2022 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT) | 978-1-6654-5938-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/DFT56152.2022.9962336

HPMA-NTRU: High-Performance Polynomial Multiplication Accelerator for NTRU

Pengzhou He[1], Yazheng Tu[1], Ayesha Khalid[2], Máire O'Neill[2], and Jiafeng Xie[1]
[1]: Department of Electrical and Computer Engineering, Villanova University, Villanova PA USA 19087
[1]: Email: {phe,ytu1,jiafeng.xie}@villanova.edu

[2]: Centre for Secure Information Technologies, Queen's University Belfast, Belfast, U.K.
[2]: Email: a.khalid@qub.ac.uk, m.oneill@ecit.qub.ac.uk

***Abstract*—Along the rapid development of large-scale quantum computers, post-quantum cryptography (PQC) has drawn significant attention from research community recently as it is proven that the existing public-key cryptosystems are vulnerable to the quantum attacks. Meanwhile, the recent trend in the PQC field has gradually switched to the hardware acceleration aspect. Following this trend, this work presents a novel implementation of a High-performance Polynomial Multiplication hardware Accelerator for NTRU (HPMA-NTRU) under different parameter settings, one of the lattice-based PQC algorithm that is currently under the consideration by the National Institute of Standards and Technology (NIST) PQC standardization process. In total, we have carried out three layers of efforts to obtain the proposed work. First of all, we have proposed a new schoolbook algorithm based strategy to derive the desired polynomial multiplication algorithm for NTRU. Then, we have mapped the algorithm to build a high-performance polynomial multiplication hardware accelerator and have extended this hardware accelerator to different parameter settings with proper adjustment. Finally, through a series of complexity analysis and implementation based comparison, we have shown that the proposed hardware accelerator obtains better area-time complexities than the state-of-the-art one. The outcome of this work is important and will impact the ongoing NIST PQC standardization process and can be deployed further to construct efficient NTRU cryptoprocessors.**

***Index Terms*—High-performance, NTRU, polynomial multiplication hardware accelerator, post-quantum cryptography (PQC)**

I. Introduction

With the rapid progression in quantum computing, it is proven that most of the existing public key cryptographic algorithms will no longer be secure as they can be solved by large-scale quantum computers executing Shor's algorithm [1]–[5], i.e., the widely used Rivest Shamir Adleman (RSA) and Elliptic-Curve Cryptography (ECC) will no longer be secure within 10-15 years [3], [6], [7]. In response, the National Institute of Standards and Technology (NIST) has started the PQC standardization process and the current third round standardization has selected three lattice-based public-key encryption scheme including NTRU [6]–[8].

NTRU is built on the lattice-based N-th degree truncated polynomial ring problem, and is the merger of two PQC candidates NTRUEncrypt and NTRU-HRSS-KEM [8] with different parameter settings. Since the announcement of the NIST third round PQC finalists, several noticeable work has been undertaken on NTRU, including cryptanalysis, implementation techniques, and side-channel attacks [8]–[10].

As the NIST PQC standardization process has progressed, efficient implementations of PQC schemes, undertaken on various operating platforms have been reported, especially FPGA-based hardware acceleration [3], [11]–[27]. Although NTRU is selected as one of the NIST PQC third round standardization finalists [7], very limited work is available in the literature for its hardware implementations. Partly because of its complicated arithmetic operation, partly because a major arithmetic operation (e.g., polynomial multiplication) has relatively large parameter setting and its efficient implementation becomes a bottleneck on the hardware platform [8]. For instance, for one family of NTRU, NTRU-HPS, requires a polynomial multiplication with degree of $N = 821$ with 12-bit point-wise multipliers, which is challenging to be implemented on the hardware platform with low-complexity. Consequently, NTRU generally has a slower operation than the other two lattice-based NIST public-key finalists [7], [8] and hence its high-speed implementations with low-complexity cost is lacking and much needed in the literature.

Based on these considerations, we propose in this work, a novel high-performance polynomial multiplication accelerator for NTRU (HPMA-NTRU), targeting different parameter settings. Particularly, we notice that the polynomial multiplication based on schoolbook algorithm can lead to a high-speed implementation for NTRU yet with optimized area usage. Towards this optimization in mind, we have made three layers of coherent interdependent efforts to carry out the proposed work. Major contributions of this work include:

- We have derived the schoolbook polynomial multiplication algorithm into a high-speed operation format for accelerator implementation.
- We have carried out a series of optimization techniques to map the algorithm into a high-performance hardware accelerator with efficient resource usage, HPMA-NTRU, which is also extended to different parameter settings.
- We have conducted implementation analysis and comparison to demonstrate the efficiency of the proposed HPMA-NTRU.

Overall, the proposed HPMA-NTRU features high-performance operation with efficiency in area-complexity, and hence is

978-1-6654-5938-9/22/$31.00 ©2022 IEEE

desirable for high-speed NTRU cryptoprocessor deployment.

The rest of this paper is organized as follows. The preliminary knowledge is introduced in Section II. The formulation of the proposed high-speed algorithm is detailed presented in Section III. The proposed hardware polynomial multiplication accelerator for NTRU is provided in Section IV. Implementation analysis and comparison are presented in Section V. Finally, conclusions are given in Section VI.

II. Preliminaries

In this section, we briefly give the introduction of NTRU and its polynomial multiplication. Interested readers can refer to the original material of [8] for details.

NTRU. NTRU is a structured lattice-based key encapsulation mechanism (KEM) whose security is built on the Nth degree truncated polynomial ring (NTRU) problem, which was originally defined in 1998 [8], [28]. Due to its long and established history, The current NIST third round submission of NTRU is a merger of previous submissions NTRUEncrypt and NTRU-HRSS-KEM. NTRU is well recognized in the research community as it is widely analyzed. The current version of NTRU satisfies the perfect correctness to build the KEM scheme. The "only disadvantage" is that it requires a relatively slower operation time than the other two NIST lattice-based finalists. Consequently, not many high-performance implementations of NTRU scheme are reported, particularly for its key component, the polynomial multiplication over rings [7].

Parameter Setting The current NIST third round submission of NTRU is composed of two families of parameter settings, i.e., NTRU-HPS and NTRU-HRSS. According to the submission recommendation [8], one can choose different parameters as desired. Meanwhile, the NIST third round submission of NTRU has its own recommended parameters for both families [8], three parameter sets for NTRU-HPS and one parameter set for NTRU-HRSS. From the recommended parameter setting, one can see that the polynomial multiplication of $n = 821$ and $q = 2^{12}$ mostly can cover almost all the related parameter sets for both NTRU families (expect that NTRU-HRSS requires $q = 2^{13}$, but with a smaller n). For the sake of simplicity, we just use the polynomial multiplication of $n = 821$ and $q = 2^{12}$ as the representative parameter set for NTRU.

Polynomial Multiplication for NTRU and Prior Works. Polynomial multiplication over rings is the key arithmetic operation of NTRU. On the other side, however, very few hardware implementations for polynomial multiplication of NTRU are available in the literature. A recent design of [29] has presented a hardware design for the combined parameter sets. A very recent report of [30] have presented several hardware designs but only focus on small q and thus is not comparable to the proposed design here.

Remark. Note that there also exist other previous designs of [31]–[36], where the targeted n and q are also not reach the parameter settings of NIST submission NTRU [8]. Hence, these reported cannot considered as NTRU related standard designs and we just mention them here. Besides that, the three mentioned designs [31]–[33] are non-constant time in execution and hence are vulnerable to timing attacks.

III. HPMA-NTRU: Mathematical Formulation

This section presents the mathematical background of the proposed algorithm for polynomial multiplication of NTRU. For simplicity of discussion, we just use the parameter set of $n = 821$ and $q = 2^{12}$ to derive the algorithm, which can be easily extended to other parameter sets.

Definition 1. Define two polynomials $A = \sum_{i=0}^{n-1} a_i x^i$ and $B = \sum_{i=0}^{n-1} b_i x^i$, where both a_i and b_i are 12-bit integer in $\mathcal{Z}_q$. Meanwhile, we also define $C = \sum_{i=0}^{n-1} c_i x^i$ (c_i is also 12-bit integer in $\mathcal{Z}_q$) as the product of A and B. Thus, we have

$$C = AB \bmod f(x), \tag{1}$$

where $f(x) = x^n - 1$. Then, for high-speed operation, we can have

$$\begin{aligned} C =& A(b_0 + b_1 x + \cdots + b_{n-1}x^{n-1}) \bmod f(x), \\ =& Ab_0 + Axb_1 + \cdots + Ax^{n-1}b_{n-1} \bmod f(x), \end{aligned} \tag{2}$$

where we can also define

$$\begin{aligned} Ax^0 =& A = A^{(0)}, \\ Ax =& A^{(1)}, \\ \cdots\ \cdots\ \cdots& \\ Ax^{n-1} =& A^{(n-1)}, \end{aligned} \tag{3}$$

which can be substituted into (2) to have

$$\begin{aligned} C =& A^{(0)}b_0 + A^{(1)}b_1 + \cdots + A^{(n-1)}b_{n-1}, \\ =& \sum_{i=0}^{n-1} A^{(i)}b_i, \end{aligned} \tag{4}$$

where the polynomial multiplication becomes the accumulation of a polynomial ($A^{(i)}$) with the corresponding b_i. In summary, we can have the algorithmic operation as

Algorithm 1: Proposed high-performance polynomial multiplication algorithmic operation for NTRU

Input : A and B are integer polynomials; // where a_i and b_i are 12-bit coefficients according to NTRU parameter setup;
Output: $C = AB \bmod (x^n + 1)$; // where c_i is 12-bit coefficient over $\mathbb{Z}_q$;
Initialization step
1 Make ready the inputs A and B;
2 $\overline{C} = \sum_{i=0}^{n-1} \overline{c_i} x^i = 0$;
Main step
3 **for** $i = 0$ *to* $n-1$ **do**
4 **for** $j = 0$ *to* $n-1$ **do**
5 $\sum_{j=0}^{n-1} \overline{c_j} x^j = \sum_{i=0}^{n-1} A^{(i)} b_i$; // $\overline{c_0}$ to $\overline{c_{n-1}}$ are executed in parallel
6 **end**
7 **end**
8 $C = \sum_{i=0}^{n-1} c_i x^i = \overline{C}$;
Final step
9 Serially deliver all the coefficients of output C;

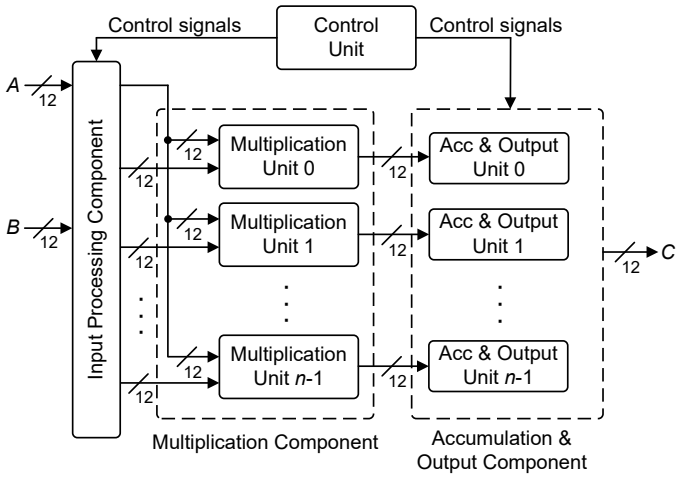

Fig. 1. The proposed hardware accelerator for NTRU: HPMA-NTRU.

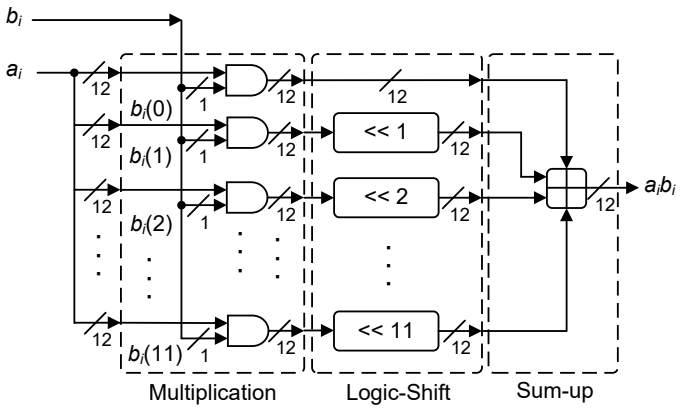

Fig. 2. The internal structure of the point-wise multiplier, where $q = 2^{12}$ and it could be extended to other values of q for NTRU.

where also exist the operation of deriving $A^{(i+1)}$ from $A^{(i)}$ $(0 \leq i \leq n-2)$ during the operations in Lines 3-7 of Algorithm 1. For this operation, we can have

$$\begin{aligned} A^{(i)} =& Ax^i = (a_0 + a_1x + \cdots + a_{n-1}x^{n-1})x^i, \\ =& a_0x^i + a_1x^{i+1} + \cdots + a_{n-1}x^{n+i-1}, \\ =& a_{n-i} + a_{n-i+1}x + \cdots + a_{n-1}x^{i+1} \\ & + a_0x^i + \cdots + a_{n-i-1}x^{n-1}, \end{aligned} \tag{5}$$

where $x^n \equiv 1$ is being substituted as $x^n - 1 \equiv 0$. From (5), we can further have

$$\begin{aligned} & A^{(i+1)} = xA^{(i)} \bmod f(x) \\ & = x(a_{n-i} + a_{n-i+1}x + \cdots + a_{n-1}x^{i+1} \\ & + a_0x^i + \cdots + a_{n-i-1}x^{n-1}) \bmod f(x) \\ & = a_{n-i-1} + a_{n-i}x + \cdots + a_{n-1}x^{i+2} \\ & + a_0x^{i+1} + \cdots + a_{n-i-2}x^{n-1}, \end{aligned} \tag{6}$$

where $x^n \equiv 1$ is being substituted again for modulo reduction. Comparing eq. (6) with eq. (5), we can see that the coefficients of $A^{(i)}$ are circularly-shifted (while the coefficients' signs remain the same) to produce $A^{(i+1)}$, which is recursively repeated among the rest of i $(0 \leq i \leq n-2)$.

IV. HPMA-NTRU: Proposed Hardware Accelerator

This section covers the architectural details of the proposed HPMA-NTRU hardware. As shown in Fig. 1, the proposed accelerator contains four major components, i.e., the Input Processing Component, the Multiplication Component, the Accumulation & Output Component, and the Control Unit. Details of each individual component will be given in the following part of this section.

Input Processing Component. The Input Processing Component is responsible for loading all the coefficients and delivering them to the Multiplication Component in a correct form, as shown in Fig. 1. It consists of a serial-in-serial-out shift-register and a special serial-in-parallel-out shift-register. The polynomial multiplication is carried out in three major phases, loading, multiplication and output phases (as explained in the subsequent sections). During the loading phase, the two shift-registers load one 12-bit coefficient every cycle. During the multiplication phase, the serial-in-serial-out shift-register will deliver one coefficient to the Multiplication Component at each cycle. Meanwhile the other shift-register will just output all the coefficients at the same time. Note that this shift-register needs to shift circularly in order to deliver correct coefficients, based on the operation of eq. (6). There is a 2-to-1 MUX at the input of this serial-in-parallel-out shift-register, which will close the loop when all the initial coefficients are loaded into corresponding registers, in order to realize the required circularly-shifting for the related coefficient according to eq. (6). The Input Processing Component takes n cycles to load all the coefficients, and then requires n cycles to deliver all the necessary coefficients to the Multiplication Component.

Multiplication Component. This Component executes the point-wise multiplications between related coefficients according to Algorithm 1. As is shown in Fig. 1, the Multiplication Component contains n parallel multiplication Units, with each unit deals with one 12-bit point-wise multiplication. Each unit takes the coefficient from the serial-in-parallel-out shift-register and one of the n coefficients delivered from the other shift-register as its another input, then calculates the product of the two coefficients and output it to the corresponding Acc & Output Unit connected to it. The detailed internal structure for the point-wise multiplication is shown in Fig. 2: first, it is designed that the multiplier multiplies one coefficient by each bit of the other coefficient, which is achieved by using AND gates; then the multiplier logic-shifts the products by proper bits the products; after logic-shifting, the multiplier sums up all the generated numbers by using a $\log_2 q$-bit adder, followed by a modulo operation in order to keep the length of the input and output the same. It takes one cycle for the multiplication unit to execute the whole process of one point-wise multiplication.

Accumulation & Output Component. As shown in Fig. 3, the Accumulation & Output Component is a custom designed shift-register combined with accumulators, consisting of n registers, n MUXes, and n 12-bit adders. This component

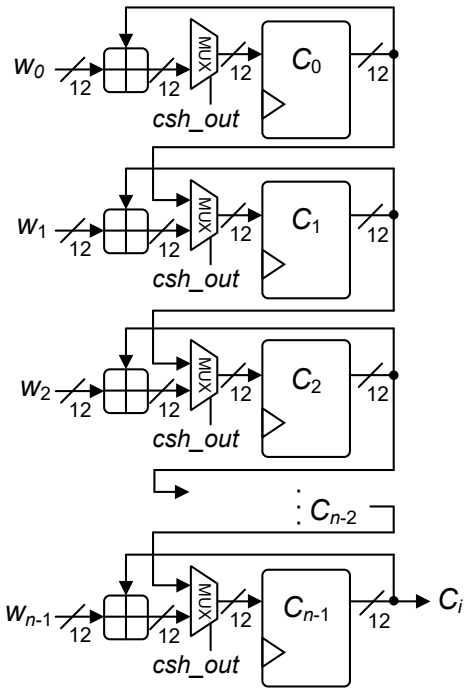

Fig. 3. The internal details of the Accumulation & Output Component, where $q = 2^{12}$ and could be extended to other values of q.

operates as both the accumulator of the products of the coefficients and the output buffer. The switch between the two functions is controlled by the MUXes connected to the registers and the control signal csh_out. Wwhen $csh\_out = 0$, the MUXes will select the accumulation of the product of the coefficients (W_i) and the value stored in the register, which ensures all the products of the coefficients are stored and accumulated in the registers. When $csh\_out = 1$, the MUXes will select the output of another register, which transforms the Accumulation & Output Component into a shift-register so that the final result can be delivered out serially.

Control Unit. The Control Unit generates control signals for all operations, timed throughout the multiplication process such as "load", "reset", "shift", and "output". To implement the function, We use a finite state machine (FSM) to deliver the proper control signals (along with some extra logic gates) to the corresponding components. In the proposed FSM, there are five consecutive states, namely "reset" (1 cycle) "load" (n cycles), "multiplication" (n cycles), "output" (n cycles), and "done" (1 cycle), where n is the polynomial degree. The proposed FSM starts with the "reset" state that clears all registers and all the shift-registers are disabled. When the "reset" signal is de-asserted, it moves into the "load" state, which lasts for n cycles. During this phase, the two shift-registers in the Input Processing Component are fully loaded with the coefficients of A and B while the registers in the Acc&Component are still being cleared. Next, the control unit automatically moves into the "multiplication" state in which the multiplication is carried out for n cycles. When calculation is done, the control unit goes to the "output" state, where the result is delivered out

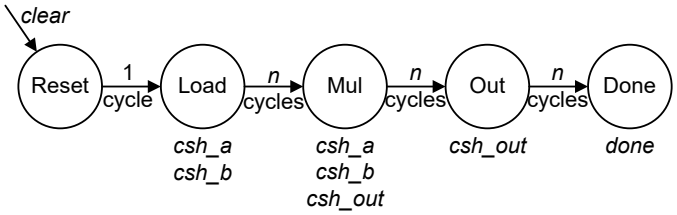

Fig. 4. Various states of the FSM for the Control Unit, generating control signals in each state.

serially. Finally, the HPMA-NTRU moves into the "done" state indicating that the whole polynomial multiplication process is completed.

Extension to the Other Parameter Settings. The proposed accelerator can be easily extended to other parameter settings of NTRU, based on the adjustment on the point-wise multiplier's bit-width (as well as the internal structure), related coefficients' bit-width $\log_2 q$, and the polynomial degree n.

V. Complexity Analysis and Comparison

Complexity Analysis. Overall, the proposed HPMA-NTRU contains two n-length 12-bit circular shift registers (CSRs, that can be other bit-width as well), n point-wise multipliers, and n accumulation units to calculate and deliver the output coefficients in a serial fashion. The whole accelerator has a computation latency of n cycles and the critical-path is mostly determined by the addition propagation period of an adder and a point-wise multiplier, as shown in Fig. 1. The complexities of the proposed HPMA-NTRU under different parameter settings are determined mostly by the specific n and q values. Finally, a control unit is also needed for the proposed accelerator.

Implementation and Comparison. The proposed hardware structure was manually coded in VHDL and simulated via Mentor Graphics ModelSim tool to verify the correctness of its functionality. When implementing the accelerator, the proposed HPMA-NTRU was implemented on the targeted device of AMD-Xilinx FPGA UltraScale+ XCZU9EG-FFVB1156-2 and Zynq-7000 xc7z100ffg1156-2 devices through Xilinx Vivado 2020.2 toolchain. The post place and route (post-PAR) results, including the area consumption such as LUT, FF, and CLB, timing information like number of computation cycles and maximum frequency (MHz), and dynamic power consumption (Watts), are listed in Table I. We have also picked all four parameter sets according to two families of NTRU [8], i.e., ($n = 509, q = 2^{11}$), ($n = 677, q = 2^{11}$), ($n = 871, q = 2^{12}$), and ($n = 701, q = 2^{13}$), for the proposed accelerator to be implemented on the mentioned FPGA devices to obtain related performance results. To further demonstrate the efficiency of the proposed accelerator, we have also listed the state-of-the-art design [29] in the same table. Note that the early designs of [31]–[36] used different parameters other than the current NTRU [8], and hence we do not include them in the comparison. Meanwhile, the very recent design of [30] used smaller parameters and we also do not include it here.

Discussions. As we can see from Table I, a larger n results in a higher area consumption: the number of LUT, FF, and CLB

TABLE I
COMPARISON OF DIFFERENT POLYNOMIAL MULTIPLIER HARDWARE ACCELERATORS FOR NTRU

NTRU Scheme	FPGA Device	Latency Cycles[1]	Freq. (MHz)	LUTs	FFs	CLBs	Power (W)	ADP
[29] $n = 821$, $q = 2^{12}$	Zynq-7000	821	70	56,218	21,406	-	-	659,356
$n = 509$, $q = 2^{11}$	Ultrascale+	509	254	36,976	12,126	5,648	0.110	74,098
$n = 677$, $q = 2^{11}$	Ultrascale+	677	248	49,137	16,075	7,745	0.147	132,155
$n = 821$, $q = 2^{12}$	Ultrascale+	821	236	72,430	21,172	11,300	0.205	251,970
$n = 701$, $q = 2^{13}$	Ultrascale+	701	223	71,028	18,994	11,661	0.232	223,276
$n = 509$, $q = 2^{11}$	Zynq-7000	509	223	36,999	12,132	6,219	0.239	84,450
$n = 677$, $q = 2^{11}$	Zynq-7000	677	226	49,261	16,080	8,148	0.123	147,565
$n = 821$, $q = 2^{12}$	Zynq-7000	821	210	71,990	21,202	11,647	0.210	281,447
$n = 701$, $q = 2^{13}$	Zynq-7000	701	201	71,321	19,554	20,270	0.282	248,736

[1]: Latency cycle does not include the input loading and output delivery.
Ultrascale+: Ultrascale+ XCZU9EG-FFVB1156-2 device.
Zynq-7000: Zynq-7000 xc7z100ffg1156-2 device.
ADP: ADP (area-delay product). area: #LUT. Delay: cycles×(1/Freq.).
The design of [29] does not report the CLB and power and is a combined design with another parameter set.

become larger as n increases. The area usage increases with n linearly, with the highest LUT = 71,979 for $n = 821$ and the lowest LUT = 44,429 when $n = 509$. This is because the proposed design is computing the point-wise multiplications in parallel, i.e., n determines the number of point-wise multipliers and registers involved within the CSRs in the Input processing Component. Thus, more registers and calculating components, such as adders and point-wise multiplier, are needed to execute the multiplication and store corresponding data. Similarly, as $\log_2 q$ increases, the area consumption becomes greater as well. Since the length of involved coefficient becomes larger and more registers, point-wise multipliers, and adders are required for the calculation.

As mentioned in the last section, the time complexity of the proposed HPMA-NTRU is strictly linear to the value of n, i.e., n cycles of multiplication during the whole process. On the other hand, the maximum frequency of the proposed design decreases as n increases, e.g., Freq.$= 254$ when $n = 509$ and Freq.$= 217$ when $n = 821$. This is also because of the parallel computation: when n increases, more multiplication and accumulation units are involved in parallel, which leads to a longer critical-path (going from the input buffer to the accumulator) for the signals to propagate. As a result, the signals take a longer time when propagating along the path and thus the frequency will drop. Enhancing frequency by shortening the critical-path in the proposed design can be seen as our future work.

Meanwhile, considering the comparison with the state-of-the-art work of [29], the proposed accelerator involves smaller or equal latency cycles but with much better operational frequency. Meanwhile, when considering the resource usage, e.g., the number of LUTs, the proposed accelerator involves comparably number of LUTs when comparing with [29]. As the proposed accelerator has much better operational frequency than [29] (though with slightly larger usage on the LUTs), the proposed HPMA-NTRU involves at least 57.3% less ADP on the Zynq-7000 FPGA device, as seen from Table I.

Side-Channel Attacks Consideration. The proposed HPMA-NTRU has constant time operation, and hence is resistant to the timing attacks [37]. While power side-channel attacks are out of the scope of this paper, but we can include that in our future works. Meanwhile, the countermeasures can also included in the further explorations.

Other Aspects of Future Works. Though the proposed HPMA-NTRU is efficient in implementation, future works can be focused more on: (i) develop efficient polynomial multiplication algorithms for complexity reduction, possibly following the trend in the binary field polynomial multiplier [38]–[41]; (ii) design full cryptoprocessors for NTRU, based on different parameter settings and security levels; (iii) develop novel hardware design methods, as presented in other PQC algorithms [19], [42], [43].

VI. CONCLUSION

We have introduced a new hardware accelerator for polynomial multiplication of NTRU in this paper, HPMA-NTRU, aiming at enhancing the high-speed operation performance of the polynomial multiplication of NTRU on the hardware platform. We have carried out three layers of efforts to finalize the proposed work. Mathematical formulation is provided first to lay the algorithmic foundation. Then, the proposed hardware accelerator is detailed presented. Lastly, complexity analysis, implementation, and comparison are also provided to show the efficiency of the proposed work. The proposed accelerator covers different parameter settings of NTRU and is expected to be useful for the ongoing NIST PQC standardization process.

VII. ACKNOWLEDGEMENT

The first two authors, Pengzhou He and Yazheng Tu, contributed equally. The work of Jiafeng Xie was supported by NIST-60NANB20D203 and in part by NSF SaTC-2020625. Ayesha Khalid, Máire O'Neill were funded by EPSRC Quantum Communications Hub (EP/T001011/1).

REFERENCES

[1] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th annual symposium on foundations of computer science*. Ieee, 1994, pp. 124–134.

[2] K. Basu, D. Soni, M. Nabeel, and R. Karri, "Nist post-quantum cryptography-a hardware evaluation study," *Cryptology ePrint Archive*, 2019.

[3] J. Xie, K. Basu, K. Gaj, and U. Guin, "Special session: The recent advance in hardware implementation of post-quantum cryptography," in *2020 IEEE 38th VLSI Test Symposium (VTS)*. IEEE, 2020, pp. 1–10.

[4] D. Soni, K. Basu, M. Nabeel, and R. Karri, "A hardware evaluation study of nist post-quantum cryptographic signature schemes," in *Second PQC Standardization Conference*. NIST, 2019.

[5] D. Soni, M. Nabeel, K. Basu, and R. Karri, "Power, area, speed, and security (pass) trade-offs of nist pqc signature candidates using a c to asic design flow," in *2019 IEEE 37th International Conference on Computer Design (ICCD)*. IEEE, 2019, pp. 337–340.

[6] P. quantum cryptography round 3 submissions. https://csrc.nist.gov/projects/post-quantumcryptography/round 3-submissions.

[7] G. Alagic, J. Alperin-Sheriff, D. Apon, D. Cooper, Q. Dang, J. Kelsey, Y.-K. Liu, C. Miller, D. Moody, R. Peralta *et al.*, "Status report on the second round of the NIST post-quantum cryptography standardization process," *US Department of Commerce, NIST*, 2020.

[8] C. Chen *et al.*, "Ntru," *NIST third round PQC submissions*, 2020.

[9] J. H. Silverman and W. Whyte, "Timing attacks on ntruencrypt via variation in the number of hash calls," in *Cryptographers' Track at the RSA Conference*. Springer, 2007, pp. 208–224.

[10] E. Carter, P. He, and J. Xie, "High-performance polynomial multiplication hardware accelerators for kem saber and ntru," *Cryptology ePrint Archive*, 2022.

[11] W. Liu, S. Fan, A. Khalid, C. Rafferty, and M. O'Neill, "Optimized schoolbook polynomial multiplication for compact lattice-based cryptography on fpga," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 10, pp. 2459–2463, 2019.

[12] W. Tan, A. Wang, Y. Lao, X. Zhang, and K. K. Parhi, "Low-latency VLSI architectures for modular polynomial multiplication via fast filtering and applications to lattice-based cryptography," *arXiv preprint arXiv:2110.12127*, 2021.

[13] M. Imran, F. Almeida, J. Raik, A. Basso, S. S. Roy, and S. Pagliarini, "Design space exploration of Saber in 65nm ASIC," in *Proceedings of the 5th Workshop on Attacks and Solutions in Hardware Security*, 2021, pp. 85–90.

[14] M. Bisheh-Niasar, R. Azarderakhsh, and M. Mozaffari-Kermani, "Instruction-set accelerated implementation of crystals-kyber," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 11, pp. 4648–4659, 2021.

[15] J. Xie, P. He, and C.-Y. Lee, "CROP: FPGA implementation of high-performance polynomial multiplication in saber kem based on novel cyclic-row oriented processing strategy," in *2021 IEEE 39th International Conference on Computer Design (ICCD)*. IEEE, 2021, pp. 130–137.

[16] T. Bao, P. He, and J. Xie, "Systolic acceleration of polynomial multiplication for KEM saber and binary ring-LWE post-quantum cryptography," in *2022 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2022, pp. 1–4.

[17] V. B. Dang, K. Mohajerani, and K. Gaj, "High-speed hardware architectures and FPGA benchmarking of crystals-kyber, ntru, and saber," *Cryptology ePrint Archive*, 2021.

[18] A. C. Mert, D. Jacquemin, A. Das, D. Matthews, S. Ghosh, S. S. Roy *et al.*, "A unified cryptoprocessor for lattice-based signature and key-exchange," *Cryptology ePrint Archive*, 2021.

[19] S. S. Roy and A. Basso, "High-speed instruction-set coprocessor for lattice-based key encapsulation mechanism: Saber in hardware," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 443–466, 2020.

[20] Y. Zhu, M. Zhu, B. Yang, W. Zhu, C. Deng, C. Chen, S. Wei, and L. Liu, "LWRpro: An energy-efficient configurable crypto-processor for Module-LWR," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 3, pp. 1146–1159, 2021.

[21] Y. Zhang, C. Wang, D. E. S. Kundi, A. Khalid, M. O'Neill, and W. Liu, "An efficient and parallel r-lwe cryptoprocessor," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 5, pp. 886–890, 2020.

[22] P. He, C.-Y. Lee, and J. Xie, "Compact coprocessor for KEM Saber: Novel scalable matrix originated processing," *The NIST Third Standardization Conference*, pp. 1–16, 2021.

[23] Y. Tu, P. He, C.-Y. Lee, D. Chasaki, and J. Xie, "Hardware implementation of high-performance polynomial multiplication for kem saber," *IEEE International Symposium on Circuits and Systems-2022 (ISCAS'22)*, pp. 1–5, 2022.

[24] B. J. Lucas, A. Alwan, M. Murzello, Y. Tu, P. He, A. J. Schwartz, D. Guevara, U. Guin, K. Juretus, and J. Xie, "Lightweight hardware implementation of binary ring-lwe pqc accelerator," *IEEE Computer Architecture Letters*, 2022.

[25] J. L. Imaña, P. He, T. Bao, Y. Tu, and J. Xie, "Efficient hardware arithmetic for inverted binary ring-lwe based post-quantum cryptography," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2022.

[26] Y. Zhang, C. Wang, A. Khalid, M. O'Neill, W. Liu *et al.*, "Ultra high-speed polynomial multiplications for lattice-based cryptography on FPGAs," *IEEE Transactions on Emerging Topics in Computing*, no. 01, pp. 1–1, 2022.

[27] S. Bian, M. Hiromoto, and T. Sato, "Filianore: Better multiplier architectures for lwe-based post-quantum key exchange," in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1–6.

[28] J. Hoffstein, D. Lieman, J. Pipher, and J. H. Silverman, "Ntru: A public key cryptosystem," *NTRU Cryptosystems, Inc.(www. ntru. com)*, 1999.

[29] Z. Qin, R. Tong, X. Wu, G. Bai, L. Wu, and L. Su, "A compact full hardware implementation of pqc algorithm ntru," in *2021 International Conference on Communications, Information System and Computer Engineering (CISCE)*. IEEE, 2021, pp. 792–797.

[30] S. Khan, W.-K. Lee, A. Khalid, A. Majeed, and S. O. Hwang, "Area-optimized constant-time hardware implementation for polynomial multiplication," *IEEE Embedded Systems Letters*, 2022.

[31] A. A. Kamal and A. M. Youssef, "An fpga implementation of the NTRUEncrypt cryptosystem," in *2009 International Conference on Microelectronics-ICM*. IEEE, 2009, pp. 209–212.

[32] B. Liu and H. Wu, "Efficient multiplication architecture over truncated polynomial ring for NTRUEncrypt system," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2016, pp. 1174–1177.

[33] E. Camacho-Ruiz, S. Sánchez-Solano, P. Brox, and M. C. Martínez-Rodríguez, "Timing-optimized hardware implementation to accelerate polynomial multiplication in the NTRU algorithm," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 17, no. 3, pp. 1–16, 2021.

[34] K. Braun, T. Fritzmann, G. Maringer, T. Schamberger, and J. Sepúlveda, "Secure and compact full NTRU hardware implementation," in *2018 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*. IEEE, 2018, pp. 89–94.

[35] B. Liu and H. Wu, "Efficient architecture and implementation for NTRUEncrypt system," in *2015 IEEE 58th International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2015, pp. 1–4.

[36] F. Farahmand, M. U. Sharif, K. Briggs, and K. Gaj, "A high-speed constant-time hardware implementation of NTRUEncrypt SVES," in *2018 International Conference on Field-Programmable Technology (FPT)*. IEEE, 2018, pp. 190–197.

[37] T. Schneider, A. Moradi, and T. Güneysu, "Parti–towards combined hardware countermeasures against side-channel and fault-injection attacks," in *Annual International Cryptology Conference*. Springer, 2016, pp. 302–332.

[38] J. Xie, P. K. Meher, M. Sun, Y. Li, B. Zeng, and Z. H. Mao, "Efficient FPGA implementation of low-complexity systolic Karatsuba multiplier over $GF(2^m)$ based on NIST polynomials," *IEEE Trans. Circuits and Systems-I*, vol. 64, no. 7, pp. 1815–1825, 2017.

[39] J.-S. Pan, C.-Y. Lee, A. Sghaier, M. Zeghid, and J. Xie, "Novel systolization of subquadratic space complexity multipliers based on toeplitz matrix–vector product approach," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 7, pp. 1614–1622, 2019.

[40] C.-Y. Lee and J. Xie, "Efficient subquadratic space complexity digit-serial multipliers over gf (2 m) based on bivariate polynomial basis representation," in *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2020, pp. 253–258.

[41] C. H. Liu, C. Y. Lee, and P. K. Meher, "Efficient digit-serial KA based multiplier over binary extension fields using block recombination approach," *IEEE Trans. Circuits & Systems I*, vol. 62, no. 8, pp. 2044–2051, 2015.

[42] A. Basso and S. S. Roy, "Optimized polynomial multiplier architectures for post-quantum KEM Saber," *Design Automation Conference (DAC)*, pp. 1–6, 2021.

[43] P. He, U. Guin, and J. Xie, "Novel low-complexity polynomial multiplication over hybrid fields for efficient implementation of binary ring-lwe post-quantum cryptography," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 11, no. 2, pp. 383–394, 2021.