# A High-Level Synthesis Approach to the Software/Hardware Codesign of NTT-based Post-Quantum Cryptography Algorithms

Duc Tri Nguyen, Viet B. Dang and Kris Gaj
*Department of Electrical and Computer Engineering, George Mason University*, Fairfax, VA, U.S.A.
{dnguye69, vdang6, kgaj}@gmu.edu

***Abstract*—Due to an emerging threat of quantum computing, one of the major challenges facing the cryptographic community is a timely transition from traditional public-key cryptosystems, such as RSA and Elliptic Curve Cryptography, to a new class of algorithms, collectively referred to as Post-Quantum Cryptography (PQC). Several promising candidates for a new PQC standard can have their software and hardware implementations accelerated using the Number Theoretic Transform (NTT). In this paper, we present an improved hardware architecture for NTT, with the hardware-friendly modular reduction, and demonstrate that this architecture can be efficiently implemented in hardware using High-Level Synthesis (HLS). The novel feature of the proposed architecture is an original memory write-back scheme, which assists in preparing coefficients for performing later NTT stages, saving memory storage used for precomputed constants. Our design is the most efficient for the case when $log_2N$ is even. The latency of our proposed architecture is approximately equal to $(Nlog_2N + 3N)/4$ clock cycles. As a proof of concept, we implemented the NTT operation for several parameter sets used in the PQC algorithms NewHope, FALCON, qTESLA, and CRYSTALS-DILITHIUM.**

***Index Terms*—Post-Quantum cryptography, Number Theoretic Transform, Lattice-based, High-Level Synthesis, programmable logic**

## I. Introduction

A threat of quantum computers triggered an effort aimed at designing a new class of cryptographic algorithms, collectively referred to as Post-Quantum Cryptography (PQC) [1]. These algorithms have two common features: a) there are no known attacks capable of breaking these cryptosystems, even assuming the availability of full-scale quantum computers, b) all PQC algorithms can be implemented using traditional computing platforms, based on standard semiconductor technology, such as microprocessors and FPGAs. In the standardization process currently run by the National Institute of Standards and Technology (NIST), 26 candidates remain and need to be evaluated from the point of view of their hardware efficiency [1]. These candidates represent 5 major families: lattice-based, code-based, hash-based, isogeny-based, and multivariate. A large number of candidates and a high complexity of the majority of them make hardware benchmarking extremely challenging. In order to mitigate these difficulties, using High-Level Synthesis (HLS) has been proposed [2]. To make this approach fair, the performance and resource utilization of the HLS-based designs should be comparable to that obtained using the traditional Register-Transfer Level (RTL) approach, or at least the expected overhead should be comparable across all compared algorithms.

Several lattice-based candidates in Round 2 of the NIST PQC standardization process use operations in the polynomial ring $R_q = Z_q[x]/(x^N + 1)$, with $N$ being a power of 2. A multiplication in this ring can be sped up by using the Number Theoretic Transform (NTT). The input and output of a transform is a polynomial with $N$ coefficients, which are integers in the range of 0 to $q$-1. All internal operations of NTT are performed modulo a prime $q$.

## II. Background

### *A. Number Theoretic Transform*

Multiplications in $R_q$ can be performed efficiently in software and hardware using the Number Theoretic Transform (NTT). NTT is very similar to the Fast Fourier Transform (FFT), and can be obtained from FFT by replacing $e^{-2\pi ik/N}$ with the respective powers of the $N$th primitive root of unity in $Z_q$, denoted by $\omega_N$, and doing transformation over the field $Z_q$, instead of over the field of complex numbers [3]. By using NTT and Inverse NTT (INTT), a multiplication in $R_q$ can be computed as:

$$c = INTT^{-1}(NTT(a) * NTT(b))$$

for $a, b, c \in R_q$, with $q \equiv 1 \bmod 2N$. The complexity of this computation is $O(NlogN)$.

If $\psi = \sqrt{\omega_N} \bmod q$ exists, then the multiplication in $R_q$ does not require padding input with $N$ zero coefficients, and performing NTT calculations with $2N$ points. Instead, the coefficients of the input polynomial must be multiplied by $\psi^i \bmod q$ before the Forward NTT. After the Inverse NTT, the coefficients of the output polynomial must be multiplied by $\psi^{-i} \bmod q$. In this paper, we divide the polynomial multiplication into 5 steps:

1) PSIS_MUL: Coefficient-wise multiplication by $\psi^i$
2) NTT: Forward NTT transform

TABLE I

Selected NTT-based Round 2 PQC candidates investigated in this study. Major parameters of NTT: $N$ and $q$. Parameters $k$ and $m$ of the Longa-Naehrig modular reduction, and the number of adders required to perform functions K-RED and K-RED-2x.

| **Candidate(s)** | **Security Category** | $N$ | $q$ | $log_2 q$ | $k$ | $2^m$ | **#K-RED adders** | **#K-RED-2x adders** |
|---|---|---|---|---|---|---|---|---|
| **NewHope** and **FALCON** | 5 | 1024 | $12,289$ | 14 | $2+1$ | $2^{12}$ | 2 | 5 |
| **qTESLA** | 3 | 1024 | $8,404,993$ | 24 | $2^9+1$ | $2^{14}$ | 2 | 5 |
| **CRYSTALS-Dilithium** | 1, 2, 3 | 256 | $8,380,417$ | 23 | $2^{10}-1$ | $2^{13}$ | 2 | 5 |

3) COEF_MUL: Coeffcient-wise multiplication of two polynomials
4) INTT: Inverse NTT transform
5) IPSIS_MUL: Coefficient-wise multiplication by $\psi^{-i}$.

Furthermore, we treat PSIS_MUL, COEF_MUL, and IPSIS_MUL as operations performed in the MUL mode, and NTT and INTT as operations performed in the NTT mode.

### B. Modular Reduction

An efficient method of performing modular reduction during the NTT computations was proposed in [4]. This method takes advantage of the special form of $q$: $q = k \cdot 2^m + 1$, where $k$ is odd and $k < 2^m$. In Table I, we demonstrate that for at least four Round 2 PQC candidates, there exist parameter sets with $q$ meeting the aforementioned condition. In such cases, the modular reduction $C \bmod q$ can be replaced by the simpler function K-RED($C$) [4]. This function returns an integer $D$, such that $D \equiv kC \bmod q$ and $|D| < q + |C|/2^m$. Although this function alone does not completely reduce the value of $C$, it is still referred to as a reduction because it brings $D$ to the close vicinity of the desired range. For longer computations, additional reductions may need to be applied to a limited number of intermediate values. In this case, as an optimization, two successive reductions K-RED can be merged into a single reduction K-RED-2x($C$) [4].

For the majority of candidates, KRED and KRED2x can be rewritten, using the shift and add operations, by taking advantage of the special form of $k$. For example, for CRYSTALS-DILITHIUM, $k = 2^{10} - 1$, and thus
K-RED($C$) $= kC_0 - C_1 = (2^{10} - 1)C_0 - C_1 =$
$= (C_0 \ll 10) - C_0 - C_1$, and
K-RED-2x($C$) $= k^2 C_0 - kC_1 + C_2 =$
$(C_0 \ll 20) - (C_0 \ll 11) + C_0 - (C_1 \ll 10) + C_1 + C_2$.
For all algorithms shown in Table I, K-RED and K-RED-2x use two and five adders, respectively.

## III. Previous Work

The previous comparable hardware implementations of NTT targeted the PQC schemes such as Ring-LWE [5], [6], [7], as well as the corresponding lattice-based signatures schemes [8]. The most recent efforts aimed specifically at the efficient implementations of the PQC Key Encapsulation Mechanism (KEM) NewHope [9] [10].

## IV. Methodology

### A. Hardware Architecture

In the context of NTT, a Radix-R means that R coefficients are loaded and computed at the same time. We have simplified the Radix-4 hardware architecture presented in [6] for the case of even values of $log_2 N$, used by the selected variants (security categories) of the PQC Round 2 candidates summarized in Table I. The simplified block diagram is shown in Fig. 1, and the corresponding algorithm is shown as Algorithm 1. Similarly to the hardware design from [6], the proposed architecture has the 2x2 butterfly structure, meaning, it consists of two layers of NTT, with two butterfly units per each layer. The square box $m_1$ represents the reduction function KRED, and the square box $m_2$ the reduction function KRED2x.

When s is set to 0, the circuit operates in the MUL mode, defined in Section II-A. First, four coefficients are loaded from the RAM at the top of the block diagram and placed in the registers located in the diagram immediately below the RAM. We have named four computational lines shown in the block diagram A, B, C, and D, respectively. These lines correspond to four sets of indices: $4i$, $4i+1$, $4i+2$, and $4i+3$ with $i \in [0, 1, 2, \dots N/4)$.

The number of cascade registers placed in front of each SIPO differs by one, in order to prevent the case in which two SIPOs are full and attempt to write results back to the RAM in the same clock cycle. When the following outputs of SIPOs: $A_{1st}$, $C_{2nd}$, $B_{3rd}$, and $D_{4th}$ are available, they are concatenated and written back to the RAM. The same happens every clock cycle afterwards until the computations are completed.

When s is set to 1, the circuit operates in the NTT mode. Four coefficient go through the 2x2 butterfly structure, and the results are written to the respective SIPOs, and then to the RAM, in the way similar as for the case of s=0. The reduction function KRED is applied after each addition and subtraction performed in layer 1, not followed by a multiplication. It is also applied after loading coefficients to the first registers in lines C and D. Coefficients in lines B and D are multiplied by $\omega_N^i$ or $\omega_N^{-i}$, depending on whether the circuit computes NTT or INTT. Afterward, the coefficients in lines B and C are swapped, in order to perform computations of the next NTT layer. When $SIPO_A$ is full, four coefficients available at the outputs $A_{1st}$, $C_{2nd}$, $B_{3rd}$, $D_{4th}$ are concatenated, and stored back

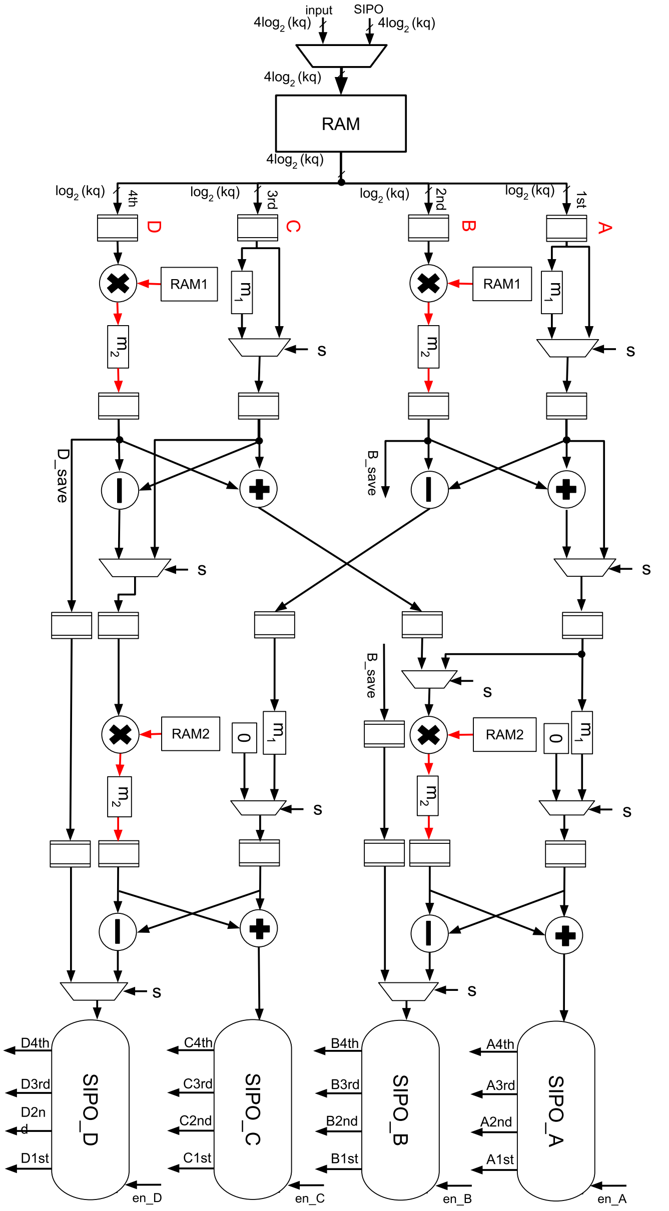

Fig. 1. Block diagram of the proposed hardware architecture to perform fast polynomial multiplication using NTT.

to the RAM. After one clock cycle, the same happens with results accumulated in $SIPO_C$, and then $SIPO_B$ and $SIPO_D$.

The precomputed values of all constants are stored in the dual-port memories RAM1 and RAM2. The total amount of memory required is $2.5N$ in RAM1 and $3N$ in RAM2, the bitwidth is equal to $log_2 q$.

The approximate total number of clock cycles, for the case when $log_2(N)$ is even, is: PSIS_MUL: $N/4$; NTT: $N/4 \cdot log_2 N/2$; COEF_MUL: $N/4$; INTT: $N/4 \cdot log_2 N/2$; IPSIS_MUL: $N/4$; Total: $(N log_2 N + 3N)/4$. The red lines in Fig. 1 represent four likely critical paths in this design.

**Algorithm 1** When s = 1 NTT operation, else coefficient-wise multiplication

```
for $(l = 1; l < log_2(N); l = l + 2)$ do
    $m = 1 \ll (l - 1)$;
    $\omega_{idx1} = N \gg l; \omega_{idx2} = N \gg (l + 1)$;
    $NTT_{idx} = NTT_{idx3} = 0; NTT_{idx4} = N/4$;
    for $(j = 0; j < m; j = j + 1)$ do
        for $(k = 0; k < N/4; k = k + m)$ do
            $b = k + j$;
            $A = RAM[b]_{63...48}; B = RAM[b]_{47...32}$;
            $C = RAM[b]_{31...16}; D = RAM[b]_{15...0}$;
            if $s = 0$ then
                $idx1 = (b \ll 2) + 1; idx2 = (b \ll 2) + 3$;
            else
                $idx1 = idx2 = NTT_{idx}$;
            end if
            if $s \neq 0$ then
                $A =$ K-RED$(A); C =$ K-RED$(C)$;
            end if
            $B =$ K-RED-2x$(B * RAM1[idx1])$;
            $D =$ K-RED-2x$(D * RAM1[idx2])$;
            $B_{save} = B; D_{save} = D$;
            $A' = A + B; B' = C + D$;
            $C' = A - B; D' = C - D$;
            $A = A'; B = B'; C = C'; D = D'$;
            if $s = 0$ then
                $B = A; D = C$;
            end if
            if $s = 0$ then
                $idx3 = (b \ll 2); idx4 = (b \ll 2) + 2$;
            else
                $idx3 = NTT_{idx3}; idx4 = NTT_{idx4}$;
            end if
            $B =$ K-RED-2x$(B * RAM2[idx3])$;
            $D =$ K-RED-2x$(D * RAM2[idx4])$;
            if $s = 0$ then
                $A = 0; C = 0$;
            else
                $A =$ K-RED$(A); C =$ K-RED$(C)$;
            end if
            $A = A + B; B = A - B$;
            $C = C + D; D = C - D$;
            if $s = 0$ then
                $B = B_{save}; D = D_{save}$;
            end if
            $SIPO_A \leftarrow A; SIPO_B \leftarrow B$;
            $SIPO_C \leftarrow C; SIPO_D \leftarrow D$;
        end for
        $NTT_{idx} = NTT_{idx} + \omega_{idx1}$;
        $NTT_{idx3} = NTT_{idx3} + \omega_{idx2}$;
        $NTT_{idx4} = NTT_{idx4} + \omega_{idx2}$;
    end for
end for
```

TABLE II
RESULTS OF THE IMPLEMENTATIONS OF THE NTT UNIT FOR SELECTED ROUND 2 PQC CANDIDATES, USING ZYNQ ULTRASCALE+.

| Algorithms | N | q | | DSPs | BRAM 36K | LUT | FF | Slices | Max Freq (MHz) | Clock Cycles | Latency ($\mu$s) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **NewHope & FALCON** | 1024 | 12,289 | RTL | 4 | 5 | 849 | 802 | 163 | 476 | 1,324 | 2.78 |
| | | | HLS | 4 | 5 | 865 | 822 | 175 | 455 | 1,324 | 2.91 |
| | | | HLS/RTL | 1.0 | 1.0 | 1.02 | 1.02 | 1.07 | 0.96 | 1.0 | 1.05 |
| **qTESLA** | 1024 | 8,404,993 | RTL | 8 | 8 | 1,286 | 2,160 | 283 | 467 | 1,363 | 2.92 |
| | | | HLS | 8 | 8 | 1,939 | 3,423 | 453 | 455 | 1,363 | 2.99 |
| | | | HLS/RTL | 1.0 | 1.0 | 1.51 | 1.58 | 1.60 | 0.97 | 1.0 | 1.03 |
| **CRYSTALS-DILITHIUM** | 256 | 8,380,417 | RTL | 8 | 2 | 1,899 | 2,041 | 392 | 445 | 294 | 0.66 |
| | | | HLS | 8 | 2 | 1,977 | 2,329 | 401 | 434 | 294 | 0.67 |
| | | | HLS/RTL | 1.0 | 1.0 | 1.04 | 1.14 | 1.02 | 0.97 | 1.0 | 1.02 |

TABLE III
COMPARISON WITH THE RESULTS OF THE PREVIOUS IMPLEMENTATION OF THE NTT UNIT [10], FOR ZYNQ-7000. $N = 1024, q = 12,289$.

| | DSPs | BRAM 18K | Slices | LUTs | FFs | Max Freq (MHz) | Clock cycles | Latency ($\mu$s) |
|---|---|---|---|---|---|---|---|---|
| Kuo et al. [10] | 8 | 10 | N/A | 2,832 | 1,381 | 150 | 2,616 | 17.44 |
| This work RTL | 4 | 10 | 357 | 898 | 1,117 | 188 | 2,032 | 10.80 |
| This work HLS | 4 | 10 | 811 | 1,521 | 2,695 | 180 | 2,032 | 11.29 |
| HLS/RTL | 1.00 | 1.00 | 2.27 | 1.69 | 2.41 | 0.96 | 1.00 | 1.04 |

## V. RESULTS

The maximum clock frequency and resource utilization have been generated by performing logic synthesis, placing, and routing using Vivado 2018.3. Our target platform is Zynq UltraScale+ MPSoC. The choice of this platform is consistent with our plan to extend our hardware accelerators for NTT into full software/hardware codesigns of the entire PQC candidates. Our results, shown in Table II, indicate that the latency in clock cycles, the number of DSPs, and the number of BRAMs is almost identical for the RTL- and HLS-based implementations. The penalty for using HLS in terms of the maximum clock frequency and latency varies between 2% and 5%. The overhead in terms of the resource utilization is below 14% for all investigated candidates, except qTESLA.

The latency of the entire polynomial multiplication, assuming that one operand is already in the NTT domain as a result of precomputations, was reported in [6] as $(Nlog_2N/4 + N/2)$ clock cycles. [5] does not make the same assumption and reports the latency equal to $(3Nlog_2N + N)/4$ clock cycles. The latency of our design, making the same assumption as [6], is $(Nlog_2N + 3N)/4$, and thus is very similar to that reported in [6], and significantly smaller than that reported in [5].

All results in Table III were generated using Zynq-7000. The reported latency of the design by Kuo et al. [10] does not include the Order-Reverse step, which typically contributes 40% of latency, while the designs presented in this paper include the latency of the Reordering step. Thus, it is clear that both the RTL and HLS designs presented in this paper outperform the design by Kuo et al. in terms of all performance metrics (except the number of FFs for the HLS design).

## REFERENCES

[1] "Post-Quantum Cryptography Standardization," https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization, 2019.

[2] F. Farahmand, V. B. Dang, D. T. Nguyen, and K. Gaj, "Evaluating the Potential for Hardware Acceleration of Four NTRU-Based Key Encapsulation Mechanisms Using Software/Hardware Codesign," in *10th International Conference on Post-Quantum Cryptography, PQCrypto 2019*, ser. LNCS. Chongqing, China: Springer, May 2019.

[3] J. M. Pollard, "The Fast Fourier Transform in a Finite Field," *Mathematics of Computation*, vol. 25, no. 114, p. 10, Apr. 1971.

[4] P. Longa, M. Naehrig, P. Longa, and M. Naehrig, "Speeding up the Number Theoretic Transform for Faster Ideal Lattice-Based Cryptography," in *Cryptology and Network Security - CANS 2016*, vol. 10052. Cham: Springer International Publishing, 2016, pp. 124–139.

[5] D. D. Chen, N. Mentens, F. Vercauteren, S. S. Roy, R. C. C. Cheung, D. Pao, and I. Verbauwhede, "High-Speed Polynomial Multiplication Architecture for Ring-LWE and SHE Cryptosystems," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 1, pp. 157–166, Jan. 2015.

[6] C. Du, G. Bai, and X. Wu, "High-Speed Polynomial Multiplier Architecture for Ring-LWE Based Public Key Cryptosystems," in *Proceedings of the 26th Edition on Great Lakes Symposium on VLSI - GLSVLSI '16*. Boston, Massachusetts, USA: ACM Press, 2016, pp. 9–14.

[7] C. P. Renteria-Mejia and J. Velasco-Medina, "High-Throughput Ring-LWE Cryptoprocessors," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 8, pp. 2332–2345, Aug. 2017.

[8] T. Guneysu, V. Lyubashevsky, and T. Poppelmann, "Lattice-Based Signatures: Optimization and Implementation on Reconfigurable Hardware," *IEEE Transactions on Computers*, vol. 64, no. 7, pp. 1954–1967, Jul. 2015.

[9] T. Oder and T. Guneysu, "Implementing the NewHope-Simple Key Exchange on Low-Cost FPGAs," in *LATINCRYPT 2017*, Havana, Cuba, Sep. 2017.

[10] P.-C. Kuo, W.-D. Li, Y.-W. Chen, Y.-C. Hsu, B.-Y. Peng, C.-M. Cheng, and B.-Y. Yang, "High Performance Post-Quantum Key Exchange on FPGAs," Cryptology ePrint Archive 2017/690, Feb. 2018.