CARiMoL: A Configurable Hardware Accelerator for Ring and Module Lattice-Based Post-Quantum Cryptography

Afifa Ishtiaq
Department of Electrical Engineering
NUST, SEECS
aishtiaq.msee18seecs@seecs.edu.pk

Muhammad Shafique
Department of Electrical and Computer Engineering
New York University, Abu Dhabi
ms12713@nyu.edu

Osman Hassan
Department of Electrical Engineering
NUST,SEECS
osman.hasan@seecs.edu.pk

***Abstract*—CARiMoL is a novel run-time Configurable Hardware Accelerator for Ring and Module Lattice-based post-quantum cryptography. It's flexible design can be configured to key-pair generation, encapsulation, and decapsulation for NewHope and CRYSTALS-Kyber schemes using same hardware. CARiMoL offers run-time configurability for multiple security levels of NewHope and CRYSTALS-Kyber schemes, supporting both Chosen-Plaintext Attack (CPA) and Chosen-Ciphertext Attack (CCA) secure implementations. To the best of our knowledge, it is the first systematically designed full scale hardware accelerator for CCA-complaint multiple LBC schemes that supports run-time reconfigurability without the use of processor such as ARM Cortex series or soft core such as popular RISC-V processors. CARiMol performs logic sequencing on run-time and eliminates the cycle overhead associated with fetch and decode instructions. For the simultaneous use of Ring-LWE and Module-LWE, CARiMoL's single hardware accelerator has 7× less area overhead as compared to combined standalone design of these schemes. CARiMoL exploits parallelism and extensive resource sharing among the different LBC schemes to achieve high performance and efficiency. Despite its reconfigurability, CARiMoL offers substantial speedup compared to the state-of-the-art, i.e., 9× over NewHope-1024, 10× over NewHope-512, 17× over CRYSTALS-Kyber-1024, and 18× over CRYSTALS-Kyber-512.**

***Index Terms*—Lattice-Based Cryptography, LBC, Key Encapsulation Mechanisms, KEM, Learning-With-Errors, LWE, Module-LWE, Ring-LWE, CRYSTALS-Kyber, NewHope, Post-Quantum Cryptography, Hardware, Accelerator, Area, Performance, Efficiency, Reconfiguration, Reuse.**

I. Introduction

Due to the advances in the field of quantum computing, by virtue of Shor's algorithm [1] standardized public key cryptography primitives, including RSA and Digital Signature Algorithm (DSA), are becoming vulnerable to attacks. To overcome these challenges, the National Institute of Standards and Technology (NIST) initiated a Post-Quantum Cryptography (PQC) competition in 2016 to standardize the quantum resilient cryptosystems [2]. However, most of the proposed algorithms through this effort provide security against either the Chosen-Plaintext Attacks (CPA) or Chosen-Ciphertext Attacks (CCA) secure. On the other hand, Lattice-Based Cryptography (LBC) [2] has emerged as one of the most prominent candidate accounting for 3 out of 4 Public Key Encryption (PKE)/Key Encapsulation Mechanism (KEM) schemes, and 2 out of 3 for signature schemes in the round 3 finalists. These LBC implementations are known for their efficiency, primarily due to their inherent linear algebra based matrix/vector operations.

A number of hard mathematical problems are used to develop LBC schemes. Among them, the most commonly used one is the Learning-With-Errors (LWE) problem. The LWE or standard lattice based schemes employ matrix/vector multiplications that are costly in terms of both space and computational complexity but are considered relatively more secure. Ring-LWE or ideal lattice-based scheme alternatively employ vectors in ring lattices and hence require a polynomial multiplication instead of matrix-vector multiplication, and are thus more efficient but relatively less secure. Module lattices-based schemes (Module-LWE) are another variant of standard lattices, introduced as a trade-off between the security and efficiency provided by LWE and Ring-LWE, respectively. Several recently reported attacks exploit the algebraic structure of lattices, where they state that the dimension of the module makes a big difference [3]–[5]. Therefore, these attacks are less effective against Module-LWE as compared to the comparable Ring-LWE constructions.

Targeted Research Problem: To overcome the evolving security demands, *a run-time configurable yet compact and unified hardware accelerator for PQC algorithms is required that can switch between Ring-LWE/Module-LWE schemes and their different security levels with a very low configuration overhead, while providing high performance efficiency at a low area cost.* The targeted research problem is important because:

1) A run-time configurable hardware design allows to change the mode of operation on the fly and saves the need for different distinct hardware accelerators to accommodate different security requirements or application use-cases, at minimal switching overhead.
2) Multiple functionalities and algorithms on the unified hardware space save a considerable area.
3) Full-scale hardware accelerator enables execution time efficiency by accelerating computations as compared to software implementations.

A. State-of-the-Art and their Limitations

Several works have been reported on LBC accelerators (targeting both FPGAs and ASIC platforms). They perform public key exchange and authentication [6]–[11]. Prominent works are summarized below:

1) **ASICs with little/no Flexibility:** Most of the reported LBC hardware implementations target one type (or one scheme) of lattice structure problem. They lack configurability to support multiple lattice structures, i.e., LWE/Module-LWE/Ring-LWE with little or no support for different security levels [6], [8], [9].
2) **Hardware/Software Co-Design:** Another implementation approach couples a RISC/ARM processor with an FPGA fabric to accelerate the compute-intensive modules. However, this class supports only certain variants of LWE/Ring-LWE based schemes [10], [11] and lacks configurability to Module-LWE scheme.
3) **Application Specific Instruction set Processors (ASIP):** A recent work [7] extends a RISC-V core with application-specific instruction set extensions to support different Module-LWE/Ring-LWE operations and their CCA/CPA-secure implementations. However, these are not run-time configurable rather they are programmable through instruction sets, and therefore, limits the design to fully exploit parallelism and maximally share the hardware resources within and across different processing modules. Also, instruction-based techniques have their limits on maximum achievable performance, as they waste quite some during the fetch, decode and other pipeline stage as compared to a full-scale hardware design.

Key Scientific Challenges: The state-of-the-art has not addressed a full-scale hardware accelerator that is run-time configurable to support Module-LWE along with Ring-LWE/LWE and a CCA-secure key exchange mechanism. To bridge this gap, the following challenges need to be addressed:

1) Identify and derive the relation between different PQC modes to support a whole key exchange mechanism using the same hardware.
2) Identify the structural similarities of Ring-LWE and Module-LWE and leveraged them to design a unified hardware accelerator.
3) Exploiting the relation between implementations of different security strengths to support multi-layer security.
4) Designing a full-scale hardware accelerator for run-time configurability with low configurability overhead.
5) Converting CPA-secure design to CCA-secure KEM without the need for additional hardware.

B. Motivational Case Study

We present a case study to highlight the resource saving potential for designing CARiMoL due to reuse of common hardware between NewHope [12] and CRYSTALS-Kyber [13] LBC schemes. Also, CARiMoL motivates that use of single hardware for key-pair generation, key encapsulation, and key decapsulation will lead to considerable decrease in area overhead as compared to the sum of areas of their standalone implementations. We started with a baseline implementation of the NewHope and CRYSTALS-Kyber operations. This is comprised of individual hardware modules for key-pair generation, key encapsulation, and key decapsulation. CARiMoL is then optimized by exploiting extensive hardware sharing of common blocks between the two LBC schemes and by reuse of the blocks between their respective mode of operation (key-pair generation, key encapsulation, and key decapsulation). As Fig. 2 shows, considerable FPGA resource savings in CARiMoL can be achieved by such a fusion of hardware modules. Comparison with HLS baseline implementation [8] is also provided. NewHope and CRYSTALS-Kyber have similarities in their arithmetic and modular reduction blocks, that can be combined together in the same hardware, leading to a considerable decrease in resource utilization of CARiMoL, besides the configurability. Moreover, it also reduces the area overhead for the simultaneous support of standalone implementations of key-pair generation, key encapsulation, and key decapsulation. Our proposed design uses only one Hash core configurable to different parameters as compared to baseline implementations which have an additional Hash core for CCA support. Thus, it justifies why our utilization is less as compared to CRYSTALS-Kyber KeyEncaps. with maximum utilization amongst individual implementations.

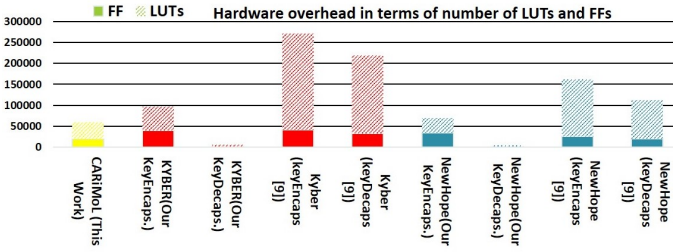


Fig. 1: Hardware overhead in terms of number of LUTs and FFs for our designed baseline, HLS baseline and CARiMoL (yellow) implementation of NewHope (blue) and Kyber (red). FFs are shown in solid colors and LUTs as hashed patterns.

C. Novel Contributions and Concept Overview

To address the above-mentioned scientific challenges, we make the following key contributions (see an overview in Fig. 1).

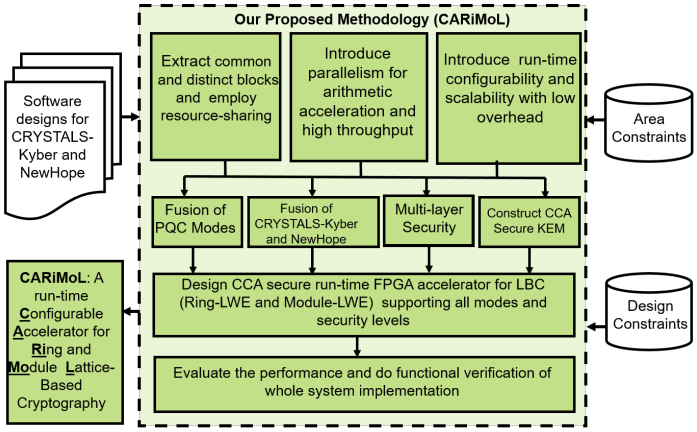


Fig. 2: The overview of our CARiMoL's contributions

1) CARiMoL is a reconfigurable, resource-shared hardware accelerator designed for two major lattice-based PQC

schemes, i.e., CRYSTALS-Kyber and NewHope KEM. This is achieved by exploiting the structural and arithmetic similarities within the two LBC schemes through efficient hardware reuse.

2) The key-pair generation, encapsulation, and decapsulation in key exchange algorithms share several modules and are not meant to run simultaneously. To maximally utilize hardware resources, we undertake the idea of heavily exploiting the reusability of major blocks within these PQC modes to achieve area saving.
3) CARiMoL offers a security-performance trade-off by dynamically switching between any of the four different user-specified security configurations, i.e., CRYSTALS-Kyber 512/1024 and NewHope 512/1024 at run time to facilitate varying application scenarios and security requirements without the need for separate hardware accelerators for different security level.
4) CARiMoL, for CRYSTALS-Kyber employs an optimized memory bank layout to enhance parallelism during compute intensive operations i.e., Number Theoretic Transform, matrix/vector multiplications, and addition.
5) CARiMoL allows to support an authenticated-key-exchange scheme, i.e., CCA-Secure KEM by applying Fujisaki–Okamoto transformation on the CPA-secure public key encryption scheme.

CARiMoL performs logic sequencing using control unit, and require small overhead of few cycles to support run-time reconfigurability. A thorough verification of CARiMoL is carried out against the 100 Known-Answer-Tests (KATs) for each KEM algorithm and their respective security levels. A benchmarking of its post place-and-route implementation for the Xilinx ZYNQ-ZCU11 FPGA is carried out against state-of-the-art, which demonstrates 4× to 18× improved performance for our proposed design.

II. CARiMoL's Design Methodology

The top-level architecture of CARiMoL is shown in Fig. 3 and detailed flow of steps in Fig. 4. The starting point is the C implementations of CRYSTALS-Kyber [13] and NewHope [12] that are submitted to the NIST. For all the operations of these two schemes, first step is to develop the baseline verilog HDL implementations, capable of standalone execution. However, there are many modules that are common not only in the operations of a certain PQC scheme, but also in the operations of multiple different PQC schemes. Consequently, a fusion of PQC operations is carried out, as discussed in Section II-A. To support configurability for the NewHope and CRYSTALS-Kyber schemes, their structural similarities at modular and arithmetic level are analyzed while investigating their area-performance trade-offs (Section II-B). The obtained hardware accelerator is composed of two parts: (1) A Control Unit, and (2) A Datapath Unit.

The control unit determines the operational sequencing, linkage and parameterization of the constituent modules in the datapath as per the configuration word (Fig. 5). The configuration word specifies two fields: (1) A 2-bit *Mode* determines the operation that the accelerator is configured to perform, i.e., key-pair generator, encapsulator, or decapsulator. (2) A 5-bit *Opcode* specify the selected algorithm (NewHope or CRYSTALS-Kyber) and the security level.

The datapath contains all the critical blocks configurable to support any Mode and Opcode for KEM (Section II-A). On-chip memories (i.e., BRAMs in case of FPGA-based designs) store all intermediate data from various processing blocks, as well as a public key, secret key, plaintext, and ciphertext. The processing blocks (Fig. 3) have access to the memory banks via the Memory Controller. All the processing blocks are designed to have appropriate interfaces with read ports (rd_en, rd_addr, rd_data) and write ports (wr_en, wr_addr, wr_data) to ease the interaction with the Memory Controller. The Memory Controller controls the reading and writing of every polynomial in the BRAM based on logic sequencing. The amount of intermediate data produced by modules during the KEM process increases dramatically as the security level increases. The Verify module performs the xor operation on the shared secrets generated by encapsulation (SS) and decapsulation (SS') to check if the output is decoded correctly or not and raise the pass/fail flag.

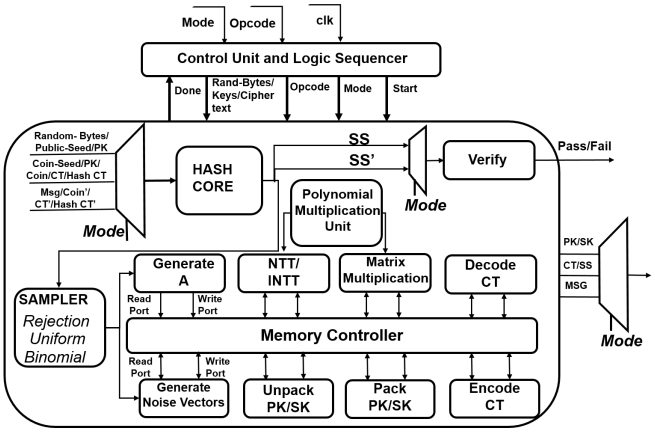


Fig. 3: A top-level architectural view of CARiMoL

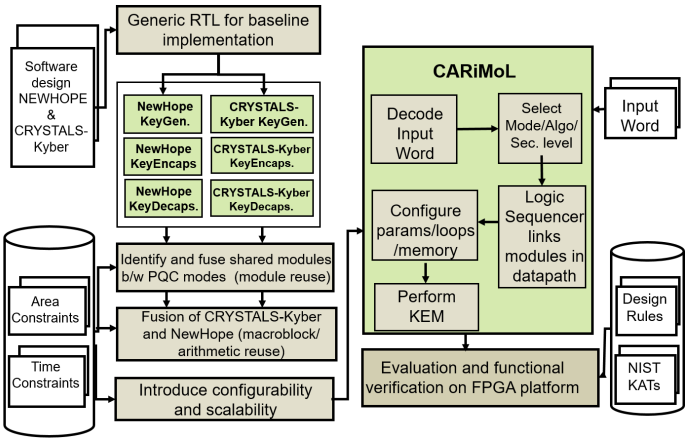


Fig. 4: Our Design methodology (gray boxes show new design steps and our accelerators are shown in green boxes)

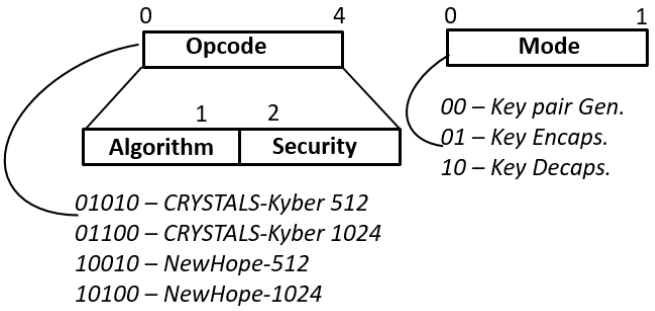


Fig. 5: Input Word configuration details

A. The Fusion of PQC Modes

The three PQC modes in the key exchange mechanism share several modules as indicated by numbered states (Fig. 6). To accommodate these PQC modes in the same hardware design, and to save additional hardware, we reuse these modules. Their linkage is done by control unit which performs logic sequencing based on input Mode, which can be set to 0, 1, 2 specifying that the hardware accelerator operates as a key-pair generator, encapsulation, and decapsulation, (represented in Fig. 6 as solid blue, red and yellow lines), respectively. Whereas, dashed yellow line represents re-encapsulation.

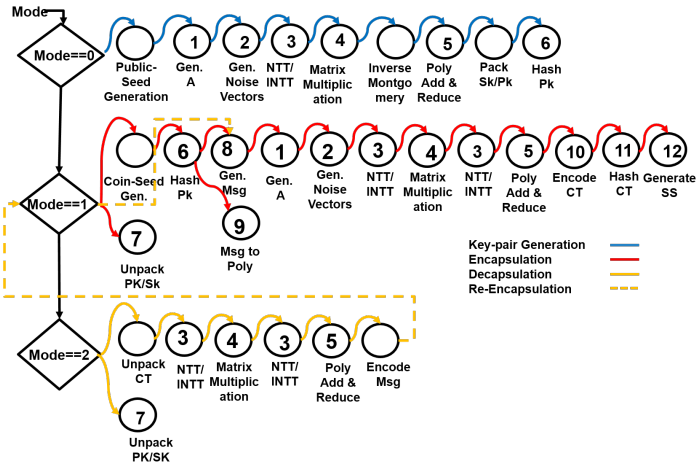


Fig. 6: Logic sequencing for various modes in CARiMoL

B. The Fusion of CRYSTALS-Kyber and NewHope

1) Hash Core with Sampler: In both algorithms, several hash functions including SHAKE-128/256 and SHA3-256/512, used as either pseudorandom number generator or as Extendable Output Functions, are needed. Hash controller (Fig.7) configures the hash parameters depending upon the Mode and Opcode (Fig.3). The base of SHA-3 and its variants is the Keccak algorithm, which needs 24 rounds of permutation operations with different round constants. Pseudo-random bits, generated by the Hash core, are stored in the 1600-bit Keccak state register. The data produced by the permutation stage is either fed as it is to generate seeds, hash outputs of keys or ciphertexts, and shared secrets or are used to perform one of the following three types of operations – rejection sampling in $[0,q)$, binomial sampling with standard deviation σ, and uniform sampling in $[-\eta, \eta]$ for $\eta < q$. Our design supports new version of NewHope appeared as a response of the work presented in [14], which exploits the incorrect oracle cloning of some NIST KEM PQC candidates to perform key-recovery attacks. The new version of NewHope includes a domain separation for the SHAKE calls in order to make each hash call independent. That is, all hash calls with the same input size use a domain separator label (e.g., a nonce). Our Hash core also supports absorption of the input message and squeezing of output dependent on input shake rates, number of blocks and message length.

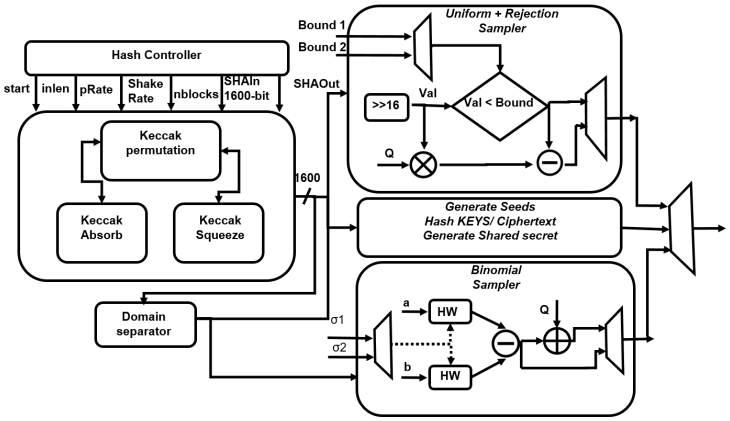


Fig. 7: Unified Hash core with the sampler

2) Memory Layout and Access Optimization: The polynomials A, S, and E are stored in a standard way for NewHope but an optimized memory layout to enhance parallelism is adopted to store these polynomials in case of CRYSTALS-Kyber. True dual port BRAMS are used. In Fig. 8, we compare the standard memory layout to our memory layout after the sampling stage. This layout enables extensive parallelism to process polynomials with multiple vectors undergoing the same operation simultaneously. Consequently, the computation cycles for NTT and matrix multiplication are reduced by half. For instance, during the key-pair generation in CRYSTALS-Kyber-1024, NTT of the polynomials S and E is computed, while both S (0, 1, 2, 3) and E (0, 1, 2, 3) comprises of 4 vectors. Our memory layout for S and E polynomials, as shown in Fig. 8 (on the upper right side) indicates that S0, S1, and E0, E1 memories are processed in parallel, reducing the NTT cycle computation time to half. Additionally, this layout also makes the computation of $AS + E$ more efficient.

3) Unified Polynomial Multiplier: In both NewHope and CRYSTALS-Kyber algorithm, arithmetic operations over polynomial vector include forward and inverse-NTT, point-wise multiplication, and polynomial vector addition, etc. These operations are the most expensive parts in terms of area and time. A representative formula is A.S + E, which consists of all the typical polynomial vector operations. The main component used by NewHope and CRYSTALS-Kyber in the computation of NTT, barret reduction, and matrix/vector multiplication is

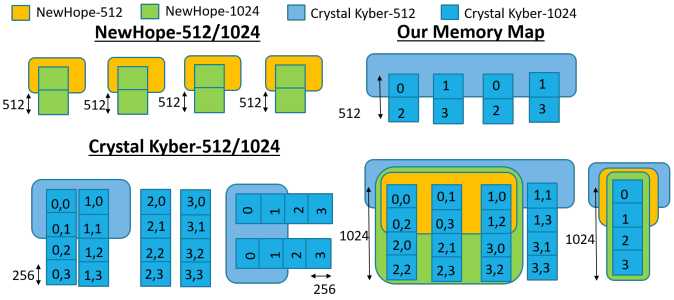


Fig. 8: Optimized memory layout accelerating polynomial multiplication (NTT/INTT and matrix/vector multiplication)

the montgomery reduction. All these operations are fused into one unified multiplier hardware (Fig. 9). It comprises of modular and point-wise multiplications. Multipliers 2 and

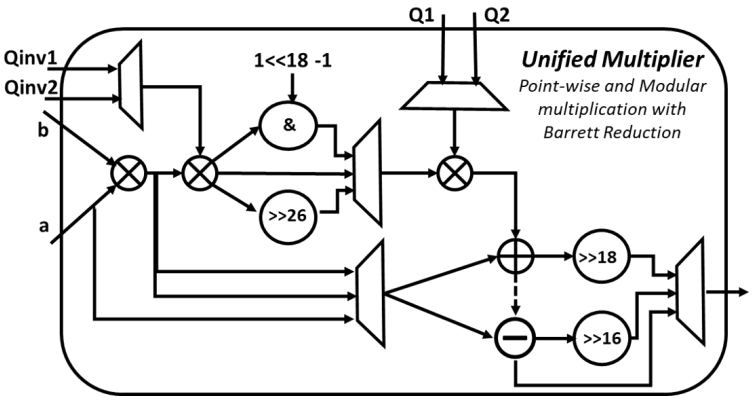


Fig. 9: Our unified multiplier block (fqmul)

3 in Fig. 9 do modular multiplication and are implemented using shifts so this unit consumes only 1 multiplier block. The architecture (Fig. 10) can be configured on the basis of Mode and Opcode to support Cooley-Tukey NTT (NewHope) and the Gentleman-Sande NTT with early abort and special base case multiplication (Kyber) with *fqmul* as the bottom module. This structure process two vectors in parallel for NTT/INTT, barrett reduction, and matrix/vector multiplication.

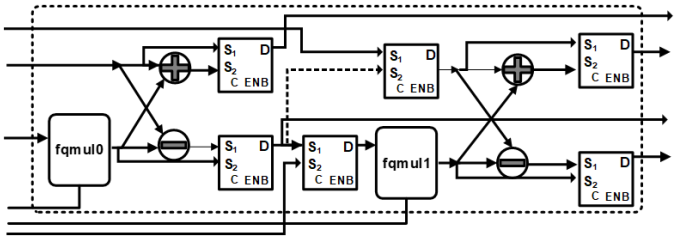


Fig. 10: The unified polynomial multiplier for NTT/INTT, Barrett reduction, and matrix/vector multiplication.

C. Multi-Layer Security

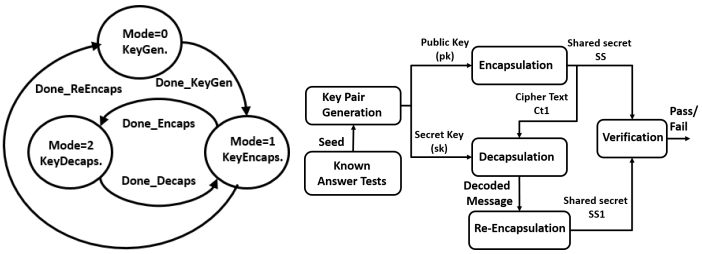
To support different NIST-compliant levels of security for CRYSTALS-Kyber and NewHope, the configuration and parameterization of the counters dealing with internal loops and memory depths is adjusted. Based on the mode of operation and the security level specified (Opcode [2:0]), the loop counters are accordingly configured (see details in Table I).

TABLE I: Scalable, configurable loops for multi-layer security

CRYSTALS-Kyber Standard	CRYSTALS-Kyber This Work	NewHope Standard	NewHope This Work
Generate A **Loop 1&2 :** **0 to KYBER_K** • KYBER_K= 2 for level 1 • KYBER_K= 4 for level 5	**Generate A** **Loop 1&2 :** **0 to Opcode[2:0]** • Opcode[2:0] = 2 for level 1 • Opcode[2:0] = 4 for level 5	**Generate A** **Loop 1&2 :** **0 to NEWHOPE_N/64** • NEWHOPE_N= 512 for level 1 • NEWHOPE_N= 1024 for level 5	**Generate A** **Loop 1&2 :** **0 to Opcode[2:0]<<2** • Opcode[2:0] = 2 for level 1 • Opcode[2:0] = 4 for level 5
Generate Noise Vectors **Loop1 : 0 to 32** **Loop2 : 0 to 8** **KeyGen** • Nonce: 0 to 3 for level1 • Nonce: 0 to 5 for level5 **KeyEncaps.** • Nonce: 0 to 4 for level1 • Nonce: 0 to 6 for level5	**Generate Noise Vectors** **Loop1 : 0 to 32** **Loop2 : 0 to 8** **Nonce:Opcode[2:0]/2 - 1 + Mode** • Mode = 0 for KeyGen • Mode = 1 for KeyEncaps	**Generate Noise Vectors** **Loop 1&2 :** **0 to NEWHOPE_N/64** **KeyGen** • Nonce: 0 to 1 for level1 • Nonce: 0 to 2 for level5 **KeyEncaps.** • Nonce: 0 to 2	**Generate Noise Vectors** **Loop 1&2 :** **0 to NEWHOPE_N/64** **Nonce: Opcode[2:0]/2 - 1 + Mode** • Mode = 0 for KeyGen • Mode = 1 for KeyEncaps

D. Conversion of CPA to CCA-Secure KEM

The fusion and reconfiguration of PQC modes on the same hardware benefits the conversion of a CPA-secure PKE to a CCA-secure KEM. An already available Hash core is reconfigured for Fujisaki–Okamoto transformation. Similarly, decapsulation involves re-encapsulation but only requires reconfiguration to encapsulation mode (see details in Fig. 11a).



(a) PQC mode configuration state machine (b) Functional verification setup

Fig. 11: CARiMoL configuration and verification for CCA key exchange mechanism

III. Experimental Results and Comparisons

We implemented and evaluated our design using Vivado 2019.1 on Xilinx Zynq-ZCU111 FPGA board at an operating frequency of 200MHz with the speed grade -2. We used NIST provided KATs as input test vectors in simulation for the functional verification of key exchange mechanism for both algorithms (Fig.12a & 12b). In Table II, we compare CARiMoL with existing hardware-accelerated implementations of NIST Round 2 lattice-based schemes. The last column of Table II gives a speedup that CARiMoL offers for that protocol when compared with that implementation. From the comparison results provided in Table II, we make the following key observations:

1) A fair comparison of CARiMoL with stand-alone dedicated implementations has some inherent difficulties. These dedicated cryptographic accelerators are designed

TABLE II: Comparison of CARiMoL (post place-and-route results) with state-of-the-art LBC hardware implementations

Algorithm	Platform	Freq. (MHz)	Protocol	FFs/LUTs	Cycles	Thr.(MB/s)	Speedup
Chosen-Ciphertext Attakcs (CCA) secure Implementations							
CARiMoL	FPGA (Zynq-ZCU111)	200	NewHope-512-CCA-KEM-Encaps	20,508 /39,130	34,500	6.3	-
			NewHope-1024-CCA-KEM-Encaps		69,000	6.24	-
			Kyber-512-CCA-KEM-Encaps		19,891	7.4	-
			Kyber-1024-CCA-KEM-Encaps		36,465	8.6	-
Banerjee et al. [7]	ASIC (40nm)	72	NewHope-512-CCA-KEM-Encaps	-	136,077	0.575	10.95x
			NewHope-1024-CCA-KEM-Encaps		236,812	0.661	9.44x
			Kyber-512-CCA-KEM-Encaps		131,698	0.40	18.5x
			Kyber-1024-CCA-KEM-Encaps		223,469	0.50	17.2x
Basu et al.[8]	FPGA-HLS (Artix-7)	66.66	NewHope-512-CCA-KEM-Encaps	26,257/135,689	136,077	0.53	11.88x
			Kyber-512-CCA-KEM-Encaps	40,720/230,540	56,345	0.87	8.5x
Chosen-Plaintext Attakcs (CPA) secure Implementations							
Order et al.[9]	FPGA (Artix-7)	117	NewHope-1024-Simple-Encrypt	4,635/4,498	179,292	1.4	4.45x
Fritzmann et al. [10]	FPGA (Zynq-7000)	-	NewHope-1024-CPA-PKE-Encrypt	7,303/26,606	589,285	-	-

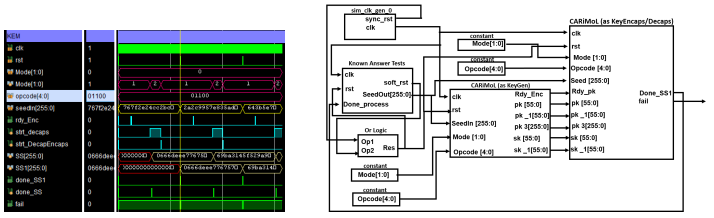
with the goal of achieving high performance and often have little or no flexibility.

2) High-level synthesis (HLS) generated designs in [8] map large arrays to distributed RAMs, increasing the LUTs utilization and lack reuse of the modules. Our design consumes 3× to 4× less LUTs than the HLS designs by efficiently using BRAMs and hardware reuse. Sequential execution of functions increases the latency of the HLS designs, while our design exploits an optimized memory layout achieving parallelism as well as reduction in the processing latency resulting in 8× to 11× speedup.
3) The proposed CRYSTALS-Kyber and NewHope implementations are 9× to 18× faster than [7] that couples RISC-V with the crypto-core. Our design has its own logic sequencer and allows parallel processing of multiple polynomials, whereas, the sequential execution of instructions in [7] limits the design to fully exploit parallelism. These designs also have a lower operating frequency due to a longer critical path.
4) [9] is a CPA only implementation, targeting low-cost FPGAs with server only and client only implementations each taking ≈5K LUTs and ≈170K cycles. CARiMoL's higher consumption is justified since it supports both server and client as well as both security levels of NewHope as well as CRYSTALS-Kyber. Considering throughput, CARiMoL being CCA-secure, provides 4× speedup along with support to both security levels as it involves extensive parallelism.
5) Notably, the design of [10] uses a RISC-V processor with NTT and SHA accelerators to implement the NewHope protocol, but our design is a full-scale hardware accelerator and thus takes 8x less cycles.

CARiMoL offers a variety of performance optimizations in hardware implementations, the amount of total clock cycles for both encryption and decryption of the proposed design reduces significantly. Moreover, our design accommodates key pair generation, encapsulation and decapsulation for both NewHope and CRYSTALS-Kyber on a single hardware by exploitation of structural and arithmetic similarities. Parallel execution and pipelining in hardware, design techniques, such as the integration of arithmetic functions in a single module, and reusing most of the functional modules during different modes and algorithms, use of logic sequencer instead of programmed instructions lead to the achievement of 4× to 18× performance improvement.

IV. Conclusion

This work presents CARiMoL, a fully-functional and configurable hardware accelerator for Ring and Module Lattice-based post-quantum cryptography. It offers all operations and security levels of NewHope and CRYSTALS-Kyber lattice based schemes, that are part of NIST PQC round 2 and 3. CARiMoL supports run-time configurability to switch to any flavor of these two prominent LBC schemes, i.e., any security level and any operation (Key pair generation/encapsulation/decapsulation), CCA/CPA secure modes, as per the application requirements. In future, we plan to extend the flexibility/re-configurability of the proposed design to support other prominent lattice based schemes including Saber.



(a) CRYSTALS-Kyber-1024 Verification (b) Simulation Setup

Fig. 12: Experimental Setup

References

[1] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. 35th Annual Symposium on Foundations of Computer Science*, 1994, pp. 124–134.

[2] D. Moody, "Post-quantum cryptography: Nist's plan for the future," in *Talk given at PQCrypto 16 Conference*, Fukuoka, Japan, February 2016. [Online]. Available: https://pqcrypto2016.jp/data/pqc2016_nist_announcement.pdf

[3] P. Campbell, M. Groves, and D. Shepherd, "Soliloquy: A cautionary tale," *In ETSI 2nd Quantum-Safe Crypto Workshop*, pp. 1–9, 2014.

[4] R. Cramer, L. Ducas, C. Peikert, and O. Regev, "Recovering short generators of principal ideals in cyclotomic rings," *EUROCRYPT 2016*.

[5] R. Cramer, L. Ducas, and B. Wesolowski, "Short stickelberger class relations and application to ideal-SVP," *Advances in Cryptology– EUROCRYPT 2017*.

[6] P.-C. Kuo, W.-D. Li, Y.-W. Chen, Y.-C. Hsu, B.-Y. Peng, C.-M. Cheng, and B.-Y. Yang2a, "High performance post-quantum key exchange on FPGAs," *Cryptology ePrint Archive, Report 2017/690, 2017*.

[7] U. Banerjee, T. S. Ukyab, and A. P.Chandrakasan, "Sapphire: A configurable crypto-processor for post-quantum lattice-based protocols," *IACR Transactions on CHES*.

[8] D. S. K. Basu, M. Nabeel, and R. Karri, "In NIST post-quantum cryptography- a hardware evaluation study." *Cryptology ePrint Archive*.

[9] T. Oder and T. Guneysu, "Implementing the NewHope-Simple Key Exchange on low-cost FPGAs," *Cryptology ePrint Archive, Report 2017/690, 2017*.

[10] T. Fritzmann, U. Sharif, D. Müller-Gritschneder, C. Reinbrecht, U. Schlichtmann, and J. Sepulveda, "Towards reliable and secure post-quantum coprocessors based on RISC-V," *In 2019 DATE*.

[11] A. Jati, N. Gupta, A. Chattopadhyay, and S. K. Sanadhya, "Side-channel protected post-quantum cryptoprocessor," *Cryptology ePrint Archive: Report 2019/765*.

[12] T. Poppelmann, E. Alkim, R. Avanzi, J. Bos, L. Ducas, A. de la Piedra, P. Schwabe, D. Stebila, M. R. Albrecht, E. Orsini, V. Osheter, K. G.Paterson, G.Peer, and N. P. Smart, "Newhope – algorithm specifications and supporting documentation," *Technical report, NIST, 2019*.

[13] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanckk, P. Schwabe, G. Seile, and D. Stehl, "CRYSTALS – Kyber: a CCA-secure module-lattice-based KEM," *Technical report, NIST, 2019*.

[14] H. D. M. Bellare and F. Günther, "Oracle cloning and read-only indifferentiability," *Cryptology ePrint Archive*.