Towards Efficient Kyber on FPGAs: A Processor for Vector of Polynomials

Zhaohui Chen[1,2] , Yuan Ma[2]*, Tianyu Chen[2], Jingqiang Lin[2] and Jiwu Jing[1]

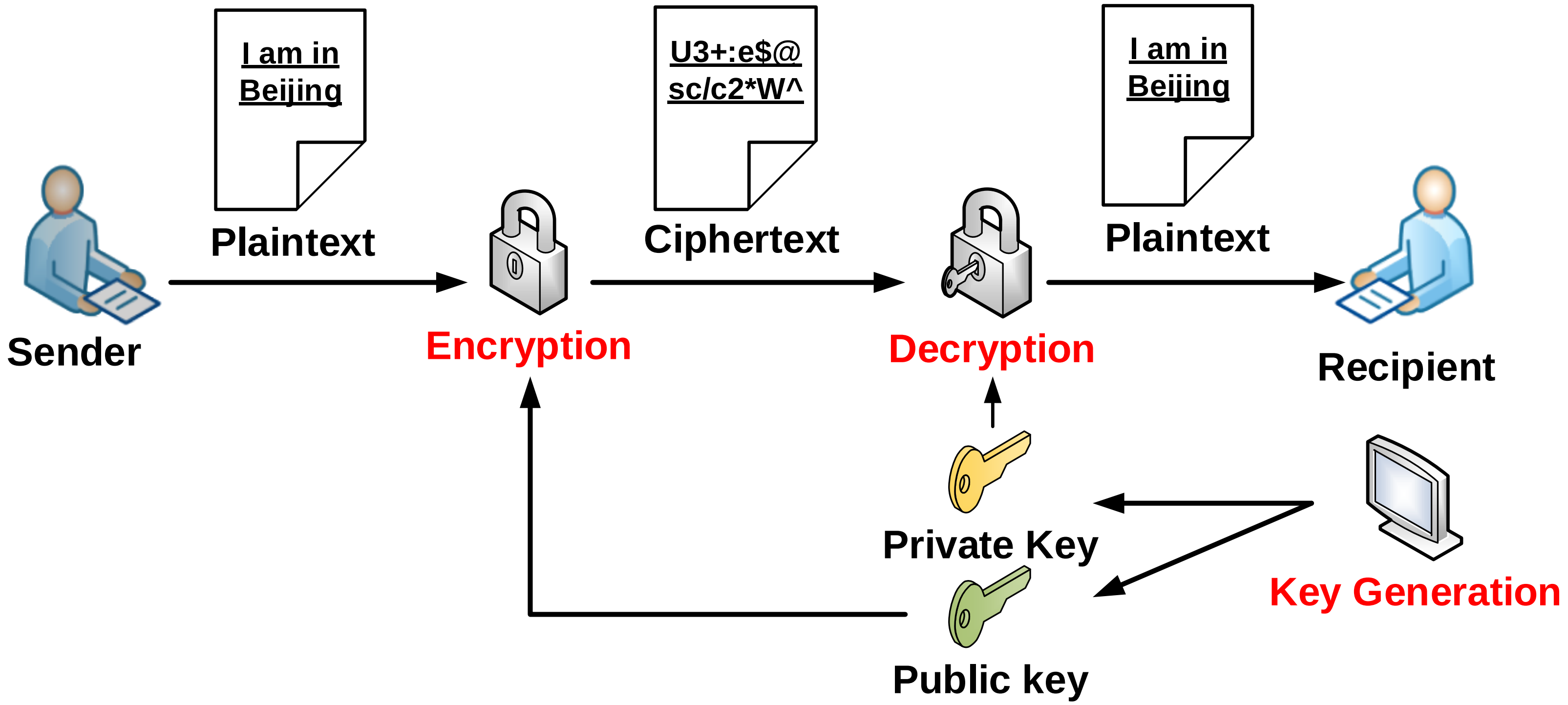
[1]School of Computer Science and Technology, University of Chinese Academy of Sciences
[2]State Key Laboratory of Information Security, Institute of Information Engineering, CAS

Beijing, China

15. Jan. 2020

Public Key Cryptography

- **Different** keys are used for encryption and decryption
- Algorithm triples {**KeyGen, Encryption, Decryption**}
- Standardized algorithm (Classic)
  - RSA, see PKCS#1, ANSI X9.31, IEEE 1363



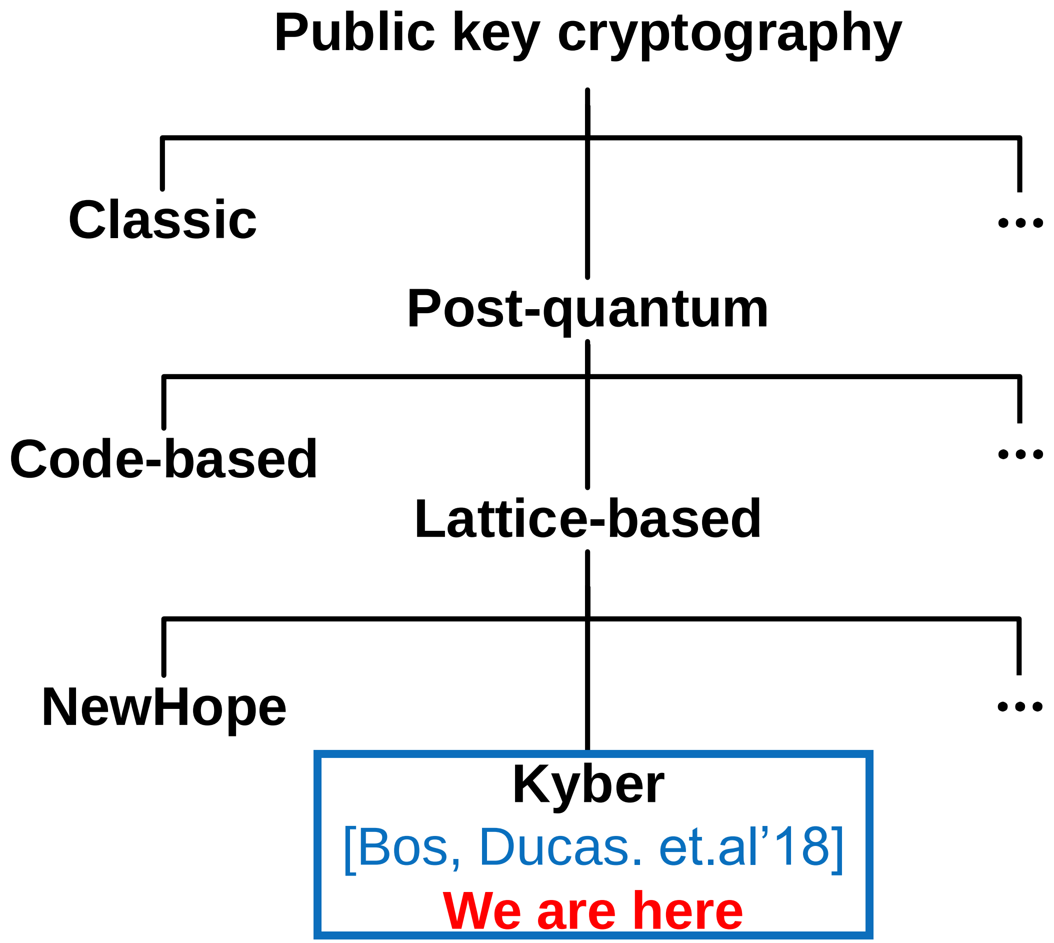
An Imminent Threat

- Quantum algorithms
  - Shor's algorithm needs about 1 billion qubits [Shor'97]
  - 20 million qubits break RSA in 8 hours [Gidney'19]
- Quantum computers
  - In May 2017, **USTC** developed a **10**-qubit circuit
  - In Oct. 2017, **Intel** announced a **17**-qubit chip
  - In Nov. 2017, **IBM** established a **50**-qubit computer
  - In Mar. 2018, **Google** announced a **72**-qubit chip
  - In Aug. 2018, **Rigetti Computing** plans to build and deploy a **128**-qubit system

Algorithm optimization space and computing power growth should be considered

New Cryptography Schemes

- Post-quantum cryptography = Quantum-resistant
  - ➢ Kyber is a new scheme based on Module-LWE problem



A Brief Introduction to Kyber

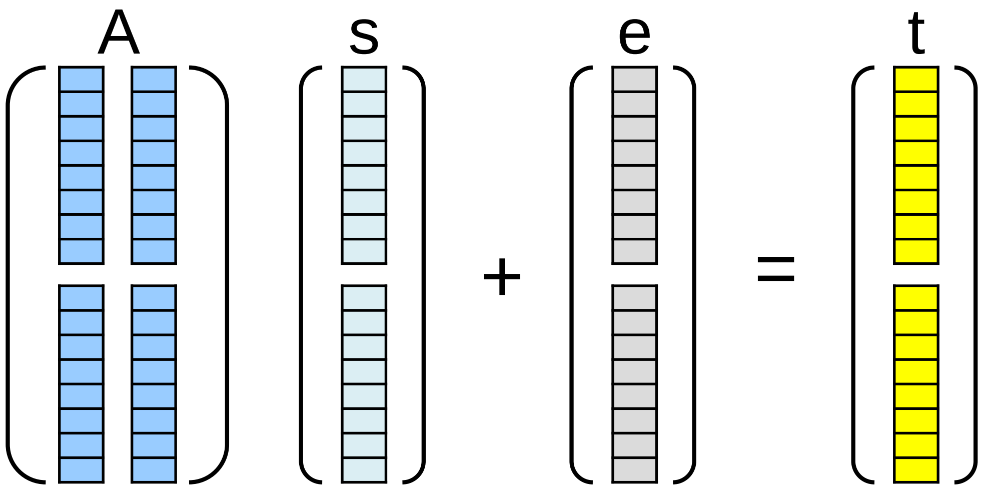
- **Kyber.*KeyGen(A)*:** Choose two polyvec s, e from β^k_η and compute $\boxed{t = As + e}$. The public key is (A, t) and the private key is s.
- **Kyber.*Enc(A, t, m)*:** The message m is first encoded to $\overline{m}$. Sample polyvec r, e_1 from β^k_η and e_2 from β_η. The ciphertext then consists of polyvec $\boxed{u = A^T r + e_1}$ and polynomial $v = t^T r + e_2 + \overline{m}$.
- **Kyber.*Dec(s, u, v)*:** Compute $m' = v - s^T u$ and recover the original message m from m' using a decoder.

- Computing over **vector of polynomials** (polyvec)
  - 13-bit coefficients, modulo 7681
  - n-dimensional polynomial, n=256
  - k-dimensional vector, depends on security level (2,3,4)
- Example, **As + e** is computational intensive

Multiplication over polyvec

- Example: for Kyber512, k=2
  - t = As + e

$$\mathbf{As} = \begin{bmatrix} \mathbf{A}(0,0)\,\mathbf{s}(0) + \mathbf{A}(0,1)\,\mathbf{s}(1) \\ \mathbf{A}(1,0)\,\mathbf{s}(0) + \mathbf{A}(1,1)\,\mathbf{s}(1) \end{bmatrix}$$



in the figure, n=8

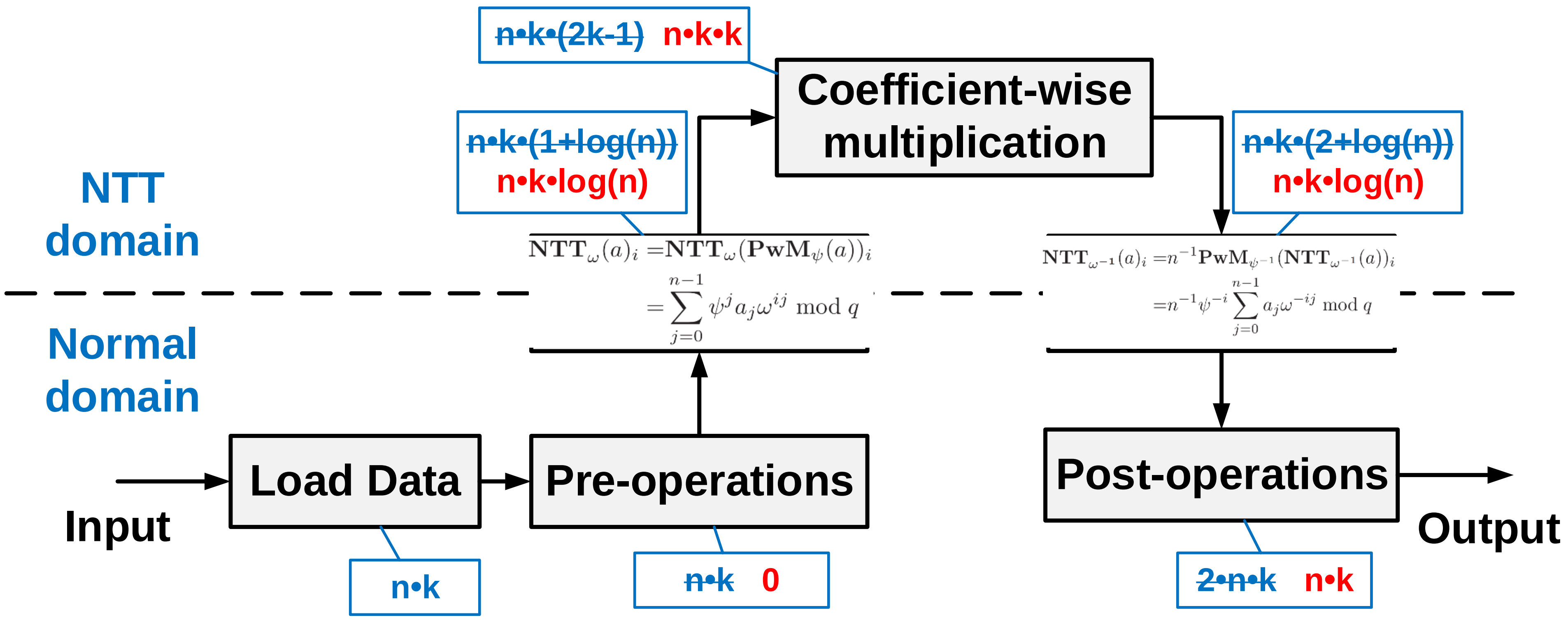
- Number-Theoretic Transformation (NTT) [Po.'12]
  - Execute NTT transformation for k polynomials respectively in a polyvec.

Overview

- **Optimizing the Control Logic**
  - Merge operations to avoid idle cycles
- **Optimizing Memory Access Scheme**
  - Dual-column sequential storage structure
  - in-place computation without bit-reversal
- **Optimizing Arithmetic Logic Unit**
  - Gentlemen-Sande (GS) butterfly
  - Integrate NTT, multiply-accumulate and multiply-add
  - Pipelined implementation

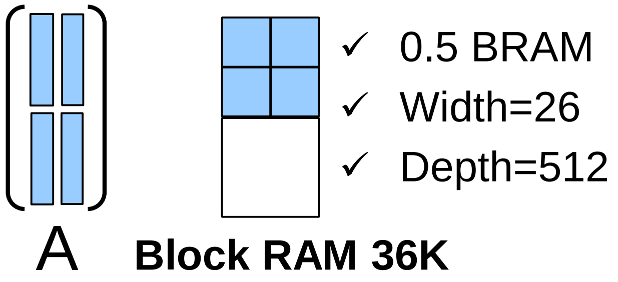
Optimizing Arithmetic Operations

- By adjusting control logic
  - Save 29.4% cycles for Kyber512 (k=2)
  - Save 33.3% cycles for Kyber1024 (k=4)

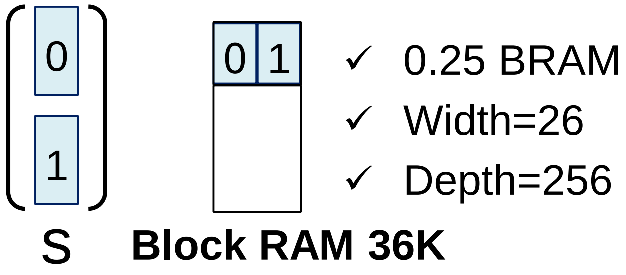

~~n•k•(2k-1)~~ n•k•k
Coefficient-wise multiplication
~~n•k•(1+log(n))~~ n•k•log(n)
~~n•k•(2+log(n))~~ n•k•log(n)
NTT domain
Normal domain
$\mathbf{NTT}_\omega(a)_i = \mathbf{NTT}_\omega(\mathbf{PwM}_\psi(a))_i = \sum_{j=0}^{n-1} \psi^j a_j \omega^{ij} \bmod q$
$\mathbf{NTT}_{\omega^{-1}}(a)_i = n^{-1}\mathbf{PwM}_{\psi^{-1}}(\mathbf{NTT}_{\omega^{-1}}(a))_i = n^{-1}\psi^{-i}\sum_{j=0}^{n-1} a_j \omega^{-ij} \bmod q$
Input
Load Data
Pre-operations
Post-operations
Output
n•k
~~n•k~~ 0
~~2•n•k~~ n•k

Optimizing Memory Access Scheme

- Break the bottleneck
  - Read 2 coefficients and writeback 2 coefficients
  - Block RAM slice can be configured as
    - 2 read ports
    - 2 write ports
    - 1 read and 1 write ports
- Dual-column sequential scheme
  - Increase memory width
  - Example, Kyber512 (k=2)



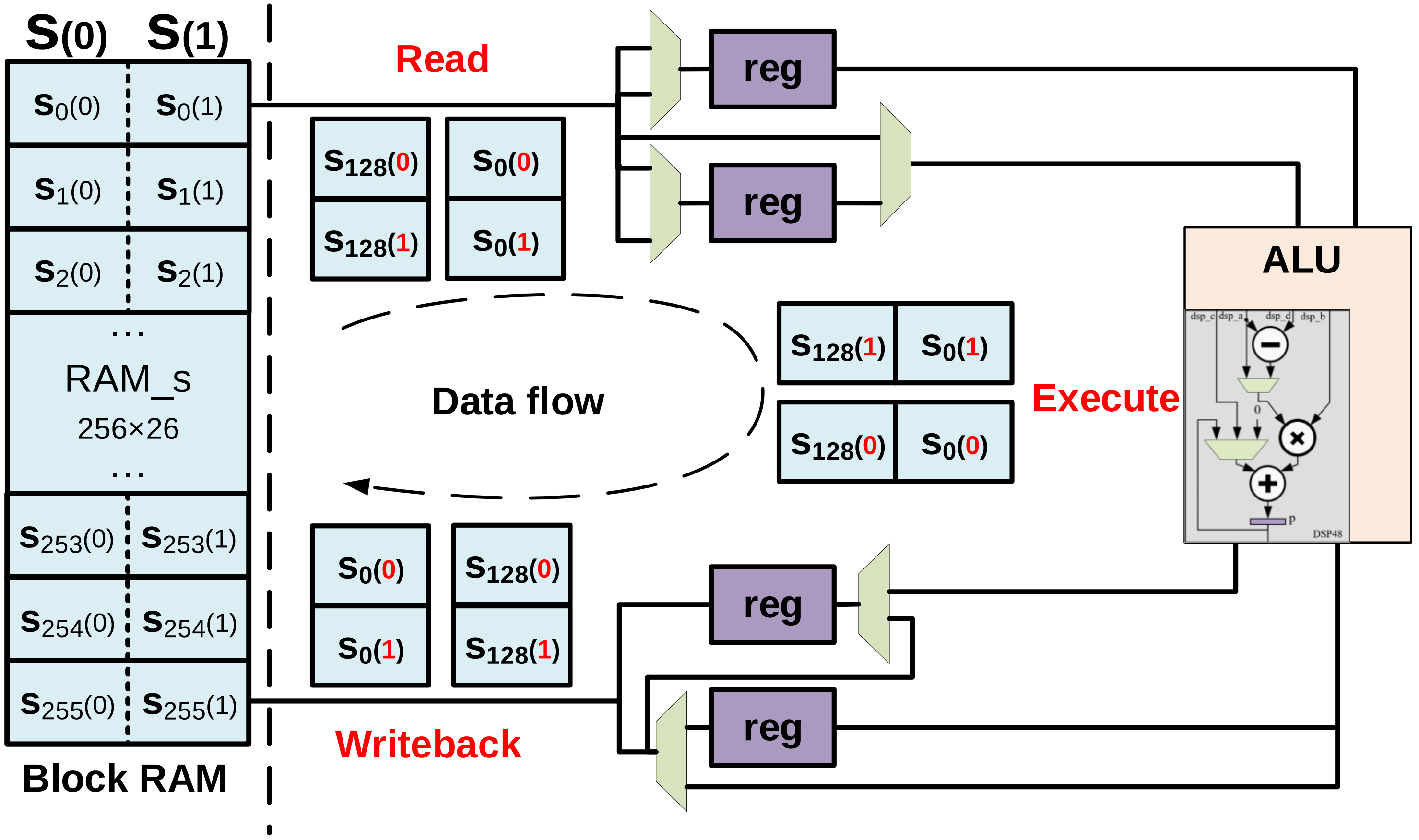
- ✓ 0.5 BRAM
- ✓ Width=26
- ✓ Depth=512



- ✓ 0.25 BRAM
- ✓ Width=26
- ✓ Depth=256

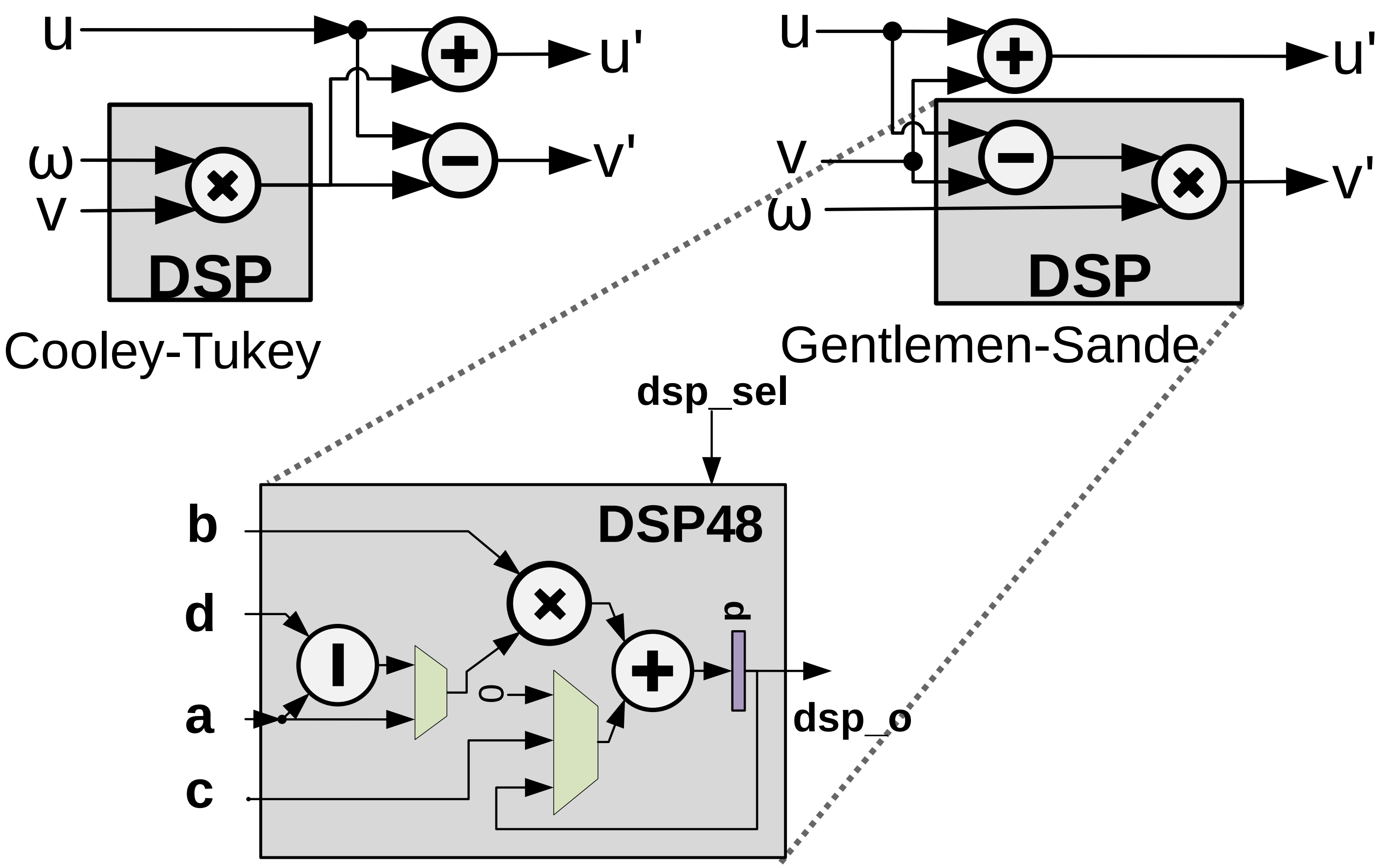
Optimizing Memory Access Scheme

- In-place NTT transformation
  - Transform 2 polynomials with time-multiplexed ALU
  - Swap data before and after execution



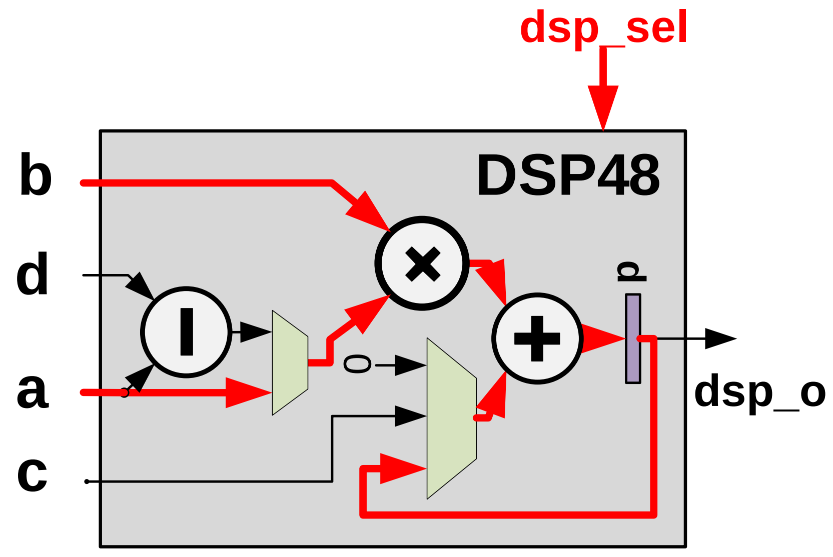
Optimizing Arithmetic Logic Unit

- Make full use of the DSP slice
  - Cooley-Tukey vs. Gentlemen-Sande butterfly

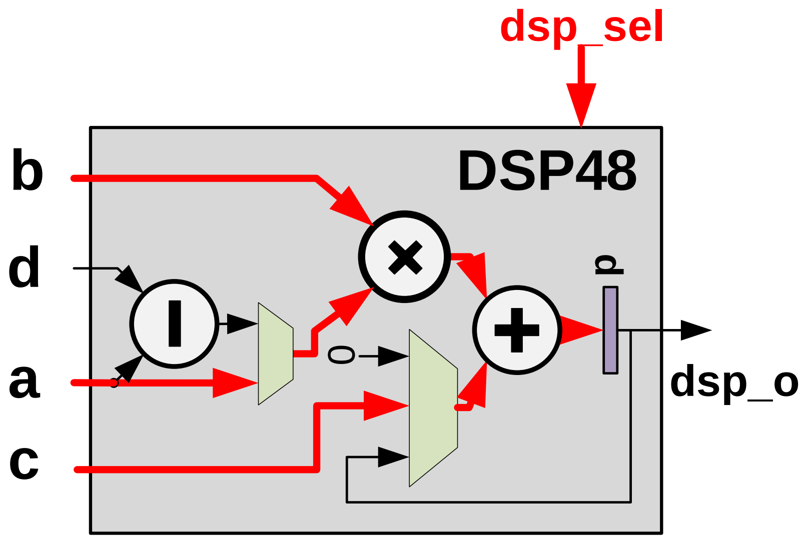


Optimizing Arithmetic Logic Unit

- Integrate functions in one processor
  - Forward NTT and inverse NTT transformation
  - Multiply-accumulate over polyvec, like $\mathbf{a_0 \circ s_0 + a_1 \circ s_1}$
  - Multiply-add over polyvec, like $\mathbf{a_0 \circ s_0 + e_0}$



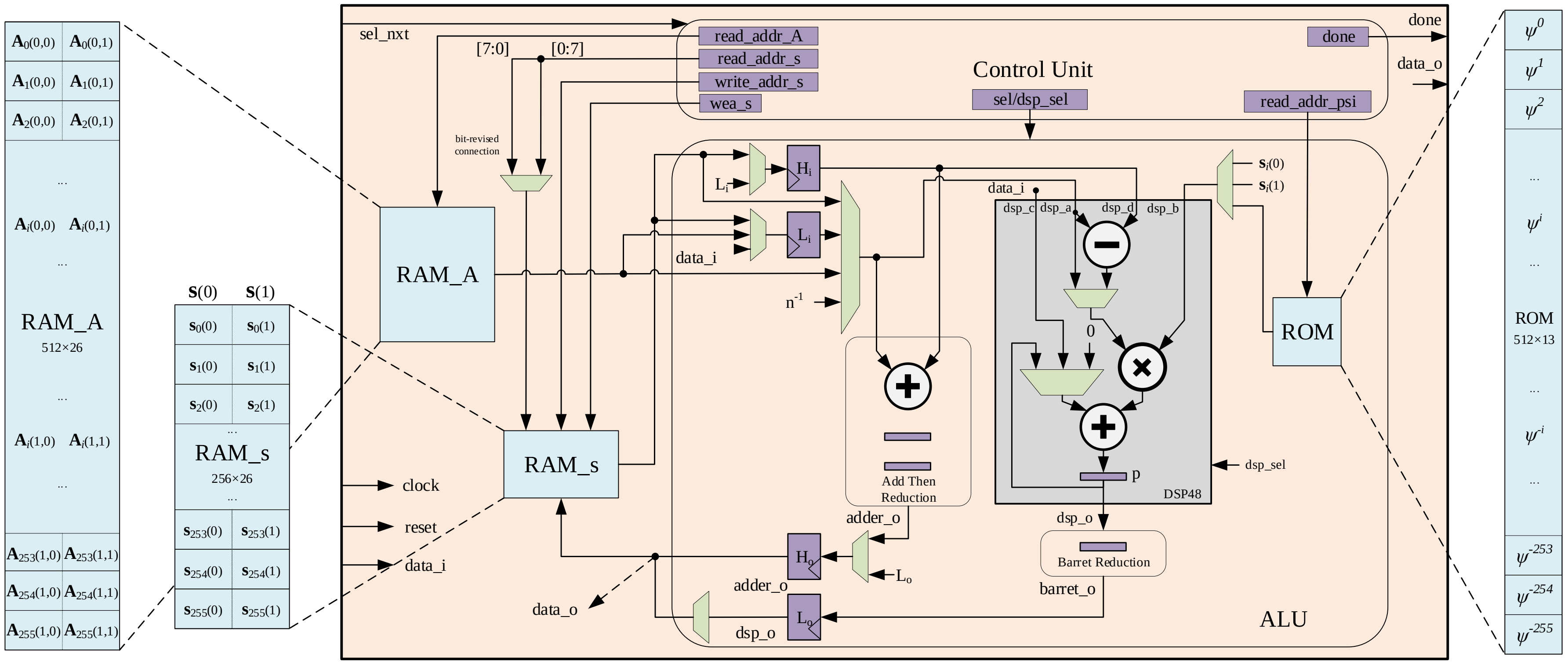
Multiply-accumulate



Multiply-add

Optimizing Arithmetic Logic Unit

- Design a pipelined structure
  - Maximize the usage of DSP/BRAM internal registers
  - Trade-off between area and performance



Implementation on FPGAs

- Experimental Setup
  - Implemented in Verilog HDL available at:
    - https://github.com/cccisi/Kyber_ASPDAC
    - Built on XILINX Vivado® 2018.2 design suite for Artix-7
    - Also built on XILINX ISE ® 14.7 design suite for Spartan-6
  - Post-place and route
- Results
  - Maximum frequency
    - 130MHz for Kyber1024
    - 31684 NTT operations per second
  - Area
    - The processor occupies 477 LUTs and 237 FFs

Implementation on FPGAs

■ Comparisons on NTT efficiency

Processor	Area			Operations/s		Efficiency
	LUTs	FFs	DSPs			
This Work	477	237	1	NTT:	31684	66.4
				MUL:	9050	19.0
Oder, Güneysu’17	415	251	2	NTT:	3487	9.3
				MUL:	1545	3.7
Kuo, Yang’17	2832	1381	8	NTT:	59337	21.9
				MUL:	NA	NA

- Compared with [Oder, Güneysu’17]
  - Integrated more functions
  - Over 5x advantages in performance
- Compared with [Kuo, Yang’17]
  - Over 55.2% performance with about 17.2% logical units

Conclusion

- We achieved high-efficiency post-quantum algorithm on resource-constrained hardware
- Broke memory bottleneck for higher performance
- Made full use of FPGA internal resources
- Implemented the key modules for Kyber512/1024, but **NOT** the entire Kyber cryptosystem

Future works

- Countermeasures against possible attacks, side-channel attack …
- Which post-quantum algorithms are the outstanding ones in practical applications?

Thank You For Your Attention!

中国科学院大学

University of Chinese Academy of Sciences

Reference

- [Shor’97] P. W. Shor, “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer,” SIAM J. Comput., vol. 26, no. 5, pp. 1484–1509, 1997.
- [Gidney’19] Gidney, Craig, and Martin Ekerå. "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits." *arXiv preprint arXiv:1905.09749*, 2019.
- [Bos, Ducas. et.al’18] J. W. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehle, “CRYSTALS - Kyber: A CCA-secure module-lattice-based KEM,” in IEEE European Symposium on Security and Privacy, EuroS&P, pp. 353–367, 2018.
- [Poppelmann, Güneysu’12] T. Poppelmann and T. Güneysu, “Towards efficient arithmetic for lattice-based cryptography on reconfigurable hardware,” in LATINCRYPT 2012, pp. 139–158, 2012.
- [Oder, Güneysu’17] T. Oder and T. Güneysu, “Implementing the NewHope-Simple key exchange on low-cost FPGAs,” in LATINCRYPT, 2017.
- [Kuo, Yang’17] P.-C. Kuo, W.-D. Li, Y.-W. Chen, Y.-C. Hsu, B.-Y. Peng, C.-M. Cheng, and B.-Y. Yang, “High performance post-quantum key exchange on FPGAs,” IACR Cryptology ePrint Archive, p. 690, 2017.