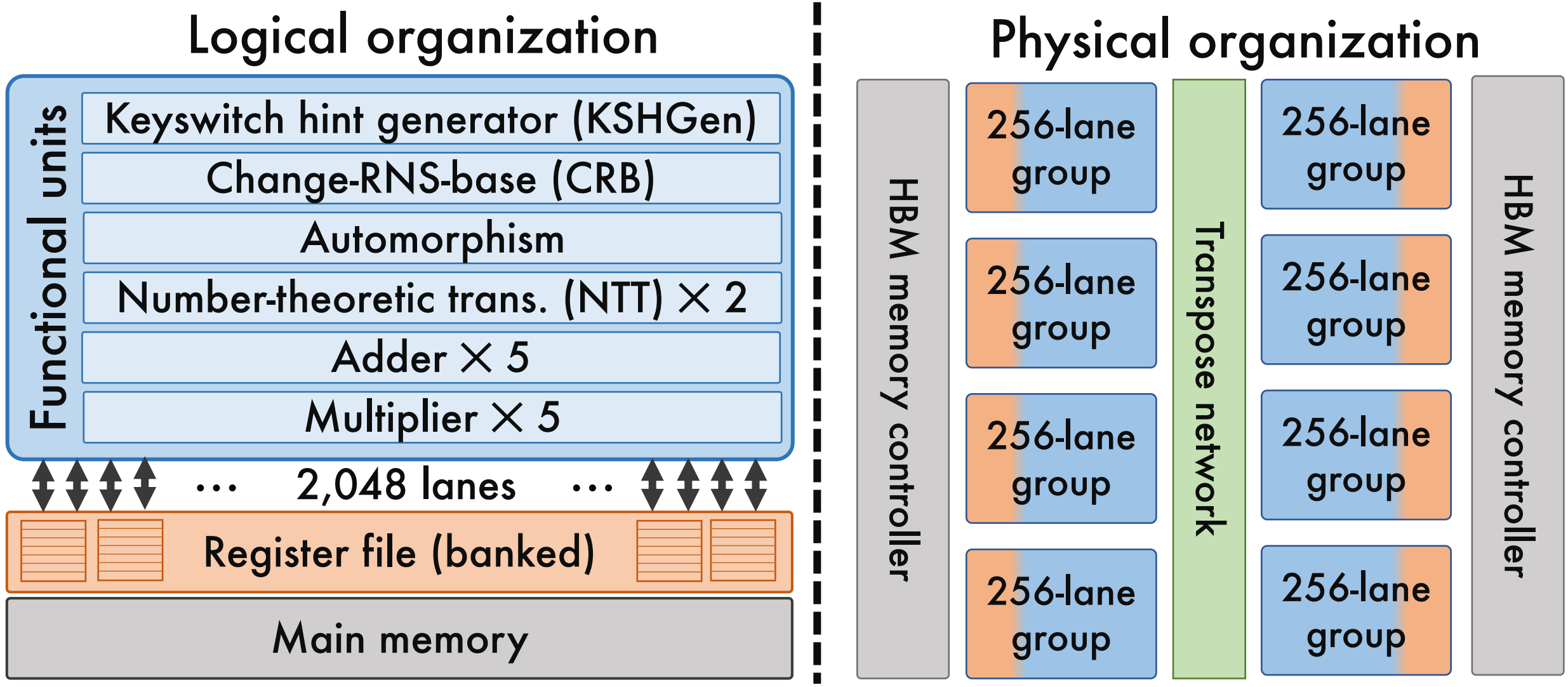
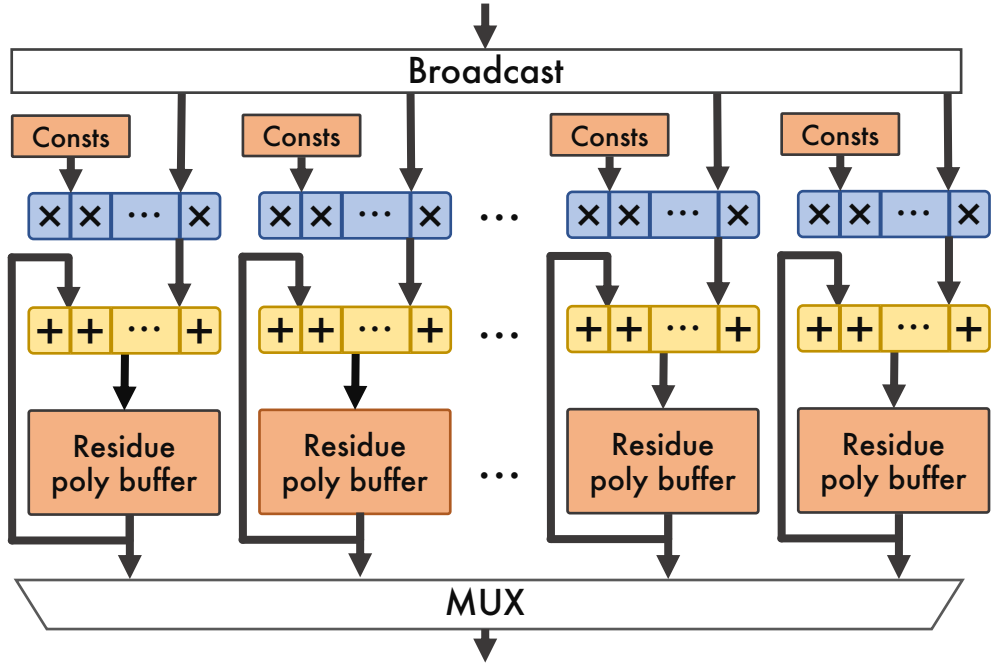
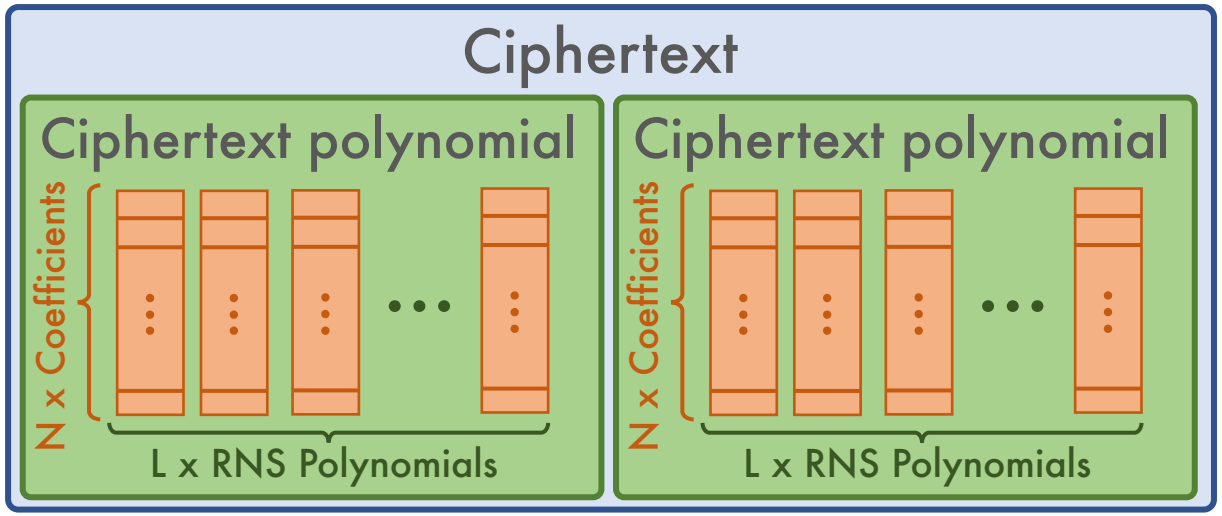
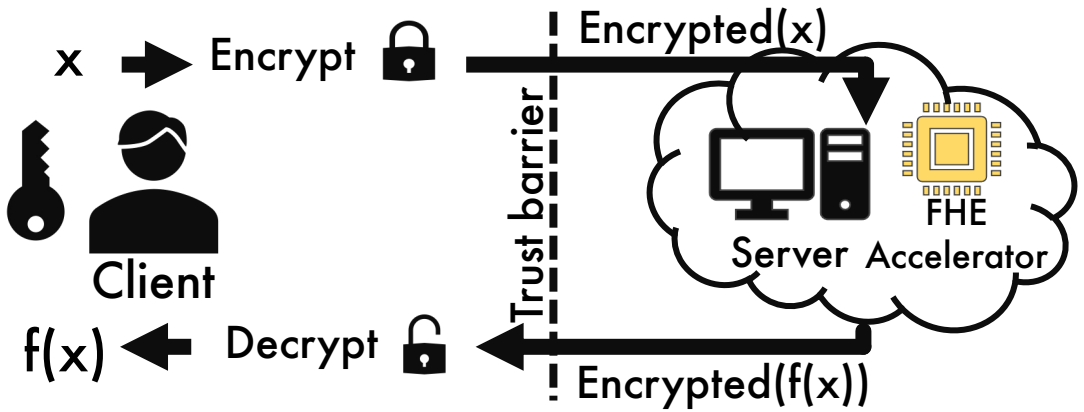
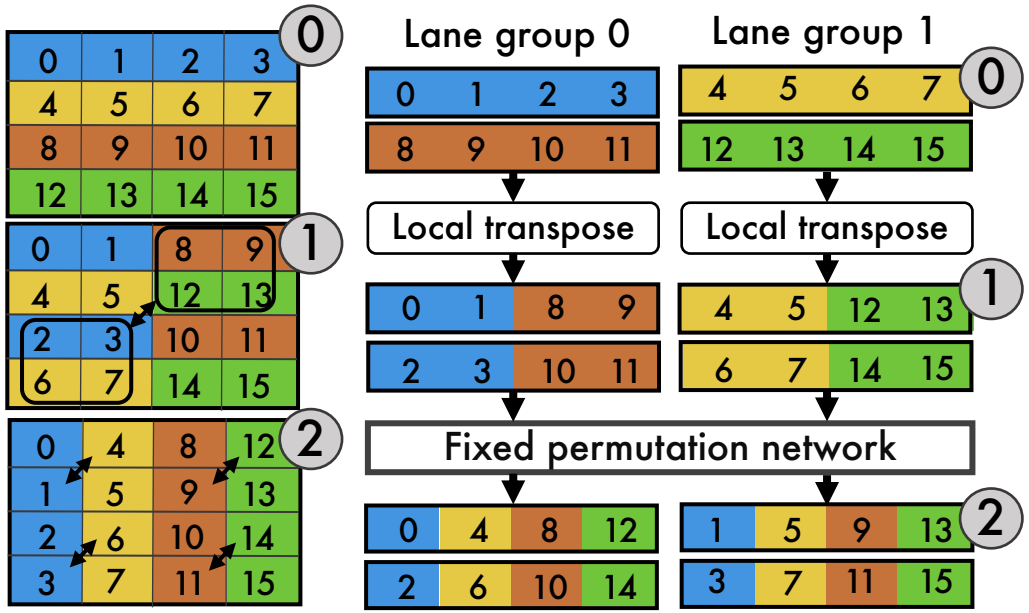

Logical organization
Functional units
Keyswitch hint generator (KSHGen)
Change-RNS-base (CRB)
Automorphism
Number-theoretic trans. (NTT) × 2
Adder × 5
Multiplier × 5
… 2,048 lanes …
Register file (banked)
Main memory
Physical organization
HBM memory controller
256-lane group
256-lane group
256-lane group
256-lane group
Transpose network
256-lane group
256-lane group
256-lane group
256-lane group
HBM memory controller

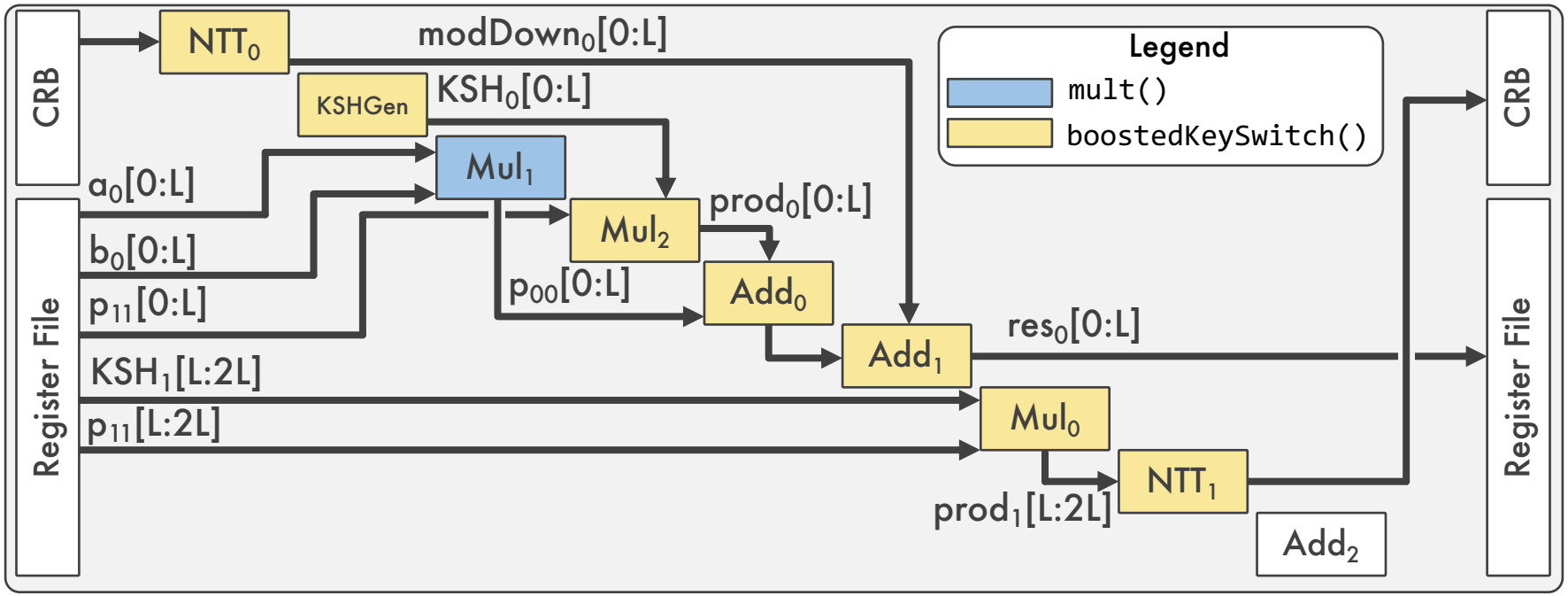

Broadcast
Consts
Consts
Consts
Consts
Residue poly buffer
Residue poly buffer
Residue poly buffer
Residue poly buffer
MUX

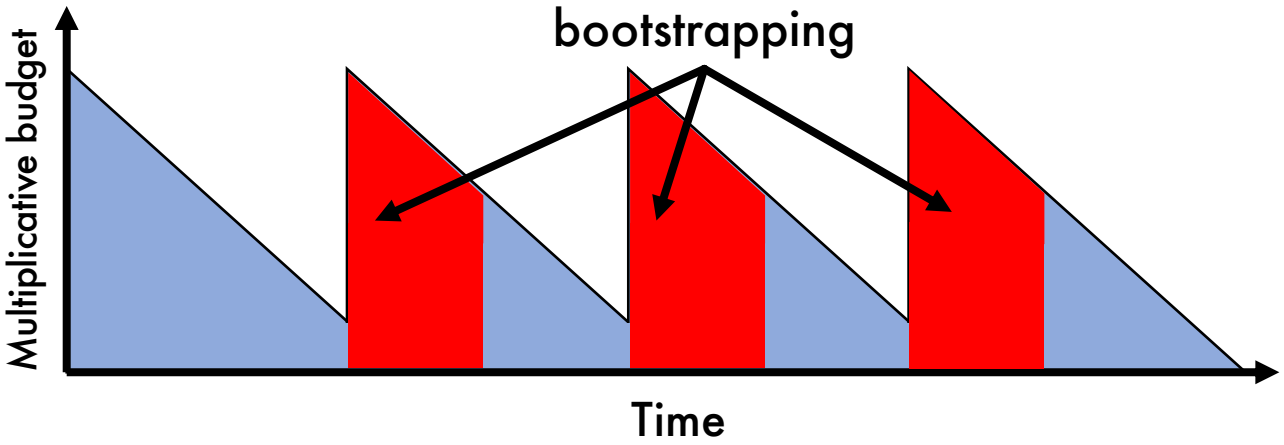

Ciphertext
Ciphertext polynomial
N x Coefficients
L x RNS Polynomials
Ciphertext polynomial
N x Coefficients
L x RNS Polynomials

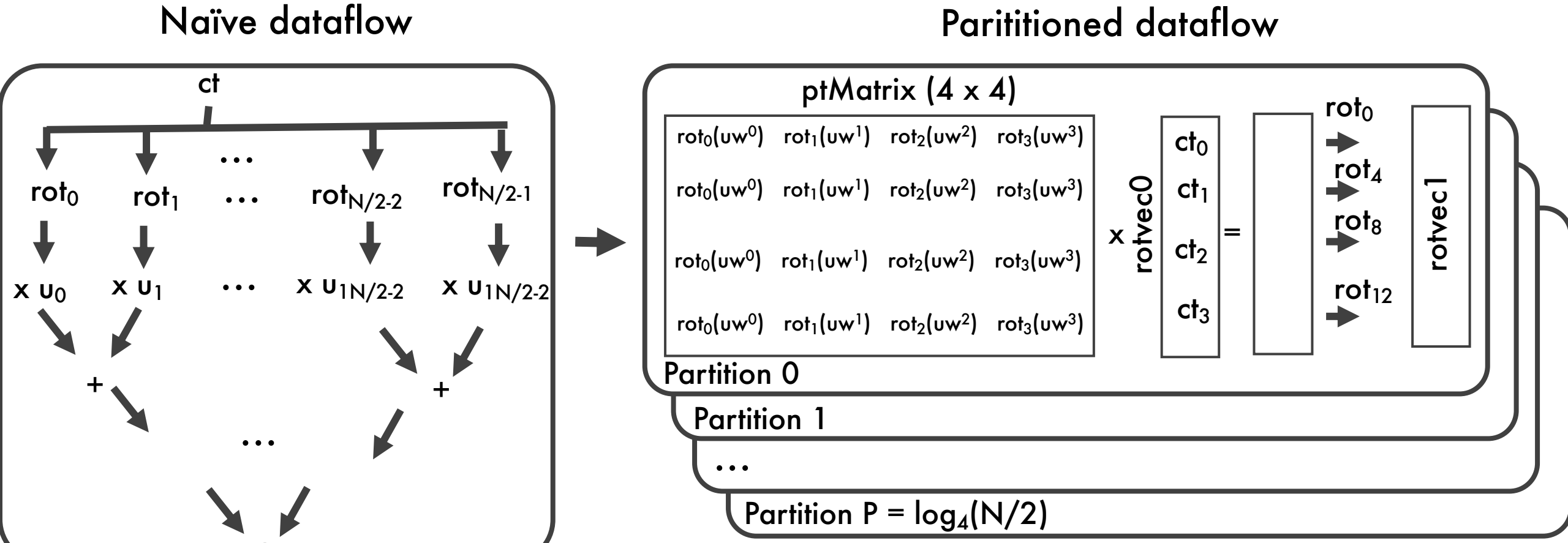

x
Encrypt
Encrypted(x)
Trust barrier
Client
Server
FHE
Accelerator
f(x)
Decrypt
Encrypted(f(x))

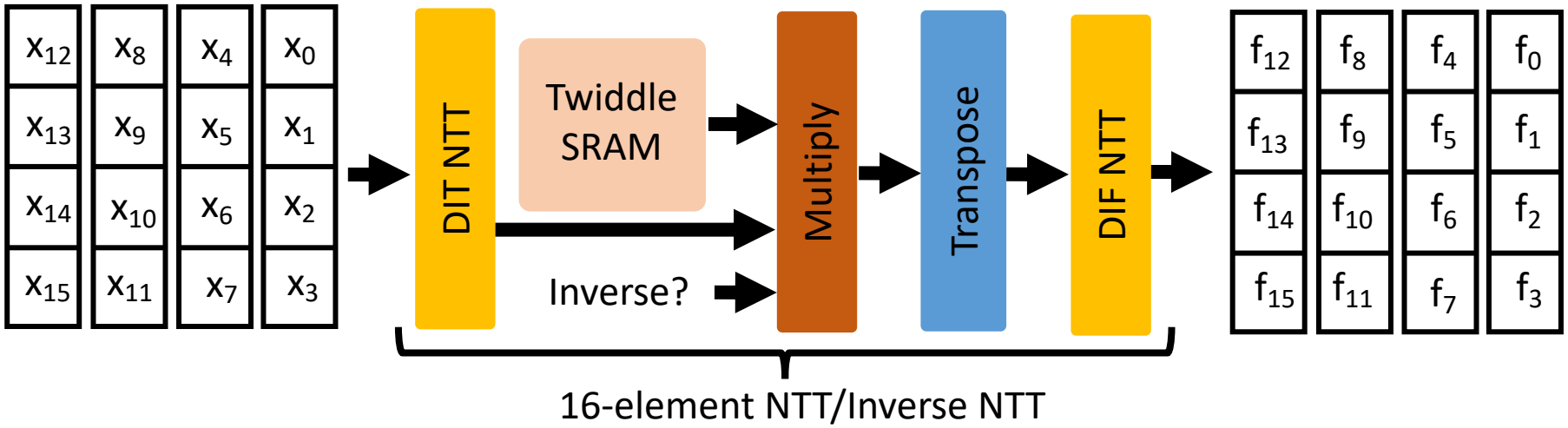
```
1  def mult((a0[0:L], a1[0:L]), (b0[0:L], b1[0:L])):
2    p00[0:L] = a0[0:L] * b0[0:L]
3    p01+10[0:L] = a0[0:L]*b1[0:L] + a1[0:L]*b0[0:L]
4    (ks0, ks1) = KS_new(a1[0:L] * b1[0:L])
5    res0[0:L] = p00[0:L] + ks0[0:L]
6    res1[0:L] = p01+10[0:L] + ks1[0:L]
7    return (Rescale(res0[0:L]), Rescale(res1[0:L]))
8
9  def KS_new(p11[0:L]):
10   p11[L:2L] = INTT_CRB_NTT(p11[0:L], [L:2L])
11   for i = 0, 1:
12     prodi[0:2L] = p11[0:2L] * KSHi[0:2L]
13     modDowni[0:L] = INTT_CRB_NTT(prodi[L:2L], [0:L])
14     ksi[0:L] = prodi[0:L] + modDowni[0:L]
15   return (ks0[0:L], ks1[0:L])
16
17 def Rescale(resi[0:L]):
18   xINTTi = INNT(resi[L-1], L-1)
19   subMei[0:L-1] = [NTT(xINTTi, j) for j in [0:L-1]]
20   return resi[0:L-1] - subMei[0:L-1]
```

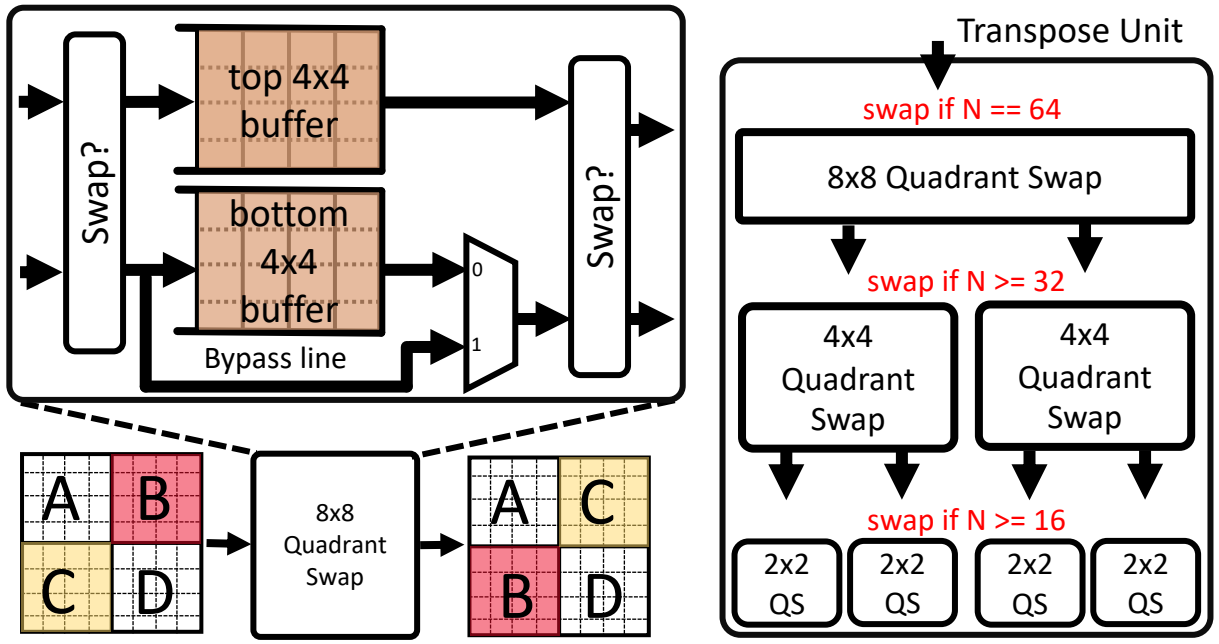

0
1
2
Lane group 0
Lane group 1
Local transpose
Local transpose
Fixed permutation network

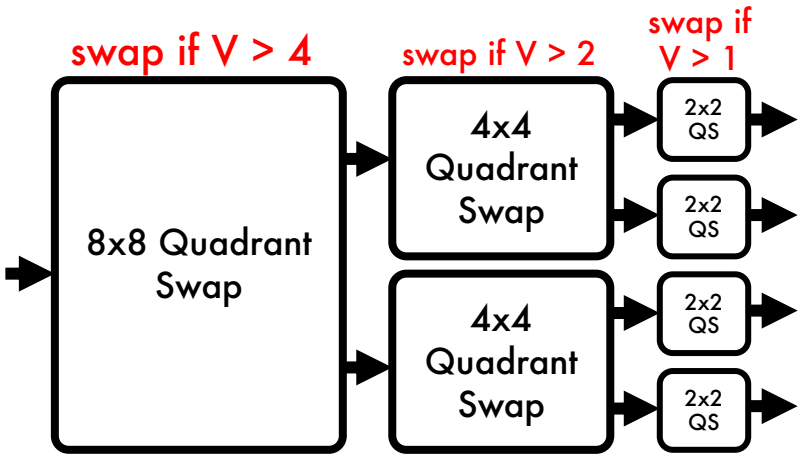

CRB
Register File
NTT0
modDown0[0:L]
KSHGen
KSH0[0:L]
Mul1
a0[0:L]
b0[0:L]
p11[0:L]
Mul2
prod0[0:L]
p00[0:L]
Add0
Add1
res0[0:L]
KSH1[L:2L]
p11[L:2L]
Mul0
NTT1
prod1[L:2L]
Add2
Legend
mult()
boostedKeySwitch()
CRB
Register File

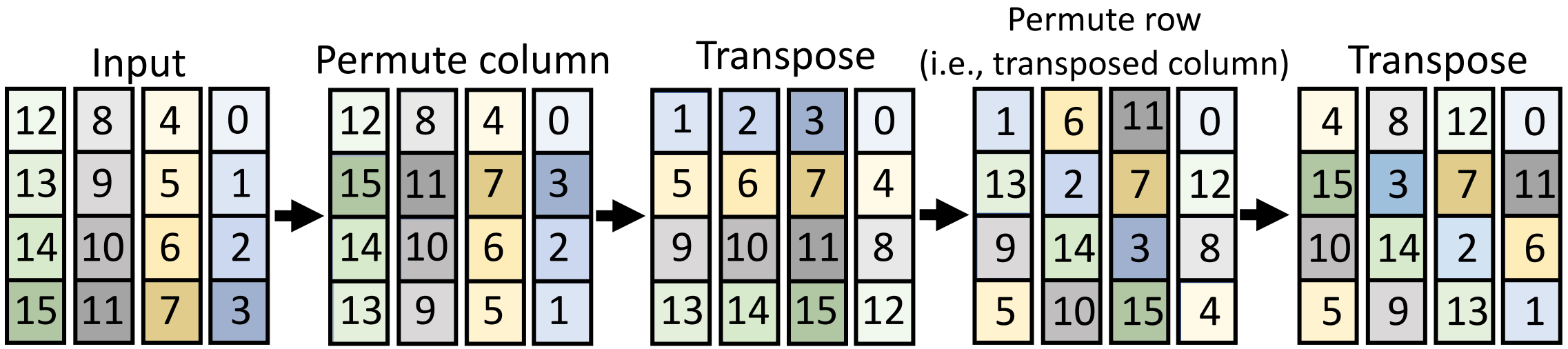

bootstrapping
Multiplicative budget
Time

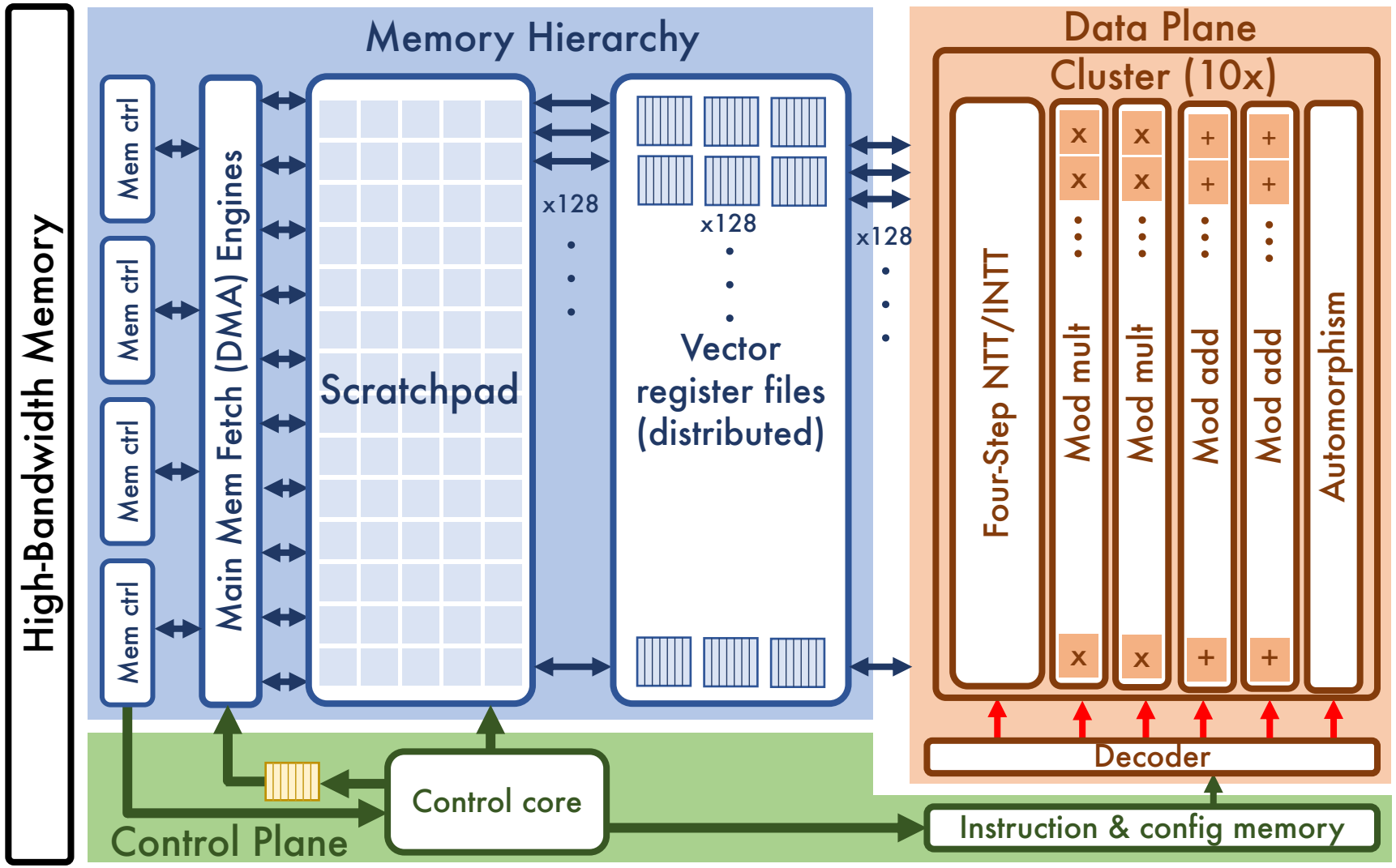

Naïve dataflow
ct
rot0
rot1
rotN/2-2
rotN/2-1
x u0
x u1
x u1N/2-2
x u1N/2-2
+
+
+
Paritititioned dataflow
ptMatrix (4 x 4)
rot0(uw0) rot1(uw1) rot2(uw2) rot3(uw3)
rot0(uw0) rot1(uw1) rot2(uw2) rot3(uw3)
rot0(uw0) rot1(uw1) rot2(uw2) rot3(uw3)
rot0(uw0) rot1(uw1) rot2(uw2) rot3(uw3)
x
rotvec0
ct0
ct1
ct2
ct3
=
rot0
rot4
rot8
rot12
rotvec1
Partition 0
Partition 1
Partition P = log4(N/2)

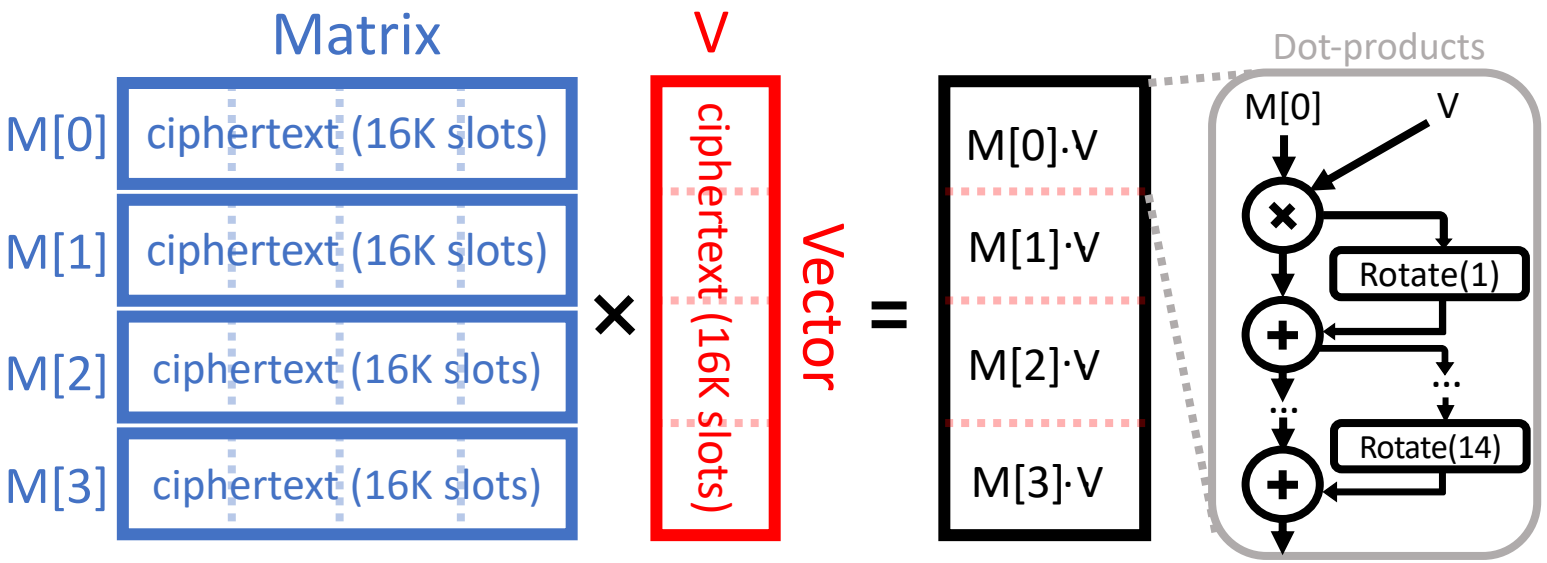

x12 x8 x4 x0
x13 x9 x5 x1
x14 x10 x6 x2
x15 x11 x7 x3
DIT NTT
Twiddle SRAM
Inverse?
Multiply
Transpose
DIF NTT
f12 f8 f4 f0
f13 f9 f5 f1
f14 f10 f6 f2
f15 f11 f7 f3
16-element NTT/Inverse NTT

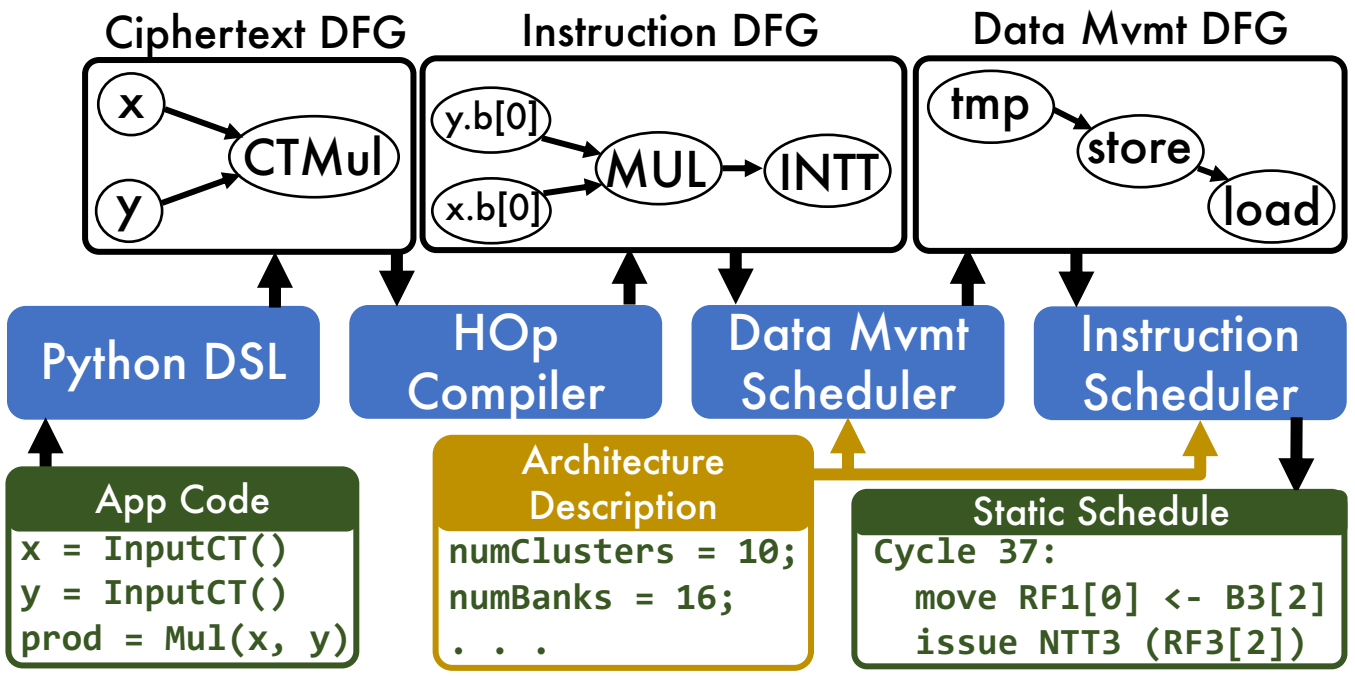

Swap?
top 4x4 buffer
bottom 4x4 buffer
Bypass line
0
1
Swap?
A
B
C
D
8x8 Quadrant Swap
A
C
B
D
Transpose Unit
swap if N == 64
8x8 Quadrant Swap
swap if N >= 32
4x4 Quadrant Swap
4x4 Quadrant Swap
swap if N >= 16
2x2 QS
2x2 QS
2x2 QS
2x2 QS

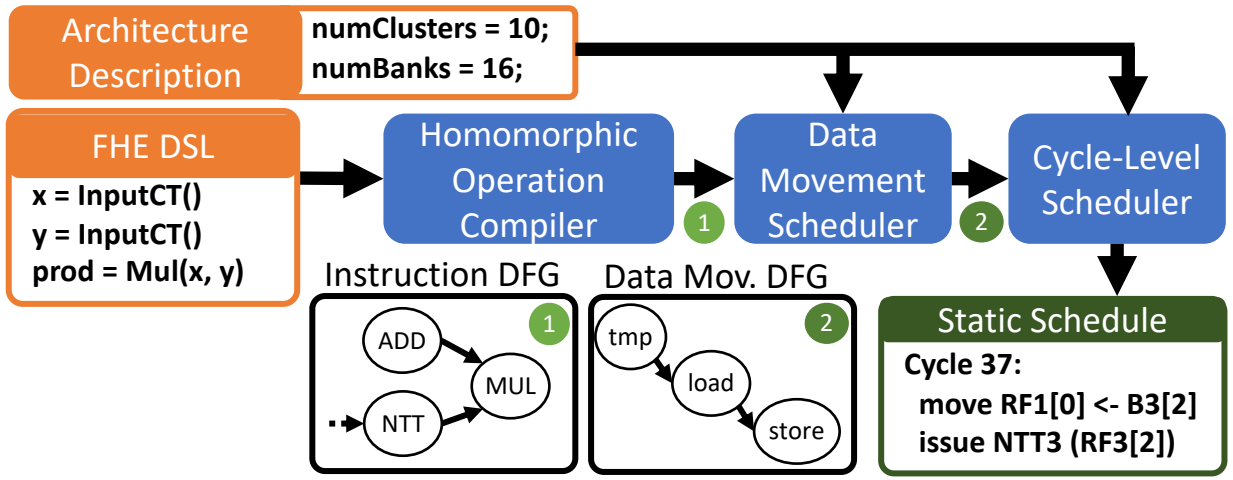

swap if V > 4
swap if V > 2
swap if V > 1
8x8 Quadrant Swap
4x4 Quadrant Swap
4x4 Quadrant Swap
2x2 QS
2x2 QS
2x2 QS
2x2 QS

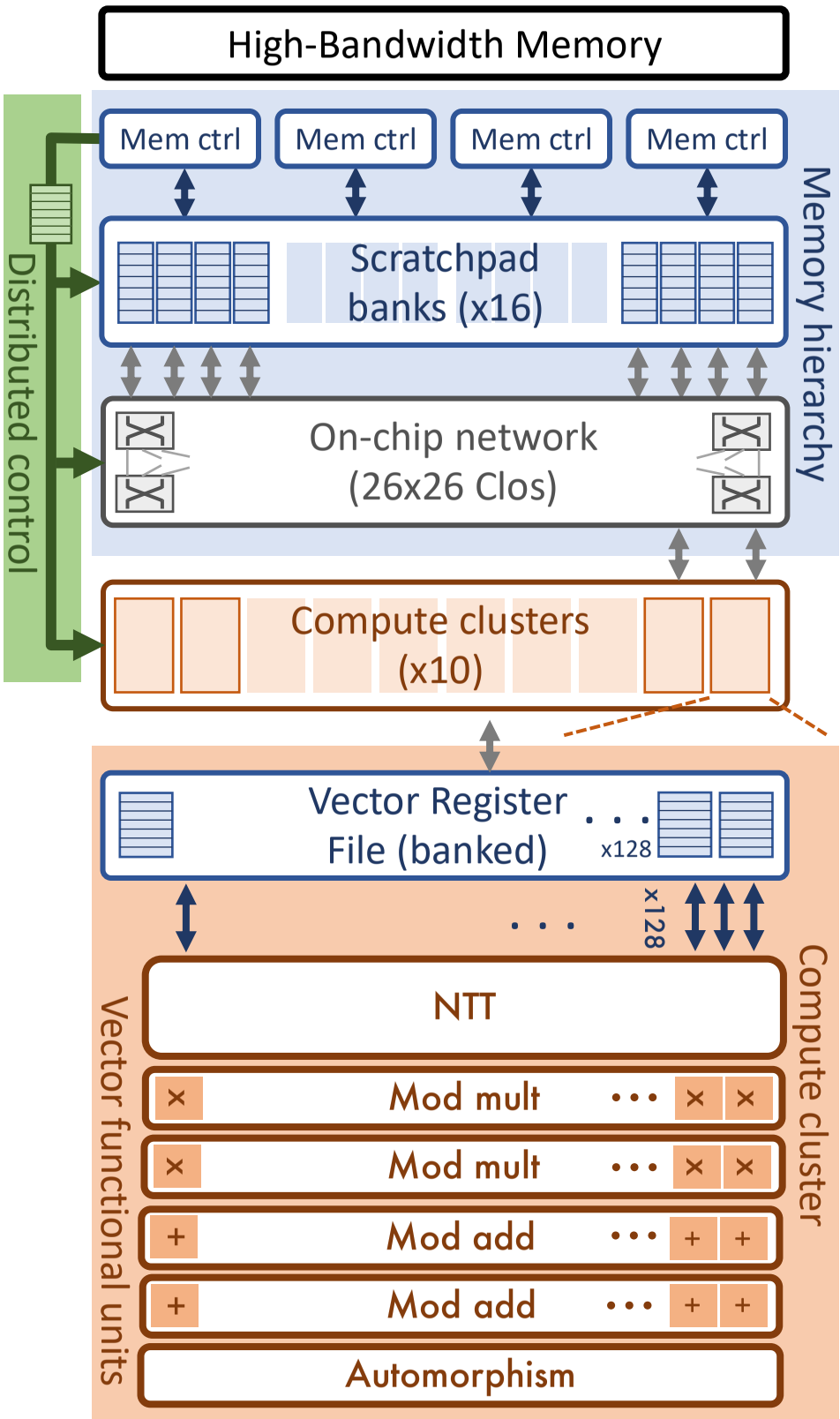

Input
12 8 4 0
13 9 5 1
14 10 6 2
15 11 7 3
Permute column
12 8 4 0
15 11 7 3
14 10 6 2
13 9 5 1
Transpose
1 2 3 0
5 6 7 4
9 10 11 8
13 14 15 12
Permute row
(i.e., transposed column)
1 6 11 0
13 2 7 12
9 14 3 8
5 10 15 4
Transpose
4 8 12 0
15 3 7 11
10 14 2 6
5 9 13 1

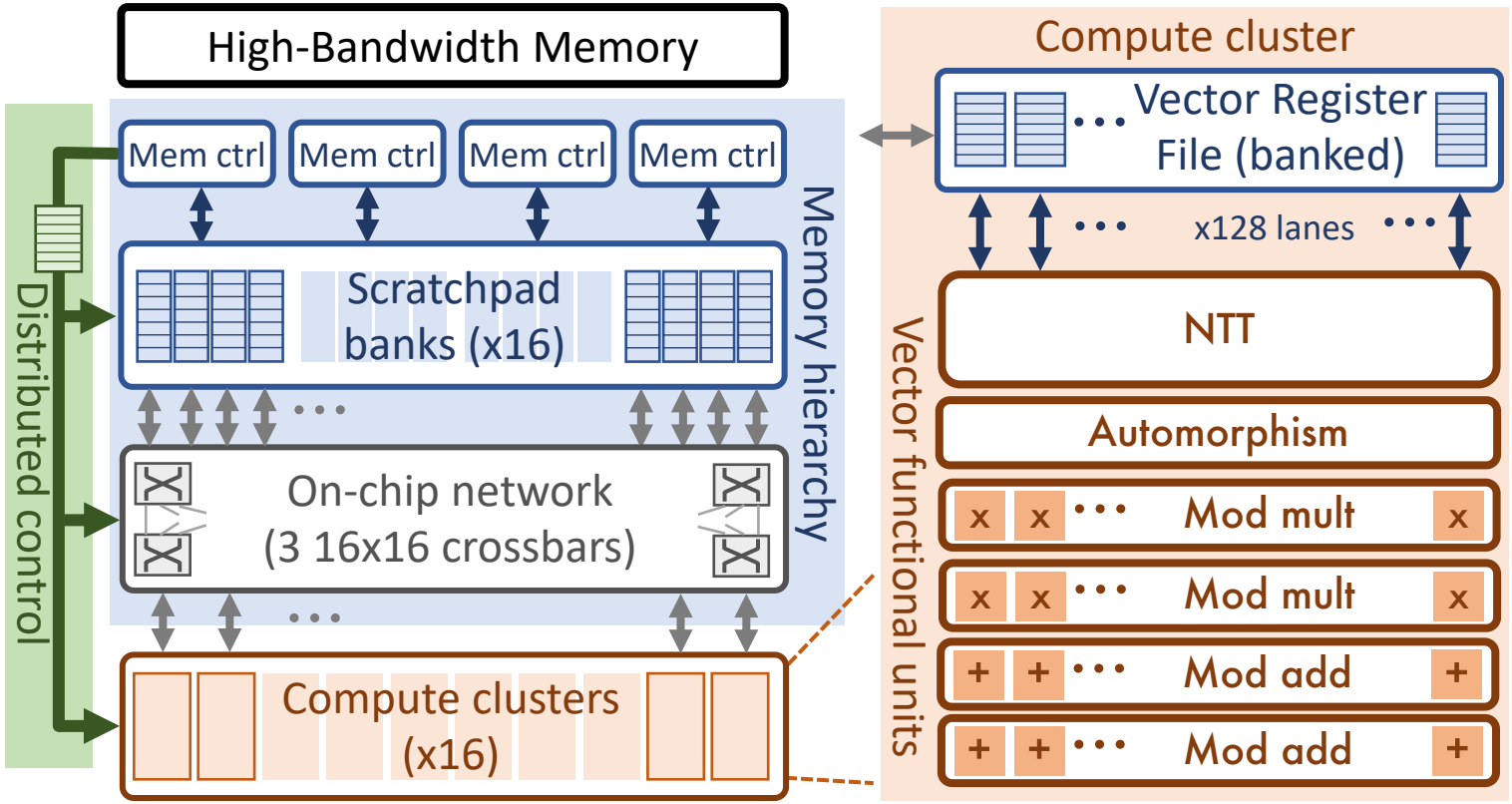

High-Bandwidth Memory
Memory Hierarchy
Mem ctrl
Mem ctrl
Mem ctrl
Mem ctrl
Main Mem Fetch (DMA) Engines
Scratchpad
x128
x128
Vector register files (distributed)
x128
Data Plane
Cluster (10x)
Four-Step NTT/INTT
Mod mult
Mod mult
Mod add
Mod add
Automorphism
Decoder
Control Plane
Control core
Instruction & config memory

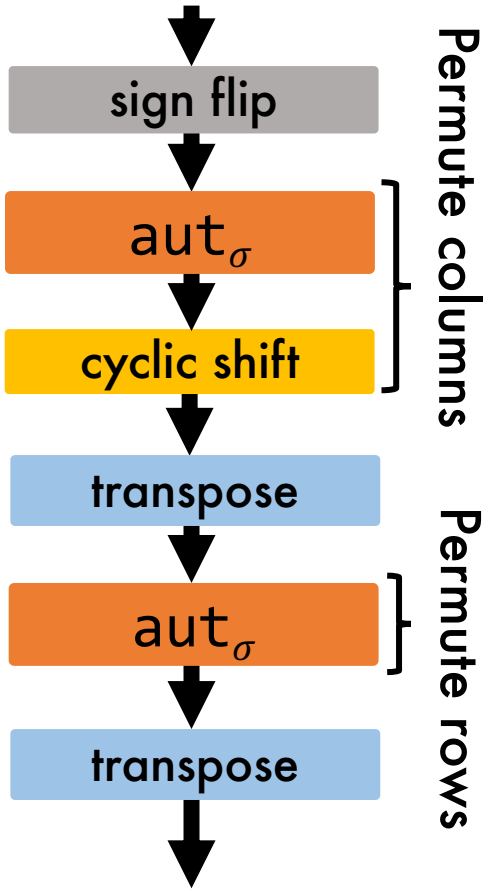

Matrix
V
M[0]
ciphertext (16K slots)
M[1]
ciphertext (16K slots)
M[2]
ciphertext (16K slots)
M[3]
ciphertext (16K slots)
×
ciphertext (16K slots)
Vector
=
M[0]·V
M[1]·V
M[2]·V
M[3]·V
Dot-products
M[0]
V
Rotate(1)
Rotate(14)

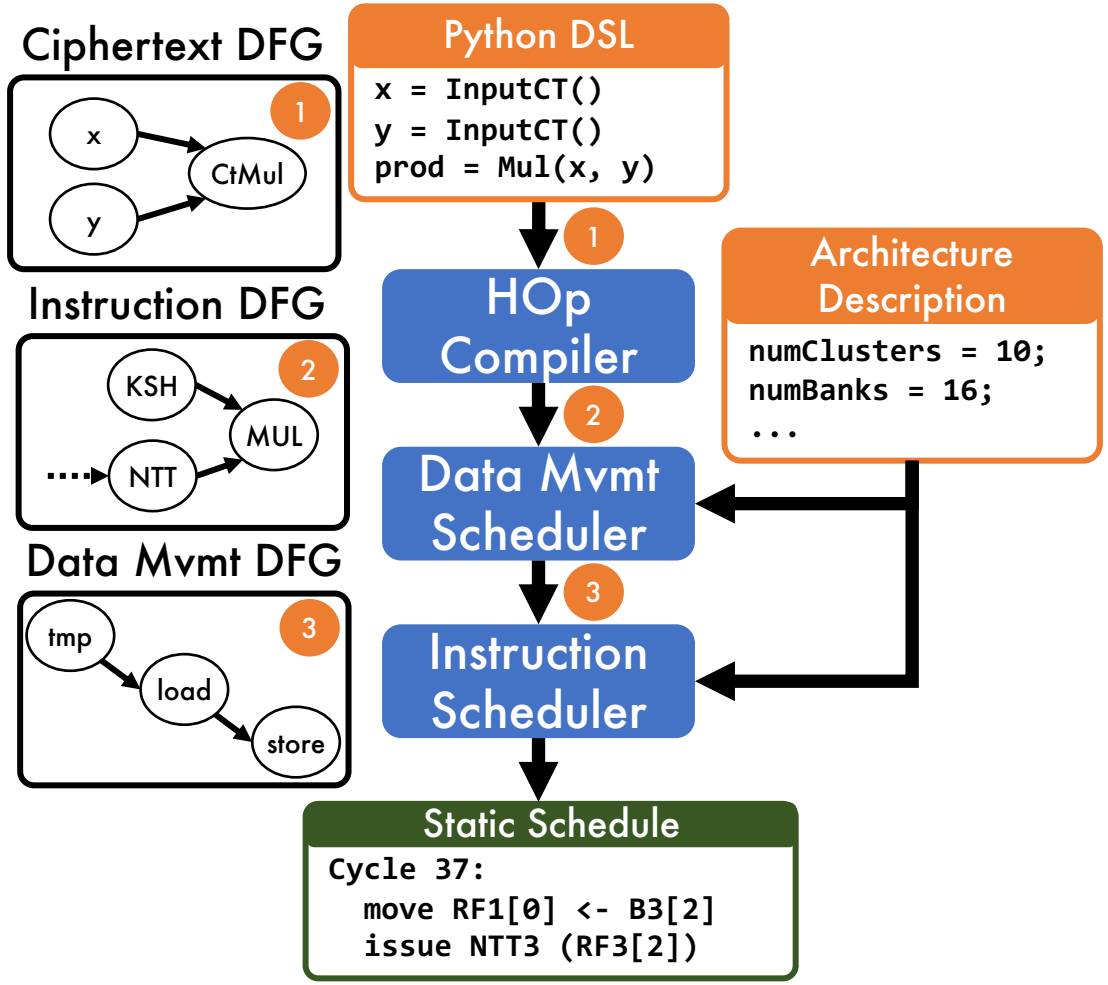

Ciphertext DFG
Instruction DFG
Data Mvmt DFG
x
y
CTMul
y.b[0]
x.b[0]
MUL
INTT
tmp
store
load
Python DSL
HOp Compiler
Data Mvmt Scheduler
Instruction Scheduler
App Code
x = InputCT()
y = InputCT()
prod = Mul(x, y)
Architecture Description
numClusters = 10;
numBanks = 16;
. . .
Static Schedule
Cycle 37:
move RF1[0] <- B3[2]
issue NTT3 (RF3[2])

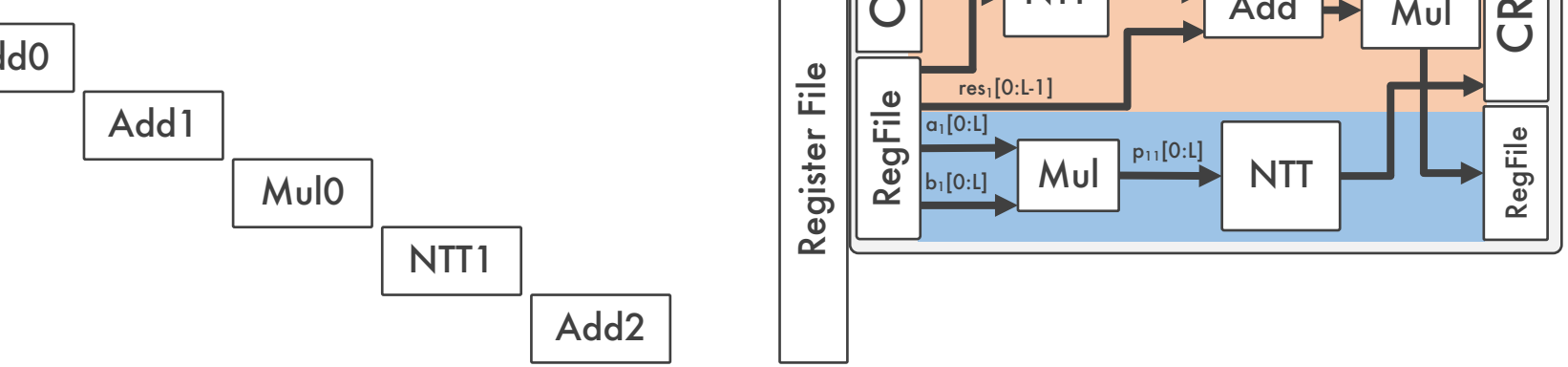

Architecture Description
numClusters = 10;
numBanks = 16;
FHE DSL
x = InputCT()
y = InputCT()
prod = Mul(x, y)
Homomorphic Operation Compiler
1
Data Movement Scheduler
2
Cycle-Level Scheduler
Instruction DFG
ADD
MUL
NTT
1
Data Mov. DFG
tmp
load
store
2
Static Schedule
Cycle 37:
move RF1[0] <- B3[2]
issue NTT3 (RF3[2])

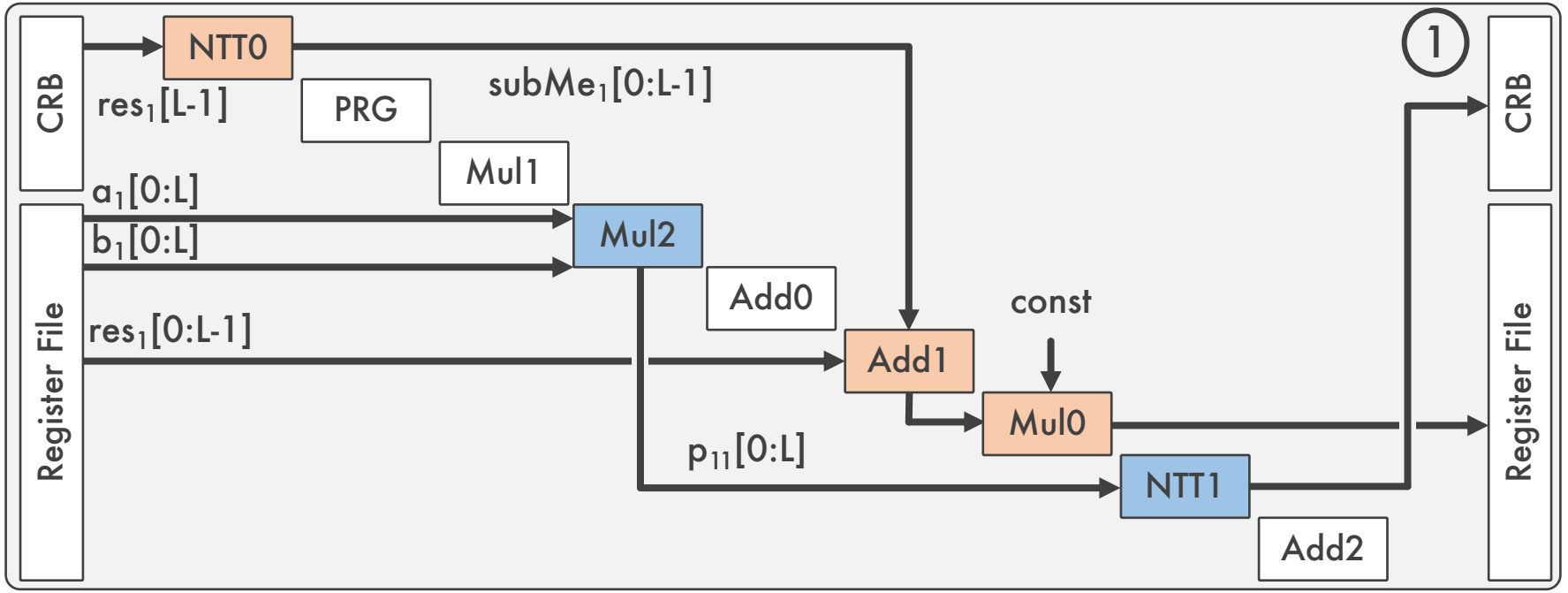

High-Bandwidth Memory
Mem ctrl
Mem ctrl
Mem ctrl
Mem ctrl
Memory hierarchy
Distributed control
Scratchpad banks (x16)
On-chip network (26x26 Clos)
Compute clusters (x10)
Vector Register File (banked)
x128
x128
Compute cluster
Vector functional units
NTT
Mod mult
Mod mult
Mod add
Mod add
Automorphism

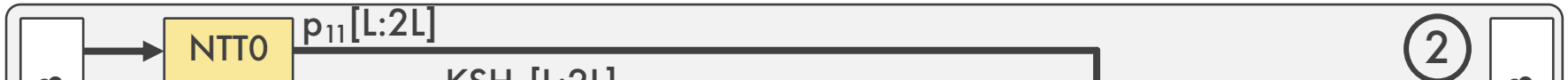

High-Bandwidth Memory
Mem ctrl
Mem ctrl
Mem ctrl
Mem ctrl
Memory hierarchy
Distributed control
Scratchpad banks (x16)
On-chip network (3 16x16 crossbars)
Compute clusters (x16)
Compute cluster
Vector Register File (banked)
x128 lanes
NTT
Automorphism
Mod mult
Mod mult
Mod add
Mod add
Vector functional units

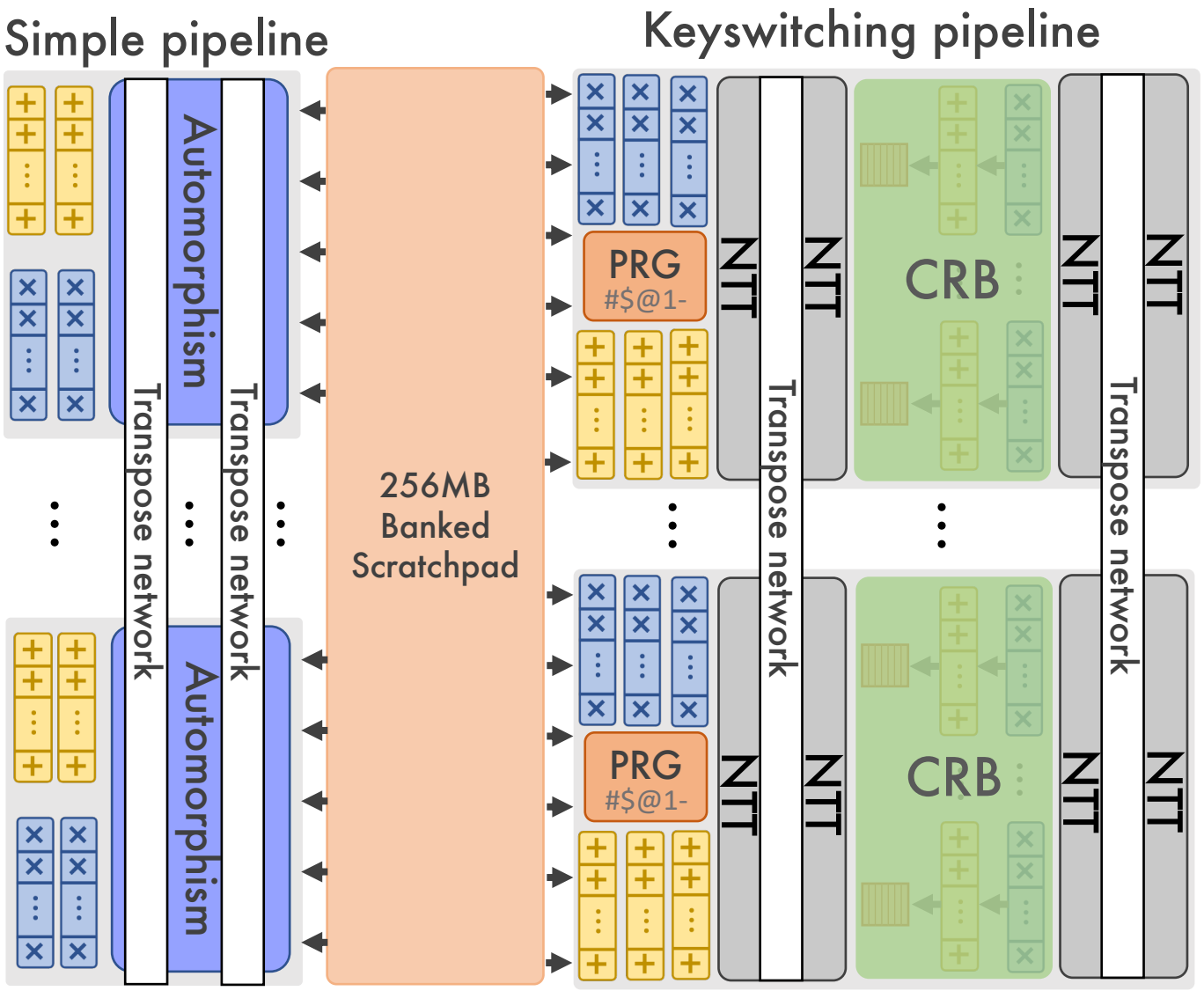

sign flip
aut_σ
cyclic shift
transpose
aut_σ
transpose
Permute columns
Permute rows

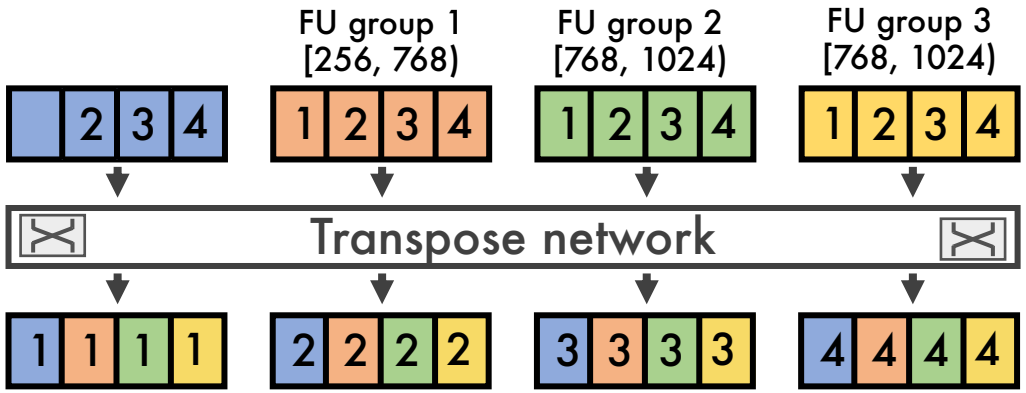

Ciphertext DFG
x
y
CtMul
1
Instruction DFG
KSH
MUL
NTT
2
Data Mvmt DFG
tmp
load
store
3
Python DSL
x = InputCT()
y = InputCT()
prod = Mul(x, y)
1
HOp Compiler
2
Data Mvmt Scheduler
3
Instruction Scheduler
Architecture Description
numClusters = 10;
numBanks = 16;
...
Static Schedule
Cycle 37:
move RF1[0] <- B3[2]
issue NTT3 (RF3[2])

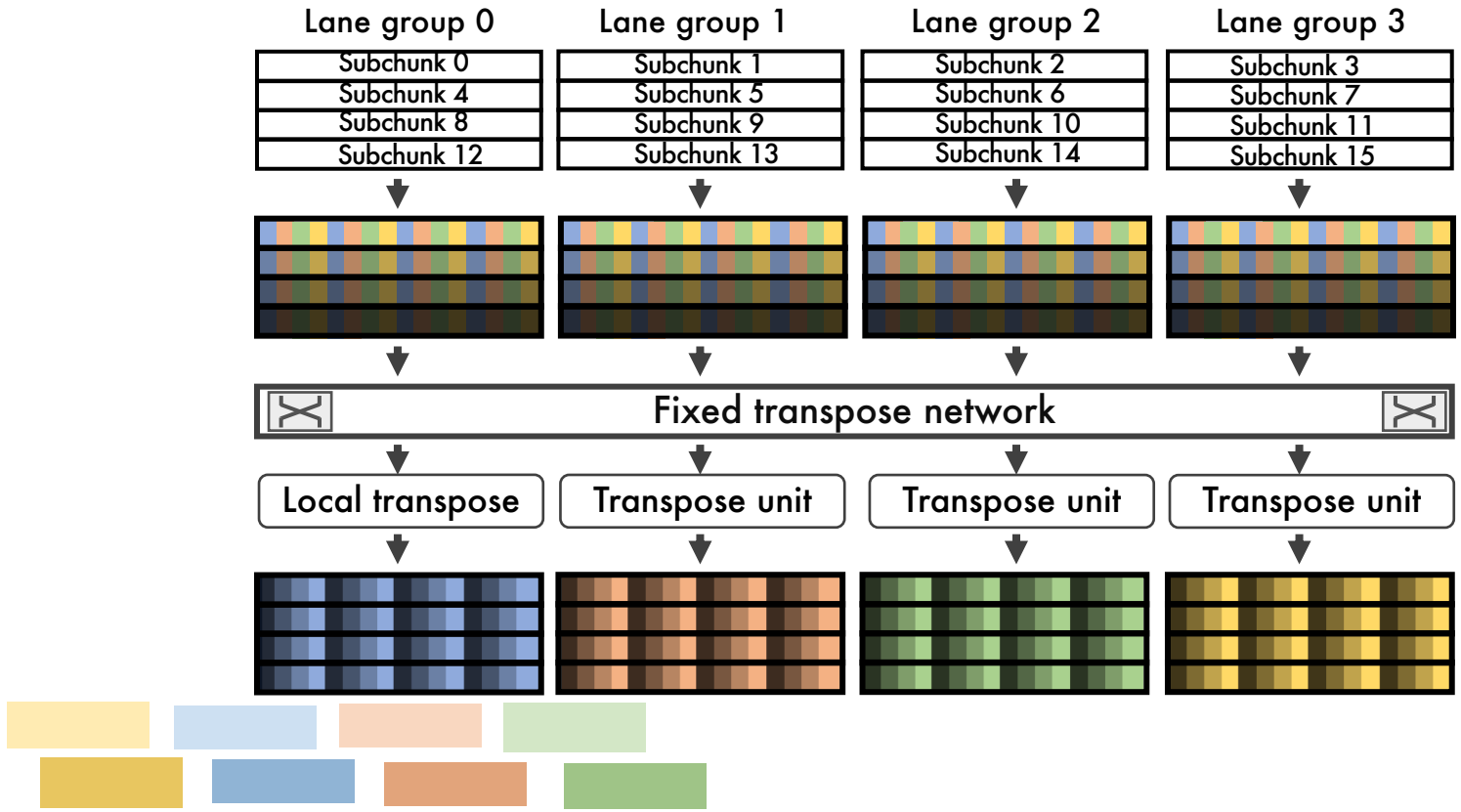

dd0
Add1
Mul0
NTT1
Add2
Register File
RegFile
$res_1[0:L-1]$
$a_1[0:L]$
$b_1[0:L]$
Mul
$p_{11}[0:L]$
NTT
Add
Mul
RegFile

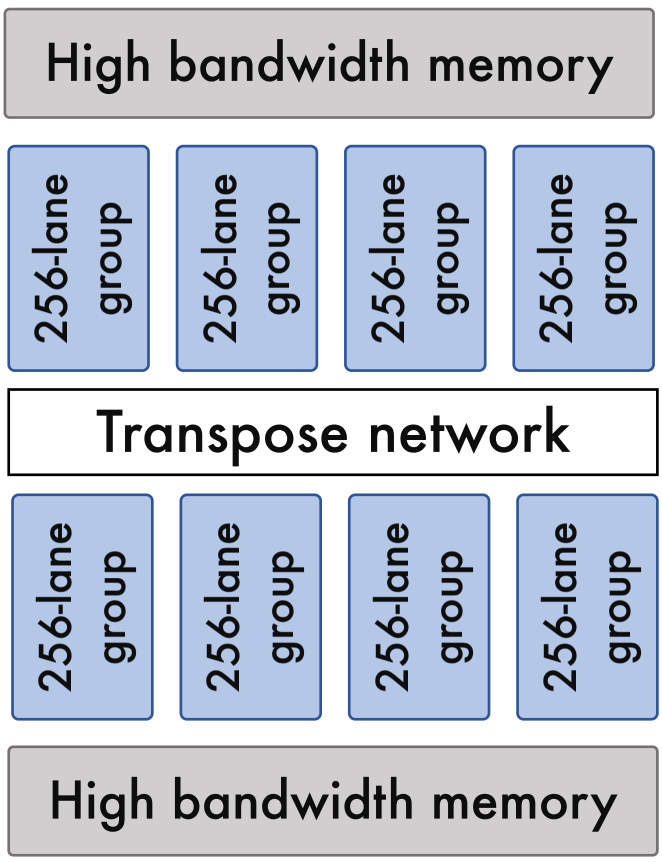

1
CRB
NTT0
$res_1[L-1]$
PRG
$subMe_1[0:L-1]$
Mul1
$a_1[0:L]$
$b_1[0:L]$
Mul2
Add0
const
$res_1[0:L-1]$
Add1
Mul0
$p_{11}[0:L]$
NTT1
Add2
Register File
CRB
Register File

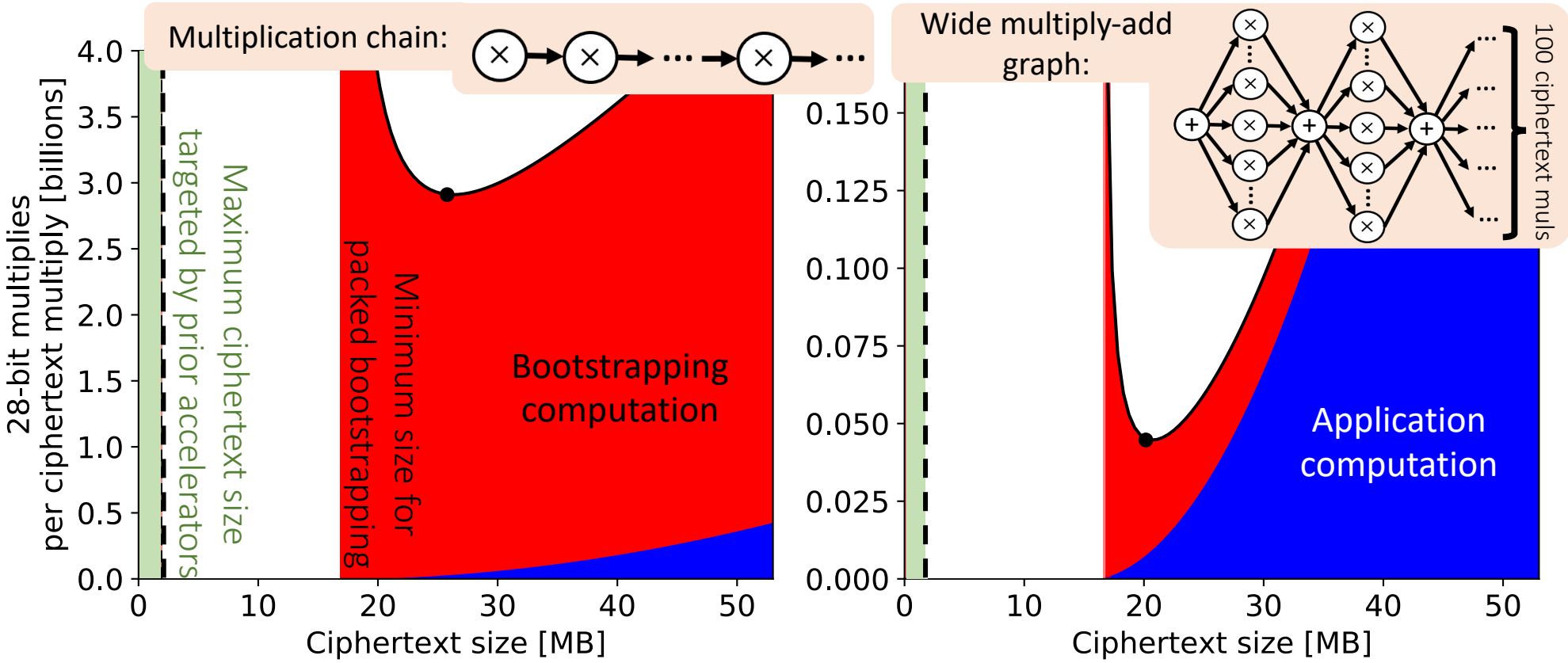

NTT0
$p_{11}[L:2L]$
2


Simple pipeline
Keyswitching pipeline
Automorphism
Transpose network
Transpose network
256MB
Banked
Scratchpad
PRG
#$@1-
NTT
NTT
Transpose network
CRB
NTT
NTT
Transpose network
PRG
#$@1-
NTT
NTT
Transpose network
CRB
NTT
NTT
Transpose network
Automorphism


FU group 1
[256, 768)
FU group 2
[768, 1024)
FU group 3
[768, 1024)
2 3 4
1 2 3 4
1 2 3 4
1 2 3 4
Transpose network
1 1 1 1
2 2 2 2
3 3 3 3
4 4 4 4


Lane group 0
Lane group 1
Lane group 2
Lane group 3
Subchunk 0
Subchunk 4
Subchunk 8
Subchunk 12
Subchunk 1
Subchunk 5
Subchunk 9
Subchunk 13
Subchunk 2
Subchunk 6
Subchunk 10
Subchunk 14
Subchunk 3
Subchunk 7
Subchunk 11
Subchunk 15
Fixed transpose network
Local transpose
Transpose unit
Transpose unit
Transpose unit


High bandwidth memory
256-lane group
256-lane group
256-lane group
256-lane group
Transpose network
256-lane group
256-lane group
256-lane group
256-lane group
High bandwidth memory


Multiplication chain:
Wide multiply-add graph:
100 ciphertext muls
28-bit multiplies per ciphertext multiply [billions]
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.150
0.125
0.100
0.075
0.050
0.025
0.000
Maximum ciphertext size targeted by prior accelerators
Minimum size for packed bootstrapping
Bootstrapping computation
Application computation
0
10
20
30
40
50
Ciphertext size [MB]
Ciphertext size [MB]