低延时的浮点正弦余弦函数硬件实现算法

梁峰[1]，刘春锐[2]，李孝聪[1]，邱广波[2]，张继[2]，陈振娇[2]，李薇敏[2]，曹琪[1]，雷绍充[1]
（1. 西安交通大学微电子学院，710049，西安；2. 中国电子科技集团公司第五十八研究所，214035，江苏无锡）

摘要：针对正弦余弦函数硬件实现中坐标旋转数字计算机(CORDIC)算法难以满足低延时、高精度的不足之处，提出了一种单精度浮点数的正弦余弦函数硬件实现算法。该算法基于正弦余弦函数的数学性质把浮点数输入指数的范围分为[－126，－16]、[－15，21]和[22，126]3 个不同区域，并分别采用泰勒 0 阶近似法、泰勒 1 阶近似法和直接计算 3 种计算方法；对其中的泰勒 1 阶近似法进行了定点化、诱导函数化简、预计算输出指数等多种优化来减少算法的计算量，增加并行度。给出了优化后的泰勒 1 阶近似法的 4 级流水线硬件实现。通过遍历测试，该算法的计算精度与 C 语言 math 库单精度正弦余弦函数的计算精度最多仅相差 1 个误差单位(ulp)。在 UMC55nm 的工艺下电路可达 250 MHz 的时钟频率，完成一次运算仅需 4 个时钟周期，具有低延时的特点。

关键词：正弦余弦函数；泰勒 1 阶近似；坐标旋转数字计算；硬件实现
中图分类号：TN402 **文献标志码**：A
DOI：10.7652/xjtuxb202111012 **文章编号**：0253-987X(2021)11-0106-09

OSID 码

A Low Latency Floating-Point Sine and Cosine Function Hardware Implementation Algorithm

LIANG Feng[1], LIU Chunrui[2], LI Xiaocong[1], QIU Guangbo[2], ZHANG Ji[2],
CHEN Zhenjiao[2], LI Weimin[2], CAO Qi[1], LEI Shaochong[1]
(1. School of Microelectronics, Xi'an Jiaotong University, Xi'an 710049, China;
2. The 58th Research Institute of China Electronics Technology Group Corporation, Wuxi, Jiangsu 214035, China)

Abstract: In order to solve the problem that coordinate rotation digital computer (CORDIC) algorithm is difficult to meet the requirement of low latency and high accuracy in hardware implementation, a low latency floating-point sine and cosine function hardware implementation algorithm is proposed. This algorithm divides the range of the input floating-point number's exponent into 3 regions of [－126, －16], [－15, 21] and [22, 126] based on the mathematical properties of sine and cosine. In the three regions, Taylor 0-order approximation, Taylor 1st-order approximation, and direct calculation methods are used, respectively. Taylor 1st-order approximation method is optimized by converting to fixed-point, induction formula reduction, and precomputing output exponent to reduce calculation complexity and improve parallelism. A 4-stage pipeline hardware implementation of the optimized Taylor 1st-order approximation is provided. By the way of traversal test, the accuracy of the proposed algorithm is only 1 ulp (unit in the last place) away from that of the C-language math library at most. In UMC55nm process, the circuit can reach 250 MHz clock frequency and the total latency per operation is only 4 clock

收稿日期：2021-03-06。 作者简介：梁峰(1976—)，男，副教授；雷绍充(通信作者)，男，教授，博士生导师。 基金项目：国家自然科学基金资助项目(61474093)。
网络出版时间：2021-07-16 网络出版地址：http://kns.cnki.net/kcms/detail/61.1069.T.20210716.0927.002.html

cycle, which means the property of low latency.

Keywords: sine and cosine function; Taylor first-order approximation; coordinate rotation digital computer algorithm; hardware implementation

具有正弦余弦函数计算能力的集成电路已被广泛应用于各个领域。通用芯片 CPU、GPU 和 DSP 中也有专门进行正弦余弦函数运算的硬件单元，例如 TI 的 TMX320 系列 DSP 芯片，其上有一个硬件纹理映射单元(TMU)，专门用于实现超越函数的快速计算。

传统的正弦余弦函数的硬件实现，主要采用坐标旋转数字计算机(CORDIC)算法[1]。CORDIC 算法的优势在于电路面积小，仅用移位寄存器和加法器/减法器就可以实现计算。但其缺点在于，它的收敛速度较慢，通常一轮迭代只能增加一个有效数字。对于单精度浮点运算，CORDIC 算法需要较多的时钟周期才能使结果达到单精度浮点所要求的精度。基于此，有许多改进的 CORDIC 算法如 Para-CORDIC[2-3]、Hybrid CORDIC[4]、Scale-Free CORDIC[5-7]等，或者使用循环展开的方式在一个时钟周期内做多轮迭代[8-11]。这些方法在一定程度上缩短了 CORDIC 算法的延时，但是仍满足不了低延时计算的需求。

除 CORDIC 算法外，有些设计采用了多项式拟合法[12-13]、泰勒级数展开法[14-15]、幂级数展开法[16-17]等算法。这些算法收敛速度较快，但是需要大量的乘法运算才能得到单精度浮点所需精度。

为了解决正弦余弦函数计算延时较大这一问题，本文提出了一种低延时的浮点正弦余弦函数硬件算法。该算法采用分段的思想将单精度浮点数输入按照指数范围的不同划分为 3 个不同区域，在这 3 个不同区域内分别采用不同的算法来保证计算的精度。同时，保证整体硬件模块在 4 个周期内完成计算。

本文结构主要分为算法、电路设计和实验结果 3 个部分。算法部分采用了 3 个算法，分别是泰勒 0 阶近似法、泰勒 1 阶近似法[18]和直接计算。由于泰勒 0 阶近似和直接计算比较简单，本文重点分析泰勒 1 阶近似算法。相应的，电路设计部分也将重点放在泰勒 1 阶近似算法的实现上。在实验结果部分，本文使用 verilog 硬件描述语言实现了提出的算法和电路结构，完成了精度测试，并在联电 UMC 55nm 的工艺和 Altera Cyclone IV 的平台上进行了逻辑综合。结果表明，该浮点正弦余弦函数硬件算法的计算精度与标准 C 语言 math 库单精度正弦余弦函数的计算精度相比，最多仅相差 1 个误差单位(ulp)，并且能够在 4 个时钟周期内完成运算，具有低延时的特点。

1　低延时浮点正弦余弦函数硬件算法

本节算法所述的正弦余弦函数的数学形式为 $\sin(2\pi x)$和 $\cos(2\pi x)$。如果要计算 $\sin y$ 和$\cos y$，可做变换 $y=2\pi x$，将其转换为 $\sin(2\pi x)$和$\cos(2\pi x)$。在设计中使用 $\sin(2\pi x)$和 $\cos(2\pi x)$的形式，是因为在这种形式下，可以容易地将运算定点化，减少计算量，降低延时。

1.1　将输入的单精度浮点数划分区域

IEEE754 标准所规定的一个规格化的单精度浮点数[19] x 有 32 位，其中符号位为 $x[31]$；指数位有 8 位，从第 31 位到第 23 位，表示为 $x[30:23]$，它带有 127 的偏阶；尾数位有 23 位，从第 22 位到第 0 位，表示为 $x[22:0]$，它省略了一个隐含的整数位 1。单精度浮点数 x 转换为一个实数 $(-1)^{x[31]}(1+x[20:0]2^{-23})2^{x[30:23]-127}$。为了避免歧义，在上下文中用“指数”这一用语表示 $x[30:23]$减去偏阶 127；用“尾数”表示$\{1'b1, x[22:0]\}$，其中 b 代表二进制，括号代表将其连接为 23 位二进制数。

对于正弦余弦函数，根据其数学特性，将输入按区间进行分段计算，正弦函数分段如图 1 所示。为了硬件容易实现，可按输入单精度浮点数的指数范围进行区间划分。一方面，根据三角函数的性质，当输入单精度浮点数的指数大于等于 22 时，该实数为 $(-1)^{x[31]}(1+x[22:0]2^{-23})2^{e}$，其中 e 为指数，$e\geqslant 22$。此时该实数必然是整数或整数点五的形式(此时 $e=22$ 且 $x[0]=1$)。对 $\cos(2\pi x)$函数，如果该实

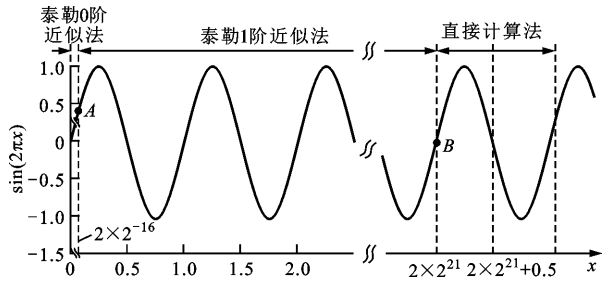

图 1　正弦函数分段示意图

Fig. 1　Sine function segmentation diagram

数为整数，则结果为 1.0；如果该实数为整数点五的形式，则结果为－1.0。对 $\sin(2\pi x)$函数，无论该实数为哪种形式，结果均为 0.0。由于指数大于等于 22 时，计算非常简单，因此在指数为 22 处进行分段。另一方面，分析三角函数的数学性质可知，当输入很小的时候有 $\sin(2\pi x)\approx 2\pi x$，$\cos(2\pi x)\approx 1.0$，且输入越小等号两边的值越接近。因此必然存在一个指数使得 $\sin(2\pi x)$与 $2\pi x$ 的误差以及 $\cos(2\pi x)$与 1.0 的误差小于单精度浮点数所能表示的最小误差。经过计算得知，该指数为－16，因此在指数为－16处进行分段。于是很自然的就将输入的单精度浮点数按指数范围划分为[－126，－16]、[－15，21]和[22，126]3 个区间。

为了节省电路面积，可以将正弦和余弦函数合并为一个函数，此处定义了一个 1 位的指示信号 c_s，当计算正弦函数时 $c_s=0$，当计算余弦函数时 $c_s=1$。分段算法的步骤如伪代码所示。

当输入的指数在[－15，21]之间时，本文使用泰勒 1 阶近似法进行计算。在伪代码中，用函数 Taylor1(x，c_s)表示在这个区间内的计算方法。

伪代码：分段算法步骤

输入：单精度浮点数 x，指示信号 c_s

输出：单精度浮点数 $y=\sin(2\pi x)$或 $y=\cos(2\pi x)$

```
if (c_s==0) then
    if (x[30:23]≥149) then y=0.0;
    else if (x[30:23]≤111) then y=2πx;
    else y=Taylor1(x,c_s);endif
else
    if (x[30:23]≥150) then y=1.0;
    else if (x[30:23]==149) then y=
    (－1.0)^x[0];
    else if (x[30:23]≤111) then y=1.0;
    else y=Taylor1(x,c_s);endif
endif
```

泰勒 1 阶近似法的主要步骤为：首先从输入的浮点数中提取符号标志位 s_x，诱导公式标志位 f_1、f_2，构造一个定点数 f。然后通过诱导公式[20]等一系列变换，将对浮点数 x 的正弦或余弦值的计算转换为对定点数 f 的正弦值的计算，得到 $\sin(2\pi x)$或 $\cos(2\pi x)$的定点数结果 z。最后将该定点数结果 z 转换成浮点数格式。

1.2 将浮点运算转换为定点运算

将输入的单精度浮点数转换为定点数的一般方法的过程如图 2 所示。

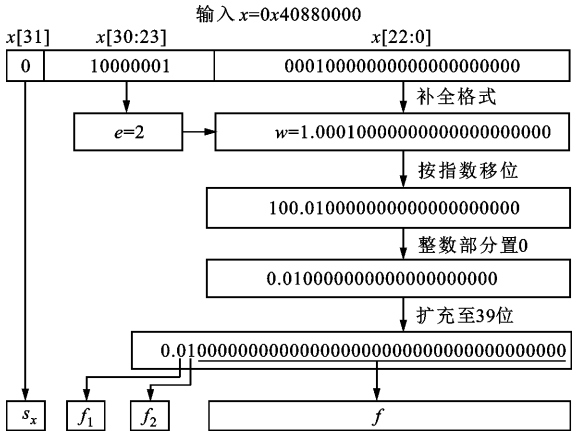

图 2 浮点数转换为定点数的一般方法

Fig. 2 General method of converting floating-point numbers to fixed-point numbers

在指数为[－15，21]的区间内，可将浮点数转换为定点数。按一般方法，首先应当计算出浮点数 x 的指数 e，并补全尾数 $w=\{1'b1,x[22:0]\}$。如果 $e>0$，则应该将尾数 w 的小数点右移 e 位；如果 $e<0$，则应该将尾数的小数点向左移 $|e|$ 位。考虑到 $\sin(2\pi x)$和 $\cos(2\pi x)$的周期均为 1，所以移位后的定点数的整数部分可以直接置 0。

为了保证定点数的格式固定，考虑到整数部分必为 0，规定使用 1 位做整数部分，小数点固定在整数部分的 0 后面。由于指数为－15 的时候，小数部分位数最长，为 $23+|e|=38$ 位，因此规定所有定点数都将小数部分扩充到 38 位。该定点数记为 F。

用统一移位的方法得到上述定点数 F。具体操作为：计算移位数 $s_r=x[30:23]-112$，然后将尾数按 $m=\{16'h0001,x[22:0]\}$的方式补齐到 39 位，其中 h 代表该数为 16 进制数，最后将 m 左移 s_r 位。图 3 给出了浮点数转换为定点数的统一移位的方法。

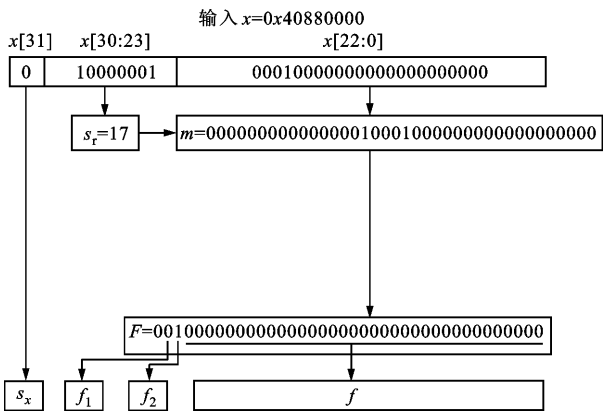

图 3 浮点数转换为定点数的统一移位的方法

Fig. 3 Uniform shift method for converting floating-point numbers to fixed-point numbers

输入的指数为 -15 时，将 $x[22:0]$ 前面补上 $16'h0001$，变成 $m=\{16'h0001,x[22:0]\}$，正好是所需的定点数格式。指数为其他值时，只需计算它比 -15 大多少，就左移多少位。因此，统一移位的方法是正确的。下面给出一个具体的例子。

图 2 和图 3 所示为浮点数 $0x40880000$ 转换为定点数的过程。图 2 所示为一般方法，先补全格式，然后按指数移位，整数部分置 0，最后扩充到 39 位。图 3 所示为统一移位的方法。可以看到两种方法所得的定点结果是一致的。

1.3　使用诱导公式将正弦余弦函数的计算转换为 $[0,\pi/2)$ 内的定点正弦函数计算

考虑到正弦和余弦函数关于 π 的诱导式为

$$\left.\begin{aligned}\sin(2\pi(x+0.5)) &= -\sin(2\pi x)\\ \cos(2\pi(x+0.5)) &= -\cos(2\pi x)\end{aligned}\right\} \tag{1}$$

关于 $\pi/2$ 的诱导式为

$$\left.\begin{aligned}\sin(2\pi(x+0.25)) &= \cos(2\pi x)\\ \cos(2\pi(x+0.25)) &= -\sin(2\pi x)\end{aligned}\right\} \tag{2}$$

根据 1.2 节所述已将浮点数 x 转换成了一个 39 位的定点数 F。在该定点表示下，$F[37]$ 表示 0.5 所在位，$F[36]$ 表示 0.25 所在位，因此记 $f_1=F[37]$，用于标记该定点数 F 的正弦或余弦值的计算是否可以使用诱导式(1)化简，同理记 $f_2=F[36]$，表示是否可以使用诱导式(2)化简。结合变量 f_1、f_2、c_s 以及符号位 s 判断需要使用的诱导式，就可以推导出最终符号位 s_y 以及是否需要计算 $[0,\pi/2)$ 内余弦函数的指示信号 c，见表 1。

表 1 中 s_y 列为符号指示位，当化简后出现负号时，s_y 设置为 1；c 列为余弦计算指示位，当化简后需要计算余弦函数时，c 设置为 1。

使用上述诱导公式之后，对定点数 F 的正弦或余弦函数值计算就转换成了对 $F[35:0]$ 的正弦或余弦计算，记 $f=F[35:0]$，其中 $F[35:0]$ 表示定点数的第 35 位到第 0 位。考虑到 $0\leqslant 2\pi 2^{-38}f<\pi/2$，因此 $\sin(2\pi 2^{-38}f)$ 或 $\cos(2\pi 2^{-38}f)$ 是取值在 $[0,\pi/2)$ 内的正弦或余弦函数。

再考虑在 $[0,\pi/2)$ 内的正弦和余弦有如下关系

$$\cos(2\pi x)=\sin(2\pi(0.25-x)) \tag{3}$$

将 $x=2^{-38}f$ 代入式(3)可得下式

$$\cos(2\pi 2^{-38}f)=\sin(2\pi(0.25-2^{-38}f))=\sin(2\pi 2^{-38}(2^{36}-f)) \tag{4}$$

因此，所有 $[0,\pi/2)$ 内的余弦函数的计算都可以转换为正弦函数的计算。$2^{36}-f$ 实际上就等于取 f 的补码 $\sim f+1$，其中 $\sim$ 表示按位取反。此处有一个例外情况：如果 $f=0$，那么取反加 1 后，位宽就会超出 36 位，导致后面无法计算。该例外情况可以单独考虑：$f=0$ 和 $c=1$ 时，相当于在求 $\cos(0)=1.0$，这个情况下，可直接输出结果 1.0。

表 1　使用诱导公式化简后的指示信号

Table 1　Indicative signal table after simplification using the induced formulas

c_s	s_x	f_1	f_2	化简后形式	s_y	c
0	0	0	0	sin	0	0
0	0	0	1	cos	0	1
0	0	1	0	−sin	1	0
0	0	1	1	−cos	1	1
0	1	0	0	−sin	1	0
0	1	0	1	−cos	1	1
0	1	1	0	sin	0	0
0	1	1	1	cos	0	1
1	0	0	0	cos	0	1
1	0	0	1	−sin	1	0
1	0	1	0	−cos	1	1
1	0	1	1	sin	0	0
1	1	0	0	cos	0	1
1	1	0	1	−sin	1	0
1	1	1	0	−cos	1	1
1	1	1	1	sin	0	0

1.4　使用泰勒 1 阶近似计算 $[0,\pi/2)$ 内的正弦函数

经过 1.3 节的变换，需要计算定点数 f 的正弦函数值 $\sin(2\pi 2^{-38}f)$。为了快速计算该正弦函数值，这里使用了泰勒 1 阶近似算法。它的核心思想是：将输入定点数 f 按照前 m 位和后 $36-m$ 位划分为主要部分和次要部分[21]，分别记作 f_3 和 f_4，也即 $f_3=f[35:36-m]$，$f_4=f[35-m:0]$，其中 $f[35:36-m]$ 代表定点数 f 的第 35 位到第 $36-m$ 位，$f[35-m:0]$ 代表定点数 f 的第 $35-m$ 位到第 0 位。此时的 f_3 和 f_4 仍然是定点数，f_3 对应的实数为 $2^{-(m+2)}f_3$，f_4 对应的实数为 $2^{-38}f_4$。

考虑任意函数 $g(x)$ 的泰勒级数展开式，令它在 $x_0=2^{-(m+2)}f_3$ 处展开，表达式如下

$$g(x)=\sum_{n=0}^{\infty}\frac{g^{(n)}(2^{-(m+2)}f_3)}{n!}(x-2^{-(m+2)}f_3)^n \tag{5}$$

令 $x=2^{-38}f=2^{-(m+2)}f_3+2^{-38}f_4$，代入级数得到

$$g(2^{-38}f)=\sum_{n=0}^{\infty}\frac{g^{(n)}(2^{-(m+2)}f_3)}{n!}(2^{-38}f_4)^n \tag{6}$$

将 $g(x)=\sin(2\pi x)$ 代入式(6)，保留 $n=0$ 和 $n=1$

两项可以得到

$$\sin(2\pi 2^{-38}f)=\sin(2\pi 2^{-(m+2)}f_3)+2\pi\cos(2\pi 2^{-(m+2)}f_3)2^{-38}f_4 \tag{7}$$

式(7)即为所述泰勒1阶近似算法。

为了加快计算速度,可以使用查找表计算 $\sin(2\pi 2^{-(m+2)}f_3)$ 和 $2\pi\cos(2\pi 2^{-(m+2)}f_3)$ 两项。在图4中,LutSin(f_3)和LutCos(f_3)表示这两个查找表。这两个查找表的值仅和 f_3 有关,若 f_3 的位宽 m 取得越宽,则主要部分 f_3 越接近定点数 f 的值,最终的结果也越精确。查找表的输出形式是类似于浮点数的形式。以 $s_t,s_e=\text{LutSin}(f_3)$ 为例,s_t 是一个 k 位的定点数,它有1位的整数位和 $k-1$ 位的小数位,s_e 是8位的指数位。例如:取 $m=4,k=24$,输入 $f_3=(1101)_2=13$ 时,有 $\sin(2\pi 2^{-6}\times 13)=(1.11101001111101000001011)_2\times 2^{-1}$。因此,当查找表LutSin($f_3$)的输出 s_t 为 $k=24$ 位的二进制数111101001111101000001011时,输出 s_e 为-1。

使用高级语言按上述方法建模,计算得出,当查找表参数取 $m=13$ 和 $k=32$ 时可以保证该泰勒1阶近似算法的精度与标准C语言math库单精度正弦余弦函数的计算精度相比,最多仅相差1 ulp。因此本算法中参数 $m=13$ 和 $k=32$ 固定不变。如果不需要高精度,可以减小查找表参数 m 和 k 以降低精度,同时减少查找表的大小。

当取查找表参数 $m=13$ 和 $k=32$ 之后,为避免使用浮点乘法和浮点加法,需要将式(7)的计算转换为整数运算。考虑式(7),中间计算过程中最小的数出现在 $2\pi\cos(2\pi 2^{-(m+2)}f_3)2^{-38}f_4$ 这一项上,它的最小值出现在 f_3 为全1且 $f_4=1$ 的情况下。在查找表参数为 $m=13$ 和 $k=32$ 的情况下可计算出这一最小值为 $2\pi\cos(2\pi 2^{-15}(2^{13}-1))2^{-38}\times 1=(1.00111011110100111100110001111101)_2\times 2^{-48}=(100111011110100111100110001111101)_2\times 2^{-79}$。这个数是本算法中间过程里可能出现的最小的数,因此规定中间计算过程使用如下的80位定点数格式:整数部分使用1位来表示,小数部分使用79位来表示。

式(7)的计算过程如下:首先根据 f_3 查表得到 $s_t,s_e=\sin(2\pi 2_{-15}f_3)$;$c_t,c_e=2\pi\cos(2\pi 2_{-15}f_3)$。令 $t_1=s_t$,计算 $t_2=c_tf_4$。然后计算移位数 $n_1=s_e-31+79=s_e+48$;$n_2=c_e-31-38+79=c_e+10$。最后将 t_1 和 t_2 分别移位到规定的80位定点数格式,相加得到 z。得到80位定点数结果的过程如图4所示。

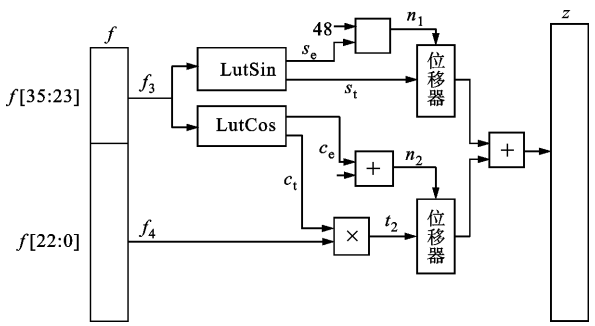

图4 得到80位定点数结果的流程图

Fig. 4 Flowchart for getting 80-bit fixed-point numbers

1.5 将计算结果定点数转换成单精度浮点格式

根据1.4节所述,z 为定点数 f 的正弦函数值,它是一个80位的定点数。将 z 转换为单精度浮点数的步骤如图5所示。

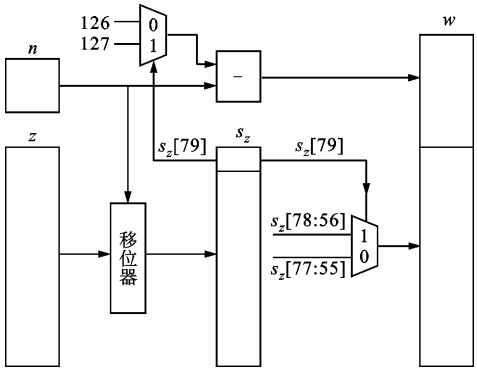

图5 80位定点数转换为浮点数的流程图

Fig. 5 Flowchart for converting 80-bit fixed-point numbers to floating-point numbers

将定点数 z 转换为单精度浮点数最直接的方法是:计算 z 的前缀0个数(即高位第一个1前面的0的个数)作为指数,取高位第一个1后面的23位作为尾数。由于计算前缀0的个数与后面的运算是串行的,会导致较大的延时。为减小这部分延时,本文提出了一个方法,其核心思想是:对[0,π/2)内的正弦函数,其函数值和输入是非常相近的,通常只会相差至多一个指数位,因此可以预先计算输入的指数,然后先行移位一次,再修正移位后的结果。

具体实现流程如图5所示:首先计算 f 的前缀0的个数 n,然后将 z 左移 n 位,移位后可以保证最高位是1或者次高位是1。如果最高位是1,则指数为$-n$;如果次高位是1,指数为$-(n+1)$。尾数取最高位或次高位的1后面的23位。使用该方法可以提前计算出移位数 n,从而减小了延时。

该方法的正确性可以通过如下方程式证明。假设 f 的前缀 0 的个数为 n，则 $2^{35-n}\leqslant f<2^{36-n}$，于是有 $\sin(2\pi 2^{-(n+3)})\leqslant\sin(2\pi 2^{-38}f)<\sin(2\pi 2^{-(n+2)})$。通过遍历计算可知 $\sin(2\pi 2^{-(n+3)})>2^{-(n+1)}$ 和 $\sin(2\pi 2^{-(n+2)})<2^{-(n-1)}$。于是可以得到 $2^{-(n+1)}<\sin(2\pi 2^{-38}f)<2^{-(n-1)}$。这说明结果的指数只可能为 $-(n+1)$ 或 $-n$，因此该方法是正确的。

使用本文提出的方法将定点数转换为浮点数后，再结合 1.3 节中所述的符号位 s_y，将它添加到相应的位置上，就得到了 Taylor1(x，c_s)函数的计算结果。

2　电路设计

2.1　总体电路结构

正弦余弦函数的电路实现通常有流水线式和迭代式两种结构，本文采用流水线式结构来进行电路设计。

浮点正弦余弦函数计算电路的结构如图 6 所示，采用了 4 级流水线的方式。第 1 级实现了浮点数转换为定点数的逻辑，第 2 级和第 3 级实现了[0，π/2)内正弦函数值的计算，第 4 级实现了定点数结果转换成浮点结果。

图 6 仅显示了硬件实现主要部分的数据通路。输入指数范围在[−126，−16]，[22，126]这两个区间时，由于算法较简单，图 6 未给出其数据通路。另外，$c=1$ 并且 $f=0$ 这一特殊情况的数据通路在图中也未表示。

2.2　各流水级的具体实现

流水线第 1 级运算过程的主要任务有 3 个。一是将输入的浮点数 x 转换为定点数 f，它的实现方式为先计算移位数 $s_r=x[30:23]-112$，然后将补位后的尾数 $m=\{16'h0001,x[22:0]\}$ 左移 s_r 位。二是产生符号指示位 s_y。符号指示位 s_y 由输入符号位 s_x、指示信号 c_s、诱导公式标志位 f_1 和 f_2 共同得到。根据表 1，可以得到逻辑表达式 $s_y=f_1\oplus(c_s\&f_2)\oplus(\sim c_s\&s_x)$。三是产生余弦计算指示位 c。根据表 1，可得出其逻辑表达式为 $c=c_s\oplus f_2$。电路中使用了一个多路选择器，根据 c 信号的取值选择 f 或者 f 的补码。

流水线第 2 级为[0，π/2)内的正弦函数计算的查找表部分。查找表常用的实现方式有组合逻辑实现和 RAM/ROM 方式实现。这两种方式各有优缺点。组合逻辑实现的优势是查找表操作可在 1 个时钟周期内完成，无需特殊工艺，缺点是电路面积较大。RAM/ROM 方式实现的优势为占用芯片面积小，缺点为需要特殊工艺。本设计使用了组合逻辑方式实现查找表。根据 1.4 节所述，查找表参数固定为 $m=13$ 和 $k=32$，因此查找表 LutSin 和 LutCos 的输入为一个 13 位的整数 f_3，输出是 32 位的整数 s_t、c_t 和 8 位的指数 s_e、c_e。组合逻辑实现该查找表会占用一定的面积，并且延时较大，因此本设计单独分配了一个时钟周期用于查表。

流水线第 3 级为[0，π/2)内正弦函数的计算部分，主要运算为移位运算和乘加运算。首先使用查找表的指数信息 s_e 和 c_e 计算移位数 n_1 和 n_2，同时计算 32b×23b 的整数乘法运算 c_tf_4；然后将 s_t 和 c_tf_4 的结果分别移 n_1 和 n_2 位并相加，得到一个 80 位的定点数结果 z。同时在这一个周期内计算定点数 f 的前缀 0 的个数 n，用于最后将定点结果转换成浮点结果。

流水线第 4 级将定点结果 z 转换成浮点结果，使用 1.5 节所述方法，首先将 z 左移 n 位变成 s_z，此

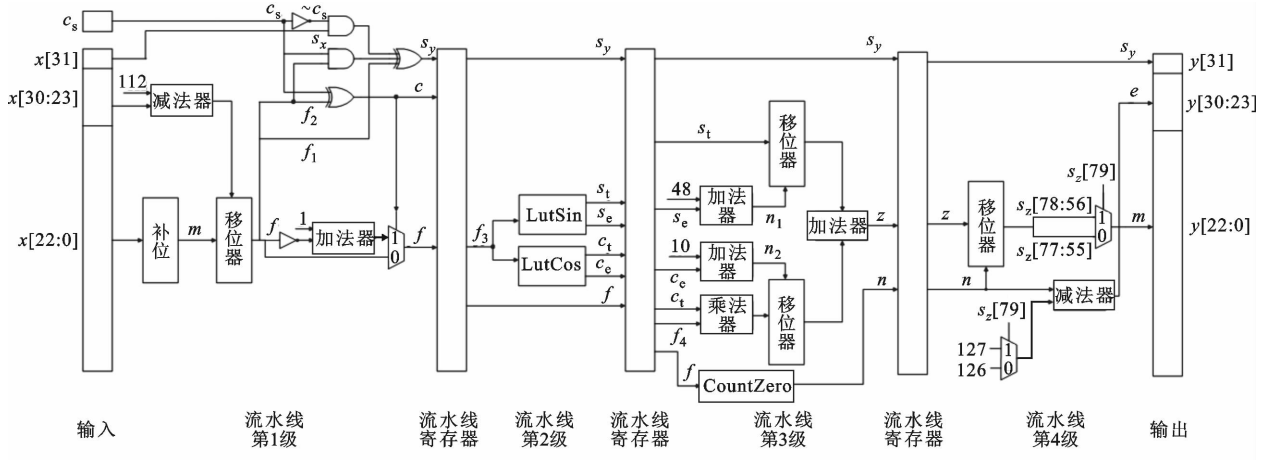

图 6　浮点正弦余弦函数计算电路结构

Fig. 6　Floating-point sine and cosine function calculation circuit structure

时 s_z 最高位为 1，或者次高位为 1。因此只需用二选一多路选择器根据最高位 $s_z[79]$ 选取最高位或次高位的 1 后面的 23 位作为尾数 m，并且选择指数 e 为 $127-n$ 或 $126-n$。将尾数 m、指数 e 和符号位 s_y 按照单精度浮点数的格式拼在一起，即可得到单精度浮点输出的结果。

3 实验结果

3.1 精度测试

按照第 2 节所述，使用 verilog 硬件描述语言实现了本文提出的高精度低延时浮点正弦余弦函数硬件算法，并和 C 语言 math 库进行了对比。本次测试的计算机平台为 Intel Core i5-7400 @3.00 GHz (4 核)CPU，内存 24 GB，C 编译器采用 gcc4.9.2。仿真平台为 windows 下的 Modelsim SE-64 10.4 版本。测试方法为对所有的规格化浮点数进行遍历测试，将电路仿真值与 C 语言 math 库结果进行对比。

表 2 为正弦函数遍历测试结果，表 3 为余弦函数遍历测试结果。表中“范围”一列为输入指数范围的区间，它与 1.1 节所述的划分方法是一致的；完全一致占比表示仿真值与 C 语言 math 库结果完全一致的个数占在该输入指数范围内遍历总数的百分比；误差 1 ulp 占比表示仿真值与 C 语言 math 库结果有 1 ulp 误差的个数占在该输入指数范围内遍历总数的百分比。

表 2 正弦函数遍历测试结果

Table 2 Sine function traversal test results

范围	完全一致占比/%	误差 1 ulp 占比/%
[−126,−16]	83.24	16.76
[−15,21]	95.68	4.32
[22,126]	100.00	0

表 3 余弦函数遍历测试结果

Table 3 Cosine function traversal test results

范围	完全一致占比/%	误差 1 ulp 占比/%
[−126,−16]	100.00	0
[−15,21]	94.70	5.30
[22,126]	100.00	0

遍历结果显示，对于正弦函数，在输入的指数范围为[−126，−16]时，结果存在 1 ulp 的误差。这是由于在该区间内使用了 $\sin(2\pi x)=2\pi x$ 公式，需要对输入做乘法，而 π 是一个无理数，将 π 定点化会产生截断误差，导致某些情况下它与参考结果相差 1 ulp。对于输入的指数范围为[−15，21]之间时，由于采用泰勒一阶近似算法需要查找表，查找表也会产生截断误差，导致随机的 1 ulp 误差。在输入指数为[22，126]之间时，按数学理论结果为 0，没有进行截断，因此不存在误差。

对于余弦函数，在输入的指数范围为[−126，−16]和[22，126]之间时，不存在 1 ulp 的误差。这是因为在这些区间上的余弦函数值都为 1.0 或者 −1.0，没有截断操作，因此不存在截断误差。对于输入的指数范围为[−15，21]时，与正弦函数同理，由于采用的泰勒 1 阶近似算法需要查找表，会产生截断误差，导致产生随机的 1 ulp 误差。

3.2 电路性能

本文首先使用联电 UMC 55nm CMOS 工艺库，在 SS 工艺角、1.08 V、150 ℃、RVT 条件下，按照 250 MHz 的时序约束，对电路进行逻辑综合。本文还在 FPGA 平台上，使用 Altera Cyclone IV 的 EP4CE115F29C7 芯片，在 Slow 1 200 mV 85 ℃ Model 下进行逻辑综合实现。两种平台下正弦余弦函数电路性能如表 4 所示。

表 4 正弦余弦函数电路性能

Table 4 Sine and cosine function circuit performance

平台	时钟频率/MHz	面积或资源	总延时/ns
UMC 55nm	250	247 329 μm^2	16
Cyclone IV	64	42 743 LUT	62.5

查找表(LUT)、逻辑片(Slice)、逻辑单元(LE)均为电路的基本组成单元。

本设计在 UMC 55nm 的 SS 工艺角下可达到 250 MHz 的时钟频率，完成一次正弦余弦函数运算的延时仅为 16ns，具有低延时的特点。本设计由于采用了大查找表和乘法器等单元，电路面积相对较大，为 247 329 μm^2。在 Cyclone IV 平台上，本设计在最差工艺角下可达到 64.02 MHz 的时钟频率，完成一次浮点正弦余弦函数运算延时为 62.5 ns，占用了 42 734 个 LUT。由于本设计采用组合逻辑实现大查找表，因此占用了大量 LUT 资源。

表 5 给出了本文设计与其他文献中正弦余弦函数电路的性能对比。表中列出的正弦余弦函数电路的最大误差均不超过 1 ulp。在 ASIC 设计中，文献[14]中采用 TCORDIC 算法实现了双精度正弦余弦函数，在 40 nm CMOS 的 TT 工艺角下，需要 64 个周期才能完成运算，总延时 38.4 ns。文献[8]使用循环展开 CORDIC 算法实现了双精度正弦余弦函

数，在 TSMC65nm 工艺下，需要 15 个周期完成运算，总延时 75 ns。文献[20]中采用基于查找表的泰勒展开算法实现了单精度正弦余弦函数，在 VirtexII 平台下，总延时为 85 ns。文献[18]中采用基于查找表的泰勒展开算法实现了 16 位定点正弦余弦函数，在 StratixII 平台下，需要 3 个周期完成运算，总延时 31.3 ns。

表 5　不同方案时电路延时、面积性能对比

Table 5　Circuit delay and area comparison among different schemes

方案	类型	延时/ns	面积或资源	平台
本文	单精度	16	$247\times10^3\ \mu m^2$	UMC55 SS
文献[14]	双精度	38.4	$194\times10^3\ \mu m^2$	40nm TT
文献[8]	双精度	75	$293\times10^3\ \mu m^2$	65nm TT
文献[20]	单精度	89	2 081Slice	Virtex II
文献[18]	16 位定点	31.3	828LE	Stratix II

4　结　论

本文提出了一种高精度低延时的浮点正弦余弦函数硬件实现算法。它将输入按照指数划分为 3 个不同的区域，在这 3 个区域内分别采用了不同的算法，达到了单精度浮点运算所要求的精度。本文着重说明了输入指数在[－15，21]之间所用的泰勒 1 阶近似算法，该算法通过一系列诱导公式，将原计算转换为[0，π/2)内的正弦函数计算，然后，将其进一步转换为硬件友好的查找表运算和乘加运算，从而实现了低延时。

在电路设计部分，本设计使用了四级流水线结构。实验结果表明，该硬件电路的计算结果与 C 语言 math 库的结果至多只相差 1 ulp。在联电 UMC55nm 的 SS 工艺角下，该电路能够达到 250 MHz 的时钟频率，延时仅 16 ns。在某些需要低延时的应用场景中，本文提出的高精度低延时正弦余弦函数硬件算法具有重要使用价值。

参考文献：

[1] VOLDER J E. The CORDIC trigonometric computing technique [J]. IRE Transactions on Electronic Computers, 1959, EC-8(3): 330-334.

[2] JUANG T B, HSIAO S F, TSAI M Y. Para-CORDIC: parallel CORDIC rotation algorithm [J]. IEEE Transactions on Circuits and Systems: I Regular Papers, 2004, 51(8): 1515-1524.

[3] CHEN L, LOMBARDI F, JIE H, et al. A fully parallel approximate CORDIC design [C]// Proceedings of the IEEE International Symposium on Nanoscale Architectures. Piscataway, NJ, USA: IEEE, 2016: 197-202.

[4] SHUKLA R, RAY K C. Low latency hybrid CORDIC algorithm [J]. IEEE Transactions on Computers, 2014, 63(12): 3066-3078.

[5] MAHARATNA K, BANERJEE S, GRASS E, et al. Modified virtually scaling-free adaptive CORDIC rotator algorithm and architecture [J]. IEEE Transactions on Circuits and Systems for Video Technology, 2005, 15(11): 1463-1474.

[6] MOROZ L, MYKYTIV T, HERASYM M. Improved scaling-free CORDIC algorithm [C]// Proceedings of the East-West Design and Test Symposium. Piscataway, NJ, USA: IEEE, 2013: 1-5.

[7] JAIME F J, SANCHEZ M A, HORMIGO J, et al. Enhanced scaling-free CORDIC [J]. IEEE Transactions on Circuits and Systems: I Regular Papers, 2010, 57(7): 1654-1662.

[8] HOU Nanxin, WANG Mingjiang, ZOU Xiafeng, et al. A low latency floating point CORDIC algorithm for sin/cosine function [C]// Proceedings of the 2019 4th IEEE International Conference on Signal and Image Processing. Piscataway, NJ, USA: IEEE, 2019: 751-755.

[9] MAHARATNA K, EL-SHABRAWY K, AL-HASHIMI B. Reduced Z-datapath CORDIC rotator [C]// Proceedings of the 2008 IEEE International Symposium on Circuits and Systems. Piscataway, NJ, USA: IEEE, 2008: 3374-3377.

[10] GISUTHAN B, SRIKANTHAN T. Flat CORDIC: a unified architecture for high-speed generation of trigonometric and hyperbolic functions [C]// Proceedings of the 43rd IEEE Midwest Symposium on Circuits and Systems. Piscataway, NJ, USA: IEEE, 2000: 1414-1417.

[11] VILLALBA J, LANG T. Low latency word serial CORDIC [C]// Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures and Processors. Piscataway, NJ, USA: IEEE, 1997: 124-131.

[12] MICHARD R, TISSERAND A, VEYRAT-CHARVILLON N. Small FPGA polynomial approximations with 3-bit coefficients and low-precision estimations of the powers of x [C]// Proceedings of the 2005 IEEE International Conference on Application-Specific Systems, Architecture Processors. Piscataway, NJ,

USA: IEEE Computer Society, 2005: 334-342.

[13] LEE D U, CHEUNG R, LUK W, et al. Hardware implementation trade-offs of polynomial approximations and interpolations [J]. IEEE Transactions on Computers, 2008, 57(5): 686-701.

[14] ZHU Baozhou, LEI Yuanwu, PENG Yuanxi, et al. Low latency and low error floating-point sine/cosine function based TCORDIC algorithm [J]. IEEE Transactions on Circuits and Systems: I Regular Papers, 2017, 64(4): 892-905.

[15] SHERAMIN G, BABAIE S, ASL R K. Data-oriented architecture of sine and cosine functions [C] // Proceedings of the 2010 3rd International Conference on Advanced Computer Theory and Engineering. Piscataway, NJ, USA: IEEE, 2010: V6-32-34.

[16] TIWARI H D. Vedic processor based trigonometric calculations using power series [C] // Proceedings of the 2019 5th International Conference on Advanced Computing and Communication Systems. Piscataway, NJ, USA: IEEE, 2019: 118-123.

[17] MANDELBAUM D M, MANDELBAUM S G. A fast, efficient parallel-acting method of generating functions defined by power series, including logarithm, exponential, and sine, cosine [J]. IEEE Transactions on Parallel and Distributed Systems, 1996, 7(1): 33-45.

[18] 刘小明, 洪一. 基于查找表和 Taylor 展开的正余弦函数的实现 [J]. 现代电子技术, 2009, 32(13): 165-166.
LIU Xiaoming, HONG Yi. Implementation of sine and cosine function based on look-up table and Taylor expansion [J]. Modern Electronics Technique, 2009, 32(13): 165-166.

[19] IEEE. IEEE standard for floating-point arithmetic: 754-2019 [S]. Piscataway, NJ, USA: IEEE, 2019: 1? 84.

[20] DETREY J, DE DINECHIN F. Floating-point trigonometric functions for FPGAs [C] // Proceedings of the 2007 International Conference on Field Programmable Logic and Applications. Piscataway, NJ, USA: IEEE, 2007: 29-34.

[21] NGUYEN M X, DINH-DUC A V. Hardware-based algorithm for sine and cosine computations using fixed point processor [C] // Proceedings of the 2014 11th International Conference on Electrical Engineering/ Electronics, Computer, Telecommunications and Information Technology. Piscataway, NJ, USA: IEEE, 2014: 1-6.

(编辑 武红江)