High-Speed NTT-based Polynomial Multiplication Accelerator for Post-Quantum Cryptography

Mojtaba Bisheh-Niasar[1], Reza Azarderakhsh[1,2], Mehran Mozaffari Kermani[3]

[1]CEECS Department, Florida Atlantic University
[2]PQSecure Technologies, LLC, Boca Raton, FL , USA
[3]CSE Department at University of South Florida

14-16 June 2021

ARITH 2021

Outline

1. Introduction
2. Kyber Protocol
3. Proposed Architecture for High-speed Polynomial Multiplier
4. Implementation Results and Comparison
5. Conclusion

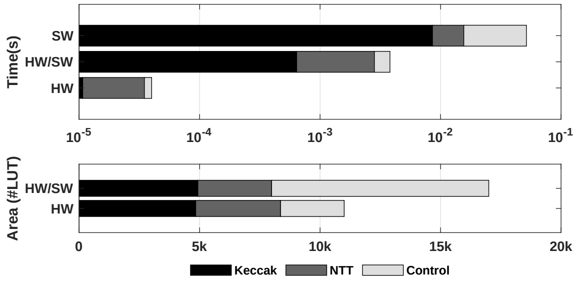
Introduction

- Motivation
  - Shor's algorithm (1994)
    - Complete break of classical cryptosystems
  - The NIST PQC standardization process
  - The importance of NTT performance in lattice-based cryptography
  - Reducing the overall complexity for a widely-deployed cryptosystem [1]
  - Taking advantages of pure hardware architectures
- CRYSTALS-Kyber
  - NIST round-3 finalist
  - Module learning with errors (Module-LWE) quantum-resistant scheme [2]
  - NTT-based scheme requiring large memory and complex memory access pattern

[1] NIST, "Submission requirements and evaluation criteria for the postquantum cryptography standardization process," National Institute of Standards and Technology, 2016.
[2] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehle, "CRYSTALSKyber: Algorithm specification and supporting documentation (version 3.0). submission to the NIST post-quantum cryptography standardization project," 2020.

Background

- Investigating three design strategies:
  - SW architecture
    - **2019:** Botros et al., Memory-efficient high-speed Cortex-M4 implementation
    - **2020:** Alkim et al., Cortex-M4 optimization for {R, M} LWE
  - HW/SW co-design architecture
    - **2019:** Basu et al., HSL-based architecture
    - **2019:** Barenjee et al., Configurable crypto-processor
    - **2020:** Alkim et al., ISA extention on RISC-V architecture
    - **2020:** Xin et al., High-performance vector processor
    - **2020:** Fritzmann et al., Tightly coupled RISC-V accelerator
  - HW architecture
    - **2020:** Dang et al., High-performance architecture
    - **2020:** Huang et al., Resource reusing approach
    - **2021:** Xing et al., Compact hardware implementation

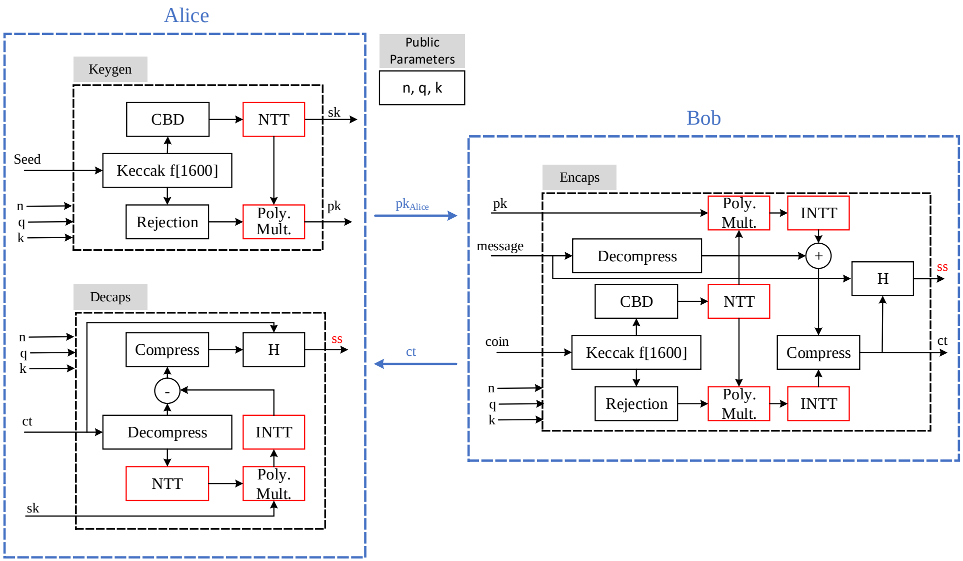


Performance (in log_{10}) and resource utilization comparison in three different Kyber implementation approaches: software (SW), hardware/ software (HW/SW), and hardware (HW).

Our Contributions

- Propose a hardware-friendly modular reduction algorithm
- Propose an improved reconfigurable hardware architecture for NTT/INTT
- Propose a parameterized design on an Artix-7 FPGA
- Efficiency improvement by 44% for NTT architecture
  - Employing 25% fewer Slice
  - Employing 80% fewer BRAM
- Propose a high-performance co-processor Kyber architecture
  - Accomplish three phases of the key exchange in 9, 12, and 19 μs

Kyber Protocol

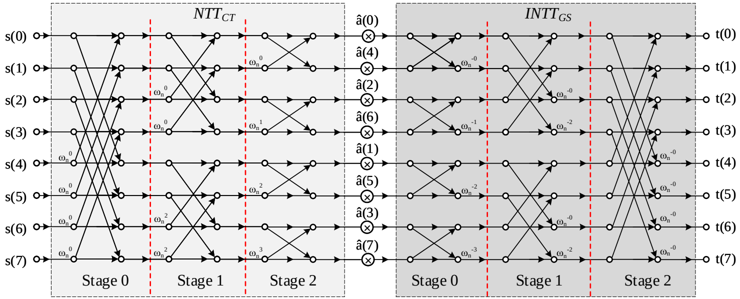


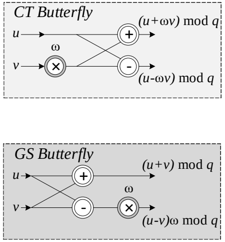
Polynomial Multiplication

- Number Theoretic Transform (NTT)

NTT: $\boxed{\hat{f}_i = \sum_{j=0}^{n-1} f_j \omega_n^{ij} \bmod q}$ INTT: $\boxed{f_i = n^{-1} \sum_{j=0}^{n-1} \hat{f}_j \omega_n^{-ij} \bmod q}$

- Polynomial Multiplication $\longrightarrow$ $f.g = \text{INTT}(\text{NTT}(f) \circ \text{NTT}(g))$





An 8-point NTT-based polynomial multiplication: (Left) Dataflow graph including CT butterfly-based NTT, point-wise multiplication, and GS butterfly-based INTT. Polynomial $\hat{a}$ is in NTT domain and s and t are in normal domain. (Right) CT and GS butterfly configurations.

Modular Reduction

- Barrett reduction
- Montgomery reduction
    - KRED and KRED- 2X [1] for special form of prime $q = k \cdot 2^m + 1$

```
Function KRED(C)
    $C_0 \leftarrow C \bmod 2^m$
    $C_1 \leftarrow C/2^m$
    return $kC_0 - C_1$
```

```
Function KRED-2x(C)
    $C_0 \leftarrow C \bmod 2^m$
    $C_1 \leftarrow C/2^m \bmod 2^m$
    $C_2 \leftarrow C/2^{2m}$
    return $k^2C_0 - kC_1 + C_2$
```

- Hardware-friendly algorithms without any multiplications $\longrightarrow$ Implementing by shifter and adder.
- Scaling the twiddle factors by $k^{-1} \cdot \omega_n^{ij}$ and $k^{-2} \cdot \omega_n^{ij}$

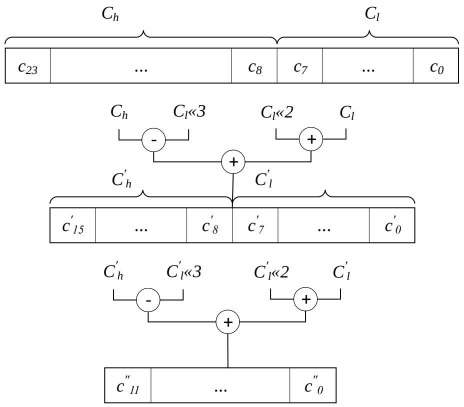
[1] P. Longa and M. Naehrig, "Speeding up the number theoretic transform for faster ideal lattice-based cryptography," in Cryptology and Network Security - 15th International Conference, CANS 2016, Milan, Italy, November 14-16, 2016, Proceedings, pp. 124–139, 2016.

Proposed K^2-RED Reduction Algorithm

- Require 4 shift and 6 addition
- Keep output width to 12 bits
- Halve required memory to store $k^{-2} \cdot \omega_n^{ij}$ ~~$, k^{-1} \cdot \omega_n^{ij}$~~

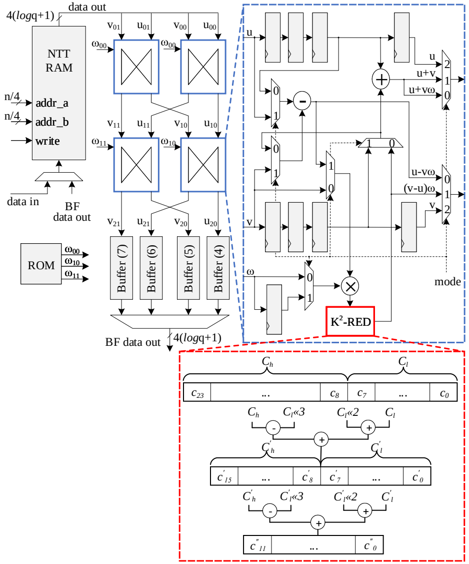
Input: A binary number $C = (c_{23}, \ldots, c_0)_2$, $k = 13$, $m = 8$, $q = 3329 = k \cdot 2^m + 1$
Output: $C^{''} = k^2 C \bmod q$

```
    Step 1:
1: $C_l = (c_7, \ldots, c_0)_2$
2: $C_h = (c_{23}, \ldots, c_8)_2$
3: $C^{'} \leftarrow k \cdot C_l - C_h$
    Step 2:
4: $C^{'}_l = (c^{'}_7, \ldots, c^{'}_0)_2$
5: $C^{'}_h = (c^{'}_{15}, \ldots, c^{'}_8)_2$
6: $C^{''} \leftarrow k \cdot C^{'}_l - C^{'}_h$
7: return $C^{''}$
```



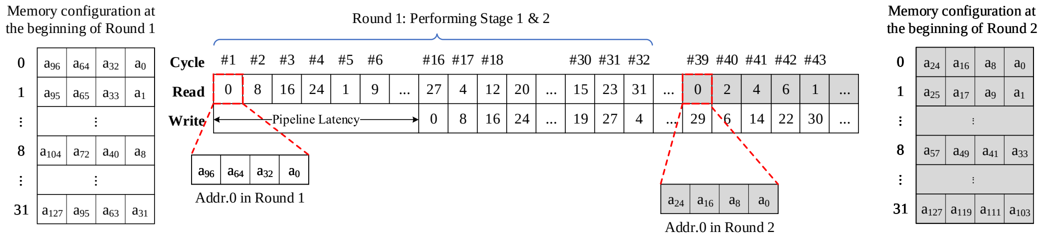
Reconfigurable Butterfly Core

- Proposed polynomial multiplication core (PMC)
  - Merge two layers of NTT/INTT
  - Perform two butterfly operations in each layer
- Avoid the bit-reverse cost in polynomial multiplication
  - $mode = 0 \rightarrow$ CT configuration
  - $mode = 1 \rightarrow$ GS configuration
  - $mode = 2 \rightarrow$ Bypass the first butterfly row to support odd number of layers
- Reduce the complexity from $\frac{n}{2}\log n$ to $\frac{n}{8}\log n$
- Reduce the required memory using the symmetry property:
  - $\omega_n^{-i} = -\omega_n^{n-i}$



Reconfigurable Butterfly Core

Memory Address and Data flow of NTT operation



Memory configuration at the beginning of Round 1
0 a_{96} a_{64} a_{32} a_0
1 a_{95} a_{65} a_{33} a_1
8 a_{104} a_{72} a_{40} a_8
31 a_{127} a_{95} a_{63} a_{31}
Round 1: Performing Stage 1 & 2
Cycle #1 #2 #3 #4 #5 #6 #16 #17 #18 #30 #31 #32 #39 #40 #41 #42 #43
Read 0 8 16 24 1 9 ... 27 4 12 20 ... 15 23 31 ... 0 2 4 6 1 ...
Write Pipeline Latency 0 8 16 24 ... 19 27 4 ... 29 6 14 22 30 ...
a_{96} a_{64} a_{32} a_0
Addr.0 in Round 1
a_{24} a_{16} a_8 a_0
Addr.0 in Round 2
Memory configuration at the beginning of Round 2
0 a_{24} a_{16} a_8 a_0
1 a_{25} a_{17} a_9 a_1
8 a_{57} a_{49} a_{41} a_{33}
31 a_{127} a_{119} a_{111} a_{103}

- Read/Store four coefficients per cycle
- $\frac{n}{4}$ reading/storing per round
- The pipeline latency:
  - 2 cycles for reading from RAM
  - 8 cycles for two butterfly operations
  - 4 cycles for buffering the results in registers

Architecture of CRYSTAL-Kyber

- High-speed core implementation of the Keccak based on [1]
  - 24 clock cycles
  - SIPO: 64-bit input, 1344-bit output
  - PISO: 1344-bit input, 64-bit output
- Binomial and rejection samplers
  - In a parallel fashion with Keccak core
  - The latency of samplers can be entirely absorbed by the Keccak core.
- Polynomial Multiplication
  - Input polynomials are in normal order.
  - The public and secret keys are in bit-reverse order.
  - The point-wise multiplication works in bit-reverse order in the NTT domain.
  - The results are transformed back to the normal domain with normal order.

[1] G. Bertoni, J. Daemen, S. Hoffert, M. Peeters, and G. V. Assche, "Keccak in VHDL," 2020.

Implementation Results

Implementation results for different modular reduction algorithms

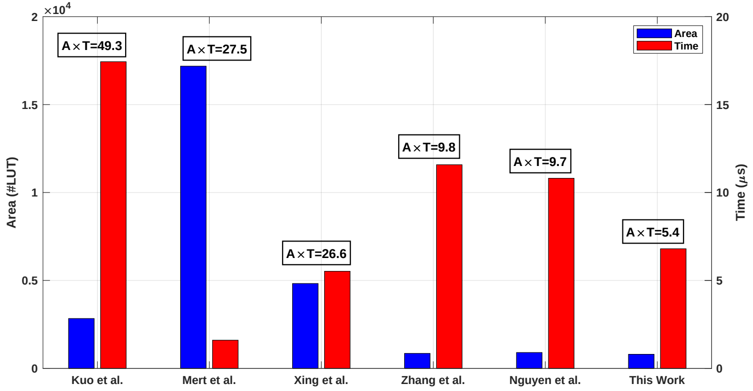
Reduction Algorithm	CPD [ns]	Area				Output Width
		#LUTs	#FFs	#Slices	#DSPs	
Barrett Reduction	1.34	59	31	26	2	12
Montgomery [1]	2.10	391	382	91	1	12
KRED [1]	1.99	80	47	31	0	16
K^2-RED	0.91	54	30	18	0	12

- K^2-RED
  - Reducing the utilized LUTs/FFs
  - Hardware-friendly architecture without any DSP module
  - Improving the maximum operating frequency
  - Halving the required memory compared with KRED [1]

[1] D. T. Nguyen, V. B. Dang, and K. Gaj, "High-level synthesis in implementing and benchmarking number theoretic transform in latticebased post-quantum cryptography using software/hardware codesign," in Applied Reconfigurable Computing. Architectures, Tools, and Applications - 16th International Symposium, ARC 2020, Toledo, Spain, April 1-3, 2020, Proceedings [postponed], pp. 247–257, 2020.

Implementation Results

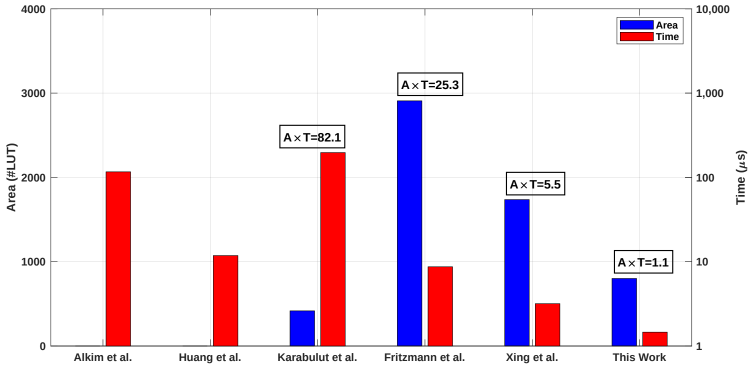
Implementation results for different NTT ($n = 1024$) implementation on FPGA



- Comparison to the best previous work
  - Achieving 44% A×T improvement
  - Achieving 1.59× speedup

Implementation Results

Implementation results for different NTT ($n = 256$) implementation on FPGA



- Comparison to the best previous work
  - Achieving 78% A$\times$T improvement
  - Achieving 2.18$\times$ speedup

Implementation Results

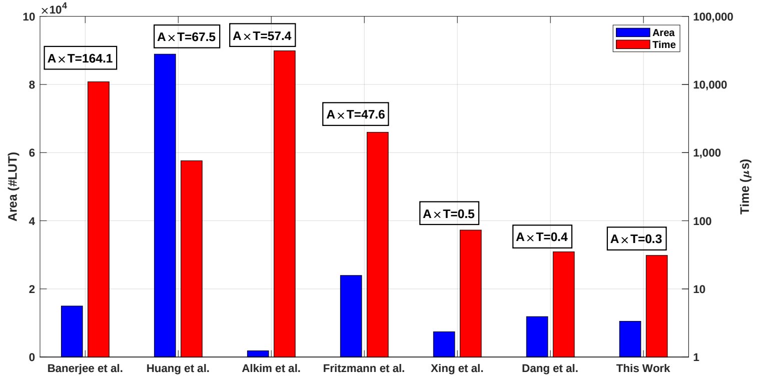
FPGA Implementation results and comparison with state-of-the-art

Scheme	Work	Platform	Area					KeyGen	Encaps	Decaps	Freq	Total Time	Throughput	$A \times T$
			#LUTs	#FFs	#Slices	#DSPs	#BRAMs	[CCs]	[CCs]	[CCs]	[MHz]	[μs]	[KEM/s]	
Kyber-512	Basu *et al.*[1]	Virtex-7	1,977,896	194,126	NA	0	0	-	31,669	43,018	67	1,115	897	2,214.2 (99.9%)
	Banerjee *et al.*	Artix-7	14,975	2,539	4,173	11	14	74,519	131,698	142,309	25	10,960	91	164.4 (99.8%)
	Fritzmann *et al.*	Zynq-7000	23,947	10,847	NA	21	32	150,106	193,076	204,843	-	-	-	47.6 (99.3%)
	Alkim *et al.*	Artix-7	1,842	1,634	NA	5	34	710,000	971,000	870,000	59	31,203	32	57.5 (99.4%)
	Huang *et al.*[1]	Artix-7	88,901	NA	141,825	354	202	-	49,015	68,815	155	760	1,315	67.8 (99.5%)
	Xing *et al.*	Artix-7	7,412	4,644	2,126	2	3	3,768	5,079	6,668	161	73	13,705	0.54 (34.0%)
	Dang *et al.*	Artix-7	11,864	10,348	3,989	8	15	-	3,025	4,395	210	35	28,301	0.42 (21.4%)
	This work	**Artix-7**	**10,502**	**9,859**	**3,547**	**8**	**13**	**1,882**	**2,446**	**3,754**	**200**	**31**	**32,258**	**0.33**
Kyber-768	Banerjee *et al.*	Artix-7	14,975	2,539	4,173	11	14	111,525	177,540	190,579	25	14,725	67	220.5 (99.8%)
	Fritzmann *et al.*	Zynq-7000	23,947	10,847	NA	21	32	273,370	325,888	340,418	-	-	-	79.8 (99.4%)
	Huang *et al.*[1]	Artix-7	110,260	NA	167,293	292	202	-	77,481	102,113	155	1,159	863	127.7 (99.6%)
	Xing *et al.*	Artix-7	7,412	4,644	2,126	2	3	6,316	7,925	10,049	161	112	8,957	0.83 (43.4%)
	Dang *et al.*	Artix-7	11,884	10,380	3,984	8	15	-	4,065	5,555	210	46	21,829	0.54 (13.0%)
	This work	**Artix-7**	**11,783**	**10,424**	**3,952**	**12**	**14**	**2,667**	**3,251**	**4,805**	**200**	**40**	**24,826**	**0.47**
Kyber-1024	Banerjee *et al.*	Artix-7	14,975	2,539	4,173	11	14	148,547	223,469	240,977	25	18,578	53	278.2 (99.7%)
	Fritzmann *et al.*	Zynq-7000	23,947	10,847	NA	21	32	349,673	405,477	424,682	-	-	-	99.4 (99.2%)
	Alkim *et al.*	Artix-7	1,842	1,634	NA	5	34	2,203,000	2,619,000	2,429,000	59	85,559	11	157.6 (99.5%)
	Huang *et al.*[1]	Virtex-7	132,918	NA	172,489	548	202	-	107,054	135,553	192	1,264	791	167.9 (99.6%)
	Xing *et al.*	Artix-7	7,412	4,644	2,126	2	3	9,380	11,321	13,908	161	157	6,381	1.16 (35.3%)
	Dang *et al.*	Artix-7	12,183	12,441	4,511	8	15	-	5,785	7,395	210	63	15,933	0.76 (1.3%)
	This work	**Artix-7**	**13,347**	**11,639**	**4,585**	**16**	**16**	**3,459**	**4,122**	**6,257**	**185**	**56**	**17,824**	**0.75**

[1] Different architectures for Encaps and Decaps are used.

Implementation Results

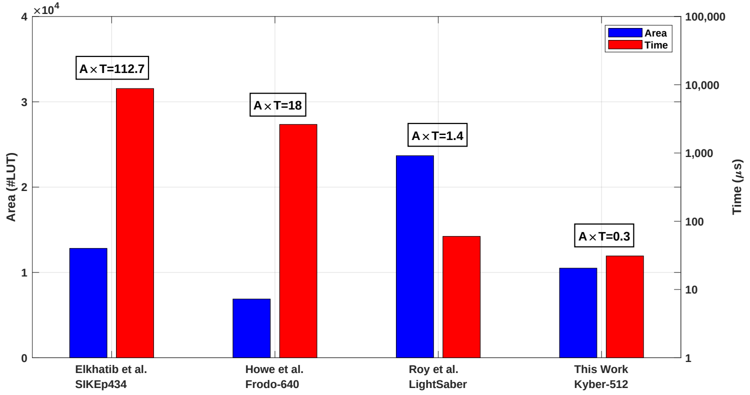
Kyber-512 FPGA Implementation results and comparison with state-of-the-art



- Comparison to the best previous works
  - Achieving 21% A×T improvement
  - Achieving 14% performance improvement

Implementation Results

Comparison with other PQC schemes in NIST security level 1.



[1] R. Elkhatib, R. Azarderakhsh, and M. Mozaffari Kermani, "Highly optimized montgomery multiplier for SIKE primes on FPGA," in 27th IEEE Symposium on Computer Arithmetic, ARITH 2020, Portland, OR, USA, June 7-10, 2020, pp. 64–71, 2020.
[2] J. Howe, M. Martinoli, E. Oswald, and F. Regazzoni, "Exploring parallelism to improve the performance of frodokem in hardware." Cryptology ePrint Archive, Report 2021/155, 2021.
[3] S. S. Roy and A. Basso, "High-speed instruction-set coprocessor for lattice-based key encapsulation mechanism: Saber in hardware," IACR Trans. Cryptogr. Hardw. Embed. Syst., vol. 2020, no. 4, pp. 443–466, 2020.

Conclusion and future works

❑ Conclusion:

- Optimizing the implementation of the NTT core
  - Merging the layers
  - Designing a configurable butterfly core
  - Proposing an efficient reduction unit
- Improving 44% efficiency in terms of A$\times$T
- Performing all KEM operations for Kyber
  - For a security level comparable to AES-128:
    - key generation, encapsulation, and decapsulation in 9, 12, and 19 μs

❑ Future works:

- Extending the design by side-channel countermeasures
- Reducing the memory utilization

Thanks for your attention.