AES 中有限域运算的优化及算法高速实现

沈启峰, 黄士坦, 杨 靓
(西安微电子技术研究所, 陕西 西安 710075)

摘 要: 介绍有限域的概念及 Rijndael 算法的结构, 详细分析了算法中基于加法、乘法的运算过程, 为使运算更适合在 FPGA 平台实线, 可使用一些技巧达到优化目的。详细阐述了使用 FPGA 高速实现运算关键部分的设计思路。针对 FPGA 设计中对速度与面积两项指标的不同要求, 给出了两种设计方案。最后, 给出算法在 FPGA 实现方式下的性能比较。

关键词: 有限域运算; Rijndael; AES; FPGA 实现
中图分类号: TP309.7 文献标识码: A 文章编号: 1005-3751(2005)12-0015-03

FPGA Optimization Implementation for AES Algorithm

SHEN Qi-feng, HUANG Shi-tan, YANG Liang
(Xi'an Institute of Microelectronic Technology, Xi'an 710075, China)

Abstract: An introduction to the concept of finite-field and the structure of Rijndael algorithm is given in this paper. It includes a detailed and optimized analysis of addition and multiplication based on finite-field and presents an implementation of key circuit on FPGA. Aiming at the different requirements for speed and area in FPGA, presents two kinds of designing scheme. At the end, the paper compares the implementation of FPGA with other reference in performance.

Key words: finite-field arithmetic; Rijndael; AES; FPGA implementation

0 引 言

随着计算机与数据通信技术的高速发展, 基于各种安全算法的加解密硬件广泛应用于卫星通信、网关服务器、机顶盒、视频传输以及其它高速数据传输业务中。网络通信数据安全逐渐成为计算机科学技术的热点领域。从技术角度讲, 网络安全除了依赖安全的网络通信协议及应用协议外, 更多地取决于网络设备如交换机、路由器等所提供的加解密功能。

自 20 世纪 70 年代以来一直广泛使用的"数据加密标准"(DES) 日益显出衰老的痕迹, AES(Advanced Encryption Standard) 是美国国家标准与技术研究所(NIST) 筹划的, 旨在取代 DES, 以保护 21 世纪敏感政府信息的新型加密标准。1997 年 4 月 NIST 开始公开征集, 经过 3 年的评比、审定。2000 年 10 月 2 日, NIST 宣布 Rijndael 算法当选 AES。该算法是迭代分组密码算法, 其分组长度和密钥长度都可改变, 算法的扩充形式允许分组长度和密钥长度以 32bit 的步长, 从 128bit 到 256bit 范围内进行特定的变化。算法的主要优点是: 设计简单, 密钥安装快, 需要的内存空间少, 在所有平台上运行良好, 支持并行处理, 可抵御目前所有已知攻击[1]。

抽象代数的发展引入了群、环、域的概念, 使得代数运算达到了高度的统一, 为密码学提供了有利的数学工具。AES 在设计理论上很大程度地依赖有限域的特殊性质, 其中, 基于有限域的运算为算法的安全性提供可靠保障。

1 AES 算法原理

AES 的结构可分为密钥扩展阶段和 4 个加密阶段(包括一次混淆运算和三次代换运算)。下面依次介绍(以 128 分组情况)[2~5]。

①密钥扩展:

AES 算法的密钥长度用 Nb 表示(Nb 反映了密钥中的 32 位字的数目), 分别取值 4, 6, 8, 对应 128, 192, 256 三种分组情况。其运算的轮数用 Nr 表示, 分别取 10, 12, 14。密钥扩展一共生成 $Nb(Nr+1)$ 个字, 每一个字有 32 位, 算法要求初始时有 Nb 个字, Nr 轮运算中的每一轮都要求有 Nb 个字。

②字节代换:

用一个 S 盒函数完成分组中的按字节的代换。可用代数式表达为 $b_{ij} = S[a_{ij}]$, $S[\,]$ 是一个变换函数, 功能表现为 8 输入 8 输出的查找表。

③行移位:

一个简单的置换操作, 即输入分组的 STATE 矩阵的

收稿日期: 2005-03-09
作者简介: 沈启峰(1975—), 男, 江苏苏州人, 硕士研究生, 主要研究方向为嵌入式计算机系统结构、数字图像信号处理、FPGA 设计。

第一行保持不变, 第二行循环左(右) 移一个字节, 第三行循环左(右) 移二个字节, 第四行循环左(右) 移三个字节。

④列混淆:

一个利用在域上 GF(2^8) 的算术特性的代换。其代数式为

$$\begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}$$

⑤轮密钥加:

利用当前分组和扩展密钥的一部分进行按位异或操作。代数式表达为 $b_{ij} = a_{ij} \oplus W_{ij}$。

AES128 算法结构简单, 对加密和解密操作, 算法由轮密钥加开始, 接着执行九轮迭代运算, 每轮都包含所有 4 个阶段的代换, 然后执行只包含 3 个阶段的最后一轮运算。图 1 描述了包含全部加密轮的结构。

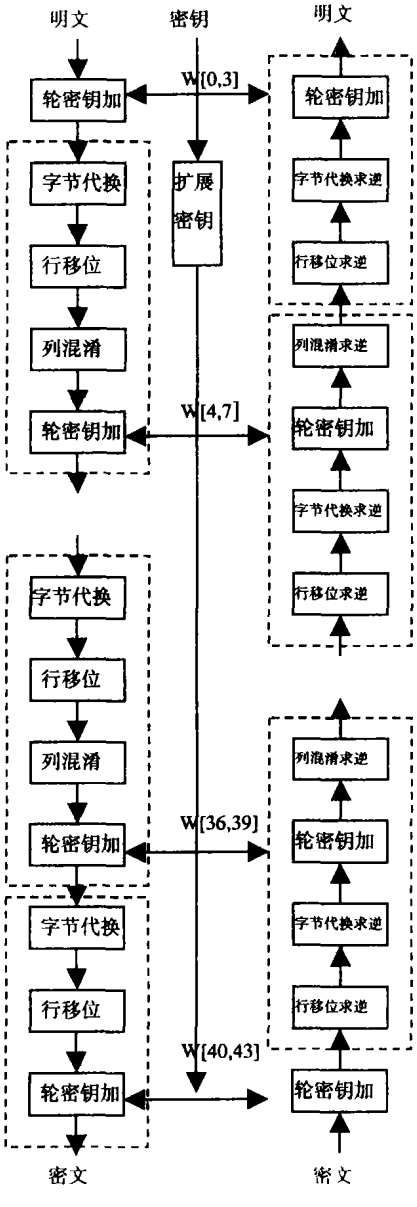

图 1 AES128 工作流图

2 有限域内的运算

所谓有限域就是具有有限个元素的域。集合中元素的个数称为域的阶, m 阶域存在当且仅当 m 是某素数的幂, 即存在某个整数 n 和素数 p, 使得 $m = p^n$。p 称为有限域的特征。通常记 x 阶有限域为 GF(x)。

AES 使用有限域 GF(2^8) 上的加法、乘法运算。在文中, 对于有限域的加法、乘法, 分别用符号“ ⊕” 和“·” 表示。使用的既约多项式为 $m(x) = x^8 + x^4 + x^3 + x + 1$。

(1) 加法。

加法操作就是两个多项式的系数在 GF(2^8) 上相加。如果将 GF(2^8) 中的元素看成是 8 位的串, 那么加法就等价于按位异或操作。其实现电路是异或门。若用硬件描述语言(如 VHDL) 实现有限域加法逻辑仅需一条语句, 例如: $c = a$ xor b

举例: $f(x) = x^6 + x^4 + x^2 + x + 1$, $g(x) = x^7 + x + 1$, 则

$f(x) \oplus g(x) = x^6 + x^4 + x^2 + x + 1 + x^7 + x + 1 = x^7 + x^6 + x^4 + x^2$

(2) 乘法。

乘法操作和一般多项式乘法差不多, 但系数是在 GF(2^8) 上相乘, 并且结果多项式需要模多项式 $m(x)$。仍以上面的 $f(x)$, $g(x)$ 为例,

$f(x) \times g(x) = x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$

$f(x) \cdot g(x) = f(x) \times g(x) \bmod m(x) = x^7 + x^6 + 1$

(3) 基于乘法的简化。

简单的异或运算不能完成 GF(2^8) 的乘法, 但是可以通过使用一些技巧达到简化运算的目的, 考虑 GF(2^8) 上的多项式

$f(x) = b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$, 将它乘上 x, 可得:

$$x \cdot f(x) = (b_7x^8 + b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x) \bmod m(x)$$
$$= b_6x^7 + b_5x^6 + b_4x^5 + (b_3 \oplus b_7)x^4 + (b_2 \oplus b_7)x^3 + b_1x^2 + (b_0 \oplus b_7)x + b_7$$

观察此式, 如果 $x_7 = 0$, 那么结果是一个次数小于 8 的多项式, 不需进一步计算。如果 $b_7 = 1$, 那么可以进行除 $m(x)$ 取余运算:

$$x \cdot f(x) = (b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x) + (x^4 + x^3 + x + 1)$$

这表明 $f(x)$ 乘以 x (如 00000010) 的运算可以通过左移一位后按位异或 00011011(可用多项式表示为 $x^4 + x^3 + x + 1$) 来实现。总结如下:

$$x \cdot f(x) = \begin{cases} b_6b_5b_4b_3b_2b_1b_00 & b_7 = 0 \\ (b_6b_5b_4b_3b_2b_1b_00) \oplus 00011011 & b_7 = 1 \end{cases} \quad (1)$$

乘以一个高于一次的多项式可以重复使用上式来实现, 举例如下:

验证计算式 $f(x) \times g(x) \bmod m(x) = x^7 + x^6 + 1$ 成立, 即在有限域 GF(2^8) 上计算表达式 (01010111) × (10000011)。根据公式(1), 先计算出 x 幂乘的中间结果。

(01010111) × (00000010) = (10101110)

(01010111) × (00000100) = (01011100) ⊕

(00011011) = (01000111)

(01010111) × (00001000) = (10001110)

(01010111) × (00010000) = (00011100) ⊕ (00011011) = (00000111)

(01010111) × (00100000) = (00001110)

(01010111) × (01000000) = (00011100)

(01010111) × (10000000) = (00111000)

所以,

(01010111) × (10000011) = (01010111) × [(00000001) ⊕(00000010) ⊕(10000000)] = (01010111) ⊕(10101110) ⊕(00111000) = (11000001)

计算结果(11000001)等价于多项式 $x^7 + x^6 + 1$。验证完毕。

3　AES 中有限域运算的优化及 FPGA 设计

AES 算法的 4 个阶段中, 列混淆是一个基于 GF(2^8) 代数运算的数据代换过程。由于运算中的系数是常数, 所以可以通过对运算的分解做出优化:

$$b_{0,j} = 02ga_{0,j} \oplus 03ga_{1,j} \oplus 01ga_{2,j} \oplus 01ga_{3,j}$$
$$= 02ga_{0,j} \oplus (02 \oplus 01)ga_{1,j} \oplus 01ga_{2,j} \oplus 01ga_{3,j}$$
$$= 02g(a_{0,j} \oplus a_{1,j}) \oplus 01ga_{1,j} \oplus 01ga_{2,j} \oplus 01ga_{3,j}$$

记有限域运算 $02ga$ 为 $X2$, $T = a_{0,j} \oplus a_{1,j} \oplus a_{3,j}$, $U = a_{i,j} \oplus a_{i+1,j}$, 上式化简为

$$b_{0,j} = X2(a_{0,j} \oplus a_{1,j}) \oplus a_{0,j} \oplus T$$

整个运算中只包含一个有限域运算 $X2$ 模块和若干个异或运算。具体运算电路设计如图 2、图 3 所示, 它有效节约了设计资源。

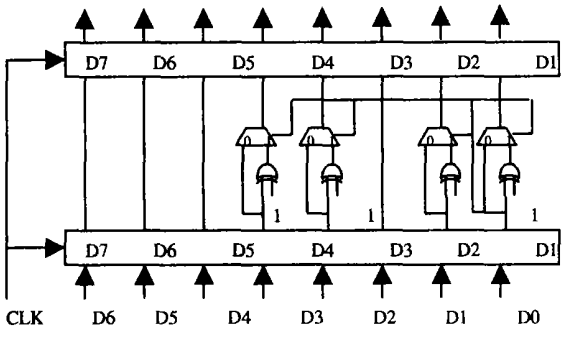

图 2　X2 乘法单元电路实现

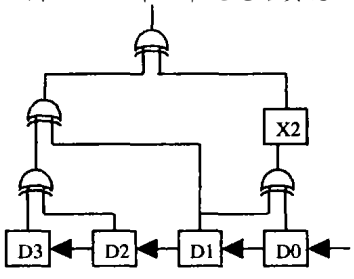

图 3　AES 中有限域乘法运算电路

在 FPGA 设计中, 还可以使用查表的方法来取代乘法操作, 代价是要花费一定的存储空间。定义一个包含 256 字节的表 $X2$, 实现 8 位输入 8 位输出的线性变换, 使得 $X2[i] = 02gi$, 由于该运算频繁使用, 故采用 FPGA 中的查找表(Look- Up- Table)完成。目前 FPGA 中多使用 4 输入的 LUT, 所以每一个 LUT 可以看成一个有 4 位地址线的 16x1 的 RAM。使用 8 个 LUT 即可完成了 8- 8 变换。将输入的 8 位数据分为高 4 位和低 4 位, 分别作为 4 个 LUT 组成的 RAM 中查找地址, 再将输出的 2 个 4 位数据合并, 完成了 8- 8 的数据置换操作(如图 4 所示)。

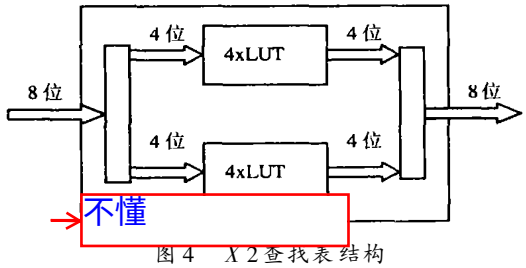

图 4　X2 查找表结构

AES 的逆运算结构与加密相似, 但逆列混淆运算使用的系数表不同, 额外定义一个查询表则浪费资源, 通过对算法分析, 在列混淆多项式 $c(x)$ 与逆列混淆多项式 $d(x)$ 满足关系 $d(x) = (04x^2 + 05) \cdot c(x) \cdot (\mathrm{mod}\, x^4 + 01)$, 因而, 逆列混淆可使用一个简单的预处理过程加上一个列混淆运算来实现。预处理运算公式推导如下:

$$a_0 = 5 \cdot a_0 \oplus 4 \cdot a_2$$
$$= 2 \cdot (2 \cdot a_0) \oplus 1 \cdot a_0 \oplus 2 \cdot (2 \cdot a_2)$$
$$= 2 \cdot (2 \cdot (a_0 \oplus a_2)) \oplus a_0$$
$$a_1 = 2 \cdot (2 \cdot (a_1 \oplus a_3)) \oplus a_1$$
$$a_2 = 2 \cdot (2 \cdot (a_0 \oplus a_2)) \oplus a_2$$
$$a_3 = 2 \cdot (2 \cdot (a_1 \oplus a_3)) \oplus a_3$$

根据公式, 设计如图 5 所示的电路。

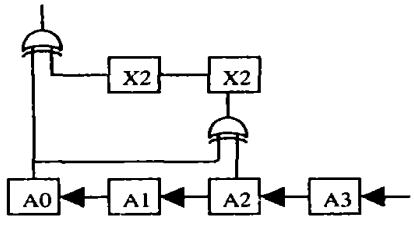

图 5　逆列混淆运算时预处理电路

如果设计资源紧张, 可以使用该处理电路来节省资源, 但这样做使得逆列混淆处理时间等于一个列混淆运算加上与处理的时间, 故降低了速度, 即用 FPGA 设计中的"用速度换面积"。而在资源充分, 对速度要求较高的设计中, 可以采用额外定义一个查找表的方式, 来加快处理速度, 即设计方法中常用的"用面积换速度"。

4　结　论

详细分析了有限域代数在高级加密算法 AES 算法中

(下转第 21 页)

言权。

(3)符合商品生产的规律。商品的开发、生产、销售应由市场规律及运作机制来进行指导和调控,因此 COTS 产品也应从传统软件产品的生产模式中走出来,让市场在该类产品的开发研制中起着积极的作用。而分类 AHP 模型权值的确定则避免了单纯地让开发机构来确定需求优先级所造成的与市场需求之间的脱节,强调了 COTS 产品中市场的特性。

改进的 AHP 模型也面临着一些问题:

* 需求调查问卷的设计。由于问卷是构造成对比较矩阵的原始数据源,因此问卷的形式、内容以及样本的选择都需要一些科学的方法来指导。

* 各 AHP 模型权值的确定。由于分类后的 AHP 模型较多,因此在确定各模型的权值之前应做到目的明确。COTS 作为一种商品,其生产和销售还涉及非软件领域所能起作用的方面(如销售渠道,营销策略等),因此各模型权值的确定也需要科学的方法来指导。

* 后续工作的进行。在对需求优先级有了定量的分析结果之后,应依据机构的目标开展后续的研发工作,如在有限资源的约束下如何最大化满足需求;如何应对边际成本趋于无限增大的问题;如何依据市场的现有特点制定相应的开发战略等[10]。

4 结束语

需求开发是软件开发过程中的一项重要工作,面对软件产品中的众多需求,在有限的资源限制条件下全部满足是不切实际的,对需求的优先级进行设定,从而最大限度的满足诸多风险承担人的利益是一个较好的选择。对于 COTS 产品而言,其需求优先级的设定又面临着客户不确定性及商品生产的客观规律,因此无法简单套用传统的优先级设定方法。文中利用在复杂问题的决策领域有着较好应用的 AHP 模型并对其进行了相应的改进,并将其运用到了 COTS 产品需求优先级的确定中。

参考文献:

[1] Gottesdiener E. Requirements by Collaboration: getting it right the first time[J]. IEEE Software, 2003, 20(2): 52-55.

[2] Wiegers K. Habits of Effective Analysts Software Development Magazine, 2000, 8(10): 62-65.

[3] Rupp C. Requirements and Psychology[J]. IEEE Software, 2002, 19(3): 16-18.

[4] Nuseibeh B, Easterbrook S. Requirements Engineering: A Roadmap[A]. Proceedings of International Conference on Software Engineering(ICSE-2000)[C]. Limerick, Ireland: ACM Press, 2000.

[5] Wiegers K. First Things First: Prioritizing Requirements[J]. Software Development Magazine, 1999, 7(10): 24-30.

[6] Karlsson J, Ryan K. A Cost-Value Approach for Prioritizing Requirements[J]. IEEE Software, 1997, 14(5): 67-74.

[7] Jung Ho-Won. Optimizing Value and Cost in Requirements Analysis[J]. IEEE Software, 1998, 15(4): 74-78.

[8] IEEE. IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications[Z]. Los Alamitos, CA: IEEE Computer Society Press, 1998.

[9] 姜启源. 数学模型(第 2 版)[M]. 北京: 高等教育出版社, 1993. 305-335.

[10] Messerschmitt D G, Szyperski C, Marketplace Issues in Software Planning and Design[J]. IEEE Software, 2004, 21(3): 62-70.

(上接第 17 页)

的应用,基于算法本身的结构特点对计算做出优化,从理论上推导优化计算过程,并用 FPGA 实现了基于有限域 $GF(2^8)$运算的高速电路。在 Xilinx 的 ISE5.2 集成的环境下综合,测试结果为,非反馈模式下系统最高工作频率为 67MHz,系统最大数据流量 779MB/s,具有较好的性能比。表 1 中给出了与国内外部分相关测试数据的比较。

表 1　实验数据比较

方案	器件	使用资源 Slice/BlockRam	数据流量 (MBps)	性能比 MBps/Slice
文献[6]	XCV100E	854/1340	1523	1.7
文中设计	XCV300E	881/18	779	0.88
文献[7]	XCV200E	235/2	180	0.76
文献[8]	XCV300E	660/20	298	0.45

参考文献:

[1] 陈　丹. Rijndael 算法的研究[J]. 计算机应用与软件, 2004(2): 79-81.

[2] Stallings W. 密码编码学与网络安全: 原理与实践(第 3 版)[M]. 刘玉珍, 等译. 北京: 电子工业出版社, 2004.

[3] Daemen J, Rijmen V. AES proposal: Rijndael V2[S]. 1999.

[4] von Eilert B, Würfel H, Riesener O, et al. Implementierung des Advanced Encryption Standards in Hardware, Elektronik[EB/OL]. http://www.elektroniknet.de. 2003-08.

[5] Gaj K, Chodowiec P. Fast implementation and fair comparison of the final candidates for advanced encryption standard using field programmable gate arrays[A]. Proc RSA Security Conference-Cryptographer's Track[C]. San Francisco, CA: [s.n.], 2001.

[6] Bremen University of Applied Sciences. AES Crypto Core Family-Overview[EB/OL]. http://www.i3m.hs-bremen.de. 2003-07.

[7] CAST INC. CAST AES Xilinx Core Datasheet[S]. 2003.

[8] 佟玉伟, 陆浪如, 李　明, 等. 基于 FPGA 先进加密算法(AES)的并行实现[J]. 交通与计算机, 2002(6): 20-22.