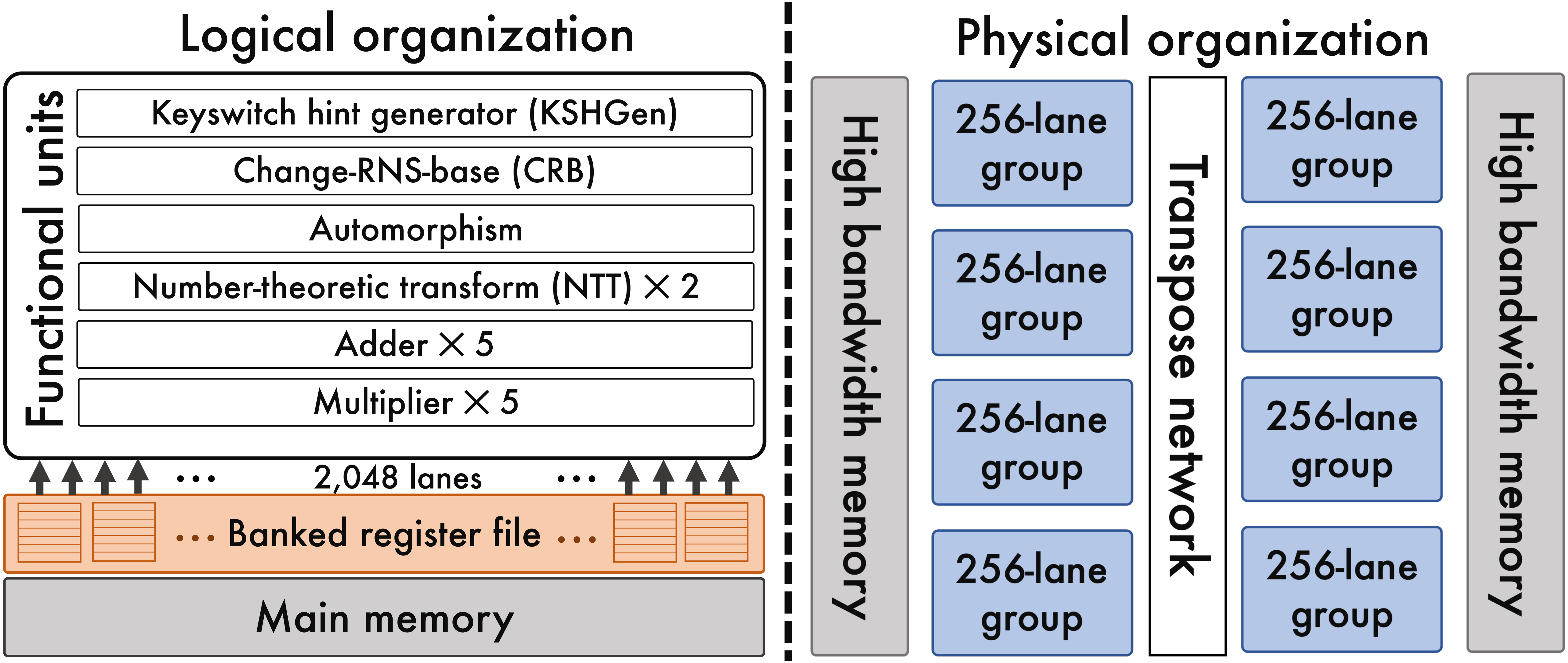
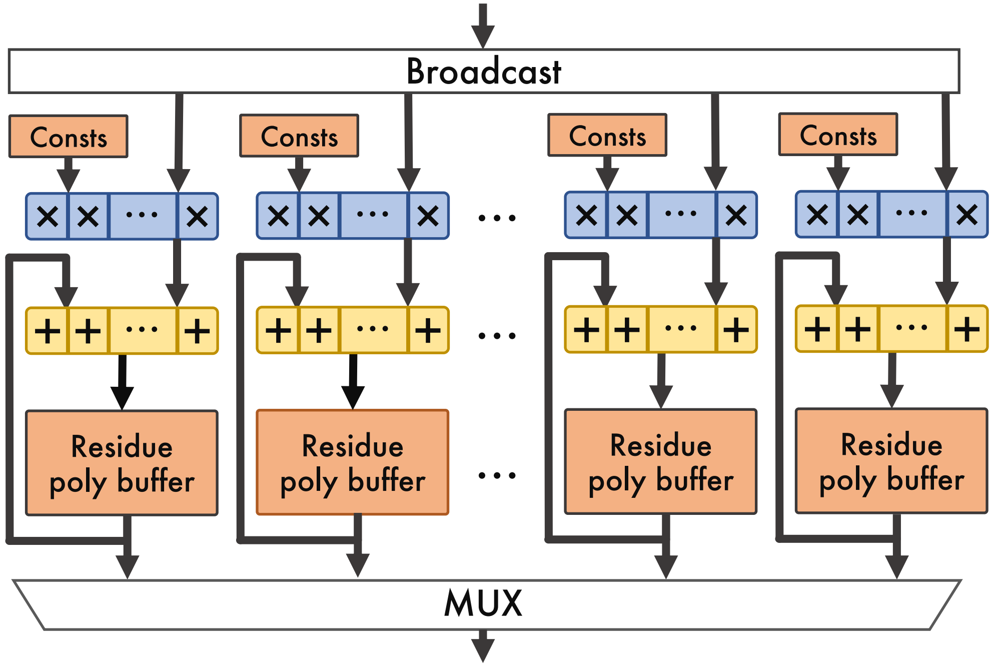
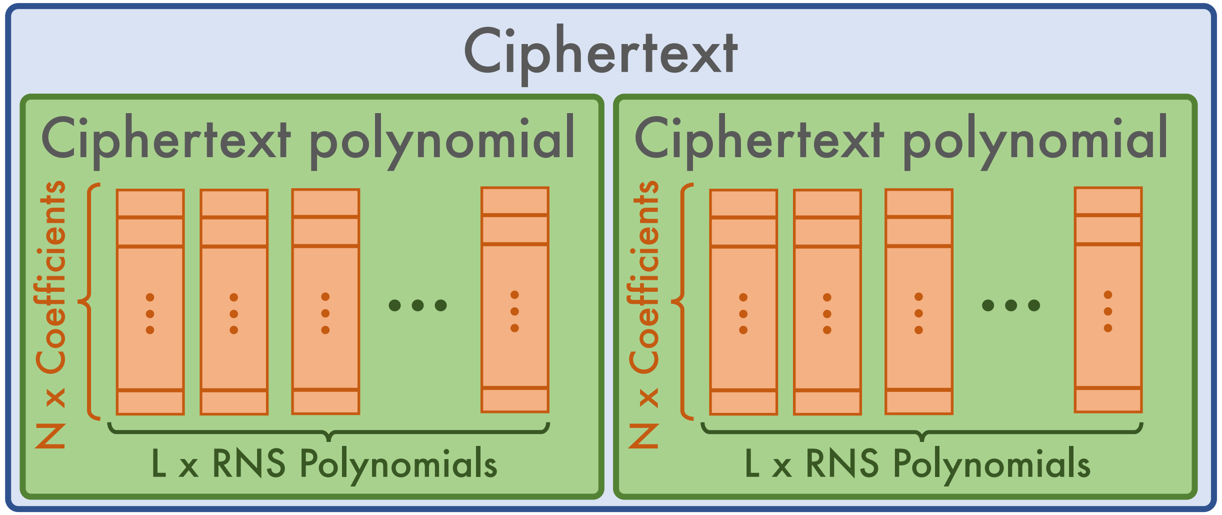
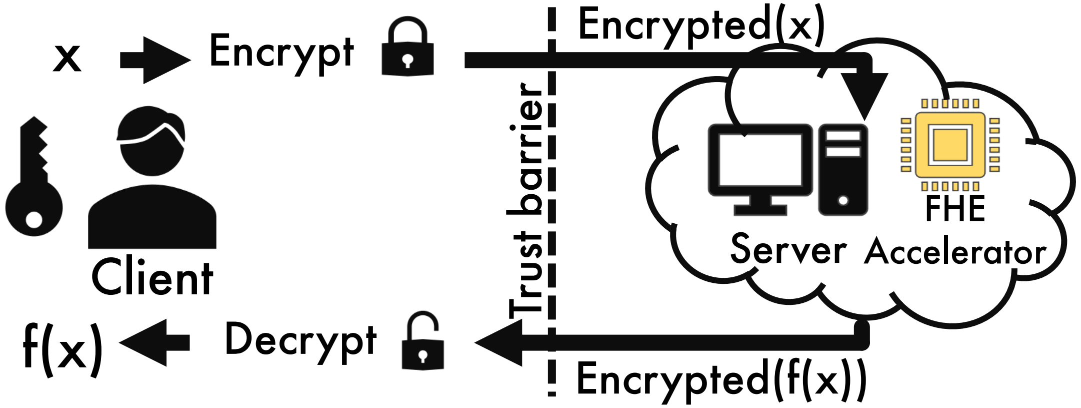
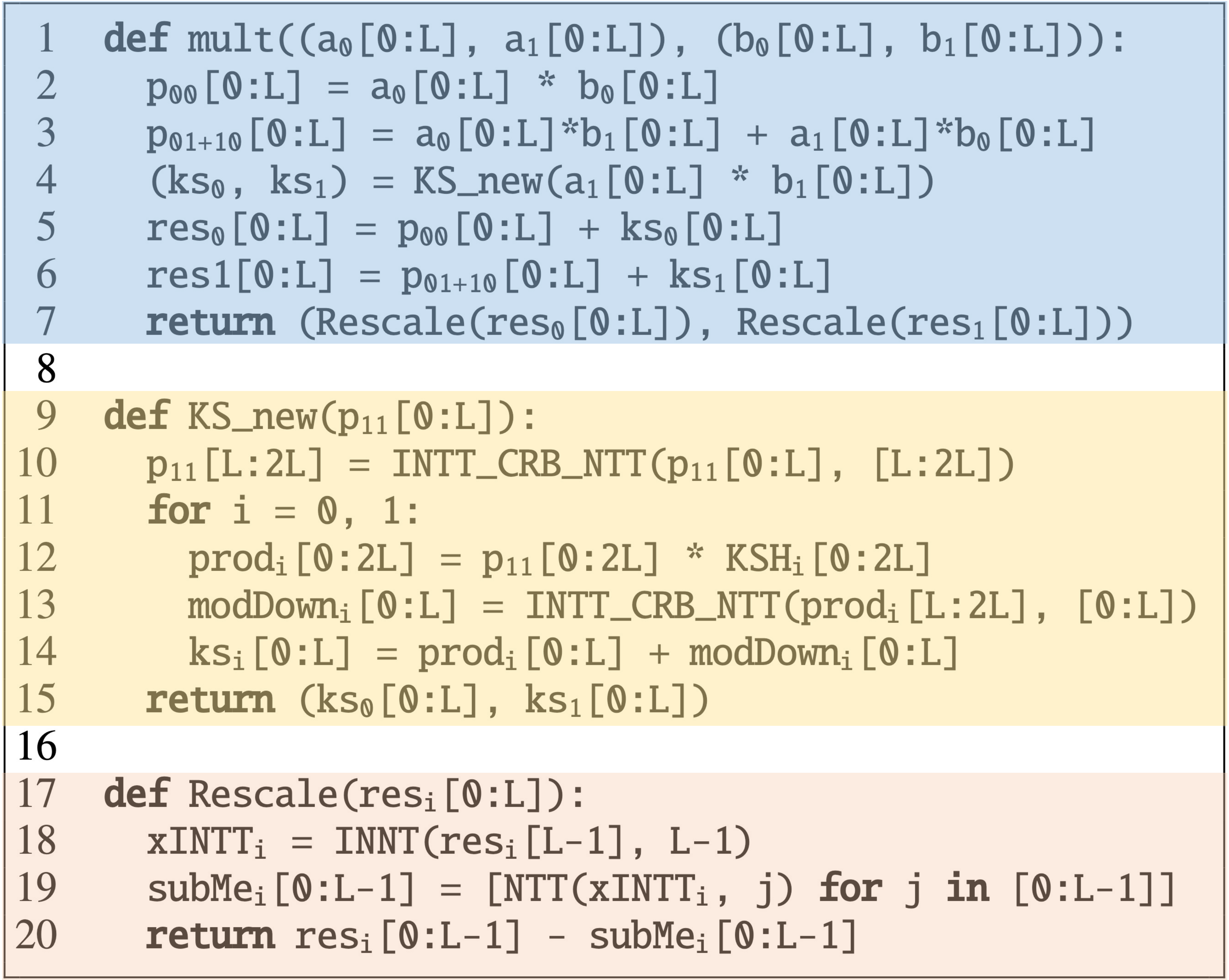

Logical organization
Functional units
Keyswitch hint generator (KSHGen)
Change-RNS-base (CRB)
Automorphism
Number-theoretic transform (NTT) × 2
Adder × 5
Multiplier × 5
2,048 lanes
Banked register file
Main memory
Physical organization
High bandwidth memory
256-lane group
256-lane group
256-lane group
256-lane group
Transpose network
256-lane group
256-lane group
256-lane group
256-lane group
High bandwidth memory

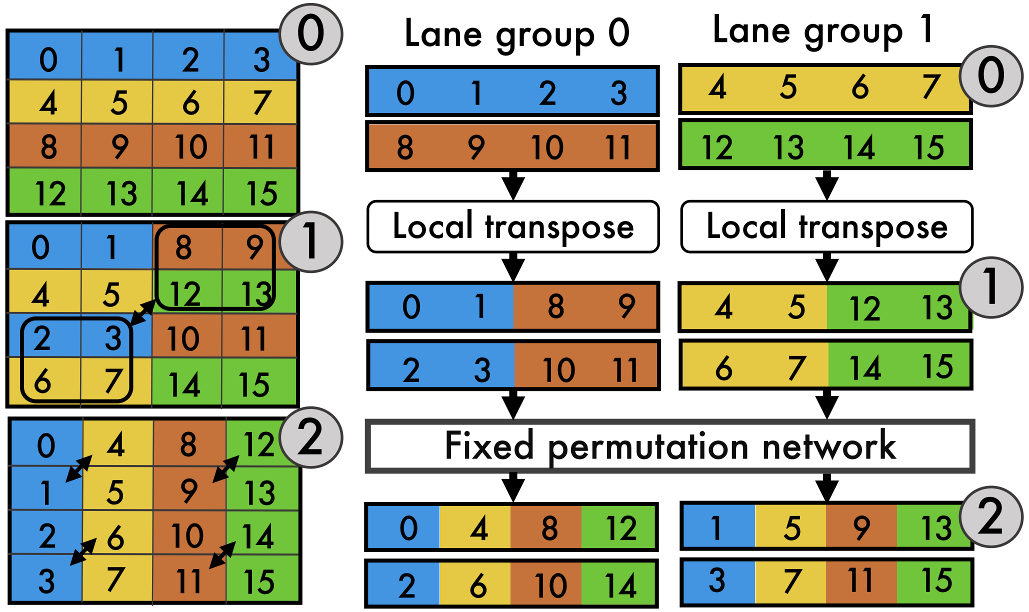

Broadcast
Consts
Consts
Consts
Consts
Residue poly buffer
Residue poly buffer
Residue poly buffer
Residue poly buffer
MUX

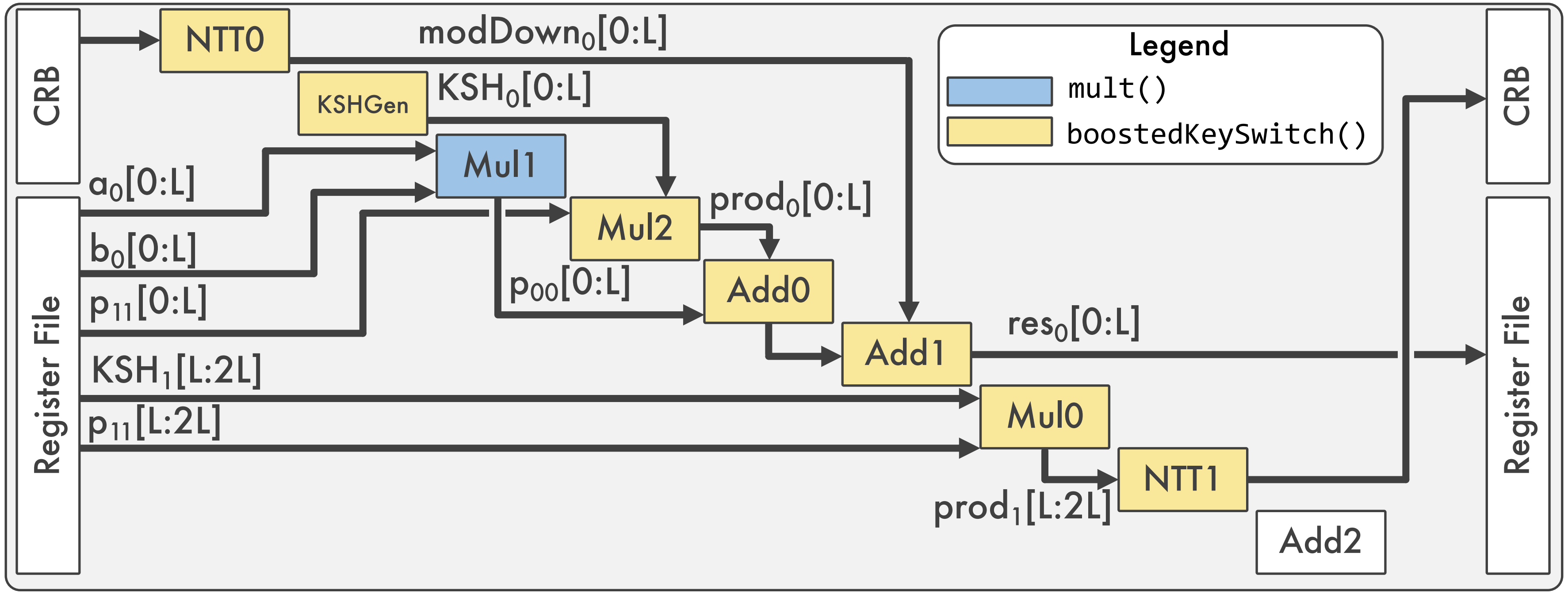

Ciphertext
Ciphertext polynomial
N x Coefficients
L x RNS Polynomials
Ciphertext polynomial
N x Coefficients
L x RNS Polynomials

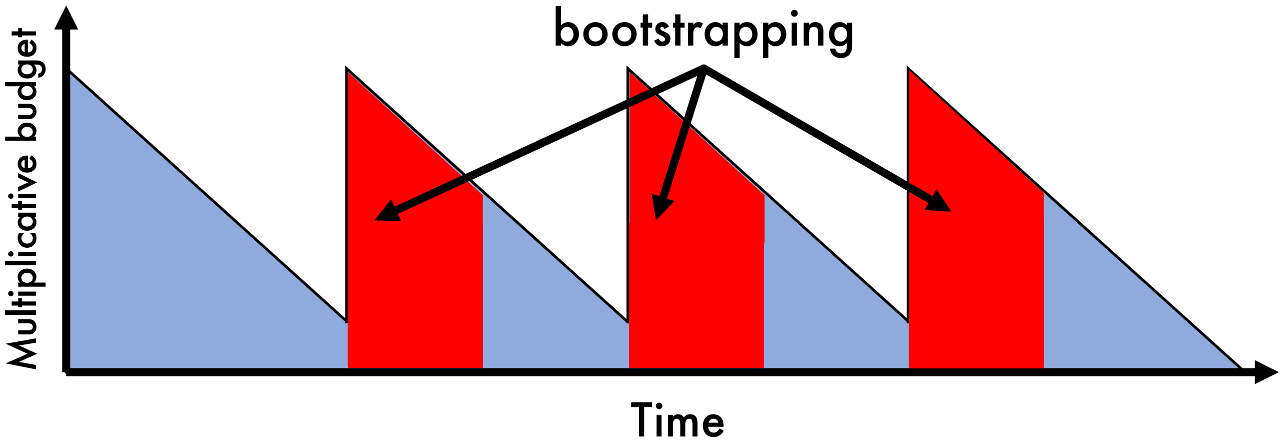

x
Encrypt
Encrypted(x)
Trust barrier
Client
Server
FHE
Accelerator
f(x)
Decrypt
Encrypted(f(x))

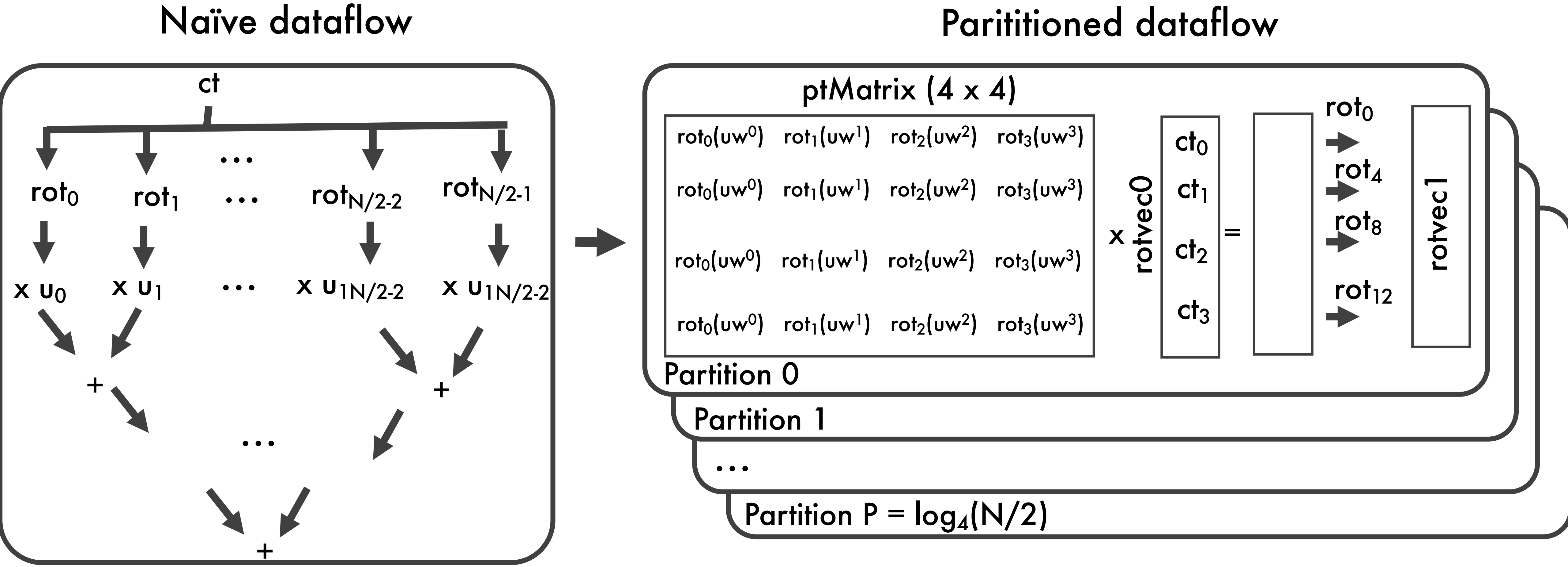


```
def mult((a_0[0:L], a_1[0:L]), (b_0[0:L], b_1[0:L])):
  p_00[0:L] = a_0[0:L] * b_0[0:L]
  p_01+10[0:L] = a_0[0:L]*b_1[0:L] + a_1[0:L]*b_0[0:L]
  (ks_0, ks_1) = KS_new(a_1[0:L] * b_1[0:L])
  res_0[0:L] = p_00[0:L] + ks_0[0:L]
  res1[0:L] = p_01+10[0:L] + ks_1[0:L]
  return (Rescale(res_0[0:L]), Rescale(res_1[0:L]))

def KS_new(p_11[0:L]):
  p_11[L:2L] = INTT_CRB_NTT(p_11[0:L], [L:2L])
  for i = 0, 1:
    prod_i[0:2L] = p_11[0:2L] * KSH_i[0:2L]
    modDown_i[0:L] = INTT_CRB_NTT(prod_i[L:2L], [0:L])
    ks_i[0:L] = prod_i[0:L] + modDown_i[0:L]
  return (ks_0[0:L], ks_1[0:L])

def Rescale(res_i[0:L]):
  xINTT_i = INNT(res_i[L-1], L-1)
  subMe_i[0:L-1] = [NTT(xINTT_i, j) for j in [0:L-1]]
  return res_i[0:L-1] - subMe_i[0:L-1]
```


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


CRB
Register File
NTT0
modDown_0[0:L]
KSHGen
KSH_0[0:L]
Mul1
a_0[0:L]
b_0[0:L]
p_{11}[0:L]
Mul2
prod_0[0:L]
Add0
p_{00}[0:L]
Add1
res_0[0:L]
KSH_1[L:2L]
p_{11}[L:2L]
Mul0
NTT1
prod_1[L:2L]
Add2
Legend
mult()
boostedKeySwitch()
CRB
Register File

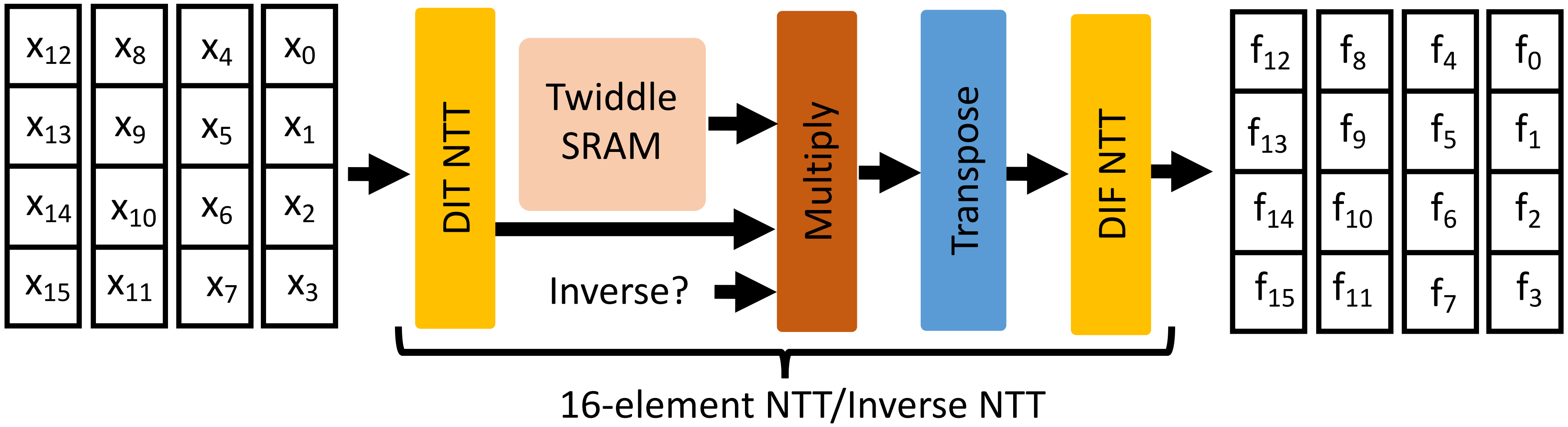

bootstrapping
Multiplicative budget
Time

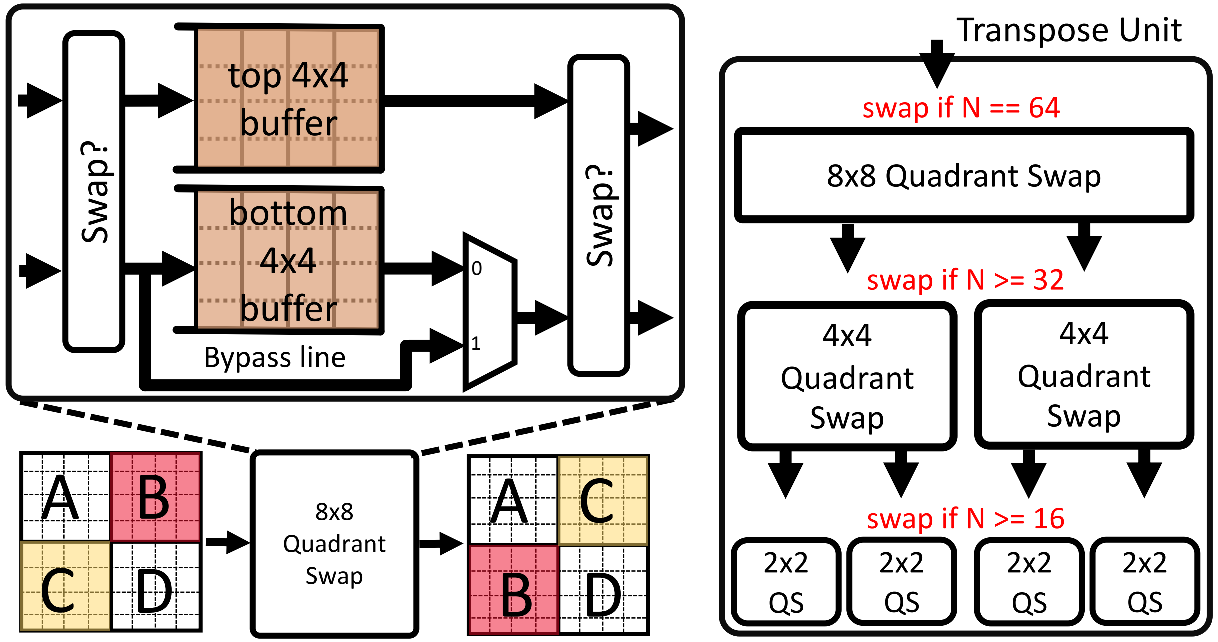


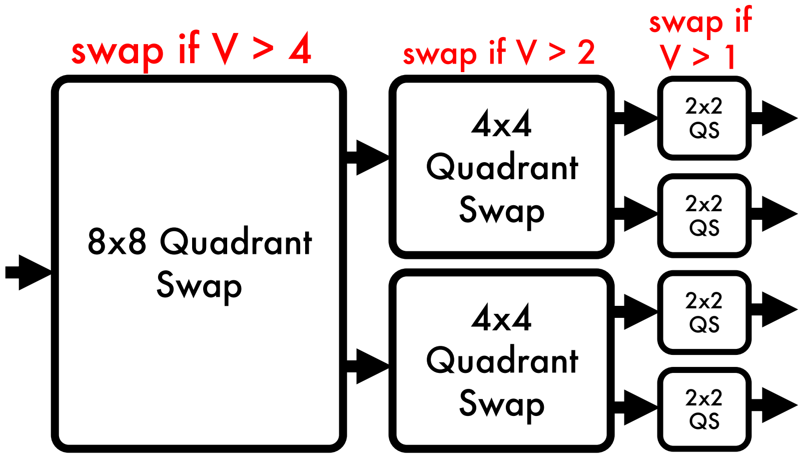
Naïve dataflow

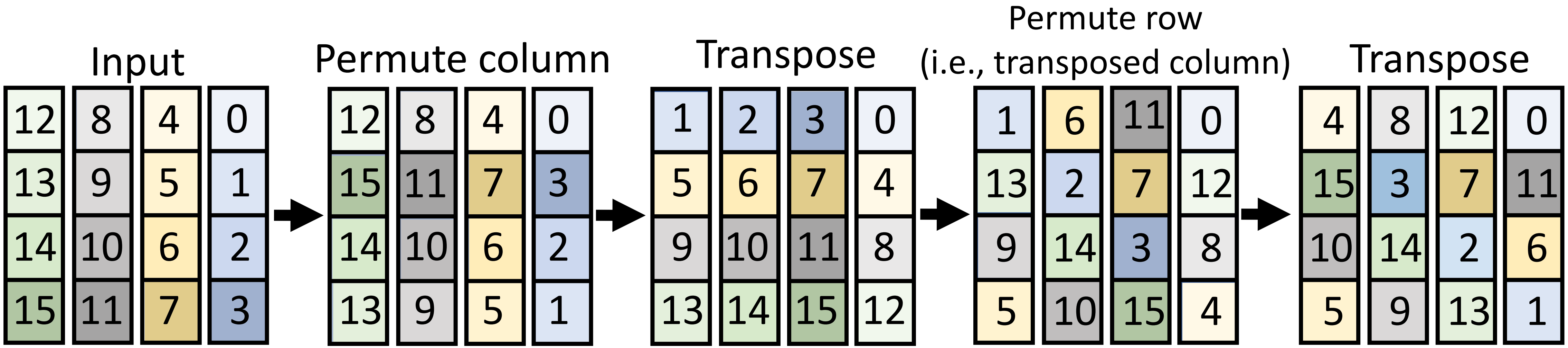
Parititioned dataflow

ct
rot$_0$
rot$_1$
…
rot$_{N/2-2}$
rot$_{N/2-1}$
x u$_0$
x u$_1$
x u$_{1N/2-2}$
x u$_{1N/2-2}$
+
+
…
+
ptMatrix (4 x 4)
rot$_0$(uw^0) rot$_1$(uw^1) rot$_2$(uw^2) rot$_3$(uw^3)
rot$_0$(uw^0) rot$_1$(uw^1) rot$_2$(uw^2) rot$_3$(uw^3)
rot$_0$(uw^0) rot$_1$(uw^1) rot$_2$(uw^2) rot$_3$(uw^3)
rot$_0$(uw^0) rot$_1$(uw^1) rot$_2$(uw^2) rot$_3$(uw^3)
x
rotvec0
ct$_0$
ct$_1$
ct$_2$
ct$_3$
=
rot$_0$
rot$_4$
rot$_8$
rot$_{12}$
rotvec1
Partition 0
Partition 1
…
Partition P = log$_4$(N/2)


x_{12} x_8 x_4 x_0
x_{13} x_9 x_5 x_1
x_{14} x_{10} x_6 x_2
x_{15} x_{11} x_7 x_3
DIT NTT
Twiddle SRAM
Inverse?
Multiply
Transpose
DIF NTT
f_{12} f_8 f_4 f_0
f_{13} f_9 f_5 f_1
f_{14} f_{10} f_6 f_2
f_{15} f_{11} f_7 f_3
16-element NTT/Inverse NTT


Swap?
top 4x4 buffer
bottom 4x4 buffer
0
1
Bypass line
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

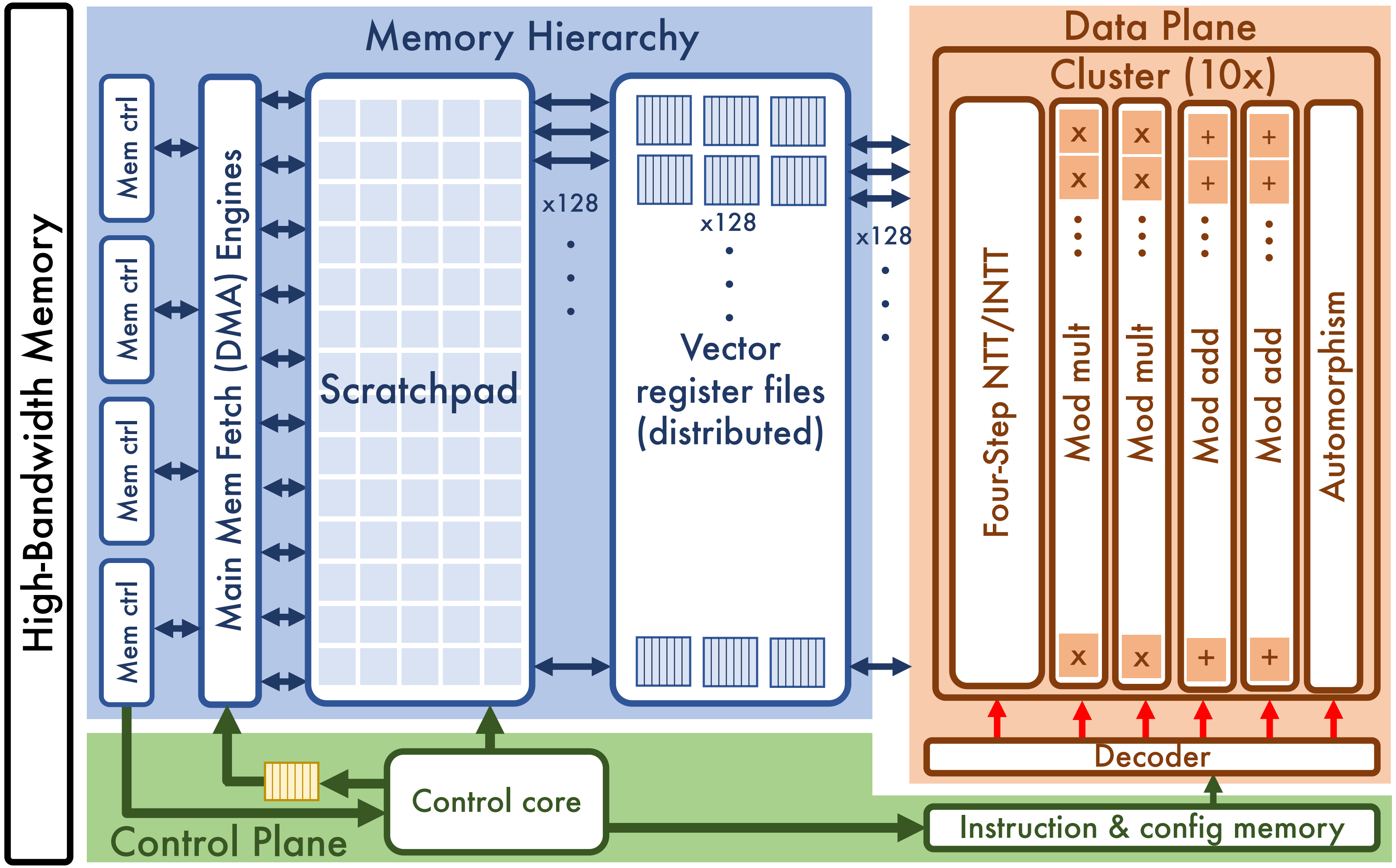
1	2	3	0
5	6	7	4
9	10	11	8
13	14	15	12

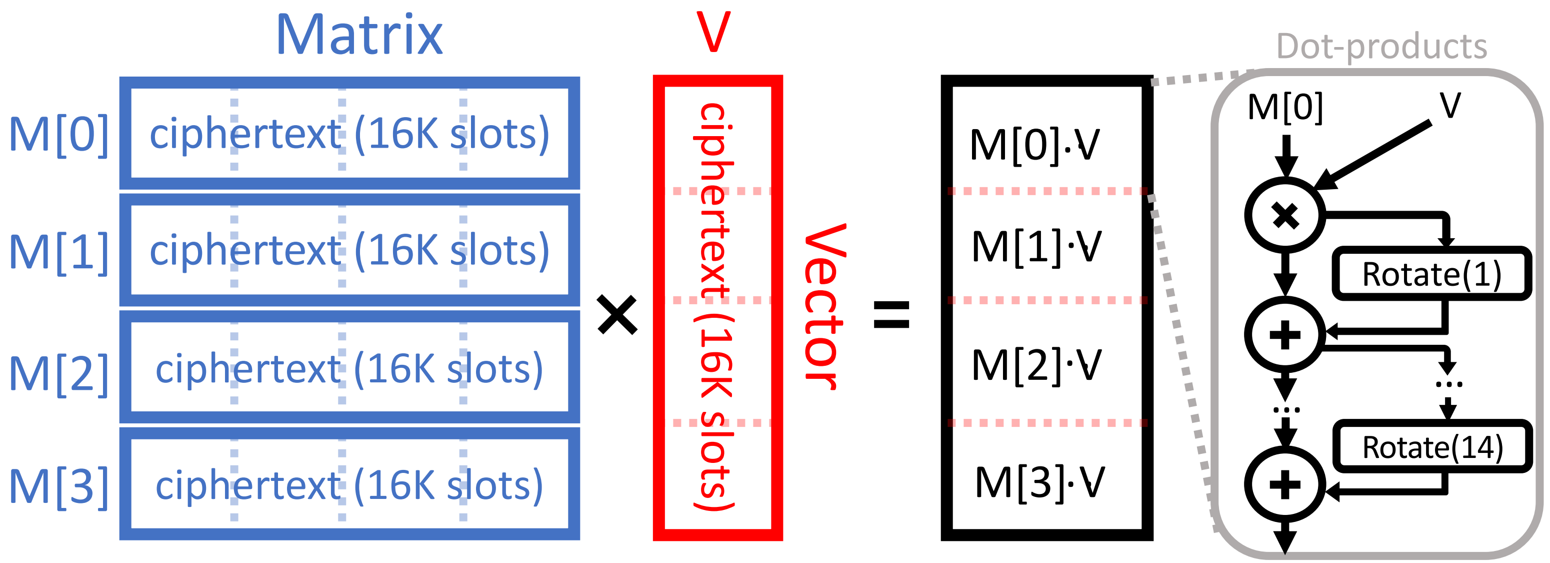
Permute row
(i.e., transposed column)

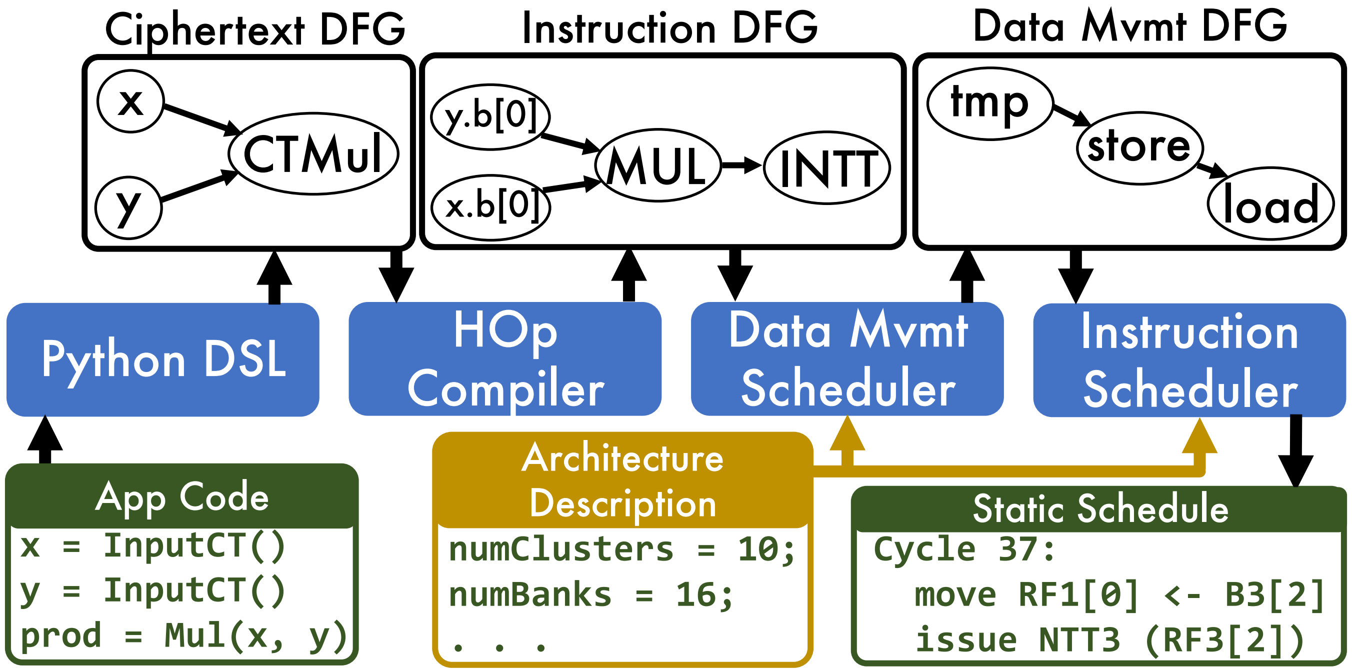
1	6	11	0
13	2	7	12
9	14	3	8
5	10	15	4

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
x
+
Decoder
Instruction & config memory
Control core
Control Plane


Matrix
V
M[0]
M[1]
M[2]
M[3]
ciphertext (16K slots)
ciphertext (16K slots)
ciphertext (16K slots)
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)



Python DSL

HOp Compiler

Data Mvmt Scheduler

Instruction Scheduler

App Code

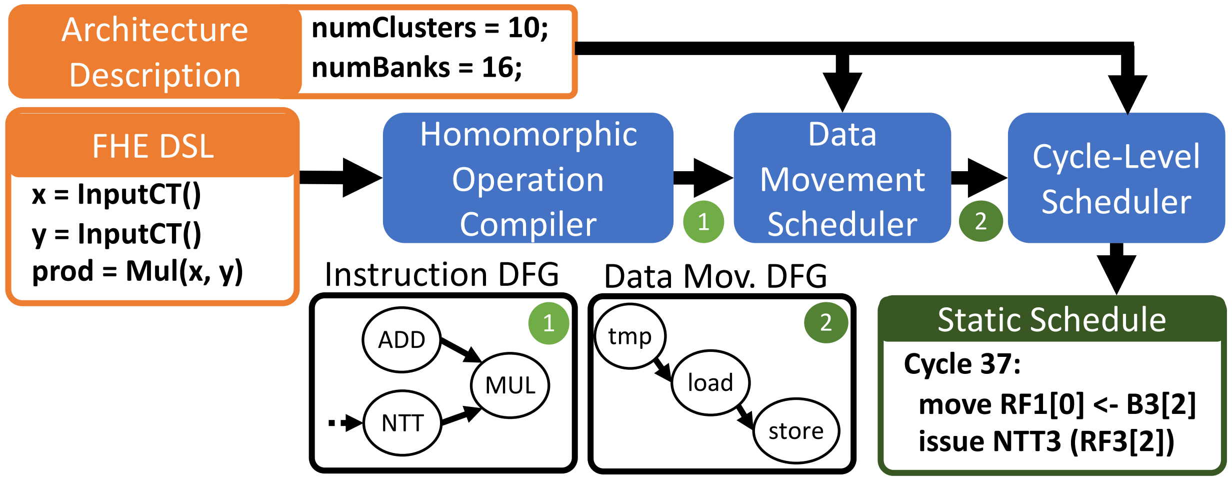
```
x = InputCT()
y = InputCT()
prod = Mul(x, y)
```

Architecture Description

```
numClusters = 10;
numBanks = 16;
. . .
```

Static Schedule

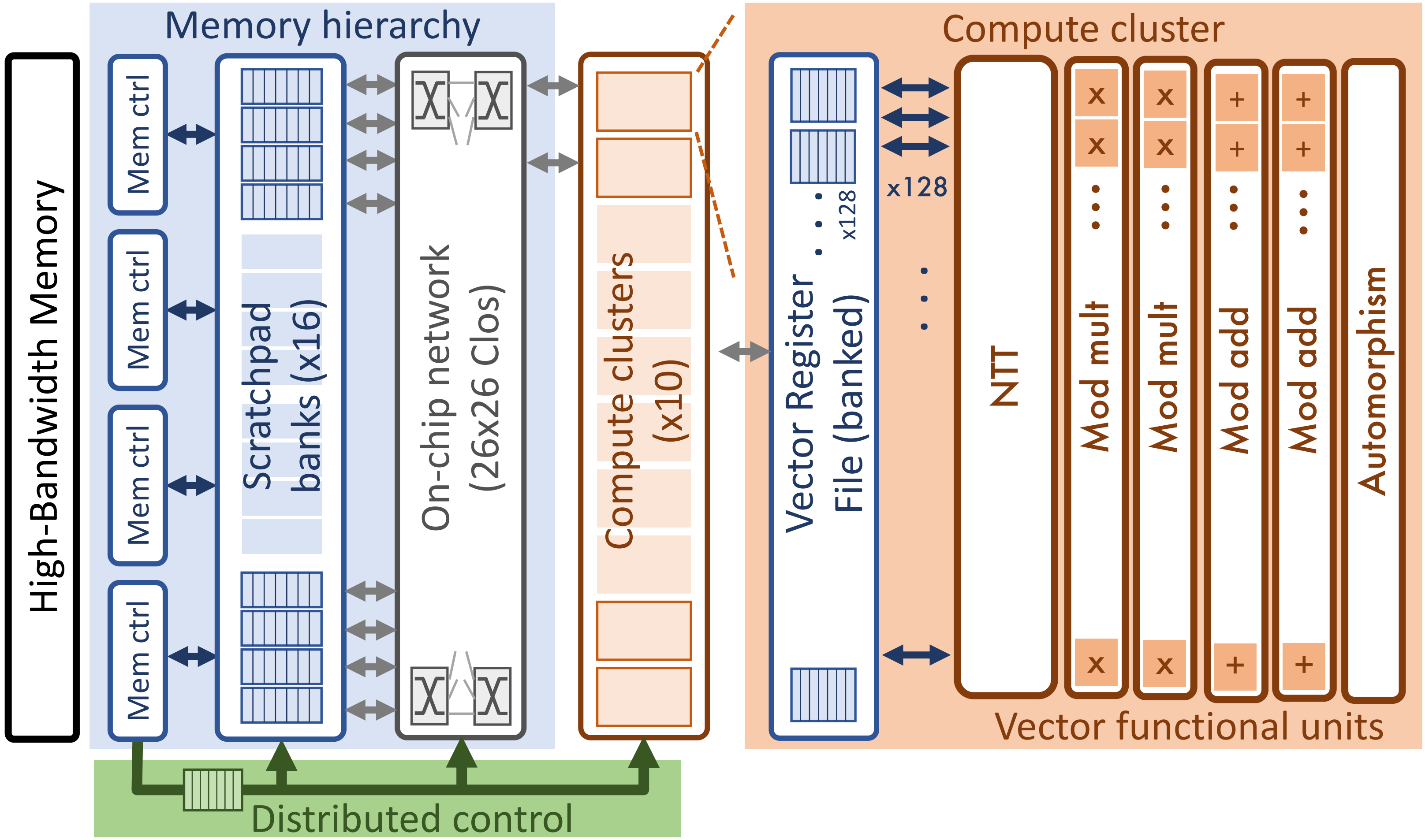
```
Cycle 37:
  move RF1[0] <- B3[2]
  issue NTT3 (RF3[2])
```

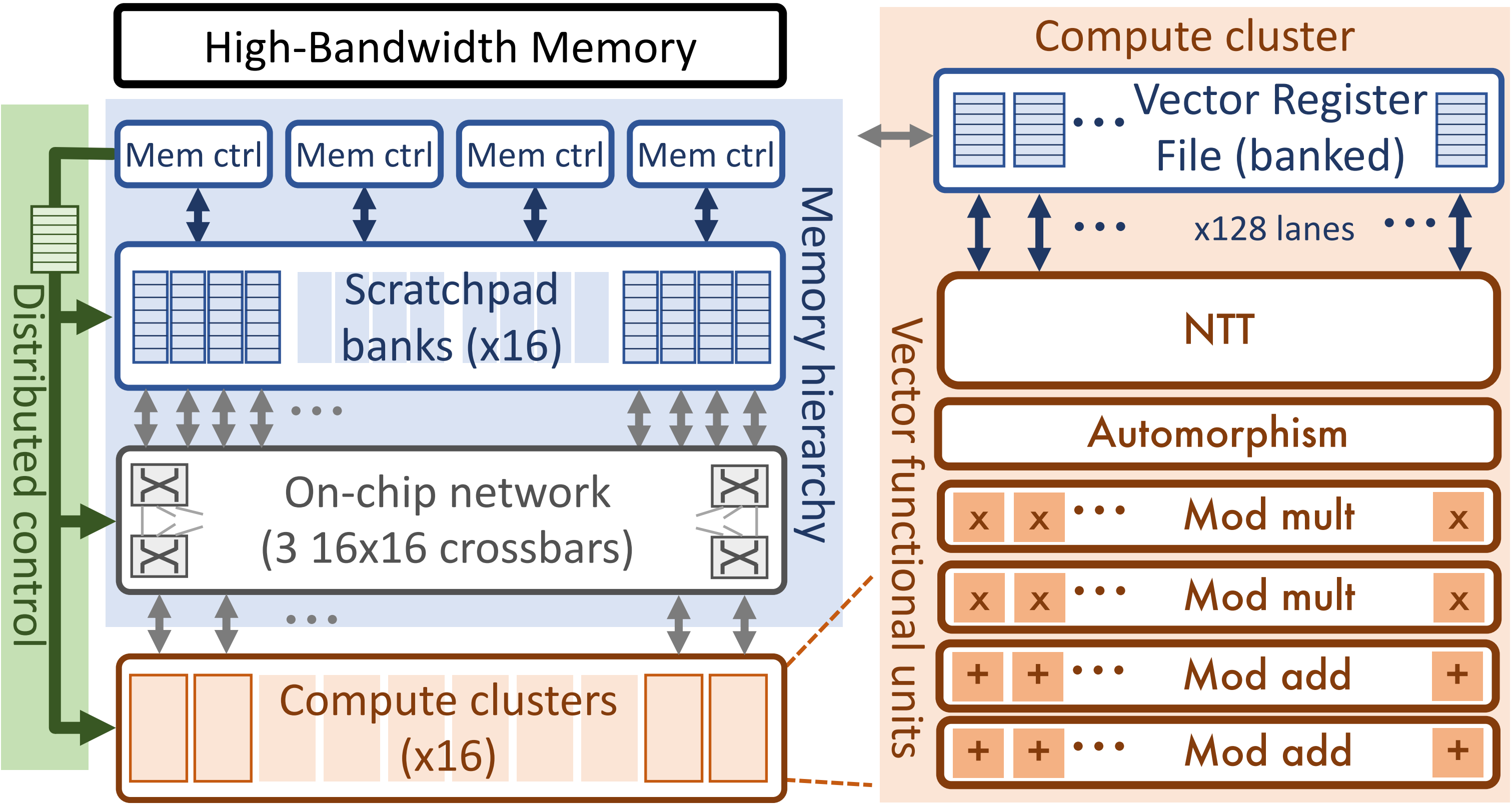


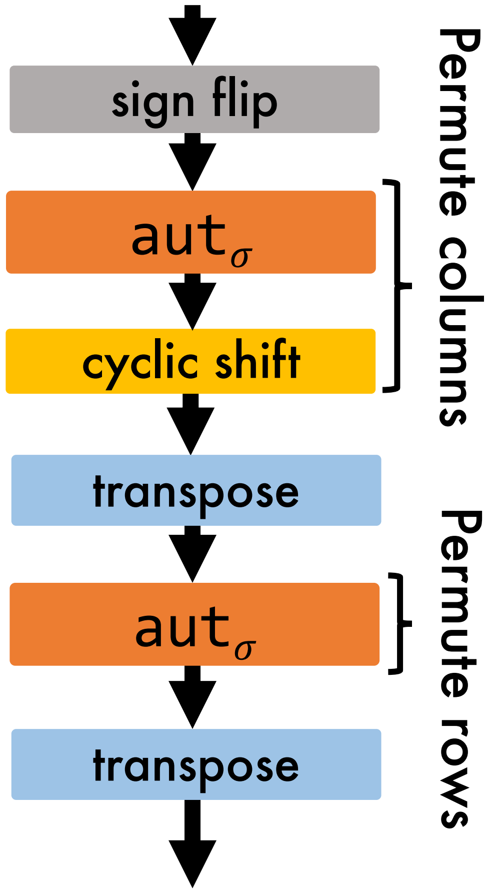
```
numClusters = 10;
numBanks = 16;
```

```
x = InputCT()
y = InputCT()
prod = Mul(x, y)
```

```
Cycle 37:
  move RF1[0] <- B3[2]
  issue NTT3 (RF3[2])
```


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


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



sign flip

$\mathtt{aut}_\sigma$

cyclic shift

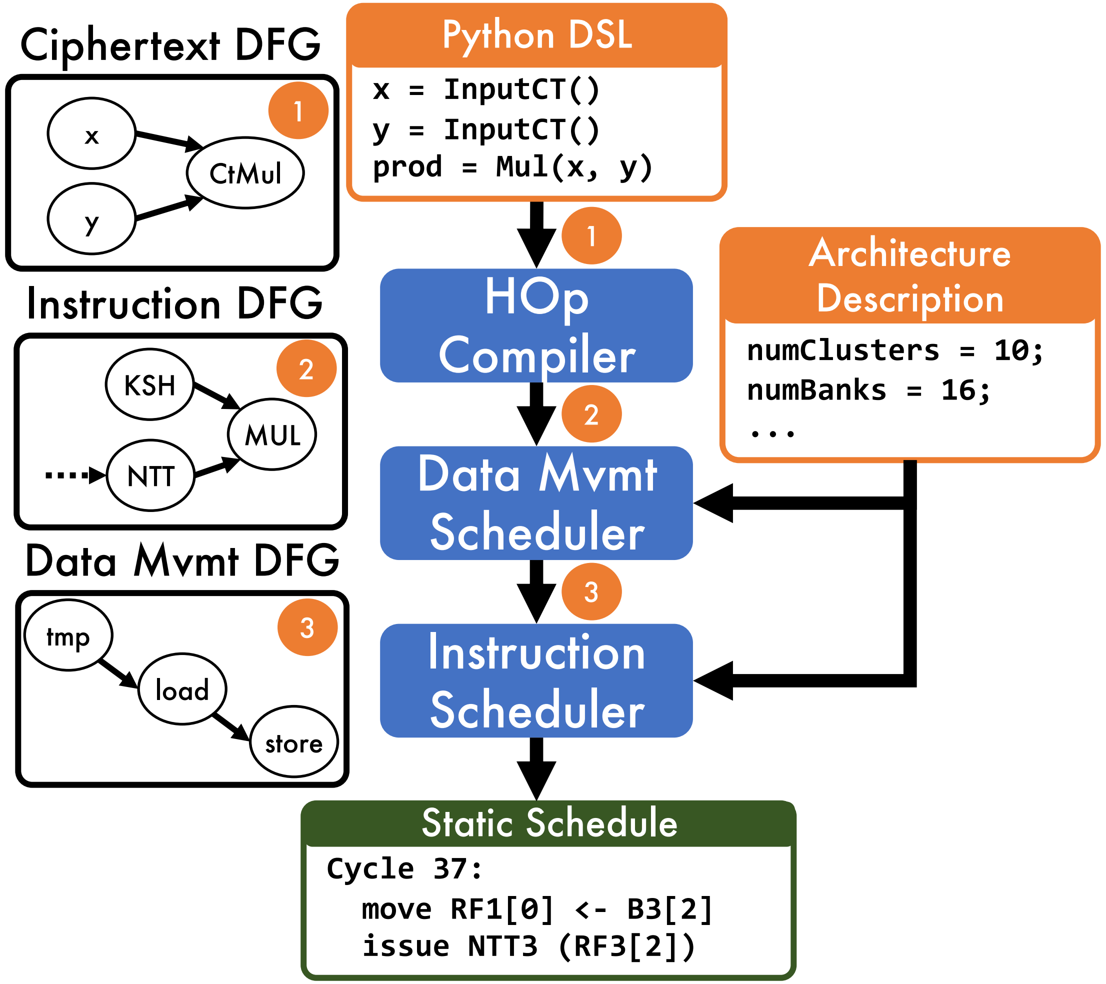
transpose

$\mathtt{aut}_\sigma$

transpose

Permute columns

Permute rows



Python DSL

```
x = InputCT()
y = InputCT()
prod = Mul(x, y)
```

HOp Compiler

Data Mvmt Scheduler

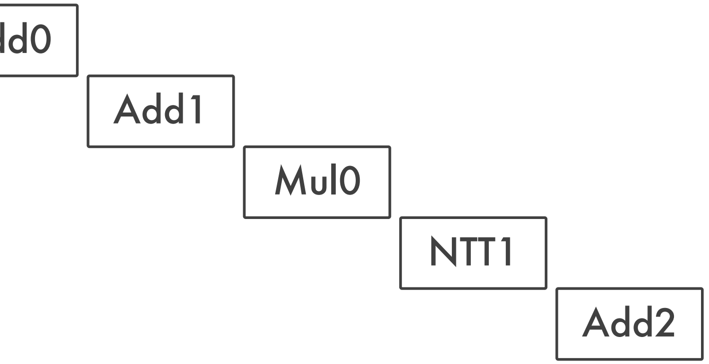
Instruction Scheduler

Architecture Description

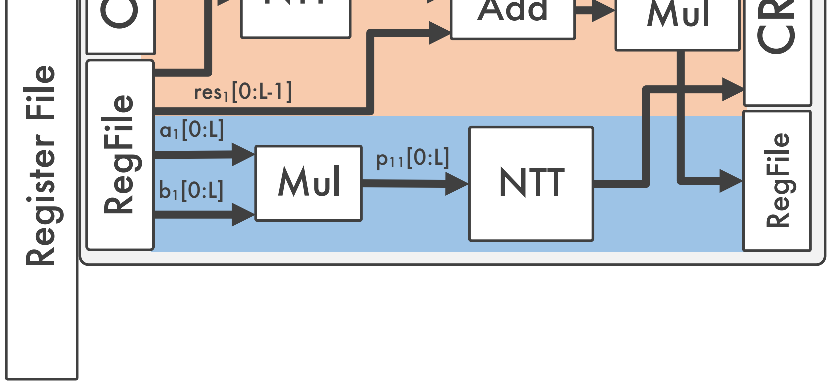
```
numClusters = 10;
numBanks = 16;
...
```

Static Schedule

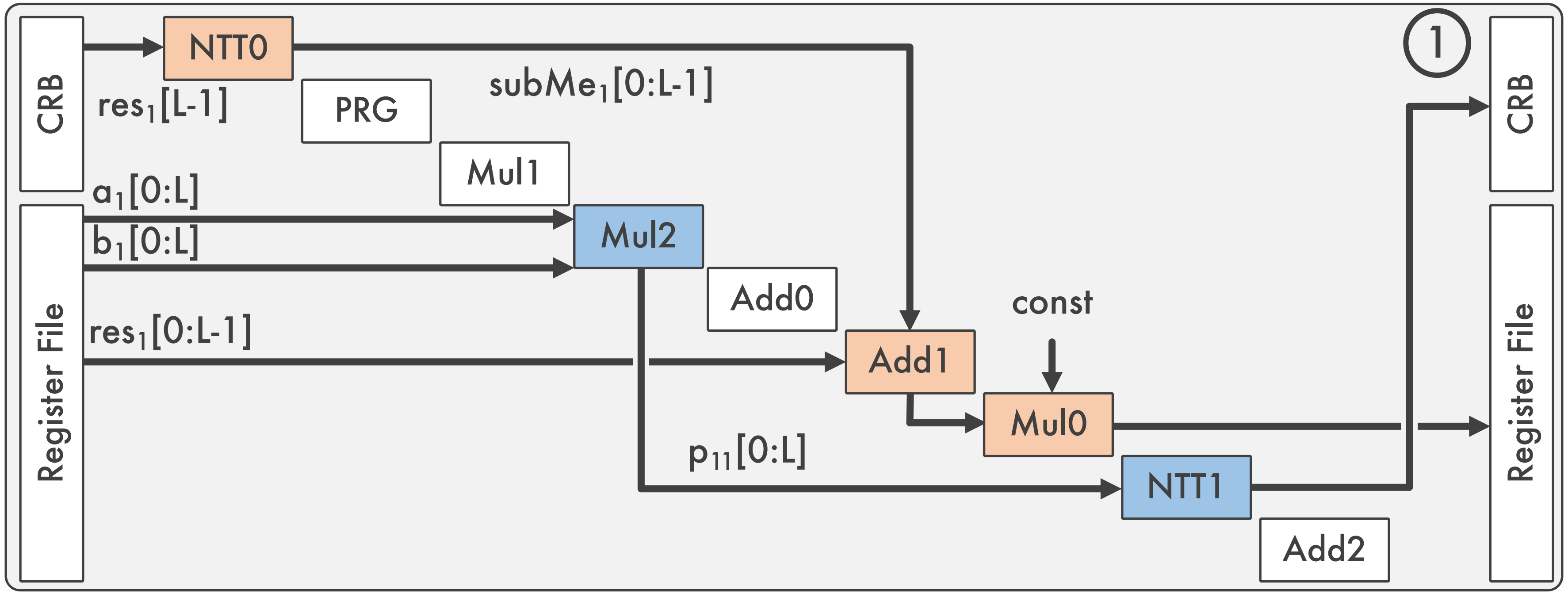
```
Cycle 37:
  move RF1[0] <- B3[2]
  issue NTT3 (RF3[2])
```



dd0
Add1
Mul0
NTT1
Add2

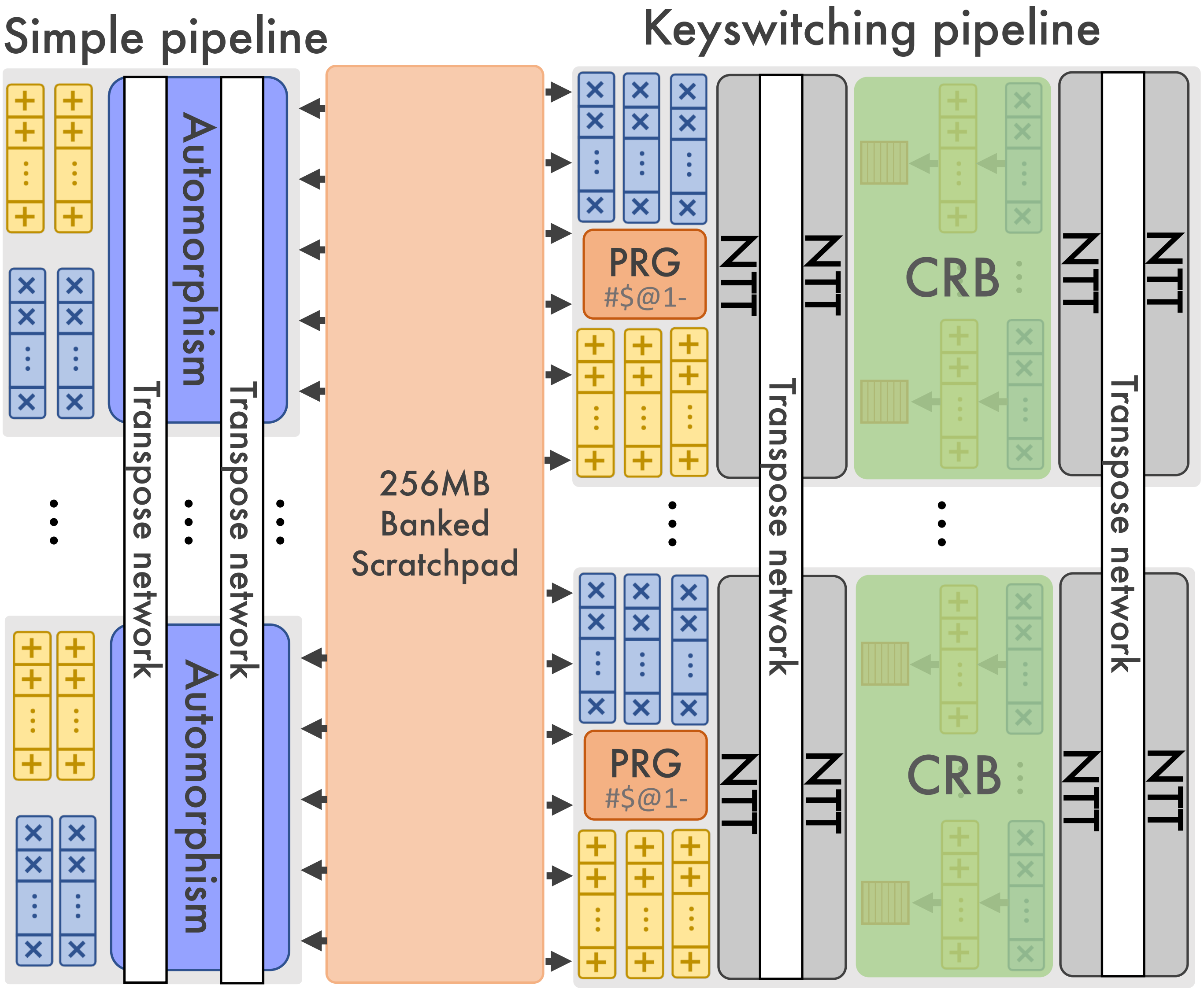


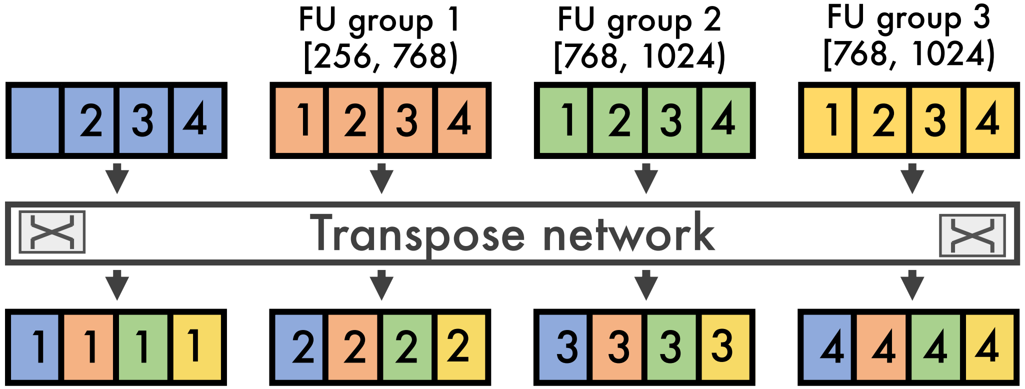
Register File
C
NTT
Add
Mul
CR
RegFile
$res_1[0:L-1]$
$a_1[0:L]$
$b_1[0:L]$
Mul
$p_{11}[0:L]$
NTT
RegFile

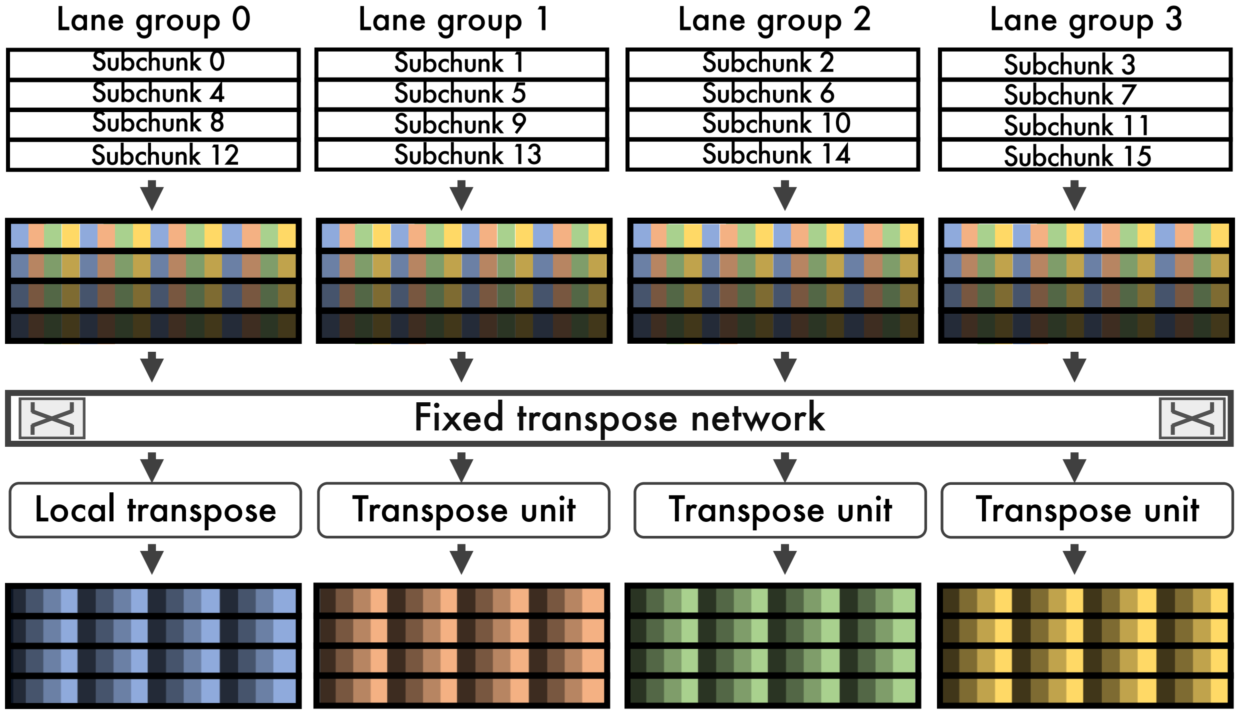


CRB
NTT0
$res_1[L-1]$
PRG
$subMe_1[0:L-1]$
Mul1
$a_1[0:L]$
$b_1[0:L]$
Mul2
Add0
Register File
$res_1[0:L-1]$
Add1
const
Mul0
$p_{11}[0:L]$
NTT1
Add2
1
CRB
Register File

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

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


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
Subchunk 0
Subchunk 4
Subchunk 8
Subchunk 12

Lane group 1
Subchunk 1
Subchunk 5
Subchunk 9
Subchunk 13

Lane group 2
Subchunk 2
Subchunk 6
Subchunk 10
Subchunk 14

Lane group 3
Subchunk 3
Subchunk 7
Subchunk 11
Subchunk 15

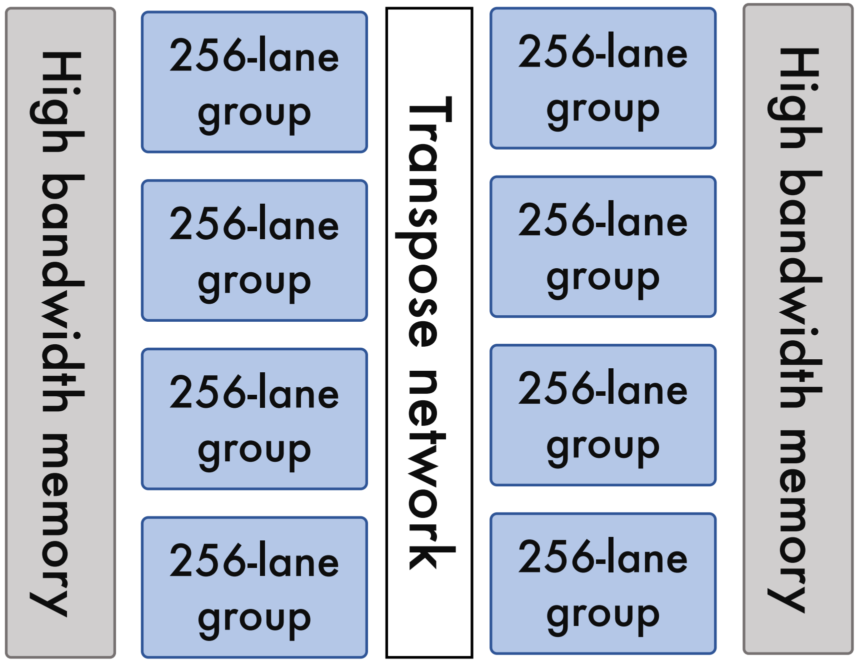
Fixed transpose network

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