

一种高效模 $(2^n - 2^p)$ 乘法器的设计

周 璐, 李 磊

(电子科技大学 电子科学技术研究院, 四川 成都 611731)

摘 要: 结合两类修正方法, 提出了一种高效的模 $(2^n - 2^p)$ 乘法器($n \geq 2p$)的实现方法. 与文献[1]中设计比较, 本文乘法器结构上少了一级加法, 并且综合结果也显示平均面积和延迟分别有 10% 和 13% 的减小.

关键词: 余数系统; 模 $(2^n - 2^p)$ 乘法器; 修正

中图分类号: TN403

文献标识码: A

文章编号: 1000-7180(2013)09-0141-03

An Effective Architecture for Designing Modulo $(2^n - 2^p)$ Multipliers

ZHOU Lu, LI Lei

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

Abstract: In this paper, a new architecture is proposed to design modulo $(2^n - 2^p)$ multipliers on the condition $n \geq 2p$. When compared with the reference multipliers [1], the proposed multiplier reduces both time delay and hardware requirements significantly. The proposed design is very suitable for VLSI implementation. It can be completed in one cycle.

Key words: residue number systems (RNS); modulo; multiplier

1 引言

余数系统 $RNS^{[1-2]}$ 是通过一组两两互质余数基产生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, 对于整数 X , $0 \leq X < M$, 可以唯一表示为 $X = \langle x_1, x_2, \dots, x_L \rangle$, 其中 $M = m_1 \times m_2 \times \dots \times m_L$, $x_i = \langle X \rangle_{m_i}$, $\langle X \rangle_{m_i}$ 表示 X 对于模 m_i 的余数.

余数系统和二进制的相互转化耗费资源过大, 造成整体系统的低效耗时. 在大多数情况下, 模 $(2^n - 2^p)$ 的运算复杂度决定余数系统和二进制转化算法的性能, 所以对模 $(2^n - 2^p)$ 乘法器的研究是非常有意义的. 近几年来, 对模 $(2^n - 2^p \pm 1)$ 乘法器的研究已很成熟^[3-5], 但忽略了对模 $(2^n - 2^p)$ 乘法器的研究. 现有的办法一般是采用文献[1]中的结构来设计模 $(2^n - 2^p)$ 乘法器. 该方法硬件资源开销比较大, 需要两个乘法器、两个加法器、一个 CSA (Carry

Save Adder) 压缩器阵列和一个组合逻辑电路; 延迟比较大. 本文提出的模 $(2^n - 2^p)$ 乘法器需要一个乘法器、一个改进的模 $(2^{n-p} - 1)$ 加法器和一个 CSA 压缩器阵列, 这种结构减少了一级加法器.

2 修正方法

为了解决设计中出现的进位溢出问题, 介绍以下两类修正方式.

对于任意的整数 X , 都有:

$$\langle X \times 2^n \rangle_{2^n - 2^p} = \langle X \times 2^p \rangle_{2^n - 2^p} \quad (1)$$

2.1 第一类修正方法

当 Z 为 $(n+1)$ 位的整数时, 则有:

$$\begin{aligned} \langle Z[n:0] \rangle_{2^n - 2^p} &= \langle Z[n-1:0] + 2^n \times Z[n] \rangle_{2^n - 2^p} \\ &= \langle Z[n-1:0] + (2^n - 2^p) \times Z[n] + 2^p \times Z[n] \rangle_{2^n - 2^p} \\ &= \langle Z[n-1:0] + 2^p \times Z[n] \rangle_{2^n - 2^p} \\ &= \langle Z[n-1:0] + \underbrace{0 \dots 0}_{(n-p-1) \text{ bits}} \# \underbrace{Z[n]}_{p \text{ bits}} \# \underbrace{0 \dots 0}_{p \text{ bits}} \rangle_{2^n - 2^p} \quad (2) \end{aligned}$$

其中 $Z[w:v]$ 表示 X 中的 v 到 w 位的对应数, 符号 $\#$ 表示不同位相连.

当 Z 为 $2n$ 位的整数时, 则有:

$$\begin{aligned} & \langle Z[2n-1:0] \rangle_{2^{n-2^p}} \\ &= \langle Z[n-1:0] + 2^p \times Z[2n-1:n] \rangle_{2^{n-2^p}} \\ &= \underbrace{\langle Z[n-1:0] \rangle_{(n-2^p) \text{ bits}}} + \underbrace{\langle Z[2n-p-1:n] \rangle_{p \text{ bits}}} \# 0 \cdots 0 \\ &= 0 \cdots 0 \# \underbrace{Z[2n-1:2n-p]}_{p \text{ bits}} \# 0 \cdots 0 \rangle_{2^{n-2^p}} \end{aligned} \quad (3)$$

2.2 第二类修正方法

本文所提出的模 $(2^n - 2^p)$ 乘法器需要使用到模 $(2^{n-p} - 1)$ 加法去修正溢出问题. 基于进位修正和并行前缀结构, 提出了模 $(2^m - 1)$ 加法器.

这种加法器包含四个模块, 即数据预处理、前缀运算、进位修正和求和模块, 其中进位修正模块解决了传统模 $(2^m - 1)$ 加法器使用两级二进制加法器的问题, 节省了延时和面积.

模 $(2^m - 1)$ 加法可以描述为: $\langle A + B \rangle_{2^m-1} = Z$, 满足 $A + B < 2^{m+1} - 2$, 且 A, B 都为 m 位.

2.2.1 数据预处理模块

数据预处理根据加数 A, B 按位生成前缀运算所需的前缀运算对:

$$\begin{aligned} g_i &= A_i \cdot B_i \\ ps_i &= A_i \oplus B_i \end{aligned} \quad (4)$$

其中, g_i 和 ps_i 分别表示第 i 位的进位生成和部分和信息, $\cdot$ 和 $\oplus$ 分别表示“与”、“异或”逻辑操作.

2.2.2 前缀运算模块

前缀运算模块用于生成求和模块所需的各比特进位信息, 采用 LF (Ladner and Fischer) 结构, 可以减小面积和延时. 定义 $\otimes$ 操作为:

$$(g, p) \otimes (g^1, p^1) = (g \mid p \cdot g^1, p \cdot p^1) \quad (5)$$

其中 $\mid$ 表示“或”逻辑操作.

LF 前缀运算为:

$$(P_{i,j}, G_{i,j}) = (P_{i,m}, G_{i,M}) \otimes (P_{m-1,j}, G_{m-1,j}) \quad (6)$$

2.2.3 进位修正模块

对于模 $(2^m - 1)$ 加法, 只有当 $A + B + 1 \geq 2^m$ 时, $\langle A + B \rangle_{2^m-1} = A + B + 1$, 反之, $\langle A + B \rangle_{2^m-1} = A + B$. 规定 c_i^0 为 $A + B (c_m = 0)$ 对应的进位输出结果, c_i^1 为 $A + B (c_m = 1)$ 对应的进位输出结果, c_i^m 为 $\langle A + B \rangle_{2^m-1}$ 即模运算真正需要的进位输出结果. c_i^0 可以由前缀运算模块输出 $G_{i-1,0}$ 得到, 即 c_i^0

$= G_{i-1,0}$. 因为 $c_m = 1$, 所以 $c_i^1 = G_{i-1,0} \mid P_{i-1,0} \cdot c_m^1$ 决定了真正进位 c_i^m 的值, 当 $c_m^1 = 1$, 说明 $A + B + 1 \geq 2^m$, 则 $c_i^m = c_i^1$, 反之, $c_i^m = c_i^0$. 所以 c_i^m 可以由以下得到:

$$\begin{aligned} c_i^m &= (c_m^1 \cdot c_i^1) \mid (\overline{c_m^1} \cdot c_i^0) \\ &= G_{i-1,0} \mid (P_{i-1,0} \cdot c_m^1) \end{aligned} \quad (7)$$

其中, $\overline{c_m^1}$ 为 c_m^1 的取反操作. c_i^m 可以由一个与门和或门来得到.

最后通过求和模块得到最终结果 Z :

$$Z_i = ps_i \oplus c_i^m \quad (8)$$

规定以上结构的模 $(2^m - 1)$ 加法操作符为 $\#$, 则有:

$$\begin{aligned} Z &= A \# B = \langle A + B \rangle_{2^m-1} \\ A + B &< 2^{m+1} - 2 \end{aligned} \quad (9)$$

这种模 $(2^m - 1)$ 加法器可以在一个周期内完成, 并且面积上也有优势.

本文中提出的模 $(2^n - 2^p)$ 乘法器结合上述第二类修正方法结构的模 $(2^{n-p} - 1)$ 加法器来修正溢出问题.

3 模 $(2^n - 2^p)$ 乘法器的结构

在本部分中, 将介绍一种新提出的模 $(2^n - 2^p)$ 乘法器. 假设 $A = a_{n-1} \cdots a_i \cdots a_0$, $B = b_{n-1} \cdots b_i \cdots b_0$, A 是被乘数, B 是乘数. 令 $A[n-1:0] \times B[n-1:0] = P[2n-1:0]$, 利用第一类修正方式, 此乘法可以表示为:

$$\begin{aligned} & \langle A[n-1:0] \times B[n-1:0] \rangle_{2^n-2^p} \\ &= \langle P[2n-1:0] \rangle_{2^n-2^p} \\ &= \langle P[n-1:p] \rangle_{p \text{ bits}} \# 0 \cdots 0 + \\ & \quad \underbrace{P[2n-p-1:n]}_{(n-2^p) \text{ bits}} \# 0 \cdots 0 + \underbrace{P[2n-1:2n-p]}_{p \text{ bits}} \# 0 \cdots 0 \\ & \quad + \underbrace{0 \cdots 0}_{n-p \text{ bits}} \\ & \quad \# P[p-1:0] \rangle_{2^n-2^p} \end{aligned} \quad (10)$$

使用一个 $(n-p)$ 位的 CSA 结构, 将式(10)中前三项的求和转换为:

$$\begin{aligned} & \underbrace{P[n-1:p]}_{(n-2^p) \text{ bits}} \# 0 \cdots 0 + \underbrace{P[2n-p-1:n]}_{p \text{ bits}} \# 0 \cdots 0 \\ & + 0 \cdots 0 \# \underbrace{P[2n-1:2n-p]}_{p \text{ bits}} \# 0 \cdots 0 \\ & \xrightarrow{\text{CSA}} L[n-1:p] 0 \cdots 0 + \underbrace{H[n-1:p]}_{p+1 \text{ bits}} \# 0 \cdots 0 \end{aligned}$$

$$= L[n-1:p] \overbrace{0 \cdots 0}^{p \text{ bits}} + H[n-2:p \# n-1] \# \overbrace{0 \cdots 0}^{p \text{ bits}} \quad (11)$$

其中, $L[n-1:p]$ 和 $H[n-1:p]$ 分别为 CSA 的部分和输出和进位输出。

$\langle A[n-1:0] \times B[n-1:0] \rangle_{2^n-2^p}$ 可以表示为:

$$\begin{aligned} & \langle A[n-1:0] \times B[n-1:0] \rangle_{2^n-2^p} \\ &= \langle L[n-1:p] \# \overbrace{0 \cdots 0}^{p \text{ bits}} + \\ & \quad H[n-2:p] \# H[n-1] \# \overbrace{0 \cdots 0}^{p \text{ bits}} \\ & \quad + \overbrace{0 \cdots 0}^{n-p \text{ bits}} \# P[p-1:0] \rangle_{2^n-2^p} \quad (12) \end{aligned}$$

修正式(12)中前两项进位溢出的问题。由于 $L[n-1:p]$ 和 $H[n-1:p]$ 不可能同时为 $2^{(n-p)}-1$, 满足 $L[n-1:p] + H[n-1:p] < 2^{(n-p)+1} - 2$, 所以式(13)结合第二类修正方法结构的模 $(2^{(n-p)}-1)$ 加法器进行修正为:

$$\begin{aligned} & T[n-1:p] \\ &= L[n-1:p] \# H[n-2:p] \# H[n-1] \\ &= \langle L[n-1:p] + H[n-2:p \# n-1] \rangle_{2^{(n-p)}-1} \quad (13) \end{aligned}$$

利用上述结果, 模 $2^n - 2^p$ 乘法器最终结果 $Y[n-1:0]$ 为:

$$Y[n-1:0] = T[n-1:p] \# P[p-1:0] \quad (14)$$

图1为本文提出的模 $(2^n - 2^p)$ 乘法器结构。

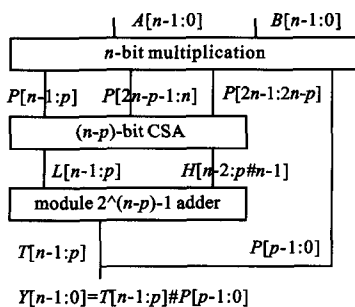


图1 模 $(2^n - 2^p)$ 乘法器结构

4 比较结果

本文中模 $(2^n - 2^p)$ 乘法器与文献[1]中的设计分别通过 Verilog 硬件描述语言进行建模, 并基于 TSMC 90nm CMOS 工艺库在 Synopsys Design-Compiler 下综合。

与文献[1]对比, 当 p 为 1、2 和 3 时, 本文设计平均延时分别节省了 4%、10% 和 15%, 平均面积分别节省了 4%、3% 和 19%。当 p 越接近 $n/2$ 时, 效果越明显。如表 1 所示。

表 1 综合对比结果

(n, p)	本文	文献[1]
	[Area(um2), Delays(ns)]	[Area(um2), Delays(ns)]
(4, 1)	[763, 1.06]	[777, 1.12]
(4, 2)	[601, 1.02]	[648, 1.14]
(6, 1)	[1307, 1.24]	[1680, 1.42]
(6, 2)	[1412, 1.34]	[1547, 1.50]
(6, 3)	[1505, 1.40]	[1754, 1.64]
(8, 1)	[2639, 1.66]	[2880, 1.78]
(8, 2)	[2520, 1.68]	[2596, 1.84]
(8, 3)	[2487, 1.66]	[2684, 1.84]

5 结束语

本乘法器结合改进的模 $(2^{(n-p)}-1)$ 加法器, 提出了一种性能更好的模 $(2^n - 2^p)$ 乘法器。整个设计简单高效, 并且可以在一个更短的时钟周期内完成模乘法运算。

参考文献:

- [1] Hiasat A A. New efficient structure for a modular multiplier for RNS[J]. IEEE Trans Computers, 2000, 49(2): 170-174.
- [2] Soderstrand M, Jenkins W, Jullien G, et al. Residue number system arithmetic modern applications in digital signal processing [M]. Piscataway, NJ, USA: IEEE Press, 1986.
- [3] Didier L S, Rivaille P Y. A generalization of a fast rns conversion for a new 4-modulus base[J]. Circuits and Systems II: Express Briefs, IEEE Transactions on, 2009, 56(1): 46-50.
- [4] Li L, Hu J, Chen Y. An universal architecture for designing modulo $(2^n - 2^p - 1)$ multipliers[J]. IEICE Electronics Express, 2011, 9(3): 193-199.
- [5] Li Lei, Zhou Lu, Zhou Wanting. An improved architecture for designing modulo $(2^n - 2^p + 1)$ multipliers [J]. IEICE Electronics Express 2012, 9(14): 1141-1146.

作者简介:

周璐女, (1988-), 硕士。研究方向数字 SoC 和 ASIC 设计。

李磊男, (1982-), 副研究员。研究方向专用集成电路 (ASIC) 和 SoC 的设计。