

Fully Homomorphic Encryption (FHE)

- FHE enables **computation** over **encrypted data**.

Outsourcing the Computation of $f(x) = (x + \alpha)^2$

①

$\llbracket x \rrbracket = \text{Encrypt}(x)$

②

$\llbracket w \rrbracket = \text{PtAdd}(\llbracket x \rrbracket, \alpha)$

③

$\llbracket y \rrbracket = \text{Mult}(\llbracket w \rrbracket, \llbracket w \rrbracket)$

④

$y = \text{Decrypt}(\llbracket y \rrbracket)$

FHE Applications

- Medical, Financial, Supply chain, Marketing, etc.
- Machine learning: **Logistic regression training**

Training Dataset

Vector Inner Product

Sigmoid Function

Update

Vector Inner Product

Sigmoid Function

Trained Model

30 iterations

Plaintext Circuit

Encrypted Circuit

[[Training Dataset]]

[[Inner Product]]

PolyEval

Update

[[Inner Product]]

PolyEval

[[Trained Model]]

3 iterations + Bootstrapping

10 iterations

- Problem:** **Bootstrapping** is the major bottleneck.

SimFHE: Custom Simulator

- SimFHE:
 - A **python-based architectural simulator** for modeling CKKS operations at subroutine level.
 - Set **scheme parameters**.
 - Set **architecture parameters**.
 - Keeps track of **compute operations** as well as **DRAM transfers** for a given cache size.
- Study **compute & memory trade-off**.
- Explore **various optimizations** and **select parameters** to optimize throughput.

Analysis using SimFHE

- Arithmetic intensity analysis for CKKS operations and an end-to-end application.
 - < 1 Op/byte

Primitive Operations in CKKS Scheme

Arithmetic Intensity

PtAdd

Add

PtMult

Mult

Rotate

Conjugate

HRotate

Bootstrapping Steps

Arithmetic Intensity

CoeffToSlot

PolyEval

SlotToCoeff

Logistic Regression Training Steps

Arithmetic Intensity

InnerProduct

PolyEval

LR Iteration

Bootstrapping

- Bootstrapping has **low arithmetic intensity**.
- Limited **memory bandwidth** is the **bottleneck**.

Memory-Aware Design (MAD) Techniques

- Memory access pattern optimization:
 - Regorganize **low-level memory access pattern** for faster data access.

Limb Index

Slot Index

31

30

29

28

27

26

25

24

23

22

21

20

19

18

17

16

15

14

13

12

11

10

9

8

7

6

5

4

3

2

1

0

(a) Baseline address mapping

Lower-order bits for limb index

Higher-order bits for slot index

Higher-order bits for limb index

Lower-order bits for slot index

31

30

29

28

27

26

25

24

23

22

21

20

19

18

17

16

15

14

13

12

11

10

9

8

7

6

5

4

3

2

1

0

(b) Optimized address mapping

Byte index within a row

Column Index

Bank Index

Bank group

Rank Index

Row Index

	Limb-wise access	Slot-wise access	Full Switch
Baseline	2.3 ms	9.2 ms	11.5 ms
Optimized	2.5 ms	2.2 ms	4.7 ms

- Hierarchical **caching optimizations**:
 - Compute as much as possible to **maximize data reuse**.
 - Re-order operations to **maximize cache utilization**.

Cache Optimizations

DRAM Transfers (in GB)

Base line

O(1)-limb Cache

β -limb Cache

α -limb Cache

Limb Re-order

- Algorithmic optimizations:
 - Double-hoisting to **reduce orientation switch**.
 - Improve arithmetic intensity** of low level operations.

Algorithmic Optimizations

DRAM Transfers (in GB) and Operations (in GOP)

Memory-Optimized Baseline

ModDown Merge

ModDown Hoisting

Key Compression

Key Observations

- 8.2x** improvement in **bootstrapping throughput**.

Bootstrapping Throughput Comparison

Throughput

Jung et al. (GPU)

Bossuat et al. (Optimized CPU)

Our Work

- 3.2x** improvement in **arithmetic intensity**.

Optimized Logistic Regression Training Steps

Arithmetic Intensity

InnerProduct

PolyEval

LR Iteration

Bootstrapping

- Memory bandwidth** is still a **bottleneck**.
- CPU/GPU solutions** are limited by main memory bandwidth.
- Existing **ASIC proposals** are too **expensive**:
 - Need large on-chip memory and register file.
 - Need 12nm/7nm technology nodes.

FAB: FPGA-based Accelerator

- FAB - an **FPGA-based accelerator** for **bootstrappable** FHE workloads.
 - First ever bootstrapping implementation on FPGA.
 - Secure and practical parameter set**
 - Balanced FPGA design**.
 - Not memory-bound**.
 - Only **256 functional units** operating at 300MHz.
 - High data rates** from/to main memory at 450MHz.
 - Effective utilization** of limited on-chip memory.
 - Modified datapath** for basic FHE operations like KeySwitch.
 - Smart Operation scheduling**.

FAB: Overall Architecture

- Four components:
 - Host CPU** offloads RTL design to the FPGA.
 - RTL design packaged as **kernel code**.
 - 8GB HBM2** memory stacks.
 - 100G Ethernet **CMAC subsystem**.

Cloud Server

Host CPU (X86) (Host Code)

AXI4-lite

PCIE

Global Memory

FPGA

CMAC Subsystem

CMAC

Tx Adapter

Rx Adapter

256

Mod Mult

Mod Add

Mod Sub

Auto-morph

R F

NTT Address Generation

URAM Address Generation

BRAM Address Generation

Control Logic

Wr FIFO

Rd FIFO

URAM Bank c0

URAM Bank c1

URAM Misc. Bank

BRAM Bank c0

BRAM Bank c1

BRAM Misc. Bank

HBM2 (Stack 0)

HBM2 (Stack 1)

100G Ethernet Switch

Other FPGAs

FAB: Evaluation

- Designed in **Verilog** and implemented on **Xilinx Alveo U280** deployed in **cloud environment**.
- Basic FHE operations'** performance:

	FAB	GPU	Speedup (Time)
Add	0.04 ms	0.16 ms	3.85x
Mult	1.71 ms	2.96 ms	1.73x
Rescale	0.19 ms	0.49 ms	2.62x
Rotate	1.57 ms	2.55 ms	1.62x

- Bootstrapping** performance:

	Time (micros)	Speedup (Time)	Speedup (Cycles)
CPU	101.78	213x	2485x
GPU	0.740	1.55x	6.35x
FAB	0.477	-	-

- Logistic regression** model training with 8 FPGAs:

	Time (sec)	Speedup (Time)	Speedup (Cycles)
CPU	37.05	456x	5318x
GPU	0.775	9.5x	39x
FAB-1	0.103	1.3x	1.3x
FAB-2	0.081	-	-

Conclusion

- Affordable** and **practical** FHE acceleration solution.
 - MAD** techniques provide
 - Optimizations feasible with **small cache** sizes, agnostic of platforms.
 - FAB** provides
 - Practical performance** using **FPGA** at a fraction of ASIC cost.

QR Code 1

QR Code 2