中国人民解放军信息工程大学

硕士学位论文

面向分组密码处理的可重构设计技术研究

姓名：杨晓辉

申请学位级别：硕士

专业：计算机应用技术

指导教师：戴紫彬

20070401

摘 要

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：

论文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，结合可重构处理结构的设计方法，提出了一种可重构密码处理结构模型 RCPA。该模型的设计特点是粗粒度、混合型互连网络、类 VLIW/EPIC 计算模型、动态与静态配置相结合的可重构处理结构。

论文研究了分组密码处理基本元素重构技术，设计了可重构密码处理元素 RCE。RCE 可根据配置指令进行重构，灵活完成不同算法所需的运算功能。经 FPGA 验证以及 ASIC 的综合结果表明 RCE 在设计上不仅具有较优的性能还具有较高的灵活性，满足了绝大多数分组密码算法的需求。

论文分析了多种常见的分组密码算法在可重构密码处理模型 RCPA 上的映射，并设计实现了一款基于 RCPA 模型的验证原型。该原型在 FPGA 上成功进行了模拟验证并完成了在 0.18μm CMOS 工艺标准单元库下逻辑综合以及布局布线工作。

实验结果表明，在 RCPA 验证原型上执行的分组密码算法都可达到较高的性能。对于大多数分组密码算法，其密码处理性能与通用高性能微处理器处理性能相比提高了 10~20 倍；与其它一些专用可重构密码处理结构处理性能相比提高了 1.1~5.1 倍。结果说明本文研究的 RCPA 模型既能保证分组密码算法应用的灵活性，又能够达到较高的性能。

关键词：密码处理，可重构，分组密码，可重构密码处理模型，可重构密码处理元素，算法映射

Abstract

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 efficiency 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:

This paper has analysed the operation and processing structure characteristics of popular block cipher algorithms, and proposed a reconfigurable cipher processing architecture (RCPA) combining the design method of reconfigurable processing architecture. RCPA is a reconfigurable cipher processing architecture with coarse-grained, mixed interconnection, VLIW/EPIC computation, static and dynamic reconfiguration mode.

This paper has researched reconfigurable technology of the basic element in the block cipher processing and designed the Reconfigurable Cipher processing Element (RCE) in the RCPA. These RCEs can be reconfigured according to the configuration instruction, and achieved the function which different crypto algorithms needed flexibly. The RCEs have been verified via Altera FPGA and synthesized under 0.18μm CMOS technology. The result demonstrated that the RCEs can achieve both high agility and high performance, therefore it can satisfy the most of block cipher algorithms demands.

This paper has analysed the mapping of many mainstream block cipher algorithms on RCPA. We have implemented a prototype based on RCPA, and the prototype is realized using Altera's FPGA. Synthesis, place and route of RCPA have accomplished under 0.18μm CMOS technology.

The experiment results indicate that on the prototype based on RCPA, the performance of many block ciphers is 10~20 times higher than that of high-performance general purpose processor and 1.1~5.1 times higher than that of other specialized reconfigurable framework. The results prove that RCPA can guarantee high flexibility for most of block cipher algorithms and can achieve relatively high performance.

Keywords: Cipher Processing, Reconfigurable, Block Cipher, RCPA, RCE, Algorithm Mapping

原创性说明

本人声明所提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。尽我所知，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和撰写过的研究成果，也不包含为获得信息工程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标示谢意。

学位论文题目：面向分组密码处理的可重构设计技术研究

学位论文作者签名：[signature] 日期：2007 年 4 月 20 日

学位论文版权使用授权书

本人完全了解信息工程大学有关保留、使用学位论文的规定。本人授权信息工程大学可以保留并向国家有关部门或机构送交论文的复印件和电子文档，允许论文被查阅和借阅；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。

学位论文题目：面向分组密码处理的可重构设计技术研究

学位论文作者签名：[signature] 日期：2007 年 4 月 20 日

作者指导教师签名：[signature] 日期：2007 年 4 月 20 日

第一章 绪论

1.1 研究动机和思路

1.1.1 研究动机

随着各种信息网络的迅速普及和发展，信息技术已经渗透到政治、经济、军事和社会生活中，并在上述的领域发挥着越来越重要的角色。但人们在享有网络和信息技术带来的巨大便利地同时，如果不对网上传输的重要信息加以保护的话，这些信息很容易被泄露、篡改和假冒，从而给国家和社会造成重大损失，甚至危害国家安全。为了解决信息的安全性问题，人们采用了许多措施，其中，对信息数据进行加密就是一个行之有效的措施。

传统的数据加密方式有两类：基于通用微处理器方式和专用密码芯片（ASIC）方式。基于通用微处理器方式又称软件加密方式，即在通用计算机上完成数据加密/解密操作；基于专用密码芯片方式，即完全依靠专用硬件实现某种特定的加密/解密算法。上述的两种方式具有各自的特点，通用微处理器方式应用范围广泛，其完备的指令集使得它可以执行任何加解密计算任务，具有很大的灵活性，但由于密码算法与硬件电路不能很好的匹配，因此其加解密运算速度不高；专用密码芯片是针对于专门的密码算法而设计的专用硬件电路。因此其加解密速度快、性能高、功耗低、面积小。但灵活性差，应用范围窄，难以适用于其他密码算法。而且两种加密方式都有安全上的隐患。通用处理器加密方式，易受到各种攻击，算法的安全和密钥的储存均存在着安全隐患。专用密码芯片由于它们所依赖的硬件体系结构是刚性的，实现的算法是不可变的，然而任何算法都有被攻破的可能，因此专用密码芯片难以长期保证信息的安全性。此外专用密码芯片设计的过程较为复杂，算法可能在设计阶段和生产阶段存在着泄露的安全隐患。

针对上述的两种加密方式的优缺点，我们试图研究一种创新性的密码芯片，它既能够克服专用密码芯片只能实现特定密码算法的弊端，又解决通用微处理器加密方式速度慢的缺点，在密码算法实现的性能和灵活性之间获得较好折衷。该芯片可以在比微处理器具备更高效率的同时，取得比专用密码芯片更高的灵活性。更进一步的，当芯片的硬件结构与密码应用匹配得较好，且拥有足够的处理并行度的时候，它可以获得与微处理器相比更高的吞吐率和更低的功耗。此外，还可以很好的解决安全上的隐患。

1.1.2 研究思路

经过对上述的两种传统加密方式优缺点的分析，可以看出通用微处理器方式的加密速度慢、性能低，专用密码芯片由于实现的密码算法是确定的且不可更改，因此难以满足不同密码用户多层次的安全性需要和密码算法不断升级换代的需求。要使密码系统能够灵活，快速地实现多种不同的密码算法，则该密码系统的硬件结构必须是可变的，或者说有一定程度的柔性，从而可以与不同的密码算法相匹配[1]。因此，如何在密码芯片的灵活性

与高效性之间取得平衡，已经成为 VLSI 系统的设计者需要解决的基本问题之一。

为解决这一问题我们采用了可重构系统的设计思想。所谓可重构系统，是指由一个可重构的硬件处理单元和一个软件可编程的处理器结合在一起所构成的计算系统[1]。可重构系统允许用户改变可重构处理单元的结构与功能，以适应不同的应用需求。由此可见，可重构系统具有体系结构可变的特征，因此，可以借鉴可重构系统的设计思想来设计密码系统，使密码系统的硬件逻辑电路能够根据不同的密码算法的需求，重新组织，构成不同的电路结构，实现不同的功能，从而匹配不同的算法。称这样的密码系统为可重构密码系统[1]。可重构密码芯片就是采用可重构密码系统的实现方法来设计的密码芯片。

1.2 可重构密码处理

1.2.1 可重构计算的概念及其特征

可重构计算是近些年发展起来的一个新的计算研究领域，它以可重构硬件为基础，打破了传统软件和硬件的界限，将软硬件融为一体，同时满足了对性能和灵活性的要求。“可重构”这个概念的外延十分广泛，可以应用于很多场合。本文涉及的“可重构计算”是指对结构固定的硬件计算部件根据某种特定应用的计算需求进行配置，并在辅助设备(包括外围控制硬件和软件)的协同下完成相应的计算任务。可重构计算通常是在建立一个通用的计算部件上，能够用传统的编程方法完成数据加强型的应用任务。可重构计算系统在诸如多媒体图像处理、数据加密、仿真、滤波等许多面向计算的应用方面深具潜力。

可重构系统的计算功能部件通常由可编程逻辑电路实现，使它在体系结构上介于通用处理器(General Purpose Processor, GPP)和 ASIC 之间(如图 1.1 所示)。传统的处理器采用通用而固定的结构即空间维上的固定不变，利用时间维上的可变性通过编程来完成特定的计算。ASIC 则在时间维不可变，而使用定制的功能单元(如加法器、乘法器等)利用空间维的展开来实现计算任务，使原始数据顺序通过逻辑电路而得到相应处理，得到最后的计算结果。可重构计算综合了二者的优点，在空间维和时间维上均可变，通过对可重构硬件进行配置，从而使之由一个通用的计算平台转化为一个专用的硬件系统，以完成具体的计算任务，兼顾灵活性和计算高性能。这种有益的折衷使得可重构计算成为了继通用微处理器、ASIC 之后的第三个研究热点[2,3,4]。

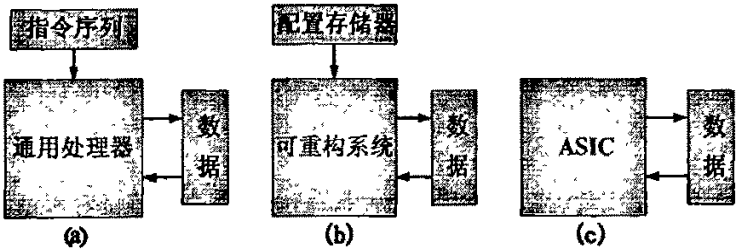


图 1.1 可重构系统与通用计算系统和专用计算系统之间的比较

在可重构计算系统的研究中，国内外的大学和科研机构研制了较多的可重构体系结构，其中有一些可重构计算系统得到较为广泛的应用如 FPGA(Field Programmable Gate

Arrays)、Chameleon 等。这些可重构计算的体系结构主要有以下特征：

1. 可重构计算系统的体系结构主要由灵活的可重构处理单元和可变的互连网络来实现。

2. 可重构处理单元具有确定的功能集，并有粗粒度和细粒度的区别，每个处理单元处理粒度应适合应用需求。

3. 互连网络根据可重构处理单元的分布状况(如线性、网状、多向网状等)可以有全局/分段总线、局部连线、NN 连线等多种选择。这些互连网络形成的拓扑通常体现出了一定的规则性。

4. 可重构计算系统大多是作为通用处理器的加速部件或协处理器来完成一些数据密集型的计算任务。因此，可重构计算系统大多是作为通用处理器计算功能的补充而存在的，完成面向特定应用的工作。

5. 细粒度和粗粒度可重构计算系统各有特点。细粒度结构适于实现位级逻辑，重构性能最好，可以满足大多数运算的要求，但对常用的的算术和逻辑操作时并不是高效的。粗粒度可重构结构根据应用的需求采用适当的粒度来完成计算任务，从而达到高性能。在粒度的选择上一般采用 2^n 位(如 2, 4, 8 等)作为可重构结构的粒度，因为这符合机器计算的字长特点而有利于大数据操作位宽的链接。

1.2.2 可重构密码处理研究现状

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处理优化，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，但存在灵活性差，不利于各部门、各区域之间秘密信息的互通。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[5]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。三类密码处理系统之间计算效率及灵活性的比较如图 1.2 所示。

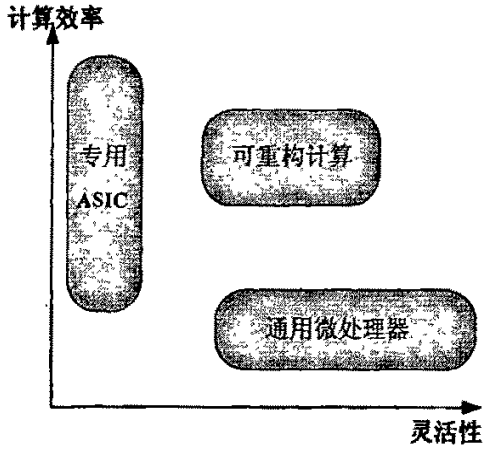


图 1.2 三类密码处理系统之间计算效率及灵活性的比较

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[6,7,8,9,10]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的协处理器。PipeRench 使用了硬件虚拟化和虚拟流水线的概念，流水线可动态可重构和部分动态可重构。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。是一个针对于流水线应用的加速器，采用了部分快速动态流水线重构和运行时对配置流和数据流的调度。

MorphoSys[11,12,13]可重构系统 1999 年由 Califorlia 大学提出，是一个粗粒度的可重构 SOC 系统。该系统具有 16 bit 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) [14,15,16,17,18,19,20]是由华盛顿大学 Carl Ebeling 研究小组人员首先提出并设计的。PaPiD 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。PaPiD 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 PaPiD-Ⅰ，PaPiD-Benchmark 和 PaPiD-Ⅱ。

COBRA(Cryptographic(Optimized for Block Ciphers) Reconfigurable Architecture) [21,22]

是由美国马萨诸赛大学的电子计算机工程系AJ Elbirt和德国波鸿大学的Christof Paar分析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[23,24,25]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(Reconfigurable Multimedia Array Coprocessor)。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-II RISC 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。北京科技大学研制了 RELOG_DIGG 系统可重构密码结构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，不少的可重构计算体系结构主要面向多媒体处理、无线通讯等领域的应用并不专门针对密码处理的，只对某些算法具有较好的映射。可重构单元的增强型处理器和通用密码处理器[26]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，本论文结合可重构体系结构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。

1.2.3 可重构密码处理研究意义

密码处理芯片是构建信息安全系统的基础和核心部件。目前在我国，军用以及商用密码芯片大多是采用 ASIC 设计技术设计的专用密码芯片。这类芯片具有速度高但灵活性差的特点，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[28]。

- 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IPSec 允许的算法就有：DES、3DES、Blowfish、CAST、IDEA、RC4、RC6、Diffie-Hellman、ECDH 等[28]。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实现的算法，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运算子密钥来参与计算，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[28]。

- 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[27]。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，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 研究内容以及创新之处

1.3.1 研究内容

本课题根据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征，研究了一种面向分组密码应用的可重构处理结构模型以及可重构密码运算模块的实现原理。

本文涉及的具体的研究内容有以下几个方面：

[1] 对现有的主要分组密码算法结构特点进行研究与分析，主要包括了分组密码算法的操作特征以及基本运算单元的构成。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；分组密码算法可由 7 类基本运算元素

运算构成。

[2] 对当前可重构密码处理现状进行了分析，重点研究了可重构系统结构的设计方法和实现，总结了可重构设计中的重构粒度、互连网络、计算模型、编程深度以及配置策略等关键问题。

[3] 根据分组密码的处理结构的特点，结合可重构处理结构的设计方法研究了一种可重构密码处理结构模型 RCPA。该模型的设计特点是粗粒度、混合型互连网络、类 VLIW/EPIC 计算模型、动态与静态配置相结合的可重构处理结构。通过对 RCPA 特征参数取值定义，对模型进行性能分析。

[4] 在可重构密码处理模型 RCPA 基础上，对可重构密码处理元素 RCE 进行了研究和设计。RCE 设计包括了移位单元、有限域乘法、模加/减、模乘、基本逻辑运算、S 盒替代和比特置换等密码处理元素。这些基本运算元素可根据 RCE 配置指令进行重构，灵活完成不同算法所需的运算功能。

[5] 在 FPGA 上进行综合及验证可重构密码处理元素 RCE 设计的正确性并且进行 ASIC 移植工作，给出基于 ASIC 的综合结果以及对其性能进行了分析。

[6] 对可重构密码处理模型 RCPA 进行了算法映射验证以及对其映射性能进行了分析与评估。最后基于 RCPA 模型，采用 Verilog 硬件描述语言编码，完成了一种验证原型的设计。

1.3.2 主要创新点

[1] 提出了一种可重构密码处理结构模型 RCPA。该模型是根据可重构处理结构的设计方法结合分组密码的处理结构的特点而构造。模型中包括可重构密码处理单元模块 RCU、互连模块 ICM、存储模块 MAM 以及配置控制模块 CCM。可以根据算法的不同实现可变并行度的流水处理并且流水线的深度可以配置。RCPA 采用静态和动态混合重构方式，支持多配置文件，从横向和纵向两个方向配置各个密码处理单元。通过对 RCPA 定义特征参数便可得到具有不同规模和适用性的可重构密码处理结构。

[2] 研究并设计了可重构密码处理元素 RCE。主要包括：采用桶形移位器的设计原理，在实现 8 比特位宽移位的基础上，可扩展完成 16 比特、32 比特以及 64 比特位宽的任意移位；基于“Shift-and-add”算法实现了 GF(2^8)域上的任意不可约多项式和乘数多项式的有限域 GF(2^8)乘法；以最小的操作粒度为 8 比特的加法器，通过对进位级联链的控制来实现模 2^8、2^{16}、2^{32} 的加/减操作以及 PHT 变换；以 16 比特乘法器为基本乘法器扩展实现了模 $2^{16}+1$、2^{32}、$2^{32}-1$ 乘法；使用了并行查找方案，利用 RAM 完成了可配置 S 盒替代电路的设计，支持了 8×8、8×32、6×4、4×4 等不同类型的 S 盒；利用 BENES 网络结构模型设计了置换单元，实现了 32-32、64-64、128-128 的任意比特置换。

[3] 研究并实现了一款基于 RCPA 模型的验证原型。该原型具有粗粒度、混合型互连网络、类 VLIW/EPIC 计算模型、动态与静态配置相结合的设计特点。可实现多种常用的

分组密码算法。

1.4 论文安排

第一章为绪论。介绍了本文研究的动机、思路及意义，介绍了可重构密码处理结构的研究背景和该领域的研究现状，最后阐述了研究的主要内容及主要创新点。

第二章深入分析了分组密码算法的处理结构特点和基本操作特征，为本文的可重构密码处理模型的设计奠定了基础。

第三章深入研究了面向分组密码处理的可重构结构的设计特点和方法，提出了可重构密码处理模型 RCPA。详细描述了该模型的结构和功能组成，并对 RCPA 的计算模型、编程深度、配置策略等关键问题进行了讨论。最后通过对 RCPA 特征参数的定义对模型进行了性能分析。

第四章研究可重构密码处理元素 RCE 硬件实现原理。介绍了移位单元、有限域 $GF(2^8)$ 乘法单元、模运算单元、逻辑运算单元、S 盒替代单元以及置换单元可重构设计方法。

第五章对各可重构密码处理元素 RCE 用硬件描述语言实现，并在 FPGA 上进行验证以及性能测试。此外还给出了 RCE 基于 ASIC 综合的结果并进一步分析了 RCE 的实现性能。

第六章讨论分组密码算法在 RCPA 上的映射，并对其性能进行分析与评估。最后给出了基于 RCPA 模型的一种验证原型的实现结果。

第七章为总结与展望。对研究工作进行了总结，阐述了本课题所做的主要工作，并指出了研究工作的不足之处和展望了该研究领域进一步的发展前景。

第二章 分组密码算法处理结构分析

现代密码算法主要分为两大类：对称密码和非对称密码，对称密码根据加解密操作位数的不同分为分组密码和序列密码两类；非对称密码加解密密钥不同，又称公钥密码。分组密码和公钥密码是现代安全系统中使用最多的两类算法，很多主流分组密码和公钥密码算法是公开的，为本论文的研究提供了基础。本章针对目前常用的公开分组密码算法进行分析总结。

2.1 分组密码算法简介

分组密码(Block Cipher) 又称秘密钥密码(Secret Key Cipher)是用于数据加解密的主要算法。利用分组密码对明文进行加密时，首先需要对明文进行分组，每组的长度都相同，然后对每组明文分别加密得到等长的密文。分组密码在设计上的特点是加密密钥与解密密钥相同。分组密码的安全性应该主要依赖于密钥，而不依赖于对加密算法和解密算法的保密。因此，分组密码的加密和解密算法是可以公开的[28]。

分组密码具有速度快、易于标准化和便于软硬件实现等特点，通常是信息与网络安全中实现数据加密、数字签名、认证及密钥管理的核心体制，可以用于重点信息的加密。使用分组密码容易实现同步，因为一个密文组的传输错误不会影响其他组，丢失一个明密文不会对其后续组解密的正确性产生影响。

分组密码通常具有比较整齐的数据位宽，需要大量重复的操作，执行速度比较快，很适合硬件实现，在密码领域的使用频度最大。分组密码的加解密算法结构非常规整，加密和解密过程的计算结构相同，只是某些对应操作和使用的常数、S 盒等略有不同。分组密码的加解密处理过程通常分为三部分：初始变换、中间变换和末尾变换。初始变换和末尾变换只执行一次，中间变换反复执行多轮，每轮的计算结构大体相同，加解密过程利用密钥扩展得到的子密钥对明文或密文进行一系列算术、逻辑、置换及代替等操作，密码算法的安全性主要依靠中间变换的强度来体现。很多情况下，密码算法的加解密过程要使用大于密钥量的子密钥参与处理，子密钥是由密钥扩展形成的，有些算法的子密钥就等于算法密钥，有些算法的密钥扩展过程比较简单，但有些算法密钥扩展需要的计算量很大。需要复杂计算的密钥扩展过程通常会使用和密码算法相同或相似的运算类型及操作模块[28]。

2.2 分组密码算法的一般结构

2.2.1 分组密码算法的数学模型

任何可计算的算法都具有基本的数学模型，其算法的描述也都可以用数学符号来表述。分组密码算法的数学模型可抽象为明文数据，密钥，加密过程，密文数据，解密过程。

其过程为：将明文信息编码形成的序列分成等长的分组，即输入明文数据或称待加密数据，同时输入密钥序列，在加密密钥的控制下通过加密算法变换成等长的输出密文数据

(加密过程)。加密过程生成的密文数据，经过存储或传输，再输入解密密钥序列。在解密密钥的控制下通过解密算法变换成等长的输出明文数据(解密过程)。

分组密码是将明文消息编码后的数字(通常是 0 与 1)序列 x_1，x_2，… 划分成长为 m 的组 $x=(x_1, x_2, \cdots, x_m)$，每个明文组(长为 m 的向量)分别在密钥 $k=(k_1,k_2,\ldots,k_t)$的控制下变换成等长的输出数字序列 $y=(y_1, y_2, \cdots, y_n)$(长为 n 的向量)，整个运行过程的的数学模型如图 2.1 所示[29]。

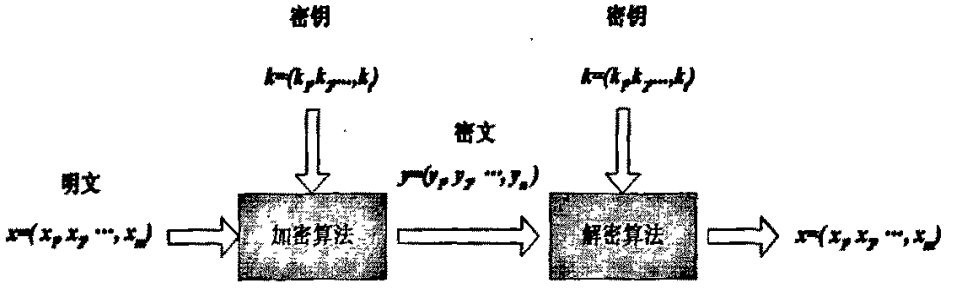


图 2.1 分组密码加解密模型

上图中，若 $n>m$，则为有数据扩展的分组密码；若 $n<m$，则为有数据压缩的分组密码；若 $n=m$，则为无数据扩展和压缩的分组密码，通常研究的均为这种情况。本文设明文 x 和密文 y 均为二元(0 与 1)序列。设 F_2 是二元域，F_2^s 表示 F_2 上的 s 维向量空间。假定明文空间和密文空间均为 F_2^m，密钥空间 S_k 是 F_2^t 的一个子集合。m 是明文和密文的分组长度，t 是密钥的长度。

2.2.2 分组密码的整体结构特征

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，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。

一、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：

$$L_i=R_{i-1} \ ; \ R_i=L_{i-1} \oplus f(R_{i-1},K_i)$$

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：

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

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[29]。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。

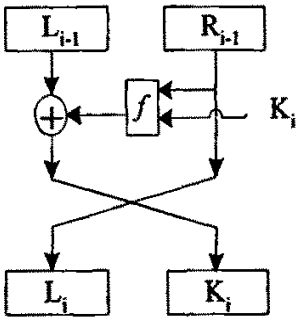


图 2.2 Feistel 网络型结构

二、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。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（Advanced Encryption Standard）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[60]。

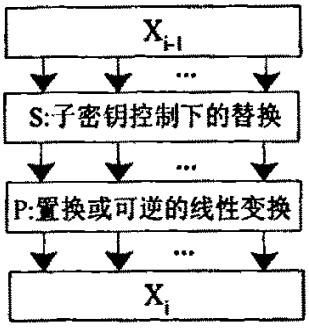


图 2.3 SP 网络型结构

三、LM 网络结构特征分析

从操作特性上来看，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[30,31]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 S1、S2、S3 和 S4 用 T1、T2、T3 和 T4 代替时，图 2.4 中所示的 MA 结构的输入不变，当 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 算法的实现效率较低，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。

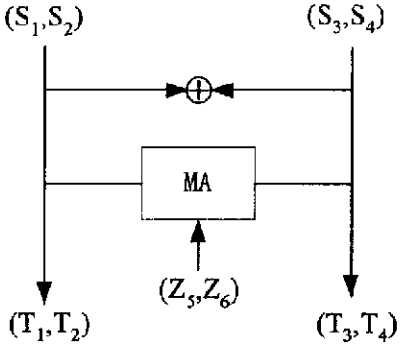


图 2.4 LM 型结构中的轮函数

2.3 分组密码基本操作分析

2.3.1 分组密码的基本操作总结

通过上一节对分组密码结构的分析可以看出，目前大多数分组密码算法的设计都基于一些相似的设计理论和结构模型，因此分组密码算法可分为：基于 Feistel 网络或扩展 Feistel 网络结构的分组密码算法，典型算法包括 DES、Lucifer、FEAL、Khufu、Khafre、LOKI91、LOKI97、GOST、CAST、CAST-128、CAST-256、Blowfish、Twofish、RC5 、RC6 等；基于 SP 网络结构，包括 CRYPTON、SAFER+、SHARK 、Rijndael 及 SERPENT 等；基于 LM 结构的不同代数群的混合运算，典型的例子为 IDEA 和 MMB 等。基于相同或相似设计理论的分组密码处理结构也很相似，涉及的操作类型有较大的交集[28]。因此我们可以得出这样的结论：很多不同的分组密码算法具有相同或相似的基本操作成分，或者说同一基本操作成分在不同的算法中出现的频率很高。如：异或、移位、置换、S 盒代替、模乘以及模加 / 减运算等。

表 2.1 基本运算操作使用频度表

基本运算操作	使用频率	基本运算操作	使用频率
逻辑运算	97.56%	模减运算	48.78%
S 盒变换	73.17%	置换运算	24.39%
移位运算	85.36%	有限域乘法运算	17.07%
模加运算	48.78%	模乘运算	17.07%

表 2.1 列出对 DES、IDEA、AES 候选算法等 41 种公开的分组密码算法的统计结果。算法中涉及到的基本操作使用频率统计[21,27]如表中所示。

表 2.2 归纳了当前主要的分组密码算法的分组位宽大小、密钥位宽大小、相关的基本操作以及其主要操作的运算宽度。

表 2.2 分组密码算法中的基本操作及其运算位宽

算法	模加/减	模乘	移位	置换	S 盒替代	逻辑运算	有限域乘法	分组/密钥长度	运算宽度
DES/ 3DES			*28(L)	64-64 32-48 48-32 *64-56 *56-48	6×4	32 xor 48 xor		64/64	32
IDEA	2^{16}	$2^{16}+1$	*128(L)			16 xor		64/128	16
MMB		$2^{32}-1$				32 xor		128/128	32
TEA	2^{32}		32（L）			32 xor		64/128	32
3-WAY			32（L）	96-96	3×3	32 xor		96/96	32
Blowfish	2^{32}				8×32	32 xor		64/64	32
RC5	2^{32}		32（L）			32 xor		64+/64+	32
CAST	2^{32}				8×32	32 xor		64/64	32
CAST256	2^{32}		32（L）		8×32	32 xor		128/128+	32
Khufu			32（L）		8×32	32 xor		64/64+	32
Khafre			32（L）		8×32	32 xor		64/64+	32
FEAL	2^{8}		*8（L）			8 xor 32 xor		64/64	32
GOST	2^{32}		32（L）		4×4	32 xor		64/256	32
CRYPTON			32	128-128	8×8	8 xor		128/128+	32
NSSU	2^{32}		32（L）		4×4	32 xor		64/256	32
LOKI91			32（L）	32-48 32-32	12×8	32 xor		64/64	32
LOKI97	2^{64}			64-64	13×8	64 xor		128/128+	64
SAFER+	2^{8}		*8（L）		8×8	8 xor		128/128+	8

Rijndael			32		8×8	32 xor	$GF(2^8)$	128/128+	32
RC6	2^{32}	2^{32}	32 (L)			32 xor		128/128+	32
Mars	2^{32}	2^{32}	32		8×32	32 xor 32 and		128/128+	32
Twofish	2^{32}		32		8×8	32 xor	$GF(2^8)$	128/128+	32
SERPENT			32 (L)	128-128	4×4	32 xor		128/128+	32
Square			32		8×8	32 xor	$GF(2^8)$	128/128	32
E2	2^{64}	2^{32}		64-64	8×8	8 xor		128/128+	16
SHARK	2^{64}				8×8		$GF(2^8)$	64/128	8

注：1. * 表示只在密钥产生过程中使用的运算单元。

2. + 表示密钥位宽取 128-256 位的一些值。

3. 移位后面的括号表示移位的方向，没有括号表示算法中左右移位都存在。

通过以上两个表对多种分组密码算法的分组位宽、密钥位宽、基本操作种类、操作频度以及运算位宽的分析可以得出分组密码的操作特点为：

(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[5]。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[28]；LM 结构的代数群运算类算法更着重于算术运算，主要运算为模乘、模加、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、模加运算、置换运算、模乘运算。

(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 1 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，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并行查找相同或不同的 S 盒。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。表 2.2 列出了几种不同的 S 盒代替的模式常见的有 4-4 S 盒、6-4 S 盒、8-8 S 盒、8-32 S 盒，目前以 8-8 S 盒最为流行；S 盒的大小从 512b-80kb 不等[28]。

(5) 大数据位宽才能有效扩展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[28]。128 比特的置换在分组密码设计中也较为常见。表 2.2 所列出的置换操作指的是位级置换，即单个数据位在一定范围内位置交换的操作。还有很多算法用到

了字节或更大粒度的置换操作，如 AES 的行移位就可以看作是 128 位数据的字节置换操作，3-WAY, SAFER+等算法使用更复杂的字节置换网络[28]。

2.3.2 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)[32]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[28]。

(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[28]。

(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[5]，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以及基本操作特征进行研究与分析，主要包括了分组密码算法的整体操作结构特征以及基本运算单元的构成。

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；总结出了分组密码算法可由 7 类基本运算元素构成，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元素在常见分组密码算法中的使用情况。为本文的可重构密码处理模型的研究与设计提供了依据，奠定了理论基础。

第三章 面向分组密码的可重构处理模型的研究与设计

从上述的章节可以看出，当前可重构结构模型无论在结构还是在具体操作支持上，其出发点都是要匹配不同的应用需求，众多研究中以多媒体应用居多，有些结构和控制机制不匹配密码算法处理特点。在这一领域所提出的众多模型中，没有专门针对分组密码算法处理模型的研究，本章将根据分组密码处理的特点，提出了可重构密码处理模型(RCPA: Reconfigurable Cipher Processing Architecture)，并给出模型中具体模块的构成和配置控制策略。该模型架构支持了分组密码算法的分组内的横向并行处理和多分组间的纵向并行或流水处理两种处理模式。

3.1 可重构分组密码处理模型设计的关键问题

在设计可重构分组密码处理模型时，必须充分考虑分组密码处理结构的特征，同时需要结合可重构体系结构设计要素。因此，本节将结合第二章对分组密码处理结构特点的研究，来讨论可重构分组密码处理模型设计中的重构粒度、互连网络的结构、计算模型选择以及配置策略等几个关键问题[33]。

3.1.1 重构粒度

粒度是可重构分组密码处理结构设计中首要考虑的因素之一，也是结构设计最重要的因素之一。粒度指一个系统可重构组件或可重构处理单元（RCU）操作数据的大小[33]。处理单元的粒度分为细粒度，粗粒度，混合粒度等。

在细粒度系统中，可重构处理单元包括典型的逻辑门，触发器，查找表 LUT 等。它们按“位”操作，完成有限状态机的布尔功能。细粒度可重构阵列通常是基于 FPGA 的系统。FPGA 结构具有细粒度的可编程逻辑块和可编程互连网络，为系统提供了很大的灵活性，但细粒度可重构结构在完成计算任务时有如下缺点[23,32]：首先，硅片利用率低，功耗大。其次配置数据量大，配置编译时间长。最后，在逻辑层编程，开发难度相对较高。

在粗粒度系统中，可重构处理单元包括完整的功能单元，如完成“字”级以上的数据操作的 ALUs 或乘法器，更适合完成字大小数据的规整算术运算。在数据加密和多媒体应用中经常存在这种运算。粗粒度可重构系统的例子如 MorphoSys (16 位数据路径)、RaPID(16 位数据路径)、REMARC (16 位数据路径)[34]和 COBRA (32 位数据路径)。粗粒度可重构结构具有多位宽数据路径和复杂的操作算子能有效的解决基于细粒度结构方案的缺点。

当在实际应用中既包括细粒度操作又包括粗粒度操作时，选择粒度通常需要采取折中。而混合粒度单元既可完成位级操作，又可配置完成字级操作。如 PipeRench，RAW[35,36]等。但该结构设计复杂，这增加了应用算法映射的复杂度。

通过对分组密码算法的研究，可以发现，分组密码的基本计算粒度除了位级置换和部分算法的 S 盒操作外，算法操作的位宽都是字节的整数倍。因此，粗粒度可重构密码处

理结构更能匹配密码算法的处理。采用较粗粒度的单元将避免大量布线及较低的布线效率等问题，减小配置存储容量，减轻结构的配置压力，使可重构密码处理结构更易采用静态和动态配置技术。但同时，粗粒度单元上实现比其粒度小的操作会浪费计算资源，从而也会浪费一定量的布线。可重构密码处理结构的基本处理粒度是固定的，大粒度操作要被拆分成基本粒度进行处理，因此选择合适的密码处理粒度是首要准则。由于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，其中 32 位位宽占大多数。在可重构密码处理模型 RCPA 设计上又需要针对分组密码算法采取并行处理。因此，处理数据粒度设为 32 比特可以满足大多数分组密码算法的需要。密码算法大多数操作的位宽比较规整，且有明显的倍数关系（1-4 个字节），对于计算位宽小于 32 比特的操作，可在可重构处理单元硬件设计上支持 8 位和 16 位多种位宽的处理操作。

3.1.2 互连网络

可重构分组密码处理结构不仅由可重构处理单元粒度决定，还和可重构处理单元的互连网络有关。基本的互连结构取决于可重构处理单元的全局排列方式。同时需要考虑系统针对的应用领域和完成的复杂性。很明显，互连结构的类型直接影响到应用映射的复杂性。目前，可重构计算阵列主要的结构主要有基于二维 Mesh 型阵列结构、线性阵列结构以及基于水平交叉开关的 Crossbar 型阵列结构。用 Crossbar 连接各可重构处理单元，这样可实现任意连接。在另一些结构中，可重构处理单元被排列成一个或多个线性阵列或者采用 Mesh 型和 Crossbar 型混合结构。

二维 Mesh 型结构将可重构处理单元按照二维阵列进行排列，最简单的互连方式为每个单元水平和垂直的与其四周的单元相连。可重构 Mesh 模型具有简单性、规则性、可扩展性等特点，是很多可重构结构采用的结构模型，但对于某些非常适合特定处理结构(如 Systoic 阵列)的应用，这种模型并不是最佳的。Mesh 模型结构典型应用是 REMARC、MorphoSys 等可重构系统。

水平交叉开关的 Crossbar 型阵列结构允许可重构处理单元间的任意连接，是最强大的通信网络。Crossbar 结构可以同时提供多个数据通路，一个 Crossbar 结构由 N×N 交叉矩阵构成。当交叉点(X,Y)闭合时，数据就从 X 输入端输出到 Y 输出端。理论上来说，可以实现一个全连通的纵横开关网络，该纵横开关网络由 N 条总线实现，每个输入端口可以连接任何一个输出端口。然而，特别是在输入输出端口个数增大的时候，全局总线和大量的开关使得该网络的功耗和面积大大增加[24]。目前这种结构的系统仅完成一个简化的水平连接。层次化水平连接的典型代表有 COBRA 系统和 PADDI 可重构系统[23,37]。

线性阵列结构主要针对流水线应用，部分的可重构体系结构中的可重构阵列，采用了基于一个或者多个线性阵列的排列方式。在此结构中，相邻可重构处理单元被连接为一个或多个线性阵列，线性阵列结构中的每个功能处理单元都可以作为流水线的一级。因此这种结构非常适合用来映射流水线算法。线性阵列结构的典型如 PipeRench 系统和 RaPiD

系统。

在分组密码算法处理过程中，分组通常划分为较小的子分组（子块）进行处理，分组密码中子分组的处理一般能够并行。并且密码算法常常要处理多个分组数据，在无反馈模式下，这些分组处理之间没有相关性，设置多个处理单元可以实现多个分组的流水操作；在反馈模式下，采用交替技术，也可以实现多个分组数据在多个处理单元上的流水操作。分组密码算法并行和流水处理结构如图 3.1 所示。该处理结构既提高了算法的执行效率又有效增大了运算吞吐率。

开发分组密码横向和纵向的并行性是设计可重构密码处理结构的目标，适度并行和深度流水结构是较好的选择。通过前面分析可以得出分组密码处理具有很多流式处理的特点，而线性阵列是映射流水结构的有效方式。分组密码处理常常体现对分的原则，即把一个分组分成计算粒度相等的几部分并行处理，这几部分处理的中间结果很可能存在数据相关性，需要交叉传递给下一级操作，很多算法的置换网络也体现着对分或交叉对分的规律性。互连的主要作用就是使对分操作的结果能够方便地进入下级流水线处理，支持可重构密码处理结构纵向的流水操作。根据以上的分析，可得到可重构分组密码处理系统的互连网络宜采用水平交叉开关 Crossbar 型结构和并行的线性阵列组成的混合结构。

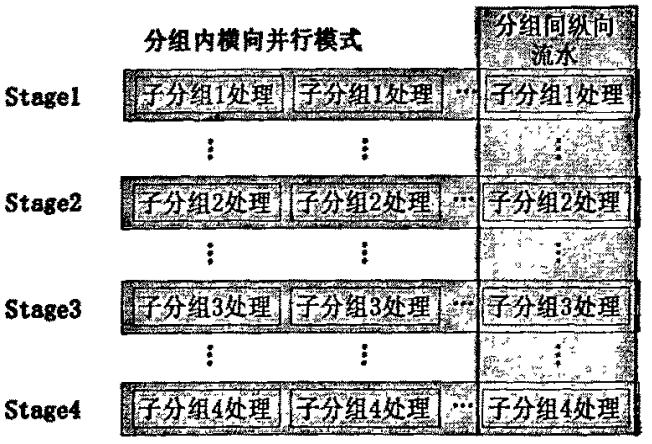


图 3.1 分组密码算法并行处理结构

3.1.3 计算模型

目前可重构处理的计算模型主要包括 SIMD (Single Instruction Multiple Data，单指令流多数据流)、MISD (Multiple Instruction Single Data Stream，多指令流单数据流)、VLIW (Very Long Instruction Word，超长指令字)/EPIC(Explicitly Parallel Instruction Computing，显式并行指令计算)等。

可重构处理 SIMD 计算模型对于数字信号处理中的低精度数据[33]以及并行化的处理是具有较好的支持。SIMD 操作在同一时间对多个不同可重构功能部件的数据执行相同的指令，这样利用了并行性且提高了吞吐量。然而，单纯的 SIMD 方式由一个控制部件为所有的可重构功能部件提供程序，所有可重构功能部件执行同样的指令，使得系统获得空

间上的并行性，但使灵活性变差，当程序只需一部分可重构功能部件操作时，所有的可重构功能部件都仍要进行相同的操作。这样，不但系统降低了利用率，而且增大了不必要的功耗。该计算模型的代表是 MorphoSys 可重构系统。

VLIW/EPIC 计算模型具有较长的机器指令字，机器指令字具有固定的格式(一种或者多种)，每条指令字中包含着多个独立的字段，字段中的操作码被送往不同的功能部件并行执行。采用 VLIW/EPIC 技术避免了 SIMD 方式的缺点，在引入并行化而加速计算执行的同时，增加了可重构处理的灵活性。这种结构硬件控制相对简单，计算密集型应用内在并行性很明显， VLIW/EPIC 计算结构要求高指令带宽，依赖编译处理，二进制兼容性差。PipeRench 以及 COBRA 等可重构系统均使用了此种计算模型。

MISD 计算结构是从 VLSI 并行结构设计中的脉动阵列（systolic arrays）结构而来，数据流水地逐一通过多个处理单元构成的阵列，在每个处理单元执行不同的指令流。脉动阵列结构能够非常好地匹配某些特定算法的处理结构，流水化程度高，非常适于 VLSI 实现，多用于数字信号处理。但该计算结构可编程性差，适用范围窄[5]。

分组密码算法处理结构具有两个方向的并行性使我们考虑了 VLIW/EPIC 计算模型的结构。如果采用单纯的 SIMD 方式，则使得所有可重构密码处理单元在同一时间对不同的数据执行相同的指令，这使得可重构密码处理结构的纵向并行或流水线设计难以实现，而且灵活性变差。系统降低了利用率，而且增大了不必要的功耗。由于 VLIW/EPIC 每条指令字中包含着多个独立的字段，字段中的配置码被送往不同的功能部件，从而避免了 SIMD 方式的缺点。可重构密码处理单元使用同一组配置寄存器，通过 VLIW/EPIC 方式可以为每个可重构密码处理单元选择不同的配置，很好的支持了密码算法适度并行和深度流水结构，灵活处理分组密码算法的应用。

3.1.4 编程深度与配置策略

编程深度是指存储在可重构处理单元中的配置程序或文件的数量。一个可重构处理单元可能含有单个配置文件或多个配置文件。对于单配置文件系统，只有一个配置文件驻留在可重构处理单元内，因此可重构处理单元的功能局限于当前装载的配置文件。而在多配置文件系统中，同时有多个配置文件驻留在可重构处理单元内，这使得可以通过切换配置文件的方法很方便地实现不同的功能，而不必从外部重新装载配置文件。

对可重构处理部件的配置一般分为静态重构和动态重构两种[27,33,38]。

1. 静态重构(Static Reconfiguration)：如果装载配置文件的过程必须在中断运算执行的情况下进行，这样的重构称为静态重构。即运行前一次加载完成之后，在运算过程中其结构不再改变。必须等到运行结束后才能进行下一次重构。而且在重构时，必须对可重构逻辑单元全部重新配置，直至新的配置数据加载完毕，才能开始执行新的操作。

2. 动态重构(Runtime Reconfiguration)：如果重构的过程可以和运算执行同时或交叉进行，这样的重构称为动态重构。即系统在运行时，可重构逻辑单元的功能可以实时进行

全部或部分重构。全部重构时，可重构逻辑单元全部重新配置；而部分重构时，只对需要改变的重构逻辑资源进行重新配置。静态重构和动态重构配置与执行过程如图 3.2 所示。

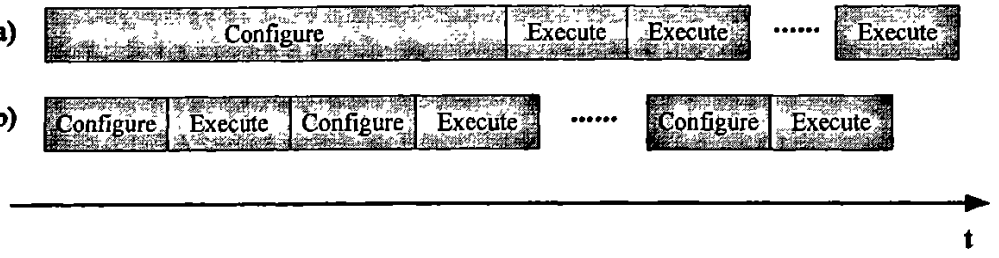


图 3.2 (a) 静态重构 (b) 动态重构

单配置文件的可重构系统通常具有静态重构的特性，而动态可重构系统大多都是基于多配置文件。动态可重构系统中的可重构逻辑单元可以在执行某个配置文件的同时，方便地通过切换来改变执行的配置文件。这种配置策略能够很大程度上减少重新系统重构的时间，增强了可重构系统的性能。

通过以上的分析，结合分组密码算法处理结构的特点以及降低指令系统设计难度和指令译码控制逻辑的复杂性。可重构密码处理结构应采用多配置文件系统，这可以通过在可重构密码处理结构内部设置配置 RAM 或寄存器堆来实现。同时有多个配置文件驻留可重构密码处理单元在内，这使得可以通过动态切换配置文件很方便地实现可重构密码处理单元不同的功能，而不必从外部重新装载配置文件。多配置文件系统提供了灵活性，当需要改变密码算法时，只需更改配置文件即可。在配置方式选择上采用静态重构和动态重构相结合的方式是较好的选择。在执行密码运算前，选择静态配置方式，将可重构密码处理单元的多个功能单元配置文件在运算前一次加载完成，执行过程中无需再重新加载。而在配置可重构密码处理结构的互连网络以及切换可重构密码处理单元静态配置文件时采用动态配置方式，可以达到动态重构密码处理单元以及动态配置流水线操作的目的，因此采用静态重构和动态重构相结合的重构方式既可以有效减少指令字的长度，增加指令字中所包含的有效操作的个数，又有效提高加 / 解密处理速度以及可重构系统的资源利用率和灵活性。

3.2 可重构密码处理模型设计

可重构密码处理结构模型的研究需要借鉴其它结构和模型研究成果。目前许多可重构计算结构模型主要面向多媒体处理、无线通讯等应用领域。有些模型虽然支持密码处理但其结构并不是密码算法的最佳匹配结构，只对个别的分组密码算法具有较好的加速，但支持的广泛性不够。结合上述的现状和存在的问题，我们根据前面研究的可重构结构设计技术结合分组密码处理结构的特点，在本节提出一种可重构密码处理模型 RCPA（Reconfigurable Cipher Processing Architecture），并将详细介绍 RCPA 的组成和各部件的功能。

3.2.1 可重构密码处理模型 RCPA 概述

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，我们提出了可重构密码处理模型（RCPA：Reconfigurable Cipher Processing Architecture），如图 3.3 所示。模型中包括可重构密码处理单元模块（RCU：Reconfigurable Cipher processing Unit），互连模块（ICM：Inter-Connection Module），存储模块（MAM：Memory Access Module）以及配置控制模块（CCM：Configuration and Control Module）。该模型的设计特点是粗粒度、Crossbar 和线性阵列混合型互连网络、类 VLIW/EPIC 计算模型、动态与静态配置相结合的可重构处理结构。RCPA 可根据算法的需要实现可变并行度的流水处理，在横向和纵向两个方向组织各个可重构密码处理单元。RCPA 具有的特征参数可根据应用的需求进行定义，便可得到具有不同规模和适用性的可重构密码处理结构。

配置控制模块 CCM 用来控制整个可重构密码处理模型 RCPA 的操作，主要功能是控制完成内部各个模块的配置操作，控制内部各个模块之间的数据传递以及与外部的数据交互；互连模块 ICM 的作用是完成各级 RCU 之间的数据交互操作以及存储模块与 RCU 之间的互连；存储模块 MAM 用来对加解密算法中的密钥、常数以及运算中的临时数据进行存储；可重构密码处理单元模块 RCU 负责对数据的运算处理，根据分组密码算法的特点运算数据宽度设为 32 比特能够匹配大多数算法的要求。

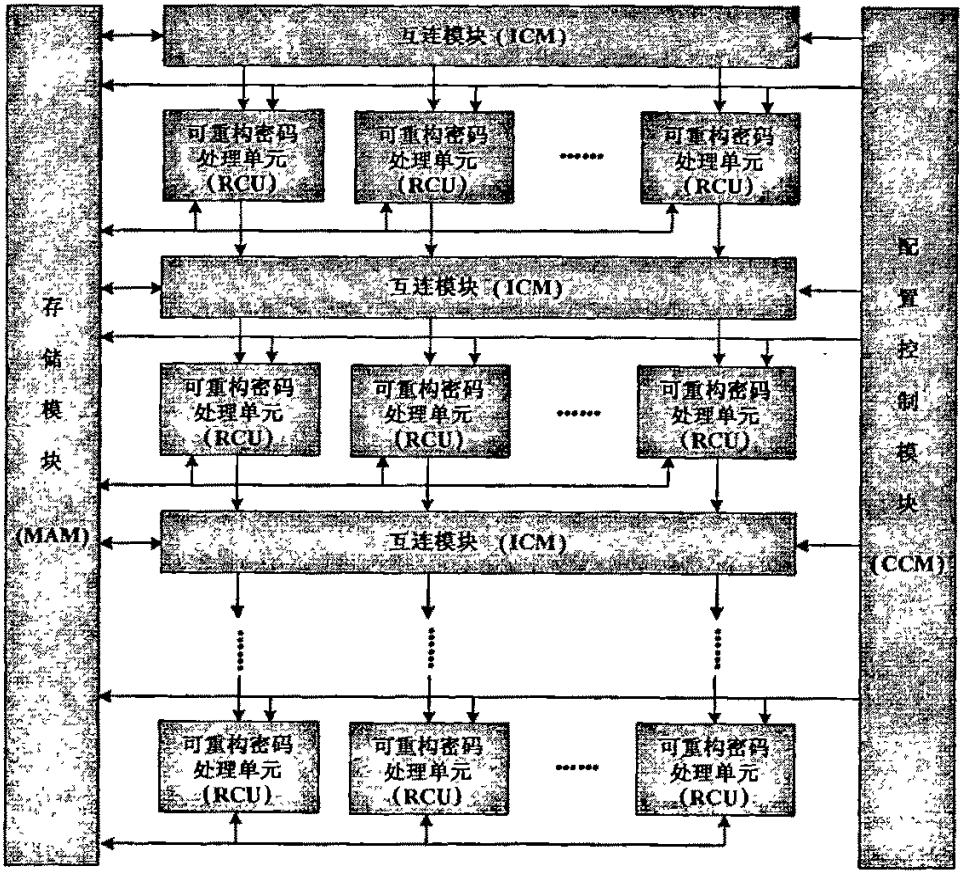


图 3.3 可重构密码处理模型 RCPA 架构图

在可重构密码处理模型 RCPA 中，外部输入数据在配置控制模块 CCM 控制下首先进入到存储模块中，然后再将其读出输入到互连模块 ICM 以及可重构密码处理单元 RCU 中进行运算。各级 RCU 的输入来自互连模块 ICM 和储存模块 MAM，输出则写入储存模块 MAM 或者直接进入下一级的互连模块 ICM。在执行算法前，配置控制模块 CCM 需要对部分可重构密码处理单元 RCU 或互连模块 ICM 进行静态重构；算法执行时，配置控制模块 CCM 再根据算法的要求对 RCU 和 ICM 注入部分配置信息进行动态重构，从而构成完整的算法硬件电路。为支持可重构密码处理模型 RCPA 的流水操作，可在各个 RCU 和 ICM 的输出端插入可配置的寄存器，配置控制模块 CCM 通过对其进行配置来决定整个 RCPA 运算处理流水线。此外配置控制模块 CCM 还可对无需参与运算的 RCU 和 ICM 进行旁路处理，以较小系统时延提高性能。

3.2.2 可重构密码处理单元 RCU

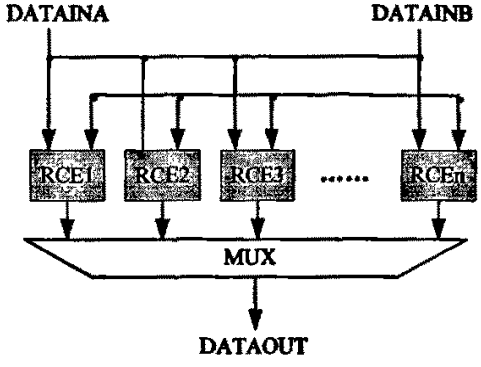
可重构密码处理单元 RCU 是 RCPA 的核心功能部件，RCU 在配置控制模块 CCM 控制下完成各种配置指令的执行。可重构处理单元 RCU 由可重构处理元素（RCE：Reconfigurable Cipher processing Element）阵列构成。

通过第二章对大量分组密码算法的特点的分析可以看出，很多不同的密码算法具有相同或相似的基本操作成分，或者说同一基本操作成分在不同的算法中出现的频率很高。这些基本的操作成分对应的硬件资源通过注入不同的配置信息就可以被不同的密码算法所共用。因此我们对这些密码基本操作成分进行可重构处理元素 RCE 设计，就能够以这些可重构处理元素 RCE 来构造一套逻辑电路来实现多种算法。经过对分组密码算法基本操作成分的分析可以得到可重构处理元素 RCE 包括了：基本逻辑运算元素、移位运算元素、比特置换元素、S 盒替代元素、模加 / 减法运算元素、模乘法运算元素、有限域乘法运算元素。根据分组密码运算的特点，可重构处理单元 RCU 处理的数据位宽设为 32 比特适合大多数密码算法的要求。相应可重构处理元素 RCE 处理位宽也同样为 32 比特，可支持 8, 16 位多种位宽的处理操作。可重构处理元素 RCE 的电路设计将在下一章详细说明。

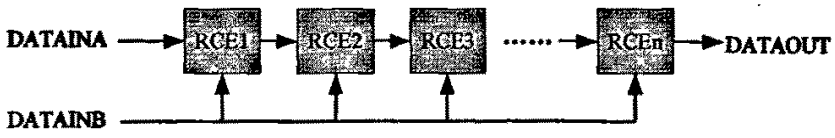
可重构处理单元 RCU 内部的可重构处理元素 RCE 连接方式可分为并行连接、串行连接和串并混合连接。三种连接结构如图 3.4 所示。在并行连接方式中，各可重构处理元素 RCE 可进行并行运算，通过数据选择器可将相应的 RCE 运算的结果输出。串行连接方式中，各可重构处理元素 RCE 按照一定的次序串行连接，对不需要执行运算的 RCE 进行旁路。并行连接方式设计简单易于实现，并且使 RCU 电路内部电路延迟和路径延迟最小，但每次只能执行一种的运算。串行连接方式在设计中要结合密码算法的处理特点，对算法的处理顺序进行分析并找出最佳的连接顺序。该连接方式使 RCU 每次可以依次执行多种 RCE 的运算，提高了执行效率。但串行方式设计使 RCU 电路内部电路延迟和路径延迟达到了最大，降低了整体的最高运算时钟频率。串并混合连接则是根据两种连接方式的优缺点，采取的一种折衷连接方式。在并连时，通过数据选择器选择所需的 RCE 运算的结果；

串连时则对不需要执行运算的 RCE 进行旁路。该方案在 RCU 电路内部既有较小的电路延迟和路径延迟，又具有较高的执行效率。

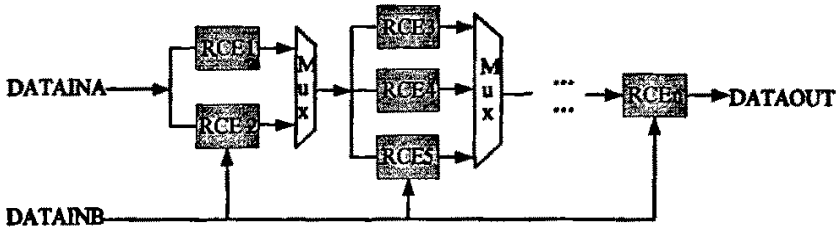
为了支持整个可重构密码处理模型 RCPA 可以进行流水处理操作，可重构处理单元 RCU 需要在数据输出端设置可配置寄存器。通过对该输出寄存器进行配置，可方便的实现 RCU 输出的运算结果是否需要寄存一级输出。从而可以根据需要对可重构密码处理模型 RCPA 的流水线级数进行配置。



(a) 可重构处理元素RCE并行连接方式



(b) 可重构处理元素RCE串行连接方式



(c) 可重构处理元素RCE串并混合连接方式

图 3.4 可重构处理元素 RCE 连接方式

3.2.3 互连模块 ICM

互连模块 ICM 主要用来实现各级 RCU 之间的数据传输以及存储模块与 RCU 之间的互连；根据前面的分析可知，互连模块 ICM 内部的交换网络宜采用水平交叉开关 Crossbar 型结构。Crossbar 结构可以同时提供多个数据通路，一个 Crossbar 结构由 N×N 交叉矩阵构成。当交叉点(X,Y)闭合时，数据就从 X 输入端输出到 Y 输出端。理论上来说，可以实现一个全连通的纵横开关网络，该纵横开关网络由 N 条总线实现，每个输入端口可以连接任何一个输出端口。

随着通信技术和计算机技术的迅速发展，交换网络的研究日益深入，出现了 BENES、CLOS、BUTTERFLY、BANYAN 等形式的交换网络[39]。这些交换网络通过多级互连网络

减少了 Crossbar 的开关个数，而且这种网络的硬件实现要比用 N 个 N 选 1 的交叉开关实现节省面积。例如 N×N 的 BENES 网络[40]是由 $2\log_2 N-1$ 级 2×2 的开关构成，每级具有 N/2 个开关，因此开关总数是 $N\log_2 N-N/2$。通过改变各级节点开关的状态互连模块 ICM 网络的复杂度大得多，因为它们必须支持更为复杂的开关类型。另一方面，多级网络的时延也要更大一些。图 3.5 表示了一个 8×8 BENES 内部开关网络结构。

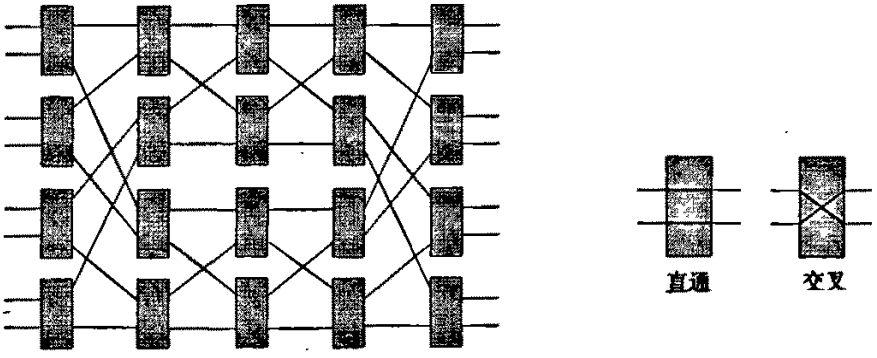


图 3.5 8×8 BENES 网络结构

由于 Crossbar 交换网络和密码学中的置换网络功能非常相似，因此可以用 Crossbar 交换网络实现分组密码算法中的大数据位宽的比特置换操作。经过对分组密码的处理结构特点分析得出，大数据位宽的比特置换操作，例如 64×64 比特置换，128×128 比特置换通常用在算法的初始变换和末变换中。因此在 RCPA 模型中存储模块与第一级 RCU 之间设置了互连模块 ICM，用来完成大数据位宽的初始置换或末置换。

由于分组密码处理常常体现对分的原则，即把一个分组分成计算粒度相等的几部分并行处理，这几部分处理的中间结果很可能存在数据相关性，需要交叉传递给下一级操作。因此在各级 RCU 之间的 Crossbar 交换网络互连模块 ICM 即 Crossbar 交换网络用来完成级间的数据传递。进一步分析可以得出各级 RCU 之间的交叉传递是以字节为单位的交叉传递，所以在设计上各级 RCU 之间的 Crossbar 交换网络是以字节为基本单位的字节置换网络。例如，在 DES 算法中 Crossbar 交换网络是以 4 个字节（32 比特）与下一级进行数据交叉传递；而在 Rijndael 算法中利用 Crossbar 提供的字节置换网络就可方便实现 128 位数据的行移位变换。

综上所述，互连模块 ICM 在设计上必须能够灵活地支持所实现算法的互连要求，同时必须减小功耗和面积的开销。因此 Crossbar 交换网络宜采用 BENES、Omega-flip 以及 LPS 型交换网络结构。这些交换网络实质上等同于置换网络，不仅可以完成数据的交换也可以完成算法中的置换操作。存储模块与第一级 RCU 的互连网络在设计上可采用比特置换网络，以满足算法中的初始比特置换或末比特置换；而各级 RCU 之间的互连网络则采用以字节为单位的置换网络。这样，互连模块 ICM 的开关数以及级数会有效的减少，既降低了硬件资源的开销又提高了互连网络的性能。

3.2.4 存储模块 MAM

存储模块 MAM 是可重构密码处理模型 RCPA 的重要部件，负责对加解密算法中的密钥、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 MAM 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 MAM 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。表 3.1 列出了一些主要的分组密码算法中所需要的密钥总量。

表 3.1 分组密码算法密钥量统计

算法	单轮最大密钥量（bit）	密钥数据路径位宽（bit）	密钥参与运算轮数	密钥总量（bit）
DES	1×48	48	16	768
3DES	1×48	48	48	2304
AES-128	4×32	32	11	1408
AES-192	4×32	32	13	1664
AES-256	4×32	32	15	1920
IDEA	6×16	16	9	832
RC6	2×32	32	22	1408
Twofish	4×32	32	66	1344
SERPENT	4×32	32	33	4224
Mars	4×32	32	18	1280

表中可以看出，密钥量需求最大的为 SERPENT 算法，其容量为 4224 比特。既可以用 132 个 32 比特的寄存器堆来实现，也可以用 $2^8 \times 32$ 容量的 RAM 实现。存储模块 MAM 在配置控制模块 CCM 控制下完成数据的同步读写操作，实现数据的存储功能。

3.2.5 配置控制模块 CCM

配置控制模块 CCM 负责对整个可重构密码处理模型的处理过程进行控制。CCM 功能应包括以下两部分：

一、完成对整个 RCPA 的配置操作，即将配置信息注入到 RCU、ICM 以及 MAM 中。

RCPA 配置信息的存放可以有三种形式：

第一种形式：RCPA 中每个 RCU 内部的 RCE 中以及互连模块 ICM 中都设置独立的专有配置寄存器。在系统静态配置时，通过配置控制模块 CCM 将配置信息注入到各个

RCE 中以及互连模块 ICM 内的专有配置寄存器中。当需要执行某个配置时，配置控制模块 CCM 动态产生相应的配置地址信号选择所需的配置文件。该方式由于每个 RCE 以及 ICM 内专有配置寄存器可存不同的配置，配置控制模块 CCM 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。因此其优点是配置控制模块 CCM 设计简单，配置执行简单。缺点是 RCE 以及 ICM 电路占用面积大，编程深度低。

第二种形式：将 RCPA 中所有 RCU 内部的以及互连模块 ICM 的配置信息都置于配置控制模块 CCM 内部的配置指令存储单元中。在算法执行时，配置控制模块 CCM 将配置信息动态的注入到相应 RCU 和 ICM 的内部电路中。该方法的优点在于减少了 RCE 和 ICM 内部电路的占用面积，而且在一定程度上提高了编程深度和计算灵活性。缺点是配置信息主要依靠 CCM 动态配置，增加了 CCM 设计的复杂性和配置指令的长度。

第三种形式：结合两种形式的优缺点，采用了一种折衷方案。将 RCPA 中所需要的配置信息，分别放置在 RCU 内部的 RCE 和互连模块 ICM 的内部专有配置寄存器以及控制模块 CCM 内部的配置指令存储单元中。将一部分配置信息通过静态配置的方式写入到 RCE 和 ICM 的专有配置寄存器中，当需要执行某个配置时，配置控制模块 CCM 只需要动态产生相应的配置地址信号选择所需的配置文件。另一部分信息存放在 CCM 内部的配置指令存储单元中，在算法执行的每个时钟周期，CCM 将配置信息动态输入到相应 RCU 和 ICM 的内部电路中。该方式既有效减少了配置指令的长度，降低了译码的复杂性又保持了较高的编程深度和重构灵活性。

通过以上的分析，考虑分组密码算法处理的特点以及降低指令系统设计难度和指令译码控制逻辑的复杂性。在配置方式选择上应采用静态重构和动态重构相结合的方式。在执行密码运算前，配置控制模块 CCM 可根据算法的需要向 RCU、ICM 以及 MAM 注入静态配置信息。如向 RCU 中的 S 盒注入查表信息、向 ICM 注入开关控制信息以及向 MAM 注入算法所需要的常数值等。这些注入的配置信息将写入 RCU 和 ICM 相应模块的配置寄存器中，在算法执行过程中，这些配置信息无需改变直接读出至 RCU 和 ICM 内部电路中。对于在算法处理过程中需要频繁改变的运算功能部件 RCU 和 ICM，并在其配置信息量不大的情况下可以使用 CCM 来动态配置。此外，由于可以对 RCU 和 ICM 的输出寄存器动态配置，因此利用 CCM 可达到动态配置整个 RCPA 流水线操作的目的。由此 RCPA 经过 CCM 的静态和动态配置构成了完整的密码算法数据通路。采用静态重构和动态重构相结合的重构方式既可以有效减少 CCM 设计的复杂性和配置指令的长度，又有效提高加 / 解密处理的效率以及可重构系统的灵活性。

二、实现 RCPA 内部各模块之间的数据交换以及与外部的数据交换。

配置控制模块 CCM 完成内部各模块之间的数据交换主要是通过指令来完成 RCU 和 ICM 与 MAM 之间的数据传递。实现与外部的数据交换即完成上层编译产生的配置指令数据写入至 RCPA 中指令 RAM、外部的运算数据由输入数据缓冲区写入到 MAM、运算完毕的数据由 MAM 输出至输出数据缓冲区。

配置控制模块 CCM 根据功能的需求，其基本结构图如图 3.6 所示。

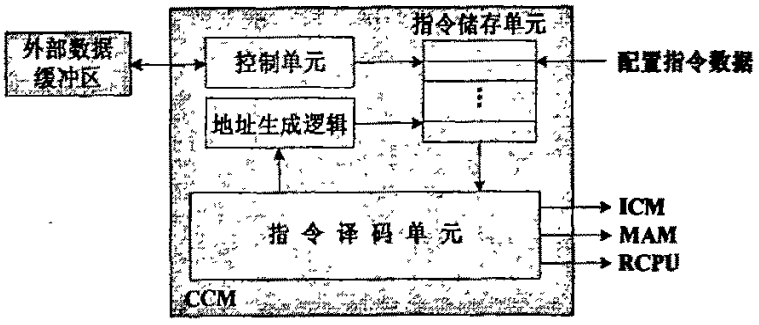


图 3.6 配置控制模块 CCM 结构图

CCM 由指令储存单元、地址生成逻辑、控制单元以及指令译码单元四个部分组成。由上层编译产生的配置指令数据进入指令储存单元的输入端时，控制单元产生相应的读使能和读地址，这样配置指令数据就写入了指令储存单元。在执行静态和动态重构配置时，控制单元发出读使能有效同时地址生成逻辑产生读地址，配置指令便从指令储存单元读出。指令译码单元对指令进行译码产生、ICM 以及 MAM 所需的配置信息注入到相应模块中，此外译码还产生 MAM 的读写控制信号，地址生成电路的控制信号以及其他所需的控制信号，如标志寄存器控制信号等。控制单元还负责与外部数据缓冲区的数据交换。在控制单元的控制下 RCPA 可以通过数据缓冲区与上层的数据进行交换。根据前面的章节对分组密码算法处理结构的分析，得到 RCPA 配置控制指令系统采用类超长指令字 VLIW/EPIC 结构。通过 VLIW/EPIC 方式可以同时为多个可重构密码处理单元 RCU、互连模块 ICM 以及存储模块 MAM 配置不同的数据。因此 VLIW/EPIC 方式很好的支持了 RCPA 对密码算法处理的并行和深度流水结构，有效提高了分组密码算法处理的效率。

3.3 RCPA 模型性能分析

RCPA 是根据分组密码算法处理结构的特点提出的可重构密码处理框架，其很好地匹配了密码处理的特点。相对其它的通用密码处理结构来说，其性能增益主要源于对密码处理的并行化以及深度流水线结构设计。

通过前面对分组密码的分析可以得知，算法分组中的一步操作很多情况下是化解为几个更小处理位宽的子分组并行处理，并且处理位宽以 32 比特为主。RCPA 中每一级流水线能够支持的子分组并行处理数量随算法的不同而有所不同，并且与操作的分割和调度有关。基于上述特点，RCPA 可以根据不同的算法重构成不同深度和宽度（一级流水线上处理单元的个数，也称流水线并行度[28]）的流水线结构。

在对 RCPA 模型流水结构进行性能分析之前，先定义 RCPA 模型中的几个特征参数。将 RCPA 模型中，每一行 RCU 的个数设为 R(通常设为 4 的整数倍)，RCU 的行数设为 C_r，ICM 的个数设为 C_i。RCPA 流水线宽度定义为 P、流水线深度定义为 H、可用的流水线条数定义为 L。设分组长度为 N，则各参数存在下列关系：

流水线最大深度为：$H=(C_r+C_l)\times(\frac{R}{N}\times 32)$

流水线条数最大为：$L=\frac{R}{N}\times 32$

流水线宽度为：$P=\frac{N}{32}$

图3.7显示了RCPA模型对不同的分组长度重构为不同流水线宽度以及流水线深度结构，图中的RCPA模型特征参数R=4。

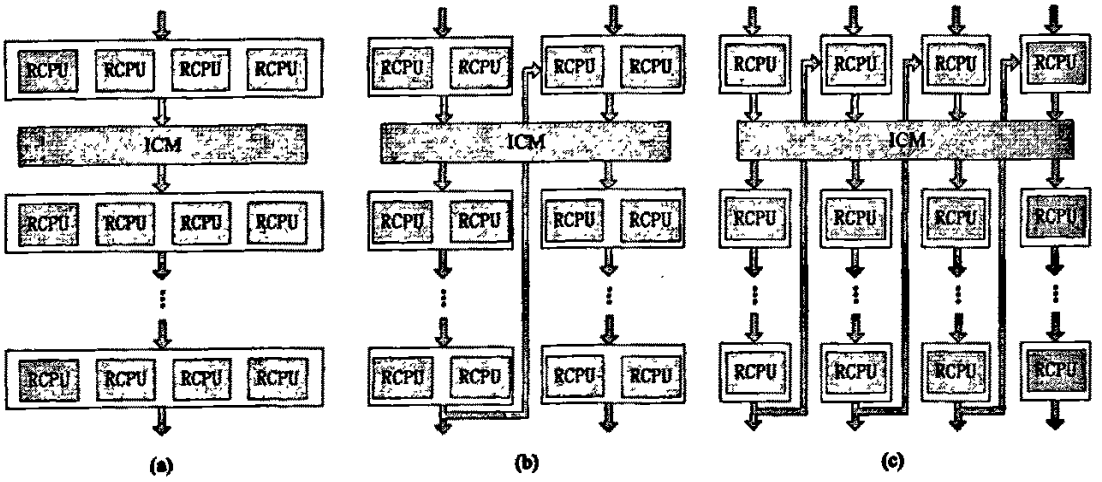


图3.7 RCPA模型重构为不同流水线宽度以及流水线深度结构

图3.7(a)中N=128比特，可通过动态配置实现一条流水线宽度为4，最大深度为C_r+C_l的流水线；图3.7(b)中N=64比特，可实现流水线宽度为2，流水线深度可配置为一条$2\times(C_r+C_l)$的流水线或两条深度为C_r+C_l的流水线；图3.7(c)中N=32比特，可实现流水线宽度为1，流水线深度可配置为一条$4\times(C_r+C_l)$的流水线或两条$2\times(C_r+C_l)$的流水线或四条深度为C_r+C_l的流水线。单条流水线深度的增加或多条不相关的流水线并行处理多个分组，使RCPA性能得到很大的提高。

下面进一步分析其性能，假设一个待处理的数据集中有M个密码分组，每个分组长度为N，进行某一密码算法处理，以最大流水线深度的方式执行该算法，算法具有r轮的操作，每轮操作可由最多k步子分组的并行操作构成。则：

(1) 在32位通用密码处理器结构下，理想情况下的最小处理周期数为：

$$CP_1=M\times\frac{N}{32}\times k\times r$$

(2) 在RCPA下，假设操作以最大流水线深度h执行，RCU个数可满足k步子分组并行则：

$$h=(C_r+C_l)\times(\frac{R}{N}\times 32),$$

可得到理想情况下的最小处理周期数为：

$$CP_2 = m + r \times k \times \left\lceil \frac{M}{h} \right\rceil - 1 \quad \text{其中} \quad m = \begin{cases} M \bmod h & M \neq hn \quad n = 1,2,3\cdots \\ h & M = hn \quad n = 1,2,3\cdots \end{cases}$$

(3) 在全流水专用硬件实现（ASIC）的情况下，流水线深度可实现为 r，算法所有并行操作均可并行实现，此时的处理周期数为：

$$CP_3 = M + r - 1$$

从上面的简单的处理性能分析可以看出，三种密码算法处理结构随着处理分组数 M 的增大处理效率存在较大的差距。在 RCPA 实现中，由于可支持子分组并行执行以及单向流水操作，可达到比通用密码处理器更高的性能。全流水专用硬件（ASIC）可以获得最高的性能，但其灵活性和安全性是较低的。例如取 N=128，k=4，r=10，R=4，C_r=4，C_l=4，在相同时钟频率的支持下，CP1、CP2 和 CP3 随着处理分组数 M 的增加取值变化如图 3.8 所示。从图中三种结构的粗略评价来看，RCPA 结构对比通用密码处理器处理的加速比在 M 取 50 至 300 范围内平均约为 30，是专用硬件结构的五分之一。RCPA 针对密码处理的加速比主要源于处理结构的并行处理机制和深度流水线机制，这两种机制能使最多 H 个分组并行执行。

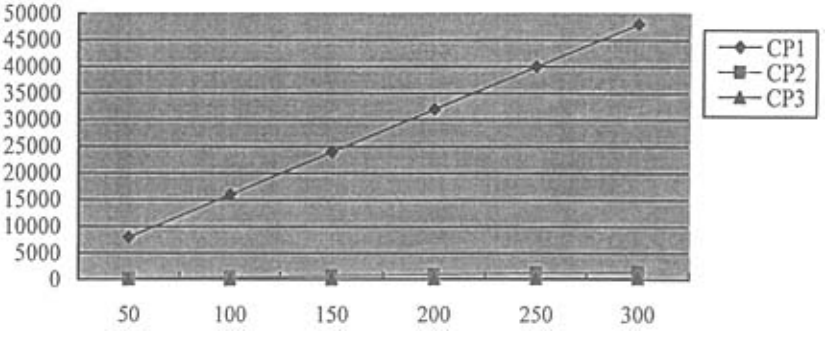


图 3.8 通用密码处理结构、RCPA 和专用密码处理结构性能比较

3.4 本章小结

在设计可重构分组密码处理模型时，必须充分考虑分组密码处理结构的特征，同时需要结合可重构体系结构设计要素。因此本章首先分析了可重构分组密码处理模型设计中的重构粒度、互连网络、计算模型、编程深度以及配置策略等几个关键问题。通过研究本文得出可重构分组密码处理模型的设计特点应是粗粒度、混合型互连网络、类 VLIW/EPIC 计算模型、动态与静态配置相结合的可重构处理结构。

本章提出的 RCPA 模型中包括了可重构密码处理单元模块 RCU、互连模块 ICM、存储模块 MAM 以及配置控制模块 CCM。RCPA 可以根据算法的不同，实现可变并行度的流水处理并且流水线的深度可以配置。RCPA 采用静态和动态混合重构方式，支持多配置文件，从横向和纵向两个方向配置各个密码处理单元。因此该模型支持了分组密码算法的分组内的横向并行处理和多分组间的纵向并行或流水处理两种处理模式，很好地匹配了密

码处理的特点。通过对 RCPA 定义特征参数便可得到具有不同规模和适用性的可重构密码处理结构。最后本章对 RCPA 性能进行分析，结果表明其性能远高于通用密码处理器的性能。

第四章 可重构密码处理元素 RCE 研究与设计

可重构密码处理元素 RCE 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，使每个可重构密码处理元素 RCE 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。可重构密码处理元素 RCE 的设计思想是：在可重构密码处理元素 RCE 中设置某些可被重复配置的功能参数接口，通过对功能参数接口的动态或静态配置从而改变可重构密码处理元素 RCE 内部的电路结构，满足不同密码算法的应用需求。本章将对 RCPA 模型中的可重构密码处理元素 RCE 硬件实现原理进行研究。重点介绍了移位单元、有限域 $GF(2^8)$乘法单元、模加/减运算单元、模乘运算单元、S 盒替代单元以及置换单元等可重构密码处理元素 RCE 的设计方法。

4.1 移位单元研究与设计

4.1.1 实现原理研究

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。循环移位运算在分组密码算法中使用较为普遍，数据宽度 N 通常为 8 比特、32 比特、64 比特及 128 比特；循环移位方向有左移与右移两种；循环位数有两种类型：固定位数与可变位数（即移位位数随机）；逻辑移位运算数据宽度 N 多为 32 比特，移位方向为左移和右移，移位位数主要取 1~32 位。表 4.1 列出了主要的一些分组密码算法及分组移位运算。

表 4.1 典型分组密码算法移位操作

算法	主要移位运算
IDEA	128 比特宽度，固定循环移位
RC5	W 比特宽度，位数随机，循环左/右移位 W=32，64
RC6	32 比特宽度，位数随机，循环左/右移位
Mars	32 比特宽度，位数随机，循环左/右移位
Rijndael	以 32 比特为单位，进行 128 比特宽度的移位
CAST-256	32 比特宽度，位数随机，循环左/右移位
CRYPTON	32 比特宽度，位数随机，循环左/右移位
SERPENT	32 比特宽度，固定循环及逻辑移位
SAFER+	8 比特宽度，固定循环移位
Twofish	32 比特宽度，固定循环移位

通过以上的分析可以得出，由于移位数据宽度差别较大，并且 32 比特位宽的移位最

为普遍。因此，移位运算电路设计在功能上要求可以实现任意位数的32比特位宽、16比特位宽及8比特位宽的逻辑左移、逻辑右移、循环左移、循环右移。此外，还可以通过级联实现64比特位宽的移位。

移位运算的通常算法是以移1位运算为基础，循环移位 k 位是通过调用移1位基本运算 k 次实现的。这种算法的电路实现，通常是以线性反馈移位寄存器LFSR为基础进行设计。利用LFSR构造的双向移位寄存器的移位操作受时钟的控制，移几位就需要几个时钟。因此，移位速度受最高时钟频率及移位位数的影响。采用这种算法进行循环移位运算很难满足高速数据处理的需求。

循环移位运算的另一种算法是桶形移位(Barrel Shift)原理实现移位功能。该算法采用分级的方法，整个移位值被分解成移2的指数位，最大移位宽度 N 是由 Log_2N 级构成的。对第i $(0 \le i \le \log_2 N-1)$级设置相应的控制信号，控制其将该级的结果移 2^i 位或者不移位将上一级的结果传至下一级；这样，将移位位数用二进制编码表示，作为各级的控制信号，就可以实现0至 N 的任意一种移位结果。这种算法的电路实现，通常可以采用数据选择器或传输门实现，移位速度不再受最高时钟频率的影响，其速度以对数方式取决于移位宽度[41]。

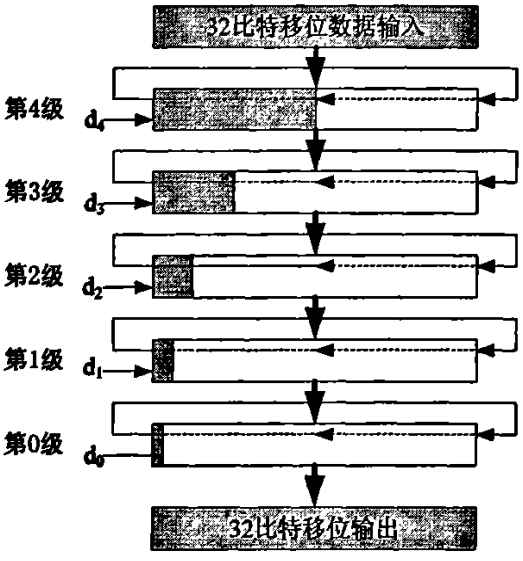


图4.1桶形移位原理实现循环左移示意图

图4.1给出了一个32比特数据采用桶形移位算法实现循环左移的实例：由于移位宽度为32比特，所以级数 N=Log232=5，电路由5级循环移位电路构成。在此电路设计中，第4级移位电路完成 2^4=16比特循环左移；第3级移位电路完成 2^3=8比特循环左移；第2级移位电路完成 2^2=4比特循环左移；第1级移位电路完成 2^1=2比特循环左移；第0级完成 2^0=1比特循环左移。任意小于32-bit长度的移位均可以拆分为16-bit、8-bit、4-bit、2-bit和1-bit长度移位的组合。每一级的移位使能信号可以由移位长度的二进制代码得到，设移位长度为：$d_4d_3d_2d_1d_0$（二进制），则 d_4 为第一级的移位使能信号；d_3 为第二级的移位使能信号，……，最后一级由 d_0 控制。当 $d_i(0 \le i \le 4)$ 为1时，则相应第i级进行 2^i 比特

的循环左移；为 0 时则不移位将 i+1 级结果传至 i-1 级。

通过以上的分析，在电路设计中考虑采用多级互联(Multilevel Connection)的桶形移位设计原理实现完整的移位运算。

4.1.2 电路设计

移位运算单元电路整体框图如图 4.2 所示，移位电路模块可以实现任意位数的 8 比特位宽、32 比特位宽及 64 比特位宽的逻辑左移、逻辑右移、循环左移、循环右移。

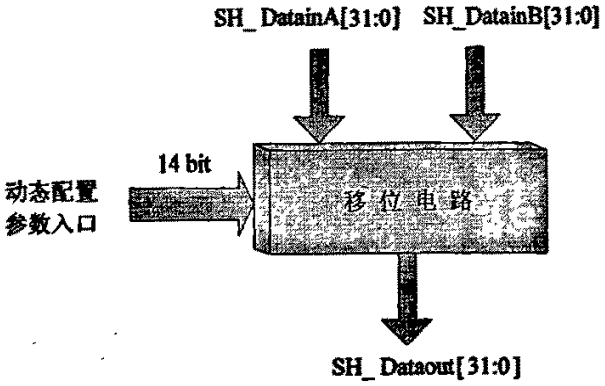


图 4.2 移位单元电路整体框图

移位电路模块的数据输入为两个 32 比特的数据，分别为 SH_DatainA 和 SH_DatainB。电路输出为 SH_Dataout，动态功能配置参数接口设有 SH_Width、SH_opsel、SH_mode、SH_smode、SH_CR、SH_SR。通过对这些功能参数接口的配置从而改变移位电路的内部结构，满足不同的应用需求。功能参数接口定义如表 4.2 所示

表 4.2 移位单元功能参数接口定义

功能参数接口	功能参数描述
SH_CR[4:0]	移位位数来自立即数
SH_SR[4:0]	移位位数来自寄存器
SH_smode	0：固定移位模式，移位位数为立即数
	1：可变移位模式，移位位数为寄存器值
SH_opsel	0：选择 8、16、32 比特移位模式
	1：选择 64 比特级联移位模式
SH_Width[1:0]	00：8-bit 位宽移位
	01：16-bit 位宽移位
	10：32-bit 位宽移位

	11：保留
SH_mode[1:0]	00：逻辑左移
	01：逻辑右移
	10：循环左移
	11：循环右移

移位单元的电路内部结构如图 4.3 所示分为高低两组移位电路，并且在两组的电路设计中均采用了多级互联的桶形移位设计原理。SH_DatainA 和 SH_DatainB 分别为高低两组移位电路的输入。每组移位电路设计均分为了 5 级 32 比特寄存器：第一级完成 16-bit 长度移位；第二级完成 8-bit 长度移位；第三级完成 4-bit 长度移位；第四级完成 2-bit 长度移位；最后一级完成 1-bit 长度移位。其中高组的移位电路在第二级使用了 2 个 16 比特寄存器级联，在第三到第五级寄存器中使用了 4 个 8 比特寄存器级联电路。这样高组移位电路就可在 SH_Width 和移位位数信号控制下独立完成 32 比特位宽、16 比特位宽或者 8 比特位宽的任意移位。

在整体的移位电路中设置低组移位寄存器是用级联方式来支持 64 比特的移位操作。64 比特的移位实现原理是：先将 64 比特的数据分为高低 32 比特分别输入到移位电路高组和低组移位寄存器中，高组移位寄存器进行 32 比特位宽的移位，同时低组移位寄存器也进行相同的方式移位，并将低组移位寄存器移出的数据输入到高组移位寄存器中。从而高组移位寄存器的输出数据就为 64 比特移位结果的高 32 比特数据。下一个时钟周期将输入 64 比特数据高低 32 比特互换，则高组移位寄存器的输出数据就为 64 比特移位结果的低 32 比特数据。至此，电路经过 2 个时钟周期，完成了 64 比特的移位运算。

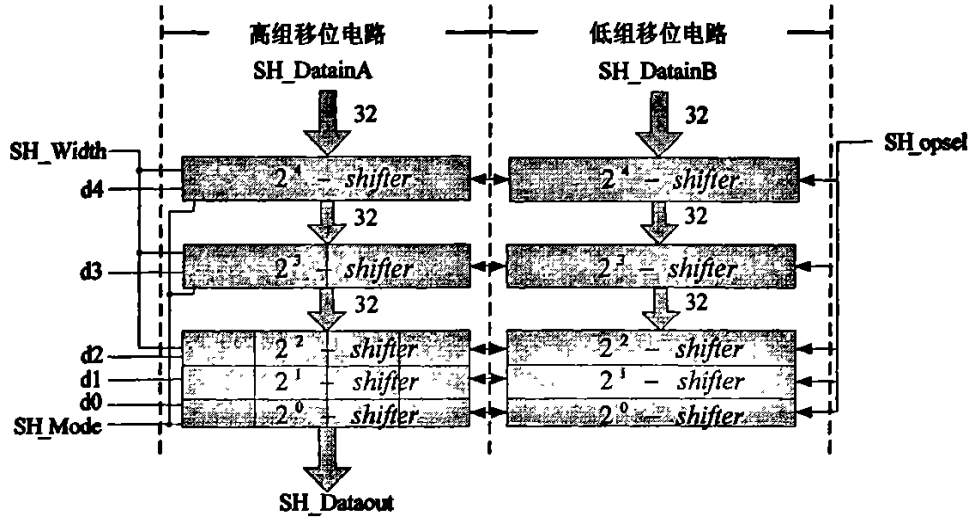


图 4.3 移位单元电路内部电路图

4.2 有限域 $GF(2^8)$乘法单元研究与设计

4.2.1 实现原理研究

分组密码应用中，有限域 $GF(2^n)$乘法运算主要在 $GF(2^8)$，$GF(2^9)$以及 $GF(2^7)$域上。其中在 $GF(2^8)$域上乘法运算最为常见，占到了全部有限域乘法的 57.14%，如常用的 Rijndael 以及 AES 候选算法 Twofish。因此，本文针对有限域 $GF(2^8)$上乘法运算单元进行研究和设计，使其可满足 $GF(2^8)$域上的任意乘法运算。

首先分析 $GF(2^n)$上乘法运算：在 $GF(2^n)$上，我们设 $f(x)$是不可约多项式，$f(x)$称为域多项式，$GF(2^n)$中元素可视为小于 n 次的多项式。设 $GF(2^n)$上的两个元素为：

$$a(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1x + a_0$$

$$b(x) = b_{n-1}x^{n-1} + b_{n-2}x^{n-2} + \cdots + b_1x + b_0$$

定义两者的乘积为：

$$c(x) = a(x)b(x) \mod f(x) = c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \cdots + c_1x + c_0$$

其中：$f(x) = x^n + f_{n-1}x^{n-1} + f_{n-2}x^{n-2} + \cdots + f_1x + f_0$

根据乘法的定义，$GF(2^n)$上的乘法运算实际是对多项式的系数进行运算，其实现的基本方法是采用“Shift-and-add”算法[42,43]。该算法需要执行多次移位，且加法是串行操作。算法实质上是根据乘数，将被乘数不断移位后，将各结果作模 2 加，得到原始乘积，再进行模运算，就可以得到最终乘法结果。在实现中，移位分为向左移与向右移两种方式，并且通常采用边移位边做模运算的方式进行。采用向左移位方式时，是根据乘数的第 i 位的值确定移位位数，若第 i 位的值是 1，则被乘数向左移 i 位；若第 i 位的值为 0，则被乘数不参与运算。由以上分析可知，利用逻辑左移很容易实现被乘数的移位，同时对每次移位结果进行模运算，最后将各结果作模 2 加，即可得到最终结果。“Shift-and-add”算法描述如下：

Shift-and-add 算法描述

Input：二元多项式 $a(x)$，$b(x)$，最高次项系数最多为(n-1)

Output：$c(x) = a(x)b(x) \mod f(x)$

(1) $c \leftarrow 0$

(2) For i= n-1 to 1 do

  If a_i=1 then $c \leftarrow c+b$

  $c = c \cdot x \bmod f(x)$

(3) If a_0=1 then $c \leftarrow c+b$

  Return (c)

根据移位相加“Shift-and-add”算法以及有限域 $GF(2^8)$的性质，对于具有任意不可约多项式的 $GF(2^8)$域上的乘法，假设用多项式 x 乘以 $a(x)$有：

$$r(x) = (a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0)\cdot x \ mod \ (x^8 + f_7x^7 + f_6x + f_5x^5 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0) \quad (4\text{-}1)$$

由上式（4-1）可得：

$$r(x) = (a_7f_7 + a_6)x^7 + (a_7f_6 + a_5)x^6 + (a_7f_5 + a_4)x^5 + (a_7f_4 + a_3)x^4 + (a_7f_3 + a_2)x^3 + (a_7f_2 + a_1)x^2 + (a_7f_1 + a_0)x + a_7f_0 \quad (4\text{-}2)$$

由式（4-2）可得到任意不可约多项式的 $GF(2^8)$域上的 x 乘法电路。接着将 x 乘法电路依次串行级联起来，便得到了 $GF(2^8)$域上任意不可约多项式的基本乘法运算电路。将多个基本乘法运算电路并联以及异或操作，便构成了有限域 $GF(2^8)$乘法矩阵运算电路[44]。

根据以上对有限域 $GF(2^8)$乘法运算的分析可知，运算过程需要乘数多项式以及不可约多项式来参与。由于在分组密码算法实际应用中，有限域 $GF(2^8)$乘法通常使用在矩阵乘法形式中，乘数多项式和不可约多项式一般是较为固定的。因此，可对这两种运算参数可以采取静态配置的方式注入到有限域乘法运算电路中。所以，在电路内部需要设置静态配置寄存器堆用来存储乘数多项式和不可约多项式信息。根据对分组算法的分析可得，乘数多项式信息一般为 128 比特，不可约多项式信息为 8 比特，静态配置寄存器堆大小可设为 4×136 比特可满足算法需求。

4.2.2 电路设计

有限域 $GF(2^8)$乘法单元整体电路结构如图 4.4 所示。电路分为两个部分，分别是有限域 $GF(2^8)$ 矩阵乘法电路和静态配置寄存器堆。有限域 $GF(2^8)$ 矩阵乘法电路模块是整个电路的核心模块，负责完成有限域 $GF(2^8)$上的矩阵乘法运算工作。静态配置寄存器堆的功能是完成有限域上 128 比特乘数多项式以及 8 比特不可约多项式信息的存储工作。

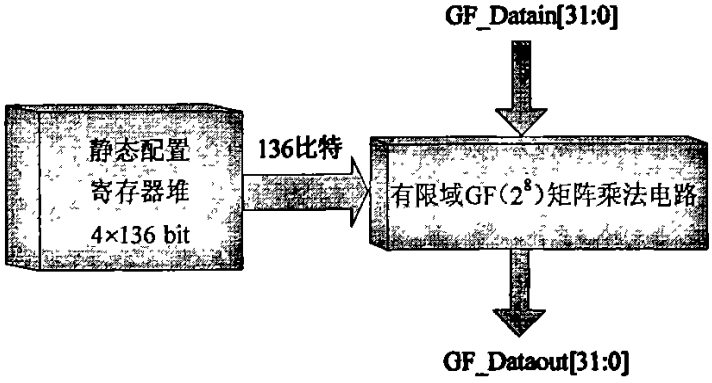


图 4.4 有限域 GF(2^8)乘法单元整体电路图

当系统在运算前，进行静态配置时，配置数据在配置时钟、配置地址和写使能控制下写入至静态配置寄存器堆中。在进行密码运算时，通过动态配置指令给出读使能和读地址，将需要的配置信息读入到有限域 $GF(2^8)$乘法器中，从而完成整个有限域乘法运算。表 4.3 对有限域 $GF(2^8)$乘法单元功能参数接口的定义进行了说明。

表 4.3 有限域乘法单元功能参数接口定义

功能参数接口	功能参数描述
GF_conf_data	静态配置数据输入，包括乘数多项式以及不可约多项式
GF_conf_wen	静态配置寄存器写使能，静态配置使能
GF_conf_addr	配置寄存器读写地址
GF_conf_ren	静态配置寄存器读使能，动态配置使能

根据公式（1.2）可得到任意不可约多项式的 $GF(2^8)$域上的 x 乘法电路，电路如图 4.5 所示。

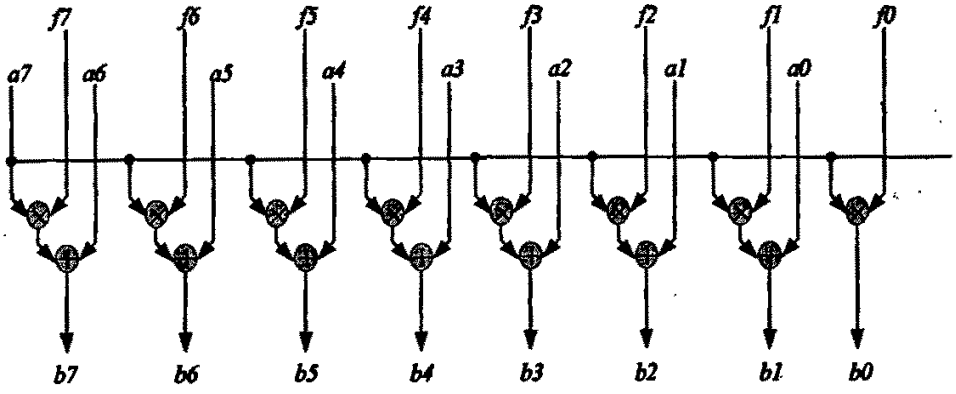


图 4.5 有限域 GF(2^8)上的 x 乘法电路

$(f_7 \cdots f_1 f_0)$为 $GF(2^8)$域不可约多项式二进制表示的低八位，可将此电路定义为 xt 电路。设计了最基本的 xt 电路，接着将 7 个 xt 电路依次串行级联起来，再将相应的结果进行三级异或运算，便得到了如图 4.6 所示的 $GF(2^8)$域上任意不可约多项式的乘法运算电路，称为基本有限域乘法电路，记为 Gfmult。

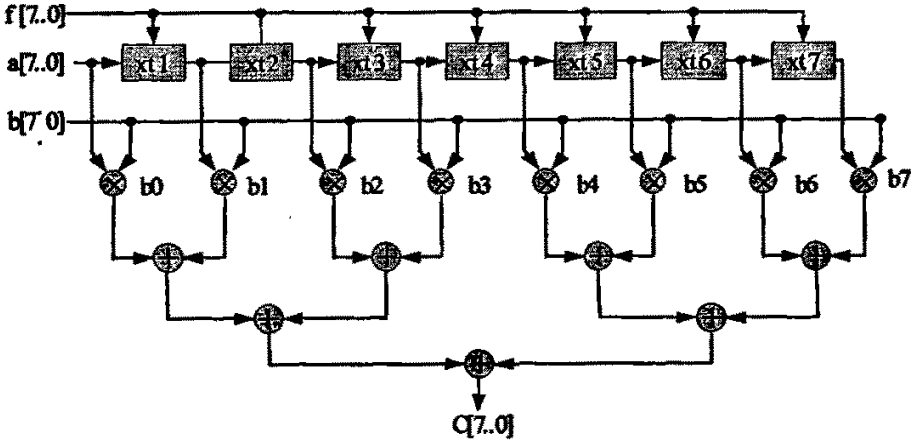


图 4.6 基本有限域 GF(2^8)乘法电路

$f[7..0]$即 xt 电路中的$(f_7 \cdots f_1 f_0)$，为 $GF(2^8)$域上不可约多项式二进制表示的低八

位，*a[7..0]* 与 *b[7..0]* 表示两个相乘的 $GF(2^8)$ 域上的多项式。可以看出式中 $c(x) = a(x)b(x) \bmod f(x)$，$a(x)$，$b(x)$以及 $f(x)$均是实时输入的，因此该 $GF(2^8)$域上乘法器的不可约多项式 $f(x)$和乘数多项式 $b(x)$均可配置。在多数分组密码应用中，如 Twofish、Square、Rijndael 等算法通常将所有的 8 比特的乘数多项式以固定矩阵方式给出，表示为矩阵乘法形式。根据这一特点，本文如图 4.7 所示将四路基本有限域 $GF(2^8)$乘法电路进行并联，再将结果进行异或操作，这样便得到了矩阵乘法中 1 行×1 列的 8 比特结果。进一步将该电路进行 4 路并联，输入 32 比特的运算数据、128 比特的乘数多项式以及 8 比特的不可约多项式便完成了的完整的矩阵乘法电路。

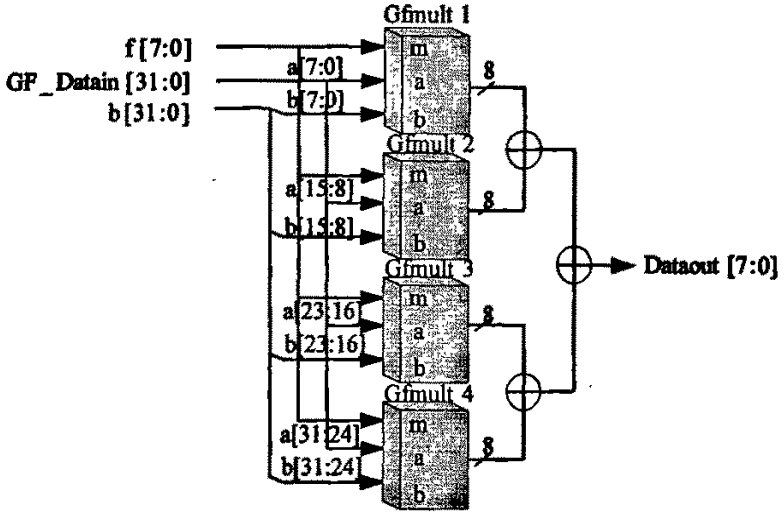


图 4.7 有限域 $GF(2^8)$矩阵乘法中 1 行×1 列电路

4.3 模加/减单元研究与设计

4.3.1 实现原理研究

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 8、16、32。例如，FEAL、SAFER+算法中使用了模 2^8 加法，IDEA 使用了模 2^{16} 加法，Twofish 、Mars、RC6、CAST 等算法使用了模 2^{32} 加法。此外，在分组密码算法中，经常使用到 PHT(Pseudo-Hadamord Transform)操作。该操作执行 A+2B 运算，可看作为模加/减功能的一种变型。因此，要求模加减单元不仅可以执行模加与模减功能，还具有 PHT 操作功能，同时支持模 2^8、2^{16}、2^{32} 的运算。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。对于串行结构的加法器，在实现上只需将 n 位 CPA 或 CLA 的最高位全加器的进位输出舍去。

以上分析可以看出，分组密码运算中的模加/减运算单元计算的最小的操作粒度为 8 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 2^8、2^{16}、2^{32} 的加/减操作。电路在实现上需要 4 个 8 位的加法器，从而可进行 1 个 32bit 模 2^{32} 加/减操作、同时

进行 2 个 16bit 模 2^{16} 加/减操作、同时进行 4 个 8bit 模 2^{8} 加/减操作。8 位加法器可以采用多种方法来实现，典型的加法器类型主要有：串行进位加法器、超前进位加法器、保留进位加法器等。

（1）串行进位加法器（Carry Propagate Adder，CPA）

串行进位加法器是采用 n 个一位全加器线性级联形成 n 位加法器。其第 i 全加器的输出取决于本位以前的所有位的输入。对于这种方式，高位运算必须等待低位进位来到后才能进行。对于 n 级的 CPA 加法器来说，它的延迟时间是一位全加器的 n 倍，因此不适合 ASIC 实现。而目前主流的 FPGA 内部，基本逻辑单元之间均设有快速进位链，非常适合 CPA 的实现。

（2）超前进位加法器（Carry Look-ahead Adder，CLA）

超前进位加法器利用超前进位链能有效减少进位延迟的特性，将进位由进位逻辑电路产生，各进位彼此独立，不依赖于进位传播的一种加法器。因此，具有延迟小、速度高的特点 ，适合在 ASIC 内实现。但是超前进位加法器的扩充是比较困难的，因为不同级数上的进位逻辑不同，级数越高则所需的逻辑越复杂，对于位数比较长的情况，可以采用分块的方案，即先建立比较小的 CLA 基本块，再在以它为基本块建立更大的 CLA，一般地 n 比特直接实现的 CLA 的延迟是 $O(log_2 n)$，显然比 CPA 要快很多，但此速度的提高是以牺牲面积为代价的。CPA 消耗的面积为 $O(n)$，而分块的 CLA 的面积消耗为 $O(nlog_2 n)$。

（3）进位保留加法器（Carry Save Adder，CSA）

n 位 CSA 是由 n 个非串行连接的一位全加器并行组成的，它的功能是将三个 n 比特整数 A，B，C 相加，产生两个整数结果 C' 和 S，即 $C'+S=A+B+C$。因为输入的操作数 A、B 和 C 同时送入加法器，因此整个加法器的延迟时间为一位全加器的延迟时间。CSA 的延迟时间与长度无关，不会随着操作数的增加而增加，同时它也很容易扩展，n-bit 的 CSA 的面积便是 n 倍的一位全加器的面积。CSA 将三个操作数相加，产生两个结果，可看作一个 3:2 的压缩过程，因而 CSA 适用于连加运算的高速实现。

4.3.2 电路设计

模加/减单元整体电路如图 4.8 所示，电路由移位电路和模加/减运算电路两部分构成，移位电路的功能是用来产生 PHT 运算需要的 2B，可由逻辑左移一位来实现。由于模加/减运算需要 8、16、32 不同的操作位宽，因此移位电路相应的具有 8、16、32 不同位宽的移位功能。模加/减运算电路是整个电路的核心部分，用来对不同位宽的操作数进行模加/减及 PHT 运算。

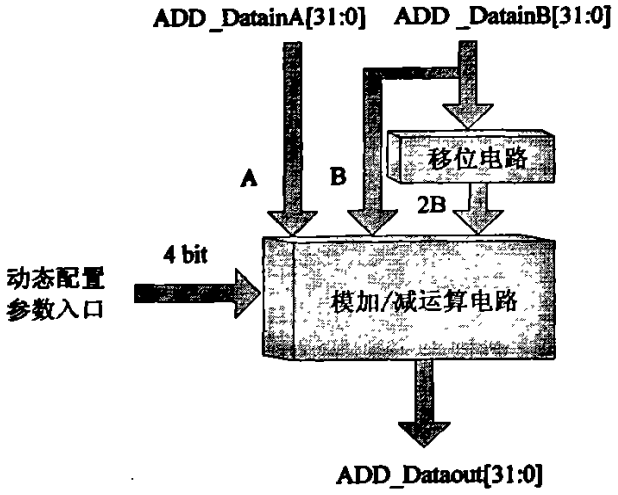


图 4.8 模加/减单元整体电路图

模加/减运算电路由 4 个级联的 8 位模加/减器数据选择器和译码电路所构成。可以实现 8 位，16 位，32 位的模 2^8、2^{16}、2^{32} 加/减运算，此外还可实现 8 位，16 位，32 位的 A+2B（PHT）运算，其中 8 位模加/减器采用了超前进位加法器实现。模加/减运算电路数据路径如图 4.9 所示。

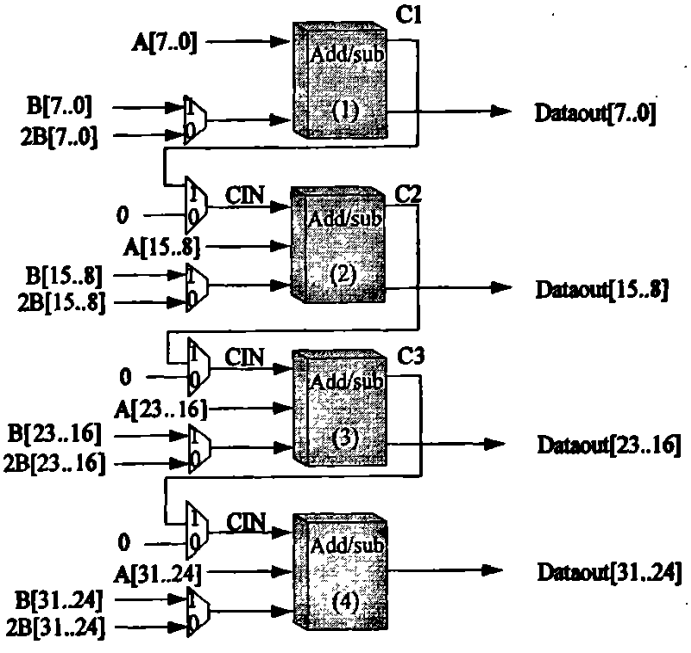


图 4.9 模加/减运算电路数据路径

外部输入功能参数配置信号，经过译码电路产生模加/减运算电路所需要的的数据选择器控制信号以及各加法器级联信号，从而可以用来控制模加/减电路不同的操作位宽和操作模式。模加/减运算单元功能参数接口定义如表 4.4 所示。

表 4.4 模加/减单元功能参数接口定义

功能参数接口	功能描述
ADD_mode[1:0]	00：模加运算
	01：模减运算
	10：执行 A+2B（PHT）运算
	11：保留
ADD_width[1:0]	00：8 比特输入位宽，模 2^8 运算
	01：16 比特输入位宽，模 2^{16} 运算
	10：32 比特输入位宽，模 2^{32} 运算
	11：保留

4.4 模乘单元研究与设计

4.4.1 实现原理研究

通过对分组密码算法的分析可以看出，模乘运算在 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=(A×B) mod 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 $2^n\pm1$，且参数 n 大多为 16 或 32。表 4.5 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 4.5 分组密码的模乘操作类型

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

通过以上的分析可得，由于分组密码运算中的模乘运算单元计算的最小的操作粒度为 16 比特。在研究中，我们发现可通过数学变换将 32 比特的模乘运算分解为 16 比特的乘法运算，再由 16 比特的乘法电路再加上外围的 16 位的加法电路，模修正电路以及数据选择器从而可以实现模 2^{16}、模 $2^{16}+1$、模 2^{32}、模 $2^{32}-1$ 的乘法运算，这样既扩展了本模块的功能，又在相应的程度上节约了面积。因此，电路的设计中采用以 16 比特乘法器为基本结构的乘法器以及加法器阵列来完成所需要的模乘运算。

进行模 2^{32} 乘法运算时，电路以 16 位的乘法器为基本单元，通过组合来实现 32 位的模乘运算。其实现基本原理是利用了乘法的分配律，见下式：

$$A\times B=(A_H\times 2^{16}+A_L)\times(B_H\times 2^{16}+B_L)$$

$$= A_H \times B_H \times 2^{32} + (A_H \times B_L + B_H \times A_L) \times 2^{16} + A_L \times B_L$$

其中：A和B均为32比特，A_H为A的高16比特，A_L为A的低16比特；B_H为B的高16比特，B_L为B的低16比特。由于两数相乘后要进行模2^{32}操作，所以$A \times B \bmod 2^{32} = (A_H \times B_L + B_H \times A_L) \times 2^{16} + A_L \times B_L$。进一步的分析可以看出$(A_H \times B_L + B_H \times A_L) \times 2^{16}$后要进行模$2^{32}$操作，只需将$A_H \times B_L$和$B_H \times A_L$相乘的低16位相加即可。最后再与$A_L \times B_L$的高16位相加，结果就是$A \times B \bmod 2^{32}$的高16位。$A_L \times B_L$的低16位就是$A \times B \bmod 2^{32}$的低16位。

在整体电路设计中，16位乘法器是核心运算部件，也是延时最大的运算单元。因此必须考虑采用优化设计加快乘法的处理速度。乘法操作的完成依赖于所有部分积之和的计算，为了加快乘法的处理速度，主要从两个方面来考虑：

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。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[45]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：

$$Y = -2^{n-1} Y_{n-1} + \sum_{i=0}^{n-2} 2^i Y_i, \quad Y_{-1} = 0 \tag{4-3}$$

那么乘数Y也可以表示为：

$$Y = \sum_{i=0}^{n/2-1} (Y_{2i-1} + Y_{2i} - Y_{2i+1}) 2^{2i}, \quad Y_{-1} = 0 \tag{4-4}$$

其中n是偶数，如果是奇数位的数则需要扩展1位。根据式(4-4)可以知道部分积只有位数的一半，而且根据Y_{2i-1}，Y_{2i}，Y_{2i+1}的不同，与被乘数相乘得到的部分积也不同。

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或Wallace树形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[46]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，前者的安排是串行的，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。Wallace树型阵列乘法器，实际上就是采用Wallace树加法阵列来完成部分积合并的。而且构成Wallace树的等效进位保存加法器的数目，与标准的进位保留加法器阵列中的数目是完全相同的。不同的只是Wallace树加法阵列的加法器单元之间的连线经过了重新连接，使得需要最长的合并电路延迟的部分积。从而与串行阵列乘法器相比，在不增加任何逻辑门的情况下，将整个阵列的延迟特性从O(n)缩减到O(log(n))。

实现模$(2^n \pm 1)$乘法主要有两种实现方式：一种是在进行乘运算时直接进行模修正。例

如采用基于 Ma 算法[47]的方案；另一种方法为采用 Low±High 算法[58]方案，该方法是在乘运算完毕后再进行模修正。为实现模乘单元对于分组密码算法通用性，本文采取了第二种 Low±High 方案。下面介绍基于 Low±High 算法的模修正设计原理。

对于模 2^n-1 乘法模修正设计原理运算如下，设 x，y 均为 n 比特，则模 (2^n-1) 乘法可表达为:

$$c = xy \bmod (2^n-1) = (c_0 \cdot 2^{2n} + c_H \cdot 2^n + c_L) \bmod (2^n-1) \quad \text{其中，} c_0 \in \{0,1\},\ 0 \le c_L,\ c_H \le 2^n;$$

2^n -1 模修正算法为，当 $x=y=2^n$ 时，可知 $c_0=1$，$c_L=c_H=0$；由于 $2^n=(0+1) \bmod (2^n-1)=1 \bmod (2^n-1)$，所以 $c=1\times1=1$。其余情况下（$c_0=0$），当 $c_L+c_H<2^n-1$ 时，$c=c_L+c_H$；当 $c_L+c_H \ge 2^n-1$ 时，$c=c_L+c_H-2^n+1$。式中 c_H 和 c_L 分别为 n 比特乘法运算结果高低的 n 比特数据。因此，模 (2^n-1) 乘法可以通过一个 n 位无符号乘法器与一个 n 位模 (2^n-1) 加法器实现，如图 4.10 所示。

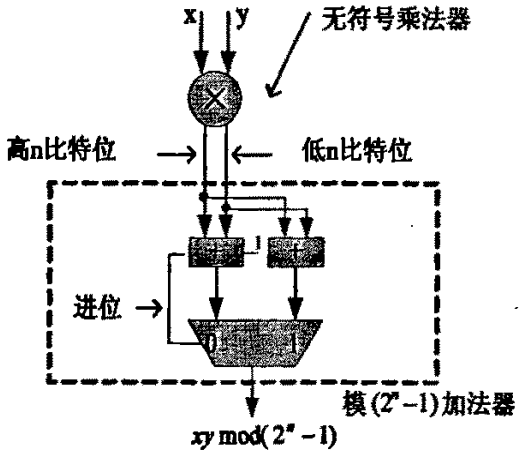


图 4.10 基于乘法器的模 2^n-1 乘法器结构图

同理可得到模 2^n+1 乘法模修正设计原理运算，设 x，y 均为 n 比特，则模 (2^n+1) 乘法可表达为:

$$c = xy \bmod (2^n+1) = (c_0 \cdot 2^{2n} + c_H \cdot 2^n + c_L) \bmod (2^n+1) \quad \text{其中，} c_0 \in \{0,1\},\ 0 \le c_L,\ c_H \le 2^n;$$

2^n +1 模修正算法为，当 $x=y=2^n$ 时，可知 $c_0=1$，$c_L=c_H=0$；由于 $2^n=(0-1) \bmod (2^n+1)=-1 \bmod (2^n+1)$，所以 $c=(-1)\times(-1)=1$。其余情况下（$c_0=0$），当 $c_L \ge c_H$ 时，为 $c=c_L-c_H$；当 $c_L<c_H$ 时，$c=c_L-c_H+2^n+1$。式中 c_H 和 c_L 分别为 n 比特乘法运算结果高低的 n 比特数据。因此，模 (2^n+1) 乘法可以通过一个 n 位无符号乘法器与两个 n 位加法器实现，如图 4.11 所示。

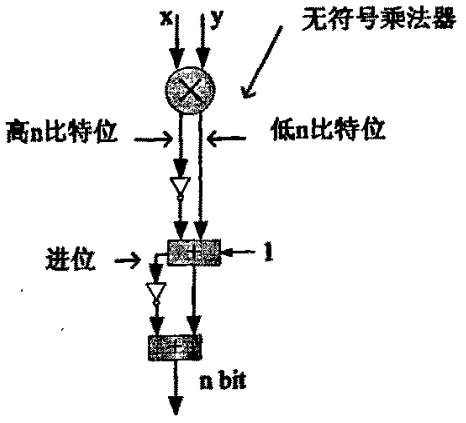


图 4.11 基于乘法器的模 2^n+1 乘法器结构图

4.4.2 电路设计

根据以上的分析可知，整个模乘运算电路需要由 4 个 16×16 乘法器以及 4 个 16 位加法器、2 个 2^{16}+1 模修正电路、1 个 2^{32}-1 模修正电路、2 个 4 选 1 的数据选择器组成。从功能上分析，该运算单元可实现 1 个 32bit 模 2^{32} 或 2^{32}-1 的乘法操作、同时进行 2 个 16bit 模 2^{16} 或 2^{16}+1 的乘法操作。表 4.6 定义了模乘单元功能参数接口。

表 4.6 模乘单元功能参数接口定义

功能参数接口	功能描述
MUL_mode[1:0]	00：模 2^{16} 乘法运算
	01：模 2^{16}+1 乘法运算
	10：模 2^{32} 乘法运算
	11：模 2^{32}-1 乘法运算

模乘运算电路结构如图 4.12 所示。图中，A_H 和 A_L 分别为 32 比特 A 的高低 16 比特；同样 B_H 和 B_L 分别为 32 比特 B 的高低 16 比特。MUX1 输出为模乘单元输出的低 16 比特；MUX2 输出为模乘单元输出的高 16 比特。

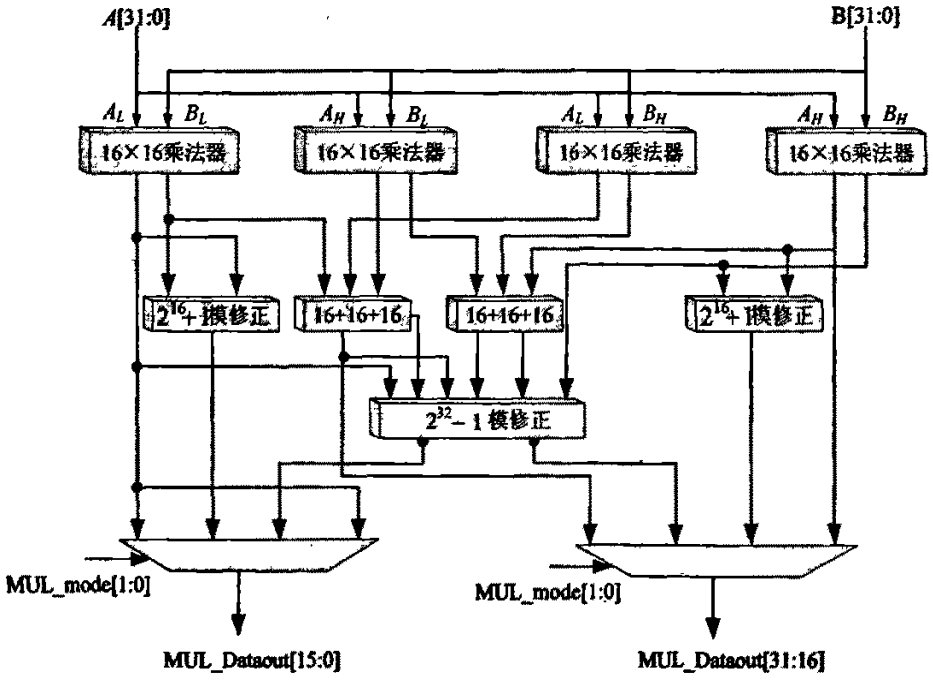


图 4.12 模乘单元电路结构

基于前面对乘法器设计的研究，本文在 16×16 乘法器电路设计上使用了如下优化方案。首先采用改进的二阶 Booth 编码来减少部分积的数目；其次利用 Wallace 树型结构来加快部分积加法运算的速度；最后采用超前进位加法器 CLA 计算最后乘积项加法的结果。该乘法器的电路结构如图 4.13 所示。

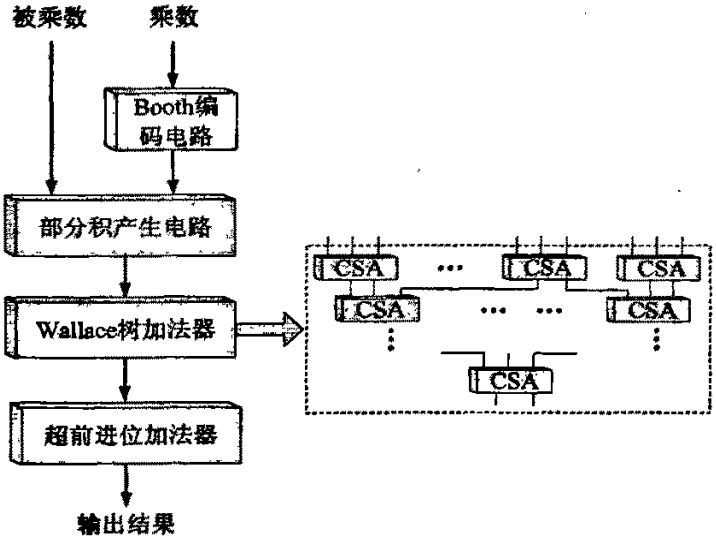


图 4.13 16×16 乘法器电路示意图

4.5 S盒替代单元研究与设计

4.5.1 实现原理研究

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乱作用。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$S(x)=(f_1(x),\cdots,f_m(x))$：$F_2^n \to F_2^m$，通常简称S是一个$n\times m$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[59]。

分组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；S盒的大小也很不相同，从512b~80kb。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。表4.7列出了不同S盒查找表模式的使用频率表。

表4.7 S盒查找表模式使用频度表

查找表模式	使用频度
8×8查找表	33.33%
8×32查找表	30%
4×4查找表	13.33%
6×4查找表	6.66%
其他模式	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

$x=\sum_{i=0}^{n-1}x_i 2^i=(x_{n-1},x_{n-2},\cdots x_1,x_0)_2$的$n\times m$S盒布尔函数的通式[48,49]为

$$f(x)=\sum_{i=0}^{2^n-1}c_i x_{n-1}^{i_{n-1}}x_{n-2}^{i_{n-2}}\cdots x_2^{i_2}x_1^{i_1}x_0^{i_0} \tag{4-5}$$

其中，系数 $c_i \in \{0,1\}$，$i=\sum_{k=0}^{n-1} i_k 2^k = (i_{n-1}, i_{n-2}, \cdots i_1, i_0)_2$。通项记为 $p_i(x)=\prod_{t=0}^{n-1} x_t^{i_t} = x_{n-1}^{i_{n-1}} x_{n-2}^{i_{n-2}} \cdots x_2^{i_2} x_1^{i_1} x_0^{i_0}$，则 $f(x)=\sum_{i=0}^{2^n-1} c_i p_i(x)$。从(4-5)式中可以看出当输入数据后，$p_i(x)$作为最小项就定下来了，对输出$f(x)$产生的影响的只有最小项系数$c_i$，通过改变$c_i$就可以使S盒实现任意的布尔函数的变换。

文献[27]和[50]详细论述了基于布尔函数理论实现任意n输入m输出的可编程 S 盒的设计方法。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实现较为繁琐，处理速度慢。更主要的是该方法对于实现$n\times m$的S盒所需要的配置编码为$m\times 2^n$比特，将随着输入n的个数成指数级增长。例如，对于常见8×8的S盒需要配置编码2048比特，这就造成了配置难度太大而难以实现。

2. 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$n\times m$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。

综合上述的两种实现方法，本文采用了基于查找表LUT实现方式。前面章节分析了，分组算法中常常需要并行查找相同或不同的S盒。而且使用S盒有两种方式：一种是每轮操作使用相同的S盒；另一种是每轮操作使用不同的S盒。因此，一方面为提高S盒替代电路的执行效率，在设计上使用了一种并行查找(Parallel Table Lookup PTLU)方案[51]。另一方面，考虑到在分组密码算法中经常会有多个不同的S盒查找表，在电路设计中采用了分页的设计方法。该方法将整个RAM根据不同的查找表模式分为多个不同的页面，每个页面就是一张完整的S盒查找表。在查表时通过对多个页面的调度实现了算法所需要的不同S盒替代。

4.5.2 电路设计

根据上面对S盒的设计研究，考虑到要实现的S盒替代单元输入n与输出m的位宽各不相同，因此电路采用RAM组的设计方式，即用两个$2^8\times 8$大小的RAM构成一个RAM组。本文在设计中使用了四个双端口$2^8\times 8$的RAM组并联电路，在每组RAM中划分不同的页，页面的多少与不同查找表的储存容量大小有关。每组RAM将4×4查找表划分为16页，6×4查找表划分为4页，8×32与8×8查找表划分为2页。在查表时，电路需要设置输入转换电路，将查表的输入数据按照不同查找表模式与页面控制信号装配成实际的查表数据，分别进入各个RAM组进行查表。每组RAM输出的2个8比特均分割为高低4

比特，再通过数据选择器来选择所需要的 8 比特输出。在查表前需要对 RAM 进行静态配置，将 S 盒代替表信息以 S 盒静态配置数据方式写入到 RAM 中，在时钟上升沿写入到 RAM 中。S 盒替代单元功能参数定义如表 4.8 所示。

表 4.8 S 盒替代单元功能参数接口定义

功能参数接口	功能描述
Sbox_conf_Data [31:0]	S 盒静态配置数据
Sbox_conf_addr[7:0]	S 盒静态配置地址
Sbox_ren	S 盒查表使能
Sbox_type[1:0]	00：8×8 查找表模式
	01：8×32 查找表模式
	10：4×4 查找表模式
	11：6×4 查找表模式
Sbox_off[3:0]	S 盒页面编号（地址偏移量）

S 盒替代电路如图 4.14 所示，电路整体由配置地址生成电路、页面地址调度电路、输出变换电路以及 4 个并行 RAM 组构成，其中每组 RAM 由两个 $2^8\times8$ 的 RAM 构成，电路使用了总体存储容量为 16K。当对 S 盒进行静态配置时，32 比特配置数据从高位至低位依次分为四个 8 比特数据写入到四个 RAM 组中。每个 RAM 组的 8 比特配置地址即写地址由配置地址生成电路产生，送至各个 RAM 组。当进行 S 盒查表时，由页面地址调度电路产生各个 RAM 组所需的 16 比特读地址，再依次按从高至低输入到相应的 RAM 组中。每个 RAM 组查表读出的 2 个 8 比特数据输入到输出变换电路以产生最后的 32 比特输出结果。

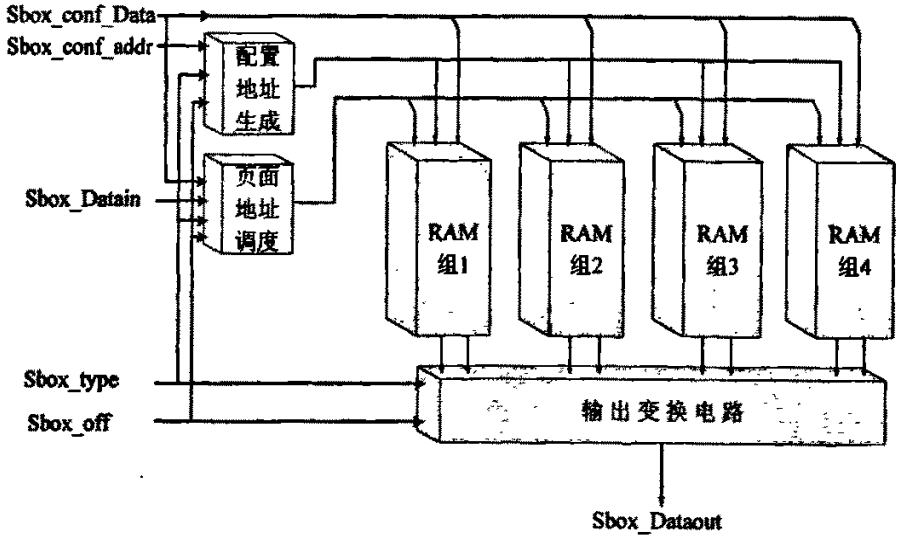


图 4.14 S 盒替代单元电路结构图

由于 S 盒替代单元在静态配置和查表时需要支持小于 8 比特输入的 4×4 查找表模式和 6×4 查找表模式。对于 2^8×8 的 RAM，其读写地址由于小于 8 比特，因此其高位需要页面号来填补构成 8 比特的读写地址。配置地址生成模块以及页面地址调度模块则对不同类型的查找表进行地址填充以及根据类型输出相应的读写地址进入到各 RAM 组中。电路主要由填充电路以及 4 选一的 8 比特位宽的数据选择器来实现。每个 RAM 组输出 2 个 8 比特的数据，为了实现多种不同输出的查找表模式，输出变换电路将这每个 8 比特数据，分为高低 4 比特进行选择输出，最后从每组 RAM 的 16 比特输出中选择需要的 8 比特页面信息输出。这样满足了不同 S 盒查表模式的页面输出需要。该电路再实现上主要采用了两个 4 比特位宽的 2 选 1 数据选择器将输出相应的 8 比特查表结果输出。

4.6 置换单元研究与设计

4.6.1 实现原理研究

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。例如：在 DES 中，有六种不同种类的置换，Twofish 中有两种，SERPENT 中也有两种。表 4.9 为一些常见分组密码算法中用到的置换。

表 4.9 主要分组密码算法置换种类表

置换种类	输入位宽	输出位宽
DES：P 置换	32	32
DES：E 置换	32	48
DES：初始终结置换	64	64
LOKI91：E 置换	32	48
LOKI91：P 置换	32	32
LOKI97：P 置换	64	64
SERPENT：初始置换	128	128
SERPENT：终结置换	128	128
DES：子密钥扩展置换	64	56
DES：子密钥扩展置换	56	48

在对密码学中的置换运算单元研究中发现它和交换网络功能非常相似，因此可以用分

组密码算法中的大数据位宽的比特操作来代替 Crossbar 交换网络。即置换单元不仅可以完成密码算法中的置换操作同时也可实现 RCPA 中的互连网络 ICM 功能。根据重构元素的最大适应性原则，一个 N×N 的置换单元应该实现 N 输入、N 输出的所有的选择变换，即 N×N 置换单元的 N 个输出中的任何一个能够选择 N 个输入中的任何一个。按照该原则设计的可重构比特置换单元所需要的配置编码的宽度和它所能实现的选择变换的个数可由下述定理描述：

对于 N 比特置换，需要从的结果空间 $N!$ 中挑选出来一个作为结果，因此至少需要 $log_2(N!)$比特作为置换操作指定。可以证明：$N!=O(N^N)$，$log_2(N!)=O(Nlog_2N)$[52]。这意味着用于指定一个置换的比特数是 $Nlog_2N$。设一个 N×N 置换单元能够实现其输入到输出的所有选择变换，即其 N 个输出中的任何一个能够选择 N 个输入中的任何一个，则该置换单元需要 $Nlog_2N$ 位配置编码，它能够实现的选择变换的个数为 N^N，即 N 个 N 选 1。

文献[27]和[50]给出了实现 N×N 的任意置换方法，该方法采用了 N 个 N 选 1 数据选择器来实现。该方案能够实现的选择变换个数为 N^N，需要配置编码为 $Nlog_2N$ 位。此方法易于实现，但是其占用硬件资源过大，可以计算出 N 选 1 数据选择器相当于 log_2N 级 N-1 个 2 选 1 数据选择器，因此共需要 N×(N-1)个 2 选 1 数据选择器。此外，该方案需要的配置编码过长，例如 128×128 的置换需要 896 比特编码宽度。

文献[57]借鉴通信交换网络方面的研究方法和系统架构来实现密码学中的比特置换，提出了一种合适密码算法中比特置换的网络，对于 N×N 的任意比特置换，这种网络的硬件实现在不改变功能的前提下，要比前面提到的用 N 个 N 选 1 的数据选择器实现降低面积，减少配置编码长度。文献[57]选取了 BENES 网络来设计 N×N 的任意置换，其在实现上需要 $2Nlog_2N-N$ 个 2 选 1 数据选择器，与使用 N 个 N 选 1 的数据选择器需要的 N×(N-1)个 2 选 1 数据选择器相比极大降低了硬件资源。此外该网络的配置编码需要（$2log_2N-1$）×N/2 比特位宽，与前一方案相比配置编码得到了减少。表 4.10 列出了两种方案实现 128×128 置换的对比情况。通过比较不难看出，在实现上选用文献[57]提出的方案是较好的选择。

表 4.10 两种方案实现的 128×128 置换比较表

实现方法	2 选 1 数据选择器数目	配置编码长度
文献 57	1664	832
文献 3 及 51	16256	896

4.6.2 电路设计

比特置换单元整体电路结构如图 4.15 所示。主要由两部分电路构成，分别为比特置换网络和静态配置寄存器堆。比特置换网络是整个单元的核心，本文采用了文献[57]所提出的利用 BENES 交换网络实现任意比特置换的方法来设计可重构的置换运算单元，该电路能实现 128 比特位宽的任意置换。用户可以根据需要对该网络进行配置，实现所需要的置

换。静态配置寄存器堆对配置信息进行存储，并完成对比特置换网络的配置。BENES 网络的配置信息算法由软件实现，生成相应的置换网络的配置信息后，再以静态配置指令的方式注入到电路的静态配置寄存器堆。电路在功能上可实现 4 个 32×32 置换、2 个 64×64 置换、1 个 128×128 置换。

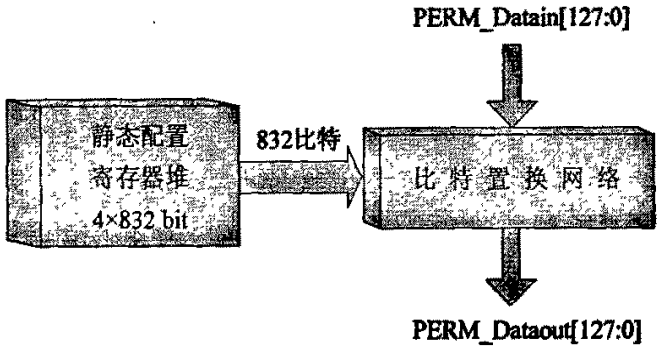


图 4.15 比特置换单元整体结构图

表 4.11 对比特置换单元功能参数接口的定义进行了说明。

表 4.11 置换单元功能参数接口定义

功能参数接口	功能参数描述
PERM_conf_data	静态配置数据输入
PERM_conf_wen	静态配置寄存器写使能，静态配置使能
PERM_conf_addr	配置寄存器读写地址
PERM_conf_ren	静态配置寄存器读使能，动态配置使能

由于密码算法中大都置换运算的变化频率较少，而且一个 128 比特位宽的置换网络需要长达 832 比特的配置信息，因此在比特置换单元设计上设置了静态配置寄存器堆。在系统运算前，将算法需要的配置信息以静态配置方式写入到配置寄存器堆中。这样在执行密码算法时，通过对寄存器堆的读地址动态配置使配置信息读入到置换网络中，从而完成所需的置换运算。本文在设计上使用了 4 组 832 比特的寄存器堆，可支持 4 种不同的置换运算。

置换网络内部电路结构如图 4.16 所示，电路依据文献[57]提出的采用 BENES 网络实现方案。整个网络由 13 级开关构成，每级分为一列共 64 个 2×2 开关及其后的连线。13 级网络共有 13 列 13×64 个开关及 13 级连线。每个开关由两个数据选择器构成，由静态配置寄存器堆送入配置信息 Perm_con 控制其数据的选通，每个开关对应 1bit 控制信息，当其为 1 时，数据交叉通过开关，当其为 0 时，数据直接通过开关，每级置换网络需要 64 比特的配置信息，整个网络共需 832-bit 控制信息。

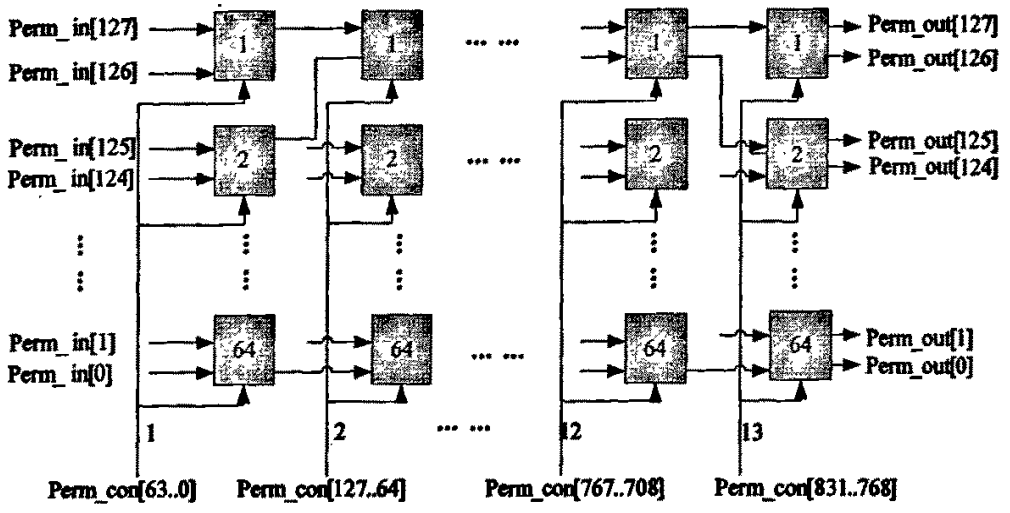


图 4. 16 BENES 网络置换电路结构

4.7 本章小结

可重构密码处理元素 RCE 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。本章在可重构密码处理模型 RCPA 基础上，对可重构密码处理元素 RCE 硬件实现原理进行了研究和设计。重点介绍了移位单元、有限域 GF(2^8)乘法单元、模加/减运算单元、模乘运算单元、S 盒替代单元以及置换单元等可重构密码处理元素 RCE 的设计原理和方法。这些基本运算元素可根据 RCE 配置指令进行重构，灵活完成不同算法所需的运算功能。

可重构密码处理元素 RCE 设计工作主要包括了：采用桶形移位器的设计原理，实现了 8 比特、16 比特、32 比特以及 64 比特位宽的任意移位；利用“Shift-and-add”算法实现了 GF(2^8)域上的可配置不可约多项式和乘数多项式的有限域 GF(2^8)乘法；完成了可重构模 2^8、2^{16}、2^{32} 的加/减以及 PHT 变换电路的设计；实现了可重构模 2^{16}、$2^{16}+1$、2^{32}、$2^{32}-1$ 乘法；完成了可配置 S 盒替代电路的设计，支持了 8×8、8×32、6×4、4×4 等不同类型的 S 盒；采用文献[57]提出的 BENES 网络结构模型设计了置换单元，可实现 32-32、64-64、128-128 的任意比特置换。

第五章 RCE 实现与性能分析

本文采用 VerilogHDL 硬件描述语言完成了 RCPA 中的可重构密码处理单元 RCE 的 RTL 级模型的建立，同时编写了测试激励。在 Mentor 公司的 ModelSim6.0 仿真环境下进行功能仿真，并采用 Altera 公司的 QuartusII6.0 软件进行了综合、布局布线以及 FPGA 验证。为了更好的评估其性能以及考虑其 ASIC 实现，本文又采用了 Synopsys 公司的 Design Compiler 工具在 0.18μm CMOS 工艺标准单元库下进行了逻辑综合。

本文选用 StratixII 系列中的 EP2S180F1020I4 芯片作为 FPGA 目标器件，它的规模为 143,520 个自适应查找表(ALUT)等效为 179,400 个逻辑单元(LE)。Altera 推出的 StratixII 系列 FPGA 基于 1.2V、90nm 工艺，具有多达近 18 万个等效逻辑单元、9MB 嵌入式存储单元、高性能的嵌入数字信号处理(DSP)块和高带宽 I/O 能力[53]。本章首先介绍了 RCE 在 FPGA 下的实现，而后进一步分析了 RCE 基于 ASIC 实现的性能。

5.1 RCE 单元基于 FPGA 的综合

本文使用 Altera 公司的 StratixII 系列 FPGA 中的 EP2S180F1020I4 芯片作为目标芯片，使用 QuartusII6.0 工具进行代码编译、逻辑综合，布局布线。同时使用 Mentor 公司的 ModelSim 6.0 进行功能仿真以及时序仿真。在仿真验证过程中，将各 RCE 运算单元的运算数据以及功能参数配置信息通过编写 Testbench 文件方式输入，最终得到的仿真结果与预期的实现结果相一致，验证了各 RCE 数据通路以及配置控制功能的正确性。表 5.1 给出了各单元基于 FPGA 的实现性能。

表 5.1 RCE 单元基于 FPGA 实现性能

RCE 单元	最大路径延迟	占用资源		
		逻辑单元(ALUT)	存储单元(bits)	DSP
移位单元	12.983 ns	660	0	0
有限域乘法单元	19.887 ns	794	683	0
模加/减单元	11.599 ns	430	0	0
模乘单元	7.881 ns	234	0	8
S 盒单元	8.439 ns	96	16384	0
置换单元	23.969 ns	2500	3328	0
逻辑运算单元	7.025 ns	32	0	0

从表中可以看出本设计经 QuartusII6.0 综合及布局布线后，有限域乘法单元和置换单元中的静态配置寄存器堆以及 S 盒单元中的 RAM 在 EP2S180F1020I4 器件内部由内置的存储单元来实现；模乘单元中的乘法器电路则采用了 EP2S180F1020I4 器件内部的高速嵌入

数字信号处理 DSP 模块内部的 9×9 乘法器实现。

5.2 RCE 模块基于 ASIC 的综合

上一节我们对各个 RCE 单元模块成功进行了仿真验证，并在 Altera 的 EP2S180F1020I4 器件上成功测试了各 RCE 单元模块功能正确性。为了更准确的评估其性能以及完成 RCE 单元向 ASIC 的移植工作，本文基于 0.18μm CMOS 工艺标准单元库对 RCE 进行逻辑综合，综合工具使用 Synopsys 公司的 Design Complier for Solaris。表 5.2 给出了各 RCE 单元综合结果报告。从上面的综合报告可以看出 RCE 单元的延迟均小于 5ns，因此可以得出可重构密码处理单元 RCU 的时钟周期长度最短可定为 5ns，即时钟频率最高为 200Mhz。

表 5.2 RCE 单元在 DC 上的综合结果

RCE 单元	面积	等效门数	最大路径延迟
移位单元	19286	1929	1.64 ns
有限域乘法	85473	8547	4.22 ns
模加/减单元	12826	1283	3.56 ns
模乘单元	127260	12726	4.62 ns
S 盒替代	1160372	116037	1.63 ns
置换单元	49816	4982	2.3 ns
逻辑运算单元	3426	343	1.36 ns

5.3 RCE 单元性能分析

在不同类型的可重构密码处理系统中，可重构处理元素的电路面积和运算速度有着较大的差别。而对可重构处理元素的评价单一的从运算速度或者电路面积去衡量一个设计的性能都是不全面的，所以通常以面积与速度的乘积作为比较全面的衡量设计性能优劣的标准，乘积数值越小表明设计的性能越理想。时间面积积通常是指在处理数据长度固定条件下，完成运算所需时间与电路面积的乘积。下面给出了时间面积积的计算公式：

$$\text{时间面积积} = \text{面积（门数）} \times \frac{\text{数据长度}}{\text{时钟频率} \times \text{单位处理数据量}}$$

本文选择了文献[27][50]提出的 RELOG_DIGG 系统的重组元素以及文献[21]COBRA 可重构元素与本设计进行性能比较。图 5.1 给出了本文设计的 RCE 与这两种系统的重构元素性能比较。为了便于比较，本文在时间面积积的计算上不同的单元选择了不同长度的运算数据。例如移位单元和模加/减选择 3.2kbit 长度的处理数据，而 S 盒则选择 32bit 长度处理数据进行横向比较。这里需要说明的是由于不同系统中的可重构元素在功能上有所不同，因此性能比较的数据是在这些可重构元素具有的相同功能基础上得出的。

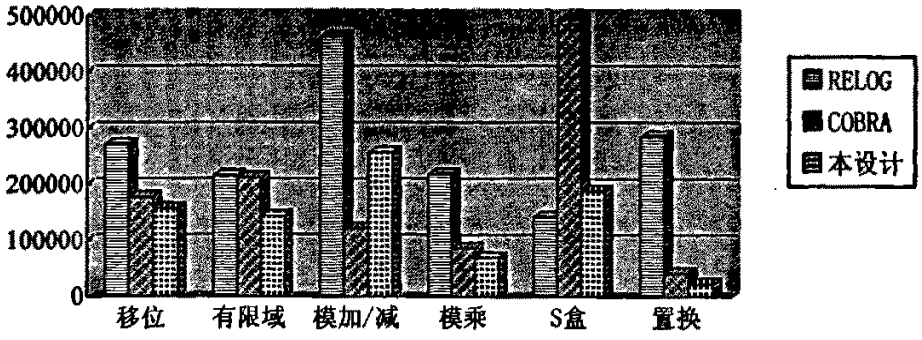


图 5.1 可重构元素时间面积积性能比较

从§5.2 节的综合结果以及图 5.1 的比较可以看出本文的 RCE 设计与当前其他可重构密码系统的可重构元素相比有两个方面的优势：一是在于具有较优的性能，电路延迟较小，支持可重构密码处理的高速实现，而且电路规模适中；二是具有较高的灵活性，可以通过配置不同的功能参数满足绝大多数密码算法的需求。

5.4 本章小结

可重构密码处理单元 RCE 采用 VerilogHDL 硬件描述完成了 RTL 级模型的建立。本章首先给出了可重构密码处理元素 RCE 在 Altera 公司 StratixII 系列中的 EP2S180F1020I4 下的实现性能。为了更好的评估其性能以及考虑其 ASIC 移植工作，本文对 RCE 采用 Synopsys 公司的 Design Compiler 工具在 0.18μm CMOS 工艺标准单元库下进行了逻辑综合，给出了基于 ASIC 的综合结果以及对其性能进行了分析和比较。经 FPGA 验证以及 ASIC 的综合结果表明 RCE 在设计上不仅具有较优的性能还具有较高的灵活性，满足了绝大多数分组密码算法的需求。

第六章 RCPA 上的算法映射与实现验证

通过第二章对分组密码算法的分析可知 DES 算法是一种具有典型 Feistel 结构模型的分组密码算法，而 Rijndael 算法则是 SP 网络结构的分组密码算法的代表算法。因此，通过研究和分析 DES 算法和 Rijndael 算法，并把该两种典型算法成功映射到 RCPA 上，对于我们验证该模型能否对于 Feistel 和 SP 网络结构模型的分组密码算法具有较好的适应性具有重要意义。因此，本文以 DES 算法以及 Rijndael 算法为例，讨论分组密码算法在 RCPA 上的映射，并对其性能进行分析与评估。最后给出了我们基于 RCPA 模型的一种验证原型的实现结果。

6.1 RCPA 上分组密码算法映射举例

分组密码算法在 RCPA 上的映射比较简单易于实现，其映射特点如下：

1. 分组密码算法的初始变换和末变换通常计算规模不大，可由一级 RCU 单元或 ICM 模块来完成。

2. 分组密码的每轮变换中的一步或多步运算可在一级 RCU 完成，同级 RCU 的在无数据相关的条件下可并行执行。

3. RCPA 可根据需要对算法的流水级重构，算法的每步运算操作可化成单向流形式直接对应 RCPA 的可重构流水线，同级流水线的操作在映射上没有数据相关性，布局简单。

4. RCPA 上对算法的分组拆分具有固定的模式，通常将大的分组数据拆分为 32 比特字块，输入到同一级 RCU 中进行算法映射。

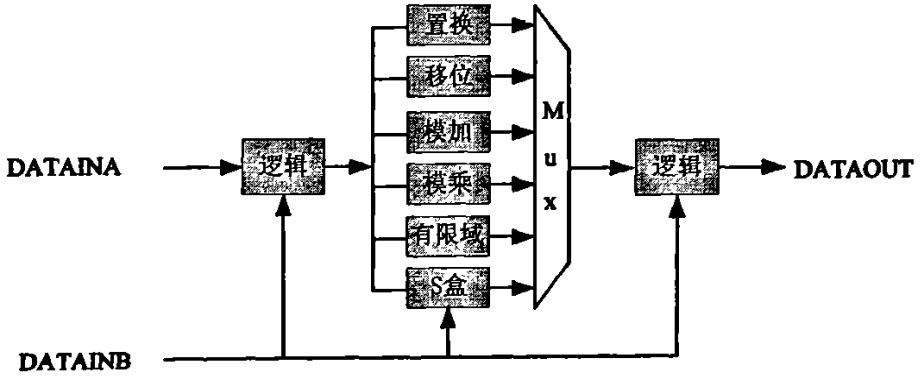


图 6.1 本文选用的 RCE 串并混合连接方式

为了便于说明算法的映射过程，这里先定义 RCPA 模型中的几个特征参数：每行 RCU 的个数 R 为 4，RCU 的行数 C_r 设为 4，ICM 的个数 C_i 设为 4。此外，在 RCU 内部的可重构处理元素 RCE 连接方式上，选择一种简单的串并混合连接。该连接方式如图 6.1 所示。由于逻辑运算单元是密码运算中最常用的运算单元，利用率很高，所以在 RCU 内部设置了两个逻辑运算单元分别置于 RCU 的前部和后部，而将其他运算单元并行连接，对不需要的运算单元可进行旁路处理。这样极大提高了 RCU 内部运算的灵活性。由于逻辑运算

单元本身电路延迟很小，所以采用该结构，对电路 RCU 不会产生较大的延迟。

6.1.1 DES 算法在 RCPA 上的映射

DES 算法是 Feistel 模型的典型代表，它以 64 比特为分组对数据加密，通过一个初始置换将明文分组分成左半部分和右半部分，各 32 位长。然后进行 16 轮完全相同的运算，这些运算称为函数 f，在运算过程中数据与密钥结合。经过 16 轮后，左右半部分合在一起经过一个末置换，这样算法就完成了。函数 f 包括了扩展置换将数据的右半部分扩展成 48 位，并通过一个异或操作与 48 位密钥结合，通过 8 个 S 盒将这 48 位替代成新的 32 位数据，再将其置换一次[30]。

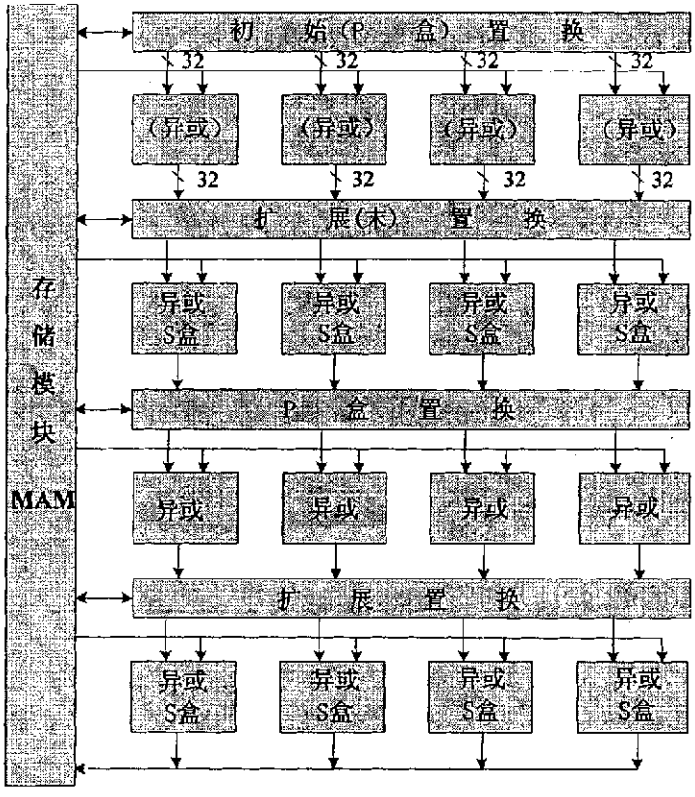


图 6. 2 DES 算法 RCPA 上的加解密映射图

DES 算法 RCPA 上的加解密映射过程如图 6.2 所示。其映射过程是：由于 DES 算法是 64 比特的明/密文输入，所以在该架构中可以同时并行处理 DES 的 2 个分组数据。根据算法，128 比特明/密文数据先进入第一级 ICM 互连模块，进行 2 组初始 64 比特置换；接下来将第一级 RCU 旁路，将置换后的数据输入至第二级 ICM 互连模块用来完成 32 位到 48 位的扩展置换；接下来第二级 RCU 将完成与密钥的异或运算同时完成 S 盒代替运算；最后第三级 ICM 和第三级 RCU 执行 P 盒置换及异或运算；这样 DES 算法的一轮运算就全部完成了。RCPA 的第四级 ICM 和第四级 RCU 将完成下一轮的扩展置换、异或运算以及 S 盒代替运算。当 RCPA 全部执行轮运算时，第一级 ICM 将执行 P 盒置换同时第一级 RCU 将做异或处理。轮运算结束后，第二级 ICM 将完成最后的末置换。

DES 算法在 RCPA 上的流水线可配置为 6 级：第一级 ICM 和第一级 RCU 构成第一级流水线，完成初始置换或 P 盒置换及异或操作；第二级流水线由第二级 ICM 构成，完成扩展置换或末置换操作；第二级 RCU 构成第三级流水线，完成异或和 S 盒操作；第四级流水线由第三级 ICM 和第三级 RCU 构成，完成 P 盒置换及异或操作；第四级 ICM 构成第五级流水线，完成扩展置换操作；最后一级流水线由第四级 RCU 构成，完成异或和 S 盒操作。DES 算法在 RCPA 上流水线工作的时空图如图 6.3 所示。

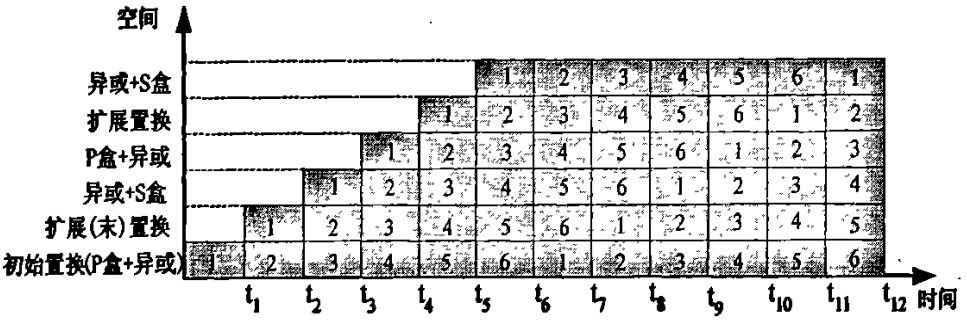


图 6. 3 DES 算法在 RCPA 上流水线工作的时空图

6.1.2 Rijndael 算法在 RCPA 上的映射

Rijndael 算法是 SP 网络的典型代表，它支持以 128、192、256 比特为分组对数据加密。算法核心由三部分组成：初始圈密钥加法，圈变换和末尾圈变换。初始圈密钥加法是将输入的明密文与初始子密钥进行异或。算法中除了末尾圈变换省略列混合变换外，每圈变换包含字节代替变换、行移位变换、列混合变换和圈密钥加法。Rijndael 算法的圈数随着密钥长度的不同而变化，下面以分组长度和密钥长度都是 128 位圈数为 10，进行映射描述。

Rijndael 算法 RCPA 上的加解密映射过程如图 6.4 所示。其映射过程是：首先将第一级 ICM 互连模块进行旁路，128 比特明/密文数据进入第一级 RCU，与初始密钥执行异或操作完成初始圈密钥加法，随后在 RCU 内进行 S 盒查表操作；接下来的第二级 ICM 互连模块用来完成行移位运算；而后第二级的 RCU 将使用有限域乘法元素完成列混合操作同时完成异或运算；这样 Rijndael 算法的一轮运算就全部完成了。接着将 RCPA 的第三级 ICM 进行旁路，数据进入第三级 RCU 和第四级 ICM 中执行 S 盒替代和行移位操作；最后第四级 RCU 将完成列混合操作和异或运算。当 RCPA 进行初始圈密钥加法时，第一级 RCU 需要执行异或操作；当 RCPA 执行圈变换时，第一级 RCU 只需要完成 S 盒替代；当 RCPA 执行末尾圈变换时，第四级将 RCU 不执行列混合操作只进行异或运算。

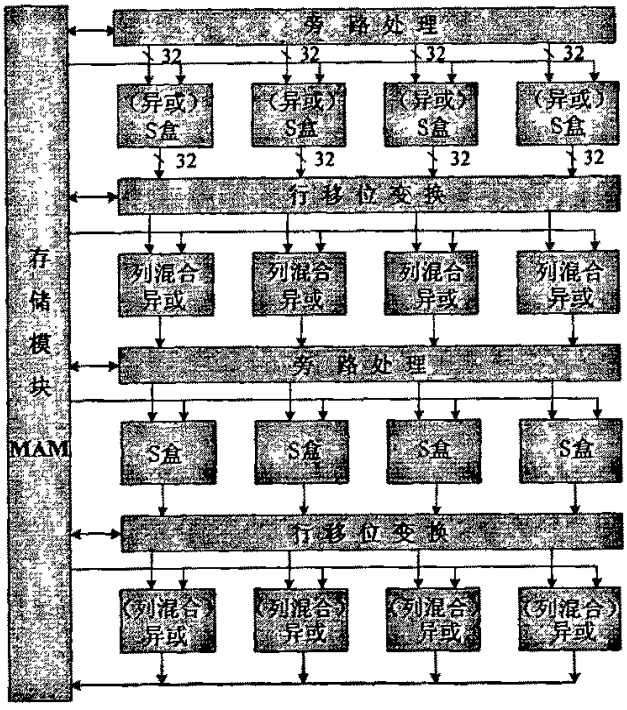


图 6.4 Rijndael 算法 RCPA 上的加解密映射图

Rijndael 算法在 RCPA 上的流水线可配置为 4 级：第一级 RCU 和第二级 ICM 构成第一级流水线，完成异或、S 盒替代和行移位操作；第二级流水线由第二级 RCU 构成，完成列混合和异或操作；第三级流水线则由第三级 RCU 和第四级 ICM 构成，完成 S 盒替代和行移位操作；最后一级流水线由第四级 RCU 构成，完成列混合和异或操作。Rijndael 算法在 RCPA 上流水线工作的时空图如图 6.5 所示。

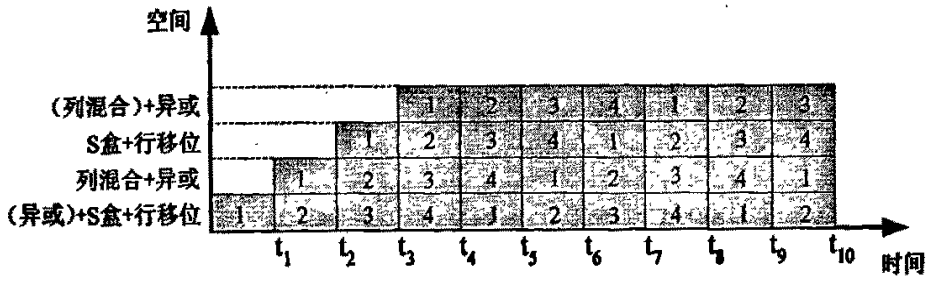


图 6.5 Rijndael 算法在 RCPA 上流水线工作的时空图

6.2 RCPA 算法映射性能分析

分组密码算法在 RCPA 上映射的性能指标可用吞吐率(Throughput)来表示，那么 RCPA 上算法映射的吞吐率可定义为

$$\text{吞吐率} = \frac{128\text{比特分组长度} \times \text{实际映射流水线级数}}{\text{每个分组运算周期数}} \times \text{时钟频率} \quad (\text{bits/s})$$

通常把吞吐率换算为 Mbits/s 的单位。根据前面 RCPA 的实现性能，本文以 RCPA 在

180MHz 频率下计算其运算吞吐率。由于 RCPA 的算法映射吞吐率与模型中的结构特征参数有着直接的联系。因此需要在不同的参数定义下分析其性能。表 6.1 给出了几种常见的分组密码算法在 RCPA 上的处理性能。表中结构类型以 R-C_r-C_l表示，其中为每行 RCU 的个数；C_r为 RCU 的行数；C_l为 ICM 的个数。当 R=4 时，RCPA 可同时处理 2 组以 64 比特为分组的算法，而对于 RC6 和 Twofish 等算法也可同时进行 2 个分组运算。

表 6.1 RCPA 上的算法映射性能

算法	结构类型	运算周期数（Clocks/Block）	吞吐率（Mbits/s）
DES	4-4-4	8.33	2764.8
	4-2-2	16.66	1382.4
	4-1-1	50	460.8
Rijndael	4-4-4	4.83	4767
	4-2-2	9.66	2383
	4-1-1	29	794
IDEA	4-4-4	14.25	1616.84
	4-2-2	28.5	808.42
	4-1-1	57	404.21
RC6	4-4-4	30.5	1510.82
	4-2-2	61	755.41
	4-1-1	122	377.71
Twofish	4-4-4	28	1616.84
	4-2-2	56	808.42
	4-1-1	112	404.21
SAFER+	4-4-4	24	960
	4-2-2	48	480
	4-1-1	96	240
SERPENT	4-4-4	48.5	475.04
	4-2-2	97	237.52
	4-1-1	194	118.76
CRYPTON	4-4-4	4.75	4850.56
	4-2-2	9.5	2425.28
	4-1-1	38	606.32
CAST-128	4-4-4	20	1152
	4-2-2	40	576
	4-1-1	80	288

从表 6.1 可以看出在 RCPA 上执行的分组密码算法都可达到较高的性能。每种分组密码算法在 RCPA 上使用的流水线级数可以有不同的配置。与文献[21]提出的 COBRA 系统

相比，RCPA 的性能平均提高了 1.3 倍以上，而且可支持实现 DES 算法以及 IDEA 算法。RCPA 与文献[27]提出的 RELOG_DIGG 系统相比，RCPA 可以支持更广泛的各种结构的分组密码算法，且性能提高了 5 倍以上。RCPA 性能与软件实现比较提高了 4.8~51.8 倍，其中与具有高主频的通用微处理器（Pentium4 2.1GHz）相比，RCPA 的性能对于大多数分组密码算法可提高 10~20 倍左右。图 6.6 给出了结构类型为 4-1-1 下的 RCPA 的性能比较图。在图中的软件实现中，DES、IDEA、RC6、Rijndael、Twofish 和 SERPENT 算法选择了 Crypto++ v5.1 Benchmarks 在 Pentium4 2.1GHz 的下的公开测试结果[54]；SAFER+和 CRYPTON 算法选择了文献[55]和[56]基于 NIST AES 分析平台在 Pentium Pro 200Mhz 的公开测试结果。其他硬件选取具体为 DES 以及 IDEA 为 RELOG_DIGG 性能[27]；RC6、Rijndael 和 Twofish 为 COBRA 系统在 SP-1-1 架构下的性能[21]；SERPENT 和 CRYPTON 为 RHCA 系统实现性能[28]。

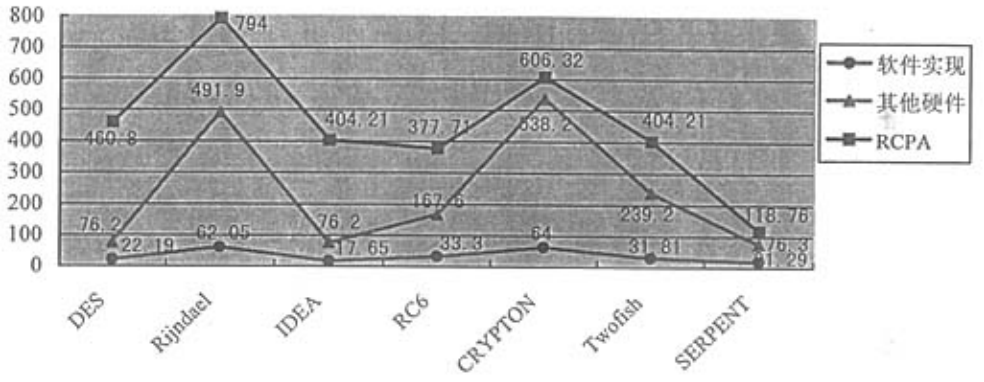


图 6.6 RCPA 算法映射性能比较(Mbits/s)

6.3 基于 RCPA 的一种验证原型设计实现

6.3.1 验证原型设计

基于前面对 RCPA 的结构分析、算法映射以及性能分析后，本文最后基于 RCPA 设计了一种验证原型。该原型数据通路的特征结构采用了 4-1-1 类型，RCU 内部的可重构处理元素 RCE 连接方式上选择了 § 6.1 节所示的串并混合连接方式。

验证原型的计算模型采用 VLIW/EPIC 结构，这种结构硬件控制相对简单，计算密集型应用内在并行性很明显，且不需很多转移预测，在这种结构上运行能够达到较高的实际指令级并行度。正是由于以上特点，VLIW/EPIC 结构在很大程度上支持了 RCPA 模型的处理需求——横向并行且纵向流水执行。在专用指令集设计中，本文借鉴了 Intel 安腾处理器的 EPIC 指令集设计思想，紧密结合 RCPA 自身结构，定义了专用指令集。由不同类型指令的重组可并行、高效的实现多种分组密码算法，显著提高了芯片自身硬件资源的利用率及芯片的密码运算效率。该专用指令集将指令分为了静态配置指令、短指令、长指令、超

长指令以及控制指令等5种类型。指令长度设为192比特，指令RAM地址设为11比特，存储容量为$2^{11}\times192$比特。

验证原型的存储模块MAM采用了寄存器堆实现，根据应用将其分为了通用寄存器堆和密钥寄存器堆两个部分，存储容量共计 5120 比特。其中通用寄存器的作用主要是保存密码算法的主密钥、中间计算结果和运算所需的常数值，是数据传输与密码运算的中转站，并且各种密码运算的最终结果在运算完成后都必须先存入通用寄存器，再由专用数据传输指令将其存入输出寄存器等待输出，存储容量设为32×32（深度×位宽）比特即1024比特；密钥寄存器堆依据分组密码运算的特点设计，通常只用来存储轮子密钥与常数值。密钥寄存器堆物理上分为4块，每块存储容量为32×32（深度×位宽）比特，因此存储容量为4096比特。密钥寄存器堆具有4个读端口和4个写端口，对应于每块有一个固定的读端口和一个写端口。若算法所需要的密钥存储容量大于4096比特时，如SERPENT算法需要4224比特的密钥，可将剩余的密钥存入到通用寄存器中，这样就满足了分组密码算法的密钥存储需求。

验证原型设置了输入/输出接口单元作为数据缓冲区，主要包括输入寄存器和输出寄存器。输入寄存器为输入数据的缓冲区，输入数据包括明/密文分组、主密钥、IV 向量等。经过对常用分组密码算法的分析，可知分组长度一般为64/128/256位，因此可将输入寄存器设计为 16×32=512 位，既满足现有分组密码算法需要，又保留一定的扩展空间。输出寄存器的设计与输入寄存器相同，它是输出数据的缓冲区，其容量也是根据分组长度而定的，所以容量仍设为512位。

6.3.2 实现结果

验证原型采用Verilog语言描述，通过Altera公司的QuartusII6.0软件进行了综合以及布局布线，最后下载至StratixII的EP2S180F1020I4器件上得到了正确的验证。RCPA验证原型基于FPGA实现性能如表6.2所示。从表中可以看出验证原型中的RCU单元内部的静态配置寄存器堆、S盒单元中的RAM、指令RAM以及内部的寄存器堆均使用了器件内部的存储单元来实现；而RCU单元中的乘法器内部的高速嵌入数字信号处理DSP模块内部的9×9乘法器实现。

表6.2 验证原型基于FPGA实现性能

器件型号	最高时钟频率	占用资源		
		逻辑单元(ALUT)	存储单元(bits)	DSP
EP2S180F1020I4	47.74 MHz	32863	458752	32

为了准确的评估 RCPA 验证原型的实现性能以及完成最终的 ASIC 实现。我们使用Synopsys公司的Design Complier for Solaris工具，采用0.18μm CMOS 工艺标准单元库及相应负载模型和RAM硬核对分组密码协处理器进行逻辑综合，综合报告如表6.3所示。

表 6.3 验证原型基于 ASIC 实现性能

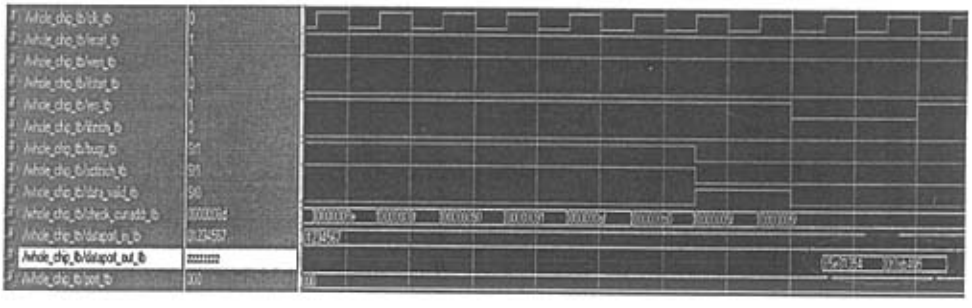
面 积	等效门数	最大路径延迟	最高时钟
14899092	148.99 万门	5.55ns	180MHz

6.3.3 仿真验证

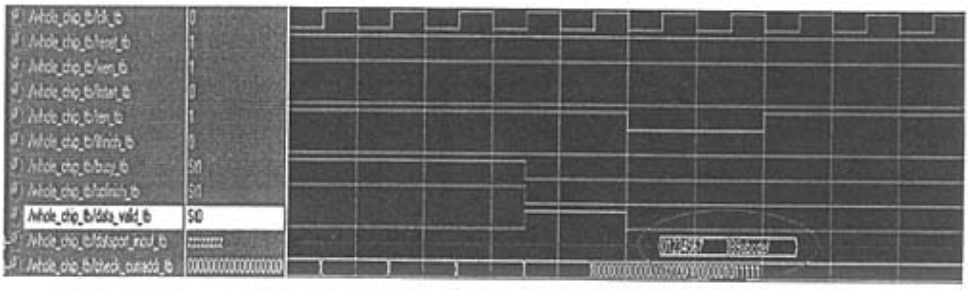
为测试 RCPA 验证原型的正确性，本文使用了多种分组密码算法对该原型进行验证，结果显示均正确，验证算法包括了 DES、3DES、AES-128、AES-192、AES-256、IDEA、Twofish、SAFER+、RC6、SERPENT。图 6.7 给出了 DES 以及 AES-128 算法 ECB 模式在 ModelSim 6.0 下的仿真结果。

DES 算法 ECB 模式加密仿真结果如图 6.7(a)所示：明文为 0x01234567，0x89abcdef，密钥为 0x13345779，0x9bbcdff1，加密结果 0x85e81354，0x0f0ab405。DES 算法解密仿真结果如图 6.7(b)所示：将上述加密结果进行解密，还原出明文 0x01234567，0x89abcdef。

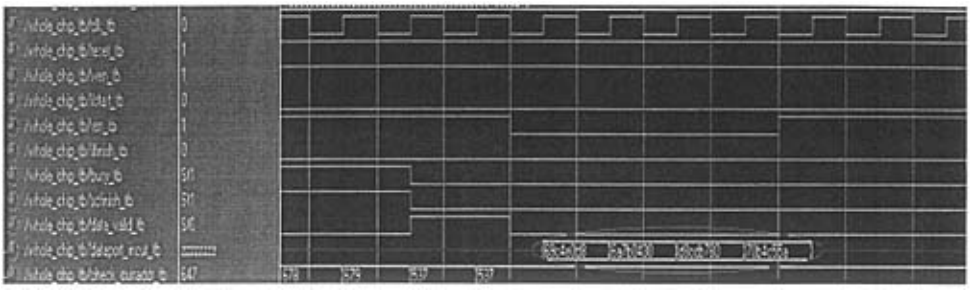
AES-128 算法 ECB 模式加密仿真结果如图 6.7(c)所示：密钥为 0x00010203，0x04050607，0x08090a0b，0x0c0d0e0f，明文为 0x00112233，0x44556677，0x8899aabb，0xccddeeff，加密结果 0x69c4e0d8，0x6a7b0430，0xd8cdb780，0x70b4c55a。AES-128 算法解密仿真结果如图 6.7(d)所示：将上述加密结果进行解密，还原出明文 0x00112233，0x44556677，0x8899aabb，0xccddeeff。



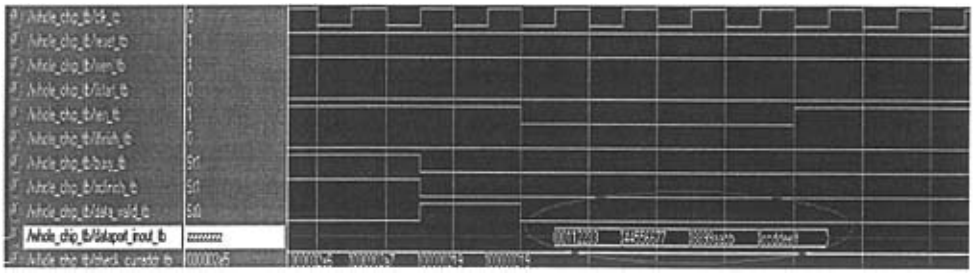
(a) DES加密仿真结果



(b) DES解密仿真结果



(c) AES-128加密仿真结果



(d) AES-128解密仿真结果

图 6.7 DES 以及 AES-128 算法 ECB 模式仿真结果

6.4 本章小结

本章首先以 DES 算法以及 Rijndael 算法两种典型算法为例，讨论了分组密码算法在 RCPA 上的具体映射过程。而后研究和分析了多种常见的分组密码算法在可重构密码处理模型 RCPA 上的映射性能，并对其映射性能进行了分析与评估并进行了横向比较。设计实现了一款基于 RCPA 模型的验证原型，该原型已在 FPGA 上成功进行了模拟验证并完成了在 0.18μm CMOS 工艺标准单元库下逻辑综合以及布局布线工作。

实验结果表明，在 RCPA 验证原型上执行的分组密码算法都可达到较高的性能。对于大多数分组密码算法，其密码处理性能与通用高性能微处理器处理性能相比提高了 10~20 倍；与其它一些专用可重构密码处理结构处理性能相比提高了 1.1~5.1 倍。结果说明本文研究的 RCPA 模型既能保证分组密码算法应用的灵活性，又能够达到较高的性能。

第七章 总结与展望

7.1 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的模型、可重构密码处理元素的设计方法、密码算法在模型上的映射以及验证原型实现。力求使该可重构密码处理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] 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单元的构成。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。

[2] 根据所分析的分组密码处理结构的特点，结合可重构处理结构的设计方法提出了一种可重构密码处理结构模型 RCPA。该模型的设计特点是粗粒度、混合型互连网络、类 VLIW/EPIC 计算模型、动态与静态配置相结合的可重构处理结构。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单元。通过对 RCPA 特征参数取值定义，对模型进行性能分析。

[3] 对可重构密码处理 RCPA 模型中的可重构密码处理元素 RCE 进行了研究和设计。RCE 设计包括了移位单元、有限域乘法、模加/减、模乘、基本逻辑运算、S 盒替代和比特置换等密码处理元素。这些基本运算元素可根据 RCE 配置指令进行重构，灵活完成不同算法所需的运算功能。

[4] 各可重构密码处理元素 RCE 在 FPGA 上进行了综合及得到了正确的仿真验证。此外还进行了 ASIC 移植工作，采用了 Synopsys 公司的 Design Compiler 工具在 0.18μm CMOS 工艺标准单元库下进行了逻辑综合以及对其综合结果进行了性能分析和比较。

[5] 研究和分析了多种常见的分组密码算法在可重构密码处理模型 RCPA 上的映射，对其映射性能进行了分析与评估并进行了横向比较。结果表明 RCPA 结构能够达到高于通用高性能微处理器和其它一些专用可重构密码处理结构的性能。

[6] 基于 RCPA 模型，采用 Verilog 硬件描述语言编码，完成了一种验证原型的设计。该原型在 FPGA 上成功进行了模拟验证并完成了在 0.18μm CMOS 工艺标准单元库下逻辑综合以及布局布线工作。

7.2 展望

由于时间和工作量等因素的限制，本文的工作还不十分完善，还有需要改进与优化的方面，还有很多理论与具体实现上的工作需要进一步去完成。未来将在以下几个方面继续

展开工作：

[1] 目前本文在基于 RCPA 架构的验证原型上已经实现了 10 余种分组密码算法，下面需要进一步对其他分组密码算法进行验证，以充分证明其设计上对分组密码算法的适应性。

[2] 本文提出的 RCPA 模型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，将其同样适用于 RSA 等公钥算法的硬件实现。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。

[3] 目前 RCPA 架构验证原型上所实现的算法都是根据专用指令集手工编制的，指令代码较为简洁。但若要使映射算法高效执行，则需要了解 RCPA 架构验证原型的体系结构。这说明了支持 RCPA 架构验证原型上层的编译器软件工具的研究和设计将是本文未来的一个课题。

[4] 对低功耗设计技术需要进行深入研究。随着通信技术的快速发展，低功耗的设计对未来保密通信处理具有十分重要的意义。为了提高性能，RCPA 将多个可重构密码处理单元集成在一起，因此对低功耗的要求就更为迫切。

参考文献

[1] 曲英杰. 可重构密码协处理器的概念及其设计原理[J]. 计算机工程与应用，2003，12：7-9.

[2] 李仁发，周祖德，陈幼平等. 可重构计算的硬件结构[J]. 计算机研究与发展，2003，40(3)：500-506.

[3] 张宇. 面向存储器级耦合的可重构硬件加速部件的研究与模拟[D]. 长沙：国防科技大学，2005：

[4] 熊华. 可重构计算部件数据耦合器的体系结构设计[D]. 杭州：浙江大学，2005：

[5] 倪晓强. 并行向量密码处理的结构设计与研究[D]. 长沙：国防科技大学，2004：

[6] Herman Schmit, David Whelihan, Andrew Tsai, Matthew Moe, Benjamin Levine, R. Reed Taylor. A Virtualized Programmable Datapath in 0.18 Micron Technology [R]. Department of Electrical and Computer Engineering Carnegie Mellon University.

[7] Seth Copen Goldstein, Herman Schmit, Mihai Budiu, Srihari Cadambi, Matt Moe, R. Reed Taylor. A Reconfigurable Architecture and Compiler[J]. COMPUTER, April 2000：2-9.

[8] Seth Copen Goldsteiny, Herman Schmit, Matthew Moe, Mihai Budiuy, Srihari Cadambi, R.Reed Taylor, Ronald Laufer. A Coprocessor for Streaming Multimedia Acceleration[A]. In: The 26th Annual International Symposium on Computer Architecture[C]. Atlanta, Georgia, May 1999.

[9] Cadambi S, Weener J, Goldstein S C. Managing Pipeline-Reconfigurable FPGAs[A]. In: Proceedings ACM/SIGDA Sixth International Symposium on Field Programmable Gate Arrays[C]. 1998.

[10] Budiu M, Goldstein S C. Fast Compilation for Pipelined Reconfigurable Fabrics FPGA[R]. 1999.

[11] M. H. Lee, H. Singh, G. Lu, N. Bagherzadeh, F. J. Kurdahi. Design and Implementation of the MorphoSys Reconfigurable Computing Processor[J]. Journal of VLSI Signal Processing Systems, 2000, 24: 164-172.

[12] Singly H., Ming-Hau Lee, Guangming Lu, etal. MorphoSys: an integrated reconfigurable system for data-parallel and computation-intensive applications[A]. In: IEEE Transactions on Computers[C]. 2000, 49(5): 465-481.

[13] 季爱明. 二维阵列型可重构计算设计空间搜索方法研究[D]. 杭州：浙江大学，2006：

[14] Chris Fisher, Kevin Rennie, Guanbin Xing, Stefan G. Berg, Kevin Bolding, John Naegle, Daniel Parshall, Dmitriy Portnov, Adnan Sulejmanpasic, Carl Ebeling. An Emulator for Exploring RaPiD Configurable Computing Architectures[R]. Department of Computer Science and Engineering University of Washington.

[15] Shawn A. Phillips . Automating Layout of Reconfigurable Subsystems for Systems-on-a-Chip[D]. Washington: University of Washington, 2004:

[16] Carl Ebeling, Darren C. Cronquist, and Paul Franklin. RAPID-Reconfigurable Pipelined Datapath[A]. In: The 6th International Workshop on Field Programmable Logic and Applications[C]. 1996: 126-135.

[17] Darren C. Cronquist, Paul Franklin, Chris Fisher and Carl Ebeling. Architecture Design of Reconfigurable Pipelined Datapaths[R]. Department of Computer Science and Engineering

University of Washington.

[18] Carl Ebeling, Darren C. Cronquist, Paul Franklin and Chris Fisher. RAPID-A Configurable Computing Architecture for Compute-Intensive Application[R]. Department of Computer Science and Engineering University of Washington， November, 1996.

[19] Carl Ebeling. The General Rapid Architecture Description[R]. Department of Computer Science and Engineering University of Washington.

[20] Chris Fisher, Kevin Rennie, Guanbin Xing, Stefan G. Berg, Kevin Bolding, John Naegle, Daniel Parshall, Dmitriy Portnov, Adnan Sulejmanpasic, Carl Ebeling. An Emulator for Exploring RaPiD Configurable Computing Architectures[R]. Department of Computer Science and Engineering University of Washington.

[21] Adam J. Elbirt. Reconfigurable Computing For Symmetric-Key Algorithms[D]. Massachusetts: Electrical and Computer Engineering Department University of Massachusetts Lowell, 2002:

[22] AJ Elbirt. Instruction-Level Distributed Processing for Sysmmetric-Key Cryptography[A]. In: Proceedings of the Seventeenth International Parallel and Distributed Processing Symposium-IPDPS[C]. April 2003.

[23] 李宏亮等. 粗粒度可重构计算体系结构综述[J]. 高性能计算技术，2006，6(3)：1-5.

[24] 白永强. 面向多媒体的粗粒度可重构处理单元的结构研究[D]. 西安：西北工业大学，2006：

[25] T.Miyamori and K.Olukotun. REMARC: Recongurable Multimedia Array Coprocessor[A]. In: IEICE Trans, Information System[C]. 1999：389-397.

[26] Charles Henri-Gros，Alan Keefer，Ankur Singla. Xtensa+，A Crypto Processor for Embedded Applications[EB].：http://suif.standard.edu/cs343/projects/AES1/paper.pdf

[27] 曲英杰. 可重组密码逻辑的设计原理[D]. 北京：北京科技大学，2002：

[28] 姜晶菲. 可重构密码处理结构的研究与设计[D]. 长沙：国防科学技术大学，2004：

[29] 冯登国，裴定一. 密码学导引[M]. 北京：科学出版社，2001：107.

[30] (美)施奈尔著, 吴世忠等译. 应用密码学：协议、算法与C源程序[M]. 北京：机械工业出版社, 2001：189.

[31] 赖溪松，韩亮等. 计算机密码学及其应用[M]. 北京：国防工业出版社，2001：159.

[32] A. Menezes，P. van Oorschot，S. Vanstone. Handbook of Applied Cryptograph[M]. CRC Press, 1996：231.

[33] 杨博涵. 针对多媒体图像处理的可重构处理元设计[D]. 西安：西北工业大学，2005：

[34] T. Miyamori, K. Olukotun. A Quantitative Analysis of Reconfigurable Coprocessors for Multimedia Applications[A]. In: Proc. of IEEE Sym. on FCCM98[C]. 1998: 2-11.

[35] R.D. Hudson, D.I. Lehn, P.M. Athanas. A Run-Time Reconfigurable Engine for Image Interpolation[A]. In: IEEE Symp. on FPGAs for Custom Computing Machines[C]. Napa Valley, 1998: 88-95.

[36] Michael Bedford Taylor . The Raw Processor Specification[EB] . : ftp://ftp.cag.csail.mit.edu/pub/raw/documents/RawSpec99.pdf.

[37] D.Chen and J.Rabaey . Reconfigurable Multi-Processor IC for Rapid Prototyping of Algorithmic-Specific High-Speed Data Paths[J]. IEEE Journal of Solid-State Circuits, 1992, 27(12):

54-61.

[38] 尹永生. 可重构多流水计算系统研究[D]. 合肥：合肥工业大学，2006：

[39] Zhijie Jerry Shi. Bit Permutation Instructiong: Architecture, Implementation, and Cryptographic Properties[D]. Princeton：Princeton University, 2004:

[40] Jiling Zhong. Upper Bound Analysis and Routing in Optical Benes Networks[D]. Georgia: Georgia State University, 2005:

[41] 张文婧，吕述望. 一种适用于分组密码算法芯片的 IP 核设计研究[J]. 计算机工程与应用，2002，22：60-62.

[42] Han Yongfei, Leong Peng-chor. Fast Algorithms for Elliptic Curve Cryptisystems over Binary Finite Field[A]. In: Advances In Cryptology- ASLACRYPT99[C]. 1999:76-85.

[43] Markus Hutter, Johann Grobschadl, Guy-Armand Kamendje . A Versatile and Scalable Digit-Serial/Parallel Multiplier Architecture for Finite Fields GF(2^m) [A]. In: Proceedings of the 4th International Conference on Information Technology: Coding and Computing[C]. 2003: 692–700.

[44] Thorsten R.Staake. Design of a Reconfigurable Hardware Architecture for Encryption Algorithms[D]. Darmstadt: Darmstadt University of Technology, 2003:

[45] 王新刚等. 一种并行乘法器的设计与实现[J]. 计算机应用研究，2004，7：135-137.

[46] Jan M. Rabaey, Anantha Chandrakasan, Borivoje Nikolic. Digital Integrated Circuits: A Design Perspective[M]. Pearson Prentice Hall, 2003.

[47] Yutai Ma. A Simplified Architecture for Modulo (2n+1) Multiplication[J]. IEEE TRANSACTIONS ON COMPUTERS, 1998, 47 (3).

[48] 韦宝典等. 求 S 盒布尔函数表达式的一种新算法[J]. 通信学报，2003，24(11)：106-111.

[49] 苏海亮. 对称算法空间的可重组逻辑研究与 SOC 设计[D]. 贵州：贵州大学，2006：

[50] 曲英杰等. 可编程 S 盒及反馈移位寄存器的设计方法[J]. 计算机工程与应用，2001，11：48-51.

[51] A.Murat Fiskiran, Ruby B.Lee. Fast Parallel Table Lookups to Accelerate Symmetric-key Cryptography[A]. In: The Proceedings of the International Conference on Information Technology Coding and Computing[C]. Las Vegas, Nevada, USA, 2005: 4-6.

[52] T.H.Cormen, C.E, Leiserson, and R.L Rivest. Introduction to Algorithms[M]. Cambridge, MIT Press, 1994.

[53] Altera Corporation. StratixII Device Handbook[EB]. http://www. altera.com, 2005.

[54] Crypto++ 5.1 Benchmarks[EB]. http://www.eskimo.com/ ~weidai/benchmarks.html.

[55] Cylink Corporation. SAFER+ Cylink Corporation's Submission for the Advanced Encryption Standard[R]. Ventura, CA: Standard First Advanced Encryption Standard Candidate Conference, August 1998:

[56] Chae Hoon Lim. Specification and Analysis of CRYPTON Version 1.0[R]. Cryptography&Network Security Center, Future Systems, Inc., 1999:

[57] 向楠，戴紫彬，徐劲松. 一种基于 BENES 网络的可重构比特置换系统设计[J]. 计算机工程，2007，24：(待发表).

[58] J.-L. Beuchat . Modular Multiplication for FPGA Implementation of the IDEA Block Cipher[R]. Laboratoire de l'Informatique du Parall´elisme, Ecole Normale Sup´erieure de Lyon, 46

All´ee d'Italie, 69364 Lyon Cedex 07, Sept. 2002 :

[59] 刘景伟．分组密码中关键问题的研究[D]．西安：西安电子科技大学硕士论文，2004：

[60] 周旋．分组密码的设计与分析[D]．长沙：国防科学技术大学，2003：

作者简历　攻读硕士学位期间完成的主要工作

一、 个人简历

杨晓辉，男，1978 年 8 月出生，安徽蚌埠人。

2000 年 7 月毕业于解放军信息工程学院，无线电技术专业；

2004 年 9 月入解放军信息工程大学电子技术学院攻读硕士研究生，计算机应用技术专业；

二、攻读硕士学位期间发表的学术论文

[1] Design of General Multiplication in GF(2^8) and FPGA Implementation. 2006 1th International Symposium On Pervasive Computing and Applications Proceedings. 2006.8 IEEE Catalog Number:06EX1359 pp. 503-507，第一作者 (ISTP 检索)

[2] The Research And Design Of Reconfigurable Cryptographic Chip Based On Block Cipher. ICSICT-2006 8th International Conference on Solid-State and Integrated-Circuit Technology. 2006.10 IEEE Catalog Number:06EX1294 pp. 1972-1975，第一作者

[3] The Researching and Implementation of Reconfigurable Hash Chip Based On FPGA. Journal of System Engineering and Electronics. Volume 18, Number 1, 2007. pp. 183-187，第一作者

[4] 一种基于 FPGA 的可重构密码芯片的设计与实现. 电子技术应用. 2006 年第 8 期，第一作者

[5] 基于 FPGA 的 SHA-256 算法实现. 微计算机信息. 2006 年第 4 期中期，第一作者

[6] A Design and Implementation of Reconfigurable Shift Unit using FPGAs. 2006 1th International Symposium On Pervasive Computing and Applications Proceedings. 2006.8 IEEE Catalog Number:06EX1359 pp. 543-545，第四作者 (ISTP 检索)

[7] 基于 EPIC 技术的密码处理器体系结构研究与设计. 电子技术应用. 2007 年第 2 期，第三作者

三、攻读硕士学位期间的科研情况

[1] 公安部项目《公网(GPRS/CDMA)移动数据接入系统》，本人的主要工作是参与 64 位高速数据加密卡中部分加密算法的设计与 FPGA 实现。

[2] 武器装备预研项目《密码专用微处理器的研究》，本人的主要工作是参与处理器中的专用密码处理单元的研究与设计。

[3] 解放军信息工程大学电子技术学院自研项目《高速数据加密卡设计》，本人主要参与研究密码重构技术以及分组密码关键参数重构技术。

[4] 军队"2110 工程"重点建设项目安全专用芯片实验室立项的自研项目《可重构分组密码协处理器》，本人主要参与分组密码协处理器整体架构、可重构运算单元、协处理器接口的研究与设计。

致　谢

时间飞逝，转眼三年的硕士学习生活即将结束，回首这几年的求学经历，我有幸得到许多人的关心和帮助，我科研成果的取得与他们的付出息息相关，无论现在还是将来，我都将深深地记住他们，真诚地感谢他们。

首先感谢我的导师戴紫彬教授！三年来戴紫彬教授在学业上给与了我悉心的指导和全力的支持，不断督促我刻苦钻研，勤奋思考。他针对我学业的每个阶段提出新的要求，引导我克服各种困难，在学习工作中迈向一个又一个新的台阶。是他严谨求实的作风影响着我，给我的学业乃至人生莫大的帮助。戴老师在学术上独到的见解和丰富的实践经验令人钦佩，也给我的课题很大的启示。他对事业执着追求的精神和为之所做出的奉献牺牲更成为鞭策我前进的动力和榜样。恩师的教导不仅是我学业的动力，也是我人生中的一份永远的温暖和感动！硕士学业即将完成之际，谨向导师致以最衷心的谢意！

衷心感谢信息安全系统工程教研室苏锦海主任、刘建国副教授、孙万忠教员和教研室其他老师。他们在我进入课题后，给予我很多具体的指导和帮助，在他们的帮助下我顺利完成课题实践和硕士论文。

感谢课题组陈韬教员、于学荣、孟涛、仲洗海、李伟等同学对本课题的大力支持。没有他们的帮助，最后试验难以完成。

感谢学院、系、队的各级领导对我的关怀，他们为我在学习、生活等方面提供了许多便利条件。感谢九队的全体同学，感谢易青松、刘晶晶、孟强、迟鹏、王峰等同学对我学习和生活上的帮助。

最深的谢意给我远方的父母和弟弟，他们不远千里的时时关怀给了我莫大的安慰，给我最温暖的心灵港湾，愿亲爱的父母健康如意，愿亲爱的弟弟事业有成。

感谢三年来所有帮助过我的人！

衷心地谢谢你们！

面向分组密码处理的可重构设计技术研究

作者： 杨晓辉
学位授予单位： 中国人民解放军信息工程大学
被引用次数： 4次

本文读者也读过(10条)

1. 郑东 可重构分组密码处理系统实现研究[学位论文]2009
2. 周信坚 基于可重构技术的密码分析系统模型研究[学位论文]2008
3. 孟涛.戴紫彬.MENG Tao.DAI Zi Bin 可重构S盒运算单元的设计与实现[期刊论文]-电子技术应用2007,33(5)
4. 曲英杰 可重构密码处理器设计思想探讨[期刊论文]-北京电子科技学院学报2003,11(1)
5. 赵成 基于FPGA的动态可重构系统实现密码算法的研究[学位论文]2009
6. 李冬华 领域内可重构功能单元自动生成技术研究[学位论文]2006
7. 尹勇生 可重构多流水计算系统研究[学位论文]2006
8. 王莉 密码算法的可重构系统实现研究[学位论文]2007
9. 向纯洁 控制网络节点可重构通信协议栈的建模和评价[学位论文]2007
10. 郑东.王友仁.张砦.ZHENG Dong.WANG You-ren.ZHANG Zhai 基于遗传算法的快速可重构S盒硬件设计[期刊论文]-信息与控制2009,38(3)

引证文献(3条)

1. 褚有睿.欧阳旦.王志远 一种改进的分组密码可重构处理结构设计[期刊论文]-计算机系统应用 2010(8)
2. 褚有睿.王志远.欧阳旦 基于可重构密码模块的VPN安全网关[期刊论文]-计算机工程 2011(5)
3. 孟涛.戴紫彬 分组密码处理器的可重构分簇式架构[期刊论文]-电子与信息学报 2009(2)