第 34 卷 第 20 期 计 算 机 工 程 2008 年 10 月
Vol.34 No.20 Computer Engineering October 2008
·安全技术· 文章编号：1000—3428(2008)20—0175—02 文献标识码：A 中图分类号：TP309

基于硬件的 AES 算法

张九华，胡廉民
(乐山师范学院物理与电子信息科学系，乐山 614004)

摘 要：分析 AES 算法原理，构建基于 FPGA 的硬件实现框架，描述数据加解密单元和密钥扩展单元的工作机制和硬件结构，引入核心运算模块复用的设计思想，在不影响系统效率的前提下降低芯片资源的使用率，并对该系统结构进行了芯片级的验证。实验结果表明，在 38 MHz 工作频率下，该系统的处理速度为 405 Mb/s。

关键词：高级加密标准；分组密码；加密

Hardware-based AES Algorithm

ZHANG Jiu-hua, HU Lian-min
(Department of Physics and Electronic Information, Leshan Teachers College, Leshan 614004)

【Abstract】 This paper analyzes the basic principle of the AES algorithm, and presents a hardware implementation structure based on FPGA. The principle description and hardware implementation structure of data encryption modules and key expansion modules are presented. In the structure, the core computing module reuse design idea is joined which can reduce the use of chip resources without impacting system efficiency. The chip-level test of the system shows the system processing speed can reach 405 Mb/s based on the clock frequency of 38 MHz.

【Key words】 Advanced Encryption Standard(AES); block cipher; encryption

随着分布式计算和并行处理技术的发展、差分攻击和线性攻击技术的完善，原有的 56 bit 密钥的 DES 体制已经难以满足公用数据加密标准算法的要求。为此，美国国家标准和技术研究所(NIST)于 2000 年最终确定 Rijndael 为 AES 算法。Rijndael 算法是一种快速、高效的分组密码加解密算法，易于实现。本文给出一种基于 FPGA 的硬件实现方法，采用“基本结构”，因为“基本结构”和反馈、非反馈方式相比，有相同的效率，但其占用资源较少。实验证明了该方法的有效性。

1 算法描述

Rijndael 算法是一种迭代分组密码算法，采用的是替代/置换网络(SPN)，对一个 128 bit 的数据块进行加解密操作。加密时，首先将输入的 128 bit 数据排成 4×4 的字节矩阵，然后根据不同密钥长度，进行 10 轮(128 bit 密钥)、12 轮(192 bit 密钥)或 14(256 bit 密钥)轮的运算。其中，每个轮函数由以下几层组成：

(1)第 1 层(盒变换)为非线性层，将一个 8×8 的 S 盒应用于阵列的每个字节。

(2)第 2 层(行移位变换)和第 3 层(列混合)是线性混合层，将 4×4 的阵列按行位移，按列混合。

(3)在第 4 层(加密钥变换)中，轮密钥被异或到阵列的每一个字节。

每一轮的基本操作流程如图 1 所示。

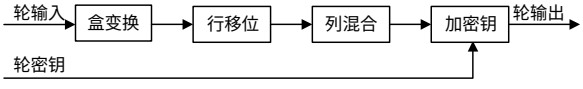

图 1 轮变换过程

当进行轮操作时，最后一轮的操作和前面的操作不同，它没有列混合这一步。解密过程和加密过程类似，将轮变换中的 4 个部分分别替换为相反的操作即可[1]。

2 算法的硬件设计

2.1 总体结构

一个密码算法的数据一般要完成以下几项工作：数据的输入/输出，密钥的扩展，加解密的运算等。根据以上密码算法的基本原则，Rijndael 算法的整体结构如图 2 所示。

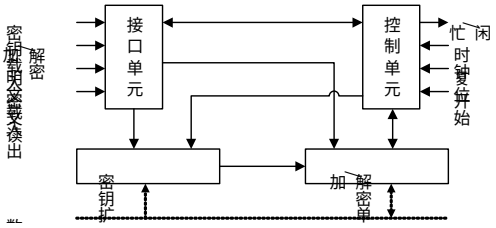

图 2 Rijndael 算法的整体结构

在图 2 中，虚线代表数据线；实线代表控制线。控制信号分别从接口模块和控制单元进入，数据和密钥通过数据总线进入。依靠地址和密钥载入、数据载入的配合来控制数据总线上数据的流向，根据控制模块的忙闲指示来决定加解密是否完成，根据读出信号来将加/解密结果输出。当然也可将控制模块和接口模块相结合，效果相同，但会增加一定的复杂度。下面主要讨论密钥扩展模块和加解密单元的设计，这也是整个算法的核心部分。

基金项目：四川省教育厅科研基金资助项目(2006B074)
作者简介：张九华(1976－)，男，讲师、硕士，主研方向：信息安全；胡廉民，讲师
收稿日期：2007-11-20 **E-mail：**lstcwd@163.com

2.2 加解密单元

当采用基本结构进行设计时，往往同时采用同步设计。由于轮运算的 4 个步骤都是逻辑运算，因此可以在一个周期内完成一轮或多轮的运算。采用在一个周期内完成全部加密运算的方法，速度最快可以达到 1 950 Mb/s，这也是反馈模式下的最快速度[2]，但是由于没有复用运算结构，对芯片资源的消耗很大。考虑到资源和速度的均衡，在本文的设计中，一个周期内只完成一轮运算，保证每轮的运算结构下一轮都可以复用，这样的设计可以节省大量的芯片资源。轮运算的整体结构如图 3 所示。

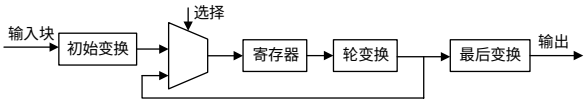

图 3　轮运算结构

多路选择器通过选择信号将初始变换后的输出存入寄存器。接下来在每个时钟周期内进行一次轮变换，然后将轮中间状态反馈回寄存器，当到达最后一轮时，进行最后的变换输出。由此可得：

系统吞吐量＝输入块长度/时钟周期×加密的轮数　　(1)

分析式(1)可知，只要能简化轮变换中的逻辑运算，就可以减少单轮变换的时钟周期，从而提高整个系统的吞吐量。对于轮变换的 4 个步骤来说，行移位(反行移位)运算是不占用任何资源的。同样，加密钥运算也可以只用简单的异或门来实现。关键的路径延迟在于盒变换和列混合运算[3]。可以选择查找表的方法，因为这种方法在实现GF(2^8)的模逆时，具有最快的速度，而且由于现在的FPGA芯片中很多都拥有大量的片内ROM，可以方便地将查找表固化在芯片内部。在Rijndael算法中，加密和解密同时需要 16 个查找表，每个查找表需要 2 048 bit，加上密钥扩展需要的 4 个查找表，合计需要 20×4 096＝81 920 bit的存储空间来储存。为了降低系统的资源占用，将 128 bit的数据分成 4 个部分，来分别进行查找，这样查找表资源可以降低到原来的 1/4[4-5]，但同时原来的一轮变换现在就需要 4 个周期来完成，系统的速度会大大降低，同时增加了整个设计的复杂度。可以采用对 128 bit数据同时查找的方法。

在列混合运算中，对轮中间状态的每一列进行矩阵相乘，需要完成GF(2^8)中的乘法{02}X,{03}X。假定X是轮中间状态的一个字节，表示为{X_7,X_6,X_5,X_4,X_3,X_2,X_1,X_0}，那么{02}X和{03}X能够通过如式(2)、式(3)的化简计算得到：

{02}X={X_6,X_5,X_4,X_3,X_2,X_1,X_0,0}xor{0,0,0,X_7,X_7,0,X_7,X_7}　　(2)

{03}X={02}X xor X　　(3)

0 xor X = X，故只需要 4 个异或门就可以完成式(2)的操作。为了便于简化，将{02}X设计成一个XTime 模块，具体结构如图 4 所示。根据列混合运算规则[6-7]，设计的最简单的列混合运算的结构如图 5 所示。

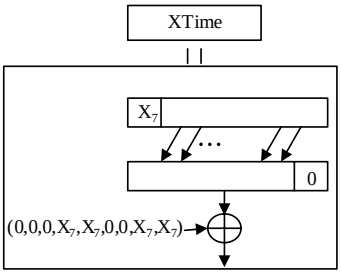

图 4　Xtime 结构

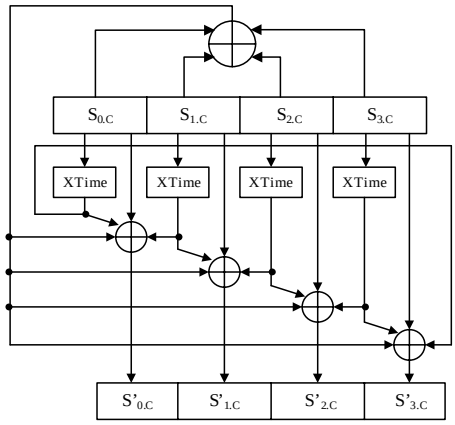

图 5　列混合运算结构

对于反列混合的变换，用和列混合变换相类似的方法，只是反列混合变换需要用到更多的 XTime 模块，因此，相应的延迟会更大一些。

2.3 密钥扩展单元

密钥扩展的方法通常有内部扩展和外部扩展两种方法。如利用外部扩展，只需按照 AES 算法说明逐步施行即可。但如利用内部扩展，则必须考虑到密钥扩展的速度，它必须小于加解密模块运行的时间，否则会系统出现时序错误。利用外部密钥扩展，事先需要大量的存储空间来存储中间密钥，为了降低存储成本，本文采用了内部扩展的方式，设计的基本结构如图 6 所示。

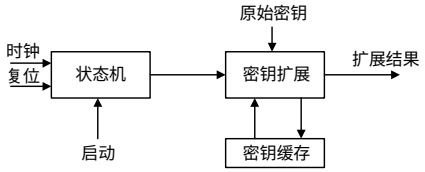

图 6　密钥扩展模块结构

在密钥扩展过程中，首先由控制模块发出启动信号，促使状态机启动，然后在时钟控制下变换状态，根据状态的不同，完成密钥扩展的算法，由于在扩展过程中需要以前的扩展结果，因此密钥的缓存是不可缺少的。但考虑到密钥扩展算法的规律性，所需的存储空间比起外部扩展来说是大大降低了。同时因为密钥扩展也是逻辑运算的组合，而且其关键路径小于加解密算法的关键路径，所以子密钥的扩展会比加解密率先完成，不会出现错误。

3　系统性能分析和结论

整体结构采用 RTL 级的 VHDL 语言实现，开发环境采用 XilinxISE6.1 和 MODELSIM5.7SE，目标器件选择了 Xilinx virtex 系列的 XCV-1000-5，其内部含有丰富的逻辑资源和片内 ROM，利于本设计的实现。整个设计的最大的系统频率为 38 MHz，不考虑测试部分所占用的验证周期，工作时实际最大的系统吞吐量可以达到 405 Mb/s。整个设计消耗 1 435 个 CLBs、18 个 ROM 块。

相比于其他各种实现方法，基本结构的实现是最简单的，其内部没有使用流水设计，因此，其降低了控制的复杂度，同时由于对各种加密模式都有相同的效果，故可以很方便地应用于各种应用平台上而不需要大量的修改。本文主要讨论的是基本结构的一般设计，如何将之进一步应用到各种结构方式(如：CFB,CBC,OCB)中，是下一步工作的方向。

(下转第 179 页)