SM3 算法的 FPGA 设计与实现

丁冬平,高献伟

(北京电子科技学院 电子系,北京 100070)

摘 要:在分析 SM3 算法的基础上详细介绍了目前 Hash 函数的 4 种硬件实现策略,同时给出了迭代方式和基于充分利用时钟周期的循环展开方式下的 FPGA 实现。该循环展开方式有效地减少了一半的工作时钟数和 11%的运算时间,吞吐量提高了 11%,且占用的硬件资源较少。

关键词:SM3;迭代方式;循环展开方式;FPGA;VHDL

中图分类号:TN918.4 **文献标识码**:A **文章编号**:1674-7720(2012)05-0026-03

Design and implementation of SM3 algorithm on FPGA

Ding Dongping, Gao Xianwei

(Department of Electronics, Beijing Electronic Science and Technology Institute, Beijing 100070, China)

Abstract: This paper introduces four ways of the Hash function hardware implemention based on the SM3 algorithm and the iterative way. And FPGA realization is given based on the sufficient use of the clock cycle circle-unroll way. This circle-unroll way availably reduces the number of work clock cycle into half and 11% operation times consumption, improves 11% throughput and occupies little hardware resource.

Key words: SM3; iterative way; circle-unroll way; FPGA; VHDL

Hash 函数是密码学中最基本的模块之一,在密码学中扮演着极其重要的角色,广泛应用于数字签名、消息鉴别和伪随机数生成器等领域,是近几年密码学研究的热点领域[1]。

Hash 函数是将任意长度的信息通过算法变成固定长度的输出,且这个变换过程是不可逆的。Hash 函数的不可逆特性使得攻击者不能通过 Hash 值推出任何部分的原始信息。因为对于原始信息中的任意一个比特数据发生改变,其 Hash 值都将发生明显变化。此外,Hash 函数还具有碰撞约束性,即不能找到一个输入使其输出结果等于一个已知的输出结果,或者不能同时找到两个不同的输入使其输出结果完全一致。正是这些特性,使得 Hash 值可以用来验证信息是否被修改。

1 SM3 算法简介

为了满足电子认证服务系统等应用需求,国家密码管理局于 2010 年 12 月发布了 SM3 密码 Hash 算法。该算法适用于商用密码应用中的数字签名和验证、消息认证码的生成与验证以及随机数的生成,可满足多种密码应用的安全需求。SM3 算法能够对任何小于 2^{64} bit 的数据进行计算,输出长度为 256 bit 的 Hash 值。

SM3 算法包括预处理、消息扩展和计算 Hash 值三部分。预处理部分由消息填充和消息分组两部分组成。首先将接收到的消息末尾填充一个“1”,再添加 k 个“0”,使得填充后的数据成为满足 Length=448 mod 512 bit 的数据长度,再在末尾附上 64 bit 消息长度的二进制表示数,然后将消息分成 512 bit 的子块,最后将每个 512 bit 的消息子块扩展成 132 个字 $W_0, W_1, \cdots, W_{67}, W_0', W_1', \cdots, W_{63}'$ 用于 Hash 值的计算。SM3 算法计算流程图如图 1 所示。

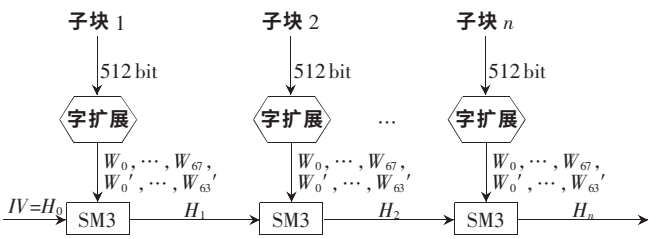

图 1 SM3 算法计算流程图

SM3 算法的 Hash 运算主要是在压缩函数部分,压缩函数共包含 64 轮,每轮包括 12 步运算,64 轮循环计

算结束后，再将计算结果与输入到本轮计算的初始数据进行异或运算，即上一次 Hash 运算的 Hash 值输出 H_i 与输入到本轮计算的初始数据异或得到本次 Hash 值输出 H_{i+1}。H_n 即为最终的 Hash 值，H_0 为设计者提供的初始值 IV。

2 Hash 算法的硬件实现策略

在通过 FPGA 编程实现 Hash 算法过程中，提高吞吐量以及减少硬件资源占用是衡量硬件实现算法的重要指标，高效率、低功耗以及面积优化设计都是 FPGA 设计中最受关注的方面。目前为止，Hash 算法的 FPGA 实现策略大致有以下 4 种方式[1-2]，如图 2 所示。

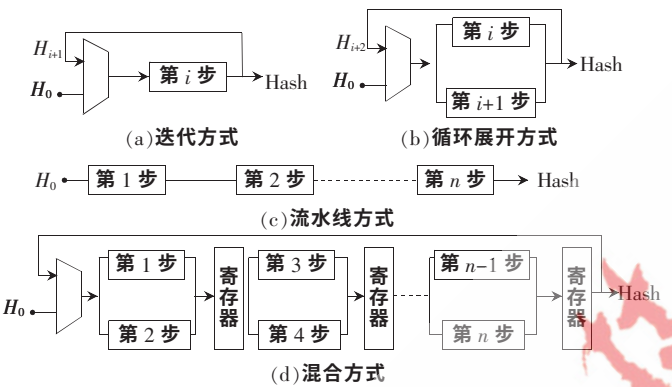

图 2 Hash 函数的硬件实现方式

(1)迭代方式：该方式将单步运算的结果重新反馈到输入端，在节约硬件资源的同时造成了较大的时延，虽然效率较低，但比较实用。

(2)循环展开方式：该方式根据算法的具体特性，将多步运算合并成单步运算，以加大并行运算规模的方式来提高单步运算的效率。

(3)流水线方式：该方式将所有单步运算全部在时钟的控制下予以实现，每个时钟均有输出。全流水线时的吞吐量达到最高，但是硬件资源消耗相当大。由于 Hash 函数的运算特点，该方式很少在实际中使用。

(4)混合方式：该方式实现的算法能在面积和速度上取得平衡。

3 SM3 算法的 FPGA 实现

由于 SM3 算法消息扩展部分的软硬件实现的效率相差不大，因此本文着重讨论该算法的计算部分在 FPGA 上的两种实现方式。

3.1 迭代方式

由于 SM3 算法的每轮计算过程大致相同，因此可以采用迭代方式实现。实现过程中，将存放常数 T_j 和 IV 的常量矩阵利用 ROM 结构实现。分析 SM3 算法的消息扩展和压缩函数的计算过程与特点可以看出，预先通过组合逻辑计算全部 $W_0, W_1, \cdots, W_{67}, W_0', W_1', \cdots, W_{63}'$ 的值需要消耗大量的硬件资源。而在每轮的压缩函数计算过程中，只需使用相应的一组 W_j 和 W_j'，因此便无需预先将 $W_0, W_1, \cdots, W_{67}, W_0', W_1', \cdots, W_{63}'$ 值全部计算出来，可以利用时钟的控制，在每次运算压缩函数之前，预先计算将要被使用的一组 W_j 和 W_j'，显然这将使获得每轮压缩函数运算结果消耗 2 个时钟周期。加上初始值的输入、明文输入以及 Hash 结果输出共消耗的 3 个时钟周期，采用迭代方式进行一次 SM3 算法需要消耗 1+1+1+64×2=131 个时钟周期。

3.2 循环展开方式

仔细分析 SM3 算法的运算过程及迭代方式实现 SM3 算法的设计过程可知，时间主要耗费在消息扩展和压缩函数的计算上[3]。在 SM3 算法的迭代方式实现中，每轮压缩函数的运算和消息扩展运算中均需消耗一个时钟周期，尤其是在进行消息扩展过程中，每组 W_j 和 W_j' 计算量都比较小，利用一个时钟周期去进行运算实在过于浪费。如果在一个时钟周期里进行两组 W_j 和 W_j' 的计算，同时把一个时钟中本来只进行一轮压缩函数的运算也增加到两轮，这样不仅能更充分地利用一个时钟周期提高计算速度，而且整个 SM3 算法核心运算过程的时钟消耗也将缩短到 64 个时钟周期。

3.3 FPGA 实现结果

本文采用 Altera 公司 Stratix II 系列的 EP2S90F1508C3 芯片，以 Quartus II 8.1 为开发环境[4]，采用硬件描述语言 VHDL 进行 SM3 算法的 FPGA 实现。SM3 算法实现的整体结构可分为库函数模块和主程序模块两大模块[1,5]。在 SM3 算法库函数模块中定义了 6 个左循环移位函数 ROL7、ROL9、ROL12、ROL15、ROL19、ROLk 和 4 个函数 FF、GG、P0、P1，均用组合逻辑资源实现，常数 T_j 和 IV 的常量矩阵利用 ROM 结构实现。主程序中定义了实体端口(如图 3 所示)，编译生成的模块图如图 4 所示。用状态机对运算过程进行控制，SM3 算法的主程序中包含了 s00、s01、s02、s03、s04 和 s05 6 个状态。

```
entity sm3 is
  port(
       clk:in std_logic;
       reset:in std_logic;
       inen:in std_logic;
       Min:in std_logic_vector(511 downto 0);
       Vout:out std_logic_vector(255 downto 0);
       ready:out std_logic);
end sm3;
```

图 3 实体端口定义程序

图 4 实体端口模块

以 2010 年 12 月国家密码管理局发布 SM3 算法所附录的运算示例中提供的数据为标准，将实验仿真所得到的计算数据与该标准进行对照，对于一个 512 bit 分组和两个 512 bit 分组，采用迭代方式实现和采用循环展开方式实现均计算出了正确的 Hash 值“66c7f0f4 62eeedd9 d1f2d46b dc10e4e2 4167c487 5cf2f7a2 297da02b 8f4ba8e0”和“debe9ff9 2275b8a1 38604889 c18e5a4d 6fdb70e5 387e5765 293dcba3 9c0c5732”。实验仿真结果分别如图 5~图 8 所示。

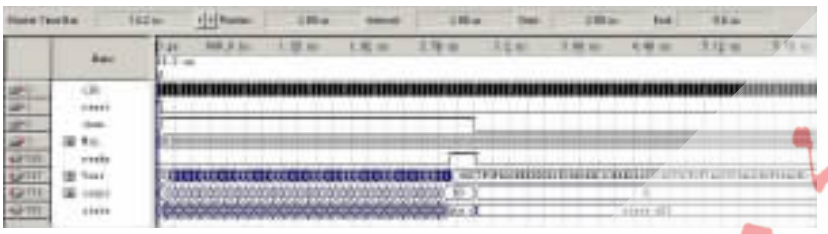

图 5 迭代方式一个分组仿真结果

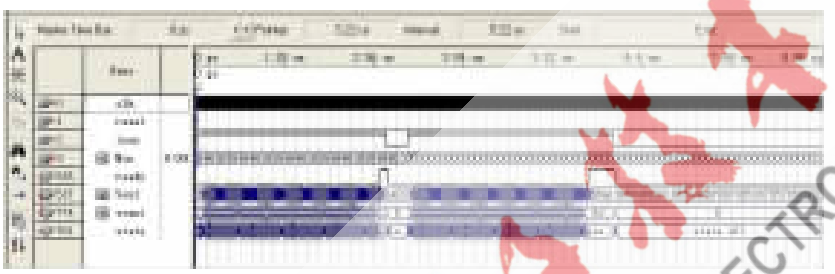

图 6 迭代方式两个分组仿真结果

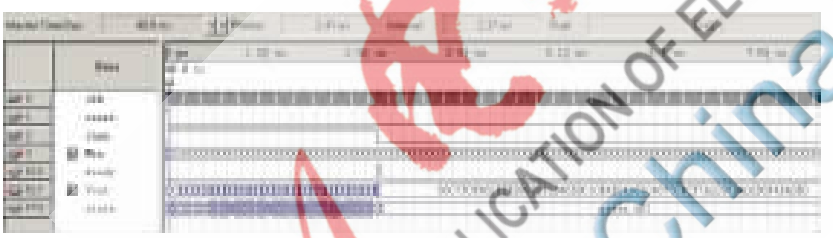

图 7 循环展开方式一个分组仿真结果

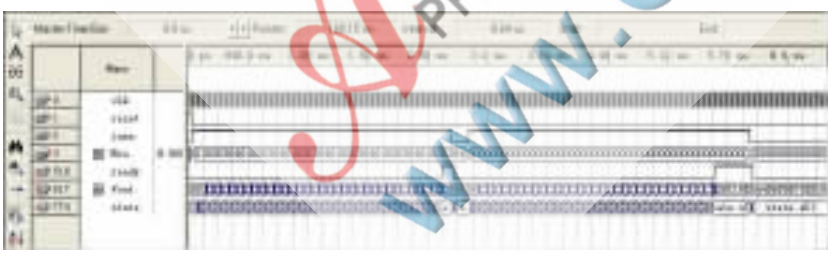

图 8 循环展开方式两个分组仿真结果

表 1 比较了两种实现方式下的硬件资源占用、工作主频、所需时钟数、计算时间及吞吐量。由表 1 可得，循环展开方式的吞吐量是迭代方式的 1.11 倍，即工作效率提高了 11%，循环展开方式的计算时间为迭代方式的 0.9 倍，即时间消耗减少了 11%。

表 1 实验数据(1 为迭代方式，2 为循环展开方式)

	硬件资源占用(ALUTS)	工作主频/MHz	所需时钟数 Cycles	计算时间/μs	吞吐量/Mb/s
1	1 244	121.33	131	1.08	474.21
2	1 936	68.86	67	0.973	526.21

本文成功将 SM3 算法进行了 FPGA 实现，并将基于充分利用时钟周期的循环展开方式与迭代方式实现的 SM3 算法相比较，虽然硬件资源占用有所增加，但其时间消耗减少了 11%，吞吐量提高了 11%。由此可见，对于 SM3 算法的实现，采用循环展开方式比完全的迭代方式效率更高，时间消耗更少，而占用资源的增加与取得的收效相比并不算多，这对追求高效率、低资源消耗的移动设备至关重要。

参考文献

[1] 高献伟，路而红.密码算法 FPGA 实现技术[M].北京：西苑出版社，2010.

[2] RODRIGUEZ-HENRIQUEZ R. SAQIB N A, Díaz Pérez A, et al. Cryptographic Algorithms on reconfigurable hardware [M]. Springer, 2006: 189-220.

[3] 曾旭，高献伟，路而红，等.HASH 算法 MD5 的高速实现 [J]. 成都信息工程学报，2009，24(2)：129-132.

[4] 李洪伟，袁斯华. 基于 Quartus II 的 FPGA/CPLD 设计 [M]. 北京：电子工业出版社，2006.

[5] 刘欲晓，方强.EDA 技术与 VHDL 电路开发应用实践[M].北京：电子工业出版社，2009.

(收稿日期：2011-11-05)

作者简介：

丁冬平，男，1986 年生，硕士研究生，主要研究方向：密码通信技术。

高献伟，男，1970 年生，硕士，教授，主要研究方向：混合信号集成电路中的算法。