从另一个角度看补码

对于多数人谈到补码应该就会想到“补码 = 原码 取反 + 1”，这个是没有问题的，那么能不能从另外一个角度去看补码呢？大家想过没有1位的二进制原码可以表示 0 和 1，那一位的补码能表示什么？也是 0 和 1 ？实际上 1 位的补码能表示的是 0 和 -1，为何会这样，本文后面做解答。

在了解补码之前先了解一下码制和数制，以便于理解。

1、码制

常用的编码方式有独热码，格雷码，ASCII码等等，本质就是给事物或者状态起一个用于区别彼此的名字而已，常见的就是每个人的名字，大家都是人，所以需要一个名字或者身份证编码来进行区分，由此可知，码制编码的结果不能用来辨别大小，即不能说张三比李四大。

在数字电路中，码值一般用在状态机各个状态的编码，原因是很多时候状态机的各个状态并没有大小关系，而是需要一个编码来区别各个不同状态。原码的编码方式的符号位为 1 时表示这个数是负数，为 0 时表示这个数是正数，那么符号位的编码方式就是采用的码制编码。

2、数制

常用的数制编码有二进制、八进制、十进制、十六进制等等。需要明确权重的概念，对于多位数，处在某一位上的 “l” 所表示的数值的大小，称为该位的位权。例如十进制第 2 位的位权为 10，第 3 位的位权为 100；而二进制第 2 位的位权为 2，第 3 位的位权为 4，对于 N 进制数，整数部分第 i 位的位权为 $N^{(i-1)}$，而小数部分第 j 位的位权为 N^{-j}。n 位二进制整数，m 位二进制小数的权重分布如下：

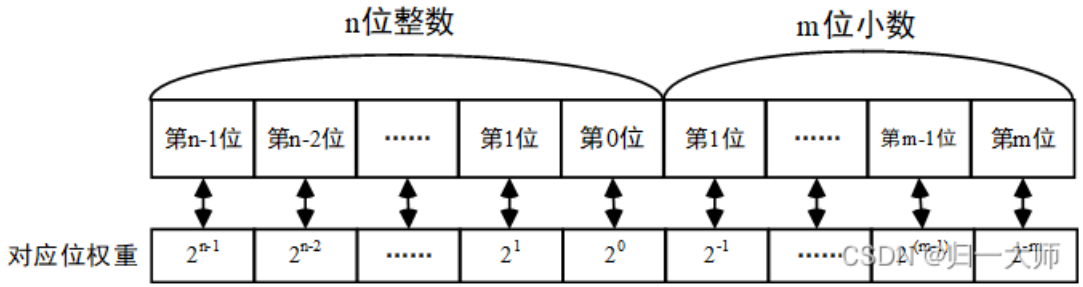

图1 二进制数的权重

3、原码

原码是最简单的有符号数编码方式，最高位是符号位，为 1 时表示该数是负数，为 0 时表示该数是正数。如下所示是 +5 和 -5 原码的二进制编码。

图2 +5的原码

图3 -5的原码

原码的 +5 与 -5 相加结果如下，得到的却是 -10，显然不是正确结果。分析原码的符号位是采用的码制进行编码，而数据位是采用的数制进行编码，但是码制是不能进行算术运算，所以结果会出错。

图4 原码+5与-5相加

如下图所示原码还会出现 +0 与 -0，造成编码浪费。

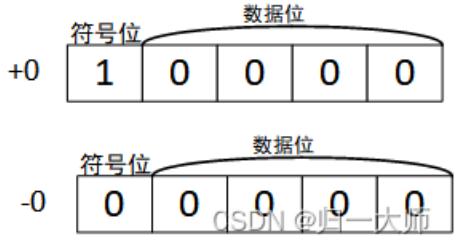

图5 原码中0的表示方式

4、补码

4.1、由原码推补码

用一种方法解除上面原码存在的问题，想要得到的计算结果当然是相反数相加结果得到 0，并且不会出现 +0 与 -0 这种现象。既然相加 +5 与 -5 相加结果为 0，将 0 和 5 的编码固定，利用 0 - 5 就可以推出 -5 的编码，过程如下图：

图6 二进制 0 减 5

得到 -5 的编码是 11011，11011 与 -5 的原码 10101 对比很容易得，11011 是原码符号位保持不变，数据位取反加 1 的结果。下图将 0 - 10 得到 -10 的编码，也可以得到相同的结果。

图7 二进制0减10

-10 的编码是 10110，10110 与 -10 的原码 11010 对比很容易得，10110 是原码符号位保持不变，数据位取反加 1 的结果。将这种编码称为补码，正数依旧与原码保持一致，负数的符号位与原码保持一致，数据位等于原码取反加一。

下图是补码的 -5 与 +5 相加的运算，可知相加结果为 0。

图8 补码 +5 与 -5 相加

4.2、补码符号位的权重

由图8运算知道，补码在进行运算时，符号位是参与运算的，那么说明补码的符号位就是采用数制进行编码的，进而可知补码的符号位是有权重的，权重具体十多少可以进行推测。

以上面得到 -5 的补码为 11011 例：

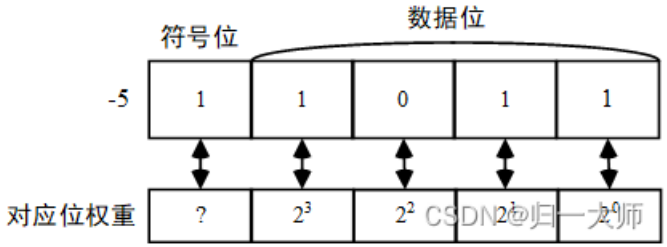

图9 补码权重分布

由此可得 1x？ + 1x 2^3 + 0x2^2 + 1x2^1 + 1x2^0 = ? + 8 + 2 + 1 = ？ + 11 =-5，得到 ? = -16=-2^4。

同理 -10 的补码为 10110，对应的对应位权重与上面分布一致，可以得到：1x？ + 0x2^3 + 1x2^2 + 1x2^1 + 0x2^0 = ? + 4 + 2 = -10，得到 ？ = -16 = -2^4。而 -5 与 -10 在此处的符号位都位于第 5 位，由此可以推出，对于 n 位补码数据，补码符号位的权重是 -2^{n-1}。

4.3、补码权重的应用——补码与十进制转换

大家为什么可以直接无符号二进制转换成十进制，或者十进制转成无符号二进制，那是因为知道二进制各个位的权重关系。以前大家如果想将十进制转成 二进制补码 ，或者将二进制补码转换成十进制，大多都会先将二进制补码或者十进制转换成原码，然后通过补码 = 原码取反 + 1 得到补码或者十进制数据，这也是计算机所使用的方式。

而如今已经知道了补码所有位的权重，n 位二进制补码，符号位权重为 -2^{n-1}，其余位的权重与无符号二进制一致。对于十进制与补码的转换就不再需要经过原码这个中间转换过程。

4.3.1、二进制补码转十进制

例如计算出补码 11011 对应的十进制数：

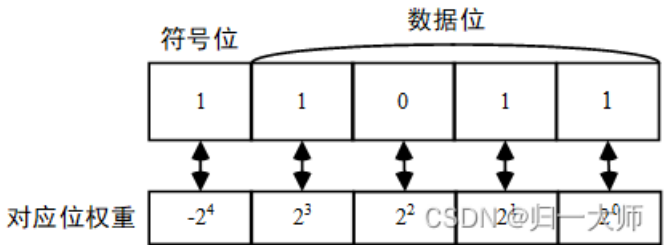

图10 5位补码权重分布

所以十进制为：$1x(-2^4) + 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0 = -16 + 8 + 2 + 1 = 5$

由上可知，二进制补码转十进制的方式与无符号数二进制转十进制方法一致，只是补码最高位符号位权重多一个负号而已。

至此，大家是否还记得文章开头的问题，一位的补码能表示的数据是什么？对于1为的数据无非是 0 或者 1，当为 0 时表示的就是 0，当为 1 时，此时这位数据作为补码的符号位，此时符号位的权重是 -2^0，即 -1。所以一位有符号数能表示的数据是 0 和 -1。

这还能解决平时书中会有答案，没有过程的问题，请问补码 1000_0000 对应的十进制数的大小是？

正常使用 原码 = 补码取反 + 1 时不进行位扩展将得不到正确答案，那如果知道符号位代表的权重，这不是送分题？

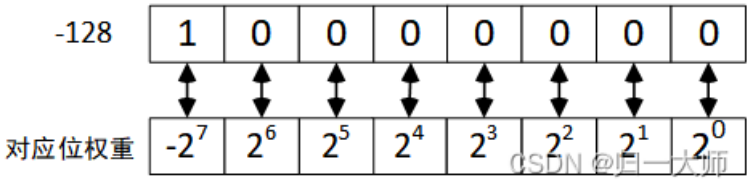

图11 补码1000_0000权重分布

最高位权重为 -128，其余为 0，所以 1000_0000 就是 -128。还可以知道这个数据是八位补码能表示的最小数据，因为此时正权重的数据位的值全部为 0。所以八位补码能表示的数据范围是 [-128,127]。n 位二进制能表示的数据范围是 $[-2^{n-1},(2^{n-1})-1]$ 也就不难理解了。

4.3.2、十进制转二进制补码

由于正数的补吗与原码一致，所以这里只拿负数进行举例，比如 -15，先考虑符号位，肯定负数最大值选择 -16，从而得到 -15 = -16 + 1，所以得到 -15 的补码为 10001 。在比如 -26，先考虑符号位，符号位最大值为 -32，得到 -26 = -32 + 4 + 2，从而写出 -26的补码为 100110。熟练之后简单的数可以直接写出其补码。

4.4、补码权重的应用——有符号数扩展

大家在使用FPGA或者其他数字电路对位宽比较敏感的器件时，对有符号数的计算，常常需要在计算之前先对位宽进行扩展，扩展的时候都是对符号位进行扩展，可能以前不是很理解，但是当你知道补码符号位权重之后就应该比较明白了。

对于位扩展，扩展前后必须保证大小不变，对于正数符号位为 0，扩展符号位大小肯定不会变，那么负数呢？

下面将 7 位的补码 1010101 扩展为 9 位补码：

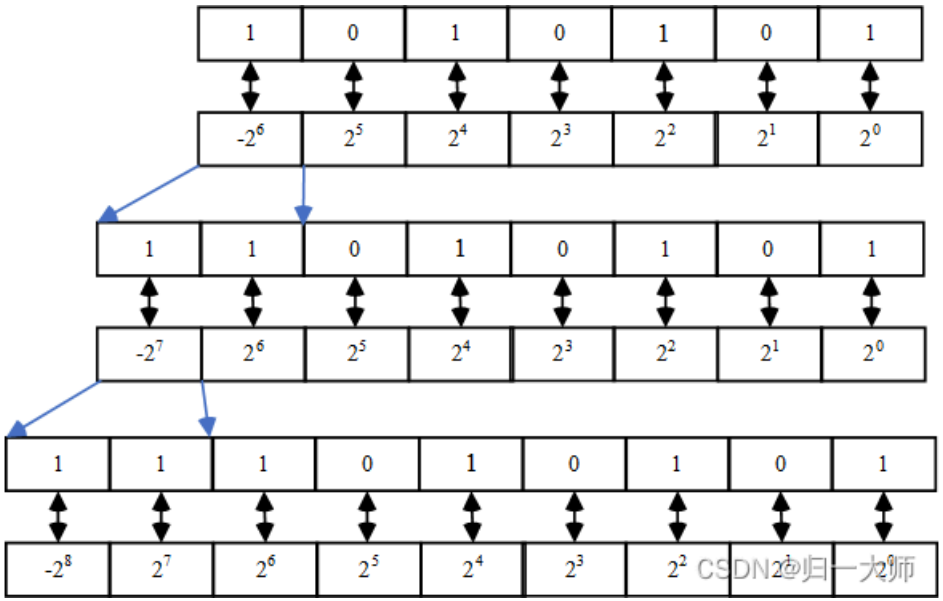

图12 补码符号位扩展时权重变化

由上图知七位扩展成八位时，只有最高两位发生了变化，其余位其实是没有变化的，所以只需要分析变化的这两位即可。7 位数据最高位的大小是 $1x(-2^6) = -2^6$，而扩展成八位时，最高位和次高位表示的大小为 $1x(-2^7) + 2^6 = (-2 + 1)x2^6 = -2^6$，所以大小保持不变。

将 8 位数据扩展为 9 位数据时也只有最高位两位发生变化，分析得到的结果也会跟上面一致。由于符号位权重的绝对值是最大的，并且是次高位的二倍，当 n 位的补码被扩展成 n+1 位时，n 位补码的符号位将变为 n+1 位补码数据位，此时 n+1 位的符号位数据与次高位权重之和还是等效于扩展前的 n 位补码符号位的权重，其余各位保持不变，所以在对符号位进行扩展之后，能够保证大小保持不变。

既然能够对数据进行扩展，那么也可以对数据进行压缩吧，同样思路就是如果补码从高位到低位，与符号位相同的数据都可以与符号位进行合并，直到遇到与符号位不同的数据为止，下图是将 1111011101 压缩为 1011101 的过程。

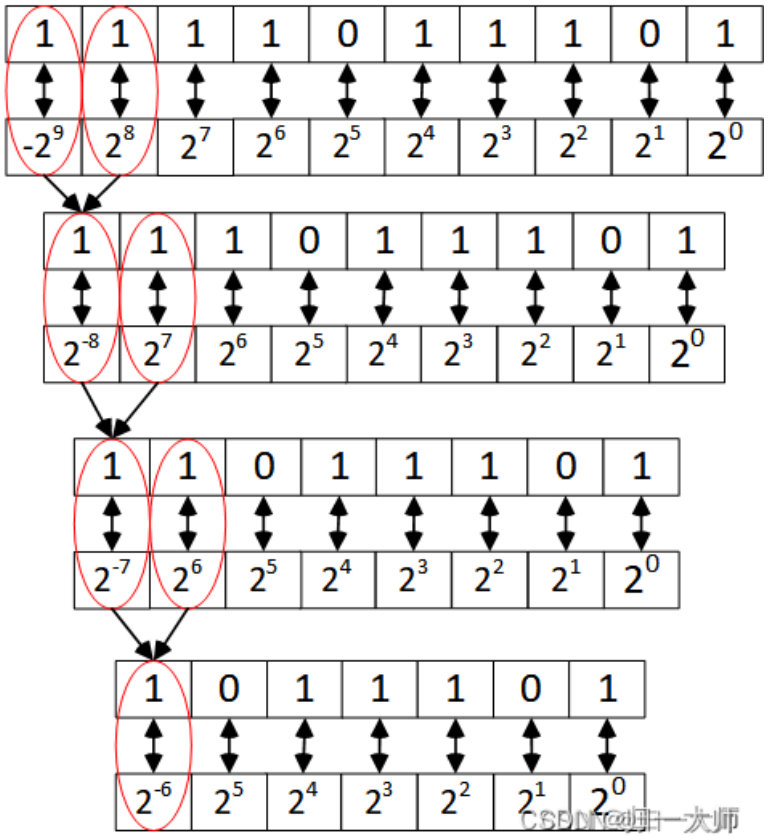

图13 补码高位相同时进行压缩

1111011101 的高四位全一的数据与权重相乘在求和运算的结果却与 1011101 的最高位权重是一致的，均为 -2^6，而其余位是相同的，故这两个数相等。

缩减其实也很好用，比如平时有人问你补码 1111110101 的数据对应的十进制数时，其实缩减后就是 10101 = -16 + 5，直接回答 -11 就行，根本不用计算。

5、总结

对于补码的理解其实是很重要的，不需要去死记书上所说的公式，能够根据原码的缺陷理解补码的编码规则，知道补码与原码本质区别，符号位编码方式的不同，从而得到补码符号位有权重，就可以用更简单的方式理解补码能表示的数据范围，以及补码对应位数能表示的最小值，有符号数位扩展的原理。

您的支持是我更新的最大动力！将持续更新工程，如果本文对您有帮助，还请多多点赞👍、评论💬和收藏⭐！