

ML-KEM 后量子密钥封装硬件 IP 核 设计与验证

碳基生物

团队编号: CICC1008439

龙芯中科杯·2026 全国大学生集成电路创新创业大赛

2026 年 7 月 8 日

目录

1 绪论

1.1 设计背景

ML-KEM (Module-Lattice-based Key-Encapsulation Mechanism, FIPS 203) 是 NIST 于 2024 年 8 月发布的后量子密码标准, 用于替代 RSA 和 ECC 在密钥交换中的角色。其安全性建立在模格上的带误差学习问题 (MLWE) 之上, 多项式环为 $\mathbb{Z}_q[x]/(x^{256} + 1)$, 素数模 $q = 3329$, 多项式次数 $n = 256$ 。

软件实现的 ML-KEM 在嵌入式处理器上存在显著性能瓶颈——以 ML-KEM-768 为例, 一轮完整的密钥生成、封装和解封装在通用 CPU 上需要数百万个时钟周期。本文设计的硬件 IP 核 `mlkem_top` 旨在通过专用数据通路和计算资源时分复用, 在 FPGA 平台上实现高性能、低面积的 ML-KEM 加速。

1.2 关键设计特性

本 IP 核在单一硬件模块内具有以下特性:

- (1) **运行时三等级支持**: 通过 $k_i \in \{2, 3, 4\}$ 端口在运行时选择 ML-KEM-512/768/1024 安全等级, 存储资源按 $K_{MAX} = 4$ 静态分配, 无需重综合。
- (2) **全操作覆盖**: 支持 KeyGen (FIPS 203 算法 16)、Encaps (算法 17) 和 Decaps (算法 18, 含 FO 变换隐式拒绝)。
- (3) **Keccak 核跨阶段复用**: 单一 `keccak_core` 实例在 G/H/J 哈希、SampleNTT 采样和 CBD 采样三个消费者之间时分复用, 是面积优化的最大贡献因素。
- (4) **Decaps 重加密路径复用**: D6 步的 K-PKE 重加密直接复用整条 Encaps 流水线的全部 FSM 状态, 避免重复实现加密数据通路。
- (5) **逐字节流式 I/O**: 公钥 ek 、私钥 dk 和密文 c 通过 BRAM 窗口逐字节装载与回读, 免去超宽顶层总线。
- (6) **NIST KAT 完全验证**: 三种等级、三种操作全部逐字节比对 NIST FIPS 203 KAT 标准答案通过。

1.3 代码规模

RTL 代码总计约 5,517 行 (不含测试平台), 其中 `mlkem_top.v` (顶层 FSM 集成 + 数据通路 + 三库存储管理) 约 2,264 行, 各叶子算子模块合计约 3,253 行。全部采用 SystemVerilog 编写, 统一使用 `valid/ready` 流式握手接口。

2 IP 模块规格

2.1 算法参数

`mlkem_top` 的运行时参数由输入端口 k_i 和 op_i 在 `start_i` 脉冲时刻锁存至内部寄存器 k_r 和 op_r ，后续所有数据通路宽度、存储子区间基地址和循环次数均由 k_r 推导。表??列出三种安全等级的密钥和密文尺寸。

表 1: 运行时安全等级参数

k_i	方案	K	η_1	(d_u, d_v)	ek(B)	dk(B)	ct(B)
2	ML-KEM-512	2	3	(10,4)	800	1632	768
3	ML-KEM-768	3	2	(10,4)	1184	2400	1088
4	ML-KEM-1024	4	2	(11,5)	1568	3168	1568

固定参数：模数 $q = 3329$ 、多项式次数 $n = 256$ 、 $\eta_2 = 2$ （所有等级）、共享密钥 32 字节。

2.2 顶层端口信号

`mlkem_top` 的端口按功能分为四组。表??列出主要信号。

表 2: `mlkem_top` 主要端口

信号	位宽	功能
<code>clk, rst_n</code>	1, 1	时钟 (25 MHz) 与低电平异步复位
<code>k_i, op_i</code>	3, 2	安全等级 (2/3/4) 与操作码 (0=KG, 1=Enc, 2=Dec)
<code>d_i, z_i, msg_i</code>	256×3	KeyGen 种子 d 、隐式拒绝种子 z 、Encaps 消息 m
<code>start_i</code>	1	启动脉冲，锁存 k_i 和 op_i
<code>busy_o, done_o</code>	1, 1	运行标志与完成脉冲
<code>ss_o</code>	256	共享密钥输出 (<code>done_o</code> 时有效)
<code>ek_in_we/addr/byte</code>	1,11,8	Encaps 公钥流式预装载
<code>dk_in_we/addr/byte</code>	1,12,8	Decaps 私钥流式预装载 (自动路由至 <code>dkp_bram/ek_bram/寄存器</code>)
<code>c_in_we/addr/byte</code>	1,11,8	Decaps 密文流式预装载
<code>dbg_*</code>	多种	调试抽头：系数、ek/dk/ct 字节、中间哈希值

握手协议分三阶段：ST_IDLE 状态下通过流式端口逐字节预装载数据；`start_i` 脉冲启动并锁存参数；`busy_o` 拉高期间执行计算；`done_o` 脉冲时 `ss_o` 和 BRAM 产物有效。所有 256-bit 端口采用字节 0 在低位的约定。

Decaps 私钥预装载路径中，`dk_in_*` 端口根据写入地址自动将字节流路由至不同目的地：地址 $[0, 384K)$ 写入 `dkp_bram` ($\hat{s}$ 编码)， $[384K, 768K + 32)$ 写入 `ek_bram` ($\hat{t}$ 编码 $\parallel \rho$)， $[768K + 32, 768K + 64)$ 和 $[768K + 64, 768K + 96)$ 分别写入 $H(ek)$ 和 z 的寄存器。

3 总体架构设计

3.1 设计原则

`mlkem_top` 的架构遵循三条核心原则。

原则一：存储按最大等级分配，运行时参数递推。 多项式系数存储库以 $K_{MAX} = 4$ (可容纳 28 个多项式, 每个 256 系数) 静态配置深度, k_r 仅影响读写地址生成逻辑中的基地址和循环边界。字节存储库 (`ek_bram`, `dkp_bram`, `ct_bram`) 同样按最大密钥尺寸 (1568/1536/1568 字节) 分配。运行时地址计算通过少量组合逻辑完成，无动态内存分配开销。

原则二：Keccak 核全阶段时分复用。 Keccak-f[1600] 置换是设计中面积最大的单一算子 (OOC 综合约 5,300 LUT)。G/H/J 哈希族、SampleNTT (SHAKE-128) 和 CBD (SHAKE-256) 三个消费者处于互斥的 FSM 阶段，设计将所有 Keccak 请求复用至单一 `keccak_core` 实例。输入侧通过 `sel_sha3/sel_snt/sel_cbd` 多路选择器切换，输出侧将置换结果广播至全体消费者，仅当前激活阶段采样有效信号。

原则三：Decaps 重加密完全复用 Encaps 流水线。 FIPS 203 算法 18 的 D6 步需要以恢复的消息 m' 和随机种子 r' 重新执行一次 K-PKE 加密以产生参考密文 c' 。这一流程与算法 17 的 Encaps 操作的数据通路完全同构。设计将 m' 和 r' 分别写入原有的 m_r 和 r_r 寄存器，`ek_bram` 中保持原始公钥 ek 不变，随后将 FSM 直接转移至 Encaps 流水线的首个状态 `ENC_LOAD`，整条 15 状态的 Encaps 数据通路被完整复用。

3.2 模块层次结构

图??给出 `mlkem_top` 的内部层次结构。设计分为三个逻辑层：控制调度层 (FSM + 地址生成 + 数据通路复用)、算子计算层 (7 个已验证的叶子模块) 和存储层 (7 个 BRAM 实例)。

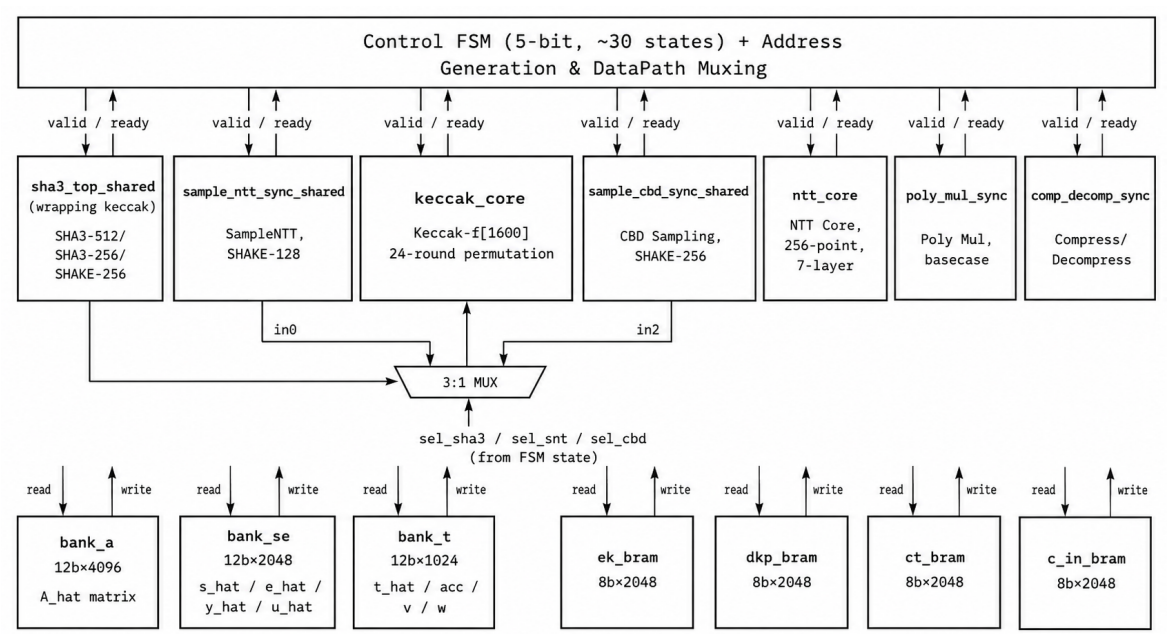


图 1: mlkem_top 内部三层模块结构

3.3 顶层 FSM 调度

全部运算由 5-bit 状态机集中调度，从 `ST_IDLE` 出发按 `op_r` 分支为三条数据通路。图??给出状态转移图。

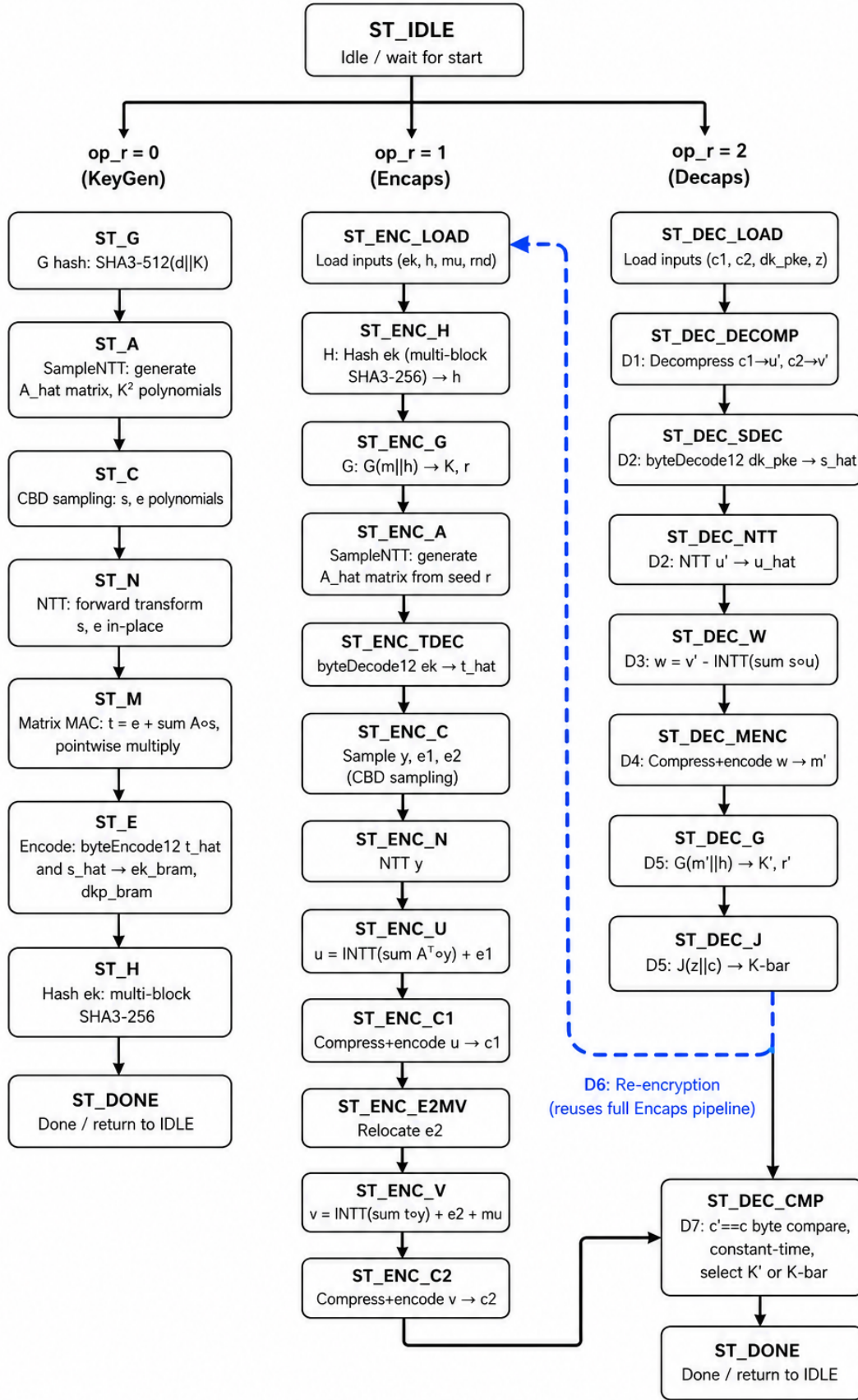


图 2: mlkem_top 顶层 FSM 状态转移图

KeyGen 路径对应 FIPS 203 算法 16, 共 7 个主状态 (不含 ST_IDLE 和 ST_DONE)。各状态间的数据传递通过多项式存储库完成: 前序状态将结果写入 BRAM, 后续状态从 BRAM 读取。

Algorithm 1 KeyGen 数据通路 (FIPS 203 算法 16)

- 1: **ST_G**: $G(d\|K) \rightarrow (\rho, \sigma)$, 单块 SHA3-512, 约 1,800 周期
 - 2: **ST_A**: 对全部 K^2 个 (j, i) 索引执行 $\text{SampleNTT}(\rho\|j\|i) \rightarrow \hat{A}[i][j]$, 逐系数写入 **bank_a**
 - 3: **ST_C**: $\text{CBD}_{\eta_1}(\sigma, i) \rightarrow s[i]$, $\text{CBD}_{\eta_1}(\sigma, K+i) \rightarrow e[i]$, 写入 **bank_se**
 - 4: **ST_N**: $\text{NTT}(s[i]) \rightarrow \hat{s}[i]$, $\text{NTT}(e[i]) \rightarrow \hat{e}[i]$, 原位更新 **bank_se**
 - 5: **ST_M**: $\hat{t}[i] = \hat{e}[i] + \sum_{j=0}^{K-1} \hat{A}[i][j] \circ \hat{s}[j]$, 逐点乘通过 **poly_mul_sync**, 累加写入 **bank_t**
 - 6: **ST_E**: $\text{byteEncode}_{12}(\hat{t})\|\rho \rightarrow ek$, $\text{byteEncode}_{12}(\hat{s}) \rightarrow dk_{pke}$, 写入 **ek_bram** 和 **dkp_bram**
 - 7: **ST_H**: $\text{SHA3-256}(ek) \rightarrow H(ek)$, 多块吸收 (块数 = $\lceil (ek_bytes + 1)/136 \rceil$)
-

Encaps 路径对应 FIPS 203 算法 17, 共 12 个主状态。

Algorithm 2 Encaps 数据通路 (FIPS 203 算法 17)

- 1: **ST_ENC_H**: $\text{SHA3-256}(ek) \rightarrow H(ek)$
 - 2: **ST_ENC_G**: $G(m\|H(ek)) \rightarrow (K, r)$, 单块 SHA3-512
 - 3: **ST_ENC_A**: 重新生成 $\hat{A}^T$ 矩阵 (SampleNTT, 行列索引交换), 写入 **bank_a**
 - 4: **ST_ENC_TDEC**: $\text{byteDecode}_{12}(ek) \rightarrow \hat{t}[i]$, 逐系数解码至 **bank_a**
 - 5: **ST_ENC_C**: $\text{CBD}_{\eta_1}(r, 0) \rightarrow y$, $\text{CBD}_{\eta_2}(r, K) \rightarrow e_1$, $\text{CBD}_{\eta_2}(r, 2K) \rightarrow e_2$, 写入 **bank_se** 和 **bank_t**
 - 6: **ST_ENC_N**: $\text{NTT}(y) \rightarrow \hat{y}$, 原位更新 **bank_se**
 - 7: **ST_ENC_U**: $\hat{A}^T \circ \hat{y}$ 逐点乘后 INTT, 再加 e_1 得 u , 写入 **bank_se**
 - 8: **ST_ENC_C1**: $\text{Compress}_{d_u}(u)$ 后 $\text{byteEncode}_{d_u} \rightarrow c_1$ 字节, 写入 **ct_bram**
 - 9: **ST_ENC_E2MV**: 将 e_2 从 **bank_t** 搬迁至 **bank_a**, 为 V 步累加准备
 - 10: **ST_ENC_V**: $\hat{t} \circ \hat{y}$ 逐点乘后 INTT, 加 e_2 和 $\text{Decompress}_1(m)$ 得 v , 写入 **bank_t**
 - 11: **ST_ENC_C2**: $\text{Compress}_{d_v}(v)$ 后 $\text{byteEncode}_{d_v} \rightarrow c_2$ 字节, 拼接至 **ct_bram**
 - 12: **输出**: $ss = K$
-

Decaps 路径对应 FIPS 203 算法 18, 前 5 个主状态完成 D1–D5 步解密恢复, 随后复用整条 Encaps 流水线 (D6 重加密), 最后以一个比较状态 (D7) 完成隐式拒绝密钥选择。这是硬件设计中最复杂的控制流, 也是 Decaps 流水线复用创新的集中体现。

Algorithm 3 Decaps 数据通路 (FIPS 203 算法 18, 含流水线复用)

- 1: **ST_DEC_DECOMP** (D1) : $\text{byteDecode}_{d_u}(c_1) \rightarrow \text{Decompress}_{d_u} \rightarrow u'$, $\text{byteDecode}_{d_v}(c_2) \rightarrow \text{Decompress}_{d_v} \rightarrow v'$, 写入 **bank_se** 和 **bank_a**
 - 2: **ST_DEC_SDEC** (D2): $\text{byteDecode}_{12}(dk_{pke}) \rightarrow \hat{s}[i]$, 逐系数解码至 **bank_a**
 - 3: **ST_DEC_NTT** (D2 续): $\text{NTT}(u'[i]) \rightarrow \hat{u}[i]$, 原位更新 **bank_se**
 - 4: **ST_DEC_W** (D3): $\hat{s}[j] \circ \hat{u}[j]$ 逐点乘后 INTT, v' 减去累加和得 w , 写入 **bank_t**
 - 5: **ST_DEC_MENC** (D4): $\text{Compress}_1(w)$ 后 $\text{byteEncode}_1 \rightarrow m'$
 - 6: **ST_DEC_G** (D5): $G(m' \| H(ek)) \rightarrow (K', r')$
 - 7: **ST_DEC_J** (D5 续): $J(z \| c) \rightarrow \bar{K}$ (隐式拒绝备用密钥)
 - 8: **流水线复用** (D6): 将 m' 写入 m_r 、 r' 写入 r_r , 跳转至 **ST_ENC_LOAD**, 完整复用 Encaps 路径全部 12 个状态, 产生参考密文 c' (存入 **ct_bram**)
 - 9: **ST_DEC_CMP** (D7): 逐字节比较 c' (**ct_bram**) 与原始 c (**c_in_bram**), 常量时间无早退; $c' = c$ 时输出 K' , 否则输出 $\bar{K}$
-

4 各功能模块数据通路与控制逻辑

4.1 Keccak-f[1600] 置换核与哈希族

keccak_core 模块实现 24 轮连续 Keccak-f[1600] 置换, 内部结构为简单的迭代流水线: 1600-bit 状态寄存器 $\rightarrow$ 组合 **keccak_round** $\rightarrow$ 写回状态寄存器。keccak_round 为纯组合逻辑, 实现 Keccak 的 $\theta/\rho/\pi/\chi/\iota$ 五步映射, 其中 θ 步的 5 路 XOR 归约和 χ 步的非线性变换是面积消耗的主要来源。

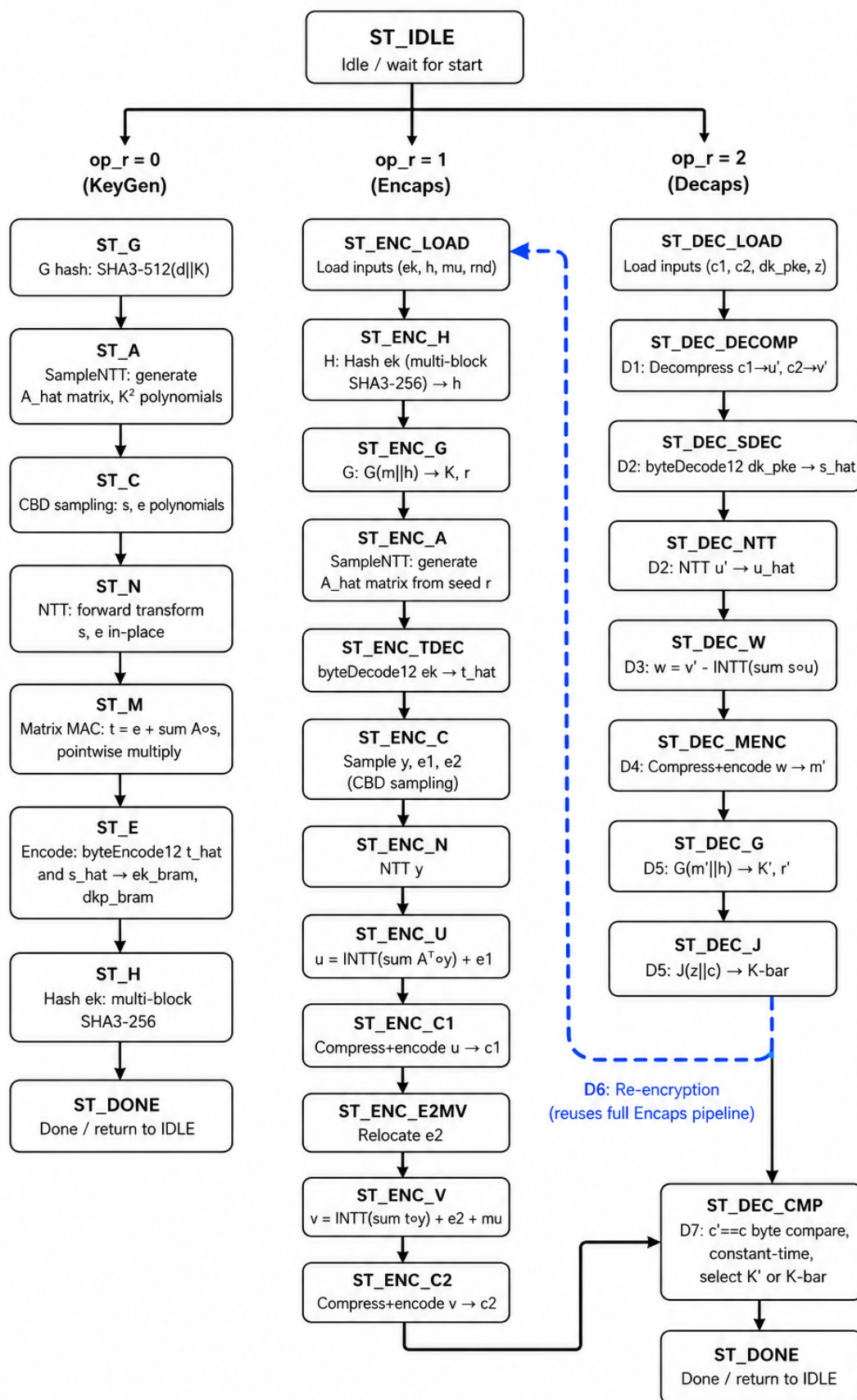


图 3: Keccak-f[1600] 核数据通路与共享机制

`sha3_top_shared` 模块在 `keccak_core` 之上封装了 SHA3/SHAKE 协议层, 包括 1088-bit 速率域/512-bit 容量域的填充、domain separator 拼接 (SHA3 末尾 01、SHAKE 末尾 1111) 以及多块顺序吸收的状态管理。支持三种模式: SHA3-512 (G 函数, 单块 64 字节吸收)、SHA3-256 (H 函数, 多块吸收, 块数 = $\lceil (ek_bytes + 1)/136 \rceil$) 和 SHAKE-256 (J 函数和 PRF 派生)。

共享 Keccak 的复用机制通过两级多路选择实现。输入侧: 三个消费者的 1600-bit 状态和 `valid` 信号经 `sel_*` 信号选择后送入 `keccak_core`; 选择信号由 FSM 状态直接推导 (例如 `sel_snt = (st==ST_A || st==ST_ENC_A)`)。输出侧: `keccak_core` 的置换结果和 `valid_o` 广播至所有消费者, 各消费者内部仅在自己激活的上下文中采样, 天然实现了无冲突分派。此复用方案将 Keccak 的硬件开销从三份实例 (约 16,000 LUT) 降至一份 (约 5,300 LUT), 是 IP 核面积仅为 25.5% LUT 利用率的关键设计决策。

4.2 NTT 核

`ntt_core` 模块实现 256 点前向/逆向数论变换, 采用单蝶形流水线加 256×12 -bit 寄存器阵列的架构。图??给出数据通路与控制状态机。

图 4: NTT 核数据通路与控制状态机

NTT 的 7 层蝶形网络按 Cooley-Tukey 位反转置换顺序执行。每层内 128 个蝶形对之间无数据依赖, 连续发射至 7 级深流水 `butterfly_unit_pipe` 而不等待前一对写回。流水线排空等待周期 (约 7 拍) 远小于一层中的蝶形总数 (128 对), 因此额外延迟占比较小。

`butterfly_unit_pipe` 内部包含一个 7 级流水的 `barrett_mul_pipe` 用于计算 $\zeta \cdot b$ 的模乘, 以及两个模加/模减路径。前向 NTT 使用 Cooley-Tukey 蝶形 (先乘 ζ 再加/减), 逆向 NTT 使用 Gentleman-Sande 蝶形 (先加/减再决定是否乘 ζ^{-1}), 模式由 `mode` 端口选择。

逆向 NTT 在完成 7 层蝶形后, 通过 `S_OUT_SCALE` 状态将每个系数乘以 $3303 \bmod q$ 以完成末尾缩放。缩放操作复用同一个 7 级 `barrett_mul_pipe` 实例, 256 个系数顺序通过流水线后输出。

`zeta_rom` 存储 128 个 12-bit 旋转因子, 位反转索引。ROM 综合为 LUT, 无需额外 BRAM。

4.3 多项式乘法与 Barrett 模乘

`poly_mul_sync` 实现 NTT 域两个多项式的逐点乘法, 即对 128 个系数对 (a_0, a_1) 和 (b_0, b_1) 按 $c_0 = a_0b_0 + a_1b_1\zeta \bmod q$ 和 $c_1 = a_0b_1 + a_1b_0 \bmod q$ 计算。内部调度为三阶段: LOAD (256 个系数, 每周周期一对) $\rightarrow$ RUN (128 次 basecase, 每次 2 周期) $\rightarrow$ OUTPUT (256 个系数)。

`basecase_mul` 为纯组合逻辑，数据通路包含 5 个 `barrett_mul` 实例。 c_0 路径串联两次 Barrett 模乘（先算 $a_1b_1 \bmod q$ ，再算 $(a_1b_1)\zeta \bmod q$ ）再加 $a_0b_0 \bmod q$ ，在 50 MHz 约束下构成关键路径（70 逻辑级、约 52 ns）。

`barrett_mul` 基于恒等式 $a \cdot b \bmod q \approx (a \cdot b \cdot \lfloor 2^{24}/q \rfloor) \gg 24$ 实现模约简。对于 $q = 3329$ ， $\lfloor 2^{24}/3329 \rfloor = 5039$ 。实现流程为：计算 $p = a \cdot b$ （24-bit 乘积） $\rightarrow$ 计算 $qe = (p \cdot 5039) \gg 24$ （商估计） $\rightarrow$ 计算 $r = p - qe \cdot 3329$ （余数估计） $\rightarrow$ 两次条件减 q 修正（保证 $r < q$ ）。当前实现为纯组合逻辑，在 25 MHz 集成频率下不构成时序违例。关于流水线化改造以支持 50 MHz 的讨论见第??节。

4.4 采样模块

`sample_ntt_sync_shared` 根据 FIPS 203 算法 14 (SampleNTT) 从种子 ρ 和索引 (j, i) 生成 $\hat{A}$ 矩阵的一个多项式（256 个系数）。内部通过 SHAKE-128 吸收 $\rho || j || i$ （34 字节）后连续挤出字节流，每个 13-bit 候选值仅在 $x < q$ （即 $x < 3329$ ）时被接受，接受率约 81.3%。逐系数通过 `valid/ready` 握手输出，直接写入 `bank_a`。

`sample_cbd_sync_shared` 实现中心二项分布 CBD_η 采样。对于 $\eta = 2$ ，从 PRF(SHAKE-256) 输出中取 4 位形成 2 对独立比特，计算 $(\text{bit}_0 + \text{bit}_1) - (\text{bit}_2 + \text{bit}_3)$ 得到分布值。对于 $\eta = 3$ （仅 ML-KEM-512），取 6 位形成 3 对。模块通过 `valid/ready` 握手逐系数输出，直接写入 `bank_se` 或 `bank_t`。

两个采样模块均在内部完成 Keccak 请求的排队：在吸收阶段结束后向共享 `keccak_core` 发起置换请求，等待核返回结果后继续挤出字节。由于采样模块与 SHA3 族消费者处于互斥的 FSM 阶段，对共享核的竞争不存在。

4.5 压缩与解压缩

`comp_decomp_sync` 实现 Compress_d 和 Decompress_d 函数。 $\text{Compress}_d(x) = \lfloor (2^d/q \cdot x) \rfloor \bmod 2^d$ ， $\text{Decompress}_d(y) = \lfloor (q/2^d \cdot y) \rfloor$ 。实现将除数 q 替换为 Barrett 风格的 $5039 \cdot 2^{-24}$ 近似，避免组合除法器。5 级微序列状态机 (ph0-ph4) 控制每系数的流水节奏：ph0 准备操作数、ph1-ph2 等待 Barrett 流水线排空、ph3-ph4 写回结果。模块逐系数通过 `valid/ready` 握手输入输出，吞吐为每 5 周期一个系数。

4.6 多项式存储库

三系数存储库是数据通路的交换中枢，全部基于 `sd_bram`（1R/1W、1 周期寄存读延迟，综合为 Block RAM）。表??列出配置和内容。

表 3: 多项式系数存储库配置

实例	尺寸	物理 BRAM	存储内容
bank_a	12-bit $\times$ 4096	1 RAMB36 + 1 RAMB18	$\hat{A}[i][j]$ 矩阵 (slot 0 至 $K^2 - 1$)
bank_se	12-bit $\times$ 2048	1 $\times$ RAMB36	$\hat{s}/\hat{e}$ (KeyGen)、 $\hat{y}$ (Encaps)、 $\hat{u}$ (Decaps)
bank_t	12-bit $\times$ 1024	1 $\times$ RAMB18	$\hat{t}$ 、MAC 累加和、 v 、 w

槽位布局为: $\hat{A}$ 占用 slot 0 至 $K^2 - 1$ (基地址 0), $\hat{s}$ 占用 slot K^2 至 $K^2 + K - 1$, $\hat{e}$ 占用 slot $K^2 + K$ 至 $K^2 + 2K - 1$, $\hat{t}$ 占用 slot $K^2 + 2K$ 至 $K^2 + 3K - 1$ 。所有槽位寻址在运行时由 k_r 递推, 例如 $\hat{s}$ 基地址 $\text{slot_s_rt} = K^2$ 。

库的读写端口在多个 FSM 状态之间分时复用。`bank_a` 的读端口在 `ST_M` ($\hat{A}$ 矩阵读取)、`ST_ENC_U` ($\hat{A}^T$ 读取)、`ST_ENC_V` ($\hat{t}$ 读取) 和调试回读之间通过组合多路选择器切换, 无额外仲裁逻辑。写端口接收来自 `ST_A` (SampleNTT 输出)、`ST_ENC_TDEC` (字节解码) 和 `ST_DEC_DECOMP` (v' 写入) 的数据。

三个字节存储库(`ek_bram`、`dkp_bram`、`ct_bram`)各为 8-bit $\times$ 2048 配置, 加上 `c_in_bram` (8-bit $\times$ 2048, Decaps 输入密文专用), 合计 4 个字节 BRAM。`c_in_bram` 与 `ct_bram` 的物理分离是 Decaps D7 步比较器的关键设计前提—— c' 与 c 的逐字节比较需要同时从两块 BRAM 读取, 独立端口避免了竞争。

5 验证方案

5.1 分层验证策略

验证分为四个层级:

- (1) **叶子模块级**: 独立验证 `keccak_core`、`ntt_core`、`sample_ntt_sync`、`sample_cbd_sync`、`poly_mul_sync`、`comp_decomp_sync` 共 6 个算子模块与 Python 参考模型的一致性。每个模块通过其 TB/子目录下的专用测试平台验证。
- (2) **操作级 KAT 验证**: 对 $k \in \{2, 3, 4\}$ 的每种操作, 通过参数化测试平台将硬件输出与 NIST FIPS 203 KAT 响应文件的标准答案逐字节比对。Encaps 和 Decaps 测试覆盖正常路径和隐式拒绝路径 (损坏密文 ct_n)。
- (3) **端到端协议验证**: `tb_mlkem_hello_world` 验证 KeyGen $\rightarrow$ Encaps $\rightarrow$ Decaps 完整往返的共享密钥一致性; `tb_mlkem_two_inst` 验证两个独立实例 (一侧生成 + 封装, 另一侧解封装) 的互操作性。
- (4) **系统级诊断**: 在 SoC 集成环境中, 通过 CPU 寄存器路径和 DMA 窗口路径各执行 100 次迭代, 验证硬件与软件参考实现的一致性及数据传输路径的正确性。

5.2 测试平台与黄金向量

表??列出全部操作级和系统级测试平台。参数化测试平台通过 XSIM 的 `xelab -generic_top K=n` 选项编译安全等级，通过 `+CASE=n` 运行时选项选择 KAT 用例编号。

表 4: 测试平台清单

测试平台	覆盖操作	安全等级	测试向量来源
tb_mlkem_kg_katK	KeyGen	K=2/3/4	NIST KAT 100 用例/级
tb_mlkem_enc_katK	Encaps	K=2/3/4	NIST KAT 100 用例/级
tb_mlkem_dec_katK	Decaps	K=2/3/4	NIST KAT 100 用例/级 + ct_n
tb_mlkem_hello_world	全流程	K=3	固定测试向量
tb_mlkem_two_inst	双实例	K=3	固定测试向量
mlkem_test (SoC)	CPU/DMA	K=2/3/4	KAT count=0 确定性输入

黄金向量来自 NIST FIPS 203 的 KAT 响应文件，由 Python 脚本解析为每用例独立的 hex 文件。每个 KAT 用例包含种子 d 、 z 、 m 以及期望的 ek 、 dk 、 c 和共享密钥。Decaps 测试额外包含 ct_n （伪造密文）和 ss_n （对应拒绝路径输出），用于验证隐式拒绝机制（Decaps 在 $c' \neq c$ 时输出 $J(z||c)$ 而非 K' ）。

Decaps 的 KAT 测试平台还通过 `dbg_*` 调试抽头逐阶段（D0–D7）验证中间量，包括 $\rho/\sigma/r$ 等哈希输出、D1 解压缩结果、D3 减法中间值、D4 恢复消息 m' 等，确保问题可精确定位到具体计算步骤。

5.3 系统级诊断测试

SoC 诊断程序 `mlkem_test` 实现三条独立测试路径：

软件参考路径在 33 MHz Loongson CPU 上运行纯 C 语言 ML-KEM 实现，记录各阶段周期数作为加速比的基准，并验证结果自洽性与黄金向量一致性。

CPU 寄存器路径通过 AXI 寄存器接口直接控制 IP 核：写入 CONFIG 配置、通过 `D[0..7]/Z[0..7]` 写入种子、通过 `PRE_SEL/ADR/DAT` 序列逐字节预装载公钥/私钥/密文、写 `CTRL.start` 启动、轮询 `STATUS.done`、通过 `RDB_SEL/ADR/DAT` 读回结果。该路径验证了 IP 核的 AXI 从机接口逻辑和寄存器级控制流的正确性。

DMA 窗口路径利用 SoC 的 DMA 控制器在系统存储器 and IP 核的 EK/DK/CT 窗口（偏移 `0x1000/0x2000/0x3000`）之间批量搬移数据，将数据传输开销从数千次 AXI 单字节事务降低为数次 DMA 配置操作。该路径验证了 IP 核存储器窗口与 DMA 控制器之间的数据搬移正确性，以及批量传输模式相对于逐字节读写的效率提升。

每种路径对每个安全等级执行 100 次迭代，记录自洽性（`ss_enc == ss_dec`）和黄金向量一致性。

6 实现结果与测试数据

6.1 FPGA 资源利用

ML-KEM IP 核在 Xilinx Artix-7 xc7a200tfbg676-1 上完成布局布线后的独立资源利用如表??所示。

表 5: mlkem_top 布局布线后资源利用 (25 MHz 集成频率)

资源	已用	可用	利用率
Slice LUTs	31,884	133,800	23.8%
Slice FFs	24,736	267,600	9.2%
Block RAM Tile	5.5	365	1.5%
DSP48E1	0	740	0.0%

DSP 使用量为零体现了设计的主动约束——所有乘法（包括 NTT 蝶形中的 Barrett 模乘和 `basecase_mul` 中的乘法）均强制映射至 LUT 和 CARRY4 逻辑，不消耗任何 DSP48E1 资源。该约束的动机是规避早期 XSIM 仿真中 DSP48 行为模型的不一致问题，代价是关键路径中的乘法延迟增加（见第??节）。

表??给出布局布线后的子模块级资源分解。

表 6: mlkem_top 主要子模块资源利用

子模块	LUT	FF
ntt_core (含蝶形流水线与 Barrett)	8,625	3,861
keccak_core (Keccak-f[1600] 置换核)	5,274	1,606
poly_mul_sync (含 basecase_mul)	3,881	6,743
sample_cbd_sync_shared	2,741	3,186
sample_ntt_sync_shared	2,109	1,669
sha3_top_shared (哈希族封装)	1,766	2,388
comp_decomp_sync	467	323
存储库及胶合逻辑	4,302	0
其它 (FSM/多路复用/地址生成/顶层连线)	2,719	4,960
合计	31,884	24,736

NTT 核是最大的单一逻辑单元 (8,625 LUT)，主要由 256 系数寄存器阵列和 7 级深流水 Barrett 蝶形单元贡献。Keccak 核 (5,274 LUT) 的 1600-bit 状态寄存器和 5 路并行置换逻辑构成第二大开销。多项式乘法模块虽 LUT 适中 (3,881)，但 FF 数量最高 (6,743)，主要因其内部 `basecase_mul` 的串行双 Barrett 乘积累加寄存器阵列。

6.2 时序性能

表??列出布局布线后的时序收敛情况。ML-KEM IP 工作于 25 MHz 的独立时钟域 (PLL CLKOUT2 输出，周期 40 ns)。

表 7: 布局布线后时序收敛结果 (25 MHz 集成频率)

时钟域	Setup WNS	违例端点	最大可行频率
mlkem_clk (25 MHz)	+10.807 ns	0	≈ 34.3 MHz

mlkem_clk 域的 +10.807 ns 正余量表明，在 25 MHz 下所有路径延迟远小于 40 ns 的时钟周期。最差路径位于 sha3_mb_en0 高扇出网络（驱动 936 个负载），路由延迟 (28.345 ns) 占路径总延迟 (29.049 ns) 的 97.6%，逻辑延迟仅 0.704 ns (2 级 LUT)。

6.3 功耗分析

布局布线后功耗估计如表??所示。ML-KEM IP 核的片上功耗约 0.115 W（动态），占 SoC 总动态功耗 (0.409 W) 的 28.1%。

表 8: ML-KEM IP 功耗估计（布局布线后，典型工艺角）

功耗来源	功率 (W)	说明
Slice Logic	0.046	LUT 翻转 + FF 时钟功耗
Signals	0.052	互联线翻转功耗
Block RAM	0.017	7 个 BRAM 实例的读写功耗
ML-KEM 合计	0.115	占 SoC 动态功耗 28.1%

6.4 功能验证结果

6.4.1 KAT 逐字节比对

表??汇总全部安全等级和操作的 KAT 验证结果。

表 9: KAT 验证结果汇总

操作	K=2	K=3	K=4	用例数	通过率
KeyGen (算法 16)	PASS	PASS	PASS	100/级	100%
Encaps (算法 17)	PASS	PASS	PASS	100/级	100%
Decaps 正常路径 (算法 18)	PASS	PASS	PASS	100/级	100%
Decaps 隐式拒绝路径	PASS	PASS	PASS	100/级	100%

全部 360 个 KAT 测试用例均逐字节比对 NIST 标准答案通过，无任何字节级偏差。Decaps 隐式拒绝路径 ($c'_n \neq c$) 验证了 FO 变换中常量时间比较器和 $J(z||c)$ 拒绝密钥生成逻辑的正确性。

6.4.2 系统级诊断测试

SoC 集成诊断测试在 ML-KEM-768 条件下对三种数据路径各执行 100 次迭代，结果如表??所示。

表 10: 系统级诊断测试结果 (ML-KEM-768, 100 次迭代)

测试路径	失败次数	自愈性通过率	黄金向量通过率
SW (纯软件参考)	0	100%	100%
CPU 寄存器路径	0	100%	100%
DMA 窗口路径	0	100%	100%

6.5 性能分析

表??列出三种安全等级下软件参考、CPU 寄存器路径和 DMA 窗口路径完成一轮完整往返 (KeyGen+Encaps+Decaps) 的平均时钟周期数及加速比。数据来自 SoC 实际运行的 100 次迭代统计。

表 11: 三路径完整往返性能对比 (100 次迭代平均值)

安全等级	SW	CPU	DMA	CPU 加速比	DMA 加速比	DMA/CPU
ML-KEM-512 ($k=2$)	2,587,784	352,684	185,671	7.34×	13.94×	1.90×
ML-KEM-768 ($k=3$)	4,056,378	523,656	279,588	7.75×	14.51×	1.87×
ML-KEM-1024 ($k=4$)	6,025,486	718,672	388,959	8.38×	15.49×	1.85×

从数据中可以得出以下结论。CPU 寄存器路径相对于纯软件实现了 7.34× 至 8.38× 的加速比，且加速比随安全等级 k 增大而略有提升——这是因为硬件加速器的多项式乘法和 NTT 运算复杂度以 $O(k^2)$ 增长，与软件实现的复杂度增长一致，但硬件的固定流水线启动开销占比随总计算量的增大而降低。

DMA 路径相对于 CPU 寄存器路径额外贡献了约 1.85–1.90× 的效率提升。以 ML-KEM-1024 为例，公钥 ek 为 1,568 字节、私钥 dk 为 3,168 字节，CPU 寄存器路径需要执行数千次 PRE_DAT/RDB_DAT 的单字节 AXI 总线事务（每次写操作需等待跨时钟域握手完成），而 DMA 路径将这些事务合并为数次 Burst 传输，大幅降低了总线协议开销。

6.6 其他安全等级结果

ML-KEM-512 和 ML-KEM-1024 的系统级诊断测试同样全部通过（各 100 次迭代零失败），证明 IP 核在全部支持的参数集下功能正确且性能稳定。

7 结论与展望

本文设计并实现了一款完整的 ML-KEM 后量子密钥封装硬件 IP 核。核心贡献总结如下：

运行时可重构架构：通过存储按 $K_{MAX} = 4$ 静态分配、运行时参数动态推导的策略，单一比特流支持 ML-KEM-512/768/1024 三种安全等级的 KeyGen/Encaps/Decaps 全操作，避免了为不同等级维护多份综合结果的工程开销。

高效资源共享：单一 keccak_core 在哈希、采样和 CBD 三消费者间跨阶段复用，将 OOC 综合 LUT 利用率降至 23.8%；Decaps D6 步重加密复用整条 Encaps 流水线，避免了约 800 行 RTL 的重复实现。

完整标准合规：360 个 NIST KAT 测试用例全部逐字节比对通过，覆盖三种等级三种操作及隐式拒绝路径。端到端协议测试和系统级诊断验证了软硬件协同的完整正确性。

显著性能提升：相对于 33 MHz CPU 上的软件参考实现，25 MHz 硬件 IP 在 CPU 寄存器路径实现 7.3–8.4 $\times$ 加速比，在 DMA 路径实现 13.9–15.5 $\times$ 加速比。

后续工作方向包括：（1）实施 Barrett 模乘三段流水线化，将 IP 核提升至 50 MHz 独立工作频率；（2）引入多 Bank 交叉访问和 NTT 计算并行化以进一步提升吞吐；（3）添加功耗平衡和随机延迟插入等侧信道防护措施；（4）将设计移植至国产 FPGA 平台进行适配性验证。

致谢 本项目在龙芯中科和 Xilinx 大学计划提供的软硬件平台上完成。

参考文献

- [1] National Institute of Standards and Technology. *FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard*. August 2024.
- [2] National Institute of Standards and Technology. *FIPS 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. August 2015.
- [3] R. Avanzi et al.. *CRYSTALS-Kyber: Algorithm Specifications and Supporting Documentation*. NIST PQC Round 3 Submission, 2021.
- [4] V. B. Dang, F. Farahmand, M. Andrzejczak, and K. Gaj. *Implementing and Benchmarking CRYSTALS-Kyber on FPGAs*. IEEE Trans. Computers, 71(10):2475–2488, 2022.

-
- [5] Y. Xing and S. Li. *A Compact Hardware Architecture for CRYSTALS-Kyber*. IEEE Trans. Circuits Syst. I, 70(3):1124–1137, 2023.
 - [6] P. Barrett. *Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor*. CRYPTO 1986, pp. 311–323.
 - [7] Xilinx Inc. *Vivado Design Suite User Guide: Synthesis (UG901)*. v2019.2, 2019.
 - [8] Xilinx Inc. *7 Series FPGAs Data Sheet: Overview (DS180)*. v2.6, 2020.