

一种基于 Dadda 树的乘法器设计

李路路, 何 春, 宗竹林, 章凌宇

(电子科技大学 电子科学与技术研究院, 四川 成都 611731)

摘 要: 在基带信号处理芯片中, 面积和速度是两个关键的指标. 文中在改进的 booth 算法基础上, 采用了 Dadda 树压缩算法, 通过对压缩器基本单元的改进, 同时对符号位和尾部零填充进行优化设计; 不仅保持了 Wallace 树结构的并行计算优势, 而且面积上也得到了很大的改善; 同时相对于 Wallace 树结构的规则结构也更利于版图设计. 压缩结果采用了多层 CLA 块技术, 使得乘法器的速度得到进一步的提高. 在 $0.13\mu\text{m}$ 的 SMIC 八层金属 CMOS 工艺下, DC (Design Compiler) 综合结果表明, 芯片面积为 $20633.59\mu\text{m}^2$, 最大延迟仅为 3.00ns .

关键词: 基 4-booth 编码; Dadda 树结构; 进位保留加法器; 4:2 压缩器; 多层 CLA 块技术

中图分类号: TP332.2

文献标识码: A

文章编号: 1000-7180(2011)05-0176-04

A Kind of Multiplier Design Based on Dadda Tree

LI Lu-lu, HE Chun, ZONG Zu-lin, ZHANG Ling-yu

(Research Institute of Electronic Science and Technology of UESTC, Chengdu 611731, China)

Abstract: In the baseband signal processing chip, the area and speed are two key indicators. In this paper, we present a design which is based on the modified booth algorithm, using the Dadda tree compression algorithm with the improvement of the basic unit, while using the method of the sign bit and zero-filling of the tail to optimize the design, not only to maintain the Wallace tree structure of the advantages of parallel computing, and area also been greatly improved. Comparing to the Wallace tree structure, the rules of the structure is more conducive to layout. With compression results using a multi-CLA block technology, the multiplier's speed is further improved. Under the SMIC $0.13\mu\text{m}$ CMOS process with the eight-layer metal, the synthesis result of this design by DC (Design Compiler) showed that the chip area is $20633.59\mu\text{m}^2$, and the maximum delay is only 3.00ns .

Key words: base 4-booth encoding; Dadda tree structure; CSA; 4:2 compressor; multi-block CLA technique

1 引言

在 GPS 卫星接收机基带处理芯片中, 需要对大量的数据进行处理. 由于 GPS 定位要求的实时性要求非常高, 因此在基带处理芯片中对乘法器的性能提出了更高的要求. 文中研究了有/无符号 16×16 乘法器的体系结构, 根据乘法器实现原理, 采用改进的 booth 算法^[1], 优化了部分积的产生过程. 同时, 通过一种规则的 Dadda 树结构实现部分积的压缩, 使得该乘法器的结构更加规则, 保证了速度和面积上的优势. 文中采用快速进位链加法器算法对压缩的结果进行处理, 进一步提高了乘法器的性能.

2 乘法器的设计原理

乘法器的设计实现流程如图 1 所示. 可以看出, 乘法器的实现流程包括三个环节: (1) 部分积的产生——采用改进的 booth 算法进行设计, 将被乘数和乘数压缩成若干部分积; (2) 部分积的压缩——采用 Dadda 树压缩算法, 将得到的部分积压缩为和项和进位项; (3) 加法电路——将得到的和项和进位项压缩为最终的乘积结果.

2.1 改进的基 4-booth 编码算法

改进的基 4-booth 编码算法根据乘数的相邻三位进行编码来决定对被乘数的操作.

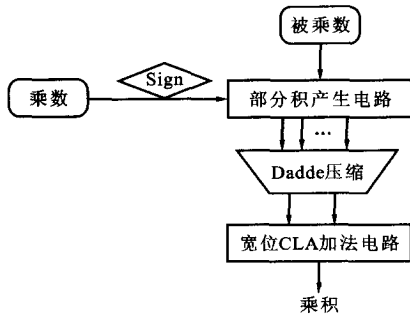


图 1 乘法器的设计流程

首先分析有符号数的乘法操作,设乘数为 A ,被乘数为 B ,乘积为 C 。则 A 可以通过式(1)表示^[2]为:

$$\begin{aligned} A &= -A_{15} \times 2^{15} + \sum_{k=0}^{14} A_k \times 2^k \\ &= \sum_{k=0}^7 (-2A_{2k+1} + A_{2k} + A_{2k-1}) \times 2^{2k} \\ &= \sum_{k=0}^7 E_k \times 2^{2k} \end{aligned} \quad (1)$$

那么乘积结果 C 通过式(2)表示为:

$$\begin{aligned} C &= B \times A \\ &= B \times (-A_{15} \times 2^{15} + \sum_{k=0}^{14} A_k \times 2^k) \\ &= B \times \sum_{k=0}^7 (-2A_{2k+1} + A_{2k} + A_{2k-1}) \times 2^{2k} \end{aligned} \quad (2)$$

从式(2)可以看出有符号 16×16 乘法产生了 8 个部分积。下面对于无符号的乘法过程进行推算,设乘数为 A ,被乘数为 B ,乘积为 C 。乘数 A 表示如式(3)所示:

$$\begin{aligned} A &= \sum_{k=0}^{15} A_k \times 2^k \\ &= A_{15} \times 2^{16} - A_{15} \times 2^{15} + \sum_{k=0}^{14} A_k \times 2^k \\ &= A_{15} \times 2^{16} + \sum_{k=0}^7 E_k \times 2^{2k} \end{aligned} \quad (3)$$

从式(3)可以看出无符号乘法产生了 9 个部分积,其中 8 个部分积的产生逻辑与有符号数的相同,另一个部分积单独处理。

通过上述的数学分析,基 4-booth 改进算法操作如表 1 所示。其中设乘数为 A ,被乘数为 B ,乘数中相邻的三位为 $A_{2k+1}, A_{2k}, A_{2k-1}$, X 表示与 B 的乘积因子。

2.2 部分积压缩算法

作为乘法器的核心部分,该模块负责将生成的部分积按权值进行错位相加。典型的结构^[3]有三种:线性结构、Wallace 树结构、Dadda 树结构。Dadda 树

结构不仅保持了 Wallace 树结构的并行计算优势,而且具有相对规整的结构,从一定程度上降低了连线的复杂度,因此在芯片的设计上是一个理想的选择。

表 1 改进的 Booth 算法操作方式

A_{2k+1}	A_{2k}	A_{2k-1}	E_k	X	对求和的影响
0	0	0	0	0	取 0 相加
0	0	1	1	1	取 B 相加
0	1	0	1	1	取 B 相加
0	1	1	2	2	对 B 左移相加
1	0	0	-2	-2	取 B 的补码并左移
1	0	1	-1	-1	取 B 的补码相加
1	1	0	-1	-1	取 B 的补码相加
1	1	1	0	0	取 0 相加

Dadda 树结构的基本设计单元是一个 4:2 压缩器。该压缩器是一个可以配置的进位保留加法器(CSA)^[4],可以根据压缩过程中的需要进行扩展。其实现的电路结构如图 2 所示。

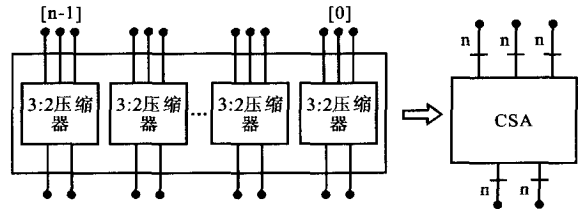


图 2 进位保留加法器基本结构

CSA 内部电路的基本组成单元是一个 3:2 压缩器,即全加器。通过分析基本单元的结构,可以在资源上进行优化,优化策略如图 3 所示。

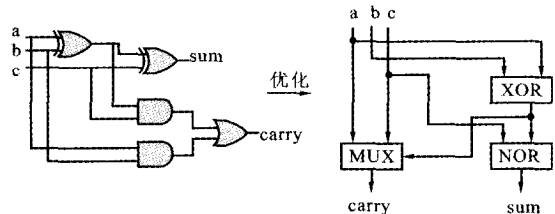


图 3 基本单元的优化策略

由于需要将所生成的部分积在后续的部分积压缩过程中按照其权值错位相加,因此,需要对部分积进行符号位填充,以满足加法的需要。但是,如果将所有部分积都扩展到与最终乘积相同的位数,则会对硬件造成不必要的浪费。

当乘法为无符号乘法时,符号位填充为 0,不需考虑。故只需讨论前 8 个部分积在乘法为有符号乘

法时的情况,以第一个部分积 $PP_0 = B \times P_0$ 为例,设符号填充位为 $n, n = 0, 1$.

当符号位为 1 时,完整的部分积^[3]扩展为 32 位,即:

$$PP'_0 = \sum_{n=17}^{31} 2^n + PP_0 = 2^{32} - 2^{17} + PP_0 \quad (4)$$

由于最多只有 32 位部分积,所以第 33 位属于

溢出位,也就是说由 booth 算法产生的部分积只需要扩展到 18 位即可满足加法错位压缩要求.

对 Dadda 树压缩的原理进行设计,考虑到 4:2 压缩器的结构和面积的优化,对不同权值的部分积进行适当的符号位扩展和尾部零填充,具体实现过程如表 2 所示.

表 2 Dadda 树压缩原理实现过程

		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSA 11	pp0																																
	pp1																																
	pp2																																
CSA 21	pp3																																
	s11																																
	c11																																
CSA 12	pp4																																
	pp5																																
	pp6																																
CSA 22	pp7																																
	s12																																
	c12																																
CSA 3	s21																																
	c21																																
	s22																																
CSA 4	s3																																
	c3																																
	c22																																
CSA 5	pp8																																
	s4																																
	c4																																
s_wait																																	
c_wait																																	

表 2 说明了 dadda 树设计思想,将有符号数的 8 个部分积部分分为两部分,分别采用 18 位的 4:2 压缩器进行中间结果的计算.前四个部分积 pp0~pp3 计算得到 s21 和 c21 两个中间结果,后四个部分积 pp4~pp7 得到 s22 和 c22;最后将得到的四个中间结果 s21~c22 通过一个 25 位的 4:2 压缩器得到有符号部分积的压缩结果 s4 和 c4.

在压缩过程中充分利用了基本单元 3:2 压缩器,设计过程中充分考虑了符号位的有效填充,同时避免了过多的尾部零填充.图 4 展示了 Dadda 树结构部分积压缩器实现的电路结构示意图,图中虚线框内为基本的 4:2 压缩器.

2.3 CLA 宽位加法器设计

对于上述得到的两个压缩结果,需要进一步通过加法器才能得到最终的结果.文中基于超前进位(CLA)思想,采用多层 CLA 块技术设计 32 位的加法器.

进位产生项逻辑^[4]表达式如式(5)所示:

$$g = a \& b \quad (5)$$

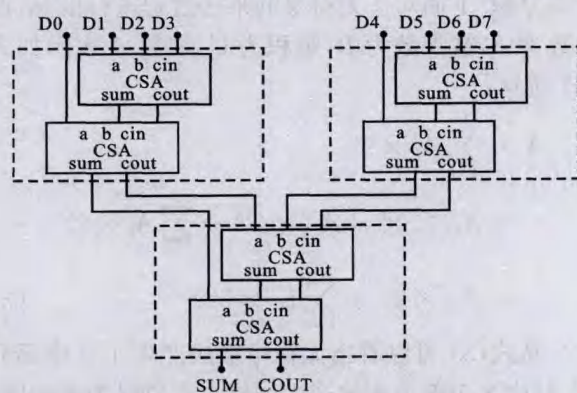


图 4 dadda 树结构实现框图

传播项逻辑表达式如式(6)所示:

$$p = a \oplus b \quad (6)$$

由此可得基本单元的传播项 $g[0], g[1], g[2], g[3]$ 以及传播项 $p[0], p[1], p[2], p[3]$.

进位链产生逻辑表达式如式(7)所示:

$$c[i+1] = g[i] + p[i] \& c[i] \quad (7)$$

由此可得四个进位项 $c[1], c[2], c[3]$ 和 $c[4]$, 如果对 $n(n > 4)$ 位的加法器仍然采用式(7)进行迭

代运算, 即 $c[4]$ 将作为下一级的进位输入, 那么就会由于通过最长时延路径的门数的增加而引起太长的硬件延时^[4]. Von Neumann 和其他人的研究表明最长进位链的平均进位链长度为 $\log_2(n)$ ^[5].

对于 32 位的加法器的平均长度为 5, 因此没有必要将进位电路的长度扩展在整个字长上, 可以分割成几个较小的部分^[4]. 多层 CLA 技术就是基于这一原理的, 在传播项和进位项产生的同时得到了块产生信号 g 和块传播信号 p , 其逻辑表达式如式(8)和式(9)所示.

$$g = g[3] + p[3] \& g[2] + p[3] \& g[2] \& p[1] + p[3] \& g[2] \& p[1] \& p[0]$$

(8)

$$p = p[3] \& p[2] \& p[1] \& p[0]$$

(9)

多层 CLA 块技术 32 位结构图如图 5 所示.

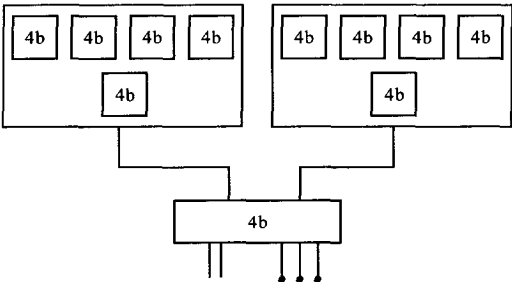


图 5 多层 CLA 块技术 32 位加法器结构图

该设计中采用了 3 层 CLA 技术, 通过下一层的进位输出作为本层下一个 CLA 块的进位.

3 测试结果及性能分析

该相关器的测试结果如图 6 所示. 对于每一个数据采用随机数进行测试, 同时对有符号和无符号运算进行比较, 其中 sign 为 0 时表示无符号运算, sign 为 1 为有符号数运算. 测试结果表明, 该设计完成了预期功能.



图 6 Dadda 树乘法器测试结果

设计在 SMIC 0.13 μ m 八层金属的 CMOS 工艺流程下实现, 使用 design compile 进行综合, 得到的综合结果如表 3 所示.

表 3 综合结果

工艺库	条件	延迟/ns	面积/ μm^2
SMIC 0.13	Typical	3.00	20633.59

4 结束语

文中介绍了一种基于 Dadda 树的 GPS 基带芯片中乘法单元的设计原理和优化策略, 分析了有/无符号乘法的压缩算法. 从基本单元, 符号位的扩展以及尾部零填充三个方向对压缩器进行优化, 最后阐述了基于 CLA 的宽位加法器的设计. 在 SMIC 0.13 μ m 八层金属的 CMOS 工艺流程下综合表明, 该设计大大减少了路径延迟, 进一步降低了芯片的面积, 同时规则的压缩结构也便于芯片设计.

参考文献:

[1] Rizalafande Che Ismail, Razaidi Hussin. High performance complex number multiplier using booth-wallace algorithm[C]// Proceedings of International Conference on Semiconductor Electronics. Kuala Lumpur: IEEE, 2006:786-790.

[2] Li Qingzheng, Liang Guixuan, Bermak Amine. A high-speed 32-bit signed/unsigned pipelined multiplier[C]// Proceedings of the 5th IEEE International Symposium on Electronic Design, Test and Applications, DELTA. Washington, DC, USA: IEEE, Computer Society, 2010: 207-211.

[3] 李楠, 喻明艳. 16 $\times$ 16 快速乘法器的设计与实现[J]. 微电子学与计算机: 2008 年, 25(4): 156-159.

[4] John P. Uyemura. 超大规模集成电路与系统导论[M]. 周润德, 译. 北京: 电子工业出版社, 2005.

[5] Behrooz Parhami. Computer arithmetic [M]. New York: Oxford University Press, 2000.

作者简介:

李路路 男, (1984-), 硕士研究生. 研究方向为通信与信号处理, 集成电路设计.

何 春 女, (1972-), 副研究员, 硕士研究生导师. 研究方向为通信与信号处理, 集成电路设计.

宗竹林 男, (1983-), 博士研究生. 研究方向为通信与信号处理, 集成电路设计.