

同步加速器磁场控制状态矩阵的优化乘法器设计

龙银东^{1,2} 乔为民¹ 林飞宇^{1,2} 敬 岚¹

1 (中国科学院近代物理研究所 兰州 730000)

2 (中国科学院研究生院 北京 100049)

摘要 同步加速器的磁场控制系统系高精度的实时控制系统。随着 CSR 工程的进一步改造要求,以及数字化处理技术的不断发展,对浮点乘法的运算速度以及相应占用资源也提出更高要求。本文通过 VHDL 语言和 BOOTH 算法实现的优化乘法器,采用了结合 3-2 压缩器和 4-2 压缩器的树型结构,并以 CycloneIII 芯片 EP3C25F256 为硬件环境,进行了功能和时序仿真,与其他设计的仿真结果比较,验证了该浮点乘法器的正确和高速特性。

关键词 大规模集成电路,浮点数,浮点乘法器,规格化

中图分类号 TP332.2⁺1

20 世纪 80 年代,兰州重离子加速器(HIRFL)建成后,通过创新的物理思想和技术路线,取得了合成 20 多种远离 β 稳定线新核素的一系列成果。其控制系统也经历了不断的创新与改造过程。近年来,随着兰州重离子加速器冷却储存环(HIRFL-CSR)的建设,进一步提高系统的稳定性和精确度显得尤为重要^[1]。

当今,随着集成芯片的不断发展和现代控制理论的进一步完善,建立全数字的控制系统以使加速器的束流更佳,已成为可能。本文以 ALTERA 公司的 Cyclone III 3C25F256 芯片作为硬件平台,Quartus II 8.0 作为硬件编程环境^[2],为同步加速器磁场控制系统(图 1)的进一步数字化改造开发了浮点数优化乘法器。通过 BOOTH 算法和结合 3-2 压缩器与 4-2 压缩器的新型树型结构^[3],实现了针对 16 位尾数精度的优化乘法器。

1 整体设计概述

1.1 浮点数规格化标准

为使每个规格化浮点数都有唯一的机器数形式,有必要统一规格化浮点数的编码格式。目前,国际上普遍采用的是 IEEE754 标准^[4],该标准规定了 4 种浮点机器数格式,每种格式的浮点机器数都由数符字段、阶码字段和尾数字段三部分组成。这 4 种浮点数格式分别是标准单精度浮点数格式、单精度扩展的浮点数格式、双精度浮点数格式、双精

度扩展的浮点数格式。其主要技术是对阶码部分采用了偏移码的概念,使浮点数之间进行快速方便的数学计算,同时对尾数部分作规格化处理,从而能得到比实际位数多 1 位的有效精度^[3,5]。

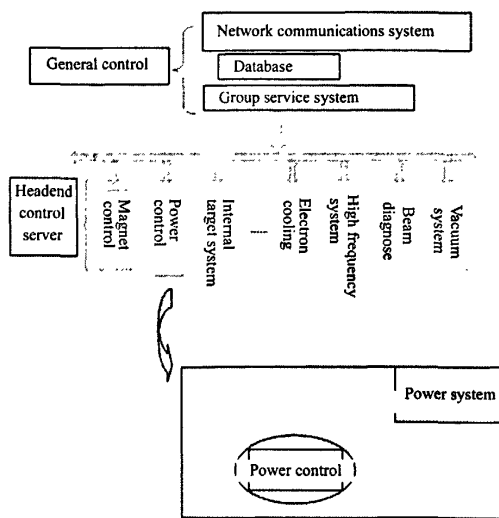


图 1 同步加速器磁场控制系统框图
Fig.1 The chart of synchronous accelerator's control system.

考虑到节省 FPGA 器件资源、提高运算速度,我们结合 IEEE 754 标准,在满足精度的前提下,提出了对 23 位浮点数的格式化标准,以满足本次同步加速器数字化改造的万分之一精度要求^[1]。如图 2 所示,第 22 位为浮点数的数符字段,表示该浮点数

第一作者:龙银东,男,1982 年出生,2005 年毕业于贵州大学,现为中国科学院近代物理研究所在读博士研究生,核技术及应用专业,从事数字信号处理及计算机自动控制方面的研究

收稿日期:2009-02-28,修回日期:2009-06-15

的正负特性；第 21 到第 16 位为阶码字段，表示规格化的阶数，且偏移码设为 31。此外，由于精度为万分之一，以及 IEEE 754 标准的多 1 位精度，虽然尾数计算时会带来一定误差，但由于尾数设置为 16 位，即实际完成了对 17 位有效数的二进制计算，能满足设计需求。



图 2 24 位浮点数的标准化格式
Fig.2 Normal format of 24 bit floating number.

1.2 前期预处理

规格化浮点数存在 6 种特殊的情况，分别是零、正无穷、负无穷、非数以及正向规格化和负向规格化。这里提出的正向规格化和负向规格化，是为了使整个设计能更好地与 ALTERA 公司提供的 IP 包对接，减少逻辑漏洞存在的可能——因为硬件平台为 ALTERA 公司的 Cyclone III。因此，在进行浮点数乘法前做了一系列的预处理，即当输入为上述几种情况时，则不将乘数输入运算单元，而是直接由预处理将输出赋以相应的处理结果，以保证运算的正确性。

表 1 六种特殊情况的格式(IEEE 754-1985 标准)
Table 1 The six kinds of number format
(the IEEE 754-1985 standard).

Meaning	Sign field	Exponent field	Mantissa field
Zero	Don't care	All 0's	All 0's
Positive denormalized	0	All 0's	Non-zero
Negative denormalized	1	All 0's	Non-zero
Positive infinity	0	All 1's	All 0's
Negative infinity	1	All 1's	All 0's
Not-a-Number (NaN)	Don't care	All 1's	Non-zero

1.3 乘法器整体设计概述

浮点数乘法主要由符号位运算、阶数位运算、尾数位运算以及结果的规格化处理组成^[5]。因为符号位运算可简单地由异或实现，阶数位也是简单加减法的运用，所以尾数部分才是影响整个浮点乘法运算速度和占用资源的主要环节。同时，对结果的规格化处理不仅须在尾数操作中予以实现，还须在最终结果输出时对阶数位进行规格化处理，这样才能保证结果的规格化和正确性。

对尾数部分的运算，在选用了 BOOTH 算法的同时，还采用了 4-2 压缩器和 3-2 压缩器相结合 的树型结构，使得整个数据处理过程更为快速，资源占用更为合理，达到了优化、快捷的目的。

2 尾数运算的实现

乘法器尾数运算部分主要有四部分：使用 BOOTH 算法产生部分积、利用新型树状结构和超前进位加法器实现部分积相加、对计算结果实现规格化处理、对其进行的舍入操作——因为 17 bit 的尾数运算已超过系统所需的 16 位精度，因此，对尾数结果采用默认的舍去操作。

2.1 改进的 BOOTH 算法

改进的 BOOTH 算法也即基 4 的 BOOTH 算法。在目前的乘法器设计中，基 4 的 BOOTH 算法是部分积生成过程中普遍采用的算法。对 N 位符号数乘法而言，常规乘法运算会产生 N 个部分积，但如果对乘数 B 进行基 4 的 BOOTH 编码，则仅需考虑乘数 B 的相邻 3 位(相邻高位、本位和相邻低位)，就可得到一组对应的部分积，且编码后产生的部分积个数仅为以前的一半，大大减少了乘法器的运算量。因此，对于符号数乘法而言，基 4 的 BOOTH 算法既方便又快捷；而对无符号数，则只需对其高位作 '0' 扩展，其他处理方法不变即可。当然，扩展后有可能导致部分积的个数加 1，但它很好地保证了硬件上的一致性，有利于实现对部分积的相加，对于尾数为 16 位(实际运算为 17 位)的标准格式下的无符号数，只需对它做高位 '0' 扩展，即可满足 BOOTH 算法中关于操作数必须为偶数的要求，即尾数达到 18 位，部分积个数相应变为 9 个^[3]。

表 2 改进的 BOOTH 算法编码方式
Table 2 The coding mode based on the improved BOOTH arithmetic.

Multiplier	Product
000	+0
001	+M
010	+M
011	+2M
100	-2M
101	-M
110	-M
111	-0

2.2 部分积相加的树型结构

对于部分积相加，加法器阵列结构是决定乘法器性能的重要因素，不同的阵列结构会影响完成一次乘法所需要的时间。重复阵列(Iterative Array，简称 IA)和 Wallace 树型结构就是最为典型的两种加法器阵列结构。IA 结构规整，易于版图实现，但速度

最慢。理论上,Wallace 树型结构是进行乘法操作最快的加法器阵列结构,但传统的 Wallace 树型结构电路互连复杂,版图实现困难。为此,人们推出了一些连接关系较为简单的树型结构,例如 ZM 树和 OS 树。它们都是将 IA 树分为几段,每段称为子树,子树内部连接采用 IA 结构,而子树间采用树型连接,以此来降低连接复杂度,但是这种方法降低了部分积相加的速度。

人们也尝试了对加法阵列中基本加法单元的改进。Wallace 最早提出的方案中,是以 CSA(进位保留加法器)作为基本单元构建加法阵列的。其基本方法是:通过 CSA 部件,以 3:2 的压缩比对部分积进行逐级压缩,直到最后只产生两个输出为止,再通过进位传递加法器对产生的这两个伪和与局部进位相加得出真正的结果。

经过研究和实践,人们发现 4-2 压缩器(实际上是 5-3 计数器)具有较好的平衡性和对称性,用其作为基本加法单元构成的乘法器在总体性能上具有一定的优势,因此 4-2 压缩器也就成为了目前乘法器中较多采用的加法单元^[6,7]。

本设计是尾数为 16 bit 的标准格式浮点数,共产生了 9 个部分积,因此采用了 CLA(超前进位加法器)与 4-2 压缩器、3-2 压缩器相结合的树型结构。它不仅发挥了 CLA 版图面积小的优点,还体现了压缩器压缩比高、速度快的长处,特别是随着 3-2 压缩器与 4-2 压缩器的结合构造,使针对 9 个部分积特殊情况下的结构具有了良好的对称性和规整性,资源也更为节省^[7]。与其它结构相比,虽然在布局布线上有一定的复杂度,但与传统的 Wallace 树相比,已有相当的改进,特别在结构面积和速度上都达到相对理想的效果。如图 3 所示,即为部分积相加的树型结构图。

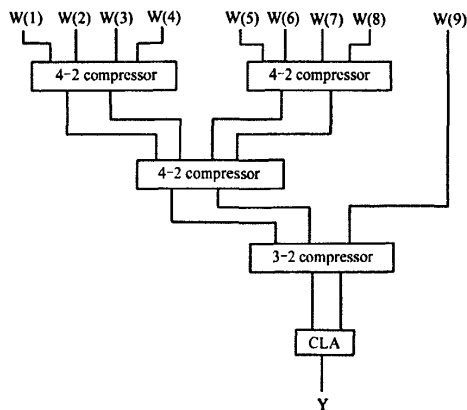


图 3 部分积相加的树型结构图
Fig.3 The tree-structure for adding the products.

3 结果的规格化处理

对于规格化的浮点数乘法,结果的规格化最为重要,保证输入、输出数据的规格化,才能对整个系统实现有效的数据处理。本设计的规格化处理主要体现在阶数位相加和尾数位相乘阶段中。

首先,对阶数位而言,因为规格化处理采用了偏移码,因此,在实现两阶码相加后须减去一个偏移码,同时,在完成尾数位相乘后,还应判断是否须对阶数位做出相应的调整^[3]。

对于尾数位,需考虑计算结果中第一个‘1’的所在位置,并由于乘数采用了 IEEE 754 标准,尾数采用了改进的 BOOTH 算法,故进行尾数运算时须先做“01”的位与操作。因此,若部分积相加后的第 33 位为‘1’,则阶码须进行加一操作,反之,阶码保持不变。最终的尾数即从计算结果中第一个‘1’的后一位算起,取满 16 bit 即可^[3,8]。这样就完成了整个规格化浮点数的乘法运算和规格化处理。但因为并行计算存在逻辑不稳定的部分,也就是所需延时部分,因此,须需要通过对时钟延时进行输出控制,以保证输出结果的正确性。

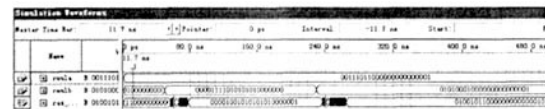


图 4 浮点乘法器的时序仿真
Fig.4 Timing simulation of the float-number-multiplier.

4 结论

以 Cyclone III 芯片 EP3C25F256 为硬件环境和逻辑覆盖测试理论为基础,对此优化乘法器进行了功能和时序仿真,此设计逻辑结果正确、输出满足规格化、运算速度快,能满足同步加速器磁场控制数字化改造所需的高速数字处理能力,且为后期磁场控制系统的数字化改造打下基础。

表 3 为该优化乘法器与 ALTERA 公司的 IP 核中的多种算法比较的结果,由表 3 可知:

- 1) 针对 ALTERA 公司的芯片,只有用该公司的 IPcore 才能使用芯片提供的专业元件,增加芯片的利用率;
- 2) 在不使用专业元件的前提下,本设计的优化乘法器远小于 IPcore 提供的乘法器的资源占用率,可以证明此乘法器在资源的使用上更具参考性;
- 3) 对数据更新率在 >25 M 且为大批量数据处理时,IPcore 具有的流水线功能较为适用,但对于所处理有效数据更新率 <25 M 的情况,此乘法器的

实时性更好,因其全程时间远小于 IPCore 提供的各级流水线运算时间,因此,此乘法器是完全适用于 AD 采样率<25 M 的控制系统当中;

4) 此外,此设计的乘法器不存在 IPcore 的黑盒子问题,代码完全可读,不存在盲区,利于今后的进一步优化和改进。

表 3 算法进行比较的结果
Table 3 Comparison of different algorithms.

	专有元器件 Professional component	资源 Resource	速度 Speed	知识产权 Intellectual property right
IP 核: 5 级流水线 IPcore:5-step pipelining	Yes	Le:256/24,624 EM_9bit:7/132	$\Delta t = 148.26 \text{ MHz}$ $T = 60 \text{ ns}$	No
IP 核: 5 级流水线 IPcore:5-step pipelining	No	Le:902/24,624 EM_9bit:0/132	$\Delta t = 124.24 \text{ MHz}$ $T = 60 \text{ ns}$	No
IP 核: 11 级流水线 IPcore:11-step pipelining	Yes	Le:323/24,624 EM_9bit:7/132	$\Delta t = 200 \text{ MHz}$ $T > 55 \text{ ns}$	No
IP 核: 11 级流水线 IPcore:11-step pipelining	No	Le:1,119/24,624 EM_9bit:0/132	$\Delta t = 154.82 \text{ MHz}$ $T > 55 \text{ ns}$	No
本文乘法器 This multiplier	No	Le:881/24,624 EM_9bit:0/132	$\Delta t = T = 40 \text{ ns}$	Yes

参考文献

1 龙银东, 乔卫民, 赵龙, 等. 核技术, 2008, 31(8): 603-607
LONG Yindong, QIAO Weimin, ZHAO Long, *et al.* Nucl Tech, 2008, 31(8): 603-607

2 http://www.altera.com/literature/rn/rn_qts_80sp1.pdf

3 王克义, 鲁守智, 蔡建新, 等. 微机原理与接口技术教程. 北京: 北京大学出版社, 2004
WANG Keyi, LU Shouzhi, CAI Jianxin, *et al.* Microcomputer Theory and Interface Technology. Beijing: Publishing House of Beijing University, 2004

4 IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Std 754-1985

5 沈明发, 易清明, 黄伟英, 等. 暨南大学学报(自然科学报), 2002, 23(5): 19-24

SHEN Mingfa, YI Qingming, HUANG Weiyang, *et al.* J Jinan Univ (Nat Sci), 2002, 23(5): 19-24

6 雷伏容. VHDL 电路设计. 北京: 清华大学出版社, 2006
LEI Furong. VHDL circuit design. Beijing: Publishing House of Tsinghua University, 2006

7 吴琼, 肖梓祥. 现代电子技术, 2003, (4): 46-48
WU Qiong, XIAO Zixiang. Mod Electron Tech, 2003, (4): 46-48

8 Armstrong J R, Gray F G 著. 李宗伯, 王蓉晖, 王蕾, 等, 译. VHDL 设计表示和综合(第二版). 北京: 机械工业出版社, 2003
Armstrong J R, Gray F G LI Zongbo, WANG Ronghui, WANG Lei, *et al.*, trans. VHDL design representation and synthesis (Second ed). Beijing: Machinery Industry Press, 2003

Improved multiplier used in the state-matrix for magnetic control of a synchrotron

LONG Yindong^{1,2} QIAO Weimin¹ LIN Feiyu^{1,2} JING Lan¹

1(Institute of Modern Physics, Chinese Academy of Sciences China, Lanzhou 730000, China)

2(Graduate University of Chinese Academy of Sciences China, Beijing 100049, China)

Abstract The magnetic control of a synchrotron is a high precision real time system. With the development of digital processing, the HIRFL-CSR project (Cooler Storage Ring of the Heavy Ion Research Facility in Lanzhou) has higher requirements on floating multiplier's speed and used-resource. In this paper, we report the improved floating multiplier realized by the VHDL and the Booth algorithm, based on the tree-structure consisting of the 4-2 and 3-2 compressors. It takes the EP3C25F256 as the hardware environment to get the timing-emulate and function-emulate. Comparing with other designs, this floating multiplier has better in correction and speed.

Key words LSI, Floating number, Float-number-multiplier, Specification
CLC TP332.2⁺1