祖冲之算法硬件实现与研究

周 威 王 博 潘伟涛
(西安电子科技大学 西安 710071)

摘 要：祖冲之(ZUC)算法是我国自主研制，纳入新一代宽带无线移动通信系统的国际加密标准，考虑现阶段祖冲之流密码算法的实现多为软件，效率和速度还有待提高。为适应大数据时代对高速数据实时加密的需求以及进一步推广 ZUC 算法的使用，结合 ZUC 算法特性，利用硬件描述语言 VHDL 对其进行高效的 FPGA 硬件设计，并使用 Xilinx 公司 ISE 软件进行综合仿真验证设计正确性，最后将模块封装成 IP 软核。结合 Xilinx 公司的 ARM 与 FPGA 联合平台设计相应的接口软件进行实际测试，性能分析和资源评估，为 ZUC 算法提供了一种高效硬件设计参考。最后给出 ZUC 模块在实际加密视频数据的应用，与纯软件实现相同功能相比，系统性能提升了 3 倍以上，可以满足实时视频数据的加密。

关键词：祖冲之(ZUC)算法 ；VHDL；FPGA
中图分类号：TP 309.7 TN7 **文献标识码**：A **国家标准学科分类代码**：520.1060

ZUC hardware implementation research

Zhou Wei Wang Bo Pan Weitao
(Xidian University, Xi'an 710071, China)

Abstract: ZUC algorithm is developed in China by the Chinese academy of sciences and other units, in new generation broadband wireless mobile communication system international encryption standard. And at present, the algorithm is realized by software, and the efficiency and speed of the ZUC stream cipher algorithm needs to be improved. In order to adapt to the era of big data demand for high-speed real-time data encryption, and further promote the use of ZUC algorithm, considering about the feature of ZUC algorithm for efficient, we use hardware description language VHDL design FPGA hardware and integrated simulation using Xilinx ISE software company design is correct. Finally, it will be encapsulated into the IP soft core module. According to the design of ARM and FPGA platform for Xilinx company corresponding interface software for actual testing, performance analysis and resources assessment, for ZUC algorithm provides a highly efficient hardware design reference. Finally, we use the ZUC module in video data encryption. Compared with the pure software to realize the same function, the system performance improved nearly 3 times. It can satisfy the real-time video data encryption.

Keywords: ZUC atgorithm; VHDL; FPGA

1 引 言

祖冲之(ZUC)算法由中国科学院等单位研制，由中国多家单位共同推动纳入 3GPP 新一代宽带无线移动通信系统(LTE)国际标准[1]。这是我国商用密码算法首次走出国门，参与国际标准制定。并且已被国际 3G 组织推荐为 4G 无线通信的第三套国际加密和完整性标准的侯选算法。该算法提高了我国在通信领域的地位和影响[2]，对我国移动通信产业和商用密码产业发展有着重大意义[3]。而现在针对祖冲之流密码算法的实现多为软件，效率和速度还有待提高。而且现阶段相关研究只是对 ZUC 算法本身进行了简单的优化实现，应用性不强，工作十分有限。文献[4]第一次在 FPGA 上使用流水线简单实现了 ZUC 算法，文献[5]对 FPGA 的设计进行了进一步的优化使吞吐率提高到 2 Gbps 以上。文献[6]进一步做出了优化，但实现方法但过于依赖实验平台，应用价值不高。文献[7]只是提供了加密算法的软件实现和调用接口，并没有考虑 ZUC 算法硬件实现的接口问题。所以在深入分析 ZUC 算法的基础上，查阅相关文献[8]，结合 FPGA 高速并行等特性，利用硬件描述语言对其进行硬件高效设计实现。然后使用 Xilinx 公司的 ISE 与 Vivado 软件进行了仿真验证，

收稿日期：2015-03

结果表明，该方法在芯片资源占用仅为601个Slice的主频140 Hz情况下达到了4.5 Gps的吞吐量，与目前已知最优实现方法在相同时钟频率相比[3]，芯片资源占用少，单位面积吞吐量提高28%左右，可以在减少芯片硬件资源占用的同时，快速高效的实现ZUC算法。并且推广硬件ZUC模块的应用，在参考国内CPU与FPGA接口设计后进行优化，设计了高效的专用总结接口模块与参考驱动设计，对现有设计相比大大提高了接口CPU的工作效率与读写效率保证了ZUC模块的在实际应用的高吞吐率。最后，结合上述设计实际应用于视频数据的加密中，1 s可加密30帧以上的视频数据。可以很好的解决现阶段无压缩编码实时视频加密的难题，满足了现阶段实时视频加密的需求，有着良好的应用前景与市场空间。

2 ZUC算法模块分析

ZUC算法是一个面向字的同步流密码[3]。它需要一个128位的初始密钥和一个128位的初始矢量(IV)作为输入，输出一串32位字的密钥流(因此，这里将每一个32位的字称为密钥字)。密钥流可以用来加密/解密。本节首先描述ZUC算法总体结构，然后分部分进行模块化设计[9]与简要说明。

2.1 总体结构

如图1所示，ZUC有3个逻辑层。顶层是一个16段的线性反馈移位寄存器(LFSR)，中间层是比特重组部分(BR)，底层是一个非线性函数F[1]。

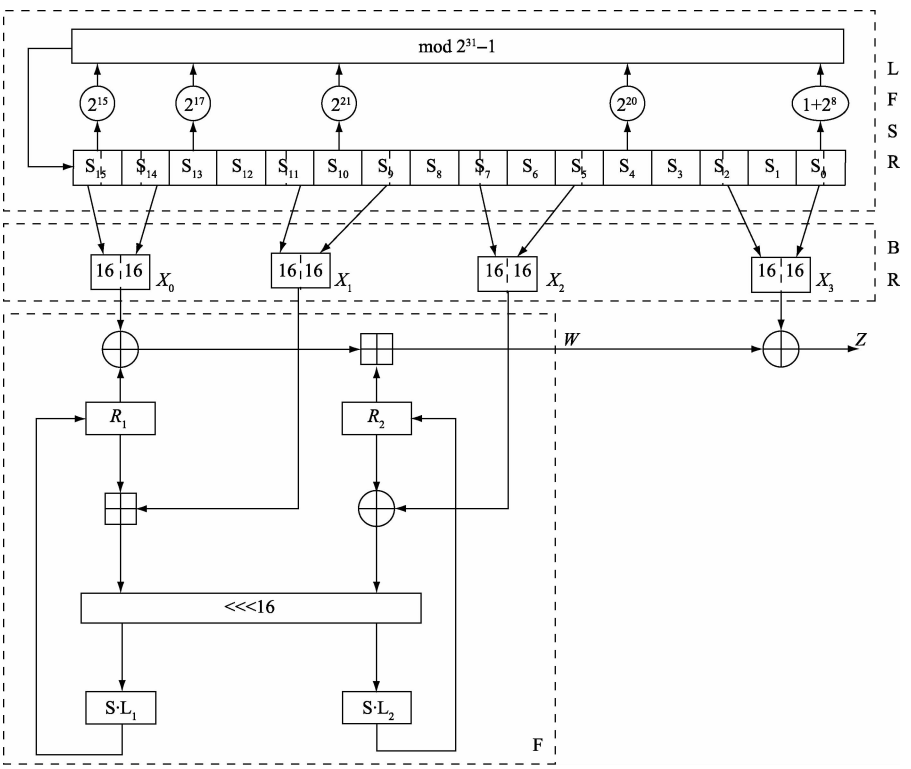

图1 ZUC算法的总体结构

2.2 线性反馈移位寄存器

线性反馈移位寄存器(LFSR)有16个31位的单元(s_0, s_1,…, s_{15})，每个单元$s_i(0\leqslant i\leqslant 15)$仅限在下列集合中取值$\{1,2,3,\cdots,2^{31}-1\}$。LFSR包含两个操作模式：初始化模式和工作模式[5]。

初始化模式时，LFSR接收一个31位的输入字u，u是通过去掉非线性函数F输出的32位字W的最右边的位获得的。也就是，$u=W>>1$。更具体地说，初始化时：

第1步：计算$2^{15}s_{15}+2^{17}s_{13}+2^{21}s_{10}+2^{20}s_4+(1+2^8)s_0 \bmod (2^{31}-1)$得到变量$v$；

第2步：$s_{16}=(v+u) \bmod (2^{31}-1)$；其中如果$s_{16}$为0，用$2^{31}-1$替换；

第3步：$s_1, s_2, \cdots, s_{15}, s_{16}$左移1组。

工作模式时，LFSR不接收任何输入，流程如下：

首先，计算$s_{16}=2^{15}s_{15}+2^{17}s_{13}+2^{21}s_{10}+2^{20}s_4+(1+2^8)s_0 \bmod (2^{31}-1)$；其中如果$s_{16}$为0，用$2^{31}-1$替换；

然后，$s_1, s_2, \cdots, s_{15}, s_{16}$左移1组。

2.3 比特重组模块

该算法的中间层是比特重组部分。该层从LFSR单元抽取128位，形成4个32位的字。这里的前3个字会在底层的非线性函数F中使用。最后一个字将涉及产生密钥流。

假设 s_0，s_2，s_5，s_7，s_9，s_{11}，s_{14}，s_{15} 是 LFSR 里的 8 个单元。比特重组从上面的 8 个单元里按如下方式形成 4 个 32 位的字 X_0，X_1，X_2，X_3：

$X_0 = s_{15H} \| s_{14L}$；$X_1 = s_{11L} \| s_{9H}$；$X_2 = s_{7L} \| s_{5H}$；$X_3 = s_{2L} \| s_{0H}$。

注解：s_i 是 31 位的整数，因此 s_{iH} 是指 s_i 的从第 30 ～ 15 位，而不是第 31 ～ 16 位，对于 $0 \leqslant i \leqslant 15$。

2.4 非线性函数的计算

非线性函数 F 包括 2 个 32 位的记忆单元 R_1 和 R_2。输入 X_0，X_1 和 X_2，其中 X_0，X_1 和 X_2 来自比特重组的输出，然后 F 输出一个 32 位的字 W，函数 F 的具体过程如下：

第 1 步：计算 W_1 W_2 W_3 分别为 $(X_0 \oplus R_1) \boxplus R_2$；$R_1 \boxplus X_1$；$R_2 \oplus X_2$；

其中$\boxplus$为模 2^{32} 的加法。

第 2 步：计算 R_1 和 R_2 分别为 $S(L_1(W_{1L} \| W_{2H}))$；$S(L_2(W_{2L} \| W_{1H}))$。

L_1 和 L_2 都是 32 位到 32 位的线性变换，其定义如下：

S 表示 S 盒运算，32×32 的 S 盒由 4 个并列的 8×8 的 S 盒组成，也就是说，$S = (S0, S1, S2, S3)$，这里 $S0 = S2$，$S1 = S3$。，因此只需定义两个 S 盒 $S0$ 和 $S1$，这样的结构化的设计方法，降低了硬件实现的代价。

2.5 密钥流产生

随着 LFSR 的初始化，密钥加载过程将会把初始的密钥和初始矢量扩展到 16 个 31 位的整数。假设 128 位的初始密钥 k 和 128 位的初始向量 iv 如下：

$k = k_0 \| k_1 \| k_2 \| \cdots \| k_{15}$ 和 $iv = iv_0 \| iv_1 \| iv_2 \| \cdots \| iv_{15}$

接着 k 和 iv 被加载到 LFSR 的 s_0，s_1，…，s_{15} 如下：

首先假设 D 是一个由 16 个 15 位子字符串组成的 240 位的长常字符串：

$D = d_0 \| d_1 \| \cdots \| d_{15}$

然后对 $0 \leqslant i \leqslant 15$，使 $s_i = k_i \| d_i \| iv_i$。

3 ZUC 算法硬件高效优化实现

ZUC 算法实现采用硬件描述语言 VHDL 实现，首先进行模块化设计[9]，包括比特重组模块、非线性函数 F 模块、移位寄存器初始化模块、移位寄存器工作和顶层设计模块。在顶层设计模块中，通过调用各子模块来实现 ZUC 算法的密钥流产生。顶层调用模块设计成时序逻辑电路，利用有限状态机对子模块进行调用，可以很好地控制算法的执行。

ZUC 的算法分为 2 个阶段：初始化阶段和工作阶段。在第 1 阶段，将密钥和初始向量 IV 初始化，也就是，时钟控制着密码运行但不产生输出。第 2 阶段是工作阶段，在这个阶段，随着每一个时钟脉冲，它都会产生一个 32 位字的输出。

整体算法流程如图 2 所示：算法启动，开始初始化阶段依次读入初始密钥值和初始化向量 IV，进行算法初始化阶段；初始化结束后进入算法工作阶段，丢弃第 1 组数据，并给出工作标志位，依次产生密钥流，

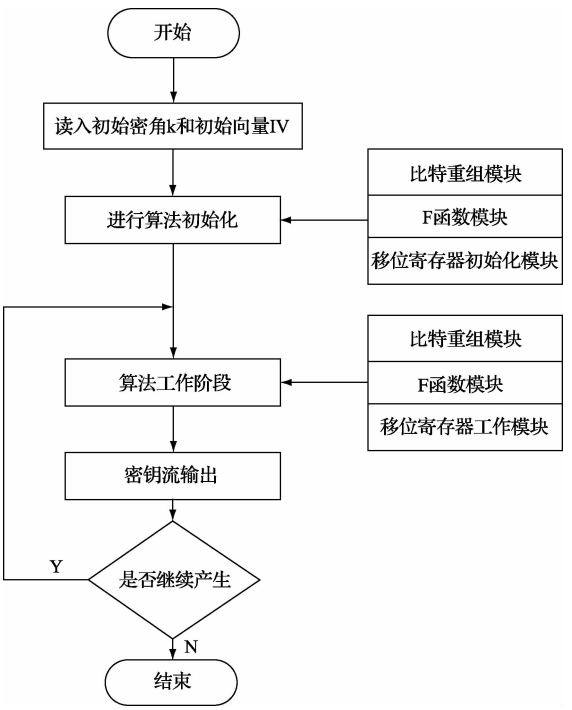

图 2 ZUC 算法顶层模块设计流程

首先本文考虑异步时序产生的延迟可能会导致竞争与冒险，和组合逻辑的不易控制等问题，这里所有的状态转移均采用同步时序逻辑，保证时钟节拍可以精确控制数据流动。

根据算法和外部可控的要求设计了 5 个状态分别为密钥加载算法初始化和算法工作状态以及密钥流输出状态和外部是否继续产生控制状态。

同时本文为了充分利用 FPGA 并行处理的能力将算法初始化和工作阶段的比特重组和 F 函数以及移位寄存器设计为不同的模块每个时钟进行并行处理并且对 2 的幂指数运算采用移位运算替换，大大提高了算法的处理速度。

4 功能验证与性能分析

4.1 验证平台

为了验证 FPGA 硬件设计的正确性同时为了软件方便调用与测试的需要：选用 Xilinx 公司的微处理器和 FPGA 联合处理芯片 ZYNQ-7020 为目标器件进行测试，硬件验证平台如图 3 所示。

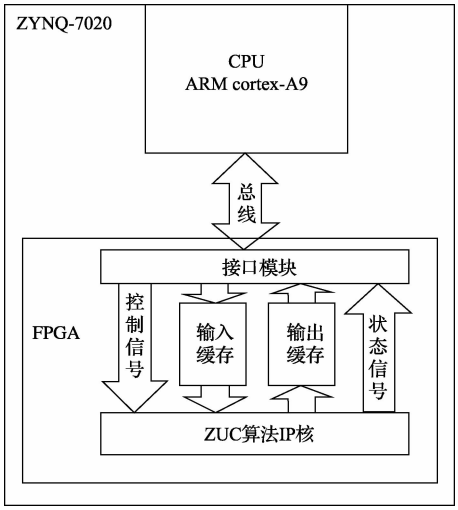

图3 硬件验证平台

4.2 验证方案

考察硬件密码产品的性能主要是从加解密吞吐量，加解密速度，资源占用率以及与纯软件设计的比较这几个方面，这里为了充分验证设计的流密码算法的高效性，研究CPU与FPGA的接口设计[10]，给出了一种合适的总结接口设计和接口驱动软件设计，最后在Xilinx公司的ZYNQ平台嵌入式终端进行设计验证。这里ZYNQ的处理器为主频667 MHz ARM Cortex-A9处理器，通过AXI总线与FPGA互联，其他处理器设计类似。下面分部分简述验证方案。

4.2.1 ZUC IP核总线接口

本设计将ZUC算法作为硬核IP挂接到系统总线，系统与ZUC_IP的传输使用ARM处理器控制方式实现。根据ARM硬核嵌入式系统的工作要求和ZUC算法的特性，设计了4个模块：系统接口模块、输入缓冲区模块、输出缓冲区模块、密码算法模块。

其中总线接口模块是ZUC IP核与ARM处理器通信的桥梁。负责系统与ZUC密码硬核直接的通信协调，通过控制寄存器和输入寄存器控制密码算法的密钥流产生流程，通过状态寄存器和输出寄存器获得系统状态，读出处理数据。接口整体设计如图4所示。

4.2.2 接口驱动软件设计实现

当建立好产生ZUC密钥流的IP核后，需要通过上层处理器（ARM）驱动软件模块对其进行调用，处理IP核端口与AXI总线上的数据传输，并用获得的密钥流进行加解密。整个系统接口软件控制流程如图4所示。

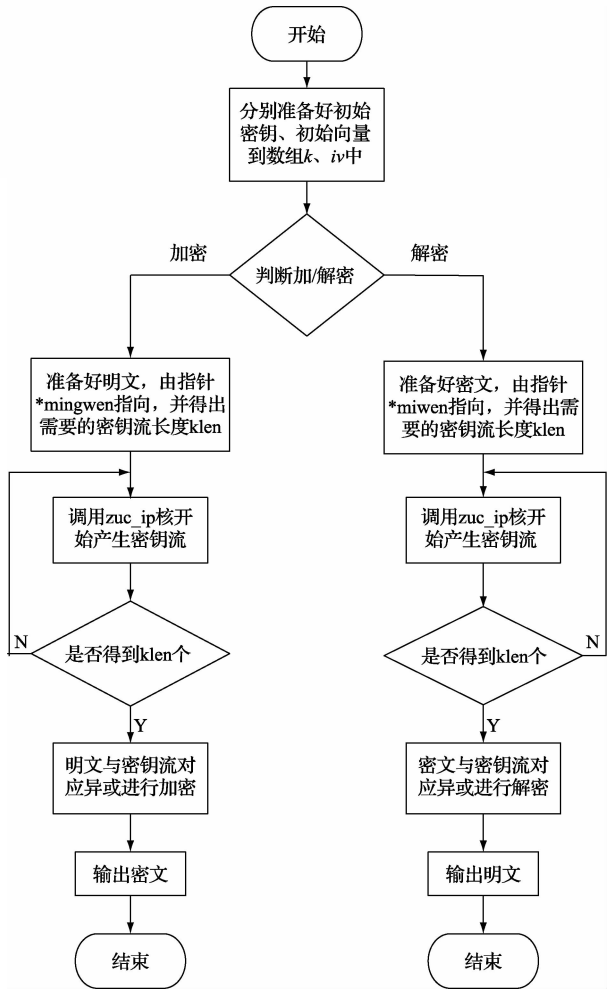

图4 ZUC软件程序流程

4.3 结果分析

4.3.1 结果验证

为了验证ZUC IP及总线连接正确性和接口软件的正确性，将ARM处理器获得的前3组密钥流输出到串口进行验证。采用初始密钥为3d4c4be9 6a82fdae b58f641d b17b455b、初始矢量IV为84319aa8 de6915ca 1f6bda6b fbd8c766，在工作阶段产生密钥流14flc272 3279c419 4bbea41d……（如图5）

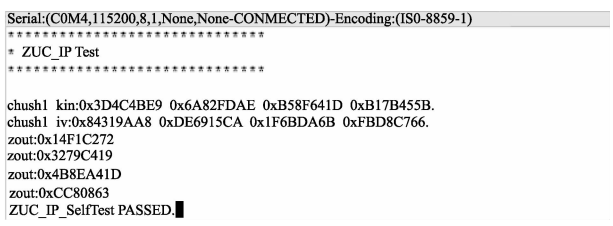

图5 ZUC接口模块测试

测试数据与官方数据一致，证明 SM3_IP 驱动软件设计正确，系统总线通信正常。

4.3.2 资源分析

使用 Xilinx 公司的 ISE 工具对 ZUC IP 核代码进行综合，目标器件为 Zynq-7020，综合后的资源占用情况如表 1 所示。

表 1 FPGA 资源统计

硬件占用资源（ALUTs）	寄存器数（Slice Registers）	存储单元利用率	工作主频
19 553%	6 011%	1%	142.7 MHz

可见本文设计的 ZUC IP 核硬件占用资源较少。

4.3.3 吞吐量分析

系统综合后给出时钟分析报告可知，ZUC 模块在 ZYNQ 平台时钟频率可达到 142 MHz，初始化需要 32 个时钟然后每个时钟可产生 32 bit 密钥流，吞吐量可达：142 M×32＝4.5 Gbps。（理论吞吐量＝工作主频×32）。

可见，用 FPGA 设计的 ZUC 流密码算法模块使其吞吐量大大提升。

5 视频数据加密应用

5.1 测试结构

为了进一步验证 ZUC 高速加密模块的性能，把模块实际应用在视频加密中的设计中[11]，并与软件加密性能进行比较，进行鲁棒性分析。

为了保证应用与上述硬件验证的一致，这里仍然选用 Xilinx 公司的微处理器和 FPGA 联合处理芯片 ZYNQ-7020 为核心的 zedboard 板卡为目标器件进行实验，主要实验平台如图 6 所示。

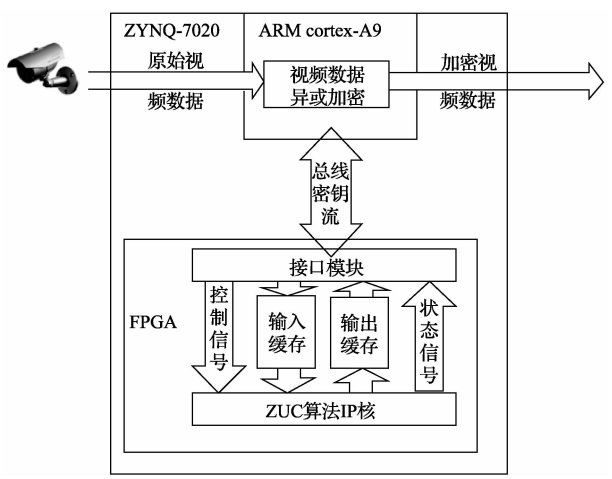

图 6 视频加密测试

5.2 应用流程

根据图 6，首先将摄像头采集的原始视频送入内存，然后将数据通过 AXI 总线送入 ARM 处理器，ARM 通过调用驱动实时读取 FPGA 产生的密钥流，将密钥流与视频数据进行异或加密，完成对视频数据的祖冲之流密码加密。然后将加密完成的数据可以存入板卡固定的存储器，或者直接送入网卡进行有线或者无线传输，方便实用。

5.3 测试结果

5.3.1 加解密效果

如图 7(a)是经过加密以后的视频数据，每一帧图片都呈现白噪声的状态而且连续播放也无法得出有关原始视频的任何信息。对原始视频数据进行了很好的保护。

图 7(b)是经过 PC 解密处理以后还原的原始视频，可以看到对原始视频的完整恢复。从而完成了对实时视频数据的加密。

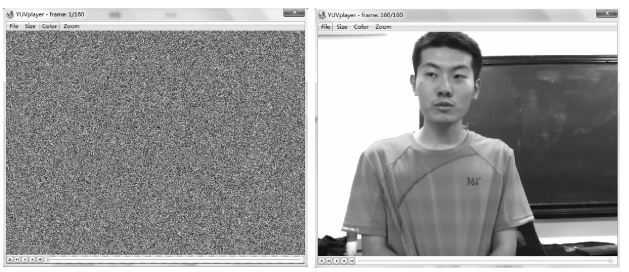

图 7 视频加解密对比

5.3.2 软硬件加密速度对比

利用 ISE 工具综合得到时钟报告如图 8 所示，可以得出时钟频率可达到 142 MHz，这里为了保持视频加密的稳定性和方便计算，选用 100 MHz 时钟数据线进行传输。初始化需要 32 h 然后每个时钟可产生 32 bit 密钥流，吞吐量可达：100 M×32＝3.2 Gbps。

```
Minimum period: 7.007ns (Maximum Frequency: 142.710MHz)
Minimum input arrival time before clock: 1.048ns
Maximum output required time after clock: 0.511ns
Maximum combinational path delay: No path found
```

图 8 测试时钟报告

首先用板卡加密约 70 MB 视频数据测试时间结果如图 9 所示，由于 ARM 需要异或加密和驱动调用的时间比密钥流产生的理论速率稍慢，加密用时 0.248 s。

然后将相同的视频数据在主频 2.5 GHz 内存 12 G 64 位操作系统的 PC 上同样完成 ZUC 加密算法用时如图 9 所示。

图 9 PC 软件加密

可以看到加密相同大小的视频数据，使用设计的硬件加密芯片可以比高端 PC 处理速度还要提高 3 倍以上。1 s 可加密 30 帧以上的视频数据。可以满足实时视频数据的加密。

5.3.3 鲁棒性分析

为了测试设计模块的稳定性和鲁棒性，分别选用50 MHz、100 MHz、150 MHz总线时钟频率和100 M、500 M、1 G左右的数据进行加密，均无发现密钥流编码和加解密错误，后续会继续加大测试信息量，但这些已可见该设计模块稳定性良好，拥有良好的市场前景。

6 应用前景

该文设计FPGA高效ZUC加密模块，可以留片生成固定的加密芯片，也可以用于FPGA的设备当做可以编辑的软核做加密处理，方便实用，占用硬件资源少，加密速度快，可以很好的解决现阶段无压缩编码实时视频加密的难题[12]。满足了现阶段实时视频加密的需求。可以运用于物联网下实时视频[13-14]，语音等大数据的加密，也可以用于现阶段云端大数据的加解密，有着广泛的应用前景。

7 结 论

利用FPGA的并行高速处理的特点，高效的设计和实现了ZUC算法，并给出了一种相应的总线和接口软件设计，在Xilinx FPGA的ZYNQ-7020平台进行硬件测试。实验结果说明本文设计的ZUC模块吞吐量可达4.5 Gbps以上，速度远远大于PC软件处理，方便实用，最终还给出一个应用在视频加密数据中的参考设计案例，对ZUC算法的推广和使用有着重要作用，有着广阔的市场前景。

参 考 文 献

[1] ETSI/SAGE TS 35.222-2011, Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3. Document 2: ZUC Specification [S].

[2] 冯秀涛. 3GPP LTE国际加密标准ZUC算法[J]. 信息安全与通信保密，2011，12(9)：45-46.

[3] 杜红红，张文英. 祖冲之算法的安全分析[J]. 计算机技术与发展，2012，22(6)：151-155.

[4] LIU Z, ZHANG L, PAN J J W. Efficient piplined stream C ipher ZUC algorithm in FPGA[C]. The First International Workshop on ZUC Algorithm，2010.

[5] KITSOS P, SKLAVOS N, SKODRAS A N. An FPGA implementation of the ZUC stream cipher[C]. 14th Euromicro Conference of Digital System Design (DSD)，2011：814-817.

[6] 李歌，陶琳，高献伟. 基于FPGA的祖冲之算法研究与实现[J]. 北京电子科技学院学报，2012，20(4)：13-18.

[7] ETSI/SAGE TS 35.223-2011, Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 &128-EIA3. Document 1: 128-EEA3&128-EIA3 Specification[S].

[8] 江丽娜，高能，马原，等. 祖冲之序列密码算法IP核的设计与实现[J]. 信息网络安全，2012(8)：219-222.

[9] 张松，李筠. FPGA的模块化设计方法[J]. 电子测量仪器仪表学报，2014，28(5)：560-565.

[10] 胡亚平. FPGA与CPU高速接口的实现[J]. 国外电子测量技术，2013，32(4)：66-68.

[11] 李梦瑶. 基于FPGA的视频图像加密系统的设计与研究[J]. 信息通信，2014(8)：50-51.

[12] 谈宇奇，王雪，林奎成. 基于视频压缩感知的传感器网络行人目标辨识方法[J]. 仪器仪表学报，2014，35(11)：2434-2439.

[13] 陈学涛，郑力明. 实时视频传输系统[J]. 计算机系统应用，2013，22(10)：60-64.

[14] 侯琛，赵千川，李海涛，等. 物联网中的嵌入式终端[J]. 电子测量技术，2014，37(10)：113-118.

作 者 简 介

周威，1993年出生，本科生。主要研究方向为安全芯片设计，操作系统安全，图像处理与信息隐藏技术。
E-mail：chunwei123hu@163.com

潘伟涛，1981年出生，博士，副教授，硕士生导师。主要研究方向为通信系统芯片设计、三网融合(HINOC)、数字芯片门级电路信息提取技术研究。
E-mail：wtpan@mail.xidian.edu.cn