ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CƠ KHÍ

ଌ * ଌ

Tập bài giảng

HỆ THỐNG CƠ ĐIỆN TỬ 2

Biên soạn: PGS. TS. TRẦN XUÂN TÙY

ĐÀ NẴNG - 2007

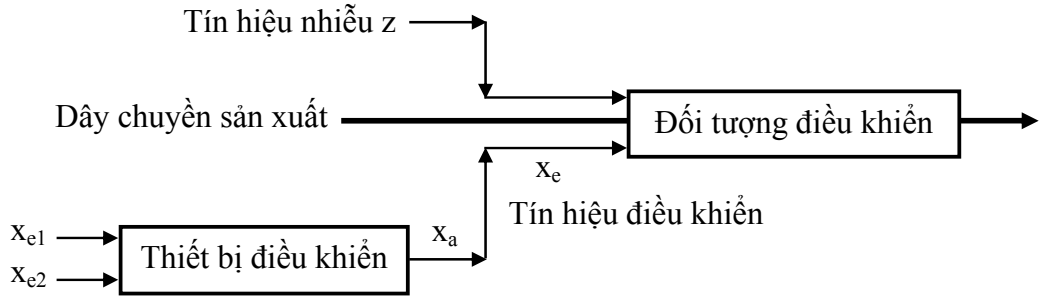
CHƯƠNG 1: **ĐIỀU KHIỂN LOGIC**

1.1. KHÁI NIỆM QUÁ TRÌNH ĐIỀU KHIỂN

"Điều khiển" là quá trình của một hệ thống, trong đó dưới tác động của một hay nhiều đại lượng vào thi đại lượng ra sẽ thay đổi theo một quy luật nhất định.

1.1.1. Hệ thống điều khiển

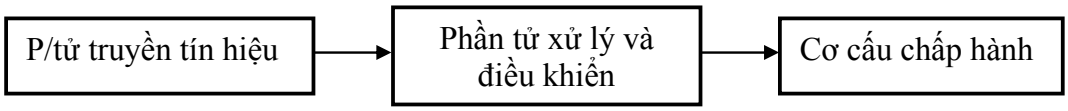
Hệ thống điều khiển bao gồm thiết bị điều khiển và đối tượng điều khiển, được thể hiện như sơ đồ hình 1.1.



Hình 1.1. Sơ đồ hệ thống điều khiển

Đối tượng điều khiển: Thiết bị, máy móc trong kỹ thuật.

Thiết bị điều khiển: Các phần tử truyền tín hiệu, phần tử xử lý và điều khiển, cơ cấu chấp hành, thể hiện như sơ đồ hình 1.2.



Hình 1.2. Các phần tử của hệ thống điều khiển

Trong đó:

Phần tử truyền tín hiệu: nhận những giá trị của đại lượng vậy lý và là đại lượng vào...

Ví dụ: công tắc, nút bấm, công tắc hành trình, cảm biến, …

Phần tử xử lý tín hiệu và điều khiển: xử lý tín hiệu vào theo một quy tắc logic, làm thay đổi trạng thái của phần tử điều khiển, điều khiển dòng năng lượng theo yêu cầu để làm thay đổi trạng thái của cơ cấu chấp hành.

Ví dụ: van đảo chiều, van chắn (van một chiều, van logic OR, van logic AND), van tiết lưu, van áp suất, rơle, phần tử khuếch đại, phần tử chuyển đổi tín hiệu, …

Cơ cấu chấp hành: thay đổi trạng thái của đối tượng điều khiển, là đại lượng ra của mạch điều khiển.

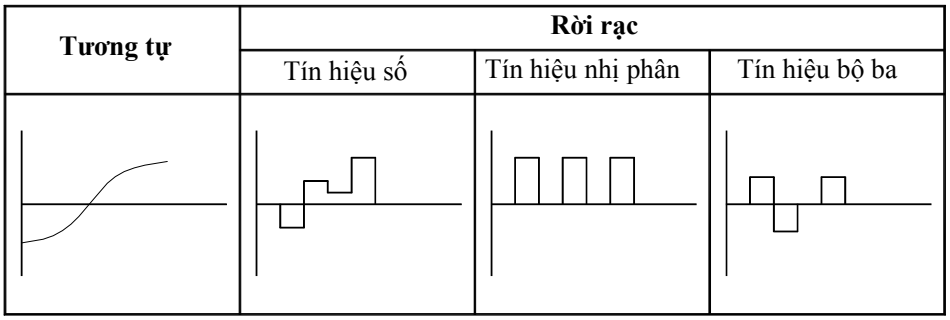
Ví dụ: xilanh, động cơ, bộ biến đổi áp lực.v.v.

Tín hiệu điều khiển: đại lượng ra x_a của thiết bị điều khiển và đại lượng vào x_e của đối tượng điều khiển.

Tín hiệu nhiễu z: đại lượng được tác động từ bên ngoài vào hệ thống và gây ảnh hưởng xấu đến hệ thống điều khiển.

1.1.2. Các loại tín hiệu điều khiển

Thông tin (tín hiệu vào x_e và tín hiệu ra x_a) để cho mạch điều khiển hoạt động theo một quy luật định sẵn có thể thực hiện được như tín hiệu áp suất, giá trị áp suất được gọi là thông số tín hiệu. Tín hiêu tương tự (liên tục) và tín hiệu rờI rạc được thể hiện qua hình 1.3.

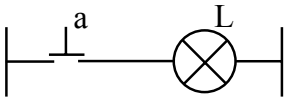


Hình 1.3. Phân loại tín hiệu

1.2. CÁC PHẦN TỬ LOGIC

Trong điều khiển logic có hai trạng thái, đó là trạng thái "0" và trạng thái "1".

Ví dụ 1:



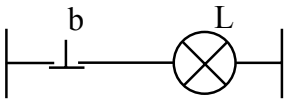
Nếu a = 0 thì L = 0

Nếu a = 1 thì L = 1

Ta có thể viết L = a

Trong đó: a là nút ấn thường mở; L là đèn tín hiệu.

Ví dụ 2:



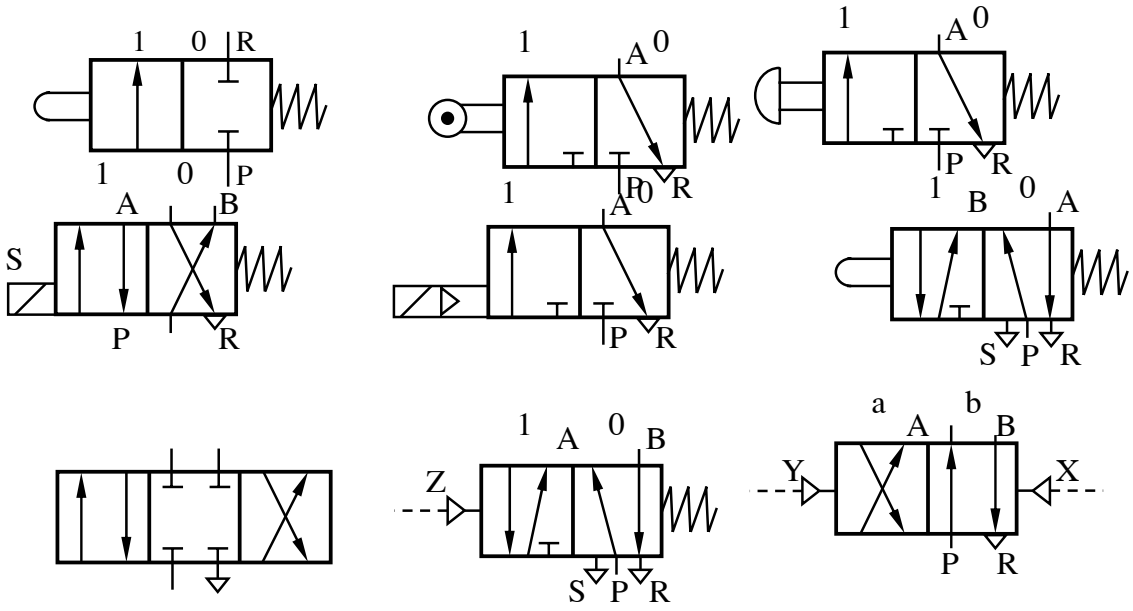
Nếu b = 0 thì L = 1

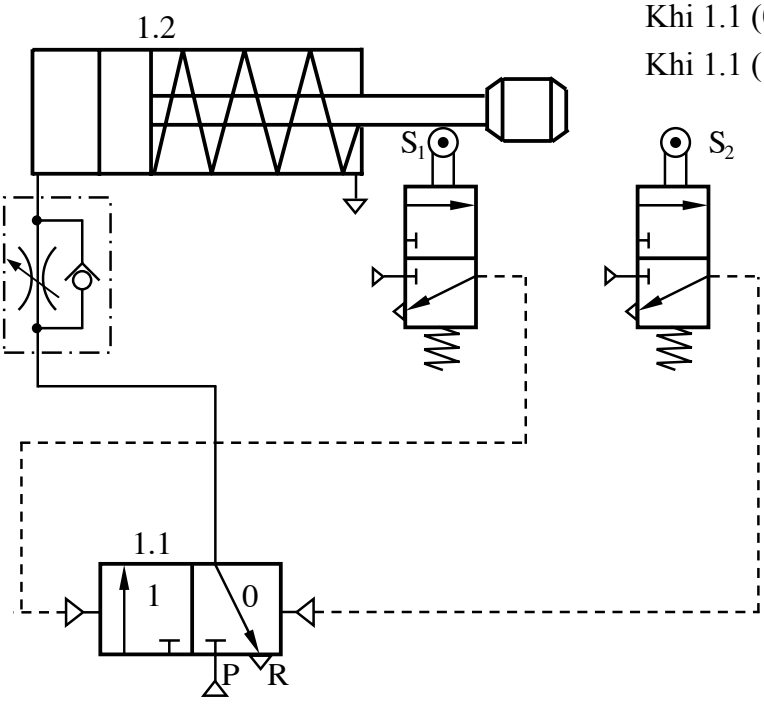
Nếu b = 1 thì L = 0

Ta có thể viết $L = \bar{b}$

Trong đó: b là nút ấn thường đóng; $L = \bar{b}$ là phủ định của b

Ví dụ 3: Một phần tử và sơ đồ mạch điều khiển logic khí nén thể hiện như hình 1.3.

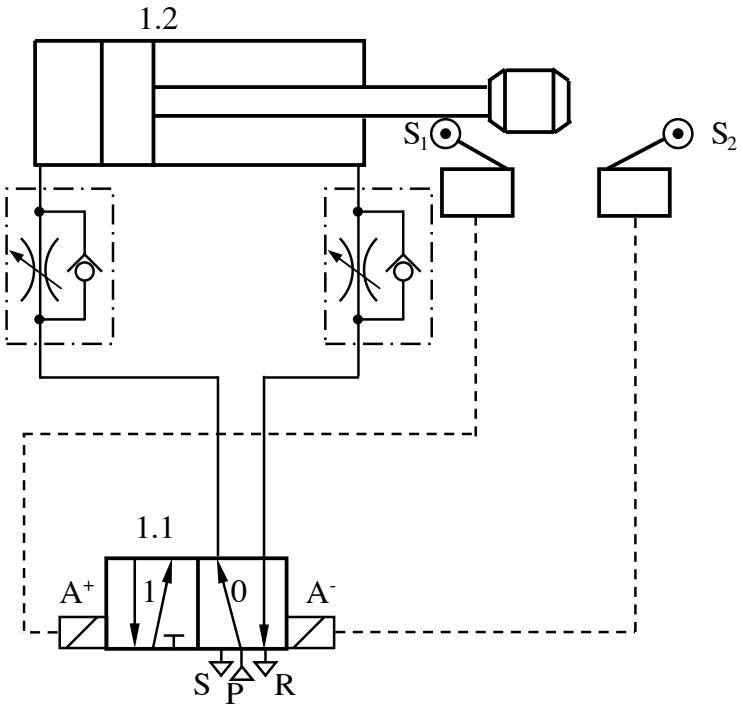




Khi 1.1 (0) ⇒ 1.2 lùi về

Khi 1.1 (1) ⇒ 1.2 duỗi thẳng

Hình 1.4. Sơ đồ logic khí nén



Hình 1.5. Sơ đồ logic điện khí nén

Khi 1.1 (0) (có tín hiệu A^-) $\Rightarrow$ 1.2 lùi về

Khi 1.1 (1) (có tín hiệu A^+) $\Rightarrow$ 1.2 duỗi thẳng

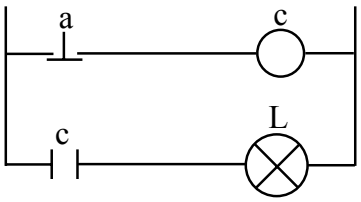
Các phần tử logic cơ bản được ký hiệu như ở bảng sau (tiêu chuẩn EU và USA):

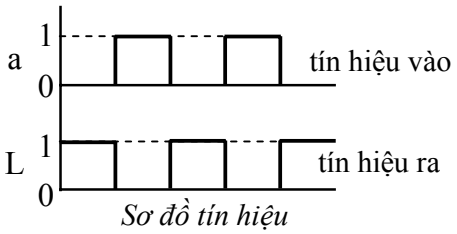
Số TT	*Ký hiệu*	*Tên gọi*
1	1 — *Theo tc EU*; *Theo tc USA*	NOT
2	& — *Theo tc EU*; *Theo tc USA*	AND
3	& — *Theo tc EU*; *Theo tc USA*	NAND
4	≥1 — *Theo tc EU*; *Theo tc USA*	OR
5	≥1 — *Theo tc EU*; *Theo tc USA*	NOR
6	=1 — *Theo tc EU*; *Theo tc USA*	XOR (EXC-OR)

1.2.1. Phần tử logic NOT (Phủ định)

Ta có phương trình logic $L = \bar{a}$

Phần tử NOT được biểu diễn: khi ấn nút a, rơle c mất điện ⇒ bóng đèn L tắt; ngược lại khi nhả nút a, rơle c có điện ⇒ bóng đèn L sáng.



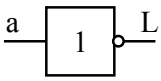


Sơ đồ tín hiệu

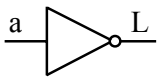
Bảng chân lý

a	L
0	1
1	0

Ký hiệu



Theo tc EU

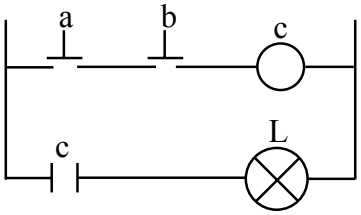


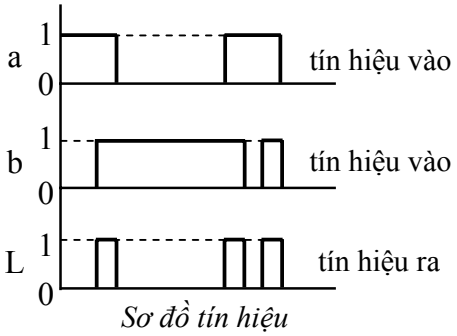
Theo tc USA

1.2.2. Phần tử AND (Và)

Phương trình logic L = a.b

Phần tử AND (và) được biểi diễn: khi ấn nút a đồng thời ấn nút b, rơle c có điện ⇒ bóng đèn L sáng.



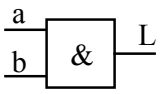


Sơ đồ tín hiệu

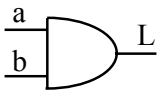
Bảng chấn lý

a	b	L
0	0	0
0	1	0
1	0	0
1	1	1

Ký hiệu



Theo tc EU

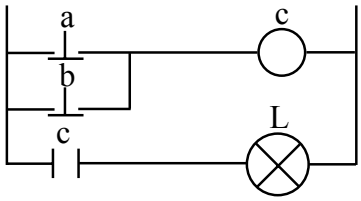


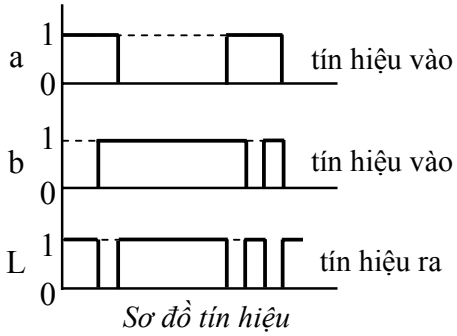
Theo tc USA

1.2.3. Phần tử logic NAND (Và - Không)

Phương trình logic $L = \overline{a.b} = \bar{a} + \bar{b}$

Phần tử logic NAND được biểu diễn: khi ấn nút a đồng thời ấn nút b, rơle c mất điện ⇒ bóng đèn L tắt.



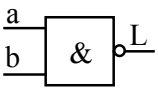


Sơ đồ tín hiệu

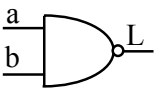
Bảng chân lý

a	b	L
0	0	1
0	1	1
1	0	1
1	1	0

Ký hiệu



Theo tc EU

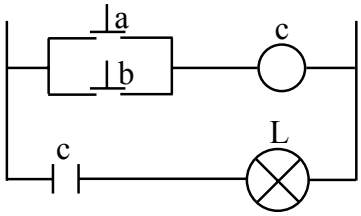


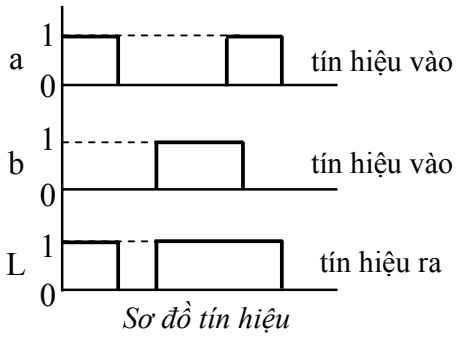
Theo tc USA

1.2.4. Phần tử logic OR (Hoặc)

Phương trình logic L = a + b

Phần tử hoặc được biểu diễn: khi ấn nút a hoặc b, rơle c có điện ⇒ bóng đèn L sáng.



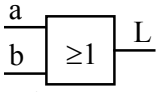


Sơ đồ tín hiệu

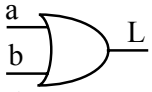
Bảng chân lý

a	b	L
0	0	0
0	1	1
1	0	1
1	1	1

Ký hiệu



Theo tc EU

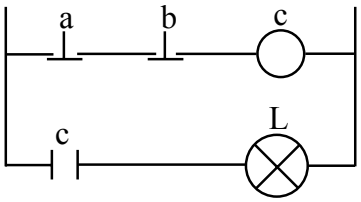


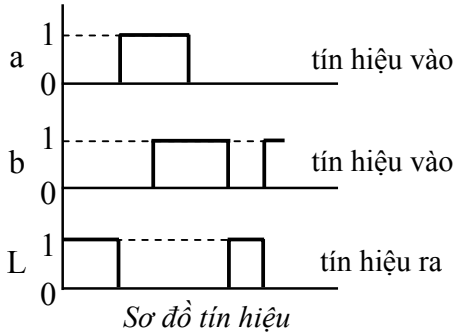
Theo tc USA

1.2.5. Phần tử logic NOR (Hoặc - Không)

Phương trình logic $L = \overline{a + b} = \overline{a}.\overline{b}$

Phần tử logic NOR được biểu diễn: khi một trong 2 nút ấn a hoặc b được thực hiện, thì đèn L tắt. Đèn L sang khi không có tín hiệu nào thực hiện.



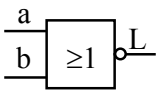


Sơ đồ tín hiệu

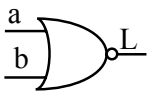
Bảng chân lý

a	b	L
0	0	1
0	1	0
1	0	0
1	1	0

Ký hiệu



Theo tc EU

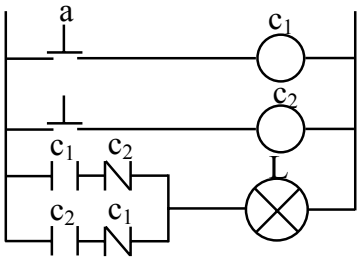


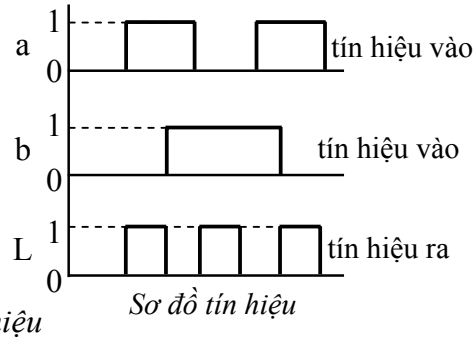
Theo tc USA

1.2.6. Phần tử logic XOR (EXC - OR)

Phương trình logic $L = a.\bar{b} + \bar{a}.b$

Phần tử logic XOR được biểu diễn: khi ấn nút a hoặc b, rơle c_1 hoặc c_2 có điện $\Rightarrow$ đèn L sáng; khi ấn cả 2 nút đồng thời $\Rightarrow$ đèn L tắt.



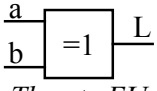


Sơ đồ tín hiệu

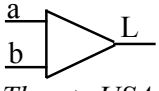
Bảng chân lý

a	b	L
0	0	0
0	1	1
1	0	1
1	1	0

Ký hiệu



Theo tc EU

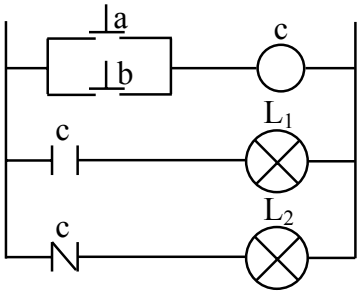


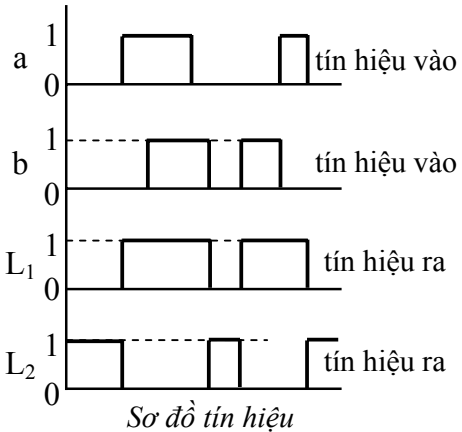
Theo tc USA

1.2.7. Phần tử logic OR/NOR

Phương trình logic: $L_1 = a + b$; $L_2 = \overline{a + b} = \bar{a}.\bar{b}$

Phần tử OR/NOR có hai tín hiệu ra L_1, L_2 được biểu diễn: khi chưa ấn nút a hoặc b, rơle c chưa có điện ⇒ bóng đèn L_1 tắt, L_2 sáng; khi ấn nút a hoặc b, rơle c có điện ⇒ bóng đèn L_1 sáng, L_2 tắt.



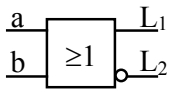


Sơ đồ tín hiệu

Bảng chân lý

a	b	L_1	L_2
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

Ký hiệu

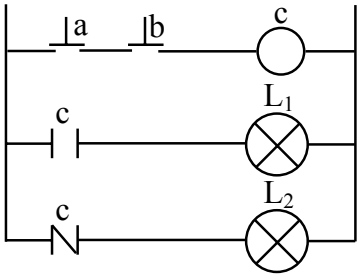


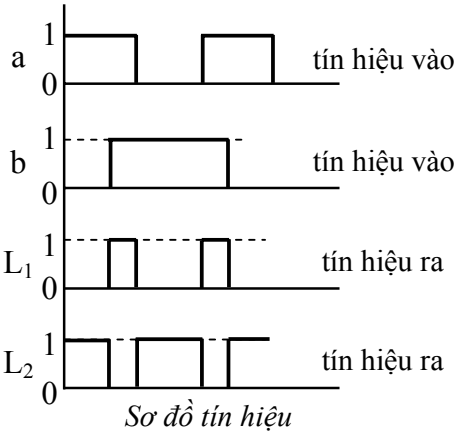
Theo tc EU

1.2.8. Phần tử logic AND - NAND

Phương trình logic: $L_1 = a.b$; $L_2 = \overline{a.b} = \bar{a} + \bar{b}$

Phần tử logic AND - NAND có hai tín hiệu ra L_1, L_2 và được biểu diễn: khi chưa tác động nút ấn a và b ⇒ L_1 tắt, L_2 sáng; khi ấn a đồng thời ấn b, rơle c có điện ⇒ S_1 sáng, L_2 tắt.



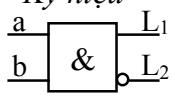


Sơ đồ tín hiệu

Bảng chân lý

a	b	L_1	L_2
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Ký hiệu



Theo tc EU

1.3. LÝ THUYẾT ĐẠI SỐ BOOLE

Trong kỹ thuật điều khiển, giá trị của các tín hiệu vào và tín hiệu ra được viết dưới dạng biến số của đại số Boole.

1.3.1. Các quy tắc cơ bản của đại số Boole

(ta có thể quy ước để thuận tiện việc tính toán: trong lý thuyết đại số Boole phần tử logic AND là "."hoặc "$\wedge$"; phần tử logic OR là "+" hoặc "$\vee$")

Phép toán liên kết AND (và): L = a.b.c (hoặc có thể viết $L = a \wedge b \wedge c$)

Cụ thể:

$1.1.1 = 1 \ (1 \wedge 1 \wedge 1 = 1)$

$1.0.0 = 0 \ (1 \wedge 0 \wedge 0 = 0)$

$1.1.0 = 0 \ (1 \wedge 1 \wedge 0 = 0)$

$1.0.1 = 0 \ (1 \wedge 0 \wedge 1 = 0)$

$0.1.1 = 0 \ (0 \wedge 1 \wedge 1 = 0)$

$0.0.0 = 0 \ (0 \wedge 0 \wedge 0 = 0)$

Phép toán liên kết OR (hoặc): L = a +b +c (hoặc có thể viết $L = a \vee b \vee c$)

Cụ thể:

$1+1+1 = 1 \ (1 \vee 1 \vee 1 = 1)$

$1+0+0 = 1 \ (1 \vee 0 \vee 0 = 1)$

$1+1+0 = 1 \ (1 \vee 1 \vee 0 = 1)$

$0+1+1 = 1 \ (0 \vee 1 \vee 1 = 1)$

$1+0+1 = 1 \ (1 \vee 0 \vee 1 = 1)$

$0+0+0 = 0 \ (0 \vee 0 \vee 0 = 0)$

Phép toán liên kết NOT (phủ định): $S = \bar{a}$

Cụ thể:

$\bar{0} = 1$

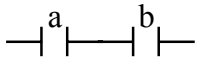
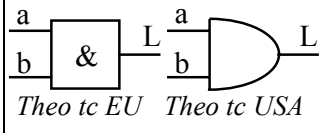
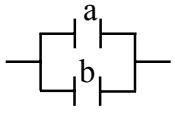
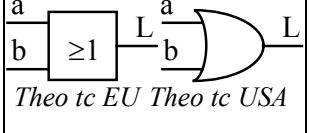
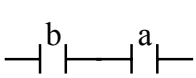
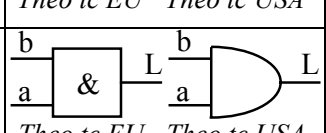
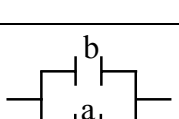
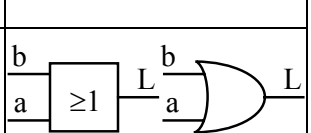
$\bar{1} = 0$

a. Quy tắc hoán vị: Các toán tử a và b có thể hoán vị cho nhau

$L_1 = a.b = b.a \ (S = a \wedge b = b \wedge a)$

$L_2 = a + b = b + a \ (S = a \vee b = b \vee a)$

Ta có thể biểu diễn như ở bảng dưới:

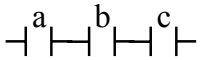
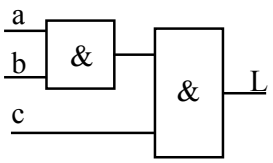
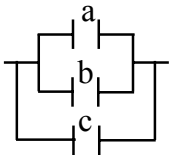
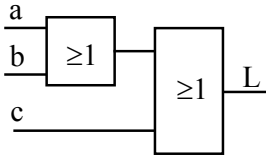
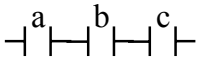
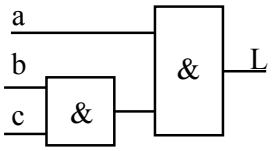
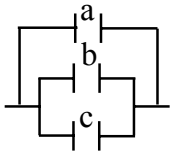
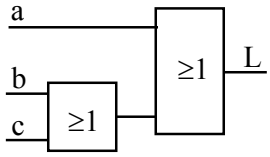
a.b = b.a		a + b = b + a	
Sơ đồ mạch điện	Sơ đồ logic	Sơ đồ mạch điện	Sơ đồ logic
			
			

b. Quy tắc kết hợp:

$L_1 = a.b.c = (a.b).c = a.(b.c)$ $\{L = a \wedge b \wedge c = (a \wedge b) \wedge c = a \wedge (b \wedge c)\}$

$L_2 = a + b + c = (a + b) + c = a + (b + c)$ $\{L = a \vee b \vee c = (a \vee b) \vee c = a \vee (b \vee c)\}$

Ta có thể biểu diễn như ở bảng dưới:

(a.b).c = a.(b.c)		(a + b) + c = a + (b + c)	
Sơ đồ mạch điện	Sơ đồ logic	Sơ đồ mạch điện	Sơ đồ logic
			
			

c. Quy tắc phân phối: Phép toán liên kết AND, OR và NOT được kết hợp với nhau

$L_1 = (a.b) + (c.d) = (a + c).(a + d).(b + c).(b + d)$

$L_2 = (a + b).(c + d) = (a.c) + (a.d) + (b.c) + (b.d)$

$L_3 = a.(b + c) = (a.b) + (a.c)$

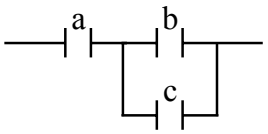
$L_4 = a + (b.c) = (a + b).(a + c)$

Ta có thể biểu diễn sơ đồ mạch điện và sơ đồ logic như sau (chỉ biểu diễn S_3, S_4):

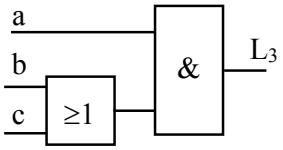
$L_3 = a.(b + c) = (a.b) + (a.c)$

$L_3 = a.(b + c)$

Sơ đồ mạch điện



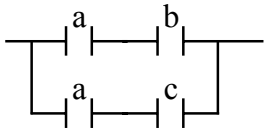
Sơ đồ logic



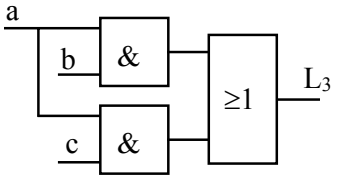
a	b	c	b.c	L_3
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

$L_3 = (a.b) + (a.c)$

Sơ đồ mạch điện



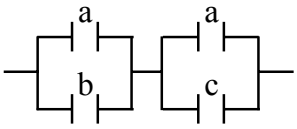
Sơ đồ mạch logic

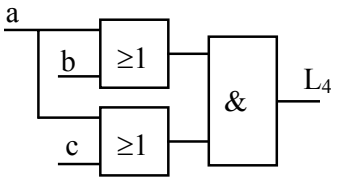


a	b	c	a.b	a.c	L_3
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

$L_4 = a + (b.c) = (a + b).(a + c)$

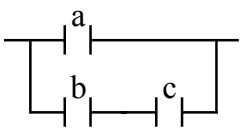
$L_4 = (a + b).(a + c)$

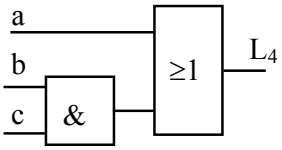




a	b	c	a+b	a+c	L_4
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	1	1	1
1	0	0	1	1	1
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

$L_4 = a + (b.c)$





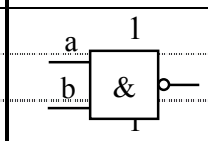
a	b	c	b.c	L_4
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

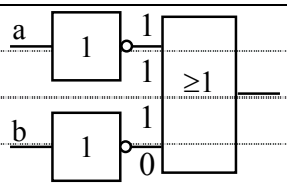
d. Quy tắc nghịch đảo (quy tắc Morgan)

Phép toán liên kết AND được chuyển đổi thành phép toán liên kết OR bằng phép toán phủ định NOT và phép toán liên kết OR được chuyển đổi thành phép toán liên kết AND bằng phép toán phủ định NOT:

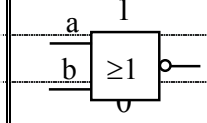
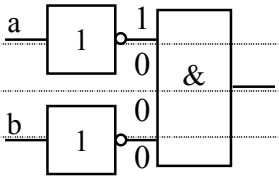
$\overline{a.b} = \overline{a} + \overline{b}$; $\overline{a.b.c} = \overline{a} + \overline{b} + \overline{c}$

a	b	$\overline{a}$	$\overline{b}$	a.b	$\overline{a.b}$	$\overline{a+b}$
0	0	1	1	0	1	1
0	1	1	0	0	1	0
1	0	0	1	0	1	0
1	1	0	0	1	0	0



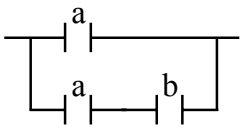


$\overline{a+b} = \bar{a}.\bar{b}$; $\overline{a+b+c} = \bar{a}.\bar{b}.\bar{c}$

a	b	$\bar{a}$	$\bar{b}$	a+b	$\overline{a+b}$	$\overline{a.b}$
0	0	1	1	0		
0	1	1	0	1		
1	0	0	1	1		
1	1	0	0	1		

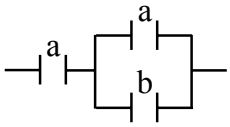
e. Quy tắc hấp thụ

$a + (a.b) = a$



a	b	a.b	a+(a.b)
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

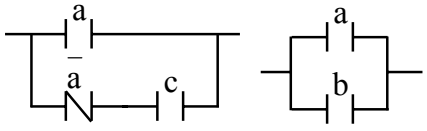
$a.(a + b) = a$



a	b	a + b	a.(a+b)
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	1

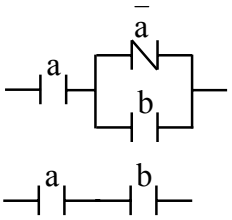
f. Quy tắc bù

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



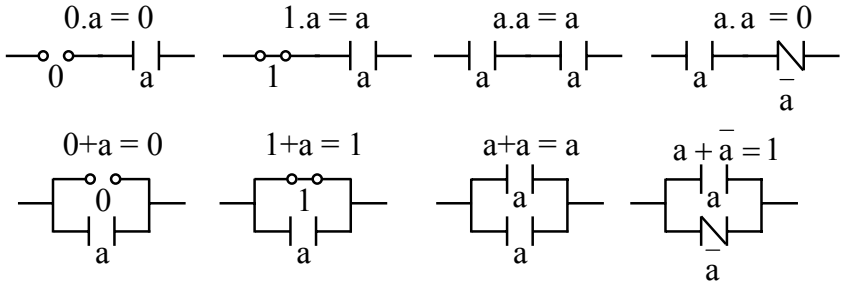
a	b	$\bar{a}.b$	$a + \bar{a}.b$	a+b
0	0	0	0	0
0	1	1	0	1
1	0	1	1	1
1	1	1	1	1

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



a	b	$\bar{a}+b$	$a.(\bar{a}+b)$	a.b
0	0	1	0	0
0	1	1	0	0
1	0	0	0	0
1	1	1	1	1

g. Quy tắc đơn giản các liên kết



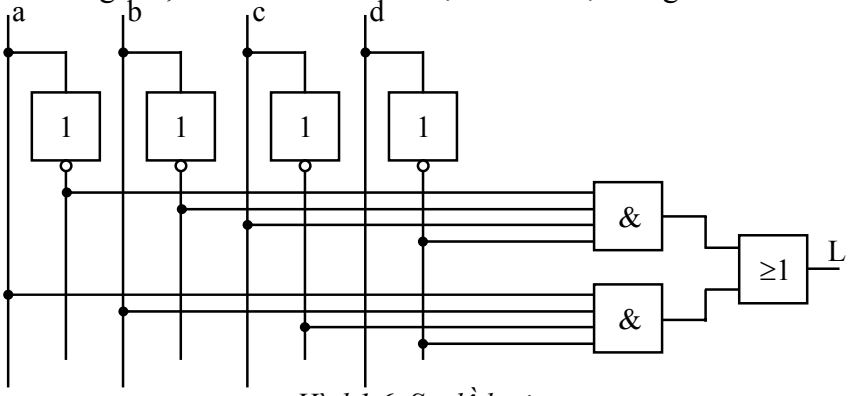
1.3.2. Ví dụ minh hoạ đại số Boole

Ví dụ 1: Từ phương trình logic sau đây

$$L=(\bar{a}.\bar{b}.c.\bar{d})+(a.b.\bar{c}.\bar{d})$$

Hãy thiết kế sơ đồ mạch logic, sao cho số phần tử logic ít nhất và sử dụng số phần tử logic đơn giản với số cổng vào càng ít càng tốt.

Từ phương trình logic S, ta có thể thiết kế được sơ đồ mạch logic như sau:



Hình1.6. Sơ đồ logic

Sơ đồ logic trên bao gồm: 4 phần tử NOT: $\bar{a},\bar{b},\bar{c},\bar{d}$

2 phần tử AND với 4 cổng vào

1 phần tử OR với 2 cổng vào

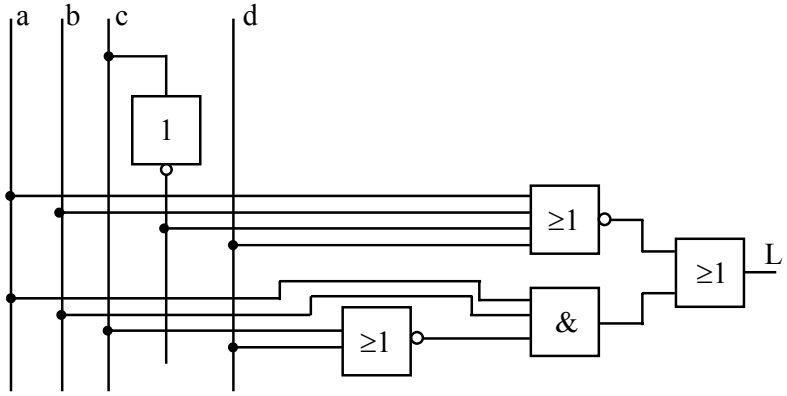
$\Rightarrow$ ta có 7 phần tử

Theo quy tắc Morgan, ta biến đổi như sau:

$$\bar{a}.\bar{b}.c.\bar{d}=\overline{a+b+\bar{c}+d}$$

Và $a.b.\bar{c}.\bar{d}=a.b.(\overline{c+d})$

Ta có: $L = \left(\overline{a + b + \bar{c} + d}\right) + \left(a.b.\left[\overline{c + d}\right]\right)$



Hình1.7. Sơ đồ logic

Sơ đồ mạch logic sau khi biến đổi gồm 5 phần tử:

1 phần tử NOT

1 phần tử NOR với 4 cổng vào

1 phần tử OR với 2 cổng vào

1 phân tử NOR với 2 cổng vào

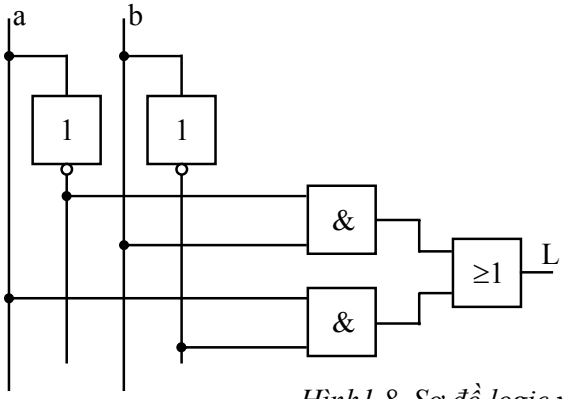
1 phần tử AND với 3 cổng vào

$\Rightarrow$ Như vậy sau khi biến đổi thì số phần tử sẽ ít hơn.

Ví dụ 2: Hãy đơn giản mạch điều khiển có phương trình logic sau đây:

$$L = \left(a.\bar{b}\right) + \left(\bar{a}.b\right)$$

Từ phương trình trên, ta có sơ đồ logic và bảng chân lý sau:



a	b	L
0	0	0
0	1	1
1	0	1
1	1	0

Hình1.8. Sơ đồ logic và bảng chân lý

Theo quy tắc phân phối, ta biến đổi như sau:

$$L = \left(a.\bar{b}\right) + \left(\bar{a}.b\right) = \left(a + \bar{a}\right).\left(a + b\right).\left(\bar{b} + \bar{a}\right).\left(\bar{b} + b\right)$$

Theo quy tắc đơn giản liên kết, ta có:

$$\left(a + \bar{a}\right) = 1 \text{ và } \left(\bar{b} + b\right) = 1$$

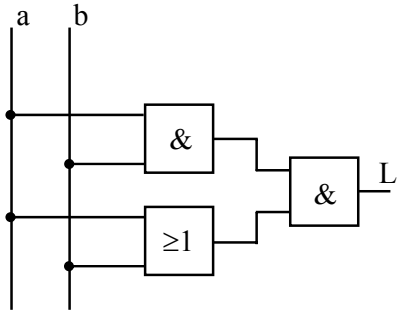
Như vậy phương trình được viết lại như sau:

$$L = \left(a + b\right).\left(\bar{b} + \bar{a}\right)$$

Theo quy tắc Morgan: $\left(\bar{b}+\bar{a}\right)=\overline{b.a}$

$\Rightarrow$ Phương trình logic đơn giản: $L=(a+b).\left(\overline{b.a}\right)$

Ta có sơ đồ mạch logic đơn giản với 3 phần tử:



a	b	L
0	0	0
0	1	1
1	0	1
1	1	0

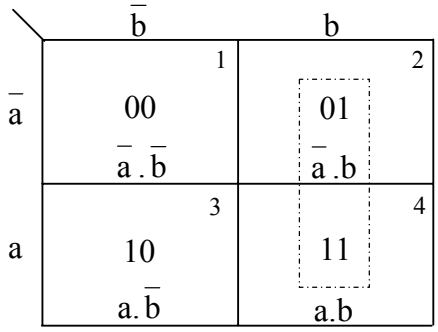
Hình1.9. Sơ đồ logic và bảng chân lý

1.4. BIỂU ĐỒ KARNAUGH

Để đơn giản mạch logic hay mạch công tác bằng quy tắc đại số Boole thì khá phức tạp. Vào năm 1953 nhà toán học Karnaugh (người Anh) đã phát triển một phương pháp giải bằng biểu diễn đồ thị, gọi là *biểu đồ Karnaugh.* Nhờ phương pháp biểu đồ Karnaugh mà ta có thể sử dụng ít quy tắc để đơn giản những phương trình logic phức tạp với nhiều biến.

Biểu đồ Karnaugh bao gồm nhiều khối và biểu diễn tất cả khả năng dạng phép hội tụ toàn phần. Dạng phép hội tụ toàn phần là phép toán liên kết AND, bao gồm tất cả các biến và phủ định của biến.

1.4.1. Biểu đồ Karnaugh với 2 biến



a	b	L
0	0	0
0	1	1
1	0	0
1	1	1

Các khối của dòng thứ nhất (1 và 2) gồm phủ định của biến a, khối của dòng thứ 2 (3 và 4) biến a.

Tương tự khối của cột thứ nhất (1 và 3) bao gồm phủ định của biến b, khối của cột thứ 2 (2 và 4) bao gồm biến b.

Ví dụ: Có phương trình logic với 2 biến sau:

$$L=(a.b)+\left(\bar{a}.b\right)$$

Điều kiện để phương trình trên có tín hiệu "1" ở cổng ra L là khối 2 và 4. Với 2 biến ta có $2^2 = 4$ dạng phép hội toàn phần. Khối 2 và 4 được gạch chéo.

Trong biểu đồ Karnaugh là 2 dạng phép hội toàn phần có trong phương trình nằm kế cận nhau (cột 2). Hai dạng phép hội toàn phần kế cận nhau có tính chất là một trong hai biến có giá trị thay đổi, thì biến thứ 2 không thay đổi. Như ở trên, biến có giá trị thay đổi là b $\Rightarrow$ ta biến đổi phương trình trên như sau:

$$b.(a + \bar{a}) = L$$

$$a + \bar{a} = 1$$

$$b.1 = S \Rightarrow b = L$$

Ta thấy thoả mãn phương trình logic trên, do đó chỉ cần tín hiệu b.

Trong biểu đồ Karnaugh có 2 dạng phép hội toàn phần nằm kế cận nhau, thì lúc nào ta cũng có thể đơn giản được. (Nằm kế cận nhau có nghĩa là trong cùng một dòng hoặc trong cùng một cột)

1.4.2. Biểu đồ Karnaugh với 3 biến

Với 3 biến ta có $2^3 = 8$ dạng phép hội toàn phần nằm trong 8 vùng (được ký hiệu vùng 1 đến vùng 8) và được biểu diễn trên biểu đồ Karnaugh sau:

	$\bar{c}$	c	
$\bar{a}$	1 000 $\bar{a}.\bar{b}.\bar{c}$	2 001 $\bar{a}.\bar{b}.c$	$\bar{b}$
$\bar{a}$	3 010 $\bar{a}.b.\bar{c}$	4 011 $\bar{a}.b.c$	b
a	5 110 $a.b.\bar{c}$	6 111 $a.b.c$	b
a	7 100 $a.\bar{b}.\bar{c}$	8 101 $a.\bar{b}.c$	$\bar{b}$

Dòng thứ 1 gồm: $\bar{a}, \bar{b}, c, \bar{c}$

Dòng thứ 2 gồm: $\bar{a}, b, c, \bar{c}$

Dòng thứ 3 gồm: $a, b, c, \bar{c}$

Dòng thứ 4 gồm: $a, \bar{b}, c, \bar{c}$

Cột thứ 1 gồm: a và $\bar{a}, b$ và $\bar{b}, \bar{c}$

Cột thứ 2 gồm: a và $\bar{a}$, b và $\bar{b}$, c

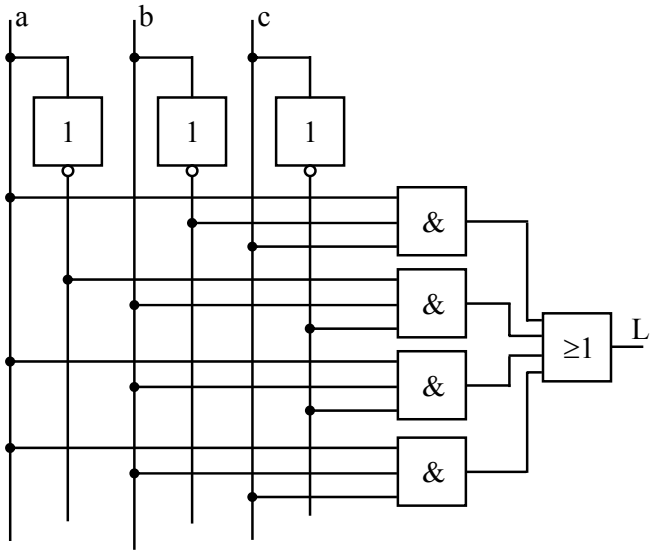
Ví dụ: ta có phương trình logic với 3 biến sau:

$$L = (\bar{a}.b.\bar{c}) + (a.\bar{b}.c) + (a.b.\bar{c}) + (a.b.c)$$

Theo biểu đồ Karnaugh, ta có phương trình logic trên với 4 khối được gạch chéo tương ứng.

Phương trình logic trên gồn có: 3 phần tử NOT
4 phần tử AND với 3 cổng ra
1 phần tử OR với 4 cổng vào

Sơ đồ mạch logic và bảng chân lý của phương trình trên là:



a	b	c	L
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Hình1.10. Sơ đồ mạch logic và bảng chân lý

Ta sử dụng biểu đồ Karnaugh để đơn giản sơ đồ mạch logic trên:

Trong biểu đồ có 2 miền lân cận, đó là:

Miền thứ 1 gồm khối 3 $(\bar{a}.b.\bar{c})$ và 5 $(a.b.\bar{c})$

Miền thứ 2 gồm khối 6 $(a.b.c)$ và 8 $(a.\bar{b}.c)$

∗ *Miền thứ 1:* khối 3 và 5 ta có:

$$L = (\bar{a}.b.\bar{c}) + (a.b.\bar{c})$$

Hay $L = (b.\bar{c})(\bar{a} + a)$

với $(\bar{a} + a) = 1 \Rightarrow L = b.\bar{c}$

∗ *Miền thứ 2:* khối 6 và 8 ta có:

$$L = (a.b.c) + (a.\bar{b}.c)$$

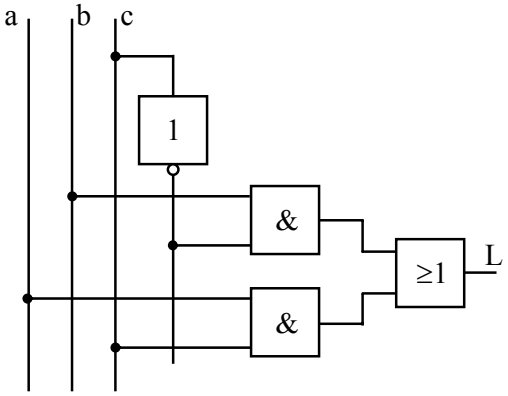
Hay $L = (a.c)(\bar{b} + b)$ với $\bar{b} + b = 1$

$\Rightarrow L = a.c$

Vậy phương trình logic được đơn giản bằng biểu đồ Karnaugh là:

$$L = \left(b.\bar{c}\right) + \left(a.c\right)$$

Và sơ đồ logic lúc này sẽ là:



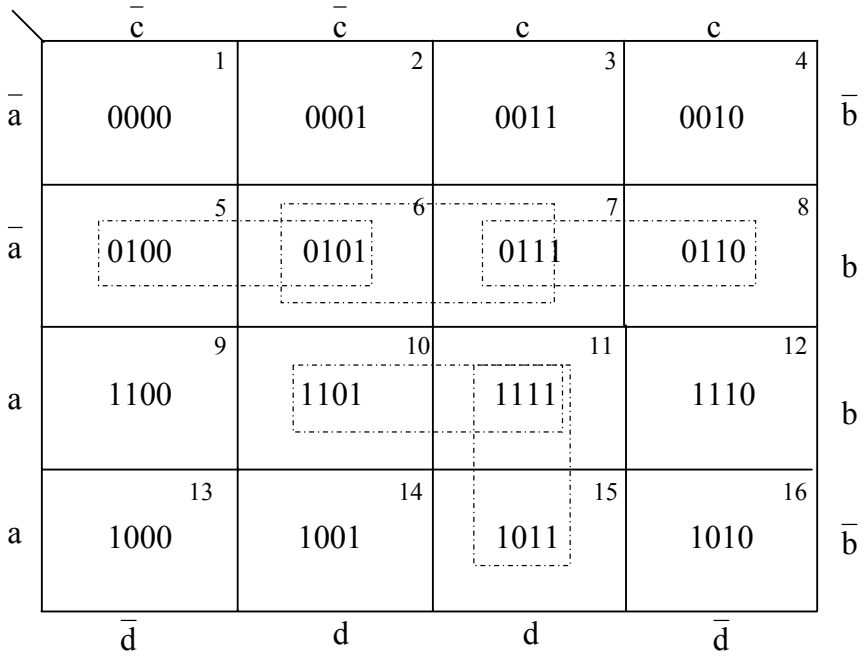
a	b	c	L
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Hình 1.11. Sơ đồ logic và bảng chân lý

Sơ đồ này chỉ còn lại 4 phần tử (đơn giản hơn rất nhiều so với sơ đồ ban đầu).

1.4.3. Biểu đồ Karnaugh với 4 biến

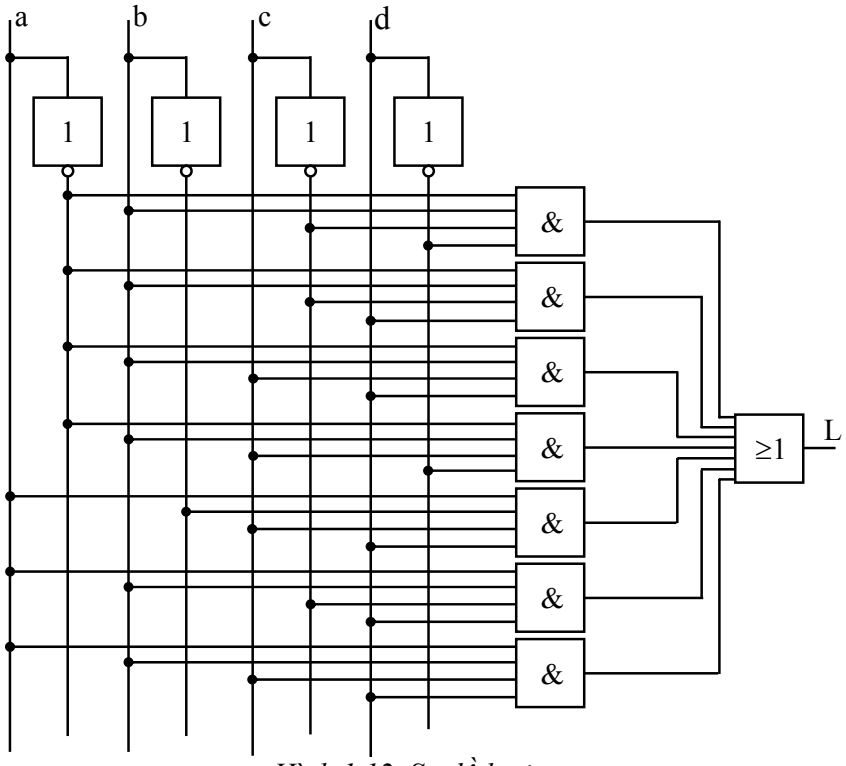
Với 4 biến ta có $2^4 = 16$ dạng phép hội toàn phần nằm trong 16 khối. Thiết lập biểu đồ Karnaugh với 4 biến cũng tương tự như biểu đồ 3 biến, tuy nhiên số khối tăng gấp đôi. Biểu đồ Karnaugh được lập như sau:



Ví dụ 1: đơn giản phương trình logic sau bằng biểu đồ Karnaugh:

$$L = (\bar{a}.b.\bar{c}.\bar{d}) + (\bar{a}.b.\bar{c}.d) + (\bar{a}.b.c.d) + (\bar{a}.b.c.\bar{d}) + (a.b.\bar{c}.d) + (a.b.c.d) + (a.\bar{b}.c.d)$$

Sơ đồ mạch logic của phương trình logic trên là:



Hình 1.12. Sơ đồ logic

Sơ đồ này gồm: 7 phần tử AND với 4 cổng vào
4 phần tử NOT
1 phần tử OR với 7 cổng vào

$\Rightarrow$ 12 phần tử

Bây giờ ta đơn giản mạch logic trên bằng biểu đồ Karnaugh. Theo phương trình logic trên, ta đánh dấu các khối tương ứng và chia ra thành các miền (có 3 miền được chia).

Miền thứ 1 gồm: khối 5, 6, 7 và 8

Miền thứ 2 gồm: khối 6, 7, 10 và 11

Miền thứ 3 gồm: khối 11 và 15

∗ *Miền thứ 1:* khối 5, 6, 7 và 8

$$(\bar{a}.b.\bar{c}.\bar{d}) + (\bar{a}.b.\bar{c}.d) + (\bar{a}.b.c.d) + (\bar{a}.b.c.\bar{d})$$

Ta chia miền thứ nhất thành 2 miền nhỏ: A + B

Trong đó:

+/ Miền nhỏ A gồm khối 5 và 6, ta có:

$$A = (\bar{a}.b.\bar{c}.\bar{d}) + (\bar{a}.b.\bar{c}.d) = (\bar{a}.b.\bar{c})(\bar{d} + d) \text{ mà } \bar{d} + d = 1$$

Vậy sau khi đơn giản miền nhỏ A, ta được:

$$A = (\bar{a}.b.\bar{c})$$

+/ Miền nhỏ B gồm khối 7 và 8, ta có:

$$B = (\bar{a}.b.c.d) + (\bar{a}.b.c.\bar{d}) = (\bar{a}.b.c).(d + \bar{d}) \text{ mà } d + \bar{d} = 1$$

Vậy sau khi đơn giản miền nhỏ B, ta được:

$$B = (\bar{a}.b.c)$$

Như vậy miền thứ 1 được viết lại là: $A + B = (\bar{a}.b.\bar{c}) + (\bar{a}.b.c)$

Theo quy tắc phân bố, ta viết lại như sau:

$$(\bar{a}.b.\bar{c}) + (\bar{a}.b.c) = (\bar{a}.b).(\bar{c} + c) \text{ mà } \bar{c} + c = 1$$

$\Rightarrow$ Miền thứ 1 được viết đơn giản thành: $(\bar{a}.b)$

∗ *Miền 2:* khối 6, 7, 10 và 11

$$(\bar{a}.b.\bar{c}.d) + (\bar{a}.b.c.d) + (a.b.\bar{c}.d) + (a.b.c.d)$$

Tương tự ta cũng chia miền 2 thành 2 miền nhỏ: C + D

Trong đó:

+/ Miền nhỏ C gồm khối 6 và 7, ta có:

$$C = (\bar{a}.b.\bar{c}.d) + (\bar{a}.b.c.d) = (\bar{a}.b.d).(\bar{c} + c) \text{ mà } \bar{c} + c = 1$$

$$\Rightarrow C = (\bar{a}.b.d)$$

+/ Miền nhỏ D gồm khối 10 và 11, ta có

$$D = (a.b.\bar{c}.d) + (a.b.c.d) = (a.b.d).(\bar{c} + c)$$

$$\Rightarrow D = (a.b.d)$$

Như vậy miền thứ 2 được viết lại là: $C + D = (\bar{a}.b.d) + (a.b.d) = (b.d).(\bar{a} + a)$

$\Rightarrow$ Miền thứ 2 được đơn giản thành: (b.d)

∗ *Miền thứ 3:* gồm khối 11 và 15, ta có:

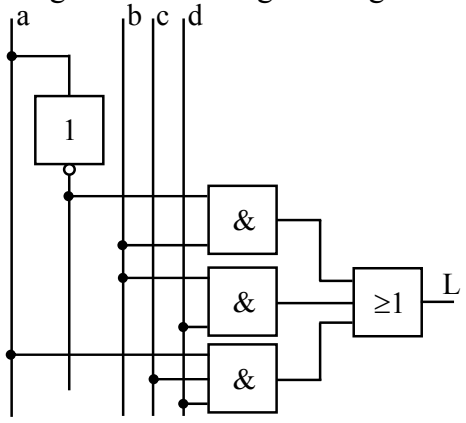
$$(a.b.c.d) + (a.\bar{b}.c.d) = (a.c.d).(b + \bar{b})$$

Như vậy miền 3 sau khi đơn giản là: (a.c.d)

Vậy phương trình logic sau khi đơn giản bằng biểu đồ Karnaugh được viết lại là:

$$L = (\bar{a}.b) + (b.d) + (a.c.d)$$

Ta có sơ đồ mạch logic sau khi đơn giản bằng biểu đồ Karnaugh là:

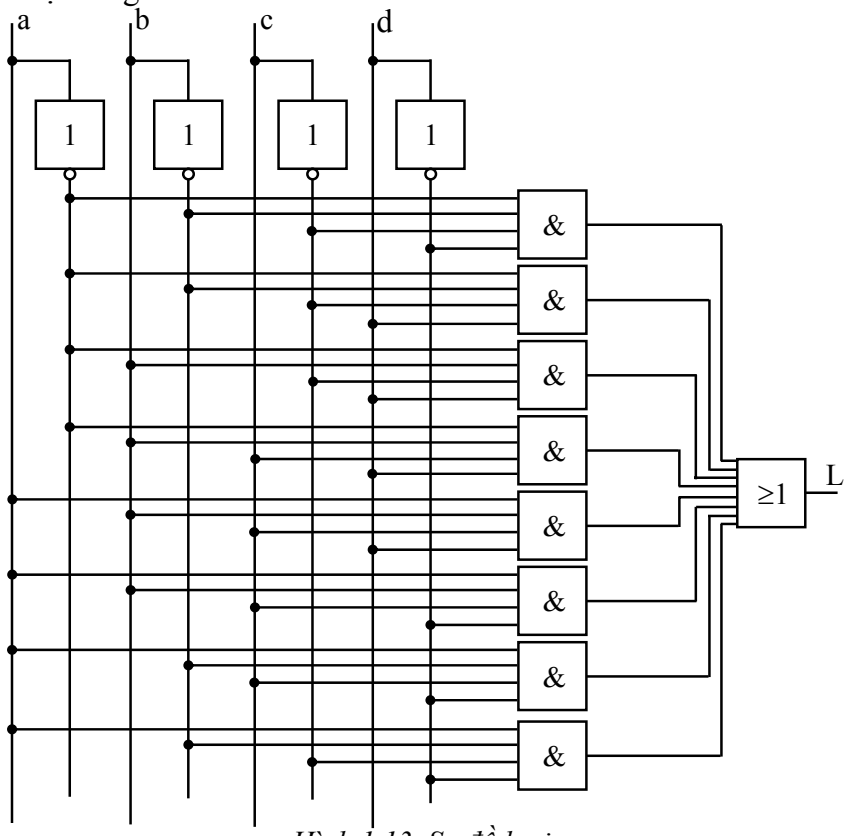


Sơ đồ này còn 5 phần tử (nhờ biểu đồ Karnaugh giảm được 7 phần tử).

Ví dụ 2: đơn giản phương trình logic bằng biểu đồ Karnaugh:

$$L = (\bar{a}.\bar{b}.\bar{c}.\bar{d}) + (\bar{a}.\bar{b}.\bar{c}.d) + (\bar{a}.b.\bar{c}.d) + (\bar{a}.b.c.d) + (a.b.c.d) + (a.b.c.\bar{d}) + (a.\bar{b}.c.\bar{d}) + (a.\bar{b}.\bar{c}.\bar{d})$$

Ta có sơ đồ mạch logic như sau:



Hình 1.13. Sơ đồ logic

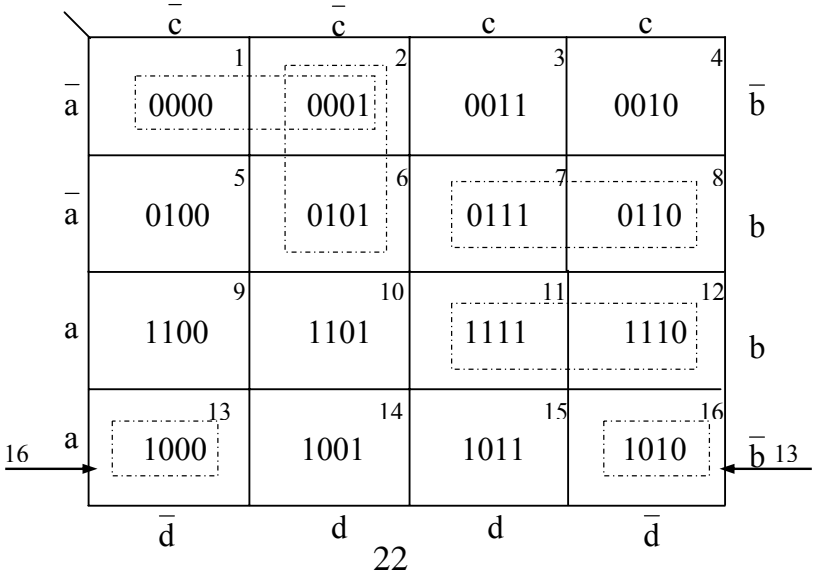
Sơ đồ mạch logic này gồm: 4 phần tử NOT

8 phần tử AND với 4 cổng vào

1 phần tử OR với 8 cổng vào

$\Rightarrow$ 13 phần tử.

Ta có biểu đồ Karnaugh của phương trình trên là:



Khi biểu đồ Karnaugh được cuộn lại thành dạng hình trụ thẳng đứng, thì khối 13 và khối 16 sẽ là những khối nằm lân cận nhau.

Theo biểu đồ ta có 4 miền lân cận, đó là:

Miền thứ 1: khối 1 và 2

Miền thứ 2: khối 6 và 7

Miền thứ 3: khối 11 và 12

Miền thứ 4: khối 13 và 16

∗ *Miền thứ 1:* khối 1 và 2, ta có:

$$\left(\bar{a}.\bar{b}.\bar{c}.\bar{d}\right)+\left(\bar{a}.\bar{b}.\bar{c}.d\right)=\left(\bar{a}.\bar{b}.\bar{c}\right).\left(\bar{d}+d\right)=\left(\bar{a}.\bar{b}.\bar{c}\right)$$

Sau khi đơn giản miền 1, ta có: $\left(\bar{a}.\bar{b}.\bar{c}\right)$

∗ *Miền thứ 2:* khối 6 và 7

$$\left(\bar{a}.b.\bar{c}.d\right)+\left(\bar{a}.b.c.d\right)=\left(\bar{a}.b.d\right).\left(\bar{c}+c\right)=\left(\bar{a}.b.d\right)$$

Sau khi đơn giản miền 2, ta có: $\left(\bar{a}.b.d\right)$

∗ *Miền thứ 3:* khối 11 và 12

$$\left(a.b.c.d\right)+\left(a.b.c.\bar{d}\right)=\left(a.b.c\right).\left(d+\bar{d}\right)=\left(a.b.c\right)$$

Sau khi đơn giản miền 3, ta có: (a.b.c)

∗ *Miền thứ 4:* khối 12 và 16

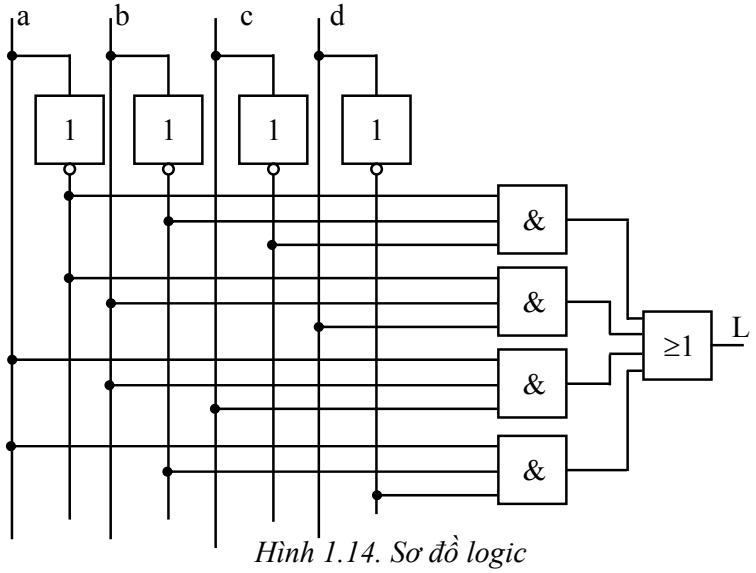
$$\left(a.\bar{b}.c.\bar{d}\right)+\left(a.\bar{b}.\bar{c}.\bar{d}\right)=\left(a.\bar{b}.\bar{d}\right).\left(c+\bar{c}\right)=\left(a.\bar{b}.\bar{d}\right)$$

Sau khi đơn giản miền 4, ta có: $\left(a.\bar{b}.\bar{d}\right)$

Vậy phương trình logic sau khi đơn giản bằng biểu đồ Karnaugh là:

$$L=\left(\bar{a}.\bar{b}.\bar{c}\right)+\left(\bar{a}.b.d\right)+\left(a.b.c\right)+\left(a.\bar{b}.\bar{d}\right)$$

Sơ đồ mạch logic của phương trình sau khi đơn giản là:



Hình 1.14. Sơ đồ logic

Sau khi đơn giản còn lại 9 phần tử, ta có thể tiếp tục đơn giản bằng quy tắc Morgan:

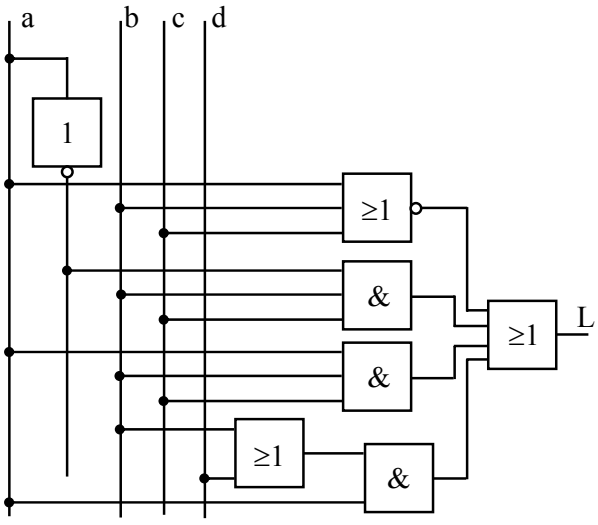
$$L = \left(\bar{a}.\bar{b}.\bar{c}\right) + \left(\bar{a}.b.d\right) + \left(a.b.c\right) + \left(a.\bar{b}.\bar{d}\right)$$

Ta có: $\left(\bar{a}.\bar{b}.\bar{c}\right) = \left(\overline{a + b + c}\right)$

$\left(a.\bar{b}.\bar{d}\right) = a.\left(\overline{b + d}\right)$

$\Rightarrow \quad L = \left(\overline{a + b + c}\right) + \left(\bar{a}.b.d\right) + \left(a.b.c\right) + \left[a.\left(\overline{b + d}\right)\right]$ (đây là kết quả cuối cùng)

Sơ đồ mạch logic là:



Hình 1.15. Sơ đồ logic

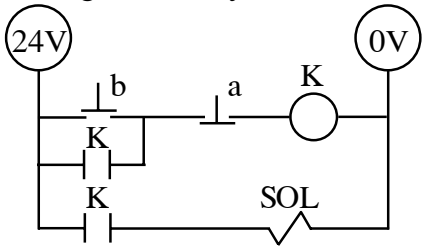
Sơ đồ này còn lại 7 phần tử: 1 phần tử NOT
3 phần tử AND
2 phần tử NOR
1 phần tử OR với 4 cổng vào.

Ví dụ 3: trang 151 (điều khiển khí nén của Nguyễn Ngọc Phương)

1.5. PHẦN TỬ NHỚ

Các phần tử đã được trình bày có đặc điểm là tín hiệu ra trong mômen thời gian phụ thuộc vào tín hiệu vào, điều đó có nghĩa là khi tín hiệu vào mất, thì tín hiệu ra cũng mất. Trong thực tế tín hiệu thường là dạng xung, khi tín hiệu tác động vào là dạng xung, tín hiệu ra thường là tín hiệu duy trì. Như vậy cần phải có phần tử duy trì tín hiệu.

Ví dụ: trong kỹ thuật điện, ta gọi là tự duy trì

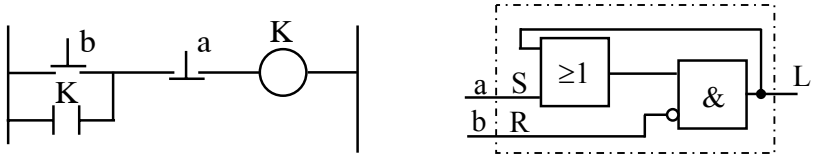


Khi ấn nút b, dòng điện đi qua rơle K làm tiếp điểm K được đóng lại $\Rightarrow$ có dòng điện qua cuộn dây. Như vậy dòng điện trong mạch vẫn duy trì, mặc dù nút ấn b nhả ra.

Dòng điện duy trì cho đến lúc nào ấn nút a. Thời gian tự duy trì dòng điện trong mạch, là khả năng nhớ của mạch điện. Trong kỹ thuật điều khiển gọi là phần tử nhớ Flipflop. Phần tử Flipflop có 2 cổng vào, cổng thứ nhất ký hiệu S (SET) và cổng thứ 2 ký hiệu R (RESET), như vậy phần tử Flipflop cũng có thể gọi cách khác là phần tử RS-Flipflop.

1.5.1. Phần tử RS - Flipflop

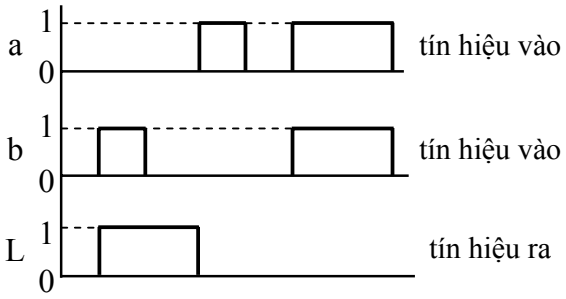
a. Phần tử RS - Flipflop có RESET trội hơn:



Hình 1.16. Phần tử nhớ (mạch điện tự duy trì và phần tử RS – Flipflop có RESET trội hơn)

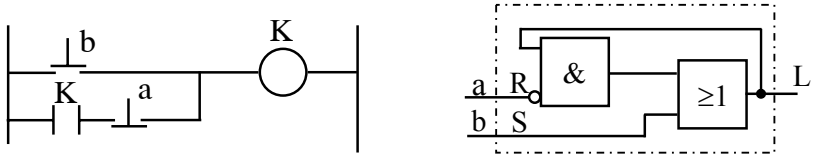
Nếu cổng SET (b) có giá trị "1", thì tín hiệu ra L có giá trị "1" và được nhớ (mặc dù ngay sau đó tín hiệu ở cổng SET mất đi) cho đến khi cổng RESET (a) có giá trị "1", thì phần tử Flipflop sẽ quay trở về vị trí ban đầu. Khi cổng SET và cổng RESET có cùng giá trị "1", thì L có giá trị "0".

Ta có bảng giá trị của phần tử RS - Flipflop như sau:



a	b	L
0	0	Không thay đổi
0	1	1
1	0	0
1	1	0

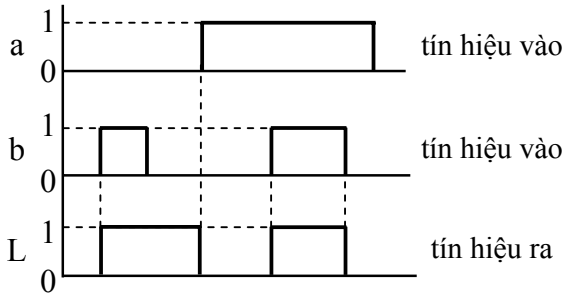
b. Phần tử RS - Flipflop có SET trội hơn:



Hình Phần tử nhớ (mạch điện tự duy trì và phần tử RS – Flipflop có SET trội hơn)

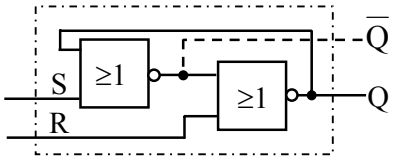
Nếu cổng SET (b) có giá trị "1", thì tín hiệu ra L có giá trị "1" và được nhớ (mặc dù ngay sau đó tín hiệu ở cổng SET mất đi) cho đến khi cổng RESET (a) có giá trị "1", thì phần tử Flipflop sẽ quay trở về vị trí ban đầu. Khi cổng SET và cổng RESET có cùng giá trị "1", thì L có giá trị "1".

Ta có bảng giá trị của phần tử RS - Flipflop như sau:



a	b	L
0	0	Không thay đổi
0	1	1
1	0	0
1	1	1

Phần tử RS - Flipflop với 2 phần tử NOR có 2 cổng ra Q *và* $\overline{Q}$, *được biểu diễn như sau:*



Hình1.17. Phần tử RS – Flipflop với 2 cổng ra Q *và* $\overline{Q}$

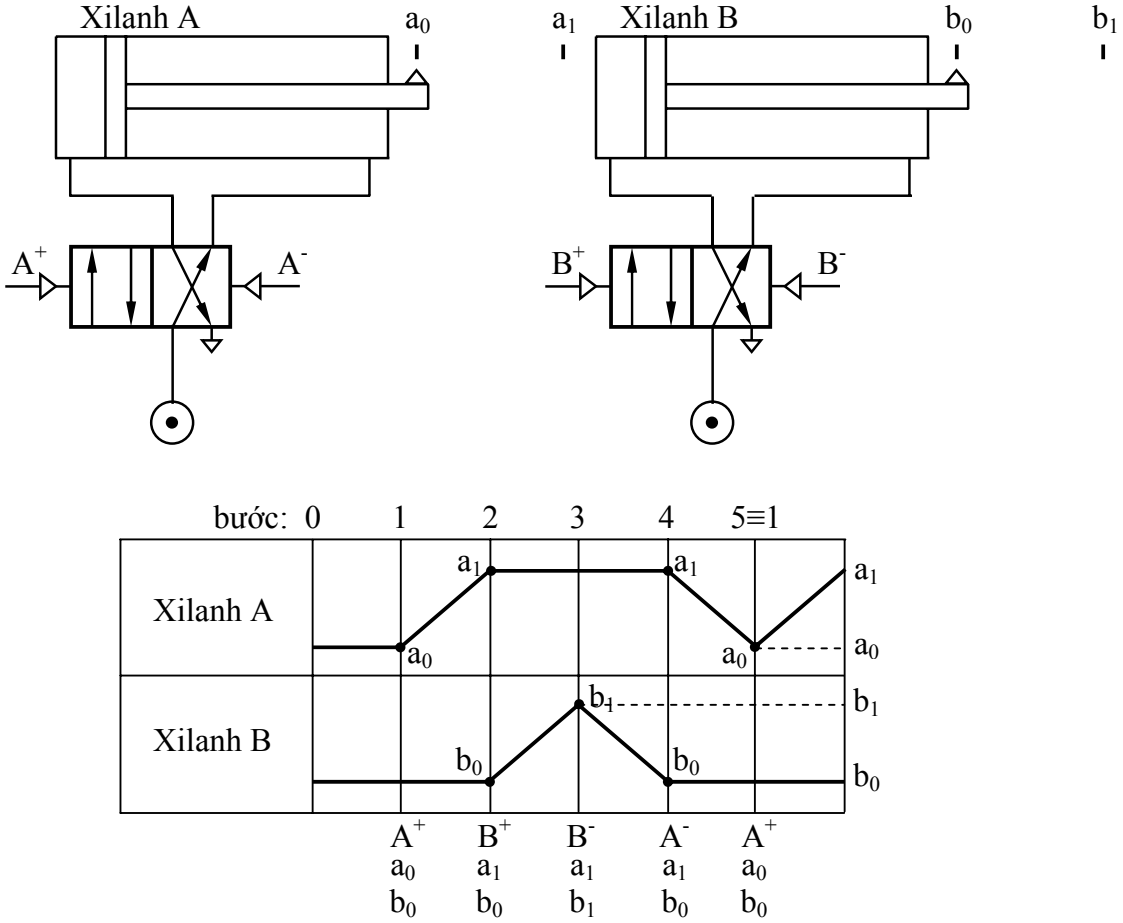
CHƯƠNG 2: **THIẾT KẾ MẠCH ĐIỀU KHIỂN LOGIC KHÍ NÉN - ĐIỆN KHÍ NÉN**

Thiết kế ra một mạch điều khiển tự động tốI ưu và kinh tế là hết sức quan trọng. Chương này giới thiệu phương pháp thiết kế mạch điều khiển khí nén, điện khí nén khí nén bằng phương pháp biểu đồ Karnaugh. Trình tự thiết kế được thể hiện qua các ví dụ cụ thể.

2.1. THIẾT KẾ MẠCH KHÍ NÉN CHO QUY TRÌNH VỚI 2 XILANH

Giả sử quy trình làm việc của một máy khoan gồm hai xilanh: khi đưa chi tiết vào xilanh A sẽ đi ra để kẹp chi tiết. Sau đó piston B đi xuống khoan chi tiết và sau khi khoan xong thì piston B lùi về. Sau khi piston B đã lùi về thì xilanh A mới lùi về.

Ta có sơ đồ khí nén và biểu đồ thời gian (biểu đồ trạng thái) như sau:



Hình 2.1. Sơ đồ khí nén và biểu đồ trạng thái

Từ biểu đồ trạng thái, ta xác định điều kiện để các xilanh làm việc:

Bước 1: piston A đi ra với tín hiệu điều khiển A^+

$$A^+ = a_0.b_0$$

Bước 2: piston B đi ra với tín hiệu điều khiển B^+

$$B^+ = a_1.b_0$$

Bước 3: piston B lùi về với tín hiệu điều khiển B^-

$$B^- = a_1.b_1$$

Bước 4: piston A lùi về với tín hiệu điều khiển A^-

$$A^- = a_1.b_0$$

$\Rightarrow$ Phương trình logic:

$$\left.\begin{aligned} A^+ &= a_0.b_0 \\ B^+ &= a_1.b_0 \\ B^- &= a_1.b_1 \\ A^- &= a_1.b_0 \end{aligned}\right\} \equiv$$

So sánh các phương trình trên, ta thấy điều kiện để thực hiện B^+ và A^- giống nhau $\Rightarrow$ Như vậy về phương diện điều khiển thì điều đó không thể thực hiện được.

Để có thể phân biệt được các bước thực hiện B^+ và A^- có cùng điều kiện ($a_1.b_0$) thì cả 2 phương trình phải thêm điều kiện phụ. Trong điều khiển người ta sử dụng phần tử nhớ trung gian (ký hiệu x và $\bar{x}$ là tín hiệu ra của phần tử nhớ trung gian).

Phương trình logic trên được viết lại như sau:

$$\begin{aligned} A^+ &= a_0.b_0 \\ B^+ &= a_1.b_0.\bar{x} \\ B^- &= a_1.b_1 \\ A^- &= a_1.b_0.x \end{aligned}$$

Để tín hiệu ra $\bar{x}$ của phần tử nhớ trung gian thực hiện bước 2 (B^+), thì tín hiệu đó tín hiệu đó phải được chuẩn bị trong bước thực hiện trước đó (tức là bước thứ 1). Tương tự như vậy để tín hiệu ra x của phần tử nhớ trung gian thực hiện bước 4 (A^-), thì tín hiệu đó phải được chuẩn bị trong bước thực hiện trước đó (tức là bước thứ 3).

Từ đó ta viết lại phương trình logic như sau:

$$\begin{aligned} A^+ &= a_0.b_0.\bar{x} \\ B^+ &= a_1.b_0.\bar{x} \\ B^- &= a_1.b_1.x \\ A^- &= a_1.b_0.x \end{aligned}$$

Chuẩn bị trước

Thêm

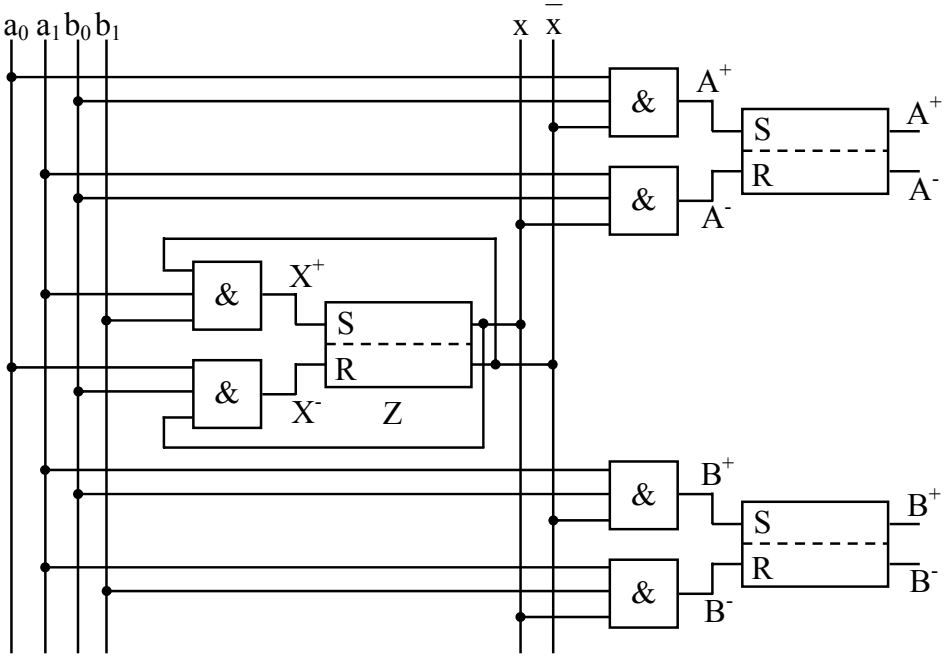
Trong quy trình thêm một phần tử nhớ trung gian (Z), ta có tín hiệu ra để điều khiển phần tử nhớ là:

$$\begin{cases} X^+ = a_1.b_1.\bar{x} \\ X^- = a_0.b_0.x \end{cases}$$

Như vậy ta có 6 phương trình không trùng nhau:

$$A^+ = a_0.b_0.\overline{x}$$
$$B^+ = a_1.b_0.\overline{x}$$
$$B^- = a_1.b_1.x$$
$$A^- = a_1.b_0.x$$
$$X^+ = a_1.b_1.\overline{x}$$
$$X^- = a_0.b_0.x$$

Với 6 phương trình trên ta có sơ đồ mạch logic như sau:

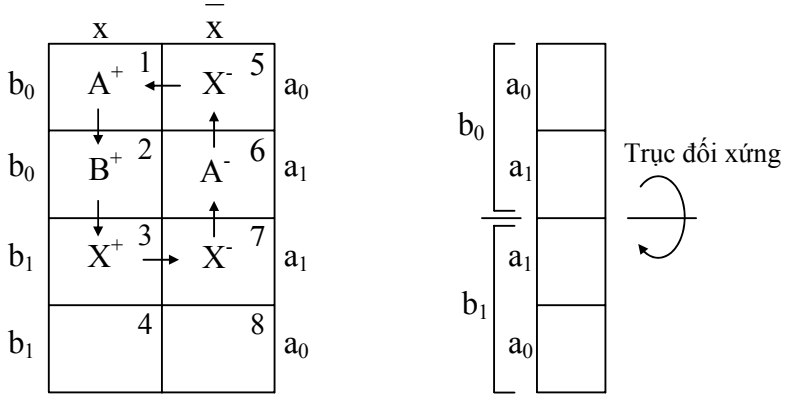


Hình 2.2. Sơ đồ mạch logic

Rút gọn bằng phương pháp biểu đồ Karnaugh:

Thiết lập biểu đồ Karnaugh: ta có 3 biến a_1 và phủ định a_0
b_1 và phủ định b_0
x và phủ định $\overline{x}$

⇒ Biểu đồ Karnaugh với 3 biến được biểu diễn như sau:



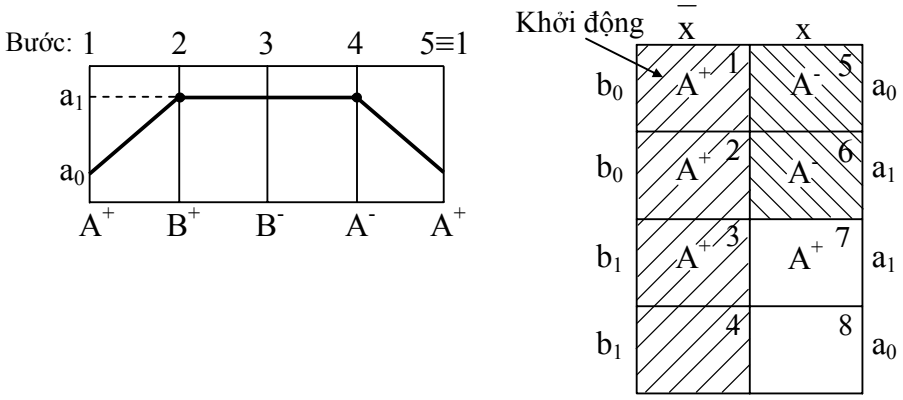
Hình 2.3. Biểu đồ Karnaugh với 3 biến

Các công tắc hành trình sẽ biểu diễn qua trục đối xứng nằm ngang, biến của phần tử nhớ trung gian biểu diễn qua trục đối xứng thẳng đứng. Trong điều khiển giả thiết rằng, khi công tắc hành trình (ví dụ a_0) bị tác động thì công tắc hành a_1 sẽ không tác động.

Không xảy ra trường hợp cả 2 công tắc hành trình a_0 và a_1 cùng tác động đồng thời hoặc cả 2 công tắc tác động đồng thời.

∗ Bây giờ ta đơn giản hành trình của xilanh A bằng biểu đồ Karnaugh:

Theo biểu đồ trạng thái, ta thiết lập được biểu đồ Karnaugh cho xilanh A:



Hình 2.4. Biểu đồ Karnaugh cho xilanh A

Bước thực hiện thứ nhất là piston A đi ra (A^+) và dừng lại cho đến bước thực hiện thứ 3. Sang bước thứ 4 thì piston A lùi về (A^-).

Các khối 1, 2, 3 và 7 ký hiệu A^+ và các khối 5, 6 ký hiệu A^-.

Đơn giản hành trình của xilanh A (A^+) sẽ được thực hiện trong cột thứ nhất ($\bar{x}$). Ta có phương trình logic của A^+ là:

$$A^+ = a_0.b_0.\bar{x}.S_0 \quad (\text{với } S_0 \text{ là nút khởi động})$$

Cột thứ nhất ($\bar{x}$) gồm các khối 1, 2, 3 và 4, trong đó khối 4 là trống.

$$\Rightarrow A^+ = a_0.b_0.\bar{x} + a_1.b_0.\bar{x} + a_1.b_1.\bar{x} + a_0.b_1.\bar{x}$$

hay: $A^+ = (a_0 + a_1).b_0.\bar{x} + (a_1 + a_0).b_1.\bar{x} = b_0.\bar{x} + b_1.\bar{x} = (b_0 + b_1).\bar{x}$

$$\Rightarrow A^+ = \bar{x}.S_0$$

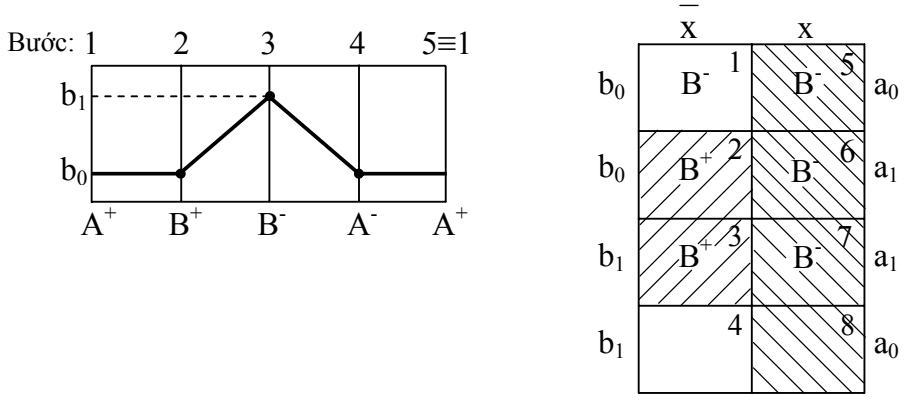
Tương tự, ta có phương trình logic của A^-:

$$A^- = a_1.b_0.x \quad \text{đơn giản khối 5 và 6}$$

$$\Rightarrow A^- = a_1.b_0.x + a_0.b_0.x = (a_1 + a_0).b_0.x$$

$$\Rightarrow A^- = b_0.x$$

∗ Phương pháp tương tự như xilanh A, ta đơn giản hành trình của xilanh B bằng biểu đồ Karnaugh:



Hình 2.5. Biểu đồ Karnaugh cho xilanh B

Ta có phương trình logic ban đầu của B^+:

$B^+ = a_1.b_0.\bar{x}$ đơn giản khối 2 và 3

$B^+ = a_1.b_0.\bar{x} + a_1.b_1.\bar{x} = (b_0 + b_1).a_1.\bar{x}$

$\Rightarrow B^+ = a_1.\bar{x}$

Và $B^- = a_1.b_1.x$ đơn giản cột x gồm các khối 5, 6, 7 và 8, trong đó khối 8 là trống.

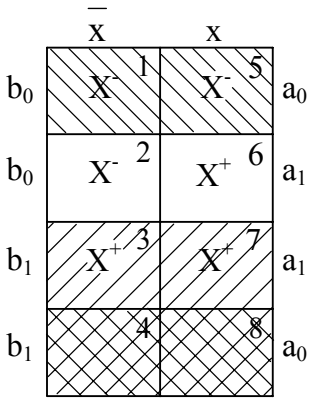
Ta có: $B^- = a_0.b_0.x + a_1.b_0.x + a_1.b_1.x + a_0.b_1.x = (a_0 + a_1).b_0.x + (a_1 + a_0).b_1.x$

$= b_0.x + b_1.x = (b_0 + b_1).x$

$\Rightarrow B^- = x$

∗ Đơn giản phần tử nhớ trung gian (X) bằng biểu đồ Karnaugh:

Biểu đồ Karnaugh cho thấy rằng phần tử nhớ trung gian ở vị trí SET bắt đầu trong khối 3, giữ vị trí đó cho đến khối 7 và 6. Từ khối 5 bắt đầu vị trí RESET và giữ vị trí đó cho đến khối 1 và 2.



Hình 2.6. Biểu đồ Karnaugh cho phần tử nhớ trung gian

Phương trình logic ban đầu của X^+:

$X^+ = a_1.b_1.\bar{x}$ đơn giản X^+ ở miền gồm các khối 3, 7, 4, và 8, ta có:

$X^+ = a_1.b_1.\bar{x} + a_1.b_1.x + a_0.b_1.\bar{x} + a_0.b_1.x$

$= (\bar{x} + x).a_1.b_1 + (\bar{x} + x).a_0.b_1 = (a_1 + a_0).b_1$

$\Rightarrow X^+ = b_1$

Phương trình logic ban đầu của X^-:

$X^- = a_0.b_0.x$ đơn giản X^- ở miền gồm các khối 1, 5, 4 và 8, ta có:

$X^- = a_0.b_0.\bar{x} + a_0.b_0.x + a_0.b_1.\bar{x} + a_0.b_1.x = (\bar{x} + x).a_0.b_0 + (\bar{x} + x).a_0.b_1$

$= (b_0 + b_1).a_0$

$\Rightarrow X^- = a_0$

(Khối trống 4 và 8 được phép sử dụng chung cho cả X^+ và X^-)

Vậy phương trình logic sau khi đơn giản là:

$A^+ = \overline{x}.S_0$ (S_0: là nút khởi động)

$A^- = b_0.x$

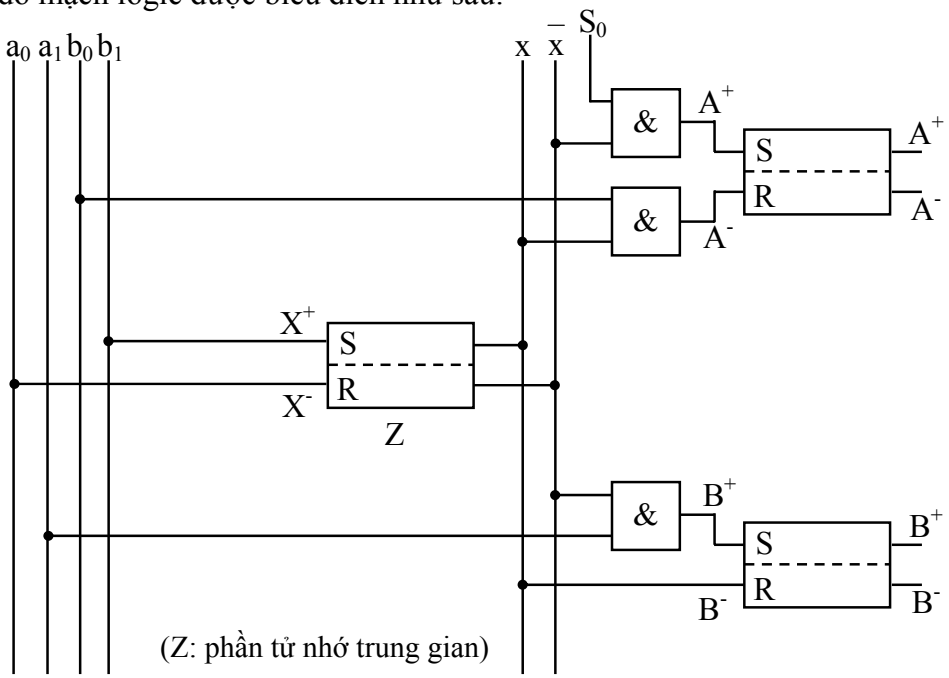
$B^+ = a_1.\overline{x}$

$B^- = x$

$X^+ = b_1$

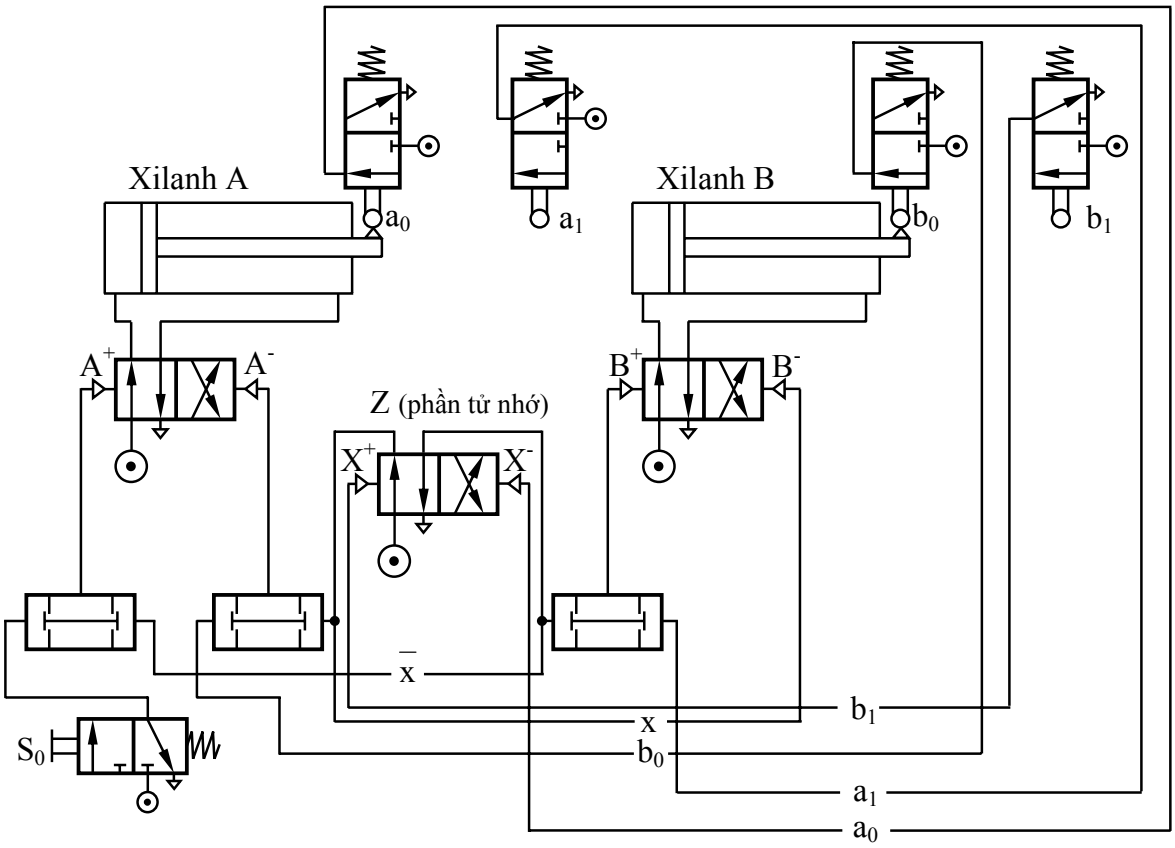
$X^- = a_0$

Sơ đồ mạch logic được biểu diễn như sau:



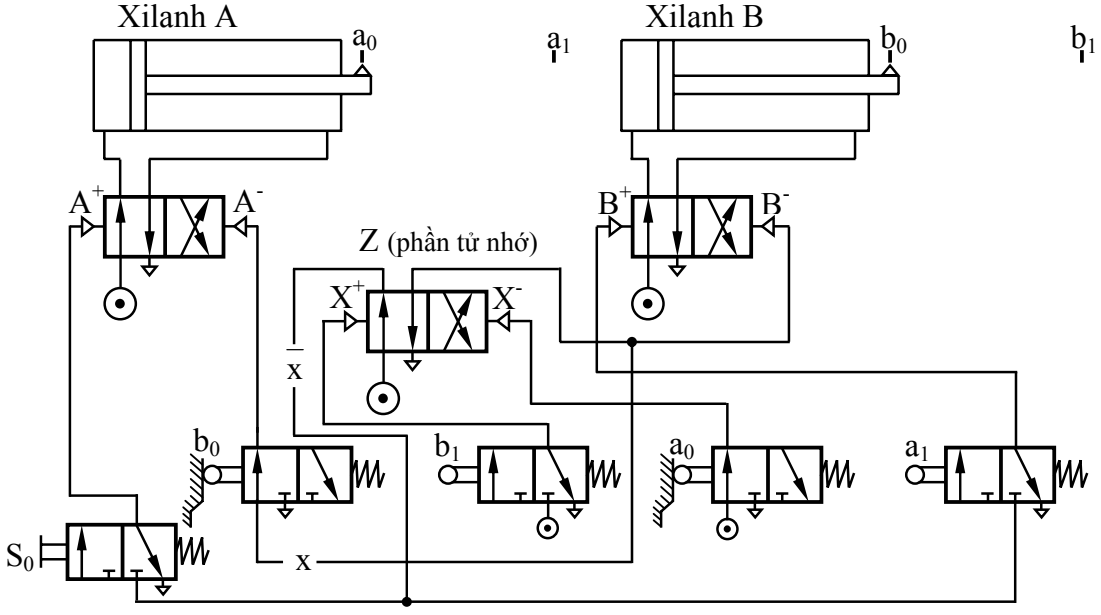
Hình 2.7. Sơ đồ mạch logic sau khi đơn giản

Sơ đồ mạch lắp ráp khí nén được biểu diễn:



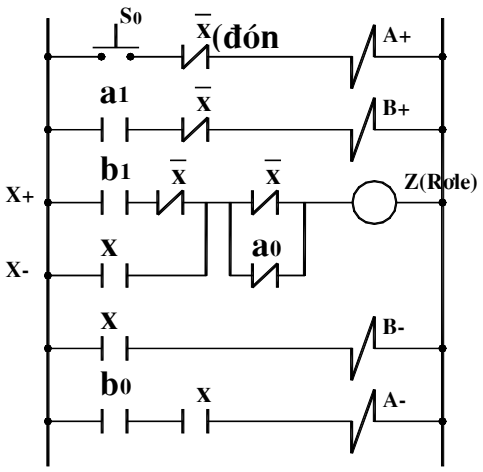
Hình 2.8. Sơ đồ mạch lắp ráp

⇒ Sơ đồ nguyên lý làm việc của mạch khí nén đơn giản như sau:



Hình 2.9. Sơ đồ nguyên lý mạch điều khiển khí nén

Với phương pháp giải tương tự như ở trên. Nếu ta thay hai van đảo chiều điều khiển bằng tín hiệu khí nén bằng hai van điện thì ta có sơ đồ mạch điện điều khiển:

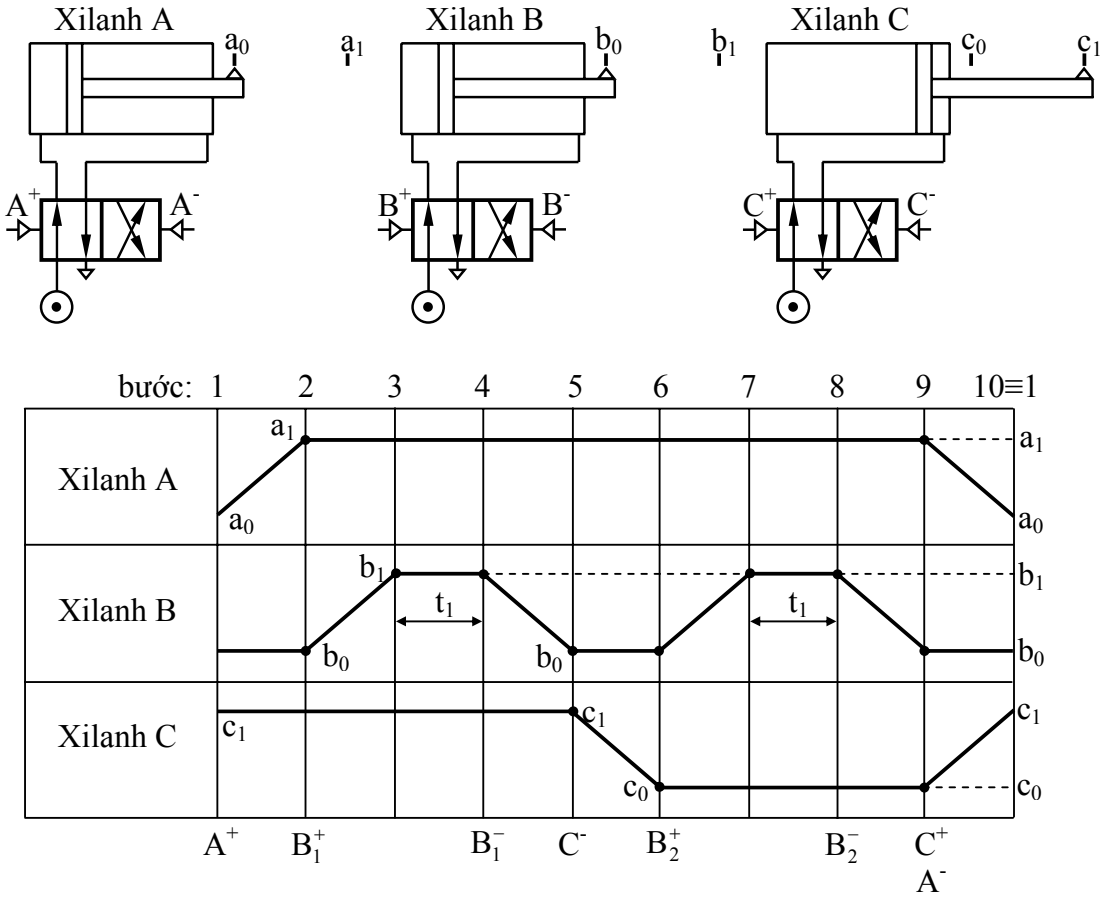


Hình 2.10. Sơ đồ nguyên lý mạch điều khiển bằng điện

2.2. THIẾT KẾ MẠCH KHÍ NÉN CHO QUY TRÌNH VỚI 3 XILANH

Giả sử, quy trình của máy làm sạch chi tiết gồm 3 xilanh: chi tiết đưa vào và sẽ được kẹp bằng xilanh A đi ra. Sau đó xilanh B sẽ thực hiện quy trình làm sạch một phía của chi tiết bằng vòi phun trong khoảng thời gian t_1. Sau đó chi tiết sẽ được chuyển sang vị trí đối diện bằng xilanh C. Tại vị trí này chi tiết sẽ thực hiện quy trình làm sạch phía thứ 2 của chi tiết này bằng vòi phun trong khoảng thời gian t_1. Sau khi thực hiện xong , xilanh C trở về vị trí ban đầu, đồng thời xilanh A sẽ lùi về $\Rightarrow$ chi tiết được tháo ra.

Ta có sơ đồ khí nén và biểu đồ trạng thái như sau:



Hình 2.11. Sơ đồ và biểu đồ trạng thái

A^+: kẹp chi tiết

B_1^+, B_2^+: quá trình thực hiện làm sạch chi tiết bắt đầu

B_1^-, B_2^-: quá trình thực hiện làm sạch chi tiết kết thục

C^+: chi tiết ở vị trí 1

C^-: chi tiết ở vị trí 2

A^-: tháo chi tiết

Thiết lập phương trình logic:

Bởi vì lệnh B^+ và B^- của piston B trong quá trình thực hiện được lặp lại 2 lần, cho nên B_1^+, B_2^+ và B_1^-, B_2^- sẽ được liên kết bằng phần tử OR.

Lệnh C^+ và A^- được thực hiện đồng thời, cho nên phương trình logic giống nhau.

Ta có phương trình logic cho A^+:

$$A^+ = a_0.b_0.c_1$$

Phương trình logic cho B^+:

$$B^+ = (a_1.b_0.c_1) + (a_1.b_0.c_0)$$

Phương trình logic cho B^-:

$$B^- = (a_1.b_1.c_1) + (a_1.b_1.c_0)$$

Phương trình logic cho C^-:

$$C^- = a_1.b_0.c_1$$

Phương trình logic cho C^+, A^-:

$$C^+ = a_1.b_0.c_0$$

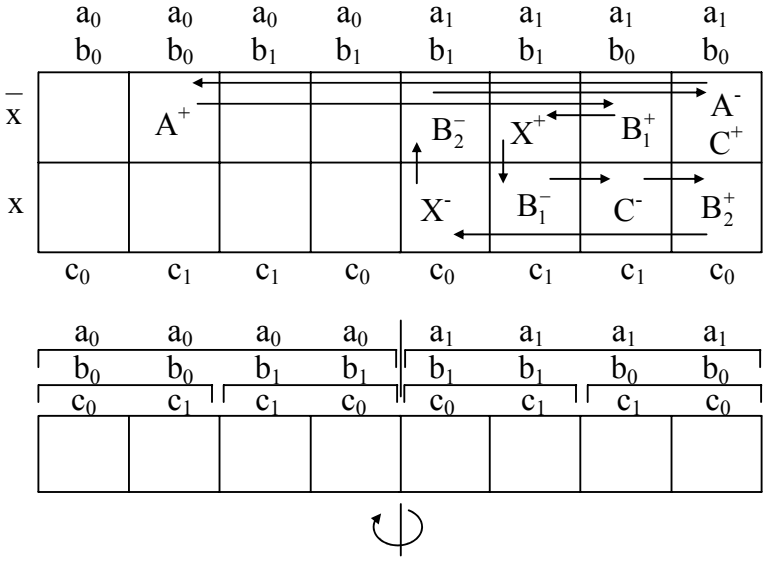
$$A^- = a_1.b_0.c_0$$

$$\Rightarrow A^- = C^+$$

Phương trình logic với các điều kiện:

Bởi vì phương trình logic cho B_1^+ và C^-, cũng như B_2^+ và C^+/A^- giống nhau, cho nên phải thêm điều kiện phụ, đó là phần tử nhớ trung gian. Lệnh SET của phần tử nhớ trung gian sẽ nằm khối ở giữa B_1^+ và B_1^-. Lệnh RESET của phần tử trung gian sẽ nằm khối giữa B_2^+ và B_2^-.

Biểu đồ Karnaugh được biểu diễn như sau:



Hình 2.12. Biểu đồ Karnaugh với 4 biến

Ta có:

$A^+ = a_0.b_0.c_1.\overline{x}$

$B^+ = (a_1.b_0.c_1.\overline{x}) + (a_1.b_0.c_0.x)$

$B^- = (a_1.b_1.c_1.x) + (a_1.b_1.c_0.\overline{x})$

$C^- = a_1.b_0.c_1.x$

$C^+ = a_1.b_0.c_0.\overline{x}$

$A^- = a_1.b_0.c_0.\overline{x}$

$X^+ = a_1.b_1.c_1.\overline{x}$

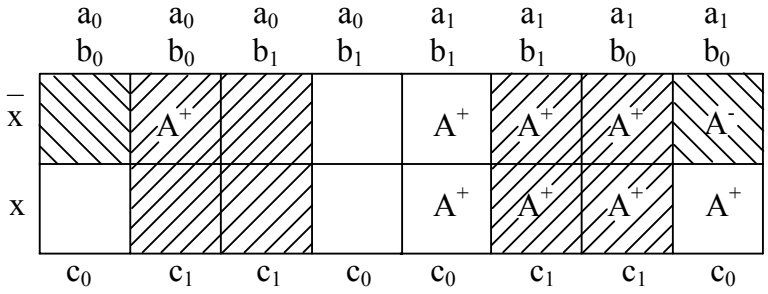
$X^- = a_1.b_1.c_0.x$

∗ *Đơn giản hành trình của xilanh A bằng biểu đồ Karnaugh (A^+, A^-)*

(*Ghi chú:* đối với những quy trình phức tạp, ta đơn giản biểu đồ Karnaugh bằng quy tắc sau đây:

- Nới rộng ra miền của khối
- Mỗi khối chỉ ghi một bước thực hiện
- Những khối trống có thể kết hợp với khối đã ghi bước thực hiện
- Những miền được tạo ra phải đối xứng qua trục đối xứng
- Số khối của miền được tạo ra phải là lũy thừa của 2.).

Theo quy tắc đó, ta đơn giản xilanh A như sau:



Hình 2.13. Biểu đồ Karnaugh cho xilanh A

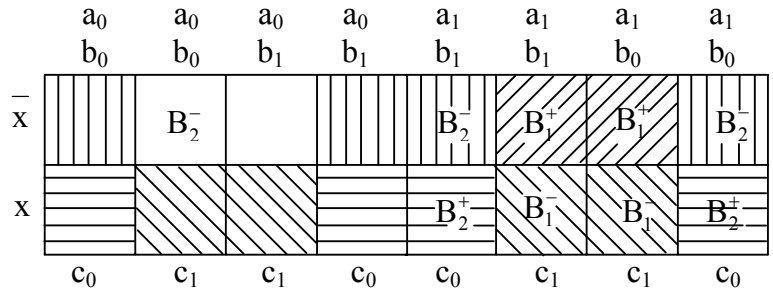
Ta có, phương trình logic sau khi đơn giản:

$A^+ = c_1.S_0$ (S_0: nút ấn khởi động)

$A^- = b_0.c_0.\overline{x}$

∗ *Đơn giản hành trình của xilanh B bằng biểu đồ Karnaugh* (B_1^+, B_2^+ và B_1^-, B_2^-)

Biểu đồ Karnaugh cho xilanh B được biểu diễn như sau:



Hình 2.14. Biểu đồ Karnaugh cho xilanh B

Ta có, phương trình logic sau khi đơn giản:

$$B_1^+ = a_1.c_1.\bar{x}$$

$$B_2^+ = c_0.x$$

$$\Rightarrow B^+ = (a_1.c_1.\bar{x}) + c_0.x$$

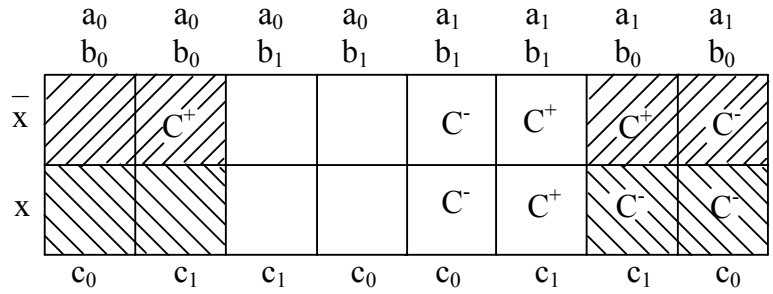
$$B_1^- = c_1.x$$

$$B_2^- = c_0.\ \bar{x}$$

$$\Rightarrow B^- = (c_1.x) + (c_0.\ \bar{x})$$

∗ *Đơn giản hành trình của xilanh C (C^+, C^-)*

Biểu đồ Karnaugh cho xilanh C được biểu diễn như sau:



Hình 2.15. Biểu đồ Karnaugh cho xilanh C

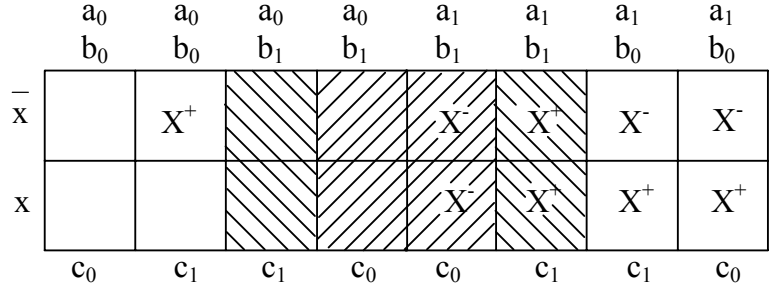
Ta có, phương trình logic sau khi đơn giản:

$$C^+ = b_0.\bar{x}$$

$$C^- = b_0.x$$

∗ *Đơn giản hành trình của phần tử nhớ trung gian (X^+, X^-)*

Biểu đồ Karnaugh cho phần tử nhớ trung gian được biểu diễn như sau:



Hình 2.16. Biểu đồ Karnaugh cho phần tử nhớ trung gian

Ta có, phương trình logic sau khi đơn giản:

$X^+ = b_1.c_1$

$X^- = b_1.c_0$

Phương trình logic của quy trình sau khi đơn giản bằng biểu đồ Karnaugh:

$A^+ = c_1.S_0$

$A^- = b_0.c_0.\overline{x}$

$B^+ = (a_1.c_1.\overline{x}) + c_0.x$

$B^- = (c_1.x) + (c_0.\overline{x})$

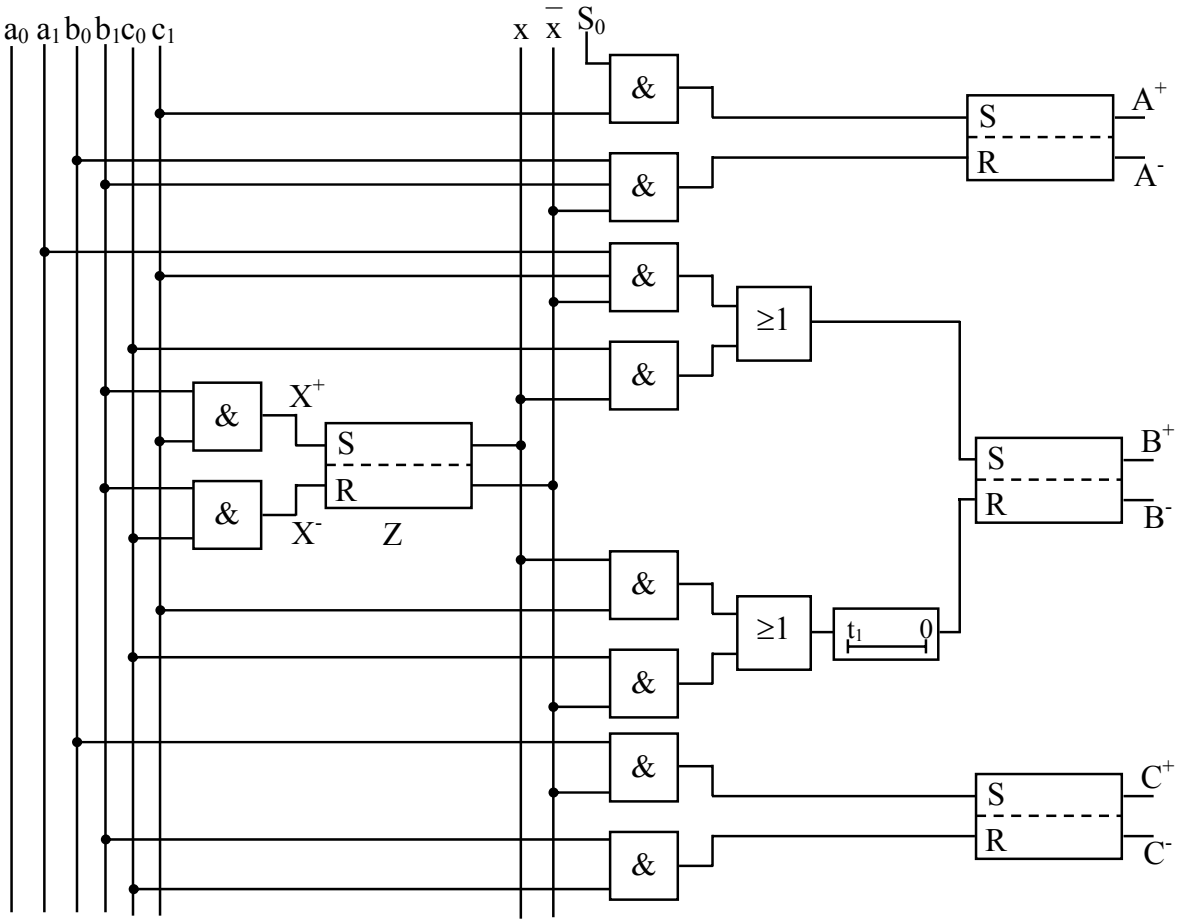
$C^+ = b_0.\overline{x}$

$C^- = b_0.x$

$X^+ = b_1.c_1$

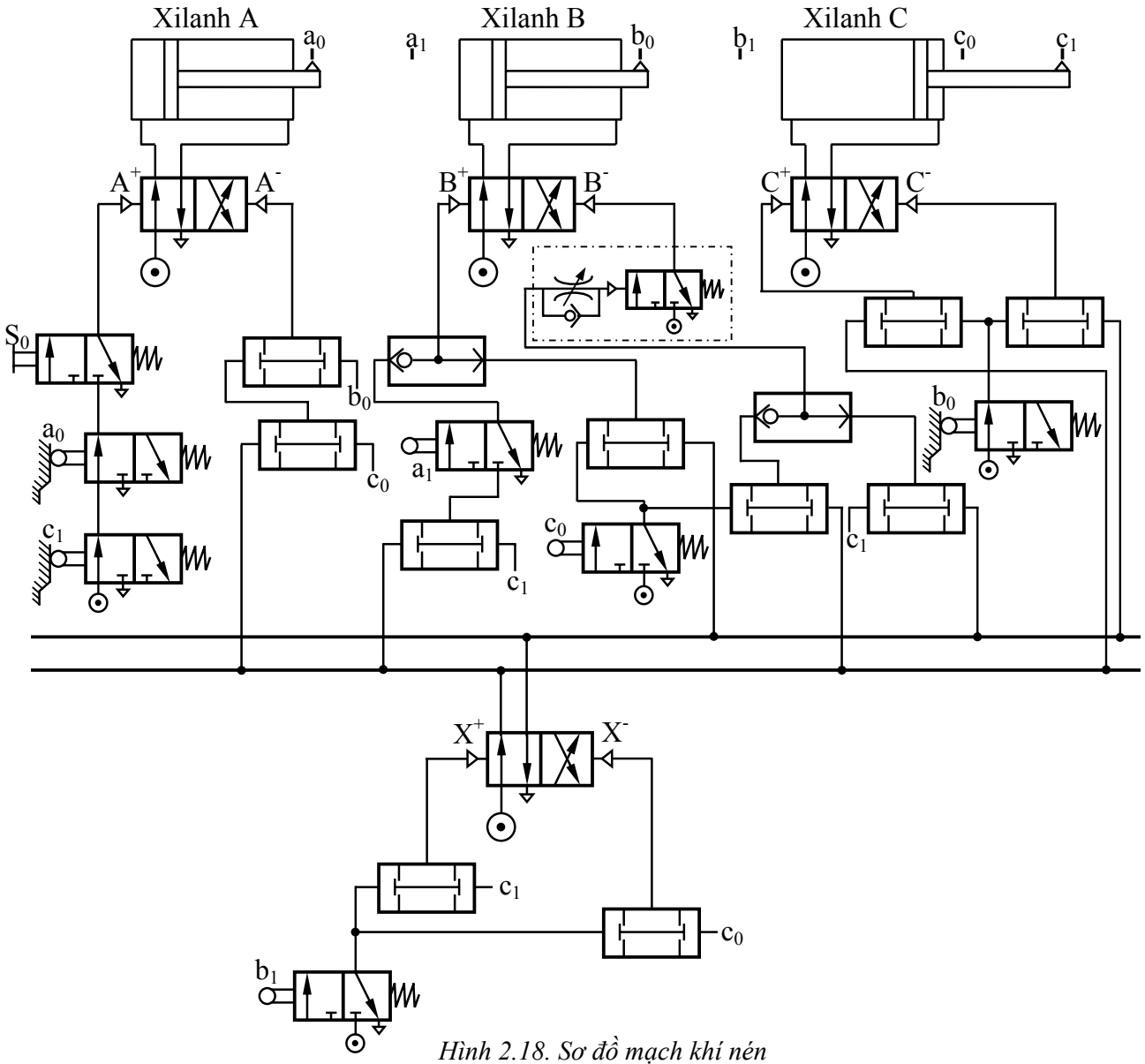
$X^- = b_1.c_0$

Sơ đồ mạch logic của quy trình được biểu diễn:



Hình 2.17. Sơ đồ mạch logic

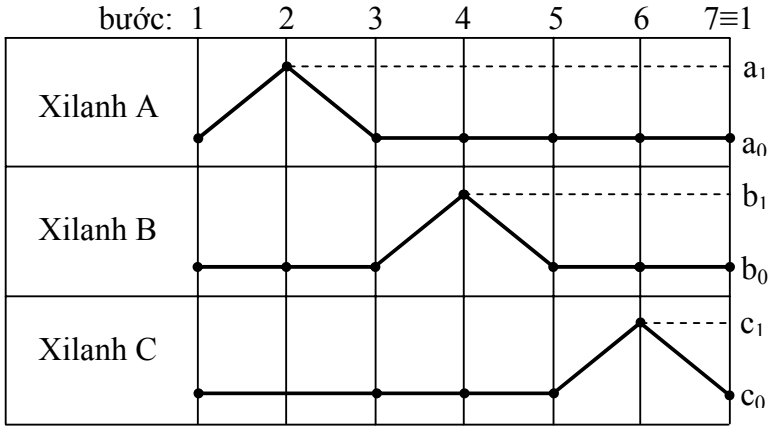
Sơ đồ nguyên lý mạch điều khiển bằng tín hiệu khí nén:



Hình 2.18. Sơ đồ mạch khí nén

2.3. THIẾT KẾ MẠCH KHÍ NÉN VỚI 2 PHẦN TỬ NHỚ TRUNG GIAN

Giả sử, quy trình công nghệ được biểu diễn qua biểu đồ trạng thái sau:



Hình 2.19. Biểu đồ trạng thái

Phương trình logic của quy trình:

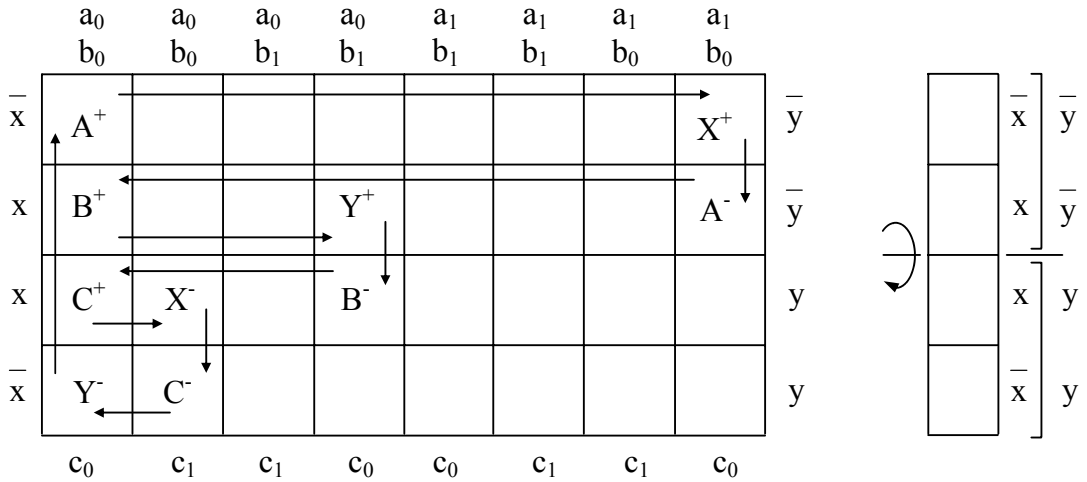
Từ biểu đồ trạng thái, ở các vị trí 1, 3 và 5 phương trình logic của các xilanh A^+, B^+ và C^+ giống nhau. Cho nên để phân biệt được các hành trình trên, ta phải thêm 2 phần tử nhớ trung gian (ký hiệu X và Y). Phương trình logic của quy trình được viết như sau:

$A^+ = a_0.b_0.c_0.\bar{x}.\bar{y}$ $B^+ = a_0.b_0.c_0.x.\bar{y}$ $C^+ = a_0.b_0.c_0.x.y$ $X^+ = a_1.b_0.c_0.\bar{x}.\bar{y}$

$A^- = a_1.b_0.c_0.x.\bar{y}$ $B^- = a_0.b_1.c_0.x.y$ $C^- = a_0.b_0.c_1.\bar{x}.y$ $X^- = a_0.b_0.c_1.x.y$

$Y^+ = a_0.b_1.c_0.x.\bar{y}$ $Y^- = a_0.b_0.c_0.\bar{x}.y$

Biểu đồ Karnaugh đươc biểu diễn như sau: (tín hiệu điều khiển của phần tử nhớ trung gian được biểu diễn đối xứng qua trục)



Hình 2.20. Biểu đồ Karnaugh với 2 phần tử nhớ trung gian

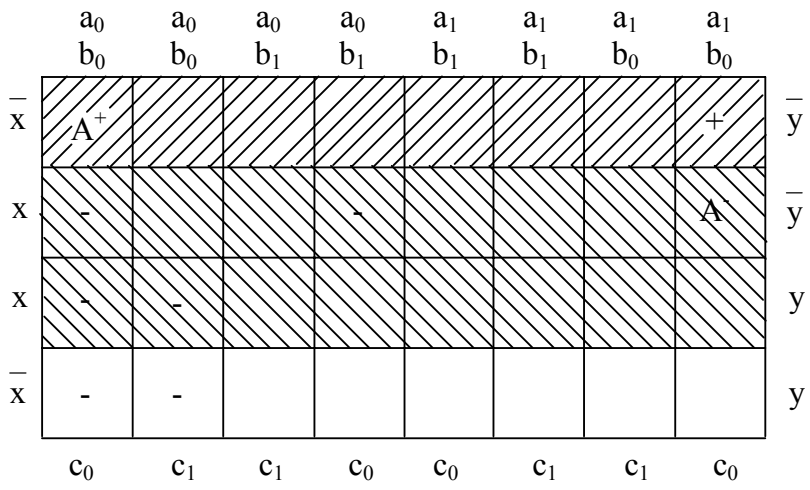
Đơn giản các hành trình bằng biểu đồ Karnaugh:

∗ Đơn giản hành trình của xilanh A^+, A^- được biểu diễn:

Ta có, phương trình logic sau khi đơn giản:

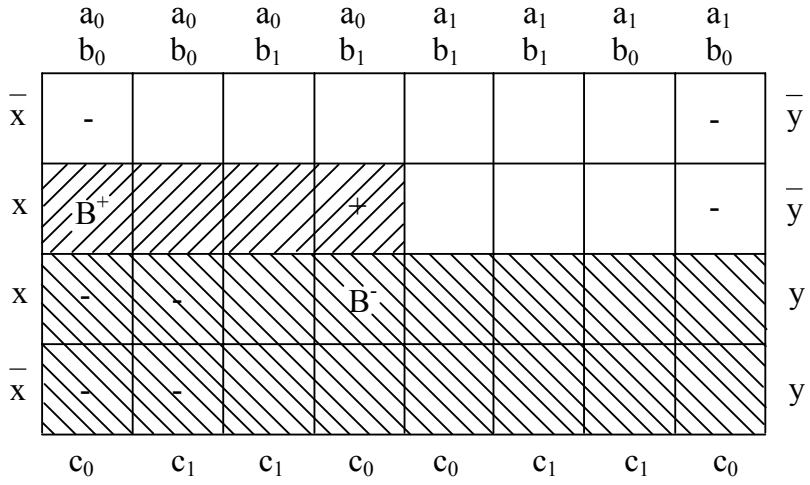
$A^+ = \bar{x} . \bar{y} . S_0$ (S_0: nút khởi động)

$A^- = x$



Hình 2.21. Biểu đồ Karnaugh cho xilanh A^+ và A^-

∗ Đơn giản hành trình của xilanh B^+, B^- được biểu diễn:



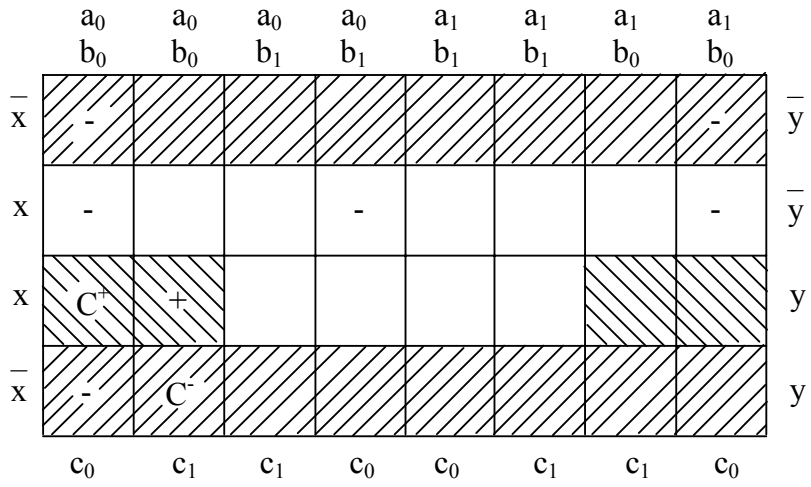
Hình 2.22 Biểu đồ Karnaugh cho xilanh B^+ và B^-

Ta có, phương trình logic sau khi đơn giản:

$B^+ = a_0.x.\bar{y}$

$B^- = y$

∗ Đơn giản hành trình của xilanh C^+, C^- được biểu diễn:



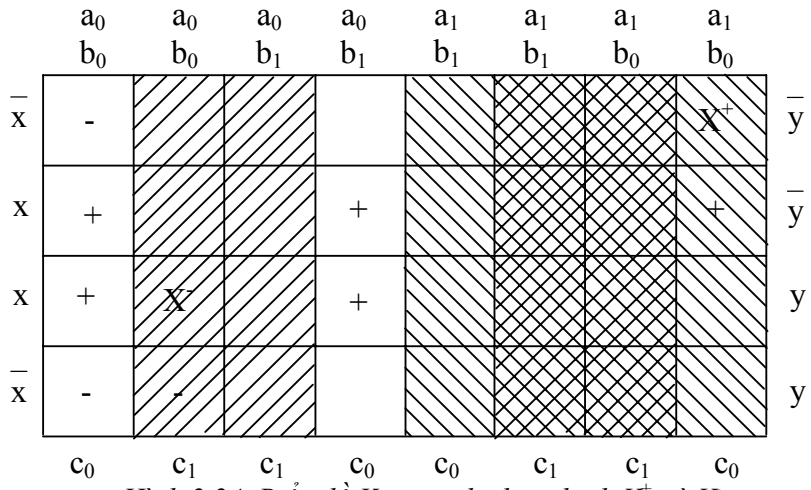
Hình 2.23. Biểu đồ Karnaugh cho xilanh C^+ và C^-

Ta có, phương trình logic sau khi đơn giản:

$C^+ = b_0.x.y$

$C^- = x$

∗ Đơn giản hành trình của xilanh X^+, X^- được biểu diễn:



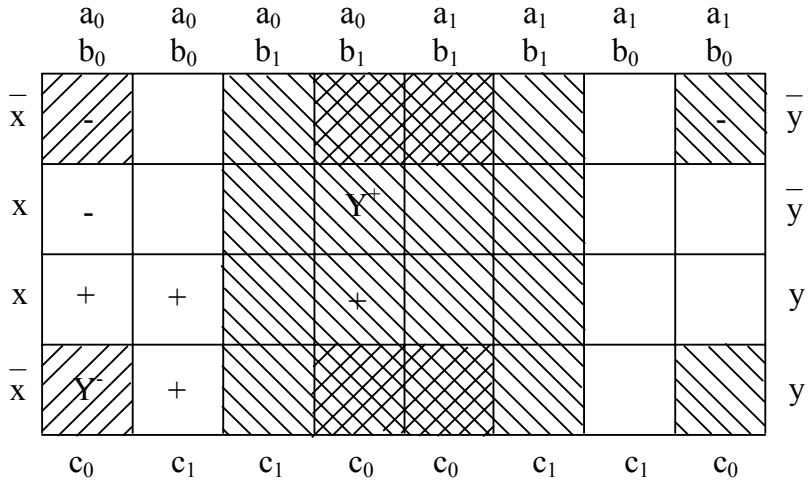
Hình 2.24. Biểu đồ Karnaugh cho xilanh X^+ và X^-

Ta có, phương trình logic sau khi đơn giản:

$X^+ = a_1$

$X^- = c_1$

∗ Đơn giản hành trình của xilanh Y^+, Y^- được biểu diễn:



Hình 2.25. Biểu đồ Karnaugh cho xilanh Y^+ và Y^-

Ta có, phương trình logic sau khi đơn giản:

$Y^+ = b_1$

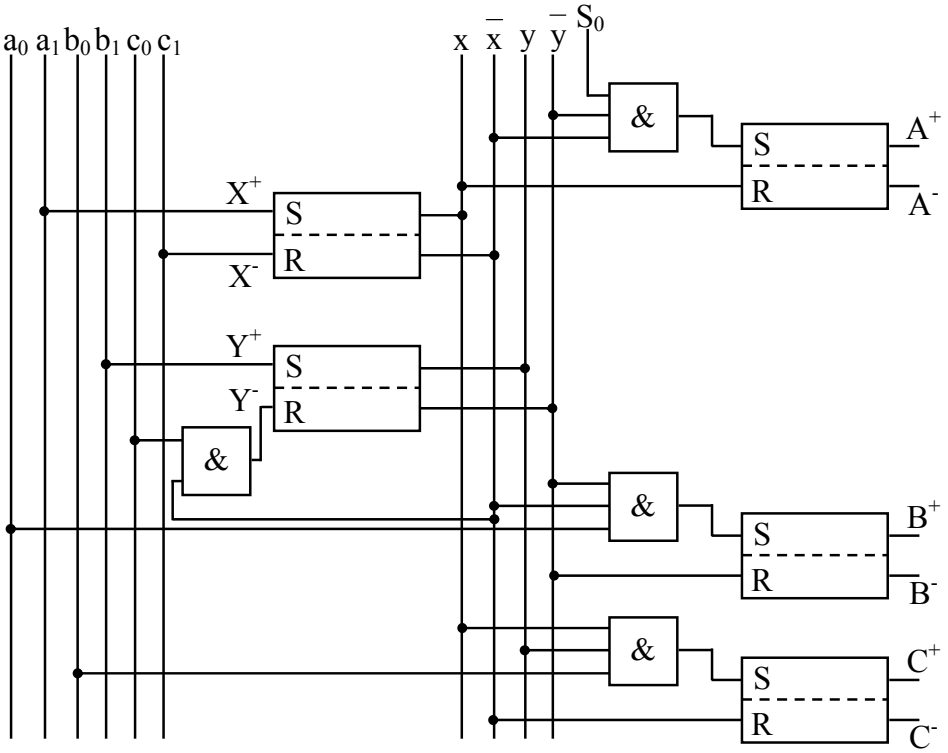
$Y^- = c_0 . \bar{x}$

Phương trình logic của quy trình sau khi đơn giản bằng biểu đồ Karnaugh:

$A^+ = \bar{x} . \bar{y} . S_0$ $B^+ = a_0.x.\bar{y}$ $C^+ = b_0.x.y$ $X^+ = a_1$ $Y^+ = b_1$

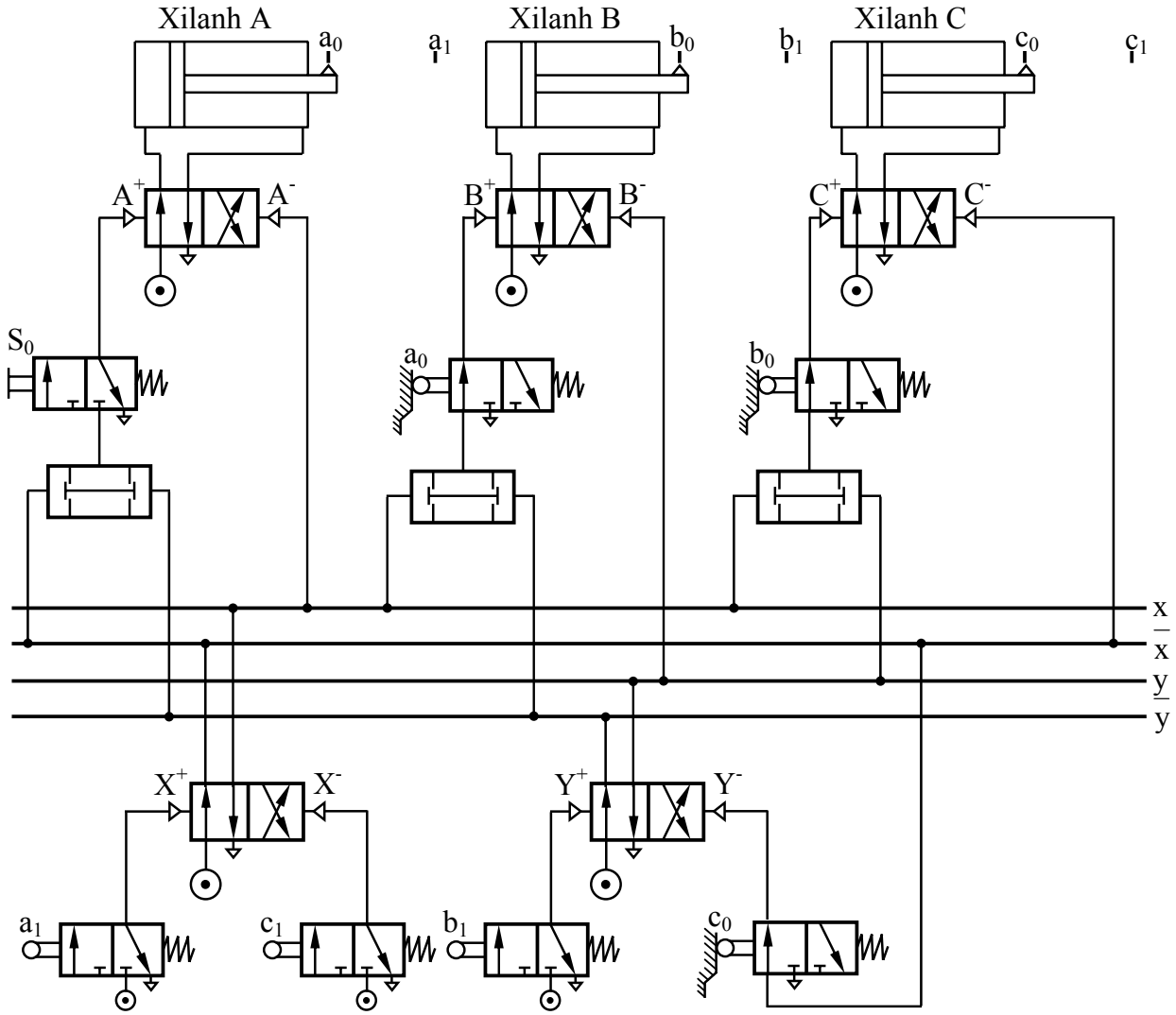
$A^- = x$ $B^- = y$ $C^- = x$ $X^- = c_1$ $Y^- = c_0 . \bar{x}$

Sơ đồ mạch logic sau khi đơn giản bằng biểu đồ Karnaugh:

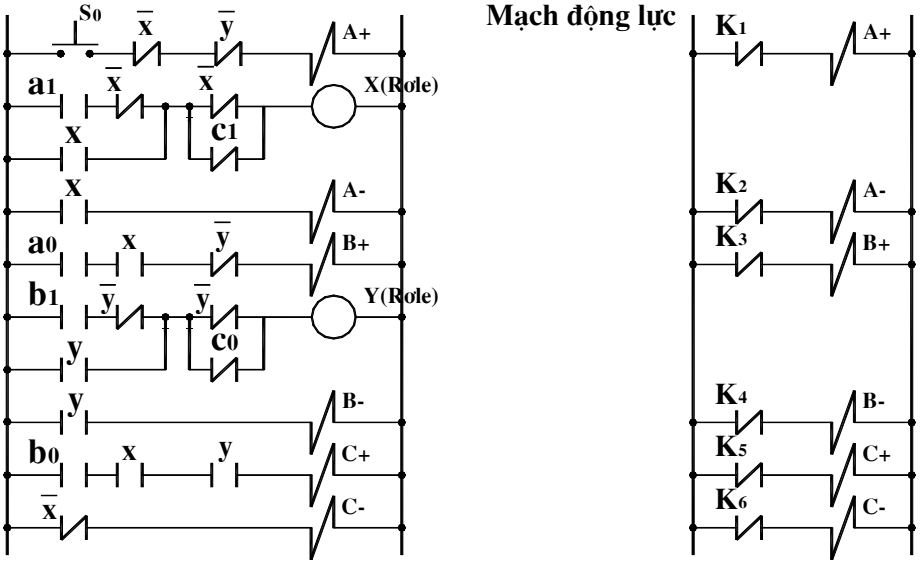


Hình 2.26. Sơ đồ mạch logic

Sơ đồ nguyên lý mạch điều khiển bằng tín hiệu khí nén:



Hình 2.27. Sơ đồ mạch khí nén



Hình 2.28. Sơ đồ nguyên lý mạch điều khiển bằng điện

CHƯƠNG 3: ĐIỀU KHIỂN LOGIC KHẢ LẬP TRÌNH

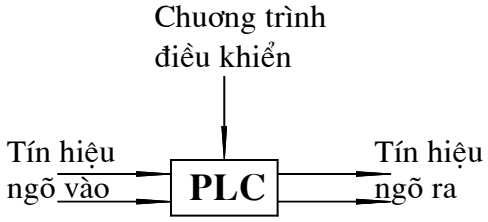
3.1. BỘ ĐIỀU KHIỂN PLC:

3.1.1. Thiết bị điều khiển logic lập trình:

PLC đầu tiên xuất hiện vào năm 1969. Ngày nay chúng ta được sử dụng rộng rãi. Từ các thiết bị nhỏ, độc lập sử dụng khoảng 20 đầu vào/đầu ra digital đến các hệ thống nối ghép theo mạch module có thể sử dụng rất nhiều đầu vào/đầu ra, xử lý các tín hiệu digital hoặc analog. Ngoài ra, chúng còn thực hiện các chế độ điều khiển tỷ lệ-tích phân-đạo hàm (PID)

Thiết bị logic lập trình được (PLC- Programmable Logic Controler) là dạng thiết bị điều khiển đặc biệt dựa trên bộ vi xử lý, sử dụng bộ nhớ lập trình được để lưu trữ các lệnh và thực hiện các chức năng. Chẳng hạn phép tính logic, định giờ, đếm, thuật toán để điều khiển máy và các quá trình.

PLC được thiết kế để dễ cài đặt hoặc thay đổi chương trình. Thuật ngữ logic được sử dụng vì việc lập trình chủ yếu liên quan đến các hoạt logic thực thi và chuyển mạch.



Hình 1.1 Thiết điều khiển logic lập trình

Các thiết bị nhập (bộ cảm biến, các công tắc,...) và các thiết bị xuất trong hệ thống được điều khiển (các động cơ, các van,...) được nối kết với PLC. Thiết bị điều khiển sẽ giám sát các tín hiệu vào và các tín hiệu ra theo chương trình này và thực hiện các quy tắc điều khiển đã được lập trình.

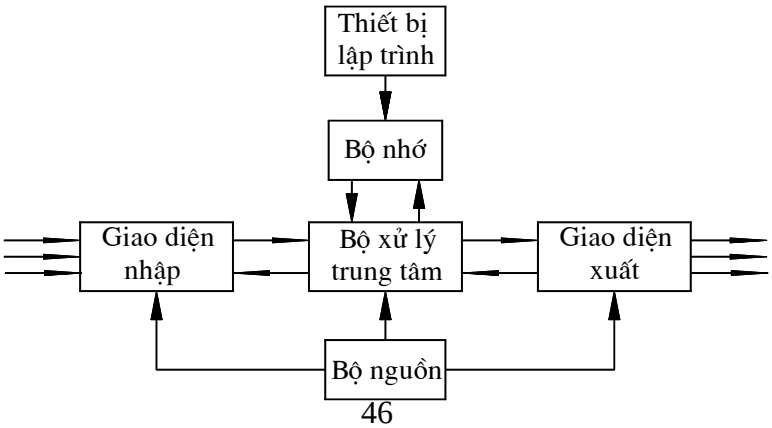
Các PLC có ưu điểm chính là có thể sử dụng cùng một thiết bị điều khiển cơ bản cho nhiều hệ thống điều khiển. Để sửa đổi hệ thống điều khiển và các quy tắc đang được sử dụng, người vận hành chỉ cần nhập tập lệnh khác (không cần mắc nối lại dây). Nhờ vậy, hệ thống rất linh hoạt, hiệu quả.

Các PLC tương tự máy tính, nhưng máy tính được tối ưu hóa cho các tác vụ tính toán và hiển thị; còn PLC được chuyên biệt cho các tác vụ điều khiển và môi trường công nghiệp. Vì vậy, các PLC:

- Được thiết kế và tăng bền để chịu đựoc rung động, nhiệt, ẩm và tiếng ồn
- Có sẵn giao diện cho các thiết bị nhập và xuất
- Được lập trình dễ dàng với ngôn ngữ lập trình dễ hiểu, chủ yếu giải quyết các phép toán logic và chuyển mạch.

3.1.2. Phần cứng:

Hệ thống PLC thông dụng có 5 bộ phận cơ bản:



Hình 1.2 Hệ thống PLC

a. Bộ xử lý trung tâm (CPU): Là linhkiện chứa bộ xử lý, biên dịch các tín hiệu nhập và thực hiện các hoạt động điều khiển theo chương trình đươc lưu trong bộ nhớ của CPU, Truyền các quyết định dưới dạng tín hiệu hoạt động đến các thiết bị xuất. Cấu hình CPU tùy thuộc vào bộ vi xử lý. Nói chung, CPU có:

- Bộ thuật toán và logic (ALU) chịu trách nhiệm xử lý dữ liệu, thực hiện các phép toán số học (cộng, trừ) và các phép toán logic AND, OR, NOT và EXCLUSIVE-OR.
- Bộ nhớ (các thanh ghi) bên trong bộ xử lý, được sử dụng để lưu thông tin liên quan đến sự thực thi chương trình.

- Bộ điều khiển được sử dụng để điều khiển chuẩn thời gian của các phép toán

b. Bộ nguồn: Có nhiệm vụ chuyển đổi điện áp AC thành điện áp thấp DC(5V) cần thiết cho bộ xử lý và các mạch điện trong các module giao diện nhập và xuất.

c. Thiết bị lập trình: Sử dụng để lập chương trình cần thiết vào bộ nhớ của bộ xử lý. Chương trình được viết trên thiết bị này, sau đó được chuyển đến bộ nhớ của PLC.

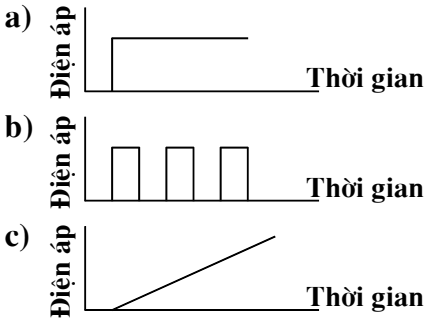
d. Bộ nhớ: Là nơi lưu chương trình được sử dụng cho các hoạt động điều khiển, dưới sự kiểm tra của bộ vi xử lý. Trong PLC có nhiều loại bộ nhớ:

- Bộ nhớ chỉ đọc (ROM) cung cấp dung lượng lưu trữ cho hệ điều hành và dữ liệu cố định được CPU sử dụng.
- Bộ nhớ truy cập ngẫu nhiên (RAM) dành cho chương trình của người dùng
- Bộ nhớ truy cập ngẫu nhiên (RAM) dành cho dữ liệu. Đây là nơi lưu trữ thông tin theo trạng thái của các thiết bị nhập, xuất, các giá trị của đồng hồ thời chuẩn, các bộ đếm và các thiết bị nội vi khác. RAM dữ liệu đôi khi được xem là bảng dữ liệu hoặc bảng ghi. Một phần của bộ nhớ này, khối địa chỉ, dành cho các địa chỉ ngõ vào và ngõ ra, cùng với trạng thái của các ngõ vào và ngõ ra đó. Một phần dành cho dữ liệu được cài đặt trước, và một phần khác dành để lưu trữ các giá trị của bộ đếm, các giá trị của đồng hồ thời chuẩn,...
- Bộ nhớ chỉ đọc có thể xóa và lập trình được (EPROM) là các ROM có thể được lập trình, sau đó chương trình này được thường trú trong ROM.

Người dùng có thể thay đổi chương trình và dữ liệu trong RAM. Tất cả các PLC để lưu chương trình do người dùng cài đặt và dữ liệu chương trình. Tuy nhiên, để tránh mất mát chương trình khi nguồn công suất bị ngắt, PLC sử dụng ắc quy để duy trì nội dung RAM trong một thời gian. Sau khi được cài đặt vào RAM, chương trình có trể được tải vào vi mạch của bộ nhớ EPROM, thường là module có khóa đối với PLC, do đó chương trình trở thành vĩnh cửu. Ngoài ra còn có các bộ đệm tạm thời, lưu trữ các kênh nhập/xuất.

Dung lượng lưu trữ của bộ nhớ đươc xác định bằng số lượng từ nhị phân có thể lưu trữ được. Như vậy, nếu dung lượng bộ nhớ là 256 từ, bộ nhớ đó có thể lưu trữ 256x8= 2048 bit, nếu sử dụng các từ 8 bit, và 256x16= 4096 bit, nếu các từ được sử dụng là 16 bit. Kích cỡ bộ nhớ thường được chuyên biệt theo số lượng vị trí lưu trữ khả dụng với 1K biểu diễn số 2^{10}=1024. Các nhà sản xuất cung cấp vi mạch bộ nhớ với các vị trí lưu trữ theo nhóm 1, 4 và 8 bit. Bộ nhớ 4Kx1x1024 bit vị trí. Bộ nhớ 4Kx8 có 4x8x1024 bit vị trí. Thuật ngữ byte được sử dụng cho từ có độ dài 8 bit. Vì vậy, bộ nhớ 4Kx8 có thể lưu trữ 4096 byte. Với bus địa chỉ 16 bit, bạn có thể có 2^{16} địa chỉ khác nhau, và với các từ 8 bit được lưu trữ ở mỗi địa chỉ, bạn có thể có 2^{16}x8 địa chỉ lưu trữ, và để sử dụng bộ nhớ có dung lượng 2^{16}x8/2^{10}= 64Kx8, bạn có thể có cấu hình gồm bốn vi mạch nhớ 16Kx8.

e. Các phần nhập và xuất: Là nơi bộ xử lý nhận thông tin từ các thiết bị ngoại vi và truyền thông tin đến các thiết bị bên ngoài. Tín hiệu nhập có thể từ các công tắc, các bộ cảm biến, các tế bào quang điện trong cơ cấu đếm, các bộ cảm biến nhiệt độ, các bộ cảm biến lưu lượng,... Các thiết bị xuất có thể đến cuộn dây của bộ khởi động động cơ, các van Solenoid,... Các thiết bị nhập xuất có thể được phân loại theo kiểu tín hiệu cung cấp, rời rạc, digital hoặc analog. Các tín hiệu cung cấp, rời rạc hoặc digital là các thiết bị có tín hiệu ON hoặc OFF. Công tắc là thiết bị cung cấp tín hiệu rời rạc, có hoặc không có điện áp. Về cơ bản, các thiết bik digital có thể được xem là các thiết bị rời rạc, với chuỗi các tín hiệu ON-OFF. Các thiết bị analog cung cấp các tín hiệu có độ lớn tỉ lệ với giá trị của biến đang được giám sát.Ví dụ, bộ cẩm biến nhiệt độ có thể cung cấp điện áp tỉ lệ với nhiệt độ.

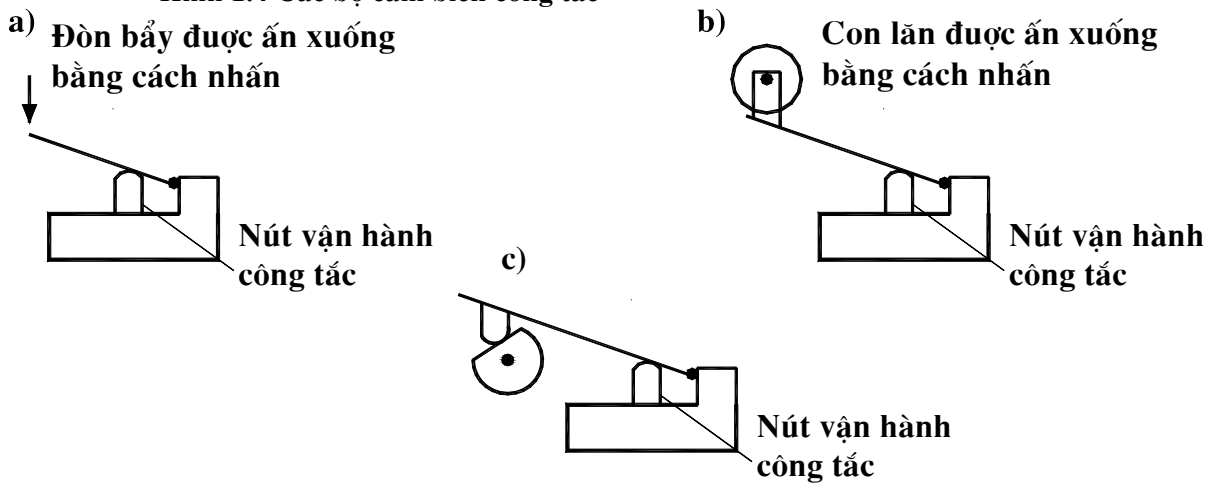


Hình 1.3 Các loại tín hiệu
a)rời rạc; b)digital; c)analog

- Các thiết bị nhập:

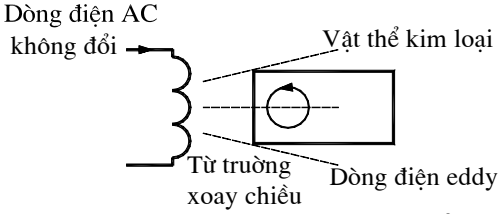
+ Các bộ cảm biến cung cấp tín hiệu digital/rời rạc (có-không), các ngõ ra có thể được nối kết dễ dàng với cổng nhập của PLC. Các bộ cảm biến cung cấp tín hiệu analog phải chuyển thành tín hiệu digital trước khi nhập vào cổng PLC. Sau đây là một số bộ cảm biến thông dụng:

Hình 1.4 Các bộ cảm biến công tắc



Hình 1.5 Các công tắc giới hạn được vận hành bằng a)đòn bẩy; b)con lăn
c) cam (có thể quay với vận tốc không đổi và đóng mở công tắc theo khoảng thời gian nhất định)

+ Các công tắc gián tiếp được sử dụng để phát hiện sự hiện hữu của vật thể mà không

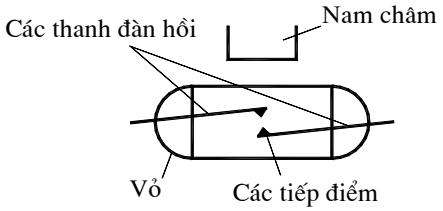


Hình 1.6 Công tắc gián tiếp kiểu dòng điện eddy

tiếp xuác với vật thể đó. Công tắc này có nhiều dạng, một số chỉ phù hợp với các vật thể kim loại.

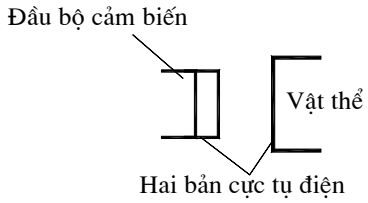
Công tắc gián tiếp kiểu cảm ứng gồm cuộn dây quấn quanh lõi sắt. Khi một đầu của lõi sắt được đặt gần vật thể kim loại có chứa sắt, sẽ có sự thay đổi về lượng của lõi kim loại kết hợp với cuộn dây, do đó, làm thay đổi độ cảm ứng của lõi kim loại. Sự thay đổi này có thể được giám sát bằng mạch cộng hưởng, sự hiện diện của vật thể kim loại có chứa sắt sẽ làm thay đổi dòng điện trong mạch. Dòng điện này có thể được sử dụng để kích hoạt mạch công tắc điện tử, tạo thành thiết bị đóng — ngắt. Vật thể có thể bị phát hiện ở khoảng cách 2 — 15 mm.

+ Công tắc lưỡi gà: Công tắc này gồm hai dải sắt từ đàn hồi, xếp chồng nhưng không tiếp xúc với nhau được gắn vào vỏ thủy tinh hoặc chất dẻo. Khi nam châm hoặc cuộn đay mang dòng điện đến gần công tắc, các dãi sắt sẽ bị từ hóa và hút nhau, làm các tiếp điểm đóng. Nam châm làm đóng các tiếp điểm khi cách công tắc khoảng 1 mm. Vì vậy, công tắc này được sử dụng rộng rãi trong các thiết bị chống trộm để phát hiện khi cửa bị mở; nam châm gắn lên cửa và công tắc lưỡi gà gắn lên khung cửa. Khi cửa mở công tắc sẽ mở.



Hình 1.7 Công tắc luỡi gà

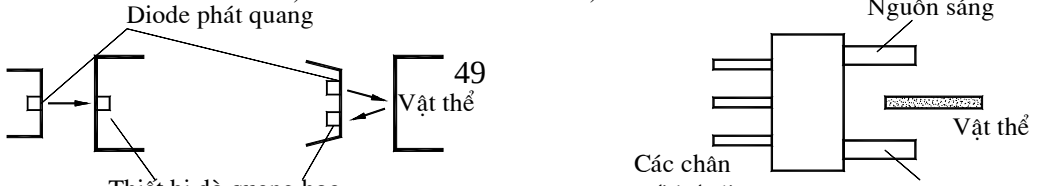
+ Công tắc gián tiếp được sử dụng với các vật thể kim loại và phi kim loại là công tắc kiểu điện dung. Điện dung của tụ được xác định bằng khoảng cách giữa hai bản cực, khoảng cách càng nhỏ điện dung càng cao. Bộ cảm biến của công tắc kiểu điện dung là một trong hai bản cực của tụ điện, bản kia là vật thể kim loại. Sự tiếp cận của vật thể kim loại được phát hiện nhờ sự thay đổi điện dung. Bộ cảm biến cũng có thể được sử dụng để phát hiện nhờ sự thay đổi điện dung. Bộ cảm biến cũng có thể được sử dụng để phát hiện các vật thể kim loại vì điện dung của tụ phụ thuộc vào chất điện môi giữa hai bản. Trong trường hợp này, các bản cực là bộ cảm biến và dây nối đất, vật thể phi kim loại là chất điện môi. Sự thay đổi điện dung có thể được sử dụng để kích hoạt mạch công tắc điện tử và tạo thành thiết bị đóng — ngắt. Công tắc kiểu điện dung có thể được sử dụng để phát hiện các vật thể khi chúng cách đầu bộ cảm biến khoảng 4-60 mm.



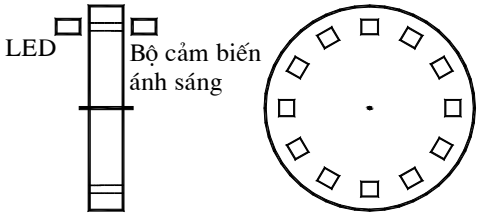
Hình 1.8 Công tắc kiểu điện dung

+ Các thiết bị chuyển mạch quang điện có thể vânh hành theo kiểu truyền phát, vật thể cần phát hiện sẽ chắn chùm sáng (thường là bức xạ hồng ngoại), không cho chúng chiếu tới thiết bị dò (Hình 1.9(a)); hoặc theo kiểu phản xạ, vật thể cần phát hiện sẽ phản chiếu chùm sáng lên thiết bị dò (Hình 1.9(b)). Trong cả hai kiểu, cực phát bức xạ thông thường là diode phát quang (LED). Thiết bị dò bức xạ có thể là transistor quang, thường là hai transistor, được gọi là cặp Darlington. Cặp Darlington làm tăng độ nhạy của thiết bị. Tùy theo mạch được sử dụng, đầu ra có thể được chế tạo để chuyển mạch đến mức cao hoặc mức thấp khi ánh sáng đến transistor. Các bộ cảm biến được cung cấp dưới dạng các hộp cảm nhận sự có mặt của các vật thể ở khoảng cách ngắn, thường nhỏ hơn 5 mm. Hình 1.9(c) minh họa bộ cảm biến chữ U, trong đó vật thể ngăn chặn chùm sáng.

a) b) c)



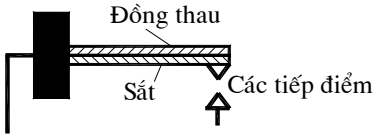
+ Bộ mã hóa: Thuật ngữ mã hóa được sử dụng cho thiết bị cung cấp tín hiệu ra digital theo sự dịch chuyển góc hoặc tuyến tính. Bộ mã hóa gia số tìm các thay đổi chuyển dịch góc hoặc tuyến tính từ vị trí chuẩn cho trước, còn bộ mã hóa tuyệt đối cung cấp vị trí góc hoặc tuyến tính thực tế.



Hình 1.10 Dạng cơ bản của bộ mã hóa gia số

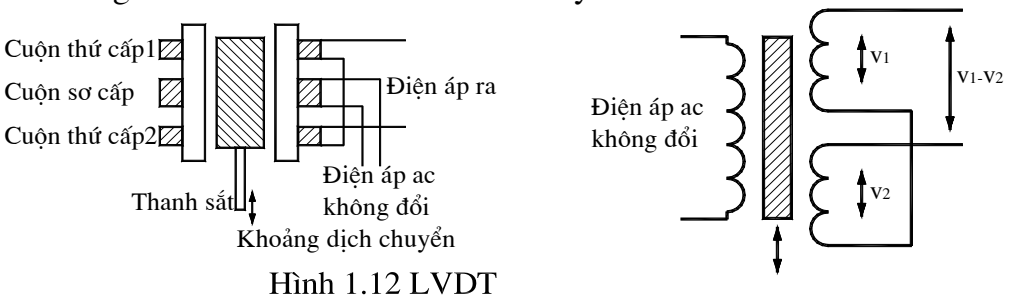
+ Các bộ cảm biến nhiệt độ:

Dạng đơn giản của bộ cảm biến nhiệt độ có thể được sử dụng để cung cấp tín hiệu đóng-ngắt khi nhiệt độ đạt đến giá trị xác định, là phần tử lưỡng kim . Phần tử này gồm hai dải kim loại khác nhau, ví dụ, đồng thau và sắt, được gắn với nhau. Hai kim loại này có hệ số dãn nở khác nhau. Khi nhiệt độ tăng, dải lưỡng kim sẽ uốn cong, do một trong hai kim loại có hệ số dãn nở nhiệt lớn hơn. Kim loại dãn nở cao hơn sẽ ở mặt lồi của phần cong. Khi nguội, hiệu ứng uốn cong xảy ra theo chiều ngược lại. Sự chuyển động này của dải lưỡng kim có thể được sử dụng để ngắt các tiếp xúc điện, từ đó, ở nhiệt độ nhất định, sẽ đóng-ngắt dòng điện trong mạch. Thiết dị này có độ chính xác cao, nhưng được sử dụng phổ biến trong các bộ điều nhiệt của hệ thống nhiệt gia dụng.



Hình 1.11 Dạng cơ bản của bộ mã hóa gia số

+ Các bộ cảm biến khoảng dịch chuyển: là biến áp vi sai biến thiên tuyến tính (LVDT), thiết bị này cung cáp điện áp ra theo vị trí của thanh sắt. LVDT gồm ba cuộn dây đối xứng suốt hành trình thanh sắt di chuyển.



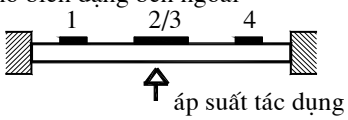
Hình 1.12 LVDT

Khi dòng điện xoay chiều được đưa vào cuộn sơ cấp, điện áp xoay chiều được tạo ra trong hai cuộn dây thứ cấp. Khi lõi sắt ở chính giữa hai cuộn dây thứ cấp, điện áp sinh ra trong hai cuộn thứ cấp bằng nhau. Các đầu ra từ hai cuộn dây thứ cấp được nối kết sao cho tín hiệu ra kết hợp của chúng khác với điện áp của hai cuộn dây thứ cấp. Khi thanh sắt ở chính giữa, điện áp xoay chiều trên hai cuộn thứ cấp bằng nhau, vì vậy, không có điện áp ra. Khi thanh sắt dịch chuyển ra khỏi vị trí giữa, lệch về phía một trong hai cuộn dây thứ cấp không bằng nhau. Sự chênh lệch điện áp giữa hai cuộn dây thứ cấp phụ thuộc vào vị trí của thanh sắt. Điện áp ra từ LVDT là điện áp xoay chiều. Điện áp này thường được chuyển thành điện áp dc. Analog và được khuếch đại trước khi dẫn vào kênh analog của PLC.

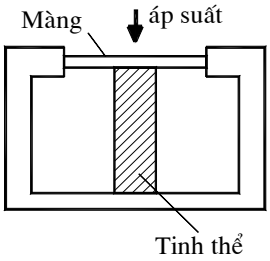
+ Các bộ cảm biến áp suất:

Các bộ cảm biến áp suất thông dụng cung cấp các đáp ứng liên quan đến áp suất là kiểu màng và kiểu xếp. Kiểu màng gồm một đĩa mỏng bằng kim loại hoặc chất dẻo, được định vị theo chu vi. Khi áp suất ở hai phía của màng khác nahu, tâm màng bị lệch. Độ lệch này tương ứng với chênh lệch áp suất ở hai phía, và có thể phát hiện nhờ các đồng hồ biến dạng được gắn với màng (Hình 1.13(a)), hoặc sử dụng độ lệch này để nén tính thể áp điện (Hình 1.13(b)). Khi tinh thể điện áp bị nén, sẽ có sự chuyển dịch tương đối các điện tích dsương và âm trong trong tinh thể đó và các bề mặt phía ngoài của tinh thể sẽ tích điện. Do đó hiệu điện thế xuất hiện.

a) Bốn biến dạng kế:
2 dùng cho biến dạng tâm
2 cho biến dạng bên ngoài



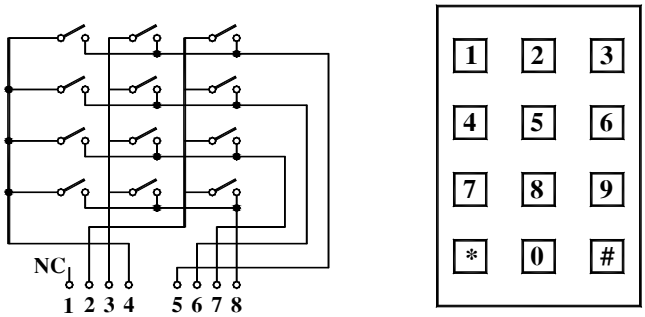
b)



Hình 1.13 Bộ cảm biến áp suất
a)Bộ cảm biến lực; b)Kiểu áp điện

+ Bàn phím:

Nhiều máy sử dụng bàn phím nhỏ để nhập các lệnh xác lập điều kiện đwocj yêu cầu cho các ngõ ra, nhiệt độ hoặc tốc độ. Các bàn phím này thường có các nút khi được nhấn xuống sẽ vận hành các đệm cao su silicon dẫn điện để thực hiện các tiếp xúc. Thay vì nối từng phím riêng lẻ và dùng 12 đầu vào, các phím được nối kết thành hàng và cột, việc ấn phím riêng lẻ có thể cung cấp đầu ra theo cột và đầu ra theo hàng duy nhất cho phím đó. Điều này làm giảm đầu vào cần thiết cho PLC.



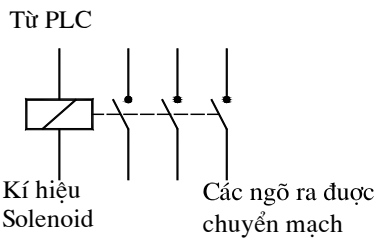
Hình 1.14 Bàn phím 12 chiều

- Các thiết bị xuất:

Các cổng ra của PLC có kiểu rơle hoặc bộ cách điện quang với các kiểu transistor hoặc triac tùy theo các thiết bị được nối kết với chúng sẽ được đóng hoặc mở. Nói chung, tín hiệu digital từ kênh xuất của PLC được sử dụng để điều khiển thiết bị kích hoạt, sau đó thiết bị kích hoạt điều khiển quá trình nào đó. Thuật ngữ thiết bị kích hoạt được sử dụng cho thiết bị biến đổi tín hiệu điện thành hoạt động có công suất cao hơn, sau đó hoạt động này sẽ điều khiển quá trình.

+ Công tắc tơ:

Các Solenoid quyết định số lượng thiết Bị kích hoạt điều khiển ngõ ra. Khi dòng điện đi qua Solenoid, từ trường được sinh ra, từ trường này có thể hút các bộ phận kim loại sắt trong vùng lân cận. Về bản chất, Contactor là một dạng Rơle, sự khác nhau là thuật ngữ Rơle được sử

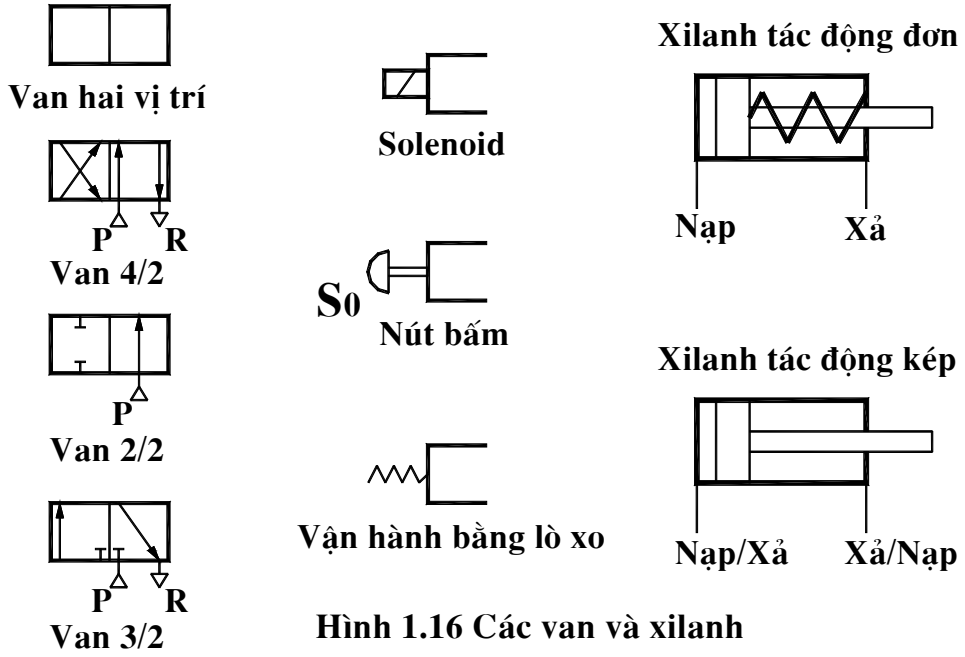


Hình 1.15 Contactor

Dụng cho thiết bị chuyển mạch các dòng điện nhỏ,
thấp hơn 10A, còn thuật ngữ contactor được sử dụng
cho thiết bị chuyển mạch dòng điện lớn, có thể đến hàng trăm ampere.

+ Các van điều khiển hướng:

Một ví dụ khác về việc sử dụng Solenoid làm thiết bị kích hoạt là van vận hành bằng Solenoid. Van này có thể được sử dụng để điều khiển hướng lưu thông của khí nén hoặc dầu ép, và cũng được sử dụng để vận hành các thiết bị khác, chẳng hạn, chuyển động của piston trong xilanh.

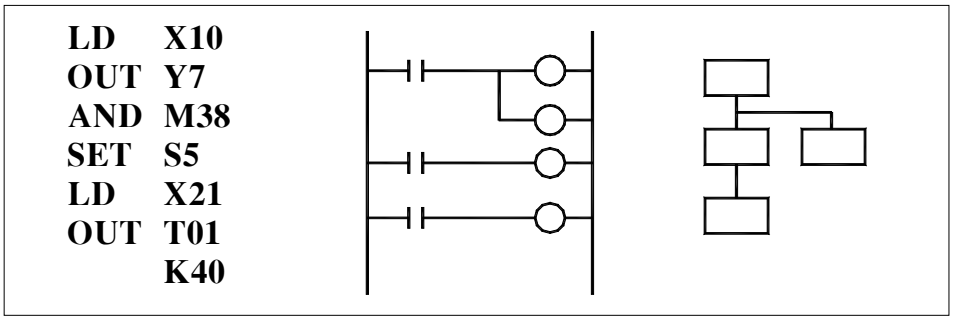


Hình 1.16 Các van và xilanh

3.2. CÁC THIẾT BỊ LOGIC CHUẨN

3.2.1. Chương trình:

Chương trình là một chuỗi các lệnh nối tiếp nhau dược viết theo ngôn ngữ mà PC có thể hiểu được. Có 3 dạng chương trình: instruction, ladder và SFC/STL. Không phải tất cả các công cụ lập trình đều có thể làm việc được cả 3 dạng trên. Bộ lập trình bằng tay chỉ làm việc được với dạng instruction trong khi hầu hết các công cụ lập trình đồ họa sẽ làm việc cả dạng instruction và ladder. Các phần mềm chuyên dụng sẽ cho phép làm việc với dạng SFC.



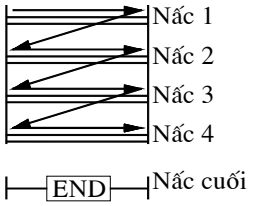
Dạng instruction **Dạng ladder** **Dạng SFC**

3.2.2. Các thiết bị cơ bản dùng trong lập trình: Có 6 thiết bị cơ bản "thiết bị bit", nghĩa là các thiết bị này có hai trạng thái ON hoặc OFF, 1 hoặc 0.

Có nhiều ngôn ngữ lập trình nhưng mục đích việc ssử dụng nhắm đến người không đòi hỏi kiến thức cao về lập trình. Do đó, việc lập trình kiểu bậc thang được nghiên cứu và ứng dụng. Đây là phương pháp viết chương trình có thể chuyển thành mã máy nhờ phần mềm chuyên dùng cho bộ vi xử lý của PLC.

- X hoặc I: dùng để chỉ ngõ vào vật lý gắn trực tiếp vào PC
- Y hoặc Q hoặc O: dùng để chỉ ngõ ra nối trực tiếp từ PC
- T: dùng để xác định giờ có trong PC
- C: dùng để xác định thiết bị đếm có trong PC
- M và S: dùng như là các Pơle hoạt động trong PC

Để vẽ sơ đồ thang, cần tuân thủ các quy ước sau:



Hình 1.17 Quét chương trình thang

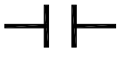
- Các đường dọc trên sơ đồ biểu diễn đường công suất, các mạch được nối kết giữa các đường này.
- Mỗi nấc thang xác định một hoạt động trong quá trình điều khiển.
- Sơ đồ thang được đọc từ trái sang phải và Từ trên xuống. Nấc ở đỉnh thang được đọc từ trái sang phải. Tiếp theo, nấc thứ hai tính từ trên xuống được đọc từ trái sang phải,... Khi ở chế độ hoạt động, PLC sẽ đi từ đầu đến cuối chương trình thang, nấc cuối của chương trình thang được ghi chú rõ ràng, sau đó lặp lại như đầu. Qúa trình lần lượt đi qua tất cả các nấc của chương trình được gọi là chu trình.
- Mỗi nấc bắt đầu với một hoặc nhiều ngõ vào và kết thúc với ít nhất một ngõ ra. Thuật ngữ ngõ vào được dùng cho hoạt động điều khiển, chẳng hạn đóng các tiếp điểm công tắc , được dùng làm ngõ vào PLC. Thuật ngữ ngõ ra được sử dụng cho thiết bị được nối kết với ngõ ra của PLC.
- Các thiết bị điện được trình bày ở điều kiện chuẩn của chúng. Vì vậy, công tắc thường mở được trình bày trên sơ đồ thang ở trạng thái mở. Công tắc thường đóng được trình bày ở trạng thái đóng.
- Thiết bị bất kỳ có thể xuất hiện trên nhiều các nấc thang. Ví dụ, có thể có rơle đóng mạch một hoặc nhiều thiết bị. Các mẫu tự và/hoặc các số giống nhau được sử dụng để ghi nhãn cho thiết bị trong từng trường hợp.
- Các ngõ vào và ra được nhận biết theo địa chỉ của chúng, ký hiệu tùy theo nhà sản xuất PLC. Đó là địa chỉ ngõ vào hoặc ngõ ra trong bộ nhớ PLC. Các PLC Mitsubishi series F sử dụng mẫu tự X đứng trước các phần tử nhập, Y đứng trước các phần tử xuất, và sử dụng các số theo sau:

  Các ngõ vào: X400-407, 500-507, 510-513 (24 ngõ vào khả dĩ)

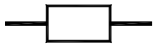
  Các ngõ ra: Y-437, 530-537 (16 ngõ ra khả dĩ)

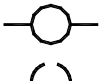
Toshiba cuãng sử dụng mẫu tự X và Y với các ngõ vào, chẳng hạn, X000 và X001, cà các ngõ ra Y000 và Y001. Siemens sử dụng mẫu tự I cho ngõ vào và Q cho ngõ ra, ví dụ, I0.1 và Q2.0. Sprecher+Schuh đánh số ngõ vào bằng X và ngõ ra bằng Y, ví dụ, X001 và Y001. Allen Bradley sử dụng I và O, ví dụ, I:21/01 và O:22/01.

Các ký hiệu tiêu chuẩn được sử dụng cho thiết bị nhập và xuất:


Các tiếp điểm ngõ vào thường mở


Các tiếp điểm ngõ vào thường đóng

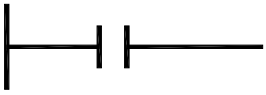
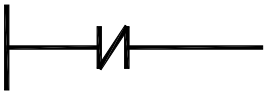

Lệnh đặc biệt


hoặc **Thiết bị xuất**

3.2.3. Các lệnh cơ bản:

Phương pháp này ssử dụng mã nhớ, mỗi mã tương ứng với một thành phần của thang. Các mã được sử dụng khác nhau tùy theo nhà sản xuất, mặc dù tiêu chuẩn IEC 1131-3 đã đwocj đề sản xuất. Đối với nắc khởi đầu, luôn luôn phải sử dụng mã nấc khởi đầu. Mã này có thể là LD, A, L hoặc STL, để biểu thị nấc thang khởi đầu với các tiếp điểm mở; hoặc LDI, LND, LD NOT, AN, LN hoặc STR NOT, để cho biết nấc khởi đầu với các tiếp điểm đóng. Tất cả các nấc phải kết thúc bằng ngõ ra. Mã ngõ ra có thể là OUT hoặc =.

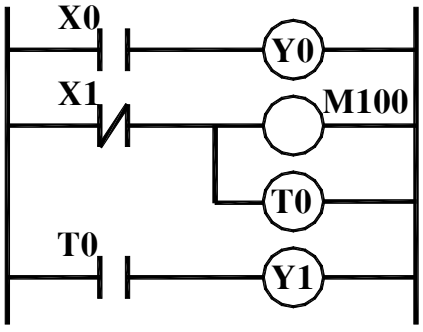
a. Load, Load Inverse:

Lệnh gợi nhớ	Chức năng	Dạng mẫu	Thiết bị	Số bước chương trình
LD (LoaD)	Tác vụ logic khởi tạo-loại contact NO		X, Y, M, S, T, C	1
LD (LoaD Inverse)	Tác vụ logic khởi tạo-loại contact NC		X, Y, M, S, T, C	**1**

Các đặc điểm cơ bản cần nhớ:

- Lệnh LD và LDI nối trực tiếp đầu bên trái
- Lệnh LD và LDI cũng được dùng để xavs định một khối chương trình khi dùng lệnh ORB và ANB.

Ví dụ:




```
LD      X    0
OUT     Y    0
LDI     X    1
OUT     M    100
SPK     19
LD      T    0
OUT     Y    1
```

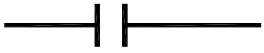
b. Out:

Lệnh gợi nhớ	Chức năng	Dạng mẫu	Thiết bị	Số bước chương trình
OUT	Tác vụ logic cuối- loại		Y, M,S,	Y, M: 1 S, cuộn M chuyên dùng:2

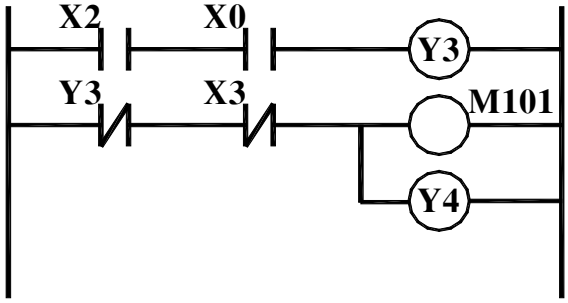
Các điểm cơ bản cần nhớ:

- Lệnh OUT nối trực tiếp với đầu bên phải
- Lệnh OUT không thể dùng để điều khiển thiết bị ngõ vào loại X
- Nhiều lệnh OUT có thể được nối song song

c. And, And Inverse:

Lệnh gợi nhớ	Chức năng	Dạng mẫu	Thiết bị	Số bước chương trình
AND (AND)	Nối tiếp các contact NO (thường mở)		X, Y, M, S, T, C	1
ANI (And Inverse)	Nối tiếp các contact NC (thường đóng)		X, Y, M, S, T, C	1

Ví dụ:

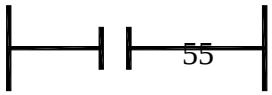



```
LD    X  2
AND   X  0
OUT   Y  3
ANI   X  3
OUT   M  101
AND   T  1
OUT   Y  1
```

Các điểm cơ bản cần nhớ:

- Lệnh AND và ANI được dùng để nối tiếp thêm một contact. Có thể nối nhiều contact thành một chuỗi nối tiếp nếu cần.
- Việc sử lý thêm một cuộn dây qua một contact, lệnh OUT đầu tiên được gọi là ngõ ra "follow-on". Các ngõ ra follow-on cho phép xử lý ngõ ra theo đúng trình tự đã nghi.

d. Or, Or Inverse:

Lệnh gợi nhớ	Chức năng	Dạng mẫu	Thiết bị	Số bước chương trình
OR (OR)	Nối song song các contact NO (thường mở)		X, Y, M, S, T, C	1

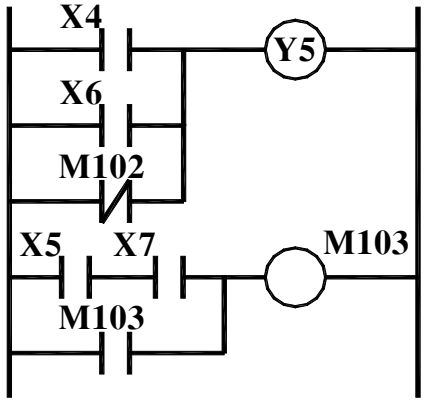
Các đặc điểm cần nhớ:

- Lệnh OR và ORI được dùng để nối song song một contact. Để nối một khối có nhiều contact nối tiếp, song song với khối khác, ta dùng lệnh ORB.
- Một bên của lệnh OR/ORI luôn nối với đầu bên trái

Giới hạn ngoại vi:

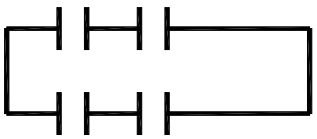
Mặc dù không giới hạn số contact mắc song song, nhưng một số bảng điều khiển lập trình, màn hình và máy in sẽ không thể nào hiển thị hoặc in chương trình nếu nó vượt giới hạn của phần cứng. Mỗi dòng hoặc mỗi nhánh của chương trình ladder nên chứa tối đa 10 contact và 1 cuộn dây. Số ngõ ra follow-on nên giới hạn tối đa là 24.

Ví dụ:



LD	**X**	**4**
OR	**X**	**6**
ORI	**M**	**102**
OUT	**Y**	**5**
LD	**X**	**5**
AND	**X**	**7**
OR	**M**	**103**
OUT	**M**	**103**

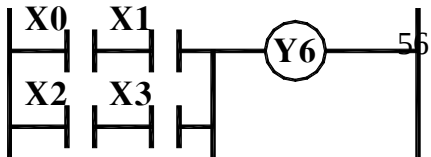
e. Or Block:

Lệnh gợi nhớ	Chức năng	Dạng mẫu	Thiết bị	Số bước chương trình
ORB (OR Block)	Nối song song nhiều mạch contact		X, Y, M, S, T, C	1

Các điểm cơ bản cần nhớ:

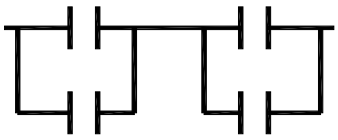
- Lệnh ORB là độc lập và không kết hợp với bất kỳ thiết bị hay con số nào
- Lệnh ORB được dùng để nối song song nhiều mạch contact (thường là các khối nối tiếp) với khối phía trước. Các khối nối tiếp là các khối có nhiều contact nối tiếp nhau hay dùng trong lệnh ANB.
- Để khai báo điểm đầu của một khối dùng lệnh LD hay LDI. Sau một khối nối tiếp, nối nó vào khối trước bằng lệnh ORB.

Ví dụ:



LD	**X**	**0**
AND	**X**	**1**
LD	**X**	**2**
AND	**X**	**3**

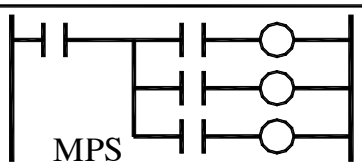
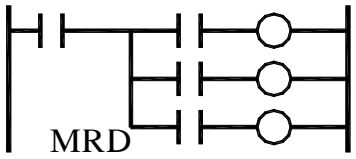
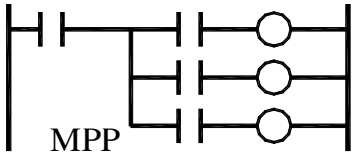
f. And Block:

Lệnh gợi nhớ	Chức năng	Dạng mẫu	Thiết bị	Số bước chương trình
ANB (And Block)	Nối tiếp các mạch song song		X, Y, M, S, T, C	1

Các điểm cơ bản cần nhớ:

- Lệnh ANB là độc lập và không kết hợp với bất kỳ thiết bị hay con số nào
- Lệnh ANB được dùng để nối tiếp nhiều mạch contact (thường là các khối song song), với khối phía trước. Các khối song song là các khối có nhiều contact nối song song nhau hay dùng trong lệnh ORB.
- Để khai báo điểm bắt đầu của một khối dùng lệnh LD hay LDI. Sau một khối nối tiếp, nối nó vào khối trước bằng lệnh ANB.

g. MPS, MRD và MPP:

Lệnh gợi nhớ	Chức năng	Dạng mẫu	Thiết bị	Số bước chương trình
MPS (Point Store)	Luu kết quả hiện hành của các tác vụ trong PC		Không có	1
MRD (Read)	Đọc kết quả hiện hành của các tác vụ trong PC		Không có	1
MPP (PoP)	Lấy ra(gọi ra và loại bỏ) kết quả đã luu		Không có	1

Các điểm cơ bản cần nhớ:

- Các lệnh này cần để nối các cuộn dây ngõ ra vào bên trái của bất kỳ contact nào. Nếu không có thì lệnh này chỉ có thể nối vào bên phải của contact cuối cùng.
- MPS lưu điểm nối của mạch ladder, vì vậy nó có thể được gọi lại nhiều lần để nối nhiều nhánh cuộn dây hơn.
- MRD gọi lại hoặc đọc dữ liệu của điểm nối đã được lưu trước đó và nối nó với contact tiếp theo.
- MPP lấy ra (gọi ra và loại bỏ) điểm nối đã lưu từ vùng lưu tạm sau khi contact kế bị buộc nối vào điểm này.
- Đối với mỗi lệnh MPS phải có một lệnh MPP tương ứng.
- Lệnh MPP phải được dùng để nối mạch contact/cuộn dây cuối cùng.
- Ở bất kỳ bước lập trình nào, số cặp lệnh MPS-MPP hoạt động phải không được quá 11.

Cách dùng MPS, MRD và MPP:

- Khi chương trình được viết ở dạng ladder thì tất cả các lệnh MPS, MRD, MPP sẽ tự động được thêm vào khi thực hiện chuyển chương trình (program conversion) sang dạng intruction. Nừu xem chương trình dạng intruction sau khi chuyển se có các lệnh MPS, MRD, MPP.
- Khi chương trình được viết ở dạng intruction, người sử dụng phải nhập vào toàn bộ tất cả các lệnh MPS, MRD và MPP cần thiết.

h. Master Control và Master Control Reset:

Lệnh gợi nhớ	Chức năng	Dạng mẫu	Thiết bị	Số bước chương trình
MC (Master Control)	Chỉ ra điểm đầu của một khối điều khiển chính (master control block)		Y, M (cho phép thêm cuộn M chuyên dùng loại NO)	1
MCR (Master Control Reset)	Chỉ ra điểm kết thúc của một khối điều khiển chính		N được đặt lại	1

N: Chỉ mức lồng (N0 đến N7)

Các điểm cơ bản cần nhớ:

- Khi ngõ vào X0=ON thì tất cả các lệnh giữa MC và MCP được thi hành. Khi X0=OFF, tất cả các thiết bị được đặt lại (reset) trừ các bộ định thì, bộ đếm và các thiết bị được điều khiển bằng lệnh SET/RST.
- Sau khi thực hiện lệnh MC, đường bus (tại điểm LD, LDI) dịch chuyển đến điểm sau lệnh MC. Lệnh MCR đưa điểm này vào đường bus ban đầu. Sau khi lệnh MC được thiết lập, cần phải thêm một con trỏ lồng mức N, số mức lồng có thể chọn từ khoảng N0 tới N7. Mức lồng cao nhất là "0" và thấp nhất là "7". Mỗi mức lồng có thể được đặt lại (reset) bằng cách chỉ định mức trong lệnh MCR. Khi mức lồng bị đặt lại thì tất cả các mức thấp hơn nó cũng đặt lại.

- Lệnh MC cũng có thể dùng nhiều lần nếu cần thiết, bằng cách thay đổi con số nhận dạng của thiết bị Y và M. Nếu dùng cùng một số nhận dạng thì nó xử lý như là cuộn dây kép. Các mức lồng có thể được gấp đôi lên nhưng khi chúng bị reset thì tất cả các mức trong đó đều bị reset chứ không phải chỉ một mức ghi trong lệnh MC.

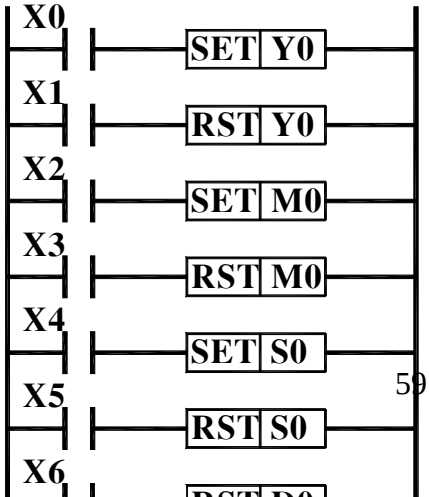
i. Set và Reset:

Lệnh gợi nhớ	Chức năng	Dạng mẫu	Thiết bị	Số bước chương trình
SET (SET)	Đặt một thiết bị bit lên ON (vĩnh viễn)	SET N	Y, M, S	Y, M: 1 S, cuộn chuyên dùng: 2
RST (ReSet)	Đặt một thiết bị bit xuống OFF (vĩnh viễn)	RST N	Y, M, S D, V, Z	D, thanh ghi D chuyên dùng, V và Z: 3

Các điểm cơ bản cần nhớ:

- Một khi X0 bật on, Y0 hoạt động và duy trì ON ngay cả sau khi X1 tắt OFF. Khi X1 bật ON, Y0 tắt OFF và duy trì OFF ngay cả sau khi X1 tự nó chuyển thành OFF (điều này cũng đúng đối với M0 và S0 trong ví dụ)
- SET và RST có thể được dùng cho cùng một thiết bị bao nhiêu lần tùy ý. Tuy nhiên, trạng thái của lệnh cuối cùng được kích hoạt mới là trạng thái có ảnh hưởng.
- Lệnh RST cúng có thể được dùng để reset nội dung của các dữ liệu như yhanh ghi dữ liệu (dât register), thanh ghi chỉ mục (index register)...Hiệu quả tương đương với việc chuyển "KO" vào thiết bị dữ liệu.

Ví dụ:




```
LD    X  0
SET   Y  0
LD    X  1
RST   Y  0
LD    X  2
SET   M  0
LD    X  3
RST   M  0
LD    X  4
SET   S  0
LD    X  5
RST   S  0
LD    X  6
```

j. Bộ định thì và bộ đếm (Out and Reset)

Lệnh gợi nhớ	Chức năng	Dạng mẫu	Thiết bị	Số bước chương trình
OUT (OUT)	Điều khiển một cuộn dây bộ định thì hoặc bộ đếm		T, C	Bộ đếm 32 bit: 5 Khác: 3
RST (ReSet)	Đặt lại bộ định thì và bộ đếm, cuộn dây, contact và các giá trị hiện hành		Y, M, S D, V, Z	D, thanh ghi D chuyên dùng, V và Z: 3

k. Xung cạnh lên và xung cạnh xuống:

Lệnh gợi nhớ	Chức năng	Dạng mẫu	Thiết bị	Số bước chương trình
PLS (PuLSe)	Kích xung khi có cạnh lên		Y, M (không cho phép dùng cuộn M chuyên dùng)	2
PLF (PuLSe Failling)	Kích xung khi có cạnh lên		Y, M (không cho phép dùng cuộn M chuyên dùng)	2

Các điểm cơ bản cần nhớ:

- Khi lệnh PLS được thi hành, các thiết bị Y và M hoạt động trong khoảng thời gian một chu kỳ sau khi tín hiệu ngõ vào đã bật ON.
- Khi lệnh PLF được thi hành, Y và M HOạT Đẫng trong khoảng thời gian một chu kỳ sau khi tín hiệu ngõ vào đã tắt OFF.

- Khi trạng thái của PC thay đổi từ RUN sang STOP và trở lại RUN với tín hiệu ngõ vào vẫn là ON, thì PLS M0 hoạt động trở lại. Tuy nhiên nếu dùng cuộn dây M có nguồn pin nuôi (được chốt) thay cho M0 thì nó sẽ không hoạt động lại. Đối với thiết bị được chốt để có thể hoạt động lại thì ngõ vào điều khiển phải tắt OFF trong quá trình chuyển trạng thái RUN/STOP/RUN trước khi nó được kích một lần nữa.

k. No Operation và End:

Các điểm cơ bản cần nhớ:

- Thêm lệnh NOP trong chương trình để giảm tối thiểu sự thay đổi số bước chương trình khi thay đổi hay soạn thảo chương trình.
- Có thể hoạt động của một mạch bằng cách thay lệnh đã lập trình bằng lệnh NOP.
- Thay lệnh LD, LDI, ANB hoặc ORB bằng lệnh NOP sẽ làm cho mạch thay đổi đáng kể, có thể gây ra lỗi ở nhiều nơi trong chương trình.
- Sau khi thực hiện chức năng "All clear operation" thì tất cả các lệnh hiện hành trong chương trình sẽ được ghi chồng bằng lệnh NOP.
- Khi đặt lệnh END trong chương trình có tác dụng buộc kết thúc quá trình quét chương trình hiện hành và tiến hành cập nhật các ngõ vào và ngõ ra.
- Chèn lệnh END vào giữa chương trình giúp tìm lỗi cho chương trình vì phần sau lệnh END bị vô hiệu và cách ly khỏi vùng kiểm lỗi. Nhớ xóa các lệnh END khỏi những khối đã kiểm tra rồi.
- Khi lệnh END được thi hành thì bộ định thì watchdog tự động được làm tươi.

3.2.4. Các lệnh ứng dụng: Các hàm từ FNC 0 ÷ FNC 9

- CJ (Condition Jump): Nhảy đến con trỏ đích đã định
- CALL (Call Subroutine): Gọi chương trình con hoạt động
- SRET (Subroutine Return): Trở về từ trình con
- IRET (Interrupt Return): Trở về từ chương trình ngắt
- EI (Enable Interupt): Cho phép các ngõ vào ngắt
- DI (Disable Interupt): Vô hiệu việc xử lý chương trình ngắt
- I (Interupt Pointer): Chỉ định điểm bắt đầu của một chương trình ngắt
- FEND (First End): Dùng để chỉ cuối khối chương trình chính
- WDT (Watchdog Timer reset): Dùng để làm tươi bộ định thì watchdog trong suốt thời gian quét chương trình
- FOR (Start of a For/Next Loop): Xác định vị trí bắt đầu và số lần lặp của vòng lặp
- NEXT (End a For/Next Loop): Xác định vị trí cuối của vòng lặp

3.2.5. Nhóm lệnh về dịch chuyển và so sánh: Các hàm từ FNC 10 ÷ FNC 19

- CMP (Compare): So sángh hai giá trị dữ liệu, cho biết kết quả <, = và >

- ZCP (Zone Compare): So sánh dữ liệu với một giá trị của dữ liệu, cho biết kết quả <, = và >
- MOV (Move): Di chuyển dữ liệu từ vùng nhớ đến vùng nhớ khác
- SMOV (Shift Move): Di chuyển dữ liệu từ vùng nhớ này đến vùng nhớ khác
- CML (Compliment): Sao chép và nghịch đảo chuỗi bit nguồn sang đích
- BMOV (Block Move): Sao chép một khối nhiều phần tử dữ liệu đến đích mới
- FMOV (Fill Move): Sao chép một dữ liệu đơn đến dãy đích mới
- XCH (Exchange): Hoán đổi dữ liệu trong thiết bị xác định
- BCD (Binary Coded Decimal): Chuyển đổi số nhị phân sang BCD hay chuyển đổi dữ liệu đấu chấm động sang dạng khoa học
- BIN (Binary): Chuyển đổi các số BCD sang nhị phân tương ứng hay chuyển đổi dữ liệu dạng khoa học sang thập phân

3.2.6. Nhóm lệnh về xử lý số học: Các hàm từ FNC 20 ÷ FNC 29

- ADD (Addition): Cộng hai dữ liệu nguồn, kết quả lưu ở thiết bị đích
- SUB (Subtraction): Trừ hai dữ liệu nguồn, kết quả lưu ở thiết bị đích
- MUL (Multipcation): Nhân hai dữ liệu nguồn, kết quả lưu ở thiết bị đích
- DIV (Divation): Chia dữ liệu nguồn cho dữ liệu nguồn khác, kết quả lưu ở thiết bị đích
- INC (Increment): Thiết bị đích được tăng lên 1 mỗi khi dùng lệnh này
- DEC (Decrement): Thiết bị đích được giảm xuống 1 mỗi khi dùng lệnh này
- WAND (Word AND): Thực hiện logic AND trên hai thiết bị nguồn, kết quả lưu trong thiết bị đích
- WOR (Word OR): Thực hiện logic OR trên hai thiết bị nguồn, kết quả lưu trong thiết bị đích
- WXOR (Word Exclusive): Thực hiện logic XOR trên hai thiết bị nguồn, kết quả lưu trong thiết bị đích
- NEG (Negation): Thực hiện đổi dấu nội dung thiết bị đích

3.2.7. Nhóm lệnh về quay và dịch chuyển chuỗi bit: Các hàm từ FNC 30 ÷ FNC 39

- ROR (Rotation Right): Chuỗi bit của thiết bị đích được quay phải ‘n’ vị trí mỗi lần thi hành lệnh này
- ROL (Rotation Left): Chuỗi bit của thiết bị đích được quay trái ‘n’ vị trí mỗi lần thi hành lệnh này
- RCR (Rotation Right with Carry): Chuỗi bit của thiết bị đích được quay phải với 1 bit được trích qua cờ nhớ
- RCL (Rotation Left with Carry): Chuỗi bit của thiết bị đích được quay trái với 1 bit được trích qua cờ nhớ
- SFTR (Shift Right): Trạng thái của thiết bị nguồn được sao chép vào ngăn xếp bit và di chuyển qua phải
- SFTL (Shift Left): Trạng thái của thiết bị nguồn được sao chép vào ngăn xếp bit và di chuyển qua trái
- WSFR (Word Shift Right): Trạng thái của thiết bị nguồn được sao chép vào ngăn xếp bit và di chuyển qua phải
- WSFL (Word Shift Left): Trạng thái của thiết bị nguồn được sao chép vào ngăn xếp bit và di chuyển qua trái
- SFWR (Shift Register Right): Lệnh này tạo một ngăn xếp FIFO có độ dài n, phải dùng kèm với lệnh SFRD

- SFRD (Shift Register Left): Đọc và loại bỏ ngăn xếp FIFO. Phải dùng với lệnh SFWR

3.2.8. Nhóm lệnh về xử lý dữ liệu: Các hàm từ FNC 40 ÷ FNC 49

- ZRST (Zone Reset): Thực hiện reset dãy thiết bị
- DECO (Decode): Gía trị dữ liệu nguồn Q sẽ set bit thứ n của thiết bị bit
- ENCO (Encode): Vị trí bit hoạt động của thiết bị nguồn xác định giá trị của thiết bị đích
- SUM (The Sum Of Active Bits): Số lượng các bit bằng một trong các dãy chỉ định được lưu trong thiết bị đích
- BON (Check Specified Bit Status): Trạng thái thiết bị của bit xác định được biểu thị bằng cách kích hoạt bit cờ được chọn
- MEAN (Mean): Tính giá trị trung bình
- ANS (Annunciator Set): Lệnh này khởi động một bộ định thì. Khi vượt quá thời gian định thì sẽ kích hoạt cờ trạng thái tương ứng
- ANR (Annunciator Reset): Reset cờ trạng thái mức thấp nhất
- SQR (Square Root): Thực hiện phép toán căn số
- FLT (Float (Floating Point)): Dùng chuyển đổi dữ liệu sang dạng dấu chấm động và ngược lại

3.2.9. Nhóm lệnh về xử lý tốc độ cao: Các hàm từ FNC 50 ÷ FNC 59

- REF (Refresh): Trạng thái hiện hành của các ngõ vào/ra chỉ định đwocj đọc lại vào PC
- REFF (Refresh and Filler Adjust): Các ngõ vào X0 đến X7 được làm tươi và các bộ lọc ngõ vào của chúng được gán trị mới
- MTR (Input Matrix): Đa hợp n bằng ngõ vào trong một tập các thiết bị và chỉ có thể dùng một lần
- HSCS (High Speed Counter Set): Khi giá trị của bộ đếm bằng giá trị định trước thì set ngõ ra đã định
- HSCR (High Speed Counter Reset): Khi giá trị của bộ đếm bằng giá trị định trước thì reset ngõ ra đã định
- HSZ (High Speed Counter Zone Compare):

Hoạt động 1: Gía trị hiện hành của bộ đếm tốc độ cao được kiểm tra với khoảng giá trị xác định

Hoạt động 2: Khoảng giá trị xác định được giữ trong một bảng dữ liệu điều khiển trực tiếp các ngõ ra

Hoạt động 3: Khoảng giá trị xác định được giữ trong một bảng dữ liệu điều khiển trực tiếp tần số của PLSY bằng khoảng cách dùng D8132

- SPD (Speed Detect): Đếm số xung của encoder trong một khoảng thời gian cho phép. Kết quả có thể dùng để tính tốc độ
- PLSY (Pulse Y Output): Phát xung với tần số xác định
- PWM (Pulse Width Modulation): Tạo một chuỗi xung có độ rộng thay đổi

3.2.10. Nhóm các lệnh khác: Các hàm từ FNC 60 ÷ FNC 69

- IST (Initial Stale): Thiết lập hệ thống điều khiển
- SER (Search): Tạo một danh sách thống kê về các giá trị đwocj tìm thấy trong một stack dữ liệu
- ABSD (Absolute Drum): Kích hoạt nhiều kiểu ra tùy thuộc giá trị của bộ đếm
- INCD (Incremental Drum): Kích tuần tự từng ngõ ra tùy thuộc giá trị bộ đếm

- TTMR (Teachinh Timer): Giám át khoảng thời gian của tín hiệu và đặt dữ liệu thời gian đó vào thanh ghi dữ liệu
- STMR (Special Timer-Definable): Cung cấp bộ định thì loại off-delay, one shot và bộ định thì nhấp nháy
- ALT (Alternate State): Thiết bị đích tuần tự thay đổi trạng thái mỗi khi lệnh này hoạt động
- RAMP (Ramp-Variable Value): Tạo một giá trị trên đường dốc giữa hai giá trị dữ liệu cố định
- ROTC (Rotary Table Control): Điều khiển sự di chuyển của bàn quay
- SORT (Sort Data): Sắp thứ tự dữ liệu trong một bảng theo vùng được chọn trong khi vẫn duy trì toàn vẹn mẫu tin

3.2.11. Nhóm lệnh về nhập xuất dữ liệu: Các hàm từ FNC 70 ÷ FNC 79

- TKY (Ten Key Input): Đọc 10 phím thập phân và kết hợp các giá trị thập phân đọc được thành một số đơn
- HKY (Hexadecimal Input): Đọc 16 phím và kết hợp các giá trị thập phân đọc được thành một số đơn
- DSW (Digital Switch-Thumbwheel Input): Cho phép đọc n bộ chọn nhấn nhập số BCD
- SEGD (Seven Segmant Decoder): Dữ liệu thập lục phân được giải mã thành dạng dữ liệu điều khiển đèn 7 đoạn
- SEGL (Seven Segment with Latch): Dùng để ghi dữ liệu ra bộ hiển thị 4 chữ số, tối đa hai bộ
- ARWS (Arrow Switch): Tạo bảng (panel) nhập dữ liệu số
- ASC (ASCII Code): Một chuỗi chữ số chuyển thành mã ASCII
- PR (Print to a Display): Xuất dữ liệu ASCII cho bộ hiển thị
- FROM (Read From a Special-Function Block): Dữ liệu được đọc từ bộ nhớ đệm của các khối chức năng chuyên dùng gắn vào
- TO (To): Dữ liệu được ghi vào các bộ nhớ đệm của các khối chức năng chuyên dùng gắn vào

3.3. THIẾT KẾ CHƯƠNG TRÌNH

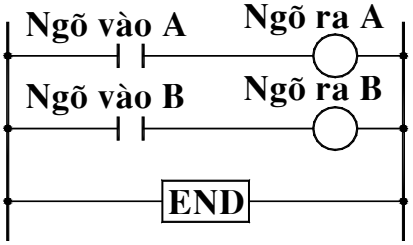
3.3.1. Các bước lập trình: Ngôn ngữ lập trình, sự tiếp cận vấn đề có hệ thống có thể cải thiện khả năng tạo ra các chương trình chất lượng cao trong thời gian ngắn. Kỹ thuật thiết kế có hệ thống gồm các bước sau:

- Xác định yêu càu đối với ngõ vào và ngõ ra
- Xác định thuật toán sẽ được sử dụng. Thuật toán là thứ tự các bước xác định phương pháp giải quyết vấn đề. Điều này thường được thực hiện bằng lưu đồ hoặc viết bằng thuật giải mã (pseudocode), kể cả sử dụng các từ BEGIN, DO, END, IF-THEN-ELSE, WHILE-DO,...
- Thuật toán đuệoc diễn dịch thành các lệnh để có thể nhập vào PLC.
- Kiểm tra và gỡ rối chương trình
- Chương trình được lập thành tài liệu để mọi người sử dụng hoặc sửa đổi sau này đều hiểu sự hoạt động của chương trình đó.

3.3.2. Các lưu đồ thuật giải mã:

a. Chuỗi thứ tự hoạt động:

Khi có tín hiệu vào khởi động, ngõ ra A hoạt động. Khi A hoàn tất, A vận hành ngõ

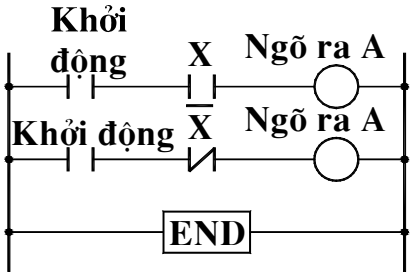


Hình 1.19 Chuỗi

vào B và ngõ ra B xuất hiện.

b. Điều kiện:

Khi xuất hiện tín hiệu vào khởi động, Ngõ ra là A khi có tín hiệu vào đến X, ngược Lại ngõ ra là B.



Hình 1.20 Điều kiện

c. Vòng lặp:

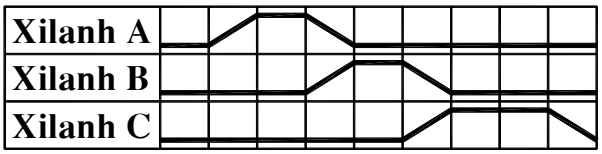
Bộ đếm có thể được sử dụng ở vị trí vòng lặp được lặp lại với số lần xác định, nhận xung tín hiệu vào mỗi lần vòng lặp xảy ra và dừng chuỗi vòng lặp khi hoàn tất số vòng lặp được yêu cầu

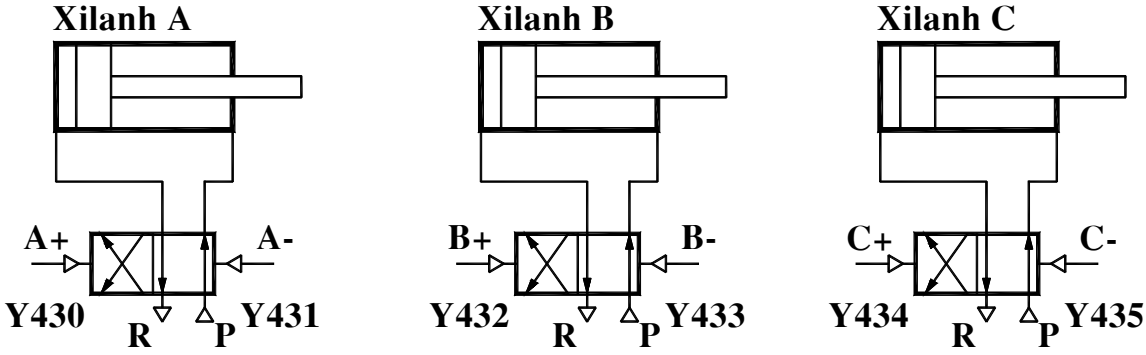
3.3.3. Một số hoạt động của chương trình mẫu:

a. Ví dụ 1:

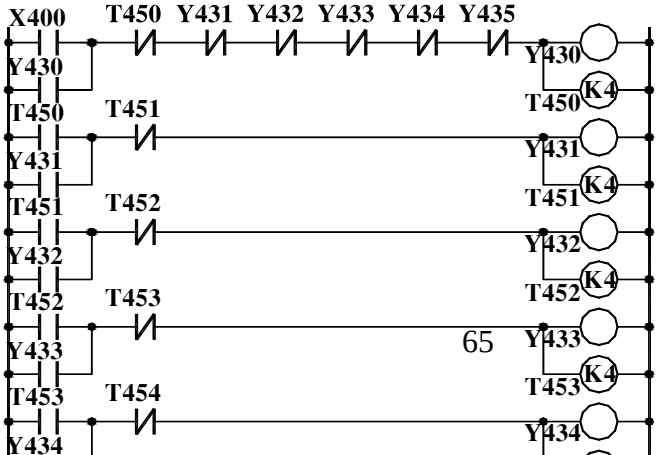
Xét tác vụ gồm ba xilanh A, B và C lần lượt hoạt động theo thứ tự A tiến về bên phải, A tiến về bên trái, B tiến về bên phải, B tiến về bên trái, C tiến về bên phải, C tiến về bên trái (chuỗi này thường được viết là A+, A-, B+, B-, C+, C-)

Chu trình:





Hình 1.21 Các van điều khiển



Piston A di chuyển qua phải

Piston A di chuyển qua trái

Piston B di chuyển qua phải

Piston B di chuyển qua trái

Piston C di chuyển qua phải

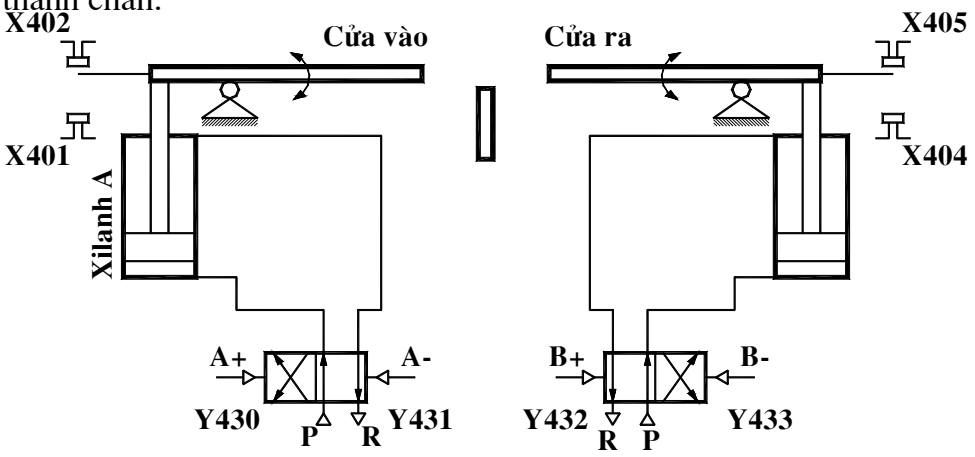
Thứ tự lệnh của chương trình nêu trên của Mitsubushi:

LD	X400	Khởi động công tắc
OR	Y430	
ANI	T450	
ANI	Y431	
ANI	Y432	
ANI	Y433	
ANI	Y434	
ANI	Y435	
OUT	Y430	Piston A di chuyển sang phải
OUT	T450	Đồng hồ định giờ T450 khởi động
LD	T450	
OR	Y431	
ANI	T451	
OUT	Y431	Piston A chuyển sang trái
OUT	T451	Đồng hồ định giờ T451 khởi động
LD	T451	
OR	Y432	
ANI	T452	
OUT	Y432	Piston B chuyển sang phải
OUT	T452	Đồng hồ định giờ T452 khởi động
LD	T452	
OR	Y433	
ANI	T453	
OUT	Y433	Piston B chuyển sang trái
OUT	T453	Đồng hồ định giờ T453 khởi động
LD	T453	
OR	Y434	
ANI	T454	
OUT	Y434	Piston C chuyển sang phải
OUT	T454	Đồng hồ định giờ T454 khởi động
LD	T454	
OR	Y435	
ANI	T455	
OUT	Y435	Piston C chuyển sang trái
OUT	T455	Đồng hồ định giờ T455 khởi động
END		

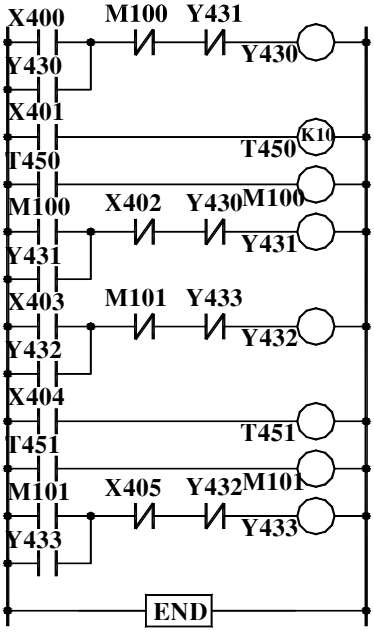
X400 là công tắc khởi động, khi đóng sẽ có tín hiệu ra từ Y430 và đồng hồ định giờ T450 khởi động. Công tắc khởi động bị xác lập, K=4, các tiếp điểm thường đóng của đồng hồ định giờ T450 mở, ngắt mạch Y430, cấp nguồn cho Y431 và khởi động đồng hồ định giờ T451. Piston A di chuyển về bên trái. Trên nấc 2, các tiếp điểm T450 bị khóa, do đó, tín hiệu ra của các linh kiện này vẫn tiếp tục cho đến hết thời gian xác lập. Khi đó, các tiếp điểm thường đóng của đồng hồ định giờ T451 mở và các tiếp điểm thường mở của đồng hồ định giờ T451 đóng. Điều này làm ngắt mạch Y431 và cấp nguồn cho Y432, khởi động đồng hồ định giờ T452. Piston B di chuyển sang phải. Mỗi nấc tiếp theo sẽ khích hoạt Solenoid kế tiếp. Do đó, tuần tự từng ngõ ra được cấp năng lượng.

b. Ví dụ 2:

Xét hoạt động của các van khí nén vận hành các thanh chắn ở bãi đậu xe. Thanh chắn vào sẽ mở khi nạp đúng số tiền vào hộp thu, thanh chắn ra sẽ mở khi phát hiện xe đến gần thanh chắn.



Hình 1.23 Hệ thống Van - Piston



Nâng thanh chắn vào
X400 là công tắc được vận hành bằng đồng xu
Y430 là ngõ ra đến Solenoid 1
Đồng hồ định giờ T450 cung cấp thời gian thanh chắn vào 10s
M100 là rơle nội
X401 là ngõ vào báo hiệu thanh chắn nâng lên
Hạ thanh chắn vào
Y431 là ngõ ra đến Solenoid 2
X402 là ngõ vào báo hiệu thanh chắn hạ xuống
Nâng thanh chắn ra
Y432 là ngõ ra đến Solenoid 3
X403 là ngõ vào khi xe đến gần thanh chắn ra
Thời gian duy trì thanh chắn ra, 10s
M101 là rơle nội
X404 báo hiệu thanh chắn nâng lên
Hạ thanh chắn ra
Y433 là ngõ ra đến Solenoid 4
X405 báo hiệu thanh chắn ra hạ xuống

Hình 1.24 Chương trình thanh chắn xe

Ngõ vào:

X400	Công tắc vận hành bằng đồng xu
X401	Ngõ vào khi thanh chắn vào nâng lên
X402	Ngõ vào khi thanh chắn vào hạ xuống

X403 Ngõ vào khi xe đến gần thanh chắn ra
X404 Ngõ vào khi thanh chắn ra nâng lên
X405 Ngõ vào khi thanh chắn ra hạ xuống

Ngổ ra:

Y430 Van A, solenoid 1
Y431 Van B, solenoid 2
Y432 Van C, solenoid 3
Y433 Van D, solenoid 4

Thứ tự lệnh của chương trình nêu trên của Mitsubishi:

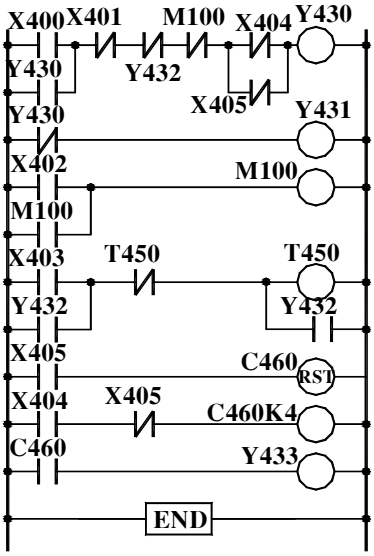
```
LD    X400
OR    Y430
ANI   M100
ANI   Y431
OUT   Y430
LD    X401
OUT   T450
K     10
LD    T450
OUT   M100
LD    M100
OR    Y431
ANI   X402
ANI   Y430
OUT   Y431
LD    X403
OR    Y432
ANI   M101
ANI   Y433
OUT   Y432
LD    X404
OUT   T451
K     10
LD    T451
OUT   M101
LD    M101
OR    Y433
ANI   X405
ANI   Y433
OUT   Y433
END
```

Ngõ ra Y430 đến solenoid 1 nâng thanh chắn vào khi tín hiệu ra từ bộ cảm biến ở hộp đựng tiền cung cấp ngõ vào X400. Y430 bị khóa và duy trì trạng thái hoạt động cho đến khi rơle nội M100 mở. Tuy nhiên, ngõ ra này sẽ không xuất hiện nếu thanh chắn đang hạ xuống và có ngõ ra Y431 đến solenoid 2. Đồng hồ định giờ T450 dùng để giữ thanh chắn trên cao trong 10s, khởi động bằng ngõ vào X402 từ bộ cảm biến cho biết thanh chắn trên cao. Khi hết thời gian này, ngõ ra Y431 được mở, kích hoạt solenoid 2 và hạ thanh chắn xuống. Thanh chắn ra được nâng lên do ngõ ra Y432 đến solenoid 3 khi bộ cảm biến phát hiện xe và cấp ngõ vào X401. Khi thanh chắn lên cao, đồng hồ

định giờ T451 được sử dụng để giữ thanh chắn trên cao trong 10s, khởi động nhờ ngõ vào X404 từ bộ cảm biến cho biết thanh chắn trên cao. Khi hết thời gian này, ngõ ra Y433 được mở, kích hoạt solenoid 4 và thanh chắn hạ xuống.

c. Ví dụ 3:

Xét bài toán dây chuyền sản xuất gồm băng tải chuyển các chai đến thiết bị đóng gói, các chai đực tải trên băng chuyền, được kiểm tra để bảo đảm đầy, đã đóng nắp và số lượng chai (4) đang được đóng gói vào thùng là đúng. Các hoạt động điều khiển được yêu cầu là: nếu chai không đầy sẽ dừng băng chuyền; kích hoạt máy đóng nắp khi chai vào đúng vị trí, trong thời gian này băng chuyền dừng; đếm bốn chai và kích hoạt máy đóng gói, băng chuyền dừng nếu có chai khắc đến điểm đóng gói trong thời gian này; phát âm thanh cảnh báo khi dừng băng chuyền.



Y430 là ngõ ra đến băng chuyền. X400 là nút khởi động, X401 là nút dừng. Băng chuyền dừng khi Y432, M100, X404 hoặc X405 đực kích hoạt

Y431 là ngõ ra đến thiết bị cảnh báo, mở khi băng chuyền dừng

M100 là rơle nội được kích hoạt khi X402 đóng do chai không đầy. Sau đó M100 dừng băng chuyền

T450 là đồng hồ định giờ, dừng băng chuyền trong thời gian đã chọn để đóng nắp chai, Y432 cấp năng lượng cho máy đóng nắp và dừng băng chuyền

Cài đặt lại bộ đếm khi máy đóng gói có đủ 4 chai

Ngõ vào X404 khi chai được phát hiện. X405 mở khi đang đóng gói. 4 chai đã đếm

Y433 cấp năng lượng cho máy đóng gói khi C460 đếm được 4 chai

Hình 1.25 Chương trình điều khiển Mitsubishi

Thứ tự lệnh của chương trình nêu trên của Mitsubishi:

LD	X400	Nấc thứ 1
OR	Y430	
ANI	X401	
ANI	Y432	
ANI	M100	
LDI	X404	
ORI	X405	
ANB		
OUT	Y430	
LDI	Y430	Nấc thứ 2
OUT	Y431	
LD	X402	Nấc thứ 3
OR	M100	
OUT	M100	
LD	X403	Nấc thứ 4
OR	Y431	
ANI	T450	
OUT	T450	
K	2	2 giây để đóng nắp

```
OUT     T450
LD      X405    Nấc thứ 5
RST     C460
LD      X404    Nấc thứ 6
ANI     X405
OUT     C460
K       4       4 chai được đếm
LD      C460    Nấc thứ 7
OUT     Y433
END             Kết thúc chu kỳ
```

Việc phát hiện chai đầy hay không có thể được thực hiện bằng bộ cảm biến quang điện, sau đó bộ cảm biến này có thể được dùng để kích hoạt công tắc ngõ vào X402. Sự hiện diện của chai ở máy đóng nắp cũng có thể được nhận biết bằng bộ cảm biến quang điện ngõ vào X403. Tín hiệu vào bộ đếm chai cũng có thể xuất phát từ bộ cảm biến quang điện ngõ vào X404. Các ngõ vào khác có thể là các công tắc khởi động ngõ vào X400 và dừng ngõ vào X401 đối với băng chuyền và tín hiệu ngõ vào X405 từ máy đóng gói khi máy đang vận hành, đã nhận đủ 4 chai và chưa nhận thêm chai khác.