Trường Đại Học Đà Lạt
Khoa Vật Lý

Bài Giảng Tóm Tắt
Kỹ Thuật Điện Tử Số

Lê Văn Tùng

Lưu Hành Nội Bộ

2019

Mục lục

1 Hệ Đếm Và Biểu Diễn Số **1**

1.1 Giới thiệu 1

1.2 Các hệ số đếm 2

1.3 Chuyển đổi giữa các hệ đếm 4

1.4 Mã BCD 8

1.5 Mã Gray 8

1.6 Mã ASCII 10

1.7 Phương pháp kiểm tra lỗi 10

2 Tính Toán Số Học **12**

2.1 Giới thiệu 12

2.2 Phép tính cộng và trừ 12

2.3 Phép tính nhân số nhị phân 14

2.4 Phép tính chia số nhị phân 14

2.5 Trong các hệ đếm khác 15

3 Cổng Cửa Luận Lý **17**

3.1 Giới thiệu 17

3.2 Biến số và hằng số Boolean 17

3.3 Bảng sự thật 17

3.4 Phép toán OR 18

3.5 Phép toán AND 19

3.6 Phép toán NOT 19

3.7 Cổng kết hợp 20

3.8 Các cổng luận lý khác 21

3.9 Các đinh lý Boolean 23

4 Các Phương Pháp Đơn Giản Hóa Mạch **26**

4.1 Giới thiệu . . . 26
4.2 Tổng của tích và tích của tổng . . . 26
4.3 Đơn giản hóa mạch . . . 27
4.4 Giản đồ Karnaugh . . . 28
4.5 Phương pháp Quine-McCluskey . . . 31

5 Mạch Tính Toán **34**

5.1 Giới thiệu . . . 34
5.2 Mạch kết hợp . . . 34
5.3 Thực hiện mạch kết hợp . . . 34
5.4 Mạch tích hợp . . . 36

6 Các Họ Vi Mạch Số **41**

6.1 Giới thiệu . . . 41
6.2 Các họ vi mạch . . . 41
6.3 Giao tiếp giữa các họ linh kiện . . . 51
6.4 Một số loại IC thông dụng . . . 51

7 Linh Kiện Flip-Flop **55**

7.1 Giới thiệu . . . 55
7.2 Bộ chốt cổng NAND và NOR . . . 55
7.3 Xung số . . . 57
7.4 Một số loại Flip-flop . . . 59
7.5 Tham số thời gian của Flip-flop . . . 62
7.6 Mạch tạo xung . . . 63

8 Mạch Tuần Tự **67**

8.1 Giới thiệu . . . 67
8.2 Bộ đếm bất đồng bộ và đồng bộ . . . 67
8.3 Bộ đếm MOD . . . 69
8.4 Bộ đếm đồng bộ . . . 72
8.5 Thanh ghi . . . 75

9 Biến Đổi Tương Tự - Số **78**

9.1 Giới thiệu . . . 78

9.2 Biến đổi số sang tương tự 78
9.3 Biến đổi tương tự sang số 81
9.4 Thu thập dữ liệu 84

10 Thiết Bị Nhớ **88**
10.1 Giới thiệu 88
10.2 Một số thuật ngữ 88
10.3 ROM 89
10.4 RAM 92

1

Hệ Đếm Và Biểu Diễn Số

1.1 Giới thiệu

Tìm hiểu về các hệ số đếm giúp ta tiếp cận và nắm bắt cách dữ liệu được trình bày, lưu trữ và xử lý bởi các hệ thống điện tử. Đây là một trong những kiến thức cơ bản của điện tử số.

Có hai cách để trình bày một giá trị bất kỳ, cách thức đầu tiên được gọi là *thể hiện tương tự*, với giá trị nằm trong khoảng liên tục giữa hai cực trị. Ví dụ, nhiệt độ trong phòng có thể đo được là 27°C hoặc 26,94°C hoặc 27,1345°C tùy thuộc vào độ chính xác của thiết bị đo. Tương tự, khi đo điện áp rơi trên hai đầu của một linh kiện trong mạch điện tử, giá trị đo được có thể thay đổi bất kỳ quanh một điểm.

Cách thứ hai là *thể hiện số*, giá trị quan tâm được thể hiện bằng các bước rời rạc. Theo ví dụ trên, nếu ta chọn bước đo là 1°C thì các chỉ số sẽ là 27°C, 28°C hoặc 26°C. Như vậy, *thể hiện tương tự* của kết quả đo mang một giá trị liên tục trong khi *thể hiện số* cho ra giá trị rời rạc.

Các hệ thống sử dụng *thể hiện số* hay còn gọi là các hệ thống số có những ưu điểm sau:

- Hệ thống số dễ thiết kế hơn hệ thống tương tự.
- Lưu trữ thông tin trong hệ thống số dễ dàng hơn.
- Đảm bảo tính chính xác trong toàn hệ thống.
- Hệ thống số ít chịu tác động bởi nhiễu.
- Các vi mạch số dễ chế tạo hơn vi mạch tương tự.

Tuy nhiên, hệ thống số cũng có giới hạn của nó. Thế giới thực là *tương tự* và việc xử lý chuyển đổi sang hệ thống số cần thời gian.

1.2 Các hệ số đếm

Rất nhiều hệ số đếm được sử dụng trong công nghệ điện tử số. Thông dụng nhất là hệ thập phân, nhị phân, bát phân và thập lục phân. Trong đó, hệ thập phân được chúng ta sử dụng hằng ngày.

Hệ thập phân

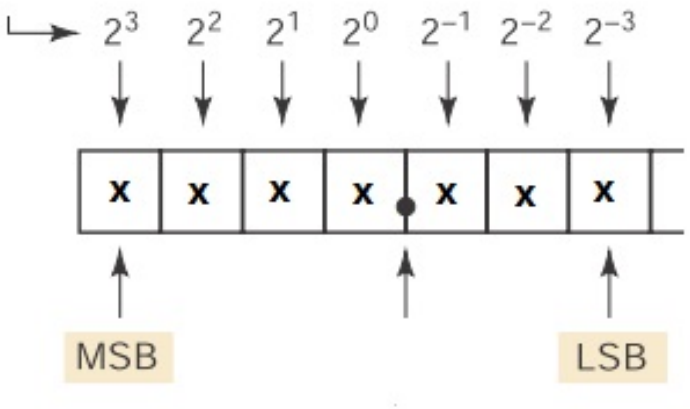
Sử dụng 10 chữ số là 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 để thể hiện các giá trị. Hệ thập phân tính giá trị theo vị trí của chữ số. Ví dụ số 365 được hiểu là 3 trăm 6 chục 5 đơn vị, với 3 mang trọng số lớn nhất và 5 có trọng số nhỏ nhất.

Đối với số 3,14 ta có $3 \times 1 + 1 \times 0,1 + 4 \times 0,01$. Cần lưu ý rằng dấu phẩy (,) được sử dụng trong hệ đếm kiểu Việt nam thay cho dấu chấm thập phân.

Hệ nhị phân

Tuy rằng chúng ta quen dùng hệ thập phân nhưng rất khó thiết kế các hệ thống điện tử sử dụng thang thập phân. Do vậy, hệ nhị phân (binary) được sử dụng vì tính đơn giản, chỉ có hai mức 0 và 1.

Cũng giống như hệ thập phân, hệ nhị phân cũng áp dụng tính giá trị theo vị trí. Cách tính giá trị một số nhị phân được thể hiện trong Hình 1.1. Chữ số có trọng số lớn nhất được gọi là MSB (Most Significant Bit) và chữ số trọng số nhỏ nhất gọi là LSB (Least Significant Bit).



Hình 1.1 Tính giá trị dựa theo vị trí nhị phân

Cách đếm theo hệ phị phân được trình bày trong Hình 1.2

Hệ bát phân và thập lục phân

Hệ bát phân (Octal) gồm tám ký hiệu: 0, 1, 2, 3, 4, 5, 6 và 7 nên cơ số của hệ là 8. Hệ cơ số 8 có thể được biểu diễn thành 2^3 . Do đó, mỗi ký hiệu trong hệ này có thể thay thế bằng 3 bit trong hệ nhị phân.

Hệ thập lục phân hay hệ Hexa (Hexadecimal). Hệ gồm 16 ký hiệu là 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F nên còn gọi là hệ cơ số 16. Người ta thường dùng chữ H (hoặc h) sau con số để chỉ số thập lục phân.

Decimal	Binary	Octal	Hexidecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Hình 1.2 Đếm theo các hệ khác nhau

Số bù và dấu

Bit là một từ viết tắt của chữ số nhị phân và là đơn vị nhỏ nhất mang thông tin. Nó có giá trị “0" hoặc “1". Một *byte* có tám bit và là đơn vị dữ liệu cơ bản của các hệ thống máy tính. Một *word* là một chuỗi các bit và có độ lớn phụ thuộc vào từng hệ máy tính. Trong khi đó một *nibble* là nhóm của bốn bit, hay một nửa *byte*.

Bù 1 (1's complement) là một số nhị phân thu được bằng cách hoán đổi giá trị tất cả các bit. Ví dụ, bù 1 của (10010011) là (01101100).

Bù 2 của một giá trị nhị phân được tính bằng cách cộng thêm '1' vào số bù 1 của nó. Ví dụ, bù 2 của (10010011) là (01101101). Điểm đặc biệt là thực hiện bù 2 của một số bù 2 chính là phép đảo biến nó trở lại số ban đầu.

Để thể hiện số âm và dương, bit có giá trị lớn nhất (MSB) thể hiện dấu với '0' là dấu cộng và '1' là dấu trừ. Các bit còn lại thể hiện độ lớn. Trong hệ 8 bit, ví dụ số +9 sẽ là (00001001) và số -9 sẽ là (10001001). Như vậy với hệ 8 bit, vùng giá trị thập phân khi sử dụng dấu là từ -127 đến +127.

Trong cách viết bù 1, cách viết số dương không thay đổi. Số âm được thể hiện bằng cách lấy bù 1 của số dương. Ví dụ, +9 là (00001001) và -9 là (11110110). Tương tự như trên, với cách viết bù 1 của hệ 8 bit có thể thể hiện một số thập phân trong khoảng giá trị từ -127 đến +127.

Đối với cách viết bù 2, MSB mang dấu cộng khi có giá trị '0' và trừ khi là '1'. Các bit còn lại sử dụng để chỉ giá trị. Cách viết số dương giống như trường hợp của số nhị phân mang dấu hoặc bù 1. Số âm được thể hiện bằng cách viết bù 2 của số đó khi mang dấu dương. Ví dụ, +9 là (00001001) và -9 là (11110111).

1.3 Chuyển đổi giữa các hệ đếm

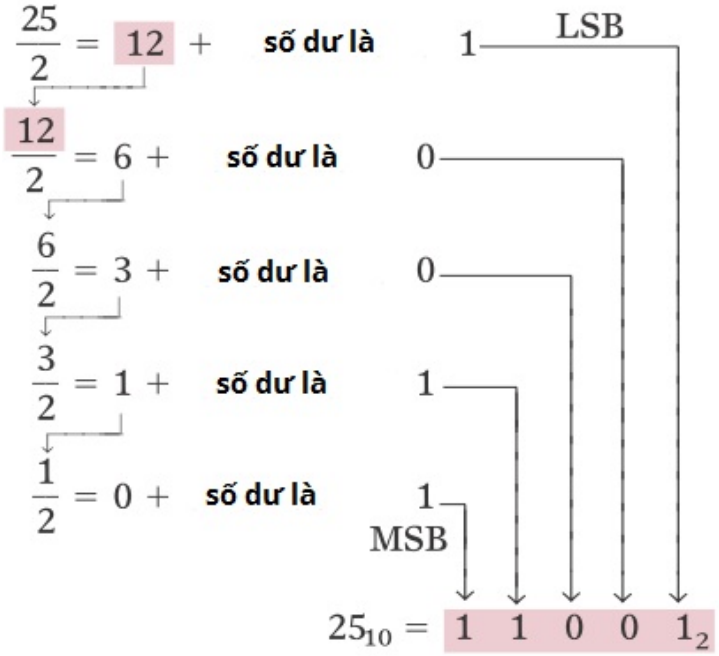
Hệ nhị phân và thập phân

Bất kỳ số nhị phân nào cũng có thể chuyển sang số thập phân bằng cách tính tổng của các bit nhị phân với trọng số của nó. Ví dụ, số nhị phân (1001,0101) sẽ được tính:

- Phần số nguyên là $(1001) = 1 \times 2^0 + 0 \times 2^1 + 0 \times 2^2 + 1 \times 2^3 = 9$
- Phần sau dấu thập phân $(,0101) = 0 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} = 0,3125$
- Kết quả (1001,0101) = 9,3125

Phương pháp thông dụng nhất để chuyển đổi số thập phân sang nhị phân là lặp lại phép chia cơ số. Lấy số thập phân cần chuyển đổi chia cho 2, ghi lại số dư và kết quả, nếu phép chia là chia hết thì ghi '0' và ghi '1' nếu có dư. Chú ý rằng số dư đầu tiên là LSB và số dư cuối cùng là MSB khi viết lại số nhị phân. Xem minh họa theo Hình 1.3.

Đối với số thập phân có phần lẻ thì cách tính sẽ khác đối với phần lẻ. Ví dụ như số thập phân 13,375



Hình 1.3 Chuyển số thập phân sang nhị phân

- Phần nguyên 13 = (1101)
- Phần lẻ 0,375 sẽ được tính từng bước:
  - $0,375 \times 2 = 0,75$ có phần nguyên là '0'
  - $0,75 \times 2 = 1,5$ có phần nguyên là '1'
  - $0,5 \times 2 = 1$ có phần nguyên là '1'
- Như vậy phần lẻ 0,375 = (0,011)
- Kết quả số 13,375 = (1101,011)

Hệ bát phân và thập phân

Giá trị thập phân của số (137,21) trong hệ bát phân được tính như sau:

- Phần nguyên $(137) = 7 \times 8^0 + 3 \times 8^1 + 1 \times 8^2 = 95$
- Phần lẻ $(0,21) = 2 \times 8^{-1} + 1 \times 8^{-2} = 0,265$
- Kết quả (137,21) = 95,265

Để chuyển đổi số thập phân sang bát phân, quy trình cũng giống như hệ nhị phân nhưng với cơ số bằng 8. Ví dụ số thập phân 73,75 sẽ được tính:

- Phần nguyên 73 = (111)
- Phần lẻ $0,75 \times 8 = 6$
- Kết quả 73,75 = (111,6)

Hệ thập lục phân và thập phân

Giá trị thập phân của số (1E0,2A) trong hệ thập lục phân là:

- Phần nguyên $(1E0) = 0 \times 16^0 + 14 \times 16^1 + 1 \times 16^2 = 480$
- Phần lẻ $(2A) = 2 \times 16^{-1} + 10 \times 16^{-2} = 0,164$
- Kết quả (1E0,2A) = 480,164

Tương tự như trên, chuyển số thập phân sang thập lục phân sẽ tính với cơ số 16. Ví dụ số thập phân 82,25 sẽ được tính:

- Phần nguyên 82 = (52)
- Phần lẻ $0,25 \times 16 = 4$
- Kết quả số thập phân 82,25 = (52,4)

Hệ nhị phân và các hệ đếm

Một số bát phân bất kỳ có thể được chuyển sang nhị phân bằng cách thay từng chữ số bát phân bằng 3 bit nhị phân tương ứng. Ngược lại, một số nhị phân có thể được chuyển sang bát phân bằng cách chia các phần nguyên và lẻ thành từng nhóm 3 bit. Có thể thêm số '0' vào các nhóm ngoài cùng nếu cần để đủ nhóm 3 bit.

Ví dụ chuyển số bát phân (374,26) sang nhị phân như sau:

- Chia nhóm (374,26) = (011 111 100,010 110)
- Bỏ các số '0' ngoài cùng sẽ thành (374,26) = (11111100,01011)

Đối với số nhị phân (1110100,0100111) sẽ được chuyển sang bát phân:

- Tạo nhóm (1110100,0100111) = (1 110 100,010 011 1) = (001 110 100,010 011 100)
- Viết lại theo bát phân (1110100,0100111) = (164,234)

Đối với hệ thập lục phân, quy trình cũng tương tự nhưng nhóm các bit sẽ có 4 chữ số.

Ví dụ chuyển đổi số thập lục phân (17E,F6) sang nhị phân:

- Tạo nhóm (17E,F6) = (0001 0111 1110,1111 0110)
- Bỏ các số '0' ngoài cùng (17E,F6) = (101111110,1111011)

Và với số nhị phân (1011001110,011011101) sang thập lục phân như sau:

- Chia nhóm (1011001110,011011101) = (10 1100 1110,0110 1110 1)
- Thêm số '0' cho đủ nhóm 4 bit (10 1100 1110,0110 1110 1) = (0010 1100 1110,0110 1110 1000)
- Kết quả (1011001110,011011101) = (2CE,6E8)

Trong trường hợp cần chuyển giữa các hệ bát phân và thập lục phân, cách đơn giản nhất là sử dụng hệ nhị phân làm trung gian. Một cách khác là dùng hệ thập phân làm trung gian nhưng không tiện lợi bằng hệ nhị phân.

Dấu chấm động

Được sử dụng để thể hiện các số một cách linh hoạt, từ đó giúp việc thực hiện tính toán dễ dàng hơn. Số với dấu chấm động thường được viết dưới dạng:

$$N = m \times b^e \tag{1.1}$$

với m là định trị, phần nguyên e được gọi là số mũ và b là cơ số của hệ đếm. Đối với hệ nhị phân $N = m \times 2^e$, còn hệ thập phân thì $N = m \times 16^e$

Ví dụ, số thập phân 3,1415 có thể được viết thành 31415×10^{-4} hoặc $0,31415 \times 10^1$. Số thập lục phân 234,AE viết lại là $2,34AE \times 16^2$

Đối với số nhị phân $(110,1011) = (0,1101011) \times 2^3 = (0,1101011e+0011)$. Với $e+0011$ thể hiện số mũ là +3. Trong ví dụ với số $(-0,00000101) = (-0,101 \times 2^{-5}) = -0,101e-0101$ với $e-0101$ thể hiện phần mũ là -5.

Số với dấu chấm động thường được thể hiện theo chuẩn IEEE-754 và IEEE-754 cải tiến. Nó được dùng trong các đơn vị xử lý số học của máy tính và vi xử lý.

1.4 Mã BCD

Khi các số, chữ số được trình bày theo từng nhóm thì chúng được xem như đã mã hóa. Các nhóm ký tự đó được gọi là mã. Phổ biến nhất là mã Morse, với một chuỗi các ký tự chấm và gạch thể hiện bảng chữ cái.

Như vậy, quá trình mã hóa nhị phân trực tiếp là khi một số thập phân được viết lại thành một số nhị phân. Nếu mỗi chữ số trong một số thập phân được thể hiện bởi các bit nhị phân tương ứng, kết quả được gọi là mã BCD (binary-coded-decimal). Vì chữ số thập phân tối đa là 9, mã hóa BCD cần bốn bit thể hiện từng chữ số.

Ví dụ số thập phân 478, từng chữ số sẽ được mã hóa sang nhị phân tương ứng: $4 \rightarrow (0100), 7 \rightarrow (0111), 8 \rightarrow (1000)$. Như vậy, với 4 bit nhưng chỉ thể hiện 10 giá trị thập phân tương ứng, các giá trị 4 bit còn lại được gọi là khoảng cấm. Trong các máy sử dụng mã BCD, nếu bất kỳ số nhị phân 4 bit nào trong vùng cấm xuất hiện thì đó là dấu hiệu của lỗi.

Mã BCD không phải là một hệ thống số đếm mới. Nó chỉ là sự thể hiện nhị phân của hệ 10 với từng chữ số thập phân được mã hóa theo nhị phân. Ví dụ như số 137 thập phân:

- Nhị phân hoàn chỉnh 137 = (10001001)
- Mã BCD 137 = 0001 0011 0111

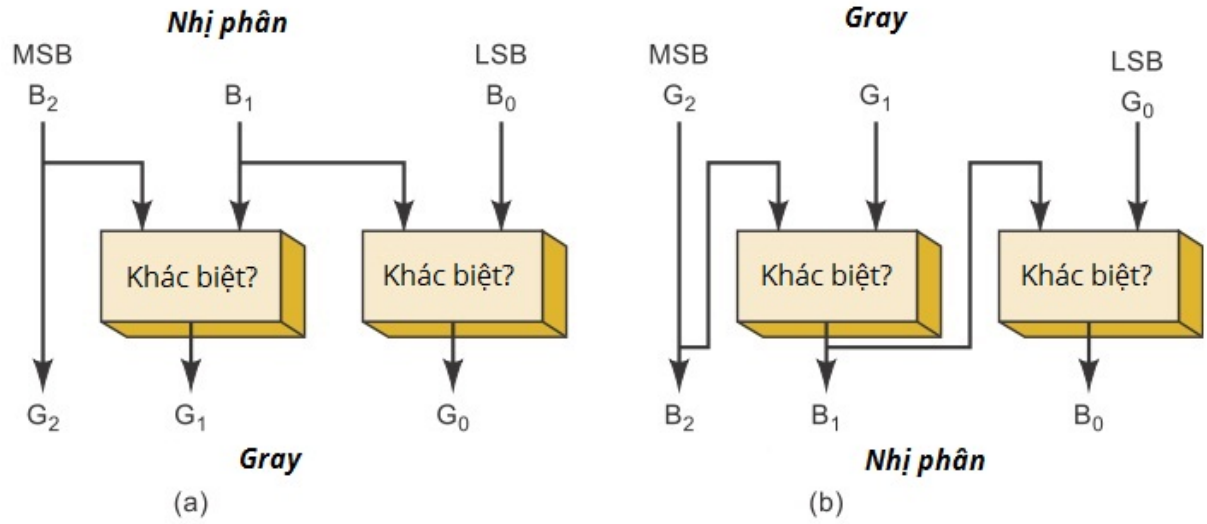
Để chuyển đổi một số nhị phân sang BCD hay ngược lại, người ta thường dùng hệ đếm thập phân làm trung gian. Tất cả các số cần chuyển đổi trước hết sẽ được chuyển sang thập phân.

1.5 Mã Gray

Các hệ thống điện tử vận hành ở tốc độ rất cao và thường xuyên thay đổi trạng thái phụ thuộc các yếu tố đầu vào. Tuy nhiên, khi có quá nhiều thay đổi tại đầu vào, hệ thống đối mặt với nguy cơ nhận định sai giá trị. Trong quá trình đếm hệ nhị phân, một số bit thay đổi đồng thời. Để tránh khả năng hệ thống điện tử nhận định sai giá trị, mã Gray được đưa vào sử dụng.

Yếu tố đặc biệt của mã Gray là chỉ có 01 bit thay đổi trạng thái khi giá trị đếm thay đổi tuần tự. Để chuyển từ mã nhị phân sang Gray, bắt đầu từ MSB theo Hình 1.4. So sánh MSB nhị phân với bit kế cận (B1). Nếu chúng giống nhau thì G1 = 0; nếu khác nhau thì G1 = 1. So sánh B1 với B0 để tìm G0.

Tương tự, chuyển từ mã Gray sang nhị phân cũng giữ nguyên MSB. Bit nhị phân tiếp theo được tìm bằng cách so sánh bit nhị phân bên trái với bit tương ứng trong mã Gray. Nếu giống nhau, bit sẽ là 0 và khác nhau thì bit là 1. Ứng dụng gần gũi nhất của mã Gray là mã hóa chuyển



Hình 1.4 Chuyển đổi nhị phân và mã Gray

động trục động cơ. Ngoài ra, trong truyền thông, mã Gray được dùng vì có khả năng giảm thiểu tối đa lỗi.

Decimal	Binary	Hexadecimal	BCD	GRAY
0	0	0	0000	0000
1	1	1	0001	0001
2	10	2	0010	0011
3	11	3	0011	0010
4	100	4	0100	0110
5	101	5	0101	0111
6	110	6	0110	0101
7	111	7	0111	0100
8	1000	8	1000	1100
9	1001	9	1001	1101
10	1010	A	0001 0000	1111
11	1011	B	0001 0001	1110
12	1100	C	0001 0010	1010
13	1101	D	0001 0011	1011
14	1110	E	0001 0100	1001
15	1111	F	0001 0101	1000

Hình 1.5 Tổng hợp so sánh giữa các hệ đếm và mã

1.6 Mã ASCII

Ngoài dữ liệu số, máy tính còn xử lý các thông tin không phải số. Chúng có thể là ký tự, chữ viết hoặc dấu đặc biệt. Một trong những mã này là ASCII, sử dụng 7 bit. Mã ASCII được dùng để truyền thông tin giữa các máy tính hoặc thiết bị ngoại vi. Ngoài ra, thông tin từ bàn phím cũng được lưu trữ dưới dạng mã ASCII.

Trong các hệ thống lớn, người ta còn sử dụng mã EBCDIC. Nó sử dụng 8 bit để mã hóa thông tin. Với nhiều hệ thống khác nhau, ngoài mã ASCII và EBCDIC thì vẫn còn một số loại mã khác ít thông dụng hơn.

Dec	Hex	Oct	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0	0	000	NULL	32	20	040	&#032;	Space	64	40	100	&#064;	@	96	60	140	&#096;	`
1	1	001	Start of Header	33	21	041	&#033;	!	65	41	101	&#065;	A	97	61	141	&#097;	a
2	2	002	Start of Text	34	22	042	&#034;	"	66	42	102	&#066;	B	98	62	142	&#098;	b
3	3	003	End of Text	35	23	043	&#035;	#	67	43	103	&#067;	C	99	63	143	&#099;	c
4	4	004	End of Transmission	36	24	044	&#036;	$	68	44	104	&#068;	D	100	64	144	&#100;	d
5	5	005	Enquiry	37	25	045	&#037;	%	69	45	105	&#069;	E	101	65	145	&#101;	e
6	6	006	Acknowledgment	38	26	046	&#038;	&	70	46	106	&#070;	F	102	66	146	&#102;	f
7	7	007	Bell	39	27	047	&#039;	'	71	47	107	&#071;	G	103	67	147	&#103;	g
8	8	010	Backspace	40	28	050	&#040;	(	72	48	110	&#072;	H	104	68	150	&#104;	h
9	9	011	Horizontal Tab	41	29	051	&#041;	)	73	49	111	&#073;	I	105	69	151	&#105;	i
10	A	012	Line feed	42	2A	052	&#042;	*	74	4A	112	&#074;	J	106	6A	152	&#106;	j
11	B	013	Vertical Tab	43	2B	053	&#043;	+	75	4B	113	&#075;	K	107	6B	153	&#107;	k
12	C	014	Form feed	44	2C	054	&#044;	,	76	4C	114	&#076;	L	108	6C	154	&#108;	l
13	D	015	Carriage return	45	2D	055	&#045;	-	77	4D	115	&#077;	M	109	6D	155	&#109;	m
14	E	016	Shift Out	46	2E	056	&#046;	.	78	4E	116	&#078;	N	110	6E	156	&#110;	n
15	F	017	Shift In	47	2F	057	&#047;	/	79	4F	117	&#079;	O	111	6F	157	&#111;	o
16	10	020	Data Link Escape	48	30	060	&#048;	0	80	50	120	&#080;	P	112	70	160	&#112;	p
17	11	021	Device Control 1	49	31	061	&#049;	1	81	51	121	&#081;	Q	113	71	161	&#113;	q
18	12	022	Device Control 2	50	32	062	&#050;	2	82	52	122	&#082;	R	114	72	162	&#114;	r
19	13	023	Device Control 3	51	33	063	&#051;	3	83	53	123	&#083;	S	115	73	163	&#115;	s
20	14	024	Device Control 4	52	34	064	&#052;	4	84	54	124	&#084;	T	116	74	164	&#116;	t
21	15	025	Negative Ack.	53	35	065	&#053;	5	85	55	125	&#085;	U	117	75	165	&#117;	u
22	16	026	Synchronous idle	54	36	066	&#054;	6	86	56	126	&#086;	V	118	76	166	&#118;	v
23	17	027	End of Trans. Block	55	37	067	&#055;	7	87	57	127	&#087;	W	119	77	167	&#119;	w
24	18	030	Cancel	56	38	070	&#056;	8	88	58	130	&#088;	X	120	78	170	&#120;	x
25	19	031	End of Medium	57	39	071	&#057;	9	89	59	131	&#089;	Y	121	79	171	&#121;	y
26	1A	032	Substitute	58	3A	072	&#058;	:	90	5A	132	&#090;	Z	122	7A	172	&#122;	z
27	1B	033	Escape	59	3B	073	&#059;	;	91	5B	133	&#091;	[	123	7B	173	&#123;	{
28	1C	034	File Separator	60	3C	074	&#060;	<	92	5C	134	&#092;	\	124	7C	174	&#124;	|
29	1D	035	Group Separator	61	3D	075	&#061;	=	93	5D	135	&#093;	]	125	7D	175	&#125;	}
30	1E	036	Record Separator	62	3E	076	&#062;	>	94	5E	136	&#094;	^	126	7E	176	&#126;	~
31	1F	037	Unit Separator	63	3F	077	&#063;	?	95	5F	137	&#095;	_	127	7F	177	&#127;	Del

Hình 1.6 Bảng mã ASCII cơ bản.

1.7 Phương pháp kiểm tra lỗi

Một ưu điểm của các hệ thống số là khả năng kiểm tra và sửa lỗi. Trong quá trình làm việc, lỗi phát sinh và có thể gây ảnh hưởng nghiêm trọng đến tính năng của hệ thống. Để kiểm tra và sửa lỗi, người ta thêm vào các bit đặc biệt.

Mã chẵn lẻ

Là một bit được thêm vào chuỗi bit dữ liệu nhằm phát hiện lỗi có thể phát sinh trong quá trình xử lý, lưu trữ và truyền thông tin. Bit thêm vào (parity bit) có thể là '0' hoặc '1' tùy vào việc sử dụng mã chẵn hoặc mã lẻ.

Tuy nhiên, mã chẵn lẻ không thể phát hiện lỗi nếu số bit lỗi là chẵn. Ngoài ra, nó không thể xác định vị trí của bit bị lỗi.

Mã lặp

Đơn giản là việc lặp lại việc truyền một bit nhiều lần. Trong quá trình nhận dữ liệu, các bit được kiểm tra theo nhóm định sẵn. Nếu xuất hiện lỗi thì các bit trong nhóm đó sẽ khác nhau. Như vậy, bit thông tin có khả năng tự sửa sai khi đối chiếu số lượng bit giống nhau trong cùng nhóm lặp.

Tuy nhiên, truyền bit theo mã lặp không thể sửa trên 2 bit nếu dùng mã lặp 3. Để xác định và tự sửa lỗi khi dùng mã lặp cho nhiều bit, số lượng bit phải lặp lại trong quá trình truyền phải tăng lên. Điều này khiến dung lượng lưu trữ tăng theo và giảm hiệu suất truyền thông.

Mã CRC

Cho phép bảo vệ dữ liệu trước các lỗi khi chỉ sử dụng tối thiểu tài nguyên. Thực hiện mã CRC đòi hỏi quá trình tính toán dữ liệu trước khi chèn các bit CRC. Xác suất phát hiện lỗi phụ thuộc số bit kiểm tra. Đối với các lỗi 1 bit hoặc 2 bit, mã CRC đảm bảo phát hiện 100%.

Mã Hamming

Nếu số lượng các bit kiểm tra phù hợp, không chỉ lỗi mà vị trí bit bị lỗi cũng có thể được phát hiện. Mã Hamming có khả năng phát hiện và sửa lỗi 1 bit trong chuỗi thông tin với độ dài bất kỳ. Mặc dù mã Hamming có thể xác định lỗi 2 bit, nó không thể phát hiện vị trí lỗi.

Mã Hamming phổ biến nhất là chuỗi 7 bit với 4 bit dữ liệu và 3 bit chẵn lẻ, gọi là Hamming(7,4).

2

Tính Toán Số Học

2.1 Giới thiệu

Trong chương trước, dữ liệu số và ký tự được thể hiện bằng nhiều phương thức khác nhau. Bước tiếp theo là nghiên cứu cách thức xử lý dữ liệu đã có. Đối với dữ liệu nhị phân, hai phép toán được sử dụng là tính toán số học và luận lý. Các phép toán số học cơ bản bao gồm phép cộng, phép trừ, phép nhân và phép chia. Trong khi đó, tính toán luận lý cơ bản có phép tính AND, OR và NOT.

2.2 Phép tính cộng và trừ

Phép tính cộng hai số nhị phân được thực hiện giống như cộng hai số thập phân. Đối với phép tính cộng hai số thập phân, chữ số có giá trị nhỏ nhất được thực hiện trước. Kết quả của phép cộng được ghi lại và số nhớ *carry* (nếu có) được cộng vào chữ số tiếp theo.

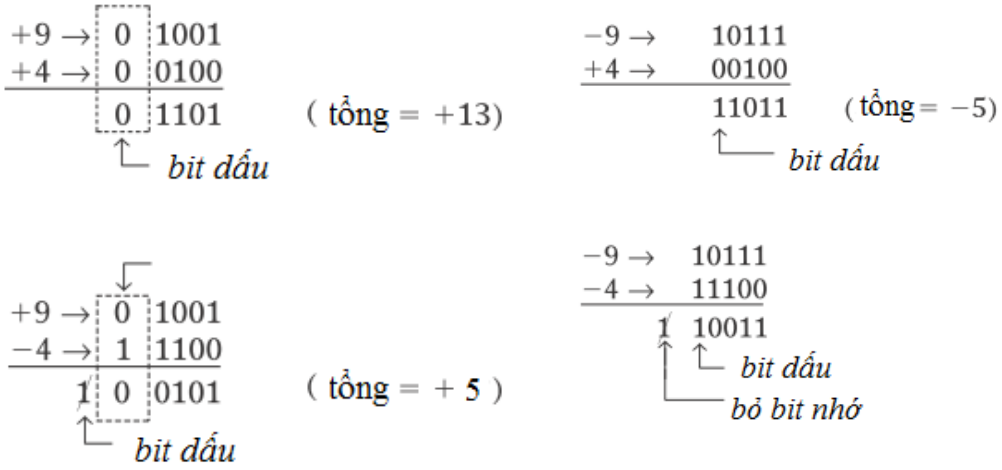
Áp dụng nguyên tắc như số thập phân, cộng nhị phân trên thực tế đơn giản hơn vì có ít trạng thái. Tương tự, phép tính trừ hai số nhị phân cũng giống như số thập phân. Thực hiện phép tính tuần tự từ chữ số có giá trị nhỏ nhất. Kết quả được ghi lại và số mượn (nếu có) được trừ vào chữ số tiếp theo.

Sử dụng hệ bù 2

Đối với số nhị phân có dấu, phép tính cộng trừ được thực hiện với hệ bù 2. Bit thể hiện dấu cũng được tính giống như các bit giá trị khác.

Cộng hai số dương: Thực hiện trực tiếp như bình thường. Trong đó, các số phải được viết có cùng độ dài bit. Minh họa trong Hình 2.1.

Cộng số dương lớn với số âm nhỏ hơn: Số âm sẽ được viết sang dạng bù 2. Chú ý rằng số nhớ thuộc về bit dấu sẽ được loại bỏ.



Hình 2.1 Cộng số nhị phân có dấu

Cộng số dương với số âm lớn hơn: Viết số âm dưới dạng bù 2 và tính như trên. Trong ví dụ ở Hình 2.1, kết quả mang cùng dấu âm của số lớn nên nó đang ở dạng bù 2. Để tìm kết quả cuối cùng, thực hiện phép tính đảo của số bù 2.

Cộng hai số âm: Viết cả hai số ở dạng bù 2. Kết quả cũng ở dạng bù 2 nên phải thực hiện tính đảo để có kết quả cuối cùng.

Cộng hai số đối nhau: Kết quả dĩ nhiên bằng 0.

Trừ hai số dùng hệ bù 2

Trên thực tế việc tính trừ trong hệ bù 2 là áp dụng phép tính cộng. Để thực hiện phép tính, ta áp dụng các bước sau:

- Đảo số trừ: biến nó sang số tương đương với dấu đảo.
- Cộng vào số bị trừ: kết quả của phép cộng thể hiện sự khác biệt giữa số trừ và số bị trừ.

Theo ví dụ trên, phép trừ 4 cho 9 cũng chính là phép cộng (+9) với (-4). Bằng việc sử dụng hệ bù 2, các phép tính cộng trừ được tính toán trên cùng một mạch giúp giảm chi phí phần cứng.

Tràn nhị phân

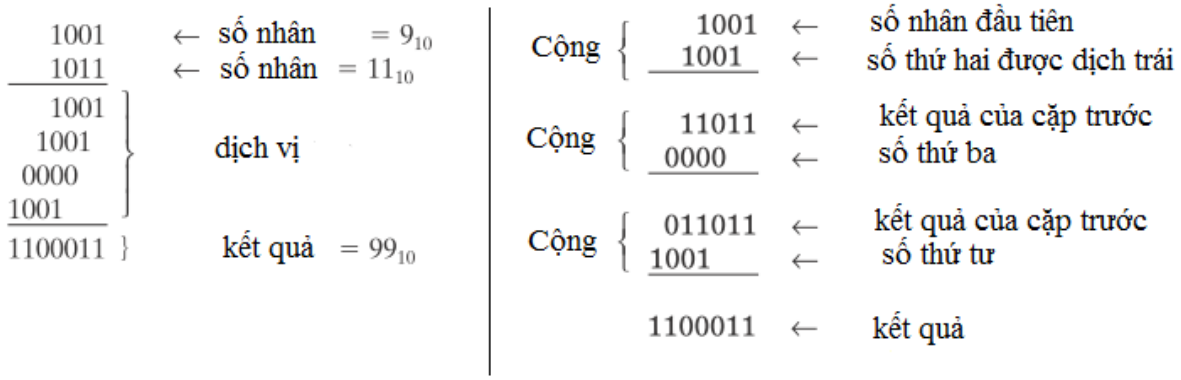
Trong các ví dụ trên, các số cộng trừ và kết quả đều có 1 bit dấu và 4 bit giá trị. Tuy nhiên, kết quả các phép tính là nhỏ nên vẫn nằm trong khoảng 4 bit. Nếu như kết quả của phép tính có giá trị lớn cần số bit lớn hơn thì sẽ xảy ra hiện tượng tràn.

Tràn chỉ xảy ra khi cộng hai số cùng dấu. Lỗi tràn có thể được kiểm tra bằng cách đối chứng bit dấu của của kết quả với bit dấu của các số hạng. Đối với phép tính trừ dùng hệ bù 2, tràn cũng có thể xảy ra.

Để đảm bảo các phép tính là chính xác, máy tính phải bổ sung mạch dò lỗi tràn.

2.3 Phép tính nhân số nhị phân

Cũng giống như phép tính nhân số thập phân nhưng có phần đơn giản hơn vì kết quả chỉ có hai giá trị. Quan sát quá trình nhân ta thấy số nhân được dịch sang trái một đơn vị. Các máy tính số thông thường chỉ có thể cộng hai số nhị phân một lần. Sau mỗi lần nhân một chữ số, các kết quả được cộng lại với nhau như minh họa trong Hình 2.2.



Hình 2.2 Nhân hai số nhị phân

Đối với phép nhân dùng hệ bù 2, phép tính cũng được thực hiện giống như hệ nhị phân chuẩn. Nếu hai số nhân là số dương, chúng được nhân với nhau bình thường. Nếu hai số nhân mang dấu âm, đầu tiên chúng sẽ được chuyển sang hệ bù 2. Kết quả của phép nhân được giữ nguyên với bit dấu là '0'.

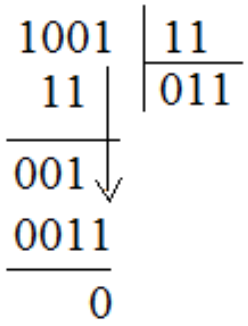
Nếu một trong hai số nhân là âm nó sẽ được chuyển sang dạng bù 2. Kết quả được chuyển sang dạng bù 2 và mang dấu âm với bit dấu bằng '1'.

2.4 Phép tính chia số nhị phân

Cũng tương tự như phép tính nhân, phép tính chia nhị phân tuân theo nguyên tắc giống phép chia số thập phân. Trong các máy tính số, phép tính trừ trong quá trình tính chia thường dùng hệ bù 2.

Phép chia của số có dấu được thực hiện giống như trong phép nhân. Các số âm được chuyển sang dương nhờ bù 2, sau đó phép chia được thực hiện. Nếu số chia và số bị chia có dấu đối

nhau, kết quả được chuyển sang dạng bù 2 và mang dấu âm. Nếu hai số chia cùng dấu, kết quả giữ lại và mang dấu dương.



Hình 2.3 Chia hai số nhị phân

Như vậy, phép tính cộng là phép tính cơ bản dùng để thực hiện các phép tính còn lại. Bằng việc kết hợp với các mạch dịch, mạch ghi nhớ, máy tính số có thể thực hiện các phép tính chỉ bằng cách tận dụng mạch tính cộng.

2.5 Trong các hệ đếm khác

Phép tính trong mã BCD

Như đã biết, trong nhiều hệ thống tính toán, mã BCD được sử dụng để thể hiện số thập phân. Mỗi chữ số thập phân được trình bày bằng 4 bit nhị phân.

Cộng nhỏ hơn 9: Việc thực hiện tính toán diễn ra bình thường như đối với hệ nhị phân. Vì tổng hai chữ số không vượt quá 9, không sinh ra số nhớ, do vậy kết quả vẫn đúng.

Cộng lớn hơn 9: Tuy nhiên, khi phép tính cộng có kết quả lớn hơn 9 và tạo ra số nhớ thì kết quả sẽ sai. Như đã biết, mã BCD có vùng cấm gồm 6 giá trị. Như vậy, khi kết quả phép tính rơi vào vùng cấm thì phải cộng thêm 6 hoặc (0110).

Phép tính trừ trong mã BCD khó hơn phép tính cộng. Nó đòi hỏi chuyển đổi sang dạng bù 2 và cộng. Vì giới hạn của chương trình, chúng ta không xét các phép tính khác trong mã BCD.

Phép tính trong hệ thập lục phân

Lấy phép tính trong hệ thập phân làm chuẩn, phép tính trong hệ thập lục phân cũng tương tự như vậy. Tuy nhiên, chữ số lớn nhất là ‘F’ chứ không phải ‘9’. Đối với phép tính cộng:

- Cộng hai chữ số trong dạng thập phân.

- Nếu tổng nhỏ hơn hoặc bằng 15, viết lại dưới dạng thập lục phân.
- Nếu tổng lớn hơn 15, trừ đi 16 và nhớ 1 vào số tiếp theo.

Số thập lục phân là cách viết gọn lại của số nhị phân. Phép trừ trong hệ thập lục phân được thực hiện như trong hệ nhị phân. Đầu tiên chuyển đổi sang số nhị phân, thực hiện theo hệ bù 2 và kết quả cuối cùng được đổi lại thành thập lục phân.

3

Cổng Cửa Luận Lý

3.1 Giới thiệu

Cổng cửa luận lý là các mạch điện tử xây dựng nên những biểu thức luận lý cơ bản nhất, hay còn gọi là biểu thức Boolean. Có ba cổng cửa luận lý cơ bản là cổng OR, cổng AND và cổng NOT. Các cổng cửa luận lý khác là kết quả của việc tổng hợp từ ba cổng cơ bản này. Đại số Boolean không chỉ dùng để phân tích và đơn giản hóa hệ thống. Nó còn được sử dụng để thiết kế mạch luận lý tùy theo mối quan hệ giữa đầu ra và đầu vào.

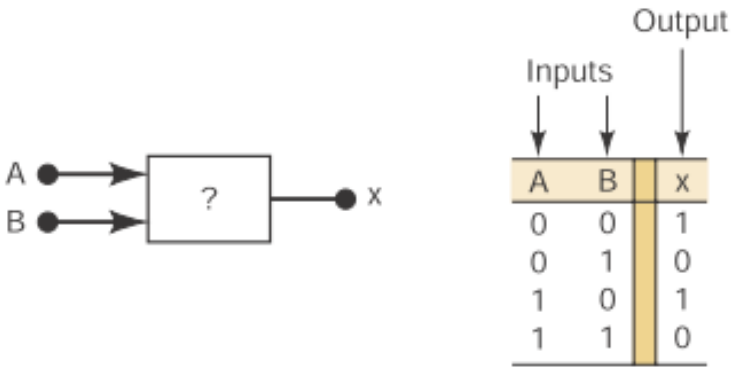
3.2 Biến số và hằng số Boolean

Đại số Boolean khác với đại số thông thường vì các biến số và hằng số Boolean chỉ có thể nhận một trong hai giá trị, '0' hoặc '1'. Trong các mạch điện tử, mức điện áp có thể được thể hiện bằng biến Boolean.

Như vậy, giá trị Boolean không thể hiện một số thực mà là trạng thái của một biến điện áp, hay gọi là mức luận lý. Vì chỉ có hai giá trị, đại số Boolean có phần dễ hơn đại số thông thường. Trong đại số Boolean, chỉ có ba phép toán là : OR, AND và NOT.

3.3 Bảng sự thật

Là một cách thể hiện trạng thái đầu ra của mạch luận lý tương ứng với giá trị đầu vào. Hình 3.1 minh họa một bảng sự thật cho mạch luận lý hai lối vào. Bảng sự thật liệt kê đầy đủ các trạng thái luận lý kết hợp tại đầu vào A và B cùng với mức luận lý tại lối ra x.

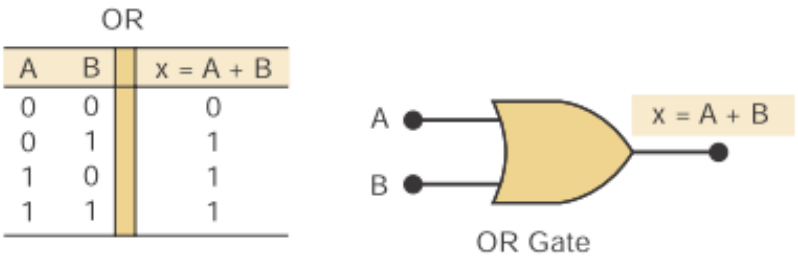


Hình 3.1 Bảng sự thật

Một bảng sự thật có thể có nhiều trạng thái kết hợp phụ thuộc số đầu vào của mạch. Đối với mạch có N lối vào, số trạng thái kết hợp bằng 2^N.

3.4 Phép toán OR

OR là phép toán cơ bản đầu tiên trong các phép toán Boolean. Lấy ví dụ là lò vi sóng, đèn bên trong sáng khi cửa mở HOẶC đèn được bật. Ký tự A được dùng để đại diện cho công tắc bật đèn và B dùng cho cửa. Ký tự x thể hiện trạng thái đèn. Bảng sự thật trong Hình 3.2.



Hình 3.2 Bảng sự thật phép toán OR và biểu tượng OR hai lối vào

Biểu thức Boolean cho phép toán OR là $x = A + B$.

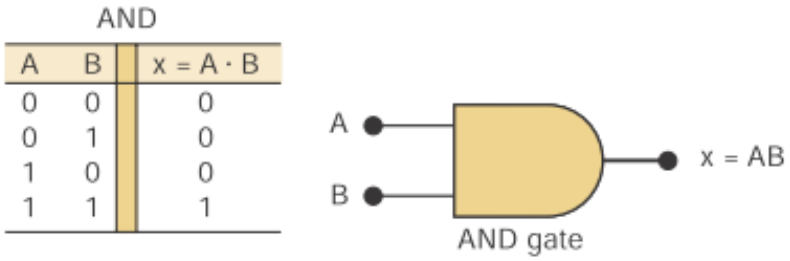
Trong đó, dấu '+' không thể hiện cho phép cộng thông thường, nó đại diện cho OR. Trong đại số Boolean, 1 là trạng thái cao, do vậy $1 + 1 = 1$ chứ không phải bằng 2.

Cổng OR trong các mạch điện tử là mạch có 2 hoặc nhiều lối vào với lối ra được tính bằng OR các trạng thái lối vào. Đối với các cổng OR nhiều lối vào, ví dụ 4, thì biểu thức Boolean có thể được viết thành $x = A + B + C + D$.

3.5 Phép toán AND

Đây là phép toán cơ bản thứ hai. Lấy ví dụ với máy giặt, nó chỉ quay khi nhấn nút hoạt động VÀ cửa máy đóng kín. Tương tự như trên, ta có hai lối vào quyết định trạng thái lối ra với biểu thức $x = A.B$

Trong đó dấu '.' đại diện cho phép tính AND chứ không phải dấu nhân bình thường. Tuy nhiên, tác động của AND lên các biến Boolean cũng tương tự như phép tính nhân.



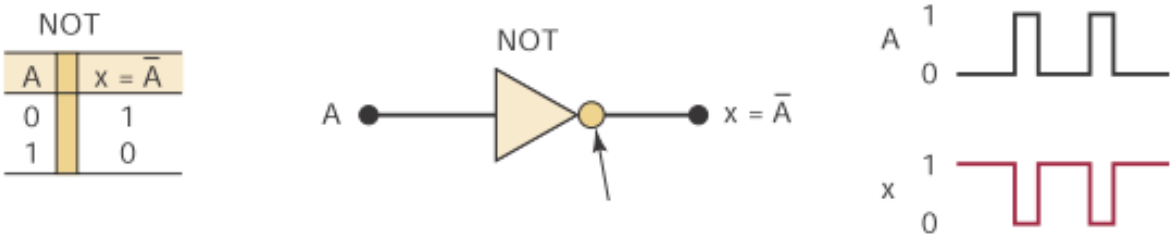
Hình 3.3 Bảng sự thật phép toán AND và biểu tượng AND hai lối vào

Đôi khi dấu '.' không được ghi nên biểu thức trở thành $x = AB$. Trường hợp có nhiều lối vào thì biểu thức được viết tương tự như trên, ví dụ $x = A.B.C = ABC$.

Cổng AND cũng được thể hiện như cổng OR với điểm khác biệt nhỏ ở biểu tượng.

3.6 Phép toán NOT

Không giống như OR và AND, phép toán NOT chỉ thực hiện trên biến một đầu vào. Ví dụ thực hiện phép toán NOT trên biến A là $x = \bar{A}$, minh họa trong Hình 3.4. Trong đó, dấu gạch trên thể hiện phép toán NOT hoặc thường gọi là phép đảo.

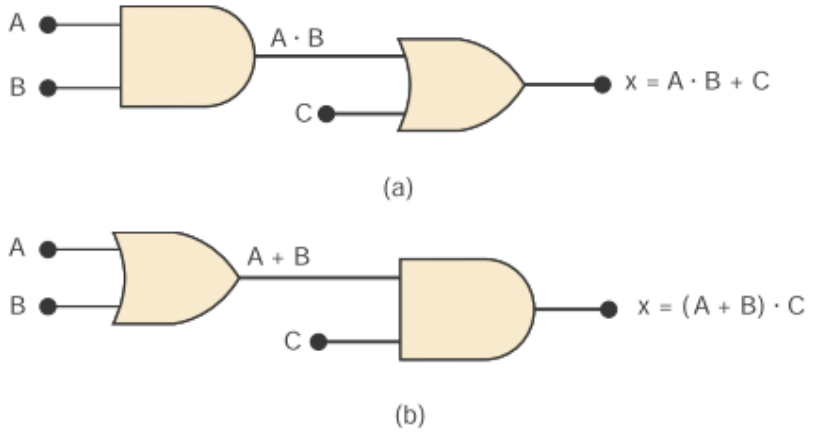


Hình 3.4 Bảng sự thật phép toán NOT và biểu tượng NOT

Cổng NOT hay gọi là cổng đảo. Nó chỉ có một lối vào và trạng thái lối ra luôn luôn nghịch đảo với lối vào.

3.7 Cổng kết hợp

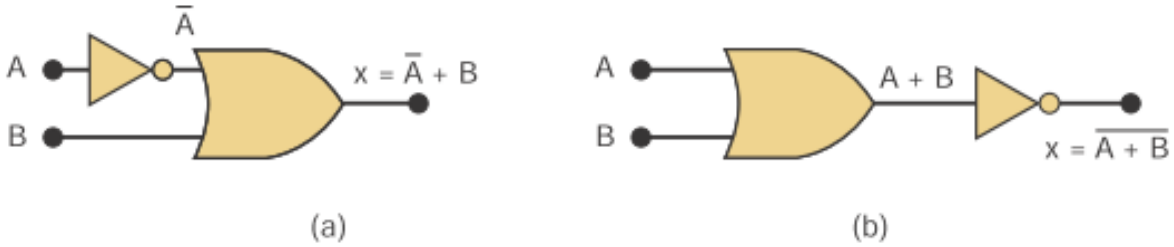
Tất cả các mạch luận lý cho dù phức tạp thế nào cũng có thể được biểu diễn bởi ba phép toán cơ bản trên vì các cổng OR, AND và NOT là thành phần cơ bản của các hệ thống số. Ví dụ trong Hình 3.5.



Hình 3.5 Mạch luận lý với biểu thức Boolean

Mạch có ba đầu vào A, B, và C với một lối ra *x*. Biểu thức thể hiện lối ra *x* được viết dựa trên các phép toán kết hợp của AND và OR. Cũng giống như đại số thông thường, phép toán AND được tính trước trừ khi trong biểu thức có thành phần trong dấu ngoặc.

Trong khi đó, mạch có cổng NOT có sự khác biệt như trong Hình 3.6. Cổng NOT cuối cùng đảo trạng thái của cả biểu thức trước đó.

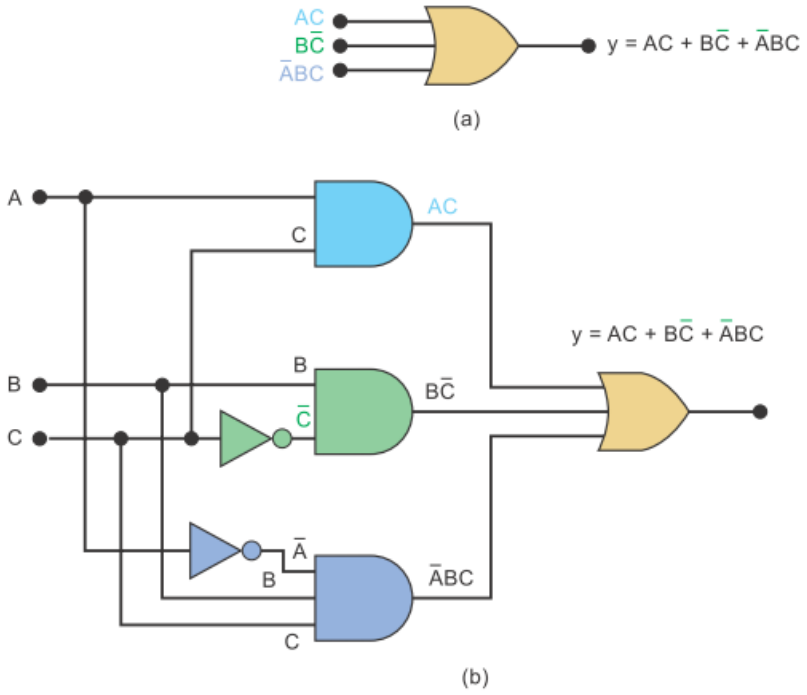


Hình 3.6 Mạch luận lý có cổng NOT

Để thể hiện đầy đủ trạng thái của lối ra, thông thường ta sẽ lập bảng sự thật với các trạng thái lối vào thay đổi. Đồng thời với đó là kết quả qua từng cổng luận lý và cuối cùng là biểu thức kết hợp.

Xây dựng mạch từ biểu thức Boolean

Khi hoạt động của một mạch điện tử được định nghĩa bởi một biểu thức Boolean, ta có thể dễ dàng triển khai sơ đồ mạch dựa trên biểu thức đó. Ví dụ trong Hình 3.7

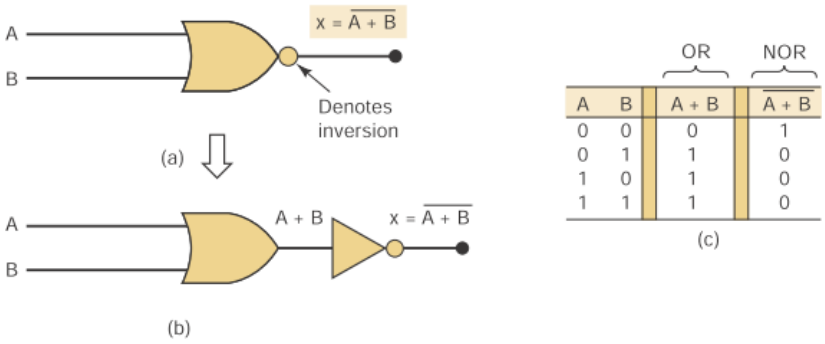


Hình 3.7 Xây dựng mạch dựa trên biểu thức Boolean

3.8 Các cổng luận lý khác

Cổng NOR

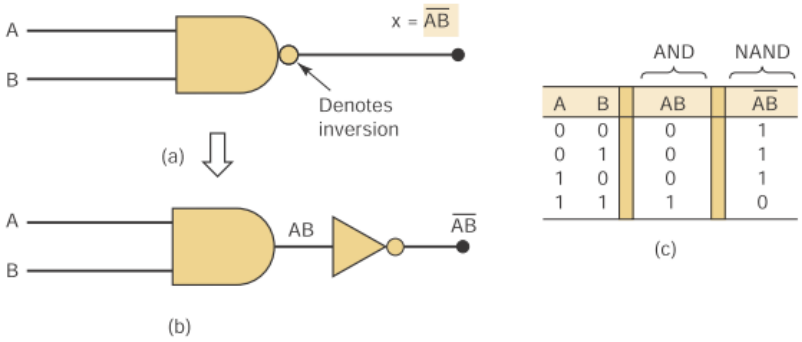
Được xây dựng dựa trên cổng OR và NOT. Hình minh họa 3.8 với cổng NOR hai lối vào.



Hình 3.8 Cổng NOR và bảng sự thật

Cổng NAND

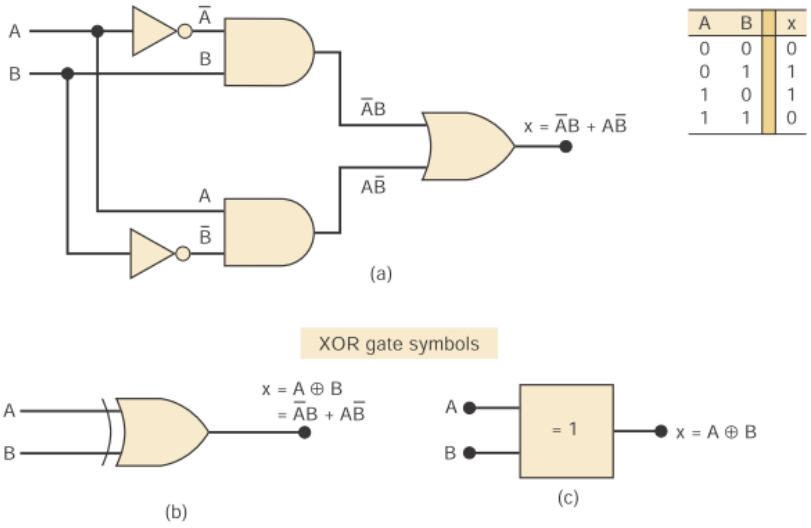
Được xây dựng dựa trên cổng AND và NOT. Hình minh họa 3.9 với cổng NAND hai lối vào.



Hình 3.9 Cổng NAND và bảng sự thật

Cổng XOR

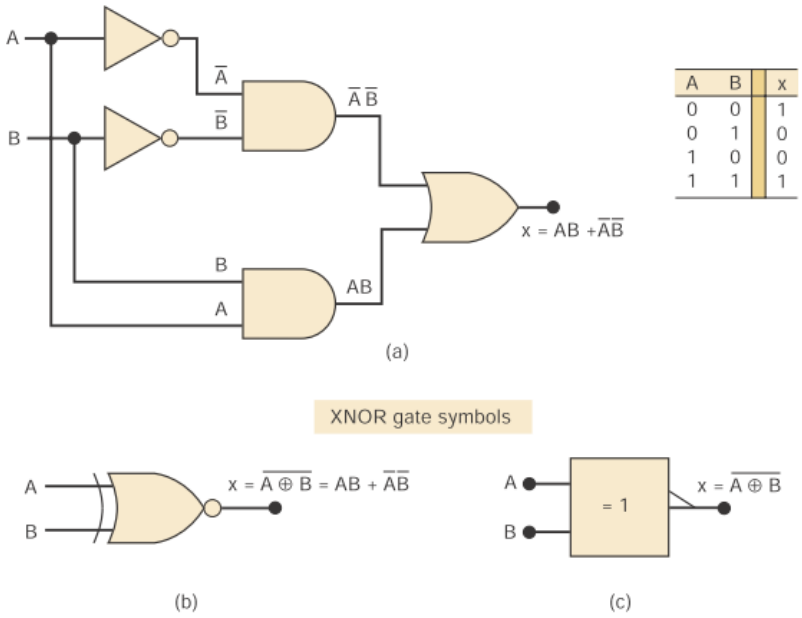
Có thể được viết là EX-OR có hai lối vào và một lối ra. Cổng XOR thường chỉ có hai lối vào, tuy nhiên nó cũng có thể được xây dựng với nhiều lối vào. Biểu thức của cổng XOR hai lối vào $x = A \oplus B = \bar{A}B + A\bar{B}$



Hình 3.10 Cổng XOR và bảng sự thật

Cổng XNOR

Có thể được viết là EX-NOR có hai lối vào và một lối ra. Cổng XNOR được thực hiện bằng cách đảo cổng XOR. Biểu thức của cổng XNOR hai lối vào $x = \overline{A \oplus B} = AB + \bar{A}\bar{B}$



Hình 3.11 Cổng XNOR và bảng sự thật

3.9 Các đinh lý Boolean

Để đơn giản hóa biểu thức luận lý người ta áp dụng các định lý Boolean. Nhóm định lý thứ nhất:

- $x.0 = 0$
- $x.1 = x$
- $x.x = x$
- $x.\bar{x} = 0$
- $x + 0 = x$
- $x + 1 = 1$
- $x + x = x$
- $x + \bar{x} = 1$

Nhóm định lý thứ hai:

- $x + y = y + x$
- $x.y = y.x$
- $x + (y + z) = (x + y) + z = x + y + z$
- $x(yz) = (xy)z = xyz$

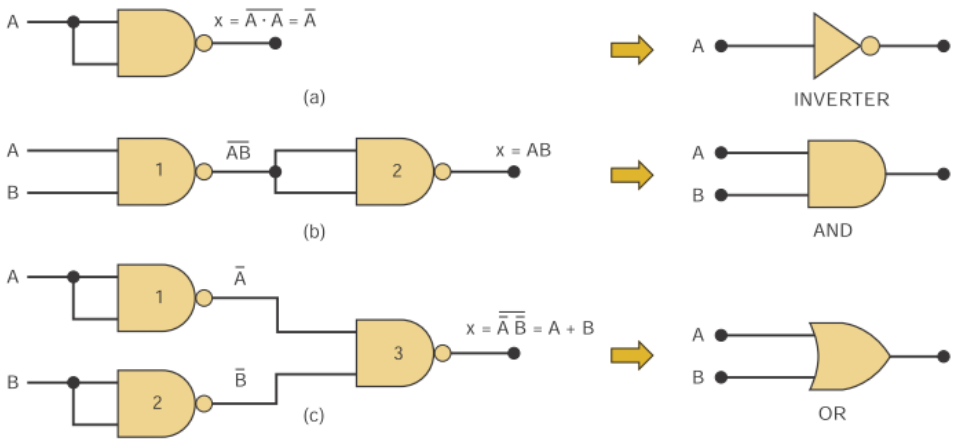
- $x(y+z) = xy+xz$
- $(w+x)(y+z) = wy+xy+wz+xz$
- $x+xy = x$
- $x+\bar{x}y = x+y$
- $\bar{x}+xy = \bar{x}+y$

Định lý DeMorgan được sử dụng rất nhiều nhằm giản ước các biểu thức có phép toán đảo của tổng hoặc tích:

- $(\overline{x+y}) = \bar{x}.\bar{y}$
- $(\overline{x.y}) = \bar{x}+\bar{y}$

Tính phổ quát của cổng NAND và NOR

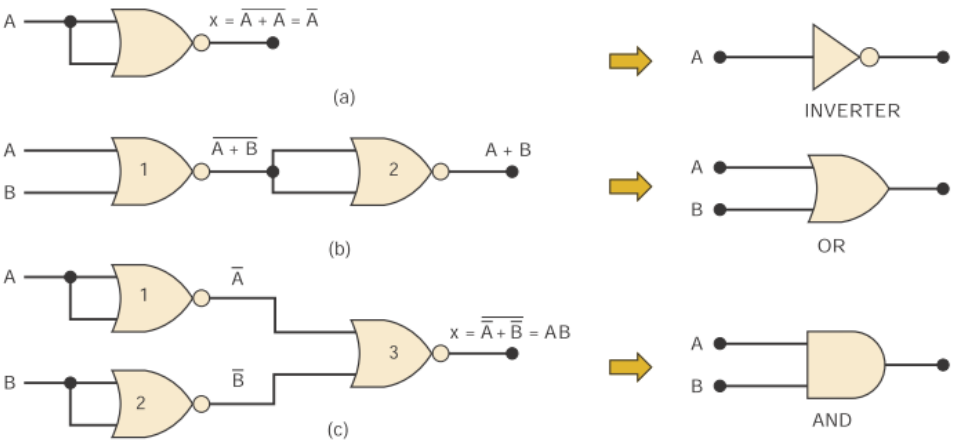
Ngoài việc sử dụng các cổng cửa căn bản là OR, AND và NOT. Bất kỳ biểu thức nào cũng có thể được triển khai chỉ bằng việc sử dụng cổng NAND (hình 3.12) hoặc cổng NOR (hình 3.13).



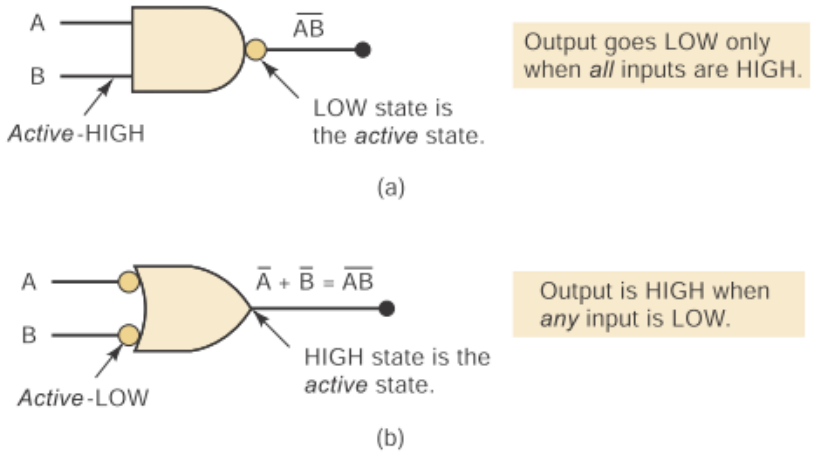
Hình 3.12 Xây dựng các cổng cửa khác bằng cổng NAND

Diễn dịch biểu tượng luận lý

Khi một lối vào hoặc lối ra của một biểu tượng mạch luận lý không có ký hiệu dấu tròn thì nó được xem là tác động mức cao. Ngược lại, khi dấu tròn được vẽ thêm lên biểu tượng đó thì nó được xem là tác động mức thấp. Hình 3.14 minh họa cổng NAND sau khi biến đổi qua định lý Boolean. Hình 3.14a với cổng NAND chuẩn, nó có lối ra tác động mức thấp và các lối vào tác



Hình 3.13 Xây dựng các cổng cửa khác bằng cổng NOR



Hình 3.14 Diễn dịch cổng NAND

động mức cao. Nghĩa là lối ra chỉ xuống thấp khi tất cả lối vào lên cao. Trong khi đó Hình 3.14b được diễn giải là lối ra lên cao khi bất kỳ lối vào nào xuống thấp.

Quy tắc chung: Khi vẽ mạch, lựa chọn cổng luận lý sao cho lối ra có dấu tròn nối tới cổng với lối vào cũng có dấu tròn, và lối ra không có dấu tròn đi vào cổng có lối vào không có dấu tròn.

Dán nhãn tác động mức thấp đối với các tín hiệu bằng thanh ngang bên trên. Ví dụ: $\overline{Reset}, \overline{EN}, \overline{Clear}$.

4

Các Phương Pháp Đơn Giản Hóa Mạch

4.1 Giới thiệu

Mạch luận lý kết hợp không có tính năng nhớ, trạng thái lối ra phụ thuộc vào các trạng thái hiện thời của lối vào. Để đơn giản hóa mạch kết hợp, hai phương pháp được áp dụng là: dùng các định lý Boolean và kỹ thuật giản đồ.

4.2 Tổng của tích và tích của tổng

Kỹ thuật đơn giản hóa mạch đầu tiên là phân tích mạch và thể hiện ở dạng tổng của tích (Sum-of-products). Ví dụ:

- $ABC + \bar{A}B\bar{C}$
- $AB + \bar{A}\bar{B}C + \bar{C}D + D$
- $\bar{A}B + \bar{C}D + EF + G\bar{H}$

Trong những biểu thức trên đều chứa các phép toán AND và được OR lại với nhau. Chú ý rằng, trong dạng này, dấu của phép NOT không phủ lên hơn một biến cùng một số hạng.

Tích của tổng (POS) với dạng biểu thức chứa các số hạng OR và được AND với nhau. Ví dụ:

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

Phương pháp này dựa trên dạng tổng của tích (SOP) nên không được trình bày thêm.

Danh pháp tổng và tích

Các ký hiệu $\sum$ và $\prod$ được dùng để thể hiện biểu thức SOP hoặc POS. Ví dụ:

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

Các thứ hạng sẽ được viết theo thứ tự tăng dần, các biến thật mang giá trị 1 và biến bù thể hiện bằng 0. Biểu thức được viết lại:

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

Các giá trị thể hiện (0001), (0101), (1000), (1001) và 1111. Chúng có thể được viết lại dưới dạng thập phân $f(A,B,C,D) = \sum 1,5,8,9,15$

Dạng bù của $f(A,B,C,D)$ là $f'(A,B,C,D)$ bao gồm các giá trị còn lại trong dải giá trị tạo bởi các biến có trong hàm. Như vậy, trong ví dụ trên đối với hàm SOP thì $f'(A,B,C,D) = \sum 0,2,3,4,6,7,10,11,12,13,14$.

Đặc biệt, lấy ví dụ nếu một hàm Boolean SOP $f(A,B,C) = \sum 0,1,4,7$ thì POS $f(A,B,C) = \prod 2,3,5,6$ và $f'(A,B,C) = \sum 2,3,5,6 = \prod 0,1,4,7$.

4.3 Đơn giản hóa mạch

Mục đích là nhằm tối giản việc sử dụng biến hoặc số hạng sau khi biểu thức luận lý được hình thành. Biểu thức mới có tính năng tương đồng biểu thức cũ nhưng sử dụng ít cổng cửa hơn.

Áp dụng định lý Boolean

Dựa khá nhiều vào kinh nghiệm và tính sáng tạo của người thực hiện. Ngoài ra, kết quả cuối cùng cũng khó có thể xác định mức độ tối giản. Quy trình gồm các bước chính:

- Biểu thức được chuyển về dạng SOP bằng cách sử dụng định lý DeMorgan và Boolean khác
- Sau khi có dạng SOP, lấy thừa số chung để giảm số lượng các số hạng.

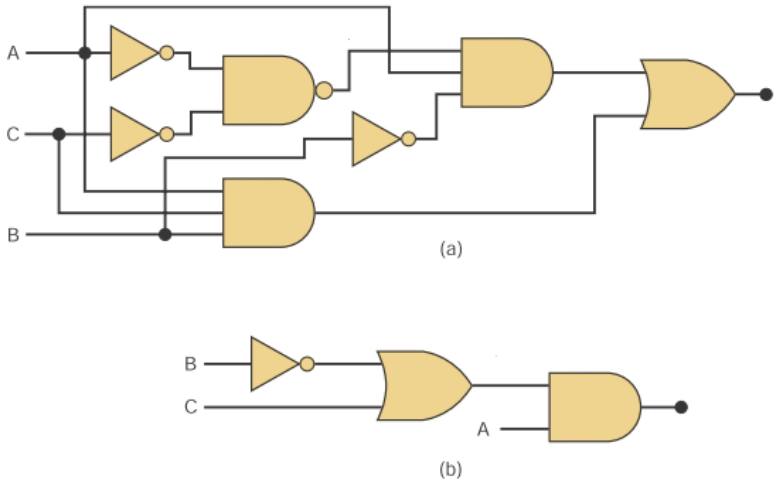
Ví dụ minh họa trong Hình 4.1. Biểu thức lối ra $z = ABC + A\bar{B} \cdot (\overline{\bar{A}\bar{C}})$. Sử dụng định lý DeMorgan để đơn giản các số hạng có phép toán NOT.

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

Tiếp tục áp dụng các định lý Boolean và gom thừa số:

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

Quy trình thiết kế:



Hình 4.1 Đơn giản hóa mạch theo SOP

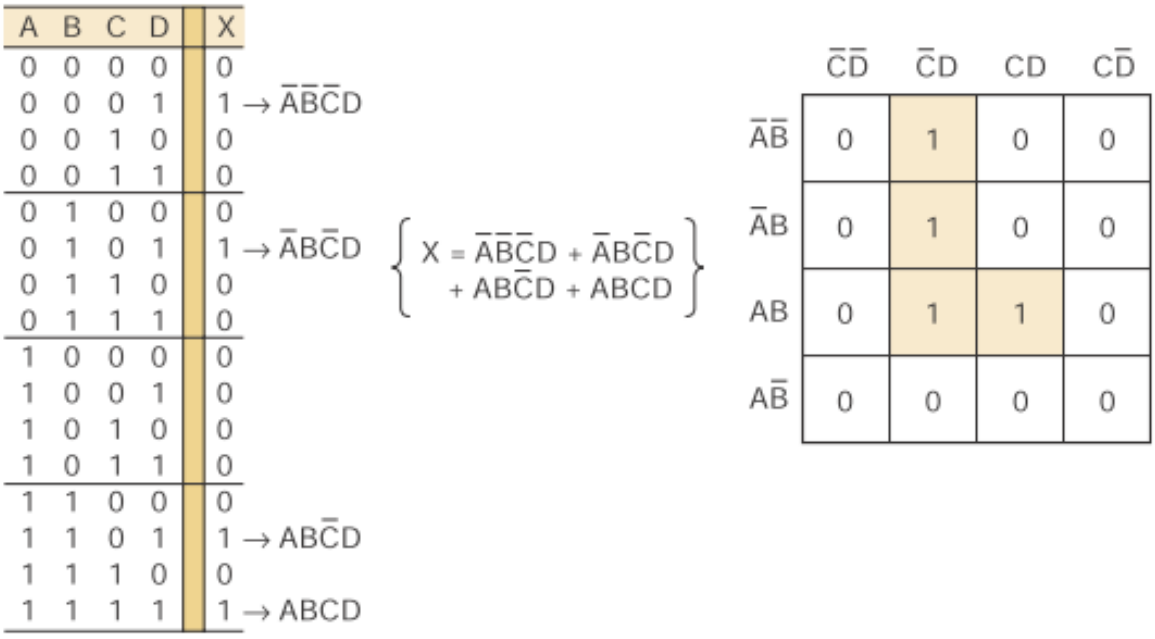
- Lập bảng sự thật mô tả hoạt động của mạch
- Viết số hạng dùng AND cho các trường hợp lối ra bằng 1
- Viết biểu thức tổng của tích cho lối ra
- Đơn giản hóa lối ra nếu có thể
- Thực hiện mạch bằng cổng cửa luận lý

4.4 Giản đồ Karnaugh

Đây là phương pháp đơn giản hóa biểu thức hoặc chuyển đổi bảng sự thật sang mạch luận lý. Trên thực tế, giản đồ Karnaugh chỉ có thể áp dụng với số lượng biến nhỏ hơn 6. Giống như bảng sự thật, giản đồ Karnaugh thể hiện mối quan hệ giữa trạng thái lối ra và lối vào. Minh họa giản đồ Karnaugh trong Hình 4.2

1. Bảng sự thật cho thấy giá trị lối ra lần lượt đối với sự kết hợp của các giá trị lối vào. Từng trường hợp trong bảng sự thật được thể hiện bởi một ô trong giản đồ Karnaugh.
2. Các ô trong giản đồ K được viết sao cho những ô liền nhau theo chiều ngang và chiều dọc chỉ khác nhau một biến. Đồng thời, từng ô trên hàng đầu được xem như nối tiếp với hàng cuối. Có thể tưởng tượng rằng giản đồ được uốn lại để các hàng nối lại với nhau. Tương tự, các ô ở cột ngoài cùng bên trái nối tiếp với các ô tại cột phải.
3. Thứ tự các ô được ghi phải đảm bảo chiều ngang và dọc có thể uốn lại thành ô nối tiếp. Theo chiều dọc: $\bar{A}\bar{B}, \bar{A}B, AB, A\bar{B}$ và chiều ngang từ trái qua: $\bar{C}\bar{D}, \bar{C}D, CD, C\bar{D}$

4. Một khi giản đồ K đã được điền đầy đủ với giá trị ‘0’ và ‘1’, biểu thức SOP cho lối ra được thực hiện bằng cách OR các ô chứa ‘1’.



Hình 4.2 Giản đồ Karnaugh và bảng sự thật

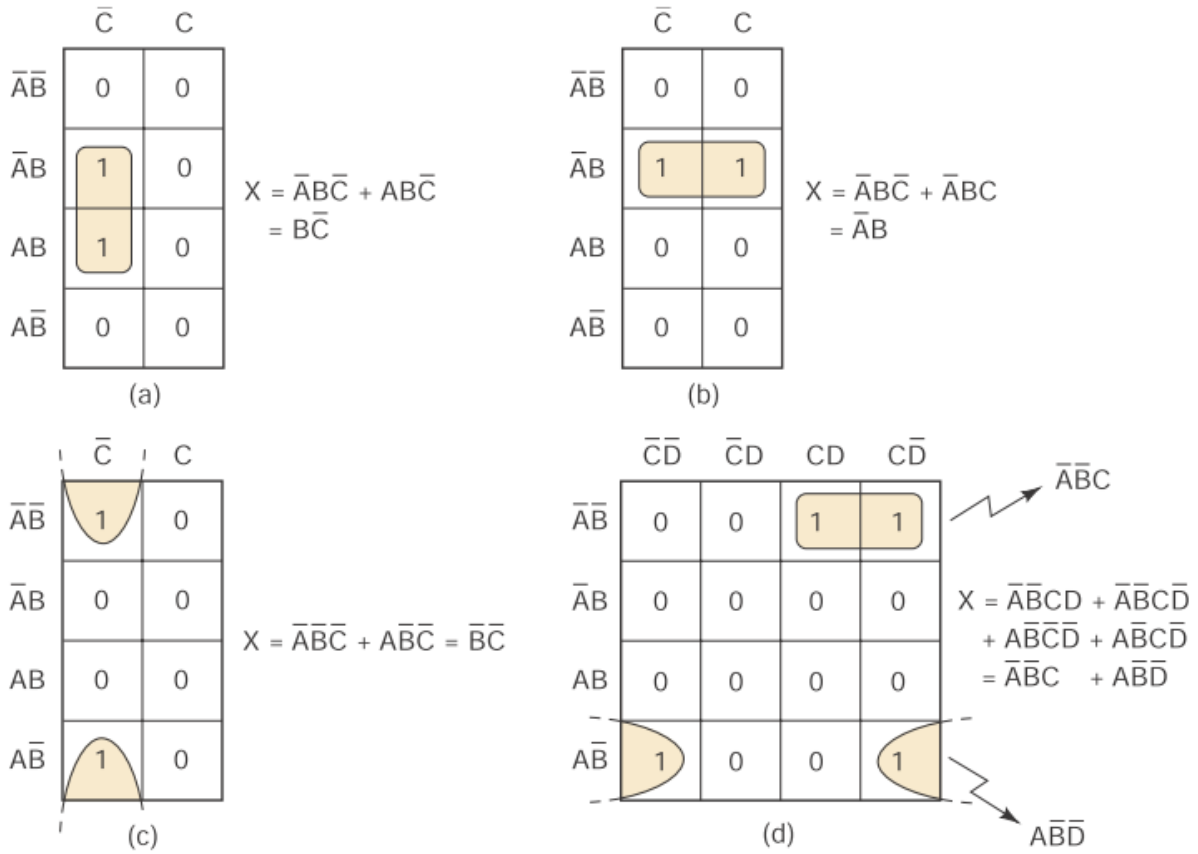
Để đơn giản hóa biểu thức, những ô chứa ‘1’ có thể được gom lại với nhau. Hình 4.3 dùng cho trường hợp gom hai ô kề nhau. Gom hai ô kề nhau loại bỏ biến mà xuất hiện đồng thời ở cả hai dạng bù nhau.

Hình 4.4 dùng cho trường hợp gom bốn ô kề nhau. Gom bốn ô kề nhau loại bỏ hai biến mà xuất hiện đồng thời ở cả hai dạng bù nhau.

Hình 4.5 dùng cho trường hợp gom tám ô kề nhau. Gom tám ô kề nhau loại bỏ ba biến mà xuất hiện đồng thời ở cả hai dạng bù nhau.

Khi một biến xuất hiện ở dạng bù nhau trong cùng một nhóm được gom thì sẽ bị loại bỏ. Các biến không đổi trong các ô của nhóm sẽ được giữ lại. Quy trình áp dụng giản đồ K:

1. Lập giản đồ K và viết ‘1’ vào ô tương ứng theo bảng sự thật. Các ô còn lại viết 0.
2. Gom các nhóm 1 và xác định các ô chứa ‘1’ độc lập.
3. Gom các nhóm 8 đối với cả các ô chứa ‘1’ đã được gom.
4. Gom các nhóm 4 cho các ô chứa ‘1’ chưa được gom. Hạn chế gom nhiều nhóm trùng lặp.
5. Gom nhóm đôi cho các ô chứa ‘1’ chưa dùng.
6. Tính tổng OR của các số hạng sau khi gom.



Hình 4.3 Gom hai ô kề nhau

Biểu diễn giản đồ K từ một biểu thức

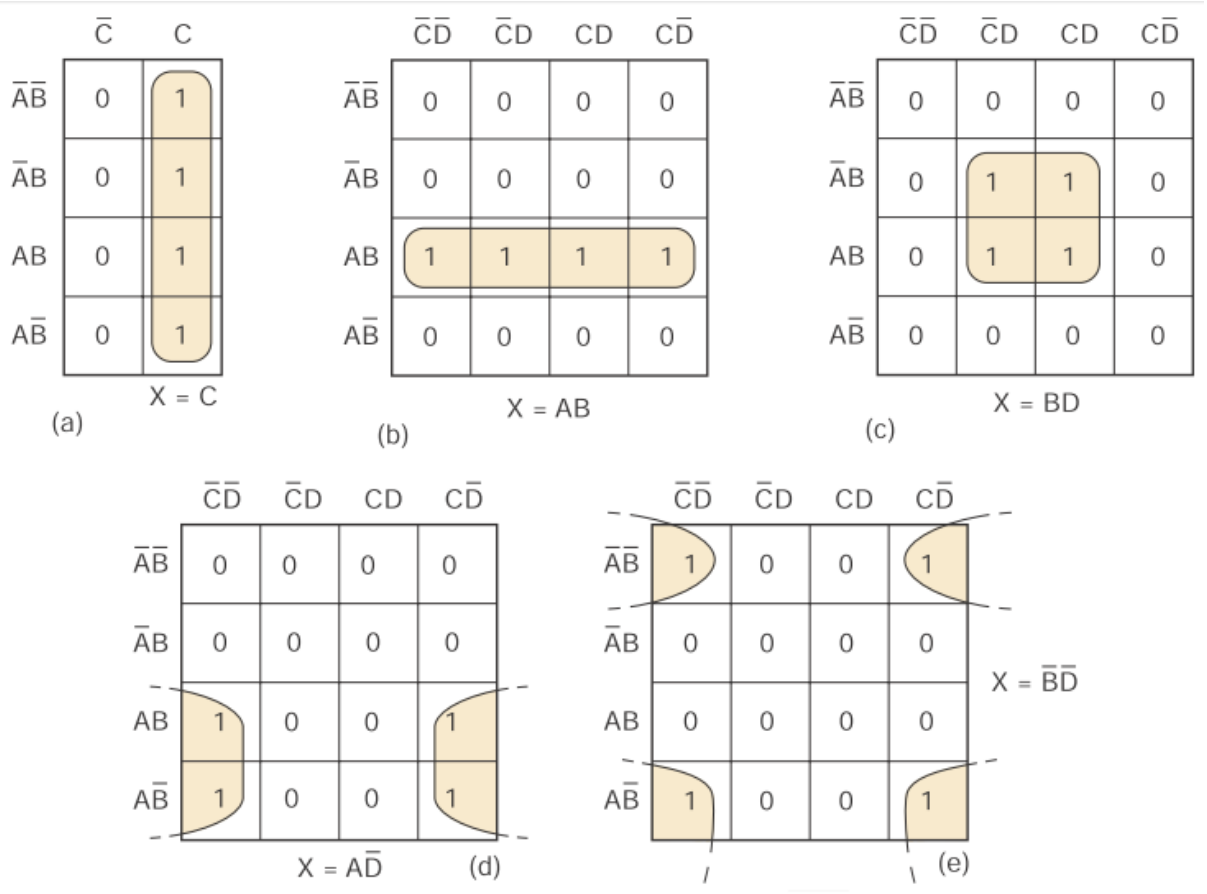
Nếu như trạng thái lối ra được thể hiện bằng biểu thức thay vì bảng sự thật, giản đồ K được biểu diễn như sau:

- Chuyển biểu thức về dạng SOP
- Với mỗi số hạng, viết ‘1’ vào từng ô của giản đồ K nếu nó chứa tổ hợp các biến giống nhau.
- Viết ‘0’ vào các ô còn lại.

Các trường hợp đặc biệt

Một số mạch được thiết kế có trạng thái lối ra không xác định đối với một số trạng thái lối vào nhất định. Nghĩa là những trường hợp lối vào này không cần quan tâm đến trạng thái lối ra. Để thể hiện những trạng thái này, trong giản đồ K chúng được đánh dấu ‘x’ vào các ô tương ứng.

Khi tiến hành đơn giản hóa trên giản đồ K, những ô ‘x’ có thể được nhận giá trị 0 hoặc 1 tùy ý. Mục đích nhằm tối ưu hóa việc gom các ô của giản đồ.



Hình 4.4 Gom bốn ô kề nhau

Như vậy, giản đồ K cho phép đơn giản hóa mạch một cách tuần tự và có quy luật rõ ràng hơn việc áp dụng định lý Boolean. Tuy nhiên, phụ thuộc vào kinh nghiệm và kỹ năng tính toán Boolean, sử dụng các định lý Boolean để đơn giản hóa mạch hoặc biểu thức vẫn là một sự lựa chọn của nhiều người.

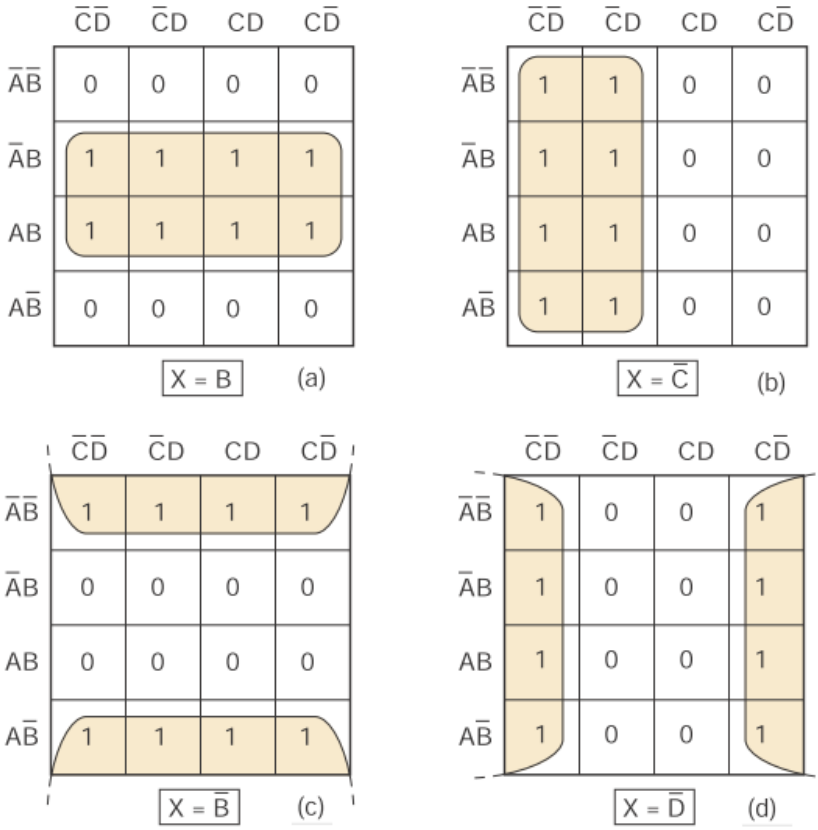
Đối với những mạch có số biến lớn, các kỹ thuật phức tạp hơn được triển khai. Hầu hết những kỹ thuật này được lập trình trên máy tính để đem lại kết quả nhanh và chính xác.

4.5 Phương pháp Quine-McCluskey

Thường dùng cho những biểu thức có số biến lớn hơn 5. Phương pháp này dựa trên định lý bù: $XY + X\bar{Y} = X$ với X có thể là biến, biểu thức hoặc số hạng trong khi Y là biến. Quy trình đơn giản hóa biểu thức như sau:

1. Biểu thức Boolean được triển khai mở rộng

2. Các số hạng trong biểu thức được chia thành nhóm dựa vào số lượng ‘1’ của chúng.



Hình 4.5 Gom tám ô kề nhau trong giản đồ K

3. Những số hạng của nhóm đầu được đối chiếu với số tiếp theo của nhóm sau để tìm kiếm tương hợp. Những số được coi là tương hợp khi chúng chỉ khác nhau một giá trị. Cặp số tương hợp được thay bằng một số với vị trí của giá trị lệch được thay bằng dấu $(-)$. Những số hạng trong bảng đầu tiên không có tương hợp được đánh dấu $(*)$. Những số tương hợp có dấu (v).

4. Những số hạng trong nhóm hai được so sánh với nhóm ba. Tuần tự các bước như trên.

5. Quá trình tiếp tục cho đến hết các nhóm. Sau lần đầu, các số hạng tương hợp được nhóm thành bảng lần hai.

6. Đối với bảng lần hai, khi so sánh các số hạng thì dấu $(-)$ cũng phải xét sự trùng nhau. Quá trình tiếp nối cho đến khi không thể tạo bảng mới.

7. Biểu thức tối giản bao gồm các số hạng sót lại.

Ví dụ: $\bar{A}BC + \bar{A}\bar{B}D + A\bar{C}D + B\bar{C}\bar{D} + \bar{A}B\bar{C}D$ sẽ được triển khai thành $\bar{A}BCD + \bar{A}BC\bar{D} + \bar{A}\bar{B}CD + \bar{A}\bar{B}\bar{C}D + AB\bar{C}D + A\bar{B}\bar{C}D + AB\bar{C}\bar{D} + \bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D$

Lập nhóm và tạo bảng được minh họa trong Hình 4.6.

Tiếp theo, từng số hạng của nhóm 1 được đối chiếu với nhóm 2. Kết quả trong Hình 4.7

A	B	C	D
0	0	0	1
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	1
1	1	0	0
1	1	0	1

A	B	C	D	
0	0	0	1	✓
0	1	0	0	✓
0	0	1	1	✓
0	1	0	1	✓
0	1	1	0	✓
1	0	0	1	✓
1	1	0	0	✓
0	1	1	1	✓
1	1	0	1	✓

A	B	C	D	
0	0	–	1	✓
0	–	0	1	✓
–	0	0	1	✓
0	1	0	–	✓
0	1	–	0	✓
–	1	0	0	✓
0	–	1	1	✓
0	1	–	1	✓
–	1	0	1	✓
0	1	1	–	✓
1	–	0	1	✓
1	1	0	–	✓

Hình 4.6 Phân nhóm và lập bảng lần 1

A	B	C	D	
0	–	–	1	*
–	–	0	1	*
0	1	–	–	*
–	1	0	–	*

0001	0011	0100	0101	0110	0111	1001	1100	1101		
✓	✓		✓		✓				0– –1	$P \rightarrow \overline{A}.D$
✓			✓			✓		✓	– –01	$Q \rightarrow \overline{C}.D$
		✓	✓	✓	✓				01– –	$R \rightarrow \overline{A}.B$
		✓	✓				✓	✓	–10–	$S \rightarrow B.\overline{C}$

Hình 4.7 Bảng lần hai và các số hạng tương hợp gốc

Mỗi số hạng được viết lại bằng một ký tự, sau đó chúng được kiểm chứng lần lượt. Bước tiếp theo là viết biểu thức dạng POS: $(P+Q)(P)(R+S)(P+Q+R+S)(R)(P+R)(Q)(S)(Q+S)$

Có thể thấy các số hạng đặc biệt P,Q,R,S được sử dụng toàn bộ để thể hiện các số hạng gốc nên biểu thức gốc có thể được viết lại thành $\bar{A}D+\bar{C}D+\bar{A}B+B\bar{C}$

5

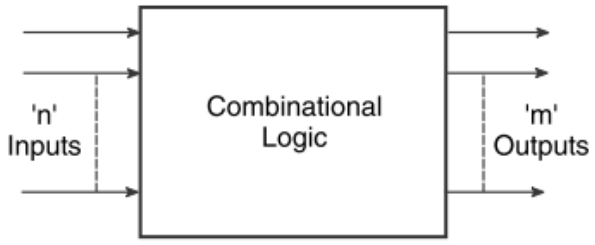
Mạch Tính Toán

5.1 Giới thiệu

Trong chương này, chúng ta cùng tìm hiểu về những mạch kết hợp và thành phần tạo nên chúng. Những cổng cửa luận lý trình bày trong chương trước là nền tảng để xây dựng mạch kết hợp.

5.2 Mạch kết hợp

Trạng thái đầu ra tại bất cứ thời điểm nào cũng chỉ phụ thuộc vào tổ hợp trạng thái các lối vào. Những mạch điện tử khác được gọi là mạch tuần tự sẽ trình bày trong chương sau. Hình 5.1 minh họa sơ đồ khối của mạch kết hợp với *n* biến lối vào và *m* biến lối ra.



Hình 5.1 Mạch tổ hợp tổng quát

Các đầu vào có thể được phân chia theo mục đích khác nhau, một số đầu vào có thể ở dạng bù (tác động mức thấp).

5.3 Thực hiện mạch kết hợp

Các bước cơ bản để thiết kế mạch như sau:

1. Xác định vấn đề
2. Xác định biến vào/ra
3. Mô tả mối quan hệ giữa các biến vào/ra
4. Lập bảng sự thật
5. Viết biểu thức Boolean cho biến lối ra dựa trên các biến đầu vào
6. Đơn giản hóa biểu thức
7. Thực hiện mạch

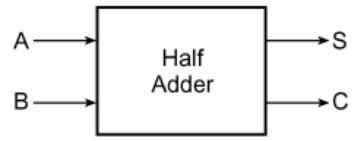
Trong quá trình thực hiện mạch, một số chú ý cần quan tâm:

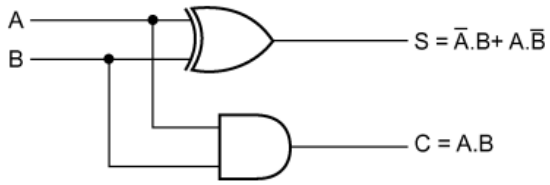
- Sử dụng tối thiểu số lượng cổng cửa, mỗi cổng cửa có càng ít lối vào càng tốt.
- Giảm thiểu đường kết nối.

Mạch cộng bán phần

Mạch có chức năng cộng 2 bit. Biểu thức SUM $S = A\bar{B} + \bar{A}B$ và CARRY $C = AB$ minh họa trên Hình 5.2

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



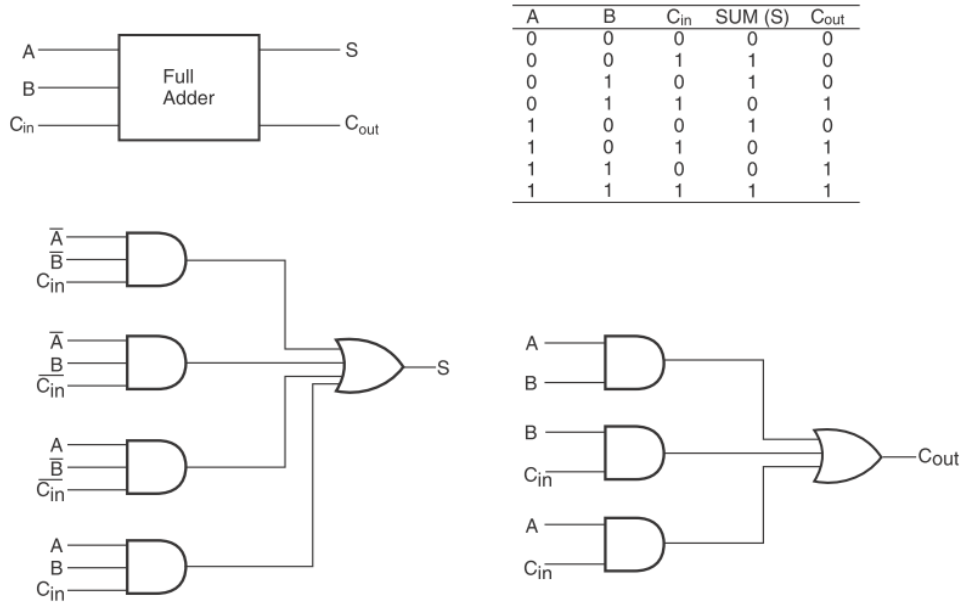


Hình 5.2 Mạch cộng bán phần

Mạch trên được thực hiện dưới dạng đơn giản nhất. Về mặt lý thuyết, mạch có thể được thực hiện chỉ bằng cổng NAND hoặc NOR.

Mạch cộng toàn phần

Được dùng để cộng 3 bit với lối ra SUM và CARRY. Trong quá trình thực hiện mạch cộng, lối ra CARRY từ mạch trước bổ sung thêm 1 bit nên mạch cộng toàn phần là cần thiết để thực hiện đầy đủ mạch cộng. Mạch cộng được trình bày trong Hình 5.3.



Hình 5.3 Mạch cộng toàn phần

Biểu thức Boolean có được: $S = \bar{A}\bar{B}C_{in} + \bar{A}B\bar{C_{in}} + A\bar{B}\bar{C_{in}} + ABC_{in}$ và $C_{out} = \bar{A}BC_{in} + A\bar{B}C_{in} + ABC_{in} + ABC_{in}$. Sử dụng giản đồ K và thu được $C_{out} = BC_{in} + AB + AC_{in}$

Mạch cộng toàn phần có thể được xem như kết hợp của hai mạch cộng bán phần và cổng OR. Biểu thức viết lại: $S = \bar{C_{in}}(\bar{A}B + A\bar{B}) + C_{in}(AB + \bar{A}\bar{B}) = \bar{C_{in}}(\bar{A}B + A\bar{B}) + C_{in}(\overline{\bar{A}B + A\bar{B}})$

Tương tự, $C_{out} = AB(1 + C_{in}) + C_{in}(\bar{A}B + A\bar{B}) = AB + C_{in}(\bar{A}B + A\bar{B})$

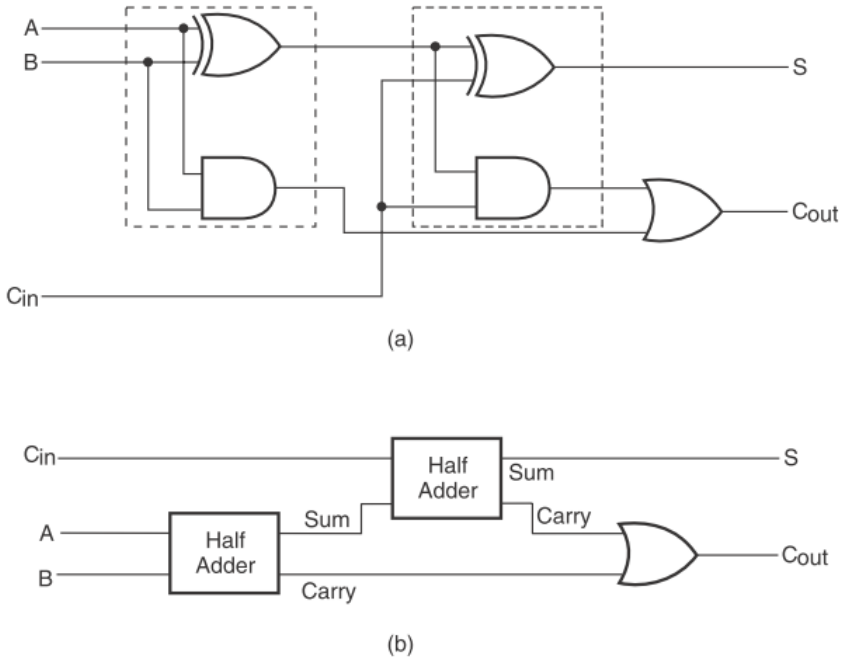
Mạch trừ: Dựa vào nguyên tắc trừ số nhị phân, mạch trừ được xây dựng trên nền tảng của mạch cộng như minh họa trong Hình 5.5. Mạch trừ toàn phần cũng được ghép từ các mạch trừ bán phần.

Đơn vị luận lý và số học: ALU (*arithmetic logic unit)* là một khối chức năng có thể thực hiện đồng thời các phép toán luận lý và số học.

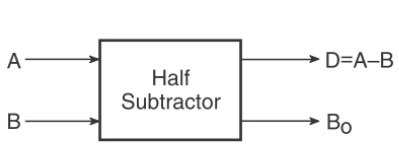
5.4 Mạch tích hợp

Mạch dồn kênh

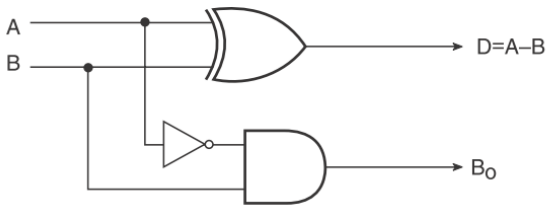
MUX (*multiplexer*) có nhiều hơn một lối vào, một lối ra và các đường chọn kênh như minh họa trên Hình 5.6.



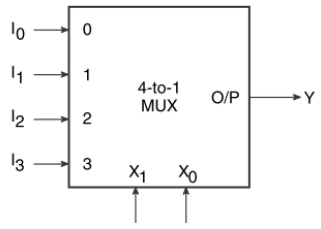
Hình 5.4 Mạch cộng toàn phần xây dựng từ mạch bán phần



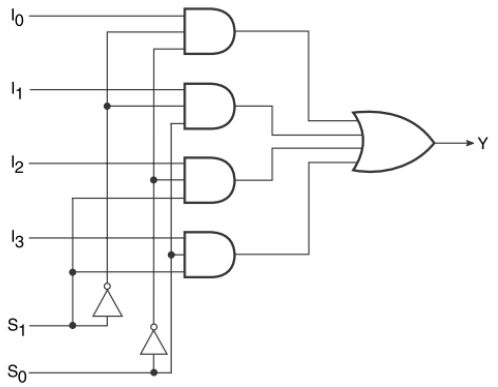
A	B	D	B_O
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



Hình 5.5 Mạch trừ bán phần trên nền tảng mạch cộng



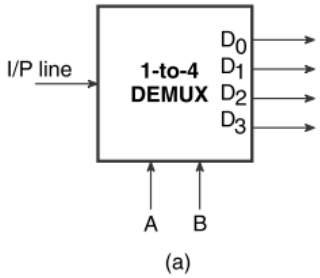
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



Hình 5.6 Mạch dồn kênh 4-1 và bảng sự thật

Mạch phân kênh

Hình 5.7 minh họa tổng quát mạch phân kênh có một lối vào, 2^n lối ra với n đường lựa chọn.



(a)

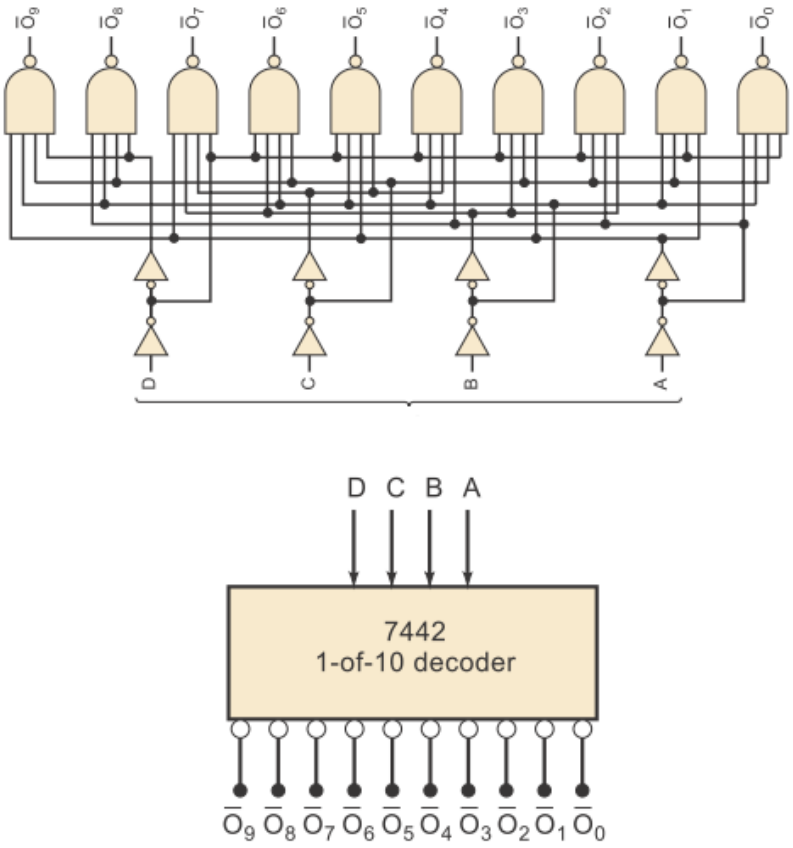
I/P	Select		O/P			
	A	B	D_0	D_1	D_2	D_3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

(b)

Hình 5.7 Mạch phân kênh dồn kênh 4-1 và bảng sự thật

Mạch mã hóa BCD sang thập phân

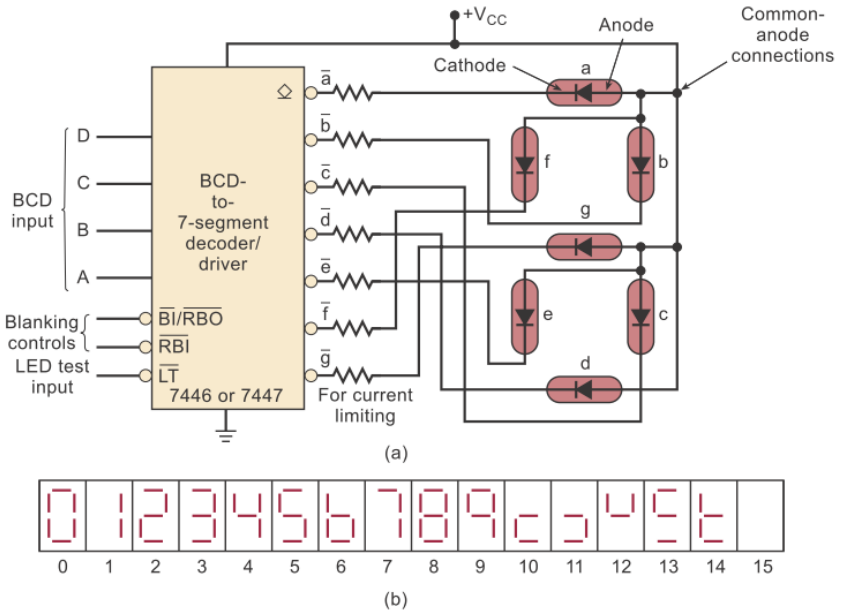
Mỗi khi trạng thái các đầu vào BCD phù hợp, đầu ra thập phân sẽ thay đổi trạng thái. Trong trường hợp đầu vào BCD nhận giá trị vùng cấm, đầu ra không có thay đổi. Minh họa trong Hình 5.8 còn được gọi là mã hóa 4-sang-10.



Hình 5.8 Mạch mã hóa trên IC 7442

Mã hóa BCD sang 7 đoạn

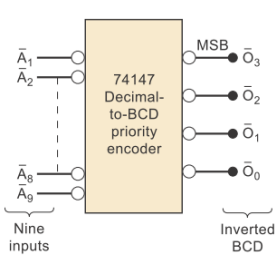
Để hiển thị thông tin cho người dùng, một dạng phổ biến và dễ sử dụng nhất là mã 7 đoạn. Đèn LED hiển thị giá trị thập phân hoặc một số ký tự nhất định. Có hai loại LED 7 đoạn là Anode chung và Cathode chung với mức luận lý bù nhau. IC giải mã từ BCD sang 7 đoạn phải được sử dụng phù hợp với loại đèn LED.



Hình 5.9 Kết nối LED A chung với IC giải mã

Mạch mã hóa thập phân sang BCD

Một chip giải mã thông thường có 9 đầu vào và 4 lối ra BCD. Chip giải mã và bảng sự thật minh họa trong Hình 5.10



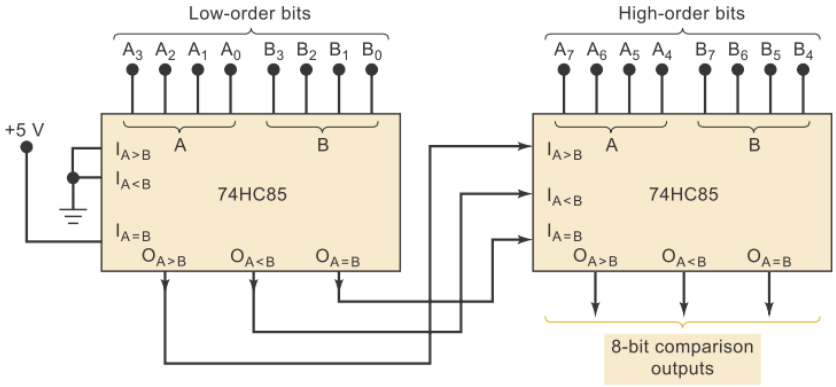
$\bar{A}_1$	$\bar{A}_2$	$\bar{A}_3$	$\bar{A}_4$	$\bar{A}_5$	$\bar{A}_6$	$\bar{A}_7$	$\bar{A}_8$	$\bar{A}_9$	$\bar{O}_3$	$\bar{O}_2$	$\bar{O}_1$	$\bar{O}_0$
1	1	1	1	1	1	1	1	1	1	1	1	1
X	X	X	X	X	X	X	X	0	0	1	1	0
X	X	X	X	X	X	X	0	1	0	1	1	1
X	X	X	X	X	X	0	1	1	1	0	0	0
X	X	X	X	X	0	1	1	1	1	0	0	1
X	X	X	X	0	1	1	1	1	1	0	1	0
X	X	X	0	1	1	1	1	1	1	0	1	1
X	X	0	1	1	1	1	1	1	1	1	0	0
X	0	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	0

X = either 0 or 1

Hình 5.10 IC mã hóa thông dụng

Mạch so sánh độ lớn

Cho phép so sánh hai giá trị nhị phân và cung cấp kết quả so sánh tại lối ra. Minh họa trên Hình 5.11 là mạch so sánh 8 bit. Người ta thường kết hợp nhiều chip để có thể so sánh số nhị phân giá trị lớn.



Hình 5.11 Mạch so sánh số nhị phân

6

Các Họ Vi Mạch Số

6.1 Giới thiệu

Các mạch điện tử tích hợp được sản xuất bởi nhiều cấu hình mạch và công nghệ khác nhau. Mỗi cách tiếp cận được gọi là một họ. Nhờ những tiến bộ trong công nghệ sản xuất, các hệ thống điện tử được thu gọn và có độ tin cậy cao. Những họ vi mạch thông dụng gồm có RTL, DTL, TTL, MOS, ECL và Bi-CMOS.

Tuy nhiên, những vi mạch số thường được sử dụng để chế tạo các mạch công suất thấp, hay còn gọi là mạch xử lý thông tin. Nó khó có khả năng chịu được điều kiện làm việc với điện áp và dòng điện cao do tản nhiệt kém.

6.2 Các họ vi mạch

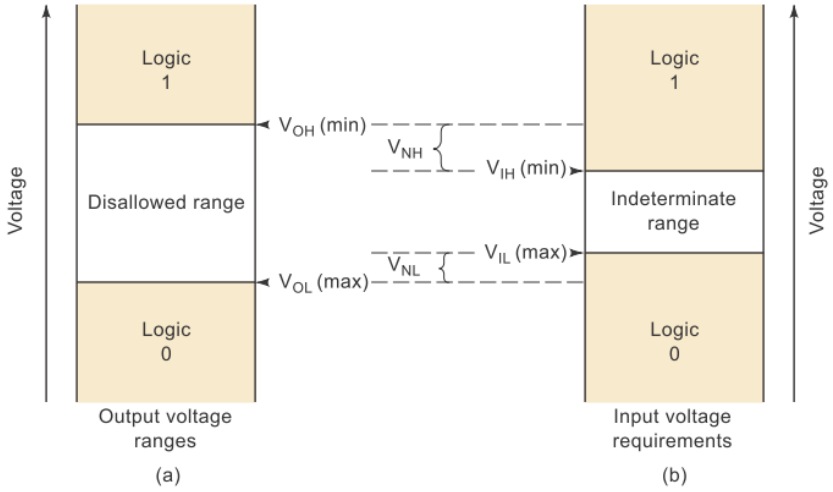
Một số thuật ngữ

Fan Out được định nghĩa là số lối vào tối đa mà một lối ra có thể cấp tín hiệu chính xác. Giả sử một cổng luận lý có fan-out bằng 10 nghĩa là nó có thể cấp tín hiệu cho tối đa 10 lối vào của các cổng cửa gắn sau nó. Nếu vượt quá số lượng trên, tín hiệu không còn chính xác nữa.

Độ trễ đường truyền (Propagation Delay) phát sinh do tín hiệu cần thời gian để truyền qua mạch điện. Hai loại độ trễ cơ bản là độ trễ chuyển đổi trạng thái từ thấp lên cao và từ cao xuống thấp.

Nguồn là yếu tố cần thiết để cho các mạch hoạt động. Công suất tiêu tán bằng điện áp cấp nguồn nhân với dòng tiêu thụ trung bình: $P_{D_{avg}} = I_{CC_{avg}} \times V_{CC}$

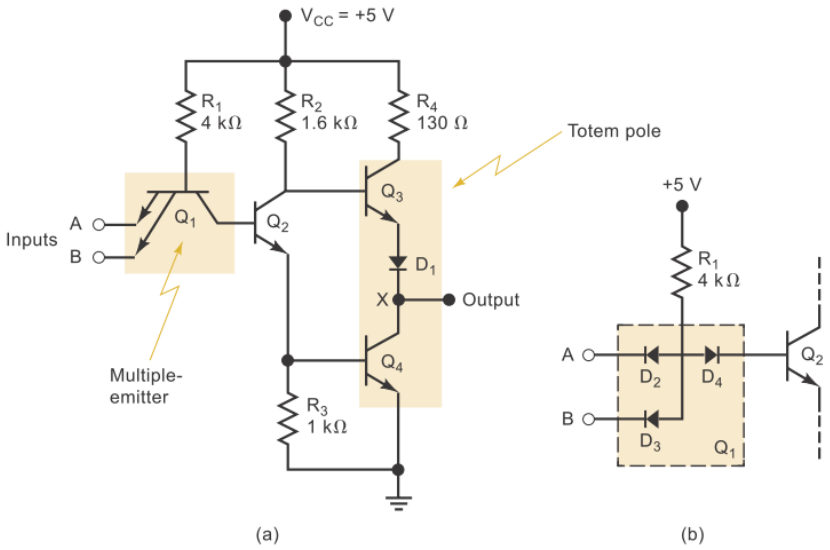
Kháng nhiễu (Noise Immunity) thể hiện khả năng chịu nhiễu của mạch mà không gây ra các thay đổi ở trạng thái lối ra. Nhiễu có thể được sinh ra do điện áp thay đổi bởi nhiều yếu tố liên quan khi hoạt động. Minh họa ngưỡng chịu nhiễu trong Hình 6.1.



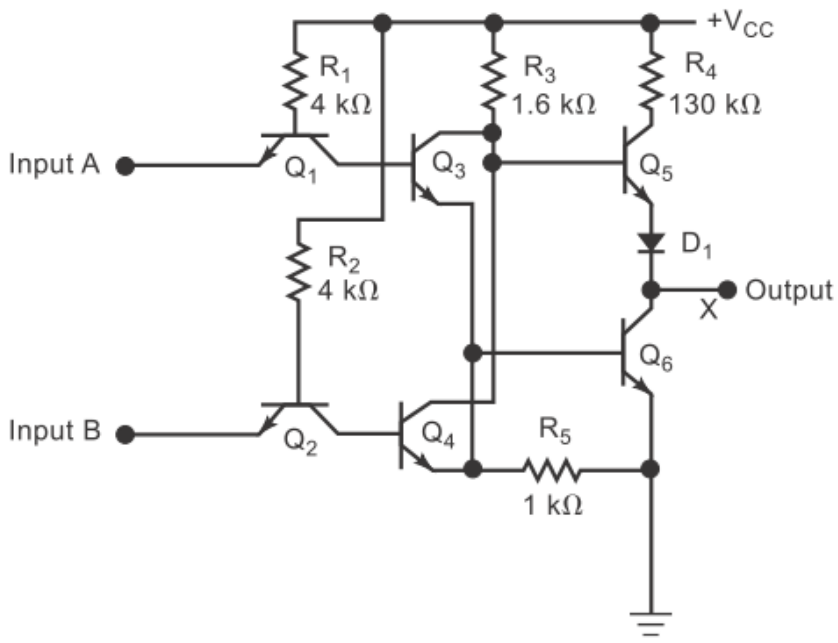
Hình 6.1 Nhiễu điện áp và ngưỡng trạng thái

Họ TTL

Rất nhiều họ vi mạch vẫn áp dụng chuẩn TTL (Transistor Transistor Logic). Nó cho phép kết nối nhiều thiết bị phức tạp của hệ thống điện tử số và cho phép dòng tải cao. Cổng NAND với công nghệ TTL được trình bày trong Hình 6.2.



Hình 6.2 Cổng NAND và mô hình diode tương tự cho Q1



Hình 6.3 Cổng NOR chuẩn TTL

Kết nối Q3 và Q4 như trên còn gọi là *totem pole*. Nó hạn chế dòng tiêu thụ ở trạng thái lối ra thấp đồng thời có tốc độ thay đổi trạng thái cao. Tuy nhiên, tốc độ chuyển trạng thái cao - thấp không đồng đều.

Các mạch TTL có cấu trúc khá giống nhau. Cổng NAND và AND sử dụng nhiều transistor lối ra E hoặc lối vào đa mối mối diode. Cổng NOR và OR sử dụng các transistor tách biệt tại lối vào. Trong cả hai trường hợp, trạng thái lối vào cao sẽ ngắt transistor và dòng rò nhỏ. Ngược lại, lối vào thấp sẽ mở transistor và dòng thu lớn. Trên thực tế, không phải tất cả các mạch TTL sử dụng cấu hình totem-pole.

Một số đặc tính:

Mạch TTL sử dụng điện áp nguồn nuôi chuẩn V_{cc} 5V, dao động quanh khoảng 4,5 đến 5,5V. Cụ thể giá trị được trình bày trong tài liệu (datasheet) của từng linh kiện. Mỗi lối ra của linh kiện họ TTL có một giới hạn về dòng điện chảy ra hoặc vào tương ứng với mức trạng thái cao hay thấp. Ngưỡng giới hạn dòng điện là cơ sở để xác định fan-out của linh kiện. Nó thường được chú thích đầy đủ trong tài liệu của linh kiện.

Khi lối vào để hở, trạng thái được coi như ở mức cao. Trạng thái đầu vào cao khi bị để hở còn gọi là đầu vào trôi. Để tránh những trường hợp đầu vào trôi gây ảnh hưởng đến hoạt động của mạch, chúng phải được kết nối với mức điện thế cố định. Ví dụ, đầu vào không sử dụng của cổng NAND cần nối lên cao qua một điện trở, hay còn gọi là điện trở pull-up. Hoặc nối đầu vào không sử dụng với một đầu vào khác miễn sao đảm bảo trạng thái luận lý đúng với mạch đang sử dụng.

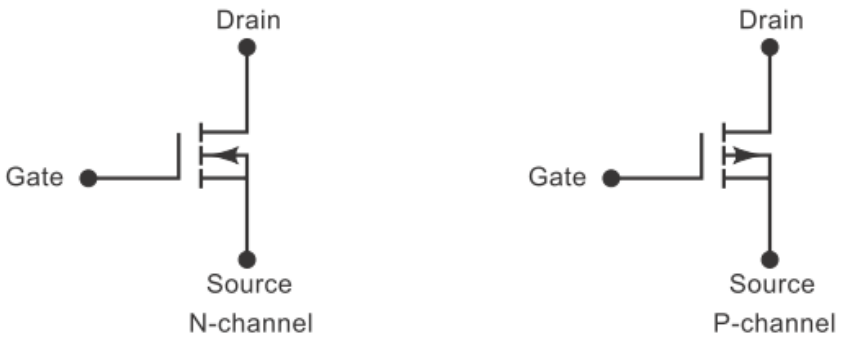
Đối với họ TTL, tại lối ra trong quá trình chuyển đổi trạng thái từ thấp lên cao sẽ phát sinh dòng điện xung trong thời gian rất ngắn. Để hạn chế tác động tiêu cực, một tụ điện nhỏ tần số cao cần được gắn tại chân nguồn của linh kiện. Nó còn gọi là tụ *decoupling* với giá trị khoảng $0,01$ đến $0,1uF$.

Chuẩn TTL còn có một số tiêu chuẩn con với tính năng về tốc độ và công suất khác nhau. Chuẩn Schottky TTL (họ 74S) sử dụng transistor kèm Schottky diode có tốc độ cao nhưng công suất tiêu tán lớn hơn do sử dụng điện trở nhỏ. Chuẩn Schottky công suất thấp (họ 74LS) sử dụng điện trở lớn nhằm giảm công suất nhưng có tốc độ thấp hơn họ TTL 74S. Ngoài ra, còn có họ TTL Schottky cải tiến và công suất thấp (họ 74AS và 74ALS). Gần đây nhất, họ TTL tốc độ cao được chế tạo với công nghệ mới cho phép vừa giảm công suất vừa tăng tốc độ chuyển trạng thái.

Họ MOS

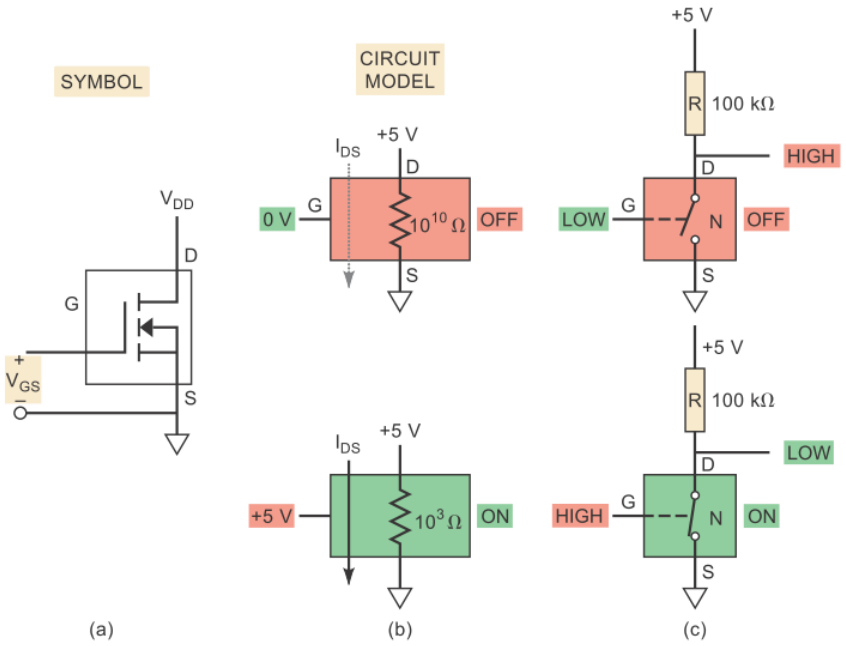
Công nghệ MOS (Metal Oxide Semiconductor) cho phép sản xuất linh kiện với giá thành rẻ, kích thước nhỏ và công suất tiêu thụ thấp. Đồng thời MOS cũng dễ chế tạo hơn công nghệ TTL. Tuy nhiên, linh kiện MOS chống nhiễu tĩnh điện kém, đòi hỏi thao tác bảo quản và sử dụng phức tạp hơn.

MOSFET có hai loại thông dụng là *enhancement* và *depletion*. MOSFET với hai kênh N và P được minh họa trong Hình 6.4.

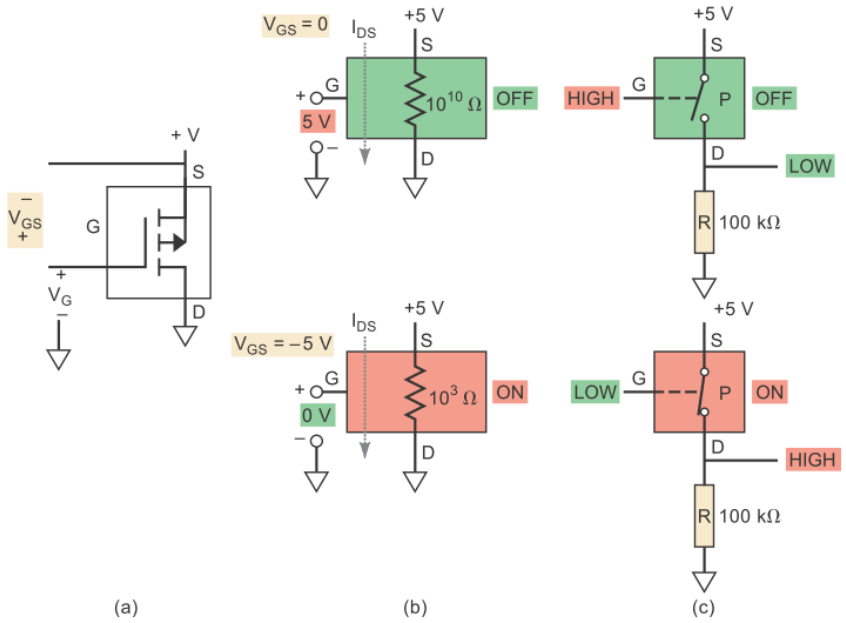


Hình 6.4 MOSFET kênh N và P loại tăng cường

Về cơ bản, MOSFET cũng là một transistor với nhiều chế độ hoạt động. Tuy nhiên với lý thuyết điện tử số, ta chỉ xem xét MOSFET hoạt động ở chế độ công tắc đóng/mở. Ví dụ N-MOS hoạt động trên Hình 6.5 và P-MOS trên Hình 6.6.



Hình 6.5 MOSFET kênh N và hoạt động của cổng NOT

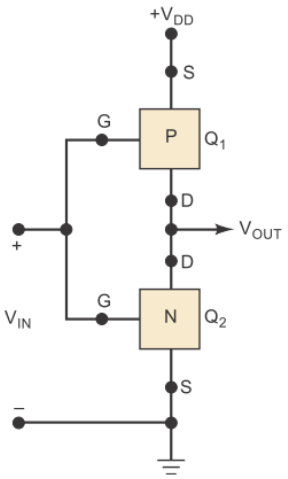


Hình 6.6 MOSFET kênh P và hoạt động của cổng NOT

Họ CMOS

Nếu chỉ dùng transistor MOS đơn, khi hoạt động dòng điện qua mạch gây tiêu tán công suất dưới dạng nhiệt. Bằng cách sử dụng cả N-MOS và P-MOS, mạch được chế tạo có tốc độ cao hơn và giảm công suất tiêu thụ. Công nghệ này còn gọi là CMOS (Complementary MOS).

Cổng NOT dùng CMOS được minh họa trong Hình 6.7. Mức trạng thái cho CMOS là $+V_{DD}$ cho ‘1’ và ‘0’ với $0V$.

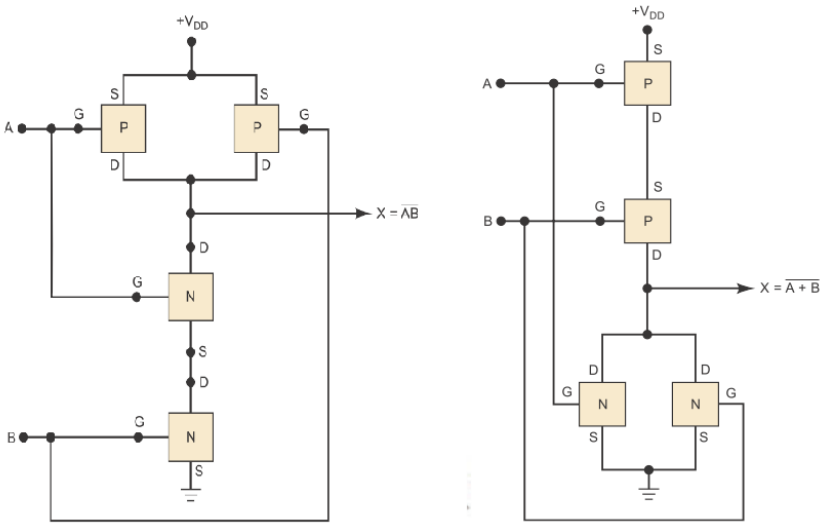


V_{IN}	Q_1	Q_2	V_{OUT}
$+V_{DD}$ (logic 1)	OFF $R_{OFF} = 10^{10}\ \Omega$	ON $R_{ON} = 1\ k\Omega$	≃ 0 V
0 V (logic 0)	ON $R_{ON} = 1\ k\Omega$	OFF $R_{OFF} = 10^{10}\ \Omega$	≃ $+V_{DD}$

$V_{OUT} = \overline{V_{IN}}$

Hình 6.7 Cổng NOT dùng công nghệ CMOS

Dựa trên nguyên tắc tương tự, cổng NAND và cổng NOR (hình 6.8) cũng được chế tạo theo công nghệ CMOS.



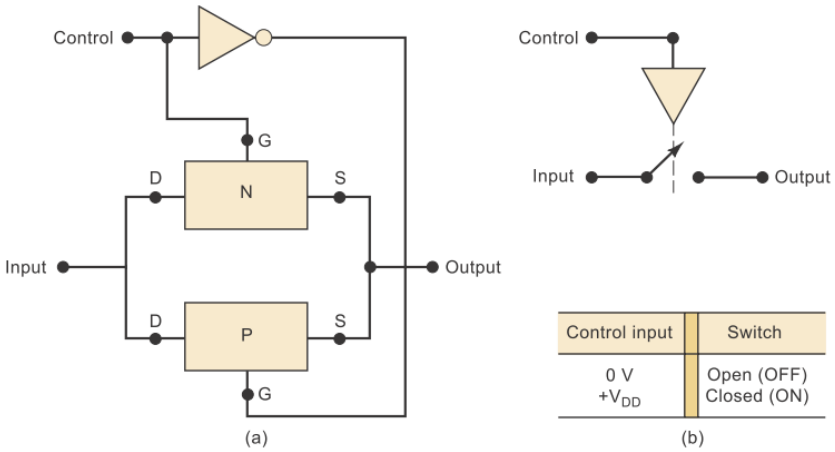
Hình 6.8 Cổng NAND và NOR

Cổng truyền (transmission gate) là một cổng đặc biệt chỉ có thể thực hiện bởi CMOS. Nó cho phép tín hiệu truyền theo hai chiều khi kích hoạt. Về nguyên tắc, nó giống như một công tắc cơ khí nhưng có điện trở tầm 200 Ω.

Một số đặc tính họ MOS:

Họ 4000/14000 có tuổi đời lâu nhất. Linh kiện họ này có công suất tiêu tán thấp, hoạt động ở vùng điện áp dải rộng (từ 3 đến 15V). Tuy nhiên chúng có tốc độ chậm, dòng tải thấp và ít tương thích với các vi mạch thế hệ mới.

Họ 74HC/HCT có tốc độ rất cao so với họ 74LS. Chúng có chân ra tương thích với họ TTL nhưng chỉ có 74HCT là tương thích mức điện áp. Họ 74AHC/AHCT là cải tiến của họ HC với



Hình 6.9 Cổng truyền CMOS

tốc độ cao hơn, công suất thấp. Họ 74AC/ACT có tính năng tương đương họ TTL nhưng khác cấu hình chân ra và không tương thích mức điện áp.

Họ **BiCMOS** là sự kết hợp của TTL và CMOS. Nó có tốc độ cao của TTL, công suất thấp của CMOS. Linh kiện BiCMOS chỉ được áp dụng cho những khối chức năng đặc biệt. Họ 74BCT/74ABT là những biến thể của BiCMOS với tính năng ưu việt của tốc độ cao, công suất thấp.

Vì CMOS tiêu thụ dòng rất ít nên ngưỡng trạng thái của CMOS khác với TTL. Đồng thời, công suất tiêu tán cũng ở mức thấp. Điện trở đầu vào của CMOS rất cao nên fan-out của CMOS về lý thuyết lớn hơn so với TTL. Yếu tố thay đổi fan-out là do tần số hoạt động của mạch.

Khác với TTL, lối vào chân CMOS không sử dụng phải luôn luôn kết nối đến mức điện áp cố định hoặc nối đến chân khác.

Tất cả các thiết bị điện tử đều nhạy cảm đối với tĩnh điện ở các mức độ khác nhau. Họ MOS rất dễ bị hư hỏng do tĩnh điện (ESD) và cần chú ý một số điểm sau:

- Tiếp đất cho tất cả các thiết bị nhằm ngăn tĩnh điện tích tụ và gây hư hại mạch.
- Tiếp đất cho người dùng với vòng đeo tay chuyên dụng.
- Bảo quản IC trong giấy nhôm hoặc túi dẫn điện để tránh điện áp chênh lệch giữa các chân IC.
- Hạn chế tiếp xúc với chân IC và gắn IC vào mạch ngay sau khi tháo khỏi túi bảo vệ.
- Không thể chân IC hở để tránh tích tụ tĩnh điện.

Trong quá trình sản xuất CMOS, các mối nối N-P hình thành transistor kí sinh. Chúng có thể bị kích hoạt (latch-up) và phá hủy IC. Hiện tượng này thường xảy ra do xung điện áp cao hoặc nhiễu. Để hạn chế latch-up, cần thiết phải gắn diode và tụ lọc nguồn.

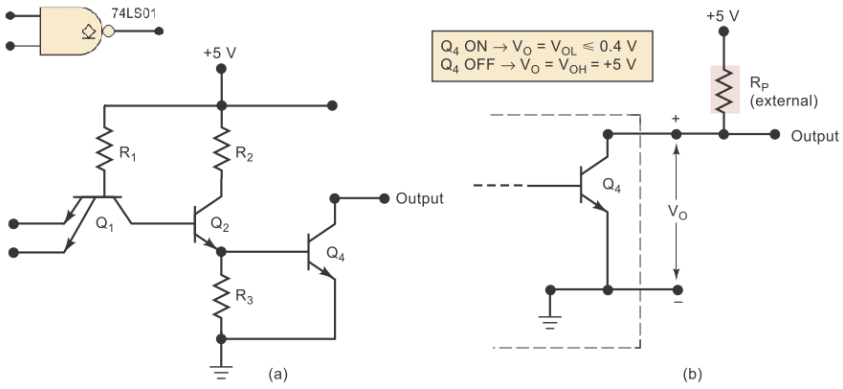
Công nghệ điện áp thấp ra đời do đòi hỏi tăng mật độ linh kiện. Ngưỡng điện áp phải giảm xuống để tránh hiện tượng đánh thủng khi lớp cách điện giữa các linh kiện ngày càng mỏng đi. Một số họ linh kiện mới hoạt động tại 2,5V hoặc thậm chí 1,8V.

Một số họ CMOS điện áp thấp như 74LVC, 74ALVC, 74LV, 74AVC, 74AUC, 74AUP, 74CBT, 74LVT, 74ALVT,. . .

Đặc điểm lối ra

Trong quá trình hoạt động, một số linh kiện dùng chung đường dây để truyền tín hiệu. Điều này đồng nghĩa với việc các lối ra của chúng nối với nhau và tín hiệu trên đường dây trở thành không xác định. Như vậy, lối ra linh kiện họ CMOS không bao giờ được nối với nhau. Tương tự, lối ra của họ TTL cũng không được nối với nhau.

Cực thu/cực máng hở: Đối với họ TTL, điện trở kéo lên (pull-up) được loại bỏ để tạo ra cực thu hở (open collector). Đối với CMOS gọi là cực máng hở (open drain). Vì không có điện trở nội, khi hoạt động với linh kiện cực thu hở phải treo trở ngoài lên cao, giá trị trung bình khoảng 10 kΩ. Minh họa cực thu hở trên Hình 6.10.



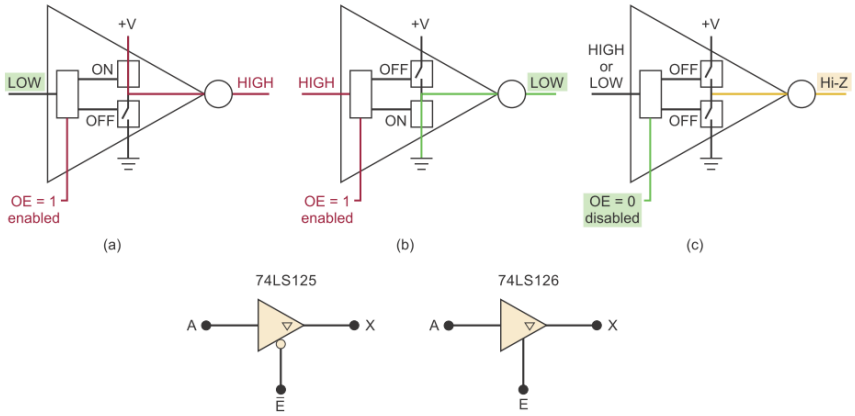
Hình 6.10 TTL cực thu hở và ký hiệu

Nếu các lối ra nối chung với nhau và treo trở cao, nó hoạt động giống như cổng AND với lối ra ở trạng thái cao thường xuyên. Cách nối như vậy chỉ được thực hiện với TTL cực thu hở và CMOS cực máng hở.

Xu hướng sử dụng linh kiện cực thu/cực máng hở đã không còn phổ biến. Hiện nay, nó thường được dùng để làm bộ đệm lối ra cho phép hoạt động với dòng tải và điện áp lớn.

Lối ra ba trạng thái bao gồm cao, thấp và trở kháng cao (Hi-Z). Trạng thái Hi-Z xảy ra khi cả hai điện trở treo (pull-up & pull-down) bị vô hiệu hóa. Lối ra có trở kháng cao đối với cả mức thế cao và đất.

Linh kiện lối ra ba trạng thái (tristate) có một lối vào điều khiển (enable) với nhãn *E* hoặc *OE*, minh họa trên Hình 6.11. Lối ra của các IC ba trạng thái có thể được nối với nhau. Tuy nhiên, mỗi lần chỉ có thể kích hoạt một lối ra nếu không trạng thái kết hợp của nhiều lối ra sẽ không xác định.



Hình 6.11 Cổng NOT với lối ra ba trạng thái

Bộ đệm tristate được dùng để điều khiển việc truyền tín hiệu từ lối vào đến lối ra. Nó thường được dùng trong mạch nơi các đường tín hiệu được nối chung với nhau, hay gọi là *bus*. Mỗi thời điểm, bộ đệm tristate sẽ cho phép một tín hiệu chạy trên bus. Ngoài ra còn có IC tristate, chúng có thể được nối với đầu ra của các IC khác trên cùng một đường dây (bus).

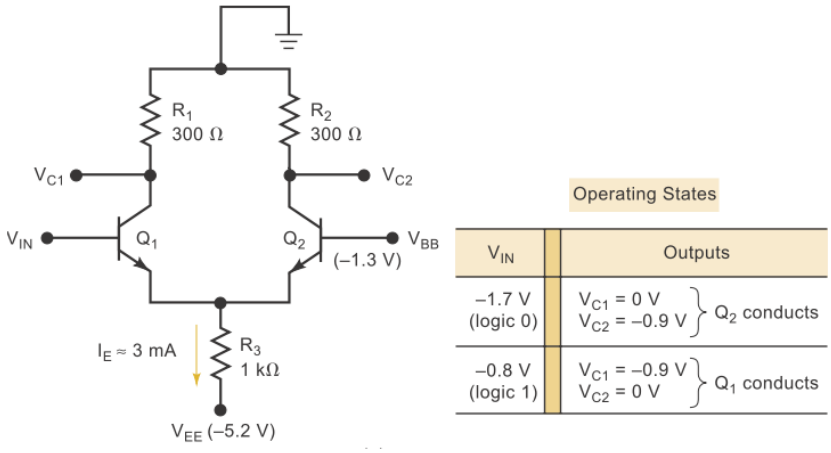
Họ ECL

Họ TTL sử dụng transistor ở chế độ bão hòa nên tốc độ phụ thuộc vào thời gian nạp của transistor ở chế độ bão hòa. Để tăng tốc độ do không đặt transistor vào chế độ bão hòa, họ ECL (Emitter-coupled Logic) hoạt động dựa trên nguyên tắc đóng mở dòng điện của các transistor. Do vậy nó còn có tên gọi CML (Current-mode Logic).

Mạch ECL cơ bản là một mạch khuyếch đại vi sai như Hình 6.12. Điện áp V_{EE} sinh ra dòng I_E ổn định cho phép dòng điện qua Q_1 hoặc Q_2 phụ thuộc điện áp V_{IN}.

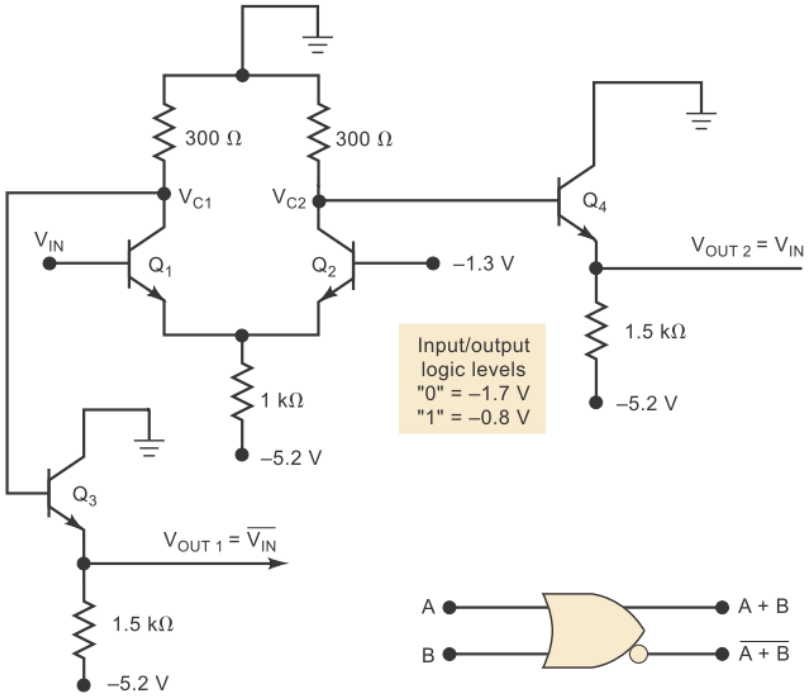
Tốc độ chuyển trạng thái cao nhất mà họ ECL có thể đạt được là 500ps và mạch hoạt động có xung nhịp lên đến 1,4GHz. Một số đặc tính của họ ECL hiện nay:

- Transistor không rơi vào bão hòa nên tốc độ chuyển trạng thái cao, vượt qua TTL và CMOS.
- Mức trạng thái cao ‘1’ là -0,8V và thấp ‘0’ là -1,7V.
- Tuy nhiên, ECL có ngưỡng chênh lệch điện áp thấp. Mức tối thiểu 150mV khiến nó rất khó áp dụng trong môi trường công nghiệp.



Hình 6.12 Mạch ECL cơ bản

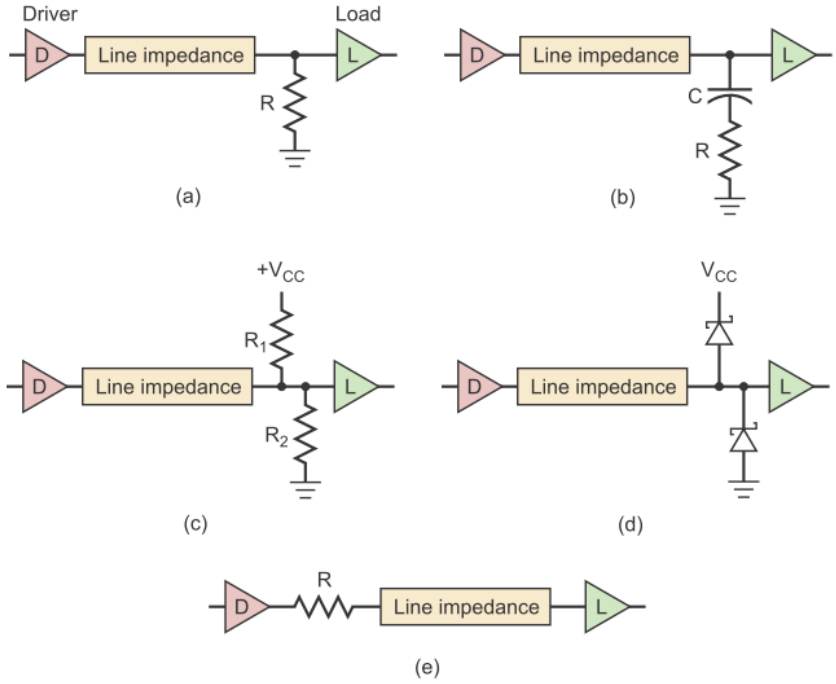
- Cổng cửa ECL thường kèm lối ra đảo do vậy không cần sử dụng thêm cổng đảo (hình 6.13). Đồng thời nó có dòng tải cao nên thuận lợi cho việc truyền tín hiệu qua đường cáp.
- Fan-out trung bình là 25.
- Công suất tiêu tán khoảng 25mW, cao hơn họ 74AS.
- Dòng tiêu thụ khi ECL hoạt động là ổn định bất kể khi chuyển trạng thái. Đây là lợi thế rất lớn so với TTL hoặc CMOS khi không phát sinh xung nhiễu.



Hình 6.13 Lối ra kèm mạch giữ trạng thái và biểu tượng cổng NOR/OR

6.3 Giao tiếp giữa các họ linh kiện

Nhiều hệ thống số có thể dùng chung đường bus tốc độ cao, đường truyền này có thể rất dài so với kích thước linh kiện. Do đặc tính của đường truyền hoạt động như điện dung ký sinh, từ dung và điện trở kí sinh sẽ khiến tín hiệu thay đổi (AC) phát sinh những hiệu ứng không mong muốn lên tín hiệu gốc. Các kỹ thuật chống tín hiệu nhiễu đường truyền được trình bày trong Hình 6.14.



Hình 6.14 Kỹ thuật kết nối đường truyền

Tuy nhiên các kỹ thuật này đều có điểm yếu. Do vậy các nhà sản xuất phải thường xuyên nghiên cứu và cải tiến để có thể hạn chế những khuyết điểm trên.

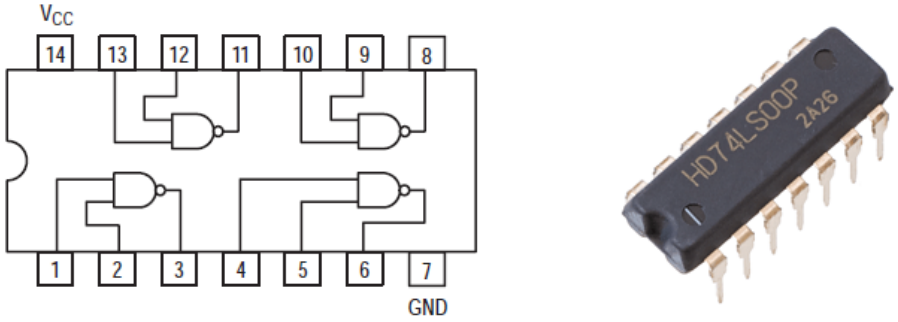
Giao tiếp giữa các họ IC cũng có đặc trưng riêng. Một ví dụ điển hình là TTL và CMOS, ngưỡng điện thế của hai họ khác xa nhau. Do vậy, giữa các họ IC phải có phương thức giao tiếp riêng. Phương pháp đơn giản nhất là sử dụng trở treo cao hoặc IC chuyển mức điện áp để đảm bảo ngưỡng điện áp hoạt động của các IC.

6.4 Một số loại IC thông dụng

Có hàng nghìn loại IC khác nhau được sử dụng trong nhiều loại mạch khác nhau. Trong phần này, chúng ta chỉ kể tên một số loại IC thường được sử dụng trong học tập và thực hành điện tử số. Những loại IC này bao gồm các cổng cửa cơ bản, bộ định thời, phát xung, mạch đếm và giải mã.

Cổng cửa cơ bản

NAND IC thông dụng nhất trong họ TTL là 74x00. Với x thể hiện họ của linh kiện, ví dụ họ LS thì viết 74LS00. Để đọc thứ tự chân, chúng ta quan sát vỏ IC để tìm điểm đánh dấu, có thể là góc khuyết hay chấm mực. Thông thường, chân có đánh dấu là chân số 1; hoặc bên trái của góc khuyết là chân số 1. Từ vị trí chân số 1 chúng ta sẽ đếm tuần tự để tìm các chân còn lại. Vị trí chân cấp nguồn thì phụ thuộc quy định của nhà sản xuất, ví dụ như trên Hình 6.15.



Hình 6.15 Cổng NAND thông dụng theo công nghệ xuyên lỗ

Đối với họ IC kiểu CMOS, các IC thường được ghi tên bắt đầu là 40xx. Cổng NAND CMOS có số chân và vị trí tương tự 74LS00 là 4011. Trên thực tế, người ta chế tạo rất nhiều phiên bản khác nhau của cổng NAND.

Chú ý: Cách gọi IC kiểu CMOS có thể gây nhầm lẫn vì một số IC sản xuất theo công nghệ CMOS vẫn bắt đầu tên gọi với 74. Sự khác biệt nằm ở mức điện áp sử dụng và cách thức giao tiếp với các loại IC khác. Do vậy chúng ta cần tìm hiểu kỹ về tính năng của linh kiện trước khi sử dụng. Các thông số này được thể hiện trong tài liệu đi kèm (datasheet).

AND IC họ TTL thông dụng nhất là 74x08. CMOS là 4081

OR IC họ TTL thông dụng nhất là 74x32. CMOS là 4071

NOT IC họ TTL thông dụng nhất là 74x04. CMOS là 4069

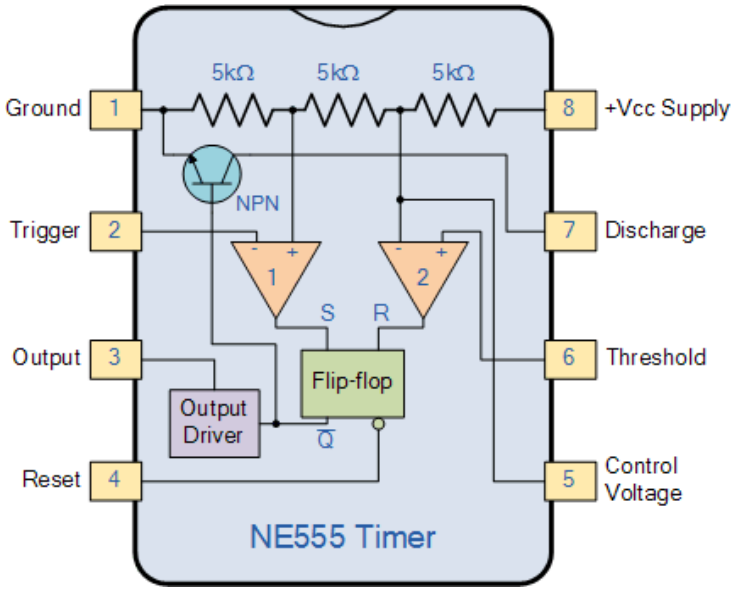
NOR IC họ TTL thông dụng nhất là 74x02. CMOS là 4001

XOR IC họ TTL thông dụng nhất là 74x86. CMOS là 4030

XNOR IC họ TTL thông dụng nhất là 74x266. CMOS là 4077

IC phát xung

Các mạch phát xung có thể được chế tạo từ các cổng cửa cơ bản. Tuy nhiên, những mạch này có độ chính xác không cao, khó kiểm soát. Một số loại IC phát xung thông dụng có thể được kể đến như họ 555, 74x121, 74x123, 4098B.



Hình 6.16 IC định thời 555

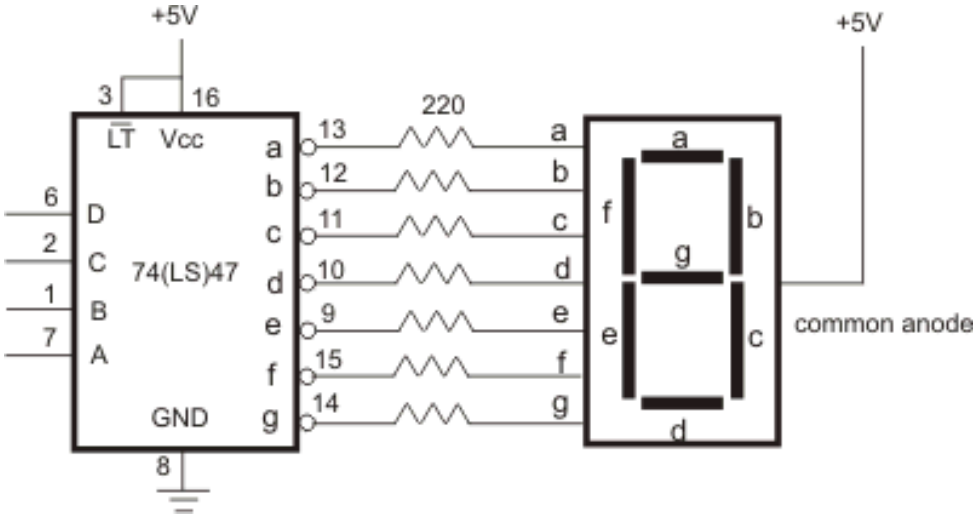
IC đếm

Mạch đếm có thể được thực hiện đơn giản chỉ bằng một IC. Mạch đếm nhị phân 4bit phổ biến là 74x393. Mạch đếm thập phân 4bit thường được sử dụng nhiều là 74x90, 74x390.

IC giải mã

Để chuyển đổi giữa các hệ đếm, người ta cũng chế tạo các loại IC giải mã. Những IC này giúp giải quyết những tác vụ đơn giản một cách nhanh chóng, không cần thiết kế lại mạch hoặc sử dụng linh kiện lập trình. Điển hình trong việc giải mã là IC 74x47, nó giúp giải mã BCD sang đèn LED 7 đoạn như minh họa trên Hình 6.17. IC 74x48 cũng có chức năng tương tự nhưng được sử dụng với loại LED C chung. Điểm khác biệt giữa hai IC này là trạng thái lối ra, thể hiện bởi dấu ‘o’ như trên hình minh họa.

Để giải mã từ BCD sang thập phân, IC 74x42 là thông dụng nhất. Hoặc 74x44 dùng để giải mã từ Gray sang thập phân. Ngoài ra, nhóm IC phân kênh trộn kênh cũng có thể được gom vào nhóm IC giải mã với tên gọi đa hợp và giải đa hợp. Trong 74x138 hoặc 74x154 là các IC phổ biến trên thị trường.



Hình 6.17 IC giải mã BCD sang LED 7 đoạn

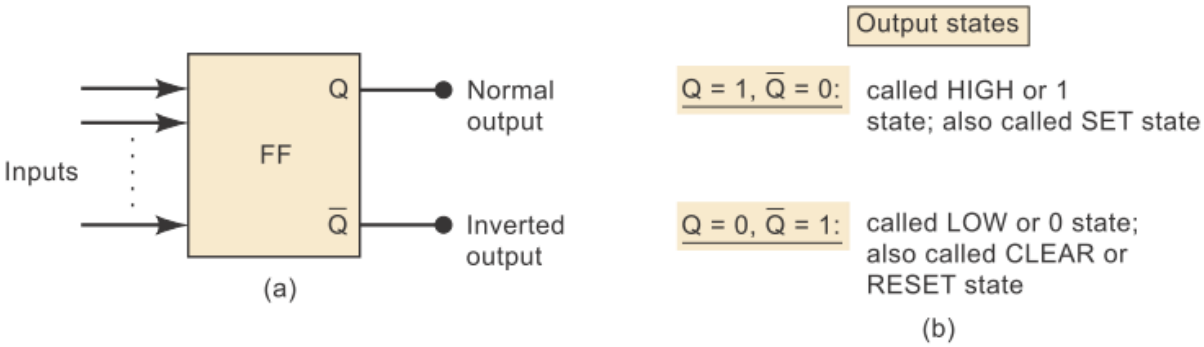
7

Linh Kiện Flip-Flop

7.1 Giới thiệu

Các mạch kết hợp có đầu ra chỉ phụ thuộc trạng thái hiện tại của đầu vào vì nó không có các thành phần nhớ. Hầu hết các hệ thống điện tử số là tổng hợp của mạch kết hợp và các thành phần nhớ. Như vậy, đầu ra của hệ thống sẽ phụ thuộc vào cả tín hiệu đầu vào và thông tin nhớ.

Thành phần nhớ cơ bản nhất là flip-flop (viết tắt FF). Nó được cấu thành bởi các cổng cửa luận lý. Bản thân các cổng cửa này không có khả năng lưu trữ thông tin do vậy chúng phải được kết hợp theo những cách nhất định. Hình 7.1 minh họa biểu tượng tổng quát một flip-flop.

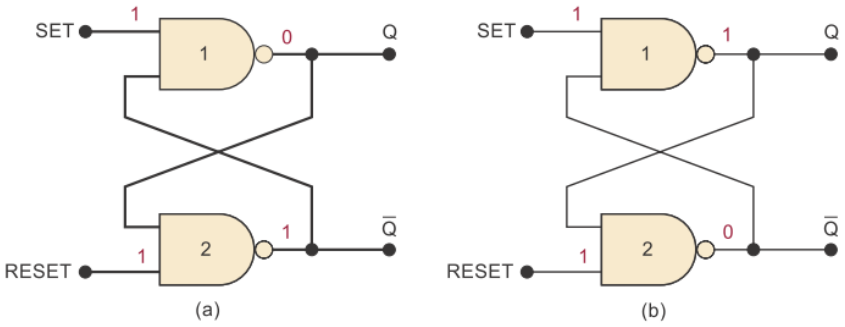


Hình 7.1 Flip-flop cơ bản và trạng thái lối ra

7.2 Bộ chốt cổng NAND và NOR

NAND latch

Flip-flop cơ bản nhất được tạo bởi hai cổng NAND hoặc NOR. Phiên bản cổng NAND tạo thành bộ chốt (latch) như trong Hình 7.2. Lối ra Q và $\bar{Q}$ luôn mang trạng thái đảo của nhau.

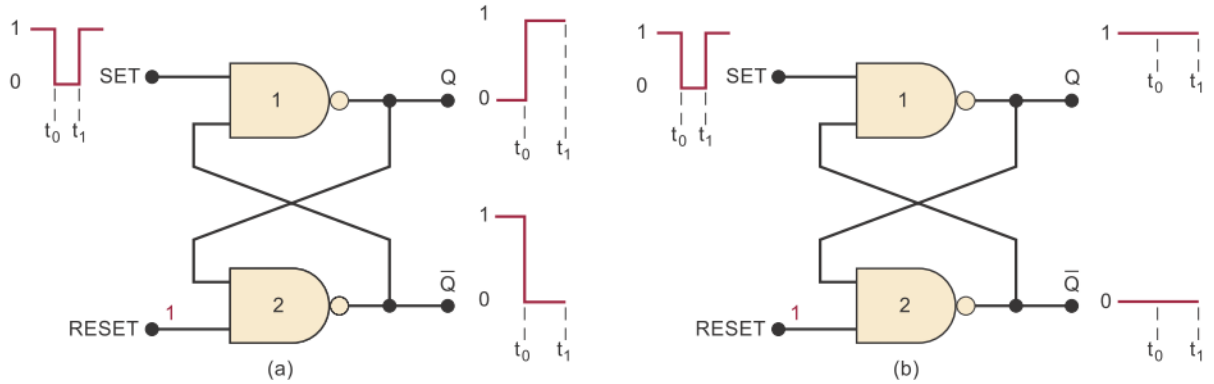


Set	Reset	Output
1	1	No change
0	1	Q = 1
1	0	Q = 0
0	0	Invalid*

*Produces Q = $\overline{Q}$ = 1.

Hình 7.2 Flip-flop chốt cơ bản

Để thay đổi trạng thái FF, xét trường hợp như Hình 7.3. Chân SET tạm thời hạ xuống thấp trong khi RESET vẫn giữ cao. Lối ra Q sẽ lên cao và $\bar{Q}$ xuống thấp. Trường hợp 2, khi Q = 1, SET xuống thấp nhưng không thay đổi trạng thái lối ra do $\bar{Q}$ = 0 từ trước. Như vậy, SET có xung thấp sẽ khiến bộ chốt có lối ra Q = 1.



Hình 7.3 Hoạt động của bộ chốt

Ngược lại, khi chân SET giữ cao và xung thấp đưa vào RESET. Khi Q = 0 thì lối ra không có thay đổi. Khi lối ra Q = 1, xung RESET thấp sẽ khiến $\bar{Q}$ lên cao và Q về thấp. Do vậy, xung thấp tại RESET sẽ xóa trạng thái FF. Chân RESET có thể còn được viết là CLEAR.

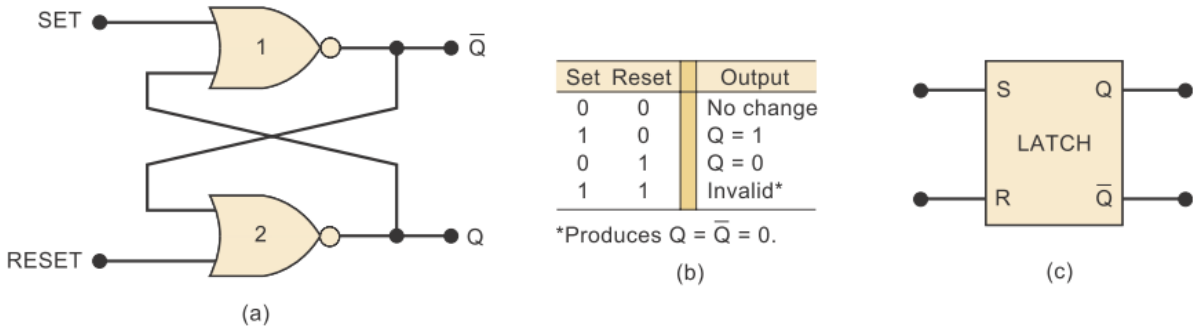
NOR latch

Bộ chốt cổng NOR cũng tương tự như dùng cổng NAND chỉ khác vị trí Q và $\bar{Q}$ đổi cho nhau như Hình 7.4. Hoạt động được tóm tắt như sau:

- Khi SET = RESET = 0. Trạng thái bình thường của FF với lối ra không đổi.
- Khi SET = 1, RESET = 0. Lối ra Q = 1 và giữ nguyên khi SET về 0.

- Khi SET = 0, RESET = 1. Xóa lối ra Q = 0 và giữ nguyên khi RESET về 0.
- Khi SET = 1, RESET = 1. Trạng thái lối ra không xác định do đó không nên sử dụng.

Hoạt động của chốt NOR giống như NAND chỉ khác là chân SET và RESET tác động mức cao thay vì thấp, và trạng thái bình thường có SET = RESET = 0.



Hình 7.4 Bộ chốt NOR

Khi mới cấp nguồn cho FF, trạng thái lối ra và lối vào sẽ rất khó xác định. Để đảm bảo FF hoạt động đúng, các chân SET và RESET phải được kích hoạt để FF về lại trạng thái ổn định.

7.3 Xung số

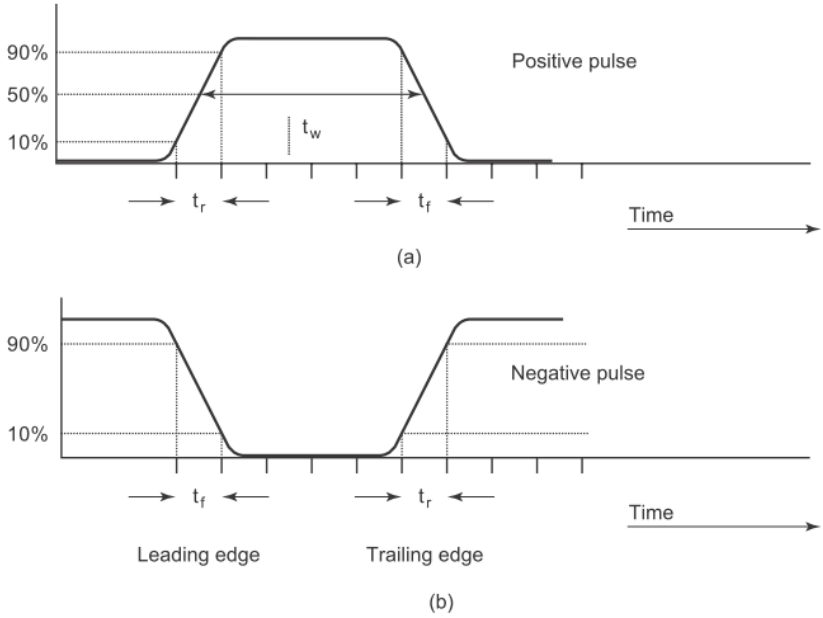
Trong quá trình hoạt động của FF-SR vừa nêu trên, tín hiệu đưa vào chuyển trạng thái từ mức không tác động sang tác động khiến linh kiện phản ứng. Sau đó, tín hiệu trở về trạng thái cũ. Những tín hiệu như vậy còn được gọi là xung (pulses). Xung khiến mạch hoạt động khi ở mức cao thì gọi là xung dương và ngược lại, là xung âm.

Khi chuyển mức từ thấp lên cao hoặc từ cao xuống thấp, xung có hai sườn và được gọi là sườn dương và sườn âm như minh họa trên Hình 7.5.

Các hệ thống số có thể hoạt động đồng bộ (synchronous) hoặc bất đồng bộ (asynchronous). Trong hệ thống đồng bộ, thời điểm lối ra thay đổi trạng thái có thể được xác định bởi tín hiệu đồng hồ (clock). Tín hiệu này cung cấp cho hầu như tất cả các thành phần của hệ thống. Chu kỳ của xung đồng hồ được đo giữa các sườn xung cùng loại kế tiếp nhau.

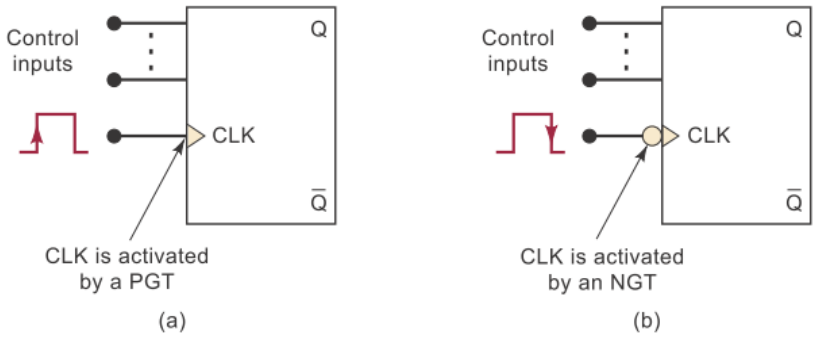
Flip-flop dùng xung đồng bộ

Các loại FF dùng xung đồng hồ có một lối vào dán nhãn CLK, CK hoặc CP. Hầu hết chúng tác động bằng sườn xung. Ngoài ra, các FF này còn có một hoặc nhiều lối vào điều khiển. Các lối vào này không có tác dụng cho đến khi xung CLK ở trạng thái tác động. Hình 7.6 minh họa loại



Hình 7.5 Xung dương và xung âm

sườn xung tác động. Như vậy, các chân điều khiển quyết định trạng thái lối ra khi có xung CLK tác động.



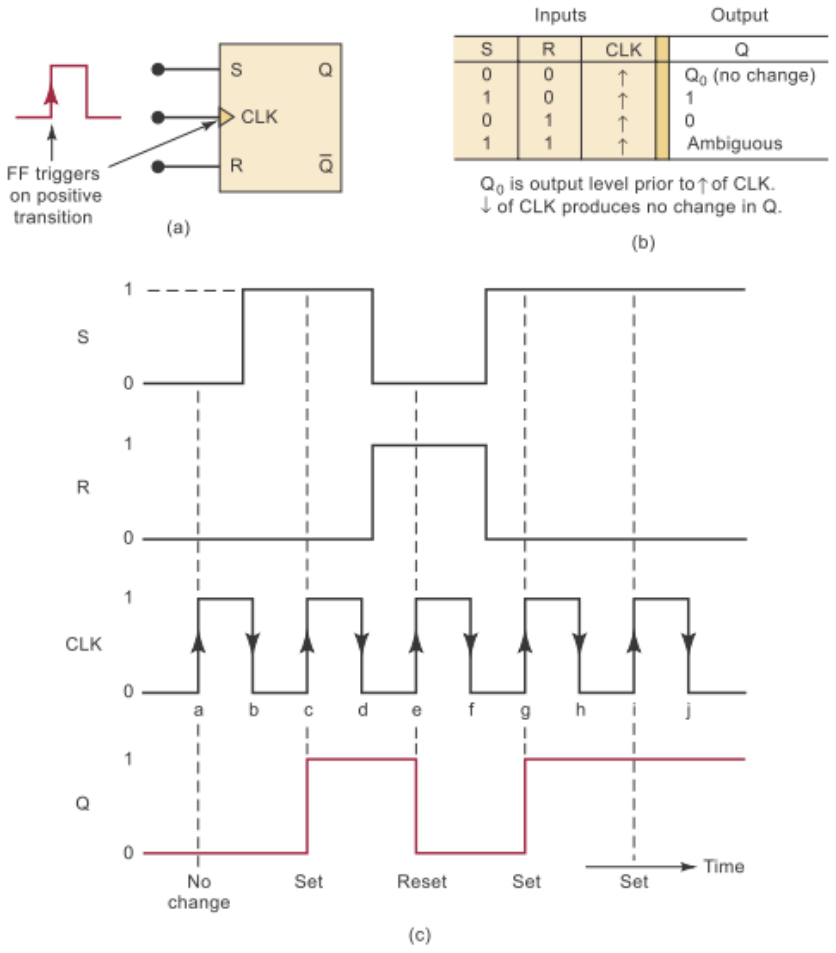
Hình 7.6 Sườn xung tác động của các loại FF

Để đảm bảo FF hoạt động chính xác, lối vào điều khiển phải có trạng thái ổn định trong một khoảng thời gian tối thiểu $t_S(min)$ trước khi xung đồng hồ tác động, và một khoảng tối thiểu $t_H(min)$ sau khi xung đồng hồ tác động.

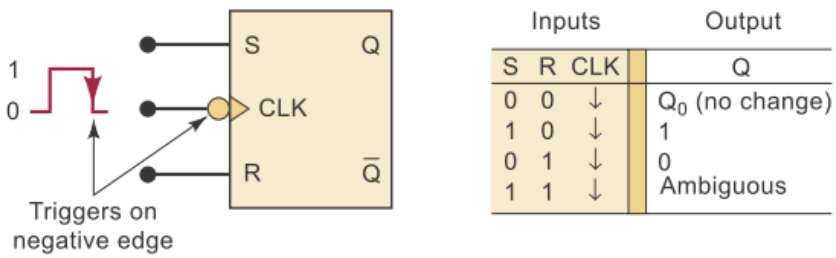
7.4 Một số loại Flip-flop

Flip-flop S-R

Hình 7.7 minh họa một FF S-R kích thích bằng sườn xung dương. FF chỉ thay đổi trạng thái khi xung đồng hồ chuyển từ 0 sang 1. Biểu tượng và bảng hoạt động của một FF kích thích bằng sườn xung âm được trình bày trong Hình 7.8



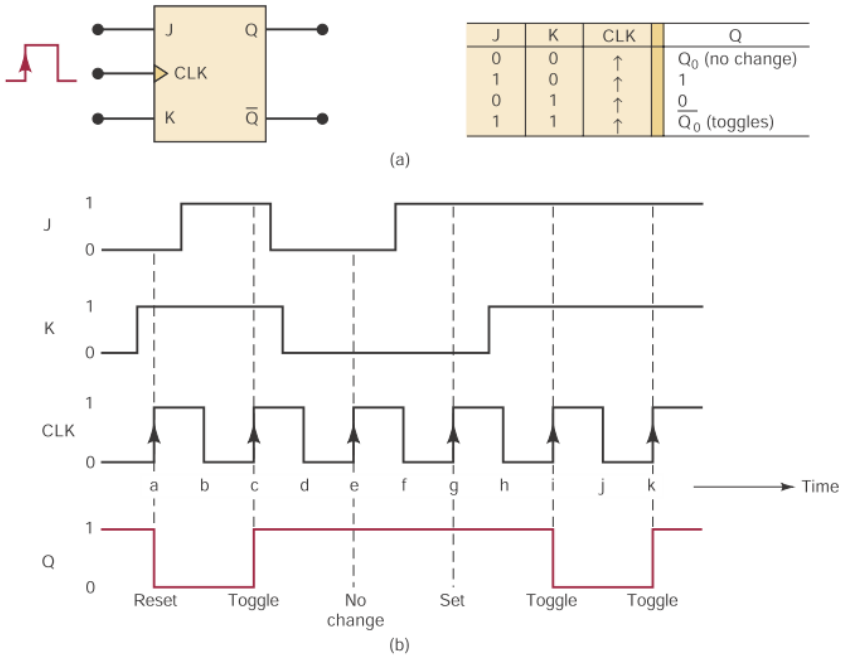
Hình 7.7 FF-SR đồng bộ và giản đồ xung



Hình 7.8 FF-SR đồng bộ kích thích sườn âm

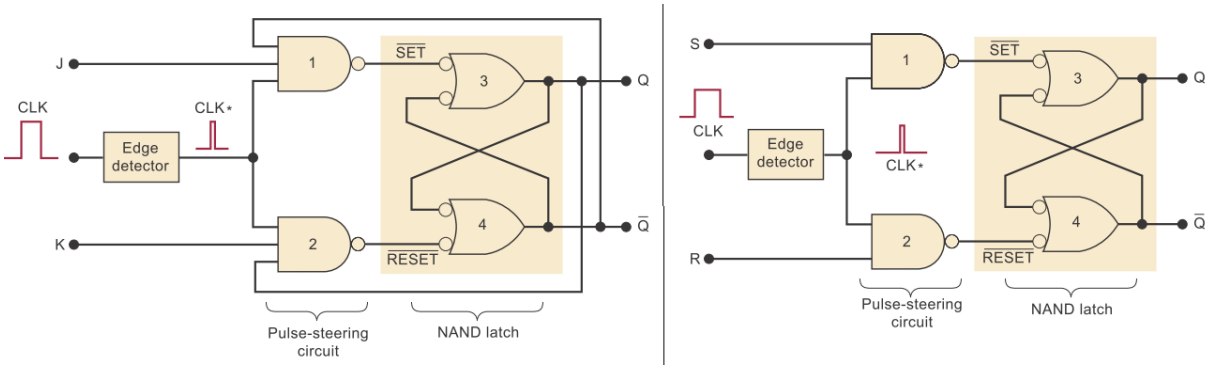
Flip-flop J-K

Hình 7.9 minh họa FFJK kích thích bằng sườn xung dương. Khác với FFSR, FFJK có thể hoạt động với điều kiện J = K = 1. Với mỗi xung kích thích, trạng thái lối ra Q sẽ bị đảo ngược của trạng thái trước đó.



Hình 7.9 Flip-Flop J-K

Tương tự như FFSR, FFJK cũng có loại tác động bằng sườn xung âm. Hình 7.10 minh họa cấu trúc đơn giản hóa của hai loại FF.

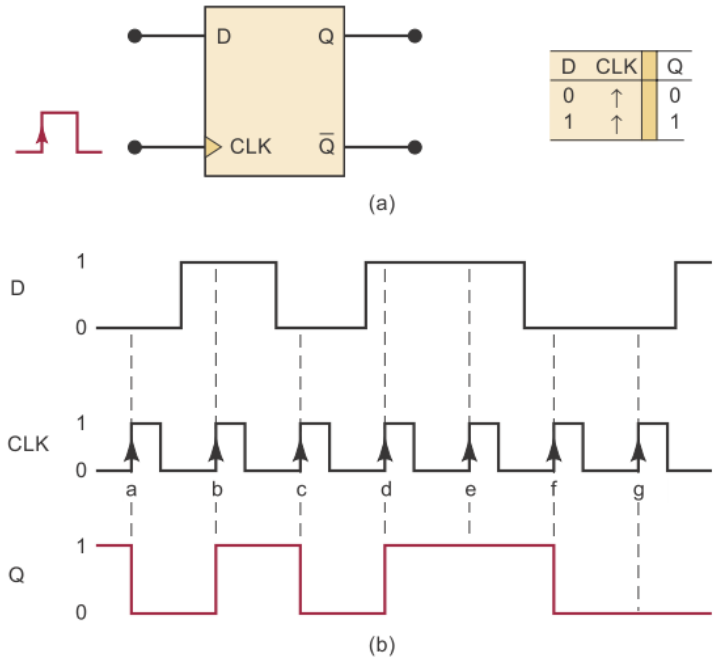


Hình 7.10 Bên trong FFJK và FFSR

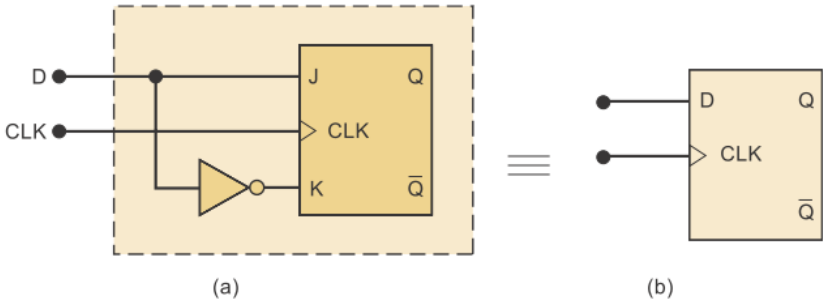
Flip-flop D

Hình 7.11 trình bày bảng chức năng và biểu tượng của FFD dùng xung dương. Không giống FFSR hay FFJK, FFD chỉ có một đầu điều khiển D đại diện cho dữ liệu. Trạng thái lối ra sẽ giống như lối vào điều khiển khi có sườn dương xung của tín hiệu CLK. FFD dùng sườn xung

âm cũng hoạt động tương tự. Có thể xây dựng FFD bằng FFSR hoặc FFJK. Hình 7.12 minh họa FFD thực hiện bằng FFJK.



Hình 7.11 Flip-flop D



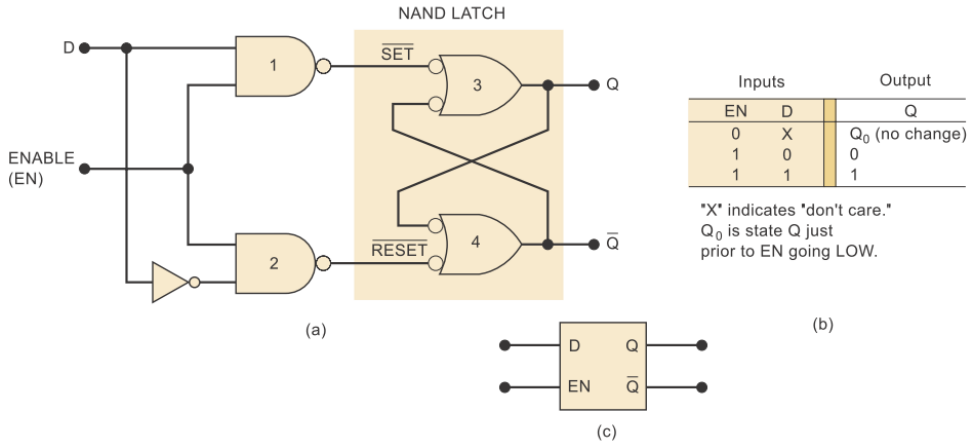
Hình 7.12 Cấu tạo cơ bản Flip-flop D

FFD được ứng dụng trong truyền dữ liệu song song. Các đầu ra của một mạch nhị phân chỉ nhận dữ liệu khi tín hiệu đồng bộ CLK tác động lên các FFD trước mỗi đường truyền.

Một biến thể của FFD là bộ chốt D (D latch) như trong Hình 7.13. Linh kiện này không sử dụng mạch dò sườn xung.

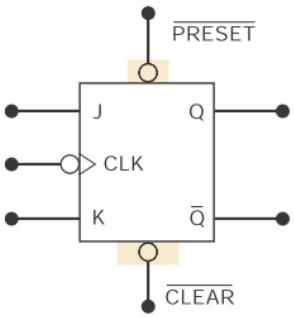
Lối vào bất đồng bộ

Các flip-flop SR, JK và D có các đầu vào điều khiển hay còn gọi là đầu vào đồng bộ vì tác động của chúng được đồng bộ hóa với đầu vào CLK. Ngoài ra, hầu hết các FF còn có một hoặc nhiều lối vào bất đồng bộ, độc lập với các đầu vào khác, còn gọi là đầu vào vượt cấp (override). Nó thay đổi trạng thái của FF bất cứ khi nào tác động.



Hình 7.13 Bộ chốt D

Hình 7.14 minh họa biểu tượng và bảng trạng thái của FFJK với lối vào bất đồng bộ. Các lối vào này tác động bằng mức trạng thái chứ không phải sườn xung.



J	K	Clk	PRE	CLR	Q
0	0	↓	1	1	Q (no change)
0	1	↓	1	1	0 (Synch reset)
1	0	↓	1	1	1 (Synch set)
1	1	↓	1	1	$\overline{Q}$ (Synch toggle)
x	x	x	1	1	Q (no change)
x	x	x	1	0	0 (asynch clear)
x	x	x	0	1	1 (asynch preset)
x	x	x	0	0	(Invalid)

Hình 7.14 FFJK với đầu vào bất đồng bộ

7.5 Tham số thời gian của Flip-flop

Luôn luôn tồn tại một khoảng thời gian giữa hai thời điểm thay đổi trạng thái của lối vào và lối ra trên FF. Khoảng thời gian này được gọi là độ trễ truyền qua (propagation delay). Dựa vào tham số này, các FF sẽ được tính toán tần số hoạt động tối đa. Một số FF có độ trễ khoảng *ns* và có thể hoạt động ở 20 MHz.

Tham số thứ hai là thời gian giữ trạng thái của xung, hay còn gọi là độ rộng xung. Tín hiệu CLK cũng như tín hiệu đầu vào bất đồng bộ phải đảm bảo độ rộng xung tối thiểu để FF có thể hoạt động như thiết kế.

Một số vấn đề khác có thể xảy ra khi kết nối nối tiếp nhiều FF dùng chung xung CLK. Trạng thái lối ra của những FF phía sau sẽ không xác định vì chúng chịu ảnh hưởng từ độ trễ lối ra của những FF phía trước đó.

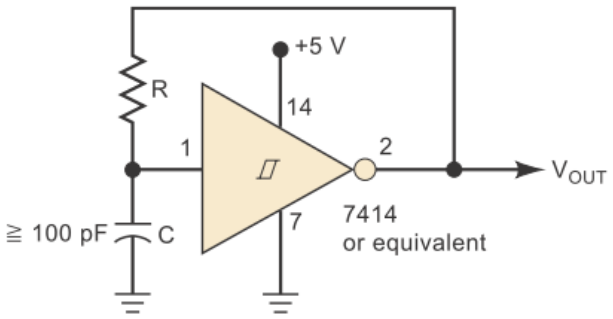
7.6 Mạch tạo xung

Để cung cấp xung nhịp cho các mạch điện tử hoặc FF, người ta thường sử dụng các mạch tạo xung hoặc còn gọi là mạch tạo dao động. Các loại tín hiệu lối ra có thể là sóng sin, sóng vuông hay tam giác. Hiện có rất nhiều loại mạch tạo xung khác nhau, từ mạch đơn lẻ đến IC phát xung chuyên nghiệp. Trong phần này, chúng ta chỉ làm quen với một số mạch tạo xung vuông cơ bản và thường được sử dụng.

Dao động đa hài

Mạch tạo dao động đa hài sẽ có lối ra không ổn định ở bất cứ trạng thái nào. Nó sẽ liên tục lật trạng thái cao/thấp và do vậy thường được sử dụng làm nguồn xung đồng hồ cung cấp cho các mạch đếm.

Loại mạch đầu tiên cần xem xét được xây dựng dựa trên cổng NOT Schmitt-trigger. Trên Hình 7.15, tín hiệu tại lối ra Vout có dạng xung vuông với tần số phụ thuộc vào giá trị R và C. Với mỗi loại IC, giá trị RC phải được lựa chọn phù hợp nếu không mạch sẽ không hoạt động.

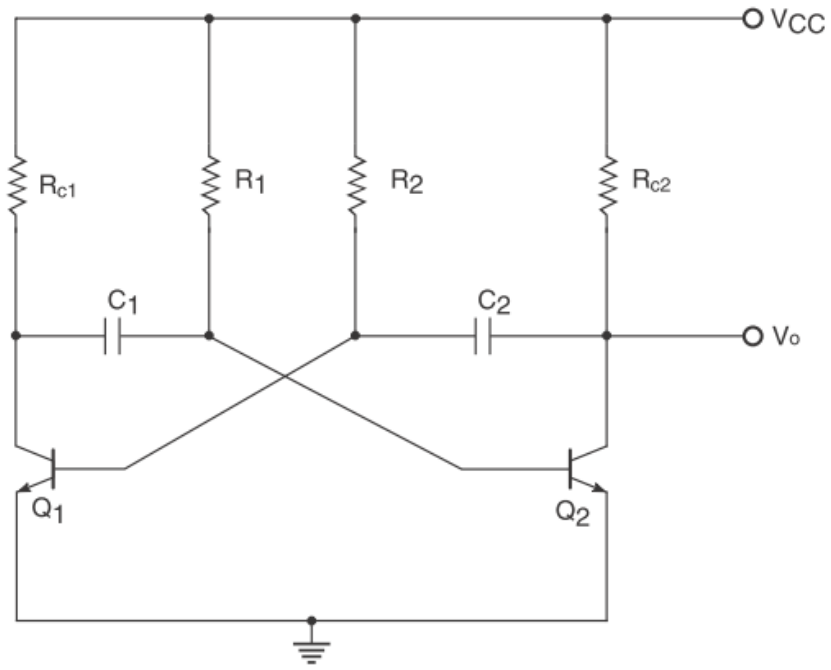


IC	Frequency	
7414	≈ 0.8/RC	(R ≦ 500 Ω)
74LS14	≈ 0.8/RC	(R ≦ 2 kΩ)
74HC14	≈ 1.2/RC	(R ≦ 10 MΩ)

Hình 7.15 Mạch dao động đa hài dùng cổng NOT

Loại mạch thứ hai có độ chính xác cao hơn ở tần số thấp và tính toán được. Nó sử dụng một cặp transistor và các RC tương ứng. Lối ra của mạch sẽ thay đổi trạng thái cao/thấp liên tục theo một chu kỳ nhất định. Hình 7.16 thể hiện một mạch dao động đa hài cơ bản. Thời gian lối ra duy trì trạng thái thấp hoặc cao phụ thuộc vào R và C. Nếu như giá trị $R_1C_1 = R_2C_2$ thì lối ra sẽ có xung vuông chuẩn.

Nguyên lý hoạt động của mạch như sau: giả sử transistor Q_2 dẫn trước, do vậy lối ra Vo ở mức thấp. Tụ C_2 sẽ được nạp qua điện trở R_2 từ nguồn Vcc. Khi chân tụ C_2 nối với cực nền (B) của Q_1 có mức điện áp vượt qua thế ngưỡng nó sẽ khiến Q_1 dẫn. Sụt áp đột ngột tại cực thu (C) của Q_1 thể hiện rõ trên cực nền (B) của Q_2 bởi vì tụ không thể nạp ngay lập tức. Chính vì vậy Q_2 bị cắt và lối ra nhảy lên trạng thái cao. Tụ C_1 được nạp qua R_1 trong khi Q_1 đang dẫn. Một khi



Hình 7.16 Mạch dao động đa hài dùng transistor

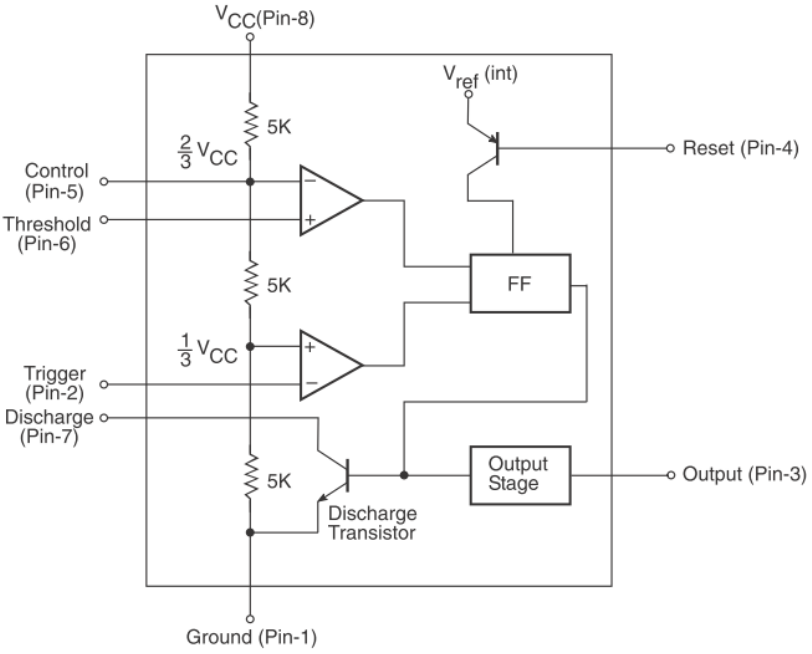
cực nền (B) của Q_2 vượt thế ngưỡng, nó dẫn trở lại và lối ra chuyển sang trạng thái thấp. Quá trình này lặp đi lặp lại nhờ cặp transistor thay nhau dẫn.

Chu kỳ xung của mạch được tính bằng $T = T_1 + T_2$ trong đó: $T_1 = 0,69(R_1C_1)$ và $T_2 = 0,69(R_2C_2)$.

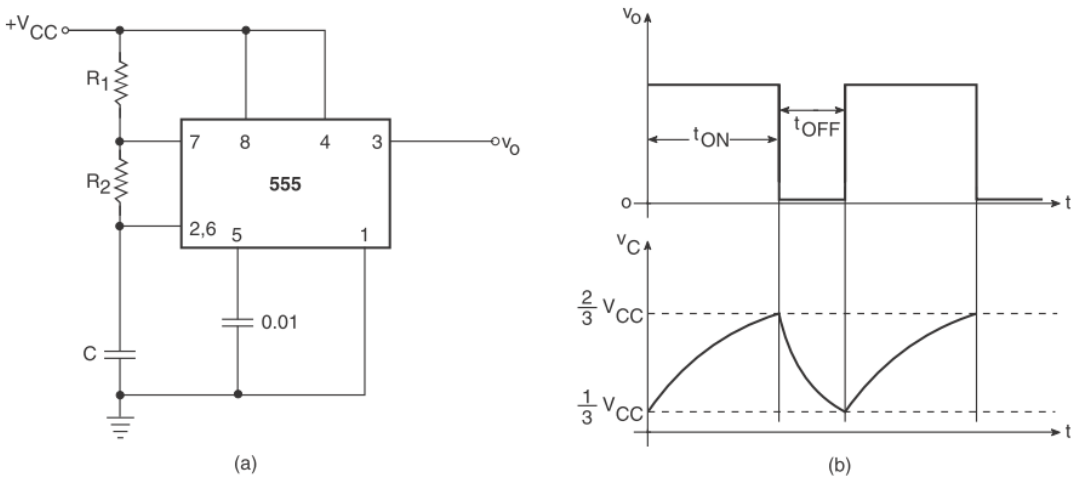
Cuối cùng chúng ta xét mạch sử dụng một IC thông dụng. Có rất nhiều IC chuyên dùng để phát xung, trong đó IC-555 là rẻ tiền và phổ biến nhất. Hình 7.17 minh họa cấu tạo bên trong IC-555. Nó bao gồm hai bộ so sánh, một FF, một transistor xả và ba điện trở. Các điện trở đóng vai trò như một cầu phân áp để đặt ngưỡng so sánh.

Mạch phát xung vuông dùng IC-555 cơ bản được minh họa trên Hình 7.18a. Ban đầu tụ C không có điện, lối ra ở trạng thái cao. Transistor xả ngưng dẫn giúp nạp tụ C thông qua R_1 và R_2. Khi điện áp trên tụ C vượt ngưỡng $+2Vcc/3$, bộ so sánh lật trạng thái khiến FF thay đổi theo, lối ra nhảy xuống mức thấp và transistor xả được kích hoạt. Tụ C xả điện qua R_2 và transistor. Khi điện áp tại tụ C rơi xuống dưới mức $+Vcc/3$ thì bộ so sánh đảo lại trạng thái khiến FF lật, lối ra nhảy lên trạng thái cao. Transistor xả bị ngắt và quy trình này lặp lại khiến mạch hoạt động như một mạch dao động. Chân số 4 (Reset) thường nối lên mức cao, nếu đặt mức điện áp thấp sẽ khóa tác động của chân số 2.

Chu kỳ xung của mạch được tính bằng $T = T_{cao} + T_{thp}$ trong đó: $T_{cao} = 0,69(R_1 + R_2)C$ và $T_{thp} = 0,69(R_2)C$. Giản đồ xung có thể thấy trên Hình 7.18b.



Hình 7.17 Bên trong một IC555

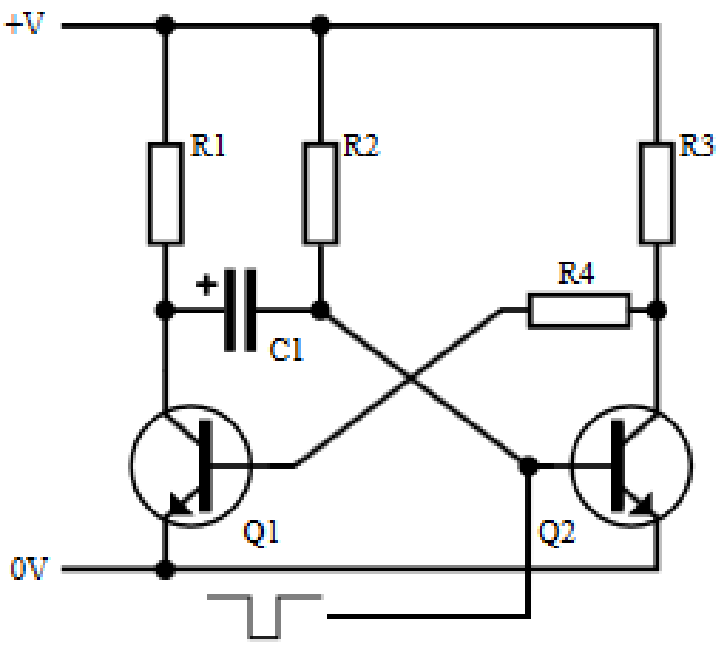


Hình 7.18 Mạch phát xung vuông dùng IC555

Dao động đơn hài

Mạch này còn có tên gọi khác là mạch dao động đơn ổn. Mạch chỉ tạo ra một xung khi nhận vào một xung kích thích mà không duy trì dao động. Đầu tiên xét mạch dao động đơn hài dùng transistor như trên Hình 7.19. Khi mới cấp điện, transistor Q1 ngưng trong khi Q2 dẫn, mạch ở trạng thái ổn định. Khi có một xung âm đặt vào cực nền (B) của Q2, nó ngưng dẫn khiến điện áp cực nền (B) của Q1 tăng, Q1 sẽ dẫn. Tụ C1 được nạp qua trở điện R_2. Sau một khoảng thời gian, khi điện áp tăng lên sẽ khiến Q2 dẫn và mạch trở về trạng thái ổn định như ban đầu.

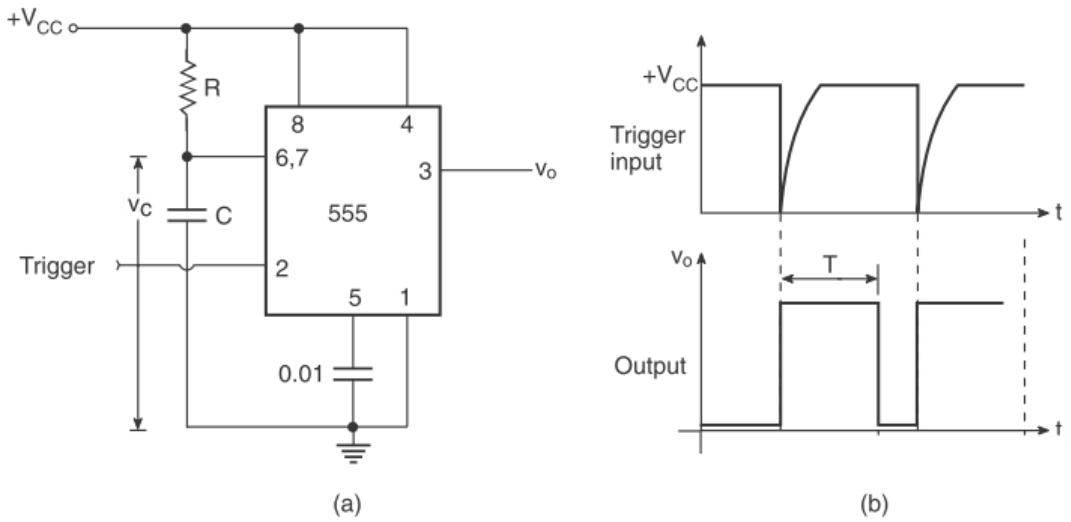
Khoảng thời gian của xung lối ra được tính bằng $t = 0,69R_2C_1$. Lối ra sẽ giữ trạng thái cao khi xung lối vào giữ trạng thái, sau khi xung lối vào kết thúc thì lối ra sẽ giữ cao trong một khoảng thời gian nhất định. Mạch cũng có thể được kích hoạt bằng một xung dương vào cực nền



Hình 7.19 Mạch dao động đơn hài dùng transistor

(B) của Q1. Nhưng lúc này, lối ra chỉ tạo ra một xung có độ rộng nhất định bất chấp độ rộng xung lối vào.

Mạch dao động đơn hài sử dụng IC-555 được minh họa trên Hình 7.20a. Chân số 2 được treo ở trạng thái cao, một xung kích thích âm được đưa vào khiến lối ra chuyển lên trạng thái cao. Tụ C được nạp thông qua điện trở R, khi điện áp vượt ngưỡng $+2Vcc/3$ thì lối ra trở về mức thấp. Khoảng thời gian này được tính bằng $T = 1,1RC$. Giản đồ xung được thể hiện trên Hình 7.20b.



Hình 7.20 Mạch dao động đơn hài dùng IC-555

Trên thực tế, người ta sử dụng rất nhiều loại mạch tạo xung khác nhau tùy thuộc vào yêu cầu của từng hệ thống. Mỗi loại mạch có ưu điểm và nhược điểm khác nhau, từ giá thành, độ phức tạp cho đến tính chính xác.

8

Mạch Tuần Tự

8.1 Giới thiệu

Flip-flop có thể được sử dụng cho nhiều ứng dụng từ bộ đếm, lưu dữ liệu nhị phân đến truyền dữ liệu, ... Rất nhiều trong số chúng nằm trong nhóm mạch tuần tự. Mạch tuần tự là mạch với các đầu ra sẽ nhận những giá trị đã định sau mỗi xung đồng bộ.

8.2 Bộ đếm bất đồng bộ và đồng bộ

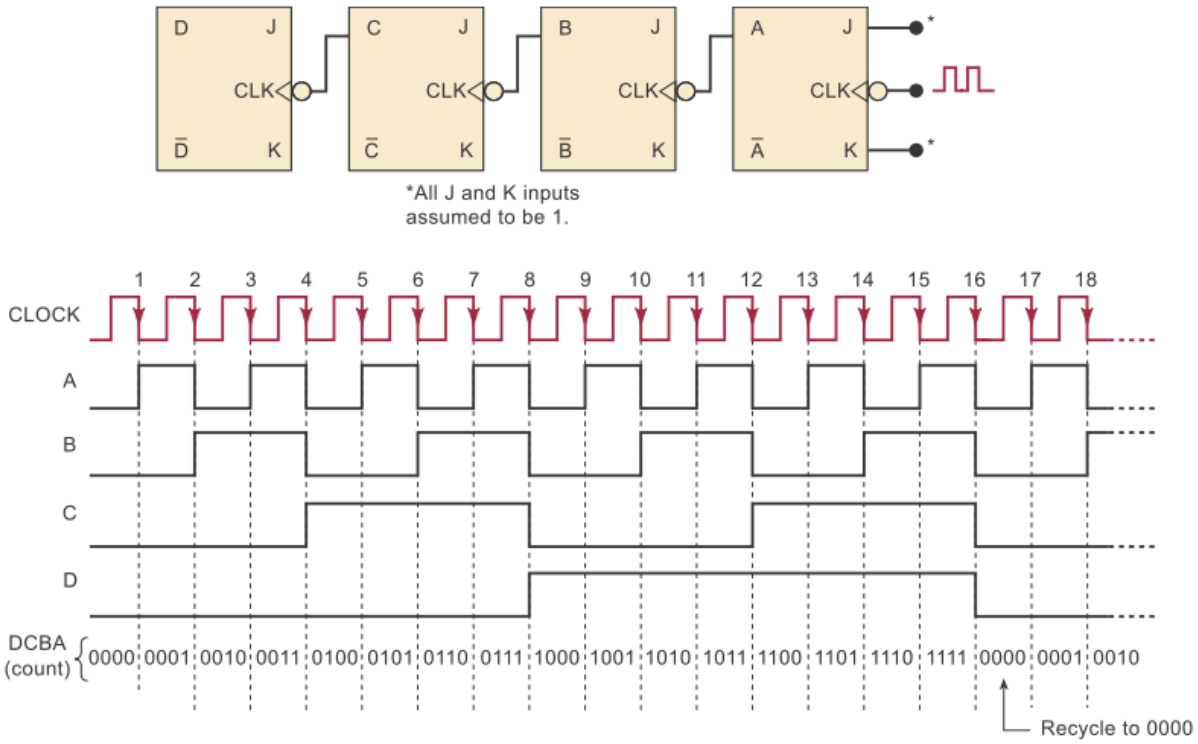
Bộ đếm bất động bộ (Ripple counter) 4 bit cơ bản được minh họa trên Hình 8.1. Lối vào JK của các FF được gán bằng 1. Chú ý cách vẽ đường tín hiệu từ phải sang chỉ nhằm thể hiện rõ hơn bộ đếm theo thứ tự MSB đến LSB.

Số MOD nói chung bằng với trạng thái mà mạch đếm hoàn tất một chu kỳ. MOD của mạch đếm trong ví dụ là 16 vì có 4 FF nên $2^4 = 16$. Trong bất cứ bộ đếm nào, tín hiệu tại lối ra của FF cuối cùng sẽ có tần số bằng với tần số xung đồng hồ chia cho số MOD của bộ đếm đó.

Độ trễ truyền qua

Tuy rằng bộ đếm ripple đơn giản nhưng yếu điểm là phụ thuộc vào độ trễ truyền qua của từng FF. Độ trễ của cả bộ đếm bằng tổng độ trễ của từng FF. Như vậy, tại tần số cao bộ đếm sẽ không hoạt động như mong muốn.

Để đảm bảo bộ đếm hoạt động đúng chức năng, chu kỳ xung đồng hồ phải lớn hơn tổng thời gian trễ: $T_{clk} \geq N \times t_d$ với N là số FF và t_d là độ trễ. Tính theo tần số: $f_{max} = \frac{1}{N \times t_d}$

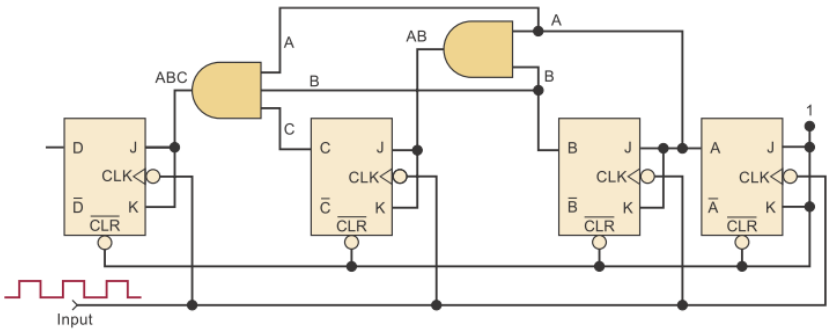


Hình 8.1 Mạch đếm bất đồng bộ

Có thể thấy, bộ đếm bất đồng bộ không khả dụng tại tần số cao, đặc biệt với bộ đếm nhiều bit. Một vấn đề nữa là dò trạng thái lối ra của bộ đếm. Tại một thời điểm nhất định, các trạng thái lối ra không thể hiện đúng như thiết kế. Những lỗi này thường gọi là *glitch*.

Bộ đếm đồng bộ

Để tránh trường hợp bộ đếm sai do bị cộng dồn độ trễ qua các FF, bộ đếm đồng bộ (song song) có tất cả các FF được kích thích đồng thời. Hình 8.2 minh họa một bộ đếm MOD-16 đồng bộ. Đầu vào JK của FF được nối chung, chúng đạt trạng thái cao khi tất cả lối ra của các FF thứ hạng thấp hơn có trạng thái cao.



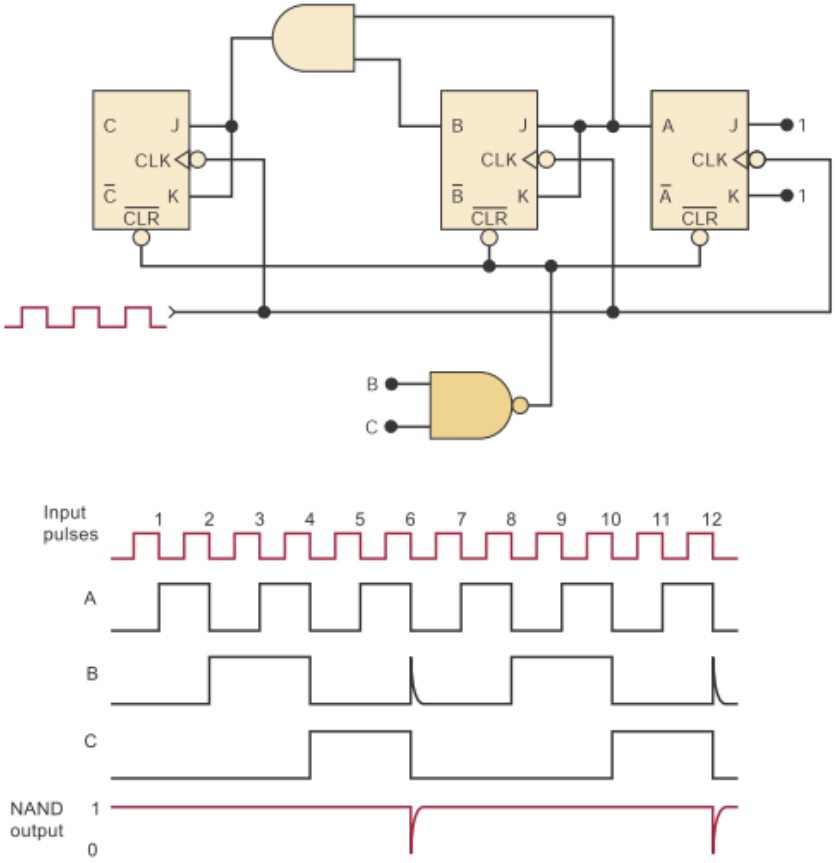
Hình 8.2 Mạch đếm đồng bộ 4 bit

Vì các FF có lối vào CLK chung nên độ trễ chỉ phụ thuộc vào một FF và cổng AND đi kèm. Giá trị này không đổi bất kể số lượng FF có trong mạch.

8.3 Bộ đếm MOD

Bộ đếm bất đồng bộ

Với N flip-flop, số đếm tối đa của một bộ đếm là 2^N. Trên cơ sở bộ đếm này, ta có thể tạo ra các bộ đếm có số đếm nhỏ hơn 2^N bằng cách cho bộ đếm bỏ qua một số trạng thái trong chuỗi đếm. Ví dụ minh họa trong Hình 8.3.



Hình 8.3 Mạch đếm MOD6 dựa trên 3 flip-flop

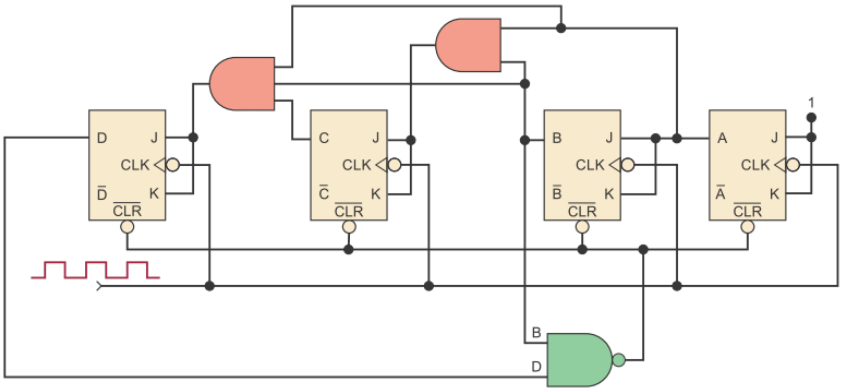
Cổng NAND có tác dụng xóa flip-flop bất đồng bộ. Thời gian xóa tạo bởi NAND rất nhỏ (chỉ vài ns) khiến bộ đếm trở về trạng thái ban đầu, số đếm bằng 000. Sau đó, lối ra NAND lại được bật lên cao. Như vậy, cho dù bộ đếm có đạt trạng thái '110', nó chỉ tồn tại trong thời gian rất ngắn nên bộ đếm chỉ có 6 giá trị.

Để xây dựng bộ đếm MOD có số đếm X, quy trình đơn giản như sau:

- Tìm số FF nhỏ nhất sao cho $2^N \geq X$ và tạo bộ đếm bình thường. Nếu $2^N = X$ thì bộ đếm đã hoàn thành, không cần sửa đổi.

- Trường hợp còn lại, sử dụng cổng NAND để xóa bất đồng bộ các FF.
- Xác định FF có lối ra trạng thái cao khi đếm bằng X; lối ra của các FF này là lối vào của cổng NAND.

Hình 8.4 minh họa bộ đếm MOD10 với số đếm từ 0000 đến 1001 nên còn được gọi là bộ đếm BCD.



Hình 8.4 Mạch đếm MOD10 dùng 4 flip-flop

Bộ đếm lên và xuống

Bộ đếm xuống đồng bộ được xây dựng tương tự như trên chỉ khác việc sử dụng lối ra đảo của các FFJK. Kết hợp cả hai phương pháp, ta có thể xây dựng bộ đếm lên xuống với đầu vào điều khiển như minh họa trong Hình 8.5. Sử dụng chân điều khiển Up/Down để lấy trạng thái từ lối ra thường hoặc lối ra đảo của FFJK.

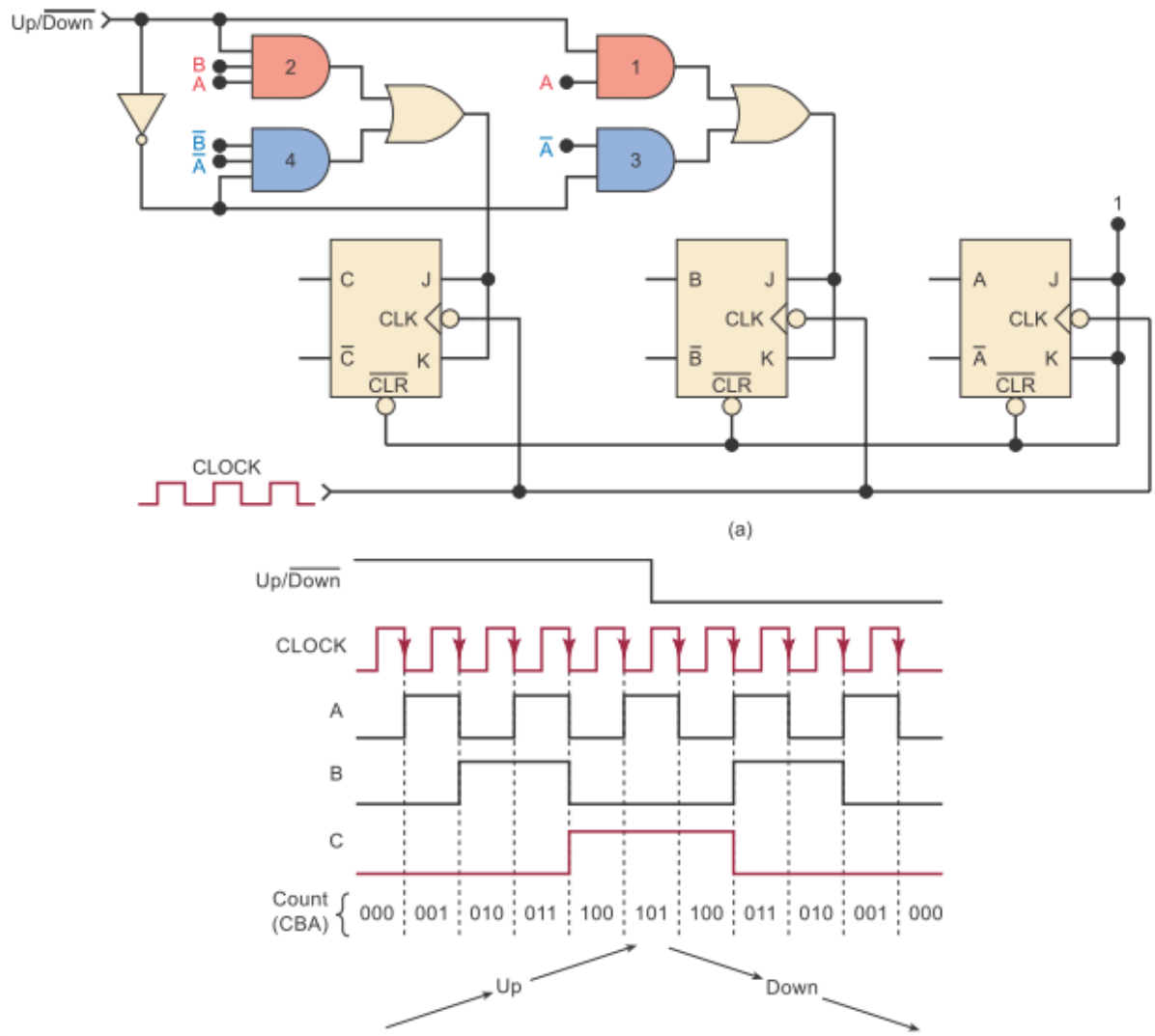
Ký hiệu (Up/$\overline{Down}$) được sử dụng để chỉ trạng thái đếm lên khi tác động mức cao và đếm xuống khi tác động mức thấp.

Bộ đếm có số đếm thay đổi được

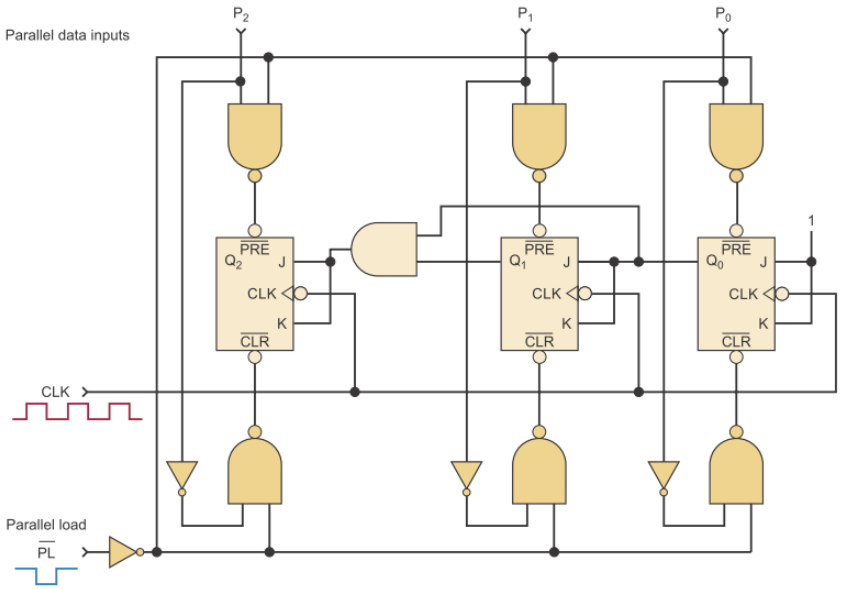
Một số bộ đếm có thể thay đổi được giá trị ban đầu. Hình 8.6 minh họa một bộ đếm lên thay đổi được sử dụng FFJK. Chân CLEAR và PRESET bất đồng bộ được sử dụng để thay đổi trạng thái FF. Các bước tiến hành:

- Gán giá trị mong muốn cho các chân đầu vào song song P_2, P_1 và P_0.
- Đưa một xung thấp đến lối vào nạp song song $\overline{PL}$.

Trong khi $\overline{PL}$ có trạng thái thấp, lối vào CLK bị mất tác dụng. Một khi $\overline{PL}$ trở về trạng thái cao, FF hoạt động theo CLK với giá trị bắt đầu được lấy từ P_2, P_1 và P_0.



Hình 8.5 Mạch đếm MOD8 có khả năng đếm lên/xuống



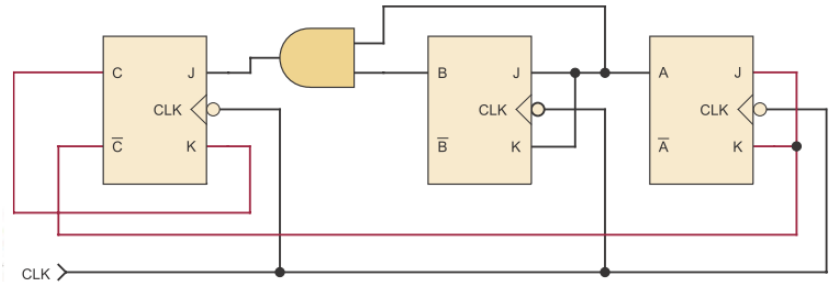
Hình 8.6 Mạch đếm lên có giá trị đầu thay đổi được

8.4 Bộ đếm đồng bộ

Phân tích bộ đếm

Để phân tích một bộ đếm có sẵn, bảng trạng thái FF là một công cụ rất hữu ích. Hình 8.7 minh họa một bộ đếm đồng bộ có sự thay đổi ở lối vào. Biểu thức điều khiển như sau:

- $J_C = A.B$
- $K_C = C$
- $J_B = K_B = A$
- $J_A = K_A = \bar{C}$



PRESENT State			Control Inputs						NEXT State		
C	B	A	J_C	K_C	J_B	K_B	J_A	K_A	C	B	A
0	0	0	0	0	0	0	1	1	0	0	1
0	0	1	0	0	1	1	1	1	0	1	0
0	1	0	0	0	0	0	1	1	0	1	1
0	1	1	1	0	1	1	1	1	1	0	0
1	0	0	0	1	0	0	0	0	0	0	0
1	0	1	0	1	1	1	0	0	0	1	1
1	1	0	0	1	0	0	0	0	0	1	0
1	1	1	1	1	1	1	0	0	0	0	1

Hình 8.7 Phân tích mạch đếm với bảng trạng thái

Có thể thấy đây là bộ đếm MOD5 với ba số đếm ngoài vùng hoạt động. Một bộ đếm nếu rơi vào vùng cấm mà có thể tự trở về trạng thái hoạt động thì gọi là bộ đếm tự điều chỉnh. Ngược lại, bộ đếm không tự điều chỉnh sẽ không thể trở về trạng thái hoạt động khi số đếm rơi vào vùng cấm.

Thiết kế bộ đếm đồng bộ

Trong mạch đếm đồng bộ, các FF sẽ nhận xung đồng hồ CLK cùng thời điểm. Giá trị tại các đầu vào phải đạt trạng thái xác định trước mỗi xung để đảm bảo trạng thái lối ra như mong muốn.

Bảng kích thích J-K được minh họa trong Hình 8.8. Cột đầu tiên gồm danh sách chuyển đổi trạng thái có thể xảy ra. Hai cột tiếp theo thể hiện trạng thái hiện tại và kế cận của FF trong khi cột cuối cùng là mức trạng thái JK cần thiết để chuyển đổi trạng thái FF.

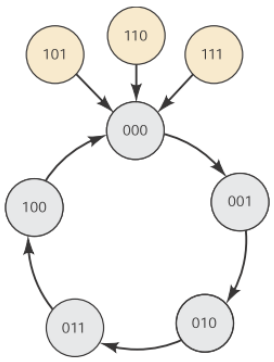
Transition at FF Output	PRESENT State Q_n	NEXT State Q_{n+1}	J	K
0 → 0	0	0	0	x
0 → 1	0	1	1	x
1 → 0	1	0	x	1
1 → 1	1	1	x	0

Hình 8.8 Bảng kích thích J-K

Quy trình thiết kế mạch đếm đồng bộ đơn giản như sau:

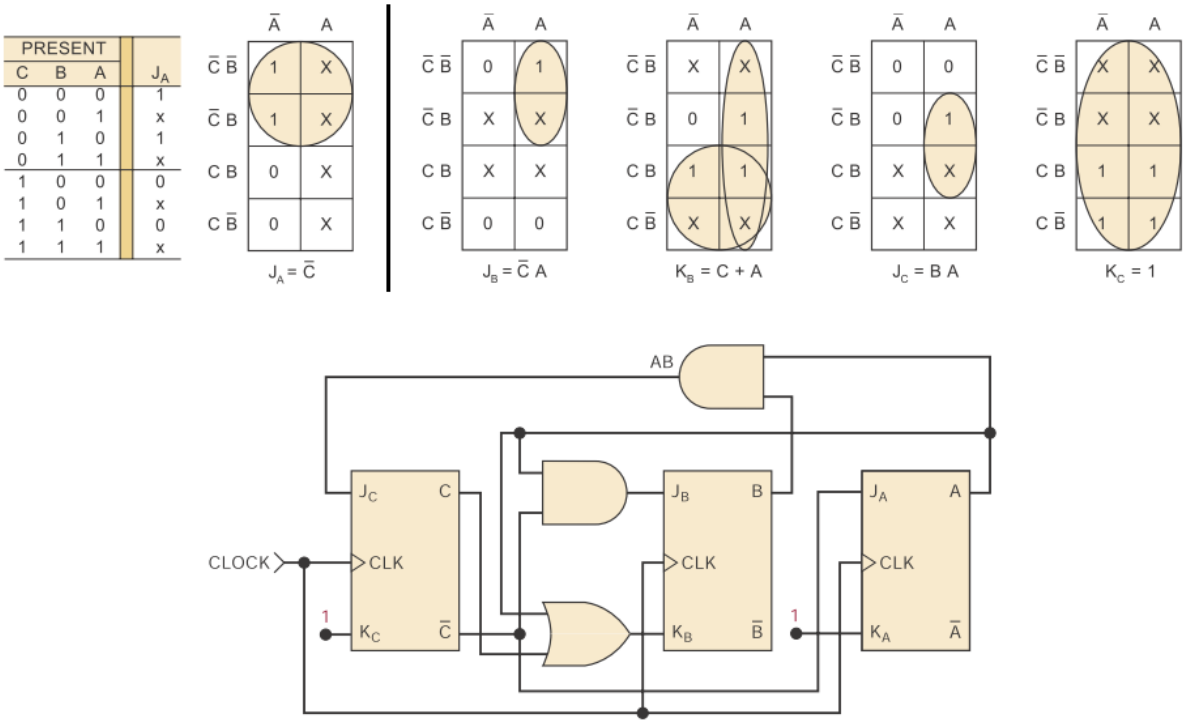
1. Xác định số FF cần thiết và chuỗi đếm theo yêu cầu.
2. Vẽ lưu đồ chuyển trạng thái với tất cả các trạng thái có thể xảy ra; kể cả trạng thái cấm.
3. Sử dụng lưu đồ để tạo bảng liệt kê tất cả các trạng thái
4. Bổ sung vào bảng một cột cho từng lối vào J-K. Với mỗi trạng thái hiện thời, chỉ định mức trạng thái từng lối vào J-K để tạo ra chuyển trạng thái tiếp theo.
5. Thiết kế mạch cần thiết để tạo ra mức trạng thái theo yêu cầu tại từng lối vào J-K.
6. Thực hiện mạch hoàn chỉnh.

Quy trình được trình bày lần lượt qua ví dụ tạo mạch đếm MOD5 minh họa trong Hình 8.9



	PRESENT State			NEXT State								
	C	B	A	C	B	A	J_C	K_C	J_B	K_B	J_A	K_A
Line 1	0	0	0	0	0	1	0	x	0	x	1	x
2	0	0	1	0	1	0	0	x	1	x	x	1
3	0	1	0	0	1	1	0	x	x	0	1	x
4	0	1	1	1	0	0	1	x	x	1	x	1
5	1	0	0	0	0	0	x	1	0	x	0	x
6	1	0	1	0	0	0	x	1	0	x	x	1
7	1	1	0	0	0	0	x	1	x	1	0	x
8	1	1	1	0	0	0	x	1	x	1	x	1

Hình 8.9 Các bước thiết kế mạch đếm MOD5 theo yêu cầu

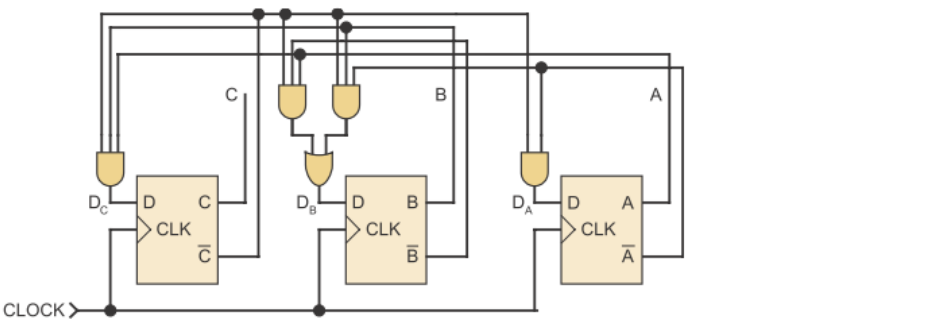


Hình 8.10 Các bước thiết kế mạch đếm MOD5 (tiếp theo)

Sử dụng FFD

Có thể sử dụng FFD để thiết kế mạch đếm thay cho FFJK. Theo ví dụ trên, ba bước đầu tiên trong quá trình thiết kế cũng tương tự như của FFJK. Riêng bước thứ 4 có thay đổi vì FFD chỉ có 1 đầu vào. Quy trình được trình bày trên Hình 8.11.

PRESENT State			NEXT State			Control Inputs		
C	B	A	C	B	A	D_C	D_B	D_A
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0



Hình 8.11 Thiết kế mạch đếm MOD5 dùng FFD

Một số thuật ngữ

Máy trạng thái (state machine) dùng để chỉ những mạch hoạt động qua một loạt các trạng thái đã định trước và được điều khiển bởi các đầu vào điều khiển và xung đồng hồ. Như vậy, những mạch đếm đã xét cũng có thể được gọi là các máy trạng thái.

Thuật ngữ máy trạng thái thường được dùng để mô tả các mạch tuần tự khác. Chúng có thể có dạng đếm không theo quy luật. Điểm phân biệt giữa mạch đếm và máy trạng thái là một bên dùng để đếm sự kiện, một bên dùng để điều khiển sự kiện.

Mô hình Mealy để chỉ các mạch có tín hiệu lối ra được điều khiển đồng thời bởi tín hiệu tại lối vào thường và đầu vào bổ sung, trong khi đó **mô hình Moore** thì không có. Do vậy, mạch kiểu Moore có lối ra đồng bộ với xung đồng hồ. Ngược lại, lối ra của mạch kiểu Mealy có thể được thay đổi bất đồng bộ.

8.5 Thanh ghi

Các thanh ghi thường được xây dựng bằng FF vì đặc tính lưu giữ trạng thái của nó. Phân loại thanh ghi dựa trên cách thức dữ liệu đi vào và ra:

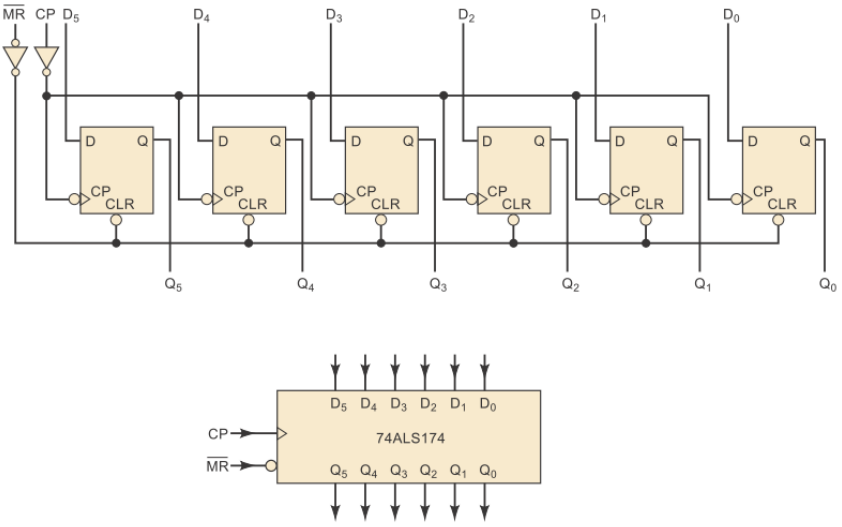
1. Vào ra song song (PIPO). Ví dụ: 74xx174
2. Vào ra nối tiếp (SISO). Ví dụ: 74xx166
3. Vào song song/ ra nối tiếp (PISO). Ví dụ: 74xx165
4. Vào nối tiếp/ ra song song (SIPO). Ví dụ: 74xx164

Các loại thanh ghi và biến thể của chúng đều được sản xuất sẵn dưới dạng IC nên rất thuận tiện cho người thiết kế. Hình 8.12 minh họa sơ đồ mạch và biểu tượng của một thanh ghi dịch vào ra song song dạng IC.

Mạch đếm thanh ghi dịch

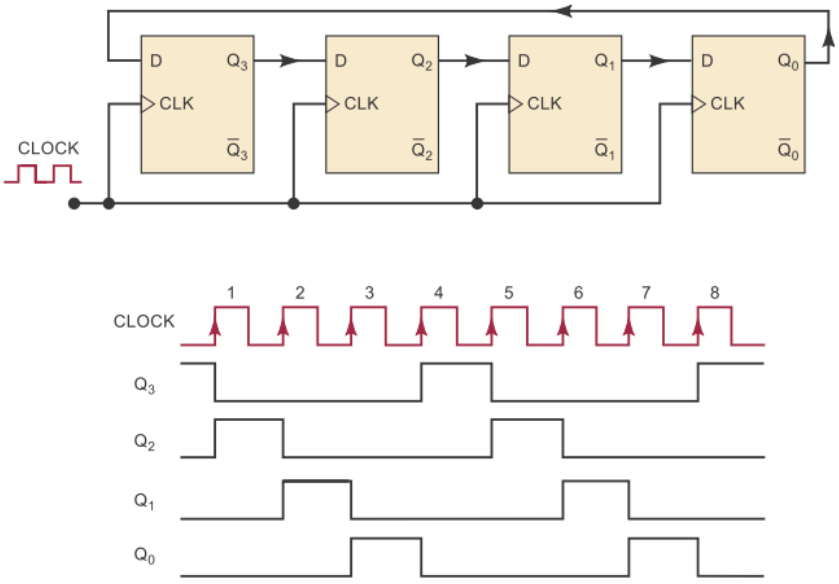
Mạch đếm vòng (Ring counter) được tạo thành bằng cách sử dụng FF cuối cùng của thanh ghi dịch đưa dữ liệu vào FF đầu tiên. Trong hầu hết các trường hợp, chỉ có một số ‘1’ trong thanh ghi và nó được dịch vòng tròn khi có xung đồng hồ.

Bằng việc kết nối 4 FF-D theo cách trên, mạch đếm MOD4 được hình thành cho dù nó không sinh ra chuỗi đếm nhị phân bình thường. Vì mỗi lần đếm nó đều tạo ra một trạng thái khác nhau của tổ hợp các FF nên mạch đếm MOD-N sẽ được tạo bởi N flip-flop như minh họa trong Hình



Hình 8.12 IC 74ALS174 PIPO

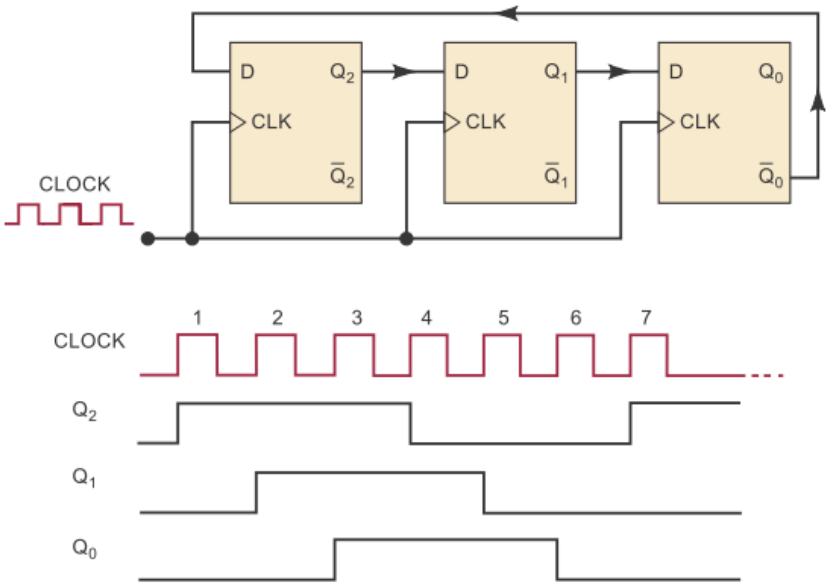
8.13. Để bắt đầu, một xung hiệu sẽ được cấp cho một FF (ví dụ chân $\overline{PRE}$) trong khi xóa các FF còn lại (chân $\overline{CLR}$).



Hình 8.13 Mạch đếm MOD4 - ring

Tuy rằng mạch đếm vòng dùng nhiều FF hơn mạch đếm nhị phân, nó vẫn được dùng vì dễ giải mã và chuỗi trạng thái lối ra dễ nhận biết hơn.

Mạch đếm Johnson được tạo thành từ mạch đếm vòng. Với cấu hình tương tự, chỉ có một điểm khác là lối ra đảo của FF cuối cùng sẽ được sử dụng để đưa vào FF đầu. Giả sử ban đầu tất cả FF mang giá trị '0', giản đồ sóng của mạch đếm được thể hiện trong Hình 8.14. Như vậy, để tạo ra mạch đếm MOD-N Johnson thì cần N/2 FF.



Hình 8.14 Mạch đếm MOD6 Johnson

9

Biến Đổi Tương Tự - Số

9.1 Giới thiệu

Đại lượng số chỉ có thể nhận hai giá trị là ‘0’ hoặc ‘1’, thấp hoặc cao, đúng hoặc sai,.... Trên thực tế, một đại lượng bất kỳ ví dụ như điện áp có thể rơi vào một khoảng nhất định. Ta có thể gom từng khoảng giá trị thành một giá trị số. Ngược lại, đại lượng tương tự có thể có giá trị trải rộng và liên tục. Đồng thời, tính chính xác và giá trị cụ thể của từng đại lượng tương tự này mang ý nghĩa lớn đối với hệ thống. Bộ biến đổi số sang tương tự và ngược lại đóng vai trò là cầu nối giữa hai hệ thống hoàn toàn khác nhau.

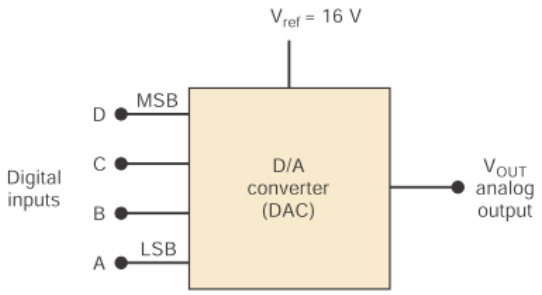
9.2 Biến đổi số sang tương tự

Về cơ bản, biến đổi số sang tương tự D/A (digital-to-analog) là quá trình chuyển giá trị thể hiện bằng số (nhị phân hoặc BCD) sang đại lượng điện áp hoặc dòng điện có giá trị tương đương. Hình 9.1 minh họa bộ biến đổi D/A với đầu vào tham chiếu. Điện áp tham chiếu được sử dụng để xác định giá trị tối đa tại lối ra.

Tổng quát, lối ra tương tự $= K\times$ lối vào số, với K là hệ số nhân và là hằng số đối với từng bộ DAC. Hệ số K mang đơn vị của giá trị tại lối ra. Trong ví dụ, K = 1V, $V_{OUT} = 1V\times$ lối vào số.

Đặc điểm của DAC

Về mặt kỹ thuật, lối ra của DAC không hoàn toàn là một đại lượng tương tự vì nó chỉ có thể nhận một giá trị xác định. Do vậy, có thể xem lối ra của DAC vẫn là số. Tuy nhiên, khi tăng số bit lối vào, dải giá trị lối ra tăng lên và sự khác biệt giữa các giá trị này giảm xuống. Điều này hình thành một lối ra có dải giá trị rộng và giống như đại lượng tương tự.



D	C	B	A	V_{OUT}	
0	0	0	0	0	Volts
0	0	0	1	1	
0	0	1	0	2	
0	0	1	1	3	
0	1	0	0	4	
0	1	0	1	5	
0	1	1	0	6	
0	1	1	1	7	
1	0	0	0	8	
1	0	0	1	9	
1	0	1	0	10	
1	0	1	1	11	
1	1	0	0	12	
1	1	0	1	13	
1	1	1	0	14	
1	1	1	1	15	Volts

Hình 9.1 Biến đổi DAC 4 bit với lối ra điện áp

Trọng số lối vào: Mỗi lối vào số có trọng số khác nhau. Trọng số được tăng gấp đôi qua từng bit, bắt đầu từ LSB.

Độ phân giải của DAC được định nghĩa là sự thay đổi nhỏ nhất có thể xảy ra tại lối ra tương tự gây bởi sự thay đổi ở lối vào số. Trong ví dụ trên, độ phân giải của DAC là 1V. Độ phân giải còn được gọi là độ lớn bước DAC. Với một DAC có N bit, số bước liên tục được tạo ra là $2^N - 1$. Độ phân giải của DAC còn có thể được tính theo %.

Lối ra DAC không thể mang giá trị trong một dải liên tục mà chỉ là một giá trị nhất định trong một tập giá trị. Một DAC nhiều bit sẽ giúp tăng độ phân giải. Độ phân giải sẽ được quyết định dựa trên yêu cầu của hệ thống.

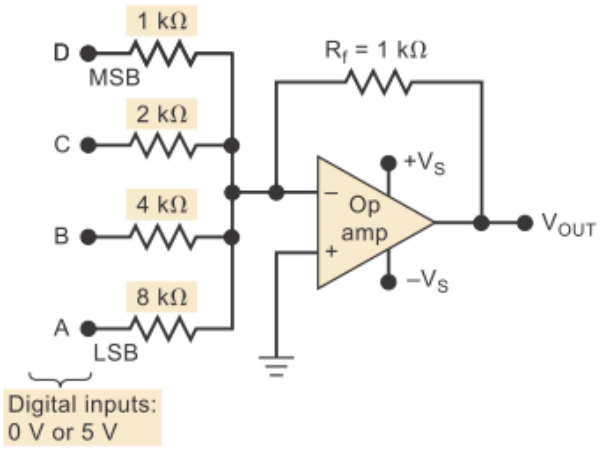
Mạch biến đổi D/A

Có nhiều phương pháp biến đổi D/A và IC DAC hiện nay rất phổ biến. Hình 9.2 minh họa mạch DAC sử dụng khuyếch đại thuật toán như bộ cộng. Lối ra của op-amp được tính như sau: $V_{OUT} = -(V_D + \frac{1}{2}V_C + \frac{1}{4}V_B + \frac{1}{8}V_A)$

Độ chính xác của mạch phụ thuộc hai yếu tố: một là độ chính xác của điện áp tại các lối vào và điện trở phản hồi, hai là độ chính xác của mức điện áp chuẩn. Điện trở có thể đạt độ chính xác cao nhưng mức điện áp lối vào cần được xử lý khác nhau.

Trên thực tế, việc sử dụng thang điện trở như trên đối với DAC nhiều bit sẽ khó có thể áp dụng vì cách biệt giá trị điện trở quá lớn. Do vậy, mạch DAC dùng trở được sửa đổi như minh họa trong Hình 9.3.

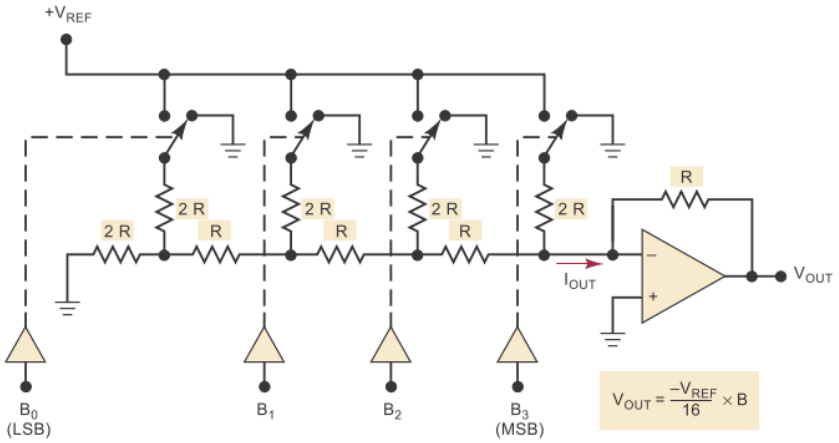
Điện trở được sử dụng chỉ có hai giá trị R và 2R. Dòng điện ra phụ thuộc vào bốn công tắc tương ứng nhị phân $B_3B_2B_1B_0$. Dòng ra được đưa vào bộ biến đổi dòng sang áp để thu giá trị $V_{OUT} = \frac{-V_{REF}}{16} \times B$, trong đó B là giá trị nhị phân 4 bit đầu vào.



Input code

D	C	B	A	V_{OUT} (volts)
0	0	0	0	0
0	0	0	1	−0.625 ← LSB
0	0	1	0	−1.250
0	0	1	1	−1.875
0	1	0	0	−2.500
0	1	0	1	−3.125
0	1	1	0	−3.750
0	1	1	1	−4.375
1	0	0	0	−5.000
1	0	0	1	−5.625
1	0	1	0	−6.250
1	0	1	1	−6.875
1	1	0	0	−7.500
1	1	0	1	−8.125
1	1	1	0	−8.750
1	1	1	1	−9.375 ← Full-scale

Hình 9.2 Biến đổi DAC dùng thang điện trở



Hình 9.3 Biến đổi DAC dùng thang điện trở R/2R

Thông số của DAC

Độ phân giải phụ thuộc vào số bit của DAC. Một DAC 10bit sẽ có độ phân giải nhỏ hơn một DAC 8bit.

Độ chính xác được quy định bởi các nhà sản xuất. Hai chỉ số cơ bản là sai số tuyến tính và sai số tổng. Sai số tổng thường được cung cấp dạng % thể hiện độ lệch của lối ra so với giá trị thực. Sai số tuyến tính thể hiện độ lệch tối đa của bước DAC so với bước lý tưởng.

Sai số lệch chuẩn được xác định bằng giá trị lối ra khi lối vào bằng 0. Nếu không hiệu chỉnh sai số lệch chuẩn, nó sẽ ảnh hưởng đến độ chính xác của các lần biến đổi sau này.

Thời gian đáp ứng ổn định là khoảng thời gian cần thiết để lối ra DAC chạy từ 0 đến giá trị tối đa khi giá trị nhị phân tại lối vào chuyển hết từ ‘0’ sang ‘1’.

Tính đơn hướng thể hiện khi lối ra DAC chỉ có một chiều tăng khi giá trị nhị phân đầu vào tăng lần lượt.

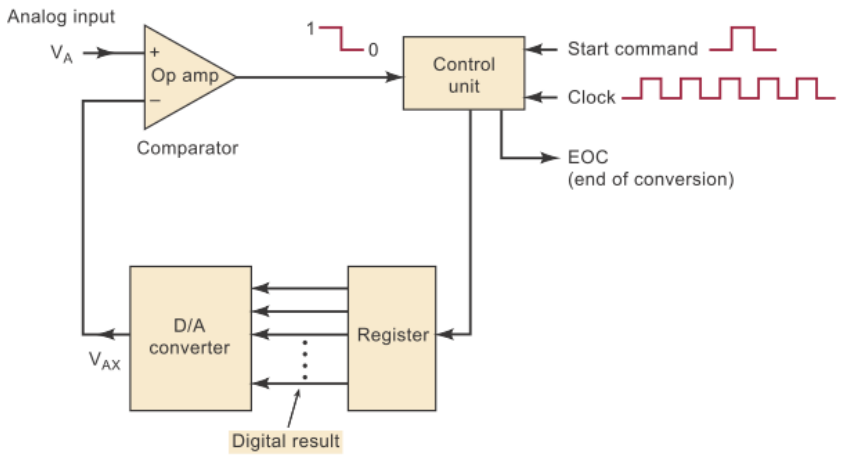
IC DAC điển hình là AD7524 là DAC 8bit sử dụng phương pháp trở thang R/2R. Nó có thời gian ổn định khoảng 100ns với sai số tổng ±2%.

9.3 Biến đổi tương tự sang số

Bộ biến đổi tương tự - số (analog-to-digital converter) ADC biến tín hiệu điện đầu vào thành tín hiệu dạng số ở đầu ra sau một khoảng thời gian trễ. ADC thường tốn thời gian hơn là DAC.

Một số loại ADC tận dụng DAC cho quá trình biến đổi. Hình 9.4 minh họa sơ đồ khối tổng quát của một loại ADC. Các bước hoạt động như sau:

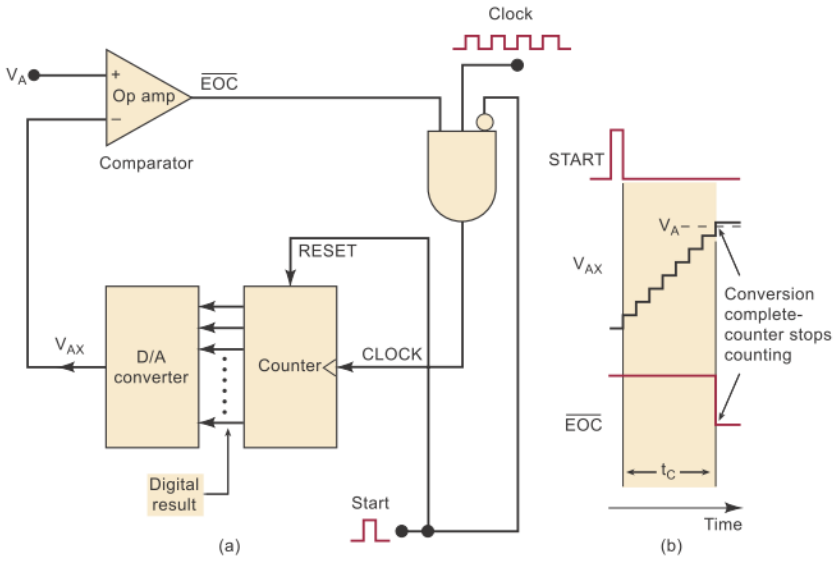
- Xung lệnh bắt đầu kích hoạt hệ thống
- Phụ thuộc tần số xung đồng hồ, bộ điều khiển thay đổi giá trị thanh ghi
- Giá trị nhị phân trong thanh ghi được đổi sang dạng tương tự bởi DAC
- Bộ so sánh thay đổi trạng thái dựa trên giá trị nhận vào. Thanh ghi dừng thay đổi giá trị
- Bộ điều khiển ra lệnh kết thúc quá trình.



Hình 9.4 Biến đổi ADC sử dụng thành phần DAC

ADC bước

Một trong những dạng ADC đơn giản nhất là ADC bước. Nó sử dụng một bộ đếm nhị phân để xác định giá trị so sánh với đầu vào. Hình 9.5 minh họa một ADC bước và cách thức hoạt động đơn giản hóa.



Hình 9.5 Biến đổi ADC bước

Với loại ADC này, sai số phát sinh do độ lớn bước của DAC. Nếu giảm độ lớn của bước, ta có thể giảm sai số loại này. Tuy nhiên, luôn luôn có sự khác biệt giữa giá trị thực và giá trị số tại đầu ra. Đây gọi là sai số lượng hóa. Ngoài ra, ADC còn có các sai số khác do độ chính xác của linh kiện.

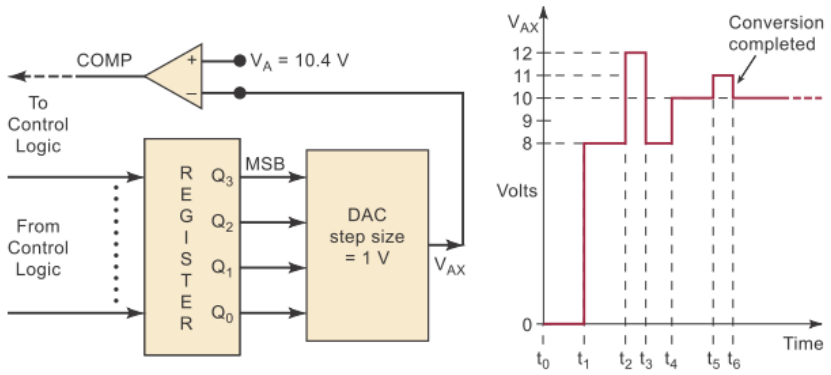
Thời gian biến đổi được định nghĩa là khoảng thời gian giữa xung bắt đầu và xung kết thúc biến đổi. Thời gian biến đổi tối đa cho một bộ ADC N bit bằng $t_C(max) = (2^N - 1) \times T$ với T là chu kỳ của xung đồng hồ.

Tuy rằng ADC bước khá đơn giản, nó mang một nhược điểm là thời gian biến đổi sẽ tăng gấp đôi với mỗi bit được thêm vào. Như vậy, để tăng độ chính xác phải đánh đổi thời gian biến đổi. Do đó, ADC bước thường được dùng trong các hệ thống có tần số hoạt động thấp.

ADC xấp xỉ liên tục

Đây là loại ADC khá phổ biến, tuy rằng có cấu trúc mạch phức tạp hơn ADC bước nhưng lại có thời gian biến đổi nhanh hơn. Ngoài ra, bộ biến đổi xấp xỉ liên tục cho thời gian biến đổi không phụ thuộc giá trị đầu vào tương tự.

SAC (successive approximation converter) không sử dụng bộ đếm để cung cấp giá trị cho DAC mà dùng thanh ghi. Hình 9.6 minh họa một ví dụ với SAC 4 bit. Kết quả của SAC luôn có giá trị thấp hơn giá trị đầu vào thực tế. Trong khi đó, ADC bước có lối ra mang kết quả lớn hơn giá trị đầu vào một bước.



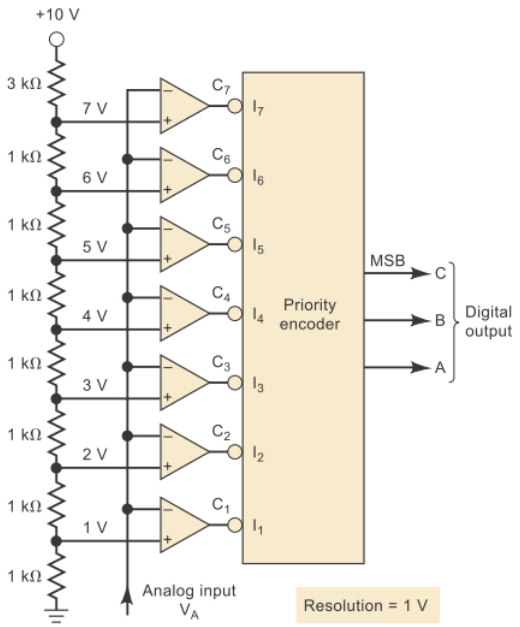
Hình 9.6 Biến đổi SAC 4 bit

Thời gian biến đổi của SAC phụ thuộc vào số bit. Do vậy bất kể giá trị đầu vào, thời gian biến đổi không thay đổi. SAC được dùng nhiều trong các mạch thu thập dữ liệu vì đặc tính thời gian ổn định và tốc độ cao.

IC ADC0804 là một SAC thông dụng với 20 chân.

ADC tốc độ cao

Loại ADC (flash) này có mạch rất phức tạp. Một ADC 6bit cần 63 bộ so sánh trong khi bộ ADC 8bit cần 255 bộ so sánh. Nguyên lý hoạt động của ADC flash được minh họa trong Hình 9.7.



Analog in	Comparator outputs							Digital outputs		
V_A	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C	B	A
0–1 V	1	1	1	1	1	1	1	0	0	0
1–2 V	0	1	1	1	1	1	1	0	0	1
2–3 V	0	0	1	1	1	1	1	0	1	0
3–4 V	0	0	0	1	1	1	1	0	1	1
4–5 V	0	0	0	0	1	1	1	1	0	0
5–6 V	0	0	0	0	0	1	1	1	0	1
6–7 V	0	0	0	0	0	0	1	1	1	0
> 7 V	0	0	0	0	0	0	0	1	1	1

Hình 9.7 Biến đổi ADC flash 3 bit

Để đạt độ phân giải cao hơn, số lượng điện trở phân áp và bộ so sánh phải tăng lên. Một cách tổng quát, bộ ADC flash N bit cần $2^N - 1$ bộ so sánh và 2^N điện trở và mạch giải mã tương ứng.

Thời gian biến đổi chỉ phụ thuộc độ trễ truyền qua của bộ so sánh và giải mã. Chính vì vậy, ADC flash có tốc độ biến đổi nhanh nhất trong các loại ADC.

IC AD9020 là một ADC flash 10bit với thời gian biến đổi chỉ khoảng 17ns.

Một số phương pháp A/D khác

ADC bám được phát triển dựa trên ADC bước. Thời gian ADC bước chuyển từ 0 lên giá trị chuyển đổi rồi trở về 0 là lãng phí. ADC bám hay ADC bước lên/xuống sử dụng bộ đếm lên/xuống để giảm thiểu thời gian lãng phí. Thay vì trở về 0 sau mỗi lần chuyển đổi, bộ đếm sẽ đếm lên hoặc xuống từ giá trị của lần chuyển đổi trước với chiều đếm phụ thuộc kết quả so sánh.

ADC hai sườn dốc có thời gian biến đổi lâu nhất nhưng đơn giản. Nó không sử dụng DAC mà phụ thuộc dòng nạp và xả tụ điện. Dòng điện tạo bởi điện thế đầu vào nạp cho tụ điện trong một khoảng thời gian nhất định. Điện áp trên tụ tỉ lệ với điện áp vào. Sau đó, tụ được xả điện tuyến tính theo điện áp tham chiếu. Khi điện áp tụ về 0, thời gian xả được ghi lại nhờ một bộ đếm. Số đếm tỉ lệ với điện áp lối vào.

ADC hai sườn dốc kháng nhiễu tốt và ít chịu tác động bởi nhiệt độ thay đổi. Với tốc độ chuyển đổi chậm, nó thường được sử dụng trong các hệ đo.

ADC biến áp sang tần số không sử dụng DAC mà dùng bộ dao động tuyến tính điều khiển bởi áp (VCO). Tần số từ bộ VCO tỉ lệ với điện áp đầu vào. Kết quả từ bộ đếm tần sẽ tỉ lệ với điện áp đặt tại lối vào.

Tuy phương pháp này đơn giản, độ chính xác lại không cao vì khó chế tạo VCO có sai số thấp. Ứng dụng thường thấy là trong môi trường công nghiệp, nhiễu cao và tín hiệu nhỏ.

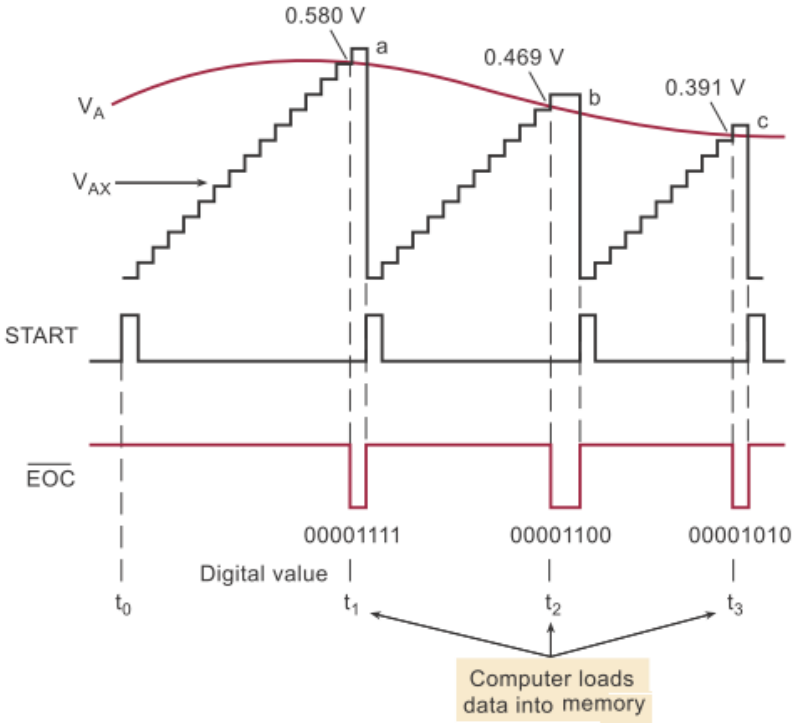
Bộ A/D Sigma/Delta là thiết bị hoạt động dựa trên nguyên tắc lấy mẫu tín hiệu. Giá trị điện áp được thể hiện bằng mật độ số ‘1’ trong dữ liệu thu được. Điều chế sigma/delta được dùng trong cả A/D và D/A.

9.4 Thu thập dữ liệu

Lấy mẫu tín hiệu

Có rất nhiều ứng dụng mà dữ liệu dạng tương tự phải được số hóa để lưu trữ. Quá trình này còn được gọi là thu thập dữ liệu. Lấy mẫu tín hiệu tương tự là thu thập giá trị của một điểm dữ liệu, điểm đó được gọi là một mẫu.

Hình 9.8 minh họa quá trình thu thập dữ liệu số của tín hiệu tương tự. Dạng sóng bậc thang là kết quả của ADC bước. Toàn bộ quy trình cung cấp tín hiệu điều khiển được máy tính kiểm soát.



Hình 9.8 Thu thập dữ liệu qua ADC

Tái dựng tín hiệu từ dữ liệu số của mạch trên đòi hỏi giá trị thời gian lấy mẫu. Do vậy, khi thu thập dữ liệu, người ta lấy mẫu theo khoảng thời gian nhất định. Tần số lấy mẫu tuân theo nguyên tắc Nyquist của xử lý tín hiệu số. Hình 9.9 minh họa quá trình lấy mẫu và tái dựng tín hiệu.

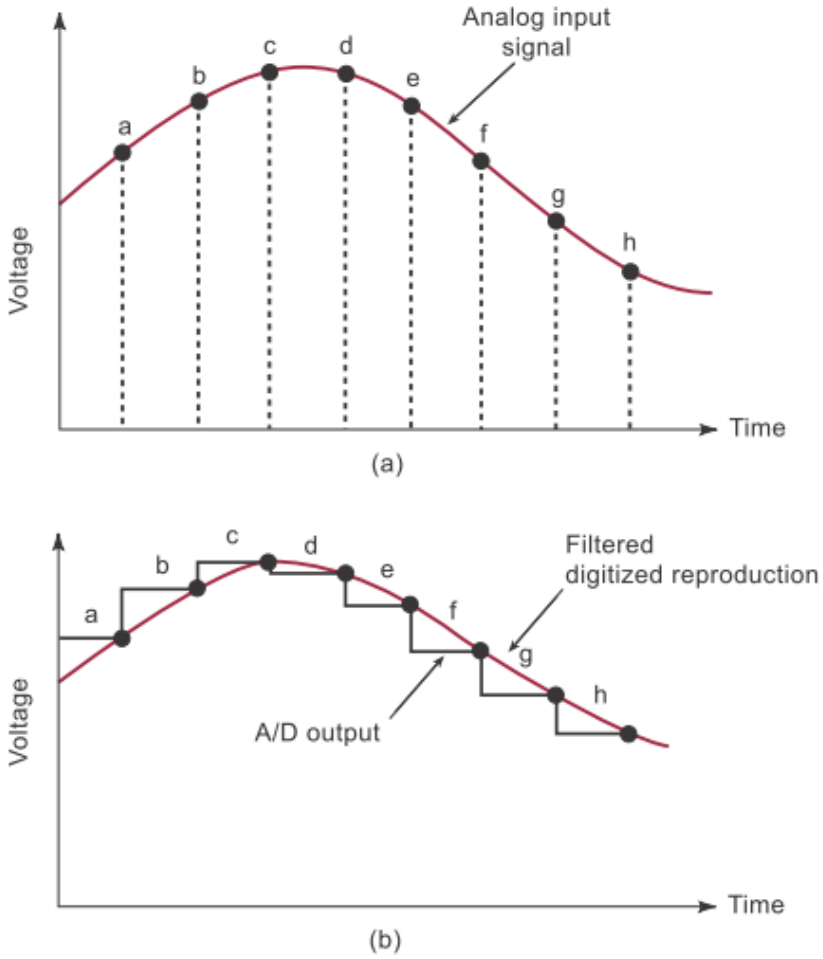
Biệt danh là hiện tượng xảy ra khi tần số lấy mẫu thấp hơn yêu cầu theo quy tắc Nyquist. Ví dụ trong Hình 9.10 minh họa dạng sóng có tần số 1,9kHz trong khi nó được lấy mẫu 2kHz. Nếu kết nối các điểm lấy mẫu, rõ ràng tín hiệu có tần số hoàn toàn khác thực tế.

Mạch lấy mẫu và giữ

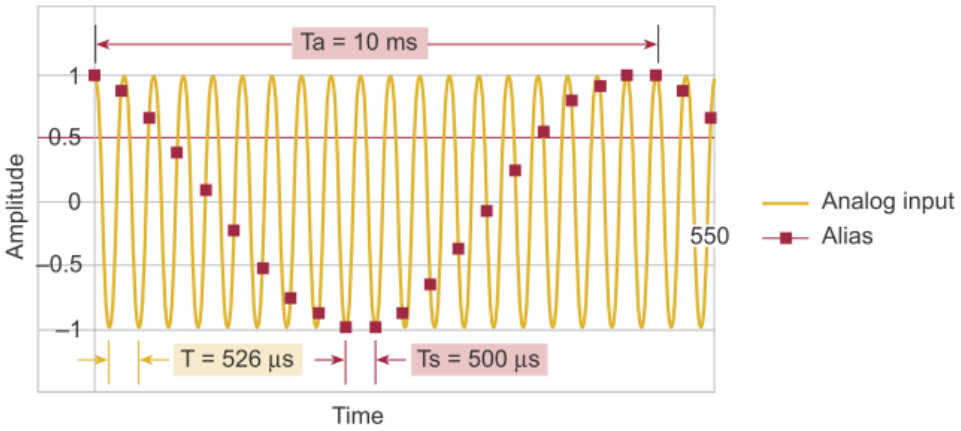
Trong quá trình biến đổi, tín hiệu tại lối vào ADC có thể thay đổi khiến kết quả bị ảnh hưởng. Để quá trình biến đổi ổn định, mạch giữ và lấy mẫu (sample-and-hold) được sử dụng để giữ mức tín hiệu. Mạch S/H sử dụng mạch khuyếch đại đệm để nạp tụ giữ C_h. Tín hiệu điều khiển cấp tín hiệu lấy mẫu và tụ sẽ giữ mức tín hiệu trước khi đưa vào ADC. Quá trình được minh họa trong Hình 9.11.

Thời gian tín hiệu điều khiển đóng mạch lấy mẫu còn được gọi là thời gian thu thập. Nó phụ thuộc vào giá trị tụ giữ và đặc tính mạch S/H.

Xử lý tín hiệu số (digital signal processing) là một dạng đặc biệt của bộ vi xử lý có tính năng chuyên biệt xử lý tính toán chuỗi dữ liệu số. Dữ liệu số thường được đưa vào DSP thông qua ADC. Ứng dụng chính của DSP là lọc và xử lý tín hiệu tương tự sau biến đổi. DSP sẽ được trình



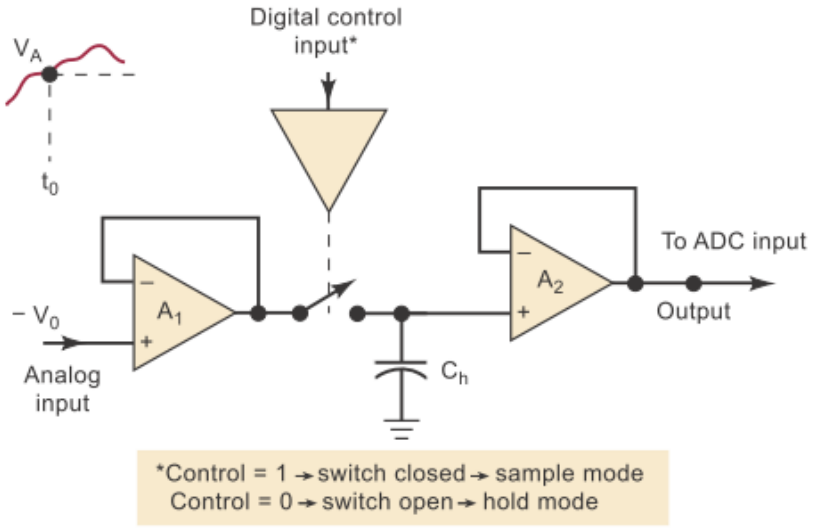
Hình 9.9 Thu thập dữ liệu và tái dựng tín hiệu tương tự



Hình 9.10 Tín hiệu biệt danh do lấy mẫu sai tần số

bày trong một môn học khác, tuy nhiên các thành phần cơ bản của DSP đều được xây dựng dựa trên kiến thức về điện tử số.

DSP có rất nhiều ứng dụng, ví dụ, nó có khả năng thực hiện bộ lọc một cách linh hoạt nhờ lập trình mà không phải thay đổi mạch lọc như thực hiện bằng mạch RLC. DSP được áp dụng nhiều



Hình 9.11 Sơ đồ khối mạch lấy mẫu và giữ

trong các hệ thống xử lý âm thanh, hình ảnh, biến đổi tín hiệu, mã hóa truyền thông,.... Ngoài ra, DSP còn cho phép thực hiện các giải pháp số thay cho những phương pháp tương tự trước kia.

10

Thiết Bị Nhớ

10.1 Giới thiệu

Lợi thế của hệ thống số đối với hệ thống tương tự là khả năng lưu trữ thông tin và dữ liệu trong thời gian dài. Khả năng lưu trữ giúp các hệ thống số linh hoạt và có thể đáp ứng với nhiều điều kiện làm việc khác nhau.

Một trong các thành phần nhớ chính là flip-flop. Các thanh ghi tạo thành từ FF là thiết bị nhớ tốc độ cao và được sử dụng nhiều trong máy tính. Ngoài ra, dữ liệu số còn có thể được lưu giữ dưới dạng điện tích trên các tụ. Có rất nhiều loại thiết bị nhớ được phân loại dựa trên nguyên tắc chế tạo, hoạt động và ứng dụng.

10.2 Một số thuật ngữ

Để thuận tiện cho quá trình tìm hiểu về thiết bị nhớ. Một số thuật ngữ cơ bản sẽ được trình bày:

Tế bào nhớ là thành phần lưu trữ một bit riêng lẻ (0 hoặc 1). FF hoặc tụ là ví dụ của tế bào nhớ.

Từ nhớ là một nhóm các tế bào nhớ được gom lại theo một yêu cầu cụ thể. **Byte** là thuật ngữ chỉ nhóm 8 bit.

Dung lượng dùng để chỉ rõ bao nhiêu bit có thể được lưu trữ trong một thiết bị nhớ. Thông thường, thuật ngữ '1K' được dùng để chỉ $1024 = 2^{10}$ đơn vị.

Mật độ là đại lượng chỉ số bit trên một đơn vị không gian.

Địa chỉ là số chỉ vị trí của từ nhớ trong bộ nhớ. Mỗi từ nhớ có một địa chỉ riêng.

Phép đọc là hành động chuyển từ nhớ nhị phân từ một địa chỉ xác định sang thiết bị nhớ khác.

Phép ghi là hành động từ nhớ mới được đặt vào vị trí cụ thể trên bộ nhớ.

Thời gian truy cập cũng là thông số đo tốc độ hoạt động của bộ nhớ. Đây là thời gian cần thiết để thực hiện xong phép đọc.

Bộ nhớ bay hơi chỉ chung các thiết bị nhớ cần năng lượng điện để lưu trữ thông tin. Nếu ngừng cấp điện, thông tin sẽ bị mất.

Bộ nhớ truy cập ngẫu nhiên (Random Access Memory - RAM) có thời gian truy xuất như nhau bất kể địa chỉ của từ nhớ.

Bộ nhớ truy cập tuần tự (Sequential Access Memory - SAM) có thời gian truy xuất khác nhau phụ thuộc địa chỉ từ nhớ.

Bộ nhớ đọc/ghi (Read/Write Memory) cho phép ghi và đọc thực hiện như nhau.

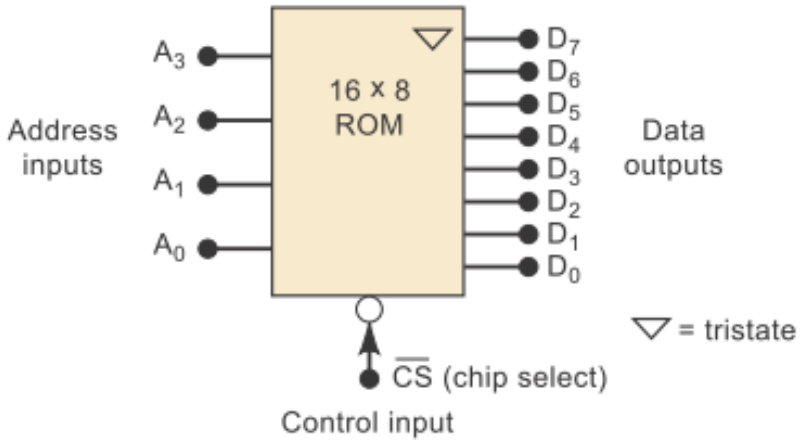
Bộ nhớ chỉ đọc (Read Only Memory - ROM) thường chỉ cho phép ghi một lần. Sau đó, thông tin chỉ có thể được đọc ra.

Bộ nhớ tĩnh cho phép thông tin được lưu trữ liên tục khi nguồn điện được cung cấp. Trong khi đó **Bộ nhớ động** đòi hỏi thông tin phải được ghi lại nhiều lần cho dù đang cung cấp nguồn điện.

10.3 ROM

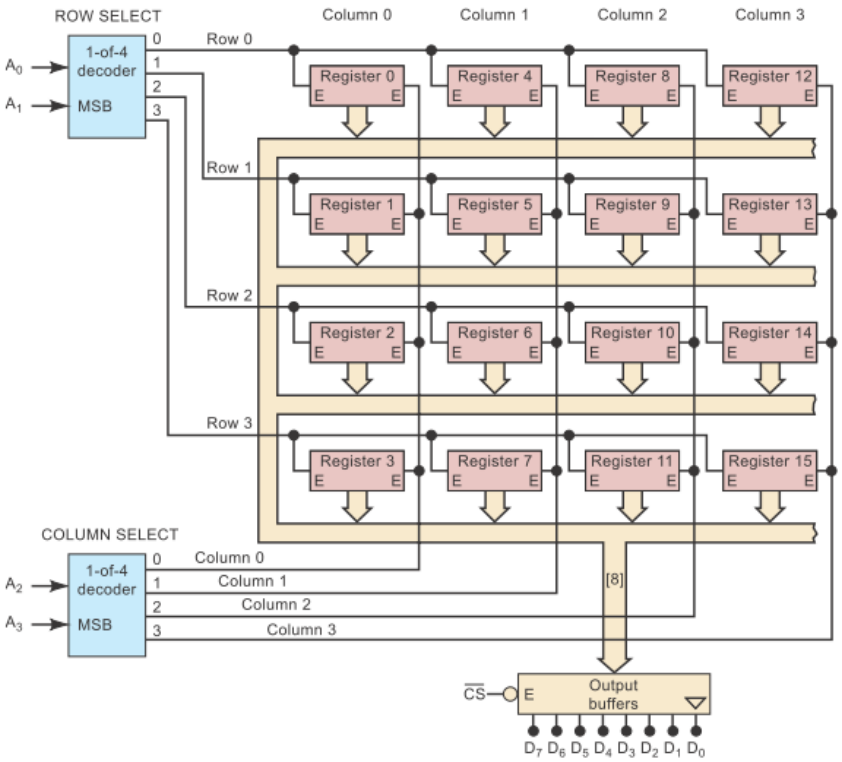
Cơ sở

Hình 10.1 minh họa một ROM với ba luồng tín hiệu: địa chỉ, đầu vào điều khiển và dữ liệu lối ra.



Hình 10.1 Sơ đồ khối ROM

Kiến trúc của ROM khá phức tạp và được đơn giản hóa trong Hình 10.2.



Hình 10.2 Kiến trúc tổng quát của ROM

Các loại ROM

MROM (Mask ROM) có thông tin được lưu trữ ngay tại thời điểm sản xuất. Vì chi phí sản xuất đơn lẻ rất đắt nên MROM chỉ được sản xuất hàng loạt. MROM thường có lối ra ba trạng thái cho phép nó có thể được sử dụng chung hệ thống đường dây (bus).

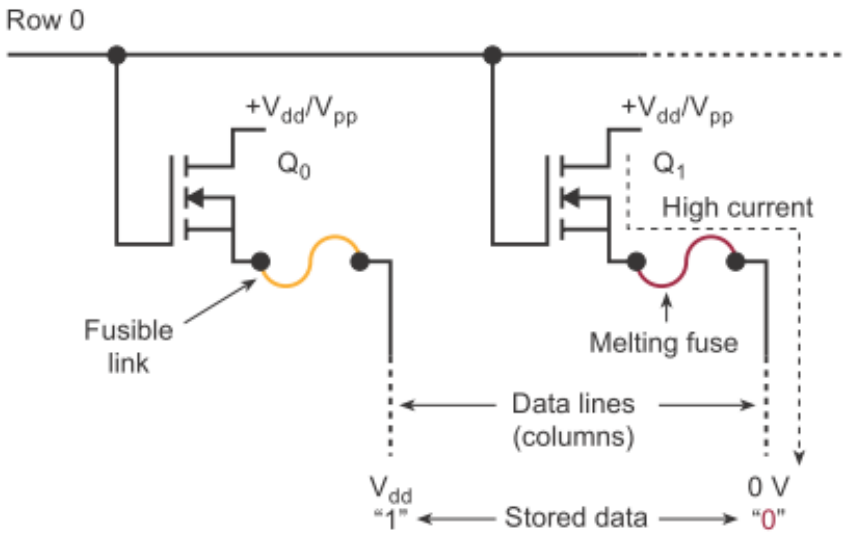
PROM (Programmable ROM) cho phép người dùng lập trình nạp dữ liệu thay vì tại nhà máy sản xuất. Một khi người dùng nạp dữ liệu, PROM chỉ cho phép đọc nên còn được gọi tên khác là OTP ROM. Hình 10.3 minh họa PROM với đường dây có thể "đốt" để ghi dữ liệu bởi người dùng.

EPROM (Erasable PROM) cũng có thể được nạp dữ liệu bởi người dùng. Tuy nhiên, nó cho phép người dùng xóa và ghi lại dữ liệu khi cần thiết. Hình 10.4 minh họa một EPROM có thể được xóa bởi tia cực tím.

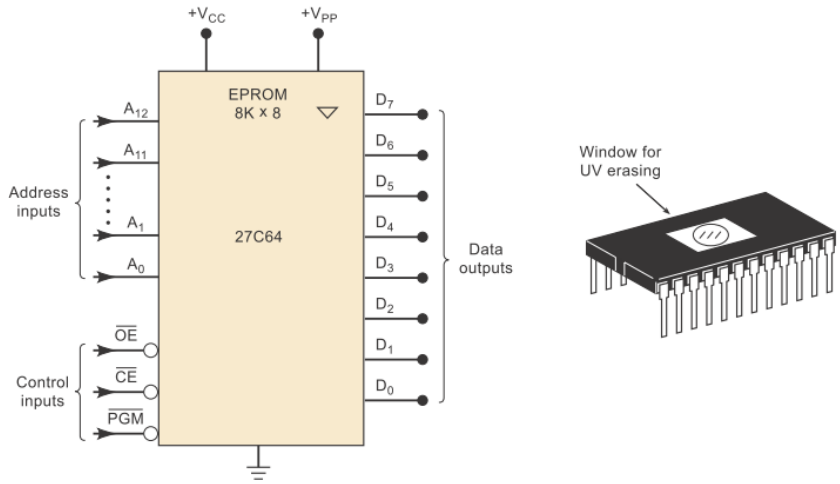
EEPROM là cải tiến của EPROM cho phép người dùng xóa dữ liệu bằng điện thay vì tia UV. Ngoài khả năng xóa dữ liệu nhanh chóng mà không cần dụng cụ đặc biệt, EEPROM còn cho phép tác động từng byte dữ liệu riêng lẻ.

CD-ROM (Compact disk ROM) là một dạng đặc biệt của ROM. CD-ROM có nhiều loại khác nhau, khả năng viết xóa và dung lượng cũng khác nhau.

Bộ nhớ Flash cải tiến từ EEPROM. EPROM có tốc độ truy xuất cao, mật độ cao và giá thành rẻ. Tuy nhiên, quá trình xóa và nạp lại dữ liệu đòi hỏi phải cách ly khỏi hệ thống. EEPROM có



Hình 10.3 Nguyên tắc nạp dữ liệu PROM



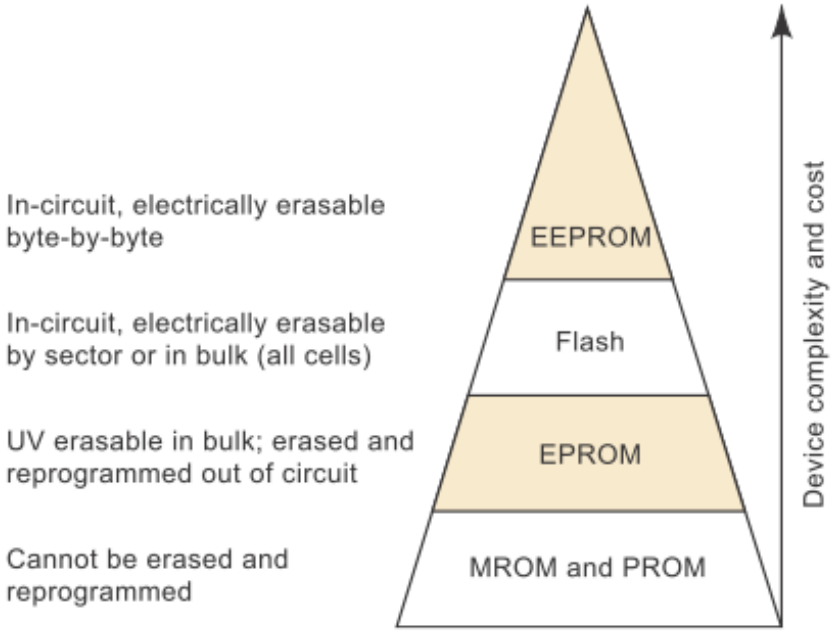
Hình 10.4 IC EPROM 27C64

khả năng nạp xóa trên mạch và tác động từng byte riêng lẻ nhưng có mật độ thấp và giá thành cao.

Bộ nhớ flash cho phép nạp xóa trên mạch, giá thành rẻ và mật độ cao, đồng thời tốc độ truy xuất khá cao. Về cấu trúc, bộ nhớ flash có tế bào nhớ khá giống EPROM. Tên gọi flash có được là nhờ tốc độ và khả năng nạp xóa dữ liệu. Hình 10.5 so sánh độ phức tạp và giá thành các loại ROM.

Ứng dụng ROM

Trong các hệ thống điện tử hiện nay, vi điều khiển đóng vai trò rất quan trọng. Chương trình hoạt động của chúng được nạp vào ROM. Hầu hết các vi điều khiển hiện nay tích hợp bộ nhớ trong. Ngoài ra, một số vi điều khiển còn hỗ trợ một vùng EEPROM để lưu trữ thông tin đặc biệt.



Hình 10.5 Đặc trưng các loại bộ nhớ ROM

Nhu cầu truyền tải dữ liệu giúp ROM ngày càng phát triển. Hầu hết các thiết bị điện tử giải trí đều sử dụng bộ nhớ flash vì tính năng đặc biệt của nó.

Trong các hệ thống máy tính, chương trình hoạt động thường được lưu trữ ở bộ nhớ ngoài. Khi cấp nguồn, một chương trình khởi động (bootstrap) được lưu trong ROM sẽ thực hiện những thao tác đầu tiên giúp hệ thống khởi động (booting).

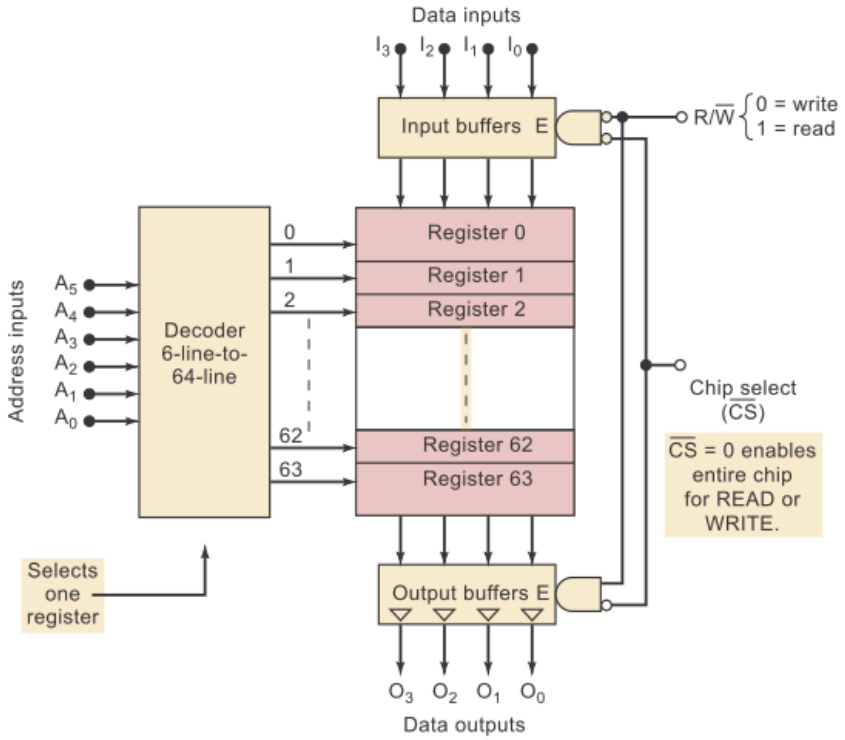
ROM còn được sử dụng để lưu trữ dữ liệu bất biến. Nó có thể là bảng dữ liệu, bảng mã hóa và giải mã, ... Trong các máy phát hàm (phát xung), ROM được dùng để lưu trữ giá trị tương ứng của các loại dạng sóng khác nhau. Với ưu thế giá thành rẻ, tốc độ cao, công suất thấp so với đĩa từ giúp các loại bộ nhớ dần trở thành thiết bị lưu trữ bổ sung cho các hệ thống lớn.

10.4 RAM

Cơ sở

RAM thường được dùng để lưu trữ tạm thời dữ liệu và chương trình. Điểm bất lợi chính của RAM là mất dữ liệu khi ngắt nguồn điện. Một số CMOS RAM sử dụng một lượng điện rất nhỏ trong chế độ chờ nhằm lưu trữ thông tin khi nguồn điện chính bị cắt. Ngược lại, lợi thế của RAM là khả năng đọc và ghi dễ dàng.

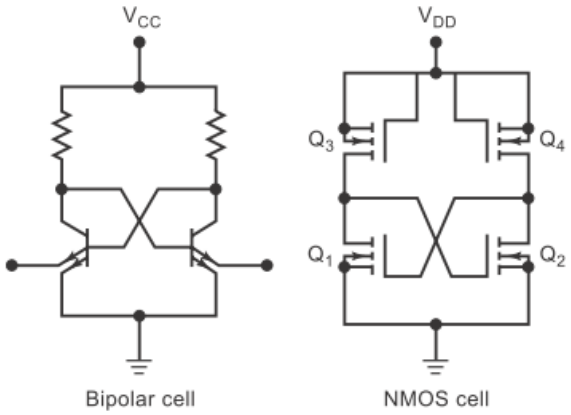
Cũng như ROM, có thể xem RAM được cấu thành từ các thanh ghi. Mỗi thanh ghi lưu giữ một từ nhớ và có địa chỉ riêng. Hình 10.6 minh họa kiến trúc được đơn giản hóa của một RAM.



Hình 10.6 Tổ chức của một RAM 64x4

Phân loại

RAM tĩnh (static RAM) lưu trữ dữ liệu khi còn nguồn điện. Về cơ bản, tế bào RAM tĩnh là các FF. SRAM có tốc độ cao, mạch đơn giản và được chế tạo bởi nhiều công nghệ khác nhau. Hình 10.7 minh họa SRAM công nghệ NMOS.



Hình 10.7 Tế bào nhớ SRAM

RAM động (dynamic RAM) đòi hỏi dữ liệu phải được thường xuyên tái nạp trong quá trình hoạt động. Điều này tăng sự phức tạp cho các mạch hỗ trợ đi kèm. Đây là một điểm bất lợi so với SRAM. Tuy nhiên, RAM động có dung lượng cao, tiêu thụ công suất thấp và tốc độ hoạt động tương đối cao.

Hầu hết các bộ nhớ của máy tính cá nhân đều sử dụng DRAM vì dung lượng cao và tiêu hao ít năng lượng. Tuy nhiên, vẫn có một lượng nhỏ SRAM giành riêng cho tác vụ đòi hỏi tốc độ xử lý cao.

Hiện nay, công nghệ chế tạo DRAM rất phát triển. Nhiều loại DRAM được ra đời với tính năng ngày càng cao, ví dụ như: FPM DRAM, EDO DRAM, SDRAM, DDRSDRAM, SLDRAM, DRDRAM, . . .

Tài Liệu Tham Khảo

[1] Tocci, R.J., Widmer, N.S. and Moss, G.L., 2011. Digital systems: principles and applications.

[2] Maini, A.K., 2007. Digital electronics: principles, devices and applications. John Wiley & Sons.

[3] Giáo trình kỹ thuật số và điện tử số khác.