Indonesian Journal of Electrical Engineering and Computer Science
Vol. 33, No. 2, February 2024, pp. 795∼803
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v33.i2.pp795-803

Efficient number theoretic transform accelerator for CRYSTALS-Kyber

Toan Nguyen, Hoang Anh, Hung Nguyen, Trang Hoang, Linh Tran
Department of Electronics Engineering, Ho Chi Minh City University of Technology, Vietnam National University Ho Chi Minh City, Ho Chi Minh City, Vietnam

Article Info

Article history:

Received Sep 19, 2023
Revised Nov 26, 2023
Accepted Dec 4, 2023

Keywords:

Accelerator
Cryptography
CRYSTALS-Kyber
FPGA
Hardware implementation
Number theoretic transform

ABSTRACT

The national institute of standards and technology (NIST) has presented its draft of the module-lattice-based key-encapsulation mechanism standard (MLBKEMS), choosing cryptographic suite for algebraic lattices (CRYSTALS)-Kyber as the base encryption. Existing hardware implementations of modern cryptography will need to process the new standard efficiently. The primary process in CRYSTALS-Kyber key-encapsulation mechanism (KEM) is the number theoretic transform (NTT), which requires heavy computing power. This paper contributes an efficient hardware accelerator for NTT and inverse NTT (INTT) by CRYSTAL-Kyber parameters. The proposed design utilizes the K-RED algorithm for reducing polynomial multiplication. It also incorporates the Brent-Kung method for efficient modular addition and subtraction operation with an address generator to control the sequences of computation. On the Xilinx Artix 7 field programmable gate array (FPGA), our design achieves 262 MHz clock speed, utilizing only 1405 LUTs.

Corresponding Author:

Linh Tran
Department of Electronics Engineering, Ho Chi Minh City University of Technology
Vietnam National University Ho Chi Minh City
Ly Thuong Kiet, Ho Chi Minh City, Vietnam
Email: linhtran@hcmut.edu.vn

1. INTRODUCTION

The national institute of standards and technology (NIST) shows their draft of module-lattice-based key-encapsulation mechanism standard (MLBKEMS) on federal information processing standards (FIPS) 203 [1]. The standard specifies a key-encapsulation mechanism (KEM) that uses a module lattice method called ML-KEM, based on cryptographic suite for algebraic lattices (CRYSTALS)-Kyber specification. CRYSTALS-Kyber or Kyber is a module lattice-based KEM, which needs multiplication of polynomials over a polynomial ring [2]. The main bottleneck in most current implementations of CRYSTALS-Kyber is its method of fast multiplication using number theoretic transform (NTT). Software implementation of Kyber would need to iterate many times through the loops of NTT and inverse NTT (INTT), making this the most time-consuming operation [3]. Unlike the elliptic curve cryptography (ECC) or Rivest–Shamir–Adleman (RSA) mechanism, which has a primarily linear computation method, NTT could utilize parallelization in computing to increase throughput.

Current hardware implementations of Kyber and similar lattice-based post quantum cryptography (PQC) attempt to improve the algorithm performance with many methods. Two main approaches to improve NTT performance are improving the butterfly unit structure and the communication between stages of

NTT, which includes fetching data from memory and arranging them in order for the following NTT stages. Zhang *et al.* [3] merge the pre-processing of NTT and post-processing of INTT with an optimized butterfly unit for modular operations on NEWHOPE prime q. Our previous work [4] uses a similar approach for NTT and INTT, with a tweaked modular algorithm for Kyber's prime called Exact KRED and a 2x2 butterfly unit configuration and low-complexity NTT and INTT algorithm. Ye *et al.* [5] improves upon the work of Zhang et al., hard coded the modular reduction algorithm to NEWHOPE prime q. They also introduce a fully pipelined structure between stages of NTT and compute with NTT cores. Bit-Parallel NTT (BP-NTT) [6] architecture uses static random access memory (SRAM) and bit-parallel modular multiplication to improve their NTT process. Barret reduction is a popular modular algorithm for optimizing the butterfly unit [7]. Fritzman and Sepulveda [8] discuss possible side-channel attacks on NTT hardware implementation and propose a low-power design with a single-port RAM. To enhance data throughput, a ping-pong memory access scheme is proposed in [9].

Accelerating hardware algorithm has many applications and applies to different platforms such as Xilinx field programmable gate array (FPGA) and application specific integrated circuit (ASIC) [10]-[13]. Paludo and Sousa [14] integrated their butterfly unit to a 5-stage pipeline linux-ready RISC-V. They rated the performance on both FPGA and 28 ASIC. CoHA-NTT is the first NTT-based polynomial multiplication operations architecture with run-time and compile-time reconfigurability [15]. Geelen *et al.* [16] introduce a new instruction set architecture (ISA) extension to their RISC-V architecture to integrate NTT. Kuang *et al.* [17] vectorized the NTT algorithm to combine it with their RISC-V design better. Chen *et al.* [18] improve the NTT in their processor, taking advantage of dual memory access. The integration of NTT across different platforms to support the new PQC is evolving [19]-[22].

In this work, we design an efficient NTT accelerator for CRYSTALS-Kyber PQC, with unified block memory and a hybrid NTT/INTT algorithm. Our contribution includes:

- A dual-configuration three-stage butterfly unit optimized for efficient modular operation on CRYSTALS-Kyber parameters.
- A low-complexity NTT/INTT hardware architecture with dual butterfly units.
- An improved hybrid NTT/INTT algorithm and address generation using BRAM address sequence to improve overall speed.

The remainder of the document is structured in the following manner: section 2 provides the background information on CRYSTALS-Kyber, NTT, Brent-Kung Adders, and K-RED modular reduction. In section 3, we present our suggested hardware architecture for the NTT Accelerator. Our discoveries and results are detailed in section 4. Finally, section 5 serves as the conclusion for our paper.

2. PRELIMINARIES

2.1. CRYSTALS-Kyber scheme

CRYSTALS-Kyber is part of a cryptographic suite called CRYSTALS. Lattice-based cryptography is grounded in the complexity of solving lattice problems, which are believed to be complicated even for quantum computers [23]. In its specification [2], Kyber provides 3 versions of its KEM: Kyber-512, Kyber-768, and Kyber-1024. They also provide the 90s versions, which replace the Keccak SHA3 implementation with the SHA2 family. The parameters for each version of Kyber are outlined in Table 1, taken from the specifications.

Table 1. The specifications for each version of Kyber with distinct parameters

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}

2.2. Number-theoretic transform

In CRYSTALS-Kyber, Bos *et al.* [2] use NTT for fast multiplications between polynomials. The counterpart of the NTT is referred to as the INTT. With NTT and INTT, the multiplication results of the equation $h = f * g$ could be calculated from (1), which improves the speed of polynomials multiplication [2]. For software implementation and evaluation, Scott [24] show detailed points to adapt from an effective reference C implementation of PQC. We employ the negative wrap convolution (NWC) version of the NTT algorithm. Pöppelmann and Güneysu [25] to prevent the inclusion of zero-padding coefficients in the input

polynomials f and g for NTT and INTT [4]. Based on the previous two algorithms from [4], we re-write a new hybrid NTT/INTT algorithm in Algorithm 1.

$$h = INTT * (NTT(f) * NTT(g)) \tag{1}$$

Algorithm 1 Hybrid NTT and INTT algorithm

Input: $a(x) = a_1, a_2 \dots a_n$ for NTT
Input: $\hat{a}(x) = \hat{a}_1, \hat{a}_2 \dots \hat{a}_n$ for INTT
Input: Pre-computed twiddle factor $\zeta[i] = \gamma^{BitReverse}[i]$
Output: $NTT(a(x))$ for NTT; $INTT(\hat{a}(x))$ for INTT
Intialize $k = 0$ (NTT) or $k = n$ (INTT)
for $m = 0; m < \log_2 n; m++$ **do**
  $len \leftarrow (n/2 >> m) or (1 << m)$
  for $i = 0; i < n; i = j + len$ **do**
    $\omega \leftarrow \zeta[++k]$ (NTT) or $-\zeta[--k]$ (INTT)
    for $j = i; j < i + len; j++$ **do**
      $r_1 \leftarrow a_j + len$ (NTT) or $(\hat{a}_j - \hat{a}_{j+len})$ (INTT)
      $u_1 \leftarrow r_1 * \omega$
      $r_2 \leftarrow u_1$ (NTT) or $\hat{a}_j + len$ (INTT)
      $u_2 \leftarrow a_j + r_2$ (NTT) or $\hat{a_j} + r_2$ (INTT)
      $t_1 \leftarrow u_2$
      $t_2 \leftarrow (a_j - u_1)$ (NTT) or u_1 (INTT)
      a_j (NTT) or $\hat{a}_j$ (INTT) $\leftarrow t_1$
      $a_j + len$ (NTT) or $\hat{a}_j + len$ (INTT) $\leftarrow t_2$
    end for
  end for
end for

Each NTT stage requires a Cooley-Tukey (CT) butterfly unit, and each INTT state requires a gentleman-sande (GS) butterfly unit. Using NWC, the butterfly unit configuration must account for the pre-processing and post-processing accordingly. Figure 1 provides the low complexity butterfly unit diagrams for NWC NTT and INTT [4]. Figure 1(a) is the CT butterfly unit structure for NTT, and Figure 1(b) is the GS butterfly unit structure.

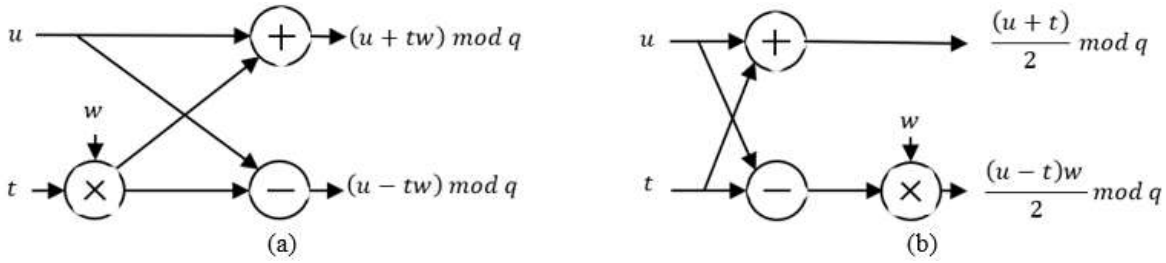

Figure 1. The figure showcases two butterfly units with the reduced complexity algorithm; (a) CT butterfly unit and (b) GS butterfly unit

2.3. Brent-Kung adder

The carry-lookahead adder (CLA) has a parallel prefix adder (PPA) variant called the Brent-Kung adder CLA. We use this adder structure to improve the performance of butterfly unit modular addition and subtraction. In contrast to the Kogge-Stone adder, it has less wiring congestion and is simpler in structure. It was first proposed by Brent and Kung [26] (KSA).

In Brent-Kung adders, the carry is computed simultaneously, dramatically reducing the operation time. Also, the carry has to travel through fewer stages, lowering power consumption. Brent-Kung revolutionized carry generation and propagation by introducing an operator o defined as $(a1, b1)\ o\ (a2, b2) = (a1 \vee (b1 \wedge a2), b1 \wedge b2)$, along with the function $(Gi, Pi) = (g1, p1)$ for $i = 1$; otherwise $(gi, pi)o(Gi - 1, Pi - 1)$

for $i = 2, 3, \ldots n$. The operator $o(Gn, Pn)$, as defined, can be computed in a tree-like structure, similar to Figure 2.

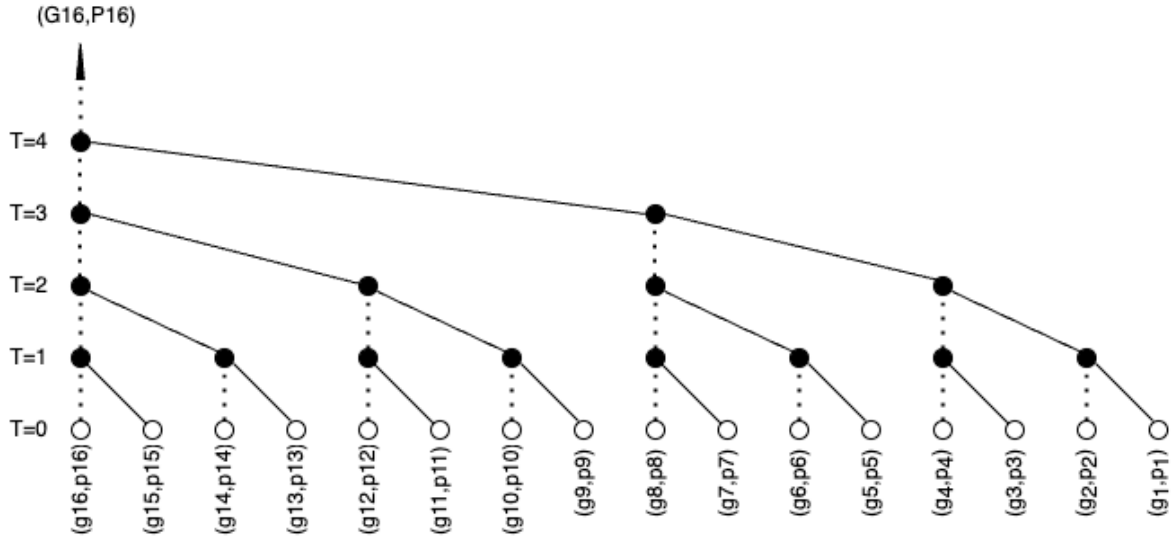

Figure 2. Brent-Kung adder tree-like structure

2.4. K-RED modulo reduction

The K-RED algorithm 2 [3], [27] takes any integer C as input and produces an integer D such that $D \equiv kC \pmod q$, and $|D| < q + |C|/2^m$. While this function doesn't strictly reduce the value of C modulo q," it is termed a reduction because it brings D within the desired range. It's important to note that for $|C| > (2^m/(2^m - 1))q$, we observe $|D| < |K - RED(C)|$, meaning it effectively reduces the size of C. For the CRYSTALS-Kyber parameters, $q = 3329 = 132^8 + 1$. In this context, with $k = 13$, K-RED returns $13C_0 - C_1 \equiv 13C \pmod q$ using the equivalence $132^8 \equiv -1 \pmod q$ [3].

Algorithm 2 KRED (C)

$C_0 \leftarrow C \; mod \; 2^m$
$C_1 \leftarrow C/2^m$
Return

3. HARDWARE IMPLEMENTATION

In Figure 3, the comprehensive hardware architecture for the NTT accelerator is depicted, featuring two butterfly units designed for both NTT and INTT operations. The read-only memory (ROM) stores data and Two dual-port random access memory (RAM) stores polynomials. The controller and address generator define input, NTT, INTT, and output states through an I/O interface. At the outset, the data is stored in the RAM, with polynomial coefficients represented as 16-bit integers within the processor. Next, the address generator module generates the addresses of the data that need to be read from the RAM. These data are then fetched from the RAM and sent to the dual butterfly units for computation. Simultaneously, the address generator also generates the addresses of the corresponding twiddle factors stored in the ROM, which are required by the dual butterfly units. All the necessary data undergo computation, and the resulting values are pushed back to the RAM at the same addresses. Figure 4 shows the controller's state machine, which controls all the processes from input to output. The state machine resets to the reset or idle stage, waiting for the start signal to start input data to RAM, calculate based on the mode selection (NTT or INTT), and output data back to data_out.

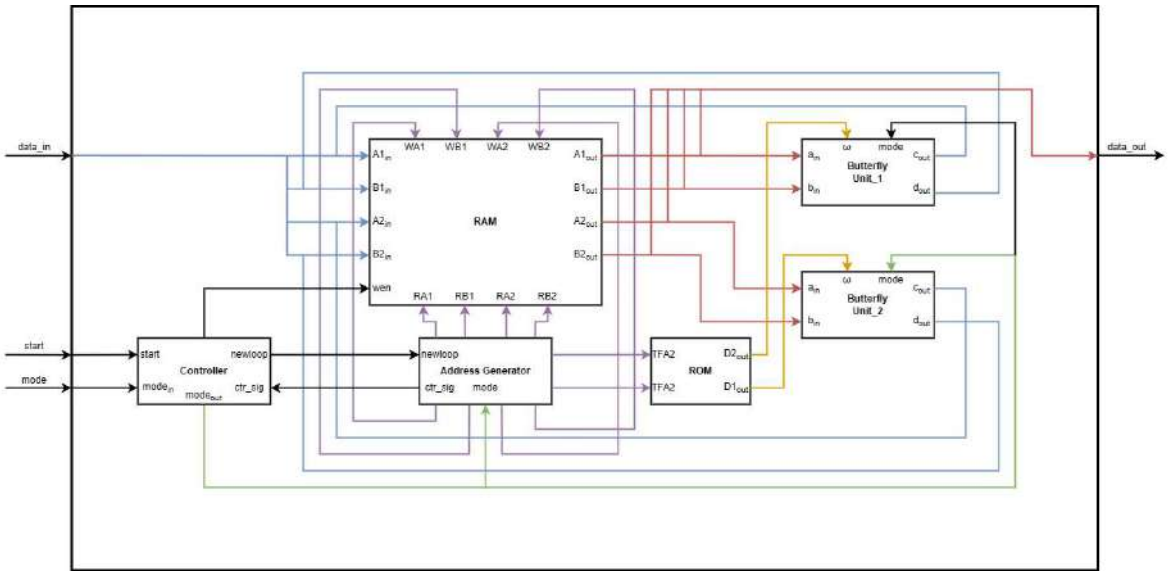

Figure 3. The block diagram of the hardware implementation for NTT accelerator

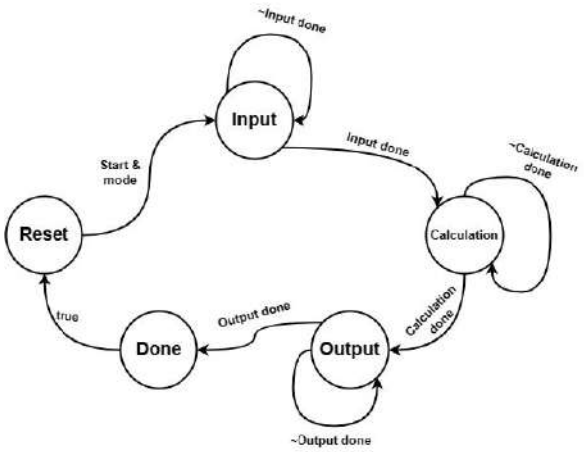

Figure 4. State machine of the NTT accelerator

3.1. The address generator

The address generator module serves as a control and address generator for a specific application, providing the necessary signals and addresses to ensure the data sequence for each cycle in a layer. Each NTT/INTT operation for Kyber $n = 256$ has to go through 8 layers. Each layer has 64 butterfly unit operations for a dual butterfly unit configuration. Based on the mode and the current layer, the module generates specific control signals and calculates address values for different layers of operation. It uses conditional and case statements to determine the behavior based on the mode and layer values. Depending on the layer value, it generates the address signals to RAM and ROM, which are used to read the value of polynomials and twiddle factors. The address sequence of each operation (NTT, INTT, Input, and Output) is stored in BRAM to reduce critical paths, enhance the circuit's speed, and reduce LUT resource consumption.

3.2. Hardware design of the butterfly unit

The butterfly unit handles three operations: modular addition, subtraction, and multiplication. There are the multiplication and the modular reduction steps for modular multiplication. With CRYSTALS-Kyber prime $q = 3329$, each polynomial coefficient's required bit length is 12-bit. The inputs for the butterfly unit

and data in memory are stored as 16-bit for consistency with the software model using reference C code. Furthermore, at the first step, where polynomials are randomly created in Kyber, they are not required to be modular with prime q by default. So, the initial input into the butterfly unit can be larger than the prime q, and every modular operation unit must prepare for these inputs. Figure 5 shows the block diagram of the butterfly unit. It has 16-bit inputs a and b, 16-bit outputs c and d, a $mode$ selection between CT NTT and GS INTT, and a 16-bit ω input for the twiddle factors. Modular addition and modular subtraction use Brent-Kung adders. The modular multiplication operations use a digital signal processing (DSP) core and a K-RED reduction block. The latency for each butterfly CT/GS operation is 9 clocks.

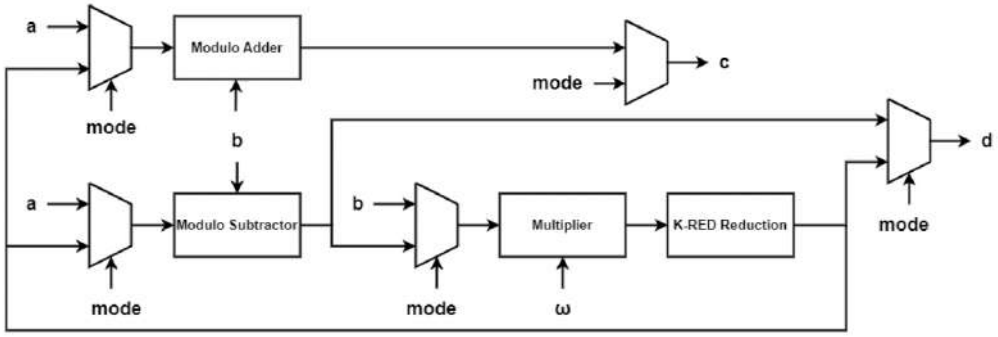

Figure 5. The block diagram of the butterfly unit

3.3. KRED modulo reduction

The K-RED reduction module is designed for $q = 3329$ of Kyber. It is divided into 3 steps as presented in Algorithm 3. The first step is getting the low and high bits and then extending the low bits. The second step is shifting extended low bits 10 bits to the left. The final calculation gives the modular reduction result $S = C \ mod \ q$.

Algorithm 3 K-RED reduction for CRYSTALS-Kyber

Input: $q = 3329, C = (C_{31}, C_{30} \ldots C_0), k = 13, m = 8$
Output: $S = kC_0 - C_1 = C \ mod \ q$
$C_{low} \leftarrow$ zero-extend $(C_7, ..., C_0)$ to 24-bit
$C_1 = C_{high} \leftarrow (C_{31}, ..., C_8)$
$C_0 = C_{low} << 4$
Return: $S = C_0 - (C_1 + C_{low} + C_{low} << 1)$

3.4. RAM and ROM

The RAM module consists of two block RAM (BRAM) configured in dual port mode, as shown in Figure 3. There is a 256x16-bit memory array that stores the data in the RAM. When the write enables signal we is asserted ($we = 1$), the input data ($DA1in, DA2in, DB1in, DB2in$) is written into the RAM at the specified write addresses ($A1wadd, A2wadd, B1wadd, B2wadd$). When the write enables signal we is de-asserted ($we = 0$), the output data ($DA1out, DB1out, DA2out, DB2out$) is read from the RAM at the read addresses ($A1radd, B1radd, A2radd, B2radd$).

The ROM module can be accessed based on specific addresses provided by the address generator module as inputs. The dual-port ROM holds the pre-computed twiddle factor ω. It contains a 256x16-bit memory array. The initial values of the ROM are pre-loaded in the parameter for CRYSTALS-Kyber. The ROM consists of 128 signed 16-bit values.

4. RESULTS AND DISCUSSION

We simulate and synthesize our design on Xilinx Vivado 23.1 FPGA platform Artix-7. The design occupies 1405 LUTs, 190 flip flops (FFs), 10 DSP blocks, and 11 BRAM and runs at the speed of 262 Mhz. We compare the proposed design with existing relevant references on NTT/INTT accelerators based on several

key metrics, including configuration (butterfly unit), area utilization (LUTs, FFs, DSPs, and BRAM), and operating speed. Regarding area utilization, the Karatsuba algorithm [28] achieves a relatively low LUT count compared to other designs. However, it has a moderate usage of FFs and a limited number of DSPs and BRAMs. The low-comp design [3] demonstrates a significant reduction in LUTs but utilizes more FFs. The QISC design [20] shows a high utilization of DSPs but lacks BRAM usage. Our proposed design, as shown in Table 2, offers a comparable LUT count compared to other designs while maintaining a suitable utilization of FFs, DSPs, and BRAMs.

In terms of speed, our design achieves a higher clock speed while using fewer FFs than other references. This is mainly because the usage of BRAM resource to store the address sequence, the twiddle factors and data rather than using LUT for logic address generation. More FFs and pipeline stages could be added to improve performance in applications where speed is essential. Each design's butterfly unit (block utilization) configuration is highlighted in Table 2. Our proposed design adopts a 2-butterfly unit configuration, similar to [3], [20], [28], which simplifies integration into existing systems and processor structures such as RISC-V. It is essential to consider the platform on which these designs were implemented; different FPGA platforms could vary in synthesis results, making direct comparisons impossible. Generally, our design is more efficient regarding area efficiency and is more configurable than existing references.

Table 2. The proposed method in comparison with relevant references for NTT/INTT accelerators

Design	Butterfly unit	Area				Speed
	Config	LUTs	FFs	DSPs	BRAM	[MHz]
Exact-KRED [4][2]	2x2	1401[2]	2929	4	1	237
HS-NTT [29][1]	2x2	801	717	4	2	222
Karatsuba [28][1]	2	1737	1167	2	3	161
QISC [20]	2	2908	170	9	0	-
Low-comp [3][1]	2	741	330	2	5	245
Proposed method [1]	2	1405	190	10	11	262

5. CONCLUSION

In this paper, we have contributed an efficient hardware accelerator for both NTT and INTT for CRYSTAL-Kyber. We integrated the Brent-Kung method to enhance computational efficiency, optimizing modular addition and subtraction operations. When implemented on the Xilinx Artix-7 FPGA platform, our design achieves a clock speed of 262 MHz while consuming only 1405 LUTs. Our design excels in area utilization with high clock speed. In future research, we aim to improve the speed performance by increasing the scalability and configurability of the design. We could include more complex BU structure such as 2x2 or 4x2 to enhance the efficiency of NTT and INTT operations. Another direction would be trying to improve the overall throughput by a mesh of parallel and pipelined BU and NTT/INTT overhead for each layer, so that we could calculate different polynomials at the same time. The design could be used in large scale in PQC accelerator cards in data centers or implemented as a small cryptography accelerator in internet of things (IoT) application.

ACKNOWLEDGEMENTS

This research is funded by Vietnam National University Ho Chi Minh City (VNU-HCM) under grant number DS2022-20-05. We would like to thank Ho Chi Minh City University of Technology (HCMUT), VNU-HCM for the support of time and facilities for this study.

REFERENCES

[1] D. Moody, "Module-lattice-based key-encapsulation mechanism standard," *National Institute of Standards and Technology*, 2023. doi: 10.6028/NIST.FIPS.203.ipd.

[2] J. Bos *et al.*, "CRYSTALS - Kyber: a CCA-secure module-lattice-based KEM," in *Proceedings-3rd IEEE European Symposium on Security and Privacy, EURO S and P 2018*, Apr. 2018, pp. 353–367, doi: 10.1109/EuroSP.2018.00032.

[3] 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*, vol. 2020, no. 2, pp. 49–72, 2020, doi: 10.13154/tches.v2020.i2.49-72.

[4] H. Nguyen and L. Tran, "Design of polynomial NTT and INTT accelerator for post-quantum cryptography CRYSTALS-Kyber," *Arabian Journal for Science and Engineering*, vol. 48, no. 2, pp. 1527–1536, Feb. 2023, doi: 10.1007/s13369-022-06928-w.
[5] T. Ye, Y. Yang, S. R. Kuppannagari, R. Kannan, and V. K. Prasanna, "FPGA acceleration of number theoretic transform," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12728 LNCS, pp. 98–117, 2021, doi: 10.1007/978-3-030-78713-4_6.
[6] J. Zhang, M. Imani, and E. Sadredini, "BP-NTT: fast and compact in-SRAM number theoretic transform with bit-parallel modular multiplication," *Proceedings - Design Automation Conference*, vol. 2023-July, 2023, doi: 10.1109/DAC56929.2023.10247691.
[7] D. W. Kim, D. I. Maulana, and W. Jung, "Kyber accelerator on FPGA using energy-efficient LUT-based barrett reduction," in *Proceedings - International SoC Design Conference 2022, ISOCC 2022*, Oct. 2022, pp. 83–84, doi: 10.1109/ISOCC56007.2022.10031533.
[8] T. Fritzmann and J. Sepulveda, "Efficient and flexible low-power NTT for lattice-based cryptography," in *Proceedings of the 2019 IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2019*, May 2019, pp. 141–150, doi: 10.1109/HST.2019.8741027.
[9] C. Zhang *et al.*, "Towards efficient hardware implementation of NTT for Kyber on FPGAs," *Proceedings - IEEE International Symposium on Circuits and Systems*, vol. 2021-May, 2021, doi: 10.1109/ISCAS51556.2021.9401170.
[10] B. M. K. Younis and A. K. Younis, "Hardware accelerator for anti-aliasing Wu's line algorithm using FPGA," *Telkomnika (Telecommunication Computing Electronics and Control)*, vol. 19, no. 2, pp. 672–682, Apr. 2021, doi: 10.12928/TELKOMNIKA.v19i2.18158.
[11] P. Visconti, R. Velazquez, C. Del-Valle-Soto, and R. De Fazio, "FPGA based technical solutions for high throughput data processing and encryption for 5G communication: A review," *Telkomnika (Telecommunication Computing Electronics and Control)*, vol. 19, no. 4, pp. 1291–1306, Aug. 2021, doi: 10.12928/TELKOMNIKA.v19i4.18400.
[12] M. K. Metwaly *et al.*, "Smart integration of drive system for induction motor applications in electric vehicles," *International Journal of Power Electronics and Drive Systems*, vol. 12, no. 1, pp. 20–28, Mar. 2021, doi: 10.11591/ijpeds.v12.i1.pp20-28.
[13] T. Gomathi and M. Shaby, "An efficient and effective energy harvesting system using surface micromachined accelerometer," *International Journal of Power Electronics and Drive Systems*, vol. 13, no. 2, pp. 1068–1074, 2022, doi: 10.11591/ijpeds.v13.i2.pp1068-1074.
[14] R. Paludo and L. Sousa, "NTT architecture for a linux-ready RISC-V fully-homomorphic encryption accelerator," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 7, pp. 2669–2682, Jul. 2022, doi: 10.1109/TCSI.2022.3166550.
[15] K. Derya, A. C. Mert, E. Öztürk, and E. Savaş, "CoHA-NTT: a configurable hardware accelerator for NTT-based polynomial multiplication," *Microprocessors and Microsystems*, vol. 89, p. 104451, Mar. 2022, doi: 10.1016/j.micpro.2022.104451.
[16] R. Geelen *et al.*, "BASALISC: programmable hardware accelerator for BGV fully homomorphic encryption," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2023, no. 4, pp. 32–57, Aug. 2023, doi: 10.46586/tches.v2023.i4.32-57.
[17] H. Kuang, Y. Zhao, and J. Han, "A high-speed NTT-based polynomial multiplication accelerator with vector extension of RISC-V for saber algorithm," in *APCCAS 2022 - 2022 IEEE Asia Pacific Conference on Circuits and Systems*, Nov. 2022, pp. 592–595, doi: 10.1109/APCCAS55924.2022.10090293.
[18] Z. Chen, Y. Ma, T. Chen, J. Lin, and J. Jing, "Towards efficient Kyber on FPGAs: a processor for vector of polynomials," in *Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC*, Jan. 2020, vol. 2020-January, pp. 247–252, doi: 10.1109/ASP-DAC47756.2020.9045459.
[19] P. Karl, J. Schupp, T. Fritzmann, and G. Sigl, "Post-quantum signatures on RISC-V with hardware acceleration," *ACM Transactions on Embedded Computing Systems*, Jan. 2023, doi: 10.1145/3579092.
[20] T. Fritzmann, G. Sigl, and J. Sepúlveda, "Risq-v: tightly coupled risc-v accelerators for post-quantum cryptography," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 4, pp. 239–280, 2020, doi: 10.13154/tches.v2020.i4.239-280.
[21] T. Fritzmann, U. Sharif, D. Müller-Gritschneder, C. Reinbrecht, U. Schlichtmann, and J. Sepulveda, "Towards reliable and secure post-quantum co-processors based on RISC-V," in *Proceedings of the 2019 Design, Automation and Test in Europe Conference and Exhibition, DATE 2019*, Mar. 2019, pp. 1148–1153, doi: 10.23919/DATE.2019.8715173.
[22] Z. Azad, G. Yang, R. Agrawal, D. Petrisko, M. Taylor, and A. Joshi, "RACE: RISC-V SoC for en/decryption acceleration on the edge for homomorphic computation," *Proceedings of the International Symposium on Low Power Electronics and Design*, 2022, doi: 10.1145/3531437.3539725.
[23] D. Micciancio and O. Regev, "Lattice-based cryptography," *Post-Quantum Cryptography*, pp. 147–191, 2009, doi: 10.1007/978-3-540-88702-7_5.
[24] M. Scott, "A note on the implementation of the number theoretic transform," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10655 LNCS, pp. 247–258, 2017, doi: 10.1007/978-3-319-71045-7_13.
[25] T. Pöppelmann and T. Güneysu, "Towards efficient arithmetic for lattice-based cryptography on reconfigurable hardware," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7533 LNCS, pp. 139–158, 2012, doi: 10.1007/978-3-642-33481-8_8.
[26] R. P. Brent and H. T. Kung, "A regular layout for parallel adders," *IEEE Transactions on Computers*, vol. C–31, no. 3, pp. 260–264, Mar. 1982, doi: 10.1109/TC.1982.1675982.
[27] P. Longa and M. Naehrig, "Speeding up the number theoretic transform for faster ideal lattice-based cryptography," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10052 LNCS, pp. 124–139, 2016, doi: 10.1007/978-3-319-48965-0_8.
[28] Y. Xing and S. Li, "A compact hardware implementation of cca-secure key exchange mechanism crystals-kyber on FPGA," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2021, no. 2, pp. 328–356, Feb. 2021, doi: 10.46586/tches.v2021.i2.328-356.
[29] M. Bisheh-Niasar, R. Azarderakhsh, and M. Mozaffari-Kermani, "High-speed NTT-based polynomial multiplication accelerator for post-quantum cryptography," in *Proceedings - Symposium on Computer Arithmetic*, Jun. 2021, vol. 2021-June, pp. 94–101, doi: 10.1109/ARITH51176.2021.00028.

BIOGRAPHIES OF AUTHORS

Toan Nguyen received the B.S. degree in Electronics and Telecommunications Engineering from Ho Chi Minh City University of Technology, VNU-HCM, Vietnam (2023). Currently, he is pursuing a Master of Electronics at the Faculty of Electrical-Electronics Engineering, Ho Chi Minh City University of Technology VNU-HCM. He can be contacted at email: nhtoan.sdh20@hcmut.edu.vn.

Hoang Anh is pursuing a B.S. degree in Electronics and Telecommunications Engineering from Ho Chi Minh City University of Technology, VNU-HCM, Vietnam. He is researching efficient algorithms and hardware design on various FPGA platforms. He can be contacted at email: anh.phamhoanganhov@hcmut.edu.vn.

Hung Nguyen received the B.S., M.S. degree in Electronics and Telecommunications Engineering from Ho Chi Minh City University of Technology, Vietnam (2019, 2022), Currently, he is working as a lecturer at the Faculty of Electrical-Electronics Engineering, Ho Chi Minh City University of Technology VNU-HCM. He can be contacted at email: ngthung@hcmut.edu.vn.

Trang Hoang received his Ph.D. degree in Microelectronics from CEA-LETI and University Joseph Fourier, France, in 2009. He is an Associate Professor in Electronics Engineering at Ho Chi Minh City University of Technology, Vietnam National University Ho Chi Minh City VNU-HCM, Vietnam and the Dean of the Graduate School, Ho Chi Minh City University of Technology, VNU-HCM. His fields of research interest are in the domains of ASIC/FPGA implementation, IC architecture, MEMs, wireless communications, wireless security, signal processing, optimisation for IC design and wireless networks, AI, and quantum computing. He can be contacted at email: hoangtrang@hcmut.edu.vn.

Linh Tran received the B.S. degree in Electrical and Computer Engineering from the University of Illinois, Urbana–Champaign (2005), M.S. and Ph.D. in Computer Engineering from Portland State University (2006, 2015). Currently, he is working as a lecturer at the Faculty of Electrical-Electronics Engineering Ho Chi Minh City University of Technology VNU-HCM. His research interests include quantum/reversible logic synthesis, computer architecture, hardware-software co-design, efficient algorithms and hardware design targeting FPGAs, and data analysis. He can be contacted at email: linhtran@hcmut.edu.vn.