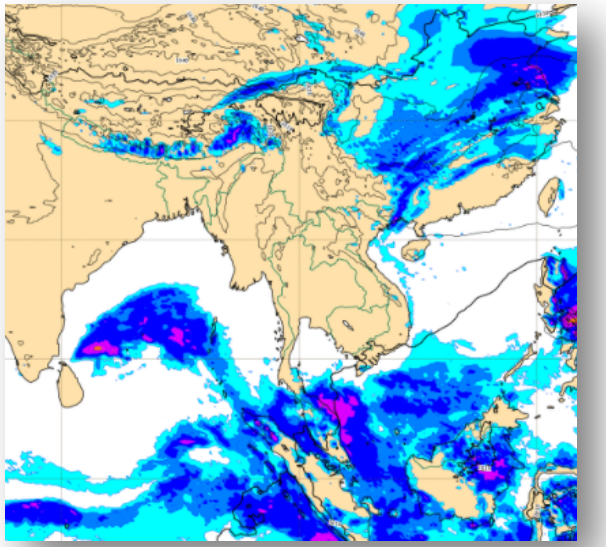
EMS: Efficient Memory Subsystem Synthesis for Spatial Accelerators

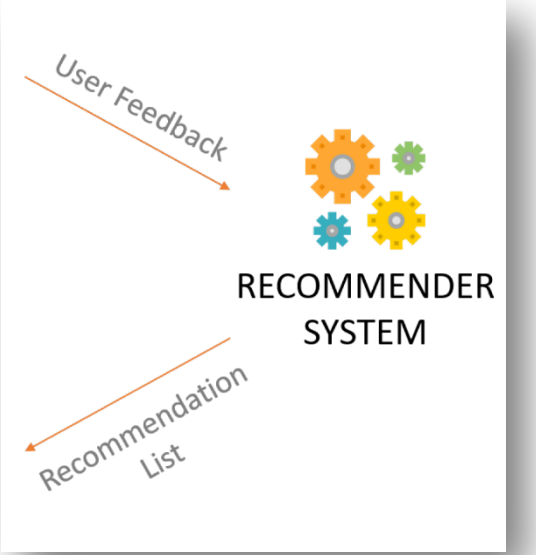
Liancheng Jia, Yuyue Wang, Jingwen Leng, and Yun Liang

Center for Energy-efficient Computing and Applications, Peking University

A wide Usage of Tensor Applications

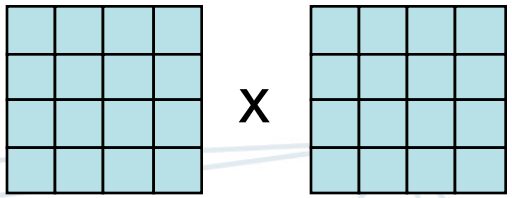


Scientific Computation

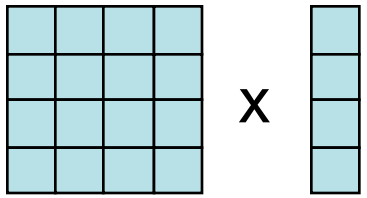


Recommendation System

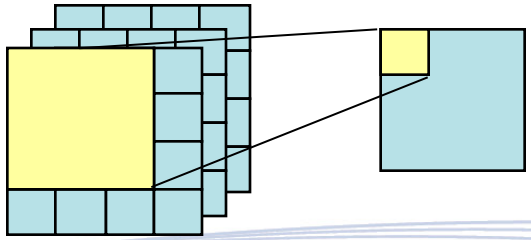
Computer Vision



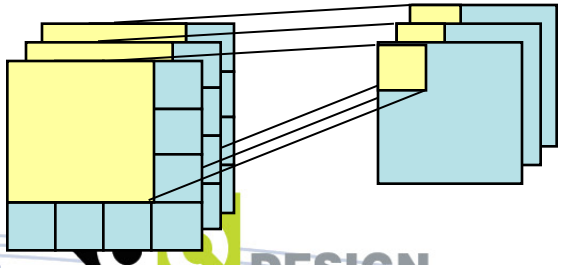
GEMM



GEMV

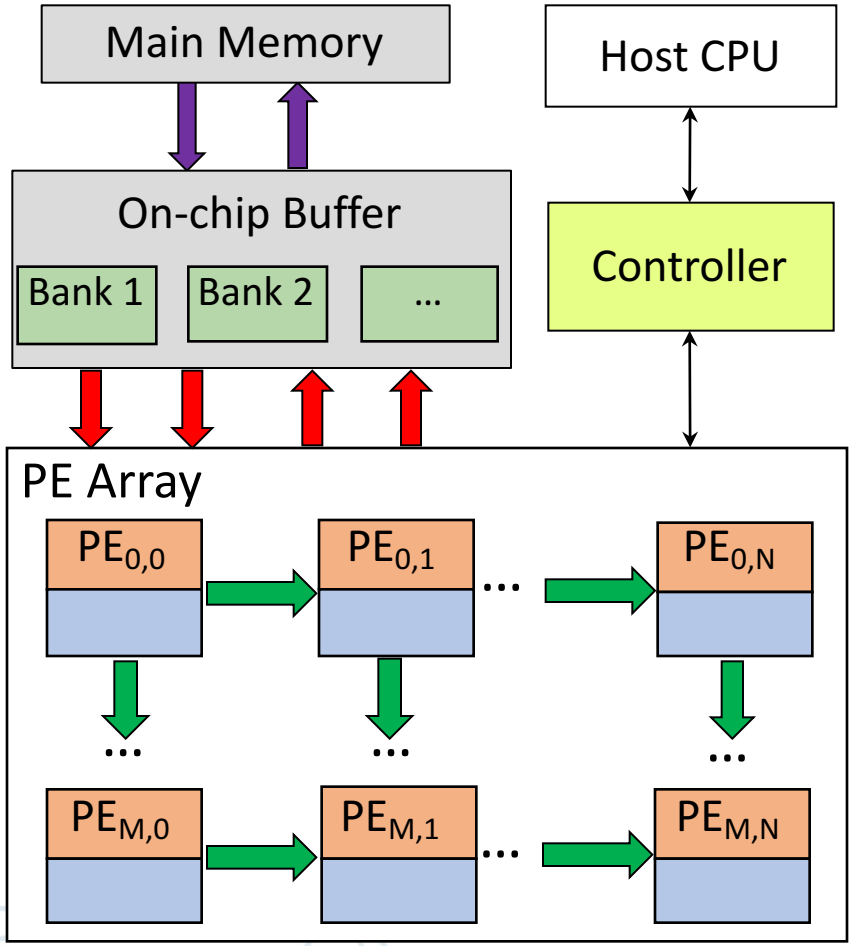


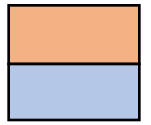
Conv2D



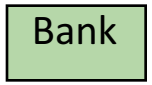
Depthwise-Conv2D

Spatial Accelerator Architecture




PE


Controller

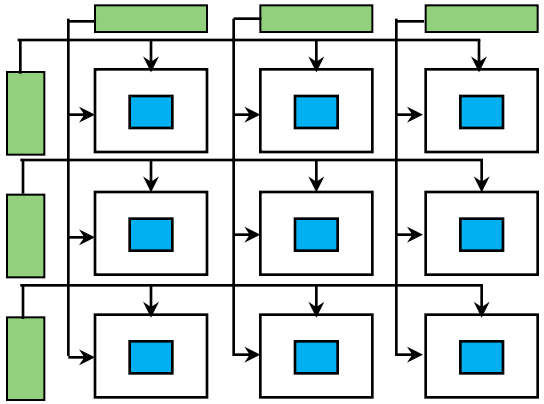


On-chip memory bank

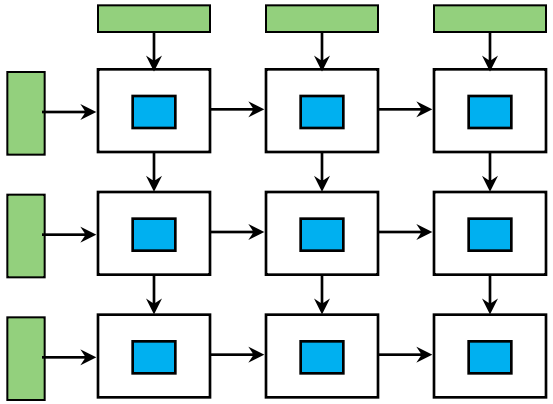

PE array dataflow interconnection


PE-memory interconnection

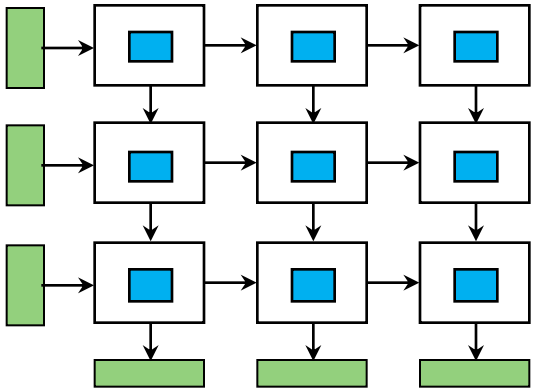
PE Array Dataflow



Multicast

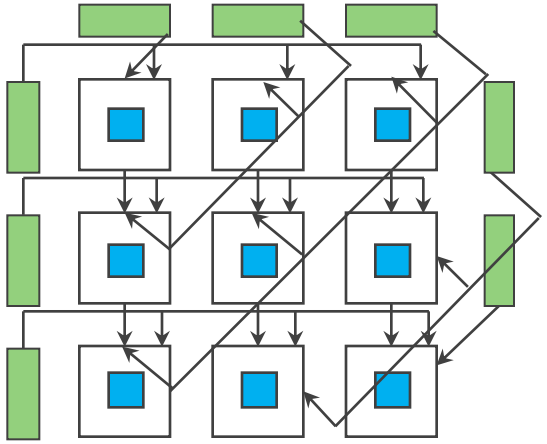


Output stationary systolic

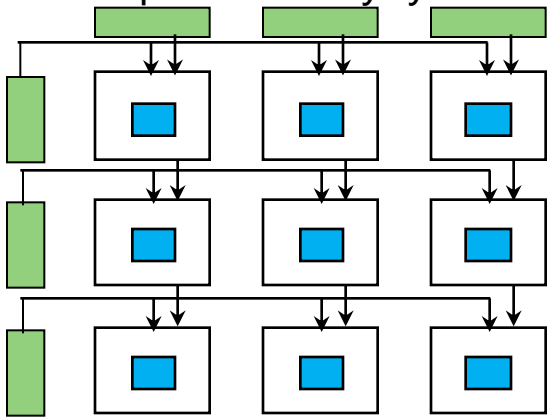


Weight stationary systolic
(Applied by TPU)

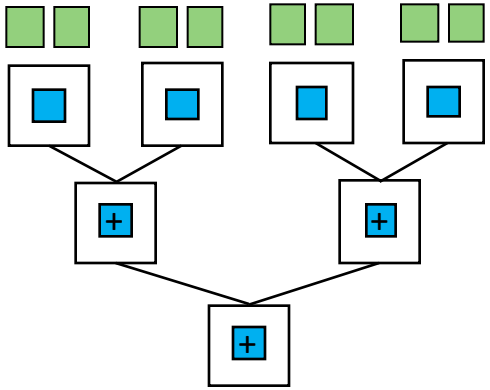
.....



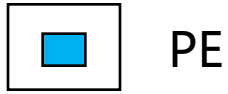
Row Stationary
(Applied by Eyeriss)



Multicast+systolic
(Applied by ShiDiannao)



Reduction Tree
(Applied by MAERI)

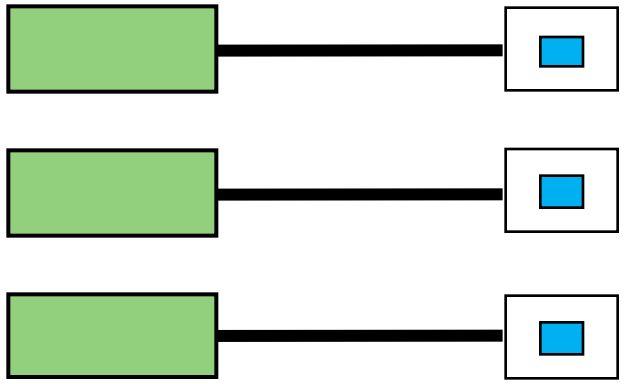


PE

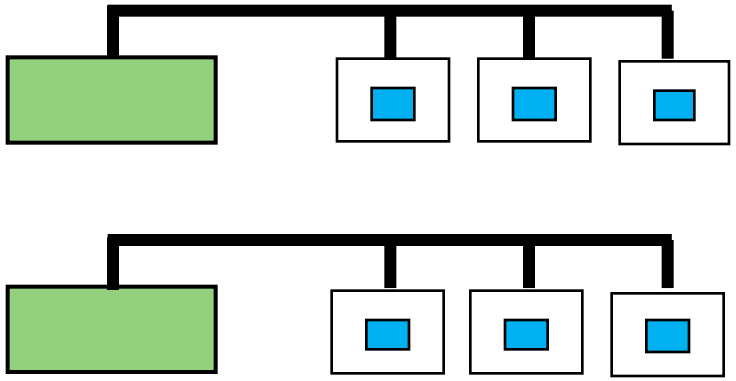
Memory bank

Jouppi, Norman P., et al. "In-datacenter performance analysis of a tensor processing unit." ISCA 2017
Chen, Yu-Hsin, et al. "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks." JSSC 2016
Du, Zidong, et al. "ShiDianNao: Shifting vision processing closer to the sensor." ISCA 2015
H Kwon et al. MAERI: Enabling Flexible Dataflow Mapping over DNN Accelerators via Reconfigurable Interconnects ASPLOS 2018

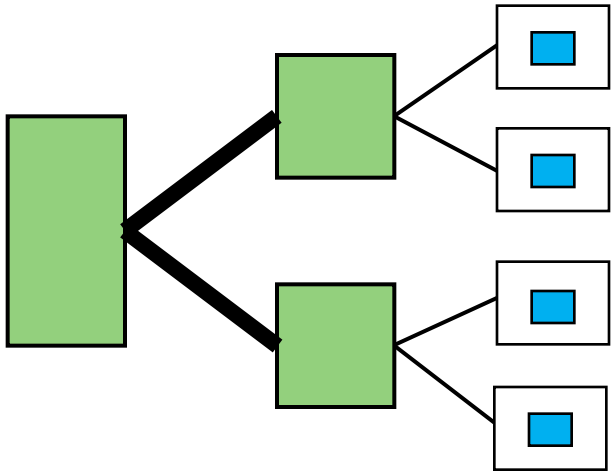
PE-memory Interconnection



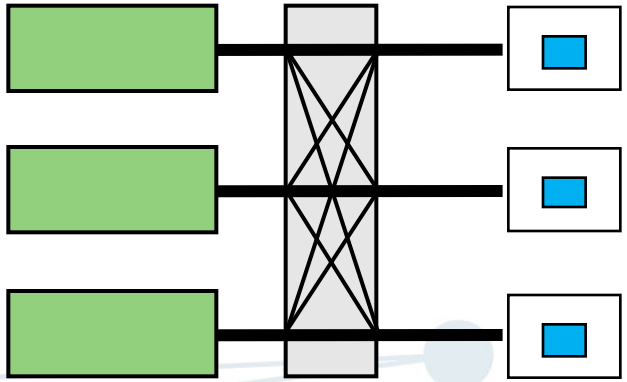
Unicast



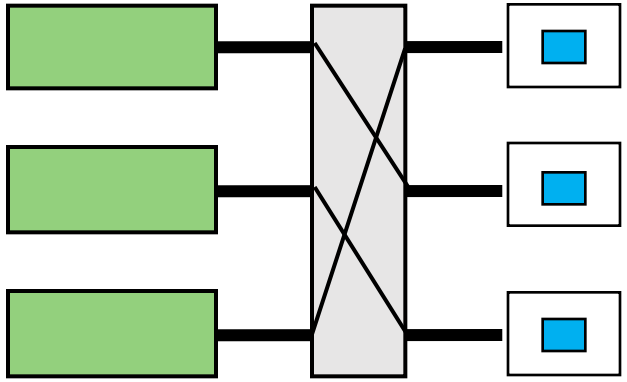
Multicast



Fat Tree

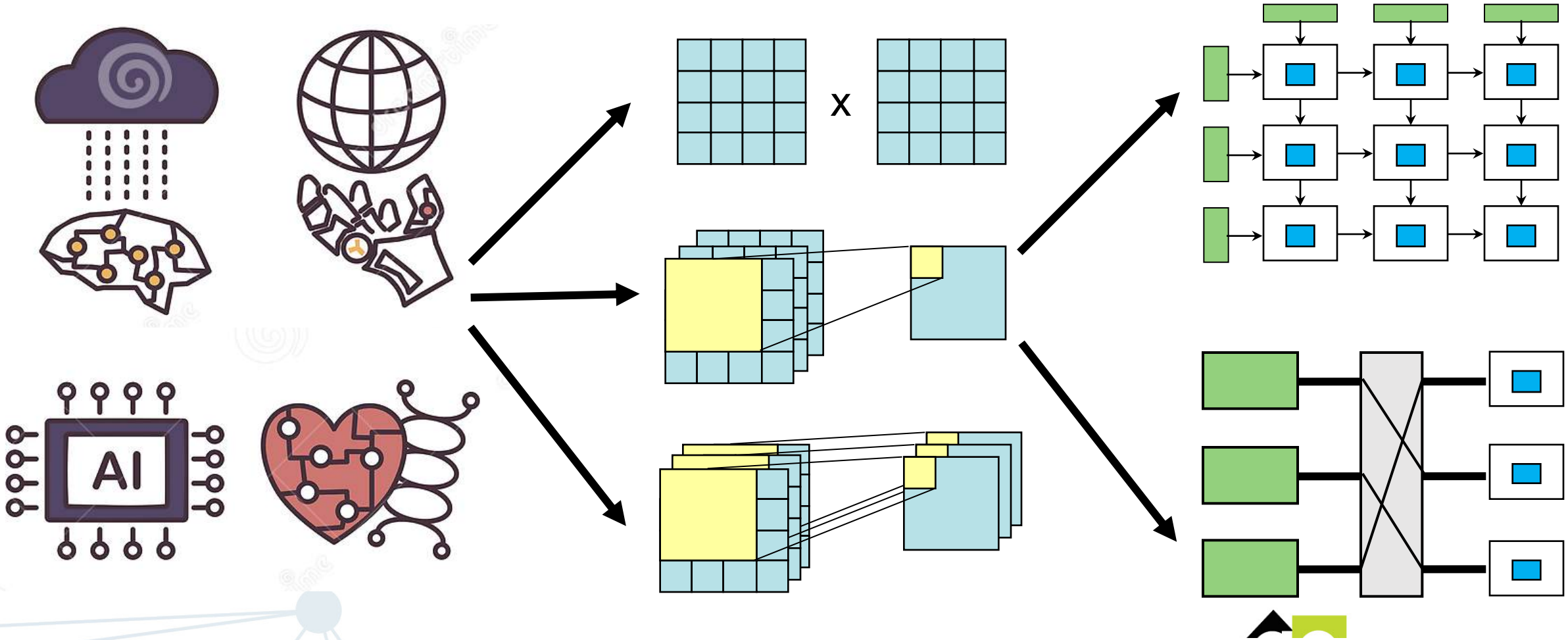


Crossbar



Rotation

The Challenge



Applications

Algorithms

Dataflow &
Interconnection choices

Outline

- Memory-level data reuse analysis
- Build memory data mapping
- Efficient PE-memory interconnection

Theory: Space-time Transformation

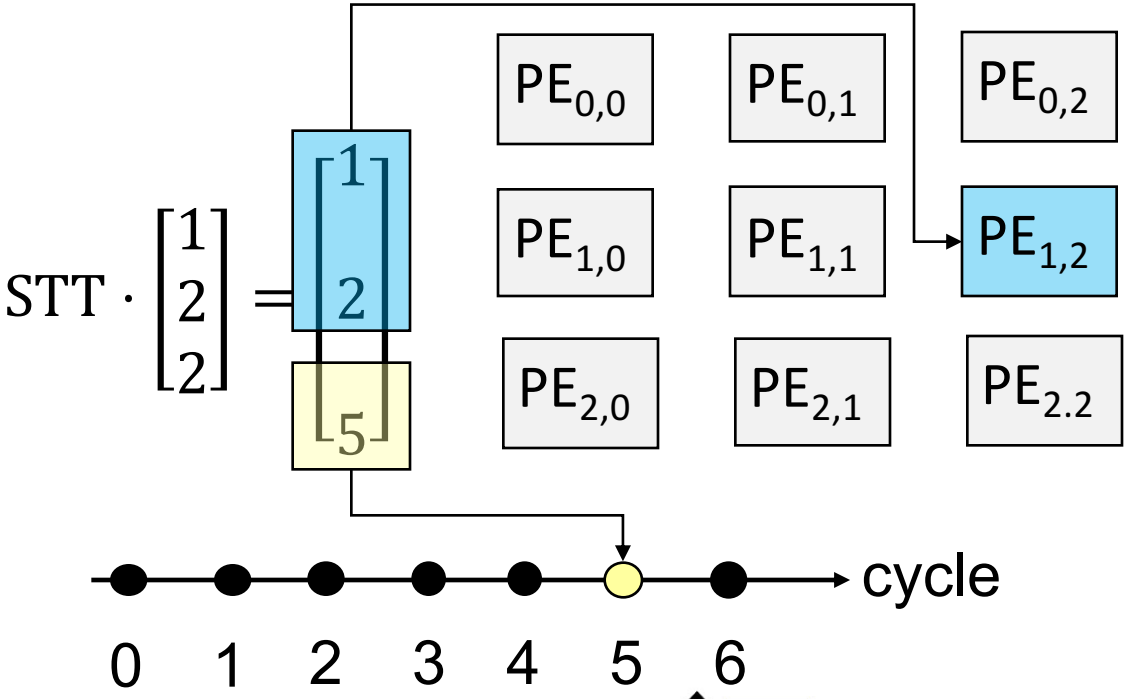
```
loop nest:
for  (i = 0; i < 3; i++)
  for  (j = 0; j < 3; j++)
    for  (k = 0; k < 3; k++)
      instance [i,j,k]:
      C[i,j] += A[i,k] * B[k,j];
```

Space-Time Transformation (STT):

- Transform loop iteration domain into hardware space and time
- PE array has two space dimension and one time dimension
- Can be expressed with Matrix Multiplication

Example:

$$STT \times \begin{bmatrix} i \\ j \\ k \end{bmatrix} = \begin{bmatrix} x \\ y \\ t \end{bmatrix} \quad STT = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$



STT With Multi-dimensional Time

GEMM : 3-D loop nest

```
for i = 1 : I
  for j = 1 : J
    for k = 1 : K
      MAC(C[i][j],A[i][k],B[k][j])
```

$$STT \times \begin{bmatrix} i \\ j \\ k \end{bmatrix} = \begin{bmatrix} x \\ y \\ t \end{bmatrix}$$

Conv2D : 6-D loop nest

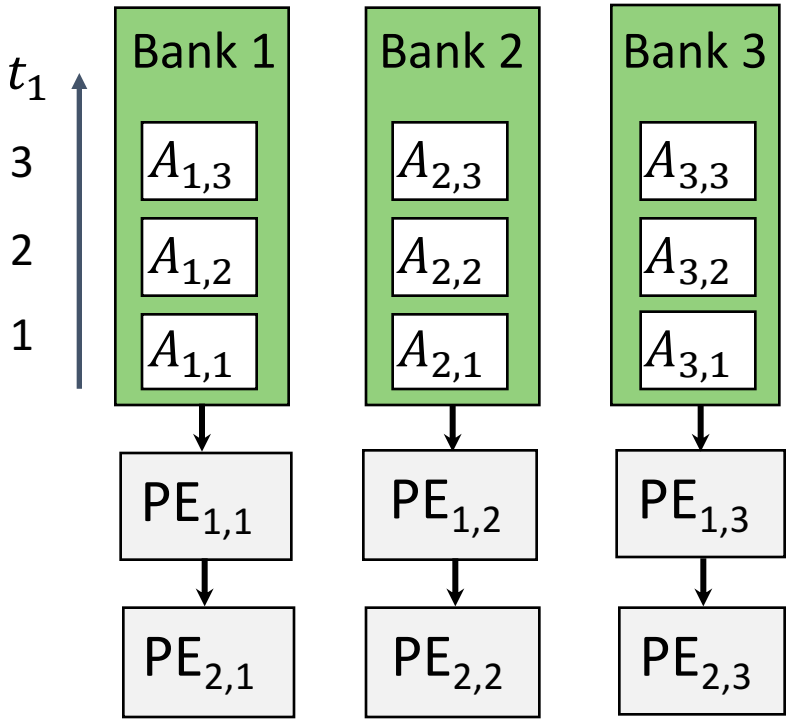
```
for c = 1 : C
  for h = 1 : H
    for w = 1 : W
      for k = 1 : K
        for kh = 1 : KH
          for kw = 1 : KW
            MAC(Output[k][h][w],
            Input[c][h+kh][w+kw],
            Filter[k][c][kh][kw])
```

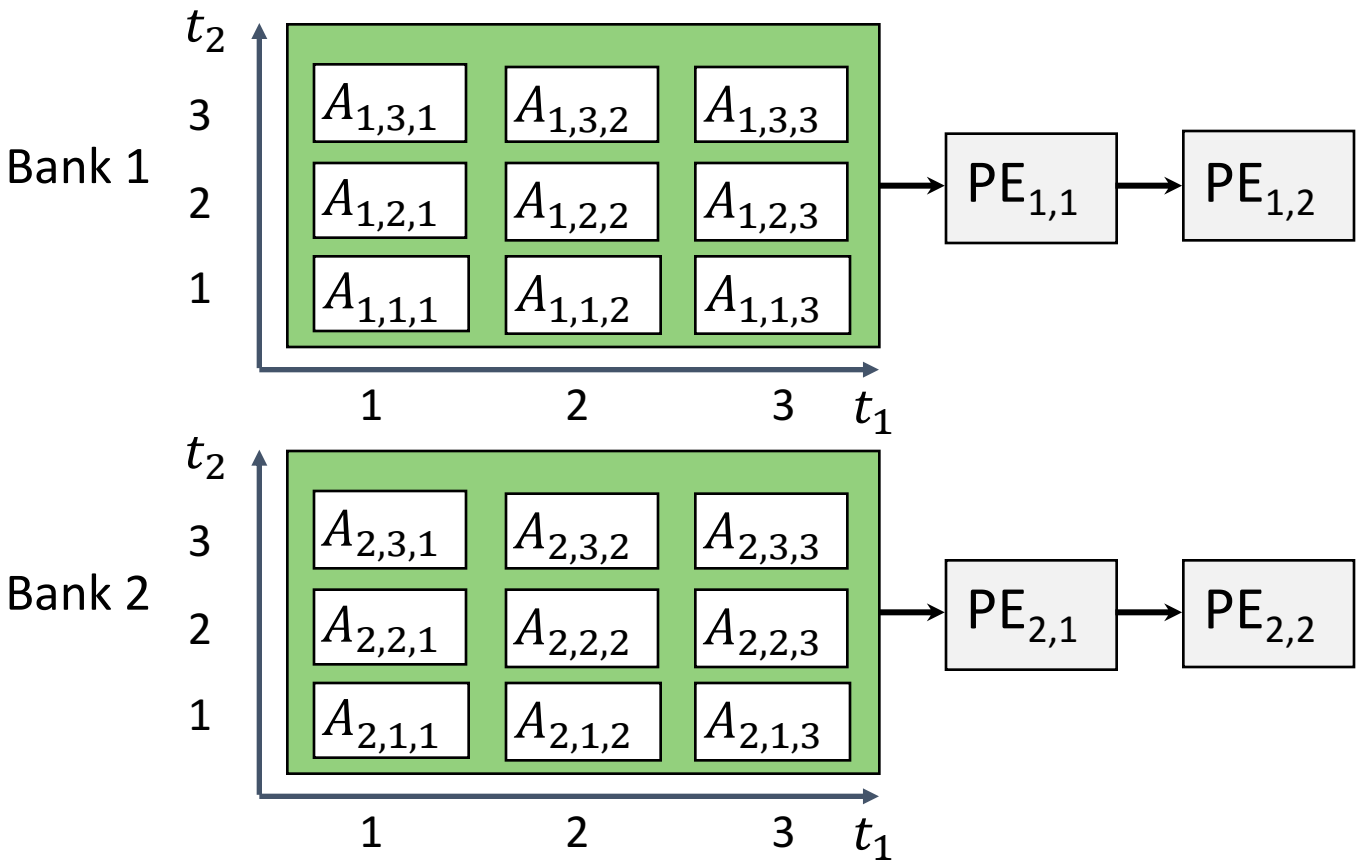
$$STT \times \begin{bmatrix} c \\ h \\ w \\ k \\ kh \\ kw \end{bmatrix} = \begin{bmatrix} x \\ y \\ t_1 \\ t_2 \\ t_3 \\ t_4 \end{bmatrix}$$

Spatial part

Temporal part

Multidimensional Time in Spatial Accelerator

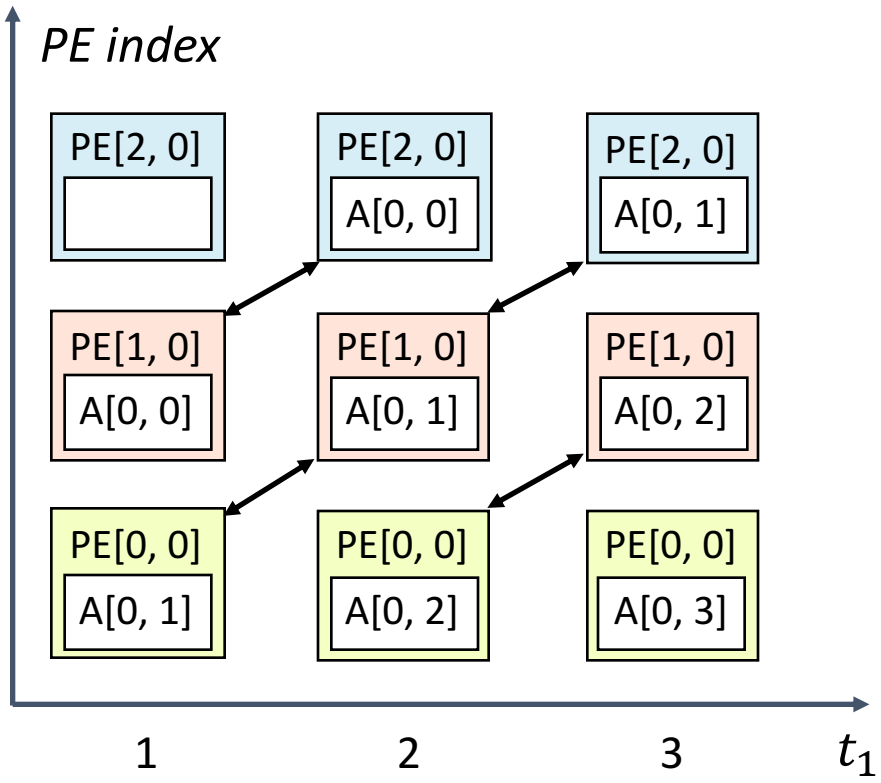




only 1 time dimension

2 time dimensions

Reuse Vector in space-time analysis



Definition: **Reuse vector** [*dx, dy, dt*]

- For any *x, y, t* ,Tensor index accessed at PE[*x, y*] cycle *t* equals to PE[*x+dx, y+dy*], cycle *t+dt*

- In this example,
  [*dx, dy, dt*]=[1, 0, 1]

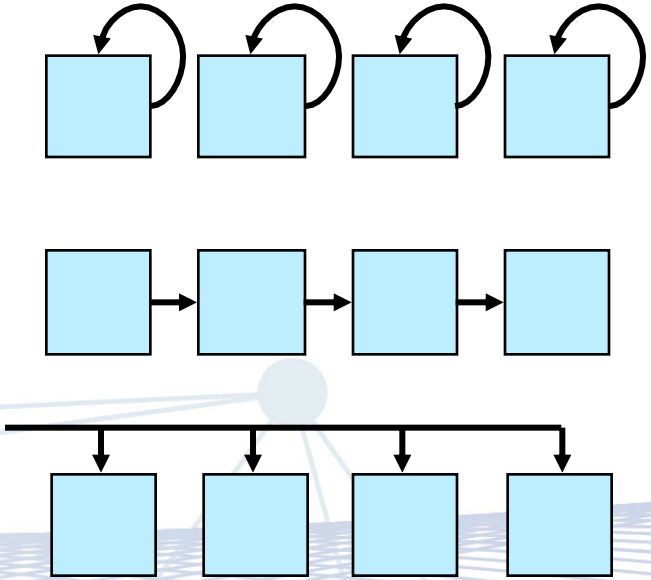
Reuse vector can be determined with STT

Dataflow and memory reuse in STT

Identify two types of reuse vector: dataflow and memory

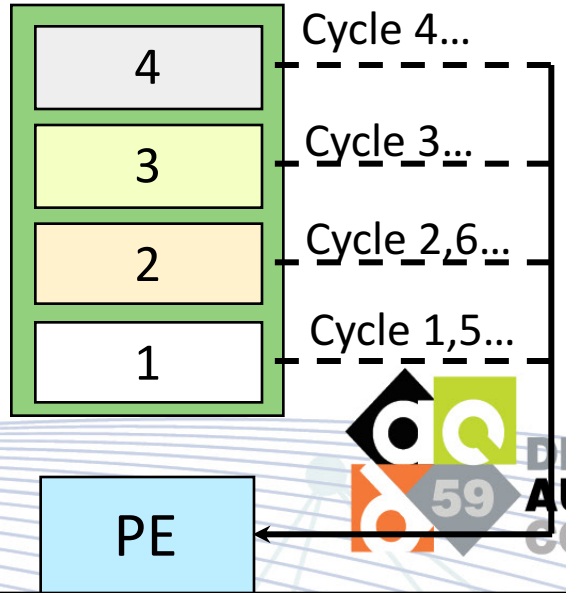
Dataflow reuse: Tensorlib

- Spatial part: same or different PE
- Temporal part: continuous time
- Implementation: PE array dataflow



Memory reuse: EMS

- Spatial part: same PE
- Temporal part: discontinuous time
- Implementation: repeated access of memory address



Use STT to model memory-level reuse

Find reuse vector for different memory access patterns

cycle	(t_1, t_2)	index
1	(1, 1)	1
2	(2, 1)	2
3	(3, 1)	3
4	(4, 1)	4
5	(1, 2)	1
6	(2, 2)	2
	…	…

Reuse vector $[t_1, t_2] = [0, 1]$

cycle	(t_1, t_2)	index
1	(1, 1)	1
2	(2, 1)	2
3	(3, 1)	3
4	(4, 1)	4
5	(1, 2)	2
6	(2, 2)	3
	…	…

$[t_1, t_2] = [1, -1]$

cycle	(t_1, t_2)	index
1	(1, 1)	1
2	(2, 1)	1
3	(3, 1)	1
4	(4, 1)	1
5	(1, 2)	2
6	(2, 2)	2
	…	…

$[t_1, t_2] = [1, 0]$
(Stationary)

Spatial part in reuse vector = 0

Building data mapping for on-chip memory

Step 1: Loop transformation

```
for c = 0 to C-1
 for y = 0 to Y-1
  for x = 0 to X-1
   for fx = 0 to FX-1
    ...
     Compute statement
```

DDR Level

```
for c_o = 0 to C_O-1
 for y_o = 0 to Y_O-1
  for x_o = 0 to X_O-1
```

SRAM Level

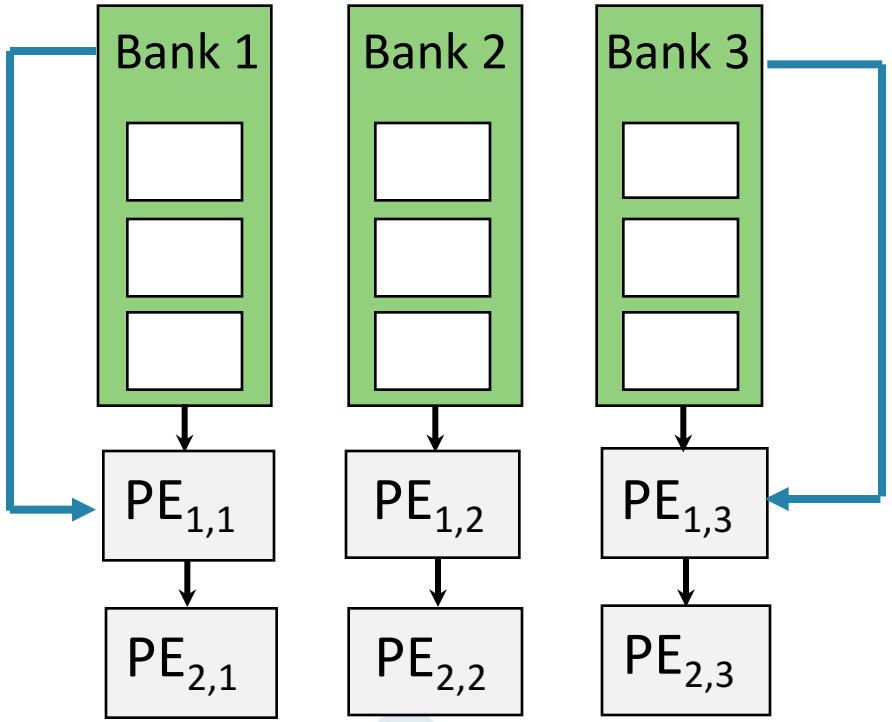
```
for c_i = 0 to C_I-1
 for y_i = 0 to Y_I-1
  for x_i = 0 to X_I-1
   for fx = 0 to FX-1
    ...
```

PE Level

```
Compute statement
```

Building data mapping for memory banks

Step 2: Memory bank binding



Bank ID	PE index
1	$PE_{1,1}$
2	$PE_{1,2}$
3	$PE_{1,3}$
…	…

Building data mapping for memory banks

Step 3: Memory data mapping

```
for every memory banks:
 for t1 in 0:T1
  for t2 in 0:T2
    …
    calc tensor index with
the formula
```

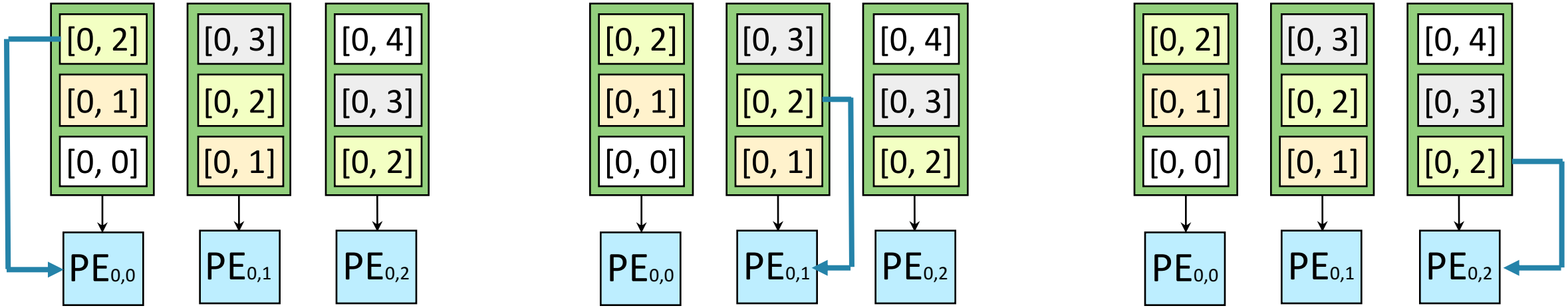
$$Data\ idx = TA \cdot STT^{-1} \begin{bmatrix} x \\ y \\ t_1 \\ t_2 \\ \dots \end{bmatrix}$$

PE index bound with memory bank index

Temporal loop iterator index

Inter-bank data reuse

Different PE, reuse in discontinuous time steps



$t_1 = 1, t_2 = 1$ $\quad$ $t_1 = 1, t_2 = 2$ $\quad$ $t_1 = 1, t_2 = 3$

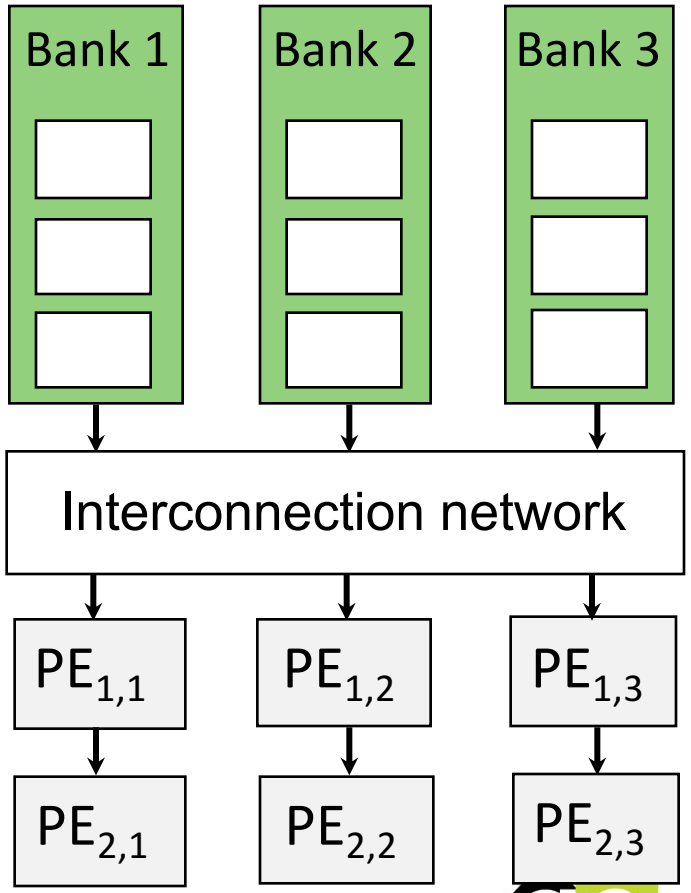
Implementing inter-bank reuse

Requirement

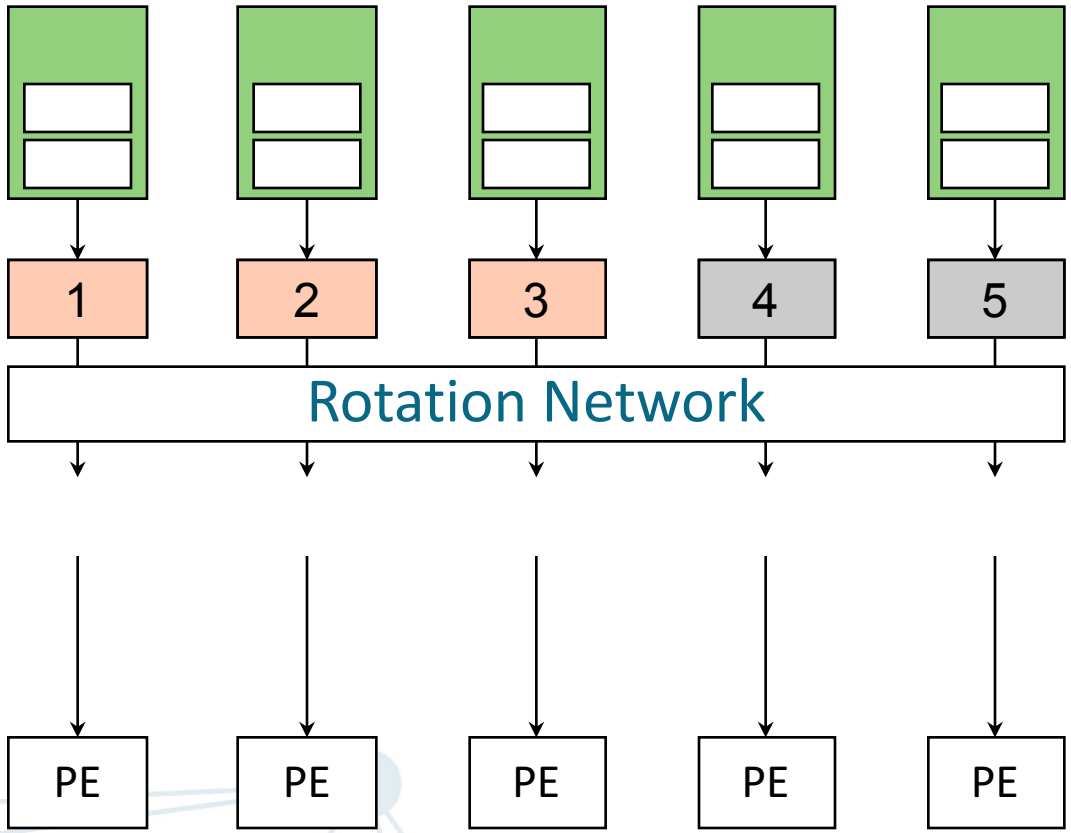
- Data transfer in same direction
- Small hardware cost

Solution

- rotation (shuffle) network

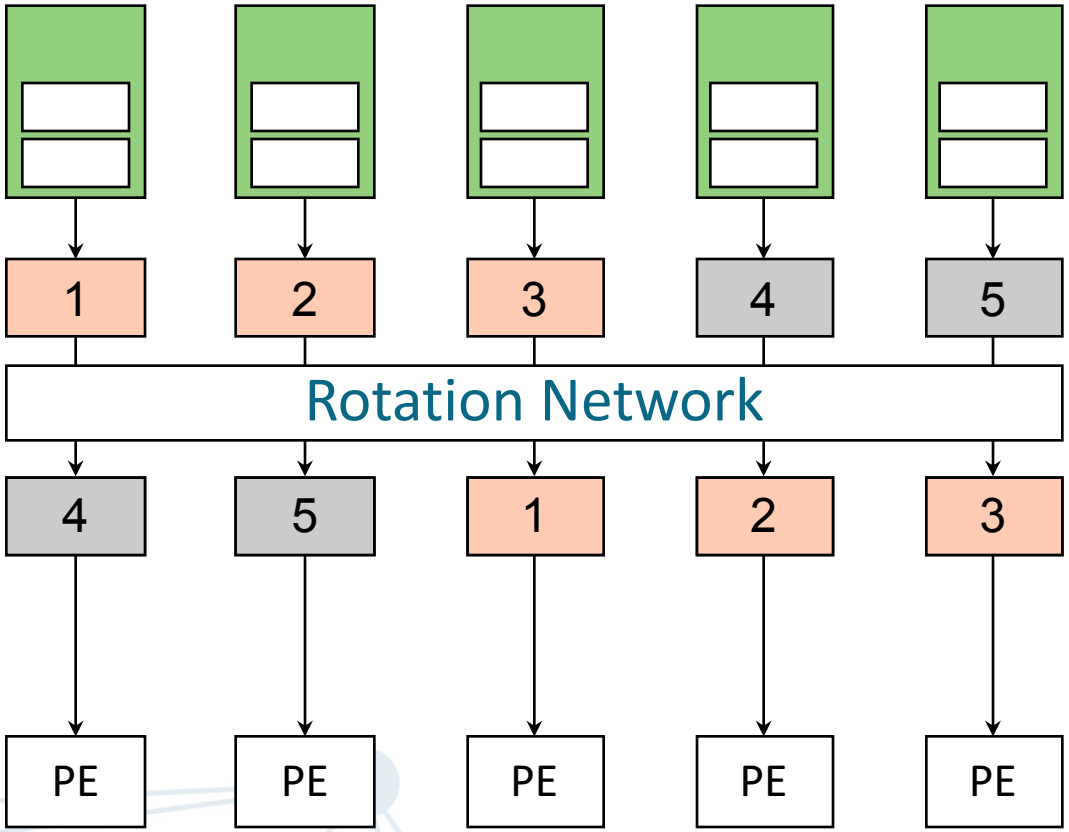


Rotation (Shuffle) Network



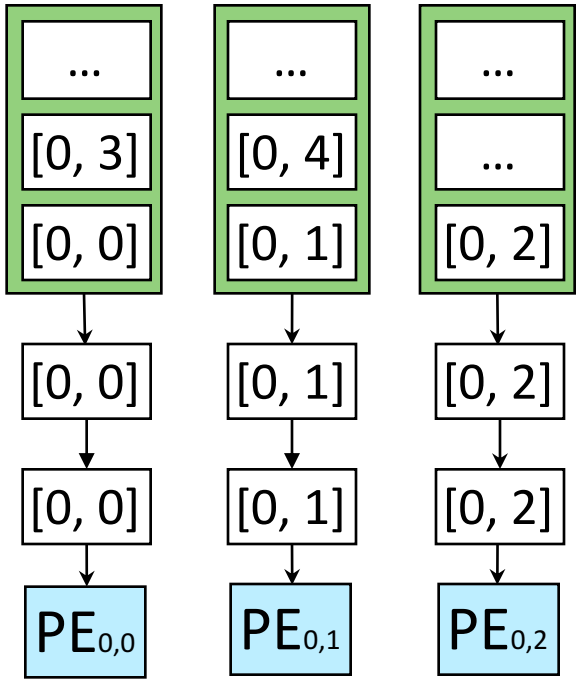
- Data from memory banks shift in the rotation network with the same length
- Overflowed part is filled to the end
- The shift length changes in different cycles

Rotation (Shuffle) Network



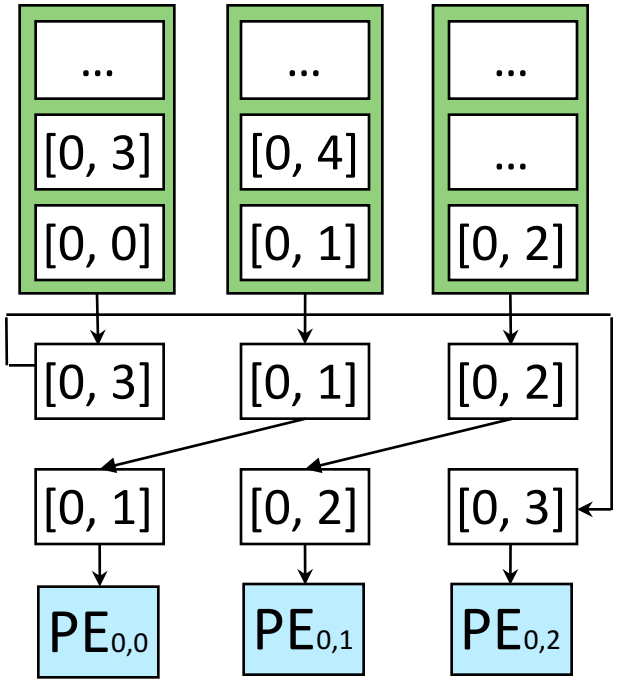
- Data from memory banks shift in the rotation network with the same length
- Overflowed part is filled to the end
- The shift length changes in different cycles

Rotation Network



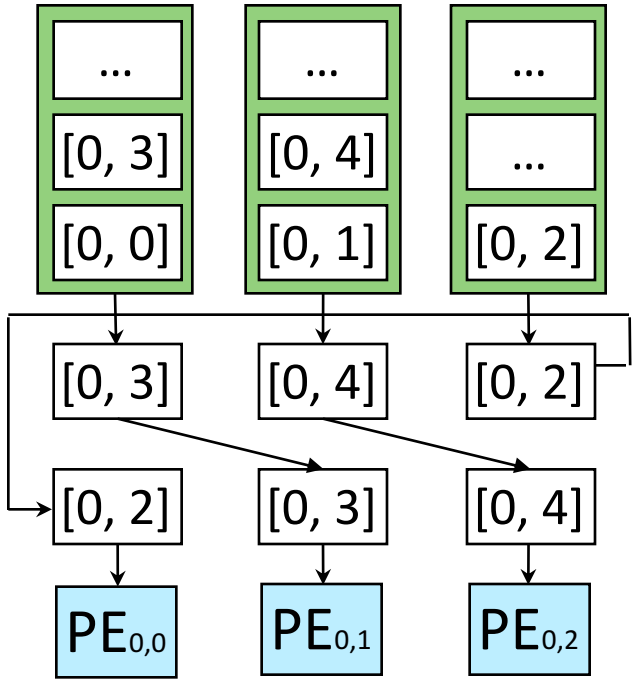
Cycle index	$PE_{0,0}$	$PE_{0,1}$	$PE_{0,2}$	Rotate
[0, 0]	[0, 0]	[0, 1]	[0, 2]	0
[0, 1]	[0, 1]	[0, 2]	[0, 3]	1
[0, 2]	[0, 2]	[0, 3]	[0, 4]	2
[1, 0]	[1, 0]	[1, 1]	[1, 2]	0
[1, 1]	[1, 1]	[1, 2]	[1, 3]	1

Rotation Network



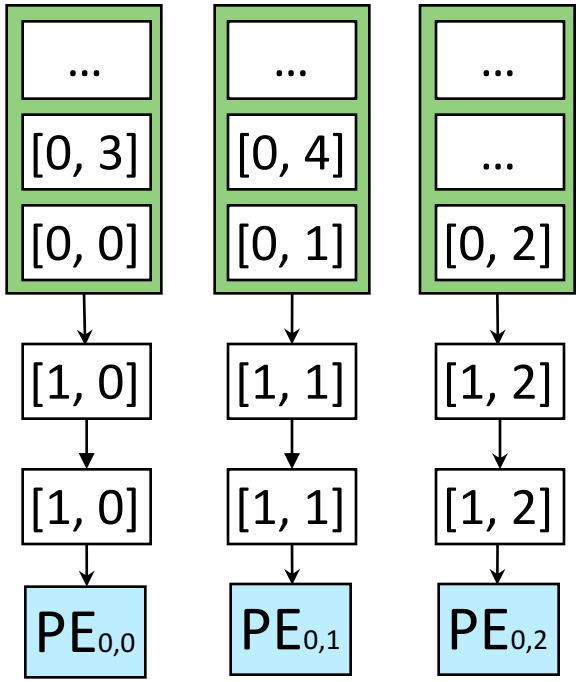
Cycle index	$PE_{0,0}$	$PE_{0,1}$	$PE_{0,2}$	Rotate
[0, 0]	[0, 0]	[0, 1]	[0, 2]	0
[0, 1]	[0, 1]	[0, 2]	[0, 3]	1
[0, 2]	[0, 2]	[0, 3]	[0, 4]	2
[1, 0]	[1, 0]	[1, 1]	[1, 2]	0
[1, 1]	[1, 1]	[1, 2]	[1, 3]	1

Rotation Network



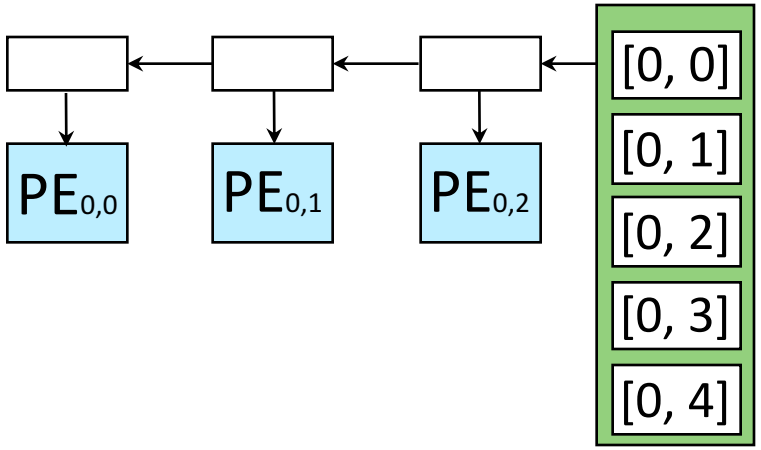
Cycle index	$PE_{0,0}$	$PE_{0,1}$	$PE_{0,2}$	Rotate
[0, 0]	[0, 0]	[0, 1]	[0, 2]	0
[0, 1]	[0, 1]	[0, 2]	[0, 3]	1
[0, 2]	[0, 2]	[0, 3]	[0, 4]	2
[1, 0]	[1, 0]	[1, 1]	[1, 2]	0
[1, 1]	[1, 1]	[1, 2]	[1, 3]	1

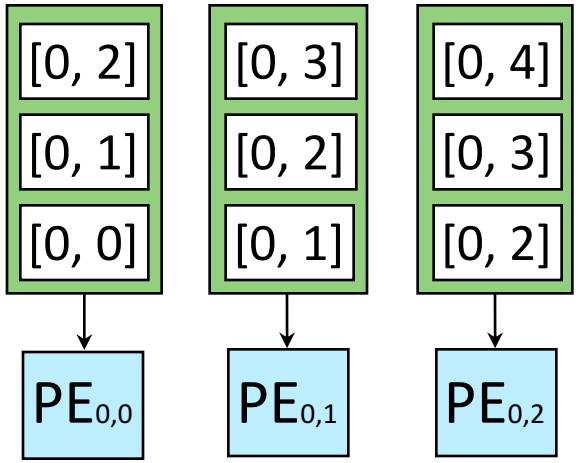
Rotation Network

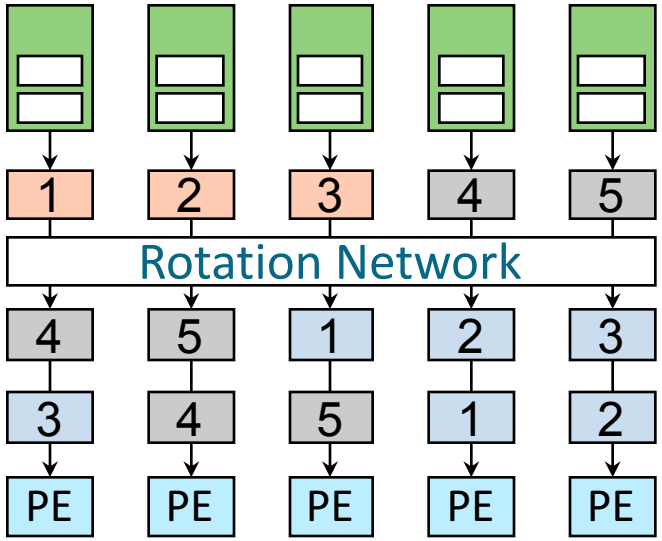


Cycle index	$PE_{0,0}$	$PE_{0,1}$	$PE_{0,2}$	Rotate
[0, 0]	[0, 0]	[0, 1]	[0, 2]	0
[0, 1]	[0, 1]	[0, 2]	[0, 3]	1
[0, 2]	[0, 2]	[0, 3]	[0, 4]	2
[1, 0]	[1, 0]	[1, 1]	[1, 2]	0
[1, 1]	[1, 1]	[1, 2]	[1, 3]	1

Comparison with other interconnections







Systolic interconnection

Only 1 memory bank ✓
Large pipeline overhead ✗
No data duplication ✓

Direct interconnection

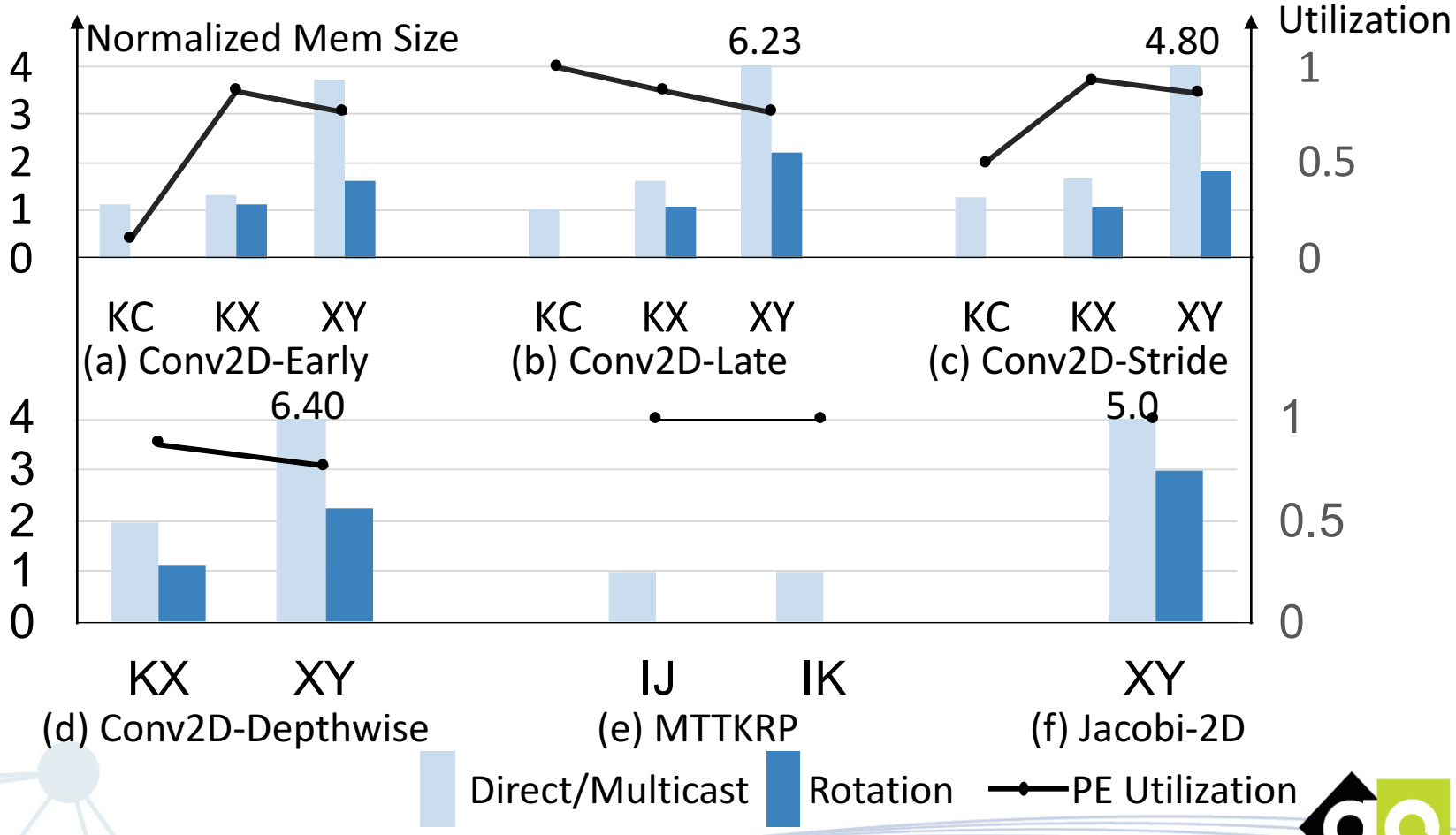
Many memory banks ✗
Small pipeline overhead ✓
Data duplication ✗

Rotation interconnection

Many memory banks ✗
Small pipeline overhead ✓
No data duplication ✓
Support inter-bank reuse ✓

Experiment Results

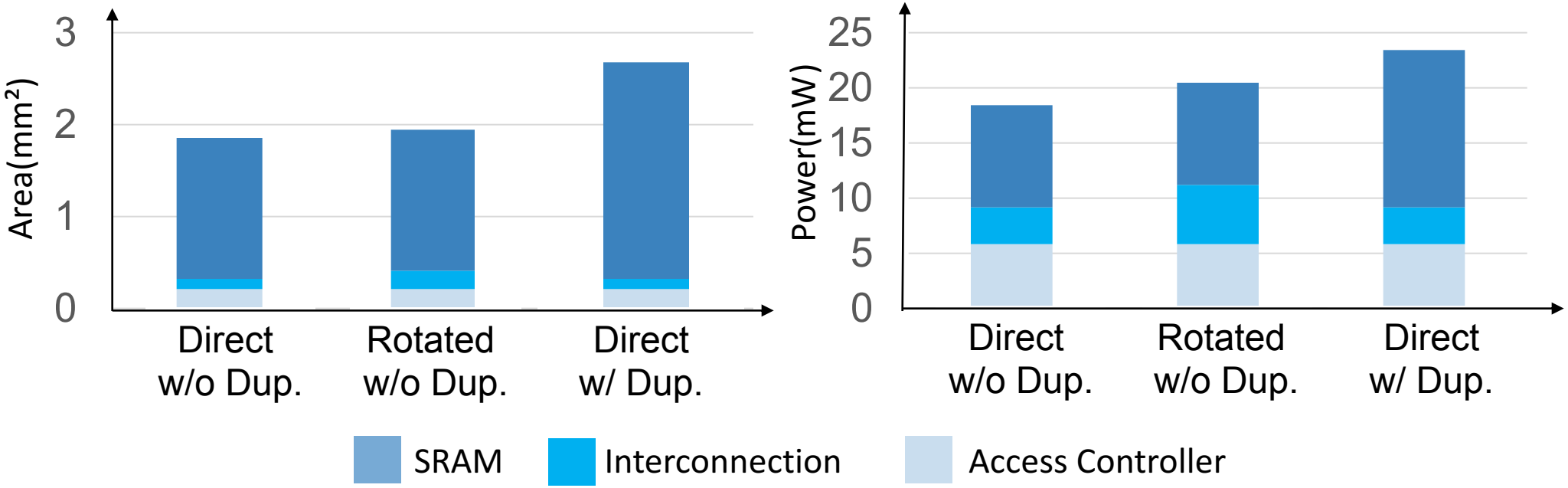
Memory size reduction



28% memory size reduction with rotation network

Experiment Results

Hardware overhead evaluation



13% power reduction, 28% area reduction

Experiment Results

FPGA Performance

	Susy [ICCAD'20]	AutoSA [FPGA'21]	Tensorlib [DAC'21]	EMS-WS	EMS-OS
LUT(%)	35%	58%	73%	76%	83%
DSP(%)	84%	77%	75%	73%	73%
BRAM(%)	30%	30%	73%	53%	53%
Freq(MHz)	220	272	245	301	295
Throughput (Gop/s)	551	950	626	731	717

10% frequency optimization with physical optimization

Summary

- EMS: Efficient Memory Subsystem Synthesis for Spatial Accelerators

- Memory-level data reuse analysis
  - Extended space-time transformation to model memory-level reuse
- Build memory data mapping
  - Based on STT analysis result
- Efficient PE-memory interconnection
  - Support direct, multicast and rotation interconnection