数字逻辑电路基础

南京大学
计算机科学与技术系
袁春风
email: cfyuan@nju.edu.cn
2015.6

布尔代数

- **关于0和1的一套数学运算体系起源于1850年前后英国数学家乔治•布尔（George Boole）的工作，因此称为布尔代数。**
  - **0和1分别代表逻辑值“假”和“真”**
  - **通过逻辑关系可以构建基于0和1的布尔代数运算**
  - **最基本的逻辑运算有：与（AND）、或（OR）、非（NOT），运算符分别为“•”（“∧”）、“+”（“∨”）、“¯”（“¬”）**

真值表：反映输入与输出之间的关系

A	B	A•B	A+B	$\overline{A}$	A⊕B
0	0	0	0	1	0
0	1	0	1	1	1
1	0	0	1	0	1
1	1	1	1	0	0

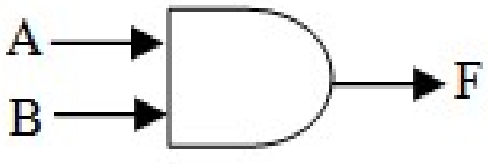
任何一种逻辑表达式都可写成这三种基本运算的逻辑组合。例如，异或（XOR）运算的逻辑表达式为：

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

异或运算也称不等价运算。

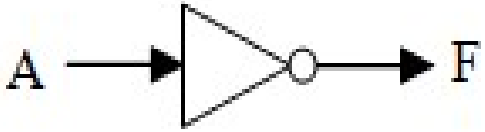
一位逻辑门电路

- **可通过逻辑门电路来实现逻辑运算**
  - **三种基本门电路：与门、或门、非门**
  - **其他门电路可以由三种基本门电路组合形成（如异或门电路）**

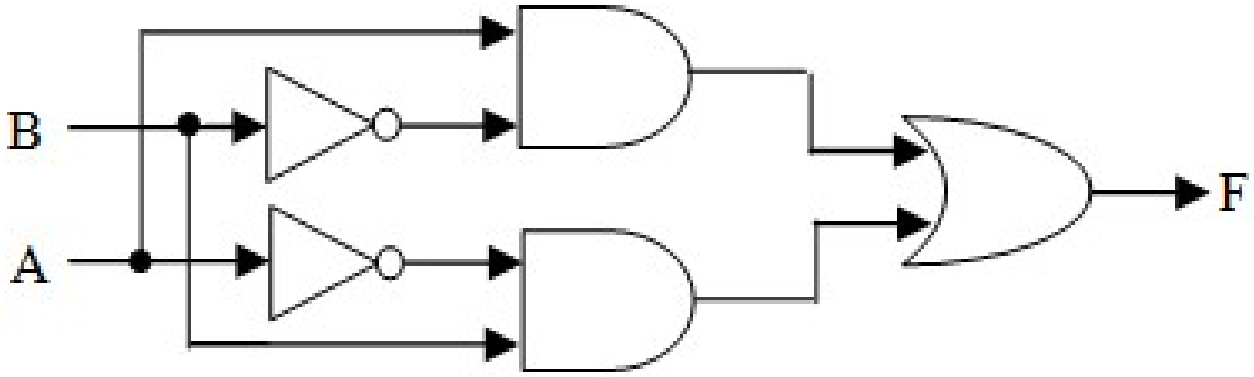


$F=A \cdot B$

$F=A+B$



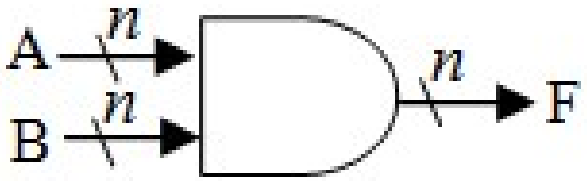
$F=\overline{A}$



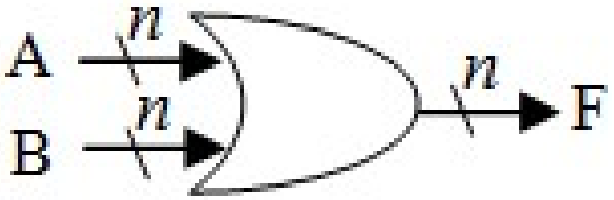
$F=\overline{A} \cdot B + A \cdot \overline{B} = A \oplus B$

n位逻辑门电路

- 对于n位逻辑运算，只要重复使用n个相同的门电路即可
  - 例如，若$A=A_{n-1}A_{n-2}\cdots A_1\ A_0$，$B=B_{n-1}\ B_{n-2}\cdots B_1\ B_0$，则与运算$F=A\bullet B$，实际上是按位相与，即$F_i=A_i\bullet B_i$（$0\le i\le n-1$）
  - 假定逻辑值位数为n，则按位与、按位或、按位取反、按位异或的逻辑符号如图所示



$F=A\bullet B$



$F=A+B$

$F=\overline{A}$

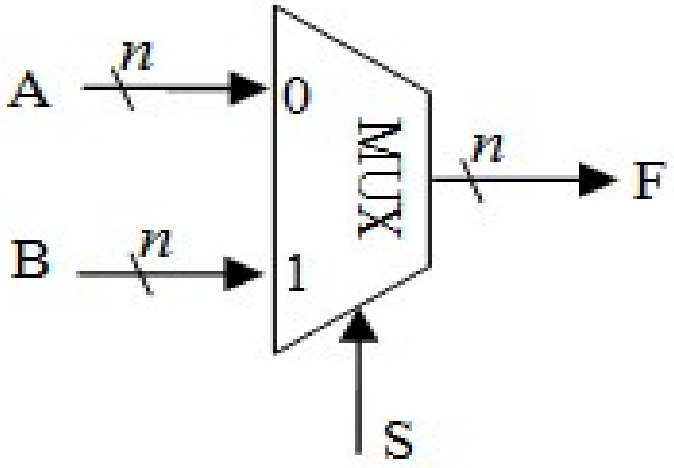
$F=A\oplus B$

组合逻辑部件

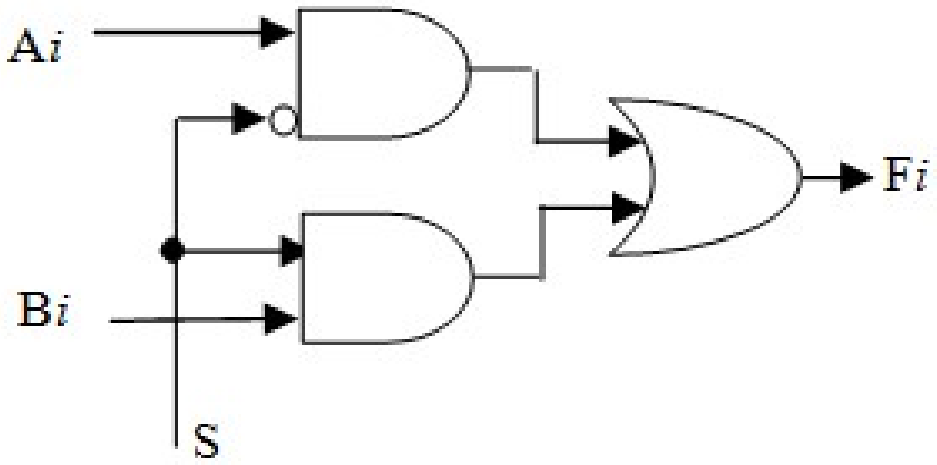
- 根据电路是否具有存储功能，将逻辑电路划分为两种类型
  - 组合逻辑电路：没有存储功能，其输出仅依赖于当前输入
  - 时序逻辑电路：具有存储功能，其输出不仅依赖于当前输入，还依赖于存储单元的当前状态
- 可以利用基本逻辑门电路构成一些具有特定功能的组合逻辑部件（功能部件）
  - 如译码器、编码器、多路选择器、加法器等
- 实现一个功能部件的过程
  - 用一个真值表描述功能部件的输入和输出之间的关系
  - 根据真值表确定逻辑表达式
  - 根据逻辑表达式实现逻辑电路

多路选择器

- 最简单的多路选择器（MUX）是二路选择器
  - 有两个输入端A和B，一个输出端F，并有一个控制端S，其功能是：当S为0时，F=A；当S为1时，F=B。
- k路选择器应有k路输入，因而控制端S的位数应是「$\log_2 k$」
  - 例如，三路或4路选择器，S有两位；5~8路选择器，S有3位。



二路选择器符号



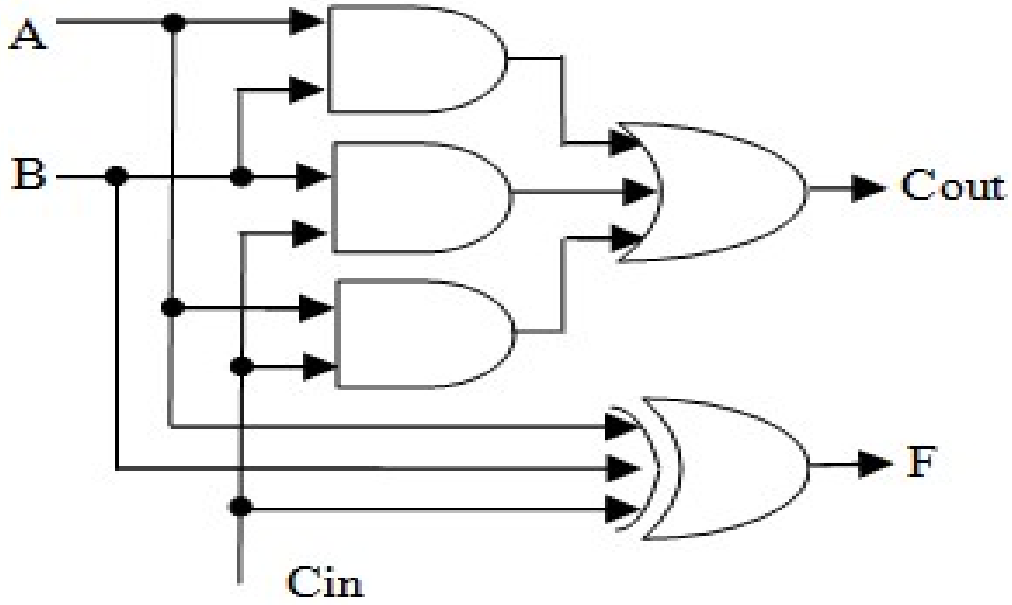
一位二路选择器逻辑电路

一位加法器（全加器）

- **一位加法器称为全加器**
  - **两个加数为A和B，低位进位为Cin，和为F，向高位的进位为Cout**
  - **化简后，逻辑表达式如下**

$$F=A\oplus B\oplus Cin$$

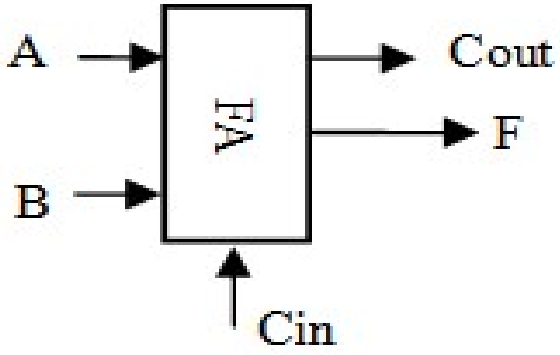
$$Cout=A\cdot B+A\cdot Cin+B\cdot Cin$$



全加器逻辑电路

A	B	Cin	F	Cout
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

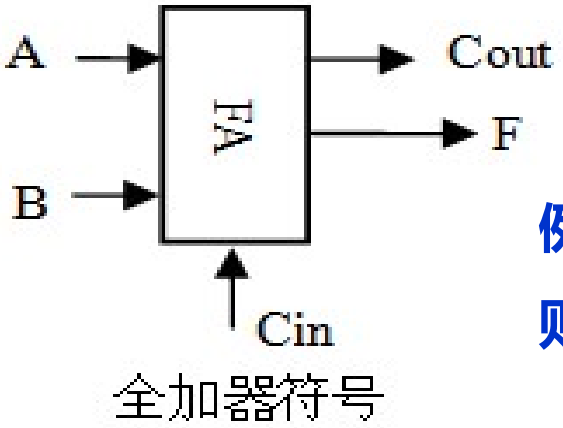
全加器真值表



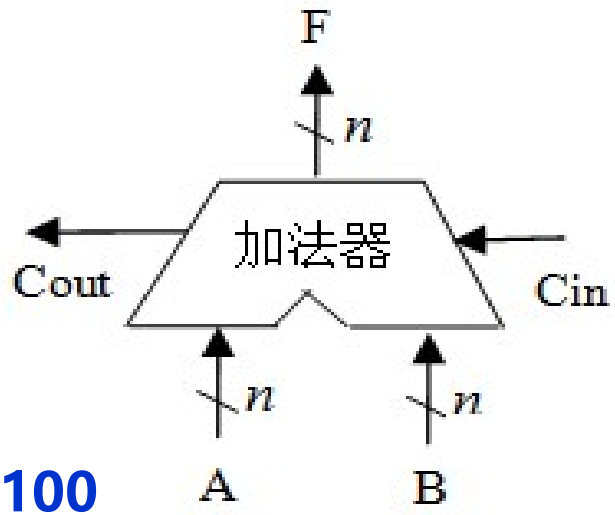
全加器符号

n位加法器

- **n位加法器可用n个全加器实现**



全加器符号

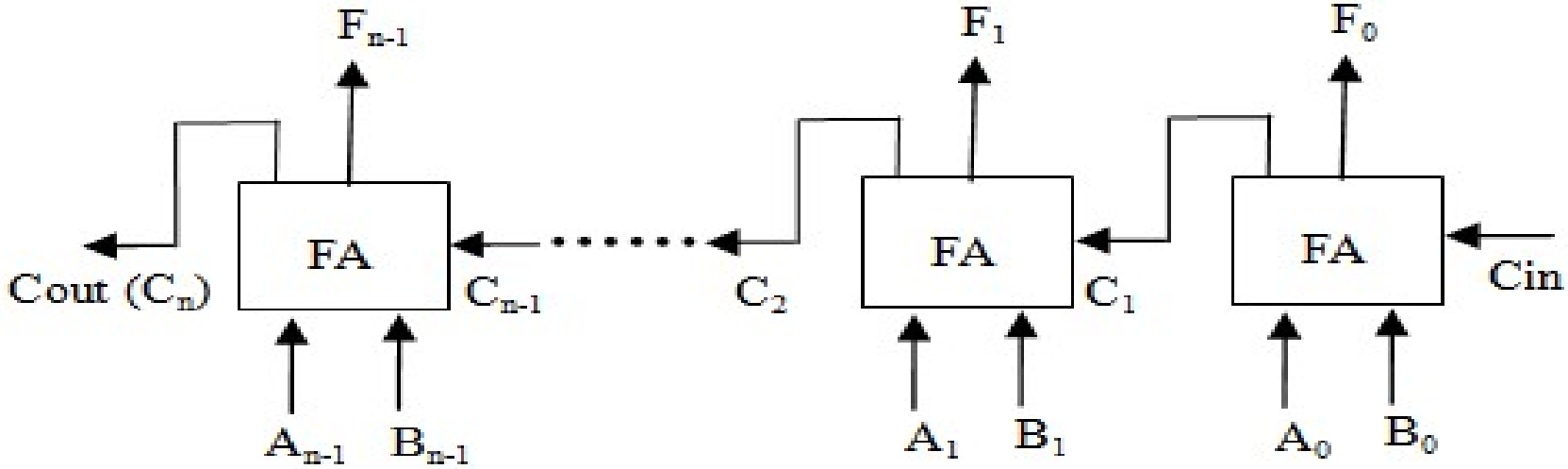


加法器符号

例：n=4，A=1001，B=1100

则：F=0101，Cout=1

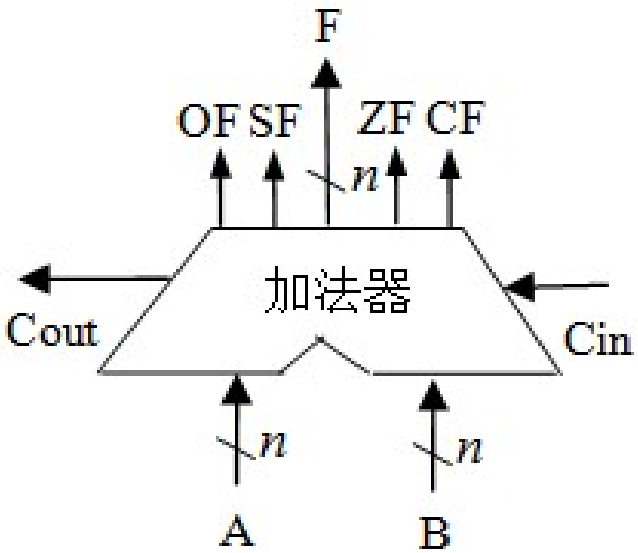
无符号整数加



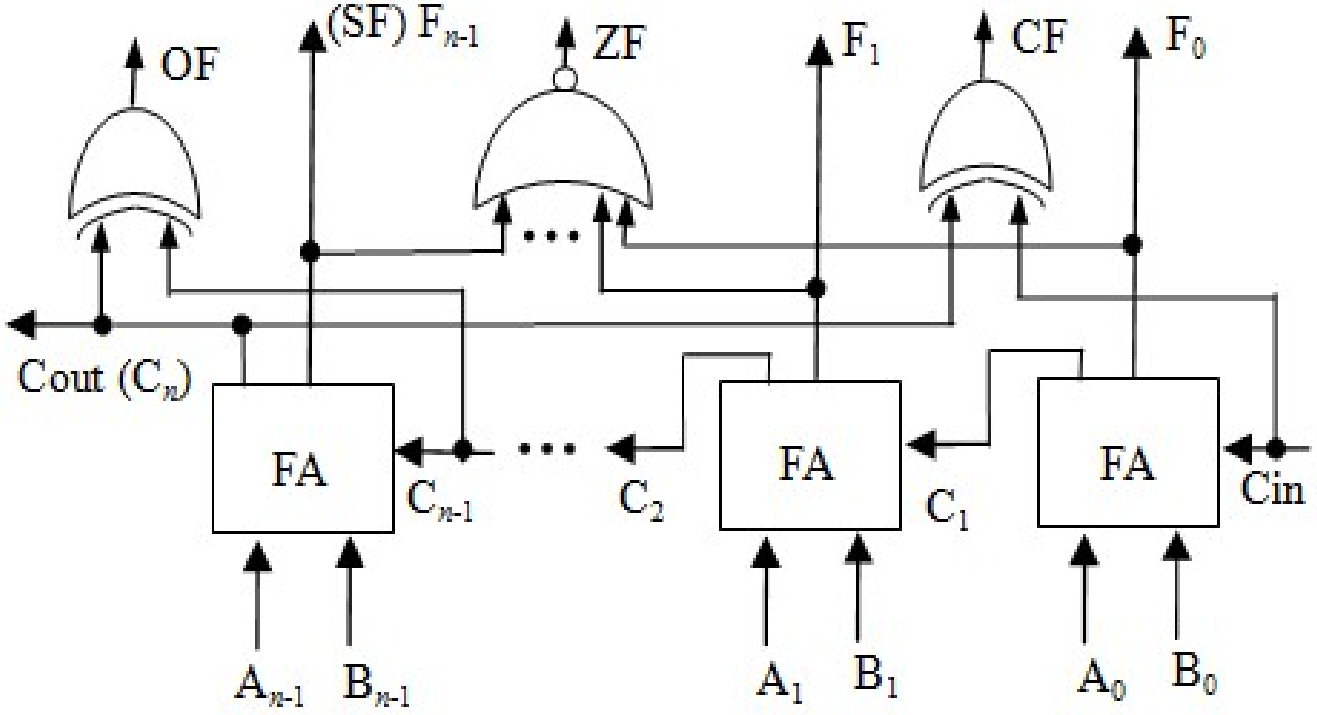
重要认识：加法由逻辑部件实现，而其他所有算术运算部件都基于加法器和逻辑运算实现，因此，所有算术运算是基于0和1以及逻辑运算实现的！

n位带标志加法器

- n位加法器无法用于两个n位带符号整数（补码）相加，无法判断是否溢出
- 程序中经常需要比较大小，通过（在加法器中）做减法得到的标志信息来判断



带标志加法器符号



带标志加法器的逻辑电路

溢出标志OF：

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

符号标志SF：

$SF=F_{n-1}$

零标志ZF=1当且仅当F=0；

进位/借位标志CF：

$CF=Cout \oplus Cin$

n位整数加/减运算器

先看一个C程序段：

```
int x=9, y=-6, z1, z2;
z1=x+y;
z2=x-y;
```

补码的定义　假定补码有n位，则：

$[X]_{补}=2^n+X \quad (-2^n \le X < 2^n, \mathrm{mod}\ 2^n)$

问题：上述程序段中，x和y的机器数是什么？z1和z2的机器数是什么？

回答：x的机器数为$[x]_{补}$，y的机器数为$[y]_{补}$；

z1的机器数为$[x+y]_{补}$；

z2的机器数为$[x-y]_{补}$。

因此，计算机中需要有一个电路，能够实现以下功能：

已知 $[x]_{补}$ 和 $[y]_{补}$，计算$[x+y]_{补}$ 和 $[x-y]_{补}$。

根据补码定义，有如下公式：

$[-y]_{补}=\overline{[y]}_{补}+1$

$[x+y]_{补}=2^n+x+y=2^n+x+2^n+y=[x]_{补}+[y]_{补} \ (\mathrm{mod}\ 2^n)$

$[x-y]_{补}=2^n+x-y=2^n+x+2^n-y=[x]_{补}+[-y]_{补} \ (\mathrm{mod}\ 2^n)$

n位整数加/减运算器

- 补码加减运算公式

$[A+B]_{补} = [A]_{补} + [B]_{补}\ (\mathrm{mod}\ 2^n)$

$[A-B]_{补} = [A]_{补} + [-B]_{补}\ (\mathrm{mod}\ 2^n)$

– 实现减法的主要工作在于：求$[-B]_{补}$

问题：如何求$[-B]_{补}$？

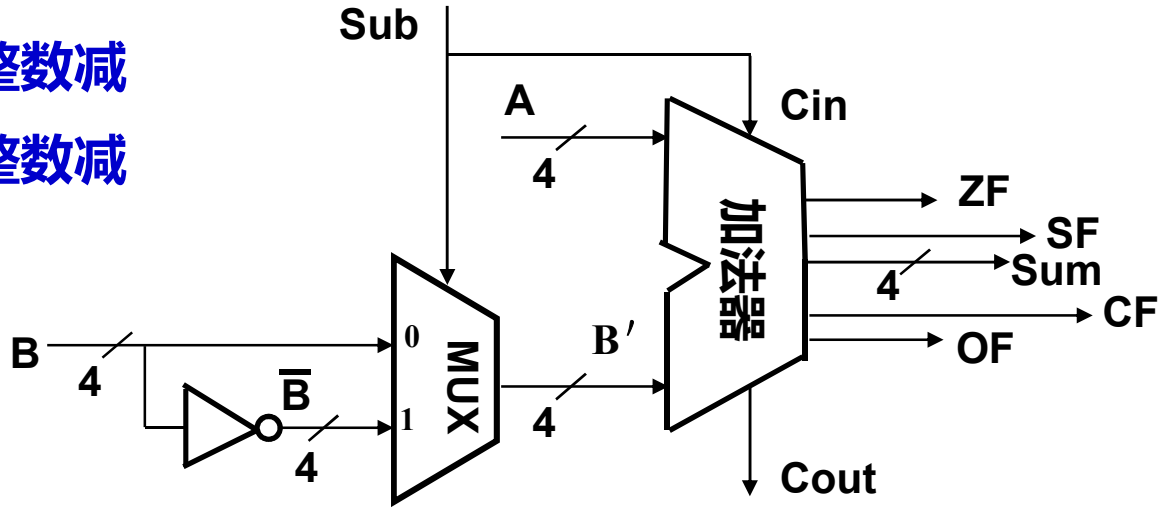
$[-B]_{补} = \overline{[B]}_{补} + 1$

- 利用带标志加法器，可构造整数加/减运算器，进行以下运算：

无符号整数加、无符号整数减

带符号整数加、带符号整数减

当Sub为1时，做减法
当Sub为0时，做加法

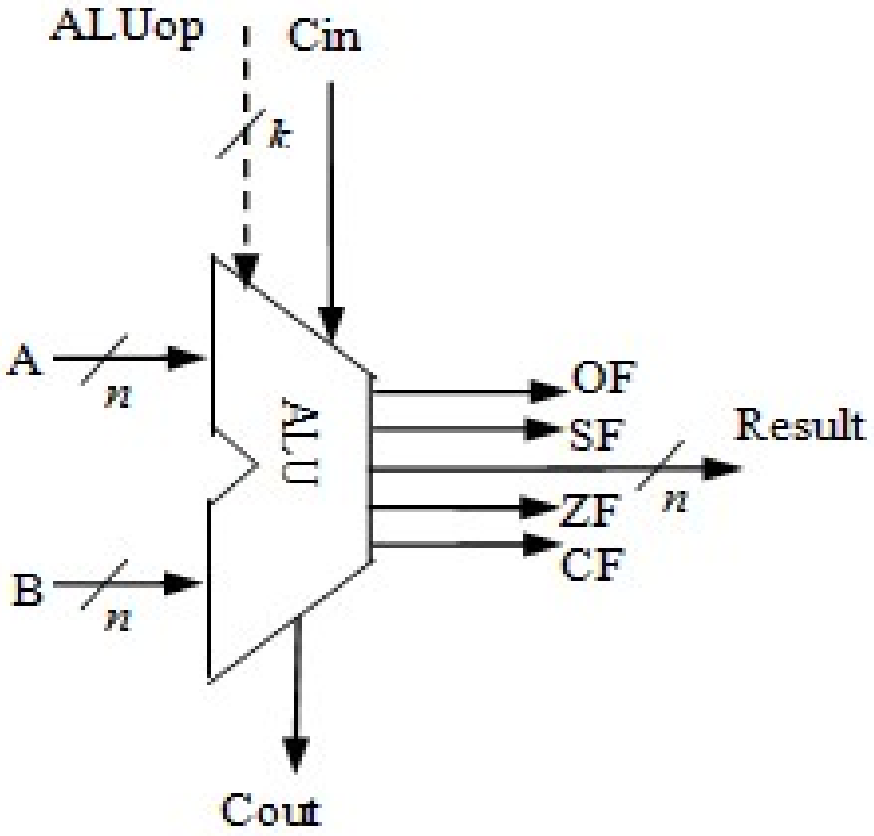


整数加/减运算部件

在整数加/减运算部件基础上，加上寄存器、移位器以及控制逻辑，就可实现ALU、乘/除运算以及浮点运算电路

算术逻辑部件（ALU）

- 进行基本算术运算与逻辑运算
  - 无符号整数加、减
  - 带符号整数加、减
  - 与、或、非、异或等逻辑运算
- 核心电路是带标志加法器
- 输出除和/差等，还有标志信息
- 有一个操作控制端（ALUop），用来决定ALU所执行的处理功能。ALUop的位数k决定了操作的种类，例如，当位数k为3时，ALU最多只有2^3=8种操作。



ALU符号

ALUop	Result	ALUop	Result	ALUop	Result	ALUop	Result
000	A加B	010	A与B	100	A取反	110	A
001	A减B	011	A或B	101	A⊕B	111	未用

回顾：认识计算机中最基本的部件

CPU：中央处理器；PC：程序计数器；MAR：存储器地址寄存器

ALU：算术逻辑部件；IR：指令寄存器；MDR：存储器数据寄存器

GPRs：通用寄存器组（由若干通用寄存器组成）