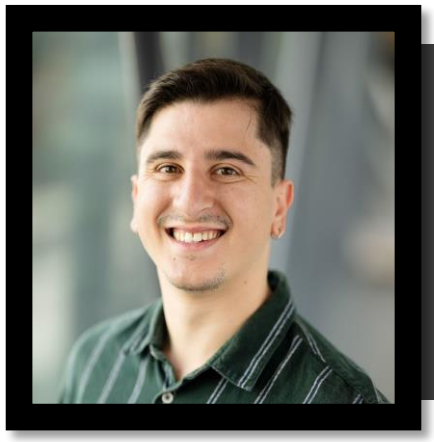
Optimizing HQC using Frobenius Additive FFT on a RISC-V-based System-on-Chip

Antonio Ras, Mikael Carmona, Antoine Loiseau, Simon Pontié, Emanuele Valea, Guénaël Renault, Benjamin Smith



Antonio RAS

PhD Candidate, **CEA-LETI**
Security of Components Laboratory (LSCO)

Sixth PQC Standardization Conference
25 September 2025

Agenda

1. HQC: a quantum-safe cryptosystem
2. Improving multiplication operation
3. ANDROMEDA hardware solution
4. Experimental results
5. Conclusion

Agenda

1. **HQC: a quantum-safe cryptosystem**
2. Improving multiplication operation
3. ANDROMEDA hardware solution
4. Experimental results
5. Conclusion

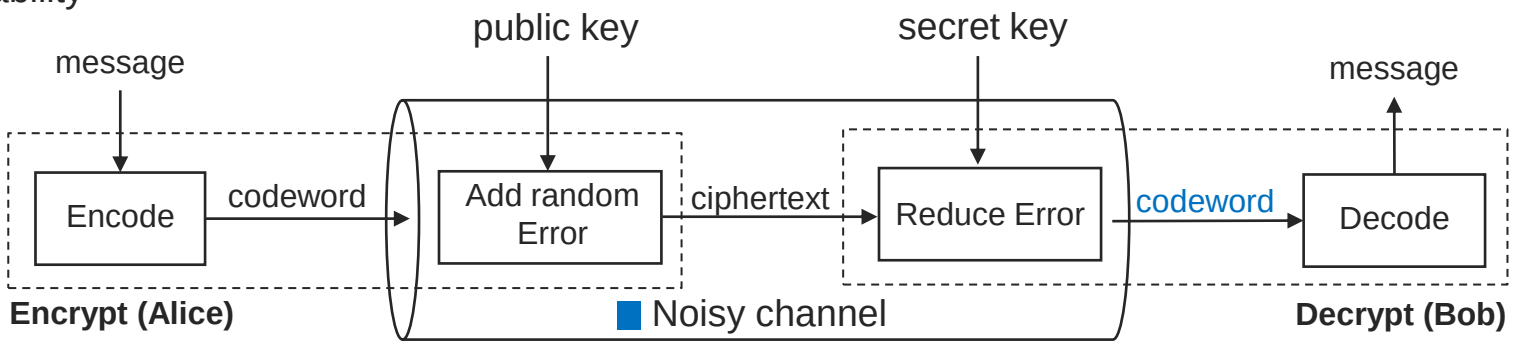
HQC cryptosystem

➢ Hamming Quasi-Cyclic (HQC)

- *Quasi-Cyclic Syndrome Decoding* hard problem (NP-complete)
- Based on error correcting code theory
- Decode function has fixed correction bits capability

- Three different security levels

Instance	n_1	n_2	n	w	$w_r = w_e$	security
hqc-128	46	384	17,669	66	75	128
hqc-192	56	640	35,851	100	114	192
hqc-256	90	640	57,637	131	149	256



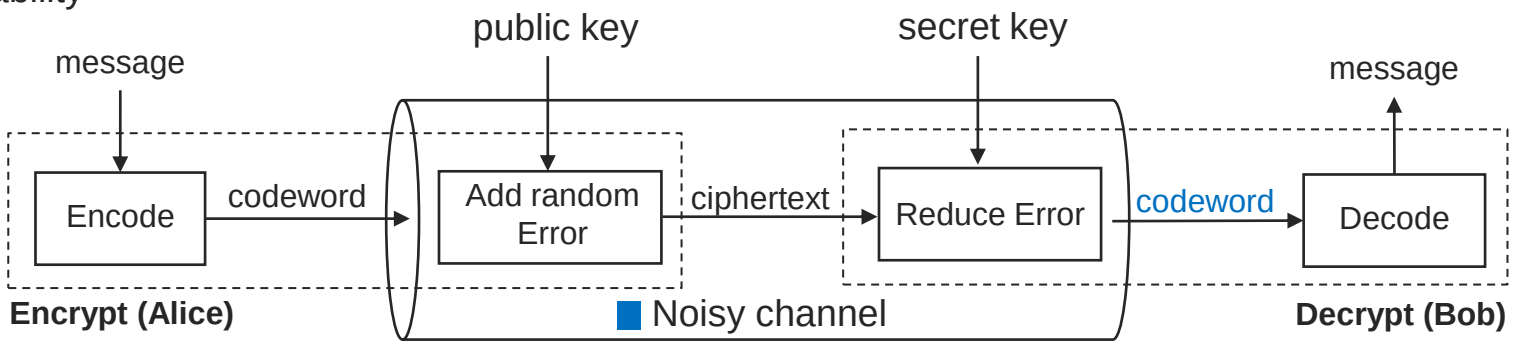
HQC cryptosystem

Hamming Quasi-Cyclic (HQC)

- *Quasi-Cyclic Syndrome Decoding* hard problem (NP-complete)
- Based on error correcting code theory
- Decode function has fixed correction bits capability

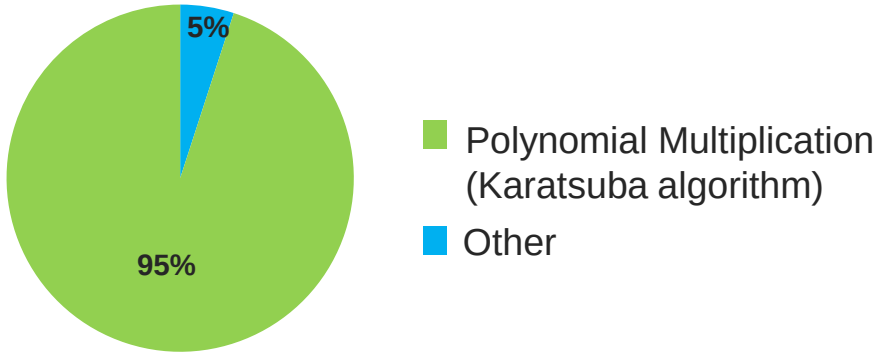
- Three different security levels

Instance	n_1	n_2	n	w	$w_r = w_e$	security
hqc-128	46	384	17,669	66	75	128
hqc-192	56	640	35,851	100	114	192
hqc-256	90	640	57,637	131	149	256



HQC software profiling

- Identify operation's bottleneck over HQC execution time
- Profiling on RISC-V based platform



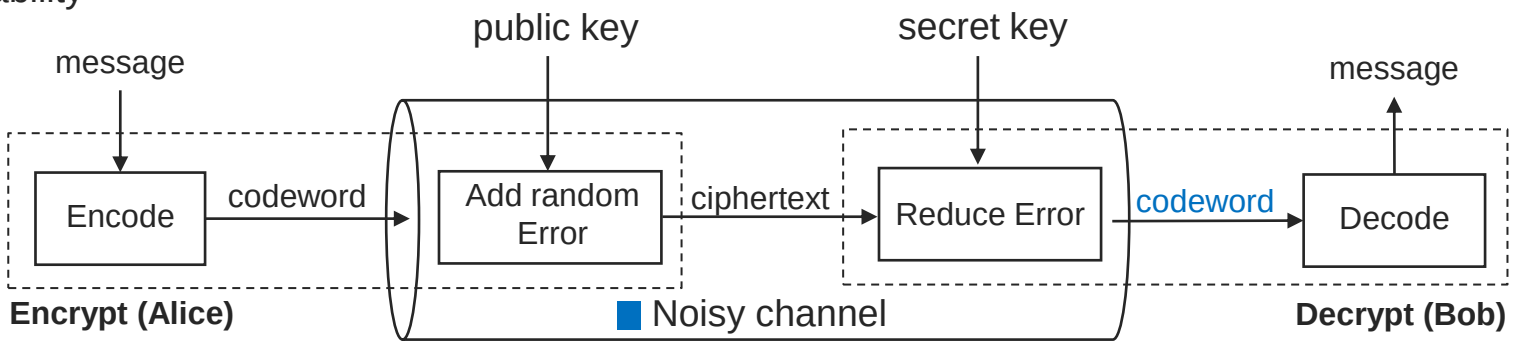
HQC cryptosystem

➢ Hamming Quasi-Cyclic (HQC)

- *Quasi-Cyclic Syndrome Decoding* hard problem (NP-complete)
- Based on error correcting code theory
- Decode function has fixed correction bits capability

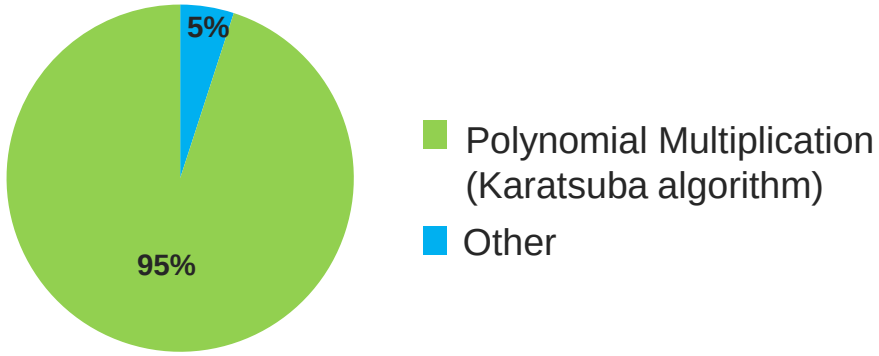
- Three different security levels

Instance	n_1	n_2	n	w	$w_r = w_e$	security
hqc-128	46	384	17,669	66	75	128
hqc-192	56	640	35,851	100	114	192
hqc-256	90	640	57,637	131	149	256



➢ HQC software profiling

- Identify operation's bottleneck over HQC execution time
- Profiling on RISC-V based platform



Our contribution

The alternative **Frobenius Additive FFT** for HQC polynomial multiplication and its first state-of-the-art **hardware implementation**!

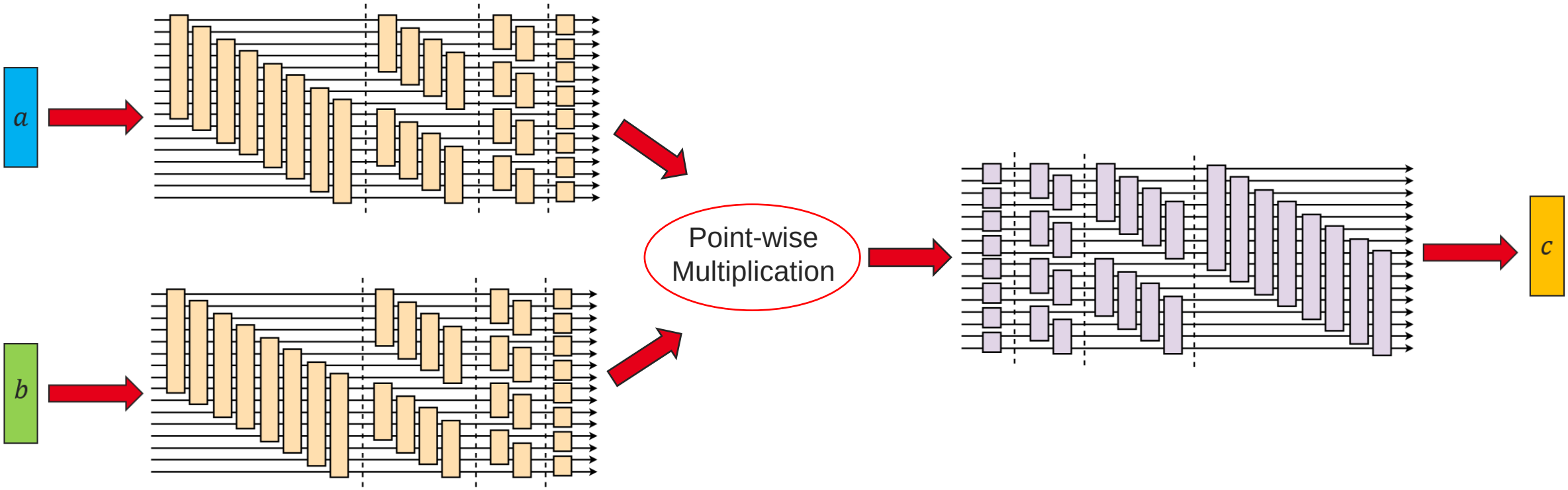
Agenda

1. HQC: a quantum-safe cryptosystem
2. **Improving multiplication operation**
3. ANDROMEDA hardware solution
4. Experimental results
5. Conclusion

FFT-based polynomial multiplication

How to perform multiplication

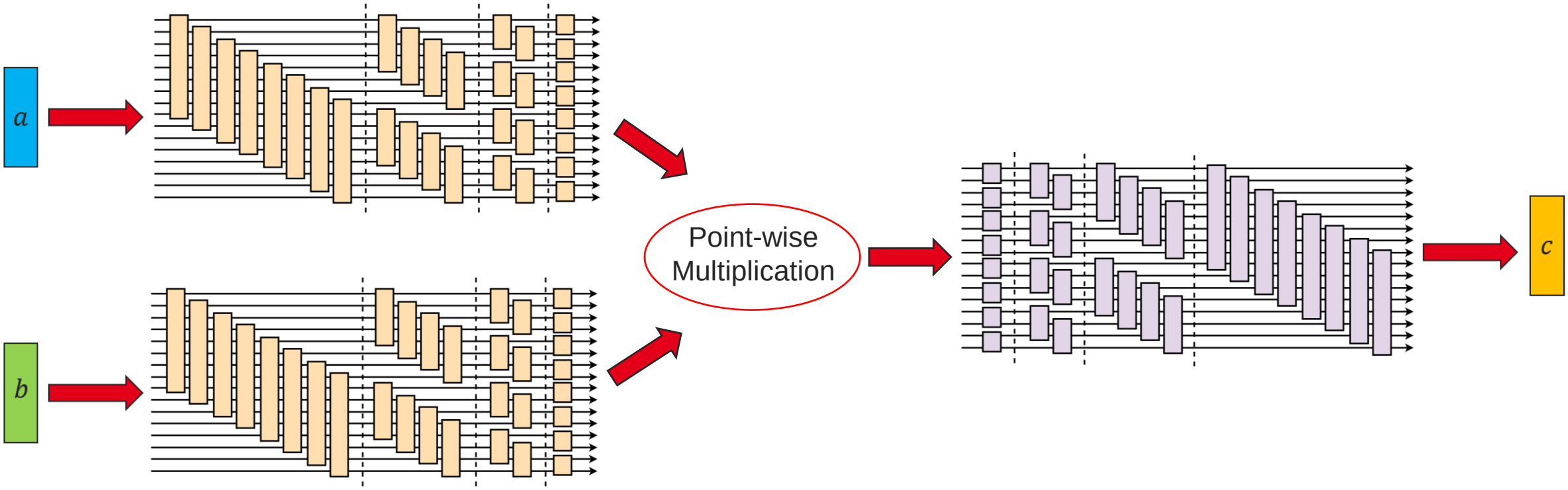
- Evaluating input polynomials in a different domain using an evaluation points set → **FFT step**
- Perform multiplication between evaluated points → **PWM step**
- Interpolate back to get final result → **Inverse FFT (IFFT) step**



FFT-based polynomial multiplication

How to perform multiplication

- Evaluating input polynomials in a different domain using an evaluation points set → **FFT step**
- Perform multiplication between evaluated points → **PWM step**
- Interpolate back to get final result → **Inverse FFT (IFFT) step**

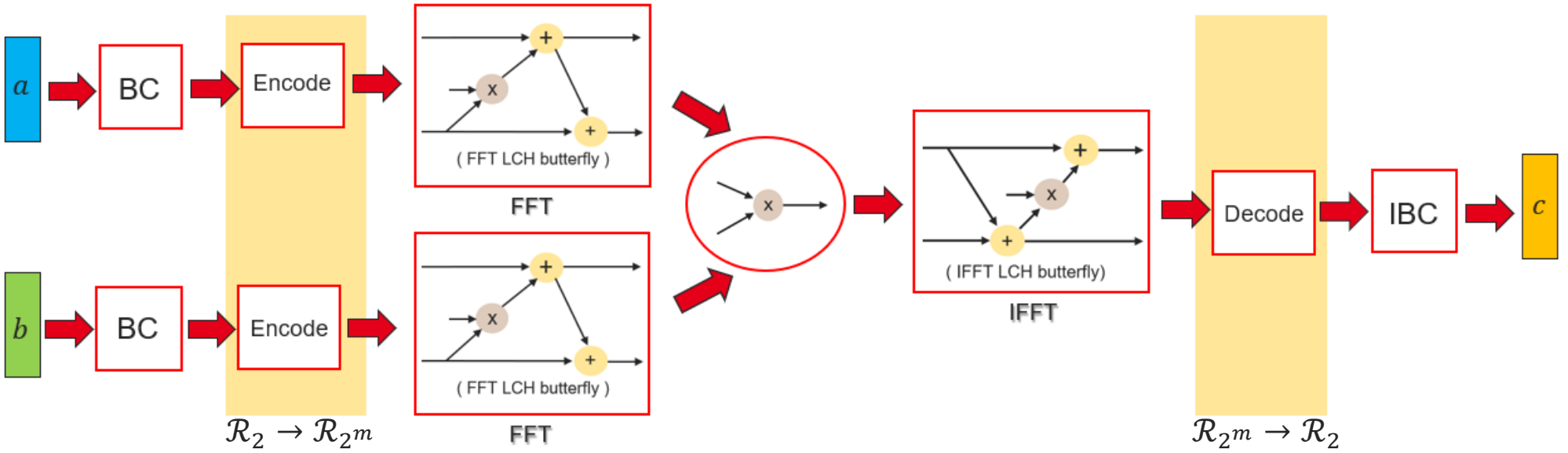


FFT and IFFT are performed using different **processing elements,** and

Frobenius Additive FFT (FAFFT)

➢ FFT-like Binary power-of-two polynomial multiplication

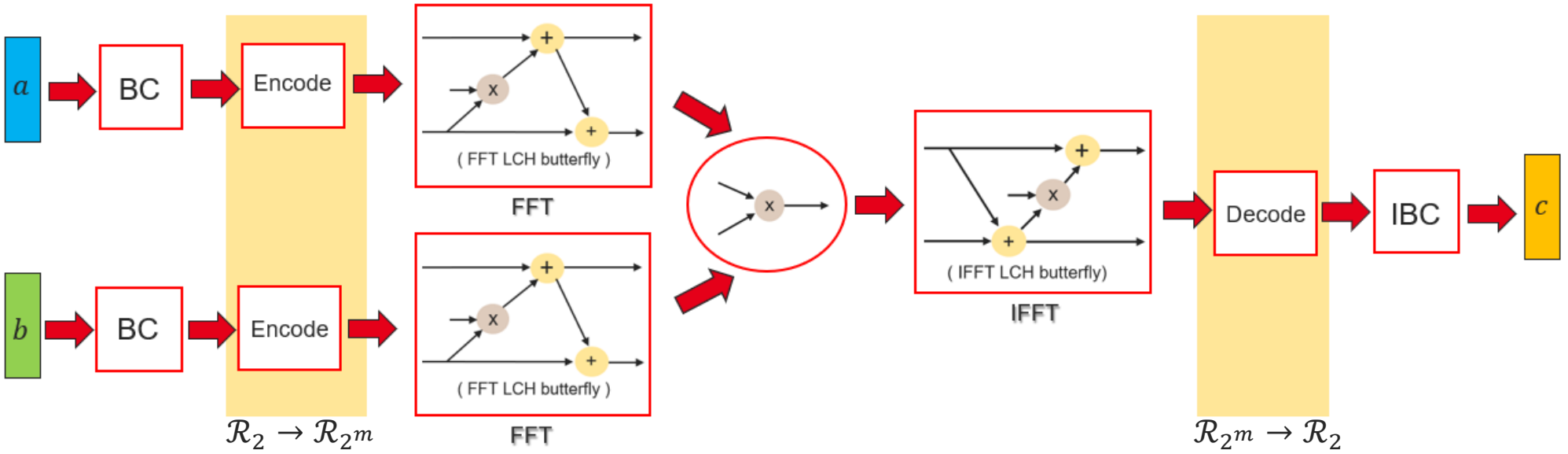
- Alternative to Karatsuba approach to improve HQC performance
- New polynomial basis using **Cantor basis**
- Encode and Decode : **Frobenius map** reduce evaluation points



Frobenius Additive FFT (FAFFT)

➢ FFT-like Binary power-of-two polynomial multiplication

- Alternative to Karatsuba approach to improve HQC performance
- New polynomial basis using **Cantor basis**
- Encode and Decode : **Frobenius map** reduce evaluation points



Which operation we compute in practice ?

- $\mathcal{R}_{2^m} := \mathbb{Z}_2[x]/\, g(x)$
- $m = 32$ and $g(x) = x^{32} + x^{22} + x^2 + x + 1$

Carryless addition

$$h = f \oplus g$$

Modular carryless multiplication

$$h = (f * g) \; mod \; g(x)$$

Shift-and-Add strategy

[CCK+18] Chen, Cheng, Li et al. «Multiplying boolean Polynomials with Frobenius Partitions in Additive Fast Fourier Transform», 2018

HQC using FAFFT in software

➢ FAFFT implementation

- Implementation in C, without ASM optimization
- Inspired by state-of-the-art solution **[CCK+18] [CCK21]**
- **First state-of-the art FAFFT integration in HQC**

➢ HQC using FAFFT in software

- RISC-V based embedded platform
- Result in number of clock cycles
- **[AMAB+24]** as C baseline implementation on RISC-V platform
- Speed up **3.52x / 4.98x / 8.08x** for respective HQC-X

Platform	Work	Level	KEM.KeyGen	KEM.Encaps	KEM.Decaps
RISC-V (CV32E40P)	SW HQC [AMAB+24]	HQC-128	43,592,464	87,574,395	132,101,254
		HQC-192	132,528,662	265,882,733	399,562,345
		HQC-256	242,546,831	486,241,524	731,405,854
	Standard SW HQC + FAFFT (This Work)	HQC-128	12,208,482 [**3.57**×]	24,882,981 [**3.52**×]	38,099,269 [**3.47**×]
		HQC-192	26,347,517 [**5.03**×]	53,528,998 [**4.97**×]	81,043,401 [**4.93**×]
		HQC-256	29,545,963 [**8.21**×]	60,238,075 [**8.07**×]	91,987,793 [**7.95**×]

[AMAB+24] C. Aguilar, N. Aragon et al. «HQC (Hamming Quasi-Cyclic),», 2024
[CCK+18] Chen, Cheng, Li et al. «Multiplying boolean Polynomials with Frobenius Partitions in Additive Fast Fourier Transform», 2018
[CCK21] Chen et al, "Optimizing BIKE for the Intel Haswell and ARM Cortex-M4", 2021

Agenda

1. HQC: a quantum-safe cryptosystem
2. Improving multiplication operation
3. **ANDROMEDA hardware solution**
4. Experimental results
5. Conclusion

Accelerating FAFFT in hardware

➢ FAFFT profiling

- Result in number of clock cycles
- FFT-like and PWM steps around 75% of execution time
- **Hardware-based acceleration strategy**

Function	Calls	HQC-128		HQC-192/HQC-256	
		Clock cycles	[Total %]	Clock cycles	[Total %]
`BasisCvt`	×2	185,965	1,7	413,039	1,8
`Encode`	×2	680,206	6,4	1,368,694	6,0
$\texttt{FFT}_{\text{LCH}}$	×2	5,122,525	48,2	11,133,447	49,1
$\texttt{PWM}_{\text{LCH}}$	×1	432,141	4,0	864,269	3,8
$\texttt{IFFT}_{\text{LCH}}$	×1	2,559,225	24,1	5,562,638	24,6
`Decode`	×1	1,403,060	13,2	2,806,495	12,5
`iBasisCvt`	×1	175,913	1,7	367,138	1,6
Other	-	72,669	0,7	137,455	0,6
	Total	**10,630,704**	**100**	**22,653,175**	**100**

Accelerating FAFFT in hardware

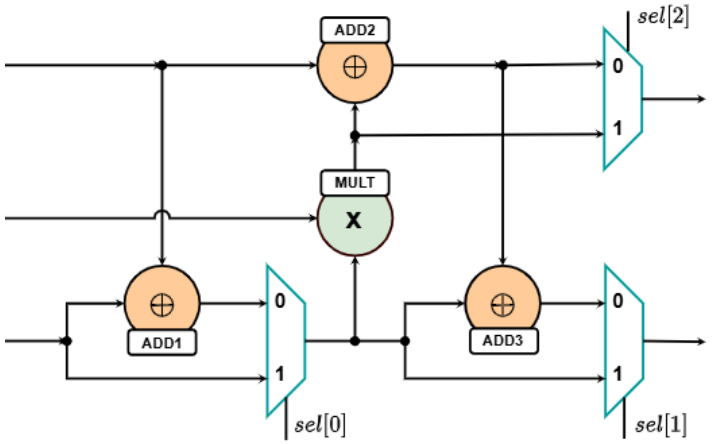
➢ FAFFT profiling

- Result in number of clock cycles
- FFT-like and PWM steps around 75% of execution time
- **Hardware-based acceleration strategy**

Function	Calls	HQC-128 Clock cycles	HQC-128 [Total %]	HQC-192/HQC-256 Clock cycles	HQC-192/HQC-256 [Total %]
`BasisCvt`	×2	185,965	1,7	413,039	1,8
`Encode`	×2	680,206	6,4	1,368,694	6,0
$\texttt{FFT}_{\text{LCH}}$	×2	5,122,525	48,2	11,133,447	49,1
$\texttt{PWM}_{\text{LCH}}$	×1	432,141	4,0	864,269	3,8
$\texttt{IFFT}_{\text{LCH}}$	×1	2,559,225	24,1	5,562,638	24,6
`Decode`	×1	1,403,060	13,2	2,806,495	12,5
`iBasisCvt`	×1	175,913	1,7	367,138	1,6
Other	-	72,669	0,7	137,455	0,6
	Total	**10,630,704**	**100**	**22,653,175**	**100**

➢ Carry-Free Butterfly (CFB)

- Configurable hardware design
- Manage FFT, PWM and IFFT butterfly
- Two carryless operations
- Karatsuba-based multiplier approach



Accelerating FAFFT in hardware

➢ FAFFT profiling

- Result in number of clock cycles
- FFT-like and PWM steps around 75% of execution time
- **Hardware-based acceleration strategy**

Function	Calls	HQC-128 Clock cycles	HQC-128 [Total %]	HQC-192/HQC-256 Clock cycles	HQC-192/HQC-256 [Total %]
BasisCvt	×2	185,965	1,7	413,039	1,8
Encode	×2	680,206	6,4	1,368,694	6,0
FFT_{LCH}	×2	5,122,525	48,2	11,133,447	49,1
PWM_{LCH}	×1	432,141	4,0	864,269	3,8
IFFT_{LCH}	×1	2,559,225	24,1	5,562,638	24,6
Decode	×1	1,403,060	13,2	2,806,495	12,5
iBasisCvt	×1	175,913	1,7	367,138	1,6
Other	-	72,669	0,7	137,455	0,6
	Total	**10,630,704**	**100**	**22,653,175**	**100**

➢ Carry-Free Butterfly (CFB)

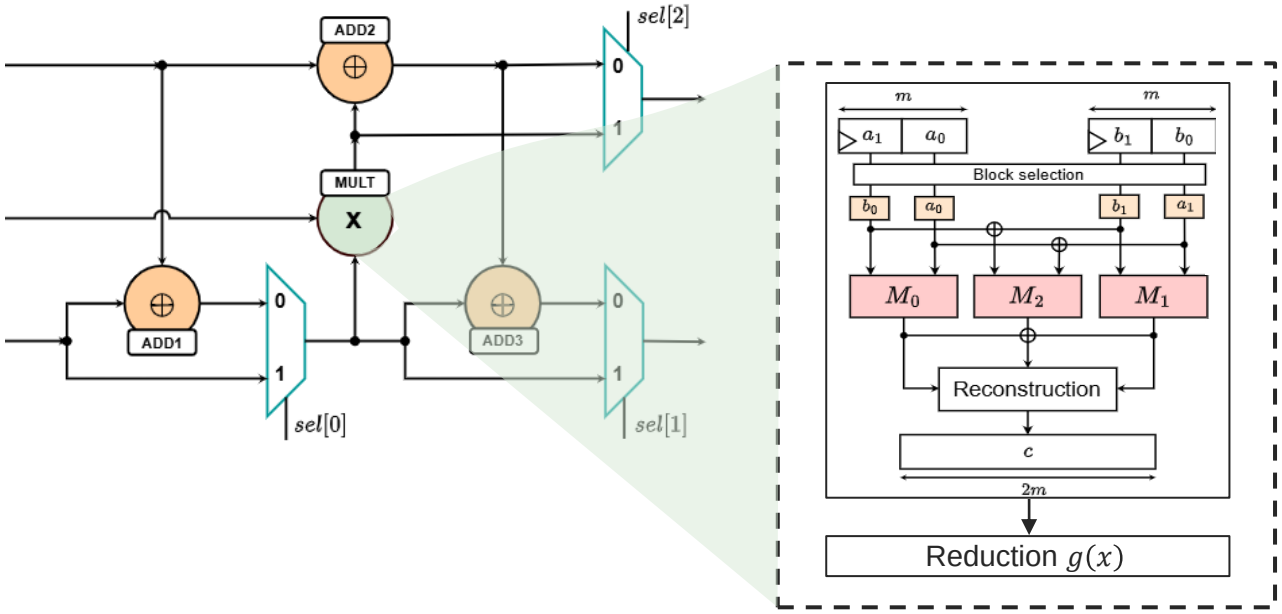
- Configurable hardware design
- Manage FFT, PWM and IFFT butterfly
- Two carryless operations
- Karatsuba-based multiplier approach

$$a = (a_1, a_0) \qquad b = (b_1, b_0)$$

$$z_0 = a_0 \times_{M_0} b_0 \qquad z_1 = a_1 \times_{M_1} b_1$$

$$z_2 = [(a_1 \oplus a_0) \times_{M_2} (b_1 \oplus b_0)] \oplus z_0 \oplus z_1$$

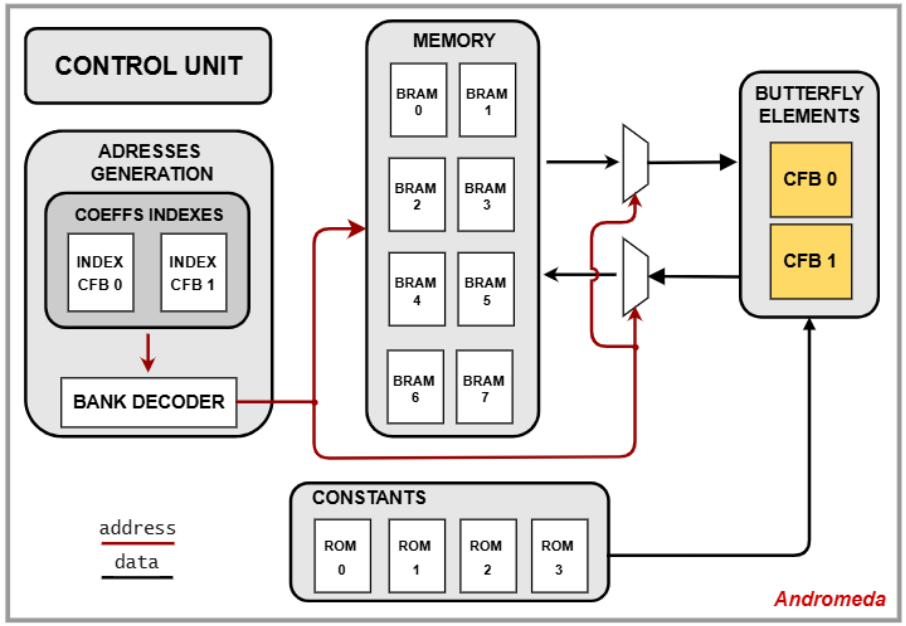
- Schoolbook-based M_x multiplier



ANDROMEDA hardware design

➢ Design Rationale

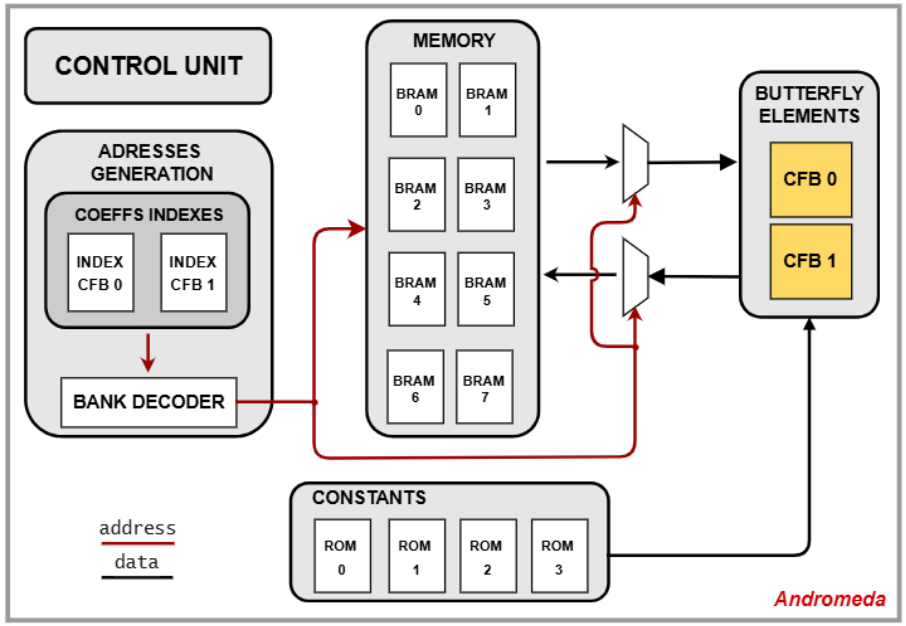
- **Loosely-coupled accelerator**
- **Four main components**
  - **Butterfly elements :** CFBs valorisation
  - **Address generation :** Coefficient pairs selection
  - **Memories:** Polynomial and Constants
  - **Control Unit :** Orchestrate everything
- Maximum frequency of **125 MHz, Artix-7 FPGA**



ANDROMEDA hardware design

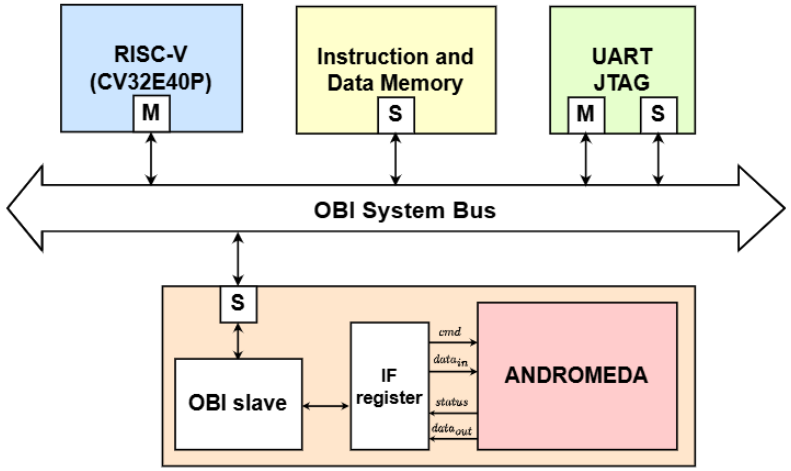
➢ Design Rationale

- Loosely-coupled accelerator
- Four main components
  - **Butterfly elements :** CFBs valorisation
  - **Address generation :** Coefficient pairs selection
  - **Memories:** Polynomial and Constants
  - **Control Unit :** Orchestrate everything
- Maximum frequency of **125 MHz, Artix-7 FPGA**



➢ System-on-Chip integration

- **RISC-V based system on chip on Kintex-7 FPGA**
- 256 KB data and instruction memory
- OBI system bus
- All NIST security levels



Agenda

1. HQC: a quantum-safe cryptosystem
2. Improving multiplication operation
3. ANDROMEDA hardware solution
4. **Experimental results**
5. Conclusion

Experimental results

Absence of existing hardware FAFFT implementation in state-of-the-art

➢ HQC Performance

- Result in number of clock cycles
- **[AMAB+24]** as C baseline implementation on RISC-V platform
- Speed up **2.74x / 2.79x / 2.38x** over alternative HQC + FAFFT, in software, respective HQC-X
- Overall acceleration of **9.64x / 13.87x / 19.22x** over baseline HQC

Platform	Work	Level	KEM.KeyGen	KEM.Encaps	KEM.Decaps
RISC-V (CV32E40P)	SW HQC [AMAB+24]	HQC-128	43,592,464	87,574,395	132,101,254
		HQC-192	132,528,662	265,882,733	399,562,345
		HQC-256	242,546,831	486,241,524	731,405,854
	Standard SW HQC + FAFFT	HQC-128	12,208,482	24,882,981	38,099,269
		HQC-192	26,347,517	53,528,998	81,043,401
		HQC-256	29,545,963	60,238,075	91,987,793
	Standard HW-SW HQC + FAFFT (This Work)	HQC-128	4,308,552 [**2.83**×]	9,093,313 [**2.74**×]	14,395,188 [**2.65**×]
		HQC-192	9,233,023 [**2.85**×]	19,273,343 [**2.78**×]	29,659,916 [**2.73**×]
		HQC-256	11,963,005 [**2.47**×]	25,072,159 [**2.40**×]	40,604,459 [**2.27**×]

[AMAB+24] C. Aguilar, N. Aragon et al. «HQC (Hamming Quasi-Cyclic),», 2024

Experimental results

Absence of existing hardware FAFFT implementation in state-of-the-art

➢ HQC Performance

- Result in number of clock cycles
- **[AMAB+24]** as C baseline implementation on RISC-V platform
- Speed up **2.74x / 2.79x / 2.38x** over alternative HQC + FAFFT, in software, respective HQC-X
- Overall acceleration of **9.64x / 13.87x / 19.22x** over baseline HQC

Platform	Work	Level	KEM.KeyGen	KEM.Encaps	KEM.Decaps
RISC-V (CV32E40P)	SW HQC [AMAB+24]	HQC-128	43,592,464	87,574,395	132,101,254
		HQC-192	132,528,662	265,882,733	399,562,345
		HQC-256	242,546,831	486,241,524	731,405,854
	Standard SW HQC + FAFFT	HQC-128	12,208,482	24,882,981	38,099,269
		HQC-192	26,347,517	53,528,998	81,043,401
		HQC-256	29,545,963	60,238,075	91,987,793
	Standard HW-SW HQC + FAFFT (This Work)	HQC-128	4,308,552 [**2.83**×]	9,093,313 [**2.74**×]	14,395,188 [**2.65**×]
		HQC-192	9,233,023 [**2.85**×]	19,273,343 [**2.78**×]	29,659,916 [**2.73**×]
		HQC-256	11,963,005 [**2.47**×]	25,072,159 [**2.40**×]	40,604,459 [**2.27**×]

➢ ANDROMEDA hardware resources

- Our two configurable CFB design only accounts for 14% of total resource utilization

	LUT	FF	BRAM	DSP	SEC
ANDROMEDA	**2482**	**1158**	**8**	**0**	**2356**
⌊ 1× CFB butterfly	600	127	0	0	166

[AMAB+24] C. Aguilar, N. Aragon et al. «HQC (Hamming Quasi-Cyclic),», 2024

Agenda

1. HQC: a quantum-safe cryptosystem
2. Improving multiplication operation
3. ANDROMEDA hardware solution
4. Experimental results
5. **Conclusion**

Conclusion

Quantum-safe transition

- Current public-key cryptography weak against quantum computer
- Future NIST standard scheme : **Hamming Quasi-Cyclic (HQC)**
- Alternative method for the main HQC's bottleneck : **Polynomial Multiplication**

Our contribution

- Implementation and integration of **Frobenius Additive FFT (FAFFT) in HQC**
- Dedicated hardware design, **ANDROMEDA**, targeting main FAFFT's bottleneck
- Performance improvements for software-only and hardware-software co-design scenario in RISC-V based System-on-Chip
- This work has been published in the Digital System Design (DSD) 2025 conference

Conclusion

Quantum-safe transition

- Current public-key cryptography weak against quantum computer
- Future NIST standard scheme : **Hamming Quasi-Cyclic (HQC)**
- Alternative method for the main HQC's bottleneck : **Polynomial Multiplication**

Our contribution

- Implementation and integration of **Frobenius Additive FFT (FAFFT) in HQC**
- Dedicated hardware design, **ANDROMEDA**, targeting main FAFFT's bottleneck
- Performance improvements for software-only and hardware-software co-design scenario in RISC-V based System-on-Chip
- This work has been published in the Digital System Design (DSD) 2025 conference

Ongoing work

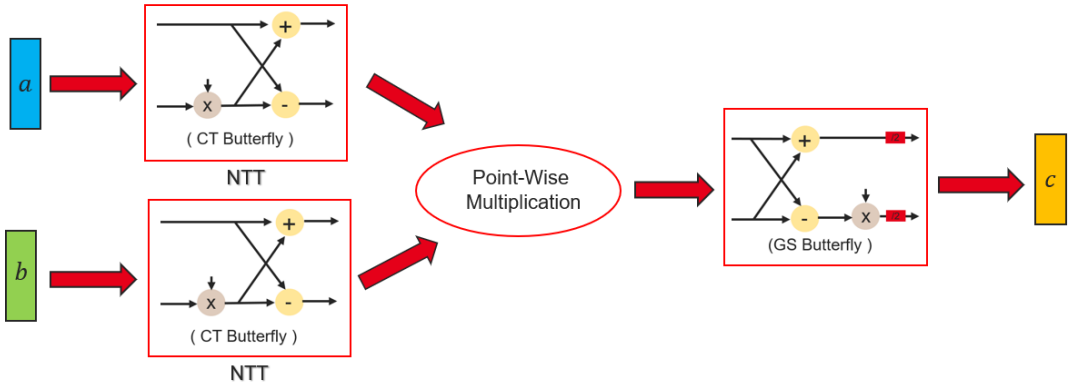
- Hardware-resources sharing on crypto-agility application between **HQC** and **ML-KEM**

Crypto-agility: ML-KEM & HQC

Sharing strategy

Polynomial multiplication operations:

- ***Frobenius Additive FFT*** **(FAFFT) for HQC**
- ***Number-Theoretic-Transform*** **(NTT) for ML-KEM**

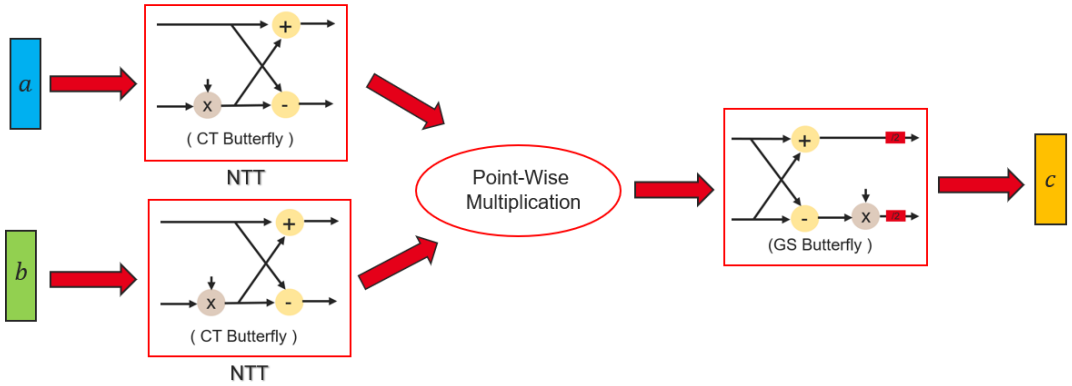


Crypto-agility: ML-KEM & HQC

Sharing strategy

Polynomial multiplication operations:

- ***Frobenius Additive FFT*** **(FAFFT) for HQC**
- ***Number-Theoretic-Transform*** **(NTT) for ML-KEM**



Hardware proposal

Super-Butterfly:

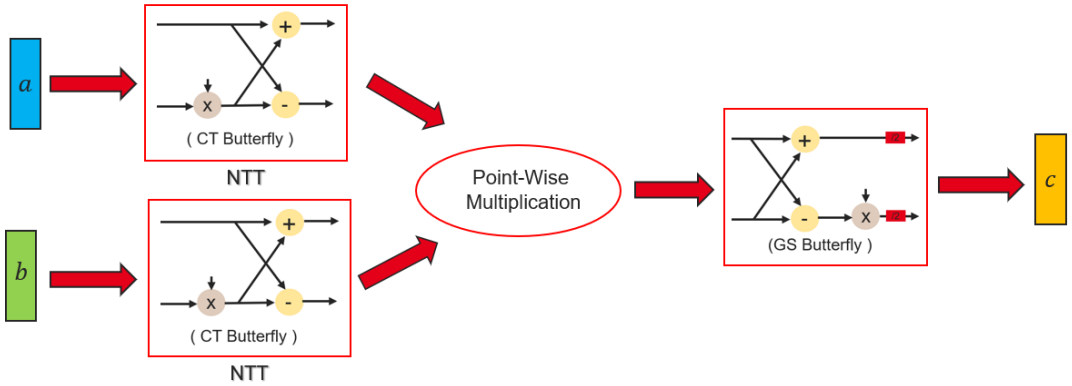
- Configurable hardware design for butterfly elements execution
- Carry-Free Butterfly (HQC) and NTT-based design (ML-KEM)
- **Goal:** Hardware-resource saving respect to standardalone designs

Crypto-agility: ML-KEM & HQC

Sharing strategy

Polynomial multiplication operations:

- ***Frobenius Additive FFT*** **(FAFFT) for HQC**
- ***Number-Theoretic-Transform*** **(NTT) for ML-KEM**



Hardware proposal

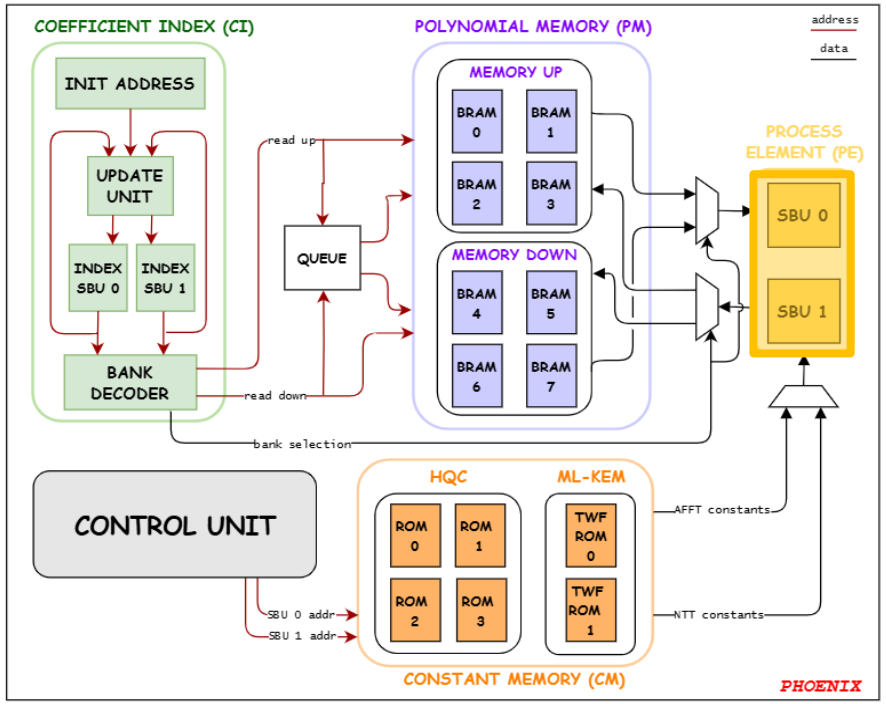
Super-Butterfly:

- Configurable hardware design for butterfly elements execution
- Carry-Free Butterfly (HQC) and NTT-based design (ML-KEM)
- **Goal:** Hardware-resource saving respect to standardalone designs

PHOENIX:

- Loosely-coupled accelerator for FAFFT and NTT operation
- Super-butterfly valorisation

Ras et al, «PHOENIX: Crypto-Agile Hardware Sharing for ML-KEM and HQC»

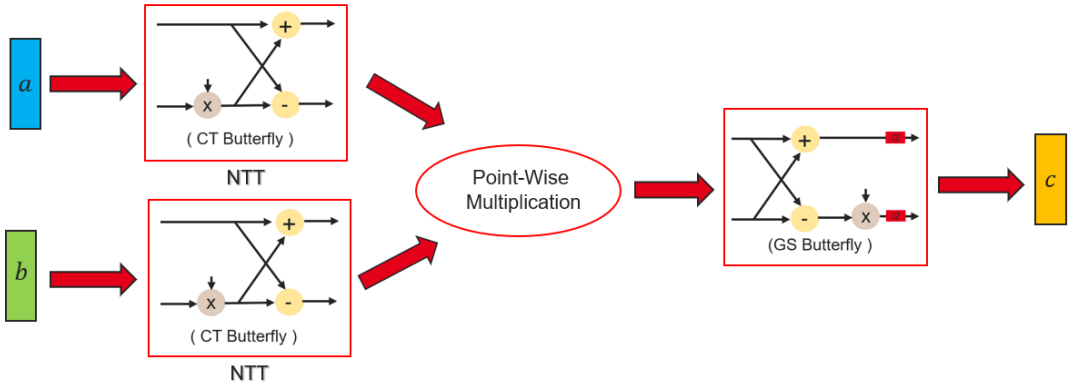


Crypto-agility: ML-KEM & HQC

Sharing strategy

Polynomial multiplication operations:

- ***Frobenius Additive FFT*** **(FAFFT) for HQC**
- ***Number-Theoretic-Transform*** **(NTT) for ML-KEM**



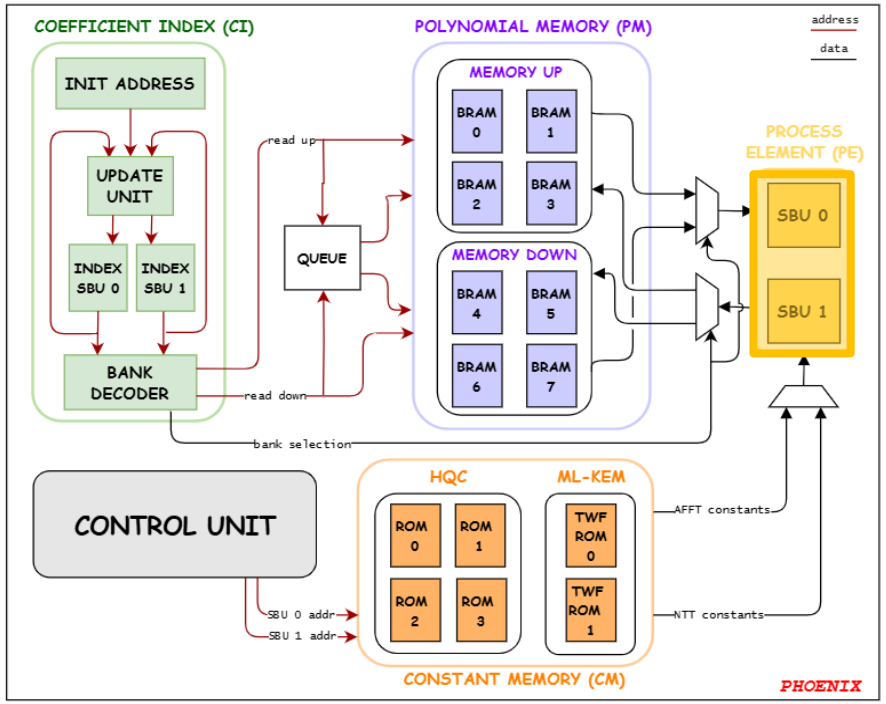
Hardware proposal

Super-Butterfly:

- Configurable hardware design for butterfly elements execution
- Carry-Free Butterfly (HQC) and NTT-based design (ML-KEM)
- **Goal:** Hardware-resource saving respect to standardalone designs

PHOENIX:

- Loosely-coupled accelerator for FAFFT and NTT operation
- Super-butterfly valorisation

Ras et al, «PHOENIX: Crypto-Agile Hardware Sharing for ML-KEM and HQC»

- **New result will be available soon!**