中山大學 软件工程学院

SUN YAT-SEN UNIVERSITY SCHOOL OF SOFTWARE ENGINEERING

计算机组成原理

授课老师：吴炜滨

大纲

- 定点运算
  - 移位运算
  - 加减法运算
  - 乘法运算
  - 除法运算

大纲

- 定点运算
  - 移位运算
    - 移位运算的数学意义
    - 算术移位规则
    - 算术移位的硬件实现
    - 算术移位与逻辑移位的区别

移位运算

■ 移位的数学意义

- 实现数据的放缩

$$15.\mathrm{m} = 1500.\mathrm{cm}$$

小数点右移 2 位

$$N = (\mathrm{d}_{n-1}\mathrm{d}_{n-2}\cdots\mathrm{d}_1\mathrm{d}_0.\mathrm{d}_{-1}\mathrm{d}_{-2}\cdots\mathrm{d}_{-m})_r$$
$$= \mathrm{d}_{n-1}r^{n-1} + \mathrm{d}_{n-2}r^{n-2} + \cdots + \mathrm{d}_1r^1 + \mathrm{d}_0r^0 + \mathrm{d}_{-1}r^{-1} + \cdots + \mathrm{d}_{-m}r^{-m}$$
$$= \sum_{i=-m}^{n-1} d_i r^i$$

移位运算

移位的数学意义

- 实现数据的放缩

$$15.\mathrm{m} = 1500.\mathrm{cm}$$

小数点右移 2 位

这里的左右指的是：小数点不动，各位数字的移动方向

- 机器用语
  - 15 相对于小数点 左移 2 位 （小数点不动）
  - 计算机中左移
    - 绝对值扩大（二进制，左移一位扩大两倍)
  - 计算机中右移
    - 绝对值缩小（二进制，右移一位缩小为原来的1/2）
  - 在计算机中，移位与加减配合，能够实现乘除运算

算术移位规则

算术移位

- 有符号数的移位
- 使机器数的移位与其对应真值进行移位的效果相同
  - 左移一位，对应真值的绝对值扩大两倍
  - 右移一位，对应真值的绝对值缩小为原来的1/2

算术移位规则

- 仅改变真值的绝对值
  - 符号位不变，仅移动数值位
- 数值位空位添补规则：使其与真值进行移位的效果相同
  - 真值进行移位时，数值位空位添补规则
    - 正数/负数：左移/右移，空位都添0

$$x = +0.x_1x_2\ldots x_k \qquad\qquad x = -0.x_1x_2\ldots x_k$$

算术移位规则

算术移位规则

- 符号位不变，仅移动数值位
- 数值位空位添补规则：使其与真值进行移位的效果相同

真值	码　制	添补代码
正数	原码、补码、反码	0
负数	原　码	0
	补　码	左移 添 0
		右移 添 1
	反　码	1

$$x = +0.x_1x_2\ldots x_k$$
$$[x]_{原、补、反} = 0.x_1x_2\ldots x_k$$

$$x = -0.x_1x_2\ldots x_k$$
$$[x]_{原} = 1.x_1x_2\ldots x_k$$

$$x = -0.x_1x_2\ldots x_k100\ldots000$$
$$[x]_{补} = 1.\overline{x_1}\overline{x_2}\ldots\overline{x_k}100\ldots000$$

$$x = -0.x_1x_2\ldots x_k$$
$$[x]_{反} = 1.\overline{x_1}\overline{x_2}\ldots\overline{x_k}$$

举例

- 设机器数字长为 8 位（含 1 位符号位），写出A = +26时，三种机器数左、右移一位和两位后的表示形式及对应的真值，并分析结果的正确性。

解：$A = +26 = +11010$

则 $[A]_{原} = [A]_{补} = [A]_{反} = 0{,}0011010$

移位操作	机　器　数 $[A]_{原} = [A]_{补} = [A]_{反}$	对应的真值
移位前	0,0011010	+26
左移一位	0,0110100	+ 52
左移两位	0,1101000	+104
右移一位	0,0001101	+13
右移两位	0,0000110	+ 6

举例

- 设机器数字长为 8 位（含 1 位符号位），写出A = –26时，三种机器数左、右移一位和两位后的表示形式及对应的真值，并分析结果的正确性。

解：$A = -26 = -11010$

原码

移位操作	机　器　数	对应的真值
移位前	1,0011010	– 26
左移一位	1,0110100	– 52
左移两位	1,1101000	– 104
右移一位	1,0001101	– 13
右移两位	1,0000110	– 6

举例

- 设机器数字长为 8 位（含 1 位符号位），写出A = –26时，三种机器数左、右移一位和两位后的表示形式及对应的真值，并分析结果的正确性。

解：$A = -26 = -11010$

$[A]_{原} = 1{,}0011010$

补码

移位操作	机　器　数	对应的真值
移位前	1,1100110	– 26
左移一位	1,1001100	– 52
左移两位	1,0011000	– 104
右移一位	1,1110011	– 13
右移两位	1,1111001	– 7

举例

- 设机器数字长为 8 位（含 1 位符号位），写出A = –26时，三种机器数左、右移一位和两位后的表示形式及对应的真值，并分析结果的正确性。

解： $A = -26 = -11010$

$[A]_{原} = 1,0011010$

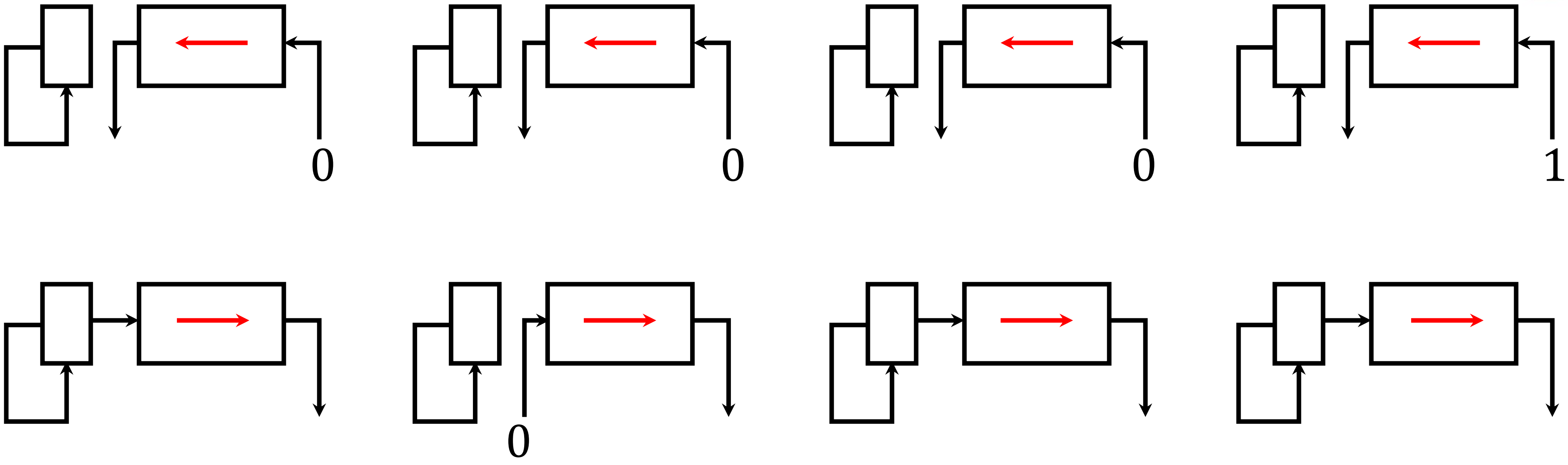
反码

移位操作	机　器　数	对应的真值
移位前	1,1100101	– 26
左移一位	1,1001011	– 52
左移两位	1,0010111	– 104
右移一位	1,1110010	– 13
右移两位	1,1111001	– 6

算术移位的硬件实现

左移

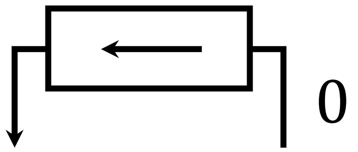
右移



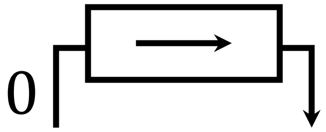
	(a) 真值为正	(b) 负数的原码	(c) 负数的补码	(d) 负数的反码
←丢 1	出错	出错	正确	正确
→丢 1	影响精度	影响精度	影响精度	正确

算术移位和逻辑移位的区别

- **算术移位**
  - 有符号数的移位，符号位要保持不动
- **逻辑移位**
  - 无符号数的移位，所有位都会参与移位运算
- **逻辑左移**
  - 低位添 0，高位移丢



- **逻辑右移**
  - 高位添 0，低位移丢



算术移位和逻辑移位的区别

- 例如

	01010011		10110010
逻辑左移	10100110	逻辑右移	01011001
算术左移	00100110	算术右移（补码）	11011001

大纲

- 定点运算
  - 加减法运算

用补码作加减法运算

用原码作加减法运算的问题

- 需要根据操作数的符号，确定作加法还是减法，并设计、使用加法器/减法器
- ±0原码不统一

要求	数1	数2	实际操作	结果符号
加法	正	正	加	正
加法	正	负	减	可正可负
加法	负	正	减	可正可负
加法	负	负	加	负

用补码作加减法运算

用反码作加减法运算的问题

- ±0反码不统一
- 电路实现较麻烦
- 反码：多用于原码与补码间转换的过渡

用补码作加减法运算

- ±0补码统一
- 补码：找到一个与负数等价的正数来代替负数，实现了加减法形式的统一
  - 原理：计算机当中机器字长有限，使其存在模，运算溢出的最高位将被丢掉
- 现代计算机中都采用补码作加减法运算

大纲

- 定点运算
  - 加减法运算
    - 补码加减法运算的公式
    - 补码加减法溢出的判断
    - 补码加减法的硬件配置

加减法运算

补码加减运算公式

- 加法
  - 连同符号位一起相加，和的符号，通过计算过程自动产生，符号位产生的进位自然丢掉：计算结果按2^{n+1}取模（对整数）或按2取模（对小数）

整数 $$[A+B]_{补} = [A]_{补} + [B]_{补} \pmod{2^{n+1}}$$

小数 $$[A+B]_{补} = [A]_{补} + [B]_{补} \pmod{2}$$

加减法运算

■ 补码加减运算公式

- 减法
  - $[-B]_{补}$由$[B]_{补}$连同符号位在内，每位取反，末位加1而得
  - 连同符号位一起相加，和的符号，通过计算过程自动产生，符号位产生的进位自然丢掉：计算结果按2^{n+1}取模（对整数）或按2取模（对小数）

$$A - B = A + (-B)$$

整数 $[A - B]_{补} = [A + (-B)]_{补} = [A]_{补} + [-B]_{补} \pmod{2^{n+1}}$

小数 $[A - B]_{补} = [A + (-B)]_{补} = [A]_{补} + [-B]_{补} \pmod{2}$

举例

- 设机器字长为5位（含1位符号位），$A = 0.1011$，$B = -0.0101$，求 $[A+B]_{补}$

解：

$$
\begin{array}{rcl}
[A]_{补} & = & 0.1011 \\
+[B]_{补} & = & 1.1011 \\
\hline
[A]_{补}+[B]_{补} & = & \boxed{1}\,0.0110 = [A+B]_{补}
\end{array}
$$

丢掉

验证

$$
\begin{array}{r}
0.1011 \\
-\ 0.0101 \\
\hline
0.0110
\end{array}
$$

$$\therefore A + B = 0.0110$$

举例

- 设机器字长为5位（含1位符号位），$A=-9$，$B=-5$，求 $[A+B]_{补}$

解：

$$
\begin{array}{rcl}
[A]_{补} & = & 1,0111 \\
+[B]_{补} & = & 1,1011 \\
\hline
[A]_{补}+[B]_{补} & = & \boxed{1}\,1,0010 = [A+B]_{补}
\end{array}
$$

丢掉

验证

$$
\begin{array}{r}
-1001 \\
+\ -0101 \\
\hline
-1110
\end{array}
$$

$\therefore A+B=-1110$

举例

- 设机器数字长为 8 位（含 1 位符号位）且 $A = 15$， $B = 24$，用补码求 $A - B$

解： $A = 15 = 0001111$

$B = 24 = 0011000$

$[A]_{补} = 0,0001111$ $[B]_{补} = 0,0011000$

$$
\begin{array}{rcl}
[A]_{补} & = & 0,0001111 \\
+[-B]_{补} & = & 1,1101000 \\
\hline
[A]_{补}+[-B]_{补} & = & 1,1110111 = [A-B]_{补}
\end{array}
$$

$\therefore A - B = -1001 = -9$

练习

- 设 $x=\frac{11}{16}$，$y=\frac{7}{16}$，用补码求 $x+y$（假设机器字长为5位，包括一位的符号位，而且这台机器是一个纯小数的定点机）

解：

$$x=\frac{1011}{10000}=0.1011 \qquad y=\frac{111}{10000}=0.0111$$

$$[x]_{补}=0.1011 \qquad [y]_{补}=0.0111$$

$$[x+y]_{补}=0.1011+0.0111=1.0010$$

$$\therefore\ x+y=-0.1110=-\frac{14}{16}$$ 错

因为正确运算结果大于1，超出小数定点机能表示的范围[-1,1)

练习

- 设机器数字长为 8 位（含 1 位符号位） 且 $A=-97$，$B=+41$，用补码求 $A-B$

解： $A=-1100001$　　　$B=0101001$

$[A]_{补}=1{,}0011111$　　　$[B]_{补}=0{,}0101001$

$[-B]_{补}=1{,}1010111$

丢掉

$[A-B]_{补}=1{,}0011111+1{,}1010111=10{,}1110110$

$\therefore A-B=+1110110=+118$

错　因为正确运算结果超出该整数定点机能表示的范围[-128, +127]

溢出判断：补码定点加减运算

一位符号位判溢出

- 参加加法的两个数（减法时为被减数$[A]_{补}$和“求补”以后的减数$[-B]_{补}$）符号不同，不会发生溢出
- 参加加法的两个数（减法时为被减数$[A]_{补}$和“求补”以后的减数$[-B]_{补}$）符号相同，其结果的符号与原操作数的符号不同，即为溢出
  - 硬件实现
    - 数值最高位的进位 ⊕ 符号位的进位 = 1　　溢出
    - 如

两个正数相加：$1 \oplus 0 = 1$

两个负数相加：$0 \oplus 1 = 1$

} 有 溢出

两个负数或一正一负相加：$1 \oplus 1 = 0$

两个正数或一正一负相加：$0 \oplus 0 = 0$

} 无 溢出

溢出判断：补码定点加减运算

- 一位符号位判溢出
  - 硬件实现
    - 数值最高位的进位 ⊕ 符号位的进位 = 1　　　溢出
    - 记录数值最高位的进位、符号位的进位
    - **把这两个进位送入到一个异或电路**
    - 异或电路输出等于1，给溢出标志置1表示发生了溢出，如果输出等于0，就没有发生溢出

溢出判断：补码定点加减运算

- 两位符号位判溢出：变形补码
- 变形补码
  - 有两位符号位
  - 小数

$$[x]_{补'} = \begin{cases} x & 1 > x \geq 0 \\ 4 + x & 0 > x \geq -1 \end{cases} \pmod 4$$

  - x 为真值
  - 正数：00.XXXXXX（符号位：00，数值部分：与x相同）
  - 负数：11.XXXXXX（符号位：11，数值部分：x数值部分每位取反，末位加一）

溢出判断：补码定点加减运算

■ 变形补码

- 有两位符号位
- 整数

$$[x]_{补'} = \begin{cases} 00,\ x & 2^n > x \ \geq \ 0 \\ 2^{n+2} + x & 0 > x \ \geq -2^n \pmod{2^{n+2}} \end{cases}$$

- x 为真值，n 为整数数值位的位数
- 正数：00，XXXXXX（符号位：00，数值部分：与x相同）
- 负数：11，XXXXXX（符号位：11，数值部分： x数值部分每位取反，末位加一）

溢出判断：补码定点加减运算

- 变形补码
  - 运算规则：2位符号位连同数值部分一起运算，高位符号位产生的进位自动丢掉
    - 小数

$$[x+y]_{补'} = [x]_{补'} + [y]_{补'} \pmod 4$$

$$[x-y]_{补'} = [x]_{补'} + [-y]_{补'} \pmod 4$$

    - 整数

$$[x+y]_{补'} = [x]_{补'} + [y]_{补'} \pmod{2^{n+2}}$$

$$[x-y]_{补'} = [x]_{补'} + [-y]_{补'} \pmod{2^{n+2}}$$

溢出判断：补码定点加减运算

两位符号位判溢出：变形补码

- 结果的双符号位 相同　　未溢出

00.×××××　　00,×××××

11.×××××　　11,×××××

- 结果的双符号位 不同　　溢出

10.×××××　　10,×××××

01.×××××　　01,×××××

- 最高符号位 代表其 真正的符号
- 第二个符号位是运算的数值溢出的部分

补码加减法的硬件配置

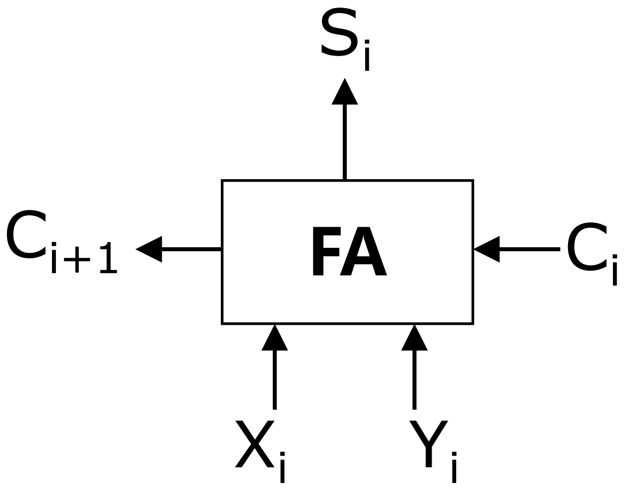
■ 二进制加法运算

- 各位逐位相加，进位从右至左传递
- 硬件实现：从一位加法入手

$$
\begin{array}{rcccc}
 & X_n \cdots\cdots & X_2 & X_1 & X_0 \\
+ & Y_n \cdots\cdots & Y_2 & Y_1 & Y_0 \\
\hline
 & ?_n \cdots\cdots & ?_2 & ?_1 & ?_0
\end{array}
$$

补码加减法的硬件配置

- 带进位链的一位全加器
  - 算术运算变成逻辑电路



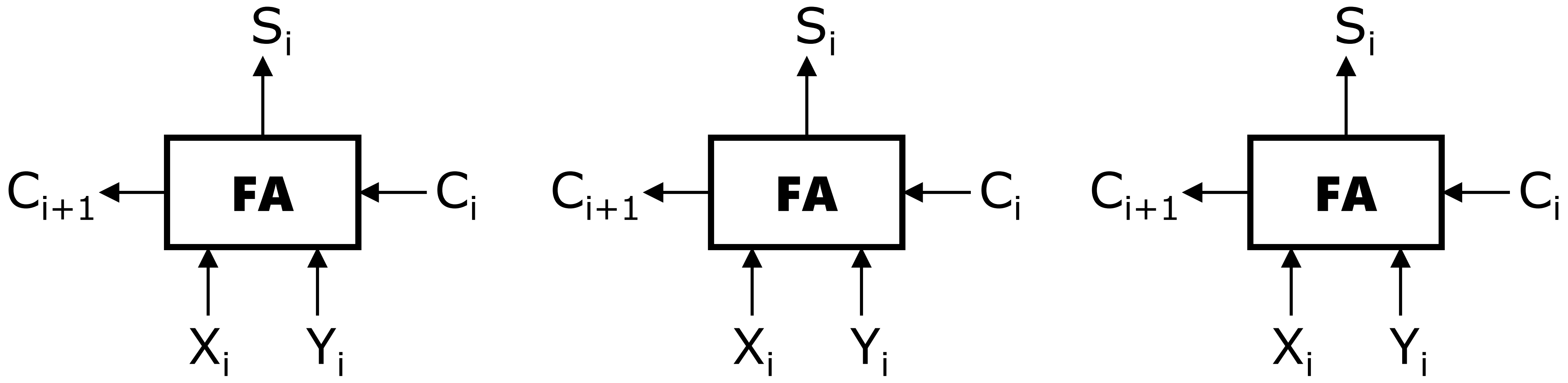
被加数X_i	加数Y_i	低位进位输入C_i	和数S_i	高位进位输出C_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$S_i = X_i \oplus Y_i \oplus C_i$$

$$C_{i+1} = X_i Y_i + (X_i \oplus Y_i) C_i$$

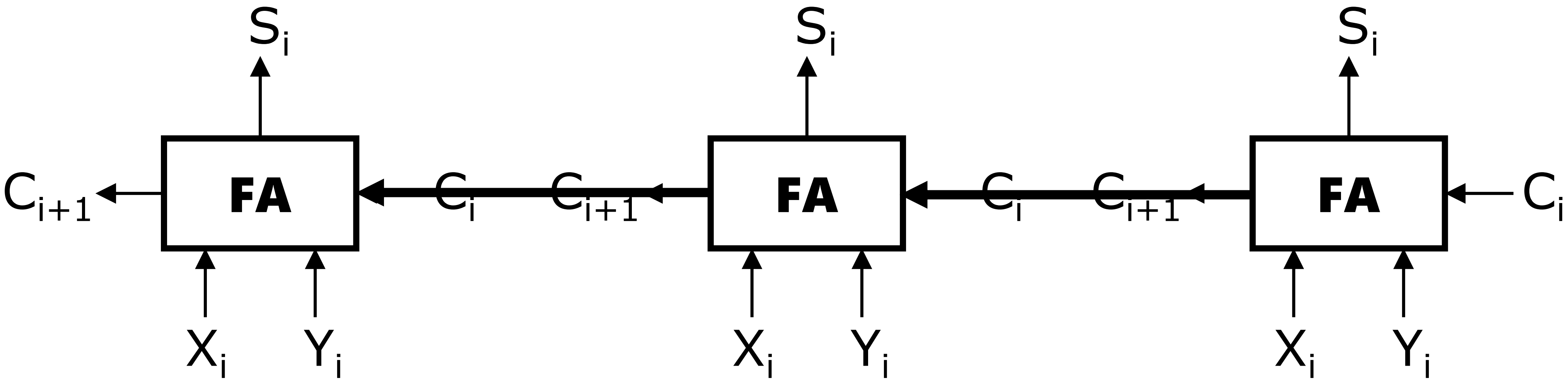
补码加减法的硬件配置

- n+1位加法器
  - 包含n+1个全加器



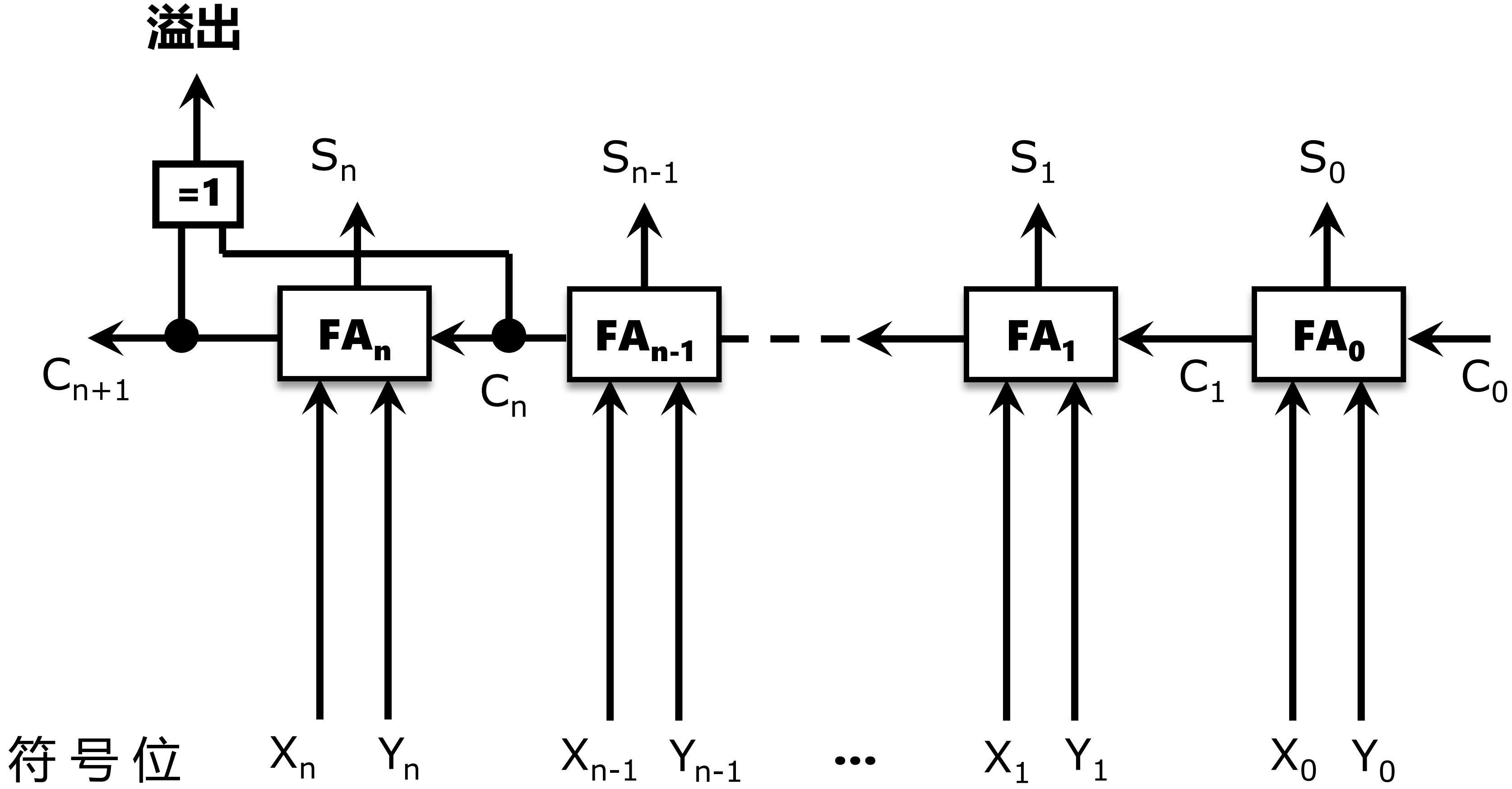
补码加减法的硬件配置

- n+1位加法器
  - 包含n+1个全加器
  - 将n+1个一位全加器串联
  - 低位进位输出连接到高位进位输入



补码加减法的硬件配置

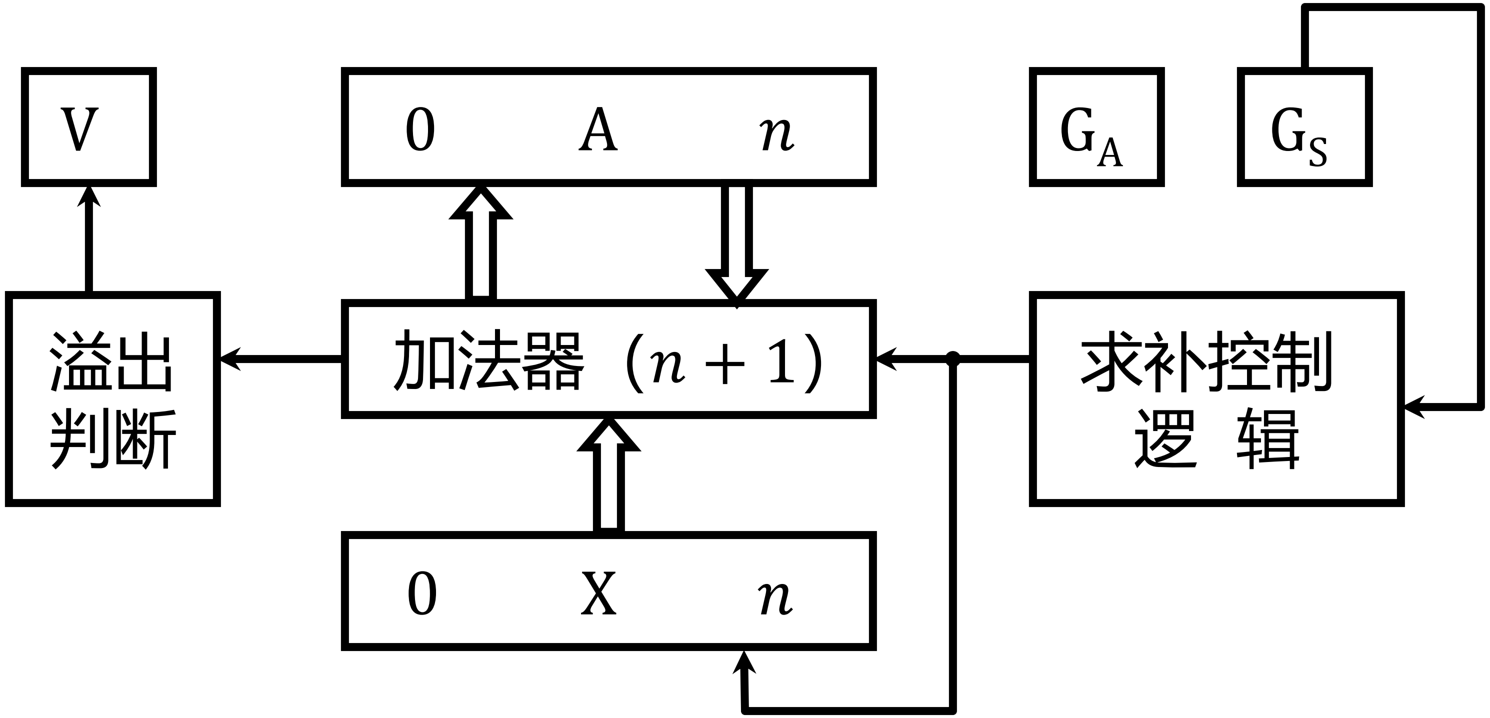
- n+1位加法器
  - 增加溢出判断（单符号位）



补码加减法的硬件配置

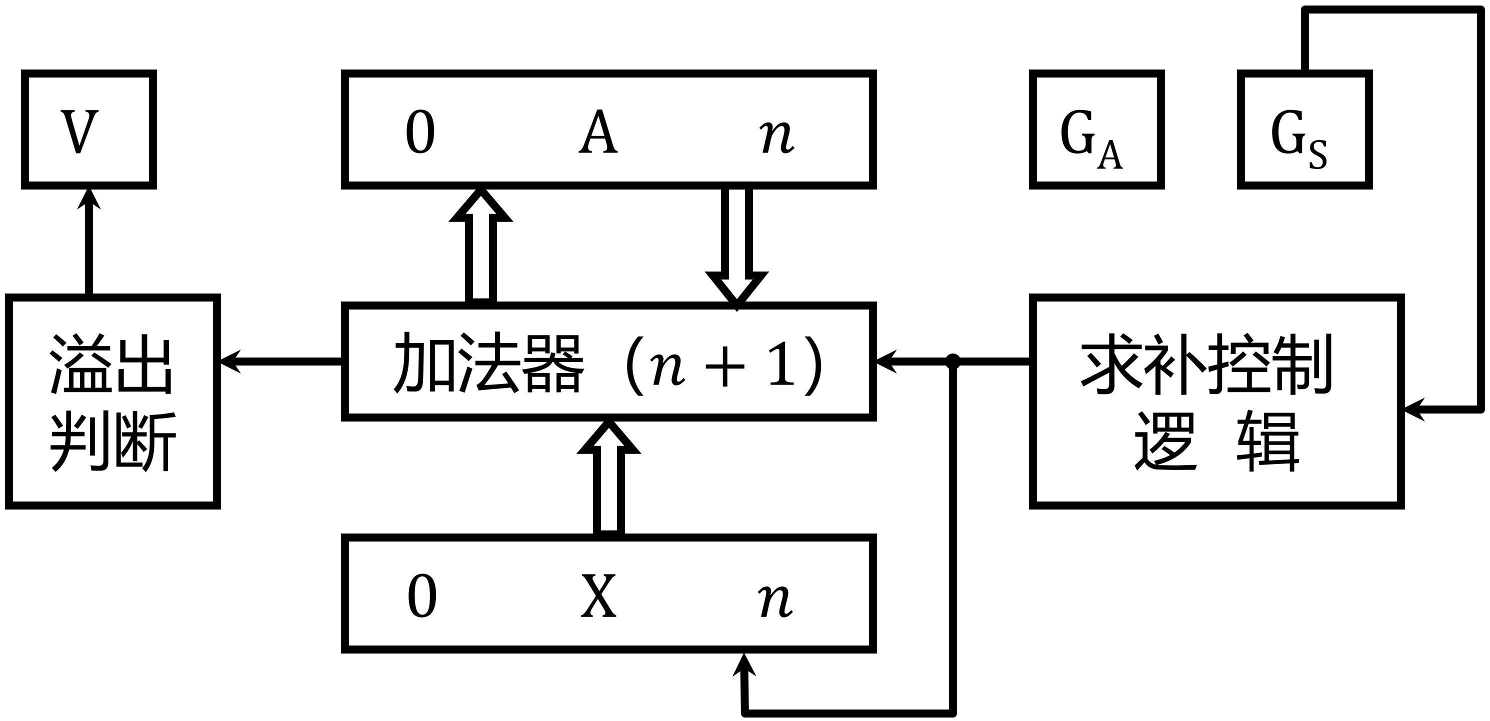
补码定点加减法的硬件配置

- 寄存器A、X 、加法器均 $n+1$ 位（1位符号位，n位数值位）
- A：被加数（或被减数）的补码， X：加数（或减数）的补码
- G_A：加法标记， G_S：减法标记，V：溢出标记



补码加减法的硬件配置

- 补码定点加减法的硬件配置
  - 当作减法时，用减法标记 G_S 控制求补逻辑
    - 将$\bar{X}$送至加法器，同时使加法器的最末位外来进位为1，来求减数的补码



谢谢！