

一种改进的模 2^n-1 乘法器

敖思远, 李 磊

(电子科技大学 电子科学技术研究院, 四川 成都 611731)

摘 要: 在余数系统中, 2^n-1 是一种最普遍应用的模, 文中提出一种改进方法来设计修改模 2^n-1 Booth 乘法器. 当 n 为偶数时, 将比以往方法减少很多部分项. 实验结果表明, 新改进的乘法器与以前相比在面积上节省约 9%, 在时间延时上节省约 2.4%.

关键词: 余数系统; Booth 乘法器; 华莱树

中图分类号: TN742.2

文献标识码: A

文章编号: 1000-7180(2011)04-0107-04

A New Proposed Modulo 2^n-1 Multiplier

AO Si-yuan, LI Lei

(Research Institute of Electronic Science and Technology of UESTC, Chengdu 610054, China)

Abstract: 2^n-1 is one of the most commonly used moduli in Residue Number Systems. In this paper, we propose a new method for designing modified Booth modulo 2^n-1 multipliers, which, for even values of n , require one less partial product than the already known. the proposed multipliers compared to earlier solutions offer savings up to 9 percent and up to 2.4 percent in the implementation area and execution delay, respectively.

Key words: index terms residue number systems; Booth multiplier; Wallace tree

1 引言

在余数系统中, 每个操作数对于一组整数都编码为一组余数向量, 这组整数是一组基^[1]. 例如, A 是一个整数, $\{m_1, m_2, \dots, m_n\}$ 是一组基, A 在余数系统中可以表示为:

$$A = \{A_1, A_2, \dots, A_n\} \quad (1)$$

其中, $A_i = \langle A \rangle_{m_i}, i = 1, 2, \dots, M. \langle A \rangle_{m_i}$ 表示的是把 A 对 m_i 取模. 两个操作数在余数系统中表示为 $C = A \circ B = \{C_1, C_2, \dots, C_M\}$, 其中 $C_i = \langle A_i \circ B_i \rangle_{m_i}, i = 1, 2, \dots, M.$ $\circ$ 表示乘法、加法、减法等操作^[2]. 余数系统能够将一个整数转换为一组可以并行执行的更小的整数, 而且它是免进位的数值系统, 可以不通过 $(A_j, B_j), i \neq j$ 进行计算.

最有效的模 2^n-1 乘法器是基于查找表提出的^[3]. 但是随着模的增大, ROM 的大小也成指数增长, 因此对于不是很大的模来说, 仅仅建立于组合硬

件的方法很合适. 在文献[4]中, 提出了基于 Wallace trees 的模 2^n-1 乘法器, 但其中的部分项与操作数长度是成比例关系. 为了解决这个问题, 文献[5]提出了修改的模 2^n-1 Booth 乘法器. 文中提出一种新型改进的模 Booth 乘法器, 此结构与文献[5]相比, 在速度和面积上都有较大幅度改善.

2 模 2^n-1 Booth 乘法器原理

对 $A \times B$ 采用修改的 Booth 算法, 即 2 位重新编码^[6]. 此算法中, 乘数 B 每 3 位编码为一组, 编码为 $\{-2, -1, 0, +1, +2\}$ 中的一个元素. 表 1 和图 1 分别为编码的真值表 and 对应实现电路. 通过此种编码方式, 乘数 A 可相应表示为部分项, 部分项的各位由 booth 选择器得到. 真值表如表 2 所示. 对应电路如图 2 所示. 然后通过几级加法简化为 2 个数, 最后通过模加法器得到结果. 其中模加法器采用[7]所提出的结构, 能减小传输延时、优化加法器的扇出、减少加法的级数.

表 1 Booth 编码真值表

x_{2i+1}	x_{2i}	x_{2i-1}	s	x_2	x_1
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	1	1	0
1	0	0	1	0	1
1	0	1	1	0	1
1	1	0	1	0	1
1	1	1	1	0	0

表 2 Booth 选择器的真值表

s	x_2	x_1	d_i
0	0	0	0
0	0	1	a_i
0	1	0	a_{i-1}
1	0	0	1
1	0	1	$\bar{a}_i$
1	1	0	$\bar{a}_{i-1}$

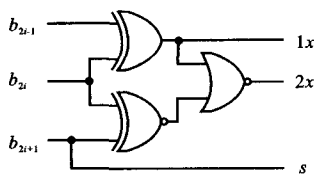


图 1 Booth 编码器

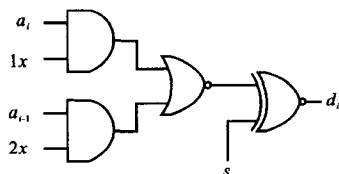


图 2 Booth 选择器

3 改进的模 $2^n - 1$ Booth 乘法器

假设 $A = a_{n-1}a_{n-2}\cdots a_0$, $B = b_{n-1}b_{n-2}\cdots b_0$ 是两个以 $2^n - 1$ 为模表示出的 n 位的数, A 是被乘数, B 是乘数。

3.1 乘数 B 的 Booth 编码方法

为了减少乘法结果的数目, 把 B 写为:

$$B = \sum_{i=0}^{n-1} b_i 2^i$$

$$= (b_0 - 2b_1) + \sum_{i=1}^{\frac{n}{2}} (b_{2i-1} + b_{2i} - 2b_{2i+1}) 2^{2i}$$

对 n 分以下两种情况:

(1) 当 n 为偶数

$B = b_{n-1}2^n + (b_0 - 2b_1) + \sum_{i=1}^{\frac{n}{2}-1} (b_{2i-1} + b_{2i} - 2b_{2i+1}) 2^{2i}$, 把 B 表示为 $|B|_{2^n-1}$, 得到:

$$|(b_{n-1} + b_0 - 2b_1) + \sum_{i=1}^{\frac{n}{2}-1} (b_{2i-1} + b_{2i} - 2b_{2i+1}) 2^{2i}|_{2^n-1}$$

由于 $b_{-1} = b_{n-1}$, 得到:

$$B = \left| \sum_{i=0}^{\lfloor \frac{n+1}{2} \rfloor - 1} (b_{2i-1} + b_{2i} - 2b_{2i+1}) 2^{2i} \right|_{2^n-1} \quad (2)$$

对于上式中的 b_1 , b_0 和 b_{n-1} 可以直接进行 Booth 编码。

然而在文献[5]中, 对于同样的输入用了两个不同的编码, 即 b_1 , b_0 和 0 与 0, 0 和 b_{n-1} , 因而此结构就少了一项。对应硬件电路就少了一个 Booth 编码器, n 个 Booth 选择器和 n 个全加器以及它们之间的连线。

(2) 当 n 为奇数

由于 $b_{-1} = b_n = 0$, 得到:

$$B = \sum_{i=0}^{\lfloor \frac{n+1}{2} \rfloor - 1} (b_{2i-1} + b_{2i} - 2b_{2i+1}) 2^{2i} \quad (3)$$

其编码方式与文献[5]相同。

至此, 将式(2)和式(3)进行合并, 得到 n 为任意值的情况下, B 的完整表达式:

$$B = \left| \sum_{i=0}^{\lfloor \frac{n+1}{2} \rfloor - 1} (b_{2i-1} + b_{2i} - 2b_{2i+1}) 2^{2i} \right|_{2^n-1} \quad (4)$$

3.2 $A \times B$ 对 $2^n - 1$ 取模分析

$A \times B$ 对 $2^n - 1$ 取模可表示为:

$$|AB|_{2^n-1} = \left| \sum_{i=0}^{\lfloor \frac{n+1}{2} \rfloor - 1} (b_{2i-1} + b_{2i} - 2b_{2i+1}) 2^{2i} \right|_{2^n-1}$$

$$= \left| \sum_{i=0}^{\lfloor \frac{n+1}{2} \rfloor - 1} PP_i \right|_{2^n-1} \quad (5)$$

其中 $PP_i = |A(b_{2i-1} + b_{2i} - 2b_{2i+1}) 2^{2i}|_{2^n-1}$ 。

由于 $b_{e,i} = (b_{2i-1} + b_{2i} - 2b_{2i+1})$ 可由基 $\{-2, -1, 0, 1, 2\}$ 表示, 用 $b_{e,i} 2^{2i} = s 2^j$ 将式(5)中的部分结果代替, 得到 $PP_i = |s A 2^j|_{2^n-1}$ 。

(1) 当 $s = +1$, 得到:

$$|s A 2^j|_{2^n-1} = |a_{n-1-j} a_{n-2-j} \cdots a_0 a_{n-1} \cdots a_{n-j}|_{2^n-1}$$

$$= a_{n-1-j} a_{n-2-j} \cdots a_0 a_{n-1} \cdots a_{n-j}$$

(2) 当 $s = -1$, 得到:

$$|s A 2^j|_{2^n-1}$$

$$\begin{aligned}
 &= |(2^n - 1) - a_{n-1-j}a_{n-2-j} \cdots a_0a_{n-1} \cdots a_{n-j}|_{2^n-1} \\
 &= |\overline{a_{n-1-j}a_{n-2-j} \cdots a_0a_{n-1} \cdots a_{n-j}}|_{2^n-1} \\
 &= \overline{a_{n-1-j}a_{n-2-j} \cdots a_0a_{n-1} \cdots a_{n-j}}
 \end{aligned}$$

(3) 当 $s = 0$, 得到:

$$|sA2^j|_{2^n-1} = 0 = \underbrace{000 \cdots 00}_n.$$

通过以上三种情况分析,可得图3所示模 $2^{16}-1$ 乘法器整体架构.如前面所述,对于相同的输入,文中是用一个Booth乘法器来编码输入 $\{b_1, b_0, b_{15}\}$.

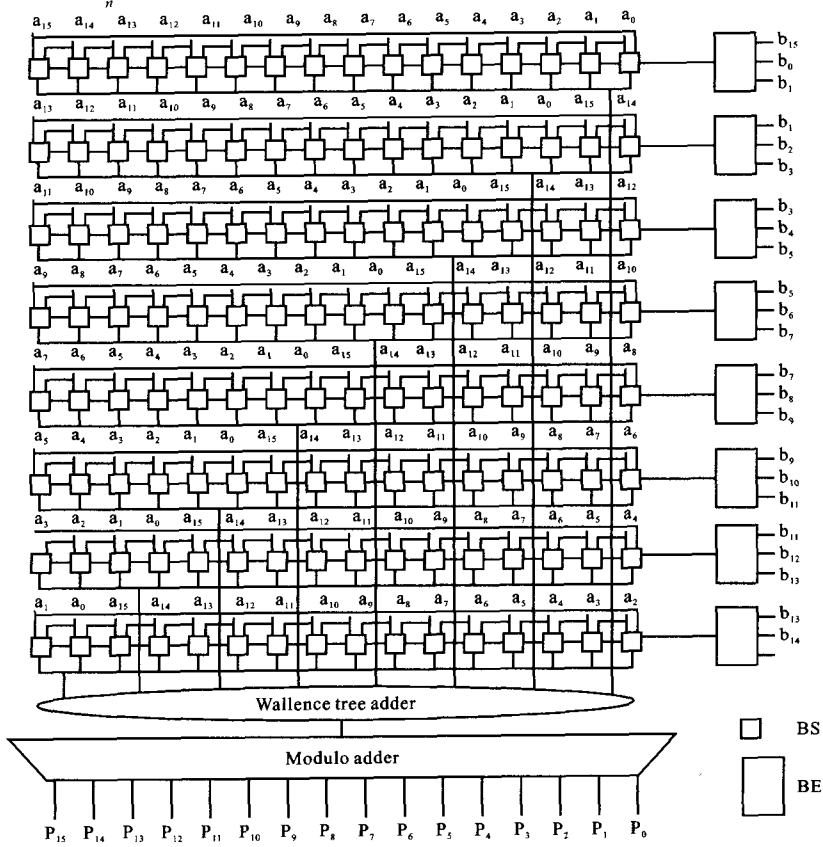


图3 提出的模 $2^{16}-1$ Booth乘法器

4 结果比较

将文中提出的乘法器与文献[4]、[5]中的乘法器相比较,以此来证明本结构能够有效减少面积和延时.

文献[4]中提出的乘法器:

面积为:

$$A[3] = n^2 + n(n-2)A_{FA} + A_{PA_n}$$

延迟为:

$$T[3]_{WAL} = 1 + k(n)T_{FA} + T_{PA_n}$$

文献[5]中提出的乘法器:

面积为:

$$\begin{aligned}
 A(4) &= \left(\left\lfloor \frac{n}{2} \right\rfloor + 1\right)A_{BE} + n\left(\left\lfloor \frac{n}{2} \right\rfloor + 1\right)A_{BS} + \\
 &n\left(\left\lfloor \frac{n}{2} \right\rfloor - 1\right)A_{FA} + A_{PA_n}
 \end{aligned}$$

延迟为:

$$T[4]_{WAL} = T_{BE} + T_{BS} + k\left(\left\lfloor \frac{n}{2} \right\rfloor + 1\right)T_{FA} + T_{PA_n}$$

文中提出的乘法器:

面积为:

$$\begin{aligned}
 A_{Proposed} &= \left\lfloor \frac{n}{2} \right\rfloor A_{BE} + n\left\lfloor \frac{n}{2} \right\rfloor A_{BS} + \\
 &n\left(\left\lfloor \frac{n}{2} \right\rfloor - 1\right)A_{FA} + A_{PA_n}
 \end{aligned}$$

延迟为:

$$\begin{aligned}
 T_{Proposed, WAL} &= T_{BE} + T_{BS} + k\left(\left\lfloor \frac{n}{2} \right\rfloor\right)T_{FA} + \\
 &T_{PA_n}
 \end{aligned}$$

其中, A_{FA} 表示全加器的面积, T_{FA} 表示延时. A_{BE} 表示Booth编码器的面积, T_{BE} 表示它的延时. A_{BS} 表示Booth选择器的面积, T_{BS} 表示它的延时. A_{PA_n} 表示模 2^n-1 加法器($PA(n)$)的面积, T_{PA_n} 表示它的延时. $k(x)$ 表示Wallace tree FAs的深度.

文中设计与文献[4] 和文献[5] 比较,对于 $n = 4, 8, 16, 32$ 时,在时间和面积上的比较如表 3 所示。由于忽略了布线和扇入、扇出,上述比较也只是近似结果。时间单位是 ns,面积单位是 μm^2 。

表 3 理论计算面积和时间比较

n	文献[4]		文献[5]		文中		比文献[4]节省%		比文献[5]节省%	
	面积	延时	面积	延时	面积	延时	面积	延时	面积	延时
4	112	16	143	18	90	14	19.6	14	37.1	22.2
8	504	26	497	28	396	24	21.4	24	20.3	14.3
16	2080	36	1805	34	1608	34	22.7	34	10.9	0.0
32	8352	46	6773	44	6384	44	23.6	44	5.7	0.0

为了得到更精确的结果,让电路在静态 CMOS VLSI 下工作,首先用硬件描述语言实现各种乘法器,然后映射到 smic13 技术库中,用 Synopsys Design Compiler 进行综合,在综合过程中优化时序,然后尽可能的减少面积。所有结果都是在最坏参数下提出的,比较结果如表 4 所示。

表 4 实测面积与延时比较

n	文献[4]		文献[5]		文中		比文献[4]节省%		比文献[5]节省%	
	面积	延时	面积	延时	面积	延时	面积	延时	面积	延时
4	1324	3.09	1492	3.12	1246	2.73	5.9	11.7	16.4	12.5
8	4444	4.41	4303	4.50	3677	4.24	17.2	3.9	14.5	5.8
16	15769	5.76	14032	5.51	12771	5.38	19.0	6.6	9.0	2.4
32	58747	7.18	48337	6.67	45358	6.5	22.8	9.5	6.2	2.5

5 结束语

文中提出了一种新的模 $2^n - 1$ Booth 乘法器设计方法,并将其与已有的模乘法器在时间和速度上进行比较。当 n 为偶数时,将比以往的方法减少很多部分项;实验结果表明,新改进的乘法器与以前相比在面积上节省约 9%,在时间延时上节省约 2.4%。

参考文献:

[1] 马上,胡剑浩. 余数系统在 VLSI 设计中的基本问题研究与进展[C]//中国通信集成电路技术与应用研讨会文集. 成都:中国通信学会,2006.

[2] 叶春,张曦煌. 基于 FPGA 乘法器架构的 RNS 与有符号二进制量转换[J]. 微电子学与计算机,2005,22(11): 148—150.

[3] Skavantzios A, Rao P B. New multipliers modulo $2^n - 1$ [J]. IEEE Transactions on Computers, 1992,41(8):957—961.

[4] Wang Z, Jullien G A, Miller W C. An algorithm for multiplication modulo $(2^n - 1)$ [C]// Proceedings of 39th Midwest Symposium on Circuits and Systems. Ames, IA, USA:IEEE, 1996(8):1301—1304.

[5] Zimmerman R. Efficient VLSI implementation of modulo $(2^n \pm 1)$ addition and multiplication[C]// Proceeding of 14th IEEE Symposium on Computer Arithmetic. Adelaide, Australia: IEEE, 1999:158—167.

[6] 李军强,李东生,李奕磊,等. 32X32 高速乘法器的设计与实现[J]. 微电子学与计算机,2009,26(12):23—26.

[7] Kalampoukas L, Nikolos D, Efstathiou C, et al. High-speed parallel-prefix modulo $2^n - 1$ adders[J]. IEEE Transactions on Computers, 2000,49(7):673—680.

作者简介:

敖思远 女,(1986—),硕士. 研究方向为数字电路设计方面的研究。