

一种 AES 算法中 S 盒和逆 S 盒替换的表达式方法

覃晓草, 李树国

(清华大学 微电子学研究所, 北京 100084)

摘 要: S 盒替换与逆 S 盒替换是 AES 算法性能的主要瓶颈, 它直接影响 AES 芯片的运算速度. 在优化 Q-M 化简法基础上, 提出了一种实现 AES 算法中 S 盒替换和逆 S 盒替换的表达式方法, 这种表达式方法相比于普遍使用的查表法, 其延时减小了 8.5%, 面积减小了 27.4%, 功耗减小了 17%.

关键词: AES 算法; S 盒替换; 逆 S 盒替换; Q-M 化简法; 表达式方法

中图分类号: TP309

文献标识码: A

文章编号: 1000-7180(2014)01-0112-04

An Expression Method to Implement S-Box and Inverse S-Box Substitution for AES Algorithm

QIN Xiao-cai, LI Shu-guo

(Institute of Microelectronics, Tsinghua University, Beijing 100084, China)

Abstract: S-Box substitution and inverse S-Box substitution are the major bottleneck for AES algorithm, which directly affect the speed of AES chip. On the basis of optimizing Q-M simplifying method, this paper proposes an expression method to implement the S-Box substitution and inverse S-Box substitution in AES algorithm. Compared with the widespread use of the look-up-table method, the expression method can reduce the delay, area and power of circuit by 8.5%, 27.4% and 17% respectively.

Key words: AES algorithm; S-Box substitution; inverse S-Box substitution; Q-M simplifying method; expression method

1 引言

AES 算法是一种对称分组密码算法^[1], 其中 S 盒替换与逆 S 盒替换是 AES 算法中唯一的非线性变换^[2], 因此 S 盒替换与逆 S 盒替换是 AES 算法的核心部分^[3]. 同时也是提高 AES 算法性能的主要瓶颈.

现有的 S 盒替换与逆 S 盒替换普遍采用的是查表法实现. 所谓查表法中的“表”指的是 AES 算法中的两个盒子, 分别用于加密时的 S 盒替换与解密时的逆 S 盒替换^[1]. 当前, 普遍的查表法不管是 S 盒替换还是逆 S 盒替换, 都是根据输入值通过 AES 算法标准规定的表格来查出对应的输出值. 这种查表法原理简单, 实现容易, 但是这种方法实际上是一个大

的 256 选一多路选择器, 其电路的延时较长, 面积较大^[4], 是一种低效的实现方式.

本文在优化 Q-M 化简法^[5]基础上, 提出一种实现 AES 算法中 S 盒替换和逆 S 盒替换的表达式方法. 与查表法相比, 这种实现方法减小了延时和电路面积, 并且降低了电路功耗, 使得电路性能得到了提高.

2 Q-M 化简法原理及其改进

2.1 Q-M 化简法原理

Q-M 化简法是对逻辑函数进行化简的方法, 与卡诺图化简法的基本思想大致相同. 不同之处在于, 手工进行卡诺图化简仅适用于变量个数较少的情况下, 而 Q-M 化简法更适用于计算机分析和处

收稿日期: 2013-02-20; 修回日期: 2013-04-25

基金项目: 国家“八六三”计划项目(2012AA012402); 国家自然科学基金(61073173); 清华大学自主研发计划(2011Z05116)

理化简变量个数较多的逻辑函数^[5].

理解 Q-M 化简法之前需要介绍下面几个定义^[6]:

蕴涵项:在函数的“与或”表达式中,每个“与”项被称为该函数的蕴涵项.例如,最小项是蕴涵项,由 2ⁿ个最小项合并得到的“与”项也是蕴涵项.也就是说,在函数卡诺图中,任何一个方格所对应的最小项或者卡诺图中的 2ⁿ个方格所对应的“与”项都是函数的蕴涵项.

质蕴涵项:若函数的一个蕴涵项不是该函数中其他蕴涵项的子集,则此蕴涵项称为质蕴涵项.显然,在函数卡诺图中,按照最小项合并规律,如果某个卡诺圈不可能被其他更大的卡诺圈包含,那么该卡诺圈所对应的“与”项为质蕴涵项.

最小覆盖:包含函数全部最小项的最小质蕴涵项集合.

Q-M 化简法的化简步骤一般可以粗略概括为以下四步^[6]:

第一步:将函数表示成最小项表达式,并用二进制码表示每一个最小项.

第二步:找出函数的全部质蕴涵项.寻找函数全部质蕴涵项的方法是先将 n 个变量函数中的相邻最小项合并,消去相异的一个变量,得到 $(n-1)$ 个变量的“与”项.这时如果存在不能合并的最小项,它便是所寻找的部分质蕴涵项.再将相邻的 $(n-1)$ 个变量的“与”项合并,消去相异的变量,得到 $(n-2)$ 个变量的“与”项,这时如果存在不能合并的 $(n-1)$ 个变量的“与”项,它也是所寻找的部分质蕴涵项.如此进行下去,直到不能再合并为止.这时,所得到的全部不能再合并的“与”项,即为所要求的全部质蕴涵项.

第三步:找出函数的必要质蕴涵项.方法是先画出质蕴涵表,在表上找出只属于一个质蕴涵项的最小项,则包含该最小项的质蕴涵项即为必要质蕴涵项.

第四步:找出函数的最小覆盖.方法是多次使用行消去与列消去法,具体过程可以参见文献^[5].

2.2 改进的 Q-M 化简法

通过使用 Q-M 化简法来化简 S 盒替换与逆 S 盒替换的函数表达式,发现所得的函数表达式并不是最简的,仍有可以合并的项.因此,在 Q-M 化简法和文献^[7]的基础上,提出改进的 Q-M 化简法,可以得到“与”项项数更少的函数表达式.

改进的 Q-M 化简法的详细化简步骤概括为以下七步:

第一步:将函数表示成最小项表达式;

第二步:找出函数的全部质蕴涵项;

第三步:找出必要质蕴涵项,输出结果;

第四步:去除必要质蕴涵项以及各个必要质蕴涵项所对应的最小项后,得到一个新的表;

第五步:在新的表中找到包含最小项个数最多的那个质蕴涵项(若是有多项质蕴涵项包含最小项个数最多,则任选其一),输出这个质蕴涵项.

第六步:去除第五步输出的这个质蕴涵项以及该质蕴涵项所对应的最小项后,得到一个新的表;

第七步:重复第五步和第六步,直到没有能够再合并的最小项为止,循环结束.

3 用 Q-M 法产生 AES 算法中实现 S 盒和逆 S 盒替换的表达式

AES 算法中 S 盒和逆 S 盒的输入变量是 8 位,同时输出变量也是 8 位.根据 AES 算法标准中的 S 盒和逆 S 盒,写出输出变量的真值表.每个真值表对应一位输出变量,因此 S 盒的 8 位输出可得到 8 个真值表,同样,逆 S 盒的 8 位输出也得到 8 个真值表.先将真值表表示为一个最小项表达式,再使用上面提到的改进的 Q-M 化简法,对最小项表达式进行逻辑化简,得到化简后的函数表达式.因此, S 盒替换得到 8 个 S-Box 函数表达式,同样,逆 S 盒替换得到 8 个 INV-S-Box 函数表达式.依据加密或解密的不同模式,选择是输出 S-Box 函数表达式还是输出 INV-S-Box 函数表达式.

由于 S 盒与逆 S 盒的输入变量有 8 个,通过手工进行逻辑化简获取 S 盒替换和逆 S 盒替换的表达式几乎是做不到的,因此用 C 语言编程实现了改进的 Q-M 化简法,借助于计算机求得函数表达式的逻辑化简结果.以 S 盒替换的 S-Box 函数表达式 SBox[7;0]为例,设输入变量为 a ,其逻辑表达式如下:

$$\begin{aligned} \text{SBox}[7] = & \sim((\sim(\sim a[7] \& \sim a[6] \& \sim a[5] \& \sim a[4] \& a[3] \& a[2] \& \sim a[0])) \& (\sim(a[6] \& \sim a[5] \& a[4] \& \sim a[3] \& a[2] \& a[1] \& \sim a[0])) \& \dots \& (\sim(\sim a[7] \& a[6] \& a[5] \& a[4] \& \sim a[1] \& a[0])))) \end{aligned}$$

$$\begin{aligned} \text{SBox}[6] = & \sim((\sim(a[7] \& a[6] \& \sim a[5] \& \sim a[4] \& \sim a[3] \& a[2] \& a[1] \& a[0])) \& (\sim(\sim a[7] \& \sim a[5] \& \sim a[4] \& \sim a[3] \& a[2] \& a[1] \& \sim a[0])) \& \dots \& (\sim(a[7] \& \sim a[6] \& a[5] \& a[4] \& \sim a[1] \& a[0])))) \end{aligned}$$

⋮

$$\text{SBox}[1] = \sim((\sim(a[7] \& a[6] \& \sim a[5] \& \sim a[4] \& \sim a[3] \& \sim a[2] \& \sim a[1] \& \sim a[0])) \& (\sim(a[6] \& \sim a[5] \& a[4] \& a[3] \& \sim a[2] \& a[1] \& \sim a[0])) \& \dots \& (\sim(a[7] \& a[6] \& a[3] \& \sim a[2] \& a[1] \& \sim a[0]))))$$

$$\text{SBox}[0] = \sim((\sim(a[7] \& \sim a[6] \& \sim a[5] \& a[4] \& \sim a[3] \& \sim a[2] \& \sim a[1] \& a[0])) \& (\sim(\sim a[7] \& \sim a[6] \& \sim a[5] \& \sim a[3] \& a[2] \& \sim a[1] \& a[0])) \& \dots \& (\sim(a[7] \& a[6] \& a[5] \& a[3] \& \sim a[2] \& \sim a[0]))))$$

在上述表达式中,采用了“与非与非”即为“与或”的两次取反的形式,而不是单纯的“与或”的形式.这是基于“与非”门的延时要小于“与”门延时的事实,实验结果也验证了这一点.因此,本文的函数表达式都采用了“与非与非”的形式.

依据同样的方法,化简得到 16 个所有 S 盒替换的 S-Box 表达式与逆 S 盒替换的 INV-S-Box 表达式,由于每个表达式过长,限于篇幅,仅列出各个表达式的“与非”项项数如表 1 所示(表达式中形如 $\sim(a[7] \& \sim a[6] \& \sim a[5] \& \sim a[4] \& a[3] \& a[2] \& \sim a[0])$ 的项为一个“与非”项).

表 1 S-Box 函数表达式与 INV-S-Box 函数表达式各表达式的“与非”项项数

输出变量第 n 位	S-Box“与非”项项数	INV-S-Box“与非”项项数
7	51	45
6	50	53
5	52	50
4	47	53
3	51	49
2	47	52
1	51	46
0	46	48

通过研究上述这 16 个表达式,发现这些函数表达式之间含有很多可以公用的逻辑电路单元.因此,可以将这些能够公用的逻辑电路单元单独提出来,使得这些函数表达式实现电路资源公用.下面是单独提出来的公用逻辑电路单元,其中, a 为 S 盒与逆 S 盒的输入变量,变量 c, d, e, f, g, h 为中间变量.

$$c[0] = \sim a[1] \& \sim a[0],$$

$$c[1] = \sim a[1] \& a[0],$$

$$c[2] = a[1] \& \sim a[0],$$

$$c[3] = a[1] \& a[0],$$

$$d[0] = \sim a[3] \& \sim a[2],$$

$$d[1] = \sim a[3] \& a[2],$$

$$d[2] = a[3] \& \sim a[2],$$

$$d[3] = a[3] \& a[2],$$

$$e[0] = \sim a[5] \& \sim a[4],$$

$$e[1] = \sim a[5] \& a[4],$$

$$e[2] = a[5] \& \sim a[4],$$

$$e[3] = a[5] \& a[4],$$

$$f[0] = \sim a[7] \& \sim a[6],$$

$$f[1] = \sim a[7] \& a[6],$$

$$f[2] = a[7] \& \sim a[6],$$

$$f[3] = a[7] \& a[6], g[0] = d[0] \& c[0],$$

$$g[1] = d[0] \& c[1], g[2] = d[0] \& c[2],$$

$$g[3] = d[0] \& c[3], g[4] = d[1] \& c[0],$$

$$g[5] = d[1] \& c[1], g[6] = d[1] \& c[2],$$

$$g[7] = d[1] \& c[3], g[8] = d[2] \& c[0],$$

$$g[9] = d[2] \& c[1], g[10] = d[2] \& c[2],$$

$$g[11] = d[2] \& c[3], g[12] = d[3] \& c[0],$$

$$g[13] = d[3] \& c[1], g[14] = d[3] \& c[2],$$

$$g[15] = d[3] \& c[3], h[0] = f[0] \& e[0],$$

$$h[1] = f[0] \& e[1], h[2] = f[0] \& e[2],$$

$$h[3] = f[0] \& e[3], h[4] = f[1] \& e[0],$$

$$h[5] = f[1] \& e[1], h[6] = f[1] \& e[2],$$

$$h[7] = f[1] \& e[3], h[8] = f[2] \& e[0],$$

$$h[9] = f[2] \& e[1], h[10] = f[2] \& e[2],$$

$$h[11] = f[2] \& e[3], h[12] = f[3] \& e[0],$$

$$h[13] = f[3] \& e[1], h[14] = f[3] \& e[2],$$

$$h[15] = f[3] \& e[3].$$

基于这些公用单元,对 S 盒替换的 S-Box 函数表达式 $\text{SBox}[7:0]$ 进一步化简,得到其表达式如下,

$$\text{SBox}[7] = \sim((\sim(h[0] \& d[3] \& \sim a[0])) \& (\sim(a[6] \& e[1] \& g[6])) \& \dots \& (\sim(h[7] \& c[1])))$$

$$\text{SBox}[6] = \sim((\sim(h[12] \& g[7])) \& (\sim(a[7] \& e[0] \& g[6])) \& \dots \& (\sim(h[11] \& c[1])))$$

⋮

$$\text{SBox}[1] = \sim((\sim(h[12] \& g[0])) \& (\sim(a[6] \& e[1] \& g[10])) \& \dots \& (\sim(f[1] \& g[10])))$$

$$\text{SBox}[0] = \sim((\sim(h[9] \& g[1])) \& (\sim(f[0] \& \sim a[5] \& g[5])) \& \dots \& (\sim(f[3] \& a[5] \& d[2] \& \sim a[0])))$$

上述公用单元提取后的表达式与提取前的表达式相比,公用单元提取后的表达式对应的电路面积变小,扇出变少,从而使得延时减小。

用上述改进 Q-M 化简法得到 S 盒替换的 8 个函数表达式和逆 S 盒替换的 8 个函数表达式,分别用于加密和解密模式,电路结构如图 1 所示。S 盒替换与逆 S 盒替换的输入变量为 a , 输出变量为 b 。通过 MUX 二选一的选择信号 Encrypt 进行选择, Encrypt 为 1 时,即为加密模式,输出变量 b 选择输出 S-Box 函数表达式,而 Encrypt 为 0 时,即为解密模式,输出变量 b 选择输出 INV-S-Box 函数表达式。

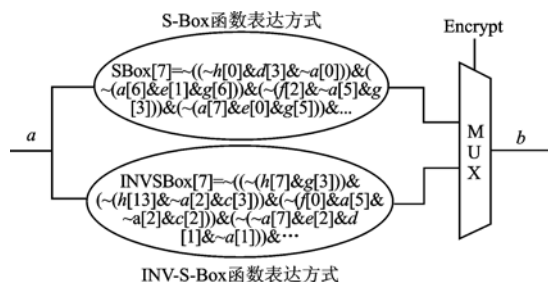


图 1 改进的 Q-M 化简法实现 S 盒替换与逆 S 盒替换的电路结构

4 验证、比较

对于表达式结果的功能验证,将 8 位输入变量 a 由 00000000 遍历到 11111111 的这 256 个值进行了功能性完全验证,结果与 AES 算法标准的 S 盒和逆 S 盒完全一致。由此说明本文得到化简后的表达式是正确的。

基于 SMIC 0.18 μm 工艺对改进 Q-M 化简法得到的实现 S 盒替换与逆 S 盒替换的 16 个函数表达式进行综合,结果如表 2 所示。同时为了进行对比,也对查表法进行了综合,在表 2 中给出了结果。

表 2 改进后的表达式法与查表法实现的对比

性能指标	查表法	改进后的表达式法
面积(等效门数)	2 646 门	1 922 门
功耗	2.535 5 mW	2.113 mW
延时	1.77 ns	1.62 ns

由表 2 可知,改进后的表达式法实现了 S 盒替换与逆 S 盒替换。相比于查表法,延时减小了 8.5%,面积减小了 27.4%,功耗减小了 17%。

5 结束语

本文在优化 Q-M 化简法的基础上,提出了一种实现 AES 算法中 S 盒替换和逆 S 盒替换的表达式方法,功能验证表明此方法具有很好的改进性。同时,本文提出的改进后的表达式化简方法也可用于其它的数字逻辑化简,如 7-3 压缩、15-4 压缩、大数乘法器的阵列压缩等应用都具有重要的借鉴意义。

参考文献:

- [1] National Institute of Standards and Technology (NIST). FIPS PUB 197. Advanced Encryption Standard (AES) [S]. America: Federal Information Processing Standards Publications, 2001.
- [2] 胡春燕, 易波. 基于 AES 加密算法的 S 盒优化设计[J]. 微计算机信息, 2012, 28(10): 358-360.
- [3] 韩少男, 李晓江. 实现 AES 算法中 S-BOX 和 INV-S-BOX 的高效方法[J]. 微电子学, 2010, 40 (1): 103-107.
- [4] 高磊, 戴冠中. AES 算法中 SubBytes 变换的高速硬件实现[J]. 微电子学与计算机, 2006, 23(7): 47-49.
- [5] 张冰. 逻辑函数的另一种化简方法—Q-M 化简法[J]. 新疆教育学院学报, 2003, 19(2): 85-88.
- [6] 刘明亮, 饶敏. 数字逻辑基础[M]. 北京: 北京航空航天大学出版社, 2000.
- [7] 徐俊平, 程利新. 改进的 Q-M 逻辑函数化简方法[J]. 计算机工程, 2011, 37 (20): 30-32.

作者简介:

覃晓草 女, (1988—), 硕士研究生. 研究方向为信息安全算法的大规模数字集成电路设计与实现。

李树国 男, (1963—), 教授, 博士生导师. 研究方向为信息安全算法的大规模数字集成电路设计与实现。