# Presto: A Unified RISC-V-Compatible SoC for Multi-Scheme FHE Acceleration over Module Lattice

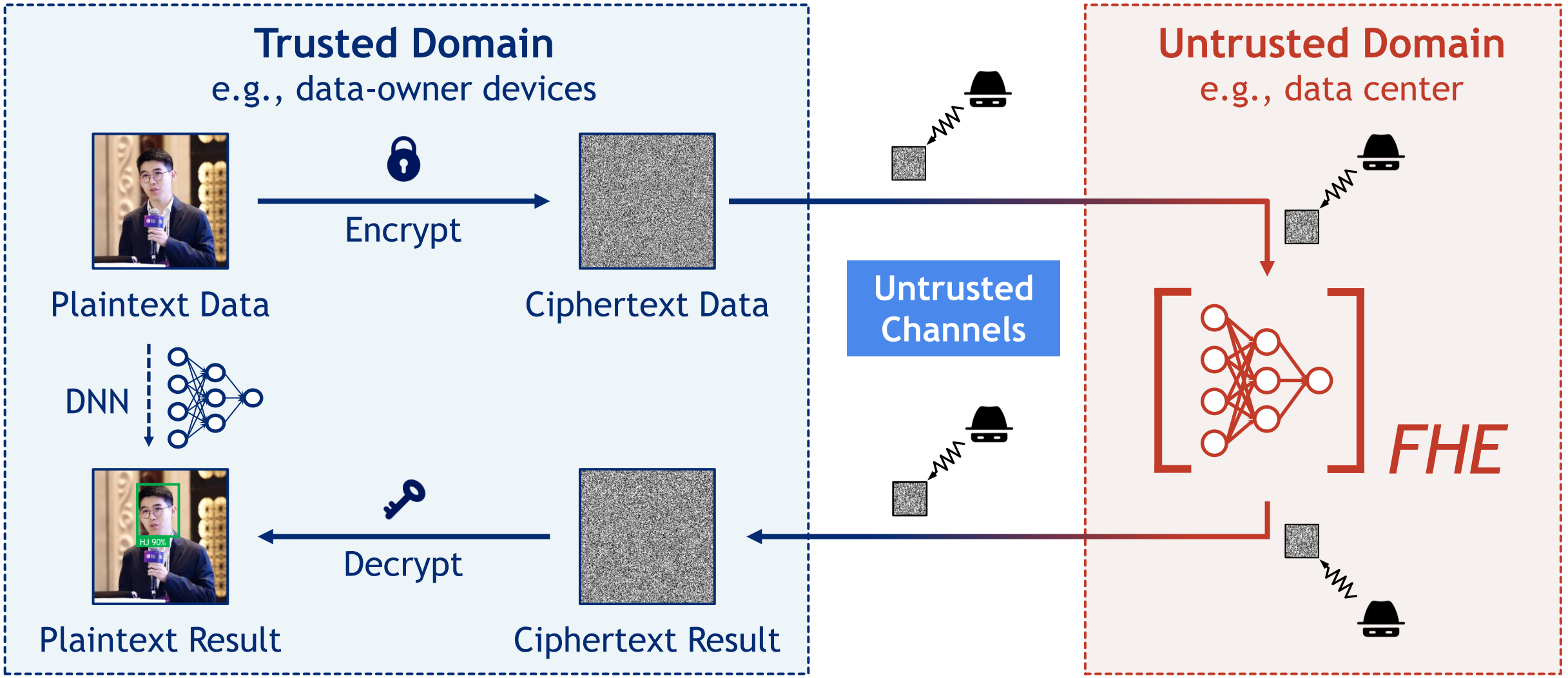
Luchang Lei[1], Yu Duan[1,3*], Cheng Peng[1,3*], Yongqing Zhu[2,3*], Gangfeng Du[3], Zhenyu Guan[2], Huazhong Yang[1], Yongpan Liu[1], Zhen Gu[3], Song Bian[2], Hongyang Jia[1]

[1] Tsinghua University, Beijing, China
[2] Beihang University, Beijing, China
[3] InsightOne Tech Co., Ltd. Shenzhen, China

# Outline

- **Introduction on fully homomorphic encryption (FHE)**
- **Enabling scalable MS-FHE with module-learning-with-error (MLWE)**
  - RISC-V vector-extension-style instruction-set architecture (ISA)
  - Programming model for multi-scheme FHE (MS-FHE)
- **Programmable and scalable FHE accelerator**
  - Architecture overview
  - Microarchitectural design for efficient communication
- **Prototype and demonstrations**
- **Summary**

# How Fully Homomorphic Encryption Works



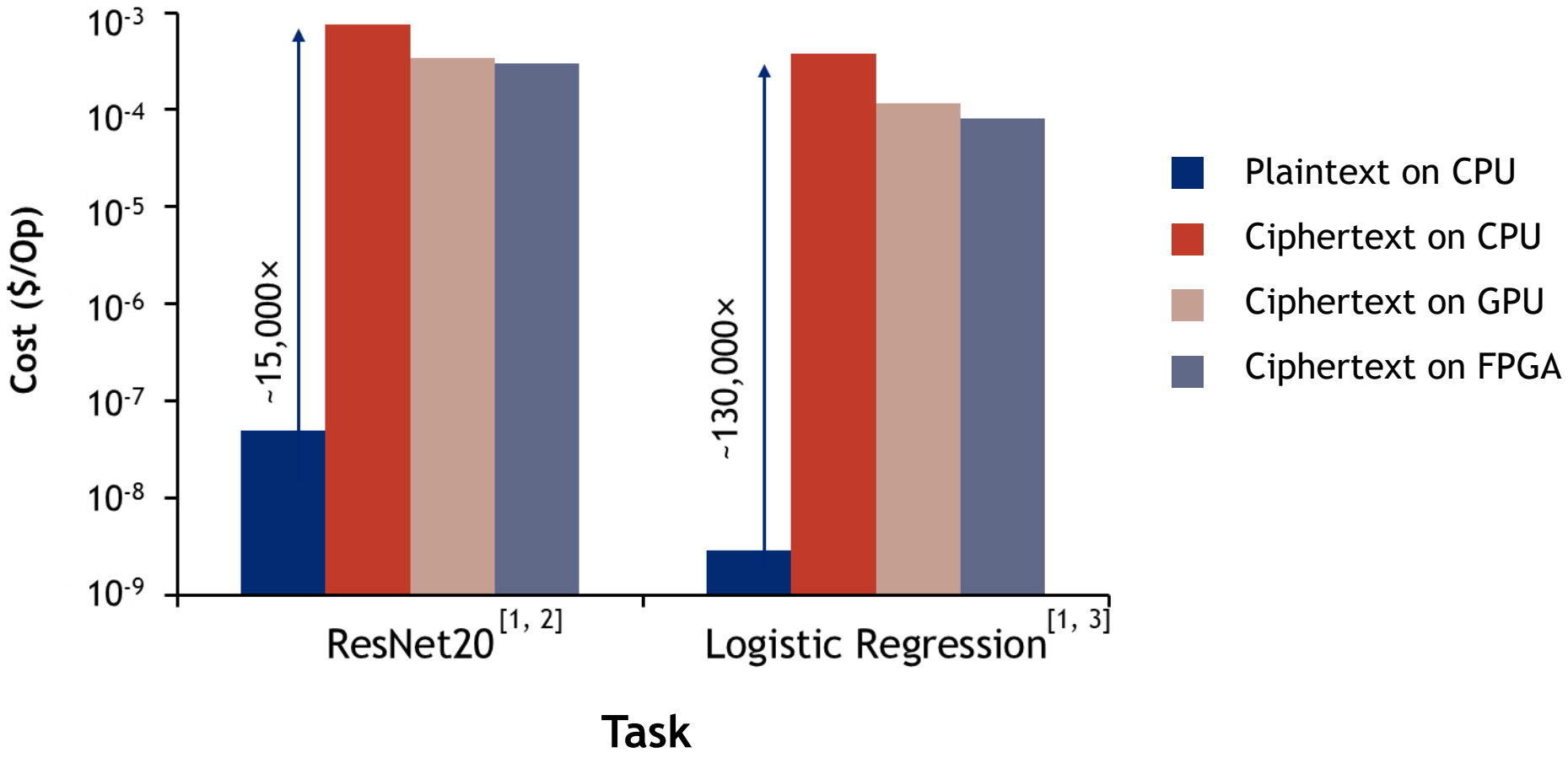

**FHE makes private data available (for computing) but invisible**

# Application of FHE


**Guaranteed security** is welcomed everywhere, however…

# Need of Aggressive Acceleration for FHE

- CPU/GPU: High costs of computation

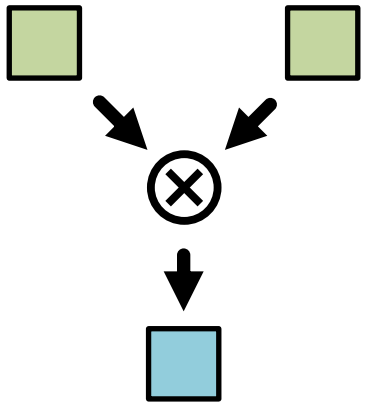



[1] Y.Yang, HPCA'23. [2] D. Kim, IEEE Access. [3] W. Jung, IACR'21.

# Ciphertext and Compute Inflation in Lattice-FHE

## Plaintext

floating point: **32b**

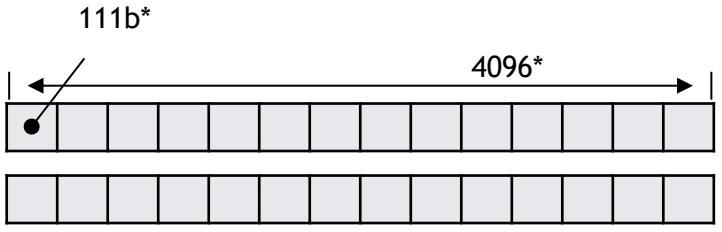


**1 floating point multiply**

Encrypt

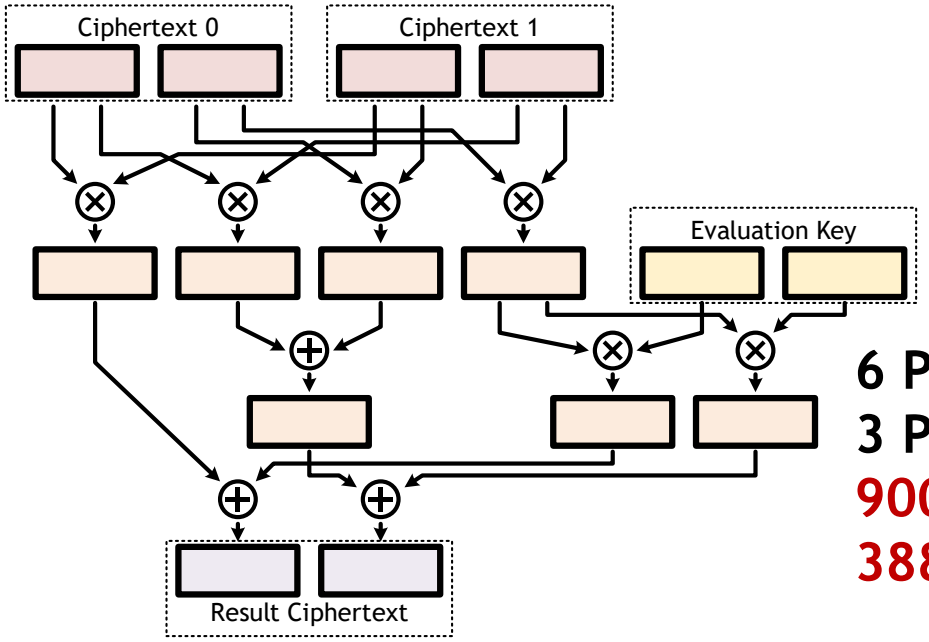
FHE

## FHE Ciphertext



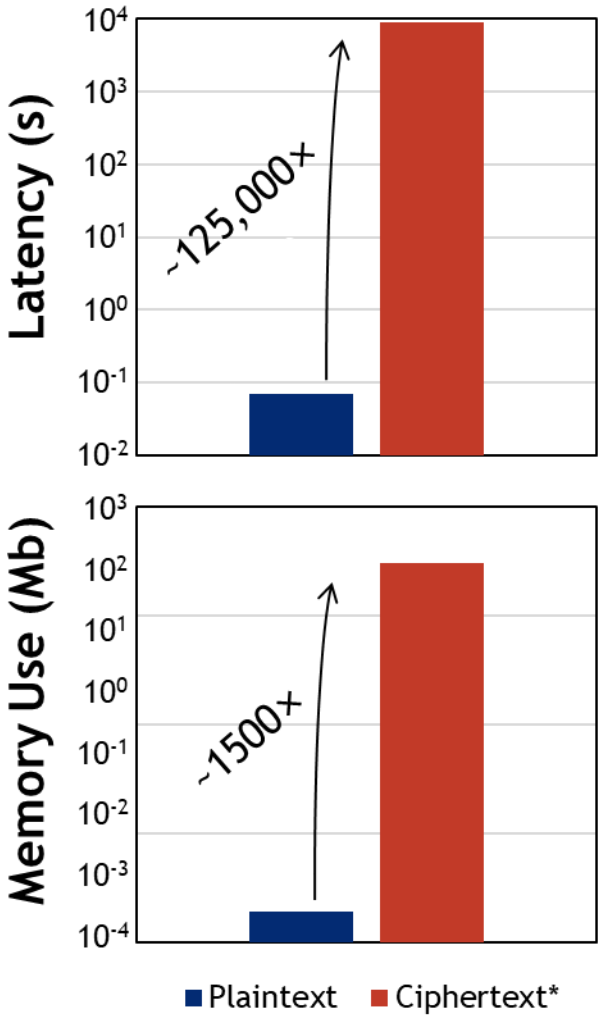

**RLWE representation:**
**2 Polynomials, 111kB***

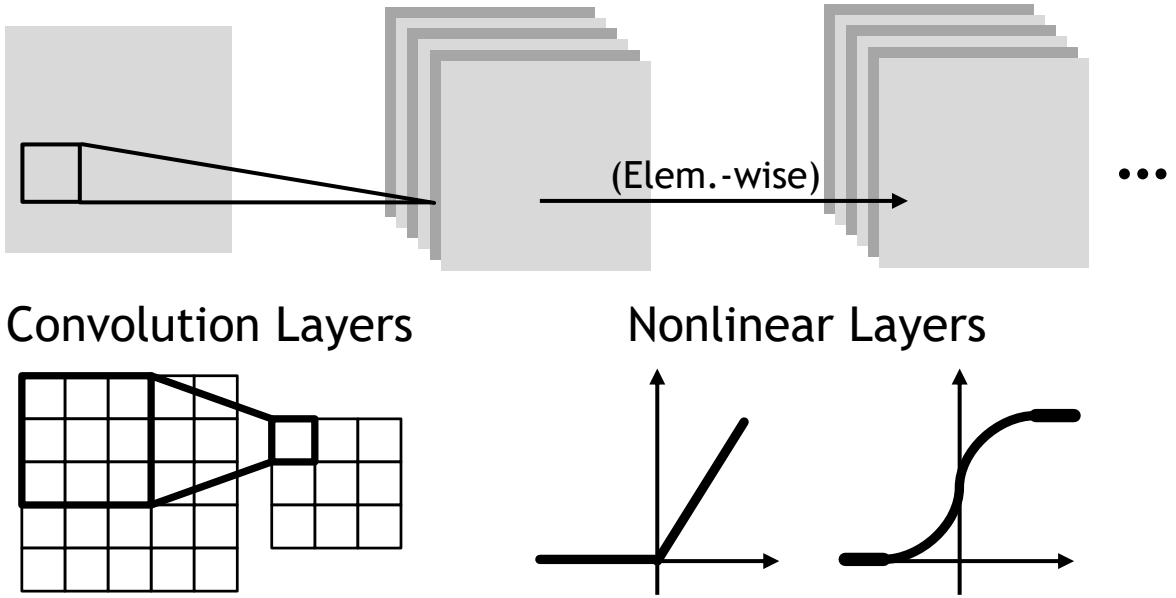
*Typical FHE parameter set in CKKS




**6 Polynomial multiply**
**3 Polynomial add**
**900k×** more computation
**388.5 kB** intermediate data

# Neural-Network Computation in FHE




**Plaintext Operation**

MAC Arithmetic

Boolean Logic

**Typical FHE Scheme**

CKKS Scheme

TFHE Scheme

**Different operations require various FHE schemes**

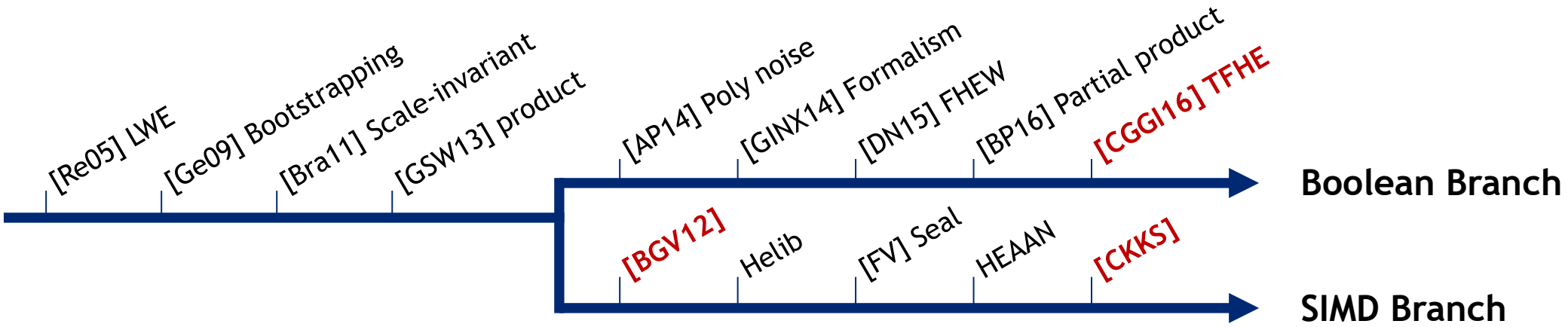
*https://github.com/snu-ccl/FHE-MP-CNN

# Outline

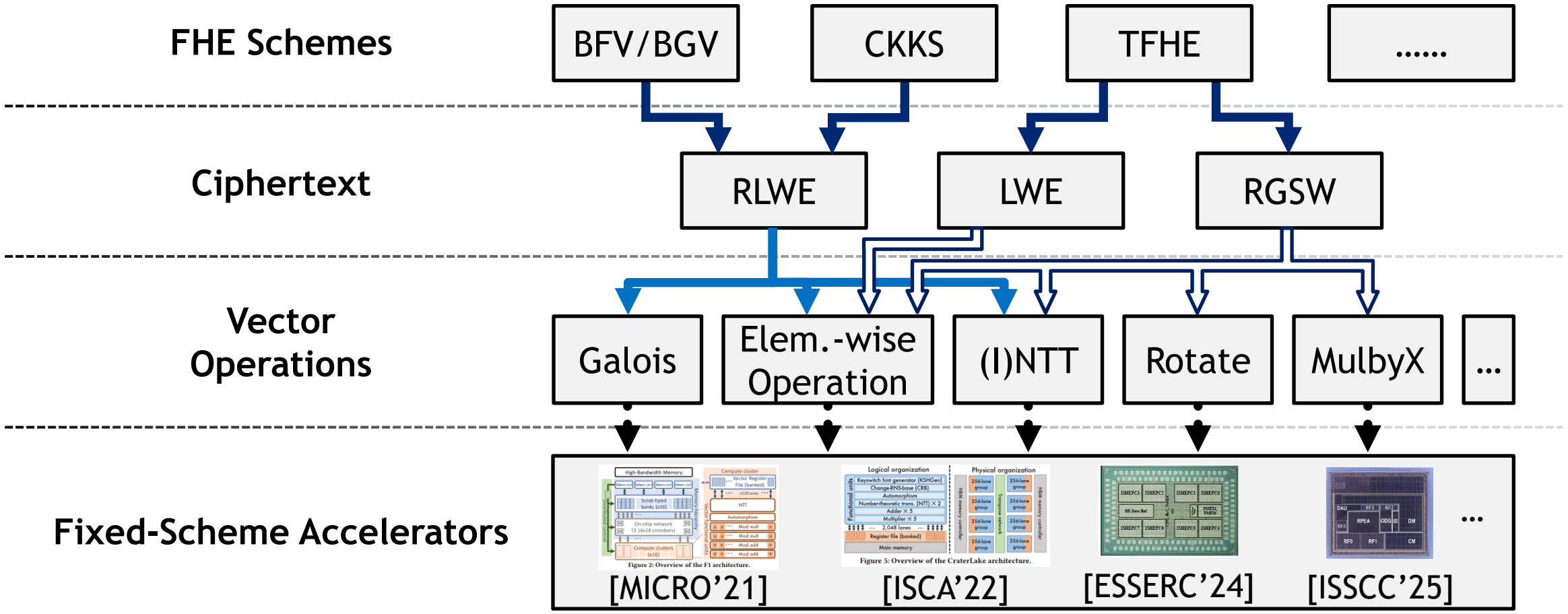
- Introduction on fully homomorphic encryption (FHE)
- **Enabling scalable MS-FHE with module-learning-with-error (MLWE)**
  - RISC-V vector-extension-style instruction-set architecture (ISA)
  - Programming model for multi-scheme FHE (MS-FHE)
- Programmable and scalable FHE accelerator
  - Architecture overview
  - Microarchitectural design for efficient communication
- Prototype and demonstrations
- Summary

# Various FHE Schemes

| Scheme | Plaintext | Typical Application |
|---|---|---|
| BFV/BGV | Integer Vec. | Encrypted Password Check,<br>Caller ID Lookup, etc. |
| CKKS | Fixed-Point Vec. | Encrypted Image Processing,<br>Privacy-Preserving Machine Learning, etc. |
| TFHE | Bit | Transciphering, Key Management, etc. |



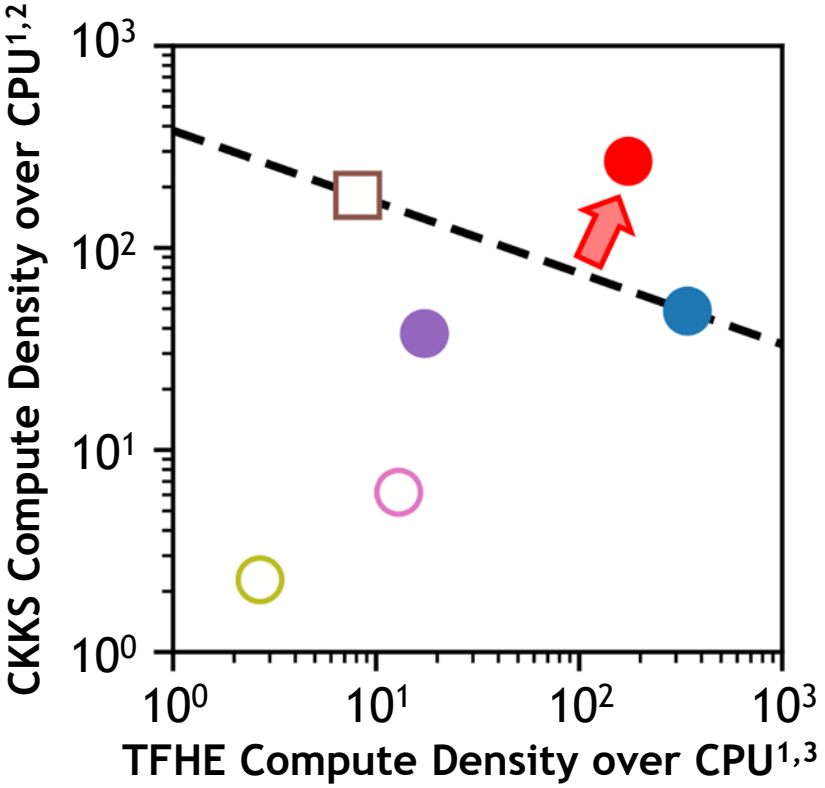
# FHE Schemes and Accelerators



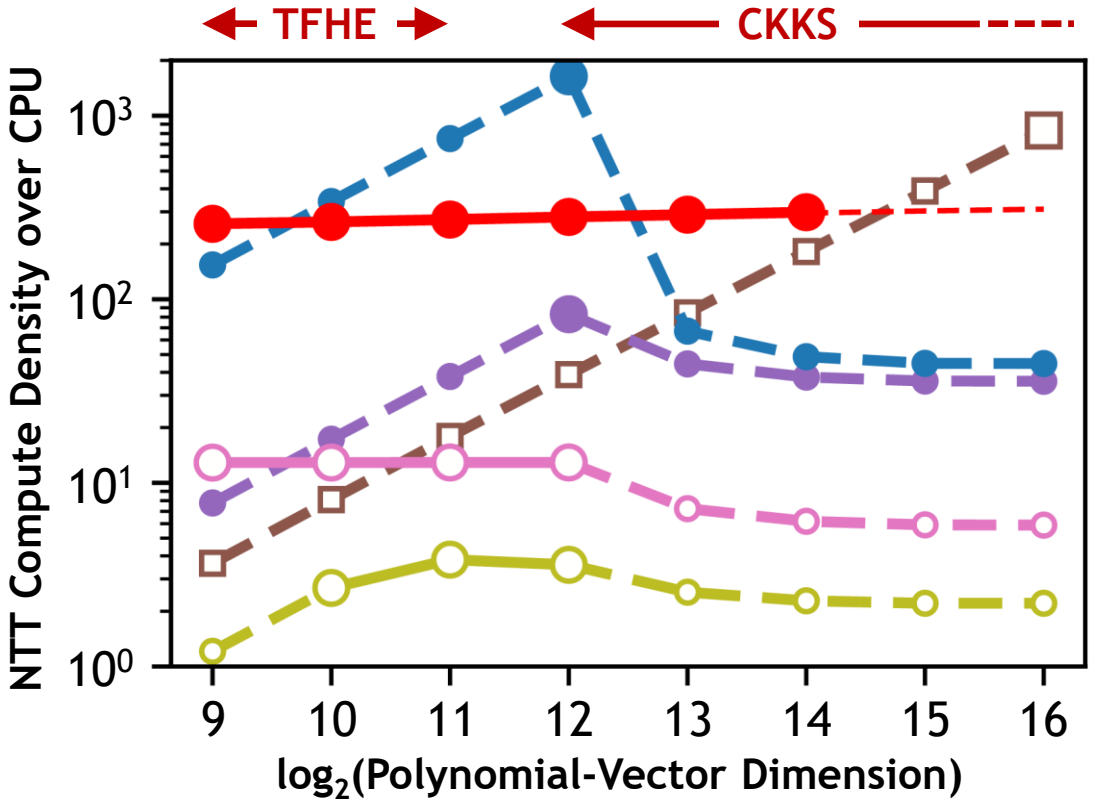

**Previous accelerators only optimize for a limited subset of vector operations**

# Challenges of Supporting Multi-Scheme FHE

## Acceleration vs. FHE schemes




## Acceleration vs. polynomial-vector dimensions

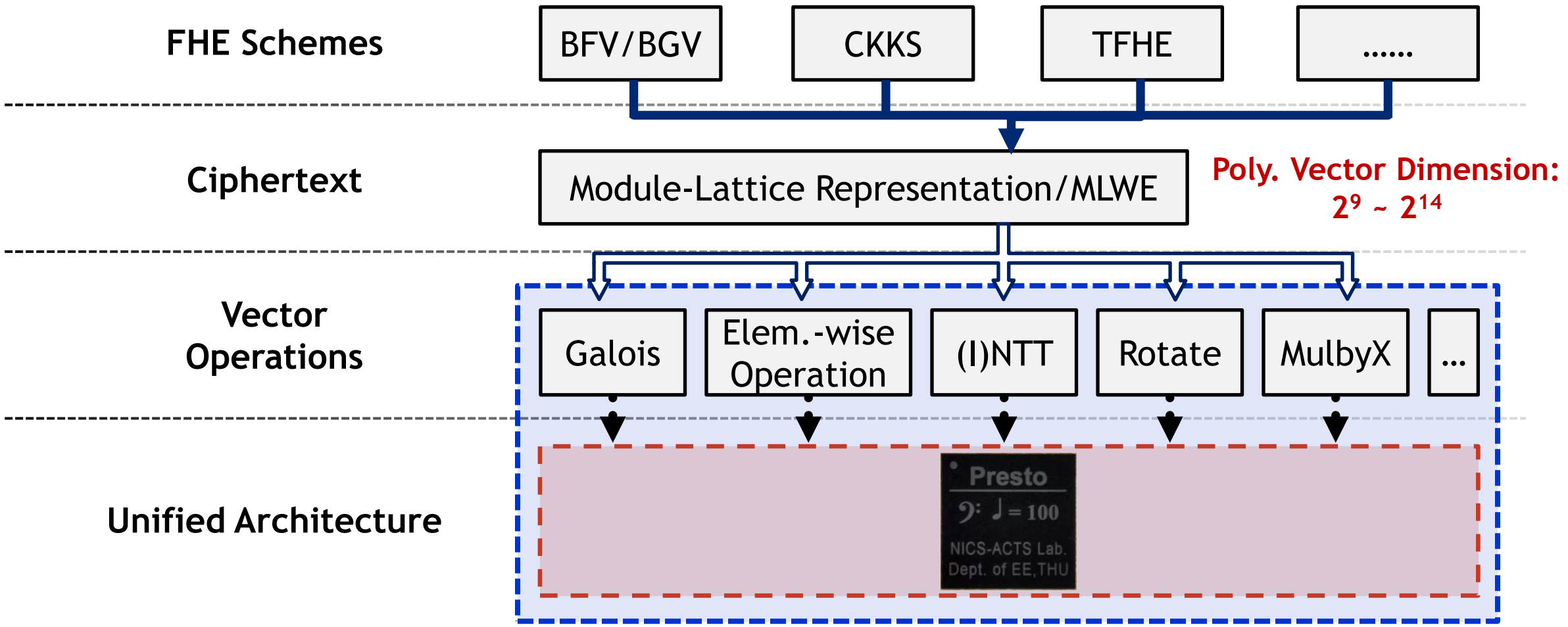



ISCA'22 ISSCC'25 HPCA'24 Natively Reported Dim. Tapeout 14nm

ESSCIRC'23 MICRO'23 **This Work** Estimated Performance Simulation 28nm

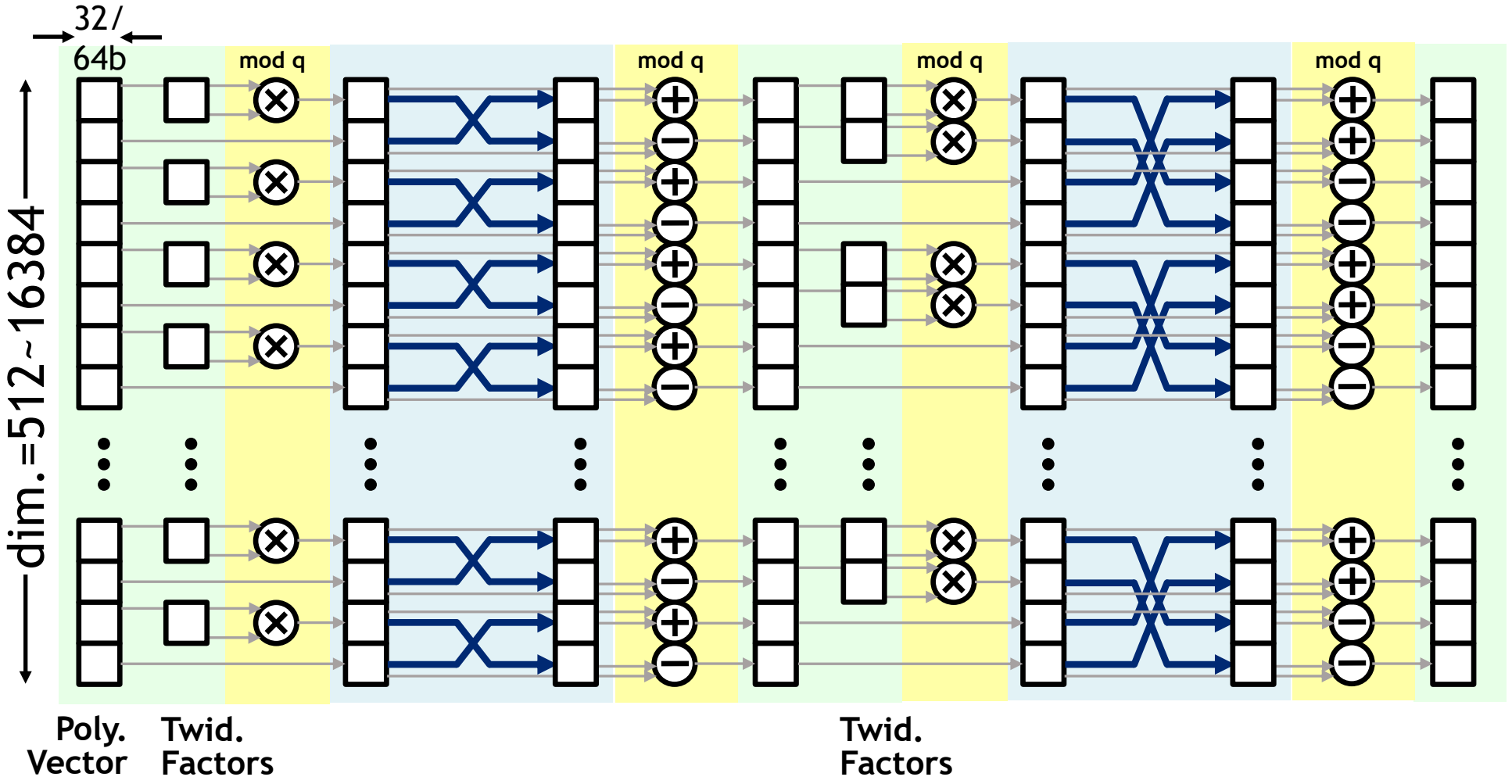
[1]Norm. to single-thread Intel Xeon Gold 5318Y CPU. [2]Measured by number-theoretic transform with N = $2^{14}$. [3]Measured by programmable bootstrapping with N = $2^{10}$.

# FHE Schemes and Accelerators



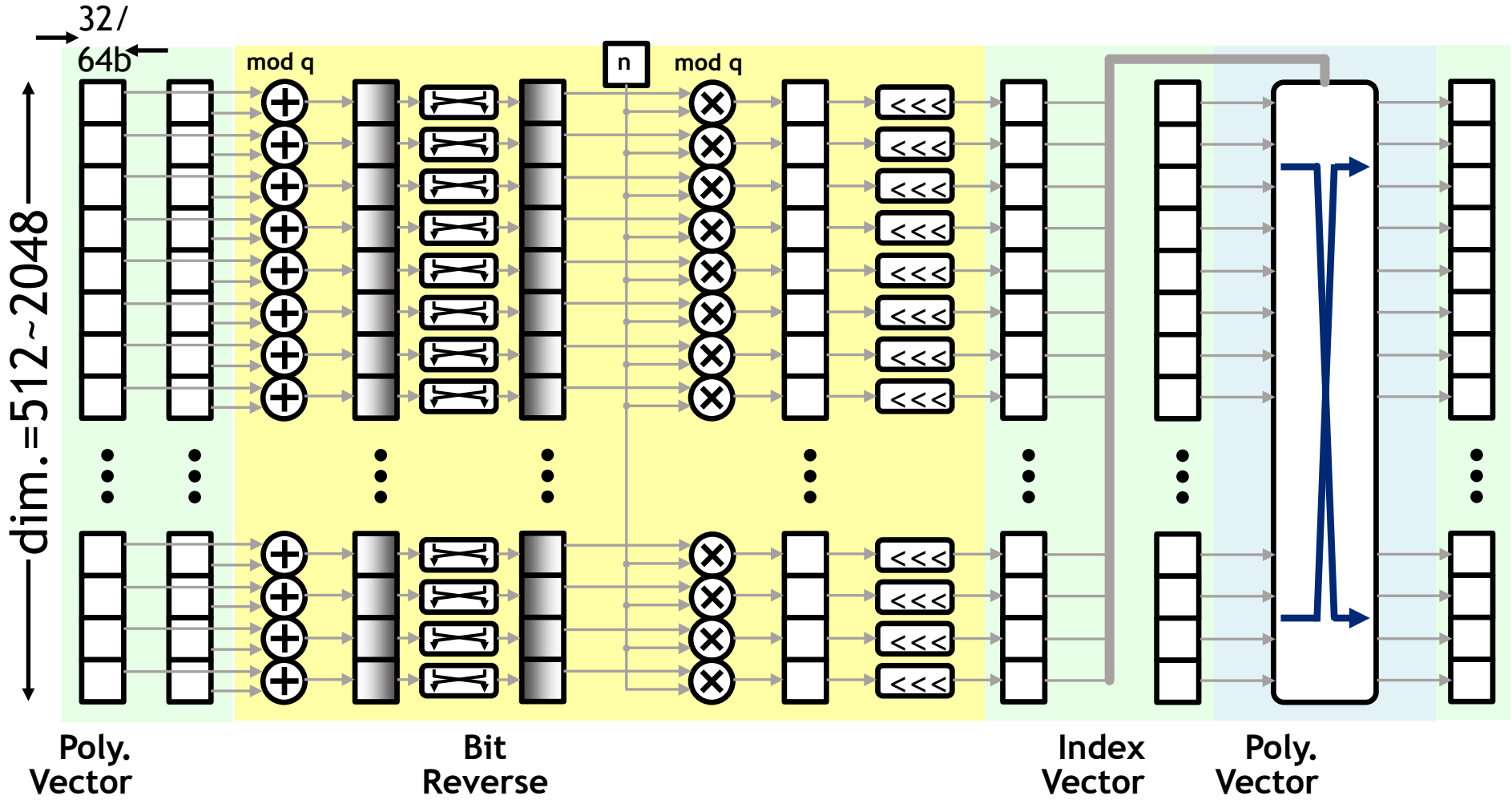

Goal: Programmable and efficient architecture for multiple FHE schemes

# Operator Decomposition in FHE




Lattice-based FHE employs highly parallel access, compute and permutation

# Operator Decomposition in FHE




E.g., Galois Dataflow

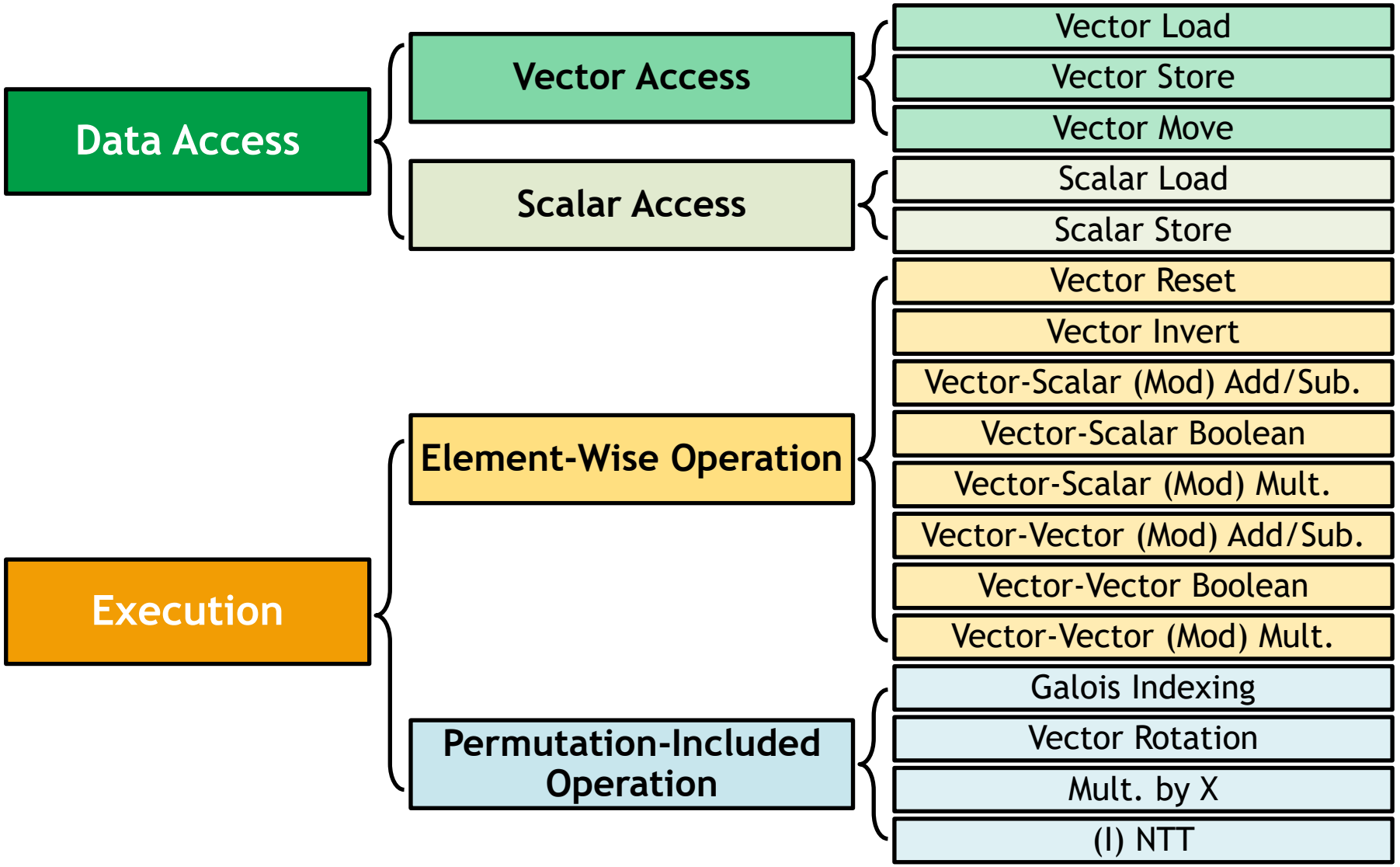
Vector Access

Element-wise Compute

Vector Permutation

Lattice-based FHE employs highly parallel access, compute and permutation

# Vector Operation Set




- Supported instr. covers the funda-mental operators in **both CKKS and TFHE schemes**
- Scalar access **reuse the memory data-path** in RISC-V CPU
- Vector access through **specialized memory interface**

# RISC-V Compatible ISA

| inst[4:2]<br>inst[6:5] | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111<br>(>32b) |
|---|---|---|---|---|---|---|---|---|
| 00 | LOAD | LOAD-FP | custom-0 | MISC-MEM | OP-IMM | AUIPC | OP-IMM-32 | 48b |
| 01 | STORE | STORE-FP | custom-1 | AMO | OP | LUI | OP-32 | 64b |
| 10 | MADD | MSUB | NMSUB | NMADD | OP-FP | OP-V | custom-2<br>/rv128 | 48b |
| 11 | BRANCH | JALR | reserved | JAL | SYSTEM | OP-VE | custom-3<br>/rv128 | ≥80b |

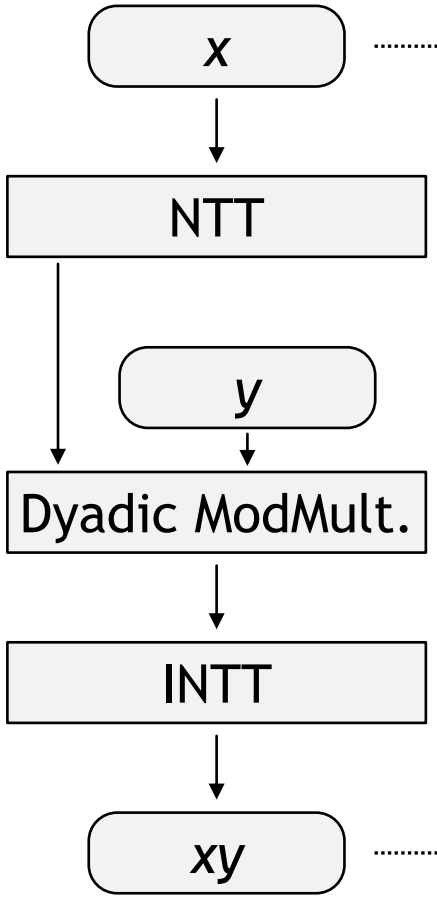
**RISC-V base opcode map (RISC-V spec. Tab.70)**

**Custom RISC-V Compatible ISA Extension**

- custom-0: OP-HE, arithmetic FHE operation
- custom-2: OP-HE-LOAD, accelerator vector/scalar loading
- custom-3: OP-HE-STORE, accelerator vector/scalar storing

# ISA Implementation Example

**E.g. FHE Dot-Product: $\langle x, y \rangle$**

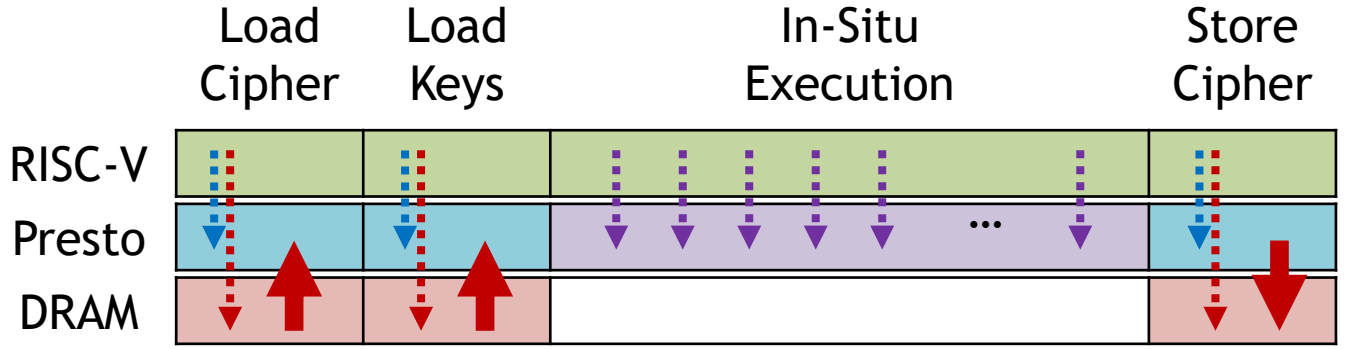



**RISC-V Compatible Assembly Code**

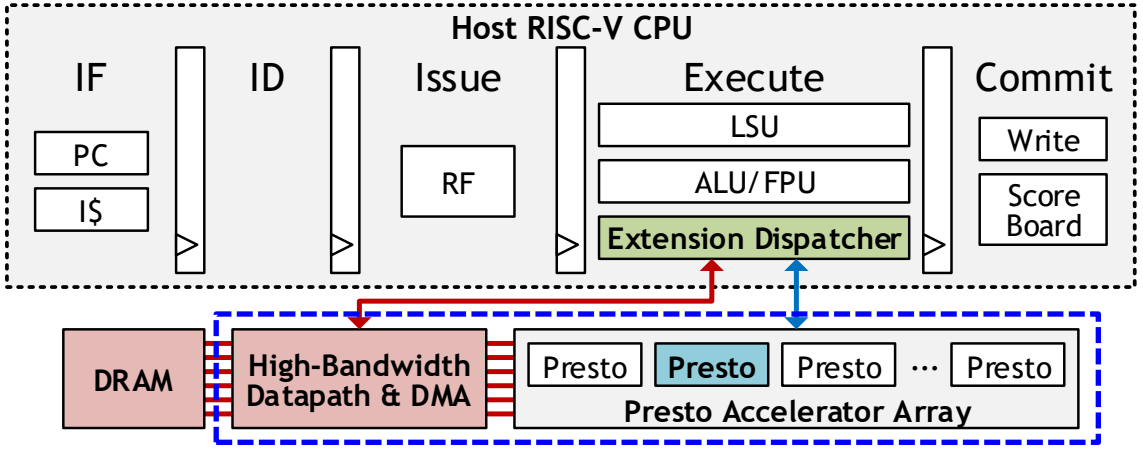
```
vld.x.ve [x], 0          # accv[0] = master[x]
vld.vb [tf0], 0          # accb[0] = master[tf0]
vhentt.vvbs 0            # accv[0] = ntt(accv[0], accv[0])
vld.vb [y], 0            # accb[0] = master[y]
vhemodmul.vvs 0          # accv[0] = modmult(accv[0], accb[0])
vld.vb [tf1], 0          # accb[0] = master[tf1]
vheintt.vvbs 0           # accv[0] = intt(accv[0], accv[0])
vst.x.ve 0, [xy]         # master[xy] = accv[0]
```

# Program. Model for RISC-V Compatible Execution

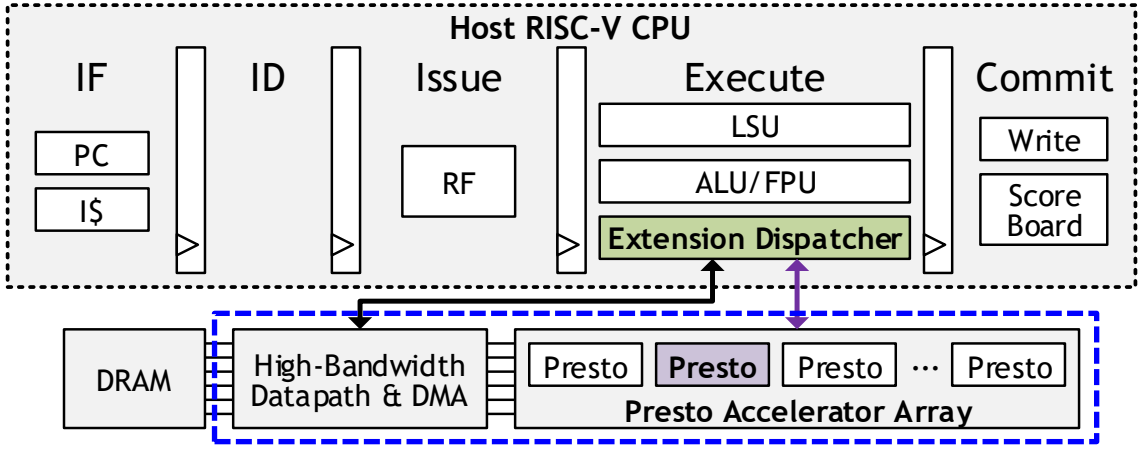
- Presto co-processor receives MS-FHE instructions from the dispatcher in a RISC-V CPU pipeline
- Presto runs in an out-of-order style




Example of TFHE PBS




Extended Load/Store Instruction

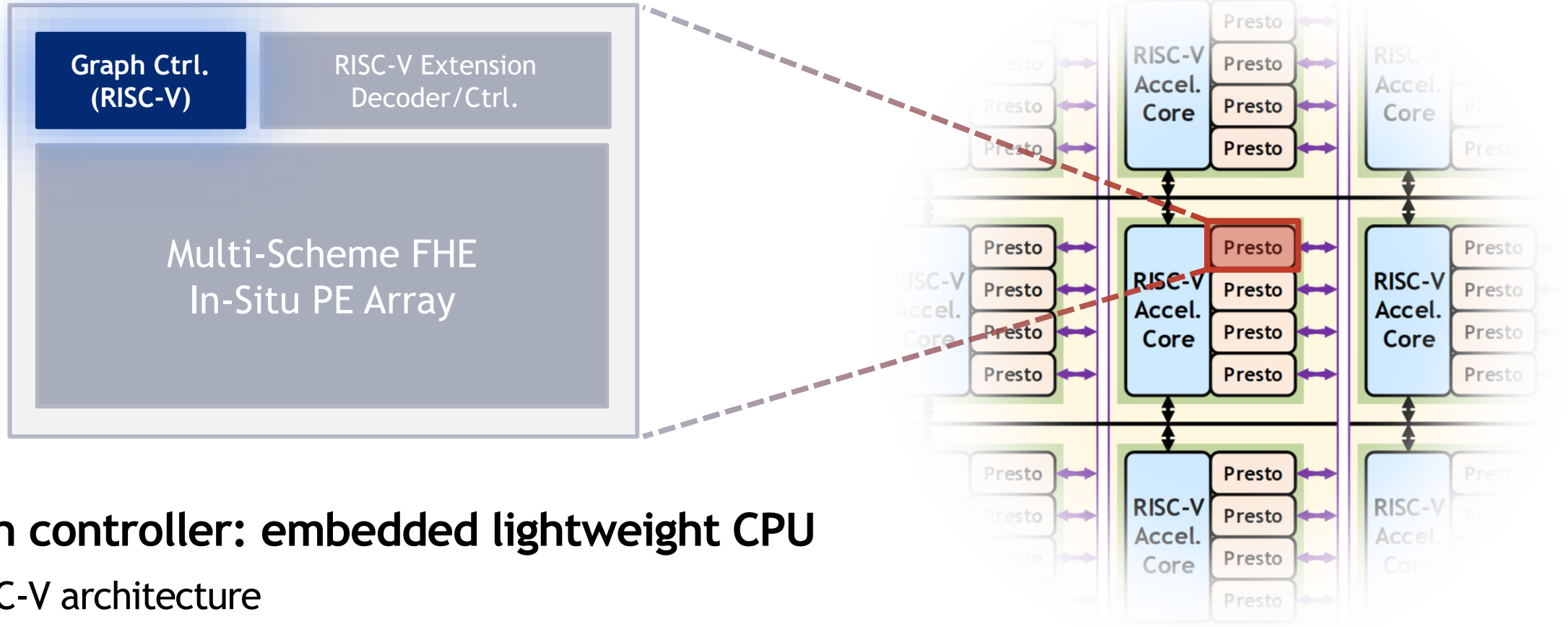



Extended Execution Instruction

# Outline

- Introduction on fully homomorphic encryption (FHE)
- Enabling scalable MS-FHE with module-learning-with-error (MLWE)
  - RISC-V vector-extension-style instruction-set architecture (ISA)
  - Programming model for multi-scheme FHE (MS-FHE)
- **Programmable and scalable FHE accelerator**
  - Architecture overview
  - Microarchitectural design for efficient communication
- Prototype and demonstrations
- Summary

# Architecture Overview

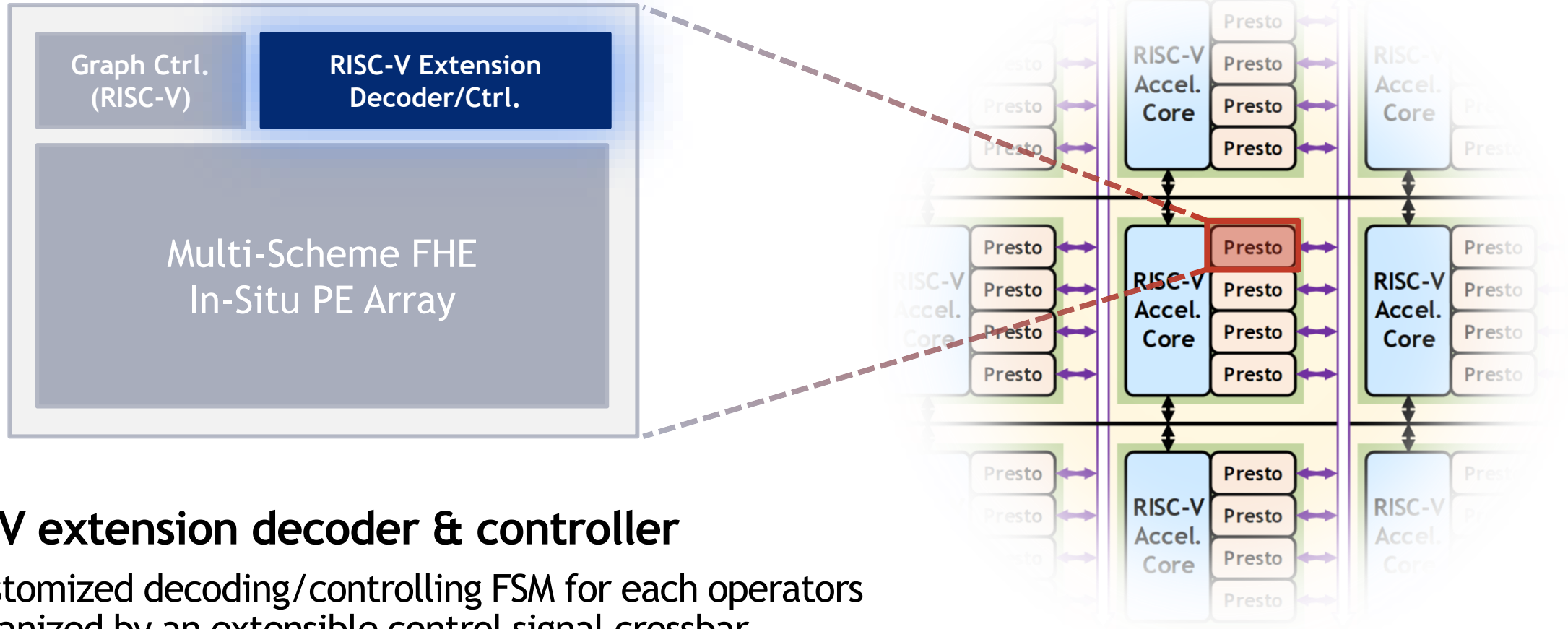



## Graph controller: embedded lightweight CPU

- RISC-V architecture
- Flexible backend optimization for better performance
- Reserve compatibility for future update

Compatibility to RISC-V environment helps scale out
This work focuses more on the Presto coprocessor

# Architecture Overview

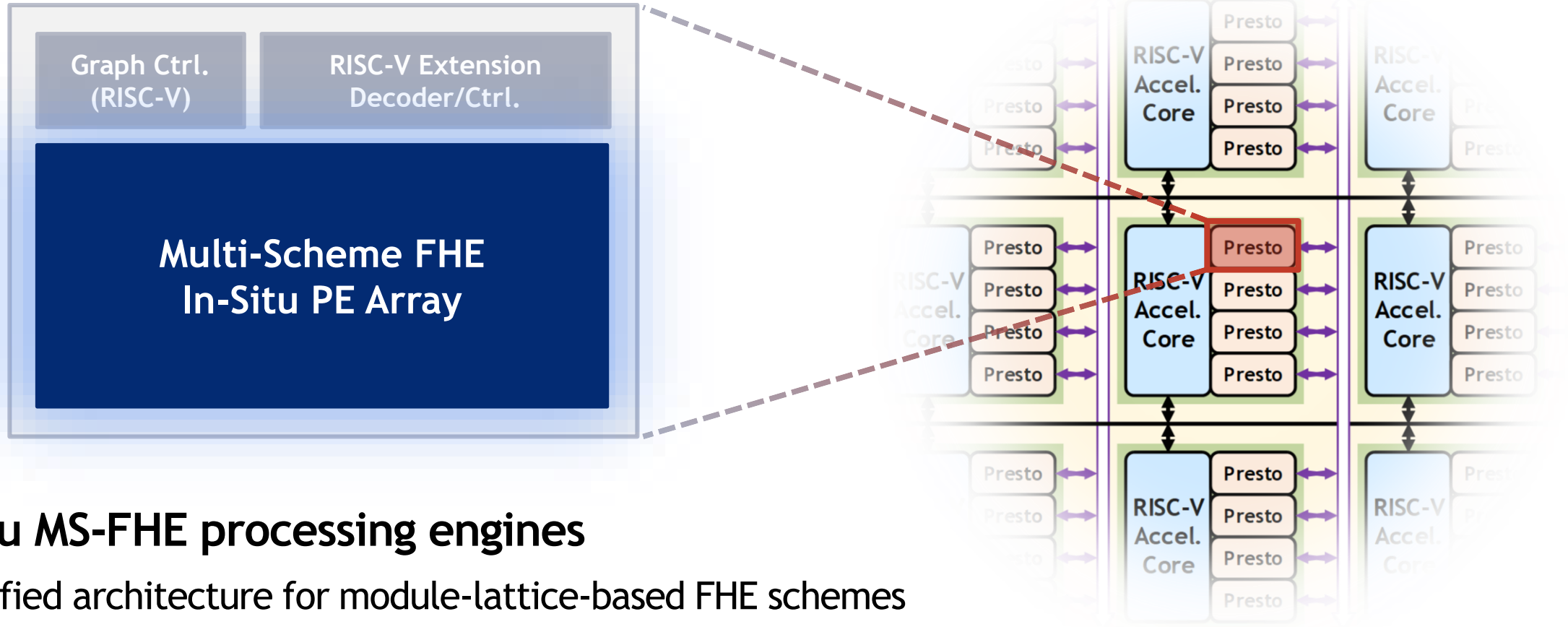



## RISC-V extension decoder & controller

- Customized decoding/controlling FSM for each operators organized by an extensible control signal crossbar
- Support out-of-order instruction issue with a 32-depth instruction queue

Compatibility to RISC-V environment helps scale out
This work focuses more on the Presto coprocessor

# Architecture Overview

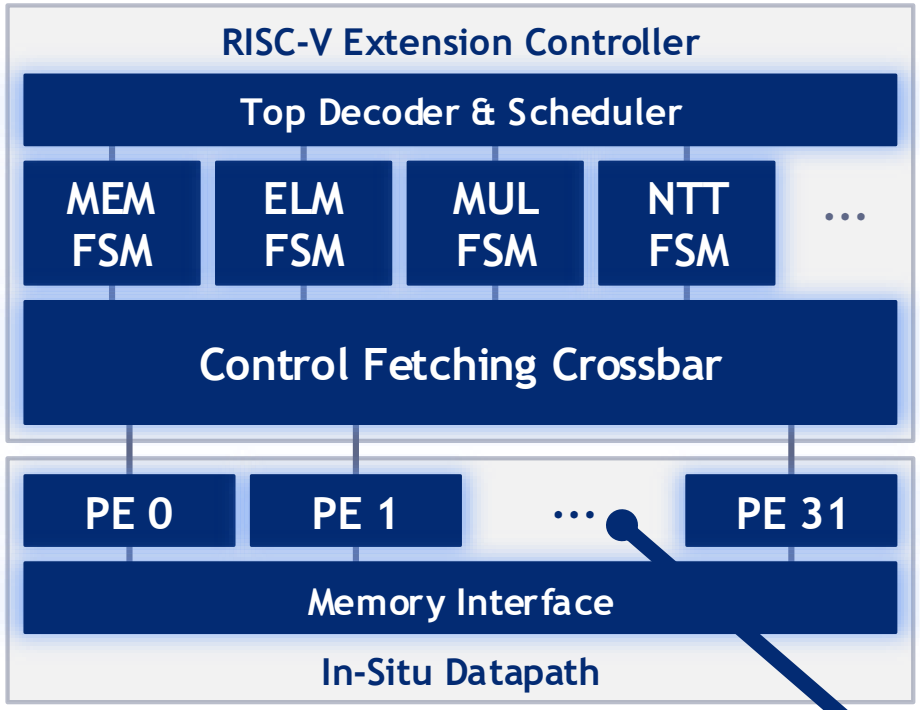



## In-situ MS-FHE processing engines

- Unified architecture for module-lattice-based FHE schemes
- Store 64 vectors in 512-dimension
- Support a complete set of CKKS and TFHE operators

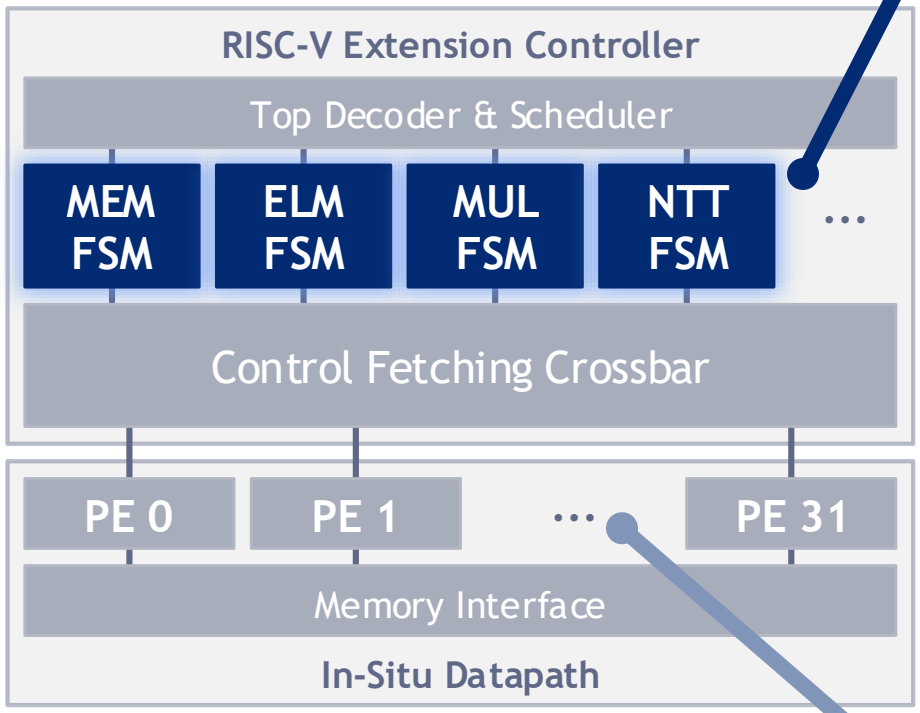
Compatibility to RISC-V environment helps scale out
This work focuses more on the Presto coprocessor

# Microarchitecture of Presto Coprocessor




32 PEs, each stores 2 vectors in 512 dimension, supporting all FHE vector execution locally

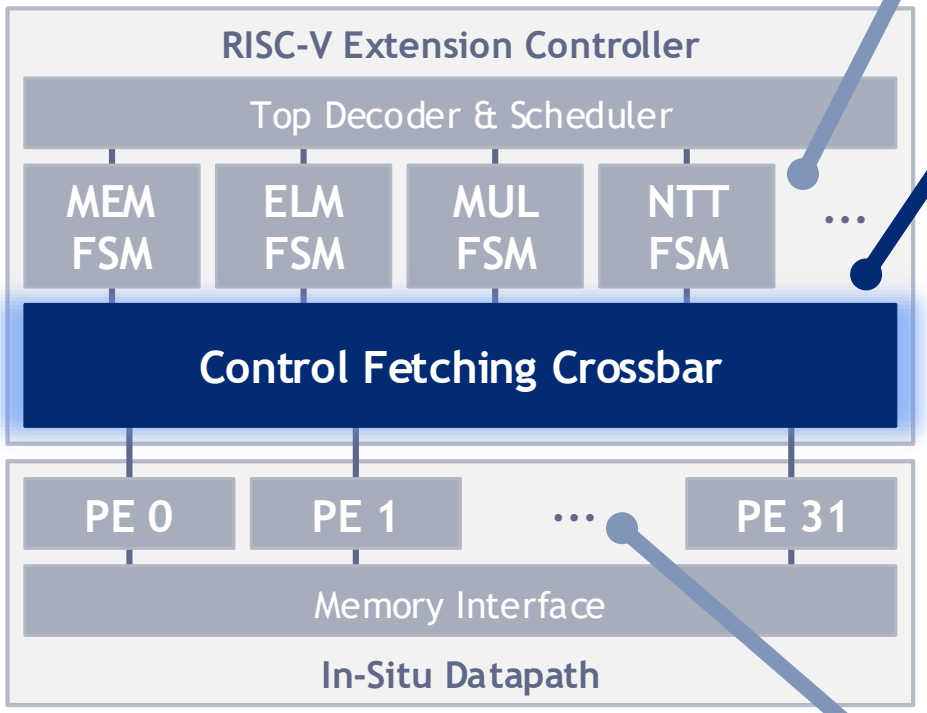
# Microarchitecture: Instruction Decoding




Control FSMs, generate cycle-level control signals for PEs

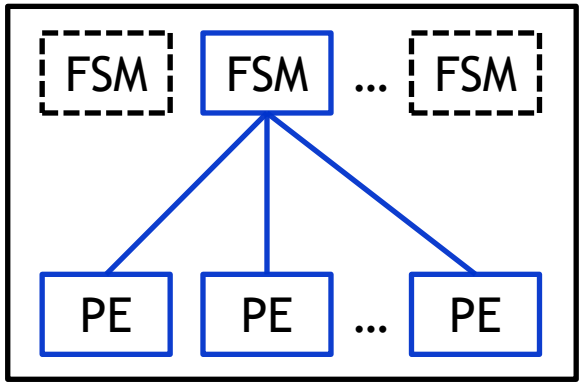
32 PEs, each stores 2 vectors in 512 dimension, supporting all FHE vector execution locally

# Microarchitecture: Control Path Steering

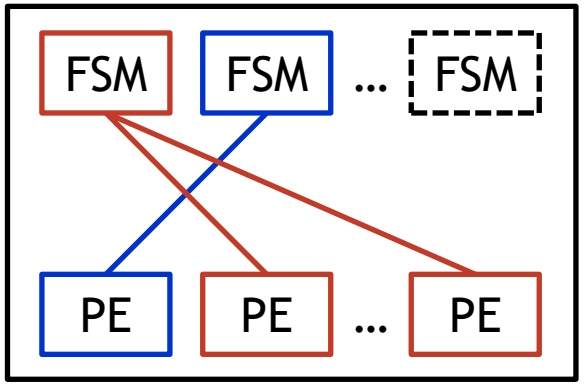



Control FSMs, generate cycle-level control signals for PEs

Crossbar between FSMs and PEs, enable out-of-order issue of instructions



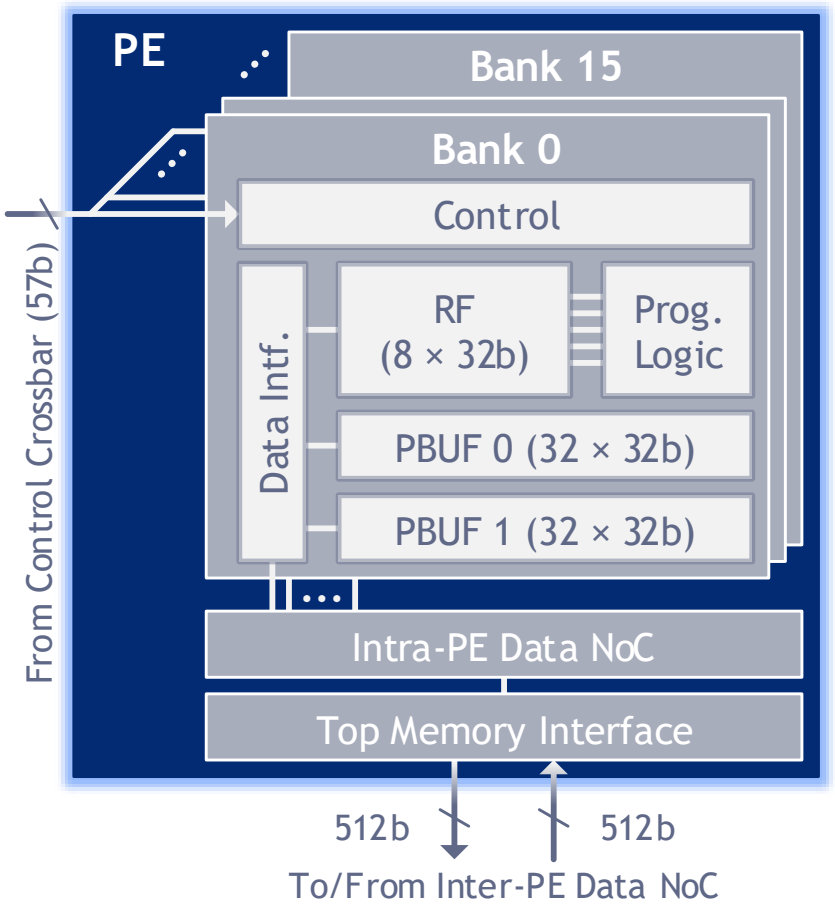

One-multiple fetch




Asynchronous fetch

32 PEs, each stores 2 vectors in 512 dimension, supporting all FHE vector execution locally

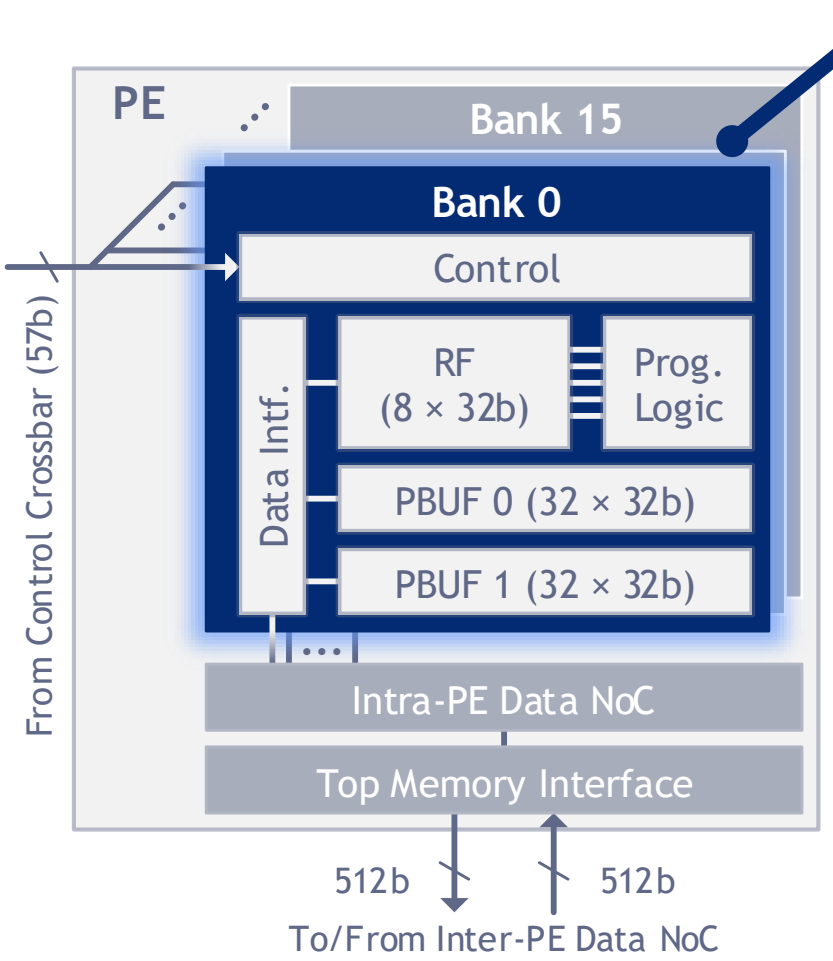
# Microarchitecture: In-Situ PE




## Processing Engine (PE)

- Accommodating 2 polynomials in 512-dimension
- Support all vector operations with parallelism 16
- Intra-PE NoC enables permutation-based operator with dimension less than 512

# Microarchitecture: In-Situ PE



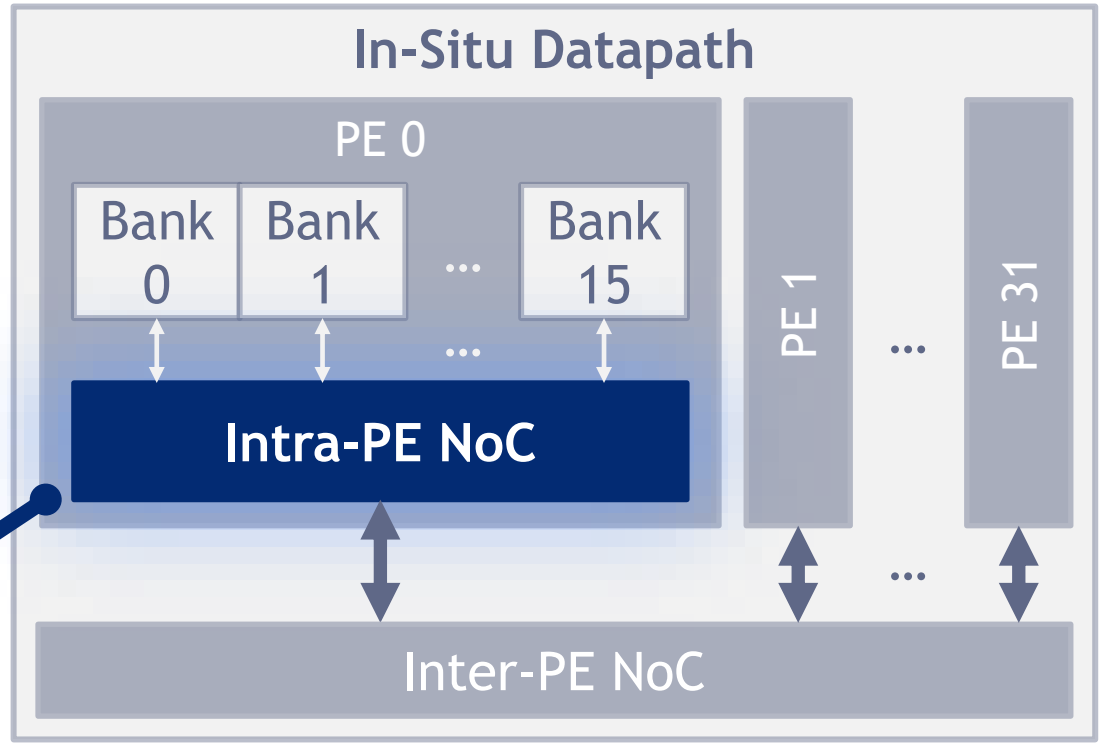

## In-situ near-memory processing bank

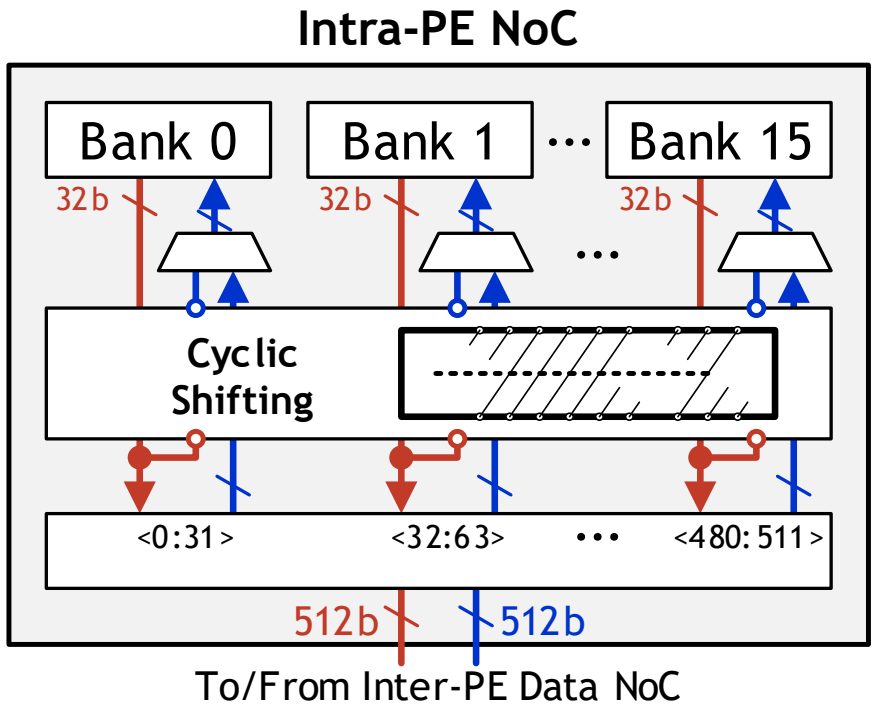
- FHE logic with sufficient programmability
- Register File (RF) for scalar operand and intermediate data
- Two latch-array-based Polynomial Buffers (PBUFs)

## Processing Engine (PE)

- Accommodating 2 polynomials in 512-dimension
- Support all vector operations with parallelism 16
- Intra-PE NoC enables permutation-based operator with dimension less than 512

# Hybrid Network-on-Chip for Cipher Polynomials

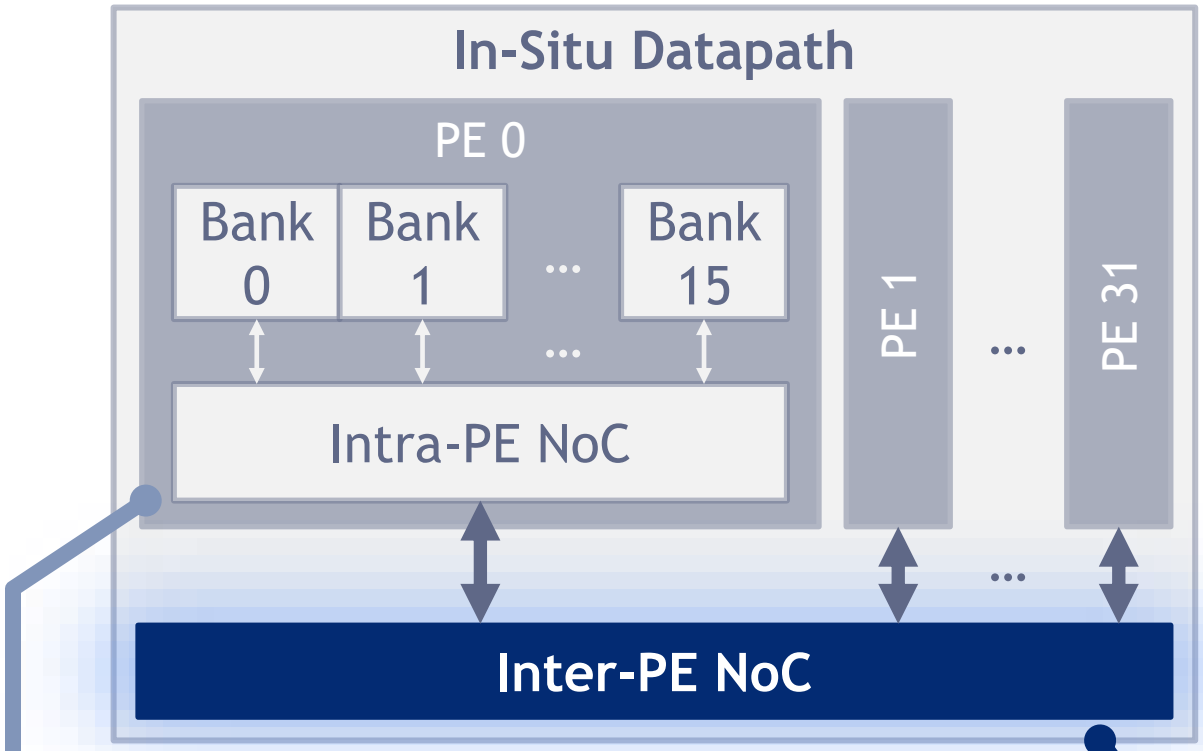


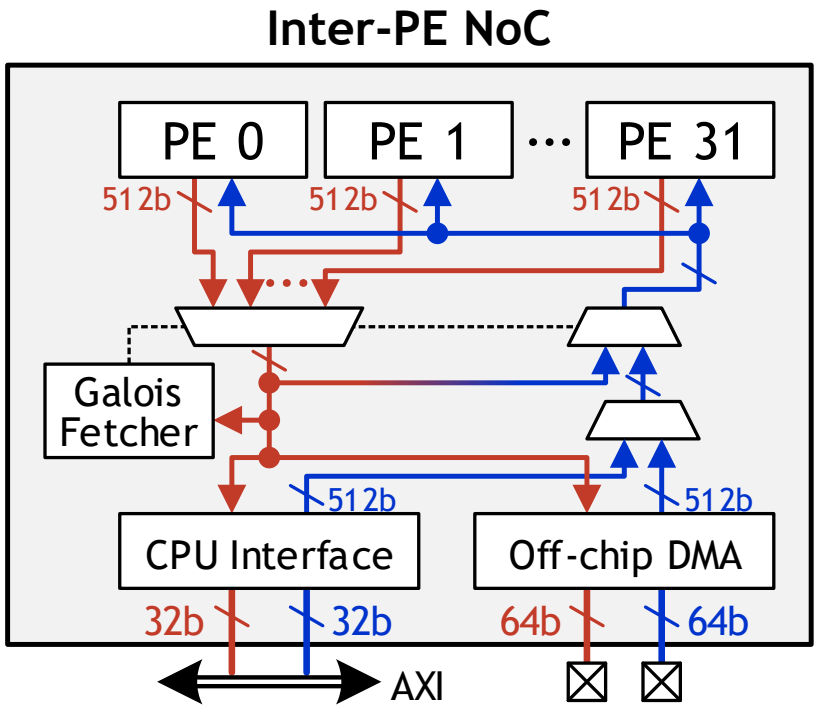




## Intra-PE (Inter-bank) NoC

- Fully configurable cyclic permutation path
- Enabling permutation with dim.≤512
- All banks transfer data synchronously

# Hybrid Network-on-Chip for Cipher Polynomials




## Intra-PE (Inter-bank) NoC

- Fully configurable cyclic permutation path
- Enabling permutation with dim.≤512
- All banks transfer data synchronously

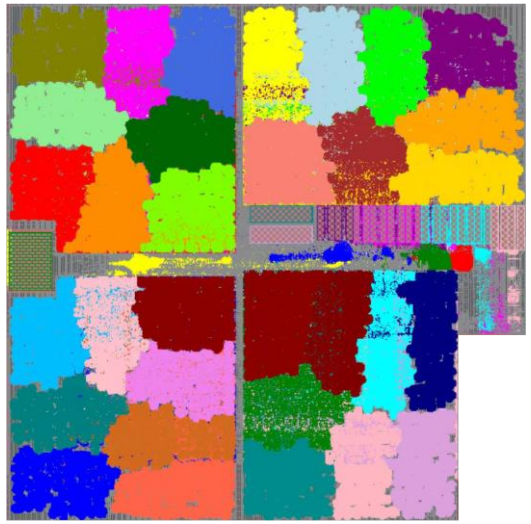
## Inter-PE NoC

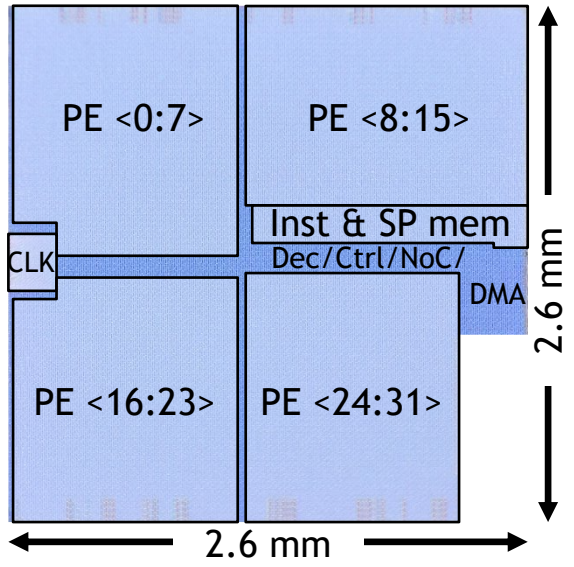
- Fully configurable write-back path
- Enabling permutation with dim.>512
- Efficient Galois indexing with attached fetcher

# Outline

- Introduction on fully homomorphic encryption (FHE)
- Enabling scalable MS-FHE with module-learning-with-error (MLWE)
  - RISC-V vector-extension-style instruction-set architecture (ISA)
  - Programming model for multi-scheme FHE (MS-FHE)
- Programmable and scalable FHE accelerator
  - Architecture overview
  - Microarchitectural design for efficient communication
- **Prototype and demonstrations**
- Summary

# Summary of Silicon Prototype

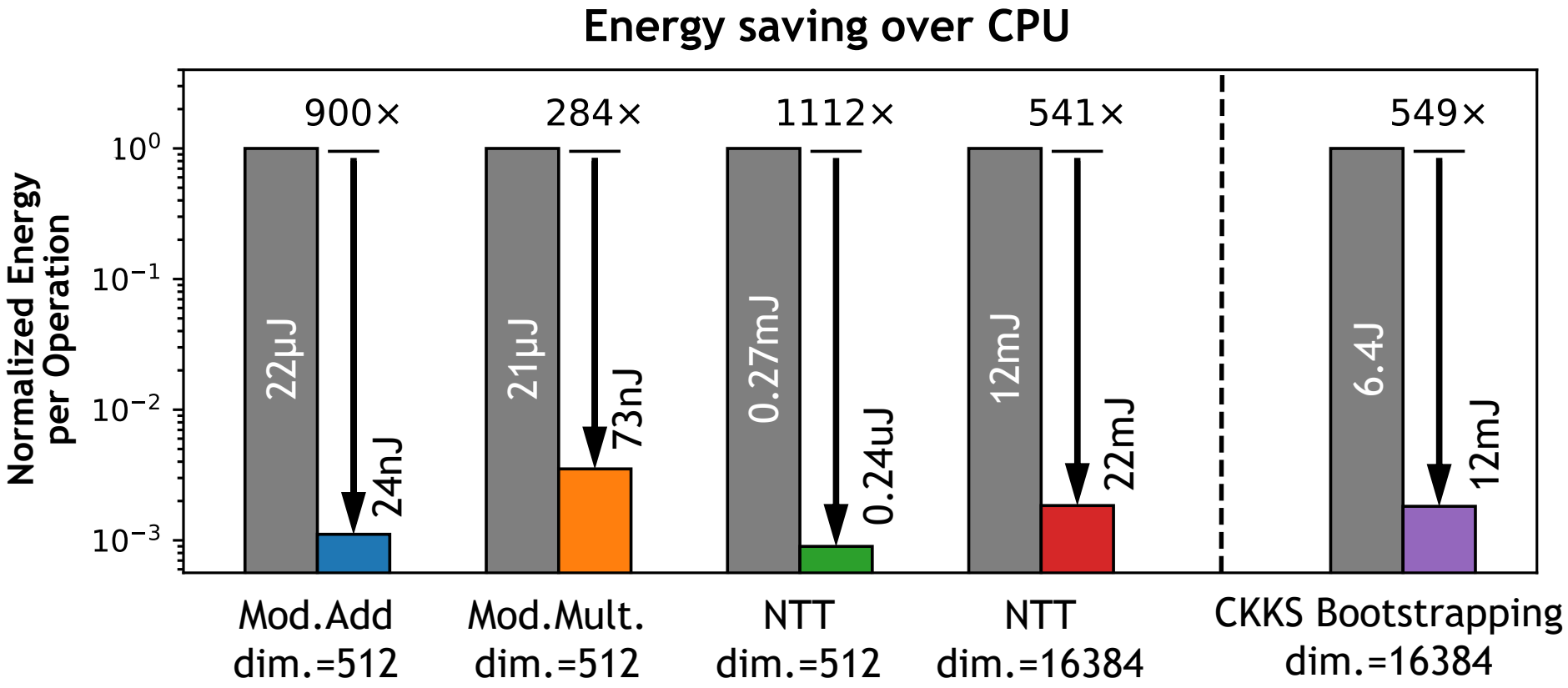





- A Presto co-processor is fabricated in 28nm CMOS, taking 5.32mm$^2$
- Fully synthesizable design, using latch-based register file for better density
- 3.23M NTT/s peak throughput (Poly. Dim = 512)
- 17.1M NTT/J peak energy efficiency (Poly. Dim = 512)

| Summary | |
|---|---|
| Technology | 28nm CMOS |
| Core Area | 6.77 mm$^2$ |
| PE Area | 5.32 mm$^2$ |
| PE Capacity | 1152 kb |
| PE Storage Density | 216.54 kb/mm$^2$ |
| Supply Voltage | 0.6 – 1.2 V |
| Frequency | 100MHz – 1GHz |
| Supported HE Tasks | (Module-Lattice) CKKS, TFHE |
| Supported Poly. Dim. | 512 – 16384 |
| Supported Bit Width | Natively 32 Expandable |

# Vector Operations Performance




*__This work__: measured at 1.2V 1GHz

**Reference CPU System**

Written in C++

Testbench tasks

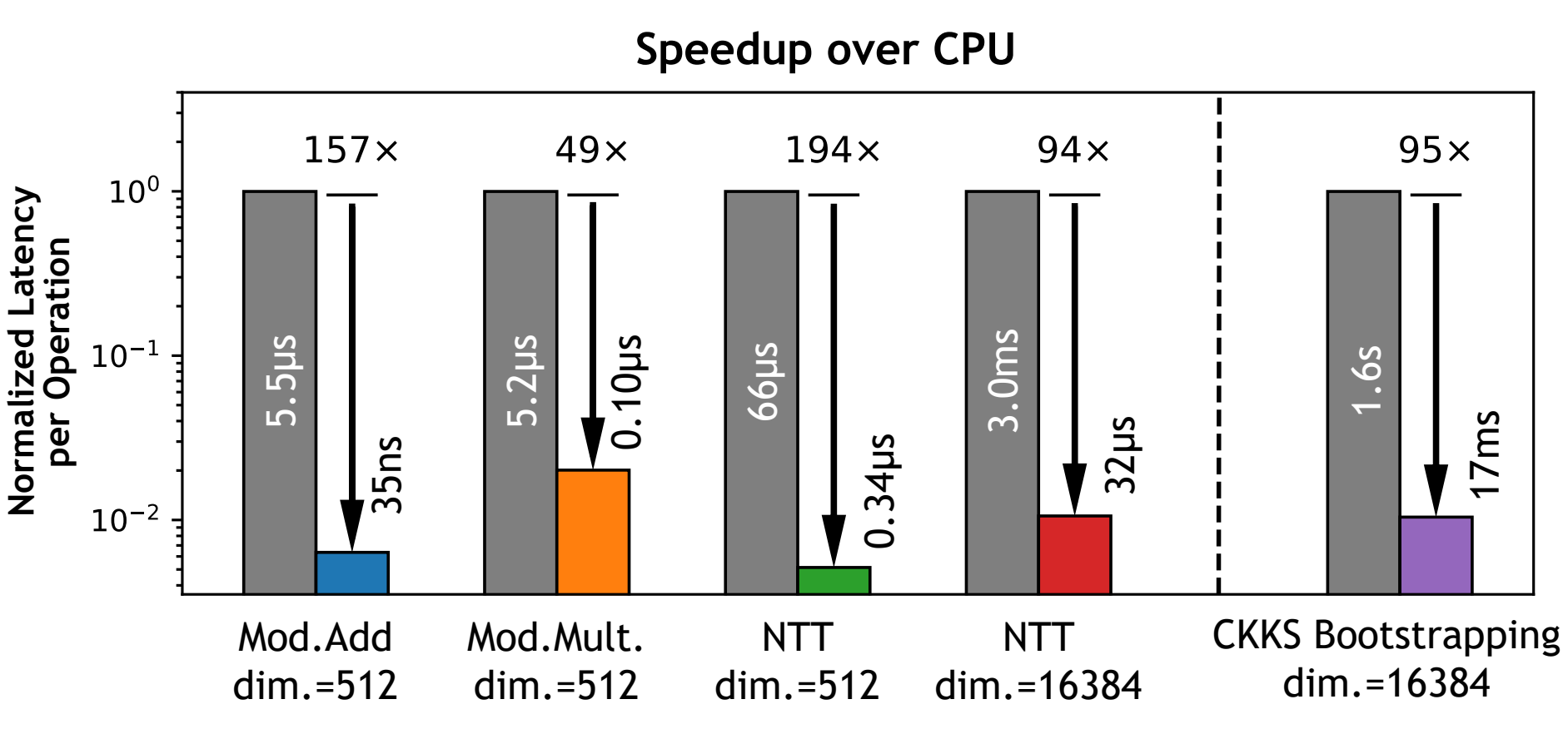
Microsoft SEAL

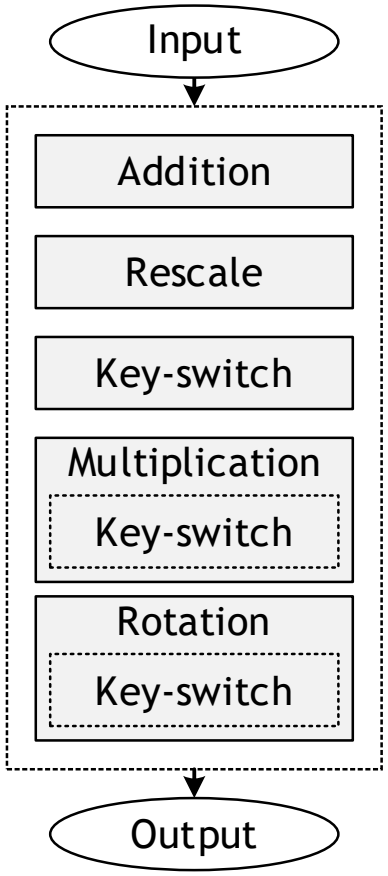
Platform: DELL R750

CPU: Intel Xeon Gold 5318Y

Memory: 512GB

# Vector Operations Performance

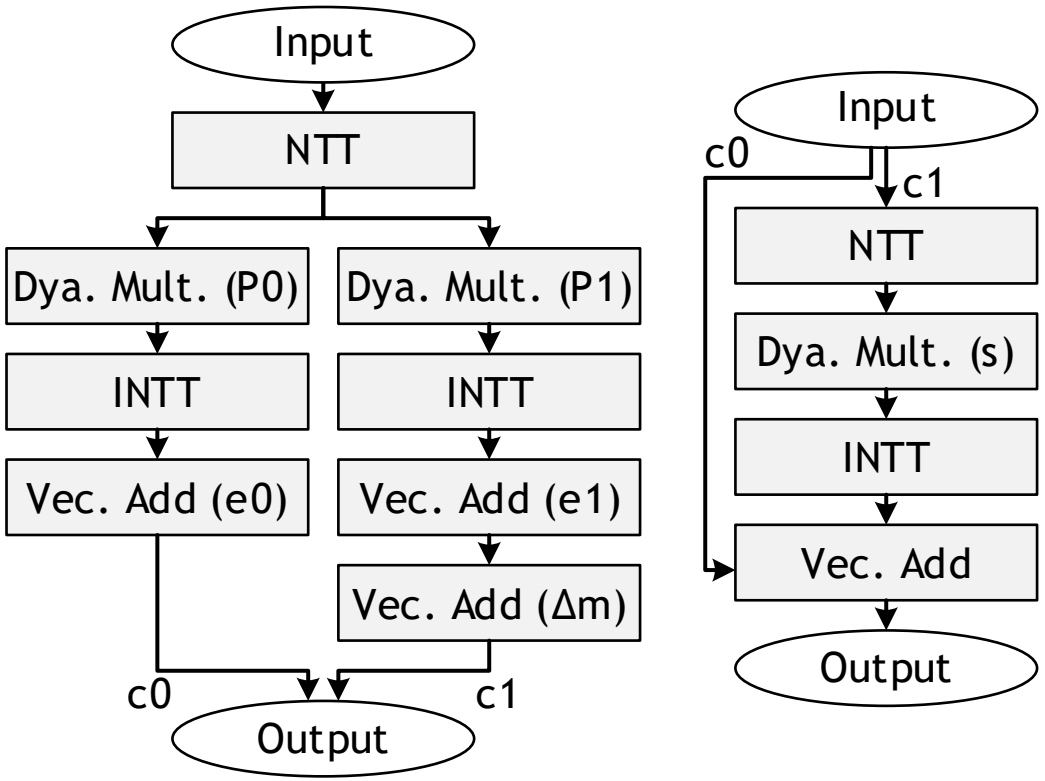


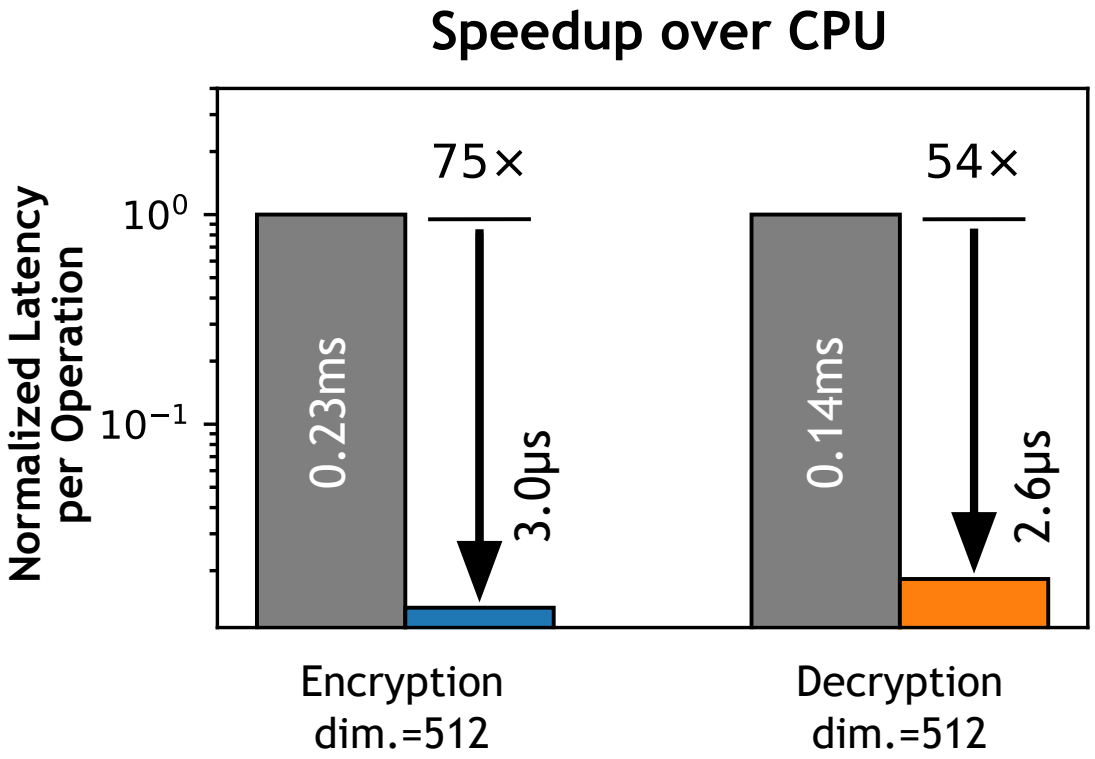

*__This work__: measured at 1.2V 1GHz
__CPU:__ normalized to a single-thread Intel Xeon Gold 5318Y with a C++-language-based FHE library customized from SEAL

E.g., CKKS Bootstrapping with dim.=16384

# Demo: RLWE Encryption/Decryption




**BFV Enc./Dec. Dataflow**
with dim. = 512




*__This work__: measured at 1.2V 1GHz
__CPU:__ normalized to a single-thread Intel Xeon Gold 5318Y with a C++-language-based FHE library customized from SEAL

# Demo: TFHE-Based AES Transciphering

- FHE-level safety with a symmetric-key-encryption-level user cost
  - Low-cost plug-and-play key management infrastructure


**Cellphone**

AES Ciphertext

AES

1G

**Homomorphic Transciphering**

Homomorphic Decryption HomDec

$\mathrm{HomDec}_{c2}(\mathcal{E}_{c1}(m))$

FHE

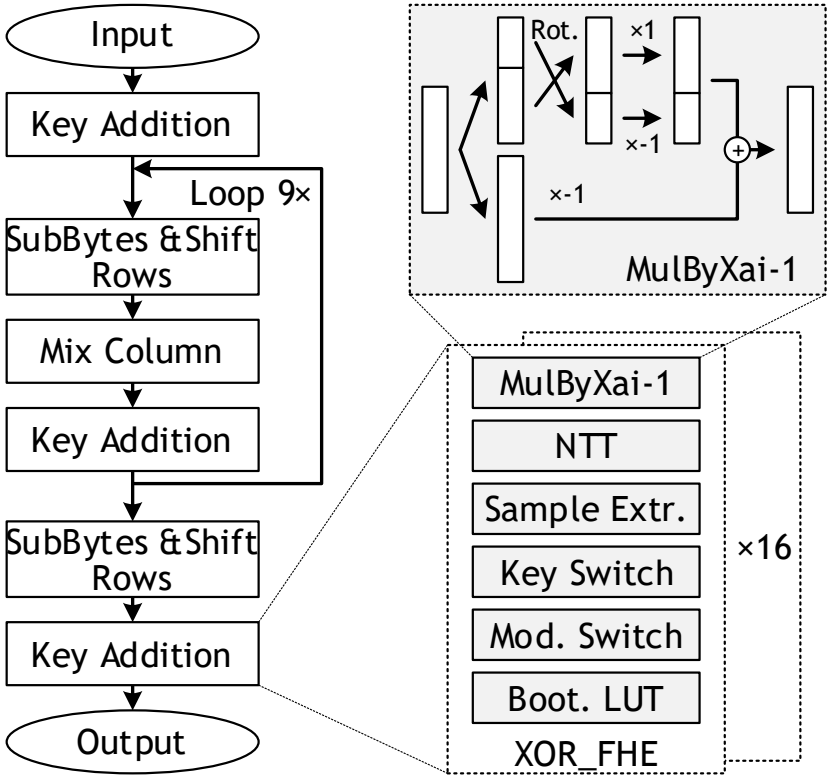
2T

**Computing Unit**

FHE Ciphertext

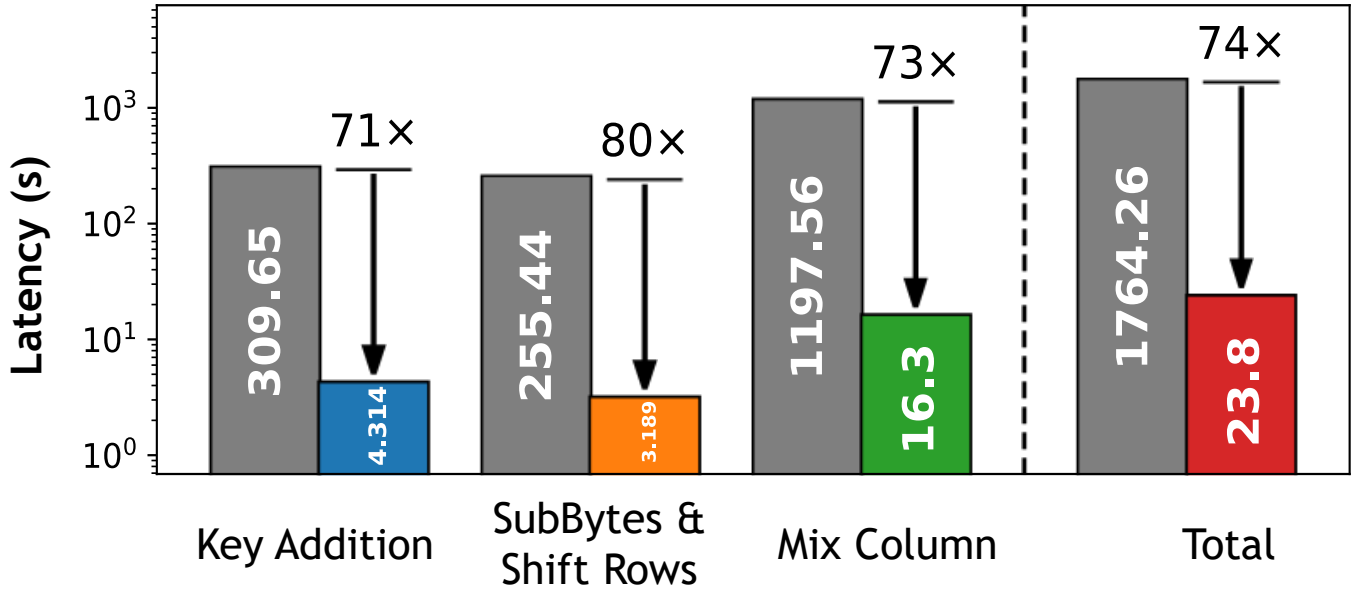
# Demo: TFHE-Based AES Transciphering

## AES Transciphering Dataflow




N = 2048, k = 1, l = 3, iteration = 10, TFHE library

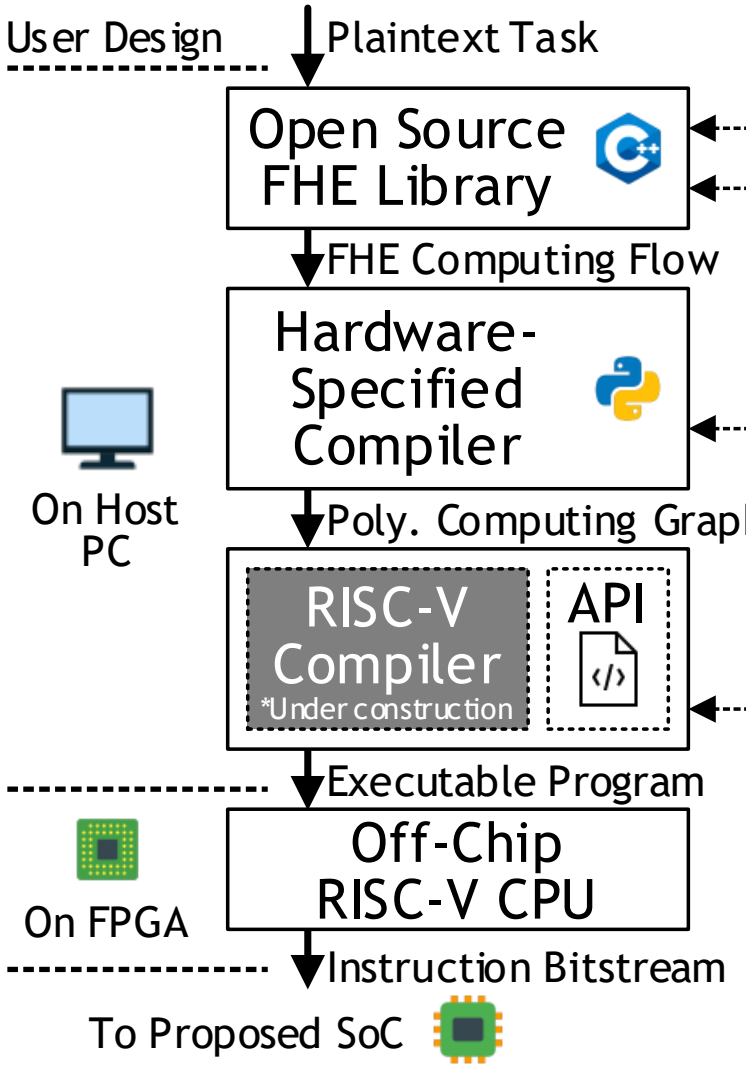
## Speedup over CPU




***This work**: measured at 1.2V 1GHz
**CPU**: normalized to a single-thread of Intel Xeon Gold 5318Y with a C++-language-based FHE library customized from TFHE

- **Presto accelerator achieves 74× speedup to CPU (w/o tech. node norm.)**

# Software stack




**SEAL Library for CKKS**

```
multiply_rlwe_plain_inplace(cipher,
  plainClient);
rlwe_mod_switch_scale_to_next_inplace(cipher);
...
```

**TFHE Library**

```
LweSample *ct =
  new_gate_bootstrapping_ciphertext_array
  (3, params);
BootsOR(&ct[2], &ct[0], &ct[1], &(key->cloud));
...
```

**HE Specified Compiler**

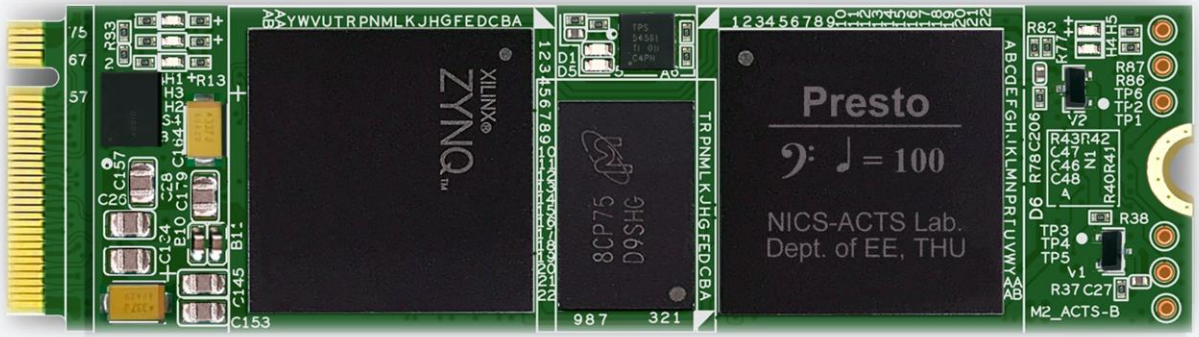
```
graph.append(0, 0, NTT, modulo = 0x79001)
graph.append(1, 1, NTT, modulo = 0x79001)
graph.append(0, 1, mult)
flow = generate_PE_flow(graph)
...
```

**RISC-V Extended Instruction**

```
// vld.s.x ss1 xs1 # read ss1 = xs1
{6'b0, vm, xs1, 5'b0, 3'b011, ss1, 7'b1011011}
// vst.x.s xs1 ss1 # load xs1 = ss1
{6'b0, vm, xs1, 5'b0, 3'b011, ss1, 7'b1111011}
...
```

# FHE Acceleration Solutions with Presto

- **Presto M.2 for portable devices**




- PCIe Gen2 ×4 interface to the host system
- 1GB on-board memory for cipher & key storage
- AMD/Xilinx Zynq FPGA as bridge logic with 3.2GB/s peak bandwidth to Presto SoC
- Ethernet and AMD PetaLinux ready for developing/debugging

- **Next step**
- **PCIe acceleration card for workstations**

# Summary

- Fully homomorphic encryption (FHE) enables highly secure private computing, however requires aggressive acceleration
- Different application requires various FHE schemes with different polynomial dimensions, making challenges for a unified architecture
- Module-learning-with-errors (MLWE) serves a bridge for unifying the execution flow for CKKS and TFHE scheme
- Programmable multi-scheme FHE acceleration with RISC-V compatible instructions is demonstrated (with supporting SW libs)
- The 28nm prototype shows >70× speedup to a server-CPU core for FHE evaluation

* Acknowledgements: funding provided in part by NSFC

# Thanks for your attention!

Presto: A Unified RISC-V-Compatible SoC for Multi-Scheme FHE Acceleration over Module Lattice