第 34 卷 第 17 期 计 算 机 工 程 2008 年 9 月
Vol.34 No.17 Computer Engineering September 2008
·工程应用技术与实现· 文章编号：1000—3428(2008)17—0229—03 文献标识码：A 中图分类号：TP303

基于跳跃式 Wallace 树的低功耗 32 位乘法器

李 伟，戴紫彬，陈 韬
(解放军信息工程大学电子技术学院，郑州 450004)

摘 要：为了提高乘法器的综合性能，从 3 个方面对乘法器进行了优化设计。采用改进的 Booth 算法生成各个部分积，利用跳跃式 Wallace 树结构进行部分积压缩，通过改进的 LING 加法器对压缩结果进行求和。在 FPGA 上进行验证与测试，并在 0.18 μm SMIC 工艺下进行逻辑综合及布局布线。结果表明，与采用传统 Wallace 树结构的乘法器相比，该乘法器的延时减少了 29%，面积减少了 17%，功耗降低了 38%，能够满足高性能的处理要求。

关键词：Booth 算法；跳跃式 Wallace 树；乘法器；LING 加法器

Low-power 32-bit Multiplier Based on Leapfrog Wallace Tree

LI Wei, DAI Zi-bin, CHEN Tao
(Institute of Electronic Technology, PLA Information Engineering University, Zhengzhou 450004)

【Abstract】 In order to improve performance of multiplier, this paper adopts modified Booth algorithm to generate partial product, proposes the leapfrog Wallace tree architecture to compress partial product, and introduces the modified LING adder to compute the final sum of the result of Wallace tree. The design is realized by using Altera's FPGA. Synthesis, placement and routing of 32-bit multiplier are accomplished on 0.18 μm SMIC process. Compared with conventional multiplier with traditional Wallace tree, the multiplier reduces the multiplication time, the power dissipation and the area of multiplier by 29%, 38% and 17%.

【Key words】 Booth algorithm; leapfrog Wallace tree; multiplier; LING adder

1 概述

在专用集成电路设计中，面积小但功能强大的CPU内核可以为设计提供很大的方便。而乘法器是CPU中的一个重要部件，其速度和面积直接影响整个CPU的性能。自文献[1]提出Booth乘法器后，就不断有人提出基于Booth算法的乘法器设计，但乘法器的处理过程大致相同，都是先生成部分积然后进行相加。本文针对乘法器的设计特点，从改进Booth二阶算法、跳跃式Wallace树[2]和LING加法器[3]3 个方面对乘法器进行了优化设计。在硬件实现上，改进Booth二阶算法编码电路简单，部分积的产生速度快、功耗小；与传统Wallace树相比，跳跃式Wallace树部分积的压缩速度快，减少了电路内部的伪翻转，动态功耗得到明显的降低；LING加法器与常见的加法器相比在速度上具有明显的优势，能够有效实现对压缩结果的相加，使整体性能得到明显提升。实验结果表明，本文设计的乘法器可以满足高性能的处理要求。

2 Booth 算法分析

目前很多设计都采用高基的 Booth 算法，以求得到较好的加速效果。但 Booth 算法在带来较少的部分积累加次数的同时，也由于部分积不能直接通过移位生成而给 Booth 编码的设计带来了很多问题。如表 1 所示，本文针对 32 bit 乘法运算对 Booth 算法进行了分析比较。

表 1 Booth 算法分析与比较

Booth 算法类型	部分积个数	所需额外电路
无任何算法	32	无
Booth3 算法	11	移位、取反、3M
Booth4 算法	8	移位、取反、3M、5M、6M、7M
冗余 Booth3 算法	11+1(常数)+3(进位合并)=15	移位、取反、3M
改进 Booth2 算法	17	移位、取反

Booth 算法在部分积压缩上效果明显，同时引入了额外的电路，如移位、取反、倍数电路，其中，移位、取反电路可以通过简单的硬连线与取反逻辑实现，采用 0.18 μm SIMC 工艺，其带来的延迟对电路的影响很小，但倍数生成电路(如 3M, 5M, 6M, 7M)则不易实现而且延迟较大。

Booth4 算法产生 8 个部分积，对部分积的压缩效率最好，但所需的额外电路最多，引入了 3M, 5M, 6M, 7M 倍数电路。对于 3M 倍数电路的实现，目前通常采用进位传递加法器通过 1M+2M 来实现，延迟较大，并造成面积的增大。

冗余 Booth3 编码产生 11 个部分积，并根据进位权值的不同，将所有进位信息合并为 3 个部分积，再加上一个抵消数，共 15 个部分积。其部分积的压缩效果并不明显，而且还需要 3M 倍数的生成电路。

Booth3 算法产生 11 个部分积，但也需要引入 3M 倍数电路。处理方法与 Booth4 算法相同，每进行一次 3M 倍数运算都要引入一个 32 bit 的加法器，其产生的延迟也成为整个乘法关键路径中的一部分。同时资源消耗很大。

改进Booth二阶算法[4]在 32 bit的乘法运算中产生 17 个部分积，虽然压缩效果不如前几种Booth算法，但在Booth编码电路中不涉及倍数产生电路，只是简单的连线延迟。相对于其他Booth算法，在硬件实现方面具有速度快、面积小和功耗低的特点。而且Wallace树是一种并行的压缩方式，多产生的部分积不会增加整体压缩树的延迟。因此，对于 32 bit的乘法，从硬件实现上考虑，改进Booth二阶算法在速度、面积

作者简介：李 伟(1983－)，男，硕士研究生，主研方向：信息安全，体系结构；戴紫彬，教授、博士；陈 韬，讲师、硕士
收稿日期：2007-10-10 **E-mail：**try_1118@163.com

与功耗方面相比其他Booth算法具有明显的优势。

3 跳跃式压缩树的分析与实现

3.1 延迟分析

目前在乘法器的设计中，压缩树的压缩方法大多采用进位保存加法器(CSA)或 4-2 压缩器实现，在这 2 种压缩器中，电路都是由简单的与或非门所构成。由于各种门电路存在不同的延迟，使得电路在有效信号到来之前一直处于不定状态，这不仅造成延迟的增大，而且电路无效翻转的增多也导致了功耗的增大。为了避免上述情况，只有尽可能地让各级电路的输入信号同时到达，才能使电路在较短的时间内稳定下来。本文在研究了基本门电路的延迟后得出，CMOS门电路的结构非常简单，便于构造和分析，但异或门的构成相对复杂，设 1 级门的延迟为 1 td，则与、或、与非、或非逻辑门的延迟近似为 1 td，异或门的延迟为 2 td[5]。

根据以上结论，对 CSA 的输出延迟进行分析。CSA 的电路结构如图 1 所示，有 3 个输入端 *E, F, G*；输出端为进位 *C* 与伪和 *S*，进位 *C* 的延迟为 2 td，伪和 *S* 的延迟为 4 td。在 4-2 压缩器中，采用优化后的电路结构见文献[6]，整体 4-2 压缩器的延迟相当于 CSA 的 1.5 倍。4-2 压缩器的输出进位 *C* 的延迟为 4 td，伪和 *S* 的输出延迟为 6 td。

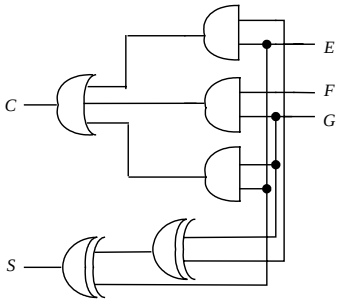

图 1 CSA 电路结构

3.2 跳跃式 Wallace 树结构

Wallace 树结构如图 2 所示，它是目前普遍采用的压缩方式，能够有效地对部分积进行压缩，压缩速度快。

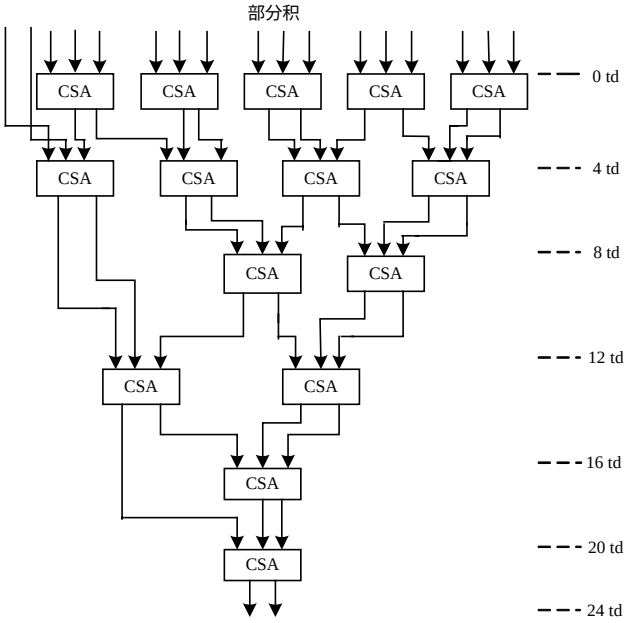

图 2 传统的 Wallace 树结构

在这种结构中，由于进位 *C* 与伪和 *S* 的输出延时不同，输出较快的进位信号 *C* 要等伪和 *S* 的输出稳定后才能进入下一级压缩电路，造成了电路的无效翻转。结合文献[2]中 54 bit 乘法器的设计，给出了 32 bit 乘法器的跳跃式 Wallace 树结构，如图 3 所示。

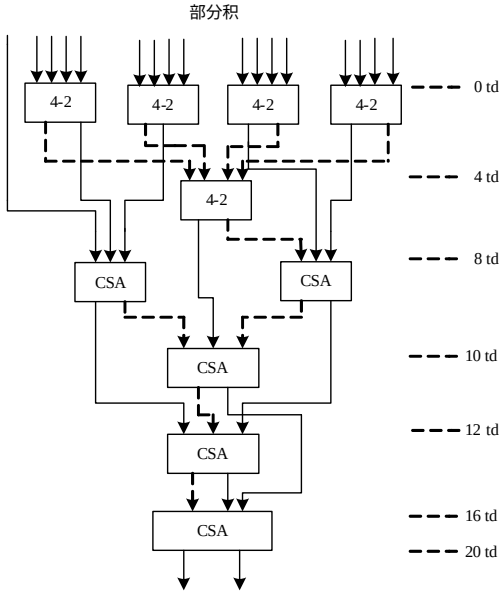

图 3 跳跃式的 Wallace 树结构

考虑到压缩树的电路面积，采用了 4-2 压缩器与 CSA 的混合结构，针对进位端 *C* 与伪和 *S* 输出延迟不同的问题，跳跃式 Wallace 树结构将压缩器进位输出 *C* 与伪和 *S* 分开送入下一级压缩电路，使得数据能够同时到达压缩电路的每一级，减少了电路的无效翻转，从而降低了电路的动态功耗，同时，由于跳跃式 Wallace 树结构中不存在数据等待的问题，与传统的 Wallace 树相比，其延时减少了 4 tb，加快了对部分积的压缩。

4 末级加法器优化

4.1 加法器分析

经过跳跃式 Wallace 树结构后，得到 2 个输出数据，须经过一级加法器得到最终结果。在设计中，加法器的延迟往往会影响整个乘法电路的性能，特别是在 32 bit 的乘法运算中，末级加法需要 64 bit 的加法器实现，其延迟更是整个乘法电路关键路径的一个重要部分，因此，加法器的优化格外重要。

在加法器的设计中，关键问题在于进位间的延迟，如常见的CPA加法器就是通过提前计算出各位的进位数加快运算速度的。目前普遍采用的方法是将 4 bit的CPA级联起来构成更高位数的加法器。但在这种方式中，CPA级间进位的传递成为了主要的延迟，特别是在 64 bit的长位数加法器中，延迟问题更加突出。为了得到更好的性能，本设计采用了LING加法器结构[3]。LING加法器是基于CLA加法器[7]改进设计的。在LING加法器中，CLA中的本地进位和进位传递 (*G*和*P*)被有相似功能的信号(分别称为*H*和*I*)代替，通过改进算法减少了产生*H*和*I*信号的执行时间，特别是*H*信号的产生，体现出了LING加法器的优势。其原理如下：

传统的 4 bit CLA 本地进位公式为

$$G = g_0^3 = g_3 + p_3g_2 + p_3p_2g_1 + p_3p_2g_0$$

其中， $g = ab,\ p = a + b$ 。

因为

$$g_3 = a_3b_3 = (a_3b_3)(a_3 + b_3) = p_3^+ g_3$$

$$p_3 = p_3^+ = a_3 + b_3$$

所以

$$G = p_3^+ g_3 + p_3^+ g_2 + p_3^+ p_2 g_1 + p_3^+ p_2 p_1 g_0 = p_3^+ (g_3 + g_2 + p_2 g_1 + p_2 p_1 g_0) = p_3^+ H \quad (1)$$

将 g 与 p 代入式(1)得

$$H = a_3b_3 + a_2b_2 + a_1a_2b_1 + a_1b_1b_2 + a_0a_1a_2b_0 + a_0a_1b_0b_2 + a_0a_2b_0b_1 + a_0b_0b_1b_2 \quad (2)$$

式(2)定义了一种新的信号类型 H，称为伪进位产生信号，用来代替 CLA 中的 G 信号。在 CLA 加法器中，G 信号的产生往往需要先产生 p 和 g 信号，如果从输入端直接产生 G 信号，同样将 g 与 p 代入 G 的表达式后，得到一个 15 项的线或逻辑，其中有 8 项都为五输入的与逻辑，相比之下，LING 加法器中伪进位信号 H 的产生可以直接从输入端得到，只需 8 项线或逻辑，每项中最多为 4 输入的与门，由此得出产生 H 信号的延迟远小于产生 G 信号的延迟，但这些信号的使用方式与 G 与 P 是一致的。当需要真正的 G 或 P 时，可以通过 H 和 I 再加上一个本地信号来产生。其他结构与传统的 CLA 非常相似，LING 加法器的关键路径上包含 3 个或非门和一个异或门的延迟，而 CLA 加法器的关键路径上为 4 个或非门和一个异或门的延迟，这就使得 LING 加法器具有更好的性能。

4.2 加法器性能比较

本文将 LING 加法器与常见的加法器从延迟、面积 2 个方面进行了分析，并在 Synopsys 的 DC(Design Compiler)上进行了逻辑综合与优化，其综合结果如表 2 所示。

表 2　64 bit 加法器的性能分析

加法器类型	最大延迟/ns	面积/μm²
CPA	2.71	6 759
Brent-Kung Adder[8]	2.41	6 905
Kogge-Stone Adder[9]	2.14	9 875
CLA	2.17	7 106
Synopsys Adder	2.21.	6 831
LING	2.07	6 988

由表 2 可以得出，LING 加法器在延迟上较其他加法器具有明显的优势，在面积上比 Brent-Kung, CLA 加法器小，综合面积、延迟 2 个方面考虑，在性能要求较高的场合，LING 加法器能够更好地满足需求。

5 乘法器性能测试

将本文的乘法器在 DC 上进行逻辑综合、优化，并与其他类型的乘法器进行分析比较，其综合结果如表 3 所示。

表 3　32 bit 乘法器的性能比较

乘法器类型	最大延迟/ns	面积/μm²
传统 Wallace 树+CPA	4.28	98 764
跳跃式 Wallace 树+CPA	4.05	80 964
跳跃式 Wallace 树+CLA	3.44	82 324
Synopsys Mult	3.21	72 269
跳跃式 Wallace 树+LING	3.01	81 765

可以看出，本文的设计由于采用了的跳跃式结构，压缩树的延迟明显减少，而且在末级加法器方面采用了延迟小的 LING 加法器，使得整体乘法关键路径在延迟方面与其他结构的乘法器相比具有很大的优势，而且面积与传统 Wallace 树乘法器相比明显地减少了。此外，本设计在功耗方面也具有较优的性能，该乘法器在 Magma 公司的 Blast Rail 上进行了功耗分析，并与传统的乘法器进行了对比，结果如表 4 所示。

表 4　乘法器功耗分析

乘法器	源漏功耗 /nW	内部功耗 /μW	动态功耗 /mW	总功耗 /mW
跳跃式 Wallace+LING	148.1	32.8	0.528	0.560
传统 Wallace 树+CLA	387.6	104.6	0.820	0.904

本文所提出的乘法器与常规的乘法器相比，功耗降低了 38%，其原因如下：

(1)采用了 Booth2 阶算法，编码电路简单，电路功耗小。

(2)跳跃式 Wallace 树结构与常规 Wallace 树相比，减少了电路的无效翻转，降低了电路的动态功耗。

(3)LING 加法器的关键路径相比 CLA 加法器减少了一级逻辑门。

6 结束语

本文从Booth算法、跳跃式Wallace树、LING加法器 3 个方面对 32 bit乘法器进行了优化设计，并在 0.18 μm SIMC 工艺上进行了逻辑综合与优化。结果表明，其乘法延时为 3.01 ns，面积为 0.08 mm²，功耗为 0.561 mW。核心频率达到 330 MHz，与传统的Wallace树相比，速度提高了 29%，面积减少了 17%，功耗降低了 38%，综合性能与传统的乘法器相比具有明显的优势，因此，能够满足各种高性能CPU的要求。

参考文献

[1] Booth A. A Signed Binary Multiplication Technique[J]. Quarter Journal of Mechanics and Applied Mathematics, 1951, 4(2): 236-240.

[2] 孙海珺, 邵志标, 迟晓明, 等. 基于冗余算法和跳跃式结构的 54 位乘法器的研究[J]. 西安交通大学学报, 2006, 40(2): 191-194.

[3] Bewick G W. Fast Mulplication Algorithms and Implementation[D]. Stanford, USA: Stanford University, 1994.

[4] MacSorley O L. High-speed Arithmetic in Binary Computers[J]. Proceedings of the IRE, 1961, 49(1): 67-91.

[5] Shivaling S M, Poras T B. High Performance Low Power Array Multiplier Using Temporal Tiling[J]. IEEE Transactions on Very Large Scale Integration(VLSI) Systems, 1999, 7(1): 121-124.

[6] 赵忠民, 林正浩. 一种改进的 Wallace 树型乘法器的设计[J]. 电子设计应用, 2006, 7(8): 113-116.

[7] 徐东明. 一种快速的改进 CLA 加法器设计[J]. 西安邮电学院学报, 2005, 10(1): 6-9.

[8] Brent P R, Kung T H. A Regular Layout for Parallel Adders[J]. IEEE Transactions on Computers, 1982, 32(3): 260-264.

[9] Kogge P, Stone H. A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations[J]. IEEE Transactions on Computers, 1973, 22(8): 783-787.