应用于后量子密码的高速高效 SHA-3 硬件单元设计

刘冬生, 陈　勇, 熊思琦, 杨　朔, 胡　昂

华中科技大学光学与电子信息学院　武汉　中国　430074

摘要　随着量子计算技术的高速发展, 传统的公钥密码体制正在遭受破译的威胁, 将现有加密技术过渡到具有量子安全的后量子密码方案上是现阶段密码学界的研究热点。在现有的后量子密码(Post-Quantum Cryptography, PQC)方案中, 基于格问题的密码方案由于其安全性, 易实施性和使用灵活的众多优点, 成为了最具潜力的 PQC 方案。SHA-3 作为格密码方案中用于生成伪随机序列以及对关键信息散列的核心算子之一, 其实现性能对整体后量子密码方案性能具有重要影响。考虑到今后 PQC 在多种设备场景下部署的巨大需求, SHA-3 的硬件实现面临着高性能与有限资源开销相互制约的瓶颈挑战。对此, 本文提出了一种高效高速的 SHA-3 硬件结构, 这种结构可以应用于所有的 SHA-3 家族函数中。首先, 本设计将 64 bit 轮常数简化为 7 bit, 既减少了轮常数所需的存储空间, 也降低了运算复杂度。其次, 提出了一种新型的流水线结构, 这种新型结构相比于通常的流水线结构对关键路径分割得更加均匀。最后, 将新型流水线结构与展开的优化方法结合, 使系统的吞吐量大幅提高。本设计基于 Xilinx Virtex-6 现场可编程逻辑阵列(FPGA)完成了原型实现, 结果显示, 所设计的 SHA-3 硬件单元最高工作频率可达 459 MHz, 效率达到 14.71 Mbps/Slice。相比于现有的相关设计, 最大工作频率提高了 10.9%, 效率提升了 28.2%。

关键词　后量子密码; 哈希算法; 硬件实现; SHA-3

中图法分类号　TN402　DOI 号 10.19363/J.cnki.cn10-1380/tn.2021.11.03

Design of High-Speed and High-Efficiency SHA-3 Hardware Unit for Post-Quantum Cryptography

LIU Dongsheng, CHEN Yong, XIONG Siqi, YANG Shuo, HU Ang

School of Optical and Electronic Information, Huazhong University of Science and Technology, Wuhan 430074, China

Abstract　With the rapid development of quantum computing technology, traditional public-key cryptosystems are being threatened by deciphering. The transition from existing encryption technology to post-quantum cryptographic schemes with quantum security is a research hotspot in cryptography at this stage. Among the existing post-quantum cryptographic schemes, the cryptographic scheme based on lattice problem has become one of the most potential PQC schemes due to its advantages of small public key, fast speed and good diversity. As the crucial component in lattice-based PQC schemes, Secure Hash Algorithm-3 (SHA-3) is used as hash functions and extendable-output functions to generate streams of uniformly random numbers, which are then sampled as the pseudorandom matrix or noise polynomials. Considering the great demand for PQC schemes in future diversified applications, the implementation of SHA-3 faces the challenge of limited hardware resource and high performance. In this paper, an efficient hardware architecture of SHA-3 is presented, which is two times unrolled with two inside pipeline registers (IPR) inside of the transformation round and two output pipeline registers (OPR) between adjacent rounds. The proposed design can be employed in all SHA-3 modes and support both one-block and multi-block messages. Firstly, through analyzing the characteristics in detail, this paper simplified the original 64-bit round constants in ι operation of Keccak-p[1600,24] to 7 bits in order to reduce resource consumption. Secondly, a novel pipeline technique inside of the transformation round is developed. Compared with the conventional pipeline technique, it divides the critical path precisely and improves the speed while keeping the resource consumption at a low level. Thirdly, a SHA-3 architecture based on unrolling and pipelining technology is proposed. Unrolling reduces the number of cycles needed for operation. Pipelining brings the advantages of improving the speed and the amount of messages that can be hashed in parallel. With this composite architecture, the throughput can be further enhanced. To the best of our current knowledge, this paper presents the most high-speed and efficient hardware implementation of SHA-3 on Xilinx Virtex-6 FPGA: maximum frequency of 459 MHz and hardware efficiency (throughput/area) of 14.71 Mbps/Slices. When compared to the state-of-the-art related designs, our implementation can realize a 10.9% improvement in max frequency and 28.2% improvement in hardware efficiency.

通讯作者: 陈勇, 硕士研究生, Email: yorkchen576@qq.com。

本课题得到国家自然科学基金面上项目(No. 61874163), 国家自然科学基金重点项目(No. 62134002)资助。

收稿日期: 2021-08-30; 修改日期: 2021-10-13; 定稿日期: 2021-10-19

Key words post-quantum cryptography; hash algorithm; hardware implementation; SHA-3

1 引言

量子计算机技术和量子算法的不断发展为现有的经典密码体制的破译提供了更有力、更致命的攻击方法, 使其不再有足够的安全性保障现代社会的信息安全。如著名的 Shor 量子算法可以在多项式时间内分解大整数和求解离散对数, 因此对于传统公钥密码体制如 RSA、ECC、DSA、ElGamal 等, 采用增加密码长度和参数大小来抵御安全攻击的方式也不再有效。为了抵御量子计算机的攻击, 后量子密码(Post-Quantum Cryptography, PQC)应运而生。

2020 年 7 月, 在 NIST 公布的后量子密码标准候选算法第三轮筛选中, 基于格的算法占据主导地位[1]。格密码在安全性上能抵御已知的量子攻击, 且具有公钥小, 计算速度快, 功能多样的优点, 更易于集成化、小型化、芯片化。因此, 基于格的 PQC 方案具有重要的研究与应用价值。在众多格密码方案中, SHA-3 作为其中重要的核心算子不仅可以对关键信息进行散列, 同时可以产生随机数比特流, 从而用于采样模块生成满足条件的采样值。表 1 显示了应用于最新一轮评选的后量子密码方案的 SHA-3 函数家族。可以看到, 进入第三轮评估的 PQC 方案中至少用到了一种 SHA-3 函数。在诸多格密码方案中, SHA-3 函数是通用的核心算子, 占用了约 20%以上的资源开销和处理周期[2-3]。

表 1 第三轮后量子密码方案中的 SHA-3 函数

Table 1 SHA-3 in third-round PQC schemes

后量子密码方案	SHA-3 函数
Classic McEliece	SHAKE256
CRYSTAL-KYBER	SHA3-256, SHA3-512, SHAKE128, SHAKE256
NURU	SHA3-256, SHAKE256
SABER	SHA3-256, SHA3-512, SHAKE128
CRYSTAL-DILITHIUM	SHAKE128, SHAKE256
FALCON	SHAKE256

自 Keccak 算法被评选为第三代安全散列算法(Secure Hash Algorithm-3, SHA-3)以来, 国内外对该算法的研究热度不断提升。根据不同的目标平台和实现要求, 在 SHA-3 算法的优化方向主要集中在以下三个方面: 更低的资源开销[4-5], 更高的实现性能[6-7]以及性能面积的折中设计[8-9]。

本文从 SHA-3 的原理出发, 研究了应用于格密码的高速高效的 SHA-3 硬件实现和优化, 提出了一种全新的展开级流水线结构。本文首先介绍了 SHA-3 关键环节的原理, 然后提出了 SHA-3 的硬件实现方式, 具体包括轮常数的简化存储以及新型的流水线技术, 并在此基础上运用了展开的结构完成了 SHA-3 高速高效的硬件设计, 最后给出了本设计在 FPGA 上的实践过程和结果以及与前人工作的对比。

2 SHA-3 算法分析

哈希函数是把任意长度的输入消息串变化成固定长的输出串且由输出串难以推算输入串的一类函数。其安全性主要体现在抗碰撞攻击、抗原象攻击和抗第二原象攻击三个方面。在 SHA-3 算法的评选过程中, 由意法半导体公司的 Guido Bertoai Bertoai、Jean Daemen Daemen、Gilles Van Assche Assche 与恩智半导体公司的 Micha Michaëël Peeters 联合设计的 Keccak 算法由于采用了不同于传统 Merkle- Damagard 结构的新型海绵结构, 具有可证明的良好安全性, 并且很容易在安全强度和处理速度之间取得平衡, 因此在 2012 年被确定为 SHA-3 的获胜算法[10]。

SHA-3 算法最终提交的是 Keccak-f[1600], 迭代轮数为 24 轮。该算法的新颖之处在于提出的新型海绵结构。海绵结构使用有限的状态, 接收任何长度的输入位流且能满足任何长度的输出。该结构主要由内部函数 f、分组位长度 r 和填充算法 pad 这三组参数定义。海绵结构分两个阶段进行处理: 吸收阶段和挤压阶段。在吸收阶段, 先采用多重位速率填充的方法对输入的数据进行填充, 再将填充后的消息分为 M1, M2, … , Mn 共 n 块, 每块为 r 位, 依次将该消息块依次与产生的 r 位内部状态做异或, 然后将 Keccak-f[1600]函数作用于该状态, 生成的新状态再与下一消息块进行同样的操作。如图 1 所示, M1 先与海绵结构的初始状态(空状态)异或并输入 f 函数进行处理并产生输出, M2 再与该输出异或, 如此循环往复 n 次, 直到所有的消息块都被处理, 之后进入压缩阶段。在压缩阶段中, 若所需输出消息摘要长度短于吸收阶段的输出, 则直接截取相应长度, 跳过此

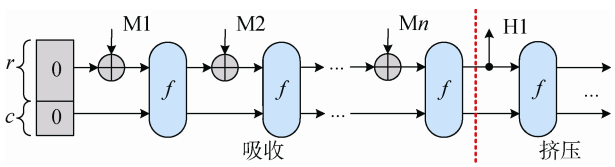


图 1 海绵函数结构

Figure 1 Sponge construction of Keccak

阶段; 反之, 则需再执行 Keccak-f[1600]函数, 并将结果串联起来, 直到得到所需长度的输出。

迭代轮函数的实现过程如下: 在置换函数 f 中, 取 1600-bits 状态, 将其转换为一个 5*5*64 的三维数组作为输入, 并基于该数组进行 24 轮变换, 每一轮变换包括 $\theta, \rho, \pi, \chi, \iota$ 5 个步骤。Keccak-f[1600] 5 个步骤中的前 4 步都是在三维状态矩阵中进行不同方向的行(row)列(column)和道(lane)变换, 将所有元素混淆和扩散。而最后一步是将一组各不相同的轮常量添加到元素中以打破其他 4 个变换的对称性。下式为变换步骤的表达:

1. 步骤θ: 对于 $0\leqslant x<5$ 且 $0\leqslant y<5$,

$$C[x]=A[x,0]\oplus A[x,1]\oplus A[x,2]\oplus A[x,3]\oplus A[x,4] \tag{1}$$

$$D[x]=C[x-1]\oplus rot(C[x+1],1) \tag{2}$$

$$A[x,y]=A[x,y]\oplus D[x] \tag{3}$$

2. 步骤ρ和π: 对于 $0\leqslant x<5$ 且 $0\leqslant y<5$,

$$B[y,2x+3y]=rot(A[x,y],r[x,y]) \tag{4}$$

3. 步骤χ: 对于 $0\leqslant x<5$ 且 $0\leqslant y<5$,

$$A[x,y]=B[x,y]\oplus((\sim B[x+1,y])\wedge B[x+2,y]) \tag{5}$$

4. 步骤ι,

$$A[0,0]=A[0,0]\oplus RC[i] \tag{6}$$

表 2 表示了步骤ι中所采用的轮常数。

表 2 轮常数值(16 进制)

Table 2 Value of round constants

RC[0]	0x0000000000000001	RC[12]	0x000000008000808B
RC[1]	0x0000000000008082	RC[13]	0x800000000000008B
RC[2]	0x800000000000808A	RC[14]	0x8000000000008089
RC[3]	0x8000000080008000	RC[15]	0x8000000000008003
RC[4]	0x000000000000808B	RC[16]	0x8000000000008002
RC[5]	0x0000000080000001	RC[17]	0x8000000000000080
RC[6]	0x8000000080008081	RC[18]	0x000000000000800A
RC[7]	0x8000000000008009	RC[19]	0x800000008000000A
RC[8]	0x000000000000008A	RC[20]	0x8000000080008081
RC[9]	0x0000000000000088	RC[21]	0x8000000000008080
RC[10]	0x0000000080008009	RC[22]	0x0000000080000001
RC[11]	0x000000008000000A	RC[23]	0x8000000080008008

3 SHA3 硬件设计

3.1 轮常数的简化

在 Keccak-f[1600]五轮算法中的最后一轮ι变换中采用了轮常数与 S[0,0]道元素相异或以打破原有变换对称性的形式。通过表 2 可以分析出, 在这 24 组 64 位轮常数中只有其中的 0, 1, 3, 7, 15, 31 和 63 位可能产生变化, 而余下的 57 位均为零。为了减少运算量以及存储空间, M. M. Wong [9]提出了将 64 比特轮常数 SRC(simple round constants)简化为 8 比特的方法, 在本文设计的硬件结构中, 进一步将其压缩为 7 比特, 简化后的轮常数见表 3。

仅存储 7 个非零位的优势如下:

(1) 降低资源消耗: 将原有的 64 位缩减为 7 位, 所需寄存器大幅缩减, 硬件开销减小了 90%左右;

(2) 简化计算: 轮常数用于参与异或运算, 采用本设计中的简化值, 只需要运算 SRC 中的 7 位与 S[0,0]中对应元素的异或结果, 其他位直接输出原值即可。

表 3 简化的轮常数(2 进制)

Table 3 Value of simplified round constants

SRC[0]	0b0000001	SRC[12]	0b0111111
SRC[1]	0b0011010	SRC[13]	0b1001111
SRC[2]	0b1011110	SRC[14]	0b1011101
SRC[3]	0b1110000	SRC[15]	0b1010011
SRC[4]	0b0011111	SRC[16]	0b1010010
SRC[5]	0b0100001	SRC[17]	0b1001000
SRC[6]	0b1111001	SRC[18]	0b0010110
SRC[7]	0b1010101	SRC[19]	0b1100110
SRC[8]	0b0001110	SRC[20]	0b1111001
SRC[9]	0b0001100	SRC[21]	0b1011000
SRC[10]	0b0110101	SRC[22]	0b0100001
SRC[11]	0b0100110	SRC[23]	0b1110100

3.2 改进的流水线技术

在 SHA-3 的硬件结构设计中一般采用流水线结构, 基于流水线技术, 在 Keccak 的轮函数算法中数据可以并行传输, 这对于提高系统的时钟频率和增加数据吞吐量有明显的优势。但与此同时, 硬件开销也会相应增加。在硬件设计过程中, 流水线寄存器一般插入在轮与轮之间或者每轮内部[11], 以减小决定运行时延的关键路径。亟待解决的问题是, 如何改进流水线结构以达到硬件开销和运行时间的相对平衡, 实现功耗和性能的折中。

目前普遍的解决方案是插入两组流水线寄存器, 分别位于每轮结束的输出端和迭代函数变换的步骤间。常规方法是在θ变换和ρ变换之间插入流水线寄存器[6,9], 如图 2(a); 或在π变换和χ变换之间插入流水线寄存器[12-13], 如图 2(b)。由于ρ变换和π变换只涉及移位操作, 没有逻辑门参与运算, 因此这两种方法在关键路径的改善上并无区别。但在本文提出的

结构中, 通过改变流水线寄存器的位置, 可以将关键路径进一步缩短, 使得整个流水线架构的性能大幅提升。

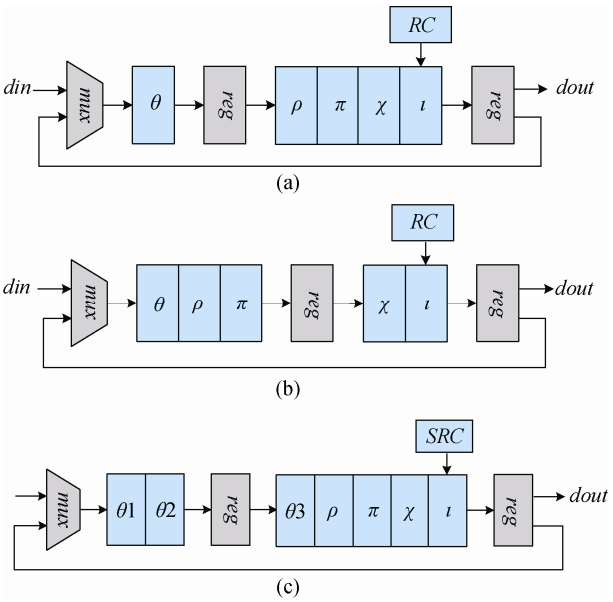


图 2　传统的流水线结构(a)(b)和改进后的流水线结构(c)

Figure 2　Conventional pipeline structure (a)(b) and proposed pipeline structure(c)

改进的流水线结构如图 2(c)所示, 本设计同样采用了两组流水线, 第二组寄存器仍然位于每轮结束的输出端, 第一组寄存器也安插在轮变换结构的内部。不同的是, 本设计将θ变换分为三个步骤(θ1, θ2, θ3), 并将该组流水线寄存器插入θ2 和θ3 之间, θ1~θ3 的变换如式(1)~(3)所示。从式(1)~(6)中可以看出, 每个步骤需要的逻辑门数量如下: (1) θ1 操作需要 4 个异或门; (2) θ2 操作需要 1 个异或门; (3) θ3 操作需要 1 个异或门; (4) ρ和π变换不需要逻辑门; (5) χ 变换需要 1 个异或门、1 个非门和 1 个与门; (6) ι变换需要一个异或门。因此, 如果把流水线寄存器插入θ变换和ρ变换之间或π变换和χ变换之间, 则关键路径上的运算要经过 6 个逻辑门, 而在本文提出的结构中, 二级流水的第一级包含θ1 和θ2 变换的 5 个异或门, 第二级由θ3、ρ、π、χ和ι变换中的 3 个异或门、1 个非门和 1 个与门构成。由此可见, 把流水线寄存器插入θ2 变换和θ3 变换之间, 则将关键路径上的逻辑门由 6 个缩减为 5 个。

具体结构如图3, 该电路将1600-bits的输入平均划分为 25 组, 每组 64-bits, 将其输入迭代运算模块参与θ、ρ、π、χ和ι运算, 产生 1600-bits 的输出, 再将其输出作为下一个循环的输入, 如此往复 24 轮, 得到最终的输出。在此过程中, 流水线寄存器插入在θ2 和θ3 之间以及每轮的输出端, 使得流水线的两个部分最长路径均经过 5 个逻辑门。值得注意的是, 与传统结构相比, 该设计在θ3 部分增加了 5 个 64-bits 寄存器, 以保证时序和运算结果的准确性。虽然增加寄存器带来了一些额外的硬件开销, 但其带来的运行速度上的优势是更加显著。

3.3　SHA-3 架构

提高吞吐量一般有如下两种方法, 展开和流水线结构。所谓展开是指在一个时钟周期内实现 Keccak-f[1600]的多轮迭代。比如原有一个时钟周期只能实现一次迭代运算, 因此执行 Keccak 算法的 24 轮迭代需要 24 个周期; 若经过展开之后每周期迭代运算的数量增加为两个, 则实现 24 轮的迭代只需要 12 个时钟周期。缺点是, 相应的资源消耗也增加了一倍。而上文所提到的流水线技术, 不仅可以在迭代轮内部增加流水线寄存器, 也可以将其插入展开后的迭代轮之间, 以提高运行效率。如何将这两种技术结合起来, 确定展开因子和流水线的组合方式, 以达到低功耗和高性能的折中, 是本设计中所解决的问题。

更高的展开因数和流水线级数可以带来吞吐量更大的提升, 但是同时会带来硬件资源开销的成倍提升, 使得硬件效率反而下降。经过评估和比较各种不同的硬件结构, 本设计选择了展开因数为 2 的硬件结构。此种结构既实现了高吞吐量, 同时保持了较低的资源开销, 能实现最大的硬件效率。其中两个内部流水线寄存器(inside pipeline registers, IPR)位于迭代运算单元内部, 两个输出流水线寄存器(output pipeline registers, OPR)位于相邻迭代运算单元之间。同时, 上文所讨论的简化的轮常数与改进的流水线结构在该结构中也有所体现。

如图 4 所示, SHA-3 的核心硬件结构由主要由控制单元和两个迭代运算单元组成, 每次轮转可以进行两轮迭代运算, 完成 24 次迭代运算仅需 12 次轮转。控制单元用于控制 SHA-3 的工作进程, 而每个迭代函数用于实现 Keccak-f[1600]中 θ, ρ, π, χ, ι 5 个运算步骤。除迭代函数 5 个步骤对应的硬件电路之外, 迭代运算单元内部在θ2 和θ3 之间插入了 IPR, 两个迭代运算单元的输出端也有 OPR。

工作过程如下:

(1) 输入: 1600-bits 数据通过 din 端输入二路复用器, 由控制单元决定并判断输入迭代运算单元 1 的数据来自 din 端或者二路输出选择器的反馈;

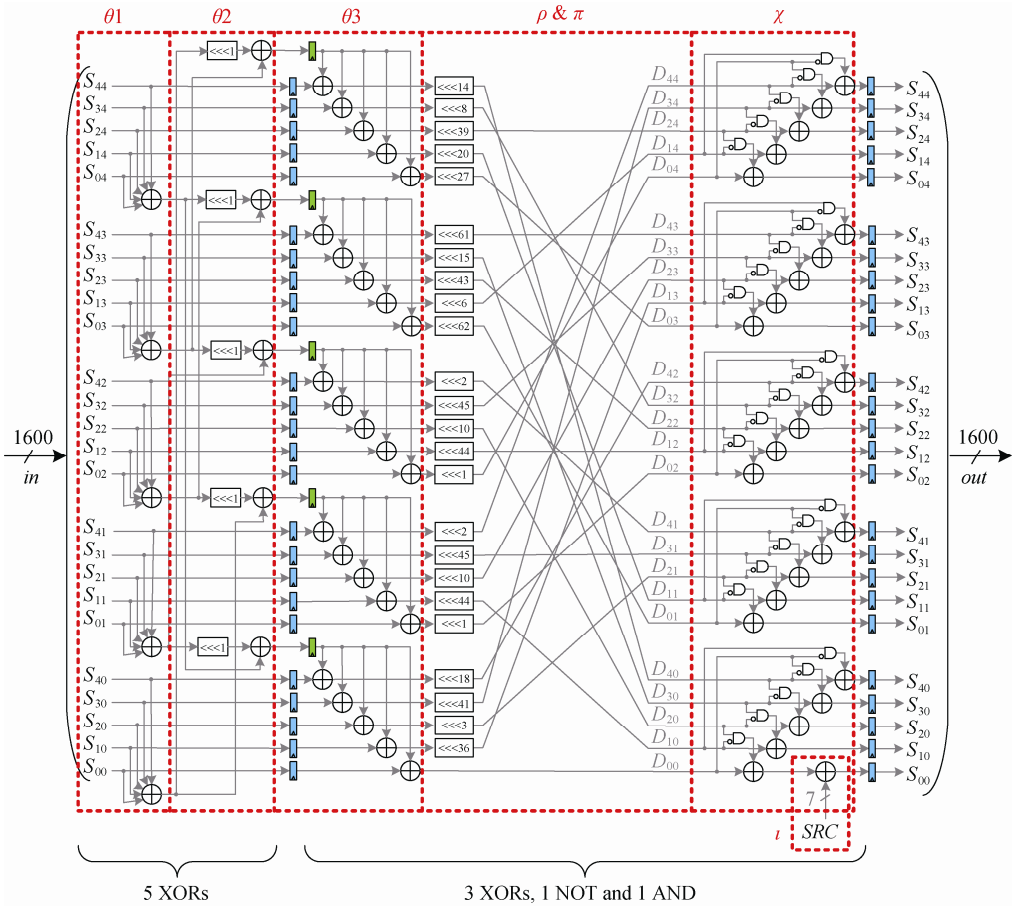


图 3　转换轮中的流水线寄存器

Figure 3　Pipeline Registers in the Transformation Round

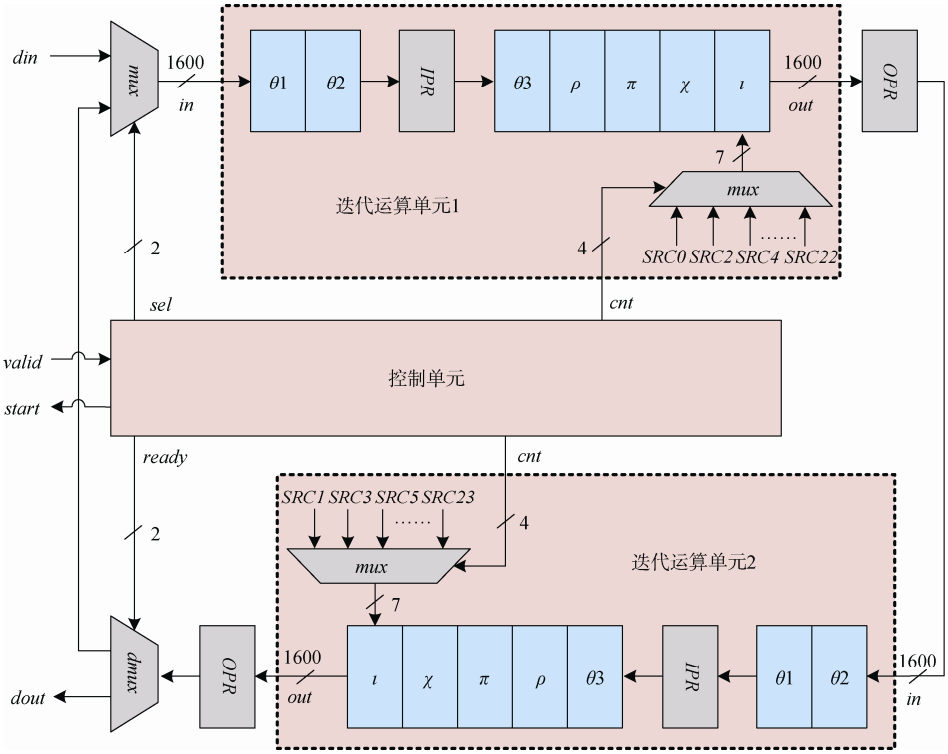


图 4　SHA-3 核心硬件结构

Figure 4　SHA-3 hardware architecture

(2) 迭代运算单元 1: 数据依次经过θ, ρ, π, χ, ι 5 个运算步骤, 其中参与ι运算的SRC为偶数轮的轮常数(SRC0, SRC2,…,SRC22), 控制单元根据轮数选择特定的轮常数通过十二路复用器输出, 在ι步骤中与S[0,0]道的数据相异或, 再从迭代运算单元中输出, 进入 OPR;

(3) 迭代运算单元 2: 经过 OPR 后数据进入迭代运算单元 2, 与迭代运算单元 1 不同的是 SRC 为基数轮的轮常数(SRC1, SRC3,…,SRC23), 其他步骤相同。输出之后经由 OPR 输入二路选择器;

(4) 循环及输出: 由控制单元根据时钟周期和轮数判断输出是否为最终输出, 若满足 24 次迭代运算, 则通过 dout 端输出, 若仍在迭代周期内, 则跳转至步骤(2), 继续循环直到满足条件。

4 实验结果及对比

本文中的设计采用 Verilog-HDL 编写, 环境是 Xilinx ISE 14.7 Design Suite, 在 Xilinx Virtex-6 的 XC6VLX75T-2ff784 FPGA 上完成了验证。

表 4 展示了本设计与其他相关文献中设计的指标参数对比, 吞吐量和效率的计算方法如式(7)(8)所示。其中 r 代表比特率(bitrate), 在 SHA3-512 中 r 等于 576, f 代表最大频率, N 代表同时处理的数量块数量, 此结构中为 4, *cycles* 代表运算周期数, 此结构中为 48, *area* 代表硬件资源消耗。

$$\text{Throughput}=\frac{r\times f\times N}{\text{cycles}} \tag{7}$$

$$\text{Efficiency}=\frac{\text{Throughput}}{\text{area}} \tag{8}$$

如表 4 所示, 本设计中的 SHA-3 硬件架构在 Virtex-6 FPGA 上最大工作频率达到了 459 MHz, 吞吐量达到了 22.03 Gbps。此外, 本设计共消耗硬件资源 1498 Slices, 实现了 14.71 Mbps/Slices 的高效率。

表 4 基于 SHA3-512 与相关工作的比较

Table 4 Comparison result of SHA3-512 with related works

文献	FPGA	最大频率(MHz)	资源(Slices)	吞吐量(Gbps)	效率(Mbps/Slices)
本设计	**Virtex-6**	**459**	**1498**	**22.03**	**14.71**
[7]	Virtex-6	397	1649	9.55	5.80
[14]	Virtex-6	310	1249	7.43	5.95
[15]	Virtex-6	414	1432	9.93	6.93
[16]	Virtex-6	391	2296	18.77	8.17
[17]	Virtex-6	392	4117	37.63	9.14
[9]	Virtex-6	344	1406	16.51	11.47

图 5 显示了与一些先前工作的比较结果。其中, Michail[17]在使用 4 个 OPR 展开 4 次时实现了最高吞吐量。M. M. Wong [9]在分别使用两个 OPR 和 IPR 并进行两次展开时达到最高效率。值得注意的是, 流水线级数和展开因子的增加使得吞吐量更高, 但资源消耗将相应增加。Michail[17]具有较高的展开因子和流水线级数, 但文中所达到的高吞吐量是以巨大的硬件开销为代价的。本设计的目标是提高吞吐量, 同时尽可能降低资源消耗, 因此没有选择更高的流水线级数或展开因子。本设计与 M. M. Wong [9]中的 2 个 IPR 和 2 个 OPR 具有相同的两次展开架构, 但由于简化轮常数的应用和新型的流水线技术, 提高了 33.4%的吞吐量和 28.2%的硬件效率。由此可见, 与以前的工作相比, 在包括最大频率、面积、吞吐量和效率在内的主要性能指标中, 本设计的频率和效率达到了最佳水平。

5 结论

本设计提出了一种应用于后量子密码的高速高效 SHA-3 硬件结构, 可显著提高运算速度和效率。本设计采用 7-bits 的简化轮常数代替 64-bits 的轮常数, 且通过改善流水线的结构来改善关键路径, 从而提升系统频率。同时, 本设计采用了二次展开的硬件结构, 并在运算过程中插入和每轮结束后插入流水线寄存器以提高效率。在 Xilinx Virtex-6 FPGA 上的验证表明, 本设计的最高工作频率为 459MHz, 效率可达 14.71 Mbps/Slice。与前人工作相比, 最高工作频率提高了 10.9%, 平均效率提高了约 28.2%。

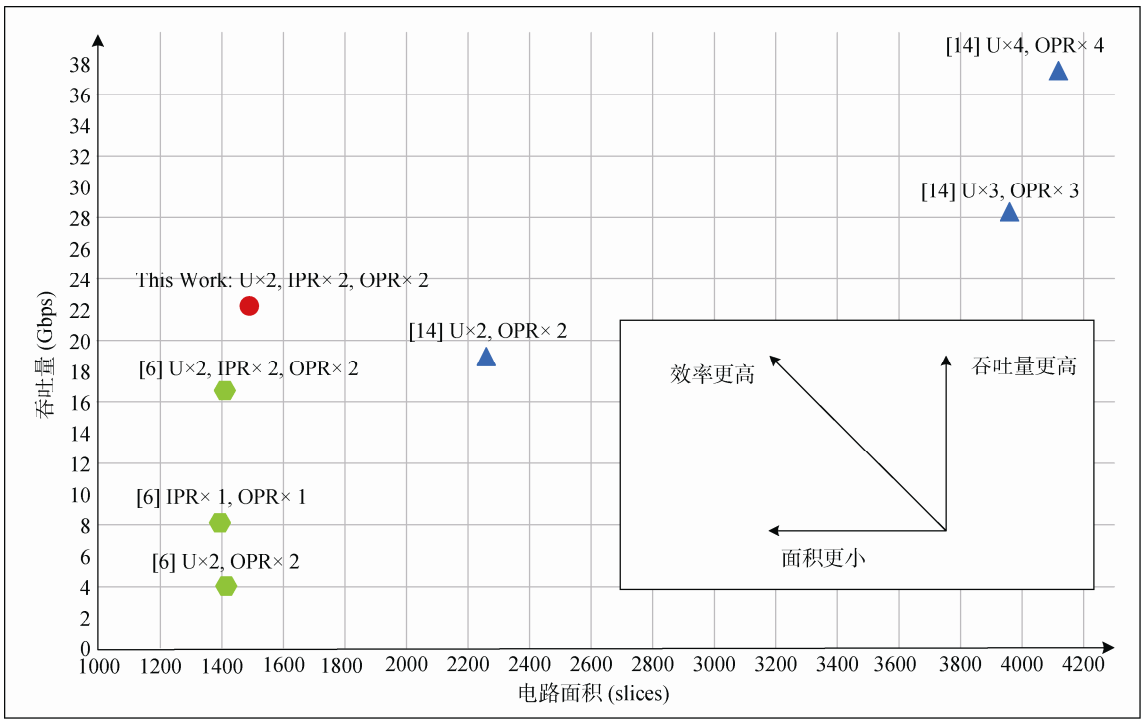


图 5　与前人相关工作的比较(U=展开系数, IPR=内部流水线寄存器数量, OPR=输出流水线寄存器数量)

Figure 5　Comparison of some previous works (U = the factor of unrolling, IPR = the number of inside pipeline registers, OPR = the number of output pipeline registers)

参考文献

[1] D. Moody, G. Alagic, et al. Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process. https://csrc.nist.gov/publications/detail/ nistir/8309/final. NIST, July. 2020.

[2] Sinha Roy S, Basso A. High-Speed Instruction-Set Coprocessor for Lattice-Based Key Encapsulation Mechanism: Saber In Hardware[J]. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020: 443-466.

[3] Bisheh-Niasar M, Azarderakhsh R, Mozaffari-Kermani M. High-Speed NTT-based Polynomial Multiplication Accelerator for CRYSTALS-Kyber Post-Quantum Cryptography. https://www.esearchgate.net/publication/351345144. May. 2021.

[4] Sundal M, Chaves R. Efficient FPGA Implementation of the SHA-3 Hash Function[C]. *2017 IEEE Computer Society Annual Symposium on VLSI*, 2017: 86-91.

[5] Jungk B, Stöttinger M. Serialized Lightweight SHA-3 FPGA Implementations[J]. *Microprocessors and Microsystems*, 2019, 71: 102857.

[6] Sideris A, Sanida T, Dasygenis M. High Throughput Pipelined Implementation of the SHA-3 Cryptoprocessor[C]. *2020 32nd International Conference on Microelectronics*, 2020: 1-4.

[7] Athanasiou G S, Makkas G P, Theodoridis G. High Throughput Pipelined FPGA Implementation of the New SHA-3 Cryptographic Hash Algorithm[C]. *2014 6th International Symposium on Communications*, *Control and Signal Processing*, 2014: 538-541.

[8] Akin A, Aysu A, Ulusel O C, et al. Efficient Hardware Implementations of High Throughput SHA-3 Candidates Keccak, Luffa and Blue Midnight Wish for Single- and Multi-Message Hashing[C]. *The 3rd international conference on Security of information and networks*, 2010: 168-177.

[9] Wong M M, Haj-Yahya J, Sau S, et al. A New High Throughput and Area Efficient SHA-3 Implementation[C]. *2018 IEEE International Symposium on Circuits and Systems*, 2018: 1-5.

[10] Dworkin M J. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions[R]. National Institute of Standards and Technology, 2015.

[11] E. Homsirikamol, M. Rogawski, and K. Gaj. Comparing Hardware Performance of Round 3 SHA-3 Candidates using Multiple Hardware Architectures in Xilinx and Altera FPGAs[C]. *Ecrypt II Hash Workshop*, 2011.

[12] Mestiri H, Kahri F, Bedoui M, et al. High Throughput Pipelined Hardware Implementation of the KECCAK Hash Function[C]. *2016 International Symposium on Signal*, *Image*, *Video and Communications*, 2016: 282-286.

[13] Kahri F, Mestiri H, Bouallegue B, et al. High Speed FPGA Implementation of Cryptographic KECCAK Hash Function Crypto-Processor[J]. *Journal of Circuits*, *Systems and Computers*, 2016, 25(4): 1650026.

[14] Gangwar P, Pandey N, Pandey R. Novel Control Unit Design for a High-Speed SHA-3 Architecture[C]. *2019 IEEE 62nd International Midwest Symposium on Circuits and Systems*, 2019: 904-907.

[15] El Moumni S, Fettach M, Tragha A. High Throughput Implementation of SHA3 Hash Algorithm on Field Programmable Gate Array (FPGA)[J]. *Microelectronics Journal*, 2019, 93: 104615.

[16] Ioannou L, Michail H E, Voyiatzis A G. High Performance Pipelined FPGA Implementation of the SHA-3 Hash Algorithm[C].

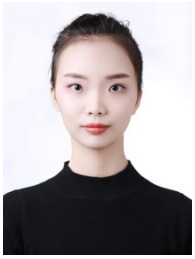
2015 4th Mediterranean Conference on Embedded Computing, 2015: 68-71.

[17] Michail H E, Ioannou L, Voyiatzis A G. Pipelined SHA-3 Implementations on FPGA: Architecture and Performance Analysis[C]. *The Second Workshop on Cryptography and Security in Computing Systems*, 2015: 13–18.

刘冬生 于 2007 年在华中科技大学微电子学与固体电子学专业获得博士学位。现任华中科技大学光电学院教授，博士生导师。研究领域为 CMOS 集成电路设计。研究兴趣包括：高能效芯片设计、硬件安全及安全芯片设计、SoC 及软硬件协同设计等。Email：dsliu@hust.edu.cn

杨朔 于 2021 年在华中科技大学集成电路设计与集成系统专业获得学士学位。现在华中科技大学电子科学与技术专业攻读硕士研究生学位。研究领域为硬件安全及安全芯片设计。Email：wxkhturf@163.com

胡昂 于 2020 年在华中科技大学微电子学与固体电子学专业获得工学博士学位，现在华中科技大学计算机科学与技术学院开展博士后研究工作。研究领域为集成电路设计，研究兴趣包括：低功耗无线收发器芯片、全数字锁相环等。Email：ang_hu@hust.edu.cn



熊思琦 现在华中科技大学集成电路设计 与集成系统专业攻读学士学位。研究领域为硬件安全及安全芯片设计。Email：xiong_si_qi@foxmail.com

陈勇 于 2019 年在南京理工大学光电信息科学与工程专业获得学士学位。现在华中科技大学软件工程专业攻读硕士研究生学位。研究领域为硬件安全及安全芯片设计。Email：yorkchen576@qq.com