

一种在智能卡芯片中实现 SHA-1 算法的方法

匙嘉敏, 白国强, 高志强

(清华大学 微电子学研究所, 北京 100084)

摘 要: SHA 是由美国国家安全局(NSA)设计的安全杂凑算法. 该算法主要应用在通讯完整性验证以及数字签名认证领域. 以面积优化为目标, 从系统设计入手到模块级设计, 以具体设计为实例, 在智能卡芯片中以较小的面积代价实现了 SHA-1 算法, 对于类似的杂凑算法设计具有普遍的参考价值.

关键词: 杂凑算法; 智能卡; SHA-1; 面积

中图分类号: TP302.2

文献标识码: A

文章编号: 1000-7180(2010)02-0068-04

SHA-1 Design in Smart Card Chip

CHI Jia-min, BAI Guo-qiang, GAO Zhi-qiang

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

Abstract: The SHA (Secure Hash Algorithm) family is a set of related cryptographic hash functions designed by the National Security Agency (NSA) and published by the National Institute of Standards Technology (NIST). SHA-1 is one of them. The main applications of the SHA-1 function are met in the fields of communication integrity and signature authentication. Targeting the minimal area, an implementation of SHA-1 Algorithm in smart card chip is introduced. The Method introduced by this paper cost less area. And it can be referenced by the related design about hash algorithm implementation.

Key words: hash algorithm; smart card; SHA-1; area

1 引言

为了提高网络的安全性, 带有安全认证功能的智能卡芯片应运而生. 这种安全芯片中一个必不可少的单元就是 SHA 模块. SHA-1 算法是 SHA 算法家族中应用最广泛的算法. 智能卡芯片通常具有面积小、成本低的特点, 对于高性能的公钥密码算法, 一秒钟内完成几次运算就能满足要求. 因此, 在智能卡芯片中实现 SHA-1 算法应该进行面积的优化, 尽量减小硬件开销. 文中就是针对具体的项目, 介绍在—款智能卡芯片中如何以最小的面积代价实现 SHA-1 算法.

2 SHA-1 算法介绍

SHA 算法(Secure Hash Algorithm)又称为安全

杂凑函数, 它的安全性体现在两个方面:

(1) 找到对应相同输出值的不同输入值在运算上是不可行的, 这就是通常所说的防冲突性;

(2) 由已知的输出值反推出其输入值在运算上是不可行的, 即很难从结果推导出它的初始状态.

SHA-1 算法是 SHA 算法的一种, 其对长度不超过 2^{64} 二进制位的消息产生 160 位的消息摘要输出.

算法实现的步骤如下:

步骤 1: 对输入消息进行填充位操作.

首先填充消息使其二进制位长度恰好为比 512 的整数倍小 64 位的数. 填充方法是附一个 1 在消息后面, 后接所要求的多个 0, 然后在其后附上 64 位的消息长度(填充前的长度), 使填充后的消息长度恰好是 512 位的整数倍.

步骤 2: 将填充后的消息, 分成 N 个 512 位的

块.用 $M_0, M_1, M_2, \dots, M_N$ 来表示 N 个消息块.

步骤3:主循环操作

对于总长为 $N \times 512$ 位的消息块,进行 N 次主循环操作.每个主循环操作 512 位数.每次的操作步骤如下所示.

(1) 初始化 hash 变量

第一轮循环用 5 个 32 位的固定常量初始化.

以后每轮循环用上一轮计算的 hash 运算的结果值进行初始化.

(2) hash 算法操作

① 80 个 32 位临时值 W_i (W_0 至 W_{79}) 的生成.

② 对 32 位临时值 W_i 和 hash 变量初始值的循环处理生成本次运算 160 位的消息摘要.

步骤4:输出最后一次循环产生的 160 位消息摘要.

主循环操作分为四轮,每轮 20 次操作.每一次主循环输出的 A, B, C, D, E 值作为下一个主循环操作的输入值.最后一次输出的 A, B, C, D, E 值与本次的输入值相加后的输出是本次 SHA-1 运算的结果值.输出的 160 位消息摘要由 A, B, C, D, E 级联而成.

3 智能卡芯片设计实例

智能卡芯片如图 1 所示.

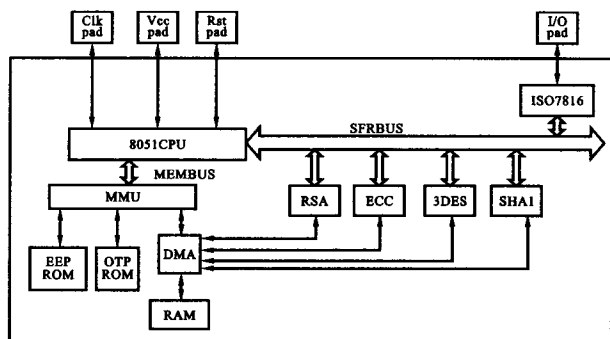


图 1 智能卡芯片结构图

其中, CPU 是一款兼容 8051 指令集的 8 位 CPU. CPU 有两套总线, 一套存储器总线 (MEM)、一套 SFR 总线.

MEM 总线上挂有 EEPROM、OTPROM 等程序存储器和 RAM 数据存储器. 通过 MMU 对这些 MEM 总线上的存储器进行管理. MMU 的控制寄存器由 CPU 通过 SFR 总线来设置. SFR 总线上挂有多个加密协处理器和串口控制电路, 另外, 各加密协处理器也通过 SFR 总线控制 DMA 模块的行为.

OTP ROM 中存放 COS (Card Operating System). 利用测试电路将 COS 下载到 OTP ROM 中.

COS 解析来自 ISO 7816 串口的外部命令与数据, 调用各相关模块对外部数据进行相应处理. COS 在 EEPROM 中建立文件系统, 将处理后的数据保存在 EEPROM 中, 在需要将结果送出时, 从文件系统中读取数据, 并通过 ISO 7816 串口送出.

在加密协处理器进行运算的过程中要在 RAM 区中频繁地读取数据. 明文数据输入、密钥的输入、加解密结果的输出, 以及中间结果的数据存取都需要访问 RAM 区. 为了提高数据搬移的效率, 系统中设计了 DMA 模块, 为 RAM 区和加密协处理器之间提供了直接通路. 并且在 RAM 区中划出一部分空间作为加密协处理器和 CPU 共享的空间.

利用加密协处理器进行运算的过程可以简述如下: CPU 将要加密的数据写入指定的 RAM 区, 启动加密协处理器后, 加密协处理器直接到相应的 RAM 区取出数据进行运算, 运算的结果写回相应的 RAM 区, 供 CPU 读取.

4 SHA-1 协处理器的设计

根据 SHA-1 算法的运算流程, SHA-1 算法协处理器按功能可分为 6 个子模块: 补位单元、常量数组、寄存器堆、主循环运算、控制单元和消息摘要输出模块. 其中, 补位单元完成输入信息的补位和分块操作; 常量数组模块提供常量值 K_i 和初始化值 $H_0 \sim H_4$, 同时也包括 W_i 产生电路; 寄存器堆用来存储 SHA-1 主循环运算需要的 80 个 W_i 值, 根据步数计数器的值输出相应的 W_i 值; 主循环运算模块进行 4 轮杂凑运算, 每轮 20 次, 共 80 次循环运算; 消息摘要输出完成主循环输出值与初始值的加法运算, 输出结果作为本次杂凑运算的最终结果; 控制模块用来产生控制信号控制其他 5 个模块的运行. 协处理器中各子模块之间的相互关系可以用图 2 进行表示.

再分析 SHA-1 协处理器实现的功能, 可将运算过程分为两大部分: 输入数据的预处理和杂凑算法的运算. 输入数据的预处理部分完成输入数据的填充位操作和分块操作, 与整个算法的运算关系不大, 用软件实现比较方便. 另外, 数据的预处理操作用软件实现可以简化整个硬件协处理器的功能, 节约面积, 同时又不破坏硬件协处理器的安全性. 因此, 本设计的补位单元使用软件实现.

常量数组产生常量值 K_i 和初始化值 $H_0 \sim H_4$, 这些可以用 6 个常量寄存器来实现. 另外, 该模块和寄存器堆模块一起需要实现 W_i 产生和存储的电路功

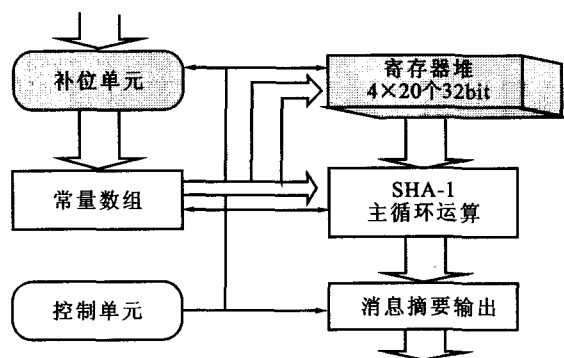
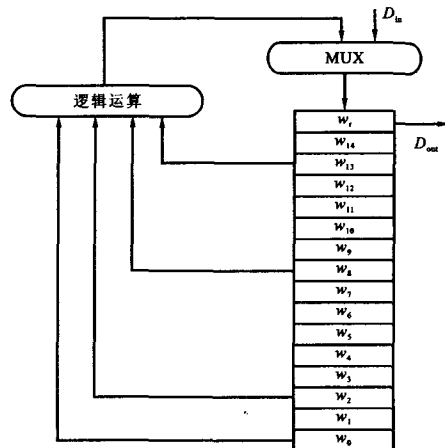


图2 SHA-1 协处理器的顶层结构图

能,电路具体实现的功能可描述为:原始消息字的接收存储和为主循环运算模块提供运算过程中所需的变量 W_t 。每个消息输入块的输入长度是 512 位,而 W_t 进行运算的基本单位为 32 位比特长度的消息字,因此输入消息可划分为 16 个原始消息字。每 512 位消息块的处理需要 80 个扩展字 $W_0, W_1, \dots, W_{79}$ 。其中 $W_0, W_1, \dots, W_{15}$ 是由原始的 512 位信息划分而成,后面的 64 个 W_t 是由 $W_0, W_1, \dots, W_{15}$ 扩展而来的。

扩展公式为 $W_t = S^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16})$ 。因此,需要 80 个 32 位的寄存器来保存 80 个 W_t , 耗费比较多的硬件资源。分析 W_t 的扩展公式,当计算 W_t 时,需要用到 $W_{t-3}, W_{t-8}, W_{t-14}$ 和 W_{t-16} , 并且计算出 W_t 后,不再需要 W_{t-16} 。因此, W_t (当 $t > 15$ 时) 的值只与前面 16 个消息有关。电路结构可以采用 16 个 32 位寄存器保存原始状态字和中间数据,在本电路中它们分别是 $W_0, W_1, \dots, W_{14}, W_t$ 寄存器。 W_t 产生模块结构如图 3 所示。

图3 W_t 产生模块结构原理图

在开始运算的 15 个时钟周期,分别对 $W_0, W_1, \dots, W_{14}$ 这 15 个寄存器用输入的原始状态字进行初始化操作。同时,在开始的 16 个时钟周期,图 3 中的

多路选择器 MUX 选通输入数据 D_{in} (32 位),使 W_t 获得输入的原始状态字作为输出;而在随后的 64 个时钟周期,多路选择器 MUX 选通逻辑运算单元的输, W_t 获得扩展状态字作为输出。

这样,本设计通过时分复用的方式,只用了 16 个 32 位寄存器就分时保存、处理了 80 个消息消息字。节省了 64 个 32 位寄存器的硬件资源。

主循环运算模块是整个 SHA-1 算法的核心部分,它完成 4 轮 80 次迭代运算,每次运算用下式表示:

$$A, B, C, D, E \leftarrow (E + f(t, B, C, D) + S^5(A)) + W_t + K_t, A, S^{30}(B), C, D$$

其中, A, B, C, D, E 表示缓存的 5 个字 (32 位); t 表示步骤数 ($0 \leq t \leq 79$); $f_t(B, C, D)$ 表示步骤 t 的原始逻辑函数; S^k 表示 32bit 参数循环左移 k 位; W_t 表示由当前 512bit 输入分组导出的一个 32bit 字; K_t 表示一个常数,根据轮数使用四个不同的值;符号 $+$ 表示模 232 加法。

上式中的非线性函数 (f_t) 有 3 个 32bit 字输入并产生 1 个 32bit 字输出,每个非线性函数执行一组按位的逻辑操作,即第 n 位的输出是这 3 个输入中第 n 个比特的一个函数。当 $0 \leq t \leq 19$, 函数是一个条件函数:如果 B 则 C , 否则 D 。当 $20 \leq t \leq 39$ 和 $60 \leq t \leq 79$, 函数产生一个检验位。当 $40 \leq t \leq 59$, 如果有两个和三个参数为真,则函数为真^[1]。

根据 SHA-1 算法的运算特点,文中的设计采用每个时钟周期进行一次单步运算的硬件结构,使用 5 个 32 位寄存器存储中间数据。这种实现方法与循环打开的实现方法和并行处理插入寄存器的实现方法比较起来面积更小。根据文献[2],采用并行处理插入寄存器的实现方法,虽然可以提高运算的最大工作速率,但是与文中的实现方法比较,面积要增大 10% 左右。另外,循环打开的设计方案采用每个时钟周期进行两个单步运算的硬件结构^[3-4],虽然可以提高协处理器的运算速度,但是也带来了面积的代价。对于工作频率比较低的智能卡芯片来说,显然本文的设计方案更加有利于面积的优化。这种设计方案, A 变量的更新是电路的关键路径。关键路径上有 4 个 32 位二输入加法器,再加上 W_t 产生模块产生的延时,整个电路的关键路径上的延时完全能够满足系统最大 48MHz 工作频率的要求。

另外,每 512 位信息处理在 80 个时钟节拍内完成。每 20 个时钟周期为一轮,每轮的差异在于函数 $f_t(*)$ 和常数 K_t 的不同。由于每轮操作之间的这种

相似性,可以对其进行复用,将四级计算变为一,这样就实现了四轮逻辑运算用同一迭代结构实现,从而大幅减少逻辑运算面积。

5 验证和性能分析

对 SHA-1 协处理器的仿真过程分为两个阶段。在设计的初期,对杂凑算法模块单独进行仿真验证,根据杂凑算法接口的管脚定义编写激励,进行仿真验证。在系统联调阶段,将 SHA-1 算法模块放在智能卡芯片系统中,根据系统定义编写测试激励,进行仿真验证。SHA-1 协处理器经过各级仿真验证,功能和时序完全正确。

图 4 列出了进行 SHA-1 运算时,当输入值为“61626364 62636465 63646566 64656667 65666768 66676869 6768696a 68696a6b 696a6b6c 6a6b6c6d 6b6c6d6e 6c6d6e6f 6d6e6f70 6e6f7071”时的输出波形图。



图 4 SHA-1 运算输出结果的波形图

图 4 中的 op_result[255:0] 变量是接口模块中的内部信号,它代表 SHA-1 运算的输出结果值。当然,验证过程中还对其他随机产生的明文进行了仿真验证。

本 SHA-1 协处理器使用 Verilog-HDL 进行 ASIC 设计实现,使用 Cadence 公司的 NC-Verilog 进行了功能验证。并且整个设计通过了后仿真验证、FPGA 验证以及流片后的测试。

芯片采用 Smic 0.18 μ m EEPROM 工艺,4 层金属布线,5V/15V 供电。内部标准单元供电为 1.8V。SHA-1 协处理器与其他模块一起作为一个完整的网表进行综合,这样有利于接口电路的优化。

SHA-1 协处理器使用 Synopsys 公司的 Design Compiler 进行了综合,目标库为 SMIC. 18 μ s 标准单元库,门数为 12 500 门(包括 SHA-1 与系统的接口电路)。SHA-1 协处理器完成一次 512 位运算需要 81 个时钟周期,其中 1 个周期做状态转换,80 个周

期做运算。

文献[5]中的 SHA-1 处理器电路为一个流水线的一部份,面积开销为 14 000 门,与文中串行结构的 SHA-1 协处理器相比面积增加了约 10%;而文献[5]中执行一次 512 位运算需要 81 个时钟周期,运算单元的速率与本课题的 SHA-1 运算单元处理速率相同。

6 结束语

文中通过分析 SHA-1 算法的结构特点,以软硬件结合的方式完成了 SHA-1 算法协处理器的功能。在协处理器的硬件设计过程中提出了对面积优化的设计方案,以较少的硬件代价,满足了芯片的系统需求。SHA-1 协处理器通过了集成电路流片后的测试,设计的正确性得到了充分验证。

文中提出的 SHA-1 协处理器的设计实现方法可以在其他以面积优化为目标的设计中借鉴。对于类似的杂凑算法设计具有普遍的参考价值。

参考文献:

- [1] FIPS PUB 180-2. Secure hash standard(SHA-1)[S]. National Institute of Standards and Technology(NIST), 2002.
- [2] Marco Macchetti, Luigi Dadda. Quasi-pipeline hash circuits[C]// Proc of the 17th IEEE Symposium on Computer Arithmetic. Ztaly: Milan, 2005.
- [3] Yu Mingyan, Zhou Tong, Wang Jinxiang, et al. An efficient asic implementation of SHA-1 engine for TPM[C]// 2004 IEEE Asia-Pacific Conference on Circuits and Systems. [S. L], 2004: 873-876.
- [4] 郭文平,刘政林,陈毅成,等.高吞吐率、低能耗的 SHA-1 加密算法的硬件实现[J].微电子学与计算机,2008,25(5):82-85.
- [5] Kang Y K, Kim D W, Kwon T W. An efficient implementation of hash function processor for IPSEC[C]//South Korea, Proc of Third IEEE Asia-Pacific Conference on ASICs. IEEE Computer Society, 2002: 93-96.

作者简介:

匙嘉敏 女,(1973-),硕士.研究方向为密码算法实现。