A Monolithic Hardware Implementation of Kyber: Comparing Apples to Apples in PQC Candidates

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

6-8 October 2021

LatinCrypt 2021

Outline

1. Introduction
2. Proposed Architecture
3. Implementation Results and Comparison
4. Conclusion

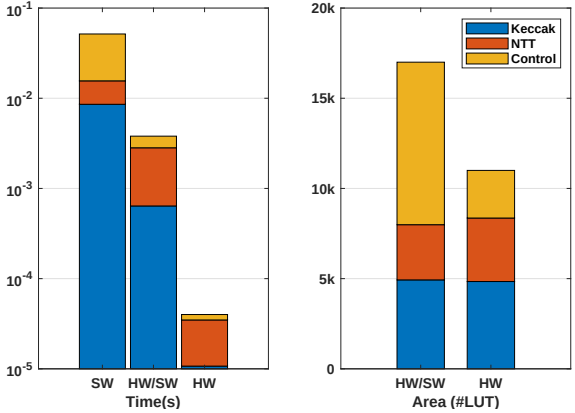
Introduction

- Motivation
  - Threat of large-scale quantum computers for classical cryptosystems
  - Importance of hardware accelerator to show PQC candidates' differentiation
  - Lack of pure hardware implementation
  - Lack of a fair comparison (apples to apples) with a common foundation
  - Efficiency comparison over ASIC platform as a common foundation
- CRYSTALS-Kyber
  - NIST round-3 finalist
  - Module learning with errors (Module-LWE) quantum-resistant scheme [1]
  - Making a dedicated core instead of a unified core for several schemes

[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

- SW architecture
  - **2019:** Botros et al., Memory-efficient high-speed implementation
  - **2020:** Alkim et al., Cortex-M4 optimization for {R, M} LWE
- HW/SW co-design architecture
  - **2019:** Basu et al., HLS-based architecture
  - **2019:** Barenjee et al., Targeting power consumption optimization
  - **2020:** Fritzmann et al., Tightly coupled RISC-V accelerator
  - **2020:** Alkim et al., Lightweight design
  - **2020:** Xin et al., High-performance architecture
- HW architecture
  - **2020:** Dang et al., Based on RTL methodology
  - **2020:** Huang et al., Relying on memory units
  - **2021:** Xing et al., Compact hardware implementation
  - **2021:** Bisheh-Niasar et al., Highly optimized NTT core
  - **2021:** Bisheh-Niasar et al., Instruction-set architecture



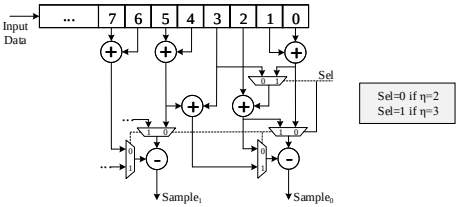
Performance and resource utilization comparison in three different Kyber implementation approaches: software (SW), hardware/ software (HW/SW), and hardware (HW).

Our Contributions

- CRYSTALS-Kyber KEM on the application-specific integrated circuit (ASIC) platform
- Increasing the efficiency on the NTT core:
  - Propose a reconfigurable architecture for NTT/INTT
  - Increase throughput using fewer resources
  - Reduce the overall area and memory consumption
- Highly parallel architecture in polynomial sampling cores
  - Absorb the latency of Keccak core
- Performance improvement for KEM coprocessor architecture
  - Perform key generation, encapsulation, and decapsulation in 21.3, 33.8, and 50 μs
  - 96% efficiency improvement compared to the best previous work on ASIC

Sampling Units

- Keccak:
  - Serial-in parallel-out (SIPO) buffer for Keccak input
  - parallel-in serial-out (PISO) for Keccak output
- Binomial Sampling:
  - Optimized architecture for both values of η
- Rejection Sampling:
  - Constant rounds to form a constant-time implementation



Configurable Binomial Sampling Unit

Total Round	Keccak Outputs (bit)	Total Samples	Required Valid Sample	Failure Probability
3	4,032	336	256	0.0083
4	5,376	448	256	2.2E-32
5	6,720	560	256	2.3E-79

Failure probabilities in Kyber rejection sampling for performing different Keccak rounds

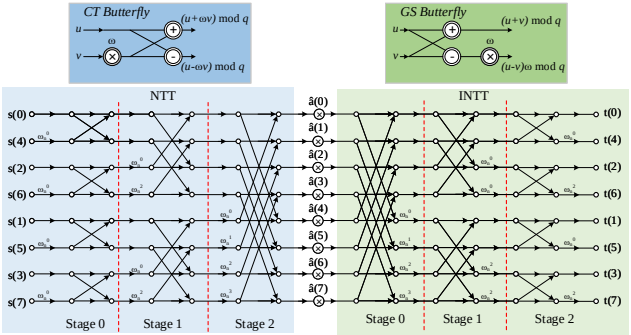
Polynomial Multiplication

- Number Theoretic Transform (NTT)

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

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

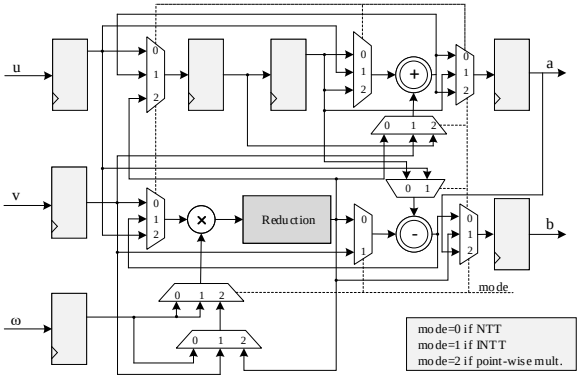
- Polynomial Multiplication $\longrightarrow$ $f.g = \text{INTT}(\text{NTT}(f) \circ \text{NTT}(g))$



Dataflow graph includes 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.

Configurable Butterfly Core

- NTT architecture includes:
  - Two RAM blocks
  - An address generator
  - A pre-computed twiddle factor ROM
  - An arithmetic unit
- Avoid the bit-reverse cost in polynomial multiplication
  - *mode* $= 0 \rightarrow$ NTT using CT configuration
  - *mode* $= 1 \rightarrow$ INTT using GS configuration
  - *mode* $= 2 \rightarrow$ Point-wise multiplication
- Reduce the execution time to $\frac{n}{2}\log\frac{n}{4}$
- Implement doubled bandwidth scheme



Proposed configurable butterfly architecture

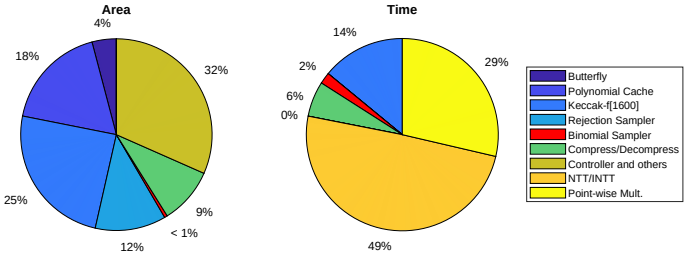
Implementation Results

- VHDL as the design entry to the Synopsys Design Compiler
- Total time includes Encaps + Decaps, as the key generation can be done offline.
- NAND gate in 65-nm library $= 1.35\ \mu m^2$
- Implementation results for Kyber KEM on 65-nm ASIC:

Protocol	Area		Freq	Cycles			Total Time
	Logic Gates [kGE]	SRAM [kB]	[MHz]	KeyGen [CCs]	Encaps [CCs]	Decaps [CCs]	[μs]
Kyber-512	95	10	200	4,267	6,769	10,015	83.9
Kyber-768	93	22	200	6,641	9,683	13,569	116.3
Kyber-1024	104	24	200	9,971	13,278	17,676	154.8

Implementation Results

- Keccak:
  - The most resource consuming block: 25%
  - Compact version: more delay in sampling operations
- Butterfly unit:
  - The most time consuming operation
  - Lightweight arithmetic: 4%
- Sampling units:
  - 13% of total resources
  - Hidden latency with parallel architecture



Area breakdown (left) and time breakdown (right) during encapsulation of Kyber-512.

NTT Comparison

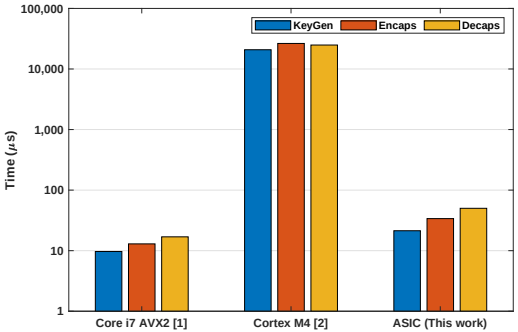
❑ Comparison with existing hardware-based implementations of NTT for Kyber KEM:

Work	Platform	Tech [nm]	Freq [MHz]	NTT [CCs]	INTT [CCs]	Point-wise Mult. [CCs]
Karabulut et al.	Virtex-7	-	NA	43,756 (92×)	NA	NA
Alkim et al.	Artix-7	-	59	6,868 (14×)	6,367	2,395
Chen et al.	Artix-7	-	130	2,055 (4.3×)	NA	7,197
Huang et al.	Artix-7	-	155	1,834 (3.9×)	NA	NA
Bisheh-Niasar et al.	Artix-7	-	222	324	324	NA
Fritzmann et al.	ASIC	65	25	2,056 (4.3×)	NA	NA
Fritzmann et al.	ASIC	65	45	1,935 (4.0×)	1,930	NA
Banerjee et al.	ASIC	40	72	1,289 (2.7×)	NA	NA
Xin et al.	ASIC	28	300	41	NA	NA
This Work	**ASIC**	**65**	**200**	**474**	**602**	**1,289**

- Xin et al.: 17.3× speedup consuming 137× more resources
- Bisheh-Niasar et al.: 31% performance improvement occupying a 2×2 butterfly units

Comparison with state-of-the-art

❑ Comparison with Kyber-512 implementation on other platforms:



❑ Proposed ASIC architecture is:

- 2.8× slower than Intel Core i7 CPU at 3,492 MHz
- 600× faster than Cortex-M4 Discovery board at 24 MHz

[1] Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehle, D.: CRYSTALS-Kyber: Algorithm specification and supporting documentation (version 3.0). submission to the NIST postquantum cryptography standardization project, (2020)
[2] Botros, L., Kannwischer, M.J., Schwabe, P.: Memory-efficient high-speed implementation of Kyber on Cortex-M4. In: Progress in Cryptology - AFRICACRYPT 2019 - 11th International Conference on Cryptology in Africa, Rabat, Morocco, July 9-11, 2019, Proceedings. 209–228 (2019)

Comparison with state-of-the-art

❑ ASIC Implementation results for Kyber KEM:

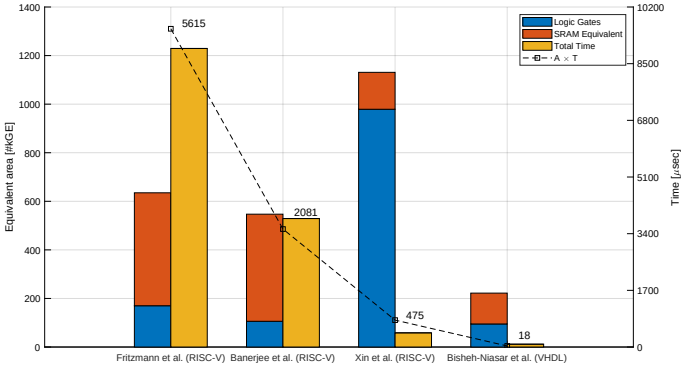
Work	Tech	Area		Total Area†	Freq	Latency			Total Time	$A \times T$
		Logic Gates	SRAM			KeyGen	Encaps	Decaps		
	[nm]	[kGE]	[kB]	[kGE]	[MHz]	[kCCs]	[kCCs]	[kCCs]	[μs]	[GE×s]
Kyber-512										
Basu et al.	65	1,341	-	3,531	200	-	-	43	-	-
Fritzmann et al.	65	170	465§	635	45	150	193	205	8,844	5,615
Banerjee et al.	40	106	40.25	547	72	75	132	142	3,806	2,081
Xin et al.	28	979	12	1,131	300	19	46	80	420	475
This work	**65**	**95**	**10**	**222**	**200**	**4**	**7**	**10**	**84**	**18**
Kyber-768										
Fritzmann et al.	65	170	465§	635	45	273	326	340	14,800	9,398
Banerjee et al.	40	106	40.25	547	72	112	178	191	5,125	2,803
This work	**65**	**93**	**22**	**372**	**200**	**7**	**10**	**14**	**116**	**43**
Kyber-1024										
Fritzmann et al.	65	170	465§	635	45	350	405	425	18,444	11,711
Banerjee et al.	40	106	40.25	547	72	149	223	241	6,444	3,524
Xin et al.	28	979	12	1,131	300	40	82	136	727	822
This work	**65**	**104**	**24**	**409**	**200**	**10**	**14**	**18**	**155**	**63**

† The total area is calculated based on the reported fabric dimension corresponding to their technology.

§ The reported numbers are in kGE.

Comparison with state-of-the-art

- Comparison with other AISC implementations of Kyber-512:



- Comparison to the best previous works
  - Achieving 96% efficiency improvement
  - Achieving 5× speedup

Comparison with state-of-the-art

❑ Comparison with existing PQC implementations in NIST security level 1:

Protocol	Platform	Area (Gates Equivalent) or (LUTs/ FFs/ Slices/ DSPs/ BRAMs)	Freq [MHz]	Time [us]
SIKEp434 [1]	Virtex-7	12,818/ 18,271/ 5,527/ 195/ 32	249.6	8,800
Frodo-640 [2]	Artix-7	6,881/ 5,081/ 1,947/ 16/ 12.5	149	2,621
LightSaber [3]	ASIC	742 kGE[‡]	400	5
Kyber-512 [This work]	**ASIC**	**222 kGE**	**200**	**84**

[‡] The reported area is 0.38 mm^2 in 40 nm process.

[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] Zhu, Y., Zhu, M., Yang, B., Zhu, W., Deng, C., Chen, C., Wei, S., Liu, L.: A high-performance hardware implementation of saber based on karatsuba algorithm. Cryptology ePrint Archive, Report 2020/1037, 2020

Conclusion and future works

❑ Conclusion:

- Implementing a configurable butterfly core
- Reducing the latency with highly parallel architecture
- Improving 96% efficiency in terms of A$\times$T
- Performing all KEM operations for Kyber:
  - key generation, encapsulation, and decapsulation in 21.3, 33.8, and 50 μs

❑ Future work:

- Extending the design by side-channel countermeasures

Thanks for your attention.