Architectural Tradeoffs for Long Polynomial Modular Multiplication

Sin-Wei Chiu and Keshab K. Parhi, *Fellow, IEEE*
Department of Electrical and Computer Engineering
University of Minnesota, Minneapolis, MN, USA
{chiu0091, parhi}@umn.edu

***Abstract*—Polynomial multiplication over the quotient ring is a critical operation in Ring Learning with Errors (Ring-LWE) based cryptosystems that are used for post-quantum cryptography and homomorphic encryption. This operation can be efficiently implemented using number-theoretic transform (NTT)-based architectures. Among these, pipelined parallel NTT-based polynomial multipliers are attractive for cloud computing as these are well suited for high throughput and low latency applications. For a given polynomial length, a pipelined parallel NTT-based multiplier can be designed with varying degrees of parallelism, resulting in different tradeoffs. Higher parallelism reduces latency but increases area and power consumption, and vice versa. In this paper, we develop a predictive model based on synthesized results for pipelined parallel NTT-based polynomial multipliers and analyze design tradeoffs in terms of area, power, energy, area-time product, and area-energy product across parallelism levels up to 128. We predict that, for very long polynomials, area and power differences between designs with varying levels of parallelism become negligible. In contrast, area-time product and energy per polynomial multiplication decrease with increased parallelism. Our findings suggest that, given area and power constraints, the highest *feasible* level of parallelism optimizes latency, area-time product, and energy per polynomial multiplication.**

***Index Terms*—Post-quantum cryptography (PQC), number theoretic transform (NTT), polynomial multiplication, predictive model, parallel processing, pipelining, folding.**

I. Introduction

Polynomial multiplication over the quotient ring is a foundational operation in Ring Learning with Errors (Ring-LWE) based cryptosystems [1], a class of encryption methods that have gained prominence as resilient post-quantum alternatives to classical cryptography. Ring-LWE schemes extend the Learning with Errors (LWE) problem — a well-known problem in lattice-based cryptography — into ring structures, allowing more efficient encryption operations without compromising security. The significance of Ring-LWE lies in its resistance to attacks from both classical and quantum computers, making it a crucial component in the emerging field of post-quantum cryptography [2].

In Ring-LWE cryptosystems, polynomial multiplication is the most computationally intensive operation and frequently becomes a performance bottleneck. This operation can be efficiently implemented using the number-theoretic transform (NTT), which converts polynomial multiplication into NTT, point-wise multiplication (PWM), and inverse-NTT (INTT), significantly reducing computational complexity. NTT-based architectures are particularly suitable for hardware implementations, where parallelism and pipelining can be leveraged to achieve high-throughput and low-latency. Among various NTT-based designs, pipelined parallel polynomial multipliers are especially appealing for hardware applications due to their ability to deliver superior computational speed with minimal latency [3]. However, optimizing these architectures for practical deployment requires a careful evaluation of tradeoffs associated with efficiency, resource usage, and scalability.

Increasing the level of parallelism can significantly reduce latency by enabling more operations to be processed simultaneously. However, this approach requires a larger number of processing elements (PEs), which in turn increases area and power consumption. The area in these architectures is dominated by PEs. Conversely, lower parallelism reduces area and power requirements but results in higher latency. Furthermore, the area in these architectures is dominated by the number of delay elements. Achieving an optimal balance between parallelism, area, and power consumption is thus critical for designing an efficient architecture that meets performance requirements without exceeding hardware constraints.

The contributions of this paper are two-fold. First, we developed a *predictive model* based on synthesized results to evaluate performance metrics, including area, area-time product (ATP), power, energy per polynomial multiplication, and area-energy product (AEP). Second, using insights from this predictive model, we propose a design strategy for selecting optimal levels of parallelism for implementation of pipelined parallel polynomial modular multiplication.

The remainder of this paper is organized as follows. Section II provides background on the preliminaries of polynomial multiplication over the quotient ring. Section III describes the architectures of pipelined parallel polynomial multipliers with varying levels of parallelism. Section IV introduces our predictive model for pipelined parallel polynomial multipliers. Section V presents our results and insights derived from the model. Finally, Section VI concludes the paper.

II. Background

A. Polynomial multiplication over the quotient ring

Polynomial multiplication over the quotient ring $R_{n,q}$ is a critical operation in Ring-LWE based cryptosystems. $R_{n,q} =$

This research was supported in part by the National Science Foundation under grant number CCF-2243053.

$\mathbb{Z}_q(x)/(x^n+1)$ is a quotient ring of polynomial ring $\mathbb{Z}_q(x)$, where the degree of the polynomial is smaller than n and the coefficients of the polynomial are in the range of 0 to $q-1$. The parameter q is the modulus and has the property of $q \equiv 1 \mod 2n$.

A polynomial $a(x)$ in the quotient ring $\mathbb{Z}_q(x)/(x^n+1)$ can be expressed as:

$$a(x) = a_0 + a_1 x + \ldots + a_{n-1}x^{n-1},\ a_i \in [0, q-1] \quad (1)$$

Given two polynomials $a(x)$ and $b(x)$. The polynomial multiplication over the quotient ring $R_{n,q}$ is defined as:

$$p(x) = a(x) \times b(x) \mod (x^n+1, q) \quad (2)$$

The modular reduction of (x^n+1) and coefficient-wise modular reduction over q ensure $p(x)$ lies within the quotient ring $R_{n,q}$. The coefficient p_i of $p(x)$ can be represented as:

$$p_i = \sum_{j=0}^{i} a_j b_{i-j} - \sum_{j=i+1}^{n-1} a_j b_{n+i-j} \mod q,\ i \in [0, n-1] \quad (3)$$

Equation 3 also represents the modular form of negative-wrapped convolution, implying that polynomial multiplication over the quotient ring can be interpreted as a coefficient-wise modular negative-wrapped convolution.

It is well known that convolution can be computed efficiently using the Fast Fourier Transform (FFT) with time complexity of $O(n \log n)$ [4]. Similarly, negative-wrapped convolution can be computed efficiently using low-complexity NTT and INTT, also achieving time complexity of $O(n \log n)$ [5], [6].

B. Low-complexity NTT and INTT

The low-complexity NTT and INTT (LC-NTT/INTT) can be used to efficiently compute negative-wrapped convolution [5]. By integrating pre- and post-processing weighted operations directly into the NTT/INTT butterflies, LC-NTT/INTT eliminates the need for additional modular multipliers before and after the butterfly stages. This approach is especially advantageous in parallel architectures, where it reduces both area and power.

C. Modular Reduction

Computation over the quotient ring requires coefficient-wise modular reduction to ensure that values remain within the quotient ring. For modular addition and subtraction, a single conditional subtraction is applied to bring the output back to the range $[0, q-1]$. For modular multiplication, one effective approach is to use Barrett reduction [7], which replaces costly long division with multiplication, making the operation more efficient. Given two inputs a and b of word-length u, the modular multiplication r of a and b modulo q is defined as:

$$r = a \times b \mod q \quad (4)$$

Using Barret reduction, we can rewrite Equation 4 as:

$$a \times b \mod q = (a \times b) - \left(\frac{(a \times b) \times m}{2^{2u}}\right) \times q \quad (5)$$

$$= (a \times b) - ((a \times b \times m) \gg 2u) \times q \quad (6)$$

where $m = \lfloor \frac{2^{2u}}{q} \rfloor$ and $\gg$ denotes the right shift operation. The floor function ensures that the output of Equation 6 falls within the range $[0, 2q-1]$, A conditional subtraction is then applied to bring the result r back into the range $[0, q-1]$.

III. Pipelined Parallel Polynomial Multiplier Architectures

A polynomial multiplier for computation over the quotient ring consists of 3 stages, LC-NTT, point-wise multiplication (PWM), and LC-INTT. All coefficient computations in $\mathbb{Z}_q$ require modular reductions. Figure 1 illustrates the high-level block diagram of a polynomial multiplier for computation over the quotient ring.

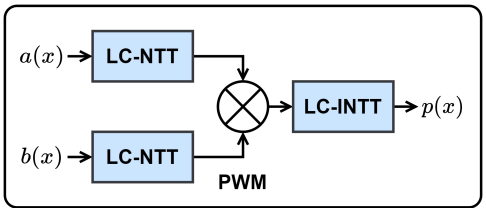


Fig. 1: The high-level block diagram of a polynomial multiplier for computation over the quotient ring.

A parallel architecture [8], [9] can support varying levels of parallelism. For a parallel LC-NTT/INTT, the parallelism level L can take values of 2^i, where i ranges from 1 to $\log_2 n$. Within each parallel LC-NTT/INTT block, there are $s = \log_2 n$ stages, and each stage contains $L/2$ Processing elements (PEs). Figure 2 illustrates the high-level block diagrams of parallel LC-NTT/INTT architectures for various levels of parallelism from 2 to L-parallel.

Each PE in LC-NTT/INTT consists of a modular multiplier and a butterfly unit. The modular multiplier, shown in Figure 3, is designed based on Equation (6).

From [5], we can observe that the butterfly units in LC-NTT follow the decimation-in-time form, while the butterfly units in LC-INTT follow the decimation-in-frequency form.

In LC-NTT, the butterfly unit contains a butterfly operation and a modular multiplier at the lower input as shown in Fig. 4(a). ω_n is the n-th root of unity, where $\omega_n^n \equiv 1 \pmod q$. ψ_{2n} is the 2n-th root of unity, where $\psi_{2n}^{2n} \equiv 1 \pmod q$ and $\psi_{2n}^2 \equiv \omega_n \pmod q$. In LC-INTT the butterfly unit contains a butterfly operation, a multiplication by 2^{-1} block at the upper output, and a modular multiplier at the lower output as shown in Fig. 4(b). It is important to note that multiplications by 2^{-1} occur at both the upper and lower outputs. These operations arise from decomposing the n^{-1} multiplications at the end of the INTT into s stages. This approach is advantageous because, in modular arithmetic, multiplication by 2^{-1} can be implemented efficiently without performing an actual multiplication [5].

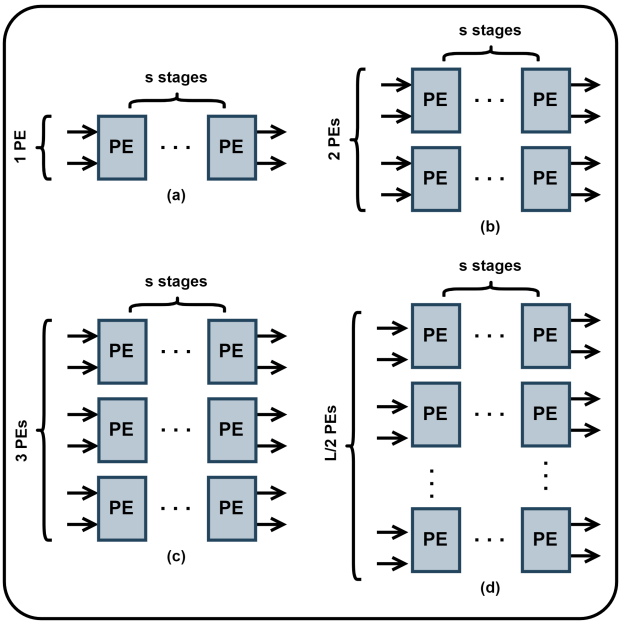


Fig. 2: The high-level block diagrams of parallel LC-NTT/INTT architectures for various levels of parallelism: (a) 2-parallel (b) 4-parallel (c) 8-parallel (d) L-parallel.

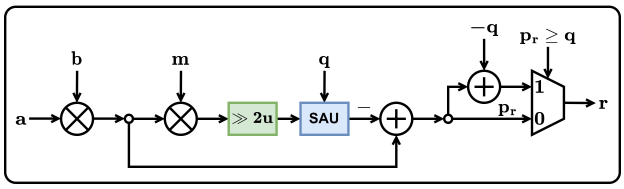


Fig. 3: The block diagrams of modular multiplier based on Equation (6).

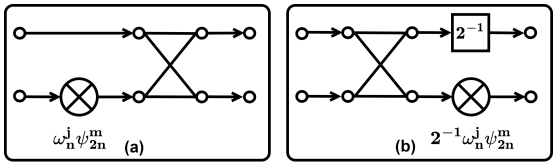


Fig. 4: The block diagram of the (a) LC-NTT butterfly unit and (b) LC-INTT butterfly unit.

The parallel architectures are designed using folding transformation [10], a technique that groups operations into folding sets—ordered sequences of tasks executed by the same functional unit. Here we give a simple example of the folding transformation of a 2-parallel 4-point LC-NTT. Given the folding sets below:

$$A = \{A_0, A_1\} \quad B = \{B_1, B_0\}$$

we can derive the data-flow graph (DFG) of the 2-parallel 4-point LC-NTT (Figure 5a). Each circle in the DFG represents a multiply-and-butterfly operation prior to folding. For instance, (A_0, A_1) are folded into the same processing element, PE A. The numbers above each circle denote the specific time at which each operation is executed. From the derived DFG, we can construct the 2-parallel 4-point LC-NTT architecture, as shown in Figure 5b. Between the PEs, a commutator circuit, implemented as a delay-switch-delay (DSD) unit, is used for intermediate data reordering.

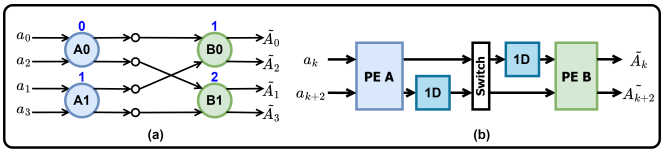


Fig. 5: The 2-parallel 4-point LC-NTT (a) DFG (b) derived architecture.

Table I summarizes the pipelined parallel polynomial multiplier architectures in terms of the number of PEs, delay elements, and PWMs in each architecture for various levels of parallelism. The delay elements include the delay elements in DSD, and additional pipelining cutsets inserted into both the PEs and the PWMs. The parameter p represents the number of pipelining cutsets within each butterfly unit, while the pointwise multiplier contains $p-1$ pipelining cutsets.

TABLE I: The summary of the number of blocks

Design	#PEs$^{\alpha}$	#DEs$^{\beta}$ in DSD	#DEs$^{\beta}$ in pipe	#PWMs
2-parallel	$3\log_2 n$	$3n-6$	$(3\log_2 n+2)p-2$	2
4-parallel	$6\log_2 n$	$3n-12$	$(6\log_2 n+4)p-4$	4
L-parallel	$\frac{3}{2}L\log_2 n$	$3n-(3L)$	$(\frac{3}{2}L\log_2 n+L)p-L$	L

$^{\alpha}$: Processing elements
$^{\beta}$: Delay elements

IV. Predictive Model for Pipelined Parallel Polynomial Multiplier

In this section, we present our predictive model for the pipelined parallel polynomial multiplier. The model is built using a custom fitting function and actual synthesized area and power data. The custom fitting function is created based on Table I in terms of parameters n, L, p, p_A, m_A, d_A. Here, p_A, m_A, and d_A denote the area of PE, PWM, and a delay element. In our proposed predictive model, the area of a pipelined L-parallel design $A(n, L)$ is defined by the equation:

$$A(n,L) = \alpha_A p_A \left(\frac{3}{2}L\log_2 n\right) + \beta_A m_A L + \gamma_A d_A \left((3n-4L) + \left(\frac{3}{2}L\log_2 n + L\right)p\right) + b_A \quad (7)$$

where α_A, β_A, γ_A, and b_A are coefficients that account for the routing and optimizations applied during the synthesis step for the complete pipelined parallel polynomial multiplier. A similar equation can be used to define the power of a pipelined L-parallel design, $P(n, L)$, by replacing the terms related to area with their power counterparts:

$$P(n,L) = \alpha_P p_P \left(\frac{3}{2} L \log_2 n\right) + \beta_P m_P L + \gamma_P d_P \left((3n - 4L) + \left(\frac{3}{2} L \log_2 n + L\right) p\right) + b_P \quad (8)$$

Leveraging Equations (7) and (8), we can construct a nonlinear predictive model for the pipelined parallel polynomial multiplier using data points from fully synthesized blocks. In the next section, we present the implementation of the predictive model and discuss the major findings derived from its results.

V. Results

In this section, we explain the implementation of the model and discuss the findings from the results. The first step of implementing the model is obtaining the data points from fully synthesized blocks. We map and synthesize complete 2-parallel and 4-parallel polynomial modular multipliers to a 28nm technology (global operation voltage at 0.9 VDD) node using SystemVerilog HDL and Synopsys Design Compiler. We evaluate different parameter sets, fixing the size of q at 30 bits, frequency f at 500 MHz, and varying the polynomial length from 16 to 4096. Tables II and III summarize the area and power consumption of the 2-parallel and 4-parallel polynomial multipliers.

TABLE II: Area and power of 2-parallel polynomial mdoular multipliers

n	16	64	256	1024	4096
Area (mm^2)	0.062	0.099	0.168	0.365	1.074
Power (mW)	14.093	24.188	54.695	166.94	604.316

TABLE III: Area and power of 4-parallel polynomial modular multipliers

n	16	64	256	1024	4096
Area (mm^2)	0.121	0.184	0.282	0.513	1.292
Power (mW)	27.132	42.345	78.52	196.664	648.203

The next step is to obtain the parameters p_A, m_A, d_A, b_A, p_P, m_P, d_P, and b_P in Equations (7) and (8). We achieve this by mapping and synthesizing one PE, PWM, and delay element individually, fixing the size of q at 30 bits, and frequency at 500 MHz. Table IV summarizes the area and power consumption of the individual blocks

TABLE IV: Area and power consumption of the individual blocks

Block	PE	PWM	Delay element
Area (μm^2)	4236.75	3512.5	71.82
Power (mW)	1.428	1.025	0.053

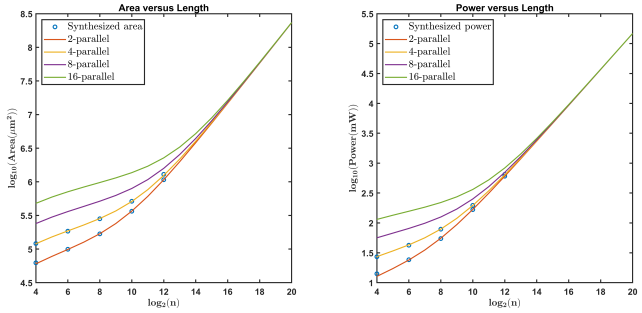
After obtaining the synthesized data points and the required parameters for fitting, we use the MATLAB fittype function to design the predictive model. The predictive model is created using nonlinear least squares fitting to the synthesized data points, specifically using the bisquare weights method. Table V summarizes the model coefficient values obtained through nonlinear least squares fitting. These model coefficient values complete our predictive model for the pipelined parallel polynomial multiplier. The model can be used to evaluate performance metrics for different values of n and L.

TABLE V: Model coefficient values

Coef.	p_A	m_A	d_A	b_A	p_P	m_P	d_P	b_P
Value	1.007	0.563	1.027	-2941	0.3542	2.704	0.8845	-4.028

Figure 6 illustrates the area and power versus polynomial length fitting plot for different levels of parallelism (with L up to 16 and n up to 2^{20}). The y-axis is plotted on a logarithmic scale (base 10), while the x-axis (n) is plotted on a logarithmic scale (base 2). The blue circles represent the synthesized data points. When n is small, the area and power are proportional to L. However, as n increases, the area and power values converge to the same value independent of L. This is because, when n is small, the design is PE-dominant. However, as n increases, the design becomes delay element-dominant, as the number of delay elements grows linearly.

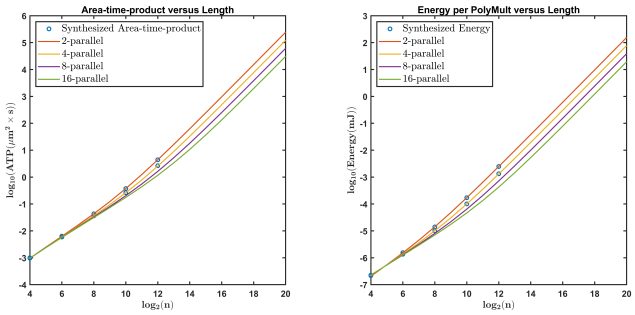
Figure 7 illustrates the ATP and energy versus polynomial length fitting plot for various L. ATP is defined as the area-block processing period (BPP) product, where BPP is the time period required to generate n sample outputs after the first sample is produced. For a pipelined parallel polynomial multiplier, the BPP is $((n/L) \cdot (1/f))$. Energy represents the energy required to compute one length-n polynomial multiplication over the quotient ring, which is calculated as the power-BPP product. Unlike area and power, ATP and energy for various L show minimal differences when n is small. However, as n increases, the ATP and energy are inversely related to L.



(a) Area versus $\log_2 n$. (b) Power versus $\log_2 n$.

Fig. 6: Area and power versus polynomial length.

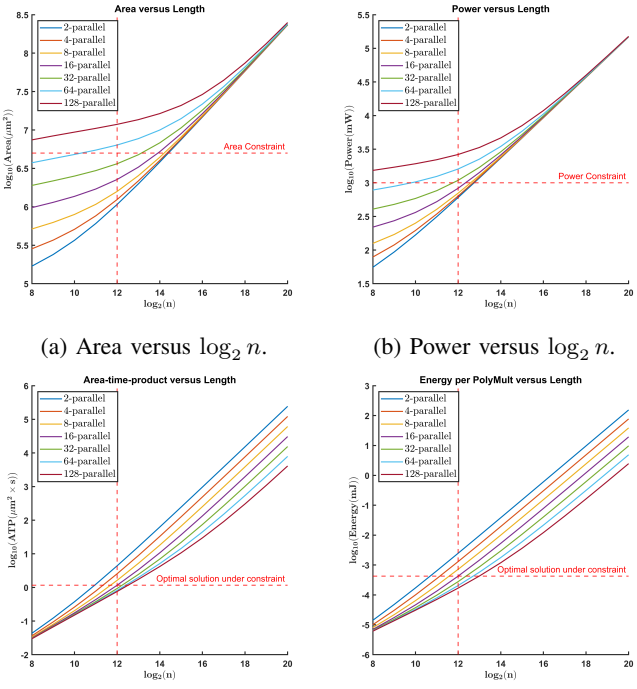
Utilizing this model, we can further predict architectures with even higher levels of parallelism. Figure 8 illustrates the area, power, ATP, and energy versus polynomial length for $L \leq 128$. The trends observed in Figure 8 align with those shown in Figures 6 and 7. The major finding from these figures is that given area and power constraints, *the highest feasible level of parallelism* optimizes latency, area-time product, and energy per polynomial multiplication. For example, assume a



(a) ATP versus $\log_2 n$. (b) Energy versus $\log_2 n$.

Fig. 7: ATP and energy versus polynomial length.

5 mm^2 area constraint (Figure 8a) and a 1 W power constraint (Figure 8b). The designs that satisfy both constraints are those with L ranging from 2 to 16. Considering the performance metrics of latency (BPP), ATP (Figure 8c), and energy per polynomial multiplication (Figure 8d), the 16-parallel design provides the optimal solution.

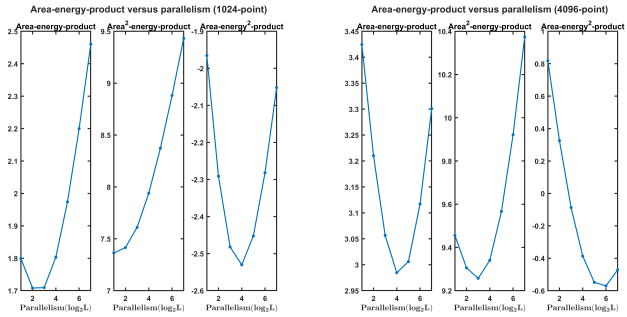


(a) Area versus $\log_2 n$. (b) Power versus $\log_2 n$.

(c) ATP versus $\log_2 n$. (d) Energy versus $\log_2 n$.

Fig. 8: Area, power, ATP, and energy versus polynomial length for $L \leq 128$.

In addition to the four performance metrics, we also consider the area-energy product (AEP). There are three AEP variants: area-energy product, area^2-energy product, and area-energy^2 product. Figure 9 illustrates the AEP for $n = 1024$ and $n = 4096$. For $n = 1024$, the optimal L values for the three variants are 4, 2, and 16, respectively. When $n = 4096$, the plots shift to the right, and the optimal L values become 16, 8, and 64, respectively.



(a) $n = 1024$. (b) $n = 4096$.

Fig. 9: AEP versus levels of parallelism ($\log_2 L$).

VI. Conclusion

In this work, we have presented a predictive model for pipelined parallel NTT-based polynomial multipliers. Our analysis reveals that increasing parallelism significantly reduces latency but comes at the cost of higher area and power consumption. However, for very long polynomial lengths, the differences in area and power across various parallelism levels become negligible, while the area-time product and energy consumption decrease inversely with parallelism. These findings highlight the importance of selecting the optimal level of parallelism, balancing area, power, and performance tradeoffs to achieve efficient hardware.

References

[1] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in *Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29*. Springer, 2010, pp. 1–23.

[2] D. J. Bernstein and T. Lange, "Post-quantum cryptography," *Nature*, vol. 549, no. 7671, pp. 188–194, 2017.

[3] 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, 2023.

[4] A. Oppenheim and R. Schafer, *Discrete-Time Signal Processing*. Pearson Education, 2011.

[5] N. Zhang, B. Yang, C. Chen, S. Yin, S. Wei, and L. Liu, "Highly efficient architecture of NewHope-NIST on FPGA using low-complexity NTT/INTT," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 49–72, 2020.

[6] S.-W. Chiu and K. K. Parhi, "Long Polynomial Modular Multiplication Using Low-Complexity Number Theoretic Transform [Lecture Notes]," *IEEE Signal Processing Magazine*, vol. 41, no. 1, pp. 92–102, 2024.

[7] P. Barrett, "Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor," in *Conference on the Theory and Application of Cryptographic Techniques*. Springer, 1986, pp. 311–323.

[8] M. Ayinala, M. Brown, and K. K. Parhi, "Pipelined parallel FFT architectures via folding transformation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 6, pp. 1068–1081, 2011.

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

[10] K. 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.