文章编号:1001 - 9081(2011)02 - 0540 - 03
doi:10.3724/SP.J.1087.2011.00540

基于 FPGA 的 F_2^m 域椭圆曲线点乘的快速实现

魏东梅,杨　涛
(西南科技大学 信息工程学院,四川 绵阳 621010)
(weidongmei_2004@163.com)

摘　要:椭圆曲线点乘的实现速度决定了椭圆曲线密码算法(ECC)的实现速度。采用蒙哥马利点乘算法,其中模乘运算、模平方运算采用全并行算法,模逆运算采用费马·小定理并在实现中进行了优化,完成了椭圆曲线点乘的快速运算。采用 Xilinx 公司的 Virtex-5 器件族的 XCV220T 作为目标器件,完成了综合与实现。通过时序后仿真,其时钟频率可以达到 40 MHz,实现一次点乘运算仅需要 14.9 μs。

关键词:椭圆曲线密码算法;模乘;点乘
中图分类号: TP309; TP302　　**文献标志码**:A

Fast implementation of point multiplication over elliptic curve F_2^m based on FPGA

WEI Dong-mei, YANG Tao
(*College of Information Engineering, Southwest University of Science and Technology, Mianyang Sichuan* 621010, *China*)

Abstract: The implementation speed of Elliptic Curve Cryptography (ECC) depends on the implementation speed of elliptic curve point multiplication. Point multiplication of elliptic curve using Montgomery algorithm was proposed in this paper. Parallel algorithm was used in modular multiplication algorithm and modular square algorithm, as well as Fermat's Little Theorem was used and optimized in modular inversion, thus the fast operation of elliptic curve point multiplication was implemented. Synthesis and implementation were realized in a Xilinx device of XC5VLX220T. Through timing simulation, the clock frequency can achieve 40MHz. It takes only 14.9μs to carry out one point multiplication operation.

Key words: Elliptic Curve Cryptography (ECC); modular multiplication; point multiplication

椭圆曲线密码算法(Elliptic Curve Cryptography,ECC)与 RSA 和数字签名算法(Digital Signature Algorithm, DSA)相比,在保证相同安全强度下所需的密钥长度短,特别适用于无线系统或者存储受限的设备。在 ECC 中,核心运算椭圆曲线点乘($Q = kP$),是 ECC 快速实现的关键。在 ECC 的实现中,可以基于 F_q 域或者 F_2^m 域,文献[1]讨论了在 F_q 域中在仿射坐标下实现椭圆曲线标量乘。由于 F_2^m 域中的加法运算就是进行异或运算,特别适合硬件实现,因此 ECC 的硬件实现一般采用在 F_2^m 域中实现。目前在硬件实现中对于改进椭圆曲线点乘运算做了很多研究[2-6]。文献[2]采用分层结构,关键运算单元采用 Karatsuba-Ofman 算法,基于快速傅里叶变换(Fast Fourier Transform, FFT)思想对输入数据进行分解,但其没有对求逆算法进行讨论。文献[3]通过对点加、倍加进行结构优化实现快速的点乘算法。文献[4]采用 LSD 算法对模乘运算实现了优化。本文采用实现效率高的蒙哥马利点乘算法,在模乘、模逆以及整体系统结构上进行了优化,达到了良好的效果。

1　椭圆曲线点乘的设计

椭圆曲线点乘运算由模加、模乘、模平方以及模逆运算组成。在仿射坐标下进行点乘运算需要多次的模逆运算,由于模逆运算的复杂度以及所耗费的时间很多,为了避免多次的模逆运算,普遍采用在投影坐标下计算椭圆曲线点乘。相比于倍加-点加椭圆曲线点乘算法,蒙哥马利椭圆曲线点乘算法的复杂度更低。

蒙哥马利点乘的整个运算过程分为坐标转换、主循环的点加和倍加运算以及最后的模逆运算 3 部分,其实现效率主要由主循环的点加和倍加运算决定。点加、倍加运算由模乘和模平方运算组成,因此其实现效率主要由模乘与模平方运算决定。模逆运算有高的复杂度,而且在点乘运算中必不可少,也须对其进行相应的优化。

由于在 ECC 中,多项式 $f(x)$ 为稀疏多项式,广泛采用的多项式是三项式或者五项式,即 $f(x)$ 的二进制表征仅仅只有 3 或者 5 位为"1",其余值全为 0,因此在模运算中有冗余信息。根据这一思想,本文的模乘运算采用一种并行的方式予以实现。在后面的讨论中,本文采用的是三项式基 $f(x) = x^m + x^k + 1$,其中 $m = 191, k = 9$。

1.1　系统的设计

由于模平方运算可以方便地用较少的资源以并行方式实现[7],因此主循环的实现效率主要由模乘运算决定。在主循环中,点加与倍加之间的数据量没有依赖,可以并行进行。表 1为一次循环中,点加和倍加分别采用并行、串行、部分并行的方式所需的乘法器数目与时间开销。

表 1　椭圆曲线点乘主循环的乘法器开销与复杂度

方案编号	点加	倍加	点加-倍加	乘法器	复杂度
1	串行	串行	串行	1	6
2	并行	并行	并行	6	2
3	部分并行	并行	并行	4	2
4	部分并行	并行	串行	2	3

收稿日期:2010 - 07 - 27。

作者简介:魏东梅(1974 -),女,四川绵阳人,讲师,硕士研究生,主要研究方向:信息安全;　杨涛(1972 -),男,四川绵阳人,副教授,博士研究生,主要研究方向:MEMS、无线传感网络。

如表 1 所示,采用全串行的方案是虽然只使用了一个乘法器,但是花费的时间最长。由于在点加运算中,第二次乘法运算依赖于第一次乘法的输出,因此在点加运算中采用部分并行的方案较为合理,为了取得时间和资源的折中,选择第 4 种方案,即采用两个乘法器进行并行的运算,完成一次循环需要 3 个时钟周期。

1.2　模乘运算的设计

模乘运算是椭圆曲线点乘实现高速度的关键,为了提高模乘算法的效率,文献[2]采用了 Karatsuba-Ofman 算法,基于 FFT 思想对输入数据进行分解等。本文采用的是从右向左移位加算法,算法如下。

输入:不为零的最高位为 m 的多项式 $f(x)$;不为零的最高位至多为 $m-1$ 的多项式 $a(x)$ 、$b(x)$ 。

输出:$c(x)=(a(x)\times b(x)) \bmod f(x)$ 。

1)若 $a(0)=$ '1',则 $c\leftarrow b$;否则 $c\leftarrow 0$;

2) j 从 1 到 $m-1$ 循环。

① $b\leftarrow (b\times x) \bmod f(x)$;

② 若 $a(j)=$ '1',则 $c\leftarrow c+b$ 。

3)返回 c。

采用串行移位的方法,一个时钟完成一次移位操作,并进行相应的异或运算,则需要 m 个时钟周期,所需时间长。采用全并行的方式,首先并行地得到①项所有 b_i 值,然后采用并行方式完成②的运算。

由表达式:$b\leftarrow (b\times x) \bmod f(x)$,故其每一次的输出值,当移位后最高位的值为'1'时,进行取模运算,输出值为移位后的 b_i 与 $f(x)$ 异或;否则保持不变。基于这一思想,其运算过程如表 2 所示,由于 $b(191)$、$b_i(191)$ 都为'1',为了方便省略掉这一项。

表 2　b_i 值的求取过程

数据	191 位	190 位	…	11 位	10 位	9 位	…	3 位	2 位	1 位
b	$b(190)$	$b(189)$	…	$b(10)$	$b(9)$	$b(8)$	…	$b(2)$	$b(1)$	$b(0)$
b_1	$b(189)$	$b(188)$	…	$b(9)$	$b_1(9)$	$b(7)$	…	$b(1)$	$b(0)$	$b_1(0)$
b_2	$b(188)$	$b(187)$	…	$b_1(9)$	$b_2(9)$	$b(6)$	…	$b(0)$	$b_1(0)$	$b_2(0)$
…	…	…	…	…	…	…	…	…	…	…
b_9	$b(181)$	$b(180)$	…	$b_8(9)$	$b_9(9)$	$b_1(0)$	…	$b_7(0)$	$b_8(0)$	$b_9(0)$
b_{10}	$b(180)$	$b(179)$	…	$b_9(9)$	$b_{10}(9)$	$b_2(0)$	…	$b_8(0)$	$b_9(0)$	$b_{10}(0)$
…	…	…	…	…	…	…	…	…	…	…
b_{181}	$b(9)$	$b_1(9)$	…	$b_{180}(9)$	$b_{181}(9)$	$b_{173}(0)$	…	$b_{179}(0)$	$b_{180}(0)$	$b_{181}(0)$
b_{182}	$b_1(9)$	$b_2(9)$	…	$b_{181}(9)$	$b_{182}(9)$	$b_{174}(0)$	…	$b_{180}(0)$	$b_{181}(0)$	$b_{182}(0)$
b_{183}	$b_2(9)$	$b_3(9)$	…	$b_{182}(9)$	$b_{183}(9)$	$b_{175}(0)$	…	$b_{181}(0)$	$b_{182}(0)$	$b_{183}(0)$
…	…	…	…	…	…	…	…	…	…	…
b_{189}	$b_8(9)$	$b_9(9)$	…	$b_{188}(9)$	$b_{189}(9)$	$b_{181}(0)$	…	$b_{187}(0)$	$b_{188}(0)$	$b_{189}(0)$
b_{190}	$b_9(9)$	$b_{10}(9)$	…	$b_{189}(9)$	$b_{190}(9)$	$b_{182}(0)$	…	$b_{188}(0)$	$b_{189}(0)$	$b_{190}(0)$

观察表 2,190 个 191 位的 b_i,只与 b、$b_i(0)$、$b_i(190)$ 有关。

1)对于 $b_i(0)$,有:

$b(190)=$ '1',则 $b_1(0)=$ '1'; $b_1(190)=$ '0', $b_1(0)=$ '0';

$b_1(190)=$ '1', $b_2(0)=$ '1'; $b_2(190)=$ '0', $b_2(0)=$ '0'。

以此类推:

$b_i(190)=$ '1', 则 $b_{i+1}(0)=$ '1'; $b_i(190)=$ '0',则 $b_{i+1}(0)=$ '0'。

故 $b_1(0)\sim b_{181}(0)$ 仅与 $b(190)\sim b(9)$ 有关,可以一次并行地得到;对于 $b_{182}(0)\sim b_{190}(0)$ 以及 $b_{182}(9)\sim b_{190}(9)$,与 $b_1(9)\sim b_9(9)$ 有关,下面先讨论 $b_i(9)$ 的计算,然后再讨论 $b_{182}(0)\sim b_{190}(0)$ 的求取。

2) 对于 $b_i(9)$,有:

$b(190)=$ '1',则 $b_1(9)=\text{not } b(8)$;$b(190)=$ '0',则 $b_1(9)=b(8)$。

$b_1(190)=$ '1', 则 $b_2(9)=\text{not } b1(8)=\text{not } b(7)$; $b_1(190)=$ '0',则 $b_2(9)=b1(8)=b(7)$。

以此类推:

当 $i\leqslant 9$:$b_i(190)=$ '1', 则 $b_i(9)=\text{not } b(9-i)$; $b_i(190)=$ '0',则 $b_i(9)=b(9-i)$。

当 $i>9$:$b_i(190)=$ '1', 则 $b_i(9)=\text{not } b_{i-9+1}(0)$; $b_i(190)=$ '0',则 $b_i(9)=b_{i-9+1}(0)$。

故 $b_1(9)\sim b_{181}(9)$ 仅与 $b(190)\sim b(9)$ 有关,可以一次并行地得到;对于 $b_{182}(0)\sim b_{190}(0)$ 以及 $b_{182}(9)\sim b_{190}(9)$,与 $b_1(9)\sim b_9(9)$ 有关,而这 9 个值也只与 $b(190)\sim b(181)$ 有关,故 b_i 中所有的数据都只与输入数据 b 有关,故可以一次全并行地得到所有的 b_i 值。然后将所有 b_i 值进行异或运算,得到模乘运算的输出。

1.3　模逆运算的设计

模逆算法的实现,集中起来主要有基于扩展的欧几里得算法及其相关变形算法以及基于费马·小定理的模逆算法。前者是由反复的求度以及移位运算所组成,在时间开销上较大;后者是将模逆运算转换为求模幂运算。本文采用费马·小定理实现模逆算法,其中模乘、模平方运算采用并行方式予以实现。算法如下:

输入:$a(x)\in GF(2^m)$,不可约多项式 $f(x)$。

输出:$a^{-1}(x) \bmod f(x)=a^{2^m-2}(x) \bmod f(x)$。

为了减少模幂运算次数,采用费马·小定理实现可以采用多种加法链[8]。为了减少寄存器的使用,使每次输出值只与当前值或者前一个值有关,采用两个寄存器分别存储当前值或者前一个值,采用如下加法链。

$1\rightarrow 2\rightarrow 4\rightarrow 6\rightarrow 8\rightarrow 14\rightarrow 16\rightarrow 30\rightarrow 32\rightarrow 62\rightarrow 64\rightarrow 126\rightarrow 190$

并行模乘运算由于所占资源较多,考虑到布线延时、逻辑延时,整体延时较大;而模平方运算占用资源小,为了综合利用运算时间,使模乘运算与模平方运算的延迟基本相同,通过实验,模平方运算最高输出 16 级级联输出,即一个时钟周期完成 16 次幂的运算,故模平方运算可以输出 1 次幂、2 次幂、4 次幂、8 次幂以及 16 次幂,其结构如图 1 所示。

当模幂次数小于等于 16 时,通过一个时钟周期完成一次模幂运算;而在进行 32 次幂或者 64 次幂运算时,通过 2 个或者 4 个时钟周期完成模幂运算,其运算过程如表 3 所示。

完成模逆运算共需要 12 次模乘运算,21 次模幂运算。在部分步骤,模乘运算与模幂运算可以同时进行。

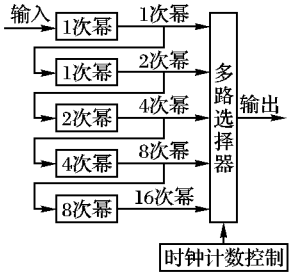

图 1　模逆运算中模幂运算结构

表 3 基于费马·小定理乘逆运算的运算过程

乘法次序	乘法器输入	乘法器输出	乘法次序	乘法器输入	乘法器输出
初始化		$\beta_0 = \alpha = \alpha^{2^1-1}$	7	$\beta_5^{2^{16}}\beta_6$	$\beta_7 = \alpha^{2^{30}-1}$
1	$\beta_0^{2^1}\beta_0$	$\beta_1 = \alpha^{2^2-1}$	8	$\beta_6^{2^{16}}\beta_6$	$\beta_8 = \alpha^{2^{32}-1}$
2	$\beta_1^{2^2}\beta_1$	$\beta_2 = \alpha^{2^4-1}$	9	$\beta_7^{2^{32}}\beta_8$	$\beta_9 = \alpha^{2^{62}-1}$
3	$\beta_1^{2^4}\beta_2$	$\beta_3 = \alpha^{2^6-1}$	10	$\beta_8^{2^{32}}\beta_8$	$\beta_{10} = \alpha^{2^{64}-1}$
4	$\beta_2^{2^4}\beta_2$	$\beta_4 = \alpha^{2^8-1}$	11	$\beta_9^{2^{64}}\beta_{10}$	$\beta_{11} = \alpha^{2^{126}-1}$
5	$\beta_3^{2^8}\beta_4$	$\beta_5 = \alpha^{2^{14}-1}$	12	$\beta_{11}^{2^{64}}\beta_{10}$	$\beta_{12} = \alpha^{2^{190}-1}$
6	$\beta_4^{2^8}\beta_4$	$\beta_6 = \alpha^{2^{16}-1}$			

2 椭圆曲线点乘的 FPGA 实现

采用蒙哥马利算法实现椭圆曲线点乘运算的系统结构如图 2 所示。在控制器的作用下，多路选择器、多路分配器输出对应的模乘运算单元、模平方运算单元所对应的输入数据，通过两个时钟周期完成一次点加运算，一个时钟周期完成一次倍加运算。因此完成主循环部分需要 $3 \times degk$（$degk$ 是 k 的度，k 的二进制表征中左边第一个不为零的二进制的位置）个时钟周期；然后进行求模逆运算，求模逆运算复用主循环的模乘运算单元，在两级寄存器以及多路选择器的配合下完成，如图 2 虚线框所示。

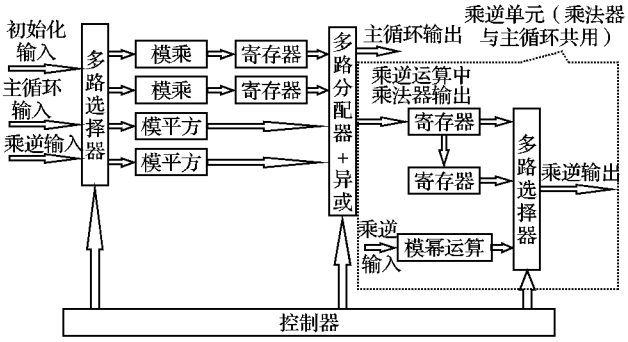

图 2 基于蒙哥马利算法的椭圆曲线点乘运算的系统结构

3 实验结果分析与性能比较

本文采用硬件描述语言 VHDL 作为设计输入，使用 ANSI X9.62-1998 的一组测试向量，并通过 modelsim 进行了功能仿真。目标器件为 Xilinx 公司的 virtex-5 器件族的 xc5vlx220T，通过综合与布局布线，在 40 MHz 完成了时序后仿真，输出结果完全正确。本文所采用的测试向量的 $degk$ 为 190，故完成一次椭圆曲线点乘运算共需要的时钟周期为 $190 \times 3 + 25 = 595$ 个。表 4 为本设计与相关工作的比较，通过与其他文献的比较，本设计的速度有了显著的提高。

表 4 本文与相关工作的比较

文献	域	器件	所需时间/μs
[2]	$GF(2^{192})$	XCV3200E	56.44
[3]	$GF(2^{256})$	EP2C35F672C6	5524.00
[4]	$GF(2^{163})$	Virtex4-LX200	36.00
[5]	$GF(2^{163})$	EP2S90F1508C	77.90
[6]	$GF(2^{163})$	XC2V2000	47.00
本文	$GF(2^{191})$	xc5vlx110T	14.90

4 结语

本文利用在 ECC 下的多项式基是稀疏多项式的特点，设计了全并行的模乘算法，并采用并行平方模块级联的方法，在模逆运算中采用费马·小定理并优化了迭代顺序与迭代次数。在点乘运算中将实现速度与资源之间取得折中，系统结构采用部分并行方式，完成一次点乘运算需要 14.9 μs，与以往的实现算法相比，大大地提高了运算速度。由于基于 ECC 的多项式基是稀疏多项式，尽管本文采用的是三项式基，但是对于其他的多项式基，并行模乘、模平方运算以及费马·小定理的优化同样适合；而对于其他位数的 F_2^m 域，仅仅是修改设计参数就可以实现，因此本设计具有一定的通用性。

参考文献：

[1] 刘连浩，申勇. 椭圆曲线密码体制中标量乘法的快速算法[J]. 计算机应用研究，2009，26(3)：1104－1108.

[2] SAQIB N A，RODRIGUEZ-HENRIQUEZ F，DIAZ-PEREZ A. A parallel architecture for fast computation of elliptic curve scalar multiplication over $GF(2^m)$[C] // Proceedings of the 18th International Parallel and Distributed Processing Symposium. Washington，DC：IEEE Computer Society，2004：144.

[3] 杨先文，杨洋，李峥. $GF(2^m)$ 域上 ECC 通用加速器设计与实现[J]. 计算机工程与设计，2008，29(12)：3026－3029.

[4] 陈婧，蒋俊洁，王石，等. 基于 FPGA 的高速椭圆曲线标量乘法结构[J]. 计算机研究与发展，2008，45(11)：1947－1954.

[5] 邹侯文，王峰，唐屹. 椭圆曲线点乘 IP 核的设计与实现[J]. 计算机应用，2006，26(9)：2131－2133.

[6] ANSARI B，HASAN M A. High-performance architecture of elliptic curve scalar multiplication[J]. IEEE Transactions on Computers，2008，57(11)：1443－1453.

[7] TAKAGI N，YOSHIKI J，TAGAKI K. A fast algorithm for multiplicative inversion in $GF(2^m)$ using normal basis[J]. IEEE Transactions on Computers，2001，50(5)：394－398.

[8] WU HUAPENG. Low compleixty bit-paralel finite field arithmetic using ploynominal basis[C]// Proceedings of the First International Workshop on Cryptographic Hardware and Embedded Systems. New York：ACM，1999：280－291.

（上接第 539 页）

[4] BRENNA L E，REED I S. Theory of adaptive radar[J]. IEEE Transactions on Aerospace and Electronic Systems，1973，9(2)：237－252.

[5] 王永良，彭应宁. 空时自适应信号处理[M]. 北京：清华大学出版社，2000：58－93.

[6] SARKAR T K，WANG H，PARK S，*et al*. A deterministic least-squares approach to space-time adaptive processing (STAP)[J]. IEEE Transactions on Antennas and Propagation，2001，49(1)：91－103.

[7] LAPIERRE F D，VERLY J G. New solutions to the problem of range dependence in bistatic STAP radars[C]// Proceedings of the 2003 IEEE Radar Conference. Washington，DC：IEEE Computer Society，2003：452－459.

[8] HAYWARD S D. Adaptive beam forming for rapidly moving arrays[C]// Proceedings of CIE International Conference of Radar. Washington，DC：IEEE Computer Society，1996：480－483.

[9] ZATMAN M. The properties of adaptive algorithms with time varying weights[C]// Proceedings of IEEE Sensor Array and Multichannel Signal Processing Workshop. Washington，DC：IEEE Computer Society，2000：82－86.

[10] 李明，廖桂生. 利用基于导数更新的双基机载雷达杂波距离依赖性补偿方法[J]. 电子与信息学报，2009，31(9)：2059－2064.

[11] WANG YONGLIANG，BAO ZHENG，LIAO GUISHENG. Three united configuration on adaptive spatial-temporal processing for airborne surveillance radar system[C]// International Conference on Signal Processing. Beijing：[s.n]，1993：381－386.