数字逻辑基础

主 讲：何宾

Email：hebin@mail.buct.edu.cn

2014.06

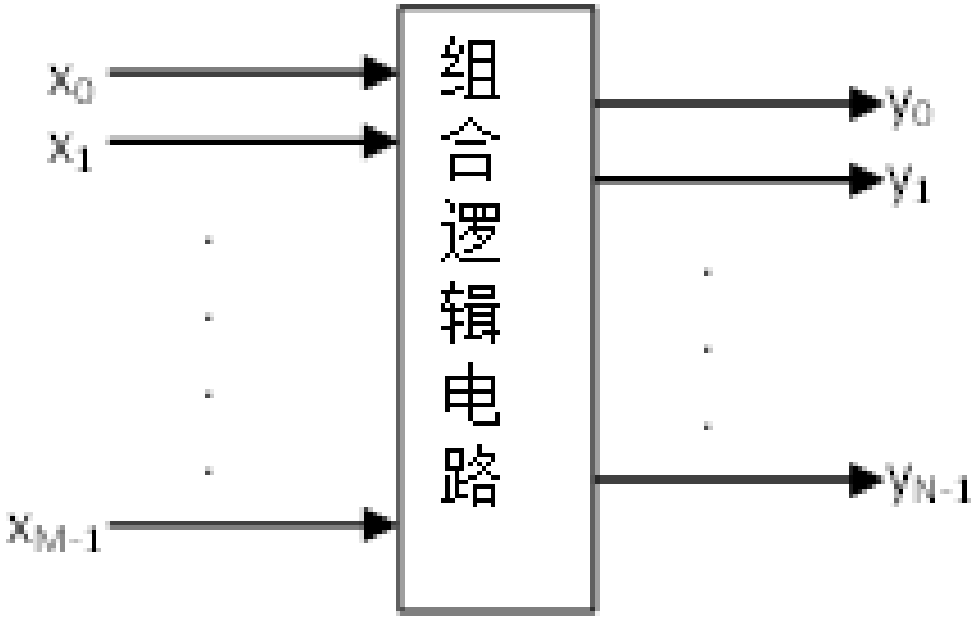
组合逻辑电路
--概念

我们都知道人的大脑具有复杂的推理和记忆功能，人的大脑可以指挥人的四肢进行工作。人是在不断地记忆新知识的过程中成长的。

- 数字系统和人的大脑和四肢有些相似，即：一个完整的数字系统应该包含有推理部分，又应该包含有记忆(存储）部分，还有执行部分。
- 在数字系统中，所谓的推理和执行部分称之为组合逻辑电路；而记忆（存储）部分称之为时序逻辑电路。

组合逻辑电路
--概念

组合逻辑电路（combinational logic）是一种逻辑电路，它的任一时刻的输出，仅仅与当前时刻的逻辑输入变量的取值有关。



组合逻辑电路
--概念

组合逻辑电路的输入和输出可以用下面的关系描述：

$$y_0 = f_0(x_0, x_1, \ldots, x_{M-1})$$
$$y_1 = f_1(x_0, x_1, \ldots, x_{M-1})$$
$$\cdots\cdots\cdots$$
$$y_{N-1} = f_{N-1}(x_0, x_1, \ldots, x_{M-1})$$

其中：

- $x_0, x_1, \ldots, x_{M-1}$为M个逻辑输入变量。
- $y_0, y_1, \ldots, y_{N-1}$为N个逻辑输出变量。
- $f_0()$，$f_1()$，…，$f_{N-1}()$表示M个输入和N个输出的布尔逻辑表达式。

组合逻辑电路
--概念

时序逻辑电路是一种特殊的组合逻辑电路。

- **它的任何一个时刻的输出，不仅与当前时刻的逻辑输入变量的取值有关，而且还和前一时刻的输出有关。其具体的原理将在后面详细说明。**

组合逻辑电路
--类型

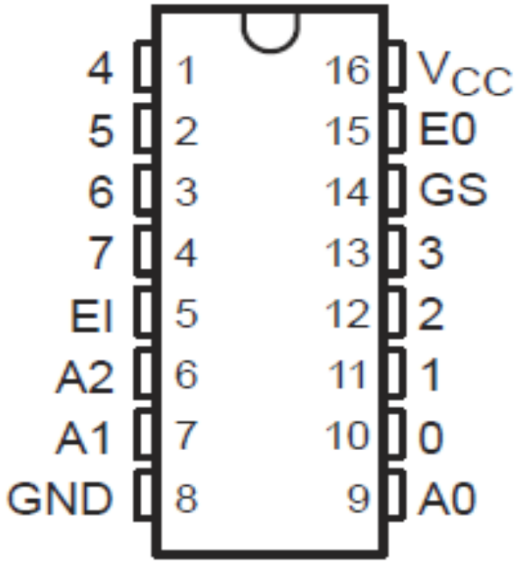
组合逻辑电路包含：

- 编码器
- 译码器
- 数据选择器
- 数据比较器
- 加法器
- 减法器
- 乘法器

组合逻辑电路
--编码器

在数字系统中，编码器用于对原始逻辑信息进行变换。

- **以74LS148 8-3线编码器为例，说明编码器的实现原理。**



其中：

- **EI**（芯片上的第**5**个引脚），选通输入端，低电平有效。
- **EO**（芯片的第**15**引脚），选通输出端，高电平有效。
- **GS**（芯片的第**14**引脚）组选择输出有效信号，用作片优先编码输出端。当**EI**有效，并且有优先编码输入时，该引脚输出为低。
- **A0**、**A1**、**A2**（芯片的第**9**、**7**和**6**引脚），编码输出端，低电平有效。

组合逻辑电路
--编码器

8-3线编码器的真值表

E_1	0	1	2	3	4	5	6	7	A_2	A_1	A_0	GS	E0
1	×	×	×	×	×	×	×	×	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	×	×	×	×	×	×	×	0	0	0	0	0	1
0	×	×	×	×	×	×	0	1	0	0	1	0	1
0	×	×	×	×	×	0	1	1	0	1	0	0	1
0	×	×	×	×	0	1	1	1	0	1	1	0	1
0	×	×	×	0	1	1	1	1	1	0	0	0	1
0	×	×	0	1	1	1	1	1	1	0	1	0	1
0	×	0	1	1	1	1	1	1	1	1	0	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1

组合逻辑电路
--编码器

- 当EI为低有效时，编码器才能正常工作。
- 74LS148输入端优先级别的次序依次为7，6，…，0 。当某一输入端有低电平输入，且比它优先级别高的输入端没有低电平输入时，输出端才输出相应该输入端的代码。
  - 例如：当输入5为逻辑0，且输入6和输入7为逻辑高时，则此时输出代码010，为5（101）的反码表示。

组合逻辑电路
--编码器

对真值表使用布尔代数进行化简，得到下面的逻辑表达式：

$$\overline{A0} = (\bar{1}\cdot 2\cdot 4\cdot 6 + \bar{3}\cdot 4\cdot 6 + \bar{5}\cdot 6 + \bar{7})\cdot \bar{E}_1$$

$$\overline{A1} = (\bar{2}\cdot 4\cdot 5 + \bar{3}\cdot 4\cdot 5 + \bar{6} + \bar{7})\cdot \bar{E}_1$$

$$\overline{A2} = (\bar{4} + \bar{5} + \bar{6} + \bar{7})\cdot \bar{E}_1$$

$$\overline{E0} = 0\cdot 1\cdot 2\cdot 3\cdot 4\cdot 5\cdot 6\cdot 7\cdot \bar{E}_1$$

$$\overline{GS} = \overline{E0}\cdot \bar{E}_1$$

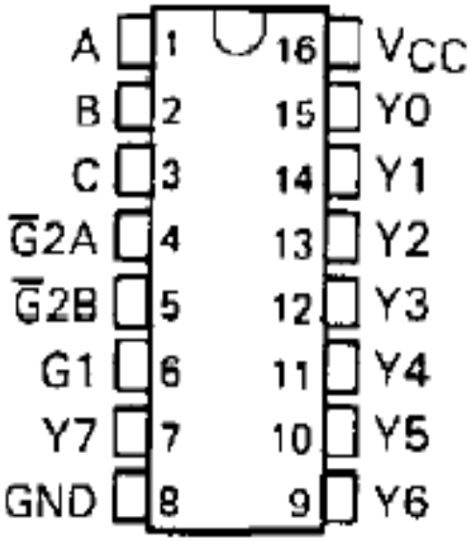
组合逻辑电路
--译码器

译码器是电子技术中的一种多输入多输出的组合逻辑电路。负责将二进制代码翻译为特定的对象（如逻辑电平等）。

- 功能与编码器相反。
- 典型的，3-8译码器，7段码译码器。

组合逻辑电路
--译码器

3-8译码器



编码从C，B，A引脚输入。输出Y7~Y0用来表示输入编码的组合。

组合逻辑电路
--译码器

74LS138 3-8译码器的真值表

G_1	G_{2A}	G_{2B}	C	B	A	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
×	1	×	×	×	×	1	1	1	1	1	1	1	1
×	×	1	×	×	×	1	1	1	1	1	1	1	1
0	×	×	×	×	×	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	0	0	1	1	1	1	0	1	1	1
1	0	0	1	0	1	1	1	1	1	1	0	1	1
1	0	0	1	1	0	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0

组合逻辑电路
--译码器

在一个时刻，输出引脚Y7~Y0中只有一位为低，其余输出均为高，即：

$$\text{当 } CBA = n \text{时，} n \subseteq \{0,1,2,3,4,5,6,7\}$$

输出满足：

$$Y_n = \text{‘0’}, Y_m = \text{‘1’}, m \subseteq \{0,1,2,3,4,5,6,7\} \text{且} m \neq n$$

74LS138译码器正常工作的前提条件是：

$$G_1 = '1', \bar{G}_{2A} = '0', \bar{G}_{2B} = '0'$$

同时有效。

组合逻辑电路
--译码器

根据给出的3-8译码器的真值表，通过化简得到输出Y0~Y7的逻辑表达式：

$$Y0 = (G_1 \cdot \overline{G_{2A}} \cdot \overline{G_{2B}}) \cdot \bar{C} \cdot \bar{B} \cdot \bar{A}$$

$$Y1 = (G_1 \cdot \overline{G_{2A}} \cdot \overline{G_{2B}}) \cdot \bar{C} \cdot \bar{B} \cdot A$$

$$Y2 = (G_1 \cdot \overline{G_{2A}} \cdot \overline{G_{2B}}) \cdot \bar{C} \cdot B \cdot \bar{A}$$

$$Y3 = (G_1 \cdot \overline{G_{2A}} \cdot \overline{G_{2B}}) \cdot \bar{C} \cdot A \cdot B$$

$$Y4 = (G_1 \cdot \overline{G_{2A}} \cdot \overline{G_{2B}}) \cdot C \cdot \bar{B} \cdot \bar{A}$$

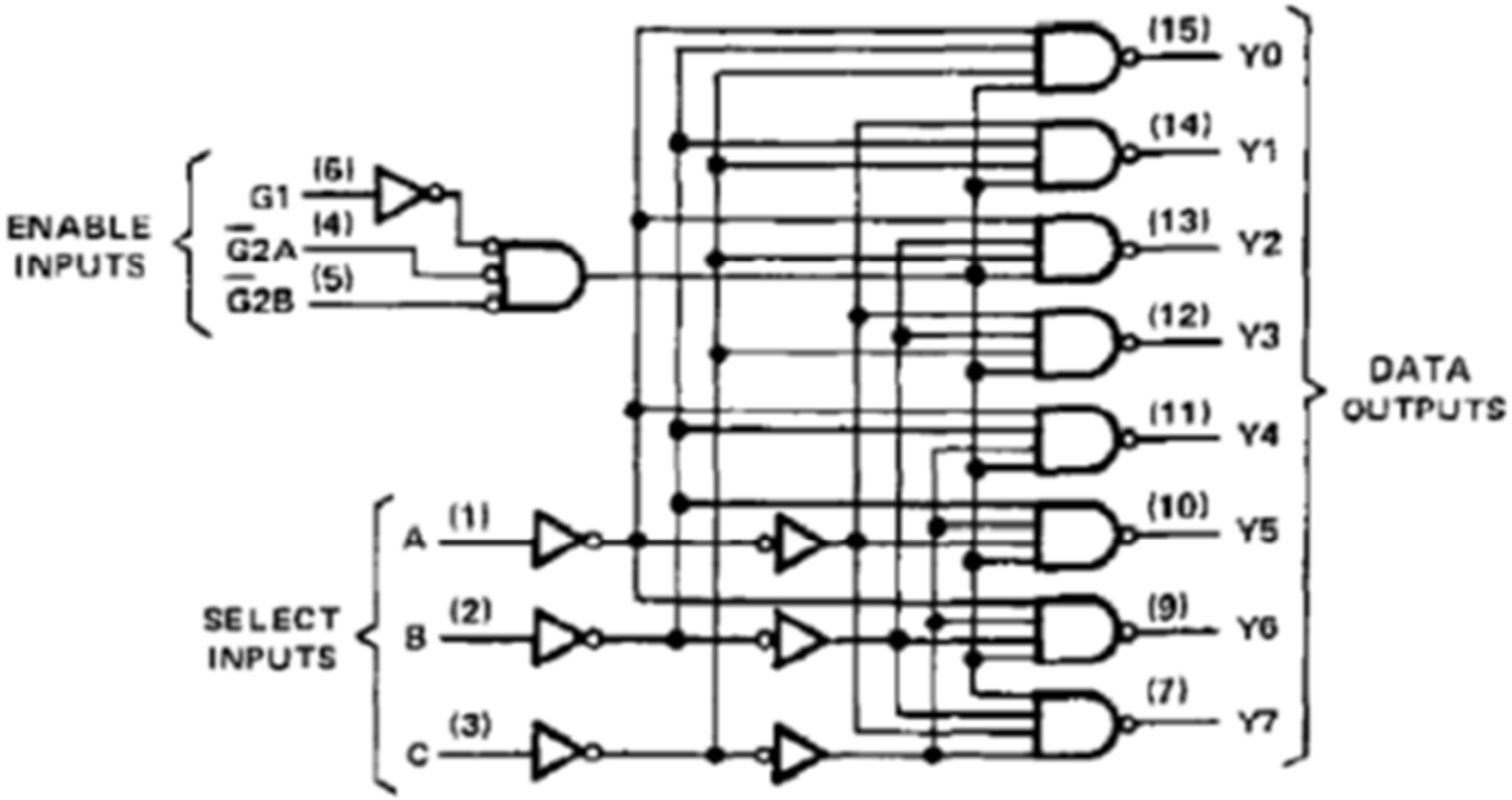
$$Y5 = (G_1 \cdot \overline{G_{2A}} \cdot \overline{G_{2B}}) \cdot C \cdot \bar{B} \cdot A$$

$$Y6 = (G_1 \cdot \overline{G_{2A}} \cdot \overline{G_{2B}}) \cdot C \cdot B \cdot \bar{A}$$

$$Y7 = (G_1 \cdot \overline{G_{2A}} \cdot \overline{G_{2B}}) \cdot C \cdot B \cdot A$$

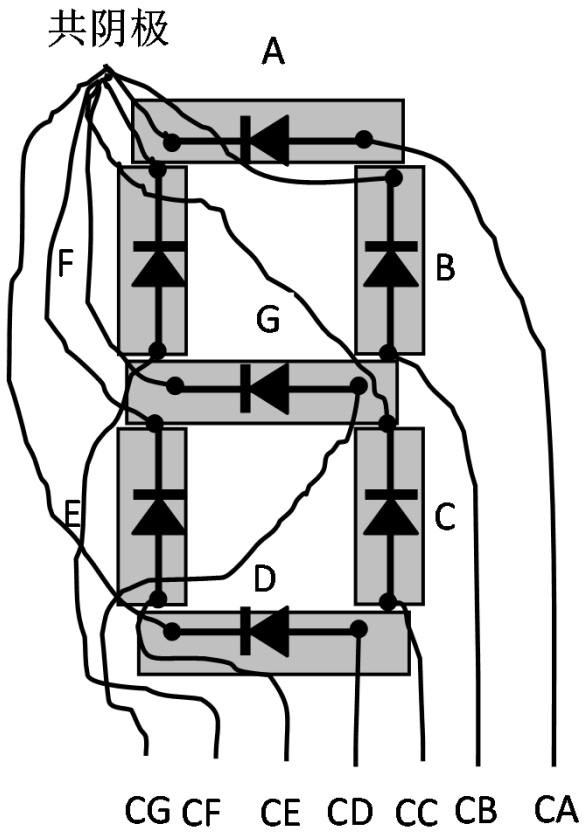
组合逻辑电路
--译码器

74LS138内部结构



组合逻辑电路
--译码器

七段码译码器



共阴极 7 段数码管原理

7段数码管亮灭控制的最基本原理：

当有电流流过7段数码管a,b,c,d,e,f,g的某一段时，该段就发光。

组合逻辑电路
--译码器

（1）V_{CA}-$V_{公共端}$<V_{th}时，A段灭。否则，A段亮。

（2）V_{CB}-$V_{公共端}$<V_{th}时，B段灭。否则，B段亮。

（3）V_{CC}-$V_{公共端}$<V_{th}时，C段灭。否则，C段亮。

（4）V_{CD}-$V_{公共端}$<V_{th}时，D段灭。否则，D段亮。

（5）V_{CE}-$V_{公共端}$<V_{th}时，E段灭。否则，E段亮。

（6）V_{CF}-$V_{公共端}$<V_{th}时，F段灭。否则，F段亮。

（7）V_{CG}-$V_{公共端}$<V_{th}时，G段灭。否则，G段亮。

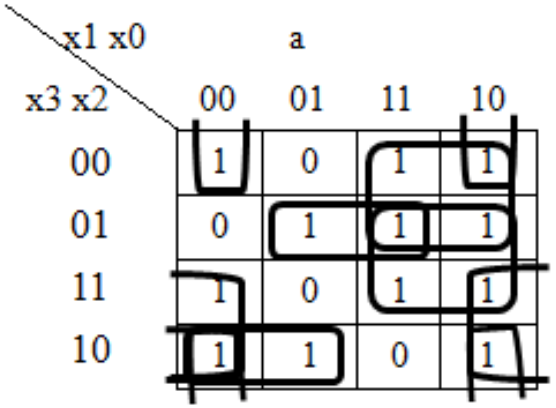
注：V_{th}为七段数码管各段的门限电压。

组合逻辑电路
--译码器

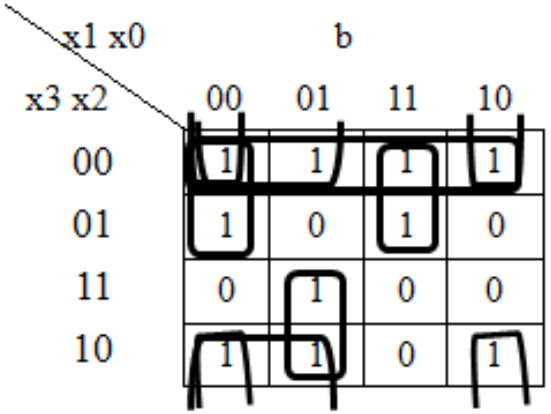
x_3	x_2	x_1	x_0	g	f	e	d	c	b	a
0	0	0	0	0	1	1	1	1	1	1
0	0	0	1	0	0	0	0	1	1	0
0	0	1	0	1	0	1	1	0	1	1
0	0	1	1	1	0	0	1	1	1	1
0	1	0	0	1	1	0	0	1	1	0
0	1	0	1	1	1	0	1	1	0	1
0	1	1	0	1	1	1	1	1	0	1
0	1	1	1	0	0	0	0	1	1	1
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	1	1	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	1	0	0
1	1	0	0	0	1	1	1	0	0	1
1	1	0	1	1	0	1	1	1	1	0
1	1	1	0	1	1	1	1	0	0	1
1	1	1	1	1	1	1	0	0	0	1

组合逻辑电路
--译码器

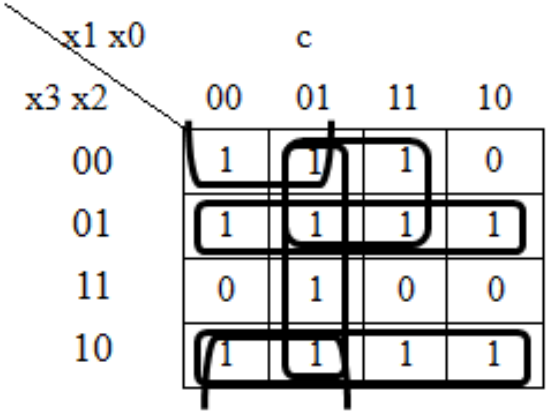
根据这个真值表的描述，使用卡诺图表示7段码和二进制码的对应关系，并进行化简。



(a) a 输出的卡诺图

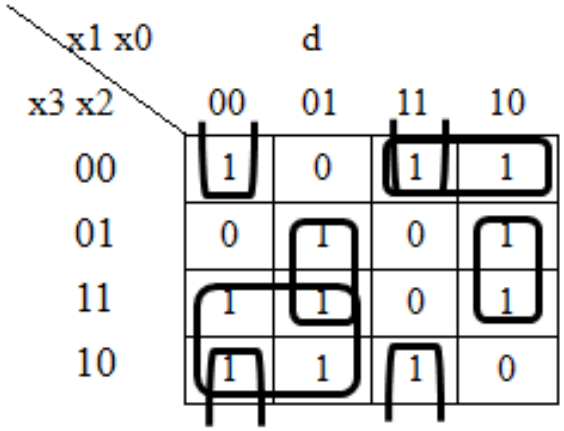


（b）b 输出的卡诺图

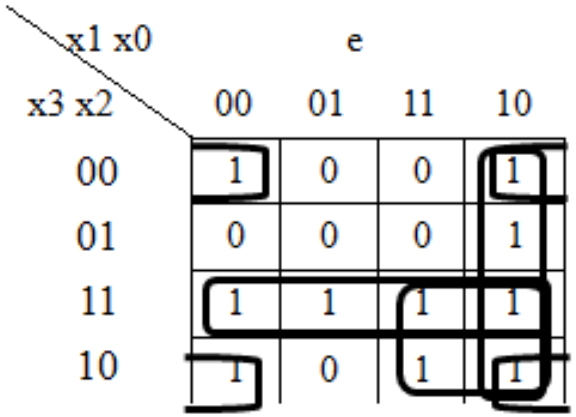


（c）c 输出的卡诺图

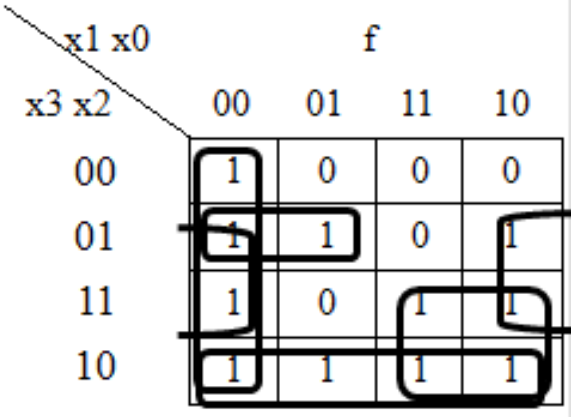
组合逻辑电路
--译码器



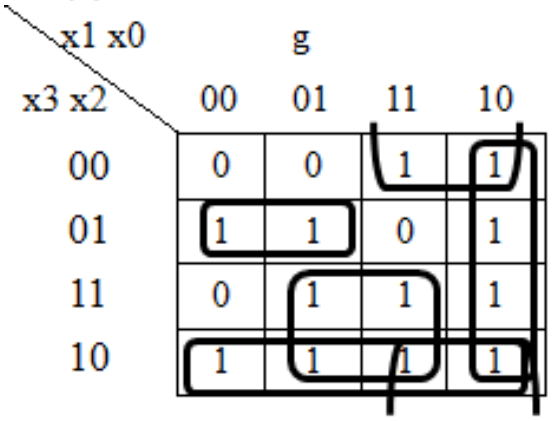
(d) d 输出的卡诺图



（e）e 输出的卡诺图



（f）f 输出的卡诺图



（g）g 输出的卡诺图

组合逻辑电路
--译码器

使用卡诺图化简后，得到a,b,c,d,e,f,g和x_3,x_2,x_1,x_0对应关系的逻辑表达式为：

$$a = \bar{x}_2\,\bar{x}_0 + x_3\,\bar{x}_2\,\bar{x}_1 + x_3\,\bar{x}_0 + x_2x_1 + \bar{x}_3\,x_1 + \bar{x}_3\,x_2x_0$$

$$b = \bar{x}_3\,\bar{x}_2 + \bar{x}_2\,\bar{x}_1 + \bar{x}_2\,\bar{x}_0 + \bar{x}_3\,x_1x_0 + x_3\,\bar{x}_1\,\bar{x}_0 + \bar{x}_3\,\bar{x}_1\,\bar{x}_0$$

$$c = \bar{x}_2\,\bar{x}_1 + x_3\,\bar{x}_2 + \bar{x}_1\,x_0 + \bar{x}_3\,x_2 + \bar{x}_3\,x_0$$

$$d = \bar{x}_2\,\bar{x}_1\,\bar{x}_0 + x_3\,\bar{x}_1 + \bar{x}_2\,x_1x_0 + x_2\,\bar{x}_1\,x_0 + \bar{x}_3\,\bar{x}_2\,x_1 + x_2x_1\,\bar{x}_0$$

$$e = \bar{x}_2\,\bar{x}_0 + x_1\,\bar{x}_0 + x_3x_2 + x_3x_1$$

$$f = \bar{x}_1\,\bar{x}_0 + x_3\,\bar{x}_2 + x_3x_1 + \bar{x}_3\,x_2\,\bar{x}_1 + \bar{x}_0\,x_2$$

$$g = \bar{x}_2\,x_1 + x_1\,\bar{x}_0 + x_3\,\bar{x}_2 + x_3x_0 + \bar{x}_3\,x_2\,\bar{x}_1$$

组合逻辑电路
--码转换器

格雷码转换器

Gray（格雷码）的编码特点是相邻的两个编码中，只有一位不同。

二进制码对应的Gray码真值表映射关系

x_3	x_2	x_1	x_0	g3	g2	g1	g0	x_3	x_2	x_1	x_0	g3	g2	g1	g0
0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0
0	0	0	1	0	0	0	1	1	0	0	1	1	1	0	1
0	0	1	0	0	0	1	1	1	0	1	0	1	1	1	1
0	0	1	1	0	0	1	0	1	0	1	1	1	1	1	0
0	1	0	0	0	1	1	0	1	1	0	0	1	0	1	0
0	1	0	1	0	1	1	1	1	1	0	1	1	0	1	1
0	1	1	0	0	1	0	1	1	1	1	0	1	0	0	1
0	1	1	1	0	1	0	0	1	1	1	1	1	0	0	0

组合逻辑电路
--码转换器

对上面的真值表进行化简，得到二进制码到gray码转换的逻辑表达式：

$$g_i = x_{i+1} \oplus x_i$$

组合逻辑电路
--二进制数到BCD码转换器

二进制数					二进制码十进制数(BCD)					
十六进制	b3	b2	b1	b0	p4	p3	p2	p1	p0	BCD
0	0	0	0	0	0	0	0	0	0	00
1	0	0	0	1	0	0	0	0	1	01
2	0	0	1	0	0	0	0	1	0	02
3	0	0	1	1	0	0	0	1	1	03
4	0	1	0	0	0	0	1	0	0	04
5	0	1	0	1	0	0	1	0	1	05
6	0	1	1	0	0	0	1	1	0	06
7	0	1	1	1	0	0	1	1	1	07
8	1	0	0	0	0	1	0	0	0	08
9	1	0	0	1	0	1	0	0	1	09
A	1	0	1	0	1	0	0	0	0	10
B	1	0	1	1	1	0	0	0	1	11
C	1	1	0	0	1	0	0	1	0	12
D	1	1	0	1	1	0	0	1	1	13
E	1	1	1	0	1	0	1	0	0	14
F	1	1	1	1	1	0	1	0	1	15

组合逻辑电路
--二进制数到BCD码转换器

b3 b2 \ b1 b0	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	0	1	1	0
10	0	1	1	0

(a) p0 的卡诺图

b3 b2 \ b1 b0	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	1	1	0	0
10	0	0	0	0

(b)p1 的卡诺图

b3 b2 \ b1 b0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	1	1
10	0	0	0	0

(c)p2 的卡诺图

b3 b2 \ b1 b0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	0	0
10	1	1	0	0

b3 b2 \ b1 b0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	0	1	1

组合逻辑电路
--二进制数到BCD码转换器

$$p_0 = b_0$$

$$p_1 = b_3 b_2 \overline{b_1} + \overline{b_3} b_1$$

$$p_2 = \overline{b_3} b_2 + b_2 b_1$$

$$p_3 = b_3 \overline{b_2} \overline{b_1}$$

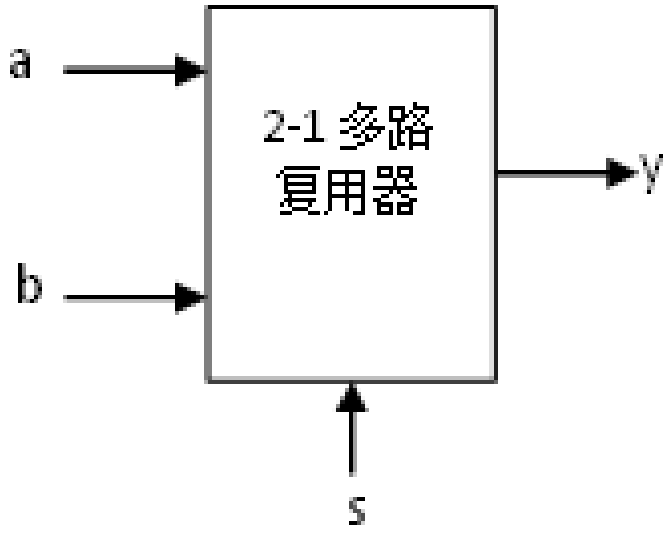
$$p_4 = b_3 b_2 + b_3 b_1$$

组合逻辑电路
--数据选择器

在数字逻辑中，数据选择器（multiplexer，MUX），也称为多路复用器。它从多个输入的逻辑信号中选择一个逻辑信号输出。

组合逻辑电路
--数据选择器

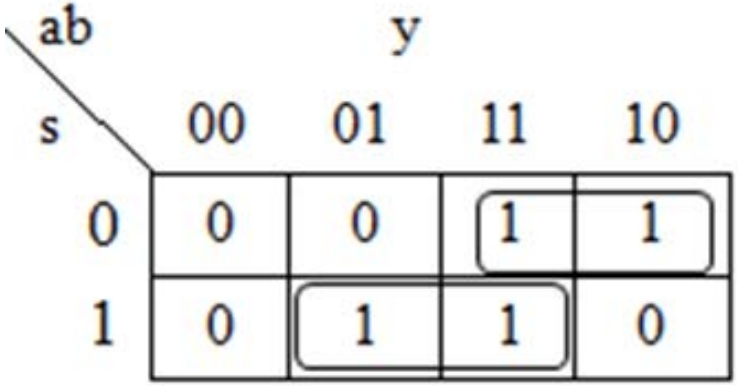
2:1多路复用器



2-1多路复用器真值表

s	a	b	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

组合逻辑电路
--数据选择器



s \ ab	00	01	11	10
0	0	0	1	1
1	0	1	1	0

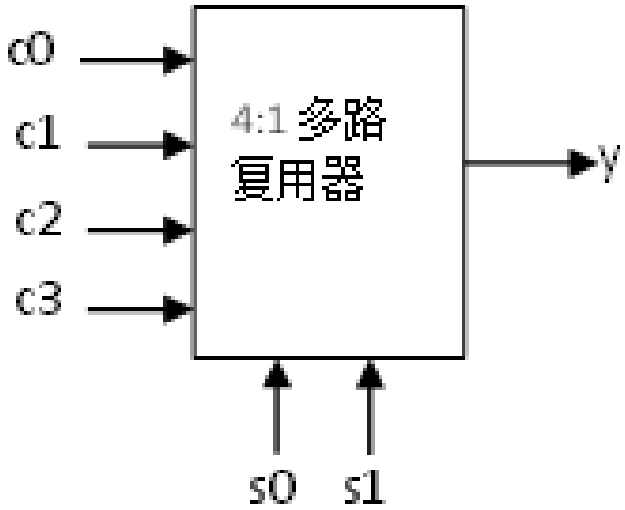
y

多路复用器的最简逻辑表达式：

$$y = a \cdot \bar{s} + b \cdot s$$

组合逻辑电路
--数据选择器

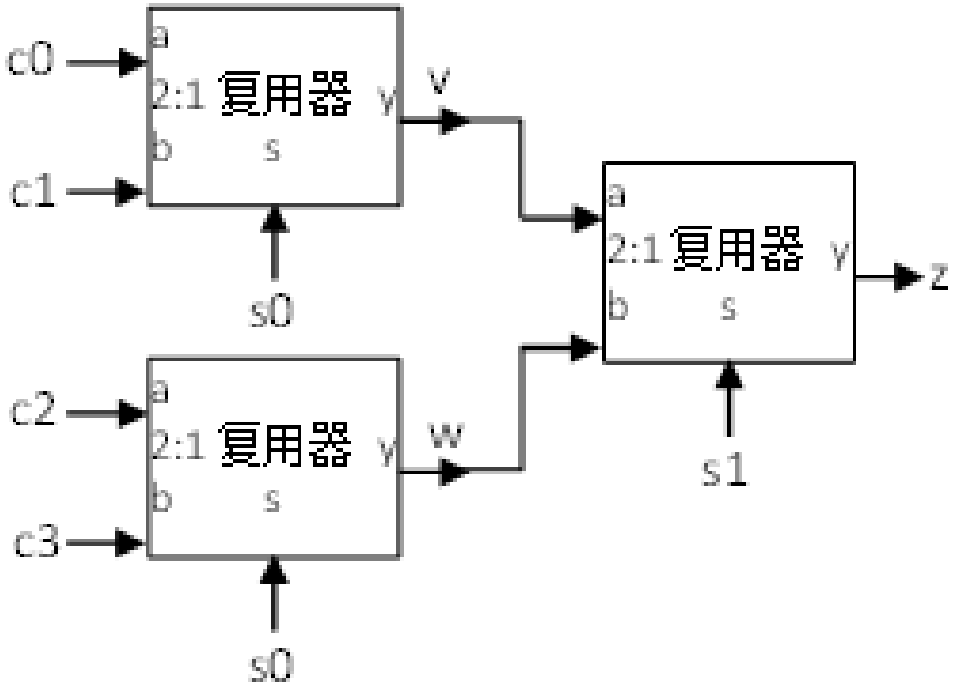
4:1 多路复用器的实现原理



4-1多路复用器真值表

s1	s0	y
0	0	c0
0	1	c1
1	0	c2
1	1	c3

组合逻辑电路
--数据选择器



$$v = c_0 \cdot \bar{s}_0 + c_1 \cdot s_0$$

$$w = c_2 \cdot \bar{s}_0 + c_3 \cdot s_0$$

$$z = v \cdot \bar{s}_1 + w \cdot s_1$$

$$z = (c_0 \cdot \bar{s}_0 + c_1 \cdot s_0) \cdot \bar{s}_1 + (c_2 \cdot \bar{s}_0 + c_3 \cdot s_0) \cdot s_1$$

$$= c_0 \cdot \bar{s}_0 \cdot \bar{s}_1 + c_1 \cdot s_0 \cdot \bar{s}_1 + c_2 \cdot \bar{s}_0 \cdot s_1 + c_3 \cdot s_0 \cdot s_1$$

组合逻辑电路
--数据选择器

典型的数据选择器专用集成电路

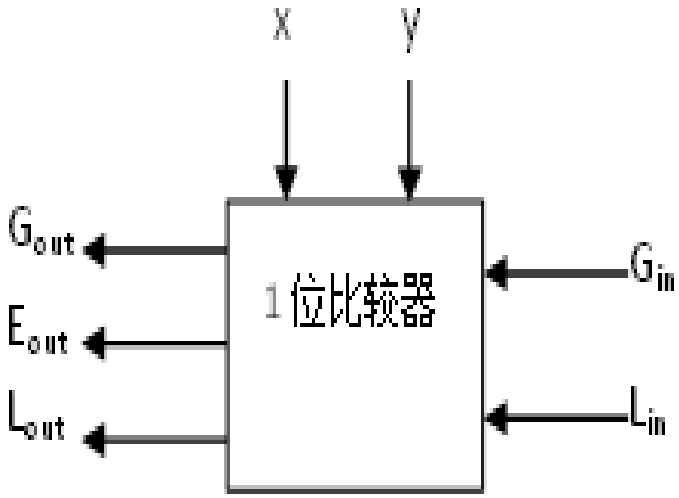
数据选择器	功能	输出状态
74LS157	四个2选1数据选择器	输出原变量
74LS158	四个2选1数据选择器	输出反变量
74LS153	两个4选1数据选择器	输出原变量
74LS352	两个4选1数据选择器	输出反变量
74LS151	一个8选1数据选择器	输出反变量
74LS150	一个16选1数据选择器	输出反变量

组合逻辑电路
--数据比较器

多位数字比较器的实现通过两步实现：

- 首先实现1位的数字比较器。
- 然后通过级联1位的数字比较器，实现多位的数字比较器。

组合逻辑电路
--数据比较器



比较器的功能满足下面的条件：

- 如果x>y，或者x=y且G_{in}=1，则G_{out}=1。
- 如果x=y且G_{in}=0,L_{in}=0，则E_{out}=1。
- 如果x<y，或者x=y且L_{in}=1，则L_{out}=1。

组合逻辑电路
--数据比较器

一位比较器的真值表描述

x	y	G_{in}	L_{in}	G_{out}	E_{out}	L_{out}
0	0	0	0	0	1	**0**
0	0	0	1	0	0	**1**
0	0	1	0	1	0	**0**
0	0	1	1	1	0	**1**
0	1	0	0	0	0	**1**
0	1	0	1	0	0	**1**
0	1	1	0	0	0	**1**
0	**1**	**1**	**1**	**0**	**0**	**1**

x	y	G_{in}	L_{in}	G_{out}	E_{out}	L_{out}
1	**0**	**0**	**0**	**1**	**0**	**0**
1	0	0	1	1	0	**0**
1	0	1	0	1	0	**0**
1	0	1	1	1	0	**0**
1	1	0	0	0	1	**0**
1	1	0	1	0	0	**1**
1	1	1	0	1	0	**0**
1	1	1	1	1	0	**1**

组合逻辑电路
--数据比较器

G_{in} L_{in} / xy	00	01	11	10
00	0	0	1	1
01	0	0	0	0
11	0	0	1	1
10	1	1	1	1

(a) G_{out} 卡诺图映射

G_{in} L_{in} / xy	00	01	11	10
00	1	0	0	0
01	0	0	0	0
11	1	0	0	0
10	0	0	0	0

(b) E_{out} 卡诺图映射

G_{in} L_{in} / xy	00	01	11	10
00	0	1	1	0
01	1	1	1	1
11	0	1	1	0
10	0	0	0	0

(c) L_{out} 卡诺图映射

$$G_{out} = x \cdot \bar{y} + x \cdot G_{in} + \bar{y} \cdot G_{in}$$

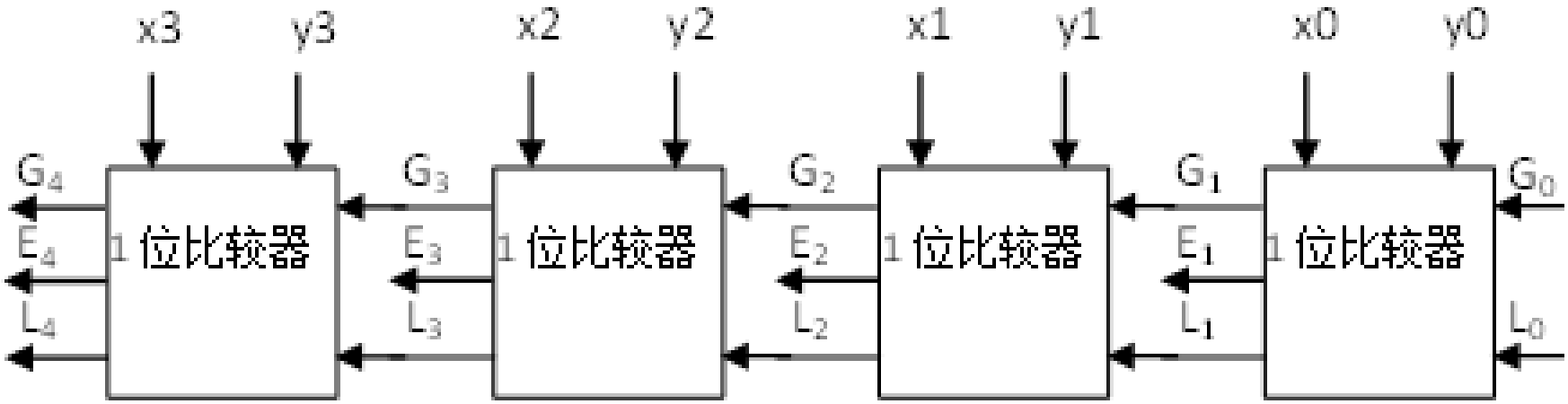
$$E_{out} = \bar{x} \cdot \bar{y} \cdot \bar{G}_{in} \cdot \bar{L}_{in} + x \cdot y \cdot \bar{G}_{in} \cdot \bar{L}_{in}$$

$$L_{out} = \bar{x} \cdot y + \bar{x} \cdot L_{in} + y \cdot L_{in}$$

组合逻辑电路
--数据比较器

多位比较器的实现原理

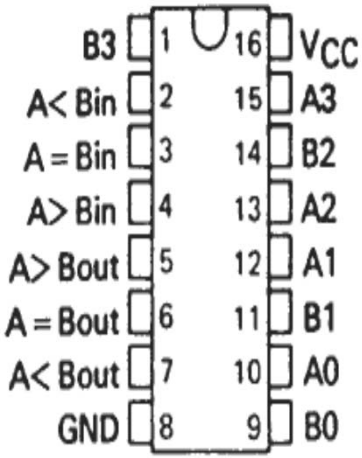
- 多位比较器可以由一位比较器级联得到。



组合逻辑电路
--数据比较器

74LS85是一个专用的4位比较器芯片。

比较输入				级连输入			输出		
A3，B3	A2，B2	A1，B1	A0，B0	A>B	A<B	A=B	A>B	A<B	A=B
A3>B3	×	×	×	×	×	×	H	L	L
A3<B3	×	×	×	×	×	×	L	H	L
A3=B3	A2>B2	×	×	×	×	×	H	L	L
A3=B3	A2<B2	×	×	×	×	×	L	H	L
A3=B3	A2=B2	A1>B1	×	×	×	×	H	L	L
A3=B3	A2=B2	A<B1	×	×	×	×	L	H	L
A3=B3	A2=B2	A1=B1	A0>B0	×	×	×	H	L	L
A3=B3	A2=B2	A1=B1	A0<B0	×	×	×	L	H	L
A3=B3	A2=B2	A1=B1	A0=B0	H	L	L	H	L	L
A3=B3	A2=B2	A1=B1	A0=B0	L	H	L	L	H	L
A3=B3	A2=B2	A1=B1	A0=B0	×	×	H	L	L	H
A3=B3	A2=B2	A1=B1	A0=B0	H	H	L	L	L	L
A3=B3	A2=B2	A1=B1	A0=B0	L	L	L	H	H	L



组合逻辑电路
--加法器

在电子学中，加法器（adder）是一种用于执行加法运算的数字电路部件，是构成中央处理单元(CPU)中算术逻辑单元的基础。

- 在中央处理单元中，加法器主要负责计算地址、索引等数据。
- 加法器也是其他一些硬件，例如二进制数乘法器的重要组成部分。

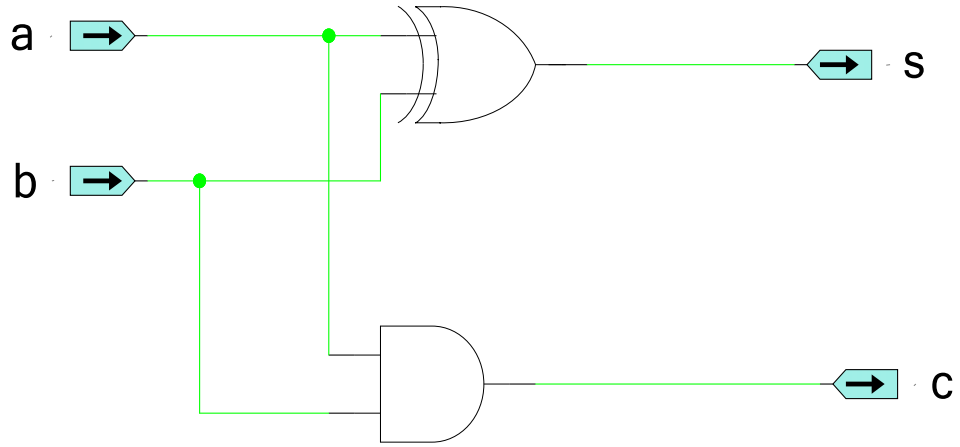
组合逻辑电路
--加法器

一位半加器的实现

根据真值表可以得到半加器的最简逻辑表达式：

半加器真值表

a_0	b_0	s_0	c_1
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



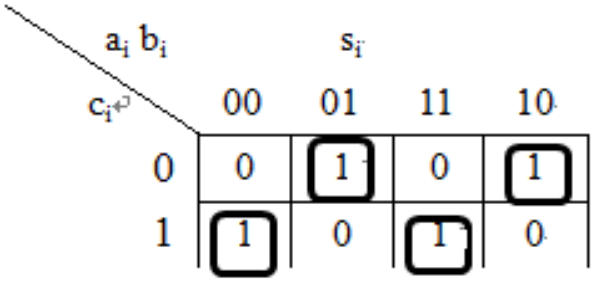
$$s_0 = \bar{a}_0 \cdot b_0 + a_0 \cdot \bar{b}_0 = a_0 \oplus b_0 \qquad c_1 = a_0 \cdot b_0$$

组合逻辑电路
--加法器

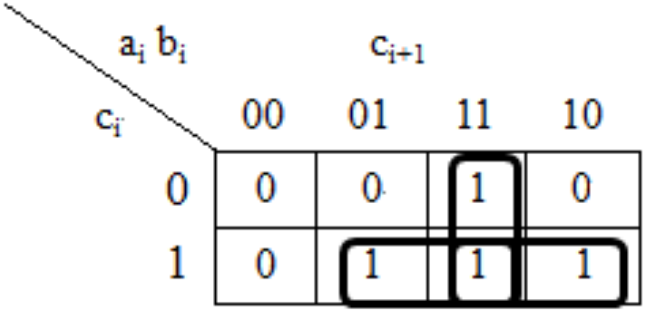
一位全加器的实现

全加器真值表

c_i	a_i	b_i	s_i	c_{i+1}
0	0	0	0	**0**
0	0	1	1	**0**
0	1	0	1	**0**
0	1	1	0	**1**
1	0	0	1	**0**
1	0	1	0	**1**
1	1	0	0	**1**
1	1	1	1	**1**



(a) S_i 卡洛图映射



(b) C_{i+1} 卡诺图映射

组合逻辑电路
--加法器

可以得到全加器的最简逻辑表达式：

$$s_i = \bar{a}_i \cdot b_i \cdot \bar{c}_i + a_i \cdot \bar{b}_i \cdot \bar{c}_i + \bar{a}_i \cdot \bar{b}_i \cdot c_i + a_i \cdot b_i \cdot c_i$$

$$= \bar{c}_i \cdot (a_i \oplus b_i) + c_i \cdot \overline{(a_i \oplus b_i)}$$

$$= (a_i \oplus b_i \oplus c_i)$$

$$c_{i+1} = a_i \cdot b_i + b_i \cdot c_i + a_i \cdot c_i$$

$$= a_i \cdot b_i + c_i \cdot (a_i + b_i)$$

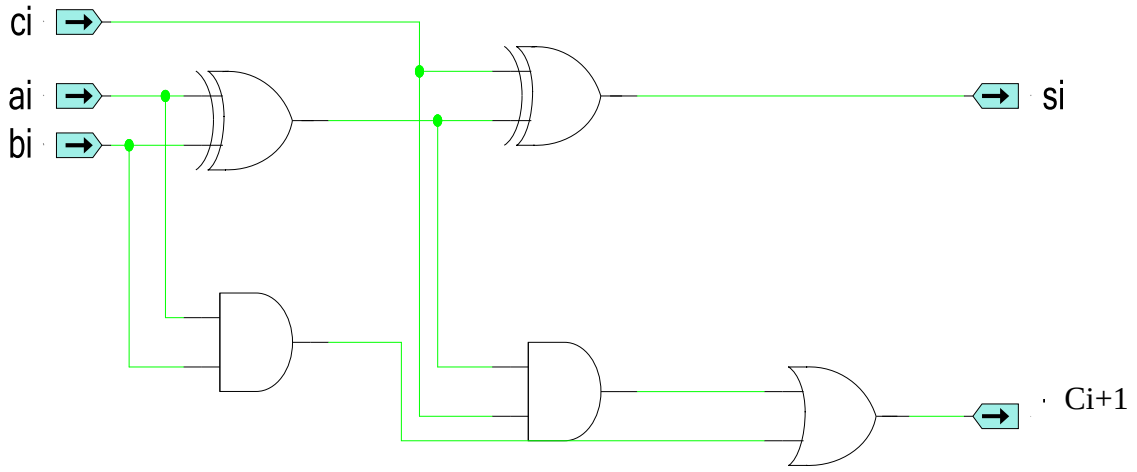
$$= a_i \cdot b_i + c_i \cdot (a_i \cdot (b_i + \bar{b}_i) + b_i \cdot (a_i + \bar{a}_i))$$

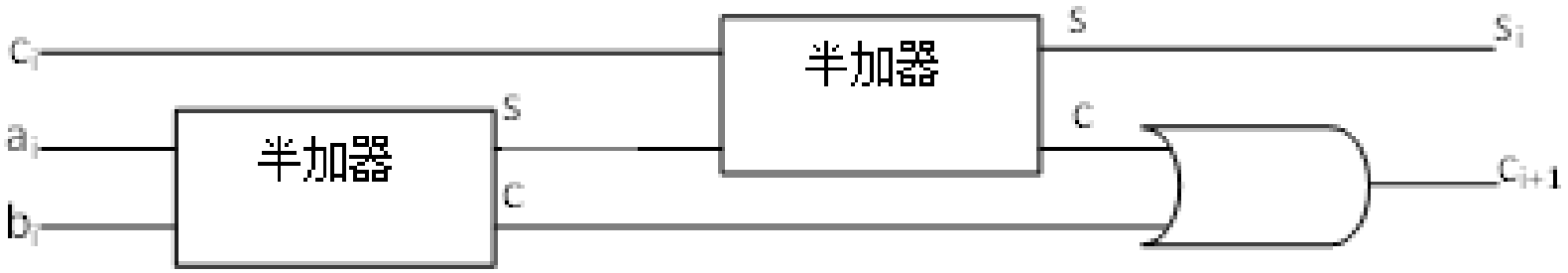
$$= a_i \cdot b_i + c_i \cdot (a_i \cdot \bar{b}_i + b_i \cdot \bar{a}_i)$$

$$= a_i \cdot b_i + c_i \cdot (a_i \oplus b_i)$$

组合逻辑电路
--加法器

根据最简的逻辑表达式，可以得到全加器逻辑图

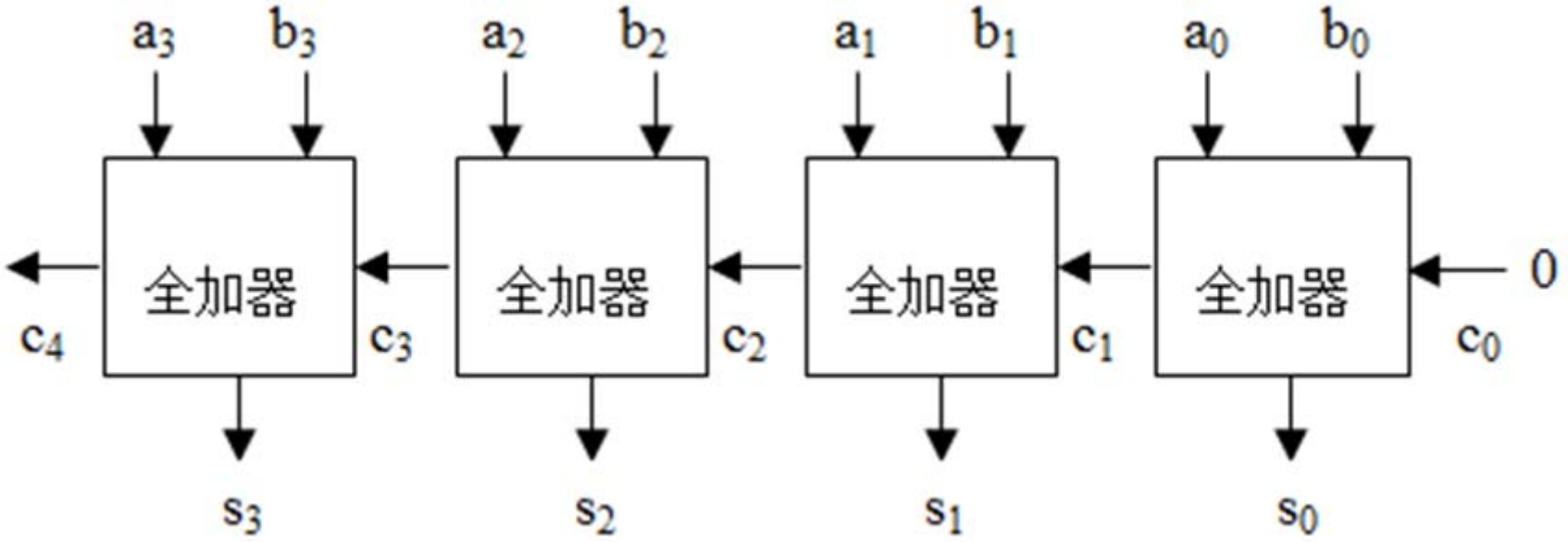




组合逻辑电路
--多位全加器的实现

串行进位加法器

- 多位全加器可以由一位全加器级联而成。
- 前一级全加器的进位做为下一级进位的输入。
- 串行进位加法器需要一级一级的进位，有很大的进位延迟。



组合逻辑电路
--多位全加器的实现

超前进位加法器　为了减少多位二进制数加减计算所需的时间，出现了一种比串行进位加法器速度更快的加法器，这种加法器称为超前进位加法器。

- 假设二进制加法器的第i为输入为ai和bi，输出为si，进位输入为ci，进位输出为ci+1。则有：

$$s_i = a_i \oplus b_i \oplus c_i$$

$$c_{i+1} = a_i \cdot b_i + a_i \cdot c_i + b_i \cdot c_i = a_i \cdot b_i + c_i \cdot (a_i + b_i)$$

组合逻辑电路
--多位全加器的实现

令：$g_i = a_i \cdot b_i$, $p_i = a_i + b_i$

则：$c_{i+1} = g_i + p_i \cdot c_i$

- 当a_i和b_i都为1的时候，g_i=1，产生进位c_{i+1}=1
- 当a_i或b_i为1的时候，p_i=1，传递进位c_{i+1}=c_i
- g_i定义为进位产生信号，p_i定义为进位传递信号。
- g_i的优先级高于p_i，也就是说：
  - 当g_i=1的时候，必然存在p_i=1，不管c_i多少，必然存在进位。
  - 当g_i=0，而p_i=1时，进位输出为c_i，跟c_i之前的逻辑有关。

组合逻辑电路
--多位全加器的实现

以4位超前进位加法器为例

- 假设四位被加数和加数为a和b，进位输入为c_{in}，进位输出为c_{out}
- 对于第i位产生的进位：
  - $g_i = a_i \cdot b_i$，$p_i = a_i + b_i$（i=0，1，2，3）

$$c_0 = c_{in}$$
$$c_1 = g_0 + p_0 \cdot c_0$$
$$c_2 = g_1 + p_1 \cdot c_1 = g_1 + p_1 \cdot (g_0 + p_0 \cdot c_0) = g_1 + p_1 \cdot g_0 + p_1 \cdot p_0 \cdot c_0$$
$$c_3 = g_2 + p_2 \cdot c_2 = g_2 + p_2 \cdot g_1 + p_2 \cdot p_1 \cdot g_0 + p_2 \cdot p_1 \cdot p_0 \cdot c_0$$

$$c_4 = g_3 + p_3 \cdot c_3 = g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0 + p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot c_0$$
$$c_{out} = c_4$$

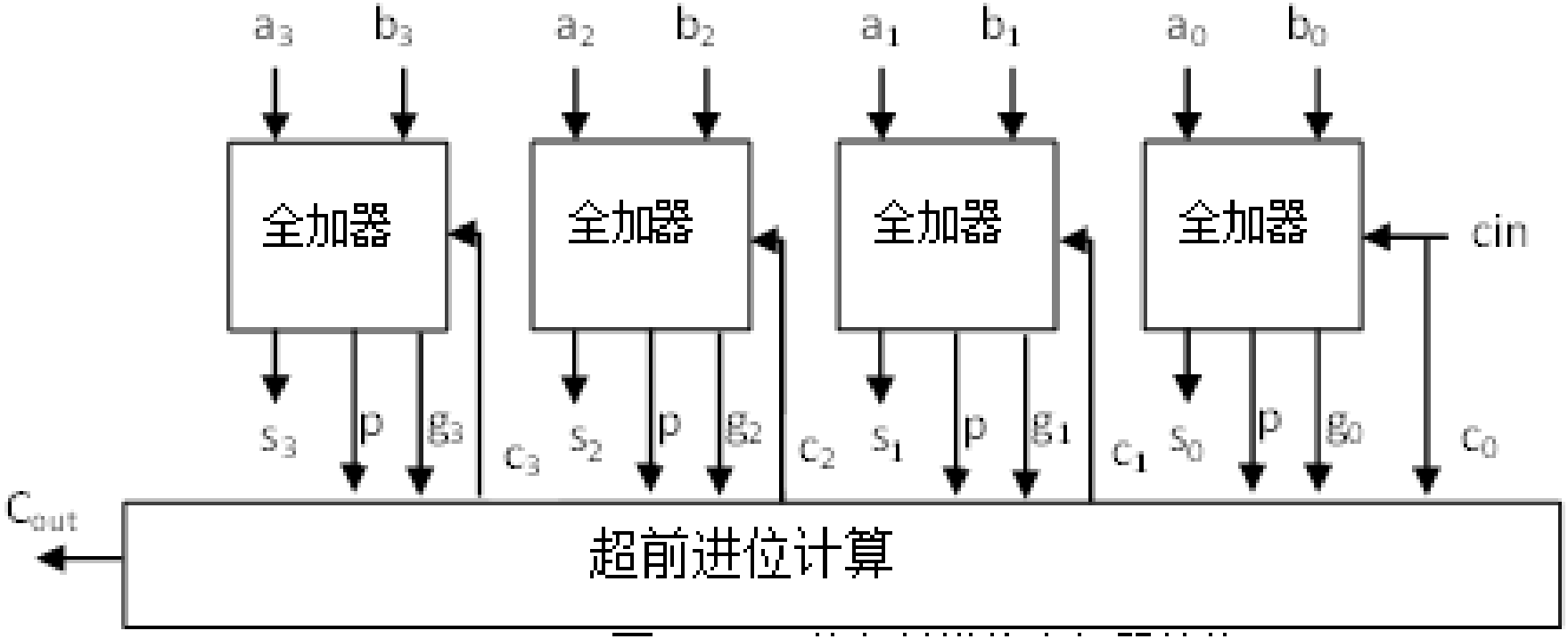
组合逻辑电路
--多位全加器的实现

可以看出，各级的进位彼此独立，只与输入数据和Cin有关，将各级之间的进位级联传播给去掉了。因此，大大降低了进位产生的延迟。

- 每个等式与只有三级延迟的电路对应，第一级延迟对应进位产生信号和进位传递信号，后两级延迟对应上面的积之和。
- 同时，可以得到第i为的和为：

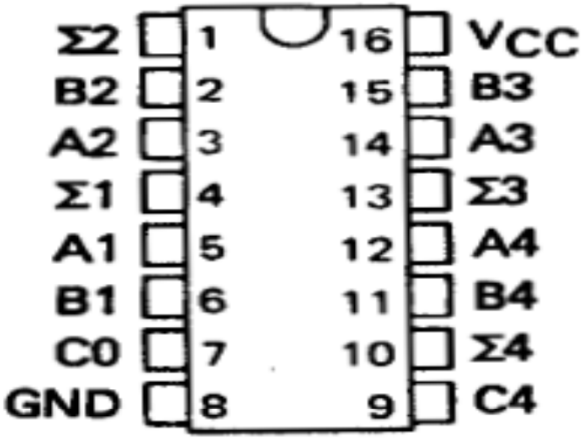
$$s_i = a_i \oplus b_i \oplus c_i = g_i \oplus p_i \oplus c_i$$

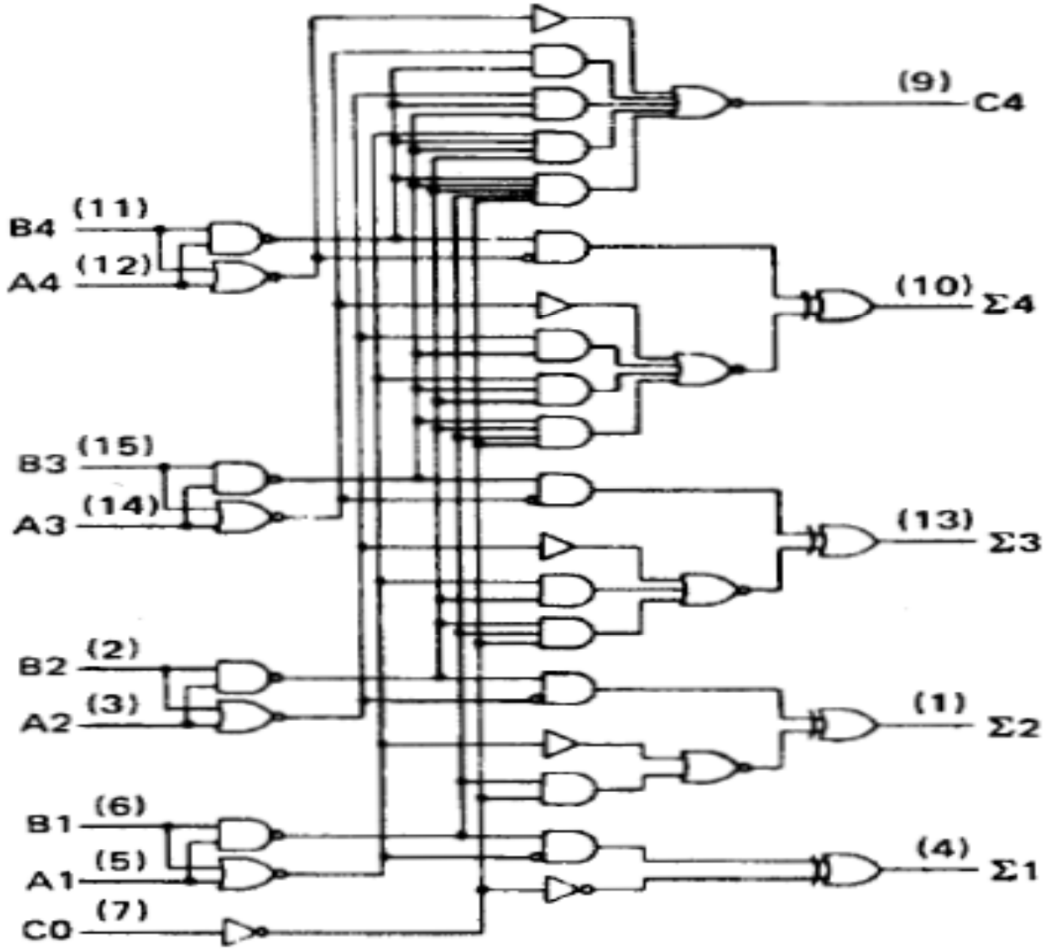
组合逻辑电路
--多位全加器的实现



组合逻辑电路
--多位全加器的实现

74LS283超前进位加法器





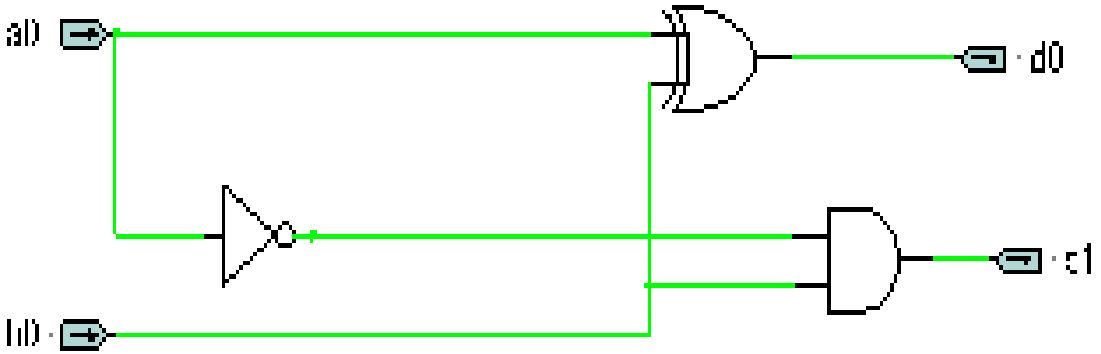
组合逻辑电路
--减法器

一位半减器的实现原理

减法器的设计很类似加法器的设计过程，可以通过真值表实现。一旦设计了一个位片减法器电路，就可以将其复制N次，创建一个N位的减法器。

半加器真值表

a_0	b_0	d_0	c_1
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



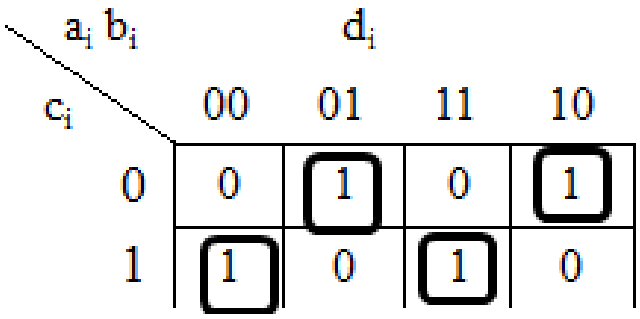
$$d_0 = \bar{a}_0 \cdot b_0 + a_0 \cdot \bar{b}_0 = a_0 \oplus b_0$$

$$c_1 = \bar{a}_0 \cdot b_0$$

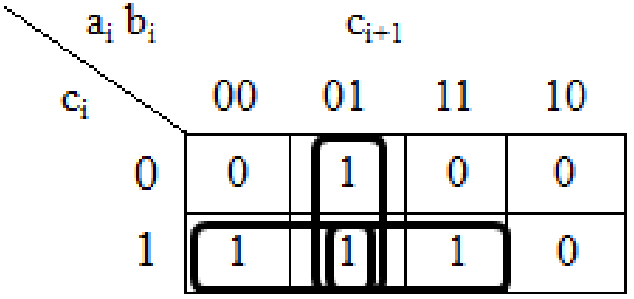
组合逻辑电路
--减法器

一位全减器的实现原理

c_i	a_i	b_i	d_i	c_{i+1}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1



(a) d_i 卡诺图映射



(b) c_{i+1} 卡诺图映射

组合逻辑电路
--减法器

$$d_i = \bar{a}_i \cdot b_i \cdot \bar{c}_i + a_i \cdot \bar{b}_i \cdot \bar{c}_i + \bar{a}_i \cdot \bar{b}_i \cdot c_i + a_i \cdot b_i \cdot c_i$$

$$= \bar{c}_i \cdot (a_i \oplus b_i) + c_i \cdot \overline{(a_i \oplus b_i)}$$

$$= (a_i \oplus b_i \oplus c_i)$$

$$c_{i+1} = c_i \cdot b_i + \bar{a}_i \cdot b_i + \bar{a}_i \cdot c_i$$

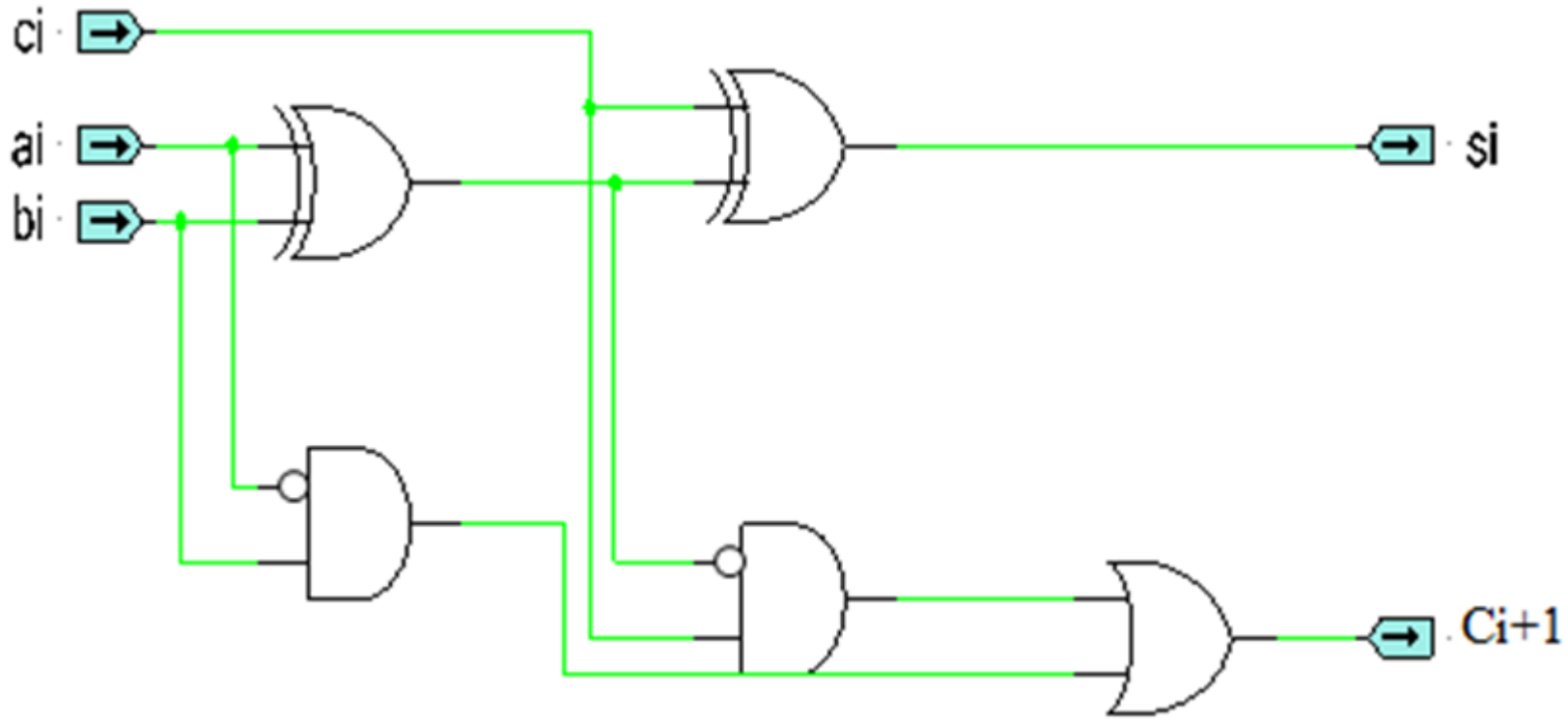
$$= \bar{a}_i \cdot b_i + c_i \cdot (\bar{a}_i + b_i)$$

$$= \bar{a}_i \cdot b_i + c_i \cdot (\bar{a}_i \cdot (b_i + \bar{b}_i) + b_i \cdot (a_i + \bar{a}_i))$$

$$= \bar{a}_i \cdot b_i + c_i \cdot (\bar{a}_i \cdot \bar{b}_i + b_i \cdot a_i)$$

$$= \bar{a}_i \cdot b_i + c_i \cdot (a_i \oplus b_i)$$

组合逻辑电路
--减法器



组合逻辑电路
--负数的表示

在数字电路中，用于执行算术功能的部件经常需要处理负数。所以，必须要定义一种表示负数的方法。

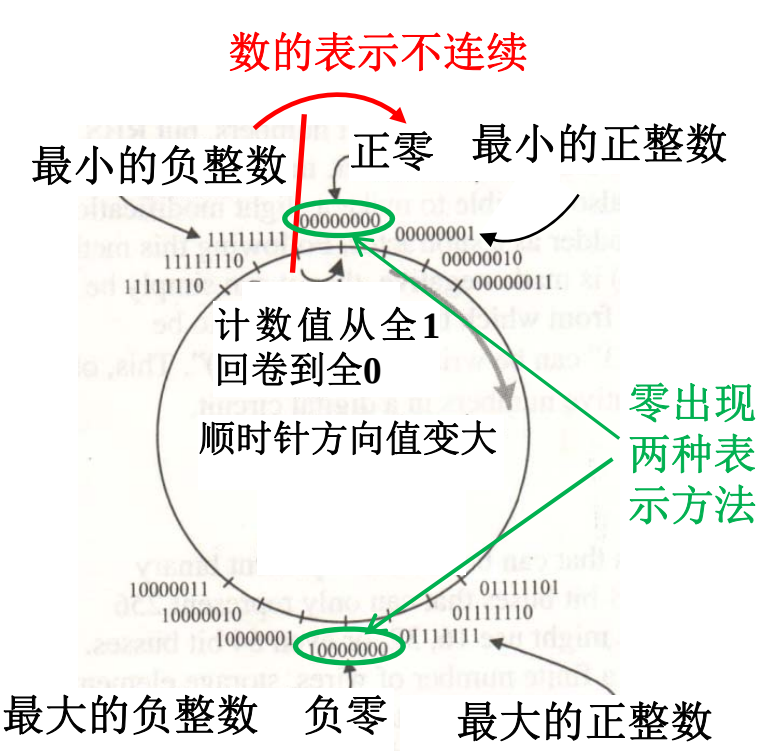
- 一个N位的系统总共可以表示2^N个数，一个有用的编码就是使用一半可用的编码$(2^N/2)$表示正数，另一半表示负数。
  - 可以将一个比特位设计成一个符号位，用于区分正数和负数。如果符号位是‘1’，则所表示的数为负数；否则，所表示的数为正数。
  - 最高有效位（most significant bit，MSB）作为符号位。如果该位为0，表示是一个正数，则配置数的幅度时，可以将其忽略。

组合逻辑电路
--负数的表示

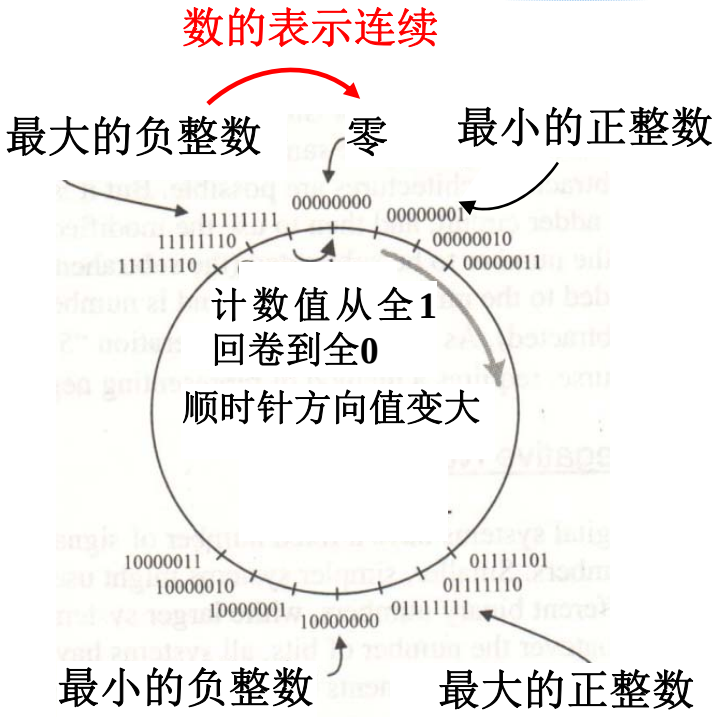
在所有可能的负数编码方案中，经常使用两种：符号幅度和二进制补码。

- 符号幅度表示法
  - 在一个八位的符号幅度系统中，$(16)_{10}$表示为$(00010000)_2$
  - $(-16)_{10}$表示为$(10010000)_2$。
- 二进制补码表示法
  - 在二进制补码表示法中，MSB仍是符号位。MSB为1时表示负数，MSB为0时表示正数。
  - 在二进制补码表示法中，整数0由一个字长为N位的全零二进制数表示。其余的$2^N - 1$个数表示正整数和负整数。

组合逻辑电路
--负数的表示



符号幅度表示法



补码表示法

组合逻辑电路
--负数的表示

+17转换为-17的二进制补码计算公式：

- +17对应的原码为00010001。
- 所有位按位取反后变成11101110。
- 结果加1，得到-17的补码为11101111。

原码按位取反，加1就得到负数的补码

组合逻辑电路
--负数的表示

-35转换为+35的二进制补码计算公式：

- **-35对应的补码为11011101。**
- **所有位按位取反后变成00100010。**
- **结果加1，得到+35的补码为00100011。**

组合逻辑电路
--负数的表示

-127转换为+127的二进制补码计算公式：

- -127对应的补码为10000001。
- 所有位按位取反后变成01111110。
- 结果加1，得到+127的补码为01111111。

组合逻辑电路
--负数的表示

+1转换为-1的二进制补码计算公式：

- +1对应的原码为00000001。
- 所有位按位取反后变成11111110。
- 结果加1，得到-1的补码为11111111。

组合逻辑电路
--加法器/减法器

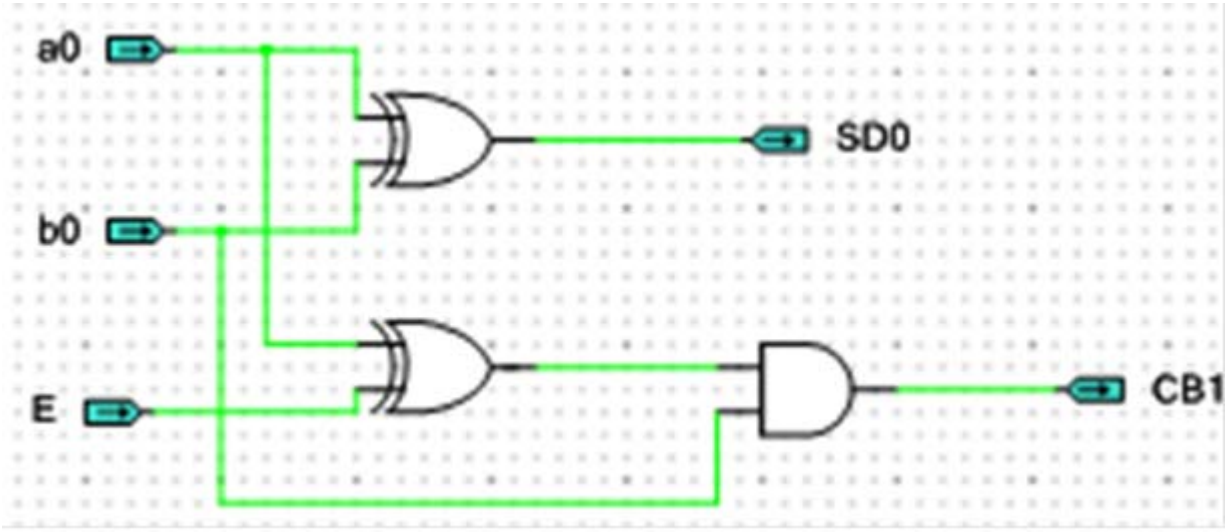
一位加法器/减法器的实现原理

- 假设一个逻辑变量E，用来选择是半加器还是半加器功能，规定：
  - E=0时，为半加器；
  - E=1时，为半减器。
- 二者之间的差别可以用下面的逻辑关系式表示：

$$\bar{E} \cdot a + E \cdot \bar{a}$$

组合逻辑电路
--加法器/减法器

半加器和半减器就可以使用一个逻辑结构实现。



其中：

- SD0为半加器/半减器的和/差结果；
- CB1为半加器/半减器的进位/借位。

组合逻辑电路
--加法器/减法器

全减器真值表

c_i	a_i	b_i	d_i	c_{i+1}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1

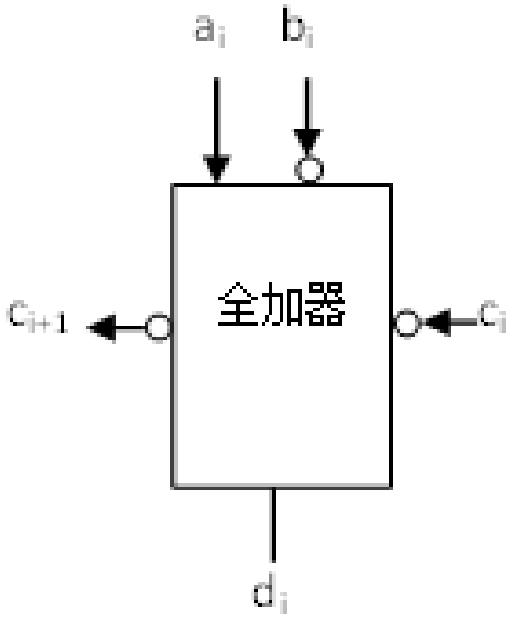
重排的全加器真值表

c_i	a_i	b_i	s_i	c_{i+1}
1	0	1	0	1
1	0	0	1	0
1	1	1	1	1
1	1	0	0	1
0	0	1	1	0
0	0	0	0	0
0	1	1	0	1
0	1	0	1	0

重要：全减器和全加器的c_i和c_{i+1}互补，b_i也互补。

组合逻辑电路
--加法器/减法器

如果对c_i和b_i进行求补，则全加器可以用作全减器使用。



组合逻辑电路
--加法器/减法器

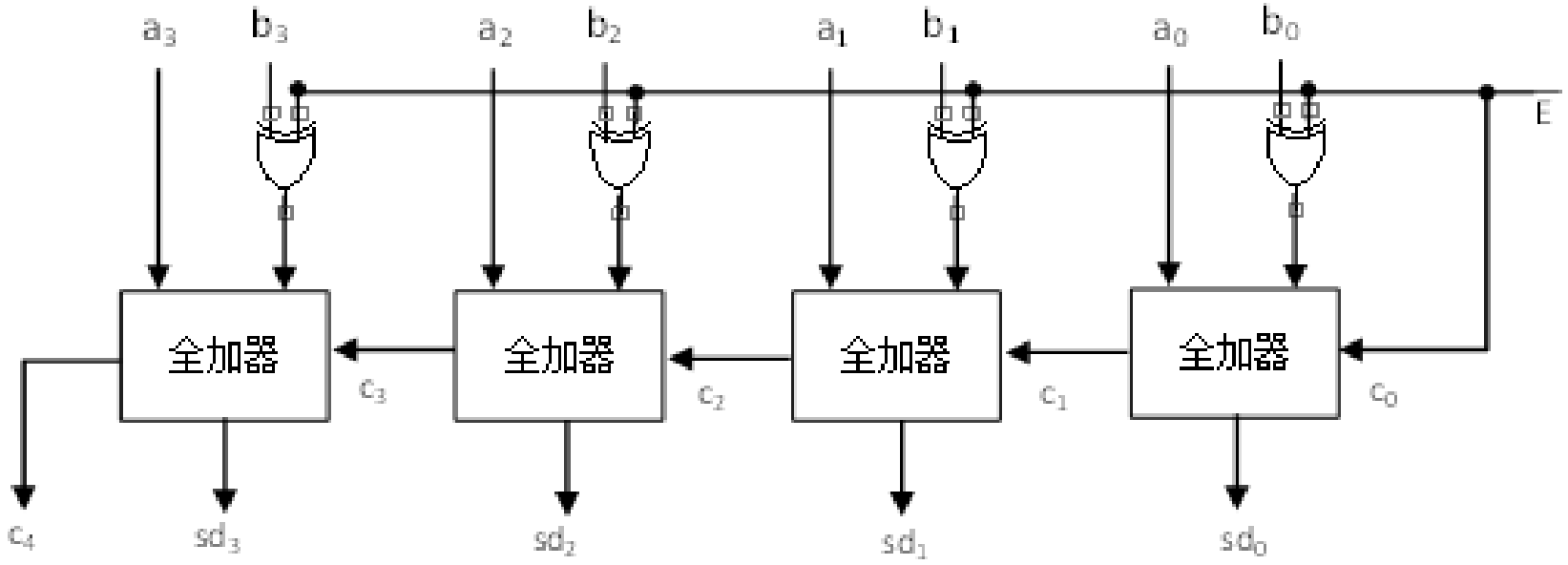
如果把上图的结构级联可以生成一个4位的减法器。

但是将取消进位输出和下一个进位输入的取补运算。

- 这是因为最终的结果只是需要对最初的进位输入c_0取补。
  - 对于加法器来说c_0为0；
  - 对于减法器来说，c_0为1。
- 等效于a和$\overline{b}$的和加1。这其实就是补码的运算。这样，使用加法器进行相减运算，只需要用减数的补码，然后相加。

组合逻辑电路
--加法器/减法器

多位加法器/减法器的实现原理



注意：当用作减法时（**E=1**），则输出进位c_4是输出借位的补。

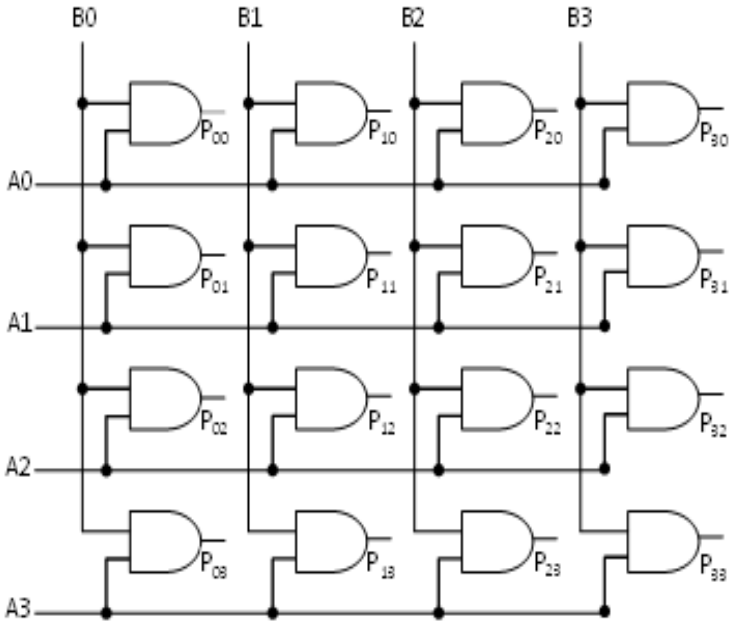
组合逻辑电路
--乘法器

二进制的乘法器是数字电路的一种元件，它可以将两个二进制数相乘。乘法器是由更基本的加法器组成的。

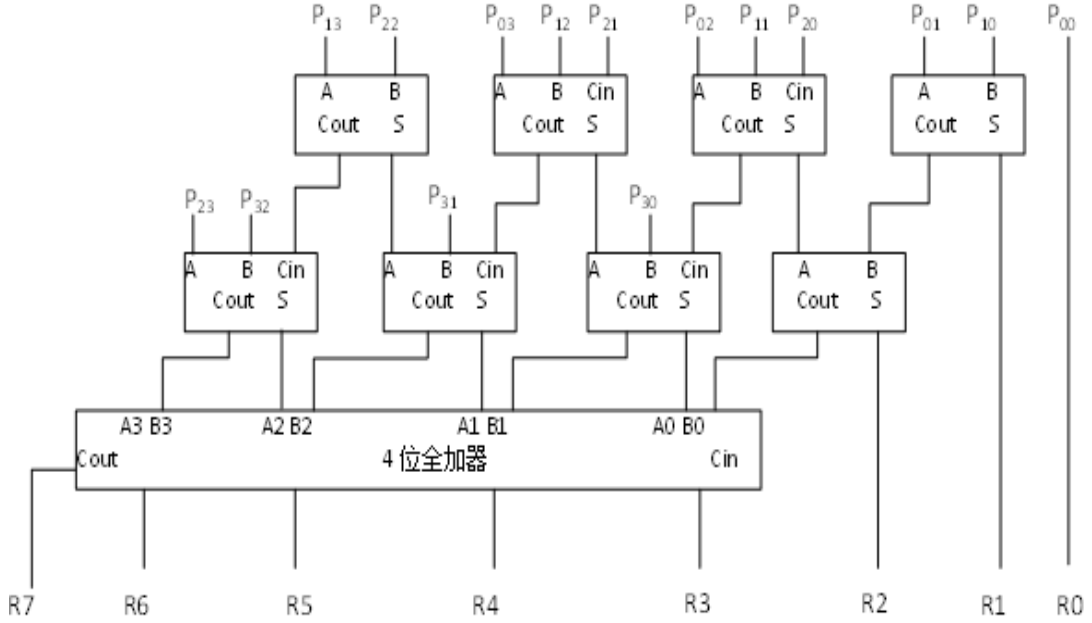
1 0 0 1	$A_3\ A_2\ A_1\ A_0$
× 1 0 1 1	× $B_3\ B_2\ B_1\ B_0$
1 0 0 1	$P_{03}\ P_{02}\ P_{01}\ P_{00}$
1 0 0 1	$P_{13}\ P_{12}\ P_{11}\ P_{10}$
0 0 0 0	$P_{23}\ P_{22}\ P_{21}\ P_{20}$
1 0 0 1	$P_{33}\ P_{32}\ P_{31}\ P_{30}$
1 1 0 0 0 1 1	$R_7\ R_6\ R_5\ R_4\ R_3\ R_2\ R_1\ R_0$

乘法运算的本质实际上是“部分乘积移位求和”的过程。

组合逻辑电路
--乘法器



(a) A×B 得到部分积



(b) 部分积相加得到结果