FPGA based Hardware Accelerator for Lattice Based Cryptography

Debapriya Basu Roy

Department of Computer Science and Engineering,
Indian Institute of Technology Kanpur, India

SPACE 2024

Hardware Acceleration

How can we deploy ?

Cryptographically secure algorithms are proposed.

Involves large number multiplication

Involves operations on polynomials

Slow !

Software Implementations

Large integer multiplication, modular inversion in ECC (~256 bits or larger), Polynomial multiplication in PQC

Optimized software implementation of ECC: 1816351 CCs
Hardware implementation of ECC: 10465 CCs.

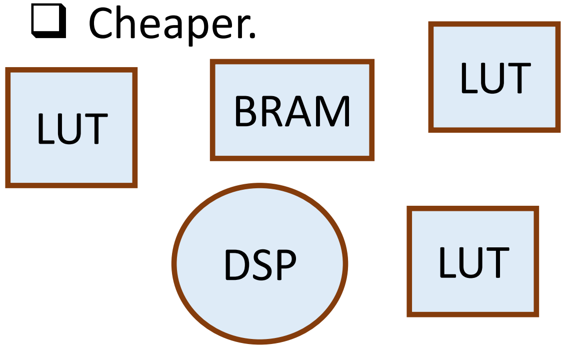
Software implementation of PQC: 145687 CCs.
Hardware implementation of PQC: ~6000 CCs.

Fast !

Hardware Acceleration

FPGA

- Programmable Logic Blocks.
- Cheaper.



ASIC

- Fast.
- Expensive and time consuming process.
- Error in the design will cause retention of entire process.

Koppermann, Philipp, et al. "Low-latency X25519 hardware implementation: Breaking the 100 microseconds barrier." *Microprocessors and Microsystems* 52 (2017): 491-497.

Sanal, Pakize, et al. "Kyber on ARM64: Compact implementations of Kyber on 64-bit ARM Cortex-A processors." *International Conference on Security and Privacy in Communication Systems*. Cham: Springer International Publishing, 2021.

Introduction to Classical Cryptography

Clever Person With Powerful Computer

Alice

M Public Network

Bob

Key K

Symmetric Key Cryptography

0 1 0 .... 1 0

Uniformly Random Bits

Public Key Cryptography

Public Key P_1

Private key P_2

<u>Factorization</u>

$N = p \cdot q$

p and q are large primes.

<u>Discrete Logarithm Problem</u>

Given g^X, find X.

G is a cyclic group.

Quantum Computing: Threat to Public Key Cryptography

Progress on Quantum Computing

2018
72 qubit Bristlecone processor

→

2019
Quantum Supremacy achieved by Google

→

2020-21
China claims to have Quantum Supremacy

→

2023
IBM’s 1123 qubit “Condor” processor

→

?

Progress on Post-Quantum Cryptography

2017
Round 1: 69 Submissions

→

2019:
Round 2: 26 Submissions

→

2020
Round 3: 15 Submissions

→

2022
Round 4: 8 Submissions and Attack on SIKE

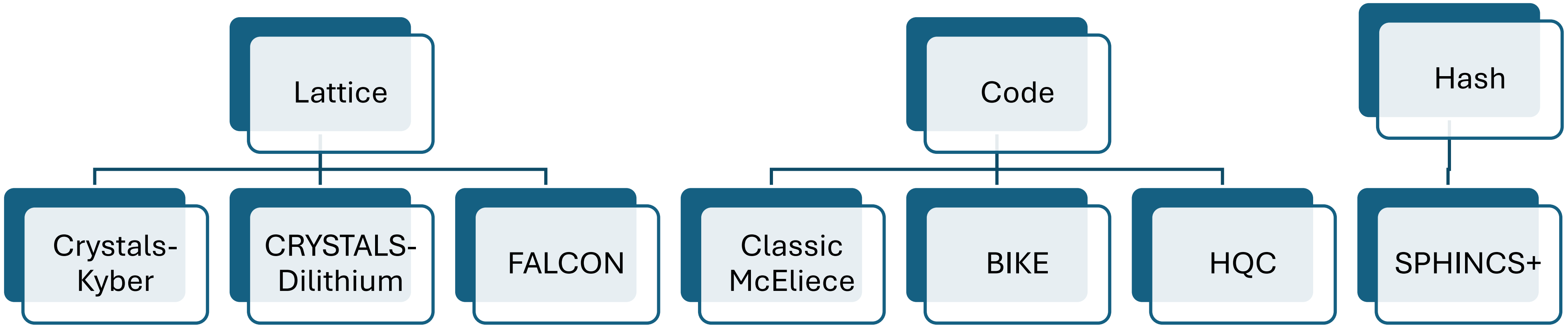
→

2024: PQC Standards
FIPS-203, FIPS-204, FIPS-205

NIST PQC Standardization

Key Encapsulation Mechanism (KEM)	Digital Signature	Other Candidates for KEM
CRYSTALS-KYBER (FIPS-203)	CRYSTALS-Dilithium (FIPS-204)	BIKE
	FALCON	Classic McEliece
	SPHINCS+ (FIPS-205)	HQC

The PQC Algorithms of Immediate Interest



Classification of PQC Algorithms

NIST Short Signature PQC Algorithm Call: https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures

Short Signature Algorithms: Round 2 (Released on 25th October, 2024)

Code and Lattice Based

- CROSS
- LESS
- HAWK

Multivariate

- MAYO
- QR-UOV
- SNOVA
- UOV

MPC in the Head

- Mirath
- MQQM
- PERK
- RYDE
- SDith

Symmetric

- FAEST

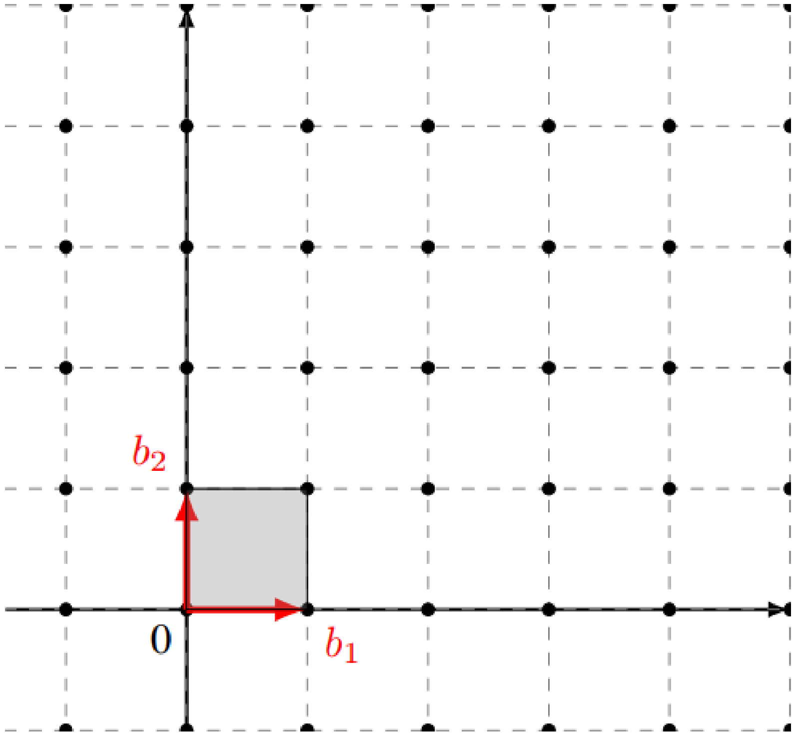
Isogeny

- SQIsign

Lattice

Let $\{v_1, v_2, v_3 \dots, vn\} \in R^m$ be a set of linearly independent vectors. The lattice **L** generated by $\{v_1, v_2, v_3 \dots, vn\}$ is the set of integer linear combinations of $\{v_1, v_2, v_3 \dots, vn\}$.

$$L = \{a_1 v_1 + \cdots a_n v_n | a_1, \cdots a_n \in \mathcal{Z}\}$$



- If m =n, the corresponding lattice is a full rank lattice
- Lattice is closed under addition
- $\{v_1, v_2, v_3 \dots, vn\}$ is the basis of the lattice.
- Mathematical hard problems that are based on lattice:
  - Shortest Vector Problem
  - Closest Vector Problem
  - **Learning with error**

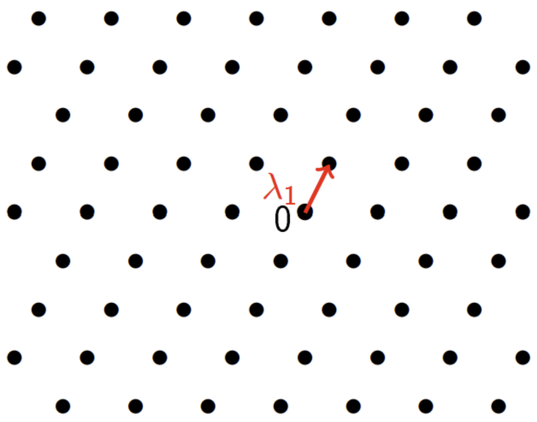
Example of Lattice

$$n = 2, \qquad v_1 = \begin{bmatrix} 4 \\ 3 \end{bmatrix}, \qquad v_2 = \begin{bmatrix} -3 \\ 2 \end{bmatrix}, \qquad Lattice\ L\colon\ (a_1 v_1 + a_2 v_2), \qquad where\ a_1\ and\ a_2\ are\ integers.$$

$$If\ a_1 = 0, L = \begin{bmatrix} -3 \\ 2 \end{bmatrix}, \qquad If\ a_2 = 0, L = \begin{bmatrix} 4 \\ 3 \end{bmatrix}, \qquad If\ a_1 = 1, a_2 = 1, L = \begin{bmatrix} 1 \\ 5 \end{bmatrix}$$

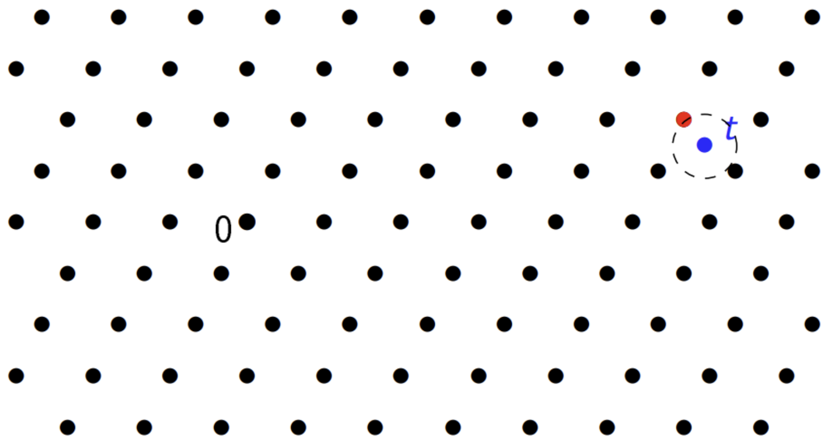
$$Lattice\ L\ can\ also\ be\ written\ as\colon \begin{bmatrix} 4 & -3 \\ 3 & 2 \end{bmatrix} . \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$$

Closest Vector Problem



Find a shortest (in Euclidean norm) non-zero vector $\|\boldsymbol{x}\|_2 := \sqrt{x_1^2 + \cdots + x_n^2}.$

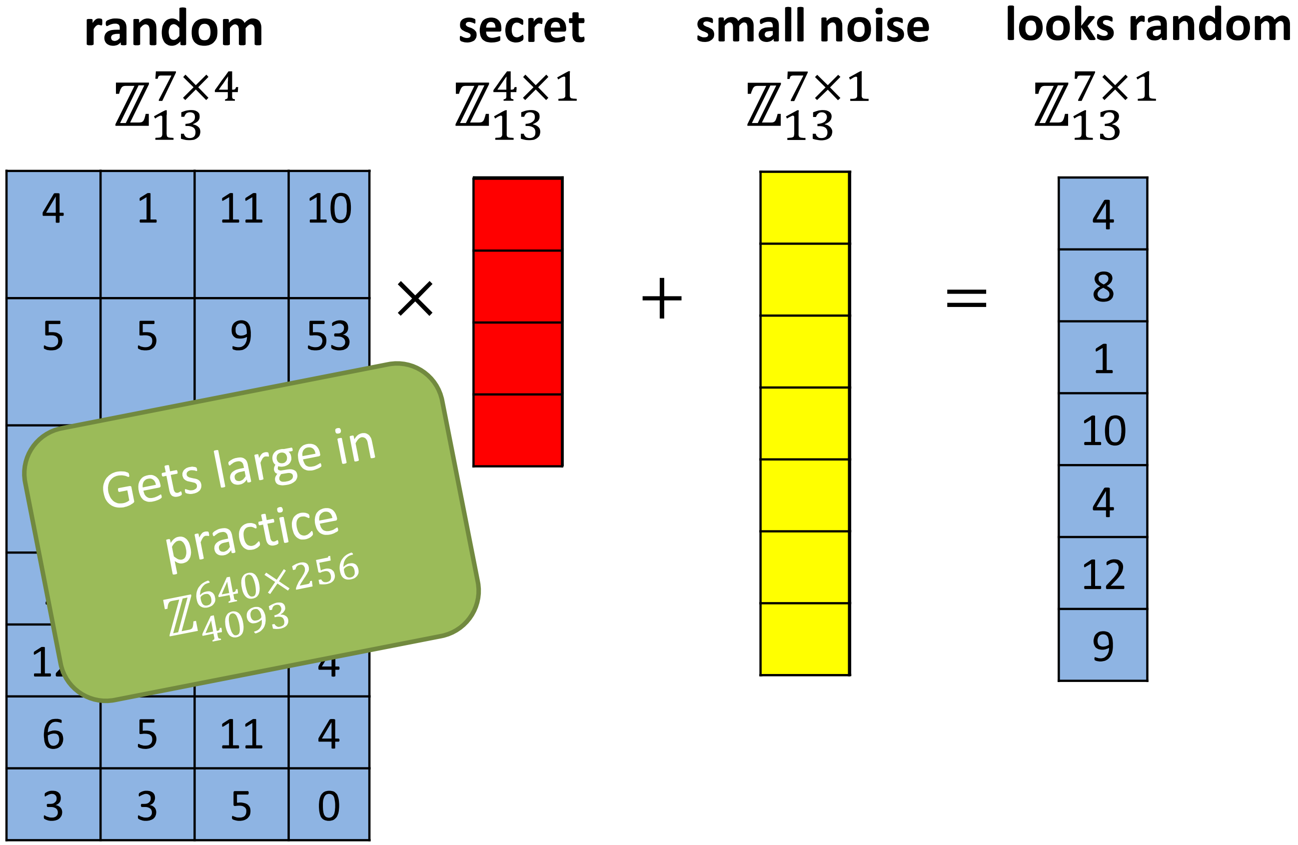
Shortest Vector Problem



Given a target point t, find a point of the lattice closest to t

https://apelletm.pages.math.cnrs.fr/page-perso/documents/presentations/lattice_based_crypto.pdf

Learning With Error (LWE)



Blue is given; Find **red** ➔ Learning with Errors (LWE) Problem

Source: Tim Güneysu, Tutorial@CHES 2017 - Taipei

An Example of LWE PKE (*k=2*, *q=17*, and *n=4*)

Key-Generation

t = A * s + e

Private: s | Public: t, A

Sampler

e , r , e_1 , e_2, A

$$A = \begin{pmatrix} 6x^3 + 16x^2 + 16x + 11 & 9x^3 + 4x^2 + 6x + 3 \\ 5x^3 + 3x^2 + 10x + 1 & 6x^3 + x^2 + 9x + 15 \end{pmatrix} \quad s = (-x^3 - x^2 + x, -x^3 - x) \quad e = (x^2, x^2 - x)$$

$$t = (16x^3 + 15x^2 + 7, 10x^3 + 12x^2 + 11x + 6)$$

Encryption

u = A^Tr + e_1 **v = t^Tr + e_2 + m**

To encrypt a message m_b (11), we have to convert it to binary representation and then multiply with q/2.

$$m_b = 1x^3 + 0x^2 + 1x^1 + 1x^0 = x^3 + x + 1 \qquad m = \left\lfloor \frac{q}{2} \right\rceil \times m_b = 9x^3 + 9x + 9$$

$$r = (-x^3 + x^2, x^3 + x^2 - 1) \qquad e_1 = (x^2 + x, x^2) \qquad e_2 = -x^2 - x$$

$$\begin{aligned} u &= A^T r + e_1 \\ &= (11x^3 + 11x^2 + 10x + 3, 4x^3 + 4x^2 + 13x + 11) \end{aligned}$$

$$\begin{aligned} v &= t^T r + e_2 + m \\ &= (7x^3 + 6x^2 + 8x + 15) \end{aligned}$$

Decryption

$$m_n = v - s^T u$$

$$m_n = e^T r + e_2 + m + s^T e_1 \quad = 7x^3 + 14x^2 + 7x + 5$$

7 – Closest to 9 (q/2) or 1
14 – Closest to 17 (q) or 0
7 – Closest to 9 (q/2) or 1
5 – Closest to 9 (q/2) or 1

Decrypted Message: 1 0 1 1

https://medium.com/@hwupathum/crystals-kyber-the-key-to-post-quantum-encryption-3154b305e7bd

CRYSTALS-Kyber

Algorithm 4 KYBER.CPAPKE.KeyGen(): key generation

Output: Secret key $sk \in \mathcal{B}^{12 \cdot k \cdot n/8}$
Output: Public key $pk \in \mathcal{B}^{12 \cdot k \cdot n/8+32}$

$d \leftarrow \mathcal{B}^{32}$
$(\rho, \sigma) := \mathrm{G}(d)$
$N := 0$
for i from 0 to $k-1$ **do**
  for j from 0 to $k-1$ **do**
    $\hat{\mathbf{A}}[i][j] := \mathsf{Parse}(\mathsf{XOF}(\rho, j, i))$
  end for
end for
for i from 0 to $k-1$ **do**
  $\mathbf{s}[i] := \mathsf{CBD}_{\eta_1}(\mathsf{PRF}(\sigma, N))$
  $N := N + 1$
end for
for i from 0 to $k-1$ **do**
  $\mathbf{e}[i] := \mathsf{CBD}_{\eta_1}(\mathsf{PRF}(\sigma, N))$
  $N := N + 1$
end for
$\hat{\mathbf{s}} := \mathsf{NTT}(\mathbf{s})$
$\hat{\mathbf{e}} := \mathsf{NTT}(\mathbf{e})$
$\boxed{\hat{\mathbf{t}} := \hat{\mathbf{A}} \circ \hat{\mathbf{s}} + \hat{\mathbf{e}}}$
$pk := (\mathsf{Encode}_{12}(\hat{\mathbf{t}} \bmod^{+} q) \| \rho)$
$sk := \mathsf{Encode}_{12}(\hat{\mathbf{s}} \bmod^{+} q)$
return (pk, sk)

Algorithm 5 KYBER.CPAPKE.Enc(pk, m, r): encryption

Input: Public key $pk \in \mathcal{B}^{12 \cdot k \cdot n/8+32}$
Input: Message $m \in \mathcal{B}^{32}$
Input: Random coins $r \in \mathcal{B}^{32}$
Output: Ciphertext $c \in \mathcal{B}^{d_u \cdot k \cdot n/8 + d_v \cdot n/8}$

$N := 0$
$\hat{\mathbf{t}} := \mathsf{Decode}_{12}(pk)$
$\rho := pk + 12 \cdot k \cdot n/8$
for i from 0 to $k-1$ **do**
  for j from 0 to $k-1$ **do**
    $\hat{\mathbf{A}}^T[i][j] := \mathsf{Parse}(\mathsf{XOF}(\rho, i, j))$
  end for
end for
for i from 0 to $k-1$ **do**
  $\mathbf{r}[i] := \mathsf{CBD}_{\eta_1}(\mathsf{PRF}(r, N))$
  $N := N + 1$
end for
for i from 0 to $k-1$ **do**
  $\mathbf{e}_1[i] := \mathsf{CBD}_{\eta_2}(\mathsf{PRF}(r, N))$
  $N := N + 1$
end for
$e_2 := \mathsf{CBD}_{\eta_2}(\mathsf{PRF}(r, N))$
$\hat{\mathbf{r}} := \mathsf{NTT}(\mathbf{r})$
$\boxed{\mathbf{u} := \mathsf{NTT}^{-1}(\hat{\mathbf{A}}^T \circ \hat{\mathbf{r}}) + \mathbf{e}_1}$
$\boxed{v := \mathsf{NTT}^{-1}(\hat{\mathbf{t}}^T \circ \hat{\mathbf{r}}) + e_2 + \mathsf{Decompress}_q(\mathsf{Decode}_1(m), 1)}$
$c_1 := \mathsf{Encode}_{d_u}(\mathsf{Compress}_q(\mathbf{u}, d_u))$
$c_2 := \mathsf{Encode}_{d_v}(\mathsf{Compress}_q(v, d_v))$
return $c = (c_1 \| c_2)$

Continue ...

Algorithm 6 KYBER.CPAPKE.Dec(sk, c): decryption

Input: Secret key $sk \in \mathcal{B}^{12 \cdot k \cdot n/8}$
Input: Ciphertext $c \in \mathcal{B}^{d_u \cdot k \cdot n/8 + d_v \cdot n/8}$
Output: Message $m \in \mathcal{B}^{32}$

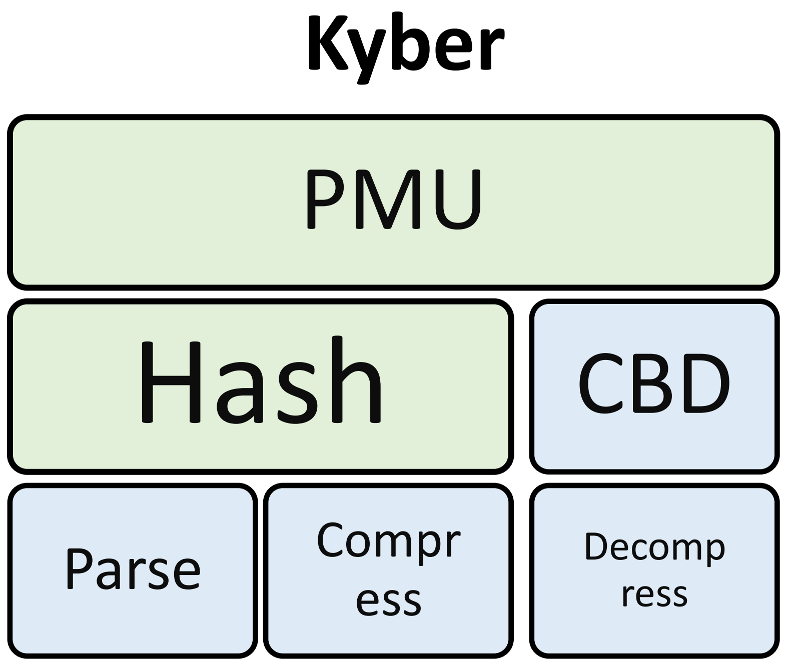
1: $\mathbf{u} := \mathsf{Decompress}_q(\mathsf{Decode}_{d_u}(c), d_u)$
2: $v := \mathsf{Decompress}_q(\mathsf{Decode}_{d_v}(c + d_u \cdot k \cdot n/8), d_v)$
3: $\hat{\mathbf{s}} := \mathsf{Decode}_{12}(sk)$
4: $m := \mathsf{Encode}_1(\mathsf{Compress}_q\boxed{(v - \mathsf{NTT}^{-1}(\hat{\mathbf{s}}^T \circ \mathsf{NTT}(\mathbf{u})), 1)})$ $\quad \triangleright\ m := \mathsf{Compress}_q(v - \mathbf{s}^T\mathbf{u}, 1))$
5: **return** m

$$
\begin{aligned}
v - s^T \cdot u &= t^T + e_2 + m - s^T (A^T r + e_1) \\
&= (As + e)^T r + e_2 + m - (As)^T r + s^T \cdot e_1 \\
&= \underline{(As)^T r - (As)^T r} + \underline{e^T \cdot r + e_2 + s^T e_1 + m} \\
&= m + (small)
\end{aligned}
$$

	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}

Security Category	Attack Type
1	Key search on a block cipher like AES-128.
2	Collision search on a 256-bit hash function like SHA256.
3	Key search on a block cipher like AES-192.

A Hardware Designer’s Perspective towards designing Kyber



- Faster and Lightweight Design.
- Majority of the operations involves **Polynomial Multiplication Unit and Hashing**.
- For large degree polynomial multiplication, **Number Theoretic Transformation (NTT)** is used.

Number Theoretic Transformation

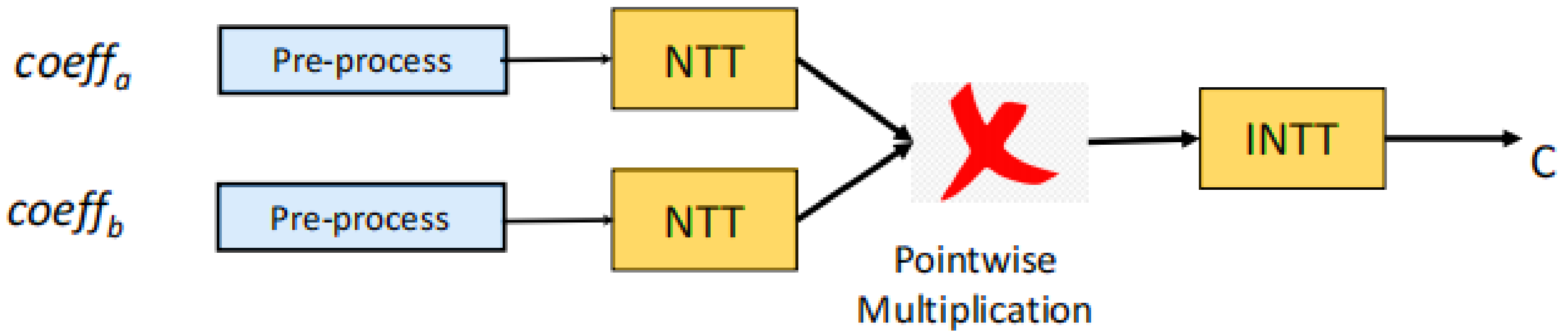
Poly_A : $(A_0, A_1, A_2, \ldots \ldots, A_{n-1})$ $n = 256$ **Poly_B :** $(B_0, B_1, B_2, \ldots \ldots, B_{n-1})$

Polynomial Multiplication Methods

Schoolbook Multiplication $O(n^2)$

Karatsuba Multiplication $O(n^{\log_2 3})$

Number Theoretic Transformation $O(n\log n)$



Convolution: The convolution product of two polynomial a,b are same as multiplying the two polynomials

Multiplication is defined as $NTT^{-1}(NTT(A).NTT(B))$

Continue…

Setup:

- Input: sequence of n positive integers
- Choose M such that 1<=n<M and every input value is in the range [0,M).
- Choose an integer k>=1 and define q=kn+1 such that q>=M. It can be shown that we can make sure that q is prime.
- As q is prime, the multiplicative group Z_q has q-1=kn number of elements. Then this group have at least one generator g.
- Define $w=g^k$ mod q. This w will be the primitive n^{th} root of unity

Positive Wrapped Convolution

$$C(x) = A(x) \times B(x) \bmod (x^n - 1) = \sum_{i=0}^{n-1} c_i x^i$$

$$\text{where } c_i = (\sum_{j=0}^{i} a_j.b_{i-j} + \sum_{j=i+1}^{n} a_j.b_{n+i-j}) \bmod q$$

Negative Wrapped Convolution with NTT

$$\tilde{A}_j = \sum_{i=0}^{n-1} \gamma^i \cdot {\omega_n}^{ij} \cdot A_i, A_j = \frac{1}{n} \cdot \gamma^{-j} \cdot \sum_{i=0}^{n-1} {\omega_n}^{-ij} \cdot \tilde{A}_j$$

Positive Wrapped Convolution with NTT

$$NTT_j^A = \sum_{i=0}^{n-1} {\omega_n}^{ij} \cdot A_i, \; {NTT^{-1}}_j^A = \frac{1}{n} \sum_{i=0}^{n-1} {\omega_n}^{ij} \cdot A_i$$

Example of NTT/NTT^{-1} in Positive Wrapped Convolution

$$Let\ G(x) = 1 + 2x + 3x^2 + 4x^3, where\ w = 3383\ and\ q = 7681, \qquad find\ NTT\ (G(x))$$

$$\begin{bmatrix} \omega^{0\times0} & \omega^{0\times1} & \omega^{0\times2} & \omega^{0\times3} \\ \omega^{1\times0} & \omega^{1\times1} & \omega^{1\times2} & \omega^{1\times3} \\ \omega^{2\times0} & \omega^{2\times1} & \omega^{2\times2} & \omega^{2\times3} \\ \omega^{3\times0} & \omega^{3\times1} & \omega^{3\times2} & \omega^{3\times3} \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \Rightarrow \begin{bmatrix} \omega^0 & \omega^0 & \omega^0 & \omega^0 \\ \omega^0 & \omega^1 & \omega^2 & \omega^3 \\ \omega^0 & \omega^2 & \omega^4 & \omega^6 \\ \omega^0 & \omega^3 & \omega^6 & \omega^9 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \Rightarrow \begin{bmatrix} \omega^0 & \omega^0 & \omega^0 & \omega^0 \\ \omega^0 & \omega^1 & \omega^2 & \omega^3 \\ \omega^0 & \omega^2 & \omega^0 & \omega^2 \\ \omega^0 & \omega^3 & \omega^2 & \omega^1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

$$\begin{bmatrix} 3383^0 & 3383^0 & 3383^0 & 3383^0 \\ 3383^0 & 3383^1 & 3383^2 & 3383^3 \\ 3383^0 & 3383^2 & 3383^0 & 3383^2 \\ 3383^0 & 3383^3 & 3383^2 & 3383^1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 3383 & 7680 & 4298 \\ 1 & 7680 & 1 & 7680 \\ 1 & 4298 & 7680 & 3383 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \Rightarrow \begin{bmatrix} 10 \\ 913 \\ 7679 \\ 6764 \end{bmatrix}$$

$$Find\quad NTT^{-1}(NTT\big(G(x)\big))$$

$$n^{-1} \begin{bmatrix} \omega^0 & \omega^0 & \omega^0 & \omega^0 \\ \omega^0 & \omega^{-1} & \omega^{-2} & \omega^{-3} \\ \omega^0 & \omega^{-2} & \omega^{-0} & \omega^{-2} \\ \omega^0 & \omega^{-3} & \omega^{-2} & \omega^{-1} \end{bmatrix} \begin{bmatrix} 10 \\ 913 \\ 7679 \\ 6764 \end{bmatrix} \Rightarrow 5761 \begin{bmatrix} 4298^0 & 4298^0 & 4298^0 & 4298^0 \\ 4298^0 & 4298^1 & 4298^2 & 4298^3 \\ 4298^0 & 4298^2 & 4298^0 & 4298^2 \\ 4298^0 & 4298^3 & 4298^2 & 4298^1 \end{bmatrix} \begin{bmatrix} 10 \\ 913 \\ 7679 \\ 6764 \end{bmatrix} \Rightarrow 5761 \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 4298 & 7680 & 3383 \\ 1 & 7680 & 1 & 7680 \\ 1 & 3383 & 7680 & 4298 \end{bmatrix} \begin{bmatrix} 10 \\ 913 \\ 7679 \\ 6764 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

Satriawan, A., Syafalni, I., Mareta, R., Anshori, I., Shalannanda, W., & Barra, A. (2023). Conceptual review on number theoretic transform and comprehensive review on its implementations. *IEEE Access*.

Assignment

Example of negative wrapped convolution:

Write a python code to covert a polynomial into NTT domain where q=8380417 and n=256 using the formula.

Why NTT is Faster

Algorithm 1 Decimation in Time NTT Algorithm

Input: $P_N = \{a_1, a_2, a_3, ..., a_n\}$ $P_N \in Z_q$
Output: NTT(P)
P=bitreverse(P_N)
for $m \leftarrow 2$ to n **do**
  $\omega = 1$, $\omega_m = \omega_n^{n/m}$
  for $j \leftarrow 0$ to $\frac{m}{2}$ **do**
    for $k \leftarrow 0$ to n **do**
      $u_1 = P[k+j]$, $t_1 = P[k+j+\frac{m}{2}] \cdot \omega$
      $P[k+j] = (u_1 + t_1)\ mod\ q$
      $P[k+j+\frac{m}{2}] = (u_1 - t_1)\ mod\ q$
      $k = k + m$
    end for
    $\omega = (\omega_m \cdot \omega)\ mod\ q$
    $j = j + 1$
  end for
  $m = 2 * m$
end for

- The pointwise multiplication has a complexity of O(n) only
- The multiplication and addition operations involved in these two algorithms are done in the field Z_q

Algorithm 2 Decimation in Frequency NTT^{-1} Algorithm

Input: $\hat{P} = \{b_1, b_2, b_3, ..., b_n\}$ $P \in Z_q$
Output: $\text{NTT}^{-1}(\hat{P})$
for $m \leftarrow n$ to 1 **do**
  $j_1 = 0$, $h = \frac{m}{2}$
  for $i \leftarrow 0$ to h **do**
    $j_2 = j_1 + t - 1$
    $index = h + i$
    $Omega = (\omega^{-1})^{(bitreverse(index))}$
    for $j \leftarrow j_1$ to j_2 **do**
      $u_1 = \hat{P}[j]$, $t_1 = \hat{P}[j+t] \cdot \omega$
      $\hat{P}[j] = (u_1 + t_1)\ mod\ q$
      $\hat{P}[j+t] = ((u_1 - t_1) \cdot Omega)\ mod\ q$
      $j = j + 1$
    end for
    $j_1 = j_1 + 2 * t$
    $i = i + 1$
  end for
  $m = \frac{m}{2}$
end for
for $i \leftarrow 0$ to n **do**
  $\hat{P}[i] = \hat{P}[i] \cdot \frac{1}{n}$
  $i = i + 1$
end for

- Both Decimation in Time and Decimation of Frequency algorithm has a complexity of O(nlogn)
- They apply divide and conquer approach to achieve this acceleration

Assignment

Write a python code to covert a polynomial into NTT domain where q=3329 and n=256 using the DIT NTT algorithm.

Components in NTT/ NTT^{-1}/ PWM

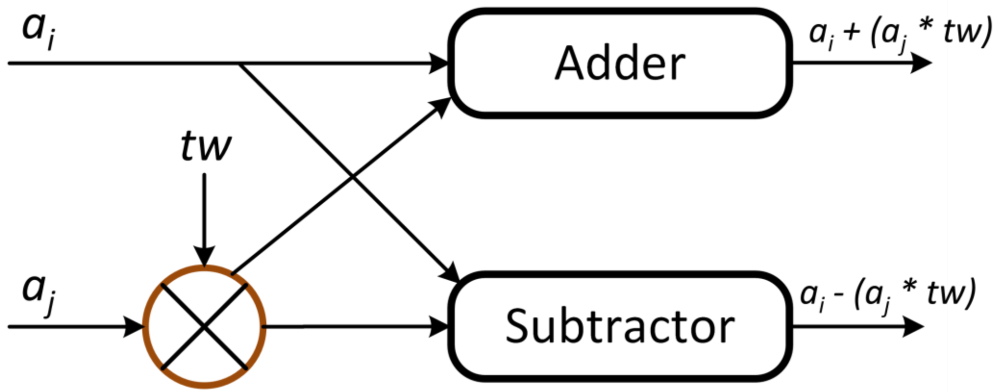
Modular Adder

Modular Subtractor

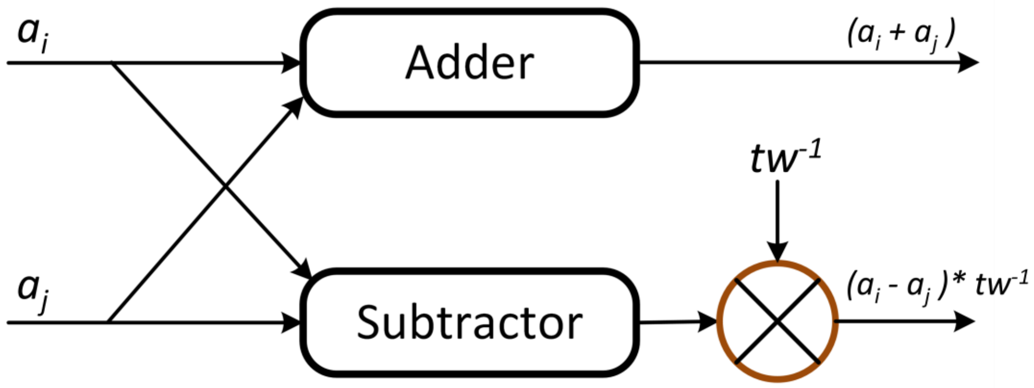
Modular Multiplier

Memory Unit

Barrett/ Montgomery/ K-Red



Cooley-Tukey NTT Butterfly Unit



Gentleman-Sande NTT^{-1} Butterfly Unit

- Radix-2 Cooley-Tukey / Gentleman-Sande Architecture.
- Coefficients are fetched and processed through the BFUs iteration wise.

Assignment

Design a modular adder/subtractor for Kyber in FPGA

Modular Multipliers

a= 2238, b=1276, (a*b)%3329=2735

Montgomery

Algorithm Mont_Reduc(R,P,T): Montgomery Reduction

Input: R, P, T. P is the prime, $R = 2^k > P$, $0 \le T < PR$, $gcd(P, R) = 1$

Output: TR^{-1} mod P

1 Compute P' such that $RR^{-1} - PP' = 1$ (can be computed using extended Euclidean algorithm);
2 $m = T \times P'$ mod R ;
3 $t = (T + mP)/R$;
4 **if** $t \ge P$ **then**
5 $\quad t = t - P$;
6 **end**
7 return t;

Montgomery (a, b) = 475

Where R = 2^{12} , and R^{-1}= 2704

(2704 * 2735) % 3329 = 475

Barrett

Algorithm Barrett modular multiplication.

Input: A modulus $M \in \mathbb{N}_{\ge 2}$ of length $k = \lfloor \log_b M \rfloor + 1$ in base $b \in \mathbb{N}_{\ge 2}$, the integer reciprocal $\mu = \lfloor b^{2k}/M \rfloor$ of M, and integers $x, y \in \mathbb{Z}_M$.

Output: $(x \cdot y)$ mod M.

1: $z \leftarrow x \cdot y$
2: $\tilde{q} \leftarrow \lfloor \lfloor z/b^{k-1} \rfloor \mu / b^{k+1} \rfloor$
3: $r \leftarrow z - \tilde{q} \cdot M$
4: **while** $r \ge M$ **do**
5: $\quad r \leftarrow r - M$ ▷ Extra reduction
6: **end while**
7: **return** r

Barrett (a, b) = 2735

Where k=12 and μ=5039

K-Red

Algorithm K-RED Algorithm

1: **Input:** $C = a \times b$
2: **Output:** $K \times C \ mod \ q$
3: $d_0 = C[m:0], d_1 = C \gg m$
4: return $(K \times d_0 - d_1)$

K= 2^8*13 + 1

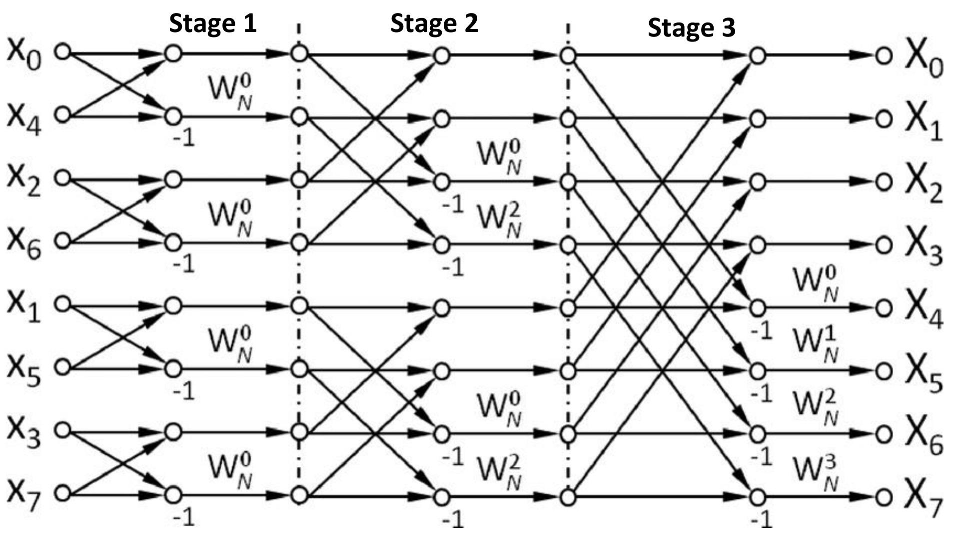
K-Red (a, b) = 2265, where K=13

(13*2735) % 3329= 2265

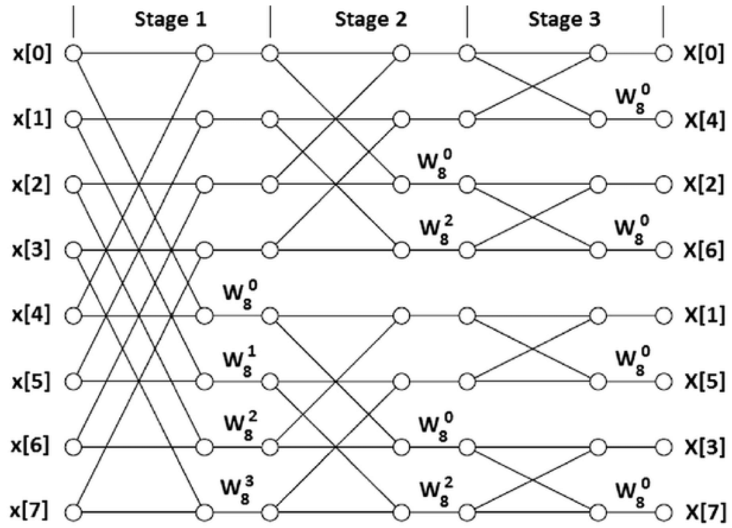
Assignment

Design a modular multiplier for Kyber in FPGA using Montgomery Multiplication

Challenges in Designing Efficient NTT Architecture



Radix 2 DIT Signal Flow for 8 point NTT, Note the bit reversal of the input



Radix 2 DIF Signal Flow for 8 point NTT, note the bit reversal of the output.

- Pipelined Memory Access
- Avoiding rearrangement of coefficients between NTT/ NTT-1
- Low Overhead Modular Multiplier
- Supporting NTT/NTT^{-1}/ PWM
- Multiplication with N^{-1} during NTT^{-1}

Pöppelmann, Thomas, Tobias Oder, and Tim Güneysu. "High-performance ideal lattice-based cryptography on 8-bit ATxmega microcontrollers." *International conference on cryptology and information security in Latin America*. Cham: Springer International Publishing, 2015.

NTT Multiplication in CRYSTALS-Kyber

- Polynomial multiplication in Kyber is equivalent to negative wrapped convolution with modulus q = 3329 and n = 256.
- But 512th primitive root of unity does not exist for Kyber.
- Rather, Kyber has 256th root of unity. So, Kyber performs incomplete NTT i.e. NTT of odd and even coefficients are computed independently

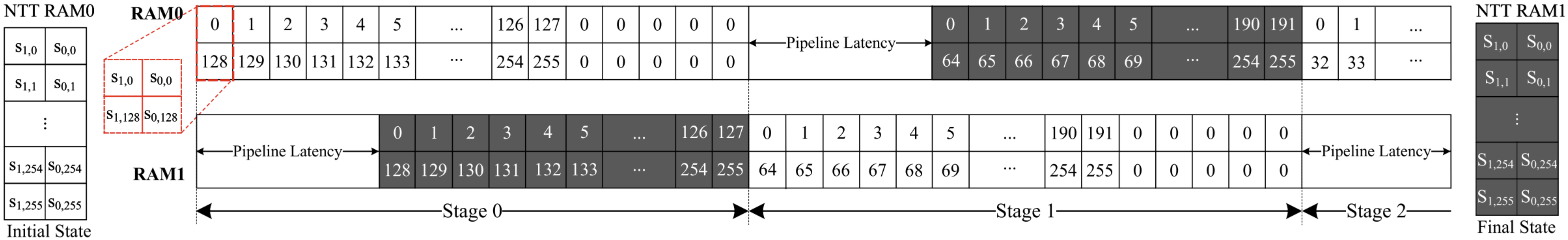
Pointwise Multiplication in NTT domain for CRYSTALS-Kyber

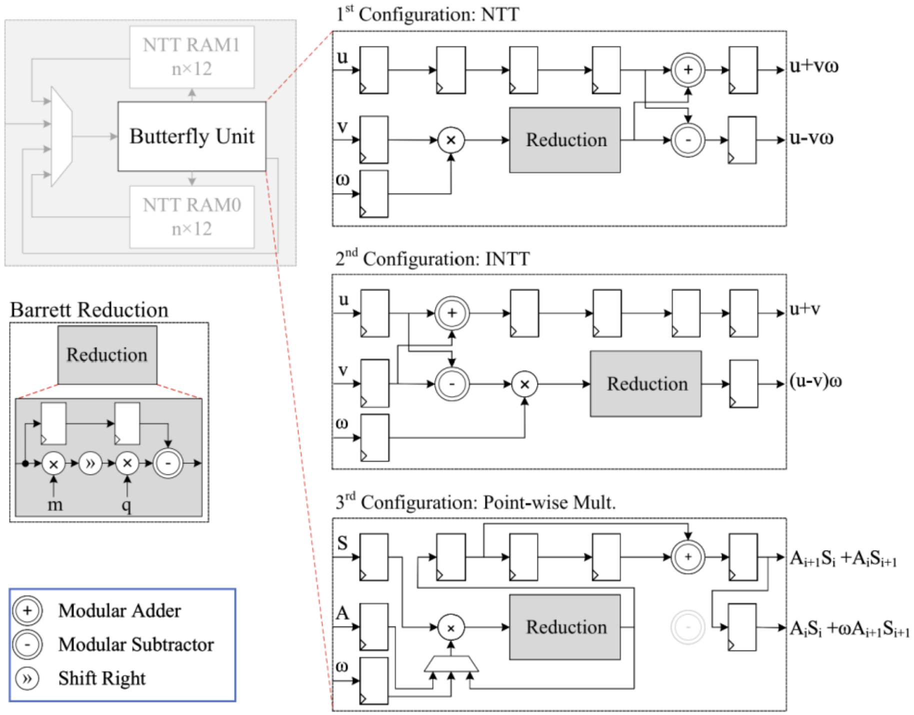
$$\hat{h}_{2i} = \hat{f}_{2i}\hat{g}_{2i} + \hat{f}_{2i+1}\hat{g}_{2i+1} \cdot \zeta^{2br(i)+1} \qquad \hat{h}_{2i+1} = \hat{f}_{2i}\hat{g}_{2i+1} + \hat{f}_{2i+1}\hat{g}_{2i}$$

$$\begin{aligned}
&\mathrm{PWM0}: s_0 = \hat{f}_{2i} + \hat{f}_{2i+1},\ s_1 = \hat{g}_{2i} + \hat{g}_{2i+1},\ m_0 = \hat{f}_{2i} \cdot \hat{g}_{2i},\ m_1 = \hat{f}_{2i+1} \cdot \hat{g}_{2i+1}\\
&\mathrm{PWM1}: s_2 = m_0 + m_1,\ m_2 = s_0 \cdot s_1,\ m_3 = m_1 \cdot \zeta^{2br(i)+1},\ \hat{h}_{2i} = m_0 + m_3,\ \hat{h}_{2i+1} = m_2 - s_2
\end{aligned}$$

Xing, Yufei, and Shuguo Li. "A compact hardware implementation of CCA-secure key exchange mechanism CRYSTALS-KYBER on FPGA." *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2021): 328-356.

An example of NTT Multiplication Architecture of Kyber





- ❑ RAM0 and RAM1 is accessed alternatively.
- ❑ Have support for NTT/ NTT^{-1}/PWM
- ❑ An overhead of 737 LUTs and 6 DSPs.
- ❑ Have to wait until write operations of each stage finishes.
- ❑ Consumes 474/602/1289 CCs for NTT/NTT^{-1}/PWM.

Bisheh-Niasar, Mojtaba, Reza Azarderakhsh, and Mehran Mozaffari-Kermani. "Instruction-set accelerated implementation of CRYSTALS-Kyber." *IEEE Transactions on Circuits and Systems I: Regular Papers* 68.11 (2021): 4648-4659.

Unified NTT Multiplication Unit for CRYSTALS-Kyber and CRYSTALS-Dilithium (VLSI-Design 2024)

Suraj Mandal, Debapriya Basu Roy

Department of Computer Science & Engineering

Indian Institute of Technology Kanpur

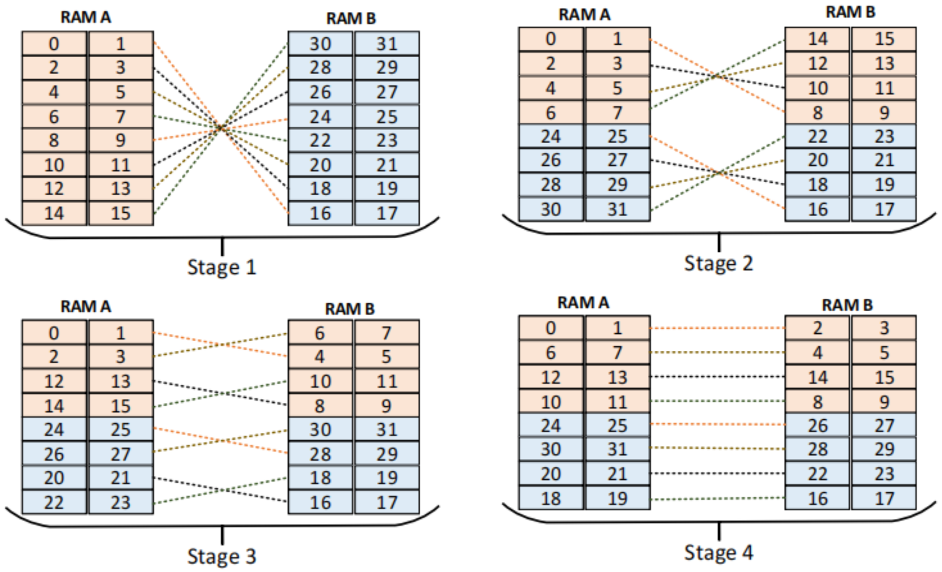
dbroy@cse.iitk.ac.in

Features of the Work

- ✓ Pipelined Memory access (without stalls).
- ✓ Efficient modular reduction unit using the prime structure.
- ✓ Removing the multiplication with N^{-1} during NTT^{-1}.
- ✓ Configure the adder/subtractor for multiplying 2^{-1}.
- ✓ Reconfigurable butterfly unit (able to perform NTT/NTT^{-1}/PWM).
- ✓ Unified NTT Multiplication for Kyber and Dilithium. DSP blocks/ Modular adder/Modular subtractors are shared modules.

Pipelined Memory Access

Pipelined Memory Access without Stalling



Algorithm 1 Pipelined Memory Addressing for NTT/NTT^{-1}

procedure ADDRESSING(ch, d)
  if ch==1 **then**
    $start = 1, end = \log_2 d$
  else
    $start = \log_2 d, end = 1$
  end if
  for i=start to end **do**
    $m = 2^i, p_2 = \frac{m}{2}$
    for j=0 to N by p_2 **do**
      $k_1 = j, k_2 = p_2 - 1 + j$
      for k=0 to p_2 **do**
        if k is even **then**
          $address_A = k_1, address_B = k_2$
        else if k is odd **then**
          $address_A = k_2, address_B = k_1$
          $k_1 = k_1 + 1, k_2 = k_2 - 1$
        end if
      end for
    end for
  end for
  return $(address_A, address_B)$
end procedure

Conditions for No Stalling

$$Pipeline\ Depth \leq \frac{Depth\ of\ Coefficient\ Memory}{2}.$$

Design of Efficient Montgomery Multiplier

Algorithm 1. Montgomery Reduction

Require: An m-bit modulus M, Montgomery radix $R = 2^m$, two m-bit operands A and B, and m-bit pre-computed constant $M' = -M^{-1} \bmod R$

Ensure: Montgomery product $(Z = (A \cdot B) \cdot R^{-1} \bmod M)$

1: $T \leftarrow A \cdot B$

2: $Q \leftarrow T \cdot M' \bmod R$

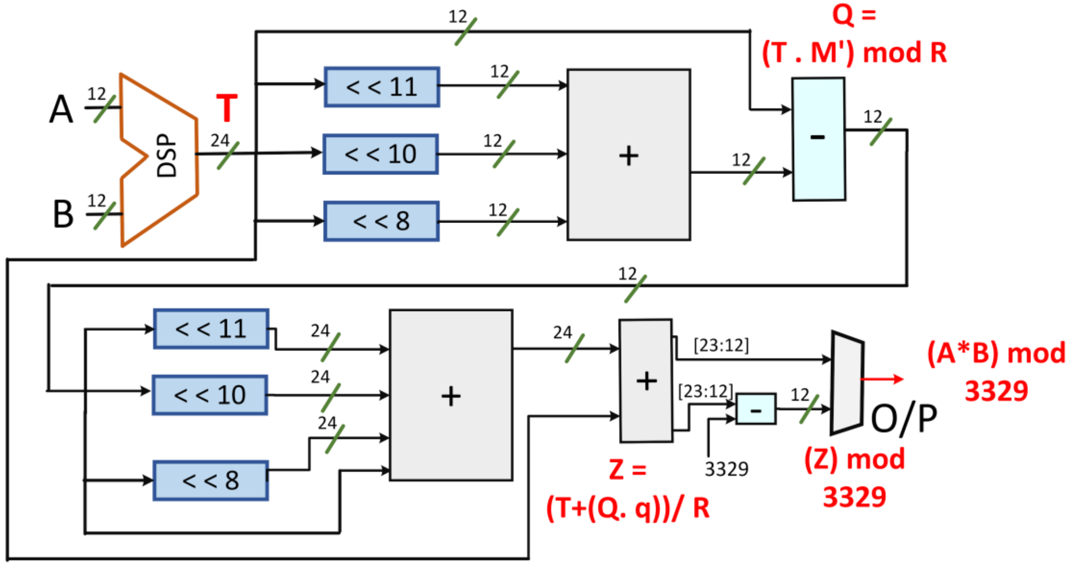
3: $Z \leftarrow (T + Q \cdot M)/R$

4: **if** $Z \geq M$ **then** $Z \leftarrow Z - M$ **end if**

5: **return** Z

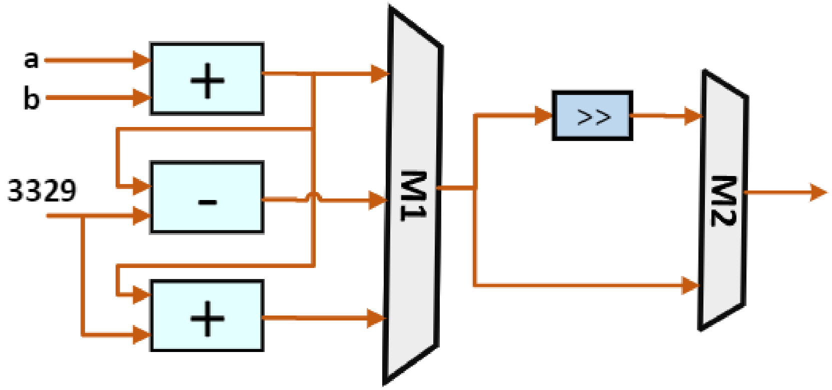
- ✓ Prime structure M= $2^{11} + 2^{10} + 2^8 + 1$
- ✓ Structure of M'= $2^{11} + 2^{10} + 2^8 - 1$
- ✓ Omits the use of DSP blocks for multiplications.

Montgomery Multiplier with one DSP

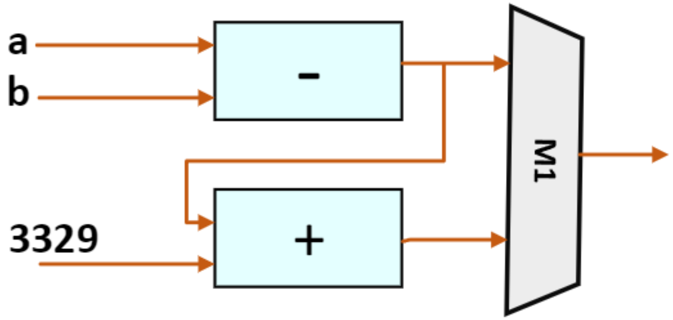


Modular Adder/Subtractor

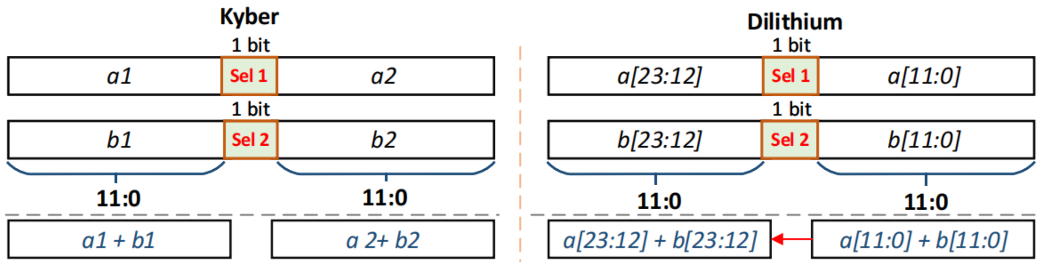
Modular Adder



Modular Subtractor



Shared Adder for Kyber and Dilithium



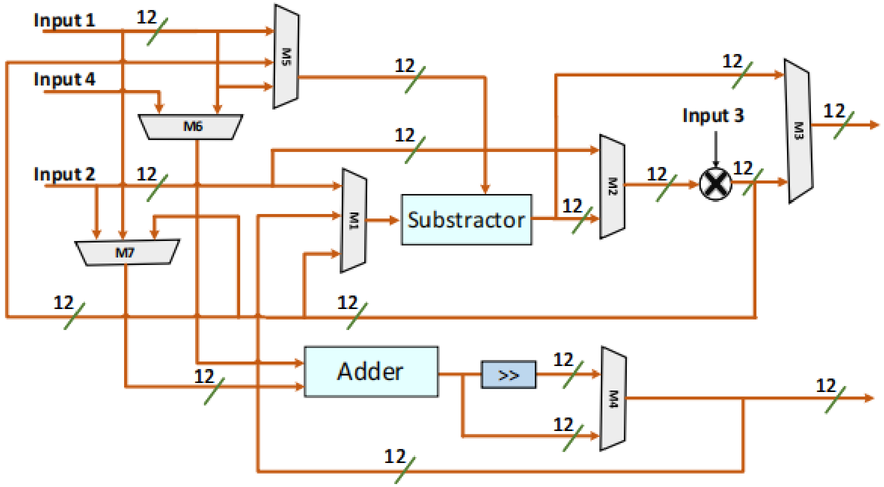
- ✓ M1 and M2 in modular adder is configured to perform $2^{-1}*(a+b)$ during NTT^{-1}.
- ✓ In modular subtractor, multiplication with 2^{-1} has been precomputed with twiddle factors.

Proposed Unified NTT Multiplication Unit

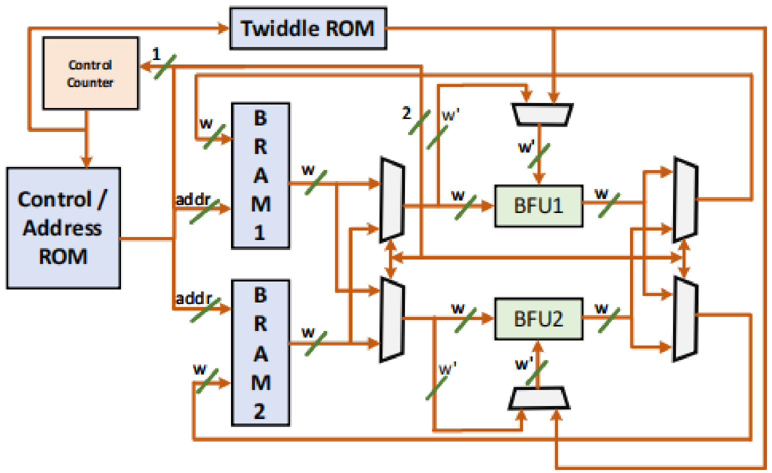
Our Proposal

Design 1: The first design is based on 2 radix-2 BFUs for Kyber that can also be used as 1 radix-2 BFU of Dilithium.
Design 2: The second design is based on 4 radix-2 BFUs of Kyber that can be used as 2 radix-2 BFUs of Dilithium.
Design 3: Our third design is based on 8 radix-2 BFUs of Kyber that can be used as 4 radix-2 BFUs of Dilithium.

BFU Unit



NTT Architecture



Experimental Results

Results for Standalone Implementation

NTT Type		Board	LUT	FFs	BRAM	DSP	Latency			Freq. (Mhz)	ADP-LUT			ADP-DSP	ADP-BRAM
							NTT	INTT	PWM		NTT	INTT	PWM		
Kyber	[3]	Artix-7	801	717	2	4	324	324	-	222	1169	1169	-	6	3
	[17]		360	145	2	3	940	1203	1289	115	2943	3766	4035	25	16
	[17]		737	290	4	6	474	602	1289	115	3038	3858	8261	25	16
	[2]		1579	1058	3	2	448	448	256	161	4394	4394	2511	6	8
	[5]		880	999	1.5	2	448	448	256	222	1776	1776	1015	4	3
	[4]		609	640	4	2	490	490	-	256	1166	1166	-	8	15
	TW		**799**	**916**	**2**	**2**	**448**	**448**	**256**	**310**	**1155**	**1155**	**660**	**3**	**3**
	TW	ZCU+ 102	**698**	**865**			**448**	**448**	**256**	**500**	**625**	**625**	**357**	**2**	**2**
Dilithium	[8]	Artix-7	9018	6292	2	16	256	256	-	250	9234	9234	-	16	2
	[10]	Zynq 7000	2386	932	2	8	256	256	64	217	2815	2815	704	9	2
	[9]	Artix-7	524	759	1	17	533	536	-	311	898	903	-	29	2
	[11]		2759	2037	7	4	512*	512*	128*	163	8666	8666	2167	13	22
	[11]	ZCU+	2759	2037	7	4	512*	512*	128*	391	3613	3613	903	5	9
	TW	Zynq 7000	**698**	**771**	**2.5**	**2**	**1024**	**1024**	**256**	**279**	**2562**	**2562**	**640**	**7**	**9**
		Artix-7	**690**	**771**			**1024**	**1024**	**256**	**273**	**2588**	**2588**	**647**	**8**	**9**
		ZCU+ 102	**724**	**769**			**1024**	**1024**	**256**	**413**	**1795**	**1795**	**449**	**5**	**6**

Results for Unified Implementation

NTT Type		Board	LUT	FFs	BRAM	DSP	K/D	Latency			Freq. (Mhz)	ADP-LUT			ADP-DSP	ADP-BRAM
								NTT	INTT	PWM		NTT	INTT	PWM		
Unified	[1]	ZCU+ 102	3487	1918	3*	4	K	224	224	128	270	2893	2893	1653	3	2
							D	512	512	128	270	6612	6612	1653	8	6
	TW		**1384**	**1220**	**4.5**	**2**	**K**	**448**	**448**	**256**	**387**	**1602**	**1602**	**916**	**2**	**5**
							D	**1024**	**1024**	**256**	**387**	**3662**	**3662**	**916**	**5**	**12**
			2893	**2356**	**4.5**	**4**	**K**	**224**	**224**	**128**	**342**	**1895**	**1895**	**1083**	**3**	**3**
							D	**512**	**512**	**128**	**342**	**4331**	**4331**	**1083**	**6**	**7**
			5909	**3376**	**5.5**	**8**	**K**	**112**	**112**	**64**	**294**	**2251**	**2251**	**1286**	**3**	**2**
							D	**256**	**256**	**64**	**294**	**5145**	**5145**	**1286**	**7**	**5**
		Artix-7	**1315**	**1280**	**4.5**	**2**	**K**	**448**	**448**	**256**	**263**	**2240**	**2240**	**1280**	**3**	**8**
							D	**1024**	**1024**	**256**	**263**	**5120**	**5120**	**1280**	**8**	**18**
			3105	**2389**	**4.5**	**4**	**K**	**224**	**224**	**128**	**200**	**3478**	**3478**	**1987**	**4**	**5**
							D	**512**	**512**	**128**	**200**	**7949**	**7949**	**1987**	**10**	**12**
			6201	**3562**	**5.5**	**8**	**K**	**112**	**112**	**64**	**165**	**4209**	**4209**	**2405**	**5**	**4**
							D	**256**	**256**	**64**	**165**	**9621**	**9621**	**2405**	**12**	**9**

State-of-the-art Works on designing Efficient NTT Multiplication Unit

	PQC Scheme	Features in NTT Multiplication unit	BFU Unit
[1]	Kyber	Unified NTT multiplication for Saber and Dilithium.	Unified NTT butterfly unit for Saber and Dilithium.
[2]		No rearrangement between NTT and NTT^{-1}.	Reconfigurable BFUs to execute NTT/NTT^{-1}/PWM.
[3]		Reduced area and memory consumption	Cooley-Tukey/Gentleman-Sande based reconfigurable architecture.
[4]		Ping-pong memory access.	Cooley-Tukey/Gentleman-Sande based reconfigurable architecture.
[5]		Pipelined NTT multiplication.	Performs NTT/NTT^{-1}/PWM mode.
[6]		Optimized storage requirement.	Reconfigurable butterfly unit.
[7]		Mixed radix-2/4 NTT multiplication unit.	Reconfigurable mixed radix-2/4 BFU able to perform NTT/NTT^{-1}
[8]		Mixed radix-2/4 NTT multiplication unit.	Reconfigurable mixed radix-2/4 BFU able to perform NTT/NTT^{-1}/PWM
[9]		Radix-2 NTT with improved memory access.	Performs NTT/NTT^{-1}/PWM
[10]		Radix-8/16 Mixed Radix NTT Multiplication.	Performs NTT/NTT^{-1}/PWM

References

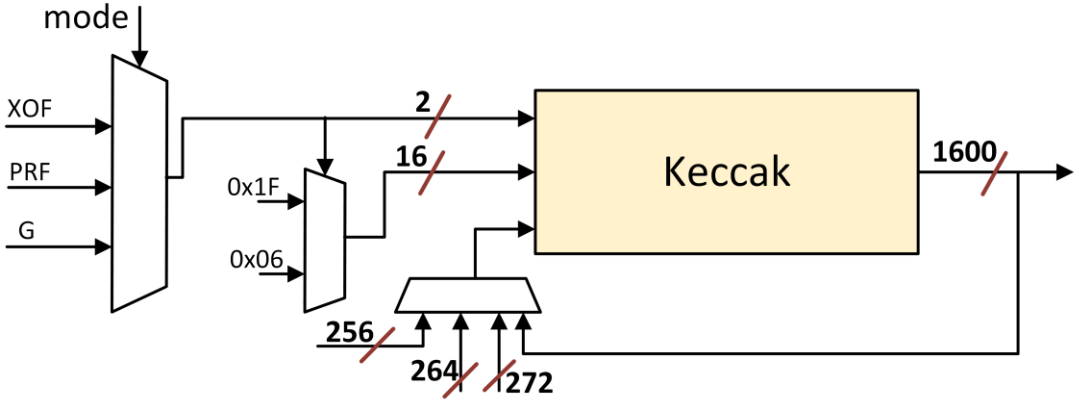
1. Aikata, Aikata, et al. "A unified cryptoprocessor for lattice-based signature and key-exchange." *IEEE Transactions on Computers* 72.6 (2022): 1568-1580.
2. Xing, Yufei, and Shuguo Li. "A compact hardware implementation of CCA-secure key exchange mechanism CRYSTALS-KYBER on FPGA." *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2021): 328-356.
3. Bisheh-Niasar, Mojtaba, Reza Azarderakhsh, and Mehran Mozaffari-Kermani. "High-speed NTT-based polynomial multiplication accelerator for post-quantum cryptography." *2021 IEEE 28th Symposium on Computer Arithmetic (ARITH)*. IEEE, 2021.
4. Zhang, Cong, et al. "Towards efficient hardware implementation of NTT for kyber on FPGAs." *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2021.
5. Dang, Viet Ba, Kamyar Mohajerani, and Kris Gaj. "High-speed hardware architectures and FPGA benchmarking of CRYSTALS-Kyber, NTRU, and Saber." *IEEE Transactions on Computers* 72.2 (2022): 306-320.
6. Bisheh-Niasar, Mojtaba, Reza Azarderakhsh, and Mehran Mozaffari-Kermani. "Instruction-set accelerated implementation of CRYSTALS-Kyber." *IEEE Transactions on Circuits and Systems I: Regular Papers* 68.11 (2021): 4648-4659.
7. Duong-Ngoc, Phap, and Hanho Lee. "Configurable mixed-radix number theoretic transform architecture for lattice-based cryptography." *IEEE Access* 10 (2022): 12732-12741.
8. Guo, Wenbo, and Shuguo Li. "Highly-efficient hardware architecture for crystals-kyber with a novel conflict-free memory access pattern." *IEEE Transactions on Circuits and Systems I: Regular Papers* (2023).
9. Mandal, Suraj, and Debapriya Basu Roy. "KiD: A Hardware Design Framework Targeting Unified NTT Multiplication for CRYSTALS-Kyber and CRYSTALS-Dilithium on FPGA." 2024 37th International Conference on VLSI Design and 2024 23rd International Conference on Embedded Systems (VLSID). IEEE, 2024.
10. Mandal, Suraj, and Debapriya Basu Roy. "Winograd for NTT: A Case Study on Higher-Radix and Low-Latency Implementation of NTT for Post Quantum Cryptography on FPGA." IEEE Transactions on Circuits and Systems I: Regular Papers (2024).

Assignment

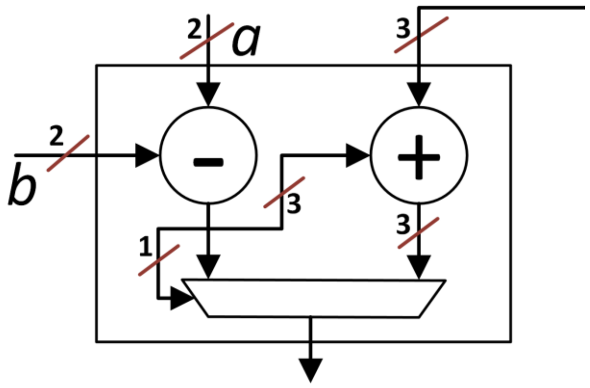
Write a python code to implement NTT multiplication unit for Kyber (NTT/ NTT^{-1}/PWM).

Other Modules of Kyber

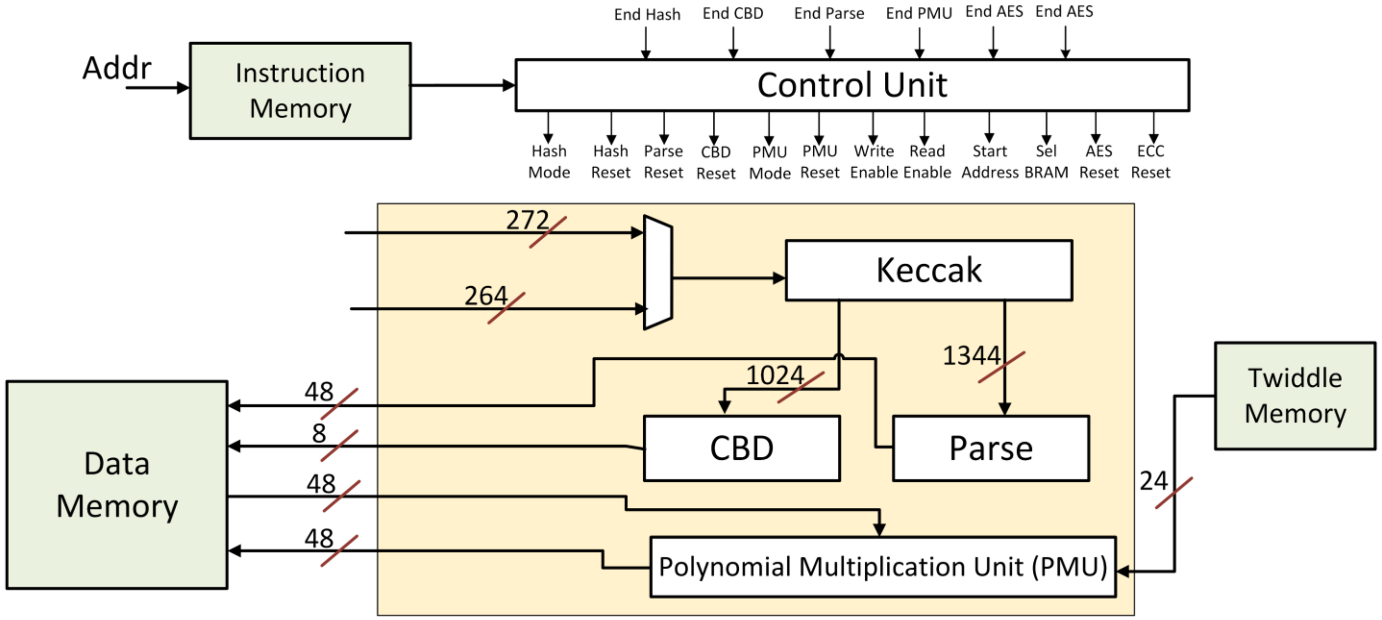
Keccak



CBD



Keygen Architecture



Execution of Kyber-Keygen

operation	cycles
$d \xleftarrow{\$} \mathcal{B}^{32}$	
$\{\rho, \sigma\} \leftarrow \mathrm{G}(d)$	79
$\mathbf{s}_0 \leftarrow \mathrm{PRF}(\cdot)$	79
$\hat{\mathbf{s}}_0 \leftarrow \mathrm{NTT}(\mathrm{CBD}_{\eta_1}(\mathbf{s}_0)), A_{00} \leftarrow \mathrm{Parse}(\mathrm{XOF}(\cdot)), \mathbf{s}_1 \leftarrow \mathrm{PRF}(\cdot)$	512
$\hat{\mathbf{s}}_1 \leftarrow \mathrm{NTT}(\mathrm{CBD}_{\eta_1}(\mathbf{s}_1)), A_{01} \leftarrow \mathrm{Parse}(\mathrm{XOF}(\cdot)), \mathbf{s}_2 \leftarrow \mathrm{PRF}(\cdot)$	512
$\hat{\mathbf{s}}_2 \leftarrow \mathrm{NTT}(\mathrm{CBD}_{\eta_1}(\mathbf{s}_2)), \mathbf{e}_0 \leftarrow \mathrm{PRF}(\cdot)$	512
$\hat{\mathbf{e}}_0 \leftarrow \mathrm{NTT}(\mathrm{CBD}_{\eta_1}(\mathbf{e}_0))$	512
$acc \leftarrow A_{00}\hat{\mathbf{s}}_0 + \hat{\mathbf{e}}_0, A_{02} \leftarrow \mathrm{Parse}(\mathrm{XOF}(\cdot))$	256
$acc \leftarrow A_{01}\hat{\mathbf{s}}_1 + acc, A_{02} \leftarrow \mathrm{Parse}(\mathrm{XOF}(\cdot)), A_{10} \leftarrow \mathrm{Parse}(\mathrm{XOF}(\cdot))$	256
$\hat{\mathbf{t}}_0 \leftarrow A_{02}\hat{\mathbf{s}}_2 + acc, A_{10} \leftarrow \mathrm{Parse}(\mathrm{XOF}(\cdot)), \mathbf{e}_1 \leftarrow \mathrm{PRF}(\cdot)$	256
$\hat{\mathbf{e}}_1 \leftarrow \mathrm{NTT}(\mathrm{CBD}_{\eta_1}(\mathbf{e}_1)), A_{11} \leftarrow \mathrm{Parse}(\mathrm{XOF}(\cdot))$	512
$acc \leftarrow A_{10}\hat{\mathbf{s}}_0 + \hat{\mathbf{e}}_1, A_{12} \leftarrow \mathrm{Parse}(\mathrm{XOF}(\cdot))$	256
$acc \leftarrow A_{11}\hat{\mathbf{s}}_1 + acc, A_{12} \leftarrow \mathrm{Parse}(\mathrm{XOF}(\cdot)), A_{20} \leftarrow \mathrm{Parse}(\mathrm{XOF}(\cdot))$	256
$\hat{\mathbf{t}}_1 \leftarrow A_{12}\hat{\mathbf{s}}_2 + acc, A_{20} \leftarrow \mathrm{Parse}(\mathrm{XOF}(\cdot)), \mathbf{e}_2 \leftarrow \mathrm{PRF}(\cdot)$	256
$\hat{\mathbf{e}}_2 \leftarrow \mathrm{NTT}(\mathrm{CBD}_{\eta_1}(\mathbf{e}_2)), A_{21} \leftarrow \mathrm{Parse}(\mathrm{XOF}(\cdot))$	512
$acc \leftarrow A_{20}\hat{\mathbf{s}}_0 + \hat{\mathbf{e}}_2, A_{22} \leftarrow \mathrm{Parse}(\mathrm{XOF}(\cdot))$	256
$acc \leftarrow A_{21}\hat{\mathbf{s}}_1 + acc, A_{22} \leftarrow \mathrm{Parse}(\mathrm{XOF}(\cdot))$	256
$\hat{\mathbf{t}}_2 \leftarrow A_{22}\hat{\mathbf{s}}_2 + acc$	256
transmit $pk = \hat{\mathbf{t}}\|\rho$ to client side, h=H(pk)	696

Scheduling in for Kyber768 key-generation

Xing, Yufei, and Shuguo Li. "A compact hardware implementation of CCA-secure key exchange mechanism CRYSTALS-KYBER on FPGA." *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2021): 328-356.