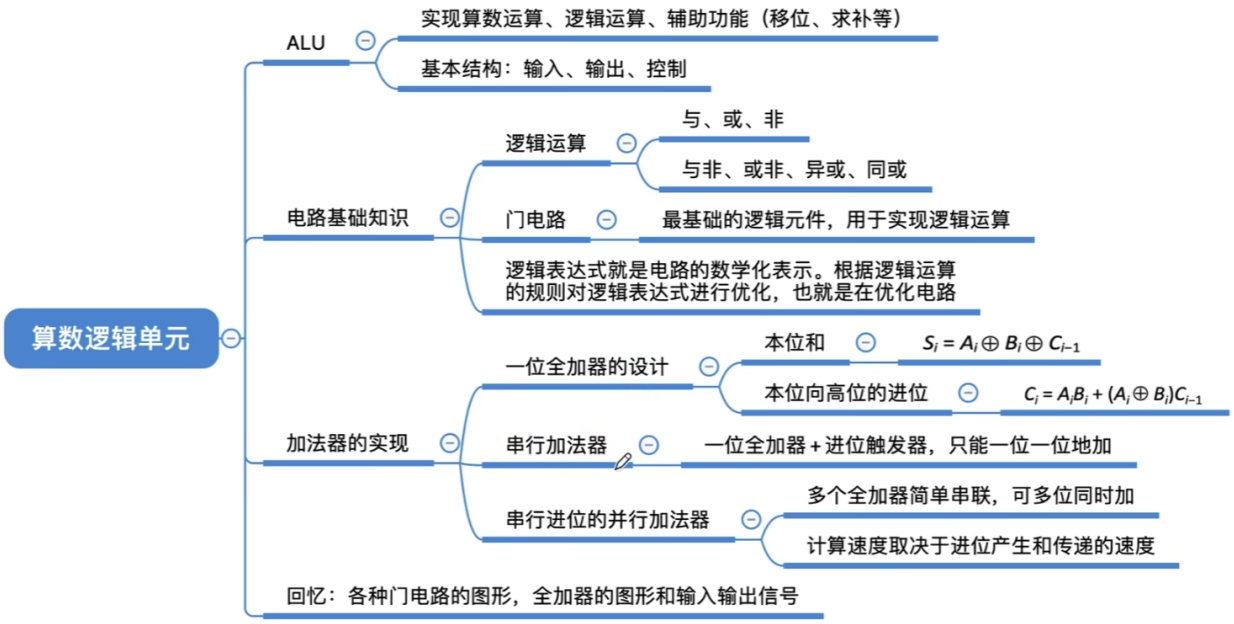
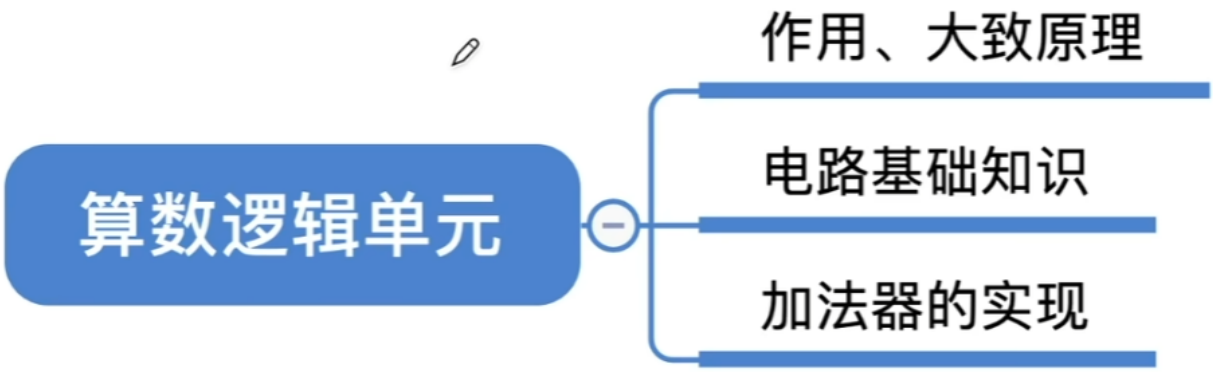
基本运算部件



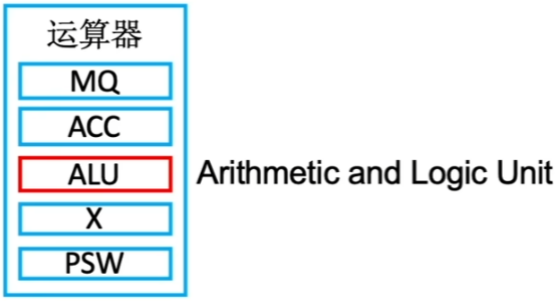


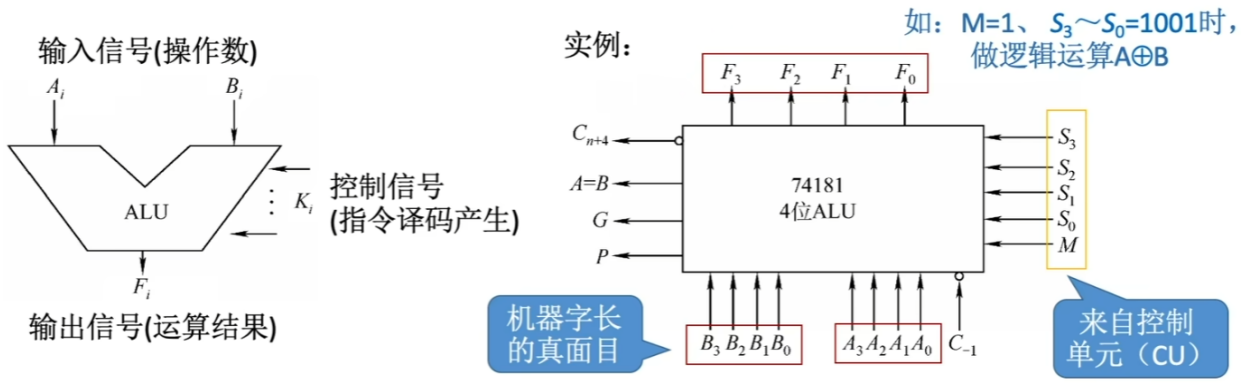
算术逻辑单元（ALU）

算术运算：加、减、乘、除等

逻辑运算：与、或、非、异或等

辅助功能：移位、求补等





最基本的逻辑运算

类比C语言“&&” 与　“||” 或　“!” 非

表达式　Y = A · B　Y = A + B　$Y = \overline{A}$

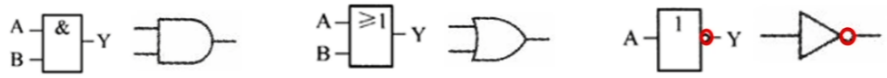
真值表

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

A	Y
0	1
1	0

门电路



输入和输出高/低电平信号(5V/1V)

优先级：与>或

（类比乘法、加法）

AB+CD 先算与再算或

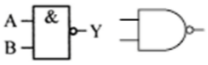
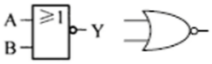
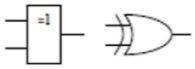
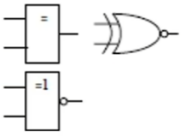
A(C+D)=AC+AD 分配律

ABC=A(BC) 结合律

A+B+C=A+(B+C) 结合律

Tips: 本质上逻辑表达式是对电路的数学化描述，简化逻辑表达式，就是在简化电路，就是在省钱。

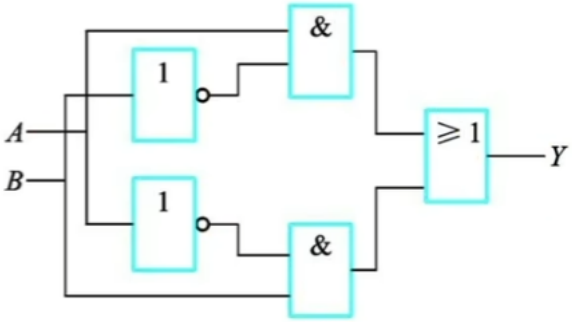
复合逻辑

	与非	或非	异或	同或
表达式	$Y=\overline{A\cdot B}$ $\overline{A}+\overline{B}$	$Y=\overline{A+B}$ $\overline{A}\cdot\overline{B}$	$Y=A\oplus B$	$Y=A\odot B$
真值表	A B Y 0 0 1 0 1 1 1 0 1 1 1 0	A B Y 0 0 1 0 1 0 1 0 0 1 1 0	A B Y 0 0 0 0 1 1 1 0 1 1 1 0	A B Y 0 0 1 0 1 0 1 0 0 1 1 1
电路符号				

A和B不同

$\rightarrow A=0$且$B=1$或$A=1$且$B=0$

$\rightarrow\overline{A}\cdot B+A\cdot\overline{B}$



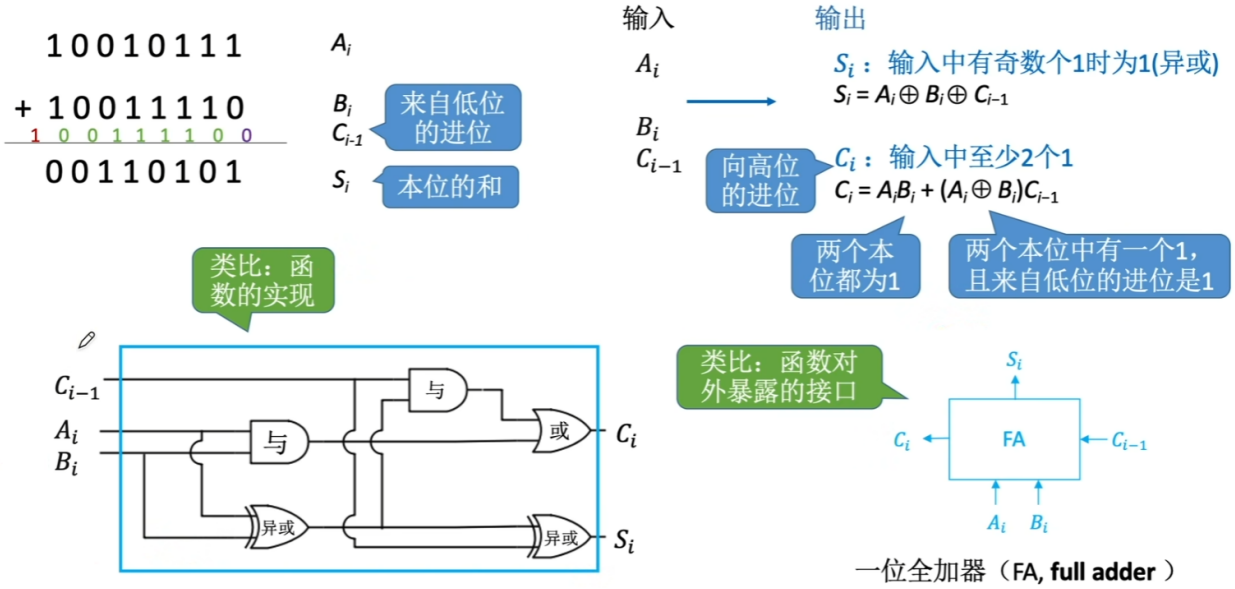
异或门可用与、或、非 组合实现

反演律：

$\overline{A+B}=\overline{A}\cdot\overline{B}$

$\overline{A\cdot B}=\overline{A}+\overline{B}$

一位全加器

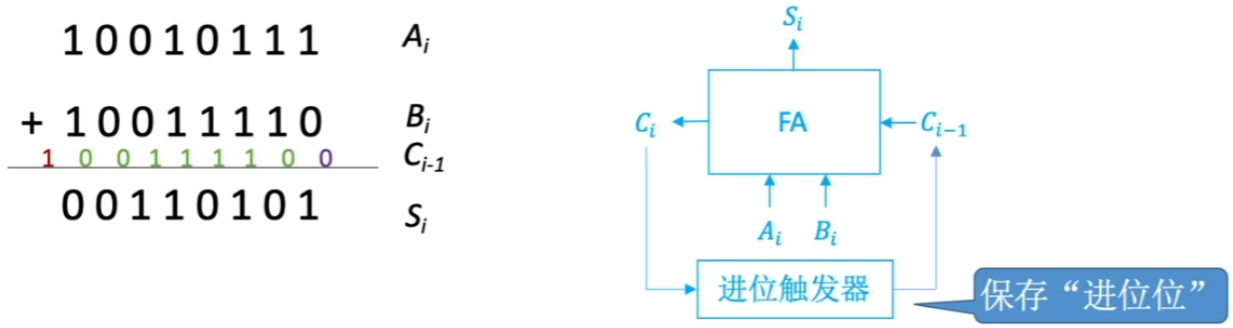


串行加法器

只有一个全加器，数据逐位串行送入加法器中进行运算。

进位触发器用来寄存进位信号，以便参与下一次运算。

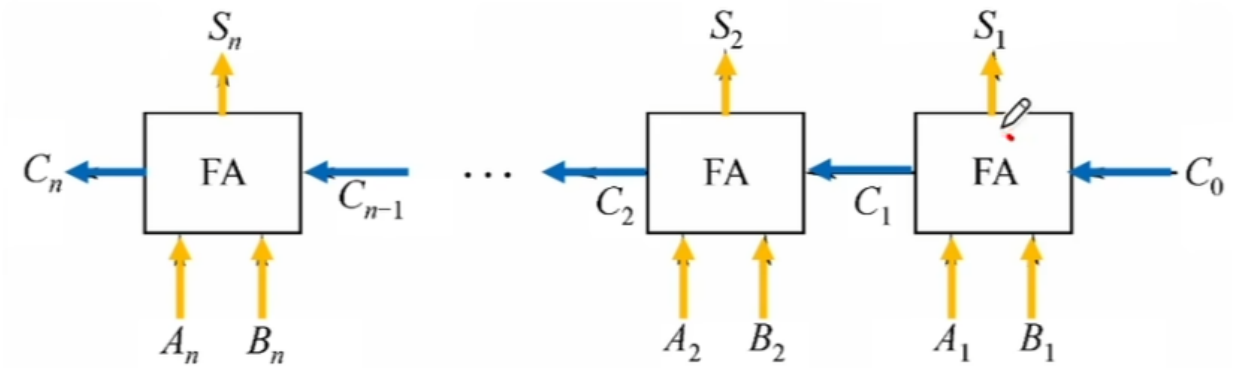
如果操作数长n位，加法就要分n次进行，每次产生一位和，并且串行逐位地送回寄存器。

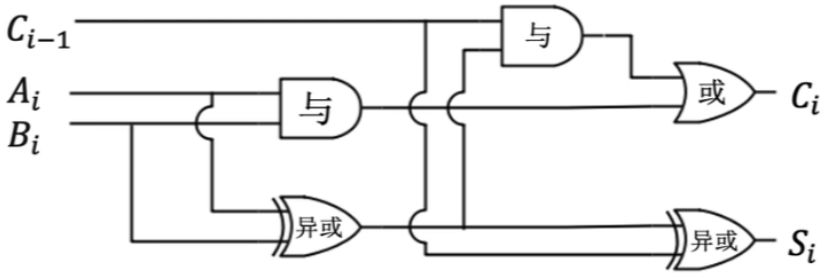


并行加法器

串行进位的并行加法器：把n个全加器串接起来，就可进行两个n位数的相加。

串行进位又称为行波进位，每一级进位直接依赖于前一级的进位，即进位信号是逐级形成的。

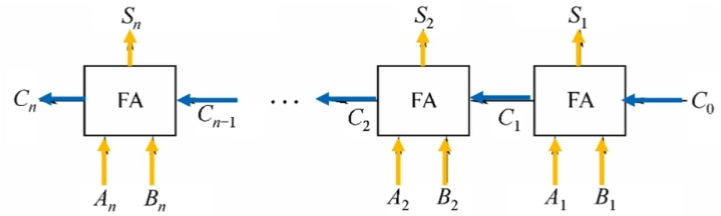




$C_i = A_iB_i + (A_i \oplus B_i)C_{i-1}$

$S_i = A_i \oplus B_i \oplus C_{i-1}$

如何更快的产生进位?



$C_i = A_iB_i + (A_i \oplus B_i)C_{i-1}$

$C_i = A_iB_i + (A_i \oplus B_i)\,(A_{i-1}B_{i-1} + (A_{i-1} \oplus B_{i-1})C_{i-2})$

$C_i = A_iB_i + (A_i \oplus B_i)\,(A_{i-1}B_{i-1} + (A_{i-1} \oplus B_{i-1})(A_{i-2}B_{i-2} + (A_{i-2} \oplus B_{i-2})\,C_{i-3}))$

……

终有一天可以展开到 C_0

记:

$G_i = A_iB_i$

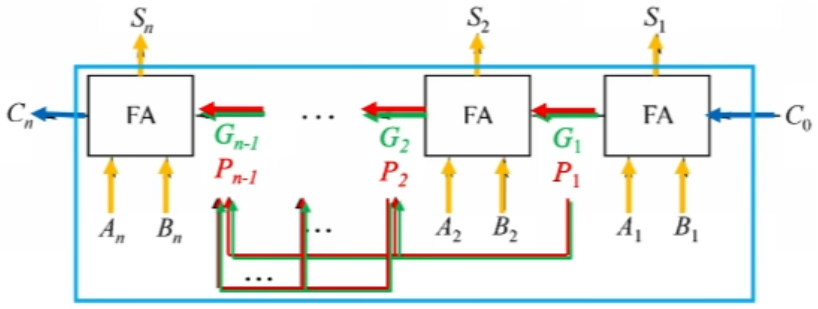
$P_i = A_i \oplus B_i$

刚开始就有的信息

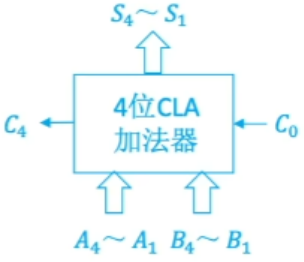
结论:第 i 位向更高位的进位 C_i 可根据 被加数、加数的第 1~i 位, 再结合C_0 即可确定

并行加法器的优化

并行进位的并行加法器:各级进位信号同时形成,又称为先行进位、同时进位



由4个FA和一些新的线路、运算逻辑组成



$G_i = A_i B_i \qquad P_i = A_i \oplus B_i$

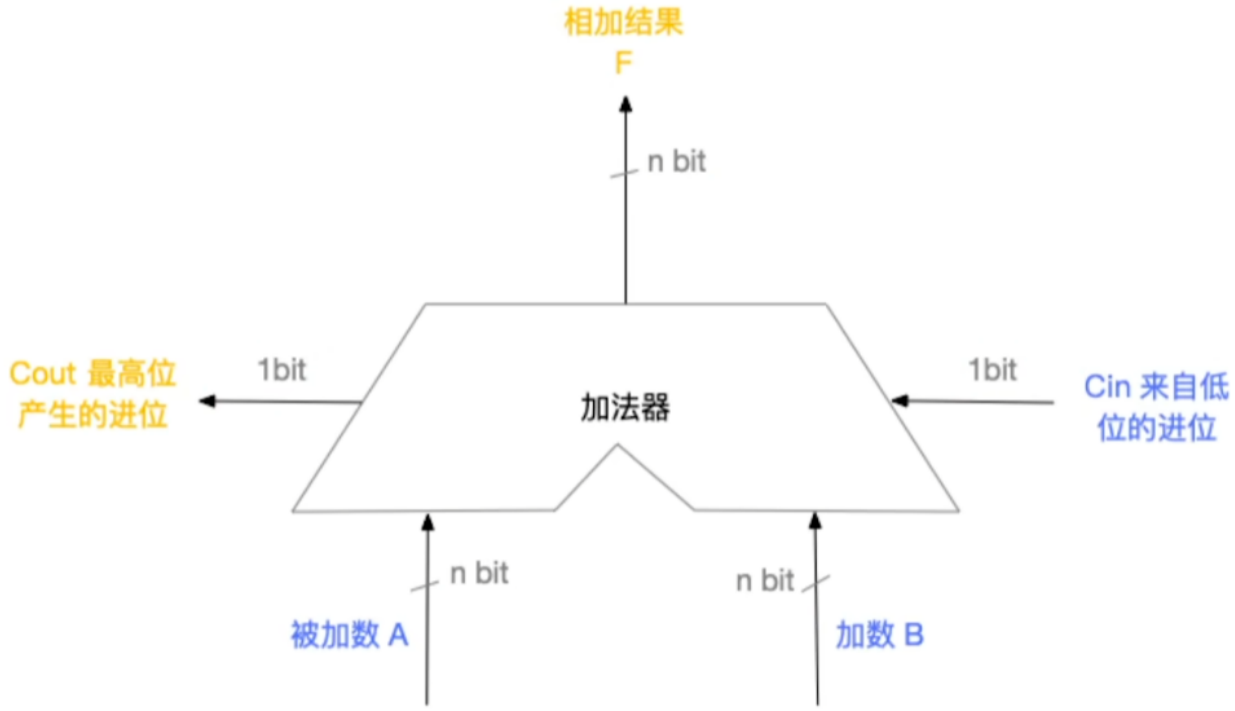
$C_i = A_i B_i + (A_i \oplus B_i) C_{i-1} = G_i + P_i C_{i-1}$

$C_1 = G_1 + P_1 C_0$
$C_2 = G_2 + P_2 C_1 = G_2 + P_2 G_1 + P_2 P_1 C_0$
$C_3 = G_3 + P_3 C_2 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_0$
$C_4 = G_4 + P_4 C_3 = G_4 + P_4 G_3 + P_4 P_3 G_2 + P_4 P_3 P_2 G_1 + P_4 P_3 P_2 P_1 C_0$
……

继续套娃会导致
电路越来越复杂

补码加减运算器

加法器原理



例：
A=1000，B=0111，Cin=0
则 F=1111， Cout=0

A=1000，B=0111，Cin=1
则 F=0000，Cout=1

补码加/减法运算方法

n bit补码 X + Y，按位相加即可

n bit补码 X − Y：将补码Y全部按位取反，末位+1，得到$[-Y]_{补}$，减法变加法

例1：4bit补码，X=-8，Y=7。$X_{补}$=1000，$Y_{补}$=0111

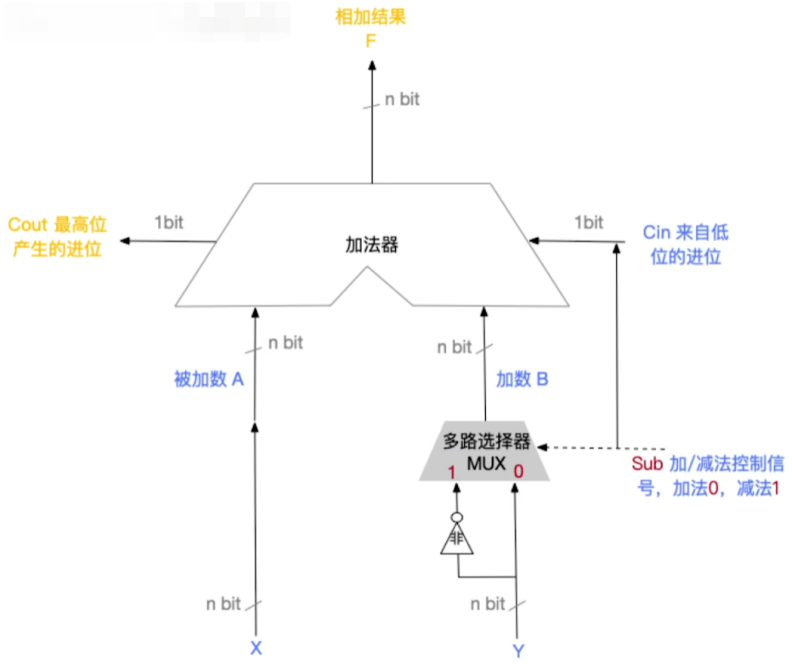
X+Y = 1111B

X-Y = 1000 + (1000+1) = 10001　　运算结果只保留低四位，最高位进位丢弃（发生溢出）

例2：4bit补码，X=3，Y=4。$X_{补}$=0011，$Y_{补}$=0100

X+Y = 0111B

X-Y = 0011 + (1011+1) = 1111B



例1：4bit补码，X=-8，Y=7
$X_{补}$=1000，$Y_{补}$=0111

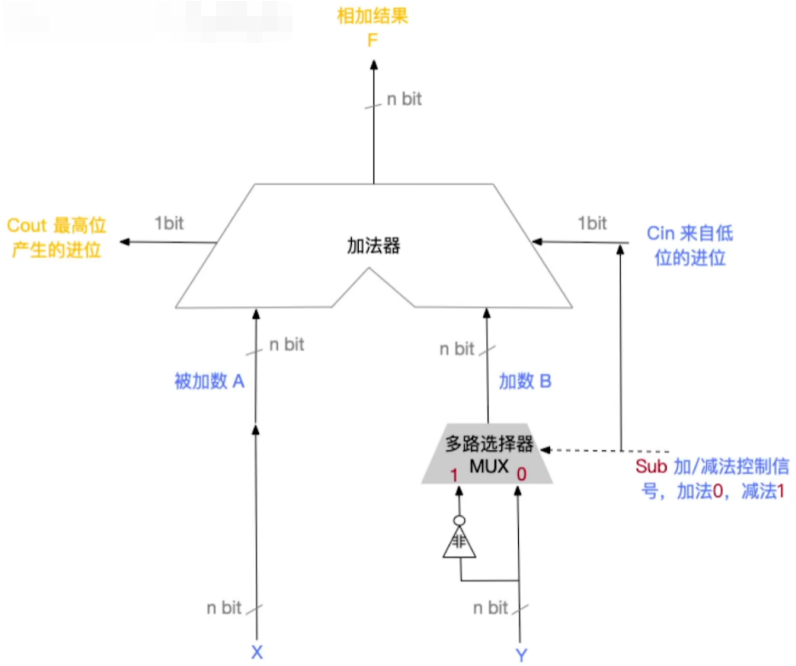
X+Y = 1111B = -1D ✅
X-Y = 1000 + (1000+1) = 10001=1D ❌
运算结果只保留低四位，最高位进位丢弃

例2：4bit补码，X=3，Y=4。$X_{补}$=0011，$Y_{补}$=0100

X+Y = 0111B = 7D ✅
X-Y = 0011 + (1011+1) = 1111B = -1D ✅

n bit补码 X + Y，按位相加即可

n bit补码 X − Y：将减数Y全部按位取反，末位+1，得到$[-Y]_{补}$，减法变加法



例1：无符号数 X=8，Y=7
用4bit表示，X=1000B， Y=0111B

X+Y = 1111B = 15D ✅
X-Y = 1000 + (1000+1) = 10001 = 1D ✅
运算结果只保留低四位，最高位进位丢弃

例2：无符号数 X=3，Y=4
用4bit表示，X=0011B， Y=0100B

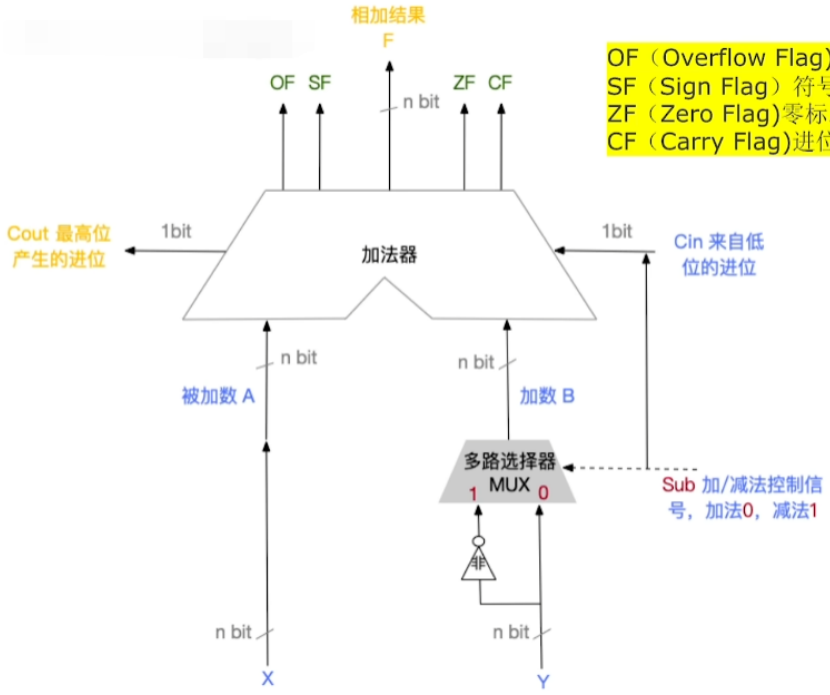
X+Y = 0111B = 7D ✅
X-Y = 0011 + (1011+1) =1111B= 15D ❌

无符号整数的加法/减法也可用该电路实现

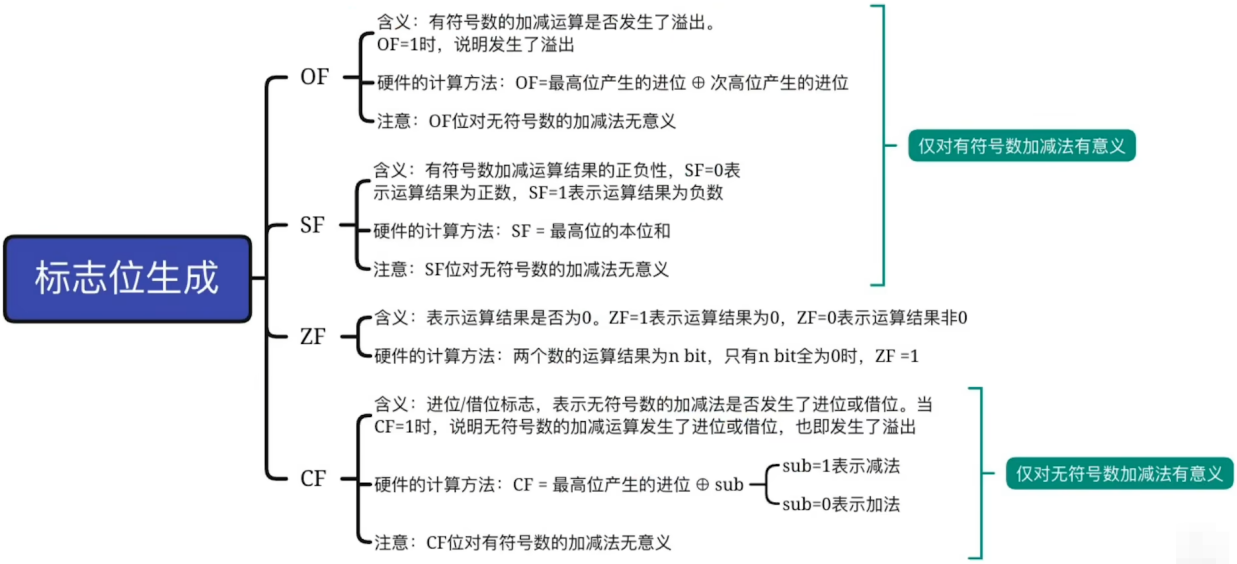
n bit无符号数 X + Y，按位相加即可

n bit无符号数 X – Y：将减数Y全部按位取反，末位+1，减法变加法

标志位的生成



OF（Overflow Flag)溢出标志。溢出时为1,否则置0。
SF（Sign Flag）符号标志。结果为负时置1,否则置0.
ZF（Zero Flag)零标志，运算结果为0时ZF位置1,否则置0.
CF（Carry Flag)进位/借位标志，进位/借位时置1,否则置0.


标志位生成
OF
含义：有符号数的加减运算是否发生了溢出。
OF=1时，说明发生了溢出
硬件的计算方法：OF=最高位产生的进位 ⊕ 次高位产生的进位
注意：OF位对无符号数的加减法无意义
SF
含义：有符号数加减运算结果的正负性，SF=0表示运算结果为正数，SF=1表示运算结果为负数
硬件的计算方法：SF = 最高位的本位和
注意：SF位对无符号数的加减法无意义
仅对有符号数加减法有意义
ZF
含义：表示运算结果是否为0。ZF=1表示运算结果为0，ZF=0表示运算结果非0
硬件的计算方法：两个数的运算结果为n bit，只有n bit全为0时，ZF =1
CF
含义：进位/借位标志，表示无符号数的加减法是否发生了进位或借位。当CF=1时，说明无符号数的加减运算发生了进位或借位，也即发生了溢出
硬件的计算方法：CF = 最高位产生的进位 ⊕ sub
sub=1表示减法
sub=0表示加法
注意：CF位对有符号数的加减法无意义
仅对无符号数加减法有意义