双有限域模乘和模逆算法及其硬件实现

陈光化[①]　朱景明[①]　刘　名[②]　曾为民[①]
[①](上海大学微电子中心　上海　200072)
[②](上海市电站自动化技术重点实验室　上海　200072)

摘　要：有限域上的模乘和模逆运算是椭圆曲线密码体系的两个核心运算。该文在 Blakley 算法的基础上提出一种 radix-4 快速双有限域模乘算法，该算法采用 Booth 编码技术将原算法的迭代次数减少一半，并利用符号估计技术简化约减操作；在扩展 Euclidean 求逆算法的基础上提出一种能够同时支持双有限域运算的高效模逆算法，该算法不仅避免了大整数比较操作，而且提高了算法在每次迭代过程中的移位效率。然后针对这两种算法特点设计出一种能够同时完成双有限域上模乘和模逆操作的统一硬件结构。实现结果表明：256 位的模乘和模逆统一硬件电路与同类型设计相比较，在电路面积没有增加的情况下，模乘运算速度提高 68%，模逆运算的速度也提高了 17.4%。

关键词：椭圆曲线密码体系；模乘；模逆；双有限域；Blakley 算法；扩展 Euclidean 算法

中图分类号：TN918.2; TN492　　**文献标识码**：A　　**文章编号**：1009-5896(2010)09-2095-06
DOI: 10.3724/SP.J.1146.2009.01258

Dual-field Modular Multiplication Algorithm and Modular Inversion Algorithm with VLSI Implementation

Chen Guang-hua[①]　Zhu Jing-ming[①]　Liu Ming[②]　Zeng Wei-min[①]
[①](*Research and Development Center*, *Shanghai University*, *Shanghai* 200072, *China*)
[②](*Shanghai Key Laboratory of Power Station Automation Technology*, *Shanghai University*, *Shanghai* 200072, *China*)

Abstract: Modular multiplication and modular inversion algorithms are the kernel of elliptic curve cryptography. In this paper, a radix-4 dual-filed modular multiplication algorithm which is based on Blakely's algorithm is proposed. Booth's algorithm is used to halve the iteration number of the Blakely's algorithm, and the sign estimation is employed to simplify the modular reduction operation. A dual-field modular inversion algorithm which is derived from extended Euclidean algorithm is presented to avoid magnitude comparison and shift efficiently. Furthermore, a unified hardware structure which can support dual-field modular multiplication and modular inversion operation is designed, and the result of the hardware implementation shows that the circuit can run 68% faster for modular multiplication and 17.4% faster for modular inversion without area increasing than others.

Key words: Elliptic Curve Cryptography (ECC); Modular multiplication; Modular inversion; Dual-field; Blakley's algorithm; Extended Euclidean algorithm.

1　引言

椭圆曲线密码体系(Elliptic Curve Cryptography，ECC)与传统的 RSA 密码体系相比具有加密强度高，密钥尺寸短，占用带宽少等诸多优点[1]，因此受到越来越来越多的关注。ECC 算法根据椭圆曲线选取的有限域不同又可分为基于素数域 GF(P)和二进制域 GF(2^m)两种。无论在哪个有限域，模乘和模逆运算都是作为其中的核心运算，同时也是相对复杂的运算操作，因此如何快速高效实现这两个运算对于 ECC 的应用意义重大。

适合素数域或者二进制域的模乘和模逆算法有很多，但大多数只是在其中的一个有限域上实现。然而随着 ECC 研究的不断深入及其应用的日益广泛，对于双有限域模乘和模逆算法的研究正在成为一种趋势和热点，近年来有些文献提出能够同时支持两种有限域的模乘算法[2,3]和模逆算法[4]，但是组合在一起的算法在运算速度上和单独的算法相比并不具备优势。由于在双有限域上设计算法面临着如下的难点：首先是素数域和二进制域的表现形式不同；其次是素数域和二进制域的加减法运算不同；再者素数域和二进制域的取模或者约减运算也不同。因此一个优秀的双有限域算法不仅要能够解决

2009-09-25 收到，2009-12-31 改回
上海大学“十一五”211 建设项目资助课题
通信作者：陈光化　chghua@shu.edu.cn

以上难题，完成两种有限域算法的有效合并，还应该具备良好的单个有限域的算法性能，Wang 等人[4]在扩展 Euclidean 算法的基础上提出的一种能够同时支持两种有限域的模逆算法，比较好地实现了这一目标。Ma 等人[5]针对扩展 Euclidean 算法中存在的大整数比较问题，提出了一种可以避免大整数比较的算法结构；Yan 等人[6]则针对其每次迭代中只能右移一位的情况提出了一种具有更高移位效率的改进算法。

本文在 Blakley 算法基础上提出一种改进的 radix-4 快速双有限域模乘算法；在扩展 Euclidean 求逆算法的基础上提出一种能够同时支持 GF(P)和 GF(2^m)两种有限域运算的模逆算法。而且，针对这两种算法特点设计出一种能够同时支持双有限域上模乘和模逆操作的统一硬件结构。在本文第 2 节介绍了 radix-4 快速双有限域模乘算法；第 3 节提出了基于扩展 Euclidean 算法的双有限域模逆算法；第 4 节设计了一种能够同时支持双有限域模乘和模逆运算的统一硬件结构；第 5 节给出了实现硬件电路的仿真以及性能比较；最后部分总结了全文。

2 双有限域模乘算法

由于 Blakley 算法中只有简单的加减和移位操作，并且可以直接输出最终结果，非常易于硬件实现，更重要的一点是该算法在两种有限域上的运算步骤非常相似，比较适合用来设计双有限域统一算法。但是该算法中存在普通加减法的进位传播延时和大数比较问题，一种改进方法是采用 CSA 加法器代替 CPA 加法器来消除进位传播延时，但同时由于要采用查找表技术会使得设计面积大幅增加，并且与模数 P 以及操作数相关的预处理也限制了其灵活性[7]；还有一种是采用冗余数法，通过对操作数进行 Booth 编码以减少迭代次数并利用冗余加法代替普通加法来消除进位传播延时，但同时冗余数和二进制数之间的转换代价会随着操作数位长的增加而增大。

本文在该算法基础上采用 Booth 算法[8]对乘数进行编码，按照从最低有效位到最高有效位的顺序每次处理操作数的两位，直接将原算法中循环迭代次数减少一半，并且减少了中间结果的累加次数；为了避免原算法每次约减前的大整数比较操作，本文采用一种符号估计技术[9]来提高约减的效率。只需要依据中间结果的最高有效位和次高有效位的值就可以进行下一步的约减操作，不仅提高了算法的运算速度，也简化了算法在实现过程中控制单元的设计；由于该算法在两种有限域上的运算步骤基本相同，因此只需将素数域的算法移植到二进制上即可完成双有限域模乘算法的设计，其伪代码如下：

算法 1 双有限域模乘算法见表 1。

表 1 双有限域模乘算法

输入 n 位二进制数 X,Y,P, 及域判断变量 field，且 $P \geqslant X$, $Y \geqslant 0$ 。

输出 $S = X*Y \bmod P$。

```
1   S=0, PPG=0, cj=0, y_1=y;
2   for(int i=0;i<n+2;i=i+2){
    2.1  y_2=2*y_1;
    2.2  Case(x_{i+1},x_i,c_j){
         "000": PPG=0, c_{j+1}=0;
         "001": PPG=y_1, c_{j+1}=0;
         "010": PPG=y_1, c_{j+1}=0;
         "011": PPG=y _2, c_{j+1}=0;
         "100": PPG=-y_2, c_{j+1}=1;
         "101": PPG=-y_1, c_{j+1}=1;
         "110": PPG=-y_1, c_{j+1}=1;
         "111": PPG=0, c_{j+1}=1;
         }
    2.3  c_j= c_{j+1};
    2.4  y_1=2*y_2;
    2.5  S=S+PPG;}
3   return(S mod P);
```

在步骤 2.1，2.4，2.5 采用符号估计技术来避免大数比较从而直接完成与模数“P”有关的加、减或者异或操作，如表 2 所示。其中“n+2”，“n+1”分别表示相应操作数的第 n+2 位和第 n+1 位的数值，“field”表示域选择变量，“field”为 0 表示二进制域，为 1 表示素数域。

表 2 采用符号估计技术的计算规则

n+2, n +1	field	y_2	y_1	S
00	0	y_2	y_1	S
00	1	y_2	y_1	S
01	0	$y\_2 \oplus P$	$y\_1 \oplus P$	$S \oplus P$
01	1	y_2–P	y_2–P	S–P
10	0	——	——	——
10	1	y_2+P	y_1+P	S+P
11	0	——	——	——
11	1	y_2+P	y_1+P	S+P

3 双有限域模逆算法

扩展 Euclidean 算法在素数域和二进制域上都用简单的加法、减法(异或)和移位操作代替了复杂

耗时的乘法和除法运算，非常适合硬件实现，更重要的是其在两种域的运算步骤上有很多相似之处，很容易将两种域上的算法统一起来，王健[10]基于该算法提出了一种能够同时支持双有限域的模逆算法。然而在其算法的每次迭代中都存在大整数的比较操作，不仅占用硬件资源，而且浪费时钟周期；其次，当操作数中出现多个连续的"0"时，该算法每次迭代中只能右移一位的操作就显得不够高效。为此，本文在扩展 Euclidean 算法的基础上提出了一种能够同时支持两种有限域的改进模逆算法。该算法用单分支结构代替原算法中双分支结构，使其在每次迭代中只处理某个固定操作数，从而避免了大整数比较操作；并且当操作数中有连续多个"0"时，用同时右移两位甚至三位的操作代替原先每次迭代只能右移一位来提高算法的运算效率，改进后的算法伪代码如下所示。

算法 2　双有限域模逆算法见表 3。

表 3 双有限域模逆算法

输入　n 位的二进制数 a，p 以及域判断变量 field，其中 p 为素数或者二进制既约多项，且 $0<a<P$。

输出　n 位的二进制 b，其中 b 满足 $ab\equiv 1 \bmod p$。

```
1   u ← p, v ← a, g1 ← 0, g2 ← 1.
    if (a is even) goto 3.
    else goto 4;
2   set v ← −v;
3   if (v/2 is even)   v ← v/4, g2 ← (g2/4) mod p.
    else   v ← v/2, g2 ← (g2/2) mod p.
    if (v is even) goto 3.
    else goto 4;
4   u1←v.
    if (field=1) v ←(u−v)/2, else v←(u⊕v)/2.
    u ← u1.
    b ← g2.
    if (g2<0) goto 6, else goto 5;
5   if (u=1) return b.
    g1_1 ← g2.
    if (field=1 and v>0)
        if (g1[0]=g2[0])    g2← (g1−g2)/2.
        else                g2← (g1−g2+p)/2.
    else if (field=1 and v≤0)
        if (g1[0]=g2[0])    g2← (g2−g1)/2.
        else                g2← (g2−g1+p)/2.
    else
        if (g1[0]=g2[0])    g2← (g1⊕g2)/2.
        else                g2←(g1⊕g2⊕p)/2.
    g1← g1_1.
    if (v<0) goto 2.
    else if (v is even ) goto 3.
    else goto 4;
6   g2 ← g2+p;
```

在上述算法的伪代码中，当"field"为 1 时表示素数域上的求逆操作，为 0 时表示二进制域上的求逆运算。符号"⊕"表示异或操作，步骤 3 中"$g_2\leftarrow(g_2/4) \bmod p$"在不同的条件下要进行相应的操作，"$g_2\leftarrow(g_2/2) \bmod p$"操作与上述情况类似，具体操作情况如表 4 所示，u_1 和 g_{1_1} 只是用来对算法进行解释，在实际硬件实现过程中是不需要的。

表 4 步骤 3 中的各种可能操作

field	conditions	operations
1	$g_2>0$ and $g_2[1:0]=00$	$g_2=(g_2>>2)$
	$g_2<0$ and $g_2[1:0]=00$	$g_2=(g_2>>2)+p$
	($g_2[1:0]=01$ and $p[1]=0$) or ($g_2[1:0]=11$ and $p[1]=1$)	$g_2=(g_2+3p)>>2$
	($g_2[1:0]=01$ and $p[1]=1$) or ($g_2[1:0]=11$ and $p[1]=0$)	$g_2=(g_2+p)>>2$
	the others	$g_2=(g_2+2p)>>2$
0	$g_2[1:0]=00$	$g_2=(g_2>>2)$
	($g_2[1:0]=01$ and $p[1]=0$) or ($g_2[1:0]=11$ and $p[1]=1$)	$g_2=(g_2\hat{}p)>>2$
	($g_2[1:0]=01$ and $p[1]=1$) or ($g_2[1:0]=11$ and $p[1]=0$)	$g_2=((g_2\hat{}3p))>>2$
	the others	$g_2=(g_2>>2)\hat{}(p>>1)$

4　硬件实现结构

为了实现在 GF(P)和 GF(2^m)两个有限域上硬件单元结构的统一，本文采用了一种被称作双有限域加/减法器的单元，其结构如图 1 所示[2]。将 n 个双有限域加/减法器单元进行级联就构成了一个完整的 n 位的双有限域加/减法器，通过这种单元搭建设计中的所有加/减法器，可以实现在两种有限域内相对统一的硬件结构，将额外的硬件开销降低至最小。

4.1 模乘单元设计

模乘算法完成的操作主要包括部分积的累加、中间结果的取模约减以及对输入操作数的 Booth 编码，其结构如图 2 所示。主要由 3 个部分组成，用来存放操作数、中间结果和最终结果的寄存器堆，用来执行加减法和编码操作的计算单元(1 个 Booth

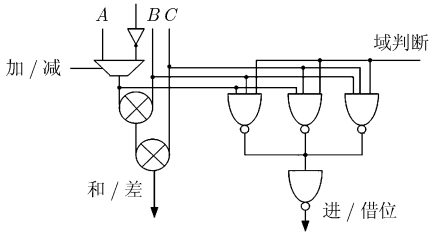

图 1 双有限域加/减法器单元结构

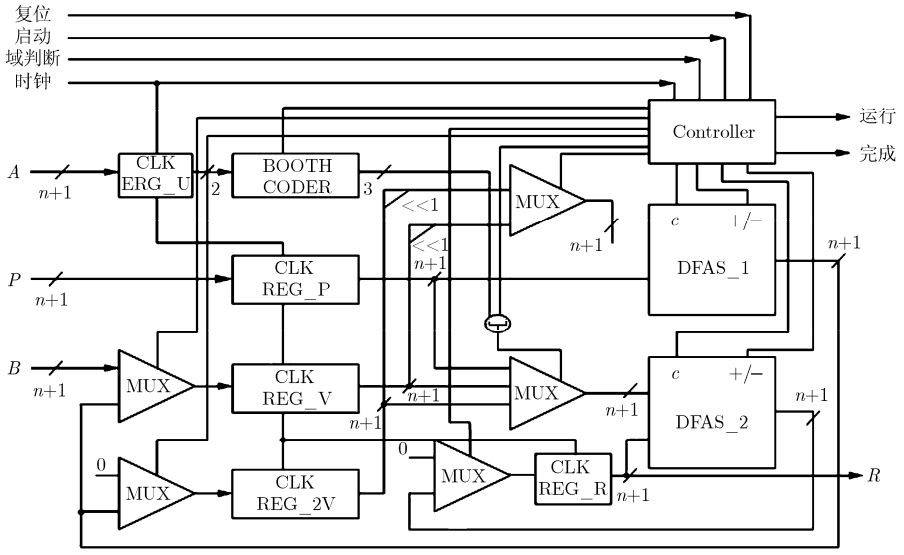

图 2 模乘单元的数据通路

编码器和 2 个双有限域加/减法器)，用来控制整个模乘运算流程的控制单元。

模乘运算开始前，乘数 A，被乘数 B 以及模数 P 分别写入到寄存器 REG_U，REG_V，REG_P 中，并将寄存器 REG_2V 和 REG_R 清零。BOOTH CODER 编码器在每次循环中按照从最低有效位到最高有效位的顺序处理乘数的两位，对乘数进行重编码操作；在每次迭代中被乘数都要完成两次移位操作，每次左移一位，两次移位操作完成后都是调用 DFAS_1 对移位结果进行符号估计约减操作；而 DFAS_2 主要是完成部分积的累加操作以及对部分积的符号估计约减操作。图中的 REG_U，REG_V，REG_2V，REG_P，REG_R 均为 256 位，而一般情况下运算时间与数据处理位宽 W 的取值成反比，W 的取值越大，运算越快，但是相应的芯片面积也会越大，本文综合考虑时间和面积确定本单元中的 W=32。

4.2 模逆单元设计

模逆单元也是主要由 3 个部分组成：一个是用来存储操作数以及中间和最终结果寄存器堆，包括两个双有限域加/减法器计算单元，控制整个模逆的计算流程控制单元。双有限域模逆单元的数据通路如图 3 所示。

模逆运算初始化时，寄存器 REG_U，REG_V，REG_G1，REG_G2，REG_P，REG_P3 分别被赋值操作数 A，模数 P，0，1，模数 P，$3P$ (模数 P

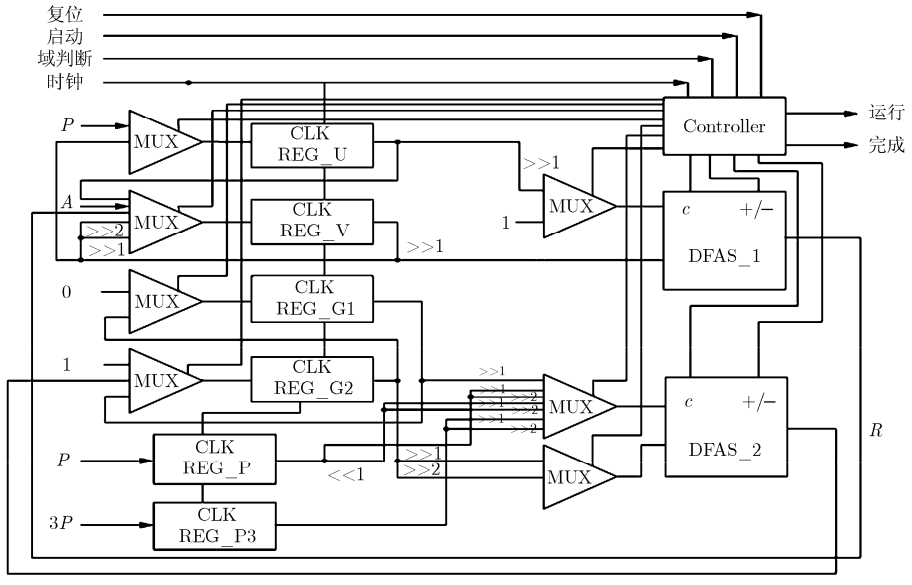

图 3 模逆单元的数据通路

的 3 倍)。双有限域加/减法器 DFAS_1 主要是用来完成寄存器 REG_U 和寄存器 REG_V 内容的减法操作(或者对应二进制域上的异或操作)和 REG_V 内容的取反操作；而 DFAS_2 主要用来完成寄存器 REG_G1，REG_G2 内容的减法操作(或者对应二进制域上的异或操作)和寄存器 REG_G2 内容的取模运算。图中的 REG_U，REG_V，REG_G1，REG_G2，REG_P，REG_P3 均为 256 位寄存器，综合考虑时间和面积因素我们将本单元的数据处理宽度 W 也设定为 32 位，最终的运算结果将存放在 REG_G1 中。

4.3 模乘和模逆单元统一架构设计

在采用上述的双有限域加/减法器后，模乘模块的计算单元主要包括两个双有限域加/减法器和一个 booth 编码器，而模逆模块的计算单元主要就是两个双有限域加/减法器，因此可以将这两个计算单元的硬件资源复用；另外，两个模块用来存放操作数以及中间结果和最终结果的寄存器堆也可以复用，因此模乘和模逆单元的统一架构也可以主要分为与模逆单元相似的 3 个部分，只是控制单元较原来要复杂一些，如图 4 所示。图中“MODE”为一个 2 位二进制数的域选择变量，“00”，“01”，“10”，“11”分别表示电路在进行二进制模乘运算、二进制域模逆运算、素数域模乘运算、素数域模逆运算；MI_ALU 表示该统一硬件电路的计算单元，包括两个双有限域加/减法器和一个 Booth 编码器；当电路进行模乘操作初始化时，操作数“X”，“Y”，“P”分别写到“REG_U”，“REG_V”，“REG_P”中，而

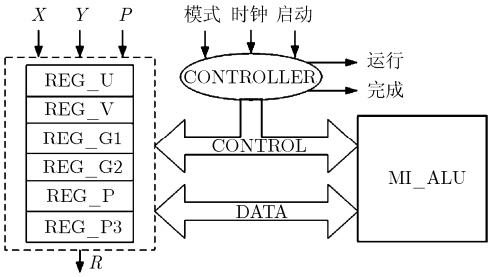

图 4 模乘和模逆单元的统一硬件框图

在进行模逆操作初始化时，除了注意将输入“Y”设定为 1 写入“REG_G2”中，其它如图 3 中的操作基本相同。

5 仿真与综合结果

二进制域上选用既约多项式 $P(Z)=Z^{233}+Z^{74}+1$，而在素数域上选用模数 $P=2^{224}-2^{96}+1$。模乘和模逆操作在素数域上的仿真波形如图 5 和图 6 所示，二进制域上的仿真波形也完全正确。

素数有限域上的模乘操作，输入的操作数 a=0x00000000_53f2fdee_00c4bb9b_d7f20a19_2e8f0608_35af165b_d919ae51_2365fc6d，b=0x00000000_4389169d_0d4330b1_07c1f8c9_f3578a75_234bd366_f4fa0999_c5972c0a，仿真结果如图 5 所示，与标准结果 r=0x0000013e _B11d a482_21ee4ada_d89f1cfc_dd853c0e_ b49 9a1ec_6bbcb421_ c99b8971 相一致。

素数有限域上的模逆操作，输入的操作数 a=0x00000000_1907f524_d5db74fc_8028d812_c7c5

图 5 GF(P)上模乘运算仿真波形

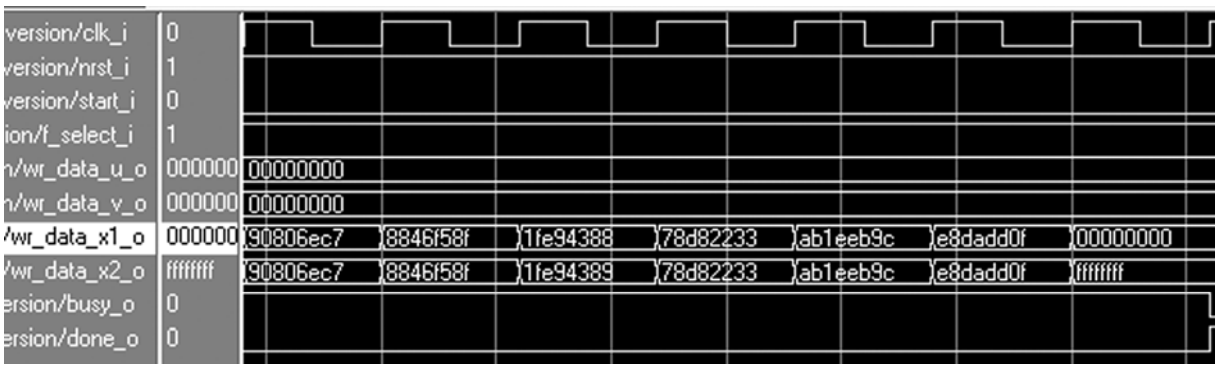

图 6 GF(P)上模逆运算仿真波形

63c5_d11ffd04_aae42961_aed6c8df，仿真结果如图 6 所示，与标准结果 b=0x00000000_e8dadd0f_ab1eeb9c_78d82233_1fe94388_8846f58f_90806ec7_f7fd7b7a 相一致。

本设计由 Verilog HDL 语言实现并采用 Synopsys 公司的 Design Compiler 进行综合，选用的是 SMIC 0.18 μm 的工艺库。DC 综合后的结果表明：256 位的模乘和模逆统一硬件电路与其它同类型的设计相比较在电路面积没有增加的情况下，模乘运算速度提高 68%，而模逆运算的速度也提高了 17.4%。具体信息参见表 5。

表 5 电路性能比较

	本设计	王健[10]	Savas [2]
时钟频率(MHz)	167	167	80
模乘运算速度(μs)	2.11	–	6.6
模逆运算速度(μs))	25.36	30.72	–
电路面积(等效门数)	51838	54628	–

6 结束语

本文对双有限域上椭圆曲线密码体系的两个核心运算的实现算法进行了研究。在 Blakley 算法的基础上提出一种改进的 Radix-4 快速双有限域模乘算法，该算法采用 Booth 算法对乘数进行编码以减少迭代次数，并采用符号估计技术避免原算法中大整数比较操作；在扩展 Euclidean 求逆算法的基础上提出一种双有限域高效模逆算法，该算法不仅避免了原算法中的大整数比较而且提高了操作数的移位效率。然后针对这两个算法特点设计出一种能够同时完成双有限域上模乘和模逆操作的统一硬件结构。实现结果表明：256 位的统一硬件电路与其它同类型设计相比较，电路面积没有增加，而运算速度显著提高。

参 考 文 献

[1] Hankerson D, Menezes A, and Vanstone S. Guide to Elliptic Curve Cryptography. New York: Springer Verlag New York Inc, 2004: 25–147.

[2] Savas E and Koc C K. A scalable and unified multiplier architecture for finite fields GF(P) and GF(2^m). Cryptographic Hardware and Embedded Systems(CHES) 2000, Worcester, MA, USA, Augst 17–18, 2000: 277–292.

[3] Chiou C W, Lee C Y, and Lin J M. Unified dual-field multiplier in GF(P) and GF(2k). *Information Security*, 2009, 3(2): 45–52.

[4] Wang Jian and Jiang An-ping. A high-speed dual field arithmetic unit and hardware implementation, ASICON'07, Guilin, China, Oct. 22–25, 2007: 213–216.

[5] Ma Shi-wei, Hao Yuan-ling, and Pan Zhong-qiao. Fast implementation for modular inversion and scalar multiplication in the elliptic curve cryptography, IITA '08, Beijing, China, Dec. 20–22, 2008: 488–492.

[6] Yan Xiao-dong and Li Shu-guo. Modified modular inversion algorithm for VLSI implementation, ASICON'07, Guilin, China, Oct. 22–25, 2007: 90–93.

[7] Shieh M D, Chen J H, and Lin W C. A new algorithm for high-speed modular multiplication design. *Circuits and Systems*, 2009, 56(9): 2009–2019.

[8] Hussin R, Shakaff A Y M, and Idris N. An efficient modified Booth multiplier architecture electronic design, ICED'08, Beijing, China, Dec. 1–3, 2008: 1–4.

[9] Nibouche O, Nibouche M, and Bouridane A. New iterative algorithm for modular multiplication, ICECS 2001, St. Julians. Malta, Sept. 2–5, 2001: 879–882.

[10] 王健. 椭圆曲线加密体制的双有限域算法及其硬件实现. [博士论文], 北京大学, 2008.
Wang Jian. A dual-field algorithm for elliptic curve cryptosystem and its hardware implementation. [Ph.D. dissertation], Peking University, 2008.

陈光化： 男，1972 年生，博士，副研究员，从事信号处理、集成电路设计等研发工作.