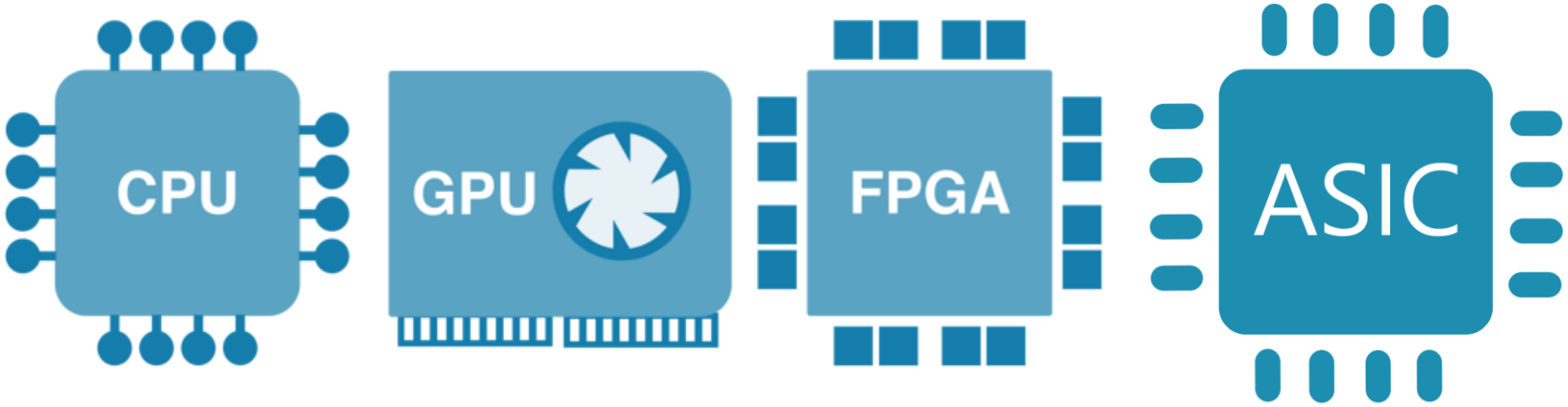
Too much BS (Bootstrapping)

Hardware Acceleration for FHEW

Jonas Bertels, Michiel Van Beirendonck, Furkan Turan,
Ingrid Verbauwhede

May 2023,
FHE.ORG

Hardware Acceleration with FPGA's

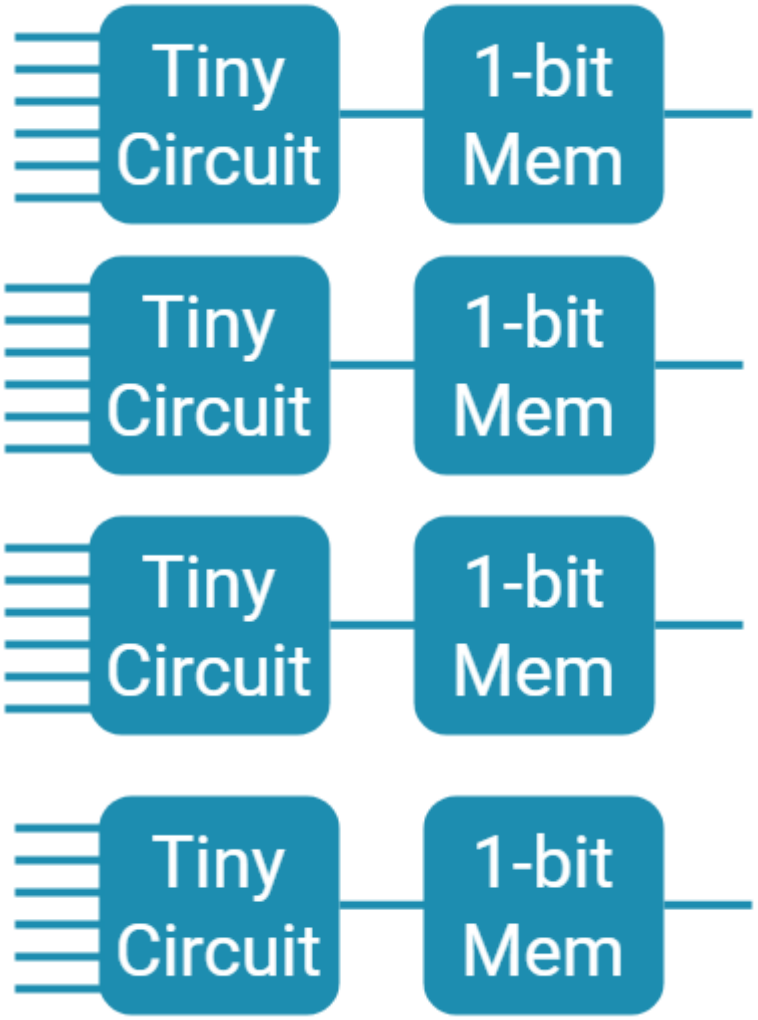


EASY PROGRAMMING ⟷ HIGH EFFORT

CHEAP ⟷ EXPENSIVE

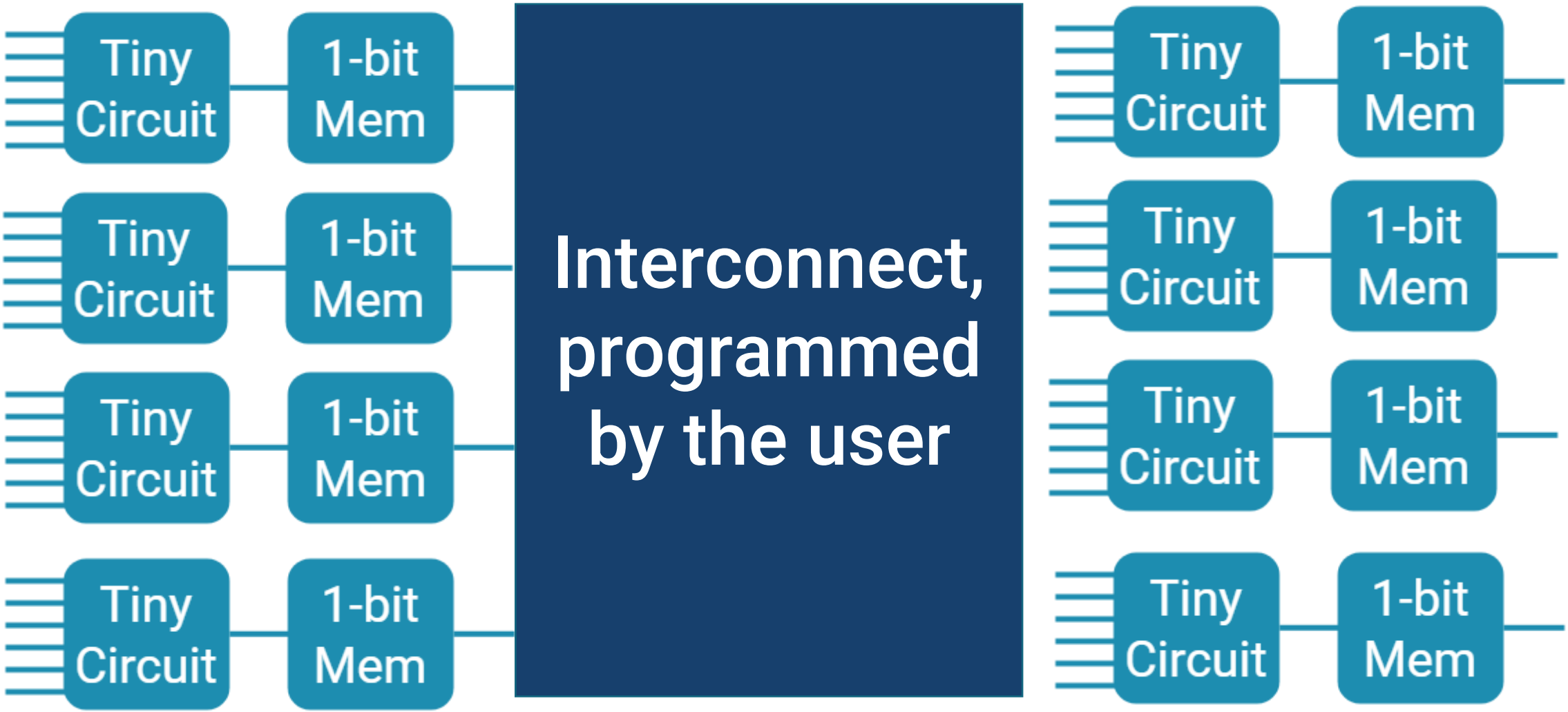
FLEXIBILITY ⟷ EFFICIENCY

The innards of an FPGA



… x1000000!

The innards of an FPGA



Tiny circuit?

Look-up Table:

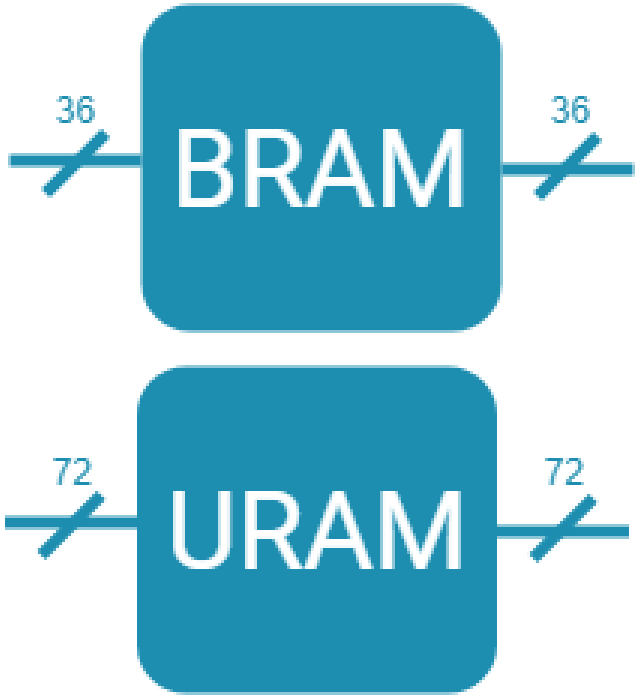
Any f: $\mathbb{Z}_{2^6} \to \mathbb{Z}_2$

Digital Signal Processor:

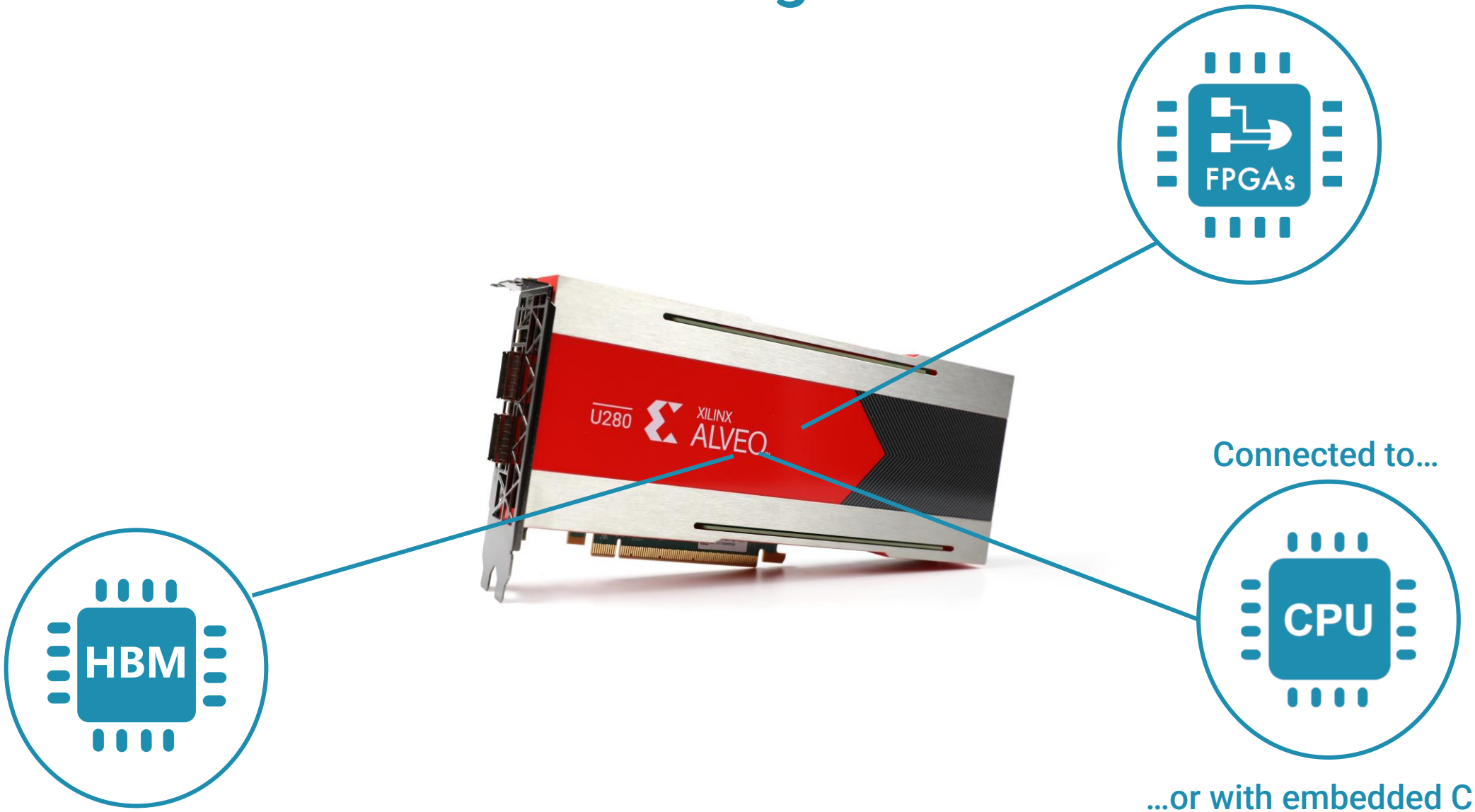
$*,+: \mathbb{Z}_{2^{18}} \times \mathbb{Z}_{2^{27}} \to \mathbb{Z}_{2^{48}}$

Block RAM/ Ultra RAM

Small memory



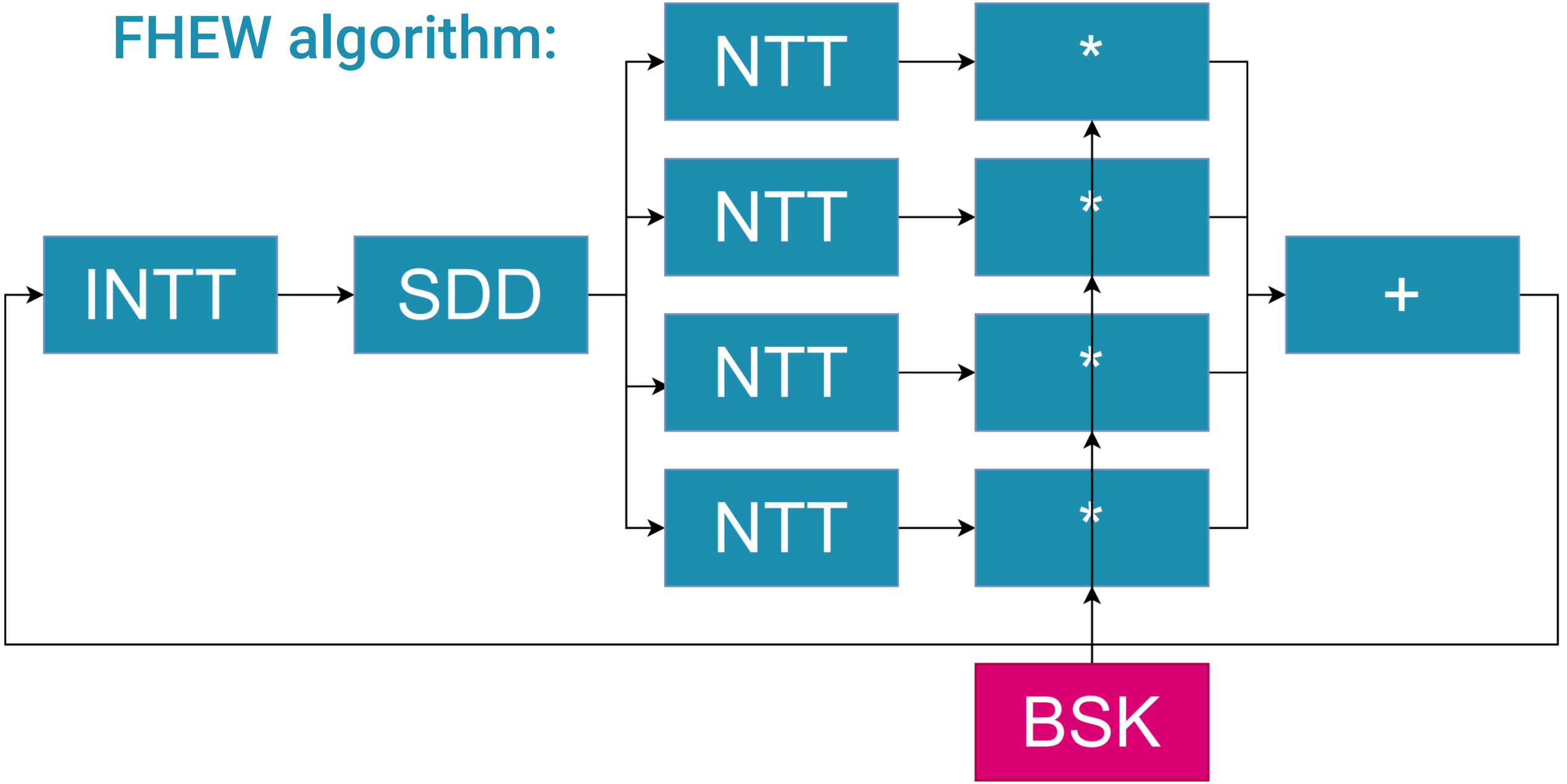
FPGA’s and their surroundings



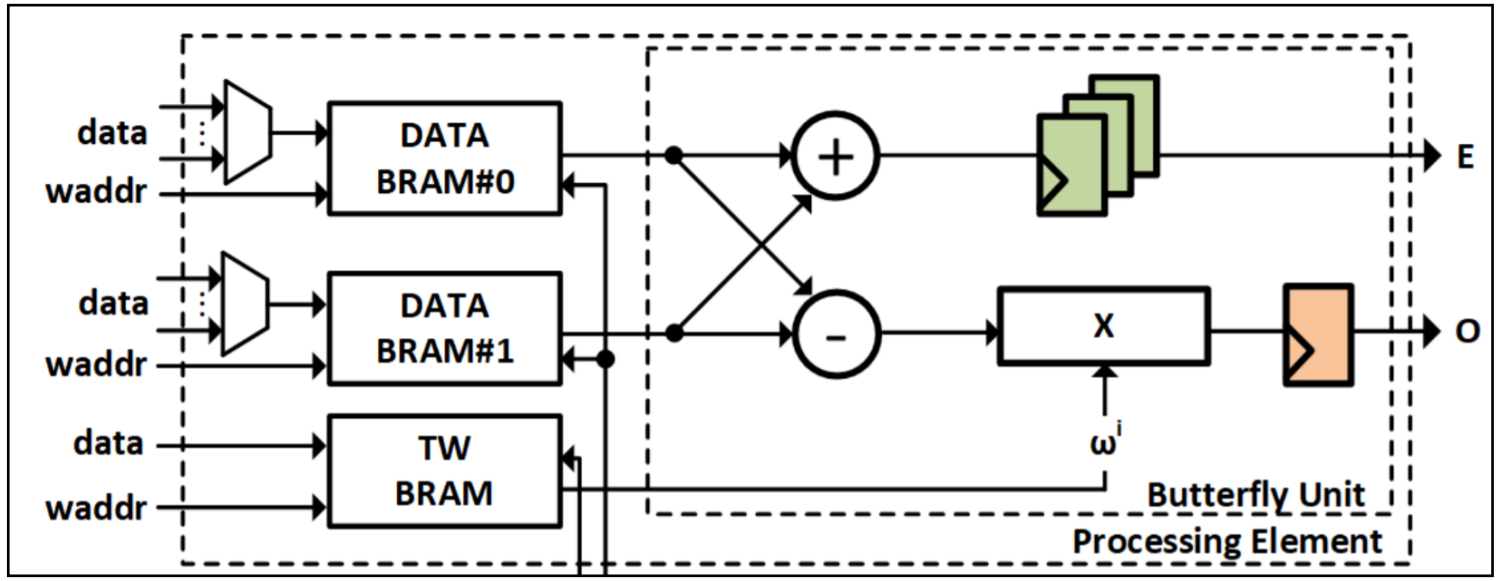
On to FHEW!

… now that you know all the blocks of an FPGA

FHEW algorithm:



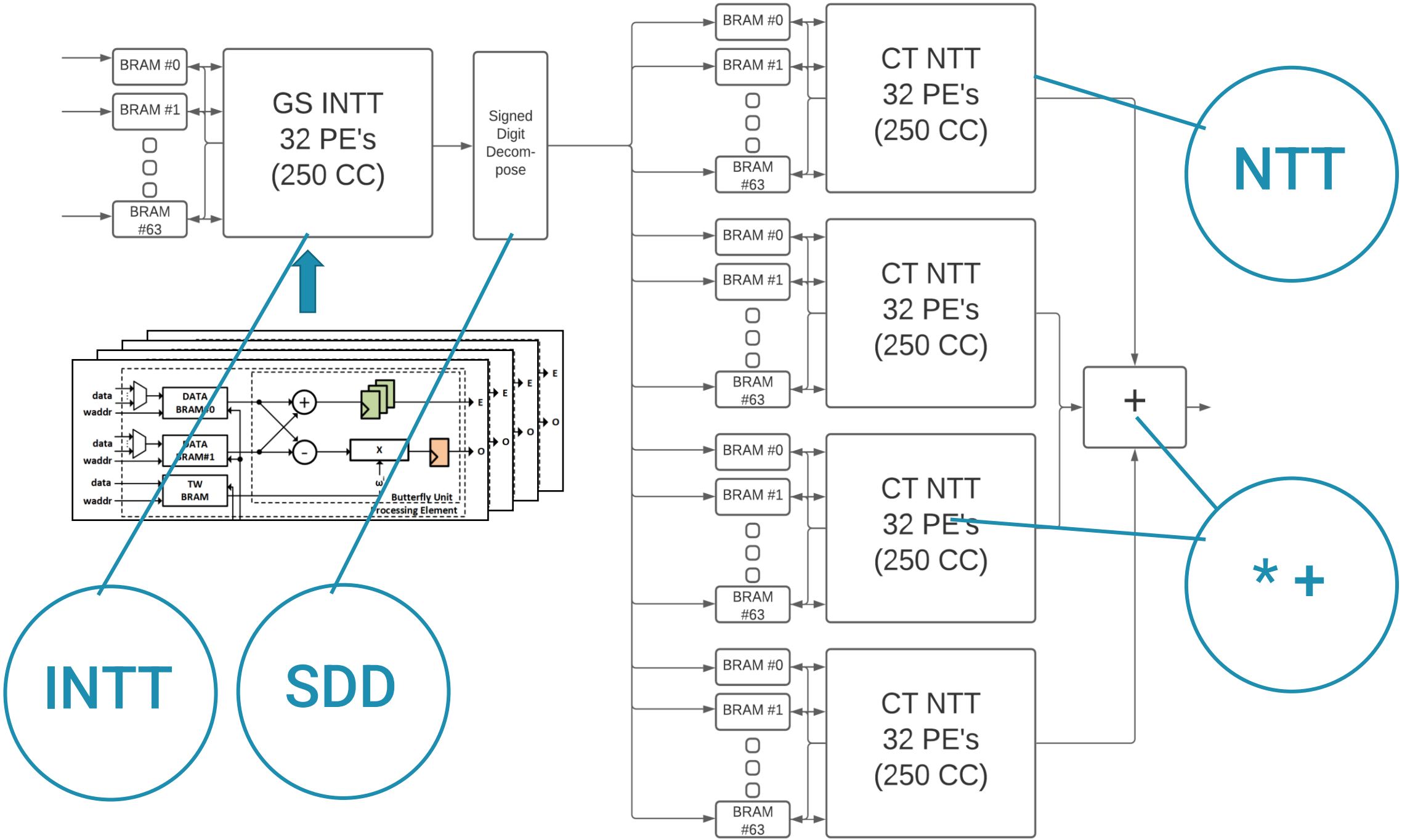
Mert et al.’s NTT design



A number of Processing Elements (PE) in parallel

Best open-source area/throughput NTT design in 2021

Very modular design, designed for Lattice-based crypto, not for FHE


BRAM #0
BRAM #1
BRAM #63
GS INTT
32 PE's
(250 CC)
Signed
Digit
Decom-
pose
data
waddr
DATA
BRAM#0
DATA
BRAM#1
TW
BRAM
X
ω
E
O
Butterfly Unit
Processing Element
CT NTT
32 PE's
(250 CC)
+
NTT
* +
INTT
SDD

That was the plan…

2 big problems: data size too big (scheme problem),
and NTT design was for small chips (hardware problem)

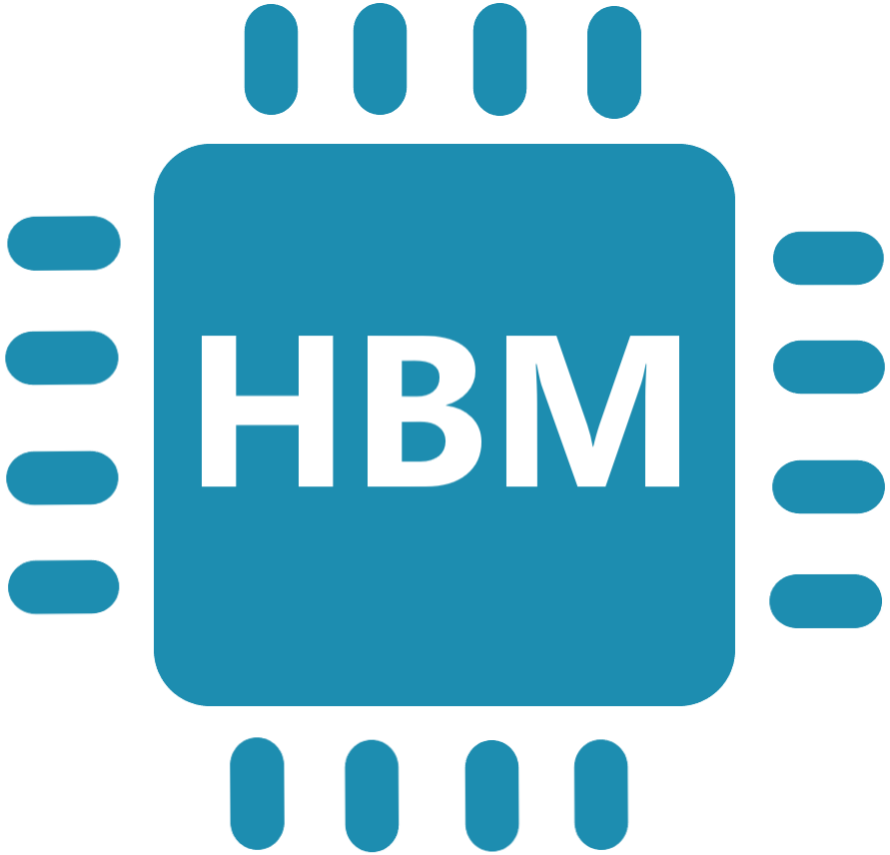
Problem 1: Bootstrapping key is too large!

ACC: made up of 2 vectors of size N=1024

a: vector of ***512*** values, indexes **BSK**, affects control

BootStrapping Key (BSK): ***512*****1024*****2*****23*****2*****2*****4*** values!

Problem 1: Bootstrapping key!



FHEW: 1.3 Gigabytes of bootstrapping key!

FPGA: 40 MB of memory

HBM: 8 GB @460 GB/s

Programming HBM = not easy!

Datapath design and debugging: 3 months

Making the HBM-interface work: 6 months and ongoing

The HBM interface must be efficient! 460 GB/s is not fast for 1.3 GB

The a-vector determines the indexing of the HBM interface

A naïve HBM interface will memory-bind you!

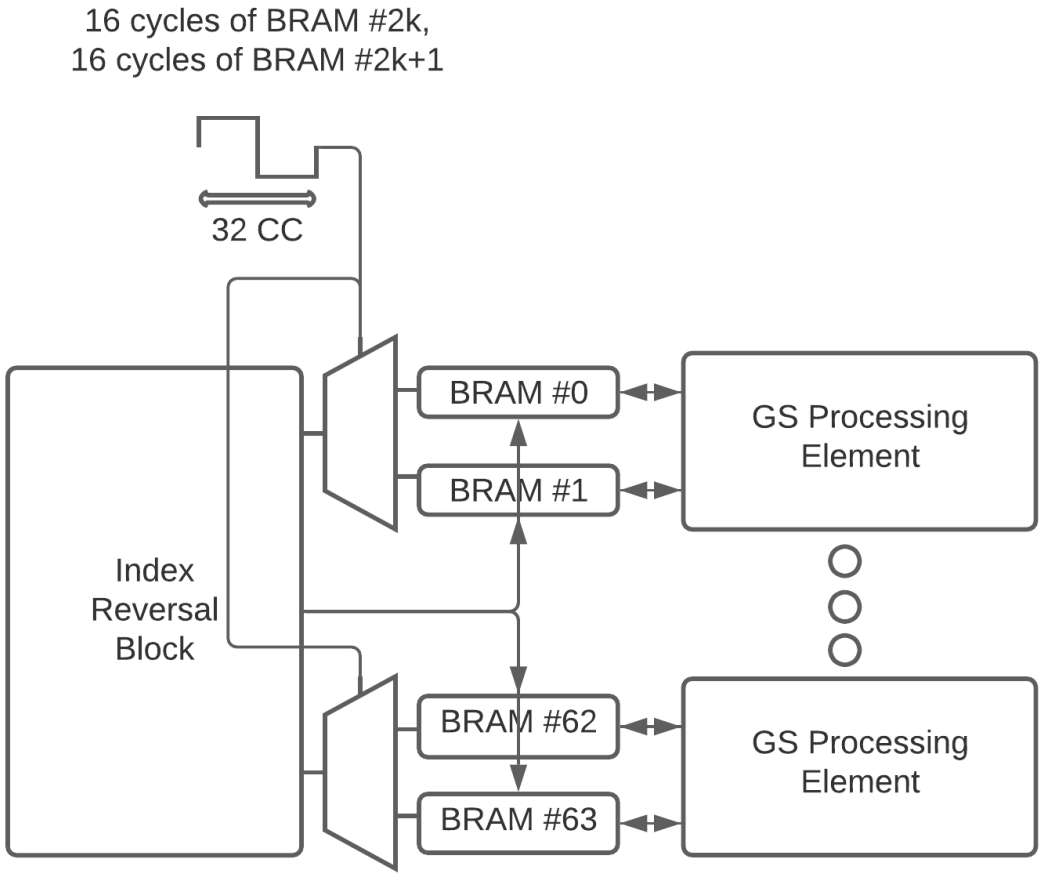
New paper!

Efficient FHEW Bootstrapping with Small Evaluation Keys, and Applications to Threshold Homomorphic Encryption

Yongwoo Lee[1], Daniele Micciancio[2], Andrey Kim[1], Rakyong Choi[1], Maxim Deryabin[1], Jieun Eom[1], and Donghoon Yoo[1]

Parameter set	Method	Runtime [ms]	Key size [MB]	Fail. prob.
128_Ours/AP	Alg. 7	**80.1**	**12.67**	$2^{-43.51}$
128_Ours/AP	AP	127.8	776.45	$2^{-40.22}$
128_tGINX	GINX*	89.7	40.45	$2^{-35.79}$
128_bGINX	GINX	84.1	20.91	$2^{-37.26}$

Problem 2 (Hardware problem)



We are targeting high throughput on big chips

Mert's design assumed relatively small chips (or lots of work per NTT)

We must compute lots of permutations

Results:

Task	Input	Add $a_i * s_i$ to **ACC**	Output	1/16th Bootstrap
µs	20.52	36.16	20.49	1143.27

TABLE 4.1: Simulation run time

HW takes 17 ms (in simulation)

SW takes 137 ms

Future work:

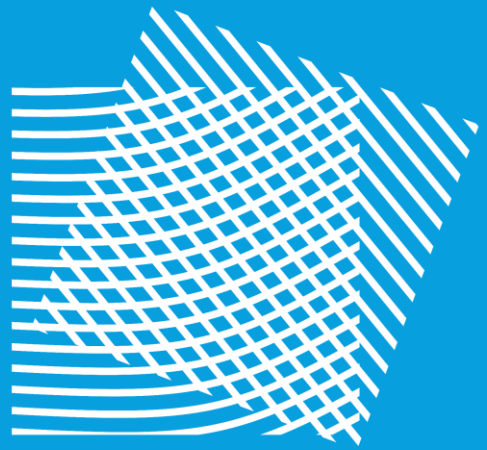
Parameter set	Method	Runtime [ms]	Key size [MB]	Fail. prob.
128_Ours/AP	Alg. 7	**80.1**	**12.67**	$2^{-43.51}$
128_Ours/AP	AP	127.8	776.45	$2^{-40.22}$
128_tGINX	GINX*	89.7	40.45	$2^{-35.79}$
128_bGINX	GINX	84.1	20.91	$2^{-37.26}$

Use a scheme that doesn't require handling >40 MB's

Better NTT's, designed for the parameter set of FHEW/TFHE/FINAL
(We can do much better than this, you will see...)

…so much improvement still possible!

COSIC WEB



COSIC

PAPER