

## 高速 Booth 编码模 $(2^n - 1)$ 乘法器的设计

李 磊, 胡剑浩, 敖思远

(电子科技大学 通信抗干扰国家重点实验室, 四川 成都 611731)

**摘 要:** 在余数系统中  $(2^n - 1)$  是最普遍应用的模, 提出了一种新的 booth 编码结构, 并基于提出的 booth 编码结构, 提出了一种高速模  $(2^n - 1)$  乘法器. 该乘法器采用 CSA 或者 Wallace Tree 结构可以进一步提高运算速度. 此乘法器在一个时钟周期内可以完成所需运算, 简单高效.

**关键词:** 乘法器; Booth 编码算法; Wallace 树形结构

**中图分类号:** TN4

**文献标识码:** A

**文章编号:** 1000-7180(2011)11-0191-03

## Novel Modified Booth Modulo $(2^n - 1)$ Multipliers

LI Lei, HU Jian-hao, AO Si-yuan

(National Key Laboratory of Communication, University of Electronic Science  
and Technology of China, Chengdu 610054, China)

**Abstract:**  $(2^n - 1)$  is one of the most commonly used moduli in Residue Number Systems. In the paper we propose a novel Booth encoding architecture. Based on the proposed booth encoding architecture, we can design high speed and high-efficient modulo  $(2^n - 1)$  multipliers, which are the fastest among all known modulo  $(2^n - 1)$  multipliers.

**Key words:** multiplier; Booth coder; optimal Wallace tree

### 1 引言

余数系统 RNS<sup>[1]</sup> 是一种通过一组两两互质余数基的余数来描述数字的数值表征系统. 由  $\{m_1, m_2, \dots, m_L\}$  组成的  $L$  个余数基. 整数  $X, 0 \leq X < M$ , 其中  $M = m_1 \times m_2 \times \dots \times m_L$ , 在 RNS 中有唯一的表示方式为  $X = \{x_1, x_2, \dots, x_L\}, x_i = \langle X \rangle_{m_i}$  其中  $\langle X \rangle_{m_i}$  表示  $X$  对于模  $m_i$  的余数. 在余数系统中两个操作数进行操作, 操作符为  $\Theta$ , 可以定义为

$$\{z_1, z_2, \dots, z_L\} = \{x_1, x_2, \dots, x_L\} \Theta \{y_1, y_2, \dots, y_L\}$$

式中,  $z_i = \langle x_i \Theta y_i \rangle_{m_i}$ ,  $\Theta$  中可以是模加法, 减法或乘法. 在余数系统中这些算术运算都是并行执行的, 且处理的都是很小的余数, 而不是一个很大的数.

对于余数基的选择,  $\{2^n, 2^n - 1, 2^n + 1\}$  和  $\{2^n, 2^n - 1, 2^{n-1} - 1\}$  这两组余数基得到了广泛的应用, 因为当考虑  $\text{area} \times \text{time}^2$  时, 它们提供了最有效的电路, 并且在余数系统与二进制的互转过程中也是最

有效的<sup>[2-4]</sup>. 由此可见, 对于模  $(2^n - 1)$  乘法器的研究是非常有意义的.

模  $(2^n - 1)$  乘法器已经广泛应用于 Fermat 数值转换和余数系统以及数字信号处理中. 近年来一些模  $(2^n - 1)$  乘法器已经被提出. 最有效的模  $(2^n - 1)$  乘法器是基于查找表提出的<sup>[5]</sup>. 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[6]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[7-8]中提出了一种模  $(2^n - 1)$  Booth 乘法器.

文中提出了一种新的 booth 编码结构, 并基于提出的 booth 编码结构, 提出了一种高速模  $(2^n - 1)$  乘法器. 该乘法器采用 CSA 或者 Wallace Tree 结构可以进一步提高运算速度. 此乘法器在一个时钟周期内可以完成所需运算.

## 2 模 $(2^n - 1)$ 乘法器的结构

在本部分介绍新提出的模 $(2^n - 1)$  booth 乘法器. 假设  $A = a_{n-1} \cdots a_i \cdots a_0$ ,  $B = b_{n-1} \cdots b_i \cdots b_0$  是两个基于模 $(2^n - 1)$  表示出的数,  $A$  是被乘数,  $B$  是乘数.  $B$  可以表示为

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

$$= \begin{cases} b_n 2^n + \sum_{i=0}^{n-1} b_i 2^i & | b_n = 0, n \text{ is even} \\ b_{n+1} 2^{n+1} + b_n 2^n + \sum_{i=0}^{n-1} b_i 2^i & | b_{n+1} = 0, b_n = 0, \\ & n \text{ is odd} \end{cases} \quad (1)$$

使  $E_k = b_{k-1} + b_k - 2b_{k+1}$ , 可以得到表 1 的部分积, 其中  $PP_k = pp_{k,n-1} \cdots pp_{k,i} \cdots pp_{k,0} = \langle E_k A \rangle_{2^n-1}$ .

表 1 部分积生成

| $b_{2k+1}$ | $b_{2k}$ | $b_{2k-1}$ | $E_k$ | $PP_k$                                | $pp_{k,i}$           |
|------------|----------|------------|-------|---------------------------------------|----------------------|
| 0          | 0        | 0          | 0     | 0                                     | 0                    |
| 0          | 0        | 1          | +1    | $\langle 2^{2k} A \rangle_{2^n-1}$    | $a_i$                |
| 0          | 1        | 0          | +1    | $\langle 2^{2k} A \rangle_{2^n-1}$    | $a_i$                |
| 0          | 1        | 1          | +2    | $\langle 2^{2k+1} A \rangle_{2^n-1}$  | $a_{i-1}$            |
| 1          | 0        | 0          | -2    | $\langle -2^{2k+1} A \rangle_{2^n-1}$ | $\overline{a_{i-1}}$ |
| 1          | 0        | 1          | -1    | $\langle -2^{2k} A \rangle_{2^n-1}$   | $\overline{a_i}$     |
| 1          | 1        | 0          | -1    | $\langle -2^{2k} A \rangle_{2^n-1}$   | $\overline{a_i}$     |
| 1          | 1        | 1          | 0     | +0                                    | 1                    |

通过逻辑加和逻辑简化, 表 1 中的第 6 列含  $a_i$  和  $\overline{a_i}$  ( $\overline{x}$  表示  $x$  的补码) 的项可以表示为

$$\begin{aligned} \psi_{k,i} &= (\overline{b_{2k+1}} \overline{b_{2k}} b_{2k-1} + \overline{b_{2k+1}} b_{2k} \overline{b_{2k-1}}) a_i + (b_{2k+1} \overline{b_{2k}} b_{2k-1} + b_{2k+1} b_{2k} \overline{b_{2k-1}}) \overline{a_i} \\ &= (b_{2k} \oplus b_{2k-1}) (b_{2k+1} \oplus a_i) \end{aligned} \quad (2)$$

表 1 中的第 6 列中含  $a_{i-1}$  或  $\overline{a_{i-1}}$  的项可以表示为

$$\begin{aligned} \xi_{k,i-1} &= \overline{b_{2k+1}} b_{2k} b_{2k-1} a_{i-1} + b_{2k+1} \overline{b_{2k}} \overline{b_{2k-1}} \overline{a_{i-1}} \\ &= (b_{2k} \odot b_{2k-1}) (b_{2k+1} \overline{b_{2k}} a_{i-1} + \overline{b_{2k+1}} b_{2k} \overline{a_{i-1}}) \end{aligned} \quad (3)$$

项  $b_{2k+1} b_{2k} b_{2k-1}$  需要加到  $\psi_{k,i}$  或者  $\xi_{k,i-1}$  来产生  $pp_{k,i}$ . 但是由于  $\psi_{k,i}$  含有  $b_{2k} \oplus b_{2k-1}$  项,  $b_{2k+1} b_{2k} b_{2k-1}$  和  $\psi_{k,i}$  很难进行合并, 但是它却很容易和  $\xi_{k,i-1}$  进行合并, 是由于其中含有  $b_{2k} \odot b_{2k-1}$  这项, 因此, 得到

$$\zeta_{k,i-1} = \xi_{k,i-1} + b_{2k+1} b_{2k} b_{2k-1}$$

$$= (b_{2k} \odot b_{2k-1}) (b_{2k+1} \overline{a_{i-1}} + b_{2k} a_{i-1}) \quad (4)$$

令  $h_k = b_{2k} \oplus b_{2k-1}$ ,  $l_{k,i} = b_{2k+1} \oplus a_i$  和  $r_{k,i-1} = b_{2k+1} \overline{a_{i-1}} + b_{2k} a_{i-1}$ , 则有

$$pp_{k,i} = \psi_{k,i} + \zeta_{k,i-1} = h_k \cdot l_{k,i} + \overline{h_k} \cdot r_{k,i-1} \quad (5)$$

考虑到  $h_k = b_{2k} \oplus b_{2k-1}$  是 booth 编码逻辑方程, 图 1 给出了 booth 编码器电路. 显然,  $pp_{k,i}$  可以用一个异或门 XNOR, 两个二选一反相多路复用器来实现. 图 2 画出了二选一反相多路复用器以及逻辑表达式和整个的 booth 选择器.

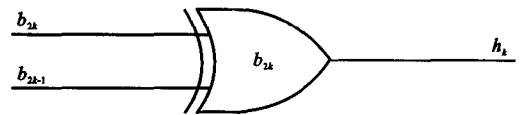


图 1 Booth encoder

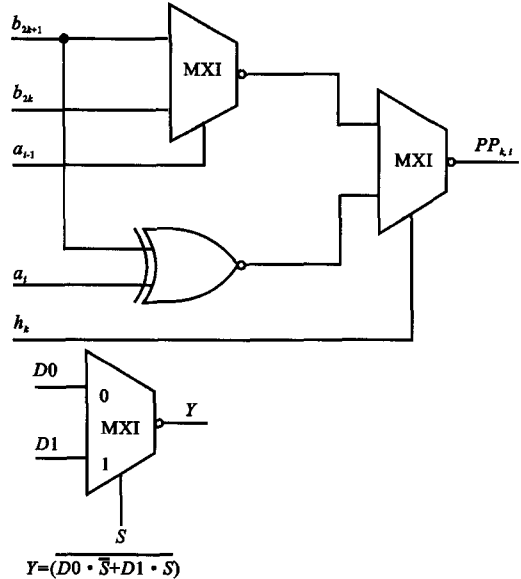


图 2 Booth selector

产生于 booth encoder 的译码信息和操作数  $A$ , 一起用来产生部分积. 部分积的每个字节通过 booth selector 生成. 产生的部分积通过几级相加最后简化为 2 个. 每一级最高位的进位输出权为, 它通过模运算后, 变为 1. 因此这些最高位的进位通过循环操作的形式被加到下一级操作数的最低位. 压缩部分积的方法有多种, 其中最常用的是 CSA 序列和 WallaceTree 结构. 一个全加器可以被看作是一个 (3:2) 的压缩器. 压缩产生的两个操作数通过一个模加法器得到最后的结果. 图 3 中给出了整个模 255 乘法器的结构.

## 3 实验数据

此设计基于 SMIC013CMOS1P8M 工艺库, 在 Synopsys DesignCompiler 下综合的结果如表 2 所

示. 显然该乘法器可以满足高速设计的需求.

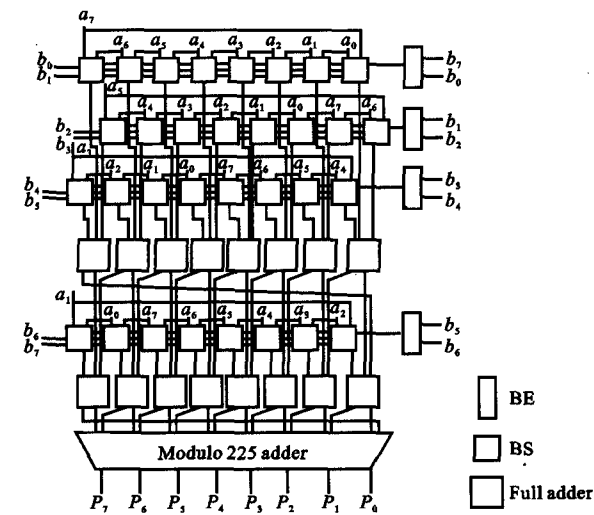


图 3 模 225 乘法器

表 2 综合结果

| $n$ | Area/ $\text{mil}^2$ | Delay/ns |
|-----|----------------------|----------|
| 4   | 1 242.50             | 2.50     |
| 8   | 3 883.65             | 3.80     |
| 16  | 13 538.46            | 5.06     |
| 32  | 48 684.82            | 6.26     |

4 结束语

本乘法器所采用的 Booth 编码算法、Wallace Tree 最优化的演算法是设计高性能通常采用的方法,此乘法器在一个时钟周期内可以完成模乘法运算. 综合结果显示可以满足高速运算的需要. 整个设计简单高效.

参考文献:

[1] Soderstrand M A, Taylor F J, Jenkins w k, et al. Residue number system arithmetic: modern applications in

digital Signal Processing[M]. piscatawing, USA: IEEE Press, 1986.

[2] Paliouras V, Stouraitis T. Novel high-radix residue number system multipliers and adders[C]// Proc IEEE Int'l Symp Circuits and Systems. orlando, USA: IEEE, 1999:451-454.

[3] Wang W, Swamy M, Ahmad M, et al. A high-speed residue-to-binary converter for three-moduli  $(2k, 2k-1, 2k-1-1)$  RNS and a scheme for Its VLSI implementation[J]. IEEE Trans Circuits and Systems II, 2000, 47(12):1576-1581.

[4] Wang Y, Song X, A bouhmid M, et al. Adder based residue to binary number converters for  $(2n-1, 2n, 2n+1)$ [J]. IEEE Trans Signal Processing, 2002, 50(7): 1772-1779.

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

[6] Wang Z, Jullien G A, Miller W C. An algorithm for multiplication modulo  $(2^n-1)$ [C]// Proc 39th Midwest Symp on Circuits and Systems. Ames, USA: IEEE, 1997:1301-1304.

[7] Zimmerman R. Efficient VLSI implementation of modulo  $(2^n \pm 1)$  addition and multiplication[C]// Proc 14th IEEE Symp on Computer Arithmetic. Adelaide, SA, Aostralia: IEEE, 1999:158-167.

[8] Efstathiou C, Vergos H T, Nikolos D. Modified booth modulo  $2n-1$  multipliers[J]. IEEE Trans Computers, 2004, 53(3):370-374.

作者简介:

李磊 男, (1982-), 博士. 研究方向为专用集成电路 (ASIC) 和 SoC 设计.

胡剑浩 男, (1971-), 教授, 博士生导师. 研究方向为数字 SoC 和 ASIC 设计.

敖思远 女, (1986-), 硕士. 研究方向为数字 SoC 和 ASIC 设计.