Design of pipelined NTT-based polynomial multiplier for Post-Quantum Cryptography CRYSTALS-Kyber

Abstract – In this paper, we present a high-speed and pipelined fully hardware polynomial multiplier design based on number theoretic transform (NTT). NIST post-quantum cryptography standardization round 3 announced CRYSTALS-Kyber as one of the finalists. As a lattice-based cryptography scheme, CRYSTALS-Kyber relies heavily on polynomial multiplication efficiency. Our work centers around designing and optimizing the polynomial multiplier architecture for accelerating hardware implementation of CRYSTALS-Kyber. This includes modifying the modular arithmetic modules, memory and data processing sequences. As a result, our designed achieved 231 MHz Fmax synthesized on Intel FPGA Cyclone V with Quartus, a 15% improvement in speed compared to similar work. Resources utilization through combinational logic path re-balance allowed us to efficiently pipelining between hardware modules.

I. Introduction

The National Institute of Standards and Technology (NIST) announced Post-Quantum Cryptography Standardization process since 2016 [1]. The goal of the standardization is to establish new cryptographic system for both classical computers and quantum computers in the future. Research and development progress in Post-Quantum Cryptography improved significantly through recent years. In 2020, third round candidates were announced, with 4 out of 5 finalists for Public-key Encryption and Key-establishment Algorithms are lattice-based cryptography scheme. Lattice-based cryptography, together with Learning with Errors problem (LWE), are proven to be safe-to-use under worst-case scenario and offer good trade-off between efficiency and security. CRYSTALS-Kyber is one of the five finalists in round 3.

CRYSTALS-Kyber is a lattice-based IND-CCA2-secure key-encapsulation mechanism (KEM), based on Module-LWE problem [2]. It also has a digital signature sibling, called CRYSTALS-Dilithium [3]. Kyber requires heavy computational effort, mostly as multiplication of polynomials over a constant-size polynomial ring. This scheme over "module lattices" offers good trade-off between efficiency and security. The key generation, encryption and decryption process however could account for a large proportion of computing power and clock cycles on microprocessors. Kyber describe a technique called Number Theoretic Transform (NTT), which reduce the complexity of polynomial multiplication from $O(n^2)$ to O(nlogn). While the scheme and technique could be parallel computed, the pure software implementation does not fully utilize the ability.

Many lattice-based cryptographical scheme are implemented on software as a proof of concept and utility, but the wide implementation of them is more likely to be processed on co-hardware and software or fully hardware solutions. Many PQCs candidates and projects have their research design on a FPGA, ASIC or co-processor for ARM, RISC V. One of the earliest full hardware implementations of PQC [3] is for Round5, this includes a hardware design of Keccak, AES-GCM and Round5 algorithm. The simplicity from Round5 modular advantages is used efficiently on hardware. In [4], a similar full hardware design of CRYSTALS-Kyber is presented, which speed-up the performance 129 times compare to Cortex-M4 processor implementation [5]. In [6], the first side-channel attack protected design of hardware CRYSTALS-Kyber is shown with impressive performance. Zhao et al. [7] analyze the software code to optimize their design, which yields positive results. Other implementation of Kyber varies from hardware and software processor hybrids [8] [9] [10] to pure hardware [5] [6] [11] [12]. More recent research on CRYSTALS-Kyber hardware implementation optimizes mostly on its polynomial multiplication process, which is also the purpose of our research. [13] present a novel modular technique called K2-RED for their NTT structure, which improve modular performance. A different modular technique is used in [14],

modified with pre-computed constants. Zhang et al. [15] present a ping-pong memory access scheme to efficiently accessing the RAM. Our proposed design improves in modular reduction technique and resource efficiency compared to [13], [14], [15].

In this paper, we present a hardware NTT-based polynomial multiplier optimized for CRYSTALS-Kyber PQC scheme. The architecture includes a uniform NTT/Invert NTT butterfly unit with improved modular functions, pre-computed parameters and a dual memory accessing sequence which speed-up the computing process. The combinational circuit are pipelined down to 2 logic levels to maximize timing result, further increase the performance of the multiplier. Our design is simulated with Intel ModelSim and synthesized with Intel Quartus for FPGA Intel Cyclone V 5CSXFC6D6F31C6.

The rest of the paper is organized as followed. Section II explains the preliminaries for Number Theoretic Transform (NTT) and CRYSTALS-Kyber. In section III, we present the hardware design for Kyber NTT-based polynomial multiplier. We compare and discuss the synthesis and simulation result in section IV and concludes the paper in section V.

II. Preliminaries

In this section, we briefly describe CRYSTALS-Kyber scheme and related mathematical information.

1. CRYSTALS-Kyber scheme

CRYSTALS-Kyber is based on the module-lattice learning-with-errors problem (MLWE [16]). The detail of Kyber could be found in its updated specification [17]. Kyber basically has two parts: a chosen-plaintext attack (CPA) secure public key encryption (PKE) or CPAPKE, a chosen-ciphertext attack (CCA) secure key encapsulation mechanism (KEM) or CCAKEM. The CPAPKE is included inside CCAKEM as a mandatory step for ciphertext and key generation. While CPAPKE has three different procedures (key generation, encrypt and decrypt), all three steps require polynomial multiplication based on NTT. NTT is included in the definition of Kyber and even the parameters of Kyber versions are tweaked to be calculated efficiently with NTT. The parameters of each Kyber version are shown in the table below.

Table 1 Parameters for each Kyber version

Version	n	k	q	η_1	η_2	(d_u,d_v)	δ
Kyber512	256	2	3329	3	2	(10,4)	2^{-139}
Kyber768	256	3	3329	2	2	(10,4)	2^{-164}
Kyber1024	256	4	3329	2	2	(11,5)	2^{-174}

Kyber has a prime *q* chosen for efficient NTT calculation, *n* bit length, k as a scaling factor for security level. The remaining parameters are chosen for balancing security, failure probability and efficiency.

2. Kyber polynomial multiplication

Kyber CPAPKE steps need multiplications in the polynomial ring $Z_q[X]/(X^n+1)$ domain. The detail of Kyber multiplication could also be found in [17]. The important factors of Kyber multiplication are number theoretic transform (NTT) and bit order. We adopt and modified the NTT/inverse-NTT general structure on programmable hardware from [18] mathematical background and suggestion for Kyber structure. With *br* is a bit-reversed function, $\hat{h}$ as $NTT(h)$ and w as primitive n-th root of unity, polynomial multiplication in Kyber (as $\hat{f} * \hat{g} = \hat{h}$) has 128 products as function below.

$$\hat{h}_{2i} + \hat{h}_{2i+1}X = (\hat{f}_{2i}\hat{g}_{2i} + \hat{f}_{2i+1}\hat{g}_{2i+1}w_n^{2br_7(i)+1}) + (\hat{f}_{2i}\hat{g}_{2i+1} + \hat{f}_{2i+1}\hat{g}_{2i})X$$

NTT in Kyber could be calculated with the butterflies of fast Fourier transform [19] [20], where NTT is implemented with Cooley-Tuckey (CT) butterfly and INTT is implemented with Gentleman-Sande (GS) butterfly. CT and GS butterfly unit are efficient method to design NTT for programmable hardware. The general process diagram for these butterfly units is depicted in figure 1.

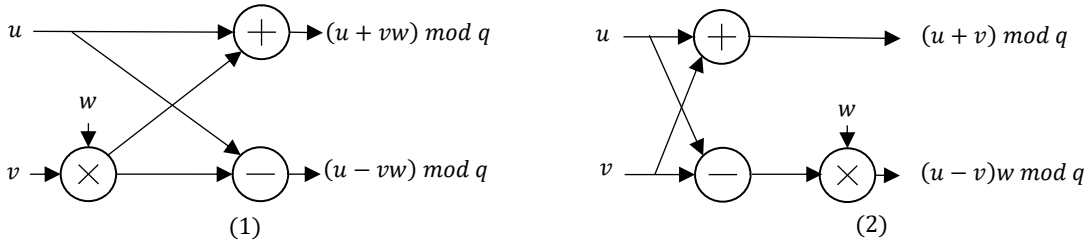

Figure 1 Number Theoretic Transform butterfly units. (1) Cooley - Tukey Butterfly. (2) Gentleman - Sande Butterfly

By pre-processing *w*, NTT algorithm could be modified to be more suitable for hardware processing. Algorithm 1 shows the iterative NTT function.

Algorithm 1 Iterative-NTT
Input: **Output:**

III. The proposed design

1. Modular reduction

We present an improved version from KRED, K-RED-2x [21] and K^2-RED [5] modular design with [22] mathematical background. KRED is similar to Montgomery modular reduction, but designed to work for a special form of modulus:

$$modulus\ q = \ k * 2^m + 1$$

With Kyber modulus prime $q = 3329$, k = 13 and m = 8. Using KRED, K-RED-2x on an input number C gives the result in the form of $k * C\ mod\ q$. Using K^2-RED produces the result at $k^2 * C\ mod\ q$, which explained in [5] to be more memory efficient and suitable for 12-bit modulus q. The modular unit is implemented after the multiplication step in the butterfly unit. To complement for k^2 multiple part in the result, pre-processing is applied to w to $w' = \ (k^{-2} * w)\ mod\ q$. The methods of KRED and K^2-RED are both implied to have cases where they overflow and does not give exact match to the correct result.

By simulating and testing on Intel ModelSim, we mapped out the cases where the original K^2-RED failed to give the correct result. We modify the sequence and give the algorithm an additional step to eliminate the result failure due to negative binaries adding. Another conditional collection is also added to the end result. The resources increase is negligible, while we could maintain the result accuracy. Our Exact-KRED is shown in algorithm 2.

Algorithm 2 Exact-KRED
Input: q modulus = 3329, binary number C = $(C_{23}, \dots, C_1, C_0)$, k = 13, m = 8 **Output: $S = \ (k^2 * C)\ mod\ q$**

1: Cl <= zero-extended($C_7, \dots, C_1, C_0$) to 16-bit
2: Ch<=($C_{23}, \dots, C_9, C_8$)
3: C' <= ((Cl<<3) – Ch) + (Cl<<2+Cl)
4: Cl'<= zero-extended($C'_7, \dots, C'_1, C'_0$) to 12-bit
5: Ch'<=signed-extended($C'_{23}, \dots, C'_9, C'_8$) to 12-bit
6: C'' <= ((Cl'<<3) – Ch') + (Cl'<<2+Cl')
7: If C'' >= q
S <= C'' + q
Else
S <= C''
8: return S

The hardware structure for Exact-KRED is presented in figure 2. By carefully pipeline the steps, we are able to keep the complex combinational logic down to 2 logic level, which improves the overall speed of the design.

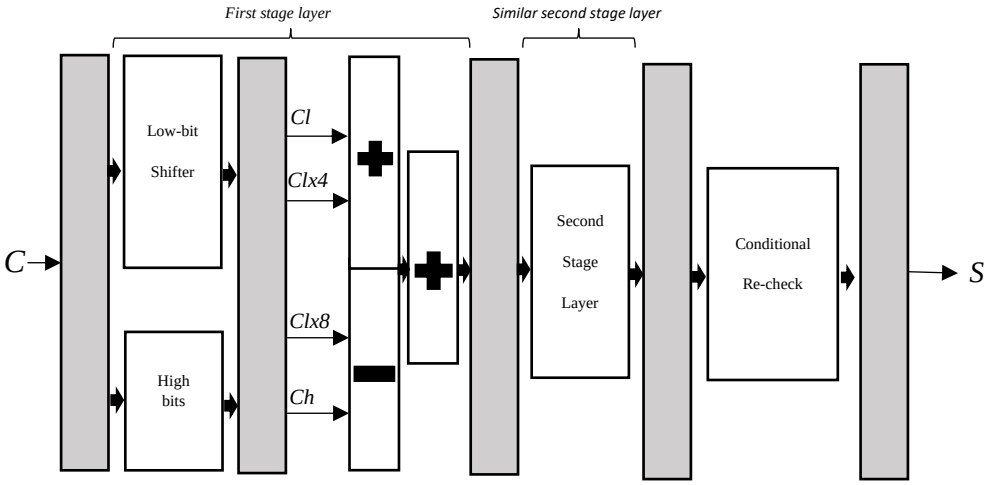

Figure 2 Exact-KRED hardware structure

2. NTT/INTT Polynomial multiplication for Kyber

To calculate NTT/INTT, the main component is the set of butterfly unit. From figure 1 process diagram, designing the hardware butterfly unit in CT/GS form is straightforward. The process would cost two arithmetic modular adders, one DSP multiplier and one Exact-KRED modular reduction unit. Detailed and pipelined structure of the butterfly unit is given in figure 3. The DSP multiplier and Exact-KRED unit have 7 cycles of delay. The path delay blocks pipeline the data and it is adjusted according to the total delay. The total delay for each CT/GS operation on the butterfly unit is 9 cycles, with pipelined inputs and output. Sel signal is used to select the mode of operation. Modular addition and subtraction with the two modular adders does not affect much on speed performance due to the small 12-bit prime *q*.

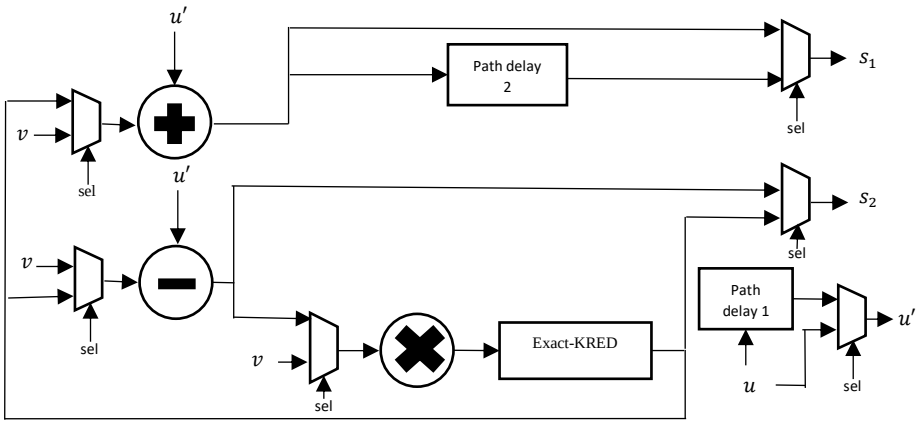

Figure 3 Butterfly unit

Our hardware polynomial multiplication design utilizes two sets of memory. One RAM for the NTT/INTT operands and three ROMs. The RAM uses Intel Cyclone V M10K which has a simple dual-port mode, allowing access for two different memory address simultaneously [22]. With dual RAM reading, our main state machine is able to feed two sets of butterfly unit at one operation. The computed data is feedback through the parameterized channel system back to the RAM. Every operation, two 24-bit data from each RAM is input into the butterfly unit set. The ROMs also feed the butterfly units with corresponding factor w' for each operation, the value is pre-computed. The general structure is shown in figure 4. The INTT operation of the system is shown in algorithm 3.

Figure 4 Hardware structure of NTT-based Polynomial Multiplier for Kyber

Algorithm 3 Inverse NTT algorithm with GS Butterfly
Input:

Output:

IV. The results

The proposed hardware design and its module is synthesized with Intel Quartus Prime 18.1 Lite. The selected chip is Intel Cyclone V 5CSXFC6D6F31C6 which is readily available at our lab.

1. Exact-KRED modular reduction
2. NTT-based polynomial multiplication core

V. Conclusion

We proposed the hardware design of an NTT-based polynomial multiplication core dedicated for CRYSTALS-Kyber family of post-quantum Cryptography. The design includes improved modular reduction hardware, uniform butterfly units, dual access memory. The design is deeply pipelined and we manually rebalanced logic level to optimize for the performance. We believe lattice-based cryptography operations on hardware has more rooms for further improvement in the future when it is further understood. It is an interesting and important topic for standardization and researches.

VI. References.

[1] NIST: Post-Quantum Cryptography Standardization https://csrc.nist.gov/Projects/post-quantum-cryptography.

[2] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, and Damien Stehlé. CRYSTALS – Kyber: a CCA-secure module-lattice-based KEM. In 2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018. IEEE, 2018. To appear. https://eprint.iacr.org/2017/634. 4, 11, 23

[3] Andrzejczak, Michal, Farnoud Farahmand, and Kris Gaj. "Full Hardware Implementation of the Post-Quantum Public-Key Cryptography Scheme Round5." *2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*. IEEE, 2019.

[4] Huang, Yiming, et al. "A pure hardware implementation of crystals-kyber PQC algorithm through resource reuse." *IEICE Electronics Express* (2020): 17-20200234.

[5] B. Leon, et al.: "Memory-efficient high-speed implementation of Kyber on Cortex-M4," International Conference on Cryptology in Africa (2019) 209-228 (DOI: 10.1007/978-3-030-23696-0_11)

[6] Jati, Arpan, et al. "A Configurable Crystals-Kyber Hardware Implementation with Side-Channel Protection." *Cryptology ePrint Archive* (2021).

[7] Zhao, Yixuan, et al. "Optimization Space Exploration of Hardware Design for CRYSTALS-KYBER." *2020 IEEE 29th Asian Test Symposium (ATS)*. IEEE, 2020.

[8] Albrecht, Martin R., et al. "Implementing RLWE-based schemes using an RSA co-processor." *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2019): 169-208.

[9] Sanal, Pakize, et al. "Kyber on ARM64: Compact Implementations of Kyber on 64-bit ARM Cortex-A Processors."

[10] Seo, Hwa-jeong, et al. "Optimized implementation of scalable multi-precision multiplication method on RISC-V processor for high-speed computation of post-quantum cryptography." *Journal of the Korea Institute of Information Security & Cryptology* 31.3 (2021): 473-480.

[11] Xing, Yufei, and Shuguo Li. "A compact hardware implementation of CCA-secure key exchange mechanism CRYSTALS-KYBER on FPGA." *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2021): 328-356.

[12] Guo, Wenbo, Shuguo Li, and Liang Kong. "An Efficient Implementation of KYBER." *IEEE Transactions on Circuits and Systems II: Express Briefs* (2021).

[13] Bisheh-Niasar, Mojtaba, Reza Azarderakhsh, and Mehran Mozaffari-Kermani. "High-Speed NTT-based Polynomial Multiplication Accelerator for CRYSTALS-Kyber Post-Quantum Cryptography." Cryptol. ePrint Arch., Tech. Rep 563 (2021): 2021.

[14] Yarman, Ferhat, et al. "A hardware accelerator for polynomial multiplication operation of CRYSTALS-KYBER PQC scheme." *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2021.

[15] Zhang, Cong, et al. "Towards Efficient Hardware Implementation of NTT for Kyber on FPGAs." *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2021.

[16] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. Designs, Codes and Cryptography, 75(3):565–599, 2015. https://eprint.iacr.org/2012/090. 4, 12, 19

[17] Avanzi, Roberto, et al. "CRYSTALS-Kyber algorithm specifications and supporting documentation." *NIST PQC Round* 2.4 (2017).

[18] T. Pöppelmann and T. Güneysu, "Towards efficient arithmetic for lattice-based cryptography on reconfigurable hardware," in Progress in Cryptology - LATINCRYPT 2012 - 2nd International Conference on Cryptology and Information Security in Latin America, Santiago, Chile, October 7-10, 2012. Proceedings, pp. 139–158, 2012.

[19] Bisheh-Niasar, Mojtaba, Reza Azarderakhsh, and Mehran Mozaffari-Kermani. "A Monolithic Hardware Implementation of Kyber: Comparing Apples to Apples in PQC Candidates." *International Conference on Cryptology and Information Security in Latin America*. Springer, Cham, 2021.

[20] Becker, Hanno, et al. "Neon NTT: Faster Dilithium, Kyber, and Saber on Cortex-A72 and Apple M1." *Cryptology ePrint Archive* (2021).

[21] Longa, Patrick, and Michael Naehrig. "Speeding up the number theoretic transform for faster ideal lattice-based cryptography." *International Conference on Cryptology and Network Security*. Springer, Cham, 2016.

[22] Intel, "Cyclone V Device Datasheet", CV-51002 datasheet, Nov. 2019