分类号： TN492　　学校代码：10363
密　级： 公开　　学　　号：2210320111

安徽工程大学
Anhui Polytechnic University

硕士学位论文

题目 星载高速 AES 密码电路设计

论 文 作 者　徐明宇
指 导 教 师　张肖强
学科（专业）　控制科学与工程（0811）
研 究 方 向　安全芯片的优化设计

论文提交日期：2024 年 05 月 31 日

分类号：TN492　　单位代码：10363
密　级：公开　　学　　号：2210320111

题　　目 星载高速 AES 密码电路设计

英文并列

题　　目 DESIGN OF SATELLITE-BASED HIGH SPEED AES CIPHER CIRCUIT

学生姓名：徐明宇

指导教师：张肖强（副教授）

专　　业：控制科学与工程（0811）

研究方向：安全芯片的优化设计

论文答辩日期：2024 年 05 月 31 日

二、安徽工程大学学位论文原创性声明

本人郑重声明：我恪守学术道德，崇尚严谨学风。所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已明确注明和引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品及成果的内容。论文为本人亲自撰写，我对所写的内容负责，并完全意识到本声明的法律后果由本人承担。

学位论文作者签名： 

日期：　2024 年 05 月 31 日

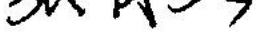
三、安徽工程大学学位论文版权使用授权书

学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅或借阅。本人授权安徽工程大学可以将本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和汇编本学位论文。

保密 □，在 ____ 年解密后适用本版权书。

本学位论文属于

不保密 ☑。

学位论文作者签名：  指导教师签名： 

日期： 2024 年 05 月 31 日 日期： 2024 年 05 月 31 日

星载高速 AES 密码电路设计

摘 要

在全球信息化的当下，密码技术作为信息安全的基石，为信息的安全传输和存储提供了有力保障。作为密码技术首选的加密标准，高级加密标准（Adavanced Encryption Standard, AES）广泛应用于各个领域。本文针对卫星加密通信领域，研究高速 AES 密码电路优化设计方法，重点解决了高速 AES 架构设计中密钥扩展单元中关键路径长度较长的关键问题，并在此基础上利用能量分析原理搭建硬件检测平台对其进行安全性分析。论文以中国航天科工集团 xxx 研究所侦察卫星加密通信领域中星载高速 AES 密码电路为研究对象，主要工作如下：

（1）以 AES-128 加密算法对电子侦察载荷 LVDS 接口数传数据进行加密，并按 848 字节为一帧实现侦察载荷 FPGA 加密通信。分模块分单元对加密电路中数据接口模块和加密运算单元进行仿真验证，并在 XC7VX690T 系列 FPGA 平台上实现。在时钟约束为 100MHz 的条件下，电路运行的最高频率为 221.73MHz，电路功耗为 0.4W，电路共用 447 个 Slice，只占 Slice 总数的 0.41%。

（2）为保证已加密的数据信息可以顺利解密，本文利用地检设备 FPGA 设计实现硬件解密模块。主要由密钥扩展和解密运算两个单元构成，在使用 Vivado 开发工具分模块分单元对电路进行波形仿真后，在 100MHz 的时钟约束下，得到电路运行的最高频率为 165.21MHz，电路功耗为 0.39W，电路共用 845 个 Slice，只占 Slice 总数的 1.11%，并在 XC7VX690T 系列 FPGA 平台上实现。

（3）为降低 AES 密码电路延迟，本文对 AES 密钥扩展单元中关键路径长度较长的问题，提出一种构造低延迟密钥扩展单元的方法。首先推导 AES 密钥扩展运算的表达式，然后利用延迟感知公共项消除算法（Delay-Aware Common Subexpression Elimination, DACSE）来提取逻辑表达式中的公共项（Common Subexpression, CS），最后基于延迟驱动二叉树结构（Delayed-Driven Binary Tree, DDBT）构建低延迟密钥扩展单元。将该方法应用于 AES 加密密钥扩展单元，并对其采用 SMIC 0.18μm 技术的 Synopsys™ 集成电路 IC 综合工具 DC 进一步验证。结果表明在 AES 加密密钥扩展单元中，以 DDBT 结构的构建的延迟电路在面积增加 10.08%的情况下电路的延迟降低 15.23%，可见优化后的设计以较少的面积增加为代价较大降低了电路延迟。

（4）基于能量分析的原理，从攻击者的角度搭建安全性检测平台，以 DPA 中均值差法为安全性分析的手段，以首轮 S 盒作为检测位置，对上述星载高速 AES 加密密码电路进行能量数据的采集分析。实验结果表明只需在存储深度为 1Kpts 的条件下，采集 10000 余条能量迹即可确定实际密钥。此外，在与 32 位总线结构的 AES 密码算法所需的检测条件比较后可得出，只需要将加密数据的输入位宽降低，就可以增加检测分析的繁杂程度，提高密码算法硬件安全性。

关键词：AES，FPGA，密钥扩展单元，关键路径，安全性检测平台

DESIGN OF SATELLITE-BASED HIGH SPEED AES CIPHER CIRCUIT

ABSTRACT

In the current era of global informatization, cryptography technology, as the cornerstone of information security, provides a strong guarantee for the safe transmission and storage of information. As the preferred encryption standard for cryptography technology, Advanced Encryption Standard (AES) is widely used in various fields. Focusing on the field of satellite encrypted communications, this paper studies the optimization design method of high-speed AES cryptographic circuits, focusing on solving the key problem of long critical path length in the key expansion unit in the design of high-speed AES architecture. On this basis, it uses energy analysis principles to build hardware detection The platform conducts security analysis on it. The paper takes the spaceborne high-speed AES cryptographic circuit in the field of reconnaissance satellite encrypted communication of the China Aerospace Science and Industry Corporation xxx Research Institute as the research object. The main work is as follows:

(1) Use the AES-128 encryption algorithm to encrypt the digital transmission data of the electronic reconnaissance payload LVDS interface, and implement the reconnaissance payload FPGA encrypted communication according to 848 bytes as one frame. The data interface module and encryption operation unit in the encryption circuit are simulated and verified by modules and units, and implemented on the XC7VX690T series FPGA platform. Under the condition that the clock constraint is 100MHz, the maximum frequency of the circuit operation is

221.73MHz, the circuit power consumption is 0.4W, and the circuit shares 447 slices, accounting for only 0.41% of the total number of slices.

(2) In order to ensure that the encrypted data information can be decrypted smoothly, this article uses the FPGA of the ground inspection equipment to design and implement the hardware decryption module. It is mainly composed of two units: key expansion and decryption operation. After using the Vivado development tool to conduct waveform simulation of the circuit by module and unit, under the clock constraint of 100MHz, the maximum frequency of the circuit operation is 165.21MHz, and the circuit power consumption It is 0.39W. The circuit shares 845 Slices, accounting for only 1.11% of the total number of Slices, and is implemented on the XC7VX690T series FPGA platform.

(3) In order to reduce the delay of the AES cryptographic circuit, this paper proposes a method to construct a low-latency key expansion unit to solve the problem of long critical path length in the AES key expansion unit. First, the expression of the AES key expansion operation is derived, then the delay-aware common subexpression elimination algorithm (DACSE) is used to extract the common subexpression (CS) in the logical expression, and finally based on the delay-driven binary tree Structure (Delayed-Driven Binary Tree, DDBT) to build a low-latency key expansion unit. The method was applied to the AES encryption key expansion unit and further verified using Synopsys™ integrated circuit IC synthesis tool DC using SMIC 0.18μm technology. The results show that in the AES encryption key extension unit, the delay circuit constructed with DDBT structure reduces the circuit delay by 15.23% when the area increases by 10.08%. It can be seen that the optimized design significantly reduces the circuit delay at the cost of less area increase.

(4) Based on the principle of energy analysis, build a security detection platform from the perspective of the attacker, use the mean

difference method in DPA as the security analysis method, use the first round S box as the detection position, and perform the above-mentioned spaceborne high-speed AES encryption password The circuit collects and analyzes energy data. Experimental results show that the actual key can be determined by collecting more than 10,000 energy traces under the condition of a storage depth of 1Kpts. In addition, after comparing with the detection conditions required by the AES cryptographic algorithm with a 32-bit bus structure, it can be concluded that only reducing the input bit width of the encrypted data can increase the complexity of the detection analysis and improve the hardware security of the cryptographic algorithm.

KEY WORDS： AES, FPGA, key expansion unit, critical path, security detection platform

目录

第 1 章 绪论

1.1 研究背景及意义

随着云计算、大数据、人工智能等新技术的发展，数据安全和隐私保护的需求日益增长。“云、大、物、移、智、链”等新技术的发展伴生了诸多信息安全问题，密码学为这些问题的解决提供了独特视角和可行路线，同样这些新技术也促进了密码学研究的深入发展和广泛应用[1]。

从监管角度来看，国内外对信息安全的重视和保护日益严格。2018 年 5 月生效的 GDPR 被称为欧盟“史上最严”条例[2]。Google 和 Facebook 在 GDPR 生效日分别收到了欧盟 39 亿欧元、37 亿欧元罚款诉讼，产生巨大影响。而我国自 2020 年实施的《中华人民共和国密码法》从国家法律层面指出了密码技术的重要性，明确了密码是国之重器，是国家重要的战略资源，直接关系国家政治安全、经济安全、国防安全和信息安全[3]。

现今实现安全功能的密码算法可分为 3 类[4]：摘要密码算法、非对称密码算法和对称密码算法，现代密码算法分类和应用范围如图 1-1 所示。

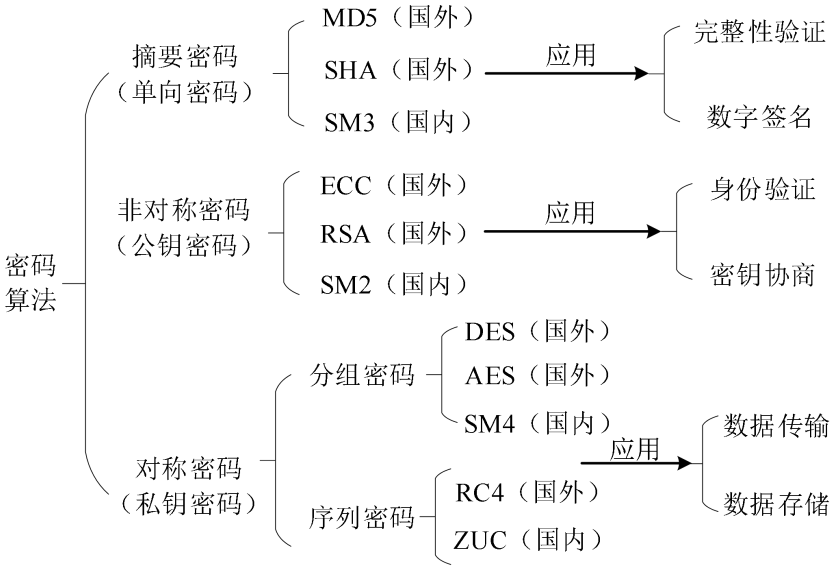


图 1-1　现代密码算法分类和应用范围

摘要密码算法又称为单向加密算法或哈希函数。该算法可以将任意长度的输入数据转化成固定长度的输出数据，且整个变换过程没有密钥。主要用于数据完整性验证、消息认证、数字签名等领域。常见的摘要密码算法有信息-摘要算法 5（Message-Digest Algorithm 5, MD5）、SM3 密码杂凑算法[5][6]、安全散列算法

（Secure Hash Algorithm, SHA）等密码算法[7]。

非对称密码算法因加密过程与解密过程中所使用的密钥不同，且其中公开公钥推不出另一个保密私钥，所以这种算法也被称为公钥密码算法。它解决了密钥管理和分发的问题，提供了更高的安全性，多应用于密码协商、数字签名以及身份验证等其安全性要求较高的领域[8]。但非对称加密算法由于计算复杂，加密和解密速度低，因此在实际应用中常常与对称密码算法结合使用，即在数据传输和存储时采用对称密码算法，而在密钥管理和分发时采用非对称密钥算法[9]。例如，SSL/TLS（Secure Sockets Layer / Transport Layer Security）协议结合了非对称密码算法和对称密码算法来确保通信安全；比特币等加密货币使用非对称加密来确保交易的安全性。常见的非对称密码算法有椭圆曲线密码算法（Elliptic Curve Cryptography, ECC）[10]、SM2 公钥密码算法[11]和 RSA 算法（Rivest-Shamir-Adleman, RSA）[12]。

对称密码算法在加密和解密的过程使用相同密钥，即加密密钥也可以用作解密密钥。这类算法运算速度快，适合处理大量数据，因此广泛应用于文件加密、网络信息传输加密、以及移动设备加密等场景。对称密码算法一般可分为分组密码算法和序列密码算法两类。其中，分组密码算法在加密过程中将输入明文划分为固定长度的字块，在密钥的控制下加密完成后，获得相同长度的输出密文字段，而解密过程则是在相同密钥的控制下进行加密算法的逆运算[13]。典型的分组密码算法有数据加密标准（Data Encryption Standard, DES）[14]、高级加密标准（Advanced Encryption Standard, AES）[15]、SM4 分组密码算法[16]等。另外，序列密码算法是对输入进行逐位加密，其安全性取决于密钥序列的伪随机性，常用的序列密码算法有 RC4 算法[17]和 ZUC 序列密码算法[18]。

高级加密标准 AES 是美国国家标准与技术研究院（National Institute of Standards and Technology, NIST）于 2001 年发布了新一代分组密码算法标准[19]，以其密钥建立时间短、安全性高、计算复杂低等特点，成为国际上使用最广泛的分组密码算法[20]。AES 算法实现主要有软件和硬件两种实现方式，其中软件实现常采用 C 语言或 Matlab 等软件编程实现，具有灵活性高、可扩展性强和可移植性好等优点，但是加密速度较低，并且由于软件运行环境的开放性，密码算法和加密信息容易被篡改和窃取[9]，而硬件实现常采用 Verilog 描述语言传输到硬件平台上实现，安全性和运行速度得到很大的提高[21]。AES 密码算法在硬件平

台上运行往往受到成本、功耗、硬件资源等限制，这给电路设计带来新的挑战，因此关于高性能 AES 密码电路硬件实现的研究越来越多。

虽然 AES 密码算法在设计之初就已经具备了较高的数学安全性，使攻击者想从数学角度来破解密码算法是十分困难的，但是当密码算法运行在具体的芯片硬件设备时，往往会泄露一些关联信息，从而帮助攻击者破解密码算法[22]。自 1996 年由 Kocher 等人提出的旁路攻击（Side Channel Attck, SCA）[23]以来，攻击者只需要窃取旁路信息就可以降低攻击难度，更加容易的破解密码算法。如图 1-2 所示的旁路攻击原理。

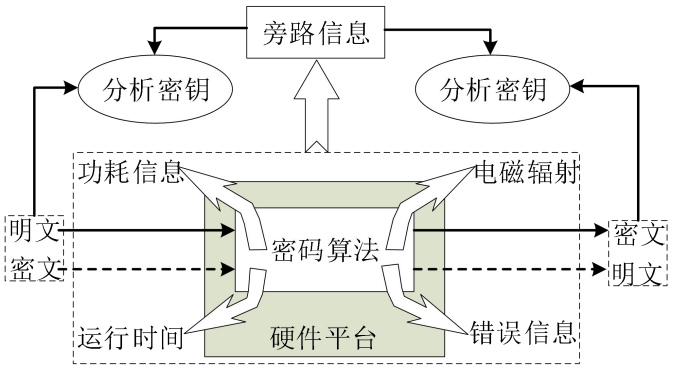


图 1-2　旁路攻击原理示意图

由旁路信息的选择可将分析方式划分为四类：错误攻击技术（Fault Attack, FA）[29-31]、时间分析（Timing Analysis, TA）[32-34]、电磁辐射分析（Electromagnetic Analysis, EMA）[35-38]与功耗攻击（Power Attack, PA）[39-40]。由于功耗攻击实现简单，密钥搜索空间较小，因此是旁路攻击中最重要、最常用的攻击手段[13]。功耗攻击又称为能量分析（Power Analysis），通过将密码芯片所消耗的能量与算法中间值和密钥之间的相关性建立联系，利用数学统计的方法缩小密钥检测范围，从而分析出实际密钥[24]。目前常用的功耗攻击可以分为简单功耗攻击（Simple Power Attack, SPA）[26][28]、差分功耗攻击（Differential Power Attack, DPA）[27]和高阶差分功耗攻击（High-Order Differential Power Attack, HO-DPA）三种[25]。

为了确保密码电路的安全性，在设计之初需要考虑到潜在的安全性问题，以攻击者的角度解析能量分析的原理，对所设计的密码电路进行安全性评估，这就使得对能量分析的研究具有重要意义。

因此，本文针对高性能 AES 硬件实现有着重要的应用前景，同样密码芯片的安全性分析也必不可少。

1.2 高速 AES 硬件实现的国内外研究现状

密码电路的硬件实现效率主要由面积、速度和功耗这三个参数判定，但在电路实现的过程中这三个参数往往是相互制约的[41]。本文以速度为目标优化参数，分别从总体架构、密钥加模块和安全性验证平台对高速 AES 硬件实现的国内外研究现状进行分析。

AES 电路的总体架构优化一般是对轮变换单元和密钥扩展单元这两部分进行的，但 AES 算法的分组数据长度为 128 位，且内部子模块的数据处理位宽并不相同，所以可基于实际情况设计不同总线结构 AES 硬件电路。文献[42]采用 32 位全展开流水线结构，实现了高速 AES 加密。文献[43]研究了 32 位数据架构的 AES，相较于 8bit 数据路径，其资源消耗略大但数据处理速度更高。文献[44]提出了一种基于流水线化 Karatsuba-Ofman 算法（KOA）的高效 GHASH 架构，以块密码模式获得更高的运行速度。文献[45]基于 VLSI 体系利用子流水线和复合域算术提出一种高速硬件架构。其中对轮变换单元的优化，大多采用分模块的方式迭代实现。文献[46]采用在轮内子模块中插入流水线的方式实现，而文献[47]则对轮变换中各组合逻辑间插入流水线实现文。除此之外，T 盒实现方式也可以使其达到更高的处理速度，如文献[48]提出一种采用 T 盒方式实现轮变换的 128 位循环结构 AES，在有效利用面积情况下，速度可以达到 1.454Gbps。因此本文选择循环结构作为 AES 加解密运算电路的总体架构，可以在电路处理速度不变的情况下大大减少电路面积。

AES 密钥加模块一直以来都是高速 AES 实现的瓶颈[49]，这是因为密钥扩展单元一般采用链式结构来实现，但链式结构会导致的密码电路的关键路径增长、延迟增大。其中文献[50]已经证明延迟驱动二叉树结构（Delayed-Driven Binary Tree, DDBT）对于密钥加模块中线性运算具有最短的关键路径，但直接用线性运算表达式构建电路会导致面积成本增大。文献[51]通过在构造前从逻辑表达式中提取公共项（Common Subexpression, CS）来减少电路面积，但是提取 CS 又会增加关键路径。由于延迟感知公共项消除算法（Delay-Aware Common Subexpression Elimination, DACSE）可以在延迟约束下提取 CS[52]，所以可以在保持 DDBT 结构中最短关键路径不变的情况下，本文采用 DACSE 算法从密钥扩展表达式中提取 CS，再以 DDBT 结构构建低延迟密钥扩展单元并将其应用于 AES 加密密钥扩展单元。

自1996年由Kocher等人利用密码芯片的电能量来获取加密设备的机密信息以来，攻击者只需要以很小的代价降低攻击难度破解密码算法，因此密码芯片在流片之前，需要对其进行安全性评估。本文以功耗信息为采集对象，以能量分析的方式对星载高速AES密码电路进行安全评估。由于所针对的密码算法和执行操作不同，因此通过分析功耗推断密钥信息的方式也将不同。文献[75]中Messserges等人利用SPA对AES密钥进行破解。文献[76]中Mangard对AES的密钥生成采用了SPA。文献[77]中Ors等人利用DPA成功破解了专用的AES ASIC（Application Specifie Integrated Circuit）实现。文献[78]中Mangard等人针对AES的硬件实现采用DPA方式有效地得到了密钥并做出了详细的说明。此后，A.Moradi等人在2016年将SCA攻击扩展到Xilinx Virtex-5、Spartan-6、Kintex-7和Artix-7 FPGA中，针对AES-256加密的位流改进了攻击方法，采用电磁SCA将搜索空间由2^{32}减小到2^{8}大大提高效率[79]。可见基于硬件能量采集平台对于EDA软件仿真采集能量来说，采集效率高且更加符合实际，因此本文采用专业的SCA设计采集板SAKURA-G，以DPA方式对所设计的密码电路验证分析。

1.3 论文章节安排

本文针对卫星加密通信领域，以中国航天科工集团xxx研究所星载高速AES密码电路为研究对象，对AES密码电路的设计方法进一步优化。重点解决了AES密钥扩展单元中关键路径长度较长、电路延迟较大等的关键问题，并在此基础上利用能量分析原理搭建的硬件检测平台对其进行安全性分析。论文研究内容及结构安排如图1-3所示。

第1章为绪论。首先概述了国家对密码技术的大力支持和使用AES密码算法硬件实现的研究背景和意义；其次对AES密码电路的优化设计的意义和相应的安全性检测手段进行分析介绍，并以速度为目标优化参数分析基于高速AES硬件实现中总体架构和密钥加模块的研究现状，并总结了在该方面目前还存在的关键问题，提出一种构造低延迟密钥扩展单元的方法；最后介绍了本文的主要研究内容和组织结构。

第2章完成该研究对象中加密通信模块。首先以AES-128加密算法对电子侦察载荷LVDS接口数传数据进行加密；其次分模块分单元对加密电路中数据接口模块和加密运算单元进行仿真验证；最后将密码电路在星载FPGA上的实现。

第3章完成该研究对象中地面协同端的解密功能。首先针对地检设备的密钥

扩展单元和解密运算单元进行设计；其次使用 Vivado 开发工具对电路分模块分单元进行仿真验证并达到相应的标准；最后将解密电路在地检 FPGA 上的实现。

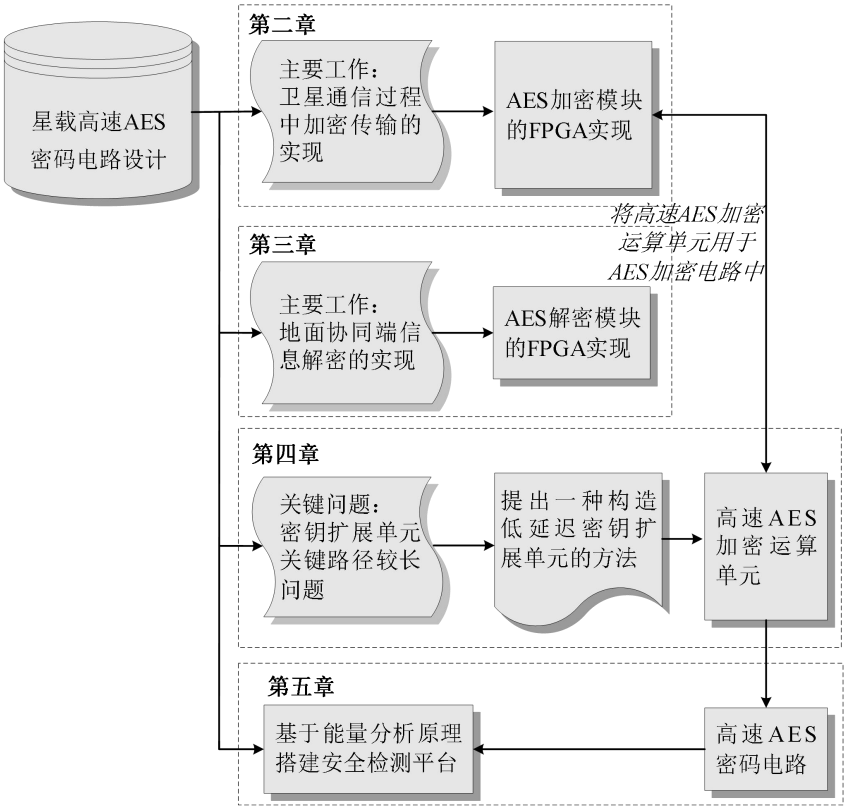


图 1-3　论文研究工作结构图

第 4 章为降低 AES 密码电路延迟，对 AES 密钥扩展单元中关键路径长度较长的问题，提出一种构造低延迟密钥扩展单元的方法。首先推导 AES 密钥扩展运算的表达式；然后利用延迟感知公共项消除算法 DACSE 来提取逻辑表达式中的公共项 CS；最后基于延迟驱动二叉树结构 DDBT 构建低延迟密钥扩展单元并将其应用于 AES 加密密钥扩展单元，采用 SMIC 0.18μm 技术的 Synopsys™ 集成电路 IC 综合工具 DC 进一步验证，得出优化后的设计以较少的面积增加为代价降低了电路延迟。

第 5 章基于能量分析的原理，从攻击者的角度搭建安全性检测平台。首先以 DPA 中均值差法为安全性分析的手段搭建安全性检测平台；然后以首轮 S 盒作为检测位置，对 32 位总线结构的 AES 密码算法进行检测分析，验证平台的准确性；最后对上述星载高速 AES 密码电路进行能量数据的采集分析，并与 32 位总线结构的 AES 密码算法所需的检测条件比较，得出只需要将加密数据的输入位宽降低，就可以增加检测分析的繁杂程度，提高密码算法硬件安全性的结论。

第 6 章对全文的总结与展望。

第 2 章 星载 AES 加密设计实现

本章针对卫星加密通信领域，以中国航天科工集团 xxx 研究所星载高速 AES 密码电路为研究对象，以 AES-128 加密算法高速 FPGA 来实现该研究对象中加密通信模块。在设计 AES 加密电路模块中，将加密模块分成“输入数据位宽转换单元”、“AES 加密运算单元”、“输出数据位宽转换单元”、“数据使能延时单元”四个部分，根据实际应用要求将数据处理与加密过程分开设计，并利用 XC7VX690T 系列 FPGA 进行仿真验证，达到实际应用要求。

2.1 高级加密标准 AES 算法

2.1.1 AES 工作模式

高级加密标准 AES 是一种基于有限域运算的分组密码标准，它的加解密运算都是在一个特定的 $GF(2^8)$域中完成。AES 的数据分组为 128 比特，同时，密钥长度可以为 128，192 或 256bit 三种。根据密钥长度不同，AES 算法分别记为 AES-128，AES-192，AES-256[9]。本文实现的目标算法为 AES-128 密码算法。通常将未加密的数据分组称为明文数据，加密之后的数据分组称为密文数据。通常将明文数据、密文数据以及 AES 算法的中间运算数据视为一个 4×4 二维字节向量，并将这个字节向量称为状态矩阵。在状态矩阵中，以列为单位进行运算，因此将状态矩阵中每一列的四个字节组成 32 比特字。同样，AES-128 密钥可以映射成 4×4 二维字节向量。AES 算法是一个迭代算法，每一个迭代可以称为轮变换，密钥长度不同，轮变换数量也不同，AES-128 的轮变换数量为 10。AES 加解密工作过程如图 2-1 所示。

在 AES-128 加密过程中，首先进行一个密钥加运算，然后是 10 轮轮变换操作，前 9 轮为普通轮变换操作，包括四个运算：S-盒运算、行移位运算、列混合运算和密钥加运算，其中 S-盒运算又称字节替换运算。最后一轮与前 9 轮稍微不同，第 10 轮轮变换没有列混合运算。在轮变换操作之前的首次密钥加运算中采用初始密钥 K_0 进行运算，在轮变换中使用的子密钥 K_i 是由初始密钥 K_0 根据密钥扩展算法产生。

AES-128 解密过程与其加密过程步骤对应。128 比特密文在解密过程中，首先进行密钥加运算，使用的密钥为末轮子密钥 K_{10}。解密过程的轮变换也是 10

轮，普通的轮变换也包括四个运算：逆行移位运算、逆 S-盒运算、密钥加运算和逆列混合运算。在解密过程中，最后一轮也与前 9 轮稍微不同，末轮轮变换没有逆列混合运算。在轮变换中使用的子密钥 K_j 可以由末轮子密钥 K_{10} 根据逆密钥扩展算法实时产生，也可以由密钥扩展算法产生后寄存到 10 轮子密钥的寄存器中，在解密运算过程中使用。

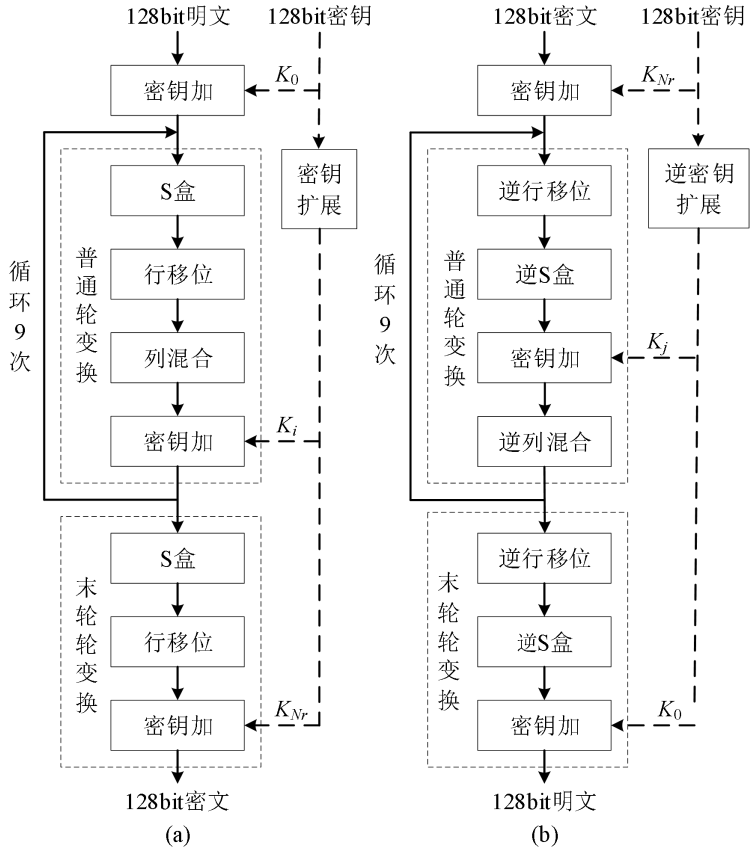
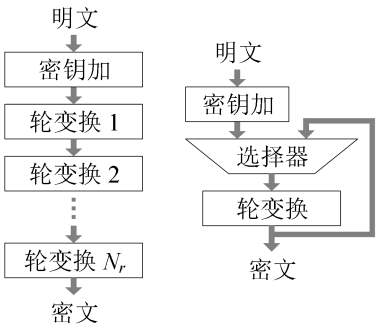


图 2-1　AES 加解密工作过程示意图

2.1.2 AES 密码电路结构

根据 AES 电路中的轮变换单元数量，AES 密码电路结构可以分成两种基本类型：全展开结构和循环结构，如图 2-2 所示。



(a) 全展开架构　　(b) 循环架构

图 2-2　AES 加解密运算单元基本实现结构图

在本文中每个时钟输入 4 比特，因此输入一个 128bit 的数据分组需要 32 个时钟。经估算，AES-128 加解密运算可以在 32 个时钟周期内完成，因此本文选择 2-2(b)所示的循环结构作为 AES 加解密运算电路的总体架构。选择循环结构可以在电路处理速度不变的情况下大大减少电路面积。

2.2 AES 加密电路模块

2.2.1 总体设计

AES 加密模块如图 2-3，是由“输入数据位宽转换单元”、“AES 加密运算单元”、“输出数据位宽转换单元”、“数据使能延时单元”四个部分组成。

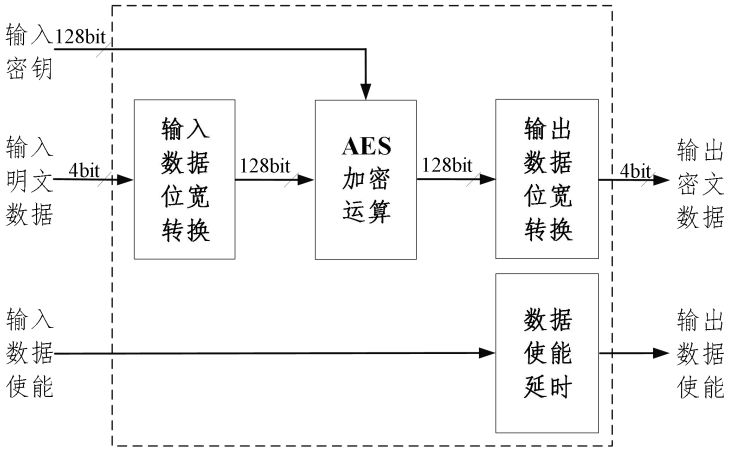


图 2-3　AES 加密电路模块总体架构图

其中，“输入数据位宽转换单元”完成输入明文数据 4 比特位宽到 128 比特位宽的转换；“AES 加密运算单元”完成 AES-128 加密运算；“输出数据位宽转换单元”完成输出密码数据 128 比特位宽到 4 比特位宽的转换；“数据使能延时单元”将输入数据使能信号延迟若干时钟后作为输出数据使能信号输出，延迟时钟个数取决于 AES 加密电路模块完成加密运算所需的总时钟个数。

2.2.2 输入数据位宽转换单元

该单元采用 32 ×4 移位寄存器来实现 4 比特位宽到 128 比特位宽的转换，其信号接口如表 2-1 所示。

在数据比特位宽的转换的过程中，需要在该单元内部添加 5 位计数器和输出使能信号产生逻辑电路，如图 2-4 所示。其中 5 位计数从 5′h00~5′h1F 循环计数，每当计数器计到 5′h1F 时，逻辑电路输出一个使能信号，从而表明 32 ×4 移位寄存器完成一组 128 比特明文数据转换，所以移位寄存器每 32 个时钟输出一组 128 比特明文数据，如图 2-5 所示。

表 2-1 “输入数据位宽转换单元”信号接口

	输入信号	数据位宽	信号含义	信号来源
1	clk	1	输入时钟	模块外部
2	DataEn	1	输入使能	模块外部
3	DataIn	4	输入明文数据	模块外部
	输出信号	数据位宽	信号含义	信号目的地
1	DataValid	1	128 比特明文数据使能	AES 加密运算单元
2	DataOut	128	128 位宽明文数据	AES 加密运算单元

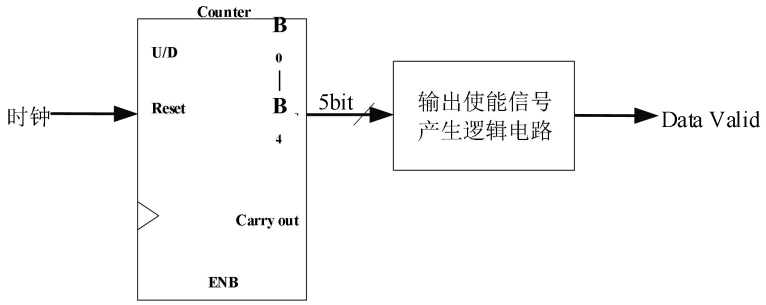


图 2-4 5 位计数器运行结构图

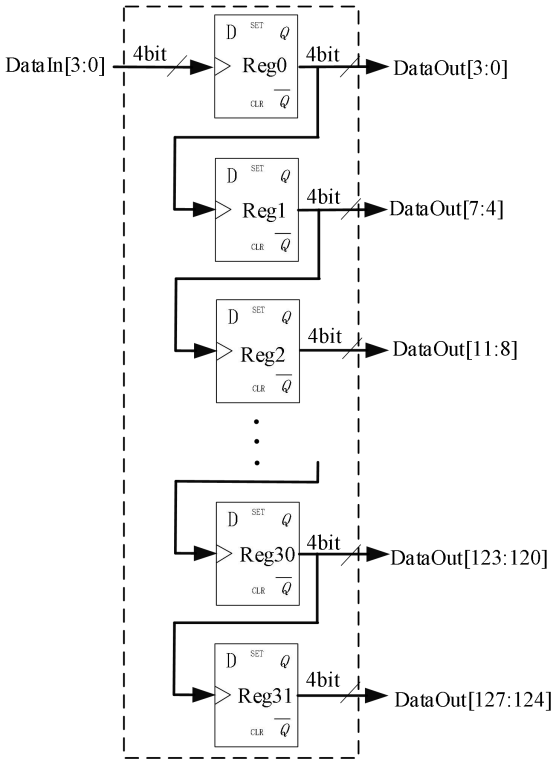


图 2-5 32×4 移位寄存器运行结构图

2.2.3 输出数据位宽转换单元

该单元采用 32 ×4 移位寄存器来实现 128 比特位宽到 4 比特位宽的转换，其信号接口如表 2-2 所示。移位寄存器每 32 个时钟输入一组 128 比特密文数据，然后依次从输出端口输出，如图 2-6 所示。

表 2-2 “输出数据位宽转换单元”信号接口

	输入信号	数据位宽	信号含义	信号来源
1	clk	1	输入时钟	模块外部
2	DataEn	1	输入使能	AES 加密运算单元
3	DataIn	128	输入 128 比特位宽密文数据	AES 加密运算单元
	输出信号	数据位宽	信号含义	信号目的地
1	DataOut	4	输出 4 比特位宽密文数据	模块外部

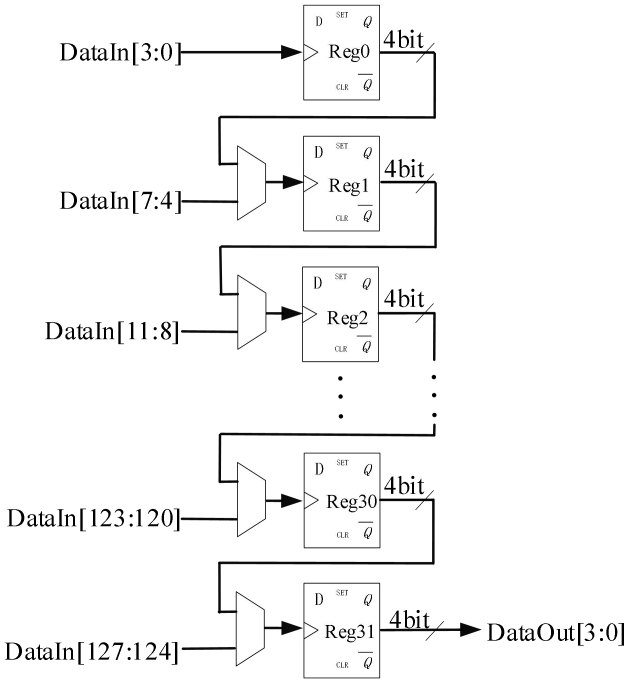


图 2-6 输出数据位宽转换单元图

2.2.4 数据使能延时单元

由于输入明文数据与输出密文数据之间的时钟个数为固定时钟个数，因此将输入数据使能信号延迟若干时钟后可以直接作为输出数据使能信号使用。输入明文数据和输出密文数据之间间隔的时钟个数可以用表 2-3 计算。

表 2-3 输入明文数据与输出密文数据之间时钟个数计算

	数据处理单元	时钟个数
1	“输入数据位宽转换单元”	32
2	“输入数据位宽转换单元”到“AES 加密运算单元”	1
3	“AES 加密运算单元”	10
4	“AES 加密运算单元”到“输出数据位宽转换单元”	1
5	“输出数据位宽转换单元”到数据输出端口	1
	总计	45

由表 2-3 可知，输入明文数据与输出密文数据之间间隔了 45 个时钟，因此将输入数据使能信号延迟 45 个时钟后作为输出数据使能信号。数据使能延时单元信号接口如表 2-4 所示。

表 2-4　数据使能延时单元信号接口

	输入信号	数据位宽	信号含义	信号来源
1	clk	1	输入时钟	模块外部
2	DataEn	1	输入数据使能	模块外部
	输出信号	数据位宽	信号含义	信号目的地
1	DataValid	1	输出数据使能	模块外部

2.3 AES 加密运算单元

2.3.1 总体架构

AES 加密运算单元主要分成三部分：密钥扩展部分、轮变换部分和使能信号产生部分，其总体架构如图 2-7 所示。密钥扩展部分由密钥扩展运算单元和 XTime 运算单元组成，其中密钥扩展运算单元用于生成各轮子密钥，XTime 运算单元用于产生密钥扩展运算中的常数 Ron。轮变换部分包括普通轮变换运算单元和末轮变换运算单元，普通轮变换运算单元完成字节替换、行移位、列混合运算和密钥加运算，而末轮变换运算单元相较前者只缺少列混合运算。使能信号产生部分主要包括一个 4 位计数器和使能信号产生电路。AES 加密运算单元的信号接口如表 2-5 所示。

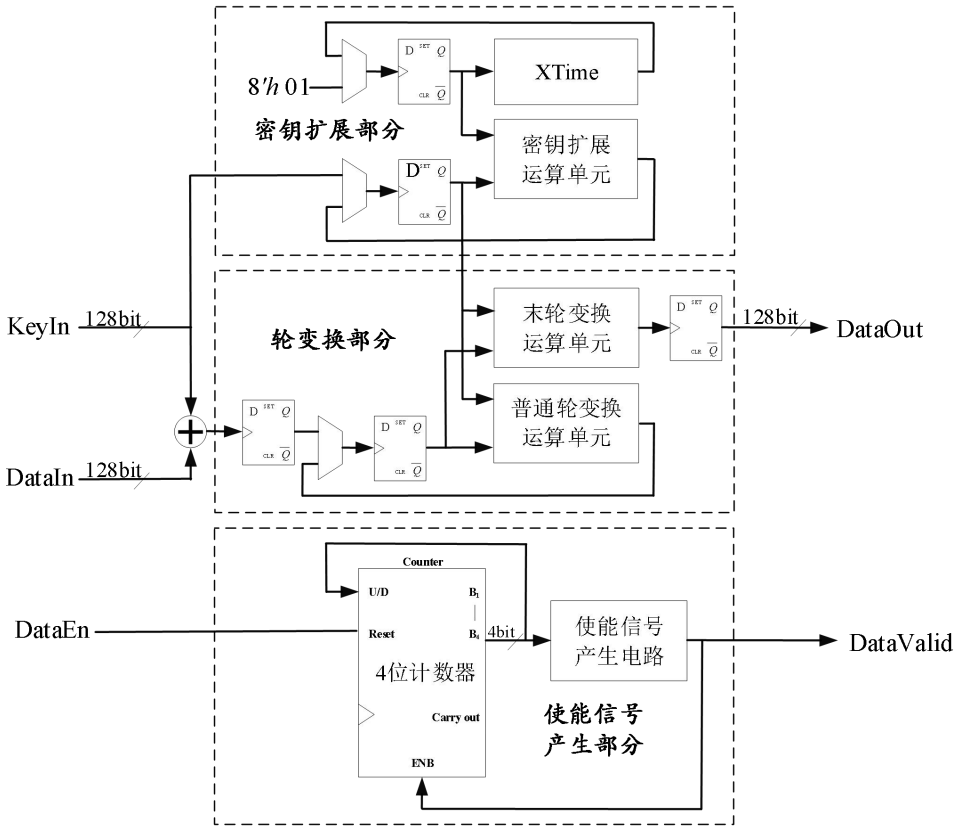


图 2-7　AES 加密运算单元总体架构图

表 2-5　AES 加密运算单元信号接口

	输入信号	数据位宽	信号含义	信号来源
1	clk	1	输入时钟	模块外部
2	DataEn	1	输入数据使能	输入数据位宽转换单元
3	DataIn	128	输入明文数据	输入数据位宽转换单元
4	KeyIn	128	输入密钥数据	模块外部
	输出信号	数据位宽	信号含义	信号目的地
1	DataValid	1	输出数据使能	输出数据位宽转换单元
2	DataOut	128	输出密文数据	输出数据位宽转换单元

2.3.2 轮变换电路实现

如前文所述，本次使用的是 AES-128，在 128 比特明文数据和 128 比特密钥数据输入之后，首先进行的是密钥加运算；接着进入普通轮变换运算单元，其中依次进行 S-盒、行移位、列混合、密钥加等运算；最后进入末轮变换运算单元。通常将 16 字节数据用状态矩阵形式来表示，现假设数据输入的状态矩阵为：

$$X=\begin{bmatrix} X_{0,0} & X_{0,1} & X_{0,2} & X_{0,3} \\ X_{1,0} & X_{1,1} & X_{1,2} & X_{1,3} \\ X_{2,0} & X_{2,1} & X_{2,2} & X_{2,3} \\ X_{3,0} & X_{3,1} & X_{3,2} & X_{3,3} \end{bmatrix} \tag{2-1}$$

其中 $X_{i,j}$ 为输入的字节变量。下面给出轮变换中各个运算电路介绍：

（1）S-盒（*SubBytes*）运算电路；该变换分别对状态的每一个字节进行替换，如图 2-8 所示。

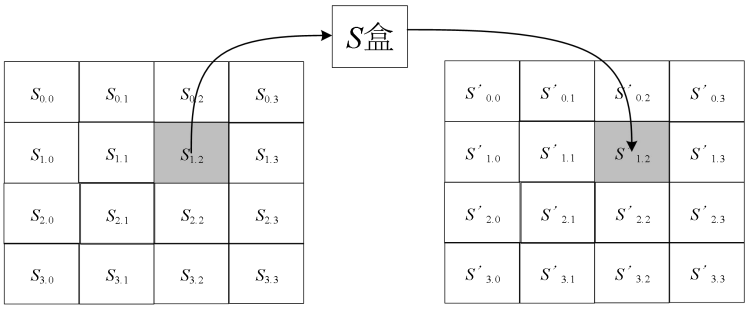


图 2-8　状态矩阵单个字节替换图

$$S=SubBytes(X):\begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{bmatrix}=\begin{bmatrix} S(X_{0,0}) & S(X_{0,1}) & S(X_{0,2}) & S(X_{0,3}) \\ S(X_{1,0}) & S(X_{1,1}) & S(X_{1,2}) & S(X_{1,3}) \\ S(X_{2,0}) & S(X_{2,1}) & S(X_{2,2}) & S(X_{2,3}) \\ S(X_{3,0}) & S(X_{3,1}) & S(X_{3,2}) & S(X_{3,3}) \end{bmatrix} \tag{2-2}$$

S-盒运算可用式 2-2 表示，其中 S 定义为字节替换运算之后的状态矩阵。针对不同优化目标，有不同的 S-盒电路可供选择，一般分为只读存储器（ROM）、查询表（LUT）、基于复合域运算。本文优先考虑电路运算速度，因此采用运算速度最高的 S-盒电路实现方法 LUT。

（2）行移位（*ShiftRows*）运算电路；该变换是将状态矩阵中每一行按照字节数进行循环左移运算，且一行的移位字节数有所不同，如图 2-9 所示。

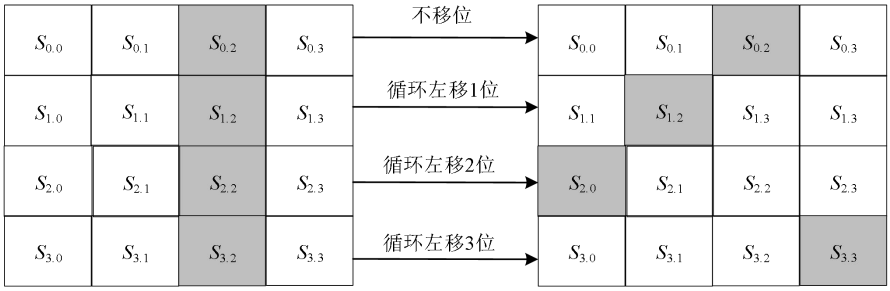


图 2-9　状态矩阵循环左移运算图

行移位运算可用式 2-3 表示，其中矩阵 H 定义为行移位运算后的状态矩阵。行移位可以通过总线顺序调换方式来实现，因而不需要消耗任何硬件资源。

$$H = ShiftRows(S):\begin{bmatrix} H_{0,0} & H_{0,1} & H_{0,2} & H_{0,3} \\ H_{1,0} & H_{1,1} & H_{1,2} & H_{1,3} \\ H_{2,0} & H_{2,1} & H_{2,2} & H_{2,3} \\ H_{3,0} & H_{3,1} & H_{3,2} & H_{3,3} \end{bmatrix} = \begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,1} & S_{1,2} & S_{1,3} & S_{1,0} \\ S_{2,2} & S_{2,3} & S_{2,0} & S_{2,1} \\ S_{3,3} & S_{3,0} & S_{3,1} & S_{3,2} \end{bmatrix} \tag{2-3}$$

（3）列混合（*MixColumns*）运算电路；该变换将状态矩阵每一列与一个矩阵执行乘法运算，如图 2-10 所示。将状态字节作为一个有限域上的元素来考虑，矩阵乘法即将每一个状态字节与多项式 $c(x) = 03x^3 + 01x^2 + 01x + 02$ 相乘并以 $x^4 + 1$ 为模。这个特定的多项式因子（以及所生成的矩阵）的选择基于宽轨迹（*wide-trail*）设计策略，它提供了对线性密码分析和差分密码分析的高抵抗能力。

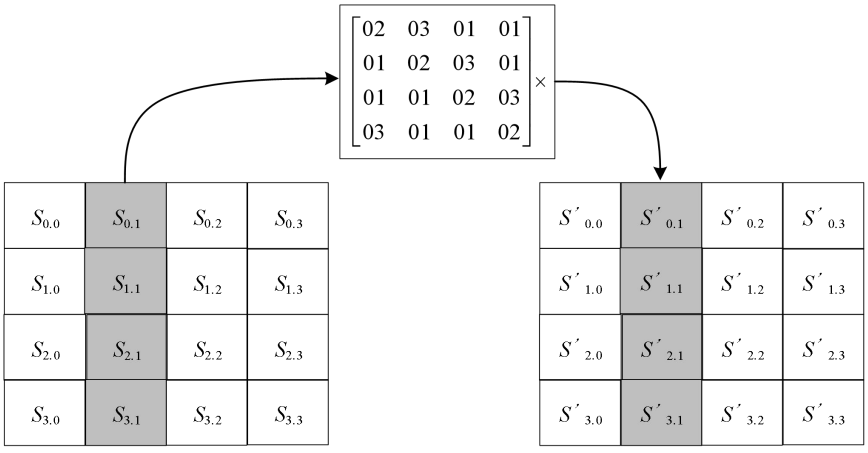


图 2-10　状态矩阵各列运算图

列混合运算可用式 2-4 表示，其中矩阵 I 定义为列混合运算后的状态矩阵，公式中的常向量{03}，{02}，{01}为十六进制表示。

$$I = MixColumns(H):$$

$$\begin{bmatrix} I_{0,0} & I_{0,1} & I_{0,2} & I_{0,3} \\ I_{1,0} & I_{1,1} & I_{1,2} & I_{1,3} \\ I_{2,0} & I_{2,1} & I_{2,2} & I_{2,3} \\ I_{3,0} & I_{3,1} & I_{3,2} & I_{3,3} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} H_{0,0} & H_{0,1} & H_{0,2} & H_{0,3} \\ H_{1,0} & H_{1,1} & H_{1,2} & H_{1,3} \\ H_{2,0} & H_{2,1} & H_{2,2} & H_{2,3} \\ H_{3,0} & H_{3,1} & H_{3,2} & H_{3,3} \end{bmatrix} \tag{2-4}$$

（4）密钥加（*AddRoundKey*）运算电路；该变换通常将轮密钥与状态矩阵进行位异或运算，如图 2-11 所示。初始的密钥加运算过程中所使用的轮密钥即为原始密钥，且轮密钥的长度等于数据分组长度。

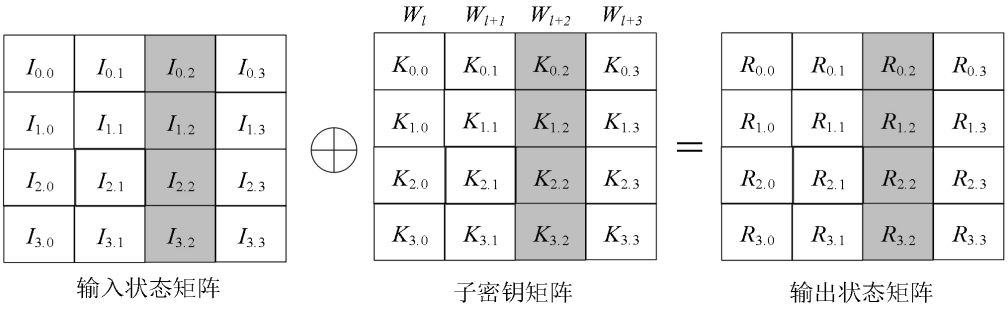


图 2-11　对应列数据密钥加运算图

密钥加运算可用式 2-5 表示，其中矩阵 *I* 是轮数据输入，矩阵 *K* 是密钥。

$$R = AddRoundkey(I, K) = I + K:$$

$$\begin{bmatrix} R_{0,0} & R_{0,1} & R_{0,2} & R_{0,3} \\ R_{1,0} & R_{1,1} & R_{1,2} & R_{1,3} \\ R_{2,0} & R_{2,1} & R_{2,2} & R_{2,3} \\ R_{3,0} & R_{3,1} & R_{3,2} & R_{3,3} \end{bmatrix} = \begin{bmatrix} I_{0,0} & I_{0,1} & I_{0,2} & I_{0,3} \\ I_{1,0} & I_{1,1} & I_{1,2} & I_{1,3} \\ I_{2,0} & I_{2,1} & I_{2,2} & I_{2,3} \\ I_{3,0} & I_{3,1} & I_{3,2} & I_{3,3} \end{bmatrix} + \begin{bmatrix} K_{0,0} & K_{0,1} & K_{0,2} & K_{0,3} \\ K_{1,0} & K_{1,1} & K_{1,2} & K_{1,3} \\ K_{2,0} & K_{2,1} & K_{2,2} & K_{2,3} \\ K_{3,0} & K_{3,1} & K_{3,2} & K_{3,3} \end{bmatrix} \tag{2-5}$$

根据公式(2-2)~(2-5)可得出轮变换公式为：

$$R = Round(X, K):$$

$$\begin{bmatrix} R_{0,0} & R_{0,1} & R_{0,2} & R_{0,3} \\ R_{1,0} & R_{1,1} & R_{1,2} & R_{1,3} \\ R_{2,0} & R_{2,1} & R_{2,2} & R_{2,3} \\ R_{3,0} & R_{3,1} & R_{3,2} & R_{3,3} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S(X_{0,0}) & S(X_{0,1}) & S(X_{0,2}) & S(X_{0,3}) \\ S(X_{1,1}) & S(X_{1,2}) & S(X_{1,3}) & S(X_{1,0}) \\ S(X_{2,2}) & S(X_{2,3}) & S(X_{2,0}) & S(X_{2,1}) \\ S(X_{3,3}) & S(X_{3,0}) & S(X_{3,1}) & S(X_{3,2}) \end{bmatrix} + \begin{bmatrix} K_{0,0} & K_{0,1} & K_{0,2} & K_{0,3} \\ K_{1,0} & K_{1,1} & K_{1,2} & K_{1,3} \\ K_{2,0} & K_{2,1} & K_{2,2} & K_{2,3} \\ K_{3,0} & K_{3,1} & K_{3,2} & K_{3,3} \end{bmatrix} \tag{2-6}$$

根据公式 2-6，矩阵 *R* 每一列还可以表达为线性方程形式：

$$\begin{cases} R_3 = \{02\}S(X_3) + \{01\}S(X_2) + \{01\}S(X_1) + \{03\}S(X_0) + K_3 \\ R_2 = \{03\}S(X_3) + \{02\}S(X_2) + \{01\}S(X_1) + \{01\}S(X_0) + K_2 \\ R_1 = \{01\}S(X_3) + \{03\}S(X_2) + \{02\}S(X_1) + \{01\}S(X_0) + K_1 \\ R_0 = \{01\}S(X_3) + \{01\}S(X_2) + \{03\}S(X_1) + \{02\}S(X_0) + K_0 \end{cases} \tag{2-7}$$

公式 2-7 为轮变换公式的一般形式，其中$\{R_3, R_2, R_1, R_0\}$为四个输出字节向量，$\{X_3, X_2, X_1, X_0\}$为四个输入字节，$\{K_3, K_2, K_1, K_0\}$为四个子密钥字节。根据公式 2-7 可构建普通轮变换电路结构如图 2-12 所示，图中的电路为 32 比特位宽，即可以处理一列状态矩阵数据，128 比特位宽的普通轮变换电路由四个该电路模块构成。

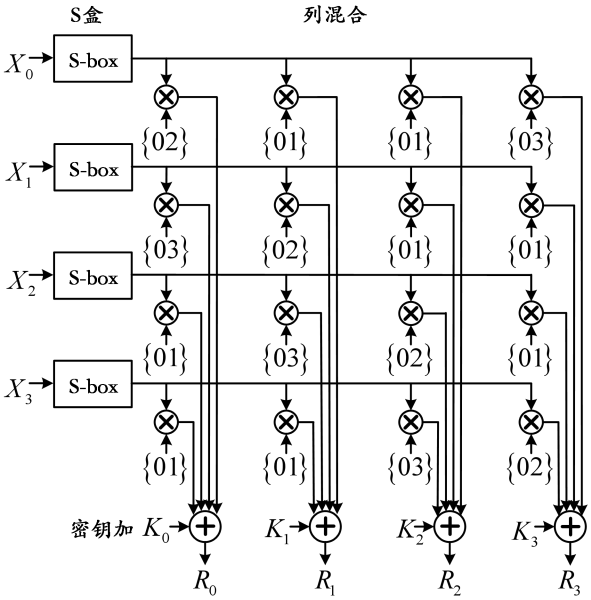


图 2-12　普通轮变换电路结构图

普通轮变换电路输入四个字节 $[X_0, X_1, X_2, X_3]$ 首先进行 S-盒运算，然后进行列混合运算，最后经过密钥加运算后输出四个字节 $[R_0, R_1, R_2, R_3]$。行移位运算在四个字节 $[X_0, X_1, X_2, X_3]$ 输入之前在状态矩阵中进行了调换，因此行移位运算在图 2-15 中没有体现。而末轮变换电路没有列混合运算，因此经过 S-盒运算之后直接进行密钥加运算。

2.3.3 使能信号产生电路实现

使能信号产生电路部分由两电路模块组成，一个是 0~11 计数器，另一个是使能信号产生的逻辑电路。AES 加密需要轮变换运算电路循环 10 次完成整个加密流程，在本文中，一个时钟完成一次轮变换运算（包括一次轮密钥扩展运算），因此需要 10 个时钟完成整个 AES 加密流程，外加输入数据（包括首次密钥加运算）1 个时钟，输出数据 1 个时钟，因此完成一组 128 比特数据加密总共需要 12 个时钟周期。使能信号产生逻辑电路在第 12 个脉冲输出一个有效信号，用于标识一组明文数据加密完成。

由上文可知，完成一组 128 比特数据的接收需要 32 个时钟，因此，AES 加密运算电路在下一组明文数据接收完成之前可以完成本次加密运算。计数器计数到 4′h0B 时，停止计数。下一组明文数据的使能信号作为计数器同步清零信号，将计数器进行清零，并开始下一轮计数。

2.4 电路实现结果

本文内容在 XC7VX690T 系列 FPGA 上实现，具体采用 xc7vx690tffg1157-1 芯片来进行仿真验证。根据以上方案进行硬件描述语言编程，再利用 Vivado 工具中的 Simulation 仿真工具，对搭建好的 AES 加密算法硬件架构进行波形仿真。由前文可知输入数据再连续输入 1696 个有效数据（848 字节）后，需要保持 128 比特密钥数据有效，且输出数据和输入数据之间应间隔 45 时钟，如图 2-13 所示。

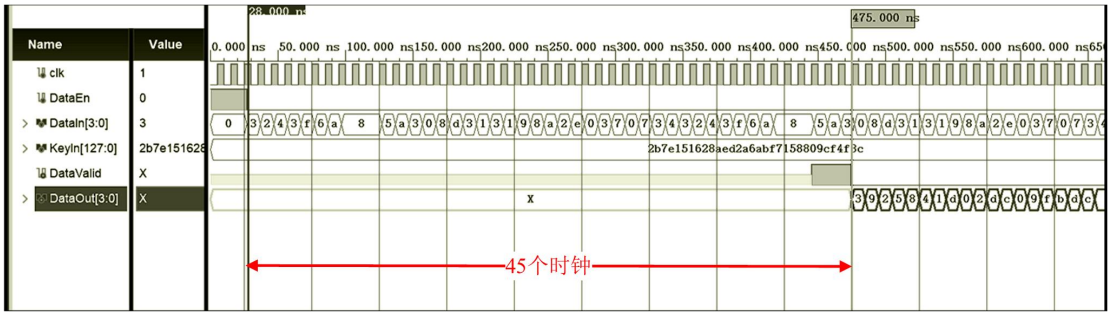


图 2-13　密钥数据有效仿真测试图

输入数据每 32 个时钟产生一组 16 字节（128 比特）明文数据送入 AES 加密单元中进行加密，则 848 字节有效数据总共需产生 53 组明文数据，如图 2-14。在一组明文数据产生后，还需要经过 12 时钟来产生密文数据，如图 2-15 所示。

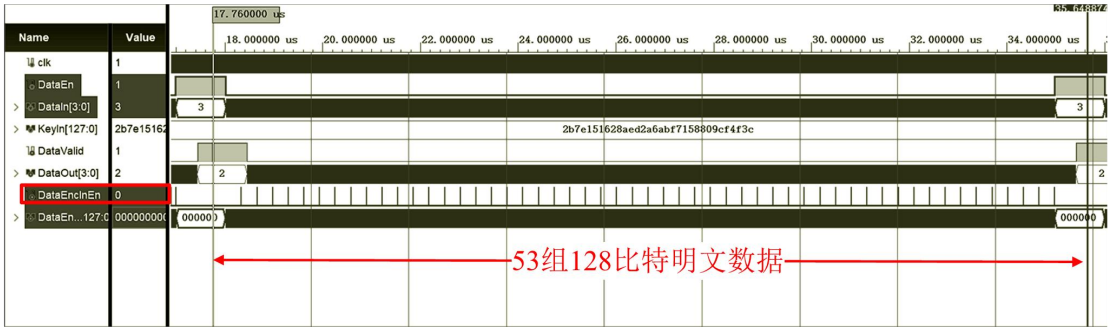


图 2-14　848 字节输入数据仿真测试图

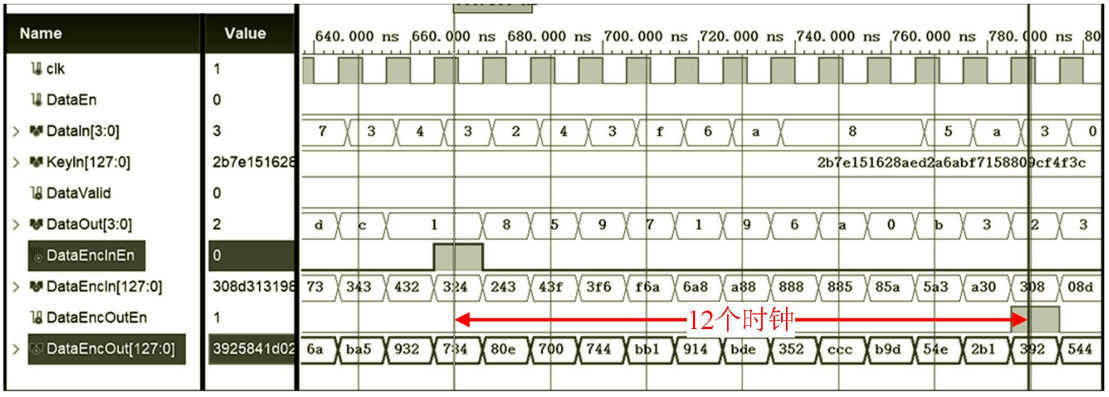


图 2-15　一组明文数据加密过程仿真测试图

在波形仿真验证之后，设置时钟约束条件，经电路综合、布局布线实现之后得到电路延时信息如图 2-16 所示。

Clock Summary

Clock	Waveform(ns)	Period(ns)	Frequency(MHz)
clk	{0.000 5.000}	10.000	100.000

Intra Clock Table

Clock	WNS(ns)	TNS(ns)	TNS Failing Endpoints	TNS Total Endpoints	WHS(ns)
clk	5.490	0.000	0	801	0.097

图 2-16　AES 加密运算单元延时信息图

由图 2-17 可知最差负时序裕量（WNS）为 5.49ns，电路运行最高频率为：

$$f_{\max}=\frac{1}{1/f_{约束}-WNS}=\frac{1}{10-5.49}=221.73MHz \tag{2-14}$$

AES 加密运算单元实现后的功耗信息如图 2-17 所示，可知电路总功耗约为 0.4W。AES 加密运算单元实现后的资源消耗信息如图 2-18 所示，可知电路总共使用 447 个 Slice，只占 Slice 总数的 0.41%。

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.4 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**25.6°C**
Thermal Margin:	59.4°C (40.7 W)
Effective ϑJA:	1.4°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

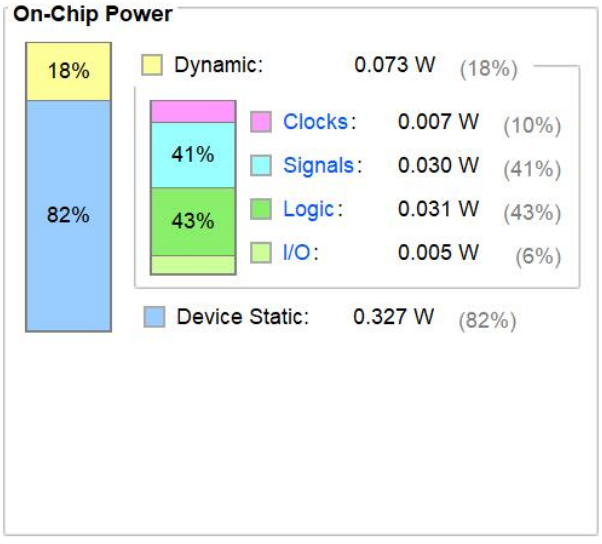


图 2-17　AES 加密运算单元功耗信息图

2. Slice Logic Distribution

Site Type	Used	Fixed	Available	Util%
Slice	447	0	108300	0.41
SLICEL	278	0		
SLICEM	169	0		
LUT as Logic	1433	0	433200	0.33
using O5 output only	0			
using O6 output only	1323			
using O5 and O6	110			
LUT as Memory	2	0	174200	<0.01
LUT as Distributed RAM	0	0		
LUT as Shift Register	2	0		
using O5 output only	1			
using O6 output only	1			
using O5 and O6	0			
Slice Registers	788	0	866400	0.09
Register driven from within the Slice	648			
Register driven from outside the Slice	140			
LUT in front of the register is unused	108			
LUT in front of the register is used	32			
Unique Control Sets	4		108300	<0.01

图 2-18　AES 加密运算单元硬件架构资源消耗图

2.5 本章小结

本章主要研究了星载高速 AES 加密电路的设计与实现。首先介绍了加密电路的总体设计方案。然后，采用分单元设计的方法，对电路中各个环节的设计及优化方法进行了详细的介绍，重点介绍了数据位宽转换和加密运算结构的设计。最后，采用 Vivado 开发工具对电路的功能进行仿真验证，并利用工具综合实现得到底层资源报告，最终成功应用于实际。

第 3 章 AES 解密设计实现

本章基于卫星加密通信领域中数据信息解密功能要求，利用地检设备 FPGA 设计实现 AES-128 解密算法。在设计 AES 硬件解密电路模块中，将解密模块分成“输入数据位宽转换单元”、“AES 解密运算单元”、“输出数据位宽转换单元”、“数据使能延时单元”、“密钥变动检测单元”、“密钥扩展运算单元”六个部分，保证密钥与密文对应，再利用 Vivado 开发工具分模块分单元对电路进行波形仿真，以 XC7VX690T 系列 FPGA 运行测试满足实际应用要求。

3.1 AES 解密电路模块

3.1.1 总体设计

AES 解密电路模块如图 3-1 所示，是由“输入数据位宽转换单元”、“AES 解密运算单元”、“输出数据位宽转换单元”、“数据使能延时单元”、“密钥变动检测单元”、“密钥扩展运算单元”等六个部分组成。其中，“输入数据位宽转换单元”、“输出数据位宽转换单元”、“数据使能延时单元”与 AES 加密电路模块功能和设计方法相同，本节不再介绍。

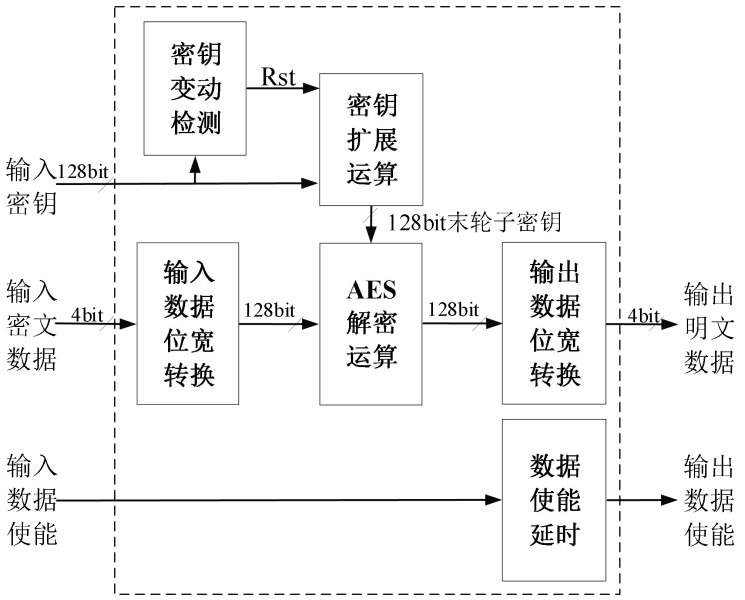


图 3-1　AES 解密电路模块总体架构图

与 AES 加密电路模块相比，解密电路模块中多了一个“密钥变动检测单元”和一个“密钥扩展运算单元”。这是因为 AES 解密运算首先使用第 10 轮子密钥 K_{10}，如图 3-1 所示，然后由 K_{10} 根据逆密钥扩展算法依次产生 K_9~K_0。因此需要

一个“密钥扩展运算单元”在 AES 解密运算之前将初始密钥 K_0 扩展成 K_{10} 输入到 AES 解密模块中待用。“密钥扩展运算单元”产生 K_{10} 之后进入待机状态，不再运行，因此添加一个“密钥变动检测单元”，当输入新的密钥 K_0 时，“密钥变动检测单元”输出一个同步复位信号，促使“密钥扩展运算单元”重新运行，产生新的 K_{10}。

3.1.2 密钥扩展运算实现

密钥扩展运算解密电路包含“密钥变动检测单元”和“密钥扩展运算单元”。“密钥变动检测电路”是为了检测密钥数据是否发送变动，如果发生变动，则促使密钥扩展电路进行密钥扩展运算，如图 3-2 所示。

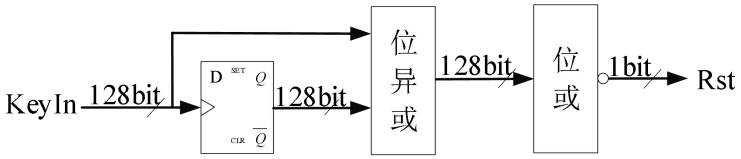


图 3-2　密钥变动检测电路结构图

在 128 比特输入后，经过一个时钟延时，延时信号和当前信号进行比较，即进行位异或运算。当密钥发生变动时，位异或运算输出不为 0，则 128 比特位或运算输出 1，经过反相后输入作为密钥扩展单元的同步复位信号。密钥变动检测单元的信号接口如表 3-1 所示。

表 3-1　密钥变动检测单元信号接口

	输入信号	数据位宽	信号含义	信号来源
1	clk	1	输入时钟	模块外部
2	KeyIn	128	输入密钥数据	模块外部
	输出信号	数据位宽	信号含义	信号目的地
1	KeyExpRst	1	密钥扩展单元复位信号	密钥扩展运算单元

“密钥扩展运算单元”与 AES 加密运算单元中的相应电路实现方法相同，其信号接口如表 3-2 所示。

表 3-2　密钥扩展单元的信号接口

	输入信号	数据位宽	信号含义	信号来源
1	clk	1	输入时钟	模块外部
2	rst	1	输入同步复位信号	密钥变动检测单元
3	KeyIn	128	输入密钥数据	模块外部
	输出信号	数据位宽	信号含义	信号目的地
1	KeyOut	128	输出末轮子密钥	AES 解密运算单元

3.2 AES 解密运算单元

3.2.1 总体架构

AES 解密运算单元是 AES 加密运算的逆运算，则 AES 解密运算单元总体架构与 AES 加密运算单元相同，如图 3-3 所示。AES 解密运算单元也分成三部分：密钥扩展、轮变换和使能信号产生。密钥扩展部分由逆密钥扩展运算单元和×{08}运算单元组成，其中逆密钥扩展运算单元用于生成各轮子密钥，×{08}运算单元用于产生逆密钥扩展运算中的常数 Ron。轮变换部分包括普通轮变换运算单元和末轮变换运算单元。使能信号产生部分包括一个 4 位计数器和使能信号产生电路。这些与在 AES 加密运算单元中实现方法相同，所以本节不再叙述。

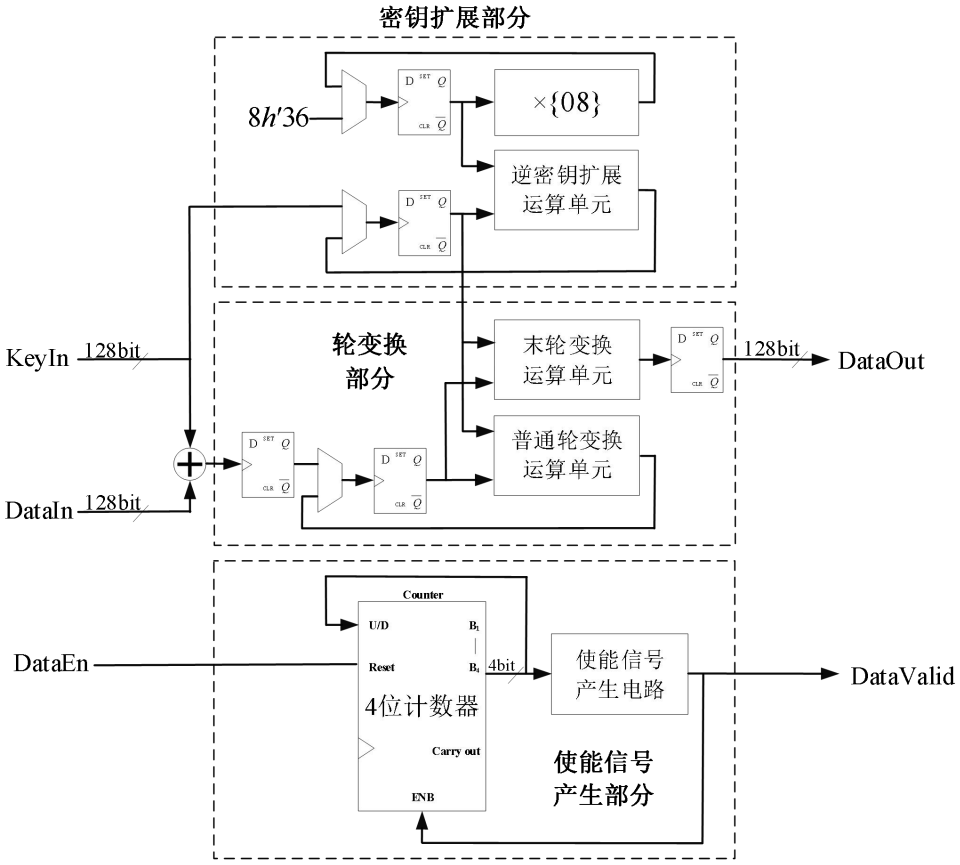


图 3-3　AES 解密运算单元总体架构图

3.2.2 轮变换电路实现

如图 3-3 所示，128 比特明文数据和 128 比特密钥数据输入之后首先进行一次密钥加运算。经过第一次密钥加运算之后，数据进入普通轮变换运算单元。如前文所述，普通轮变换依次进行逆行移位运算、逆 S-盒运算、密钥加运算和逆列混合运算等运算，而末轮变换没有逆列混合运算。在 AES 解密运算过程中，16 字节数据也表示为状态矩阵形式，假设轮变换输入的状态矩阵为：

$$\tilde{X}=\begin{bmatrix}\tilde{X}_{0,0} & \tilde{X}_{0,1} & \tilde{X}_{0,2} & \tilde{X}_{0,3}\\ \tilde{X}_{1,0} & \tilde{X}_{1,1} & \tilde{X}_{1,2} & \tilde{X}_{1,3}\\ \tilde{X}_{2,0} & \tilde{X}_{2,1} & \tilde{X}_{2,2} & \tilde{X}_{2,3}\\ \tilde{X}_{3,0} & \tilde{X}_{3,1} & \tilde{X}_{3,2} & \tilde{X}_{3,3}\end{bmatrix} \tag{3-1}$$

其中 $\tilde{X}_{i,j}$ 为输入的字节变量。下面给出轮变换中各个运算的表达式。

（1）逆行移位（*Inverse ShiftRows*）运算电路；该变换是行移位运算的逆运算，则将状态矩阵中每一行按照字节数循环进行右移运算，如图 3-4 所示。

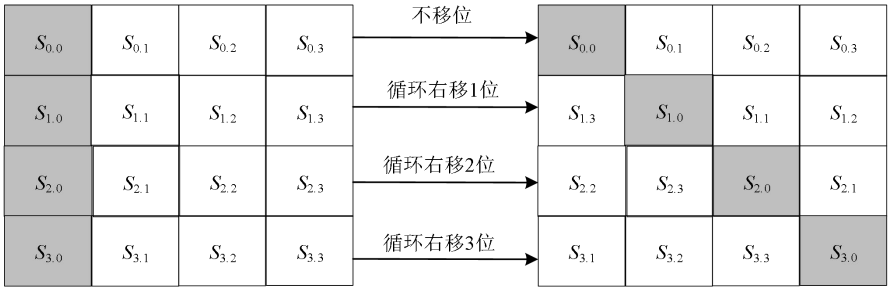


图 3-4　状态矩阵循环右移运算图

逆行移位运算可用式 3-2 表示，其中矩阵 $\tilde{H}$ 定义为逆行移位运算后的状态矩阵。逆行移位可以通过总线顺序调换来实现，因而不需要消耗任何硬件资源。

$$\tilde{H}=InvShiftRows\left(\tilde{I}\right):\begin{bmatrix}\tilde{H}_{0,0} & \tilde{H}_{0,1} & \tilde{H}_{0,2} & \tilde{H}_{0,3}\\ \tilde{H}_{1,0} & \tilde{H}_{1,1} & \tilde{H}_{1,2} & \tilde{H}_{1,3}\\ \tilde{H}_{2,0} & \tilde{H}_{2,1} & \tilde{H}_{2,2} & \tilde{H}_{2,3}\\ \tilde{H}_{3,0} & \tilde{H}_{3,1} & \tilde{H}_{3,2} & \tilde{H}_{3,3}\end{bmatrix}=\begin{bmatrix}\tilde{X}_{0,0} & \tilde{X}_{0,1} & \tilde{X}_{0,2} & \tilde{X}_{0,3}\\ \tilde{X}_{1,3} & \tilde{X}_{1,0} & \tilde{X}_{1,1} & \tilde{X}_{1,2}\\ \tilde{X}_{2,2} & \tilde{X}_{2,3} & \tilde{X}_{2,0} & \tilde{X}_{2,1}\\ \tilde{X}_{3,1} & \tilde{X}_{3,2} & \tilde{X}_{3,3} & \tilde{X}_{3,0}\end{bmatrix} \tag{3-2}$$

（2）逆 S-盒（*Inverse SubBytes*）运算电路；同样按照图 2-8 方式将输入状态矩阵中的 16 个字节一一进行替换。逆 S-盒运算可用式 3-3 表示，其中 $\tilde{S}$ 定义为逆字节替换运算之后的状态矩阵。本文中逆 S-盒电路同样采用运算速度最高的电路实现方法—查询表（LUT）。

$$\tilde{S}=InvSubBytes\left(\tilde{H}\right):\begin{bmatrix}\tilde{S}_{0,0} & \tilde{S}_{0,1} & \tilde{S}_{0,2} & \tilde{S}_{0,3}\\ \tilde{S}_{1,0} & \tilde{S}_{1,1} & \tilde{S}_{1,2} & \tilde{S}_{1,3}\\ \tilde{S}_{2,0} & \tilde{S}_{2,1} & \tilde{S}_{2,2} & \tilde{S}_{2,3}\\ \tilde{S}_{3,0} & \tilde{S}_{3,1} & \tilde{S}_{3,2} & \tilde{S}_{3,3}\end{bmatrix}=\begin{bmatrix}S^{-1}\left(\tilde{H}_{0,0}\right) & S^{-1}\left(\tilde{H}_{0,1}\right) & S^{-1}\left(\tilde{H}_{0,2}\right) & S^{-1}\left(\tilde{H}_{0,3}\right)\\ S^{-1}\left(\tilde{H}_{1,0}\right) & S^{-1}\left(\tilde{H}_{1,1}\right) & S^{-1}\left(\tilde{H}_{1,2}\right) & S^{-1}\left(\tilde{H}_{1,3}\right)\\ S^{-1}\left(\tilde{H}_{2,0}\right) & S^{-1}\left(\tilde{H}_{2,1}\right) & S^{-1}\left(\tilde{H}_{2,2}\right) & S^{-1}\left(\tilde{H}_{2,3}\right)\\ S^{-1}\left(\tilde{H}_{3,0}\right) & S^{-1}\left(\tilde{H}_{3,1}\right) & S^{-1}\left(\tilde{H}_{3,2}\right) & S^{-1}\left(\tilde{H}_{3,3}\right)\end{bmatrix} \tag{3-3}$$

（3）密钥加（*AddRoundKey*）运算电路；由于密钥加运算的逆运算为其本身，所以解密运算中的密钥加运算与加密运算中的密钥加运算相同。解密运算中的密钥加运算可用式 3-4 表示，其中矩阵 $\tilde{K}$ 为子密钥矩阵，矩阵 $\tilde{I}$ 定义为密钥加运算后的状态矩阵。

$$\tilde{I} = AddRoundKey\left(\tilde{S} + \tilde{K}\right):$$
$$\begin{bmatrix} \tilde{I}_{0,0} & \tilde{I}_{0,1} & \tilde{I}_{0,2} & \tilde{I}_{0,3} \\ \tilde{I}_{1,0} & \tilde{I}_{1,1} & \tilde{I}_{1,2} & \tilde{I}_{1,3} \\ \tilde{I}_{2,0} & \tilde{I}_{2,1} & \tilde{I}_{2,2} & \tilde{I}_{2,3} \\ \tilde{I}_{3,0} & \tilde{I}_{3,1} & \tilde{I}_{3,2} & \tilde{I}_{3,3} \end{bmatrix} = \begin{bmatrix} \tilde{S}_{0,0} & \tilde{S}_{0,1} & \tilde{S}_{0,2} & \tilde{S}_{0,3} \\ \tilde{S}_{1,0} & \tilde{S}_{1,1} & \tilde{S}_{1,2} & \tilde{S}_{1,3} \\ \tilde{S}_{2,0} & \tilde{S}_{2,1} & \tilde{S}_{2,2} & \tilde{S}_{2,3} \\ \tilde{S}_{3,0} & \tilde{S}_{3,1} & \tilde{S}_{3,2} & \tilde{S}_{3,3} \end{bmatrix} + \begin{bmatrix} \tilde{K}_{0,0} & \tilde{K}_{0,1} & \tilde{K}_{0,2} & \tilde{K}_{0,3} \\ \tilde{K}_{1,0} & \tilde{K}_{1,1} & \tilde{K}_{1,2} & \tilde{K}_{1,3} \\ \tilde{K}_{2,0} & \tilde{K}_{2,1} & \tilde{K}_{2,2} & \tilde{K}_{2,3} \\ \tilde{K}_{3,0} & \tilde{K}_{3,1} & \tilde{K}_{3,2} & \tilde{K}_{3,3} \end{bmatrix} \tag{3-4}$$

（4）逆列混合（*Inverse MixColumns*）运算电路；该运算与列混合运算不同在于所乘的常数矩阵不同，其运算可用式 3-5 表示，其中矩阵 $\tilde{R}$ 定义为逆列混合运算后的状态矩阵，公式中的常向量{09}，{0b}，{0d}，{0e}为十六进制表示。

$$\tilde{R} = InvMixColumns\left(\tilde{I}\right):$$
$$\begin{bmatrix} \tilde{R}_{0,0} & \tilde{R}_{0,1} & \tilde{R}_{0,2} & \tilde{R}_{0,3} \\ \tilde{R}_{1,0} & \tilde{R}_{1,1} & \tilde{R}_{1,2} & \tilde{R}_{1,3} \\ \tilde{R}_{2,0} & \tilde{R}_{2,1} & \tilde{R}_{2,2} & \tilde{R}_{2,3} \\ \tilde{R}_{3,0} & \tilde{R}_{3,1} & \tilde{R}_{3,2} & \tilde{R}_{3,3} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} \tilde{I}_{0,0} & \tilde{I}_{0,1} & \tilde{I}_{0,2} & \tilde{I}_{0,3} \\ \tilde{I}_{1,0} & \tilde{I}_{1,1} & \tilde{I}_{1,2} & \tilde{I}_{1,3} \\ \tilde{I}_{2,0} & \tilde{I}_{2,1} & \tilde{I}_{2,2} & \tilde{I}_{2,3} \\ \tilde{I}_{3,0} & \tilde{I}_{3,1} & \tilde{I}_{3,2} & \tilde{I}_{3,3} \end{bmatrix} \tag{3-5}$$

根据公式(3-2)~(3-5)可得出轮变换公式为：

$$\tilde{R} = Round\left(\tilde{X}, \tilde{K}\right): \begin{bmatrix} \tilde{R}_{0,0} & \tilde{R}_{0,1} & \tilde{R}_{0,2} & \tilde{R}_{0,3} \\ \tilde{R}_{1,0} & \tilde{R}_{1,1} & \tilde{R}_{1,2} & \tilde{R}_{1,3} \\ \tilde{R}_{2,0} & \tilde{R}_{2,1} & \tilde{R}_{2,2} & \tilde{R}_{2,3} \\ \tilde{R}_{3,0} & \tilde{R}_{3,1} & \tilde{R}_{3,2} & \tilde{R}_{3,3} \end{bmatrix} =$$
$$\begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \left(\begin{bmatrix} S^{-1}\left(\tilde{X}_{0,0}\right) & S^{-1}\left(\tilde{X}_{0,1}\right) & S^{-1}\left(\tilde{X}_{0,2}\right) & S^{-1}\left(\tilde{X}_{0,3}\right) \\ S^{-1}\left(\tilde{X}_{1,3}\right) & S^{-1}\left(\tilde{X}_{1,0}\right) & S^{-1}\left(\tilde{X}_{1,1}\right) & S^{-1}\left(\tilde{X}_{1,2}\right) \\ S^{-1}\left(\tilde{X}_{2,2}\right) & S^{-1}\left(\tilde{X}_{2,3}\right) & S^{-1}\left(\tilde{X}_{2,0}\right) & S^{-1}\left(\tilde{X}_{2,1}\right) \\ S^{-1}\left(\tilde{X}_{3,1}\right) & S^{-1}\left(\tilde{X}_{3,2}\right) & S^{-1}\left(\tilde{X}_{3,3}\right) & S^{-1}\left(\tilde{X}_{3,0}\right) \end{bmatrix} + \begin{bmatrix} K_{0,0} & K_{0,1} & K_{0,2} & K_{0,3} \\ K_{1,0} & K_{1,1} & K_{1,2} & K_{1,3} \\ K_{2,0} & K_{2,1} & K_{2,2} & K_{2,3} \\ K_{3,0} & K_{3,1} & K_{3,2} & K_{3,3} \end{bmatrix} \right) \tag{3-6}$$

根据公式 3-6，矩阵 *R* 每一列还可以表达为线性方程形式：

$$\begin{cases} \tilde{R}_0 = \{0e\}\left(S^{-1}\left(\tilde{X}_0\right) + \tilde{K}_0\right) + \{0b\}\left(S^{-1}\left(\tilde{X}_1\right) + \tilde{K}_1\right) + \{0d\}\left(S^{-1}\left(\tilde{X}_2\right) + \tilde{K}_2\right) + \{09\}\left(S^{-1}\left(\tilde{X}_3\right) + \tilde{K}_3\right) \\ \tilde{R}_1 = \{09\}\left(S^{-1}\left(\tilde{X}_0\right) + \tilde{K}_0\right) + \{0e\}\left(S^{-1}\left(\tilde{X}_1\right) + \tilde{K}_1\right)_1 + \{0b\}\left(S^{-1}\left(\tilde{X}_2\right) + \tilde{K}_2\right) + \{0d\}\left(S^{-1}\left(\tilde{X}_3\right) + \tilde{K}_3\right) \\ \tilde{R}_2 = \{0d\}\left(S^{-1}\left(\tilde{X}_0\right) + \tilde{K}_0\right) + \{09\}\left(S^{-1}\left(\tilde{X}_1\right) + \tilde{K}_1\right) + \{0e\}\left(S^{-1}\left(\tilde{X}_2\right) + \tilde{K}_2\right) + \{0b\}\left(S^{-1}\left(\tilde{X}_3\right) + \tilde{K}_3\right) \\ \tilde{R}_3 = \{0b\}\left(S^{-1}\left(\tilde{X}_0\right) + \tilde{K}_0\right) + \{0d\}\left(S^{-1}\left(\tilde{X}_1\right) + \tilde{K}_1\right) + \{09\}\left(S^{-1}\left(\tilde{X}_2\right) + \tilde{K}_2\right) + \{0e\}\left(S^{-1}\left(\tilde{X}_3\right) + \tilde{K}_3\right) \end{cases} \tag{3-7}$$

公式 3-7 为解密运算轮变换公式的一般形式。根据公式 3-7 可构建解密运算普通轮变换电路结构如图 3-5 所示，图中的电路为 32 比特位宽，即可以处理一列状态矩阵数据，128 比特位宽的普通轮变换电路则需要由四个图 3-5 所示的电路模块构成。

在解密运算中，普通轮变换电路输入四个字节 $[\tilde{X}_0, \tilde{X}_1, \tilde{X}_2, \tilde{X}_3]$，首先进行的是逆 S-盒运算，然后是密钥加运算，最后经过逆列混合运算后输出四个字节 $[\tilde{R}_0, \tilde{R}_1, \tilde{R}_2, \tilde{R}_3]$。行移位运算在四个字节 $[\tilde{X}_0, \tilde{X}_1, \tilde{X}_2, \tilde{X}_3]$ 输入之前在状态矩阵

中进行了调换，因此行移位运算在图 3-5 中没有体现。末轮变换电路没有逆列混合运算，因此经过逆 S-盒运算和密钥加运算之后直接输出。

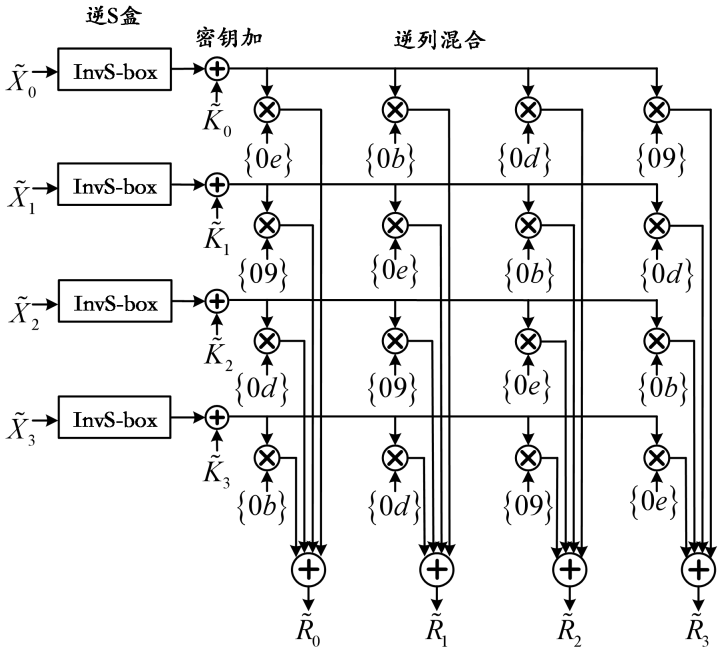


图 3-5　解密运算中的普通轮变换电路结构图

3.2.3 电路实现结果

本文内容在 XC7VX690T 系列 FPGA 上实现，具体采用 xc7vx690tffg1157-1 芯片来进行仿真验证。根据以上方案进行硬件描述语言编程，再利用 Vivado 工具中的 Simulation 仿真工具，对搭建好的 AES 加密算法硬件架构进行波形仿真。由前文可知数据使能低电平有效，当数据使能信号为低电平时，4 比特数据有效，而 128 比特密钥数据要至少提前 12 个时钟有效，以便于在 AES 解密之前产生第 10 轮子密钥 K_{10}。当密钥数据发生变动后，第 12 时钟产生相应的第 10 轮子密钥 K_{10}，如图 3-6 所示。

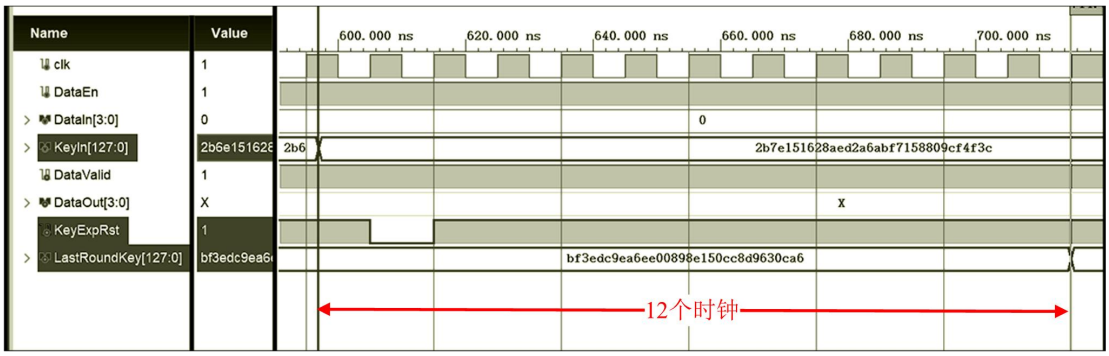


图 3-6　解密前 K_{10} 的生成仿真测试图

输入数据每 32 个时钟产生一组 16 字节（128 比特）密文数据送入 AES 解密单元中进行解密，如图 3-7。当产生一组密文数据后，经过 12 时钟后产生相应明文数据，即 AES 解密单元需要 12 个时钟完成一组明文数据加密，如图 3-8。

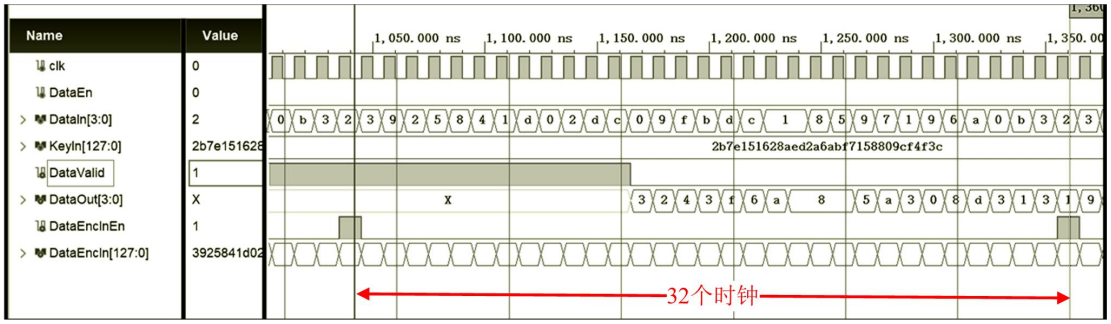


图 3-7　解密前数据处理仿真测试图

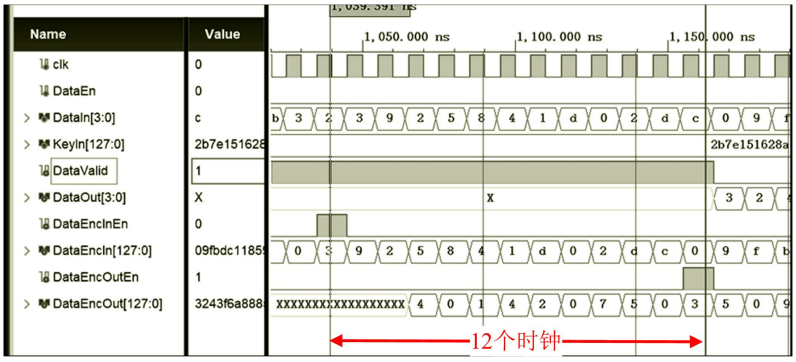


图 3-8　解密运算单元仿真测试图

在波形仿真验证之后，设置时钟约束条件，经电路综合、布局布线实现之后得到电路延时信息如图 3-9 所示。

```
------------------------------------------------------------------------------------------
| Clock Summary
| -------------
------------------------------------------------------------------------------------------

Clock  Waveform(ns)       Period(ns)      Frequency(MHz)
-----  -------------      ----------      --------------
clk    {0.000 5.000}      10.000          100.000


------------------------------------------------------------------------------------------
| Intra Clock Table
| -----------------
------------------------------------------------------------------------------------------

Clock             WNS(ns)      TNS(ns)  TNS Failing Endpoints  TNS Total Endpoints
-----             -------      -------  ---------------------  -------------------
clk                 3.947        0.000                      0                 1338
```

图 3-9　AES 解密运算单元实现后的延时信息图

由图 3-9 可知最差负时序裕量（WNS）为 3.947ns，电路运行最高频率为：

$$f_{\max}=\frac{1}{1/f_{约束}-WNS}=\frac{1}{10-3.947}=165.21MHz \tag{3-8}$$

AES 加密运算单元实现后的功耗信息如图 3-10 所示，可知电路总功耗约为 0.39W。AES 加密运算单元实现后的资源消耗信息如图 3-11 所示，可知电路总共使用 845 个 Slice，只占 Slice 总数的 1.11%。

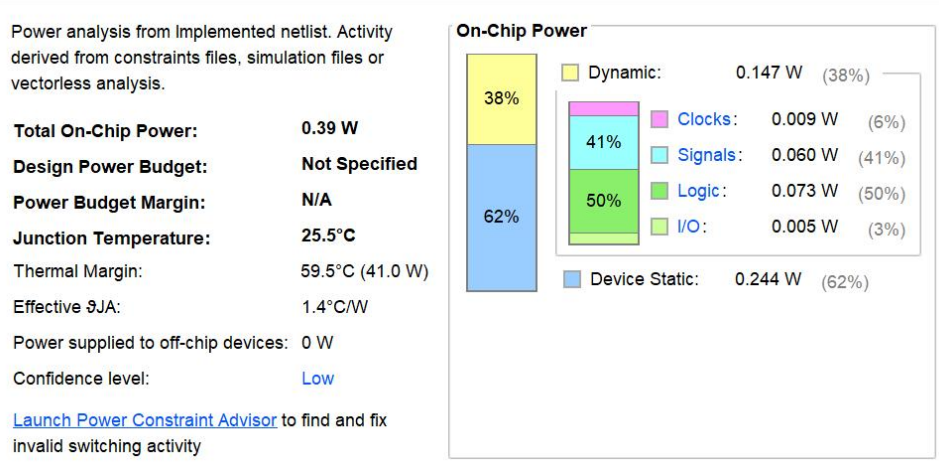


图 3-10　AES 解密运算单元实现后的功耗信息图

Site Type	Used	Fixed	Available	Util%
Slice	845	0	75900	1.11
SLICEL	446	0		
SLICEM	399	0		
LUT as Logic	2921	0	303600	0.96
using O5 output only	0			
using O6 output only	2278			
using O5 and O6	643			
LUT as Memory	2	0	130800	<0.01
LUT as Distributed RAM	0	0		
LUT as Shift Register	2	0		
using O5 output only	1			
using O6 output only	1			
using O5 and O6	0			
Slice Registers	1313	0	607200	0.22
Register driven from within the Slice	970			
Register driven from outside the Slice	343			
LUT in front of the register is unused	110			
LUT in front of the register is used	233			
Unique Control Sets	6		75900	<0.01

图 3-11　AES 解密运算单元实现后的资源消耗信息

3.3 本章小结

本章利用地检设备 FPGA 设计实现 AES-128 解密电路。首先介绍了 AES 解密电路的总体设计方案，然后对解密情况下运算过程作出详细的阐述，并采用分模块分单元设计的方法，对电路中各个环节的设计进行了详细的介绍，最后利用 Vivado 开发工具对电路的功能进行仿真验证，采用 XC7VX690T 系列 FPGA 运行测试并成功应用于实际。

第 4 章 AES 密钥扩展优化研究

AES 密钥扩展单元一般为链式结构，这会导致其关键路径长度较长，从而使 AES 密码算法难以高速实现。本章基于 AES 密钥扩展的一般表达式，提出了一种构建 AES 密钥扩展单元低延迟电路结构的构建方法。首先通过延迟感知公共项消除算法（Delay-Aware Common Subexpression Elimination, DACSE）来提取逻辑表达式中的公共项（Common Subexpression, CS），接着利用延迟驱动二叉树结构（Delayed-Driven Binary Tree, DDBT）构建低延迟密钥扩展单元，最后据此设计出 AES 加解密密钥扩展单元，以降低电路延迟并实现吞吐量饱和。

4.1 基于链式结构 AES 密钥扩展单元

AES 加解密是基于轮变换进行操作的，而轮变换的次数取决于密钥长度。本文以 AES-128 为例，来说明低延迟密钥扩展单元结构的构造方法，并且该构造方法可以很容易地扩展到 AES-192 和 AES-256 密钥算法。

4.1.1 AES 加密密钥扩展单元

在 AES-128 加密运算过程中，每轮使用的 128 位子密钥是根据密钥扩展算法从上一轮子密钥中获得的，每轮的子密钥可以表示为 4×4 字节的矩阵，子密钥矩阵如式 4-1 所示。

$$\begin{bmatrix} k_{0,0}^{i} & k_{0,1}^{i} & k_{0,2}^{i} & k_{0,3}^{i} \\ k_{1,0}^{i} & k_{1,1}^{i} & k_{1,2}^{i} & k_{1,3}^{i} \\ k_{2,0}^{i} & k_{2,1}^{i} & k_{2,2}^{i} & k_{2,3}^{i} \\ k_{3,0}^{i} & k_{3,1}^{i} & k_{3,2}^{i} & k_{3,3}^{i} \end{bmatrix} \Rightarrow \begin{bmatrix} K_0^i & K_1^i & K_2^i & K_3^i \end{bmatrix} \tag{4-1}$$

其中 $k_{m,n}^{i}$ 是第 i 轮子密钥矩阵第 n 列第 m 行的子密钥字节，K_n^i 是第 i 轮子密钥矩阵第 n 列的子密钥列向量，K_n^i 向量是一个 32 位词。

密钥扩展算法通常可用一组线性方程组来表示，其第 i 轮子密钥的扩展运算表达式可由式 4-2 表示。其中 $S_W = SubWord\left(RotWord\left(K_3^{i-1}\right)\right)$；*RotWord* 是字节循环左移运算，即输入四个字节为[a_0, a_1, a_2, a_3]，而输出四个字节为[a_1, a_2, a_3, a_0]，且该运算不占用任何逻辑资源；*SubWord* 是 32 比特字替换运算，等同于四个字节的 *SubBytes* 操作，并且加法运算被定义为 32 位异或运算；$R_c = Rcon(i)$ 为常量，*Rcon(i)*还可以写成式 4-3，且{00}和{02}都是 $GF(2^8)$域中的元素。

$$\begin{cases} K_0^i = S_W + R_c + K_0^{i-1} \\ K_1^i = K_0^i + K_1^{i-1} \\ K_2^i = K_1^i + K_2^{i-1} \\ K_3^i = K_3^i + K_3^{i-1} \end{cases} \tag{4-2}$$

$$Ron(i) = \begin{bmatrix} \{02\}^{i-1} \\ \{00\} \\ \{00\} \\ \{00\} \end{bmatrix} \tag{4-3}$$

现有文献中，AES 加密密钥扩展单元的硬件实现一般按照式 4-2 采用链式结构，如图 4-1 所示。

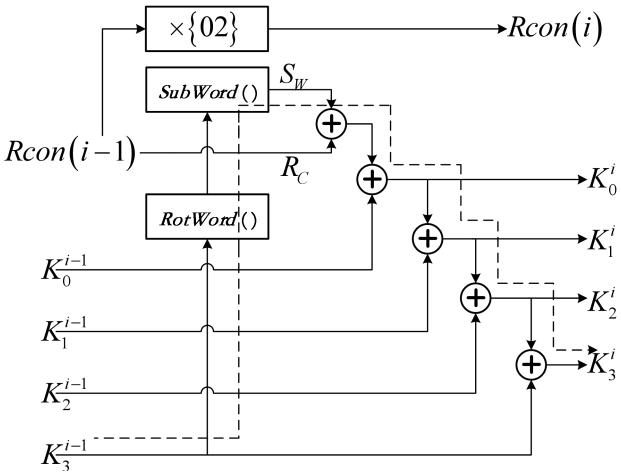


图 4-1　基于链式结构的 AES 加密密钥扩展单元图

在链式结构中，通过将后一个输出信号复用了前一个输出信号的运算资源的方法，从而达到了硬件实用面积成本降低的目的。但链式电路结构在减少电路面积的同时却增加了电路关键路径，从而使电路延时得到增加。链式结构的关键路径如图 4-1 中的虚线所示。关键路径延迟由逻辑门延迟和连线延迟组成，本文仅考虑逻辑门延迟。

在关键路径上，*RotWord* 操作只需切换总线顺序即可实现，因此 *RotWord* 单元中不需要任何逻辑资源，其 *RotWord* 单元的关键路径长度为 0。*SubWord* 单元的关键路径长度相当于 *SubBytes* 的关键路径长度，本文记为 T_{SB}。加法器的关键路径上只有一个异或门，因此加法器的关键路径长度相当于异或门的关键路径长度，本文记为 T_{XOR}。*SubWord* 之后有 5 个加法器，可得基于链式结构的 AES 密钥扩展单元的关键路径为 $T_{KE}=T_{SB}+5T_{XOR}$，其中 T_{KE} 为 AES 密钥扩展单元的关键路径。由于 *SubBytes* 的关键路径长度为 $T_{SB}=18T_{XOR}+3T_{AND}\approx 20T_{XOR}$，得 $T_{KE}=25T_{XOR}$。

4.1.2 AES 解密密钥扩展单元

解密中的密钥扩展操作又称为逆密钥扩展运算，它是加密密钥扩展操作的逆操作。解密密钥扩展可以用式 4-4 表示。

$$\begin{cases} K_0^i = S'_W + R'_c + K_0^{i-1} \\ K_1^i = K_0^{i-1} + K_1^{i-1} \\ K_2^i = K_1^{i-1} + K_2^{i-1} \\ K_3^i = K_2^{i-1} + K_3^{i-1} \end{cases} \tag{4-4}$$

其中 $S'_W = SubWord\left(RotWord\left(K_2^{i-1} + K_3^{i-1}\right)\right)$，$R'_c = Rcon^{-1}(i)$，$Rcon^{-1}(i)$为常量，可用式 4-5 表示。{36}和{8*D*}都是 $GF(2^8)$域中的元素。

$$Ron^{-1}(i) = \begin{bmatrix} \{36\} \times \{8D\}^i \\ \{00\} \\ \{00\} \\ \{00\} \end{bmatrix} \tag{4-5}$$

根据公式 4-4 可以得到解密密钥扩展运算单元结构如图 4-2 所示，整个解密密钥扩展运算单元仅需要 5 个加法器。解密密钥扩展运算单元的关键路径如图 4-4 中的虚线所示。在关键路径上，*SubWord* 之前和之后各有一个加法器，则解密密钥扩展单元的关键路径为 $T_{KE}=T_{SB}+2T_{XOR}=22T_{XOR}$。因此图 4-2 所示的解密密钥扩展单元是最优的电路结构，即单元的面积和关键路径均达到最小，解密密钥扩展单元没有进一步优化的空间。

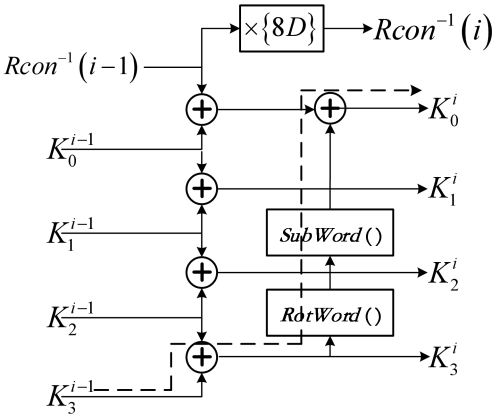


图 4-2　AES 解密密钥扩展单元图

4.1.3 AES 加解密一体化密钥扩展单元

实际应用中，加密和解密操作常常需要集成到同一个硬件中。在这种情况下，可以重复使用相同的操作单元，以减少硬件实现中的面积成本。其中 AES 密码

算法中密钥扩展操作分为加密部分和解密部分，可以将其中相同的操作单元重复使用，从而构造 AES 加解密一体化密钥扩展单元。

在传统的 AES 加解密一体化密钥扩展单元中，通常采用链式结构来降低面积成本。基于链式结构的 AES 加解密一体化密钥扩展单元如图 4-3 所示。除了 *SubWord* 单元外，AES 加解密一体化密钥扩展单元总共需要 5 个加法器和 5 个 2 对 1 复用器。一体化单元可以在两种状态下操作：加密密钥扩展状态和解密密钥扩展状态。一体化单元在两种工作状态下的关键路径如图 4-3 中的虚线所示。在加密密钥扩展状态下，一体化单元的关键路径为 $T_{KE}=T_{SB}+5T_{XOR}+4T_{MUX}$，其中 T_{MUX} 为复用器的关键路径。在解密密钥扩展状态下，一体化单元的关键路径为 $T_{KE}=T_{SB}+3T_{XOR}+2T_{MUX}$。因此，整个一体化单元的关键路径为 $T_{KE}=T_{SB}+5T_{XOR}+4T_{MUX}$。

根据图 4-3，加解密一体化密钥扩展运算操作可由式 4-6 表示。

$$\begin{cases} K_0^i = S_W'' + R_c'' + K_0^{i-1} \\ K_1^i = K_0^i / K_0^{i-1} + K_1^{i-1} \\ K_2^i = K_1^i / K_1^{i-1} + K_2^{i-1} \\ K_3^i = K_2^i / K_2^{i-1} + K_3^{i-1} \end{cases} \tag{4-6}$$

其中 $S_W'' = SubWord\left(RotWord\left(K_3^{i-1} / \left(K_2^{i-1} + K_3^{i-1}\right)\right)\right)$、$R_c'' = Rcon(i) / Rcon^{-1}(i)$ 和 ‘/’ 表示 2 对 1 多路复用器操作。

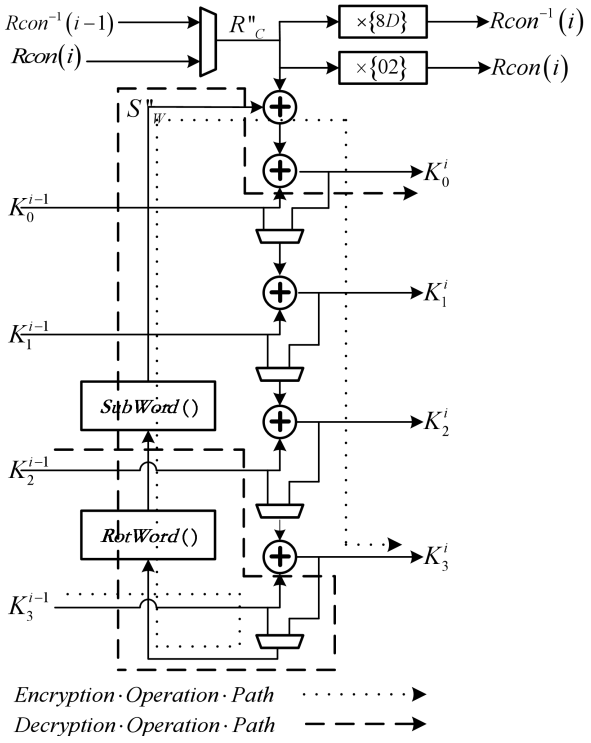


图 4-3　基于链式结构的 AES 加解密一体化密钥扩展单元图

4.2 低延迟结构构建方法

基于链式结构的密钥扩展单元虽然面积较小，但关键路径较长，会导致较大的电路延迟。针对这一问题，提出一种构造低延迟密钥扩展单元的方法。具体施工方法如下：

步骤 1：推导 AES 密钥扩展运算的表达式。

步骤 2：通过 DACSE 算法从表达式中提取 CS。

步骤 3：构建基于 DDBT 结构的密钥扩展单元。

根据上述构建方法，进一步构建低延迟 AES 加密密钥扩展单元和低延迟 AES 加解密一体化密钥扩展单元。

4.2.1 低延迟 AES 加密密钥扩展单元

由式 4-2 可得加密展开运算表达式的完整形式如式 4-7 所示。因式 4-7 线性运算表达式，则可以采用 DDBT 结构构建实现最短关键路径。而在构造 DDBT 结构之前，需要获得所有输入信号的延迟。

$$\begin{cases} K_0^i = S_W + R_c + K_0^{i-1} \\ K_1^i = S_W + R_c + K_0^{i-1} + K_1^{i-1} \\ K_2^i = S_W + R_c + K_0^{i-1} + K_1^{i-1} + K_2^{i-1} \\ K_3^i = S_W + R_c + K_0^{i-1} + K_1^{i-1} + K_2^{i-1} + K_3^{i-1} \end{cases} \tag{4-7}$$

如式 4-7 所示，输入信号包括$\left[S_W, R_c, K_0^{i-1}, K_1^{i-1}, K_2^{i-1}, K_3^{i-1}\right]$，因 AES 加密密钥扩展单元之前没有任何操作，即除了输入信号 S_W 之外，输入信号的延迟为 0。又因在输入信号 S_W 之前有一个 *Subword* 操作，输入信号 S_W 的延迟为：$18T_{XOR}+3T_{AND}\approx 20T_{XOR}$。DDBT 结构的构建方法如下所示。

```
for j=1 to N_out
    All input signals form a set S
    for i=1 to N_in-1
        Choose the two signals s1 and s2 with the least delay from S
        Construct an addition circuit s_new=s1+s2
        Remove s1 and s2 from S
        Compute the delay of s_new with t_new=max(t1, t2)+1
        Add s_new to the set S
    end
end
```

其中 N_{in} 和 N_{out} 分别表示输入信号和输出信号的数量，t_{new}、t_1 和 t_2 分别表示 s_{new}、s_1 和 s_2 的延迟。该构造方法在构造过程的每次迭代中总是选择延迟最小的

两个信号来构造电路。根据构建方法，可以构建基于 DDBT 结构的 AES 加密密钥扩展单元，如图 4-4 所示。

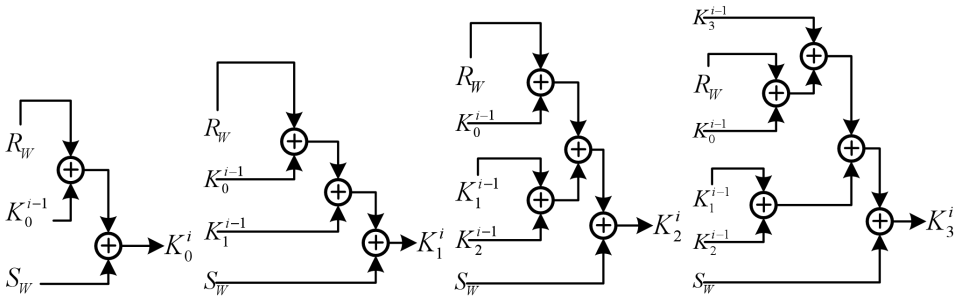


图 4-4　基于 DDBT 结构的 AES 加密密钥扩展单元图

DDBT 结构的关键路径长度计算表达式如式 4-8。其中 T_j 是第 j 个输出信号的延迟，$t_{k,j}$ 是第 j 个输出信号的第 k 个输入信号的延迟，N_j 是第 j 个输出信号的输入信号的数量。而 T_j 和 $t_{k,j}$ 是根据路径上逻辑门的数量计算的。根据式 4-8，可以得到基于 DDBT 结构的 AES 加密密钥扩展单元的关键路径长度为 T_{KE}=21T_{XOR}=T_{SB}+1T_{XOR}。在基于 DDBT 结构的 AES 加密密钥扩展运算单元中，输入信号 S_W 之后只有一次加法运算，与链式结构相比下，DDBT 结构的关键路径减少了 4T_{XOR}，大大缩短了关键路径。

$$T_{\mathrm{DDBT}} = \max_{j=0,\cdots,n-1}\left(T_j\right) = \max_{j=0,\cdots,n-1}\left(\left\lceil \log_2 \sum_{N_j} 2^{t_{k,j}} \right\rceil\right) \tag{4-8}$$

图 4-4 中的 DDBT 结构的密钥扩展单元虽然减少了关键路径长度，但也大大增加了电路面积，其整个密钥扩展电路需要 14 个 32 比特加法器，而链式结构只需要 5 个 32 比特加法器，这是因为输出信号的操作之间没有共享单元。本文采用公共项消除算法 DACSE 来提取表达式中的公共项表达式 CS，在运算表达式中共享 CS 即可以减少加法器的数量。DACSE 算法的构建如下所示。

Input the operation expressions $\boldsymbol{E}$ and delay constraint T_{con}
Create two empty sets $\boldsymbol{S}_{\mathrm{CS}}$ and $\boldsymbol{S}_{\mathrm{IV}}$
Count the occurrence frequency of each CS in $\boldsymbol{E}$
Find out the max occurrence frequency $O_{\max}$
while $O_{\max}$>1
　　Create new operation expressions $\boldsymbol{E}_{\mathbf{CS}}$=$\boldsymbol{E}$
　　Select a CS (s_1+s_2) with occurrence frequency $O_{\max}$ randomly
　　Replace the CS (s_1+s_2) in $\boldsymbol{E}_{\mathbf{CS}}$ with a new signal s_{com}
　　Compute the delay of s_{com} with t_{com}=max(t_1, t_2)+1
　　Compute the whole operation unit delay T_{KE} with (5)
　　if T_{KE}=T_{con}
　　　　Add the eliminated CS s_{com}=s_1+s_2 to set $\boldsymbol{S}_{\mathrm{CS}}$
　　　　Replace $\boldsymbol{E}$ with $\boldsymbol{E}_{\mathbf{CS}}$, that is $\boldsymbol{E}$=$\boldsymbol{E}_{\mathbf{CS}}$

```
        else
            Add the CS s1+s2 to set S_IV
        end if
        Count the occurrence frequency of each CS in E other than the CS in set S_IV
        Find out the max occurrence frequency O_max
end while
```

其中 t_{com}、t_1 和 t_2 分别表示 s_{com}、s_1 和 s_2 的延迟。DACSE 算法首先需要输入延迟约束 T_{con},而基于 DDBT 结构的 AES 加密密钥扩展单元的关键路径长度为 T_{KE}=21T_{XOR}。因此，当通过 DACSE 优化 AES 加密密钥扩展操作的表达式时，延迟约束设置为 T_{con}=21。有了这个约束，最短关键路径长度可以在 CS 的提取过程中保持不变。

DACSE 算法在 CS 提取的每次迭代中计算 T_{KE}。如果 CS 的提取导致 T_{KE} 超过时延约束 T_{con}，则放弃 CS 的提取。如果提取的 CS 保持 T_{KE} 不变，则提取的 CS 有效。在经过 DACSE 算法优化后，AES 加密密钥扩展操作可表示为：

$$\begin{cases} K_0^i = S_W + C_0 \\ K_1^i = S_W + C_1 \\ K_2^i = S_W + C_2 \\ K_3^i = S_W + C_3 \end{cases} \tag{4-9}$$

其中[C_0, C_1, C_2, C_3]为提取的公共项 CS，它的展开式为式 4-10。提取 CS 后，同样采用 DDBT 结构构造 AES 加密密钥扩展单元如图 4-5 所示。

$$\begin{cases} C_0 = R_c + K_0^{i-1} \\ C_1 = C_0 + K_1^{i-1} \\ C_2 = C_1 + K_2^{i-1} \\ C_3 = C_2 + K_3^{i-1} \end{cases} \tag{4-10}$$

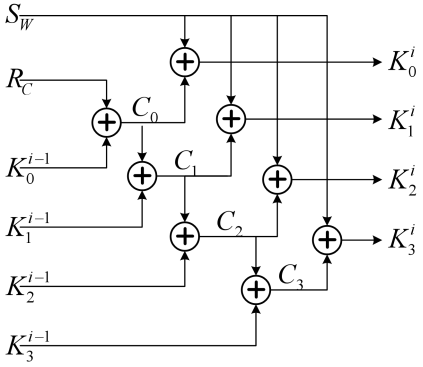


图 4-5　提取 CS 后的 AES 加密密钥扩展单元图

AES 加密密钥扩展单元的关键路径长度仍然为 $T_{KE}=T_{SB}+1T_{XOR}$，优化之后的加密密钥扩展运算单元仅使用 8 个加法器，相比未优化电路，减少了 6 个加法器。较基于链式结构的密钥扩展运算单元，在关键路径上减少了 4 个加法器，代价是加法器总数增加了 3 个，即面积上只增加了 3 个加法器。

4.2.2 低延迟 AES 加解密一体化密钥扩展单元

根据 4-7 中的加密密钥扩展表达式和 4-4 中的解密密钥扩展表达式，可以得到一体化的加解密密钥扩展运算表达式如下。

$$\begin{cases} {}^{+}K_0^i = S_W'' + R_c'' + K_0^{i-1} \\ {}^{+}K_1^i = S_W'' + R_c'' + K_0^{i-1} + K_1^{i-1} \\ {}^{+}K_2^i = S_W'' + R_c'' + K_0^{i-1} + K_1^{i-1} + K_2^{i-1} \\ {}^{+}K_3^i = S_W'' + R_c'' + K_0^{i-1} + K_1^{i-1} + K_2^{i-1} + K_3^{i-1} \\ {}^{-}K_0^i = S_W'' + R_c'' + K_0^{i-1} \\ {}^{-}K_1^i = K_0^{i-1} + K_1^{i-1} \\ {}^{-}K_2^i = K_1^{i-1} + K_2^{i-1} \\ {}^{-}K_3^i = K_2^{i-1} + K_3^{i-1} \end{cases} \tag{4-11}$$

其中 $S_W'' = SubWord\left(RotWord\left(K_3^{i-1} / \left(K_2^{i-1} + K_3^{i-1}\right)\right)\right)$，$R_c'' = Rcon(i) / Rcon^{-1}(i)$，‘/’表示 2 对 1 复用运算，$\left[{}^{+}K_0^i,\ {}^{+}K_1^i,\ {}^{+}K_2^i,\ {}^{+}K_3^i\right]$是加密密钥扩展运算的输出向量，$\left[{}^{-}K_0^i,\ {}^{-}K_1^i, {}^{-}K_2^i,\ {}^{-}K_3^i\right]$是解密密钥扩展运算的输出向量。从式 4-11 知，输入信号包括$\left[S_W'',\ R_c'',\ K_0^{i-1},\ K_1^{i-1},\ K_2^{i-1},\ K_3^{i-1}\right]$，除输入信号 S_W'' 外，其他输入信号的延迟均为 0。输入信号 S_W'' 之前有加法器、2 选 1 复用器、*Subword* 操作，则输入信号的延迟为 $T_{XOR}+T_{MUX}+T_{SB}$，其中 T_{MUX} 是多路复用器的关键路径。现假设 $T_{MUX}\approx T_{XOR}$，那么输入信号的延迟约为 $22T_{XOR}$。即可得到 DDBT 结构的关键路径长度为 $T_{KE}=23T_{XOR}$。

$$\begin{cases} {}^{+}K_0^i = C_3 \\ {}^{+}K_1^i = S_W'' + C_0 + K_1^{i-1} \\ {}^{+}K_2^i = S_W'' + C_2 \\ {}^{+}K_3^i = S_W'' + C_2 + K_3^{i-1} \\ {}^{-}K_0^i = C_3 \\ {}^{-}K_1^i = K_0^{i-1} + K_1^{i-1} \\ {}^{-}K_2^i = C_1 \\ {}^{-}K_3^i = K_2^{i-1} + K_3^{i-1} \end{cases} \tag{4-12}$$

根据式 4-11 可提取公共项 CS，通过 DACSE 算法进行优化，且 DACSE 算法的延迟约束也为 T_{con}=23。优化后得加解密一体化密钥扩展的表达式如式 4-12。

其中[C_0, C_1, C_2, C_3]为提取的公共项 CS，它的展开式为式 4-13。

$$\begin{cases} C_0 = R_c'' + K_0^{i-1} \\ C_1 = K_1^{i-1} + K_2^{i-1} \\ C_2 = C_0 + C_1 \\ C_3 = S_W'' + C_0 \end{cases} \tag{4-13}$$

由式 4-12 可构建低时延 AES 加解密一体化密钥扩展单元，如图 4-6 所示。该单元中总共需要 5 个 2 对 1 复用器和 11 个加法器，其中有 8 个加法器通过 DACSE 算法减少。在加密密钥扩展状态下，一体化单元的关键路径为 $T_{KE}=T_{SB}+T_{XOR}+2T_{MUX}$。在解密密钥扩展状态下，一体化单元的关键路径为 $T_{KE}=T_{SB}+2T_{XOR}+2T_{MUX}$。因此，整个一体化单元的关键路径为 $T_{KE}=T_{SB}+2T_{XOR}+2T_{MUX}$。与链式结构相比，DDBT 结构虽然增加了 6 个加法器，但时延却大大降低。

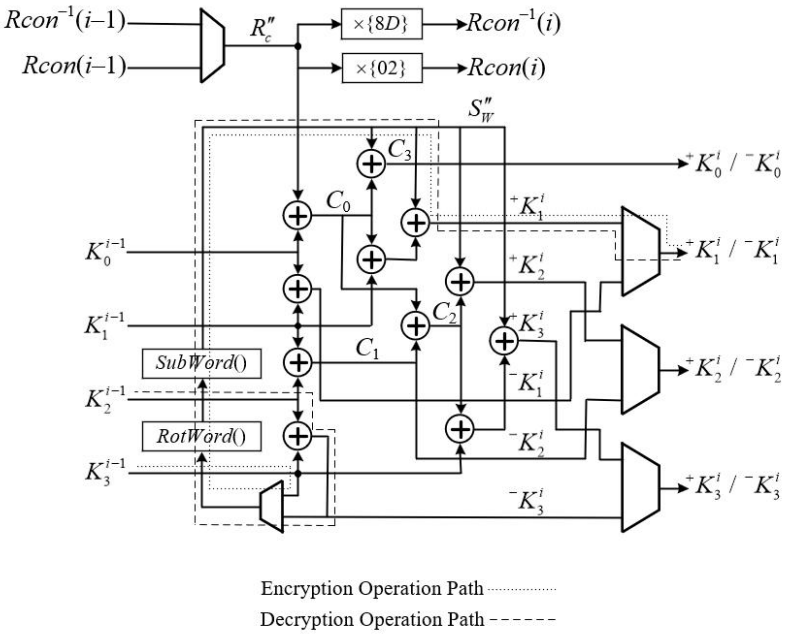


图 4-6　基于 DDBT 结构的 AES 加解密一体化密钥扩展单元图

4.3 验证与比较

4.3.1 集成电路综合验证

通过 4.2 节的优化，对于 DDBT 结构与链式结构的硬件复杂度比较如表 4-1 所示。在 AES 加密密钥扩展单元和 AES 加解密一体化密钥扩展单元中，DDBT 结构的关键路径长度较短。

表 4-1　DDBT 结构与链式结构硬件复杂度对比

类	结构	面积	长度
加密	链式	5 adders	$T_{SB}+5T_{XOR}$
	DDBT	8 adders	$T_{SB}+T_{XOR}$
一体化	链式	5 adders+5 MUX	$T_{SB}+5T_{XOR}+4T_{MUX}$
	DDBT	11 adders+5 MUX	$T_{SB}+2T_{XOR}+2T_{MUX}$.

通过对低延迟设计采用 SMIC 0.18μm 技术的 Synopsys™集成电路 IC 综合工具（Design Compiler, DC）进一步验证。综合结果如表 4-2 所示。在 AES 加密密钥扩展单元中，DDBT 结构的延迟以面积增加 10.08%的代价降低了 15.23%。而在 AES 加解密一体化密钥扩展单元中，DDBT 结构的时延仅以面积增加 12.25%的代价降低了 67.52%。从综合结果可以表明，优化后的设计以较少的面积增加为代价大大降低了电路延迟。

表 4-2　密钥扩展单元的 IC 综合结果

类	结构	面积（*gates*）	延迟（*ns*）	频率（*MHz*）
加密	链式	2550	6.37	156.99
	DDBT	2807	5.40	185.19
一体化	链式	3264	23.12	43.25
	DDBT	3664	7.51	133.16

4.3.2 与相关文献的对比

鉴于相关文献中所有 AES 密钥扩展单元的设计都是通过链式结构来构造的，其目的是为了节省硬件资源。在文献[53-60]中，128 位宽度结构用于高数据吞吐量应用。文献[61-67]中使用 8 位宽度结构用于资源有限的应用。为了在速度和面积之间做出选择，[68-73]中构造了 32 位宽度的结构。此外文献[74]中提出了一种用于 AES-256 密钥扩展操作的 256 位宽度结构。但这些设计针对的是不同的应用而使用不同的数据宽度结构。在 8 位宽度结构和 32 位宽度结构中，AES 密钥扩展单元需要迭代多次才能完成一轮子密钥扩展操作。而在 128 位宽度结构和 256 位宽度结构中，子密钥数据被并行处理以加速数据处理。

由于本章中采用 128 位宽度来构造 DDBT 结构，因此之前的工作采用 128 位宽度结构[53-60]和 256 位宽度结构[74]在表 4-3 中列出并作比较。

表 4-3 中先前工作中的频率是最大频率，除了[56]和[59]中的频率是工作频率，与之前的文献相比，本章提出的关键扩展单元具有最大的吞吐量。其吞吐量可通过以下公式计算：

$$Throughput = \frac{SubkeyLength \times Rounds \times Frequency}{Cycles} \tag{4-14}$$

表 4-3　不同实现方式下综合结果对比

类	作品	平台	面积（*gates*）	频率（*MHz*）	周期	吞吐量
加密	[45]	Xilinx Virtex	—	93.50	30	3.99
				168.40	70	3.08
	[54]	Xilinx Virtex-4	—	208.49	30	8.89
				247.19	50	6.33
	Ours	0.18μm SMIC	2807	185.19	10	23.70
一体化	[53]	Xilinx Virtex II	—	281.3	50	7.20
				305.1	90	4.34
	[55]	0.11μm standard	3072	131.24	50	3.36
	[56]	0.25μm TSMC	—	100	21	6.10
	[57]	0.6μm standard	3071	64	10	8.19
	[58]	Xilinx Virtex-E	—	26.40	10	3.38
		Xilinx Spartan-3		28.01		3.58
	[59]	0.6μm standard	3201	50	10	6.40
	[60]	Xilinx Virtex II	—	177.3	51	4.45
	[74]	90nm standard	—	131.8	14	16.87
	Ours	0.18μm SMIC	3664	133.16	10	17.04

因 AES-128 需输入 128 位初始密钥，再通过密钥扩展操作生成 10 轮 128 位子密钥进行轮变换，而 AES-256 则需输入 256 位初始密钥，生成 14 轮 128 位子密钥。本章提出的密钥扩展单元需要 10 个周期才能完成 10 轮子密钥扩展操作。由于链式结构的关键路径长度较大，在关键扩展单元中插入流水线级以加快工作频率[45,53-56,60]。但更多的流水线级数，需要更多的时钟周期来实现关键扩展操作，因此，吞吐量也随之下降。由于采用了更先进的 IC 工艺和 256 位宽度结构，[74]中提出的密钥扩展单元也具有更高的吞吐量。

4.4 本章小结

本章提出了一种低延迟电路结构构建方法来构建用于高速 AES 实现的低延迟 AES 密钥扩展单元。利用该构造方法构造了低延迟、高吞吐量的 AES 加密密钥扩展单元和 AES 加解密一体化密钥扩展单元，其中低延迟 AES 加密密钥扩展单元已应用在第二章 AES 加密运算单元并付诸于实际。现采用 Synopsys IC 设计工具对电路进行评估，并与前人研究相比较，分析各个方案在延时、面积和吞吐量方面的性能，实验结果表明，设计实现的低延迟电路表现出高速、高吞吐量特性。在许多应用中，加解密和密钥扩展并不是同时进行的，*SubBytes* 单元通常在轮变换和密钥扩展之间共享，以减少电路实现面积。基于本文提出的构造方法，还可以构造用于轮变换和密钥扩展的低延迟一体化结构。

第 5 章 能量分析及安全性检测平台设计

密码芯片设计的过程中，往往需要对其安全性能进行检测，以缩短设计周期，尽快满足应用要求。本章基于能量分析的原理，从攻击者的角度搭建硬件能量采集分析平台，并通过对不同总线结构 AES 密码电路的安全分析，验证了平台的可靠性，完成了相应的安全性评估，最后对前文所设计并应用的星载高速 AES 密码电路进行安全性分析。

5.1 能量分析

5.1.1 能量分析原理

通过对密码设备的能量消耗这一物理特性进行分析，从而恢复设备的加密信息，这种分析手段称为能量分析[25]。它基于两种能量消耗的相关性：数据与操作相关性，即密码设备的瞬时能量消耗与设备所处理的数据和设备所执行的操作相关。将一条能量迹放大后再观测，每一个时钟内的能量消耗就会变得很清楚。

如图 5-1 所示，在这条能量迹中每一个可见的尖峰都对应着电路在一个时钟周期内的能量消耗。对于不同的操作和不同的数据，各尖峰的形状也不相同，即可以利用这一特性对一条能量迹进分析攻击，该手段被称为简单能量分析 SPA。

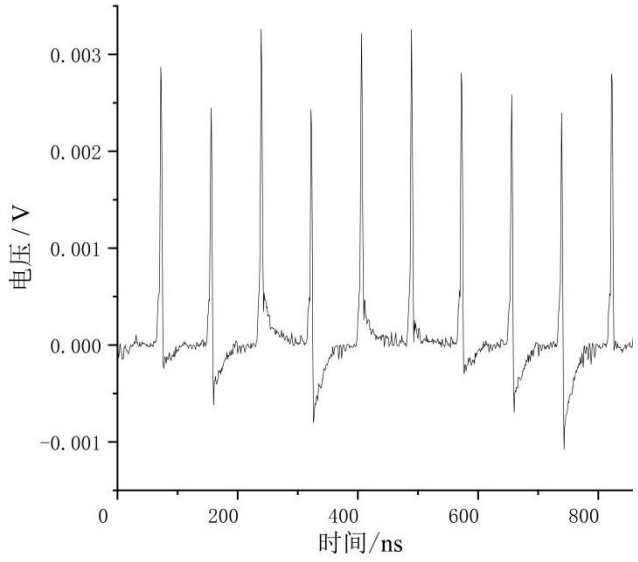


图 5-1　采样获得的能量迹放大视图

能量分析的第一步就是要建立合适的能量模型，明确能量值与数据操作数的映射关系，以便推测出密钥的值。由于数字电路能量消耗主要与信号发生 0→1 和 1→0 的转换次数有关，可以假设 0→1 和 1→0 转换消耗同样的能量，且 0→0

和 1→1 转换对能量消耗具有相同的影响。在两个等长的二进制数据串 v_0 和 v_1 之间，通过密码算法的运算使得 $v_0 \rightarrow v_1$ 转换，其对应位置不同字符的个数与能量消耗有着直接的联系，常用汉明距离（Hamming Distance, HD）模型 HD (v_0,v_1) 表示。而在实际情况下往往无法确定数据总线前续或后继处理的数据，即可假设能量消耗与被处理的数据中被置位的比特个数为正比，从而忽略在该数据之前和之后处理的数值，使用汉明重量（Hamming Weight, HW）模型。

值 v_0 和 v_1 的汉明距离等于 $v_0 \oplus v_1$ 的汉明重量，汉明重量等于逻辑值为“1”的比特个数，所以 HW ($v_0 \oplus v_1$)表示 v_0 和 v_1 中相异比特的个数，如式 5-1 所示。

$$HD(v_0, v_1) = HW(v_0 \oplus v_1) \tag{5-1}$$

在分析能量与模型的关系时，只分析仿真能量消耗与实际能量消耗之间的比例关系，即在发生 $v_0 \rightarrow v_1$ 转换之前，无论将 v_0 的所有比特置 0 或置 1，汉明重量模型与汉明距离模型事实上是等价的。能量消耗可近似表达为式 5-2 所示。

$$P \approx aHW(v_1) + b \tag{5-2}$$

其中，P 为能量消耗，a 为能量消耗与模型之间的比例系数，b 为其它能耗影响。现利用能量迹中的数据相关性，根据 HW 模型即可确定密码设备处理不同数据时能量消耗的概率分布，如以 8 比特数区分 9 个不同的汉明重量，刻画同种操作下不同数据所产生的能量消耗。

5.1.2 差分能量分析

在以攻击者的角度对目标设备进行能量分析时，对于复杂的算法和结构等情况下，简单能量分析（Simple Power Analysis, SPA）很难将有效能量与实际噪音区分开，所以需要采集大量的能量迹并针对固定时刻分析能量消耗，从而利用密码设备能量消耗的数据依赖性获取密钥，这种策略被称为（Differential Power Analysis, DPA）差分能量分析。

DPA 一般性策略包括 5 个步骤，具体如下。

第 1 步：选择所执行算法的某个中间值。该中间值应为函数 f，其中包含可变数据 d 和密钥 k 的一部分，以该中间值可恢复完整密钥 k。

第 2 步：测量密码设备在加密或解密 D 个不同数据分组时的能量消耗。过程中每个操作都涉及特定数值 d，该值 d 用于第 1 步中对中间值的计算过程。将这些数值记作向量 $d=(d_1,..., d_D)$，其中 d_i 表示第 i 次加密或解密操作对应的数据值。每次运行记录单次能量迹，并将其与相应的数据分组 d_i 关联记作能量迹集合

$t'_i=(t_{i,1},..., t_{i,T})$，$T$ 表示能量迹的长度，且可将这些能量迹记为一个 $D\times T$ 的矩阵 T。测量获得的能量迹还需要校准对齐，确保矩阵 T 中每一列 t_j 代表的是相同操作所消耗的能量。

第 3 步：计算假设中间值。在未知实际密钥的情况下，针对所选的中间值位置猜测全部 k 值可能性，并计算对应的假设中间值记为向量 $k=(k_1,..., k_K)$，K 表示 k 所有可能值的数量。以数据向量 d 和假设密钥向量 k 可以计算出假设中间值 $f(d,k)$。式 5-3 可以得到 $D\times T$ 后的矩阵 V，图 5-2 说明了该计算步骤。

$$v_{i,j} \approx f(d_i,k_j), i=1,...,D; j=1,...,K. \tag{5-3}$$

V 的第 j 列包含了基于密钥假设 k_j 所计算出的中间值，k 包含了设备密钥所有可能的值。因此，设备实际使用的值只是 k 中的一个元素，即找出设备在 D 次加密或解密运行期间处理的是 V 的哪一列，从而确定实际密钥。

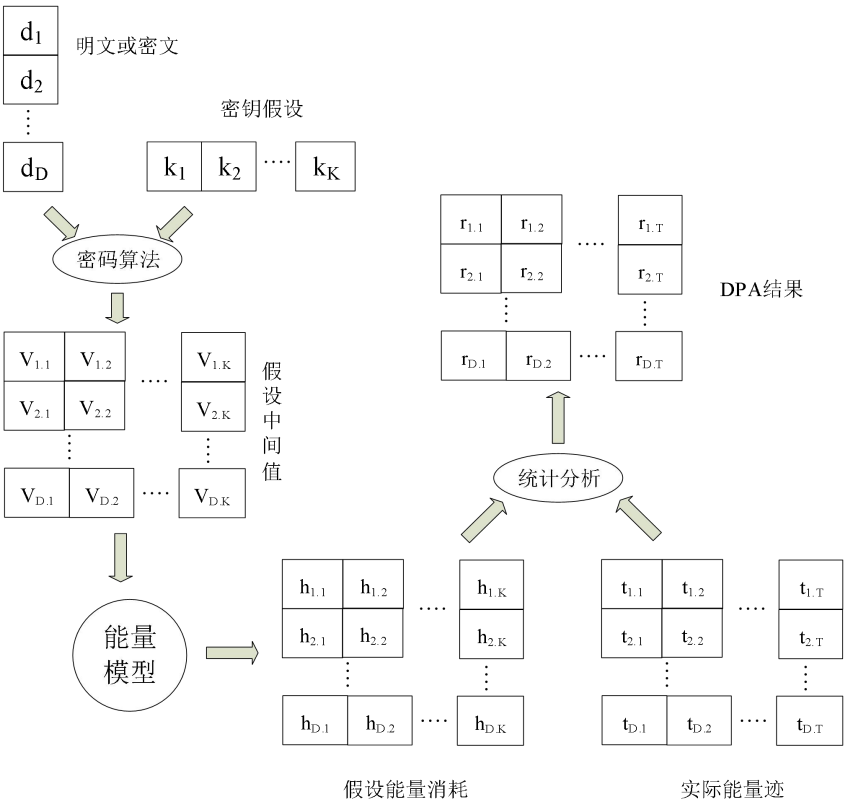


图 5-2　差分能量分析的原理示意图

第 4 步：将假设中间值 V 映射为假设能量消耗值 H。以能量模型对假设中间值 $v_{i,j}$ 模拟设备能量消耗，获得假设能量消耗值 $h_{i,j}$。此时就可以得到两组能量曲线，而能量模型的准确性主要取决于对设备操作细节的掌握，模型仿真与设备实际能量消耗特性一致性越高，分析就越有效。

第 5 步：分析对比假设值 H 与实际采集值 T。对矩阵 H 的每一列 h_i 和矩阵 T 的每一列 t_j 进行比较，使每个猜测密钥所对应的假设能量消耗值可与实际能量迹逐位比较。比较后得到大小为 $K\times T$ 矩阵 R，每一个元素 $r_{i,j}$ 包含了列 h_i 和 t_j 的相似程度，$r_{i,j}$ 值越大，列 h_i 与 t_j 的相似度越高，所记录的能量值就越依赖于这些中间值，从而确定矩阵 R 中的最大值。根据最大值的位置就可以确定正确密钥索引以及关联时刻。

5.1.3 相关系数代替法

在对假设能量消耗值和实际采集能量迹进行比较的过程中，常以比较 H 和 T 的列相关系数来确定，但这一过程的准确性与所选的能量模型有着直接的联系，通常将基于相关系数的能量分析称为（Correlation Power Analysis, CPA）相关性能量分析。本文采用的是均值差法来分析能量迹，在将矩阵 V 映射为 H 的过程中，使用二元模型来做为能量模型[25]。这意味着需要以满足 $h_{i,j}\in\{0,1\}(\forall i,j)$的方式来选择能量模型，即当汉明重量 HW($v_{i,j}$)≧4 时，令 $h_{i,j}$=1；当 HW($v_{i,j}$)<4 时，令 $h_{i,j}$=0。这样大大的简化了映射工作，但同样也增加了实际能量迹采集数量。

由二元模型生成的 0/1 假设能量矩阵 H，其每一列中的 0/1 序列是输入数据 d 和密钥假设 k_i 的函数。根据 h_i 将矩阵 T 的行分成两组，即两个能量迹子集。第一个子集中包含了 T 中那些索引值等于向量 h_i 中 0 的索引的行，第二个子集则包含了 T 中所有剩余的行。再计算各行均值，可设向量 m_{0i} 为第一个子集中行的均值，m_{1i} 为第二个子集中行的均值，如果在某个时间点上 m_{0i} 和 m_{1i} 之间具有了最大的差值，则将差值最大的 r_i 所对应的假设密钥 k_i 作为分析结果。

$$m_{1i,j}=\frac{1}{n_{1i}}\sum_{l=1}^{n}h_{l,i}\cdot t_{l,j} \tag{5-4}$$

$$m_{0i,j}=\frac{1}{n_{0i}}\sum_{l=1}^{n}(1-h_{l,i})\cdot t_{l,j} \tag{5-5}$$

$$n_{1i}=\sum_{l=1}^{n}h_{l,i} \tag{5-6}$$

$$n_{0i}=\sum_{l=1}^{n}(1-h_{l,i}) \tag{5-7}$$

$$R=M_1-M_0 \tag{5-8}$$

上式 5-4~5-8 给出均值差方法计算 DPA 矩阵 R 的公式，其中 n 表示 H 中行的数量，即攻击中所需要的能量迹数量；M_0 和 M_1 是其对应子集的均值之集合。

5.2 安全检测平台设计与实现

自能量分析越来越深入的研究，如何确保数据安全性也开始被提上日程。在密码芯片设计应用的过程中，对这些潜在的安全隐患必须提前做好防范。为此，本文基于 FPGA 设计一款安全检测平台，以攻击者的角度对密码芯片可能存在的泄露点进行能量数据的采集，并对比采集数据分析安全性。

5.2.1 安全检测平台结构设计

密码芯片的能量分析分为以设计者角度的模拟能量仿真验证和以攻击者角度的实际能量分析验证，本文据此研究并设计了安全检测平台，该平台分为硬件能量采集模块和能量数据分析模块。硬件能量采集模块主要用于采集密码芯片运行过程中瞬时能量数据，并以时间戳和瞬时电压值保存至 Txt 格式数据文件。能量数据分析模块主要对采集到的数据先处理再分析，来评估其安全性能。

（1）硬件能量采集模块

密码芯片的能量采集有两种方式：一种是软件采集，即基于专门的 EDA 工具模拟产生瞬时能量数据；另一种是硬件采集，即构建硬件密码电路来获取实时能量数据。由于软件采集依赖于 EDA 工具且相对理想，与实际条件中电磁、噪音等干扰因素影响下的密码电路差距较大，因此本文采用硬件采集方式设计安全检测平台。该平台主要由 PC 上位机软件、SAKURA-G 能量采集开发板、数字示波器构成，平台结构如图 5-3 所示。

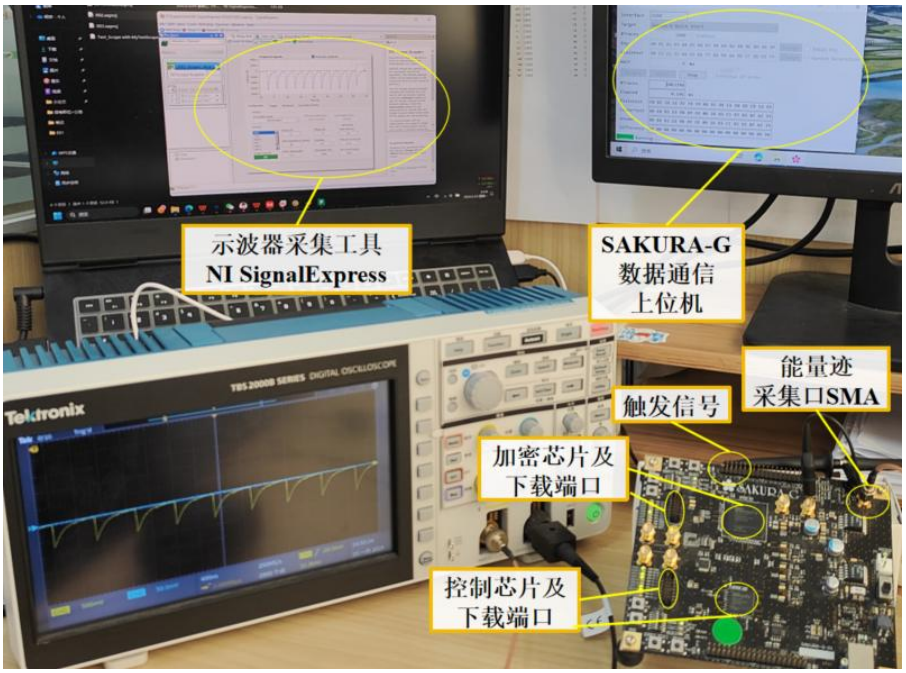


图 5-3　硬件能量采集平台结构图

能量采集阶段的具体步骤：首先将 AES 密码电路和数据控制电路分别下载到 SAKURA-G 开发板中加密芯片 Xc6slx75 与控制芯片 Xc6slx9 两种型号的 FPGA，如图 5-4 所示；接着通过控制芯片的串口通信与 PC 端的上位机连接，以验证密码电路正常运行；然后使用 Tektronix 示波器完成波形采集；最后由示波器的可交互式 IVI 驱动来与硬件交互式采集工具 NI SignalExpress 建立联系，从而将采集电压信号传输至 PC 端生成 Txt 格式的电压曲线文件。

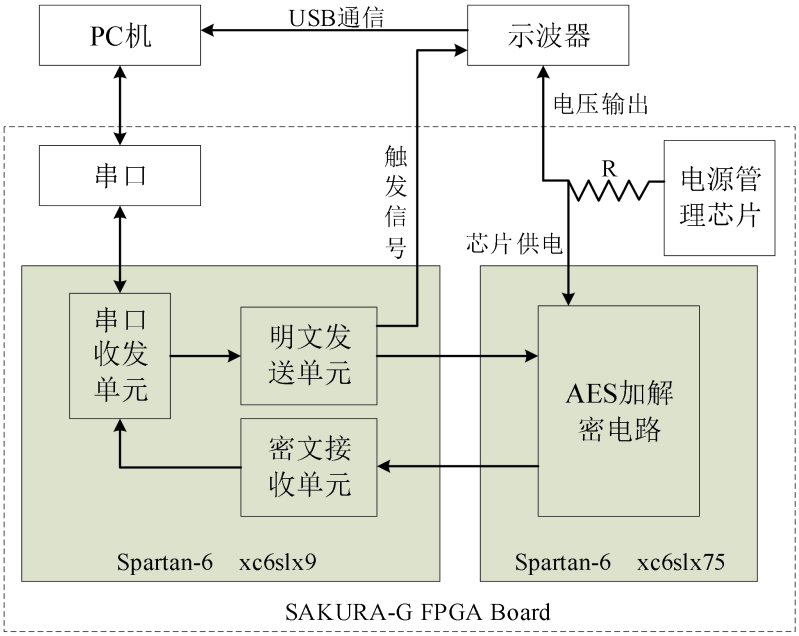


图 5-4　SAKURA-G 板载芯片功能示意图

由图 5-4 可知，Xc6slx9 作为数据控制电路的运载芯片，用于接收 PC 端串口发送过来的明文数据，并对 AES 密码运算电路施加激励将其输出结果由串口传输返回至 PC 端。此外，当 Xc6slx75 芯片接收到新的激励信号时，Xc6slx9 芯片会同步发出一个触发信号。根据该信号利用采集工具 NI SignalExpress 调整示波器触发条件，自动采集功耗数据并以 Txt 格式保存数据，便于后续分析。

在数据收集阶段，示波器的第一个通道通过 SMA 至 BNC 的接口连接到 Xc6slx75 FPGA 的电压测量端口，以记录能耗数据。示波器的第二个通道配置为触发模式，并通过高阻抗探头来捕获触发信号。一旦在第二个通道检测到信号边沿的变化，就会启动两个通道的数据捕获工作，并通过 TCP 协议将捕获到的波形数据发送至 NI SignalExpress 采集软件，其记录实况如图 5-5 所展示。

在 SAKURA-G 平台上，每个 FPGA 芯片都通过一个独立的电源芯片来供电。在电源管理芯片和 FPGA 之间，串联了一个高精度的电阻 R，用于测量电压。假

设电源电压为 V_{CC}，通过两个 SMA 连接器来监测连接到电流感应电阻两端的电压 V_R。因此，可以使用公式 5-9 来计算 FPGA 芯片消耗的能量。

$$P = V_R \frac{V_{CC} - V_R}{R} \tag{5-9}$$

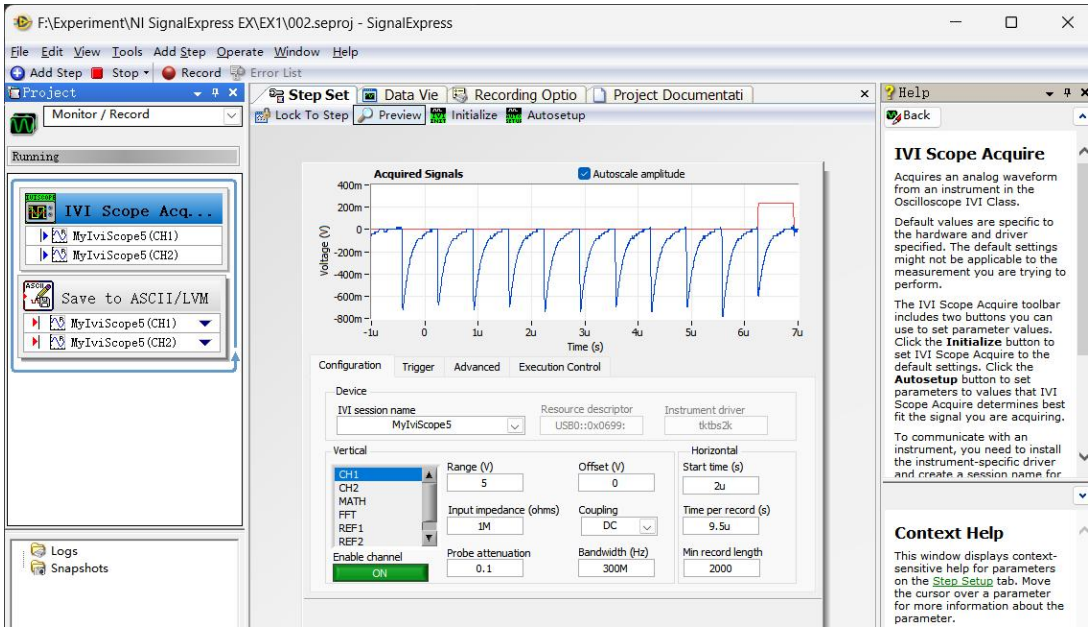


图 5-5　NI SignalExpress 记录的能量迹

（2）能量数据分析模块

在对数据分析前需要对其预处理，以便后续分析。由于所采集的能量迹是以时间戳为轴记录的瞬时电压数据，所以首先利用到式 5-9 将电压值转换为能量值。其次根据通道 2 的值，对触发信号以外的电压信号剔除，并保证所得到的能量迹可以数据对齐。最后完成 Txt 数据格式到 Excel 格式的转换，便于 Matlab 运行。图 5-6 所示为设计并使用的 C#数据预处理软件运行界面。

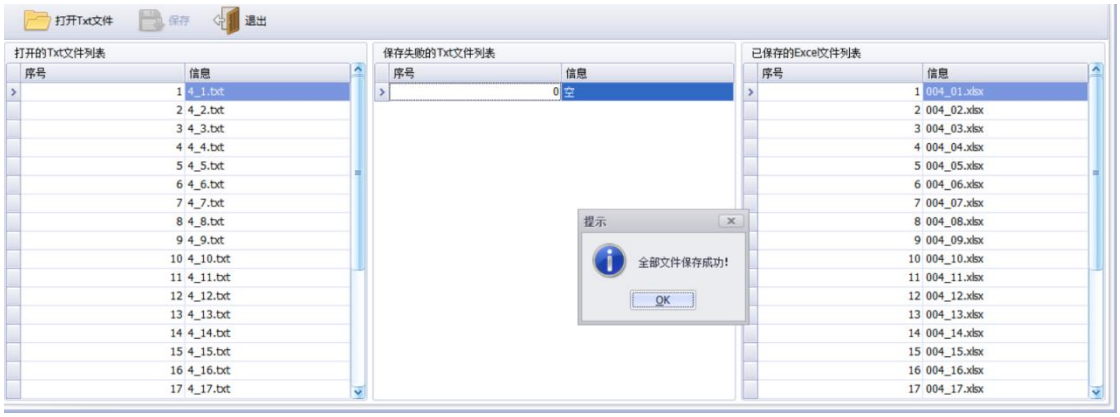


图 5-6　数据预处理软件界面图

数据预处理后，采用 Matlab 工具实现其能量数据分析流程，如图 5-7 所示。

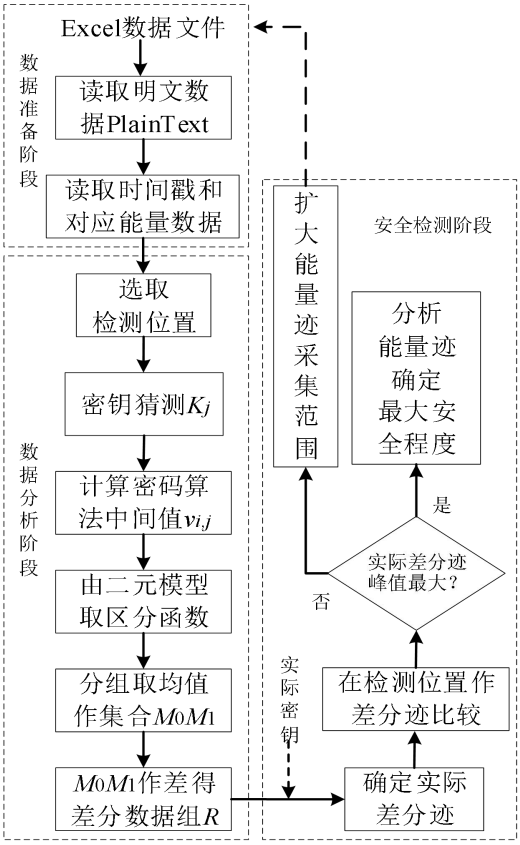


图 5-7　Matlab 数据分析流程图

Matlab 数据分析流程分为三个阶段：

第一阶段，数据准备阶段：读取数据预处理之后的 Excel 数据文件，获取每次加密操作的明文数据 *PlainText*，并且将每次采集的能量数据和时间戳与明文数据一一对应，方便下一步处理。

第二阶段，数据分析阶段：首先选取检测位置，由于初始轮密钥加为异或运算，其能量变化程度低，针对该位置的检测难度较高，而 S-盒变换是非线性运算具有数据扩散性，其数据会产生大量信号跳变及能量变化易于检测分析数据，因此本文将 AES 的首轮 S-盒的输出作为检测位置并根据该位置进行密钥猜测 K_j=8h'00~8'hff；接着计算中间值，先设相应的 8 位明文数据为 x_i，根据 AES 加密中运行轮密钥加之后再进行字节替换的过程，可得到其输出字节即为中间值 $v_{i,j}=Sbox(x_i \oplus k_j)$；最后选择 8 位数据最低位为区分准则定义区分函数 $h_{i,j}=LSB(v_{i,j})$，并将对应的能量迹划分成两组并求和作平均，得到差分数据组 $R=M_0-M_1$。

第三阶段，安全检测阶段：首先利用已知的实际密钥，从差分数据组 *R* 中找到实际差分迹；接着以实际差分迹为基础，在检测位置处与其它差分迹做对比，若实际差分迹峰值最大，则检测成功，反之则需要扩大采集能量迹的范围。最后，

根据现有的能量迹作递减分析，直至用最少的能量迹实现检测成功，对比能量迹相关参数分析密码算法的安全性。

5.2.2 安全检测平台验证

根据前文所述并结合 AES 密码算法的特点，本次实验将以总线结构为 32 位的 AES 加密算法为检测对象。数字存储示波器的采样率为 2*GSa/s*，实际密钥为 128'h 2b7e_1516_28ae_d2a6_abf7_1588_09cf_4f3c。

如图 5-8 所示，32 位总线结构 AES 加密过程首先需要将密钥和明文通过数据分组成 32 位，由于把密钥加载到“轮密钥生成”模块需要消耗 4 个时钟周期，而把明文装载到“AES 状态”模块也需要 4 个时钟周期，所以在加载明文的同时依次执行 AES 的初始 *AddRoundKey* 操作。在执行 9 轮标准加密的过程中，每一轮中都需要执行密钥扩展并且更新 AES 状态，为了对 32 比特数据执行 *SubBytes* 操作，还需要 4 个并行的 S-盒。由于加密 AES 的最后一轮不执行 *MixColumns* 操作，可在最后一个 *AddRoundKey* 操作之前，采用一个复用切换开关来绕开 *MixColumns* 操作，其他操作与标准的加密轮完全一样。在最后一轮之后，即可将加密算法得到的密文将储存在“AES 状态”模块中并输出。

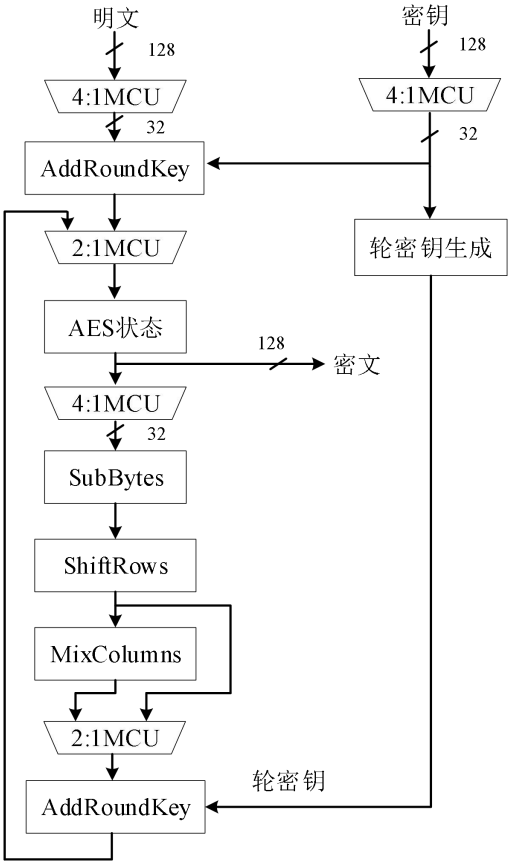


图 5-8　32 位总线结构 AES 加密过程图

基于 32 位总线结构 AES 密码算法的安全检测实验，是对后 8 位实际密钥 8'h3c 为检测目的实验。在硬件条件下存在许多未知的干扰因素，对能量迹采集范围和采集条件进行调试，得出只需采集 13000 余条能量迹就可以确定实际密钥，其中实验所用到的存储深度占用 2Mpts（实际采到的每条能量迹是由 20000 个点构成）。如图 5-9 所示，从该算法 DPA 曲线组可以看出在首轮轮变换第四位峰值处，实际密钥 k_{60}=8'h3c 所对应的 DPA 曲线其峰值最大，代表着实验检测成功，可得到该算法在该条件下最大安全性为 13000 次实验采集，且每次采集应需要由 20000 个采集点构成。

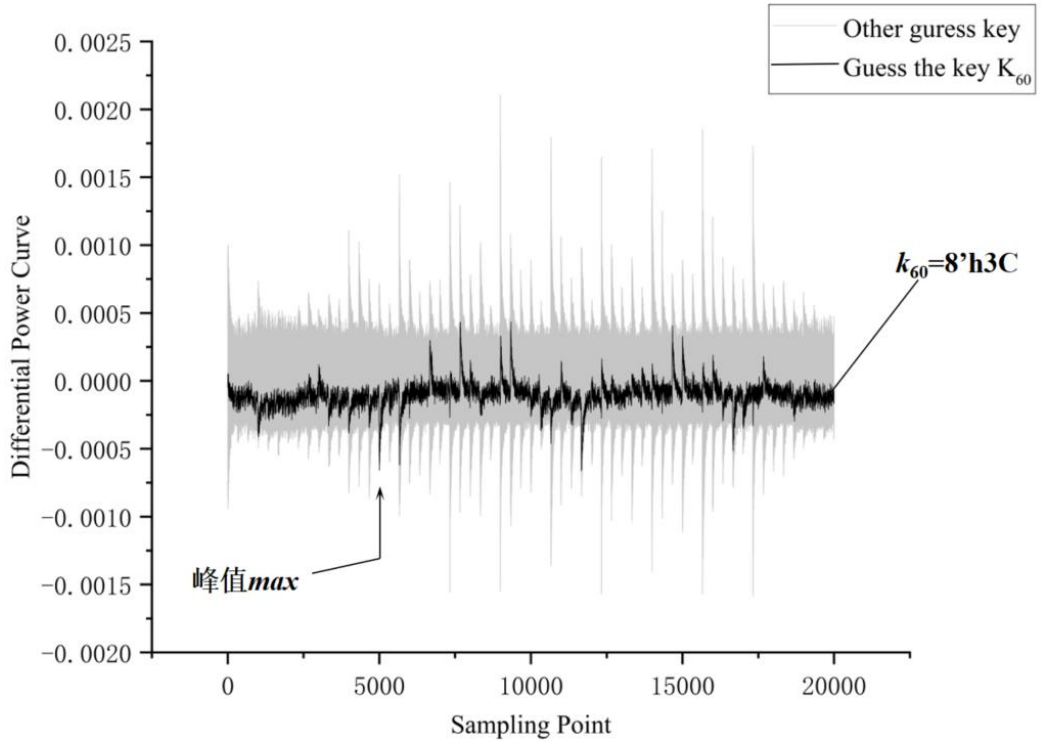


图 5-9　32 位总线结构 AES 密码算法差分能量迹

5.2.3 检测结果

本文将对前文所述的基于 128 位总线结构 AES 星载高速密码算法进行安全检测实验，密钥以首轮 128 位数据中的低 8 位为例，能量采集过程中的实际密钥为 8'h3c。经过对能量迹采集范围和采集条件的不断调试得出只需 10000 余条能量迹就可以确定实际密钥，并且实验所用到的存储深度只占用 1Kpts（实际采到的每条原始波是由 1000 个点构成）。如图 5-10 所示，从该算法 DPA 曲线组可以看出在首轮轮变换处，实际密钥 k_{60}=8'h3c 所对应的 DPA 曲线其峰值最大，代表着实验检测成功，可得到该算法在该条件下最大安全性为 10000 次实验采集，且每次采集应需要由 1000 个采集点构成。

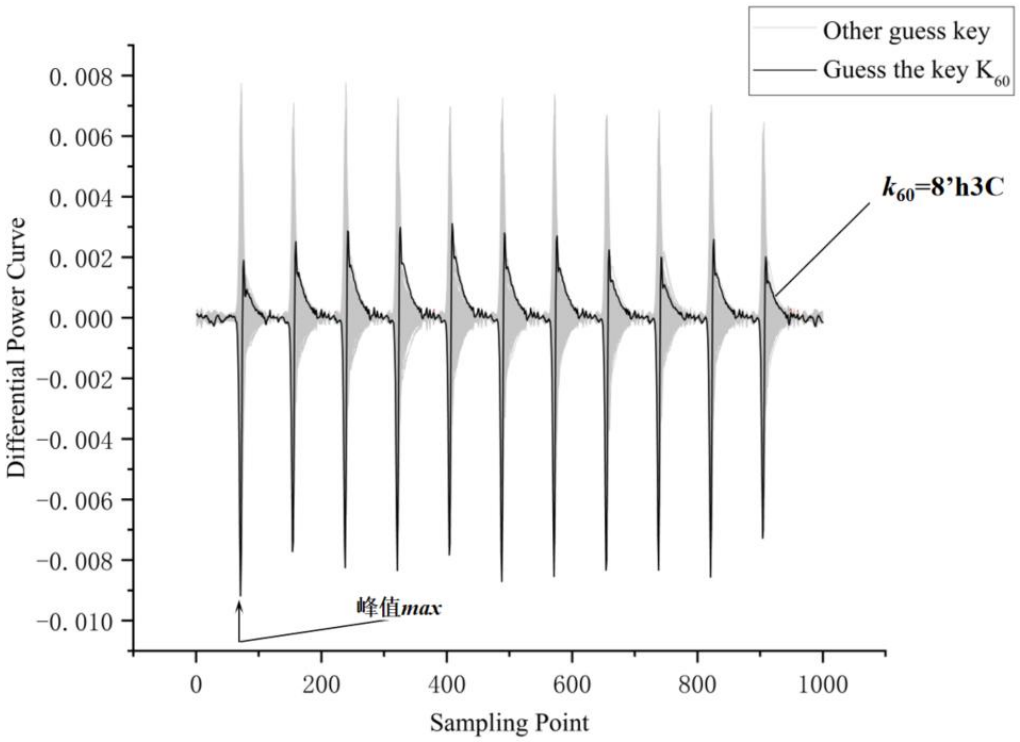


图 5-10　128 位总线结构 AES 星载高速密码算法差分能量迹

通过上述两个实验过程及结果，对比可以发现：针对 32 位总线结构的 AES 密码算法所需的检测条件比 128 位总线结构的 AES 星载高速密码算法复杂的多，即 32 位总线结构的 AES 密码算法在一定条件下更加安全。如在采集相同数量能量迹的情况下，基于 32 位总线结构的 AES 密码算法实验所采集的每条能量迹采样点个数是基于 128 位总线结构的 AES 密码算法实验的 20 倍。

这表明了在不改变 AES 算法的情况下，只需要调整加密数据的输入位宽，就可以在一定程度上影响 AES 算法实际运行的安全程度。输入数据位宽越小，安全检测实验中所需要采集和检测的过程就越繁杂，相应的其安全性就越高。

5.3 本章小结

本章第一部分介绍了能量分析。首先介绍了能量分析的原理和相关能量模型；然后具体介绍了差分能量分析法的原理和分析过程；最后重点对能量模型的选取和能量迹分析方式的使用进行了详细介绍。

本章第二部分主要介绍安全检测平台的设计与实现。首先介绍了基于硬件检测平台的结构设计；接着对平台各个部分的作用和运行方式进行逐一阐明，通过对 32 位总线结构 AES 密码算法的安全检测，为实验平台提供了验证基础；最后对前文所设计的 AES 星载高速密码算法基于该平台进行安全检测分析。

第 6 章 总结与展望

6.1 全文总结

作为密码技术首选的加密标准，AES 广泛应用于各个领域。本文针对卫星加密通信领域，以中国航天科工集团 xxx 研究所星载高速 AES 密码电路为研究对象，对卫星通信领域所面临的应用资源受限、密码电路高延迟等问题，提出高速 AES 密码电路实现方法，并为地面解密端提供相应的解密方式，保证解密过程的正常完成。在此基础上对所设计的星载高速 AES 密码电路利用安全检测平台对其安全性进行评估。本文主要研究内容总结如下：

（1）利用 AES-128 加密算法对电子侦察载荷 LVDS 接口数传数据进行加密，并分模块分单元对数据接口模块和加密运算单元进行仿真验证，成功将所设计的密码电路在星载 FPGA 上的实现。

（2）为保证所接收的加密数据信息可以顺利解密，本文针对地检设备中的 FPGA 进行解密电路的设计，电路主要由密钥扩展和解密运算两个单元构成，分模块分单元的设计解密模块以保证密钥与密文的准确对应，再进行仿真验证后成功在地检设备中实现，满足实际工程要求解决实际应用中的技术难题。

（3）利用延迟感知公共项消除算法 DACSE，对 AES 密钥扩展运算的表达式中公共项表达式 CS 进行提取，在基于延迟驱动二叉树结构 DDBT 构建低延迟的密钥扩展单元。通过 SMIC 0.18μm 技术的 Synopsys™ 集成电路 IC 综合工具 DC 的评估，得到在 AES 加密密钥扩展单元中 DDBT 结构的延迟以面积增加 10.08%的代价降低了 15.23%，降低了电路延迟。

（4）基于能量分析的原理，从攻击者的角度搭建安全性检测平台，对 32 位总线结构的 AES 密码算法电路进行能量数据的采集，并通过 Matlab 工具分析，成功获取该密码算法电路的实际密钥，验证平台的准确性。基于该平台再对上述星载高速 AES 加密密码电路进行能量数据的采集分析，实验结果表明只需在存储深度为 1Kpts 的条件下，采集 10000 余条能量迹即可确定实际密钥，确定该密码算法的实际安全性。

6.2 未来展望

未来，对星载高速 AES 密码电路的设计有以下几个展望：

（1）对所设计 AES 密码电路进行改进，保证高速的同时降低硬件资源消耗。

（2）对所搭建的安全性检测平台进行改进，提升采集数据的自动化，降低检测复杂度，还可采用 GPU 加速、软硬件结合等方法来提高检测效率。

（3）在本设计 AES 密码电路的基础上，通过应用掩蔽技术保护密码算法的中间计算结果，或采用隐藏策略对密码设备处理的中间值进行防护，可以增强密码算法的安全性。

参考文献

[1] 霍炜, 郁昱, 杨糠, 等. 隐私保护计算密码技术研究进展与应用 [J]. 中国科学: 信息科学, 2023, 53(9): 1688-1733.

[2] Albrecht J P. How the GDPR will change the world. Eur Data Protection Law Rev, 2016: 287-289.

[3] 国家密码管理局. 中华人民共和国密码法 [EB/OL], published online: https://www.oscca.gov.cn/sca/xxgk/2023-06/04/content_1057225.shtml, 2019.

[4] 杨波. 现代密码学(第 5 版) [M]. 北京: 清华大学出版社, 2022.

[5] Zheng X, Hu X H, Zhang J L, etc. An efficient and low-power design of the SM3 hash algorithm for IoT [J]. Electronics, 2019, 8(9): 1033.

[6] 国家密码管理局. SM3 密码杂凑算法 [EB/OL]. published online: http://www.sca.gov.cn/sca/xxgk/2010-12/17/1002389/files/302a3ada057c4a73830536d03e683110.pdf, 2010.

[7] 谷利泽, 郑世慧. 现代密码学教程 [M]. 北京: 北京邮电大学出版社, 2009.

[8] 谢永泉. 我国密码算法应用情况[J]. 信息安全研究, 2016, 2(11): 969-971.

[9] 张肖强. 基于复合域运算的 AES 密码电路优化设计方法研究 [D]. 南京:南京航空航天大学, 2016.

[10] D. Hankerson, A. Menezes, and S. Vanstone. Guide to Elliptic Curve Cryptography [C]. New York, NY, USA: Springer-Verlag, 2004.

[11] 牛永川. SM2 椭圆曲线公钥密码算法的快速实现研究 [D]. 山东大学, 2013.

[12] G. D. Sutter, J.-P. Deschamps, and J. L. Imana. Modular multiplication and exponentiation architectures for fast RSA cryptosystem based on digit serial computation [J]. IEEE Trans. Ind. Electron, 2011, 58(7): 3101-3109.

[13] 董礼玲. 基于 AES 算法的抗功耗分析密码芯片的优化设计研究 [D]. 南京: 南京航空航天大学硕士论文, 2016.

[14] 曾清扬. DES 加密算法的实现 [J]. 网络安全技术与应用, 2019(7): 33-34.

[15] 何明星, 林昊. AES 算法原理及其实现 [J]. 计算机应用研究, 2002: 61-63.

[16] 吕述望, 苏波展, 王鹏, 等. SM4 分组密码算法综述 [J]. 信息安全研究, 2016, 2(11): 995-1007.

[17] Sarkar S. Further non-randomness in RC4, RC4A and VMPC [J]. Cryptography

& Communications, 2015, 7(3):317-330.

[18] 冯秀涛. 3GPP LTE 国际加密标准 ZUC 算法 [J]. 信息安全与通信保密, 2011(12): 45-46.

[19] NIST, FIPS PUB. Announcing the Advanced Encryption Standard (AES) [S], published online: https://csrc.nist.gov/files/pubs/fips/197/final/docs/fips-197.pdf, 2001.

[20] 张敬源. AES 算法在数据安全加密中的应用 [J]. 现代工业经济和信息化, 2023, 13(3): 75-76.

[21] 郝玉洁，吴立军，赵洋等. 信息安全概论 [M]. 北京: 清华大学出版社, 2013.

[22] 隋谦. 一种非常紧凑的掩码 AES 的硬件设计与实现 [D]. 山东: 山东大学, 2023.

[23] Paul C Kocher. Cryptanalysis of diffie-hellman, rsa, dss, and other systems using timing attacks. In Extended abstract. Citeseer, 1995.

[24] X. Zhang, and K. K. Parhi. Implementation Approaches for the Advanced Encryption Standard Algorithm [J]. IEEE Circuits and Systems Magazine, Vol. 2, Issue. 4, pp. 24-46, 2002.

[25] Kocher P, Jaffe J, Jun B. Differential power analysis [C]. Annual international cryptology conference. Springer, Berlin, Heidelberg, 1999: 388-397.

[26] Mayer-Sommer R. Smartly analyzing the simplicity and the power of simple power analysis on smartcards [C]. Cryptographic Hardware and Embedded Systemsd CHES 2000: Second International Workshop Worcester, MA, USA, August 17-18, 2000 Proceedings 2. Springer Berlin Heidelberg, 2000: 78-92.

[27] W. Shan, X. Fu and Z. Xu. A Secure Reconfigurable Crypto IC With Countermeasures Against SPA, DPA, and EMA [J]. In IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2015, 34(7): 1201-1205.

[28] H. Mestiri, F. Kahri, B. Bouallegue, etc. A high-speed AES design resistant to fault injection attacks [J]. Microprocessors and Microsystems, 2016, 41: 47-55.

[29] Piret G, Quisquater J J. A Differential Fault Attack Technique against SPN Structures, with Application to the AES and Khazad [M]. Cryptographic Hardware and Embedded Systems, 2003:77-88.

[30] M. Tunstall, D. Mukhopadhyay, and S.S. Ali. Differential Fault Analysis of the Advanced Encryption Standard Using a Single Fault [C]. Information Security

Theory and Practice. Security and Privacy of Mobile Devices in Wireless Communication. Berlin Heidelberg, 2011, 6633: 224-233.

[31] S. Endo, Y. Li, N. Homma, etc. A Silicon-Level Countermeasure Against Fault Sensitivity Analysis and Its Evaluation [J]. In IEEE Transactions on Very Large Scale Integration of (VLSI) System, 2015, 23(8): 1429-1438.

[32] P. C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems [C]. International Conference on Advances in Cryptology, 1996:104-113.

[33] C. Rebeiro, and D. Mukhopadhyay. Micro-Architectural Analysis of Time-Driven Cache Attacks: Quest for the Ideal Implementation [J]. IEEE Transactions on Computers, 2015, 64(3): 778-790.

[34] D. Agrawal, B. Archambeault, JR. Rao, etc. The EM Side-Channel(s) [J]. Cryptographic Hardware and Embedded Systems-CHES, 2003: 29-45.

[35] A. Dehbaoui, JM. Dutertre, B. Robisson, etc. Electromagnetic Transient Faults Injection on a Hardware and a Software Implementations of AES [J]. In Proceedings of the International Workshop on Fault Diagnosis Tolerance Cryptography FDTC, 2012: 7-15.

[36] A. Barenghi, L. Breveglieri, I. Koren, etc. Fault Injection Attacks on Cryptographic Devices: Theory, Practice, and Countermeasures [J]. Proceedings of the IEEE, 2012, 100(11): 3056-3076.

[37] Goller G, Sigl G. Side channel attacks on smartphones and embedded devices using standard radio equipment [C]. Constructive Side-Channel Analysis and Secure Design 6th International Workshop COSADE, 2015: 255-270.

[38] Gandolfi K. Mourtel C, Olivier F. Electromagnetic Analysis of Concreteresults [C]. Cryptographic Hardware and Embedded Systems CHES, 2001: 251-261.

[39] Le T H, Clédière J, Canovas C, etc. A proposition for correlation power analysis enhancement [C]. Cryptographic Hardware and Embedded Systems-CHES 2006: 8th International Workshop, 2006: 174-186.

[40] Clavier C, Feix B, Gagnerot G, etc. Improved collision-correlation power analysis on first order protected AES [C]. Cryptographic Hardware and Embedded SystemsCHES 2011: 13th International Workshop, 2011: 49-62.

[41] O. Coudert. Gate Sizing for Constrained Delay/Power/Area Optimization [J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 1997, 5(4): 465-471.

[42] R. R. Rach, P. V. A. Mohan, B. S. A. Efficient implementations for AES encryption and decryption [J]. Circuits, Systems, and Signal Processing, 2012, 31(5): 1765-1785.

[43] Dao M-H, Hoang V-P, Dao V-L, etc. An Energy Efficient AES Encryption Core for Hardware Security Implementation in IoT Systems [C]. 2018 International Conference on Advanced Technologies for Communications (ATC)， 2018: 301-304.

[44] Abdellatif K M, Chotin-Avot R, Mehrez H. AES-GCM and AEGIS: Efficient and High Speed Hardware Implementations [J]. Journal of Signal Processing Systems, 2017, 88(1): 1-12.

[45] Zhang X, Parhi K K. High-speed VLSI architectures for the AES algorithm [J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2004, 12(9): 957-967.

[46] Liang Y, Li Y, Zhang C. High-throughput cost-effective and low-power AES chip design [C]. Image and Signal Processing (CISP), 2010, 3(9): 4226-4229.

[47] Chen D, Shou G, Hu Y, etc. Efficient architecture and implementations of AES [C]. Advanced Computer Theory and Engineering (ICACTE), 2010, 3(6): 295-298.

[48] P. V. S. Shastry, N. Somani, A. Gadre, B. Vispute, etc. Rolled Architecture Based Implementation of AES Using T-Box [C]. 2012 IEEE 55th International Midwest Symposium on Circuits and Systems (MWSCAS), 2012: 626-630.

[49] D.-S. Kundi, A. Aziz, N. Ikram. A high performance ST-Box based unified AES encryption/decryption architecture on FPGA [J]. Microprocessors & Microsystems, 2016, 41: 37-46.

[50] N. Petra, D. D. Caro, and A. G. M. Strollo. A Novel Architecture for Galois Fields GF(2^m) Multipliers Based on Mastrovito Scheme [J]. IEEE Transactions on Computers, 2007, 56(1): 1470-1483.

[51] X. Zhang, N. Wu, F. Zhou, etc. Low-delay parallel Chien search architecture for RS decoder [J]. IEICE Electronics Express, 2016. 1-6.

[52] X. Zhang, N. Wu, F. Zhou, X. Chen. An optimized delay-aware common subexpression elimination algorithm for hardware implementation of binary-field linear transform [J]. IEICE Electronics Express, 2014, 1-8.

[53] I. Hammad, K. El-Sankary, E. El-Masry. High-Speed AES Encryptor With

Efficient Merging Techniques [J]. IEEE Embedded Systems Letters, 2010, 2(3): 67-71.

[54] T. Q. Vinh, J.-H. Park, Y.-Ch. Kim and K.-O. Kim. An FPGA implementation of 30Gbps security module for GPON systems. 2008 8th IEEE International Conference on Computer and Information Technology, 2008, 868-872.

[55] A. Satoh, S. Morioka, K. Takano, and S. Munetoh. A compact Rijndael hardware architecture with S-box optimization [C]. International Conference on the Theory and Application of Cryptology and Information Security, 2001: 239-245,.

[56] C. C. Lu and S. Y. Tseng. Integrated design of AES (Advanced Encryption Standard) encrypter and decrypter [C]. In Process Application-Specific System Architectures Processors, 2002: 277-286.

[57] S. Mangard, M. Aigner and S. Dominikus, “A highly regular and scalable AES hardware architecture,” in IEEE Transactions on Computers, 2003: 483-491.

[58] T. Good and M. Benaissa. AES on FPGA from the fastest to the smallest [C]. In Proc. 7th Int. Workshop on Cryptographic Hardware and Embedded Systems (CHES 2005), LNCS 3659: 438-451.

[59] N. Pramstaller, S. Mangard, S. Dominikus, J. Wolkerstorfer. Efficient AES Implementations on ASICs and FPGAs [C]. 4th International Conference on AES, 2004: 98-112.

[60] L. Yi, X. Zou, Z. Liu, etc. A low-cost compact AES architecture for wireless sensor network [J]. High Technology letter, 2010, 16(21): 184-188.

[61] M. Feldhofer, S. Dominikus, J. Wolkerstorfer. Strong Authentication for RFID Systems Using the AES Algorithm [C]. 2004 Workshop on Cryptographic Hardware and Embedded Systems (CHES 2004), 2004: 357-370.

[62] P. Hämäläinen, T. Alho, M. Hännikäinen, and T. D. Hämäläinen. Design and Implementation of Low-area and Low-power AES Encryption Hardware Core [C]. Process 9th Euro Micro Conference Digital System Design (DSD2006), 2006: 577-583.

[63] M. Kim, J. Ryou, Y. Choi, and S. Jun. Low Power AES Hardware Architecture for Radio Frequency Identification [C]. 2006 International Workshop on Security (IWSEC 2006), 2006: 353-363.

[64] T. Good, and M. Benaissa. 692-nW Advanced Encryption Standard (AES) on a 0.13-μm CMOS [J]. IEEE Transaction on Very Large Scale Integration (VLSI) Systems, 2010, 18(12): 1753-1757.

[65] O. Song, J. Kim. Compact Design of the Advanced Encryption Standard Algorithm for IEEE 802.16.4 Devices [J]. Journal of Electrical Engineering & Technology, 2011, 6(3): 418-422.

[66] M.-C. Chen and W.-T. Li. An 8-Bit ROM-Free AES Design for Low-Cost Applications [J]. Mathematical Problems in Engineering, 2013: 1-6.

[67] H. K. Kim, and M. H. Sunwoo. Low Power AES Using 8-Bit and 32-Bit Datapath Optimization for Small Internet-of-Things (IoT) [J]. Journal of Signal Processing Systems 91, 2019: 1283-1289.

[68] P. Chodowiec and K. Gaj. Very Compact FPGA Implementation of the AES Algorithm [C]. Proceedings of the 5th International Workshop on Cryptographic Hardware and Embedded Systems, 2003(3): 319-333.

[69] G. Rouvroy, F. X. Standaert, J. J. Quisquater, J. D. Legat. Compact and Efficient Encryption/Decryption Module for FPGA Implementation of the AES Rijndael Very Well Suited for Small Embedded Applications [C]. International Conference on Information Technology: Coding and Computing, 2004: 583-587.

[70] P. Hämäläinen, M. Hännikäinen, and T. Hämäläinen. Efficient hardware implementation of security processing for IEEE 802.16.4 wireless networks [C]. In Process 48th IEEE Interaction Midwest Symposium on Circuits and Systems (MWSCAS 2005), 2006: 484-487.

[71] N. Yu and H. M. Heys. Investigation of compact hardware implementation of the advanced encryption standard [J]. Canadian Conference on Electrical and Computer Engineering, 2005: 1069-1072.

[72] Z. Jia, X. Zeng, J. Han, J. Chen. Very Low-cost VLSI Implementation of AES Algorithm [J]. IEEE Asian Solid-State Circuits Conference, 2006.

[73] Z.-R. Li, Y.-Q. Zhuang, C. Zhang, G. Jin. Low-power and area-optimized VLSI implementation of AES coprocessor for Zigbee system [J]. The Journal of China Universities of Posts and Telecommunications, 2009, 16(3): 89-94.

[74] P.-C. Liu, H.-C. Chang and C. -Y. Lee. A 1.69 Gb/s area-efficient AES crypto core with compact on-the-fly key expansion unit, 2009 Proceedings of ESSCIRC, 2009: 404-407.

[75] Messerges T S, Dabbish E A, Sloan R H. Investigations of Power Analysis Attacks on Smartcards [J]. Smartcard, 1999, 99: 151-161.

[76] Mangard S. A simple power-analysis (SPA) attack on implementations of the AES key expansion [C]. International Conference on Information Security and

Cryptology. Springer, 2002: 343-358.

[77] Ors S B, Gurkaynak F, Oswald E, etc. Power-analysis attack on an ASIC AES implementation [C]. International Conference on Information Technology: Coding and Computing, 2004, 2: 546-552.

[78] Mangard S, Pramstaller N, Oswald E. Successfully attacking masked AES hardware implementations [C]. International Workshop on Cryptographic Hardware and Embedded Systems, 2005: 157-171.

[79] MORADI A, SCHNEIDER T. Improved side-channel analysis attacks on Xilinx bitstream encryption of 5, 6, and 7 series [C]. 7th International Workshop, 2016: 71-87.

攻读学位期间发表的学术论文及参加的科研项目

[1] **Mingyu Xu**, Xiaoqiang Zhang, Xinxing Zheng, Zhang Xing. Security analysis and simulation of hardware circuits based on Advanced Encryption Standard algorithm of different bus structures [C]. 2024 3th International Conference on Electronic Information Engineering, Big Data and Computer Technology.

参与项目：

[1] 横向项目，中国航天科工集团 xxx 研究所项目，“AES-128 加密算法”，项目周期：2023.07.14-2024.07.14

致谢

三年千日，聚散有时。时钟不知不觉转动到此刻，意味着我的研究生生活即将结束，纵使有万般不舍，也要告别校园、老师和同学，告别我那呼朋引伴的读书岁月。立在山巅，回望来路，自己的思想和学识都有了很大的进步。在此，要多谢陪伴我一路走过的人们，有你们的陪伴，才使我的研究生生活更加丰富多彩。

一朝沐杏雨，一生念师恩。感恩我的导师张肖强副教授，您细致严谨的治学态度深深感染了我，让我在三年的硕士学习中进步良多。张老师对待学生细心又耐心，当我们在课题上感到迷茫的时候，帮助我们找到研究的方向。在科研过程中遇到问题时，张老师都会认真作答，会从专业的角度提出建议并为我们加油鼓劲。在论文撰写中，传授给我们写论文的方法和技巧。在生活上，张老师不但关心我们的身体健康，更关心我们的心理健康，希望我们在读研期间，认真科研，开心科研。我相信今后与张老师不仅是短暂的师徒关系亦是一生的挚友。同时感谢研究生三年所有的授课老师，为我传授诸多知识，在学习中给了我极大的鼓励。在此表示真挚的感谢，祝愿老师们工作顺利，平安幸福。

慈母手中线，游子身上衣。我要感谢我的家人在生活上对我的照顾，让我能够安心学习。感谢你们一直以来的爱与支持，感谢你们在我每一个开心与不开心的日子里的陪伴。家，永远是我最温暖的港湾，是我在遇到困难与压力的时候最最想要回到的地方。从蹒跚学步到远行深思，对父母的恩情自是没齿难忘，二十余年的养育之恩，不是一朝一夕，只希望父母长乐久安。

山河不足重，重在遇知己。感谢师兄们在生活和学习中提的建议，感谢我的同门和好友在生活上的照顾和学业上的支持，是你们在我这三年里留下了无数的温暖，是你们让我对这个校园又多了一份留念。初见乍惊欢，久处仍怦然，人海茫茫，得以相遇，何其有幸。祝你们未来平安喜乐，前程似锦。

一程山水一年华，独愿此去经年，一生磊落，一生纯善。