公钥密码系统中底层运算的硬件加速

摘 要

随着信息产业的迅速发展，人们对信息和信息技术的需要不断增加，信息安全也显得越来越重要。密码技术则是保障信息安全的一个重要手段。而公钥密码是现代密码学的核心，是目前解决身份鉴别与密钥交换的主要技术手段。

公钥密码算法的关键操作是有限域运算。为了保证安全，一般为大整数操作，属于计算密集型运算，效率较差。出于运算性能考虑，可以采用硬件实现公钥密码算法。本文正是围绕使用硬件提高公钥算法效率这一课题展开研究。

首先，介绍了常见公钥算法的流程和相关的数学原理，为后文思想的介绍奠定基础，使要解决的问题关键集中在底层有限域运算。然后详细描述了两种新的硬件二元域求逆方案和二元域乘法器的参数优化，从而构建了完整的公钥密码底层运算体系。

在分析了一些经典的求逆算法的基础上选择殆逆算法进行优化，利用其分阶段的特点实现了较低的传输延迟。又通过对殆逆过程的分析，发现度数变化的前后相关性规律，并利用相关性快速得到当前度数，大幅度压缩一次求逆需要的时钟周期数。而后，方案还通过分解合并殆逆算法步骤进一步压缩时钟数。实验表明，此方案由于时钟周期数和延迟两方面的优化，效率赶上甚至超过国际上的一些经典算法。

接下来，针对前一求逆方案在逻辑门延迟方面的不足，进一步改进，提出了一种双向移位结构，用反向移位取代原算法中延迟最严重的动态搜索求度数，将关键路径延迟从上一方案 $\log_2 m$ 数量级缩减到 $\log_2(\log_2 m)$数量级，提高幅度较大。分析表明，二元域双向移位殆逆模块的理论综合性能优良；实验结果进一步证明其实际执行效率也令人满意，相对于国内外很多优秀经典算法均有优势。

有限域乘法速度天然地高于求逆，但其效率也不尽如人意。本文分析了一些优秀的有限域乘法器之后，着眼于正规基，实现了 $GF(2^{233})$上最优正规基串-并乘法器，并对逻辑门网络的流水线级数和乘法器并行度两个参数进行了实验测量，根据结果进行了优化。同时，本文论证了殆逆算法第二阶段 k 取值的规律，利用此规律提出了一种查表提高效率的可行方案。

关键词：公钥密码体制;硬件加速;二元域;求逆;乘法

Hardware Acceleration of Fundamental Arithmetic in Public-Key Cryptography

ABSTRACT

With the rapid development of information industry, people depend on information technology increasingly. As a result, information security is getting more and more important. Cryptography is an important means to ensure network security. Public-Key Cryptography is the core of modern cryptography, and the primary means for solving authentication and keying exchange.

The key operation of Public-Key Cryptography arithmetic is finite field arithmetic, which is the arithmetic-intensive and inefficient operation of big integer. Public key Cryptographic algorithm is implemented by hardware for performance reason. How to improve efficiency of Public key Cryptographic algorithm by hardware are studied in this thesis.

Firstly, common Public key algorithm and related mathematical principles is introduced, which lay the foundation for the idea after text and make the key questions to resolve concentrated in underlying finite arithmetic. Then, two new hardware-inverses in binary finite fields and parameter optimization on multiplier on Binary Finite Fields are described in detail. That means a complete system for Public key Cryptographic underlying arithmetic is constructed.

On the base of analyzing some classic inverse algorithms, almost inverse is selected to be improved, because its grading characteristic can achieves lower transmission delay. The low of relevance between adjacent degrees is discovered through analysis on inverse process. Current degree is obtained quickly using relevance low, which can compress clock cycles needed in an inverse. Clock is further compressed through decomposing and recombining the steps of almost inverse algorithm. The result of experiment shows that the efficiency of new module can catch up with or even be superior to many classic algorithms because of optimization on clock and delay.

Next, to the deficiency of previous inverse scheme in logic gate delay, bidirectional shift structure is proposed in this thesis. When solving new degree, dynamic search whose delay is serious is replaced with reverse shift. Then Critical path delay is reduced to $O(\log_2(\log_2 m))$ from $O(\log_2 m)$. Analysis shows that the theoretical comprehensive properties of bidirectional-shift almost inverse module in binary fields are good. Experimental results prove that the efficiency in actual

execution is satisfactory, and has comparative advantage with most international and domestic classic algorithms.

The speed of multiplication in finite field is higher than inverse naturally, but its efficiency is not wholly satisfactory, too. After the analysis on some excellent multipliers in finite field, parallel-series multiplier on optimal normal basis in $GF(2^{233})$ is realized, pipeline stages of logic gates network and the degree of multiplier's parallelism are measured in experimentation and these two parameters are optimized in this thesis. At the same time, the law of k value in second stage of almost inverse algorithm is demonstrated in this thesis. Base on the low, a feasible scheme to raise efficiency useing checking list is proposed.

Keywords: Public-Key Cryptography; Hardware Acceleration; Binary Finite Fields; Inverse; Multiplication

插图清单

表格清单

独 创 性 声 明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标志和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 合肥工业大学 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签字：宋激龙 签字日期：2009 年 4 月 15 日

学位论文版权使用授权书

本学位论文作者完全了解 合肥工业大学 有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅或借阅。本人授权 合肥工业大学 可以将学位论文的全部或部分论文内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后适用本授权书）

学位论文作者签名：宋激龙　　导师签名：

签字日期：2009 年 4 月 15 日　　签字日期：2009 年 4 月 15 日

学位论文作者毕业后去向：

工作单位：　　电话：

通讯地址：　　邮编：

致　谢

两年半的研究生生活即将结束，回想这一阶段的经历，我心潮起伏。从开始进入课题到论文完成，身边的师长、同学、朋友和远方的亲人给了我无私的帮助，在这里我要表达一下我的谢意。

首先，衷心感谢我的导师梁华国教授，从本科毕业设计到研究生期间的研究工作，梁老师始终给与我细心的指导和不懈的支持，他渊博的知识，开阔的思路，严谨的治学精神，客观包容的学术态度，诲人不倦的教育情怀，深深地感染着我，不断地激励着我，使我受益匪浅。他还在生活中给与我不断的关心和鼓励，让我终身难忘。在此我谨向导师表示最诚挚的敬意和由衷的感谢！

衷心感谢欧阳一鸣副教授。欧阳老师的点拨给我的研究工作提供了很多灵感。感谢系统结构所的易茂祥副教授、陈田老师、黄正峰、李扬、刘军等老师，他们为我的学习和科研提供了许多无私的支持和帮助。

感谢孙科、王保青、朱兵、董少周、张岚、刘娟等研究生同学在学习和生活中对我的帮助，和你们讨论开拓了我的思路，和你们在一起总是充满了快乐。感谢覃敏东和毛蔚，你们在我的研究方向上给过我很大的帮助，和你们合作是非常惬意的。同时也感谢实验室的其他师兄师弟给我的友谊和温暖。

我还要感谢我的父母，困难时你们给我支持，脆弱时你们给我鼓励，快乐也愿与你们分享。

感谢单国华同学在学习和生活上给我的支持，让我眼前的困难不再不可逾越。

最后，衷心感谢合肥工业大学计算机与信息学院的各位老师和院系领导对我的帮助和支持，感谢为评阅论文而付出辛勤劳动的各位专家学者。

研究生的生活即将结束，新的征程即将开始，但这一段段亲情、友情和恩情将让我铭记一生。

作者：宋灏龙

2009 年 4 月

第一章 绪 论

1.1 研究的背景

计算机网络技术的飞速发展无疑给社会、企业乃至个人带来了前所未有的便利，所有这一切正是得益于互联网络的开放性和匿名性特征。然而，开放性和匿名性也决定了互联网不可避免地存在信息安全隐患。人们通过计算机网络传输的数据中包含了各种机密信息(例如军事机密信息、商业机密信息、政府机密信息等)。由于目前的网络系统缺乏足够的安全性，不能有效防止通过网络传输的信息被非法窃取和修改，因而极大地限制了计算机网络技术在日常工作中的应用。由于互联网是一个面向大众的开放系统，对于信息的保密和系统的安全性考虑得并不完备，由此引起网络安全问题日益严重，每年损失达好几十亿美元[1]。

信息安全的挑战为信息加密技术的发展提供了机遇，信息安全的核心技术——密码学这一曾经神秘的科学在信息膨胀的 e 时代唤发出勃勃生机，成为学术研究和商业应用的热点。毋庸置疑，密码学将在互联网和通信系统中起着越来越重要的作用，得到世界各国政府、军队、大学研究机构和商业集团等的高度重视。

算法的实现是构建安全基础设施的重要任务之一，其中硬件实现公钥算法在性能上有着巨大的优势，在国防、政府、企业等对大量数据有较高安全需求的场合也有着广泛的应用前景。用硬件可以设计出针对不同应用环境的专用密码芯片，这种方法不但可以在算法结构方面优化设计，而且能够在电路结构级甚至器件级进行优化设计，得到非常好的结果。随着硬件技术的飞速发展，已经有智能卡、IC 卡和电子钥匙 Ukey 等出现在电子商务的硬件大家庭中。

1.1.1 密码学概况

密码学是通过数据加密、签名和散列等技术，可以在一定程度上提高数据传输的安全性，保证传输数据的完整性。

信息主要是指存放在信息系统中的程序和数据，信息安全则是这些程序和数据在被存储、处理、执行和传输中的安全，主要包括以下几个方面：

1. 保密性，保密性指保密信息使之不被授权的实体所获取，即防止信息在非授权情况下的泄漏；
2. 完整性，信息的完整性指信息在未被授权的情况下，保持不被篡改，不被破坏和不丢失的特点；
3. 抗抵赖性，抗抵赖是指信息的行为人要对自己的行为负责，不能抵赖自己曾有过的行为，如否认自己发出过或收到过对方的信息；
4. 可靠性，可靠性指系统在运行过程中，抗干扰(包括人为、机器及网络

故障)和保持正常工作的能力。即保持工作的连续性和正确性的能力。

密码技术是保障信息安全的核心技术。密码学的研究与应用已有几千年的历史，但作为一门科学是20世纪50年代才开始的。目前密码技术已经从外交和军事领域走向公开，并且已经发展成为一门结合数学、计算机科学、电子与通信、微电子等技术的交叉学科，使用密码技术不仅可以保证信息的机密性，而且可以保证信息的完整性和确定性，防止信息被篡改、伪造和假冒。

Shannon于1949年提出了保密通信系统模型[2]，见图1-1。在图1-1中，明文x被发送之前，发送者和接收者之间使用的密钥k需要事先商定，这个密钥经商定后必须严加保密。一般说来，系统的保密性不依赖于加密体制或算法的保密(Kerckhof原则)，而只依赖于密钥。也就是说加密和解密算法是公开的，密码分析者可以知道算法与密文，但由于他并不知道密钥，因此仍难于将密文还原为明文。

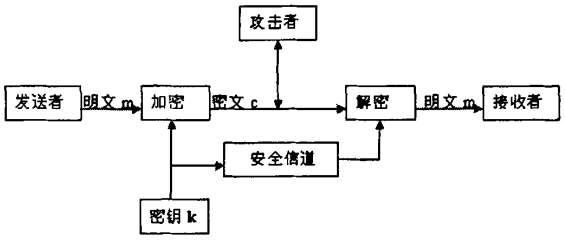


图1-1 Shannon保密通信模型

加密算法和解密算法所使用的参数称为加密密钥和解密密钥，两者可以相同，也可以不同。根据密钥的特点可将密码算法分为两类：对称(私钥)密码算法和公钥密码算法[3]。

在对称加密系统中，加密和解密采用相同的密钥。因为加解密密钥相同，需要通信的双方必须选择和保存他们共同的密钥，各方必须信任对方不会将密钥泄密出去，这样才可以实现数据的机密性和完整性[4]。

公钥密码算法又称非对称密钥算法、双钥密码算法。在公钥密码算法中，加密密钥不同于解密密钥，加密密钥可以公之于众，代表个人身份;解密密钥只有解密人自己知道，用来实现安全通信，必须保密。加密密钥和解密密钥分别称为公开密钥(简称公钥)和秘密密钥(简称私钥)。

对称密码算法的优点是加解密速度快，其缺点是:密钥的分发和管理非常复杂，不适何在大型网络中应用；无法对信息发送人的身份进行认证并检验信息的完整性，因而不能用于数字签名。公钥密码算法很好地解决了这两方面的问题，并正在产生许多新的思想和方案[4]。

公钥密码算法与对称密码算法相比有其不可取代的优势，然而它的运算量却十分浩大。在公钥密码算法中所需要的计算量比一般的加密算法如DES计算量多几个幂的数量级。因此，提高公钥算法的执行效率也一直是一项重要课题，硬件加速是一条直接有效的途径。

1.2 研究的目的和意义

信息安全已成为人们在信息空间中生存与发展的重要保证条件。因此，密码学和信息安全技术在最近二十多年来，越来越受到人们的重视，特别是“911”事件以来，信息安全业已成为各国政府和有关部门、企业、机构的重要议事内容。

随着在有限域上的离散对数问题和因子分解上不断进步、计算机运算速度的提高和计算机网络的发展，为了达到安全要求，大多数公钥密码体制的密钥也越来越大，传统的软件加/解密方式已经难以满足要求。要满足上百兆到上千兆的加密速率，必须采用硬件实现，即用微电子技术将加密软件算法转换成硬件实现的 FPGA 或 ASIC 芯片。与软件加密相比，硬件加密具有加密速度快、性能好等优势，并且便于物理保护，安全性好[5]。本文将硬件加密系统底层算法模块优化实现，是从根本上利用硬件特点，开发硬件潜能，提高系统效率。模块用于硬件密码加速引擎或安全处理器等将能够大大提高椭圆曲线等公钥密码体制效率，应用前景比较乐观。

我国的信息网络安全起步较晚，安全防护能力处于发展的初级阶段，与发达国家有较大的差距。当前，国内许多信息网络应用系统尚处于不设防状态，存在着很大的风险性和危险性；有些重要的网络应用系统使用的安全设备都是从国外直接引进的，难以保证安全利用和有效监控，密码技术特别是加密技术是信息安全技术中的核心技术，国家关键基础设施中不可能引进或采用别人的加密技术，只能自主开发。目前我国在密码技术的应用水平方面与国外还有一定的差距。国外的密码技术必将对我们有一定的冲击力，特别是在加入 WTO 组织后这种冲击力只会有增无减[6]。因而我们必须自主的开发我们自己的加密解密芯片，从而保证我们信息的安全性。因此，研究本课题有非常迫切且重要的现实意义。

1.3 研究现状

自从公钥密码的思想提出以来，国内外密码学家设计了许多优秀的公钥密码体制，其中著名的体制包括：1978 年 Rivest 等提出的 RSA 公钥体制[7]；1978 年 Merkle 与 Hellman 提出的基于背包问题的 MH 背包体制[8]，1979 年 Rabin 提出的 Rabin 体制[9]，1985 年 ElGamal 提出的 ElGamal 公钥体制[10]，1987 年 Koblitz 和 Miller 提出椭圆曲线密码公钥体制[11]，以及基于代理编码理论的 MeEliece 体制[12]和基于有限自动机理论的公钥密码体制[13]等等。公钥密码除了公钥密码体制之外，还包括数字签名技术[14]。著名的数字签名有 RSA 签名、Rabin 签名、ElGamal 签名、Schnorr 签名[15]和美国国家数字签名标准 DSS[16]。由于数字签名可以提供信息的鉴别性、完整性和不可否认性，因此，随着实际应用的需要，特殊的数字签名也被广泛的提出。主要包括：代理签名[17]、盲

签名[18]、可验证的加密签名[19]、不可否认签名[20]、前向安全签名[21]、密钥隔离签名[22]、在线/离线签名[23]、门限签名[24]、聚合签名[25]、环签名[26]、指定验证者签名[27]、确认者签名[28]，以及它们各种变型签名等等[29]。

出于运算性能上的考虑，公钥密码算法往往以硬件实现。另外，从安全性角度看，某些应用也要求采用硬件方式实现公钥密码算法，并且保证私钥不出密码芯片(即私钥为外部不可获取)。

目前，高端公钥密码专用芯片方面的研究非常广泛[30] [31] [32]，取得了较多的研究成果。2001 年，德国的 Infineon 公司推出了一款安全芯片，产品型号为 SLE66C42P。它具有数据加密标准(DES)和三重 DES 算法(3DES)的对称加/解密功能和实现椭圆曲线数字签名算法(ECDSA)的功能。它只对定义在 $GF(2^m)$，m = 192 上的椭圆曲线有效。在 5MHz 和 10MHz 的工作频率下，完成一次签名的产生所能需要的时间分别是 285ms 和 142ms，完成一次签名验证所需要时间是 540 ms 和 270ms。

2001 年，Motorola 公司推出了一款多功能安全处理器，型号为 MPC180。主要是为了实现网络协议安全(IPsec)协议而为客户端用户所设计。该芯片具有实现 DES、3DES、RC4、DM4、MD5、SHA-1、RSA、ECC 和随机数产生等算法功能。关于 ECC 算法功能，MPC180 芯片可以同时兼容素数域曲线和特征 2 域曲线。对素数域情况，只要求定义曲线的素数域 GF(p)中的素数 p 是一个规模在 64 比特到 512 比特之间的素数即可。对特征 2 域情况，也只要求定义曲线的有限域 $GF(2^m)$中的 m 是一个 64 到 512 之间的数即可。但芯片没有提供任何完整的密码算法或密码协议的实现，只提供了标量乘法的计算功能和计算椭圆曲线上点加 P+Q 和计算倍点 2P 的功能。

Athena 公司的 Teiafire 系列产品如 Exp-A1200[33]与 ChipSign 公司的 CS1015/Rubicon[34]代表了当前高端公钥密码专用芯片的较高水平。而国内目前比较成熟的设计为中兴公司研制的 SSX04 芯片[35]。国内在高端公钥密码专用芯片方面的研究与国外先进水平存在较大差距。

采用硬件方式实现 ECC 的报道最早见于文献[36]。文献[36]构造了一块专门用于执行有限域 $GF(2^{155})$上乘法运算的 VLSI 芯片，然后再利用一个高效的可编程控制器实现了基于 $GF(2^{155})$上的 ECC。利用这一芯片，加密速度大致可以达到 40kb/s。换算为签名速度大致是 130 次。2000 年文献[37]介绍了对定义在 $GF(2^{163})$上的 Koblitz 曲线采用现场可编程门阵列(FPGA)方式实现和采用 ASIC 方式实现的仿真结果。在 0.25μm 的 CMOS 工艺下，ASIC 芯片的规模是 16.5 万门电路，主频可以达到 66MHz。利用这一芯片，两种曲线的签名速度分别可以达到每秒 900 和 1500 次以上。这一速度对服务器应用是能够满足的。2000 年文献[38]还介绍了对定义在 $GF(2^{167})$上的曲线利用 Xilinx XCV400E FPGA 芯片的实现结果。这一实现中，芯片的最高工作频率可达 76.7 MHz。这时，完成

一次多倍点运算只需要 0.21ms。从而每秒可以完成 4762 次多倍点运算。国内的中兴公司也设计了一款 ECC 芯片——THECC/233-100。芯片主要包含随机数产生模块、大整数模运算模块和椭圆曲线运算模块。芯片最高工作频率 125 MHz，最高运算速度每秒完成 5000 次签名产生，总体性能较优。上海微科集成电路有限公司于 2004 年 12 月 15 宣布开发成功的 RSA/ECC 二合一密码算法协处理器芯片。该芯片最多可以完成 256 比特的 ECC 运算，每秒至少可以完成 100 次 ECC 点乘运算。

在 RSA，DH，DSA，ElGamal 等公钥密码算法中，底层关键操作均为大整数模逆和模幂乘操作，而在 ECC 密码算法中，关键操作为椭圆曲线标量乘法操作，也可分解为模逆和模幂乘。这些关键操作实现较复杂、涉及大量算术运算，十分耗时，这类操作通常为公钥密码算法运算的性能瓶颈。针对底层操作改进的研究也是热点之一。

模逆运算由于复杂度高，难度大，研究进展慢，成果不多。求逆方法分扩展欧几里德算法和基于费尔马小定理的方法两大分支。

费尔马小定理方法将除法转化成大量乘法，结构简单。Itoh[39]指出，完成一次求逆运算所需乘法次数最少为 $i(m) = [\log_2(m - 1)] + w(m - 1) - 1$ 次，其中 $w(m - 1)$表示域 $GF(2^m)$中二进制数表示中 1 的个数。在此基础上，Sand ho oh 和 Chang Han Kim[40]提出优化求逆算法 OIA (Optimal Inverse Algorithm)，该算法求逆主要用到乘法和平方运算，乘法次数和符合 Itoh 的论证，达到理论的极限。从而费尔马小定理的优化主要集中在乘法器的优化上。除此之外，一种新的脉动阵列结构正慢慢成为研究的热点，文献[41]使用这种结构实现费尔马小定理求逆。这种结构将算法中的重复操作展开为多级相同模块组成的阵列，数据逐级扩散至各模块同时被施以相应变换，当数据流出阵列时就是计算结果。这种结构规则，适合 ASIC 实现，且效率非常高，但硬件开销比较可观。

欧几里德算法这一分支形成的算法稍多，但多为在扩展欧几里德算法基础上作微小改动。扩展欧几里德算法最直接的改进就是避免了除法的二进制求逆方法[42]，其效率其实已经非常不错，之后的改进效率提高程度远不及它。殆逆算法[43]是二进制逆的一个变形，收敛更快，本文求逆部分主要针对此算法改进。Montgomery 求逆[42]和 Montgomery 乘法思想类似，用用低成本的 $zR^{-1} \bmod p$ 运算代替复杂的 $z \bmod p$，de Dormale，G.M.Bulens，P.Quisquater.J.-J[44]和 Cilardo.A，Mazzeo.A，Romano.L，Saggese.G.P.[45]在 2004 年分别给出了 FPGA 上 Montgomery 求逆的方案。求逆还发展出一种同时对多个元素运算的同时求逆方法，在特定情况下可以提高效率。Baeily 和 Paar[46][47]对其提出的最优扩域，利用 Forbeinus 映射得到更有效的求逆算法。ARAKI K，FUJITA I 和 MORISUE M 在 1989 年较早提出了一种基于欧几里德算法的硬件求逆模块，各方面性能尚有不足之处[48]。BRUNNER H，CURIGER A 和 HOFSTETTER M

提出的硬件优化欧几里德方案[49]综合性能良好。国内在欧几里德算法硬件移植方面优秀成果不是很多。袁丹寿，戎蒙恬在06年在文献[49]方案基础上改进了适应性，给出一种硬件求逆方案[50]。鲍可进，宋永刚也在06年在FPGA上实现了求逆[51]。王健，蒋安平，盛世敏在2007年实现了一种同时用于素域和二元域的求逆模块[52]，同时性能较优。欧几里德算法展开后也可用脉动阵列实现。其中文献[53]较早引入脉动阵列改进欧几里德算法，达到了很高的效率。Zhiyuan Yan，Dilip V. Sarwate等人在脉动阵列实现欧几里德算法方面作了很多工作：文献[54]着重降低各子模块和关键路径延迟，提高效率。文献[55]提出了一种二维脉动结构，降低了脉动阵列的开销。而文献[54]则全面考虑开销和效率，综合二维结构和低延迟技巧。总之它们的共同特点仍是效率高，开销大。

由于有限域加法和乘法运算易于硬件实现，因此，设计时间和空间复杂性低的乘法器成为研究热点[56]，并已经包括在IEEE和NIST标准中[57][16]。一般地，根据有限域元素不同的表示方法，乘法器可分为正规基乘法器、多项式基乘法器和对偶基乘法器;根据具体实现的方式，也可分为串行乘法器和并行乘法器[58]。

正规基乘法器最大的优点是平方运算只须一个循环移位即可完成，从而得到更高效的乘法运算[56]。1986年，Massey-Omura [59]提出了第一个正规基乘法器，简称为MO乘法器。1993年，Hasan等人[60]将不可约多项式限制为全1多项式(All-one Polynomial，AOP，系数全为1的多项式)，提出一种新颖的乘法器结构，大大降低了并行MO乘法器的复杂性。对由AOP产生的同类型有限域，C.K.Koc和B.sunar[61]将多项式基乘法器推广到正规基乘法器，提出一种新型并行正规基乘法器。同时，Mulin等人[62]给出了正规基乘法器的复杂度下限，并将能够达到该下限的正规基定义为最优正规基(ONB)。他们还定义了两种类型的正规基，即I型最优正规基和II型最优正规基。

2001年，B.sunar和C.K.Koc[71]提出一个II型最优正规基并行乘法器，该乘法器所需XG的个数比并行MO乘法器少25%；2002年，AR.Masoleh和M.A.Hasan[58]提出了一种简化的冗余MO并行乘法器，该乘法器可用于任意正规基和没有任何特殊限制的有限域，而且其复杂性低于并行MO乘法器。特别地，该乘法器的空间复杂度几乎是其它乘法器的一半。A.RMasofeh和M.A.Hasan，B.Sunar又对前面的工作做了进一步的发展，得到时间和空间复杂性都低于MO乘法器的几种正规基乘法器 [64][65][66][67]。

同时，国内许多作者对有限域乘法器的研究也做了大量工作。Fan和Dai[68]在GF(2^m)上重新定义了乘法器的输出函数，并根据该定义，提出一种基于正规基表示的域元素乘法的快速软件实现算法。2004年，鲁俊生等[69]提出了一种有限复合域上的快速乘法器，该乘法器采用串、并行计算相结合的原则，增加少量硬件规模，极大提高了乘法器的计算速度。另外，Wu等人也对正规基乘

法器进行深入的研究[70]。

GF(2^m)上的并行多项式基乘法器最早由 Baxtee 和 Schneider[71]提出。在文献[47][72]中，Mastrovito 提出了一种多项式基乘法算法，并给出了其硬件体系结构。Sunar 和 Koc[73]利用三项式，对 Mastrovito 算法给出了一个全新公式。在文献[74]，Halbutogullari 和 Koc，推广了 sunar 和 Koc 的思想，并找到对任意不可约多项式构造 Mastrovito 乘法器的方法。迄今为止，对这些特定的多项式，就 XG 的数量和时间延迟来说，Halbutogulari 和 Koc 提出的算法具有最低的时间复杂度。在文献[75]中，zhang 和 Parhi 提出一种设计 Mastrovito 乘法器的对称方法。Parhi 和 Song 又将这种方法应用于设计改进的 Mastrovito 乘法方案[76]，并对两类不可约五项式，提出了新的 Mastrovito 乘法器的复杂性结构。不同于 Mastrovito 乘法器，通过首先直接相乘 GF(2^m)中的两个元素，随后再进行取模，许多文献如[77]等就采用的这种方式。最近，Henriquez 和 Koc[78]对特殊的五项式提出了一个多项式基乘法器，尽管作者称其乘法器为 Mastrovito 乘法器，但是他们的体系结构与最初的 Mastrovito 乘法器是完全不同的。

Montgomery 算法是 1958 年由 Montgomery 提出[79]，由于它用乘法运算替代运算复杂度高的乘逆运算，大大降低了除法的复杂度，因而受到了广泛的应用。1996 年，Koc 和 Acar 成功地将 Montgomery 乘法算法移植到二元域上[80]。这以后，大量高效的二元域上 Montgomery 乘法和指数运算的软件实现被提出来，同时利用稍稍对 Koc 和 Asar 方法加以扩展的位并行乘法器和平方器也被设计得到，主要针对某一类特殊的有限域。一种可伸缩的标准化的并且同时支持二元域运算和有限域运算的乘法器结构也在 2000 年时被提出[81]。这里的改进主要是针对 Koc 和 Acar 的算法。

1.4 创新点概要及结构安排

课题的主要工作及创新如下：

1. 总结公钥密码算法和现有的有限域运算方法，分析这些方法的优劣和技巧。选定二元域进行研究，选定殆逆算法进行改进，选定正规基乘法进行优化。
2. 利用殆逆算法分阶段的特点实现较低延迟，并通过度数相关性快速求出度数，通过步骤分解重组压缩执行周期数，形成了一种较好的殆逆算法硬件移植方案。
3. 针对以上移植方案关键路径延迟复杂度较高这一缺陷，提出了双向移位结构，用反向移位代替求度数中的动态操作。使整个模块布局布线更容易，进一步降低了延迟，提高模块执行的时钟频率。
4. 通过实验测定二元域正规基乘法器的并行度和流水线级数这些参数，使乘法器在开销合理的前提下更加高效。论证了殆逆算法最后一步k的

取值范围，从而使利用查表提速成为可能。

本文按以下方式进行组织：

第一章，提出本文的研究背景以及研究的目的和意义；

第二章，介绍公钥密码技术的相关知识，并根据本文重点，详细介绍最流行的若干公钥算法；

第三章，进一步分解公钥算法，介绍有限域等公钥密码技术相关的数论基础；

第四章，先简要介绍有限域求逆算法的分类和概况，在分析比较这些方法的基础上，提出了两种新的综合性能好，执行效率突出的硬件求逆方案，并通过试验来验证方案的效果；

第五章，优选一种高效的有限域乘法方案，实现并优化其参数，并对第四章提出的求逆方案中乘法做特别优化。

第六章，总结本文工作并对以后的工作进行展望。

第二章 公钥密码算法

公开密钥密码学的概念是由 Whitfield Diffie 和 Martin Hellman 发明的，Ralph Merkle 也独立提出了此概念。它在密码学上的贡献在于使用了一对密钥——加密密钥和解密密钥，且从加密密钥推出解密密钥是不可行的。1976 年，Whitfield Diffie 和 Martin Hellman 在美国国家计算机会议上首先公布了这个概念[82]。

自从 1976 年 Difie 和 Hellman 提出了公钥密码的概念以来，出现了很多的公钥密码系统，所有的这些系统的安全性都依赖于各自所基于的数学问题。近来，其中一些公钥系统已经遭到了破译，另有一些则被证明是难以实现的。目前只有 3 种类型的系统被认为是安全和有效的。根据其依赖的数学问题，可将这些系统分为[83]。

(1) 整数因数分解问题(IFP)：如 RSA 和 Rabin-Williamse。

(2) 离散对数问题(DLP)：如美国政府数字签名算法(DSA)，Difie-Hellman 和 MQV 密钥交换方案，ElGamal 加密和签名方案，Schnorr 和 Nyberg-Rueppel 签名方案。

(3) 椭圆曲线离散对数问题(ECDLP)：如椭圆曲线 DSA (ECDSA)，椭圆曲线版本的 Diffie-Hellman 和 MQV 密钥交换方案，ElGamal 加密和签名方案，Schnorr 和 Nyberg-Rueppel 签名方案。

以上问题都没有被证明是难解的，然而经过全世界数学和计算机科学家长年的细致研究，并没有发现有效的快速算法去解决它们。通过越来越多的研究和使用它们，使我们对其相应的密码系统的安全性有了更多的信心。

现存的公钥算法其中许多是不安全的。那些被视为安全的算法，有许多却不实用，要么是密钥太大，要么密文远大于明文。只有少数几个算法既安全又实用。这些算法多针对数学上的难题。这些既安全又实用的算法，一些仅适用于密钥分配，一些仅适用于加密，还有一些仅适用于数字签名。可同时很好的用于加密和签名的算法有 RSA、ElGamal、Rabin 和 Schnorr，习惯上有人将椭圆曲线也算作一种加密算法，但是我认为椭圆曲线是一种全新的有限域，在这个有限域上可以实现前面所说的 4 种必须实现在有限域上的密码体制。这些算法的共同缺点是慢，现阶段只能用于加密随机会话密钥。将算法实现在硬件上可以有效提速，而由于算法都建立在有限域上，本次毕业设计优化有限域基本运算的硬件实现这一工作的应用范围很广，应用价值很高。

公钥加密的典型过程是，A 在发起通信前获得 B 的公钥 pkB，再用 pkB 加密数据，传输给 B，B 用自己的私钥 skB 解密看到信息。而他人因为不知道 skB 而无法看到内容。

公钥签名的典型过程是，A 用自己的私钥 skA 签名数据(为了减小运算量，

一般是数据的散列)，传输给 B，B 先获得 A 的公钥 pkA，再用 pkA 验证签名是否合法。而他人因为不知道 skA 而无法伪造能通过 pkA 验证的签名。

以上过程所呈现的各种密码算法其实是将信息看作数字，选定另外一个数作为密钥，二者进行特定的数学运算变换(加密签名等)，使原始数据面目全非，从而达到保密等效果。算法关键在于，变换后混乱的信息和密钥配合经过特定运算还能够恢复原貌，即变换后的混乱信息仍能保存原始信息或信息的摘要特征。

下面详细介绍几种典型的公钥密码算法，这些算法都是建立在有限域上，为便于描述和理解，算法建立在素域 GF(p)上。

2.1 RSA 算法

2.1.1 RSA 加密算法流程

1977 年，Rivest，Shamir 和 Adleman 联合提出了一种基于数论中欧拉定理的公钥密码系统，简称 RSA 公钥系统，它的安全性是基于大数因子分解，而因子分解在数学上是一个困难问题[84]。

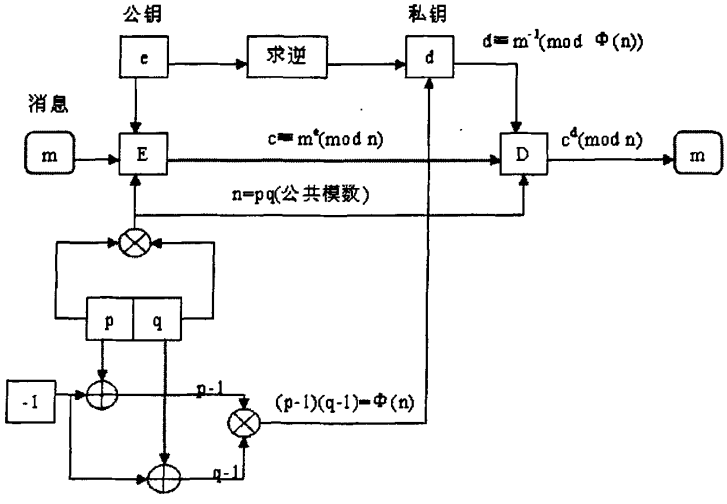


图 2-1 RSA 加密流程

RSA 加密流程可以用图 2-1 清晰的表示。

RSA 体制用户 A 的公钥和私钥的产生：

1) 随机选取两个 100 位(十进制)以上的素数 pa 和 qa。
2) 计算 na = pa × qa，Φ(na) = (pa - 1) (qa - 1)。
3) 随机选取和 Φ(na)互素的整数 ea。
4) 计算 da，满足 ea × da ≡ 1 mod Φ(na)，即计算 ea 模 Φ(na)的逆元 da。
5) 公开 na，ea 作为 pkA，记作 pkA = {na， ea}，保密 pa， qa， da， Φ(na)作为 skA，记作 skA = {pa， qa， da， Φ(na) }。

密钥产生后，加密解密运算非常简单，

加密：$c = pkA(m) = m^{ea} \bmod na$。

解密：$m = skA(c) = c^{da} \bmod na$。

下面简单证明 RSA 算法：

根据欧拉定理：对任意 $e \in Z^*_n$，其中 $Z^*_n = \{x \in Z_n | \gcd(n, x) = 1\}$，有：$e^{\Phi(n)} = 1 \bmod n$，其中 Φ 是欧拉函数。

于是有：

$skA(c) = c^{da} = m^{ea \times da} = m^{k \times \Phi(na)+1} = (m^{\Phi(na)})^k m = m \bmod na$。

2.1.2 RSA 签名方案

RSA 公钥系统既能用于加密，也能用于签名。每个用户都有三个整数 e、d 和 n，n = pq，p 和 q 是大素数。对于密钥对(e，d)，必须满足 $ed \equiv 1 \bmod \Phi(n)$。

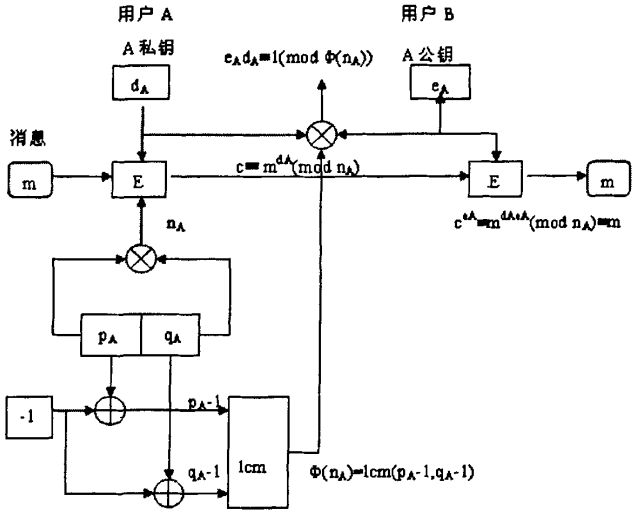


图 2-2 RSA 签名流程

RSA 签名过程可以用图 2-2 清晰的表示。设 A 是签名方，B 是验证方，流程如下：

1) A 计算 $\Phi(na) = lcm(pa - 1, qa - 1)$，这里 lcm 表示最小公倍数；
2) A 计算公钥 ea 满足 $ea \times da \equiv 1 \bmod \Phi(na)$，即求 da 关于模 $\Phi(na)$的逆元；
3) 如果发送方 A 想发送一个对应于消息 m 的已签署消息 c 给接收方 B，那么 A 使用其私钥 skA 签署消息，计算 $c = m^{da} \bmod na$。
4) B 接收到 c 后验证是否满足 $m = c^{ea} \bmod na$，满足则通过验证，不满足则否认签名。

2.1.3 RSA 关键运算分析

➢ 判断互素

RSA 运算中选取 ea 需要计算最大公因子，可以应用欧几里德算法：

定理：记 a，b 公因数 gcd(a，b)，设 a=b，则 gcd(a，b) = gcd(a，b mod a)。

应用此定理，求二数最大公约数演变为求大数模小数的余数和小数的最大公约数，从而可以逐渐简化运算，直到小数整除大数，则小数为二者最大公约数。

求余数过程其实应用减法而非除法，没有效率极低的试商过程。

就此，判断互素可分解为比较大小和减法。

➢ 素性检测或素数生成

✧ 应用费尔马小定理作素性检测

费尔马小定理：如果 p 是素数，1=a=p，那么 $a^{p-1} = 1 \bmod p$

如果我们想知道 n 是否是素数，我们在中间选取 a，看看上面等式是否成立。如果对于数值 a 等式不成立，那么 n 是合数。如果有很多的 a 能够使等式成立，那么我们可以说 n 可能是素数，或者伪素数。

就此，检测过程可分解为乘法和减法。

✧ Miller-Rabin 检测：

数分为素数和合数，一个有意义的问题是一般需要检测多少个随机整数(特定长度)才能找到一个素数。假设定义 Π(N)为小于等于 N 的素数的个数，根据数论中的素数个数定理 Π(N)约等于 $N/\ln {}_2N$。因此如果在 1-N 之间随机选取一个整数，其为素数的概率大约是 $1/\ln {}_2N$。对于 1024 比特的模数 n = p q，p 和 q 将选取为 512 比特的素数。一个随机 512 比特的整数为素数的概率约为 1/ln 512 ~ 1/355。即，一般给定 355 个随机 512 比特整数 p，其中一个会是素数(如果把范围限定为奇数，概率就加倍，大约为 2/355)。

首先选择一个待测得随即数 p，计算 b，b 是 2 整除 p-1 的次数(2^b是能整除 p-1 的最大幂数)，然后计算 m，使得 $p=1+2^bm$。

1) 选择一个小于 p 的随机数 a；
2) 设 j = 0 且 $z = a^m \bmod p$；
3) 如果 z = 1 或 z = p - 1，那么 p 通过测试，可能是素数；
4) 如果 j > 0 且 z = 1，那么 p 不是素数；
5) 设 j = j + 1。如果 j < b 且 $z \neq p - 1$，设 $z = z^2 \bmod p$，然后回到第 4 步。如果 z = p - 1，那么 p 通过测试，可能是素数；
6) 如果 j = b 且 $z \neq p - 1$，那么 p 不是素数。

此算法效率如果断言一个数 p 为“合数”，那么 p 定为合数。如果 p 为素数，则必被断言为素数。另一方面，如果 p 为合数，则此算法断言为“素数”的概率小于 1/4。观察整个算法流程，可以发现仍然可以分解为乘法和减法。

RSA 算法经过分解，可以分解为有限域上的基本运算加法，减法、乘法和求逆，这正体现了公钥算法建立在有限域上。

2.2 ElGamal 算法

ElGamal 于 1985 年提出了一个公钥加密系统。ElGamal 可以同时应用于加密和数字签名。系统同样也建立在有限域上，其安全性依赖于有限域上计算离散对数的困难。ElGamal 每次加密签名都要引入一个随机数，这使得同样的密钥处理同样的信息也能得到不同的结果，实现了一次一密。但是其缺点是加密签名后的信息量是原始信息的两倍，所以只适用于少量信息的安全需求。

2.2.1 ElGamal 签名算法流程

为了描述 ElGamal 系统，选择一个素数 p 和两个随机数 g 和 x，使其满足 $g < p$ 和 $x < p$，且 g 是模 p 的本原根。这里 x 是私钥；计算 $y = g^x \bmod p$， y、g 和 p 公钥。

要签署消息 m，分以下几步：

1) 首先选择随机数 k，使其满足 gcd(k，p-1) = 1，即和 p - 1 互素；

2) 计算 $r = g^k \bmod p$

3) 对 m 的签名为数据对(r，s)，其中 $r = 0$，$s < p - 1$；

有：$g^m = y^r r^s \pmod p$

$=(g^x)^r (g^k)^s \pmod p$

$= g^{xr+ks} \pmod p$

于是有：$m = x r + k s \bmod(p-1)$

求解 $s = (m - x r) k^{-1} \bmod(p-1)$

验证签名时，使用公钥(y，g，p)验证(r，s)是否满足 $g^m = y^r r^s \pmod p$，满足则通过，否则签名为伪造。

ElGamal 签名算法流程可以由图 2-3 表示：

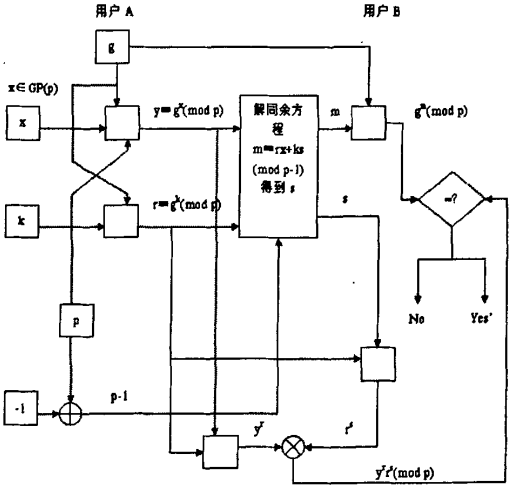


图 2-3 ElGamal 数字签名流程

ElGamal 算法也能用来加密，但是由于其密文加倍和一次一密等特点，算

法在实际应用中多用于签名，这里就不赘述加密流程了。

2.1.2 ElGamal 关键运算分析

ElGamal 密钥生成时，要求分量 g 是模 p 的本原根。本原根定义如下：

如果 a 的阶 m 等于 f(n)，则称 a 为 n 的本原根（生成元）。如果 a 是 n 的本原根，则 a，a^2，...，$a^{f(n)}$在 mod n 下互不相同且都与 n 互素。

特别地，如果 a 是素数 p 的本原根，则 a，a^2，...，a^{p-1}在 mod p 下都不相同。如果 g 不是本原根，伪造签名就有可能通过验证。

通常，找出一个本原根不是一件容易的问题。然而，如果你知道 p-1 的因子，这个问题就很容易。设 q_1，q_2...，q_n是 p-1 的素因子，为了测试一个数 g 是否是模 p 的本原根，对所有 q_1，q_2...，q_n计算 g(p-1)/q (mod p)，如果对某个 q 值，其结果为 1，那么 g 不是原根；如果对任何 q，结果不等于 1，则 g 是原根。

如果要找模 p 的原根，只需要随机地选择一个从 1 到 p-1 之间的数来测试。只要选择足够多，可以很快找到。

选择 g 的操作，仍然归结为有限域上的基本运算。

ElGama 算法经过分解，也可以分解为有限域上的基本运算，再次体现了公钥算法建立在有限域上。

2.3 椭圆曲线密码系统

椭圆曲线密码系统(ECC)[83][86]在 1985 年分别由 Victor Miller 和 Neal Koblitz 独立提出。从 1985 年以来，ECC 受到全世界密码学家、数学家和计算机科学家的密切关注。一方面，由于没有发现 ECC 明显的漏洞，使人们充分相信其安全性;另一方面，在增加 ECC 系统的实现效率上取得了长足的进步，到今日 ECC 不仅可被实现，而且成为已知的效率最高的公钥密码系统。

ECC 相对于 RSA 和 DSA 等系统吸引人的最主要的原因是解决其数学问题(即 ECDLP)的已知的最好的算法也要用完全指数时间。与之相比，RSA 和 DSA 所基于的数学问题(即因数分解 IFP 和离散对数 DLP 问题)都有亚指数时间算法。这意味着随着长度的增加，求解 ECDLP 的难度比求解 IFP 和 DLP 的难度增加得快得多。因此 ECC 仅需要更小的密钥长度就可以提供跟 RSA 和 DSA 相当的安全性，如表 1-1。

表 1-1 具有相同安全强度的 ECC 和 RSA/DSA 的密钥长度

RSA/DSA 密钥长度(bits)	ECC 密钥长度(bits)
512	106
768	132
1024	160
2048	210
21000	600

随着整数因子分解方法的不断完善、计算机速度的提高以及计算机网络的发展，作为 RSA 加解密安全保障的大整数要求越来越大。目前一般认为 RSA 需要 1024 位以上的字长才有安全保障。但是，密钥长度的增加导致了其加解密的速度大大降低，硬件实现也变得越来越困难，这对使用 RSA 的应用带来了很重的负担，从而使得其应用范围越来越受到制约。在此背景下，ECC 的优势就显得更为突出。

由于椭圆曲线密码系统的以上优点，本文实现的模块参数均适用于椭圆曲线，但参数改变后也可用于 RSA 等其他公钥密码系统。

基于椭圆曲线的密码体制取决于椭圆曲线的点的算术运算。椭圆曲线算术根据其所依赖的域的运算来定义，运算的效率是关键。因此，椭圆曲线算术运算的高效实现是至关重要的问题。

2.3.1 椭圆曲线的定义

椭圆曲线定义比较复杂，完整的定义如下。

域 K 上的椭圆曲线 E 由下述方程定义：

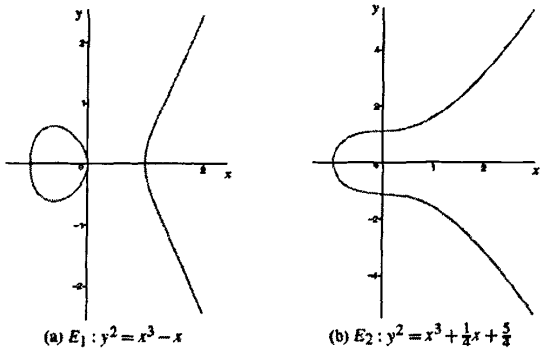
$$E:\ y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \qquad (1)$$

其中，a_1，a_2，a_3，a_4，$a \in K$ 且 $\Delta \neq 0$，Δ 是 E 的判别式，具体定义如下：

$$\left.\begin{aligned} \Delta &= -d_2^2 d_8 - 8d_4^3 - 27d_6^2 + 9d_2 d_4 d_6 \\ d_2 &= a_1^2 + 4a_2 \\ d_4 &= 2a_4 + a_1 a_3 \\ d_6 &= a_3^2 + 4a_6 \\ d_8 &= a_1^2 a_6 + 4a_2 a_6 - a_1 a_3 a_4 + a_2 a_3^2 - a_4^2. \end{aligned}\right\} \qquad (2)$$

的集合 $\{(x, y) \mid y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x - a_6 = 0\} \cup \{8\}$

8 是无穷远点。



(a) $E_1: y^2 = x^3 - x$　　(b) $E_2: y^2 = x^3 + \frac{1}{4}x + \frac{5}{4}$

图 2-4 R 上的椭圆曲线

定义的注释

1. 式 1 称为 Weierstrass 方程。

2. 我们称 E 是域 K 上的椭圆曲线，这是因为系数 a_1，a_2，a_3，a_4，a_6 均为域 K 的元素。有时我们将椭圆曲线记为 E/K 以强调椭圆曲线 A 定义在域 K 上，并称 K 为 E 的基础域。注意，如果椭圆曲线 E 定义在域 K 上，则 E 也定义在 K 的扩域上。
3. 条件△≠0 确保椭圆曲线是"光滑"的，即曲线的所有点都没有两个或两个以上不同的切线。
4. 点8 是曲线的唯一的一个无穷远点，它满足投影形式的 Weierstrass 方程。
5. 曲线 E 的 L 有理数点是满足曲线方程且坐标 x 和 y 属于 L 的点(x，y)，并认为无穷远点是 K 的所有扩域上的 L 有理数点。

下面举两个 R 上的椭圆曲线为例，考虑定义在实数域 R 上的椭圆曲线

E_1：$y^2 = x^3 - x$ 和 E_2：$y^2 = x^3 + x/4 + 5/4$

在图 2-4 给出了 $E_1(R)\backslash\{8\}$和 $E_2(R)\backslash\{8\}$。

当然，密码学中使用的椭圆曲线一般定义在有限域 GF(p)或 $GF(2^m)$等上。

➢ 简化的 Weierstrass 方程

由 Weierstrass 方程给出的定义在域 K 上的两个椭圆曲线 E_1 和 E_2

E_1：$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$

E_2：$y^2 + a_1'xy + a_3'y = x^3 + a_2'x^2 + a_4'x + a_6'$

被称为在域 K 上是同构的，如果存在 u，r，s，t ∈K 且 u≠0，使得变量变换

$$(x, y) \rightarrow (u^2x + r,\ u^3y + u^2sx + t) \qquad (3)$$

把方程 E_1 变成方程 E_2 的式(3)的变换称为变量的相容性变换。

定义在于 K 上的一个 Weierstrass 方程

E_1：$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$

能够用变量的相容性变换来简化。这种简化的方程才是椭圆曲线密码体制中实际使用的曲线。我们将分别考虑域 K 的特征等于 2 或 3 和域的特征不等于 2 和 3 两种情况。

1. 如果域 K 的特征不等于 2 或者 3，则变量的相容性变换

$$(x, y) \rightarrow \left(\frac{x - 3a_1^2 - 12a_2}{36}, \frac{y - 3a_1x}{216} - \frac{a_1^3 + 4a_1a_2 - 12a_3}{24}\right)$$

把 E 变成为曲线

$$y^2 = x^3 + ax + b \qquad (4)$$

其中 a，b∈K。曲线的判别式△ = $-16(4a^3 + 27b^2)$。

2. 如果域 K 的特征是 2，那么有两种情况要考虑。如果 $a_1 \neq 0$，那么变量的相容性变换

$$(x, y) \rightarrow \left(a_1^2x + \frac{a_3}{a_1}, a_1^3y + \frac{a_1^2a_4 + a_3^2}{a_1^3}\right)$$

把 E 变换为曲线

$y^2 + xy = x^3 + ax^2 + b$ (5)

其中 a，b∈K。这样的曲线我们称为非超奇异的，并且判别式△ = b。如果 $a_1 = 0$，那么变量的相容性变换

$(x, y) \rightarrow (x + a_2, y)$

把 E 变换成为曲线

$y^2 + cy = x^3 + ax + b$ (6)

其中，a，b，c∈K。这样的曲线我们称为超奇异的，并且判别式△=c^4。

3. 如果域 K 的特征是 3，那么也存在两种情况需要考虑。如果 $a_1^2 \neq -a^2$，那么变量的相容性变换

$$(x, y) \rightarrow \left(x + \frac{d_4}{d_2}, y + a_1 x + a_1 \frac{d_4}{d_2} + a_3\right),$$

其中 $d_2 = a_1^2 + a_2$，$d_4 = a_4 - a_1 a_3$，把 E 变换为曲线

$y^2 = x^3 + ax^2 + b$ (7)

其中 a，b∈K。我们称这样的曲线为超奇异的，并且判别式△ = $-a^3$。

2.3.2 点的运算法则

令 E 是一个定义在域 K 上的椭圆曲线。根据“弦和切线”法则，E(K)上的两个点相加得到 E(K)上的第三个点。点集合 E(K)及其这种加法运算构成一个加法交换群，并且8为其无穷远点。就是这个群被用来构建椭圆曲线密码体制。

群的加法规则最好用几何方法说明。令 $P = (x_1, y_1)$和 $Q = (x_2, y_2)$是椭圆曲线 E 上的两个不同的点，则 P 与 Q 的和 R 如下定义。首先画一条连接 P 和 Q 的直线，这条直线与椭圆曲线相交于第三点，那么这个交点关于 x 轴的对称点就是 R 点。这一几何表示示于图 2-5(a)中。

按如下方法求出 P 的倍点 R。首先在 P 点画椭圆曲线的切线，这条切线与椭圆曲线相交于第二点，这个交点关于 x 轴的对称点就是 R 点。这一几何表示示于图 2-5(b)中。

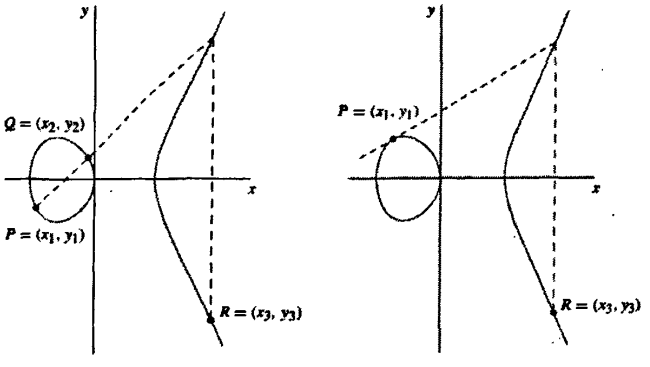


图 2-5 椭圆曲线的点和倍点运算的几何表示

可以从上述几何描述中得出群运算的代数公式。下面，我们分别对于以下三种情况给出对应于式(4)的简化 Weierstrass 方程的群运算代数公式：

1. 基础域 K 的特征不等于 2 或者 3(如 $K = F_p$，p 是大于 3 的素数)。
2. 式(5)的非超奇异椭圆曲线 E，$K = F_2^m$。
3. 式(6)的超奇异椭圆曲线 E，$K = F_2^m$。

➢ 群的运算法则(E/K：$y^2 = x^3 + ax + b$，域 K 特征不等于 2 或者 3)

1. 单位元：对于所有的 $P \in E(K)$，$P + 8 = 8 + P = P$。
2. 负元素：如果 $P = (x, y) \in E(K)$，那么 $(x, y)+(x, -y) = 8$。记点$(x, -y)$为-P，并称其为 P 的负。注意，-P 也是 E(K)上的一个点，此外$-8 = 8$。
3. 点加：令 $P = (x_1, y_1) \in E(K)$，$Q = (x_2, y_2) \in E(K)$，$P \neq \pm Q$，那么 $P + Q = (x_3, y_3)$。其中，

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2 \quad \text{and} \quad y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1.$$

4. 倍点：令 $P = (x_1, y_1) \in E(K)$，$P \neq -P$，那么 $2P = (x_3, y_3)$。其中，

$$x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1 \quad \text{and} \quad y_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3) - y_1.$$

➢ 群的运算法则(非超奇异椭圆曲线 E/F_2^m：$y^2 + xy = x^3 + ax^2 + b$)

1. 单位元：对于所有的 $P \in E(F_2^m)$，$P + 8 = 8 + P = P$。
2. 负元素：如果 $P = (x, y) \in E(F_2^m)$，那么$(x, y) + (x, -y) = 8$。记点$(x, -y)$为-P，并称其为 P 的负。注意，-P 也是 $E(F_2^m)$上的一个点，此外$-8 = 8$。
3. 点加：令 $P = (x_1, y_1) \in E(F_2^m)$，$Q = (x_2, y_2) \in E(F_2^m)$，$P \neq \pm Q$，那么 $P + Q = (x_3, y_3)$。其中

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a \qquad y_3 = \lambda(x_1 + x_2) + x_3 + y_1$$

而 $\lambda = (y_1 + y_2) / (x_1 + x_2)$

4. 倍点：令 $P = (x_1, y_1) \in E(F_2^m)$，$P \neq -P$，那么 $2P = (x_3, y_3)$。其中，

$$x_3 = \lambda^2 + \lambda + a = x_1^2 + \frac{b}{x_1^2} \quad \text{and} \quad y_3 = x_1^2 + \lambda x_3 + x_3$$

而 $\lambda = x_1 + y_1/x_1$。

➢ 群的运算法则(超奇异椭圆曲线 E/F_2^m：$y^2 + cy = x^3 + ax + b$)

1. 单位元：对于所有的 $P \in E(F_2^m)$，$P + 8 = 8 + P = P$。
2. 负元素：如果 $P = (x, y) \in E(F_2^m)$，那么$(x, y) + (x, y + c) = 8$。记点$(x, y + c)$为-P，并称其为 P 的负。注意，-P 也是 $E(F_2^m)$上的一个点，此外$-8 = 8$。
3. 点加：令 $P = (x_1, y_1) \in E(F_2^m)$，$Q = (x_2, y_2) \in E(F_2^m)$，$P \neq \pm Q$，那么 $P + Q = (x_3, y_3)$。其中，

$$x_3=\left(\frac{y_1+y_2}{x_1+x_2}\right)^2+x_1+x_2 \quad \text{and} \quad y_3=\left(\frac{y_1+y_2}{x_1+x_7}\right)(x_1+x_3)+y_1+c.$$

4. 倍点：令 $P=(x_1, y_1)\in E(F_2^m)$，$P\neq -P$，那么 $2P=(x_3, y_3)$。其中，

$$x_3=\left(\frac{x_1^2+a}{c}\right)^2 \quad \text{and} \quad y_3=\left(\frac{x_1^2+a}{c}\right)(x_1+x_3)+y_1+c.$$

2.3.3 椭圆曲线密码体制

给定有限域 F_q 和定义在有限域 F_q 上的椭圆曲线 $E(F_q)$，对于已知椭圆上的点 P，求 Q = kP (kP 称为椭圆曲线上的点乘，即 k 个 P 在椭圆曲线上的点加运算)很容易，但是反过来在已知 P 和 Q 的情况下求 k 却非常困难。这就是椭圆曲线群上的离散对数问题(ECDLP)，椭圆曲线公钥密码系统的安全性就可以基于椭圆曲线离散对数问题的难解性。类似于大数分解难，两个点求 P 和 Q 的点加很容易，而根据点加结果分解出原 P 点和 Q 点很难。于是可以说，椭圆曲线密码体制中椭圆曲线群的地位相当于之前公钥算法的基础有限域。

➢ 系统的建立

椭圆曲线密码系统必须选用安全的椭圆曲线，否则整个系统都是不安全的。所谓的安全曲线是指能抗各种已有攻击的曲线，目前对椭圆曲线攻击比较有效的方法有 MOV 攻击、Smart 方法、Pollard-ρ 方法等。为抵抗上述攻击，有限域 F_q 上的椭圆曲线 $E(F_q)$应满足：

1. 椭圆曲线群的阶(即有限域内满足椭圆曲线的点数)$\#E(F_q)$有大于 2^{160} 且大于 $4q^{1/2}$ 的素数因子；
2. $E(F_q)$不是超奇异曲线或反常曲线，这两类曲线在所有的椭圆曲线公钥密码的标准中都被指定禁止使用。

构成一个椭圆曲线密码系统所需要的参数集称为椭圆曲线域参数，包括：有限域 F_q，椭圆曲线 E，基点 G(椭圆曲线 E 上的一个点)，基点的阶 n(使得 nG = O 的最小正整数，最好是素数)，椭圆曲线群的阶$\#E(F_q)$，相关因子 h。椭圆曲线的域参数是公开信息。

如果选择的有限域是素数域 F_p，则椭圆曲线参数是一个六元组(p，a，b，G，n，h)。其中 p 决定有限域，a，b 决定椭圆曲线，G 表示基点，n 是 G 的阶，h 表示椭圆曲线群的阶与与 n 的商。

如果选择有限域是特征为 2 的有限域 $F(2^m)$，则椭圆曲线参数域是一个七元组(m，f(x)，a，b，G，n，h)，其中 m 是有限域的尺寸，f(x)是域中的多项式，a，b 决定椭圆曲线，G 表示基点，n 是 G 的阶，h 表示椭圆曲线群的阶与 n 的商。

实际应用中基于椭圆曲线的密码体制有很多，下面举一种椭圆曲线密码体制。

➢ 密钥的产生

系统建立以后，每个使用者(简称实体)执行以下计算：

1) 在区间[1，n - 1]中随机选取一个整数 d；

2) 计算点 $Q = dG$；

3) 实体的公钥包含点 Q；

4) 实体的私钥是整数 d。

➢ 椭圆曲线加密体制(ECES)

当实体 B 发送信息 M 给实体 A 时，实体 B 执行如下加密步骤：

1) 实体 B 去查找公钥库 PKDB，查到 A 的公钥 Q_A；

2) 将 M 划分为较小的数据块，$M = \{m_1, m_2, \ldots, Mr\}$ ，其中 $0 \leqq m \leqq n$；

3) 在区间[1，n - 1]上选择一个随机数 k；

4) 计算点 $X_1(x_1, y_1) = kG$；

5) 计算点 $X_2(x_2, y_2) = kQ_A$，如果 $x_2 = 0$，则回到第 3 步；

6) 计算 $c = mx_2$；

7) 传送加密数据(x_1, y_1, c)给 A。

实体 A 接收到从 B 发来的密文(x_1, y_1, c)时，执行如下的解密步骤：

1) 使用他的私钥 d，计算点 X_2：$d_AX_1 = d_A(kG) = k(d_AG) = kQ_A = X_2$；

2) 计算 $m = c(x_2)^{-1}$，恢复出数据 m。

➢ 椭圆曲线数字签名体制(ECDSA)

当实体 A 为实体 B 签名信息 M 时，A 执行下列步骤生成 ECDSA 签名：

1) 将信息 m 表示成二进制串；

2) 使用一个 hash 函数计算 hash 值 $e = H(m)$；

3) 在区间[1，n - 1]内选取一个随机数 k；

4) 计算点$(x_1, y_1) = kG$ 并设 $r = x_1 \bmod n$；

5) 利用私钥 d，计算 $s = k^{-1}(e + rd) \bmod n$；

6) A 传送给 B 信息 m 和它的签名(r，s)。

注意 当 $r = 0$ 或 $s = 0$ 则签名验证失败。但是，如果 k 是随机选取的，则 r=0 或 s=0 的概率非常之小，可以忽略不计。

当实 体B 验证 A 对信息 m 的签名(r，s)时，B 执行下列步骤对 A 的签名进行验证：

1) 查找 A 的公钥 Q；

2) 如果 $r \bmod n = 0$，则拒绝签名；

3) 计算 hash 值 $e = H(m)$；

4) 计算 $s^{-1} \bmod n$；

5) 计算 $u = s^{-1}e \bmod n$ 和 $v = s^{-1}r \bmod n$；

6) 计算点$(x_1, y_1) = uG + vQ$；

7) 接受 A 对信息 m 的签名当且仅当 $x_1 \bmod n = r$。

注意签名过程中 hash 值 e 模 n 约简，因此，为保证安全，hash 函数 H 和 n 的选择应使得 H mod n 仍是一个密码安全的 hash 函数。如果 r = 0，则签名方程 $s = k^{-1}(e + rd) \bmod n$ 就不包含私钥 d。出于安全需要，可以在生成签名的步骤(4)中强制 $r \neq 0$。

2.4 本章小结

公钥密码体制是现代密码学的核心。本章阐述了公钥密码算法的基本原理，展示了算法在安全系统中所处的地位和产生的作用。

本章还详细列举了当前实际应用和研究工作中常见的，具有代表性的若干算法。尤其从多个角度，详细描述了实际中的标准 RSA 和研究的热点椭圆曲线这两个重点。通过这些算法我们基本可以窥到公钥算法的运作流程和特点，而其他公钥密码体制和本章列举的这些例子都有异曲同工之处。

本章体现了有限域运算对于公钥密码算法的基础性地位。

第三章 相关数学原理

公钥算法都建立在有限域上，本章将详细介绍安全相关的数论知识，主要介绍有限域的基本概念、域元素的不同表示方法、有限域的相关性质。数学理论的作用不仅仅是作为公钥密码体制的基础，更对从本质上提高执行效率益处很大。

3.1 群和域

定义 1：任意给定一非空集合 G 和其上的二元运算“*”，如果满足：

1. 封闭性：对任意 a，b ∈G，存在 c ∈G，使得 a * b = c；
2. 结合率：对于任意 a，b ，c ∈G，有(a * b) * c = a * (b * c)；
3. 单位元 e 存在：即存在 e ∈G，对于任意 a ∈G，都有 e * a = a * e = a；
4. 逆元存在：对于任意 a ∈G，存在 b ∈G，使得 a * b = b * a = e，

则称集合 G 关于二元运算“*”成群，记为< G，* >。

在群< G，* >中，如果对于任意 a，b ∈G，都有 a * b = b * a，则称群< G，* >为交换群，也称为阿贝尔(Abel)群。

定义 2：令<G，*>是群。如果存在 a ∈G，对任意 b ∈G，总存在非负整数 i，使得 $b = a^i$，则称群 G 是循环群，a 称为群 G 的生成元，或者称本原元。

定义 3：设“+”，“*”是 G 上的二元运算，集合基数|G| > 1，如果满足：

1. < G，+ >是一个交换群，其单位元记为 0；
2. < G - 0，* >是交换群，其单位元记为 1；
3. 运算“*”对“+”可分配，即对任意 a，b ，c ∈G，都有 a* (b+c) = (a*b)+(a*c) 和(a+b)*c = (a*c)+(b*c)，

则称< G，+ ，* >是域。

例如：< N，+ >是自然数集关于“+”所成的群，而< Q，+，× >则是有理数集上关于“+”和“×”所成的域。

3.2 有限域

如果域 G 中元素个数有限，则称 G 为有限域或伽罗华(Galois)域，其中，G 中元素个数成为有限域 G 的阶，记为|G|。

例如，对任意给定的素数 p，小于等于 p 的非负整数集合 C 和定义在这个集合上的模 p 加法和模 p 乘法构成一个有限域 GF(p)。模 p 加法和乘法分别定义为对于任意 a，b ∈C，$a +_p b = a + b \bmod p$，$a \times_p b = a \times b \bmod p$。

有限域元素个数确定，

3.2.1 有限域上多项式

设 G 是任意有限域，G 上的多项式 f(x)定义如下：

$f(x) = f_0 + f_1x + f_2x^2 + \ldots + f_{m-1}x^{m-1} + f_mx^m$

其中，m 是非负整数，系数 $f_i \in G$，I = 0，1，...，m。我们称 m 是多项式 f(x)的次数，记为 deg(f(x))，将多项式看作有限域元素时，也称度数，在后文中经常用到。f_0 是常数项，f_m 是首系数。 如果 $f_m = 1$，称多项式 f(x)是首 1 多项式。记系数在 G 中所有含 x 的多项式组成的集合为 G[x]。

定义 4：f(x)是 G 上的不可约多项式，如果 f(x)在 G 上不能分解为次数更低的两个多项式的乘积。

定义 5：假设 f(x)，g(x) ∈G[x]，如果

d(x)是首一多项式；d(x)整除 f(x)，且整除 g(x)；对任意 h(x) ∈G[x]，如果 h(x)整除 f(x) 且整除 g(x)，则 h(x)整除 d(x)，则称 d(x)是 f(x)和 g(x)的最大公因式(Greatest Common Divisor，GCD)，记 d(x) = GCD(f(x)，g(x))。

3.2.2 用到的有限域性质

下面给出的性质和定理可参见文献[87]。

定理 1：对任意 f(x) ∈G[x]，剩余类环 G[x]/ f(x)(任意 h(x) ∈G[x]/ f(x)，当且仅当存在 l(x) ∈G[x]使得 h(x)是 l(x)除以 f(x)后的余式)是域当且仅当 f(x)是 G 上的不可约多项式，其中，G 是有限域。

定理 2 (有限域的存在性和唯一性定理)：

1. 如果 G 是有限域，则存在某个素数 p 和正整数 m≧1，使得 G 包含 p^m 个元素；
2. 对于每个素数幂阶 p^m，存在唯一的(从同构意义上讲)阶为 p^m 的有限域，记为 $GF(p^m)$。

定理 3：(子域准则)令 G 是有 $q = p^n$ 个元素的有限域，则其每个子域的阶是 p^m，且 m | n。反过来，如果 m | n，则一定存在含 p^m 个元素的子域。而且，元素 a ∈在含 p^m 个元素的子域中，当且仅当 a 的 p^m 次幂等于 a。

定理 4：如果有限域 G 是含有 q 个元素的有限域，则对每个 a ∈G，都满足 $a^q = a$。

定理 5：由有限域 G 的非零元构成的乘法群<G-0，*>是循环群。

3.3 有限域类型

3.3.1 素数域 GF(p)

令 p 是一个素数，则{0，1，...，p-1}组成一个有限域，称为素域，记为 GF(p)。其上的加法、减法、乘法、除法等运算分别定义如下：

加法：如果 a，b，则 a+b = r，其中，r 是 a+b 除以 p 的余数，即 0=r=p-1，

该操作称为模 p 加法。

减法：域元素的减法运算可以转化为加法运算，即如果 a，b $\in$GF(p)，则 a - b = a + (-b)，其中，-b $\in$GF(p)，且-b + b = 0(-b 称为 b 的加法逆元)。

乘法：如果 a，b，则 ab = s，其中，s 是 ab 除以 p 的余数，即 0=s=p-1，该操作称为模 p 乘法。

除法：域元素的除法运算可以转化为乘法运算和求逆运算，即如果 a，b $\in$GF(p)，$b \neq 0$，则 $a/b = ab^{-1}$，其中 $b^{-1} \in$GF(p)，且 $b^{-1}b = 1$(b^{-1}称为 b 的乘法逆元，一般简称逆元)。因此，对除法运算的研究一般都转化为对求逆运算的研究，本文也是如此。

显然，在素数域 GF(p)中，主要的运算是加法、乘法，求逆和取模运算，其中，计算开销最大的是乘法、求逆和取模运算。

3.3.2 二元扩域 $GF(2^m)$

二元域 $GF(2^m)$也是有限域的一种，由素域 GF(2)扩张[90]而来。

其元素可以为最高次数 m-1，系数 $a_i \in$GF(2) 的任意多项式 $a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \ldots + a_1x + a_0$，用二进制序列 $a_{m-1}a_{m-2}\ldots a_1a_0$ 表示。

其基本运算也是加法、减法、乘法和求逆四种，且均为多项式的加减乘除。多项式运算规则和我们熟悉的多项式运算相同。加法为同次项系数相加，注意这里的相加是域 GF(2)上的加法，即异或。二元域的减法和加法相同，也是按位异或。同样，乘法和我们熟悉的多项式乘法之间的差别也是用异或代替同次项系数的加。乘积式的次数会超过 m-1，用 m 次不可约多项式来约简结果。

例如：

$GF(2^3)$={000，001，010，011，100，101，110，111}

不可约多项式：　$x^3 + x + 1$

加法：　　010 + 110 = 100

乘法：　　$101 \times 011 = (1111) \bmod (x^3 + x + 1) = 100$

3.3.3 扩域 $GF(p^m)$

推广二元扩域的思想，可得到一般扩域的概念。

定义 6：令 p 是一个素数，令 f(x)是 GF(p)上次数为 m 的不可约多项式，则扩域 $GF(p^m)$中的元素由所有 GF(p)上次数小于 m 的多项式组成，即

$GF(pm) = \{ a_{m-1}x^{m-1} + \quad a_{m-2}x^{m-2} + \ldots + a_1x + a_0 \mid ai \in GF(p)\}$

一般扩域与二元扩域上的运算不同之处在于：二元扩域上的取模运算是模 2 运算，而一般扩域上的取模运算是模 p 运算，因此，此处略去一般扩域运算的介绍。

其实以上介绍的前两种有限域都是扩域得特例，而扩域又是以素数域为基

础的。

3.4 有限域的基

有限域 GF(p)的有限扩域 $GF(p^m)$可以看成是有限域 GF(p)上的 m 维向量空间。如果$\{a_0, a_1, \ldots, a_m\}$是 $GF(p^m)$在 GF(p)上的基，则对任意元素 $A \in GF(p^m)$均可唯一表示为如下形式：

$$A=a_0a_0+a_1a_1+\ldots+a_{m-1}a_{m-1}，\text{其中 } a_i \in GF(p) \qquad (2\text{-}9)$$

一般地，$GF(p^m)$在 GF(p)上的基有多种，但本文最有兴趣的是下面两种基：正规基(Normal basis，NB)和多项式基(Polynomial Basis，PB)或者称为标准基(Standard or Canonical basis)。下面我们给出这两种基的基本概念和相关性质(此处仅讨论 p=2 的情形)。

3.4.1 多项式基

如果$f(x) = x^m + \sum_{i=0}^{m-1} f_i x^i$ ($f_i \in \{0, 1\}$，i=0，1，…，m-1)是不可约多项式，则 f(x)被称为约化多项式。对每个约化多项式，总存在一个多项式基表示。即

设$N = \{a_0, a_1, \ldots, a_{m-1}\}$是$GF(2^m)$在GF(2)上的一组基，如果存在$\beta \in GF(2^m)$，使得$a_i = \beta^i$，0=i=m-1，则称N为一组多项式基。

如果使用多项式基表示，则$GF(2^m)$的每一个元素分别对应一个次数不超过 m，系数为0或1的多项式，即，对任意$a \in GF(2^m)$，总存在m个数$a_i \in \{0, 1\}$，i = 0，1，…，m-1，使得

$$A = a_{m-1}x^{m-1} + \ldots + a_1x + a_0$$

一般地，可将元素a表示为长度为m的比特串$(a_{m-1}, \ldots, a_1, a_0)$。

假设$a = (a_{m-1}, \ldots, a_1, a_0)$和$b = (b_0, b_1, \ldots, b_{m-1})$，则使用多项式基表示域元素运算的定义如下：

加法：$a + b = c = (c_{m-1}, \ldots, c_1, c_0)$，其中，$c_i = (a_i + b_i) \bmod 2$，i = 0，1，…，m-1。即域元素加法是按对应位异或。

乘法：$ab = c = (c_{m-1}, \ldots, c_1, c_0)$，其中，$c(x) = c_{m-1}x^{m-1} + c_{m-2}x^{m-2} + \ldots + c_1x + c_0$是$(\sum_{i=0}^{m-1} a_i x_i)(\sum_{j=0}^{m-1} b_j x_j) \bmod f(x)$的余式。

3.4.2 正规基(Normal basis)

假设 γ 是 GF(2)上次数为 m 的不可约多项式 f(x)(称为正规多项式)的根。如果集合 $N=\{\gamma, \gamma^2, \ldots, \gamma^{2^{m-1}}\}$中的元素是线性独立的，则集合 N 是扩域 $GF(p^m)$上的一个正规基。通过寻找正规多项式，可以建立 GF(2) 的扩域 $GF(p^m)$的正规基，而检测一个不可约多项式是否正规的方法可参见文献[91]。

由于二元扩域 $GF(2^m)$被看作是定义在 GF(2)上的 m 维向量空间，因此，可以给出下面的定义：

定义 7：任意 $\gamma \in GF(2^m)$，如果 $\gamma, \gamma^2, \ldots, \gamma^{2^{m-1}}$ 是线性独立的，则元素 γ

是正规的。

定义 8：$N=\{\gamma, \gamma^2, \ldots, \gamma^{2^{m-1}}\}$被称为由 γ 产生的正规基，如果 γ 是正规的，且 N 是基。

➢ T 型正规基的构造

1. I 型最优正规基的构造

定理 6[91]：假设 m + 1 是素数，2 是模 m + 1 的一个原根(即，2 的幂模(m + 1)生成 1～m 中的元素)，则 GF(2)上 m 个非单元的(m + 1)次单位根是线性无关的，且组成 $GF(2^m)$到 GF(2)上的一组最优正规基，记 $N=\{a^{2^i} \mid i=0, \ldots, m-1\}=\{a_j \mid j=1, \ldots, m\}$，其中 a 是一个(m+1)次本原单位根(i.e.$a^{m+1}=1$，但 $a_i \neq 1 (1 \leq i \leq m+1)$)，称 N 是 $GF(2^m)$到 GF(2)上的一组 I 型最优正规基。

2. II 型最优正规基的构造

定理 6[91]：令 $GF(2^m)$是含有 2^m 个元素的有限域。如果 2m+1 = p 是素数，且下面两个条件之一成立：

- 2 是模 p 原根；
- P=3(mod 4)，且 2 模 p 的次数为 m，

则 $\gamma=\beta+\beta^{-1}$ 生成一个 $GF(2^m)$到 GF(2)上的一组最优正规基，其中，ß 是(2m+1)次本原单位根，记 $N=\{\gamma, \gamma^2, \ldots, \gamma^{2^{m-1}}\}=\{\beta+\beta^{-1}, \beta^2+\beta^{-2}, \ldots, \beta^m+\beta^{-m}\}$，称 N 是 $GF(2^m)$ 到 GF(2)上的一组 II 型最优正规基。

下面我们介绍一种特殊正规基的构造

3. T 型高斯正规基的构造

更一般地，可推广得到 T 型高斯最优正规基的定义。

定义 9：令 q 是一个素数或者素数的幂，m，T 是正整数，Tm+1 是一个素数，且不整除 q。设 a 是 GF(q)的某个扩域上的 Tm+1 次本原单位根，γ 是 GF(Tm+1)中 T 次本原单位根。则称

$\beta=\sum_{l=0}^{m-1}\alpha^{j^l}$

为 GF(q)上的(m，T)型高斯周期(Gauss Period)。

由(m，T)型高斯周期(Gauss Period)诱导的正规基 $N=\{\beta, \beta^q, \beta^{q^2}, \ldots, \beta^{q^{m-1}}\}$ 称为 T 型高斯正规基(Gaussian Normal Bases，GNB)。

高斯正规基(GNB)是一类特殊的正规基，其上的有限域乘法运算尤其有效，因此得到大量的研究，具体可参见文献[92]。

众所周知，对任意正整数 m，$GF(2^m)$的正规基总存在，但是其 GNB 并非一定存在，仅当 $0 \neq m \bmod 8$ 时，才存在至少一个 GNB。对给定的 T 和 m，则至多存在一个 T 型 GNB。对给定的 m，如果存在多个 GNB，则 T 值越小，相应的有限域乘法越高效。

定理 7：T 型 GNB 的存在条件(见[92])

$GF(2^m)$的 T 型 GNB 存在，当且仅当

- p=Tm+1 是素数；
- gcd(Tm/k，m)=1，其中，k 是 2 模 p 的乘法阶。

➢ T 型高斯正规基(GNB)的运算

使用 T 型 GNB 表示域元素的运算定义如下：

假设 a =(a_{m-1}，…，a_1，a_0)和 b =(b_{m-1}，…，b_1，b_0)，则

加法：a + b = c =(c_{m-1}，…，c_1，c_0)，$c_i=a_i+b_i \bmod 2$，则加法按对应位异或。

平方：$a^2=(\sum_{i=0}^{m-1}\alpha_i\beta^i)^2=\sum_{i=0}^{m-1}\alpha_i\beta^{2i}=\sum_{i=0}^{m-1}(a_{i-1} \bmod m)\beta^{2^i}$ =(a_0，a_{m-1}，…，a_1)即平方运算只是一个简单的循环移位。

乘法：令 p=Tm+1，u∈GF(p)是阶为 T 的元素。定义序列 F(1)，F(2)，…，F(p-1)如下：

$$F(2^iu^j \bmod p)=i,\quad 0=i=m-1,\quad 0=j=T-1$$

对每个 l，0=l=m-1，定义 A_l 和 B_l 如下：

$$A_l=\sum_{k=1}^{p-2}\alpha_{F(k+1)+l}b_{F(p-k)+l}$$

和

$$B_l=\sum_{k=1}^{m/2}(a_{k+l-1}b_{m/2+k+p-1}+a_{m/2+k+p-1}b_{k+l-1})+A_l$$

则 ab = c =(c_{m-1}，…，c_1，c_0)，其中，当 T 是偶数时，$c_l=A_l$，当 T 是奇数时，$c_l=B_l$(0=l=m-1，下标是模 m 的余数)。

使用正规基表示 GF(2^m)上元素的相应运算算法较好的综述文章，请参见[93]，具体的软硬件执行参见[4，5]。

3.4.3 正规基和多项式基间的关系

设$f(x)=x^m+f_{m-1}x^{m-1}+\ldots+f_1x+f_0$是GF(2)上次数为m的不可约多项式，把f(x)作为约化多项式可定义有限域GF(2^m)为：

$$GF(2^m)=\{a_{m-1}x^{m-1}+\ldots+a_1x+a_0|a_i\in\{0,\ 1\}\}$$

设N = {a_0，a_1，…，a_{m-1}}是GF(2^m)在GF(2)上的一组基，如果存在ß∈GF(2^m)，使得$a_i=ß^i$，0=i=m-1，则称N为一组多项式基。如果存在γ∈GF(2^m)，使得$a_i=\gamma^{r^i}$，0=i=m-1，则称N为一组正规多项式基。当取定一组基后，可以把GF(2^m)中的元素$a_0a_0+a_1a_1+\ldots+a_{m-1}a_{m-1}$与GF(2)上的m维向量($a_0$，$a_1$，$a_3$，…，$a_{m-1}$)等同起来。

特别地，根据定理6，II型最优正规基与多项式基具有如下的转换形式。因为γ=β+β^{-1}，且ß是(2m + 1)次本原单位根，于是有

$$\gamma^{r^t}=(ß+ß^{-1})^{2^t}=ß^{2^t}+ß^{-2^t}=ß^t+ß^{-t}(0<t<p=2m+1,\ 2^s=t(\bmod p))$$

而且，当m + 1=t=2m，可以用(p-t)代替t，从而，得到下面的定理。

定理8：[92]下面两个基集合

$$M=\{ß+ß^{-1},\quad ß^2+ß^{-2},\ \ldots,\ ß^{2^{m-1}}+ß^{-2^{m-1}}\}$$

和

$$N = \{ß+ß^{-1}, ß^2+ß^{-2}, \ldots, ß^m+ß^{-m}\}$$

是等价的。

假设$a=(a_{m-1}, \ldots, a_1, a_0)$关于基M和N的表示分别为$(a_{m-1}, \ldots, a_1, a_0)$和$(a_{m-1}', \ldots, a_1', a_0')$，则由定理 8 和 $ß^{2m+1}=1$，可得到 a_j 和 a_i'之间的关系为 $a_j = a_i'$，其中，当 $k\in[1, m]$时，$j = k$，当 $k\in[m+1, 2m]$时，$j = (2m+1)-k$，$k = 2^{i-1}(\mathrm{mod}\ 2m+1)(i=1, 2, \ldots, m)$。于是，可将域元素的正规基表示和多项式基表示进行相互转换。

第四章 二元域求逆模块的两种改进

求逆是有限域基本运算中效率最差的，即使是设计专用硬件，其效率也远差于其他运算，有效提高求逆模块就是从根本上提高系统的效率。

本文提出的硬件求逆方案着重从提高执行时钟频率和压缩执行周期数两个方面入手，综合应用多处改进，有效提高了求逆效率。

4.1 现有求逆方法简介

求逆算法根据原理可以分为应用费尔马定理和欧几里德算法及变种两大类。一般认为，前者方法规整，适合硬件实现；后者复杂，效率高，适合软件实现。

4.1.1 费尔马定理

根据费尔马定理有以下结论：

有限域元素 ß 的阶为 r，k = -1 mod r，则元素 ß 的逆元 $ß^{-1} = ß^{k}$。特别地，在二元域 GF(2^m)中有 $\beta^{-1} = \beta^{2^m-2}$。

根据以上结论，求逆元只要反复做乘法即可，适合硬件求逆使用[96]。优点是硬件结构有规律，简单。但是费尔马定理要做数目可观次的乘法（乘方）。效率不高。

➢ 优化求逆算法 OIA

Itoh[39]指出，完成一次求逆运算所需乘法次数最少为 $i(m) = [\log_2(m-1)] + w(m-1) - 1$ 次，其中 w(m - 1)表示域 GF(2^m)中二进制数表示中 1 的个数。在此基础上，Sand ho oh 和 Chang Han Kim[40]提出优化求逆算法 OIA (Optimal Inverse Algorithm)，该算法求逆主要用到乘法和平方运算，乘法次数和 Itoh 的算法相同。下面给出二元域 GF(2^m)上优化求逆算法 OIA。

优化求逆算法 OIA：

1 初始化 t 为 m - 1，x 为 1，u 为 a^2；

2 While t > 0 do

2.1 While t[0] =0 and t > 1 do//t[0]为 t 最低位

2.1.1 t 从高位向低位移位一位；

2.1.2 置 u 的值为 $u \times u^{2^t}$；

2.2 置 x 的值为 x×u；

2.3 If t = 1，跳转到第 3 步结束；

else 置 u 的值为 u^2；

2.4 置 t[0]位为 0；

3 输出 x。

➢ 费尔马定理求逆的脉动式结构[41]

脉动式电路是研究的一个热点，其根本是在芯片集成度大大提高的实际前提下，利用大量简单重复单元，形成多级结构，数据源源不断的流入，又源源不断的流出。处理大量连续数据时，平均看来其时间复杂度为 O(1)，十分有吸引力。下面给出费尔马定理求逆的一种脉动式结构的基本原理，由于篇幅限制，无法详细描述。

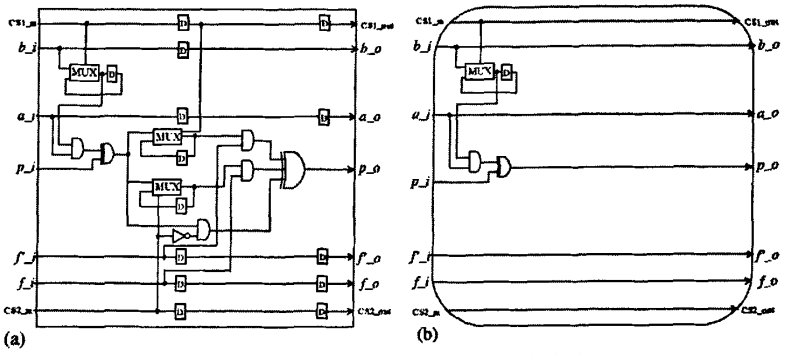


图 4-1 (a)PE5 模块内部结构 (b)PE6 模块内部结构

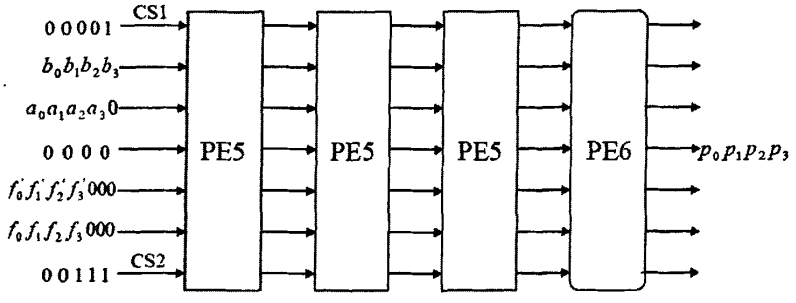


图 4-2 GF(2^4)上串行 AB^2 脉动结构

AB^2 结构求逆模块的原理是将费尔马定理变形如下：

$B^{-1} = B^{2^m-2} = [B[[...[B(B)^2]^{2...}]^2]^2]^2$

于是，求逆过程可以转化为：

1 P = B;

2 for i = m-2 to 1

2.1 $P = PB^2$;

2.2 $P = P^2$;

3 输出 P

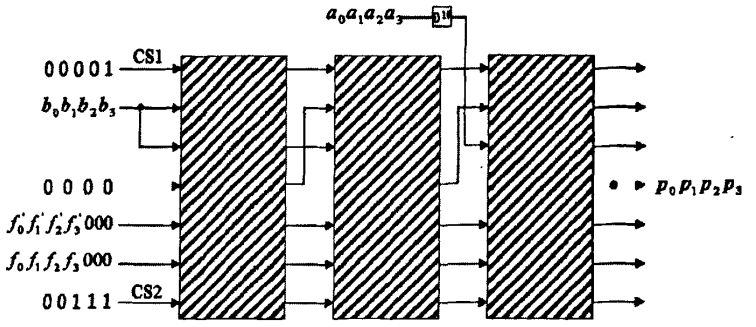


图 4-3 GF(2^4)上的 AB^2 串行脉动求逆系统结构

图 4-1，图 4-2，图 4-3 给出 GF(2^4)上求逆的硬件结构图。只要 m-1 个 AB^2 结构并列，就能完成求逆运算。

以上结构简单，适合大规模集成电路，效率很高，只是随着 m 的加大硬件开销和功耗将非常可观。在某些场合不合适。

使用正规基表示域元素后，二元域上的平方可以用一次循环移位完成[96]。同时，一般的乘法也可以免去约简的操作。特别是在例如 GF(2^{155})、GF(2^{233})等元素能以最佳正规基[62]表示的二元域，乘法器的硬件开销相对于计算的复杂性已经很小。即便如此，应用费尔马定理求逆，效率还是较差。

4.1.2 扩展欧几里得算法及其变型

扩展欧几里得算法过程比较复杂，不容易统一控制。但其优势是运算量小，速度快。本文提出的算法也是改进自扩展欧几里得算法。国际国内也不断出现将扩展欧几里得算法变形后用硬件实现的研究成果。比较有代表性的有以下一些。

- Montgomery 逆(素域 F_p 上)[42]

Montgomery 方法是经典的，用途很广的方法。其基本思想是对于特别选择的 R，用低成本的运算代替除运算。算法计算 a^{-1} 得到 $a^{-1}2^k \bmod p$，$k \in [n, 2n]$。

输入：奇整数 $p > 2$，$a \in [1, p-1]$，$n = \text{lb}\, p$ 上界。

输出：无逆，或(x, k)，其中 $n \leqq k \leqq 2n$ 且 $x = a^{-1}2^k \bmod p$.

1 $u = a$，$v = p$，$x_1 = 1$，$x_2 = 0$，$k = 0$；

2 当 $v > 0$ 时，重复执行

2.1 若 v 是偶数，则 $v = v/2$，$x_1 = 2x_1$；

否则，若 u 是偶数，则 $u = u/2$，$x_2 = 2x_2$；

否则，若 $v \geqq u$，则 $v = (v - u)/2$，$x_2 = x_2 + x_1$，$x1 = 2x1$；

否则，$u = (u - v)/2$，$x_1 = x_2 + x_1$ ，$x_2 = 2x_2$；

2.2 $k = k + 1$；

3 若 $u \neq 1$，则返回“无逆”。

4　若 $x_1 > p$，则 $x_1 = x_1 - p$，返回(x_1, k)。

➢ Brunne 等提出的二元域 GF(2^m)求逆算法[49]

```
F = F(x); // F(x)为域多项式
S = F(x); V = 0; U = 1;
R = B(x); // B(x)为待求逆元素
Δ = 0; //Δ = deg S – deg R，deg S 初值为 m，deg R 初值假设为 m
for i = 1 to 2m do
if r[m] = 0 then
    R = x·R; U = (x·U) mod F; //其实是移位操作
    Δ += 1;
else
    if s[m] = 1 then
        S = S – R; V = (V-U) mod F;
end
S = x·R;
if Δ = 0 then
    R←→S; U←→V;
    U = (x·U) mod F; Δ = 1;
else
    U = U/x mod F; Δ -= 1;
end
end
输出 U 或者 V。
```

该算法比较经典，综合效率、开销和功耗等多方面总体较优，笔者所了解近年来的算法很少综合性能有较大幅度超越此算法者。但单独某一方面则不是最优，在某一方面要求特别苛刻时不适用。

➢ Brunne 等提出的 GF(2^m)求逆算法的改进[50]

通过此改进，此算法可以适应多个二元域，但其他性能略有下降。

```
R = B(x)x^(m-k); S = F(x)x^(m-k); U = x^(m-k); ent=0;
for i = 1 to 2m do
if(cnt = 0) then (cnt 零检测)
   f = 0;
else
   f = 1;
end
case{rm, sm, f}
```

when 0xx: R = xR; S = ; U = xU mod F(x); V = V; cnt = cnt + 1;

when 100: R = xS; S = R; U = xV mod F(x); V = U; cnt = cnt + 1;

when 101: R = R; S = xS; U = U/x mod F(x); V = V; cnt = cnt + 1;

when 110: R = (S - R); S = R; U = (V - U) mod F(x); V = V - U;

cnt = ent + 1;

when 111: R = R; S = x(S - R); U = U/x mod F(x); V = V - U; cnt = cnt - 1;

end case;

if(I = deg2)then(控制循环次数)

break;

end for;

其中，m 为此逆元模块所支持的最大多项式的度。对于所有 k < m 的二元域 GF(2^k)模块均适用。用一个寄存器 deg 2 保存循环次数，其值等于 2k。当循环的次数等于 2k 时，逆元计算完成。修改算法的初始值 U(x) = A(x)xm-k 后，它支持有限域的除法。当 A(x)==1 时，计算逆元 B(x)-1；当 A(x) ≠ 1 时，计算除法 A(x)/B(x)。适应性广的模块在某些情况下很有用处，但在大多数系统中有限域完全可以确定下来，这样既不影响安全性也不会造成不方便。

➢ 脉动阵列求逆模块

上文曾提到，脉动阵列结构具有优良的时间复杂度，也特别适合大规模集成电路应用。扩展欧几里德算法的硬件实现中，脉动阵列方案近年来有很多研究[55][54]。下面给出文献[55]中体现出脉动阵列特征的欧几里得算法变形。

整个阵列结构是 i 行 j 列的同等微模块互联而成，数据从阵列的一个边进入，向对边脉动流动，而在对边既形成运算结果。以下算法中出现的变量在每个微模块中存在一份，d 则每行存在一份。为了表达方便，先定义两个符号：

$Y_{(2m-1)(j-1)}$定义为 ADD(j)，ADD(j)∧(d_{j-1} < 0)定义为 SWAP(j)。

输入元素 a，域多项式 g；输出 a^{-1}；

1) 初始化：d_0= -1；

(y_{i0}，w_{i0})初值分三种情况：当 2m - 1 ≧ i ≧ m 时取(a_{i-m}，g_{i-m})，

当 m - 1 ≧ i ≧ 1 时取(0，0)

当 i = 0 时取(1，0)；

2) for j = 1，2，...，m – 1 do

for i = 2m-1，2m -2，...，2，1do

y_{ij} 分两种情况变换：

当 ADD(j) = 0，取值 $y_{(i-1)(j-1)}$，

当 ADD(j) = 1，取值 $y_{(i-1)(j-1)} \oplus w_{i(j-1)}$；

w_{ij} 分两种情况变换：

当 SWAP(j) = 0，取值 $w_{i(j-1)}$，

当 SWAP (j) = 1，取值 $y_{(i-1)(j-1)}$；

d(j) 分两种情况变换：

当 SWAP(j) = 0，取值 d(j-1) - 1，

当 SWAP (j) = 1，取值-d(j-1) - 1；

3) for j = m，m+1，2m-1 do

for i = 2m-1，2m-2，…，j–m+1

y_{ij} 分两种情况变换：

当 ADD(j) = 0，取值 $y_{(i-1)(j-1)}$，

当 ADD(j) = 1，取值 $y_{(i-1)(j-1)} \oplus w_{i(j-1)}$；

w_{ij} 分两种情况变换：

当 SWAP(j) = 0，取值 $w_{i(j-1)}$，

当 SWAP (j) = 1，取值 $y_{(i-1)(j-1)}$；

d(j) 分两种情况变换：

当 SWAP(j) = 0，取值 d(j-1) - 1，

当 SWAP (j) = 1，取值-d(j-1) - 1；

4) 输出 $a^{-1}{}_i = w_{(m+i)(2m-1)}$，其中 i = 0，1，…，m-1

4.2 殆逆算法的硬件实现及优化

殆逆算法[43]是扩展欧几里得算法的一种改进，收敛速度更快。在软件应用中是最好的选择之一，但未见有优秀的硬件移植成果出现。本文将尽力开发其潜能，着眼于执行效率，提出改进方案。最后的试验证明，本文提出的两种改进方案在性能上赶上或超过当前其他优秀方案。尤其是算法 2 的性能非常优秀，远远超过其他算法。

4.2.1 殆逆算法简介及选用理由

求 a 的逆就是求满足 $a \times b = x^k \bmod p(x)$的 b 和 k，则 $a = b \times x^{-k}$。步骤 $a = b \times x^{-k}$相当于将乘法操作分配出来，整个计算分为关联很小的两阶段。其中前一阶段不含最复杂的乘法运算，后一阶段只含乘法，两段特点清晰明了，不混杂大大降低了模块的复杂度。复杂度降低使得降低门延迟更有可能。两阶段的另一个好处在于欧几里德求逆本身的特点，欧几里德算法是一个迭代过程，即数据流要从输出端反馈回输入端多次，这给布局布线带来了麻烦，一般来说反馈会带来很长的传输路径，从而造成很大的传输延迟，实验表明传输延迟和门延迟比例为大约 9：1。所以，减低传输延迟，对整体时钟频率的提高意义更大。而殆逆算法的两段性使得数据只在前一段反馈循环，而大规模的乘法又只出现在后一段，于是前一段规模的大大降低可以明显缩短反馈距离，从而可能降低传输延迟。模块实现后的实验结果符合这一推断。

二元域 $GF(2^m)$上殆逆算法：

1　初始化整数 k = 0 多项式的序列 B = 1，C = 0，F = a，G = p(x)；

2　循环执行：

2.1 当 F 偶数（最低位为 0），反复做 F = F/x，C = C × x，k = k + 1 直到 F 为奇数；

2.2 如果 F = 1 结束，计算 $a = b \times x^{-k}$；

2.3 如果 deg(F) < deg(G)，交换 F、G，交换 B、C；

2.4 F = F ⊕ G，B = B ⊕ C。继续循环步骤 2。

其中 deg(F)是 F 的度数，定义为多项式 F 最高项次数，也就是序列最高位 1 所在的位数。F 为偶数时，F/x 是移位操作。C×x 也是移位操作。计算 b 和 k 的过程可以分解为一系列异或和移位的操作，适合硬件实现。这也正是二元域和殆逆算法结合的好处之一，更印证了乘法摘除后，前一阶段规模大大降低的猜想。

4.2.2 利用度数相关性提高效率

经过观察比较可以发现，整个运算过程中比较费时的操作就是比较 deg(F)和 deg(G)。比较这两个参数需要同时从 F 和 G 的高位同步依次移出 F 和 G 各位作比较，直到移出的两位数据包含 1，说明 F 和 G 的某一个度数所在位已经找到，才可以得出结论，先发现 1 的那个自然是度数较大者。殆逆算法每轮循环都要比较度数，而且计算过程中由于 F 的移位是 F = F/x，从高位往低位移，度数变化趋势是越来越小，而 G 值是从 F 交换得来，变化趋势和 F 相同，那么到计算后期，因为度数已经很小，最高位 1 在低位出现，而扫描仍要从最高位开始。在公钥密码系统中，多使用千位以上的大整数，即使在每比特安全度较高的椭圆曲线密码系统中，也要求字长在 200 位以上。于是在实际的密码系统中，一次循环就要在比较操作上使用几百个周期，可以说大部分时间都消耗在度数的比较上。

经过观察我们发现，步骤 2.3 和 2.4 中，无论是否交换，F 都取到 F⊕G，而 G 取到 F 和 G 中度数比较小的一个(进一步印证度数会变小)。于是，对于 deg(G)，就取到上一组 deg(F)和 deg(G)中较小的一个；由于 F⊕G 是按位异或没有进位，deg(F)是不会因为 F⊕G 而变大，对于 deg(F)，当上一组 deg(F)和 deg(G)不相等时，异或后会变成上一组 deg(F)和 deg(G)中较大的一个；如果度数相等，那么异或后的 deg(F)就不可预计了，但是两个度数相同的序列异或，二者最高位 1 起码是要变成 0 的，新度数一定小于原度数。

总结以上分析的相关性，如表 4-1。其中 deg'(f)表示步骤 2.4 交换相加之后新度数，deg(f)表示上一组度数。

表 4-1 步骤 2.4 前后度数相关性

	deg'(f)	deg'(g)
deg(f)>deg(g)	deg(f)	deg(g)
deg(f)<deg(g)	deg(g)	deg(f)
deg(f)=deg(g)	< deg(f)	deg(g)

根据表 4-1，我们可以增加两个寄存器记录 deg(F)和 deg(G)，在步骤 2.3 中只要上一组 deg(F)≠deg(G)（多数情况下如此），deg(F)和 deg(G)随着 F 和 G 交换或不交换，新的度数就求出来了。但是度数相等时还要通过移位求得度数，而这种情况出现次数较少。

那么度数相等时 deg(F)的相关性如何利用呢？这种情况下虽然无法得到确切的新度数，但度数是减小的。图 4-4 表示步骤 2.4 相加操作前后寄存器 F 状态的变化和 deg(F)寄存器减 1 计数求新度数的过程。相加(异或)后，deg(F)寄存器还保存着旧的度数，它会指向 F 寄存器中异或之前的最高位 1 那一位，而这一位在相加时从 1 变为 0，于是高于这一位(包括这一位)的各位可以确定为 0。

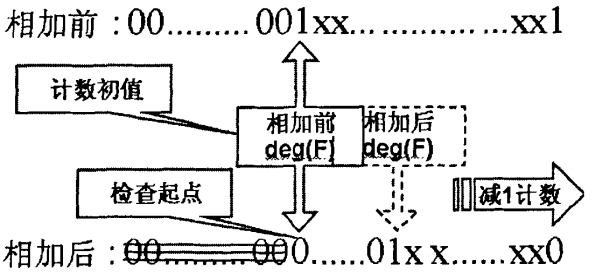


图 4-4 相加前后 F 的变化和 deg(F)变化规律

于是我们可以以 deg(F)保存的旧度数减 1 作为计数初值和搜索起点，在寄存器 F 中向低位搜索，每搜索一位 deg(F)减 1 计数，直到搜索到 1，从而找到新度数。减 1 是因为可以确定异或后那一位为 0。这样的优势是在计算后期，求度数无须检查高位大段的 0。检查的各位都是从未检查过的，避免重复，那么可以预计，整个运算花在检查上的周期数就不会超过 2m，也就是最差情况，度数一直保持相等，F 和 G 的各位都被检查过。求度数不从最高位开始检查还带来一个好处就是，让检查过程和 m 无关，于是无论 m 是多少，检查操作都是没有区别的，这将对后文要讨论的模块通用性很有益处。

4.2.3 保证每个周期度数都有减小

从上一节可以看出，整个求逆过程就是度数减小的过程，计算结束时 deg(F)为 0。基于此特征，将 deg(F)=0 作为结束条件自然比考察 F=1 开销小得多，因为寄存器 F 的长度远大于 deg(F)的长度。并且比较 F 和 m 位二进制数 1 是一个和 m 相关的操作，不同的 m 时比较 F 和 1 操作是不一样的。这会影响后文讨论的模块通用性。

度数减小最多的情况是什么呢？下面给出推导。因为 G 的初值是 m 次不可约多项式，deg(G)的初值一定是 m；deg(F)的初值最大的情况是 m-1。计算结束

时 F = 1，所以最后 deg(F) = 0；而 deg(G)没有机会为 0，最小 deg(G) = 1。假设 deg(F)初始化时取最大初值 m-1，计算结束后 deg(G)得到最小终值 1，此时度数减小最多，为：

$n_{max} = [deg_{max}(F) + deg_{max}(G)] - [deg_{min}(F) + deg_{min}(G)] = [(m-1) + (m)] - [0+1] = 2m$

从上式可以推知，因为 deg(F)控制运算结束，如果能做到每个时钟周期 deg(F)寄存器值都有减小，那么整个求逆过程使用的时钟周期将不大于 2m。需要说明两点：一、度数的减小只能发生在 deg(F)上，而 deg(G)的减小要在 deg(G)值换入寄存器 deg(F)时发生；二、判断度数是否减小要以 deg(F)寄存器为准，因为 deg(F)寄存器的值控制算法结束，影响执行周期。

考察殆逆算法主循环，其中步骤 2.1 向低位移位，每执行一次度数减 1，实现了度数减少。步骤 2.2 是结束，度数已经减少完毕，不考虑。

步骤 2.3 中，如果上一组度数不等，是没有度数减少的。但是在少数情况下，也就是上一对度数相等或这初始化时，必需求得新度数。根据上一小节分析，异或后度数至少减 1。并且求度数的方法前文介绍过，就是检查 F 中当前被 deg(F)指向那一位，如果是 1 说明当前 deg(F)的值正确，否则 deg(F)减 1 继续检查。由于 deg(F)寄存器值减小，于是计算度数也可以看作是度数有减小。大多数情况也就是度数不等时，我们如何处理呢？根据硬件特点，我们可以将步骤 2.3 中的比较和交换纳入步骤 2.4 中，而这一步只是在度数不确定时移位求出度数，可以预料这一步是较少执行到的。新的步骤 2.4 变成根据度数比较结果确定 G、deg(F)、deg(G)三个寄存器的新值是交换得来还是不变。而 F 的新值确定为 F⊕G。

表 4-2 对度数的操作

情况	F[0]	F[deg(F)]	对 deg(F)的操作
移位且求度数	0	0	-2
只移位	0	1	-1
只求度数	1	0	-1
直接执行步骤 2.4	1	1	—

值得一提的是，只要新的 F 生成了，即一轮循环中，步骤 2.3 求度数 F 的移位和步骤 2.1 中 F 的移位可以合并执行。两个操作并行时要注意，移位和求度数都要求 deg(F)减 1。由于有些情况下度数减 2，整个求逆过程周期数可从 2m 进一步缩减。

我们总结所有情况下对度数的操作如表 4-2。

接下来考察步骤 2.4，按照上文约定，这一步包括步骤 2.3 的比较交换和步骤 2.4 原来的加法（异或）。当原度数不相等时，度数交换而值不减小；当原度数相等时，度数虽有减小，但新度数的不确定导致 deg(F)寄存器值无法立刻更

新，总的来说 deg(F)寄存器值是不变的，这被看作度数不变。于是执行这个步骤的周期没有完成减小度数的任务。为了解决这个问题，我们考察 G 和 F 的最低位。

G 初始化时取 m 对应的不可约多项式，最低位为 1；而另一种 G 值的更新发生在和 F 交换时，此时的 F 是经过步骤 2.1 生成的，最低位也一定为 1。于是送入 G 的数值最低位恒为 1。F 值在和 G 相加之前是经过步骤 2.1 生成的或者通过交换取自 G，最低位也为 1。于是，步骤 2.4 中 F 和 G 的相加（异或）产生的新 F 的最低位一定为 0。也就是交换相加的下一个周期一定得到偶数 F，即交换相加周期后一定跟随一个步骤 2.1 F 的移位周期。我们可以将这个移位和交换加法合并到同一个周期执行，要做的只是将相加的结果错位送给 F，完全没有开销的增加。但是这样我们就能实现在步骤 2.4 也有度数减小。实验证明，m=233 时，步骤 2.4 平均执行 200 次左右，那么这个改进将无成本压缩掉约 200 个周期。

4.2.4 改进算法一

根据以上优化，现在给出本文的第一个改进算法。

该算法对应着硬件实现，步骤 2 一轮循环可以在一个周期内完成，且无论运行哪个分支，度数都有减小。可以预计，算法应用于软件也会有较好效果，当然软件实现不必时刻判断结束条件 deg_f != 0。

算法中除了取 F[deg_f]，所有条件判断涉及的寄存器都不大于 $1+\log_2 m$，大容量寄存器的操作都是按位操作，也就是说算法中基本所有组合逻辑都是小规模，低延迟的。关键路径就是取出 F[deg_f]并以此控制运算。

算法步骤：

1. 初始化整数 k = 0 多项式的序列 B = 1，C = 0，F = a，G = p(x)，
   整数 deg_f = m-1，deg_g = m；
2. while(deg_f != 0)

```
{
  if(F[0] == 0)// F 的移位
  {
     deg_f = F[deg_f] == 1 ? deg_f-1 : deg_f-2; F = F>>1, C = C<<1, k =
     k+1;
  }
  else
  {
     if(F[deg_f] == 0)   deg_f = deg_f-1; //求度数
     else
```

```
        {
            if(deg_f < deg_g)
            {
                F⇔G，B⇔C；deg_f ⇔deg_g；//交换
            }
            F= (F>>1)+( G>>1)，B=B+C；      //加法
            If(deg_f ==deg_g)   deg_f -=1；//……*
            deg_f -=1；
        }
    }
}
```

3. a=b×x-k

第二步中标*符号操作是可选的。这一步的含义是，当上一组度数相等时，不仅 F 和 G 的最低位都为 1，并且因为上一组度数相等，双方最高位 1 所在位也相同。于是可以肯定，此种情况下最低位和最高位 1 所在位相加后都从 1 变为 0，度数可以预计会减小 2。体现在算法中就是，如果度数相等 deg_f-2，不相等 deg_f-1。

但是，在按照此方案设计状态机时，判断加法周期后的状态转移方向需要检查 F 和 G 相加结果的第 deg_f-1 位，这相当于在上文提到的关键路径上进一步增加组合逻辑，会影响整个模块的时钟频率。而实验统计此方案可以减少约 5%的周期数，而关键路径延迟却增大约 22%，综合考虑效率是降低的，且还要增加硬件开销。于是后文的实验结果都是实现去掉步骤*的算法得到的。

之所以讨论这个效果并不好的做法，是因为这个做法在软件应用时还是有效的。

4.2.5 算法一性能分析和实验结果

➢ 算法通用性

一般情况下，硬件求逆模块只适用于单个有限域，例如 GF(2^{233})，GF(2^{191})等等。但是本算法灵活性稍强。前文曾两次提及设计避免和 m 相关的操作，于是我们得到了没有和 m 相关操作的主循环。不和 m 相关则保证任意 m 时操作都完全相同。假设应用此算法的模块寄存器 F 宽度为 w，那么主循环无须改动可直接适用于 m 不大于 w 的任意二元域 GF(2^m)上逆元。对于各二元域不同之处在于初始化时 G 要取 m 对应的不可约多项式 p(x)，deg_f 和 deg_g 也分别取 m-1 和 m；计算结束后取 B 中数据直接忽略 m 位以上的高位。于是，模块对于 m 不大于 w 的各二元域是通用的。

这种特征非常适用于实现处理器的指令，所以本算法在实现强化安全应用

的专用处理器时可以发挥很大作用。

表 4-3 性能的理论分析对比

	文献[49]	文献[48]	文献[53]	算法 1
关键门延迟	O(lb(lbm))	O(m)	O(1)	O(lbm)
面积复杂度	O(m)	O(m)	O(mlbm)	O(m)
周期数/逆	2m	2m+3~3m+2	2m	2m~(m+1)/2
通用性	否	否	是	可配置

表 4-3 给出了本算法和其他基于欧几里德算法的方法在关键门延迟、面积复杂度、每次计算消耗的周期和通用性几个方面的比较。其中 lb$x$ 表示 $\log_2 x$。表中 m 表示模块计算的字长即密钥长度。可见本文方案理论上综合门延迟和开销和文献[49]相当。

前文曾提到，电路延迟除了关键门延迟还包括连线延迟。欧几里德算法的变种都包含数据的大范围复杂反馈循环，综合工具布线时优化连线延迟比较困难；加之公钥密码算法规模大，增大了传输距离，连线延迟成为影响时钟频率的最重要因素。本文方案分为前后两级模块，划分后模块规模和复杂度也有所缩减，数据的反馈循环的距离缩短，连线延迟降低。

表 4-4 时钟频率对比

m\器件	xcv2000e	EP1S10B672C6	EP2S130F780C4
132	59.5Mhz	134.08Mhz	194.33Mhz
174	56.5 Mhz	129.08Mhz	181.98Mhz
193	54.9 Mhz	128.24Mhz	189.54Mhz
233	—	128.01Mhz	188.11Mhz

表 4-4 给出文献[50]实现文献[49]算法达到的时钟频率和本方案时钟频率的对比。文献[50]使用的器件是 xilinx 公司的 xcv2000e，本文使用档次和速度级别都与其相当的 altera 公司的器件 EP1S10B672C6 做对比，表 4 最后一列是使用当前的主流器件 EP2S130F780C4 的时序分析结果。最后一行中 233 是很多国际标准推荐的实现椭圆曲线密码系统的 m 值，鉴于椭圆曲线密码体制各方面的优势，特别给出这个结果以便对照。

除文献[50]外，文献[51]也是近期出现的比较优秀的求逆模块方案。在 EP1S10B672C6 同系列器件上，m=83 时能达到 100mHz 的频率。在 m 远小的情况下频率尚和本文方案有差距。

由于器件能力的上限，结果并不能严格符合理论分析。正如上文分析，由于本文方案分前后两步操作，数据传输范围小，容易优化连线延迟，于是能达到更高的时钟频率。

4.2.6 双向移位结构

算法一在执行效率上已经相当优秀，但是其关键路径取 F[deg_f]作为控制信号的时间复杂度仍显较高，本节将寻找一个整体结构来解决这个问题。

➢ 引入反向移位

问题出在当 deg(F) = deg(G)时，新度数还是要移位求出时。

从另一个角度看待表 4-1 中度数的规律可以得到另一个规律，就是 F 和 G 中必然存在一段从最高位开始的全 0 子序列，可以称作高端 0 序列，记作 Z_F 和 Z_G。记 F 和 G 中任意位分别为 f(n)、g(n)，其中 n 表示此位所在位置即下标，运算过程中有：

对于任意 n，若 $f(n) \in Z_F$ 且 $g(n) \in Z_G$，则 f(n) = 0，g(n) = 0。

以上形式化的描述表示，运算过程中，F 和 G 高端 0 序列的公共部分恒为 0。根据此规律我们可以引入 F 的反向移位结构。因为公共部分已经确定为 0，存储已经没有意义，于是我们可以在原算法将 F 和 G 从高向低移位的基础上增加 F_inv 和 G_inv 寄存器，对其进行从低向高的移位，将高端 0 序列的公共部分全部移除，剩下的部分我们称其为有效序列。度数较大的有效序列最高位必然是 1，根据这个最高位就可以得出度数比较结果。无论 m 有多大，比较最高位需要的电路都是几乎最简单的，性能自然也是最好的。

进一步，将 F 和 G 的各种操作也同步施加给 F_inv 和 G_inv，那么有效序列除了在寄存器中位置和 F、G 不同外，内容完全相同。这样只要连续反向移出公共的高端 0 序列，就可以连续得出度数比较结果。如此反向移位，每次比较仍然不必重新移位前几轮已经移出去的内容即公共高端 0 序列，大大节约了时间。也替代了取 F[deg_f]，即用 deg_f 来记录当前计数起点的工作，使本来的关键路径延迟降低到 O(1)。

➢ 双向移位结构

将改进算法一的主循环分为以下四个阶段，其中每阶段都可以分解为若干操作相同的时钟周期：

1.F 是偶数时正向移位（F_inv 同步正向移位）；
2.F_inv 和 G_inv 反向移动；
3.若上一轮度数相等求新度数；
4.结束判断，若没结束，F 和 G 交换相加。

其中前三阶段操作不同的寄存器，可以完全并行。而第 4 步涉及所有寄存器，需要等待前三阶段全部完成才能进行，相当于一个同步的步骤。

下面将第 2 步和第 1 步并行，也就是将 F_inv 的正向移位和反向移位同时进行。分析此时各寄存器状态，由于上一轮留下的 G_inv 和 F_inv 的最高位一定有一个 1，表示高端 0 序列已经移除完，而 F_inv 和偶数 F 同步正向移位又会造成移除未完而需要反向移位。所以恰好可让反向移位和正向移位抵消，则视 G_inv 最高位是否为 0 有两种不同的操作：为 0 时，正向移位使公共高端 0

序列未移除完，则 F_inv 固定，G_inv 反向移位；反之说明已经移除完，则 G_inv 固定 F_inv 正向移位。由以上分析也可以看出，在正反向并行情况下，两种操作之后 F_inv 中 G_inv 都不含有公共高端 0 序列，也就是第 1 步结束天然的说明前两步结束。

第 3 步求度数需要增加一个寄存器 F_deg，在交换相加时如果度数相等，则将 F_inv 中数据送入 F_deg 移位计数求度数。送入 F_deg 的数据是同度数序列异或的结果，最高位必然是 0，此后的移位和计数在最高位出现 1 时结束。

求度数和第 1 步 F 并行时，正向移位和求度数同时要求 deg_f 寄存器减 1。所以，F_deg 最高位为 0 和 F 最低位为 0 两个条件仅一个有效 deg_f 减 1，同时有效则 deg_f 减 2。度数计算完成后 F_deg 最高位一直保持状态 1，求度数工作自然停滞；直到下一次度数相等时送入最高位为 0 的数据，求度数工作又可自动开始。

第 2 步和求度数没有关系，并行时二者无相互影响。于是前三步就并行起来了，从而形成了正向反向移位并行的双向移位结构。第 4 步和前三步是准备和执行的关系，自然不能并行。而第 4 步加法结果仍然错位送回 F，所以反向移位的 F_inv、求度数的 F_deg 也要进行相应的错位。

从前文可以看出，每周期 deg_f 最多可以减 2。按照上文分析方法，假设计算结束时 deg_g 保持初值 m，最少的度数减小为：

$n_{min} = [deg_{max}(F) + deg_{max}(G)] - [deg_{min}(F) + deg_{max}(G)] = [(m-1) + (m)] - [0+m] = m-1$

在此基础上进一步假设，每个周期度数减小 2，则求逆前级使用的时钟周期最少为 $(m-1)/2$。

4.2.7 改进算法二的硬件结构

图 4-5 上方是整个求逆模块的结构，下方给出了三种 cell 的内部结构。从图中可以看出，门延迟最大的逻辑是 deg_f counter，所以整个电路的门延迟非常小；控制模块需要的信号都与其相邻，从而使传播延迟也达到最小。

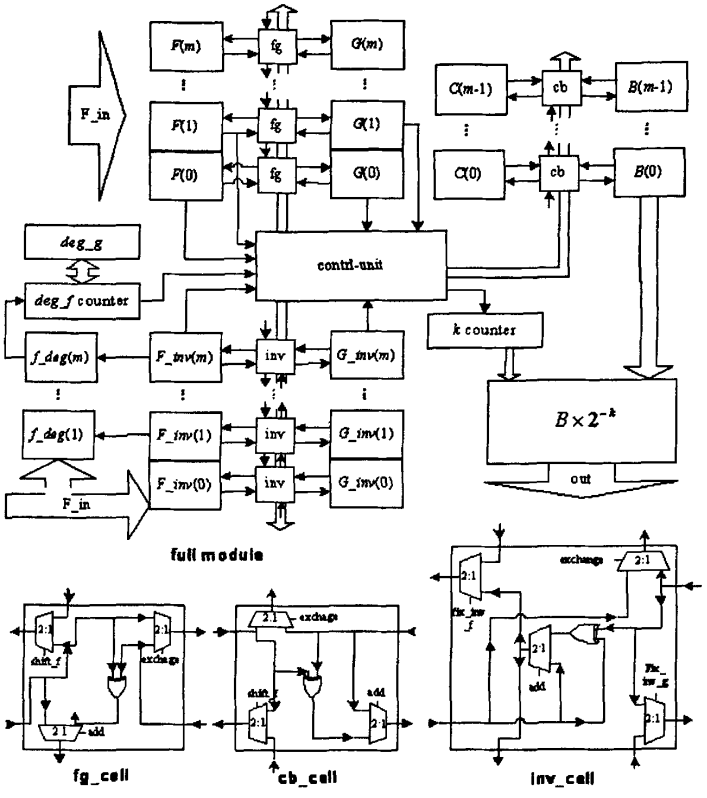


图 4-5 二元域双向移位殆逆模块硬件结构

4.2.8 算法二性能分析和实验结果

表 4-5 性能开销的理论分析对比

	文献[49]	文献[48]	文献[53]	算法 2
关键门延迟	O(lb(lbm))	O(m)	O(1)	O(lb(lbm))
面积复杂度	O(m)	O(m)	O(m×lbm)	O(m)
周期数/逆	2m	2m+3~3m+2	m	(m-1)/2~2m
面积时间积	$O(m^2lb(lbm))$	$O(m^3)$	$O(m^2\ lbm)$	$O(m^2\ lb(lbm))$

表 4-5 给出了本文算法性能和开销两方面和国际上典型的基于欧几里德算法的优秀算法的比较，应用了国际上常用的面积时间积(AT-product)评估方法，可以看出本文算法综合性能和开销和文献[49]处于同一水平，而在时钟周期数方面优于文献[49]。

算法二在传输延迟控制方面和算法一保持同样水平，但是其关键路径从取 F[deg_f]变成了 deg_f counter 自身计数和比较，有了指数级提高。即使是现阶段军事级安全需求的 2048 位密钥，最大门延迟所在的关键路径 deg_f counter 也只有 11 位就够了，11 位的计数器和比较器 8086 时期的规模，其延迟可以忽略不计。表 4-6 给出了实际时钟频率的比较结果。

表 4-6 实际时钟频率

m\器件	xcv2000e	EP1S10B672C6	EP2S130F780C4
174	56.5 Mhz	204.08Mhz	270.71Mhz
191	54.9 Mhz	188.54Mhz	270.42Mhz
233	—	178.06Mhz	269.76Mhz
998	—	153.19Mhz	186.81 Mhz

表 4 最后一列给出在 Altera 公司的 EP2S130F780C4 上静态时序分析结果，结果由 Quartus II 7.0 工具得出。第一列 m 均对应最佳正规基，以配合性能最佳的乘法器。其中 233 对应最有前途的椭圆曲线密码体制，998 则大致说明算法在主流密码体制（RSA 等）中的作用。表中第二列仍然给出文献[50]在 xcv2000e 器件上实现文献[49]算法达到的频率，并且使用同速度等级的 EP1S10B672C6 与其对比，从结果看出频率确有很大提升。尤其是随着 m 的加大，时钟频率的下降缓慢的多了。这正印证了关键门延迟 O(lb(lbm))和就近取信号两个特点。

同样，文献[51]在 EP1S10B672C6 同系列器件上，m=83 时能达到 100mHz 的频率。在 m 远小的情况下频率尚和本文方案有极大差距。

4.3 本章小结

针对组成各种公钥密码系统的基础——求逆运算，本章在经典的殆逆算法基础上提出了两种用于硬件求逆方案，相比于原算法在周期数和延迟方面有相当的提高。方案 1 综合性能和开销两方面和国际上的经典优秀算法相当方，案 2 优于或略优于国际上多种经典优秀算法。加之得益于殆逆算法分阶段的先天优势，两方案尤其是方案 2 的实际性能则可达到较高的水平。由于求逆模块在密码计算各模块中复杂度最高，延迟最大，依据本章算法的模块延迟低的特点使其能够更好的配合公钥密码系统的其他低延迟模块；此外，算法在软件应用中也可以发挥作用；综上，该算法在高性能公钥密码系统中有较大实用价值。

第五章 二元域乘法模块的优化实现

除了求逆，有限域乘法运算也极大地影响各种密码算法的加/解密速度，设计时间和空间复杂性低的乘法运算算法和乘法器变得尤其重要。根据域元素的不同表示方法，乘法器可分为多项式基乘法器、对偶基乘法器和正规基乘法器。

由于二元域加法无进位的特点，二元域乘法也相对适合硬件实现，并且本文构造的求逆模块也是基于二元域，二元域乘法器能够和其良好配合。其中正规基乘法避免了复杂的模约减，进一步应用最佳正规基又有开销低的特点，本章选择实现了基于正规基的二元域乘法器，并通过实际测算大致给出最优的并行度。二元域加法就是简单的按位异或，三种基本运算中另外的乘除法实现之后， 就形成了完整的二元域底层运算模块，这为下一步组合更复杂的模块奠定了良好基础。

5.1 多项式基乘法器

假设 A(x)，B(x)，$f(x)\in GF(2)[x]$，$C(x)=A(x)\times B(x) \bmod f(x)$，则计算 C(x)的并行多项式基乘法器做早由 Bartee 和 Schneider[71]提出，其实现需要 GF(2)上的(m^3-m)个二值输入的 XG[97]，其中 m 是其依赖的不可约多项式的次数。多项式乘法器是乘法器的主流，经典方案很多，下面介绍其中一些。

5.1.1 Mastrovito 并行 PB 乘法器

Mastrovito[47][72]提出了一种并行多项式基乘法器。该乘法器由 f-network 和 IP(m)两部分构成，其对应的硬件体系结果见下一页图 5-1。

因为 $f(x)\in GF(2)[x]$，所以 f-network 的复杂性与使用的不可约多项式有关，因此我们仅根据 BSP 模型分析 IP(m)部分的空间复杂性和时间复杂性。IP(m)部分，首先是 b_j 和 $f_{i,j}$(j=0，…，m-1)相乘，需要 m 个 AG 在一个 T_A 时间步内完成，其中 T_A 表示一个 AG 的时间延迟；接着用(m-1)个 XG 在 $\lceil \log_2 m \rceil$ 步内完成 m 个乘积的加法，该计算过程的计算时间为 $\lceil \log_2 m \rceil Tx$，其中，Tx 表示一个 XG 的时间延迟。

综上所述，该并行乘法器需要 m(m-1)个 XG 和 m2 个 AG，其时间复杂性为 $T_A+\lceil \log_2 m \rceil Tx$。

5.1.2 基于特殊多项式的 Mastrovito 并行 PB 乘法器

Sunar 和 Koc[73]利用三项式，对 Mastrovito 算法提出了一个新的公式，指出其对应的乘法器需要(m^2-1)个 XG 和 m^2 个 AG。Halbutogullari 和 Koc[74]推广了 Sunar 和 Koc 的思想，对任意不可约多项式，找到了构造 Mastrovito 乘法器的方法。迄今为止，对这些特殊的多项式，就 XG 的个数和时间延迟来说，Halbutogullari 和 Koc 算法的时间复杂度是最低的。Zhang 和 Parhi[75]提出了一

种设计 Mastrovito 乘法器的对称方法，而且，他们将这种方法应用于设计改进的 Mastrovito 乘法方案[76]。对两类不可约五项式，他们提出了新的 Mastrovito 乘法器的复杂性结果。

与 Mastrovito 乘法器相同，一个 GF(2^m)上的乘法可以先直接相乘，随后再进行取模运算，许多文献如[77]等就采用这种方式。作者 Wu[77]使用不可约三项式作为约减多项式，并指出 GF(2^m)上的一个取模运算需要(w-1)(m-1)个加法，其中，w 是不可约多项式的汉明权重。在硬件实现乘法运算时，需要((m^2-1)+(w-1)(m-1))个 XG 和 m^2 个 AG。最近，Henriquez 和 Koc[78]对特殊的五项式提出了一个 PB 乘法器，并得出其时间延迟和所需要的 XG 和 AG 的个数。尽管作者都称其乘法器为 Mastrovito 乘法器，但是他们的体系结构与最初的 Mastrovito 乘法器是完全不同的，其体系结构是单独使用两步乘法完成的。

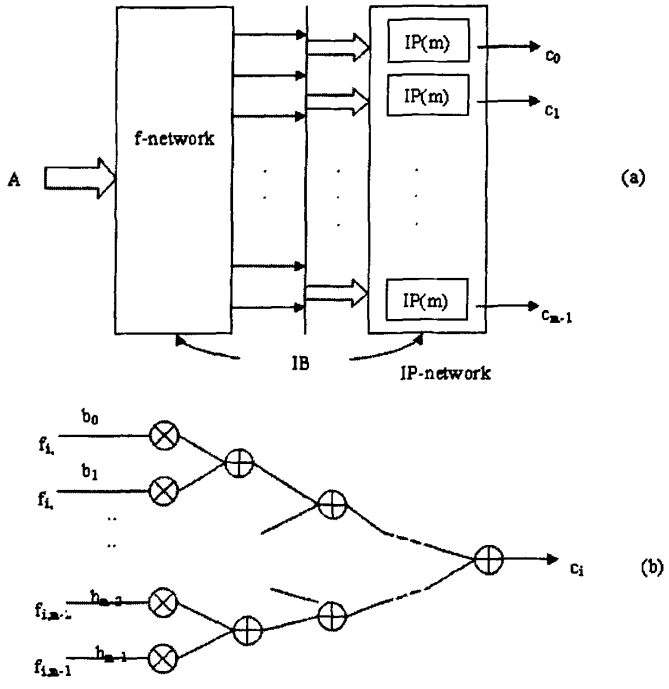


图 5-1 (a)GF(2^m)上的 Mastrovitoc 乘法器体系结构(b)IP(m)的详细结构

5.1.3 Karatsuba 乘法器

大多数有限域上的乘法运算算法都是分两步完成：第一步是多项式乘法，第二步是取模运算。即假设 A(x)，B(x)∈GF(2^m)，首先计算

$$C(x)=B(x)A(x)=(\sum_{i=0}^{m-1} a_i x^i)(\sum_{i=0}^{m-1} b_i x^i)$$

然后计算

$$C'(x)=C(x) \bmod P(x)$$

第一步通常采用经典(Classic)的多项式乘法算法，Karatsuba 算法，以及二者的混合运用算法。第二步通常采用的取模多项式有：等距多项式

(Equally-sapced Polynomial)，三项式和五项式。具体可参见文献[98]。

Karatsuba 算法由 Karatsuba 在 1962 年提出，它是第一个能够在低于 $O(m^2)$ 个操作内完成的多项式乘法[3.1.11][99]，其时间复杂度为 $O(m^{\log_2^3})$，其基本思想为：

假设

$$A(x)=\sum_{i=0}^{m-1}a_i x^i=\sum_{i=m/2}^{m-1}a_i x^i+\sum_{i=0}^{m/2-1}a_i x^i$$
$$=x^{m/2}\sum_{i=0}^{m/2-1}a_{i+m/2}x^i+\sum_{i=0}^{m/2-1}a_i x^i=x^{m/2}A^H+A^L$$
$$B(x)=\sum_{i=0}^{m-1}b_i x^i=\sum_{i=m/2}^{m-1}b_i x^i+\sum_{i=0}^{m/2-1}b_i x^i$$
$$=x^{m/2}\sum_{i=0}^{m/2-1}b_{i+m/2}x^i+\sum_{i=0}^{m/2-1}b_i x^i=x^{m/2}B^H+B^L$$

则

$$C(x)=x^m A^H B^H+(A^H B^H+A^L B^L+(A^H+A^L)(B^H+B^L))x^{\lceil m/2\rceil}+A^L B^L$$
$$=x^m C^H+C^L$$

最好的结果是结合经典算法和 Karatsuba 策略而得到，该 PB 乘法器的空间复杂度和时间复杂度如下：

$$\text{XOR gates}=(m/n)^{\log_2^3}(n^2+6n-1)-8m+2$$
$$\text{AND gates}=(m/n)^{\log_2^3}n^2$$
$$\text{Time Delay}=T_{AND}+T_{XOR}(\log_2^n+k)$$

其中 $m=2^k n$。

5.2 正规基乘法器

以上几种乘法器都有着不错的性能，因其较高的空间复杂性和缺乏规则性，在具体实现时，我们不愿意采用这种乘法器。近年来硬件乘法运算的另一个热点是正规基乘法器。本节将介绍正规基乘法器的原理和特点，并结合实际工程应用测出效率最佳的并行度，最后实现 $GF(2^{233})$ 上的正规基乘法器。

在域元素的正规基表示下，域元素的平方运算仅需对该元素的坐标进行简单循环移位即可，在硬件实现平方和开平方运算时，其开销可以忽略不计，因此，基于正规基表示的乘法运算及其乘法器设计得到了大量研究。

5.2.1 二元域正规基并行乘法器

一般地，记正规基表示的二元域元素 $A=(a_0, a_1, \ldots, a_{m-1})$，$a_i$ 表示 A 的第 i 个坐标，即 01 序列第 i 位。

将元素 A、B 分别看作行向量$(a_0, a_1, \ldots, a_{m-1})$和列向量$(b_0, b_1, \ldots, b_{m-1})$，$C=(c_0, c_1, \ldots, c_{m-1})$为它们的乘积，则计算 C 可以看作一系列矩阵叉乘运算：

$C=AB=(AM_0B, AM_1B, \ldots, AM_mB)$

其中 M_i 是一系列 01 矩阵，AM_0B 表示行向量 A 乘矩阵 M_0，得到再乘列向

量 B 得到一位，矩阵乘中矩阵向量分量的乘法是与，加法是异或。

M 如何得来呢？上文中 M_i 可以由 M_0 各列 i 次循环右移和各行 i 次循环下移得到。M_0 计算方法见文献[30，119]。而 M 的这个特点正好造就了并行的正规基乘法器。下面先举例说明乘法矩阵 M 的实际作用机理：

设 $f(x) = x^5 + x^2 + 1$ 是不可约多项式(正规多项式)，其根为 a，由 f(x)产生的有限域是 $GF(2^5)$。如果选择 $\gamma = \alpha^5$，则 $N = \{r, r^{2^1}, r^{2^2}, r^{2^3}, r^{2^4}\}$ 就是一个正规基。通过计算可得到 M 如下：

$$M = \begin{Bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{Bmatrix}$$

将矩阵乘操作转化为等式如下：

$$c0 = (a_0b_1 + a_1b_0) + (a_1b_3 + a_3b_1) + (a_2b_4 + a_4b_2) + (a_3b_2 + a_2b_3) + a_4b_4$$

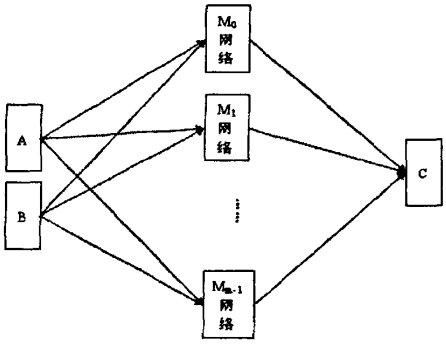


图 5-2 正规基并行乘法器

和上文所说相同，等式中乘法为与操作，加法为异或操作，于是以上等式也就是通过 A、B 计算结果的某一位可以用一个逻辑门网络实现。在计算得到各 M_i 之后，我们可以构造 m 个逻辑门网络，从而很容易得到一个并行乘法器如图 5-2。

这个乘法器只要一个时钟周期就能完成一次计算，效率非常高。但是，实际应用中，为了达到一定安全级别，要求 m 一般为 1024 位以上甚至 2048 位，而这一数字肯定会越来越高。在这种情况下，很明显，由于每个 M 网络规模并不小(2m 门以上)其硬件开销是非常惊人的，甚至达到了完全不可实用的程度；进一步考虑，A 和 B 中的数据要送给每一个 M 网络，其扇出系数也非常大；同时传输距离也不可避免的非常远，这会造成传输延迟非常严重，导致整个模块时钟频率非常低，应用在高速系统中是比较麻烦的，如果使用寄存器来分担这些传输延迟，会大大加重本来就很严重的开销问题；最后，大规模大扇出，功耗也是非常客观的。

5.2.2 二元域正规基串行乘法器

上一节的并行乘法器是通过 M_0 移位来获得不同的 M_i，从而实现运算并行。换一个角度考虑，我们也可以通过循环移位 A 和 B 来代替 M 的移位，即，将 M 固定不变，而 A 和 B 序列循环左移(上一节是右移矩阵)i 位，移位后送进同一个 M 后得到的就是 c_i，用公式表示如下：

C = (AMB，A<<1 M B<<1，...，A<<m-1 M B<<m-1)

根据以上原理，我们可以将 A 和 B 送入 M_0 网络得到第 0 位部分积，然后将二者循环左移再次送入 M_0 网络得到下一位部分积，这样循环 m 个周期，也能得到 C，这就是正规基串行乘法器的原理，其结构如图 5-3：

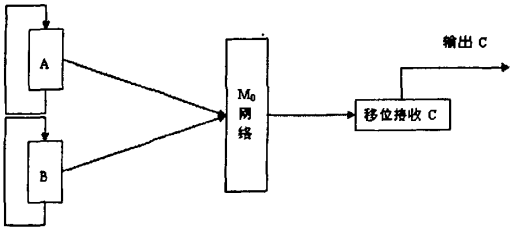


图 5-3 正规基串行乘法器

串行乘法器的优势主要是开销极小，几乎没有传输延迟，结构简单，低功耗。但是不算输入输出操作，串行乘法器执行一次乘法操作需要 m 个周期，这在有些场合也是不合适的。但是考虑求逆的时间复杂度也是 O(m)，串行乘法器和求逆构成流水线还是非常合适的。

5.2.3 II 型最佳正规基的优势

当使用正规基时，计算一个元素的平方只需一次循环移位即可完成。但是，计算两个不同元素的乘积时计算量还是很大的，而且选用不同的有限域会导致不同的计算复杂度。根据3.4节定义，不同m的二元域$GF(2^m)$在构造正规基时，根据构造过程中得到的本原单位根的次数T的不同，被分为不同类的正规基，称作T型正规基，T是二元域的性质，和m相关，而不和其他因素有关。不同的正规基，计算复杂度也是不一样的，这就是根据T分类的意义。

设$N=\{\beta, \beta^2, \beta^{2^2}, \ldots, \beta^{2^{m-1}}\}$是$GF(2^m)$上的一组正规基，$a = (a_0, a_1, \ldots, a_{m-1}) \in GF(2^m)$，$b = (b_0, b_1, \ldots, b_{m-1}) \in GF(2^m)$，$c = ab = (c_0, c_1, \ldots, c_{m-1})$，则存在

$\lambda_{i,j} \in GF(2)$，使得：

$$c_k = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} \lambda_{i,j} a_{i+k} b_{j+k} \quad (2\text{-}13)$$

令$C_N = \{(i, j) | \lambda_{i,j} \neq 0, 0=i, j=m-1\}$，则$C_N$的基数(集合的元素个数)，记为$|C_N|$，反应了在基N下做乘法的复杂度。

T型正规基的计算复杂度$|C_N|$满足下面的界：

$Tm - (T^2 - 3T + 3) = |C_N| = Tm-1$　　T是偶数

$(T+1)m - (T^2 - T + 1) = |C_N| = (T+1)m - T$　　T是奇数

根据以上分析，m 不同，乘法在该二元域上能达到的最低运算复杂度也不同。特别地，在硬件实现中，M 网络中异或门的个数为$|C_N|-1$，与门的个数为 m。所以我们选用 T 比较小的正规基(其实是选择有限域)是明智的。

可以看出，I 型正规基有可能复杂度最低。但文献[8]指出，当 $m \in [2, 2001]$ 时，有 117 个 m 的值是 I 型最优正规基，319 个 m 的值是 II 型最优正规基。显然，II 型最优正规基的数量几乎是 1 型最优正规基数量的 3 倍。说明，研究 II 型最优正规基更有意义。

$GF(2^{233})$就属于 II 型最优正规基。此外，这个长度的椭圆曲线密码体制的安全性超过 2048 位的 RSA 体制，在近期内是非常安全的，基于多方考虑，很多组织将 $GF(2^{233})$确定为下一代公钥密码体制标准的推荐二元域，比如 SET、IEEE 和 ANSI X9.62 等。所以本文选择二元域 $GF(2^{233})$实现改进求逆模块，同时也在这个二元域上实现正规基乘法器。

5.2.4 $GF(2^{233})$上 II 型最佳正规基串-并乘法器的优化实现

分析正规基乘法器的原理可以发现，正规基乘法器的并行度是完全可以人为控制的。将串行乘法器和并行乘法器原理相结合，将 A 和 B 循环移位 0 ~ n 次之后得到的 n+1 个二进制串先后送入 M_i 就能产生 C 中从 i~i+n 位的部分积。根据这一原理，我们从所有 M_i 中按照固定间隔选出若干矩阵，比如取出 M_0，M_{10}，M_{20}，...M_{10n}，让每个矩阵负责生成积 C 中一段二进制位，而由 A 和 B 循环移位不断供给各选出的 M_i，生成的二进制位 c 由 C 分段移位接收。这样就形成了正规基串-并乘法器，灵活性更强。原理图如图 5-4。

图中 M 网络的个数可以任意调整，对应不同的并行度。右侧 c 其实可以直接由部分积组合而成，图 5-4 为了说明过程才在移位接收部分积后加了一个组合过程。这样一个结构在实现时还要解决并行度的确定，流水线的设计等实际问题。

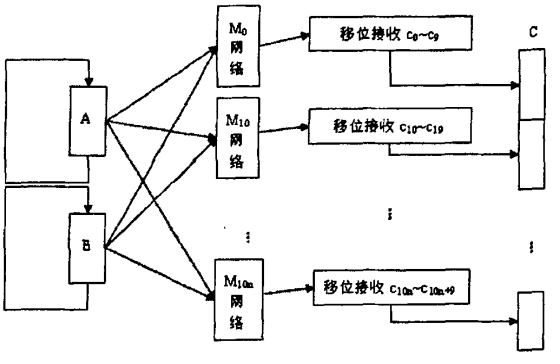


图 5-4 正规基串-并乘法器

➢ M 网络流水线设计

虽然应用最佳正规基，M 网络的逻辑门数量还是非常可观。回忆其运算表达式的结构：

$$c0 = (a_0b_1 + a_1b_0) + (a_1b_3 + a_3b_1) + (a_2b_4 + a_4b_2) + (a_3b_2 + a_2b_3) + a_4b_4$$

表 5-1 流水线级数对时钟的影响

流水线级数	达到时钟频率
纯组合逻辑	155.7Mhz
2 级流水	337.4Mhz
3 级流水	565.9Mhz
4 级流水	646.4Mhz

将括号去掉后可以发现，所有的与操作可以并行，但是所有的加(异或)操作却只能顺次作。而异或的次数为$|C_N|$-1，一般来说约有 2m 个，2m 级异或门的延迟为 $O(\log_2 m)$，在密码运算中，m 可能达到上千的级别，即使在本文实现得 $GF(2^{233})$上也是是非常严重的，所以我们要设计流水线来提高运行时钟频率，主要工作就是实验确定流水线级数。经过实验得到 $GF(2^{233})$上 M_i 流水线级数和达到的时钟频率关系如表 5-1。

从表中可以看出，随着流水级数升高，性能提升再逐渐减小。4 级流水以上硬件开销非常大，且 FPGA 器件最高频率锁定在 500Mhz，所以在本应用中 3 级流水非常合适，但是如果使用 ASIC 实现，那么 4 级流水也可以考虑。所以编写 verilog 代码时使用的是 4 级流水。随着二元域 m 的增大，流水级数也应相应增加。

➢ 最优并行度的测定

M 网络确定后，下一步就是确定最佳的并行度。随着并行度的提高，完成乘法需要的时钟周期会逐步降低。但是同时由于 A 和 B 送给多个规模不小的 M 网络，传输距离比较远，而且 M 网络越多，占用面积越大，传输绕过的路径也就越长，所以模块的时钟频率会相应下降。所以我们要找到一个时钟数和频率的积最大的并行度，才是最合理的。当然，过高的并行度带来的开销是非常惊人的，所以我们从低向高测试各并行度的性能，当开销过大时停止。实验结果如表 5-2。

表 5-2 并行度和性能关系

并行度	运算时间
串行	353.75ns
4	151.22ns
15	49.64ns
30	35.6ns

表 5-2 中并行度用 M 网络的个数表示，性能用延迟×周期数也就是执行一

次乘法需要的时间来表示，单位是纳秒。并行度 30 相对于并行度 15 提高速度已经开始下降，且并行度 30 硬件开销已经不小，更高的实验是没有意义的。随着 m 的增大，M 网络占用的面积也会加大，高并行度的延迟会更加明显。本文从实际出发，选用并行度 30，还因为此时乘法器达到的时钟频率为 224.72Mhz，和前文求逆模块最为接近，配合使用最为合适。

5.2.5 改进殆逆模块中最后一步乘法的特殊优化

第四章提出的殆逆算法的改进方案中，计算结束后还有一步 $a = b \times x^{-k}$。这一步看似简单，真的执行起来是很耗时的要做一个 x^{-1} 的 k 次方，当然 x^{-1} 可以预先存储，然后再作一次乘法。

由于密码系统多处理流数据，硬件实现此算法时可以将前面计算和 $a = b \times x^{-k}$ 做成两级流水，流水线运行起来时，乘法相当于和前面的计算并行。当然，这个乘法器的速度是有一定要求的，也就是做一次乘法的周期数不可太多。粗略估计，最后一步大致要做 $\log_2 k$ 次乘法，而前级运算出一对 b 和 k 至多需要 2m 个时钟周期，而最少情况下只有 m/2 个周期，那么一次乘法只能使用约 $m/2\log_2 k$ 个周期，即使前级运算出一对 b 和 k 使用 m/2 的情况很少见，我们要求一次乘法使用 $m/\log_2 k$ 个周期，也只能使用比特间并行度比较高的乘法器，并且计算幂的硬件是非常复杂的，开销也非常大。

考虑到，在有片上存储资源的 FPGA 中或者存储资源充裕的软件实现时，我们是否可以记录中间结果，把所有可能的 k 对应的 x^{-k} 形成一个表，那么此方案最后一步 $a=b\times x^{-k}$ 只需先取出 k 对应的表项，再计算一次乘法即可，这样就是利用存储资源来代替复杂乘法器的开销。而此时可以选用开销最小的比特间串行的乘法器，计算一次乘法使用 m 个周期，和前级模块构成二级流水非常合适。现在，k 的取值范围就非常重要，只有 k 的数目不特别大，用查表代替计算才是值得的，我们分析一下 k 的取值范围：

Deg(F)和 deg(G)在两种情况下减小：一是 F 为偶数时移位(G 交换给 F 时就减小 deg(G))；二是 deg(F)和 deg(G)相等时做 F = F + G 会使度数减小。k 累计了 F 和 G 在寄存器 F 中移位总次数，即第一种情况的度数减小总数。

m 确定后，k 值在什么情况下最大呢？前文提到过，度数减小的最大值是 2m。在此基础上我们进一步假设计算过程中保持 $deg(F) \neq deg(G)$，即所有度数的减小都是由移位而不是 F + G 产生，即都会被 k 累计，于是 k 的上限也是 2m。k 的下限出现在 F = 1 时，计算直接结束，k = 0。

基于以上结论，我们可以事先计算 x^{-1} 的 0~2m 次幂形成一个 mbit × (2m+1) 的表，那么计算 $a = b \times x^{-k}$ 即可用一次乘法的时间完成。

若域元素 $a^n \equiv 1 \bmod p(x)$，则称 n 为 a 的阶，根据此定义有 $a^{n+r} = a^r$ 也就是 a^k 的值出现有周期性。所以若 x^{-1} 的阶小于 2m，说明在 k 还没达到 2m 之前

x^{-k}的值已经出现了重复，则上文预计算的表又可以压缩到 mbit × $n_{x^{-1}}$ 。

这样的改进在片上存储丰富的 FPGA 和软件实现时是非常实用的，因为在 FPGA 中这些存储资源如果不利用也是白白浪费的；而软件实现时，存储器资源是相对便宜的资源。就算是 ASIC 实现，单一存储器成本也小于同样面积的不规则的逻辑电路。

第六章 总结与展望

6.1 总结

密码学是一门古老而又新颖的技术，公钥密码技术被认为是现代密码学的核心，受到了世界各国信息安全从业人员的广泛关注。在硬件上实现专门的安全模块甚至安全系统，可以解决密码算法效率差的问题，可以实现移动设备的安全，可以避免处理器资源过多的分配给安全操作，可以实现安全功能的透明化...但是，即使是用硬件实现，公钥密码算法的执行效率仍然和某些场合的实用要求有一定差距，并且尽量提高模块效率，也是充分开发硬件的潜力，避免浪费。基于以上情况，本文综合研究各种密码算法，提出了适用于公钥密码系统的底层硬件设计方案，提高安全算法的效率，也为进一步构建软硬件协同安全系统提供了基石。

回顾了公钥密码学的发展现状后，着重广泛研究各种流行的公钥密码体制；细致分析它们的数学理论基础；最后将注意力集中在适用于各种公钥算法的有限域底层运算上，旨在针对硬件运算模块的效率，结合实际安全强度的要求，通过利用算法规律，创新模块架构等方式，达到压缩模块执行周期，提高运行时钟频率，并尽可能降低硬件开销的目的。主要工作及创新点可归纳如下：

1. 从各个层面分析总结了公钥密码体制的运行流程，广泛调查研究各种经典的、优秀的、实用的已有算法，进行分类对比，选择改进的切入点。
2. 针对求逆运算，选择改进收敛速度快、适合硬件实现的殆逆算法。提出了二元域上殆逆算法的硬件移植方案，它充分利用了原算法分两阶段的特点，简化了结构，缩小了规模，达到了低延迟的目的。又根据度数的前后相关性快速求得新度数，压缩时钟周期；根据硬件特点分解元算法步骤，重新合并，使判断、交换和相加三个运算同时完成，进一步压缩时钟周期数。软件模拟模块运行和 Quartus 仿真结果表明，此方案能大大提高求逆运算的效率。
3. 针对上文方案存在的 AT-product(面积时间积)复杂度为较差的问题，进一步提出了一种双向移位的架构，利用反向移位代替前文方案关键路径中门延迟最严重的求度数运算。而且新的结构更方便布局布线，能进一步降低传输延迟。双向移位结构将关键门延迟复杂度从前一方案的 $\log_2 m$ 降低到 $\log_2(\log_2 m)$，效果非常明显，而开销增量只是线性的。
4. 除了求逆，密码系统中效率较差的是乘法运算。本文也对乘法运算作了一定的探讨。主要是通过广泛对比，选择二元域最佳正规基串-并乘法器进行探索，通过仿真测评找到效率相对较高的并行度参数，并实现了 $GF(2^{233})$上的最佳正规基串-并乘法器。同时，对于求逆后一阶段

的乘法运算提出了查表方法提高效率，并对查表的可行性进行了论证。

6.2 展望

虽然本文在利用硬件提高公钥密码算法效率方面的研究工作取得了一定的成绩，但是安全标准日新月异，安全需求日益提高，新的密码算法也层出不穷。这些都迫切要求我们不断寻求新的解决方案，来有效应对以上因素对安全系统效率的负面影响，在理论和应用中都还有很多方面需要进一步研究。另外，本人的领域知识有限，加之密码学本身的广博性，相关数学理论的抽象性等特点，研究肯定是存在死角的，进一步的工作是非常必要的，也是永远不能停止的。

有了零件，下一步的工作就是组装成一个系统。本文提供了最基本的底层模块，只有将其应用到具体的公钥密码系统中才能发挥其效力。比如可以将其组合成椭圆曲线密码体制的某些模块；可以将其组合成一套完整的 RSA 加密签名系统；更可以将这些算法组织起来，结合 PCI 接口模块，形成一块 PCI 加速卡，配合上相应的驱动程序，组成一套完整的软硬件协同系统，用于安全服务器；或者是作为指令应用到比如 Altera 的 nios II 平台中，形成强化安全功能的处理器等。

从底层运算层面实现安全系统的加速，是从根本上解决问题，应用方式也比较灵活。但是从更高层解决效率问题也是非常有效的。尤其是创立全新的公钥密码体制，或者使用更优秀的有限域，或者是优化执行参数等方法。

本文实现的模块都是基于 $GF(2^{233})$，这正是很多安全国际标准中推荐的椭圆曲线密码体制在近期内保证安全的密钥长度，这些标准同时也将椭圆曲线密码体制确定为下一代主流公钥体制。于是下一阶段我们可以以本文的成果为基础，研究新的椭圆曲线明文编码方案；研究更高效的点的数乘方案；甚至是实现一套完整的椭圆曲线密码系统。

当然，也可以以本文为参照，从其它运算或有限域入手，同样研究更优秀的底层运算模块，尤其是正规基乘法器，可做的工作还有很多，空间还很大。

由于时间和精力有限，本文着重于二元域底层算法硬件加速的探索和研究。在实际安全应用中，还有很多工作可以结合硬件来做。我相信这是一个值得努力的研究课题，能够在信息安全领域产生直接的巨大价值。

参考文献

[1] Alfred J, Menezes Paul C, van Oorscho, 胡磊等译. 应用密码学手册[M]. 北京: 电子工业出版社, 2003.

[2] 秦志光. 密码算法的现状和发展研究[J]. 计算机应用, 2004, 24(2):1-3.

[3] Steve Bumett, Stephen Paine著, 冯登国, 周永彬, 张振峰等译. 密码工程实践指南[M]. 北京: 清华大学出版社, 2001: 1-7.

[4] Wade Trappe, Lawrence C.Washington 著, 邹红霞, 许鹏文, 李勇奇译. 密码学概论[M]. 北京: 人民邮电出版社, 2004: 8-42.

[5] 陈超, 曾晓洋, 章倩荃. 一种新型硬件可配置公钥制密码协处理器的 VLSI 实现[J]. 通信学报, 2005, 26(1): 6-11.

[6] T.R.Padmanabhan, B.Bala Tripura Sundari.. Design Through Verilog HDL[M]. US:Wiley-IEEE Press, 2003.

[7] R.L Rivest, A.Shamir, L.Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems[J]. Communication of ACM, 1978, 21(2): 120-126.

[8] Merkle R C, Hellman M E. Hiding Information and Signatures in Trapdoor Knapsacks[J]. IEEE Transactions on Information Theory, 1978,24(5):525-530.

[9] Rabin M O. Digitalized signatures and public key functions as intractable as factorization[R]. London:MIT lab. For Computer Science, 1979.

[10] ElGamal T. A public key cryptosystem and signature scheme based on discrete logarithms[J]. IEEE Transactions on Information Theory, 1985, 31(4): 469 - 472.

[11] Koblitz N. Elliptic curve cryptosystems. Mathematics of Computation[J], 1987, 48(117):203-209.

[12] McEliece R J. A public-key cryptosystem based on algebraic coding theory[R]. California: Jet Propulsion Lab, 1978, 114-116.

[13] 陶仁骥, 陈世华. 一种有限自动机公开钥密码体制和数字签名[J]. 计算机学报, 1985: 8(6): 401-409.

[14] 曹珍富. 公钥密码学[M]. 哈尔滨: 黑龙江教育出版社, 1993.

[15] Schnorr C P. Efficient identification and signature for smart cards[J]. Journal of Cryptography, 1991, 4(3):161-174.

[16] NIST, Digital Signature Standard (DSS)[S]. Federal Information Processing Standards Publication 186, 2000.

[17] Mambo M, Usuda K, Okamoto E. Proxy signatures:Delegation of the power to sign messages[J]. IEICE Transactions on Fundamentals of Electronics,

communications and Computer Sciences, 1996, 79(9):1338-1354.

[18] Chaum D. Blind signatures for untraceable payments[M]. New York: Plenum Press, 1993, 199-203.

[19] Boneh D, Gentry C, Lynn B, et al. Aggregate and verifiably encrypted signatures from bilinear maps[A]. Advances in Cryptography–Eurocrypt 2003, LNCS[C]. Berlin: Springer-Verlag, 2003, 416-432.

[20] Chaum D, Antwerpen H van. Undeniable signatures[C]. CRYPTO'89, LNCS [A]. Berlin: Springer-Verlag, 1989, 212-216.

[21] Bellare M, Miner S. A forward-secure digital signature scheme[C]. CRYPTO'99, LNCS 1666[A]. Berlin: Springer-Verlag, 1999, 431-448.

[22] Dodis Y, Katz J, Xu S, et al. Strong Key-Insulated Signature Schemes[C]. Public Key Cryptography-PKC 2003, LNCS2567[A]. Berlin: Springer-Verlag, 2003, 130-144.

[23] Shamir A, Tauman Y. Improved online/offline signature schemes[C]. Proceedings of Advances in Cryptology:Crypto'01,LNCS2139[A]. Berlin: Springer-Verlag, 2001, 355-367.

[24] Desmedt Y. Society and group oriented cryptography:A new concept[C]. Crypto'87, LNCS293[A]. Berlin: Springer-Verlag, 1988, 120-127.

[25] Boneh D, Gentry C, Lynn B, et al. Aggregate and verifiably encrypted signatures from bilinear maps[C]. Cryptography–Eurocrypt 2003[A]. Berlin: Springer-Verlag, 2003, 416-432.

[26] Rivest R, Shamir A, Tauman Y. How to leak a secret[C]. ASIACRYPT 2001, LNCS 2248[A]. Berlin: Springer-Verlag, 2001, 552-565.

[27] Jakobsson M, Sako K, Impagliazzo R. Designated verifier proofs and their applications[C]. EUROCRYPT'96, LNCS1070[A]. Berlin: Springer-Verlag, 1996, 143-154.

[28] Chaum D. Designated confirmer signatures[C]. Eurocypt'94, LNCS950[A]. Berlin: Springer-Verlag, 1995, 86–91.

[29] Wang G, Bibliography on signatures[EB/OL]. http://icsd.i2r.a-star.edu.sg/staff/guilin/bible.htm

[30] PA Ivey, SN Walker, JM Stern, et al. An ultra-high speed public key encryption processor[C]. IEEE 1992 Custom Integrated Circuits Conference (CICC'92)[A], 1992, 19.6.1-19.6.4.

[31] Yoshinoh Fujisawa, Yasushi Fuwa. Proposal of a High-Speed Processing Method for RSA Cryptograms using High-Radix Signed-Digit Numbers[J]. Personal Computer Users' Application Technology Association, 2000,

10(l):61-70.

[32] Yoshinori Fujisawa, Yasushi Fuwa. A High-Speed Processing LSI for RSA Cryptograms using High- Radix Signed-Digit Numbers and a new algorithm of modulo operation[C]. Fifteenth International Symposium on Mathematical Theory of Networks and Systems[A], 2000.

[33] Tefafire Product Brief, www.athena-group.com .2002.

[34] CS1015/Rubicon Porduct Brief, www.chipsien.com .2002.

[35] http://www.zte.corn.cn/03productiopen3.jsp?m=261.

[36] Agnew G B, Mullin R C, Vanstone S A. An Implementation of Elliptic Curve Cryptosystems over GF(2^{155})[J]. IEEE Journal on Selected areas in Communications, 1993, 11(5):804-813.

[37] Okada S, Torii N, Kouichi I. Implémentation of Elliptic Curve Cryptographic Coprocessor over GF(2^m) on an FPGA [C]. Cryptographic Hardware and Embedded Systems-CHES 2000[A]. Berlin: Spring-verlag, 2000, 215-242.

[38] Orlando G, Paar C. A High-performance Reconfigurable Elliptic Curve Processor for GF(2^m) [C]. Cryptographic Hardware and Embedded Systems -CHES 2000[A]. Berlin: Spring-verlag, 2000, 41-56.

[39] Itoh T, Tsujii S. A fast algorithm for computing multiplicative inverses in GF(2^n) using normal bases[J]. Information And Computation, 1988, 78(3): 171-177.

[40] Sang ho oh, Chong Han Kim. Algorithm of inverse operation in GF(2^n) [J]. IEEE Transactionson Information Theory, 1998.

[41] Nam Yeun Kim, Kee Young Yoo. Systolic architectures for inversion/ division using AB2 circuits in GF(2^m)[J]. Integration,the VLSI journal, 2003, 35(1): 11–24.

[42] Hankerson DR, Menezes AJ, Vanstone SA. Guide to Elliptic Curve Cryptography[M]. New York: Springer-verlag, 2004.

[43] Schroepel R, Orman H, Malley SO, et al. Fast key exchange with elliptic curve systems[C]. D. Coppersmith. Advances in Cryptology-CRYPTO'95, LNCS963, 1995[A]. London: Springer–Verlag, 1995.

[44] de Dormale G M, Bulens P, Quisquater J J. An improved Montgomery modular inversion targeted for efficient implementation on FPGA[J]. Field-Programmable Technology, 2004, 441-444.

[45] Cilardo.A, Mazzeo.A, Romano.L, et al. Carry-save Montgomery modular exponentiation on reconfigurable hardware[J]. Design, Automation and Test in Europe Conference and Exhibition, 2004, 206-211.

[46] E.R.BerlekamP. Algebraiccodingtheory, McGRaw-Hill, 1968.

[47] E.D.Mastrovito. VLSI architectures for computation in Galois fields[D]. Linkoping Sweden: Linkoping Uni, 1991.

[48] Araki K, Fujita I, Morisue M. Fast inverters over finite field based on Euclid's algorithm[J]. IEEE Transactions IEICE, 1989, 72(11):1230-1234.

[49] Brunner H, Curiger A, Hofstetter M. On computing multiplicative inverses in GF(2^m)[J]. IEEE Transactions on Computers, 1993, 42(8): 1010-1015.

[50] 袁丹寿, 戎蒙恬. 基于改进欧几里德算法的可重构性逆元结构[J]. 上海交通大学学报, 2006, 40 (1):36-40.

[51] 鲍可进, 宋永刚. 基于 FPGA 的有限域求逆算法的改进及实现[J]. 计算机工程, 2006, 32(23):156-170.

[52] 王健, 蒋安平, 盛世敏. 同时支持两种有限域的模逆算法及其硬件实现[J]. 北京大学学报(自然科学版), 2007, 43(01).

[53] Guo J H, Wang C L. systolic array implementation of Euclid's algorithm for inversion and division in GF(2^m)[J]. IEEE Transactions on Computers, 1998, 47(10): 1161-1167.

[54] Z.Yan, D.V.Sarwate, Z.Liu. High-speed systolic architectures for finite field inversion[J]. Integration,the VLSI Journal, 2005, 38(3):338-398.

[55] Z.Yan, D.V.Sarwate. Area-Efficient Two-Dimensional Architectures for Finite Field Inversion and Division[C]. Great Lakes Symposium on VLSI[A], New York: ACM, 2005, 116-121.

[56] A.R.Masoleh. Eficient algorithms and architectores for field multiplication using gasusian nomral bases[J]. IEEE Transactions on computers, 2006, 55(1):34-47.

[57] IEEE Std 1363-2000. IEEE Standard specifications for Public-key cryptography[S]. 2000.

[58] A.R.Masoleh, M.AHasan. A new construction of massey-omura parallel multiplier over GF(2^m)[J]. IEEE Transactions on Computers, 2002, 51(5): 511-520.

[59] J.L.Masscy, J.K.omura. Computational method and apparatus for finite field arithmetic[P]. US Patent: 4587627, 1986.

[60] MA Hasan, MZ Wang, VK Bhargava. A modified massey-omura parallel multiplier for a class of finite fields[J]. IEEE Transactions on Comptuers, 1993, 42(10): 1278-1280.

[61] C.K.Koc, B.Sunar. Low-complexity bit-parallel canonical and normal basis multipliers for a class of fineit fields[J]. IEEE Transactions on Computers,

1998, 47(3):353-356.

[62] Mullin RC, Onyszchuk IM, Vanstone SA, et al. Optimal normal bases in GF(p^n)[J]. Discrete Applied Mathematics, 1989, 22 (2): 149-161.

[63] B.Sunar, C.K.Koc. An eficient optimal normal basis type II multiplier[J]. IEEE Transactions on Computers, 2001, 50(5):83-87.

[64] A.R.Masoleh, M.A.Hasan. Low complexity word level sequential normal basis multipliers[J]. IEEE Transactions on Comptuers, 2005, 54(2):98-109.

[65] 廖群英. 有限域上最优正规基的乘法表[J]. 数学学报, 2005, 48(5):947-954.

[66] H.Wu, M.A.Hasan, I.F.Blake, et al. Finite field multiplier using redundant representation[J]. IEEE Trans. Comptuers, 2002, 51(11):1306-1316.

[67] B.Sunar. A generalizedmethod for constructing sub-quadratic complexity GF(2^k) multipliers[J]. IEEE Transactions on Comptuers, 2004, 53(9): 1097-1105.

[68] H.Fan, Y.Dai. Key function of normal basis multipliers in GF(2^n)[J]. Electronics Leters, 2002, 38(23):1431-1432.

[69] 鲁俊生, 张文祥, 王新辉. 一种基于有限域的快速乘法器的设计与实现[J]. 计算机研究与发展, 2004, 41(4):755-760.

[70] H.Wu. On computation of polynomial modular reduction. http://www.cacr.math.uwater-loo.ca/tecehrport/2000/corr2000-31.pdf

[71] T.C.Bartee, D.I.Schneider. Computation with finite fields[J]. Information and Computers, 1963, 6(3):79-98.

[72] E.D.Mastrovito. VLSI designs for multiplication over finite fields GF(2^m)[C]. Applied Algebra, Algebraic Algorithms and Error-Correcting Codes[A], 1989, 297-309.

[73] B.Sunar, C.K.Koc. Mastrovito multiplier for all trinomials[J]. IEEE Transactions on Computer, 1999, 48(5):522-527.

[74] A.Halbutogullari, C.K.Koc. Mastrovito multiplier for general irreducible polynomials[J]. IEEE Tansactions on Computers, 2000, 49(5):503-518.

[75] T.Zhang, K.K.Parhi. Systematic design of original and modified Mastrovito multipliers for Gene arl irreducible polynomials[J]. IEEE Transactions on Computers, 2001, 50(7):734-748.

[76] L.Song and K.K.Parhi. Low-complexity modified Mastrovito multipliers over finite fields GF(2^m)[C]. IEEE Inetnrational Symposium on Circruits and Systems, 1999, 508-512.

[77] H.Wu. Bit-parallel finite field multiplier and squarer using polynomial basis[J]. IEEE Transactions on Computers, 2002, 51(7):750-758.

[78] F.R.Henriquez, C.K.Koc. Parallel multipliers based on special irreducible pentanomials[J]. IEEE Trans.Computers, 2003, 48(5):522-527.

[79] P.L Monigomery. Modular Multiplication without Trial Division[J]. Mathematics of Computation, 1985, 44(170):519-521.

[80] C.K.Koc, T.Acar. Montgomery Multivlication in GF(2^k)[J]. Design,Codes and Cryptography, 1998, 14(1):57-69.

[81] E.Savas, A.F.Tenca, C.K.Koc. A Scalable and Unified Multiplier Architecture for Finite Fields FG(p) and GF(2^m)[J]. Cryptographic Hardware and Emmbedded Systems(CHES2000), 2000, 7287-7292.

[82] W.Diffie, M.E.Hellman. Muliuser Cryptographic Techniques[C], AFIPS National Computer Conference[], New York:ACM, 1976, 109-112.

[83] 周玉洁, 冯登国. 公开密钥密码算法及其快速实现[M]. 北京: 国防工业出版社, 2002, 88-100.

[84] 段云所, 魏仕民, 唐礼勇等. 信息安全概论[M]. 北京: 高等教育出版社, 2003.

[85] 曹建国. RSA 公钥密码安全性的研究及其 FPGA 实现[D]. 硕士学位论文, 西南交通大学, 2006.

[86] 王张宜, 杨寒涛, 张焕国. 椭圆曲线密码的安全性分析[J]. 计算机工程, 2002, 28(5):161-163.

[87] R.Lidl, H.Niederreiter. Introduction to finite fields and their applications[M]. Cambridge: Cambridge Univ Press, 1994.

[88] 王庆先. 有限域运算和椭圆曲线数乘运算研究[D]. 博士学位论文, 电子科技大学, 2006.

[89] 黄威. 椭圆曲线密码（ECC）算法的 FPGA 实现及优化设计[D]. 硕士学位论文, 武汉理工大学, 2006.

[90] 林东岱. 代数学基础与有限域[M]. 北京: 高等教育出版, 2006.

[91] A.R.Masoleh and M.A.Hasan. A new construction of massey-omura parallel multiplier over GF(2^m)[J]. IEEE Trans. Computers, 2002, 51(5):511-520.

[92] A.R.Masoleh. Efficient algorithms and architectures for field multiplication using gaussian normal bases[J]. IEEE Transactions on computers, 2006, 55(1): 34-47.

[93] N.Koblitz, A.Menezes, S.Vanstone. The state of elliptic curve cryptography[C]. Designs, Codes and Cryptography[A], New York: Springer-Verlag, 2000, 173-193.

[94] D.Hankerson, A.Menezes, S. Vanston. Guide to elliptic curve cryptography[M]. New York: Springer-Verlag Professional Computing Series,

2004.

[95] M.Brown, D.Hankerson, J.Lpez, et al. Software implementation of the NIST elliptic curves over prime fields[C]. Topics in Cryptology-CT-RSA 2001[A], Berlin:Springer-Verlag, 2001, 250-265.

[96] IEEE. P1363/Draft Version 13.Annex A.Number-Theoretic Background[S]. New York: IEEE, 1999, 99.

[97] E.R.Berlekamp. Algebraic coding theory[M]. NewYork: McGRaw-Hill, 1968.

[98] F.Rodrfguez Henquez, C.K.Koc. On fully Parallel Karatsuba Multipliers for GF(2^m). http://security.ece.orst.edu/papers/c29fpkar.pdf .

[99] P.L.Montgomety. Five,six,and seven-term Karatsuba-like formulae[J]. IEEE Trans.Computers, 2005, 54(3):362-369.

攻读硕士学位期间发表的论文

[1] 宋灏龙，梁华国，单国华. FPGA 上二元域公钥系统中求逆模块的改进. 小型微型计算机系统.已录用

[2] 宋灏龙，梁华国，单国华. 公钥系统中高效的硬件二元域求逆模块. 计算机工程. 已录用