Area-Efficient Matrix-Vector Polynomial Multiplication Architecture for ML-KEM Using Interleaving and Folding Transformation

Weihang Tan*, Yingjie Lao†, and Keshab K. Parhi*
*Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455, USA
†Department of Electrical and Computer Engineering, Tufts University, Medford, MA 02155, USA
*{wtan, parhi}@umn.edu, †yingjie.lao@tufts.edu

***Abstract*—The ML-KEM post-quantum cryptography (PQC) scheme requires matrix-vector polynomial multiplication and polynomial arithmetic operations in the number theoretic transform (NTT) domain. Prior optimization approach KyberMat leverages the transposed-form fast filtering structure and substructure sharing technique, reducing the computational complexity. In this paper, a novel and area-efficient design builds upon the KyberMat framework, using the hierarchical interleaved folding algorithm to reduce hardware resources. Two design strategies are utilized in the proposed design. The proposed design initially scales down the NTT/inverse NTT processors via folding transformation, while utilizing a fixed number of DSPs and LUTs across different security levels of ML-KEM. This work further introduces a recursive summing unit along with the interleaving method to ensure continuous data processing and ultimately improve hardware utilization and throughput. The experimental result shows that our proposed area-efficient design achieves an average reduction of 71.55% in DSPs and 63.89% in LUTs among three different security levels, compared to the KyberMat framework.**

***Index Terms*—Post-quantum Cryptography, ML-KEM, Lattice-based Cryptography, Matrix-Vector Polynomial Multiplication, Area-Efficient Architecture**

I. Introduction

The Module-Lattice-based Key-Encapsulation Mechanism Standard (ML-KEM) [1] is a post-quantum cryptography (PQC) scheme that has been recently approved and recommended by the Federal Information Processing Standard (FIPS) and the National Institute of Standards and Technology (NIST). This scheme is believed to be secure under attack from the quantum computer, which is derived from the CRYSTALS-Kyber (Kyber) scheme [2].

The full implementation of the ML-KEM scheme depends on integrating several fundamental algorithms: cryptographic functions, conversion/compression algorithms, and arithmetic with polynomials. Among these algorithms, the arithmetic with polynomials emerges as a crucial computational bottleneck and presents a significant design challenge, as depicted in Fig. 1. The fundamental of the ML-KEM scheme is based on module-learning with errors (M-LWE) problem [3]. Since the MLWE problem is a generation of learning with errors (LWE) problem [4], the computations are based on the matrix and vector, where the entries of the matrix and vector are the polynomial over the ring. Notably, the polynomial modular multiplication is more computationally expensive than other polynomial arithmetic, which is optimized by the number-theoretic transform (NTT) [1]. Moreover, since the special parameter setting utilized in the ML-KEM requires polyphase decomposition before performing the NTT-based polynomial multiplication, the NTT-based polynomial multiplication becomes more complicated than other LWE-based cryptosystems. Thus, optimizing the NTT-based polynomial multiplication using fast polyphase decomposition [5], [6] is of great interest in the existing works [7]–[10].

Existing literature primarily focuses on either low-latency or compact designs, each targeting to different application requirements. In contrast to these prior endeavors, this present paper introduces an innovative area-efficient design that is tailored to maximize area efficiency while at the expense of latency. Such area-efficient design is a balanced solution in that neither clock cycles nor hardware resources are overly prioritized, as delineated in Fig. 1. Specifically, the *hierarchical interleaved folding algorithm* [11] combines the interleaving method and folding transformation [6], [12] systematically.

The contributions of this paper are summarized as follows. First, the top-level architecture leverages interleaving and folding transformations, so the proposed design exhibits significant resource consumption reduction across varying security levels compared to the original KyberMat framework [10]. Second, each primary building block integrated into the proposed area-efficient architecture has been meticulously optimized with a custom design approach to ensure continuous result generation, addressing the challenges associated with complex data dependencies in intermediate computations. Finally, a comprehensive FPGA experimental result shows that the design strategy used in the proposed area-efficient design optimally balances the timing and area performances.

The rest of this paper is structured as follows. Section II introduces the preliminary and mathematical background of this paper. Section III describes the proposed novel and area-efficient hardware architecture for ML-KEM. Section IV presents the performance analysis of the proposed architecture. Finally, Section V concludes the paper.

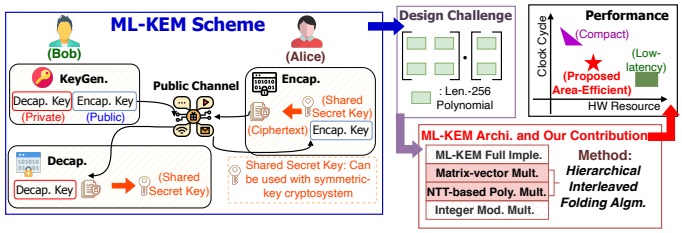

Fig. 1. The overview of ML-KEM scheme and our contributions.

II. Preliminary and Mathematical Background

A. Overview of ML-KEM

The ML-KEM scheme consists of three main algorithms, including the key generation algorithm (KeyGen), encapsulation algorithm (Encap), and decapsulation algorithm (Decap). The mechanism of ML-KEM is illustrated in Fig. 1 (blue box), which distributes the shared key between two parties (i.e., Alice and Bob) to perform the symmetric key cryptographic algorithm [1]. The operational flow of ML-KEM is graphically represented in Fig. 1 (highlighted in the blue box). It executes a shared key distribution between two parties, namely Alice and Bob, so that they can perform symmetric key cryptography algorithms [1].

ML-KEM has three parameter sets representing ML-KEM-512, ML-KEM-768, and ML-KEM-1024, respectively. To modulate the security level within the ML-KEM scheme, adjustments are made to the module dimension k, specifically setting $k = 2$, 3, or 4, while keeping the length-n polynomials over the ring $R_q = \mathbb{Z}_q/(x^n + 1)$ constant. Bold variables are employed to denote the polynomial vector, $\boldsymbol{a} \in R_q^k$, and the polynomial matrix, $\boldsymbol{A} \in R_q^{k\times k}$, wherein individual entries are polynomials. Polynomials over the ring R_q are symbolized as $a(x)$ with the fixed parameters of a polynomial length in $n = 256$ and a coefficient modulus set to a prime integer, $q = 3329$. Additionally, $\boldsymbol{a}^T$ denotes the transpose of the matrix (or vector), and $\hat{\boldsymbol{a}}$ is the NTT representation of the variable.

B. Arithmetic with Polynomials and Matrix-Vector Polynomial Multiplication under NTT representations

Polynomial modular addition and multiplication are two key operations in the arithmetic with polynomials. The polynomial modular multiplication is more expensive, so the ML-KEM scheme maps all the polynomials into the NTT representations and is followed by the polynomial multiplications in the NTT domain.

Given the coefficient modulus is set to $q = 3329$, it only allows the construction of primitive 256-th roots of unity modulo q. This constraint derives every length-256 polynomial to split into two length-128 polynomials through polyphase decomposition, a prerequisite step before either NTT or iNTT computations. Such a requirement intrinsically complicates the polynomial multiplications in the NTT domain, a step comprising the MultiplyNTTs and the BaseCaseMultiply algorithms [1]. Since all the entries in the matrix and vector are polynomials, the matrix-vector polynomial multiplication applied in the Encap and Decap algorithms ultimately becomes one of the notable design challenges within the ML-KEM, as delineated in Fig. 1 (highlighted in the purple box).

The prior work in [10] presents an optimized algorithm, KyberMat, to reduce the overall computational complexity of the matrix-vector polynomial multiplication

$$\boldsymbol{p} = \text{iNTT}(\hat{\boldsymbol{A}}^T \cdot \text{NTT}(\boldsymbol{r})). \tag{1}$$

This optimization is based on the transposed-form fast filtering structure as well as the sub-structure sharing technique. Each entry in vector $\boldsymbol{r}$ has to perform the polyphase decomposition $r_i(x) = r_{i,e}(x^2) + r_{i,o}(x^2) \cdot x$, for $i \in [0, k-1]$, which can be represented as a vector $\boldsymbol{r}_i = [r_{i,e}(x^2), r_{i,o}(x^2)]^T$. A similar process is also executed in matrix $\boldsymbol{A}$ resulting in $\hat{\boldsymbol{a}}_{ij} = [\hat{a}_{ij,e}(x^2), \hat{a}_{ij,o}(x^2)]^T$, $j \in [0, k-1]$.

C. Prior optimizations for Kyber scheme

The Kyber scheme has been previously explored in various hardware architectures, as detailed in [8]–[10], [13], [13]–[17]. These architectures are the fundamental benchmarks for the ML-KEM scheme. Specifically, the designs presented in [8], [13] emphasize their design in terms of compactness. These designs deploy a few processing elements (PEs) to create a reconfigurable architecture for both NTT/iNTT computation and point-wise multiplication. Besides, these compact designs can support all security levels of the Kyber scheme without demanding extra resources while requiring additional clock cycle consumption.

On the other hand, the original KyberMat framework implementation [10] is constructed for low-latency architecture, maintaining consistent clock cycles across the three security levels. However, this design requires a linear increment in hardware resources as the security level scales.

Different from these prior works, the proposed design introduces an innovative, area-efficient design approach, aiming to maintain a balance between timing and hardware resources.

III. Area-Efficient Design for the Matrix-Vector Multiplication

The proposed novel area-efficient design approach, which is illustrated in Fig. 2, is inspired by the *hierarchical interleaved folding algorithm* as presented in [11], and an example of using this algorithm for fast Fourier transform (FFT) is demonstrated in [18]. This algorithm consists of two critical techniques, which are folding transformation and the interleaving method.

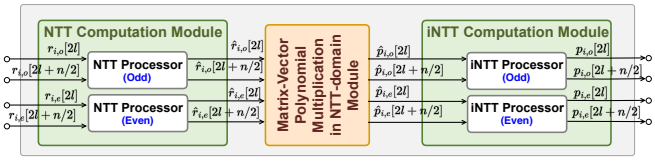

Fig. 2. Top-level architecture of area-efficient design for matrix-vector multiplication when $k = 2$.

The folding transformation facilitates computations in the architecture to be executed in a time-multiplexed way within

a singular computational unit. This results in a reduction in the required number of computational units, ultimately decreasing hardware resource consumption. As highlighted in [10], the NTT and iNTT computation modules significantly influence the total hardware resource consumption, especially concerning the number of modular multipliers employed.

The proposed design adopts the KyberMat framework from [10] and augments it based on an area-efficient design strategy. The first step is to reduce the number of NTT/iNTT processors via folding transformation. The top-level structure of the proposed area-efficient design is depicted in Fig. 2. By leveraging the folding transformation on the NTT computation module, the $2k$ NTT operations inside the KyberMat framework are mapped to merely two processors, specifically to even and odd indexed-term polynomials. Each of these NTT processors supports operations of length-128, employing the radix-2 multi-path delay commutator (R2MDC)-based architecture as presented in recent works [15], [19]–[21]. Similarly, the iNTT computation module uses the same method, resulting in only two iNTT processors. Consequently, the number of data-paths in the design is reduced to four. In contrast to the KyberMat framework, whose hardware resource increased $1.6\times$ when scaling the security level to the next level, the NTT/iNTT computation module in the proposed area-efficient architecture maintains the same architecture from ML-KEM-512 to ML-KEM-1024.

The matrix-vector multiplication in NTT-domain module undergoes the same folding transformation to instantiate the architecture to match four input data-paths. Fig. 3 demonstrates our proposed design for the example when $k = 2$. This proposed module mainly consists of k PEs, where each PE (boxed in blue in Fig. 3) comprises six modular multipliers and modular adders for point-wise multiplication and addition. One of the significant feature in the proposed area-efficient design is that the data-path operates mostly in a feed-forward manner. Each PE is connected to its adjacent PE(s) in a regular pattern. Thus, it allows data to be processed and computed rhythmically in a continuous stream so the input data can broadcast through all the PEs synchronously without any idle status. In particular, it enables $\boldsymbol{f}_i = \{f_{i,0}, f_{i,1}, f_{i,2}\}$ to perform the same entry-entry multiplication across different $\boldsymbol{g}_{ij} = \{g_{ij,0}, g_{ij,1}, g_{ij,2}\}$ on multiple PEs at the same time, achieving a significant speedup over sequential processing used in the compact design.

However, since the data dependence between intermediate results is complex, designing an efficient architecture to maintain high-throughput and full hardware utilization is challenging. For instance, consider the computational process where the product of $\boldsymbol{f}_0$ and $\boldsymbol{g}_{00}$, denoted as β_{00}, must be merged with the product of $\boldsymbol{f}_1$ and $\boldsymbol{g}_{01}$, represented as β_{01}, before performing the iNTT computation.

Due to the architectural constraint that only two NTT processors handle a single $\boldsymbol{r}_i$ at a time, the computation of $\boldsymbol{f}_0$ and $\boldsymbol{f}_1$ cannot be achieved simultaneously without further transformation. Consequently, β_{00} must wait for additional clock cycles for further computation, causing idle cycles in the

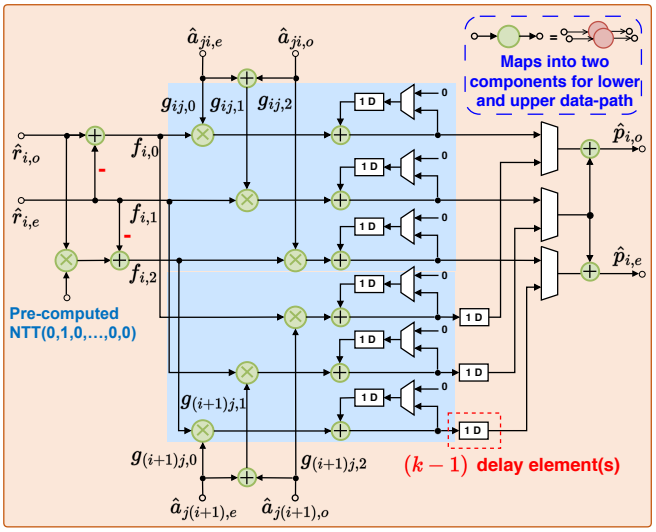

Fig. 3. Area-efficient design for matrix-vector multiplication in NTT-domain module when $k = 2$. One green circle represents two modular multipliers/modular adders for parallel data-paths.

components and the input and output stream to be interrupted and then leading to low hardware utilization and decreased throughput (sampling rate). To address this issue, we introduce the *recursive summing unit* that is integrated in each PE, as depicted in Fig. 4(b). This unit is used to sum up the entry-entry products computed by the modular multipliers, which is the auxiliary but the central component to leverage the interleaving method.

The interleaving method offers a strategic advantage by interleaving γ independent computations simultaneously in the same architecture data-paths in a time-multiplexed form without additional hardware overhead, yielding the advantage of allowing independent computations to be executed in parallel without affecting the throughput of the architecture. In the proposed design, we set the interleaving factor, γ, equal to k, enabling the proposed architecture to concurrently process k independent input data sequences, $\boldsymbol{r}_i$, which allows for generating the final result continuously.

In line with the requirements of the hierarchical interleaved folding algorithm, each delay element is expanded to γ delay elements with an interleaving factor of γ. Meanwhile, the switching rate in the control signal has to be scaled up by a factor of γ. For instance, both the number of original delay elements in NTT/iNTT processors and the control signal's switching rate double when $k = 2$, as visualized in Fig. 2. Consequently, the number of the original delay elements in NTT/iNTT processors and the switching rate for the control signal double when $k = 2$, as shown in Fig. 2. Practically, if the control signal switches every clock cycle, the switching rate will now double, so the control signal flips every two clock cycles when $\gamma = 2$.

Fig. 4 details the timing diagram for matrix-vector multiplication in NTT-domain module, and the mechanism leveraging the hierarchical interleaved folding algorithm and recursive summing unit. This is illustrated by a toy example based on

TABLE I. Performance of the proposed design and prior works for ML-KEM implemented in Artix-7 FPGA

Design	LUTs	FFs	DSPs	Freq.[MHz]	Cycles (μs)
Xing [8]	1737	1167	2	161	3200 (19.84)|5568 (34.58)|8448 (52.47)
Guo [13]	1549	788	4	159	1614 (10.15)|3298 (20.74)|4348 (27.35)
Tan [10]	15842|24954|36776	11110|19076|27840	84|144|216	222	222 (1.00)|223 (1.00)|224 (1.00)
Proposed	**5872|6488|8204**	**10040|14824|20708**	**42|48|54**	**200**	**474 (2.37)|665 (3.33)|856 (4.28)**

$\star|\star|\star$: Results for ML-KEM-512, ML-KEM-768, and ML-KEM-1024 security-level.

the length of polynomial $n' = 4$, interleaving factor $\gamma = 2$.

The timing diagram in Fig. 4(a) presents the mechanism on how two independent input sequences can be processed simultaneously within a single NTT processor, based on interleaving method. Using the provided example, instead of feeding the NTT processor (even) with consecutive samples from a single sequence, alternating samples from two sequences (distinct by blue and red) are fed into it. The advantage of the

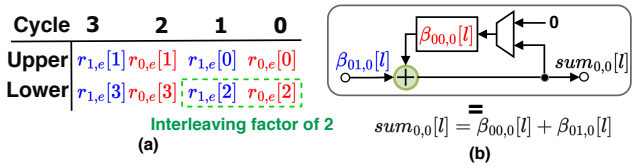

Fig. 4. Explanation for interleaving method and proposed recursive summing unit (a) Timing diagram for the two input sequences via interleaving when $n' = 4$. (b) Recursive summing unit for entry-entry products addition.

interleaving method illustrates the optimization of hardware resource usage and reduction in wait times, as demonstrated in Fig. 4(b). In this example, $r_{1,even}[0]$ is loaded at clock cycle 1 instead of loading $r_{0,even}[1]$. Therefore, after using the recursive summing unit along with the interleaving method, the product $\beta_{00,0}[l]$ is computed first and stored in the upper delay element. The other product, $\beta_{01,0}[l]$, having independent input sequences and computation, is generated in the next clock cycle and passes through the same data-path, which then adds to $\beta_{00,0}[l]$. In general, this continues recursively until $(k-1)$ summations are performed, after which the stored values in the delay element are reset to be zero.

Subsequently, the MUXs and extra delay elements are cascaded to the PEs in matrix-vector multiplication in the NTT-domain module to interleave the sums that pass through the subsequent data-paths. The number of delay elements followed by PEs is increased from zero to $(k-1)$ from top to bottom, as shown in Fig. 3. The MUXs continuously select the results computed from different PEs to the next data-paths and iNTT computation module in a time-multiplexed manner.

This area-efficient architecture consumes 28 modular multipliers for the NTT computation module and iNTT computation module at different security levels. Furthermore, $(6k+2)$ modular multipliers and $(8k+8)$ modular adders are used in the matrix-vector multiplication in the NTT-domain module. The latency of the architecture can be calculated to be $T_{Lat} = k(\frac{n}{4} + \frac{n}{2} - 2) + (k-1) + N_{pipe}$, where N_{pipe} is the additional clock cycles caused by pipelining.

IV. Experimental Result

To provide a fair evaluation and demonstrate our contributions depicted in Fig. 1 (as highlighted in the black box), we select the works from [8], [10], [13] as our reference benchmarks, which contain both compact and low-latency designs. The resultant data is presented in Table I based on the same parameters in different security levels, notably $k = \{2, 3, 4\}$, $n = 256$, and $q = 3329$.

We first compare the proposed area-efficient design to the low-latency implementation in the original KyberMat framework. It can be noticed that when the security level increases, the achieved saving of resource consumption can be proportionally expanded. Specifically, the proposed design shows a reduction of 62.97% in LUTs for ML-KEM-512, increasing to 71.55% for ML-KEM-1024, averaging a 71.55% reduction across the three security levels. This achieved improvement is mirrored in the DSPs consumption, which has an average drop of 63.89% when compared to the conventional low-latency implementation. Such improvement in the area performance is achieved at the cost of 68.14% overhead on the timing performance in terms of the execution time in microseconds.

The compact implementations presented in [8], [13] exhibit constant area consumption across different security levels, yet they demand a considerable increase in the clock cycle usages. Specifically, our proposed area-efficient design delivers a remarkable reduction in execution time by an average of 90.09% and 81.64% when compared against [8] and [13], respectively. In fact, this accelerated timing performance comes at a trade-off. Compared to [8] and [13], our design demands a 92.45% and 93.27% increment in LUTs consumption and a 95.79% and 91.58% expansion in DSPs usage, respectively.

In the performance analysis, it can be observed that the area-efficient design adeptly harmonizes the trade-offs between area and timing performance, which maintains equilibrium between clock cycles and hardware resources.

V. Conclusion

This study introduces an innovative, area-efficient architecture tailored for matrix-vector polynomial multiplication in the ML-KEM PQC scheme. The proposed design systematically tackles the challenges of complex data dependencies to ensure consistent result generation based on novel design techniques.

ACKNOWLEDGEMENT

This work is supported in part by the NSF under Grant numbers CCF-2243053 and CCF-2243052.

REFERENCES

[1] National Institute of Standards and Technology, "Module-lattice-based keyencapsulation mechanism standard," Department of Commerce, Washington, D.C., Federal Information Processing Standards Publication (FIPS) NIST FIPS 203 ipd, 2023. [Online]. Available: https://doi.org/10.6028/NIST.FIPS.203.ipd

[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 specification and supporting documentation (version 3.02)," Round-3 submission to the NIST Post-Quantum Cryptography Standardization Project, 2020, https://cryptojedi.org/papers/#kybernistr3.

[3] A. Langlois and D. Stehlé, "Worst-case to average-case reductions for module lattices," *Designs, Codes and Cryptography*, vol. 75, no. 3, pp. 565–599, 2015.

[4] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM (JACM)*, vol. 56, no. 6, pp. 1–40, 2009.

[5] D. A. Parker and K. K. Parhi, "Low-area/power parallel FIR digital filter implementations," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 17, no. 1, pp. 75–92, 1997.

[6] K. K. Parhi, *VLSI digital signal processing systems: design and implementation*. John Wiley & Sons, 1999.

[7] Y. Zhu, Z. Liu, and Y. Pan, "When NTT meets karatsuba: preprocess-then-NTT technique revisited," in *Information and Communications Security: 23rd International Conference, ICICS 2021, Chongqing, China, November 19-21, 2021, Proceedings, Part II*. Springer, 2021, pp. 249–264.

[8] Y. Xing and S. Li, "A compact hardware implementation of CCA-secure key exchange mechanism CRYSTALS-KYBER on FPGA," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 328–356, 2021.

[9] A. Aikata, A. C. Mert, M. Imran, S. Pagliarini, and S. S. Roy, "KaLi: A crystal for post-quantum security using Kyber and Dilithium," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2022.

[10] W. Tan, Y. Lao, and K. K. Parhi, "KyberMat: Efficient accelerator for matrix-vector polynomial multiplication in CRYSTALS-Kyber scheme via NTT and polyphase decomposition," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE, 2023, pp. 1–9.

[11] K. K. Parhi, "Hierarchical folding and synthesis of iterative data flow graphs," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 60, no. 9, pp. 597–601, 2013.

[12] K. Parhi, C.-Y. Wang, and A. Brown, "Synthesis of control circuits in folded pipelined DSP architectures," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 1, pp. 29–43, 1992.

[13] W. Guo, S. Li, and L. Kong, "An efficient implementation of KYBER," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 3, pp. 1562–1566, 2022.

[14] X. Hu, J. Tian, M. Li, and Z. Wang, "AC-PM: An area-efficient and configurable polynomial multiplier for lattice based cryptography," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2022.

[15] Y. Zhao, R. Xie, G. Xin, and J. Han, "A high-performance domain-specific processor with matrix extension of RISC-V for Module-LWE applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 7, pp. 2871–2884, 2022.

[16] Z. Ni, A. Khalid, M. O'Neill, W. Liu *et al.*, "HPKA: A high-performance CRYSTALS-Kyber accelerator exploring efficient pipelining," *IEEE Transactions on Computers*, 2023.

[17] Y. Zhao, S. Pan, H. Ma, Y. Gao, X. Song, J. He, and Y. Jin, "Side channel security oriented evaluation and protection on hardware implementations of Kyber," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2023.

[18] N. K. Unnikrishnan and K. K. Parhi, "Multi-channel fft architectures designed via folding and interleaving," in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2022, pp. 142–146.

[19] W. Tan, S.-W. Chiu, A. Wang, Y. Lao, and K. K. Parhi, "PaReNTT: Low-latency parallel residue number system and NTT-based long polynomial modular multiplication for homomorphic encryption," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 1646–1659, 2024.

[20] H. Nejatollahi, S. Shahhosseini, R. Cammarota, and N. Dutt, "Exploring energy efficient quantum-resistant signal processing using array processors," in *ICASSP 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 1539–1543.

[21] F. Hirner, A. C. Mert, and S. S. Roy, "PROTEUS: A tool to generate pipelined number theoretic transform architectures for FHE and ZKP applications," *Cryptology ePrint Archive*, 2023.