SoC PCIe端点模式密码卡的技术研究与实现

窦同锐,张中方,张 磊

(山东多次方半导体有限公司,济南 山东 250101)

摘 要: 密码技术是信息安全的基石,用于保障信息的完整性、机密性和不可否认性。PCIe密码卡虽广泛应用于各类场景,但其存在设计复杂、异常检测不足及固件更新不便等问题。为此,提出了一种基于SoC PCIe端点模式密码卡的设计方案。首先,使用SoC集成双模式的PCIe控制器和密码算法引擎,从而简化硬件结构;其次,设计一种灵活的软件架构,软件部分包括引导固件、PCIe EP软件框架、算法引擎和守护进程等,以拓展应用场景;最终,实现权限管理、密钥管理与密码运算等功能,并增强状态监控和异常处理能力,以支持热升级与日志记录。基于SoC芯片RK3588实现的密码卡,在数据包2MB时,DMA性能达理论值的72%。

关键词: 片上系统;密码卡;PCIe端点;直接内存访问;状态监控

DOI: 10. 11907/rjdk. 251711 扫描二维码阅读全文:

中图分类号: TP309 **文献标识码:** A **文章编号:** 1672-7800(XXXX)0XX-0001-09

Technical Research and Implementation of A PCIe Endpoint Mode Cryptographic Card in System-on-Chip

DOU Tongrui, ZHANG Zhongfang, ZHANG Lei

(*Shandong Exponent Semiconductor Co. , Ltd. , Ji'nan* 250101, *China*)

Abstract: Cryptography serves as the foundation of information security, ensuring data integrity, confidentiality, and non-repudiation. PCIe cryptographic cards, a key class of security hardware, are widely deployed but often hindered by complex designs, insufficient anomaly detection, and cumbersome firmware updates. To overcome these challenges, this paper presents a PCIe cryptographic card based on a System-on-Chip(SoC) architecture. By integrating a Dual-Mode(DM) PCIe controller and cryptographic algorithm engines into the SoC, the hardware design is significantly streamlined. A flexible software stack is also implemented, comprising boot firmware, a Linux kernel endpoint(EP)framework, algorithm engines, and service daemons. The system supports access control, key management, cryptographic operations, enhanced status monitoring, exception handling,firmware hot-upgrade, and comprehensive logging. Experimental results on the RK3588 platform demonstrate that the DMA throughput achieves 72% of the theoretical maximum with 2 MB data packets.

Key Words: system-on-chip (SoC); cryptographic card; PCIe endpoint (EP); direct memory access (DMA); state monitor

0 引言

自2012年国家密码管理局发布六项密码行业标准及密码应用表示规范以来,我国密码技术的应用场景不断拓展,逐步构建起从基础密码算法、密码产品制造到系统集成的完整产业生态[1]。以密码芯片、密码模块和密码整机为核心的密码产品体系不断完成功能迭代与性能提升。其中,PCI(Peripheral Component Interconnect express)密码卡是以PCI局部总线或者PCIe为接口,具有密码运算功能、密钥管理功能、物理随机数产生功能和设备自身安全保护措施的密码设备,凭借即插即用等特性,已成为重要的密码安全硬件载体[2]。

当前密码卡架构多围绕传输总线、主控芯片和密码算法芯片设计,侧重功能迭代与性能提升,但普遍存在硬件结构复杂、异常检测能力不足、固件更新困难等问题,制约了其维护性。随着密码卡应用日益广泛,场景日趋复杂,

收稿日期: 2025-12-03

作者简介: 窦同锐(1990-),男,CCF高级会员,山东多次方半导体有限公司副高级工程师,研究方向为高速传输总线、嵌入式系统、密码技术;张中方(1985-),男,硕士,山东多次方半导体有限公司工程师,研究方向为FPGA设计开发、SoC芯片设计和验证,以及网络安全和密码技术研究;张磊(1986-),男,山东多次方半导体有限公司工程师,研究方向为SoC芯片设计和验证,以及网络安全和密码技术研究。本文通讯作者:张中方。

上述短板愈发突出。因此,探索密码卡的新型设计范式,并提升其可维护性,对密码行业的发展具有重要意义。

1 相关研究

弓国伟[3]设计了一款SM4密码卡,使用集成ARM Cortex-A9和FPGA(Field-Programmable Gate Array)功能的ZYNQ 7020芯片,替代PCIe密码卡主控和密码算法两类芯片。虽然整体设计有一定新颖性,但选用的CH368 PCIe桥芯片最高性能只有50 MB/s[3]。高东飞[4]设计了一款PCI密码卡,主控和密码算法芯片分别选用SSX45和SSX30-D,FPGA集成了PCI协议,权限管理采用IC卡(Integrated Circuit Card),整体思路复刻了传统"主控—算法—通信"三段式设计范式。苏振宇[5]设计了一款基于FPGA和DSP(Digital Signal Processor)的PCIe密码卡,DSP作为主控芯片,选用集成PCIe IP(Intellectual Property)硬核的FPGA芯片,密码算法芯片选用SSX30-D、RSA等芯片。该方案结合高东飞[4]设计PCI密码卡的权限管理,曾是PCIe密码卡的主流设计方案,现正逐步被市场淘汰。彭帅[6]设计了一款基于安全ASIC(Application-Specific Integrated Circuit)密码芯片的PCIe密码卡,虽然利用ASIC芯片特性,实现了SM2签名速率达20000次/s,但算法支持单一且使用的PCIe桥片PEX8311早已停产[6]。

综上所述,当前国内在PCIe密码卡新型设计范式和提高密码卡可维护性方面的研究较为匮乏。针对这一现状,本文提出一种基于SoC(System On Chip)PCIe端点模式的密码卡设计范式。为验证这一范式的通用性与可行性,选取了具有代表性的SoC芯片NXP i.MX8MM、Rockchip RK3588和StarFive JH7110,涵盖不同的硬件体系架构、PCIe控制器IP实现以及内置密码算法引擎,从多维度开展方案验证。为满足不同应用场景的多样化需求,本文结合实际应用特点,设计了4种软件架构方案,并重点对其中3种典型方案进行了实现与效果评估,以验证其适用性与维护性的优势。

本文选取三未信安公司生产的SJK1727 V2.0-A型PCIe密码卡作为对比对象,该密码卡基于其自研的XS100安全芯片实现。XS100安全芯片集成了高速总线、主控单元和加密算法功能,相较于传统方案在技术上有显著进步。但其沿用了MCU(Microcontroller Unit)作为主控单元,在密码卡可维护性上没有实质改进。目前,市面上有代表性的密码安全专用芯片相关数据,如表1所示。

Table 1 Cryptographic security chip

表1 密码安全芯片

芯片厂商	山东多次方	苏州国芯	方寸微电子
芯片型号	XS100	CCP917T	TIH640V690
MCU	32位RISC-V	64位CRV7	64位RISC-V
PCIe	GEN2 X4	GEN4 X16和X4	GEN3 X1
密码算法	SM1、SM4、SM7、SM2、SM3、SM9等	SM1、SM4、SM7、SM2、SM3、SM9等	SM4、SM2、SM3、SM9、RSA等

2 SoC PCIe端点模式密码卡设计

2.1 硬件系统架构设计

现代SoC芯片普遍集成通用CPU(Central Processing Unit)核、PCIe总线接口以及密码算法加速引擎。单颗SoC芯片即可实现PCIe密码卡中高速总线、主控芯片及密码芯片的全部功能,形成高度集成化的解决方案。SoC集成的密码算法加速引擎在支持算法种类和性能方面正逐步向专用密码芯片的水平靠近,且已具备商用价值。

典型PCIe拓扑结构由根复合体(Root Complex,RC)、交换器(Switch)和端点设备(Endpoint,EP)构成,其中,PCIe密码卡是典型的端点设备[7-8]。SoC芯片默认将PCIe接口配置为根复合体模式,而PCIe密码卡作为外设需工作在端点模式。因此,在芯片选型时必须选用支持双模式的SoC,以满足设备角色定义的需求。SoC芯片PCIe密码卡硬件拓扑如图1。其中,SoC集成了PCIe控制器、密码算法。该架构在降低了系统设计复杂度的同时提高了智能性,这或将成为智能化密码硬件的重要发展方向。

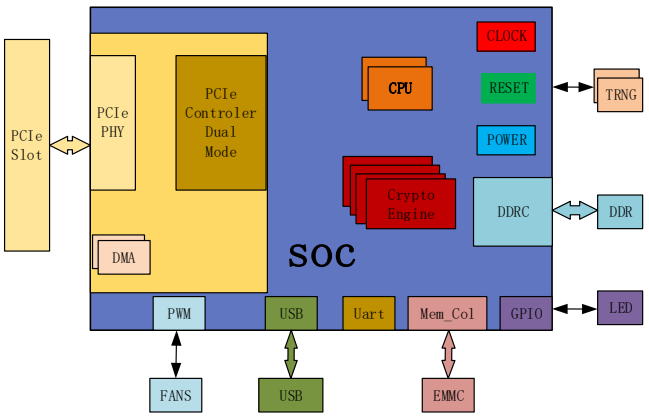

Fig. 1 Hardware topology of SoC chip-based PCIe cryptographic card

图1 SoC芯片PCIe密码卡硬件拓扑

2.2 软件系统整体流程设计

尽管基于SoC PCIe端点模式的密码卡在硬件架构上实现了高集成度,但其软件系统实现的复杂度和开发难度也相应提升。在软件技术架构设计方面,主要有以下四类实现路径。

(1)方案一。为SoC开发具备PCIe模式配置功能的裸机程序。该方案延续传统PCIe密码卡的软件架构思路,直接控制硬件资源。它适用于对实时性要求高、功能简单的场景。

(2)方案二。基于Linux内核提供的PCIe端点软件框架,结合Bootloader阶段对PCIe工作模式的初始化配置,构建密码服务守护进程。该方案采用操作系统提供的资源管理、内存调度与进程通信机制,适用于PCIe IP被Linux

EP软件框架支持的SoC平台。

(3)方案三。针对PCIe IP未被Linux内核支持的SoC,需重新实现EP软件框架,并开发应用服务。此方案虽技术门槛高,需深入理解PCIe协议栈与SoC底层硬件交互机制,但具备更高灵活性和定制化能力。

(4)方案四。采用独立PCIe配置镜像与操作系统驱动相配合的方式。该方案仅适用于支持用户自定义启动镜像的SoC平台。

结合方案二、方案四的典型架构,软件系统整体工作流程如图2所示。两种方案的核心区别体现在金手指上电后的镜像跳转流程:方案二上电后直接进入Bootloader,完成PCIe端点模式配置;方案四则跳转至专用PCIe配置镜像,依托该镜像实现PCIe接口的端点模式初始化。

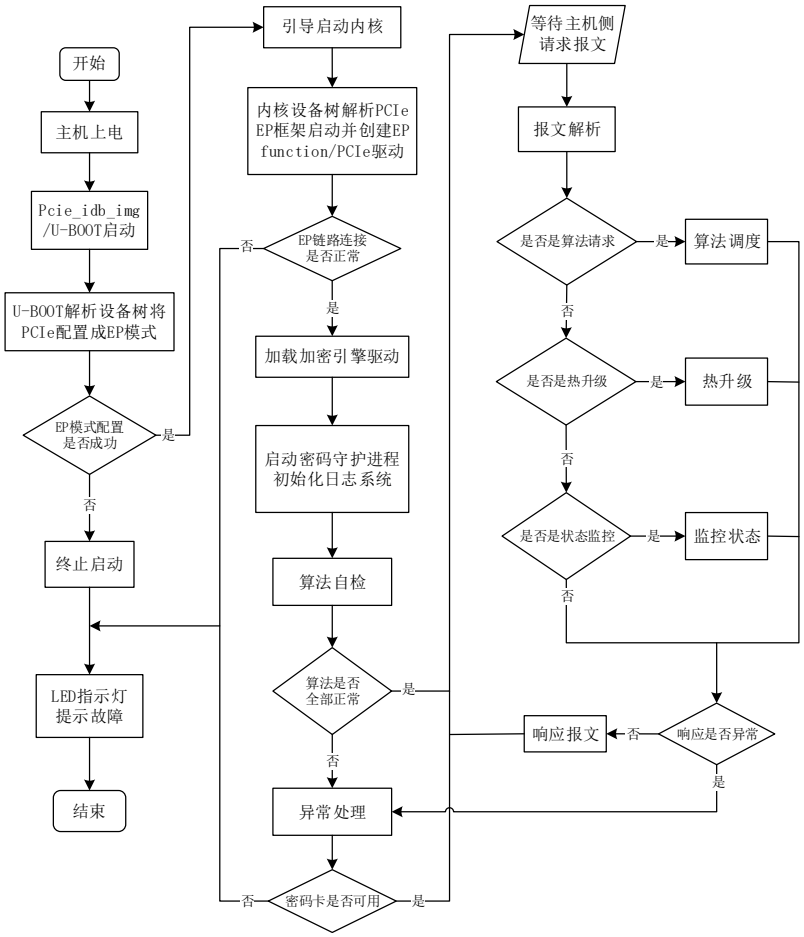

Fig. 2　Software system flowchart

图2　软件系统流程图

首先,Bootloader固件校验PCIe配置后的工作模式,复合体模式直接终止启动并LED告警,端点模式引导启动Linux内核。Linux内核加载PCIe EP驱动模块,其中,方案二创建EPF(Endpoint Function)设备实例,方案四加载专用PCIe驱动模块。核验PCIe链路状态,异常则终止启动并LED告警,正常则加载密码算法引擎驱动。

其次,系统启动密码服务守护进程,完成日志模块初始化与密码算法自检。若自检失败,系统进入异常处理流程并评估运行状态,设备完全不可用则通过LED提示严重故障,可降级运行则标记异常状态并进入服务就绪状态。

然后,守护进程进入待命状态并将设备标记为“就绪”,等待接收主机侧请求。主机端驱动加载完成后,优先校验密码卡就绪状态。守护进程接收主机请求报文后,解析并分类处理密码运算、热升级、状态监控等指令,正常处理结果回传至主机,异常问题交由异常处理模块处置。

最后,在权限管理层面,依托SoC的USB接口外接UKey或配合主机侧UKey,实现基于角色的多级访问控制,确保操作行为可审计与权限分级管控,从而提升系统的安全性与可控性。

2.3　方案关键技术

2.3.1　Linux内核PCIe EP软件框架

Linux内核构建了功能完备的PCIe子系统,主要用于支持PCIe控制器以根复合体模式运行。随着嵌入式系统与专用设备的发展,越来越多的SoC中PCIe控制器IP核开始支持双模或端点模式。为顺应硬件演进趋势,Linux内核引入了PCIe EP软件框架,用于支持SoC作为PCIe端点设备运行。该框架降低了在端点模式设备上部署Linux操作系统的复杂度,拓展了PCIe技术在嵌入式安全模块、加

速卡等场景的应用边界。

Linux 内核 PCIe EP 软件框架采用分层架构设计，系统结构如图 3 所示。其核心由 3 个部分构成：EPC（Endpoint Controller）库、EPF 库以及 configfs 配置接口层[9]。其中，i.MX8MM 和 RK3588 使用 Linux 5.x 内核版本，PCIe EP 软件框架不支持 PCIe DMA 的操作，导致 EPF 设备驱动无法使用标准接口操作 DMA。基于此，需要对 PCIe EP 软件框架进行扩展，增加统一的 DMA 操作入口。图 3 中的 dma_transfer 与 pci_epc_dma_transfer，是 PCIe EP 软件框架新增 DMA 操作的具体实现。

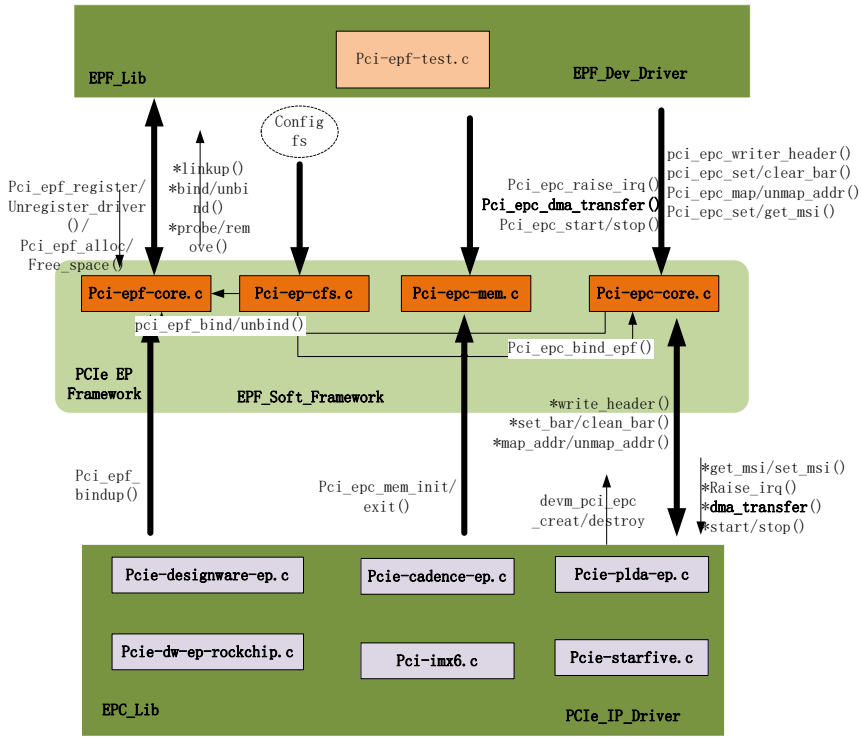

Fig. 3 PCIe EP framework

图 3 内核 PCIe EP 软件框架

2.3.2 Linux 内核重构 PCIe EP 软件框架

Linux 内核支持多种主流 PCIe IP 核，包括 Cadence、Synopsys DesignWare、Mobiveil 和 Plda 等。然而，PCIe EP 软件框架目前仅原生支持 Cadence 与 Synopsys DesignWare 两类 IP 核。这也导致使用 Mobiveil 和 Plda 等 IP 核的 SoC，在实现 PCIe 端点功能时面临框架不支持，难以直接部署为端点设备的问题。为验证此类 IP 在 Linux 环境下的端点模式可行性，本文以 Plda PCIe IP 为例，对其在 PCIe EP 软件框架中的适配性进行了深入研究。

本文将围绕框架的 3 个核心层级：EPC 库、EPF 库和 configfs 配置，系统性地探讨在各层级中实现新 IP 适配的技术路径与优化策略。

首先，在 EPC 库层，适配工作主要涉及 SoC 平台层控制逻辑与 PCIe IP 底层寄存器操作的实现。平台层需完成 PCIe 控制器工作模式的匹配与配置、相关寄存器物理基地址的映射与管理、核心功能参数的设定、DMA 中断的注册与处理，以及 EPC 实例的创建与绑定，在 IP 底层操作层面扩展支持 DMA 功能。

随后，EPF 库层与 configfs 配置层共同构成上层功能抽象与运行时配置机制。其中，EPF 库通过标准化接口封装具体设备功能，实现可复用的设备驱动模型；configfs 则提供用户空间接口，支持动态创建 EPF 实例、配置其属性并绑定至特定 EPC 控制器。得益于 PCIe EP 框架良好的模块化设计，针对不同 EPF 设备的适配主要体现为参数化配置与功能扩展，无需修改框架核心逻辑。因此，基于 Plda IP 的 SoC，仅需在 EPC 层完成底层驱动适配后，即可快速构建完整的端点设备功能。

2.3.3 独立的 PCIe 配置功能镜像

根据 PCIe 标准的上电时序规范，从电源稳定到释放 PERST#（Peripheral Reset）复位信号的最短时间不得少于 100 ms[10]。然而，标准未明确规定从 PERST#释放到 PCIe 链路退出电气空闲状态（Electrical Idle）的时间。理论上，从电源稳定至链路恢复的最短时间可接近 100 ms。尽管实际应用中多数主机平台对上电响应时间的要求相对宽松，但为确保 PCIe 外设设备能够被主机系统可靠识别并完成枚举，外设从上电到完成 PCIe 功能初始化的时间仍应控制在 100 ms 以内。

依据 eMMC（Embedded MultiMedia Card）标准规范，设备在启动过程中需执行初始化训练（Initialization and Training），该过程耗时应小于 1 s[11]。这一初始化延迟可能导致 PCIe 接口无法在主机探测窗口内完成配置，进而错过链路建立时机，导致设备枚举失败。为解决该问题，采

用双存储介质协同启动方案。通过选用无需初始化，具备快速启动特性的NOR Flash来存储PCIe功能配置镜像，从而确保设备在上电后能立即响应主机探测。同时，利用eMMC的大容量优势存储操作系统内核等大镜像文件，以兼顾启动速度与存储效率。

在具体实现中，将PCIe配置功能独立开发，仅初始化必要硬件模块，随后，将SoC的PCIe控制器配置为端点设备，并正确设置其基地址寄存器和配置空间。将该配置程序的运行基址定位至SoC片内SRAM，以提升执行效率，并将其烧录于NOR Flash的起始地址区域。系统上电后，BootRom完成基本硬件初始化并立即跳转至该PCIe配置程序，从而确保PCIe端点在主机探测周期内完成初始化，实现快速链路建立。

3　SoC PCIe端点模式密码卡的实现

在基于SoC PCIe端点模式的密码卡技术方案研究与实现过程中，本文选取了3款具有代表性的SoC芯片：i.MX8MM、RK3588和JH7110。其中，本文使用i.MX8MM芯片验证方案二，基于原生PCIe EP软件框架实现密码卡；使用StarFive JH7110芯片验证方案三，基于重构PCIe EP软件框架实现密码卡；使用Rockchip RK3588芯片验证方案四，基于独立的PCIe配置镜像实现密码卡。SoC主要特性如表2所示。

Table 2　Basic characteristics of SoC

表2　SoC主要特性

芯片型号	i.MX8MM	JH7110	RK3588
架构	ARM	RISC-V	ARM
CPU主核	A53 x4	U74 x4	A55 x4
PCIe IP	DesignWare	PLDA	DesignWare
PCIe能力	GEN2 X1	GEN2 X1	GEN3 X4
PCIe EP软件框架	支持	不支持	支持
密码算法	AES、ECC、3DES	AES、3DES、ECC	AES、3DES、ECC
国密算法	_	_	SM2、SM3、SM4

3.1　基于原生PCIe EP软件框架的端点模式实现

3.1.1　设置端点模式

板卡上电后，不同SoC芯片执行至Linux内核中PCIe EP软件框架的耗时存在一定差异，普遍在4～5 s之间。根据PCIe标准的上电时序规范，此过程耗时较长，存在错过主机探测窗口的风险[10]。从板卡上电300 ms左右，即可进入U-Boot SPL(Secondary Program Loader)阶段，并完成PCIe控制器的基本初始化与功能配置。在U-Boot SPL阶段初始化配置PCIe，并采用Linux内核PCIe EP软件框架创建设备，既降低错失主机探测窗口风险，又充分利用PCIe EP软件框架。

首先，在U-Boot SPL阶段将SoC的PCIe控制器配置为端点模式，使设备尽早对外呈现有效的PCIe外设，同时，保障主机侧能够及时完成设备枚举与链路建立；其次，在Linux内核层面，通过PCIe EP软件框架创建并加载EPF设备驱动；最后，在用户空间启动密码卡守护进程，对外提供完整的密码算法服务，完成从硬件初始化到服务上线的全链路协同设计。本方案突破了传统架构在RAM容量、存储资源及计算能力方面的硬件限制，从而提升了系统的可维护性、实时监控能力与故障诊断水平，并支持细粒度日志记录等高级运维功能。

以i.MX8MM为例，将SoC的PCIe控制器配置为端点模式。首先，内核设备树禁用原本的根复合体模式设备节点，新增并启用端点设备节点；其次，配置内核将DesignWare PCIe Controller与i.MX8 PCIe Controller选项配置为端点模式。其中，DesignWare控制器作为PCIe EP软件框架中EPC库的底层驱动实现，负责PCIe物理链路的建立与寄存器级控制；而i.MX8特定驱动模块则承担EPF设备的资源分配、中断管理及平台相关功能的实现。

3.1.2　PCIe EP软件框架DMA传输实现

Linux 5.x内核PCIe EP软件框架实现的DMA传输，是指DMA Engine驱动利用SoC内部集成的通用DMA控制器进行数据搬运的过程[12-13]。EPF的DMA数据传输模型如图4所示。该机制仅支持设备内部外设间或内存到内存的数据传输，无法直接用于主机与PCIe端点设备之间的跨系统数据交换。若用于主机与PCIe密码卡间的通信，底层则会依赖I/O地址映射进行数据搬移。然而，I/O映射带宽低且延迟高，将成为PCIe密码卡的性能瓶颈。

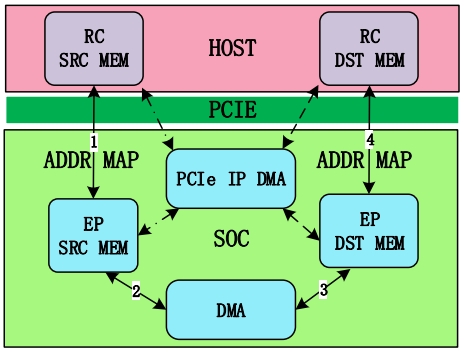

Fig. 4　EPF DMA model

图4　EPF的DMA数据传输模型

为充分释放SoC内置密码算法引擎的能力，亟需在Linux内核的PCIe EP软件框架中引入对PCIe IP集成DMA控制器的支持。实现路径主要有两种：①在DMA Engine驱动中支持PCIe IP集成DMA；②在PCIe EP软件框架中支持PCIe IP集成DMA。本文采用第二种路径，如图4中虚线所示。在选取的3款SoC平台中，i.MX8MM与RK3588的PCIe IP均集成了两个独立的DMA传输通道，而JH7110资料中未涉及PCIe控制器是否集成DMA功能。

3.2　基于重构PCIe EP软件框架的端点模式实现

3.2.1　PLDA PCIe IP介绍

Plda PCIe控制器架构如图5所示。PLDA PCIe IP由三层结构组成，分别是PCIe层、桥层和AXI(Advanced eXtensible Interface)层。

PCIe 层包括多个关键组件：PCIe 控制器、连接 PCIe 控制器与桥的 PCIe 发送/接收接口、用于访问 PCIe 配置空间的配置接口，以及管理低功耗模式和中断功能的杂项接口[14]。这些组件共同确保了 PCIe 链路的稳定建立与高效数据传输。桥层包含 Bridge 内部寄存器、最多 8 个独立 DMA 引擎模块、地址翻译模块及互连模块。其中，地址翻译模块负责在 AXI 接口与 PCIe 接口之间进行地址映射，而互连模块则负责处理输入输出流之间的互连与仲裁，以支持多通道并发操作。AXI 层集成了多种接口类型，包括 AXI4-Lite 主/从接口、主描述符接口、最多 4 个 AXI4 主接口、从接口及流接口。所有接口均按其实现顺序编号，便于系统集成与调试。

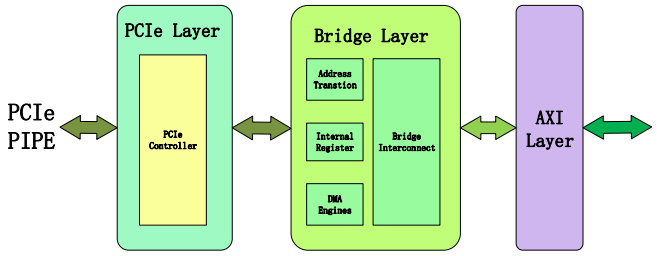

Fig. 5 PLDA PCIe controller architecture

图 5 Plda PCIe 控制器架构

3.2.2 PCIe EP 软件框架全新实现

首先，U-Boot 完成电源管理、时钟和复位模块的初始化；其次，操作 PCIe 寄存器，将 PCIe 控制器配置为端点模式，并借鉴 i.MX8MM 设备树在 JH7110 的内核设备树中新增一个类似设备节点；然后，在 Linux 内核中创建新的 EPC 设备，并对控制器及其物理层进行详细配置，实现 EPC 库中所有的函数（不包括 DMA 传输），以确保应用能够正常访问底层硬件；最终，在 Linux PCIe EP 软件框架中完成对 PLDA PCIe IP 的支持，并进行全面的系统级验证测试。由于未能获取到 JH7110 PCIe 集成 DMA 的相关信息，本文数据传输基于 I/O 映射的方式进行。尽管该方法对 PCIe 密码卡的算法性能无法完全发挥，但其 IO 性能达到了预期标准，进而验证了所提出的 PCIe EP 软件框架全新实现方案的可行性。

3.3 基于独立的 PCIe 配置镜像的端点模式实现

3.3.1 PCIe 配置镜像

为满足 PCIe 上电时序与 eMMC 规范，快速将 SoC PCIe 设置为端点模式，最优方案如下：开发 PCIe 初始化程序，编译专用镜像烧写至 NOR Flash，并将其配置为 SoC 默认启动介质。由于 RK3588 支持用户自定义 BootLoader 机制，因此，本文设计了专用引导程序 pcie_idb，运行于 SoC 内部 SRAM，在 BootROM 执行后立即加载。pcie_idb 镜像紧接标准引导镜像烧录至 NOR Flash，以确保 PCIe 链路建立后无缝转入正常启动流程。具体实现包括：①修改配置链接脚本，从而确保 pcie_idb 代码位于 SRAM 有效地址区间；②初始化基础硬件以支持调试；③配置 PCIe 电源域、时钟、复位、BAR 空间、配置空间及工作模式；④等待 LTSSM 完成链路训练，建立稳定连接；⑤编译生成 pcie_idb 镜像并烧录至 NOR Flash 起始位置，作为 BootROM 后首个执行实体。

3.3.2 PCIe 端点模式驱动

通过独立的 PCIe 配置镜像，系统在上电初期即完成与主机侧的设备识别及 PCIe 链路的稳定建立。随后，进入 Linux 内核阶段，为其编写字符设备驱动[15]。

由于 PCIe 物理链路在内核启动前已由早期引导程序建立，内核驱动无需参与链路初始化，因此，其主要职责聚焦于功能模块的资源管理与接口抽象。具体包括：对 PCIe BAR 空间的内存映射、DMA 通道的初始化与数据传输管理、中断请求的注册与处理机制，以及字符设备文件操作接口的实现，为用户空间提供了标准化的设备访问路径[16]。

针对 RK3588 SoC 的硬件特性，其片上集成最多 8 通道并行配置的密码算法引擎，同时，在驱动设计中需通过独立的配置选项来进行功能区分与资源调度。此外，用户空间应用程序还可获得统一、抽象的密码服务接口，通过透明访问底层硬件加速资源，从而实现高性能、低开销的加解密操作。

3.3.3 镜像烧录

为了确保 PCIe 配置镜像能够优先加载并完成链路初始化，需要将 NOR Flash 的启动优先级设置为高于 eMMC。在系统上电后，SoC 会首先从 NOR Flash 中读取并执行引导程序，从而实现对 PCIe 功能的早期控制。

在进行镜像烧录时，需要根据实际使用的存储介质类型进行对应配置。其中，PCIe 初始化专用镜像 pcie_idb 和用于后续正常启动流程的镜像必须烧录至 NOR Flash，以保证系统能够按照预期顺序启动并完成 PCIe 链路建立。

3.4 主机测试

在 SoC 操作系统启动后，系统首先挂载 configfs 虚拟文件系统，并通过其接口动态创建并启用 PCIe 端点功能设备。在正常情况下，主机操作系统能够识别并枚举新增的 PCIe 设备。

内核原生工具仅提供基础的测试功能，主要用于调试，且缺乏高性能数据交互能力。本文对主从端驱动及用户空间组件进行了重构与扩展，在保留链路自检与基准测试功能的基础上，提高了通信的灵活性与可扩展性，增加了自定义协议封装，从而满足了密码卡在复杂应用场景下的多样化需求。EPF 与主机测试模型如图 6 所示。

4 实验与分析

在实验阶段，对基于 SoC PCIe 端点模式实现的密码卡进行基本功能测试和异常测试。其中，异常测试主要通过构造异常场景，验证其状态监控与异常响应机制的有效性。鉴于密码算法性能高度依赖 SoC 内置的硬件加速引擎，本文更关注在 PCIe 端点模式下的整体数据传输能力与

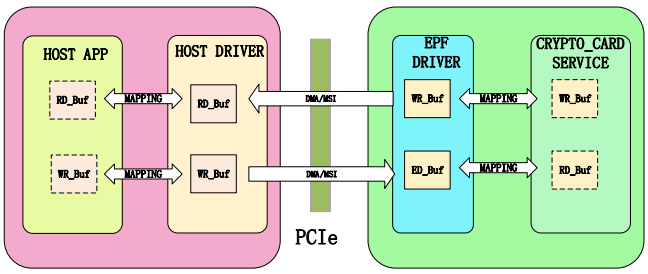

Fig. 6　EPF test model

图6　EPF测试模型

响应效率[17]。重点对基于i.MX8MM和RK3588实现的密码卡方案(即方案二与方案四)开展了全面测试。

4.1　实验环境

本实验采用基于X86架构的主机平台，除使用搭载i.MX8MM和RK3588的PCIe密码卡外，还选取了三未信安的SJK1727 V2.0-A型号PCIe密码卡作为对比测试对象[18]。实验配置如表3所示。

Table 3　Experimental setup

表3　实验配置

实验配置				
测试平台	平台	PCIe能力	操作系统	CPU
	X86	GEN2 X4	ubuntu22.04	i5-4590
测试板卡	板卡SOC	PCIe	支持密码算法	
	RK3588	GEN3 X4	AES、3DES、ECC、SM2、SM3、SM4	
	SJK1727 V2.0-A	GEN2 X4	SM1、SM4、SM7、AES、SM3、RSA、SM2、SM9	

4.2　PCIe信道性能

4.2.1　i.MX8MM PCIe IO性能

本文使用3.4测试方法，在主机侧对EPF设备进行功能与性能测试。通过在用户空间对设备I/O地址空间进行内存映射并执行读写操作，评估其通信性能与传输稳定性。i.MX8MM实测IO性能如表4所示。

Table 4　IO performance

表4　IO性能表

包长	IO写	IO读
1B	1.00Kbps	1.00Kbps
16B	16.00Kbps	16.00Kbps
128B	128.10Kbps	128.80Kbps
1KB	1.02Mbps	1.03Mbps
4KB	4.09Mbps	4.12Mbps
1MB	255.20Mbps	18.80Mbps
10MB	292.00Mbps	18.80Mbps

从表4可以看出，随着包长的增加，IO读写速率逐步上升。从1B到4KB可以发现，性能随包长增大呈线性增长；IO读性能的上限极低，甚至在1MB包长时速率仅能达到18.8M bps。这表明，采用I/O地址映射方式进行数据传输，其速率完全不可接受。

4.2.2　PCIe DMA性能

为最大限度发挥DMA传输性能，主机与设备的驱动、应用均采用轮询机制进行测试，避免中断延迟对吞吐率造成影响。其中，针对i.MX8MM平台，开展单通道DMA性能测试；对于RK3588平台，则分别测试其单通道与双通道DMA的数据传输能力。i.MX8MM和RK3588 PCIe DMA实测结果如表5、表6所示。

Table 5　i.MX8MM PCIe DMA performance

表5　i.MX8MM PCIe DMA性能表

包长	IO写	IO读
1B	960.00Kbps	1.20Mbps
16B	13.60Mbps	19.20Mbps
128B	106.40Mbps	154.00Mbps
1KB	640.00Mbps	696.00Mbps
4KB	1.54Gbps	1.27Gbps
1MB	2.62Gbps	1.67Gbps

Table 6　RK3588 PCIe DMA performance

表6　RK3588 PCIe DMA性能表

包长	DMA单通道写	DMA单通道读	DMA双通道写	DMA双通道读
1B	1.53Mbps	1.55Mbps	2.40Mbps	2.55Mbps
16B	24.49Mbps	24.83Mbps	38.00Mbps	40.00Mbps
128B	196.00Mbps	198.00Mbps	309.00Mbps	324.00Mbps
1KB	1.56Gbps	1.17Gbps	2.37Gbps	2.17Gbps
4KB	3.66Gbps	4.36Gbps	6.74Gbpss	6.34Gbps
8KB	5.75Gbps	5.69Gbps	9.17Gbpss	9.89Gbps

由表5可知，随着数据包长的增加，DMA读写速率逐步上升。包长从1B到128B时，DMA读写性能与包长呈线性关系；在包长为4KB时，DMA读写性能分别达到1.27Gbps和1.54Gbps。与表4对比可见，在包长1B时，单通道DMA读写性能是IO的近千倍；即便包长增加到1Mb，单通道DMA写性能仍是是IO的近十倍，读性能比IO读快了的近百倍。这表明，使用DMA进行数据传输的优势巨大，更适合高带宽、低延迟的传输场景[19]。

由表6可知，RK3588的PCIe工作在GEN2 X4状态下，其单通道DMA读写性能的增长方式与表5呈现的基本一致。尽管使用了4个Lane，但单通道DMA的读写性能远未达到表5的四倍，在数据包长度超过1KB时，多Lane的优势才逐渐显现。双通道DMA的读写性能在数据包小于1KB时仅为单通道DMA性能的1.5倍左右，而在数据包达到4KB后，双通道DMA的性能优势变得更加明显。

4.3　与现有产品对比

4.3.1　RK3588实现的板卡

图7为基于RK3588 SoC研制的PCIe密码卡。板卡采用PCIe3.0 X4接口与主机高速互连，搭载RK3588处理器，依托ARM64多核架构及硬件密码加速模块。同时，板载LPDDR4与eMMC，以充裕的内存和固态存储保障系统平稳运行。

4.3.2　与SJK1727 V2.0-A对比

RK3588在用户空间通过调用libkcapi接口库实现对硬件密码算法的访问。基于RK3588的PCIe密码卡工作在PCIe Gen2 x4状态下，2MB数据包时DMA写性能为1.44GB/s，达到理论带宽的72%。基于RK3588的PCIe密码卡和三未信安SJK1727V2.0-A密码卡的最优性能，如图

Fig. 7 Board based on RK3588

图7 基于RK3588的板卡

7所示。

Table 7 Cryptographic algorithm performance

表7 密码算法性能表

算法	模式	RK3588板卡/Mbps	SJK1727 V2.0-A/Mbps
SM4	ECB	246 7	550 0
	CBC	700	550 0
AES-256	ECB	326 4	700 0
	CBC	139 5	700 0
HASH	SM3	146 4	550 0
	SHA256	138 8	550 0
	SHA512	227 3	550 0

由表7可知，基于RK3588的PCIe密码卡在处理2MB包长数据时，虽能展现其算法引擎性能，但与SJK1727 V2.0-A在7.5KB包长下的性能相比，仍存在全方位的差距。结合表6可知，性能落后的主要原因是SoC内部集成的密码算法引擎远低于高端安全专用芯片。

4.4 异常测试

板卡异常测试分为状态监控与异常场景模拟两部分。其中，状态监控实时采集板卡电压、温度、电源管控和加密引擎调度等关键指标。异常场景模拟搭建多类故障场景并开展验证：①高温工况下风扇自动提速，若高温状态持续5 s未回落，电源启动高温保护、CPU切换运行模式，系统暂停业务、LED发出告警，所有业务请求均返回高温报错；②以绝缘物料包覆PCIe金手指信号引脚，链路协商失败，LED提示链路断开；③使用劣质PCIe延长线致使链路降速，系统将链路状态、协商寄存器数据保存至指定文件，LED同步显示实际链路速率；④向驱动注入算法引擎超时故障后，密码守护进程标记故障引擎，并触发其余引擎自检，依据自检结果判定是否保留部分密码服务能力。

5 结语

本文设计并实现了以SoC为PCIe端点设备的国密密码卡整体方案。区别于传统主控、算法、通信分立的三段式硬件架构，本方案采用单颗SoC集成控制、数据交互与密码运算模块，简化硬件电路，有效提升了设备稳定性与可维护性。平台搭载嵌入式Linux系统，实现设备状态监测、故障处理、固件热升级与日志留存等管理功能，进一步提升了PCIe密码卡的智能化程度与复杂工况下的运行可靠性。针对业务需求差异，本文完成了4种软件架构的开发。具体包括：第一种沿用经典外设开发模式，结构精简，适合高稳定运行场景；第二种和第三种将DMA引入PCIe EP软件框架，把密码算法程序改造为用户态守护进程，支撑模块动态加载、安全隔离与故障自愈；第四种依托SoC镜像定制能力，优化PCIe链路初始化与设备枚举逻辑，增强对各类主机平台的兼容性，保障链路快速稳定建立，满足快速启动与多平台适配需求。基于RK3588完成验证开发，并与SJK1727 V2.0-A商用密码卡开展对比测试。测试结果表明，该方案在运维管理功能上优势突出，不足在于密码运算性能弱于专用安全芯片，但SoC架构的PCIe密码卡依然展现出巨大的潜力。

本文提出的方案和实现仍有进一步改进的空间，尤其是所使用的三款SoC芯片均未获得商用密码认证，因此，基于SoC实现的PCIe密码卡不符合GM/T 0028-2024《密码模块安全要求》规范。此外，本文未解决因引入操作系统导致的中断实时性问题，也未涉及集成抗量子密码算法研究等前沿领域[20-21]。同时，对于如何通过软件编程重构密码模块以实现自定义密码算法的支持，亦未做讨论。未来的工作将集中在以下4个方面：①优化现有设计以满足商用密码认证；②提升中断处理的实时性，确保系统的高可靠性；③探索集成抗量子密码算法，增强系统对未来威胁的防御能力；④研究通过软件编程实现密码模块的灵活配置与自定义硬件密码算法，进一步提升密码卡的适应性和安全性。

参考文献：

[1] XIE Z X, DONG K X, ZHEN J. Introduction to domestic commercial cryptographic algorithms and their related standards[J]. China Standards Review, 2020(6): 12-14, 23.
谢宗晓，董坤祥，甄杰. 国产商用密码算法及其相关标准介绍[J]. 中国质量与标准导报，2020(6)：12-14，23.

[2] Technical specification for PCI cryptographic module[S]. Beijing: National Cryptography Administration, 2022.
PCI密码卡技术规范[S]. 北京：国家密码管理局，2022.

[3] GONG G W. Design and research of PCIe-Based data encryption card utilizing the SM4 national cryptographic algorithm[D]. Taiyuan: North University of China, 2023.
弓国伟. 基于PCIe的SM4国密算法数据加密卡设计与研究[D]. 太原：中北大学，2023.

[4] GAO D F. The design and implementation of PCI password card[D]. Zhengzhou: Zhengzhou University, 2013.
高东飞. PCI密码卡的设计与实现[D]. 郑州：郑州大学，2013.

[5] SU Z Y. The design and realization of high speed PCI-Express encryption card based on FPGA and DSP[D]. Jinan: Shandong University, 2012.
苏振宇. 基于FPGA和DSP的PCI-E高速密码卡设计与实现[D]. 济南：山东大学，2012.

[6] PENG S. The design and implementation of PCIE encryption card system based on the security ASIC password chip[D]. Shenyang: Liaoning University, 2017.
彭帅. 基于安全ASIC密码芯片PCIE加密卡系统的设计与实现[D]. 沈阳：辽宁大学，2017.

[7] PCI express base specification revision 3.0a[S]. Palo Alto: PCI SIG,

2015.

[8] PCI SIG. PCI express® base specification revision 7.0[EB/OL]. https://members.pcisig.com/wg/PCI-SIG/document/previewpdf/22465.

[9] KernelLinux. PCI endpoint framework[EB/OL]. https://docs.kernel.org/PCI/endpoint/index.html.

[10] PCI Express® card electromechanical specification revision 3.0[S]. Palo Alto:PCI SIG,2013.

[11] JEDEC Solid State Technology Association. JESD84-A441. Embedded multimediacard(e•mmc) e•mmc/card product standard, high capacity, including reliable write, boot, sleep modes, dual data rate, multiple partitions supports, security enhancement, background operation and high priority interrupt (mmca, 4.41)[S]. Arlington:JEDEC,2010.

[12] YU X B, WU J W, XIA H,et al. PCIe bus transmission bandwidth optimization in embedded heterogeneous intelligent computing system[J]. Journal of Computer Applications,2025,45(9):2913-2918.
喻绪邦,吴济文,夏宏,等. 嵌入式异构智能计算系统的PCIe总线传输带宽优化[J]. 计算机应用,2025,45(9):2913-2918.

[13] LI S L,JIANG H X,FU W J,et al. Design of DMA controller for multi-channel transmission system based on PCIe[J]. Journal of Computer Applications,2017,37(3):691-694,716.
李胜蓝,姜宏旭,符炜剑,等. 基于PCIe的多路传输系统的DMA控制器设计[J]. 计算机应用,2017,37(3):691-694,716.

[14] HUANG Z Z. Design and verification of a PCIe controller in an ARM SoC system[J]. Information Communications,2018(8):89-90.
黄振忠. 一种ARM Soc系统中PCIe控制器的设计与验证[J]. 信息通信,2018(8):89-90.

[15] HE W W. Design and implementation of high-speed data transmission system with PCIE bus base on FPGA architecture[D]. Chengdu:University of Electronic Science and Technology of China, 2016.
贺位位. 基于FPGA结构高速PCIe总线传输系统设计与实现[D]. 成都:电子科技大学,2016.

[16] KAVIANIPOUR H, MUSCHTER S, BOHM C. High performance FPGA-based DMA interface for PCIe[J]. IEEE Transactions on Nuclear Science, 2014, 61(2): 745-749.

[17] ZHAO K, LIU W, MU Q, et al. PCIe cryptographic accelerator based on domestic FPGA[C]// 2022 International Conference on Computing, Communication, Perception and Quantum Technology (CCPQT), 2022: 25-31.

[18] TANG L S, DOU T R, SANG H B, et al. Software virtualization of cryptographic card based on I/O front and back end model[J]. Computer Systems & Applications,2022,31(1):286-294.
唐乐爽,窦同锐,桑洪波,等. 基于I/O前后端模型的密码卡软件虚拟化[J]. 计算机系统应用,2022,31(1):286-294.

[19] SUN Z P, ZHOU K J. High-performance FPGA-GPU-CPU heterogeneous programming architecture based on PCIe[J]. Computer Engineering & Science,2021,43(4):641-651.
孙兆鹏,周宽久. 基于PCIe的高性能FPGA-GPU-CPU异构编程架构[J]. 计算机工程与科学,2021,43(4):641-651.

[20] DONG J K,HUANG Y H,FU Y S,et al. Research progress in high-performance cryptographic computing technology based on heterogeneous multicore GPU[J]. Journal of Software,2024,35(12):5582-5608.
董建阔,黄跃花,付宇笙,等. 基于异构多核心GPU的高性能密码计算技术研究进展[J]. 软件学报,2024,35(12):5582-5608.

[21] WANG D L. Design and implementation of a national standard secure gateway based on hardware acceleration[D]. Xi′an: Xidian University, 2024.
王丁磊. 基于硬件加速的国密安全网关的设计和实现[D]. 西安:西安电子科技大学,2024.

(责任编辑:陈维捷)