

**Please cite the Published Version**

Imran, Malik, Khan, Safiullah, Khalid, Ayesha, Rafferty, Ciara, Shah, Yasir Ali, Pagliarini, Samuel, Rashid, Muhammad and O'Neill, Maire (2024) Evaluating NTT/INTT implementation styles for post-quantum cryptography. IEEE Embedded Systems Letters. ISSN 1943-0663

**DOI:** <https://doi.org/10.1109/LES.2024.3410516>

**Publisher:** Institute of Electrical and Electronics Engineers (IEEE)

**Version:** Accepted Version

**Downloaded from:** <https://e-space.mmu.ac.uk/635349/>









**Usage rights:**  [Creative Commons: Attribution 4.0](https://creativecommons.org/licenses/by/4.0/)

**Additional Information:** The final version of this article was published in IEEE Embedded Systems Letters

**Enquiries:**

If you have questions about this document, contact [openresearch@mmu.ac.uk](mailto:openresearch@mmu.ac.uk). Please include the URL of the record in e-space. If you believe that your, or a third party's rights have been compromised through this document please see our Take Down policy (available from <https://www.mmu.ac.uk/library/using-the-library/policies-and-guidelines>)

# Evaluating NTT/INTT Implementation Styles for Post-Quantum Cryptography

Malik Imran , Safiullah Khan , Ayesha Khalid , Ciara Rafferty , Yasir Ali Shah , Samuel Pagliarini ,  
Muhammad Rashid  and Máire O'Neill 

**Abstract**—Unifying the forward and inverse operations of the number theoretic transform (NTT) into a single hardware module is a common practice when designing polynomial coefficient multiplier accelerators as used in the post-quantum cryptographic algorithms. This work experimentally evaluates that this design unification is not always advantageous. In this context, we present three NTT hardware architectures: (i) A forward NTT (FNTT) architecture, (ii) An inverse NTT (INTT) architecture and (iii) A unified NTT (UNTT) architecture for computing the FNTT and INTT computations on a single design. We benchmark our throughput/area and energy/area evaluations on Xilinx Virtex-7 FPGA and 28nm ASIC platforms. The standalone FNTT and INTT designs, on average on FPGA, exhibit  $4.66\times$  and  $3.75\times$  higher throughput/area and energy/area values respectively than the UNTT design. Similarly, the individual FNTT and INTT designs, on average on ASIC, achieve  $1.25\times$  and  $1.09\times$  higher throughput/area and energy/area values respectively, compared to the UNTT design.

**Index Terms**—Post-quantum cryptography, number theoretic transform, polynomial multiplication, FPGA, ASIC.

## I. INTRODUCTION

FOR many post-quantum cryptography (PQC) algorithms, the polynomial coefficient multiplication is a computational bottleneck. Consequently, the number theoretic transform (NTT) based multipliers are extensively implemented on field-programmable gate arrays (FPGAs) and application-specific integrated circuits (ASICs) platforms [1], [2]. In addition to the standardised PQC algorithms, other applications demanding NTT hardware designs include fully homomorphic encryption (FHE) [3] and high-speed network servers [4].

State-of-the-art NTT accelerators are frequently optimised for high-speed and/or area reduction by unifying the forward NTT (FNTT) and the inverse NTT (INTT) operations in a unified NTT (UNTT) design. While a unified design ensures resource optimisation, there is a corresponding increase in

This work is funded by the grant from the Engineering and Physical Sciences Research Council (EPSRC) Quantum Communications Hub (EP/T001011/1).

M. Imran, A. Khalid, C. Rafferty and M. O'Neill are with the Centre for Secure Information Technologies (CSIT), Queen's University, Belfast, Northern Ireland, UK. (e-mail: {m.imran@qub, c.m.rafferty@qub, a.khalid@qub, m.oneill@ecit.qub}.ac.uk)

S. Khan is with the Manchester Metropolitan University, Manchester, UK. (e-mail: safiullah.khan@mmu.ac.uk)

Y. A. Shah is with the Ulster University, Magee Campus, Northern Ireland, UK. (e-mail: y.shah@ulster.ac.uk)

S. Pagliarini is with Carnegie Mellon University, Pittsburgh - PA, US. (e-mail: pagliarini@cmu.edu)

M. Rashid is with the Umm Al-Qurrah University, Makkah, Saudi Arabia. (e-mail: mfelahi@uqu.edu.sa)

M. Imran and S. Pagliarini were previously affiliated with the Tallinn University of Technology, Tallinn, Estonia (e-mail: {malik.imran, samuel.pagliarini}@taltech.ee)

computation time since the FNTT/INTT have to be computed one at a time on the UNTT. Therefore, pipelining is often employed to improve the computational performance, along with various other techniques, as shown in [1], [4]–[6]. Similarly, an instruction-set accelerated implementation of PQC algorithms is described in [7].

Despite a variety of UNTT accelerators [1]–[7], an empirical comparison/evaluation has never been made between the UNTT architecture and the individual architectures for computing forward and inverse operations (FNTT and INTT architectures). Moreover, existing UNTT accelerators are frequently optimised for area and computation time, without paying due attention to the essential power and energy design parameters. This work evaluates UNTT, FNTT and INTT architectures in terms of area, timing, and energy for the CRYSTALS-Kyber PQC algorithm. The objective is to illustrate that the unified design of forward and inverse NTT operations may not always yield benefits. To do this, we have implemented the following three architectures:

- FNTT for only forward NTT computations employing the Cooley-Tukey butterfly (CT-BTF).
- INTT for only inverse NTT computations utilising the Gentleman-Sande butterfly (GS-BTF).
- UNTT for both forward and inverse NTT computations utilising a unified CT & GS butterfly design.

The presented FNTT, INTT and UNTT designs are platform agnostic as we have employed register banks instead of Block RAMs (BRAMs) for FPGA or compiled memories for ASIC. A detailed evaluation is presented using two figures of merit (FoMs): throughput/area and energy/area.

## II. PRELIMINARIES

The NTT transformations of an  $n$ -degree polynomial  $f = \sum_{i=0}^{n-1} f_i x^i$  is defined using Eq. 1.

$$\hat{f} = NTT(f) = \sum_{i=0}^{n-1} \hat{f}_i x_i \quad (1)$$

In Eq. 1,  $\hat{f}_i = \sum_{j=0}^{n-1} f_j \zeta^{(2i+1)j}$ , where  $\zeta$  is the  $2n$ -th primitive root of the unit. Similarly, the inverse NTT is computed using Eq. 2.

$$f = NTT^{-1}(\hat{f}) = \sum_{i=0}^{n-1} f_i x_i \quad (2)$$

In Eq. 2,  $f_i = n^{-1} \sum_{j=0}^{n-1} \hat{f}_j \zeta^{-i(2j+1)}$ , where  $\zeta$  is the  $2n$ -th primitive root of unit. The NTT transform and its inverse can be applied efficiently using a chain of the  $\log_2 n$  butterflies. It

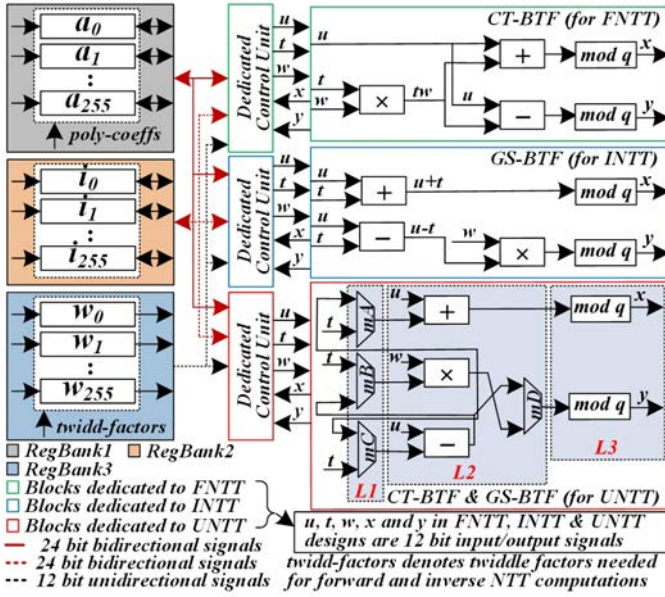


Fig. 1: Proposed NTT designs for FNTT, INTT and UNTT. Each design contains RegBanks and a controller unit. The green, blue and red outlined boxes show the components dedicated to FNTT, INTT and UNTT, respectively. We divide the components of CT-BTF & GS-BTF into three levels: L1, L2 and L3. In level L1, mux  $mB$  initiates computations for forward NTT; multiplexers  $mA$  and  $mC$  start inverse NTT. Finally, components in levels L2 and L3 operate serially, based on the outcomes of level L1.

is a divide and conquer approach that splits the input in half in each step and solves two problems of size  $\frac{n}{2}$ .

### III. PROPOSED HARDWARE ARCHITECTURES

The block diagram of the proposed hardware accelerator architectures for FNTT, INTT and UNTT is shown in Fig. 1. It includes three RegBanks, one CT-BTF, one GS-BTF, one unified CT-BTF & GS-BTF design, and three dedicated control units.

1) *RegBanks*: The three RegBanks (i.e., *Regbank1*, *RegBank2*, and *RegBank3*) are memory elements for keeping initial, intermediate and final results during or after the computations. Since the number of input polynomial coefficients (represented by  $n$ ) in the target PQC algorithm (CRYSTALS-Kyber) is 256 with a coefficient size of 12 bits, each RegBank contains 256 registers such that the size of each register is 12 bits. Thus, the overall size of a RegBank is  $256 \times 12$ . The 256 input polynomial coefficients are loaded in *RegBank1*. Similarly, the 256 twiddle factors for the corresponding forward or inverse NTT operations are loaded in *RegBank3*. Twiddle factors are the complex exponential coefficients necessary for manipulating and transforming data from the time domain to NTT and vice versa [1]. Once the initial loading into subsequent RegBanks is completed, the *RegBank1* and *RegBank2* registers are reused to hold the intermediate and the final results. Bidirectional arrows show the iterative use of *RegBank1* and *RegBank2* in Fig. 1.

2) *CT-BTF & GS-BTF* (Green & Blue Outlined Boxes in Fig. 1): The CT-BTF and GS-BTF perform the forward and inverse NTT operations, respectively. The mathematical expressions corresponding to their implementations are presented in Eq. 3 and Eq. 4, respectively.

$$CT - BTF = (u + tw) \bmod q \ \& \ (u - tw) \bmod q \quad (3)$$

$$GS - BTF = (u + t) \bmod q \ \& \ (u + tw) \bmod q \quad (4)$$

The parameter  $q = 3329$  is taken from the CRYSTALS-Kyber specification document [8]. The fundamental components to implement the above equations include an adder, multiplier, subtractor and modular reduction (i.e.,  $\bmod q$ ). The addition, multiplication and subtraction operations are implemented using the '+', '×' and '-' operators of a hardware description language (HDL). The  $\bmod q$  is implemented using the Barrett reduction algorithm, as in [2]. It can be observed from Eq. 3 and Eq. 4 that the CT-BTF and GS-BTF designs require the values for  $u$ ,  $t$  and  $w$ . Therefore, Fig. 1 take three 12-bit inputs  $u$ ,  $t$ , and  $w$ , where  $u$  and  $t$  are the input polynomial coefficients, while  $w$  is the corresponding twiddle-factor of subsequent FNTT and INTT operations. The CT-BTF and GS-BTF accelerators generate two 12-bit outputs  $x$  &  $y$  for the related FNTT or INTT operations of Eq. 3 and Eq. 4.

3) *Unified Design of CT-BTF & GS-BTF* (Red Outlined Boxes in Fig. 1): Eq. 3 & Eq. 4 reveal that both CT-BTF and GS-BTF require addition, multiplication and subtraction operations. Moreover, four modular reductions are also required (two for each CT-BTF and GS-BTF). However, using multiple arithmetic operators and modular reductions consumes additional hardware resources. Consequently, a unified design is advocated for the CT-BTF and GS-BTF by: (i) employing a singular adder, multiplier and subtractor, (ii) using only two modular reduction operations instead of four, and (iii) four  $2 \times 1$  multiplexers for routing purposes, as illustrated in the red outlined box of Fig. 1.

The unified design implements four mathematical expressions from Eq. 3 and Eq. 4. Its I/O interface contains three 12-bit inputs ( $u$ ,  $t$  and  $w$ ) and two 12-bit outputs (twiddle factors of FNTT or INTT operations). The  $clk$  and  $rst$  signals, including other related control signals, are not shown in Fig. 1. Moreover, it includes three arithmetic operators (i.e., one adder, one multiplier and one subtractor) and two reduction modules. The implementation of arithmetic operations is similar to the implementation of the green and blue portions of Fig. 1, described in section III-2. Similarly, the  $\bmod q$  is computed as implemented in [2].

In addition, the unified design contains four  $2 \times 1$  multiplexers, i.e.,  $mA$ ,  $mB$ ,  $mC$ , and  $mD$ . The control signal to all multiplexers is identical. When the control signal is '0', these four multiplexers allow the circuit to generate results for  $(u + tw) \bmod q$  and  $(u - tw) \bmod q$  in Eq. 3 to compute the NTT. Conversely, when the control signal is '1', the results are generated for the mathematical expressions of  $(u + t) \bmod q$  and  $w(u - t) \bmod q$  in Eq. 4 to compute the INTT.

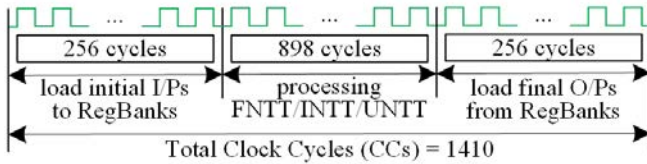


Fig. 2: Total clock cycles: 256 cycles to load  $u$ ,  $t$  &  $w$  parameters into the RegBanks; 898 cycles to implement Eq. 3 or Eq. 4; 256 cycles to store RegBanks data to the output.

4) *Controller & Clock Cycles Calculations*: Three finite state machine (FSM)-based controllers have been implemented, dedicated to FNTT, INTT and UNTT architectures. The purpose is to generate corresponding control signals to read/write data from/to RegBanks. Furthermore, the dedicated controller for the unified design also generates control signals for the routing multiplexers of CT-BTF & GS-BTF units. Therefore, the FNTT, INTT, and UNTT architectures require 1410 clock cycles<sup>1</sup> in total. The breakdown for the total clock cycles is shown in Fig. 2.

#### IV. RESULTS, EVALUATIONS AND COMPARISONS

1) *Implementation Results and Evaluations*: We have implemented our architectures in Verilog HDL using Xilinx Vivado 2023.2. The reason for choosing both FPGA and ASIC platforms is to confirm that the interpretation from experimental results is consistent across two different platforms. Various options exist for FPGA (such as Zynq SoC and AMD/Xilinx) and ASIC (65nm, 45nm, and 28nm) technologies for logic synthesis. The choice depends on specific application requirements, including performance, flexibility, power efficiency, and hardware cost. Therefore, we have presented the achieved results for AMD/Xilinx Virtex-7 (xc7vx690tffg1930-3) FPGA and 28nm ASIC platforms in Table I. Note that synthesising our designs on modern ASIC technologies will further reduce the hardware area.

To ensure a fair comparison, we have restricted the synthesis tool from using the digital signal processor (DSP) blocks for FPGA implementations and have provided area results in slices, look-up tables (LUTs) and flip-flops (FFs). The ASIC area is given in  $mm^2$ . We have obtained the power values using value change dump (VCD) files. The total power, which is a sum of static and dynamic powers, is presented for computing one forward or inverse NTT computations. Finally, in the last three columns of Table I, we show relative characteristics in terms of area, frequency, and energy using symbols  $\uparrow$  and  $\downarrow$ .

*Standalone Area, Timing & Energy Evaluations*: We present the results of the FNTT, INTT, and UNTT in Table I. The comparisons provided are direct comparisons between FNTT, INTT and UNTT. However, depending on the algorithm being implemented by a PQC accelerator, a designer would then have to consider the aggregate results from FNTT and INTT (e.g., by summing the area).

<sup>1</sup>These clock cycles only include CT-BTF & GS-BTF computations, implementing Eq. 3 and Eq. 4. The additional post-processing multiplications after INTT BTfs computation, required in the CRYSTALS-Kyber algorithm, are not considered here.

Table I reveals that the hardware resources (in slices) utilized in the FNTT design on Virtex-7 are relatively lower than those in the INTT and UNTT architectures. Also, this is true for the UNTT design, which utilises fewer slices than the INTT design. On 28nm ASIC technology, the hardware cost for the FNTT, INTT and UNTT designs is nearly comparable, with minor increases and decreases, as presented in columns three to five.

Comparing operating frequency and latency on both FPGA and ASIC platforms, the FNTT design outperforms INTT and UNTT architectures, while the INTT design outperforms the UNTT design. Note that on Virtex-7 FPGA, the latency of the UNTT design stems from the operational frequency (20MHz), due to the routing delays of four  $2 \times 1$  multiplexers. While the routing delays can be minimised by pipelining the UNTT datapath, it comes at the cost of additional resources and consumed power. Like frequency and latency, on FPGA and ASIC platforms, the FNTT and INTT designs result in higher throughput than UNTT (see column ten).

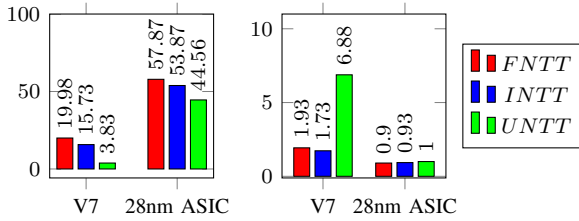
On both FPGA and ASIC platforms, the FNTT and INTT designs consume more power than the UNTT because these operate at higher circuit frequencies. Similarly, our evaluations show that the FNTT and INTT designs are more efficient (in consumed energy) than the UNTT. This efficiency comes as energy is the product of power and computation time, and as mentioned earlier, the FNTT and INTT designs outperform in computation time. Consequently, the FNTT and INTT designs consume less energy than the UNTT design.

*Simultaneous Evaluation of Area, Timing, and Energy*: We further evaluate area, timing, and energy results using two distinct FoMs in Fig 3. Fig. 3a illustrates the throughput-to-area unit ratio FoM, simultaneously evaluating timing and area. Similarly, Fig. 3b presents results for the energy-to-area unit ratio FoM, concurrently assessing energy and area. A higher ratio of throughput/area corresponds to a better design point, while a lower ratio of energy/area corresponds to a better design point. This trend is evident in both defined FoMs in Fig. 3a and Fig. 3b, where the FNTT and INTT designs outperform (on both FPGA and ASIC) in throughput/area and energy/area FoMs compared to the UNTT design. This happens due to a higher operating frequency achievement in FNTT and INTT designs. Moreover, the obtained (lower) circuit frequency due to feedback results of multiplier and subtractor to routing multiplexers in Fig. 1 causes lower FoMs for UNTT design. Using Fig.3a and Fig. 3b, we calculated the average sum of throughput/area and energy/area of FNTT and INTT designs. The calculated average sum of throughput/area values of FNTT and INTT designs on Virtex-7 and 28nm ASIC are 17.855 ( $\frac{19.98+15.73}{2}$ ) and 55.87 ( $\frac{57.87+53.87}{2}$ ), respectively. These average values indicate  $4.66\times$  and  $1.25\times$  superior throughput/area performance compared to the UNTT design. Similarly, repeating the same process for Fig. 3b, the average sum of energy/area of FNTT and INTT designs exhibits  $3.75\times$  and  $1.09\times$  better energy/area unit performance compared to UNTT design on Virtex-7 and 28nm ASIC, respectively.

2) *Comparisons, Discussions and Future Considerations*: The comparison to existing NTT accelerators is challenging due to different research objectives. While existing NTT

TABLE I: Implementation results of FNTT, INTT and UNTT architectures for  $n = 256$  and  $q = 3329$  on FPGA and ASIC. Freq is the circuit frequency; Lat is computation latency calculated as  $\frac{Clock\ Cycles}{Freq\ (MHz)}$ ; TP is the throughput; Total energy is calculated as  $\frac{Total\ Power\ (mW) \times Lat\ (\mu s)}{10^3}$ . Numbers in **blue** and **red** show superior/inferior performance by UNTT, respectively.

Device	Design	Hardware Resource Utilizations				Timing Related Information				Total Power (mW)	Total Energy ( $\mu$ J)	× Increase or Decrease ( $\uparrow/\downarrow$ )		
		FPGA			ASIC ( $mm^2$ )	Clock Cycles	Freq (MHz)	Lat ( $\mu$ s)	TP (Kbps)			Area	Freq & Lat	Energy
		Slices	LUTs	FFs										
V7	FNTT	3549	9187	9328	–	1410	100	14.10	70.92	488	6.880	–	–	–
	INTT	4209	9079	9341	–	1410	93	15.16	65.96	480	7.276	1.18 $\uparrow$	1.07 $\downarrow$	1.05 $\uparrow$
	UNTT	<b>3698</b>	<b>9298</b>	<b>9402</b>	–	1410	<b>20</b>	<b>70.50</b>	<b>14.18</b>	<b>361</b>	<b>25.450</b>	<b>1.13 <math>\downarrow</math></b>	<b>4.65 <math>\downarrow</math></b>	<b>3.49 <math>\uparrow</math></b>
28nm	FNTT	–	–	–	0.032	1410	2597	0.54	1851.85	54.535	0.029	–	–	–
	INTT	–	–	–	0.032	1410	2415	0.58	1724.13	53.349	0.030	0 $\uparrow\downarrow$	1.07 $\downarrow$	1.03 $\uparrow$
	UNTT	–	–	–	<b>0.033</b>	1410	<b>2061</b>	<b>0.68</b>	<b>1470.58</b>	<b>51.419</b>	<b>0.033</b>	<b>1.03 <math>\uparrow</math></b>	<b>1.17 <math>\downarrow</math></b>	<b>1.10 <math>\uparrow</math></b>



(a) Throughput/area unit (b) Energy/area unit

Fig. 3: FoMs as throughput/area unit and energy/area unit. Area unit is the number of slices for Virtex-7 FPGA (V7) or  $mm^2$  for 28nm ASIC.

designs [1]–[7] typically aim to unify forward and inverse NTT operations within a single design to optimise area and latency, our objective in this letter is to demonstrate that the unified design of NTT operations (forward and inverse) is not always advantageous.

Comparatively, our FNTT, INTT and UNTT designs require fewer clock cycles than the optimised NTT designs of [1]–[7]. This difference is due to the use of RegBanks in our designs as the NTT designs of [1]–[7] utilise BRAMs and SRAMs for respective platforms. However, the optimised NTT designs of [1]–[6] are better in terms of latency as they employ pipelining and parallelism approaches, which are not considered in our (FNTT, INTT and UNTT) designs. In comparison to [7] on an identical Artix-7 FPGA, our FNTT and INTT designs result in comparable latency values to compute one forward and inverse NTT operation ( $9.35\mu s$  and  $10.08\mu s$  in our FNTT and INTT designs and  $8.17\mu s$  and  $10.46\mu s$  in the reference work). On Virtex-7 FPGA, our unified design of CT-BTF & GS-BTF butterfly unit utilises lower LUTs than the flexible and reconfigurable butterfly unit design of [1].

The implemented approach in this research can be utilised for NTT realisations of several PQC algorithms, such as the NIST-standardised CRYSTALS-Dilithium, Falcon, and SPHINCS+. In addition to the NIST-standardised PQC algorithms, it can also be used for the forthcoming algorithms in the NIST standardisation of additional digital signatures like Raccoon, which uses a variant of a CRYSTALS-Dilithium. The FNTT, INTT and UNTT designs can be integrated within RISC-V implementation, to compute forward and inverse NTT operations. The described FNTT, INTT, and UNTT designs are not secured against side-channel attacks, e.g., fault attacks, and could be considered in the future by employing several countermeasures like error detection, as described in [2], [9].

## V. CONCLUSIONS

This research presents a case study investigating the performance of various micro-architectural choices of an NTT hardware accelerator, targeting the NIST-PQC CRYSTALS-Kyber algorithm with parameters  $n = 256$  and  $q = 3329$ . To our knowledge, the disjoint NTT and INTT hardware accelerators have not been fully compared against a unified NTT architecture in terms of energy consumption alongside throughput and area. Benchmarking on a Virtex-7 FPGA and 28nm ASIC reveals that the UNTT design benefits applications that demand area-optimised and power-efficient accelerators. Conversely, FNTT and INTT accelerators are better suited for applications prioritising higher processing speed with lower energy consumption. Investigating FoMs indicate that disjoint FNTT and INTT designs suit throughput/area-focused and energy/area-optimised applications.

## REFERENCES

- [1] 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, 2022.
- [2] S. Khan, A. Khalid, C. Rafferty, Y. A. Shah, M. O’Neill, W. Lee, and S. O. Hwang, “Efficient, error-resistant NTT architectures for crystals-kyber FPGA accelerators,” in *31st IFIP/IEEE International Conference on Very Large Scale Integration, VLSI-SoC 2023, Dubai, United Arab Emirates, October 16-18, 2023*. IEEE, 2023, pp. 1–6.
- [3] T. Ye, Y. Yang, S. R. Kuppannagari, R. Kannan, and V. K. Prasanna, “FPGA acceleration of number theoretic transform,” in *High Performance Computing*, B. L. Chamberlain, A.-L. Varbanescu, H. Ltaief, and P. Luszczek, Eds. Cham: Springer International Publishing, 2021, pp. 98–117.
- [4] M. Bisheh-Niasar, R. Azarderakhsh, and M. Mozaffari-Kermani, “High-speed NTT-based polynomial multiplication accelerator for post-quantum cryptography,” in *2021 IEEE 28th Symposium on Computer Arithmetic (ARITH)*, 2021, pp. 94–101.
- [5] Z. Ye, R. C. C. Cheung, and K. Huang, “PipeNTT: A pipelined number theoretic transform architecture,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 10, pp. 4068–4072, 2022.
- [6] Z. Ni, A. Khalid, D.-e.-S. Kundi, M. O’Neill, and W. Liu, “HPKA: A high-performance CRYSTALS-Kyber accelerator exploring efficient pipelining,” *IEEE Transactions on Computers*, vol. 72, no. 12, pp. 3340–3353, 2023.
- [7] 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.
- [8] P. Schwabe and J. Mann, “CRYSTALS-Kyber: Cryptographic suite for algebraic lattices,” 2022, last accessed on January 02, 2024, [Online] available at <https://pq-crystals.org/kyber/resources.shtml>.
- [9] A. Sarker, A. C. Canto, M. Mozaffari Kermani, and R. Azarderakhsh, “Error detection architectures for hardware/software co-design approaches of number-theoretic transform,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 7, pp. 2418–2422, 2023.