

安全 SOC 芯片哈希算法多类型复用 IP 核设计

刘 坚^{1,2}, 王绩伟¹, 张文婧²

(1 辽宁大学 物理学院, 辽宁 沈阳 110036; 2 北京宏思电子技术有限责任公司, 北京 100191)

摘 要: 设计了可兼容 SHA-1、SHA-224、SHA-256 三种算法的 IP 核。在深入研究算法, 对三种算法进行改进和重新设计, 加快了运算速度、提高了运算核性能。体现出节省面积, 资源优化, 移植性好, 满足使用不同密码算法用户多层级安全要求和集成度优化的优势。已在多款 SOC 芯片中作为数字签名的 IP 核使用, 支持双复位、中断以及低功耗模式, 运算支持中途读写操作。

关键词: SOC; IP 核; SHA-1/224/256; 高性能; 低功耗

中图分类号: TN402

文献标识码: A

文章编号: 1000-7180(2014)10-0086-04

Design of Secure SOC Multi-type and Multiplexing IP Core of Hash Algorithm

LIU Jian^{1,2}, WANG Ji-wei¹, ZHANG Wen-jing²

(1 College of Physics, Liaoning University, Shenyang 110036, China;

2 Beijing Hongsi Electronic Technology Co. Ltd, Beijing 100191, China)

Abstract: Based on deep studying interrelation and the similarity of algorithm of SHA-1, SHA-224 and SHA-256, an IP core containing these three kinds of algorithms was developed and redesigned. It speeds up the operation and improves the performance, in particular of saving area, resource optimization and good portability. It meets multi-level security and integration optimization requirements of various clients, supporting double reset, interrupt and low-power mode, as well as reading and writing operations midway.

Key words: SOC; IP core; SHA-1/224/256; high performance; low-power

1 引言

目前哈希算法以其优越的性能成为安全技术中的核心算法, 因此设计并且实现一个高效的哈希算法 IP 核, 对于安全芯片来说显得尤为关键。近些年, 有很多 Hash 函数研究与设计^[1], 也有很多关于哈希算法的 IP 核设计, 但大多数都是停留在理论展望或者仅仅实现某一种固定的密码算法的专用密码处理芯片, 如文献[2]是基于 FPGA 的有关 SHA-1 的设计, 文献[3]是基于 FPGA 的 SHA-256 算法国内以及国外的实现。这些都是针对某一种特定的加密算法, 无法满足使用不同密码算法用户的多层级安全要求和集成度的优化^[4]。在文献[4]中, 对 MD5

和 SHA-1 设计了复用 IP 核, 但是由于 MD5 已经证明是不安全的, 而目前使用最为广泛的是 SHA-1 以及后推出的 SHA-224 和 SHA-256 算法。本文在安全 SOC 芯片中实现哈希算法多类型复用 IP 核的设计, 可以选择使用不同的加密模式, 实现 SHA-1、SHA-224 和 SHA-256 算法^[5-6]。

2 复用 IP 核的算法设计

关于 SHA-1、SHA-224 以及 SHA-256 的原始算法, 请先参考文献[5-6]。在深入学习算法的基础上, 我们以下具体讨论在此复用 IP 核(后面称为 SHA_IP)中算法的设计。为了增强 SHA_IP 的可移植性, 数据总线采用 32 比特。

收稿日期: 2013-12-09; 修回日期: 2014-01-28

基金项目: 国家自然科学基金项目(11274150); 辽宁省高等学校优秀人才支持计划(LR2013001)

2.1 运算过程的算法设计

(1) 分解消息

算法核只处理分组后的消息,不进行有关消息的添加和分组工作,所以到达 SHA_IP 的都是分组后的 512 比特消息.将这 512 比特的消息拆分成 16 个 32 比特的字,记为 $W_0, W_1, W_2, W_3, W_4, W_5, W_6, W_7, W_8, W_9, W_{10}, W_{11}, W_{12}, W_{13}, W_{14}, W_{15}$.这 16 个 32 比特的字参与运算,一个分组的运算结束,进行下一分组的运算,一直到完成最后一个分组的运算.

(2) 配置 K_t 值

根据算法选择位,来控制算法模式,SHA-224 和 SHA-256 使用相同的 K_t ,为 64 个 32 比特的数. SHA-1 使用另外的 K_t ,为 80 个 32 比特的数. K_t 为固定值,不可更改,限于篇幅,请参考文献[5-6].

(3) 初始化哈希值

将 256 比特的初始 IV,以 32 比特为单位分成八份,为 $IV_0, IV_1, IV_2, IV_3, IV_4, IV_5, IV_6, IV_7$ (IV_0 为高位). SHA_IP 支持配置默认初始 IV 和配置自定义初始 IV,将配置自定义 IV 的使能置高,即可自行配置初始的哈希值,否则认为使用默认的初始 IV. SHA_IP 中内置默认的初始 IV 如表 1 所示.

表 1 SHA_IP 中内置默认的初始 IV

	SHA-1	SHA-224	SHA-256
IV_0	32h0	32hc1059ed8	32h6a09e667
IV_1	32h0	32h367cd507	32hbb67ae85
IV_2	32h0	32h3070dd17	32h3c6ef372
IV_3	32h67452301	32hf70e5939	32ha54ff53a
IV_4	32hefcadab89	32hffc00b31	32h510e527f
IV_5	32h98badcfe	32h68581511	32h9b05688c
IV_6	32h10325476	32h64f98fa7	32h1f83d9ab
IV_7	32hc3d2e1f0	32hbefa4fa4	32h5be0cd19

(4) 迭代运算

初始化 IV 后,令 $a = IV_0, b = IV_1, c = IV_2, d = IV_3, e = IV_4, f = IV_5, g = IV_6, h = IV_7$. a, b, c, d, e, f, g, h 参与一系列迭代运算,产生最终的 a, b, c, d, e, f, g, h ,再与初始 IV 进行模 32 加法,得到最终的 IV 值,即所求的摘要. $IV_0 = IV_0 + a, IV_1 = IV_1 + b, IV_2 = IV_2 + c, IV_3 = IV_3 + d, IV_4 = IV_4 + e, IV_5 = IV_5 + f, IV_6 = IV_6 + g, IV_7 = IV_7 + h$.若运算的这组消息是最后一组,则输出的 IV 即为输出的摘要值,若运算的这组

消息为中间某一组消息,则所得的 IV 作为下一组运算的初始 IV,继续进行运算.

(5) 输出摘要

最后一组消息运算结束之后,得到最终的 $IV_0 \sim IV_7$,SHA-1 的输出摘要为 160 比特,为 $\{IV_3, IV_4, IV_5, IV_6, IV_7\}$,SHA-224 的输出摘要为 224 比特,为 $\{IV_0, IV_1, IV_2, IV_3, IV_4, IV_5, IV_6\}$,SHA-256 的输出摘要为 256 比特,为 $\{IV_0, IV_1, IV_2, IV_3, IV_4, IV_5, IV_6, IV_7\}$.

2.2 运算核的算法设计

在控制寄存器中,设置两位的算法选择位, $Mode[1:0]$,当 Mode 为 00/01 时,运算核运行 SHA-1 算法;当 Mode 为 11 时,运行 SHA-224 算法;当 Mode 为 10 时,运行 SHA-256 算法.有关 a, b, c, d, e, f, g, h 改进后的迭代运算,如图 1 所示.

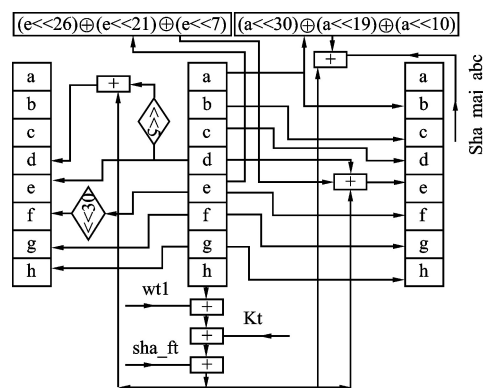


图 1 迭代运算设计

有关图 1 的说明如下:

中间的 a, b, c, d, e, f, g 为初始 IV,左边的运算为 SHA-1,右边的运算为 SHA-225/256,经过一系列的迭代运算,产生相应的摘要值. SHA-1 运算执行 80 次迭代运算,由图示中间的 d, e, f, g, h 不断迭代运算更新左边的 SHA-1 中的 d, e, f, g, h ,当 80 轮迭代完毕,将所产生的 d, e, f, g, h 与最初的 $IV_3, IV_4, IV_5, IV_6, IV_7$ 进行模 32 加法,产生最终的 160 比特摘要. SHA-224/256 运算执行 64 次迭代运算,由图示中间的 a, b, c, d, e, f, g, h 不断迭代运算更新左边的 SHA-1 中的 a, b, c, d, e, f, g, h ,当 64 轮迭代完毕,将所产生的 a, b, c, d, e, f, g, h 与最初的 $IV_0, IV_1, IV_2, IV_3, IV_4, IV_5, IV_6, IV_7$ 进行模 32 加法,产生最终的 256 比特摘要(SHA-224 算法最后的 32 比特不要).

$$sha_maj_abc = (a \& b) | (c \& (a | b));$$

$$Parity = (e \oplus f) \oplus g;$$

$$Ch = (e \& f) | (\sim e \& g);$$

$Maj = (e \& f) | (g \& (e | f));$

$\sigma_0 = \{w_1[6:0], w_1[31:7]\} \oplus \{w_1[17:0], w_1[31:18]\} \oplus \{3b000, w_1[6:0]\};$

$\sigma_1 = \{w_{14}[16:0], w_{14}[31:17]\} \oplus \{w_{14}[18:0], w_{14}[31:19]\} \oplus \{10b0, w_{14}[31:10]\};$

其中, $\{w_1[6:0], w_1[31:7]\}$ 表示对 a 循环左移位 25 位, 迭代运算中的加法均为模 32 加法. 图 2 中 sha_ft, SHA-1, 第一个 20 轮运算时, sha_ft 为 Ch; 第二个 20 轮运算时, sha_ft 为 Prity; 第三个 20 轮运算时, sha_ft 为 Maj; 第四个 20 轮运算时, sha_ft 为 Prity. 而 wt1, 以 SHA-1 算法为例, 原始算法是根据 $W_0 \sim W_{15}$, 生成 $W_{16} \dots W_{55} \dots W_{79}$, 实现 80 轮迭代算法. 在 SHA_IP 设计中使用计数器来控制, 实现对 $W_0 \sim W_{15}$ 的按字为单位的左移位处理, 一轮结束 W_1 就将值传递给 W_0 . 处理时 W_{15} 本来应该得到 W_{16} 的值, 在设计中令 $W_{15} = sha_tmp$. 由于 sha_tmp 是有关 W 的运算, 所以随着 W 的左移, sha_tmp 也实时更新. 这样令 wt1 等于 W_0 , 通过原有的 $W_0 \sim W_{15}$ 加上 sha_tmp, 即可快速高效的实现运算, 不必计算后续的 W 值. 当 SHA-1 时, $sha_tmp = (w_0 \oplus w_2 \oplus w_8 \oplus w_{13}) \ll 1$; 当 SHA-224/256 时, $sha_tmp = w_0 + \sigma_0 + w_9 + \sigma_1$.

3 复用 IP 核的电路设计

SHA_IP 支持任何时候均可读取摘要寄存器, 可以从中读取最终结果, 也可以读取中间结果, 然后到片外继续处理. 主要设置控制寄存器、摘要寄存器、消息寄存器. SHA_IP 支持使用内置的初始 IV, 也支持自定义的初始 IV, 设置好 IV 以后, 输入消息完毕触发运算核启动. 设置自定义初始 IV 和输入消息时, 都通过计数器来控制. 如图 2 所示, 通过计数器 count_iv 来实现初始化自定义 IV, 因为数据总线是 32 位的, 所以写 256 比特的摘要, 需要写 8 次, 所以计数器计 8 个数. 通过计数器 count_mess 来实现消息的写入, 写完 512 比特的消息, 计数器需要计 16 个数. 当最后 32 比特的消息写完时, 也即 count_mess 等于 15 时 (从 0 开始计), 自动启动运算核, 开始运算. IP 核可外挂 RAM, 支持大量数据的运算, 当一包 512 比特的消息运算完毕后, 继续写入 512 比特消息, 继续进行杂凑运算, 消息的大小只与 RAM 的容量有关.

内部设置 Kt 选择模块, 通过计数器 (round) 来控制, 保证迭代运算每一轮的 Kt 的正确, round 计

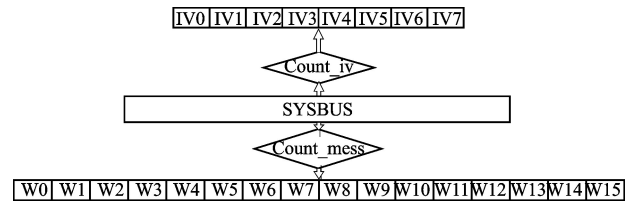


图 2 配置自定义 IV 和写入消息

数器同样控制着迭代运算的正确进行, 每个时钟上升沿的时候, 进行一次迭代运算, 同时计数器要自加 1, 当计数器记满则产生完成标志, 即迭代运算结束, 输出摘要值. SHA-1 和 SHA-224/256 计数器的值是不同的. SHA_IP 核的基本逻辑结构如图 3 所示, 图 3 简单描述了 SHA_IP 核硬件实现的内部结构, 使用 Verilog 进行代码的编写, 实现算法的功能.

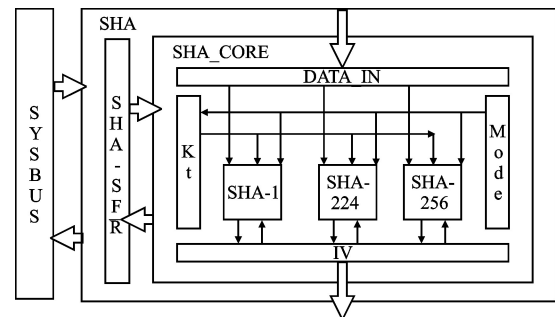


图 3 SHA_IP 基本逻辑结构图

4 仿真验证与性能分析

优化完算法、制定合理的设计方案后, 使用 Verilog 进行代码的编写, 使用 Leda 验证代码风格, 在平台上通过 CVS 仿真, 实现从模式的功能验证. 完成代码的设计和基本功能的验证后, 我们用 Synopsys 公司的 Design Compiler 工具在 $0.18 \mu m$ CMOS 工艺库下进行了综合并进行了静态时序分析, 然后使用 ICC 进行布局布线. 进行后提取仿真, 使用 Veridi 查看波形, 与正确数据对比, 验证功能的正确性.

4.1 复用 IP 核的功能验证

使用内置默认的初始 IV 进行验证, 写完消息后, 运算启动, 当消息写完后, process 信号为高, 表明运算开始, 运算结束后, process 信号恢复低电平. 写入的消息为

512h61626364626364656364656664656667656676866768696768696a68696a6b696a6b6c6d6b6c6d6e6c6d6e6f6d6e6f706e6f70718000000000000000;

分别启动 SHA-1、SHA-224 和 SHA-256, 通过

将原始数据和 VCS 仿真结果进行对比,运算得到的摘要与期望值一致,验证所设计的 IP 核功能正确.

4.2 复用 IP 核的性能分析

从多方面对比分析复用 IP 核的参数,如表 2、表 3 所示.

表 2 综合对比

运算核	资源 (门数)	面积/ μm^2	工作频率/ MHz
SHA_IP(SHA-1/224/256 复用)	8 223 77	505.734 4	100
文献[5](MD5、SHA-1 复用)	20 104 10	0521.724 8	100
SHA-1(文献[5]中的单独设计)	10 467 52	334.962 3	100

表 3 性能对比

算法类型	周期数	最大处理速度/Mb/s
SHA-1	81	632
SHA_IP	SHA-224	65
	SHA-256	65
文献[7](SHA-1)	83	290
文献[4](SHA-256)	66	679
文献[3](SHA-256)(国外)	65	819

由表 2 可知,文献[4]中实现 SHA-1 单运算核,需要资源 10 467 门,面积为 52 334.962 3 μm^2 ;实现 MD5 和 SHA-1 的复用 IP,需要资源 20 104 门,面积为 10 0521.724 8 μm^2 . 所以,SHA_IP 相比于文献[4]的两个算法的复用核,无论从资源还是面积上,都有绝对的优势. 和 SHA-1 单算法核相比,SHA_IP 在面积上略大,所以 SHA_IP 用相当于一个算法的面积实现了包含三个算法的运算核.

由表 3 可知,对复用 IP 核中的三种算法分别进行性能分析,对比 SHA-1、SHA-224 和 SHA-256 三种算法的周期数及最大处理速度,SHA_IP 在运算处理速度上有明显优势,可见它们的性能并没有因为复用而受到影响.

5 结束语

哈希算法在数字签名中占有非常重要的地位,而目前使用较多的就是 SHA-1、SHA-224 以及

SHA-256. 本文重新设计了算法,加快了运算速度,最大程度的优化了 SHA_IP 的性能. 将这三种算法设计实现一个可复用的 SHA_IP,节省了面积,实现了资源的优化配置,而且可以满足不同的安全需求,实现更多的功能. SHA_IP 采用 32 位的数据总线,移植性好,使用率更高. 在设计当中,支持硬件和软件两种复位模式,支持中断,同时支持低功耗模式,在芯片中,当关闭 SHA_IP,SHA_IP 停止工作,进入低功耗模式. 目前该 IP 核已经多款安全 SOC 芯片中使用.

参考文献:

[1] 刘飞. Hash 函数研究与设计[D]. 南京:南京航空航天大学,2012.

[2] 李磊,韩文报. FPGA 上 SHA-1 算法的流水线结构实现[J]. 计算机科学,2011,38(2):58-60.

[3] Algreto-Badillo I, Feregrino-Uribe C, Cumplido R, et al. FPGA-based implementation alternatives for the inner loop of the secure Hash Algorithm SHA-256[J]. Microprocessors and Microsystems, 2012, 37(2013):750-757.

[4] 周琴琴. 基于 Hash 函数的 MD5 和 SHA-1 加密算法研究及其硬件实现[D]. 合肥:安徽大学,2012.

[5] NIST, FIPS PUB 180-2, Secure Hash Standard[s]. USA. National Institute of Standards and Technology (NIST), 2002.

[6] NIST, FIPS PUB 180-3, Secure Hash Standard[s]. USA. National Institute of Standards and Technology (NIST), 2008.

[7] 孙黎,慕德俊,刘航. 基于 FPGA 的 SHA-1 算法的设计与实现 [J]. 计算机工程,2007,33(14):270-274.

作者简介:

刘 坚 男,(1988-),硕士研究生. 研究方向为集成电路设计.

王绩伟 男,(1968-),副教授. 研究方向为光电子材料与器件

张文婧 女,(1975-),高级工程师. 研究方向为信息安全芯片设计.