

SM3 杂凑算法的 SoPC 组件实现

邵梦丽¹, 殷新春^{1,2}, 李艳梅¹

(1. 扬州大学信息工程学院, 江苏 扬州 225000;

2. 扬州大学广陵学院, 江苏 扬州 225000)

摘要: 首先给出了 SM3 在 SoC 上的实现, 然后主要分析了算法的结构, 选择 Verilog 语言进行算法描述, 使用 ModelSim 进行仿真, 用 SoPC Builder 进行接口封装, 最后在 Cyclone IV 系列的 EP4CE22F17C8N 上进行了实现, 测试表明, 运行频率可以达 165 MHz, 吞吐量为 1 184.8 Mbit/s。

关键词: SM3; FPGA; 硬件实现; SoC

中图分类号: TP393

文献标识码: A

doi: 10.11959/j.issn.2096-109x.2017.00146

Implementation of SM3 algorithm based on SoPC component

SHAO Meng-li¹, YIN Xin-chun^{1,2}, LI Yan-mei³

(1. School of Information Engineering, Yangzhou University, Yangzhou 225000, China;

2. School of Guangling, Yangzhou University, Yangzhou 225000, China)

Abstract: Firstly the realization of SM3 on SoC was given. The structure of the algorithm was analyzed mainly, and the algorithm was realized by the Verilog hardware description language, in order to simulate this algorithm, the Altera simulation software ModelSim was chosen, SoPC Builder was used for interface package, and finally it was implemented in the Cyclone IV EP4CE22F17C8N. Its operation frequency can reach 165 MHz and its throughput rate is 1 184.8 Mbit/s.

Key words: SM3, FPGA, hardware implementation, SoC

1 引言

杂凑算法又叫散列 (Hash) 函数^[1], 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^[2]。它是一种单向密码体制, 可以将任意长度的输入经过变化后得到固定长度的输出^[3-5]。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^[2], 算法是在 SHA-2 基础上设计提出的^[6], 它是一种基于分组迭代结构的杂凑算法^[7]。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^[2]。

随着信息社会的进一步发展, 密码处理速度的要求进一步提升, 如何提升算法的运算速度成为研究热点^[8]。SM3 杂凑算法在实现方面可分为软件实现、硬件实现和软硬件协调实现。在软件实现上, 由于 SM3 算法迭代轮数多, 逻辑运算多, 软件处理速度已经越来越无法满足现阶段大数据实时处理的需要^[9]。软硬件协调指的是一部分的

收稿日期: 2016-11-28; 修回日期: 2017-01-05。通信作者: 殷新春, xcyin@yzu.edu.cn

基金项目: 国家自然科学基金资助项目 (No.61472343); 江苏省普通高校研究生科研创新计划项目 (No.KYLX15_1362)

Foundation Items: The National Natural Science Foundation of China (No.61472343), The Ordinary University Graduate Student Scientific Research Innovation Project of Jiangsu Province (No.KYLX15_1362)

运算由软件实现,另一部分的运算由硬件实现,在 SM3 的软硬件协调过程中一般是由软件实现信息填充和控制压缩迭代,而压缩迭代的具体实现由硬件实现^[10]。硬件实现主要追求的是高运行速率以及低功耗,已经有很多文献对算法进行了实现,但其实现大多不是针对 SoC 实现的。文献[1,11,12]的实现在消息扩展处使用的是移位寄存器,消息填充采用软件。文献[1]的设计目的主要与以往的 SHA-0、SHA-1 和 SHA-2 进行比较,在 CF 压缩函数处的关键路径采用 CSA 树优化进行实现,关键路径延时包括 2 级 CSA 加法延时和 1 级 LUT 延时。文献[11]设计主要与 SHA-2 系列算法进行比较,在 CF 压缩函数处的关键路径花费 2 级 CSA 延时和 2 级 32 bit 加法器延时。文献[12]的压缩函数的关键路径和文献[11]采用的是同一种方法,采用流水线技术实现消息块的压缩,实现效率高,占用硬件资源较多。文献[13]主要改进了加法路径,减少了加法器的延时。文献[14]提出了一种四合一架构,即在同一周期内实现四组压缩运算,该框架在循环展开的基础上,针对关键路径和面积进行逐一改进并给出了完整的实现,硬件实现复杂,资源消耗较多。

本文的主要目的是针对 SM3 算法的 SoC 实现,选取 Altera 公司的综合性 PLD 开发软件 Quartus II 进行开发,采用 SoPC 组件封装的方式进行实现,对硬件实现添加适当的时序约束,最终以 IP 核的方式提供使用。CF 关键路径花费 2 级 32 bit 加法延时和 1 级 CSA 运算延时,运算频率可达到 165 MHz。

2 SM3 算法简介

针对长度为 l bit ($l < 2^{64}$) 的报文消息,SM3 杂凑算法经过填充和迭代压缩,生成杂凑值,杂凑值的长度为 256 bit^[13],图 1 给出了 SM3 算法流程。

首先,输入消息,消息按规则进行消息填充,填充规则如式(1)^[15]所示。

$$l + 1 + k \equiv 448 \pmod{512} \quad (1)$$

其中, k 取值满足式(1)最小的非负整数,填充后得到的消息比特数为 512 的倍数。然后进行分块,将消息分成 $B_1, B_2, \dots, B_n$, 每个分块的长

度为 512 bit, 其中 $n = \frac{l+k+65}{512}$, 再将分组依次输入

迭代过程中进行迭代,得到结果。迭代过程包括消息扩展、CF 压缩函数和寄存器异或。消息扩展主要经过 64 轮运算,完成对 512 bit 的分组扩展,扩展生成 132 个 32 bit 的字,结果供 CF 进行运算,在 CF 中经过迭代、移位和加等运算得到 512 bit 结果,再同寄存器中的数进行异或,将迭代结果提供给下一轮压缩,如此循环,经过多次迭代,得到 512 bit 的杂凑值。

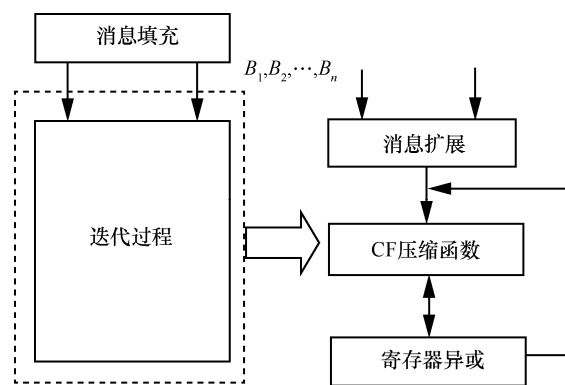


图 1 SM3 算法流程

3 SM3 算法的实现

选择 Verilog 语言进行算法实现,使用 Altera 公司提供的 SoPC 平台进行开发,该平台可以利用 Avalon 平台互连结构连接处理器和各种 IP 核模块^[16],SM3 硬件实现的接口就是依据 Avalon 的接口进行设计和实现的,在 Cyclone IV 系列的 EP4CE22F178 芯片上运行测试。

SM3 算法实现的整体结构可分为控制模块、消息扩展模块、CF 压缩模块,其中具体还包括移位寄存器、CSA 加法等。SM3 算法以 SoPC 组件的形式进行接口设计,封装的接口描述如表 1 所示。

除去顶层的 SoPC 组建封装,SM3 的模块实现如图 2 所示。

图 2 只是算法的硬件实现模块,在消息扩展之前还有一步操作,即消息填充,此步操作采用软件进行实现。消息扩展模块负责向 CF 模块中提供 W_j 和 W'_j ,CF 压缩模块负责对扩展的信息进行压缩,压缩模块中用到的寄存器数据由寄存器

模块提供,然后通过对这3个模块进行内部连线,实现控制。

表 1 SM3 封装接口

接口名称	接口类型	接口功能说明
clk	input wire	时钟输入, 1 bit
reset	input wire	模块置位, 1 bit
div_address	input wire[5:0]	读取和存储数据单元的地址
div_chipselect	input wire	模块运行选择使能接口
div_writedata	input wire[31:0]	可以根据地址向模块中写相应的数据, 如消息分组
div_readdata	output wire[31:0]	读取运行结果或运行结束标志
div_irq	output wire	压缩完成时, 发出中断请求

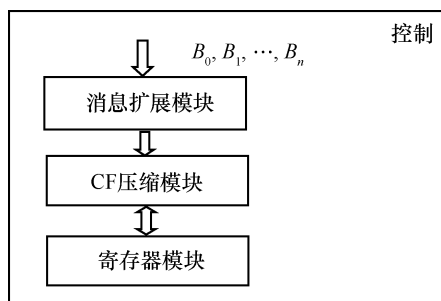


图 2 SM3 硬件实现模块

3.1 消息扩展模块

消息扩展模块主要使用移位寄存器进行实现,移位寄存器由16个32 bit寄存器组成。SoPC组件的使用和直接硬件使用方式不同,直接硬件实现模块主要注重硬件之间的时序传输,而SoPC组件则需要考虑软硬件之间的协调,为了避免硬件过于复杂,此处寄存器中的初始值由处理器通过封装接口传输,传输完成以后再进行移位等相关运算。每周消息扩展模块左移1个寄存器,将运算所得的新数据存入移位寄存器右边,同时将所需的 W_j 和 W'_j 通过相应接口传至CF模块。

消息扩展实现的状态机如图3所示。

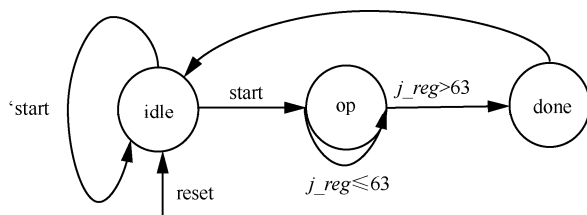


图 3 消息扩展的状态机

在图3中,状态机主要包括3个状态,分别为idle、op和done。idle是状态机的初始化状态,主要对寄存器进行初始化。op是中间运算,负责生成 W_j 和 W'_j ,以及判断是否要结束运算,生成全部 W_j 和 W'_j 花费64个周期。done表示状态机结束,并使状态机进入idle状态。

3.2 寄存器模块

消息扩展的主要目的是将 W_j 和 W'_j 传给CF压缩函数进行消息压缩,得到256 bit的压缩数据,在消息压缩之前需要对一些数据进行初始化。寄存器模块是实现新一轮新的CF压缩函数数据的初始化,迭代过程如下^[15]。

For $i=0$ to $n-1$

$$V^{(i+1)} = CF(V^{(i)}, B^{(i)})$$

End

寄存器模块主要实现 $V^{(i)}$ 的初始化及其后的 $V^{(i+1)}$,数值存放在8个32 bit的寄存器中,用A、B、C、D、E、F、G、H表示。其中, i 代表信息块号,当 i 为0时, $V^{(0)}=IV$,此时 $A=7380166f$, $B=4914b2b9$, $C=172442d7$, $D=da8a0600$, $E=a96f30bc$, $F=163138aa$, $G=e38dee4d$, $H=b0fb0e4e$ ^[13],寄存器中的数值是以初值的方式存储的,其存储方式可以使用ROM进行存储,但是考虑ROM的读取耗时,此处直接以硬件的方式进行实现。

3.3 CF压缩模块

CF压缩是SM3算法设计的关键,其实现的快慢决定SM3算法的实现速率。在压缩模块中涉及的运算包括移位、一些数值的读取和加法运算。其中加法运算速度的快慢对CF压缩运算速度起关键作用。提高CF压缩运算的一个措施是提高迭代并行度,但CF算法中关键路径的运算结果在下次迭代中需要使用,因此并行迭代并不可用。以往的文献中所采用的并行迭代并不是并行的,只是将多轮运算进行合并后优化,这样会占用大量的硬件资源。所以,本文仍然采用单轮进行实现。

另一种提高算法运算速率的方法是缩短关键路径的运行时间。有一部分文献在关键路径上使用的是CSA树优化关键路径^[1],也有使用改进算法的,如文献[13]中所使用的。本文的关键路径花费2级32 bit加法延时和1级CSA运算延时,具体实现过程如图4所示。

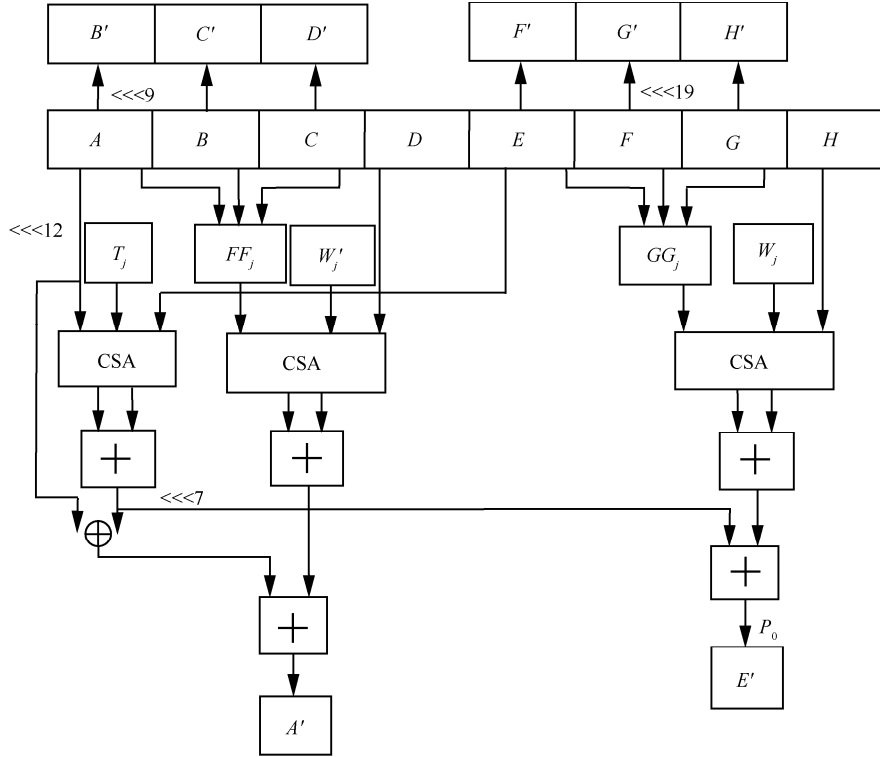


图 4 压缩函数

在图 4 中, A 、 B 、 C 、 D 、 E 、 F 、 G 、 H 、 A' 、 B' 、 C' 、 D' 、 E' 、 F' 、 G' 、 H' 分别代表字 (32 bit) 寄存器, A' 、 B' 、 C' 、 D' 、 E' 、 F' 、 G' 、 H' 存放的是运算后所得到的新值。压缩函数中所涉及的函数运算包括布尔函数 FF_j 、布尔函数 GG_j 、常量 T_j 以及置换函数 P_0 , 具体函数分别如式(2)~式(5)^[15]所示。

$$FF_j(X, Y, Z) = \begin{cases} X \oplus Y \oplus Z, & 0 \leq j \leq 15 \\ (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z), & 16 \leq j \leq 63 \end{cases} \quad (2)$$

$$GG_j(X, Y, Z) = \begin{cases} X \oplus Y \oplus Z, & 0 \leq j \leq 15 \\ (X \wedge Y) \vee (\neg X \wedge Z), & 16 \leq j \leq 63 \end{cases} \quad (3)$$

$$T_j = \begin{cases} 79cc4519, & 0 \leq j \leq 15 \\ 7a879d8a, & 16 \leq j \leq 63 \end{cases} \quad (4)$$

$$P_0(X) = X \oplus (X \ll 9) \oplus (X \ll 17) \quad (5)$$

在式(2)~式(5)中, $\wedge$ 表示 32 bit 与运算, $\vee$ 表示 32 bit 或运算, $\neg$ 表示 32 bit 非运算。在实现过程中, 布尔函数和置换函数只涉及基本门电路的运算, 硬件实现简单。常量 T_j 是通过寄存器进行赋值实现的。

信息压缩的状态机如图 5 所示。

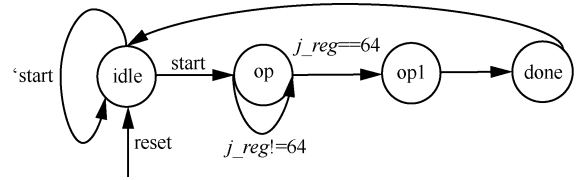


图 5 压缩函数的状态机

压缩函数运行的状态可分为 4 个, 分别为 idle、done、op 和 op1。idle 是状态机的初始状态, 主要进行数据的初始化, 从寄存器模块中读取数据给 A 、 B 、 C 、 D 、 E 、 F 、 G 、 H 这 8 个寄存器。op 和 op1 是运算部分, 主要完成压缩运算, 具体运行方式按图 5 给出的方式进行, 其中, op1 不需要预算下一组要使用的布尔函数以及常量。done 标志运算结束, 产生完成信号。

3.4 算法软件实现

SM3 以 SoPC 的组件方式进行实现, 主要是为了方便 CPU 的调用, 通过软硬件协调的方式达到硬件的最大使用率。CPU 对组件的控制主要是通过地址端口进行读写数据实现的, 其寄存器地址映射信息如表 2 所示。

算法使用 C 语言进行硬件调用, 使用到的宏变量如表 3 所示。

表 2 寄存器映射

地址类型	偏移量	说明
写地址	偏移量 0~15	存放分组信息(第 31~0 位)
写地址	偏移量 16	Start 信号寄存器, 用于产生使能脉冲(第 0 位)
写地址	偏移量 17	Fenzu 信号寄存器, 用于确定 CF 数据的初始化(第 0 位)
写地址	偏移量 18	done_tick 信号寄存器, 虚拟数据, 用于清除 done_tick 标记(第 0 位)
读地址	偏移量 19~26	分组压缩信息寄存器(第 31~0 位)
读地址	偏移量 27	ready 信号寄存器(第 0 位)
读地址	偏移量 18	done_tick 信号寄存器(第 0 位)

表 3 宏变量

宏变量	说明
#define DATA_IN_OFT 0	分组信息首地址
#define START_REG_OFT 16	start 信号地址
#define FENZU_REG_OFT 17	fenzu 信号地址
#define DATA_OUT_OFT 19	分组压缩信息首地址
#define READY_OFT 27	ready 信号寄存地址
#define DONE_REG_OFT 18	done_tick 信号寄存地址

软件实现的核心代码如下所示。

```
void sm3(alt_u8 fenzu, alt_u32 *mes)
{
    alt_u32 ready, done;
    int j=0;
    while(1)
    {
        ready=IORD(SM3_CF_BASE, READY_
OFT)&0x00000001;
        if(ready==1)
            break;
    }
    for(j;j<=15;j++)
        IOWR(SM3_CF_BASE, DATA_IN_OFT+
j, mes++);
    while(1)//等待运算完成
    {
        done=IORD(SM3_CF_BASE, DONE_RE
G_OFT)&0x00000001;
        if(done==1)
            break;
    }
}
```

4 实验

4.1 算法实现

Quartus II 是 Altera 公司的综合性 PLD 开发软件, 支持 VHDL、AHDL、VerilogHDL 等硬件描述语言以及原理图等多种设计输入形式, 可以完成从设计输入到硬件配置的完整 PLD 设计流程, 内嵌自有的综合器以及仿真器^[17]。本文算法的开发就是基于这个软件完成的。最终在 Cyclone IV EP4CE22F178 芯片上运行实现。实验所选的 SoPC 系统中包括 Nios II 处理器、JTAG UART 模块、系统 ID 模块、SRAM 控制模块、定时器模块和 SM3_CF 各 1 个, 其中 SM3_CF 是 SM3 的 SoPC 组件实现方式。实验以国家密码管理局颁布的 SM3 杂凑算法提供的运行示例进行, 对消息“abcd”进行填充, 其运行结果如图 6 所示, 结果运行正确。

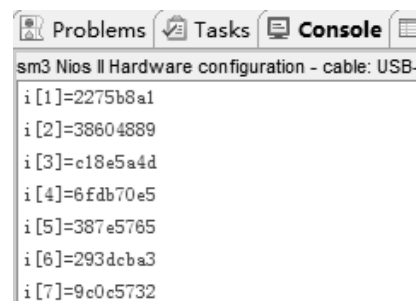


图 6 运行结果

4.2 结果分析

算法实现的资源和吞吐率^[1]统计如表 4 所示。由于实现的目标器件的不同, 同一实现方法占用的资源数也不同, 从资源使用上并不能分析出确切的优劣势, 所以本文将对算法实现的吞吐率进行了统一, 均化为 Mbit/s。表 4 中的“—”表示参考文献中并未提及的部分。

文献[1]采用 CSA 优化树作为关键路径, 在不同的硬件上得到的吞吐率稍有不同。文献[2]使用 2 种方法对算法进行实现, 一种是迭代方式, 另一种是循环展开式。文献[12]采用流水线实现方式。文献[13]对 CF 压缩函数的关键路径进行改进。文献[14]对 CF 关键路径采用四合一进行实现。

本文主要针对的是 SM3 算法在 SoC 上的实现, 选择 SoPC 作为实现途径, 在实现中考虑硬

表4 资源统计比较

文献	器件	资源	频率 (MHz)	吞吐率/(Mbit·s ⁻¹)
文献[1]	V2	1035(Slices)	105.7	795
文献[1]	V2P	1035(Slices)	124.6	938
文献[1]	Virtex-5	328(Slices)	197.2	1 483
文献[2]	EP2S90F1508C3	1244(ALUTS)	131	474.21
文献[2]	EP2S90F1508C3	1936(ALUTS)	68.86	526.21
文献[12]	—	76300(LE)	191	75 304
文献[13]	EPC20F324C6	1789(LE)	92.55	724.40
文献[14]	—	—	125	32 768
本文	EP4CE22F17C8	3933(LE)	165	1 184.8

件的使用量、运算部件的组件化和运算过程中数据读取耗时,对于寄存器的初值采用直接存储赋值方式。实现中消息扩展模块花费64个周期,CPU通过写地址对寄存器进行初始化要花费16个时钟周期,算法运行采用软硬件同步进行的方式,在写入数据的同时可以进行硬件的运行操作,所以此处忽略CPU写入花费的周期。硬件运行过程中,CPU控制开始要花费一个周期,判断硬件是否就绪需要一个周期,CF运算相对于消息扩展会延迟一个周期,完成运算时向CPU发出完成信号需花费一个周期,所以,总的花费周期是 $68(1+1+1+64+1=68)$,吞吐率为1 184.8 Mbit/s。

5 结束语

杂凑算法在密码加密中具有很重要的作用,SM3算法是国家密码局颁布的,主要用于商用密码。SM3算法的实现可分为软件实现和硬件实现,为了提高算法的运算速度,一些场合需要对算法进行硬件实现。本文SM3算法的实现主要用于SoC的实现,主要分析以往的实现算法,最终根据实际要求选取算法进行硬件实现。算法在实现上以Avalon接口为标准进行封装,封装成SoPC组件,最终在Cyclone IV EP4CE22F 17C8N上运行实现,测试表明其运行频率可达165 MHz,吞吐量为1 184.8 Mbit/s。

参考文献:

- [1] 刘宗斌,马原,荆继武. SM3散列算法的硬件实现与研究[J]. 信息安全学报, 2011,(9): 191-193.

- LIU B, MA Y, JING W, et al. Implementation of SM3 Hash function on FPGA[J]. Netinfo Security, 2011,(9):191-193.
- [2] 丁冬平,高献伟. SM3算法的FPGA设计与实现[J]. 微型机与应用, 2012,31(5):26-28.
- DING D P, GAO X W. Design and implementation of SM3 algorithm on FPGA[J]. Microcomputer & Its Applications, 2012, 31(5): 26-28.
- [3] 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):76-79.
- GUO W P, LIU Z L, CHEN Y C, et al. A high throughput and low-power implementation of the SHA-1 Hash function[J]. Microelectronics & Computer, 2008, 25(5):76-79.
- [4] HENZEN L, AUMASSON J P, MEIER W, et al. VLSI characterization of the cryptographic hash function blake[J]. IEEE Transactions on Very Large Scale Integration Systems, 2011, 19(10): 1746-1754.
- [5] MA Y, XIA L N, LIN J Q, et al. Hardware performance optimization and evaluation of SM3 Hash algorithm on FPGA[M]// Information and Communications Security. 2012:105-118.
- [6] 田椒陵. SM3算法界面设计及安全性分析[J]. 信息安全与技术, 2014,(5):24-26.
- TIAN J L. SM3 algorithm interface design and safety analysis[J]. Information Security and Technology, 2014,(5):24-26.
- [7] 郑强,高能,张令臣. 基于SM3算法的动态口令卡的设计与实现[J]. 计算机应用与软件, 2013,30(2):14-17.
- ZHENG Q, GAO N, ZHANG L C. Design and implementation of dynamic password card based on SM3 algorithm[J]. Computer Applications and Software, 2013, 30(2):14-17.
- [8] 于永鹏,严迎建,李伟. SM3算法高速ASIC设计及实现[J]. 微电子学与计算机, 2016,33(4):21-26.
- YU Y P, YAN Y J, LI W. High speed ASIC design and implementation of SM3 algorithm[J]. Microelectronics & Computer, 2016, 33(4): 21-26.
- [9] 周威,王博,张卫东. SM3算法硬件实现研究与应用[J]. 电子测量技术, 2015, 38(12):67-71.
- ZHOU W, WANG B, ZHANG W D. Research and application of SM3 hardware implementation[J]. Electronic Measurement Technology, 2015,38(12):67-71.
- [10] QU, K G, WANG A. A novel masking scheme for SM3 based

- MAC[J]. China Communications.2016.12 (6):11-21.
- [11] 朱宁龙, 戴紫彬, 张立朝. SM3 及 SHA-2 系列算法硬件可重构设计与实现[J]. 微电子学,2015,45(6):777-780+784.
- ZHU N L, DAI Z B, ZHANG L C, et al. Design and implementation of hardware reconfiguration for SM3 and SHA-2 Hash function[J]. Microelectronics,2015,45(6):777-780+784.
- [12] 蔡冰清, 白国强. SM3 杂凑算法的流水线结构硬件实现[J]. 微电子学与计算机,2015,32(1):15-18.
- CAI B Q, BAI G Q. Implementation of pipeline structure of SM3 algorithm on FPGA[J]. Microelectronics & Computer, 2015, 32(1): 15-18.
- [13] 王晓燕, 杨先文. 基于 FPGA 的 SM3 算法优化设计与实现[J]. 计算机工程,2012,38(6):244-246.
- WANG X Y, YANG X W. Optimization design and Implementation of SM3 algorithm based on FPGA[J]. Computer Engineering, 2012, 38(6):244-246.
- [14] 张倩, 李树国. SM3 杂凑算法的 ASIC 设计和实现[J]. 微电子学与计算机,2014,31(9):143-146.
- ZHANG Q, LI S G. Design and Implementation of SM3 Algorithm in ASIC[J]. Microelectronics & Computer, 2014, 31(9): 143- 146.
- [15] 国家密码管理局. SM3 密码杂凑算法[EB/OL]. http://www.oscca.gov.cn/News/201012/News_1199.htm.
- State Cryptography Administration Office of Security Commercial Code Administration. SM3 Cryptographic Hash algorithm [EB/OL]. http://www.oscca.gov.cn/News/201012/News_1199.htm.
- [16] PONG P. 基于 Nios II 的嵌入式 SoPC 系统设计与 Verilog 开发实例[M].北京: 电子工业出版社, 2015:311-323.
- PONG P. Embedded SoPC design with Nios II processor and Verilog examples[M]. Beijing: Publishing House of Electronics Industry, 2015:311-323.
- [17] 王乐井. 基于 SOPC 技术的实时图像处理系统的设计与研究[D]. 长春: 吉林大学, 2010.

WANG L J. The design and research of real time image processing system based on SOPC technology[D]. Changchun: Jilin University, 2010.

作者简介:



邵梦丽(1991-), 女, 江苏盐城人, 扬州大学硕士生, 主要研究方向为国密算法的 SoC 设计与实现。



殷新春(1962-), 男, 江苏姜堰人, 博士, 扬州大学教授、博士生导师, 主要研究方向为密码学、软件质量保障、高性能计算。



李艳梅(1991-), 女, 江苏徐州人, 扬州大学硕士生, 主要研究方向为信息安全。