整数的加减运算

本节主要内容

- 有符号数、无符号数 的加/减运算

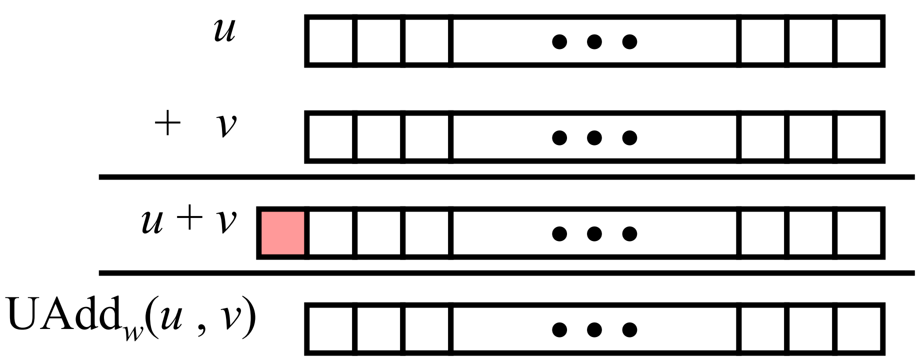
无符号加法

- 字长w: unsigned： 0 ~ (2^w-1)
- 模: 2^w
- 加法：   0 → 1 → … →   Umax → 0
- $UAdd_w(u, v) = (u + v) \bmod 2^w$

操作数: w 位

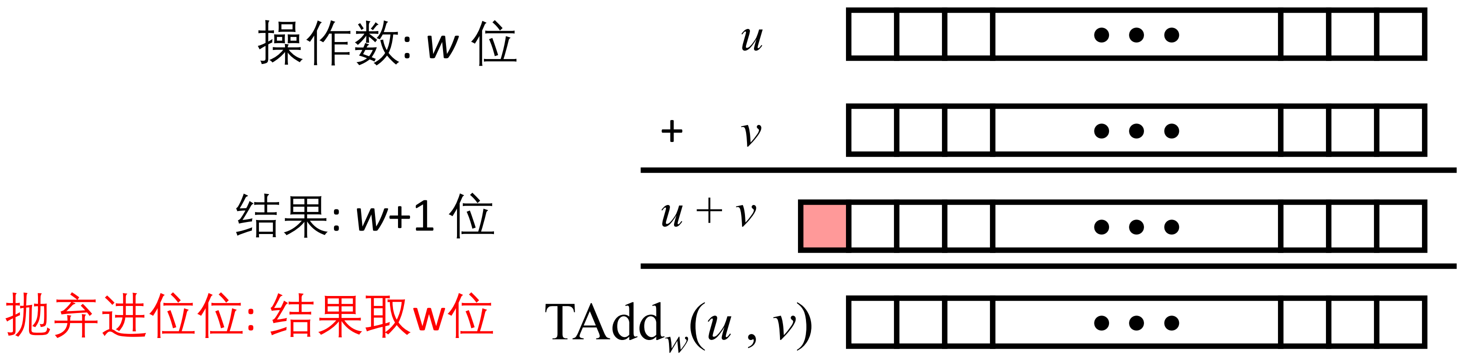
结果: w+1 位

抛弃进位位，结果为 w 位



补码加法：

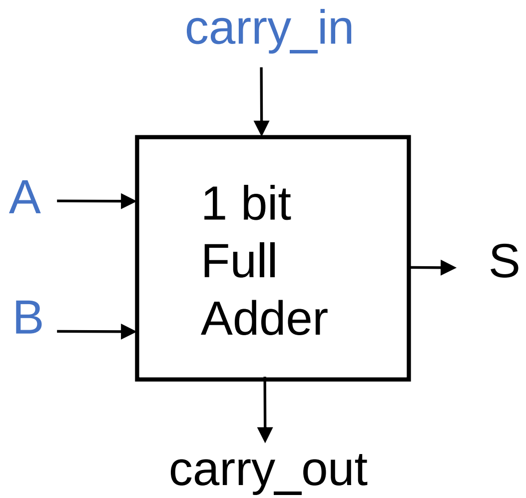
- 补码加法运算公式：$[X]_{补} + [Y]_{补} = [X + Y]_{补}$
- 与无符号数加法一样：直接抛弃最高位的进位位



- 举例：
-
```
int s, t, u, v;
s = (int) ((unsigned) u + (unsigned) v);//无符号加
t = u + v //整数相加
s == t  结果为 true
```

- 无符号整数的加法、有符号整数的加法，共用一个加法单元

一位全加器

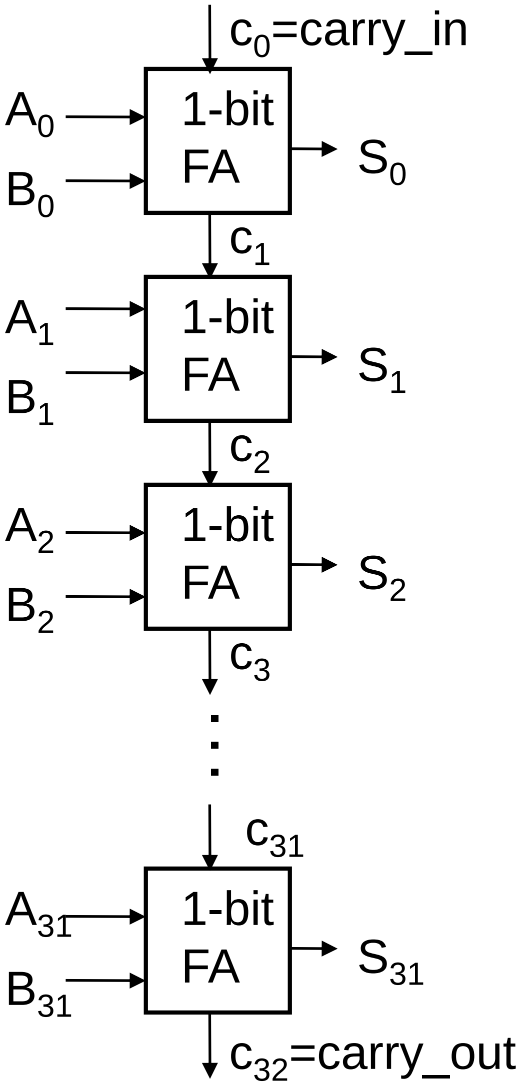


A	B	carry_in	carry_out	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

S = A xor B xor carry_in

carry_out = A & B | A & carry_in | B & carry_in

32位加法器



- 串行进位加法器
  - 优点：简单
  - 缺点：速度慢

- 并行进位加法器
  - 提前计算进位输入
  - 加法运算并行完成

补码减法

- 补码减法可以转化为加法

$[X-Y]_{补} = [X+(-Y)]_{补} = [X]_{补} + [-Y]_{补}$

已知： $[Y]_{补}$， 求 $[-Y]_{补}$

$\mathbf{[Y]_{补} + [-Y]_{补} = 0}$

$\mathbf{[-Y]_{补} = 0 - [Y]_{补} = \overline{[Y]_{补}}} +1$ 假设$Y=[8]_{10} = [00001000]_2$

```
  0 0 0 0 0 0 0 0
- 0 0 0 0 1 0 0 0
- - - - - - - - -
```

```
  1 0 0 0 0 0 0 0 0
-   0 0 0 0 1 0 0 0
- - - - - - - - - -
```

```
  1 1 1 1 1 1 1 1
- 0 0 0 0 1 0 0 0
- - - - - - - - -
  1 1 1 1 0 1 1 1      (取反)
+ 0 0 0 0 0 0 0 1      (加 1)
- - - - - - - - -
  1 1 1 1 1 0 0 0
```

❑ 如何修改加法器电路构建一个 加/减法器?

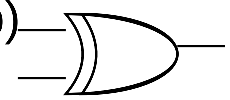
32位加/减法器

$[A+B]_{补} = [A]_{补} + [B]_{补}$

$[A-B]_{补} = [A]_{补} + [-B]_{补}$

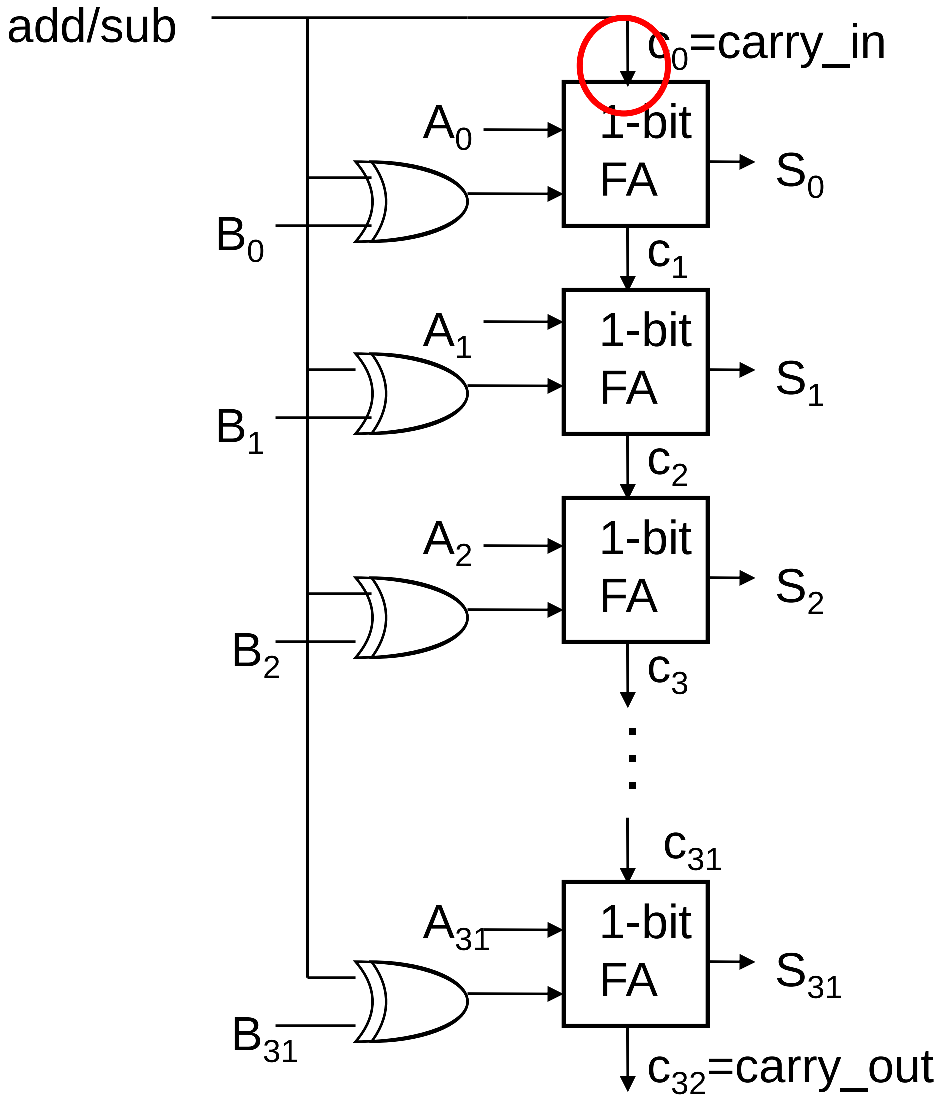
已知$[B]_{补}$，求$[-B]_{补}$： 取反， 最低位加1

control
(0=add,1=sub)
B_0



B_0 if control = 0,
$!B_0$ if control = 1

A	0111	→		0111
B	− 0110	→	+	1001
	0001		+	1
				1 0001

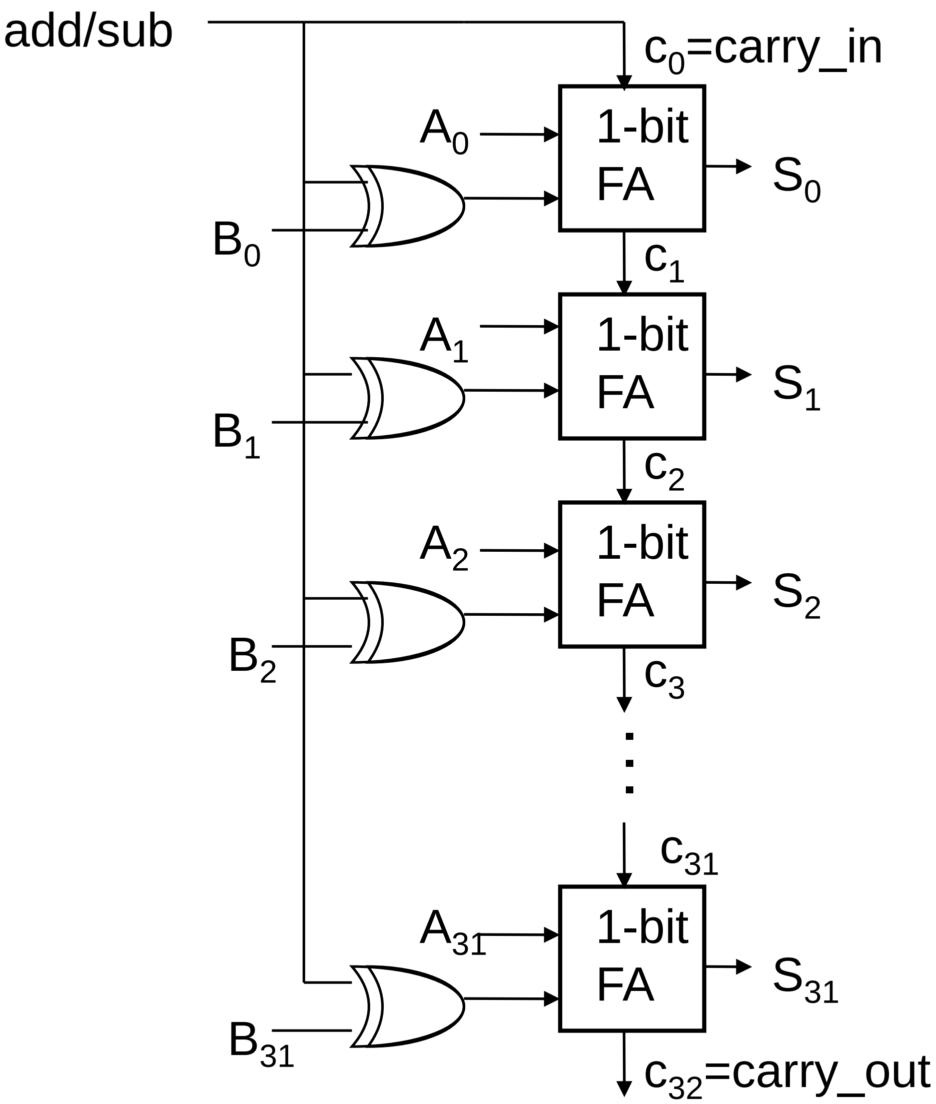


无符号数减法

- 字长w的无符号数是一个模为2^w的计量系统
- 减法： Umax → Umax -1 → … → 1 → 0 → Umax
- 减法可以转化为加法
- $USub_w(u, v) = u - v$
- $= u - v + 2^w$

$= u + (2^w - v)$

$= u + \sim v + 1$



可用于整数、无符号数的加/减法

小结

- 整数加法、无符号整数加法
  - 直接运算，抛弃最高位的进位位
- 整数减法、无符号数减法
  - 转换为加法，
  - 将减数取反加一，并和被减数相加
- 加法/减法共用一个部件
  - 还可以完全其他运算，如：与、或、异或、取反
  - 算逻运算单元（算术运算、逻辑运算）