多二进制数同步超前进位相加符号和溢出性研究

刘 杰
(阜阳师范学院 物理系,安徽 阜阳 236032)

摘要:从两个二进制补码数相加和的符号和溢出性的判断方法入手,针对多个二进制数同步超前进位加法,定义了模2^{k+M}补码编码方案,提出了判断其加法和的符号和溢出性的方法,并进行了相应的证明.最后,就模2^{k+M}补码编码方案的实施进行了论述.该方案的提出为开发利用同步超前进位加法、转变微处理器的传统运算设计方案和提高计算机的计算速度提供了重要的理论依据.

关键词:二进制;符号;补码;同步超前进位加法;模2^{k+M}补码;溢出
中图分类号:TP334.23 **文献标识码**:A **文章编号**:1007-855X(2004)05-0083-04

A Study on the Sum Symbol and Overflow of Several Binary Numbers' Synchronous Look-ahead Carry Addition

LIU Jie
(Dept. of Physics, Fuyang Teachers College, Fuyang Anhui 236032, China)

Abstract: Referring to the method of judging the sum symbol and overflow of two binary complement code numbers' addition, a mod 2^{k+M} complement code scheme, and is defined a method of judging the symbol and overflow of the result about several binary numbers' synchronous lock-ahead carry addition, is brought forward which is proved. Finally how the scheme is applied is discussed. The scheme provides an important theory to develop and make use of the synchronous look-ahead carry addition, transform the traditional operation design scheme of processor, and heighten the computing velocity of computer on it.

Key words: binary system; symbol; complement code; synchronous look-ahead carry addition; mod 2^{k+M} complement code; overflow

0 引 言

在计算机内部,信息和数据的存取都是以离散状态的"0"和"1"的形式进行的,为了进行各种算术与逻辑运算,又常采用二进制形式来表示.为了更快更有效地对有符号的两个数进行加减运算,一般较多采用补码的编码方式.这种编码方式有两种方案:一种是模2^{k+1}补码编码方案,其定义为:

$$[x]_{补}=\begin{cases}x & 0\le x<2^k\\ 2^{k+1}+x & -2^k\le x<0\end{cases}\tag{1}$$

另一种是模补码编码方案,其定义为:

$$[x]_{补}=\begin{cases}x & 0\le x<2^k\\ 2^{k+2}+x & -2^k\le x<0\end{cases}\tag{2}$$

定义中的k为二进制数的整数部分数值位的位数.当$k=0$时,上面的补码定义便是定点小数的补码编码方案.不论哪一种补码表示法,都对符号进行了编码.在模2^{k+1}补码方案中,符号位是用一位的"0"和"1"分别表示数的正和负;而在模2^{k+2}补码方案中,符号位是用两位的"00"和"11"分别表示了数的正和

收稿日期:2004-02-25.
作者简介:刘 杰(1970~),男,硕士,讲师.主要研究方向:计算机硬件电路的开发与应用.**E-mail**:liujie52@ah163.com.

负.正是由于这种编码方案,使得在对两个数进行加减运算时,可以把符号位与数值位同等处理,只要结果不超出机器所能表示的数值范围,运算后的结果就可以按 2^{k+1} 或 2^{k+2} 取模,所得新结果就是本次加减运算的正确结果,其符号位和数值位仍是用补码表示.然而,当两个同号的数相加时,其结果可能会超出机器所能表示的数的范围,发生了错误,这就是所谓的结果"溢出".对于模 2^{k+1} 补码,其结果溢出判别方法是:当两个补码数实现加减运算时,若最高数值位向符号位的进位值与符号位产生的进位输出值不相同,则表明了运算结果出现了溢出,若相同,则表明运算结果正常;对模 2^{k+2} 补码,其结果溢出判别方法是:若运算结果的两位符号位不相同,即结果为01或10,则表明结果溢出;而若运算结果的两位符号位相同,即为00或11,则表明结果正常.可以看出,对模 2^{k+1} 补码方案,其结果溢出判定需要捕获两个动态进位值,而对模 2^{k+2} 补码方案,其结果溢出判定只需知道两个符号位的状态是否相同,在实现方面,模 2^{k+2} 更容易实现,因此采用模 2^{k+2} 补码方案将是首选的两数相加减的编码方案.

现在,如果参与加减运算的二进制数不是2个,而是3个、4个,或更多,那么它们的结果符号和溢出状态又如何判定呢?如果仍是采用两个数相加减的模 2^{k+1} 补码方案或模 2^{k+2} 补码办法,并对所有数按照从前到后依次相加减的办法一步步进行,且在每一次加减之后都进行结果判定,一旦有溢出就停止计算.那么,这样的计算既不能保证运算精度,也不能提高运算速度.比如有三个数相加,前两个正数相加的结果超出机器所能表示的数的范围,但其和与后一个负数相加,其结果不再超出机器所能表示的数的范围.可是,因为前面的计算溢出已经停止了后面的计算,造成整个的计算精度的降低.退一步说,假如前面的计算溢出不停止后面的计算,仍与后面的数相加,其符号位是否就能反映结果的符号或溢出呢?再者,一步步的加减也浪费了很多时间.如能采用多个二进制数的同步超前进位加法,并且结果的符号和溢出性的判定也能够一次性同时得出,这将大大提高计算机的运算速度和运算精度.那么,如何对多个二进制数的同步超前进位加法的结果情况进行一次性判定呢?下面将就这个问题提出模 2^{k+M} 补码办法进行结果的符号和溢出性的判定,并给出相应的证明.由于补码的减法运算可以转化为补码的加法运算,所以下面的论述中,仅提加法,而不再提减法.

1 多个二进制数相加的模 2^{k+M} 补码定义

针对模 2^{k+2} 补码的定义及其应用特性,可以找出一次性判定多个二进制数同步超前进位相加的结果符号和溢出性所使用的补码方案.假如有 n 个二进制数要同步相加,其符号位的个数至少必须等于 M,其中 M 应取大于或等于 $\log_2(n+1)$ 的最小整数值.比如,3个数同步相加,则

$$m \geq M = \log_2(3+1) = 2$$

即至少必须取两位符号位才足以判定三个数同步相加的符号和溢出性;再如5个数相加,则

$$m \geq M = \log_2(5+1) = 2$$

即至少必须取三位符号位就足以判定5个数同步相加的符号和溢出性.这样,n 个相加的 k 位整数数位的二进制数的补码可定义为:

$$[x]_{补} = \begin{cases} x & 0 \leq x < 2^k \\ 2^{k+M} + x & -2^k \leq x < 0 \end{cases} \tag{3}$$

其中 M 是取大于或等于 $\log_2(n+1)$ 的最小整数.这就是模 2^{k+M} 补码定义.

如:

$$x = +10101 \qquad [x]_{补} = \underbrace{0\cdots0}_{M}10101$$

$$x = -10101 \qquad [x]_{补} = \underbrace{1\cdots1}_{M}01011$$

有了上面的 n 个数同步相加的补码定义,当进行加法计算时,就可以把符号位看作数值位,并一起进行计算,最后根据结果的符号位状态判定结果的符号及溢出性.如果符号位为全"0",则表明结果为正数;如果符号位为全"1",则表明结果为负数;如果符号位即非全"0"也非全"1",则表明结果溢出,超出了计算机所能表示的数的范围.

2　证明模 2^{k+M} 补码能够判定多个数和的符号和溢出性

下面将根据模 2^{k+M} 补码定义，以及多个数同步相加可等价于各个数依次相加的加法性质，并按照数学归纳法的步骤，从而证明模 2^{k+M} 补码完全能够用于判定多个数和的符号和溢出性.

对于一个数不与任何数相加的情况，$M = 1$，即这种补码存在一位的符号位.由于它不与任何数相加，自然不存在溢出的问题，仅用一位符号的变化状态就能够表达这个数的正与负.

对于两个数相加的情况，$M = 2$，即这种补码存在两位的符号位.当两个正数相加时，如果无最高数值位的进位，则和的符号位为00，表明了运算结果为正数的补码；而如果最高数值位产生了进位，则符号位由00变成了01，表明了运算结果的溢出.当两个负数相加时，如果无最高数值位的进位，则和的符号位将由11变成10，表明了运算结果的溢出；而如果最高数值位产生了进位，则符号位仍为11，表明了运算结果为负值.当两个互异的数相加时，无论最高数值位是否产生进位，其结果都只能是00或11，这表明了无溢出产生，也完全符合了数的内在运算本质.从上面的两个补码数相加的现象可以得出这样的结论：两个正数相加，如果结果的符号位本应为00却变成了01，则表明了数的溢出，实质上是由于最高数值位向符号位进了1；而两个负数相加时，如果结果的符号位本应为11，却变成了10，则表明了数的溢出，相当于负数的符号位被减去了1.因而用两位符号位的01和10两个状态完全可以表达出两个数相加的所有溢出状态.

对于三个数相加的情况，$M = 2$.与两个数相加的情况雷同，同样是把符号位的00表示正数，11表示负数，01、10表示溢出，唯一要讨论的情况有两种.① 当两个正数的溢出和与第三个数相加时，如果第三个数为正数，则相加结果的符号位只有两种状态：一种是没有最高数值位的进位，而保持原有的01状态；另一种是因有了最高数值位的进位而变成了10状态.但是，这两种状态都仍是溢出状态，只是溢出的深度不同而已.如果第三个数为负数，则相加的结果也存在两种状态：一种是因没有最高数值位的进位而变成00状态；另一种是因有了最高数值位的进位而变成01状态.这里的00状态表明了三个数相加，其和为正值，没有发生溢出.尽管前两个数之和发生了溢出，但是，因为第三个数为负数，减少了前两个数的和的范围，使得整个结果满足了机器要求.而01状态，只是由于第三个数对原溢出和影响太小，未能改变原溢出状态.② 当两个负数的溢出和与第三个数相加时，如果第三个数为负数，则相加结果的符号位存在两种状态：一种是因没有最高数值位的进位而变成01状态，另一种是因有了最高数值位的进位而仍保持为10状态.这两种状态，都表明了溢出，只是溢出的深度不同.如果第三个数为正数，则相加结果也存在两种状态：一种是因没有最高数值位的进位而保持为10状态；另一种是因有了最高数值位的进位而变成11状态.这里的11状态表明了两个负数的溢出和经第三个正数中和，可以变回正常状态.从上面的分析可以看出，一次正数相加产生进位，使符号位由00变成了01，发生了溢出，当再加上一个正数并产生进位时，符号位又由01变成了10，仍然还是溢出；一次负数相加未产生进位，符号位由11变成了10，发生了溢出，相当于对原符号位11进行了减1，当再加上一个负数并且还无进位时，符号位又由10变成了01，仍然还是溢出，相当于又一次对原符号位10进行了减1.从这些变化可以看出，无论什么样的三个数相加，其符号位的连续变化最多只有两次，符号位都不会从一个正常状态(如正数符号位的表示状态00)转到另一个正常态(如负数符号位的表示状态11).可见，只需要两位的符号位就足以判定三个数相加的结果情况.

参照上面的分析，假设模 2^{k+M} 补码对判定 $n-1$ 个数相加的结果是成立的，那么，结果的符号位状态至少必须包含有 n 种形式，包括用全0和全1分别表示正负的两个正常态，以及 $n-2$ 个溢出状态.下面再分析 n 个数相加的情况.

对于 n 个数相加的情况，实质上就是前 $n-1$ 个数的和再与第 n 个数相加.由于 $n-1$ 个数相加与 n 个数相加所采用的补码符号位的个数可能不同，按照补码定义，可以使用 n 个数相加所使用的补码符号位个数来编码 $n-1$ 个数的符号位(比如，7个数相加的编码可以使用8个数相加所使用的4位符号位).这样，对于前 $n-1$ 个数的和来说，符号位为全0表示正数和，为全1表示负数和，为0…001仅表示正数和的溢出，为1…110仅表示负数和的溢出，为0…010 ~ 1…101则既可能表示正数和的溢出，也可能表示负数和的溢出，不存在符号位为0…001表示负数和的溢出，为1…110表示正数和的溢出的情况.

1) 对于前 $n-1$ 个数的相加和为正数的非溢出情况,当再与第 n 个数相加时,如果第 n 个数是正数,则结果可能存在两种情况:一种是最高数值位无进位的情况,这时的符号位仍为全0,表示正数;另一种是最高数值位有进位的情况,这时的符号位变为0…001,表示结果的溢出.如果第 n 个数是负数,则结果可能存在两种情况:一种是最高数值位无进位的情况,这时的符号位变为全1,表示结果为负数;另一种是最高数值位有进位的情况,这时的符号位仍为全0,表示结果为正数.

2) 对于前 $n-1$ 个数的相加和的符号位为0…001的正数溢出的情况,当再与第 n 个数相加时,如果第 n 个数是正数,则结果可能存在两种情况:一种是最高数值位无进位的情况,这时的符号位仍为0…001,表示溢出;另一种是最高数值位有进位的情况,这时的符号位变溪0…010,表示结果溢出更严重.如果第 n 个数是负数,则结果可能存在两种情况:一种是最高数值位无进位的情况,这时的符号位变为全0,表示结果为正数,无溢出;另一种是最高数值位有进位的情况,这时的符号位仍为全0…001,表示结果仍然溢出.

对于前 $n-1$ 个数的相加和的符号位为0…001的情况,因不存在负数溢出的情况,自当不必讨论.

3) 对于前 $n-1$ 个数的和为负数的非溢出情况,当再与第 n 个数相加时,如果第 n 个数是正数,则结果可能存在两种情况:一种是最高数值位无进位的情况,这时的符号位仍为全1,表示结果为负数,无溢出;另一种是最高数值位有进位的情况,这时的符号位变为全0,表示结果为正数,无溢出.如果第 n 个数是负数,则结果可能存在两种情况:一种是最高数值位无进位的情况,这时的符号位变为1…110,表示结果为负数溢出;另一种是最高数值位有进位的情况,这时的符号位仍为全1,表示结果为负数.

4) 对于前 $n-1$ 个数的和的符号位为1…110的负数溢出的情况,当再与第 n 个数相加时,如果第 n 个数是正数,则结果可能存在两种情况:一种是最高数值位无进位的情况,这时的符号位仍为1…110,表示结果仍然是负数溢出;另一种是最高数值位有进位的情况,这时的符号位变为全1,表示结果为负数,无溢出.如果第 n 个数是负数,则结果可能存在两种情况:一种是最高数值位无进位的情况,这时的符号位变为1…101,表示结果为负数溢出;另一种是最高数值位有进位的情况,这时的符号位仍为全1…110,表示结果仍然溢出.

对于前 $n-1$ 个数的和的符号位为1…110的情况,因不存在正数溢出的情况,自当不必再讨论.

5) 对于前 $n-1$ 个数的和的符号位为0…010 ~ 1…101的溢出情况,无论与之相加的第 n 个数是正数还是负数,都不可能改变其结果的溢出性.

从上面的分析可以得出结论,模 2^{k+M} 补码方案完全能够用于判定多个数和的符号和溢出性.

3 模 2^{k+M} 补码编码方案的实施

根据模 2^{k+M} 补码定义,由于一个正确的数的 M 个符号位的值总是相同的,所以在除加法器部分以外的所有可存储的空间,其数的存储都可以按一个符号位进行,不改变外部数的存储形式,只有在加法器中 M 个符号位的使用才是必要的,这样就可以保证硬件的升级,软件的兼容,提高计算机的运算速度.下面举一例说明之.按照计算机的常规设计理念,可假设一个CPU能够对多达8个的二进制数进行同步超前进位相加.当然,这样的CPU还必须有相应的指令.根据模 2^{k+M} 补码定义,8个的二进制数的符号位需要取4位.按照指令,当进行数的相加时,每一个数的符号位的值都要同时送到加法器中的相应寄存器的4位符号位的输入端.经过计算,得出带4位符号位的和数.通过4位符号位的值,可以确定所得结果是正还是负,以及是否溢出.若溢出,则转入溢出处理,反之,则把一位的符号位和数值位回送到存储单元.

4 结束语

当采用模 2^{k+2} 补码定义对 n 个二进制数进行连续分别加法时,只要出现一次溢出,计算机将停止运算;而当采用模 2^{k+M} 补码定义对 n 个二进制数进行同步超前进位加法时,虽然其中可能有某些数如果单独相加会出现溢出,但是,因为结果的判定是在 n 个数同步运算结束之后进行的,所以这种溢出现象并不影响整体结果的判定,从而提高了计算机的运算适应性和相对精确度.尽管现代的计算机硬件内部仍采用两个

(下转第91页)

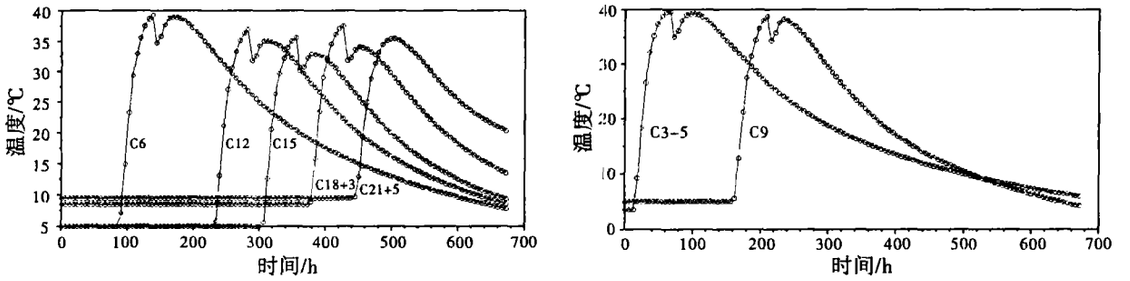

图3　边缘点的温度变化曲线

Fig. 3　The curve of temperature fluctuation of border poured cube

3　结　论

本文用 ADINA 程序对实际的工程进行混凝土浇筑温度场的仿真分析，得出以下结论：

1) ADINA 程序只要定义出“时间函数”、“生死时间”和“到达时间”就能实现对混凝土浇筑过程的模拟，是比较简单而实用的.

2) 通过仿真分析，在进行施工之前，就能预见结构物温度场的变化情况.可以客观的评价所制定的施工方案是否合理.

3) “薄层短间歇期”是大体积混凝土施工中常用的施工方法，但“薄”到什么程度？“短间歇”到什么程度？事先是未知的.通过仿真分析，并考虑温度应力，可以得到“薄”和“短”的平衡点，方便进行合理的温控设计.

参考文献：

[1] 龚召熊，张锡祥，肖汉江，等.水工混凝土的温控与防裂[M].北京：中国水利水电出版社，1999.105～111.

[2] 朱伯芳.大体积混凝土温度应力与温度控制[M].北京：中国电力出版社，1998.12～13.

[3] 赵代深，薄钟禾，李广运等.混凝土重力坝的温度应力[M].土木工程学报，1993，26(4)：29～39.

(上接第 86 页)

二进制数的超前进位加法，未能开发使用多个二进制数的同步超前进位加法.但是，大量的天文数据，气象数据，水文数据等数据处理已迫使人们去研制高速高效的超大规模计算机.在许多国家都出现了多处理器、可并行进行数据处理的超大规模计算机.可以想象，如果开发使用多个二进制数的同步超前进位加法作为计算机的核心运算部件，那么就可以减少并行处理器的数量，提高计算机的运算速度.这样以来，采用模 2^{k+M} 补码编码方案对多个二进制数同步超前进位加法结果的特性判定将是必选的、快速的、有效的使用方案.

参考文献：

[1] 王爱英.计算机组成与结构.北京：清华大学出版社，1990.12～51，73～87.

[2] 潘名莲，马争.微计算机原理.北京：电子工业出版社，1994.21～24.

[3] 张友德，赵志英.单片微型机原理、应用与实验.上海：复旦大学出版社，1992.3～19.

[4] 周明德.微型计算机硬件软件及其应用(第 2 版).北京：清华大学出版社，1988.2～13.

[5] 徐淑华，程退安.单片微型机原理及应用(第 2 版).哈尔滨：哈尔滨工业大学出版社，1994.23～41

[6] 郑学坚，朱善君.微型计算机原理及应用.北京：清华大学出版社，1987.1～14.

[7] 席先觉，李忠民.单片微型计算机及其应用.北京：高等教育出版社，1989.12～29