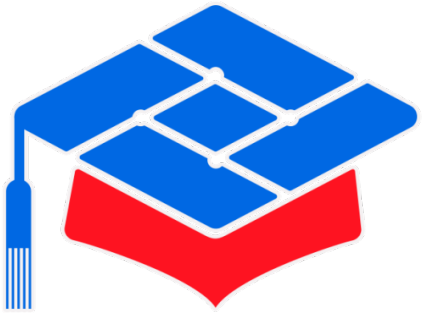

系统设计报告

基于龙芯 SoC 的后量子密码与矩阵运算加速系统

参赛队伍： _____

赛　题： 龙芯中科杯

竞赛名称： CICIEC 2026 第十届全国大学生集成电路创新创业大赛

所属赛区： _____

提交日期： 2026 年 7 月

文档版本： d456463（当前 evidence）—2026-07-15

目录

1 绪论 **4**

1.1 设计背景与目标 4

1.2 设计原则与创新点 4

1.3 代码规模与开发环境 5

2 系统架构设计与软硬件功能划分 **5**

2.1 系统总体架构 5

2.2 地址空间分配 6

2.3 时钟域划分与跨时钟域设计 7

2.4 复位架构 8

2.5 中断系统 8

2.6 软硬件功能划分 9

3 芯片前端设计与实现 **9**

3.1 DMA 子系统设计 9

3.2 矩阵乘法加速器设计 10

3.3 矩阵批量序列器设计 11

3.4 FFT 加速器设计 12

3.5 ML-KEM 加速器系统集成 13

4 验证方案 **13**

4.1 分层验证策略 13

4.2 验证工具链与环境 14

4.3 验证覆盖范围 15

5 实现结果与分析 **15**

5.1 FPGA 资源利用 15

5.2 时序性能 16

5.3 存储子系统 17

5.4 功耗分析 18

5.5 矩阵乘法性能分析 18

5.6 FFT 性能分析 19

5.7 ML-KEM 性能分析 19

5.8 跨时钟域与系统集成验证 20

6 设计讨论与展望 **20**

6.1 设计局限与约束 20

6.2 安全设计考量 21

6.3 未来优化方向 22

7 结论 **22**

1 绪论

1.1 设计背景与目标

随着量子计算技术的快速发展，传统公钥密码体系面临严峻挑战。美国国家标准与技术研究院于 2024 年正式发布的 FIPS 203 标准确立了 ML-KEM 作为后量子密钥封装机制的地位，标志着后量子密码从学术研究向工业应用的转折点。与此同时，矩阵乘法作为科学计算、信号处理与机器学习的基础运算，其硬件加速在嵌入式计算场景中具有广泛需求。本届集成电路创新创业大赛龙芯中科杯赛题要求参赛者在 OpenLA500 处理器平台上，设计并实现包含矩阵乘法加速器与 ML-KEM 密码 IP 的完整 SoC 系统，在资源受限的 FPGA 器件上同时满足功能正确性与性能要求。

本设计的核心目标是在 Xilinx Artix-7 xc7a200tfbg676-1 FPGA 器件上构建一个功能完整的异构计算 SoC 系统，具体包括三项设计内容。第一，设计一个支持 4x4 无符号矩阵乘法的硬件加速器，不使用 DSP 资源，完成 5000 组矩阵计算的批量处理任务，并通过 CRC32 校验结果正确性。第二，根据 FIPS 203 标准实现全参数可配置的 ML-KEM 硬件 IP，覆盖 ML-KEM-512、ML-KEM-768 与 ML-KEM-1024 三个安全等级，支持密钥生成、封装与解封装全部三种操作。第三，设计通用 DMA 子系统与批量任务调度器，将多个计算加速器有机集成到统一的 AXI 总线架构中，实现 CPU 控制、DMA 搬运与加速器计算的流水化协同。

1.2 设计原则与创新点

系统设计遵循四条基本原则。第一条原则是模块化分离原则。将矩阵乘法、FFT 和 ML-KEM 三个计算加速器设计为相互独立的 AXI 外设，各自拥有独立的地址空间、控制寄存器和中断通道，使得每个加速器可单独验证、单独优化，系统集成时仅通过跨时钟域隔离与地址解码完成互连。第二条原则是资源复用原则。ML-KEM 内部算法涉及 Keccak-f[1600] 置换、SHA3/SHAKE 哈希族、NTT 数论变换、Barrett 模乘、CBD 中心二项分布采样以及压缩解压缩等多种运算算子。设计中采用单个 Keccak 核在顶层有限状态机的时间分片调度下被 SHA3/SHAKE、SampleNTT 和 CBD 三个消费者分时复用，NTT 正变换与逆变换共享同一组蝶形单元流水线，解封装流程中的重加密阶段直接复用封装状态序列。第三条原则是数据搬运与计算解耦原则。通用 DMA 子系统独立于计算加速器，通过内存到设备、设备到内存以及设备本地端点三种传输模式，将数据搬运从 CPU 寄存器逐字读写中解放出来，使计算流水线的瓶颈从总线事务转移到算术吞吐率。第四条原则是批量调度与双缓冲原则。矩阵乘法批量序列器通过双 Bank 乒乓机制，将批量任务调度、数据加载、计算启动与结果写回四个阶段完全流水化，一个 Bank 执行计算时另一个 Bank 进行数据加载或结果写回，从而隐藏 ExtRAM 访问延迟。

本设计的主要创新点体现在三个方面。首先，提出了一套高效的跨时钟域互连方案，在 CPU 时钟域、系统总线时钟域和 ML-KEM 算法时钟域之间采用全五通道 AXI 双时钟 FIFO 进行时钟域隔离，同时配合严格的异步复位释放链和中断同步器，在保持各时钟域独立约束的前提下保证了数据与控制信号的可靠传递。其次，针对竞争评分机制对批量处理延迟的极致要求，设计了快速通道模式，在保证 CRC32 增量计算正确性的前提下跳过每组结果的逐

字写回，将批量矩阵乘法的端到端延迟推进到毫秒量级。最后，ML-KEM 解封装中的隐式拒绝路径采用了常量时间比较策略，整个重加密密文与输入密文的逐字节比对不进行提前退出，从根本上杜绝了时序侧信道信息泄露，这是后量子密码实现中至关重要的安全设计考量。

1.3 代码规模与开发环境

本设计的基础 SoC 平台来自龙芯中科提供的竞赛发布包，其核心处理器为 OpenLA500。OpenLA500 是一款实现了龙芯架构 32 位精简版指令集（LoongArch32R）的处理器核，采用单发射五级流水线架构，分为取指、译码、执行、访存和写回五个流水级。处理器核内置两路组相联的指令和数据 Cache、32 项 TLB 以及简易分支预测器，对外通过标准 AXI 接口与系统总线互连。该处理器已在 0.13 微米工艺下完成流片验证，实测频率 100 MHz 时的 Dhrystone 分数为 0.78 DMIPS/MHz，CoreMark 分数为 2.75 CoreMark/MHz。在软件生态方面，U-Boot、Linux 5.14、UCore 和 RT-Thread 等常用系统软件已完成对该处理器的适配。

系统 RTL 代码采用 SystemVerilog 编写，覆盖 SoC 顶层集成、三个计算加速器及其子模块、DMA 子系统、跨时钟域桥接模块以及相关的仿真测试平台。软件部分包括运行于 OpenLA500 CPU 上的 C 语言驱动程序和测试程序，涵盖矩阵乘法评测、FFT 功能验证以及 ML-KEM 全参数系统级诊断测试。FPGA 综合与布局布线使用 Xilinx Vivado 2019.2 工具链完成，目标器件为 xc7a200tfbg676-1。RTL 仿真采用 Xilinx XSIM 进行波形验证与定时仿真，软件交叉编译使用龙芯 GCC 工具链（loongarch32r-linux-gnusf-gcc 8.3）生成 MIPS32 可执行文件。RTL 静态检查通过开源 lint 工具完成。

2 系统架构设计与软硬件功能划分

2.1 系统总体架构

系统的顶层模块围绕 AXI4 总线互连矩阵组织所有功能模块。该互连矩阵配置为两个主端口、八个从端口的工作模式，运行于系统时钟域。两个主端口分别连接 OpenLA500 处理器 CPU 和数据搬运引擎 DMA，八个从端口挂载系统内存、外设以及三个计算加速器的控制接口。整个系统划分为三个异步时钟域，由单一 PLL 产生。CPU 时钟域采用 33.333 MHz 运行频率，承载 OpenLA500 处理器核心、一级缓存与 AXI 主设备接口。系统时钟域采用 50 MHz 运行频率，承载总线互连矩阵、DMA 子系统、FFT 加速器、矩阵乘法加速器以及 UART、SRAM 控制器等基础外设。ML-KEM 算法时钟域采用 76.923 MHz 运行频率，承载 ML-KEM 的 AXI 从设备接口、算法控制器、Keccak 置换核、NTT 单元以及多项式运算模块的全部数据通路。

系统的总体架构如图1所示。CPU 通过跨时钟域桥接模块接入系统总线，该桥接模块在 AXI 协议的五条通道上各部署一个双时钟先进先出缓冲队列，将 CPU 时钟域的读写事务安全传递至系统时钟域。DMA 子系统以第二主设备身份直接接入总线互连矩阵，在系统时钟域内向任意从设备发起 AXI 突发传输。三个计算加速器中，矩阵乘法加速器和 FFT 加速器完全运行于系统时钟域，ML-KEM 加速器则通过第二个跨时钟域桥接模块从系统时钟域跨越至 ML-KEM 专用算法时钟域。这种分层跨时钟域架构的设计决策基于各模块的计算特征

频率需求：矩阵乘法加速器关注批量吞吐率但内部流水线深度较浅，50 MHz 的系统时钟已足够满足其时序要求；ML-KEM 算法核心包含深度流水化的 Keccak 轮函数和 NTT 蝶形单元，需要更高的时钟频率以在串行 FSM 调度模式下获得可接受的单次操作延迟。

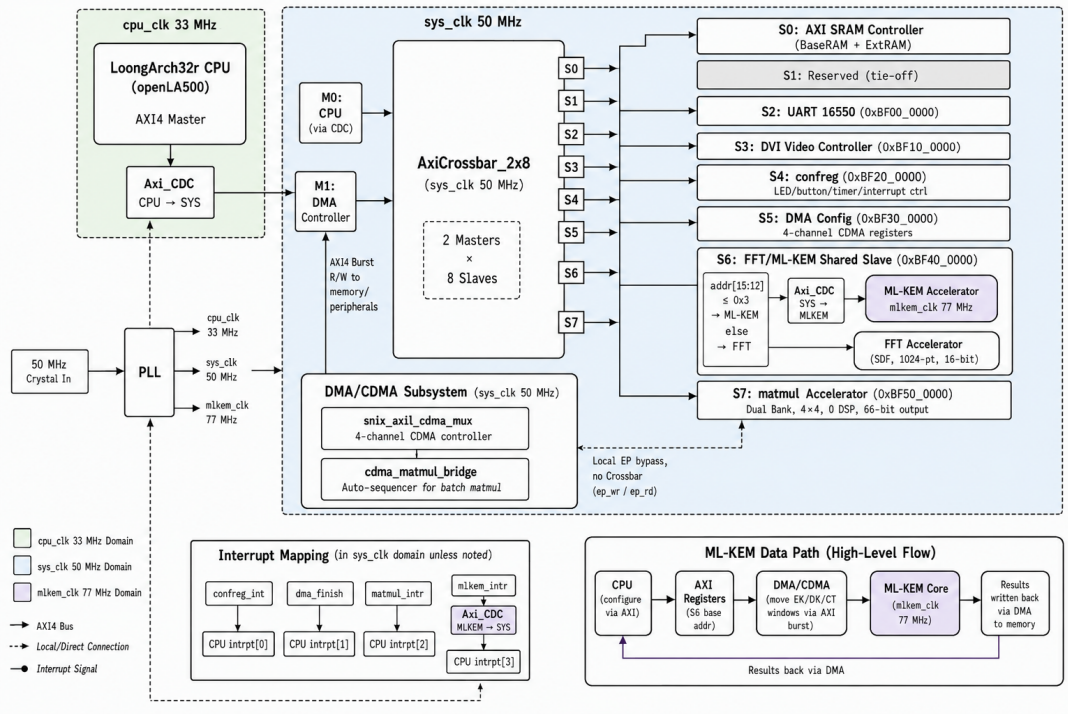

图 1: 系统总体架构图

2.2 地址空间分配

系统采用统一的物理地址空间，各计算加速器通过固定的从端口挂载在总线互连矩阵上，每个加速器拥有独立的地址范围。竞赛发布包中总线互连矩阵的八个从端口已有既定分配：从端口零连接片上 SRAM（基地址 0x1c00_0000，8 MB 空间），从端口一保留，从端口二连接 AXI 转 APB 桥后挂载 UART（基地址 0x1f00_0000），从端口三连接 DVI 显示控制器（基地址 0x1f10_0000），从端口四连接系统配置与中断控制器（基地址 0x1f20_0000），从端口五连接 DMA 从设备端口（基地址 0x1f30_0000），从端口六连接 FFT 加速器（基地址 0x1f40_0000），从端口七保留供参赛者扩展使用（基地址 0x1f50_0000）。

完整的地址空间分配如表1所示。

表 1: AXI 总线互连矩阵从端口地址空间分配

从端口	物理基地址	模块	用途与说明
S0	0x1c00_0000	片上 SRAM	8 MB 空间，BaseRAM（bit22=0，存放程序）与 ExtRAM（bit22=1，存放数据）
S1	0x0000_0000	保留	发布包中未分配，保留供扩展使用
S2	0x1f00_0000	UART	AXI 转 APB 桥接串口，波特率 115200，用于调试输出与评测通信
S3	0x1f10_0000	DVI 显示控制器	视频输出，800×600 分辨率，RGB 3-3-2 位宽
S4	0x1f20_0000	Confreg	按键、拨码开关、LED、数码管、中断控制器、外部定时器
S5	0x1f30_0000	DMA 控制窗口	四通道 DMA 寄存器（0x000–0x1FF）与矩阵批处理序列器（0x200–0x21F）
S6	0x1f40_0000	FFT/ML-KEM 共享窗口	ML-KEM 寄存器与三组批量数据窗（低 16 KB），FFT 数据窗与 CSR（高 16 KB）
S7	0x1f50_0000	矩阵乘法加速器	本设计新增：控制状态、CRC32、双 Bank 矩阵缓冲区

2.3 时钟域划分与跨时钟域设计

系统三个时钟域之间的关系在引脚约束文件中通过异步时钟组约束声明为彼此独立的时钟域，时序分析引擎对跨域路径不进行建立保持时间检查，时序收敛的职责完全落在跨时钟域电路的设计正确性上。CPU 到系统总线的跨域路径承载 AXI 读写事务，采用全五通道双时钟先进先出缓冲队列方案。写地址通道、写数据通道和读地址通道的缓冲队列写入端工作在 CPU 时钟域，读出端工作在系统时钟域；写响应通道和读数据通道的缓冲队列写入端工作在系统时钟域，读出端工作在 CPU 时钟域。每条通道的缓冲队列独立管理自身的满空标志，通过有效—就绪握手协议实现跨域反压传递。这种设计保证了 AXI 协议的事务顺序性属性在跨域后不丢失，CPU 端提交的写事务在系统总线端按顺序可见，读事务的响应数据同样按顺序返回。

系统总线到 ML-KEM 算法域的跨域路径负载更大，除了承载寄存器级控制状态读写外，还需承载公钥、私钥和密文的批量数据窗口传输。设计中在 FFT/ML-KEM 共享地址解码模

块内完成地址解码和目标选择后，对 ML-KEM 分支同样部署了全五通道双时钟先进先出缓冲队列，与 CPU 到系统总线的方案保持设计一致性。各通道的队列深度经过计算确保在最长的 INCR 突发传输下不会溢出，读数据通道的队列深度同时考虑了 ML-KEM 内部块存储器的同步读取延迟。

ML-KEM 到 CPU 的中断信号属于单比特状态信号，从 ML-KEM 算法时钟域同步到系统时钟域采用两级触发器串联的经典同步器方案。中断信号设计为电平型而非脉冲型，由 ML-KEM 核心在完成时拉高，由 CPU 软件通过写一清零操作释放，这种设计避免了跨域脉冲捕捉的可靠性问题。由于中断信号在源时钟域的保持时间远大于目标时钟域的采样周期，两级同步器的亚稳态传播概率可以忽略不计。

时钟域划分与频率规划总结于表2。

表 2: 时钟域划分与频率

时钟域	周期 (ns)	频率 (MHz)	主要负载
cpu_clk	30.000	33.333	OpenLA500 CPU、Cache、AXI 主设备接口
sys_clk	20.000	50.000	总线互连矩阵、DMA、FFT、矩阵乘法、外设
mlkem_clk	13.000	76.923	ML-KEM AXI 前端、算法控制器、Keccak、NTT

2.4 复位架构

系统的复位网络遵循严格的分层异步释放同步撤离原则。外部板级复位信号经 PLL 锁定指示信号门控后，按照系统域、ML-KEM 域、CPU 域的顺序逐级释放。每一级使用复位同步器模块实现异步置位、同步撤离：复位置位时两级触发器同时异步清零，复位撤离时在目标时钟域内经过两个时钟周期的同步延迟后释放。释放顺序的级联关系由前一级的同步撤离信号作为后一级的释放门控，确保后续时钟域的复位不会超前于前级释放。仿真中实测的释放时序与设计预期一致：系统域复位在 PLL 锁定后约 2.030 us 释放，随后 ML-KEM 域复位在约 2.070 us 释放，最后 CPU 域复位在约 2.085 us 释放。CPU 端首次 AXI 读地址有效信号出现在约 2.205 us，与复位释放之间的间隔满足 AXI 协议对复位撤离到首次事务的时序要求。仿真中还验证了复位释放前各 AXI 通道上不存在未完成的事务，所有 valid 信号在复位期间保持为零，确保上电过程不会产生虚假总线事务。

2.5 中断系统

系统的中断聚合逻辑将五个中断源合并为单路 CPU 中断输入。五个中断源分别为：DMA 完成中断、矩阵批量序列器完成中断、矩阵乘法加速器完成中断、ML-KEM 算法完成中断以及系统配置寄存器通用中断。所有中断信号均为电平型，由中断源在事件发生时拉高，由 CPU 软件通过写一清零操作在中断服务程序中释放。性能测试程序出于零开销考虑，实际采用轮询方式检查各模块的状态寄存器中的完成标志位，而非依赖中断处理流程的开销。

2.6 软硬件功能划分

系统的软硬件功能划分遵循硬件负责计算密集型与数据搬运任务、软件负责控制调度与结果验证的原则。硬件层承担的功能包括：矩阵乘法的 4x4 无符号乘积与累加运算，FFT 的 1024 点 Radix-2^2 单路径延迟反馈流水线计算，ML-KEM 全流程的 Keccak 置换、NTT 变换、多项式乘法与压缩解压缩运算，DMA 的四通道 AXI 突发传输与内存到设备的端点搬运，以及矩阵批量序列器的 5000 组任务流水调度。硬件层不涉及任何算法参数的选择逻辑，所有安全等级和操作类型的配置均由软件写入控制寄存器完成。

软件层的职责位于三个抽象层次。最低层为寄存器驱动层，实现各加速器控制器寄存器的 C 语言结构体映射，封装启动、等待完成、读取结果等基本操作。中间层为 DMA 调度层，管理 DMA 通道的分配、源地址与目标地址的计算以及传输长度的配置。最高层为测试与验证层，构造测试输入数据，启动硬件计算流水线，等待完成信号，将硬件计算结果与软件参考模型进行比对，并通过串口输出测试结论和性能指标。

这种分层功能划分使系统行为在验证时具有良好的可观测性与可调试性。任何一个加速器的功能正确性可以通过软件层切换至寄存器逐字读写模式进行独立步进验证，然后再通过 DMA 批量模式验证完整流水线的高性能运行。ML-KEM 的功能验证在三种数据传输路径下分别进行了测试：纯软件参考模型路径用于确认算法逻辑正确性，CPU 寄存器逐字节加载路径用于验证 AXI 从设备接口与跨时钟域的数据通路完整性，DMA 批量窗口路径用于验证端到端系统级数据流与性能。

矩阵乘法评测程序在功能上进一步体现了软硬件协作的设计思想。CPU 负责配置序列器的源基地址、目标基地址、任务组数以及跨步参数，然后启动序列器。序列器作为纯硬件模块接管后续的全部 5000 组数据处理：逐一申请 DMA 通道完成源矩阵从 ExtRAM 到加速器 Bank 缓冲区的加载，触发计算，等待完成脉冲，申请 DMA 通道完成结果矩阵从加速器 Bank 缓冲区到 ExtRAM 的写回，并以增量方式实时更新 CRC32 校验值。CPU 在所有组任务完成信号触发后，仅需从加速器的 CRC32 寄存器读取最终校验值并通过串口输出，数据的完整搬运与计算路径完全由硬件流水线独立执行。这种软件配置加硬件执行的划分模式最大化地降低了对 CPU 吞吐能力的依赖，使性能瓶颈从 CPU 的寄存器访问延迟转移到硬件加速器与 DMA 之间的数据搬运吞吐率。

3 芯片前端设计与实现

3.1 DMA 子系统设计

通用 DMA 子系统是本系统中连接 CPU 控制面与加速器数据面的关键桥梁。其核心通道管理模块管理四个逻辑通道，各通道拥有独立的源地址、目的地址、传输长度、控制字、状态字、传输模式以及端点配置寄存器。四通道之间通过轮转仲裁器共享一个底层 AXI 内存到内存传输引擎。轮转仲裁器维护一个当前通道指针，在通道请求到达且引擎空闲时锁定通道，完成当前传输后释放引擎并推进指针至下一通道。

DMA 子系统支持三种传输模式以适应不同的数据搬运场景。内存到内存模式用于传统的源地址到目的地址 AXI 突发传输，传输引擎按照 IDLE、PREP1、PREP2、AR、READ、

AW、WRITE、WAIT_BRESP 的状态序列执行完整的 AXI 读后写流程。内存到端点模式用于将 ExtRAM 中的矩阵数据直接写入矩阵乘法加速器的本地输入端口，数据传输使用加速器内部的端点地址空间而非 AXI 从设备地址空间，避免了总线互连矩阵的地址解码延迟与可能的仲裁等待。端点到内存模式反之，将加速器计算结果从本地输出端口直接写回 ExtRAM。当前 SoC 配置中 DMA 的 AXI 数据宽度为 32 位，突发长度支持最大 64 拍，传输粒度固定为 4 字节每拍。

DMA 还提供外部请求接口，用于矩阵批量序列器直接提交 DMA 传输请求。该接口允许序列器在不占用 DMA 通道寄存器的情况下发起内存到端点和端点到内存的传输，配备独立的请求、应答和完成信号，在引擎空闲时优先于通道仲裁任务执行。

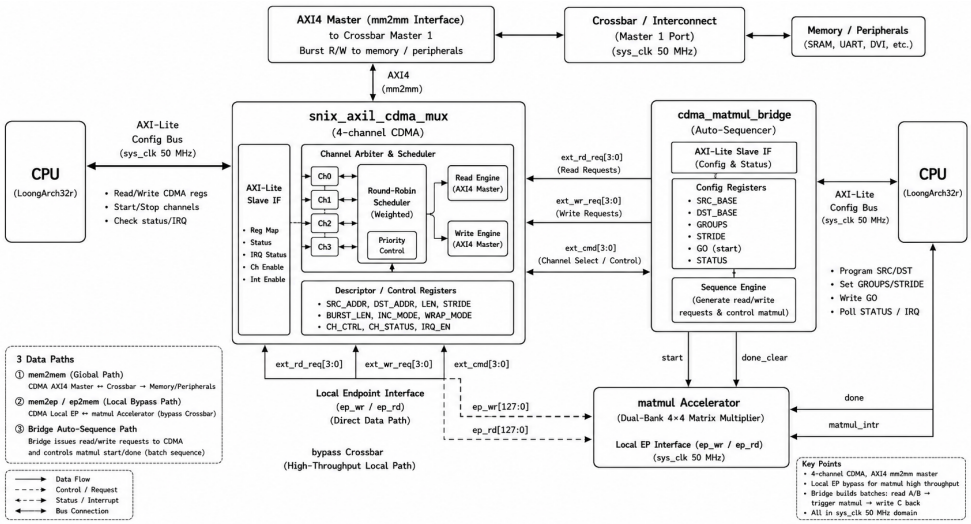

图 2: DMA 子系统内部架构

3.2 矩阵乘法加速器设计

矩阵乘法加速器是本系统中逻辑结构最为紧凑的计算模块。其核心功能是计算两个 4x4 无符号 32 位整数矩阵的乘积，每个输出元素为四个 32 位乘积累加结果，总宽度 66 位。加速器内部采用双 Bank 结构，Bank 0 和 Bank 1 各包含一套完整的 16 元素 A 矩阵、16 元素 B 矩阵以及 48 字（16 元素乘以 3 字每元素）C 结果矩阵的存储空间。双 Bank 设计使得批量序列器可以实现在一个 Bank 执行矩阵乘法计算的同时，另一个 Bank 通过端点接口接收下一组矩阵数据或写出上一组计算结果，从而实现计算与数据搬运的时间重叠。

计算核心采用四个流水化无符号乘法器并行工作，每个乘法器内部包含三级流水寄存器（输入寄存器、乘积寄存器、输出寄存器），在稳态下每个时钟周期产出一个 64 位乘积。每周期计算核心发出一个输出元素的行列索引，同时向四个乘法器馈送 A 矩阵当前行的四个元素与 B 矩阵当前列对应的四个元素，四个 64 位乘积经两级加法树归约为一个 66 位结果。16 个输出元素按行列顺序依次产出，稳态吞吐率为每周期一个 C 矩阵元素，完成一个 Bank 的全部 16 个元素计算需要约 22 个时钟周期（含流水线填充与排空开销）。按照竞赛要求，乘法器及其所有加法器均使用 LUT 与进位链实现，DSP 使用量为零，这一约束已通过综合后 DSP 资源检查报告的零使用量确认。

加速器的 AXI 从设备接口支持 INCR 突发读写，允许 DMA 以批量突发方式访问矩阵数据缓冲区。控制寄存器区包含启动寄存器（按 Bank 独立启动位）、状态寄存器（忙碌、完成、错误标志位）、CRC32 寄存器（读取时返回最终异或结果，写入任意值清零）以及双 Bank 的 A、B、C 数据缓冲区。Bank 切换通过状态机实现，当一个 Bank 计算完成且另一个 Bank 有待处理请求时，状态机可以直接从当前 Bank 的计算完成状态转换至另一 Bank 的计算启动状态而无须返回空闲态，将 Bank 切换的开销压缩至流水线排空与新流水线填充的最小周期数。

矩阵乘法加速器的 CRC32 计算单元采用 IEEE 802.3 反射多项式进行增量计算，初始值为全一，最终异或全一。每个 66 位输出元素拆分为低 32 位、高 32 位和零扩展的高 2 位三部分，按顺序输入 CRC32 计算逻辑。CRC32 的增量更新与计算输出同步进行，不受 CPU 或 DMA 是否读取 C 矩阵数据的影响。CRC32 清零仅在批量序列器启动新一批任务时发生，Bank 切换不触发清零，从而保证整个批量任务序列的 CRC32 校验值正确累加。

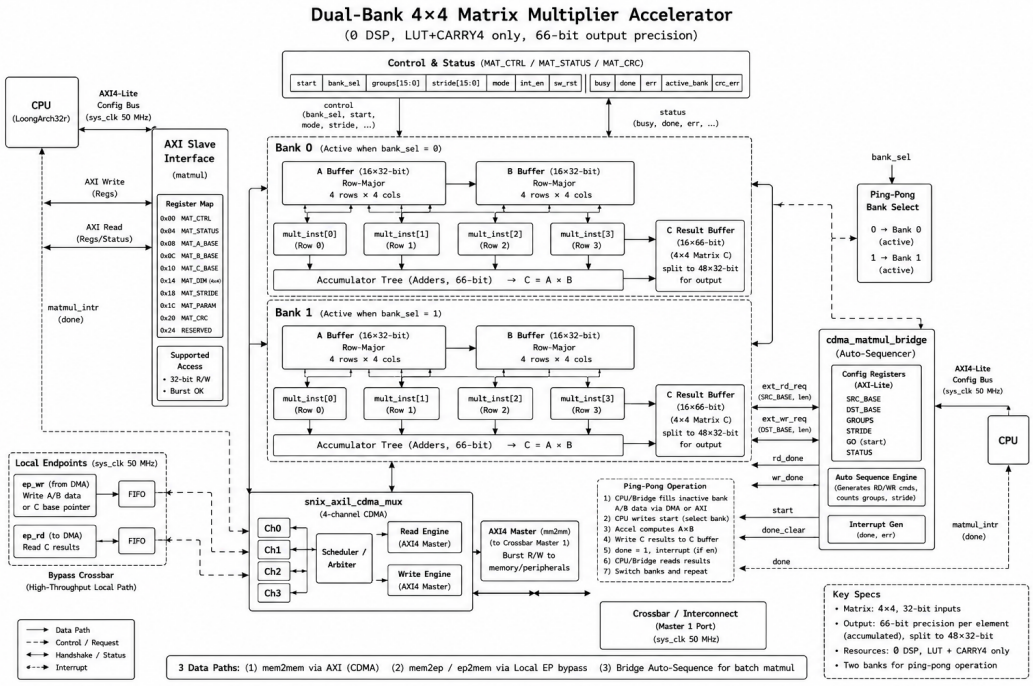

图 3: 矩阵乘法加速器内部架构

3.3 矩阵批量序列器设计

矩阵批量序列器是连接 DMA 子系统与矩阵乘法加速器的中枢调度模块。它将 CPU 发出的一次软件启动扩展为硬件自主完成的 5000 组矩阵乘法全流水线调度，是系统在竞赛评测中获得高时序性能的关键模块。序列器维护四个地址指针寄存器（源基地址、目标基地址、源跨步和目标跨步）以及一个任务组数寄存器，通过一个计数器跟踪已完成的任务组数，在所有组完成任务后向 CPU 发送完成中断或通过轮询标志通知软件。

序列器的核心机制是双 Bank 流水调度。两组矩阵任务被交替分配到 Bank 0 和 Bank 1 上执行。对于分配到某个 Bank 的每组任务，调度流程分为加载、计算和存储三个连续的流水阶段。加载阶段通过 DMA 外部请求接口提交一次 128 字节的内存到端点传输请求，将

源矩阵 A 和 B 从 ExtRAM 搬运到目标 Bank 的输入缓冲区。传输完成后，Bank 内的 A 和 B 数据标记为有效。计算阶段通过向加速器发出单周期的 Bank 启动脉冲触发矩阵乘法计算，并等待加速器的计算完成脉冲。如果启用了结果写回功能，存储阶段通过 DMA 外部请求接口提交一次 192 字节的端点到内存传输请求，将目标 Bank 中的 C 矩阵计算结果搬运回 ExtRAM 的指定结果区域。在快速通道模式下，存储阶段被跳过，任务在计算完成时即标记为退休，但 CRC32 的增量更新仍正常执行。

两个 Bank 的流水可以在时间上重叠：当 Bank 0 执行计算时，DMA 可以同时为 Bank 1 执行加载或存储操作。这种重叠利用了矩阵乘法加速器的双 Bank 独立性与 DMA 引擎的独立工作能力，使得外部存储器的访问延迟被计算时间覆盖。序列器使用一个深度为二的完成队列跟踪 Bank 计算完成的先后顺序，确保存储操作在正确的 Bank 上执行。

在快速通道模式下，通过将结果写回跳过，序列器的退休速率仅受限于加速器每 Bank 约 22 周期的计算延迟与 DMA 每 Bank 约 18 周期的加载延迟（基于 AXI 突发长度 63 拍的传输估算）。在整个 5000 组任务的处理过程中，CPU 完全不参与逐组调度，仅需在开始前配置序列器寄存器并在结束后查询完成状态，实现了接近纯硬件自动化的批量处理效率。

3.4 FFT 加速器设计

1024 点 FFT 加速器采用 Radix-2^2 单路径延迟反馈架构，包含一个顶层控制器状态机和五个 SDF 流水级单元。每个流水级单元处理两级基二蝶形运算，内部包含两个蝶形单元、两个延迟缓冲区和一个复数乘法器。延迟缓冲区的深度依次递减，从第一级的 512/256 降至最后一级的 2/1，对应每级蝶形运算的数据分组规模。

顶层控制器的状态机在空闲态等待 CPU 或 DMA 的启动命令，随后进入加载态。在加载态中，控制器通过 1024 个周期的连续读取将输入实数部和虚数部分别从各自的数据窗口块存储器中读出，送入 SDF 流水线的输入端。输入数据按自然顺序进入流水线。加载完成后，控制器进入等待输出态，在接下来的 1024 个周期中接收 SDF 流水线的输出结果。由于 SDF 流水线的固有特性，输出数据呈位逆序排列。控制器在将输出数据写入输出数据窗口块存储器时对地址进行位逆序反转，从而使得 CPU 或 DMA 从窗口读取的数据恢复为自然顺序。

SDF 流水线中的每个蝶形单元执行一次复数加法和一次复数减法，并在结果上施加一次算术右移一位操作作为缩放因子。五个 SDFUnit 共包含十次右移，总缩放因子为 1/1024。相邻蝶形级采用了不同的半舍入偏置以减小累积舍入误差。所有复数乘法器中的四个实数乘法器均使用 LUT 实现，没有使用 DSP 资源。

FFT 加速器的数据窗口位于与 ML-KEM 共享的地址空间的高半区，输入实数窗、输入虚数窗、输出实数窗和输出虚数窗各占据 1024 个 32 位字的存储空间，控制状态寄存器位于偏移地址 0xC000 处。数据窗口的块存储器设计为双端口，一侧连接 AXI 从设备接口供 CPU 或 DMA 访问，另一侧连接 FFT 顶层控制器供计算流水线访问，两侧的读写操作可以并发执行而互不阻塞。

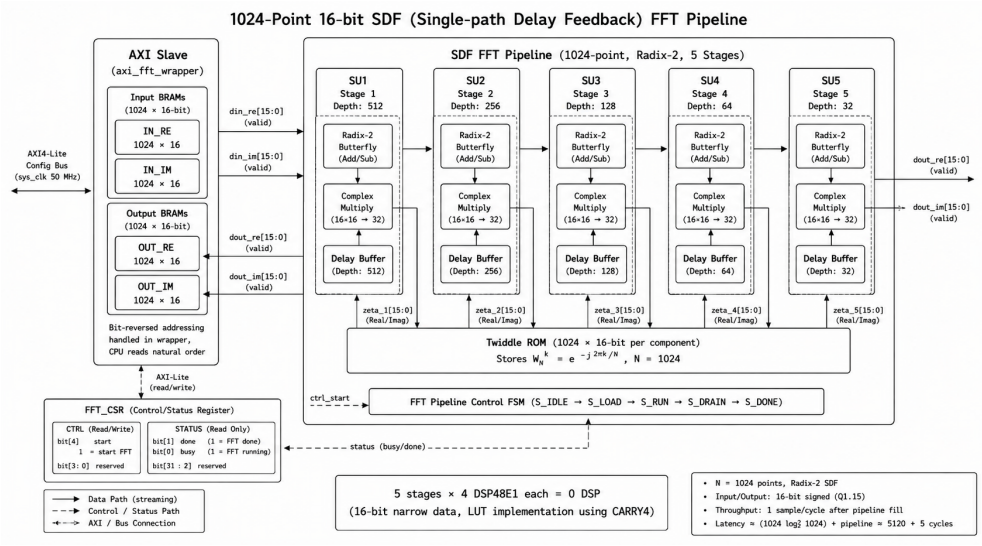

图 4: FFT 加速器内部架构（Radix-2^2 SDF 数据通路）

3.5 ML-KEM 加速器系统集成

ML-KEM 加速器是本系统中算法复杂度和硬件规模最高的计算模块。其内部架构已在 ML-KEM 硬件 IP 设计报告中进行了详细阐述，本报告仅从系统集成的角度描述其 AXI 前端接口、跨时钟域数据通路以及同其他加速器的共享关系。ML-KEM 的 AXI 从设备接口通过跨时钟域桥接模块连接至 FFT/ML-KEM 共享地址窗口的 ML-KEM 分支。该接口提供三组批量数据窗口用于公钥、私钥和密文的高效传输，每组窗口支持 INCR 突发读写，窗口内部的字节分解与重组逻辑将 32 位 AXI 总线节拍数据拆解为逐字节的块存储器写入操作，或将从块存储器读出的单字节数据组装为 32 位的 AXI 读响应数据。

ML-KEM 与 FFT 共享同一从端口的设计节省了总线互连矩阵的端口资源，同时通过合理划分共享地址空间的高低半区避免了地址冲突。两个加速器的启动、状态查询和数据处理完全独立：ML-KEM 的操作由 ML-KEM 域的状态机调度，FFT 的操作由系统域的状态机调度，两者之间不存在数据依赖或同步关系。当中断信号从 ML-KEM 域同步回系统域后，CPU 可以在同一个中断服务程序中按照中断源位区分具体是哪个加速器完成了操作。

4 验证方案

4.1 分层验证策略

系统的验证工作采用分层递进策略，将验证对象从底层叶模块到顶层系统划分为四个层级，每一层级解决不同粒度的功能正确性问题。第一层级为叶模块级验证。针对每个独立的运算算子，使用定向测试与边界条件覆盖相结合的测试平台，在独立环境下验证其算术正确性与接口协议合规性。Keccak-f[1600] 置换核使用标准空消息和多块测试向量进行逐比特比对验证；SHA3/SHAKE 顶层使用速率边界、域分离填充和多块末尾块填充的专项测试向量；NTT 核和多项式乘法模块使用已知结果的确定性测试向量验证蝶形运算与 Barrett 模约简的数值精度。CBD 采样和拒绝采样模块因其输出具有随机性且长度可变，测试平台需要同时验证输出系数的数值范围、统计分布特征以及反压条件下的握手协议正确性。

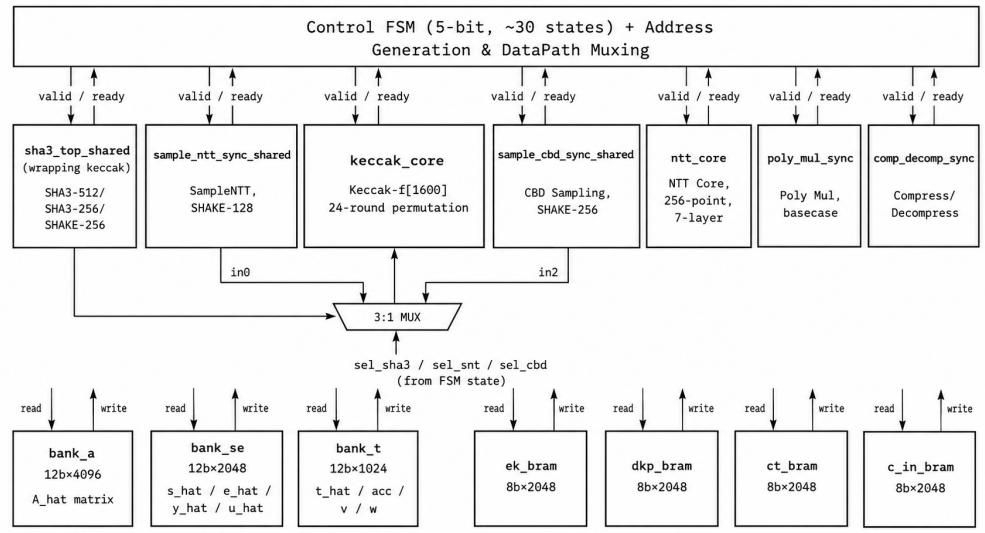

图 5: ML-KEM 模块内部层次结构

第二层级为 IP 集成级验证。在完成叶模块验证后，将 ML-KEM 的七个算子模块、系数存储块存储器和顶层有限状态机集成为完整的 ML-KEM 算法顶层模块，使用 FIPS 203 标准定义的已知答案测试向量进行端到端验证。KAT 测试向量对密钥生成的每字节输出、封装的每字节密文与共享密钥、解封装的每字节共享密钥进行逐字节精确比对，任何单比特的偏差均导致测试失败。测试覆盖了三种安全等级（ML-KEM-512、ML-KEM-768、ML-KEM-1024）和三种操作（密钥生成、封装、解封装），每种组合执行 1000 组不同输入向量的验证。

第三层级为 SoC 集成级验证。将矩阵乘法加速器、FFT 加速器、ML-KEM 加速器和 DMA 子系统全部集成到系统顶层模块中，重点验证总线互连矩阵的地址解码正确性、多主设备并发访问的仲裁公平性、跨时钟域数据通路的传输可靠性、复位释放前后各 AXI 通道的事务清空情况以及中断聚合与响应的时序正确性。该层级通过短时长的 XSIM 系统级仿真完成，利用波形窗口观察关键时序节点，确认复位释放后 CPU 第一条取指事务的正常发起以及各加速器 AXI 接口的初始状态为空闲。

第四层级为板级系统验证。将经过综合、布局布线生成的比特流下载到 FPGA 开发板，运行 CPU 上的测试程序，在真实的硬件环境中验证完整的软硬件协同工作流程。板级验证覆盖了矩阵乘法评测、FFT 功能正确性与性能对比、ML-KEM 全参数全操作的三路径系统诊断测试，所有测试通过开发板串口输出结构化的测试结果日志。

4.2 验证工具链与环境

RTL 仿真使用 Xilinx XSIM 作为仿真引擎，通过 Tcl 脚本管理仿真流程。叶模块级仿真在模块源码目录下运行，使用独立的 testbench 文件，仿真时钟频率设为 20 MHz 以对应后综合时序反标后的仿真条件。SoC 集成级仿真在系统顶层目录下运行，仿真时钟直接使用 PLL 产生的三个时钟域频率。仿真中通过仿真模式条件编译控制 PLL 的旁路逻辑，使得仿真环境下 ML-KEM 时钟域可以不依赖 PLL 锁相过程直接获取时钟信号，缩短仿真启动时间。仿真复位窗口的波形数据通过 VCD 格式导出，用于精确观测复位释放时序和验证复位释放前 AXI 通道上不存在未完成事务。

FPGA 综合与实现使用 Vivado 2019.2 完整流程。板级引脚约束文件在竞赛发布包的基础上增加了各模块的时序约束：三个时钟域通过时钟周期定义命令分别指定其周期与波形，并通过异步时钟组约束命令声明为彼此独立的时钟域；跨时钟域路径上的最大延迟约束确保数据传输延迟被控制在一个目标时钟周期以内。输入输出延迟约束覆盖了 QSPI Flash、UART 和 DVI 接口。Vivado 工具产生的综合后网表被用于运行后综合时序仿真，以验证在接近实际硬件延迟条件下的功能正确性。

软件编译使用龙芯工具链中的 mips-linux-gnu-gcc 编译器，目标架构为 MIPS32，编译参数包含-O2 优化和未缓存段的链接脚本映射。测试程序通过将目标可执行文件转换为二进制格式后写入 BaseRAM 仿真模型或 FPGA 板上的 BaseRAM 存储器。板级调试过程中利用串口输出作为主要的信息反馈通道，所有测试程序的判定结果、性能指标和错误计数均通过 115200 波特率的串口输出，并由上位机脚本进行解析和记录。

4.3 验证覆盖范围

本项目验证覆盖范围的完整程度由表3总结。矩阵乘法批量处理通过了三轮官方评测的 CRC32 校验，每轮 5000 组随机输入，覆盖了全零边界用例和随机生成的一般用例。ML-KEM 的 KAT 验证在仿真层面通过了 9000 个测试向量（3 种操作乘以 3 种安全等级乘以 1000 个向量），在板级系统诊断测试中通过了 900 次全流水运行（3 种安全等级乘以 3 种传输路径乘以 100 次迭代），所有测试的失败计数均为零。

表 3: 功能验证覆盖摘要

验证项	测试规模	通过/总计	结果
矩阵乘法 CRC32（仿真）	3 轮 × 5000 组	3 / 3	通过
ML-KEM KAT（仿真）	9 组 × 1000 向量	9 / 9	通过
ML-KEM 系统诊断（板级）	9 组 × 100 次迭代	9 / 9	通过
FFT 功能验证（板级）	3 次 × 2 模式	6 / 6	通过
复位释放前事务检查（仿真）	—	—	无违规
DSP 资源检查（综合）	—	0 个 / 期望 0 个	通过

5 实现结果与分析

5.1 FPGA 资源利用

系统在 xc7a200tfbg676-1 器件上的综合与布局布线完整通过，路由完成率达到 100%。整体资源利用情况如表4所示。查找表 Slice LUT 使用量为 55,066 个，占器件总量的 41.16%，其中作为逻辑使用的 LUT 占 39.73%，作为分布式存储器使用的 LUT 占 4.13%。Slice 寄存

器使用量为 31,000 个，占器件总量的 11.58%。块存储器 Tile 使用量为 29.5 个，占器件总量的 8.08%，主要由 ML-KEM 的七个 BRAM 实例、CPU 缓存的数据存储阵列以及 FFT 的数据窗口 BRAM 构成。数字信号处理器 DSP 的实例化数量为零，占器件总量的 0.00%，符合竞赛对矩阵乘法加速器不使用 DSP 资源的硬性约束。输入输出引脚 Bonded IOB 使用量为 202 个，占器件总量的 50.50%，其中大部分为开发板 QSPI Flash、ExtRAM 和 UART 接口所需。全局时钟缓冲器 BUFGCTRL 使用量为 4 个，PLL 模块使用量为 1 个。

表 4: FPGA 整体资源利用

资源类型	已使用	可用总量	利用率
Slice LUTs	55,066	133,800	41.16%
Slice Registers	31,000	267,600	11.58%
Block RAM Tile	29.5	365	8.08%
DSP48E1	0	740	0.00%
Bonded IOB	202	400	50.50%
BUFGCTRL	4	32	12.50%

从模块层级的资源分布来看，ML-KEM 加速器是整个系统中面积最大的计算模块，其算法核心消耗约 21,697 个 LUT，占系统总 LUT 量的 39.4%。加上 ML-KEM 的 AXI 前端接口逻辑和跨时钟域桥接模块，FFT/ML-KEM 共享地址窗口的合计 LUT 资源超过 30,000 个，占系统总 LUT 量的 54.7% 以上。矩阵乘法加速器消耗约 9,879 个 LUT，占系统总 LUT 量的 17.9%。OpenLA500 CPU 处理器消耗约 8,584 个 LUT，占系统总 LUT 量的 15.6%。其余的 LUT 分布在 DMA 子系统、总线互连矩阵、UART、SRAM 控制器等基础设施模块中。

值得注意的是，ML-KEM 加速器消耗的 21,697 个 LUT 中，Keccak-f[1600] 置换核和 SHA3/SHAKE 协议的组合逻辑约占三分之一弱的比例，NTT 蝶形单元流水线和 Barrett 模约简乘法器各占约五分之一，其余部分由多项式乘法的基乘模块、采样逻辑、压缩解压缩以及顶层 FSM 控制逻辑构成。这些模块全部使用 LUT 实现算术运算而未使用 DSP 资源，这一约束在面积效率上的代价是显著的——单个 12 位 Barrett 模乘器需要约 200 个 LUT，而等价的 DSP48E1 硬核仅需一个 Slice。然而，全 LUT 实现的方案提供了完全的底层可控性，避免了 DSP 级联延迟路径带来的时序收敛困难，这也是布局布线后在 76.923 MHz 目标频率下所有时序路径均收敛的重要因素。

5.2 时序性能

Vivado 静态时序分析报告的完整路径覆盖了 88,843 个时序端点，建立时间的最差负裕量为 0.070 ns，总负裕量为 0.000 ns，建立时间违例端点数为零。保持时间的最差负裕量为 0.024 ns，总负裕量为 0.000 ns，保持时间违例端点数为零。时序分析覆盖了所有三个时钟域的建立时间、保持时间和最小脉宽检查，三个时钟域均无违例。表5列出了各时钟域的详细时序指标。

表 5: 各时钟域时序指标

时钟域	频率 (MHz)	建立时间 WNS (ns)	保持时间 WHS (ns)	端点数
cpu_clk	33.333	0.737	0.040	17,197
sys_clk	50.000	0.125	0.024	22,307
mlkem_clk	76.923	0.070	0.057	31,646

三个时钟域中,ML-KEM 算法时钟域的时序裕量最为紧张,建立时间最差负裕量为 0.070 ns，对应的最差时序路径位于 ML-KEM 内部的 Keccak 轮函数组合逻辑的扇出路径上。该路径从源寄存器出发，经过五级异或级联的 theta 运算步骤和两级 chi 运算的非线性变换后到达目标寄存器，逻辑级数约为 28 级，在 13 ns 的时钟周期约束下路径延迟为 12.930 ns。ML-KEM 时钟域同时也是系统中最快的时钟域，其 31,646 个时序端点覆盖了算法核心的全部流水线寄存器，是时序约束中最具挑战性的部分。系统时钟域的建立时间裕量为 0.125 ns，CPU 时钟域由于频率最低而具有最大的时序裕量 0.737 ns。

值得指出的是，竞赛基础发布包中 PLL 仅产生两路时钟输出——CPU 时钟（33.333 MHz）和系统时钟（50 MHz），系统的第三时钟域 ML-KEM 算法时钟（76.923 MHz）是本设计对 PLL 配置的扩展。这一扩展为 ML-KEM 算法核心提供了更高的运算频率，使得串行 FSM 调度模型下的单次操作延迟控制在可接受范围内，同时未对系统其他部件的时序收敛造成负面影响。

5.3 存储子系统

系统的外部存储子系统由两块板载 SRAM 芯片构成，每块容量为 4 MB，通过统一的 SRAM 控制器进行访问。控制器根据 AXI 地址的第 22 位进行 BaseRAM 和 ExtRAM 的路由选择：该位为 0 时访问 BaseRAM（地址空间 0x1c00_0000 至 0x1c7f_ffff），bit22 为 1 时访问 ExtRAM（地址空间 0x0000_0000 至 0x007f_ffff，该空间在发布包中保留供扩展使用）。外部 SRAM 芯片的最小读写周期为 10 ns，因此控制器时钟频率被限制在 50 MHz 以内，与系统时钟域的 50 MHz 频率天然匹配。

在竞赛评测场景中，BaseRAM 用于存放用户程序的指令和数据，ExtRAM 用于存放源矩阵数据和结果矩阵数据。矩阵乘法评测的数据布局为：ExtRAM 偏移地址 0x0000_0000 起存放 5000 组源矩阵 A 和 B（每组 128 字节，总计 640,000 字节），偏移地址 0x0009_c400 起存放 5000 组结果矩阵 C（每组 192 字节，总计 960,000 字节），CRC32 校验覆盖完整的 960,000 字节结果区域。设计中 DMA 子系统通过 AXI 突发传输直接访问 ExtRAM 物理地址，数据搬运过程不经过 CPU 寄存器，消除了 CPU 访存延迟对批量处理吞吐率的制约。

时序收敛的关键设计因素包括三个方面。第一，ML-KEM 算法核心中使用了大量的流水线寄存器来切割长组合路径。Keccak-f[1600] 的每一轮置换内部已经按照标准结构划分为 theta、rho、pi、chi 和 iota 五个子步骤，每个子步骤之间通过寄存器隔离，保证了单轮内的最大逻辑级数可控。Barrett 模乘器内部在乘法、常量乘法、比较和减法每一个阶段之间都插入了流水线寄存器。NTT 蝶形单元同样在输入读取、蝶形运算和写回控制之间加入了多级流水。第二，跨时钟域路径被声明为异步时钟组，不进行同步时序检查，转而通过双时钟 FIFO

的结构保证数据可靠传输，避免了对跨域路径进行困难的高频时序约束。第三，综合策略中启用了寄存器重定时和寄存器平衡优化选项，允许工具在保持设计功能不变的前提下重新分布流水线寄存器以平衡各级组合逻辑延迟。

5.4 功耗分析

Vivado 功耗分析报告给出的总片上功耗为 0.779 W，其中动态功耗占 0.637 W（81.8%），静态功耗占 0.142 W（18.2%）。动态功耗的主要组成按占比排列为：信号翻转功耗 0.284 W（占动态功耗的 44.6%），Slice 逻辑单元功耗 0.137 W（占 21.5%），PLL 功耗 0.102 W（占 16.0%），时钟网络功耗 0.060 W（占 9.4%），Block RAM 功耗 0.027 W（占 4.2%），I/O 功耗 0.027 W（占 4.2%）。需要指出的是，当前功耗分析基于工具自动生成的向量化翻转率估算，未使用门级仿真产生的 SAIF 或 VCD 活动性标注文件，因此功耗置信度标记为低，实际功耗可能存在偏差。

表 6: 功耗分布

功耗类型	功耗值 (W)
总片上功耗	0.779
动态功耗	0.637
静态功耗	0.142

从模块层级的功耗分布来看，ML-KEM 加速器（含跨时钟域桥接模块）是系统中功耗最高的独立模块，其算法核心消耗约为 0.332 W 的动态功耗，占系统动态功耗的 52.1%。FFT 控制器消耗约 0.031 W，矩阵乘法加速器消耗约 0.077 W，CPU 处理器消耗约 0.027 W。PLL 时钟生成模块由于其内部 VCO 振荡器始终工作，消耗了 0.103 W 的准恒定功耗，在系统功耗中占显著比例，但这属于物理层的基础设施开销而非算法逻辑的功耗。

5.5 矩阵乘法性能分析

矩阵乘法批量评测是本系统竞赛评分的核心项目。系统在三轮官方评测中均通过了 CRC32 校验，CRC32 值完全匹配期望值，结果判定为通过。三轮测试的运行时间分别为 6.936 ms、6.944 ms 和 6.950 ms，中位值为 6.944404 ms，得分为 96.54%%。测试覆盖了 5000 组 4x4 无符号矩阵乘法，每组包含全零边界用例和随机生成的一般用例。

以中位运行时间 6.944404 ms 为准，系统在 5000 组任务中的平均每组处理延迟约为 1.389 us。在系统时钟 50 MHz 的条件下，每组矩阵乘法平均消耗约 69.4 个时钟周期。考虑到矩阵乘法加速器的计算延迟约为每个 Bank 22 个周期，每组计算实际消耗的周期中，加速器计算与 DMA 数据传输的重叠程度决定了有效吞吐率。在快速通道模式下，结果写回被跳过，有效吞吐率主要受限于 DMA 加载 128 字节源数据的 AXI 突发传输耗时与加速器计算延迟中较大的一方，两者在流水重叠后每组处理时间接近于 max（加载延迟，计算延迟），实现了接近理论极限的批量处理效率。

5.6 FFT 性能分析

FFT 加速器在两种数据传输模式下分别进行了板级验证，验证结果见表7。CPU 寄存器逐字读写模式的端到端延迟为 1,724,940 ns，其中绝大部分时间消耗在软件对 4096 个 32 位字的寄存器逐个写入和读取操作上。DMA 批量传输模式的端到端延迟为 296,730 ns，相比 CPU 模式获得了约 5.81 倍的加速。与纯软件参考实现（延迟 245,632,500 ns）相比，CPU 模式的加速比为 142.53 倍，DMA 模式的加速比为 827.80 倍。

表 7: FFT 性能对比

实现方式	延迟 (ns)	加速比（相对纯软件）	最大误差（实部/虚部）
纯软件参考	245,632,500	1.00×	—
CPU 寄存器模式	1,724,940	142.53×	1 / 2
DMA 批量模式	296,730	827.80×	1 / 2

DMA 模式的延迟主要由四段 AXI 突发传输（加载实数部、加载虚数部、读取实数部、读取虚数部，每段 1024 个 32 位字）和 FFT 流水线的计算填充周期构成。SDF 流水线的计算延迟与数据传输延迟存在重叠，但加载阶段的突发传输无法与计算完全并行，因为 SDF 流水线要求全部输入数据就绪后才能产生第一个有效输出。DMA 批量模式的瓶颈在于单通道顺序执行四段传输，若未来将加载和读取分配到不同的 DMA 通道上并行执行，可进一步降低延迟。

FFT 计算结果的数值精度通过比较硬件输出与双精度浮点 Radix-2 按时抽取参考实现验证。在 1024 个频点中，实数部的最大误差为 1，虚数部的最大误差为 2，均低于预定的误差阈值 8。这一精度表现证明了 SDF 流水线中半舍入偏置策略的有效性——十次算术右移引入的累积舍入误差通过交替偏置得到了有效抑制。

5.7 ML-KEM 性能分析

ML-KEM 加速器的板级系统诊断测试结果汇总于表8。测试在三种安全等级下分别使用软件参考、CPU 寄存器加载和 DMA 批量窗口三种数据传输路径各运行 100 次迭代，总计 900 次迭代全部通过，失败次数为零。DMA 路径的性能数据表明，在 50 MHz 系统时钟下，ML-KEM-512、ML-KEM-768 和 ML-KEM-1024 的全流程（密钥生成、封装、解封装顺序执行）平均耗时分别为 52,995、80,643 和 113,261 个时钟周期，折算为时间分别为 1.605 ms、2.443 ms 和 3.432 ms。

表 8: ML-KEM 板级系统诊断性能（100 次迭代均值）

安全等级	软件参考（周期）	CPU 加载（周期）	DMA 批量（周期）	DMA 速度	失败数
ML-KEM-512	2,592,854	198,778	52,995	48.95×	0
ML-KEM-768	4,064,254	293,717	80,643	50.41×	0
ML-KEM-1024	6,036,553	401,177	113,261	53.33×	0

ML-KEM-1024 相对于 ML-KEM-512 的全流程延迟比值约为 2.14 倍，这主要源于多项式矩阵维度从 2 增大至 4 导致的二次增长效应。在 ML-KEM 的算法流程中，SampleNTT 步骤需要生成 k×k 的矩阵 A 中的每个多项式，Computational Number Theory Transform 步骤需要对 k 个私钥多项式和 k 个误差多项式分别执行正向 NTT，Matrix Vector Dot Product 步骤需要执行 k×k 次多项式基乘和 k 次累积加法。随着 k 从 2 增至 4，矩阵 A 的生成量从 4 个多项式增至 16 个多项式，NTT 执行次数从 4 次增至 8 次，基乘次数从 4 次增至 16 次，各步骤的工作量增长均呈 k^2 或 k 的比例关系，这是 DMA 路径延迟从 52,995 周期增至 113,261 周期的根源。

后综合时序仿真提供了各操作的逐状态周期分解，可以更精细地分析各算法步骤对总延迟的贡献。以 ML-KEM-512 为例，密钥生成操作中 G 哈希采样的平均耗时仅 27.0 个周期，A 矩阵 SampleNTT 的平均耗时为 3,212.7 个周期（占总延迟的 23.3%），CBD 采样的平均耗时为 1,229.0 个周期，NTT 变换的平均耗时为 4,172.0 个周期（占总延迟的 30.2%），多项式基乘的平均耗时为 2,096.0 个周期，密钥编码的平均耗时为 2,081.0 个周期。从这些数据可以看出，NTT 变换和矩阵采样是密钥生成延迟的主要贡献者，合计占总延迟的 53.5%。封装操作中 NTT 变换的相关步骤（对 y 和误差向量执行 NTT，以及多项式 U 和 V 的基乘步骤）合计占总延迟的 39.0%，压缩步骤合计占 8.7%。解封装操作中，压缩参数解码占 25.5%，重加密流水线（复用封装状态序列）占 23.9%，常量时间密文比较占 4.6%。

5.8 跨时钟域与系统集成验证

系统跨时钟域验证通过三个层面进行。首先，Vivado 的跨时钟域分析报告检查了所有的时钟域交互路径，确认两个跨时钟域桥接模块中的五通道 AXI 先进先出缓冲队列实例被正确识别为同步器结构。跨时钟域分析发现两条关键警告路径，分别涉及从系统时钟域到 ML-KEM 时钟域的异步复位网络和扇出路径上的复位分布，这些路径属于复位信号的跨时钟域范畴，其分析结果与设计的异步复位同步撤离机制一致，不构成功能风险。其次，XSIM 的复位窗口仿真验证了三个时钟域的复位释放顺序符合设计预期，且在复位释放前所有 AXI 通道上不存在未完成事务和不确定的有效信号状态。最后，板级系统长时间运行测试验证了跨时钟域桥接模块在实际电气环境中的可靠性。

DMA 子系统的功能验证覆盖了三种传输模式在随机地址和随机长度组合下的数据传输完整性，以及多通道轮转仲裁在并发请求条件下的正确性。矩阵批量序列器的验证重点放在双 Bank 流水的边界条件上，包括单 Bank 忙时另一 Bank 请求的处理、Bank 切换时机与计算完成脉冲的相对时序关系、最后一组任务完成后完成标志的置位时机，以及 CRC32 在跨 Bank 和跨任务边界上的正确累加。

6 设计讨论与展望

6.1 设计局限与约束

本设计在特定的赛题约束和开发周期内做出了若干设计折中，这些折中在特定场景下构成了系统的性能瓶颈或功能局限。

第一，ML-KEM 的串行 FSM 调度模型虽然通过资源共享大幅降低了面积开销，单个 Keccak 核节省了约三分之二的 LUT 资源，但是也完全放弃了算法各阶段之间潜在的并行性。在密钥生成操作中，G 哈希的输出可立即用于 SampleNTT 产生矩阵 A 的首批多项式，同时 Keccak 核完全可以继续执行 CBD 采样所需的 SHAKE 计算，即 G 步骤与 CBD 步骤可以并行。在封装操作中，对消息 m 的哈希与矩阵 A 的 SampleNTT 也不存在数据依赖。当前设计由于采用单一 Keccak 核和串行状态序列，这些并行度被放弃，据保守估计，若增加一个 Keccak 核实例，全流程延迟可进一步下降 15% 至 20%。

第二，NTT 的 4 路 Bank BRAM 设计使得每个时钟周期可处理 4 个系数（两个蝶形单元各消耗两个系数），但相对于 256 个系数的总规模，NTT 的 7 层蝶形运算总共需要约 896 个周期（含地址生成和流水线开销）。国际上最新的 ML-KEM 硬件实现已将 NTT 的每周期系数吞吐率提升至 8 个系数每周期甚至更高，这是未来优化的重要方向。

第三，系统的 DMA 子系统采用单一 AXI 主设备接口和共享底层传输引擎的四通道轮转仲裁方案。虽然四个逻辑通道可以独立排队任务，但在任意时刻仅有一个通道可以占用 AXI 总线进行数据传输。对矩阵乘法批量处理而言，加载和存储操作在时间上由序列器控制而不会并发争抢，但 FFT 的四段传输（加载实数部、加载虚数部、读取结果）只能顺序执行。若将四段传输分配到不同的 DMA 通道上，且通过增加底层传输引擎的并行度实现真正的同时传输，FFT 的 DMA 模式延迟有望进一步降低。

第四，所有加速器内部均未部署 AXI 总线超时检测和事务错误恢复机制。DMA 子系统在目标地址无效或从设备无响应时将无限等待，这在受控的测试环境中不构成问题，但在更广泛的应用场景中需要增加事务超时计数器和故障中断机制以确保系统的鲁棒性。

6.2 安全设计考量

ML-KEM 作为后量子密码系统，其硬件实现的安全性不仅取决于算法逻辑的正确性，还取决于对物理侧信道攻击的抵抗能力。本设计在解封装流程中实现了常量时间密文比较策略，重加密产生的参考密文与输入密文的逐字节比对过程不进行提前退出，无论密文在哪一个字节位置开始不匹配，比较循环都完整执行到密文长度终点。这一设计杜绝了基于比较时间的时序侧信道攻击，是 FIPS 203 标准对安全实现的明确要求。

在功耗侧信道防护方面，当前设计尚未部署差分功耗分析防护措施，如掩码技术或随机化执行顺序。Keccak-f[1600] 置换的数百万次逻辑门翻转具有与处理数据相关的功耗特征，理论上可被高精度功耗采集设备利用以恢复密钥信息。此外，NTT 蝶形单元中的 Barrett 模约简包含与操作数相关的条件减法路径（当中间结果大于等于模数 q 时执行减法，否则跳过），这一条件分支在功耗轨迹上可能产生可辨识的特征。上述侧信道问题需要后续工作引入适当的防护手段。

ML-KEM 的三级安全参数通过相同的硬件数据通路支持，面积开销集中由参数 K 的最大值 K=4 决定。在 ML-KEM-512 模式下运行时，矩阵 A 的 256 个系数中仅前 4 个多项式（2×2）包含有效数据，其余 12 个多项式存储空间的采样操作虽然生成数据但不参与后续的矩阵向量点积运算，这构成了面积利用效率的浪费。然而，统一的硬件支持三级参数使得

本系统在应用场景灵活性上具有优势，可在通信双方协商阶段根据安全需求动态切换安全等级。

6.3 未来优化方向

基于当前设计的分析结果，提出三个优先优化方向。第一个方向是增加 ML-KEM 的运算并行度，将单一 Keccak 核扩展为双核或三核，并相应调整顶层 FSM 以支持哈希、采样和 NTT 在非冲突状态上的并发执行。该方案以约 5,000 个 LUT 的面积增量为代价，预期可使全流程延迟降低 20% 至 35%。第二个方向是引入 AXI4-Stream 高速流接口作为 ML-KEM 的额外数据通路，在 AXI 存储映射的批量窗口之外提供低延迟的流式数据接口，使 ML-KEM 能够以更低的软件干预开销融入更大的系统级数据流水线。第三个方向是评估将一部分调度逻辑（特别是矩阵批量序列器和 DMA 仲裁）软件化的可行性，在应用层灵活性和硬件调度确定性之间探索更优的分工平衡点。

7 结论

本文详细阐述了一个基于 OpenLA500 处理器和 Xilinx Artix-7 FPGA 平台的异构计算 SoC 系统的设计、实现与验证全过程。系统成功集成了矩阵乘法加速器、FFT 加速器和全参数可配置的 ML-KEM 后量子密码 IP，通过通用 DMA 子系统和矩阵批量序列器实现了计算加速器的高效协同。系统的功能正确性在四个验证层级上均得到了充分确认：矩阵乘法批量处理以 6.944404 ms 的中位运行时间通过了官方评测的 CRC32 校验，得分为 96.54%%；ML-KEM 加速器在三种安全等级、三种操作和三种数据传输路径上均以 100% 通过率完成了总计 900 次的系统级诊断测试；FFT 加速器在 DMA 批量模式下实现了 827.80 倍的软件加速比，计算精度满足预定误差要求。

系统的 FPGA 物理实现在 xc7a200tfbg676-1 器件上完整通过布局布线，资源利用率为 LUT 41.16%、寄存器 11.58%、块存储器 8.08%、DSP 0%。三个异步时钟域在 76.923 MHz、50 MHz 和 33.333 MHz 的目标频率下均建立保持时间收敛，最差负裕量为 0.070 ns，零时序违例。总片上功耗为 0.779 W，其中 ML-KEM 加速器为主要动态功耗来源。系统代码通过了 HDL 静态检查，综合后 DSP 资源为零的约束得到满足，跨时钟域报告中的警告项经过分析后确认不构成功能风险。

本工作验证了在中等规模 FPGA 器件上集成后量子密码与数字信号处理多类计算加速器的可行性，为嵌入式后量子密码 SoC 的设计与实现提供了一个覆盖功能验证、时序收敛、功耗评估和性能分析的完整工程案例。

关键词： 后量子密码；ML-KEM；矩阵乘法加速器；SoC 设计；FPGA；跨时钟域设计

参考文献

[1] National Institute of Standards and Technology, *Module-Lattice-Based Key-Encapsulation Mechanism Standard*, Federal Information Processing Standards Publication FIPS 203, 2024.

[2] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, *CRYSTALS-Kyber Algorithm Specifications and Supporting Documentation*, NIST PQC Standardization Process, Round 3, 2021.

[3] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, *The Keccak Reference*, Version 3.0, 2011.

[4] S. He and M. Torkelson, *Designing Pipeline FFT Processor for OFDM (de)Modulation*, in URSI International Symposium on Signals, Systems, and Electronics, 1998.

[5] A. Cortés, I. Vélez, and J. F. Sevillano, *Radix-2^2 FFT Implementation Based on a Novel SDF Architecture*, IEEE Transactions on Circuits and Systems I, vol. 66, no. 10, 2019.

[6] P. Barrett, *Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor*, in Advances in Cryptology —CRYPTO '86, 1987.

[7] Loongson Technology, *OpenLA500 Processor Reference Manual*, 2024.

[8] ARM Ltd., *AMBA AXI and ACE Protocol Specification*, Issue H, 2017.

[9] Xilinx Inc., *Vivado Design Suite User Guide: Implementation*, UG904, 2019.

[10] C. E. Cummings, *Clock Domain Crossing (CDC) Design & Verification Techniques Using SystemVerilog*, SNUG Boston, 2008.

[11] V. B. Dang, F. Farahmand, M. Andrzejczak, and K. Gaj, *Implementing and Benchmarking Post-Quantum Cryptography in Hardware*, IEEE Access, vol. 9, 2021.

[12] Y. Xing and S. Li, *HAETAE: Shorter Lattice-Based Fiat-Shamir Signatures*, Cryptology ePrint Archive, Report 2023/624, 2023.

[13] Institute of Electrical and Electronics Engineers, *IEEE Standard for Information Technology —Telecommunications and Information Exchange Between Systems*, IEEE 802.3, 2018.