Radix-8 复数除法器的设计与实现

王东[1]，郑南宁[1]，Miloš D. ERCEGOVAC[2]
（1. 西安交通大学电子与信息工程学院，710049，西安；2. 加州大学洛杉矶分校计算机科学系，90024，美国洛杉矶）

摘要：设计了一种高性能、低功耗的 Radix-8 时序复数除法器. 该复数除法器采用了逐位递归算法和操作数预变换技术，并在传统结构的基础上，选用冗余形式保留预校正变量，节省了超长进位加法器的使用，缩短了关键路径的延时. 设计还通过实部和虚部商位的合并以及基于 6 输入查找表结构的硬件优化，提高了乘加逻辑单元的资源利用率. Stratix-II 型现场可编程逻辑器件仿真验证表明，与使用超长进位加法器的传统结构相比，所设计的复数除法器的速度提高了 44%，硬件资源减少了 31%.

关键词：复数除法器；逐位递归算法；现场可编程逻辑器件
中图分类号：TP332 **文献标志码**：A **文章编号**：0253-987X(2009)10-0001-06

Design and Implementation of Radix-8 Complex Divider

WANG Dong[1], ZHENG Nanning[1], Miloš D. ERCEGOVAC[2]
(1. School of Electronics and Information Engineering, Xi′an Jiaotong University, Xi′an 710049, China;
2. Department of Computer Science, University of California at Los Angeles, Los Angeles California 90024, USA)

Abstract: A Radix-8 complex divider with high performance and low power is designed. The complex divider architecture adopts the digit-recurrence algorithm and the prescaling technique of operands, and has an optimizing function, with which the prescaled operands are kept in redundant forms so that long carry propagation adders are saved. Moreover, combination of the real and imaginary parts of the internal quotient digits and optimization for hardware mapping based on the 6-input look-up table are made to improve the utilization of logic resources. The proposed design is then implemented on Stratix-II field programmable gate array. Simulation results and comparisons with the traditional structure show that the speed of the complex divider is increased by 44% and its hardware resources are reduced by 31%.

Keywords: complex divider; digit-recurrence algorithm; field programmable gate array

随着深亚纳米集成电路技术的飞速发展，低成本大容量的高性能现场可编程逻辑器件(FPGA)成为可能. 由于 FPGA 具有较短的开发周期和可编程性，因而越来越多地被用作协处理器来构建计算机系统，处理如乘累加(MAC)、浮点数运算、方程演算等计算密集型任务. 复数除法作为一种特殊的计算被应用于如 GPS 设备[1]、智能天线[2]和射频无线测控系统[3]等许多高性能计算机系统. 利用 FPGA 器件为这些计算机系统增添专用的数学协处理器，如复数乘法器和除法器[4]，可大幅度提高其整体性能. 然而，当前大多数复数除法运算还停留在软件实现层面，这些方法都是基于下式或其变形[5]

$$\frac{a+\mathrm{i}b}{c+\mathrm{i}d}=\frac{(ac+bd)+\mathrm{i}(bc-ad)}{c^2+d^2} \tag{1}$$

最近，文献[6]提出了一种适合硬件实现的快速复数除法算法. 该算法采用了逐位递归算法，可进行伸缩精度计算，同时结合操作数预变换技术，简化了商位选择电路. 虽然文献[6]提出的是一种硬件实现

收稿日期：2009-04-10. 作者简介：王东(1981—)，男，博士生；郑南宁(联系人)，男，教授，博士生导师，中国工程院院士. 基金项目：国家基础研究发展规划资助项目(2007CB311005)；国家高技术研究发展计划资助项目(2009AA01Z307，2006AA01Z192，2006AA01Z318)；国家留学基金委资助项目(2008628039).

方案，但由于使用了超长进位加法器，因此造成关键路径延时过高，降低了除法器的整体性能. 针对这一问题，本文在文献[6]的基础上，改进了传统时序复数除法器的结构，提出了一种高性能、低功耗的设计方案，并进行了 FPGA 验证.

1 逐位递归复数除法算法

本文设计的时序复数除法器采用了逐位递归定点数除法算法，该算法可按任意基 r 分解目标除法运算、按位(digit)递归求商，其计算过程遵循下述方程

$$w(j+1)=rw(j)-q_{j+1}y \tag{2}$$

式中：$w(0)=x$ 表示复数形式的被除数；y 为除数；$w(j)$为第 j 步循环所得余数；q_{j+1}表示权值为 r^{j+1} 的商位. 算法的核心在于如何从 r 的冗余商集$\{-a,\cdots,+a\}$中选取恰当的商位值 q_{j+1}，其中$(r-1)/2\leqslant a<r$，使每循环所得余数 $w(j+1)$有界，从而保证递归方程式(2)收敛. 文献[7]对多种高基除法的商选择算法进行了比较，其中操作数预变换技术采用简化的四舍五入运算进行商的选择，非常适合硬件电路实现. 算法的思想是：首先根据截取了精度的除数 $\hat{d}$ 求出变换系数 $K=K_1+\mathrm{i}K_2=1/\hat{d}$，然后对各操作数进行缩放变换

$$\left.\begin{aligned}x^{\mathrm{R}}&=z^{\mathrm{R}}K_1-z^{\mathrm{I}}K_2\\x^{\mathrm{I}}&=z^{\mathrm{I}}K_1+z^{\mathrm{R}}K_2\\y^{\mathrm{R}}&=d^{\mathrm{R}}K_1-d^{\mathrm{I}}K_2\\y^{\mathrm{I}}&=d^{\mathrm{I}}K_1+d^{\mathrm{R}}K_2\end{aligned}\right\} \tag{3}$$

式中：$z=z^{\mathrm{R}}+\mathrm{i}z^{\mathrm{I}}$ 和 $d=d^{\mathrm{R}}+\mathrm{i}d^{\mathrm{I}}$ 分别表示原复数形式的被除数和除数；$x=x^{\mathrm{R}}+\mathrm{i}x^{\mathrm{I}}$ 和 $y=y^{\mathrm{R}}+\mathrm{i}y^{\mathrm{I}}$ 表示经过预变换后的被除数和除数. 处理后的除数 y 具有以下特性：$y^{\mathrm{R}}\approx 1-\varepsilon$ 和 $y^{\mathrm{I}}\approx 0$，其中 ε 为误差控制参数. 由此，商值可以简单选择为 $r\hat{w}(j)$四舍五入后的整数部分，其中 $\hat{w}(j)$为第 j 步循环余数截取至特定小数位后的近似值. 此外，算法还保证了商的实部和虚部相互独立，则式(2)可进一步分解为 2 部分并行求解

$$\left.\begin{aligned}w^{\mathrm{R}}(j+1)&=rw^{\mathrm{R}}(j)-q_{j+1}^{\mathrm{R}}y^{\mathrm{R}}+q_{j+1}^{\mathrm{I}}y^{\mathrm{I}}\\w^{\mathrm{I}}(j+1)&=rw^{\mathrm{I}}(j)-q_{j+1}^{\mathrm{I}}y^{\mathrm{R}}-q_{j+1}^{\mathrm{R}}y^{\mathrm{I}}\end{aligned}\right\} \tag{4}$$

与原递归式(2)相比，式(4)每个方程(实部和虚部)只各增加了一个乘积项，余数的边界条件和新递归式的收敛条件文献[6]中已给出了详细的证明，本文不再阐述.

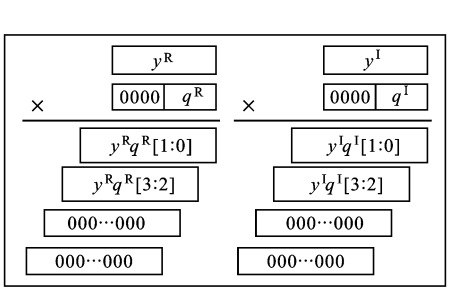
2 除法器的结构设计

对比式(3)、式(4)可以看出，预变换和递归求商具有相类似的运算结构：两者均由 4 个乘法运算和若干加法运算构成($rw(j)$可通过移位实现)，预示着 2 种运算可共享相同的硬件电路实现.

文献[6]根据上述特性，提出了一种时序复数除法器的设计方案. 在最初的 2 个时钟周期内，电路先进入预变换模式，按式(3)执行变换系数和操作数的乘累加运算，所得中间结果$(y^{\mathrm{R}},y^{\mathrm{I}})$与$(x^{\mathrm{R}},x^{\mathrm{I}})$被暂存至寄存器中. 完成预变换后，电路进入循环求商模式，商选择电路根据上一步运算所得余数 $rw(j)$选取新商值$(q_{j+1}^{\mathrm{R}},q_{j+1}^{\mathrm{I}})$，将其送入同一乘法电路求得式(4)中各乘积项，然后将所得结果与余数送入压缩器求和并完成本次循环. 虽然该实现方案共享了同一硬件电路进行预变换和求商运算，但其效率较低，主要表现为：

(1)预变换后的除数$(y^{\mathrm{R}},y^{\mathrm{I}})$被保留为定点数形式，用于完成进位和累加的 2 个加法器(CPA)只在前 2 个时钟周期内被使用，利用率不超过 23%，同时该 CPA 位于电路的关键路径上，并且进位链较长(随操作数精度不同从 24 到 80 b 不等)，严重地影响了电路的时序性能；

(2)如图 1 所示的预变换和循环求商运算的数据通路的资源共享方式，在通常情况下，其预变换系数(K_1,K_2)比商位$(q^{\mathrm{R}},q^{\mathrm{I}})$具有更高位宽，如在设计参数 $r=8$、$a=7$、$p=6$(其中 a 为可选商值的最大值，p 为预变换计算精度)时[6]，(K_1,K_2)各具有 8 b 宽度，商$(q^{\mathrm{R}},q^{\mathrm{I}})$用 4 b 表示，在求余运算时，乘数$(q^{\mathrm{R}},q^{\mathrm{I}})$需要进行符号位扩展，造成 50%的电路只用



(a)预变换模式

(b)循环求商模式

图 1 不同模式下数据通路的资源利用率

于符号位(0 或 1)累加，没有用于有意义的计算.

针对上述问题，本文提出了一种更高效的除法器结构：首先，保留预变换后的除数(y^{R}，y^{I})为冗余的进位(carry)和(sum)形式，分别记为($y_{\mathrm{s}}^{\mathrm{R}}$，$y_{\mathrm{c}}^{\mathrm{R}}$)和($y_{\mathrm{s}}^{\mathrm{I}}$，$y_{\mathrm{c}}^{\mathrm{I}}$)，重新代入式(4)得

$$\left.\begin{aligned} w^{\mathrm{R}}(j+1) &= rw^{\mathrm{R}}(j) - q_{j+1}^{\mathrm{R}} y_{\mathrm{s}}^{\mathrm{R}} - q_{j+1}^{\mathrm{R}} y_{\mathrm{c}}^{\mathrm{R}} + \\ &\quad q_{j+1}^{\mathrm{I}} y_{\mathrm{s}}^{\mathrm{I}} + q_{j+1}^{\mathrm{I}} y_{\mathrm{c}}^{\mathrm{I}} \\ w^{\mathrm{I}}(j+1) &= rw^{\mathrm{I}}(j) - q_{j+1}^{\mathrm{I}} y_{\mathrm{s}}^{\mathrm{R}} - q_{j+1}^{\mathrm{I}} y_{\mathrm{c}}^{\mathrm{R}} - \\ &\quad q_{j+1}^{\mathrm{R}} y_{\mathrm{s}}^{\mathrm{I}} - q_{j-1}^{\mathrm{R}} y_{\mathrm{c}}^{\mathrm{I}} \end{aligned}\right\} \quad (5)$$

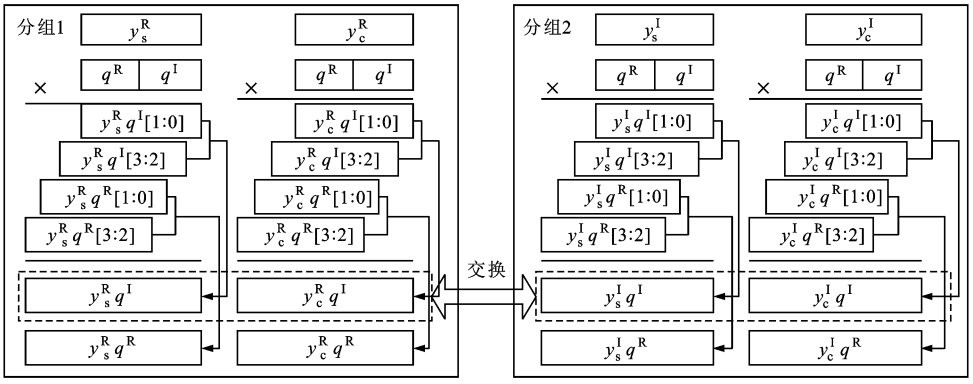


图 2　本文所设计的共享硬件资源的乘加计算数据通路

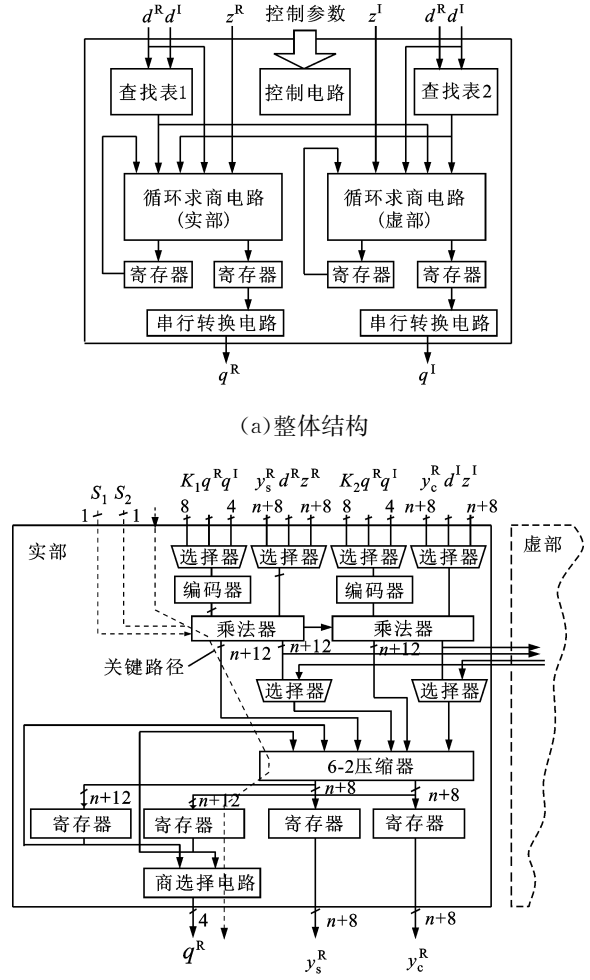
上面每个方程的第 2 和第 3 项(分组 1)均包含相同的乘数($y_{\mathrm{s}}^{\mathrm{R}}$，$y_{\mathrm{c}}^{\mathrm{R}}$)，第 4 和第 5 项(分组 2)含有相同的乘数($y_{\mathrm{s}}^{\mathrm{I}}$，$y_{\mathrm{c}}^{\mathrm{I}}$)，并且每组均对应相同的乘数组合($q^{\mathrm{R}}$，$q^{\mathrm{I}}$)或($q^{\mathrm{I}}$，$q^{\mathrm{R}}$)，则可以按照图 2 所示方式重新构造乘加电路.

在循环求商模式下，每步所得新商位(q^{R}，q^{I})被合并成一个操作数后送入乘法器，实部和虚部的各部分积按照不同分组进行累加后通过选择器交换部分结果(图中虚线部分)，得到符合式(5)的组合. 新的乘法器中，部分积的求和电路由 2 个并行的加法器代替了原有的 4-2 压缩器，因其位宽较窄，可直接利用算术模式下的可重构逻辑单元(ALUT)(包含一级逻辑单元(LUT)和一级快速进位加法链)[8]来实现. 在预变换模式下，电路仍按照图 1a 进行计算，所得的部分积分组相加后送入压缩器，完成累加.

新除法器结构的优点：①由于变换后除数保留为冗余形式，节省了 2 个超长进位加法器(CPA)，提高了关键路径上的时序性能；②通过合并操作数和重组乘积项，使乘加数据通路的硬件资源利用率提高到了 100%；③用一级并行的短进位加法器替换了原有的 4-2 压缩器，在不影响性能的前提下充分利用 FPGA 片上硬件加法器，提高了芯片资源利用率. 所付出的代价只是增加了少量的选择器电路.

3　FPGA 实现

本文选取 Altera 公司的 Stratix-II S60 FPGA 实现了所提结构的复数除法器，其结构框图如图 3 所示.



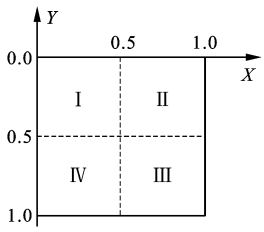
(a)整体结构

(b)核心部分逻辑结构

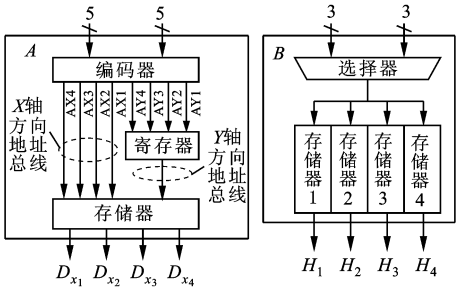
图 3　本文复数除法器结构框图

根据上述原理可知，为了能够达到100%的数据通路资源利用率，预变换系数 K 的位宽必须为商位的2倍. 因此，我们分析了不同基(从 $r=4$ 到 $r=64$)和计算精度下的设计参数[6]，选取符合条件的一组设计参数：$r=8, a=7, p=6, \sigma=3$，其中 σ 为商选择电路输入余数的小数位数. 电路的具体实现如下：输入复数形式的被除数和除数的宽度均为 $n+n$(实部和虚部各 n 位，n 为16～64的任意整数)；变换系数 K_1(实部)和 K_2(虚部)存储在2个8 b宽(1 b整数7 b小数)的系数表内，其符号位 S_1 和 S_2 单独生成并直接送入乘法器与另一乘数的符号位异或后决定部分积的符号，变换后除数($y^{\mathrm{R}}, y^{\mathrm{I}}$)被暂存在寄存器内供后续的各循环使用. 备选商数集为{－7，－6，…，＋6，＋7}，用4 b符号数表示，由商选择模块求出后合并为新8 b操作数，反馈至Booth编码器，编码为12 b乘数后送入乘法器. 与此同时，商位($q_j^{\mathrm{R}}, q_j^{\mathrm{I}}$)被送入一个在线(on-the-fly)串行转换电路[7]，合并不同权值的结果，在最后一个循环转换为 $n+2$ 位定点数，其中1位为舍入位，1位为保护位，并在输出前四舍五入至 n 位. 2个($n+8$)×12 b乘法器被用来完成乘法运算，递归式(5)的累加运算由2个($n+12$)位6-2压缩器实现，各操作数均保留为进位和形式. 每循环累加完成后，余数 $w(j)$ 被截取至小数点后 σ 位，送入商选择电路选取下一权值的商位($q_{j+1}^{\mathrm{R}}, q_{j+1}^{\mathrm{I}}$). 设计的关键路径如图3中虚线所示，从Booth编码逻辑经过乘法器、压缩阵列至商选择电路结束. 由于篇幅原因，本文下面只给出3个关键模块的详细实现.

(1)预变换系数查找表　算法中预变换系数 K 由截取了精度的除数的倒数 $1/\hat{d}$ 求出，存储在查找表中. 常见的查找表构造方法有单查找表和多子表(multipartite)[9]构造方法，这2种方法在构造二元复数倒数时生成的查找表资源均较大[10]. 根据算法要求，K_1 和 K_2 各需要建立一个查找表，按照文献[9]的方法，每个查找表输入地址位宽为15 b，数据位宽为8 b，单查找表大小约为256 kb，多子表结构大小约为144.5 kb，将占用1/4的M4K存储单元，消耗过大. 为了缩减存储资源消耗，本文采用双3次插值方法减少所需存储数据量：首先，根据输入操作数的对称性只生成落入复数坐标系第4象限的系数；再根据其归一化特性，将该象限进一步划分位4个等分区域(如图4a所示)，对于Ⅱ、Ⅲ、Ⅳ 3个子区域内系数各生成一个由2个子表组成的查找表，其结构如图4b所示，其中子表 A 负责读出横坐标 X 方向最近临4个点的值 D_{x1}～D_{x4}，子表 B 负责读出对应的双3次插值系数 H_1～H_4，然后进行4次横坐标方向插值 $D_y=\sum_{i=1}^{4} D_{x_i} H_i$，得到纵坐标 Y 方向4个最近临点函数值；最后再进行一次3次插值运算便可得到最终系数值 $K_1(K_2)$. 由于落入第4象限的系数总为正值，因此符号位被节省了. 采用该方法，最终的查找表大小缩减为4.572 kb，约为多子表方法的0.3%，只占用0.5%的M4K存储资源.



(a)坐标划分方法



(b)系数表的结构

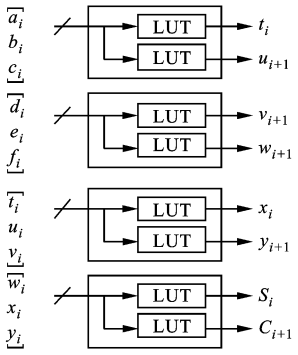
图4　预变换系数表的构造方法

(2)改进的Booth编码器　设计中每个($n+8$)×12乘法器均使用了1个8位Radix-4 Booth编码器. 由于在循环递归模式下进行了商位的合并，直接使用传统Booth编码器进行编码会导致2个子操作数间产生错误进位. 针对这一问题，本文在第3位输入逻辑处增加了一位屏蔽信号 M(如式(6)所示)，在预缩放模式下，M 信号被置1，编码电路从 y_0～y_7 位进行常规Booth编码；在循环递归模式下，M 信号被置0，编码被分成 y_0～y_3 和 y_4～y_7 位2个独立编码电路. 这样，在所得第2和第3项部分积之间不再具有进位，从而实现了实部和虚部的独立乘法运算，其电路逻辑关系遵循下式

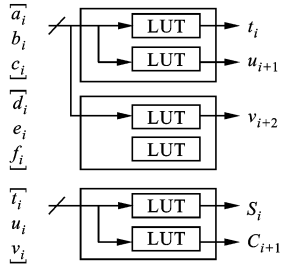
$$b_6 = y_5\bar{y}_4(\bar{y}_3+\overline{M})+\bar{y}_5 y_4 y_3 M \tag{6}$$

(3)优化了的6-2压缩器　传统的6-2压缩器通常由3级3-2压缩器实现(Quartus-Ⅱ自动综合后电路)，图5a给出了压缩器的第 i 列进位关系(下

标 i 代表权值），每个 3-2 压缩器将占用 2 个 LUT，而每列 6-2 压缩器将消耗 8 个配置成 3 输入方程的 LUT. 本文则通过使用 WYSIWYG 底层原语进行 RTL 编码，强制使用 3 个配置成 6 输入逻辑方程和 2 个配置成 3 输入方程的 LUT 的 2 级逻辑实现了 6-2 压缩器，如图 5b 所示，可节省 37.5% 的逻辑资源.



(a)传统的自动综合电路结构



(b)本文手动优化后的电路结构

图 5　2 种 6-2 压缩器实现比较

4　实验结果及分析

本文采用 VHDL 语言在逻辑层实现了所提结构的时序复数除法器，并进行了功能仿真和验证，最后在 Altera StratixII S60F672C3 FPGA 上实现了完整电路.

首先，为了验证本文除法器结构的性能，本文还同时实现了相同精度（$n=16$）的 2 个参考设计，并进行了资源和性能等多方面的比较，结果在表 1 中给出. 其中，第 1 个参考设计（方法 A）采用了文献[6]所提出的使用 CPA 的结构，第 2 个参考设计（方法 B）则直接根据公式(1)，利用 Altera 提供的乘法器和除法器 IP 进行了实现. 对比表 1 中数据可以看出，本文所提结构在关键路径上比文献[6]的时序性能提高了 44%，同时还节省了 31% 的逻辑资源，主要贡献来自于 CPA 的节省和压缩器的优化. 虽然与方法 A 相比，本文设计付出了略多的动态功耗，但却获得了 1.8 倍的运算速度和更优的资源利用率，使其特别适合于对计算能力和资源有较高要求的 FPGA 设计应用. 与方法 B 相比，本文所提出的结构只使用了其 24.5% 的资源，却达到了 2 倍以上的运算速度，特别显著的是功耗只为原功耗的 8.8%，使得本文设计特别适合于低功耗芯片的设计与集成.

表 1　不同结构除法器电路的综合结果对比

结构	逻辑单元量/个	关键路径延时/ns	最高频率/MHz	功耗/mW
本文结构	1 535	10.5	95.4	51.05
方法 A 结构	2 242	18.9	52.8	21.80
方法 B 结构	6 253	22.7	44.1	577.10

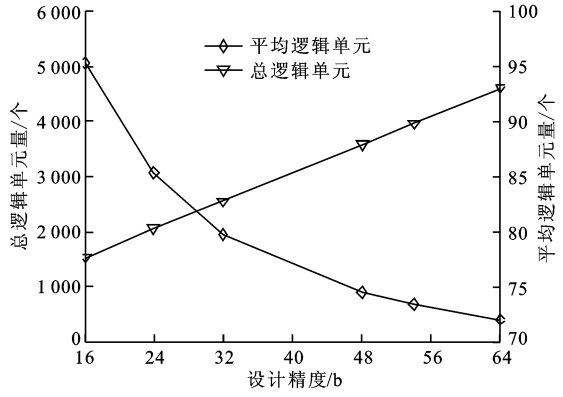
注：由于在同等 PVT 条件下相同器件的静态功耗相同，表中只对比了核心逻辑的动态功耗.

最后，本文还实现了从 16 到 64 b 共 6 个不同精度设计，涵盖了常见的单精度、双精度以及扩展精度. 图 6 对比了不同操作数精度下除法器的逻辑资源、时序性能和功耗 3 方面信息. 从图 6a 可以看出，总的逻辑资源消耗量 C 随操作数精度 n 的增大线性地升高，可由下述关系式表示

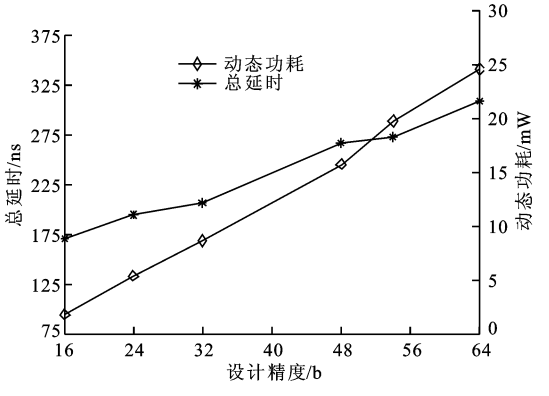
$$C = Pn + Q \tag{7}$$

式中：P 为每增加单位比特计算量所需的逻辑资源；Q 为线性函数的偏移量，包括了与精度无关的电路资源，如控制逻辑等. 根据图中数据，可以求得 $P=64$，$Q=500$. 进一步分析图中数据还可以看出，随着设计精度的增大，平均资源消耗量 C/n 呈非线性递减，并逐渐趋近于单位比特资源增量 P. 从图 6b 中可以看出，除法器的总延时和动态功耗也近似随操作数精度呈线性增长，其中总延时定义为关键路径延时乘以完整复数除法运算所需的时钟周期数，动态功耗由 Quartus-II Powerplay Analyzer 生成.

利用公式(7)以及图 6 中的数据，设计人员可在具体实现电路前快速地估算出不同精度下除法器的逻辑资源消耗情况并进行相应的时序、功耗评估. 因此，本文所提结构的时序复数除法器具有可预测的资源量与性能，适合作为独立 IP 应用于 SoC 芯片集成或作为算术逻辑单元用于专用协处理器的构建.



(a)逻辑资源消耗



(b)时耗

图6 不同设计精度的逻辑资源消耗和时耗对比

5 结 论

本文提出了一种新的高基时序复数除法器结构,与已有设计相比,节省了超长进位加法器CPA的使用,优化了关键路径的时序性能,同时还巧妙地利用了商合并提高了数据通路的资源复用率,特别是与基于Altera IP的设计相比,具有极低的功耗.此外,本文还给出了该结构除法器资源、延时和功耗信息的估算方法,方便了芯片设计人员的前期预算与评估.本文结构的复数除法器特别适用于数学协处理器设计或SoC集成,具有非常广泛的工程应用前景.为了更准确地评估关键路径的时序性能,本文设计只采用了一级流水结构,对于其他更高性能的实际应用,可考虑进一步增加流水线.

参考文献:

[1] XIANG Jianhong, GUO Lili, CHEN Ying, et al. Study of GPS adaptive antenna technology based on complex number AACA [C]// Proceedings of IEEE International Conference on Wireless Communications, Networking and Mobile Computing. Piscataway, NJ, USA: IEEE, 2008: 1-4.

[2] EDMAN F, OWALL V. Compact matrix inversion architecture using a single processing element [C]// Proceedings of IEEE International Conference on Electronics, Circuits and Systems. Piscataway, NJ, USA: IEEE, 2005: 1-4.

[3] VANDERSTEEN G. Comparison of arithmetic functions with respect to Boolean circuits [C]// Proceedings of 58th ARFTG Conference Digest RF Measurements for a Wireless World. Piscataway, NJ, USA: IEEE, 2001: 466-470.

[4] LIU Jie, WEAVER B, ZAKHAROV Y. FPGA implementation of multiplication-free complex division [J]. Electronic Letters, 2008,44(2): 95-96.

[5] SMITH R. Algorithm 116: complex division [J]. Communications of the ACM, 1962, 5(8): 435.

[6] ERCEGOVAC M, MULLER J. Design of a complex divider [C]// Proceedings of SPIE on Advanced Signal Processing Algorithms, Architecture and Implementations XII. Bellingham, WA, USA: SPIE, 2004: 51-59.

[7] ERCEGOVAC M, LANG T. Division and square root: digit-recurrence algorithms and implementations [M]. Bostons, MA,USA: Kluwer Academic Publishers, 1994.

[8] LEWIS D. The Stratix-II logic and routing architecture[C]// Proceedings of International Symposium on Field Programmable Gate Arrays. Piscataway, NJ, USA: IEEE, 2005: 14-20.

[9] DE DINECHIN F, TISSERAND A. Multipartite table methods [J]. IEEE Transactions on Computers, 2005,54(3): 319-330.

[10] WANG Dong, ERCEGOVAC M. Design and FPGA implementation of complex fixed-point dividers [EB/OL]. (2009-03-25)[2009-04-01]. http://fmdb.cs.ucla.edu/Treports/090032,pdf.

(编辑 刘杨)