

# Design and Implementation of a RISC-V ML-KEM-768 and AES-128 Cryptographic Accelerator

T. Yeswanth<sup>1</sup>, Smt. Dr. G. Jyoshna<sup>2</sup>

<sup>1</sup>PG Student, Department of Electronics and Communication Engineering, Embedded Systems and VLSI Design, Sri Krishnadevaraya University College of Engineering and Technology, Anantapuramu, Andhra Pradesh, India.

<sup>2</sup>Assistant Professor (Adhoc), Department of Electronics and Communication Engineering, Sri Krishnadevaraya University College of Engineering and Technology, Anantapuramu, Andhra Pradesh, India.

Student e-mail: <sup>1</sup>yeswanthturaka@gmail.com

**Abstract**—This paper presents a reference RTL architecture that integrates ML-KEM-768 and AES-128 in a single accelerator accessible from a RISC-V system-on-chip through an AXI4-Lite interface. The design comprises a Keccak-f[1600] permutation core, a centered-binomial noise sampler, an iterative number-theoretic transform (NTT/INTT) engine, a uniform rejection sampler, compression/decompression logic, a KEM controller, and an iterative AES-128 core. Its principal architectural feature is a hardware path that routes the ML-KEM shared-secret object directly to the AES key input, so that normal operation does not require software to read and rewrite the session key. The AES core is verified by an encryption/decryption round-trip simulation; the ML-KEM primitives are simulated individually, and the integrated controller is exercised in an end-to-end simulation. Complete ML-KEM known-answer-test validation, the full Fujisaki-Okamoto re-encryption path, and hardware synthesis results are not reported, so the cycle and throughput figures are architectural estimates rather than measurements. The work defines a practical hardware/software boundary for combining post-quantum key establishment with symmetric encryption and lists the verification steps required before deployment.

**Index Terms**—Advanced Encryption Standard (AES), AXI4-Lite, hardware accelerator, ML-KEM, number-theoretic transform (NTT), post-quantum cryptography, RISC-V.

## I. INTRODUCTION

The transition to post-quantum cryptography (PQC) is motivated by the possibility that sufficiently capable quantum computers could undermine widely deployed public-key mechanisms [14]. In response, the National Institute of Standards and Technology (NIST) standardized ML-KEM as a key-encapsulation mechanism in FIPS 203, which defines the parameter sets ML-KEM-512, ML-KEM-768, and ML-KEM-1024 [1]; the companion lattice-based digital signature standard ML-DSA is specified separately in FIPS 204 [3] and is outside the scope of this work. Once a shared secret has been established, symmetric cryptography can protect bulk data. AES remains the standard symmetric block cipher, operating on 128-bit blocks with 128-, 192-, or 256-bit keys [2]. Because Grover's algorithm offers a quadratic speed-up for unstructured key search [15], the key length of the symmetric cipher remains a design consideration.

Although the two algorithms are standardized independently, an embedded implementation must also decide how the session key moves between the key-establishment engine and the bulk-encryption engine. In a software-mediated arrangement the shared secret may pass through processor registers, memory, or

an interconnect before reaching an AES peripheral. This is undesirable when the design objective is to minimize processor-visible key material.

RISC-V is well suited to studying this problem because its open instruction-set architecture permits several accelerator-integration strategies. Prior work has coupled lattice-cryptography hardware with RISC-V processors and instruction-set architectures [7], [10], and AES accelerators have been integrated with RISC-V cores in separate work [12], [13]. These works show that lattice arithmetic and AES can each be accelerated close to a processor, but the reviewed works do not describe the hardware boundary between a PQC-derived shared secret and a subsequent AES datapath.

This paper therefore focuses on integration rather than on a new cryptographic primitive. The main contributions are:

- (i) a modular RTL architecture in which ML-KEM-768 and AES-128 reside in one accelerator and the KEM-derived 256-bit shared-secret object is routed internally to the AES key path;
- (ii) a register-oriented AXI4-Lite interface for integration with a RISC-V SoC;
- (iii) round-trip simulation of the AES core, together with simulation-level exercise of the ML-KEM primitives and controller; and
- (iv) an explicit separation of verified results, simulation-level observations, architectural estimates, and future work.

The remainder of the paper is organized as follows. Section II reviews related work. Section III describes the architecture and its functional blocks. Section IV presents the implementation and verification setup. Section V reports results, Section VI discusses security considerations and limitations, and Section VII concludes.

## II. RELATED WORK

### A. Cryptographic Background and Motivation

Public-key mechanisms based on integer factorization and discrete logarithms are vulnerable to Shor's quantum algorithm [14], which solves both problems in polynomial time on a sufficiently large quantum computer. Symmetric primitives are affected differently: Grover's search algorithm [15] gives a quadratic reduction in the cost of exhaustive key search, so the key length of a symmetric cipher remains a design consideration. NIST has standardized the key-encapsulation mechanism ML-KEM, derived from CRYSTALS-Kyber [4], in

FIPS 203 [1], and the lattice-based digital signature algorithm ML-DSA in FIPS 204 [3]. The two serve different purposes: ML-KEM establishes a shared secret between two parties, whereas ML-DSA provides authentication through digital signatures. Bulk data protection continues to rely on symmetric ciphers such as AES [2], and ML-KEM internally uses the SHA-3 and SHAKE functions defined in FIPS 202 [19]. This work concerns ML-KEM and AES; digital signatures are outside its scope.

### B. Hardware Implementations of CRYSTALS-Kyber

Xing and Li presented a standalone FPGA implementation of CRYSTALS-Kyber in which sampling and NTT-related computations are scheduled to obtain good performance with limited hardware resources [5]. Jati et al. proposed a configurable FPGA implementation with side-channel and fault protection; its low-area configuration uses 5269 LUTs and 2422 FFs, its high-performance configuration uses 7151 LUTs and 3730 FFs, the clock frequency exceeds 250 MHz on 28 nm Xilinx devices, and the protection overhead is below 5% [6]. These figures are taken from the authors' abstract. He et al. describe a lightweight architecture intended for ASICs and prototyped on a Kintex-7 FPGA, with a compact modular multiplication unit, a compression/decompression module, and a schoolbook polynomial multiplier used in place of an NTT core [11]. Chen et al. address area-efficient, high-performance polynomial arithmetic for Kyber through a polynomial ring processor on FPGAs [9].

### C. NTT, Polynomial Arithmetic, and Compression

The NTT is the most computationally demanding step of Kyber because polynomial multiplication in the ring is performed in the transform domain; the ring and transform parameters are given in the specification [4] and in FIPS 203 [1]. Zhang et al. optimized the NTT with a fast modular multiplication method and a ping-pong memory access scheme with doubled bandwidth, completing one Kyber NTT in 490 cycles with 609 LUTs, 640 FFs, and 2 DSPs on an Artix-7 FPGA [8]. Modular multiplication and compression/decompression hardware are treated explicitly in [11]. The NTT engine of the present design processes one butterfly per cycle and is described in Section III-E; its cycle count is an analytical estimate and is not compared with the figures reported in [8].

### D. Configurable Processors, Instruction-Set Support, and Integration

Banerjee et al. coupled a configurable lattice crypto-processor, programmed with custom instructions for polynomial arithmetic and sampling, to a low-power RISC-V microprocessor; the test chip was fabricated in a TSMC 40 nm low-power CMOS process, and the cryptographic core occupies 0.28 mm<sup>2</sup> with 106k logic gates and 40.25 KB of SRAM [7]. Its building blocks are constant-time, and masking countermeasures are discussed. Bisheh-Niasar et al. present an instruction-set hardware architecture for Kyber with ASIC and FPGA implementations [10]. Both approaches place lattice arithmetic behind a programmable instruction interface. The present work instead uses a memory-mapped register interface based on AXI4-Lite [16], which suits control and status

exchange without burst transfers. FPGA system-on-chip devices such as the Zynq UltraScale+ MPSoC, whose interfaces are documented in [17], provide AXI-based connections between processors and programmable logic. The processor in this work implements the RISC-V base integer instruction set [18].

### E. AES Acceleration on RISC-V

Gomes et al. present FAC-V, an FPGA-based AES coprocessor for RISC-V that can be deployed with tight or loose coupling; for a 16-byte message encrypted with AES-256 they report a performance gain of about 8000 times over software and an energy of 0.1  $\mu$ J [13]. Nguyen et al. propose AES-RV, a RISC-V accelerator with low-latency AES instruction extensions that supports all AES modes and key sizes and uses high-bandwidth buffers and a ping-pong memory transfer mechanism [12]. These works accelerate AES alone; neither addresses how the AES key is obtained from a post-quantum key-establishment stage.

### F. Comparison and Research Gap

Table I compares the reviewed implementations using only values stated in the cited sources; "Not reported" marks metrics that are not stated in the abstracts and summaries consulted for this survey.

TABLE I  
COMPARISON OF RELATED HARDWARE IMPLEMENTATIONS

| Ref. | Platform and approach                                                                      | Reported results                                                                    | Integration / protection                             |
|------|--------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|------------------------------------------------------|
| [5]  | FPGA; standalone CRYSTALS-Kyber design with scheduled sampling and NTT                     | Not reported                                                                        | Hardware only; side-channel protection not reported  |
| [6]  | FPGA (28 nm Xilinx); configurable Kyber implementation                                     | Low-area: 5269 LUTs, 2422 FFs; high-performance: 7151 LUTs, 3730 FFs; above 250 MHz | Side-channel and fault protection; overhead below 5% |
| [7]  | ASIC (TSMC 40 nm LP); configurable lattice crypto-processor with custom instructions       | Core: 0.28 mm <sup>2</sup> , 106k gates, 40.25 KB SRAM                              | Coupled to a RISC-V processor; constant-time blocks  |
| [8]  | FPGA (Artix-7); NTT core for Kyber                                                         | 490 cycles per NTT; 609 LUTs, 640 FFs, 2 DSPs                                       | NTT core only                                        |
| [9]  | FPGA; polynomial ring processor for Kyber                                                  | Not reported                                                                        | Not reported                                         |
| [10] | ASIC and FPGA; instruction-set architecture for Kyber                                      | Not reported                                                                        | Instruction-set processor                            |
| [11] | FPGA prototype (Kintex-7); ASIC-oriented lightweight Kyber using schoolbook multiplication | 4777/4993 LUTs, 2661/2765 FFs, 2.5/2.5 BRAMs, 0 DSPs (client/server)                | Hardware only                                        |
| [12] | FPGA; RISC-V AES accelerator with custom instructions                                      | Not reported                                                                        | ISA extension; all AES modes and key sizes           |
| [13] | FPGA; AES coprocessor for RISC-V                                                           | About 8000 times faster than software, 0.1 $\mu$ J (16-byte AES-256)                | Tight (RoCC) and loose (memory-mapped) coupling      |

| Ref.      | Platform and approach                                | Reported results                 | Integration / protection                                     |
|-----------|------------------------------------------------------|----------------------------------|--------------------------------------------------------------|
| This work | Simulated RTL; ML-KEM-768 and AES in one accelerator | No area, timing or power results | AXI4-Lite to 5-stage RISC-V core; no side-channel protection |

| Parameter         | Value      |
|-------------------|------------|
| Modulus $q$       | 3329       |
| Module rank $k$   | 3          |
| $\eta_1$          | 2          |
| $\eta_2$          | 2          |
| $d_u$             | 10         |
| $d_v$             | 4          |
| Encapsulation key | 1184 bytes |
| Ciphertext        | 1088 bytes |
| Shared secret     | 32 bytes   |

The reviewed works address three largely separate concerns: Kyber hardware in its complete or partial form [5], [6], [9], [11]; NTT and processor-integrated lattice acceleration [7], [8], [10]; and AES acceleration for RISC-V cores [12], [13]. Side-channel countermeasures are described in [6] and [7] but not in the AES-oriented works. Among the reviewed works, none describes a hardware path that routes a KEM-derived shared secret into an AES key input inside a single accelerator that a RISC-V core accesses through a memory-mapped interface. This observation is limited to the cited works and is not an exhaustive survey. The present design addresses this narrower integration point. Because only simulation results are available, no claim is made that it is smaller, faster, or more energy-efficient than the works in Table I.

### III. PROPOSED ARCHITECTURE

#### A. System Overview

The system has three layers: an ML-KEM-768 key-establishment layer, an AES-128 symmetric-encryption layer, and a processor-facing interface and control layer. The top-level module, *pqc\_aes\_top*, instantiates the KEM controller and the AES core and connects them through an internal shared-secret path (Fig. 1). Firmware writes control information and input data through AXI4-Lite. The KEM controller performs its internal sequence and produces the shared-secret object, from which the AES key input is driven. The processor then supplies a 128-bit plaintext or ciphertext block and starts AES encryption or decryption; status flags indicate completion.

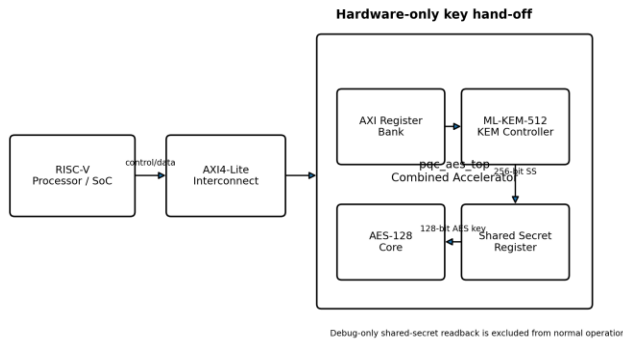


Fig. 1. Top-level RISC-V-accessible ML-KEM-768 and AES-128 accelerator architecture.

#### B. ML-KEM-768 Parameters

The ML-KEM-768 parameters used by the architecture [1], [4] are listed in Table II.

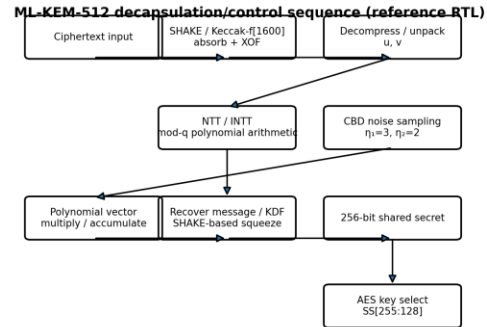
TABLE II

ML-KEM-768 PARAMETERS USED BY THE REFERENCE ARCHITECTURE

| Parameter             | Value |
|-----------------------|-------|
| Security category     | 3     |
| Polynomial degree $n$ | 256   |

#### C. Processing Flow

Fig. 2 shows the controller-level sequencing of the ML-KEM-768 processing flow, in which the controller orders the Keccak, sampling, transform, and compression blocks without requiring the processor to orchestrate every polynomial operation.



The full FO re-encryption datapath is not yet complete in the thesis implementation.

Fig. 2. ML-KEM-768 decapsulation/control sequence of the reference RTL (full Fujisaki–Okamoto re-encryption not yet complete).

#### D. RTL Module Organization

The principal modules are the top-level accelerator, KEM controller, Keccak permutation, CBD sampler, NTT/INTT engine, rejection sampler, compression/decompression logic, AES core, and AXI4-Lite register interface (Fig. 3). This organization permits independent verification of each primitive and isolates future optimization work such as modular-reduction replacement or transform parallelization.

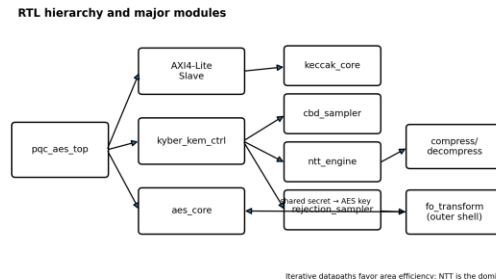


Fig. 3. RTL hierarchy and major modules of the integrated accelerator.

#### E. ML-KEM Functional Blocks

*Keccak-f[1600] and sampling.* The Keccak engine provides the permutation underlying the SHA-3/SHAKE functions [19] used throughout ML-KEM, and is implemented as an iterative

24-round organization. The centered-binomial distribution (CBD) sampler consumes pseudorandom bytes to form coefficient noise according to the parameter  $\eta$ ; for ML-KEM-768, where  $\eta = 2$ , four bits are consumed per coefficient.

**NTT/INTT engine.** The engine operates over the ring  $\mathbb{Z}_{3329}[X]/(X^{256}+1)$ , uses  $\zeta = 17$  as the primitive-root parameter [4], and organizes the transform as seven butterfly stages with one butterfly processed per clock cycle. Seven stages of 128 butterflies give 896 butterfly operations, so the transform requires at least 896 cycles before control, load, and store overhead is added. This is an analytical lower bound derived from the stage structure; it has not been verified against the RTL state machine or measured in simulation. The overhead cycles of the implemented controller have not been determined.

**Compression, decompression and rejection sampling.** The datapath contains compression/decompression logic matching the ML-KEM parameterization and a uniform rejection-sampling block used for matrix generation. These modules were simulated individually.

**KEM controller and shared-secret path.** The controller sequences Keccak processing, CBD sampling, NTT processing, compression/decompression, and the final internal state operations of the reference design. The resulting 256-bit shared-secret object is stored internally, and the 128-bit AES key input is driven from *shared\_secret\_reg[255:128]*. A debug-oriented shared-secret register is present in the reference RTL for verification only; it is not a production security feature and must be removed or access-controlled before deployment.

### F. AES-128 Core

The core implements the standard AES-128 sequence: an initial AddRoundKey, nine regular rounds of SubBytes, ShiftRows, MixColumns, and AddRoundKey, and a final round without MixColumns [2]. Decryption uses the corresponding inverse transformations. The core is iterative, processes one round per cycle, and uses a fully combinational key schedule, selected to keep the control structure simple and to avoid the state hazards encountered in an earlier iterative key-expansion approach.

### G. AXI4-Lite Interface

The accelerator is an AXI4-Lite [16] slave, chosen because the reference design is register-oriented and needs no burst transfers. The implemented register map is given in Table III.

TABLE III  
AXI4-LITE REGISTER MAP OF THE REFERENCE IMPLEMENTATION

| Offset    | Register           | Acc. | Function                                            |
|-----------|--------------------|------|-----------------------------------------------------|
| 0x00      | CTRL               | R/W  | bit 0 KEM start; bit 1 AES start; bit 2 AES decrypt |
| 0x04      | STATUS             | R    | KEM busy/done; AES done                             |
| 0x10–0x1C | KYBER_CT[0:3]      | W    | Current 128-bit ciphertext window                   |
| 0x20–0x2C | AES_DATA_IN[0:3]   | W    | 128-bit AES input block                             |
| 0x30–0x3C | AES_DATA_OUT[0:3]  | R    | 128-bit AES result                                  |
| 0x40–0x4C | SHARED_SECRET[0:3] | R    | 256-bit debug/verification object                   |

The ciphertext interface is a deliberate reference simplification: a complete ML-KEM-768 ciphertext is 1088 bytes, whereas the present map exposes a 128-bit window. A production design needs repeated transfers through a streaming/FIFO mechanism or an expanded memory interface.

## IV. IMPLEMENTATION AND VERIFICATION SETUP

The design is implemented as synthesizable Verilog RTL organized into the modular blocks of Section III. Functional verification used directed simulation with self-checking testbench concepts. The simulator and version were Xilinx Vivado Simulator 2025.2. The accelerator is attached through AXI4-Lite to a 5-stage pipelined RISC-V core implementing the RV32I base integer instruction set [18]. The AES core was checked by an encryption/decryption round-trip simulation; the Keccak, CBD, NTT/INTT, rejection-sampling, and compression/decompression blocks were simulated individually; and the KEM controller was exercised end to end as a smoke test.

The evaluation is simulation-based. The metrics used are verification status (Table IV) and architectural estimates of cycles per block and throughput, computed from the implemented organization as described in Section V-C.

## V. RESULTS AND DISCUSSION

Results are divided into verified functional results, simulation-level observations, and architectural estimates, because the evaluation is simulation-based.

### A. AES-128 Verification

The AES datapath of Fig. 4 was simulated by encrypting and then decrypting one 128-bit block (Fig. 5). The key input in this run was 256 bits wide. The plaintext 0xcfdcbcfdaedbbcfcadfdebbcfadd produced the ciphertext 0x06d4efae04690518c41d2a767a90d9ed, and decryption returned the original plaintext. This is a functional round-trip check with a user-defined vector. It is not a known-answer test, because the ciphertext was not compared with an independently computed or published expected value, and it does not demonstrate AES-128 operation with a 128-bit key. A known-answer test using the FIPS 197 vectors [2] remains to be performed.

AES-128 iterative encryption/decryption datapath

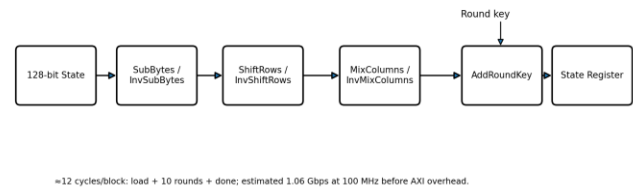


Fig. 4. AES-128 iterative encryption/decryption datapath used in the RTL implementation.

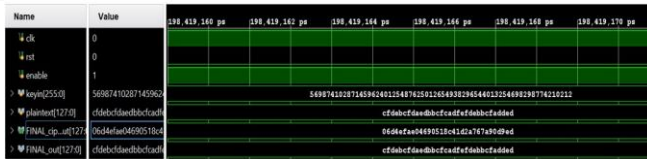


Fig. 5. Vivado Simulator waveform of the AES encryption/decryption round trip with a 256-bit key input.

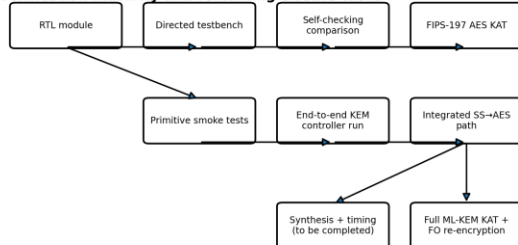
### B. ML-KEM Primitive Verification

The Keccak-f[1600], CBD, NTT/INTT, rejection-sampling, and compression/decompression blocks were individually simulated. The KEM controller was exercised end to end in simulation without deadlock and produced a 256-bit shared-secret object. These results show structural and functional integration at simulation level, but they are not complete bit-exact FIPS 203 known-answer-test (KAT) validation. Table IV and Fig. 6 summarize the verification status and methodology.

TABLE IV  
VERIFICATION STATUS OF THE REFERENCE IMPLEMENTATION

| Block / function                | Verification status                                 | Evidence level                                    |
|---------------------------------|-----------------------------------------------------|---------------------------------------------------|
| AES-128 encrypt/decrypt         | Round-trip simulation passed                        | Custom vector, 256-bit key; AES-128 KAT not run   |
| Keccak-f[1600]                  | Individually simulated                              | Functional simulation                             |
| CBD sampler                     | Individually simulated                              | Functional simulation                             |
| NTT/INTT                        | Individually simulated                              | Functional simulation; INTT final scaling remains |
| Rejection sampler               | Individually simulated                              | Functional simulation                             |
| Compress/Decompress             | Individually simulated                              | Functional simulation                             |
| KEM controller                  | End-to-end smoke test                               | Not a full ML-KEM KAT                             |
| Integrated RISC-V + accelerator | Simulated; decrypted output did not match plaintext | Failing; debugging pending                        |
| FO re-encryption                | Outer comparison/selection shell only               | Incomplete                                        |

### Verification maturity and remaining closure tasks



Green-equivalent status in the thesis: AES KAT passed; ML-KEM primitives individually exercised; full KAT/FO/synthesis remain.

Fig. 6. Verification methodology, separating vector-based verification from simulation-level checks.

Fig. 7 shows the KEM controller simulation: after the 256-bit seed is applied and start is pulsed, key\_out presents a 256-bit shared-secret value. No comparison against a reference ML-KEM implementation has been made, so this demonstrates controller operation only. Fig. 8 shows the integrated PQC-secured RISC-V simulation. In this run the 32-bit plaintext 22550977 was applied, but the decrypted output (986750166,

3242170914 and 1056486517 at successive valid pulses) did not equal the plaintext. The integrated encryption/decryption path is therefore not yet functionally verified, and no end-to-end correctness claim is made.

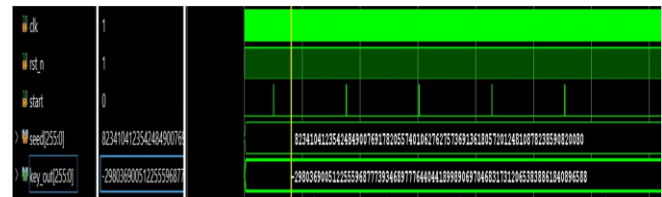


Fig. 7. Vivado Simulator waveform of the ML-KEM-768 controller (seed input and 256-bit key\_out).



Fig. 8. Integrated PQC-secured RISC-V simulation; decrypted output does not yet match the plaintext.

### C. Performance Estimates

An analytical estimate for the AES controller is approximately 12 cycles per 128-bit block, counting load, ten rounds, and completion control; this count has not been verified against the RTL state machine or measured in simulation. With block size  $B = 128$  bits and clock frequency  $f_{clk}$ , the throughput before AXI4-Lite and software overhead is

$$R = B \cdot f_{clk} / N_{cyc} \quad (1)$$

where  $N_{cyc} \approx 12$  is the estimated number of cycles per block. At an assumed  $f_{clk} = 100$  MHz, this gives approximately 8.3 million blocks per second, or about 1.07 Gbit/s. The 100 MHz clock is an assumed operating frequency, not a post-implementation timing result. Table V lists the available performance information and its evidence level.

TABLE V  
AVAILABLE PERFORMANCE INFORMATION AND ITS EVIDENCE LEVEL

| Metric                    | Value / status                       | Classification         |
|---------------------------|--------------------------------------|------------------------|
| AES block size            | 128 bits                             | Specification          |
| AES key size              | 128 bits                             | Specification          |
| AES cycles per block      | $\approx 12$                         | Analytical, unverified |
| AES throughput at 100 MHz | $\approx 1.07$ Gbit/s                | Analytical, unverified |
| NTT transform latency     | $\geq 896$ cycles (butterflies only) | Analytical, unverified |

No comparison with published implementations is made, because those works report measured results under different platforms and conditions, whereas the present figures are architectural estimates.

### VI. SECURITY CONSIDERATIONS AND LIMITATIONS

The design addresses one specific architectural exposure: the software-mediated movement of the KEM shared secret into a separate AES accelerator. Because the KEM-derived object feeds the AES key path directly, the normal control flow does

not require software to copy the key through general-purpose registers or memory, which can reduce key exposure at the software interface (Fig. 1).

This property does not establish the cryptographic security of the complete implementation. The KEM controller does not yet implement the full Fujisaki–Okamoto re-encryption and implicit-rejection path required for complete ML-KEM decapsulation; the current *fo transform* shell contains only comparison and selection logic. Side-channel resistance (timing, power, electromagnetic leakage), fault injection, masking, and secure key erasure are outside the demonstrated scope, and the debug shared-secret register must be removed or access-controlled in a production version.

Further limitations are as follows. Full ML-KEM-768 KAT validation has not been completed. The recorded AES simulation used a 256-bit key input, so AES-128 operation with a 128-bit key and the FIPS 197 known-answer vectors have not been demonstrated. The cycle counts in Section V are analytical estimates that have not been verified against the RTL or measured in simulation, and in the integrated simulation the decrypted output did not match the plaintext. The INTT final  $n^{-1} \bmod q$  scaling remains to be completed before a full inverse-transform identity can be claimed. Matrix generation currently demonstrates a representative lane rather than the complete  $3 \times 3$  matrix. The 128-bit ciphertext window is insufficient for the 1088-byte ciphertext. Finally, the work is simulation-only, and no resource utilization, timing, or power figures are reported.

### VII. CONCLUSION AND FUTURE WORK

This paper presented a reference RTL architecture integrating ML-KEM-768 and AES-128 into a single RISC-V-accessible cryptographic accelerator. Its principal contribution is the hardware integration boundary: the KEM-derived shared-secret object is routed internally to the AES key path rather than being transferred through processor software. The AES datapath passed an encryption/decryption round-trip simulation, while the ML-KEM primitives and controller were exercised in simulation. The paper does not claim complete FIPS 203 KAT validation, complete Fujisaki–Okamoto decapsulation, hardware synthesis results, or side-channel resistance.

Future work, in priority order, comprises: (1) an AES FIPS 197 known-answer test and standalone Keccak and NTT known-answer tests; (2) completion of the INTT normalization; (3) full ML-KEM-768 KAT validation against the standardized reference; (4) replication of all required matrix-A lanes; (5) completion of encapsulation and Fujisaki–Okamoto re-encryption; (6) streaming ciphertext support; (7) modular-reduction optimization; (8) FPGA synthesis, timing, and power analysis; and (9) side-channel and fault-security hardening. The verification environment should also be extended with SystemVerilog Assertions and, where appropriate, UVM components.

### ACKNOWLEDGMENT

The author thanks the Department of Electronics and Communication Engineering, Sri Krishnadevaraya University

College of Engineering and Technology, for guidance and academic support.

### REFERENCES

- [1] National Institute of Standards and Technology, “Module-lattice-based key-encapsulation mechanism standard,” Federal Information Processing Standards Publication (FIPS PUB) 203, Aug. 2024.
- [2] National Institute of Standards and Technology, “Advanced Encryption Standard (AES),” FIPS PUB 197, Nov. 2001 (updated May 2023).
- [3] National Institute of Standards and Technology, “Module-lattice-based digital signature standard,” FIPS PUB 204, Aug. 2024.
- [4] R. Avanzi *et al.*, “CRYSTALS-Kyber: Algorithm specifications and supporting documentation,” NIST Post-Quantum Cryptography Round 3 Submission, 2021.
- [5] Y. Xing and S. Li, “A compact hardware implementation of CCA-secure key exchange mechanism CRYSTALS-KYBER on FPGA,” *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2021, no. 2, pp. 328–356, Feb. 2021, doi: 10.46586/tches.v2021.i2.328-356.
- [6] A. Jati, N. Gupta, A. Chattopadhyay, and S. K. Sanadhya, “A configurable CRYSTALS-Kyber hardware implementation with side-channel protection,” *ACM Trans. Embed. Comput. Syst.*, 2024, doi: 10.1145/3587037.
- [7] U. Banerjee, T. S. Ukyab, and A. P. Chandrakasan, “Sapphire: A configurable crypto-processor for post-quantum lattice-based protocols,” *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2019, no. 4, pp. 17–61, Aug. 2019, doi: 10.13154/tches.v2019.i4.17-61.
- [8] C. Zhang *et al.*, “Towards efficient hardware implementation of NTT for Kyber on FPGAs,” in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2021, pp. 1–5.
- [9] Z. Chen, Y. Ma, T. Chen, J. Lin, and J. Jing, “High-performance area-efficient polynomial ring processor for CRYSTALS-Kyber on FPGAs,” *Integration*, vol. 78, pp. 25–35, May 2021.
- [10] M. Bisheh-Niasar, R. Azarderakhsh, and M. Mozaffari-Kermani, “Instruction-set accelerated implementation of CRYSTALS-Kyber,” *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 11, pp. 4648–4659, Nov. 2021, doi: 10.1109/TCSI.2021.3106639.
- [11] S. He, H. Li, F. Li, and R. Ma, “A lightweight hardware implementation of CRYSTALS-Kyber,” *J. Inf. Intell.*, vol. 2, no. 2, pp. 167–176, Mar. 2024, doi: 10.1016/j.jiixd.2024.02.004.
- [12] V. T. Nguyen, P. H. Pham, V. T. D. Le, H. L. Pham, T. H. Vu, and T. D. Tran, “AES-RV: Hardware-efficient RISC-V accelerator with low-latency AES instruction extension for IoT security,” *IEICE Electron. Express*, vol. 22, no. 16, Aug. 2025.
- [13] T. Gomes, P. Sousa, M. Silva, M. Ekpanyapong, and S. Pinto, “FAC-V: An FPGA-based AES coprocessor for RISC-V,” *J. Low Power Electron. Appl.*, vol. 12, no. 4, Art. no. 50, Sep. 2022, doi: 10.3390/jlpea12040050.
- [14] P. W. Shor, “Algorithms for quantum computation: Discrete logarithms and factoring,” in *Proc. 35th Annu. Symp. Found. Comput. Sci. (FOCS)*, 1994, pp. 124–134.
- [15] L. K. Grover, “A fast quantum mechanical algorithm for database search,” in *Proc. 28th Annu. ACM Symp. Theory Comput. (STOC)*, 1996, pp. 212–219.
- [16] ARM Limited, “AMBA AXI and ACE protocol specification (AXI4-Lite),” ARM IHI 0022.
- [17] AMD/Xilinx, “Zynq UltraScale+ MPSoC technical reference manual,” UG1085.
- [18] RISC-V International (formerly RISC-V Foundation), “The RISC-V instruction set manual, Volume I: Unprivileged ISA.”
- [19] National Institute of Standards and Technology, “SHA-3 standard: Permutation-based hash and extendable-output functions,” FIPS PUB 202, Aug. 2015.