第34卷 第18期 计 算 机 工 程 2008年9月
Vol.34 No.18 Computer Engineering September 2008
·安全技术· 文章编号：1000—3428(2008)18—0202—02 文献标识码：A 中图分类号：TP309.7

有限域$GF(2^m)$模逆算法的改进与实现

杨先文，李 峥
(解放军信息工程大学电子技术学院，郑州 450004)

摘 要：在椭圆曲线密码体制中，有限域$GF(2^m)$中模逆运算是最重要的运算之一。在分析一种通用有限域$GF(2^m)$模逆算法的基础上，提出改进算法。改进算法减少了原算法快速实现时的缺点，能够有效地提高算法效率。基于FPGA分别实现了$GF(2^{83})$和$GF(2^{233})$中模逆算法模块，并与2种已有实现结果进行了对比。结果表明，选取有限域$GF(2^{83})$和$GF(2^{233})$时，改进算法效率提高率分别约为72.9%和59.5%。
关键词：椭圆曲线；有限域；模逆算法；快速实现

Improvement and Implementation of Modular Inversion Algorithm on Finite Field $GF(2^m)$

YANG Xian-wen, LI Zheng
(Institute of Electronic Technology, PLA Information Engineering University, Zhengzhou 450004)

【Abstract】 Modular inversion algorithm on $GF(2^m)$ is one of the most crucial algorithms in Elliptic Curve Cryptosystem(ECC). Under the analysis of a universal modular inversion algorithm, an improved algorithm is given in this paper. It can reduce many flaws of the universal algorithm, and has many good improvements for fast implementation. FPGA implementations for the improved algorithm on $GF(2^{83})$ and $GF(2^{233})$ are presented and compared with two existing implementations respectively. The result indicates that the improved algorithm has about 72.9% higher efficiency on $GF(2^{83})$ and about 59.5% higher efficiency on $GF(2^{233})$.
【Key words】 elliptic curve; finite field; modular inversion algorithm; fast implementation

1985年，N. Koblitz[1]和V. Miller[2]分别将椭圆曲线理论引入到密码学，从此，椭圆曲线密码体制(ECC)的研究逐渐成为一个活跃的密码学分支。ECC作为一种新兴公钥密码体制，同其他常用公钥密码体制相比，具有密钥短、实现速度快、安全性高等优点，在通信技术迅猛发展的今天有着广阔的应用前景。

1 $GF(2^m)$域中模逆算法改进及分析

有限域$GF(2^m)$中耗时较多的2个运算是模乘和模逆。通常，后者需要更多的计算时间和更复杂的电路，对于$GF(2^m)$域中元素求逆运算，常用的方法有以下2种：

(1)基于Fermat小定理的求逆方法[3]；

(2)基于扩展Euclidean算法及其变形的方法[4-5]。

前者通过把有限域求逆转化为一系列模乘和模平方运算，是一种非常简单的方法，但效率相对较低[6]。由于其实现规模较小，因此常常应用于资源受限环境。后者常用的算法有扩展欧几里德算法(Extended Euclidean Algorithm, EEA)[6]、近似求逆算法(Almost Inverse Algorithm, AIA)和改进的近似求逆算法(Modified Almost Inverse Algorithm, MAIA)，其中快速实现时大多数采用MAIA[4-5]，其算法描述如下。

算法1 改进的近似求逆算法(MAIA)

输入：$A\in GF(2^m)\backslash\{0\}$, F(x)
输出：$B=A^{-1}$
步骤：
1. B←1, C←0, U←A, V←F
2. While x|U do
2.1 U←U/x
2.2 If x|B then B←B/x else B←(B+F)/x
3. If U=1 then Return B
4. If deg(U)<deg(V) then U←→V, B←→C
5. U←U+V, B←B+C
6. Goto step 2

快速实现算法1时，总共需要4个$m+1$位寄存器B, C, U和V。其中，deg(U)和deg(V)分别表示U和V中域元素的次数。步骤2中对$x|U$的判断只需判断U的最低比特位是否为0，步骤2.1和步骤2.2只需作对应的寄存器逻辑右移，而步骤5只是域中加法。因此，算法实现的关键路径是步骤4中deg(U)和deg(V)的比较并进行相应的数据互换。文献[4]中给出了一种能在一个时钟周期内比较deg(U)和deg(V)的函数设计方案，但是当选取有限域$GF(2^m)$的m值较大时，同样存在时间延迟较长的缺点。因此，如何改进对U和V中域元素次数大小的比较成为提高算法效率的关键。

事实上，算法1的本质在于始终有下列关系式成立：

$$U\equiv B\cdot A \bmod F \quad (1)$$

$$V\equiv C\cdot A \bmod F \quad (2)$$

由式(1)可知，当U中的域元素经过若干次逻辑右移后域中加法等于域中单位元时，B中的域元素也就等于A^{-1}。对算法1可以通过以下方法进行改进：

(1)增加2个$\lceil \mathrm{lb}(m+1)\rceil$位寄存器$CU$和$CV$，用来监测$U$和$V$中域元素次数的变化；

作者简介：杨先文(1983－)，男，硕士研究生，主研方向：密码工程，安全芯片；李 峥，副教授
收稿日期：2007-10-18 **E-mail：**yxw200420042004@163.com

(2)对寄存器 V 和 C 增加与寄存器 U 和 B 对应的运算法则。

基于此结合算法 1，本文提出的二次改进的近似求逆算法描述如下。

算法 2　二次改进的近似求逆算法(MMAIA)

输入：$A\in GF(2^m)\backslash\{0\}$, F(x)

输出：$B=A^{-1}$

步骤：

1. B←1, C←0, U←A, V←F, CU←m, CV←m+1
2. While CU>1 and x|U do
   2.1 CU←CU-1
   2.2 U←U/x
   2.3 If x|B then B←B/x else B←(B+F)/x
3. If CU=1 then Return B
4. Elseif CV=1 then Return C
5. Elseif CU>CV then
   5.1 U←U+V, B←B+C
   5.2 Goto step 2
6. Else V←U+V, C←B+C
7. While CV>1 and x|V do
   7.1 CV←CV-1
   7.2 V←V/x
   7.3 If x|C then C←C/x else C←(C+F)/x
8. Goto step 3

算法 2 本质也是在于式(1)和式(2)关系成立。虽然描述看似复杂，但是其逻辑比算法 1 要简单，理由如下：寄存器 CU 和 CV 中的值并不是 U 和 V 中域元素的次数，而是监测其次数的变化，只需在步骤 1 初始化后根据对应条件减 1 计数，而比较 2 个 $\lceil \mathrm{lb}(m+1) \rceil$ 位寄存器 CU 和 CV 中数的大小是容易实现的。因此，大大简化了算法 1 中通过组合逻辑计算次数的操作；此外，对寄存器 V 和 C 增加与寄存器 U 和 B 对应的运算法则，就不再需要进行 $U \leftrightarrow V, B \leftrightarrow C$ 数据互换，避免了模块内部总线宽度较大时数据互换带来的延迟，而这样做的代价是输出结果时增加了对 CU 和 CV 值的判断，这也是容易实现的。因此，快速实现时采用算法 2 优点显著，而且当选取有限域 $GF(2^m)$ 的 m 值较大时优点更加突出。

2　$GF(2^m)$域中模逆算法模块设计

采用算法 2 设计模逆模块，其状态转换只需比较寄存器 CU 和 CV 中数值大小及判断寄存器 U 和 V 最低比特位，对应的状态机设计是高效的，模逆模块的状态转换如图 1 所示。

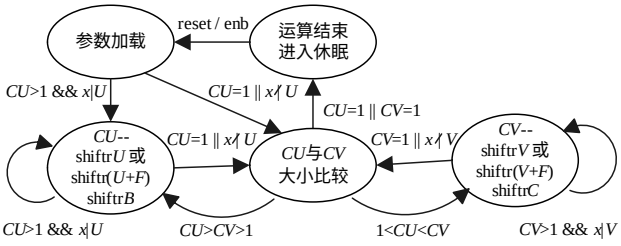

图 1　模逆模块的状态转换图

在模块内部，状态机 4 个状态分别为参数加载、操作寄存器 U 和 B、比较 CU 和 CV 大小、操作寄存器 V 和 C。当模块片选信号 enb 有效时，经过复位进入参数加载状态。在一个时钟周期内完成运算参数加载的同时，依据 $CU>1$ 和 $x|U$ 是否同时成立决定下一个时钟周期的状态输出。模块的主要工作状态是操作寄存器 U 和 B、比较 CU 和 CV 大小、操作寄存器 V 和 C，依据不同的判断条件完成对应的算法法则。至多 $2m-1$ 个时钟周期后，当检测到 CU 或 CV 为 1 时，则置结束信号 done 有效，同时通过一个选择器完成运算结果的输出，其实现原理如图 2 所示。

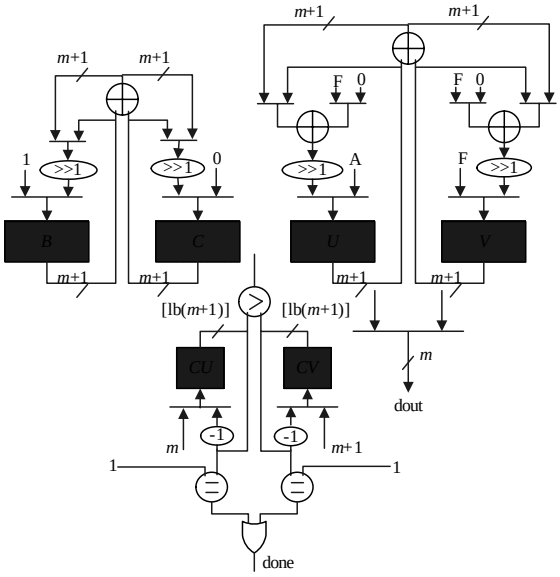

图 2　模逆模块实现原理

3　实现及结果分析

综上所述，采用 VHDL 语言作为设计工具，分别对有限域 $GF(2^{83})$, $GF(2^{233})$ 中的改进模逆算法进行了快速实现。对模块功能仿真及测试分为 2 步：(1)使用 ModelSim 软件平台对实现结果进行功能仿真，保证逻辑设计的正确性。(2)采用双端口 RAM 技术完成模块与微控制器的挂接，使用 Quartus II 软件平台，选择器件环境为 CycloneII EP2C35F672C6 完成模块的逻辑综合和时序仿真，并下载到 FPGA 开发环境进行测试。仿真及测试均得到正确的结果输出。

为了验证本文改进模逆算法的效率，使用 Leonardo Spectrum 软件平台，分别对有限域 $GF(2^{83})$, $GF(2^{233})$ 中的模逆算法实现进行了综合和时序分析，并分别与文献[4-5]中的实现结果进行的对比，分析及对比结果如表 1、表 2 所示。

表 1　$GF(2^{83})$中模逆实现分析及对比

算法	器件(Stratix)	规模/Les	频率/MHz	时间/μs
文献[4]	EP1S10F780C6	650	≤100	3.10
本文	EP1S10F780C6	817	197.4	0.84

表 2　$GF(2^{233})$中模逆实现分析及对比

算法	器件(Cyclone)	规模/Les	频率/MHz	时间/μs
文献[5]	EP1C20FC400C6	2 179	79.3	7.69
本文	EP1C20FC400C6	2 283	149.6	3.11

4　结束语

本文通过分析有限域 $GF(2^m)$ 中通用模逆算法之一(算法 1)，指出了硬件实现该算法时困难所在，并提出了一种改进形式(算法 2)。针对算法 2，选取有限域 $GF(2^{83})$, $GF(2^{233})$ 完成了模逆模块的设计与实现，仿真和测试结果表明设计与实现的正确性。经过综合和时序分析，与文献[4-5]中采用算法一的模逆实现结果进行了对比，结果表明在实现规模相当的情况下，采用本文提出的改进算法能够有效地减小关键路径延迟。

(下转第 209 页)