南京航空航天大学

硕士学位论文

可重构分组密码处理系统实现研究

姓名：郑东

申请学位级别：硕士

专业：测试计量技术及仪器

指导教师：王友仁

20090201

摘 要

随着信息技术的发展，网络应用、电子政务和电子商务的普及，信息的安全性越来越受人关注。采用可重构计算技术来设计密码处理系统，使同一硬件能够高效灵活的支持密码应用领域内的多种算法，同时满足了对性能和灵活性的要求，提高了密码系统的安全性，在军事以及商业等领域具有很大的应用价值。论文针对分组密码处理应用领域，结合了可重构结构的设计思想和方法，完成的主要工作和研究成果如下：

（1）研究了分组密码基本运算模块重构技术，设计了可重构密码运算模块 RCM。RCM 设计包括了可重构 S 盒替代模块、可重构移位模块、可重构置换模块、模加/减、模乘等可重构密码运算模块。这些基本运算模块可根据 RCM 配置指令进行重构，灵活完成不同算法所需的运算功能。各可重构密码运算模块 RCM 在 FPGA 上进行了综合，并得到正确的仿真验证。

（2）深入分析了现有的主要分组密码算法操作特征以及处理结构特点，研究并设计了一种可重构密码处理结构 RCPA。该结构的设计特点是具有可变的并行度以及可配置的流水线结构，可从横向和纵向两个方向组织可重构运算模块，能够快速实现密码处理。

（3）基于可重构密码处理结构模型 RCPA，具体针对 DES 和 AES 算法，采用 VHDL 硬件描述语言，完成了 DES/AES 可重构密码处理结构的实现，并分别对 DES 和 AES 进行了算法映射，给出了仿真波形和性能分析。

（4）研究了一种高性价比的 AES 密码系统 VLSI 实现方案，同时解决了高速加/解密时轮密钥分配的同步问题。此实现方案设计的 AES 密码系统与其他硬件实现方案比较，具有更高的性价比。

本课题研究工作受国家自然科学基金（60871009，60374008，90505013）和航空科学基金（2006ZD52044，04152068）的资助。

关键词：可重构，分组密码，DES，AES，密码处理，可重构运算模块

Abstract

Along with the development of information technology, the electronic governmental affairs, electronic commerce and the distant application of the network in general have been being used widely, the safety of the information exchanged by the network is played more and more attention to by users. The design of a cipher processing system adopts reconfigurable computing technology, which can support multiple cryptographic algorithms in the cipher application. Therefore, it can achieve crypto algorithms processing with efficient and flexibility, and it also solves the hidden trouble in the cipher processing system. The reconfigurable cipher processing system will be widely used in military and commerce fields. We focus on block cipher processing application in this paper, and research for the design idea and method of reconfigurable architecture. The main work and research fruits are given below:

(1) This Paper has researched reconfigurable technology of the basic element in the block cipher processing and designed the Reconfigurable Cipher operating Module(RCM) in the RCPA., RCMs include reconfigurable S-box alternative module, reconfigurable shift module, reconfigurable permutation module, module plus/minus module, module multiplicative module and so on. These RCMs can be reconfigured according to the configuration instruction, and achieved the function which different crypto algorithms needed flexibly. All RCMs have been synthesized via Xillinx's FPGA and correct simulations are achieved.

(2) This paper has analysed the operation and processing structure characteristics of popular block cipher algorithms, and researched a Reconfigurable Cipher Processing Architecture(RCPA), The features of the design are variable degree of parallelism and configurable pipeline structure and reconfigurable processing modules are organized from the horizontal and vertical directions. The design has hign speed of cipher processing.

(3) Specifically for AES and DES algorithm, the entire Reconfigurable Cipher Processing Architecture is achieved using VHDL hardware description language based on the model of RCPA. Respectively DES and AES algorithm are mapped and wave simulation and performance analysis are provided.

(4) This paper researches a hign ratio of performance and cost VLSI implementation of AES algorithm. At the same time, synchronization problem of round key distribution is solved, along with high-speed encryption/decryption. The results show its efficiency on hign ratio of performance

comparing with other current designs.

The research work presented here in this paper has been funded by National Natural Science Foundation of China (60871009, 60374008, 90505013) and Aeronautic Science foundation of China (2006ZD52044, 04152068).

Keywords: Reconfigurable, Block Cipher, DES, AES, Cipher Processing, Reconfigurable Operating Module

图表清单

注释表

序号	简称	英文全称	中文全称
1	AES	Advanced Encryption Standard	高级数据加密标准
2	ALU	Arithmetic Logical Unit	算术逻辑单元
3	ASIC	Application Specific Integrated Circuit	专用集成电路
4	BRU	Basic Reconfigurable Processing Unit	基本重构处理单元
5	DES	Data Encryption Standard	数据加密标准
6	FPAG	Field Programmable Gate Array	现场可编程门阵列
7	GPP	General Purpose Processor	通用微处理器
8	LUT	Look Up Table	查找表
9	MIPS	Million Instructions Per Second	每秒百万指令
10	NRE	Non-Recurring Engineering	非重复性工程
11	RC	Reconfigurable Computing	可重构计算
12	RCM	Reconfigurable Cipher operating Module	可重构密码运算模块
13	RCPA	Reconfigurable Cipher Processing Architecture	可重构密码处理结构
14	RCPM	Reconfigurable Cipher Processing Module	可重构密码处理模块
15	RHCA	Reconfigurable interconnect Hiberarchy Cipher processing Architecture	可重构层次互连密码处理结构
16	RISC	Reduced Instruction Set Computer	精简指令集计算机
17	RS	Reconfigurable System	可重构系统
18	SIMD	Single Instruction Multiple Data	单指令多数据流
19	VHDL	Very High Speed Integrated Circuit Hardware Description Language	超高速集成电路硬件描述语言
20	VLSI	Very Large Scale Integrated circuit	超大规模集成电路

承诺书

本人声明所呈交的硕士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。除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得南京航空航天大学或其他教育机构的学位或证书而使用过的材料。

本人授权南京航空航天大学可以将学位论文的全部或部分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承诺书)

作者签名：_____

日　　期：_____

第一章 绪 论

1.1 可重构密码处理系统实现的背景

在信息技术飞速发展的今天，信息技术已经渗透到政治、经济、军事和社会生活的各个方面，为了解决信息的安全性问题，人们采取了许多措施，其中，对信息数据进行加密就是一个行之有效的措施。传统的数据加密方法有两种：一种是软件加密；另一种是硬件加密。两种数据加密方法各有特点，软件加密虽然灵活，但加密速度慢，并且容易被人改动；硬件加密虽然速度快，但算法单一，且固定不变，长此以往，总有破解的一天。为了解决上述两种加密方法的缺点，我们运用可重构技术设计出可以实现多种密码算法的密码芯片，既能灵活、快速的实现多种不同的密码算法，又能避免上述安全上的隐患。

本课题就是利用可重构技术对分组密码算法中的运算模块进行可重构设计，并在此基础上对整个可重构密码处理系统展开研究和设计。

美国已于 1998 年推出了可重构加密芯片的产品[1,2]，我国一些公司和高校也开始了对可重构加密芯片的研究和开发[3]。

1.2 可重构密码处理的研究现状

目前密码算法的主要硬件处理系统可分为三类：通用微处理器处理系统、固定密码算法 ASIC 处理系统和可重构密码处理系统。通用微处理器实现方式的优点是灵活性高，可以实现所需功能。但由于一般微处理器中没有针对密码算法的特定运算指令，并且没有对密码算法运算进行优化处理，所以密码算法中诸如置换、SBox 查表等操作的运算速度很慢，影响了处理算法的效率。专用密码算法 ASIC 方式处理速度快，但灵活性差。密码算法和安全协议是不断发展的，密码算法种类的不断增加、密码标准的不断更新要求硬件能及时适应这种变化的需求。由于专用 ASIC 芯片不能做任何改变，要想支持新的算法、修改已有算法或适应新的密码标准，就必须重新设计生产全新的硬件，设计投资、非重复性工程（NRE）费用[4]等因素都给固定算法芯片的使用带来了很大的局限性。由于上述两种实现方式各有优劣，因此人们逐渐倾向于采用两者的折衷方案即用可重构计算结构来实现数据加密系统。

可重构计算技术近些年来得到了大力发展，国内外很多研究机构都提出并设计了相应的可重构计算系统。目前使用得最广泛的可重构计算系统是 FPGA，作为细粒度可重构计算系统的典型代表，它采用最基本的门阵列形式，允许用户使用触发器、小规模存储器和逻辑门实现大规模的数字电路。除此以外，还有许多其它针对特定的应用领域如 DSP、数据加密所设计的粗粒度可重构计算系统，它们采用大块（chunky）功能单元，如 ALU 和乘法器等，来实现数据计算。下面将对几种支持密码处理的粗粒度可重构计算系统进行简要地介绍。

PipeRench[5]是卡内基梅隆大学于 1998 年提出的一种基于线性阵列结构的可重构系统，其体系结构加强了处理大量混合宽度数据的计算能力。它的目标是开发具有数据流特征的数字信号处理和数据加密应用的可重构协处理器。目前 PipeRench 架构的应用领域主要是多媒体和密码处理的应用领域。

MorphoSys[6]可重构系统 1999 年由 Calirorlia 大学提出，是一个粗粒度的可重构 SOC 系统。该系统具有 16bit 的数据通路宽度、8×8 的可重构阵列以及一个类 MIPS 的处理器作为主处理器。计算模型采用了单指令流多数据流 SIMD 结构。主要面向高吞吐率和数据并行应用，适合于视频压缩、数据加密和目标识别。

RaPiD（Reconfigurable Pipelined Datapath）[7]是由华盛顿大学 Carl Ebeling 研究小组人员首先提出并设计的。RaPiD 是一种粗粒度、基于线性阵列的静态可重构结构，该体系结构对有规律的计算密集型任务有较高的性能，如在 DSP 中的脉动阵列计算。RaPiD 结构主要针对信号和图像处理领域进行了优化，因此在 DSP 的应用中表现出了较高的性能，同时对密码处理也有一定的支持。目前该结构已公布三个版本，分别为 RaPiD-I，RaPiD-Benchmark 和 RaPiD-Π。

COBRA（Cryptographic(Optimized for Block Ciphers)Reconfigurable Architecture）[8]是由美国马萨诸赛大学电子计算机工程系的 AJ Elbirt 和德国波鸿大学的 Christof Paar 研究了多种分组密码算法之后，提出的一种针对分组密码运算的硬件可重构密码体系结构。COBRA 系统支持 64 位或 128 位的输入分组长度，支持 32 位数据路径之间的数据互联。该系统实现了 Rijndael、SERPENT 以及 RC6 等算法。

REMARC[9]是 1998 年由斯坦福大学计算机系统实验室开发的可重构多媒体阵列协处理器(Reeonfigurable Multimedia Array Coproeessor)。REMARC 系统面向数据并行和高吞吐率应用。该系统由可重构协处理器与 MIPS-ⅡRISC 处理器紧耦合而成。16 位的处理元“ nanoprocessor ”构成了一个 8×8 的阵列，由全局控制单元“ global control unit ”对其进行控制。通信资源包括相邻四个处理元的互连和允许同行或同列播送的额外 32 位水平、垂直总线。同时，通信资源支持每个周期将全局程序计数值播送到处理元中，处理元根据程序计数值对其功能部件进行配置，完成 SIMD 类操作。

目前，国内一些大学针对密码处理的可重构处理结构也进行了研究，取得了一定的成果。国防科技大学在可重构密码处理结构方面做了较为深入研究，提出并设计了可重构层次互连密码处理结构 RHCA[10]。北京科技大学研制了 RELOG_DIGG[11]系统，可实现部分分组密码算法以及序列密码算法。从目前可重构密码处理体系结构研究现状来看，一些可重构计算体系结构虽然能实现一部分分组密码算法，但对分组密码算法支持的广泛性不够，可用性和适用性不太强。此外，大部分可重构计算体系结构主要面向多媒体处理、无线通讯等领域的，并不专门针对密码处理。而具有可重构单元的增强型处理器虽然对一些密码算法处理有一定的加速，但难

以达到很高的性能。另外，在可重构密码处理结构设计中，缺乏系统的设计方法和具有通用性的模型，结构设计差异性较大。针对上述现状和存在的问题，本文结合可重构体系结构设计方法和分组密码处理应用，研究专门针对分组密码处理应用的可重构结构，使其具有广泛的适应性和代表性。因此，怎样结合可重构体系结构与分组密码处理应用并最终实现一种灵活支持大多数分组密码算法的可重构处理结构成为本课题研究的重点。

1.3 可重构密码处理的研究意义

密码处理芯片是构建信息安全系统的基础和核心部件。采用 ASIC 技术设计的专用密码芯片具有速度快的特点，但灵活性差，密码芯片一旦设计完成，科研人员就无法对其进行二次开发，密码芯片兼容性差、扩展性差，在一定程度上造成互连互通困难，且在设计上存在安全性问题。随着通信技术和网络技术的飞速发展，人们对信息系统安全性的要求越来越迫切，保障信息系统安全所需投入的处理资源将越来越多。由此可以预计，采用可重构密码处理结构来实现高效灵活的密码算法芯片正逐渐成为研究的一个新的热点，其安全应用的范围也会越来越广。可重构密码处理在信息安全这一应用领域的优势体现在以下几个方面：

◆ 高效性

在网络通信中，对网络数据包进行加解密需要大量计算资源。密码算法的计算周期、密钥长度以及计算复杂性都在不断增大。密码处理平台的计算能力直接决定密码算法能否顺利应用。可重构密码处理更容易匹配各种密码处理模式，性能上远优于通用微处理器。

◆ 灵活性

现代安全协议都是独立于算法的，实际使用的密码算法可能基于会话进行协商的，如 IPSee 允许的算法就有：DES、3DES、Blowfish、CAST、IDEA、RC4、RC6、Diffie-Hellman、ECDH 等[12]。固定算法 ASIC 要支持众多算法代价非常高。可重构密码处理可以根据通信会话协商的结果改变硬件来实现相应的算法，使同一硬件能够高效地支持种类繁多的算法，具有很大的灵活性。另外，在密码算法处理过程中经常需要把初始密钥扩展成轮密钥来参与计算，可重构密码处理能够根据特定的密钥扩展定制硬件，使算法执行更加灵活。

◆ 扩展性

密码算法和安全协议是发展的，密码标准也在更新；安全协议中算法数量在增加，很多算法有不同的变型；现有算法也有被攻破和失效的可能，新的更安全的算法会被提出。可重构密码处理能及时适应上述变化，而固定算法 ASIC 灵活性差，重新设计生产全新硬件的设计投资和工程费用使其局限性很大[13]。

◆ 安全性

可重构密码处理上密码算法的设计和芯片的设计与生产过程是分离的，可重构密码处理芯片在使用之前是不含密码算法的裸片，只有在使用时才由用户将设计好的密码算法装载到芯片

上，因此在芯片的设计和生产过程中不会泄露任何密码算法的信息[14]。此外，在使用中可重构密码处理可以方便地支持各个通信部门间协议协商、更改算法，从而减小失密的可能性。

综上所述，可重构密码处理由于它在设计上具有安全性、兼容性、可扩展性、高效性等众多优点，解决了传统密码处理芯片研制过程中的安全隐患以及灵活性差的问题。因此可重构密码处理在信息安全领域具有良好的应用前景[15]，可被广泛应用于保密通信、网络终端加密设备等领域，因此该研究方向具有重要的政治、军事和经济意义。

1.3 本文研究内容及主要研究成果

1.3.1 本文研究内容

本课题遵循可重构结构的设计思想和方法，分析了分组密码处理结构的特点，研究了一种面向分组密码算法的可重构系统以及可重构密码运算模块的实现原理。论文内容主要包括以下几个方面：

第一章 绪论。介绍了论文研究的目的和意义、国内外研究现状以及本文的结构安排。

第二章 分组密码算法分析。介绍了密码算法的基本原理和分类，分析了分组密码算法的基本运算模块，并对现有的主要分组密码算法结构特点进行了研究与分析。

第三章 可重构密码处理系统及可重构运算模块研究与设计。设计了一种可重构的密码处理结构模型 RCPA（Reconfigurable Cipher Processing Architecture），同时对可重构密码运算模块 RCM（Reconfigurable Cipher operating Module）进行了研究与设计。

第四章 DES/AES 可重构密码处理结构实现与算法验证。基于可重构密码处理结构模型 RCPA，具体针对 DES 和 AES 算法，实现了 DES/AES 可重构密码处理结构，并分别对 DES 和 AES 进行了算法映射，给出了仿真波形和性能分析。

第五章 高性价比 AES 密码系统的 VLSI 实现。本章独立成章，介绍本人课题的其他相关研究成果：研究了一种高性比的 AES 密码系统 VLSI 实现方案，同时解决了高速加/解密时轮密钥分配的同步问题。此实现方案设计的 AES 密码系统与其他硬件实现方案比较，具有更高的性价比。

第六章 总结与展望。对本人的研究工作进行了系统总结和概括，针对目前工作的不足和进一步的研究提出了建议。最后，对未来的研究方向做了展望。

论文的结尾是参考文献和致谢部分。

1.3.2 主要研究成果

（1）研究并设计了一种可重构密码处理结构 RCPA。该结构采用流水线设计，提高了密码处理的速度，增大了密码处理的吞吐率。

（2）完成了可重构密码运算模块 RCM 的设计和实现。主要包括 S 盒替代、移位模块、置换模

块、模加/减和模乘等密码运算模块。这些基本运算模块可根据 RCM 配置指令进行重构，灵活完成不同算法所需的运算功能。

（3）针对 DES 和 AES 算法，采用 VHDL 硬件描述语言，完成了 DES/AES 可重构密码处理结构的实现和算法验证工作。仿真波形和实验结果表明所设计的可重构密码处理结构不仅能够正确实现 AES 和 DES 算法，而且具有较快的密码处理速度和较少的配置位。

（4）研究了一种高性价比的 AES 加/解密系统的 VLSI 实现方案。对 AES 的两个核心运算部件（字节代换和列混合）进行了精心的硬件可逆设计来减少面积开销；采用了轮间和轮内相结合的流水线结构来提高整个系统的加/解密速度；设计了一种轮密钥扩展结构来解决高速加/解时轮密钥分配的同步问题。与其他同类设计进行了比较，实验结果表明此设计具有更高的性价比。

第二章 分组密码算法分析

2.1 密码算法简介

在信息领域中，密码技术是保护信息安全的关键技术，它是集数学、通信技术和计算机科学等学科于一身的交叉学科。密码技术从根本上来说是密码算法的使用，是安全的基础和保障。密码算法的目的是为了保护信息的保密性、完整性和安全性，简单地说就是信息的防伪造与防窃取[16]。

1. 密码算法原理简述

密码系统由密码算法以及所有可能的明文、密文和密钥组成。伪装前的信息被成为明文，用某种方法伪装信息以隐藏它的内容的过程称为加密，被加密的消息称为密文，而把密文转变为明文的过程称为解密。用于加密和解密的数学函数称为密码算法。密钥是密码算法的加解密参数，所有算法的安全性都基于密钥的安全性。

整个密码系统可以用数学符合描述：S = (P,C,K,E,D)，其中P是整个明文空间，C是密文空间，K是密钥空间，E是加密算法，D是解密算法。当给定的密钥为k∈K，这时加密/解密算法表示为E_k/D_k，这样，加密/解密函数表示为（如图2.1）：

$E_k(P) = C$　(表示用加密算法对明文P加密得到密文C)

$D_k(C) = P$　(表示用解密算法对密文C解密得到密文P)

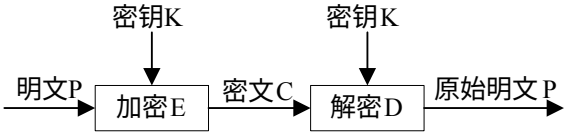


图 2.1　数据加密/解密的基本过程

这些函数还具有下面的特性：

$$D_k(E_k(P)) = P \tag{2-1}$$

这表明加密变换E_k和解密D_k变换是可逆的。

2. 密码算法分类

根据密码算法所使用的加密密钥和解密密钥是否相同，能否由加密过程推导出解密过程，或由解密过程推导出加密过程，可将密码算法分为对称密码算法(也称单钥密码算法、秘密密钥密码算法、对称密钥密码算法)和公开密钥密码算法（也称双钥密码算法、非对称密码算法、非对称密钥密码算法），除此之外，密码学中还较多使用Hash函数作为辅助的加密算法。

（1）对称密码算法

如果一个密码算法的加密密钥和解密密钥相同，或由其中一个很容易推导出另一个，该算法就是对称密码算法。对称密码的特点是速度快、安全强度高，主要用作数据加密。

对称密码根据加密模式又可分为分组密码和序列密码。分组密码的典型算法有：DES，IDEA和AES等，分组密码是目前在商业领域比较重要、使用较多的密码，广泛用于信息的保密传输和加密存储；序列密码的典型算法有：RC4，SEAL，A5等，序列密码多用于流式数据的加密，特别是对实时性要求比较高的语音和视频流的加密传输。

（2）公开密钥密码算法

如果一个密码算法的加密密钥和解密密钥不同，或者由其中的一个推导不出另一个，则该算法就是公开密钥密码算法，简称公钥密码算法。使用公钥密码的每一个用户都拥有基于特定公钥密码算法的一个密钥对(e,d)，公钥e公开，公布于用户所在系统认证中心(CA)的目录服务器上，任何人都可以访问，私钥d为所有者保管并严格保密，两者不相同且互为对方的解密密钥。

公钥密码的典型算法有：RSA，ECC，DSA，EIGamal，Diffie-Hellman(DH)密钥交换算法等。公钥密码能够用于数据加密、密钥分发、数字签名、身份认证、信息的完整性认证、信息的非否认性认证等。其中用于数据加密的算法有：RSA，ECC，EIGamal等；可以用于密钥分发的算法有：RAS，ECC，DH等；可以用于数字签名、身份认证、信息的完整性认证、信息的非否认性认证的有RSA，ECC，DSA，EIGamal等。

（3）单向密码算法

单向密码体制使用单向的散列(Hash)函数，它是从明文到密文的不可逆函数，也就是说只能加密不能还原。单向散列函数H作用于任意长度的信息M，返回一固定长度的散列值（也称摘要信息）h=H(M)。

典型的散列函数有：MD5，SHA-l，HMAC等。单向散列函数主要用在一些只需加密不需解密的场合：如验证数据的完整性、口令表的加密、数字签名、身份认证等。

2.2 分组密码算法的基本运算模块分析

任何一个密码算法都是由一系列的基本运算模块按照一定顺序连接而成的，通过对大量的密码算法进行分析和研究，我们发现密码算法具有一个显著的特征：很多不同的密码算法具有相同或相似的基本运算模块，或者说同一基本运算模块在不同算法中出现的频度很高。那么这些基本运算模块所对应的硬件资源就可以被多种不同的密码算法所共用，因此我们就能够以较少的电路规模构造一套逻辑电路来实现多种算法。

表2.1列出了我们对3DES、IDEA、AES侯选算法等34种典型的分组密码算法的统计结果[12,14,17,19,22]。

现将分组密码的操作特点归纳如下所示：

（1）分组密码运算操作大多是无符号的整数操作，包括多种位逻辑运算，算法中没有使用浮

点或定点类型。Feistel 网络结构和 SP 网络结构类算法的操作序列大都包括 S 盒代替、置换、异或、模乘、模加/减、移位等运算；LM 结构的代数群运算类算法更着重于算术运算，主要运算为模加/减、模加/减逆、模乘、模乘逆、异或等。分组密码中使用频率高的运算有：异或、S 盒替换、移位运算、模加/减运算、模加/减逆运算、置换运算、模乘运算、模乘逆运算。

表 2.1　分组密码算法基本运算模块使用频度统计

基本运算模块	使用频度
异或运算	100%
S盒变换	50%
移位运算	58.82%
置换运算	29.41%
模加运算	44.12%
模减运算	8.82%
模加逆运算	2.94%
模乘运算	26.47%
模乘逆运算	2.94%
逻辑非运算	11.76%
逻辑与运算	11.76%
逻辑或运算	8.82%
指数运算	8.82%
对数运算	5.88%

（2）分组密码算法中大量使用了逻辑移位和循环移位，其移位模式既有固定移位模式（每次移位位数固定不变）又有可变移位模式（每次移位位数依赖寄存器的值），移位的位数一般从 l 位到 32 位，移位的宽度大多为 16、28、32、64、128。

（3）分组密码算法涉及的算术运算（乘、加/减）和逻辑操作（异或、与、或等）位宽大多是字节或字节的整数倍，乘法操作多为 32 位位宽，其它算术运算也以 8、16、32 位位宽者居多。且算术运算大多带有取模操作，取模操作多是 2 的幂次，多为 2^8、2^{16}、2^{32}。

（4）分组密码算法中常常需要几个相同或不同的 S 盒进行并行运算。根据变化程度和使用方式的不同，分组密码使用的 S 盒可分为两种方式：一种是每轮操作使用相同的 S 盒，如 DES、AES、SAFER、NSSU、Blowfish；另一种是每轮操作使用不同的 S 盒，如 SERPENT。S 盒代替的模式常见的有 4*4S 盒、6*4S 盒、8*8S 盒、8*32S 盒，目前以 8*8S 盒最为流行 [18]。

（5）大数据位宽才能有效扩散输出对输入的依赖性，因此分组密码算法中采用的置换操作位宽都比较大。128 比特的置换在分组密码设计中也较为常见。

2.3 分组密码算法处理结构及其特点

目前分组密码所采用的整体结构可分为 Feistel 网络结构，SP 网络结构及 LM 结构。Feistel 结构[19]由于 DES 的公布而广为人知，已被许多分组密码所采用。Feistel 结构的最大优点是容易保证加/解密相似，但扩散较慢。而 SP 结构比较难做到这一点，但是 SP 结构的扩散特性比较好。

LM 结构则基于不同代数群的混合运算来设计的。

1. Feistel网络结构特征分析

很多分组密码算法都是 Feistel 网络或扩展 Feistel 网络型结构，其典型代表是 DES，它体现了现代密码学理论设计密码算法的原则。取一个长度为 n 的分组，然后把它分成长度为 n/2 的两半部分：L 和 R，当然 n 必须是偶数。可以定义一个迭代型的分组密码算法，其第 i 轮的输入取决于前一轮的输出：

$$L_i = R_{i-1} \tag{2-2}$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i) \tag{2-3}$$

K_i 是第 i 轮使用的子密钥，f 是任意轮函数。常见的 Feistel 网络型分组密码代表有 DES、CAST、Twofish、RC6 等。Feistel 网络型结构保证了它的可逆性，异或被用来合并左半部分和轮函数的输出，它满足：

$$L_{i-1} \oplus f(R_{i-1}, K_i) \oplus f(R_{i-1}, K_i) = L_{i-1} \tag{2-4}$$

因此，只要在每轮中 f 的输入能重新构造，使用了该结构的密码就可保证它是可逆的。它不管 f 函数如何，也不需要它可逆。我们能将 f 函数设计成我们希望的那样复杂，并且不必实现两个不同算法（分别用于加/解密），Feistel 网络的结构将自动实现这些。总的说来：Feistel 网络的优点在于加解密结构的相似性，缺点在于扩散较慢，如算法至少需要两轮才能改变每一个比特。该结构如图 2.2 所示。

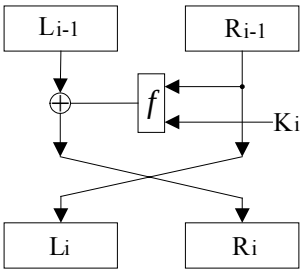


图 2.2 Feistel 网络结构图

2. SP网络结构特征分析

SP 网络是 Feistel 网络的一种推广，SP 网络中的 S，P 分别是 Substitution 和 Permutation 的首字母，SAFER、SHARK、Rijndael 等著名密码算法都采用此结构。每一轮中，首先轮输入被作用于一个由子密钥控制的可逆函数 S(非线性变换)，然后再被作用于一个置换（或一个可逆的线性变换）P。S 一般被称为混淆层，提供了分组密码算法所必须的混淆作用[20]。P 一般被称为扩散层，主要是提供快速的扩散，实现雪崩效应。该结构中轮函数一般由 S 盒层和 P 置换(或线性变换)组成，分别提供必要的混淆性和扩散性，所以通常也称 S 盒层为混淆层，称 P 置换（或线性变换）为扩散层。新的 AES 算法 Rijndael 就采用了 SP 网络，还有 SAFER+，CRYPTON，

SERPENT 等分组密码也采用了此结构。SP 网络的优点在于与 Feistel 网络相比可以得到更快的扩散速度，其缺点是加解密很难达到一致。该结构如图 2.3 所示。

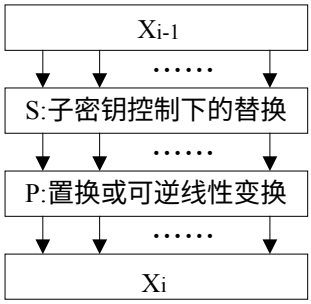


图 2.3　SP 网络型结构

3.　LM网络结构特征分析

从操作特性上看，分组密码中的另一类密码算法是基于不同代数群的混合运算来设计的，多由算术运算构成，这类算法以 IDEA 为代表，还包括 MMB 等。Xuejia Lai 和 James Massey 在 1990 年公布的 PES 和 1991 年公布的 IPES 即 IDEA 算法[21]在结构上为图 2.4 所示，它与 Feistel 网络和 SP 网络都有区别，把这种类型的密码单独归为 LM 结构，这类密码的混乱与扩散都是来自于 MA 结构，其中采用了混合不同群的运算。当 Sl、S2、S3 和 S4 用 Tl、T2、T3 和 T4 代替时，图 2.4 中所示的 MA 结构的输入不变，Z5 和 Z6 不变时，MA 结构的输出也不变，这种结构保证了加解密是相似的，其安全性基于 MA 结构。由于 MA 结构采用的数学运算较多，因此软件实现时，在一部分处理器上的效率可能不是太高，比 Rijndael 和 Twofish 算法的实现效率低，但是这样的设计使得攻击者对算法本身进行分析变得困难。

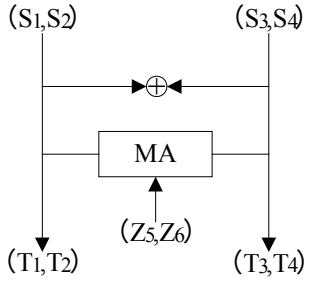


图 2.4　LM 型结构中的轮函数

4.　分组密码算法处理结构特点

从以上对分组密码的分析，可归纳出分组密码处理结构具有的明显特点，具体体现为：

（1）分组密码算法处理结构具有两个方向的并行性。在纵向上即多个分组流方向，在执行 ECB 模式下的密码算法只要硬件提供足够深度的流水支持，每个分组都可以并行处理。若密码算法以其它反馈模式执行，利用交错技术（interleave）[22]可以使相邻若干分组的处理并行执行，反馈可以间隔很多并行执行的分组进行，体现出很大的并行潜力。在一个分组的处理方向（横

向），随着密码处理安全强度的增加，分组密码的分组位宽具有越来越大的趋势（64 位~128 位），分组的一个处理步通常可分成几个较小位宽的子块进行操作。在软件实现中，由于机器位宽的限制，这些操作要串行执行，硬件实现中可以开发一个分组处理的内在并行性，分组密码处理具有横向上适度并行的能力。

（2）分组密码每轮变换中的操作之间存在前后数据相关，各轮变换之间也涉及明显的数据相关。数据相关使算法流程中很多操作必须串行执行，这种特性使单个算法的横向并行性受到限制。

（3）分组密码算法都经过多轮完成，每轮的操作基本相同。轮运算中的串行操作序列是线性序列，各操作之间没有反馈，控制简单，易于功能分割和时序划分，结合分组密码算法的纵向并行性特点，分组密码算法处理非常适合流水执行。

（4）分组密码算法通常将较大分组（64/128 位）拆分为较小的子块（8~32 位）进行计算，算法操作的位宽都是字节的整数倍，其中 32 位的运算宽度最为常见。实现统一的密码算法处理器时，则需要较为规整的数据通路和控制通路，因此选择计算位宽 32 比特能够匹配大多数算法的要求。

2.4 本章小结

本章对现有的主要公开分组密码算法处理结构特点和基本运算模块操作特征进行研究与分析，分析表明分组密码算法具有较为规律的运算位宽，适合分组内的横向并行以及分组间的纵向流水处理；总结了分组密码算法的基本运算模块的操作特征，包括了基本逻辑运算、移位运算、置换运算、S 盒代替运算、模乘运算、有限域乘法运算以及模加/减运算。归纳了这些基本运算模块在常见分组密码算法中的使用情况。为本文后续章节的可重构密码处理结构的研究与设计提供了依据，奠定了基础。

第三章 可重构密码处理系统及可重构密码运算模块设计

在第二章中，我们对分组密码算法的基本运算模块进行了分析，其中有的运算模块是重构运算模块，有的只是单独的算法在用，下面我们以此为基础来介绍整个可重构密码处理系统和系统中各可重构运算模块的设计。

3.1 可重构密码处理系统设计

可重构密码处理系统的设计实际上就是要设计一个可重构密码处理器[23]，而可重构密码处理器是采用可重构体系结构设计而成的用于对数据进行加/解密处理的集成电路芯片，其内部的逻辑电路能够根据不同密码算法的需求，重新组织，构成不同的电路结构，实现不同的功能，从而能够灵活、快速地实现多种不同的密码算法。可重构密码处理器是一种创新性的密码芯片，它很好地克服了传统的密码芯片只能实现特定密码算法的弊端，使得密码使用者能够在它上面很方便地更改密码算法或者设计新的密码算法，从而大大提高了密码系统的灵活性、安全性和扩展性。本节采用从整体到部分的方式来介绍整个可重构密码处理系统的设计。

3.1.1 可重构密码处理系统的组成

可重构密码处理系统实际上是一个用于对数据进行加/解密处理的协处理器，因此它除了具有一般的处理器都有的控制模块和存储模块之外，还具有一个用于进行加/解密运算的密码处理模块，而这个密码处理模块是采用可重构体系结构设计而成的，称之为可重构密码处理模块RCPM（Reconfigurable Cipher Processing Module）。图 3.1 给出了可重构密码处理系统的总体结构框图。

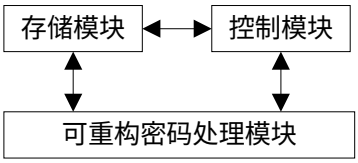


图 3.1　可重构密码处理系统的总体结构框图

3.1.2 存储模块

存储模块是可重构密码处理系统的重要部件，负责对加解密算法中的密钥、配置数据、常数以及运算中的临时数据进行存储，可作为数据传输的中间站。存储模块既可以采用 RAM 实现又可以使用寄存器堆来实现。RAM 的优势在于可以提供大容量的存储，而且占用的资源较少。不足之处是其速度不高。寄存器堆的优势在于速度快，但其占用电路面积较大，因此不适于做大容量的存储。存储模块在设计中需要考虑实际的需求情况来选择 RAM 或寄存器堆来实现。

3.1.3 控制模块

控制模块负责对整个可重构密码处理系统的处理过程进行控制。其功能主要包括以下三部分：

（1）完成对可重构密码处理模块的配置操作。

可重构密码处理模块内部的可重构运算模块以及互连模块中都设置独立的专有配置寄存器。在系统需要配置时，通过控制模块将配置信息注入到各个可重构运算模块以及互连模块内的专有配置寄存器，控制模块只需要动态产生配置寄存器的地址信号就可完成配置的执行。

（2）解决加/解过程中密钥的同步问题。

（3）实现可重构密码处理系统与外部的数据交换。

外部的运算数据由输入数据缓冲区写入到存储模块中，运算完毕的数据由存储模块输出至输出数据缓冲区。

3.1.4 可重构密码处理模块的组成与结构设计

1. 可重构密码处理模块的组成

可重构密码处理模块的设计目标是能够灵活、快速地实现多种不同的密码算法，因此它应该包括多种不同密码算法所需要的基本密码运算模块和灵活可变的内部互联网络，可重构密码处理模块的原理图如图 3.2 所示。

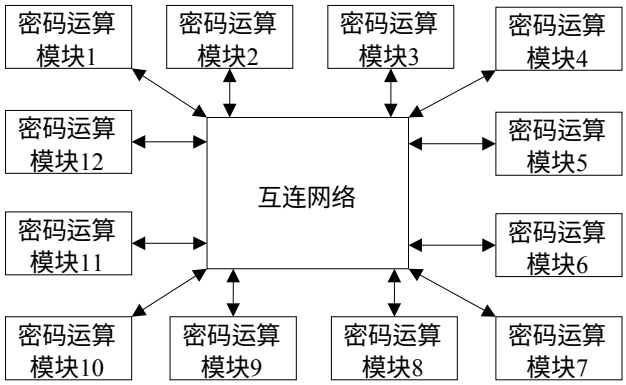


图 3.2　可重构密码处理模块的原理图

2. 可重构密码处理模块结构设计

根据密码处理适于分组（或分块数据）内并行、分组间并行或流水的特性，本文采用寄存器间接相连型连接网络来设计可重构密码处理模块，其结构如图 3.3 所示。

图 3.3 中的 RCM 表示可重构密码运算模块，REG 表示寄存器，mux 表示多路器，DBUS 表示数据总线。如果要实现更多的算法，可以根据需要扩充密码运算模块。

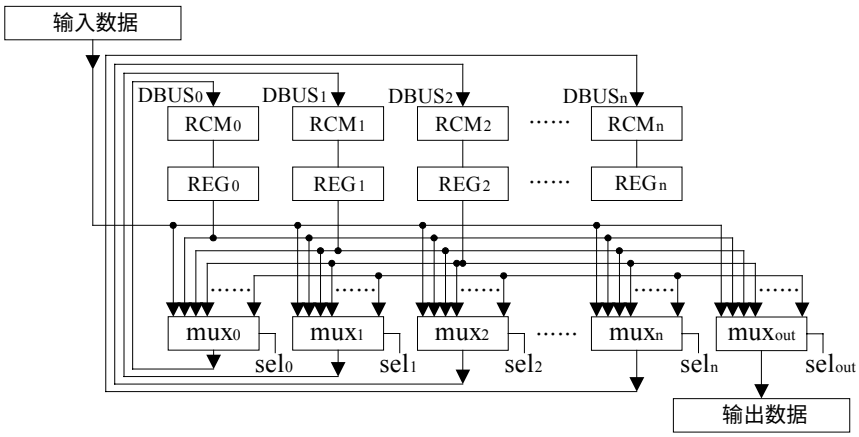


图 3.3　可重构密码处理模块的结构图

3.2 可重构密码运算模块设计

可重构密码运算模块 RCM 是可重构密码处理模型 RCPA 中的运算功能模块，是模型设计的最关键部分。其设计原则是，使每个可重构密码运算模块 RCM 能够提供最大的灵活性，满足尽可能多的分组密码运算需求。可重构密码运算模块 RCM 的设计思想是：在可重构密码运算模块 RCM 中设置某些可被重复配置的功能可控节点，通过对可控节点的动态或静态配置从而改变可重构密码运算模块 RCM 内部的电路结构，满足不同密码算法的应用需求。本节将对 RCPA 模型中的可重构密码运算模块 RCM 硬件实现原理进行研究。重点介绍了 S 盒替代模块、可重构移位模块、可重构置换模块、模加/减运算模块、模乘运算模块等可重构密码运算模块 RCM 的设计方法。

3.2.1 S 盒替代模块设计

3.2.1.1 设计方法

S 盒是许多分组密码算法中唯一的非线性部件，提供了算法所需要到混淆作用[20]，此外，S 盒的规模占算法规模的 30％[33]以上。因此它的密码强度决定了整个密码的安全强度。S 盒首次出现在 Lucifer 算法中，随后因 DES 的使用而广为流行。S 盒本质上均可看作从二元域 F_2 上的 n 维向量空间 F_2^n 到 F_2 上的 m 维向量空间 F_2^m 的映射 $S(x) = (f_1(x), \cdots, f_m(x)) : F_2^n \rightarrow F_2^m$，通常简称 S 是一个 n*m 的 S 盒。几乎所有的 S 盒都是非线性的，当参数 m 和 n 选择很大时，算法的抗攻击性越强。但反过来，m 和 n 过大将使 S 盒的规模扩大，给设计带来困难，而且增加硬件电路的存储资源。

分组密码算法中常常需要并行查找相同或不同的 S 盒，S 盒代替的输入粒度差别较大（4~13 位），以不大于 8 位者居多。通过对大量的分组密码算法分析可以得出 S 盒查找表方式常见有 8*8、8*32、4*4、6*4 四种模式。文献[14]给出了不同 S 盒查找表模式的使用频率，如表 3.1 所示。

表 3.1　S 盒查找表模式使用频度表

查找表模式	8*8 S 盒	8*32 S 盒	4*4 S 盒	6*4 S 盒	其他模式
使用频度	33.33%	30%	13.33%	6.66%	16.66%

根据以上的分析可知，8*8、8*32、4*4、6*4 四种查表模式占到了整个 S 盒查找表方式的 83.33%。因此，本文在 S 盒代替电路设计上考虑支持 8*8、8*32、4*4、6*4 四种查表模式。

目前可重构 S 盒替代电路硬件实现主要有两种方式：基于布尔函数实现方式和基于查找表 LUT 方式。下面将分别对这两种实现方式进行讨论。

（1）基于查找表 LUT 方式

该方式将 S 盒替代结果提前计算出来，得到算法的查找表，输入数据以查表的方式进行替代变换。为实现 S 盒的可重构性，基于查找表 LUT 方式的 S 盒替代电路通常是利用 RAM 来实现。对于实现 n*m 的 S 盒，RAM 需要有 n 位地址线，输出 m 位，占用 $m\times 2^{n}$ 大小的存储容量。使用查找表 LUT[24,25]方式具有结构简单容易实现，处理速度快，易于配置等优点，但其缺点是占用大量的存储资源使电路面积增大，此外，用查找表实现的可重构 S 盒通过读地址写数据的方式进行重构，大量的时间花费在数据读写上，从而降低了 S 盒重构速度。

（2）基于布尔函数实现

基于布尔函数实现就是把 S 盒替代变换等价为一组布尔逻辑函数，S 盒的输入就是布尔函数的自变量，S 盒的输出就是函数值。

布尔函数实现的原理如下，设输入为 $x=\sum_{i=0}^{n-1}x_i 2^i=(x_{n-1},x_{n-2},\Lambda\ ,x_1,x_0)_2$ 的 n*m S 盒布尔函数的通式[11]为

$$f(x)=\sum_{i=0}^{2^n-1}c_{\mathrm{i}}\cdot p_{\mathrm{i}}(x) \tag{3-1}$$

其中，系数 $c_{\mathrm{i}}\in\{0,1\}$，$i=\sum_{\mathrm{k}=0}^{n-1}i_{\mathrm{k}}2^{\mathrm{k}}=(i_{\mathrm{n}-1},i_{\mathrm{n}-2},\Lambda\ ,i_1,i_0)_2$。通项 $p_{\mathrm{i}}(x)$ 为

$$p_{\mathrm{i}}(x)=\prod_{l=0}^{n-1}x_l^{\mathrm{i}_1}=x_{\mathrm{n}-1}^{i_{n-1}}x_{\mathrm{n}-2}^{i_{n-2}}\Lambda\ x_2^{i_2}x_1^{i_1}x_0^{i_0} \tag{3-2}$$

从式（3-1）中可以看出当输入数据后，$p_{\mathrm{i}}(x)$ 作为最小项就定下来了，对输出 $f(x)$ 产生影响的只有最小项系数 c_{i}，通过改变 c_{i} 就可以使 S 盒实现任意布尔函数的变换。该方法实现的优点在于减少了电路存储资源，有效降低了电路的规模。但该方式缺点在于电路设计较为复杂。

综合上述的两种实现方法，本文以节约硬件开销为目标，选择基于布尔函数实现方式。具体电路设计见下一节。

3.2.1.2 电路设计

本节利用重构思想和硬件复用技术设计了一种可在 4*4 和 6*4 规格上选择的可重构 S 盒原

胞结构，通过设置不同配置位构造出不同的 4*4 或 6*4 S 盒。而且以本文的 4*4 和 6*4 规格的可重构 S 盒为基本单元，能够构造更大规模的 S 盒，比如通过 2 个 4*4 的 S 盒能够构造 8*8 S 盒，通过 4 个 8*8 S 盒能够构造 8*32 S 盒。

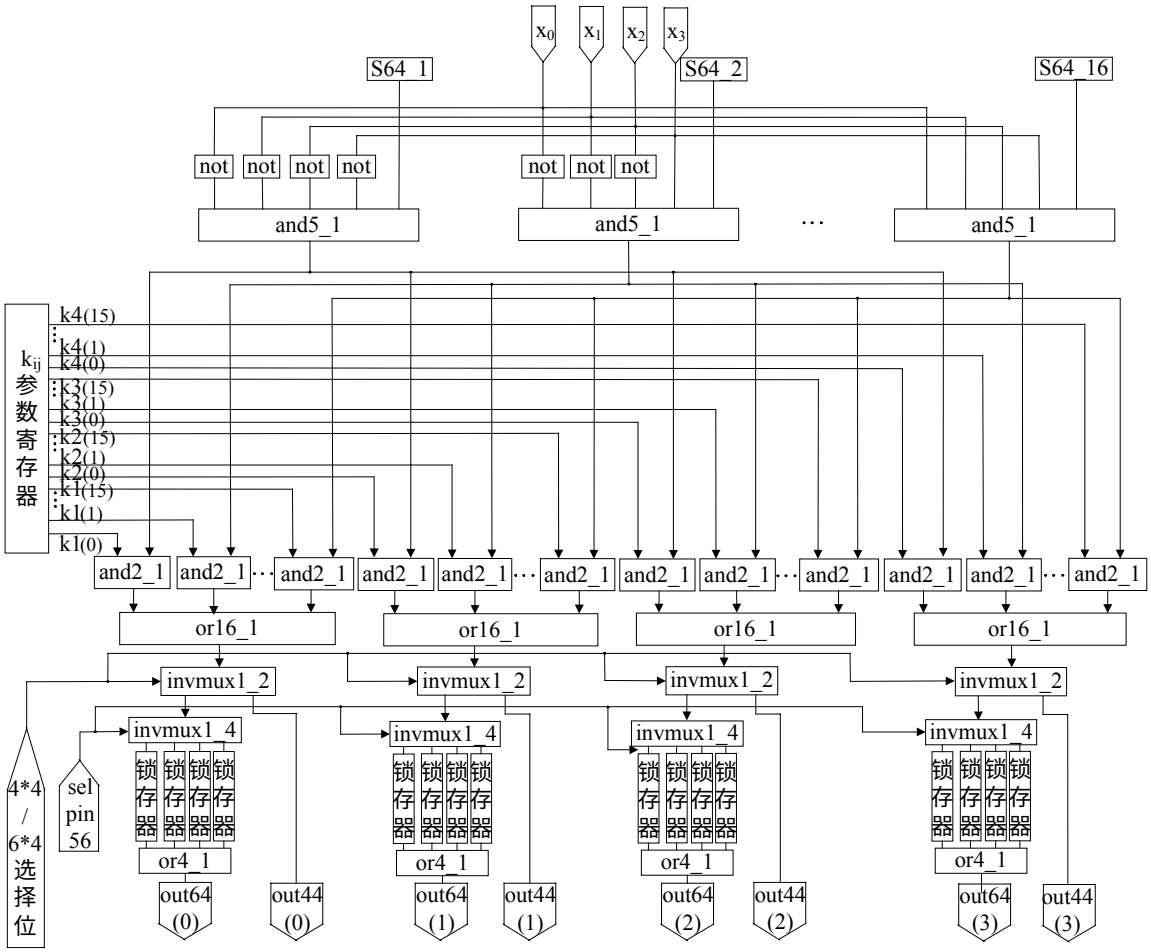


图 3.4　4*4/6*4 可重构 S 盒原胞结构图

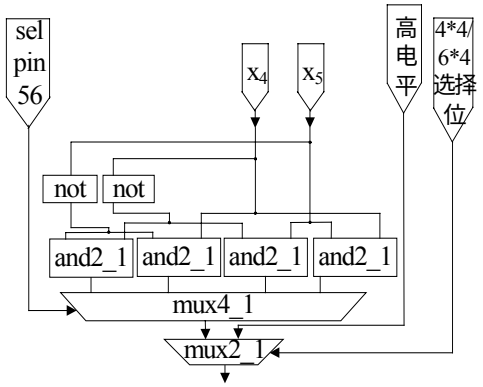


图 3.5　4*4/6*4 选择单元硬件结构图

本节所设计的 4*4/6*4 可重构 S 盒原胞结构如图 3.4 所示，图中 k_{ij} 参数寄存器存储了 S 盒

的配置位，通过改变配置位实现 S 盒的重构性；S64_i(1≤i≤16)是 4*4/6*4 选择单元，负责决定构造的是 4*4 S 盒还是 6*4 S 盒。其硬件结构如图 3.5 所示，“4*4/6*4 选择位”引脚负责决定构造的是 6*4 S 盒还是 4*4 S 盒，当“4*4/6*4 选择位”引脚为‘0’时，进行 4*4 S 盒的构造，当“4*4/6*4 选择位”引脚为‘1’时，进行 6*4 S 盒的构造；x_0、x_1、x_2、x_3、x_4、x_5 为 6 位输入数据，out44 为 4*4 S 盒的输出，而 out64 为 6*4 S 盒的输出；selpin56 是在进行 6*4 S 盒构造时，输入数据 x_4、x_5 的最小项选择位；invmux1_2 是一个二选一数据分配器（输入为 in，输出为 out0 和 out1，当选择信号 sel 为‘0’时将输入 in 传输到 out0，而当选择信号 sel 为‘1’时将输入 in 传输到 out1）；invmux1_4 是一个四选一数据分配器。

3.2.1.3 仿真验证与实验结果分析

本文所有重构运算模块的仿真模型都是用 VHDL 硬件描述语言在 Xilinx 的 FPGA VirtexE xcv3200e-6 上建立的并用 ISE6.2 软件进行了综合、布局布线，最后在 Mentor 公司的 ModelSim SE6.0 上进行了仿真验证。

图 3.4 中可重构 S 盒电路占用 FPGA 资源和最大组合路径延时情况如表 3.2 所示，为了验证可重构 S 盒功能的正确性，以 DES 的 S1 盒和一个 4*4 的 S 盒为例为例，表 3.3 给出了 DES S1 盒的定义，图 3.6 是其相应部分数据的仿真波形。表 3.4 给出了一个 4*4 S 盒的定义（命名为 S0），图 3.7 是其相应部分数据的仿真波形。

表 3.2 4*4/6*4 可重构 S 盒占用 FPGA 资源情况

Number of Slices	47 out of 32448
Number of Slice Flip Flops	12 out of 64896
Number of 4 input LUTs	84 out of 64896
Number of bonded IOBs	81 out of 808
Equivalent gate count	558
Maximum combinational path delay	21.333ns

表 3.3 DES S1 盒的定义

行＼列		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S1	0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

图 3.6　DES S1 盒部分数据的仿真波形

图 3.6 中 x[5:0]为 S 盒的 6 位输入，out64 为 6*4 S 盒（这里指 DES 的 S1 盒）的 6 位输出，而 out44 为 4*4 S 盒的 4 位输出，k1[15:0]、k2[15:0]、k3[15:0]、k4[15:0]为其配置位。DES S 盒 6 位输入 x[5:0]中，x_0x_5 确定行数，而 $x_1x_2x_3x_4$ 确定列数。由图 3.6 可以看出，当输入 x[5:0]为 100101 时，输出为 0100，由 x_0x_5 为 11 确定为第 3 行，由 $x_1x_2x_3x_4$ 为 0100 确定为第 4 列，查表 3 可得输出应为 4，二进制为 0100；同理，当输入为 111110，输出为 1000，当输入为 110101 时，输出为 1001，其余数据也可以得到相应验证。

表 3.4　S0 盒的定义

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S0	7	7	10	14	12	11	11	8	15	8	1	4	2	9	11	2

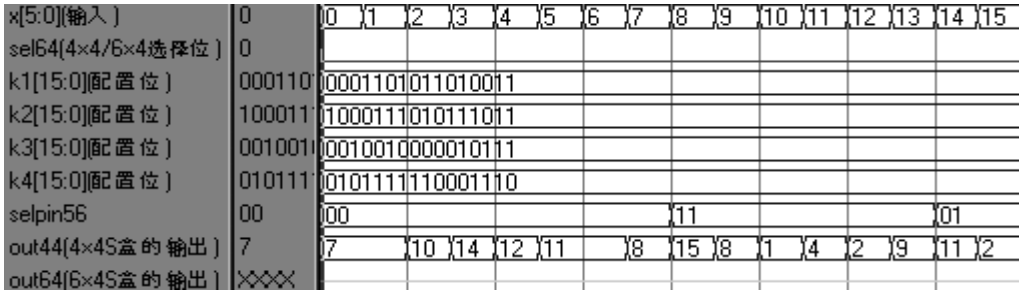


图 3.7　S0 盒仿真波形

同样，图 3.7 说明了本文所设计的可重构 S 盒电路结构能够正确实现 4*4 的 S0 盒。由此，验证了所设计的可重构 S 盒替代模块功能的正确性。

3.2.2 可重构移位模块设计

3.2.2.1 设计方法

1.　分组密码算法中的移位操作

移位操作是密码算法中常用的运算，无论是在轮运算还是在子密钥的生成运算过程中，都较为常用。通过对分组密码算法的分析知道，移位运算主要有逻辑移位和循环移位。循环移位操作在对称密码算法中应用非常广泛，但由于算法的不同，使得移位位宽和移位长度都不相同。经过总结现有的分组密码算法，它们的移位位宽包括以下几种：4-bit、8-bit、28-bit、32-bit 和 128-bit。移位操作的模式应该包括:逻辑左移、逻辑右移、循环左移和循环右移。而根据移位操

作的特点，其又可分为定长的移位和变长的移位，区别是移位长度是常量还是变量，前者如IDEA和 Twofish 算法，后者如 AES 和 RC6 算法等。并且目前常用的分组密码算法中，无论是哪种情况其移位长度都在 1-bit 到 31-bit 之间。下面给出了几种分组密码算法中使用移位操作的情况，如表 3.5 所示。

表 3.5　分组密码算法中的移位操作

算法	4-bit	8-bit	28-bit	32-bit	128-bit
DES			√		
IDEA					√
AES				√	
RC6				√	
Serpent				√ *	
Twofish	√				
Safer		√			
SEAL				√	
Mars				√	
CAST-256				√	
CRYPTON				√	
备注：	√：循环移位 *：逻辑移位				

目前移位操作很多都是通过 D 触发器[26]来实现的，但是这样的设计缺点是移的位数越多所需的时钟周期越多，就导致其工作效率较低。本节针对这一问题，对移位操作的特性进行了研究与分析，并提出了一种支持 28/32-bit 移位位宽的任意移位长度的移位模块的设计方案，它不仅性能可以满足密码处理的需求，而且任意位的移位都是在一个时钟周期内完成，最重要的是达到了通用的目的，为可重构密码处理结构的设计与实现提供了基础。

可重构移位模块的设计主要是满足不同分组密码算法的需求，通过一致的接口和不同的控制信号，共同使用一个模块，完成不同移位位宽、移位长度的移位操作。因此，通过分析分组密码算法，此移位模块的设计目标就是支持位宽为 4/8/28/32/128-bit 的移位长度在 1-bit 到 31-bit 之间的任意长度的移位操作，但是如果实现支持位宽为 4/8/28/32/128-bit 的移位必然增加很多硬件开销，由表 3.5 可见，大多数分组密码算法主要进行 28-bit 或 32-bit 位宽的循环移位，只有很少的几个进行 4-bit 或 8-bit 或 128-bit 位宽的移位，在满足其通用性的前提下，尽可能减少硬件开销，所以本文选择了支持 28/32-bit 两种位宽的可变移位长度的可重构循环移位模块进行设计。

2.　可重构循环移位模块功能说明

本节在研究了多种分组密码算法的基础上，设计了一种支持 28/32-bit 两种位宽可变移位长度的可重构循环移位模块，它主要实现 28-bit 位宽中 1～28 位任意移位长度的循环左/右移位和 32-bit 位宽中 1～32 位任意移位长度的循环左/右移位。其原理图如图 3.8 所示：

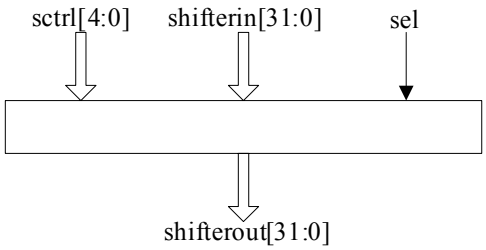


图 3.8　可重构循环移位模块原理图

shifterin[31:0]是可重构循环移位模块的 32 位输入数据端口，当要进行 28 位的循环移位时，shifterin[27:0]作为输入数据端口；sctrl[4:0]是移位位数控制端口，进行移位位数的选择；sel 是 28/32-bit 位宽选择端口，当 sel 端口为‘0’时，进行 28 位位宽中的循环移位，当 sel 端口为‘1’时，进行 32 位位宽中的循环移位；shifterout[31:0]是可重构循环移位模块的 32 位输出数据端口，当要进行 28 位的循环移位时，shifterout[27:0]作为输出数据端口。

3.2.2.2 电路设计

本节设计的可重构移位模块主要功能是实现 28/32-bit 两种位宽可变移位长度循环左移，而循环右移可以通过循环左移来实现，28 位和 32 位位宽的循环左移和循环右移关系分别如表 3.6 和表 3.7 所示。

表 3.6　28 位位宽的循环左移和右移关系

移位位数(sctrl[4:0])	循环左移位数	循环右移位数
00001	1	27
00010	2	26
00011	3	25
…	…	…
i	i	28-i
…	…	…
11011	27	1
11100	28	0

表 3.7　32 位位宽的循环左移和右移关系

移位位数(sctrl[4:0])	循环左移位数	循环右移位数
00001	1	31
00010	2	30
00011	3	29
…	…	…
i	i	32-i
…	…	…
11111	31	1
00000	32	0

其电路如图 3.9 所示：

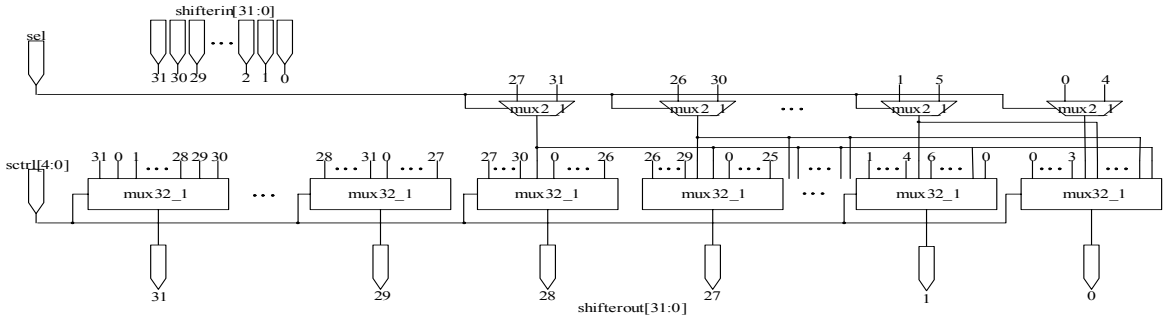


图 3.9　可重构循环移位模块电路图

3.2.2.3 仿真验证与实验结果分析

图 3.9 中可重构循环移位模块占用 FPGA 资源和最大组合路径延时情况如表 3.8 所示，通过对其配置位的设置可进行 28-bit 和 32-bit 位宽的循环移位，其仿真结果如图 3.10 所示，其中，可重构循环移位模块的输入数据 shifterin[31:0]和移位后的输出数据 shifterout32[31:0]和 shifterout28[27:0]都是用十六进制数显示的，sctrl[4:0]移位位数 sctrl[4:0]用无符号数显示的。当 sel 为‘0’，表示进行 28-bit 位宽的循环移位；当 sel 为‘1’， 表示进行 32-bit 位宽的循环移位。由图 3.10 仿真结果可见，当进行 28-bit 位宽的循环移位时，输入数据为 0000087,循环左移 5 位后输出为 00010E0，循环左移 27 位后输出为 8000043； 当进行 32-bit 位宽的循环移位时，输入数据为 0000087,循环左移 11 位后输出为 00043800，循环左移 27 位后输出为 38000004，循环左移 31 位后输出为 80000043。由此，验证了所设计的可重构循环移位模块功能的正确性。

表 3.8　可重构循环移位模块占用 FPGA 资源情况

Number of Slices	299 out of 32448
Number of 4 input LUTs	558 out of 64896
Number of bonded IOBs	98 out of 808
Equivalent gate count	4314
Maximum combinational path delay	22.998ns

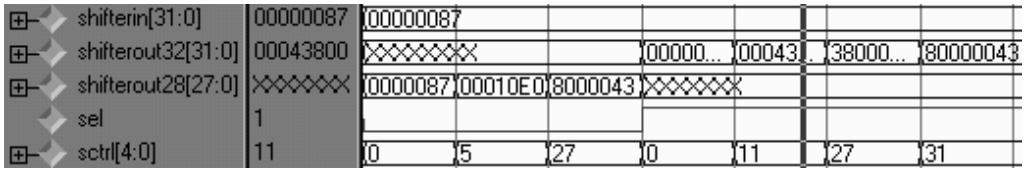


图 3.10　可重构循环移位模块的结果仿真图

3.2.3 可重构置换模块设计

3.2.3.1 设计方法

置换是密码算法中隐藏明文信息冗余度的重要手段，通过位置置换可以实现明文到密文的扩散。置换按明密文映射关系分为三类：直接置换、扩展置换和压缩置换。直接置换指明密文

间是一一映射关系，且明文的每一位都有到密文的映射；扩展置换指明密文间为一对多的映射关系，它使得密文对明文的依赖性传播得更快；压缩置换指明密文间是一一映射关系，但并不是明文的每一位都有到密文的映射。置换输入和输出位宽因算法和置换类型的不同而有所不同。分组密码中置换模块主要有：置换 8*8，置换 16*16，置换 32*32，扩展置换 32*48，压缩置换 56*48，置换 56*56，压缩置换 64*48，置换 64*64。

本节设计的置换模块能够实现上述 8 种置换，其原理图如图 3.11 所示。

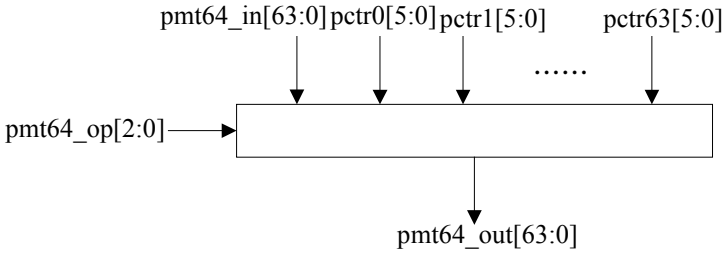


图 3.11　可重构置换模块原理图

pmt64_in 是输入，pctr0、pctr1、……pctr63 是静态编码，pmt64_op 是功能控制节点，通过功能控制节点和静态编码的变化，置换模块可以实现不同的置换功能。当 pmt64_op=“000”时，实现 8 组 8*8 置换功能；当 pmt64_op=“001”，实现 4 组 16*16 置换功能；当 pmt64_op=“010”，实现 2 组 32*32 置换功能；当 pmt64_op=“011”，实现 1 组 32*48 置换功能或 1 组 56*48 置换功能或 1 组 64*48 置换功能（具体看输入的位数，不足 64 位时，高位补 0）；当 pmt64_op=“100”，实现 1 组 56*56 置换功能；当 pmt64_op=“101”，实现 1 组 64*64 置换功能。

3.2.3.2 电路设计

本节设计的可重构置换模块主要功能是实现 8*8，16*16，32*32，32*48，56*48，64*48，56*56，64*64 置换，其硬件结构如图 3.12 所示。

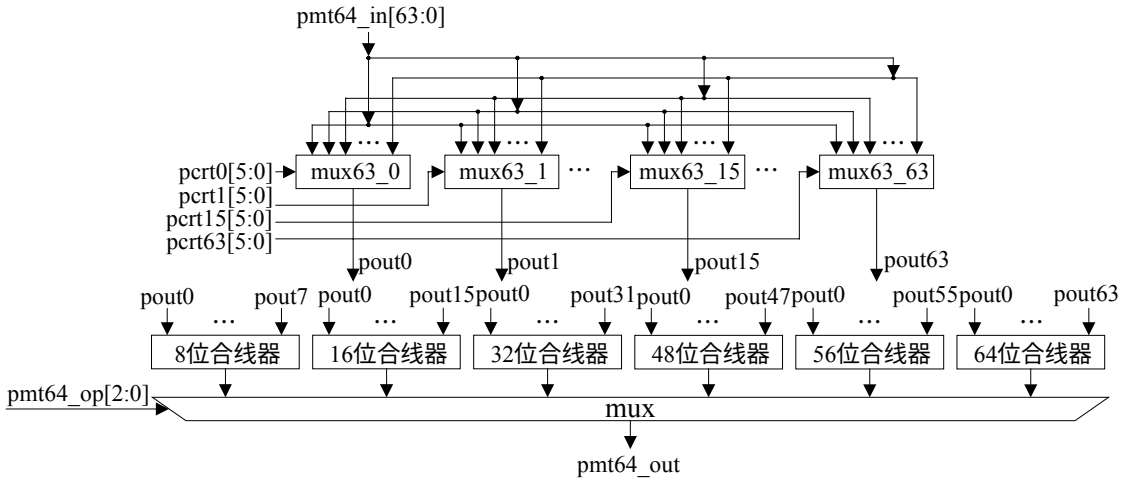


图 3.12　可重构置换模块硬件结构图

3.2.3.3 仿真验证与实验结果分析

图 3.12 中可重构置换模块占用 FPGA 资源和最大组合路径延时情况如表 3.9 所示，以 DES 的初始置换和逆初始置换为例进行仿真验证，其仿真结果如图 3.13 所示，由仿真图可见，DES 的初始置换输出结果作为逆初始置换输入而得到的输出结果完全是初始置换的输入数据，由此验证了所设计的可重构置换模块功能完全正确。

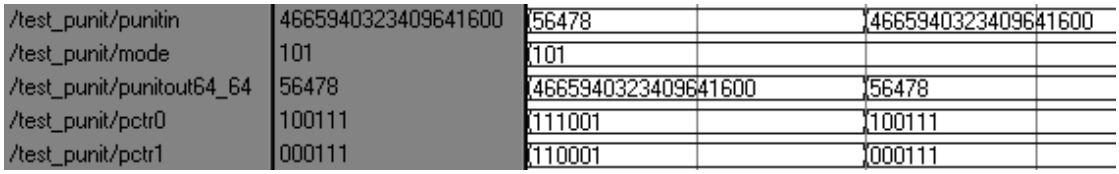


图 3.13 可重构置换模块的结果仿真图

表 3.9 可重构置换模块占用 FPGA 资源情况

Number of Slices	1281 out of 32448
Number of 4 input LUTs	2528 out of 64896
Number of bonded IOBs	771 out of 808
Equivalent gate count	20388
Maximum combinational path delay	16.03ns

3.2.4 模加/减运算模块设计

3.2.4.1 设计方法

在分组密码运算中，模加/减运算使用的频度较高，模加/减运算通常可表示为 C= (A+B) mod N。通过研究发现，分组密码运算模加/减操作中两个加数的操作位宽若均为 n，则模数 N 等于 2^n，且参数 n 大多数为 16、32。例如，IDEA 使用了模 2^{16} 加法，Twofish、Mars、RC6、CAST 等算法使用了模 2^{32} 加法。所以，要求模加减模块同时支持模 2^{16}、2^{32} 的模加与模减运算功能。分组密码运算中模 2^n 加法运算是忽略最高位进位而只保留 n-bit 和的整数加法。

以上分析可以看出，分组密码运算中的模加/减运算模块计算的最小的操作粒度为 16 比特。因此，在电路设计中考虑使用通过对进位级联链的控制来实现模 2^{16}、2^{32} 的加/减运算。电路在实现上需要 2 个 16 位的加法器，从而可进行 1 个 32-bit 模 2^{32} 加减运算，或者进行 2 个 16-bit 模 2^{16} 加减运算。

本节设计了一个模 $2^{16}/2^{32}$ 加/减运算模块，它主要实现模 2^{16} 和 2^{32} 加/减运算功能。其结构如图 3.14 所示，a 和 b 为 32 位输入数据，sel 为模式选择信号，负责决定是进行模 2^{16} 还是模 2^{32} 运算，cin 为加减选择信号，负责决定是进行模加运算还是模减运算。当操作数 a<b 时输出是负值，在操作数位宽范围内将结果取反加‘1’，即可得到此操作数范围内的最终模减运算结果，图 3.14 中设计的比较器就是用来处理这种情况的。16/32 位加/减运算器是整个模 $2^{16}/2^{32}$ 加/减运

算模块的核心部分，其电路设计如下节所示。

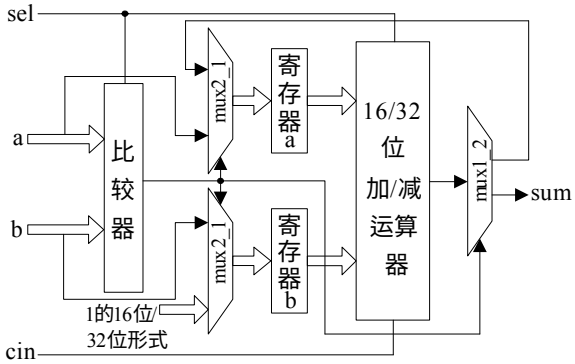


图 3.14　模 $2^{16}/2^{32}$ 加/减运算模块结构图

3.2.4.2 电路设计

16/32 位加/减运算器是整个模 $2^{16}/2^{32}$ 加/减运算模块的核心部分，其电路如图 3.15 所示。

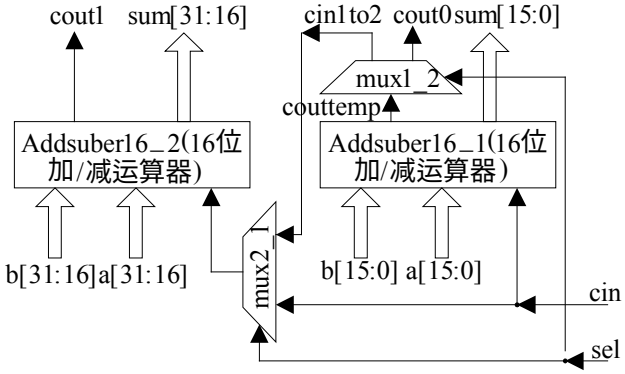


图 3.15　16/32 位加/减运算器电路图

图中，sel 为模式选择信号，当 sel =‘0’时，表示进行 16 位加/减运算，而 sel =‘1’时，表示进行 32 位加/减运算；cin 为进位输入信号，Addsuber16_1 的 cin 信号同时作为加/减运算控制信号，当 Addsuber16_1 的 cin =‘0’时，表示进行加运算，而 cin =‘1’时，表示进行减运算；图中的两个 16 位加/减运算器是其核心单元。其电路如图 3.16 所示。图 3.16 中所设计的 16 位加/减运算器采用的是组间超前进位的方式实现的。16 位加/减运算器的核心部件是四个相同结构的 4 位全加器，4 位全加器采用的是组内超前进位的方式实现的，其电路如图 3.17 所示，g_out 是由于本小组内某一位的 a_ib_i 同时为‘1’，所产生的向高组的进位，所以称它位“ 本组进位 ”，其逻辑表达式如式（3-3）所示；p_out 是由于本组内各位 a_i 和 b_i 的半加和均为‘1’，将低组向本组的进位信号传送到高组去的进位，所以称它为本组的“ 传送进位 ”，其逻辑表达式如式（3-4）所示。

$$g\_out = g4 + p4g3 + p4p3g2 + p4p3p2p1g1 \tag{3-3}$$

$$p\_out = p4 \cdot p3 \cdot p2 \cdot p1 \tag{3-4}$$

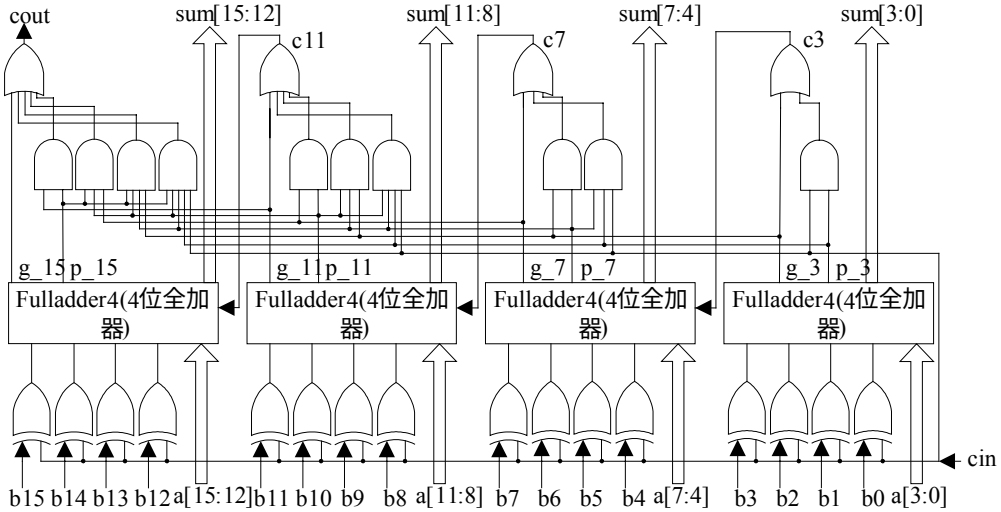


图 3.16　16 位加/减运算器电路图

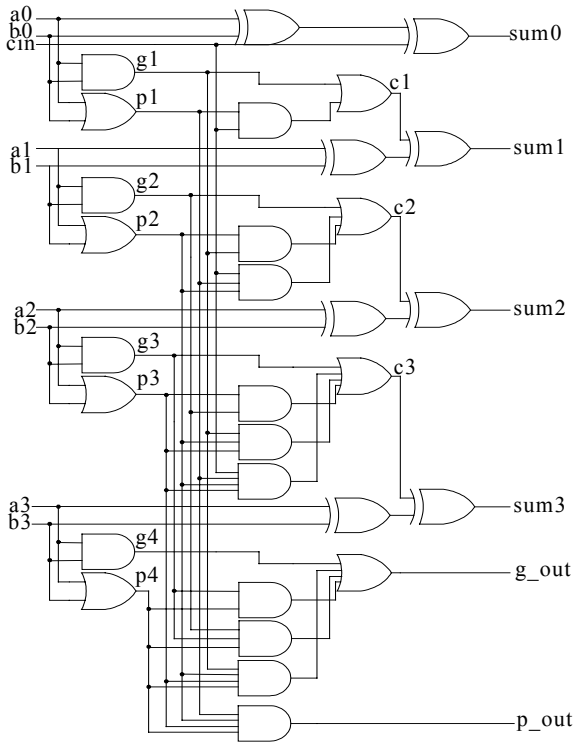


图 3.17　4 位全加器电路图

3.2.4.3 仿真验证与实验结果分析

图 3.14 中模 $2^{16}/2^{32}$ 加/减运算模块占用 FPGA 资源和最大组合路径延时情况如表 3.10 所示，通过对其配置位的设置可实现模 2^{16} 加/减运算和模 2^{32} 加/减运算，其仿真结果如图 3.18 所示，由仿真图可见，所设计的模 $2^{16}/2^{32}$ 加/减运算模块功能完全正确。

表 3.10　模 $2^{16}/2^{32}$ 加/减运算模块占用 FPGA 资源情况

Number of Slices	92 out of 32448
Number of 4 input LUTs	161 out of 64896
Number of bonded IOBs	100 out of 808
Equivalent gate count	966
Maximum combinational path delay	49.705ns

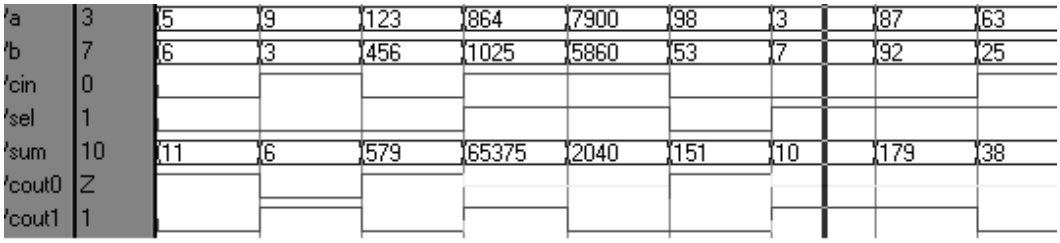


图 3.18　模 $2^{16}/2^{32}$ 加/减运算模块的结果仿真图

3.2.5 模乘运算模块设计

3.2.5.1 设计方法

通过对分组密码算法的分析可以看出，模乘运算[27]在 RC6，IDEA，MARS 等常用分组密码算法中均有使用。模乘运算通常可表示为 $c = ab \bmod N$。通过研究发现，分组密码运算模乘操作中两个乘数的操作位宽若均为 n，则模数 N 等于 2^n 或 2^n 士 1，且参数 n 大多为 16 或 32。表 3.11 列出了目前公开的分组密码算法所使用的模乘操作类型。

表 3.11　分组密码的模乘操作类型

密码算法	IDEA	RC6	E2	TWOFISH	MARS	MMB	DFC
模乘类型	$2^{16}+1$	2^{32}	2^{32}	2^{32}	2^{32}	$2^{32}-1$	2^{64}

本节设计了一种高速的可重构模乘运算模块，可以实现模 $2^{16}/2^{32}/2^{16}+1$ 乘法运算，其结构如图 3.19 所示。在模 2^{16} 乘实现过程中只要取结果的低 16 位即可，在模 $2^{16}+1$ 实现时需要加一个修正单元，修正单元实现原理如下所示：

假设 a 和 b 是两个 n 位的数据，则

$$c = ab \bmod (2^n+1) = (c_H \cdot 2^n + c_L) \bmod (2^n+1) \qquad (3\text{-}5)$$

这里 c_H 为 ab 乘积的高 n 位，c_L 为 ab 乘积的低 n 位。

由式（3-5）可得到下式：

$$c = (c_H \cdot 2^n + c_L + c_H - c_H) \bmod (2^n+1) = (c_L - c_H) \bmod (2^n+1) \qquad (3\text{-}6)$$

由式（3-6）可知：

如果 $c_L - c_H \geq 0$，则 $c = c_L - c_H$ ；

如果 $c_L - c_H \leq 0$，则 $c = c_L - c_H + 2^n + 1$。

根据上述算法理论，修正单元可以在上节模加/减运算模块的基础上稍加修改来实现，其结构如图 3.20 所示。

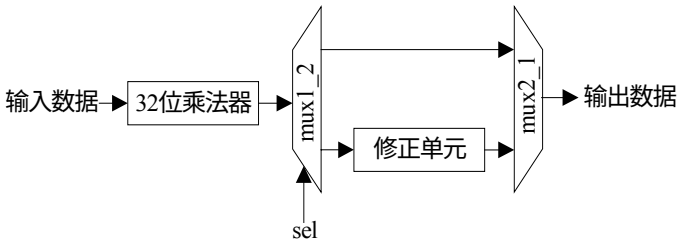


图 3.19　可重构模乘运算模块结构图

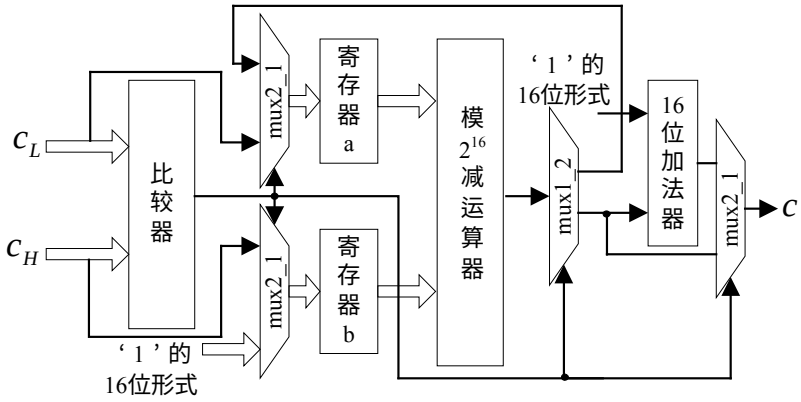


图 3.20　修正单元结构图

3.2.5.2 电路设计

在整体电路设计中，32 位乘法器是核心运算部件，也是延时最大的运算模块。本节设计了一种高速的 32 位并行无符号整数乘法器，图 3.21 给出了其整体设计结构框图。该乘法器能够高速运算是由于以下两点原因：

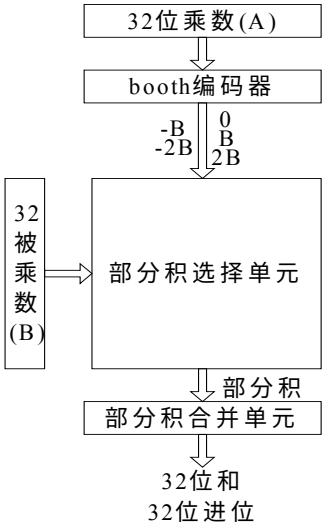


图 3.21　32 位高速整数乘法器的结构框图

（1）减少部分积的数目。这种技术实际上是对乘数进行再编码来加快部分积的形成，最典型

的就是 Booth 算法[28]。这些算法实际上是对乘数进行重新编码，即通过冗余的带符号数产生新的乘数表示形式，再按照一般的乘法步骤进行操作。在乘法器设计中大都采用改进 Booth 算法以减少部分积，简化电路和提高运算速度。改进的 Booth 算法的原理如下：

设乘数为：

$$A = \sum_{k=0}^{\frac{N}{2}-1}(-2a_{2k+1} + a_{2k} + a_{2k-1}) \times 2^{2k} \tag{3-7}$$

式中的 N 是指乘数 A 的位宽，令 $Q(k)= -2a_{2k+1} + a_{2k} + a_{2k-1}$（显然 $Q(k) \in \{-2,-1,0,1,2\}$）则式（3-7）可异简化为

$$A = \sum_{k=0}^{\frac{N}{2}-1} Q(k) \times 2^{2k} \tag{3-8}$$

乘积 $P = A \times B = \sum_{k=0}^{\frac{N}{2}-1} Q(k) \times B \times 2^{2k}$。

改进的 Booth 算法部分积选择表如表 3.12 所示。

表 3.12　改进的 Booth 算法部分积选择表

a_{j+1}	0	0	0	0	1	1	1	1
a_j	0	0	1	1	0	0	1	1
a_{j-1}	0	1	0	1	0	1	0	1
操作	0	+B	+B	+2B	-2B	-B	-B	0

由表 3.12 可以得到以下表达式：

$$\mathrm{ADD} = \overline{a_{j+1}} \,\&\, (a_j \oplus a_{j-1}) \tag{3-9}$$

$$\mathrm{SUB} = a_{j+1} \,\&\, (a_j \oplus a_{j-1}) \tag{3-10}$$

$$\mathrm{ADD2} = \overline{a_{j+1}} \,\&\, a_j \oplus a_{j-1} \tag{3-11}$$

$$\mathrm{SUB2} = a_{j+1} \,\&\, \overline{a_j} \oplus \overline{a_{j-1}} \tag{3-12}$$

$$\mathrm{NEG}_j = a_{j+1} \tag{3-13}$$

其中，ADD 和 ADD2 分别表示对被乘数 B 的操作是乘 1 和乘 2；SUB 和 SUB2 分别表示对被乘数 B 的操作是乘-1 和乘-2，NEG_j 表示对被乘数 B 操作的符号。

可以对上面公式的逻辑表达式进行优化，将逻辑进行共享，可以看出 ADD 和 SUB 都包含 $a_j \oplus a_{j-1}$，所以将它们的逻辑功能进行合并，设合并后的信号为 $1X_j$，那么有

$$1X_j = a_j \oplus a_{j-1} \tag{3-14}$$

同样将 ADD2 和 SUB2 的逻辑功能进行合并，设合并后的信号为 $2X_j$，那么有：

$$2X_j = (\overline{a_{j+1}} \,\&\, a_j \,\&\, a_{j-1}) + (a_{j+1} \,\&\, \overline{a_j} \,\&\, \overline{a_{j-1}}) \tag{3-15}$$

由逻辑表达式（3-13）、（3-14）、（3-15）可得到优化后的 Booth 编码器，其电路结构如图

3.22 所示，优化设计后的 Booth 编码器使处于关键路径的 $2X_j$ 信号逻辑级数减少。

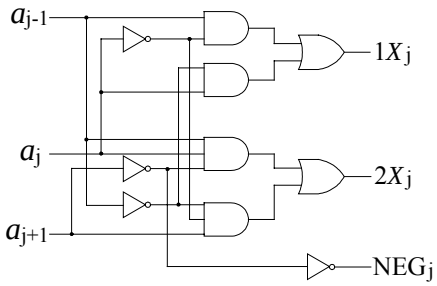


图 3.22　优化后的 Booth 编码器电路结构

这里将部分积选择单元称为部分积选择器。$1X_j$ 为第 j 个 Booth 编码器产生的 1 倍被乘数选择信号，$2X_j$ 为第 j 个 Booth 编码器产生的 2 倍被乘数选择信号，NEG_j 为第 j 个 Booth 编码器产生的第 j 个部分积的符号位信号，P_{ij} 表示第 j 个部分积的第 i 位，其中 i 表示被乘数 B 的位下标，j 表示乘数 A 的位下标，b_i 表示被乘数 B 的第 i 位，那么有部分积选择器的逻辑表达式：

$$P_{ij} = \overline{\overline{b_i \,\&\, 1X_j} \,\&\, \overline{b_{i-1} \,\&\, 2X_j}} \oplus NEG_j \tag{3-16}$$

对上式化简得：

$$P_{ij} = \overline{(b_i \,\&\, 1X_j) + (b_{i-1} \,\&\, 2X_j)} \odot NEG_j \tag{3-17}$$

根据式（3-17）可得到优化的部分积选择器，其电路结构如图 3.23 所示。

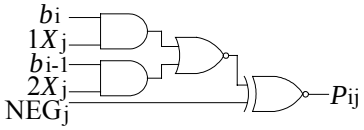


图 3.23　优化后的部分积选择器

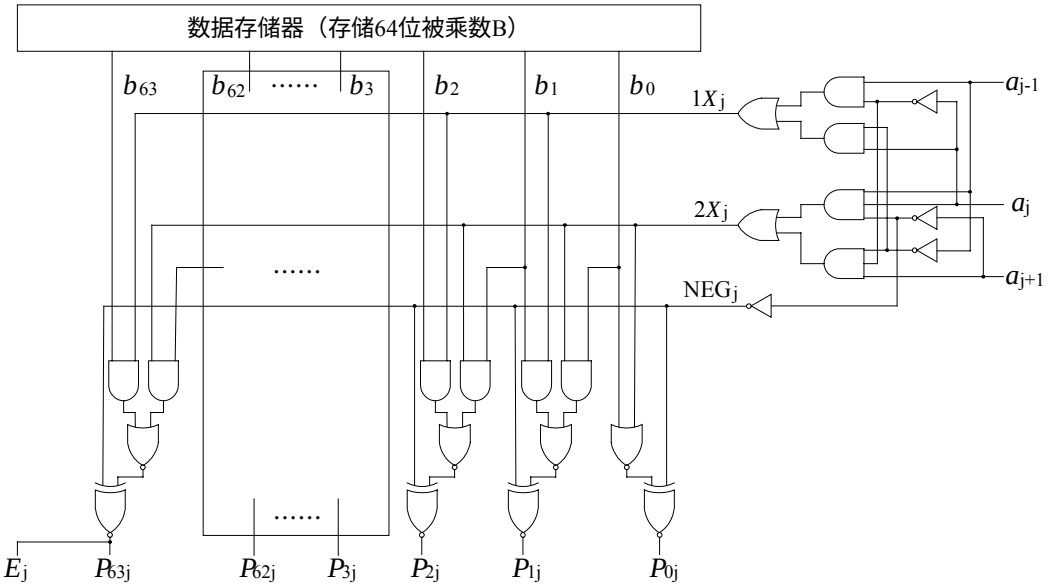


图 3.24 一组部分积逻辑实现电路图

由图 3.22 的 Booth 编码器和图 3.23 的部分积选择器实现的第 j 组部分积如图 3.24 所示，部分积最后是通过加法器进行其补码相加，图中的 E_j 是符号扩展位，E_j 与部分积的原符号位（即部分积的最高位 P_{63j}）相同。

（2）加快部分积加法运算的速度。为了加快部分积相加的速度，一般采用阵列乘法器[29]或 Wa11ace 树型[30]乘法器来加快结果的产生。

部分积加法运算通常有阵列乘法器和 wallace 树型乘法器两种实现方式。这两种方式是进位保留加法器 CSA（Carry Save Adder）排列的两种极端形式，前者是串行的，后者是全并行的。为了加快乘法器的执行速度，需要通过提高计算的并行度来实现。Wallace 树状结构是最有效的提高并行度的方法。利用 CSA 加法器通过并行地对部分积进行按列压缩达到快速得出最终结果的目的。Wallace 树基本上是由进位保留加法器阵列构成的，其结构如图 3.25 所示，Wallace 树的核心部分是一个 4-2 的压缩器，其电路如图 3.26 所示。

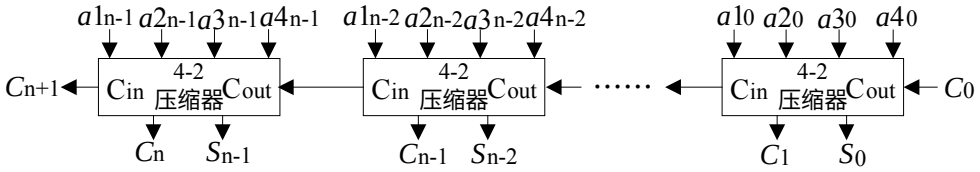


图 3.25　进位保留加法器阵列结构图

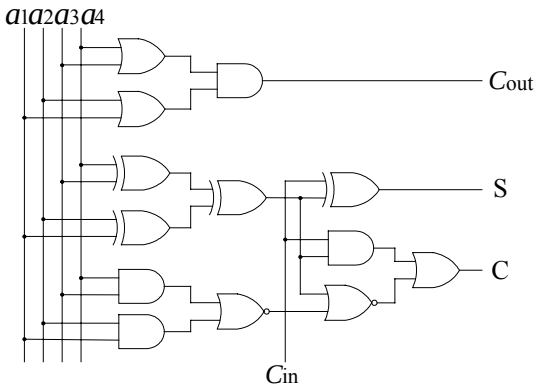


图 3.26　4-2 压缩器电路图

3.2.5.3 仿真验证与实验结果分析

本节所设计的模乘运算模块占用 FPGA 资源和最大组合路径延时情况如表 3.13 所示，其仿真结果如图 3.27 所示，a 和 b 为输入，p 为输出，sel 为模式选择信号，当 sel 为‘0’，进行模 $2^{16}/2^{32}$ 乘法运算，而当 sel 为‘1’时，进行模 $2^{16}+1$ 乘法运算。由仿真图可见，所设计的模 $2^{16}/2^{32}/2^{16}+1$ 乘运算模块功能完全正确。

表 3.13　模乘运算模块占用 FPGA 资源情况

Number of Slices	1568 out of 32448
Number of 4 input LUTs	2854 out of 64896
Number of bonded IOBs	128 out of 808
Equivalent gate count	18537
Maximum combinational path delay	151.536ns

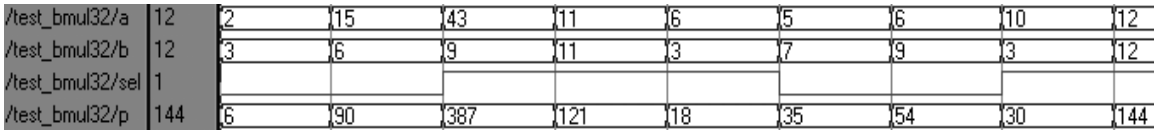


图 3.27　模乘运算模块的结果仿真图

3.3 本章小结

本章给出了可重构密码处理系统的设计方案，所设计的可重构密码处理系统包括了存储模块、控制模块、可重构密码处理模块 RCPM。其中 RCPM 是整个密码处理系统的核心部分，本章设计了一种易于流水线操作的可重构密码处理结构来实现可重构密码处理模块。在可重构密码处理模块中，可重构密码运算模块 RCM 是其最关键部分，根据可重构密码运算模块设计原则，对 RCM 硬件实现进行了研究。重点设计了 S 盒替代模块、可重构移位模块、可重构置换模块、模加/减运算模块以及模乘运算模块等可重构密码运算模块。这些基本运算模块可根据 RCM 配置指令进行重构，灵活完成不同算法所需的运算功能。本章所设计的可重构 S 盒替代模块、可重构移位模块、可重构置换模块在后面章节设计的 DES/AES 可重构密码处理模块时会用到，而模加/减运算模块以及模乘运算模块是为实现 LM 网络结构密码算法（如 IDEA）所设计的，在后面的 DES/AES 可重构密码处理结构实现中并未使用到。

第四章 DES/AES 可重构密码处理结构实现与算法验证

4.1 可重构密码处理结构实现及其性能分析

本章主要针对 DES 和 AES 密码算法，介绍了 DES/AES 可重构密码处理结构的具体实现过程。如果要实现更多类型的密码算法，只要增加所需的运算模块即可。

4.1.1 DES 算法介绍

1. DES 算法简介

图 4.1 是 DES 加密算法的框图，其中明文分组为 64 比特，密钥长为 56 比特。图的左边是明文的处理过程，有 3 个阶段，首先是一个初始置换 IP，用于重排明文分组的 64 比特数据。然后是具有相同功能的 16 轮变换，每轮中都有置换和代换运算，第 16 轮变换的输出分为左右两半，并被交换次序。最后再经过一个逆初始置换 IP^{-1}（为 IP 的逆）从而产生 64 比特的密文。

图 4.1 的右边是使用 56 比特密钥的方法。密钥首先通过一个置换函数，然后，对加密过程的每一轮，通过一个左循环移位和一个置换产生一个子密钥。其中每轮的置换都相同，但由于密钥被重复迭代，所以产生的每轮子密钥不相同。DES 的加密与解密的密钥和流程完全相同，区别仅仅是加密与解密使用的子密钥序列的施加顺序正好相反。

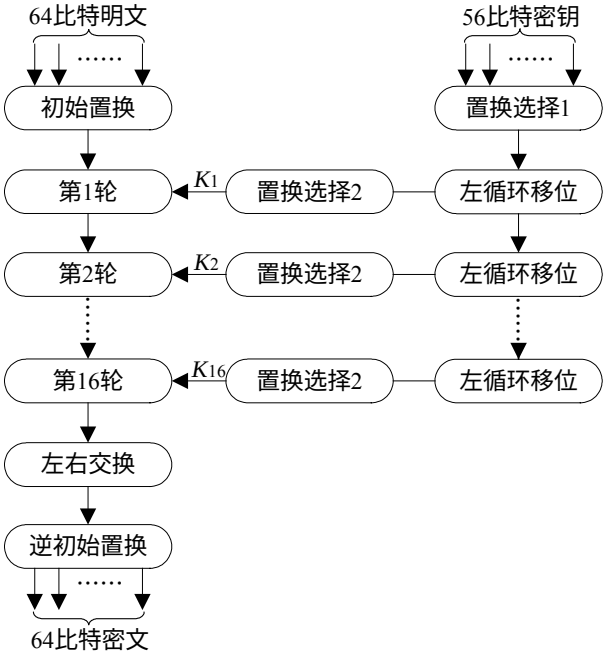


图 4.1 DES 加密算法框图

2. 轮结构

DES 加密算法的轮结构如图 4.2 所示。每轮变换可由以下公式表示：

$$L_i = R_{i-1} \tag{4-1}$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i) \tag{4-2}$$

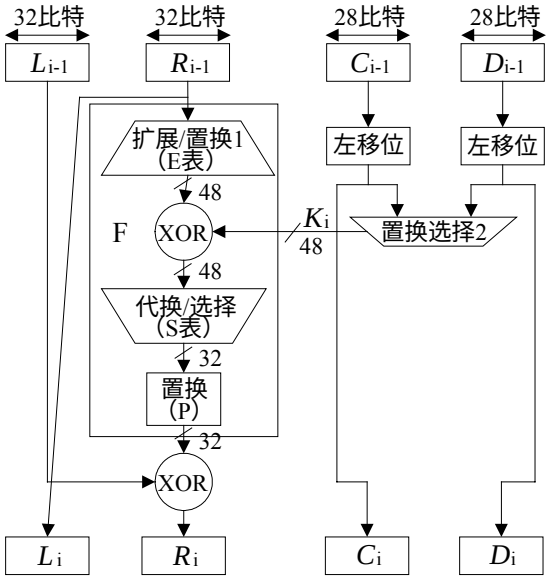


图 4.2　DES 加密算法的轮结构

DES 加密和解密过程使用相同的函数来加密和解密每个分组，加密和解密过程的唯一不同之处是密钥的次序相反。这就是说，如果各轮的加密密钥分别是 $K_1, K_2, K_3, \cdots, K_{16}$,那么解密密钥就是 $K_{16}, K_{15}, K_{14}, \cdots, K_1$。

3.　密钥的产生

由图 4.1 和 4.2 可见，输入算法的 56 比特密钥首先经过一个置换运算，然后将置换后的 56 比特分成各为 28 比特的左、右两半，分别记为 C_0 和 D_0。在第 i 轮分别对 C_{i-1} 和 D_{i-1} 进行左循环移位，移位后的结果作为求下一轮子密钥的输入，同时也作为置换选择 2 的输入。通过置换选择 2 产生的 48 比特的 K_i，即为本轮的子密钥，作为函数 $F(R_{i-1}, K_i)$的输入。

4.1.2 AES 算法介绍

1.　AES 算法简介

AES 是一个迭代型的分组密码，集安全性、高效性、灵活性于一身，但是在加密之前，需要对明文数据块做预处理。首先，把数据块写成字的形式，每个字包含 4 个字节，每个字节包含 8 比特信息；其次，把字记为列的形式。其加/解密过程如图 4.3 所示，轮函数主要由轮密钥加变换、字节代换/逆字节代换、行移位变换/逆行移位变换、列混合变换/逆列混合变换四部分组成。

轮密钥加变换 AddRoundKey 是对输入的 128 位数据和子密钥执行模 2 加法，即异或运算。

字节代换 ByteSub 是一个关于字节的非线性变换，它将状态中的每个字节非线性地变换为

另一个字节。该变换是可逆的，解密时，利用字节变换的逆变换。

行位移变换 ShiftRows 是将状态中的各行以不同的位移量进行循环移位，0 行不移，第 1 行移 1 个字节，第 2 行移 2 个字节，第 3 行移 3 个字节。

列混合变换 MixColumns 是对一个状态逐列进行变换，它将一个状态的每一列视为有限域 GF(2^8)上的一个多项式。

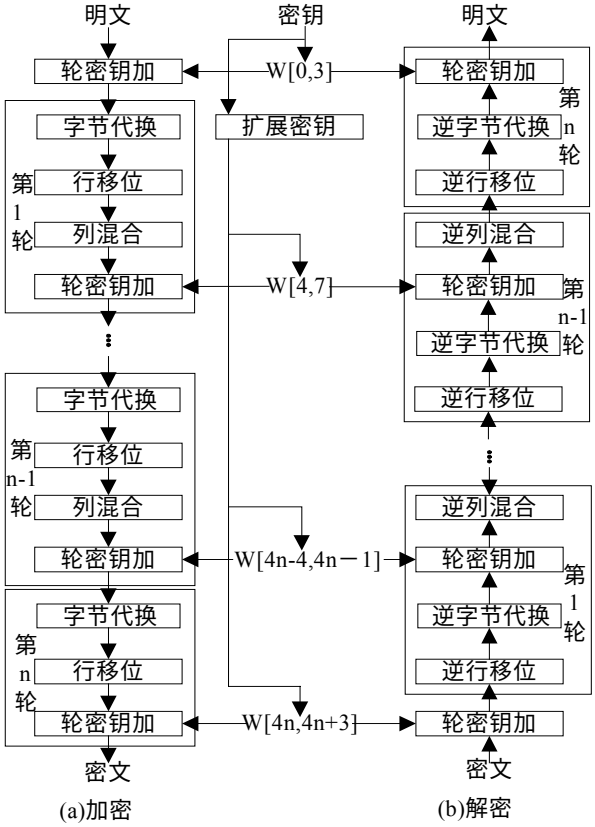


图 4.3　AES 的一般加解密过程

2.　密钥的产生

密钥产生过程就是从种子密钥(Seed Key)得到轮密钥的过程，其结构如图 4.4 所示。种子

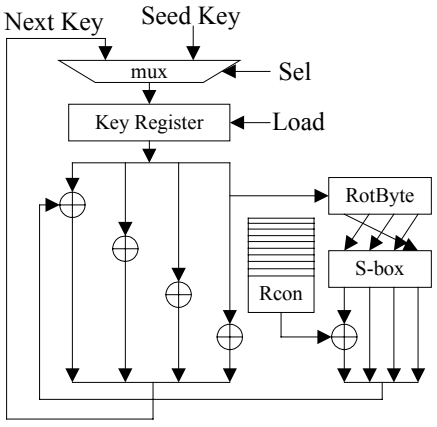


图 4.4　AES 密钥产生结构图

密钥（128 位）的[31:0]位经过以下一系列的变换作为下一轮轮密钥(Next Key)的[127:96]位，种子密钥[32:127]作为下一轮的轮密钥的[95:0]位。如此循环下去，可得到每轮的轮密钥。

1 字节循环移位 RotByte→S 盒变换 SubByte→异或轮常数 Rcon[i]→异或种子密钥的高 32 位

4.1.3 可重构密码处理结构实现及性能分析

1. 可重构密码处理结构实现

本节是以实现 DES/AES 可重构处理结构为例，来验证整个可重构密码处理结构模型 RCPA 的正确性。要实现 DES 算法和 AES 算法的可重构密码处理结构，首先要确定这个可重构密码处理结构的各可重构密码运算模块，表 4.1 列出了 DES 轮结构资源使用情况，表 4.2 列出了 AES 轮函数资源使用情况，表 4.3 列出了所设计的可重构密码处理结构资源使用情况。

表 4.1 DES 轮结构资源表

运算模块名称	规格	功能说明	数量
PMT_32*48	32*48 扩展置换	实现 32 位输入到 48 位输出的任意扩展置换	1 个
XOR_48	48 位逐位异或	实现 48 位数据的逐位异或操作	1 个
SBOX	6*4 S 盒	实现 6 输入到 4 输出的非线性代替变换	8 个
PMT_32	32*32 置换	实现 32 位输入到 32 位输出的任意置换	1 个
XOR_32	32 位逐位异或	实现 32 位数据的逐位异或操作	1 个

表 4.2 AES 轮函数资源表

运算模块名称	规格	功能说明	数量
SHFT_32	32 位循环移位	实现 32 位数据的循环移位操作	4 个
XOR_32	32 位逐位异或	实现 32 位数据的逐位异或操作	4 个
SBOX	8*8 S 盒	实现 8 输入到 8 输出的非线性代替变换	16 个
MPMUL_8	模 8 次多项式乘法	实现 AES 的列混合变换	4 个

然后按照图 3.3 进行连接，其电路结构如图 4.5 所示，图中的置换、S 盒替换、移位等运算是第三章所设计的可重构密码运算模块，通过设定不同的配置位，实现不同算法所需的运算功能。图中有的运算模块是 32 位运算，有的是 48 位运算，为了统一操作数的位宽，图中的数据总线宽度设计为 48 位，同时设计了专门的位宽处理模块，其主要功能是：将 32 位数据通过高位补 0 方法扩展到 48 位数据，也可以将 48 位数据舍去高位而得到 32 位数据。

由 DES 和 AES 算法的资源表可以得到要实现的 DES/AES 可重构密码处理结构的资源，如

表 4.3 所示。

表 4.3　DES/AES 可重构密码处理结构资源表

运算模块名称	功能说明
可重构移位模块	实现 28/32-bit 两种位宽可变移位长度循环移位
可重构置换模块	实现 8*8，16*16，32*32，32*48，56*48，64*48，56*56，64*64 置换
可重构 S 盒	实现 4*4 或 6*4 S 盒
异或模块	实现 32/48 位的逐位异或
MPMUL _8	实现 AES 的列混合变换（具体设计参见 5.2 节）

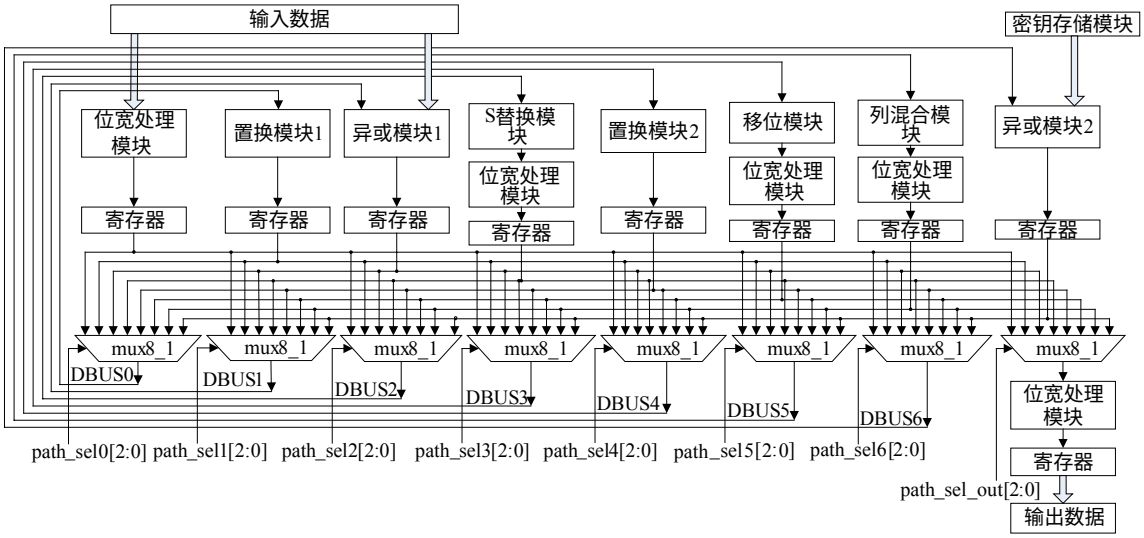


图 4.5　DES/AES 可重构密码处理电路结构图

2.　可重构密码处理结构性能分析

可重构密码处理结构性能分析主要包括三部分：资源消耗、配置位位宽、主频大小和和其他实现方案的性能对比。

表 4.4 给出了所设计的可重构密码处理结构占用 FPGA 资源情况。

表 4.5 给出了所设计的可重构密码处理结构各组成部分的配置位位宽，可计算总配置位为 1025bits，而 FPGA 实现 DES 或 AES 单一算法时的配置位流文件至少要达到 1000KB。

系统的主频估计主要依据系统中的可重构运算模块延时来确定。在本课题中，各个可重构运算模块电路延时直接影响了整个可重构密码处理系统的主频。表 4.6 给出了各可重构运算模块的电路延时。

由表 4.6 可见，所设计各可重构运算模块中，可重构移位模块的电路延时最大，因此系统的时钟周期的长度确定为 25ns，即最高主频为 40MHz，这样，所有的可重构运算模块都能在一个时钟周期内完成运算，从而使系统能获得较高的性能。

表 4.4　可重构密码处理结构占用 FPGA 资源情况

Number of Slices	2771 out of 32448
Number of Slice Flip Flops	384 out of 64896
Number of 4 input LUTs	5352 out of 64896
Number of bonded IOBs	762 out of 808
Equivalent gate count	45150

表 4.5　可重构密码处理结构中各组成部分的配置位位宽

运算模块名称	可重构移位模块	可重构置换模块	可重构 S 盒	路径选择（mux）
配置位位宽（bits）	6	387	134×4	24×4

表 4.6　各可重构运算模块的电路延时

运算模块名称	可重构移位模块	可重构置换模块	可重构 S 盒	异或模块	MPMUL _8
电路延时（ns）	22.998	16.03	21.333	7.707	19.714

AES 的分组长度是 128 位，图 4.5 中设计的可重构密码处理结构是 32 位位宽，所以需要 4 个这样的结构单元才能并行处理 AES 的 128 位明文分组，同时能够处理 4 个分组长度为 64 位的 DES 分组。与其他实现方案的性能对比如表 4.7 所示，由表 4.7 可见，本设计具有更快的密码处理速度，而且硬件开销和其他实现方案相当。此外，比文献[33]中的设计有更好的扩展性，文献[33]中的设计只能实现 DES 和 AES 两个算法，而只要增加更多的运算模块，本设计就可以实现更多的密码算法。

表 4.7　与其他实现方案的性能对比

实现方案	面积开销（门数）	频率	实现算法及吞吐率	
			算法	吞吐率
软件实现[14]	————	Pentium4 2.1（GHz）	DES	22.19（Mb/s）
			AES	62.05（Mb/s）
Gurk[31]	11.9 万	166（MHz）	AES	2.12 （Gb/s）
Ingrid[32]	17.3 万	154（MHz）	AES	2.29 （Gb/s）
Gao[33]	19 万	110（MHz）	DES	7 （Gb/s）
			AES	1.4 （Gb/s）
本设计	18.06 万	40（MHz）	DES	10.24（Gb/s）
			AES	5.12 （Gb/s）

4.2 DES 算法的映射与仿真验证

4.2.1 DES 算法映射

按照 DES 算法的需要合理的设定各可重构运算模块的配置位，得到 DES 算法中各运算模块，再设定好各多路选择器的选择位，形成 DES 算法的整个加/解运算路径，最终达到 DES 密码算法加/解密运算的目的。

DES 加/解密轮结构在 DES/AES 可重构密码处理结构上映射过程如图 4.6 所示，其映射过程是：DES 算法是 64 位的明/密文输入，64 位明/密文经过初始置换/逆初始置换后存入图中左边的存储模块中，记为 $L_{j,1}R_{j,1}$（$L_{j,1}$ 表示第 j 个分组的第 1 轮的左半部分数据（可为 64 位输入的低 32 位，也可为高 32 位，这里设为高 32 位）；$R_{j,1}$ 表示第 j 个分组的第 1 轮的右半部分数据），然后存储模块将右半部分数据 $R_{j,i-1}$ 输入 DES 的各个运算模块，由于中间有寄存器，每个运算模块的运算结果经过一个时钟周期后输出给下一个运算模块处理，整个轮结构的输出结果 $R_{j,i}$ 作为下一轮的左半部分输入数据 $L_{j,i}$ 存储在左边存储器的存储 $L_{j,i-1}$ 单元上，覆盖掉原来的 $L_{j,i-1}$ 数据。每轮运算需要 6 个时钟周期。

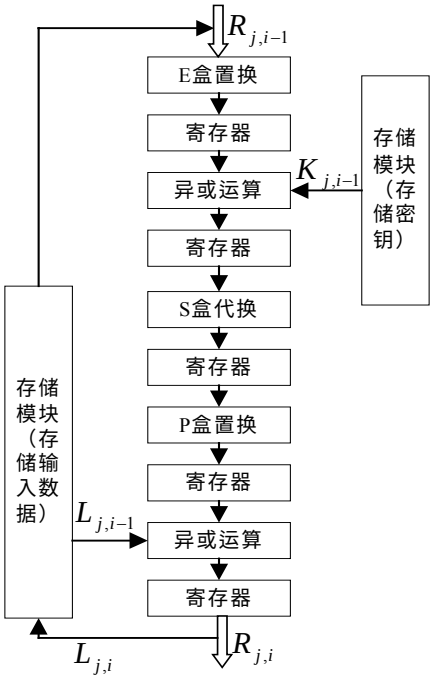


图 4.6　DES 加/解密轮结构在可重构密码处理结构上的映射图

4.2.2 仿真验证

DES 算法的加密和解密的轮结构完全一样，只是加密时输入的是 64 密文和经过密钥生成器后的 48 位轮密钥，这里我们用随机选取的 48 位密钥来代替经过密钥产生器后的 48 位轮密钥。仿真波形如图 4.7 和 4.8 所示，其中图 4.7 是加密时的仿真波形，而图 4.8 是解密时的仿真波形，

图中的信号端口可参见图 4.5 的可重构密码处理模块电路结构图，在整个加/解密过程中，用到的多路器路径选择信号为：path_sel0=“000”, path_sel1=“100”, path_sel2=“111”, path_sel3=“011”, path_sel6=“001”, path_sel_out=“010”。图 4.7 和 4.8 中测试数据的测试结果如表 4.8 和 4.9 所示，由表 4.8 和 4.9 可见，将加密时 DES 轮结构的输出结果作为解密的输入数据（解密时应将加密时输出结果的右半部分作为解密输入数据的左半部分输入，而将加密时输出结果的左半部分作为解密输入数据的右半部分输入）完全能够得到原来用于加密时的输入数据，所以，验证了所设计的可重构密码处理模块完全能够正确实现 DES 算法。

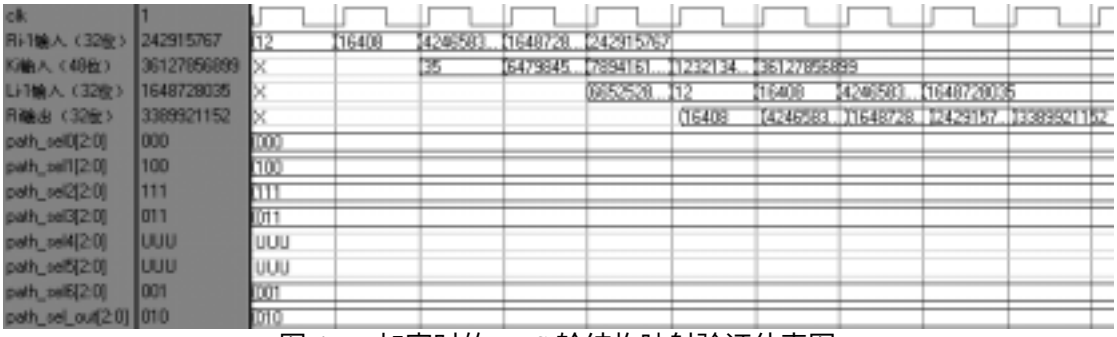


图 4.7　加密时的 DES 轮结构映射验证仿真图

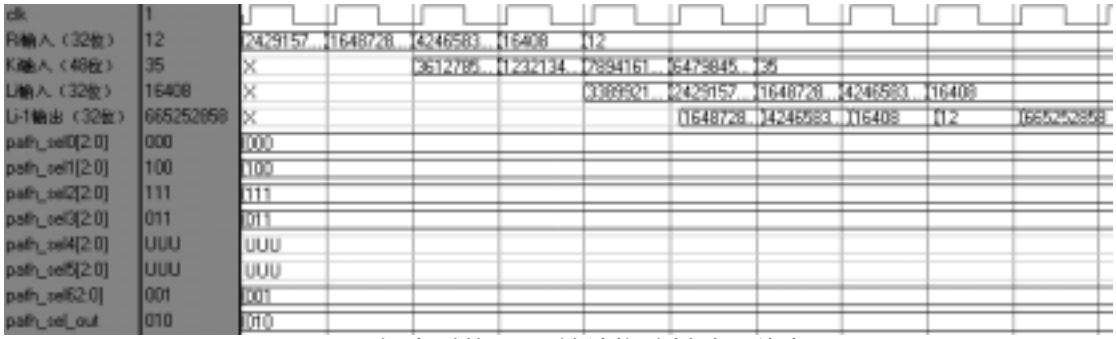


图 4.8　解密时的 DES 轮结构映射验证仿真图

表 4.8　DES 加密过程测试结果

	输入数据			输出数据	
	L_{i-1}（32 位）	R_{i-1}（32 位）	K_i（48 位）	L_i（32 位）	R_i（32 位）
第 1 组	665252858	12	35	12	16408
第 2 组	12	16408	6479845124	16408	4246583235
第 3 组	16408	4246583235	789416132	4246583235	1648728035
第 4 组	4246583235	1648728035	12321346456778	1648728035	242915767
第 5 组	1648728035	242915767	36127856899	242915767	3389921152

表 4.9　DES 解密过程测试结果

	输入数据			输出数据	
	L_i（32 位）	R_i（32 位）	K_i（48 位）	L_{i-1}（32 位）	R_{i-1}（32 位）
第 1 组	3389921152	242915767	36127856899	1648728035	242915767
第 2 组	242915767	1648728035	12321346456778	4246583235	1648728035
第 3 组	1648728035	4246583235	789416132	16408	4246583235
第 4 组	4246583235	16408	6479845124	12	16408
第 5 组	16408	12	35	665252858	12

4.3 AES 算法的映射与仿真验证

4.3.1 AES 算法映射

按照 AES 算法的需要合理的设定各可重构运算模块的配置位，得到 AES 算法中各运算模块，再设定好各多路选择器的选择位，形成 AES 算法的整个加/解运算路径，最终达到 AES 密码算法加/解密运算的目的。

AES 加密轮函数在 DES/AES 可重构密码处理结构上映射过程如图 4.9 所示，其映射过程是：AES 算法的 128 位明/密文分组分成 4 个 32 位子块并行进行处理，32 位明文子块（$plaintext_i$ 表

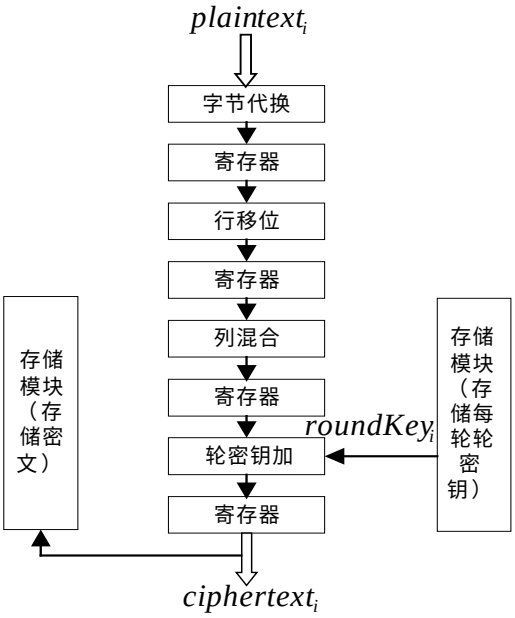


图 4.9　AES 加密轮函数在可重构密码处理结构上的映射图

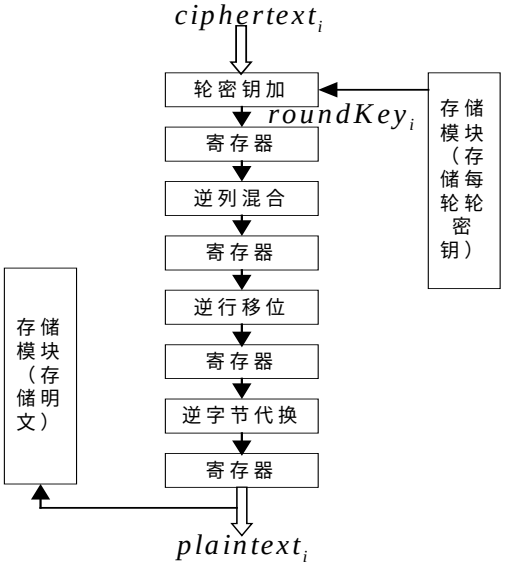


图 4.10　AES 解密轮函数在可重构密码处理结构上的映射图

示第 i 个明文分组的 32 位子块）经过 AES 各个运算模块的处理后输出到图中左边的存储器中，由于采用了流水线结构，每个 32 位明文子块的密文输出（$ciphertext_i$ 表示第 i 个密文分组的 32 位子块）相隔 1 个时钟周期，每轮运算需要 5 个时钟周期。

AES 解密轮函数在 DES/AES 可重构密码处理结构上映射过程如图 4.10 所示，其映射过程和加密轮函数类似，只是所有运算模块全部变成了相应的逆运算模块。

4.3.2 仿真验证

AES 的加密和解密是两个不同的通路，具体是通过多路器的路径选择信号进行控制的，而且解密时运算模块都是加密时运算模块的逆运算，解密时很多可重构运算模块需要设定不同的配置位来实现它的逆变换。图 4.11 和 4.12 给出了 AES 32 位轮函数的仿真波形，其中图 4.11 是加密时的仿真波形，而图 4.12 是解密时的仿真波形，图中的信号端口可参见图 4.5 的可重构处理模块电路结构图，在加密过程中，用到的多路器路径选择信号为：path_sel2=“000”, path_sel4=“011”, path_sel5=“101”, path_sel6=“110” , path_sel_out=“111”；而在解密过程中，用到的多路器路径选择信号为：path_sel2=“101”, path_sel4=“110”, path_sel5=“111”, path_sel6=“000” , path_sel_out=“011”。 图 4.11 和 4.12 中测试数据的测试结果如表 4.10 和 4.11 所示，由表 4.10 和 4.11 可见，将加密时 AES 轮函数的输出结果作为解密轮函数的输入数据完全能够得到原来用于加密时轮函数的输入数据，所以，验证了所设计的可重构密码处理模块完全能够正确实现 AES 算法。

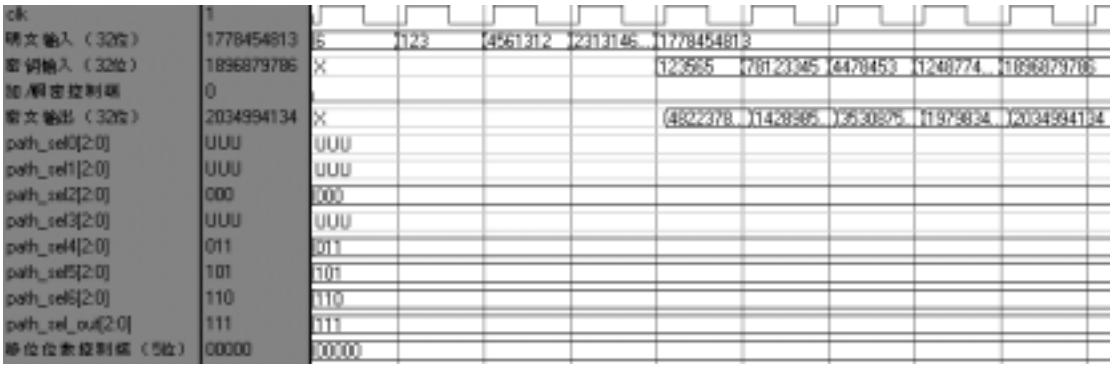


图 4.11　加密时的 AES 轮函数映射验证仿真图

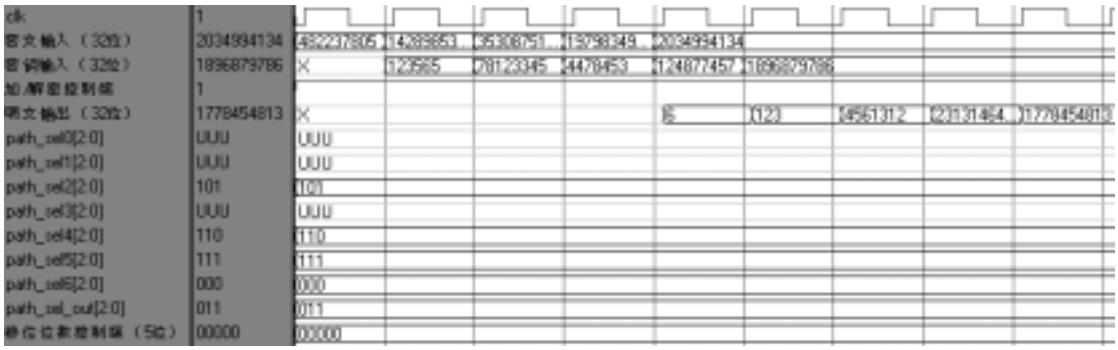


图 4.12　解密时的 AES 轮函数映射验证仿真图

表 4.10　AES 加密过程测试结果

输入数据（32 位）		输出数据（32 位）
明文	密钥	密文
6	123565	482237805
123	78123345	1428985319
4561312	4478453	3530875152
2313146454	124877457	1979834966
1778454813	1896879786	2034994134

表 4.11　AES 解密过程测试结果

输入数据（32 位）		输出数据（32 位）
密文	密钥	明文
482237805	123565	6
1428985319	78123345	123
3530875152	4478453	4561312
1979834966	124877457	2313146454
2034994134	1896879786	1778454813

4.4 本章小结

本章的目的是验证第三章所设计的可重构密码处理结构模型 RCPA 的正确性，以 DES 和 AES 算法的可重构实现为例，采用 VHDL 硬件描述语言，实现了 DES/AES 可重构密码处理结构，并分别对 DES 和 AES 进行了算法映射，最后给出了仿真波形和性能分析。实验结果证实以 RCPA 为模型设计的 DES/AES 可重构密码处理结构是完全正确的，而且与其他实现方案对比表明本设计具有更快的密码处理速度和更少的配置位。

第五章 高性价比 AES 密码系统的 VLSI 实现

美国高级加密标准(Advanced Encryption Standard，AES)[34]算法已经成为代替 DES 算法的新一代国际加密标准，占领了通信网络、银行、军队通讯等各个领域，应用前景广阔。但是 AES 算法的加/解密过程运算复杂，耗费大量的处理器时间。如果采用软件方案[35]实现，必将影响其性能；若采用硬件方案实现，将会大大地降低处理器的负担，同时能有效的提高加/解密效率和安全性。国内对相对较新的 AES 算法的研究还比较少，相应的硬件产品更是不多，很多硬件实现方案只考虑数据的加/解密速度或是硬件开销，很难综合考虑这两方面。因此，AES 密码系统在硬件开销和加/解密速度要求苛刻的领域(如高速网络中的加/解密设备)中的应用受到限制。

本章正是基于上述考虑，从硬件的角度出发改进和实现了 AES 算法。为了减少了硬件开销，本设计采用模块复用技术，对 AES 的两个核心计算部件（字节代换和列混合）进行了硬件可逆设计；为了提高了密码系统的数据吞吐量(数据吞吐量是衡量加/解密速度的标准)，采用流水线结构；同时设计的密钥扩展模块很好的解决了密钥分配的同步问题。从而使整个加/ 解密系统具有更高的性价比。AES 算法介绍见 4.1.2 节。

字节代换和列混合两个运算部件是 AES 密码系统的核心部件，占了整个系统的 80%以上[25]，所以，对其进行精心设计，能够很大程度的提高 AES 密码系统的整体性能，以下内容介绍了如何采用模块复用技术来实现字节代换和列混合这两个核心运算部件，同时考虑到这两个运算部件有较大的延时，采用了流水线结构来减少关键路径延时，本章图中的虚线表示流水线实现。

5.1 字节代换(ByteSub)硬件可逆设计与流水线实现

字节代换是 8-bit 输入到 8-bit 输出的非线性变换，独立地对状态的每个字节进行。这里的字节代换其实就是一个 8*8 的 S 盒。

传统的 AES 算法都使用了查找表(lookup table) [25]的方法来实现字节代换，不仅消耗大量的存储资源，而且有较大的延时。本文通过将 GF(2^8)域中元素变换到其复合域 GF($(2^4)^2$)，则可用组合逻辑替代查找表，并采用流水线设计方法加以实现。这样不仅节省了硬件存储单元，而且大大提高了 AES 算法模块的处理速度。更主要的是字节代换的逆变换与字节代换共用同一个硬件结构，而不用像查找表实现字节代换和其逆变换时要用两个查找表分别实现。这样更大的节省了硬件开销。

字节代换(ByteSub)由以下两个变换的合成得到：

(1) 将字节看作 GF(2^8)上的元素，映射到自己的乘法逆元，‘00’映射到自己；

(2) 对字节做（GF(2)上的，可逆的)仿射变换。

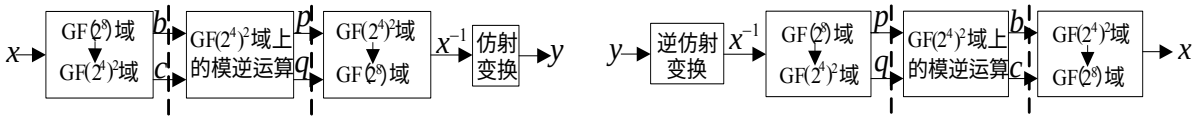
5.1.1 仿射变换及其逆变换的硬件实现

仿射变换用基本的异或门就可以实现了，具体变换可参考文献[12]。

5.1.2 GF(2^8)域中模逆运算的硬件设计与流水线实现

GF(2^8)域中模逆运算的硬件实现比较复杂，一种比较有效的实现方案将域 GF(2^8)分解为两个子域 GF$(2^4)^2$ 的复合，通过在 GF$(2^4)^2$ 域中求逆来代替 GF(2^8)域中求逆，域 GF(2^8)叫做扩展域，GF(2^4)域就是 GF(2^8)域的子域。

图 5.1 是复合域中字节代换和逆字节代换的流程图，其中字节代换用于加密过程，而逆字节代换用于解密过程。



(a) 复合域中字节代换流程图　　(b) 复合域中逆字节代换流程图

图 5.1　复合域中字节代换和逆字节代换的流程图

由有限域的知识可知，复合域 GF$(2^4)^2$ 中每一个元素都可表示为系数在 GF(2^4)上的一次多项式 bx+c，复合域上乘法定义为模一个二次不可约多项式 x^2+Ax+B 的多项式乘法（A,B 是域 GF(2^4)中的常数，一般取 A=1,B=9）。设定义有限域 GF$(2^4)^2$ 的乘法的二次不可约多项式为 x^2+Ax+B，则易验证此时 GF$(2^4)^2$ 中任一元素 bx+c 的乘法逆元是：

$$(bx+c)^{-1}=b(b^2\mathrm{B}+bc\mathrm{A}+c^2)^{-1}x+(c+b\mathrm{A})(b^2\mathrm{B}+bc\mathrm{A}+c^2)^{-1} \tag{5-1}$$

其中 $(b^2\mathrm{B}+bc\mathrm{A}+c^2)^{-1}$ 是 $(b^2\mathrm{B}+bc\mathrm{A}+c^2)$ 在 GF(2^4)上的乘法逆元。

式(5-1)求逆运算的结构示意图如图 5.2 所示：

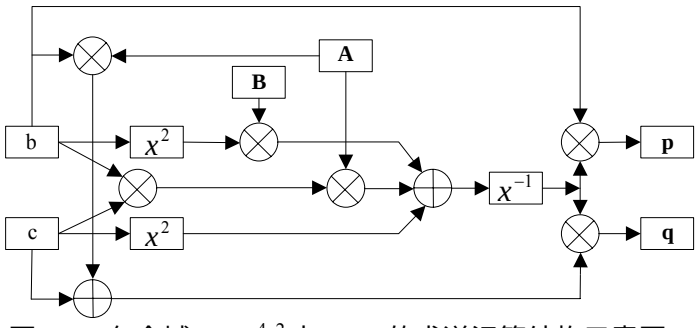


图 5.2　复合域 GF$(2^4)^2$ 中 bx+c 的求逆运算结构示意图

图 5.2 中 $p=b(b^2\mathrm{B}+bc\mathrm{A}+c^2)^{-1}, q=(c+b\mathrm{A})(b^2\mathrm{B}+bc\mathrm{A}+c^2)^{-1}$。

图 5.2 被执行之前，首先要将 GF(2^8)域中的元素映射为以 GF(2^4)域中的数为系数的多项式，这种映射是可逆的。本文采用了 O′Driscoll 转换矩阵来实现这种映射关系，O′Driscoll 转换矩的等效方程及其逆映射方程见文献[36]。

5.1.3 字节代换整体可逆设计与流水线

将字节代换和它的逆变换统一起来就可以实现字节代换可逆变换，为了减少关键路径的延

时，采用流水线结构，字节代换可逆设计及流水线结构如图 5.3 所示。

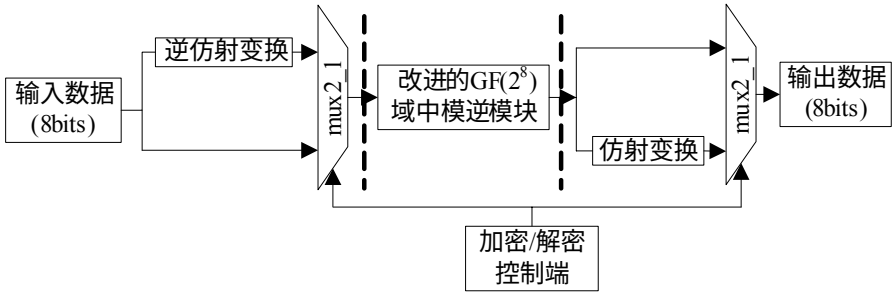


图 5.3　字节代换可逆设计与流水线实现的结构图

5.2 列混合(MixColumn)硬件可逆设计与流水线实现

在列混合变换中，将状态阵列的每个列视为 $GF(2^8)$上的多项式，再与一个固定的多项式 $p(x)$ 进行模 x^4+1 乘法。当然要求 $p(x)$是模 x^4+1 可逆的多项式，否则列混合变换就是不可逆的，会使不同的输入分组对应的输出分组可能相同。Rijindael 的设计者给出的 $p(x)$为（系数用十六进制数表示）：

$$p(x) = \text{‘03’}x^3 + \text{‘01’}x^2 + \text{‘01’}x + \text{‘02’} \tag{5-2}$$

$p(x)$是与x^4+1互素的，因此是模x^4+1可逆的。列混合变换的表达式为$b(x) = p(x) \cdot a(x)$, 也可以写为矩阵乘法，其矩阵表示如式（5-3）所示。

$$\begin{bmatrix} b_3 \\ b_2 \\ b_1 \\ b_0 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} \tag{5-3}$$

而在逆列混合变换中，每一列通过乘以 $p^{-1}(x)$进行逆变换。其中

$$p^{-1}(x) = \text{‘0b’}x^3 + \text{‘0d’}x^2 + \text{‘09’}x + \text{‘0e’} \tag{5-4}$$

对比$p(x)$和$p^{-1}(x)$表达式，可以发现$p^{-1}(x)$的系数比$p(x)$的系数复杂多了，这就意味逆混合变换的硬件实现比混合变换硬件实现要复杂的多。为了降低逆变换硬件实现的复杂度，我们利用$p(x)$对$p^{-1}(x)$进行分解。一般有两种分解方式：并行分解方式和串行分解方式。

并行分解方式如式（5-5）所示：

$$p^{-1}(x) = p(x) + e(x) \tag{5-5}$$

其中$e(x) = \text{‘08’}(x^3+x) + \text{‘0c’}(x^2+1)$。

串行分解方式如式（5-6）所示：

$$p^{-1}(x) = p(x) \cdot d(x) \tag{5-6}$$

其中$d(x) = p^{-2}(x) = \text{‘04’}x^2 + \text{‘05’}$。

由式（5-5）和（5-6）可以看出，由于串行分解只有两项系数，而且系数都比较小，所以串行分解能够产生更有效的乘法实现，也更能节省硬件资源。

5.2.1 列混合变换的硬件实现

对 32 位输入 $a = a_3a_2a_1a_0$，这里每一个字节 a_i 是 8 位的，列混合变换的矩阵表示如式（5-3）所示。通过式（5-3）易得下列等式：

$$\begin{cases} b_3 = \text{‘02’}(a_3 + a_2) + \text{‘01’}(a_2 + a_1 + a_0) \\ b_2 = \text{‘02’}(a_2 + a_1) + \text{‘01’}(a_3 + a_1 + a_0) \\ b_1 = \text{‘02’}(a_1 + a_0) + \text{‘01’}(a_3 + a_2 + a_0) \\ b_0 = \text{‘02’}(a_3 + a_0) + \text{‘01’}(a_3 + a_2 + a_1) \end{cases} \tag{5-7}$$

根据上式可以得到列混合变换的硬件实现结构，列混合硬件实现结构如图 5.4 所示，图中的 xtime 运算介绍可参考文献[12]。

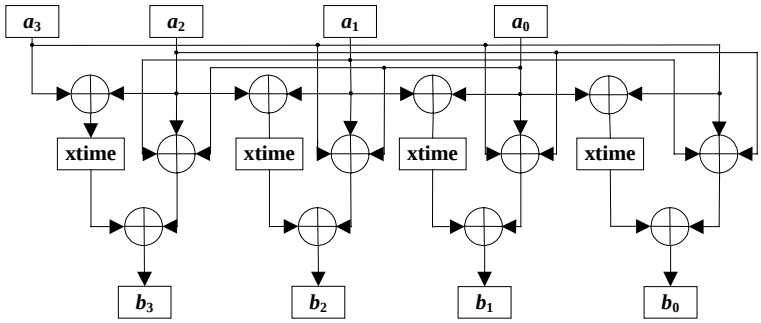


图 5.4　列混合硬件实现结构图

5.2.2 列混合可逆设计与流水线实现

逆列混合变换的数学表达式为$a(x)= b(x) \cdot p^{-1}(x)$，通过式(5-6)，逆列混合变换的数学表达式可以变为：

$$a(x) = b(x) \cdot p^{-1}(x) = b(x) \cdot p(x) \cdot d(x) \tag{5-8}$$

1. 乘 $d(x)$模块的实现

设输入 $c = c_3c_2c_1c_0$，输出 $a = a_3a_2a_1a_0$。乘 $d(x)$模块的表达式为 $a(x) = d(x) \cdot c(x)$, 由于 $d(x) = p^{-2}(x) = \text{‘04’}x^2 + \text{‘05’}$，其表达式如式（5-9）所示：

$$\begin{cases} a_3 = \text{‘04’}(c_3 + c_1) + \text{‘01’}c_3 \\ a_2 = \text{‘04’}(c_2 + c_0) + \text{‘01’}c_2 \\ a_1 = \text{‘04’}(c_3 + c_1) + \text{‘01’}c_1 \\ a_0 = \text{‘02’}(c_2 + c_0) + \text{‘01’}c_0 \end{cases} \tag{5-9}$$

根据上式可以得到乘 $d(x)$模块的硬件实现结构，乘 $d(x)$模块硬件实现结构如图 5.5 所示。

2. 列混合可逆设计与流水线实现

将列混合变换和它的逆变换统一起来就可以实现列混合可逆变换，为了减少关键路径的延时，采用流水线结构，列混合可逆设计及流水线结构如图5.6所示。

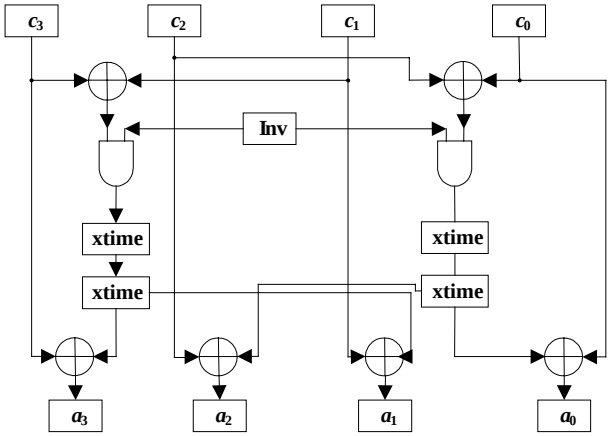


图 5.5　逆列混合变换中的乘 $d(x)$模块结构图

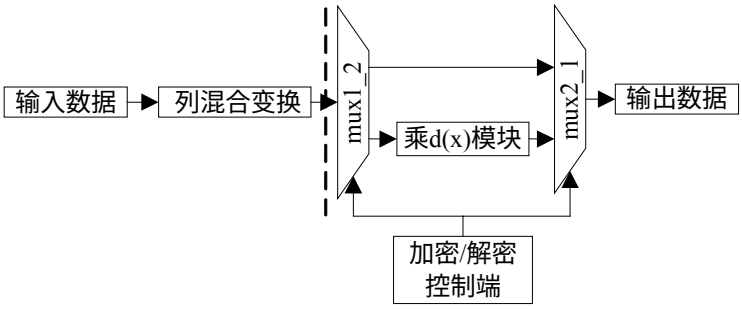


图 5.6　列混合可逆设计与流水线实现的结构图

5.3 AES 加/解密系统的整体实现

5.3.1 AES 轮间轮内双流水结构

由于 AES 算法是轮加/解密结构，为了提高运算速度与数据吞吐率，本文采用了轮间和轮内双流水线的 VLSI 结构，所谓轮间流水线结构，即在 10 轮中的每一轮之间都增加一级流水线操作，可以使数据吞吐率提高 10 倍；而轮内流水结构，即在每一轮内增加流水级数，从而将每一轮的关键路径缩短，最高时钟频率可以得到相应的提高。本文采用的轮间和轮内相结合的流水线 VLSI 结构如图 5.7 所示，该结构可以大幅度提高系统的时钟频率和系统数据吞吐率。

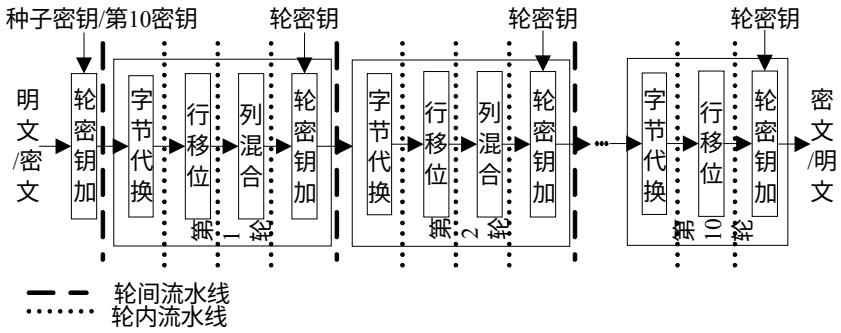


图 5.7　AES 的轮间轮内双流水线结构

本设计中各运算部件都是可逆的，有一个加/解密控制信号(sel)对其控制，当进行加密运算时，sel 为‘0’，输入明文和种子密钥，输出密文和第 10 轮轮密钥。当进行解密运算时，sel 为‘1’，输入密文和第 10 轮轮密钥，输出明文和种子密钥。1～9 轮每轮加/解密运算需要 10 个周期，最后一轮加/解密需要 8 个时钟周期。

5.3.2 密钥扩展

高速加/解密运算时轮密钥分配的同步问题一直都是高速加/解密系统的关键问题，现今密码系统的高速实现基本上都是采用将每轮的轮密钥预先计算出来，存储到 RAM 或寄存器中，加/解密的时候读取轮密钥即可。这种方式的缺点是：对 RAM 的读写具有较长的延时时间，特别是预存每轮轮密钥时要花费大量的时间来写 RAM 或寄存器，从而降低了整个系统的加/解密速度；此外，需要大量的 RAM 或寄存器来存储轮密钥，硬件开销大。本文设计了一种轮密钥扩展结构，能够很好解决上述问题，采用流水线设计，和加/解密运算保持同步，其结构如图 5.8 所示。图中，加密时，取 32 位字节代换运算结果高 8 位和轮常数进行 8 位异或运算，然后再将 8 位异或运算结果作为高位和字节代换运算结果的低 24 组合成新的 32 位数输入 32 位异或运算模块中；同样解密时，取 32 位异或运算结果高 8 位和轮常数进行 8 位异或运算，然后再将 8 位异或运算结果作为高位，和 32 位异或运算结果的低 24 组合成新的 32 位数输入 2 选 1 数据选择器中。最后一个 2 选 1 数据选择器的输入是 2 个 128 位数，加密(即 sel=‘0’)时，输入的高 32 位是 32 位异或运算的输出，低 96 位是字输入密钥的高 96 位；解密(即 sel=‘1’)时，输入的高 96 位是输入密钥的低 96 位，低 32 位是字节代换的 32 位输出。

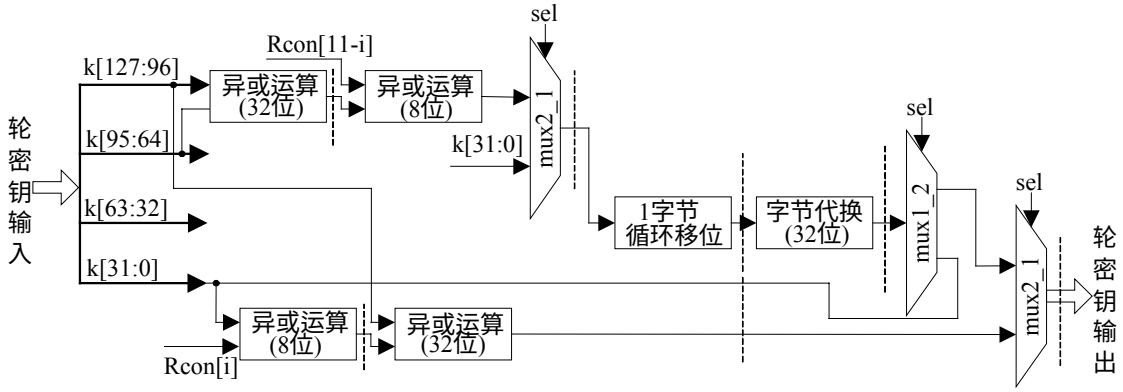


图 5.8　密钥扩展模块结构

由于加/解密使用同样的密钥扩展轮结构，而真正用于加密和解密运算的密钥是不一样的，加密的轮密钥是图 5.8 密钥扩展结构加密时产生的输出轮密钥，而解密的轮密钥是密钥扩展结构解密时产生的输出轮密钥后，再经过一个逆列混合变换得到的，密钥变换结构如图 5.9 所示。

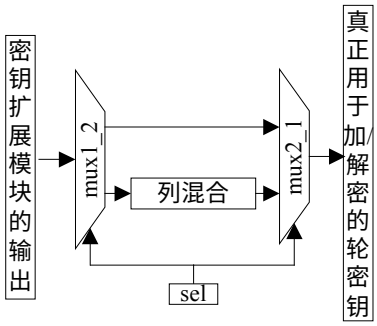


图 5.9　密钥变换结构

5.4 仿真验证与实验结果分析

本文先对 AES 密码系统的加/解密结构进行逻辑功能设计，用 VHDL 进行自顶向下的层次化编程，并采用 Xilinx 公司的 ISE6.3i 编译综合后，在 ModelSim6.0 下进行仿真验证，仿真波形如图 5.10 和 5.11 所示，其中图 5.10 是加密时的仿真波形，而图 5.11 是解密时的仿真波形。

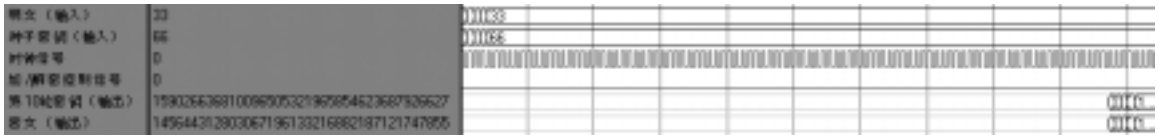


图 5.10　AES 密码系统加密仿真图

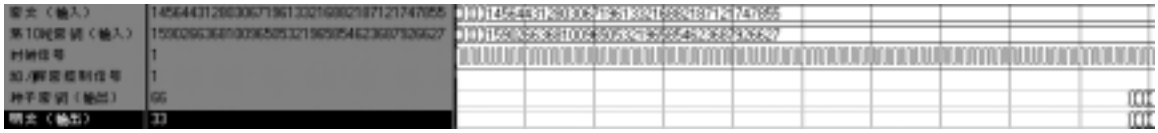


图 5.11　AES 密码系统解密仿真图

表 5.1　加密过程测试结果

	明文(输入)	种子密钥(输入)	最后一轮轮密钥(输出)	密文(输出)
第 1 组	1245	45	63319990066095012206 136818817011180387	20540709177616235085 0248798534294915042
第 2 组	18	134	99693733253995858736 269065396173628259	48000950176471321590 766542976644190571
第 3 组	33	66	15902663681009650532 1965854623687926627	14564431280306719613 3216882187121747855
第 4 组	8797967979798	123144564456	129144588506062100 98 335238036593829606	26208255802416564050 0957290902556678767
第 5 组	7451234781455 41544441444	1275464154568 979624898695	28740801036270826376 1411824419372858715	54658722159790911681 735903015409785177

由图 5.10 和 5.11 所示，明文 128 位第 1 组分组输入数据经过 98 个时钟周期排空后，后面

每一组输入数据经过一个时钟周期输出密文和第 10 轮的密钥，图中的数据全部采用无符号数表示。表 5.1 和表 5.2 给出了几组加/解密的测试结果，表中的数据全部采用无符号数表示，由表 5.1 和表 5.2 可见，把加密后的密文和最后一轮轮密钥输入此 AES 密码系统进行解密运算，完全可以得到原始的明文和种子密钥，从而验证了整个系统的正确性。FPGA 验证时，本文使用 Xilinx 公司的 Virtex 系列的 XCV3200EFG1156 器件，最高频率达到 133.067MHz，吞吐率可达到 15.86Gbps，面积为 493.47k 等效门。由表 5.3 可见，与其他硬件实现方案相比，本文的设计具有更高的性价比。

表 5.2　解密过程测试结果

	密文(输入)	最后一轮轮密钥(输入)	种子密钥(输出)	明文(输出)
第 1 组	20540709177616235085 0248798534294915042	63319990066095012206 136818817011180387	45	1245
第 2 组	48000950176471321590 766542976644190571	99693733253995858736 269065396173628259	134	18
第 3 组	14564431280306719613 3216882187121747855	15902663681009650532 1965854623687926627	66	33
第 4 组	26208255802416564050 0957290902556678767	129144588506062100 98 335238036593829606	123144564456	8797967979798
第 5 组	54658722159790911681 735903015409785177	28740801036270826376 1411824419372858715	1275464154568 979624898695	7451234781455 41544441444

表 5.3　各 AES 硬件实现方案的性能比较

	Zambreno[37]	Jarvinen[38]	Good[39]	Rudra[40]	本设计
仿真平台 (*FPGA* 型号)	*Virtex* − ∏ *XC2V*4000	*Virtex* − *E* *XCV*1000*C* − 8	*Virtex* − *E* *XCV*2000*E* − 8	ASIC	*Virtex* − *E* *XCV*3200*E* − 6
面积（KGates）	2930.274	2285.205	3104.898	256	493.47
最高时钟 （MHz）	184.1	129.2	184.8	32.0	133.067
吞吐率（Gbps）	23.57	16.50	23.65	7.5	15.86
性价比($\frac{\text{吞吐率}}{\text{面积}}$) (*Kbps* / *gates*)	8.04	7.22	7.62	29.3	32.14

5.5 本章小结

本章研究了一种高性价比的 AES 加/解密系统的 VLSI 实现方案。首先，为了减少硬件开销，对 AES 的两个核心运算部件（字节代换和列混合）进行了硬件可逆设计；其次，为了提高加/解密速度，采用了轮间和轮内相结合的流水线结构；最后，设计了一种轮密钥扩展结构，解决

了高速加/解密时轮密钥分配的同步问题。实验结果表明该设计与其它同类设计相比，具有更高的性价比。

第六章 总结与展望

6.1 总结

密码处理灵活性和高效性的结合一直是密码算法应用的追求目标，采用可重构计算技术来设计高效灵活的密码算法硬件正逐渐成为研究的一个新的热点。本文针对分组密码处理应用领域，较深入的研究了适于该领域的可重构密码处理结构模型和可重构密码运算模块的设计方法，力求使该可重构密码处理结构模型能够适应各种分组密码算法的处理需要，获得灵活性和高效性的折衷。本文已完成的研究工作包括：

（1）详细分析了当前国内外可重构密码处理的研究现状，分析了本课题研究背景、目的和意义。

（2）对现有的主要分组密码算法操作特征以及处理结构特点进行了研究与分析，主要包括了分组密码算法的操作特征以及基本运算模块的构成。分析表明分组密码算法具有较为规律的运算位宽，适合分组内的横向并行以及分组间的纵向流水处理，为可重构处理结构的设计提供了依据。

（3）根据分组密码处理结构的特点，设计了一种可重构密码处理结构模型 RCPA。该模型的设计特点是具有可变的并行度以及可配置的流水结构，可从横向和纵向两个方向组织可重构处理模块。

（4）对可重构密码处理 RCPA 模型中的可重构密码运算模块 RCM 进行了研究和设计。RCM 设计包括了可重构 S 盒替代模块、可重构移位模块、可重构置换模块、模加/减、模乘等可重构密码运算模块。这些基本运算模块可根据 RCM 配置指令进行重构，灵活完成不同算法所需的运算功能。各重构密码运算模块 RCM 在 FPGA 上进行了综合，并得到正确的仿真结果。

（5）基于密码处理结构模型 RCPA，具体针对 DES 和 AES 算法，采用 VHDL 硬件描述语言，完成了 DES/AES 可重构密码处理结构的实现，并分别对 DES 和 AES 进行了算法映射，给出了仿真波形和性能分析。

（6）研究了一种高性价比的 AES 密码系统 VLSI 实现方案，同时解决了高速加/解密时轮密钥分配的同步问题，此实现方案设计的 AES 密码系统与其他硬件实现方案比较，具有更高的性价比，文中给出了对比数据。

6.2 展望

虽然本文的研究取得可一定的进展，在一定程度上解决了灵活性和高效性之间的矛盾，提高了算法本身的安全性，但是由于时间和工作量的关系，我们的工作还不完善，对下一步的研究工作的设想和建议如下：

（1）本文所设计的可重构运算模块 RCM 还有较大的电路延时，下一步要做的是进一步优化各可重构密码运算模块，使其具有更小的电路延时，从而使整个密码处理模块的主频得以提高。

（2）目前本文根据密码处理结构模型 RCPA 构建的可重构密码处理结构只适用于少量的分组密码算法，如 DES、AES，下一步所需要做的是设计更多的可重构运算模块 RCM，构建适应更多密码算法的可重构密码处理结构，使整个可重构密码处理结构具有更好的适应性。

（3）本文设计的 RCPA 模型只适合于分组密码算法处理，下一步需要对该模型进行研究和改进，将其适用于 RSA 等公钥密码算法处理。因此，如何在统一的重构处理框架下高效灵活地支持不同种类的密码算法成为接下来研究的重点。

（4）对于密码算法的硬件实现研究，最终的目的是设计实现集成的密码芯片或密码系统。本人所从事的密码算法的可重构研究，侧重于关键技术研究，实验仿真。密码系统的芯片集成，是一项困难的课题，进行可重构密码芯片集成的目的是要设计专门的集成电路芯片或集成电路协处理器，以便利用该集成电路芯片或协处理器实现多个算法，如何规划和设计这样的集成电路芯片或集成电路协处理器，是可重构密码芯片集成中需要进一步研究的问题。

参考文献

[1] 曲英杰，李占才，王沁，适用于可编程加密芯片的可重组体系结构，计算机工程与应用，2001，37(19)：73~75

[2] Zong Wang, Arslan T, Erdogan A, Implementation of Hardware Encryption Engine for Wireless Communication on a Reconfigurable Instruction Cell Architecture, Proceedings of the 4th IEEE International Symposium on Electronic Design Test and Applications, Piscataway, NJ, USA: IEEE, 2008: 148~152

[3] 张焕国，冯秀涛，刘玉珍，演化密码与 DES 密码的演化研究，计算机学报，2003，26(12)：1678~1684

[4] 倪晓强，并行向量密码处理的机构设计与研究［硕士学位论文］，杭州，浙江大学，2005

[5] Seth Copen Goldstein, Herman Schmit, Mihai Budiu, et al., PipeRench:A Reconfigurable Architecture and Compile, IEEE Transactions on Electronic Computers, 2000, 18(9): 70~77

[6] M. H. Lee, H. Singh, G. Lu, et al., Design and Implementation of the MorphoSys Reconfigurable Computing Processor, Journal of VLSI Signal Processing Systems, 2000, 24(2): 147~164

[7] Carl Ebeling, Darren C. Cronquist, Paul Franklin, Mapping applications to the RaPiD configurable architecture, Proceedings of the 5th Annual IEEE Symposium on FPGAs for Custom Computing Machines, Los Alamitos, CA: IEEE Computer Society Press, 1997: 106~115

[8] A.J. Elbirt, Instruction-Level Distributed Processing for Sysmmetric-Key Cryptography, Proceedings of the 19th International Parallel and Distributed Processing Symposium(IPDPS'05), Piscataway, NJ, USA: IEEE, 2005: 468~480

[9] 白永强，面向多媒体的粗粒度可重构处理单元的结构研究［硕士学位论文］，西安，西北工业大学，2006

[10] 姜晶菲，可重构密码处理结构的研究与设计[博士学位论文]，长沙，国防科学技术大学，2004

[11] 曲英杰，可重组密码逻辑的研究与设计[博士学位论文]，北京，北京科技大学，2001

[12] 杨波，现代密码学(第 2 版)，北京，清华大学出版社，2007：41~68

[13] Deng Yanxiang, Hwang C. J., Liu Jianglung, An object-oriented cryptosystem based on two-level reconfigurable computing architecture, Journal of Systems and Software, 2006, 79(4): 466~479

[14] 杨晓辉，面向分组密码处理的可重构设计技术研究[硕士学位论文]，郑州，解放军信息工程大学，2007

[15] Chungping Young, Chungchu Chia, Liangbi Chen, et al., NCPA: A Scheduling Algorithm for Multi-cipher and Multi-mode Reconfigurable Cryptosystem, Proceedings of International Conference on Intelligent Information Hiding and Multimedia Signal Processing(IIHMSP'08), Piscataway, NJ, USA: IEEE, 2008: 1356~1359

[16] 孙万忠，戴紫彬，张永福，一种重构型数据加密卡的硬件设计与实现， 计算机工程与应用，2007，43(13)：111~142

[17] 冯登国，吴文玲，分组密码的设计与分析，北京，清华大学出版社，2000：67~69

[18] Nadia Nedjah, Luiza de Macedo Mourelle, Designing substitution boxes for secure ciphers, International Journal of Innovative Computing and Applications, 2007, 1(1): 86~91

[19] E.C. Laskari, G.C. Meletiou, Applying evolutionary computation methods for the cryptanalysis of Feistel ciphers, Applied Mathematics and Computation, 2007, 184 (1): 63~72

[20] 殷新春，杨洁，基于遗传算法的 S 盒的构造，计算机应用研究，2007，24(3)：91~93

[21] M. Macchetti, Wenyu Chen, ASIC hardware implementation of the IDEA NXT encryption algorithm, Proceedings of IEEE International Symposium on Circuits and Systems(ISCAS'06), Piscataway, NJ, USA: IEEE, 2006, 4843~4846

[22] R.M. Dansereau, S. Jin, R.A. Goubran, Reducing Packet Loss in CBC Secured VOIP using Interleaved Encryption, Proceedings of Canadian Conference on Electrical and Computer Engineering (CCECE'06), Piscataway, NJ, USA: IEEE, 2006, 1320~1324

[23] Zibin Dai, Wei Li, Xiaohui Yang, et al., The Research and Implementation of Reconfigurable Processor Architecture for Block Cipher Processing, Proceedings of International Conference on Embedded Software and Systems (ICESS'08), Piscataway, NJ, USA: IEEE, 2008: 587~594

[24] Andrew W. H. House, Howard M. Heys, Design of a flexible cryptographic hardware module, Proceedings of Canadian Conference on Electrical and Computer Engineering, Piscataway, NJ, USA: IEEE, 2004: 603~608

[25] 高娜娜，李占才，王沁，一种可重构体系结构用于高速实现 DES、3DES 和 AES，电子学报，2006，34(8)：1386~1390

[26] 陈永强，李茜，基于 FPGA 的可变长度移位寄存器优化设计，电子技术应用，2006，(8)：68~70

[27] David Narth Amanor, Christof Paar, Jan Pelzl, et al., Efficient Hardware Architectures for Modular Multiplication on FPGAs, Proceedings of International Conference on Field Programmable Logic and Application(FPL'05), Piscataway, NJ, USA: IEEE, 2005: 539~542

[28] Zhou Shun, O. A. Pfander, H. J. Pfleiderer, A VLSI architecture for a Run-time Multi-precision

Reconfigurable Booth Multiplier, Proceedings of the 14th IEEE International Conference on Electronics Circuits and Systems(ICECS'07), Piscataway, NJ, USA: IEEE, 2007: 975~978

[29] 董兰飞，高性能 64 位并行整数乘法器全定制设计与实现[硕士学位论文]，长沙，国防科学技术大学，2006

[30] 常静波，郭立，一种 3 级流水线 wallace 树压缩器的硬件设计，微电子学与计算机，2005，22(1)：160~165

[31] Gurkaynak F.K., Gasser D, Hug F, et al., A 2Gb/s balanced AES crypto-chip implementation, Proceedings of the 14th ACM Great Lakes Symposium on VLSI(GLSVLSI'04), Piscataway, NJ, USA: IEEE, 2004: 39~44

[32] Ingrid Verbauwhede, Design and Performance Testing of a 2.29Gb/s Rijndael Processor, IEEE Journal of Solid-state Circiuts, 2003, 38(3): 569~572

[33] 高娜娜，王沁，李占才，基于可重构 S 盒的常用分组密码算法的高速实现，计算机工程，2006，32(9)：253~255

[34] 肖国镇，白恩健，刘晓娟，AES 密码分析的若干新进展，电子学报，2003，31(10)：1549~1554

[35] Abdul S, Arshad A, Nassar I, An Efficient Software Implementation of AES-CCM for IEEE 802.11i Wireless Standard, Proceedings of 31st Annual International Computer Software and Applications Conference, Los Alamitos, CA: IEEE Computer Society Press, 2007: 689~694

[36] Cillian O 'Driscoll, Hardware Implementation Aspects of the Rijndael Block Cipher, National University of Ireland, Belfield, 2001

[37] J. Zambreno, A.Choudhary, D. Nguyen, Exploring Area/Delay Tradeoffs in an AES FPGA Implementation, Proceedings of the 14th International Conference on Field Programmable Logic and its Applications(FPL'04), Berlin, Germany: Springer-Verlag, 2004: 575~585

[38] K. Jarvinen, J.O. Skytta, M.T. Tommiska. A Fully Pipelined Memoryless 17.8 Gbps AES-128 Encryptor, Proceedings of the 2003 ACM/SIGDA Eleventh International Symposium on FPGAs, Monterey, CA, United States: Association for Computing Machinery, 2003: 207~215

[39] T. Good, M. Benaissa, AES on FPGA from the Fastest to the Smallest, Proceedings of the 7th Cryptographic Hardware and Embedded Systems, Berlin, Germany: Springer-Verlag, 2005: 427~440

[40] A. Rudra, P.K. Dubey, C.S. Jutla, et al., Efficient Rijndael Encryption Implementation with Composite Field Arithmetic, Proceedings of the Cryptographic Hardware and Embedded Systems Conference, Berlin, Germany: Springer-Verlag, 2001: 171~184

致 谢

在论文即将完成之际，首先以最诚挚的心向所有给予我指导、关心和支持的老师、同学、亲人表示最衷心的感谢！

感谢我的导师王友仁教授！论文顺利完成离不开他的悉心指导。三年的学习生活中，他是我的良师益友，对我关怀备至，让我在愉快的氛围中完成了学业。王老师的渊博的学识，严谨的治学态度，忘我的工作精神给我留下了深刻的印象，使我受益匪浅。在此，向王老师表达我深深的敬意和感谢！

在课题研究中，我还得到了实验室老师及同学的帮助，他们对我得课题提出了不少的建议，感谢实验室的同学和师弟们。

我还要感谢我的父母，对于我的成长，他们付出了毕生的精力和心血，没有他们的谆谆教导和辛勤劳动就没有我的今天。

最后，向审阅本文的专家、教授致以深深的敬意和谢意。

在学期间的研究成果及发表的学术论文

攻读硕士学位期间发表（录用）论文情况

1. 郑东, 王友仁. AES 中字节代换和列混合的硬件可逆设计. 计算机技术与发展. 已录用
2. 郑东, 王友仁, 张砦. 基于遗传算法的快速可重构 S 盒硬件设计. 信息与控制. 拟录用

可重构分组密码处理系统实现研究

作者： 郑东
学位授予单位： 南京航空航天大学