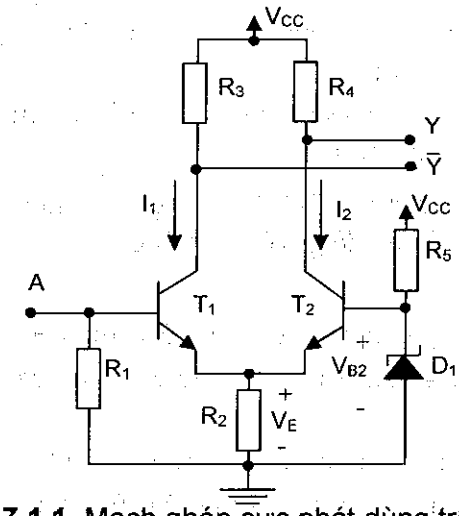
7486: gồm 4 cổng XOR.

3.7. HỌ LOGIC ECL

Họ logic này gọi là logic ghép cực phát (Emitter – Coupled Logic), hoạt động theo nguyên tắc chuyển đổi dòng. ECL rất nhanh nên được dùng trong thiết bị xử lý dữ liệu và truyền thông tốc độ cao.

3.7.1. Mạch ECL

Cụm từ "ghép cực phát" được giải thích dựa vào điện trở R_2 ở hình 3.7.1.1. R_2 nằm trong mạch cực phát của cả T_1 lẫn T_2.



Hình 3.7.1.1. Mạch ghép cực phát dùng trong logic ECL

– Trường hợp đầu vào bằng 0:

Khi A = 0, cực gốc của T_1 nối đất và T_1 không dẫn điện. Với dòng nhỏ chạy qua R_3, điện thế tại $\overline{Y}$ đạt đến V_{CC}, $\overline{Y}$ = 1. Với đường dẫn tín hiệu từ đầu vào A đến đầu ra $\overline{Y}$, mạch vận hành như một bộ đảo.

D_1 là diode Zener, duy trì điện thế tại cực gốc của T_2 không đổi.

Điện thế cực gốc – cực phát của T_2: $V_{BE2} = V_{B2} - V_E$.

Với A = 0, dòng chạy qua T_2, qua R_2 và giảm dần, nên V_E giảm theo dẫn đến V_{BE2} tăng lên. Khi điện thế cực gốc tăng, T_2 dẫn điện, dòng qua R_4 tăng. Sụt thế lớn qua R_4 làm cho điện thế tại đầu ra Y giảm và ở mức thấp.

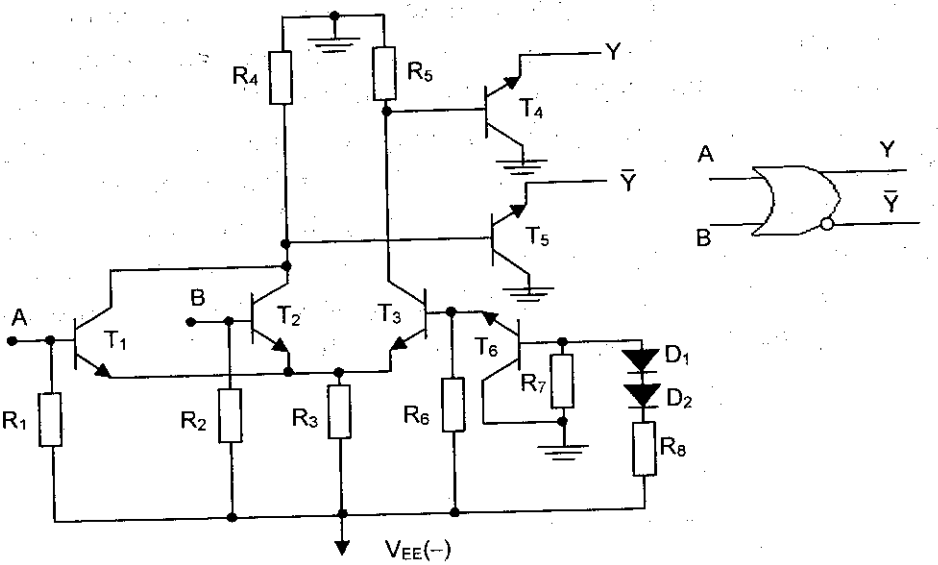
– Trường hợp đầu vào bằng 1:

Khi A = 1, hoạt động xảy ra hoàn toàn ngược lại. Điện thế dương tại cực gốc của T_1 làm tăng dòng qua T_1 và R_2, dòng này gây sụt thế trên R_3 nên điện thế tại $\overline{Y}$ giảm và ở mức thấp. Khi đó V_E tăng làm cho V_{BE2} giảm nên T_2 khóa. Với dòng nhỏ chạy qua R_4, sụt thế trên R_4 giảm làm cho điện thế tại Y tăng và ở mức cao.

Như vậy, mạch ECL luôn có hai đầu ra bù nhau, vì thế không cần gắn thêm bộ đảo. Tranzito vận hành ở chế độ không bão hòa nên thời gian trễ do truyền của mạch ngắn.

3.7.2. Sơ đồ mạch ECL hoàn chỉnh

Hình 3.7.2.1 minh họa một cổng OR/NOR ECL. Tranzito T_4 và T_5 là phần tử kích thích đầu ra. Do chỉ sử dụng một tranzito tại mỗi đầu ra nên đó chính là đầu ra hở cực góp.



Hình 3.7.2.1. Mạch OR/NOR (ECL)

Các cực góp của tất cả các tranzito trong mạch OR/NOR đều nối đất trực tiếp hoặc thông qua các điện trở. Để kháng nhiễu, mạch này vận hành với nguồn âm. Có thể sử dụng các mức điện thế trong khoảng –3V đến –8V, nhưng mức chỉ định là –5,2V.

Điện thế nguồn âm dẫn đến điện thế tín hiệu âm. Với V_{CC} = –5,2V, các mức điện thế nhỏ hơn –1,75V được xem là logic 0, các mức điện thế dương hơn –0,9V được xem là mức logic 1.

Mạch này cho phép mở rộng thêm các đầu vào bằng cách mắc song song các tranzito với tranzito T_1.

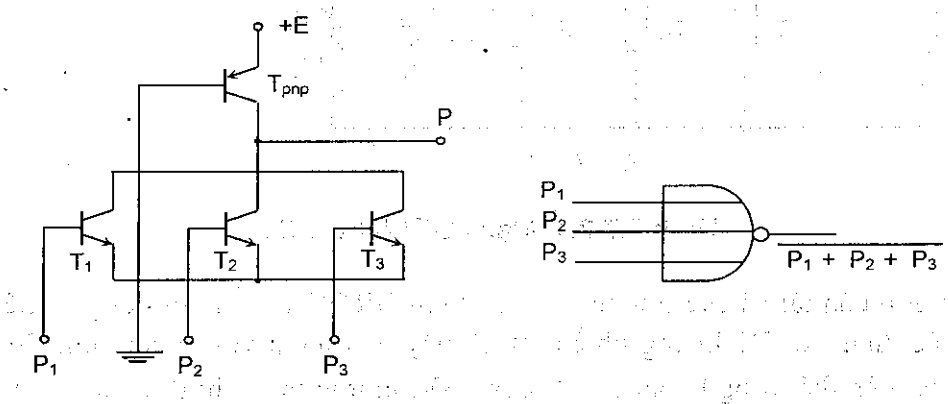
3.7.3. Đặc điểm kỹ thuật của ECL

– Trễ do truyền: 1ns
– Tần số xung nhịp: 400MHz
– Công suất tiêu hao: 25mW
– Hệ số tải: 25
– Khoảng dự trữ chống nhiễu: 250mV.

3.8. HỌ LOGIC IIL

Mạch logic tổ hợp phun ký hiệu là IIL (Integrated Injection Logic). Mạch IIL mới xuất hiện năm 1972 do hai hãng Philips (Hà Lan) và IRM (Đức) đồng thời phát minh ra.

Hình 3.8.1.1 là sơ đồ căn bản mạch IIL. Về mặt nguyên tắc hoạt động, nó giống như mạch RCTL, chỉ khác là điện trở R_k được thay bằng tranzito p – n – p (T_{pnp}). Dòng điện chạy trong mạch ở cả hai trạng thái 0 và 1 đều là dòng lỗ trống phun từ emitơ của tranzito p – n – p. Tiếp xúc emitơ – bazơ của T_{pnp} luôn được phân cực thuận mạnh vì emitơ của nó nối với nguồn dương, còn bazơ của nó nối với nguồn âm, như vậy điện áp nguồn cung cấp cũng chính là điện áp phân cực thuận cho tiếp xúc E – B của tranzito p – n – p. Dòng để làm thông dòng tranzito vào của tầng sau cũng là dòng phun của T_{pnp} (gọi dòng phun này là I_p). Muốn làm thông tin cậy tầng sau thì dòng I_p phải đủ lớn. Khi số tải lớn thì dòng I_p càng phải lớn. Ví dụ ở hình 3.8.1.2: số tải của mạch một là hai tải (một đầu vào của mạch 2 là một tải; một đầu vào của mạch 3 là một tải). Giả sử dòng phun là I_p, lúc đó dòng bazơ của mỗi tranzito vào đó sẽ là $I_p/2$, để làm thông các tranzito này dòng I_p phải đủ lớn.



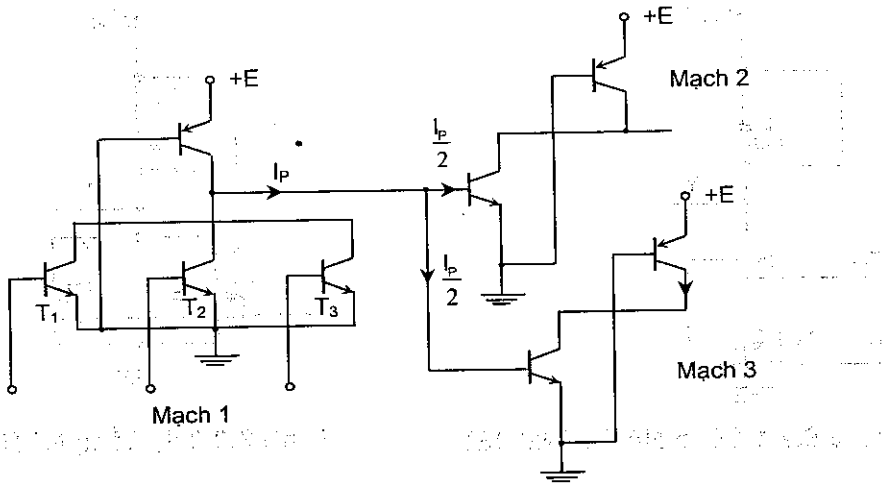
Hình 3.8.1. Sơ đồ nguyên lý của mạch IIL và ký hiệu logic của nó

Muốn dòng I_p tăng lớn ta chỉ cần tăng điện áp phân cực thuận cho tiếp xúc B – E của T_{pnp}, chính là điện áp nguồn E. Khi chỉ cần tăng E lên khoảng 60mV thì dòng I_p sẽ tăng 10 lần.

Vì vậy khi dùng mạch IIL điện áp nguồn E chỉ cần cỡ 0,7V. Khi dùng điện áp nguồn cấp cỡ như vậy, thì điện dung ký sinh cũng nhỏ (vì điện dung ký sinh ở đây là điện dung hàng rào của tiếp xúc p – n khi phân cực ngược) như thế tác động nhanh của mạch được nâng cao mà lúc đó công suất tiêu thụ của mạch vẫn nhỏ.

Về mặt chế tạo, mạch IIL chế tạo nhanh hơn mạch TTL. Mạch IIL mất 5 mask (mặt nạ) trong khi đó mạch TTL mất 7 mask (mạch dùng tranzito trường MOS kênh p mất 4 mask).

Mạch IIL có triển vọng lớn trong việc chế tạo mạch cỡ lớn.

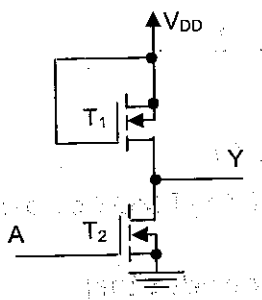


Hình 3.8.1.2. Nối liên tiếp các mạch IIL

3.9. VI MẠCH NMOS

3.9.1. Cổng NOT (NMOS)

Hình 3.9.1.1 minh họa cổng NOT (NMOS), mạch chứa 2 MOSFET kênh n. T_1 gọi là MOSFET tải, cực cổng của T_1 được nối cố định đến V_{DD} do đó nó luôn ở trạng thái dẫn và hoạt động như một điện trở tải có giá trị R_{ON}. T_2 là MOSFET chuyển mạch , T_2 sẽ dẫn hoặc khóa tùy thuộc vào điện thế tại A. Khi A ở mức logic 1, T_2 dẫn và đầu ra ở mức logic 0, còn khi A ở mức logic 0 thì T_2 khóa và đầu ra ở mức logic 1.

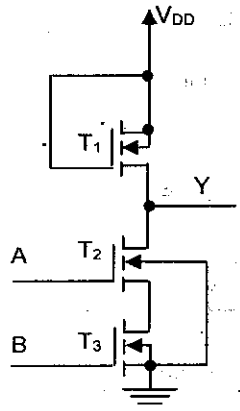


V_A	T_1	T_2	V_Y
0V	R_{on} = 100kΩ	$R_{off} = 10^{10}$ Ω	+5V
+5V	R_{on} = 100kΩ	R_{on} = 1kΩ	+0,05V

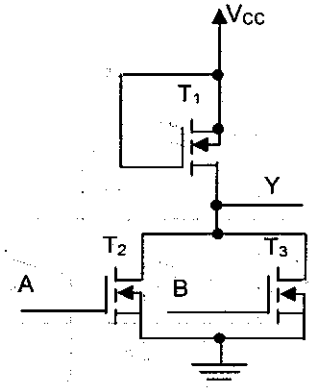
Hình 3.9.1.1. Cổng NOT (NMOS)

3.9.2. Cổng NAND NMOS

Hình 3.9.2.1 minh họa cổng NAND (NMOS), mạch chứa 3 MOSFET kênh n. T_2 và T_3 được gọi là MOSFET chuyển mạch nối tiếp, được kiểm soát bằng mức điện thế tại đầu vào A và B. Nếu A hoặc B ở mức logic 0 (0V), MOSFET tương ứng đóng, khi đó đầu ra Y ở mức logic 1 (+5V). Khi cả A và B ở mức logic 1, cả T_2 và T_3 đều mở, do vậy đầu ra Y ở mức logic 0.



Hình 3.9.2.1.Cổng NAND (NMOS)



Hình 3.9.3.1. Cổng NOR (NMOS)

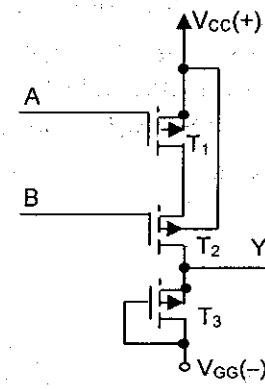
3.9.3. Cổng NOR (NMOS)

Hình 3.9.3.1 minh họa cổng NOR (NMOS), trong đó T_2 và T_3 làm chuyển mạch song song. Khi A hoặc B có mức logic 1, MOSFET tương ứng mở, đầu ra ở mức logic 0. Khi cả hai đầu vào A và B đều ở mức logic 0, cả T_2 và T_3 đều đóng, do đó đầu ra ở mức logic 1.

Các cổng AND, OR được tạo thành bằng cách kết hợp cổng NAND, NOR với cổng NOT.

3.10. VI MẠCH PMOS

Hình 3.10.1.1 minh họa cổng NOR PMOS, mạch chứa 3 MOSFET kênh p. T_3 được nối cố định với V_{gg} có điện thế âm nên T_3 luôn dẫn, nó đóng vai trò như một điện trở tải. Khi đầu vào A hoặc B ở mức logic 1, MOSFET tương ứng khóa và đầu ra được nối với V_{gg} nên có mức logic 0. Khi cả A và B cùng ở mức logic 0, cả T_1 và T_2 cùng dẫn, đầu ra được nối với V_{CC} nên có mức logic 1.



Hình 3.10.1.1. Cổng PMOS cơ bản

3.11. HỌ CMOS (Complementary – Metal – Oxide – Semiconductor)

Tranzito trong công nghệ MOS là tranzito hiệu ứng trường, gọi là MOSFET (Metal – Oxide – Silicon – Field – Effect Tranzito). Đa số IC số MOS được thiết kế hoàn toàn bằng MOSFET, không cần đến các linh kiện khác. Chữ "C" hàm ý đối xứng bù.

Ưu điểm chính của MOSFET là dễ chế tạo, phí tổn thấp, cỡ nhỏ, tiêu hao ít điện năng. Kỹ thuật chế tạo IC MOS chỉ phức tạp bằng 1/3 kỹ thuật chế tạo IC lưỡng cực. Hơn nữa, thiết bị MOS chiếm ít chỗ trên chip hơn so với tranzito lưỡng cực, diện tích chiếm bằng 1/50 so với tranzito lưỡng cực. Ngoài ra, IC số MOS thường không dùng đến các thành phần điện trở trong IC nên diện tích giảm đi rất nhiều so với tranzito lưỡng cực.

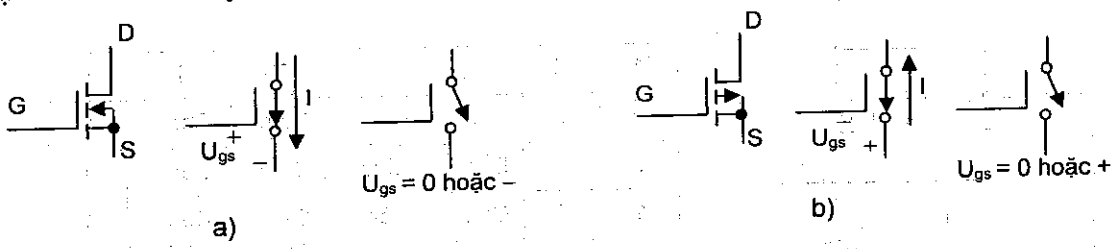
Do mật độ đóng gói cao của IC MOS nên chúng đặc biệt thích hợp cho các IC phức tạp như chip vi xử lý và chip nhớ. Cải tiến trong công nghệ IC MOS đã cho ra đời những thiết bị nhanh hơn 74LS, 74ALS TTL, với đặc điểm điều khiển dòng gần như nhau. Nhờ các ưu điểm của IC MOS mà thiết bị MOS đã hoàn toàn thống trị SSI và MSI. Họ 74ASTTL vẫn nhanh hơn linh kiện MOS tốt nhất, nhưng công suất tiêu hao lại lớn hơn nhiều.

Nhược điểm chủ yếu của thiết bị MOS là dễ bị hư hại bởi tĩnh điện. Tuy nhiên có biện pháp khắc phục, nhưng TTL vẫn bền hơn trong thực hành thí nghiệm.

3.11.1. Công nghệ MOSFET

Hình 3.11.1.1a là ký hiệu của một MOSFET kênh n với các hoạt động của nó. Khi cực cổng của MOSFET dương so với cực nguồn, đường dẫn từ cực máng đến cực nguồn vận hành như chuyển mạch đóng (dẫn điện). Lúc điện thế giữa cực cổng và cực nguồn âm hoặc bằng 0, nó vận hành như chuyển mạch mở (không dẫn điện).

Hình 3.11.1.1b là ký hiệu của một MOSFET kênh p với các hoạt động của nó, hoạt động của MOSFET kênh p đảo với MOSFET kênh n. Khi cực cổng của MOSFET âm so với cực nguồn, đường dẫn từ cực máng đến cực nguồn vận hành như chuyển mạch đóng. Lúc điện thế giữa cực cổng và cực nguồn dương hoặc bằng 0, nó vận hành như chuyển mạch mở.

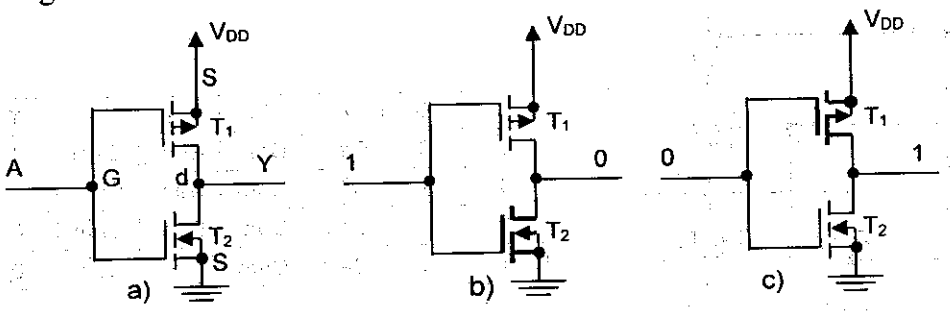


Hình 3.11.1.1.
a) MOSFET kênh n ; b) MOSFET kênh p.

Lưu ý rằng, MOSFET vận hành bằng điện thế, hầu như không có dòng chạy vào cực cổng.

3.11.2. Cổng NOT (CMOS)

Hình 3.11.2.1a là sơ đồ mạch cổng NOT gồm cả MOSFET kênh n và kênh p, biểu thị đối xứng bù.



Hình 3.11.2.1. Mạch NOT (CMOS) và các trạng thái hoạt động

Dòng quy ước chạy từ cực máng đến cực nguồn trong MOSFET kênh n, từ cực nguồn đến cực máng trong MOSFET kênh p, nên ta thấy tranzito T_1 bị đảo trong sơ đồ trên.

– Khi A = 1: T_2 được phân cực thuận, dẫn điện, đầu ra Y nối đất. T_1 không hoạt động nên V_{DD} không được chuyển tải đến đầu ra Y. Do vậy, Y = 0 với đầu vào A = 1 (hình 3.11.2.1b).

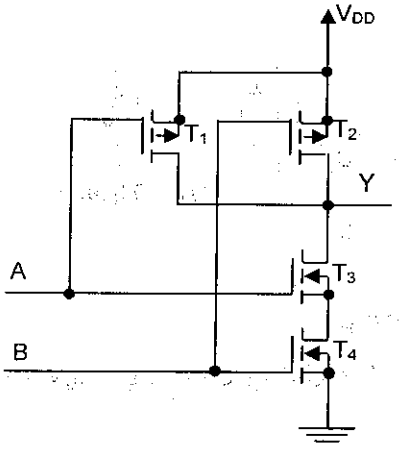
– Khi A = 0: T_2 không hoạt động do $U_{GS} = 0$, Y không nối đất. T_1 được phân cực thuận, dẫn điện, nối đầu ra Y với V_{DD}. Do vậy, Y = 1 với đầu vào A = 0 (hình 3.11.2.1c).

Trong các sơ đồ trên, tranzito được vẽ nét đậm minh họa đang ở trạng thái dẫn.

3.11.3. Cổng NAND CMOS

Hình 3.11.3.1 minh họa cổng NAND (CMOS). Các tranzito kênh n nối tiếp với nhau nên đầu ra Y chỉ nối đất khi A = B = 1.

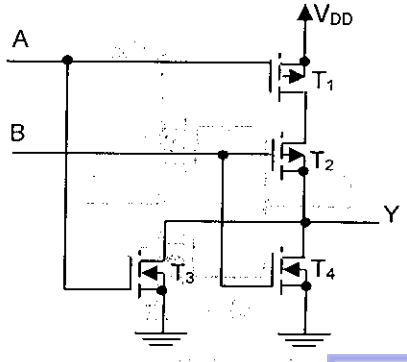
Các tranzito kênh p được nối song song với nhau, nên chỉ cần có ít nhất một đầu vào bằng 0 thì sẽ có ít nhất một tranzito dẫn điện, nối Y với V_{DD}.



A	B	T_1	T_2	T_3	T_4	Y
0	0	Dẫn	Dẫn	Cấm	Cấm	1
0	1	Dẫn	Cấm	Cấm	Dẫn	1
1	0	Cấm	Dẫn	Dẫn	Cấm	1
1	1	Cấm	Cấm	Dẫn	Dẫn	0

Hình 3.11.3.1. Mạch NAND (CMOS) và bảng trạng thái

3.11.4. Cổng NOR (CMOS)



A	B	T_1	T_2	T_3	T_4	Y
0	0	Dẫn	Dẫn	Cấm	Cấm	1
0	1	Dẫn	Cấm	Cấm	Dẫn	0
1	0	Cấm	Dẫn	Dẫn	Cấm	0
1	1	Cấm	Cấm	Dẫn	Dẫn	0

Hình 3.11.4.1. Mạch NOR (CMOS) và bảng trạng thái

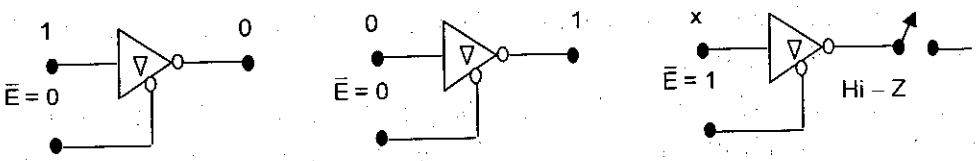
Hình 3.11.4.1 minh họa cổng NOR CMOS. Các tranzito kênh p nối tiếp với nhau nên đầu ra Y chỉ nối với V_{DD} khi A = B = 0.

Các tranzito kênh n được nối song song với nhau, nên chỉ cần có ít nhất một đầu vào bằng 1 thì sẽ có ít nhất một tranzito dẫn điện, nối Y với đất.

3.11.5. Đầu ra CMOS hở cực máng và ba trạng thái

– Đầu ra hở cực máng (OD): Thiết bị CMOS hở cực máng tương đương đầu ra hở cực góp của thiết bị TTL. Trong các thiết bị này, tầng ra chỉ gồm một MOSFET kênh n có cực máng không được nối do MOSFET kênh p ở trên đã bị loại bỏ. Phải mắc một điện trở kéo lên bên ngoài để tạo mức điện thế trạng thái cao. Cũng như các phần tử hở cực góp, đầu ra hở cực máng có thể nối dây theo hàm AND.

– Đầu ra ba trạng thái: Hình 3.11.5.1 minh họa cổng đảo CMOS ba trạng thái và hoạt động của chúng.



Hình 3.11.5.1. Bộ đảo CMOS ba trạng thái

3.11.6. Các đặc điểm của họ CMOS

a) Ưu, nhược điểm của họ CMOS

Ưu điểm:

– Có điện trở lối vào rất lớn vì luôn có 1 nửa số tranzito cấm.

– Vi mạch chỉ tiêu thụ dòng điện khi chuyển mạch (lúc chuyển mạch mới có dòng qua). Vì vậy công suất tiêu thụ cực kỳ nhỏ (cỡ nW).

– CMOS có hệ số mắc tải ở lối ra (FAN OUT) rất lớn (số cổng logic cùng loại có thể mắc vào lối ra của nó là 50 gấp 10 đến 20 lần họ TTL).

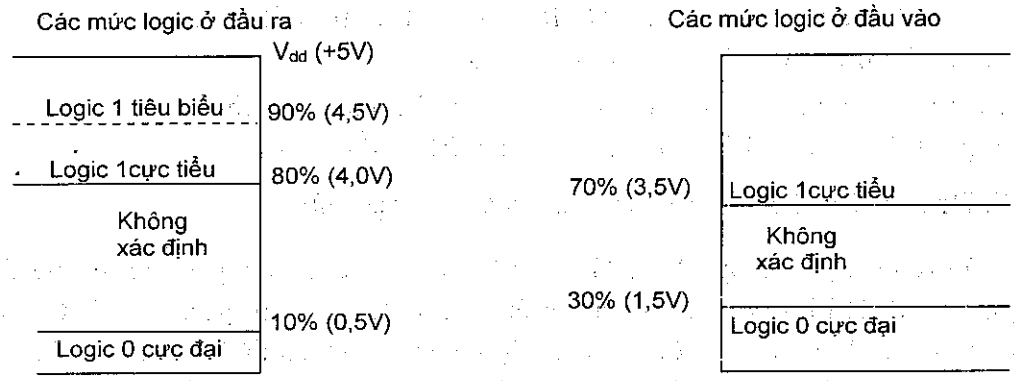
Nhược điểm:

– Tốc độ chuyển mạch thấp nên tần số làm việc không cao và phụ thuộc vào nguồn nuôi. Ví dụ: Khi V_{DD} = 5V thì f = 1 MHz, V_{DD} = 10V thì f = 1,6 MHz, V_{DD} = 15V thì f = 2 MHz.

– Vì điều khiển bằng điện áp nên lối vào dễ bị hỏng khi điện áp đặt vào lớn. Để bảo vệ lối vào người ta mắc thêm các mạch bảo vệ cực cửa khỏi bị quá áp.

b) Điện thế nguồn và điện thế tín hiệu

Chip CMOS vận hành với mức điện thế nguồn bất kỳ giữa +3V và +18V.



Hình 3.11.6.1. Các mức logic CMOS được định nghĩa theo tỷ lệ phần trăm của V_{DD}

Điện thế tín hiệu CMOS được định nghĩa theo tỷ lệ phần trăm điện thế nguồn, hình 3.11.6.1.

c) Đặc điểm kỹ thuật của họ CMOS

CMOS là họ logic công suất thấp, do vậy mà tốc độ hơi chậm, các đặc điểm quan trọng của họ CMOS như sau:

– Công suất: Phần tử CMOS tiêu hao công suất ít hơn họ 74LXX, công suất phụ thuộc vào tần số thay đổi các mức tín hiệu. Công suất chỉ tiêu hao khi tín hiệu thay đổi mức, nếu tín hiệu giữ nguyên mức thì công suất tiêu hao rất nhỏ (bình quân 0,5μW).

– Trễ do truyền: Mối quan hệ giữa công suất và tốc độ cho thấy CMOS chạy khá chậm do mức tiêu hao công suất thấp. Thời gian trễ do truyền tiêu biểu là 40ns đến 90ns, lâu hơn 74LXX. Tần số xung nhịp vào bình quân là 5MHz, nhanh hơn 74LXX.

– Hệ số tải (Fanout): Vì đầu vào đòi hỏi dòng thấp nên CMOS có hệ số fanout cao. Một đầu ra CMOS có thể kích thích từ 50 đầu vào CMOS trở lên, tuy nhiên thời gian trễ sẽ tăng lên theo số đầu vào.

d) Sử dụng phần tử CMOS

Do mức tiêu hao công suất thấp nên có thể đặt nhiều phần tử logic trên một chip mà không sợ quá nhiệt. Vì thế mà CMOS rất được ưa chuộng trong các mạch LSI và VLSI. Phần tử CMOS còn thông dụng trên chip nhớ vì mức tiêu hao công suất rất thấp khi ở trạng thái nhớ.

e) Đầu vào thả nổi

Không thể dự đoán được mức tín hiệu trên các đầu vào CMOS thả nổi, vì thế các đầu vào không dùng đến trên chip CMOS phải được nối với V_{DD} hoặc đất.

g) Giao diện phần tử CMOS với các phần tử TTL chuẩn

Khi nguồn nuôi 5V ta có thể ghép CMOS với TTL, tuy nhiên ở nguồn nuôi thấp như vậy hệ số FAN OUT giảm.

– TTL với CMOS: Một đầu ra TTL có thể kích thích 75 đầu vào CMOS. Trong trường hợp sử dụng cả 75 đầu vào tín hiệu ra có thể bị méo.

– CMOS với TTL: Một đầu ra CMOS có thể kích thích một đầu vào TTL chuẩn.

3.11.7. Các họ CMOS logic

Có hai họ IC – CMOS thường dùng, đó là họ 4000 và họ 54C/74C. Họ CMOS 54C/74C thì tương đương về chân cũng như về chức năng với họ 54/74TTL và vì thế nó trở nên rất thông dụng. Khoảng nhiệt độ hoạt động cho họ 54C là $-55^0C \div +125^0C$ và cho 74C là $-40^0C \div 85^0C$. Nó có khoảng điện áp cung cấp rộng từ 3V đến 18V.

– Xê – ri 4000/14000:

Xê – ri 4000 là xê – ri CMOS đầu tiên được hãng RCA giới thiệu, tương đương chức năng với xê – ri 14000 của hãng Motorola. Xê – ri 4000/14000 có công suất tiêu hao rất thấp và có thể vận hành với nhiều giá trị điện thế nguồn (3 đến 5V), chúng có tốc độ chậm, khả năng dòng ra thấp. Chúng không tương thích về chân cũng như về điện với bất kỳ xê – ri TTL nào.

– Xê – ri 74C:

Xê – ri CMOS này tương thích về chân và tương đương chức năng với thiết bị TTL có cùng số hiệu. Đặc điểm hiệu suất của xê – ri 74C gần giống với xê – ri 4000.

– Xê – ri 74HC/HCT (CMOS tốc độ cao):

Đây là phiên bản cải tiến của xê – ri 74C, có tốc độ chuyển mạch tăng gấp 10 lần so với xê – ri 74LS, khả năng dòng ra cao hơn nhiều so với 74C, IC 74HC/74HCT tương thích chân và tương đương chức năng với IC TTL có cùng số hiệu. 74HCT tương thích điện với TTL còn 74HC thì không. Chúng ta có thể nối trực tiếp IC 74HCT với bất kỳ IC TTL nào. Xê – ri 74HC/74HCT trở thành xê – ri CMOS thông dụng nhất.

– Xê – ri 74AC/ACT (CMOS nâng cao):

Xê – ri 74AC/ACT tương đương chức năng với nhiều xê – ri TTL nhưng không tương thích chân với TTL. Thiết bị 74AC không tương thích điện với TTL, còn 74ACT có thể nối trực tiếp với TTL. Xê – ri 74AC/ACT có ưu thế hơn hẳn so với xê – ri HC ở khả năng kháng nhiễu, trễ do truyền và tốc độ xung nhịp tối đa.

Xê – ri 74AC/ACT dùng số hiệu 5 ký số bắt đầu bằng hai ký số 11.

Ví dụ: 74AC11004 = 74HC04 hay 74ACT11293 = 74HC293.

– Xê – ri 74AHC (CMOS cao tốc nâng cao):

Đây là xê – ri CMOS mới dành cho các ứng dụng nhanh hơn, công suất tiêu hao thấp hơn, dòng kích thích thấp. Thiết bị này nhanh gấp ba lần và có thể thay thế trực tiếp thiết bị thuộc xê – ri HC, chúng có khả năng kháng nhiễu tương tự HC.

– Logic BiCMOS (CMOS lưỡng cực):

Đây là xê – ri logic kết hợp các ưu điểm ưu việt nhất của logic lưỡng cực và logic CMOS gọi là BiCMOS. Đặc điểm công suất tiêu hao thấp của CMOS và tốc độ cao của mạch lưỡng cực được tích hợp chung để tạo ra họ logic công suất tiêu hao cực

thấp, tốc độ cực cao. IC BiCMOS chưa khả dụng ở hầu hết chức năng SSI và MSI, mà bị giới hạn ở những chức năng ứng dụng trong giao diện vi xử lý và bộ nhớ. Xê – ri BiCMOS giảm được 75% công suất tiêu hao so với họ 74F với tốc độ và đặc điểm điều khiển tương đương. Thiết bị BiCMOS tương thích chân với thiết bị TTL chuẩn và hoạt động ở mức logic 5V.

– ***Các tham số đầu vào/ đầu ra với V_{dd} = 5V:***

Thông số	CMOS						
	4000	74HC	74HCT	74AC	74ACT	74AHC	74AHCT
V_{IH} (min) (V)	3,5	3,5	2,0	3,5	2,0	3,85	2,0
V_{IL}(max) (V)	1,5	1,0	0,8	1,5	0,8	1,65	0,8
V_{OH}(min) (V)	4,95	4,9	4,9	4,9	4,9	4,4	3,15
V_{OL}(max) (V)	0,05	0,1	0,1	0,1	0,1	0,44	0,1
V_{NH} (V)	1,45	1,4	2,9	1,4	2,9	0,55	1,15
V_{NL} (V)	1,45	0,9	0,7	1,4	0,7	1,21	0,7
I_{IH}(max) (µA)	1	1	1	1	1	1	1
I_{IL}(max) (µA)	1	1	1	1	1	1	1
I_{OH}(max) (mA)	0,4	4	4	24	24	8	8
I_{OL}(max) (mA)	0,4	4	4	24	24	8	8

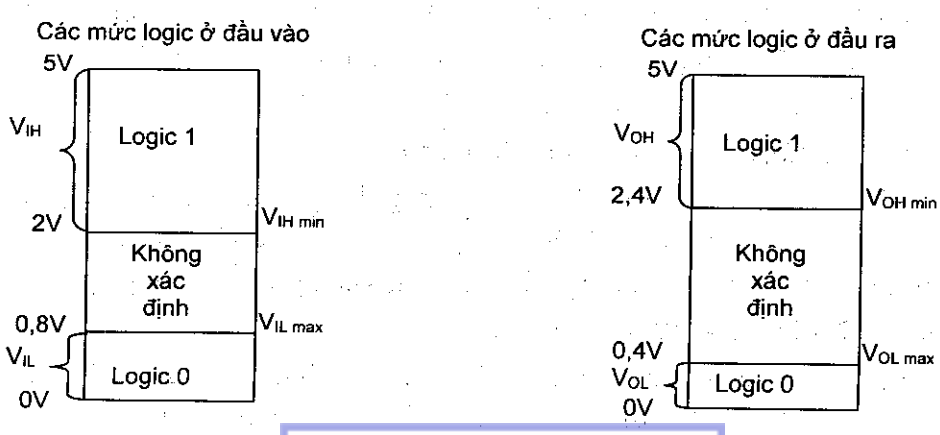
Ví dụ 3.11.7.1. Đầu ra 74AC có thể kích thích được bao nhiêu đầu vào 74LS?

Giải:

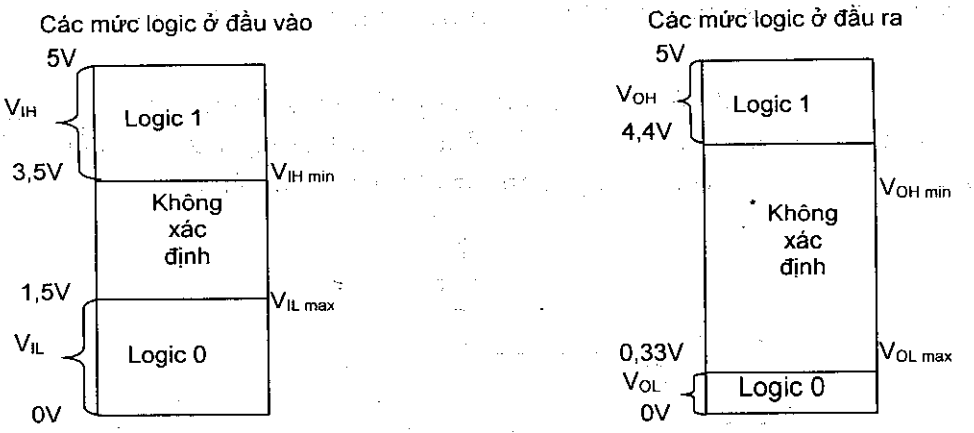
74AC có I_{OL}(max) = 24mA, 74LS có I_{IL}(max) = 0,4mA. Do đó, 74AC có thể kích thích 60 đầu vào 74LS.

BÀI TẬP CHƯƠNG 3

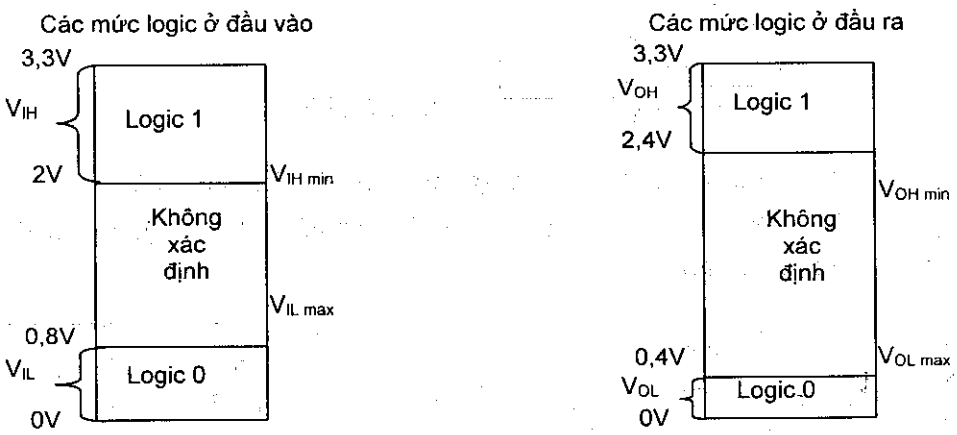
Bài 3.1. Xác định khoảng dự trữ chống nhiễu mức cao và mức thấp cho họ CMOS và TTL với các thông số cho trên hình 3.1BT.



a) Các mức logic TTL



b) Các mức logic CMOS với V_{dd} = +5V

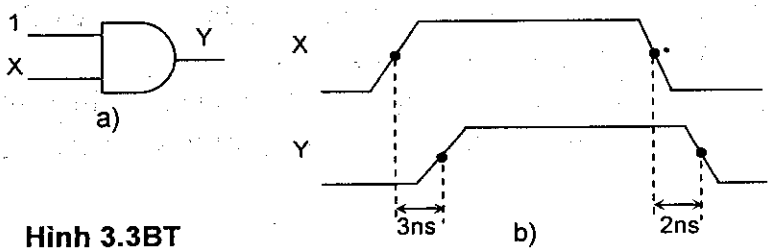


c) Các mức logic CMOS với V_{dd} = +3,3V

Hình 3.1BT.

Bài 3.2. Cho một nhóm cổng logic có I_{CCH} = 2μA và I_{CCL} = 3,6 μA. Hãy tính công suất tiêu hao cho nhóm cổng đó nếu V_{CC} = 5V.

Bài 3.3. Hãy xác định t_{pHL}, t_{pLH} và thời gian trễ của phần tử AND với dạng xung và các thông số cho trên hình 3.3BT.

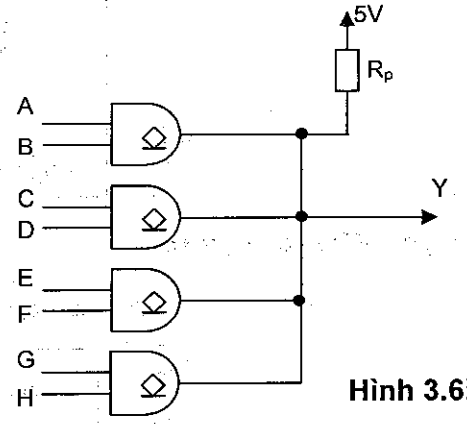


Hình 3.3BT

Bài 3.4. Một đầu ra cổng NAND 74LS00 đang kích thích 3 đầu vào 74SXX và 3 đầu vào 74XX. Kiểm tra xem có vấn đề gì về tải không.

Bài 3.5. Hãy xác định hệ số tải của cổng 74ALS00 dựa vào các tham số dòng điện và điện áp.

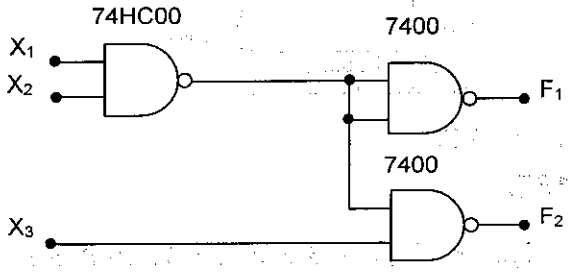
Bài 3.6. Cho bốn cổng AND hở cực góp được nối với nhau tạo thành một cổng AND được nối dây như trên hình 3.6BT. Giả thiết cổng AND (tưởng tượng) ở đầu ra có các thông số như họ TTL chuẩn. Hãy viết phương trình logic cho đầu ra Y.



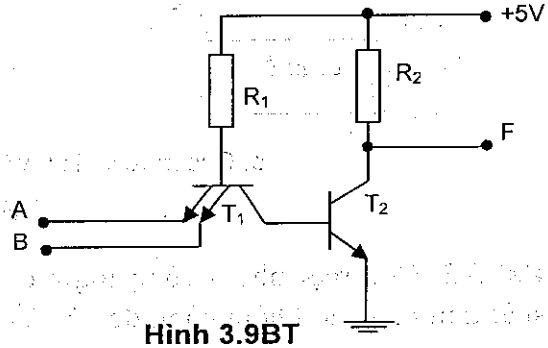
Hình 3.6BT

Bài 3.7. Đầu ra 74AHC có thể kích thích được bao nhiêu đầu vào 74ALS?

Bài 3.8. Hãy giải thích tại sao mạch ở hình 3.8BT sẽ không hoạt động như dự kiến. Có thể sửa chữa bằng cách nào?



Hình 3.8BT



Hình 3.9BT

Bài 3.9. Cho mạch TTL như trên hình 3.9BT.

a) Để phân biệt 2 trạng thái logic rõ ràng thì giữa cực C của T_1 và cực B của T_2 phải thêm linh kiện gì? Hãy giải thích về mặt vật lý?

b) Để nâng cao hệ số tải của mạch người ta phải làm như thế nào?

Bài 3.10. Hãy so sánh ưu nhược điểm của các mạch PMOS, NMOS và CMOS.

Chương 4

CÁC MẠCH LOGIC TỔ HỢP

Căn cứ vào đặc điểm và chức năng logic, ta phân loại mạch số thành hai loại chính:

1. Mạch tổ hợp (Combinational Circuits): Là các mạch có giá trị ổn định của tín hiệu lối ra ở một thời điểm bất kỳ chỉ phụ thuộc vào tổ hợp các giá trị đầu vào tại thời điểm đó, không phụ thuộc vào các đầu vào ở trạng thái trước đó. Đó là các loại mạch: cổng logic cơ bản, các bộ số học, hợp kênh, phân kênh,...

2. Mạch dãy (Sequential Circuits): là các mạch có giá trị tín hiệu lối ra không chỉ phụ thuộc vào các giá trị ở đầu vào ở thời điểm hiện tại mà còn phụ thuộc vào các giá trị đầu vào ở trạng thái trước đó. Mạch dãy là các phần tử nhớ, điển hình là các trigơ.

4.1. PHƯƠNG PHÁP THIẾT KẾ VÀ PHÂN TÍCH CÁC MẠCH LOGIC TỔ HỢP

4.1.1. Phương pháp thiết kế mạch logic tổ hợp

Với một mạch logic tổ hợp bất kỳ nếu cho trước chức năng ta đều có thể thiết kế và thực hiện được. Quá trình thiết kế bao gồm các bước tiến hành như sau:

1. Từ yêu cầu chức năng ta lập bảng chân lý cho hàm
2. Từ bảng trạng thái suy ra phương trình logic
3. Tối giản hoá hàm logic
4. Từ hàm logic tối giản thiết kế mạch thực hiện bằng các phần tử logic.

Đối với mạch tổ hợp có nhiều đầu ra, khi thiết kế nên sử dụng những phần chung cho các hàm đó để sơ đồ là tối ưu nhất.

Tuy nhiên, những bước thiết kế trên đây không phải là bắt buộc áp dụng máy móc, mà nên được vận dụng linh hoạt theo tình huống cụ thể của thiết kế thực tế.

a) Thiết kế một hàm logic tổ hợp

Quá trình thiết kế được thực hiện theo các bước đã trình bày ở trên, tùy theo từng yêu cầu cụ thể mà ta thực hiện cho hợp lý, sao cho sơ đồ nhận được là tối ưu nhất.

Trong mạch logic, các tín hiệu truyền từ đầu vào tới đầu ra theo các đường khác nhau. Đường đi dài nhất trong một mạch gọi là đường truyền của mạch, số lượng phần tử logic nằm trên đường truyền của mạch gọi là số tầng của mạch.

Như đã xét trong chương 3, ta biết rằng mỗi phần tử logic chỉ có khả năng nối với một số phần tử khác ở đầu vào và đầu ra. Khả năng đó gọi là hệ số hợp lối vào và hệ số tải đầu ra.

Như vậy, nếu mạch càng ít tầng thì tốc độ làm việc nhanh (tổng thời gian trễ của các tín hiệu khi đi qua các phần tử càng nhỏ) và ít có tranh chấp trạng thái (do trễ của tín hiệu gây ra), phân tích và thiết kế mạch đơn giản. Nhưng với mạch ít tầng thì có

thể gặp vấn đề là hệ số hợp lối vào và hệ số tải đầu ra không đáp ứng được, khi đó phải tăng số tầng của mạch lên. Số tầng của mạch càng nhiều thì tín hiệu đi từ đầu vào tới đầu ra sẽ mất nhiều thời gian hơn dẫn đến tốc độ làm việc chậm hơn. Ngoài ra với mạch nhiều tầng do các tín hiệu đi theo các đường dài ngắn khác nhau nên xuất hiện ở các đầu ra vào các thời điểm khác nhau. Điều đó có thể làm sai lệch chức năng của mạch.

Do đó, khi thiết kế mạch phải đảm bảo được các yếu tố về hệ số tải và số tầng của mạch.

Mạch hai tầng:

Ví dụ 4.1.1.1. Thiết kế mạch logic cho bởi hàm $F(A,B,C) = \sum(0,1,2,5)$ theo các yêu cầu sau:

a) Tầng 1 dùng AND, tầng 2 dùng OR

b) Tầng 1 dùng OR, tầng 2 dùng AND

c) Tầng 1 dùng NAND, tầng 2 dùng AND

d) Tầng 1 dùng NOR, tầng 2 dùng OR

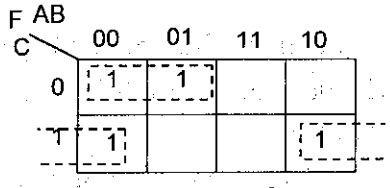
e) Tầng 1 dùng OR, tầng 2 dùng NAND

f) Tầng 1 dùng NAND, tầng 2 dùng NAND

g) Tầng 1 dùng AND, tầng 2 dùng NOR

h) Tầng 1 dùng NOR, tầng 2 dùng NOR

Giải: Tối thiểu hóa hàm, hình 4.1.1.1:



Hình 4.1.1.1. Tối thiểu hoá hàm

Ta được: $F = \bar{A}\bar{C} + \bar{B}C$ (dạng tổng các tích)

$F = (\bar{A} + C)(\bar{B} + \bar{C})$ (dạng tích các tổng)

a) Tầng 1 dùng AND, tầng 2 dùng OR: Viết phương trình dạng tổng các tích.

Vẽ sơ đồ theo phương trình: $F = \bar{A}\bar{C} + \bar{B}C$.

b) Tầng 1 dùng OR, tầng 2 dùng AND: Viết phương trình dạng tích các tổng.

Vẽ sơ đồ theo phương trình: $F = (\bar{A} + C)(\bar{B} + \bar{C})$

c) Tầng 1 dùng NAND, tầng 2 dùng AND:

Viết phương trình của hàm dạng tích các tổng, đảo hai lần từng số hạng và áp dụng định lý De Morgan cho từng số hạng đó.

Vẽ sơ đồ theo phương trình: $F = \overline{\overline{(\bar{A} + C)}}\,\overline{\overline{(\bar{B} + \bar{C})}} = \overline{(A\bar{C})}\,\overline{(BC)}$

d) Tầng 1 dùng NOR, tầng 2 dùng OR:

Viết phương trình của hàm dạng tổng các tích, đảo hai lần từng số hạng và áp dụng định lý De Morgan cho từng số hạng đó.

Vẽ sơ đồ theo phương trình: $F = \overline{\overline{A}\overline{C}} + \overline{\overline{B}\overline{C}} = \overline{A + C} + \overline{B + \overline{C}}$

e) Tầng 1 dùng OR, tầng 2 dùng NAND

Viết phương trình của hàm dạng tổng các tích, đảo hai lần và áp dụng định lý De Morgan.

Vẽ sơ đồ theo phương trình: $F = \overline{\overline{A}\overline{C} + \overline{B}\overline{C}} = \overline{\overline{\overline{A}\overline{C}}\,\overline{\overline{B}\overline{C}}} = \overline{(A + C)(B + \overline{C})}$

f) Tầng 1 dùng NAND, tầng 2 dùng NAND

Viết phương trình của hàm dạng tổng các tích, đảo hai lần và áp dụng định lý De Morgan.

Vẽ sơ đồ theo phương trình: $F = \overline{\overline{A}\overline{C} + \overline{B}\overline{C}} = \overline{\overline{A}\overline{C}\,\overline{B}\overline{C}}$

g) Tầng 1 dùng AND, tầng 2 dùng NOR

Viết phương trình của hàm dạng tích các tổng, đảo hai lần và áp dụng định lý De Morgan.

Vẽ sơ đồ theo phương trình: $F = \overline{(\overline{A} + C)(\overline{B} + \overline{C})} = \overline{(\overline{A} + C) + (\overline{B} + \overline{C})} = \overline{A\overline{C} + BC}$

h) Tầng 1 dùng NOR, tầng 2 dùng NOR

Viết phương trình của hàm dạng tích các tổng, đảo hai lần và áp dụng định lý De Morgan.

Vẽ sơ đồ theo phương trình: $F = \overline{(\overline{A} + C)(\overline{B} + \overline{C})} = \overline{(\overline{A} + C) + (\overline{B} + \overline{C})}$

Mạch nhiều tầng:

Ta thấy rằng, phương trình tối giản thường chứa nhiều thành phần hoặc nhiều biến trong một thành phần. Trong khi đó các IC thường có số đầu vào ít, nên phải tăng số tầng của mạch để giảm số đầu vào cho các tầng. Ngoài ra, khả năng tải của các phần tử thường bị hạn chế, khi khả năng tải không đáp ứng được phải thay đổi cấu trúc cũng dẫn đến số tầng tăng lên.

Ví dụ 4.1.1.2. Cho hàm $F = ABC + CD + \overline{A}\,\overline{B}D + \overline{A}\,\overline{B}C$

Thực hiện hàm bằng mạch AND, OR hai đầu vào.

Giải: Thực hiện nhóm các số hạng, khi yêu cầu đặt ra là dùng phần tử hai đầu vào, ta sẽ thực hiện nhóm các số hạng như sau là tối ưu nhất:

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

Như vậy, ta sẽ cần đến 3 phần tử OR, 4 phần tử AND và mạch được nối theo 4 cấp.

Đối với hệ hàm chúng ta sẽ xét trong phần sau.

Ví dụ 4.1.1.3. Thiết kế mạch so sánh 2 số nhị phân A và B có chức năng như sau:

Nếu A = B thì lối ra G = 1; A ≠ B thì lối ra G = 0 (mạch so sánh bằng nhau).

a) A và B là 2 số nhị phân 1 bit (mạch so sánh bằng nhau 1 bit)

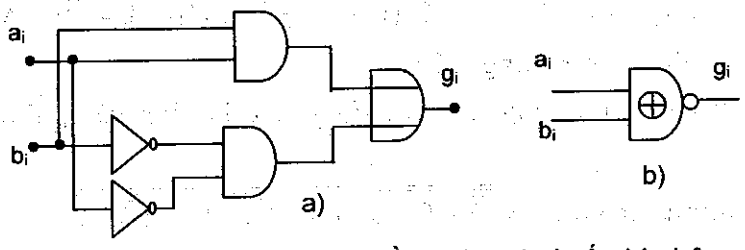
– Bảng trạng thái trên hình 4.1.1.2.

Đầu vào		Đầu ra
a_i	b_i	g_i
0	0	1
0	1	0
1	0	0
1	1	1

Hình 4.1.1.2. Bảng trạng thái mạch so sánh bằng nhau

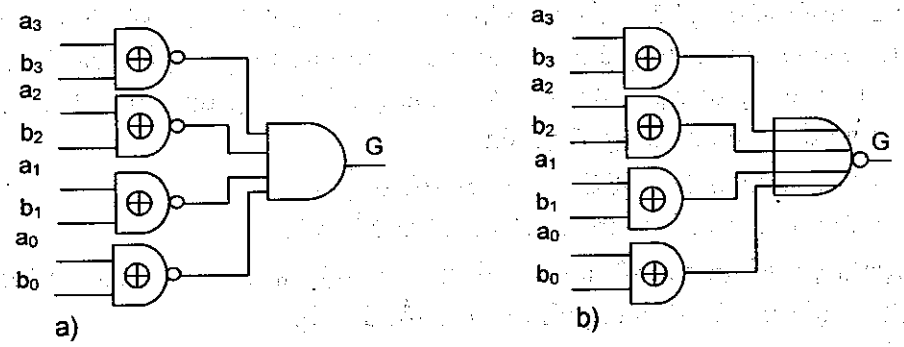
– Phương trình logic: $g_i = \overline{a_i}\ \overline{b_i} + a_i b_i = \overline{a_i \oplus b_i}$ (4.1.1.1)

– Phương trình (4.1.1.1) đã được viết ở dạng tối giản rồi ta không cần phải rút gọn nữa. Từ phương trình logic ta có thể dùng nhiều sơ đồ để thực hiện hàm logic trên, tuỳ thuộc vào các phần tử logic cho trước mà ta phải biến đổi phương trình cho phù hợp. Sơ đồ logic của bộ so sánh bằng nhau hai số nhị phân 1 bit dùng các cổng logic cơ bản được trình bày trên hình 4.1.1.3a và dùng cổng XNOR được cho trên hình 4.1.1.3b.



Hình 4.1.1.3. Mạch so sánh bằng nhau hai số nhị phân một bit

b) A và B là hai số nhị phân 4 bit (bộ so sánh bằng nhau 4 bit)



Hình 4.1.1.4. Mạch so sánh bằng nhau hai số nhị phân 4 bit

So sánh hai số nhị phân 4 bit A = $a_3\ a_2\ a_1\ a_0$ và B = $b_3\ b_2\ b_1\ b_0$, ta thấy rằng chỉ khi $a_3 = b_3$, $a_2 = b_2$, $a_1 = b_1$, $a_0 = b_0$ thì A = B (G = 1), còn trong các trường hợp khác thì A ≠ B (G = 0). Vậy nếu g_i (i = 0 ÷ 3) là lối ra của các bộ so sánh 1 bit thì không cần lập bảng chân lý ta có thể suy ra phương trình logic của bộ so sánh 4 bit: G = $g_3\ g_2\ g_1\ g_0$

Với $g_3 = \overline{a_3 \oplus b_3}$; $g_2 = \overline{a_2 \oplus b_2}$; $g_1 = \overline{a_1 \oplus b_1}$; $g_0 = \overline{a_0 \oplus b_0}$

Như vậy: $G = \overline{a_3 \oplus b_3}\ \ \overline{a_2 \oplus b_2}\ \ \overline{a_1 \oplus b_1}\ \ \overline{a_0 \oplus b_0}$

$$G = \overline{(a_3 \oplus b_3)+(a_2 \oplus b_2)+(a_1 \oplus b_1)+(a_0 \oplus b_0)}$$

– Sơ đồ logic dùng cổng XNOR và AND cho ở hình 4.1.1.4a và dùng cổng XOR và NOR cho ở hình 4.1.1.4b.

b) Thiết kế một hệ hàm logic tổ hợp

Trong trường hợp mạch có nhiều đầu ra thì ta phải đơn giản hệ hàm sao cho có thể dùng chung được càng nhiều phần tử cho các hàm càng tốt. Khi đó, cho phép tiết kiệm linh kiện, tăng tốc độ của mạch, giảm được các hư hỏng, dễ xử lý lỗi,...

Có nhiều phương pháp đơn giản hệ hàm, ở đây chúng ta sẽ xét phương pháp đơn giản và thông dụng đó là tách các implicant đơn giản chung cho cả hệ.

Tách phần chung cho các hàm của hệ tạo thành hàm f_0, phần còn lại của mỗi hàm được đơn giản bằng cách coi các tổ hợp chung là các tổ hợp không xác định. Phương trình tối giản của các hàm chính là f_0 cộng với f_i'. Tùy thuộc vào loại phần tử sử dụng mà ta biến đổi các hàm đó cho phù hợp.

Với cách thức trên chúng ta có thể áp dụng cho phương pháp tối thiểu hóa MC – Quine Cluskey hoặc bảng Karnaugh. Ở đây chúng ta sẽ xét ví dụ đối với phương pháp dùng bảng Karnaugh.

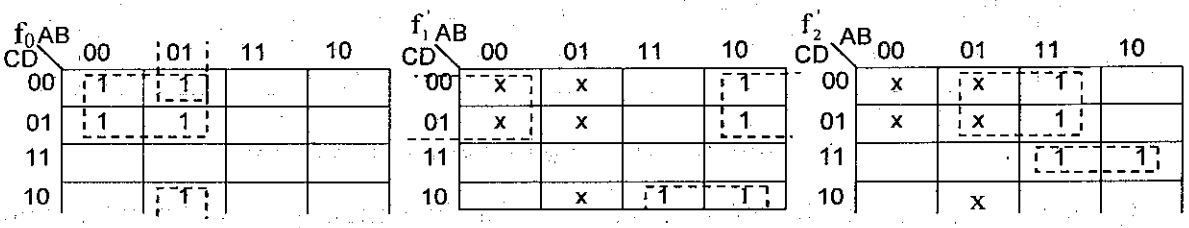
Ví dụ 4.1.1.4. Cho hệ hàm logic sau:

$$f_1(A,B,C,D) = \sum(0,1,4,5,6,8,9,10,14)$$

$$f_2(A,B,C,D) = \sum(0,1,4,5,6,11,12,13,15)$$

Vẽ sơ đồ logic thực hiện hệ hàm trên?

Giải: Xét các bảng Karnaugh trên hình 4.1.1.5.



Hình 4.1.1.5. Tối thiểu hoá các hàm

Từ bảng Karnaugh ta có:

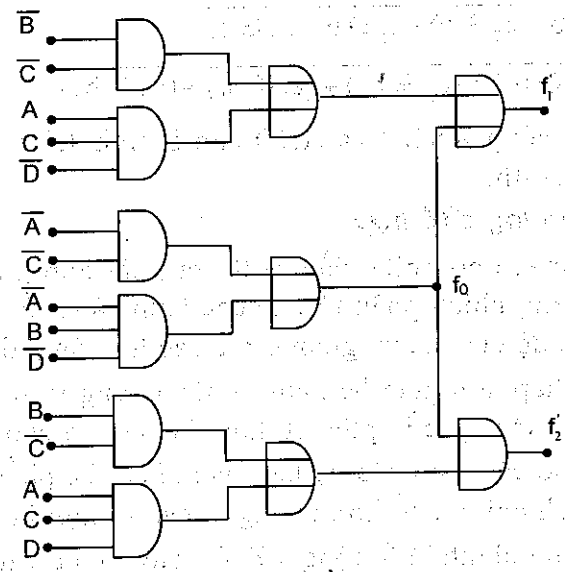
$$f_0 = \overline{A}\overline{C} + \overline{A}B\overline{D}\ ; \quad f_1' = \overline{B}\overline{C} + AC\overline{D};$$

$$f_2' = B\overline{C} + ACD$$

$$f_1 = f_0 + f_1' = \overline{A}\overline{C} + \overline{A}B\overline{D} + \overline{B}\overline{C} + AC\overline{D};$$

$$f_2 = f_0 + f_2' = \overline{A}\overline{C} + \overline{A}B\overline{D} + B\overline{C} + ACD$$

Sơ đồ logic hình 4.1.1.6.



Hình 4.1.1.6. Sơ đồ logic

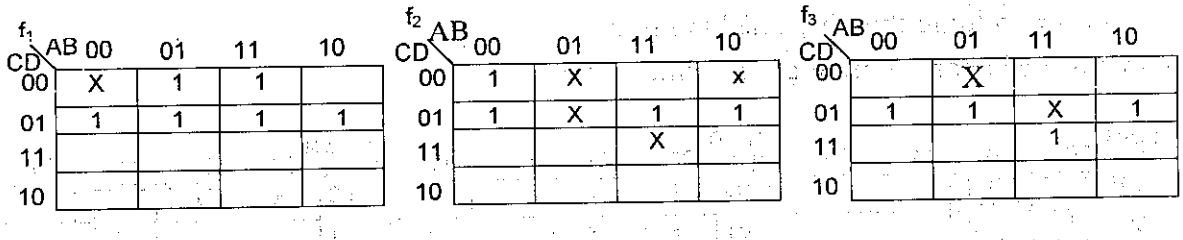
Ví dụ 4.1.1.5. Thực hiện hệ hàm sau với các phần tử hai đầu vào, hệ số tải ≤ 2.

$f_1(A,B,C,D) = \sum(1,4,5,9,12,13)$ với N = 0

$f_2(A,B,C,D) = \sum(0,1,9,13)$ với N = 4,5,8,15

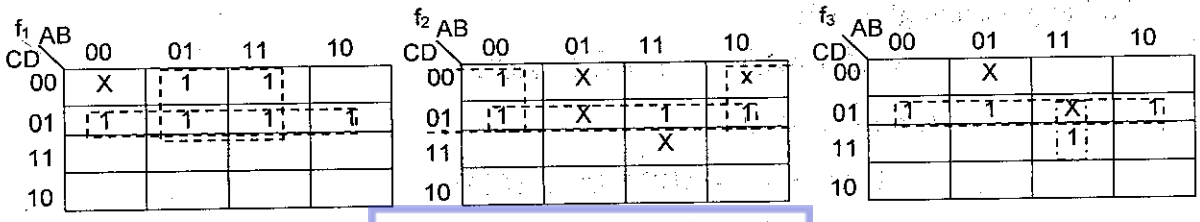
$f_3(A,B,C,D) = \sum(1,5,9,15)$ với N = 4,13

Giải: Biểu diễn hàm trên bảng Karnaugh như hình 4.1.1.7:



Hình 4.1.1.7. Bảng Karnaugh của các hàm

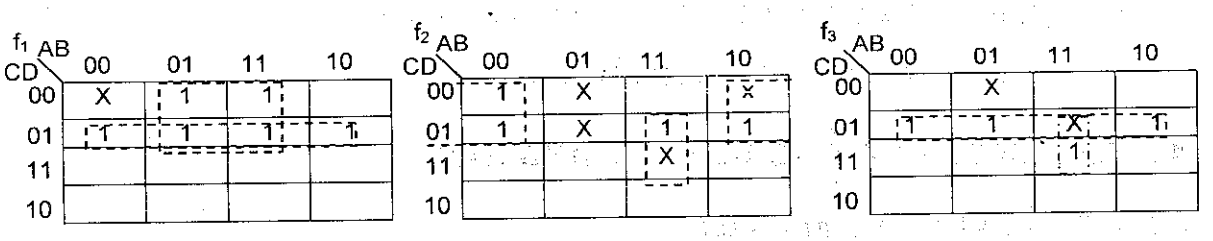
Quan sát các bảng trên hình 4.1.1.7 ta thấy, nếu chọn phương án tối thiểu hóa như hình 4.1.1.8 thì sẽ gây ra quá tải ở đầu ra $\bar{C}D$.



Hình 4.1.1.8. Tối thiểu hoá các hàm

Khi đó: $f_1 = B\overline{C} + \overline{C}D$; $f_2 = \overline{B}\overline{C} + \overline{C}D$; $f_3 = ABD + \overline{C}D$

Cả ba hàm đều có chung số hạng $\overline{C}D$ nên đầu ra phần tử AND này sẽ được đưa vào ba phần tử OR. Nhưng hệ số tải yêu cầu ≤ 2, vì vậy ta chọn phương án khác như hình 4.1.1.9.



Hình 4.1.1.9. Phương án tối ưu

Khi đó: $f_1 = B\overline{C} + \overline{C}D$; $f_2 = \overline{B}\overline{C} + ABD$; $f_3 = ABD + \overline{C}D$.

Như vậy, ta thấy số phần tử không tăng lên mà vẫn đảm bảo hệ số tải yêu cầu.

Nhận xét: Tùy thuộc vào từng yêu cầu cụ thể đối với các hàm mà ta biến đổi sao cho phù hợp, sao cho kết quả đạt được là tối ưu nhất chứ không có một công thức cụ thể đặt ra cho tất cả các hàm. Để làm được điều đó chúng ta phải vận dụng linh hoạt các định luật của đại số logic và các phương pháp tối thiểu hóa hàm logic.

4.1.2. Phân tích mạch logic tổ hợp

Từ sơ đồ logic cho trước, tìm hàm logic các đầu ra theo các đầu vào, quá trình thực hiện như sau:

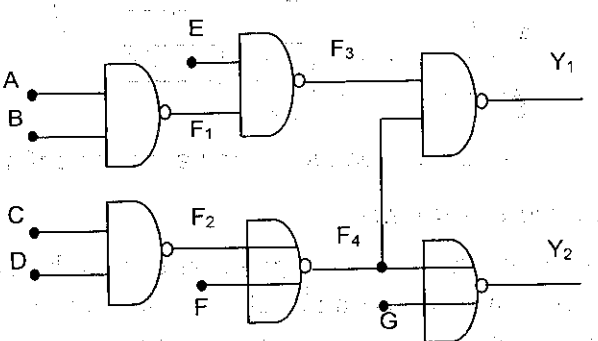
– Đặt các biến phụ vào mỗi đầu ra của mỗi phần tử logic.

– Viết phương trình của các biến phụ đó lần lượt từ đầu vào cho đến đầu ra.

– Ở phương trình cuối cùng, thay thế các biến tương ứng để rút ra được hàm logic cần tìm.

– Từ hàm logic nhận được có thể chuyển về bảng trạng thái và từ đó có thể xây dựng lại hệ thống bằng các phần tử khác.

Ví dụ 4.1.2.1. Phân tích mạch logic tổ hợp cho trên hình 4.1.2.1.



Hình 4.1.2.1. Mạch logic tổ hợp

Giải: Ta đặt các biến phụ tại các đầu ra của các phần tử logic như trên hình 4.1.2.1.

Ta có: $F_1 = \overline{AB}$; $F_2 = \overline{CD}$; $F_3 = \overline{EF_1}$; $F_4 = \overline{F_2 + F}$; $Y_1 = \overline{F_3 F_4}$; $Y_2 = \overline{F_4 + G}$

Như vậy: $F_3 = \overline{E\overline{AB}}$; $F_4 = \overline{\overline{CD} + F}$.

$$Y_1 = \overline{F_3 F_4} = \overline{F_3} + \overline{F_4} = E\overline{AB} + \overline{CD} + F = E(\overline{A} + \overline{B}) + \overline{C} + \overline{D} + F$$

$$Y_2 = \overline{F_4 + G} = \overline{F_4}.\overline{G} = (\overline{CD} + F)\overline{G} = (\overline{C} + \overline{D} + F)\overline{G}$$

4.2. CÁC MẠCH LOGIC TỔ HỢP THƯỜNG GẶP

4.2.1. Bộ so sánh (Comparator)

Trong nhiều trường hợp phải so sánh 2 số nhị phân A và B để chỉ ra được mối quan hệ giữa chúng: A > B, A < B hay A = B.

a) Bộ so sánh hai số nhị phân 1 bit

Có hai số nhị phân 1 bit a_i và b_i, từ yêu cầu đặt ra ta lập được bảng trạng thái như trên hình 4.2.1.1.

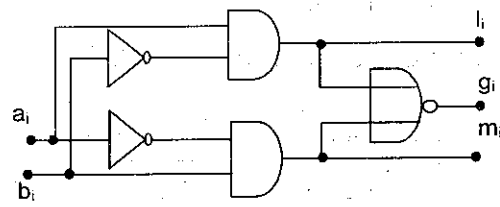
Đầu vào		Đầu ra		
a_i	b_i	l_i ($a_i > b_i$)	g_i ($a_i = b_i$)	m_i ($a_i < b_i$)
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

Hình 4.2.1.1. Bảng trạng thái của mạch so sánh hai số nhị phân 1 bit

– Phương trình logic:

$$l_i = a_i \overline{b_i}; \qquad g_i = \overline{a_i}\ \overline{b_i} + a_i b_i; \qquad m_i = \overline{a_i}\ b_i$$

– Sơ đồ logic được cho ở hình 4.2.1.2.



Hình 4.2.1.2. Mạch so sánh hai số nhị phân

b) Bộ so sánh hai số nhị phân 4 bit

Cũng giống như so sánh trong hệ thập phân, khi so sánh hai số nhị phân nhiều bit ta phải bắt đầu từ bit có trọng số cao nhất, chỉ khi nào bit có trọng số cao nhất bằng nhau thì mới tiếp tục so sánh đến bit có trọng số thấp hơn liền kề. Ý nghĩa trọng số khiến việc so sánh quyết định bởi số có trọng số lớn.

Giả sử có hai số nhị phân 4 bit: $A = a_3\ a_2\ a_1\ a_0$ và $B = b_3\ b_2\ b_1\ b_0$. Để xây dựng được sơ đồ mạch so sánh này, cần 4 mạch so sánh một bit và các mạch logic phụ trợ.

Đầu tiên ta thực hiện so sánh 2 bit có trọng số lớn nhất a_3 và b_3:

Nếu $a_3 > b_3$ thì $A > B$

Nếu $a_3 < b_3$ thì $A < B$

Nếu $a_3 = b_3$ thì so sánh tiếp a_2 với b_2

.....

Nếu $a_0 > b_0$ thì $A > B$

Nếu $a_0 < b_0$ thì $A < B$

Nếu $a_0 = b_0$ thì $A = B$

Quá trình trên có thể tóm tắt như sau:

$$A > B \Leftrightarrow (a_3 > b_3) + (a_3 = b_3)(a_2 > b_2) + (a_3 = b_3)(a_2 = b_2)(a_1 > b_1) + \\ + (a_3 = b_3)(a_2 = b_2)(a_1 = b_1)(a_0 > b_0)$$

$$A < B \Leftrightarrow (a_3 < b_3) + (a_3 = b_3)(a_2 < b_2) + (a_3 = b_3)(a_2 = b_2)(a_1 < b_1) + \\ +(a_3 = b_3)(a_2 = b_2)(a_1 = b_1)(a_0 < b_0)$$

$$A = B \Leftrightarrow (a_3 = b_3)(a_2 = b_2)(a_1 = b_1)(a_0 = b_0)$$

– Từ đó ta có phương trình logic:

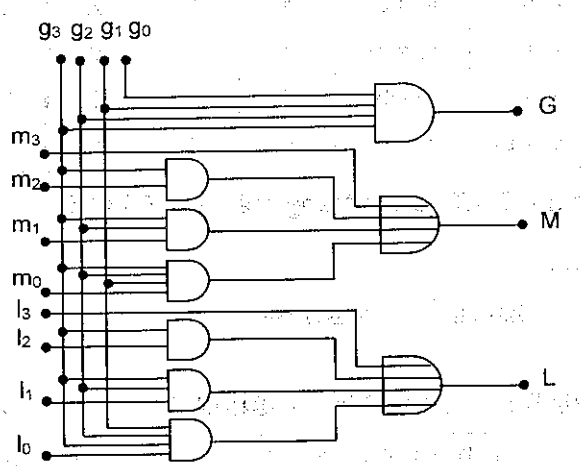
$$L(A > B) = l_3 + g_3l_2 + g_3g_2l_1 + g_3g_2g_1l_0$$

$$G(A = B) = g_3g_2g_1g_0$$

$$M(A < B) = m_3 + g_3m_2 + g_3g_2m_1 + g_3g_2g_1m_0$$

Trong đó, L, G, M là các lối ra của bộ so sánh 4 bit và l_i, g_i, m_i với $i = 0 \div 3$ là các lối ra của các bộ so sánh 1 bit.

– Sơ đồ logic: Dùng AND, OR với l_i, g_i, m_i ($i = 0 \div 3$) được đưa đến từ các bộ so sánh 1 bit được cho trên hình 4.2.1.3.



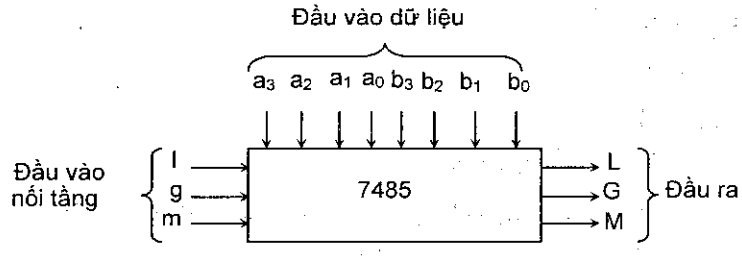
Hình 4.2.1.3. Mạch so sánh hai số nhị phân 4 bit

c) Bộ so sánh hai số nhị phân n bit

Để so sánh hai số nhị phân n bit người ta thực hiện nối tầng các bộ so sánh 4 bit. Bộ so sánh 4 bit ở trên có thêm các đầu vào nối tầng được tích hợp dưới dạng IC có tên là IC 7485 (IC 74HC85, IC 74LS85).

Ký hiệu của IC 7485 được cho trên hình 4.2.1.4.

Khi nối tầng 2 bộ so sánh, đầu ra của bộ so sánh bit thấp (có trọng số nhỏ hơn) được nối đến đầu vào nối tầng tương ứng của bộ so sánh bit cao.



Hình 4.2.1.4. Ký hiệu của IC 7485

– Bảng trạng thái của IC7485 cho trên hình 4.2.1.5.

Đầu vào so sánh				Đầu vào nối tầng			Đầu ra		
a_3b_3	a_2b_2	a_1b_1	a_0b_0	I	m	g	L	M	G
$a_3 > b_3$	X	X	X	X	X	X	1	0	0
$a_3 < b_3$	X	X	X	X	X	X	0	1	0
$a_3 = b_3$	$a_2 > b_2$	X	X	X	X	X	1	0	0
$a_3 = b_3$	$a_2 < b_2$	X	X	X	X	X	0	1	0
$a_3 = b_3$	$a_2 = b_2$	$a_1 > b_1$	X	X	X	X	1	0	0
$a_3 = b_3$	$a_2 = b_2$	$a_1 < b_1$	X	X	X	X	0	1	0
$a_3 = b_3$	$a_2 = b_2$	$a_1 = b_1$	$a_0 > b_0$	X	X	X	1	0	0
$a_3 = b_3$	$a_2 = b_2$	$a_1 = b_1$	$a_0 < b_0$	X	X	X	0	1	0
$a_3 = b_3$	$a_2 = b_2$	$a_1 = b_1$	$a_0 = b_0$	1	0	0	1	0	0
$a_3 = b_3$	$a_2 = b_2$	$a_1 = b_1$	$a_0 = b_0$	0	1	0	0	1	0
$a_3 = b_3$	$a_2 = b_2$	$a_1 = b_1$	$a_0 = b_0$	0	0	1	0	0	1

Hình 4.2.1.5. Bảng trạng thái của IC7485

– Phương trình logic:

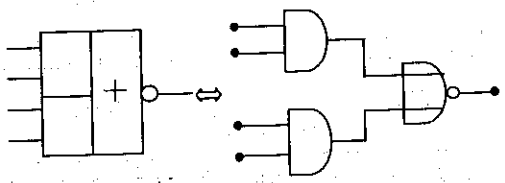
$L = l_3 + g_3l_2 + g_3g_2l_1 + g_3g_2g_1l_0 + g_3g_2g_1g_0l$

$G = g_3g_2g_1g_0g$

$M = m_3 + g_3m_2 + g_3g_2m_1 + g_3g_2g_1m_0 + g_3g_2g_1g_0m$

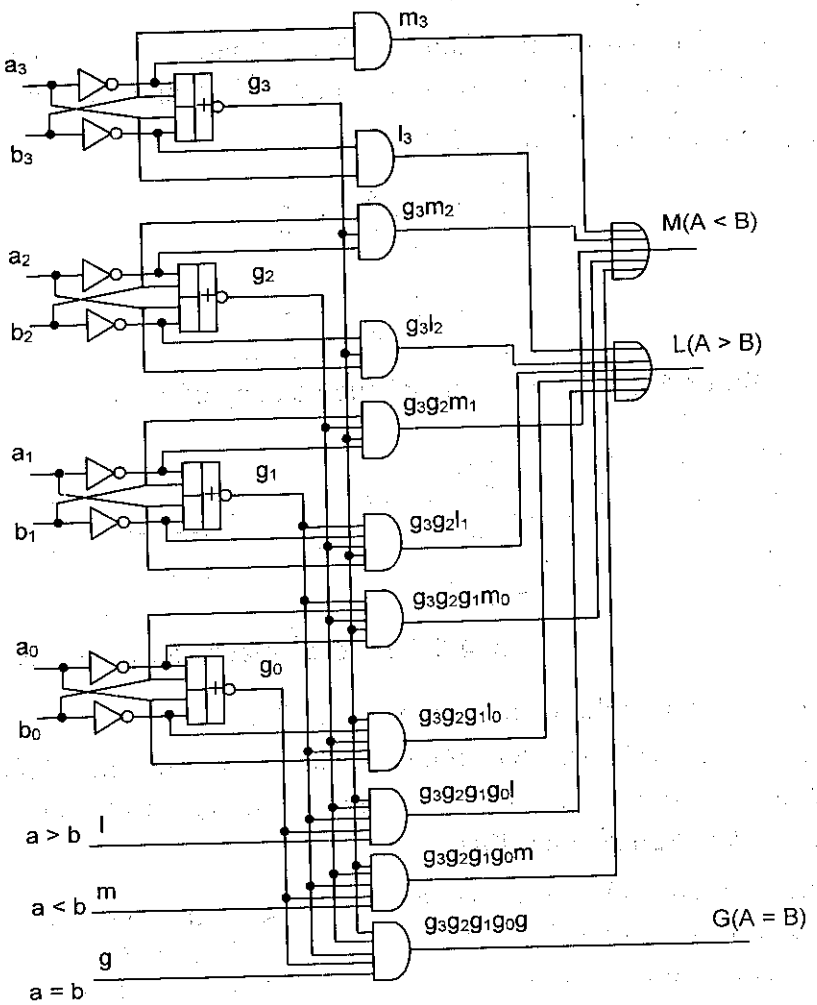
Trong đó: $l_i = a_i\overline{b_i}$; $g_i = \overline{a_i}\,\overline{b_i} + a_ib_i$; $m_i = \overline{a_i}b_i$; với $(i = 0 \div 3)$ là các đầu ra của 4 bộ so sánh 1 bit.

– Ký hiệu NORAND trên hình 4.2.1.6:



Hình 4.2.1.6. Ký hiệu NORAND

– Sơ đồ logic của IC 7485 được cho trên hình 4.2.1.7.

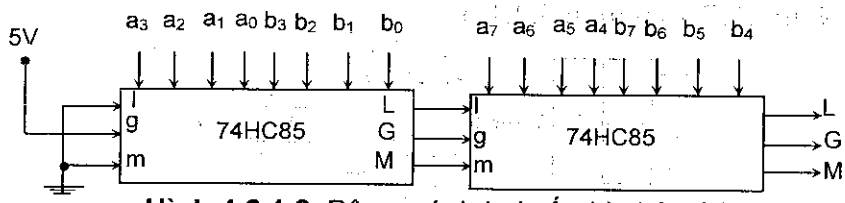


Hình 4.2.1.7. Sơ đồ logic IC 7485

Khi so sánh 4 bit thì các đầu vào nối tầng I = m = 0, g = 1.

Khi so sánh hơn 5 bit ta thực hiện nối tầng từ 2 bộ so sánh 4 bit trở lên. Đầu vào nối tầng có nhãn trùng với đầu ra.

Ví dụ 4.2.1.1. Ghép nối hai IC 7485 tạo thành bộ so sánh 2 số nhị phân 8 bit như hình 4.2.1.8.



Hình 4.2.1.8. Bộ so sánh hai số nhị phân 8 bit

4.2.2. Bộ cộng hai số nhị phân – ALU(Adder Logic Unit)

a) Bộ tổng bán phần (Half Adder – HA)

Thực hiện phép cộng hai bit nhị phân, mạch có 2 đầu vào a_i và b_i là các số hạng được cộng, 2 đầu ra là S (tổng) và C_i (số nhớ sang bit có trọng số cao hơn).

– Bảng trạng thái cho trên hình 4.2.2.1.

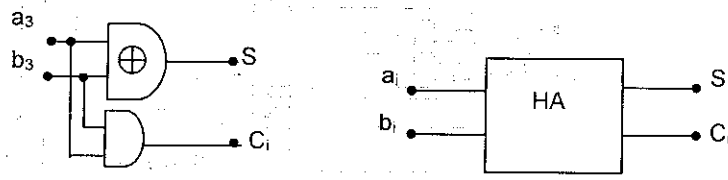
a_i	b_i	S	C_i
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Hình 4.2.2.1. Bảng trạng thái của bộ cộng bán phần

– Phương trình logic:

$$S = a_i \oplus b_i$$

$$C_i = a_i b_i$$



Hình 4.2.2.2. Sơ đồ mạch và ký hiệu của HA

– Sơ đồ mạch và ký hiệu được cho trên hình 4.2.2.2.

Ta gọi bộ cộng bán phần (bộ bán tổng) vì riêng nó chưa thực hiện được phép cộng. Ta phải dùng hai bộ bán tổng mới tạo ra được một mạch tính cộng.

b) Bộ tổng toàn phần (Full Adder: FA)

Bộ tổng toàn phần có 3 lối vào A, B và C_i (Carry In), hai lối ra là tổng S và lối ra nhớ chuyển sang hàng sau C_o (Carry Out).

– Bảng trạng thái cho trên hình 4.2.2.3.

– Hàm logic của FA:

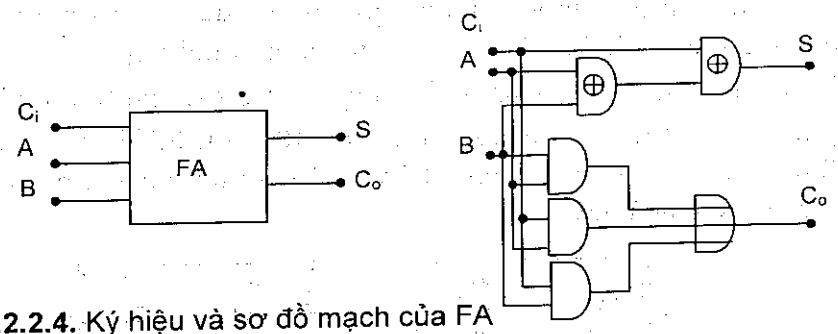
$$S = A \oplus B \oplus C_i$$

$$C_o = (A \oplus B)C_i + AB = AB + AC_i + BC_i$$

– Sơ đồ khối và mạch logic cho trên hình 4.2.2.4.

A	B	C_i	S	C_o
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

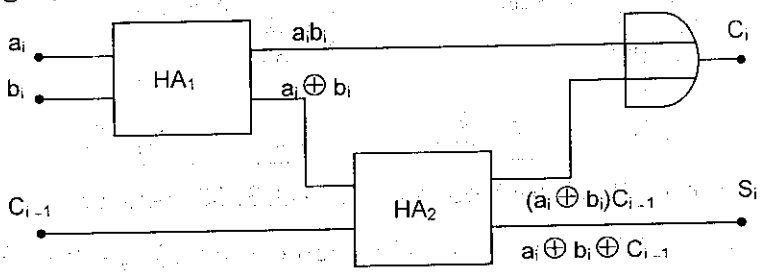
Hình 4.2.2.3. Bảng trạng thái của FA



Hình 4.2.2.4. Ký hiệu và sơ đồ mạch của FA

Nếu $C_i = 0$ thì FA trở thành HA.

– Xây dựng bộ FA từ các bộ HA như trên hình 4.2.2.5.



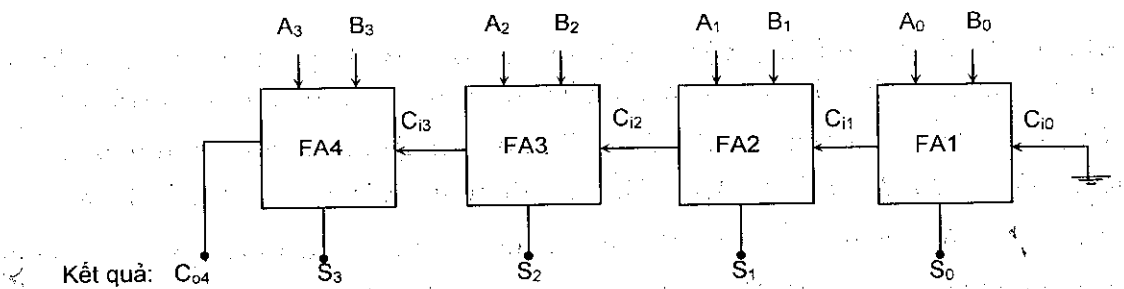
Hình 4.2.2.5. Sơ đồ mạch và ký hiệu của HA

c) Bộ cộng hai số nhị phân 4 bit

Giả sử có hai số nhị phân 4 bit: $A = a_3a_2a_1a_0$ và $B = b_3b_2b_1b_0$. Cũng tương tự như trong hệ thập phân, phép tính cộng trong hệ nhị phân được thực hiện bắt đầu từ bit có trọng số thấp nhất và số nhớ được cộng vào bit có trọng số cao hơn kề nó.

C_i:	c_{i3} c_{i2} c_{i1} 0	Bit nhớ từ các bit có trọng số thấp hơn nhớ sang
A:	a_3 a_2 a_1 a_0	
+B:	b_3 b_2 b_1 b_0	
S:	c_{o4} s_3 s_2 s_1 s_0	Kết quả có thể có số nhớ c_{o4} được sinh ra.

Do đó, phải sử dụng 4 bộ tổng toàn phần, ở bộ tổng toàn phần thứ nhất không có bit nhớ đưa vào do đó có thể thay bằng bộ tổng bán phần hoặc nối đất đầu vào C_i của bộ tổng toàn phần.

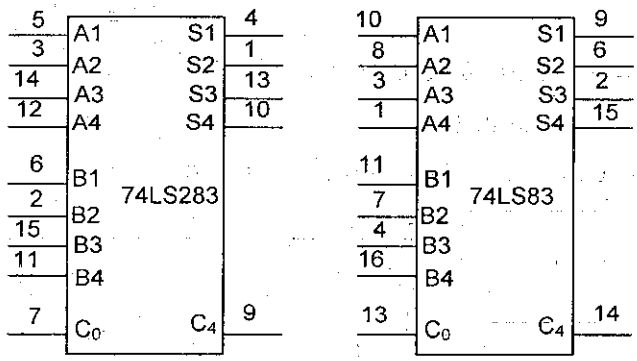


Hình 4.2.2.6. Bộ cộng hai số nhị phân 4

Các bit dữ liệu được đưa vào đồng thời, số nhớ được chuyển từ bit thấp nhất lên. Do đó, nó còn được gọi là bộ cộng song song có nhớ nối tiếp.

Sơ đồ bộ cộng hai số nhị phân 4 bit dùng 4 FA được cho trên hình 4.2.2.6.

Trong thực tế ta thường gặp các vi mạch 7483, 74LS283 là vi mạch 16 chân gồm 4 bộ FA được mắc thành mạch cộng nhị phân hai số nhị phân 4 bit. Hình 4.2.2.7 là ký hiệu logic của các mạch này.

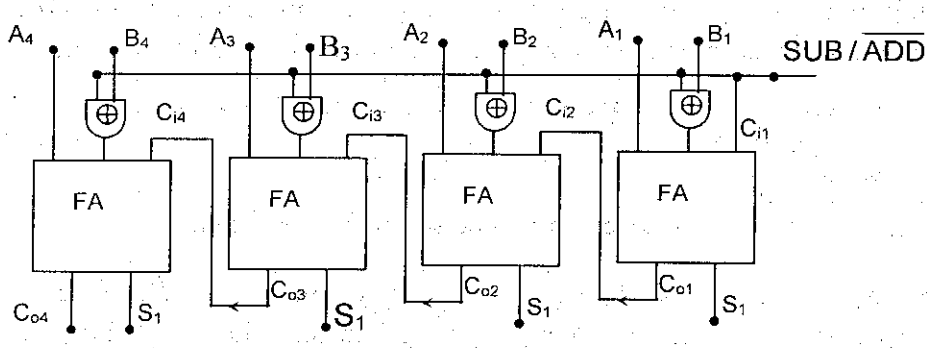


Hình 4.2.2.7. Ký hiệu logic IC 74LS283, 74LS83

Để cộng hai số nhị phân n bit, thực hiện ghép các bộ cộng 4 bit với nhau (C_4 của bộ cộng 4 bit thấp được nối với C_0 của bộ cộng 4 bit cao hơn kề nó).

d) Bộ cộng/trừ hai số nhị phân 4 bit

Muốn dùng các mạch FA để thực hiện cả hai phép tính cộng/ trừ ta cần thêm đầu vào điều khiển SUB/$\overline{\text{ADD}}$ như sơ đồ cộng /trừ 4 bit trên hình 4.2.2.8.



Hình 4.2.2.8. Sơ đồ mạch tổng /hiệu 2 số nhị phân 4 bit

Khi SUB/$\overline{\text{ADD}}$ = 0 (phép cộng): các số liệu $B_4B_3B_2B_1$ qua các cửa XOR không đổi và được đưa vào FA để làm phép cộng hai số A và B, kết quả S = $C_{o4}S_4S_3S_2S_1$.

Khi SUB/$\overline{\text{ADD}}$ = 1 (phép trừ): các số liệu $B_4B_3B_2B_1$ sẽ bị đảo khi đi qua các cửa XOR tức là $B_4B_3B_2B_1 \rightarrow \overline{B_4}\,\overline{B_3}\,\overline{B_2}\,\overline{B_1}$. Đầu C_{i1} được nối với SUB/$\overline{\text{ADD}}$ = 1 tức là C_{i1} = 1. Như vậy, số bù một $\overline{B_4}\ \overline{B_3}\ \overline{B_2}\ \overline{B_1}$ được cộng với C_{i1} = 1 trở thành số bù 2, nghĩa là mạch thực hiện A + (– B). Vậy kết quả của phép trừ là D = $C_{o4}\ S_4S_3S_2S_1$. Trong kết quả này C_{o4} không cần quan tâm, kết quả thực sự là D = $S_4S_3S_2S_1$.

Trong thực tế, ta có thể dùng vi mạch cộng nhị phân 4 bit 74LS283 hoặc 74LS83 ghép với vi mạch 74LS86 (có 4 cửa XOR) sẽ được một bộ cộng/ trừ 4 bit như sơ đồ 4.2.2.8.

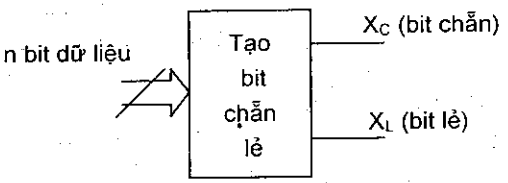
4.2.3. Mạch tạo và kiểm tra chẵn lẻ (Parity Generator/Checker)

Trong quá trình truyền dữ liệu từ nơi này sang nơi khác, do một số yếu tố có thể làm cho thông tin sai lệch đi. Có nhiều phương pháp để xác định xem thông tin truyền đi có bị lỗi hay không. Trong đó, có một phương pháp đơn giản đó là thêm một bit vào dữ liệu được truyền đi sao cho số bit 1 trong dữ liệu luôn là một số chẵn hoặc lẻ. Bit thêm vào đó được gọi là bit chẵn lẻ.

Như vậy, trong quá trình truyền dữ liệu ta phải thực hiện đưa thêm bit chẵn lẻ vào và phải kiểm tra tính chẵn, lẻ của hệ.

a) Mạch tạo bit chẵn lẻ

– Sơ đồ khối hình 4.2.3.1.



Hình 4.2.3.1. Mạch tạo bit chẵn, lẻ

Ví dụ 4.2.3.1. Với 3 bit dữ liệu là X_3, X_2, X_1 và X_C, X_L là hai bit chẵn, lẻ thêm vào dữ liệu.

Mạch có ba đầu vào tương ứng với ba bit dữ liệu, hai đầu ra là bit chẵn, lẻ.

– Bảng trạng thái cho trên hình 4.2.3.2.

X_3	X_2	X_1	X_C	X_L
0	0	0	0	1
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	0

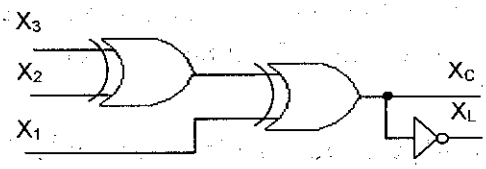
Hình 4.2.3.2. Bảng trạng thái của mạch tạo bit chẵn lẻ

– Từ bảng trạng thái ta có:

$$X_C = X_3 \oplus X_2 \oplus X_1$$

$$X_L = \overline{X_C}$$

– Sơ đồ mạch như trên hình 4.2.3.3.



Hình 4.2.3.3. Mạch tạo bit chẵn, lẻ

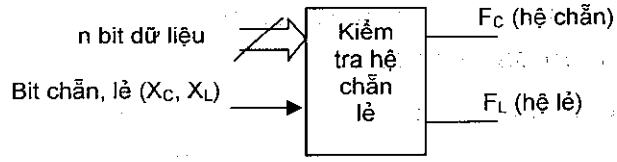
Với n bit dữ liệu quá trình thực hiện cũng tương tự như trên, ta sẽ có được:

$$X_C = X_n \oplus X_{n-1} \oplus ... \oplus X_1$$

$$X_L = \overline{X_C}$$

b) Mạch kiểm tra chẵn lẻ của hệ

– Sơ đồ khối hình 4.2.3.4.



Hình 4.2.3.4. Mạch kiểm tra chẵn, lẻ

– Bảng trạng thái cho trên hình 4.2.3.5.

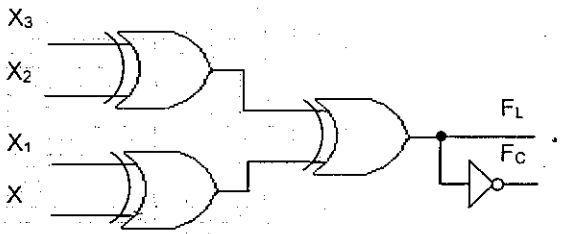
Đầu vào				Đầu ra	
Dữ liệu			Bit chẵn, lẻ	Hệ chẵn	Hệ lẻ
X_3	X_2	X_1	X	F_C	F_L
0	0	0	0	1	0
0	0	0	1	0	1
0	0	1	0	0	1
0	0	1	1	1	0
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	1	0	1
1	0	0	0	0	1
1	0	0	1	1	0
1	0	1	0	1	0
1	0	1	1	0	1
1	1	0	0	1	0
1	1	0	1	0	1
1	1	1	0	0	1
1	1	1	1	1	0

Hình 4.2.3.5. Bảng trạng thái của mạch kiểm tra chẵn lẻ

– Phương trình: $F_L = X_3 \oplus X_2 \oplus X_1 \oplus X$

$F_C = \overline{F_L}$

– Sơ đồ mạch như trên hình 4.2.3.6.



Hình 4.2.3.6. Mạch kiểm tra hệ chẵn, lẻ

Với n bit dữ liệu quá trình thực hiện cũng tương tự như trên, ta sẽ có được:

$F_L = X_n \oplus X_{n-1} \oplus ... \oplus X_1 \oplus X$

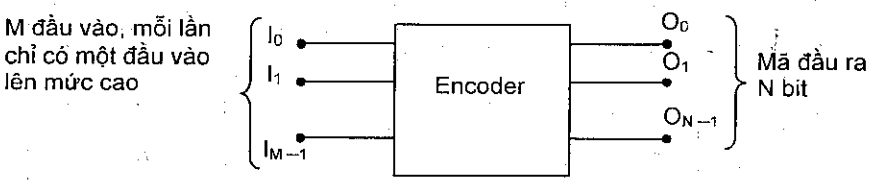
$F_C = \overline{F_L}$

4.3. CÁC MẠCH CHUYỂN ĐỔI MÃ

4.3.1. Mạch mã hoá (Encoder)

Bộ mã hoá có M đầu vào và chỉ một trong số đó được kích hoạt tại thời điểm xác định, tạo mã đầu ra N bit, tuỳ thuộc vào đầu vào nào được kích hoạt.

Sơ đồ tổng quát của bộ mã hoá có M đầu vào và N đầu ra tích cực ở mức cao được cho trên hình 4.3.1.1



Hình 4.3.1.1. Sơ đồ tổng quát của bộ mã

a) Bộ mã hoá bát phân thành nhị phân

I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	O_2	O_1	O_0
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

Hình 4.3.1.2. Bảng chân lý của bộ mã hoá bát phân thành nhị phân

Mạch có 8 đầu vào tương ứng với 8 ký số trong hệ bát phân và tạo mã đầu ra 3 bit tương ứng với các đầu vào được kích hoạt.

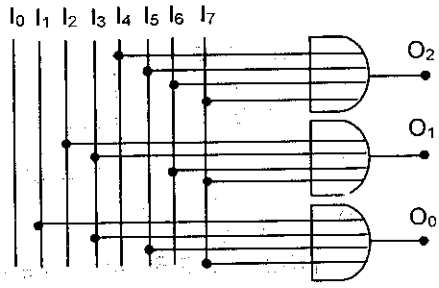
– Bảng trạng thái cho trên hình 4.3.1.2.

– Phương trình logic:

$O_2 = I_4 + I_5 + I_6 + I_7$

$O_1 = I_2 + I_3 + I_6 + I_7$

$O_0 = I_1 + I_3 + I_5 + I_7$



Hình 4.3.1.3. Mã hoá bát phân sang nhị phân

– Sơ đồ logic được cho trên hình 4.3.1.3.

Từ sơ đồ trên ta thấy rằng chỉ được phép kích hoạt mỗi lần một đầu vào, nếu cùng một lúc kích hoạt từ hai đầu vào trở lên thì đầu ra sẽ là bất kỳ một giá trị nào đó không xác định được trước.

Ví dụ, kích hoạt I_3 và I_5 cùng một lúc thì lối ra sẽ có giá trị là 111. Rõ ràng đây không phải là mã cho cả hai đầu vào được kích hoạt.

Để khắc phục nhược điểm này người ta dùng bộ mã hoá ưu tiên.

b) Bộ mã hoá ưu tiên thập phân thành BCD (IC 74147)

I_9	I_8	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	O_3	O_2	O_1	O_0
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	x	0	0	0	1
0	0	0	0	0	0	0	1	x	x	0	0	1	0
0	0	0	0	0	0	1	x	x	x	0	0	1	1
0	0	0	0	0	1	x	x	x	x	0	1	0	0
0	0	0	0	1	x	x	x	x	x	0	1	0	1
0	0	0	1	x	x	x	x	x	x	0	1	1	0
0	0	1	x	x	x	x	x	x	x	0	1	1	1
0	1	x	x	x	x	x	x	x	x	1	0	0	0
1	x	x	x	x	x	x	x	x	x	1	0	0	1

Hình 4.3.1.4. Bảng trạng thái mạch mã hoá ưu tiên thập phân thành BCD

Thứ tự ưu tiên do nhà thiết kế, ở đây ta lấy thứ tự ưu tiên từ cao xuống thấp. Nếu có nhiều tín hiệu đồng thời xuất hiện ở đầu vào thì chỉ có tín hiệu nào có mức ưu tiên cao nhất trong số đó mới được mã hoá.

Mười đầu vào tương ứng với 10 chữ số thập phân được ký hiệu $I_0 \div I_9$ và 4 đầu ra tạo thành mã nhị phân 4 bit được ký hiệu từ $O_3 \div O_0$.

– Bảng trạng thái cho trên hình 4.3.1.4.

– Phương trình logic:

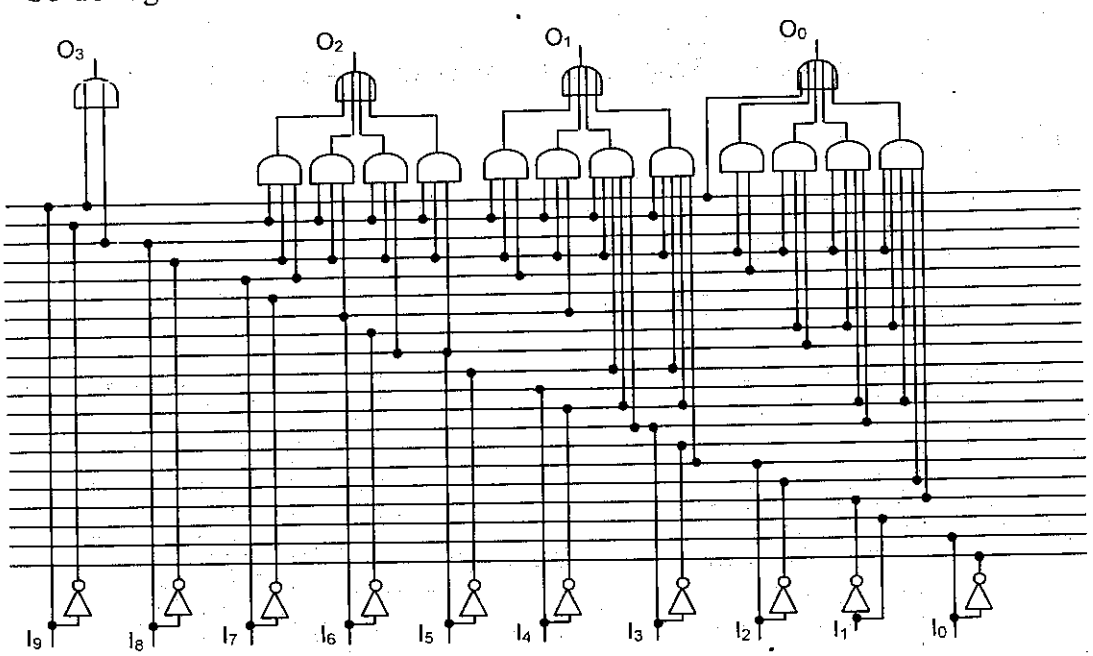
$$O_3 = I_9 + \overline{I_9}\, I_8 = I_9 + I_8$$

$$O_2 = \overline{I_9}\,\overline{I_8}\,\overline{I_7}\,\overline{I_6}\,\overline{I_5}\, I_4 + \overline{I_9}\,\overline{I_8}\,\overline{I_7}\,\overline{I_6}\, I_5 + \overline{I_9}\,\overline{I_8}\,\overline{I_7}\, I_6 + \overline{I_9}\,\overline{I_8}\, I_7$$
$$= \overline{I_9}\,\overline{I_8}\, I_7 + \overline{I_9}\,\overline{I_8}\, I_6 + \overline{I_9}\,\overline{I_8}\, I_5 + \overline{I_9}\,\overline{I_8}\, I_4$$

$$O_1 = \overline{I_9}\,\overline{I_8}\,\overline{I_7}\,\overline{I_6}\,\overline{I_5}\,\overline{I_4}\, I_3 + \overline{I_9}\,\overline{I_8}\,\overline{I_7}\,\overline{I_6}\,\overline{I_5}\,\overline{I_4}\,\overline{I_3}\, I_2 + \overline{I_9}\,\overline{I_8}\,\overline{I_7}\, I_6 + \overline{I_9}\,\overline{I_8}\, I_7$$
$$= \overline{I_9}\,\overline{I_8}\, I_7 + \overline{I_9}\,\overline{I_8}\, I_6 + \overline{I_9}\,\overline{I_8}\,\overline{I_5}\,\overline{I_4}\, I_2 + \overline{I_9}\,\overline{I_8}\,\overline{I_5}\,\overline{I_4}\, I_3$$

$$O_0 = I_9 + \overline{I_9}\,\overline{I_8}\, I_7 + \overline{I_9}\,\overline{I_8}\,\overline{I_7}\,\overline{I_6}\, I_5 + \overline{I_9}\,\overline{I_8}\,\overline{I_7}\,\overline{I_6}\,\overline{I_5}\,\overline{I_4}\, I_3 + \overline{I_9}\,\overline{I_8}\,\overline{I_7}\,\overline{I_6}\,\overline{I_5}\,\overline{I_4}\,\overline{I_3}\,\overline{I_2}\, I_1$$
$$= I_9 + \overline{I_8}\, I_7 + \overline{I_8}\,\overline{I_6}\, I_5 + \overline{I_8}\,\overline{I_6}\,\overline{I_4}\, I_3 + \overline{I_8}\,\overline{I_6}\,\overline{I_4}\,\overline{I_2}\, I_1$$

– Sơ đồ logic như trên hình 4.3.1.5.



Hình 4.3.1.5. Bộ mã hoá ưu tiên thập phân thành BCD

4.3.2. Mạch giải mã (Decoder)

Quá trình ngược với mã hoá được gọi là giải mã. Nghĩa là từ một tổ hợp giá trị của nhóm mã n chữ số hệ 2 ta tìm lại được 1 trong N ký hiệu hoặc số tương ứng với tổ hợp đó. Về thực chất các bộ giải mã cũng là các bộ biến đổi mã, chúng biến đổi từ

các mã nhị phân, BCD sang mã thập phân hay mã 7 đoạn. Để xây dựng các bộ giải mã chúng ta có thể áp dụng phương pháp thiết kế logic chúng ta đã làm quen ở các phần trước để tạo thành các bộ giải mã từ các phần tử logic cơ bản. Thực tế hiện nay người ta không làm như vậy, mà thường dùng các vi mạch giải mã có sẵn trên thị trường. Trong mục này sẽ giới thiệu một số vi mạch để người học có điều kiện nắm vững nguyên tắc hoạt động của các mạch này, dễ dàng dùng được chúng.

a) Bộ giải mã 3 đường thành 8 đường (1 trong 8)

– Bảng trạng thái cho trên hình 4.3.2.1.

Đầu vào			Đầu ra							
A	B	C	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

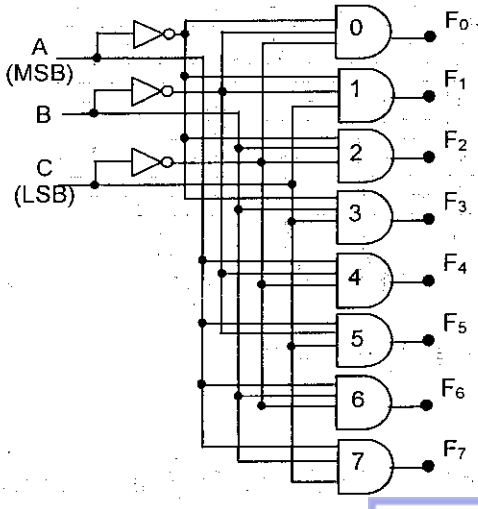
Hình 4.3.2.1. Bảng trạng thái mạch giải mã 3 sang 8

– Phương trình logic:

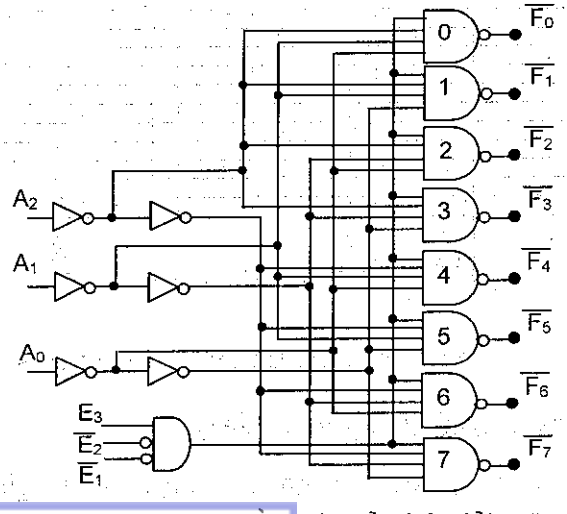
$$F_0 = \overline{A}\,\overline{B}\,\overline{C}; \quad F_1 = \overline{A}\,\overline{B}C; \quad F_2 = \overline{A}B\overline{C}; \quad F_3 = \overline{A}BC$$

$$F_4 = A\overline{B}\,\overline{C}; \quad F_5 = A\overline{B}C; \quad F_6 = AB\overline{C}; \quad F_7 = ABC$$

– Sơ đồ logic như hình 4.3.2.2.

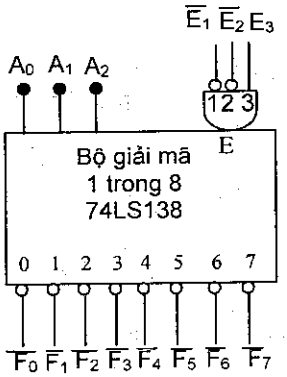


Hình 4.3.2.2. Bộ giải mã 1 trong 8



Hình 4.3.2.3. Sơ đồ logic của bộ giải mã 74LS138

Trong thực tế bộ giải mã 1 trong 8 được tích hợp dưới dạng IC có tên là bộ giải mã 74LS138. Sơ đồ logic của IC 74LS138 được cho trên hình 4.3.2.3 và ký hiệu logic được cho trên hình 4.3.2.4.



Hình 4.3.2.4. Ký hiệu logic của IC 74LS138

b) Giải mã BCD sang thập phân

Bộ giải mã BCD sang thập phân có các lối vào A, B, C, D; các lối ra là F_0, F_1, .., F_9 biểu diễn các số thập phân từ 0, 1,.., 9.

– Bảng trạng thái cho trên hình 4.3.2.5.

A	B	C	D	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9
0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1

Hình 4.3.2.5. Bảng trạng thái bộ giải mã BCD sang thập phân

– Phương trình logic:

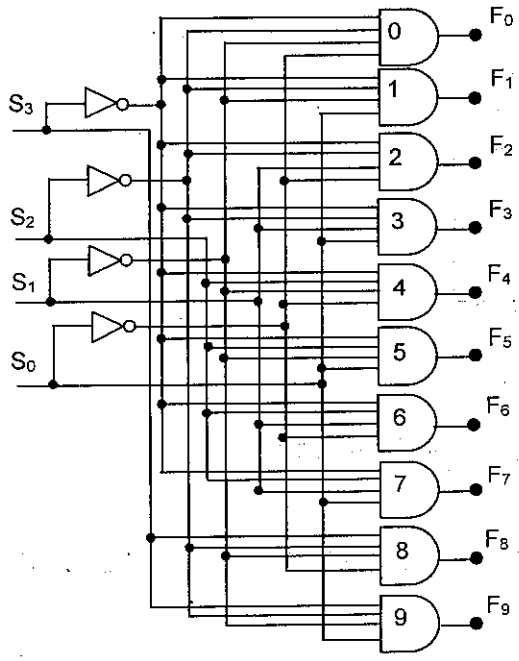
$$F_0 = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}\;;\quad F_1 = \overline{AB\,CD}\;;\quad F_2 = \overline{AB}\overline{CD}\;;\quad F_3 = \overline{A}BCD\;;\quad F_4 = \overline{A}B\overline{CD}$$

$$F_5 = \overline{A}B\overline{C}D\;;\quad F_6 = \overline{A}BC\overline{D}\;;\quad F_7 = \overline{A}BCD\;;\quad F_8 = A\overline{BCD}\;;\quad F_9 = A\overline{B}\overline{C}D$$

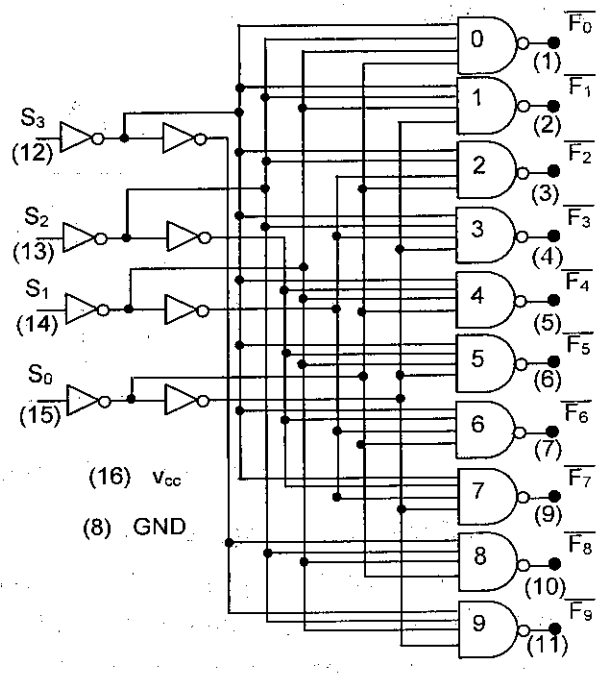
Từ hàm logic trên, ta có thể xây dựng sơ đồ giải mã này khi dùng 4 mạch NOT và 10 mạch NAND 4 lối vào. Sơ đồ mạch như trên hình 4.3.2.6.

Các vi mạch giải mã có trong thực tế như 7442, 74L42, 74LS42, 7445, 74145 các vi mạch này có 16 chân có 4 lối vào A, B, C, D và 10 chân lối ra tác động thấp 0,

1,..., 9. Các IC này tuy có ký hiệu khác nhau nhưng đều có cùng một sơ đồ logic và ký hiệu các chân giống nhau như hình 4.3.2.7.



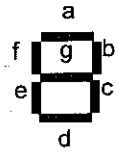
Hình 4.3.2.6. Mạch giải mã BCD sang mã10



Hình 4.3.2.7. Sơ đồ logic của IC 7442

c) Mạch giải mã BCD sang mã 7 đoạn

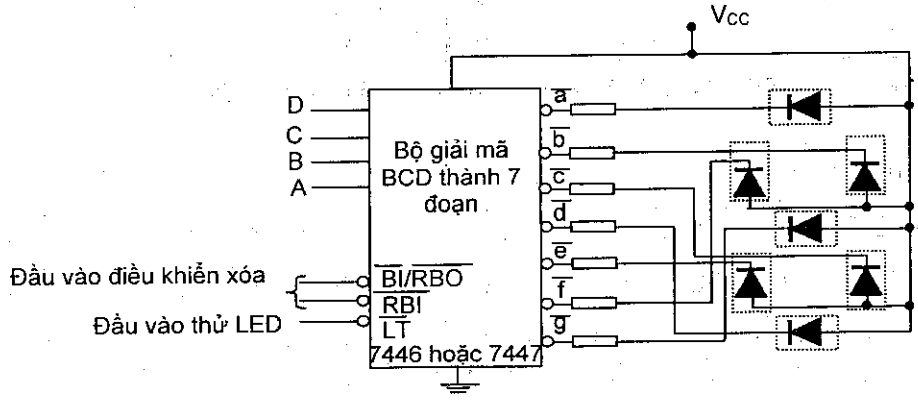
Dạng chỉ thị 7 đoạn như hình 4.3.2.8.



Hình 4.3.2.8. Dạng chỉ thị 7 đoạn

Mã nhị phân BCD được chuyển sang thập phân và hiển thị các số thập phân bằng 7 đoạn sáng, ứng với mỗi tổ hợp xác định các thanh sáng sẽ hiển thị cho ta một chữ số ở hệ 10.

Các đoạn a, b, c, d, e, f, g có thể là đèn LED mắc anôt chung hoặc catôt chung, có thể là màn hình tinh thể lỏng.



Hình 4.3.2.9. Bộ giải mã BCD thành 7 đoạn đang điều khiển một màn hình LED 7 đoạn có anôt chung

Với LED mắc anôt chung (có nghĩa là anôt của tất cả các đoạn được gắn chung với V_{CC}), catôt được nối qua các điện trở giới hạn dòng tới đầu ra phù hợp của bộ giải mã. Bộ giải mã có đầu ra tích cực ở mức thấp.

Với LED mắc catôt chung (có nghĩa là catôt của tất cả các đoạn được nối đất), bộ giải mã có đầu ra tích cực cao.

Với màn hình tinh thể lỏng LCD, bộ giải mã có đầu ra tích cực cao.

Ví dụ, sơ đồ mắc IC giải mã 7 đoạn có đầu ra tích cực ở mức thấp với đèn LED chỉ thị như trên hình 4.3.2.9.

Số thập phân	Mã BCD đầu vào				Đầu ra mã 7 đoạn						
	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

Hình 4.3.2.10. Bảng trạng thái của bộ chuyển đổi mã BCD sang mã 7 đoạn

– Bảng trạng thái của bộ chuyển đổi mã BCD sang mã 7 đoạn với đầu ra tích cực cao cho trên hình 4.3.2.10. Trong đó có 6 tổ hợp số nhị phân 4 bit còn lại (1010 đến 1111) không xuất hiện trong mã BCD vì thế nó tương ứng với các tổ hợp lỗi. Khi tối thiểu hoá hàm ta có thể cho 6 tổ hợp đó hàm không xác định hoặc bằng 0. Với trường hợp hàm không xác định thì phương trình sẽ đơn giản hơn, khi đầu vào bị lỗi thì sẽ có đoạn sáng đoạn tối nhưng không trùng với bất kỳ chữ số nào. Xét bảng Karnaugh trên hình 4.3.2.11 cho các hàm a,b,c,d,e,f,g với trường hợp hàm không xác định.

– Phương trình của các hàm:

$a = \overline{B}\,\overline{D} + A + C + BD$

$b = \overline{B} + \overline{C}\,\overline{D} + CD$

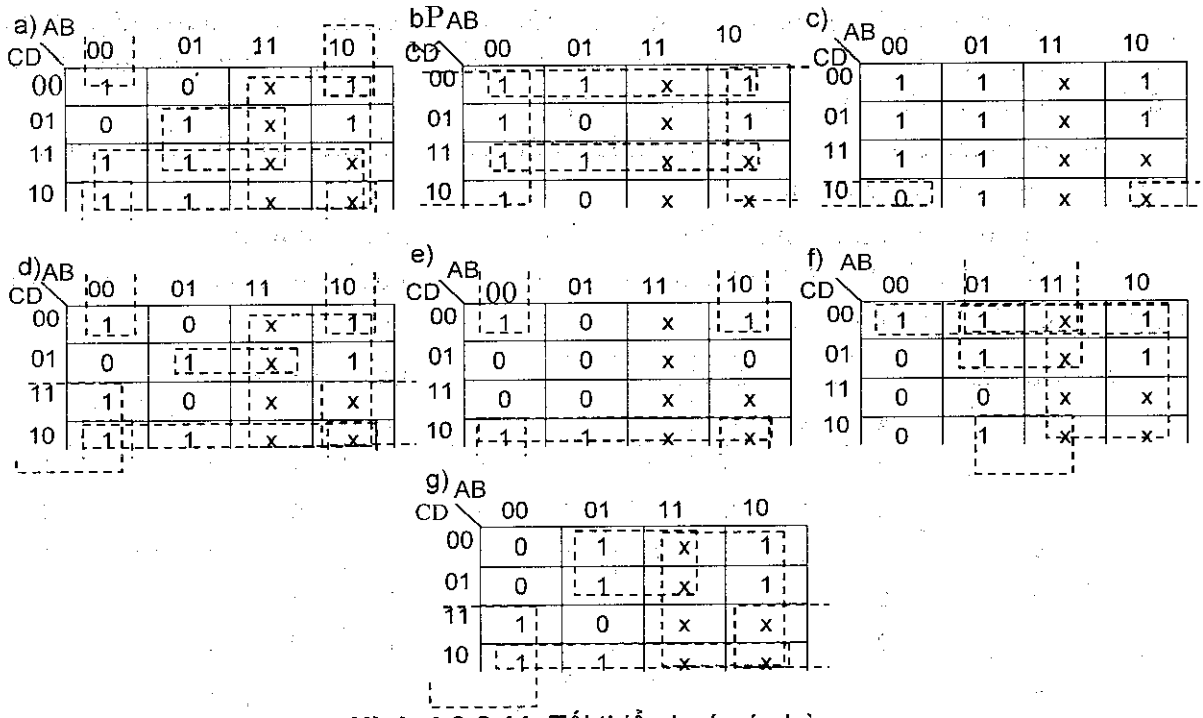
$c = B + \overline{C} + D$

$d = A + \overline{B}\,\overline{D} + \overline{B}C + B\overline{C}D$

$e = \overline{B}\,\overline{D} + C\overline{D}$

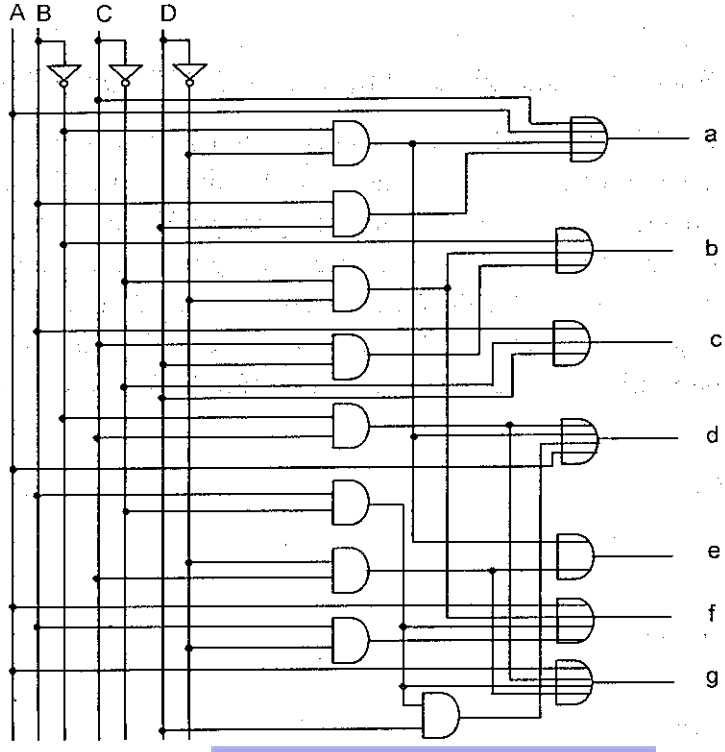
$f = A + \overline{C}\,\overline{D} + B\overline{C} + B\overline{D}$

$g = A + \overline{B}C + C\overline{D} + B\overline{C}$



Hình 4.3.2.11. Tối thiểu hoá các hàm

– Sơ đồ mạch như trên hình 4.3.2.12.

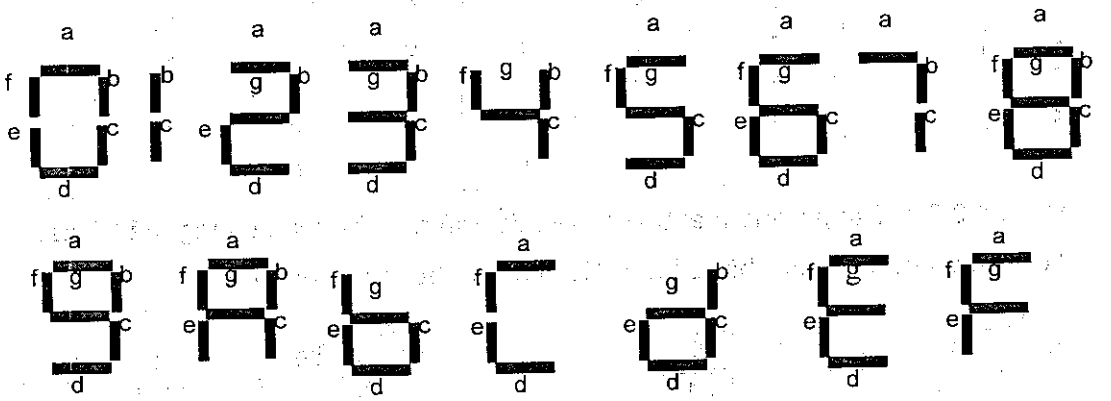


Hình 4.3.2.12. Mạch chuyển đổi mã BCD sang mã 7 đoạn

Trong thực tế người ta đã chế tạo sẵn các vi mạch để giải mã nhị phân ra 7 đoạn:

Các vi mạch 7448, 74LS48, 7449, 74LS49 là các IC giải mã 7 đoạn có lối ra tác động ở mức cao. Ta có thể dùng chúng để giải mã từ mã BCD ra thập phân. Quy luật hiển thị các chữ số thập phân của các vi mạch này về cơ bản giống như bảng chân lý trên, chỉ khác đôi chút là số 6 không dùng thanh a và số 9 không dùng thanh d. Các mạch giải mã 7 đoạn 7447A, 74L47, 74S47 là các vi mạch 16 chân, số 6 và số 9 chỉ có 5 thanh sáng giống như 7448, 7449. Vi mạch có lối ra tác động thấp (mức 0) nên đèn chỉ thị 7 đoạn có anôt chung.

Mạch giải mã MC 14495 cũng là giải mã nhị phân ra 7 đoạn. Vi mạch 16 chân, có lối ra tác động cao (mức 1) nên đèn chỉ thị 7 đoạn có catôt chung. Số 6 và số 9 có 6 thanh sáng, các số thập phân: 10, 11, 12, 13, 14, 15 được hiển thị giống như các chữ số trong hệ thập lục phân. Trên hình 4.3.2.13 minh hoạ sự hiển thị của các đèn chỉ thị số theo mã 7 đoạn khi nó được dùng với mạch giải mã MC 14495.



Hình 4.3.2.13. Sự hiển thị các chữ số của IC MC 14495

4.3.3. Các mạch chuyển đổi mã khác

Mạch chuyển đổi mã (Code Converter) là mạch logic thay đổi dữ liệu nhị phân từ dạng này sang dạng khác. Các mạch chuyển đổi mã thông dụng như: BCD sang 7 đoạn, nhị phân sang bù hai, bù hai sang nhị phân, nhị phân sang Gray, Gray sang nhị phân, BCD sang nhị phân, nhị phân sang BCD, BCD sang dư 3, dư 3 sang BCD,... Sau đây chúng ta sẽ xét một số mạch chuyển đổi mã.

a) Bộ biến đổi mã nhị phân 4 bit sang mã bù 2

Trong máy tính điện tử người ta dùng số bù 2 để biểu diễn số âm, nhờ vậy mà người ta có thể dễ dàng thực hiện phép tính trừ bằng cách cộng số bị trừ với số bù 2 của số trừ.

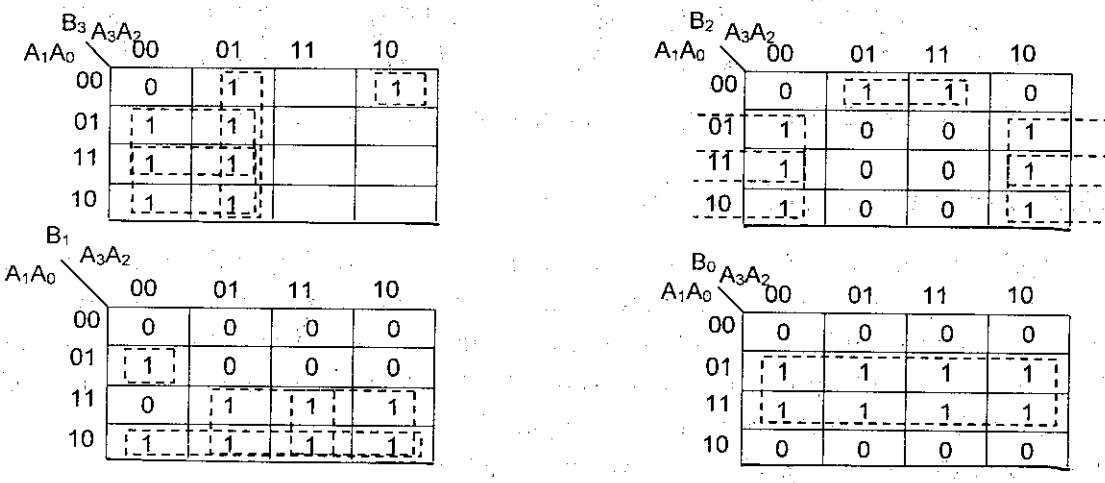
Dựa vào quy tắc tìm số bù 2 ta lập được bảng chân lý của bộ biến đổi mã có các đầu vào là A_0, A_1, A_2, A_3 là mã nhị phân 4 bit, bốn đầu ra là 4 bit mã bù 2 B_0, B_1, B_2, B_3.

– Bảng trạng thái bộ chuyển đổi từ mã nhị phân 4 bit sang mã bù 2 cho trên hình 4.3.3.1.

Số thập phân	Mã nhị phân				Mã bù 2			
	A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	1	1	1	1
2	0	0	1	0	1	1	1	0
3	0	0	1	1	1	1	0	1
4	0	1	0	0	1	1	0	0
5	0	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	0
7	0	1	1	1	1	0	0	1
8	1	0	0	0	1	0	0	0
9	1	0	0	1	0	1	1	1
10	1	0	1	0	0	1	1	0
11	1	0	1	1	0	1	0	1
12	1	1	0	0	0	1	0	0
13	1	1	0	1	0	0	1	1
14	1	1	1	0	0	0	1	0
15	1	1	1	1	0	0	0	1

Hình 4.3.3.1. Bảng trạng thái bộ chuyển đổi mã nhị phân 4 bit sang mã bù hai.

– Tối thiểu hoá các hàm bằng bảng Karnaugh như trên hình 4.3.3.2.



Hình 4.3.3.2. Tối thiểu hoá các hàm

– Phương trình logic:

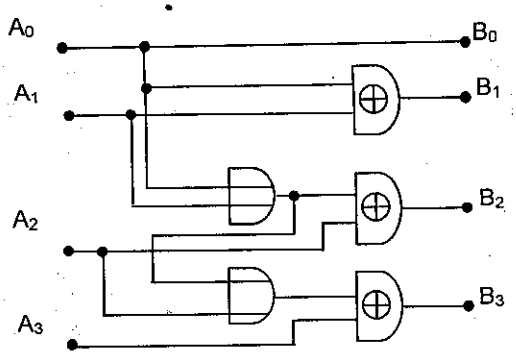
$B_0 = A_0$

$B_1 = A_0 \oplus A_1$

$B_2 = (A_0 + A_1) \oplus A_2$

$B_3 = (A_0 + A_1 + A_2) \oplus A_3$

– Sơ đồ logic như trên hình 4.3.3.3.



Hình 4.3.3.3. Mạch chuyển đổi mã nhị phân 4 bit sang mã bù hai

b) Bộ biến đổi mã nhị phân 4 bit sang mã Gray

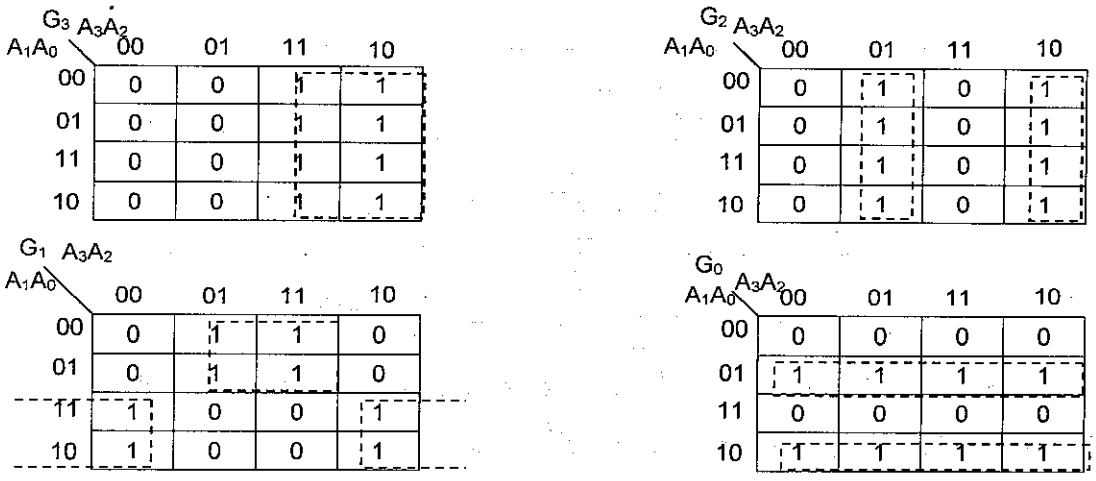
Gọi các bit của số nhị phân 4 bit tương ứng là $A_3A_2A_1A_0$, các bit của mã Gray là $G_3G_2G_1G_0$.

– Bảng trạng thái bộ chuyển đổi mã nhị phân 4 bit sang mã Gray cho trên hình 4.3.3.4.

Số thập phân	Mã nhị phân				Mã Gray			
	A_3	A_2	A_1	A_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

Hình 4.3.3.4. Bảng trạng thái bộ chuyển đổi mã nhị phân 4 bit sang mã Gray

– Tối thiểu hoá hàm bằng bảng Karnaugh như trên hình 4.3.3.5.



Hình 4.3.3.5. Tối thiểu hoá các hàm

– Phương trình logic:

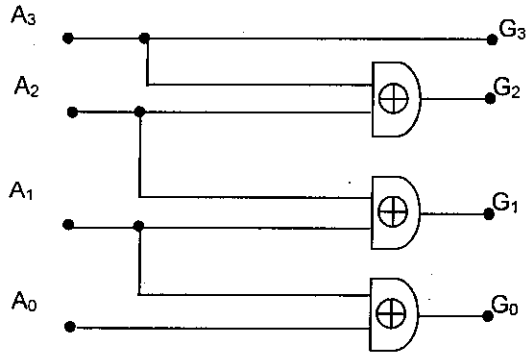
$$G_3 = A_3$$

$$G_2 = A_3 \oplus A_2$$

$$G_1 = A_2 \oplus A_1$$

$$G_0 = A_1 \oplus A_0$$

– Sơ đồ logic cho trên hình 4.3.3.6.



Hình 4.3.3.6. Mạch chuyển mã nhị phân 4 bit sang mã Gray

c) Bộ biến đổi mã Gray 4 bit sang mã nhị phân

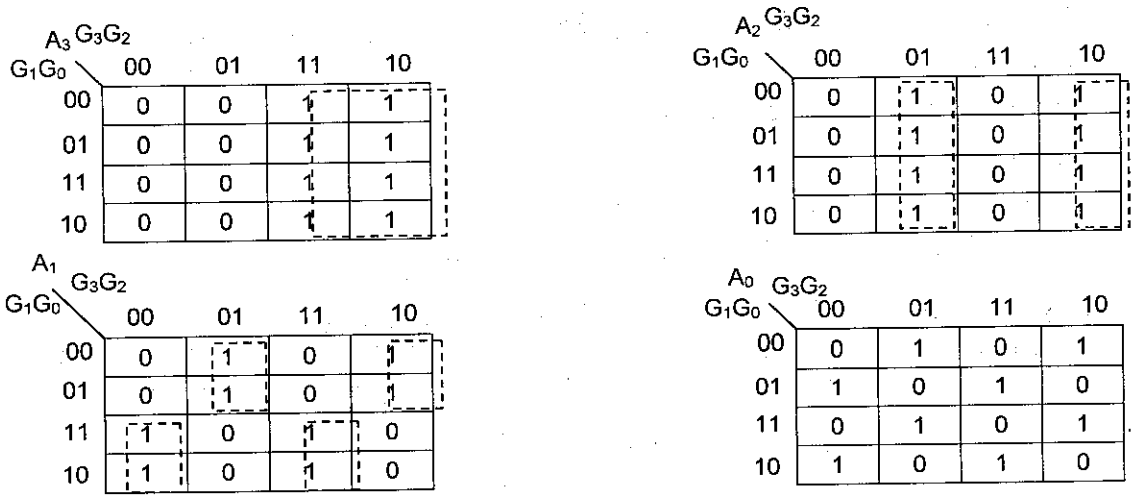
Gọi các bit của mã Gray là $G_3G_2G_1G_0$, các bit của số nhị phân 4 bit tương ứng là $A_3A_2A_1A_0$.

– Bảng trạng thái bộ chuyển đổi mã Gray 4 bit sang mã nhị phân cho trên hình 4.3.3.7.

Số thập phân	Mã Gray				Mã nhị phân			
	G_3	G_2	G_1	G_0	A_3	A_2	A_1	A_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	1	0	0	1	0
3	0	0	1	0	0	0	1	1
4	0	1	1	0	0	1	0	0
5	0	1	1	1	0	1	0	1
6	0	1	0	1	0	1	1	0
7	0	1	0	0	0	1	1	1
8	1	1	0	0	1	0	0	0
9	1	1	0	1	1	0	0	1
10	1	1	1	1	1	0	1	0
11	1	1	1	0	1	0	1	1
12	1	0	1	0	1	1	0	0
13	1	0	1	1	1	1	0	1
14	1	0	0	1	1	1	1	0
15	1	0	0	0	1	1	1	1

Hình 4.3.3.7. Bảng trạng thái mạch chuyển mã Gray 4 bit sang mã nhị phân

– Tối thiểu hoá hàm bằng bảng Karnaugh như trên hình 4.3.3.8.



Hình 4.3.3.8. Tối thiểu hoá các hàm

– Phương trình logic:

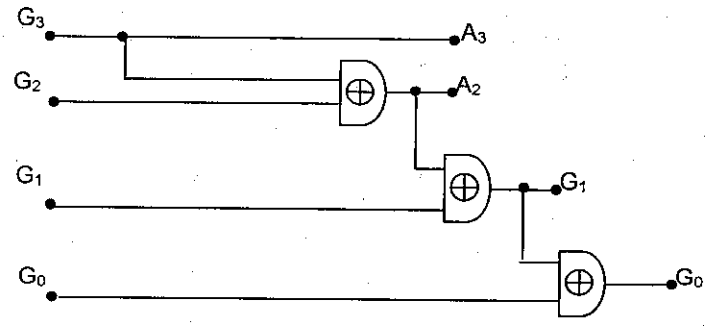
$$A_3 = G_3$$

$$A_2 = G_3 \oplus G_2$$

$$A_1 = G_3 \oplus G_2 \oplus G_1$$

$$A_0 = G_3 \oplus G_2 \oplus G_1 \oplus G_0$$

– Sơ đồ logic cho trên hình 4.3.3.9.



Hình 4.3.3.9. Mạch chuyển mã Gray 4 bit sang nhị phân

Nếu đã biết phương trình của mạch chuyển đổi mã nhị phân sang mã Gray thì theo tính chất của hàm XOR ta có thể suy ra phương trình của mạch chuyển mã Gray sang nhị phân và ngược lại.

d) Mạch chuyển đổi mã BCD sang nhị phân

Bộ chuyển đổi mã BCD sang nhị phân thường được sử dụng khi dữ liệu BCD được truyền đến máy tính để lưu trữ hoặc xử lý. Dữ liệu phải được chuyển thành nhị phân cho ALU của máy tính thao tác, vì ALU không có khả năng thi hành phép toán số học trên dữ liệu BCD.

– Bảng mã nhị phân tương ứng với các trọng số thập phân của mỗi bit BCD 2 ký số: $A_1B_1C_1D_1A_0B_0C_0D_0$ mã BCD tương ứng với số nhị phân $b_6b_5b_4b_3b_2b_1b_0$ cho trên hình 4.3.3.10.

– Phương trình của các bit nhị phân:

$b_6 = A_1$; $b_5 = B_1$;

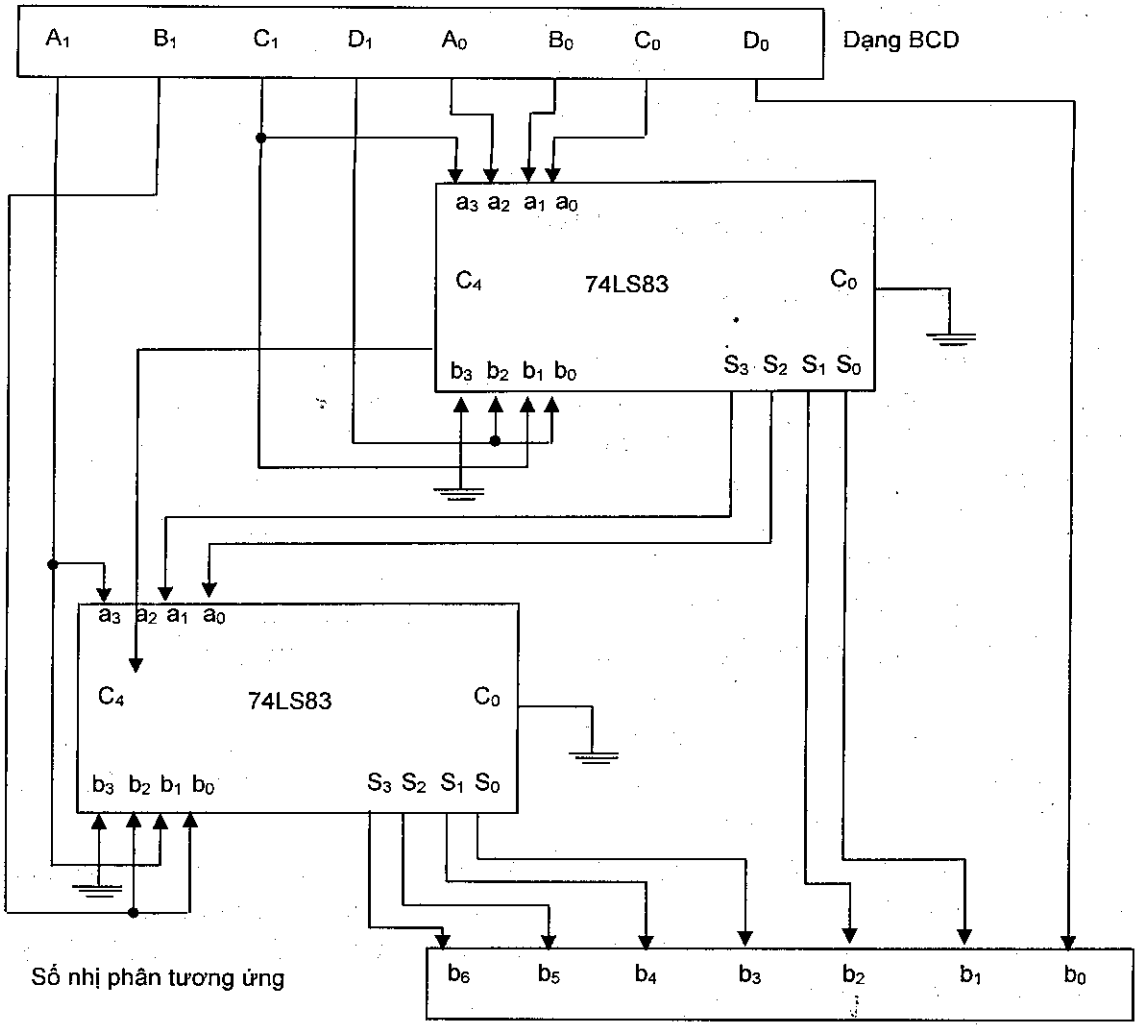
$b_4 = A_1 + C_1$; $b_3 = A_0 + D_1 + B_1$;

$b_2 = C_1 + B_0$; $b_1 = D_1 + C_0$; $b_0 = D_0$

Bit BCD	Trọng số thập phân	Mã nhị phân tương ứng						
		b_6	b_5	b_4	b_3	b_2	b_1	b_0
D_0	1	0	0	0	0	0	0	1
C_0	2	0	0	0	0	0	1	0
B_0	4	0	0	0	0	1	0	0
A_0	8	0	0	0	1	0	0	0
D_1	10	0	0	0	1	0	1	0
C_1	20	0	0	1	0	1	0	0
B_1	40	0	1	0	1	0	0	0
A_1	80	1	0	1	0	0	0	0

Hình 4.3.3.10. Bảng mã nhị phân ứng với các trọng số thập phân của mỗi bit BCD

– Sơ đồ mạch như trên hình 4.3.3.11.



Hình 4.3.3.11. Mạch chuyển đổi BCD sang nhị phân bằng IC 74LS83

e) Một số vi mạch tổ hợp

IC mã hoá:

74148/LS148/HC148	: mã hoá ưu tiên bát phân sang nhị phân
74147/LS147/HC147	: mã hoá ưu tiên thập phân sang BCD
74184, 74185	: chuyển mã BCD sang nhị phân, nhị phân sang BCD

IC giải mã:

7441, 7442/LS42	: giải mã BCD sang thập phân
7443	: giải mã dư 3 sang thập phân
7444	: giải mã Gray dư 3 sang thập phân
7445	: giải mã BCD sang thập phân
7446, 7447	: giải mã BCD sang mã 7 đoạn (đầu ra tích cực thấp)
7448, 7449	: giải mã BCD sang mã 7 đoạn (đầu ra tích cực cao)

74141, 74145/LS145 : giải mã BCD sang thập phân
74LS138 : giải mã 3 sang 8
74LS139 : giải mã 4 sang 16
74155/LS155 : giải mã 2 sang 4

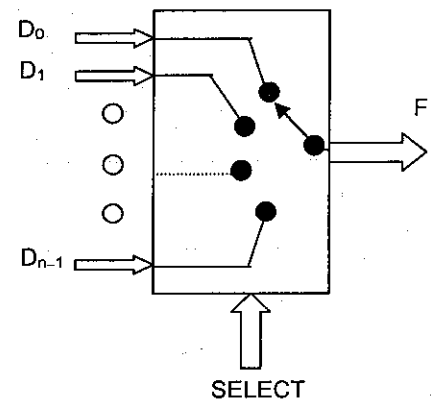
4.4. MẠCH HỢP KÊNH VÀ PHÂN KÊNH

4.4.1. Mạch hợp kênh (MUX)

Mạch hợp kênh số (digital multiplexer) là mạch logic chấp nhận nhiều đầu vào dữ liệu số, chọn ra một trong số chúng tại thời điểm xác định để chuyển đến đầu ra. Hoạt động lộ trình từ đầu vào đến đầu ra do đầu vào SELECT (còn gọi là đầu vào địa chỉ) chi phối.

Sơ đồ chức năng của một bộ hợp kênh số tổng quát được cho trên hình 4.4.1.1.

Đầu vào dữ liệu và đầu ra được vẽ ở dạng mũi tên 2 nét, hàm ý trên thực tế chúng có thể là 2 đường dữ liệu trở lên.



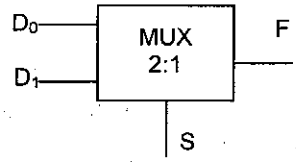
Hình 4.4.1.1. Bộ hợp kênh số tổng quát

Bộ hợp kênh hoạt động như chuyển mạch nhiều vị trí, trong đó mã dạng số áp đến đầu vào SELECT sẽ cho phép đầu vào dữ liệu nào được chuyển đến đầu ra. Nếu có N đầu vào dữ liệu thì cần có N địa chỉ khác nhau bằng cách sử dụng n ký số nhị phân tuân theo điều kiện $2^n \geq N$.

a) Bộ hợp kênh 2 đầu vào (MUX 2 : 1)

Với 2 đầu vào dữ liệu D_0, D_1 và 1 đầu vào địa chỉ S.

– Ký hiệu hình 4.4.1.2:



Hình 4.4.1.2. Ký hiệu MUX2:1

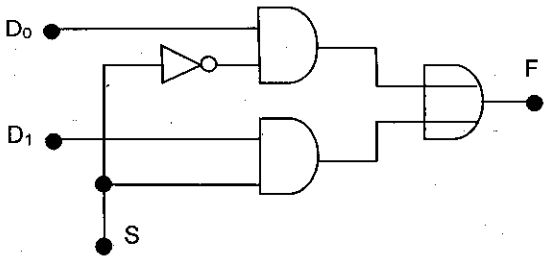
– Bảng trạng thái của MUX2:1 cho trên hình 4.4.1.3:

S	F
0	D_0
1	D_1

Hình 4.4.1.3. Bảng trạng thái của MUX2:1

– Phương trình logic của MUX2:1: $F = \overline{S}D_0 + SD_1$

– Sơ đồ logic của MUX2:1 được cho trên hình 4.4.1.4.



Hình 4.4.1.4. Sơ đồ logic của MUX 2 : 1

Một trong những nơi ứng dụng MUX 2 đầu vào là hệ thống máy vi tính sử dụng hai tín hiệu Master Clock khác nhau: xung nhịp tốc độ cao đối với một số chương trình, xung nhịp tốc độ thấp cho số khác. Hai xung nhịp này được đưa vào hai đầu vào dữ liệu (D_0 và D_1). Tín hiệu từ phần logic điều khiển của máy vi tính sẽ kích thích đầu vào S, để đầu vào này quyết định tín hiệu xung nhịp nào xuất hiện tại đầu ra F định lộ trình đến mạch khác trong máy. .

b) Bộ hợp kênh 4 đầu vào (MUX 4 : 1)

Đầu vào dữ liệu D_0, D_1, D_2, D_3 và đầu vào địa chỉ S_1, S_0. Hai đầu vào địa chỉ sẽ tạo ra 4 tổ hợp khả dĩ, mỗi đầu vào dữ liệu bị chi phối bởi 1 tổ hợp khác nhau của các mức ở đầu vào địa chỉ.

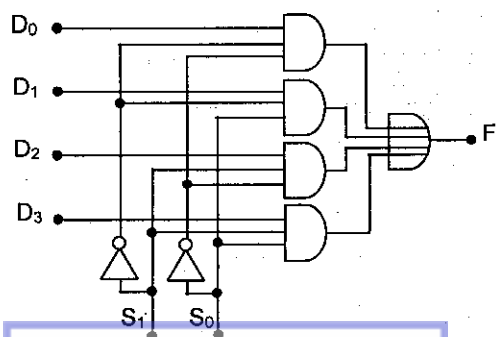
– Bảng trạng thái của MUX4:1 cho trên hình 4.4.1.5.

S_1	S_0	F
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

Hình 4.4.1.5. Bảng trạng thái của MUX 4 : 1

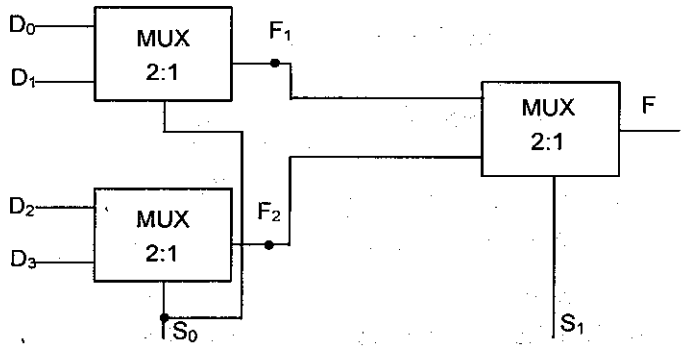
– Phương trình logic của MUX4:1: $F = \overline{S_1}\overline{S_0}D_0 + \overline{S_1}S_0D_1 + S_1\overline{S_0}D_2 + S_1S_0D_3$

– Sơ đồ logic của MUX4:1 được cho trên hình 4.4.1.6.



Hình 4.4.1.6. Sơ đồ logic MUX 4 : 1

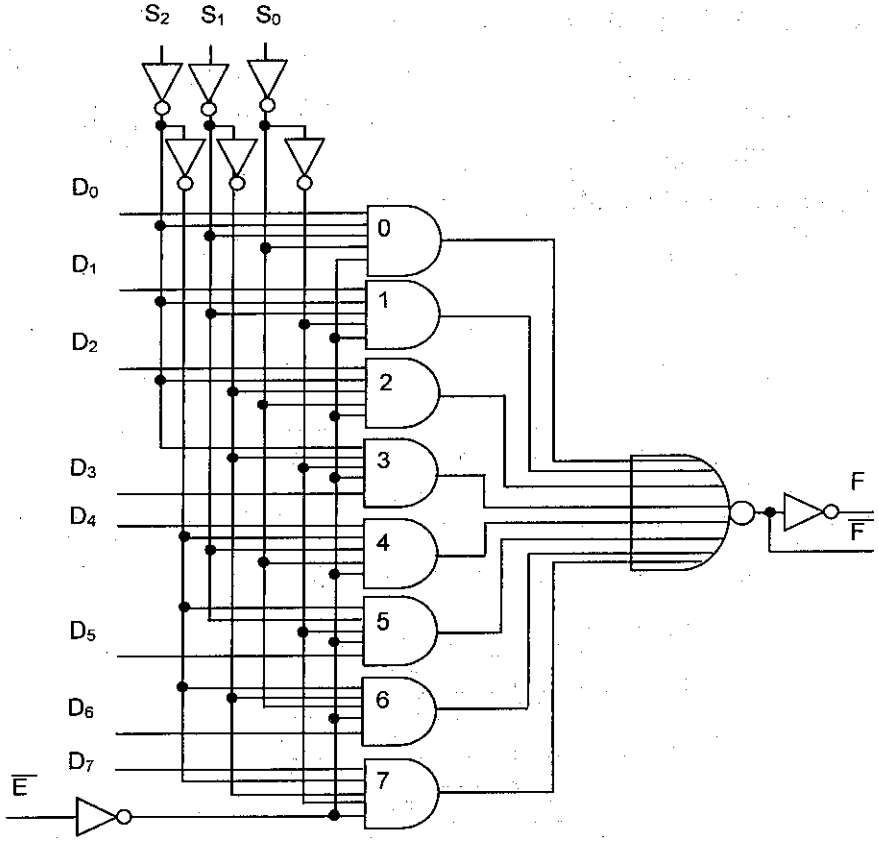
Có thể dùng MUX 2:1 để tạo thành MUX 4:1 như trên hình 4.4.1.7.



Hình 4.4.1.7. Dùng MUX 2 : 1 tạo thành MUX 4 : 1

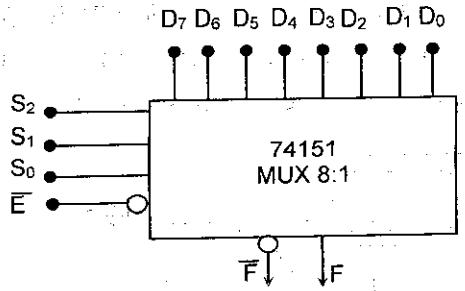
c) Bộ hợp kênh 8 đầu vào (MUX 8 : 1)

Xét sơ đồ logic của bộ hợp kênh 8 đầu vào 74151 (74LS151/HC151) được cho trên hình 4.4.1.8. Đầu vào cho phép ($\overline{E}$), khi $\overline{E}$ = 0, MUX sẽ chọn 1 trong 8 đầu vào dữ liệu đưa tới đầu ra tuỳ theo đầu vào địa chỉ ($S_2S_1S_0$). Khi $\overline{E}$ = 1 thì F = 0.



Hình 4.4.1.8. Sơ đồ logic cho bộ hợp kênh 74151

– Ký hiệu logic được cho trên hình 4.4.1.9.



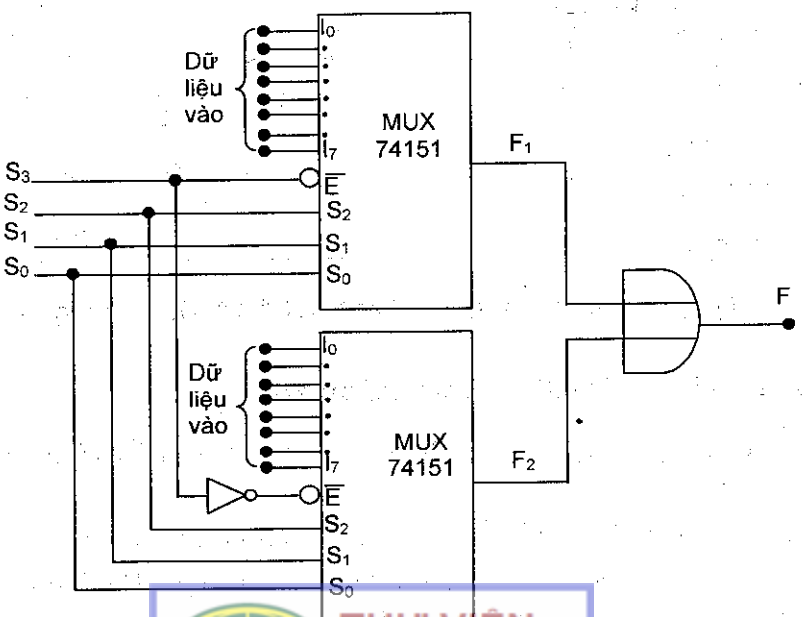
Hình 4.4.1.9. Ký hiệu logic MUX 8 : 1

– Bảng trạng thái cho trên hình 4.4.1.10.

$\overline{E}$	S_2	S_1	S_0	F
1	X	X	X	0
0	0	0	0	D_0
0	0	0	1	D_1
0	0	1	0	D_2
0	0	1	1	D_3
0	1	0	0	D_4
0	1	0	1	D_5
0	1	1	0	D_6
0	1	1	1	D_7

Hình 4.4.1.10. Bảng trạng thái của IC 74151

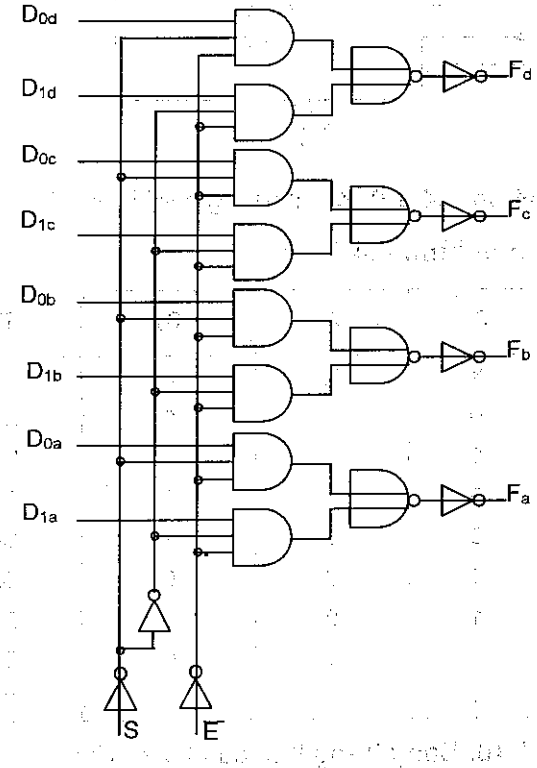
– Dùng 2 IC 74151 kết hợp thành bộ hợp kênh 16 đầu vào như trên hình 4.4.1.11



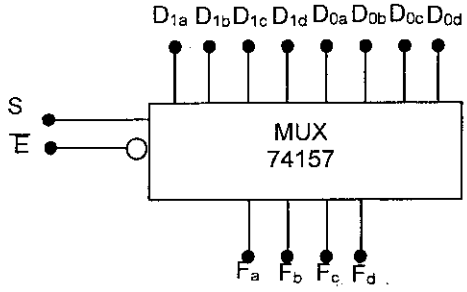
Hình 4.4.1.11. Hai IC 74151 kết hợp thành bộ dồn kênh 16

– Bộ hợp kênh chập 4 hai đầu vào (74157/LS157/HC157) được cho trên hình 4.4.1.12, ký hiệu logic của IC 74157 được cho trên hình 4.4.1.13.

– Bảng trạng thái của IC 74157 cho trên hình 4.4.1.14.



Hình 4.4.1.12. Sơ đồ logic của bộ hợp kênh 74157



Hình 4.4.1.13. Ký hiệu logic của IC 74157

$\overline{E}$	S	F_a	F_b	F_c	F_d
1	X	0	0	0	0
0	0	D_{0a}	D_{0b}	D_{0c}	D_{0d}
0	1	D_{1a}	D_{1b}	D_{1c}	D_{1d}

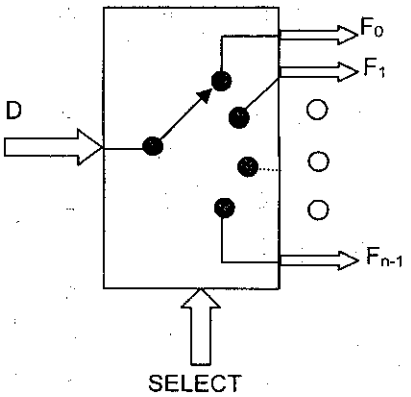
Hình 4.4.1.14. Bảng trạng thái của IC74157

4.4.2. Mạch phân kênh (DMUX – Demultiplexer)

DMUX hoạt động ngược lại với MUX: một đầu vào dữ liệu và phân phối dữ liệu cho nhiều đầu ra.

Sơ đồ khối của bộ phân kênh số tổng quát được cho trên hình 4.4.2.1.

Mã đầu vào SELECT quyết định truyền đầu vào dữ liệu (D) đến đầu ra nào. Nói cách khác, bộ phân kênh lấy một nguồn dữ liệu vào và phân phối có chọn lọc đến 1 trong số n kênh ra, tương tự 1 chuyển mạch nhiều tiếp điểm.



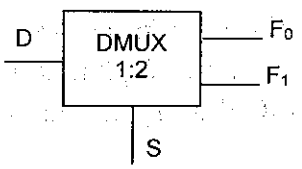
Hình 4.4.2.1. Bộ phân kênh số tổng quát

a) Bộ phân kênh 2 đầu ra (DMUX 1:2)

Một đầu vào dữ liệu D, hai đầu ra F_0, F_1, một đầu vào địa chỉ S.

– Sơ đồ khối cho trên hình 4.4.2.2a.

– Bảng trạng thái cho trên hình 4.4.2.2b:



a)

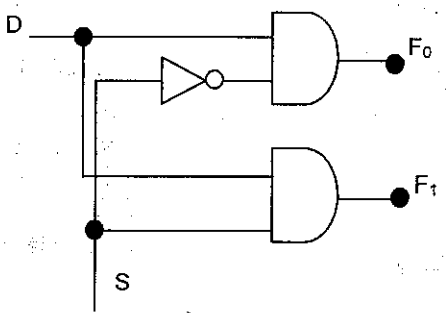
S	F_0	F_1
0	D	0
1	0	D

b)

Hình 4.4.2.2. Sơ đồ khối và bảng trạng thái của DMUX 1 : 2

– Phương trình logic: $F_0 = \overline{S}D$; $F_1 = SD$

– Sơ đồ logic được cho trên hình 4.4.2.3.



Hình 4.4.2.3. Sơ đồ logic của DMUX 1 : 2

b) Bộ phân kênh 8 đầu ra (DMUX 1 : 8)

Lối vào dữ liệu D, các lối ra $F_0 \div F_7$, cần 3 đầu vào địa chỉ S_2, S_1, S_0.

– Bảng trạng thái cho trên hình 4.4.2.4.

S_2	S_1	S_0	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7
0	0	0	D	0	0	0	0	0	0	0
0	0	1	0	D	0	0	0	0	0	0
0	1	0	0	0	D	0	0	0	0	0
0	1	1	0	0	0	D	0	0	0	0
1	0	0	0	0	0	0	D	0	0	0
1	0	1	0	0	0	0	0	D	0	0
1	1	0	0	0	0	0	0	0	D	0
1	1	1	0	0	0	0	0	0	0	D

Hình 4.4.2.4. Bảng trạng thái của DMUX 1 : 8

– Phương trình logic:

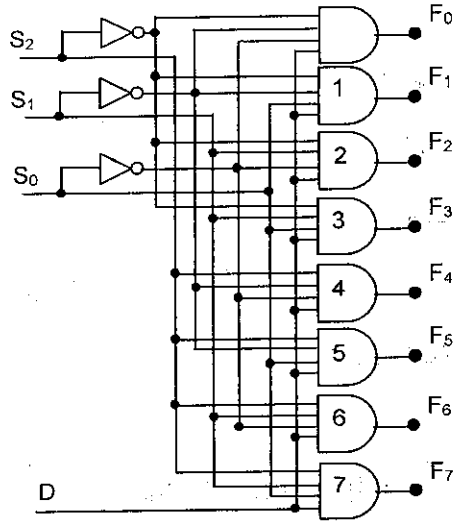
$$F_0 = \overline{S_2}\,\overline{S_1}\,\overline{S_0}D; \quad F_1 = \overline{S_2}\,\overline{S_1}S_0D; \quad F_2 = \overline{S_2}S_1\overline{S_0}D; \quad F_3 = \overline{S_2}S_1S_0D$$

$$F_4 = S_2\overline{S_1}\,\overline{S_0}D; \quad F_5 = S_2\overline{S_1}S_0D; \quad F_6 = S_2S_1\overline{S_0}D; \quad F_7 = S_2S_1S_0D.$$

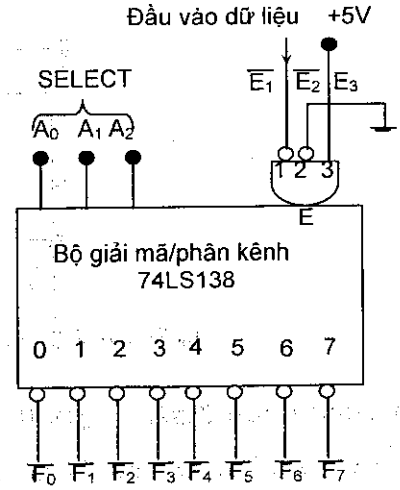
– Sơ đồ logic được cho trên hình 4.4.2.5.

Ta thấy rằng có thể sử dụng bộ phân kênh như bộ giải mã với đầu vào dữ liệu là đầu vào cho phép và ngược lại có thể sử dụng bộ giải mã như bộ phân kênh với đầu vào cho phép là đầu vào dữ liệu. Vì lẽ đó, hãng chế tạo IC thường gọi đây là bộ phân kênh giải mã – kiêm cả hai chức năng.

IC 74LS138 (mạch giải mã 3 sang 8) có ký hiệu cho trên hình 4.4.2.6 được dùng như bộ phân kênh với đầu vào $\overline{E_1}$ được chọn làm đầu vào dữ liệu (D); 2 đầu vào cho phép còn lại ($\overline{E_2}, E_3$) duy trì ở trạng thái tích cực. Mã địa chỉ là các đầu vào $A_0A_1A_2$.



Hình 4.4.2.5. Sơ đồ logic DMUX 1 : 8



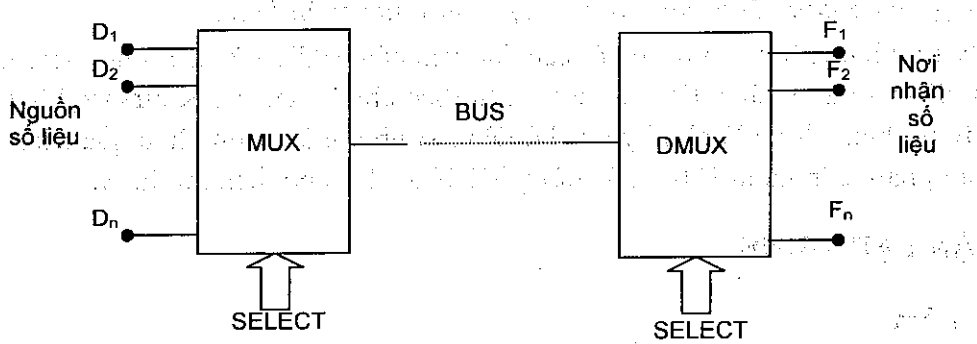
Hình 4.4.2.6. Ký hiệu IC74LS138

4.4.3. Ứng dụng

Các bộ phân kênh và hợp kênh có rất nhiều ứng dụng, một số ứng dụng sẽ được đề cập đến ở phần sau, ở đây ta nêu ra một số ứng dụng cụ thể của bộ phân kênh.

– Ứng dụng hợp kênh và phân kênh trong sơ đồ chọn và truyền số liệu theo mô hình cho trên hình 4.4.3.1.

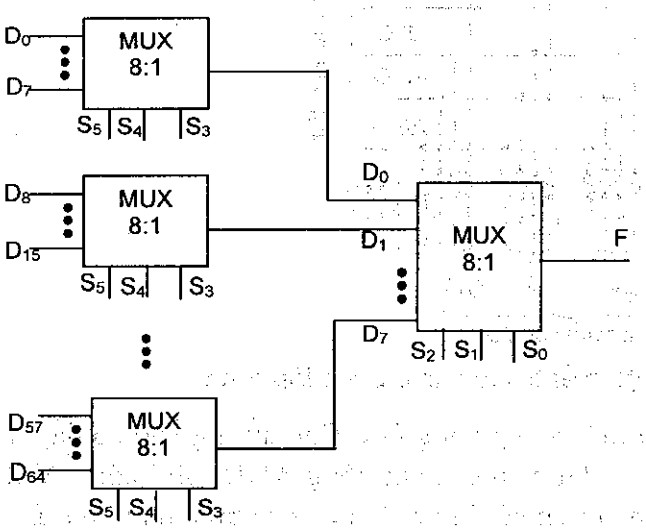
Bộ hợp kênh sẽ chọn một trong số các số liệu của nguồn số liệu D_1, D_2,... ,D_n đưa lên BUS để truyền đi. Ở đầu kia của BUS, nơi nhận số liệu, bộ phân kênh sẽ điều khiển số liệu đến nơi nhận xác định nào đó.



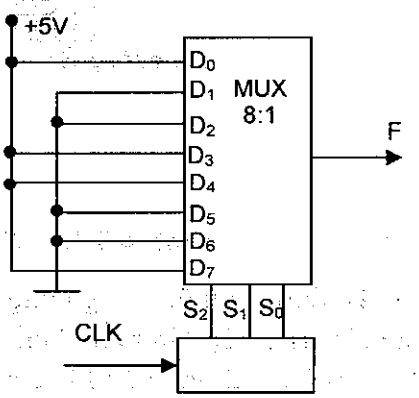
Hình 4.4.3.1. Ứng dụng hợp kênh và phân kênh trong sơ đồ chọn và truyền số liệu

Số các đường dữ liệu vào có thể tăng lên nhờ sử dụng các MUX kết nối với nhau và đầu ra cũng thế.

Ta có thể xây dựng MUX 8:1 từ các MUX 4:1 hoặc MUX 2:1, xây dựng MUX 16:1 từ các MUX có ít đầu vào hơn, v.v... Chẳng hạn, ta có thể xây dựng MUX 64:1 từ 9 MUX 8:1 như trên hình 4.4.3.2.



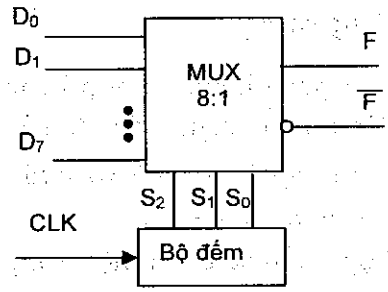
Hình 4.4.3.2. Xây dựng MUX64:1 từ các MUX8 : 1



Hình 4.4.3.4. Tạo dãy tín hiệu tuần

– Ứng dụng mạch hợp kênh biến đổi dạng thông tin vào song song thành dạng thông tin nối tiếp ở đầu ra:

Tín hiệu điều khiển của mạch hợp kênh được lấy từ đầu ra của bộ đếm, như vậy thời gian tồn tại của một bit tại đầu ra chính bằng chu kỳ của CLK. Hình 4.4.3.3 minh họa quá trình biến đổi thông tin vào song song 8 bit thành thông tin nối tiếp ở đầu ra.



Hình 4.4.3.3. Thông tin song song thành nối tiếp

– Ứng dụng mạch hợp kênh tạo dãy tín hiệu nhị phân tuần hoàn:

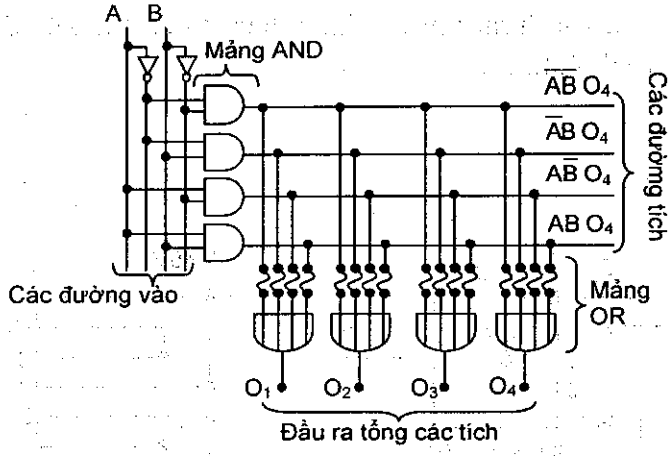
Để tạo dãy tín hiệu tuần hoàn ta sẽ nối các đầu vào của MUX với các mức logic nhất định. Tín hiệu đầu vào được đưa đến đầu ra tuần tự theo chu kỳ của CLK đưa tới bộ đếm, chu kỳ của dãy tín hiệu chính là chu kỳ của bộ đếm. Hình 4.4.3.4 minh họa quá trình tạo dãy tín hiệu nhị phân tuần hoàn 10011001 dùng MUX 8 : 1 và bộ đếm modul 8.

4.5. MA TRẬN LẬP TRÌNH

4.5.1. Khái niệm

Với mục đích làm giảm bớt số IC trong thiết kế, các nhà sản xuất đã cho ra đời các thiết bị logic cho phép lập trình (Programmable Logic Divice – PLD). PLD là IC chứa rất nhiều cổng, trigơ, thanh ghi được nối với nhau trên chip.

– Ý tưởng cơ sở được trình bày trên hình 4.5.1.1.



Hình 4.5.1.1. Ví dụ về một thiết bị logic cho phép lập trình

Ta thấy có một mảng cổng AND và một mảng cổng OR. Các đường vào có đầy đủ cả biến trực tiếp lẫn biến đảo, nó chính là đầu vào của các cổng AND. Mỗi cổng AND được nối với 2 đầu vào khác nhau tạo nên tích số duy nhất của các biến đầu vào. Đầu ra của các cổng AND được gọi là các đường tích số.

Mỗi đường tích số được nối với 1 trong 4 đầu vào của cổng OR thông qua một

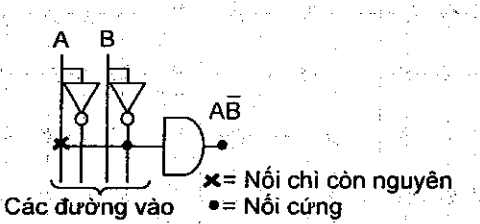
cầu chì (mối nối chì). Ban đầu, khi mọi mối nối chì còn nguyên vẹn, mỗi đầu ra OR sẽ là logic "1" không đổi (ví dụ: $O_1 = \overline{A}\overline{B} + \overline{A}B + A\overline{B} + AB = 1$).

Mỗi đầu ra trong 4 đầu $O_1 \div O_4$ đều có thể lập trình theo bất kỳ hàm nào của A và B bằng cách đốt đứt có lựa chọn các mối nối phù hợp. Một đầu vào OR bị đốt đứt sẽ tương ứng với mức logic "0". Ví dụ, nếu đốt đứt mối nối chì 1 và 4 tại cổng OR số 1 thì đầu ra O_1 sẽ là: $O_1 = \overline{A}B + A\overline{B}$.

Ta có thể lập trình mỗi đầu ra OR theo hàm tuỳ ý. Một khi cả 4 đầu ra đã được lập trình, thiết bị sẽ thường xuyên tạo hàm ở đầu ra được chọn.

– Ký hiệu PLD: Ví dụ minh hoạ ở hình 4.5.1.1 chỉ có hai biến đầu vào thế mà sơ đồ mạch đã khá là rối rắm. Nếu PLD có nhiều đầu vào hơn thì sơ đồ sẽ rất phức tạp. Vì lý do này, các nhà sản xuất PLD đã chấp nhận hệ thống ký hiệu đơn giản bớt để biểu diễn mạch điện bên trong các linh kiện này.

Hình 4.5.1.2 minh hoạ một ví dụ về hệ thống ký hiệu cho cổng AND có 4 đầu vào. Ở đây một đường đi vào cổng AND tượng trưng cho 4 đầu vào, các mối nối từ các đường biến đầu vào đến cổng AND được biểu diễn bằng dấu x hay chấm tròn. Dấu x minh hoạ mối nối chì còn nguyên, chấm tròn là mối nối cứng (hard – wired connection: mối nối không thay đổi được). Những nơi không có hai dấu nối này cho biết ở đó không tồn tại nối kết. Ví dụ ở đây, các đầu vào A và $\overline{B}$ được nối với cổng AND để tạo tích $A\overline{B}$. Các đầu vào $\overline{A}$ và B không được nối với cổng AND (không có dấu x hay chấm tròn), vì vậy chúng không tác động đến đầu ra. Điều quan trọng cần hiểu trong ký hiệu logic là cổng AND có 4 đầu vào khác nhau, mặc dù trong hình chỉ biểu diễn 1 đường duy nhất. Những đầu vào trong thực tế nối với cổng AND có ký hiệu là dấu x và/hoặc dấu chấm.



Hình 4.5.1.2. Ký hiệu AND 4 đầu vào

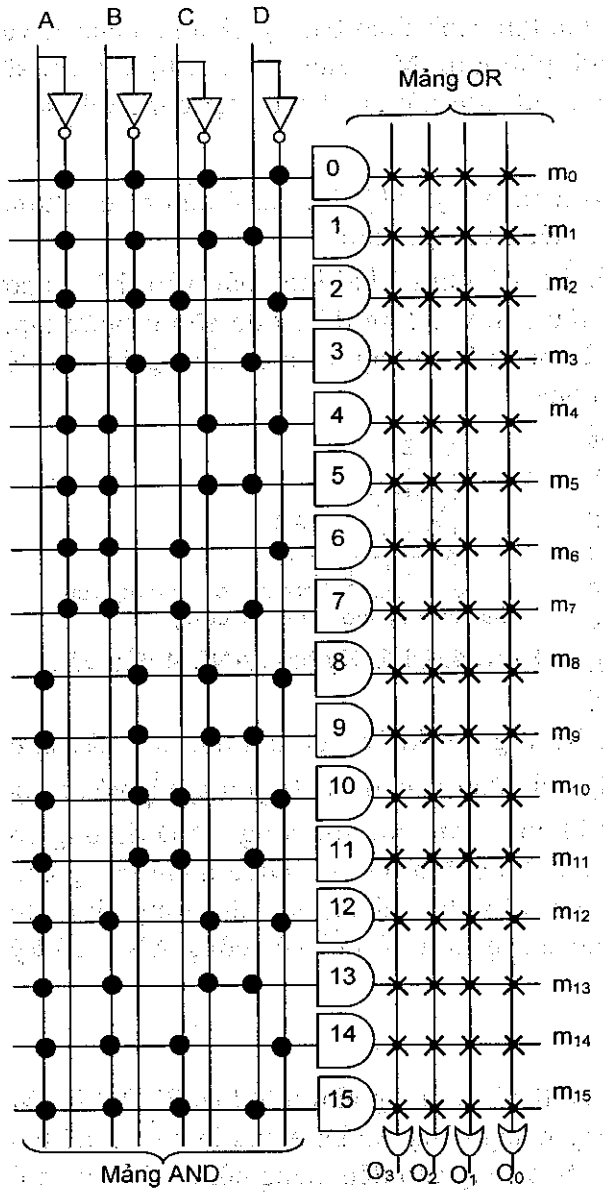
4.5.2. Cấu trúc PROM

Hình 4.5.2.1 minh hoạ một PROM 16×4, có thể hoạt động như một loại PLD. PROM này có 4 đầu vào được giải mã hoàn toàn bằng mảng cổng AND; nghĩa là mỗi cổng tạo ra một trong 16 tích AND có thể. Mối nối từ các đường vào đến mảng cổng AND là mối nối cứng, còn các mối nối từ các đường tích số đầu ra AND đến các đầu vào OR (số đầu vào của cổng OR bằng số đường tích số đầu ra AND) đều cho phép lập trình.

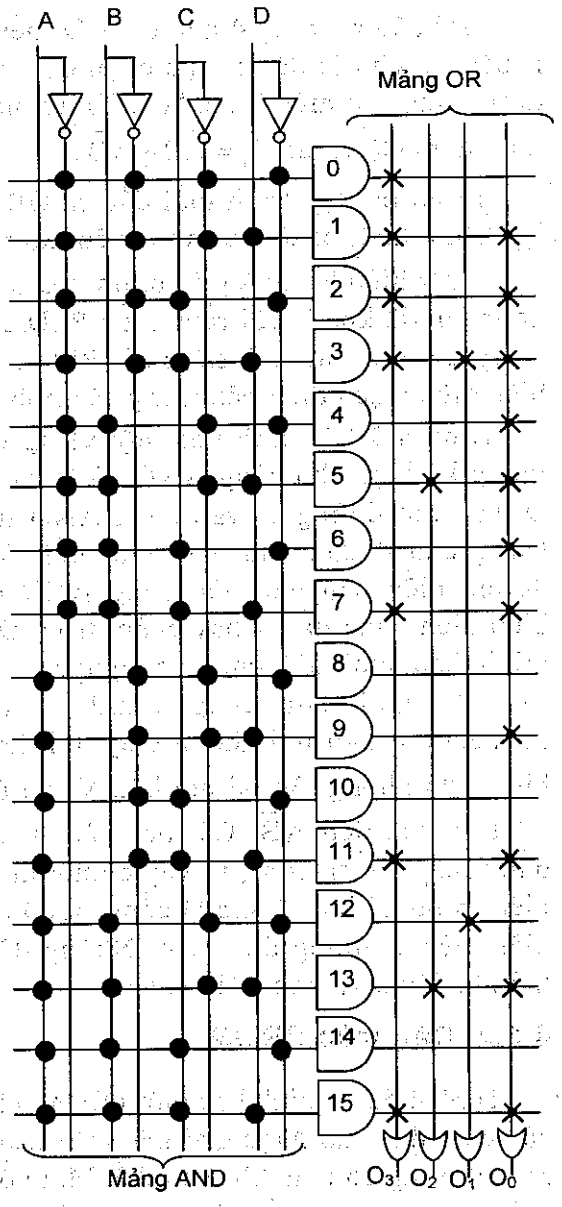
Hình 4.5.2.2 minh hoạ cách lập trình PROM để tạo 4 hàm logic xác định. Ví dụ, theo sơ đồ thì ta có $O_3 = \overline{A}\overline{B} + CD$.

PROM có thể tạo ra bất kỳ hàm logic nào từ các biến đầu vào, bởi vì nó tạo được mọi số hạng tích AND có thể có. Tuy nhiên, PROM trở nên không thiết thực khi cần đến số lượng lớn biến đầu vào, vì số nối chì sẽ nhân đôi mỗi lần thêm vào một biến. AM27S13 là một PROM thực sự được dùng như PLD, có dung lượng 512×4, sản xuất theo công nghệ Schottky TTL tốc độ cao. Vì $512 = 2^9$, nên PROM này có 9 đầu vào

địa chỉ và 4 đầu ra dữ liệu. Do vậy có thể lập trình AM27S13 để tạo 4 đầu ra, mỗi đầu ra là bất cứ hàm logic nào của 9 biến đầu vào khác nhau.



Hình 4.5.2.1. PROM 16 x 4

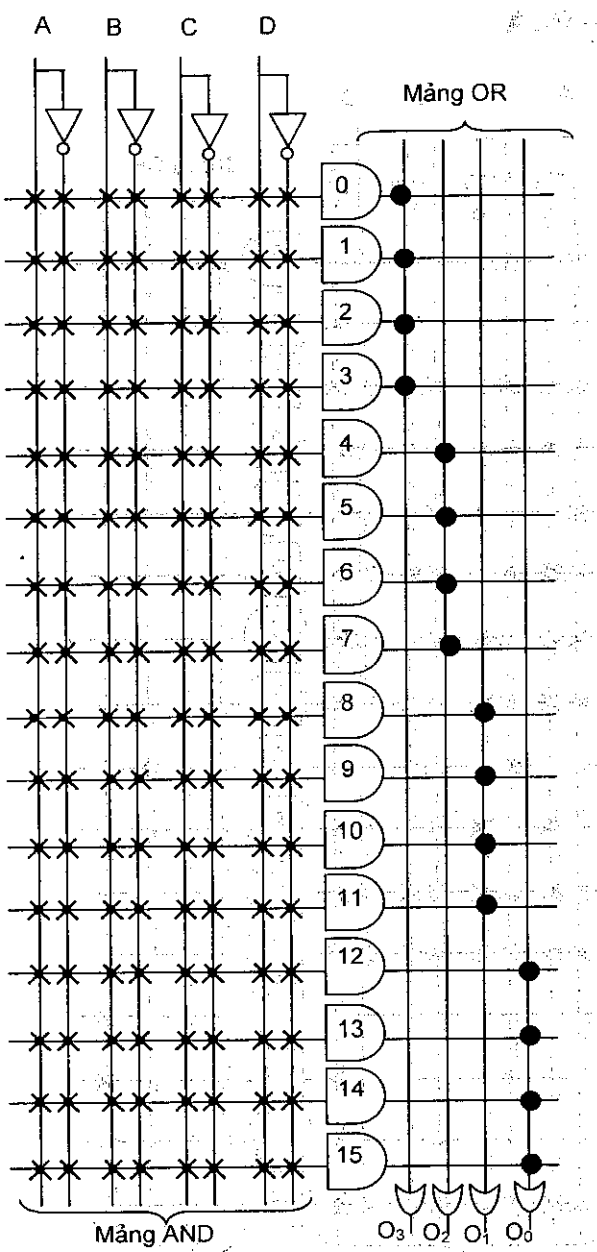


Hình 4.5.2.2. Lập trình PROM

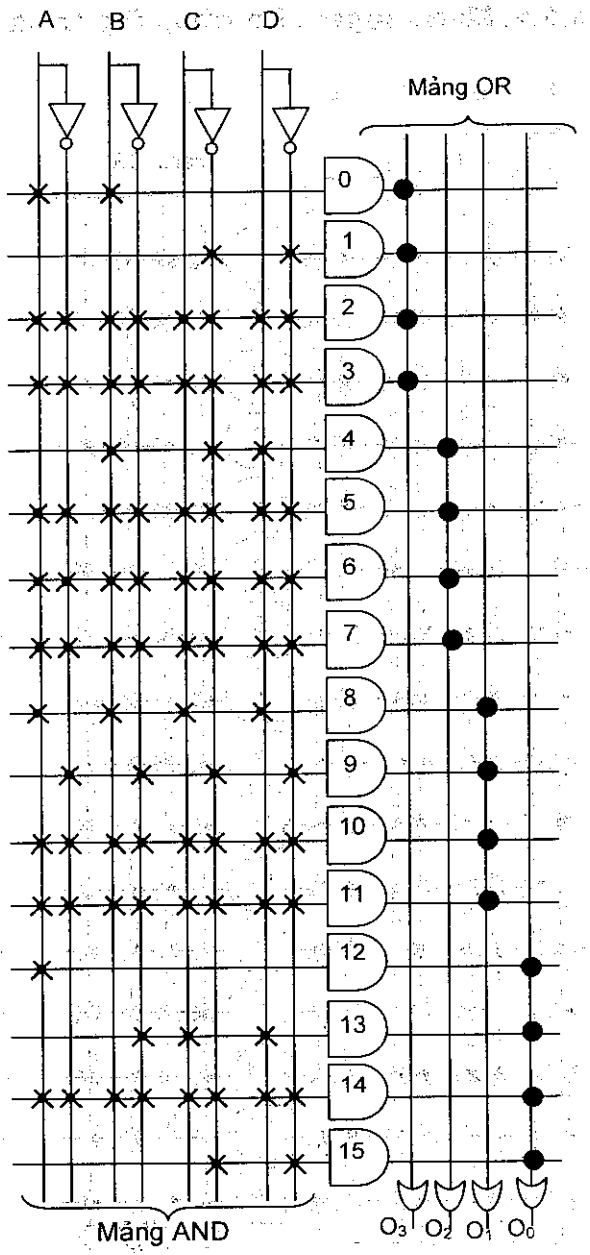
4.5.3. Logic mảng cho phép lập trình – PAL

Cấu trúc PROM rất phù hợp cho các ứng dụng đòi hỏi mỗi tổ hợp có thể của đầu vào để tạo hàm ở đầu ra. Ví dụ như bộ biến đổi mã và bảng lưu trữ dữ liệu. Tuy nhiên, ứng dụng không yêu cầu tất cả tổ hợp đầu vào phải được lập trình. Ví dụ, không phải mọi hàm logic minh hoạ trong hình 4.5.2.2 đều sử dụng tất cả các đường tích AND có sẵn. Điều này dẫn đến sự ra đời một lớp PLD được gọi là logic mảng cho phép lập trình (Programmable Array Logic – PAL). Sự hơi khác nhau về cấu trúc giữa PAL và PROM được minh hoạ ở hình 4.5.3.1.

PAL có cùng mảng cổng AND và OR với PROM, nhưng trong PAL, các đầu vào của cổng AND đều cho phép lập trình, còn đầu vào cổng OR được nối cứng. Điều này có nghĩa mọi cổng AND đều có thể lập trình được để cho ra bất kỳ tích số mong muốn nào của 4 biến đầu vào với các phần bù của chúng. Mỗi cổng OR chỉ được nối cứng với 4 đầu ra AND, giới hạn mỗi hàm chỉ có 4 tích. Nếu hàm nào yêu cầu hơn 4 tích, không thể ứng dụng nó với PAL này; mà phải dùng PAL có nhiều đầu vào OR hơn. Còn như không cần đến 4 tích, các cổng không cần thiết có thể biến thành 0.



Hình 4.5.3.1. Cấu trúc PAL

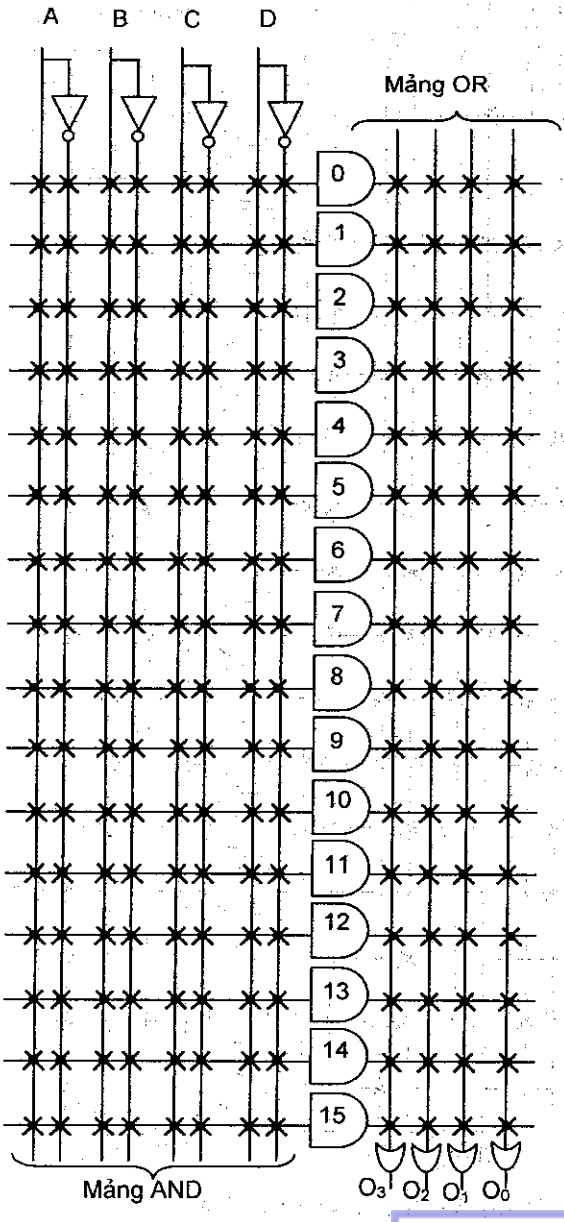


Hình 4.5.3.2. Lập trình PAL

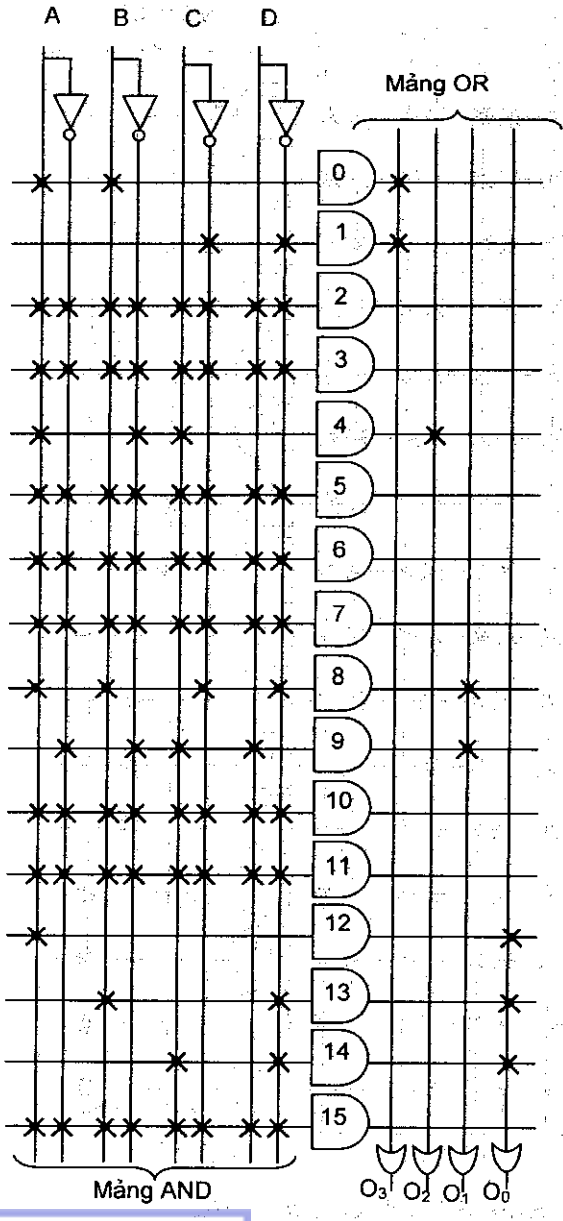
Hình 4.5.3.2 mô tả cách lập trình PAL để cho ra 4 hàm logic chỉ định, chẳng hạn ta có $O_3 = AB + \overline{CD}$. Khi cần tạo tích bằng 0 thì các cổng AND được giữ nguyên mọi mối nối chỉ ở đầu vào.

Ví dụ về mạch tích hợp PAL trong thực tế là loại PAL18L8A của công ty Texas Instrument,... được chế tạo theo công nghệ Schottky tiêu thụ ít năng lượng, có 10 đầu vào logic và 8 hàm đầu ra. Mỗi đầu ra cổng OR được nối cứng với 7 đầu ra cổng AND, vì vậy nó có thể tạo hàm có đến 7 số hạng.

4.5.4. Mảng logic cho phép lập trình – PLA



Hình 4.5.4.1. Cấu trúc PLA



Hình 4.5.4.2. Lập trình PLA

Sự khác nhau giữa PAL và PLA là PLA cho phép lập trình với cả ma trận AND và ma trận OR. Do đó, PLA có thể lợi dụng các tích số chung cho nhiều đầu ra. Một thiết bị PLA sẽ chậm hơn so với PAL do PAL có ma trận OR được nối cứng. Cấu trúc điển hình của PLA như ở hình 4.5.4.1. Hình 4.5.4.2, PLA đã được lập trình để có các đầu ra: $O_3 = AB + \overline{CD}$; $O_2 = A\overline{BC}$; $O_1 = AB\overline{CD} + \overline{AB}CD$; $O_0 = A + B\overline{D} + C\overline{D}$

4.6. THIẾT KẾ DÙNG VI MẠCH MSI, LSI

4.6.1. Thiết kế dùng MUX

MUX được sử dụng như một phần tử vạn năng để thiết kế mọi hàm logic.

Tổng quát: Một MUX 2^n:1 có thể dùng tạo hàm logic bất kỳ có n+1 biến, trong đó n biến sẽ đưa vào n đầu vào điều khiển, còn một biến cùng với các hằng số 0, 1 được đưa vào 2^n đầu vào dữ liệu tuỳ thuộc vào giá trị của hàm số đó.

Để thực hiện một hàm logic dùng MUX cho trước, có thể dùng bảng Karnaugh hoặc biến đổi trực tiếp.

a) Trường hợp biến đổi trực tiếp tiến hành theo các bước sau

1) Xác định n biến điều khiển của MUX.

2) Biểu diễn hàm số ban đầu ở dạng tổng các tích, mỗi tích có chứa một tổ hợp của n biến điều khiển.

3) Đưa tổ hợp của n biến điều khiển ra làm thừa số chung, ký hiệu D_i là hàm của những biến còn lại (i là giá trị thập phân của tổ hợp biến điều khiển).

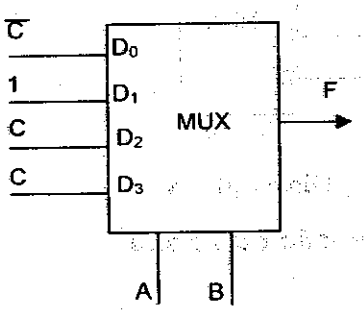
4) Tối thiểu hoá các hàm D_i ($i = 0 \div 2^n - 1$), D_i chính là giá trị đầu vào D_i của MUX đã cho.

5) Nếu D_i là hàm một biến hay hằng 0, hằng 1 thì bài toán đã giải xong. Ngược lại, tiếp tục dùng MUX hoặc các cổng logic để thực hiện hàm D_i như yêu cầu của đề bài.

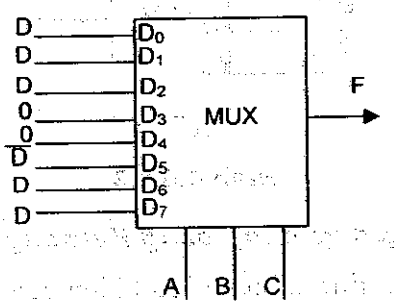
Ví dụ 4.6.1.1. Dùng MUX 4:1 để thực hiện hàm logic có 3 biến A, B, C sau:

$F(A,B,C) = \Sigma(0, 2, 3, 5, 7)$ với các biến điều khiển là A, B.

Giải: Hai biến điều khiển của MUX là A, B.



Hình 4.6.1.1. Sơ đồ (ví dụ 4.6.1.1)



Hình 4.6.1.2. Sơ đồ (ví dụ 4.6.1.2)

$$F = \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}\overline{C} + ABC$$

$$= \overline{A}\overline{B}\overline{C} + \overline{A}B(\overline{C} + C) + A\overline{B}\overline{C} + ABC = \overline{A}\overline{B}D_0 + \overline{A}BD_1 + A\overline{B}D_2 + ABD_3$$

Vậy ta có: $D_0 = \overline{C}$; $D_1 = 1$; $D_2 = C$; $D_3 = C$

Sơ đồ thực hiện hàm được mô tả trên hình 4.6.1.1.

Ví dụ 4.6.1.2. Dùng MUX 8:1 để thực hiện hàm logic có 4 biến A, B, C, D sau:

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

Giải: Chọn ba biến điều khiển của MUX 8:1 là A, B, C.

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

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

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

$$= \overline{A}\overline{B}\,\overline{C}D_0 + \overline{A}\overline{B}CD_1 + \overline{A}B\overline{C}D_2 + A\overline{B}CD_5 + AB\overline{C}D_6 + ABCD_7$$

Vậy: $D_0 = D$; $D_1 = D$; $D_2 = D$; $D_3 = 0$; $D_4 = 0$; $D_5 = \overline{D}$; $D_6 = D$; $D_7 = D$.

Sơ đồ thực hiện hàm được mô tả trên hình 4.6.1.2.

Ví dụ 4.6.1.3. Dùng MUX 4:1 để thực hiện hàm $F = AB + \overline{A}C + (\overline{B} + \overline{C})D$

Giải: Chọn A, B làm 2 biến điều khiển cho MUX 4:1.

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

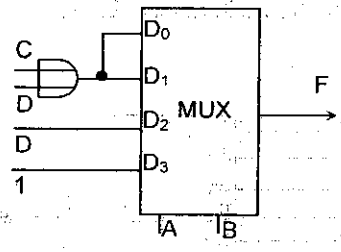
$$= AB + \overline{A}BC + \overline{A}\,\overline{B}\,C + ABD + \overline{A}BD + A\overline{B}D + \overline{A}\overline{B}D$$

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

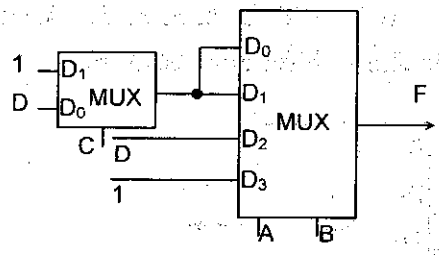
$$= \overline{A}\overline{B}D_0 + \overline{A}BD_1 + A\overline{B}D_2 + ABD_3$$

Vậy $D_0 = C + D$; $D_1 = C + D$; $D_2 = D$; $D_3 = 1$.

Các hàm D_0, D_1 lại có thể thực hiện trực tiếp dùng cổng logic (hình 4.6.1.3) hoặc dùng MUX 2:1 (hình 4.6.1.4).



Hình 4.6.1.3



Hình 4.6.1.4

b) Trường hợp dùng bảng Karnaugh tiến hành theo các bước sau

1) Xác định n biến điều khiển của MUX.

2) Tương ứng với các tổ hợp cụ thể của n biến điều khiển khoanh 2^n vùng khác nhau trên bảng Karnaugh, đánh dấu các vùng này là D_0, D_1,..., D_{2^n-1} (vùng D_i là vùng

ứng với giá trị thập phân của n biến điều khiển là i).

3) Điền giá trị của hàm cho trước vào bảng Karnaugh.

4) Tối thiểu hoá các hàm trong từng vùng D_i (không xét đến những biến điều khiển), gọi hàm số này là D_i. D_i chính là giá trị đầu vào tại D_i của MUX đã cho.

5) Nếu D_i là các hàm 1 biến hoặc hằng 0, hằng 1 thì bài toán đã giải xong. Trong trường hợp ngược lại phải tiếp tục dùng MUX hoặc các cổng logic để thực hiện hàm D_i như yêu cầu của đề bài.

Ví dụ 4.6.1.4. Dùng MUX 4:1 để thực hiện hàm logic có 3 biến A, B, C sau:

$F = \Sigma(0, 2, 3, 5, 7)$

a) Với các biến điều khiển là A, B.

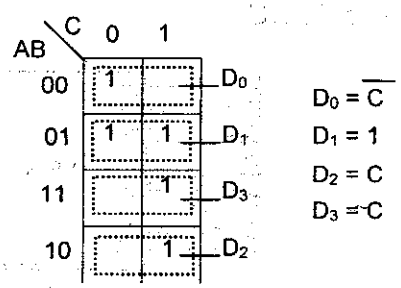
b) Với các biến điều khiển là B, C.

c) Với các biến điều khiển là A, C.

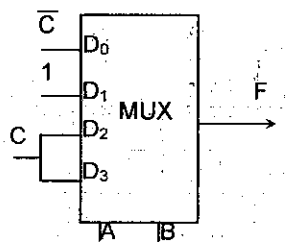
Giải:

a) Hai biến điều khiển của MUX là A, B

Tương ứng với các giá trị của A, B các vùng D_0, D_1, D_2, D_3 được biểu diễn trên bảng Karnaugh cho ở hình 4.6.1.5. D_0 ứng với AB = 00; D_1 ứng với AB = 01; D_2 ứng với AB = 10; D_3 ứng với AB = 11. Sơ đồ ở hình 4.6.1.6.



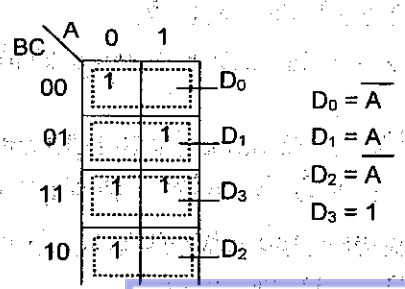
Hình 4.6.1.5



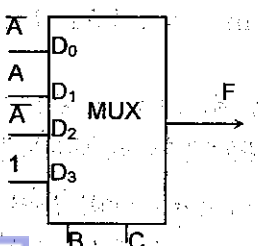
Hình 4.6.1.6

b) Hai biến điều khiển của MUX là B, C

Tương ứng với các giá trị của B, C các vùng D_0, D_1, D_2, D_3 được biểu diễn trên bảng Karnaugh cho ở hình 4.6.1.7. D_0 ứng với BC = 00; D_1 ứng với BC = 01; D_2 ứng với BC = 10; D_3 ứng với BC = 11. Sơ đồ ở hình 4.6.1.8.



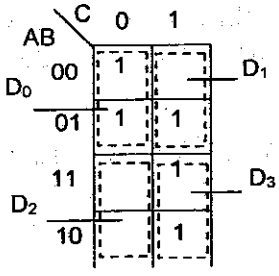
Hình 4.6.1.7



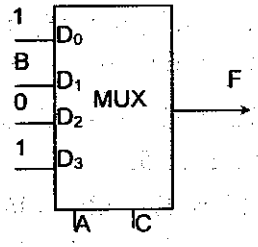
Hình 4.6.1.8

c) Hai biến điều khiển của MUX là A, C.

Tương ứng với các giá trị của A, C các vùng D_0, D_1, D_2, D_3 được biểu diễn trên bảng Karnaugh cho ở hình 4.6.1.9. D_0 ứng với AC = 00; D_1 ứng với AC = 01; D_2 ứng với AC = 10; D_3 ứng với AC = 11. Sơ đồ ở hình 4.6.1.10.



$D_0 = 1$
$D_1 = B$
$D_2 = 0$
$D_3 = 1$



Hình 4.6.1.9 **Hình 4.6.1.10**

Ví dụ 4.6.1.5. Thực hiện hàm logic 4 biến: $F(A,B,C,D) = \sum(1,2,5,7,8,10,13,15)$

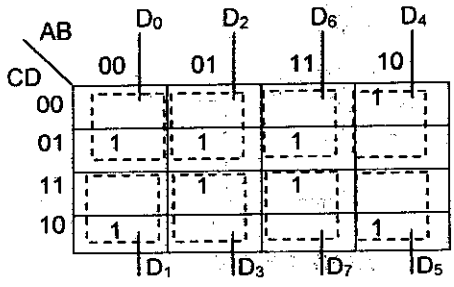
a) Dùng MUX 8:1

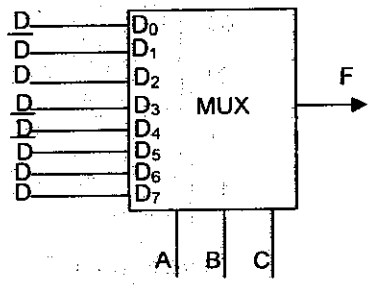
b) Dùng MUX 4: 1 và MUX 2:1

c) Dùng MUX 4:1 và cổng NAND.

Giải:

a) Chọn ba biến điều khiển của MUX 8:1 là A,B,C.





Hình 4.6.1.11 **Hình 4.6.1.12**

Từ bảng Karnaugh trên hình 4.6.1.11, ta có:

$$D_0 = D_2 = D_3 = D_6 = D_7 = D;\ D_1 = D_4 = D_5 = \overline{D}.$$

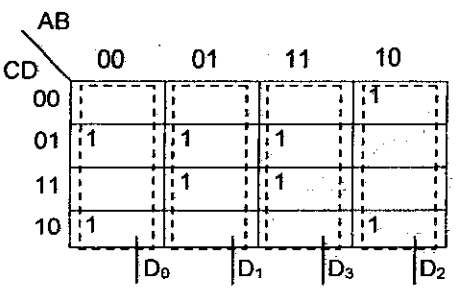
Sơ đồ tạo hàm logic dùng MUX 8:1 như ở hình 4.6.1.12.

b) Chọn hai biến điều khiển của MUX 4:1 là A,B; biến điều khiển của MUX 2:1 là C.

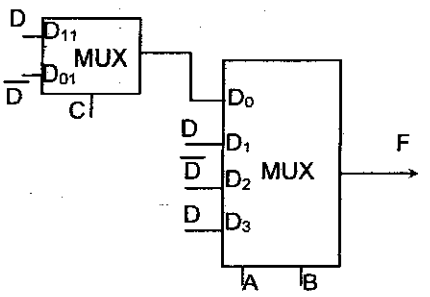
Xác định các đầu vào dữ liệu của MUX 4:1 từ bảng Karnaugh hình 4.6.1.13, ta có các đầu vào dữ liệu của MUX 8:1: $D_0 = \overline{C}D + C\overline{D}$; $D_1 = D_3 = D$; $D_2 = \overline{D}$.

Do D_0 vẫn là hàm hai biến nên ta phải dùng thêm MUX 2:1, với biến địa chỉ là C ta có các đầu vào dữ liệu của MUX 2:1: $D_{01} = D$; $D_{11} = \overline{D}$.

Sơ đồ tạo hàm logic dùng MUX 4:1 và MUX 2:1 như ở hình 4.6.1.14.



Hình 4.6.1.13

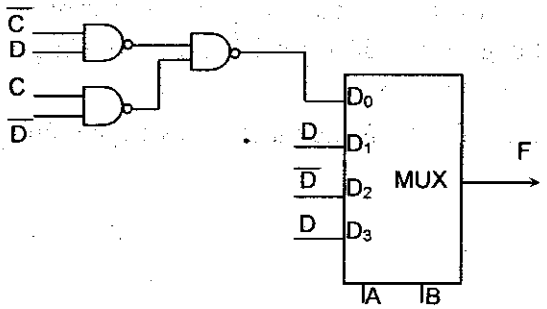


Hình 4.6.1.14

c) Với kết quả đã tìm được ở câu b, ta thay MUX 2: 1 bằng các cổng NAND, ta có:

$$D_0 = \bar{C}D + C\bar{D} = \overline{\overline{\bar{C}D}.\overline{C\bar{D}}}$$

Sơ đồ tạo hàm logic dùng MUX 4:1 và cổng NAND như ở hình 4.6.1.15.



Hình 4.6.1.15

4.6.2. Thiết kế dùng DMUX, DECODER

a) Dùng DMUX

DMUX khi đầu vào D = 1 là bộ giải mã của các tín hiệu vào địa chỉ, các đầu ra của nó tương ứng là các tích gồm đầy đủ các biến điều khiển. Do vậy để thực hiện một hàm logic cho trước, chỉ cần xác định bảng chân lý của hàm rồi từ đó dùng thêm các mạch phụ OR, AND, NAND hoặc NOR để xây dựng sơ đồ.

Ví dụ 4.6.2.1. Dùng DMUX 1:16 và các mạch NAND thiết kế mạch logic tạo hàm sau:

$$F(A, B, C, D) = \Sigma\,(5, 6, 7, 10, 11, 13, 14, 15)$$

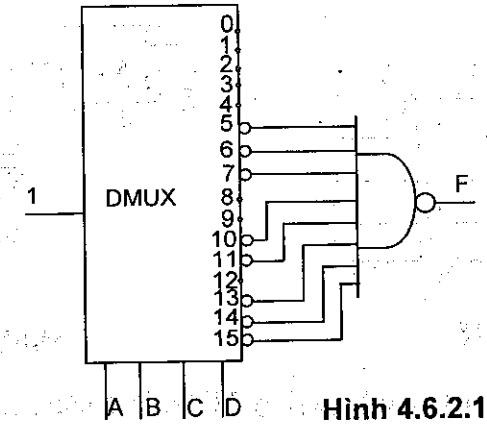
Giải:

$$F = \bar{A}\bar{B}CD + \bar{A}B\bar{C}D + \bar{A}BCD + A\bar{B}\bar{C}D + A\bar{B}CD + AB\bar{C}D + ABC\bar{D} + ABCD$$

$$F = \overline{\overline{\bar{A}\bar{B}CD} + \overline{\bar{A}B\bar{C}D} + \overline{\bar{A}BCD} + \overline{A\bar{B}\bar{C}D} + \overline{A\bar{B}CD} + \overline{AB\bar{C}D} + \overline{ABC\bar{D}} + \overline{ABCD}}$$

$$F = \overline{\bar{A}\bar{B}CD}\ \overline{\bar{A}B\bar{C}D}\ \overline{\bar{A}BCD}\ \overline{A\bar{B}\bar{C}D}\ \overline{A\bar{B}CD}\ \overline{AB\bar{C}D}\ \overline{ABC\bar{D}}\ \overline{ABCD}$$

Sơ đồ mạch thực hiện được biểu diễn trên hình 4.6.2.1, trong đó sử dụng mạch NAND 8 đầu vào.



Hình 4.6.2.1

b) Dùng DECODER

Như trên đã đề cập đến, bộ DMUX khi có đầu vào dữ liệu bằng 1 thì nó làm việc giống như một bộ giải mã. Mỗi đầu ra của bộ giải mã là một tích đầy đủ của tất cả các biến đầu vào, khi dùng bộ giải mã để thiết kế mạch ta cũng thực hiện tương tự như đối với DMUX ta đã xét ở phần trên.

Ví dụ 4.6.2.2. Dùng bộ giải mã 3 – 8 và các mạch NOR thực hiện các hàm 3 biến sau:

a) $F_1 = AB + \overline{A}\,\overline{B}\,\overline{C}$

b) $F_2 = A + B + \overline{C}$

c) $F_3 = \overline{A}B + A\overline{B}$

Giải: Giả sử quy ước trọng số cho ba biến A, B, C là:

A: 2^2; B: 2^1 ; C: 2^0

Ta có: $F_1 = m_0 + m_6 + m_7$

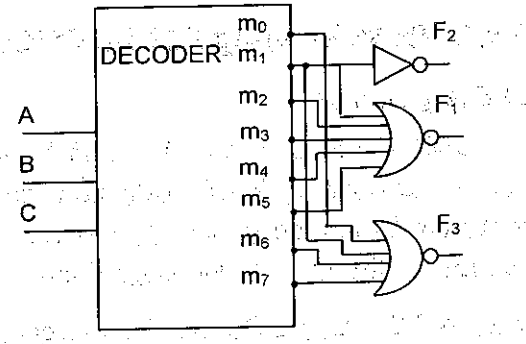
hay $\overline{F_1} = m_1 + m_2 + m_3 + m_4 + m_5$

Do đó: $F_1 = \overline{\overline{F_1}} = \overline{m_1 + m_2 + m_3 + m_4 + m_5}$

Tương tự : $F_2 = \overline{\overline{F_2}} = \overline{\overline{A}\overline{B}C} = \overline{m_1}$

$F_3 = \overline{m_0 + m_1 + m_6 + m_7}$

Sơ đồ mạch thực hiện được biểu diễn trên hình 4.6.2.2.



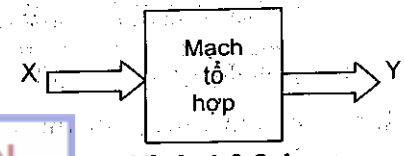
Hình 4.6.2.2

4.6.3.Thiết kế dùng ROM, PLA

a) Dùng ROM

ROM dùng làm bộ nhớ trong máy tính, ngoài ra ROM còn được dùng để thiết kế các mạch số.

Dùng ROM để thiết kế các mạch tổ hợp, mạch tổ hợp có sơ đồ khối như hình 4.6.3.1.



Hình 4.6.3.1

Khi dùng ROM để thiết kế mạch tổ hợp ta coi X như địa chỉ vào của ROM và Y như các đầu ra số liệu của ROM.

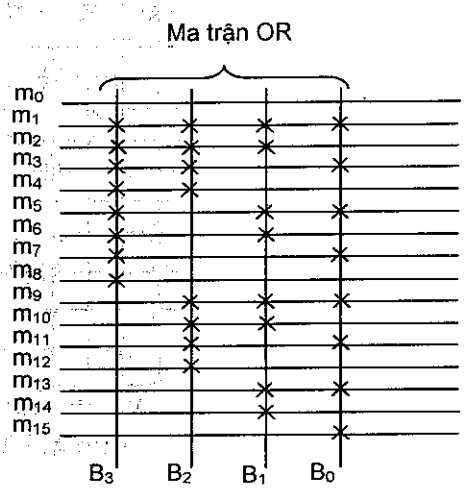
Ví dụ 4.6.3.1. Dùng ROM để xây dựng bộ chuyển đổi mã nhị phân 4 bit sang mã bù hai.

Giải: Bảng trạng thái của bộ chuyển đổi mã nhị phân 4 bit sang mã bù hai trên hình 4.6.3.2a.

ROM có ma trận AND được nối cứng do đó với 4 biến đầu vào A_3, A_2, A_1, A_0 được xem là 4 đầu vào địa chỉ của ROM sẽ tạo ra được 16 tích đầy đủ của 4 biến đầu vào. Ma trận OR cho phép lập trình do đó các nối chì bị đốt đứt để lập trình các đầu ra theo yêu cầu của bài toán như trên hình 4.6.3.2b.

Số thập phân	Mã nhị phân				Mã bù 2			
	A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	1	1	1	1
2	0	0	1	0	1	1	1	0
3	0	0	1	1	1	1	0	1
4	0	1	0	0	1	1	0	0
5	0	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	0
7	0	1	1	1	1	0	0	1
8	1	0	0	0	1	0	0	0
9	1	0	0	1	0	1	1	1
10	1	0	1	0	0	1	1	0
11	1	0	1	1	0	1	0	1
12	1	1	0	0	0	1	0	0
13	1	1	0	1	0	0	1	1
14	1	1	1	0	0	0	1	0
15	1	1	1	1	0	0	0	1

a)



b)

Hình 4.6.3.2. Bảng trạng thái và ma trận OR cho phép lập trình

b) Dùng PLA

Ví dụ 4.6.3.2. Dùng PLA để thực hiện các hàm sau:

$F_1 = ABC\overline{D} + \overline{A}D + \overline{AB}\,C$; $F_2 = AB + BC\overline{D} + \overline{AB}$

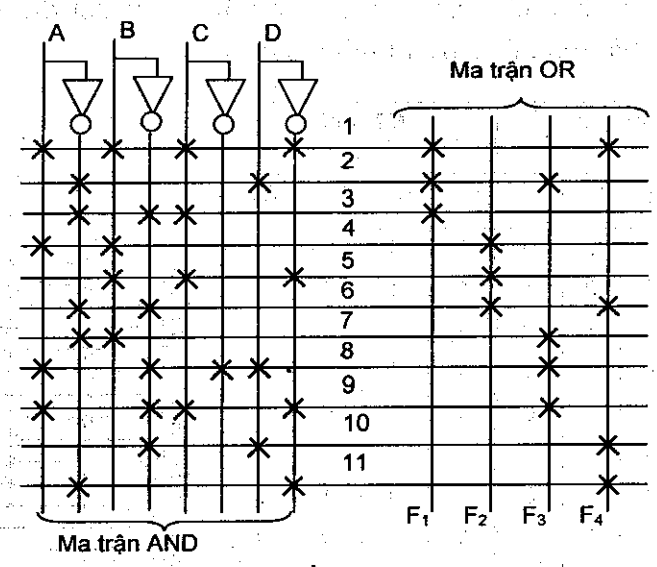
(1) (2) (3) (4) (5) (6)

$F_3 = \overline{A}\,B + \overline{A}D + A\overline{B}CD + A\overline{B}C\overline{D}$; $F_4 = ABC\overline{D} + \overline{AB} + \overline{B}D + \overline{AD}$

(7) (2) (8) (9) (1) (6) (10) (11)

Giải: Các hàm trên được biểu diễn ở dạng chuẩn tắc tuyển. Đánh dấu các tích của 4 hàm, trong đó các tích giống nhau được đánh dấu bằng cùng một con số. Tất cả có 11 tích khác nhau. Các tích này hoặc là đầy đủ cả 4 biến, hoặc là không đủ cả 4 biến. Như vậy, ma trận AND phải có 8 đầu vào tương ứng với 4 biến và 11 đầu ra tương ứng với 11 tích.

Ma trận OR phải có 11 đầu vào tương ứng với 11 tích và 4 đầu ra tương ứng với 4 hàm số F_1, F_2, F_3, F_4. Sơ đồ PLA thực hiện các hàm trên được biểu diễn trên hình 4.6.3.3.



Hình 4.6.3.3. Sơ đồ PLA thực hiện hàm

BÀI TẬP CHƯƠNG 4

Bài 4.1. Thiết kế mạch kiểm tra mã BCD, nếu mã BCD là hợp lệ thì đầu ra ở mức logic thấp, ngược lại đầu ra ở mức logic cao.

Bài 4.2. Thiết kế bộ so sánh hai số nhị phân 1 bit.

a) Chỉ dùng mạch NAND 7400.

b) Chỉ dùng mạch NOR 7402.

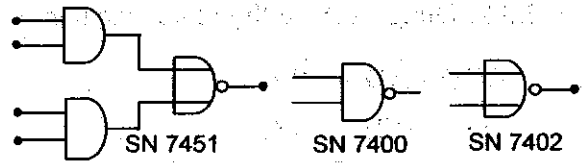
Bài 4.3. Cho mạch NORAND 7451, mạch NAND 7400 và mạch NOR 7402 hình 4.3BT.

Hãy thiết kế bộ bán tổng:

a) Chỉ dùng SN 7451 và SN 7400

b) Chỉ dùng SN 7400

c) Chỉ dùng SN 7402.



Hình 4.3BT

Bài 4.4. Xây dựng mạch tổng toàn phần chỉ dùng mạch NOR 7402.

(Gợi ý:sơ đồ tối ưu dùng 9 cổng NOR).

Bài 4.5. Dùng 7486 (4 mạch hoặc tuyệt đối hai đầu vào) và mạch 7400 (4 mạch NAND 2 đầu vào) tạo mạch tổng toàn phần.

Bài 4.6. Xây dựng mạch tổng toàn phần chỉ dùng mạch NAND 7400.

(Gợi ý: sơ đồ tối ưu dùng 9 cổng NAND)

Bài 4.7. Dùng mạch cộng hai số nhị phân 4 bit và mạch AND, thiết kế mạch nhân 2 số nhị phân 4 bit.

Bài 4.8. Thiết kế mạch biến đổi mã nhị phân 4 bit sang mã Gray chỉ dùng IC 74LS86 (4 mạch XOR 2 lối vào), từ đó thiết kế mạch biến đổi mã Gray sang mã nhị phân 4 bit.

Bài 4.9. Thiết kế mạch biến đổi mã bù hai 4 bit sang mã nhị phân 4 bit.

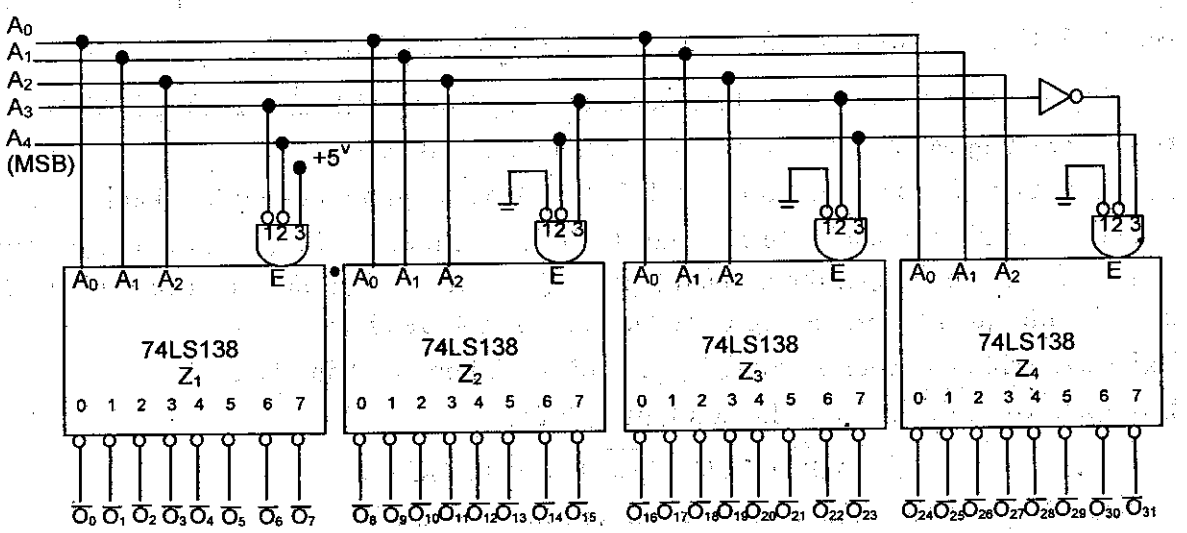
Bài 4.10. Thiết kế mạch biến đổi mã BCD8421 sang mã dư 3.

Bài 4.11. Dùng mạch NAND 74LS00 (NAND 2 lối vào) tạo thành bộ biến đổi mã nhị phân 4 bit sang mã bù nhị phân 4 bit.

Bài 4.12. Hình 4.12BT cho thấy bộ 74LS138 và một bộ đảo được sắp xếp để hoạt động như bộ giải mã 1 trong 32. Các bộ giải mã lần lượt mang nhãn từ Z_1 đến Z_4 và 8 đầu ra mỗi bộ giải mã được kết hợp thành 32 đầu ra. Mã đầu vào 5 bit A_4 A_3 A_2 A_1 A_0 sẽ kích hoạt chỉ một trong 32 đầu ra này cho mỗi đầu vào trong 32 mã đầu vào khả dĩ.

a) Đầu ra nào sẽ được kích hoạt với A_4 A_3 A_2 A_1 A_0 = 01101?

b) Khoảng mã đầu vào nào sẽ kích hoạt chíp Z_4?



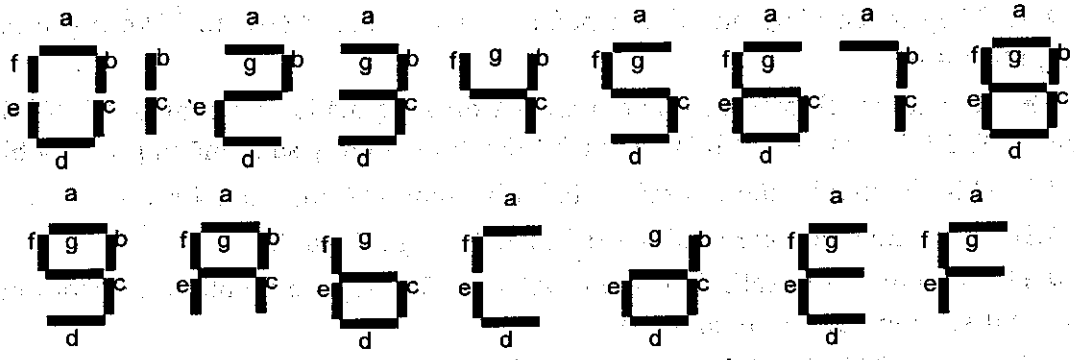
Hình 4.12BT. Bốn bộ giải mã 74LS138 tạo thành bộ giải mã 1 trong 32.

Bài 4.13. Vẽ sơ đồ logic của bộ giải mã từ BCD sang thập phân dùng các mạch logic NOT và NOR.

Bài 4.14. Từ bảng chân lý của bộ giải mã BCD sang 7 đoạn được cho ở trên hãy viết phương trình logic và xây dựng sơ đồ logic cho bộ giải mã này từ các cổng logic cơ bản.

Gợi ý: Tận dụng 6 tổ hợp không được sử dụng trong mã BCD (1010,...,1111) để tối thiểu hóa hàm.

Bài 4.15. Vẽ sơ đồ logic bộ giải mã 7 đoạn điều khiển sự hiển thị của các đèn chỉ thị số như hình 4.15BT (đầu ra tích cực ở mức cao).



Hình 4.15BT. Sự hiển thị các chữ số của IC MC 14495

Bài 4.16. Thiết kế mạch logic tổ hợp thực hiện hệ hàm sau:

$F_1(A,B,C,D) = \sum 1,4,5,6,9,11,12)$

$F_2(A,B,C,D) = \sum 4,5,6,8,11,12)$

$F_3(A,B,C,D) = \sum 0,3,8,10,11,12)$

Bài 4.17. Cho hàm logic 4 biến $F(A,B,C,D) = \sum 0,1,2,4,5,6,9,11,12)$

Xây dựng sơ đồ mạch thực hiện hàm này tầng 1 dùng OR, tầng 2 dùng NAND.

Bài 4.18. Dùng MUX 2:1 và MUX 4:1 để tạo thành MUX 8:1.

Bài 4.19. Sắp xếp nhiều bộ hợp kênh 8:1 (IC 74151) để tạo thành bộ hợp kênh 16:1.

Bài 4.20. Trình bày cách sắp xếp hai IC 74157 và một IC 74151 tạo thành bộ hợp kênh 16:1 mà không cần thêm mạch logic. Đặt tên từ D_0 đến D_{15} cho các đầu vào để biểu thị mối quan hệ tương ứng giữa đầu vào với mã lựa chọn.

Bài 4.21. Trình bày cách sử dụng bộ giải mã 7442 làm bộ phân kênh 1:8.

Bài 4.22. MUX 4:1 có lối ra là hàm logic 3 biến:

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

Hai lối vào điều khiển là A và B, tìm các lối vào dữ liệu.

Bài 4.23. Dùng MUX 8:1 để thực hiện hàm logic có 4 biến A, B, C, D sau:

a) F(A,B,C,D) = Σ(0, 2, 3, 5, 7, 10,12,15) với các biến điều khiển là A, B, C.

b) F(A,B,C,D) = Σ(0, 2, 3, 5, 7, 10,12,15) với các biến điều khiển là A, C, D.

Bài 4.24. Dùng MUX 4:1 xây dựng sơ đồ bộ giải mã BCD sang 7 đoạn ở bài tập 4.15.

Bài 4.25. Thực hiện hàm logic 4 biến: $F(A, B, C, D) = \Sigma(0, 1, 5, 6, 7, 9, 10, 14, 15)$

a) Dùng MUX 8:1.

b) Dùng MUX 4:1 và MUX 2:1.

c) Dùng MUX 4:1 và mạch NAND

d) Dùng MUX 4:1 và mạch NOR

e) Dùng MUX 2:1.

Bài 4.26. Thực hiện hàm 5 biến sau:

$F(A,B,C,D,E) = \Pi(2, 4, 5, 9, 10, 11, 12, 13, 16, 19, 23, 25, 26, 29, 30)$

a) Dùng MUX 16:1.

b) Dùng MUX 8:1 và MUX 2:1.

c) Dùng MUX 8:1 và mạch NAND

d) Dùng MUX 8:1 và mạch NOR

e) Dùng MUX 4:1.

Bài 4.27. Cho các hàm:

$F_1 = AB\overline{C} + \overline{A}BC + \overline{A}\overline{B}$; $F_2 = \overline{A}\overline{B}\overline{C} + A\overline{B}C + \overline{A}B\overline{C} + ABC$

Dùng DMUX và mạch OR thiết kế mạch có lối ra F_1, F_2.

Bài 4.28. Thiết kế mạch thực hiện hệ 4 phương trình sau:

$$F_1 = \overline{A}\overline{B}C + ABC; \quad F_3 = A + B + \overline{C}D + \overline{A}D$$

$$F_2 = \overline{A} + \overline{B} + C + D; \quad F_4 = ACD + A\overline{C}\overline{D} + B\overline{C}D + BC\overline{D}$$

dùng mạch NAND và bộ giải mã 4 – 16.

Bài 4.29. Dùng ROM để xây dựng bộ giải mã nhị phân BCD sang mã 7 đoạn (hiển thị 10 chữ số và 6 chữ cái ở bài tập 4.16).

Bài 4.30.Dùng PLA để thực hiện các hàm sau:

$$F_1 = ABC\overline{D}E\overline{F} + AEF + AB\overline{C} + BD$$

$$F_2 = AB\overline{E}F + AB\overline{C}D + AE\overline{F} + BD$$

$$F_3 = ACDE\overline{F} + AEF + ABCD + \overline{C}D$$

$$F_4 = AB\overline{E}F + BCD + ACDE\overline{F} + \overline{C}\overline{D}$$

Chương 5

CÁC MẠCH LOGIC DÃY

5.1. CÁC TRIGƠ SỐ

5.1.1. Định nghĩa và phân loại

a) Định nghĩa

Trigơ trong tiếng Anh gọi là Flip – Flop viết tắt là FF. Nó là một phần tử nhớ có hai trạng thái cân bằng ổn định tương ứng với 2 mức logic 0 và 1. Dưới tác động của các tín hiệu điều khiển ở lối vào, trigơ có thể chuyển về một trong hai trạng thái cân bằng và giữ nguyên trạng thái đó chừng nào chưa có tín hiệu điều khiển làm thay đổi trạng thái của nó. Trạng thái tiếp theo của trigơ phụ thuộc không những vào tín hiệu ở lối vào mà còn phụ thuộc vào cả trạng thái đang hiện hành của nó.

Đang chạy, nếu ngừng các tín hiệu điều khiển ở lối vào nó vẫn có khả năng giữ trạng thái hiện hành của mình trong một thời gian dài, chừng nào mà nguồn điện nuôi mạch trigơ không bị ngắt thì thông tin dưới dạng nhị phân lưu giữ trong trigơ vẫn được duy trì. Như vậy, nó được sử dụng như một phần tử nhớ.

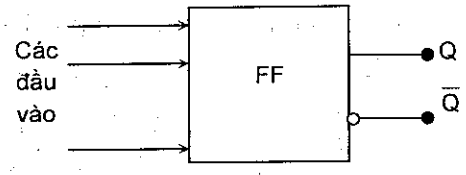
Trigơ được cấu thành từ 1 nhóm các cổng logic, mặc dù cổng logic tự thân nó không có khả năng lưu trữ, nhưng có thể nối nhiều cổng với nhau theo cách thức cho phép lưu giữ được thông tin. Mỗi sự sắp xếp cổng khác nhau sẽ cho ra các trigơ khác nhau.

Trigơ có nhiều đầu vào điều khiển và chỉ có hai đầu ra luôn ngược nhau là Q và $\overline{Q}$.

Sơ đồ khối tổng quát của một trigơ, hình 5.1.1.1:

Q: đầu ra thường; $\overline{Q}$: đầu ra đảo.

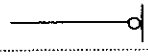
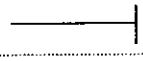
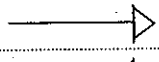
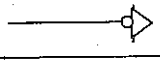
– Khi $Q=1$, $\overline{Q}=0$ ta nói FF ở trạng thái 1 hay trạng thái cao; trạng thái này còn được gọi là trạng thái Set (thiết lập).



Hình 5.1.1.1. Sơ đồ tổng quát của trigơ

– Khi $Q=0$, $\overline{Q}=1$ ta nói FF ở trạng thái 0 hay trạng thái thấp; trạng thái này còn gọi là trạng thái Reset (tái thiết lập hay xoá).

- *Các ký hiệu về tính tích cực của tín hiệu:*

Ký hiệu	Tính tích cực của tín hiệu
	Tích cực là mức thấp "L"
	Tích cực là mức cao "H"
	Tích cực là sườn dương của xung nhịp
	Tích cực là sườn âm của xung nhịp

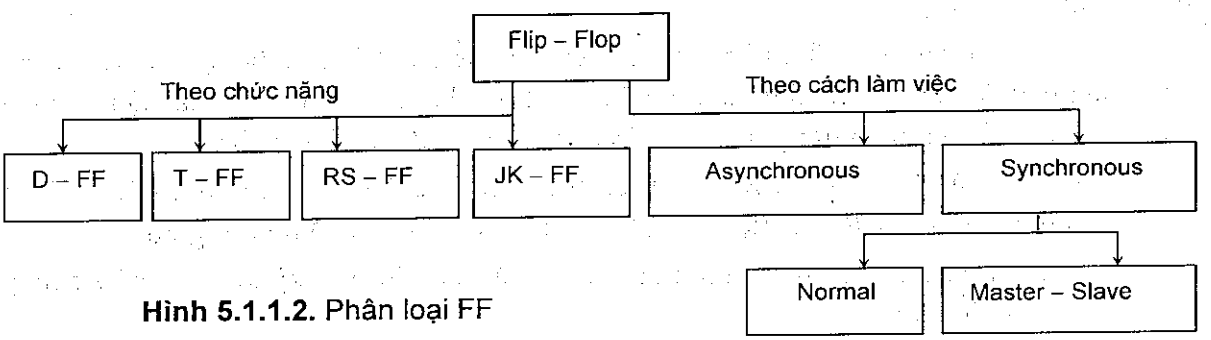
b) Phân loại trigơ

Có nhiều cách phân loại trigơ:

– Phân loại theo chức năng làm việc của các đầu vào điều khiển. Hiện nay, thường sử dụng loại trigơ 1 đầu vào (trigơ D, trigơ T) và loại 2 đầu vào (trigơ RS, trigơ JK), ngoài ra đôi khi còn có thể gặp loại trigơ nhiều đầu vào.

– Phân loại theo cách làm việc ta có loại trigơ không đồng bộ và đồng bộ. Loại đồng bộ lại được chia làm loại đồng bộ thường và loại đồng bộ chủ tớ.

• *Sơ đồ khối của sự phân loại trigơ được cho ở hình 5.1.1.2.*



Hình 5.1.1.2. Phân loại FF

c) Biểu diễn FF

Để mô tả 1 FF người ta có thể dùng:

– Bảng chân lý

– Đồ hình chuyển đổi trạng thái

– Phương trình đặc trưng.

5.1.2. Các loại trigơ và điều kiện đồng bộ

D	Q_n	Q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

R	S	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	x
1	1	1	x

J	K	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Hình 5.1.2.1. Bảng trạng thái của các trigơ D, T, RS, JK

Các trigơ đều có thể xây dựng từ các mạch tổ hợp có hồi tiếp. Ta biết rằng mạch có hồi tiếp chỉ có thể làm việc tin cậy khi điều kiện sau đây được thỏa mãn:

Mạch không rơi vào trạng thái dao động dưới tác động của bất kỳ tập hợp tín hiệu điều khiển vào nào. Điều này có nghĩa là, ứng với *mỗi tập hợp tín hiệu vào bất kỳ phải tồn tại ít nhất một trạng thái ổn định.* Trạng thái ổn định là trạng thái thỏa mãn

điều kiện $Q_{n+1} = Q_n$ (Q_n: trạng thái lối ra ở thời điểm n; Q_{n+1}: trạng thái lối ra ở thời điểm n + 1).

Theo chức năng có 4 loại FF cơ bản: D, T, RS, JK. Bảng trạng thái của các loại FF như trên hình 5.1.2.1.

Từ bảng chân lý trên ta rút ra nhận xét:

– Các D – FF và RS – FF có thể làm việc ở chế độ không đồng bộ vì với mỗi tập hợp tín hiệu vào điều khiển D – FF, RS – FF luôn luôn tồn tại ít nhất một trong các trạng thái ổn định. Bởi vì tất cả tập tín hiệu vào điều khiển D – FF, RS – FF đều có 1 trạng thái $Q_n = Q_{n+1}$.

– Các T – FF và JK – FF không thể làm việc ở chế độ không đồng bộ vì mạch sẽ rơi vào trạng thái dao động nếu như tập tín hiệu vào T = 1 hoặc JK = 11. Với các tập tín hiệu vào này không bao giờ có trạng thái $Q_n = Q_{n+1}$ (như đã in đậm ở bảng chân lý).

Như vậy, các D – FF và RS – FF có thể làm việc ở cả hai chế độ: đồng bộ và không đồng bộ còn T – FF và JK – FF chỉ có thể làm việc ở chế độ đồng bộ.

* *Chế độ không đồng bộ:* Trạng thái đầu ra sẽ thay đổi bất kỳ khi nào có sự thay đổi ở đầu vào điều khiển.

* *Chế độ đồng bộ:* Để khống chế sự thay đổi trạng thái ở đầu ra người ta đưa thêm vào FF 1 đầu vào xung nhịp (Clock). Chỉ khi nào có tác động của đầu vào xung nhịp thì FF mới thay đổi trạng thái theo đầu vào điều khiển. Xung nhịp thường là một chuỗi xung hình chữ nhật hoặc xung vuông.

Hầu hết hệ thống kỹ thuật số là đồng bộ, vì mạch đồng bộ dễ thiết kế và dễ dò lỗi hơn. Sở dĩ chúng dễ dò lỗi hơn là bởi vì đầu ra của mạch chỉ thay đổi ở những thời gian xác định.

5.1.3. Đầu vào bất đồng bộ

Đối với trigơ đồng bộ có đầu vào điều khiển và đầu vào xung nhịp. Các đầu vào điều khiển còn được gọi là đầu vào đồng bộ vì tác động của chúng lên đầu ra trigơ đồng bộ với đầu vào xung nhịp.

Hầu hết trigơ đồng bộ đều có một hoặc nhiều đầu vào bất đồng bộ là những đầu vào hoạt động độc lập với đầu vào đồng bộ và đầu vào xung nhịp. Đầu vào bất đồng bộ dùng để thiết lập FF ở trạng thái 1 hoặc xoá trigơ về trạng thái 0 bất kỳ thời điểm nào, bất chấp điều kiện các đầu vào còn lại.

Hai đầu vào bất đồng bộ Preset (thiết lập) và Clear (xoá) là những đầu vào tích cực ở mức thấp, Preset (Pr) thiết lập FF ở trạng thái 1 bất cứ lúc nào và Clear (CLR) xoá FF về trạng thái 0 vào bất cứ lúc nào.

Do đó, có thể sử dụng các đầu vào bất đồng bộ để giữ FF ở trạng thái cụ thể trong bất kỳ khoảng thời gian dự tính nào. Tuy nhiên, đầu vào bất đồng bộ rất thường được dùng để thiết lập hoặc xoá FF về trạng thái mong muốn bằng cách áp xung nhất thời.

5.1.4. Trigơ RS

Trigơ RS là 1 trigơ có hai đầu vào điều khiển R, S. S là đầu vào thiết lập "1" (Set) còn R là đầu vào xoá "0" (Reset).

Bảng trạng thái đầy đủ của RS – FF cho trên hình 5.1.4.1.

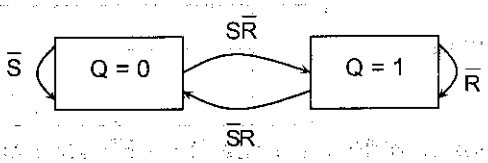
R	S	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	x
1	1	1	x

Hình 5.1.4.1. Bảng trạng thái của RS – FF

Bảng trạng thái rút gọn của RS – FF cho trên hình 5.1.4.2.

R	S	Q_{n+1}	Mốt hoạt động
0	0	Q_n	Nhớ
0	1	1	Thiết lập
1	0	0	Xoá
1	1	x	Cấm dùng

Hình 5.1.4.2. Bảng trạng thái rút gọn của RS – FF



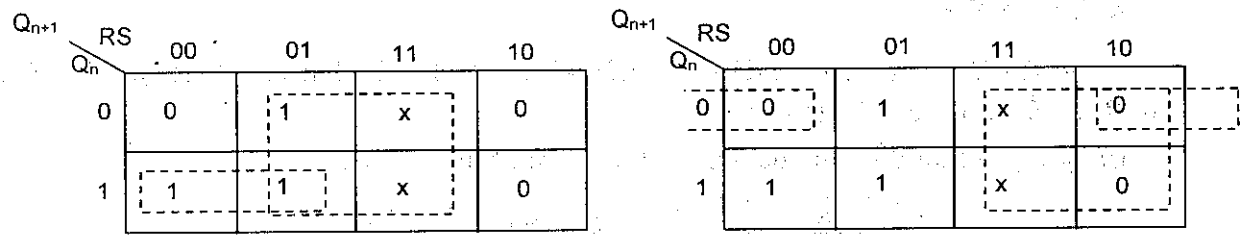
Hình 5.1.4.3. Đồ hình trạng thái của RS – FF

Trên bảng trạng thái Q_n chỉ trạng thái lối ra ở thời điểm hiện tại, Q_{n+1} chỉ trạng thái lối ra tại thời điểm tiếp theo.

Đồ hình chuyển đổi trạng thái được biểu diễn trên hình 5.1.4.3, nó biểu diễn trạng thái của các đầu vào điều khiển tương ứng với 4 trạng thái có thể ở đầu ra.

Chẳng hạn khi Q giữ nguyên giá trị 0 thì có hai khả năng xảy ra đó là trạng thái nhớ hoặc trạng thái xoá, với 2 trạng thái đó thì S luôn bằng 0 còn R bất kỳ nên chỉ cần ghi là $\overline{S}$. Hoặc là khi Q chuyển từ 0 sang 1 thì chỉ có duy nhất 1 khả năng là trạng thái thiết lập (S = 1; R = 0) nên ghi là $S\overline{R}$,....

Từ bảng trạng thái ta dùng bảng Karnaugh để tối thiểu hoá hàm như trên hình 5.1.4.4.



Hình 5.1.4.4. Tối thiểu hoá hàm theo hai cách

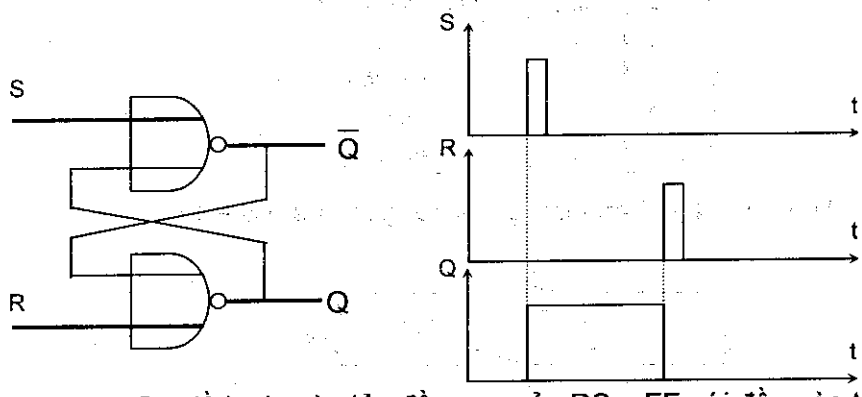
Phương trình của RS – FF: $Q_{n+1} = S + \overline{R}Q_n = \overline{R}(S + Q_n)$

Phương trình của RS – FF cho thấy: lối ra không những là hàm số của lối vào mà còn phụ thuộc vào trạng thái trước đó của lối ra.

Ta có thể xây dựng sơ đồ logic của trigơ RS từ mạch NOR, lối vào tích cực ở mức cao:

Từ phương trình dạng tích các tổng: $Q_{n+1} = \overline{R}(S + Q_n) = \overline{R + (S + Q_n)}$

Sơ đồ logic và giản đồ xung của RS – FF đầu vào tích cực cao cho trên hình 5.1.4.5.



Hình 5.1.4.5. Sơ đồ logic và giản đồ xung của RS – FF với đầu vào tích cực cao

Ta cũng có thể xây dựng trigơ RS không đồng bộ với đầu vào tích cực bởi mức logic thấp từ các phần tử NAND:

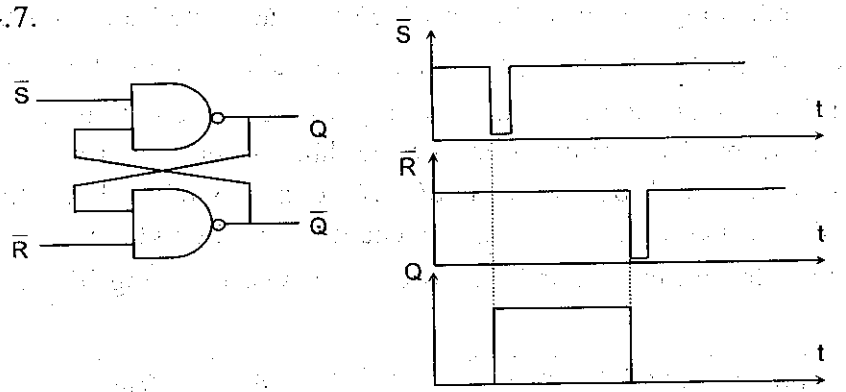
Bảng trạng thái của trigơ RS lối vào tác động bởi mức thấp cho trên hình 5.1.4.6.

$\overline{S}$	$\overline{R}$	Q_{n+1}	Mốt hoạt động
1	1	Q_n	Nhớ
1	0	0	Thiết lập
0	1	1	Xoá
0	0	x	Cấm dùng

Hình 5.1.4.6. Bảng trạng thái của RS – FF có đầu vào tích cực thấp

Từ phương trình: $Q_{n+1} = S + \overline{R}Q_n = \overline{\overline{S}.\overline{RQ_n}}$

Ta có sơ đồ logic và giản đồ xung của RS – FF đầu vào tích cực thấp cho trên hình 5.1.4.7.

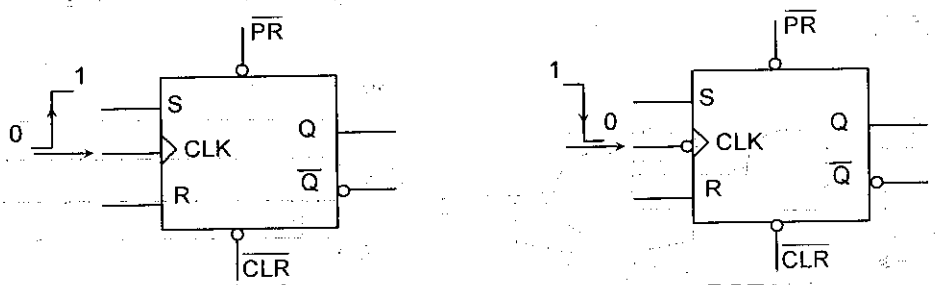


Hình 5.1.4.7. Sơ đồ logic và giản đồ xung của RS – FF đầu vào tích cực

Nhận xét: Từ giản đồ xung trên hình 5.1.4.5 và 5.1.4.7 ta thấy rằng bất cứ khi nào đầu vào của trigơ chuyển sang mức tích cực thì đầu ra sẽ thay đổi trạng thái. Điều này dẫn đến trạng thái ở đầu ra dễ bị ảnh hưởng bởi nhiễu tác động.

5.1.5. Trigơ RS đồng bộ

Trigơ RS không đồng bộ đầu ra sẽ thay đổi trạng thái bất kỳ thời điểm nào có sự tác động của đầu vào S hoặc R, vì thế trạng thái của trigơ sẽ không ổn định khi lối vào chịu ảnh hưởng của nhiễu. Để khắc phục nhược điểm trên người ta dùng trigơ RS đồng bộ (RST), nghĩa là thêm vào một đầu vào xung nhịp Clock (CLK, C_K) điều khiển chung cho cả hai lối vào. Chỉ khi nào có tác động của xung nhịp này thì trigơ mới chuyển trạng thái theo tác động của R hay S. Ký hiệu của trigơ RS đồng bộ cho trên hình 5.1.5.1.



Hình 5.1.5.1. Ký hiệu logic của trigơ RST

Sự chuyển trạng thái của trigơ RST và tất cả các loại trigơ đồng bộ khác xảy ra có thể vào thời điểm sau khi xung nhịp đã chuyển từ mức logic 0 lên mức logic 1 (sườn dương) hoặc sau khi xung nhịp đã chuyển từ mức logic 1 về mức logic 0 (sườn âm) như trên hình 5.1.5.1. Tuỳ theo cấu trúc cụ thể của từng loại trigơ, khi dùng ta cần chú ý đến ký hiệu của trigơ, nếu trên đó có ghi vòng tròn ở lối vào chân CLK hoặc trên chữ CLK có dấu gạch ngang ký hiệu hàm phủ định ($\overline{CLR}$) thì trạng thái lối ra của trigơ được xác lập khi xung chuyển từ mức logic 1 về mức logic 0.

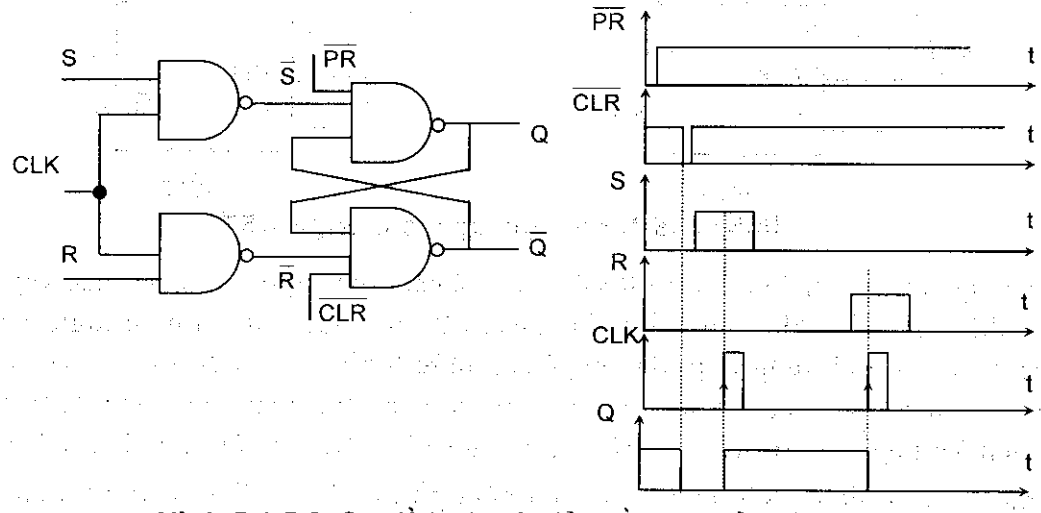
Trên ký hiệu của RST – FF ở hình 5.1.5.1 ta thấy còn có hai đầu vào $\overline{PR}$ và $\overline{CLR}$ đó là các đầu vào bất đồng bộ, có nghĩa là nó có thể tác động vào trigơ bất cứ thời điểm nào không cần quan tâm đến trạng thái của các đầu vào còn lại. $\overline{PR}$ là Preset (thiết lập) và $\overline{CLR}$ là Clear (xoá), hai đầu vào này ở mức tích cực thấp, ta có thể thiết lập hoặc xoá trigơ tại bất kỳ thời điểm nào từ các đầu vào này. Đầu vào R và S hoạt động đồng bộ với CLK nên được gọi là các đầu vào đồng bộ, đồng thời trạng thái đầu ra thay đổi theo R và S nên nó còn được gọi là các đầu vào điều khiển.

Bảng trạng thái đầy đủ của trigơ RST có các đầu vào bất đồng bộ cho trên hình 5.1.5.2.

$\overline{PR}$	$\overline{CLR}$	CLK	R	S	Q_{n+1}
0	1	x	x	x	1
1	0	x	x	x	0
1	1	0	x	x	Q_n
1	1	1	0	0	Q_n
1	1	1	0	1	1
1	1	1	1	0	0
1	1	1	1	1	x

Hình 5.1.5.2. Bảng trạng thái của RST – FF

Sơ đồ logic của trigơ RST và giản đồ xung diễn tả trạng thái hoạt động của trigơ được xác lập sau khi xung nhịp chuyển từ mức logic thấp lên mức logic cao cho trên hình 5.1.5.3.



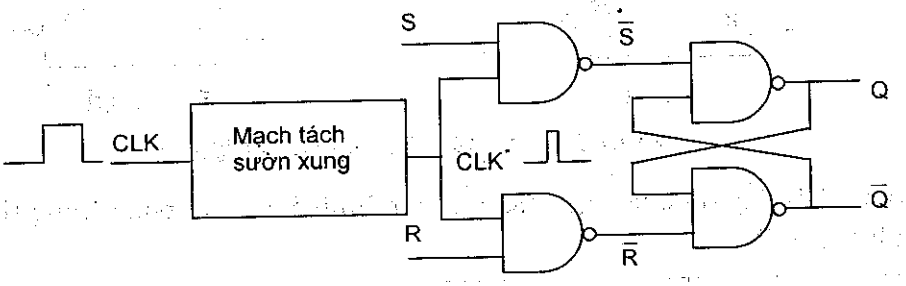
Hình 5.1.5.3. Sơ đồ logic và giản đồ xung của RST – FF

Nhận xét: Mặc dù RST – FF đã khắc phục được nhược điểm của RS – FF nhưng nó vẫn còn nhược điểm là trạng thái cấm khi R = S = 1.

**** Mạch tách sườn xung:***

Tất cả các loại FF đồng bộ đều khả dụng ở dạng IC. Mặc dù quan tâm chính của chúng ta là hoạt động bên ngoài của FF, nhưng để hiểu rõ hơn về hoạt động bên ngoài thì chúng ta cần phải xem xét mạch bên trong của FF.

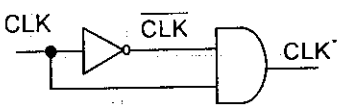
Ví dụ, sơ đồ minh hoạ trigơ RS kích bằng sườn trên hình 5.1.5.4.

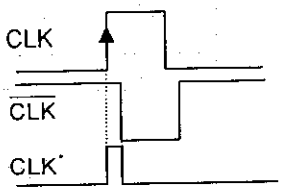


Hình 5.1.5.4. Trigơ RS kích bằng sườn

Mạch tách sườn xung sinh ra một xung nhọn hẹp đi lên (CLK^*) xảy ra đồng thời với việc chuyển trạng thái tích cực của xung đầu vào. Sơ đồ mạch tách sườn xung dùng trong FF kích bằng sườn trên hình 5.1.5.5 và hình 5.1.5.6.

– Mạch tách sườn dương:

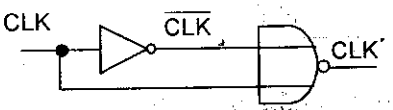


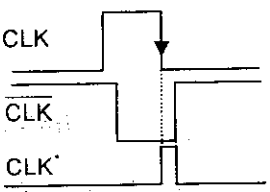


Thời gian xung CLK^* kéo dài từ 2 ÷ 5ns lúc cả CLK và $\overline{CLK}$ cùng cao (bằng thời gian trễ do truyền qua cổng NOT.)

Hình 5.1.5.5. Mạch tách sườn dương và giản đồ xung.

– Mạch tách sườn âm:





Thời gian xung CLK^* kéo dài từ 2 ÷ 5ns lúc cả CLK và $\overline{CLK}$ cùng thấp (bằng thời gian trễ do truyền qua cổng NOT).

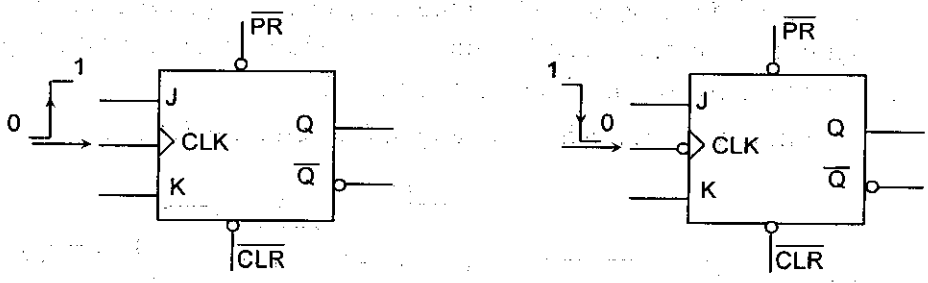
Hình 5.1.5.6. Mạch tách sườn âm và giản đồ xung.

Từ hoạt động của trigơ ta thấy FF có thể hoạt động như một mạch chốt (Latch). Khi có xung nhịp tác động FF hoạt động đúng như bảng trạng thái, khi không có xung nhịp tác động, đầu ra của FF giữ không đổi trạng thái, nghĩa là trạng thái FF bị chốt lại.

5.1.6. Trigơ JK

Trigơ RST có một tổ hợp biến cấm dùng là S = R = 1 trạng thái của trigơ này sẽ

không được xác định nếu gặp phải tổ hợp này. Ta có thể khắc phục tình trạng này bằng cách dùng hai mạch phản hồi từ Q về R và $\overline{Q}$ về S ta sẽ tạo được trigơ JK. Ký hiệu của trigơ JK cho trên hình 5.1.6.1.



Hình 5.1.6.1. Ký hiệu logic của trigơ JK

Bảng trạng thái đầy đủ của trigơ JK cho trên hình 5.1.6.2a, bảng trạng thái rút gọn cho trên hình 5.1.6.2b.

J	K	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

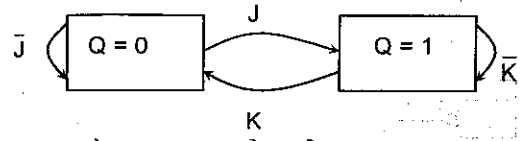
a)

t_n		t_{n+1}
J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q_n}$

b)

Hình 5.1.6.2. Bảng trạng thái đầy đủ và rút gọn của JK – FF

Đồ hình chuyển đổi trạng thái của JK – FF cho trên hình 5.1.6.3.



Hình 5.1.6.3. Đồ hình chuyển đổi trạng thái của JK – FF

Lúc này ta thấy, khi Q chuyển từ 0 sang 1 tương ứng với hai khả năng là trạng thái thiết lập và trạng thái lật nên J = 1 còn K là bất kỳ nên ghi là J. Khi Q chuyển từ 1 sang 0 tương ứng với 2 khả năng là trạng thái xoá và trạng thái lật nên K = 1 còn J là bất kỳ nên ghi là K. Hai trường hợp còn lại giống như trigơ RS.

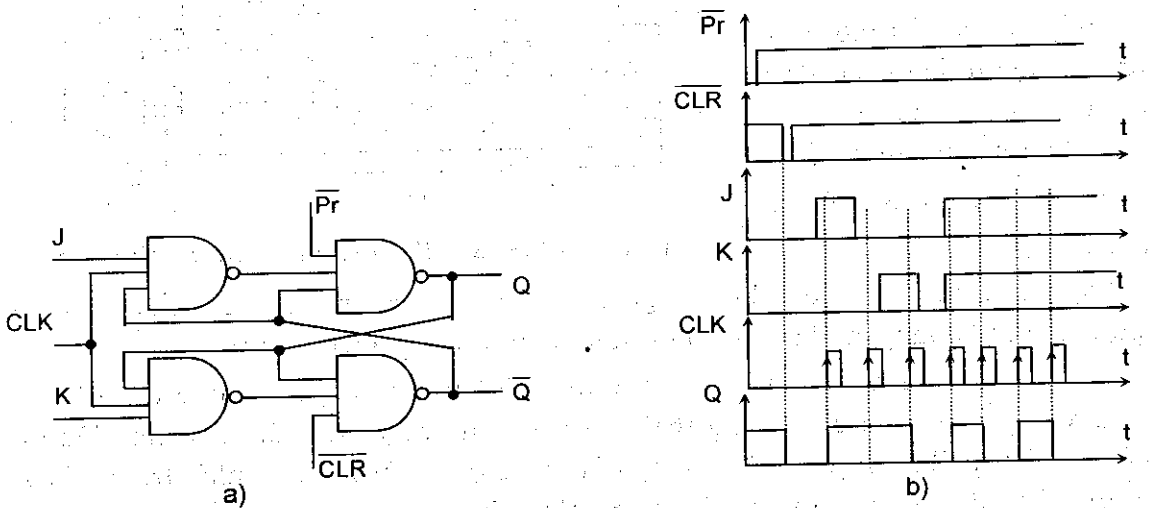
Trigơ JK là cải tiến của trigơ RS để khắc phục nhược điểm của trigơ RS, nên:

J tương ứng với S (thiết lập) và K tương ứng với R (xoá), nhưng khác với trigơ RS là trigơ JK không có trạng thái cấm, khi J = K = 1 lối ra lật trạng thái (Toggle).

Phương trình logic của trigơ JK: $Q_{n+1} = J_n \overline{Q_n} + \overline{K_n} Q_n$

Sơ đồ logic của trigơ JK cho trên hình 5.1.6.4a và giản đồ xung cho trên hình 5.1.6.4b mô tả các trạng thái hoạt động của trigơ này.

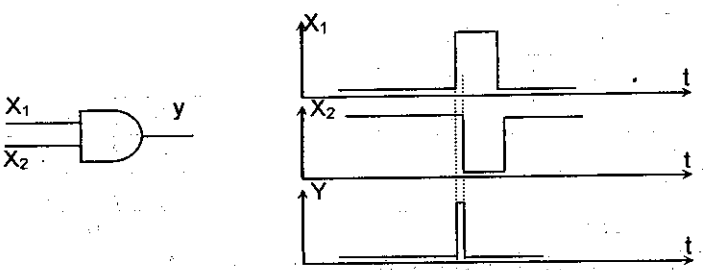
Các trigơ JK trong thực tế ngoài các lối vào J, K hoạt động đồng bộ với lối vào xung nhịp CLK, trigơ còn có các lối vào bất đồng bộ là Preset và Clear. Để trigơ hoạt động được ở chế độ đồng bộ, hai lối vào bất đồng bộ này phải để đúng mức điện áp, nếu trên ký hiệu của trigơ các lối vào bất đồng bộ có vòng tròn nhỏ hoặc dấu gạch ngang ở trên chữ ($\overline{Pr}, \overline{CLR}$) thì các chân này phải để ở mức cao.



Hình 5.1.6.4. Sơ đồ logic và giản đồ xung của trigơ JK

Khi J, K để ở mức cao cứ mỗi lần có xung nhịp tác động trigơ lại chuyển trạng thái một lần, sau hai xung nhịp tác động trigơ lại trở về trạng thái cũ (hình 5.1.6.4b), người ta dùng trường hợp này để tạo thành bộ đếm nhị phân từ các trigơ JK.

Trong các mạch logic tổ hợp có hiện tượng chạy đua vòng quanh (Race around) là sự xuất hiện tín hiệu giả (xung nhiễu) do quá trình quá độ khi hai lối vào chuyển trạng thái theo hai hướng ngược nhau nhưng sự chuyển mạch diễn ra ở hai chân không cùng một lúc, ở lối ra cửa logic xuất hiện xung kim. Sự tạo thành xung nhiễu ở các cửa logic trong hiện tượng chạy đua được minh hoạ trên hình 5.1.6.5.



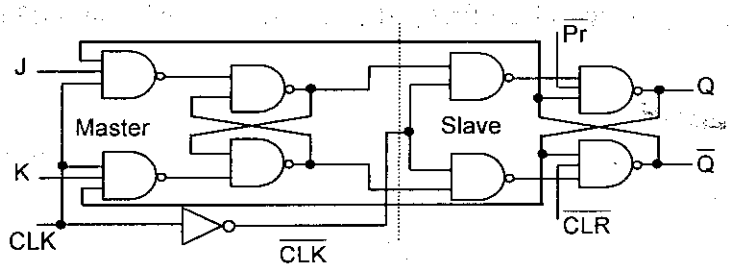
Hình 5.1.6.5. Sự hình thành xung nhiễu trong hiện tượng chạy đua

Do tác động vào X_2 bị trễ so với xung tác động vào X_1 nên có thời điểm cả X_1 và X_2 đều ở mức cao cho nên lối ra Y có xung kim nhiễu. Nếu tải của lối ra là các phần

tử nhạy như trigơ, xung nhiễu này có thể làm cho nó chuyển trạng thái không theo điều khiển. Vì vậy cần phải loại trừ hiện tượng chạy đua này. Hiện tượng này còn sinh ra do quá trình quá độ của một mạch tổ hợp nối tiếp nhiều phần tử logic làm cho sự trễ ở lối ra so với lối vào tăng dần lên. Để khắc phục hiện tượng chạy đua người ta dùng trigơ JK chủ – tớ.

Trigơ JK chủ - tớ (Master - Slave)

Sơ đồ logic và ký hiệu của trigơ JK master – slave cho trên hình 5.1.6.6. Nó được cấu tạo từ hai trigơ RST mắc nối tiếp nhau với hai mạch phản hồi từ lối ra Q và $\overline{Q}$ trở về các lối vào thiết lập và xoá.



Hình 5.1.6.6. Sơ đồ logic của trigơ JK chủ – tớ

Trigơ chủ (master) điều khiển trigơ tớ (slave), chỉ những thay đổi trạng thái của trigơ chủ mới là nguyên nhân thay đổi trạng thái lối ra của trigơ slave.

Ví dụ, khi xung nhịp chuyển từ mức logic 0 lên mức logic 1 thông tin ở lối vào JK được nạp vào trigơ chủ, trạng thái của trigơ chủ được xác lập theo tín hiệu điều khiển ở lối vào JK. Trigơ chủ chỉ thay đổi trạng thái một lần duy nhất trong khoảng thời gian kéo dài của xung nhịp. Khi xung nhịp chuyển từ mức logic 1 về mức logic 0 ($\overline{CLK}$ chuyển từ 0 lên 1) trigơ chủ ở trong trạng thái nhớ, trigơ tớ sao chép lại trạng thái của trigơ chủ. Bởi vì trong thời gian trigơ tớ xác lập trạng thái thì trigơ chủ ở trạng thái nhớ, nên các thay đổi ở bên ngoài không hề ảnh hưởng đến quá trình xác lập trạng thái lối ra của trigơ, chính vì thế hoạt động của trigơ JK master – slave mang tính dứt khoát và ổn định cao hơn là trigơ khác.

Nếu trigơ có xung nhịp tác động bởi sườn dương thì lối ra sẽ thay đổi trạng thái theo đầu vào điều khiển ứng với sườn âm của xung nhịp và ngược lại.

5.1.7. Trigơ D (Delay)

D	Q_n	Q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

a)

D	Q_{n+1}
0	0
1	1

b)

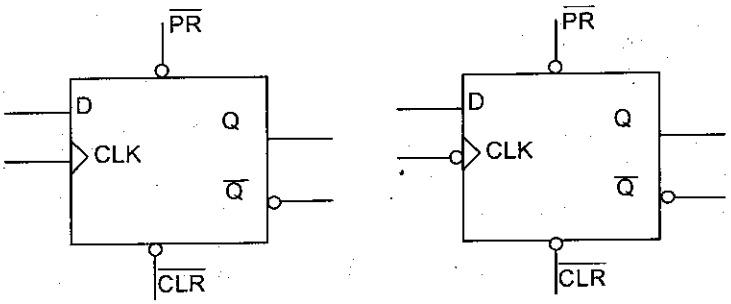
Hình 5.1.7.1. Bảng trạng thái của trigơ D

Trigơ D là loại trigơ được dùng nhiều trong các bộ ghi lưu trữ các bit thông tin nhị phân. Trigơ D có một lối vào dữ liệu được ký hiệu bằng chữ D hoạt động đồng bộ với lối vào xung nhịp CLK, trigơ D hoạt động theo nguyên tắc sau: số liệu ở lối vào

D sẽ được chuyển đến lối ra Q của trigơ sau một xung nhịp, tức là số liệu được chuyển đến lối ra chậm mất một khoảng thời gian bằng độ rộng của xung nhịp. Chính vì vậy mà nó có tên là trigơ D lấy theo chữ đầu của thuật ngữ tiếng Anh – Delay có nghĩa là trễ.

Bảng trạng thái đầy đủ và rút gọn cho trên hình 5.1.7.1a,b.

Ký hiệu logic của D – FF được cho trên hình 5.1.7.2.

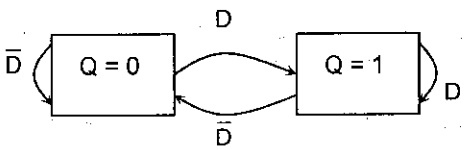


Hình 5.1.7.2. Ký hiệu của D – FF

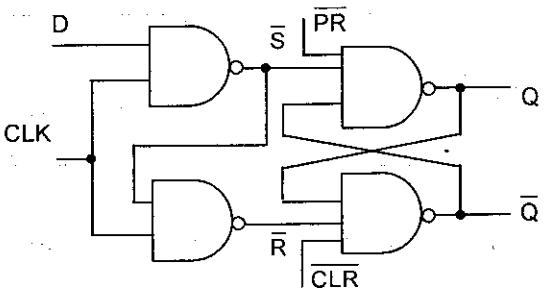
Phương trình logic của D – FF: $Q_{n+1} = D_n$.

Đồ hình chuyển đổi trạng thái của D – FF cho trên hình 5.1.7.3.

Sơ đồ của D – FF được cho trên hình 5.1.7.4.

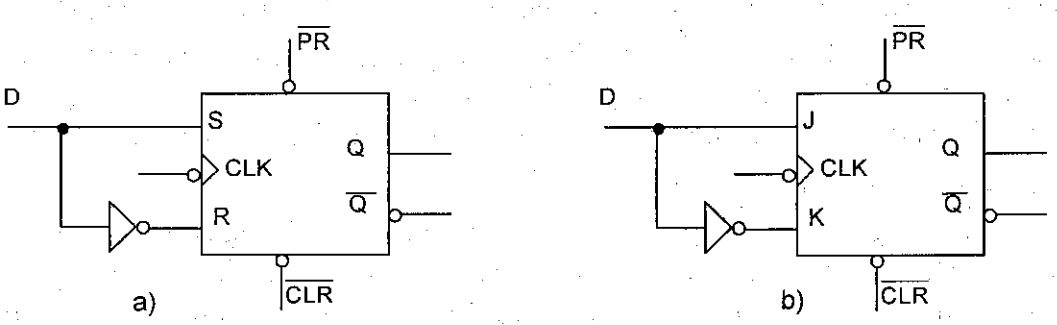


Hình 5.1.7.3. Đồ hình trạng thái của D – FF



Hình 5.1.7.4. Sơ đồ của D – FF

Trigơ D có thể xây dựng từ trigơ RS hoặc trigơ JK khi ta mắc như ở hình 5.1.7.5a và b.



Hình 5.1.7.5. Chuyển từ RS – FF hoặc JK – FF sang D – FF

5.1.8. Trigơ T (Toggle)

Trigơ T có một lối vào điều khiển được ký hiệu bằng chữ T. Trigơ T hoạt động theo nguyên tắc sau: khi đầu vào T ở mức logic 0 thì đầu ra giữ nguyên trạng thái còn khi đầu vào T ở mức logic 1 thì đầu ra lật lại trạng thái trước đó. Chính vì vậy nó có tên là T lấy theo chữ đầu của thuật ngữ tiếng Anh – Toggle có nghĩa là lật.

Bảng trạng thái đầy đủ và rút gọn của T – FF được cho trên hình 5.1.8.1a,b.

T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

a)

T	Q_{n+1}
0	Q_n
1	$\overline{Q_n}$

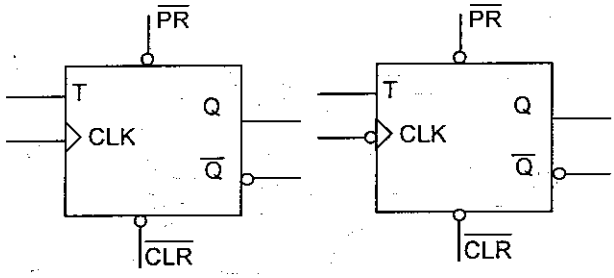
b)

Hình 5.1.8.1. Bảng chân lý của trigơ T

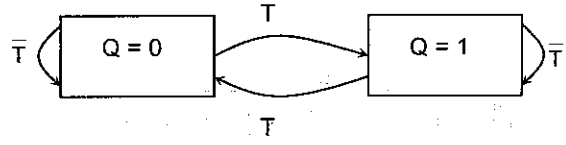
Phương trình logic của trigơ T: $Q_{n+1} = \overline{T}Q + T\overline{Q} = T \oplus Q$

Ký hiệu logic của T – FF được cho trên hình 5.1.8.2.

Đồ hình chuyển đổi trạng thái của T – FF trên hình 5.1.8.3.



Hình 5.1.8.2. Ký hiệu của T – FF



Hình 5.1.8.3. Đồ hình trạng thái của D – FF

Chức năng của T – FF chính là chức năng của JK – FF khi J = K = 1.

5.1.9. Xác định đầu vào điều khiển (đầu vào kích) cho FF

Trong nhiều trường hợp, đặc biệt khi muốn thiết kế mạch dùng FF cần phải xác định đầu vào điều khiển của FF ứng với sự chuyển đổi trạng thái cho trước Q_n sang Q_{n+1}.

Với mỗi FF, sự chuyển biến trạng thái Q_n sang Q_{n+1} chỉ xảy ra trong 4 khả năng: 0 sang 0, 0 sang 1, 1 sang 0 và 1 sang 1. Căn cứ vào chức năng của từng loại FF phải xác định giá trị đầu vào điều khiển R, S, J, K, D, T tương ứng với các chuyển đổi ấy.

Ví dụ 5.1.9.1. Xét đối với trigơ RS.

Từ bảng trạng thái của trigơ RS ta thấy khi đầu ra Q giữ nguyên trạng thái 0 thì có hai khả năng hoặc trạng thái nhớ hoặc trạng thái xoá. Với hai khả năng đó thì S luôn luôn bằng 0 còn R có thể 0 hoặc 1 (có nghĩa R là bất kỳ, ký hiệu "x"). Khi đầu ra Q chuyển từ 0 sang 1 thì chỉ có 1 khả năng là trạng thái thiết lập tức R = 0, S = 1.

Khi đầu ra Q chuyển từ 1 sang 0 thì cũng chỉ có một khả năng là trạng thái xoá tức R = 1, S = 0. Còn khi đầu ra giữ nguyên trạng thái 1 thì có hai khả năng hoặc trạng thái nhớ hoặc trạng thái thiết lập. Với hai khả năng đó thì R luôn luôn bằng 0 còn S có thể 0 hoặc 1 (ký hiệu "x").

Đối với các trigơ khác cũng xét tương tự như trên ta được bảng các đầu vào điều khiển tương ứng với sự chuyển biến trạng thái Q_n sang Q_{n+1} của các trigơ cho trên hình 5.1.9.1.

Q_n	Q_{n+1}	RS	JK	D	T
0	0	x0	0x	0	0
0	1	01	1x	1	1
1	0	10	x1	0	1
1	1	0x	x0	1	0

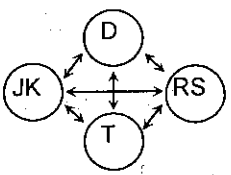
Hình 5.1.9.1. Bảng các đầu vào điều khiển

5.1.10. Chuyển đổi giữa các trigơ số

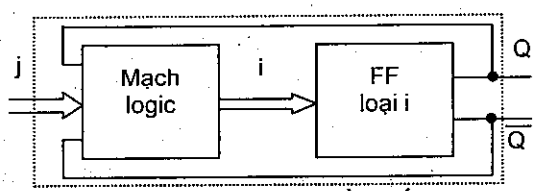
Có 4 loại trigơ đã giới thiệu là RS, JK, D, T. Trong thực tế có khi FF loại này lại được mắc mạch và được sử dụng như FF loại khác. Do đó, phải biết cách chuyển đổi giữa các trigơ số.

Với 4 loại FF có 12 khả năng chuyển đổi như trên hình 5.1.10.1.

Một trong những phương pháp để xây dựng FF loại j từ FF loại i cho trước được cho ở sơ đồ khối hình 5.1.10.2. Ở đây ký hiệu i và j là loại FF.



Hình 5.1.10.1



Hình 5.1.10.2. Sơ đồ khối

Trong sơ đồ trên, các đầu vào j là các đầu vào của FF loại j cần thiết kế, đầu ra của mạch logic là các đầu vào của FF loại i cho trước. Như vậy, bài toán chuyển đổi từ FF loại i sang j là xây dựng mạch logic tổ hợp có các đầu vào là j và Q, các đầu ra là i biểu diễn bởi hệ hàm: i = f(j, Q).

Để thực hiện chuyển đổi FF loại i sang FF loại j cần thực hiện các bước sau:

1. Xác định hệ hàm i = f(j, Q) từ bảng các đầu vào điều khiển.

2. Tối thiểu hóa các hàm này và xây dựng sơ đồ.

Ví dụ 5.1.10.1. Chuyển từ trigơ RS sang trigơ JK.

Ta có trigơ RS là trigơ cho trước và trigơ JK là trigơ cần thiết kế, như vậy đầu vào của mạch logic là J và K, đầu ra là R và S.

Xác định hệ hàm S = f(J, K, Q) và R = f(J, K, Q) từ bảng các đầu vào điều khiển trên hình 5.1.10.3.

J	K	Q_n	Q_{n+1}	S	R
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	X	0
1	1	0	1	1	0
1	1	1	0	0	1

Hình 5.1.10.3. Bảng các đầu vào điều khiển

Tối thiểu hoá các hàm S và R bằng bảng Karnaugh như trên hình 5.1.10.4.

S	JK 00	01	11	10
Q_n 0	0	0	1	1
1	x	0	0	x

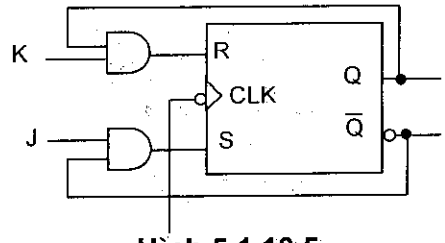
R	JK 00	01	11	10
Q_n 0	x	x	0	0
1	0	1	1	0

Hình 5.1.10.4. Tối thiểu hoá các hàm S và R

Vậy: $R = KQ_n$; $S = J\overline{Q}_n$

Sơ đồ thực hiện chuyển đổi như trên hình 5.1.10.5.

Chúng ta cũng có thể dựa vào phương trình đặc trưng của từng loại trigơ để thực hiện chuyển đổi. Cách này tiện cho trình bày viết, có thể dùng đại số logic xử lý, nhưng cần kỹ xảo nhất định. Còn với phương pháp đã nói ở trên có nhiều phiền phức chút ít nhưng trực quan, ít sai.



Hình 5.1.10.5

Ví dụ 5.1.10.2. Chuyển từ trigơ RS sang trigơ JK:

Phương trình đặc trưng của trigơ RS:

$$\begin{cases} Q_{n+1} = S + \overline{R}Q_n \\ RS = 0 \end{cases}$$

Phương trình của trigơ JK: $Q_{n+1} = J\overline{Q_n} + \overline{K}Q_n$

So sánh, ta có:

$$\begin{cases} S = J\overline{Q}_n \\ R = K \end{cases}$$

Vì điều kiện ràng buộc R.S = 0 nên ta phải kiểm tra. Khi J = K = 1 và $Q_n = 0$ thì:

$$\begin{cases} S = J\overline{Q}_n = 1 \\ R = K = 1 \end{cases}$$

không thoả mãn R.S = 0. Ta biến đổi lại: $Q_{n+1} = J\overline{Q}_n + \overline{K}Q_n = J\overline{Q}_n + \overline{KQ_n}Q_n$

So sánh lại, ta có : $\begin{cases} S = J\overline{Q}_n \\ R = KQ_n \end{cases}$

Sơ đồ chuyển đổi như đã trình bày ở phương pháp trước.

5.1.11. Một số vi mạch trigơ

1. 7470: JK – FF

– Xung nhịp C_K tác động tại sườn dương.

– PR, CLR tích cực ở mức thấp

2. 7472: trigơ chủ tớ

– Xung nhịp C_K tích cực ở mức cao

– PR, CLR tích cực ở mức cao.

3: 7473 và 74LS73: hai JK – FF

– Đầu vào xoá tích cực ở mức thấp

– Xung nhịp C_K:

+ 7473: tích cực ở mức cao

+74LS73: tích cực ở sườn âm.

4. 7476 và 74LS76: hai JK – FF

– Đầu vào thiết lập, xoá tích cực ở mức thấp.

– Xung nhịp C_K:

+ 7476: tích cực ở mức cao.

+ 74LS76: tích cực ở sườn âm.

5. 74107 và 74LS107: hai JK – FF

– Đầu vào xoá tích cực ở mức thấp

– Xung nhịp C_K:

+ 74107: tích cực ở mức cao.

+74LS107: tích cực ở sườn âm.

6. 74LS112: Hai JK – FF

– Xung nhịp C_K tích cực ở sườn âm.

– Đầu vào thiết lập, xoá tích cực ở mức thấp.

5.2. CÁC BỘ ĐẾM

5.2.1. Đặc điểm và phân loại bộ đếm

a) Đặc điểm cơ bản

Đếm là khả năng nhớ được số xung đầu vào, mạch điện thực hiện thao tác đếm được gọi là bộ đếm.

Đếm là một thao tác cơ bản cực kỳ quan trọng, bộ đếm được sử dụng vô cùng rộng rãi, từ các thiết bị đo chỉ thị số đến các máy tính điện tử số loại lớn, bất kỳ hệ thống số hiện đại nào cũng đều hiện diện bộ đếm.

b) Phân loại

Căn cứ vào sự khác biệt của tình huống chuyển đổi trạng thái các trigơ trong bộ đếm, người ta phân thành hai loại lớn: Bộ đếm đồng bộ (bộ đếm song song) và bộ đếm không đồng bộ (bộ đếm nối tiếp). Trong bộ đếm đồng bộ, các trigơ đều chịu tác động điều khiển của một xung đồng hồ duy nhất, đó là xung đếm đầu vào. Vậy sự chuyển đổi trạng thái của chúng là đồng bộ. Bộ đếm không đồng bộ thì khác, có FF chịu tác động điều khiển trực tiếp của xung đếm đầu vào, nhưng cũng có trigơ chịu tác động điều khiển của xung đầu ra của FF khác. Vậy sự chuyển đổi trạng thái của các FF không cùng lúc tức là không đồng bộ.

Căn cứ vào sự khác biệt về hệ số đếm của bộ đếm, người ta phân thành các loại: bộ đếm nhị phân, bộ đếm thập phân,...

Căn cứ vào tác động của xung đếm đầu vào mà số đếm của bộ đếm tăng hay giảm người ta phân thành 3 loại: bộ đếm thuận, bộ đếm nghịch và bộ đếm thuận nghịch.

5.2.2. Mã của bộ đếm

Quá trình đếm của bộ đếm là quá trình thay đổi từ trạng thái trong này đến trạng thái trong khác và mỗi trạng thái trong của bộ đếm được mã hóa bởi một mã cụ thể.

Cùng một bộ đếm có thể có nhiều cách mã hoá các trạng thái trong khác nhau. Các cách mã hoá khác nhau sẽ tương ứng với những mạch thực hiện khác nhau mặc dù các mạch đó có cùng chức năng, hệ số đếm K_d... Sau đây là một vài loại mã thường dùng để mã hoá trong bộ đếm.

a) Mã nhị phân

Mã nhị phân là loại mã mà các bit của nó có trọng số là 2^{n-1}– 2^{n-2}– … –8–4–2–1.

b) Mã Gray

c) Mã BCD

d) Mã Johnson

Mã Johnson là loại mã có đặc điểm:

– Nếu dùng n biến nhị phân thì sẽ mã hoá được tối đa là 2n trạng thái.

– Hai từ mã kề nhau chỉ khác nhau một bit.

Bảng mã Johnson với 5, 4, 3, 2 bit cho trên hình 5.2.2.1.

A	B	C	D	E
0	0	0	0	0
0	0	0	0	1
0	0	0	1	1
0	0	1	1	1
0	1	1	1	1
1	1	1	1	1
1	1	1	1	0
1	1	1	0	0
1	1	0	0	0
1	0	0	0	0

A	B	C	D
0	0	0	0
0	0	0	1
0	0	1	1
0	1	1	1
1	1	1	1
1	1	1	0
1	1	0	0
1	0	0	0

A	B	C
0	0	0
0	0	1
0	1	1
1	1	1
1	1	0
1	0	0

A	B
0	0
0	1
1	1
1	0

Hình 5.2.2.1. Bảng mã Johnson

e) Mã vòng

Mã vòng có đặc điểm:

– Nếu dùng n biến nhị phân thì mã hoá được n trạng thái.

– Hai từ mã kề nhau luôn luôn khác nhau ở hai biến.

– Trong từ mã chỉ có duy nhất một bit bằng 1, các bit khác bằng 0. Bit 1 được dịch từ bit có trọng số nhỏ nhất đến bit có trọng số lớn nhất tạo thành một vòng khép kín.

Ví dụ 5.2.2.1. Mã vòng 6 bit có các từ mã như bảng trên hình 5.2.2.2 (Quy ước A là bit có trọng số lớn nhất):

A	B	C	D	E	F
0	0	0	0	0	1
0	0	0	0	1	0
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	0	0	0
1	0	0	0	0	0

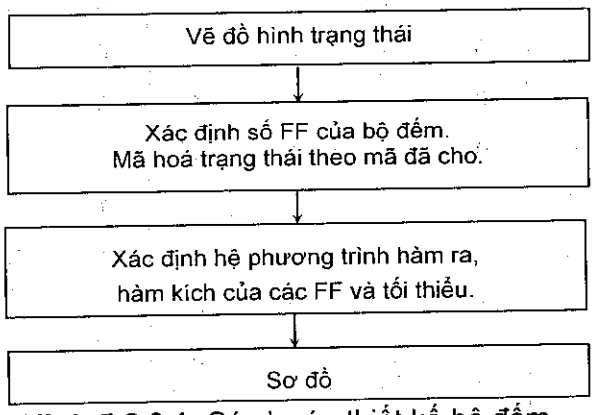
Hình 5.2.2.2. Mã vòng 6 bit

5.2.3. Các bước thiết kế bộ đếm

Các bước thiết kế bộ đếm được mô tả như hình 5.2.3.1.

Bước 1: Vẽ đồ hình trạng thái của bộ đếm:

Căn cứ vào yêu cầu của bộ đếm cần thiết kế như: hệ số đếm ($K_đ$) và một số các yêu cầu khác để xây dựng đồ hình mô tả hoạt động của bộ đếm.



Hình 5.2.3.1. Các bước thiết kế bộ đếm

Bước 2: Xác định số FF của bộ đếm, mã hoá các trạng thái trong bộ đếm theo mã đã cho.

Trước tiên phải xác định được n là số FF cần thiết kế để mã hoá cho $K_đ$ trạng thái trong của bộ đếm, n phải thoả mãn điều kiện sau:

– Đối với mã nhị phân và mã Gray : $n \geq \log_2 K_đ$

– Đối với mã vòng : $n = K_đ$

– Đối với mã Johnson : $n = (1/2)K_đ$

Sau đó tiến hành mã hoá các trạng thái trong bộ đếm theo mã đã cho.

Bước 3: Xác định hàm các đầu vào điều khiển của các FF và hàm ra:

Phương pháp xác định hàm các đầu vào điều khiển cho các FF và hàm ra của bộ đếm có thể xác định theo hai cách sau:

– Dựa vào bảng chuyển đổi trạng thái, bảng ra để xác định các phương trình đầu vào điều khiển cho các FF và phương trình hàm ra.

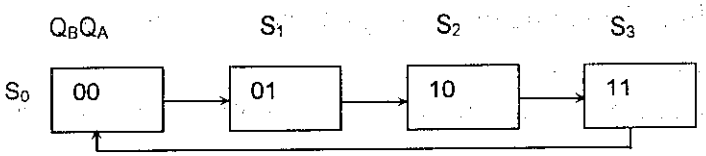
– Dựa trực tiếp vào đồ hình chuyển đổi trạng thái viết các phương trình đầu vào điều khiển cho các FF và phương trình hàm ra.

Bước 4: Sơ đồ thực hiện:

Từ các phương trình đầu vào điều khiển các FF và phương trình hàm ra, đưa ra sơ đồ mạch thực hiện.

Ví dụ 5.2.3.1. Xây dựng bộ đếm nhị phân, đếm tiến có $K_đ = 4$.

(Bước 1 và 2) Gọi số FF thiết kế mã hoá là n: $n \geq \log_2 K_đ \Rightarrow n = 2$. Như vậy, phải dùng 2 FF (A và B) để mã hoá các trạng thái trong bộ đếm. Sau đó tiến hành mã hoá trạng thái trong của bộ đếm (S_0, S_1, S_2, S_3) theo mã nhị phân như sau: 00, 01, 10, 11 (Quy ước A là bit có trọng số nhỏ nhất). Ta được đồ hình chuyển đổi trạng thái, hình 5.2.3.2.



Hình 5.2.3.2

(Bước 3) Xác định hàm các đầu vào điều khiển của FF và hàm ra:

Từ đồ hình chuyển đổi trạng thái ta thấy S_0 là trạng thái hiện tại thì S_1 là trạng thái tiếp theo, S_1 là trạng thái hiện tại thì S_2 là trạng thái tiếp theo,... S_3 là trạng thái hiện tại thì S_0 là trạng thái tiếp theo.

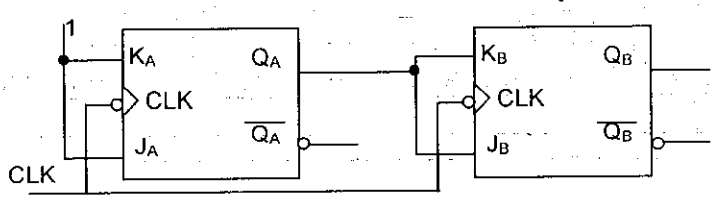
Nếu dùng trigơ JK, ta có bảng trạng thái của các đầu vào điều khiển trên hình 5.2.3.3.

Trạng thái hiện tại		Trạng thái tiếp theo		Chọ B – FF		Cho A – FF	
Q_B^n	Q_A^n	Q_B^{n+1}	Q_A^{n+1}	K_B	J_B	K_A	J_A
0	0	0	1	X	0	X	1
0	1	1	0	X	1	1	x
1	0	1	1	0	x	x	1
1	1	0	0	1	x	1	x

Hình 5.2.3.3. Bảng trạng thái các đầu vào điều khiển

Thực hiện tối thiểu hoá hàm logic với các đầu vào là Q_A và Q_B ở trạng thái hiện tại (Q_B^n, Q_A^n), đầu ra là các đầu vào điều khiển. Ta được: $K_B = J_B = Q_A$; $K_A = J_A = 1$

Sơ đồ thực hiện cho ở hình 5.2.3.4.



Hình 5.2.3.4. Bộ đếm nhị phân $K_đ = 4$.

5.2.4. Các bộ đếm nhị phân

a) Đếm nhị phân không đồng bộ

Đếm nhị phân không đồng bộ còn được gọi là đếm nối tiếp: các trigơ mắc nối tiếp với nhau, lối ra của trigơ trước được nối với lối vào xung nhịp của trigơ sau.

Đặc điểm của bộ đếm này là xung nhịp CLK không được đưa đồng thời vào các trigơ. Xung nhịp chỉ được đưa vào và làm chuyển trạng thái của FF đầu tiên, lối ra của FF trước làm chuyển trạng thái của FF tiếp theo.

Trong đếm nhị phân không đồng bộ gồm các loại đếm sau:

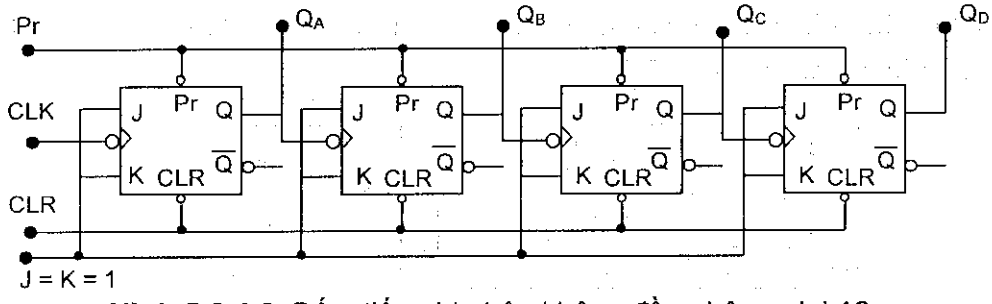
– *Đếm tiến (Up counter):*

Trạng thái ra của bộ đếm tiến modul 16 theo số xung nhịp đưa tới đầu vào cho trên hình 5.2.4.1.

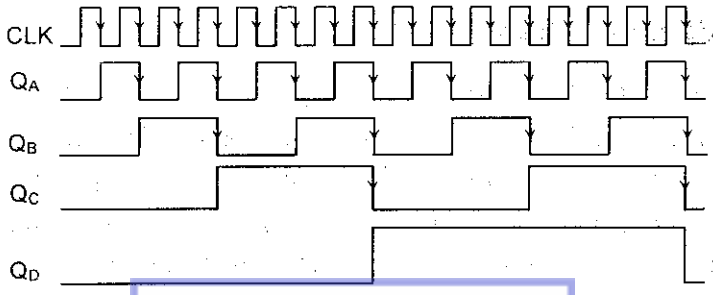
Xung nhịp	Q_D	Q_C	Q_B	Q_A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1
16	0	0	0	0

Hình 5.2.4.1. Trạng thái ra của bộ đếm tiến modul 16

Sơ đồ đếm nhị phân không đồng bộ 4 bit đếm tiến dùng trigơ JK 7476 được trình bày trên hình 5.2.4.2, giản đồ xung được trình bày trên hình 5.2.4.3.



Hình 5.2.4.2. Đếm tiến nhị phân không đồng bộ modul 16



Hình 5.2.4.3. Giản đồ thời gian đếm tiến modul 16

Như ký hiệu trên sơ đồ hình 5.2.4.2, muốn xoá ta để Pr = 1, CLR = 0, muốn thiết lập để Pr = 0, CLR = 1. Để bộ đếm làm việc ở chế độ đếm ta để Pr = CLR = 1. Dựa vào nguyên lý hoạt động của trigơ JK ta giải thích hoạt động của bộ đếm này.

+ Đầu tiên xoá mạch đếm bằng xung xoá CLR = 0. Lúc đó trạng thái lối ra của cả 4 trigơ đều chuyển về 0: $Q_AQ_BQ_CQ_D = 0000$.

+ Sau đó để Pr = CLR = 1.

+ Đặt lối vào đếm J = K = 1: Mạch đếm bắt đầu hoạt động theo trạng thái của các lối vào đồng bộ J, K và xung nhịp như giản đồ hình 5.2.4.3. Tất cả 4 trigơ đều có J = K = 1 nên khi có xung nhịp tác dụng các trigơ đều chuyển trạng thái.

Trigơ A chuyển trạng thái với mọi xung nhịp tác động chuyển từ 1 về 0.

Trigơ B chuyển trạng thái khi Q_A chuyển từ 1 về 0.

Trigơ C chuyển trạng thái khi Q_B chuyển từ 1 về 0.

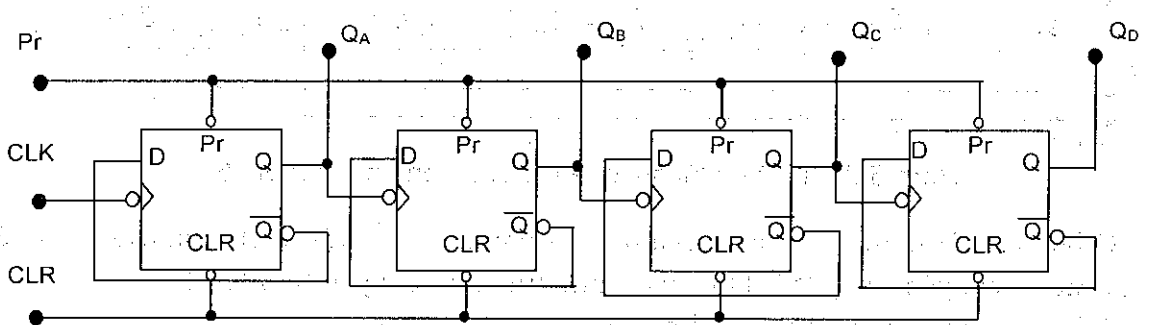
Trigơ D chuyển trạng thái khi Q_C chuyển từ 1 về 0.

Nhìn giản đồ xung trên hình 5.2.4.3 ta thấy mỗi trigơ chia tần số xung nhịp làm 2. Có 4 trigơ sẽ chia tần số xung nhịp $2^4 = 16$ lần, nếu có n trigơ sẽ có bộ chia 2^n lần. Như vậy, bộ đếm cũng là bộ chia tần.

Ta cũng có thể dùng trigơ D mắc thành bộ đếm nhị phân. Muốn vậy ta phải mắc lối ra $\overline{Q}$ của trigơ D với lối vào D của nó. Khi đó trạng thái lối ra của trigơ sẽ được xác định theo phương trình sau: $Q_{n+1} = D_n = \overline{Q_n}$

Trường hợp này tương tự như đối với trigơ JK khi các lối vào J = K = 1, nghĩa là cứ sau mỗi lần có xung nhịp tác dụng trigơ lại chuyển trạng thái một lần.

Sơ đồ đếm nhị phân 4 bit dùng trigơ D 7474 cho trên hình 5.2.4.4.



Hình 5.2.4.4. Đếm tiến nhị phân không đồng bộ modul 16 dùng D – FF

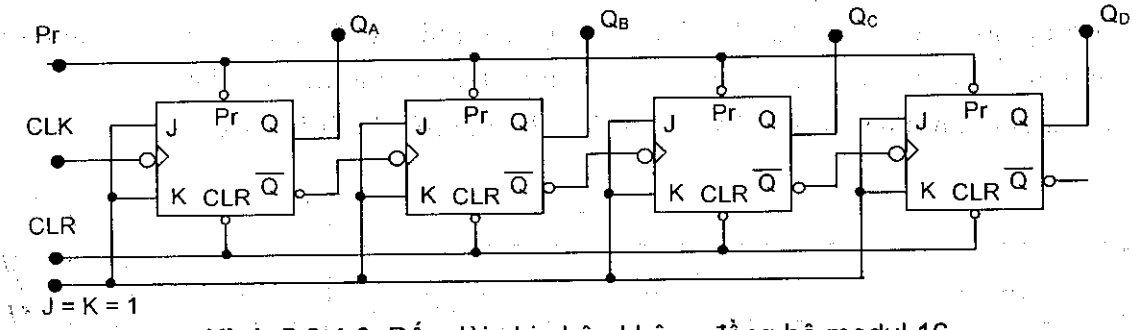
– Đếm lùi (Down counter):

Trạng thái ra của các trigơ trong bộ đếm lùi modul 16 được cho trên hình 5.2.4.5.

Xung nhịp	Q_D	Q_C	Q_B	Q_A
0	0	0	0	0
1	1	1	1	1
2	1	1	1	0
3	1	1	0	1
4	1	1	0	0
5	1	0	1	1
6	1	0	1	0
7	1	0	0	1
8	1	0	0	0
9	0	1	1	1
10	0	1	1	0
11	0	1	0	1
12	0	1	0	0
13	0	0	1	1
14	0	0	1	0
15	0	0	0	1
16	0	0	0	0

Hình 5.2.4.5. Trạng thái ra của bộ đếm lùi modul 16

Ta có thể xây dựng mạch đếm lùi nhị phân 4 bit dùng trigơ JK 7476, mắc mạch theo sơ đồ trên hình 5.2.4.6.



Hình 5.2.4.6. Đếm lùi nhị phân không đồng bộ modul 16

Ở bộ đếm lùi ta thấy lối ra $\overline{Q}$ của trigơ trước được nối vào CLK của trigơ sau nên trigơ sau sẽ chuyển trạng thái khi trigơ đứng trước nó chuyển từ 0 lên 1.

Trigơ A thay đổi trạng thái với mọi xung nhịp tác động

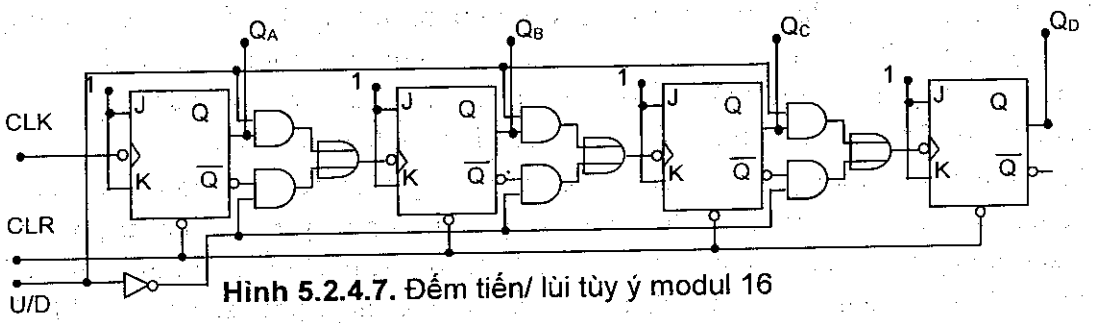
Trigơ B thay đổi trạng thái khi Q_A chuyển từ 0 lên 1.

Trigơ C thay đổi trạng thái khi Q_B chuyển từ 0 lên 1.

Trigơ D thay đổi trạng thái khi Q_C chuyển từ 0 lên 1.

– Đếm tiến/ lùi tuỳ ý:

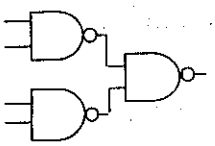
Để có một bộ vừa đếm tiến vừa đếm lùi ta thêm một đầu vào điều khiển tiến lùi UP/DOWN sơ đồ mạch đếm tiến lùi như hình 5.2.4.7.



Hình 5.2.4.7. Đếm tiến/ lùi tùy ý modul 16

Đếm tiến: Khi cho lối vào điều khiển tiến lùi U/D = "1" lối ra Q của trigơ trước nối với CLK của trigơ tiếp theo, sơ đồ tương đương như hình 5.2.4.2, ta có mạch đếm tiến.

Đếm lùi: Khi cho lối vào điều khiển tiến lùi U/D = "0" lối ra $\overline{Q}$ của trigơ trước nối với CLK của trigơ tiếp theo, sơ đồ tương đương như hình 5.2.4.6, ta có mạch đếm lùi.



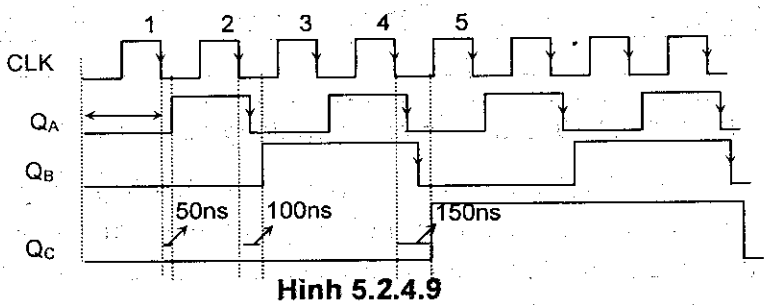
Hình 5.2.4.8

Ta có thể thay phần mạch gồm các phần tử AND, OR trong hình 5.2.4.7 bằng các phần tử NAND như hình 5.2.4.8.

– Thời gian trễ do truyền trong bộ đếm không đồng bộ:

Đếm không đồng bộ là dạng đơn giản nhất trong các bộ đếm nhị phân, vì chúng đòi hỏi ít linh kiện nhất để tạo hoạt động đếm cho trước. Tuy nhiên, chúng có một khuyết điểm lớn do nguyên lý hoạt động cơ bản của chúng gây nên: mỗi trigơ được khởi động do sự chuyển trạng thái tại đầu ra của trigơ trước đó. Mặt khác, với mỗi trigơ nó có một thời gian trễ do truyền là t_{pd}, điều này có nghĩa là trigơ thứ hai sẽ không phản ứng gì trong khoảng thời gian t_{pd} kể từ khi trigơ đầu tiên nhận được một chuyển đổi tích cực ở xung đếm, trigơ thứ ba sẽ không phản ứng gì trong khoảng thời gian $2t_{pd}$ từ lúc xảy ra hoạt động chuyển đổi, như vậy trigơ thứ n sẽ không phản ứng gì trong khoảng thời gian $(n-1)t_{pd}$ kể từ lúc xảy ra hoạt động chuyển đổi. Và như vậy phải sau khoảng thời gian nt_{pd} thì ta mới nhận được sự thay đổi ở lối ra của trigơ n.

Ví dụ 5.2.4.1. Xét dạng sóng ở các lối ra của bộ đếm nhị phân không đồng bộ 3 bit, hình 5.2.4.9.



Hình 5.2.4.9

Nhìn vào dạng sóng trên ta thấy:

Giả sử chu kỳ của xung nhịp là 1000ns và thời gian trễ do truyền của mỗi trigơ

là 50ns. Tức là trigơ A lật chậm 50ns sau khi xung nhịp thay đổi từ 1 sang 0, tương tự trigơ B lật chậm 50ns sau khi trigơ A chuyển từ 1 sang 0, tương tự với trigơ C. Như vậy, trigơ C thay đổi trạng thái trễ so với xung nhịp tác động là 150ns. Tuy vậy, ta thấy các trigơ cũng vẫn ở trạng thái đúng biểu diễn số đếm nhị phân.

Tuy nhiên tình huống sẽ trở nên xấu đi nếu xung nhịp đưa vào có tần số cao hơn. Giả sử chu kỳ của xung nhịp là 100ns và thời gian trễ do truyền của mỗi trigơ là 50ns. Lẽ ra sau xung nhịp thứ 4 chuyển từ 1 sang 0 thì bộ đếm sẽ đếm số nhị phân là 100, nhưng ở đây sau xung nhịp thứ 4 đầu ra C vẫn ở mức thấp, phải sau 150ns thì đầu ra C mới lên mức cao. Nhưng lúc này trigơ A lại ở mức cao và ta được số nhị phân là 101, như vậy sẽ không có trạng thái 100.

Có thể phòng tránh những lỗi như vậy nếu giai đoạn giữa các xung vào được kéo dài hơn tổng thời gian trễ của bộ đếm. Do đó, để bộ đếm hoạt động đúng ta cần:

$$T_{CLK} \geq n.t_{pd}.$$

Như vậy tần số lớn nhất có thể sử dụng: $f_{max} = 1/(n.t_{pd})$

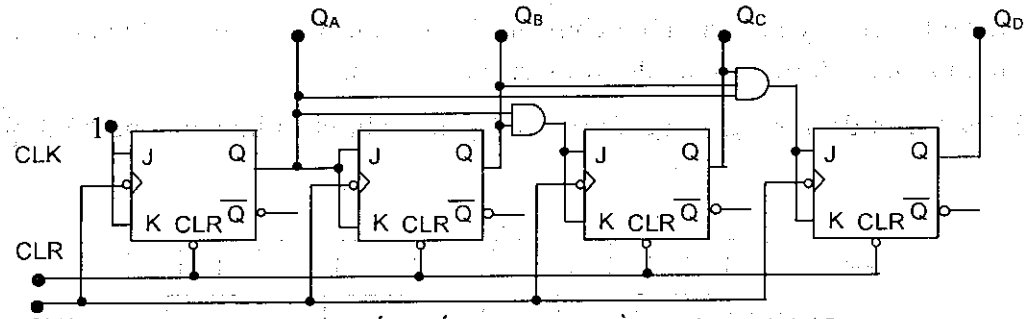
Để tăng dung lượng của bộ đếm thì số trigơ sử dụng sẽ tăng lên, khi đó thời gian trễ do truyền tích lũy sẽ tăng lên, do đó người ta sử dụng bộ đếm nhị phân đồng bộ khi đếm dung lượng lớn.

b) Đếm nhị phân đồng bộ

Đếm nhị phân đồng bộ còn gọi là đếm song song. Đếm không đồng bộ có nhược điểm là tốc độ chậm vì có quá trình trễ khi đi qua các trigơ. Để khắc phục nhược điểm đó người ta dùng mạch đếm song song, nghĩa là các xung nhịp đồng thời tác dụng vào tất cả các trigơ.

– Đếm tiến:

Sơ đồ đếm tiến nhị phân đồng bộ 4 bit cho trên hình 5.2.4.10.



Hình 5.2.4.10. Đếm tiến nhị phân đồng bộ modul 16

Từ sơ đồ trên hình 5.2.4.10 ta thấy: Tuy xung nhịp tác động đồng thời vào các trigơ nhưng chỉ trigơ nào có J = K = 1 thì nó mới chuyển trạng thái. Từ sơ đồ hình 5.2.4.10 ta có được các điều kiện chuyển trạng thái các của trigơ trong bộ đếm như sau:

Trigơ A chuyển trạng thái với mọi xung CLK.

Trigơ B chuyển khi $Q_A = 1$.

Trigơ C chuyển khi $Q_A = Q_B = 1$

Trigơ D chuyển khi $Q_A = Q_B = Q_C = 1$

Như vậy các trigơ sau chỉ chuyển trạng thái khi tất cả lối ra Q của các trigơ ở trước nó đồng thời bằng 1. Quá trình đếm của sơ đồ có thể mô tả như sau:

Khi tác dụng xung xoá CLR thì $Q_D\ Q_C\ Q_B\ Q_A = 0000$.

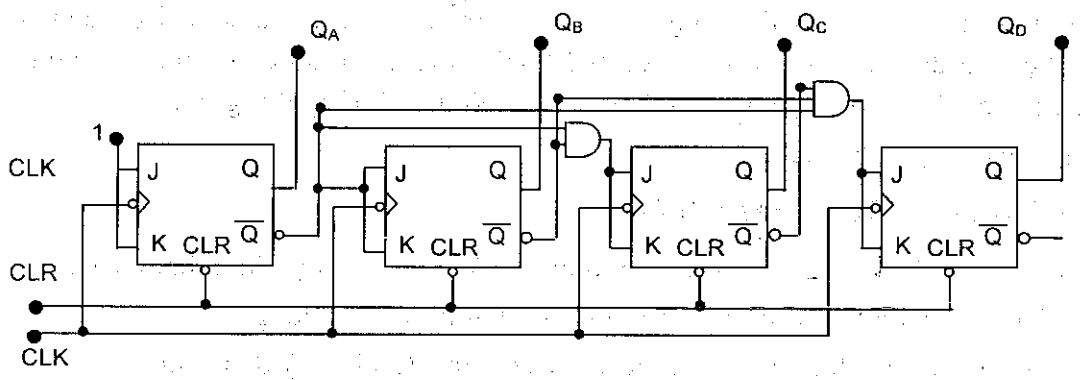
Khi có xung nhịp đầu tiên tác dụng chỉ trigơ A chuyển trạng thái từ 0 lên 1, các trigơ B, C, D không chuyển trạng thái vì J = K = 0, trạng thái lối ra của bộ đếm sau khi kết thúc xung nhịp thứ nhất là: 0001.

Khi có xung nhịp thứ hai tác dụng: J, K của trigơ B là 1 nên B và A đều chuyển trạng thái, Q_A từ 1 về 0, Q_B từ 0 lên 1; trigơ D và C vẫn chưa chuyển trạng thái, trạng thái ở lối ra của bộ đếm sau khi kết thúc xung nhịp thứ hai là: 0010.

Quá trình hoạt động của bộ đếm nhị phân đồng bộ cũng diễn ra tiếp tục như bộ đếm nhị phân không đồng bộ, nó có giản đồ xung và bảng chân lý như bộ đếm nhị phân không đồng bộ đã nêu ở trên.

– Đếm lùi:

Trong sơ đồ bộ đếm tiến trên hình 5.2.4.10 thay đầu ra Q bằng đầu ra $\overline{Q}$ ta sẽ được bộ đếm lùi như trên hình 5.2.4.11.



Hình 5.2.4.11. Đếm lùi nhị phân đồng bộ modul 16

Từ sơ đồ hình 5.2.4.11 ta có được các điều kiện chuyển trạng thái các của trigơ trong bộ đếm như sau:

Trigơ A chuyển trạng thái với mọi xung CLK.

Trigơ B chuyển khi $Q_A = 0$ ($\overline{Q_A} = 1$)

Trigơ C chuyển khi $Q_A = Q_B = 0$ ($\overline{Q_A} = \overline{Q_B} = 1$)

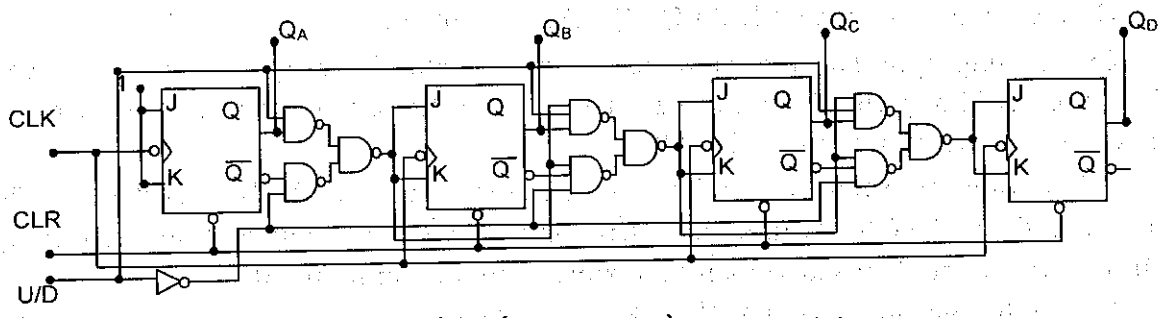
Trigơ D chuyển khi $Q_A = Q_B = Q_C = 0$ ($\overline{Q_A} = \overline{Q_B} = \overline{Q_C} = 1$)

Như vậy, các trigơ sau chỉ chuyển trạng thái khi tất cả lối ra Q của các trigơ ở trước nó đồng thời bằng 0. Bảng chân lý và giản đồ xung của bộ đếm lùi đồng bộ modul 16 như bộ đếm không đồng bộ ta đã xét ở trên.

– Đếm tiến/ lùi tuỳ ý:

Sơ đồ bộ đếm modul 16 đồng bộ đếm tiến lùi tuỳ ý cho trên hình 5.2.4.12:

Tuỳ thuộc đầu vào U/D là 1 hay 0 mà ta sẽ được bộ đếm nhị phân modul 16 tiến hay lùi tương tự như bộ đếm không đồng bộ tiến/lùi tuỳ ý.



Hình 5.2.4.12. Đếm tiến lùi tùy ý đồng bộ modul

– Ưu điểm của bộ đếm đồng bộ so với bộ đếm không đồng bộ:

Trong một bộ đếm đồng bộ, mọi trigơ sẽ thay đổi trạng thái đồng thời, điều đó có nghĩa chúng được đồng bộ hoá theo mức tích cực của xung nhịp. Do đó không giống như bộ đếm không đồng bộ, những khoảng trễ do truyền sẽ không được cộng lại với nhau mà nó chỉ gồm thời gian trễ của 1 trigơ. Thời gian trễ là như nhau bất kể bộ đếm đó có bao nhiêu trigơ. Nói chung là thời gian trễ bé hơn nhiều so với bộ đếm không đồng bộ. Do đó, bộ đếm đồng bộ có thể hoạt động ở tần số cao hơn, dĩ nhiên mạch điện của bộ đếm đồng bộ phức tạp hơn so với bộ đếm không đồng bộ.

Yêu cầu: $T_{CLK} \geq t_{pd}$ của trigơ.

5.2.5. Bộ đếm đặt lại trạng thái

Bộ đếm có $K_d = m \neq 2^n$ (bộ đếm modul bất kỳ) có thể thực hiện bằng cách:

– Thiết kế trực tiếp (bỏ qua những trạng thái không sử dụng).

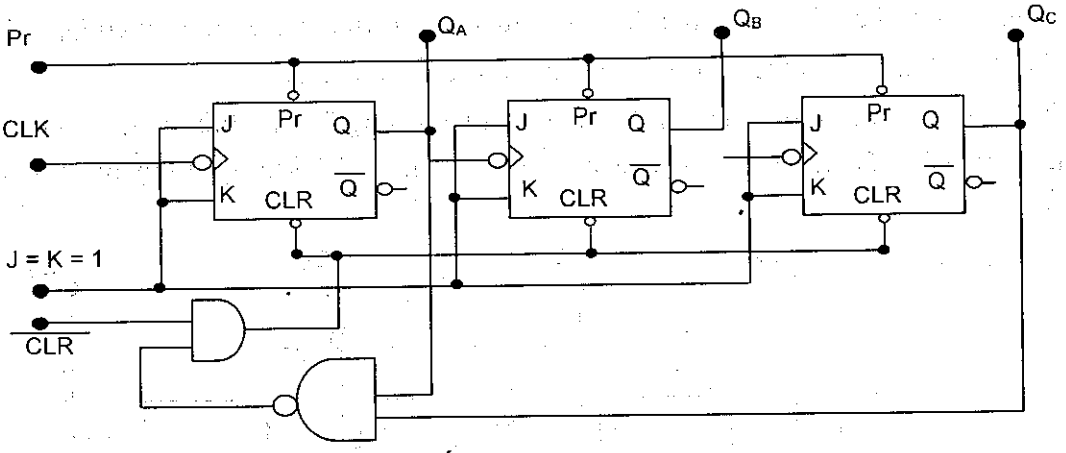
– Đặt lại trạng thái: Sử dụng bộ đếm có sẵn, để cho bộ đếm chuyển sang trạng thái m rồi dùng trạng thái này tạo tín hiệu điều khiển để xoá tất cả các FF về trạng thái 0 (trạng thái ban đầu).

Thiết kế trực tiếp đã được giới thiệu trong phần trước, trong phần này sẽ trình bày phương pháp thiết kế bộ đếm trên cơ sở dùng bộ đếm có sẵn. Với kỹ thuật vi mạch phát triển như hiện nay, cách này được sử dụng ngày càng nhiều.

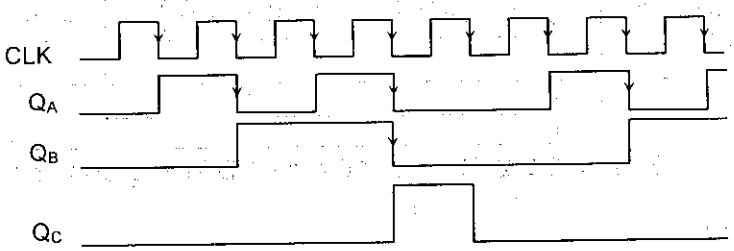
Ví dụ 5.2.5.1. Thiết kế bộ đếm có $K_d = 5$ dùng phương pháp đặt lại trạng thái.

Sử dụng bộ đếm có $K_d = 8$ và dập đi 3 trạng thái không mong muốn. Với bộ đếm modul 8, khi đếm hết xung thứ 8 thì $Q_CQ_BQ_A = 000$. Muốn cho đếm theo modul 5 thì từ mã nhị phân tương ứng với xung thứ 5 có $Q_CQ_BQ_A = 101$ phải được "dập" đi hai số 1 để cũng có 000. Muốn vậy ta phải đưa hai lối ra Q_C và Q_A qua một cổng NAND hai lối vào. Lối ra của NAND cùng với xung xoá CLR đưa qua một cổng AND hai lối vào, đầu ra của cổng AND được nối với các đầu vào xoá của các FF, bộ đếm sẽ đếm

lại từ đầu khi hết xung thứ 5. Sơ đồ mạch đếm được cho trên hình 5.2.5.1 và giản đồ xung cho trên hình 5.2.5.2.



Hình 5.2.5.1. Bộ đếm $K_đ = 5$



Hình 5.2.5.2. Giản đồ xung bộ đếm tiến $K_đ = 5$

5.2.6. Đếm 10 mã BCD

Có thể thiết kế bộ đếm modul 10 theo phương pháp trực tiếp hoặc đặt lại trạng thái.

a) Phương pháp trực tiếp

+ Đếm không đồng bộ:

Xét bảng trạng thái của bộ đếm 10 mã BCD trên hình 5.2.6.1.

Xung nhịp	Q_D	Q_C	Q_B	Q_A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	0	0	0	0

Hình 5.2.6.1. Bảng trạng thái của bộ đếm 10 mã BCD

Ta thấy:

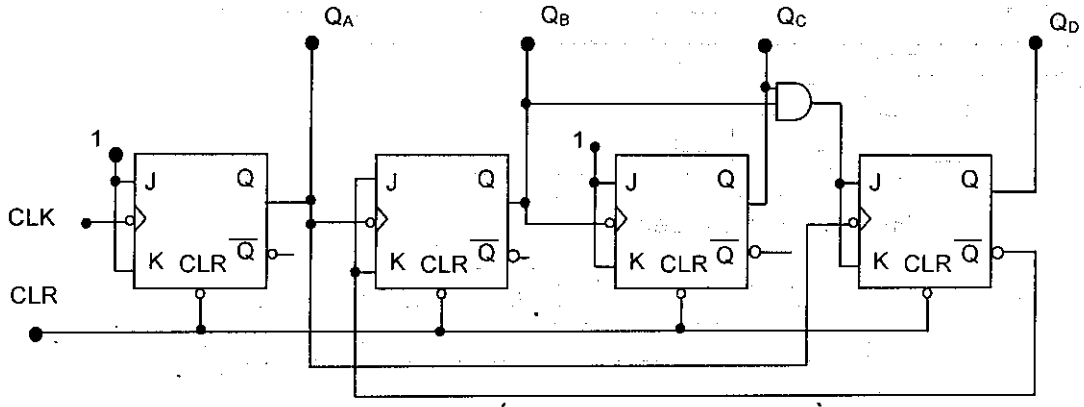
– Trigơ A thay đổi trạng thái khi có xung nhịp tác động.

– Trigơ B chỉ thay đổi trạng thái khi Q_A chuyển từ 1 xuống 0 đồng thời có $Q_D = 0$.

– Trigơ C thay đổi trạng thái khi Q_B chuyển từ 1 xuống 0.

– Trigơ D thay đổi trạng thái khi Q_A chuyển từ 1 xuống 0 đồng thời có $Q_B = Q_C = 1$.

Từ nhận xét trên ta có sơ đồ của bộ đếm 10 mã BCD không đồng bộ như hình 5.2.6.2.



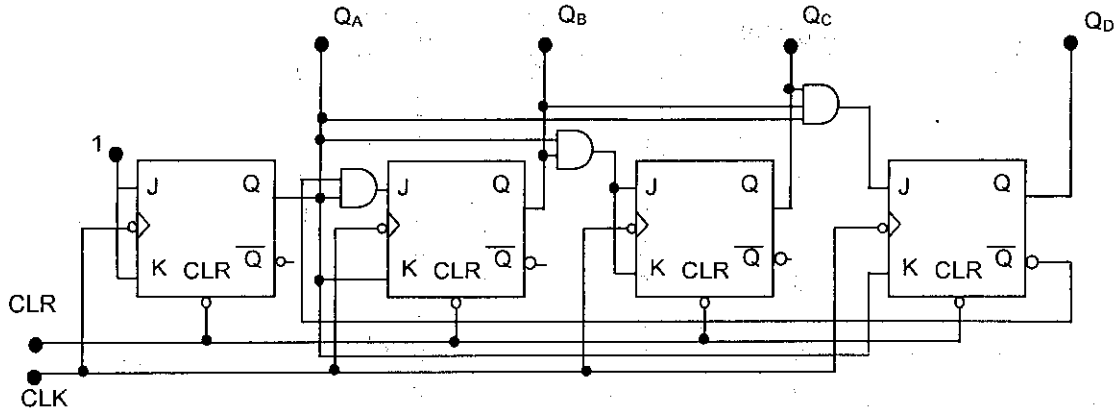
Hình 5.2.6.2. Bộ đếm 10 mã BCD không đồng bộ

+ Đếm đồng bộ:

Thiết kế theo các bước đã trình bày trong mục 5.2.3. Tuy nhiên, ở đây ta thấy có 6 tổ hợp mã nhị phân không xuất hiện ở đầu ra của bộ đếm (tương ứng từ 1010 đến 1111). Vì vậy, khi tối thiểu hoá các hàm cho giá trị của các hàm tương ứng với sáu tổ hợp đó là không xác định. Kết quả sau khi tối thiểu hoá:

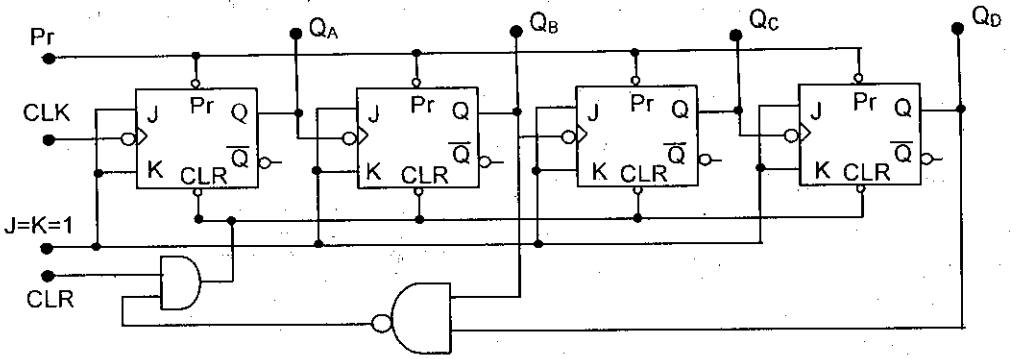
$$J_A = K_A = 1 \; ; \; J_B = \overline{Q_D} Q_A \; ; \; K_B = A \; ; \; J_C = K_C = Q_B Q_A \; ; \; J_D = Q_C Q_B Q_A \; ; \; K_D = A.$$

Sơ đồ của bộ đếm 10 mã BCD đồng bộ được cho trên hình 5.2.6.3.



Hình 5.2.6.3. Bộ đếm đồng bộ mã BCD

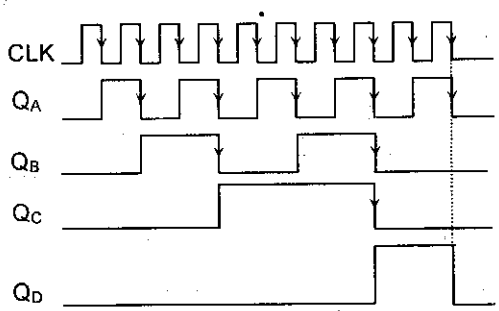
b) Phương pháp đặt lại trạng thái



Hình 5.2.6.4. Bộ đếm mã BCD đặt lại trạng thái

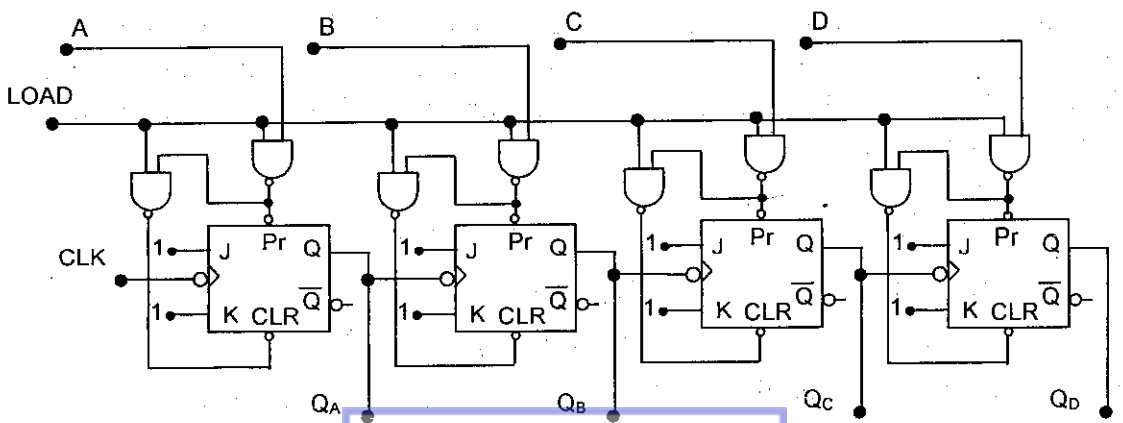
Với bộ đếm modul 16 (đồng bộ hoặc không đồng bộ), khi hết xung thứ 16 thì $Q_DQ_CQ_BQ_A$ = 0000. Muốn có bộ đếm modul 10 thì đến xung thứ 10 ta có $Q_DQ_CQ_BQ_A$ = 0000. Chúng ta biết với bộ đếm modul 16 đến xung thứ 10 thì $Q_DQ_CQ_BQ_A$ = 1010, để có được $Q_DQ_CQ_BQ_A$ = 0000 thì phải dập 2 số "1" đi. Muốn vậy, ta phải đưa hai lối ra Q_D và Q_B (có giá trị là 1 cần dập đi) vào hai lối vào của một cổng NAND. Sơ đồ bộ đếm BCD không đồng bộ như hình 5.2.6.4.

Giản đồ xung được cho trên hình 5.2.6.5.



Hình 5.2.6.5. Giản đồ xung bộ đếm mã BCD

5.2.7. Mạch đếm đặt trước



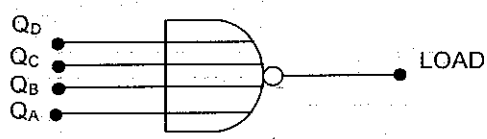
Hình 5.2.7.1. Bộ đếm đặt trước

Ngoài những bộ đếm nói trên ta còn gặp những bộ đếm bắt đầu từ một số đặt trước bất kỳ cho đến hết modul của nó. Sơ đồ của một bộ đếm đặt trước được cho trên hình 5.2.7.1.

Khi "LOAD" = 1: Mạch đếm xác lập số đặt trước.

Ví dụ: ABCD = 0110 thì $Q_DQ_CQ_BQ_A = 0110$.

Khi "LOAD" = 0: Mạch đếm tiếp tục theo xung nhịp: 0111, 1000,..., 1111, 0000.



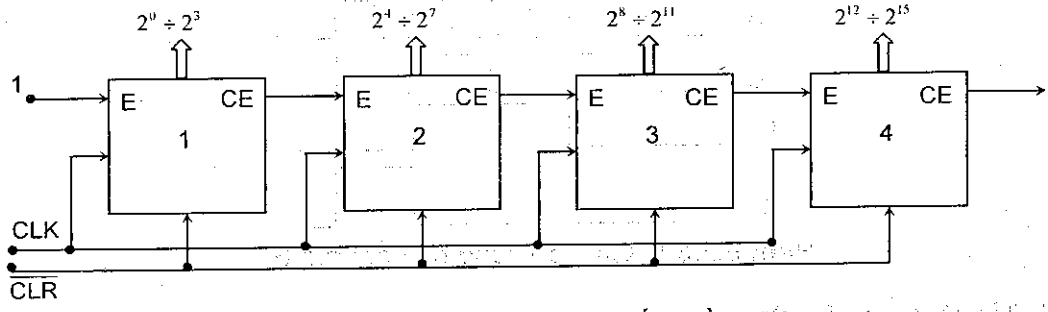
Hình 5.2.7.2

Để mạch đếm quay trở lại đếm từ trạng thái đặt trước (0110, 0111,...,1111, 0110,...) thì các đầu ra $Q_DQ_CQ_BQ_A$ cần được nối với các đầu vào của cổng NOR 4 đầu vào và lối ra của cổng NOR này điều khiển đường LOAD như hình 5.2.7.2.

5.2.8. Tăng dung lượng của bộ đếm

Khi cần đếm số lượng xung lớn hơn 15, người ta không kéo dài thêm trigơ vào sau trigơ thứ 4 mà ghép từng nhóm 4 trigơ. Việc ghép liên tiếp các bộ đếm 4 bit phải dùng tín hiệu nhớ E và CE, tín hiệu nhớ E và CE được tạo từ các mạch logic phụ.

Ví dụ 5.2.8.1. Xét bộ đếm dung lượng $(2^{16}-1)$ xung (16 bit) theo phương pháp đếm song song có sơ đồ khối cho ở hình 5.2.8.1.



Hình 5.2.8.1. Tăng dung lượng bộ đếm đồng bộ

Bộ đếm 16 bit gồm 4 bộ đếm modul 16, các bộ đếm chỉ đếm khi có tín hiệu E = 1 và tín hiệu CE chỉ bằng 1 khi cả 4 lối ra của bộ đếm modul 16 đều ở mức cao. Như vậy, khi bộ đếm thứ 1 đếm đến xung thứ 15 thì tín hiệu CE = 1 và nó kích thích lối vào E của bộ đếm thứ 2,...

Có nghĩa là:

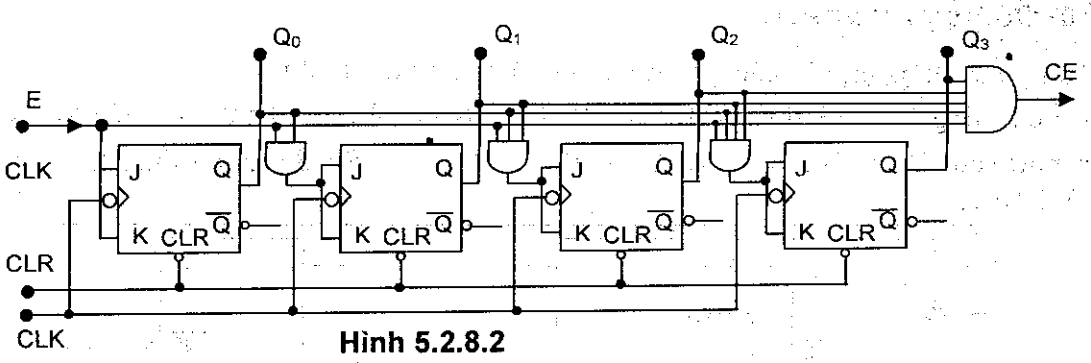
Sau 1 xung nhịp thì bộ đếm thứ nhất tăng thêm 1 đơn vị.

Sau 2^4 xung nhịp thì bộ đếm thứ 2 tăng thêm 1 đơn vị.

Sau $2^8(2^4.2^4)$ xung nhịp thì bộ đếm thứ 3 tăng thêm 1 đơn vị.

Sau $2^{12}(2^4.2^4.2^4)$ xung nhịp thì bộ đếm thứ 4 tăng thêm 1 đơn vị.

Sơ đồ bộ đếm 4 bit có logic tạo nhớ cho trên hình 5.2.8.2.



Hình 5.2.8.2

Tín hiệu điều khiển J, K và tín hiệu nhớ CE cần có đầu vào E khống chế:

$CE = E. Q_0 Q_1 Q_2 Q_3$

$K_3 = J_3 = E. Q_0 Q_1 Q_2$

$K_2 = J_2 = E. Q_0 Q_1$

$K_1 = J_1 = E. Q_0$

$K_0 = J_0 = E$

5.2.9. Các vi mạch đếm nhị phân và đếm 10 mã BCD

Ở các tiết trước chúng ta đã khảo sát cấu trúc hoạt động của các bộ đếm nhị phân và đếm 10 mã BCD. Các bộ đếm này được tạo thành từ các trigơ rời rạc. Công nghệ chế tạo các vi mạch logic hiện nay đã phát triển ở trình độ rất cao. Đã có nhiều vi mạch mức độ tích hợp trung bình và cỡ lớn (MSI, LSI) giá thành các vi mạch này rất rẻ. Trên thị trường hiện nay có rất nhiều loại vi mạch cỡ trung bình MSI là các bộ đếm hoàn chỉnh. Sau đây giới thiệu một số mạch đếm họ TTL và CMOS.

Các vi mạch đếm họ TTL:

Các vi mạch đếm nhị phân 4 bit: 7493, 74LS293, 74161, 74163, 74193, 74191

Các vi mạch đếm 10 mã BCD: 7490, 7492, 74160, 74190

7490 : mạch đếm 10 không đồng bộ

7492 : mạch đếm 10 đồng bộ mã BCD có các mode điều khiển.

7493 : mạch đếm 16 không đồng bộ mã nhị phân

74160 : mạch đếm 10 đồng bộ đặt trước, theo mã BCD

74161 : mạch đếm 16 đồng bộ mã nhị phân

74190 : mạch đếm 10 đồng bộ theo mã BCD

74191 : mạch đếm 16 đồng bộ có các mode điều khiển.

Các vi mạch đếm thuộc họ CMOS:

CD4029 : mạch đếm tiến lùi mã nhị phân và mã BCD

74C193 : mạch đếm nhị phân đồng bộ 4 bit tiến/lùi

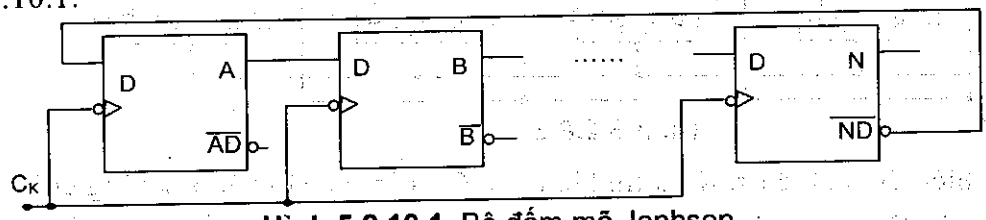
74C192 : mạch đếm 10 đồng bộ tiến/ lùi

74C90: mạch đếm 10 không đồng bộ mã BCD.

5.2.10. Bộ đếm Johnson

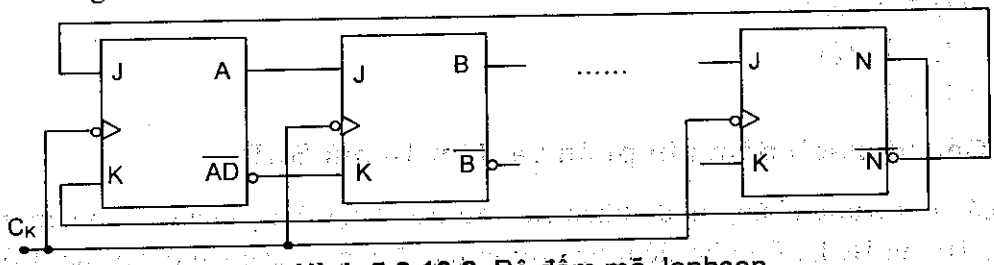
Dùng mã Johnson, với cách thiết kế như đã trình bày, ta được:

– Nếu dùng D – FF thì các FF: A, B, C,...,N có phương trình các đầu vào điều khiển như sau: $D_A = \overline{N}$, $D_B = A$, $D_C = B$ ..., $D_N = M$. Sơ đồ mạch thực hiện cho ở hình 5.2.10.1.



Hình 5.2.10.1. Bộ đếm mã Jonhson

– Nếu dùng JK – FF có sơ đồ như hình 5.2.10.2.



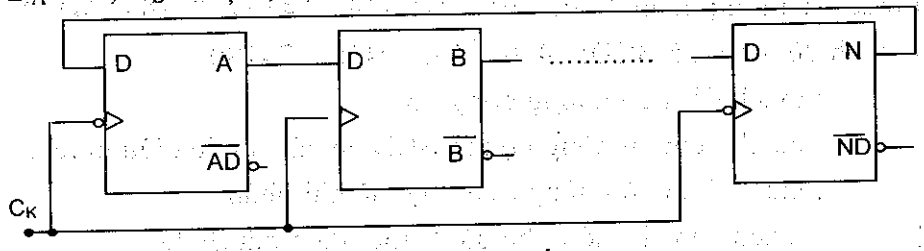
Hình 5.2.10.2. Bộ đếm mã Jonhson

5.2.11. Bộ đếm vòng

Dùng mã vòng, với phương pháp thiết kế như đã trình bày ở phần trước, ta có:

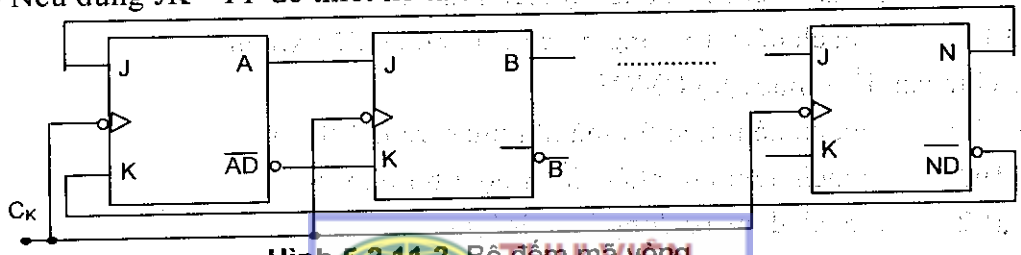
– Nếu dùng D – FF để thiết kế ta có sơ đồ như hình 5.2.11.1.

Với: $D_A = N$, $D_B = A$, ..., $D_N = M$.



Hình 5.2.11.1. Bộ đếm mã vòng

– Nếu dùng JK – FF để thiết kế ta có sơ đồ như hình 5.2.11.2



Hình 5.2.11.2. Bộ đếm mã vòng

5.3. CÁC BỘ GHI DỊCH (Shift Register)

Bộ ghi dịch còn gọi là thanh ghi dịch là các phần tử không thể thiếu được trong CPU, trong các hệ vi xử lý,... Nó có khả năng ghi giữ và dịch thông tin (sang phải hoặc sang trái).

Bộ ghi dịch cấu tạo từ một dãy phần tử nhớ đơn bit (trigơ) được mắc liên tiếp với nhau và một số cửa logic cơ bản hỗ trợ.

Muốn ghi và truyền một từ nhị phân n bit ta cần n phần tử nhớ (n trigơ). Trong các bộ ghi dịch thường dùng các trigơ đồng bộ như trigơ RST, trigơ JK, trigơ D.

Thông thường người ta hay dùng các trigơ D hoặc các trigơ khác nhưng mắc theo kiểu trigơ D để tạo thành các bộ ghi.

Có hai cách ghi:

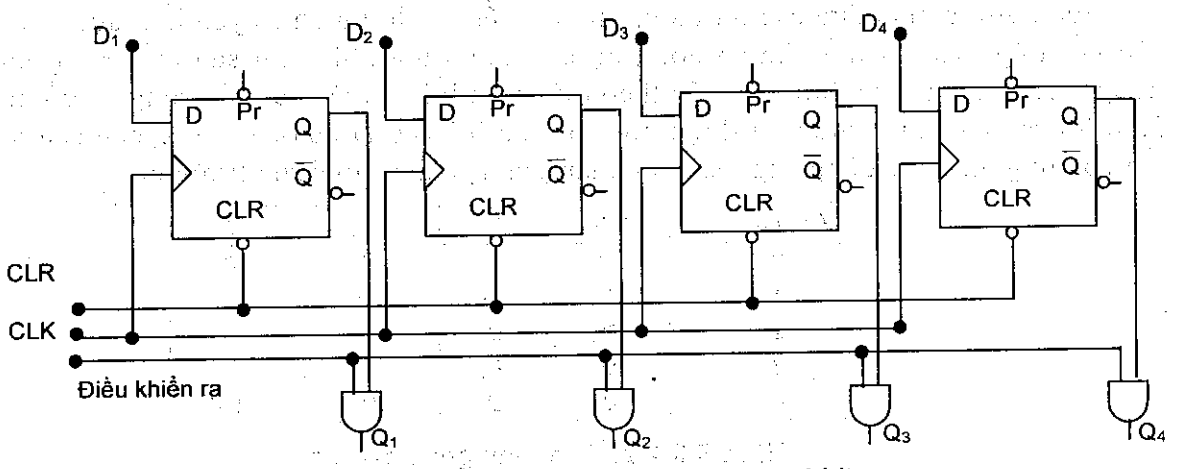
– ***Ghi song song:*** Các bit của từ nhị phân được ghi đồng thời cùng một lúc vào bộ ghi.

– ***Ghi nối tiếp:*** Các bit của từ nhị phân được đưa vào bộ ghi một cách tuần tự theo thứ tự của từ nhị phân.

5.3.1. Bộ ghi song song

Sơ đồ bộ ghi song song 4 bit cho trên hình 5.3.1.1.

Trong sơ đồ 5.3.1.1 người ta thêm 1 mạch điều khiển ra dùng 4 cổng AND 2 lối vào.



Hình 5.3.1.1. Bộ ghi song song 4 bit

Hoạt động của sơ đồ trên hình 5.3.1.1 như sau:

Trước tiên dùng xung xoá CLR = 0 để xoá, lối ra $Q_1Q_2Q_3Q_4 = 0000$

Các số liệu cần ghi được đưa vào lối vào $D_1D_2D_3D_4$.

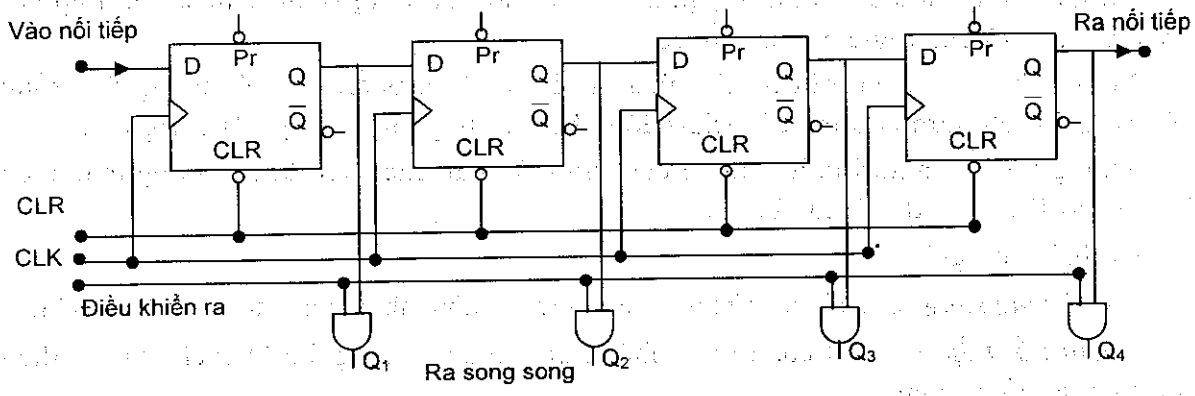
Khi có xung điều khiển ghi đưa vào lối vào CLK, dữ liệu được nạp vào bộ nhớ song song và cho lối ra song song $Q_1Q_2Q_3Q_4 = D_1D_2D_3D_4$.

Mỗi lối ra Q được đưa vào 1 lối vào của các cửa AND. Muốn cho dữ liệu ra thẳng lối ra thì lối "Điều khiển ra" phải bằng 1. Nếu chưa muốn cho dữ liệu ra lối ra thì để "Điều khiển ra" bằng 0.

5.3.2. Bộ ghi nối tiếp

a) Bộ ghi nối tiếp dịch phải có các lối ra song song và ra nối tiếp

Bộ ghi nối tiếp có thể dịch phải, dịch trái và cho ra song song hoặc ra nối tiếp. Trên hình 5.3.2.1 giới thiệu sơ đồ bộ ghi nối tiếp dịch phải có các lối ra song song và ra nối tiếp.



Hình 5.3.2.1. Bộ ghi nối tiếp dịch phải

Đây là sơ đồ chỉ có lối vào nối tiếp, còn lối ra cả song song và ra nối tiếp.

Khi cho một xung kim âm tác động vào lối vào xoá, các lối ra Q của cả 4 trigơ trong bộ ghi đều ở trạng thái 0.

Muốn ghi ta phải đưa các bit thông tin nối tiếp về thời gian truyền lần lượt vào lối vào nối tiếp theo sự điều khiển đồng bộ của các xung nhịp. Cứ sau mỗi xung nhịp trạng thái của trigơ lại được xác lập theo thông tin lối vào D của nó. Trong sơ đồ hình 5.38 lối ra của trigơ trước lại được nối với vào lối vào D của trigơ sau nên sau mỗi lần có xung nhịp tác động trigơ sau lại nhận giá trị của trigơ đứng trước nó.

Xung nhịp	Q_1	Q_2	Q_3	Q_4
0	0	0	0	0
1	D_4	0	0	0
2	D_3	D_4	0	0
3	D_2	D_3	D_4	0
4	D_1	D_2	D_3	D_4

Hình 5.3.2.2. Quá trình ghi thông tin

Giả sử ta có 4 bit số liệu $D_1D_2D_3D_4$ được truyền liên tiếp tới lối vào của bộ ghi trong đó bit D_4 đến trước nhất. Quá trình ghi thông tin diễn ra như trên hình 5.3.2.2.

Sau 4 xung nhịp thì thông tin được nạp xong, muốn đưa dữ liệu ra ở các lối ra song song ta đặt mức 1 ở lối "Điều khiển ra", lối ra của các cửa AND ở lối ra song song sẽ được xác lập theo trạng thái Q_1, Q_2, Q_3, Q_4 của các trigơ trong bộ ghi. Trong cách điều khiển dữ liệu ra song song này thông tin trong bộ ghi vẫn được duy trì.

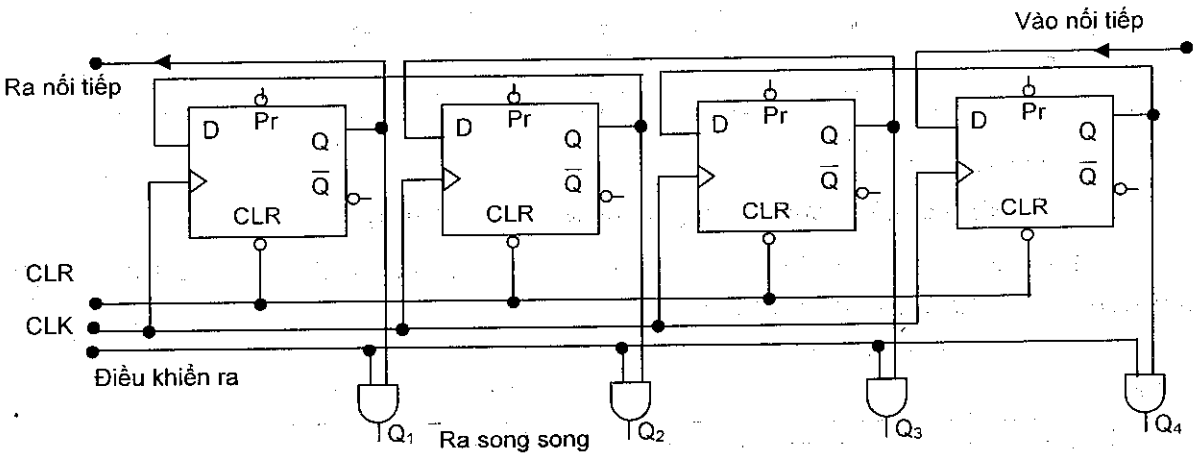
Để điều khiển dữ liệu ra nối tiếp, ta phải tác động một nhóm 4 xung nhịp ở lối vào CLK (điều khiển ghi). Sau 4 xung nhịp tác động 4 bit dữ liệu lần lượt được đưa ra khỏi bộ ghi.

Như vậy, quá trình điều khiển ghi nối tiếp 4 bit mới cũng là quá trình đưa 4 bit dữ liệu cũ ra khỏi bộ ghi qua lối ra nối tiếp.

b) Bộ ghi nối tiếp dịch trái có các lối ra song song và lối ra nối tiếp

Bộ ghi nối tiếp dịch trái có các lối ra song song và lối ra nối tiếp được trình bày trên hình 5.3.2.3. Cấu trúc của bộ ghi này cũng tương tự như bộ ghi dịch phải hình 5.3.2.1, nó chỉ khác trật tự sắp xếp các trigơ trong bộ ghi. Trigơ 4 lại là trigơ đầu, trigơ 1 là trigơ cuối.

Quá trình điều khiển xoá, điều khiển ghi vào và đưa dữ liệu ra hoàn toàn tương tự như bộ ghi dịch phải hình 5.3.2.1.



Hình 5.3.2.3. Bộ ghi nối tiếp dịch trái

Ví dụ, ta có một chuỗi dữ liệu D_1 D_2 D_3 D_4 được truyền đến lối vào của bộ ghi theo trình tự bit D_1 đến trước nhất. Quá trình ghi dịch 4 bit dữ liệu đối với bộ đếm này diễn ra như trên hình 5.3.2.4.

Xung nhịp	Q_1	Q_2	Q_3	Q_4
0	0	0	0	0
1	0	0	0	D_1
2	0	0	D_1	D_2
3	0	D_1	D_2	D_3
4	D_1	D_2	D_3	D_4

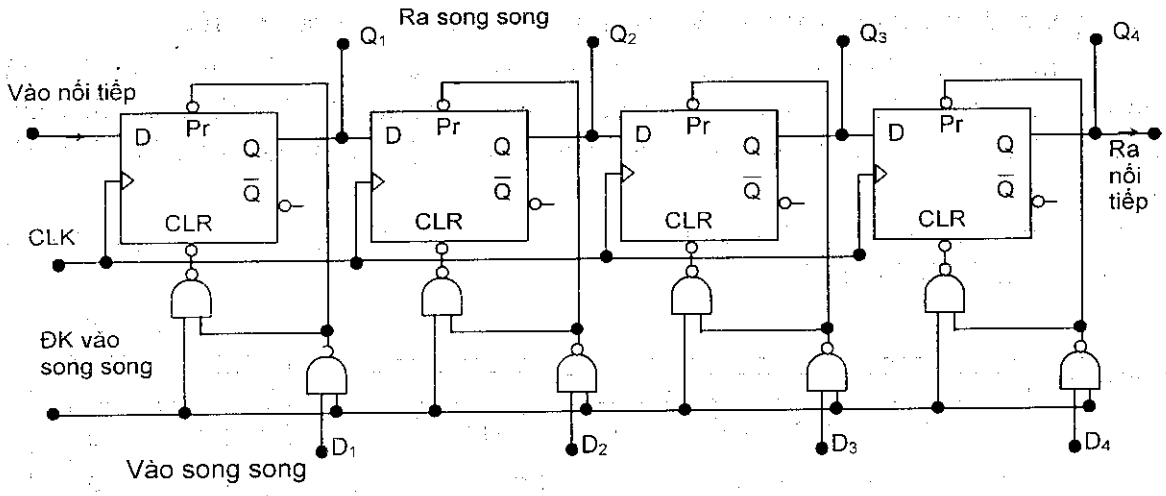
Hình 5.3.2.4

5.3.3. Mạch vừa ghi nối tiếp dịch phải, vừa ghi song song

Có thể dùng $\overline{Pr}$ và $\overline{CLR}$ bổ sung 1 đầu vào song song cho thanh ghi dịch, sơ đồ được cho trên hình 5.3.3.1.

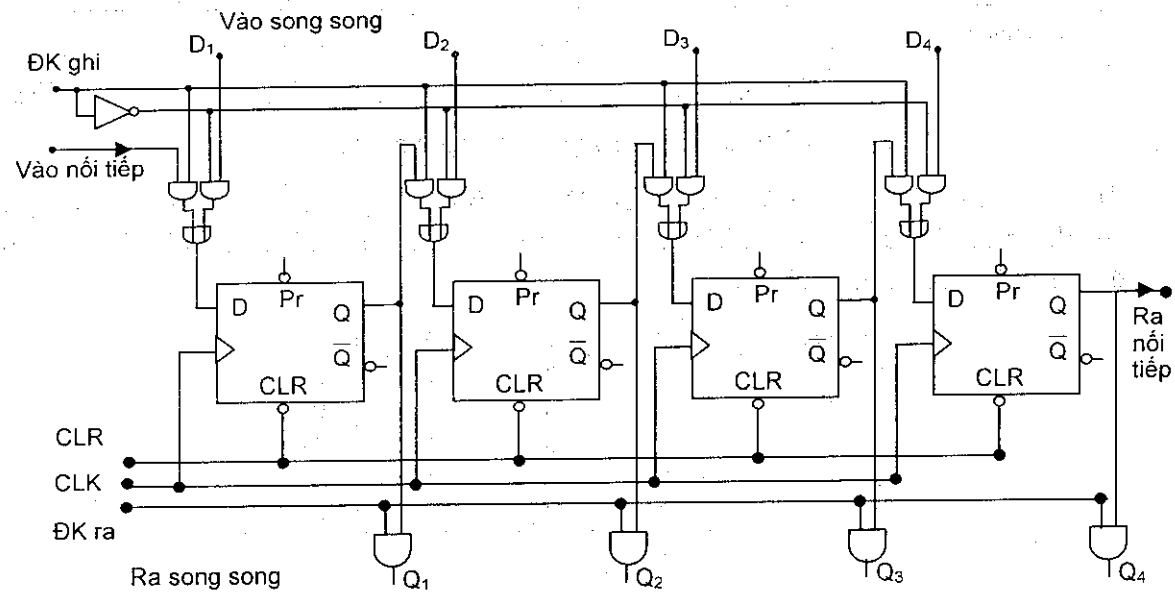
Ở hình 5.3.3.1: Để nhập giá trị vào song song, "ĐK vào song song" phải bằng 1. Sử dụng 8 cổng NAND 2 đầu vào, khi cả hai đầu vào cùng bằng 1 thì đầu ra bằng 0. Hai đầu vào $\overline{Pr}$ và $\overline{CLR}$ tích cực bởi mức thấp, được tác động bởi tín hiệu ở đầu ra các cổng NAND.

Để nhập giá trị vào nối tiếp, "ĐK vào song song" bằng 0 làm cho 8 đầu ra của 8 cổng NAND bằng 1 đóng kín các đường mạch vào song song.



Hình 5.3.3.1. Mạch vừa ghi nối tiếp vừa ghi song song

Có thể dùng thêm các phần tử logic để điều khiển quá trình ghi song song và ghi nối tiếp, sơ đồ được cho trên hình 5.3.3.2.



Hình 5.3.3.2. Mạch vừa ghi nối tiếp vừa ghi song song

Ở hình 5.3.3.2: "ĐK ghi" = 1 thì nạp vào nối tiếp, "ĐK ghi" = 0 thì nạp vào song song.

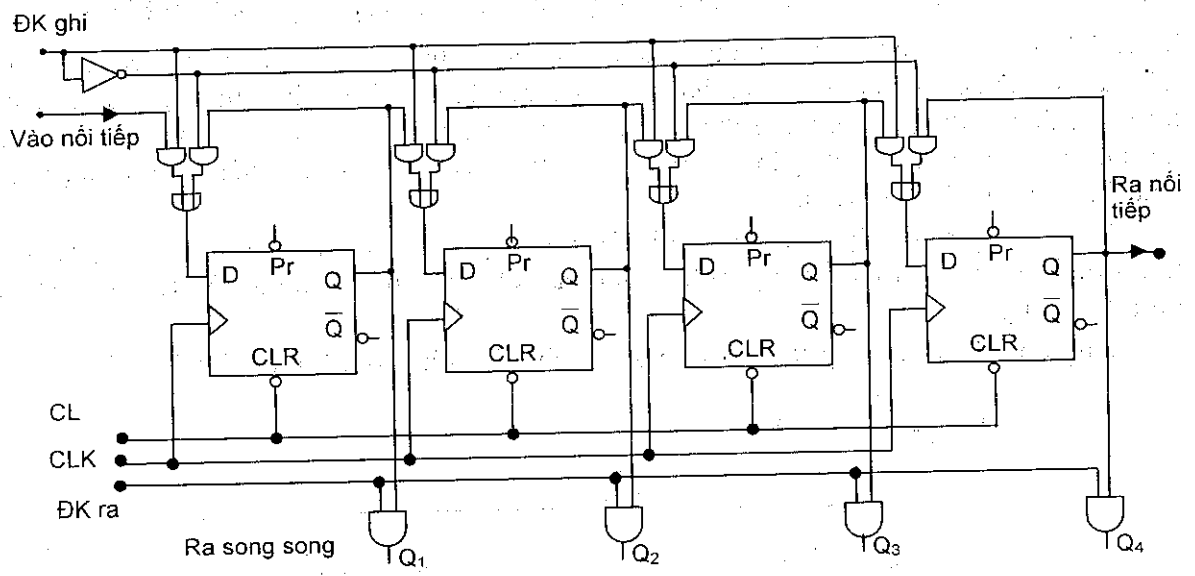
5.3.4. Bộ ghi nối tiếp thông tin có thể lưu trữ

Sơ đồ mạch cho trên hình 5.3.4.1.

Khi " ĐK ghi" = 1: cho phép ghi dịch nối tiếp, sau một nhóm xung nhịp ta ghi xong một từ nhị phân.

Khi "ĐK ghi" = 0: Q_i được nối với D_i, lúc này có xung nhịp tác động thì trạng

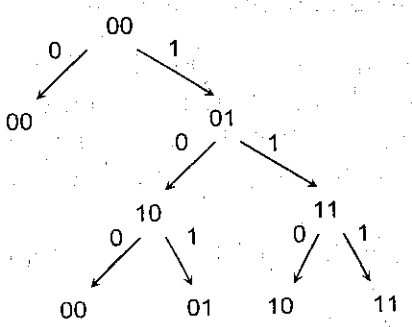
thái ở đầu ra của các FF vẫn giữ nguyên, nghĩa là thông tin được lưu giữ mãi. Muốn ghi tiếp lại cho "ĐK ghi" = 1.



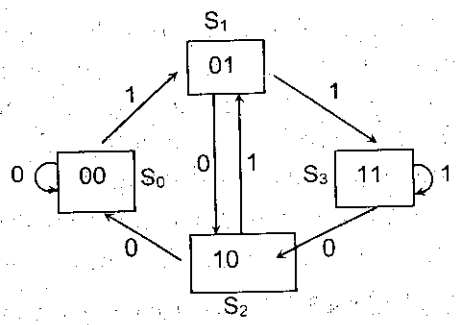
Hình 5.3.4.1. Bộ ghi nối thiếp thông tin có thể lưu

5.3.5. Đồ hình tổng quát của bộ ghi dịch

Bảng chuyển đổi trạng thái của thanh ghi dịch trái hai bit được chỉ ra trong hình 5.3.5.1. Nhờ bảng này ta có thể biết được tất cả các chuyển đổi trạng thái của mạch.



Hình 5.3.5.1. Bảng chuyển đổi trạng thái



Hình 5.3.5.2. Đồ hình tổng quát

Ví dụ 5.3.5.1. Nếu trạng thái ban đầu của bộ ghi dịch trái 2 bit là 00 thì nó có thể chuyển đến một trong hai trạng thái tiếp theo: trạng thái 00 nếu đầu vào D của FF đầu tiên (trigơ có trọng số nhỏ nhất) là 0 và là 01 nếu đầu vào D của FF đầu tiên là 1. Tương tự, nếu trạng thái ban đầu của bộ ghi dịch là 01 thì nó sẽ chuyển đến trạng thái tiếp theo là 10 hoặc 11.

Từ bảng chuyển đổi trạng thái ở hình 5.3.5.1 có thể biểu diễn ở dạng đồ hình tổng quát như trong hình 5.3.5.2. Các giá trị số ghi trên các cung chính là giá trị của hàm hồi tiếp (cũng chính là đầu vào của FF đầu tiên) ứng với sự chuyển đổi trạng thái đó.

Từ đồ hình ta nhận thấy bộ ghi dịch sẽ bị "khoá" ở trạng thái 00 nếu hàm hồi tiếp là 0, hoặc ở trạng thái 11 nếu hàm hồi tiếp là 1.

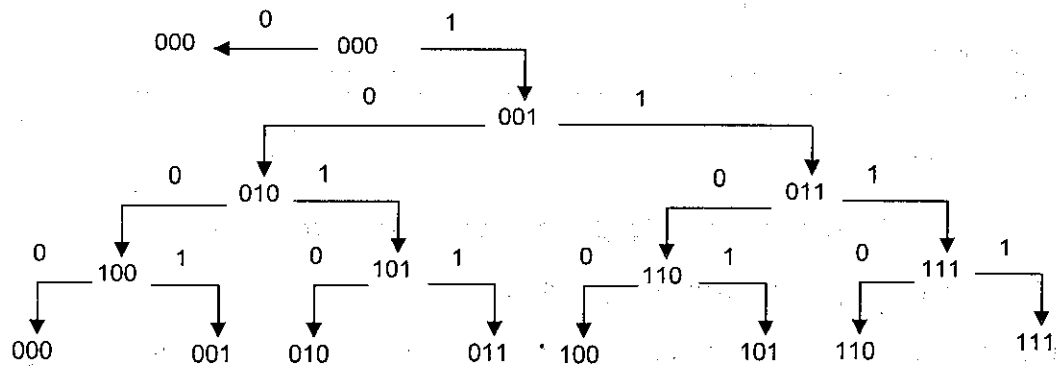
Với cách làm tương tự ta có đồ hình tổng quát của bộ ghi dịch có số bit bất kỳ.

Nhận xét: Từ đồ hình tổng quát của bộ ghi dịch luôn luôn xác định được trạng thái sẽ chuyển biến tới của bộ ghi dịch ứng với giá trị tương ứng của hàm hồi tiếp.

Ví dụ 5.3.5.2. Nếu trạng thái ban đầu của bộ ghi dịch 3 bit là 010 (S_2), thì ứng với hàm hồi tiếp là 1 mạch sẽ chuyển tới trạng thái 101 (S_5), nếu hàm hồi tiếp là 0 thì mạch sẽ chuyển tới trạng thái 100(S_4). Nếu trạng thái ban đầu của bộ ghi dịch là 110 (S_6) thì ứng với hàm hồi tiếp là 1 thì mạch sẽ chuyển tới trạng thái 101 (S_5), nếu hàm hồi tiếp là 0 thì mạch sẽ chuyển tới trạng thái 100(S_4),...

Ta thấy vì bộ ghi dịch là dịch trái nên bit có ý nghĩa nhất sẽ bị mất đi và bit hồi tiếp được đưa vào vị trí bit ít ý nghĩa nhất sao cho số bit là không đổi khi có 1 xung nhịp tác động.

Bảng chuyển đổi trạng thái của bộ ghi dịch 3 bit như trên hình 5.3.5.3.



Hình 5.3.5.3. Bảng chuyển đổi trạng thái của bộ ghi dịch 3 bit

Ví dụ 5.3.5.3. Nếu trạng thái ban đầu của bộ ghi dịch 4 bit là 0011 (S_3) thì ứng với hàm hồi tiếp là 1 thì mạch sẽ chuyển tới trạng thái 0111 (S_7), nếu hàm hồi tiếp là 0 thì mạch sẽ chuyển tới trạng thái 0110(S_6). Nếu trạng thái ban đầu là 0111 (S_7) thì ứng với hàm hồi tiếp là 1 thì mạch sẽ chuyển tới trạng thái 1111 (S_{15}), nếu hàm hồi tiếp là 0 thì mạch sẽ chuyển tới trạng thái 1110(S_6), v.v...

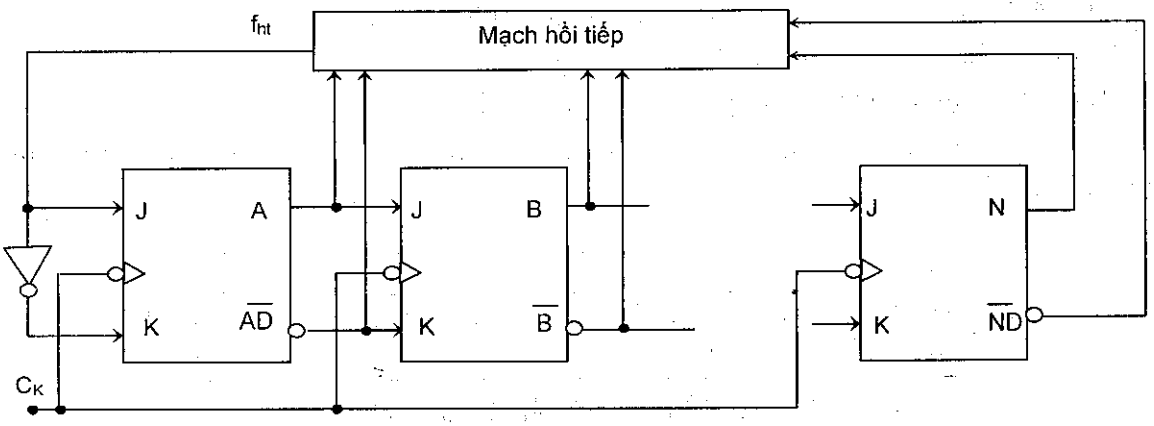
Đồ hình chuyển đổi trạng thái được sử dụng để thiết kế bộ đếm, bộ tạo dãy tín hiệu nhị phân,... như sẽ trình bày trong các phần sau.

5.3.6. Thiết kế bộ đếm dùng bộ ghi dịch

a) Sơ đồ khối

Từ đồ hình tổng quát của bộ ghi dịch ta nhận thấy rằng: Xuất phát từ một trạng thái ban đầu nào đó, ứng với 1 dãy tín hiệu hồi tiếp $f_{ht} = D_A = J_A$ xác định, sẽ có một dãy xác định các trạng thái sẽ chuyển biến tới. Nếu dãy tín hiệu của f_{ht} được chọn thích hợp sao cho dãy chuyển biến trạng thái tạo thành một chu trình kín thì đồ hình trạng thái của mạch chính là đồ hình của bộ đếm.

Như vậy, bài toán thiết kế bộ đếm dùng bộ ghi dịch trở thành bài toán thiết kế hàm hồi tiếp cung cấp cho đầu vào của bộ ghi dịch sao cho ứng với hàm này các trạng thái của mạch sẽ chuyển biến theo chu trình kín, số trạng thái trong của chu trình bằng K_d.



Hình 5.3.6.1. Sơ đồ khối của bộ đếm dùng thanh ghi dịch có hàm hồi tiếp

Sơ đồ khối của bộ đếm dùng thanh ghi dịch có hàm hồi tiếp được chỉ ra ở hình 5.3.6.1.

Các FF được mắc với nhau thành một thanh ghi dịch n bit. Đầu ra của các FF gồm cả Q và $\overline{Q}$ được sử dụng để tạo hàm hồi tiếp đưa vào đầu vào điều khiển của thanh ghi (đầu vào điều khiển của FF đầu tiên: FF – A). Giữa đầu vào và đầu ra của từng FF có mối quan hệ sau:

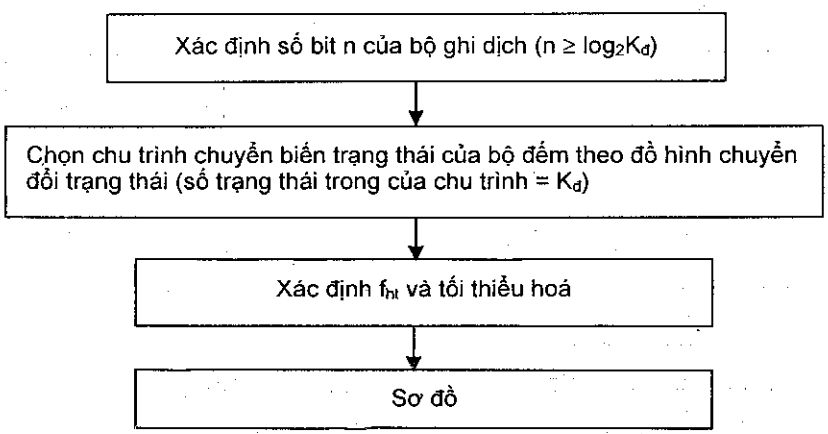
$$J_A = f_{ht}(A, B, C, \ldots, N); J_B = A; J_C = B; \ldots; J_N = M.$$

Mạch hồi tiếp này có giá trị "0" hay "1" và được đưa vào đầu vào của A – FF. Khi có xung nhịp sẽ thiết lập trạng thái của A – FF tương ứng.

Nếu bộ ghi dịch đang ở trạng thái N M....C B A = 0 0 ...0 0 1, trạng thái tiếp theo của bộ ghi dịch sẽ là 0 0 ... 0 1 0 hoặc 0 0 ...0 1 1 phụ thuộc vào giá trị của hàm hồi tiếp đưa vào đầu vào J của A – FF là 0 hay là 1.

b) Các bước thiết kế bộ đếm dùng thanh ghi dịch

Để thiết kế bộ đếm dùng bộ ghi dịch ta phải tiến hành theo các bước như ở hình 5.3.6.2.



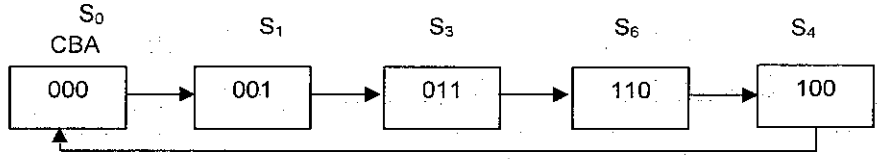
Hình 5.3.6.2. Các bước thiết kế bộ đếm dùng thanh ghi dịch

Ví dụ 5.3.6.1. Thiết kế bộ đếm 5 trạng thái dùng thanh ghi dịch.

Bước 1: Xác định số bit của bộ ghi dịch n = 3.

Bước 2: Chọn chu trình chuyển biến trạng thái.

Giả sử ta chọn dãy trạng thái theo chu trình kín có 5 trạng thái khác nhau như trong hình 5.3.6.3.



Hình 5.3.6.3. Đồ hình chuyển đổi trạng thái

Bước 3: Xác định hàm hồi tiếp.

Căn cứ vào dãy 5 trạng thái đã chọn ta có bảng trạng thái và giá trị của hàm hồi tiếp tương ứng hình 5.3.6.4. Tối thiểu hoá hàm bằng bảng Karnaugh được biểu diễn trên hình 5.3.6.5.

S	C	B	A	f_{ht}
S_0	0	0	0	1
S_1	0	0	1	1
S_3	0	1	1	0
S_6	1	1	0	0
S_4	1	0	0	0

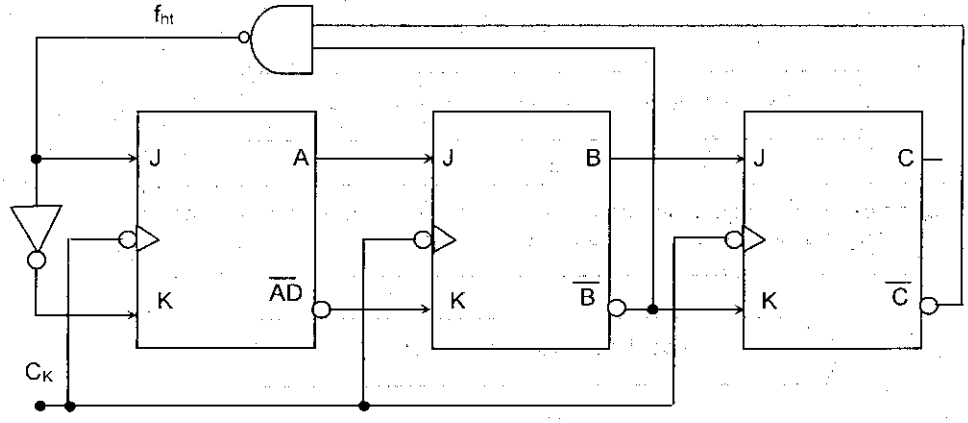
Hình 5.3.6.4. Bảng trạng thái

f_{ht} C \ BA	00	01	11	10
0	1	1	0	x
1	0	x	x	0

Hình 5.3.6.5. Tối thiểu hoá hàm

Phương trình logic của hàm hồi tiếp: $f_{ht} = \overline{DCA} + \overline{DCA} + \overline{DB}$

Sơ đồ bộ đếm 5 trạng thái dùng bộ ghi dịch cho trên hình 5.3.6.6.



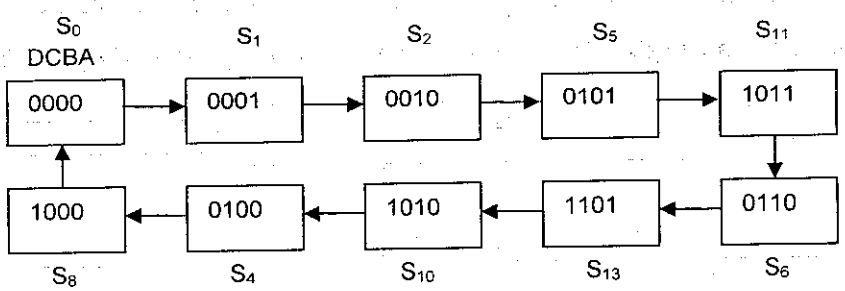
Hình 5.3.6.6. Sơ đồ bộ đếm 5 trạng thái dùng thanh ghi dịch

Ví dụ 5.3.6.2. Thiết kế bộ đếm 10 trạng thái dùng thanh ghi dịch

Bước 1: Xác định số bit của bộ ghi dịch n = 4.

Bước 2: Chọn chu trình chuyển biến trạng thái.

Giả sử ta chọn dãy trạng thái theo chu trình kín có 10 trạng thái khác nhau như trong hình 5.3.6.7.



Hình 5.3.6.7. Đồ hình chuyển đổi trạng thái

Bước 3: Xác định hàm hồi tiếp.

Căn cứ vào dãy 10 trạng thái đã chọn ta có bảng trạng thái và giá trị của hàm hồi tiếp tương ứng hình 5.3.6.8. Tối thiểu hoá hàm bằng bảng Karnaugh được biểu diễn trên hình 5.3.6.9.

S	D	C	B	A	f_{ht}
S_0	0	0	0	0	1
S_1	0	0	0	1	0
S_2	0	0	1	0	1
S_5	0	1	0	1	1
S_{11}	1	0	1	1	0
S_6	0	1	1	0	1
S_{13}	1	1	0	1	0
S_{10}	1	0	1	0	0
S_4	0	1	0	0	0
S_8	1	0	0	0	0

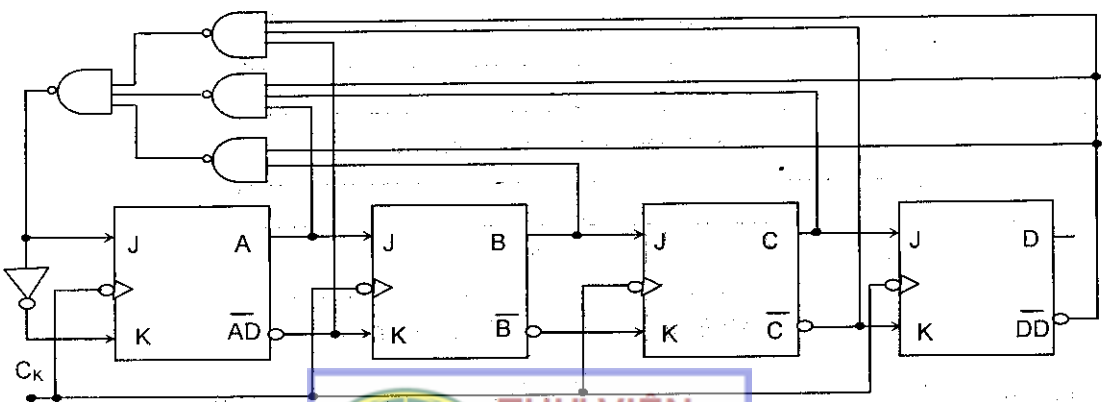
Hình 5.3.6.8. Bảng trạng thái

f_{ht} DC \ BA	00	01	11	10
00	1		X	1
01		1	X	1
11	X		X	X
10		X		

Hình 5.3.6.9. Tối thiểu hoá hàm

Phương trình logic của hàm hồi tiếp: $f_{ht} = \overline{DCA} + \overline{D}CA + \overline{D}B$

Sơ đồ bộ đếm 10 trạng thái dùng thanh ghi dịch ở hình 5.3.6.10.

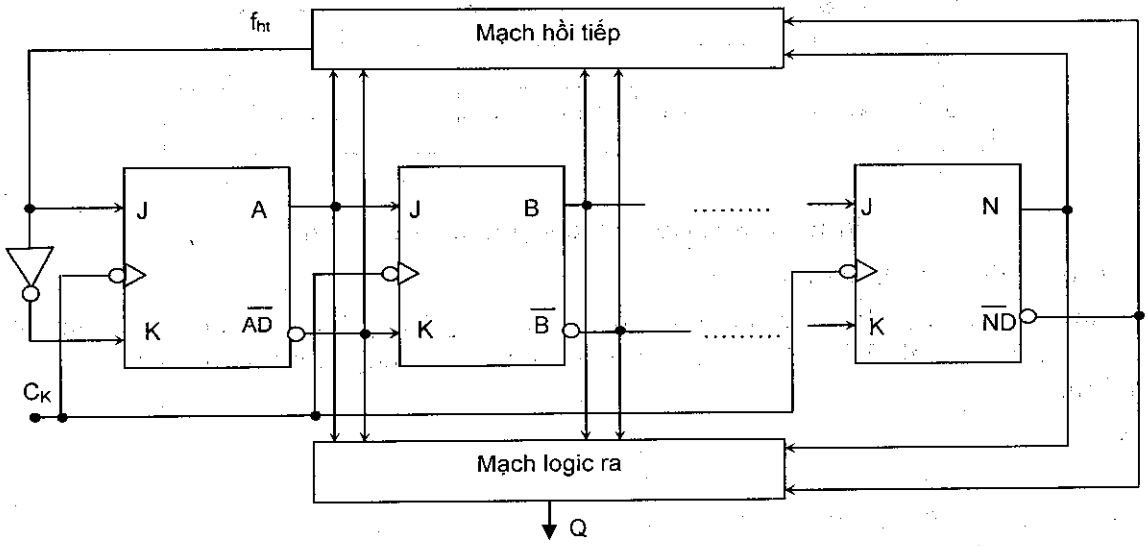


Hình 5.3.6.10. Sơ đồ bộ đếm 10 trạng thái dùng thanh ghi dịch

5.3.7. Tạo dãy tín hiệu tuần hoàn dùng thanh ghi dịch

Thanh ghi dịch có thể dùng để tạo tín hiệu tuần hoàn theo yêu cầu cho trước.

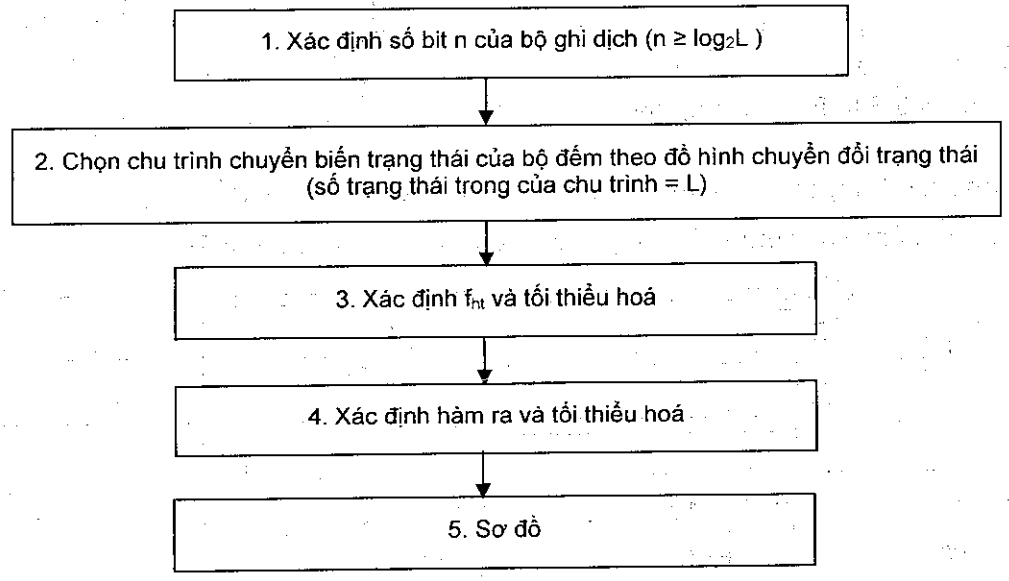
a) Sơ đồ khối của mạch tạo dãy tín hiệu tuần hoàn dùng bộ ghi dịch hình 5.3.7.1



Hình 5.3.7.1. Sơ đồ khối của mạch tạo dãy tuần hoàn dùng bộ ghi dịch

b) Các bước thiết kế mạch tạo dãy tín hiệu tuần hoàn dùng bộ ghi dịch

Bài toán thiết kế mạch tạo dãy tín hiệu nhị phân tuần hoàn có chiều dài L dùng bộ ghi dịch có lưu đồ thuật toán như hình 5.3.7.2.



Hình 5.3.7.2. Lưu đồ thuật toán

Các bước 1,2,3 tương tự các bước thiết kế bộ đếm dùng thanh ghi dịch.

Ví dụ 5.3.7.1. Xây dựng mạch tạo dãy tín hiệu nhị phân tuần hoàn 1 – 1 – 0 – 0 – 1 – 1 – 0 – 0.

Bước 1: Dãy tín hiệu tuần hoàn có chiều dài bằng L = 8, số bit của bộ ghi dịch n = 3 ($\log_2 8 = 3$).

Bước 2: Chọn chu trình chuyển biến trạng thái như sau:

$$S_0 \rightarrow S_1 \rightarrow S_2 \rightarrow S_5 \rightarrow S_3 \rightarrow S_7 \rightarrow S_6 \rightarrow S_4 \rightarrow S_0$$

S	C	B	A	f_{ht}	Q_A
S_0	0	0	0	1	1
S_1	0	0	1	0	1
S_2	0	1	0	1	0
S_5	1	0	1	1	0
S_3	0	1	1	1	1
S_7	1	1	1	0	1
S_6	1	1	0	0	0
S_4	1	0	0	0	0

Hình 5.3.7.3. Bảng giá trị của hàm hồi tiếp và hàm ra

Từ chu trình chuyển biến trạng thái ta có bảng giá trị của hàm hồi tiếp và hàm ra tương ứng như trên hình 5.3.7.3.

Bước 3: Xây dựng hàm hồi f_{ht} và tối thiểu hoá.

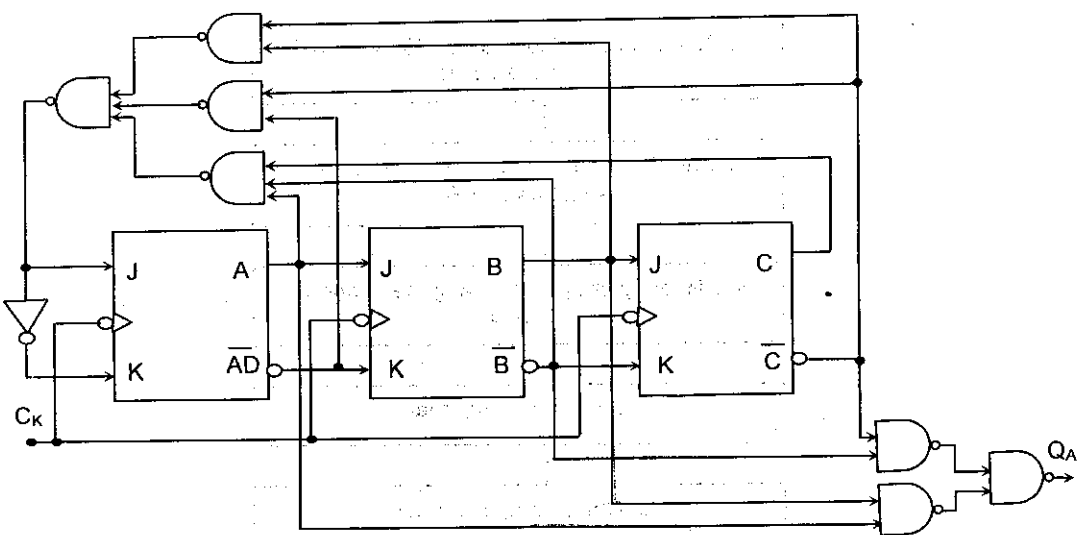
Từ hàm hồi tiếp xác định được như bảng hình 5.3.7.3, tối thiểu hoá hàm ta được:

$$f_{ht} = \overline{C}B + \overline{C}\overline{A} + C\overline{B}A$$

Bước 4: Xác định hàm logic ra Q_A và tối thiểu hoá.

Từ hàm ra như bảng hình 5.3.7.3, tối thiểu hoá hàm ta được: $Q_A = \overline{C}\overline{B} + BA$

Bước 5: Sơ đồ mạch trên hình 5.3.7.4.



Hình 5.3.7.4. Sơ đồ mạch

5.3.8. Một số vi mạch ghi dịch thường gặp

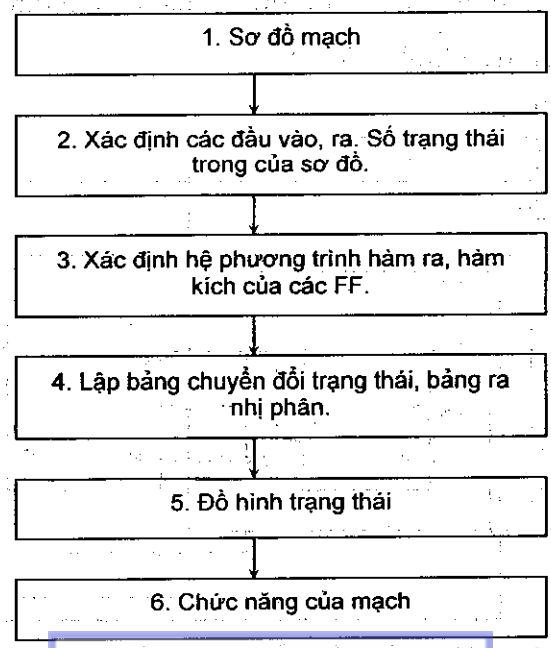
7495 : 4 bit, vào song song hoặc nối tiếp, ra song song hoặc nối tiếp.
74174 : 6 bit, vào song song, ra song song có đầu vào xoá.
74178 : 4 bit, vào song song hoặc nối tiếp, ra song song.
4731B : 64 bit, vào nối tiếp, ra nối tiếp.
74165 : 8 bit, vào song song, ra nối tiếp.
74164 : 8 bit, vào nối tiếp, ra song song có đầu vào xoá.
74194 : dịch trái, dịch phải, vào song song, ra song song.
74373 : 8 bit, vào song song, ra song song.
74374 : 8 bit, vào song song, ra song song.
54/74195 : 4bit, vào song song hoặc nối tiếp, ra song song có đầu vào xoá.
54/7491A : 8 bit, vào ra nối tiếp.
54/7496 : 5 bit, vào song song hoặc nối tiếp, ra song song có $\overline{CLR}$.
54/74199 : 8bit, vào song song hoặc nối tiếp, ra song song có $\overline{CLR}$.

5.4. MẠCH DÃY ĐỒNG BỘ

Mạch dãy đồng bộ là một mạch số bao gồm các mạch tổ hợp và các FF, hoạt động của mạch được đồng bộ bởi xung nhịp C_K. Trong thực tế, để giảm nhỏ công suất tiêu thụ, thời gian trễ và để cho mạch thực hiện đơn giản, người ta thường thiết kế sơ đồ sử dụng các JK – FF và các mạch NAND.

5.4.1. Phân tích mạch dãy đồng bộ

a) Các bước phân tích một mạch dãy đồng bộ như hình 5.4.1.1



Hình 5.4.1.1. Các bước phân tích một mạch dãy đồng bộ

1. Từ sơ đồ cho trước cần xác định chức năng từng phần tử cơ bản của sơ đồ, mối liên hệ giữa các phần tử đó.

2. Xác định số đầu vào, đầu ra của mạch, đặc điểm của các đầu vào, đầu ra đó. Để xác định được số trạng thái trong của mạch chúng ta cần phải xác định xem mạch được xây dựng từ bao nhiêu FF từ đó xác định được số trạng thái trong có thể có của mạch:

Nếu số FF là n thì số trạng thái trong có thể có của mạch là 2^n.

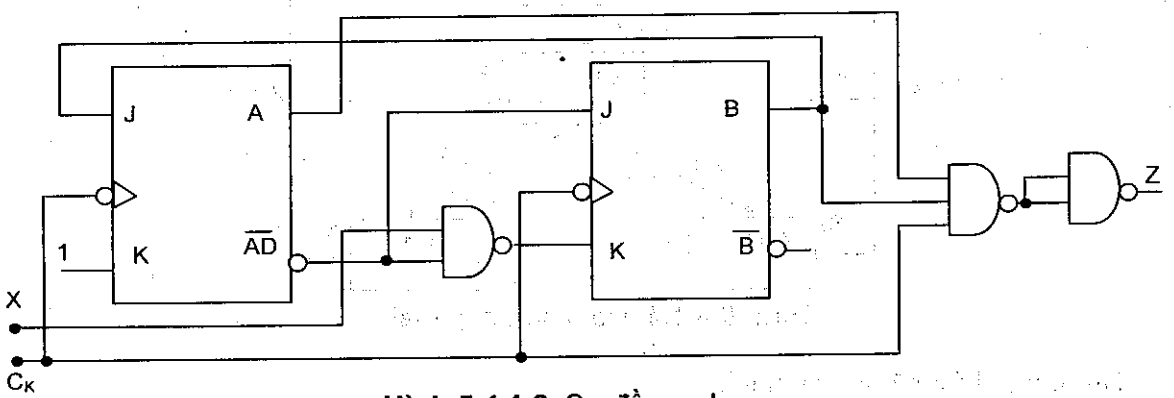
3. Dựa vào sơ đồ cho trước, chỉ ra được hệ phương trình hàm ra, hàm kích của các FF.

4. Dựa vào phương trình hàm kích, hàm ra đã xác định được ở trên, và dựa vào phương trình đặc tính của FF xác định được trạng thái chuyển biến tới và tín hiệu ra tương ứng với tín hiệu vào và trạng thái hiện tại của mạch.

5. Từ bảng trạng thái và bảng ra đã lập được ở trên, xây dựng được đồ hình trạng thái và tín hiệu ra của mạch.

6. Sau khi lập được đồ hình trạng thái, dựa vào đồ hình đó xác định được chức năng của mạch.

Ví dụ 5.4.1.1. Phân tích một mạch dãy đồng bộ có sơ đồ cho ở hình 5.4.1.2.



Hình 5.4.1.2. Sơ đồ mạch

Bước 1: Mạch sử dụng 2 trigơ JK và các cổng NAND.

Bước 2: Mạch có 2 đầu vào là tín hiệu vào X và xung nhịp tác động C_K, một đầu ra là Z. Do mạch được xây dựng từ hai FF nên số trạng thái trong có thể có của mạch là 4 (00, 01, 11, 10).

Bước 3: Xác định phương trình hàm ra và hàm kích cho các FF:

Phương trình hàm ra: $Z = A.B.C_K$

Phương trình hàm kích cho các FF A và B:

$$J_A = B;\ K_A = 1;\ J_B = \overline{A};\ K_B = \overline{X.\overline{A}} = \overline{X} + A.$$

Bước 4: Bảng trạng thái, bảng ra nhị phân:

Từ phương trình của JK – FF: $Q_{n+1} = J\overline{Q_n} + \overline{K}Q_n$

Ta có: $A_{n+1} = \overline{A_n}B_n$; $B_{n+1} = A_n\overline{B_n} + \overline{A_n}B_nX$

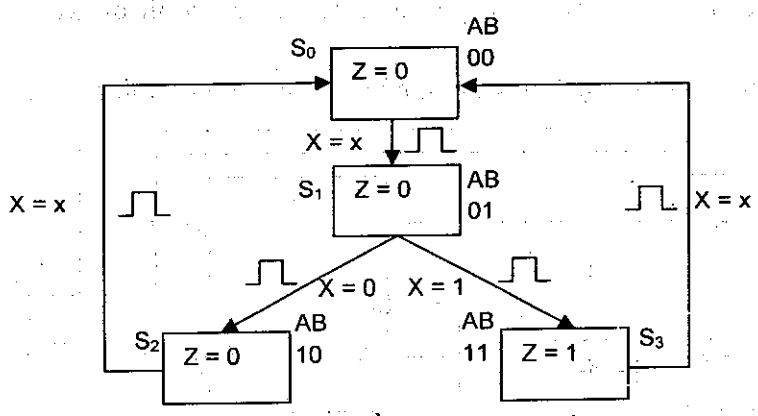
Bảng chuyển đổi trạng thái và bảng ra nhị phân theo trạng thái hiện tại (n) và tín hiệu X cho trên hình 5.4.1.3.

Trạng thái hiện tại	Trạng thái tiếp theo		Tín hiệu ra	
AB	X = 0 AB	X = 1 AB	X = 0 Z	X = 1 Z
00	01	01	0	0
01	10	11	0	0
11	00	00	1	1
10	00	00	0	0

Hình 5.4.1.3. Bảng trạng thái và bảng ra nhị phân

Z = 1 khi và chỉ khi cả A và B đều bằng 1 và có xung nhịp tác động. Cho nên khoảng thời gian Z = 1 bao giờ cũng bằng khoảng thời gian tồn tại của xung nhịp.

Bước 5: Đồ hình trạng thái như trên hình 5.4.1.4.



Hình 5.4.1.4. Đồ hình trạng thái

Bước 6: Chức năng của mạch:

Nhìn vào đồ hình chuyển đổi trạng thái ta thấy có 2 con đường chuyển đổi trạng thái ($S_0 \rightarrow S_0$) là: $S_0 \rightarrow S_1 \rightarrow S_2 \rightarrow S_0$ và $S_0 \rightarrow S_1 \rightarrow S_3 \rightarrow S_0$

– Theo con đường $S_0 \rightarrow S_1 \rightarrow S_3 \rightarrow S_0$ tín hiệu ra Z = 1 sẽ được đưa ra cùng với thời điểm có xung nhịp thứ 3.

– Theo con đường $S_0 \rightarrow S_1 \rightarrow S_2 \rightarrow S_0$ sẽ không cho tín hiệu ra (Z = 0).

Phân tích sự thay đổi trạng thái theo con đường $S_0 \rightarrow S_1 \rightarrow S_3 \rightarrow S_0$:

Chuyển đổi trạng thái đầu tiên từ S_0 đến S_1 chỉ nhờ tác động của xung nhịp, không phụ thuộc vào tín hiệu X = 1 hay X = 0 (X bất kỳ).

Chuyển đổi trạng thái thứ 2 từ S_1 đến S_3 nhờ tác động của tín hiệu vào X = 1 và xung nhịp tác động.

Còn chuyển đổi trạng thái thứ 3 từ S_3 đến S_0 chỉ nhờ tác động của xung nhịp và không phụ thuộc vào tín hiệu vào X = 1 hay X = 0 (X bất kỳ).

Mạch chỉ đưa ra tín hiệu ra Z = 1 khi dãy tín hiệu vào X có dạng 010, 011, 110, 111.

Tóm lại, mạch có chức năng kiểm tra dãy tín hiệu X vào ở dạng chuỗi có độ dài bằng 3. Nếu chuỗi tín hiệu vào có dạng là một trong 4 dãy: 010, 011, 110, 111, mạch sẽ cho tín hiệu ra Z = 1 tại thời điểm có xung nhịp thứ 3. Độ rộng của tín hiệu ra Z đúng bằng độ rộng của xung nhịp.

5.4.2. Tối thiểu hoá và mã hoá trạng thái trong

a) Phương pháp tối thiểu hoá Caldwell

– Định nghĩa các trạng thái tương đương:

Trạng thái S_i được gọi là tương đương với S_j khi và chỉ khi nếu lấy S_i và S_j là hai trạng thái ban đầu thì với mọi dãy tín hiệu vào có thể có chúng luôn luôn cho dãy tín hiệu ra giống nhau.

Nếu có nhiều trạng thái tương đương với nhau từng đôi một thì chúng tương đương với nhau (tính chất bắc cầu). Để kiểm tra một nhóm các trạng thái xem chúng có tương đương với nhau không, có thể sử dụng bảng trạng thái và tín hiệu ra như sau:

+ Nhóm các trạng thái tương đương phải có những hàng trong bảng tín hiệu ra giống nhau.

+ Nhóm các trạng thái tương đương phải có những hàng trong bảng trạng thái ở cùng một cột (ứng với cùng một tổ hợp tín hiệu vào) là tương đương. Nghĩa là ứng với cùng một tổ hợp tín hiệu vào các trạng thái sẽ chuyển biến tới của chúng là tương đương.

Điều này cho thấy thủ tục kiểm tra tính tương đương của một nhóm các trạng thái phải tiến hành tuần tự từng bước cho đến nhóm trạng thái cuối cùng. Nếu nhóm trạng thái cuối cùng này là tương đương thì nhóm trạng thái được kiểm tra là tương đương.

– Quy tắc Caldwell:

Những hàng (tương ứng với trạng thái trong) của bảng chuyển đổi trạng thái và tín hiệu ra, sẽ được kết hợp với nhau và được biểu diễn một hàng chung – đặc trưng (trạng thái đặc trưng) cho chúng nếu như chúng thoả mãn cả 2 điều kiện sau:

+ Các hàng tương ứng trong ma trận ra giống nhau.

+ Trong ma trận ra, các hàng tương ứng phải thoả mãn một trong ba điều sau:

- Các hàng trong ma trận trạng thái giống nhau.
- Các trạng thái ở trong cùng một cột nằm trong nhóm trạng thái được xét.
- Các trạng thái ở trong cùng một cột là các trạng thái tương đương.

Sau khi đã thay thế các trạng thái tương đương bằng một trạng thái chung đặc trưng cho chúng, lặp lại các công việc tìm các trạng thái tương đương (các hàng tương đương) khác, tới khi nào không thể tìm được các hàng (các trạng thái) tương đương với nhau nữa thì dừng lại. Số trạng thái trong bảng trạng thái và tín hiệu ra lúc đó là tối thiểu.

Nhược điểm của phương pháp này là khi số trạng thái (số hàng) của bảng trạng thái và tín hiệu ra là quá lớn thì công việc tối thiểu hoá mất nhiều thời gian.

b) Mã hoá trạng thái

Cần phải lựa chọn cách mã hoá trạng thái sao cho sơ đồ mạch thực hiện đơn giản nhất. Sơ đồ mạch đơn giản có lợi về nhiều phương diện: trước hết là việc chế tạo và lắp ráp sẽ đơn giản hơn, tăng độ tin cậy của mạch, số vi mạch cần thiết sẽ ít hơn...

Xét hai quy tắc mã hoá:

Quy tắc 1:

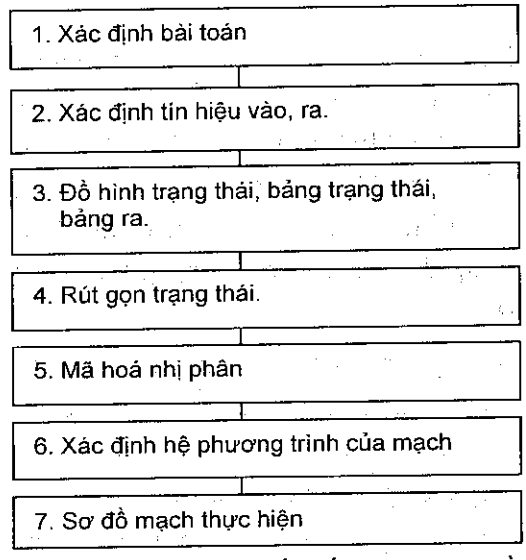
Nếu một trạng thái hiện tại S_i có thể chuyển biến đến nhiều trạng thái S_{i1}, S_{i2},.., S_{in} thì các trạng thái S_{ij} đó phải được mã hoá bằng các từ mã kế cận (nghĩa là chỉ khác nhau một bit).

Quy tắc 2:

Nếu nhiều trạng thái S_{i1}, S_{i2},.., S_{in} cùng chuyển biến đến một trạng thái tiếp theo S_i thì các trạng thái S_{i1}, S_{i2},.., S_{in} đó phải được mã hoá bằng các từ mã kế cận nhau.

5.4.3. Các bước thiết kế mạch dãy đồng bộ

Các bước thiết kế mạch dãy đồng bộ được trình bày trên hình 5.4.3.1.

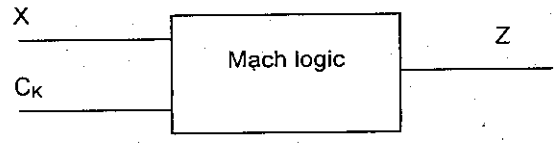


Hình 5.4.3.1. Các bước thiết kế mạch dãy đồng bộ

Ví dụ 5.4.3.1. Thiết kế một mạch dãy đồng bộ thực hiện nhiệm vụ kiểm tra dãy tín hiệu vào ở dạng nhị phân có độ dài bằng 3 được đưa vào liên tiếp trên đầu vào X. Nếu dãy tín hiệu vào có dạng là 010, 011, 110 hoặc 111 thì tín hiệu ra Z = 1, trong các trường hợp khác Z = 0.

Bước 1: Xác định bài toán

Mạch thiết kế có nhiệm vụ phát hiện dãy tín hiệu vào: nếu dãy tín hiệu vào có dạng là 010, 011, 110 hoặc 111 thì tín hiệu ra Z = 1 để báo hiệu là mạch đã nhận được một trong các dãy tín hiệu đó.



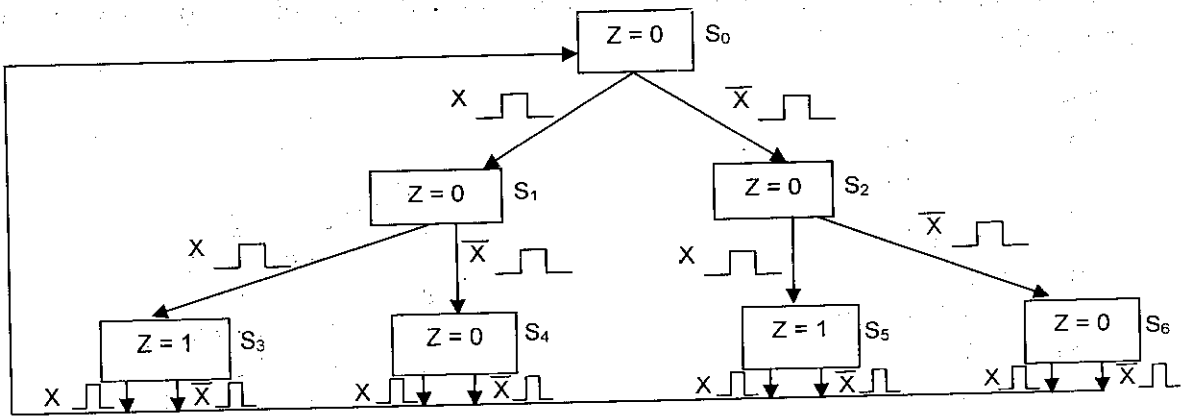
Hình 5.4.3.2

Bước 2: Xác định tín hiệu vào, ra.

Mạch phải thiết kế là mạch đồng bộ, nên ngoài đầu vào X còn có đầu vào xung nhịp C_K, một đầu tín hiệu ra là Z, hình 5.4.3.2.

Bước 3: Đồ hình trạng thái, bảng trạng thái, bảng ra.

Đồ hình trạng thái như hình 5.4.3.3.



Hình 5.4.3.3. Đồ hình trạng thái

Có thể giải thích đồ hình chuyển đổi trạng thái như sau:

– Trạng thái ban đầu S_0:

+ Khi tín hiệu vào là $X.C_K$ mạch sẽ chuyển đến trạng thái S_1.

+ Khi tín hiệu vào là $\overline{X}.C_K$ mạch sẽ chuyển đến trạng thái S_2.

– Tương tự như vậy cho các trạng thái S_1 và S_2.

– Nếu mạch ở một trong bốn trạng thái S_3, S_4, S_5, S_6:

+ Khi tín hiệu vào là $X.C_K$ hoặc $\overline{X}.C_K$ mạch sẽ chuyển về trạng thái ban đầu S_0.

Vậy khi dãy tín hiệu vào là 110 hoặc 111 (đường chuyển đổi trạng thái $S_0 \rightarrow S_1 \rightarrow S_3 \rightarrow S_0$) hay khi dãy tín hiệu vào là 010 hoặc 011 (đường chuyển đổi trạng thái $S_0 \rightarrow S_2 \rightarrow S_5 \rightarrow S_0$) thì mạch sẽ cho tín hiệu ra Z = 1 tại thời điểm của xung nhịp thứ 3. Với các đường chuyển đổi khác Z = 0. Từ đồ hình chuyển đổi trạng thái lập được bảng trạng thái, bảng ra như trên hình 5.4.3.4.

S_n	S_{n+1}		Z	
	X = 0	X = 1	X = 0	X = 1
S_0	S_2	S_1	0	0
S_1	S_4	S_3	0	0
S_2	S_6	S_5	0	0
S_3	S_0	S_0	1	1
S_4	S_0	S_0	0	0
S_5	S_0	S_0	1	1
S_6	S_0	S_0	0	0

Hình 5.4.3.4. Bảng trạng thái, bảng ra

Bước 4: Tối thiểu hoá trạng thái

Áp dụng quy tắc Caldwell: Trong bảng trạng thái, bảng ra, trạng thái S_4 tương đương với S_6, S_3 tương đương với S_5. Thay thế các trạng thái tương đương bằng một trạng thái chung đặc trưng chúng. Ví dụ, thay thế S_4, S_6 bởi S_{46}; thay thế S_3, S_5 bởi S_{35}. Từ đó lập được bảng trạng thái, bảng ra hình 5.4.3.5a.

Nhìn vào bảng hình 5.4.3.5a ta thấy ngay S_1 tương đương với S_2, gộp S_1 và S_2 ta được bảng như hình 5.4.3.5b

X	0	1
S_0	S_2 Z = 0	S_1 Z = 0
S_1	S_{46} Z = 0	S_{35} Z = 0
S_2	S_{46} Z = 0	S_{35} Z = 0
S_{35}	S_0 Z = 1	S_0 Z = 1
S_{46}	S_0 Z = 0	S_0 Z = 0

a)

X	0	1
S_0	S_{12} Z = 0	S_{12} Z = 0
S_{12}	S_{46} Z = 0	S_{35} Z = 0
S_{35}	S_0 Z = 1	S_0 Z = 1
S_{46}	S_0 Z = 0	S_0 Z = 0

b)

Hình 5.4.3.5.
a) Bảng trạng thái, bảng ra sau khi gộp S_4 và S_6, S_3 và S_5
b) Bảng trạng thái, bảng ra sau khi gộp S_1 và S_2.

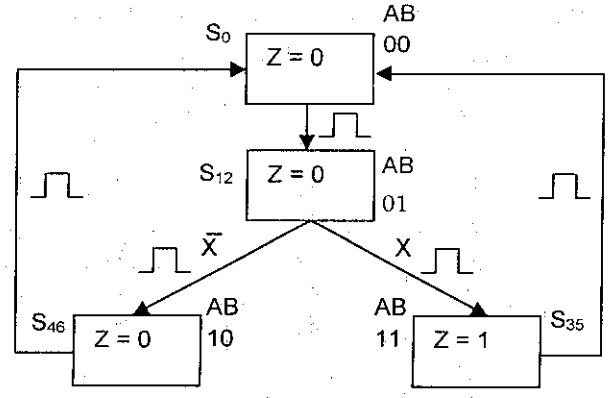
Ta không thể tìm thấy hai trạng thái nào tương đương trong bảng ở hình 5.4.3.5b. Bước tối thiểu hoá kết thúc ở đây với số trạng thái tối thiểu là 4 gồm: S_0, S_{12}, S_{35}, S_{46}.

Bước 5: Mã hoá trạng thái

Sau khi đã tối thiểu hoá mạch còn 4 trạng thái S_0, S_{12}, S_{35}, S_{46}. Mã hoá 4 trạng thái này bằng 2 biến nhị phân A, B như hình 5.4.3.6 (nếu thiết kế mạch dùng FF ta sẽ dùng 2 FF ký hiệu A, B tương ứng với 2 biến nhị phân trên).

A	B	Mã hoá S
0	0	S_0
0	1	S_{12}
1	1	S_{35}
1	0	S_{46}

Hình 5.4.3.6. Bảng mã hoá trạng thái



Hình 5.4.3.7. Đồ hình trạng thái

Từ bảng trạng thái, bảng ra đã tối thiểu hình trên 5.4.3.5b có thể xây dựng được đồ hình trạng thái như hình 5.4.3.7.

Bước 6: Xác định hệ phương trình của mạch

Từ đồ hình chuyển đổi trạng thái trên hình 5.4.3.7 ta có bảng trạng thái nhị phân và giá trị tương ứng của các đầu vào điều khiển của các trigơ A và B như trên hình 5.4.3.8.

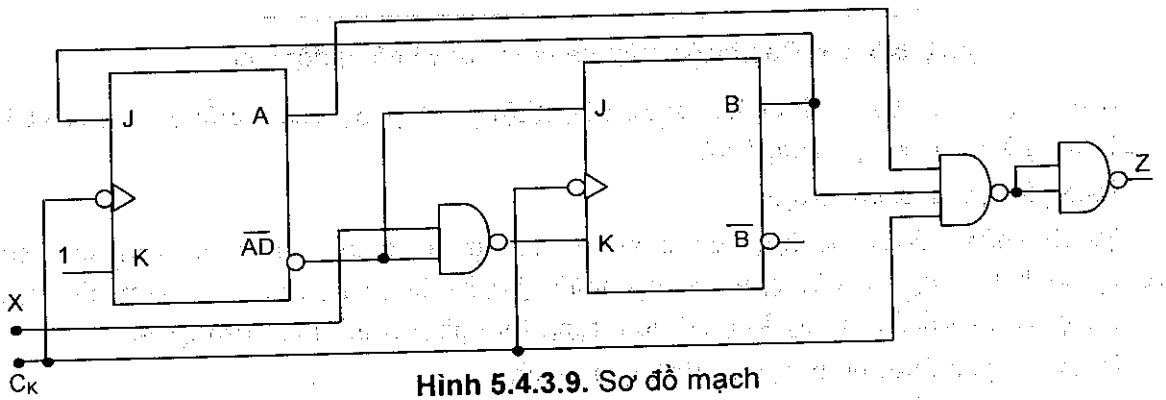
Trạng thái hiện tại	Trạng thái tiếp theo		Các đầu vào của các FF							
	X = 0	X = 1	X = 0		X = 1		X = 0		X = 1	
AB	AB	AB	J_A	K_A	J_A	K_A	J_B	K_B	J_B	K_B
00	01(Z = 0)	01(Z = 0)	0	X	0	X	1	X	1	X
01	10(Z = 0)	11(Z = 0)	1	X	1	X	X	1	X	0
11	00(Z = 1)	00 (Z = 1)	X	1	X	1	X	1	X	1
10	00(Z = 0)	00(Z = 0)	X	1	X	1	0	X	0	X

Hình 5.4.3.8. Bảng trạng thái nhị phân và giá trị tương ứng của các đầu vào điều khiển

Tối thiểu hoá các hàm J và K ta được: $J_A = B$; $K_A = 1$; $J_B = \overline{A}$; $K_B = \overline{X} + A$

Phương trình tín hiệu ra Z khi X = 1: Z = A. B. C_K.

Bước 7: Sơ đồ mạch thực hiện như trên hình 5.4.3.9.



Hình 5.4.3.9. Sơ đồ mạch

5.5. MẠCH DÃY KHÔNG ĐỒNG BỘ

Trong phần 5.4 đã nghiên cứu về các mạch dãy đồng bộ, hoạt động của chúng được điều khiển bởi các xung nhịp. Tuy nhiên trong thực tế có những mạch lại được điều khiển bởi các sự kiện không tuân theo một quy luật nào cả. Chẳng hạn một hệ thống báo động cháy sẽ chỉ hoạt động khi có hiện tượng cháy. Những mạch dãy được điều khiển bởi những sự kiện ngẫu nhiên như vậy gọi là các mạch không đồng bộ.

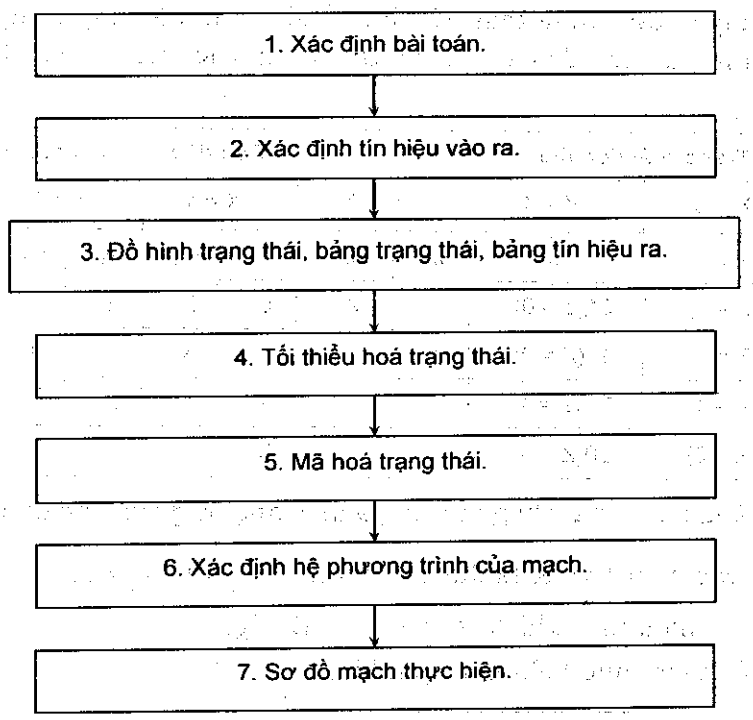
Mạch dãy không đồng bộ có thể thiết kế:

– Chỉ dùng các mạch NAND

– Dùng RS – FF không đồng bộ và các mạch NAND

5.5.1. Các bước thiết kế mạch dãy không đồng bộ

Các bước thiết kế một mạch dãy không đồng bộ được mô tả trong sơ đồ hình 5.5.1.1.



Hình 5.5.1.1. Các bước thiết kế mạch dãy không đồng bộ

Nhận xét: Các bước thiết kế mạch dãy không đồng bộ cũng giống như thiết kế mạch đồng bộ, chỉ khác bước 5 và 6.

Bước 5: Mã hoá trạng thái

Mạch không đồng bộ hoạt động không có sự tác động của xung nhịp cho nên trong mạch thường có các hiện tượng hoặc chu kỳ hoặc là chạy đua làm cho hoạt động của mạch sai đi, vì vậy khi mã hoá trạng thái phải tránh hiện tượng này.

Bước 6: Xác định hệ phương trình của mạch

Có hai cách để xác định hệ phương trình của mạch: Dựa vào bảng trạng thái, tín hiệu ra và dựa trực tiếp vào đồ hình. Cả hai cách đều có hai loại phương trình:

– Phương trình của mạch chỉ dùng NAND.

– Phương trình của mạch dùng RS – FF không đồng bộ và các mạch NAND.

Nội dung của từng phương pháp:

Cách 1: Dựa vào bảng trạng thái và tín hiệu ra

a) Chỉ dùng các mạch NAND

Ký hiệu: A, B, …, N là các biến nhị phân dùng để mã hoá các trạng thái trong của mạch.

$X_1, X_2,\ldots,X_m$ là các tín hiệu vào đã được mã hoá nhị phân

$Z_1, Z_2, \ldots, Z_n$ là các tín hiệu ra đã được mã hoá nhị phân.

Dựa vào bảng trạng thái, bảng ra xác định được hệ phương trình:

$$A_{n+1} = f_A(A_n, B_n, \ldots, N_n, X_1, X_2, \ldots, X_m)$$
$$B_{n+1} = f_B(A_n, B_n, \ldots, N_n, X_1, X_2, \ldots, X_m)$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$N_{n+1} = f_N(A_n, B_n, \ldots, N_n, X_1, X_2, \ldots, X_m)$$
$$Z_1 = g_1(A_n, B_n, \ldots, N_n, X_1, X_2, \ldots, X_m)$$
$$Z_2 = g_2(A_n, B_n, \ldots, N_n, X_1, X_2, \ldots, X_m)$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$Z_n = g_n(A_n, B_n, \ldots, N_n, X_1, X_2, \ldots, X_m)$$

Tối thiểu hoá hệ hàm này và viết phương trình ở dạng chỉ dùng NAND

b) Mạch dùng RS – FF và các mạch NAND

Trong bảng trạng thái, căn cứ vào sự thay đổi trạng thái của từng FF:

A_n sang A_{n+1}, B_n sang B_{n+1}, …, N_n sang N_{n+1}, xác định được giá trị tương ứng của đầu vào điều khiển R, S cho từng FF, từ đó viết được hệ phương trình:

$R_A = x_1(A_n, B_n, \ldots, N_n, X_1, X_2, \ldots, X_m)$

$S_A = x_2(A_n, B_n, \ldots, N_n, X_1, X_2, \ldots, X_m)$

Tối thiểu hoá các hàm và viết phương trình ở dạng dùng NAND.

Tương tự với B, C,…, N.

Ta xác định được tín hiệu ra: $Z = x(A_n, B_n, \ldots, N_n, X_1, X_2, \ldots, X_m)$

Tối thiểu hoá và viết phương trình ở dạng chỉ dùng NAND.

Cách 2: Dựa trực tiếp vào đồ hình

Ta xác định phương trình đầu vào điều khiển R, S của A – FF như sau:

$$S_A = T_{onA} + [(1)]$$
$$R_A = T_{offA} + [(0)]$$

Trong đó: T_{onA} (tập bật của A) là tập hợp A chuyển từ 0 sang 1, T_{offA} (tập tắt của A) là tập hợp A chuyển từ 1 sang 0, (1) là tập hợp A giữ nguyên giá trị 1, (0) là tập hợp A giữ nguyên giá trị 0. Với (1) và (0) được để trong dấu [ ] để chỉ đó là các tổ hợp không xác định, ta có thể dùng để tối thiểu hoá (nếu cần). Điều này cũng hoàn toàn tương tự đối với B – FF, C – FF,…

a) Nếu chỉ dùng mạch NAND

Ta có đã có phương trình của RS – FF là $Q_{n+1} = S + \overline{R}Q_n$, suy ra $A' = S_A + \overline{R_A}A$

Sau đó ta tối thiểu hoá phương trình và viết dưới dạng chỉ dùng NAND. Đối với B,C,… N cũng làm tương tự như vậy.

b) Nếu dùng RS – FF không đồng bộ và các mạch NAND

Ta cần xác định S_A, R_A theo các biến đầu vào và các biến nhị phân dùng để mã hoá:

$$R_A = x_{1A}(A_n, B_n, \ldots, N_n, X_1, X_2, \ldots, X_m)$$
$$S_A = x_{2A}(A_n, B_n, \ldots, N_n, X_1, X_2, \ldots, X_m)$$
$$\ldots\ldots\ldots\ldots\ldots\ldots$$
$$R_N = x_{1N}(A_n, B_n, \ldots, N_n, X_1, X_2, \ldots, X_m)$$
$$S_N = x_{2N}(A_n, B_n, \ldots, N_n, X_1, X_2, \ldots, X_m)$$
$$Z_1 = y_1(A_n, B_n, \ldots, N_n, X_1, X_2, \ldots, X_m)$$
$$Z_2 = y_2(A_n, B_n, \ldots, N_n, X_1, X_2, \ldots, X_m)$$
$$\ldots\ldots\ldots\ldots\ldots\ldots$$
$$Z_N = y_N(A_n, B_n, \ldots, N_n, X_1, X_2, \ldots, X_m)$$

Tối thiểu hoá hệ phương trình: viết các phương trình ở dạng dùng mạch NAND.

5.5.2. Hiện tượng chu kỳ và chạy đua

Đối với mạch dãy không đồng bộ trong mạch thường xảy ra các hiện tượng hoặc là chu kỳ hoặc là chạy đua. Những hiện tượng này làm cho mạch hoạt động sai lệch đi so với chức năng của nó. Vì vậy khi mã hoá trạng thái các mạch dãy không đồng bộ, điều kiện đầu tiên là cần tránh các hiện tượng đó.

a) Hiện tượng chu kỳ

**Định nghĩa:*

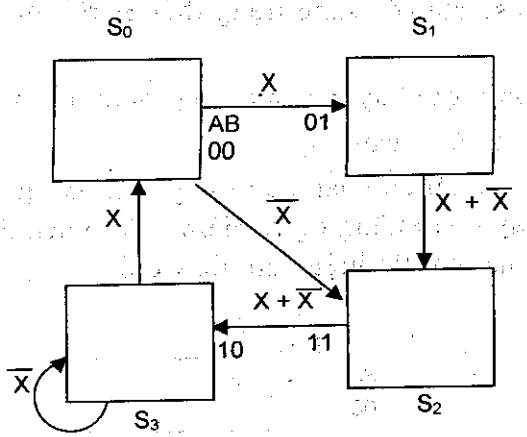
Hiện tượng chu kỳ là hiện tượng tại một tổ hợp tín hiệu vào nào đó, mạch liên tục chuyển từ trạng thái này sang trạng thái khác theo một chu kỳ kín. Nghĩa là trong quá trình đó không có trạng thái nào ổn định. Do vậy, khi thay đổi tín hiệu vào không xác định được mạch đang ở trạng thái nào trong dãy các trạng thái nói trên.

Ví dụ, ứng với một tổ hợp tín hiệu vào quá trình chuyển đổi trạng thái theo chu trình hình 5.5.2.1.

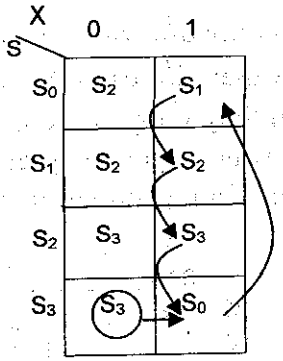
Hình 5.5.2.1. Quá trình chuyển đổi trạng thái

Trên bảng trạng thái hiện tượng chu kỳ được thể hiện ở chỗ: cột ứng với tổ hợp tín hiệu vào đó không có trạng thái nào được khoanh tròn (không có trạng thái nào ổn định).

Ví dụ 5.5.2.1. Đồ hình trạng thái của một mạch dãy không đồng bộ được biểu diễn trên hình 5.5.2.2. Việc mã hoá trạng thái sử dụng biến nhị phân A và B hoàn toàn tuỳ ý. Từ đồ hình trạng thái, lập được bảng trạng thái trên hình 5.5.2.3.

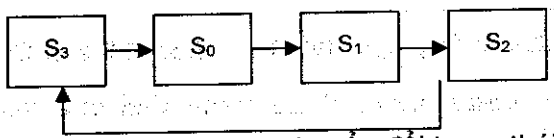


Hình 5.5.2.2. Đồ hình trạng thái



Hình 5.5.2.3. Bảng trạng thái

Giả thiết ban đầu mạch ở trạng thái S_3 (AB = 10) và X = 0. Sau đó tín hiệu vào X thay đổi từ 0 đến 1 mạch sẽ chuyển từ trạng thái S_3 sang S_0. Nếu X vẫn bằng 1, chu trình chuyển đổi trạng thái như hình 5.5.2.4:



Hình 5.5.2.4. Chu trình chuyển đổi trạng thái

khi đó mạch không có trạng thái ổn định.

b) Hiện tượng chạy đua

** Định nghĩa:*

Hiện tượng chạy đua trong mạch không đồng bộ là hiện tượng: do tính không đồng nhất của các phần tử dùng để mã hoá trạng thái, vì mạch hoạt động không đồng bộ nên khi mạch chuyển trạng thái từ S_i sang S_j mạch có thể chuyển biến trạng thái theo những con đường khác nhau.

Nếu trạng thái cuối cùng của những con đường ấy là ổn định và duy nhất thì chạy đua này là chạy đua không nguy hiểm. Ngược lại, chạy đua nguy hiểm là những cách chuyển biến trạng thái khác nhau ấy cuối cùng dẫn tới các trạng thái ổn định khác nhau, có thể trạng thái khoá và không thoát ra được.

Ví dụ về chạy đua không nguy hiểm: Một mạch dãy không đồng bộ có bảng trạng thái mô tả ở hình 5.5.2.5.

Nhìn vào bảng ta thấy nếu mạch đang ở trạng thái S_0 (AB = 00) tín hiệu vào X thay đổi từ 0 sang 1, mạch sẽ chuyển trực tiếp tới trạng thái S_1 (AB = 01). Nếu X vẫn bằng 0 trạng thái tiếp theo của mạch sẽ là S_3, nó sẽ là trạng thái ổn định cuối cùng của mạch nếu như X vẫn bằng 0.

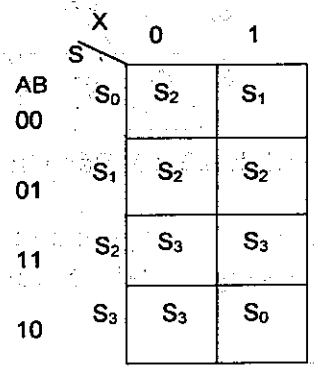
Mạch có thể thay đổi trạng thái theo những con đường khác nhau tuỳ thuộc vào thứ tự thay đổi (hay thời gian quá độ) của A và B.

– Nếu A và B thay đổi đồng thời mạch sẽ chuyển sang trạng thái S_2 rồi mới sang trạng thái S_3.

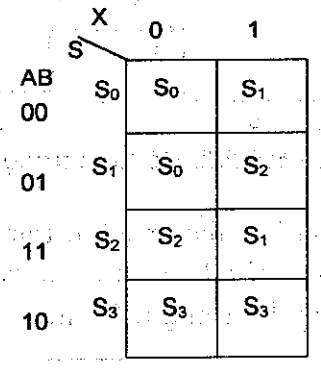
– Nếu B thay đổi trước A thì mạch sẽ lần lượt chuyển qua S_1, S_2 rồi mới sang S_3.

– Nếu A thay đổi trước B mạch sẽ chuyển từ S_0 sang S_3.

Ta nhận thấy rằng cả 3 con đường chuyển đổi đều dẫn đến trạng thái ổn định cuối cùng là S_3. Hiện tượng chạy đua này là chạy đua không nguy hiểm. Khi mạch đang ở trạng thái ổn định S_3 nó chỉ thay đổi trạng thái khi tín hiệu vào thay đổi.



Hình 5.5.2.5. Chạy đua không nguy hiểm



Hình 5.5.2.6. Chạy đua nguy hiểm

Ví dụ về chạy đua nguy hiểm: Bảng trạng thái của một mạch không đồng bộ được mô tả ở hình 5.5.2.6.

Giả thiết trạng thái ban đầu của mạch là S_0 (AB = 00) và tín hiệu vào X = 0. Nếu X thay đổi từ 0 sang 1 mạch sẽ chuyển đổi trạng thái như sau:

– Nếu A và B thay đổi đồng thời mạch sẽ chuyển tới trạng thái S_1.

– Nếu B thay đổi trước A mạch sẽ chuyển từ S_2 sang S_1.

– Nếu A thay đổi trước B mạch sẽ chuyển tới trạng thái S_3 và giữ nguyên trạng thái đó.

Ở đây trạng thái S_3 là trạng thái "khoá", khi A thay đổi trước B thì mạch sẽ rơi vào trạng thái khoá không thoát ra được.

Chạy đua này là chạy đua nguy hiểm.

5.5.3. Tối thiểu hoá và mã hoá trạng thái

a) Tối thiểu hoá trạng thái

Tối thiểu hoá trạng thái là giảm bớt số trạng thái (nếu có thể) để mạch thiết kế là đơn giản và do vậy tin cậy hơn.

Cần lưu ý rằng tại những ô trống trong bảng chuyển đổi trạng thái và bảng ra (những tổ hợp này ứng với những tổ hợp vào không xuất hiện) có thể lấy giá trị tuỳ ý để cho kết quả tối thiểu hoá trạng thái là tối giản.

b) Mã hoá trạng thái

Sử dụng các biến nhị phân để mã hoá các trạng thái trong của mạch. Gọi N là số trạng thái trong của mạch; n là số lượng biến nhị phân dùng để mã hoá, n phải thoả mãn: $n \geq \log_2 N$.

Đối với mạch dãy đồng bộ chỉ cần tối thiểu hoá và mã hoá trạng thái sao cho mạch thực hiện là đơn giản nhất.

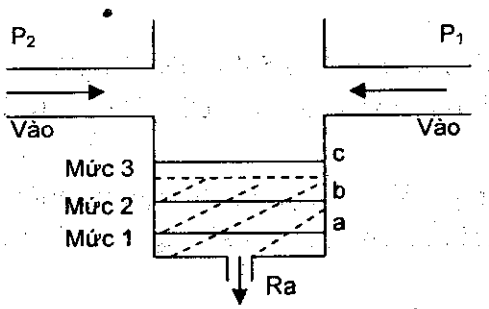
Đối với mạch dãy không đồng bộ phải mã hoá trạng thái sao cho mạch thực hiện vừa đơn giản lại vừa tránh được hiện tượng chu kỳ và chạy đua.

Để tránh được hiện tượng chu kỳ, phải lưu ý sao cho với mọi tổ hợp tín hiệu vào thì mạch phải luôn luôn có một trạng thái ổn định.

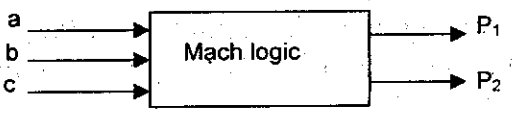
Để tránh hiện tượng chạy đua, phải mã hóa trạng thái sao cho với tất cả các chuyển biến trạng thái có thể có của mạch chỉ có duy nhất 1 biến thay đổi mà thôi.

5.5.4. Ví dụ về thiết kế mạch dãy không đồng bộ

Thiết kế mạch điều khiển bơm nước vào một tháp nước nhờ hai bơm P_1 và P_2. Cả hai bơm P_1 và P_2 được mở khi nước ở dưới mức 1 và vẫn mở cho đến khi nước chưa đạt tới mức 2. Khi nước vừa đạt tới mức 2 thì bơm P_1 ngắt, chỉ còn P_2 vẫn bơm. P_1 vẫn ngắt cho đến khi nước lại ở dưới mức 1. P_2 vẫn mở, chỉ khi nào nước đạt tới mức 3 thì P_2 mới ngắt. P_2 vẫn ngắt, chỉ mở khi nước lại xuống dưới mức 1, hệ thống được minh họa trên hình 5.5.4.1. Sơ đồ khối cho bộ điều khiển bơm cho trên hình 5.5.4.2.



Hình 5.5.4.1. Minh hoạ hệ thống



Hình 5.5.4.2. Sơ đồ khối

Một bộ cảm biến được sử dụng để đo nước đưa ra các tín hiệu như sau:

a = 1 khi mức nước lớn hơn hoặc bằng mức 1, trường hợp khác a = 0.

b = 1 khi mức nước lớn hơn hoặc bằng mức 2, trường hợp khác b = 0.

c = 1 khi mức nước lớn hơn hoặc bằng mức 3, trường hợp khác c = 0.

Mã hoá cho trạng thái của bơm: P = 1 bơm mở; P = 0 bơm ngắt.

Đồ hình trạng thái trong:

Từ các dữ liệu trên, lập được đồ hình trạng thái như trên hình 5.5.4.3.

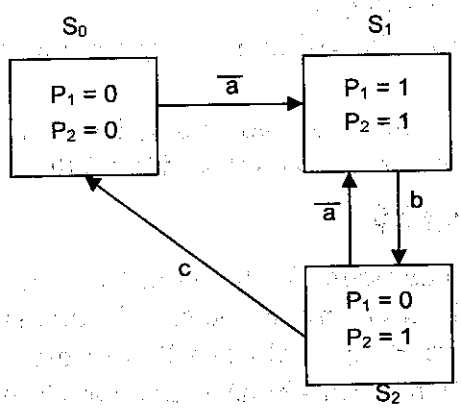
– Trạng thái S_0 tương ứng khi nước lớn hơn hoặc bằng mức 3 cả hai bơm P_1 và P_2 đều ngắt.

– Khi nước tụt xuống dưới mức 1($\bar{a}$) mạch chuyển sang trạng thái S_1 lúc đó cả hai bơm đều mở.

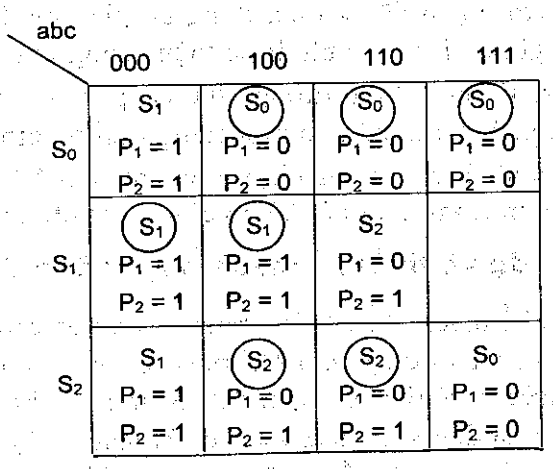
– Đến khi nước đạt mức 2 (b), mạch chuyển sang trạng thái S_2 lúc đó bơm P_2 vẫn bơm, bơm P_1 ngắt. Có hai trường hợp có thể xảy ra:

+ Nếu nước giảm xuống dưới mức 1 ($\bar{a}$) chuyển sang trạng thái S_1 cả hai bơm đều mở.

+ Nếu nước tiếp tục dâng lên đến mức 3 (c) mạch sẽ chuyển sang trạng thái S_0 và cả hai bơm đều tắt.



Hình 5.5.4.3. Đồ hình trạng thái



Hình 5.5.4.4. Bảng trạng thái

Bảng trạng thái:

Từ đồ hình trạng thái, lập được bảng trạng thái như hình 5.5.4.4. Do ý nghĩa vật lý của các biến vào, chỉ có thể xuất hiện ở đầu vào 4 giá trị sau của abc: 000, 100, 110, 111.

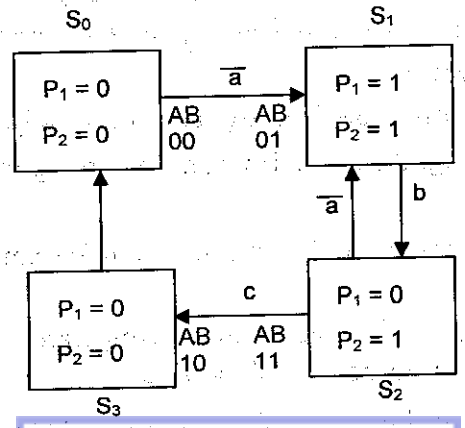
Mã hoá trạng thái:

Từ bảng trạng thái hình 5.5.4.4 thấy rằng: ứng với mỗi cột của tổ hợp tín hiệu vào mạch luôn có ít nhất một trạng thái ổn định, nghĩa là trong mạch không có hiện tượng chu kỳ.

Như vậy, khi mã hoá trạng thái chỉ còn phải tránh hiện tượng chạy đua. Để mã hoá trạng thái của mạch, cần hai biến nhị phân A, B. Và để tránh hiện tượng chạy đua đưa thêm trạng thái giả S_3 như hình 5.5.4.5.

Hệ phương trình của mạch:

Từ đồ hình trạng thái hình 5.5.4.5.



Hình 5.5.4.5. Đồ hình trạng thái

Hệ phương trình của mạch như sau:

$$S_A = b.B$$

$$R_A = \overline{B} + B.\overline{a} = \overline{B} + \overline{a}$$

$$S_B = \overline{a}.\overline{A}$$

$$R_B = c.A$$

Vậy:

$$A_{n+1} = b.B + (\overline{\overline{B} + \overline{a}})A = b.B + a.A.B$$

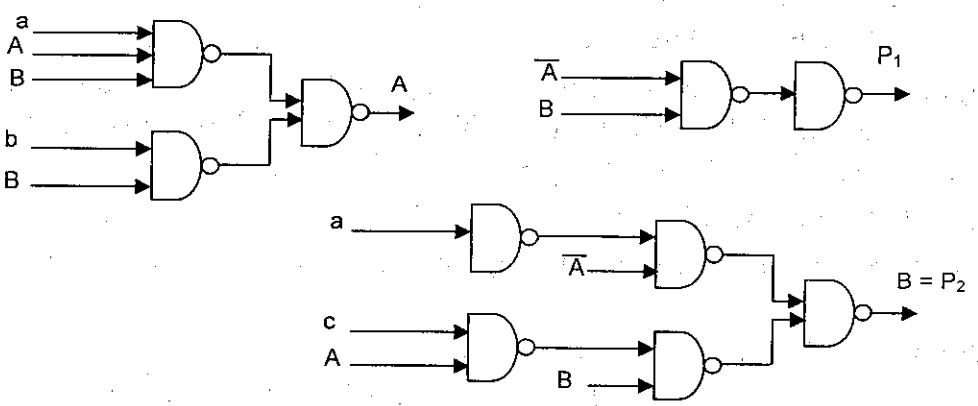
$$B_{n+1} = \overline{a}.\overline{A} + \overline{c.A}.B$$

$$P_1 = \overline{A}.B$$

$$P_2 = \overline{A}.B + A.B = B$$

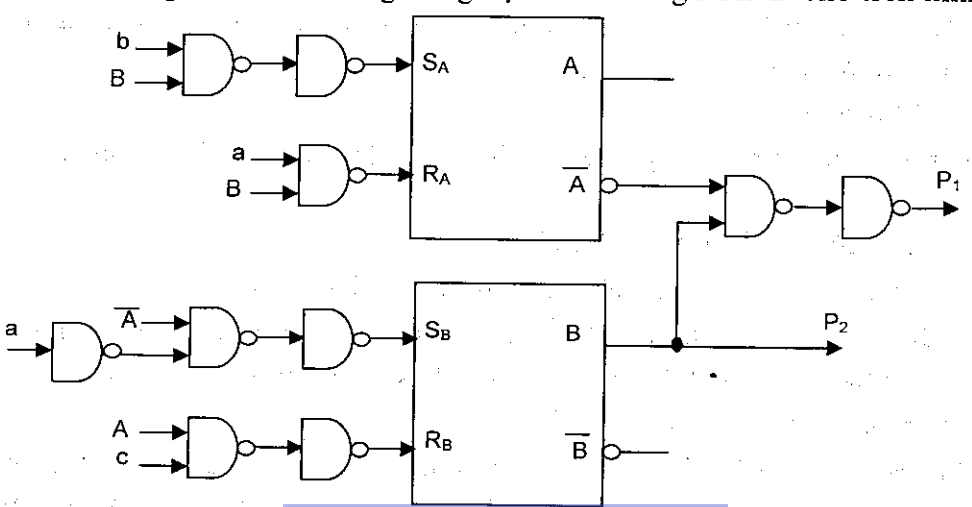
Mạch thực hiện:

– Sơ đồ chỉ dùng cổng NAND cho trên hình 5.5.4.6.



Hình 5.5.4.6. Sơ đồ chỉ dùng cổng NAND

– Sơ đồ dùng RS – FF không đồng bộ và các cổng NAND cho trên hình 5.5.4.7.



Hình 5.5.4.7. Sơ đồ dùng RS – FF không đồng bộ và các cổng NAND

5.6. CÁC BỘ NHỚ BÁN DẪN

5.6.1. Khái niệm cơ bản

a) Chức năng của bộ nhớ

Các bộ ghi mà chúng ta đã khảo sát ở mục trước thường được dùng để lưu trữ thông tin về trạng thái của máy hoặc số liệu trung gian trong một thời gian ngắn. Để lưu trữ được các chương trình điều khiển tính toán và các kết quả tính toán trong thời gian dài cần phải có các bộ nhớ dung lượng lớn. Đối với các thiết bị số và máy tính điện tử, khả năng nhớ được dữ liệu là một yêu cầu quan trọng. Ví dụ như trong máy tính, các chương trình tính toán các con số cần thiết cho phép toán phải được lưu trữ ngay trong máy. Còn trong các thiết bị điều khiển số thì các lệnh điều khiển cũng phải được lưu trữ để điều khiển quá trình vận hành theo một trình tự đã được định trước do người lập trình vạch ra. Vì vậy, các bộ nhớ đảm nhiệm chức năng lưu trữ thông tin không thể thiếu trong các thiết bị số.

b) Các thông số cơ bản của bộ nhớ

Thông tin dữ liệu được tạo thành từ một đơn vị cơ bản gọi là từ (word). Tuỳ theo từng loại máy, một từ có thể 8 bit, 16bit, 32 bit,... Các thiết bị chỉ truyền đi hay nhận vào nguyên 1 từ hay nhiều từ chứ không phải chỉ vài bit của từ.

Khi nói đến bộ nhớ, người ta quan tâm ngay đến hai thông số cơ bản:

1. Dung lượng nhớ (Capicity): là lượng thông tin hay dữ liệu lưu trữ được trong bộ nhớ.

Các đơn vị nhớ là: bit, byte, KB, MB, GB.

1byte = 8 bit; $1KB = 2^{10}$ byte = 1024 byte = 2^{13} bit

$1MB = 2^{10}$ KB = 1024 KB; $1GB = 2^{10}$ MB = 1024 MB

2. Thời gian truy cập (Access time – at): thời gian này gồm hai phần là: thời gian xác định vị trí từ và thời gian đọc nội dung từ lưu giữ trong bộ nhớ, at càng nhỏ, tốc độ làm việc của máy càng cao.

c) Phân loại bộ nhớ

Việc phân loại bộ nhớ có nhiều cách, xét về cấu trúc máy tính ta thấy có hai loại bộ nhớ:

Bộ nhớ trong: gồm có ROM, RAM, đĩa cứng.

Bộ nhớ ngoài: gồm có các đĩa mềm, các đĩa CD ROM, ổ USB.

Trong phạm vi giáo trình kỹ thuật số này chỉ đề cập đến các bộ nhớ bán dẫn, các bộ nhớ này có trong các máy tính điện tử và các thiết bị điện tử điều khiển số.

Các bộ nhớ bán dẫn được chia làm hai loại:

– Bộ nhớ truy cập ngẫu nhiên RAM (Random Access Memory), RAM là bộ nhớ bán dẫn tác động nhanh có thể ghi số liệu vào và đọc số liệu ra từ RAM ở thời điểm nào cũng được.

– Bộ nhớ chỉ đọc ra ROM (Read Only Memory) trong đó các dữ liệu đã được các nhà sản xuất ghi vào và nó chỉ được đọc ra khi dùng.

5.6.2. Bộ nhớ chỉ đọc ROM

ROM là bộ nhớ vĩnh viễn, khác với RAM thông tin chứa trong ROM không bị mất đi khi không còn nguồn điện. Các dữ liệu đã được nạp vào ROM do nhà chế tạo thực hiện khi sản xuất có thể là các hằng số vật lý, toán học như số π, số e, các công thức toán học, các hàm số lượng giác sin, cos, các bộ biến đổi mã, giải mã các ký tự v.v... Dữ liệu cũng có thể là các lệnh điều khiển khởi động máy tính, các chương trình con điều khiển sự hoạt động của máy tính hay các thiết bị điều khiển tự động. Nó chỉ dùng để đọc ra trong quá trình vận hành của thiết bị. Trong ROM còn có các loại:

PROM (Programable ROM) là bộ nhớ được chương trình hoá tại nhà sản xuất.

Các vi mạch loại PROM:

54/74S188 (32×8bit) ; 54/74S287 (256×4bit) ; 54/74S472(512×8bit).

EPROM (Erasable Programable ROM) là bộ nhớ có thể lập trình hoá có thể xoá và nạp lại được. EPROM có hai loại:

– UV – EPROM (Ultra – Violet). Bộ nhớ lập trình hoá có thể xoá bằng tia cực tím. Ta có thể dễ dàng nhận ra loại này từ hình dạng bên ngoài của nó, loại này được đóng trong vỏ nhựa có cửa sổ tròn trong suốt, qua cửa sổ ta có thể nhìn thấy chip bán dẫn ở bên trong. Nhờ có cửa sổ này ta có thể xoá nội dung thông tin ghi trong EPROM bằng cách dùng đèn tử ngoại chiếu tia cực tím vào chíp. Bộ nhớ sau khi xoá có thể lập trình lại. Khi dùng loại EPROM này cần che cửa sổ để tránh ánh sáng mặt trời dọi vào làm mất thông tin ghi trong mạch.

Các vi mạch UV – EPROM của hãng National Semiconductor:

M2708(1028×8bit) ; MM2704(512×8bit).

– E – EPROM (Electically Programable and Erasable ROM): Bộ nhớ lập trình xoá bằng xung điện 20V. Các vi mạch thuộc loại này của hãng National Semiconductor có các ký hiệu như sau:

MM4203/MM5203 2048bit (256×8bit hoặc 512×4bit); MM4204/MM5204 4096 bit (512×8bit).

5.6.3. Bộ nhớ truy cập ngẫu nhiên RAM

RAM là bộ nhớ vừa đọc ra vừa ghi dữ liệu vào. Thông tin chứa trong RAM sẽ bị mất khi không còn nguồn điện nuôi nó. Nó được chia ra các loại:

– RAM lưu trữ : Dùng CMOS nuôi bằng accur lithium liên tục được 10 năm.

– RAM không lưu trữ: Khi ngắt nguồn nuôi các dữ liệu đã nhớ bị xoá. Loại RAM này lại có hai loại:

+ RAM tĩnh (Static): Cấu tạo từ các trigơ lưỡng cực, MOS hoặc CMOS. Ta gọi là RAM tĩnh vì các dữ liệu sẽ còn nguyên vẹn khi nguồn nuôi còn duy trì.

+ RAM động (Dynamic): Cấu tạo từ các tranzito MOS hoặc CMOS có thêm các tụ điện để lưu trữ số liệu. Vì qua tụ có dòng dò nên theo thời gian điện thế trên tụ giảm dần nên số liệu không duy trì được nguyên vẹn mặc dù chưa ngắt nguồn. Để duy trì được dữ liệu chứa trong RAM động ta phải thường xuyên đưa các xung điện kích nạp điện cho tụ thao tác này được gọi là "làm tươi" dữ liệu chứa trong RAM.

Các bộ nhớ nói trên được xây dựng từ các ô nhớ (Memory Cell) các ô nhớ này có cấu tạo khác nhau ở các bộ nhớ khác nhau.

a) Bộ nhớ truy cập ngẫu nhiên RAM tĩnh

Cấu trúc tổng quát của RAM tĩnh: RAM được tổ chức dưới dạng ma trận các ô nhớ. Mỗi ô nhớ là một trigơ RST hoặc trigơ D. Để có thể ghi lưu trữ và đọc thông tin được dễ dàng, các ô nhớ được sắp xếp và định vị theo hàng và cột. Việc truy cập vào từng ô nhớ để đọc hoặc ghi thông tin được giải quyết nhờ khối giải mã địa chỉ hàng và giải mã địa chỉ cột. Theo nguyên tắc tổ chức nêu trên, người ta chế tạo các RAM tĩnh có $(2^n \times b)$ bit.

RAM có các lối vào địa chỉ (Address), các lối vào dữ liệu (Data), các lối vào điều khiển (Control) và các lối ra dữ liệu. Các RAM tĩnh dùng trong máy tính các đường dữ liệu vào ra thường kết hợp làm một nhờ dùng các cửa 3 trạng thái.

Trên hình 5.6.3.2 giới thiệu sơ đồ khối mô tả cấu trúc của RAM tĩnh.

Như ta đã thấy trên hình 5.6.3.2, cấu trúc của RAM gồm: ma trận các ô nhớ, khối giải mã địa chỉ và điều khiển đọc, viết, vào, ra dữ liệu.

Vi mạch RAM 6164 có 13 đường địa chỉ là: A_0,..., A_{12}, 4 đường điều khiển và 8 đường dữ liệu vào ra chung nhau là : D_0, ..., D_7 như hình 5.6.3.3.

Đầu vào điều khiển:

CS: Chip Selection: chip chọn; OE: Output Enable: đầu vào điều khiển cho phép ra.

WE: Write Enable: Đầu điều khiển cho phép ghi.

Hoạt động của nó tuân theo bảng trạng thái trên hình 5.6.3.4.

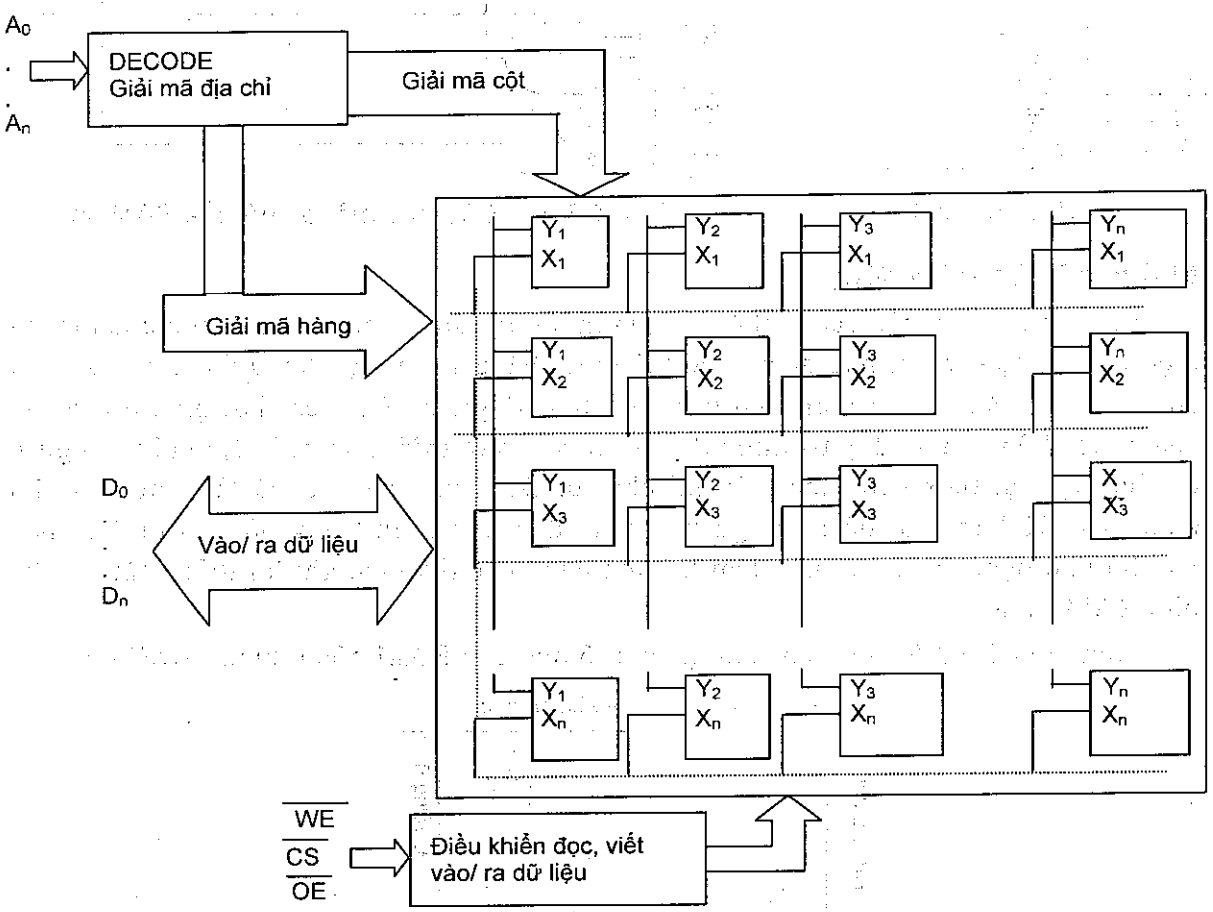
$\overline{CS}$	$\overline{WE}$	$\overline{OE}$	Mốt vận hành	Đầu vào	Đầu ra
0	0	1	Ghi số liệu	Được nối	Thả nổi
0	1	0	Đọc số liệu	Thả nổi	Được nối
0	1	1	Không làm gì	Thả nổi	Thả nổi
1	x	x	Ngừng	Thả nổi	Thả nổi

Hình 5.6.3.1. Bảng trạng thái

Trên hình 5.6.3.4 trình bày sơ đồ khối mô tả cấu trúc của một ô nhớ trong RAM có chung đường vào ra. Khối giải mã địa chỉ xác định vị trí ô nhớ cần truy cập theo

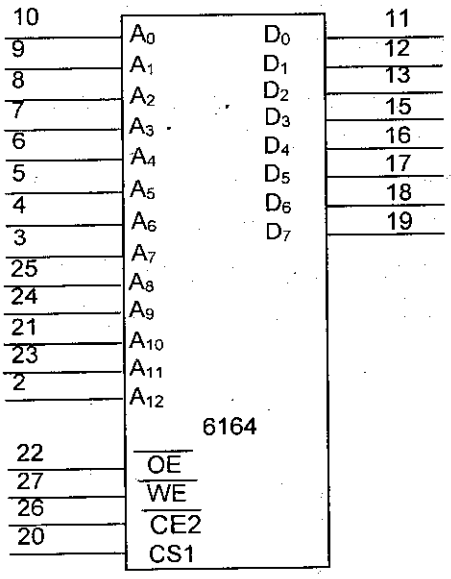
các thông tin đặt vào các lối vào địa chỉ, nó tạo ra tín hiệu cho phép truy cập vào ô nhớ. Giả sử, ta muốn ghi dữ liệu vào ô nhớ, trình tự tiến hành như sau:

$\overline{CS}=1$, $\overline{OE}=1$, $\overline{WE}=1$, lối ra hai cửa AND 1 và AND 2 ở trạng thái 0, cửa 3 trạng thái 1 và 2 thả nổi ô nhớ cách ly hoàn toàn với đường vào ra. Đưa thông tin vào lối vào địa chỉ đặt địa chỉ ô nhớ cần ghi. Đưa dữ liệu vào lối vào dữ liệu, chuyển các lối vào chọn chip về 0 và vẫn để lối vào $\overline{CS}=0$, $\overline{OE}=1$, $\overline{WE}=0$, lối ra AND 1 chuyển lên 1, lối ra AND 2 chuyển về 0 cửa 3 trạng thái 1 thông mạch cửa 3 trạng thái 2 thả nổi làm hở mạch lối ra đường vào ra trở thành lối vào dữ liệu qua cửa 3 trạng thái 1 được nạp vào ô nhớ.

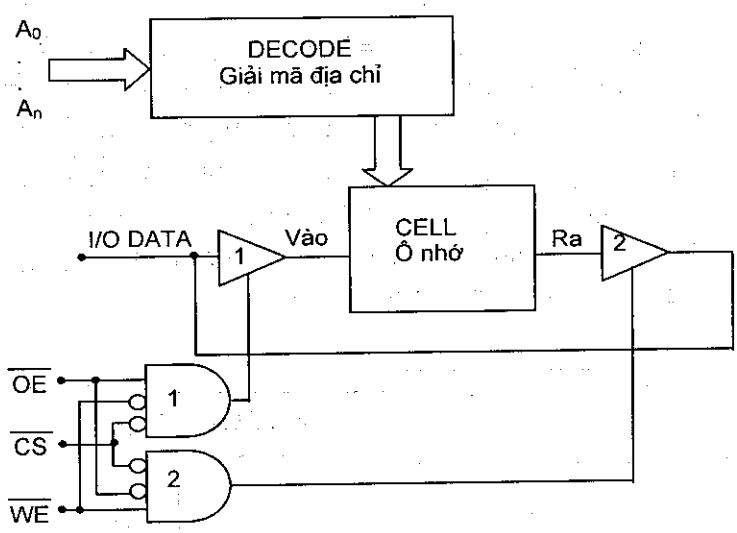


Hình 5.6.3.2. Cấu trúc của RAM

Muốn đọc dữ liệu từ ô nhớ ta cũng làm theo trình tự như thao tác ghi nhưng bước cuối ta đặt $\overline{CS}=0$, $\overline{OE}=0$, $\overline{WE}=1$, lối ra cửa AND 1 chuyển về 0, lối ra cửa AND 2 chuyển lên 1, cửa 3 trạng thái 1 thả nổi làm hở mạch lối vào, cửa 3 trạng thái 2 thông mạch dữ liệu được đưa ra qua cửa ba trạng thái 2.



Hình 5.6.3.3

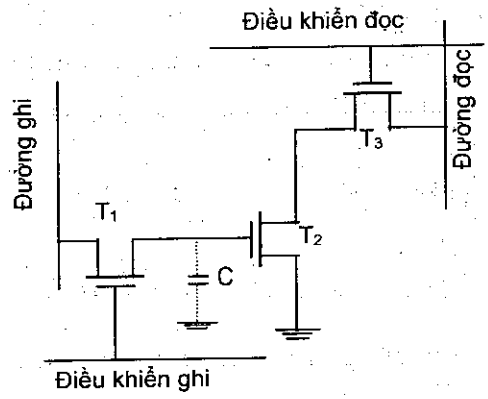


Hình 5.6.3.4. Cấu trúc một ô nhớ của RAM tĩnh

b) Các bộ nhớ RAM động

Nguyên tắc nhớ của các ô nhớ động là thông tin được lưu trữ điện tích của một tụ điện C, nếu tụ C được nạp điện ta có thông tin bằng 1, còn tụ không được nạp điện ta có thông tin bằng 0. Vì điện tích trong tụ điện sẽ giảm dần theo thời gian do trở dò của mạch điện nên để duy trì được thông tin, sau một thời gian nhất định (2ms) người ta phải nạp lại thông tin tích điện lại cho tụ. Quá trình này gọi là làm tươi bộ nhớ (refresh). Để làm tươi bộ nhớ người ta phải đọc thông tin trong bộ nhớ ra đem nạp vào bộ nhớ đệm rồi từ bộ nhớ đệm lại tiến hành các thao tác ghi lại thông tin vào bộ nhớ RAM động.

Trên hình 5.6.3.5 trình bày cấu tạo một ô nhớ của RAM động dùng MOSFET.



Hình 5.6.3.5. Cấu tạo một ô nhớ của RAM động

Mạch có 3 MOSFET, T_1, T_3 hoạt động như các khoá điện tử, các khoá này được điều khiển bằng các xung lệnh truyền qua đường điều khiển đọc, đường điều khiển

ghi. T_2 cùng T_3 tạo thành một mạch đảo. Cực cửa của T_2 tạo thành một tụ điện C. Dữ liệu nạp vào ô nhớ được lưu trữ dưới dạng điện tích nạp ở tụ điện này.

Như ta thấy trên hình 5.6.3.5 ô nhớ có 4 đường:

– Đường điều khiển đọc và điều khiển ghi thuộc về một từ (word).

– Đường "ghi" để ghi dữ liệu vào, đường "đọc" để lấy dữ liệu ra.

Thao tác ghi dữ liệu vào RAM: Giả sử T_1, T_2, T_3 đều là các tranzito MOS kênh P.

– Cho xung âm vào đường "điều khiển ghi", T_1 dẫn. Nếu đường "ghi" ở 0V tụ C không tích điện: ghi bit "0".

– Nếu đường "ghi" ở (–U), tụ C có điện tích: ghi bit "1".

Thao tác đọc:

– Cho xung âm vào đường "điều khiển đọc".

– Nếu tụ C không có điện tích (bit "0") thì T_2 cấm làm T_3 cũng cấm, đường "đọc" không có dòng điện ra: đọc "0".

– Nếu tụ C có điện tích (bit "1") thì T_2 dẫn làm T_3 cũng dẫn đường "đọc" có xung dòng điện" đọc "1".

Vì mạch luôn mất dần điện tích trên tụ điện C (do hiện tượng dò) nên dữ liệu không lưu trữ được vĩnh viễn. Do đó mạch cần được "viết" lại liên tục, chu kỳ làm tươi bộ nhớ cỡ 2ms, chính vì vậy tốc độ của RAM động chậm hơn so với RAM tĩnh.

BÀI TẬP CHƯƠNG 5

Bài 5.1. Viết bảng chân lý của trigơ JK, D, T có hai đầu vào bất đồng bộ Preset và clear tích cực ở mức thấp.

Bài 5.2. Vẽ tín hiệu Q trên các giản đồ thời gian cho ở hình 5.2BT a, b, c, d.

Bài 5.3. Thực hiện chuyển đổi:

a) Từ trigơ RS sang trigơ D, T

b) Từ trigơ JK sang trigơ RS, D, T

c) Từ trigơ D sang trigơ RS, JK, T

d) Từ trigơ T sang RS, JK, D

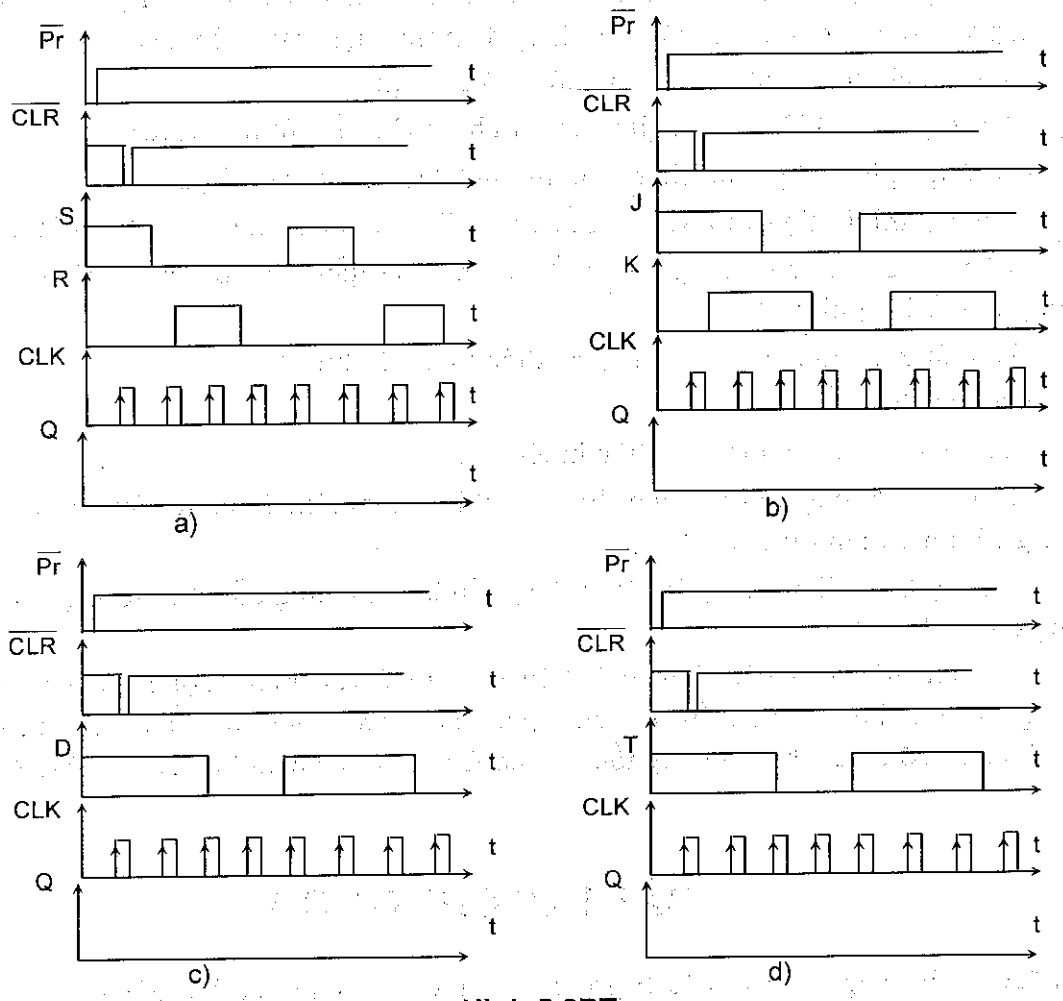
Bài 5.4. Thiết kế bộ đếm thuận nhị phân có $K_d = 8$.

Bài 5.5. Thiết kế bộ đếm nghịch nhị phân có $K_d = 8$.

Bài 5.6. Thiết kế bộ đếm Gray thuận có $K_d = 8$.

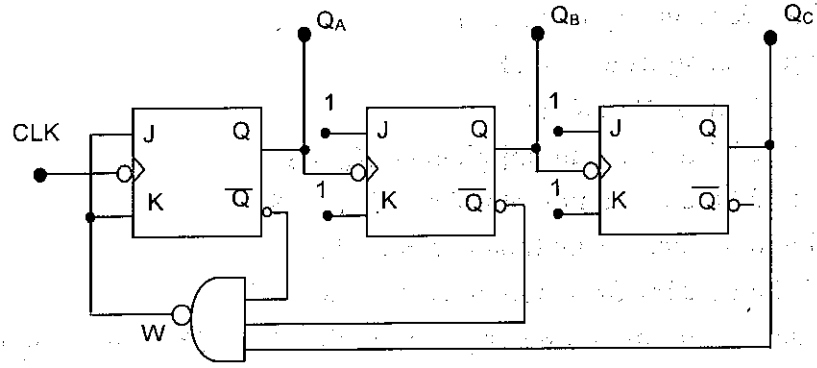
Bài 5.7. Thiết kế bộ đếm Gray nghịch có $K_d = 8$.

Bài 5.8. Giả sử bộ đếm modul 8 đang ở trạng thái 101. Cho biết trạng thái số đếm sau 13 xung áp vào?



Hình 5.2BT

Bài 5.9. Xét mạch ở hình 5.9BT. Ban đầu, tất cả đầu ra FF đều ở trạng thái 0 trước khi xung nhịp áp vào. Xác định dạng sóng tại Q_A, Q_B, Q_C và W ứng với 8 chu kỳ của đầu vào xung nhịp.

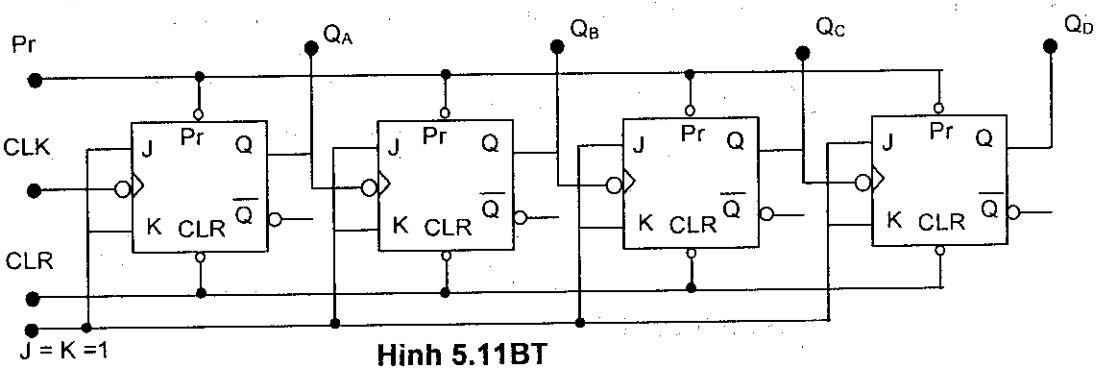


Hình 5.9BT

Bài 5.10. Một bộ đếm nhị phân được tác động bởi xung nhịp có tần số 256kHz. Tần số đầu ra từ FF cuối cùng là 2kHz.

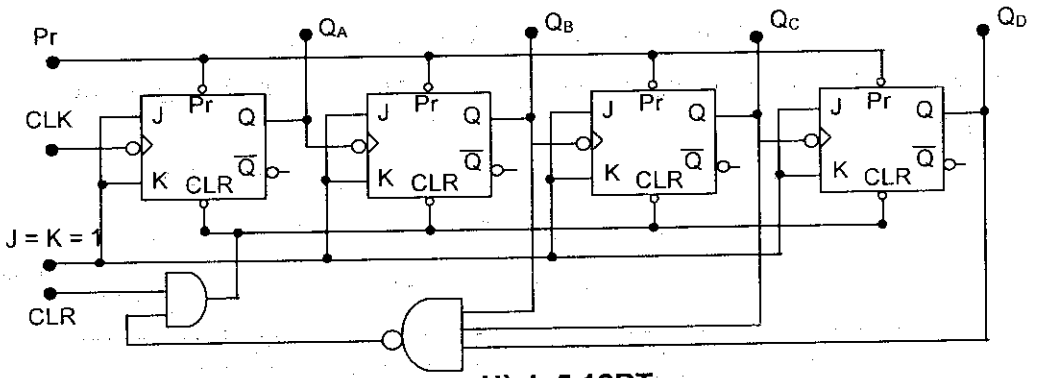
a) Tìm modul của bộ đếm ; b) Xác định khoảng đếm.

Bài 5.11. Bộ đếm ở hình 5.11BT bắt đầu ở trạng thái 0000, sau đó xung nhịp được đưa vào. Sau một thời gian, xung nhịp bị ngắt và FF bộ đếm hiển thị 0011. Có bao nhiêu xung nhịp đã xảy ra?



Hình 5.11BT

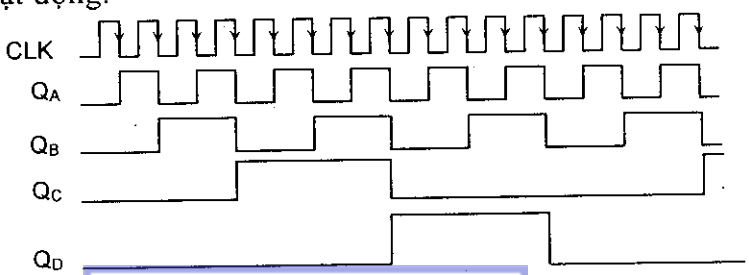
Bài 5.12. Xác định modul của bộ đếm trong hình 5.12BT và tần số tại đầu ra Q_D khi tần số xung nhịp là 30Hz.



Hình 5.12BT

Bài 5.13. Xây dựng bộ đếm modul 60 để chia tần số đường dây điện 60Hz xuống thành 1Hz.

Bài 5.14. Cho giản đồ thời gian của một bộ đếm hình 5.14BT, vẽ sơ đồ của bộ đếm đó và giải thích hoạt động.



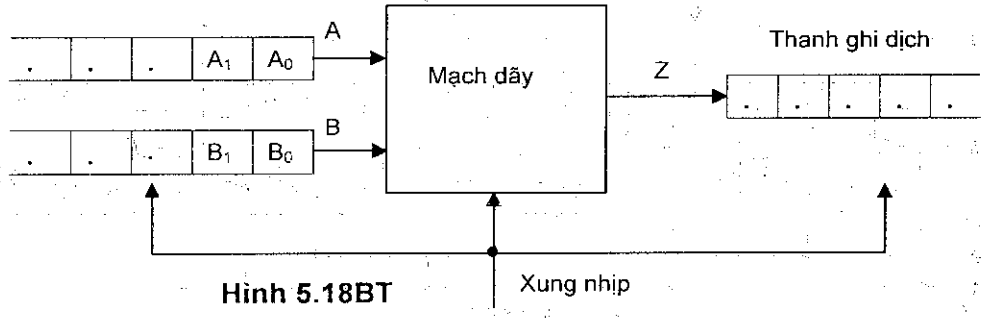
Hình 5.14BT

Bài 5.15. Vẽ sơ đồ bộ ghi dịch nối tiếp 4 bit vừa dịch phải vừa dịch trái.

Bài 5.16. Nội dung thanh ghi nối tiếp dịch phải ABCD = 0101, với D là cột có trọng số nhỏ nhất. Dãy dữ liệu vào là 10011 (bit cuối cùng là bit có trọng số nhỏ nhất), được nạp tuần tự vào thanh ghi. Hãy vẽ đồ thị thời gian của bốn đầu ra của các FF: A, B, C, D sau 5 xung nhịp.

Bài 5.17. Thiết kế bộ đếm $K_d = 12$ dùng thanh ghi dịch và mạch hồi tiếp.

Bài 5.18. Thiết kế một bộ cộng nhị phân nối tiếp cho hai số hạng theo hình 5.18BT:

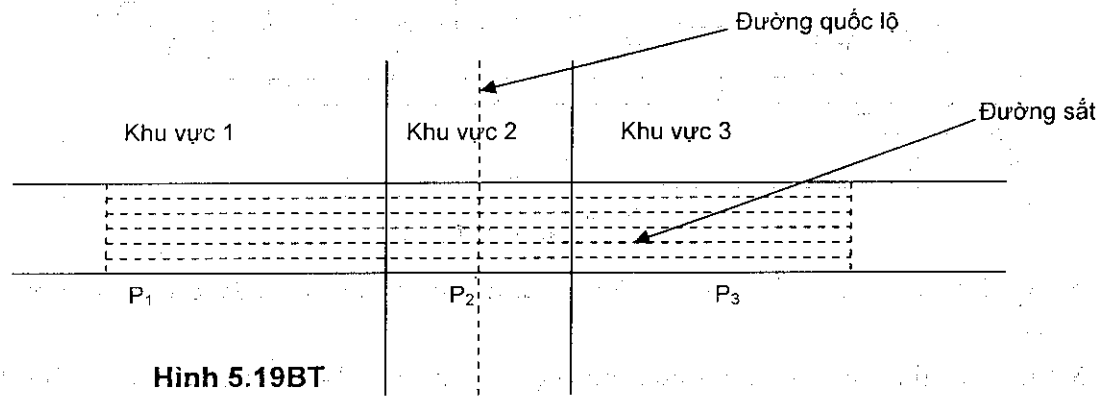


Hình 5.18BT

Bài 5.19. Thiết kế mạch điều khiển một hệ thống báo hiệu ở một ngã tư giữa đường tàu và đường giao thông hình 5.19BT. Hệ thống báo hiệu gồm chuông (c) và đèn (đ) hoạt động như sau:

Đèn sáng và chuông kêu khi xe lửa vào khu vực 1 và 3.

Đèn được tắt và chuông ngừng kêu khi xe lửa đi qua ngã tư giữa đường quốc lộ và đường sắt.



Hình 5.19BT

Chương 6

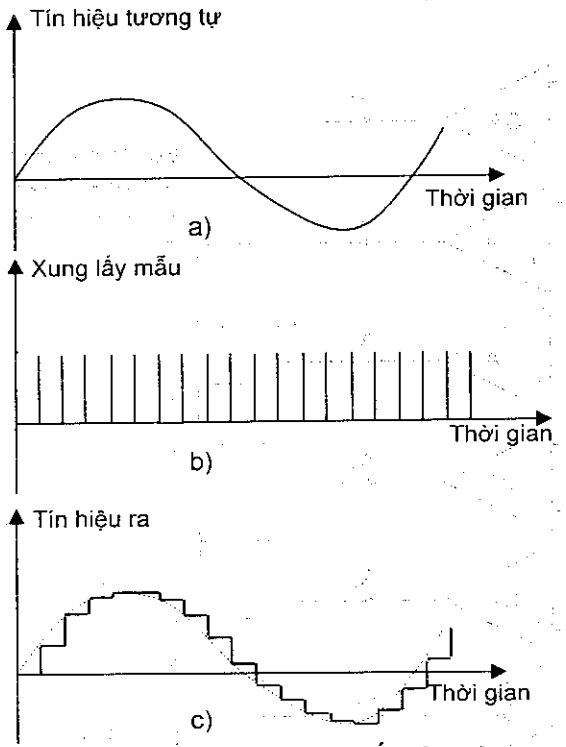
BỘ CHUYỂN ĐỔI TƯƠNG TỰ – SỐ VÀ SỐ – TƯƠNG TỰ

6.1. BỘ CHUYỂN ĐỔI TƯƠNG TỰ – SỐ ADC

Bộ biến đổi tương tự số ADC (Analog To Digital Converter) là mạch biến đổi tín hiệu tương tự thành tín hiệu số có mã số tỷ lệ với giá trị biên độ tín hiệu tương tự ở lối vào. Giá trị của mã nhị phân ở lối ra của DAC biểu diễn độ lớn của tín hiệu tương tự tại thời điểm thực hiện phép biến đổi.

6.1.1. Mạch lấy mẫu và duy trì mẫu

Một vài loại DAC đòi hỏi tín hiệu giữ nguyên không đổi trong thời gian thực hiện quá trình biến đổi, còn đối với một số khác điều kiện trên không cần thiết. Đối với loại DAC đòi hỏi tín hiệu không đổi trong quá trình biến đổi thì có thể dùng mạch lấy mẫu và duy trì mẫu đặt vào giữa lối vào của bộ DAC.

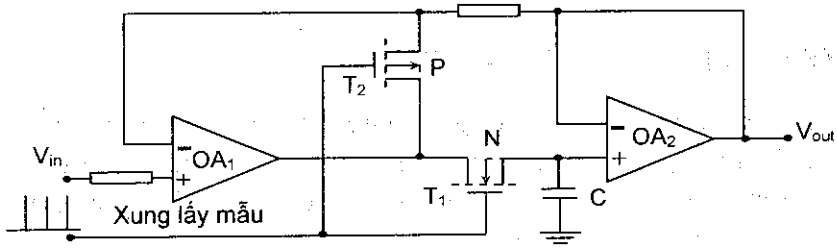


Hình 6.1.1.1. Tín hiệu lối vào và ra

Mạch duy trì mẫu và lấy mẫu có chức năng sau: lấy mẫu ở những thời điểm xác định và duy trì giá trị đó cho đến thời điểm lấy mẫu tiếp theo.

Hình 6.1.1.1 trình bày dạng tín hiệu lối vào và ra của mạch lấy mẫu và duy trì mẫu. Trong đó: hình 6.1.1.1a là tín hiệu tương tự, hình 6.1.1.1b là dãy xung điều khiển hoạt động bộ lấy mẫu, hình 6.1.1.1c là tín hiệu ra của bộ lấy mẫu.

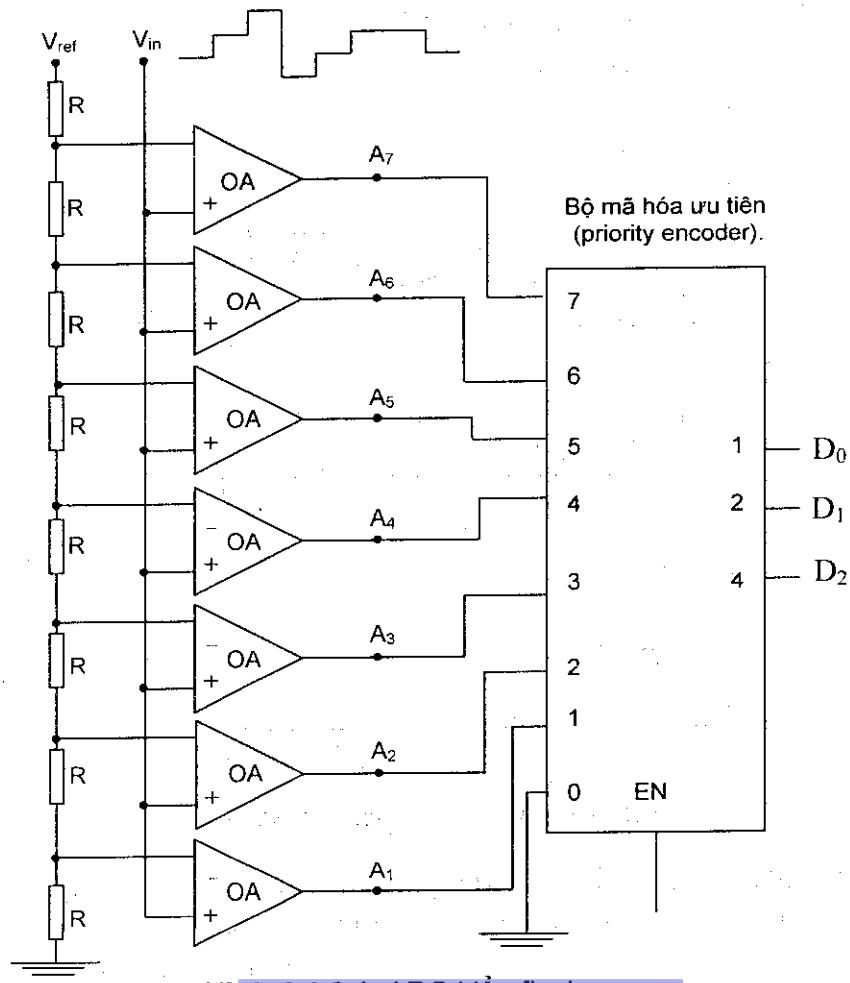
Một mạch duy trì mẫu và lấy mẫu được sử dụng phổ biến có sơ đồ như hình 6.1.1.2. Trong sơ đồ 6.1.1.2, quá trình trích mẫu và duy trì được thực hiện bởi hai tranzito loại FET hoặc MOSFET: T_1 và T_2. Khi lấy mẫu thì T_1 thông, T_2 ngắt. Trạng thái duy trì được thực hiện khi T_1 ngắt T_2 thông.



Hình 6.1.1.2. Mạch duy trì mẫu và lấy mẫu

6.1.2. Các loại biến đổi ADC

Có nhiều loại ADC tuỳ theo kết cấu của nó: ADC tức thời (ADC kiểu flash), ADC kiểu tích phân, ADC kiểu servo.



Hình 6.1.2.1. ADC kiểu flash

6.1.2.1. ADC kiểu flash

Sơ đồ bộ biến đổi tương tự số theo kiểu flash được trình bày trên hình 6.1.2.1.

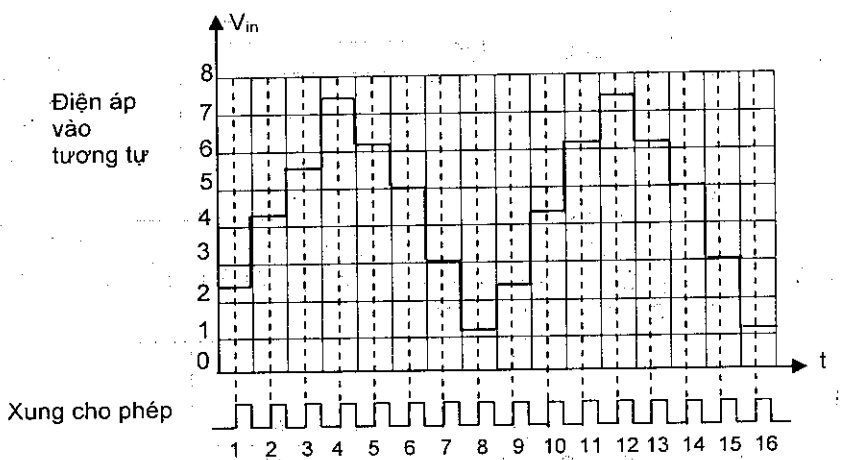
Trong sơ đồ hình 6.1.2.1, V_{in} là điện áp vào lấy từ đầu ra của bộ lấy mẫu và duy trì mẫu; EN là các xung cho phép; D_0, D_1, D_2 là các đầu ra nhị phân song song.

Đây là loại biến đổi A/D đơn giản nhất và có tốc độ nhanh nhất. Nếu bộ biến đổi có n bit ở lối ra thì ta dùng n mạch so sánh tương tự ở lối vào để so sánh tín hiệu V_A với các mức điện áp chuẩn: ΔV_A, $2\Delta V_A$, ..., $n\Delta V_A = V_{Amax}$. Với V_{Amax} là giới hạn thang đo của bộ biến đổi A/D, ΔV_A được gọi là độ phân giải. Như vậy, ở đây thang đo V_{Amax} được chia làm n mức và khoảng cách giữa các mức $\Delta V_A = V_{Amax}/n$. Biên độ của tín hiệu tương tự V_A được xác định bằng giá trị nằm giữa hai mức nào đó. Giả sử nếu V_A nằm giữa hai giá trị $10\Delta V_A$ và $11\Delta V_A$ thì 10 bộ so sánh tương tự sẽ cho giá trị ở mức cao còn các bộ khác sẽ cho giá trị ở mức thấp. Bộ giải mã ưu tiên sẽ cho kết quả là 10 và tạo ra mã nhị phân 1010 ở lối ra.

Tốc độ của bộ biến đổi A/D tức thời rất nhanh, nó chỉ phụ thuộc vào tốc độ phản ứng của các bộ so sánh tương tự mà thôi.

Nhược điểm của loại này là kích thước rất cồng kềnh, nếu lối ra cần n bit thì lối vào cần 2^n-1 bộ so sánh tương tự. Nếu n = 8 thì ta cần có 255 bộ so sánh tương tự, và bộ giải mã ưu tiên với 255 lối vào cũng rất phức tạp.

Ví dụ 6.1.2.1. Hãy xác định mã nhị phân ở đầu ra của ADC 3 bit cho trên hình 6.1.2.1, biết tín hiệu vào và xung cho phép như trên hình 6.1.2.2, V_{ref} = 8V.



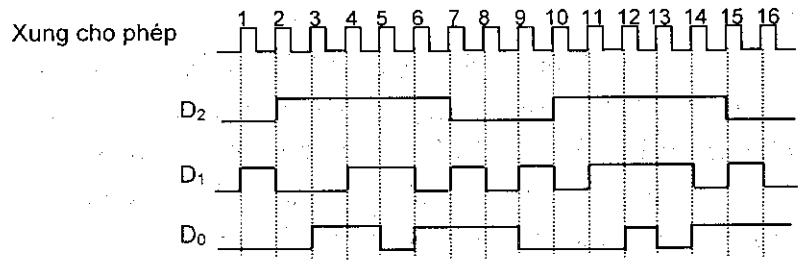
Hình 6.1.2.2

Giải:

Mã nhị phân nhận được tại đầu ra song song tương ứng $D_2D_1D_0$ tuần tự là:

010, 100, 101, 111, 110, 101, 011, 001, 010, 100, 110, 111, 110, 101, 011, 001.

Dạng sóng của tín hiệu ra theo các xung cho phép được biểu diễn trên hình 6.1.2.3.



Hình 6.1.2.3

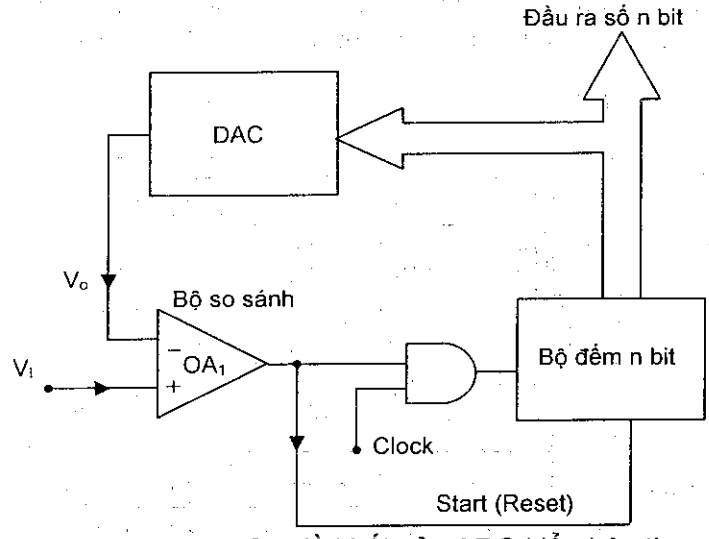
6.1.2.2. ADC kiểu bậc thang (kiểu Servo)

Trên hình 6.1.2.4 giới thiệu sơ đồ khối của bộ ADC kiểu bậc thang. Nguyên tắc hoạt động của nó như sau:

Chu trình biến đổi bắt đầu khi xung start xoá bộ đếm nhị phân n bit. Với V_o nhỏ hơn V_i lối ra bộ so sánh ở mức 1, cổng AND mở cho các xung Clock vào bộ đếm. Số đếm tăng dần tới khi V_o bắt đầu vượt quá V_I, lối ra của bộ so sánh sẽ trở về 0 và khoá cổng AND lại.

Mã số lối ra của bộ đếm lúc này tương ứng với độ lớn điện thế tương tự cần biến đổi. Nếu đo dạng sóng V_o trong một chu kỳ biến đổi, ta sẽ thấy một sóng hình bậc thang.

ADC loại này có kết cấu đơn giản nhưng có nhược điểm là thời gian biến đổi phụ thuộc vào độ lớn điện thế cần biến đổi.



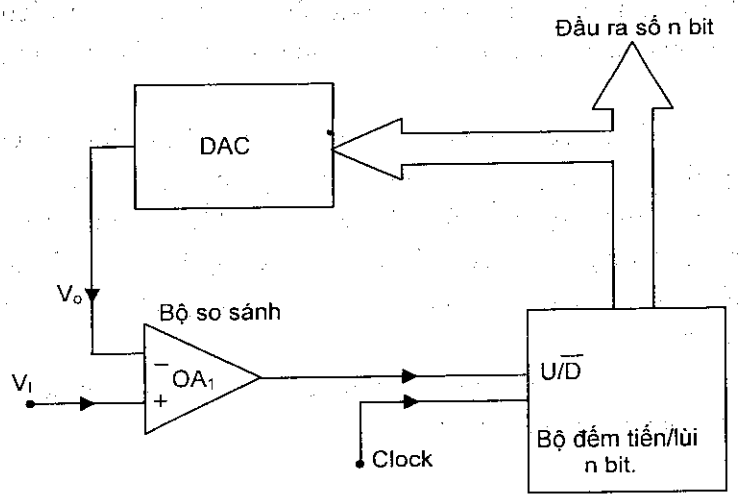
Hình 6.1.2.4. Sơ đồ khối của ADC kiểu bậc thang

6.1.2.3. ADC bám sát (Tracking)

Trên hình 6.1.2.5 vẽ sơ đồ khối của ADC bám sát.

Nếu giá trị V_I chỉ biến đổi quanh một giá trị nào đó thì loại ADC này tiện lợi hơn. nguyên tắc của nó là dùng bộ đếm lên/xuống. Mạch được thiết kế sao cho nếu $V_o < V_I$ (điện thế lối ra của bộ so sánh bằng 1), bộ đếm sẽ ở trạng thái đếm lên. Nếu $V_o > V_I$ (thế lối ra của bộ so sánh bằng 0) thì bộ đếm sẽ ở trạng thái đếm xuống. Như vậy thế

lối ra của DAC có xu hướng bám sát thế lối vào cần biến đổi.



Hình 6.1.2.5. Sơ đồ khối của ADC bám sát

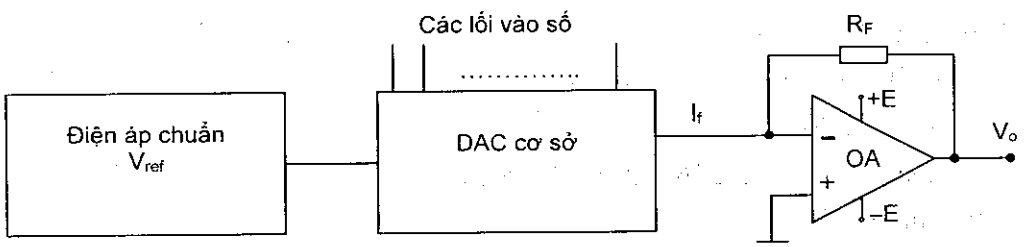
6.2. KHÁI QUÁT VỀ BỘ CHUYỂN ĐỔI SỐ – TƯƠNG TỰ DAC

6.2.1. Chức năng

DAC (Digital To Analog Converter) tiếp nhận một mã số n bit song song ở lối vào và biến đổi nó ra dòng điện hoặc điện áp tương tự ở lối ra. Dòng điện hoặc điện áp ở lối ra DAC là hàm số của mã số lối vào và phải biến thiên phù hợp với sự biến thiên của mã số này.

6.2.2. Cấu trúc

Sơ đồ khối của bộ biến đổi DAC hoàn chỉnh bao gồm 3 phần tử cơ bản như hình 6.2.2.1: một điện áp chuẩn (Referent Voltage) ổn định bên ngoài, một DAC cơ sở và một mạch khuếch đại thuật toán.



Hình 6.2.2.1. Sơ đồ khối của bộ biến đổi DAC

Với một mã số nhị phân tự nhiên ở lối vào của DAC, ta có điện áp V_0 ở lối ra là:

$$V_o = - V_{ref}(B_1.2^{-1} + B_2 2^{-2} ..... + B_n 2^{-n}) \qquad (6.2.2.1)$$

Trong đó: B_n là bit có trọng số nhỏ nhất

B_1 là bit có trọng số lớn nhất

V_{ref} là điện áp chuẩn.

DAC cơ sở được tạo thành từ những chuyển mạch tương tự được điều khiển bởi mã số ở lối vào và một loạt điện trở chính xác. Nhờ có các chuyển mạch tương tự điều khiển dòng hoặc điện áp trích từ điện áp chuẩn tạo nên dòng điện hoặc điện áp lối ra tương tự với mã số đó.

Mạch khuếch đại thuật toán dùng để chuyển đổi dòng điện thành điện áp và đồng thời có chức năng của tầng đệm. Nhưng do có một khoảng thời gian trễ trong quá trình thiết lập trạng thái ở lối ra theo sự biến đổi dòng điện ở lối vào của bộ khuếch đại thuật toán cho nên trong các ứng dụng cần tốc độ cao, có thể dùng một điện trở thay thế cho OA để thực hiện việc chuyển đổi này. Tuy nhiên cũng vì thế mà phạm vi biến đổi số – tương tự bị giới hạn.

DAC được sử dụng nhiều trong việc điều khiển nguồn nuôi cho các thiết bị kiểm tra tự động, các bộ tạo sóng và các bộ phận điều khiển quá trình.

6.2.3. Độ chính xác

Ta thấy rằng đại lượng ra tương tự không liên tục mà nhận một giá trị rời rạc tương ứng với 1 trong 2^n tổ hợp của các mã nhị phân n bit tại lối vào số của một bộ DAC n bit.

Từ công thức (6.2.2.1) ta suy ra rằng, giá trị cực đại của lối ra không đạt tới giá trị lớn nhất của thang trị số FS (Full scale) mà chỉ có thể đạt tới giá trị $FS.(2^n - 1)/2^n$.

Độ lớn của mỗi đơn vị điện áp ra được gọi là độ phân giải và ứng với bit có trọng số nhỏ nhất LSB, nó có giá trị là $FS/2^n$.

Một DAC có số bit càng lớn thì số mức điện áp ra càng lớn và giá trị của mỗi nấc càng nhỏ, như vậy tập hợp các giá trị của đại lượng tương tự càng liên tục và bộ DAC có độ chính xác càng cao.

Độ chính xác của DAC bằng $1/2^n$ với n là số bit.

Ví dụ 6.2.3.1. Tìm giá trị lớn nhất của mức điện thế đầu ra từ 1 DAC 8 bit, biết rằng ứng với đầu vào số 00110010 mức điện thế sinh ra là 1V.

Giải:

Ta có: $(00110010)_2 = (50)_{10}$

Do đó: 1V = FS.50/256

FS = 256/50

Vậy: $V_{out}(max) = FS.255/256 = 255/50 = 5.1V$

Độ phân giải là 1/50.

Độ chính xác: 1/256

6.2.4. Các loại mã số dùng cho DAC

Các loại mã số dùng cho DAC gồm có mã nhị phân tự nhiên, mã bù hai, mã BCD. Ở các bộ DAC có cùng số bit, bộ DAC sử dụng mã BCD sẽ có độ chính xác kém hơn khi dùng mã nhị phân tự nhiên. Độ chính xác của bộ DAC dùng mã BCD được tính bằng $1/10^D$ trong đó D là tổng các chữ số.

Ví dụ 6.2.4.1. Với bộ DAC có 12 bit

Dùng mã BCD độ chính xác là: $1/10^3 = 0{,}001$

Dùng mã nhị phân tự nhiên độ chính xác là: $1/2^{12} = 0{,}00024$.

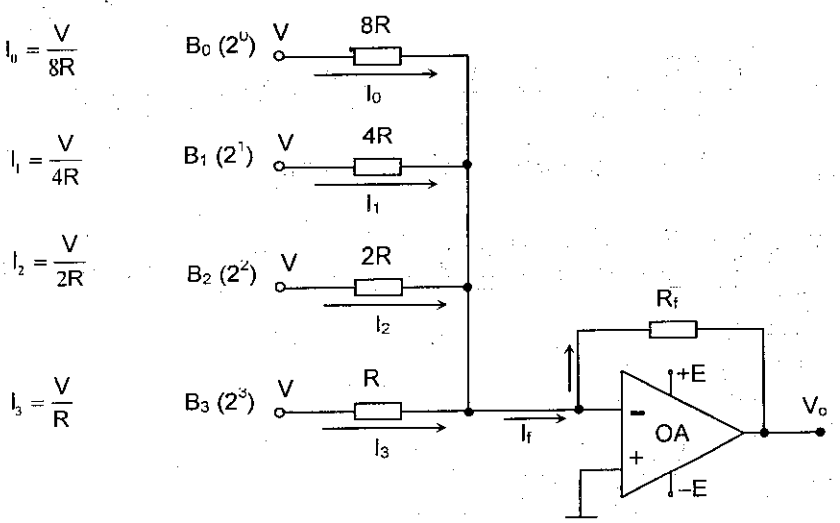
Dùng mã nhị phân tự nhiên ta có độ chính xác lớn gấp 4 lần độ chính xác dùng mã BCD.

Do đó, những DAC dạng vi mạch thường sử dụng mã nhị phân hoặc mã bù hai.

6.2.5. Các loại DAC

6.2.5.1. DAC dùng mạng điện trở trọng số

Trên hình 6.2.5.1 trình bày sơ đồ nguyên lý của DAC 4 bit dùng mạng điện trở trọng số. Trong phương pháp này để thực hiện biến đổi số – tương tự, người ta tạo ra một dòng điện I_f là tổng các dòng thành phần tương ứng I_k được chọn tương thích với mã số ở lối vào nhờ sự điều khiển bởi trạng thái các bit của mã số. Nếu điện áp vào là 0V (bit 0) thì I_k tương ứng là 0A, nếu điện áp vào là mức cao (bit 1) thì giá trị I_k phụ thuộc vào giá trị điện trở vào. Giá trị điện trở phụ thuộc vào trọng số của các bit đầu vào như hình vẽ.



Hình 6.2.5.1. DAC 4 bit dùng mạng điện trở trọng số

Dòng I_f tỷ lệ với mã số lối vào được chuyển thành điện áp ra tỷ lệ với mã số nhị phân ở lối vào nhờ mạch khuếch đại thuật toán mắc theo kiểu cộng đảo pha. Điện áp ở lối ra của bộ DAC chính là điện áp ở lối ra của bộ khuếch đại thuật toán.

Các dòng thành phần I_K được xác định theo các giá trị các điện trở trọng số R, 2R, 4R, 8R và các bit nhị phân B_k (B_k = 0 hoặc 1) theo các hệ thức sau đây:

$$I_0 = B_0.\ V/R;$$

$$I_1 = B_1.\ V/2R;$$

$$I_2 = B_2.\ V/4R;$$

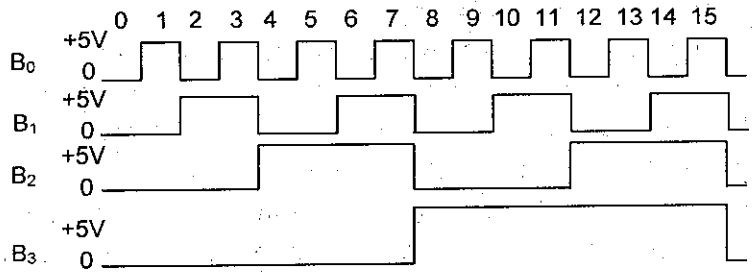
$$I_3 = B_3.\ V/8R.$$

Từ sơ đồ ta có: $V_o = -I_f R_f$ (6.2.5.1)

$$I_f = \frac{V}{R}\left(\frac{B_0}{8} + \frac{B_1}{4} + \frac{B_2}{2} + \frac{B_3}{1}\right) \quad (6.2.5.2)$$

$$V_o = -V\frac{R_f}{R}\left(\frac{B_0}{8} + \frac{B_1}{4} + \frac{B_2}{2} + \frac{B_3}{1}\right) \quad (6.2.5.3)$$

Ví dụ 6.2.5.1. Xác định điện áp ra của DAC trên hình 6.2.5.1 với R = 25kΩ, R_f = 10kΩ nếu dạng sóng của bốn bit vào cho trên hình 6.2.5.2.

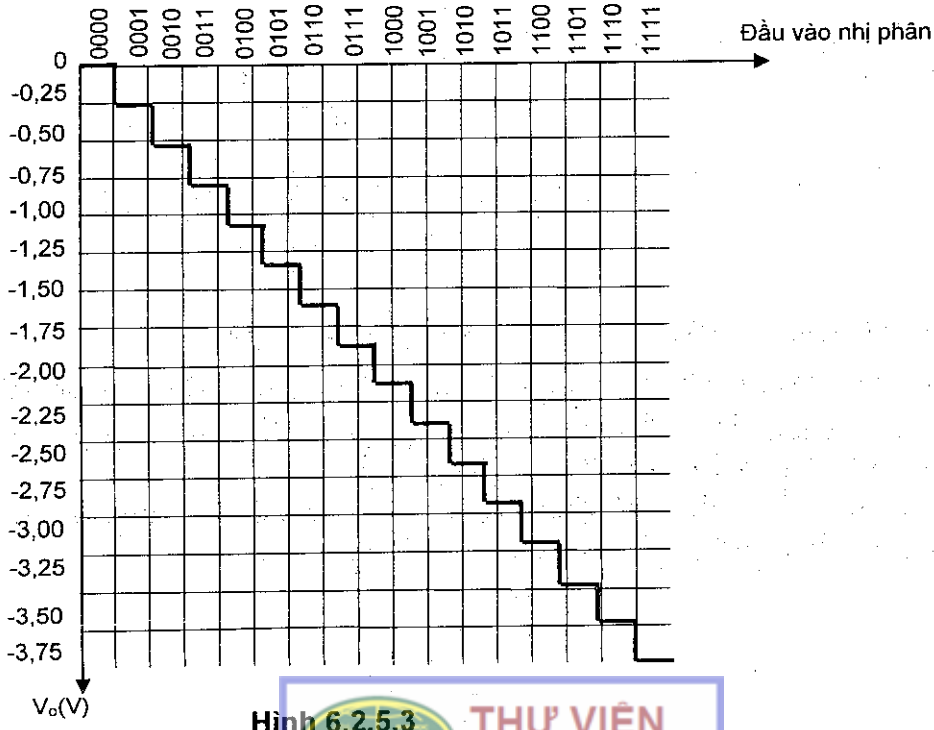


Hình 6.2.5.2

Giải:

Ta có: $V_0 = -V\frac{R_f}{R}\left(\frac{B_0}{8} + \frac{B_1}{4} + \frac{B_2}{2} + \frac{B_3}{1}\right) = -2\left(\frac{B_0}{8} + \frac{B_1}{4} + \frac{B_2}{2} + \frac{B_3}{1}\right)(V)$

Điện áp ra thay đổi theo tín hiệu vào cho trên hình 6.2.5.3.

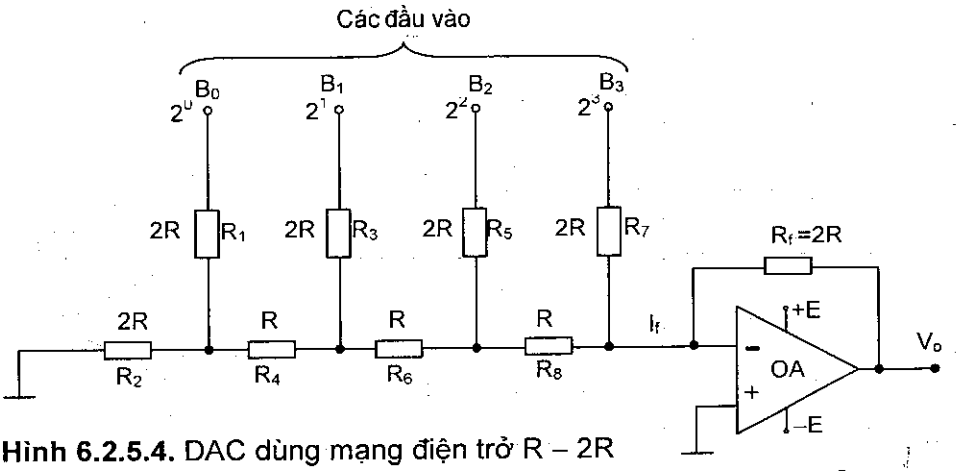


Hình 6.2.5.3

Như vậy, điện áp ra tỷ lệ thuận với mã số lối vào theo hệ số tỷ lệ là điện áp chuẩn. Nhược điểm của mạch này là số bit càng tăng thì số điện trở khác nhau về giá trị cũng tăng, việc chọn các điện trở chính xác càng khó khăn hơn. Để khắc phục nhược điểm này người ta đã đưa ra loại DAC dùng mạng điện trở R – 2R, loại này chỉ dùng có hai loại điện trở.

6.2.5.2. DAC dùng mạng điện trở R – 2R

Khác với DAC dùng điện trở trọng số, mạch DAC sử dụng mạng điện trở R–2R chỉ cần dùng 2 loại giá trị điện trở. Nhưng so với các DAC dùng điện trở trọng số có cùng số bit thì số lượng điện trở đòi hỏi phải nhiều hơn. Sơ đồ nguyên lý bộ DAC dùng mạng điện trở R – 2R được vẽ trên hình 6.2.5.4.



Hình 6.2.5.4. DAC dùng mạng điện trở R – 2R

Qua mỗi nút mạng của điện trở dòng điện lại giảm đi một nửa đúng với quy luật của mã nhị phân như trong hình vẽ. Chỉ bằng mạng điện trở R – 2R, cộng thêm bộ khuếch đại thuật toán, ta có thể xây dựng được một bộ chuyển đổi DAC. Điện trở phản hồi âm của mạch khuếch đại thuật toán nếu chọn đúng bằng R thì vùng biến thiên của điện áp ra sẽ phù hợp với kết quả tính toán cho DAC dùng điện trở trọng số. Điện áp ra trong trường hợp này cũng được xác định theo hệ thức (6.2.5.1). Tuỳ theo yêu cầu cụ thể ta có thể chọn giá trị của nó xung quanh giá trị R.

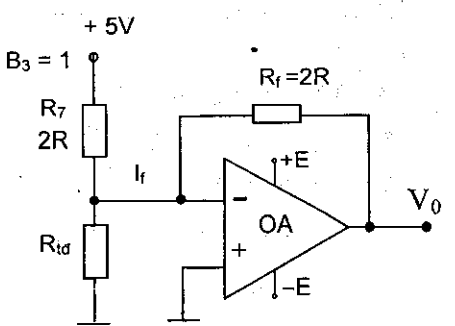
Ví dụ 6.2.5.2. Xác định giá trị điện áp ra V_0 nếu số nhị phân ở đầu vào là 1000, 0100, 0010, 0001.

Giải:

Với $B_3 = 1$, $B_2 = B_1 = B_0 = 0$ ta có sơ đồ tương đương như trên hình 6.2.5.5.

Do điện áp đưa vào các đầu vào B_2, B_1 và B_0 là 0V nên $R_{td} = 2R$. Dòng qua R_{td} bằng 0 nên dòng qua R_7 chính là dòng I_f. Vậy ta có:

$$V_o = -I_f.R_f = -\frac{5V}{2R}.2R = -5V$$



Hình 6.2.5.5.

Tính toán tương tự ta có:

Với $B_3 = 0, B_2 = 1, B_1 = B_0 = 0$ thì $V_o = -2,5V$.

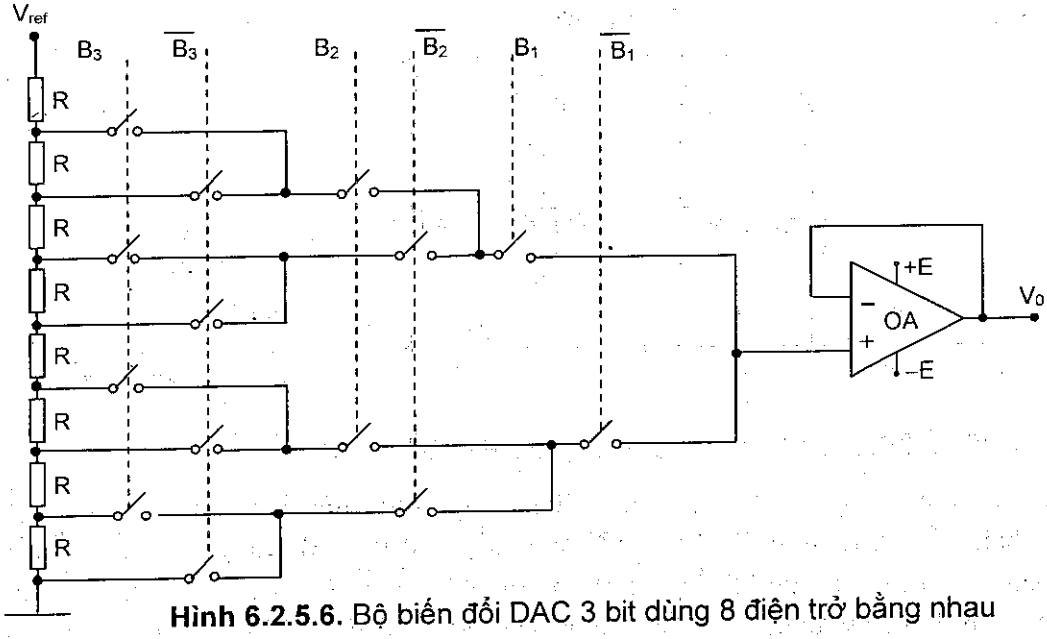
Với $B_3 = B_2 = 0, B_1 = 1, B_0 = 0$ thì $V_o = -1,25V$.

Với $B_3 = B_2 = B_1 = 0, B_0 = 1$ thì $V_o = -0,625V$.

6.2.5.3. DAC dùng 2^n điện trở bằng nhau

DAC loại này có thể được xây dựng bằng cách sử dụng 2^n điện trở có cùng 1 trị số R. Trong đó n là số bit của bộ DAC.

Ngoài số điện trở trên, DAC loại này cần đòi hỏi có $(2^n - 1)$ chuyển mạch điện tử. Trên hình 6.2.5.6 trình bày sơ đồ nguyên lý bộ biến đổi DAC 3 bit thuộc loại này.



Hình 6.2.5.6. Bộ biến đổi DAC 3 bit dùng 8 điện trở bằng nhau

Do phương pháp này đòi hỏi phải sử dụng một số lượng điện trở và chuyển mạch khá lớn nên DAC loại này được sản xuất theo công nghệ MOS – LSI (vi mạch cỡ lớn MOS). Các điện trở được ghép nối tiếp với nhau để tạo thành mạch chia điện áp. Mạch này tách điện áp chuẩn V_{ref} thành 2^n mức. Các chuyển mạch được ghép nối theo hình cây và được điều khiển bằng mã số ở lối vào. Mạch khuếch đại thuật toán ở đây được mắc theo sơ đồ lặp lại điện áp.

6.2.5.4. DAC hai biến số

Cấu trúc DAC hai biến số được trình bày trên hình 6.2.5.7.

Từ hai công thức (6.2.5.1) và (6.2.5.2) ta có:

$$V_o = -V_{ref}.(B_1.2^{-1} + B_2 2^{-2} ..... + B_n 2^{-n})$$

$$V_o = -I_f R$$

Khi giá trị của điện áp chuẩn V_{ref} thay đổi, giá trị của điện áp ra V_0 cũng thay đổi. Nếu ở các lối vào của bộ DAC giữ ở mã cố định thì điện áp ra thay đổi tuyến tính với sự thay

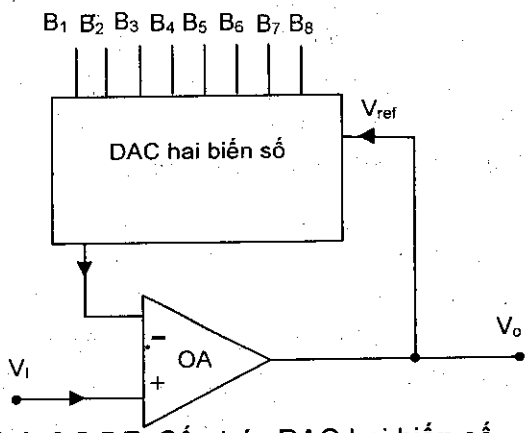
đổi của điện áp chuẩn V_{ref}. DAC làm việc với cả hai đại lượng vào là mã số và điện áp chuẩn được gọi là DAC hai biến số.

DAC hai biến số dùng trong những hệ vi xử lý để điều khiển độ nhạy của bộ khuếch đại bằng phương pháp số.

Ở đây DAC hai biến số đóng vai trò điều chỉnh mạch phản hồi âm mắc cho mạch khuếch đại thuật toán.

Gọi F là hệ số phản hồi âm được xác định theo mã số vào. Ta có:

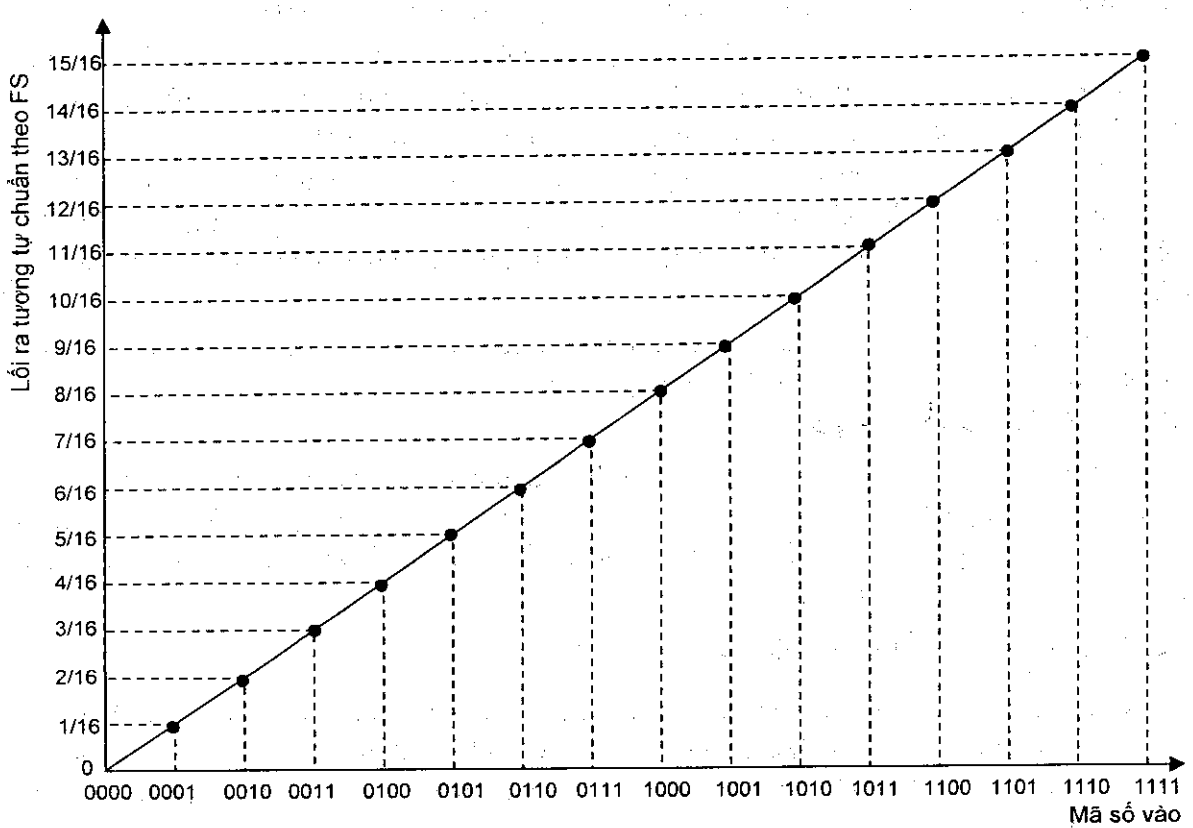
$V_I = F.V_0$; $V_0 = V_I/F$



Hình 6.2.5.7. Cấu trúc DAC hai biến số

6.2.6. ĐẶC TÍNH CỦA DAC

6.2.6.1. Đặc tính chuyển đổi số tương tự của DAC



Hình 6.2.6.1. Đặc trưng chuyển đổi số tương tự của DAC 4 bit

Đặc trưng chuyển đổi số tương tự của DAC là một đồ thị biểu diễn sự phụ thuộc điện áp lối ra của DAC vào các mã số lối vào. Trong trường hợp lý tưởng nó là đường thẳng qua điểm 0 gốc toạ độ (tương ứng với mã số lối vào DAC và điện áp ra DAC đều bằng 0)

và điểm ứng với điện áp ra cực đại và các bit của lối vào DAC đều bằng 1. Hình 6.2.6.1 là đặc trưng chuyển đổi số tương tự của DAC 4 bit.

Xác lập được đặc trưng chuyển đổi số tương tự cho phép chúng ta đánh giá được phẩm chất của DAC. Chất lượng của một bộ DAC có thể được đánh giá thông qua sai số của nó.

6.2.6.2. Những sai số của DAC

Sai số độ lệch (Offset Error), sai số khuếch đại (Gain Error), sai số tuyến tính (Linearity), sai số phi tuyến tính vi sai (Differential Nonlinearity). Các sai số này được biểu diễn theo đơn vị %FS hoặc %LSB.

Sai số độ lệch là giá trị điện áp ra của bộ DAC khi lối vào là mã ứng với 0. Sai số này được tính bằng mV hoặc %LSB hoặc %FS. Để hiệu chỉnh giảm nhỏ sai số độ lệch, ta đưa mã ứng với 0 vào lối vào của bộ DAC và điều chỉnh chiết áp độ lệch sao cho điện áp lối ra là 0V. Đây là quá trình chỉnh điểm 0 của DAC.

Sai số khuếch đại là độ lệch giữa điện áp ra tính theo lý thuyết và thực tế khi lối vào ở một mã số nào đó, thường là mã số ứng với FS – 1LSB. Sai số khuếch đại được hiệu chỉnh sau khi đã hiệu chỉnh sai số độ lệch bằng cách đưa mã ứng với FS – 1LSB tới lối vào rồi điều chỉnh chiết áp độ nhạy sao cho điện áp ra có giá trị là (FS – 1LSB).

Các sai số độ lệch và độ khuếch đại phụ thuộc vào sự thay đổi nhiệt độ của môi trường. Khi nhiệt độ môi trường thay đổi, các sai số này lại xuất hiện. Giá trị này thường vào cỡ phần nghìn cho 1^0C. Đó là sự nhạy cảm về nhiệt độ của DAC.

Sai số tuyến tính là loại sai số không thể hiệu chỉnh được và là do thiết kế của bộ DAC. Mức tuyến tính của DAC cho biết độ lệch điện áp ra so với một đường thẳng đi qua những điểm nút của đặc tuyến chuyển đổi. Mức độ tuyến tính của DAC phải nhỏ hơn hoặc bằng 1/2LSB.

Sai số phi tuyến vi sai là đại lượng cho biết độ lệch giữa giá trị thực tế và đặc trưng lý tưởng của một mạch DAC ứng với mọi thay đổi của mã số vào.

6.2.6.3. Thời gian thiết lập

Thời gian thiết lập đối với một DAC là thời gian cần thiết để điện áp ra đạt tới giá trị ổn định cuối cùng khi có sự thay đổi mã số lối vào. Thời gian thiết lập trước hết phụ thuộc vào kiểu chuyển mạch, kiểu điện trở và kiểu khuếch đại dùng để xây dựng DAC. Thời gian thiết lập không bao hàm thời gian trễ.

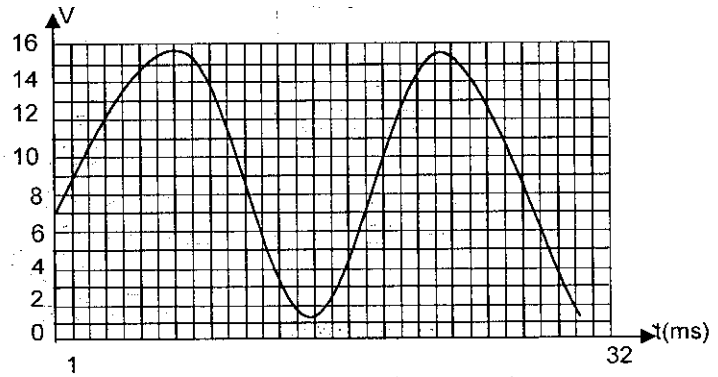
6.2.7. Điện áp chuẩn của DAC

Nguồn điện áp chuẩn là thành phần hết sức quan trọng cho hoạt động của DAC. Có loại DAC được cấy điện áp chuẩn bên trong được trích từ điện áp nuôi và trong mạch đã cấy diode ổn áp có ổn nhiệt. Nhưng các điện áp chuẩn bên ngoài vẫn được sử dụng để đạt độ chính xác cao.

Những mạch tạo nguồn điện áp chuẩn ngày nay đã được chế tạo dưới dạng vi mạch. Trong mạch dùng diode Zener ổn nhiệt được kích thích bằng nguồn dòng không đổi và tăng cường công suất bằng khuếch đại điện áp dùng thuật toán. Độ chính xác của điện áp ra rất cao đạt tới 0.01%.

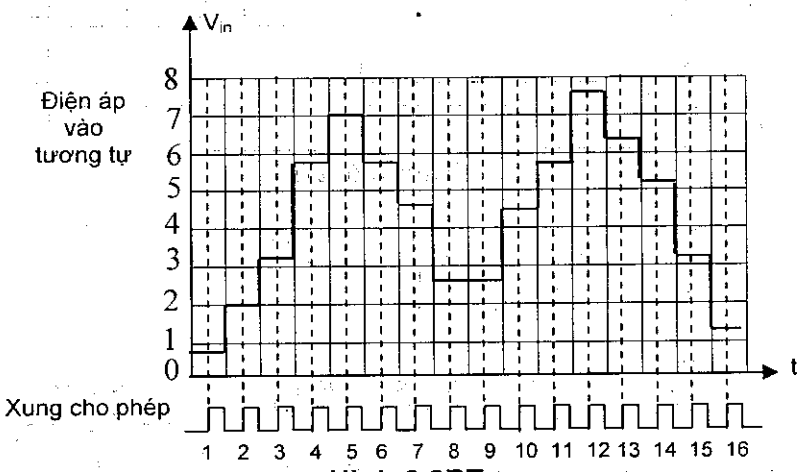
BÀI TẬP CHƯƠNG 6

Bài 6.1. Cho tín hiệu tương tự như trên hình 6.1BT. Hãy vẽ tín hiệu tương tự sau khi qua mạch lấy mẫu và giữ mẫu, biết rằng chu kỳ lấy mẫu là 1ms.



Hình 6.1BT

Bài 6.2. Hãy xác định mã nhị phân ở đầu ra của ADC kiểu flash 3 bit cho trên hình 6.1.2.1 với 16 xung tác động, biết tín hiệu vào và xung cho phép như trên hình 6.2BT, $V_{ref} = 8V$.



Hình 6.2BT

Bài 6.3. Nếu chu kỳ xung cho phép (EN) trong bài tập 6.2 tăng lên 2 lần, hãy xác định mã nhị phân nhận được ở đầu ra ADC với 8 xung tác động. Thông tin có bị mất không

Bài 6.4. Tìm giá trị lớn nhất của mức điện thế đầu ra từ 1 DAC 8 bit, biết rằng ứng với đầu vào số 00110111 mức điện thế sinh ra là 1V.

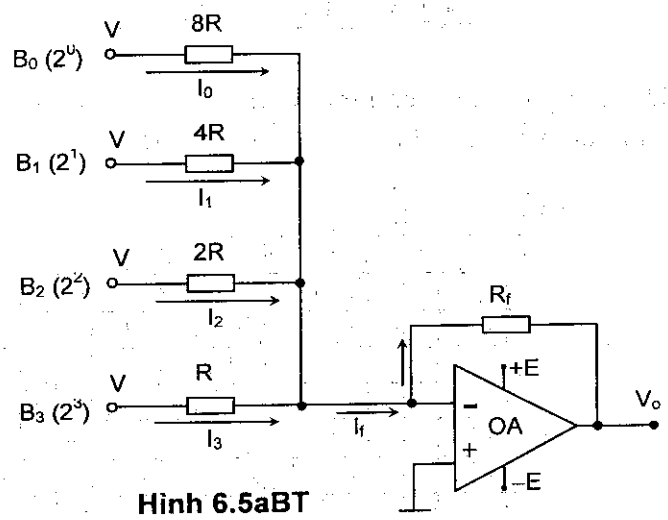
Bài 6.5. Xác định điện áp ra V_o của DAC trên hình 6.5aBT với $R = 25k\Omega$, $R_f = 10k\Omega$ nếu dạng sóng của bốn bit vào cho trên hình 6.5bBT.

Bài 6.6. Cho DAC dùng mạng điện trở R – 2R như trên hình 6.6BT.

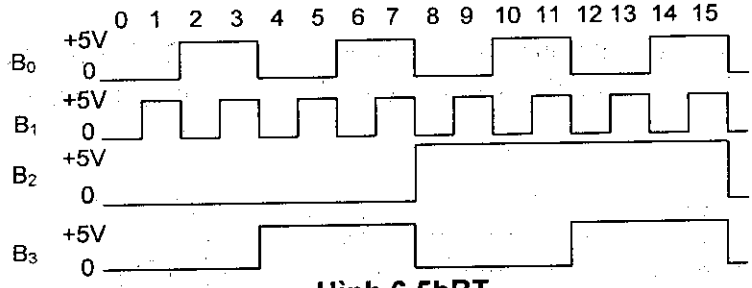
Hãy xác định điện áp ra V_o ứng với các số nhị phân ở đầu vào theo thứ tự $B_3B_2B_1B_0$ như sau: 1010, 1011.

Bài 6.7. Cho DAC dùng mạng điện trở R – 2R như trên hình 6.6BT.

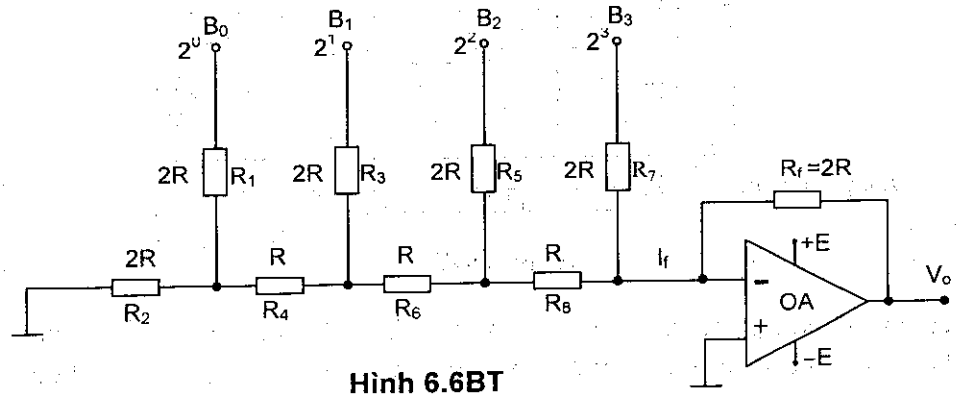
Hãy xác định điện áp ra V_o ứng với các đầu vào nhị phân cho trên hình 6.7BT.



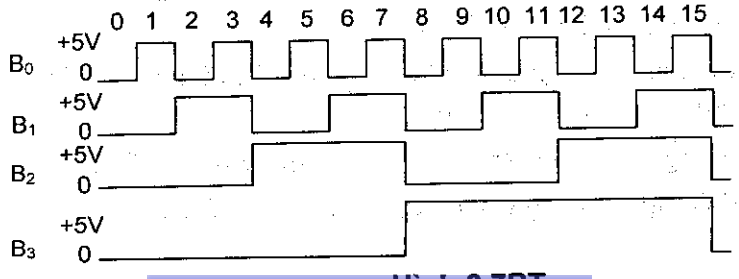
Hình 6.5aBT



Hình 6.5bBT



Hình 6.6BT



Hình 6.7BT

TÀI LIỆU THAM KHẢO

[1] Nguyễn Quốc Trung "Vi điện tử số", Nhà xuất bản Khoa học và Kỹ thuật, năm 1997.

[2] Nguyễn Thuý Vân "Kỹ thuật số", Nhà xuất bản Khoa học và Kỹ thuật, năm 1999.

[3] Nguyễn Thuý Vân "Thiết kế logic mạch số", Nhà xuất bản Khoa học và Kỹ thuật, năm 2001.

[4] Đặng Văn Chuyết "Kỹ thuật điện tử số", Nhà xuất bản Giáo dục, năm 1998.

[5] VN – GUIDE "Giáo trình chuyên ngành kỹ thuật số" 4 tập, Nhà xuất bản Thống kê, năm 2001.

[6] VN-GUIDE "Mạch số", Nhà xuất bản Thống kê, năm 2001.

[7] Bộ môn Điện tử – Đại học Thanh Hoa Bắc Kinh "Cơ sở kỹ thuật điện tử số", Nhà xuất bản Giáo dục, năm 1999.

[8] Huỳnh Đắc Thắng "Kỹ thuật số thực hành", Nhà xuất bản Khoa học và Kỹ thuật, năm 1997.

[9] Tom Floyd "Digital fundamentals", International Edition, 2006.

MỤC LỤC

Chịu trách nhiệm xuất bản:

Chủ tịch Hội đồng thành viên kiêm Tổng Giám đốc NGÔ TRẦN ÁI

Tổng biên tập kiêm Phó Tổng Giám đốc NGUYỄN QUÝ THAO

Tổ chức bản thảo và chịu trách nhiệm nội dung:

Phó tổng Biên tập NGÔ ÁNH TUYẾT

Giám đốc Công ty CP Sách ĐH – DN NGÔ THỊ THANH BÌNH

Biên tập nội dung và sửa bản in:

NGUYỄN DUY MẠNH

Trình bày bìa:

ĐINH XUÂN DŨNG

Chế bản:

MẠNH HÀ

KỸ THUẬT SỐ

(Dùng cho các trường đào tạo hệ Đại học và Cao đẳng)

Mã số: 7B810y2 – DAI

Số đăng kí KHXB : 16 - 2012/CXB/108 - 2050/GD.

In 700 cuốn (QĐ in số : 14), khổ 19 x 27 cm.

In tại Nhà in Đại học Quốc Gia Hà Nội.

In xong và nộp lưu chiểu tháng 4 năm 2012.

HEVOBCO

CÔNG TY CỔ PHẦN SÁCH ĐẠI HỌC - DẠY NGHỀ
(HIGHER EDUCATIONAL AND VOCATIONAL BOOKS JSC)
Địa chỉ: 25 Hàn Thuyên - Hà Nội - Tel: 043.9724715
Chi nhánh tại TP. Hồ Chí Minh : 240 Trần Bình Trọng - Quận 5
Email: info@hevobco.com.vn - web: http://www.hevobco.com.vn

TÌM ĐỌC SÁCH THAM KHẢO CHO CÁC TRƯỜNG CAO ĐẲNG NGHỀ VÀ TRUNG CẤP NGHỀ CỦA NHÀ XUẤT BẢN GIÁO DỤC VIỆT NAM

1. Giáo trình Cung cấp điện	*ThS. Ninh Văn Nam*
2. Giáo trình Khí cụ điện	*ThS. Vũ Đức Thoan*
3. Giáo trình Vật liệu điện	*ThS. Ninh Văn Nam*
4. Giáo trình Máy điện	*ThS. Vũ Đức Thoan*
5. Giáo trình Mạch điện	*ThS. Nguyễn Bá Khá*
6. Giáo trình Kĩ thuật cảm biến	*Vũ Quang Hồi*
7. Giáo trình Điện kĩ thuật	*ThS. Nguyễn Bá Khá*
8. Giáo trình Đo lường điện	*ThS. Nguyễn Thu Hà*

Bạn đọc có thể mua tại các Công ty Sách - Thiết bị trường học ở các địa phương hoặc các Cửa hàng của Nhà xuất bản Giáo dục Việt Nam :

Tại Hà Nội : 25 Hàn Thuyên ; 187B Giảng Võ ; 232 Tây Sơn ;
Tại Đà Nẵng : Số 15 Nguyễn Chí Thanh ; Số 62 Nguyễn Chí Thanh ;
Tại Thành phố Hồ Chí Minh : Cửa hàng 451B - 453, Hai Bà Trưng – Quận 3 ;
240 Trần Bình Trọng – Quận 5.
Tại Thành phố Cần Thơ : Số 5/5, đường 30/4 ;
Website : www.nxbgd.vn

8 934994 116107

Giá: 69.000 đ