Exploring Energy Efficient Quantum-resistant Signal Processing Using Array Processors

Hamid Nejatollahi[*], Sina Shahhosseini[*], Rosario Cammarota[†], Nikil Dutt[*]
[*]University of California, Irvine, USA
[†]Intel AI, Privacy and Security Research, USA

***Abstract*—Quantum computers threaten to break public-key cryptography schemes such as DSA and ECDSA in polynomial time, which poses an imminent threat to secure signal processing. Ring learning with error (RLWE) lattice-based cryptography (LBC) is one of the most promising families of post-quantum cryptography (PQC) schemes in terms of efficiency and versatility. Two conventional methods to compute polynomial multiplication, the most compute-intensive routine in the RLWE schemes, are convolutions and Number Theoretic Transform (NTT).**

In this work, we explore the energy efficiency of polynomial multiplier using systolic architecture for the first time. As an early exploration, we design two high-throughput systolic array polynomial multipliers, including *NTT-based* and *convolution-based*, and compare them to our low-cost sequential (non-systolic) NTT-based multiplier. Our sequential NTT-based multiplier achieves 3x speedup over the state-of-the-art FGPA implementation of the polynomial multiplier in the NewHope-Simple key exchange mechanism on a low-cost Artix7 FPGA. When synthesized on a Zynq UltraScale+ FPGA, the *NTT-based* systolic and *convolution-based* systolic designs achieve on average 1.7x and 7.5x speedup over our sequential NTT-based multiplier respectively, which can lead to generating over 2x more signatures per second by CRYSTALS-Dilithium, a PQC digital signature scheme. These explorations help designers select the right PQC implementations for making future signal processing applications quantum-resistant.

***Index Terms*—Public Key Cryptography, Lattice-based Cryptography, Acceleration, Number Theoretic Transform, Systolic Array**

I. Introduction

Industry, academia, and government are working together to realize quantum computers that can achieve unprecedented levels of performance in specific application domains, such as biology and chemistry. With the power of quantum computers, solving problems such as the discrete logarithm problem promises to nullify the effectiveness of current public-key cryptography, as illustrated by Shor's algorithm to factorize large integers in polynomial time [1]. Breaking public-key cryptography creates the need to secure signal processing using newer Quantum-resistant cryptography algorithms.

Fortunately, post-quantum cryptography (PQC) is a vibrant area of research devoted to studying alternative schemes for public-key cryptographic protocols, capable of withstanding quantum cryptanalysis attacks, and executable on classical computers [2]. The relevance of the problem is demonstrated by the evaluation and standardization process of PQC algorithms. The most relevant evaluation effort was started with the submission of potential candidates to the National Institute of Standards and Technology (NIST) in 2017 and continues with the evaluation of the candidates until the creation of recommendations for adoption by standards bodies.

Lattice-based cryptography (LBC) schemes are the most promising family of quantum-resistant schemes due to their versatility and superior performance. In both the first and second rounds of the NIST PQC competition, about half of the candidates belong to the LBC family. In this work, we adopt the systolic architecture [3] to accelerate the polynomial multiplication, which is the heart of a subset of (LBC) algorithms (i.e., ideal LBC). The compute-intensive polynomial multiplier kernel can be performed using the Schoolbook algorithm in O(N^2) time complexity [4]. The Discrete Fourier Transform (DFT) and its fast variant, Fast Fourier Transform (FFT), are the two other candidates to multiply two polynomials. The former has a time complexity of O(N^2), which involves matrix-vector multiplication in the time domain, while the latter has a complexity of O(N.logN). Both schemes are critically dependent on acceleration to achieve satisfactory performance. However, increased performance comes at the expense of energy overheads.

Towards this end, we allow signal processing designers to evaluate tradeoffs between performance and energy of quantum-resistant cryptographic schemes using systolic array architectures [3]. Thus, energy-efficient quantum-resistant signal processing schemes can be developed for a wide range of applications, from resource-constrained internet-of-things (IoT) settings to performance-driven real-time signal processing scenarios.

The contributions of this work are summarized as the following:

- Analyze, for the first time, energy consumption of various systolic array implementations of polynomial multiplier for accelerating quantum-resistant signal processing.
- Replace the NTT-based algorithm as the standard in LBC with a convolution-based multiplier that leads to 7x improvement in the execution time of the polynomial multiplier.
- Reach an order of magnitude speedup in computing polynomial multiplier over the state-of-the-art FGPA implementation of the polynomial multiplier of the NewHope-Simple key exchange mechanism. [5].

II. Background and Related Work

A lattice $L \subset \mathbb{R}^n$ is the set of all integer linear combinations of basis vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^n$. i.e., $L = \Big\{ \sum a_i \mathbf{b}_i \ : \ a_i \in \mathbb{Z} \Big\}$. LBC exploits the hardness of two problems: Short Integer Solution (SIS) and Learning With Errors (LWE) [6]. Cryptosystems based on the LWE problem, the most common one, have their foundation in the difficulty of finding the secret key sk, given (A, pk), where $pk = A * sk + e \bmod q$, given the public key pk, an error vector e with Gaussian distribution, and a matrix A of constants in $\mathbb{Z}_q^{r \times n}$ chosen randomly from a uniform distribution. Because LWE requires large keys (e.g., 11 KB for Frodo [7]), it can be impractical on devices with limited on-chip memory. To overcome this limitation, Lyubashevsky et. al [8] introduced Ring-LWE (RLWE), a derivation of LWE in which A is implicitly defined as a vector a in the ring $\mathcal{R} \equiv \mathbb{Z}_q[x]/\langle x^n + 1 \rangle$. Arithmetic operations for a `Ring-LWE`-based scheme are performed over a ring of polynomials. Let n, a power of two, and p be the degree of the lattice and a prime number (p = 1 mod $2n$), respectively. Z_p denotes the ring of integers modulo p, and x^{n+1} is an irreducible degree n polynomial. The quotient ring R_p contains all polynomials with degree less than n in Z_p, that defines R_p = Z_p/$[x^{n+1}]$ in which coefficients of polynomials are in [0,p). Degrees of the polynomials in RLWE-based schemes vary between 256 [9] and 1024 [10].

In the following, we describe two common methods to compute the polynomial multiplication.

A. Convolution-based multiplier

The easiest way to multiply two polynomials is to use the convolution (Schoolbook [11]) with the time complexity of $O(n^2)$ as shown in Algorithm 1. A convolution-based multiplier can be seen as a discrete feed-forward finite impulse response (FIR) over the polynomials in $\mathcal{R} \equiv \mathbb{Z}_q[x]/\langle x^n + 1 \rangle$. Due to the inefficiency of the convolution-based multiplier, there is no notable work that uses it for the acceleration; however, by using its systolic architecture, we can achieve considerable performance gains.

B. NTT-based multiplier

Polynomial multiplier is usually implemented by using the Number Theoretic Transform (NTT), which drops the time complexity of the polynomial multiplier from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \cdot \log n)$. Polynomials $a = a(n-1) \cdot x^{n-1} + \ldots + a(0)$ and $b = b(n-1) \cdot x^{n-1} + \ldots + b(0)$ are transformed into their NTT representations $A = A(n-1) \cdot x^{n-1} + \ldots + A(0)$ and $B = B(n-1) \cdot x^{n-1} + \ldots + B(0)$, and their multiplication can be computed coefficient-wise as $C = \sum_{i=0}^{n-1} A(i) \cdot B(i) \cdot x^i$. The result $c = a * b$ is obtained after the computation of the inverse number theoretic transform (NTT^{-1}) of C. Algorithm 2 describes the NTT-based polynomial multiplier. Figure 1 illustrates basic blocks of the NTT-based polynomial multiplier. One standard method to perform the number theoretic transform is Cooley-Tukey (CT) [12], which produces the result in the bit-reverse order by receiving the input in the correct order; the other method is Gentleman-Sande (GS) [13], which receives the input in the reverse order and produces the output in the correct order. Similar to [5], we use the Gentleman-Sande method to compute both NTT and NTT^{-1}, which needs bit-reverse calculation. As previously mentioned, NTT designs are known as *parallel NTT* due to the parallel use of butterfly processing elements. *Pipeline FFT* processor [14] is a high-throughput design of FFT that utilizes single-path delay feedback. Authors in [15] adopt *Pipeline FFT* to design a systolic array polynomial multiplier to develop a high throughput RLWE cryptoprocessor. In this work, we focus on the parallel NTT designs and leave the *Pipeline FFT* for future work.

Algorithm 1 Convolution (Schoolbook)-based Polynomial Multiplier

```
Initialization: Let $a = \{a_0, a_1, a_2, ..., a_{n-1}\}$ and $b = \{b_0, b_1, b_2, ..., b_{n-1}\} \in \mathbb{Z}_q[x]/<f(x)>$ be two polynomials with the length of $n$, where $f(x) = x^n + 1$ is an irreducible polynomial with $n$ a power of 2, and $q \equiv 1 \, mod \, 2n$ is a large prime number.
$c \leftarrow 0$
for $i = 0$ to $n - 1$ do
    for $j = 0$ to $j - 1$ do
        $sign \leftarrow (-1)^{\lfloor (i+j)/n \rfloor}$
        $index \leftarrow (i + j) \ mod \ n$
        $coeff \leftarrow a_i b_i \ mod \ q$
        $c_{index} \leftarrow integer(c_{index} + sign * coeff) \ mod \ q$
    end for
end for
Return c
```

Algorithm 2 NTT-based Polynomial Multiplier

```
Initialization: Let $a = \{a_0, a_1, a_2, ..., a_{n-1}\}$ and $b = \{b_0, b_1, b_2, ..., b_{n-1}\} \in \mathbb{Z}_q[x]/<f(x)>$ be two polynomials with length of $n$, where $f(x) = x^n + 1$ is an irreducible polynomial with $n$ a power of 2, and $q \equiv 1 \, mod \, 2n$ is a large prime number. $w$ is the $n$-th root of unity and $\phi$ is the $2n$-th root of unity ($\phi^2 = w$ mod $q$); $w^{-1}$ and $\phi^{-1}$ are the inverse of $w$ mod$q$ and $\phi$ mod$q$, respectively.
Precompute: $\{w^i, w^{-i}, \phi^i, \phi^{-i}\}$ for $i \in [0, n - 1]$
for $i = 0$ to $n - 1$ do
    $\bar{a_i} \leftarrow a_i \phi^i$
    $\bar{b_i} \leftarrow b_i \phi^i$
end for
$\bar{A} \leftarrow NTT_w^n(\bar{a})$
$\bar{B} \leftarrow NTT_w^n(\bar{b})$
$\bar{C} = \bar{A}.\bar{B}$
$\bar{c} \leftarrow iNTT_w^n(\bar{C})$
for $i = 0$ to $n - 1$ do
    $c_i \leftarrow \bar{c_i} \phi^{-i}$
end for
Return C
```

C. Previous works

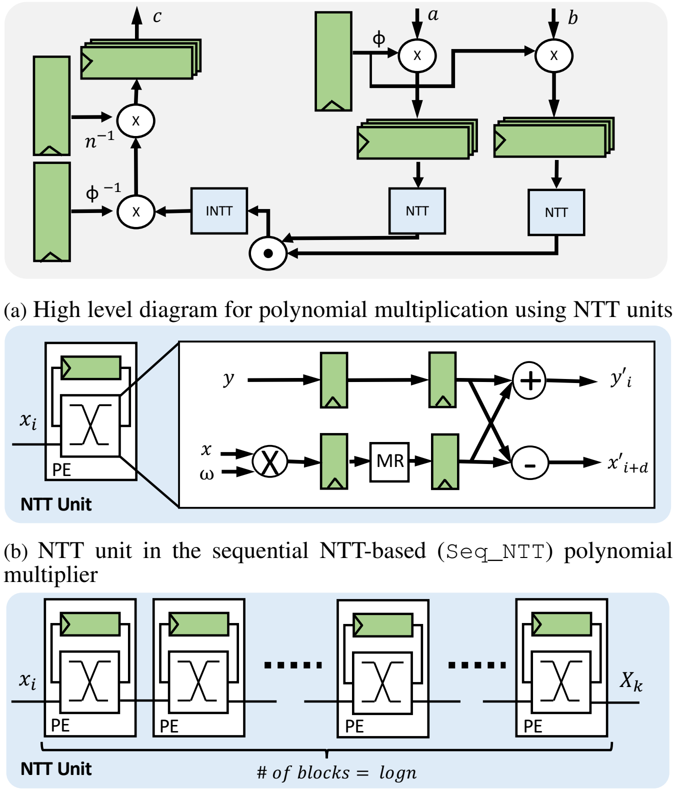
Previous efforts on the acceleration of the NTT mostly focus on the area and performance of the of the polynomial multiplier for LBC and leave the energy unexplored [16] [17]. Some efforts have evaluated the energy as well as the area and performance of NTT accelerators [18], [19], [20]. The only notable work that uses systolic arrays to accelerate polynomial multiplier reports only performance and area metrics for $n = 256$ and $n = 512$; however, we report energy as the primary goal as well as the performance and area for $n \in$ (128, 256, 512, 1024).

III. Systolic Array polynomial multiplier

We use the concept of systolic array architecture to design polynomial multipliers.

A. Systolic arrays

Today's systems are intensely designed to move data for computation. Data movement is highly expensive in terms of energy consumption and latency compared to computation. Consequently, the movement of data is the critical bottleneck in computing systems as applications become more data-intensive. To address the bottleneck, we need an alternative architecture – such as a systolic array – to process data with less data movement. A systolic array consists of a set processing elements (PE), each capable of performing simple operations. Each PE is connected to its nearest PEs and performs operations on data. Data flows from the memory cells and passes through PEs before returning to the memory cells. Systolic architecture relieves the repeated memory access problem for general-purpose computing systems, which in turn helps to reduce latency.



(a) High level diagram for polynomial multiplication using NTT units

(b) NTT unit in the sequential NTT-based (`Seq_NTT`) polynomial multiplier

(c) NTT unit in the systolic array NTT-based (`SA_NTT`) polynomial multiplier

Fig. 1: (a) Polynomial multiplication using NTT units (b) NTT unit based on only one PE which processes inputs sequentially in $(n/2)logn$ iterations. In order to perform polynomial multiplication NTT unit is executed three times (c) NTT-based systolic array polynomial multiplier encompass $logn$ PE blocks each performs butterfly operation in $n/2$ iterations.

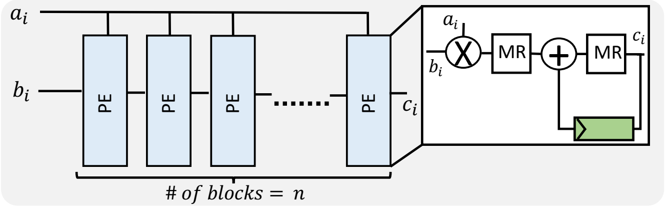


Fig. 2: Convolution-based polynomial multiplier using the systolic array. Modular reduction (MR) is performed after each multiplication and addition.

We design and implement two systolic array polynomial multipliers including NTT-based and convolution-based systolic array and compare them to the sequential NTT-based multiplier.

B. Implementation of systolic array polynomial multiplier

1) NTT-based polynomial multiplier: The conventional hardware implementation for the NTT-based polynomial multiplier uses a processing element with only one butterfly block sequentially (Figure 1-a). We use sequential NTT-based multiplier (`Seq_NTT`) as the baseline in our experiments; `Seq_NTT`, our slowest design, provides 3x speedup to compute forward and inverse number theoretic transforms compared to the implementation of [5] on a low-cost Artix7 Xilinx FPGA.

The NTT-based polynomial multiplier can be implemented using a systolic array (`SA_NTT`for the rest of the paper). Figure 1-b shows a $logn$ array of processing elements, each executes $n/2$ butterfly operations on all coefficients of the input polynomial. In other words, we cascade all $logn$ stages of the NTT-method and connect them using FIFO buffers. According to the Gentleman-Sande method, extracting parallelism between stages of the NTT is non-trivial; we can improve the performance of NTT by fusing multiple stages through the dataflow optimization of a high-level synthesis (HLS) tool.

2) Convolution-based polynomial multiplier: The time complexity of the convolution-based polynomial multiplier can be decreased to O(n) if we adjust the systolic architecture to use n-cascaded multiply-accumulators (MACs). As shown in Figure 2, each processing element in the convolution-based polynomial multiplier (`CONV`) performs modular reduction (MR) after each multiplication and addition. The significant performance improvement comes at the large area and energy overhead of performing n MACs. We use our optimized versions of Montgomery [21] and Barrett [22] reduction after each multiplication and addition, respectively, using only shifts and additions.

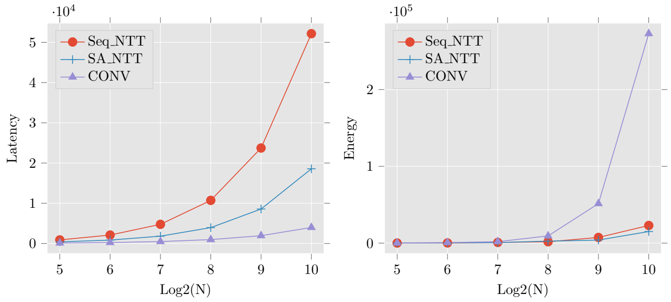
IV. Evaluation

To perform the synthesis, we use Vivado high-level synthesis (HLS) 2018.2 and select Artix7 and Zynq UltraScale+ as the target FPGA devices from resource-constrained internet-of-things (IoT) settings and performance-driven real-time signal processing scenarios, respectively. We extract the numbers of reported resources (BRAM, CLB, DSP, FF, and LUT), maximum achieved frequency, and energy from the post-implementation process of the HLS tool. The latency of the polynomial multiplier is the number of execution cycles at the maximum achieved frequency.

For the polynomial-size of N=1024, `Seq_NTT` achieves 3x speedup to compute forward and inverse number theoretic transform compared to the implementation of [5] on a low-cost Artix7 FPGA. Table I shows the synthesis results of `Seq_NTT` as the smallest design along with the two systolic array polynomial multipliers, `SA_NTT` (NTT-based) and `CONV` (convolution-based on Zynq UltraScale++. `Seq_NTT` achieves the highest maximum frequency than `SA_NTT` and `CONV` because of its sequential architecture.

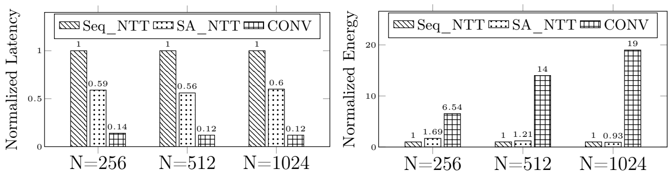
According to Figure 3, with the increase in the degree of the polynomial from 512 to 1024, the rise in the energy consumption of `CONV` is exponential due to the increase in the number of resources required to satisfy the timing constraints.

Figure 4 shows the latency and energy consumption, extracted from Table I, of the designs normalized to the `Seq_NTT`. For N=1024, e.g., NewHope scheme, `SA_NTT` reaches 3x speedup with 7% decrease in energy compared to the `Seq_NTT`; for N=256 and N=512, 1.7x improvement in latency is achieved with 69% and 21% increase in the energy, respectively. We suggest employing a systolic array NTT-based polynomial multiplier for resource-constrained devices to achieve plausible performance with low energy consumption to verify the digital signatures and/or perform encrypting and decryption. The convolution-based systolic array multiplier is suitable for high-performance servers that can tolerate higher energy consumption to generate 2x more signatures per second by CRYSTALS – Dilithium [23], a PQC digital signature scheme with a 7x speedup in the computing NTT. Forward and inverse NTT consumes around 65% of cycles in CRYSTALS – Dilithium to generate a signature.



(a) Latency (b) Energy

Fig. 3: Increase in the latency and energy of polynomial multiplier by the increase in the size of the polynomials.



(a) Latency (b) Energy

Fig. 4: Comparison of the different polynomial multiplier designs, normalized to NTT, for polynomials with degree 256, 512 and 1024.

V. Conclusion and future work

The advent of quantum computing threatens to render ineffective classical cryptographic schemes to secure signal processing applications. Emerging quantum-resistant cryptographic schemes show promise, but are hindered by the computational overhead of essential critical kernels such as a polynomial multiplication. This work explores, for the first time, the energy efficiency of array processors for implementing polynomial multipliers with degrees up to 1024.

We design and synthesize an NTT-based and a convolution-based systolic array polynomial multipliers and compare their performance, area, and energy with the sequential NTT-based counterpart. Our serial NTT-based design achieves 3x speedup on a low-cost Artix7 FPGA compare to the hardware implementation of NewHope-Simple.

On a Zynq UltraScale+ FPGA, NTT-based systolic array on average is 1.7x faster than sequential NTT-based polynomial multiplier with less than 30% increase in the energy. The convolution-based systolic array on average is 7.5x faster than serial NTT-based polynomial multiplier with a 13.5x increase in the energy overhead; thus, convolution-based systolic array polynomial multipliers are suitable for high-performance servers that can tolerate higher energy consumption.

Our future work evaluates the energy efficiency of the *Pipeline FFT* processor to perform polynomial multiplication. Additionally, we make the systolic architecture of the `CONV`more area- and energy-efficient for securing quantum resistance of future signal processing applications.

References

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, 1997.

[2] H. Nejatollahi *et al.*, "Trends, challenges and needs for lattice-based cryptography implementations: Special session," in *CODES*, 2017.

[3] H.-T. Kung, "Why systolic architectures?" *Computer*, 1982.

[4] H. Nejatollahi *et al.*, "Post-quantum lattice-based cryptography implementations: A survey," *ACM CSUR*, 2019.

[5] O. Tobias *et al.*, "Implementing the newhope-simple key exchange on low-cost fpgas," in *LATINCRYPT*, 2017.

[6] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," 2005.

[7] J. Bos *et al.*, "Frodo: Take off the ring! practical, quantum-secure key exchange from lwe," in *CCS*, 2016.

[8] V. Lyubashevsky *et al.*, "On ideal lattices and learning with errors over rings," ser. EUROCRYPT'10, 2010.

TABLE I: High-level synthesis results on Zynq UltraScale++ for three different designs of polynomial multipliers. To multiply two polynomials using NTT-based multipliers, we need to execute NTT block three times, and each includes bit-reversal, forward NTT, and point-wised multiplication of the input vector with the precomputed twiddle factors. Although the degree of the polynomials in most of the RLWE-based schemes ranges from 256 to 1024, we also provide the results for other degrees to show the trends.

Design	N	Cycles	Latency (us)	Energy (uJ)	BRAM	CLB	DSP	FF	LUT	Freq. (MHz)
Seq_NTT	64	2112	7.47	0.16	0	221	10	1120	937	282.48
	128	4776	16.79	0.62	1	364	10	2085	1,520	284.33
	256	10728	36.37	1.27	2	667	10	4281	2999	294.89
	512	23736	83.93	4.36	2	1220	10	8374	5278	282.80
	1024	52152	169.28	13.37	2	2551	10	17091	10888	308.07
SA_NTT	64	897	4.86	0.49	14	366	38	1577	1760	184.39
	128	1878	10.19	0.95	17	426	43	1906	2063	184.16
	256	4008	21.56	2.15	24	547	48	2230	2372	185.83
	512	8634	47.63	5.28	27	605	53	2610	2772	181.25
	1024	18636	101.84	12.52	29	732	58	3007	3140	182.98
CONV	64	271	1.31	0.85	1	2226	322	8746	10223	205.88
	128	533	2.65	3.39	2	4398	642	19098	21501	200.68
	256	1058	5.32	8.33	2	8580	1282	38108	47332	198.57
	512	2107	10.75	64.03	2	19301	2562	135593	161924	195.84
	1024	3720	17.40	257.23	2	49714	4282	310247	322866	195.22

[9] R. Avanzi *et al.*, "Crystals-kyber," NIST, Tech. Rep., 2017.

[10] T. Poppelmann *et al.*, "Newhope," NIST, Tech. Rep., 2017.

[11] D. E. Knuth, *The Art of Computer Programming, Volume 2 (3rd Ed.): Seminumerical Algorithms*, 1997.

[12] J. Cooley *et al.*, "An algorithm for the machine calculation of complex fourier series," *Mathematics of Computation*, 1965.

[13] W. M. Gentleman *et al.*, "Fast fourier transforms: For fun and profit," in *AFIPS*, 1966.

[14] Shousheng He *et al.*, "A new approach to pipeline fft processor," in *ICPP*, 1996.

[15] C. P. Rentería-Mejía *et al.*, "High-throughput ring-lwe cryptoprocessors," *TVLSI*, 2017.

[16] T. Pöppelmann *et al.*, "Accelerating homomorphic evaluation on reconfigurable hardware," in *Lecture Notes in Computer Science*, 2015.

[17] D. B. Cousins *et al.*, "An fpga co-processor implementation of homomorphic encryption," in *HPEC*, 2014.

[18] H. Nejatollahi *et al.*, "Synthesis of flexible accelerators for early adoption of ring-lwe post-quantum cryptography," *TECS*, 2020.

[19] U. Banerjee *et al.*, "Sapphire: A configurable crypto-processor for post-quantumlattice-based protocols," *IACR TCHES*, 2019.

[20] H. Nejatollahi *et al.*, "Flexible ntt accelerators for rlwe lattice-based cryptography," *ICCD*, 2019.

[21] P. L. Montgomery, "Modular multiplication without trial division," *Mathematics of computation*, 1985.

[22] P. Barrett, "Implementing the rivest shamir and adleman public key encryption algorithm on a standard digital signal processor," in *CRYPTO*, 1986.

[23] L. Ducas *et al.*, "Crystals-dilithium," National Institute of Standards and Technology, Tech. Rep., 2017.