

---

# 数字逻辑

## 第三章 组合逻辑电路分析与设计

北京理工大学 计算机学院

张磊

[leizhang@bit.edu.cn](mailto:leizhang@bit.edu.cn)

# 本章内容

---

## 三. 算术功能模块

- 1. 迭代组合电路
- 2. 二进制加法器
- 3. 二进制减法器
- 4. 溢出
- 5. 其它功能

# 1. 迭代组合电路

---

- 设计一个电路来处理32位二进制加法
  - 输入数量 = ?
  - 真值表行数 = ?
  - 布尔方程的输入变量个数 ?
  - 布尔方程包含非常多项
- 实际中不可行
- 那怎么办?
- 基本思想：利用规律性来简化设计

---

□ 类似的算术功能有以下规律：

- 对二进制向量进行操作
- 对每一位进行同样的子函数操作

□ 设计子函数功能模块，重复使用得到总体功能

□ 单元

- 子函数模块

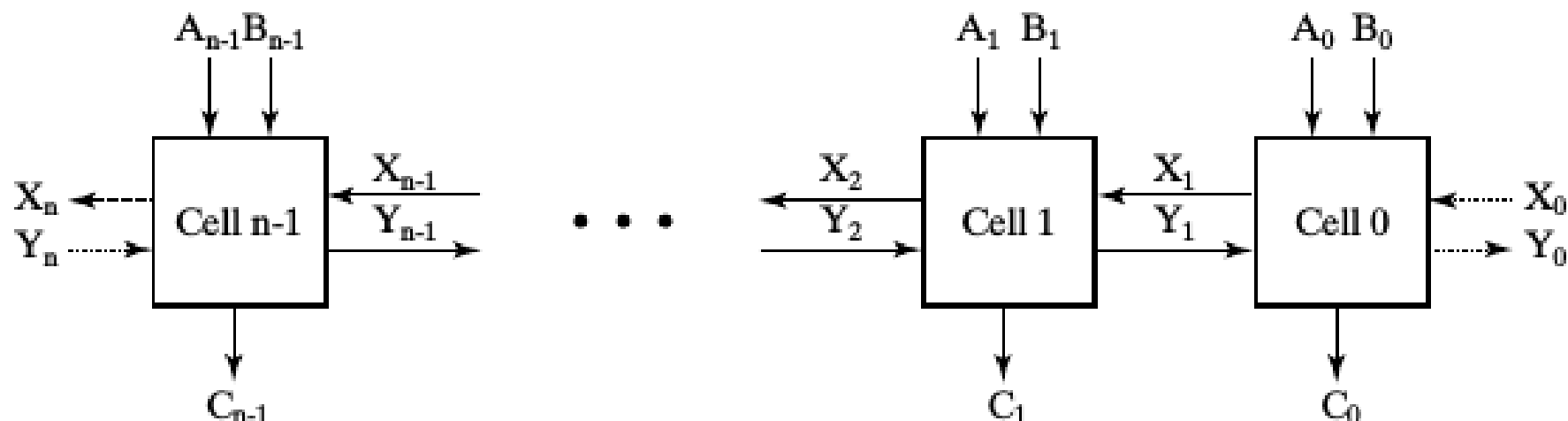
□ 迭代阵列

- 相互连接的单元的阵列

---

□ 单元

□ 迭代阵列



## 2. 二进制加法器

---

- 半加器：2输入按位加功能模块
- 全加器：3输入按位加功能模块
- 行波进位加法器：二进制加法迭代阵列

---

□ 半加器

□ 输入：X, Y

□ 输出：和位S, 进位C

<b>X</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>1</b>
<b>+ Y</b>	<b>+ 0</b>	<b>+ 1</b>	<b>+ 0</b>	<b>+ 1</b>
<hr/>	<hr/>	<hr/>	<hr/>	<hr/>
<b>C S</b>	<b>0 0</b>	<b>0 1</b>	<b>0 1</b>	<b>1 0</b>

□ 半加器

□ 输入：X, Y

□ 输出：和位S, 进位C

X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

S		Y
	0	1 <sub>1</sub>
X	1 <sub>2</sub>	3

C		Y
	0	1
X	2	1 <sub>3</sub>

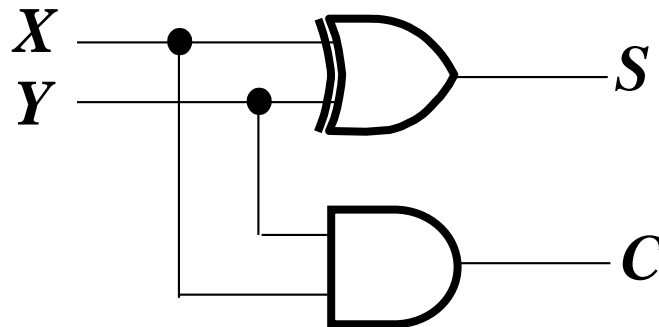


---

□ 可以得到多种表示

□ 最常见实现：

$$S = X \oplus Y$$
$$C = X \cdot Y$$



---

□ 可以得到多种表示

□ 其它实现：

$$(a) \begin{aligned} S &= X \cdot \bar{Y} + \bar{X} \cdot Y \\ C &= X \cdot Y \end{aligned}$$

$$(c) \begin{aligned} S &= \overline{(C + \bar{X} \cdot \bar{Y})} \\ C &= X \cdot Y \end{aligned}$$

$$(b) \begin{aligned} S &= (X + Y) \cdot (\bar{X} + \bar{Y}) \\ C &= X \cdot Y \end{aligned}$$

$$(d) \begin{aligned} \underline{S} &= (\underline{X} + \underline{Y}) \cdot \bar{C} \\ \bar{C} &= (\bar{X} + \bar{Y}) \end{aligned}$$

---

□ 全加器

□ 输入：X, Y, 进位Z

□ 输出：和位S, 进位C

Z	0	0	0	0
X	0	0	1	1
<u>+ Y</u>	<u>+ 0</u>	<u>+ 1</u>	<u>+ 0</u>	<u>+ 1</u>
C S	0 0	0 1	0 1	1 0

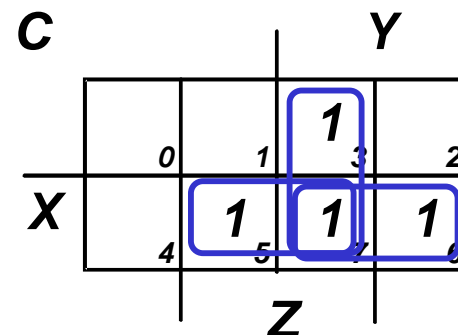
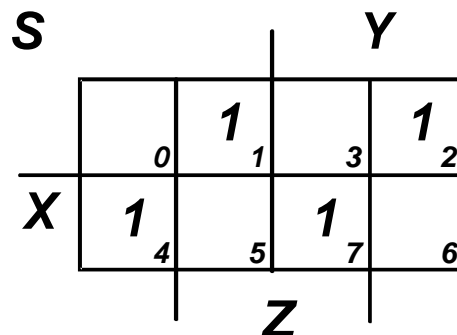
Z	1	1	1	1
X	0	0	1	1
<u>+ Y</u>	<u>+ 0</u>	<u>+ 1</u>	<u>+ 0</u>	<u>+ 1</u>
C S	0 1	1 0	1 0	1 1

□ 全加器

□ 输入：X, Y, 进位Z

□ 输出：和位S, 进位C

X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



---

□ 卡诺图化简得到

$$S = X \bar{Y} \bar{Z} + \bar{X} Y \bar{Z} + \bar{X} \bar{Y} Z + X Y Z$$
$$C = X Y + X Z + Y Z$$

□ S 是三位异或函数 (奇函数)

$$S = X \oplus Y \oplus Z$$

□ C是  $C = X Y + (X \oplus Y) Z$

➤ 项  $X \cdot Y$  是进位生成

➤ 项  $X \oplus Y$  是进位传播

## □ 全加器概要图

□  $X, Y, Z$  (上页) 是  $A, B, C$

□ 同时

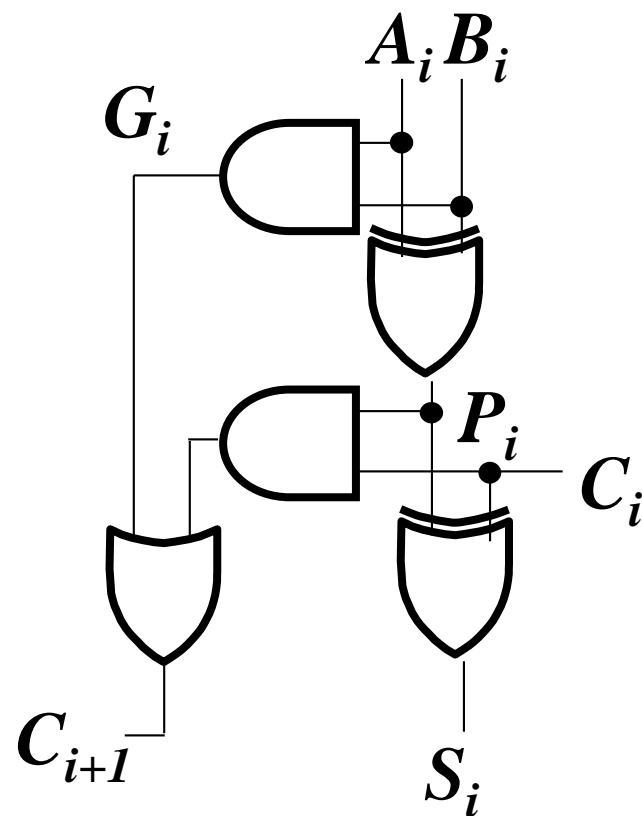
➤  $G$  = 进位生成

➤  $P$  = 进位传播

□ 组成

➤ 两个半加器

➤ 一个或门

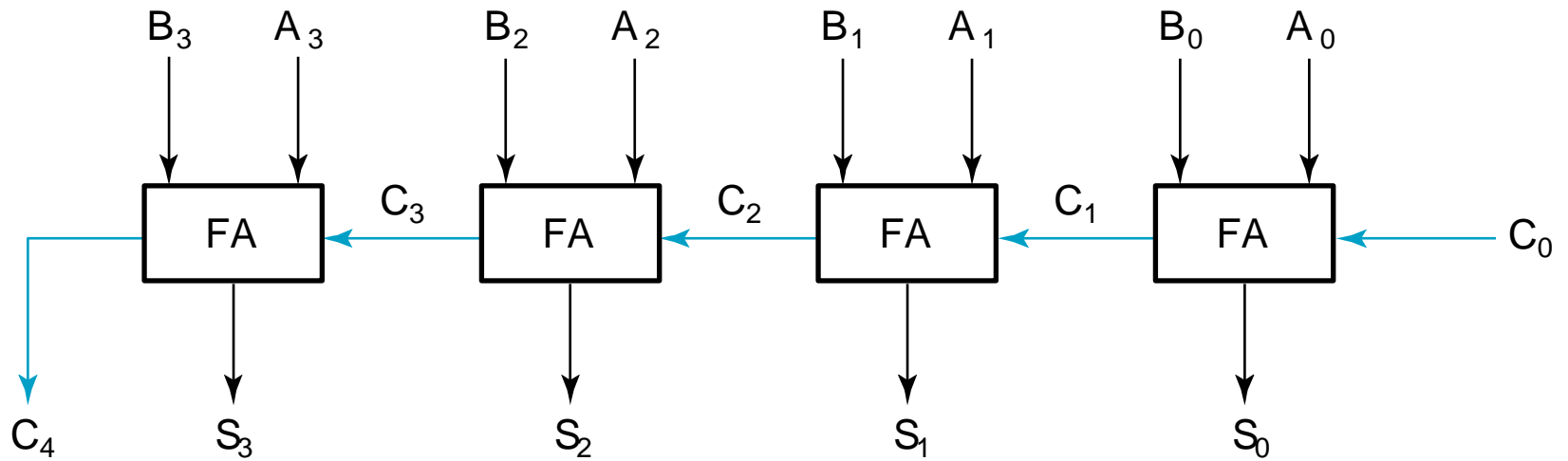


## □ 4位行波进位加法器

➤ 迭代阵列

➤ 单元

➤ 1位全加法器



### 3. 二进制减法器

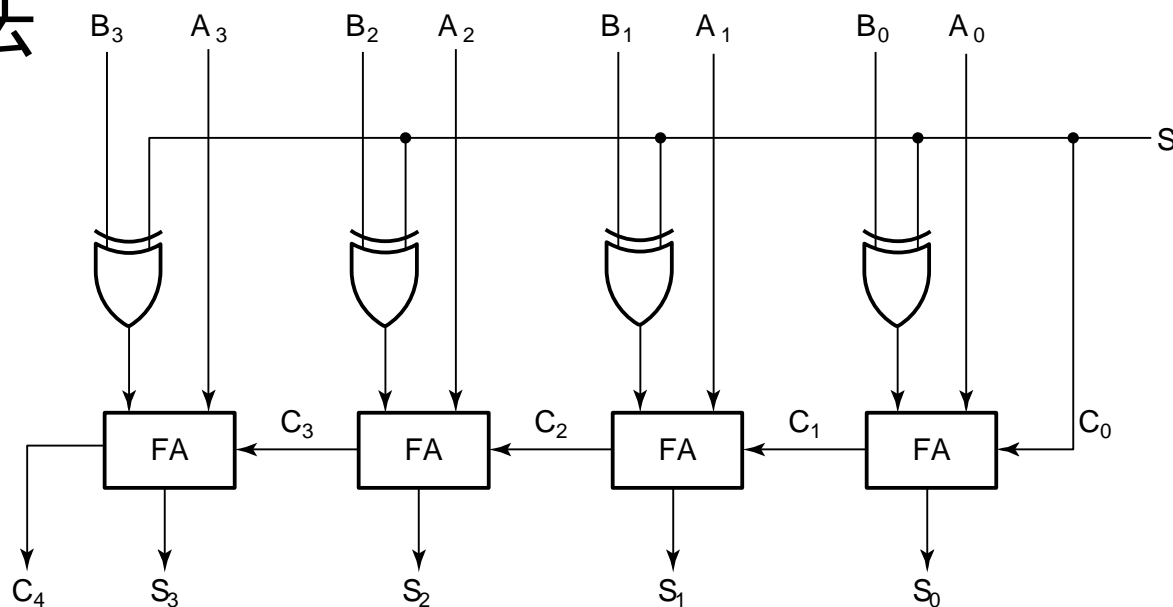
□ 减法可以按照补码的加法执行

➤ 取反加1

□ 电路如图所示，计算 $A+B$ 和 $A-B$

➤  $S=1$ ，减法

➤  $S=0$ ，加法





## □ 补码

➤ 一般的减法：将减数N从被减数M从减去 ( $M-N$ )

➤ 如果最高位没有借位，则  $M \geq N$ ，则结果非负和正确的。

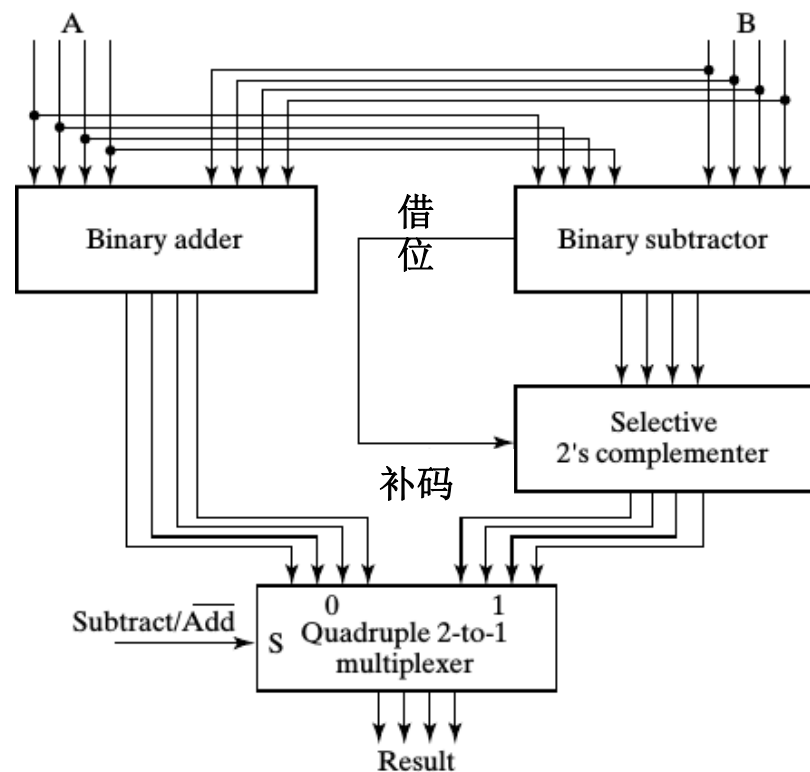
➤ 如果最高位产生借位，则  $N > M$ ，则  $M - N + 2^n$  从  $2^n$  中减去，并且给结果加上符号

$$\begin{array}{r} 0 \\ 1001 \\ - 0111 \\ \hline 0010 \end{array}$$

$$\begin{array}{r} 1 \\ 0100 \\ - 0111 \\ \hline 1101 \\ 10000 \\ - 1101 \\ \hline (-) 0011 \end{array}$$

## □ 补码

- 一般的减法：将减数N从被减数M中减去 ( $M-N$ )
- 太复杂
- $2^n - N$  是N的补码!
- 目标：
  - 共享加法和减法的逻辑



## □ 补码

- Radix complement
- $n$ 位 $r$ 进制补码定义:  $r^n - N$
- 2进制补码定义:  $2^n - N$
- 反码加1是补码

## □ 反码

- Diminished radix complement
- 减法可以按照加被减数的补码的方式执行
- 如果结果是负的, 取结果的2进制补码
  - 对于 $r$ 进制,  $(r - 1)$ 进制补码
  - 对于2进制, 1进制的补码

---

## □ 补码实现减法

- 数值的符号-补码表示
- 首位：符号，0正数，1负数
- 剩余位：二进制补码

-9: 10001001 (符号-数值)

11110111 (符号-补码) ✓

---

## □ 补码实现减法

### ➤ 加法

- ① 将两个数（包含符号位）相加，丢弃进位
- ② 如果相加的两个数符号位相同，而结果的符号不同，则有溢出发生。
- ③ 结果的符号见第一步

### ➤ 减法

- ① 将减数变成补码格式（包括符号位），然后按照加法进行。

---

## □ 补码实现减法

■ 例1:

$$\begin{array}{r} 1101 \\ + \underline{0011} \end{array}$$

■ 例2:

$$\begin{array}{r} 1101 \\ - \underline{0011} \end{array}$$

---

## □ 补码实现减法

■ 例3:

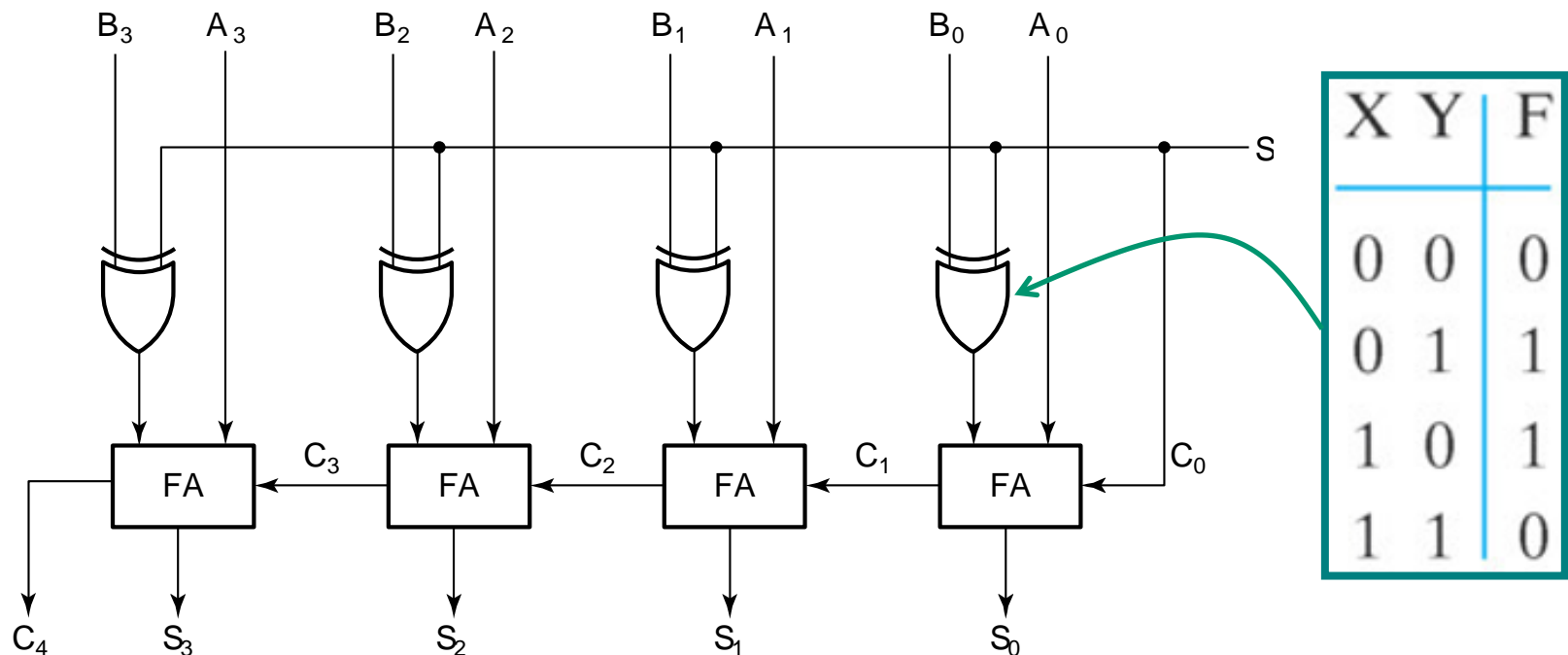
$$\begin{array}{r} 00000110 \\ - \underline{11110000} \end{array}$$

■ 例4:

$$\begin{array}{r} 11111010 \\ - \underline{11110011} \end{array}$$

## □ 补码加/减法器

- $S=1$ , 减法,  $B$ 的反码通过XOR来得到, 同时将  $C_0$  置为1.
- $S=0$ , 加法,  $B$ 直接通过门, 不改变。





## 4. 溢出检测

□ 溢出发生的原因：对于n位的加法或者减法，结果需要n+1位来存储，可能发生在

- 对两个同符号数进行相加
- 对两个异符号数进行相减


$$\begin{array}{r} 01000110 \\ + 01010000 \\ \hline 10010110 \end{array}$$


$$\begin{array}{r} 10111010 \\ + 10110000 \\ \hline 01101010 \end{array}$$

- 溢出检测可以通过检测符号位的进位输入和符号位的进位输出是否一致来判断

## 4. 溢出检测

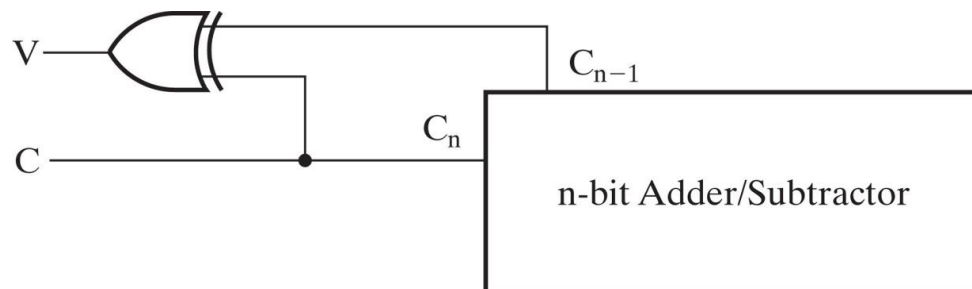
### □ 溢出正确与错误情况

	$C_n$	$C_{n-1}$						
	0	0	0	0	1	1	1	1
	0	0	0	0	1	1	1	1
	+	0	-	1	-	0	+	1
	0	0	0	0	1	1	1	1

$C_n$	$C_{n-1}$							
0	1	0	1	1	0	1	0	
0	0	0	1	1	0	1	1	
+	0	-	1	-	0	+	1	
1	1	1	0	0	0	0	0	

### □ 最简单的溢出检测实现

$$V = C_n \oplus C_{n-1}$$



## 5. 其它算术功能

---

- 压缩
- 递增
- 递减
- 乘以常数
- 除以常数
- 零填充
- 符号扩展

---

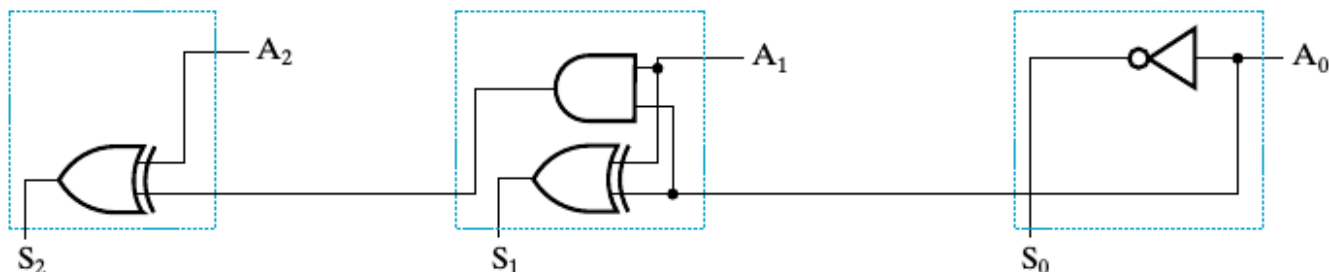
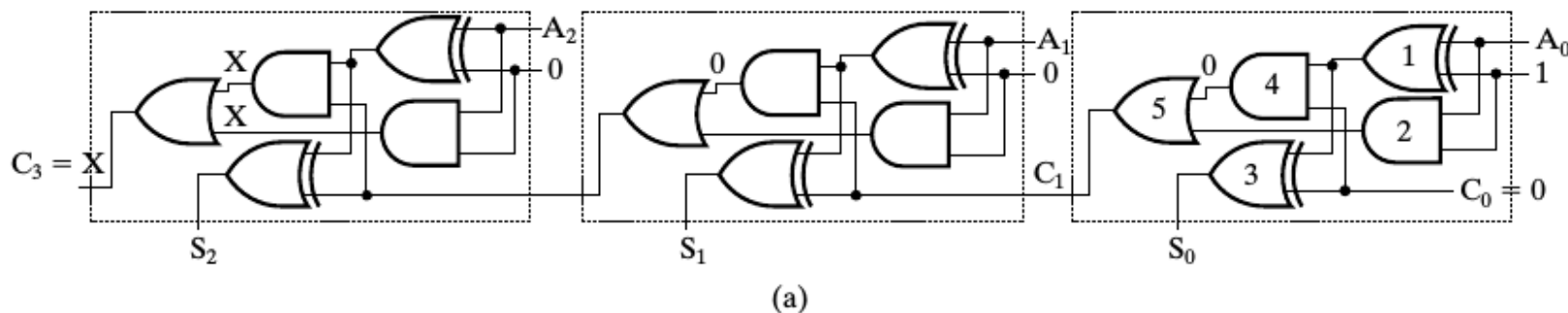
## □ 压缩

- 一种简化一个功能模块的电路逻辑来实现另外一个功能模块的技术。
  - 新函数可以通过在原函数上输入值可以实现, 输入值固定为基本函数;
  - 在这里基本函数指0和1 (不包含X和X);
  - 通过固定0和1后, 布尔方程和逻辑电路图可以通过已有的规则化简。

## □ 压缩

➤ 例子：将一个波形进位加法器压缩成一个递增器， $n=3$

**B=001**



---

## □ 递增

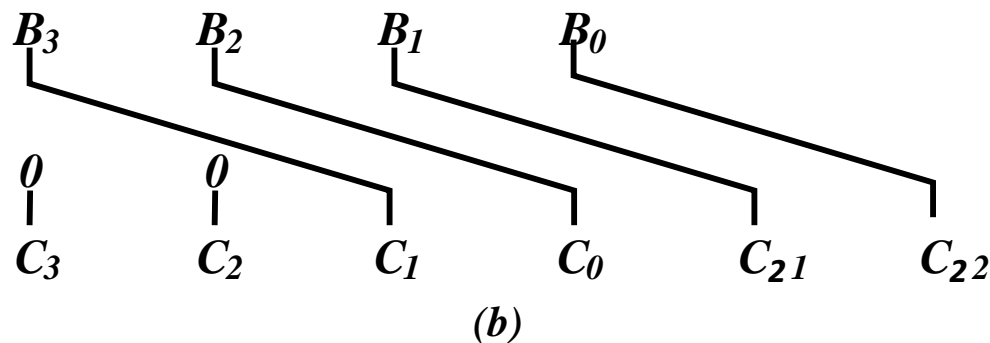
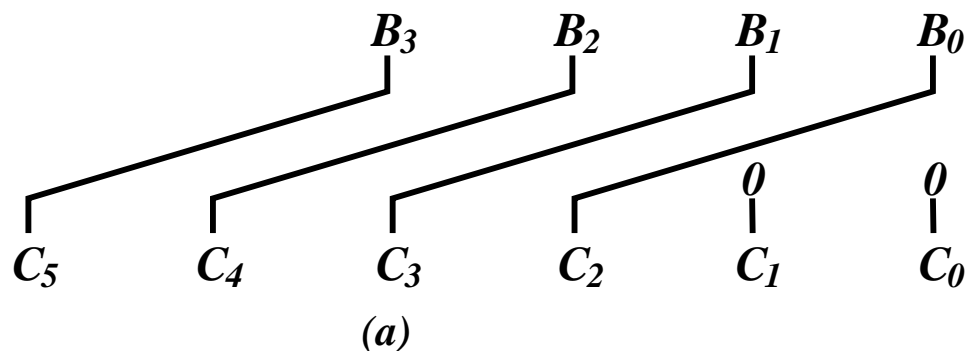
- 对一个算术变量固定值
- 固定值通常为1
- 例子:  $A + 1$ ,  $B + 4$
- 功能模块被成为递增器

## □ 递减

- 将被减数固定一个值
- 固定值经常为1
- 例子:  $A - 1$ ,  $B - 4$
- 功能模块被称为递减器

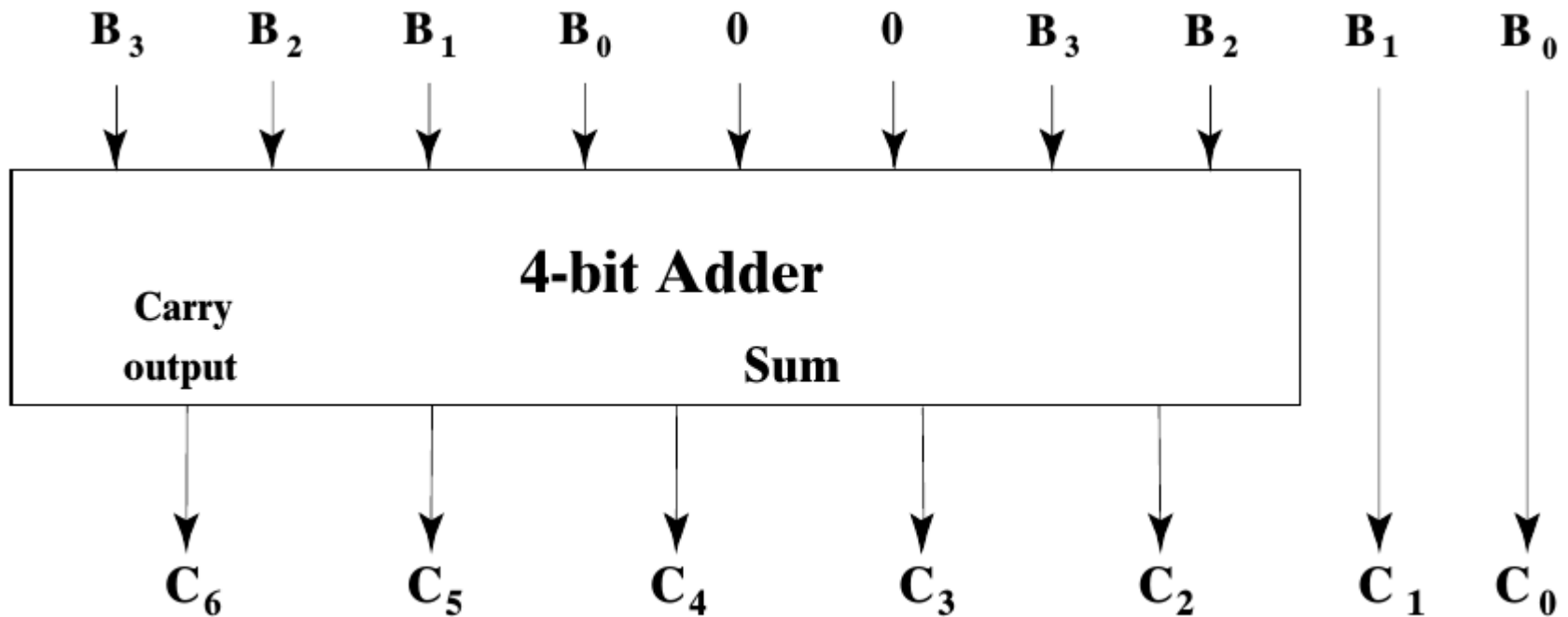
## □ 乘/除 $2^n$

- 乘以 100
  - 左移 2位
- 除以100
  - 右移 2位
  - 剩下的保留



## □ 乘以常数

➤ B(3:0) 乘以 101





---

## □ 零填充

- 对 $m$ 位的操作数填充0以使其变成 $n$ 位 ( $n > m$ ) , 可以左填充, 也可以右填充
- 例子: 11110101 填充成 16 位
  - 左填充: 0000000011110101
  - 右填充: 1111010100000000

---

## □ 符号扩展

- 在操作数左端填充符号位

- 例子

- 正操作数: 01110101 扩展到16位  
0000000001110101

- 负操作数: 11110101 扩展到16位:  
11111111111110101