第 34 卷 第 23 期 计 算 机 工 程 2008 年 12 月
Vol.34 No.23 Computer Engineering December 2008
·工程应用技术与实现· 文章编号：1000—3428(2008)23—0247—02 文献标识码：A 中图分类号：TP309.7

基于 FPGA 的有限域乘法算法的分析和比较

鲍可进，郑 博
(江苏大学计算机学院，镇江 212013)

摘 要：介绍椭圆曲线密码系统和超椭圆曲线密码系统中的乘法模块，在现有的 3 种乘法算法基础上，设计乘法的硬件框图，并用 VHDL 语言加以实现，同时对其实现速度和芯片面积进行比较。实验结果表明，在 4 个不同乘法器的实现方案中，8 bit 串并混合乘法器的整体性能较优。

关键词：现场可编程门阵列；椭圆曲线密码系统；超椭圆曲线密码系统

Analysis and Comparison of Finite Field Multiplier Algorithm Based on FPGA

BAO Ke-jin, ZHENG Bo
(College of Computer, Jiangsu University, Zhenjiang 212013)

【Abstract】 A multiplier module in Elliptic Curve Cryptosystem(ECC) and Hyper-Elliptic Curve Cryptosystem(HECC) is introduced. On the basis of three existed multiplier algorithms, the hardware configuration is designed and implemented by using VHDL. Moreover, the speed of implementation and the chip area are compared separately. Experimental results show the 8 bit parallel mixed multiplier has better performance than other multipliers.

【Key words】 Field Programmable Gate Array(FPGA); Elliptic Curve Cryptosystem(ECC); Hyper-Elliptic Curve Cryptosystem(HECC)

1 概述

有限域乘法运算是实现椭圆曲线密码系统(Elliptic Curve Cryptosystem, ECC)和超椭圆曲线密码系统(Hyper-Elliptic Curve Cryptosystem, HECC)的基础。相对于ECC，HECC具有更高加密强度，占用的带宽更少，能在较小的域上进行运算，因此，许多研究人员对其进行了研究[1]。由于现场可编程门阵列(Field Programmable Gate Array, FPGA)具有可重配置的特点，因此基于FPGA实现HECC已成为该领域的研究热点。

文献[2-3]论述了 HECC 中有限域的乘法算法，按实现的并行度可将这些算法分为 3 类：(1)全串行算法，(2)全并行算法，(3)混合算法。按运算法则可分为高位优先算法和低位优先算法。

本文分析比较几种算法的特点，并针对$GF(2^{83})$乘法器给出有限域乘法的FPGA实现方案。

2 相关知识

$GF(2^n)$域是特征为 2 的有限域，它包含 2^n个元素。构成 $GF(2^n)$ 的方法是采用多项式基表示法。在该表示法中，域 $GF(2^n)$ 的元素是次数最多为n-1 次的二进制多项式，其中，多项式系数取自$GF(2)=\{0, 1\}$：

$$GF(2^n)=\{a_{n-1}x^{n-1}+a_{n-2}x^{n-2}+\cdots+a_1x+a_0, a_j\in GF(2), 0\le j\le n-1\}$$

选择一个二进制域上的 n 次不可约多项式 $f(x)$，$f(x)=x^n+p_{n-1}x^{n-1}+p_{n-2}x^{n-2}+\cdots+p_1x+p_0$ (对任意 n，这样的多项式一定存在且能有效产生)。有限域加法由 2 个多项式系数的模 2 相加(按位异或)实现，乘法由 2 个多项式相乘后取模$f(x)$实现。

例如：有限域$GF(2^4)$的不可约多项式为 $f(x)=x^4+x+1$，则 $(x^3+x^2+1)(x^2+x+1)=x^2+1$ ，因为 $(x^3+x^2+1)(x^2+x+1)=x^5+x+1$ 且 $(x^5+x+1) \bmod (x^4+x+1)=x^2+1$，所以这 2 个多项式相乘的结果为 x^2+1。

3 有限域乘法的原理及算法分析

有限域乘法的实质是 2 个多项式相乘后再求余一个规约多项式。按运算法则可分为高位优先算法和低位优先算法。设 $A=\sum_{i=0}^{n-1}a_ix^i$，$B=\sum_{j=0}^{n-1}b_jx^j$，规约多项式为 $f(x)$，因为

$$C=AB \bmod f(x)=b_0A+b_1(Ax \bmod f(x))+b_2(Ax^2 \bmod f(x))+\cdots+b_{n-1}(Ax^{n-1} \bmod f(x)) \quad (1)$$

和

$$C=AB \bmod f(x)=(\cdots(Ab_{n-1}x \bmod f(x)+Ab_{n-2})x \bmod f(x)+\cdots+Ab_1)x \bmod f(x)+Ab_0 \quad (2)$$

所以若按式(2)的方法进行有限域乘法，每次规约操作都会面对不同次数的多项式。而高次方的规约需通过逐级降次的方法实现，即每次规约都重复前面的工作。因此，本文采用高位优先算法。

有限域乘法算法按其并行度可分为全串行算法、全并行算法以及混合算法。其中，全串行算法速度较慢，占用空间少；全并行算法要求在一个时钟内完成全部计算过程，速度快，但会牺牲大量空间，硬件开销大，若多项式阶位较高则不适用；串并混合算法结合了以上 2 种算法的优点，是本文研究的对象。全串行算法和混合算法如下：

作者简介：鲍可进(1958－)，男，教授，主研方向：嵌入式网络控制；郑 博，硕士研究生

收稿日期：2008-04-10 **E-mail：**jinmingerdj@126.com

算法 1 高位优先的全串行域乘法算法

Input: $a,b \in GF(2^n)$, reduction polynomial f

Output: $c = a \times b$

c ← 0

for i from n-1 downto 0

if($b_i = 1$) then c ← (c + a) << 1 else c ← c << 1

if(shift carry=1) then c ← c + f

if($b_0 = 1$) then c ← c + a

return c

算法 2 高位优先的混合型域乘法算法

Input: $a,b \in GF(2^n)$, reduction polynomial f

Output: $c = a \times b$

c ← 0

or i from $\lceil n/D \rceil$ downto 0

$c \leftarrow b_{\{D_i,\ldots,D_{i-D+1}\}}(ax^{D_{i-D+1}}) \bmod f + c$

return c mod f

算法 1 的实质是 2 个多项式元素的移位相加。该算法需要大量移位操作，不适合处理器的实现方法。在软件中实现移位操作是很费时的，但在硬件中却容易实现且不占用任何逻辑资源。在用 FPGA 实现时，每个时钟只完成一次迭代计算，共需 *n* 个时钟。当 *n* 较大时，采用该算法设计的乘法器实现速度较慢。

算法 2 在用 FPGA 实现时，每个时钟可完成 *D* 次迭代运算，即在同一个时钟内完成 *D* 次累加、左移和约简的过程。此时需要的时钟数为[*n*/*D*]+1 个，乘法运算的速度将成倍提高。因为速度的提高是以空间的消耗为代价，所以这里应找一个平衡点，使得在空间消耗允许的情况下，速度达到最快。

本文将实现 83 bit 全串行乘法器以及具有不同并行度 *D* 的 83 bit 串并混合乘法器，其中，*D* 分别为 4,8,16，同时给出硬件框图，进行仿真实验，最后对它们的速度和占用空间进行比较，得出实验结果。

4 算法实现的硬件结构及实验结果

4.1 高位优先全串行乘法器的实现

硬件框图如图 1、图 2 所示。

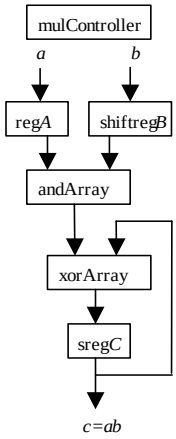

图 1 串行乘法器

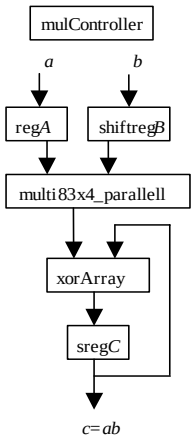

图 2 *D*=4 的串并混合乘法器

在图 1 中，所有寄存器都是 83 bit，shiftreg*B* 每次从最高位移出一位送到 andArray 模块，andArray 模块根据 shiftreg*B* 移出的一位是 1 还是 0 决定送 reg*A* 的值还是送 0 给 xorArray 模块。xorArray 模块是宽度为 83 bit 的异或门阵列。sreg*C* 也是移位寄存器，与 shiftreg*B* 不同的是 sreg*C* 要根据寄存器最高位的值判断是否需要加规约多项式。sreg*C* 的输出为 83 bit。mulController 是控制模块，它对乘法器中其他模块进行复位。另外，mulController 还对输入时钟进行计数，当达到计数值时，停止其他模块并输出结束信号。

4.2 高位优先混合乘法器的实现

在图 2 中，shiftreg*B* 每次移出 4 bit，multi83x4_parallel 是 83×4 bit 的组合电路乘法器(包括规约操作)，sreg*C* 的设计较复杂，其余模块与全串行乘法器相同。需要注意的是 shiftreg*B* 最初移出的 4 bit 是前面添加零的 4 位数(因为 83 不能被 4 整除)。

除了并行度 *D*=4 的串并混合乘法器外，本文还将实现并行度 *D* 为 8 和 16 的串并混合乘法器。这 2 个乘法器的硬件框图与图 2 相似，主要是将其中 83×4 bit 的组合电路乘法器单元(multi83x4_parallel 单元)换为 83×8 bit 和 83×16 bit 的组合电路乘法器单元，因此，这里不再单独给出它们的硬件结构框图。

4.3 实现结果及比较分析

本文的乘法器程序由VHDL语言编写，几个$GF(2^{83})$乘法器的设计在Altera公司Quartus II-5.0 上进行编译，并针对EP1S10F780C6 FPGA芯片进行综合、布局布线。由于所设计的时钟周期为 10 ns，因此最高频率可达到 100 MHz。图 3、图 4 分别是全串行乘法器、4 bit串并混合乘法器的实现结果。

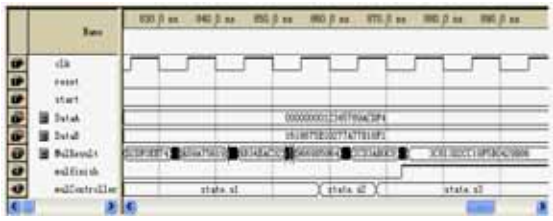

图 3 全串行乘法器实现结果

图 4 4 位串并混合乘法器实现结果

在以上波形图中，*clk* 是设计中输入的时钟周期(本文的设计中均用 10 ns)；*reset* 是复位(高电平有效)；*start* 是乘法器开始；*DataA* 和 *DataB* 是相乘的 2 个数；*MulResult* 是乘法器不同状态下的运算结果；*mulfinish* 为高电平时代表运算结束；*mulController* 显示乘法器运行的各个状态。

在乘法器设计过程中速度和空间是相互矛盾的，即速度越快，设计所需空间越大。而笔者期待的是速度快、占用空间少的设计，既不能为提高速度而占用大量空间，也不能为节省空间而是使速度过慢。这就需要找到速度和占用空间的平衡点，即时间和面积的平衡点。表 1 给出了几个乘法器设计占用的空间，并以全串行乘法器的速度为基准，对它们的速度和占用空间进行比较。

表 1 几种乘法器速度和占用空间的对比

乘法器	速度/倍数	逻辑单元数	引脚数
全串行乘法器	1	265	253
4 bit 串并混合乘法器	4	438	253
8 bit 串并混合乘法器	8	697	253
16 bit 串并混合乘法器	16	1 147	253

(下转第 251 页)