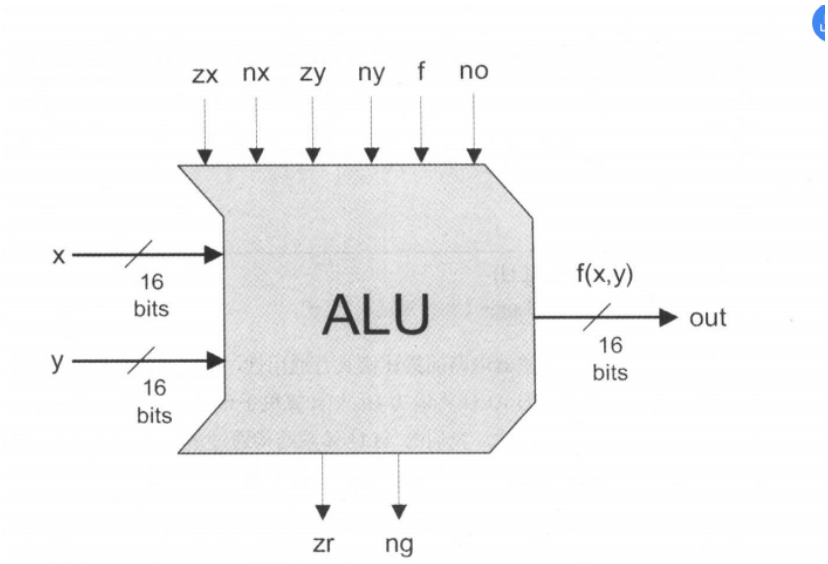
Chapter 2 Boolean Arithmetic

- *Counting is the religion of this generation, its hope and salvation
- 计算是这一代的信仰，希望和救赎
- 本章目标：ALU
- Background：
  - 二进制数
    - LSB：最右边一位
    - MSB：最左边一位
  - 二进制加法
    - 加和
    - 进位
  - 有符号二进制数
    - 2-补码/基补码
      - $$\bar{x} = \begin{cases} 2^n - x & if \quad x \neq 0 \\ 0 & otherwise \end{cases}$$
    - 按照这种编码方式，即使使用适用于正数的加法器，也可以得到正确的负数加减结果（利用模计算，易知）
    - 所有正整数首位为0，负整数首位为1
    - -x编码：
      - 所有最右边的0和左起第一个1不变，其余取反
      - 对x所有位取反，然后加1（逐位取反，末尾加一）
        - 2^n-x=(2^n-1)-x+1
- Specification
  - Adders加法器
    - Half-adder：进行两位加法
    - Full-adder：进行三位加法
    - Adder：进行两个n位加法
  - ALU：算术逻辑单元（Arithmetic Logic Unit）
    - Hack ALU
      - 门图



- 控制位——函数（对照表）

Hack 的 ALU 计算一组固定的函数 $out = f_i(x,y)$，这里 x 和 y 是芯片的两个 16-位输入，out 是芯片的 16-位输出，f_i 是位于一个函数表中的函数，该函数表由 18 个固定函数组成。我们通过设置六个称为控制位（*control bits*）的输入位来告诉 ALU 用哪一个函数来进行何种函数计算。图 2.5 给出了用伪代码表示的详细的输入/输出规范。

要注意的是，这 6 个控制位的每一位指示 ALU 来执行某个基本操作。这些操作的各种组合可以让 ALU 计算多种有用的函数。因为全部操作都是由 6 个控制位引起的，那么 ALU 可以对 2^6=64 个不同的函数进行操作。图 2.6 列出了这些函数中的 18 种。

这些位指示 如何预设 x 输入		这些位指示 如何预设 y 输入		此位用来选择 是+还是And	此位指示 如何设置out	生成的 ALU输出
zx	nx	zy	ny	f	no	out=
if zx then x=0	if nx then x=!x	if zy then y=0	if ny then y=!y	if f then out=x+y else out=x&y	if no then out=!out	f(x,y)=
1	0	1	0	1	0	0
1	1	1	1	1	1	1
1	1	1	0	1	0	-1
0	0	1	1	0	0	x
1	1	0	0	0	0	y
0	0	1	1	0	1	!x
1	1	0	0	0	1	!y
0	0	1	1	1	1	-x
1	1	0	0	1	1	-y
0	1	1	1	1	1	x+1
1	1	0	1	1	1	y+1
0	0	1	1	1	0	x-1
1	1	0	0	1	0	y-1
0	0	0	0	1	0	x+y
0	1	0	0	1	1	x-y
0	0	0	1	1	1	y-x
0	0	0	0	0	0	x&y
0	1	0	1	0	1	x|y

图 2.6　ALU 真值表。前六列二进制码表示的操作执行最右边列的函数（此处用符号!，&，和 | 来分别表示 Not，And 和 Or）。完整的 ALU 真值表包含 64 行，这里只列出了其中的 18 行

- 可以通过HDL来测试ALU的算术与逻辑功能

- 控制位：

        - zx：将x设为0
        - nx：将x按位not
        - zy，ny同上
        - f：接受以上输出
            - 1：计算x+y
            - 0：计算x&y
        - no
            - 1：计算！输出
            - 0：计算输出
        - out：输出
    - 类似于y-x的函数计算，非常之巧妙的利用了二进制的计算
    - 输出位
        - zr：out=0 true
        - ng：out<0 true
        - 在架构中很重要
- Implementation
    - Half Adder：sum与carry恰好等于Xor，And
        - Xor异或
        - 用逻辑设计实现数学计算
    - Full Adder：两个HA实现
    - 加法器：多个全加器
    - 增量器：增加1
    - ALU：Elegant
- Perspective
    - 加法器效率低下，可以通过进位预测来改进
    - 硬件软件的平衡

以上内容整理于 幕布文档