第五章 数字系统设计
Digital System Design

5.2 基本数字系统设计举例

下面通过例子介绍信息 处理单元和控制单元的设计方法。

例1. 试设计一个累加器，该累加器能执行如下表所列出的一组微操作。

可以为.不超乡网

控制变量		微操作	名称
P_1	这些控制变量是互斥的	$A \leftarrow A+B$	加
P_2		$A \leftarrow 0$	清 0
P_3		$A \leftarrow \overline{A}$	取反
P_4		$A \leftarrow A \wedge B$	与
P_5		$A \leftarrow A \vee B$	或
P_6		$A \leftarrow A \oplus B$	异或
P_7		$A \leftarrow ShrA$	右移
P_8		$A \leftarrow ShlA$	左移
P_9		$A \leftarrow A+1$	加 1
		If(A=0)then(Z=1)	检测 0

注： 累加器是信息处理单元中一个特殊的寄存器，它能执行多种微操作功能：

加法微操作、

逐次累加并暂存累加和、

并行接收、

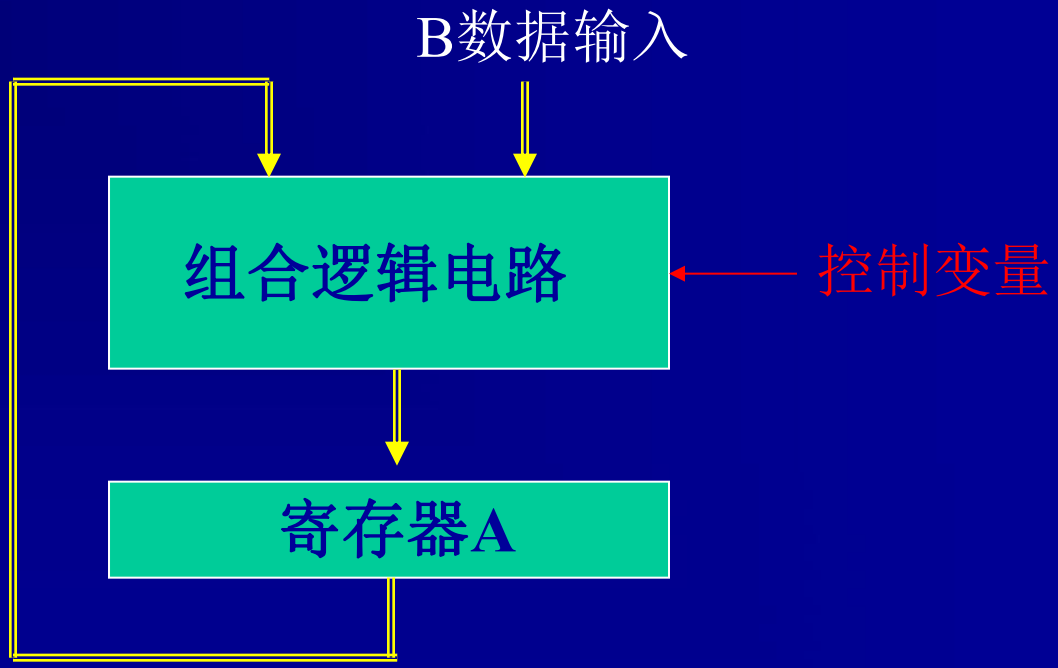
并行输出、

暂存、

移位

累加器的结构

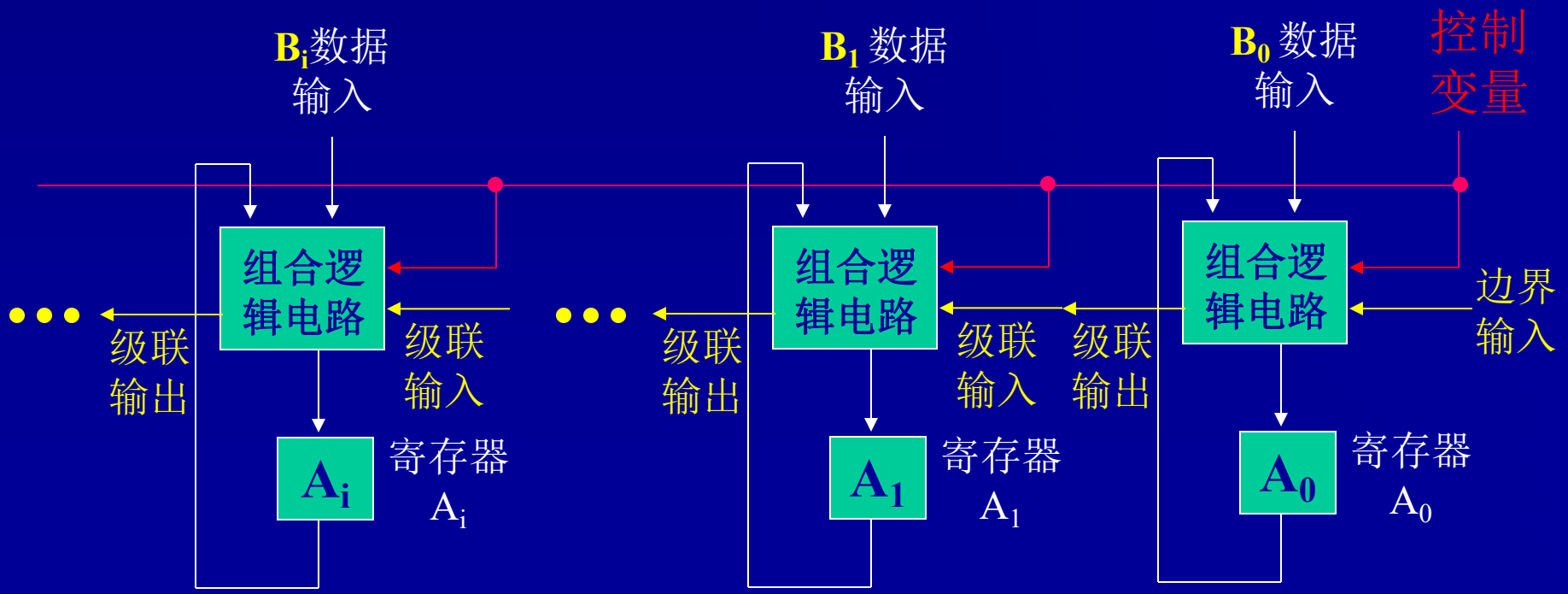
累加器由**寄存器A**和**组合逻辑电路**组成，如下框图：



寄存器A既可作为**加数**寄存器，又可作为**和数**寄存器。决定累加器微操作的各个控制变量是互斥的，在任何给定时间内只有一个控制变量被选通，产生响应的微操作。

累加器的结构

为简化累加器的设计，假设：累加器由**n个相同的单元**组成，每个单元包含了执行各种微操作所需的逻辑电路，只要完成一个单元的各部分电路设计，就可以将它们综合成累加器的一个**典型单元**，然后用若干个典型单元组成一个完整的累加器（迭代设计）。

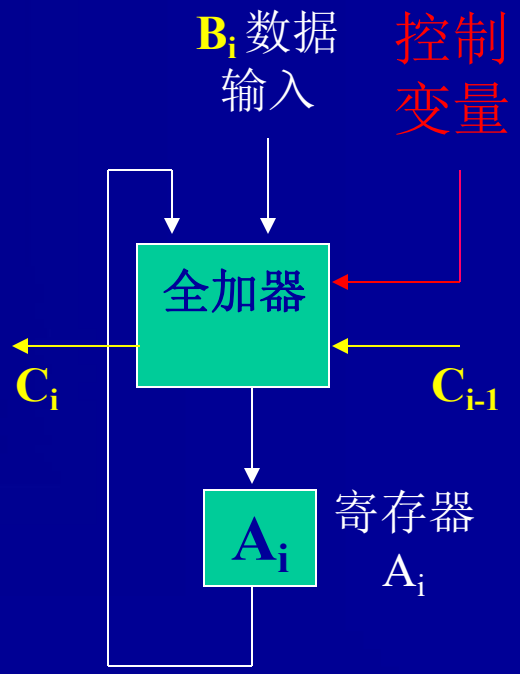


累加器的一个典型单元的设计

1. “加”微操作

这部分电路的工作用状态真值表描述如下：

现态 A_i	输入 B_i	输入 C_{i-1}	次态 A_i^{n+1}	输出 C_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



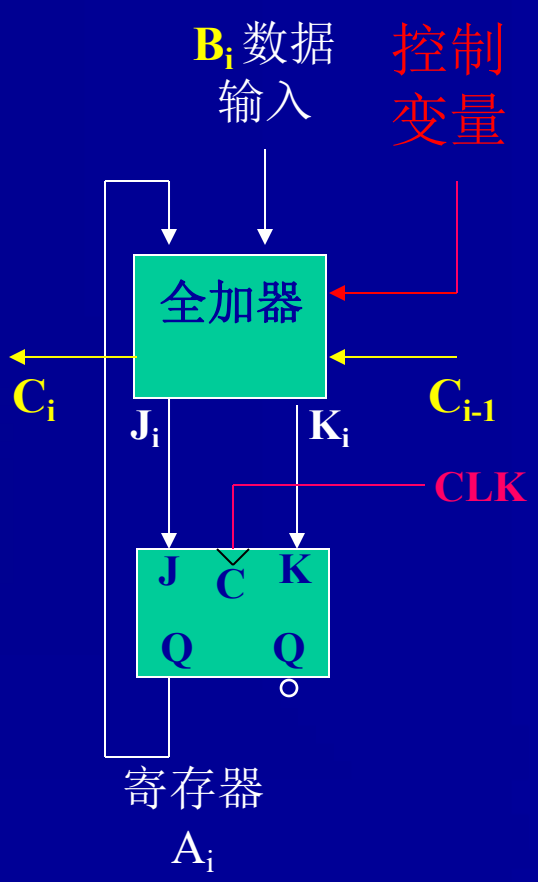
假如**寄存器A**采用**JK 触发器**，则列出的激励表如下：

累加器的一个典型单元的设计

1.“加”微操作

JK 触发器的“加”微操作的**激励表**如下：

现态 A_i	输入 B_i	输入 C_{i-1}	次态 A_i^{n+1}	激励变量 J_i	激励变量 K_i	输出 C_i
0	0	0	0	0	d	0
0	0	1	1	1	d	0
0	1	0	1	1	d	0
0	1	1	0	0	d	1
1	0	0	1	d	0	0
1	0	1	0	d	1	1
1	1	0	0	d	1	1
1	1	1	1	d	0	1



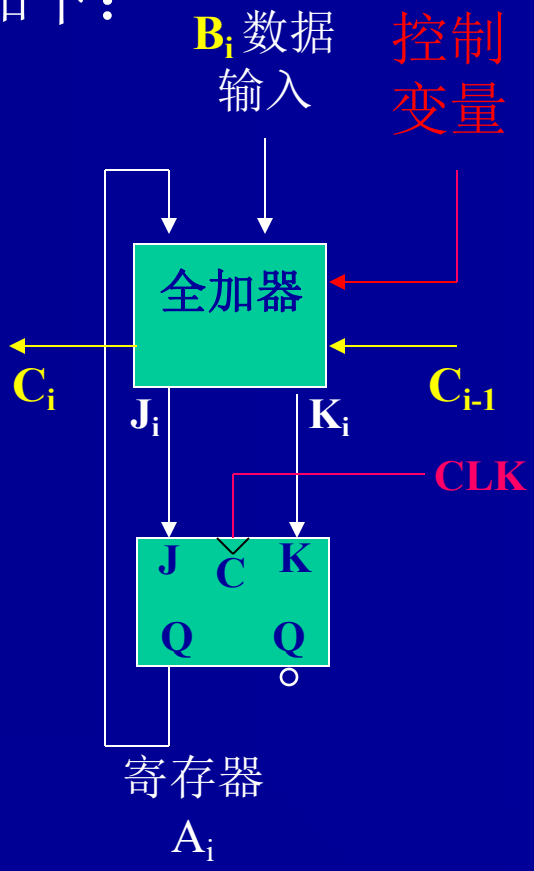
用**卡诺图**化简，则如下：

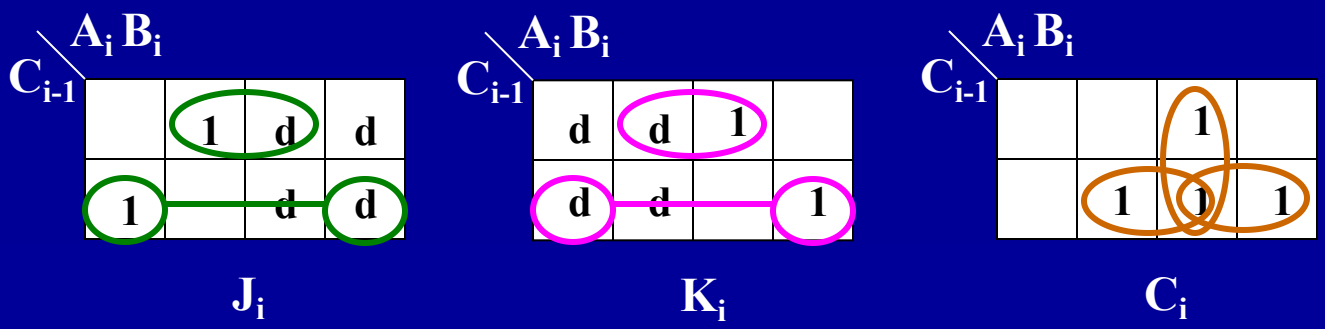
累加器的一个典型单元的设计

1.“加”微操作

JK 触发器的“加”微操作的激励表如下：

现态 A_i	输入 B_i	输入 C_{i-1}	次态 A_i^{n+1}	激励变量 J_i	激励变量 K_i	输出 C_i
0	0	0	0	0	d	0
0	0	1	1	1	d	0
0	1	0	1	1	d	0
0	1	1	0	0	d	1
1	0	0	1	d	0	0
1	0	1	0	d	1	1
1	1	0	0	d	1	1
1	1	1	1	d	0	1





累加器的一个典型单元的设计

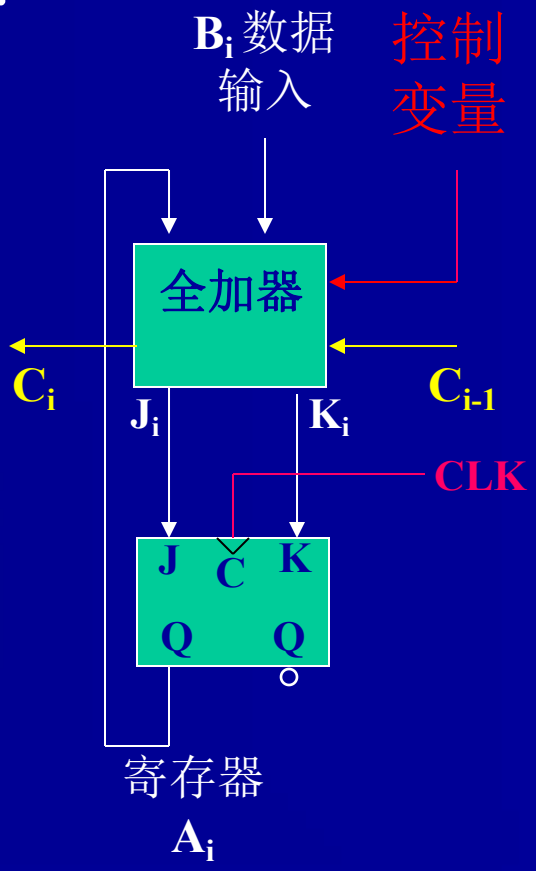
1. “加”微操作

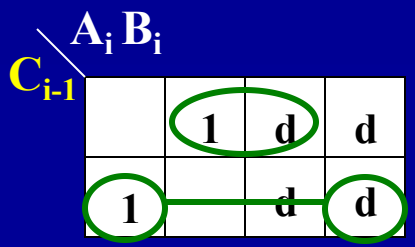
激励函数 J_i、 K_i 和输出函数 C_i 的表达式：

$$J_i = \overline{B_i}\, C_{i-1} + B_i\, \overline{C_{i-1}}$$

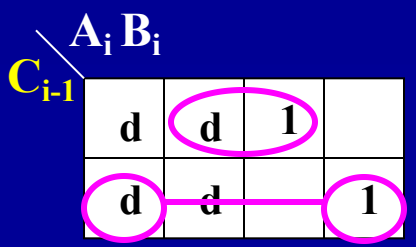
$$K_i = \overline{B_i}\, C_{i-1} + B_i\, \overline{C_{i-1}} = J_i$$

$$C_i = A_i B_i + A_i C_{i-1} + B_i C_{i-1}$$

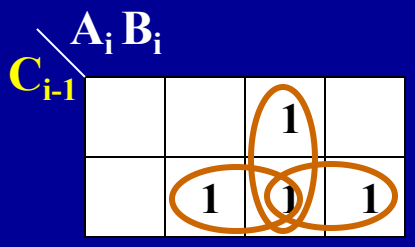




J_i



K_i



C_i

累加器的一个典型单元的设计

1. “加”微操作

激励函数 J_i、 K_i 和输出函数 C_i 的表达式：

$$J_i = \overline{B_i} C_{i-1} + B_i \overline{C_{i-1}}$$

$$K_i = \overline{B_i} C_{i-1} + B_i \overline{C_{i-1}} = J_i$$

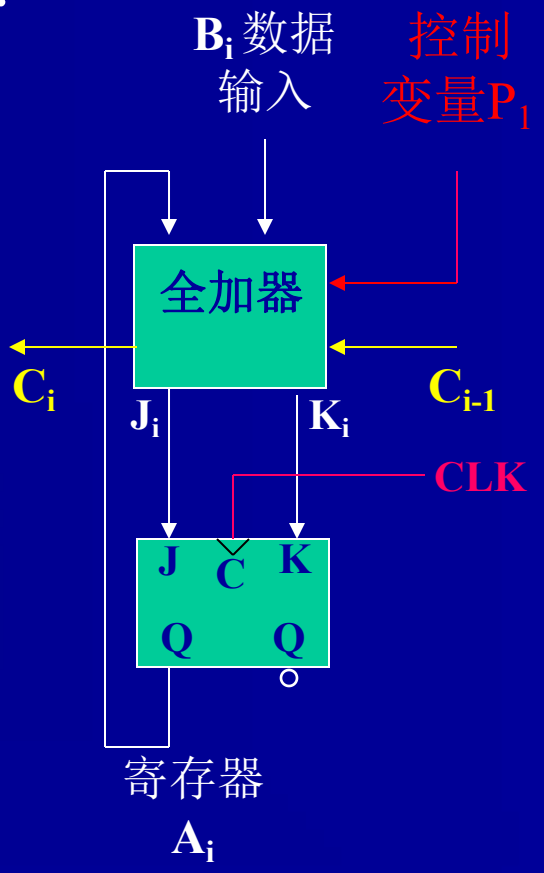
$$C_i = A_i B_i + A_i C_{i-1} + B_i C_{i-1}$$

由于“加”微操作是由控制变量P_1启动的，即仅当$P_1=1$时激励函数才能影响触发器的状态：

$$J_i = \overline{B_i} C_{i-1} P_1 + B_i \overline{C_{i-1}} P_1$$

$$K_i = \overline{B_i} C_{i-1} P_1 + B_i \overline{C_{i-1}} P_1 = J_i$$

$$C_i = A_i B_i + A_i C_{i-1} + B_i C_{i-1}$$



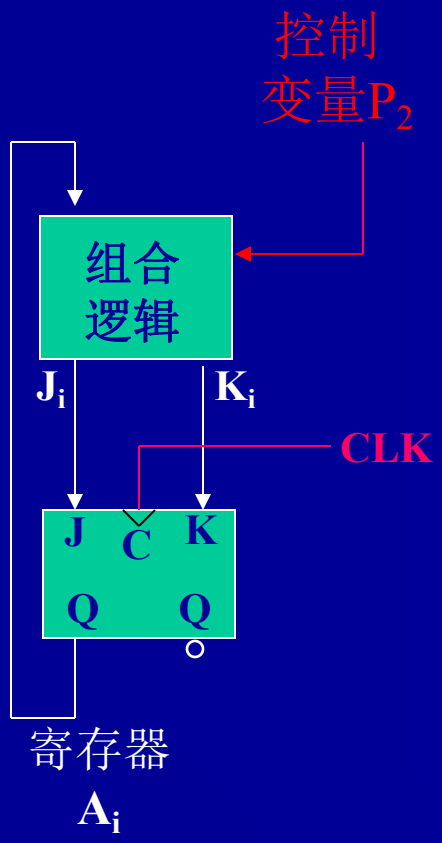
累加器的一个典型单元的设计

2. “清 0”微操作

控制变量P_2使寄存器A中所有触发器全部清 0，即仅当$P_2=1$时激励函数能使JK触发器复位：

$$J_i = 0$$

$$K_i = P_2$$



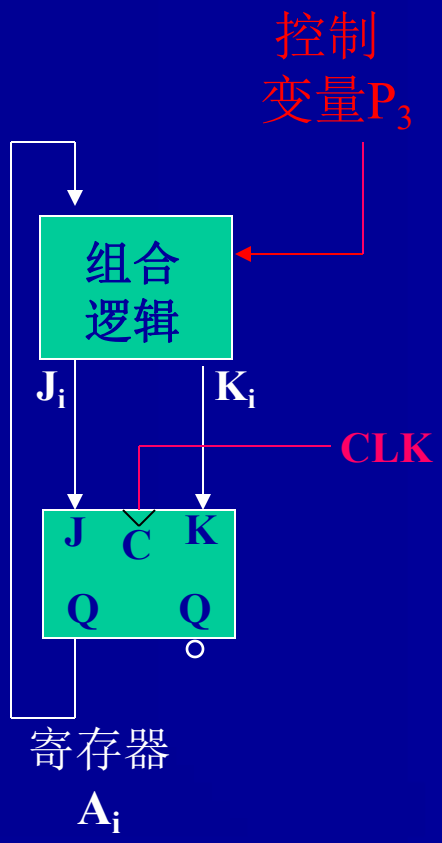
累加器的一个典型单元的设计

3. “取反”微操作

控制变量P_3使寄存器A中所有信息取反，即仅当$P_3=1$时激励函数能使JK触发器变反（计数）：

$$J_i = P_3$$

$$K_i = P_3$$



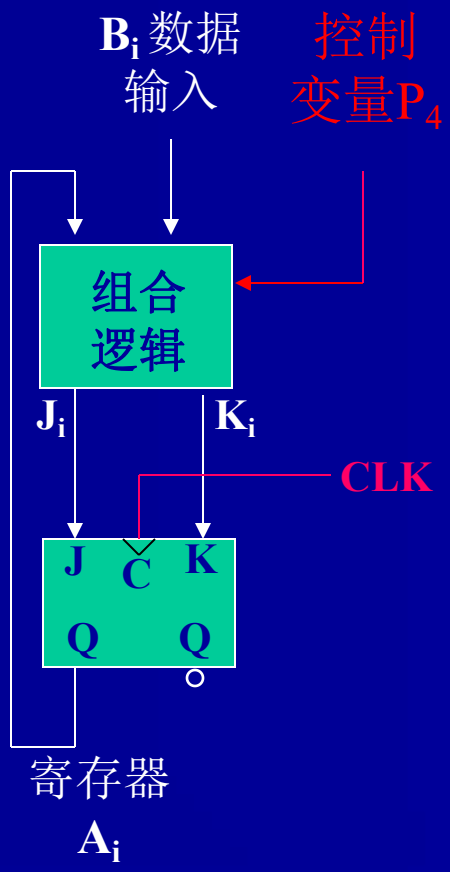
累加器的一个典型单元的设计

4.“与”微操作

控制变量P_4使寄存器A_i与B_i实现**逻辑“与”运算**，并将结果存入触发器A_i，这部分电路的工作用状态真值表描述如下：

现态 A_i	输入 B_i	次态 A_i^{n+1}
0	0	0
0	1	0
1	0	0
1	1	1

相与用JK

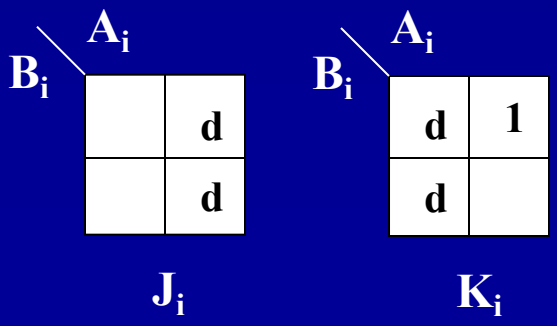


寄存器A采用**JK 触发器**，则列出的激励表如下：

累加器的一个典型单元的设计

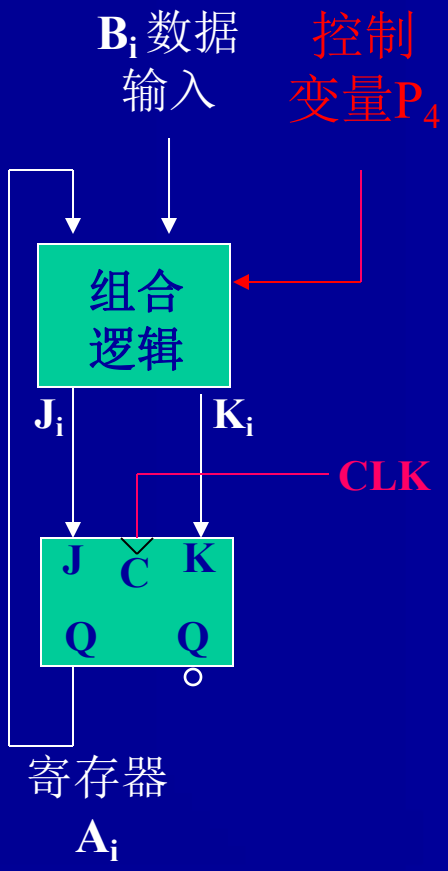
4. “与”微操作

现态 A_i	输入 B_i	次态 A_i^{n+1}	激励函数 J_i	K_i
0	0	0	0	d
0	1	0	0	d
1	0	0	d	1
1	1	1	d	0



$$J_i = 0$$

$$K_i = \overline{B_i} P_4$$

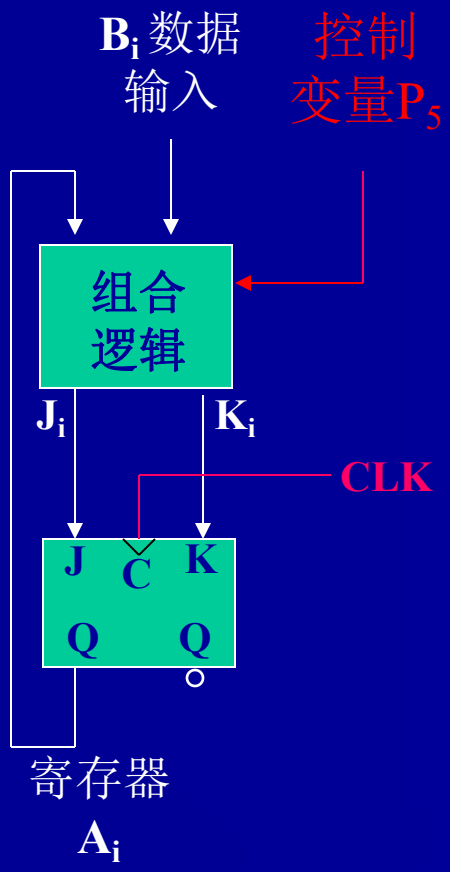


累加器的一个典型单元的设计

5.“或”微操作

控制变量P_5使寄存器A_i与B_i实现**逻辑“或”运算**，并将结果存入触发器A_i，这部分电路的工作用状态真值表描述如下：

现态 A_i	输入 B_i	次态 A_i^{n+1}
0	0	0
0	1	1
1	0	1
1	1	1

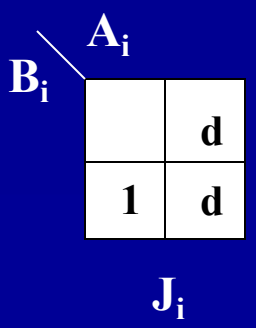


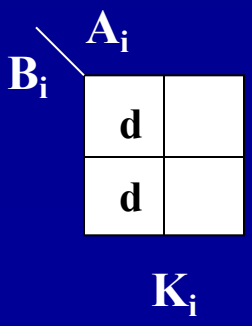
寄存器A采用**JK 触发器**，则列出的激励表如下：

累加器的一个典型单元的设计

5. “或”微操作

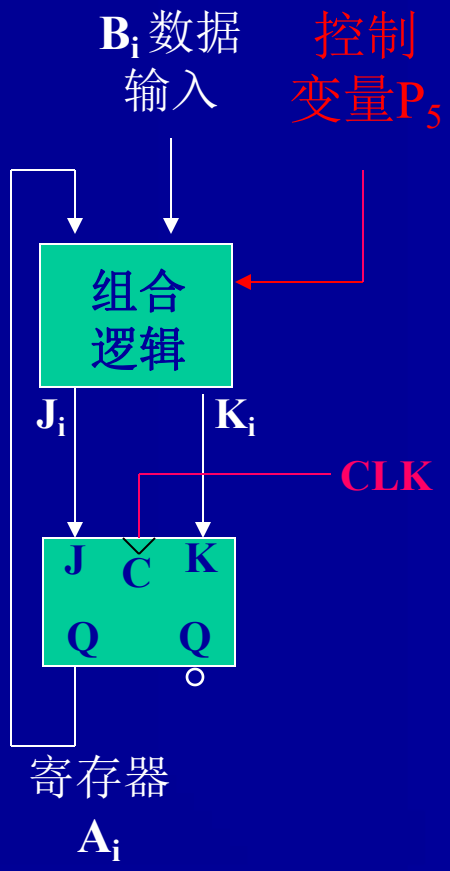
现态 A_i	输入 B_i	次态 A_i^{n+1}	激励函数 J_i	K_i
0	0	0	0	d
0	1	1	1	d
1	0	1	d	0
1	1	1	d	0





$J_i = B_i P_5$

$K_i = 0$

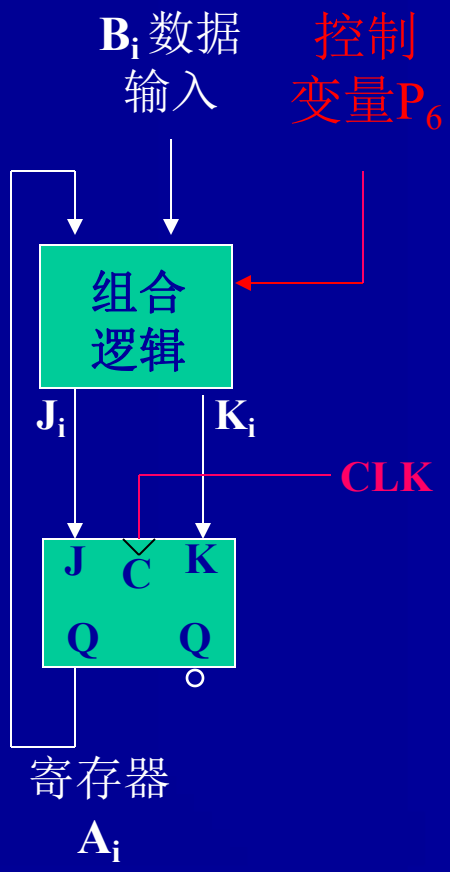


累加器的一个典型单元的设计

6.“异或”微操作

控制变量P_6使寄存器A_i与B_i实现**逻辑“异或”运算**，并将结果存入触发器A_i，这部分电路的工作用状态真值表描述如下：

现态 A_i	输入 B_i	次态 A_i^{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

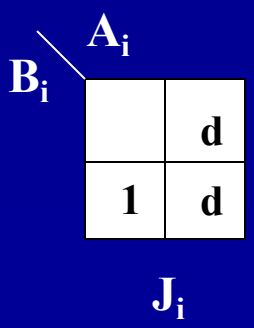


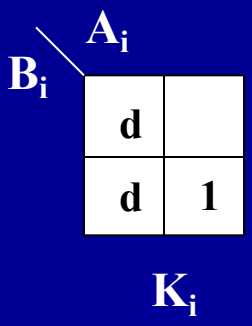
寄存器A采用**JK 触发器**，则列出的激励表如下：

累加器的一个典型单元的设计

6. “异或”微操作

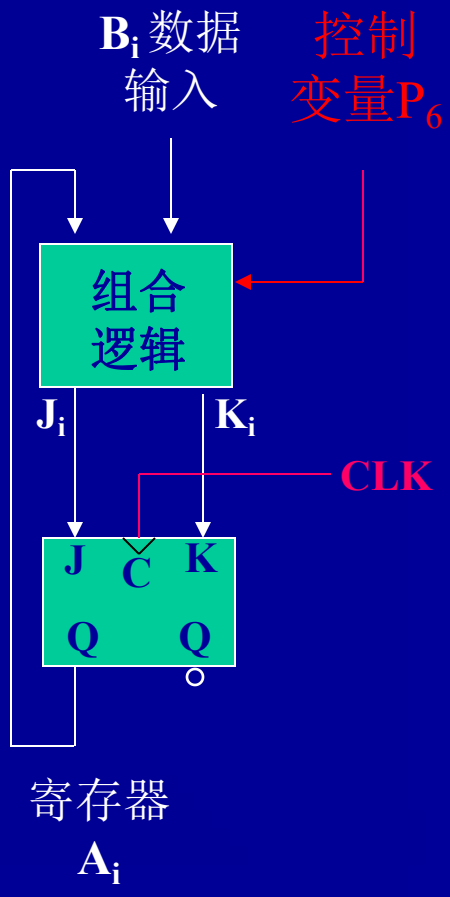
现态 A_i	输入 B_i	次态 A_i^{n+1}	激励函数 J_i	K_i
0	0	0	0	d
0	1	1	1	d
1	0	1	d	0
1	1	1	d	1





$$J_i = B_i P_6$$

$$K_i = B_i P_6$$



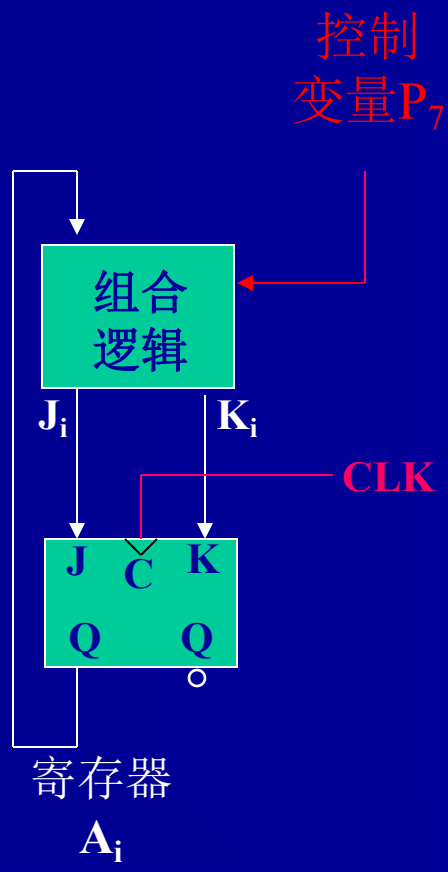
累加器的一个典型单元的设计

7. “右移”微操作

控制变量P_7使寄存器A内的信息右移一位，即将触发器A_{i+1}存入触发器A_i，则对JK 触发器的激励函数如下：

$$J_i = A_{i+1} P_7$$

$$K_i = \overline{A_{i+1}} P_7$$



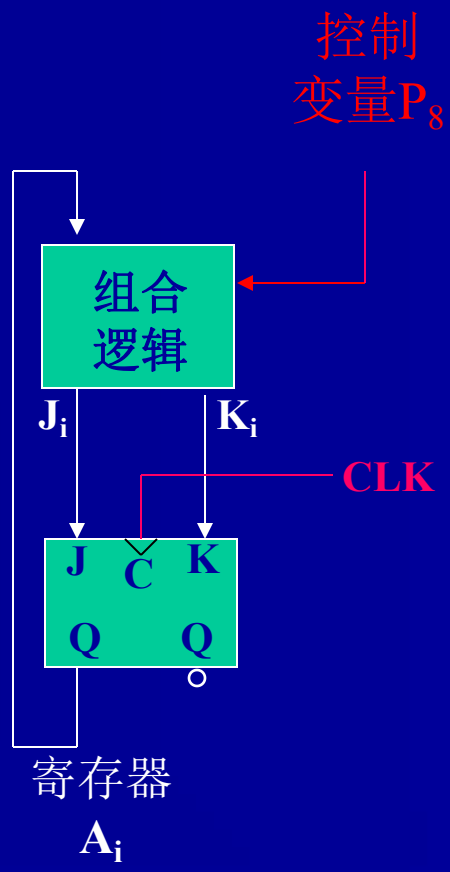
累加器的一个典型单元的设计

8.“左移”微操作

控制变量P_8使寄存器A内的信息左移一位，即将触发器A_{i-1}存入触发器A_i，则对JK 触发器的激励函数如下：

$$J_i = A_{i-1} P_8$$

$$K_i = \overline{A_{i-1}} P_8$$



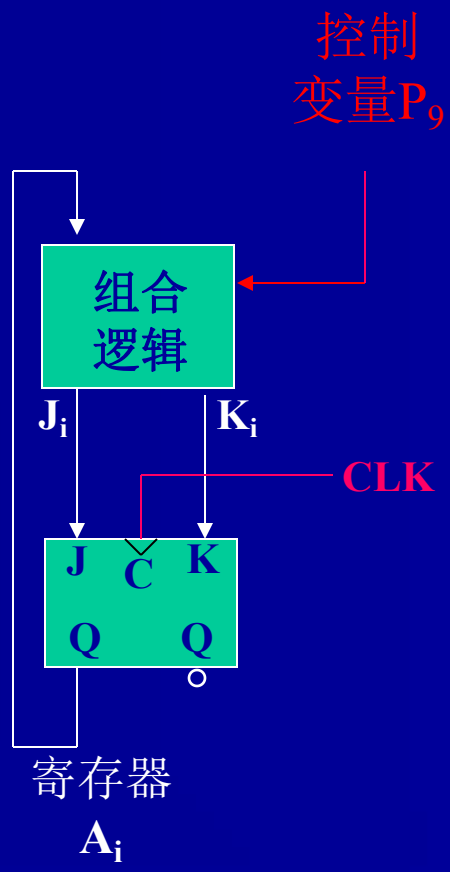
累加器的一个典型单元的设计

9. “加 1”微操作

控制变量P_9使寄存器A内的数据**加 1**，即将寄存器A的所有触发器构成**同步加 1 计数器**，则对**JK 触发器**的激励函数如下：

$J_i = K_i = Q_{i-1} \cdot Q_{i-2} \cdot \cdots \cdot Q_1 \ (i \neq 1)$ —— 进位传递函数

$J_1 = K_1 = 1$ —— 最低位呈计数态



累加器的一个典型单元的设计

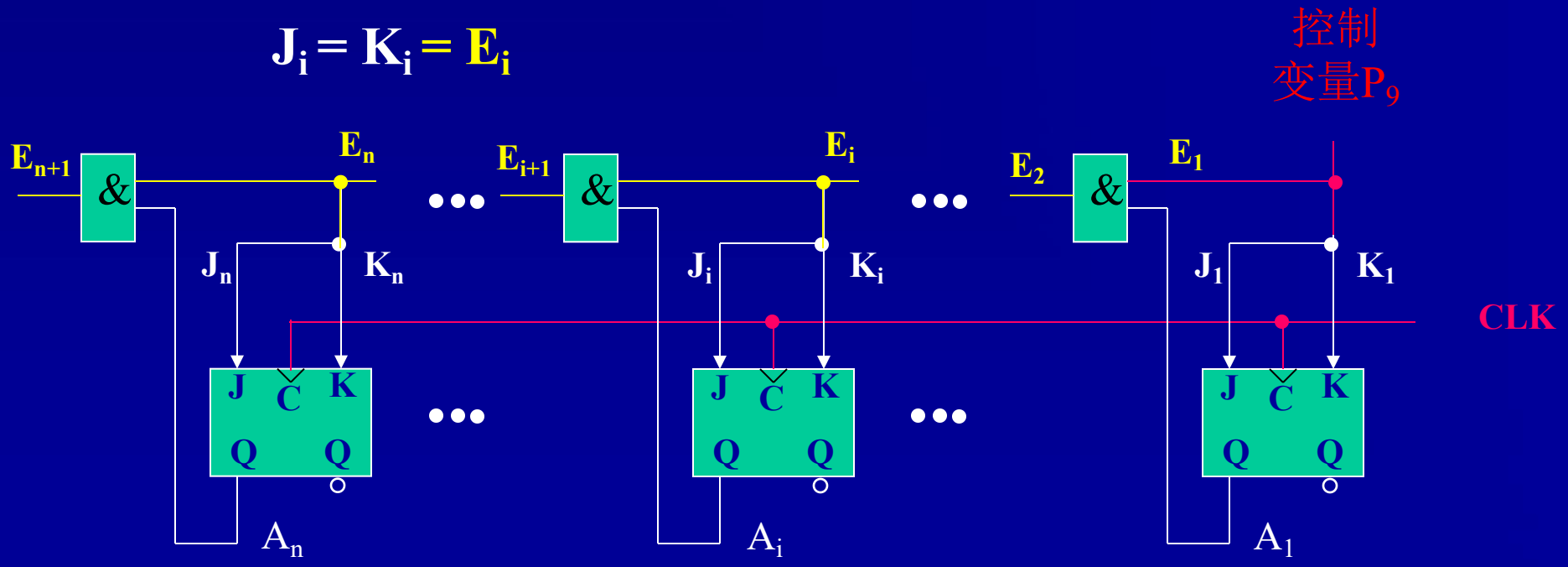
9. “加 1”微操作

当$P_9=1$时，启动“加 1”微操作：

$J_1 = K_1 = P_9 = E_1$

$E_{i+1} = A_i E_i$

$J_i = K_i = E_i$



累加器的一个典型单元的设计

10. “检测 0 ”微操作

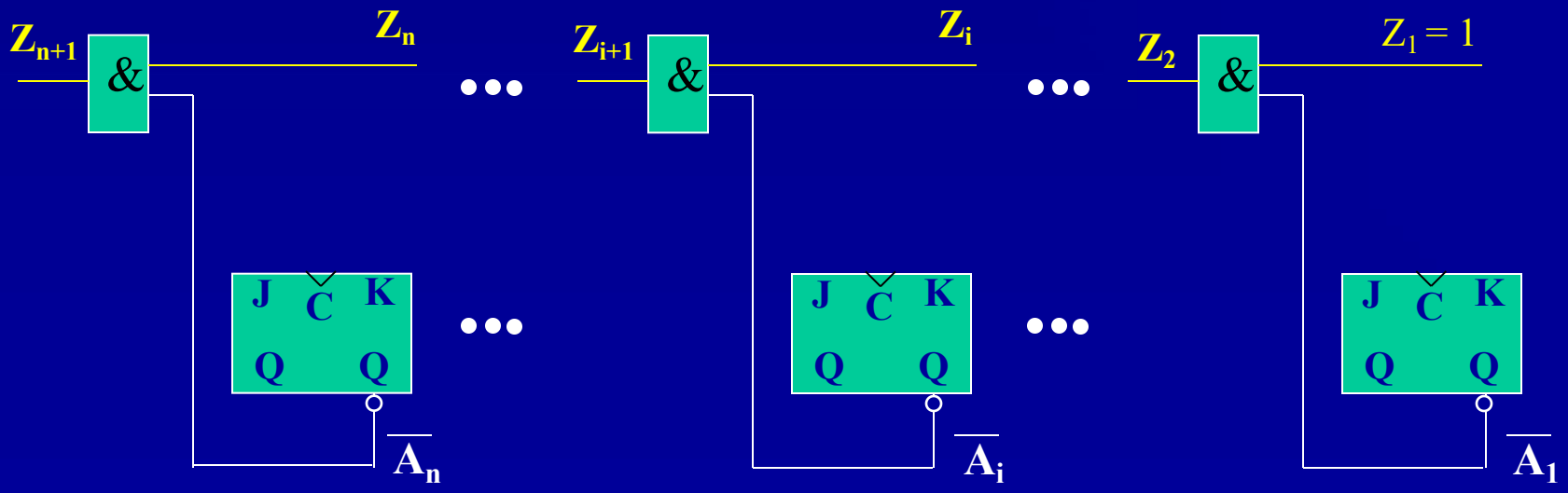
当寄存器**A**的所有触发器均为0时，输出变量**Z为1**。

该微操作**与控制变量（时钟序列）无关。**

$$Z_1 = 1$$

$$Z_{i+1} = \overline{A_i}\, Z_i$$

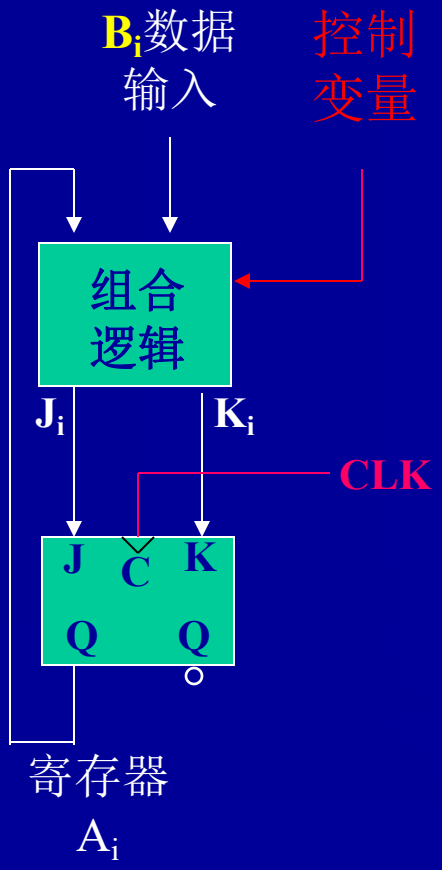
$$Z = Z_{n+1}$$



累加器的一个典型单元的设计

综合上述10种微操作，由于它们的控制变量是**互斥的**，因此在**任何时刻只有一个控制变量在起作用**，故可以将所有微操作的函数表达式**“或”起来**，构成一个典型单元的逻辑表达式。

控制变量	微操作	名称	逻辑函数表达式
P_1	A←A+B	加	$J_i = K_i = B_i\overline{C}_{i-1}P_1 + \overline{B}_i C_{i-1}P_1$, $C_i = A_iB_i + A_iC_{i-1} + B_iC_{i-1}$
P_2	A←0	清 0	$J_i = 0$, $K_i = P_2$
P_3	A←$\overline{A}$	取反	$J_i = P_3$, $K_i = P_3$
P_4	A←A∧B	与	$J_i = 0$, $K_i = \overline{B}_i P_4$
P_5	A←A∨B	或	$J_i = B_i P_5$, $K_i = 0$
P_6	A←A⊕B	异或	$J_i = B_i P_6$, $K_i = B_i P_6$
P_7	A←ShrA	右移	$J_i = A_{i+1} P_7$, $K_i = \overline{A}_{i+1} P_7$
P_8	A←ShlA	左移	$J_i = A_{i-1} P_7$, $K_i = \overline{A}_{i-1} P_7$
P_9	A←A+1	加 1	$J_1 = K_1 = P_9 = E_1$, $E_{i+1} = A_i E_i$, $J_i = K_i = E_i$
	If(A=0)then(Z=1)	检测 0	$Z_1 = 1$, $Z_{i+1} = \overline{A}_i Z_i$, $Z = Z_{n+1}$



累加器的一个典型单元的设计

$$J_i = (B_i \overline{C_{i-1}} P_1 + \overline{B_i} C_{i-1} P_1) + (P_3) + (B_i P_5) + (B_i P_6) + (A_{i+1} P_7) + (A_{i-1} P_8) + E_i$$

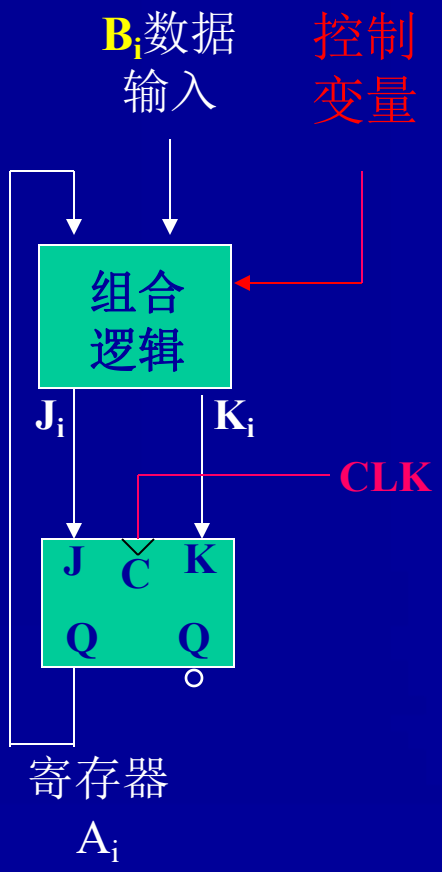
$$K_i = (B_i \overline{C_{i-1}} P_1 + \overline{B_i} C_{i-1} P_1) + (P_2) + (P_3) + (\overline{B_i} P_4) + (B_i P_6) + (\overline{A_{i+1}} P_7) + (\overline{A_{i-1}} P_8) + E_i$$

$$C_i = A_i B_i + A_i C_{i-1} + B_i C_{i-1}$$

$$E_{i+1} = A_i E_i$$

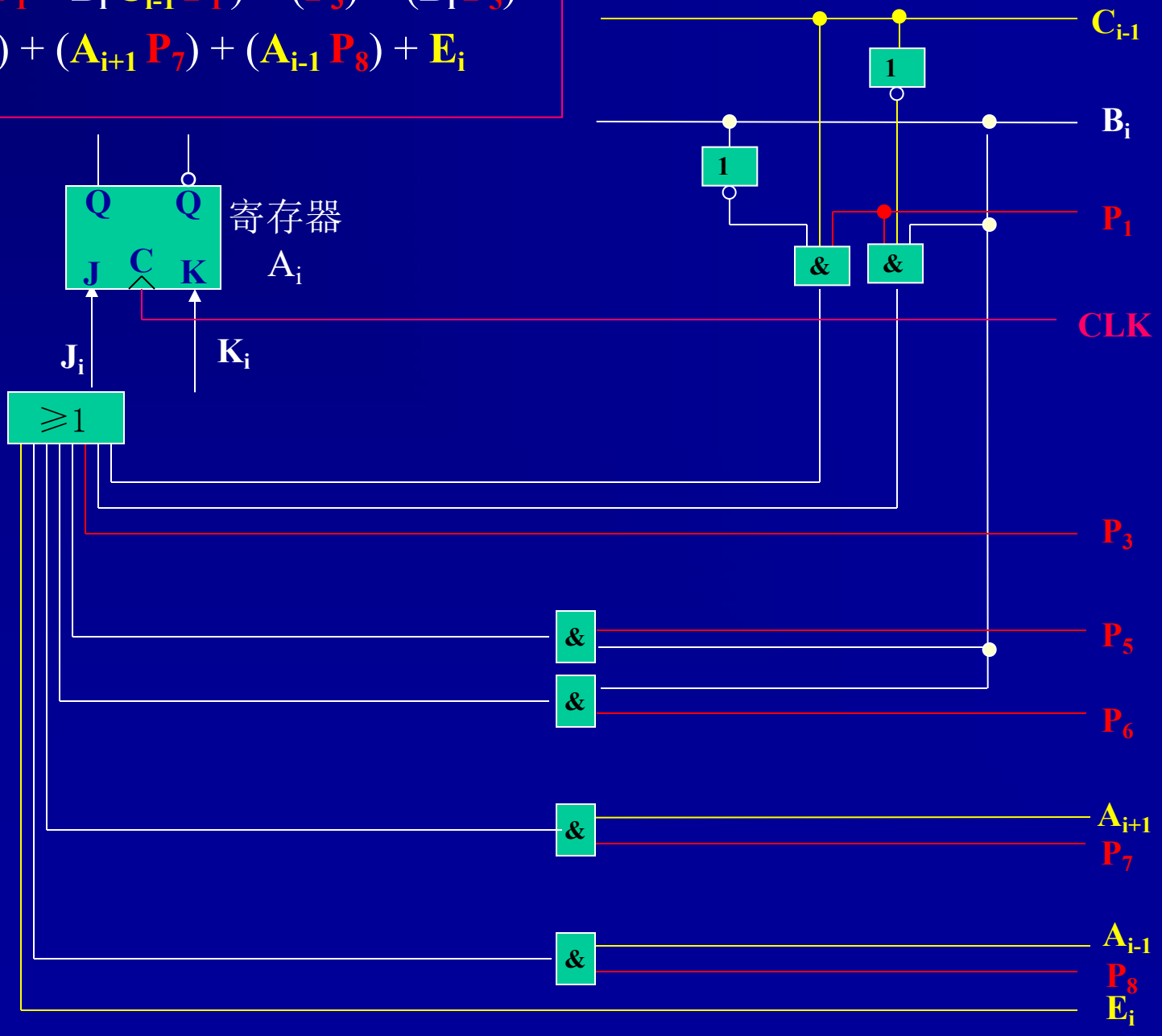
$$Z_{i+1} = \overline{A_i} Z_i$$

控制变量	微操作	名称	逻辑函数表达式
P_1	A←A+B	加	$J_i = K_i = B_i\overline{C_{i-1}}P_1 + \overline{B_i}C_{i-1}P_1$, $C_i = A_iB_i + A_iC_{i-1} + B_iC_{i-1}$
P_2	A←0	清 0	$J_i = 0$, $K_i = P_2$
P_3	A←A	取反	$J_i = P_3$, $K_i = P_3$
P_4	A←A∧B	与	$J_i = 0$, $K_i = \overline{B_i} P_4$
P_5	A←A∨B	或	$J_i = B_i P_5$, $K_i = 0$
P_6	A←A⊕B	异或	$J_i = B_i P_6$, $K_i = B_i P_6$
P_7	A←ShrA	右移	$J_i = A_{i+1} P_7$, $K_i = \overline{A_{i+1}} P_7$
P_8	A←ShlA	左移	$J_i = A_{i-1} P_7$, $K_i = \overline{A_{i-1}} P_7$
P_9	A←A+1	加 1	$J_1 = K_1 = P_9 = E_1$, $E_{i+1} = A_i E_i$, $J_i = K_i = E_i$
	If(A=0)then(Z=1)	检测 0	$Z_1 = 1$, $Z_{i+1} = \overline{A_i} Z_i$, $Z = Z_{n+1}$



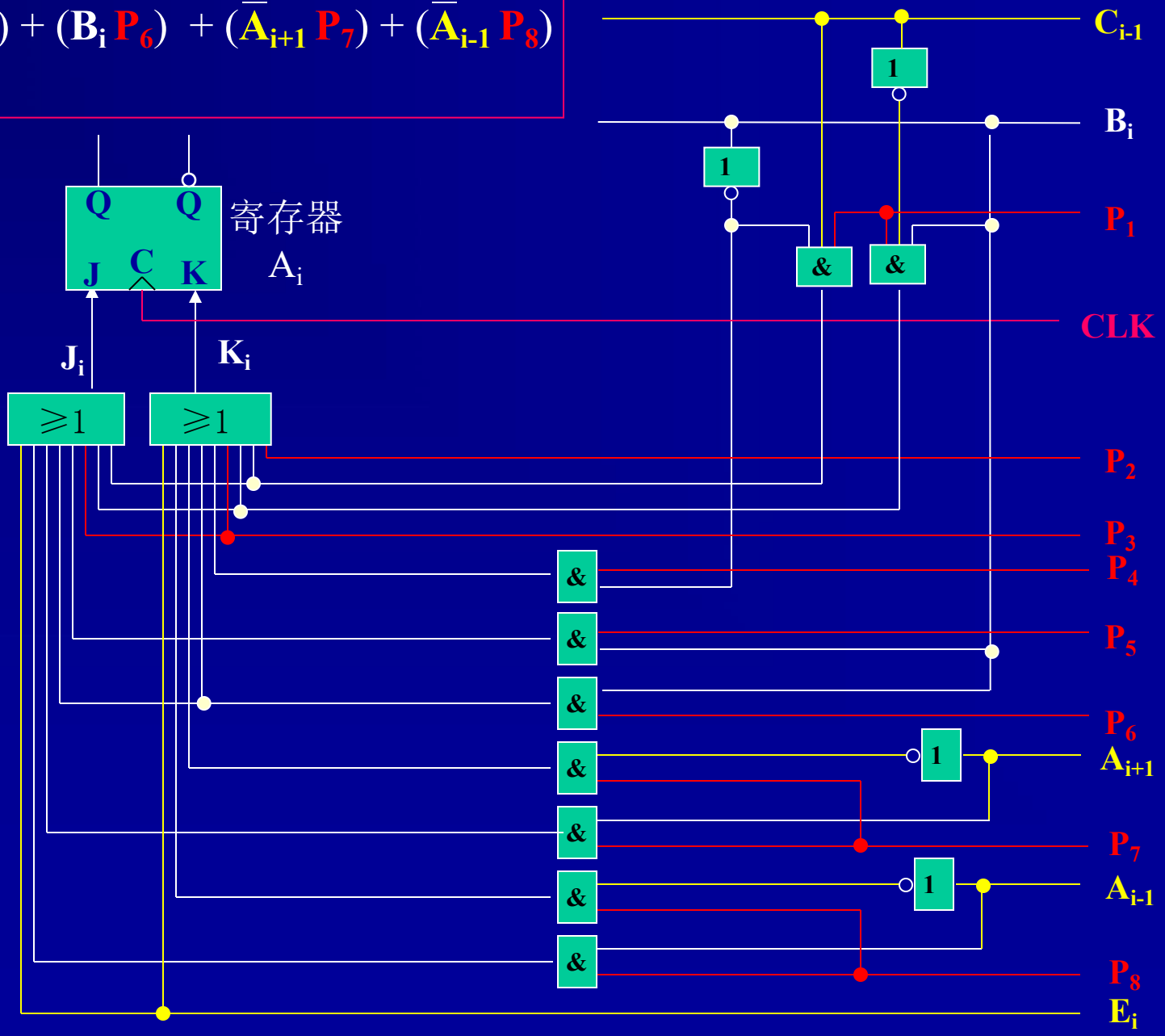
累加器的一个典型单元的逻辑电路图

$$J_i = (B_i \overline{C}_{i-1} P_1 + \overline{B}_i C_{i-1} P_1) + (P_3) + (B_i P_5) + (B_i P_6) + (A_{i+1} P_7) + (A_{i-1} P_8) + E_i$$



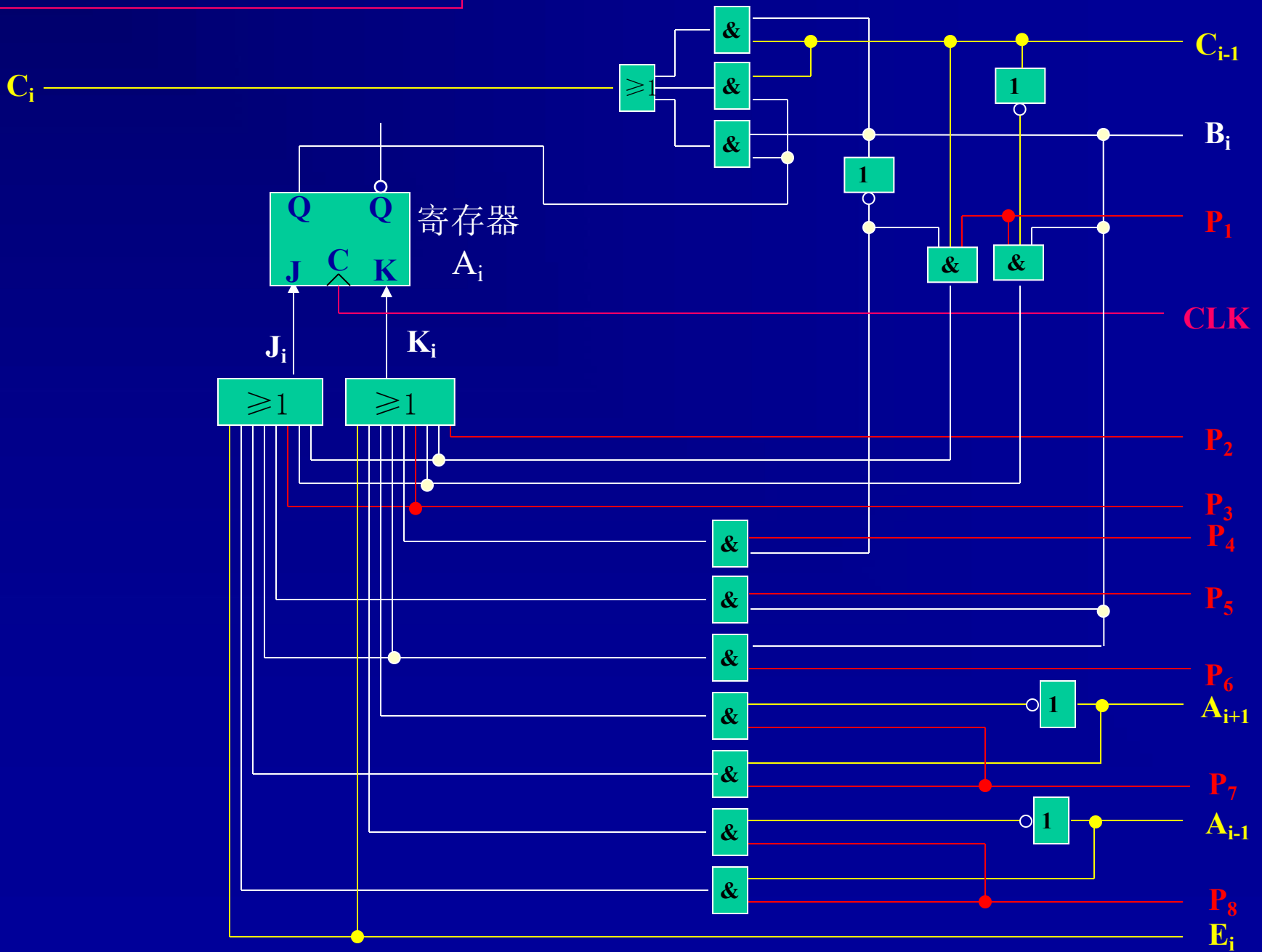
累加器的一个典型单元的逻辑电路图

$$K_i = (B_i \overline{C}_{i-1} P_1 + \overline{B}_i C_{i-1} P_1) + (P_2) + (P_3) + (\overline{B}_i P_4) + (B_i P_6) + (\overline{A}_{i+1} P_7) + (\overline{A}_{i-1} P_8) + E_i$$



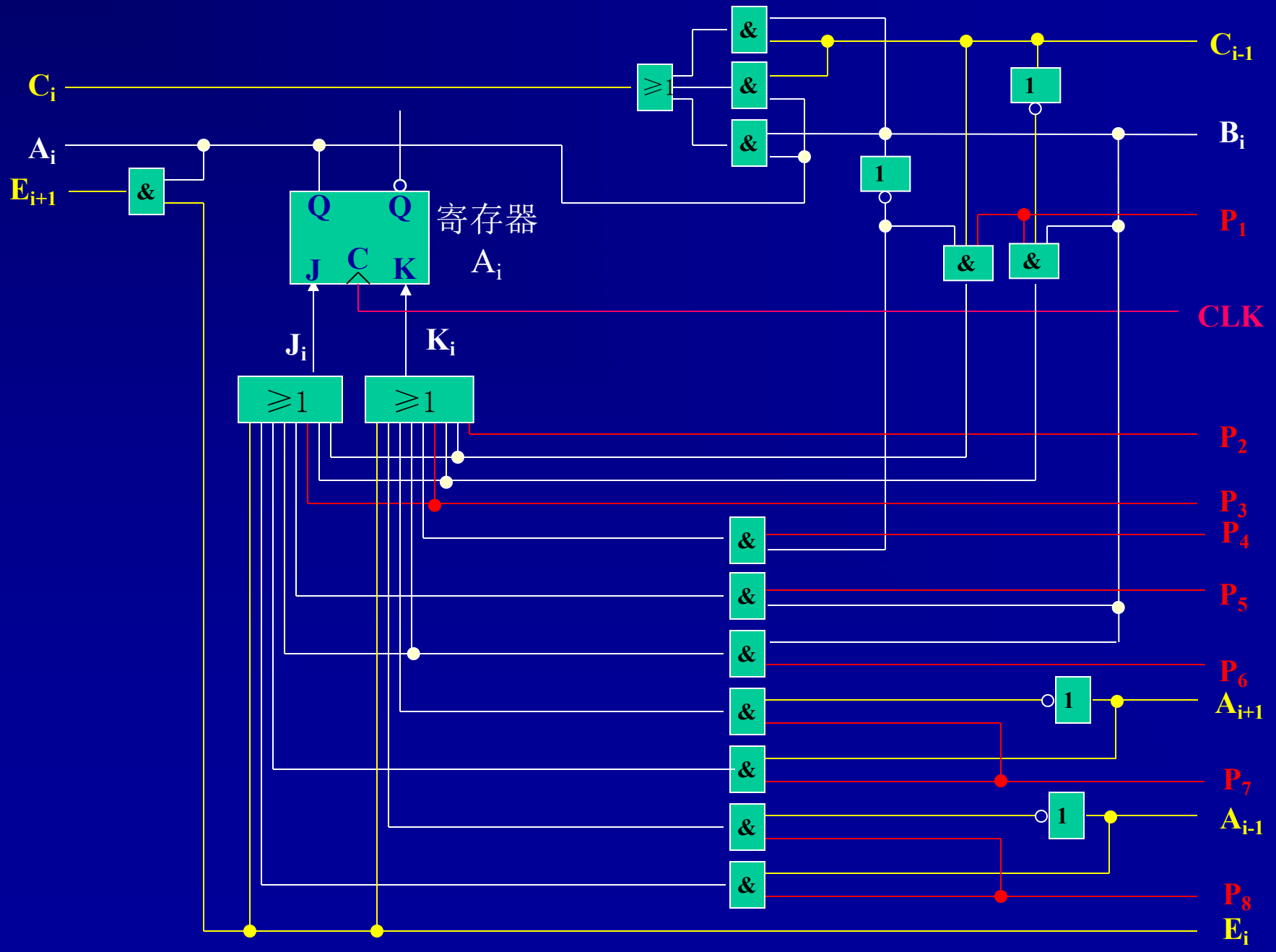
$C_i = A_i B_i + A_i C_{i-1} + B_i C_{i-1}$

累加器的一个典型单元的逻辑电路图



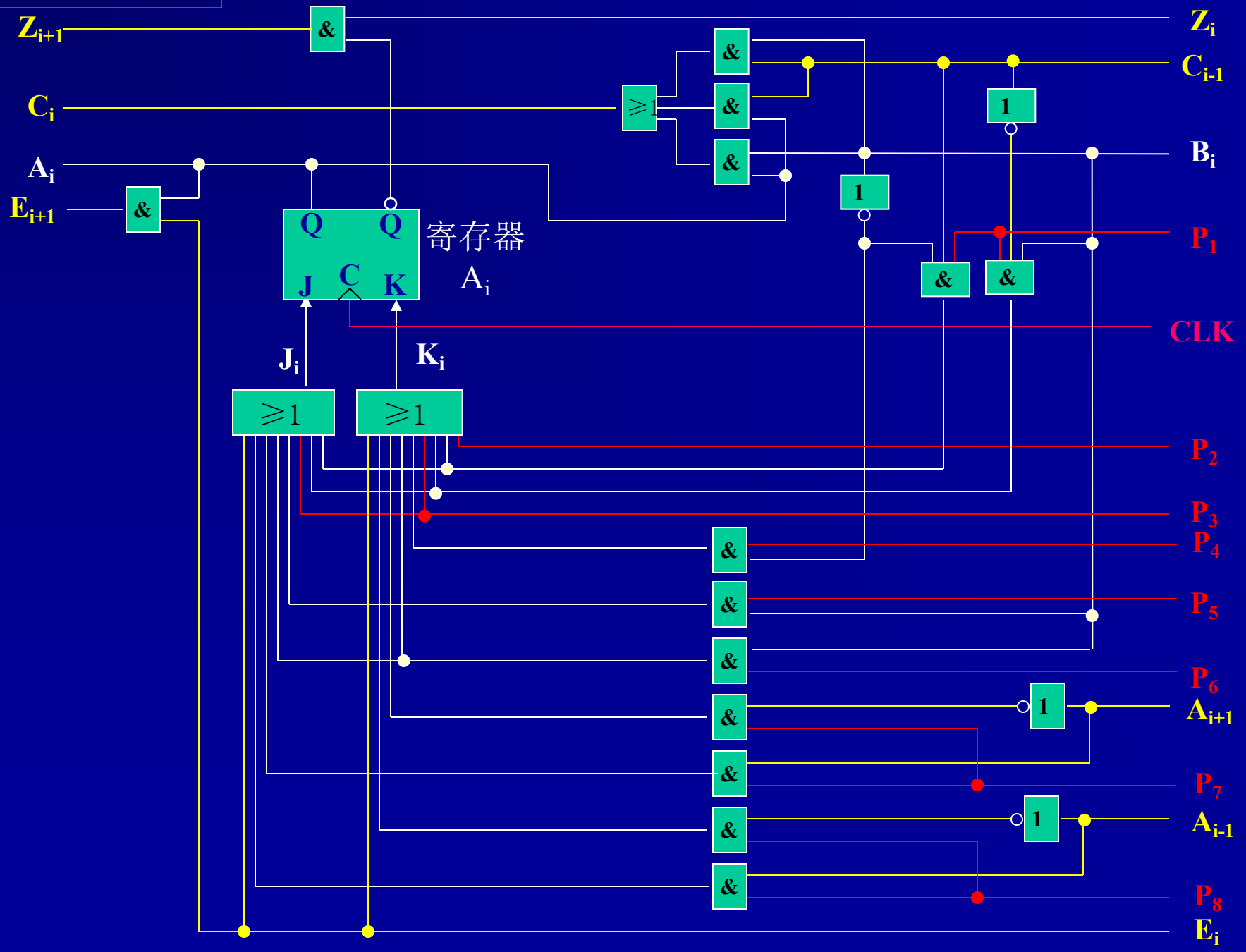
$E_{i+1} = A_i E_i$

累加器的一个典型单元的逻辑电路图

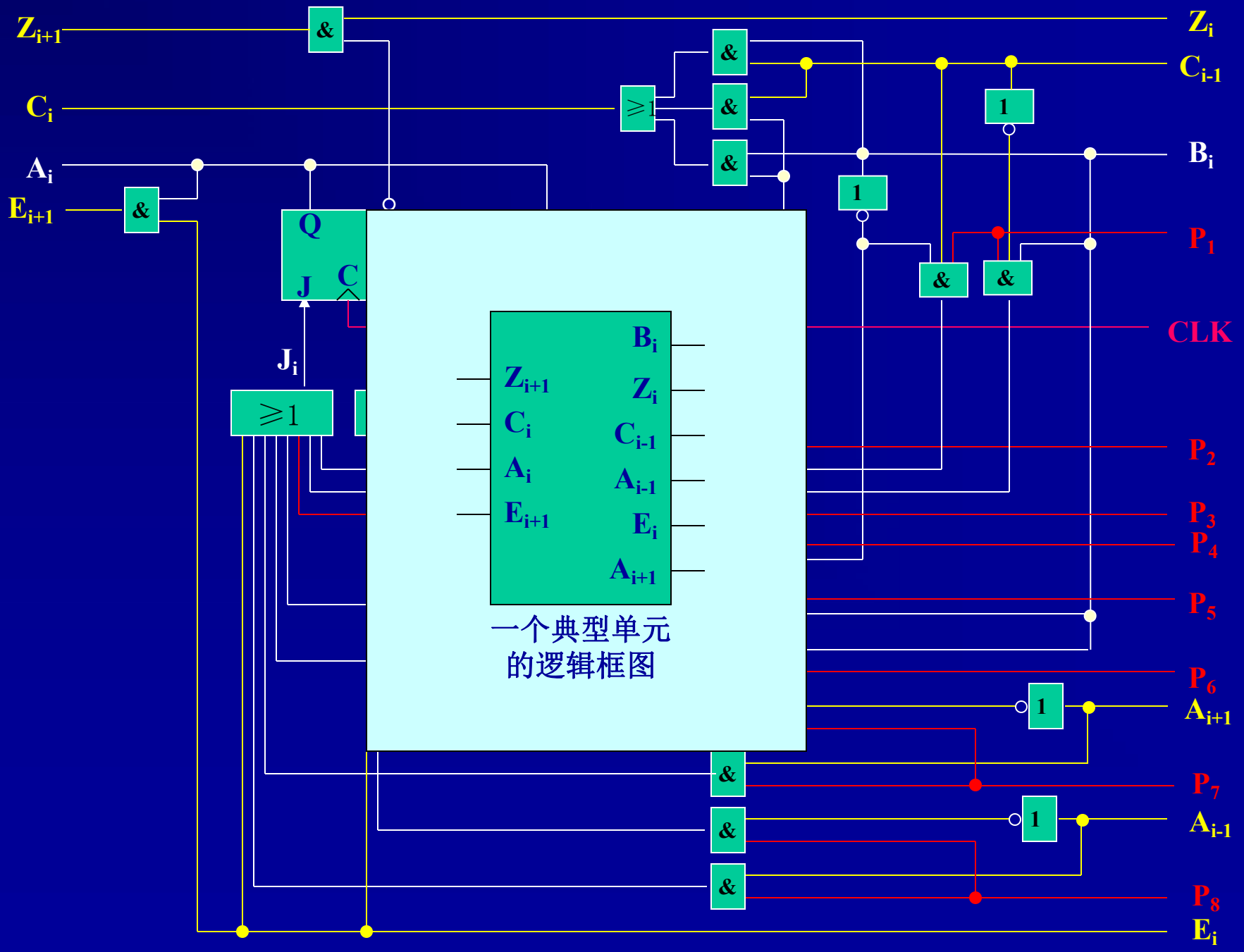


$Z_{i+1} = \overline{A_i} Z_i$

累加器的一个典型单元的逻辑电路图

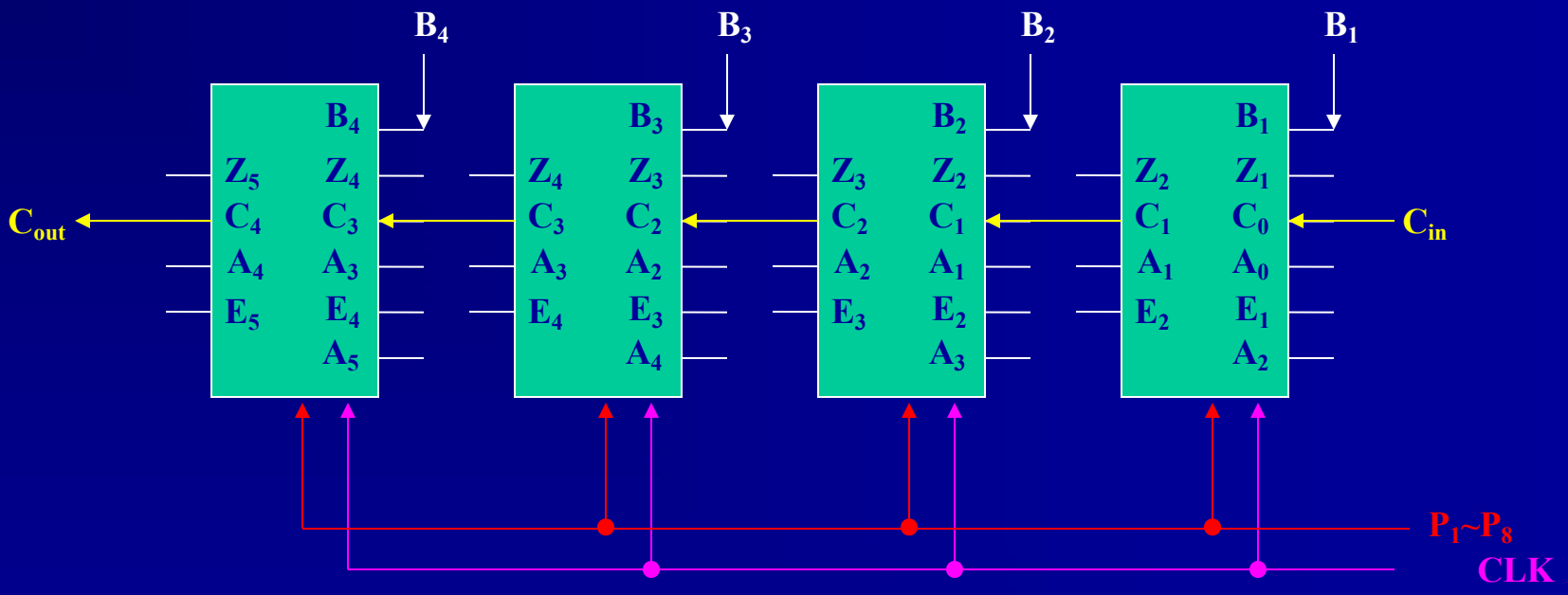


累加器的一个典型单元的逻辑电路图



一个典型单元
的逻辑框图

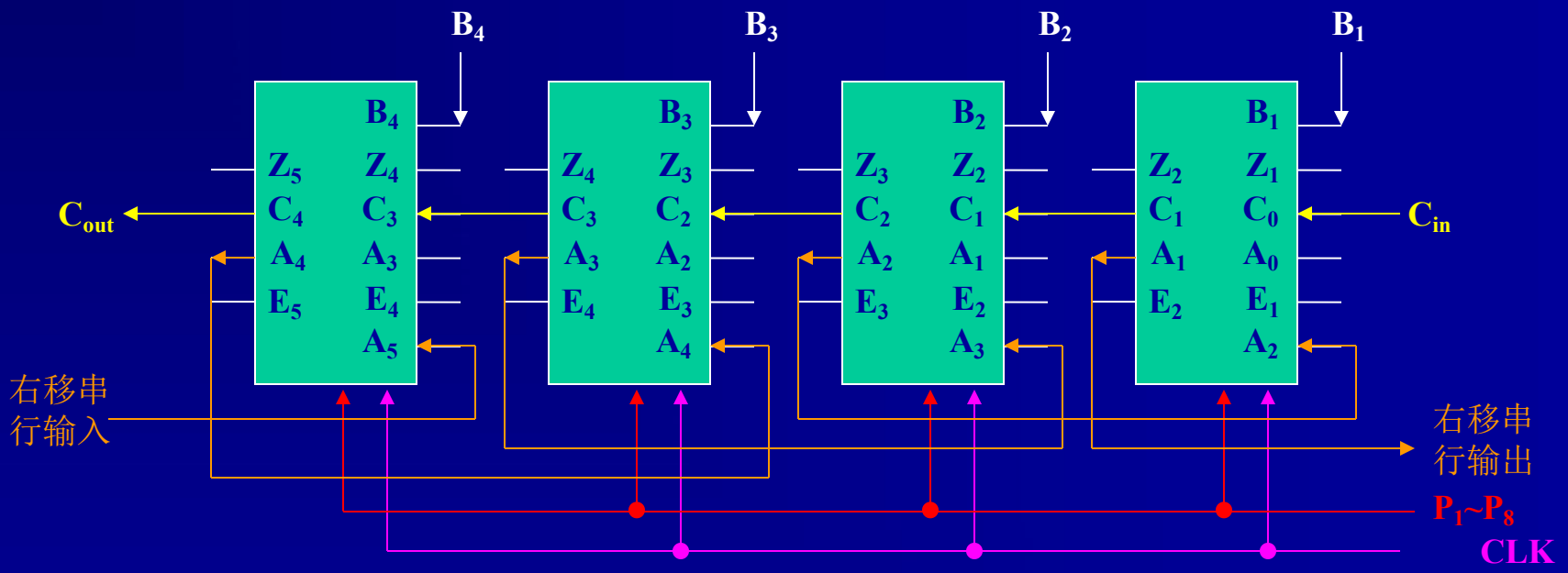
用 n 个典型单元构成一个 n 位累加器（迭代设计）。



上图所示的是一个 4 位累加器的框图。

控制变量	微操作	名称	逻辑函数表达式
P_1	A←A+B	加	$J_i = K_i = B_i\overline{C}_{i-1}P_1 + \overline{B}_i C_{i-1}P_1$, $C_i = A_iB_i + A_iC_{i-1} + B_iC_{i-1}$
P_2	A←0	清 0	$J_i = 0$, $K_i = P_2$
P_3	A←A	取反	$J_i = P_3$, $K_i = P_3$
P_4	A←A∧B	与	$J_i = 0$, $K_i = \overline{B}_i P_4$
P_5	A←A∨B	或	$J_i = B_i P_5$, $K_i = 0$
P_6	A←A⊕B	异或	$J_i = B_i P_6$, $K_i = B_i P_6$
P_7	A←ShrA	右移	
P_8	A←ShlA	左移	
P_9	A←A+1	加 1	
	If(A=0)then(Z=1)	检测 0	

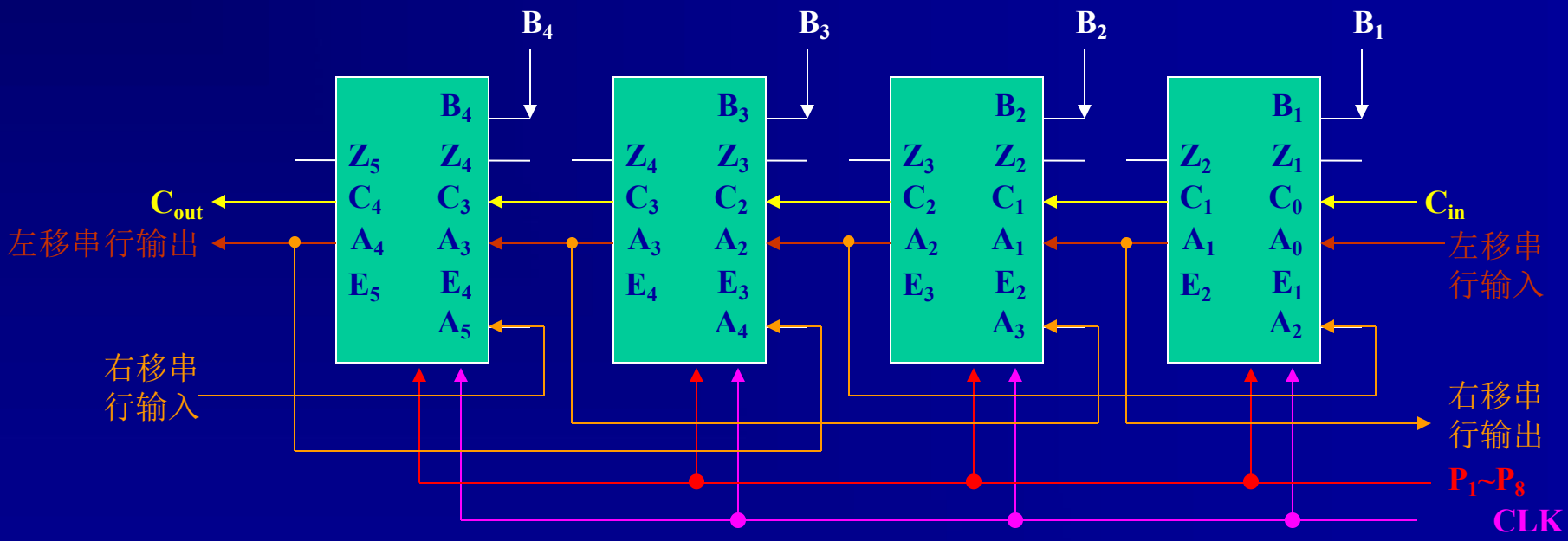
用 n 个典型单元构成一个 n 位累加器（迭代设计）。



上图所示的是一个 4 位累加器的框图。

控制变量	微操作	名称	逻辑函数表达式
P_1	A←A+B	加	$J_i = K_i = B_i\overline{C}_{i-1}P_1 + \overline{B}_i C_{i-1}P_1$, $C_i = A_iB_i + A_iC_{i-1} + B_iC_{i-1}$
P_2	A←0	清 0	$J_i = 0$, $K_i = P_2$
P_3	A←A	取反	$J_i = P_3$, $K_i = P_3$
P_4	A←A∧B	与	$J_i = 0$, $K_i = \overline{B}_i P_4$
P_5	A←A∨B	或	$J_i = B_i P_5$, $K_i = 0$
P_6	A←A⊕B	异或	$J_i = B_i P_6$, $K_i = B_i P_6$
P_7	A←ShrA	右移	$J_i = A_{i+1} P_7$, $K_i = \overline{A}_{i+1} P_7$
P_8	A←ShlA	左移	
P_9	A←A+1	加 1	
	If(A=0)then(Z=1)	检测 0	

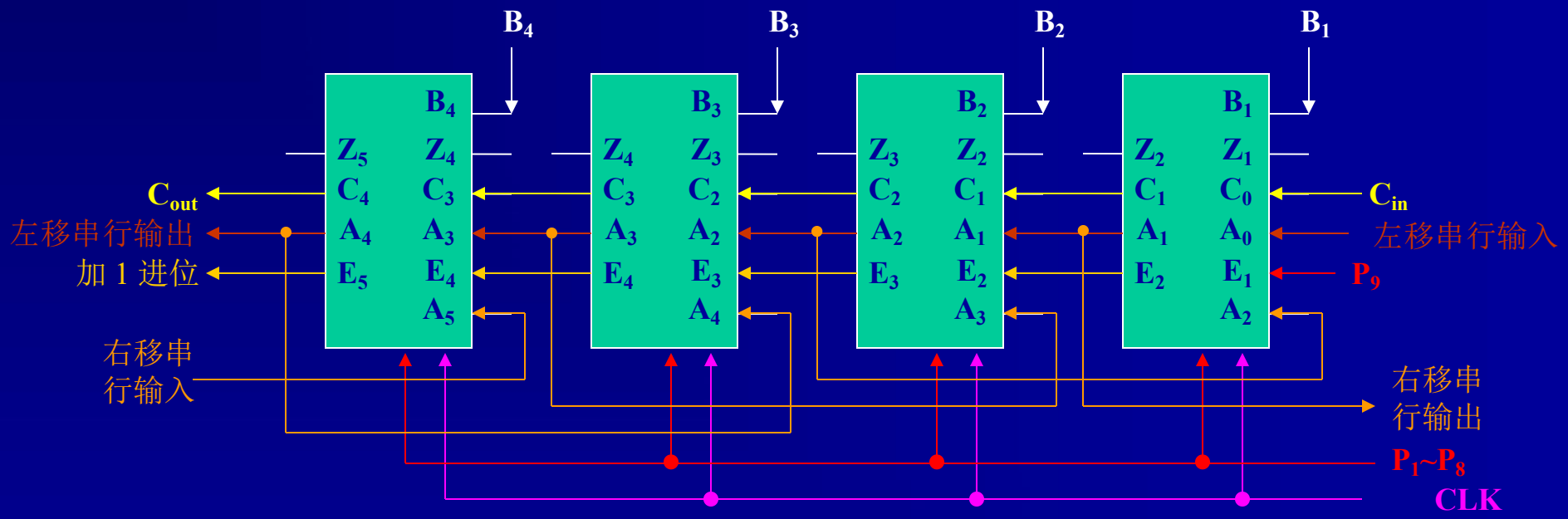
用 n 个典型单元构成一个 n 位累加器（迭代设计）。



上图所示的是一个 4 位累加器的框图。

控制变量	微操作	名称	逻辑函数表达式
P_1	A←A+B	加	$J_i = K_i = B_i\overline{C}_{i-1}P_1 + \overline{B}_i C_{i-1}P_1$, $C_i = A_iB_i + A_iC_{i-1} + B_iC_{i-1}$
P_2	A←0	清 0	$J_i = 0$, $K_i = P_2$
P_3	A←A	取反	$J_i = P_3$, $K_i = P_3$
P_4	A←A∧B	与	$J_i = 0$, $K_i = \overline{B}_i P_4$
P_5	A←A∨B	或	$J_i = B_i P_5$, $K_i = 0$
P_6	A←A⊕B	异或	$J_i = B_i P_6$, $K_i = B_i P_6$
P_7	A←ShrA	右移	$J_i = A_{i+1} P_7$, $K_i = \overline{A}_{i+1} P_7$
P_8	A←ShlA	左移	$J_i = A_{i-1} P_7$, $K_i = \overline{A}_{i-1} P_7$
P_9	A←A+1	加 1	
	If(A=0)then(Z=1)	检测 0	

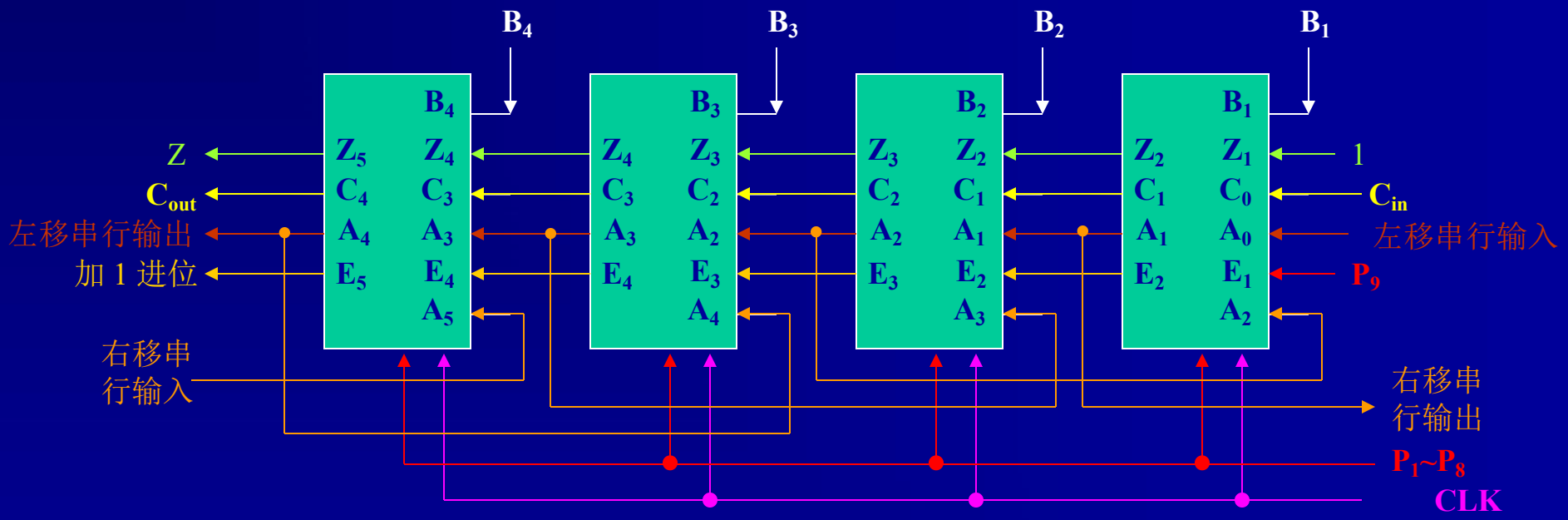
用 n 个典型单元构成一个 n 位累加器（迭代设计）。



上图所示的是一个 4 位累加器的框图。

控制变量	微操作	名称	逻辑函数表达式
P_1	A←A+B	加	$J_i = K_i = B_i\overline{C}_{i-1}P_1 + \overline{B}_i C_{i-1}P_1$, $C_i = A_iB_i + A_iC_{i-1} + B_iC_{i-1}$
P_2	A←0	清 0	$J_i = 0$, $K_i = P_2$
P_3	A←A	取反	$J_i = P_3$, $K_i = P_3$
P_4	A←A∧B	与	$J_i = 0$, $K_i = \overline{B}_i P_4$
P_5	A←A∨B	或	$J_i = B_i P_5$, $K_i = 0$
P_6	A←A⊕B	异或	$J_i = B_i P_6$, $K_i = B_i P_6$
P_7	A←ShrA	右移	$J_i = A_{i+1} P_7$, $K_i = \overline{A}_{i+1} P_7$
P_8	A←ShlA	左移	$J_i = A_{i-1} P_7$, $K_i = \overline{A}_{i-1} P_7$
P_9	A←A+1	加 1	$J_1 = K_1 = P_9 = E_1$, $E_{i+1} = A_i E_i$, $J_i = K_i = E_i$
	If(A=0)then(Z=1)	检测 0	

用 n 个典型单元构成一个 n 位累加器（迭代设计）。



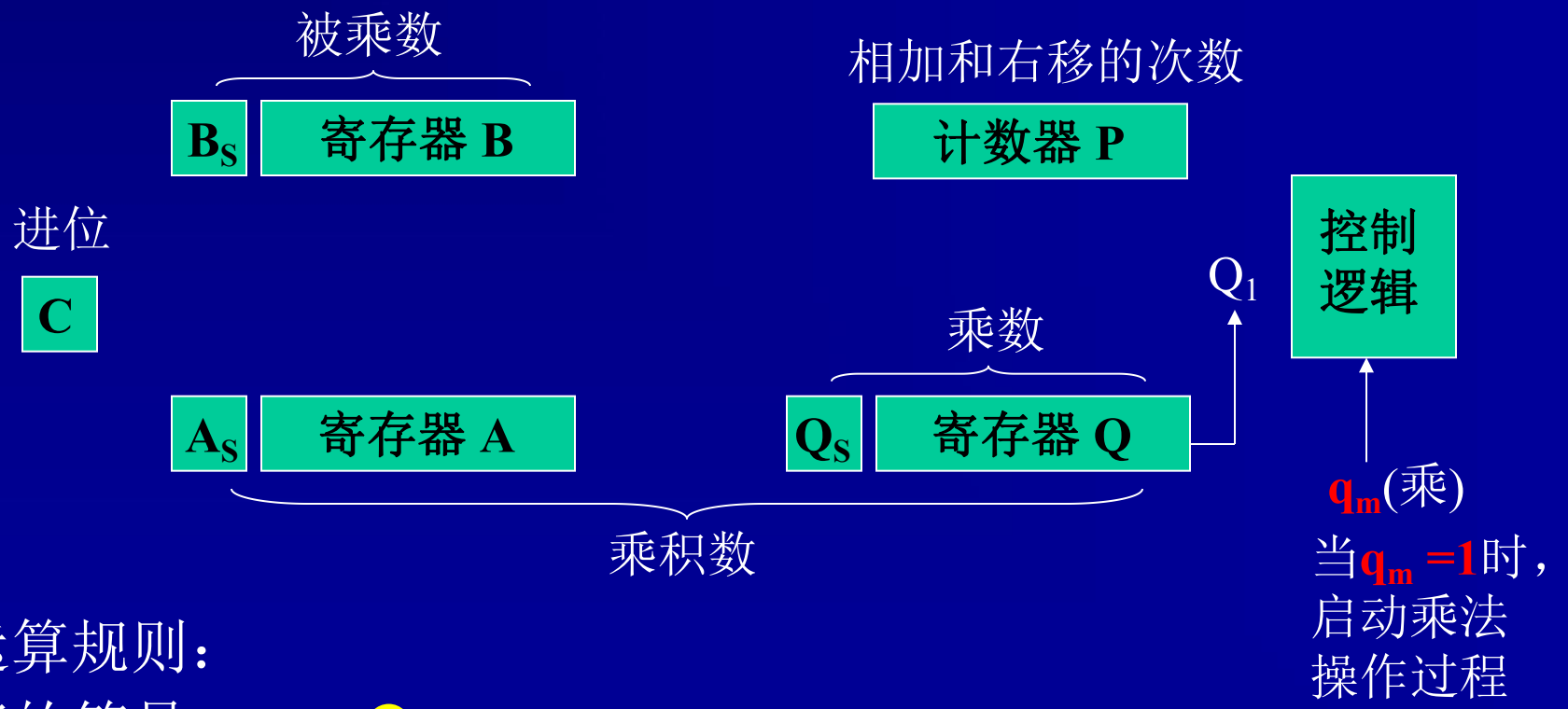
上图所示的是一个 4 位累加器的框图。

控制变量	微操作	名称	逻辑函数表达式
P_1	A←A+B	加	$J_i = K_i = B_i\overline{C}_{i-1}P_1 + \overline{B}_i C_{i-1}P_1$, $C_i = A_iB_i + A_iC_{i-1} + B_iC_{i-1}$
P_2	A←0	清 0	$J_i = 0$, $K_i = P_2$
P_3	A←A	取反	$J_i = P_3$, $K_i = P_3$
P_4	A←A∧B	与	$J_i = 0$, $K_i = \overline{B}_i P_4$
P_5	A←A∨B	或	$J_i = B_i P_5$, $K_i = 0$
P_6	A←A⊕B	异或	$J_i = B_i P_6$, $K_i = B_i P_6$
P_7	A←ShrA	右移	$J_i = A_{i+1} P_7$, $K_i = \overline{A}_{i+1} P_7$
P_8	A←ShlA	左移	$J_i = A_{i-1} P_7$, $K_i = \overline{A}_{i-1} P_7$
P_9	A←A+1	加 1	$J_1 = K_1 = P_9 = E_1$, $E_{i+1} = A_i E_i$, $J_i = K_i = E_i$
	If(A=0)then(Z=1)	检测 0	$Z_1 = 1$, $Z_{i+1} = \overline{A}_i Z_i$, $Z = Z_{n+1}$

例2. 试设计一个能对两个原码表示的定点二进制数相乘的乘法器控制单元。

设：两个二进制数均为 n 位原码，其中最左位是符号位。

硬件配置：



运算规则：

积的符号$A_S = B_S \oplus Q_S$，

积的数值由**部分积和被乘数逐次累加并右移获得**，积长**2n**位。

运算规则：

“移位—相加”算法举例

```
      10111
  ×) 10011
  ---------
      10111      乘数第1位为1
     10111       乘数第2位为1
    00000        乘数第3位为0
   00000         乘数第4位为0
  10111          乘数第5位为1
  ---------
  110110101      乘积
```

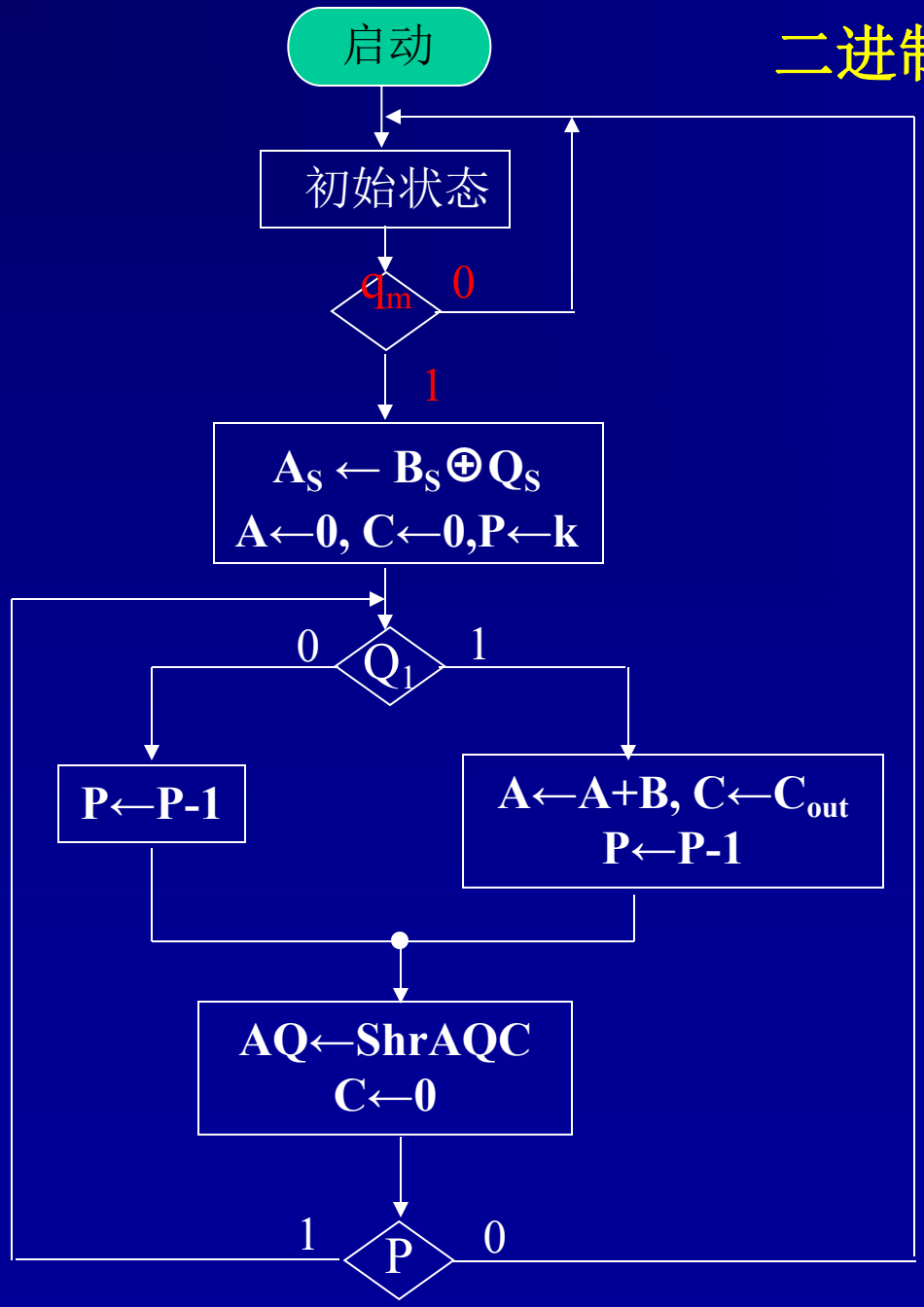
修改算法

```
  10111
  10011
  ---------
  10111          乘数第1位为1
  010111         右移，得到第1个 部分积
  10111          乘数第2位为1
  ---------
  1000101        被乘数与部分积累加
   1000101       右移，得到第2个 部分积
  01000101       乘数第3位为0，直接右移，得到第3个 部分积
 001000101       乘数第4位为0，直接右移，得到第4个 部分积
  10111          乘数第5位为1
  ---------
  110110101      被乘数与部分积累加
 0110110101      右移，得到第5个 部分积，也是最后的结果
```

算法：被乘数——左移

算法：逐次累加被乘数——右移

二进制数相乘的“算法”的逻辑流程图 (乘数的数值位数为k)

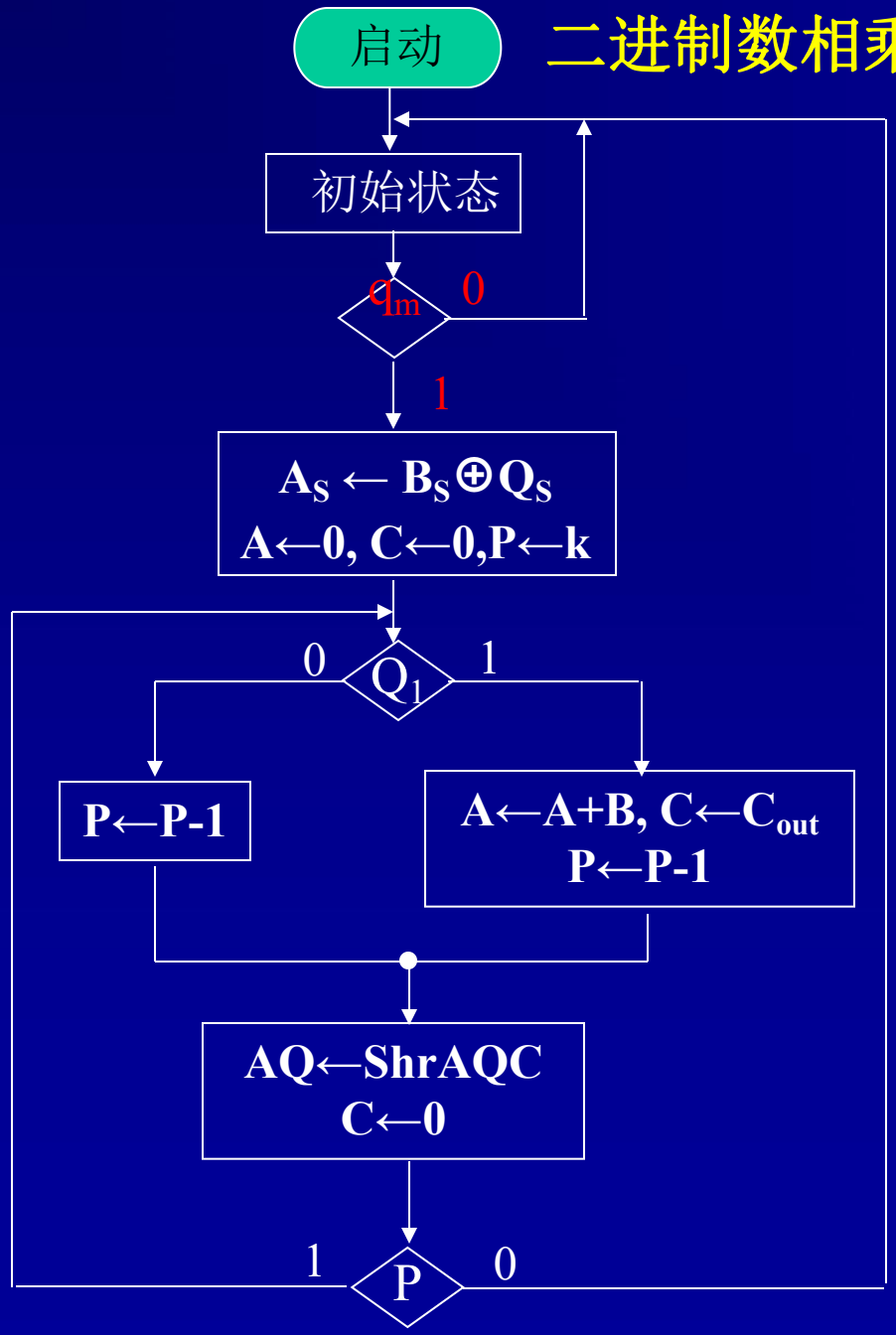


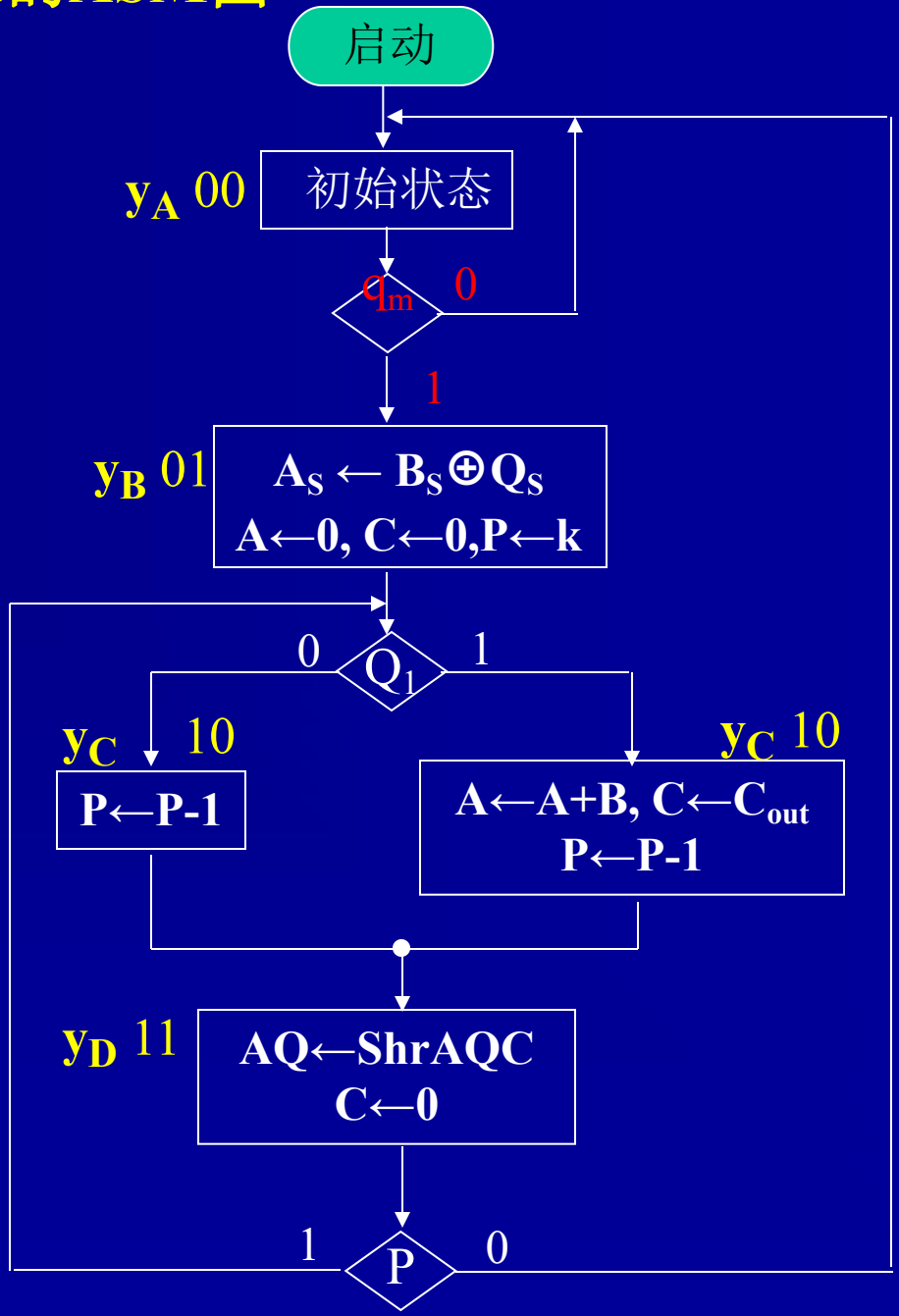
10111
10011

10111　乘数第1位为**1**
010111　右移，得到**第1个** 部分积
10111　乘数第2位为**1**
1000101　被乘数与部分积累加
1000101　右移，得到**第2个** 部分积
01000101　乘数第3位为**0**，直接右移，得到**第3个** 部分积
001000101　乘数第4位为**0**，直接右移，得到**第4个** 部分积
10111　乘数第5位为**1**
110110101　被乘数与部分积累加
0110110101　右移，得到**第5个** 部分积，也是最后的结果

算法：逐次累加被乘数——右移

二进制数相乘的ASM图





寄存器的微操作序列：

T_0： 初始状态

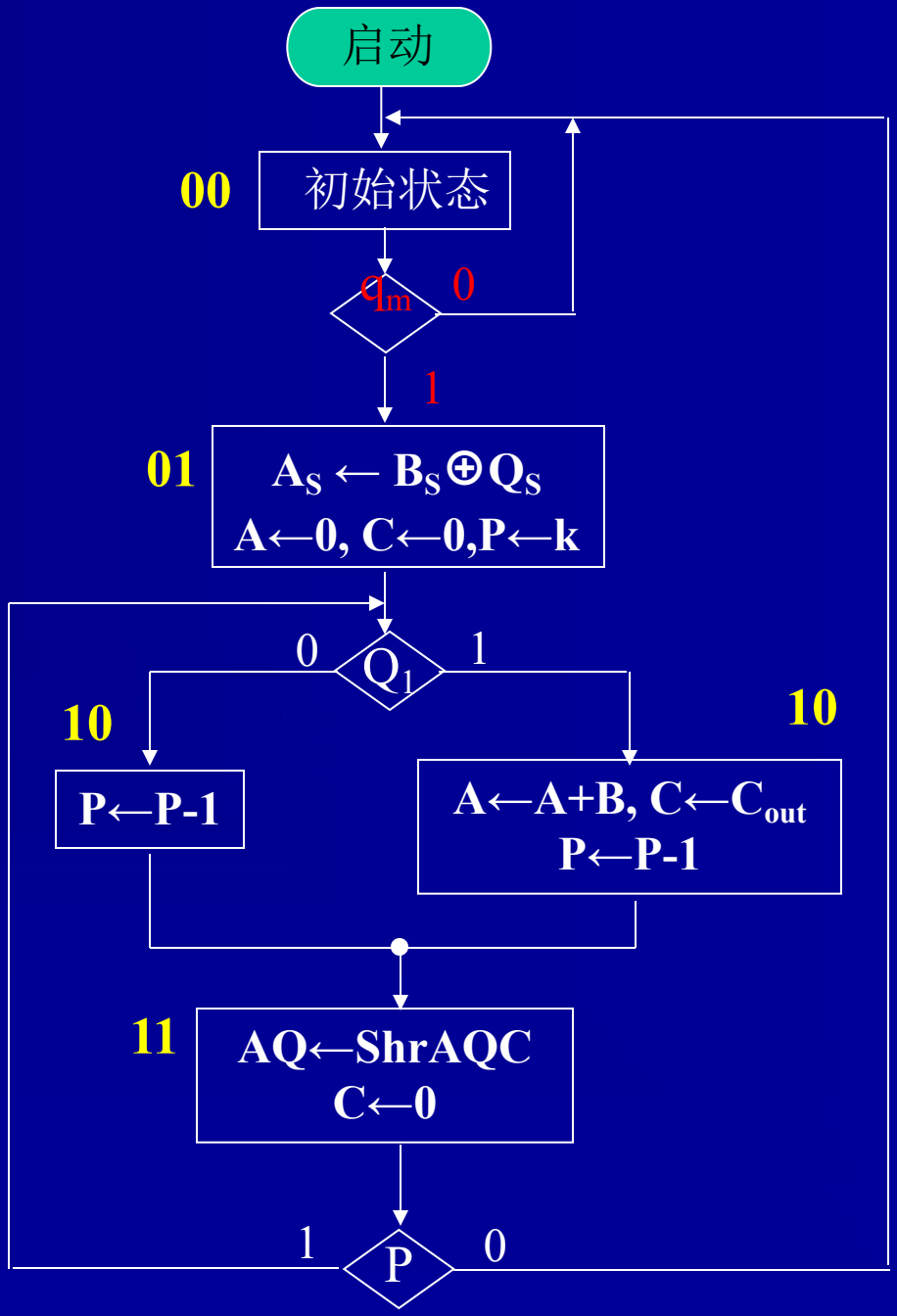
T_1： $A_S \leftarrow B_S \oplus Q_S$， A←0, C←0，P←k

$Q_1 T_2$： A←A+B，$C \leftarrow C_{out}$ P←P-1

$\overline{Q_1} T_2$： P←P-1

T_3： AQ ← ShrCAQ，C←0

乘法器的初始状态为T_0，当乘法运算控制变量q_m =1时，进入状态T_1，对进位触发器C、寄存器A、计数器P置初值，然后进入乘法操作。



乘法控制器的状态真值表

现 态		输 入			现 态		输 出				
y_2	y_1	q_m	P_{zero}	Q_1	y_2^{n+1}	y_1^{n+1}	T_0	T_1	T_2	Q_1T_2	T_3
0	0	0	d /—	d	0	0	1	0	0	0	0
0	0	1	d	d	0	1	1	0	0	0	0
0	1	d	d	d	1	0	0	1	0	0	0
1	0	d	d	0	1	1	0	0	1	0	0
1	0	d	d	1	1	1	0	0	1	1	0
1	1	d	0	d	1	0	0	0	0	0	1
1	1	d	1	d	0	0	0	0	0	0	1

注： $P_{zero} = 1$ 当计数器P的内容为0时， 否则为 $P_{zero} = 0$；

$L = Q_1 T_2$

T_0： 初始状态

T_1： $A_S \leftarrow B_S \oplus Q_S$， A←0，C←0，P←k

$Q_1 T_2$： A←A+B，C←C_{out} P←P-1

$\overline{Q_1} T_2$： P←P-1

T_3： AQ ← ShrCAQ， C←0