IAIK

TU Graz

SCIENCE
PASSION
TECHNOLOGY

OpenNTT

An Automated Toolchain for Compiling
High-Performance NTT Accelerators in FHE

Florian Krieger, Florian Hirner, Ahmet Can Mert, Sujoy Sinha Roy

October 28, 2024

Outline

Table of Contents

Fully Homomorphic Encryption (FHE)

- Computations over encrypted data
- No information is revealed
- Many different schemes

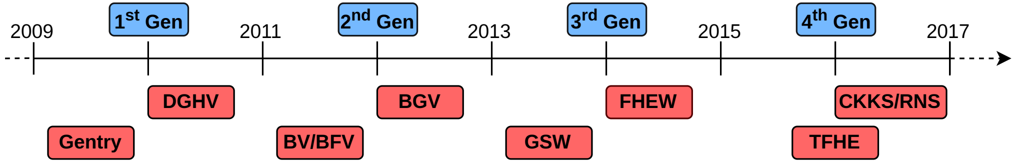
Fully Homomorphic Encryption (FHE)

- Computations over encrypted data
- No information is revealed
- Many different schemes

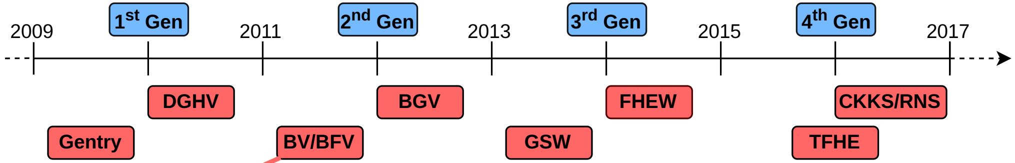
Fully Homomorphic Encryption (FHE)

- Computations over encrypted data
- No information is revealed
- Many different schemes

Different FHE Schemes



Different FHE Schemes

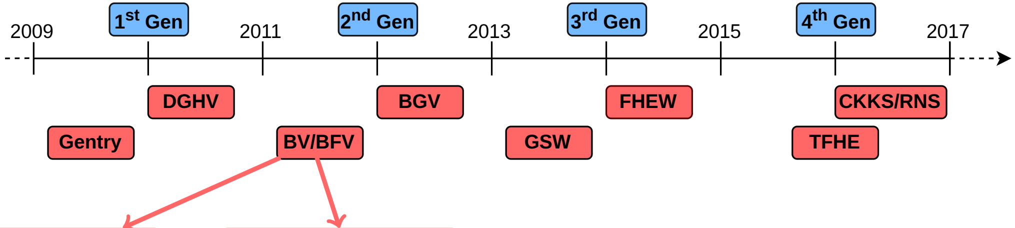


$N = 2^{10}$

$\log q = 32$b

[MOS20]

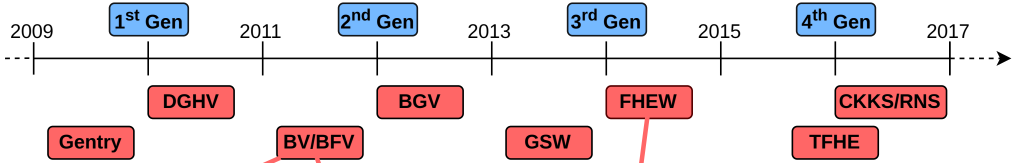
Different FHE Schemes



$N = 2^{10}$
$\log\ q = 32\text{b}$
[MOS20]

$N = 2^{12}$
$\log\ q = 6 \times 32\text{b}$
[SYYZ22]

Different FHE Schemes

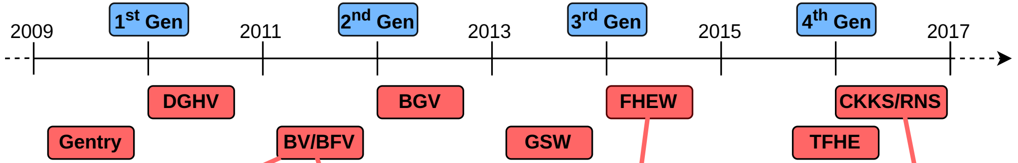


$N = 2^{10}$
$\log\ q = 32\text{b}$
[MOS20]

$N = 2^{12}$
$\log\ q = 6 \times 32\text{b}$
[SYYZ22]

$N = 2^{10}$
$\log\ q = 27\text{b}$
[BBTV23]

Different FHE Schemes



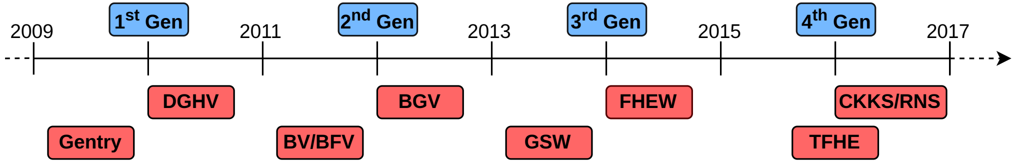
$N = 2^{10}$
$\log q = 32\text{b}$
[MOS20]

$N = 2^{12}$
$\log q = 6 \times 32\text{b}$
[SYYZ22]

$N = 2^{10}$
$\log q = 27\text{b}$
[BBTV23]

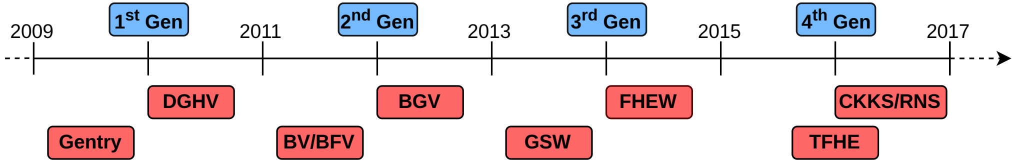
$N = 2^{13}$
$\log q = 6 \times 36\text{b}$
[ACJ24]

Different FHE Schemes



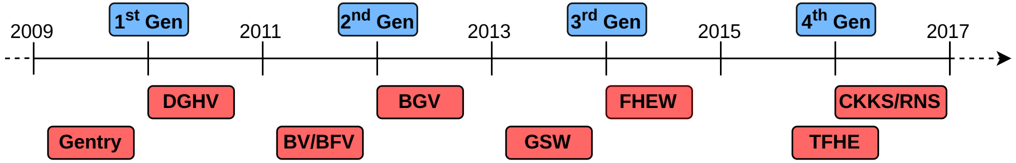
- Parameters also influenced by:
  - Application scenario
  - Bootstrapping support
  - Target platform

Different FHE Schemes



- Parameters also influenced by:
  - Application scenario
  - Bootstrapping support
  - Target platform

Different FHE Schemes



- Parameters also influenced by:
  - Application scenario
  - Bootstrapping support
  - Target platform

FHE: Large Variety in Parameters

- Different schemes $\rightarrow$ different parameters
- Influences building-block level:
  - Polynomial multiplication
  - Number Theoretic Transformation

FHE: Large Variety in Parameters

- Different schemes $\rightarrow$ different parameters
- Influences building-block level:
  - Polynomial multiplication
  - Number Theoretic Transformation

The Number Theoretic Transformation

- Computational bottleneck: NTT
  - ➔ >70% of computation time [KKK+22]
  - ➔ Hardware acceleration

Many configurations: Challenges

? How can we reduce the hardware design effort?

? How can we enhance flexibility?

Hardware design tools!

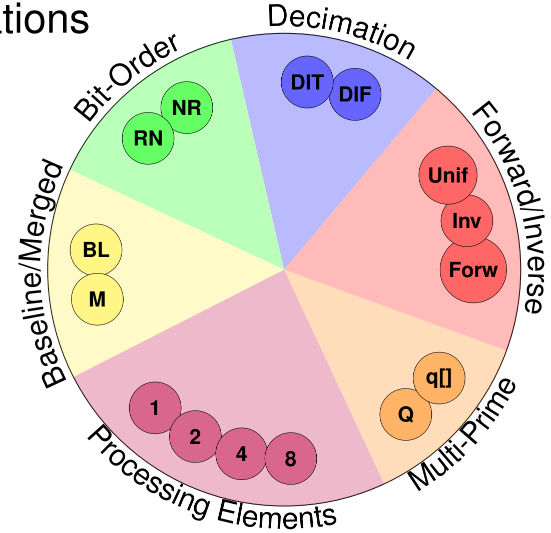
The Number Theoretic Transformation

- Computational bottleneck: NTT
  - → >70% of computation time [KKK+22]
  - → Hardware acceleration

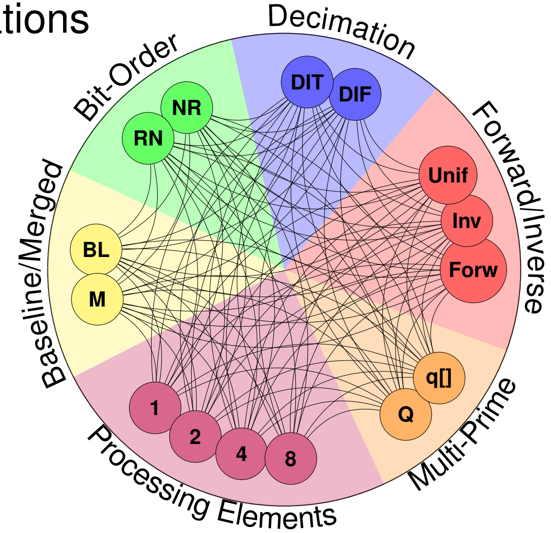
Many configurations: Challenges

- ? How can we reduce the hardware design effort?
- ? How can we enhance flexibility?
- Hardware design tools!

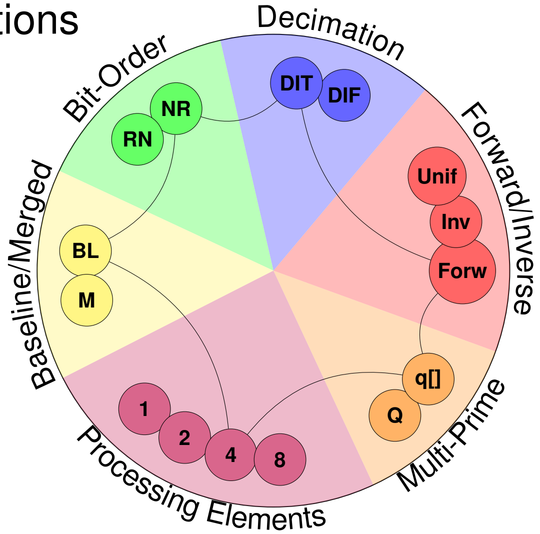
NTT Configurations



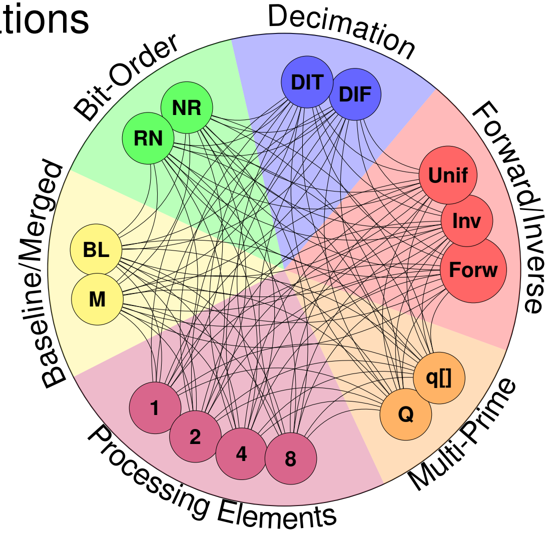
NTT Configurations



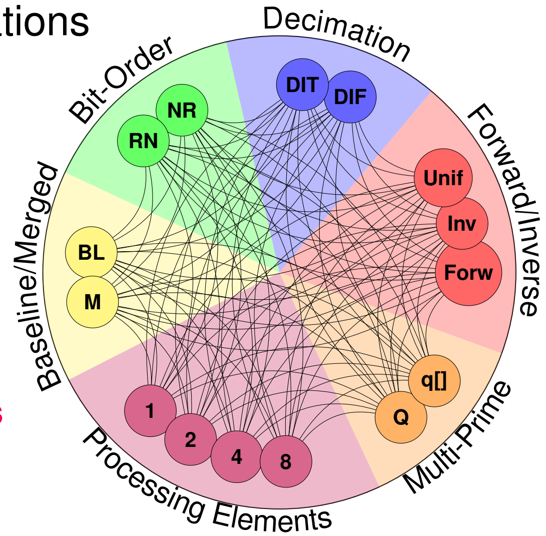
NTT Configurations



NTT Configurations



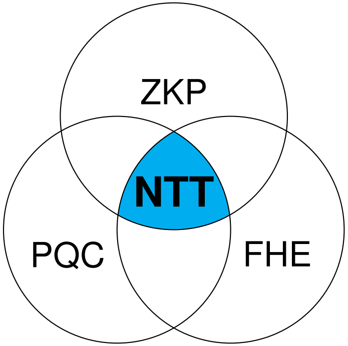
NTT Configurations



NTT design tools are challenging

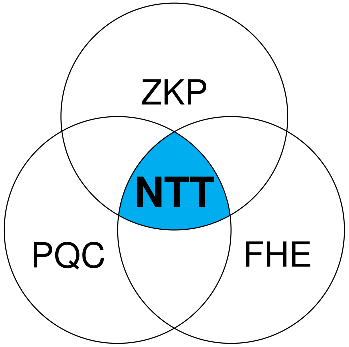
NTT: Versatile Applications

- Not only for FHE
- Post-Quantum Cryptography
- Zero-Knowledge Proof Systems



NTT: Versatile Applications

- Not only for FHE
- Post-Quantum Cryptography
- Zero-Knowledge Proof Systems



NTT: Versatile Applications

- Not only for FHE
- Post-Quantum Cryptography
- Zero-Knowledge Proof Systems

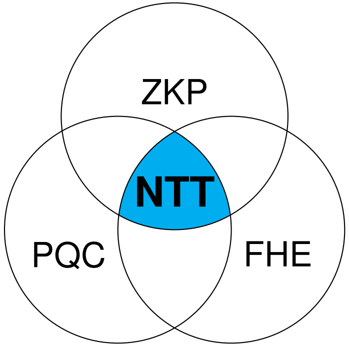
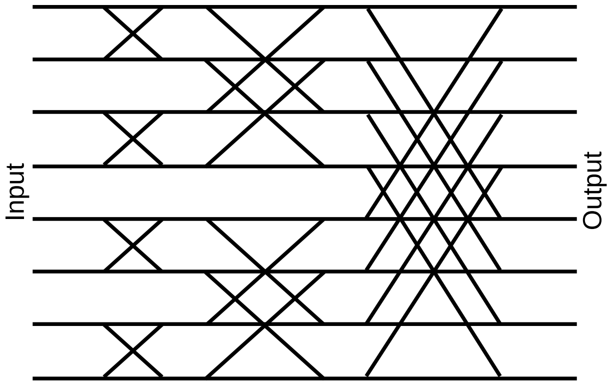


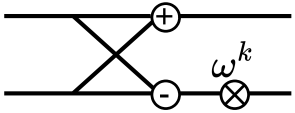
Table of Contents

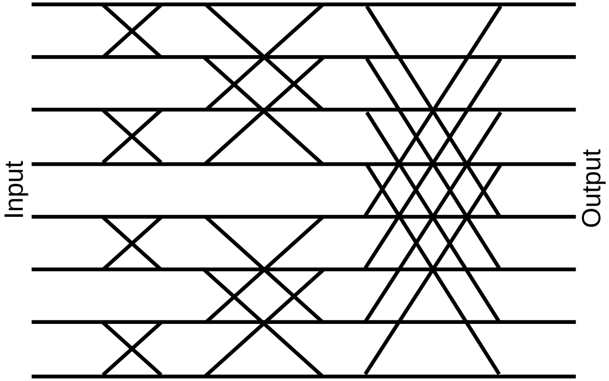
Different NTT Approaches



Different NTT Approaches

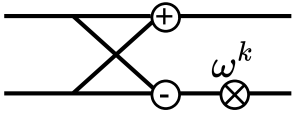
- Butterfly operation





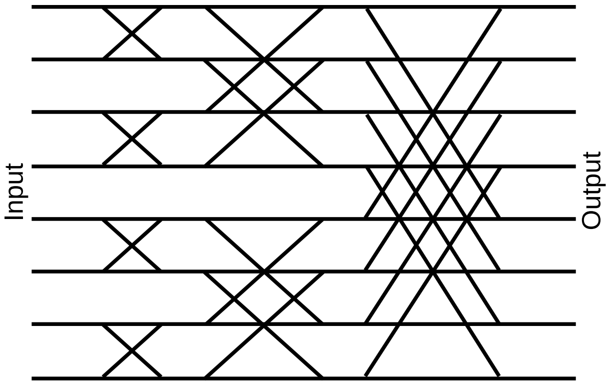
Different NTT Approaches

- Butterfly operation



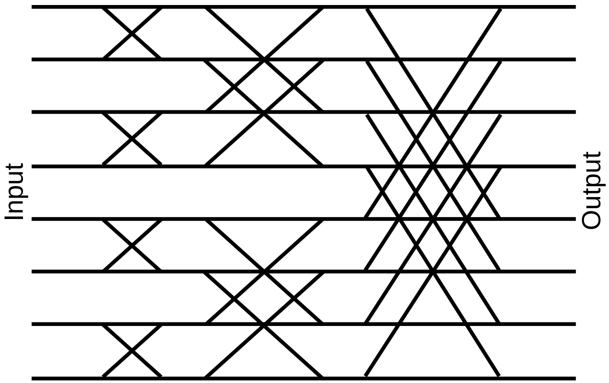
- Multiple processing elements (PE)

  ➔ Higher performance



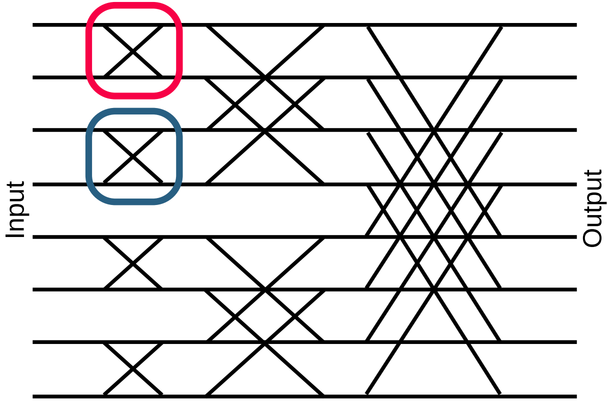
Different NTT Approaches

- Iterative NTT
  - Multi-PE
  - High-radix PE



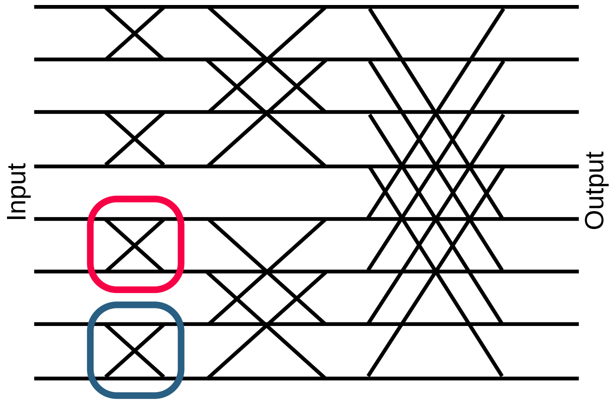
Different NTT Approaches

- Iterative NTT
    - Multi-PE
    - High-radix PE



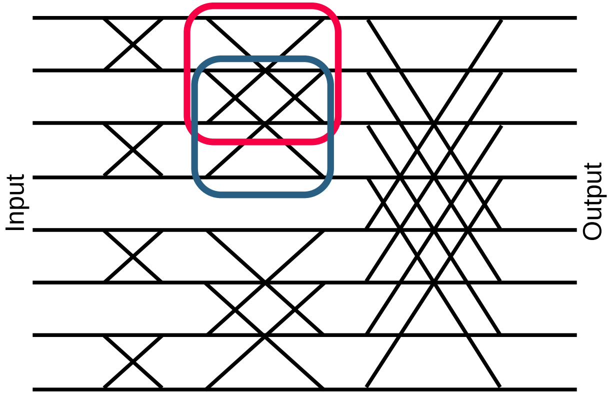
Different NTT Approaches

- Iterative NTT
  - Multi-PE
  - High-radix PE



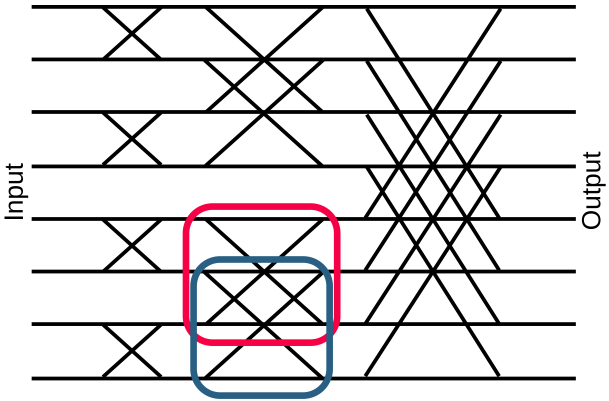
Different NTT Approaches

- Iterative NTT
  - Multi-PE
  - High-radix PE



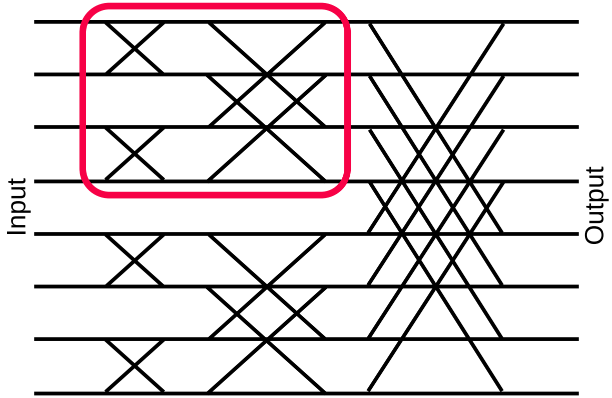
Different NTT Approaches

- Iterative NTT
  - Multi-PE
  - High-radix PE



Different NTT Approaches

- Iterative NTT
  - Multi-PE
  - High-radix PE

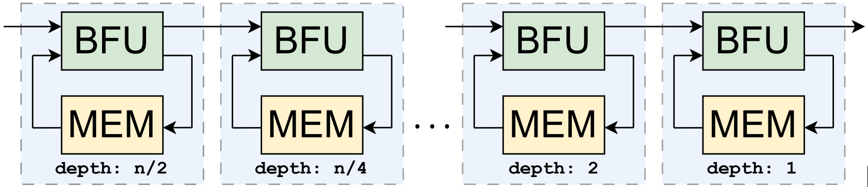


Different NTT Approaches

- Iterative NTT
  - Multi-PE
  - High-radix PE
- Pipelined NTT
  - Single-path delay feedback (SDF)
  - Multi-path delay commutator (MDC)

Different NTT Approaches

- Iterative NTT
  - Multi-PE
  - High-radix PE
- Pipelined NTT
  - Single-path delay feedback (SDF)
  - Multi-path delay commutator (MDC)



[HMR24]

Different NTT Approaches

- Iterative NTT
  - **Multi-PE**
  - High-radix PE
- Pipelined NTT
  - Single-path delay feedback (SDF)
  - Multi-path delay commutator (MDC)

Iterative Multi-PE NTT

- Twiddle factor management (TF)

Stored TF

- ⊕ Less logic
- ⊖ Large memories
- → Simpler to implement

On-the-fly generated TF

- ⊕ Less memory
- ⊖ More logic
- → Better for FHE (Multiple primes, large polys)

Iterative Multi-PE NTT

- Twiddle factor management (TF)

Stored TF	On-the-fly generated TF
➕ Less logic	➕ Less memory
➖ Large memories	➖ More logic
→ Simpler to implement	→ Better for FHE (Multiple primes, large polys)

Iterative Multi-PE NTT

- Twiddle factor management (TF)

Stored TF	On-the-fly generated TF
⊕ Less logic	⊕ Less memory
⊖ Large memories	⊖ More logic
→ Simpler to implement	→ Better for FHE (Multiple primes, large polys)

Iterative Multi-PE NTT

- On-the-fly twiddle factor generation
  - ➔ Linear twiddle factor order: $\omega^1, \omega^2, \omega^3, \ldots$
- Conflict-free memory accesses
  - ➔ Dedicated execution flow

OpenNTT

- Combines and generically
- For all relevant configurations

Iterative Multi-PE NTT

- On-the-fly twiddle factor generation
  - Linear twiddle factor order: $\omega^1, \omega^2, \omega^3, \ldots$
- Conflict-free memory accesses
  - Dedicated execution flow

OpenNTT

- Combines and generically
- For all relevant configurations

Iterative Multi-PE NTT

- On-the-fly twiddle factor generation
  - → Linear twiddle factor order: $\omega^1, \omega^2, \omega^3, \ldots$
- Conflict-free memory accesses
  - → Dedicated execution flow

OpenNTT

- Combines and generically
- For all relevant configurations

Table of Contents

Generic Processing Orders

- Derive universal memory access constraints
  - For each stage
  - For every NTT type
  - For different number of PE
- Combine with efficient twiddle factor order
  - Ensure *somewhat* linear generation order

Generic Processing Orders

- Derive universal memory access constraints
  - For each stage
  - For every NTT type
  - For different number of PE
- Combine with efficient twiddle factor order
  - Ensure *somewhat* linear generation order

Generic Processing Orders

- Derive universal memory access constraints
  - For each stage
  - For every NTT type
  - For different number of PE
- Combine with efficient twiddle factor order
  - Ensure *somewhat* linear generation order

Somewhat Linear Generation Order

- Some NTT configurations need somewhat linear generation:

  ➔ Due to memory constraints

  $\omega^0, \omega^8, \omega^1, \omega^9, \omega^2, \omega^{10}, \ldots$

  $\omega^i, \omega^j, \omega^{i+c}, \omega^{j+c}, \ldots$

- Changes across stages

Somewhat Linear Generation Order

- Some NTT configurations need somewhat linear generation:
  - ➔ Due to memory constraints

$$\omega^0, \omega^8, \omega^1, \omega^9, \omega^2, \omega^{10}, \ldots$$

$$\omega^i, \omega^j, \omega^{i+c}, \omega^{j+c}, \ldots$$

- Changes across stages

Somewhat Linear Generation Order

- Some NTT configurations need somewhat linear generation:

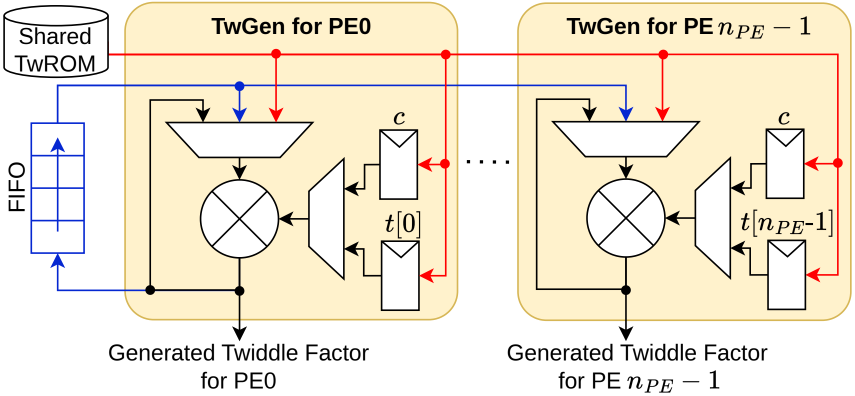
  ➔ Due to memory constraints

  $\omega^0, \omega^8, \omega^1, \omega^9, \omega^2, \omega^{10}, \ldots$

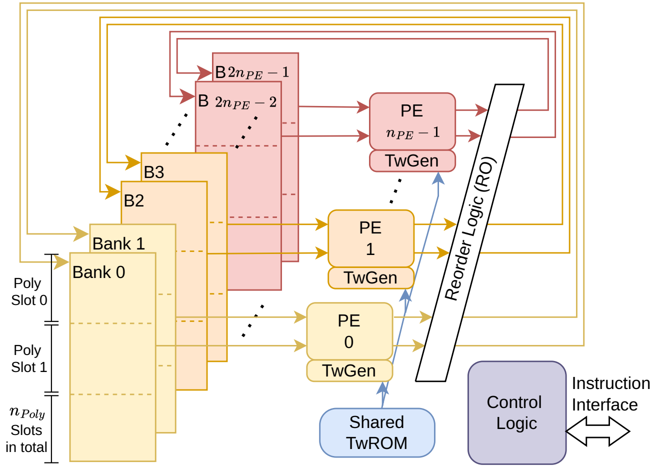
  $\omega^i, \omega^j, \omega^{i+c}, \omega^{j+c}, \ldots$

- Changes across stages

Efficient TF Generation Module

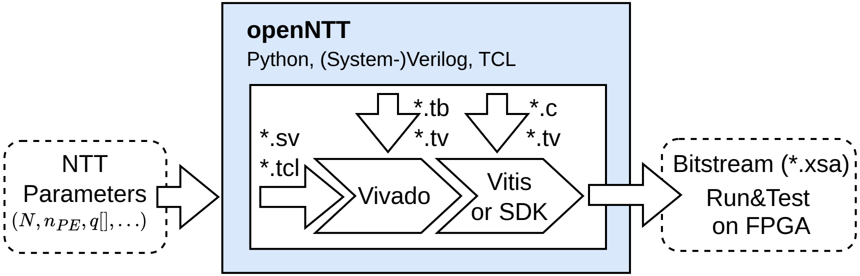

Shared TwROM
TwGen for PE0
TwGen for PE $n_{PE} - 1$
FIFO
c
$t[0]$
c
$t[n_{PE}\text{-}1]$
....
Generated Twiddle Factor for PE0
Generated Twiddle Factor for PE $n_{PE} - 1$

Flexible Hardware Architecture



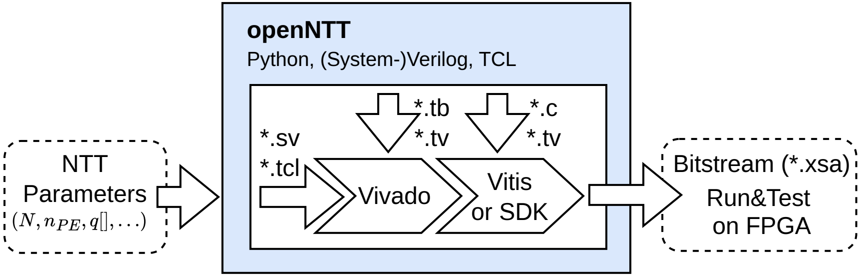
The OpenNTT Toolchain

- Overtakes all steps
- Provides testing functionality



The OpenNTT Toolchain

- Overtakes all steps
- Provides testing functionality



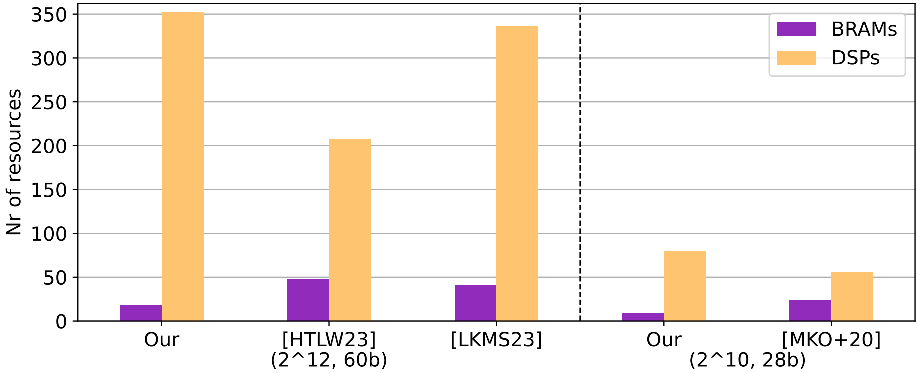
The OpenNTT Toolchain

- Supported parameters:
  - Polynomial size
  - Prime size
  - Number of primes
  - Baseline or NWC NTT
  - Decimation method
  - Coefficient orders
  - Forward/inverse/unified NTT
  - Number of PE
  - NTT only or with arithmetic
  - Memory depth

Table of Contents

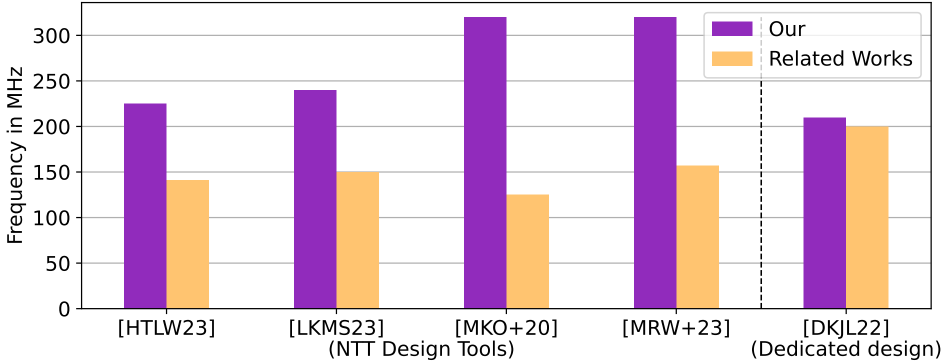
Memory vs. Logic Consumption

- Twiddle factor generation trades memory for logic



Achieved Frequency

- Benchmarked on same FPGA



Comparison Benchmarks

- NTT generation tools:
  - ➔ Prior tools use stored twiddle factors
  - 1.3× to 2.7× speedup
  - 1.8× improved ATP
- Dedicated designs:
  - Comparable performance

Comparison Benchmarks

- NTT generation tools:
  - ➔ Prior tools use stored twiddle factors
  - 1.3× to 2.7× speedup
  - 1.8× improved ATP
- Dedicated designs:
  - Comparable performance

Comparison Benchmarks

- NTT generation tools:
  - ➔ Prior tools use stored twiddle factors
  - 1.3× to 2.7× speedup
  - 1.8× improved ATP
- Dedicated designs:
  - Comparable performance

Comparison Benchmarks

- NTT generation tools:
  - ➔ Prior tools use stored twiddle factors
  - 1.3× to 2.7× speedup
  - 1.8× improved ATP
- Dedicated designs:
  - Comparable performance

Table of Contents

Conclusion and Takeaways

- Generic tool for NTT hardware accelerator design
- Open-sourced our tool
  - ➔ Contribute to faster NTT design times
  - ➔ Supports research in the field
  - ➔ Relevant for industrial and academic domains

Conclusion and Takeaways

- Generic tool for NTT hardware accelerator design
- Open-sourced our tool
  - ➔ Contribute to faster NTT design times
  - ➔ Supports research in the field
  - ➔ Relevant for industrial and academic domains

SCIENCE
PASSION
TECHNOLOGY

OpenNTT

An Automated Toolchain for Compiling High-Performance NTT Accelerators in FHE

Florian Krieger, Florian Hirner, Ahmet Can Mert, Sujoy Sinha Roy

October 28, 2024

References

[MOS20] A. C. Mert *et al.*, "Design and Implementation of Encryption/Decryption Architectures for BFV Homomorphic Encryption Scheme," in IEEE TVLSI, 2020

[SYYZ22] Y. Su *et al.*, "ReMCA: A Reconfigurable Multi-Core Architecture for Full RNS Variant of BFV Homomorphic Evaluation," in TCAS-I, 2022

[BBTV23] J. Bertels *et al.*, "Hardware Acceleration of FHEW," in DDECS, 2023

[ACJ24] R. Agrawal *et al.*, "HEAP: A Fully Homomorphic Encryption Accelerator with Parallelized Bootstrapping," in ISCA, 2024

[HMR24] F. Hirner *et al.*, "Proteus: A Pipelined NTT Architecture Generator," in TVLSI, 2024

[HTLW23] Xiao Hu *et al.*, "Ac-pm: an area-efficient and configurable polynomial multiplier for lattice based cryptography," in TCAS-I, 2023

[LKMS23] Si-Huang Liu *et al.*, "An area-efficient, conflict-free, and configurable architecture for accelerating ntt/intt," in TVLSI, 2023

[MKO+20] A. C. Mert *et al.*, "A flexible and scalable ntt hardware: applications from homomorphically encrypted deep learning to post-quantum cryptography," in DATE, 2020

[MRW+23] J. Mu *et al.*, "Scalable and conflict-free ntt hardware accelerator design: methodology, proof, and implementation," in TCAD, 2023

[DKJL22] P. Duong-Ngoc *et al.*, "Area-efficient number theoretic transform architecture for homomorphic encryption," in TCAS-I, 2022

[KKK+22] S. Kim *et al.*, "BTS: an accelerator for bootstrappable fully homomorphic encryption," in ISCA, 2022