16×16 快速乘法器的设计与实现

李 楠,喻明艳
(哈尔滨工业大学 微电子中心,黑龙江 哈尔滨 150001)

摘 要:为得到高性能的乘法器,本设计通过改进的 Booth 算法产生部分积,用一种 Wallace 树结构压缩部分积,并使用减少符号位填充和减少尾部 0 填充两种方法有效地减小了部分积压缩器的面积,最终通过超前进位加法器组得到乘积结果.采用 SMIC 0.18μm 工艺库,由 DC(Design Compiler)综合,时间延迟可达到 4.62ns,面积为 23 837μm^2.

关键词:乘法器;改进 Booth 算法;Wallace 树;面积优化;CLA 组
中图分类号:TP332.2 **文献标识码**:A **文章编号**:1000-7180(2008)04-0156-04

16×16 High-Speed Multiplier Design & Implementation

LI Nan, YU Ming-yan
(Microelectronics Center, Harbin Institute of Technology, Harbin 150001, China)

Abstract: To get a high-performance multiplier, this paper presents a design, which generates partial products by modified Booth algorithm, compresses them using a Wallace tree structure and gets the final product with a CLA array. We optimize the compressor by means of reducing the amount of symbol padding, and reducing amount of padding "0" behind addends. Using the SMIC 0.18μm process, the synthesis result of this design by DC (Design Compiler) shows that the delay can be reduced to 4.62ns and the area is 23 837μm^2.

Key words: multiplier; modified Booth algorithm; Wallace tree; area optimization; CLA array

1 引言

在语音信号、视频信号及图像处理等数字信号处理芯片中,出于计算需要,迭代算法常常需要向量的点积,从而使乘法器处于关键路径,它的性能也就显得至关重要[1].虽然不断有关于 32 位、64 位乘法器的研究出现,但由于大量的媒体信号处理只需 16 位运算就能胜任,因此对 16 位乘法器的研究仍有着相当的应用价值.

2 乘法器结构

乘法器基本工作原理大体分为三个步骤:(1)先将二进制的被乘数与乘数的每一位分别相乘,得到与乘数的位数相同个数的部分积;(2)将得到的部分积按权值错位相加,进行部分积压缩;(3)一级加法,通常采用一个超前进位加法器模块.对乘法器的设计自然集中在各模块的性能优化上,以达到速度快、面积小的目的.

2.1 部分积生成器

采用改进的 Booth 算法[2-3],其优点在于可以将部分积的个数减半,从而有效地节约了部分积压缩器的时间延迟,提高乘法器的整体性能.

在乘法 $Z = X \times Y$ 中,X,Y 为有符号和无符号整数进行硬件实现,综合两种情况,应产生 9 个部分积.

2.2 部分积压缩器

作为乘法器的核心部件,部分积压缩器的设计一直是研究的重点所在. Wallace 树结构[4]将全加器 CSA 组成树状的阵列,使多个加法并行执行,有效地减少了各级加法之间的等待延迟,成为当今大多数研究者的原始参考结构,通过改进设计,产生了各种各样的树状结构,并逐渐演化出 4:2 压缩器等

收稿日期:2007-05-21

概念.本设计也以 Wallace 树为基础,采用树状结构构建部分积压缩器,采用两种方法将全加器的数目尽量减少,以达到节约面积的目的.

(1) 减少符号位填充

在乘法器的压缩器中对 3 个数通过全加器进行相加时,3 个数的权值往往不同,在某些步,各加数的位数也可能不同.如图 1 所示.为使加法正常进行,就需要填充空位.空位有两种,一种是图 1 左上角的灰色部分,称为符号位填充.另一种是图 1 左下角的灰色部分,这里称为尾部 0 填充.减少尾部 0 填充的方法将在后面提到.

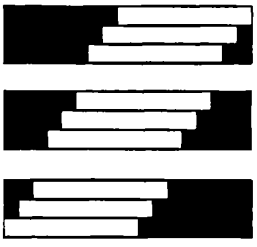

图 1　面积优化基础示意图

当乘法为无符号乘法时,符号位填充为 0,无需考虑;而注意到第 9 个部分积的最高位已经在 32 位,也无需考虑.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 $P0$ 为例,设符号填充位为 n.则填充后的数字可作如下等效:

$$P0 \to 2^{31}n + 2^{30}n + \cdots + 2^{17}n + P0 =$$
$$2^{32}n - 2^{17}n + P0 \to -2^{17}n + P0 =$$
$$\overline{nP0_{16}P0_{15}\cdots P0_0} \qquad (1)$$

由此,在全加器的三个加数中,首位权值最高的加数只需扩展一位既可实现加法的正确性.这样就使其在局部的加法中减少了符号位的填充个数.全加器树状阵列结构变为图 2 所示.

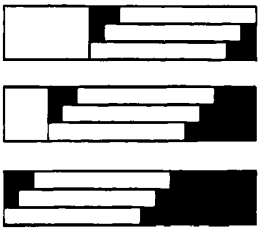

图 2　一次面积优化后示意图

在构造树形结构的时候,本设计尽量将权值相近的加数就近相加,这样也可以有效地减少相加时的错位,从而尽可能多地减少符号填充位.具体结构将在综合考虑两种面积优化方法后提出.

(2) 推迟加法减少尾部 0 填充

从前文已经看到,在加法阵列工作过程中,有些加数在尾部填充了 0,以完成加法运算.这种做法在以两个全加器组成的 4:2 压缩器中尤为常见,如文献[5]中提到的,如图 3 所示.

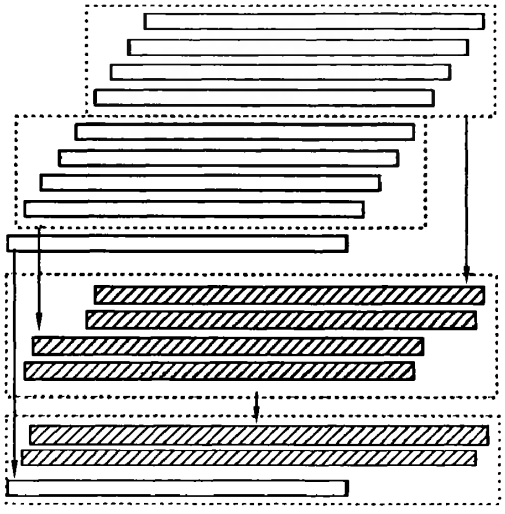

图 3　文献[5]中的压缩器结构框图

图中小方框表示 9 个部分积,虚线框分别表示 3 组 4:2 压缩器和 1 组全加器.斜线阴影框为中间结果.这种处理虽然从被填充的加数所在加法器的角度来看是必须的,但是从整体的角度来看则造成了面积的浪费.在本设计中将不足 3 个加数的加法情况尽量推迟到后续层次的加法器中进行计算,这样不但有效地利用了资源,更重要的是可以减小中间结果的位数,从根本上减小了后续层次加法器的输入位数,进而减少了加法器的个数需求.经这种方法优化后结构示意情况如图 4 所示,对比图 2,有明显改进[6].

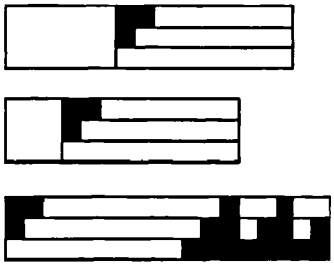

图 4　二次面积优化后示意图

如果采用文献[5]中的压缩器结构,即使应用了减少符号位填充的方法,也需要 188 个加法器.而本设计经减少尾部 0 填充,加法器的数目可以大幅减

少到 152 个.

除面积优化外,在时间延迟上,本设计也作了优化,尽量使各路径延迟较其他结构更为平衡,没有造成时间资源的浪费.本设计的 Wallace 树结构如图 5 所示.

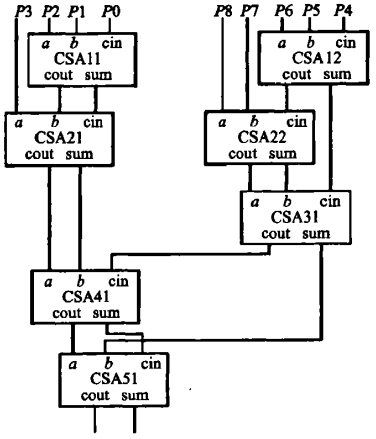

图 5 Wallace 树结构

由于在各输入均无延迟的情况下,全加器的 sum 输出端延迟约为两个 XOR 门延迟 $2d$,cout 输出端延迟约为一个 XOR 门延迟 d.因此关于全加器各端口延迟量,有如下关系:$t_{sum}=\max(t_a+2d, t_b+2d, t_{cin}+d)$, $t_{cout}=\max(t_a+d, t_b+d, t_{cin}+d)$. 通过分析可知本设计在时间延迟方面也有优势.

两设计综合比较如表 1 所示.

表 1 两设计综合比较

	文献[5]中设计	本设计
加法器数	188	152
总延迟 t_{S51}	$8d$	$6d$
总延迟 t_{C51}	$7d$	$6d$

回到面积问题上,综合上述两种节约面积的方法,可以得到最终的具体压缩器结构,其详细情况如表 2 所示.

在表 2 中,阴影部分的转移即为延迟进行的加法部分.Smn 和 Cmn 分别表示全加器组 CSAmn 的两组输出,斜线阴影部分为尾部 0 填充.另外,如前文所提到的,在 Booth 编码过程中,为得到部分积,存在将被乘数 X 与负数相乘的情况,即一些部分积将在变号操作后得到,这就需要"取反加 1",而如果单独用一个加法器来完成这个动作,将比较浪费资源.在本设计中,采取将所有的 8 个"加 1"作为压缩器中某些加数的一部分,穿插进来,表中 cn 即表示这些加入进来的部分.

表 2 经二次优化后的部分积压缩器

		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	P0											17	17	17	17	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	P1											17	17	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
	P2											17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
CSA 21	S11									17	17	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
	C11									17	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					
	P3									17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
CSA 12	P4			17	17	17	17	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
	P5			17	17	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								
	P6			17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0										
CSA 22	S12	17	17	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0										
	C12	17	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0											
	P7	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0												
CSA 31	S22	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0												
	C22	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0													
	P8	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	c7		c6		c5		c4							
CSA 41	S21	17	17	17	17	17	17	17	17	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
	C21	17	17	17	17	17	17	17	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					
	C31	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0							
CSA 51	S31	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
	S41	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
	C41	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			c3		c2		c1		c0
S51		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
C51		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	

3　超前进位加法器

部分积经压缩器压缩后,得到的是 32 位和 31 位的两个数字,需要继续相加才能得到最终结果.这里采用超前进位加法器,它的优势在于可以先行求得多位加法各位间的进位值,从而节约了等待进位所需要的时间延迟.通过进位产生信号 g 和进位传播信号 p 进行计算进位 c. p, q, c 的公式见式(2).

$$
\begin{aligned}
&g[i] = a[i]b[i], p[i] = a[i] + b[i]\\
&\text{cout}[0] = g[0] + p[0]\text{cin}\\
&\text{cout}[1] = g[1] + p[1]g[0] + p[1]p[0]\text{cin}\\
&\text{cout}[3] = g[2] + p[2]g[1] + p[2]p[1]g[0] +\\
&\qquad p[2]p[1]p[0]\text{cin}\\
&\text{cout}[3] = g[3] + p[3]g[2] + p[3]p[2]g[1] +\\
&\qquad p[3]p[2]p[1]g[0] +\\
&\qquad p[3]p[2]p[1]p[0]\text{cin}\\
&\cdots\cdots
\end{aligned}
\tag{2}
$$

显然,位数越大,所需门数越多,如一概照搬计算,将损失大量芯片面积.故本设计采取将 32 位加法分段处理的方法,每 4 位一组,分别计算进位,再将每一组看作一个单位,得到各组的总进位产生信号 G 和总进位传播信号 P,计算方法见式(3),进而计算各单位之间的总进位 $C[i]$,构成超前进位加法器层次阵列.在不浪费面积的前提下,有效地达到超前进位的目的.

$$
\begin{aligned}
G[i] &= g[4i+3] + p[4i+3]g[4i+2] +\\
&\qquad p[4i+3]p[4i+2]g[4i+1] +\\
&\qquad p[4i+3]p[4i+1]g[4i]p[i]\\
&= p[4i+3]p[4i+2]p[4i+1]p[4i]
\end{aligned}
\tag{3}
$$

4　结果分析

本设计应用改进 Booth 算法,在部分积压缩器的设计上采用两种方法,有效地减少了全加器的数目,节约了芯片面积,同时也考虑了时间延迟的优化.共使用全加器 148 个,半加器 4 个.对本设计采用 SMIC 0.18μm 工艺库,通过 synopsys 公司的工具 Design Compiler 进行综合,结果如表 3 所示.

表 3　综合结果

工艺库/μm	条件	延迟/ns	面积/μm²
SMIC 0.18	typical	4.62	23 837

参考文献:

[1] David A Patterson, John L Hennessy. 计算机系统结构——量化研究方法[M]. 3 版. 北京:电子工业出版社, 2004:99-105.

[2] Booth A D. A signed binary multiplication technique[J]. Journal of Mechanics and Applied Mathematics, 1951, 4(2):236-240.

[3] David A Patterson, John L Hennessy. 计算机组成和设计(硬件/软件接口)[M]. 2 版. 北京:清华大学出版社, 2003:196-199.

[4] Wallace C S. A suggestion for a fast multiplier[J]. IEEE Transactions on Electronic Computers, 1964, 13(2):14-17.

[5] 李小进,初建朋,赖宗声,等. 定点符号高速乘法器的设计与 FPGA 实现[J]. 微电子学与计算机,2005,22(4):119-125.

[6] 刘鸿瑾,张铁军,侯朝焕.基于快速舍入的双精度浮点乘法器的设计[J].微电子学与计算机,2006,23(6):162-165.

作者简介:

李　楠　男,(1983-),硕士研究生. 研究方向为大规模集成电路设计.

(上接第 155 页)

[6] 吕坤,高珊. 一种基于改进的 BP 神经网络算法的布匹瑕疵分类器[J].微电子学与计算机,2006, 23(3):88-90.

[7] 崔金魁,杨扬,颉斌. 一种基于集成 BP 网络的手写汉字识别方法[J]. 微电子学与计算机,2006, 23(8):121-124.

[8] 胡志军,王建国,王鸿斌. 基于优化 BP 神经网络的 PID 控制研究与仿真[J].微电子学与计算机, 2006,23(12):138-140.

作者简介:

张　蕾　女,(1980-),硕士研究生. 研究方向为模式识别、图像处理.

普杰信　男,(1959-),教授,硕士生导师.研究方向为人工智能与模式识别、图像处理.

范庆辉　男,(1978-),助教.研究方向为嵌入式系统、图像处理.