POLITECNICO DI MILANO

An Efficient Unified Architecture for Polynomial Multiplications in Lattice-based Cryptoschemes

Francesco Antognazza, Alessandro Barenghi, Gerardo Pelosi, Ruggero Susella

ICISSP 2023, Lisbon

Introduction

Advancements of quantum computers require new asymmetric crypto schemes

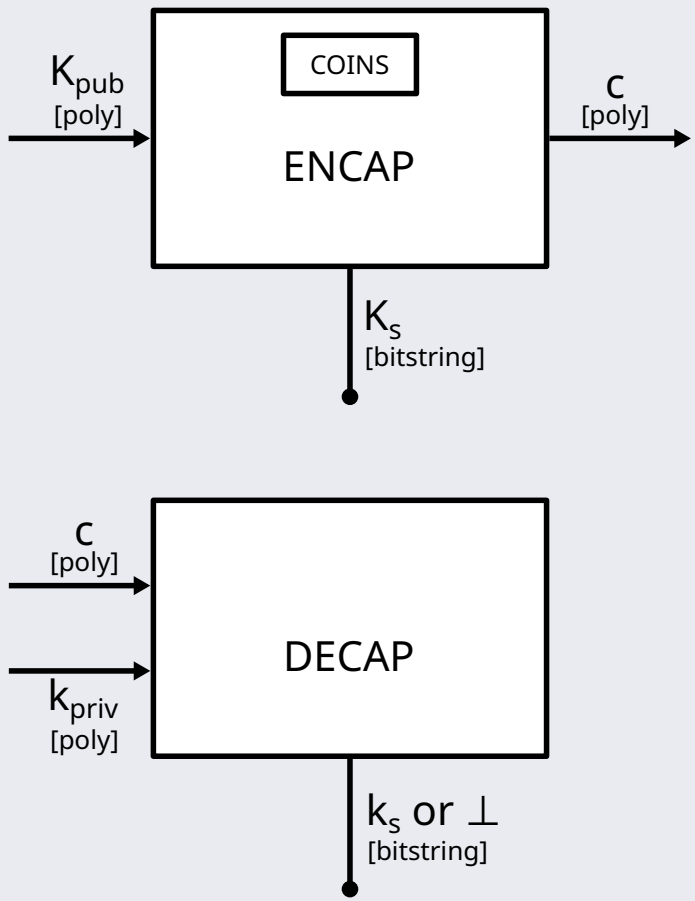
- replace both **signature** schemes and **Key Encap. Mechanisms**
- protection of today's data requires immediate deployment

In 2016 NIST asked for candidates:

- lattice-based proposals promising
  - **Kyber** (winner), NTRU HPS/HRSS, Saber, NTRU Prime
- final standard in 2024

Organizations already started to standardize and/or deploy them:

- **NTRU Prime** (OpenSSH), **NTRU HPS/HRSS** (IETF)



No clear standard in the immediate future

Motivation and goals

Challenges and goals

- produce a hardware accelerator for multiple lattice-based schemes
  - polynomial multiplication is the best candidate due to the large contribution in the latencies of keygen/encapsulation/decapsulation
- optimize the solutions based on the algebraic rings of the algorithms
- produce **scheme-specific** optimized solutions, and an **unified** design

Results improving state-of-the-art

- considerable area savings and latency reductions of our designs tailored for NTRU and NTRU Prime outperform the current state-of-the-art
- the unified design compatible with Kyber, NTRU and NTRU Prime requires a 15% frequency reduction and a negligible area increase w.r.t. NTRU Prime-tailored solution

Outline

1. Background
2. *x*-net polynomial multiplier architecture
3. Design Space Exploration
4. Conclusions

Background

Quotient polynomial rings

All lattice-based schemes use arithmetic of polynomials $a(x) = \sum_{i=0}^{n-1} a_i x^i$ in the quotient polynomial ring $\mathcal{R} = \mathbb{Z}/\langle \pi(x) \rangle$

Coefficients are either in $\mathbb{Z}_q$ or $\mathbb{Z}_p \implies$ poly rings $\mathcal{R}_q$ and $\mathcal{R}_p$

Quotient polynomial rings

All lattice-based schemes use arithmetic of polynomials $a(x) = \sum_{i=0}^{n-1} a_i x^i$ in the quotient polynomial ring $\mathcal{R} = \mathbb{Z}/\langle \pi(x) \rangle$

Coefficients are either in $\mathbb{Z}_q$ or $\mathbb{Z}_p \implies$ poly rings $\mathcal{R}_q$ and $\mathcal{R}_p$

scheme	**q**	**p**	**n**	$\pi(\mathbf{x})$
Kyber	prime	5, 7	256	$x^n + 1$
Saber	2^i	7, 9, 11	256	$x^n + 1$
NTRU	2^i	3	prime	$x^n - 1$
NTRU Prime	prime	3	prime	$x^n - x - 1$

p is a small odd number, hence poly in $\mathcal{R}_p$ have small coefficients; the ones in $\mathcal{R}_q$ are said to have large coefficients

Quotient polynomial rings

All lattice-based schemes use arithmetic of polynomials $a(x) = \sum_{i=0}^{n-1} a_i x^i$ in the quotient polynomial ring $\mathcal{R} = \mathbb{Z}/\langle \pi(x) \rangle$

Coefficients are either in $\mathbb{Z}_q$ or $\mathbb{Z}_p \implies$ poly rings $\mathcal{R}_q$ and $\mathcal{R}_p$

scheme	**q**	**p**	**n**	$\pi(\mathbf{x})$
Kyber	prime	5, 7	256	$x^n + 1$
Saber	2^i	7, 9, 11	256	$x^n + 1$
NTRU	2^i	3	prime	$x^n - 1$
NTRU Prime	prime	3	prime	$x^n - x - 1$

p is a small odd number, hence poly in $\mathcal{R}_p$ have small coefficients; the ones in $\mathcal{R}_q$ are said to have large coefficients

$\pi(x)$ gives a cyclic or nega-cyclic structure to the rings of Kyber, Saber and NTRU:

$$x \cdot a(x) = \pm a_{n-1} + \sum_{i=0}^{n-2} a_i x^{i+1}$$

$\mathcal{R}_q \times \mathcal{R}_q \rightarrow \mathcal{R}_q$ polynomial multiplications

The most expensive used operation is the polynomial multiplication

$$r(x) = a(x) \cdot b(x) \bmod \pi(x)$$

All considered cryptographic algorithms provide 3 security levels, comparable with AES128, AES192, and AES256.

Scale the underlying problem's dimension by varying:

- the maximum grade of polynomial ring (non-module schemes)
- the rank of matrix of polynomials (module-based schemes)

cryptoscheme	category	polynomial max. grade	module rank k	poly multiplications encap.	poly multiplications decap.
NTRU	quotient-ring	509, 677, 701, 821	/	1	$2^\star$
str. NTRU Prime	quotient-ring	653, 761, 857	/	1	3
NTRU LPRime	Ring-LWR	653, 761, 857	/	2	3
Saber	Module Ring-LWR	256	2,3,4	$k^2 + k$	$k^2 + 2k$
Kyber	Module Ring-LWE	256	2,3,4	$k^2 + k$	$k^2 + 2k$

$\mathcal{R}_q \times \mathcal{R}_q \rightarrow \mathcal{R}_q$ polynomial multiplications

Algorithms to perform a polynomial multiplication

algorithm	asymptotic complexity	parameters compatibility q	n
Schoolbook	n^2	any	any
Katatsuba	$n^{\log(3)/\log(2)}$	any	even
Toom-Cook i	$n^{\log(2i-1)/\log(i)}$	any	divisible by i
NTT	$n \log_2 n$	prime valori	power-of-two

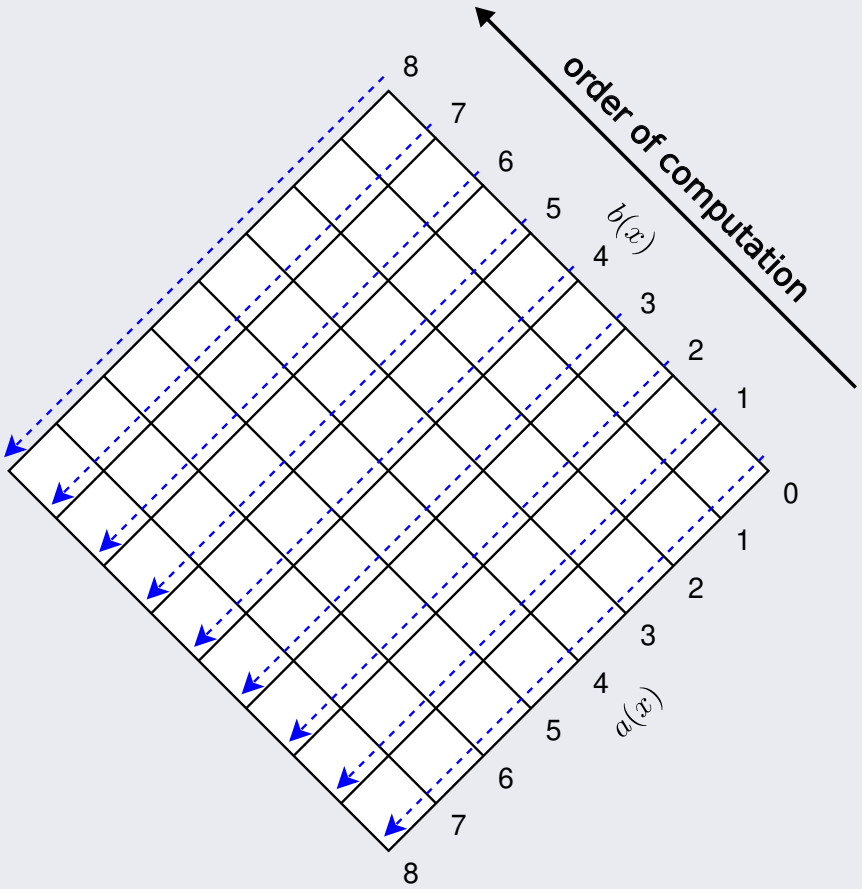
Balance the complexity of the algorithm's logic, with the parallelism of inner operations

The interaction with memories usually has strict constraints, limiting the theoretical parallelism

x-net polynomial multiplier architecture

Introduction to x-net architecture

The x-net architecture is a schoolbook algorithm (n^2 complexity): using n parallel MAC units, the computation is completed in n CC



$$r(x) = a(x) \cdot b(x)$$

$a(x)$ copied locally in Flip-Flops
$r(x)$ in FFs copied back to memory

One coeff. of $b(x)$ processed per CC by all n MAC units

Input: $a(x) \in \mathcal{R}_q,\ a(x) = \sum_{i=0}^{n-1} a_i x^i$
$b(x) \in \mathcal{R}_q,\ b(x) = \sum_{i=0}^{n-1} b_i x^i$
Output: $r(x) \in \mathbb{Z}[x],\ r(x) = a(x) \cdot b(x)$

forall r_i **in** $r(x)$ **do**
  $r_i \leftarrow 0$
for $j := n-1$ **to** 0 **do** // n clock cycles
  parallel for $i := 0$ **to** $n-1$ **do**
    $r_{(i+j)} \leftarrow r_{(i+j)} + a_i \cdot b_j$
return $r(x)$

Introduction to x-net architecture

But $r(x) \in \mathcal{R}_q \Rightarrow$ take the remainder of the division by $\pi(x)$

Approach not viable since it requires an expensive poly division

Introduction to x-net architecture

But $r(x) \in \mathcal{R}_q \Rightarrow$ take the remainder of the division by $\pi(x)$

Approach not viable since it requires an expensive poly division

Solution: perform cheaper $\mathrm{mod}\,\pi(x)$ every clock cycle

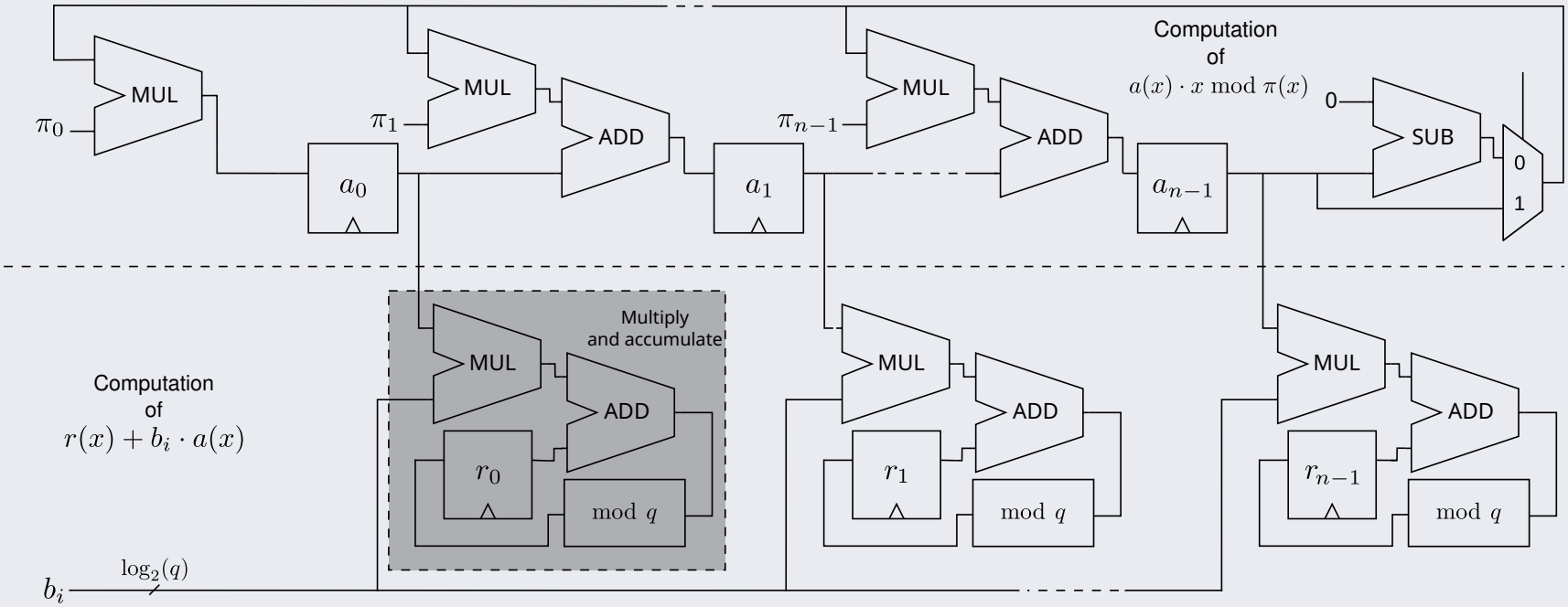
Two simultaneous tasks:

- accumulate a sequence of coefficient-by-polynomial mults.
- multiplication of a poly by x, followed by $\mathrm{mod}\,\pi(x)$ reduction

Input: $a(x) \in \mathcal{R}_q,\ a(x) = \sum_{i=0}^{n-1} a_i x^i$
$b(x) \in \mathcal{R}_q,\ b(x) = \sum_{i=0}^{n-1} b_i x^i$
Output: $r(x) \in \mathcal{R}_q,\ r(x) = a(x) \cdot b(x) \bmod \pi(x)$
Data: $\pi(x) \in \mathbb{Z}_q[x]$, monic, with degree n

$r(x) \leftarrow 0$
for $i = 0$ **to** $(n-1)$ **do** // n clock cycles
  $r(x) \leftarrow r(x) + b_i \cdot a(x)$ // n MACs
  $a(x) \leftarrow a(x) \cdot x \bmod \pi(x)$ // via LFSR
return $r(x)$

Introduction to x-net architecture

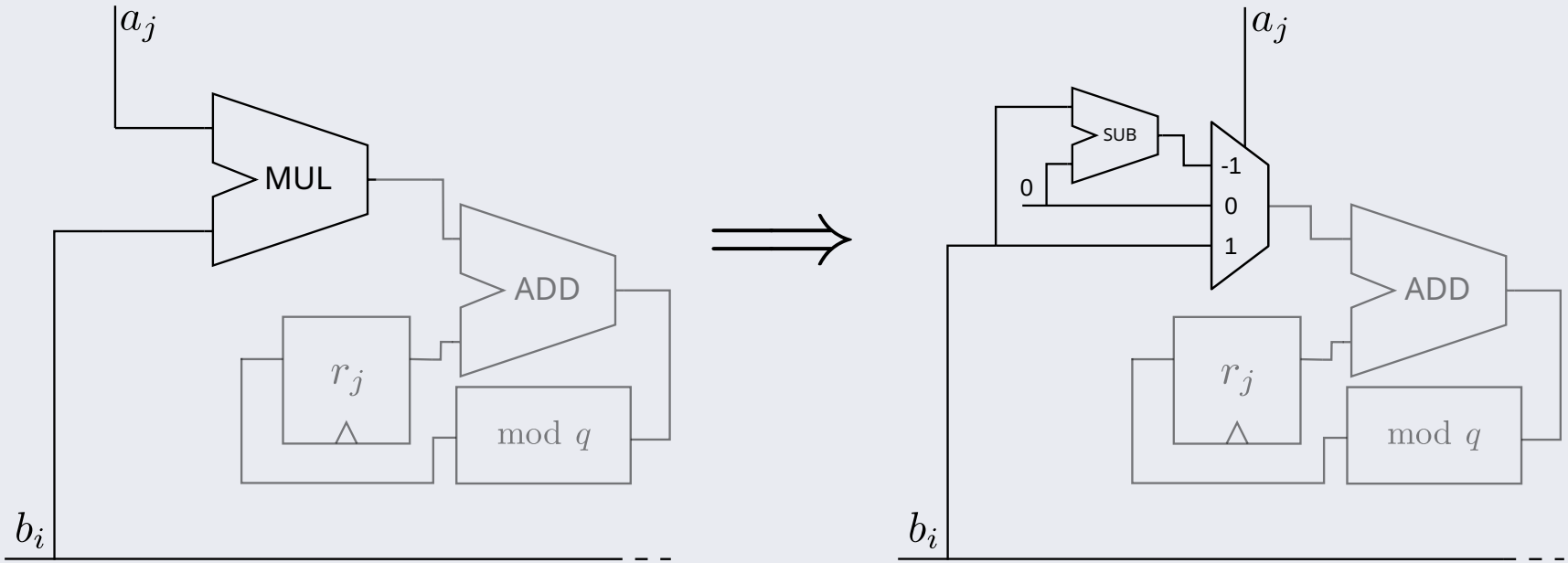


Advantages

- access to the memory is always sequential
- can use a single shared memory bus to access operands and result

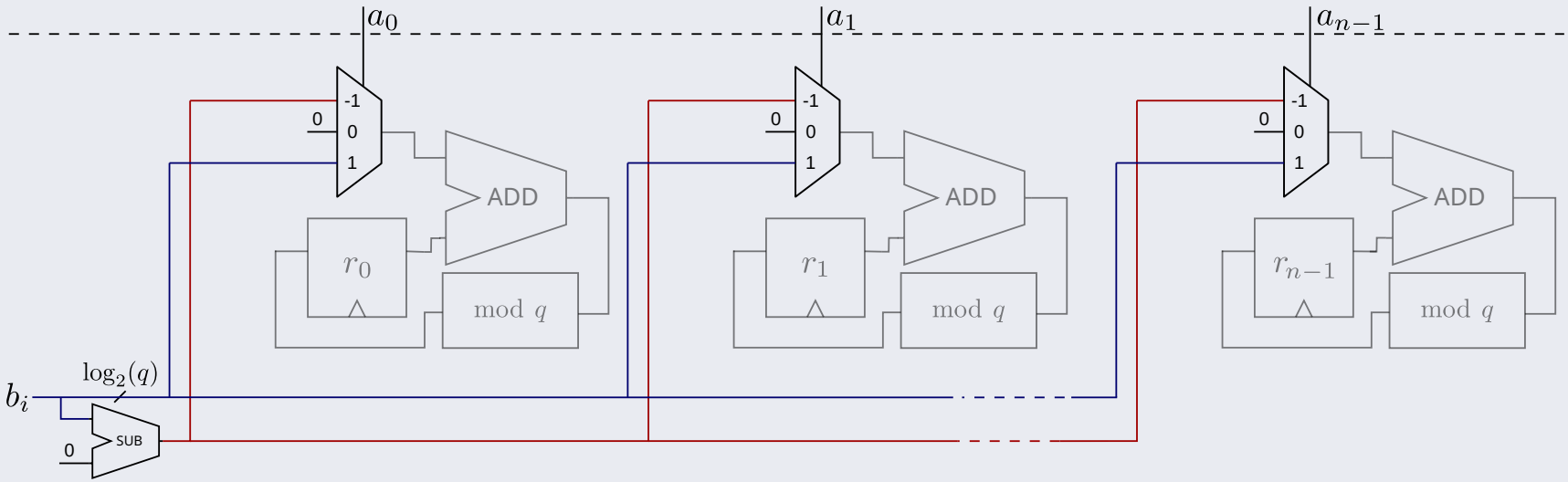
Optimizations applicable to x-net architecture

One operand has small coefficients ($\in \mathcal{R}_p$):
within each MAC unit, replace the scalar multiplier with a multiplexer



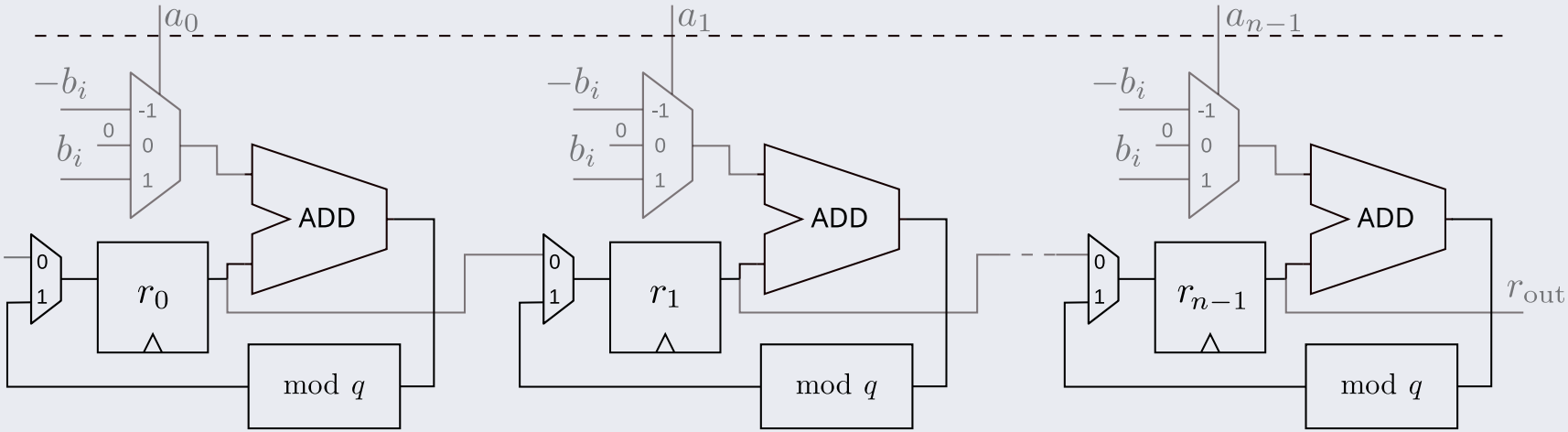
Optimizations applicable to *x*-net architecture

Precompute the outcomes once, and distribute to every MAC unit

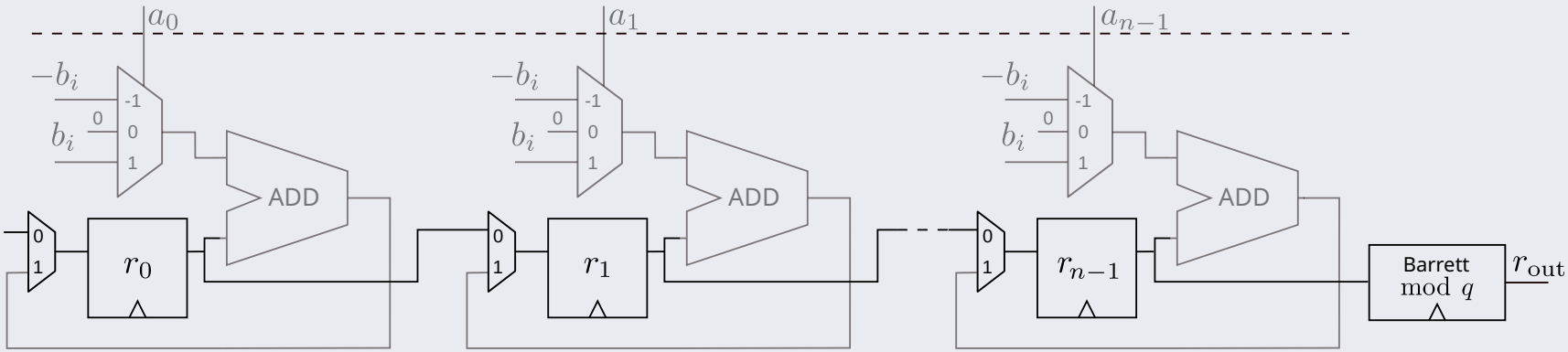


Optimizations applicable to *x*-net architecture

Reduce the accumulator result $\bmod q$ using comparison and additions



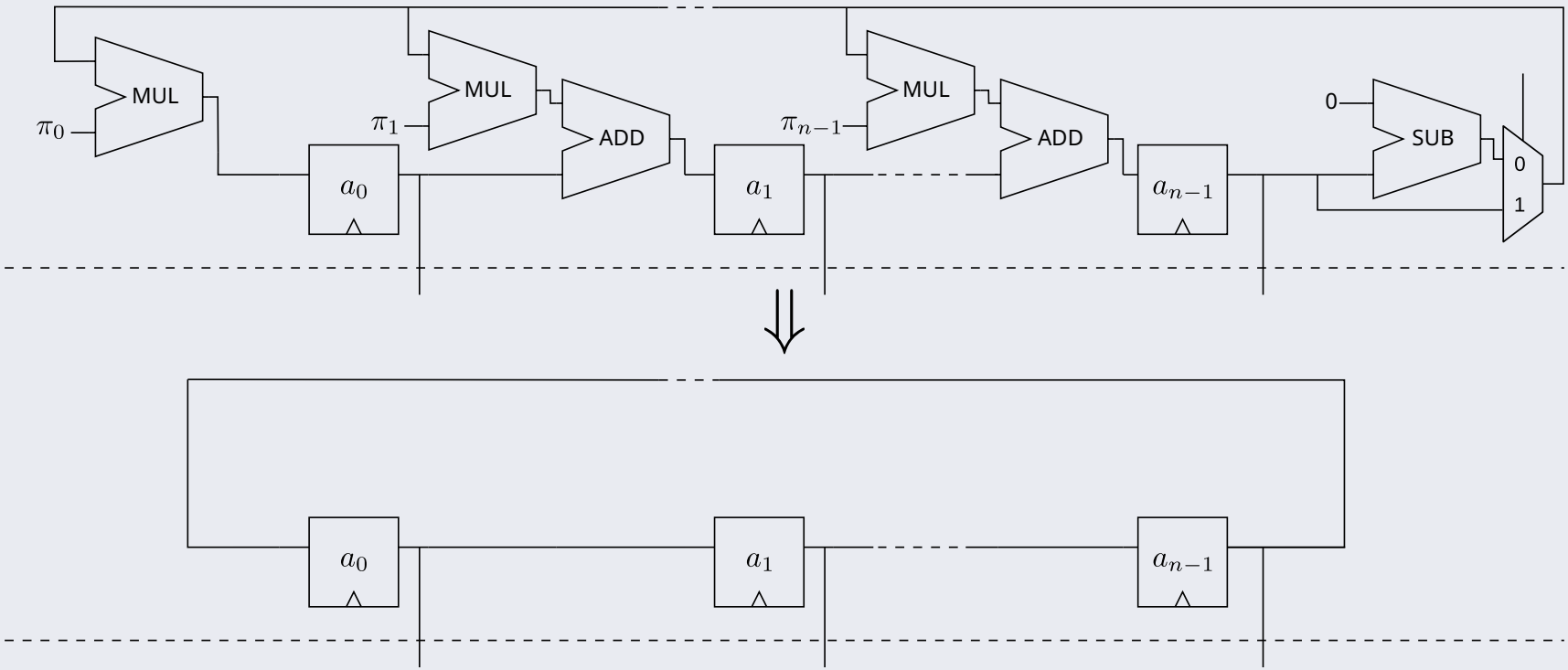
Perform it during the read-out phase using a general Barrett reduction



Optimizations applicable to x-net architecture

Taps of LFSR are in correspondence of non-zero coefficients of $\pi(x)$

Only few of them are active: remove unused adders in feedback net

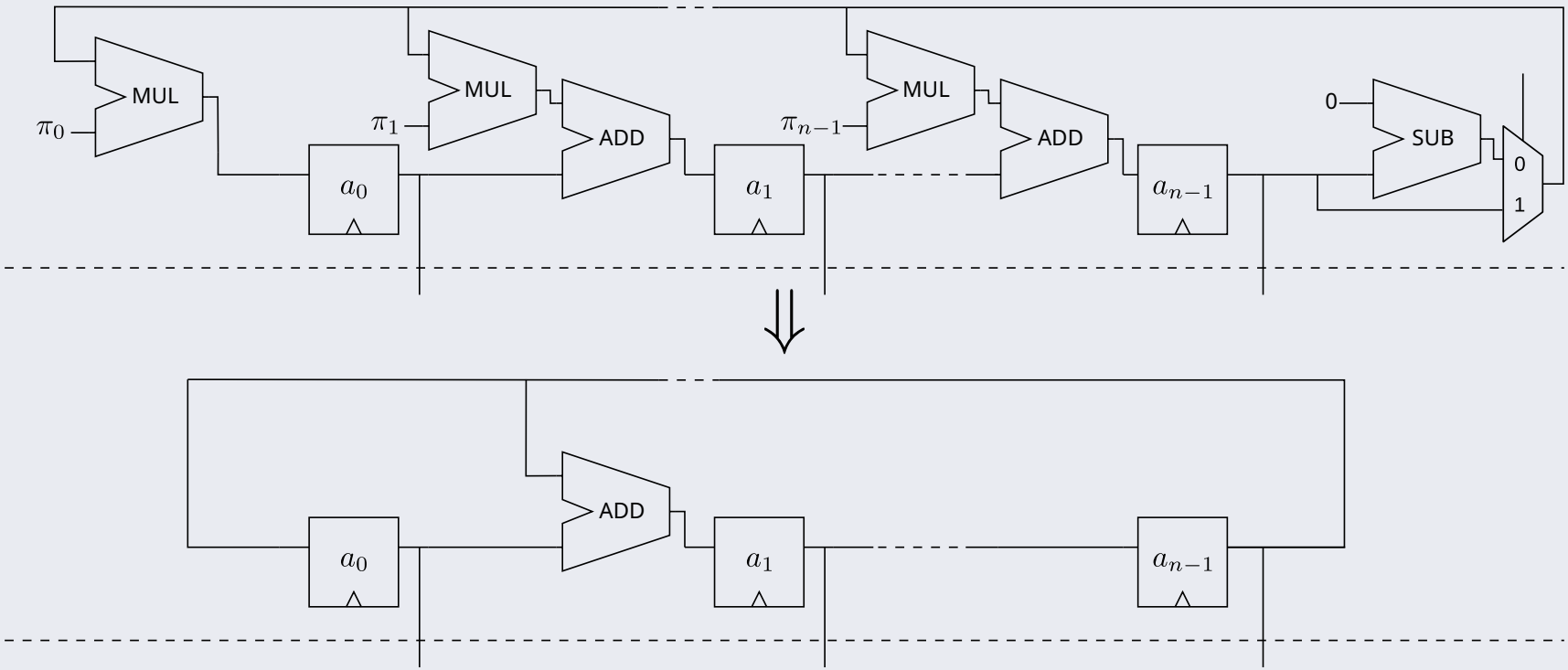


Feedback network for NTRU HPS/HRSS

Optimizations applicable to x-net architecture

Taps of LFSR are in correspondence of non-zero coefficients of $\pi(x)$

Only few of them are active: remove unused adders in feedback net

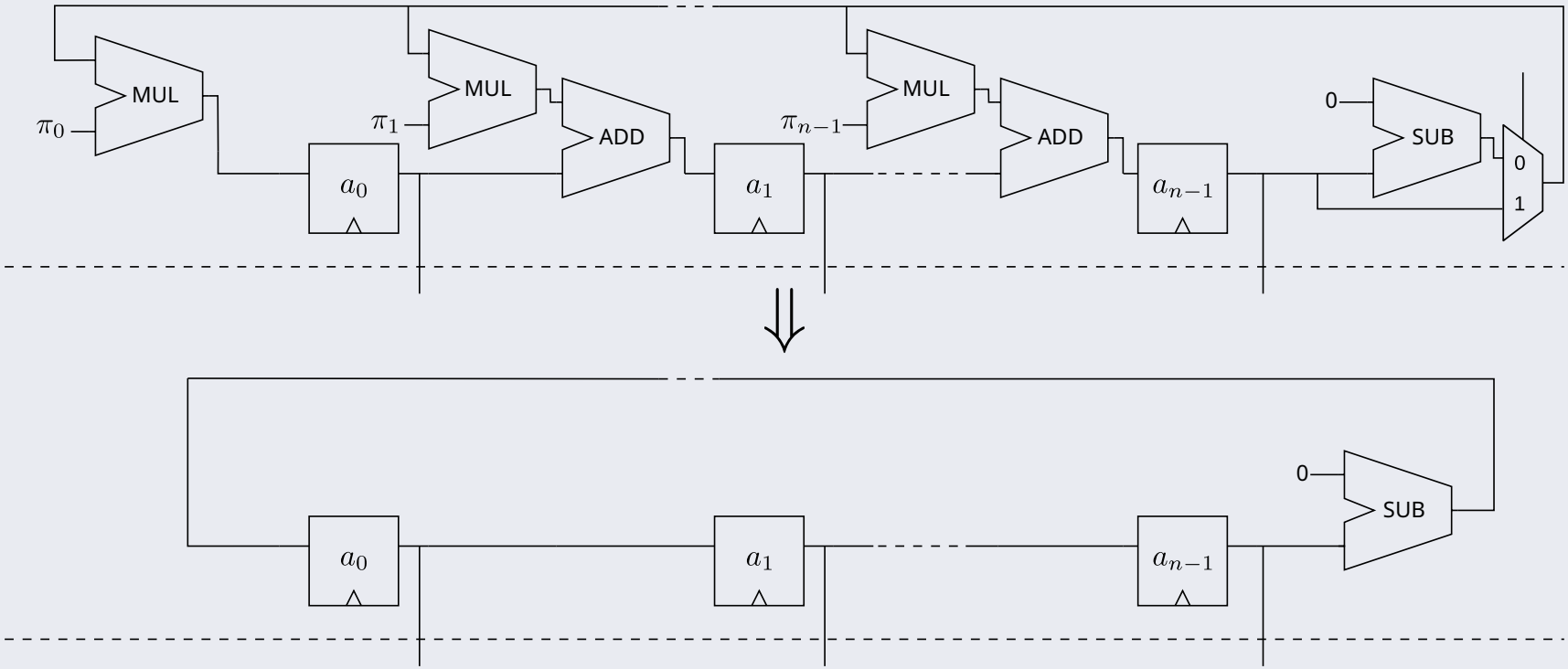


Feedback network for NTRU Prime

Optimizations applicable to x-net architecture

Taps of LFSR are in correspondence of non-zero coefficients of $\pi(x)$

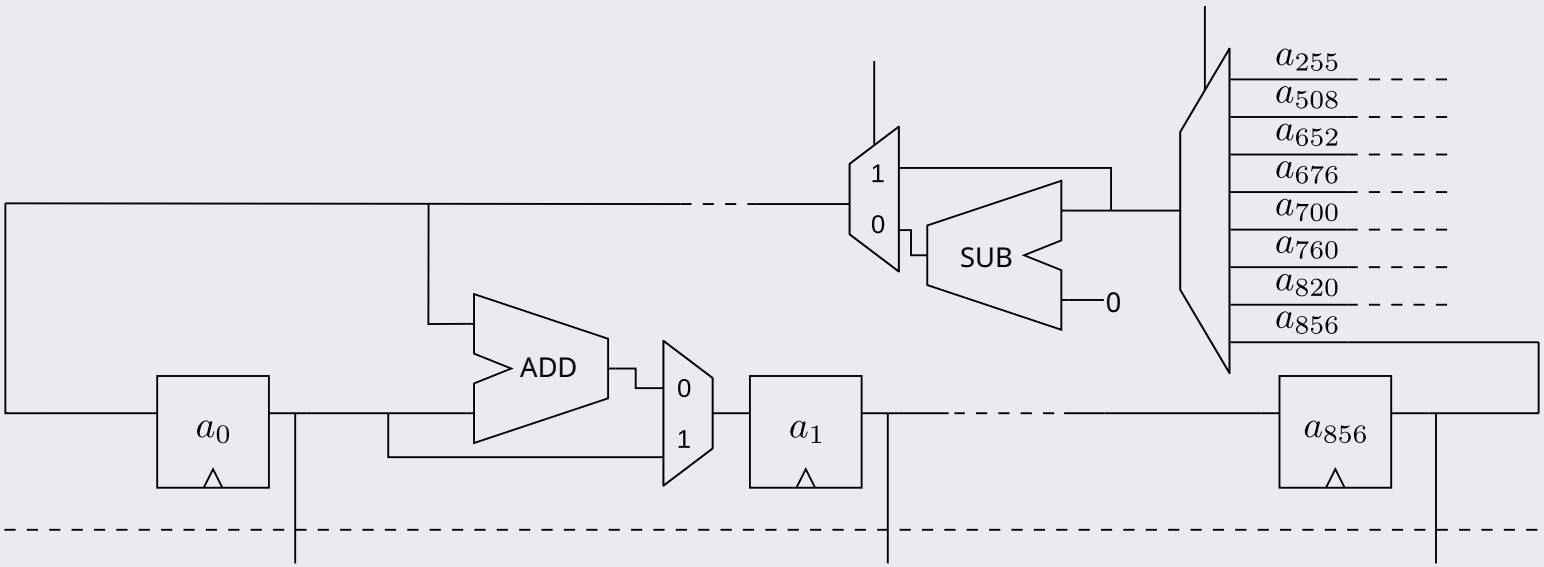
Only few of them are active: remove unused adders in feedback net



Feedback network for Saber and Kyber

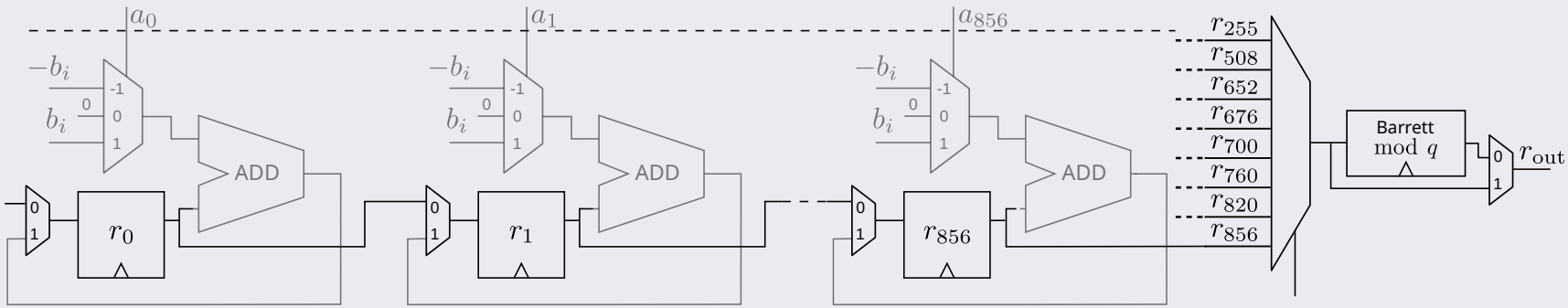
Unified architecture

- size the LFSR in order to accomodate the largest polynomial
- selection of the appropriate coefficient to feedback
  - optional flip of the sign (Saber, Kyber)
- conditionally enable addition on second register (NTRU Prime)



Unified architecture

- fix the number of MAC units as the largest one required
- accumulator sized to accomodate the maximum possible value
  - FFs per MAC unit: $\lceil \log_2 (\max\{2p_i q_i n_i : (p_i, q_i, n_i) \in \text{param. sets}\}) \rceil$
- reduction $\bmod q$ upon read-out
  - select the correct accumulator coefficient
  - either with Barrett module or truncation depending on the value of q



Speed-up computation

3 phases taking n CC each: `load` operand, `compute`, `read` result

Key idea: pack more coefficients in a transferred memory block

Fetch blocks of α coefficients of $a(x)$

Each clock cycle rotate the LFSR by α positions during `load` phase

Fetch blocks of β coefficients of $b(x)$

In the `compute` phase, every clock cycle compute $\beta \cdot n$ coefficient-wise multiplications, and rotate the LFSR by β positions

Send blocks of γ coefficients of $r(x)$

Shift by γ positions the data from the accumulator during `read` phase.
For coefficient $\bmod q$ reduction during read-out, use γ parallel Barrett modules

$$\text{Poly mult. latency} = \lceil n/\alpha \rceil + \lceil n/\beta \rceil + \lceil n/\gamma \rceil$$

Design Space Exploration

Setup of the Design Space Exploration

- Xilinx UltraScale+ ZCU106 FPGA `xczu7ev-ffvc1156-3-e`
- Vivado 2021.1
- `Flow_AlternateRoutability` synthesis strategy
- `Performance_NetDelay_high` implementation strategy

Repeated synthesis with target frequency from a binary search depending on previous synthesis results

Automated parsing of area and time reports from Vivado, joined with data from simulations (latency and performance counters)

Up to 30 parallel synthesis using 128 `x86_64` cores and 512 GB RAM

Design Space Exploration of scheme-specific architecture

Parameters

- every polynomial ring defined by the the parameter sets of Kyber, Saber, NTRU and NTRU Prime
- pack 1 or 4 small coefficients (α), and 1 or 2 large coefficients (β and γ)
- coefficient ring reduction at read-out or every clock cycle

138 designs, with $\sim$ 8 synthesis per design to establish the maximum working frequency with a binary search algorithm

$$\Longrightarrow \sim 1104 \text{ synthesis}$$

Output

- Area: LUTs, FFs, CLBs
- Latency: CC and μs of encapsulation and decapsulation
- Efficiency: Time-Area product (CLBs $\cdot$ μs)

Design Space Exploration of scheme-specific architecture

Results

- time spent in poly multiplications is larger in module-based schemes; the difference increases with the security level
- delaying the $\bmod q$ on read-out yelds:
  - a massive drop in logic resources utilization
  - slight increase of working frequency
  - moderate increase in FFs
- $\beta = 2$ has worse efficiency due to considerable network congestion

work	parameter set	CLB	freq.	CC	LUT	FF	BRAM
x-net	sntrup761	6757	312	762	38798	21768	2
[1]	sntrup761	9699	255	762	65207	32929	6
x-net	ntruhrss701	3088	588	702	18383	10898	2
[1]	ntruhrss701	5476	300	702	33230	32327	6
x^2-net	ntruhps4096821	7766	412	413	46293	12029	2
[2]	ntruhps4096821	8728	187	412	54478	9227	-
x^2-net	firesaber	5602	338	129	30809	4654	2
[2]	firesaber	3427	310	128	22127	7841	-

Design Space Exploration of the unified design

Parameters

- support all the polynomial rings defined by the parameter sets of Kyber, Saber, NTRU and NTRU Prime up to security level 5 (AES256 equiv.)
- pack 4 small coefficients (α), and 1 large coefficient (β and γ)

Results

supported ciphers	CLB	freq.	LUT	FF	CARRY8
NTRU, NTRU Prime Saber, Kyber	15155	250	92575	27171	3452
NTRU, NTRU Prime Kyber	11318	250	72089	25439	3442

When comparing the results w.r.t. the largest and slowest specialized design:

- only 36% of area penalty
  - leaving out the support to Saber, penalty drops to less than 2%
- the running frequency is just 15% slower

Conclusions

Conclusions

We produced a **parametrized** polynomial multiplier architecture which can **adapt at synthesis-time** to multiple quotient polynomial rings used by most important post-quantum lattice-based cryptographic KEMs:

- supports Kyber, Saber, NTRU and NTRU Prime
- our solutions outperform the current state-of-the-art for NTRU and NTRU Prime in terms of area ($-10\% \sim -40\%$), working frequency ($96\% \sim 120\%$), and efficiency

We designed an **unified** multiplier architecture capable of **selecting at runtime** the desired quotient polynomial ring

- low frequency drop (15%) and small area increase ($2\% \sim 36\%$) w.r.t. NTRU Prime SL5

Thanks for your attention

Francesco Antognazza
PhD student - Politecnico di Milano
email: francesco.antognazza@polimi.it

References I

[1] Farnoud Farahmand et al. “Evaluating the Potential for Hardware Acceleration of Four NTRU-Based Key Encapsulation Mechanisms Using Software/Hardware Codesign”. In: *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019, Chongqing, China, May 8-10, 2019 Revised Selected Papers*. Ed. by Jintai Ding and Rainer Steinwandt. Vol. 11505. Lecture Notes in Computer Science. Springer, 2019, pp. 23–43. DOI: 10.1007/978-3-030-25510-7_2. URL: https://doi.org/10.1007/978-3-030-25510-7_2.

[2] Elizabeth Carter, Pengzhou He, and Jiafeng Xie. “High-Performance Polynomial Multiplication Hardware Accelerators for KEM Saber and NTRU”. In: *IACR Cryptol. ePrint Arch.* (2022), p. 628. URL: https://eprint.iacr.org/2022/628.