·网络空间安全· 文章编号：1000-3428(2021)12-0147-09 文献标志码：A 中图分类号：TP309

# 基于OpenCL的3DES算法FPGA加速器

吴健凤[1]，郑博文[2]，聂 一[2]，柴志雷[1,3]
（1.江南大学 人工智能与计算机学院，江苏 无锡 214122；2.江南大学 物联网工程学院，江苏 无锡 214122；
3.数学工程与先进计算国家重点实验室，江苏 无锡 214215）

**摘 要：**在数字货币、区块链、云端数据加密等领域，传统以软件方式运行的数据加解密存在计算速度慢、占用主机资源、功耗高等问题，而以Verilog/VHDL等方式实现的现场可编程门阵列(FPGA)加解密系统又存在开发周期长、维护升级困难等问题。针对3DES算法，提出一种基于OpenCL的FPGA加速器设计方案。设计具有48轮迭代的流水并行结构，在数据传输模块中采用数据存储调整、数据位宽改进策略提高内核实际带宽利用率，在算法加密模块中采用指令流优化策略形成流水线并行架构，同时采用内核矢量化、计算单元复制策略进一步提高内核性能。实验结果表明，该加速器在Intel Stratix 10 GX2800上可获得111.801 Gb/s的吞吐率，与Intel Core i7-9700 CPU相比性能提升372倍，能效提升644倍，与NvidiaGeForce GTX 1080Ti GPU相比性能提升20%，能效提升9倍。

**关键词：**OpenCL框架；现场可编程门阵列；加解密算法；3DES算法；流水并行结构

**开放科学（资源服务）标志码（OSID）：**

**中文引用格式：**吴健凤，郑博文，聂一，等.基于OpenCL的3DES算法FPGA加速器[J].计算机工程，2021，47(12)：147-155，162.

**英文引用格式：**WU J F，ZHENG B W，NIE Y，et al.FPGA accelerator for 3DES algorithm based on OpenCL[J].Computer Engineering，2021，47(12)：147-155，162.

# FPGA Accelerator for 3DES Algorithm Based on OpenCL

WU Jianfeng[1]，ZHENG Bowen[2]，NIE Yi[2]，CHAI Zhilei[1,3]
（1.School of Artificial Intelligence and Computer Science，Jiangnan University，Wuxi，Jiangsu 214122，China；
2.School of Internet of Things Engineering，Jiangnan University，Wuxi，Jiangsu 214122，China；
3.State Key Laboratory of Mathematical Engineering and Advanced Computing，Wuxi，Jiangsu 214215，China）

**【Abstract】**Nowadays，encryption and decryption algorithms are widely used in digital currency，blockchain，cloud data encryption and other fields.Traditional software-based data encryption is limited in the calculation speed while occupying many host resources and having high power consumption.Also，Field Programmable Gate Array（FPGA）-based encryption and decryption implemented in Verilog/VHDL suffer from the long development cycles and difficult maintenance and upgrades. To address the above problems，a design scheme of a FPGA accelerator for 3DES algorithm based on OpenCL is proposed. In the scheme，a pipeline parallel structure with 48 iterations is designed by adjusting data storage，improving data bit width，optimizing instruction stream，vectorising Kernels and replicating compute units.For the data transmission module，the actual bandwidth utilization of the Kernel is improved by adjusting data storage and increasing data bit width.For the algorithm encryption module，the instruction stream is optimized to form a pipeline parallel architecture.In addition，the performance of the Kernel is further improved by kernel vectorization and compute unit replication strategies.The experimental results show that the accelerator provides a throughput rate of 111.801 Gb/s on Intel Stratix 10 GX2800.Compared with the Intel Core i7-9700 CPU，the proposed accelerator improves the performance by 372 times and the energy efficiency by 644 times. Compared with the Nvidia GeForce GTX 1080Ti GPU，the proposed accelerator improves the performance by 20% and the energy efficiency by 9 times.

**【Key words】**OpenCL framework；Field Programmable Gate Array（FPGA）；encryption and decryption algorithm；3DES algorithm；pipeline parallel structure

**DOI：**10. 19678/j. issn. 1000-3428. 0059799

**基金项目：**国家自然科学基金（61972180）；数学工程与先进计算国家重点实验室开放基金（2018A04）。

**作者简介：**吴健凤（1996—），女，硕士研究生，主研方向为信息安全、计算机体系结构；郑博文、聂 一，硕士研究生；柴志雷，教授、博士。

**收稿日期：**2020-10-20 **修回日期：**2020-12-07 **E-mail：**zlchai@jiangnan.edu.cn

## 0 概述

将加解密算法用于数字货币、区块链、云端数据加密等领域时[1-2]，算法应具备高强度计算能力。因此，当前服务器端允许包含异构计算平台以增强特定工作负载的性能，同时改善整个系统的维护成本。OpenCL是异构平台的开放框架，其中内核（Kernel）程序可以在多核CPU上，同时也可以在GPU、现场可编程门阵列（Field Programmable Gate Array，FPGA）、DSP上编译执行[3]。当前服务器端除了采用ASIC或GPU处理大批量数据外，考虑到能效等因素也会大规模部署FPGA。

目前，已有许多基于OpenCL的加解密算法加速器的研究。文献[4]设计一种基于OpenCL的MD5算法加速器架构，提出结合优化CPU端内存分配、复制内核计算单元的方法，其较未经优化方法性能提高了6.1倍。文献[5]提出基于OpenCL内核的Kuznyechik算法流水线架构设计，结合轮密钥按位模2加消除依赖、查找表、分解线性变换为布尔函数等方法，在Intel Arria 10系列实现了41 Gb/s的吞吐率，同时占用不超过10%的FPGA资源。文献[6]评估了可扩展的多FPGA架构上AES加密内核的性能，在带有Stratix A3 FPGA的单个M506模块上，通过使用OpenCL SIMD4达到了较高的吞吐率。文献[7]结合OpenCL工具设计SHA1哈希算法，与基于硬件的描述语言相比，使用Intel FPGA SDK工具进行少量的内核代码更改即可实现电路的改变，有效节省了系统开发时间，其结合循环展开、循环流水等策略，在Altera Stratix V系列器件上达到3 033 Mb/s的吞吐率，相比于CPU性能提升了14倍。

DES算法是1972年美国IBM公司设计的对称密码体制加密算法。随着软硬件的快速发展，DES算法已被证实不够安全。为了克服DES算法的缺陷，1999年美国NIST发布了新版本的DES标准[8]，指出DES仅能用于遗留的系统，同时3DES将取代DES成为新的标准。在国内，中国人民银行的智能卡技术规范已支持3DES[9]，电子支付系统将3DES方案用于数据的加解密[2]。目前3DES算法在国内外有着广泛的应用，因此，更高效地实现3DES加密具有重要意义。

现有针对3DES算法的优化方法以使用硬件描述语言（Hardware Description Language，HDL）居多[10-11]。虽然HDL仍是在FPGA上开发时序关键型设计的合适选择，但在HDL中开发应用程序需要付出较大代价并且容易出错。高层次综合工具（High-Level Synthesis，HLS）是高性能计算界针对FPGA编程的替代解决方案[12]。使用HLS可以让几乎没有FPGA开发经验的用户充分利用FPGA的优势。目前，有2种支持利用FPGA开发OpenCL应用的商业编译器，一个是用于OpenCL的Intel FPGA软件开发套件（Software Development Kit，SDK），支持Cyclone、Stratix和Arria系列的FPGA平台[13-15]，另一个是Xilinx，为基于OpenCL的Kintex系列和Virtex-7 FPGA产品提供了完整的SDAccel开发环境[16]。

本文采用数据存储调整、数据位宽改进、指令流优化、内核矢量化、计算单元复制等策略，设计并实现一种基于OpenCL的加解密算法FPGA加速器架构，以应用广泛的3DES算法为例，介绍内核程序的设计过程。

## 1 3DES算法原理

3DES算法以DES算法为基础，其通过进行3次DES加密增强算法复杂性，从而保障安全性[17]。DES算法包含16轮迭代，使用56 bit密钥，而3DES算法包含48轮迭代，使用168 bit密钥。

### 1.1 DES算法

DES算法将明文按64 bit进行分组，参与计算的密钥长度固定为64 bit（有效位数56 bit）。加密流程主要包括初始置换、16轮循环迭代、逆初始置换3个部分。其中，16轮迭代过程中参与计算的子密钥由56 bit密钥扩展而来[8]。

DES算法流程如图1所示，其中，64 bit的输入明文经初始置换分为$L_0$和$R_0$两部分，然后进行16轮相同的迭代运算，最后经过逆初始置换得到64 bit的输出密文。在每一轮迭代中，包含一次异或运算和f函数运算。

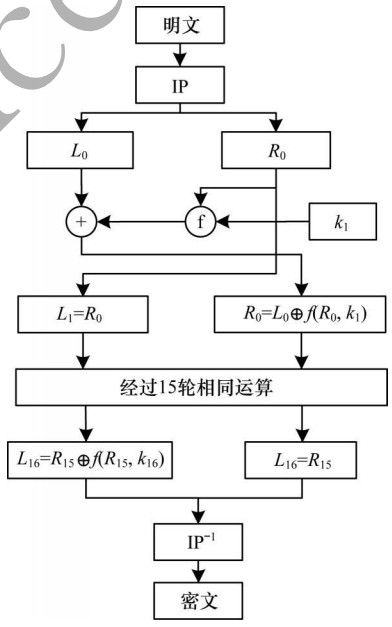



**图1 DES算法流程**

**Fig.1 Procedure of DES algorithm**

f函数原理如图2所示，其中，输入为该轮的$R_{i-1}$和该轮对应的子密钥$k_i$，$R_{i-1}$经E盒扩展后的结果与48 bit的子密钥$k_i$进行异或运算，之后经S盒变换、P盒置换得到32 bit的输出。S盒将6 bit的输入转换

为4 bit的输出，这也是算法中唯一的非线性变换，极大地提高了算法的安全性。S盒的置换部分使用查找表实现，其将8个S盒的内容存储于片上ROM，有效提高了算法的计算效率。DES算法的子密钥生成模块输入为56 bit的密钥，经16轮迭代生成16个子密钥$k_i$，分别用于DES算法的16轮迭代计算模块。在子密钥生成模块中，每轮迭代包含循环左移和密钥置换操作。

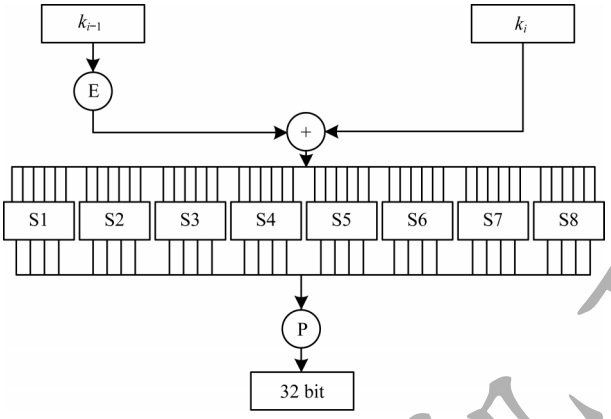



**图2　f函数原理示意图**

**Fig.2　Schematic diagram of f function principle**

### 1.2　3DES算法

3DES算法在DES算法的基础上发展而来，其输入为64 bit的明文，输出为64 bit的密文。与DES算法不同，3DES算法包含192 bit（有效长度168 bit）的密钥。令$E_{k_i}(I)$和$D_{k_i}(I)$分别表示使用DES密钥$k_i$对数据块$I$的DES加密和解密运算。3DES的加密操作如式（1）所示，其将64 bit的输入块$I$转换成64 bit的输出块$O_1$。3DES的解密操作如式（2）所示，其将64 bit的输入块$I$转换成64 bit的输出块$O_2$。

$$O_1 = E_{k_3}(D_{k_2}(E_{k_1}(I))) \tag{1}$$

$$O_2 = D_{k_1}(E_{k_2}(D_{k_3}(I))) \tag{2}$$

## 2　基于OpenCL的FPGA设计

OpenCL为开发人员提供了抽象的内存层次结构以生成有效代码，适合目标设备的内存层次结构。OpenCL内存结构由全局内存（Global Memory）、常量内存（Constant Memory）、局部内存（Local Memory）、私有内存（Private Memroy）4种类型构成[18]。工作项在处理单元（Processing Element，PE）上运行，可以访问对应的私有内存，工作组在一个计算单元（Compute Unit，CU）上运行。同一工作组中的工作项拥有共同的局部内存。

OpenCL程序包含主机端程序和内核程序两部分。如图3所示，内核程序运行时系统会创建一个整数索引空间，工作项对应执行索引空间的一个实例，工作组是工作项的集合，同一工作组中的工作项共享内存并可以实现组内同步。工作项在全局索引空间中的坐标为该工作项的全局ID，工作项在工作组中的坐标为该工作项的局部ID。

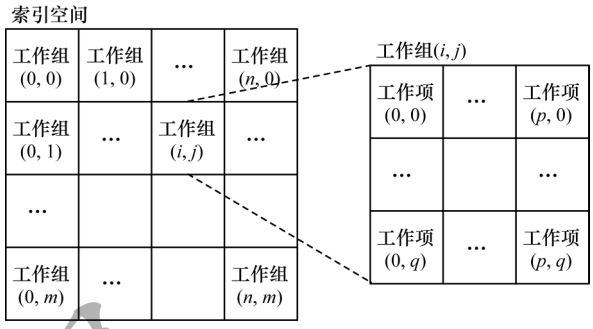



**图3　OpenCL索引空间示意图**

**Fig.3　Schematic diagram of OpenCL index space**

本文采用IntelFPGA SDK实现3DES算法加速器设计。在FPGA上构建系统之前，SDK支持在CPU上仿真OpenCL应用程序。软件仿真利用CPU模拟FPGA硬件特性，通常用于功能验证。目前，Intel的工具链不支持硬件仿真[16]。Intel FPGA SDK包含一个编译OpenCL内核以创建优化硬件镜像的离线编译器，该编译器将内核代码转换成为中间Verilog形式，然后通过Quartus II软件将其编译为二进制镜像，该镜像可在程序运行时加载至FPGA端。由于编译过程需要数小时来应用适当的优化并设计出硬件镜像，因此编译过程是离线的，主机程序仅在运行时才加载硬件镜像。构建完成后，将创建主机可执行文件和二进制文件，以在FPGA上运行目标程序[19]。

## 3　基于OpenCL的3DES算法FPGA加速器

本文基于OpenCL实现3DES算法FPGA加速器的设计，包含主机端（HOST）程序设计与设备端程序设计两部分：主机端程序结合3DES算法加密的原理，完成主机端程序对明文数据的读取、初始化、存储、OpenCL运行时环境的创建以及对Kernel的调度与管理；设备端程序设计针对3DES算法内核计算模块进行优化并形成流水线并行架构。同时，采用数据存储调整、数据位宽改进策略有效提升实际带宽利用率，采用指令流优化技术针对算法中的循环迭代进行改进，提高计算的并行度，采用内核矢量化、计算单元复制进一步提升内核性能。

### 3.1　主机端程序设计

主机端完成明文数据的读取、初始化、存储、内核调度、管理等工作。Intel FPGA SDK提供了OpenCL平台API及运行时API：平台API定义了主机端程序发现OpenCL设备所用的函数以及这些函数的功能；运行时API用于管理上下文来创建命令队列以及运行时发生的其他操作。通过调用OpenCL API可实现主机端对内核的调度与管理[17]，CPU端程序流程如图4所示。

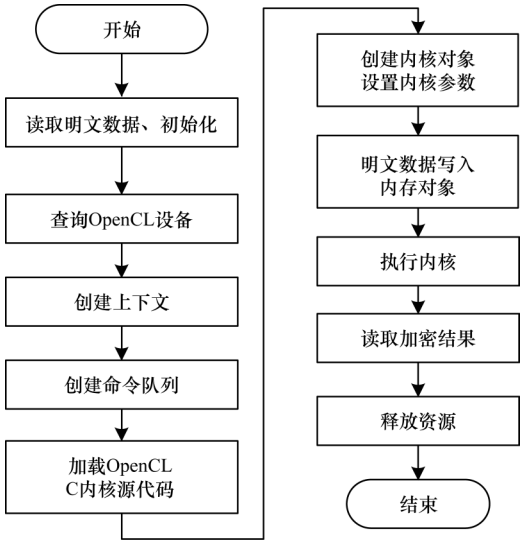



图4 主机端程序流程

Fig.4 Procedure of HOST program

### 3.2 优化策略

3DES算法内核模块包括明文数据输入缓存、算法加密模块和密文数据输出缓存3个模块，如图5所示。明文数据输入缓存完成从全局内存读取明文数据，通过使用数据存储调整、数据位宽改进提高实际带宽利用率；算法加密模块基于FPGA完成3DES算法的加密计算，通过数据循环展开、循环流水形成流水线并行计算架构；密文数据输出缓存模块将数据从FPGA片上传输至外部DDR中。

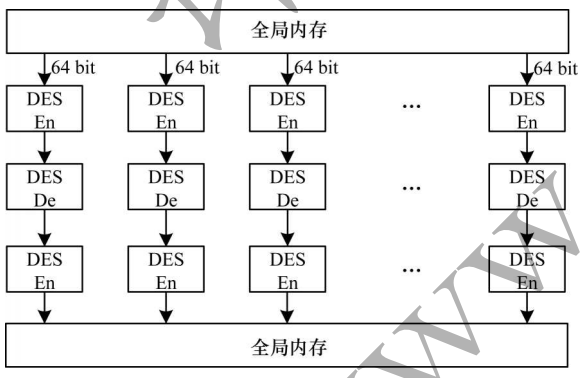



图5 内核模块示意图

Fig.5 Schematic diagram of Kernel module

#### 3.2.1 数据存储调整

由主机端传输的数据存储于片外DDR中，对应的数据类型为__global；常量内存位于片上缓存单元，对应的数据类型为__constant；局部内存的物理地址为片上RAM资源，对应的数据类型为__local；私有内存的物理地址为片上寄存器资源，对应的数据类型为__private。由于片上不同资源的大小、延迟、吞吐率存在差异，因此合理分配数据存储位置对于算法性能提升有较大的影响[19]。

不同内存类型的性能参数如表1所示。全局内存类型具有最大的吞吐率及容量，但同时也存在较大的访存延迟，主机端传输的数据存储于全局内存中，因此提高内存带宽的实际利用率对于系统性能的提升是有效的。局部内存对工作组中的所有工作项可见，与私有内存相比，在访存延迟相当的情况下，其具有更高的吞吐率及更大的容量，但同一工作组中的工作项执行后需要通过使用屏障保证数据一致性，这在一定程度上增加了延迟。因此，将参与3DES计算的变量存储于私有内存中，工作项访问位于私有内存中相应的明文数据块并完成3DES算法加密。针对f函数计算模块的S盒和E盒变换，由于是频繁访问的数据且其值在计算过程中保持不变，因此将其存储于常量内存，对应的物理地址为片上ROM，从而在加快访问速度的同时避免访存冲突。

表1 不同内存类型的性能参数

Table 1 Performance parameters of different memory types

| 内存类型 | 延时/cycle | 吞吐量/(GB·s$^{-1}$) | 容量/MB |
|---|---|---|---|
| 全局内存 | 240 | 34.133 | 8 000 |
| 局部内存 | 2 | ~8 000 | 66 |
| 私有内存 | 2/1 | ~240 | 0.2 |

#### 3.2.2 数据位宽改进

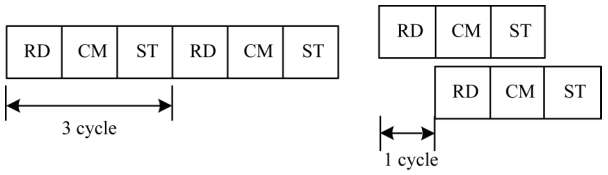
工作项执行内核程序的一个实例，如果工作项处理的数据位宽不固定，则编译器会使用更多的资源以满足可能的数据位宽，但同时也会对程序的优化编译有所限制。

基于3DES算法输入数据长度为64 bit且输出数据长度为64 bit的前提，将单工作项处理的数据长度调整为8 Byte。若将数据长度调整为4 Byte，则需要2个工作项来完成一个明文块的加密操作，此时工作项间的数据需要同步以保证数据一致性，这会增加额外的时间开销；若将数据长度调整为16 Byte，此时单工作项的处理数据量为原来的1倍，理论上内核执行时间增加1倍，则将单工作项的行为定义为从全局内存中搬运8 Byte的数据至私有内存，针对8 Byte数据进行3DES加密计算，再将计算的结果从私有内存搬运至全局内存中。通过获得工作项的全局ID可实现工作项与明文数据的一一对应，从而避免工作项间的同步操作。

#### 3.2.3 指令流优化

指令流优化主要使用循环展开和循环流水来提高程序的并行度。循环展开可指导离线编译器将OpenCL Kernel转换为硬件镜像的方式。通过使用

循环展开可形成有效的流水,而流水线架构能够缩短整体的执行时间。如图6所示,假设每步操作需要1个时钟周期,未形成流水线型设计时,内核在下次计算时延迟了3个时钟周期,而在使用流水线型设计后,只延迟了1个时钟周期。




(a)未形成流水线架构的情况　(b)形成流水线型架构的情况

**图6　内核计算架构**

**Fig.6　Architecture of Kernel computing**

循环展开在消耗一定硬件资源的前提下可降低数据读取与存储的次数,节省计算所需的时间,形成流水线型架构,从而提高并行度。如表2所示,在内存数据Load模块,针对8 Byte的明文数据与3个8 Byte的子密钥数据读取进行循环展开形成内存合并,将对内存数据的32次Load操作降低为4次更宽的Load操作;在内存数据Store模块,针对8 Byte的密文数据存储进行循环展开形成内存合并,将使内存的8次Store操作减少为1次更宽的Store操作;在迭代计算模块,针对子密钥生成模块的16次循环左移和密钥置换、DES计算模块的16轮轮换计算模块进行循环展开,指导编译器生成多套单次迭代所需的硬件结构,节省了迭代计算所需的时间。表2数据显示,未进行循环展开时,内核的执行时间为1 110.096 ms,使用循环展开后,内核的执行时间降低为46.620 ms,可见通过循环展开取得了较好的优化效果。

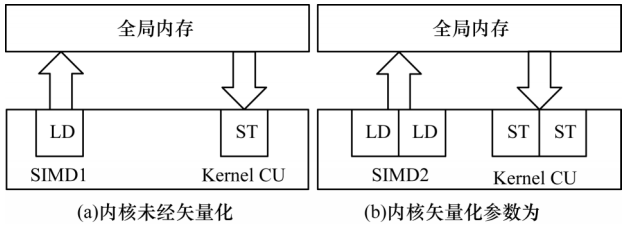
**表2　循环展开的优化效果**

**Table 2　Optimization effect of loop unrolling**

| 方案 | 内存Load次数 | 内存Store次数 | 迭代计算次数 | 时间/ms |
|---|---|---|---|---|
| 未循环展开 | 32 | 8 | 32 | 1 110.096 |
| 循环展开 | 4 | 1 | 2 | 46.620 |

### 3.2.4　内核矢量化

内核矢量化允许多个工作项以SIMD的方式执行内核程序的实例。矢量化指导编译器生成多个矢量通道,使得工作项可以同时存取并处理多个数据[20]。如图7所示,内核矢量化参数设定为2后编译器会合并内存访问。与未经矢量化相比,矢量化后的内核单次Load和Store的数据量为原先的2倍。使用内核矢量化时,需要同时指定工作组大小,且内核矢量化的参数能被工作组大小整除。内核矢量化的参数只能是2的指数,由于硬件资源的限制,因此实验中可设定的最大矢量化参数为16。




(a)内核未经矢量化　(b)内核矢量化参数为

**图7　内核矢量化示意图**

**Fig.7　Schematic diagram of kernel vectorization**

以加密128 MB数据量为例描述不同内核矢量化参数下系统内存带宽及吞吐率的变化情况。在SIMD2方案中,设置矢量化参数为2后工作组中的工作项平均分布在2个SIMD通道中,此时单工作项执行的工作量为原来的2倍,同时编译器会合并内存访问,单工作项一次可从内存中加载2个明文数据块进行加密并一次将2个数据块的加密结果存储到全局内存中[19]。如表3所示,随着矢量化参数的增加,内核的执行时间得到降低,内存带宽及系统的吞吐率得到提升。

**表3　内核矢量化的优化效果**

**Table 3　Optimization effect of Kernel vectorization**

| 方案 | 内存带宽/(MB·s$^{-1}$) | 时间/ms | 吞吐率/(Gb·s$^{-1}$) |
|---|---|---|---|
| SIMD1 | 5 790.1 | 46.577 | 21.985 |
| SIMD2 | 10 522.0 | 25.629 | 39.955 |
| SIMD4 | 17 910.7 | 14.961 | 68.445 |
| SIMD8 | 23 506.6 | 11.112 | 92.153 |
| SIMD16 | 27 592.7 | 9.423 | 108.670 |

在本文设计中,工作组大小为512,内核矢量化参数为16,每个工作组中的工作项分布在16个SIMD通道中。编译器实现16个SIMD通道后,每一个工作项的计算工作量为原先的16倍,相应的全局工作组大小减少为原来的1/16。

### 3.2.5　计算单元复制

通过计算单元复制策略可提高具有常规内存访问模式的内核性能。Intel FPGA SDK编译器支持为内核生成多个计算单元,通常每个计算单元可以同时执行多个工作组,从而提高内核的吞吐率。使用计算单元复制后,FPGA中的硬件调度器将工作组分派到其他可用的计算单元。只要计算单元尚未达到其最大容量,就可以将其用于工作组分配[19]。

如图8所示,本结合参数为16的内核矢量化策略,利用FPGA硬件调度器将工作组分配至2个计算单元中执行,理论上可使内核的运行时间缩短为原来的一半。然而,虽然通过使用多个计算单元可提高系统的吞吐率,但也会增加对于全局内存带宽的竞争以及硬件资源的使用。

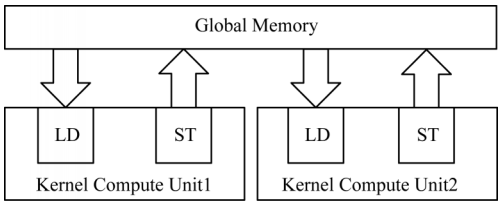



图8 计算单元复制策略示意图

Fig.8 Schematic diagram of compute unit replication strategy

### 3.3 加速器总体架构设计

如图9所示,基于OpenCL的3DES算法加速器架构由主机(HOST)端和设备端2个部分组成,其中,HOST端负责与OpenCL程序外部环境的数据交互、与设备端的数据交互及Kernel的调度与管理,设备端负责3DES算法的计算任务。

设备端包含明文数据输入缓存、3DES算法加密计算、密文数据输出缓存3个模块。其中,明文数据输入缓存、密文数据输出缓存位于设备端的全局内存区域,3DES算法加密的中间数据存储于设备端的私有内存区域。

在图9中,PE单元为一个工作项的处理单元,每个PE单元拥有相应的私有内存用于存储运算的中间数据,一个PE单元完成一个明文块的3DES加密计算。针对全局内存与私有内存的数据传输模块,结合OpenCL内存模型及全局内存、私有内存存在访存差异的特点,采用改进数据存储位置、调整数据位宽策略提高内核实际带宽利用率;针对算法加密计算模块,结合3DES算法加密的原理,采用循环展开、循环流水策略形成流水线并行架构,同时结合使用内核矢量化策略形成更宽的矢量计算通道从而有效提升算法的性能,采用计算单元复制策略进一步提高FPGA端计算的吞吐率。

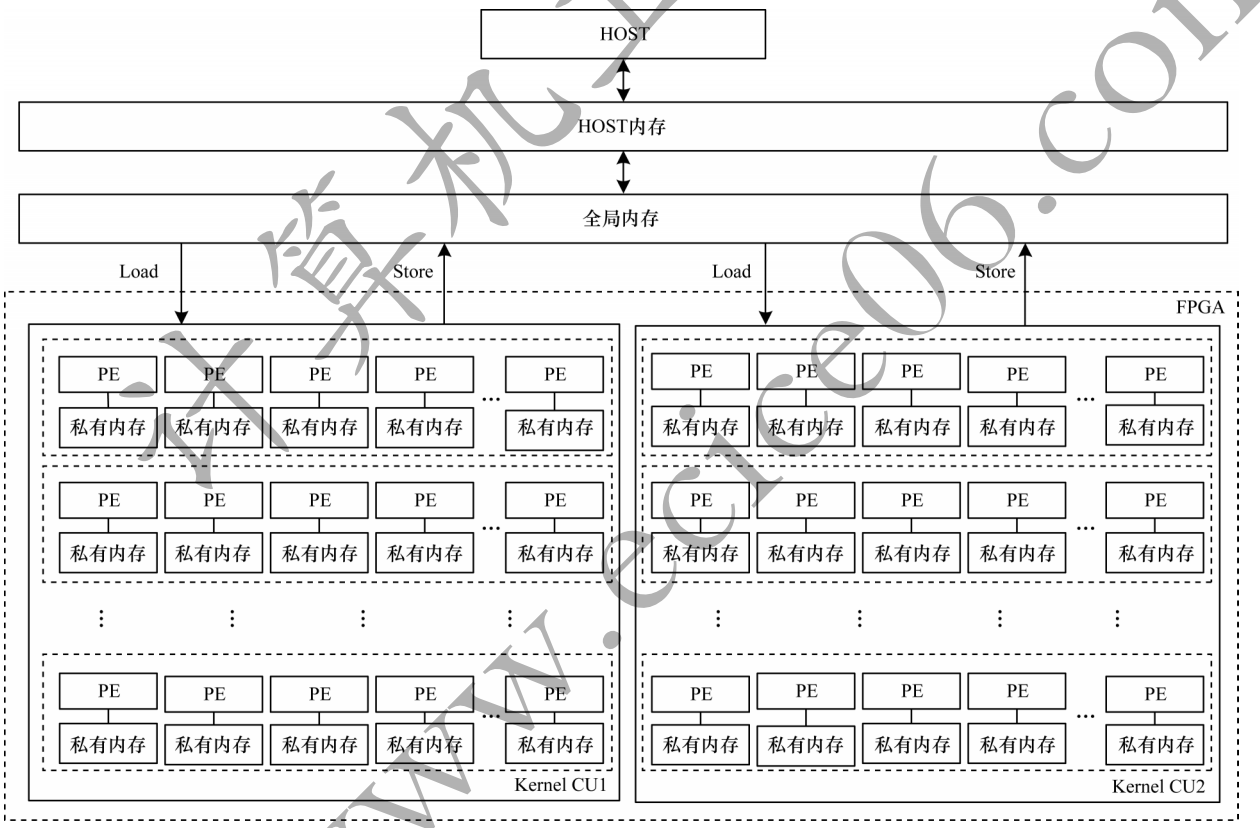



图9 3DES算法FPGA加速器总体架构

Fig.9 The overall architecture of FPGA accelerator for 3DES algorithm

## 4 实验

### 4.1 实验环境

对本文设计的加速器进行实验验证,软件环境为CentOS Linux release 7.7.1908+GCC V4.8.5,OpenCL版本为Intel FPGA SDK for OpenCL 19.3,硬件组合为Intel Xeon E5-2650 V2的CPU+Intel Stratix 10 GX2800的FPGA,该款FPGA包含1 866 240个ALUT,内存带宽为34 GB/s,资源情况如表4所示[14]。

表4 FPGA端资源情况

Table 4 FPGA resources

| 资源 | 数量 |
|---|---|
| ALUT | 1 866 240 |
| FF | 3 732 480 |
| RAM | 11 721 |
| DSP | 5 760 |

### 4.2　实验结果

在不同优化策略下，以加密128 MB数据、单工作项处理8 Byte明文块为例，结合FPGA端内存带宽、工作频率、内核执行时间及资源消耗情况描述内核的性能变化。内存带宽及时钟频率通过在编译器编译时加入性能计数器（-profile）获得，通过aocl report指令调用Intel FPGA Dynamic Profiler for OpenCL工具获得内存带宽及工作频率的信息，通过clGetEventProfilingInfo函数获得内核的执行时间，通过aoc -rtl指令生成内核的分析报告，获得资源消耗的详细信息。实验中记录的时间是算法的绝对执行时间，不包含主机与设备之间的数据传输时间，时间的统计结果通过多次测试取平均值获得。

### 4.3　实验结果分析

表5展示了不同优化策略下FPGA端内存带宽、时钟、内核执行时间及资源占用变化情况。，其中，未优化的内核其内存带宽为1 916.9 MB/s且FPGA板卡的工作频率为306.2 MHz，内核运行时间为1 349.181 ms。下文对不同方案下的实验数据进行分析。

**表5　不同优化方案下FPGA端性能及资源情况**

**Table 5　FPGA performance and resources under different optimization schemes**

| 方案 | 内存带宽/(MB·s$^{-1}$) | 时钟/MHz | ALUTs/% | FFs/% | RAMs/% | 时间/ms |
|---|---|---|---|---|---|---|
| 未优化 | 1 916.9 | 306.2 | 7.30 | 5.43 | 1.00 | 1 349.181 |
| 数据存储调整 | 2 277.8 | 325.0 | 8.01 | 5.53 | 15.64 | 1 306.545 |
| 数据位宽改进 | 4 710.7 | 322.5 | 1.91 | 1.38 | 10.01 | 1 354.537 |
| 指令流优化 | 5 779.7 | 372.2 | 3.30 | 1.40 | 2.56 | 46.620 |
| SIMD8 | 23 274.5 | 373.3 | 19.41 | 6.27 | 2.07 | 11.132 |
| SIMD16 | 27 534.1 | 372.2 | 38.57 | 12.32 | 2.42 | 9.425 |
| SIMD8+CU4 | 27 105.5 | 366.7 | 77.33 | 24.97 | 6.65 | 9.409 |
| SIMD16+CU2 | 28 102.3 | 366.7 | 77.03 | 24.59 | 4.28 | 9.243 |

数据存储调整结合OpenCL内存模型及FPGA板卡不同硬件资源存在访存延迟及吞吐率差异的特点，将HOST端传输的数据存储类型由常量类型（__constant）更改为全局变量类型（__global），将内存的实际带宽利用率、板卡的工作频率提升至325 MHz。这是因为全局类型的变量其存储位置为FPGA的片外DDR，且理论带宽可达到34 GB/s，而常量类型变量在内核运行时会自动由DDR转存至FPGA片上缓存，这在一定程度上增加了额外的数据移动。

数据位宽改进将单工作项处理的数据位宽确定为8 Byte，此时内存带宽提升了一倍多，这是因为在数据位宽确定的情况下，编译器能够结合长度信息做更好的优化，且使用8 Byte长度的位宽可最大化单工作项处理数据的效率，同时减少不必要资源的使用。结合数据位宽改进后，设备端占用的逻辑资源大幅减少，由最初的8.01%降至1.91%。

指令流优化包含循环展开和循环流水两部分。使用指令流优化后，系统整体性能的提升较为明显。由表5可知，使用指令流优化后内存带宽提升至5 779.7 MB/s，时钟频率达到372.2 MHz，同时内核的运行时间由最初的1 349.181 ms缩短至46.620 ms。这是因为在结合数据存储调整、数据位宽改进的前提下，采用循环展开和循环流水策略可以达到较好的并行度。实验结果表明，在未结合数据位宽改进的情况下，采用循环展开循环流水策略后内核的运行时间为437.053 ms。这是因为如果未结合数据位宽改进，编译器无法针对循环做有效的展开，且会尽量使用更多的资源和更粗粒度的优化来满足可能的数据长度，这在一定程度上限制了编译器的优化能力。由表5可知，使用循环展开、循环流水策略后，消耗的逻辑资源由原先的1.91%增加至3.3%，这与循环展开增加逻辑资源的消耗相符合。

内核矢量化实现内核中多个工作项以单指令多数据（SIMD）的方式参与运算。结合工作组大小为512，将内核矢量化的参数指定为8，此时全局工作组的大小减少为原来的8倍。内核矢量化后内存带宽由5 779.7 MB/s提升至23 274.5 MB/s。这是因为矢量化内核会指导编译器合并内存访问，将对全局内存的8次Load操作合并为1次更宽的矢量Load操作且内核函数计算模块包含较少的分支语句，有利于形成较合适的SIMD通道。内核矢量化后FPGA端的工作频率为373.3 MHz，内核的执行时间由原先的46.620 ms缩短至11.132 ms，可见内核矢量化在性能上产生明显的提升。内核矢量化会增加FPGA端资源的消耗，由表5可知，其占用的逻辑资源由3.3%上升至19.41%。

将矢量化的参数由8提升至16，此时可形成更宽的内存访问操作。由表5可知，内存带宽进一步提高至27 534.1 MB/s，内核的执行时间缩短至9.425 ms，系统资源的消耗由19.41%增加至38.57%。

结合内核矢量化与计算单元复制的组合可进一步提高内核的性能。内核矢量化参数为8时，可复制的最大计算单元数为4；内核矢量化参数为16时，

可复制的最大计算单元数为2。

结合矢量化参数为8、4个计算单元复制后内存带宽为27 105.5 MB/s，相比于SIMD16有一定程度的下降，这是因为计算单元的增加导致对带宽的竞争，内核的执行时间缩短至9.409 ms，逻辑资源的消耗为77.33%。使用参数为16的内核矢量化与2个计算单元复制的组合，其内存带宽为28 102.3 MB/s，工作频率为366.7 MHz，内核的计算时间缩短至最低的9.243 ms，逻辑资源的占用达到77.03%。在有限的资源下，内核矢量化参数为16、计算单元复制数为2时获得了最佳的内核性能。下文将结合不同大小的明文数据量，进一步描述结合内核矢量化参数为16、计算单元复制数为2的内核性能变化。

### 4.4 不同数据量下系统吞吐率变化

为直观描述内核的性能，以64 KB、1 MB、8 MB、64 MB、128 MB、256 MB、512 MB、1 024 MB的数据为例，描述内核在不同数据量下的吞吐率变化情况。内核吞吐率的计算公式[21]如式(3)所示：

$$T=\frac{N\times B}{E} \tag{3}$$

其中：$T$为吞吐率；$N$为3DES加密的次数；$B$为单次3DES加密的明文块大小；$E$为内核的执行时间。不同数据量下内核的执行时间及吞吐率的变化情况如表6所示。可以看出：在数据量较小的情况下，内核的吞吐率不能很好地反映算法的真实性能；随着数据量的增加，内核的吞吐率在增加后趋于稳定；在计算的数据量大于128 MB后，内核的吞吐率保持在111.801 Gb/s左右。

**表6 不同数据量下的吞吐率**

**Table 6 Throughput rates under different data volumes**

| 数据量 | 时间/ms | 吞吐率/(Gb·s$^{-1}$) |
|---|---|---|
| 64 KB | 0.032 | 16.000 |
| 1 MB | 0.109 | 73.394 |
| 8 MB | 0.628 | 101.911 |
| 64 MB | 4.645 | 110.226 |
| 128 MB | 9.243 | 110.787 |
| 256 MB | 18.387 | 111.383 |
| 512 MB | 36.708 | 111.583 |
| 1 024 MB | 73.273 | 111.801 |

### 4.5 与不同方案的对比

为了验证本文方案的有效性，与其他文献方案进行比较，并与CPU、GPU平台实现结果进行比较。

#### 4.5.1 与其他文献方案的比较

文献[11,22-25]皆采用基于Verilog/VHDL的设计方案。本文基于OpenCL实现FPGA的设计，采用数据存储调整、数据位宽改进、指令流优化、内核矢量化等策略实现3DES算法加速器的设计。如表7所示，与基于Verilog/VHDL实现的方案相比，本文方案有效解决了开发周期长、维护升级困难等问题，同时频率达到了366.7 MHz，吞吐率达到111.801 Gb/s，取得了较明显的性能提升。

**表7 不同方案的加速性能**

**Table 7 Acceleration performance of different schemes**

| 方法 | 频率/MHz | 吞吐率 |
|---|---|---|
| 文献[11]方案 | 277.0 | 709.1 Mb/s |
| 文献[22]方案 | 348.9 | 797.593 Mb/s |
| 文献[23]方案 | 240.0 | 15 360 Mb/s |
| 文献[24]方案 | 50.0 | 3.2 Gb/s |
| 文献[25]方案 | 360.0 | 23.040 Gb/s |
| 本文方案 | 366.7 | 111.801 Gb/s |

#### 4.5.2 与CPU、GPU实现结果的比较

结合CPU、GPU平台验证本文方案的性能。OpenSSL是基于密码学的开发工具包，包含丰富的密码算法库。Hashcat是一种快速密码恢复工具，支持OpenCL框架。本文的对比对象为CPU端的OpenSSL库，其版本为1.0.2，CPU型号为Intel Core i7-9700；GPU端的Hashcat，其版本为5.0.0，GPU型号为NvidiaGeForce GTX 1080Ti。

由图10可知，本文实现方案相比于CPU性能提升372倍，相比于GPU性能提升20%。由图11可知，本文实现方案相比于CPU能效比提升644倍，相比于GPU能效比提升9倍。

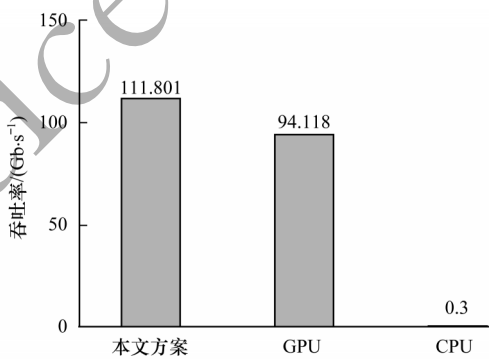



**图10 不同平台下的吞吐率**

**Fig.10 Throughput rate under different platforms**

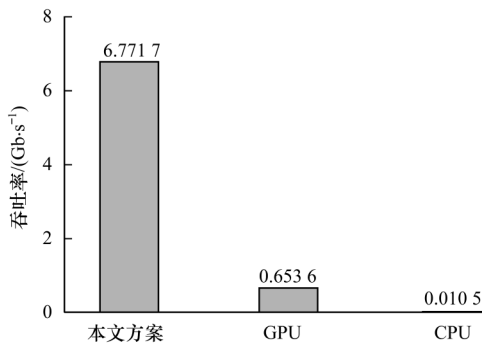



**图11 不同平台下的能效比**

**Fig.11 Energy efficiency ratio under different platforms**

## 5　结束语

本文针对传统数据加解密计算速度慢、功耗高、占用主机资源的问题,以及Verilog/VHDL方式实现的FPGA加解密系统开发周期长、维护升级困难的问题,提出一种基于OpenCL的3DES算法FPGA加速器架构设计方案。结合OpenCL内存模型与FPGA端硬件资源的对应关系优化数据存储模块,同时对私有内存与全局内存的数据传输模块,采用循环展开、数据位宽改进策略提高内存带宽的实际利用率,对3DES算法计算模块,采用指令流优化提高计算的并行度,形成流水线型架构。在此基础上,结合内核矢量化、计算单元复制策略进一步提高内核的吞吐率。实验结果表明,本文设计的加速器能够有效提升3DES算法的速度与能效方面,满足数字货币、区块链、云端数据加密等高强度计算领域的计算要求。为进一步提高该加速器的通用性和性能,后续将针对非对称加密算法和哈希算法进行设计,同时优化主机端与FPGA端数据的传输性能,开发实现支持算法类别更多的加解密算法计算平台。

## 参考文献


[1] 郭策,陈广财. 基于区块链技术的一种对病人隐私数据的加密保护方案[J]. 福建质量管理,2018(22):262.
GUO C,CHEN G C. An encryption protection scheme for patient privacy data based on blockchain technology[J]. Fujian Quality Management,2018(22):262. (in Chinese)

[2] PCI Security Standards Council. PCI DSS and PA-DSS glossary of terms,abbreviations,and acronyms v3. 2[EB/OL]. [2020-09-13]. https://www. pcisecuritystandards. org/documents/PCI_DSS_Glossary_v3-2. pdf? agreement=true&time=1496432377875.

[3] MUNSHI A, FUNG J, GINSBURG D, et al. OpenCL programming guide[M]. [S. l. ]:Addison-Wesley Professional, 2012.

[4] JIN Z, FINKEL H. Evaluation of MD5 Hash kernel on OpenCL FPGA platform[C]//Proceedings of 2018 IEEE International Parallel and Distributed Processing Symposium Workshops. Washington D. C. ,USA:IEEE Press,2018:1026-1032.

[5] KOROBEYNIKOV A. Effective implementation of 'kuznyechik' block cipher on FPGA with OpenCL platform[C]//Proceedings of 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering. Washington D. C. ,USA:IEEE Press,2019:1683-1686.

[6] GAO S, CHRITZ J. Characterization of OpenCL on a scalable FPGA architecture[C]//Proceedings of 2014 IEEE International Conference on Reconfigurable Computing and FPGAs. Washington D. C. ,USA:IEEE Press,2014:1-6.

[7] JANIK I, KHALID M A S. Synthesis and evaluation of SHA-1 algorithm using altera SDK for OpenCL[C]// Proceedings of 2016 IEEE International Midwest Symposium on Circuits and Systems. Washington D. C. ,USA:IEEE Press,2016:1-4.

[8] Federal Information Processing Standards Publication. Data Encryption Standard(DES)[EB/OL]. [2019-11-20]. https://csrc. nist. gov/publications/sp#800-67.

[9] 张焕国,杜瑞颖,傅建明,等. 信息安全工程师中级教程[M]. 北京:清华大学出版社,2016.
ZHANG H G,DU R Y,FU J M,et al. Information security engineer tutorial[M]. Beijing:Tsinghua University Press, 2016. (in Chinese)

[10] 任芳,杨承睿,陈雷华. 一种基于FPGA的3DES加密算法实现[J]. 西安工程大学学报,2011,25(4):555-559.
REN F,YANG C R,CHEN L H. 3DES implementation based on FPGA [J]. Journal of Xi' an Polytechnic University,2011,25(4):555-559. (in Chinese)

[11] 朱欣欣,李树国. 基于FPGA的高性能3DES算法实现[J]. 微电子学与计算机,2015,32(9):54-59.
ZHU X X,LI S G. High-performance 3DES algorithm implement based on FPGA[J]. Microelectronics and Computer,2015, 32(9):54-59. (in Chinese)

[12] NANE R, PILATO C, CHOI J, et al. A survey and evaluation of FPGA high-level synthesis tools[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,2016,35(10):1591-1604.

[13] Intel Corporation. Intel FPGASDK for OpenCL standard edition: Cyclone VSoC getting started guide[EB/OL]. [2019-11-20]. https://www. intel. com/content/dam/www/programmable/us/en/pdfs/literature/hb/opencl-sdk/aocl_c5soc_getting_started. pdf.

[14] Intel Corporation. Intel FPGASDK for OpenCL:Intel stratix 10 GXFPGA development kit reference pkatform porting guide[EB/OL]. [2019-11-20]. https://www. intel. com/content/dam/www/programmable/us/en/pdfs/literature/hb/opencl-sdk/ug-aocl-s10pciedk-platform. pdf.

[15] Intel Corporation. Intel FPGASDK for OpenCL:Intel Arria 10 GXFPGA development kit reference platform porting guide[EB/OL]. [2019-11-24]. https://www. intel. com/content/dam/www/programmable/us/en/pdfs/literature/hb/opencl-sdk/ug-aocl-a10pciedk-platform. pdf.

[16] JIN Z, FINKEL H. Optimizing parallel reduction on OpenCLFPGA platform—a case study of frequent pattern compression[C]//Proceedings of 2018 IEEE International Parallel and Distributed Processing Symposium Workshops. Washington D. C. ,USA:IEEE Press,2018:27-35.

[17] 王煜. 3DES加密技术的新探讨与改进[J]. 网络空间安全,2016,7(8):30-32.
WANG Y. New discussion and improvement of 3DES encryption technology[J]. Cyberspace Security,2016,7(8): 30-32. (in Chinese)

[18] Intel Corpration. Intel FPGA SDK for OpenCL Pro edition programming guide 19. 3[EB/OL]. [2019-11-20]. https://www. intel. com/content/dam/www/programmable/us/en/pdfs/literature/hb/opencl-sdk/archives/aocl_programming_guide-19-3. pdf.

[19] Intel Corpration. Intel FPGASDK for OpenCL Pro edition best practices guide[EB/OL]. [2019-11-20]. https://www. intel. com/content/dam/www/programmable/us/en/pdfs/literature/hb/opencl-sdk/archives/aocl-best-practices-guide-19-3. pdf.


(下转第162页)

（上接第155页）

[20] JIN Z M, FINKEL H. OpenCL kernel vectorization on the CPU, GPU, and FPGA: a case study with frequent pattern compression [C]//Proceedings of 2019 IEEE Annual International Symposium on Field-Programmable Custom Computing Machines. Washington D. C., USA: IEEE Press, 2019: 330.

[21] MOHAMED A, NADJIA A. SHA-2 hardware core for virtex-5 FPGA[C]//Proceedings of 2015 IEEE International Multi-Conference on Systems, Signals and Devices. Washington D. C., USA: IEEE Press, 2015: 1-5.

[22] GURUPRASAD S P, CHANDRASEKAR B S. An evaluation framework for security algorithms performance realization on FPGA [C]//Proceedings of 2018 IEEE International Conference on Current Trends in Advanced Computing. Washington D. C., USA: IEEE Press, 2018: 1-6.

[23] RAKANOVIĆ D, STRUHARIK R. IP core for AES256 and TDES algorithms with AXI interface[C]//Proceedings of 2016 Telecommunications Forum. Washington D. C., USA: IEEE Press, 2016: 1-4.

[24] ROSAL E D, KUMAR S. A fast FPGA implementation for triple DES encryption scheme[J]. Circuits & Systems, 2017, 8(9): 237-246.

[25] 李斌, 周清雷, 斯雪明, 等. 混合可重构的DES算核高效能口令恢复方案[J]. 计算机工程与科学, 2020, 42(10): 1887-1896.
LI B, ZHOU Q L, SI X M, et al. A high-efficiency password recovery scheme based on hybrid reconfigurable DES computing kernel[J]. Computer Engineering and Science, 2020, 42(10): 1887-1896. (in Chinese)

编辑　金胡考