Acceleration of FHE/PCQ workloads on AI ASICs

George Alexakis

Electrical and Computer Engineering
Democritus University of Thrace, Xanthi, Greece

Why FHE and PQC?

Post-Quantum Cryptography

- Secure against quantum computers
- Replaces RSA/ECC
- Already standardized

Why FHE and PQC?

Post-Quantum Cryptography

- Secure against quantum computers
- Replaces RSA/ECC
- Already standardized

- ML-KEM (Key Encapsulation)
- ML-DSA (Digital Signatures)

Why FHE and PQC?

Post-Quantum Cryptography

- Secure against quantum computers
- Replaces RSA/ECC
- Already standardized

Fully Homomorphic Encryption

- Compute on encrypted data
- Enables encrypted cloud AI
- Computationally intensive

- ML-KEM (Key Encapsulation)
- ML-DSA (Digital Signatures)

Why FHE and PQC?

Post-Quantum Cryptography

- Secure against quantum computers
- Replaces RSA/ECC
- Already standardized

- ML-KEM (Key Encapsulation)
- ML-DSA (Digital Signatures)

Fully Homomorphic Encryption

- Compute on encrypted data
- Enables encrypted cloud AI
- Computationally intensive

- BGV/BFV (Integers)
- TFHE/FHEW (Booleans)
- CKKS (Real & Complex Numbers)

The Common Computational Kernel

- Polynomial multiplication dominates execution time

$$\boldsymbol{c(x) = a(x) \cdot b(x) \pmod q}$$

- Quadratic complexity ($\boldsymbol{O(n^2)}$)

$$\boldsymbol{c_k = \sum_{i=0}^{k} a_i b_{k-i} \pmod q}$$

The Common Computational Kernel

- Polynomial multiplication dominates execution time

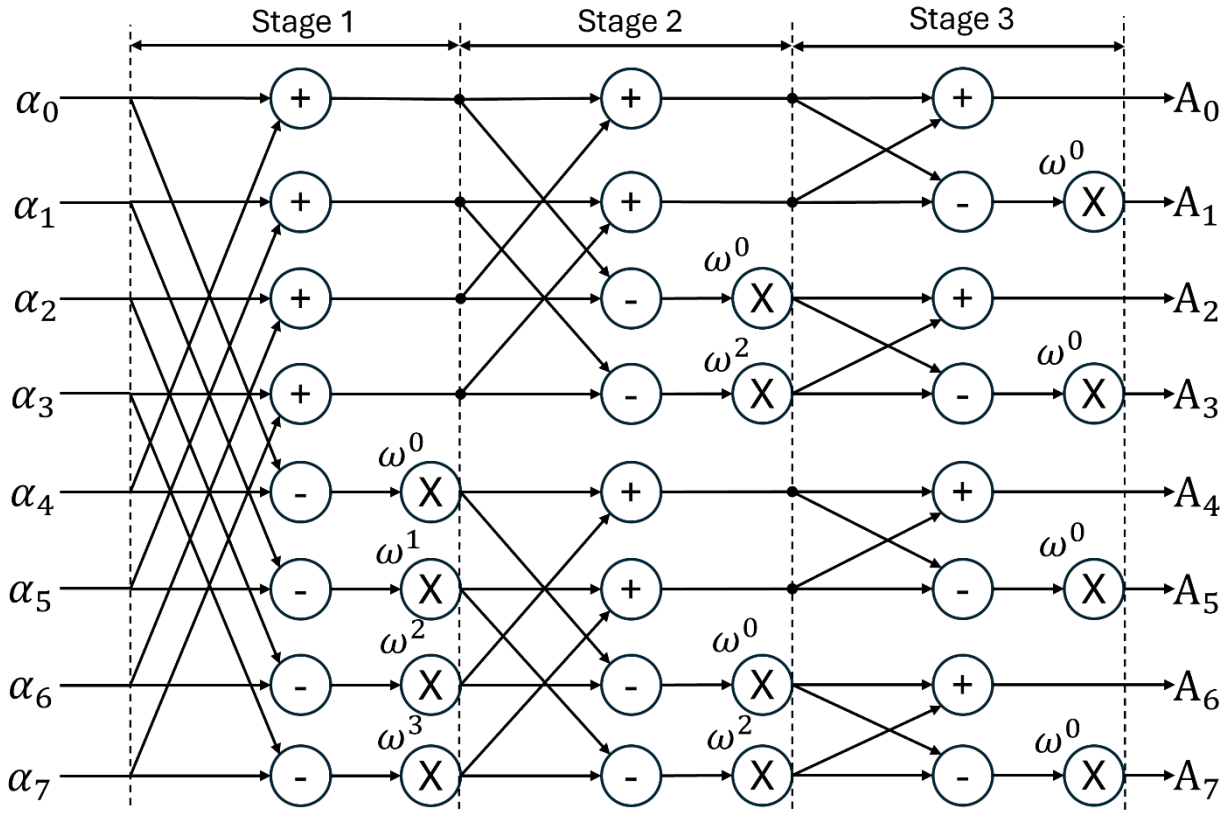
$$c(x) = a(x) \cdot b(x) \pmod q$$

- Quadratic complexity ($O(n^2)$)

$$c_k = \sum_{i=0}^{k} a_i b_{k-i} \pmod q$$

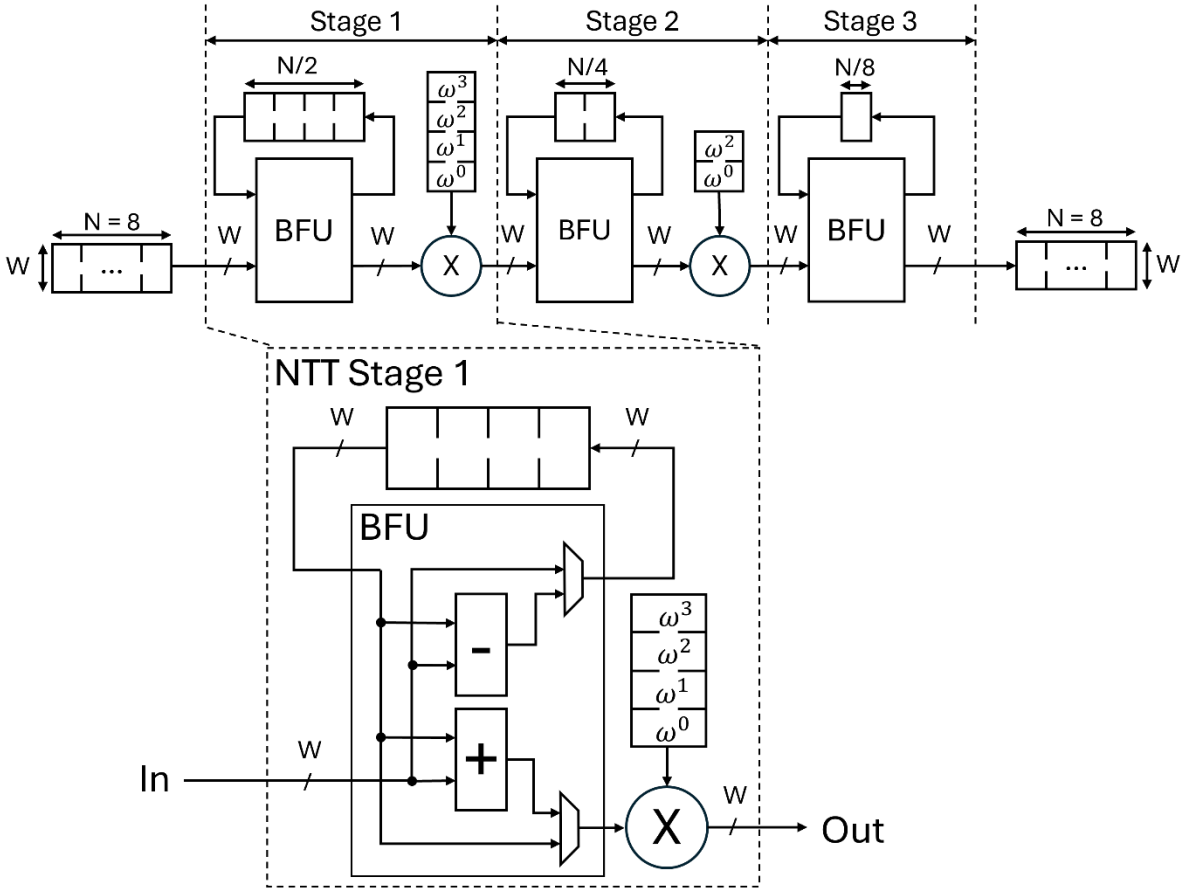
- Number Theoretic Transform (NTT)

$$A_k = \sum_{i=0}^{n-1} a_i \omega^{ik} \pmod q$$



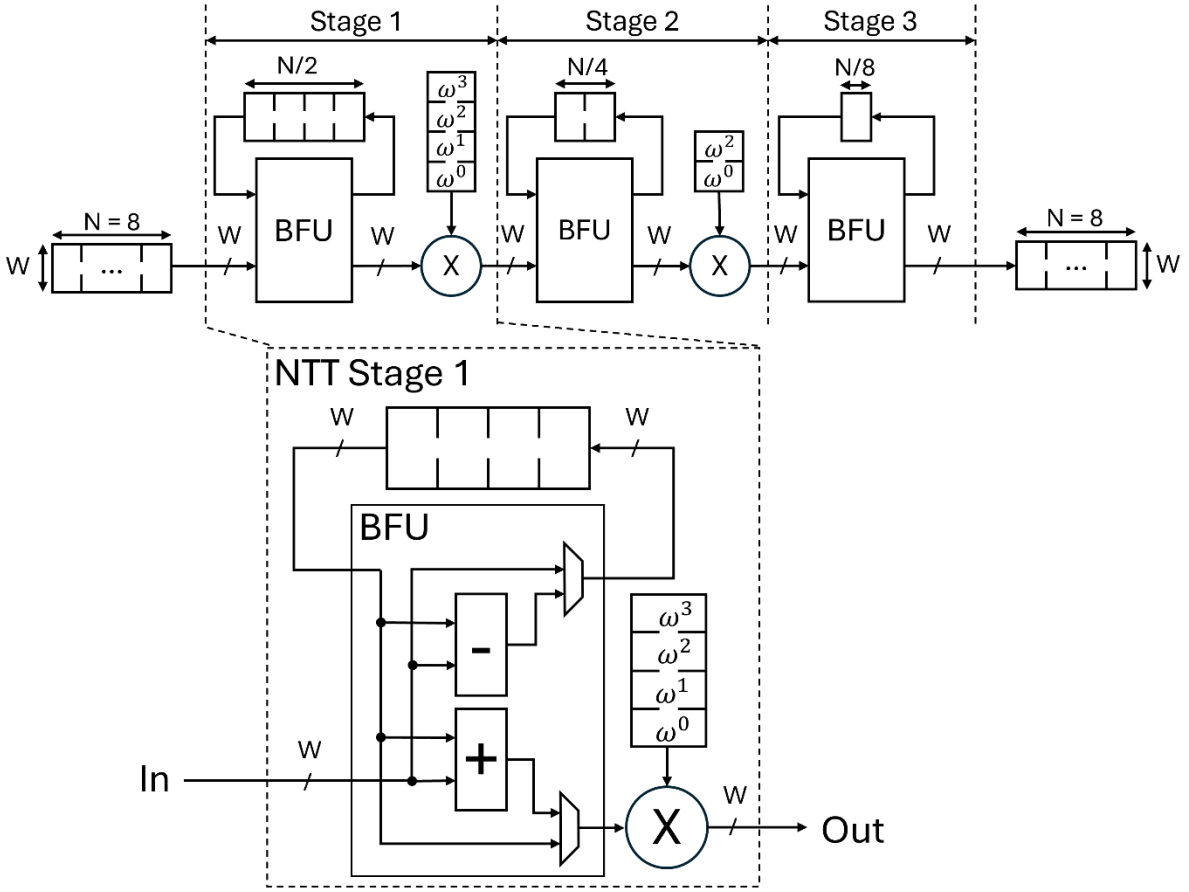
Acceleration Strategies for NTT

Custom NTT Accelerators



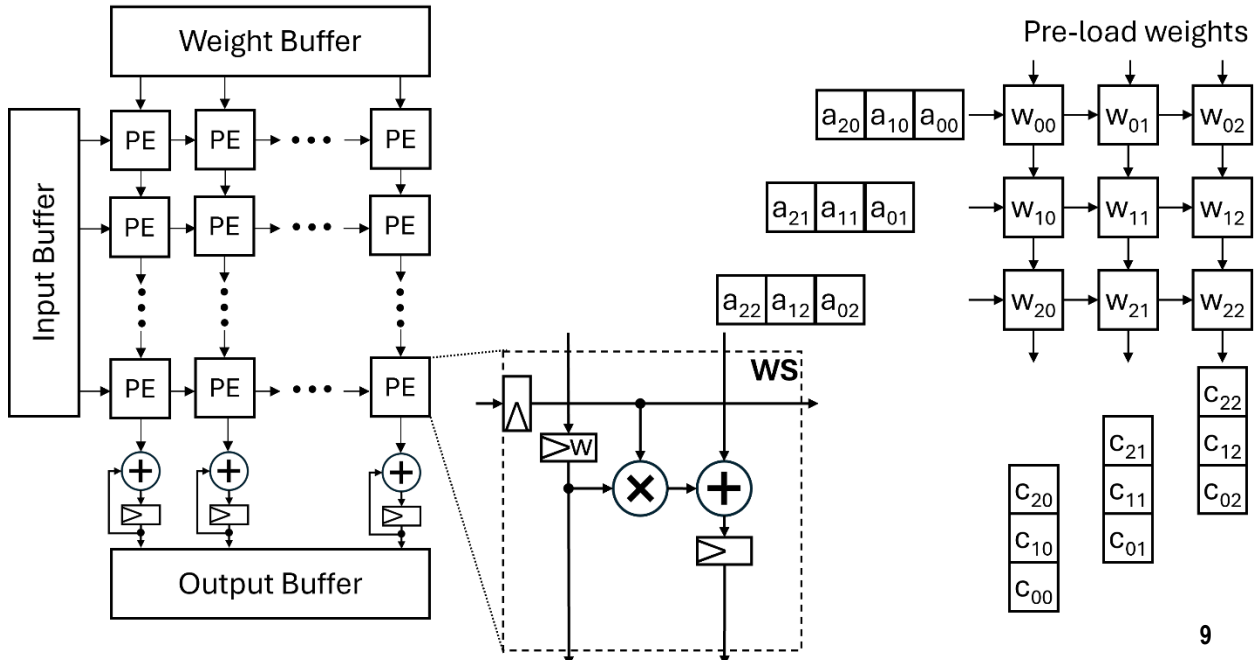
Acceleration Strategies for NTT

Custom NTT Accelerators



Mapping NTT to AI ASICs

$$A_k = \sum_{i=0}^{n-1} a_i \omega^{ik} \Rightarrow \left[A \right] = \left[\quad \omega \quad \right] \left[a \right]$$



Enhancing AI ASICs with High Precision Arithmetic

Challenge

- AI accelerators target 8-bit arithmetic
- Applications require higher bit widths

Enhancing AI ASICs with High Precision Arithmetic

Challenge

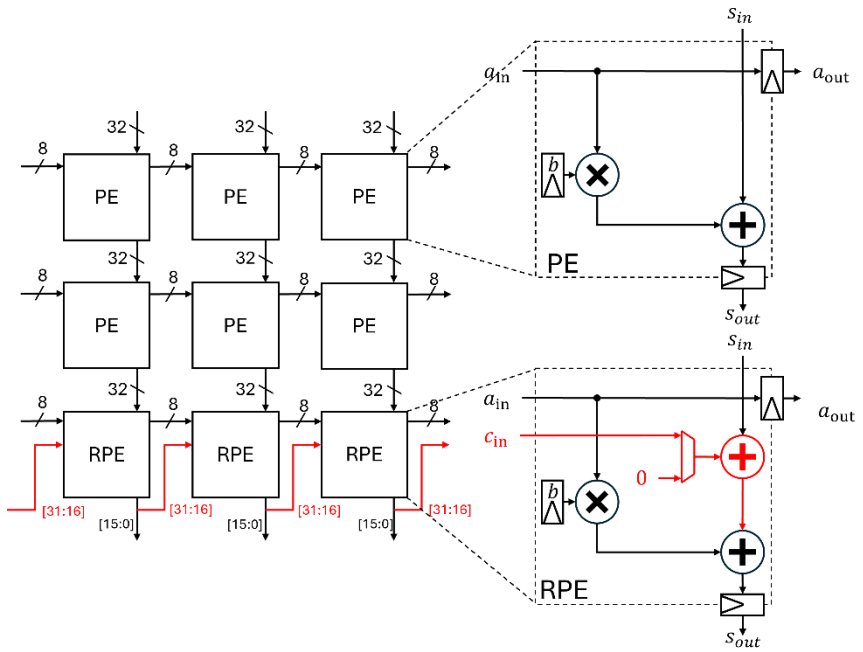
- AI accelerators target 8-bit arithmetic
- Applications require higher bit widths

$$\left[A\right] = \left[\quad \omega \quad\right]\left[a\right]$$

$$\left[A\right] = \left[\quad \omega \quad\right]\left[\!\left[a\right]\!\right]$$

Our Approach

- Introduce lightweight high-precision
- Extend processing elements



Enhancing AI ASICs with High Precision Arithmetic

Challenge

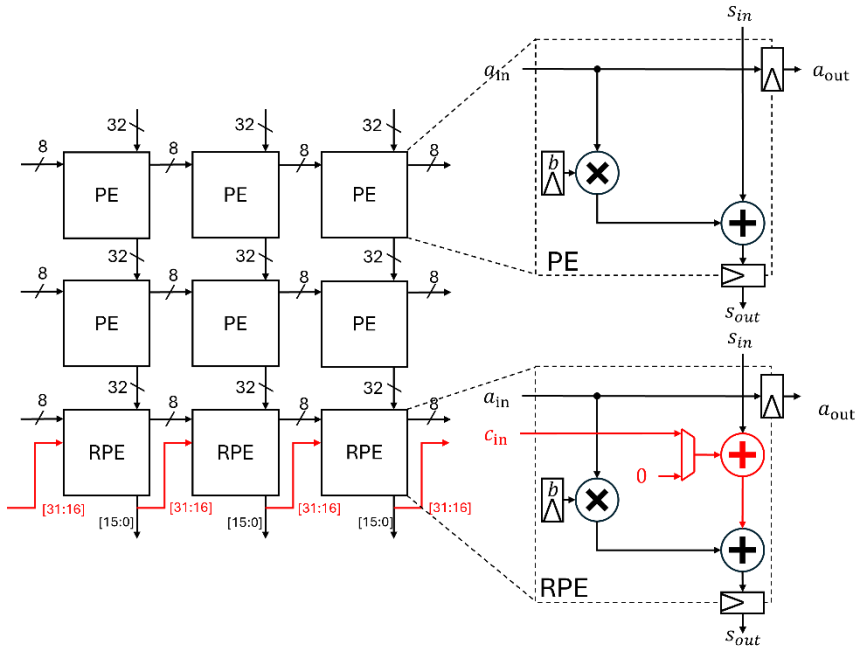
- AI accelerators target 8-bit arithmetic
- Applications require higher bit widths

$$\left[A\right] = \left[\quad \omega \quad\right]\left[a\right]$$

$$\left[A\right] = \left[\quad \omega \quad\right]\left[\!\left[a\right]\!\right]$$

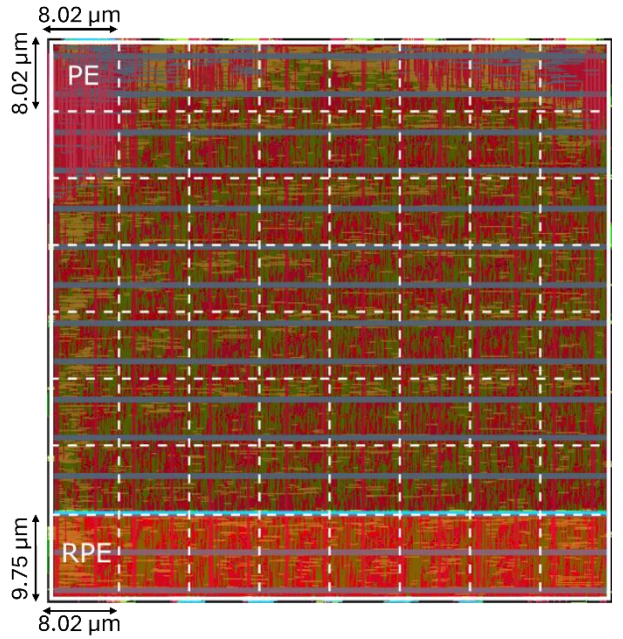
Our Approach

- Introduce lightweight high-precision
- Extend processing elements



Benefits

- Higher performance
- Minimal area and power overhead



Fault Detection for Reliable NTT Execution

- Compute a single check value from the input using a pre-computed vector $\boldsymbol{y}$.
- Independently compute the check value from the NTT output using a constant vector $\boldsymbol{x}$.
- Low-overhead detection with minimal impact on performance and no full re-computation.