Balancing Efficiency and Flexibility for DNN Acceleration via Temporal GPU-Systolic Array Integration

Cong Guo[1], Yangjie Zhou[1], Jingwen Leng[1], Yuhao Zhu[2], Zidong Du[3],
Quan Chen[1], Chao Li[1], Bin Yao[1], Minyi Guo[1]

[1]Shanghai Jiao Tong University, [2]University of Rochester,
[3]Institute of Computing Technology, Chinese Academy of Sciences

Biography

- Cong Guo
  - First-year Ph.D. student at Shanghai Jiao Tong University
  - Interested in computer architecture and high performance computing.

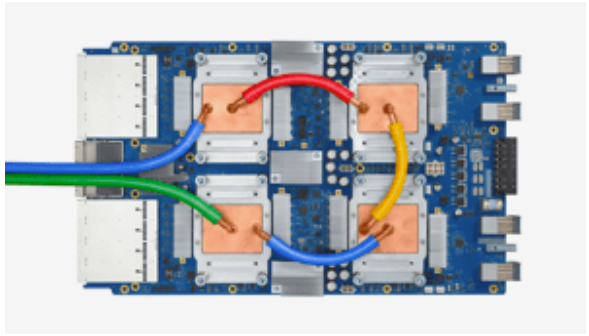
- Jingwen Leng
  - Associate professor
  - John Hopcroft Center for Computer Science
    Department of Computer Science and Engineering
    Shanghai Jiao Tong University

Outline

- Introduction
- Simultaneous Multi-mode Architecture
- Evaluation

Introduction

Efficiency



Google TPU [ISCA'16]

Specialized for

GEMM (General Matrix Multiply)

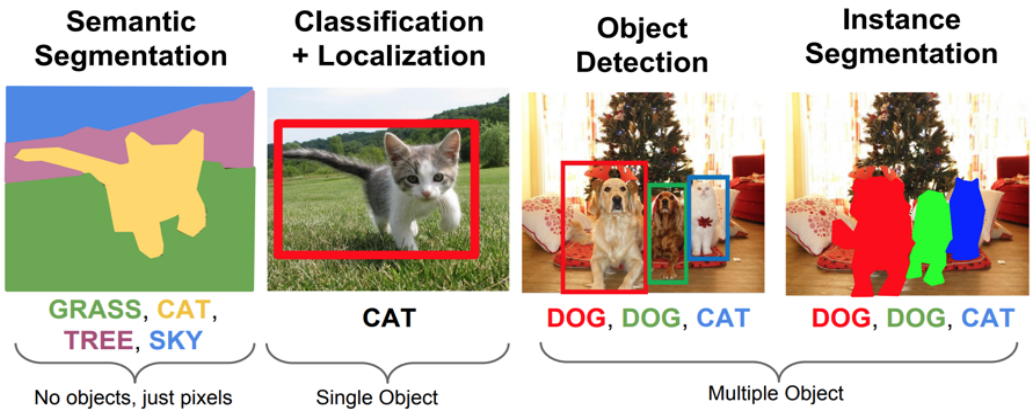
Nvidia GPU [Volta, 17]

General-purpose Core

Flexibility

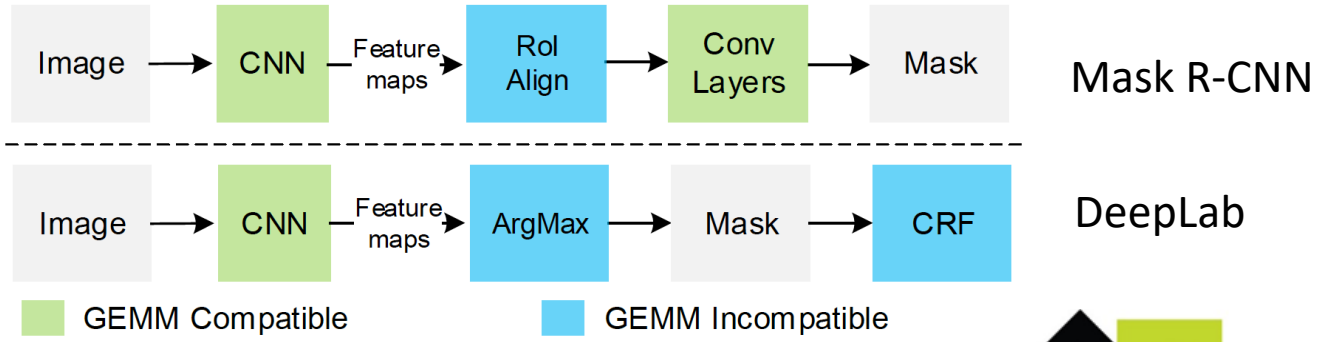
Inefficiency for TPU

- TPU v2, 1 core, 22TFLOPS
- GPU V100, 15TFLOPS



- TPU profiling
  - Mask R-CNN
    - CNN/FC: 20% faster than GPU
    - Total: 75% slower than GPU
  - DeepLab
    - CNN/FC: 40% faster than GPU
    - ArgMax: 2x slowdown than GPU
    - CRF: 10x worse than GPU

Hybrid models:

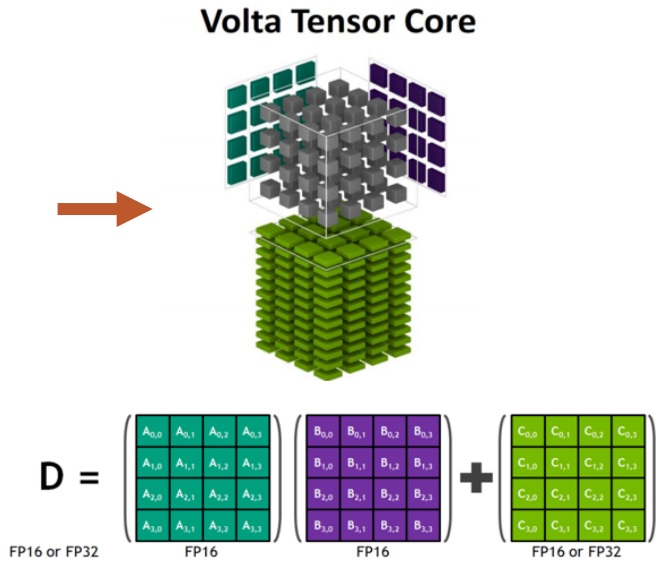


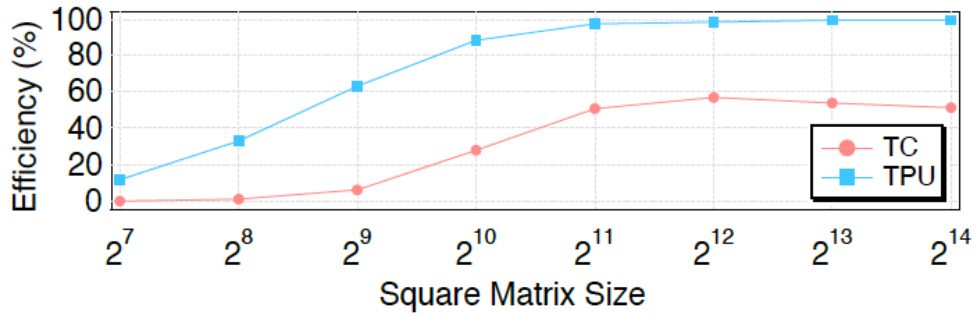
Inefficiency for GPU

- Spatial integration
- Explicit synchronization
- Fixed shape GEMM

- Performance inefficiency

- Tensor Core
  - Efficiency < 60%
- TPU
  - Efficiency > 99%





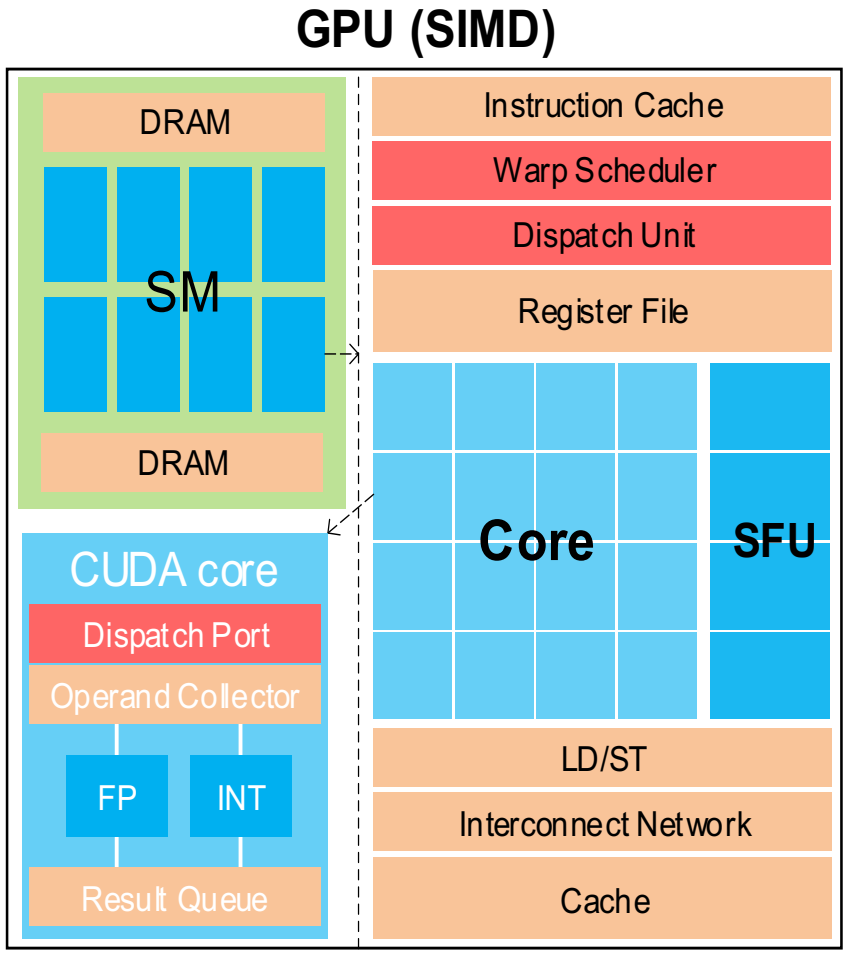
Efficiency: Achieved FLOPS divided by the peak FLOPS

Outline

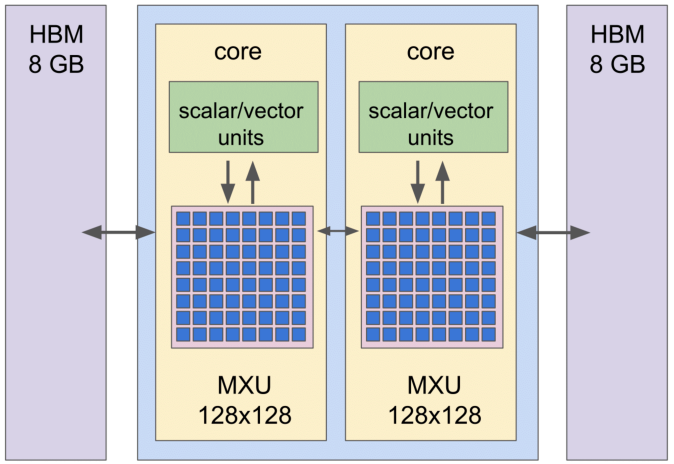
- Introduction
- Simultaneous Multi-mode Architecture
- Evaluation

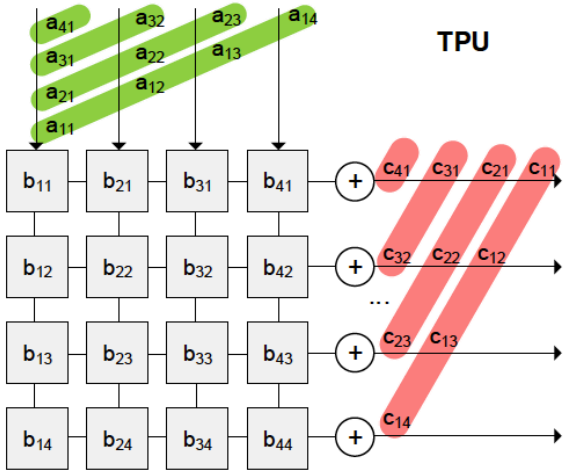
GPU

- GPU
- Papalism
  - Single Instruction Multiple Data
  - Massive threads
  - Warp active mask
- Memory
  - Register file, vector access
  - Shared memory, scalar access
- Communication
  - PE array with shared memory
    - Warp Shuffle Instruction



TPU





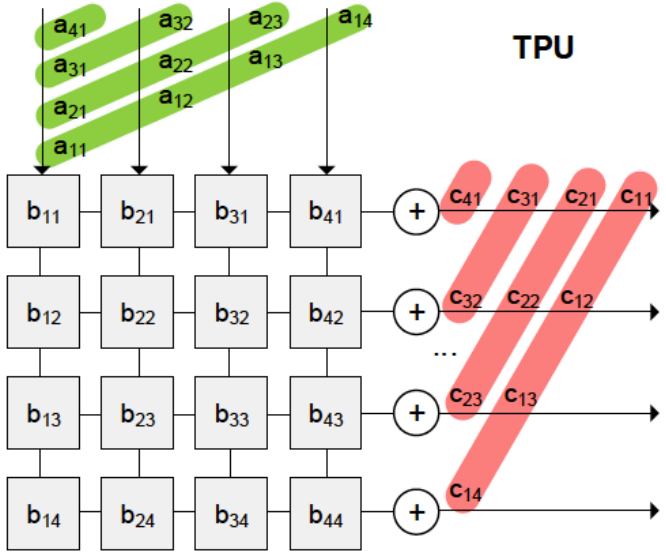
- TPU
- Papalism
  - 2D Systolic Array (MISD)
  - High concurrency
  - Active/Non-Active (one inst. two status)
- Memory
  - Weight, vector access (continuous)
  - Input/output, scalar access(discontinuous)
- Communication
  - Interconnected PE array

Similarity

- GPU
- Parallelism
  - Single Instruction Multiple Data
  - Massive threads
  - Warp active mask
- Memory
  - Register file, vector access
  - Shared memory, scalar access
- Communication
  - PE array with shared memory

- TPU
- Parallelism
  - 2D Systolic Array (MISD)
  - High concurrency
  - Active/Non-Active (one inst. two status)
- Memory
  - Weight, vector access (continuous)
  - Input/output, scalar access(discontinuous)
- Communication
  - Interconnected PE array

SMA Hardware Design



Similar to GPU

Challenges:

1: Massive output scalar accesses

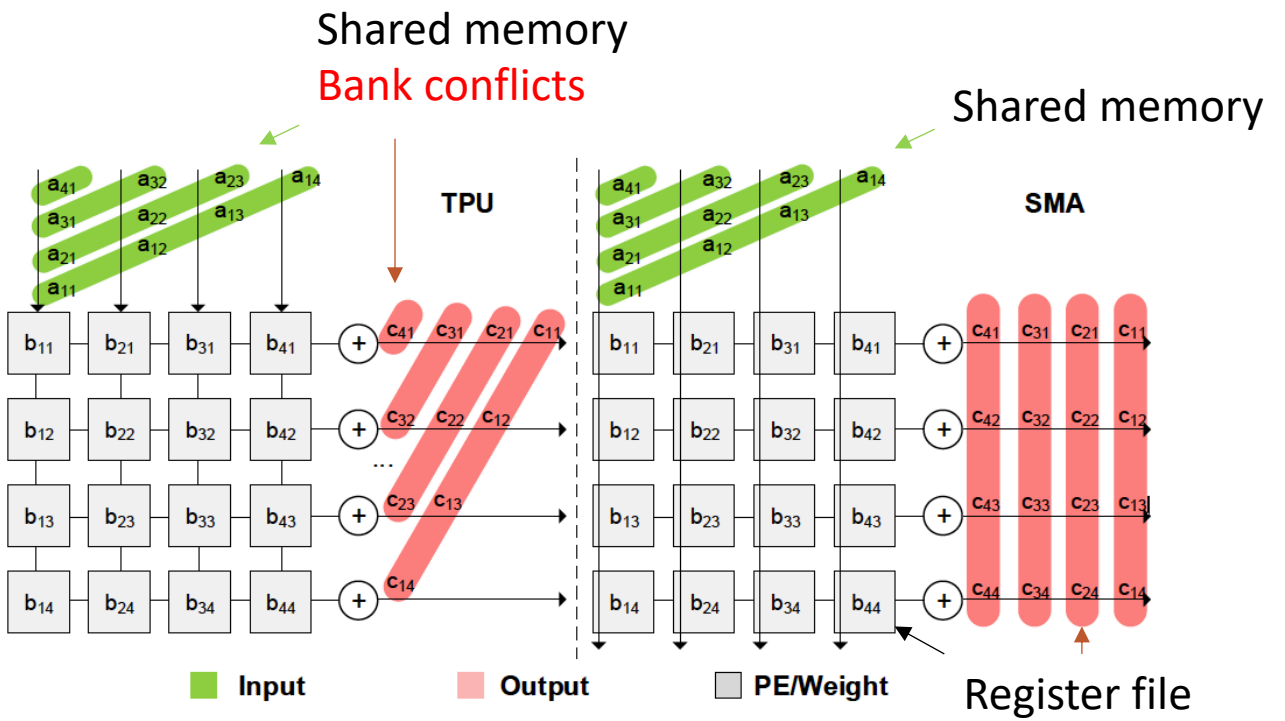
2: Inter-PE Communication

3: How to control the systolic array

Simultaneous Multi-mode Architecture (SMA)

Massive output scalar accesses

- Semi-broadcast
  - Weight
    - Preload
    - Register file
  - Output
    - Vector access
    - Register file
  - Input
    - Scalar access
    - Shared memory



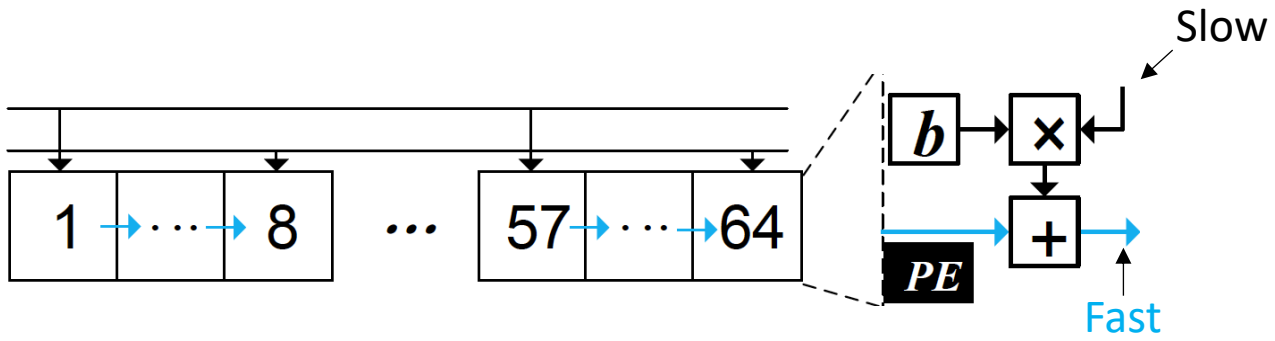
Weight-stationary

Semi-broadcasted
Weight-stationary

Partial Sum Communication

- One-way wires
  - Horizontal neighbor PEs
    - Fast
    - latency-sensitive
    - Negligible overhead
  - Vertical PEs
    - Slow
    - Broadcast, Prefetch
    - latency-insensitivity

Partial Sum need low latency



Without PE layout reconfiguration

Instruction Control

- A new instruction:
  - LSMA (Load, Store and Multiply-accumulate)
- A systolic controller

Per SM

Input : 8 x 2 x 4 Bytes

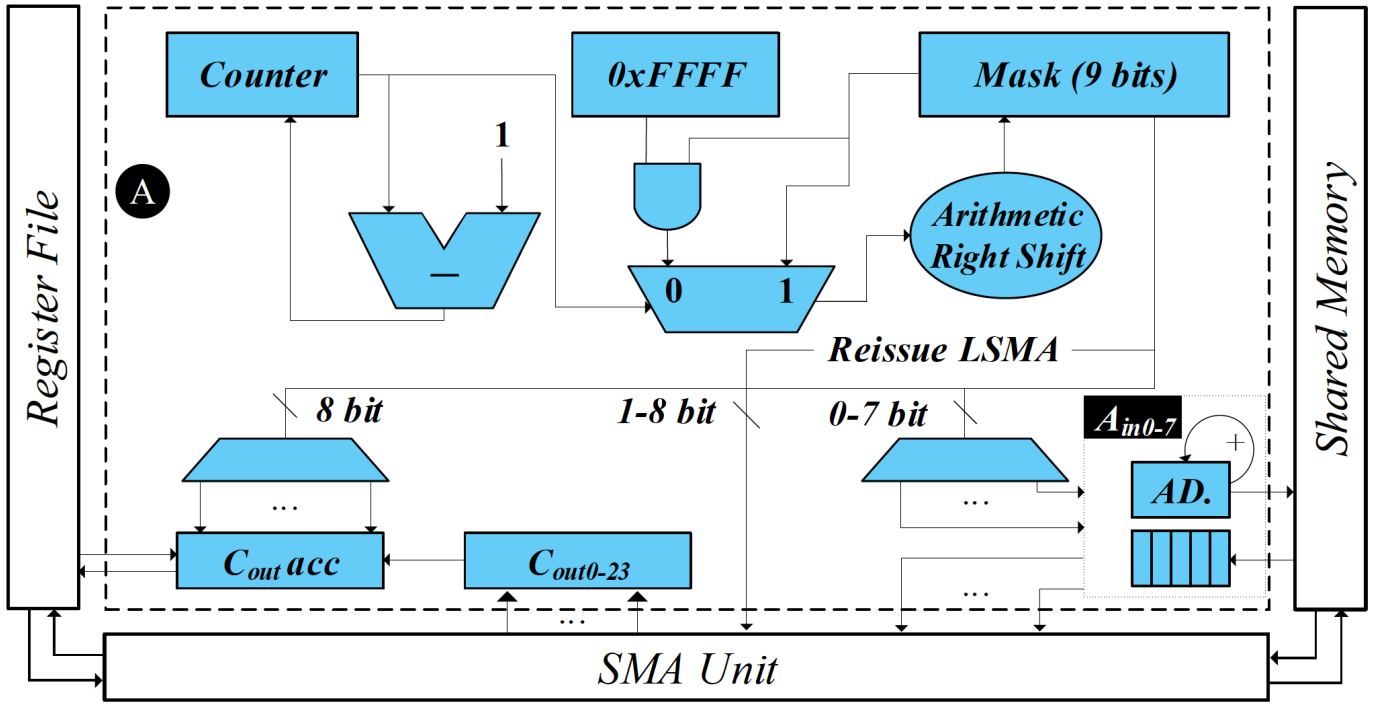
Output : 24 x 2 x 4 Bytes

Total : 256 Bytes

Area Overhead < 0.1%

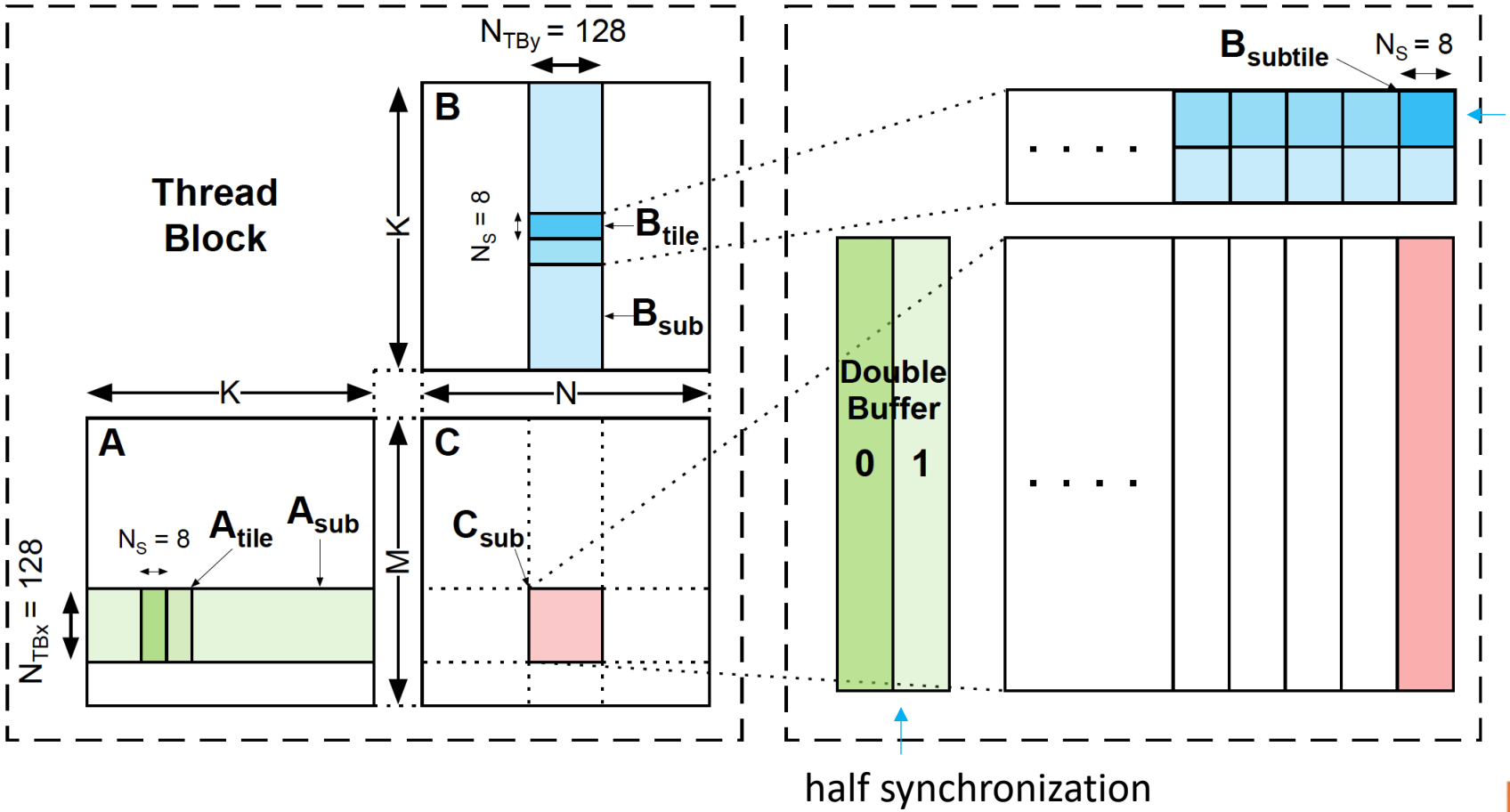
256KB Register file

128KB L1 Cache/Shared memory



$$\text{LSMA } B \Rightarrow C[out] \leftarrow A[in] \times B + C[in]$$

Software Design: Tiling GEMM



Based on CUTLASS 1.3

Outline

- Introduction
- Simultaneous Multi-mode Architecture
- Evaluation

Evaluation

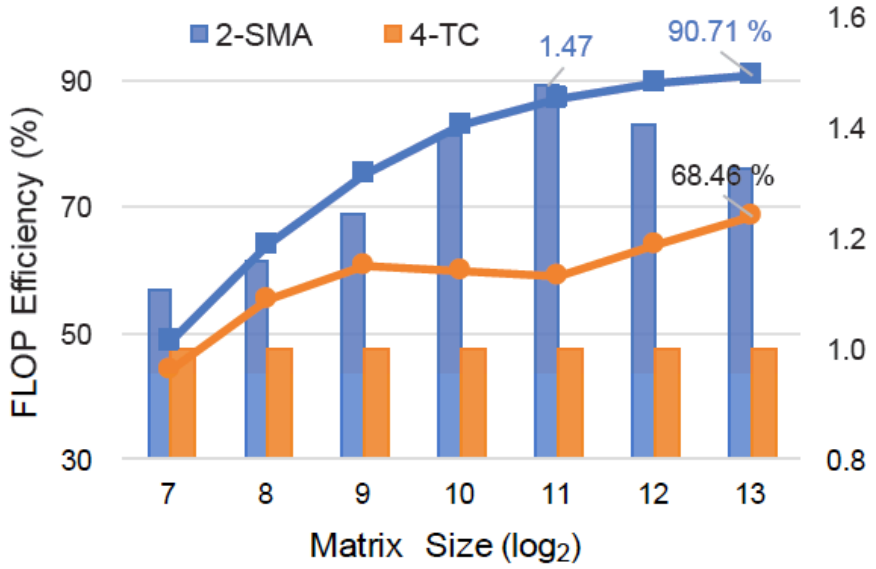
- Methodology
  - GPGPUsim-4.0
  - GEMM based on CUTLASS

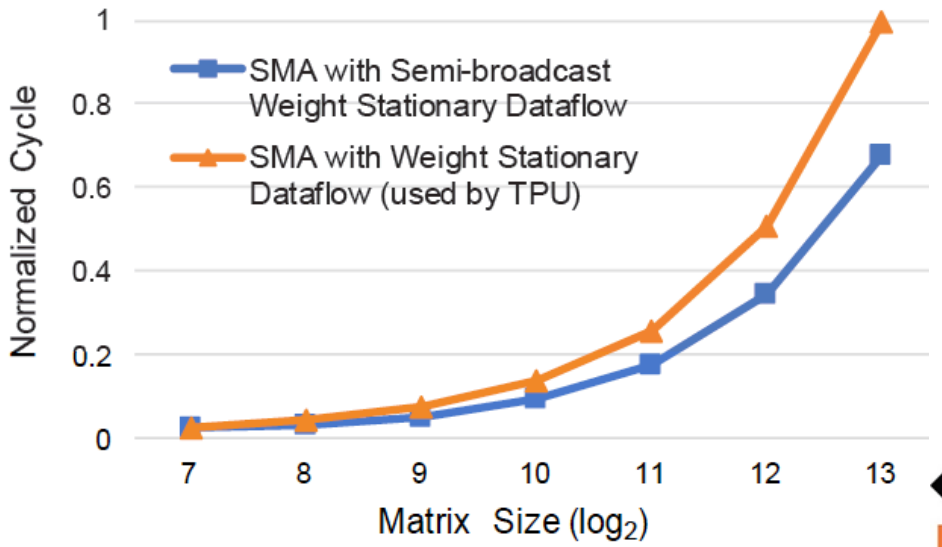
Platform	2-SMA	4-TC	3-SMA
Unit	2 SMA	4 TC	3 SMA
Precision	FP16	FP16	FP16
PE	256	256	384

	GPGPU	SMA
Baseline	Volta	Volta
SMs	80	80
CUDA Core/SM	64 FP32 units	3 8×8 SMA unit
Tensor Core/SM	4 (256 FP16 units)	
Shared Memory/SM	32 banks Configurable up to 96KB	32 banks (8 for all SMA units) Configurable up to 96 KB
Register File/SM	256 KB	256 KB

Iso-FLOP

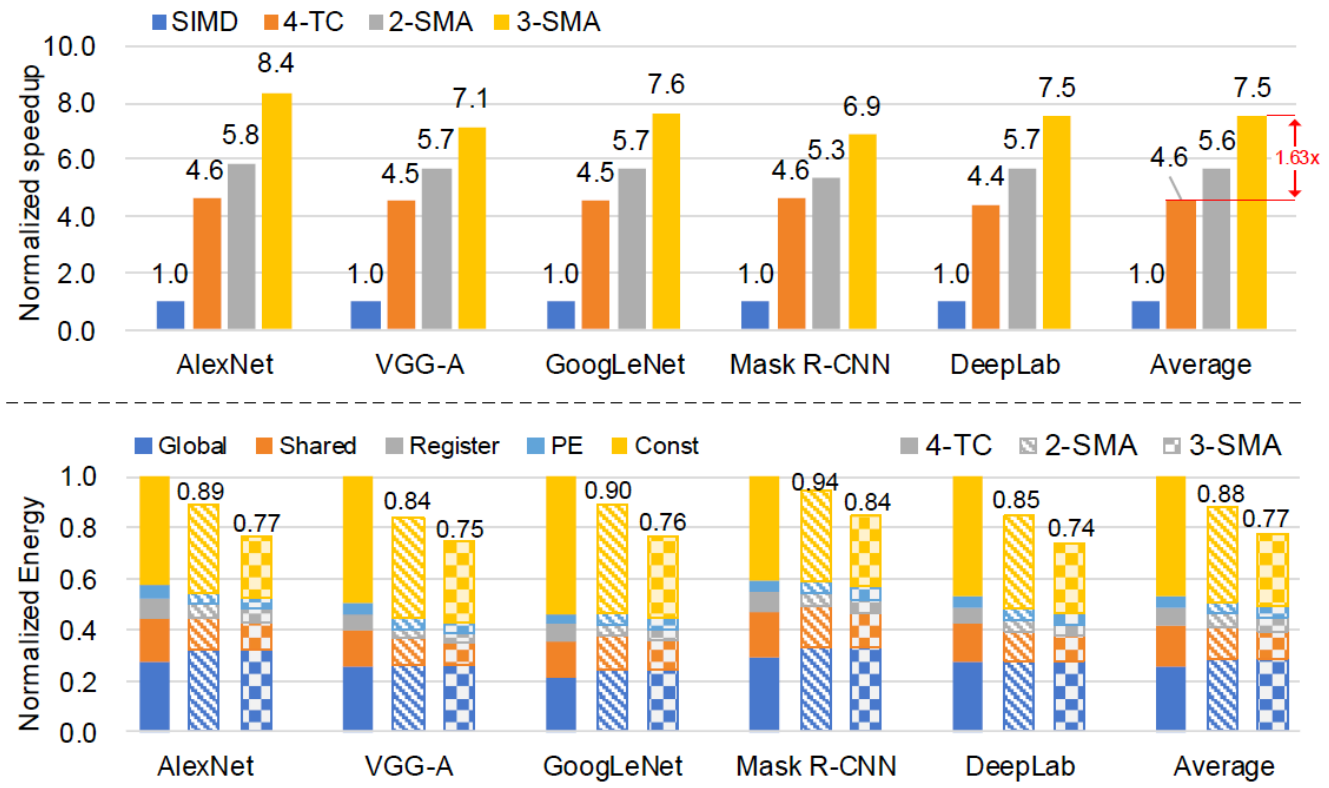
- Square GEMM
  - 2-SMA efficiency > 90%, 30% higher than 4-TC.
  - SMA (broadcast) 20% - 40% higher than non-broadcast.



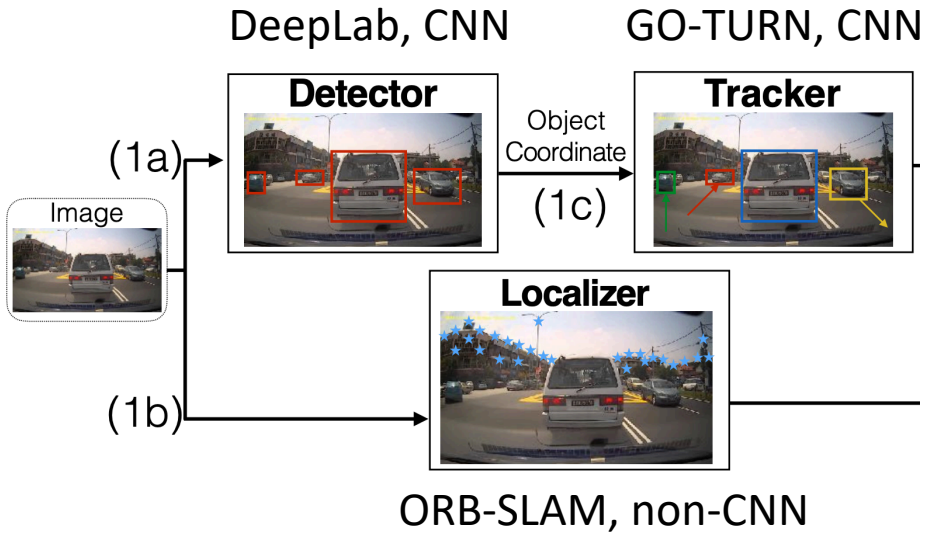


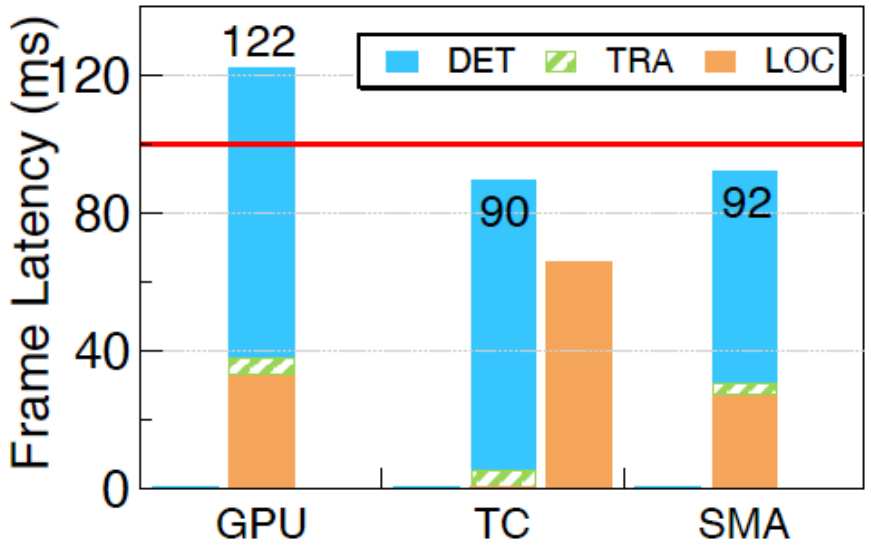
Iso-Area

- 5 networks
  - 3-SMA 63% faster, 23% less energy than 4-TC on average.



End-to-end application (autopilot)

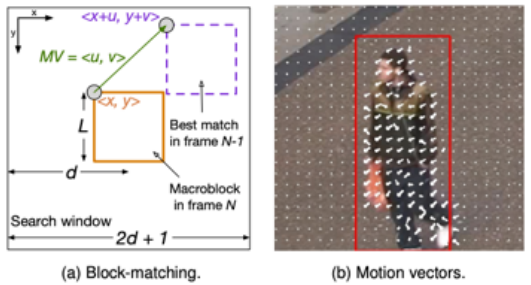


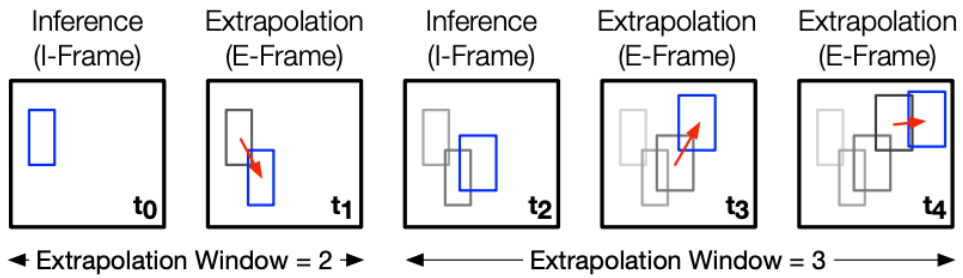


Same area	CUDA	CUDA + Tensor	SMA
GEMM Speedup	1.0x	0.5x + 1.0x	1.5x
Platform	GPU	TC	SMA

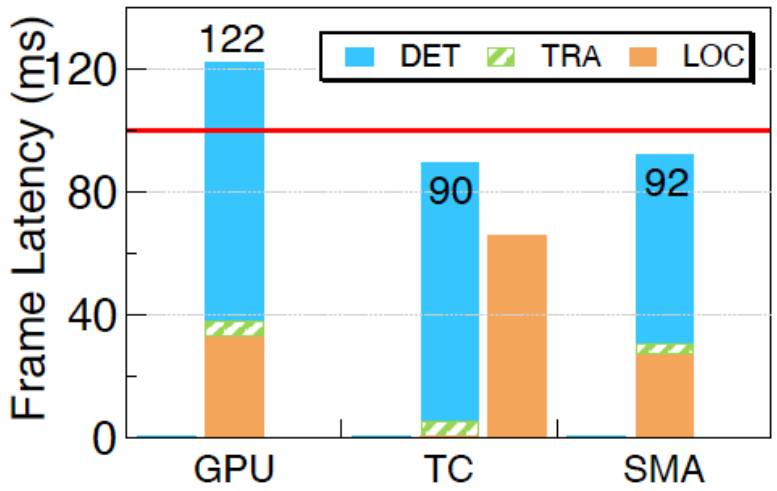
[Shih-Chieh Lin, etc.， ASPLOS'18]

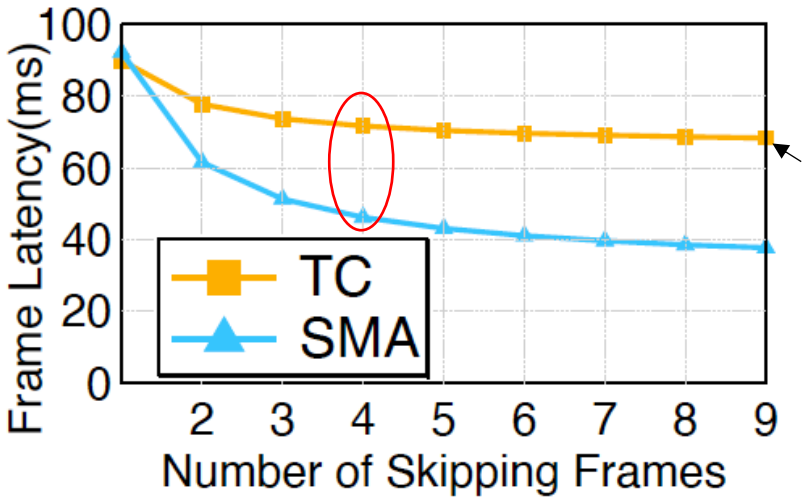
End-to-end application (autopilot)





[Euphrates, ISCA'18]





CUDA core non-GEMM Bottleneck

N = 4, SMA Latency Reduction 50%, More Flexibility

Summary

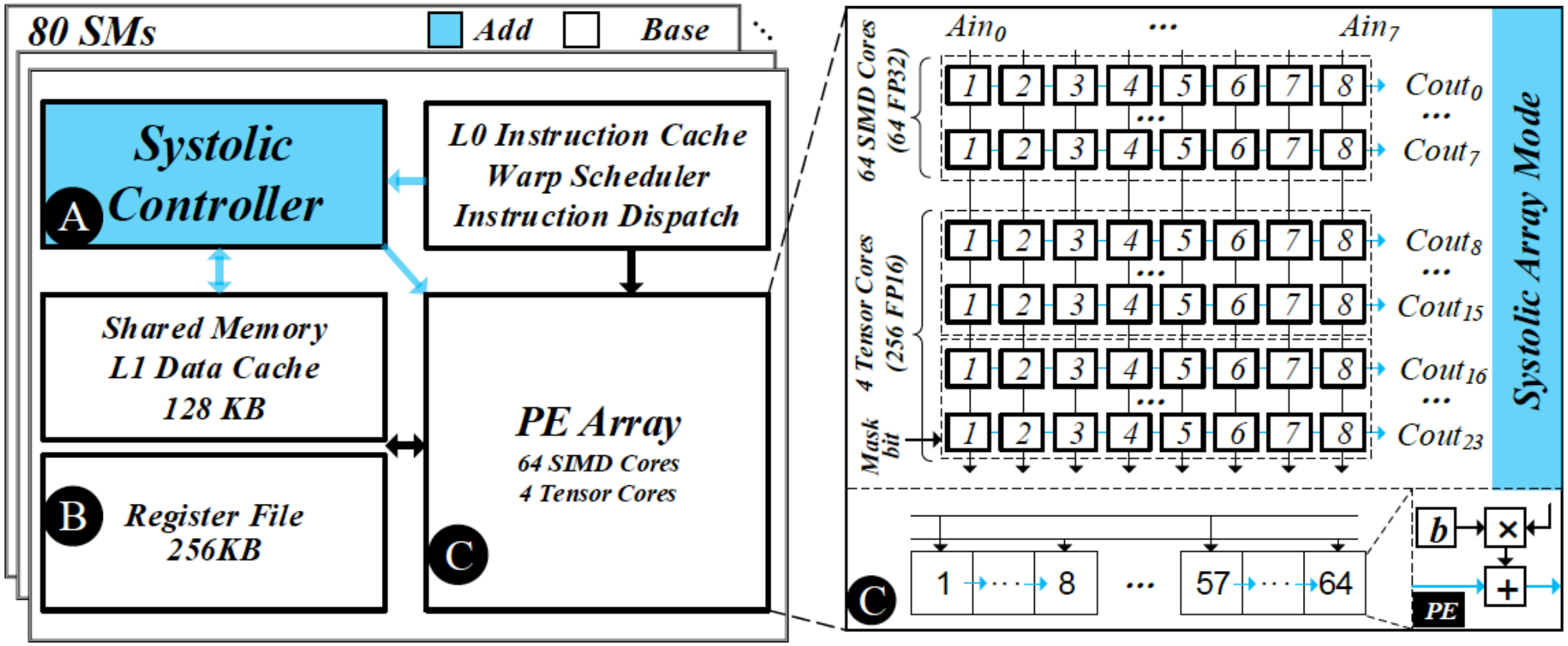
- Hardware
  - Parallelism similarity
  - Memory and communication
  - Systolic controller
- Software
  - Tiling GEMM
- Evaluation
  - Efficiency
  - Flexibility

Questions

Thank you!

Backup Slides

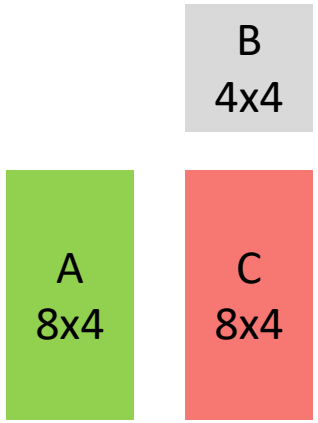
SMA Hardware Design

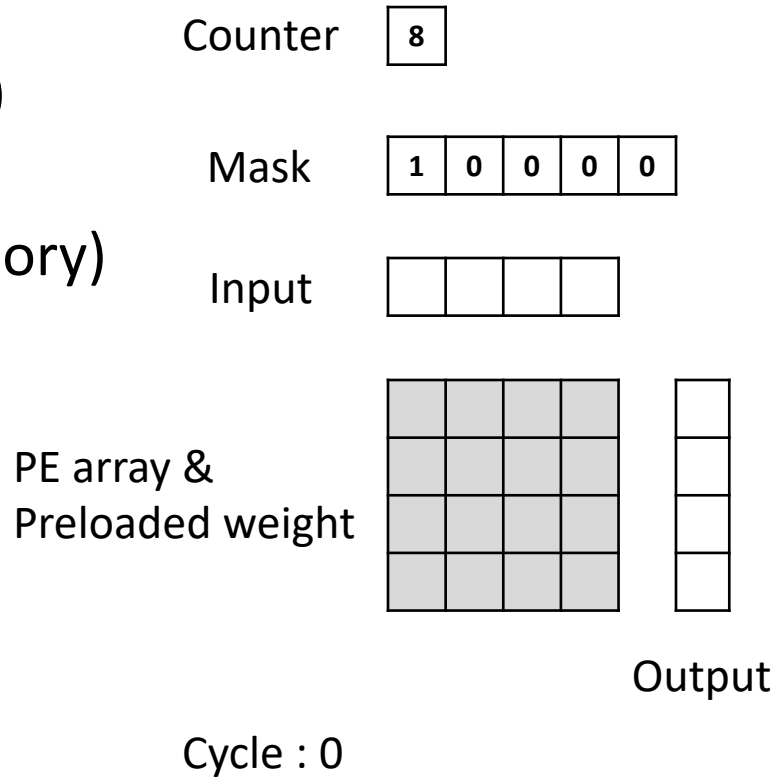


Simultaneous Multi-mode Architecture (SMA)

LSMA execution

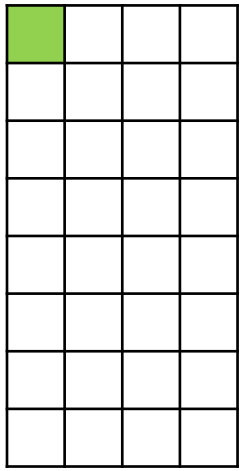
- 5-bit mask for 4x4 array
- Counter (input row number)
- Preload weight
- Prefetch input (Shared memory)
- Execute
- Store output (Register file)

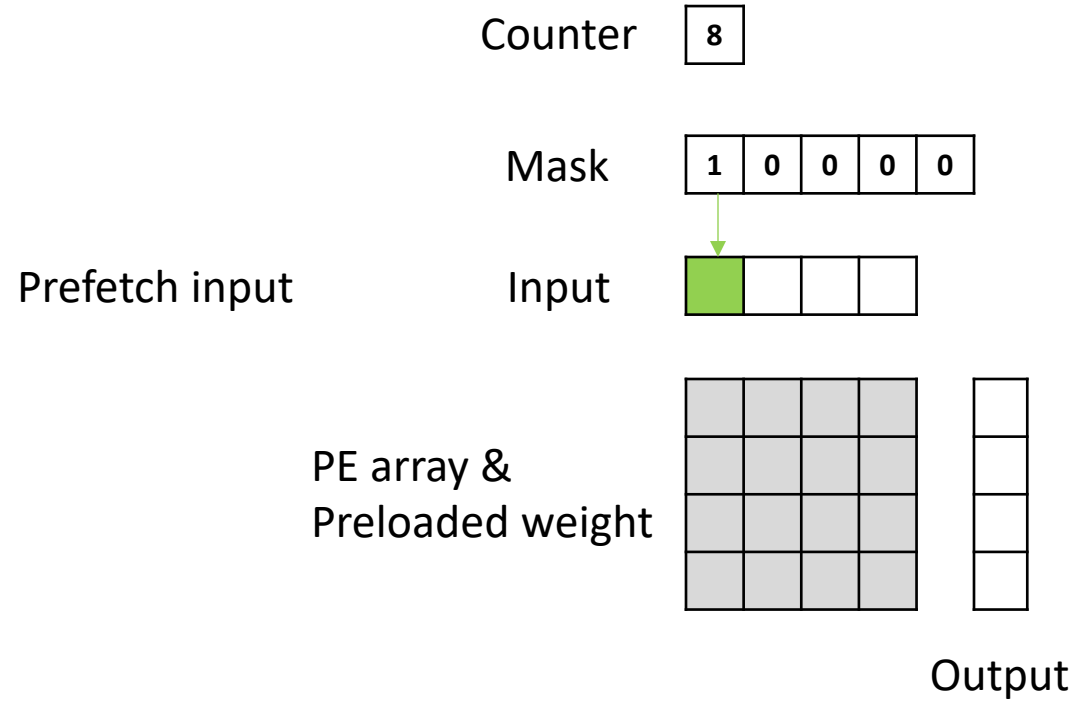




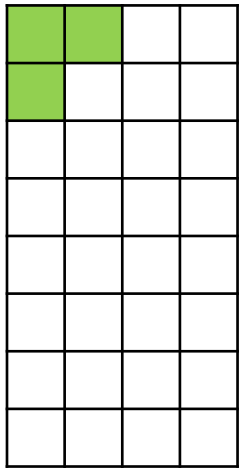
Cycle : 1

Discontinuous

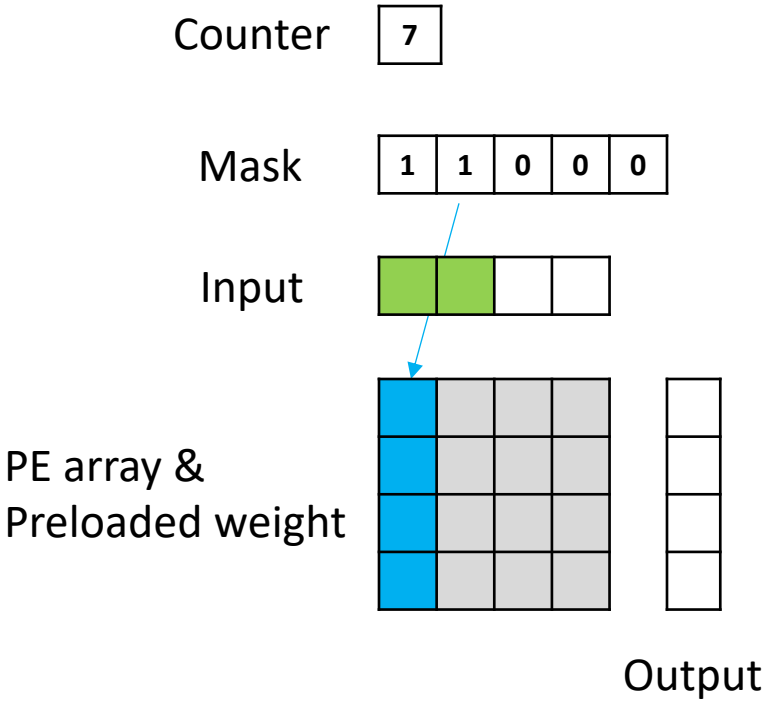




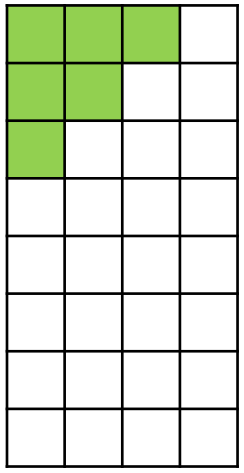
Cycle : 2

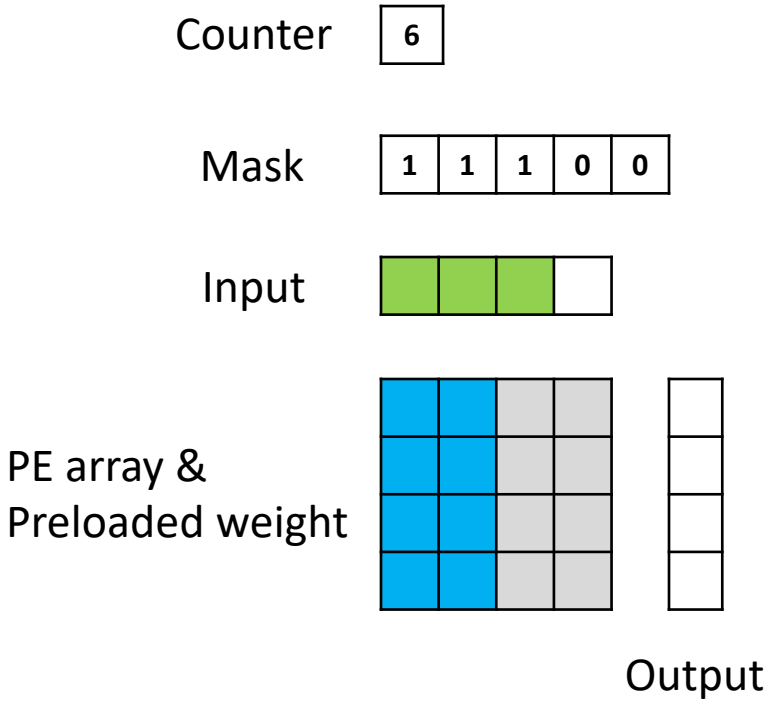


Execute

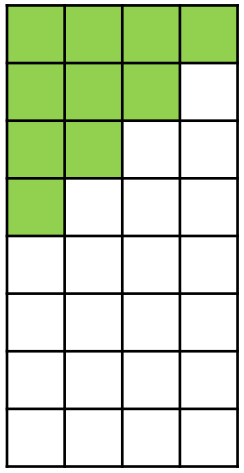


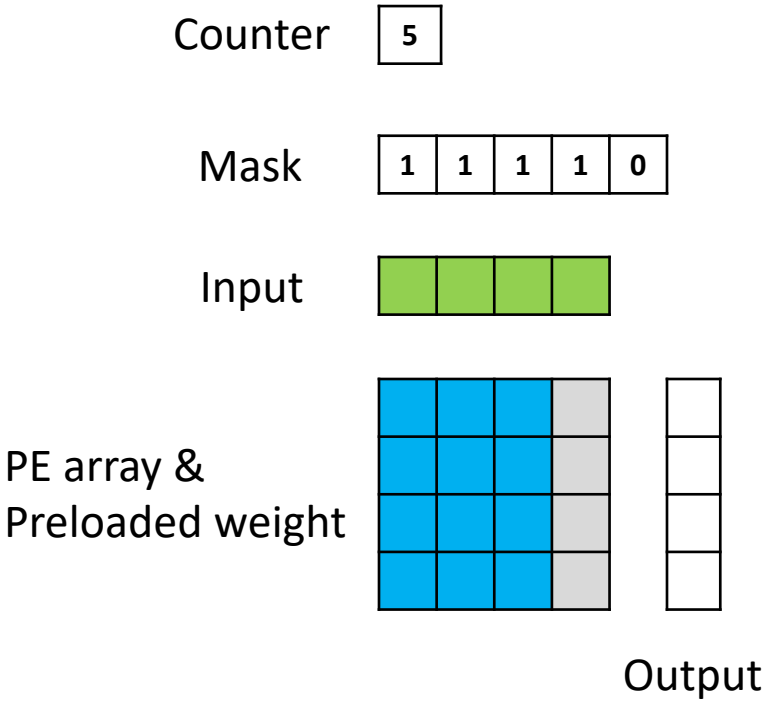
Cycle : 3



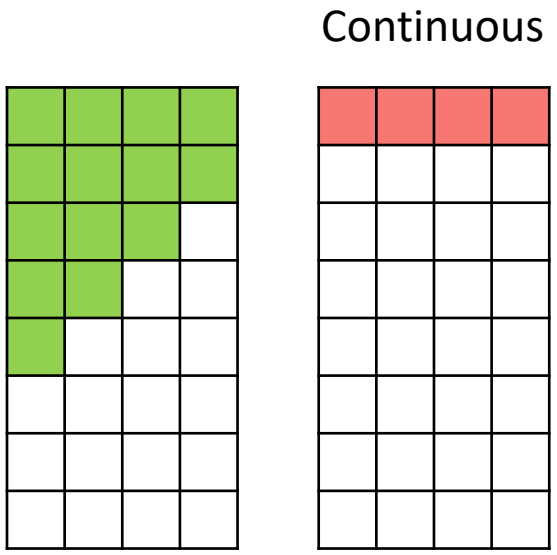

Counter
6
Mask
1 1 1 0 0
Input
PE array &
Preloaded weight
Output
Cycle : 3

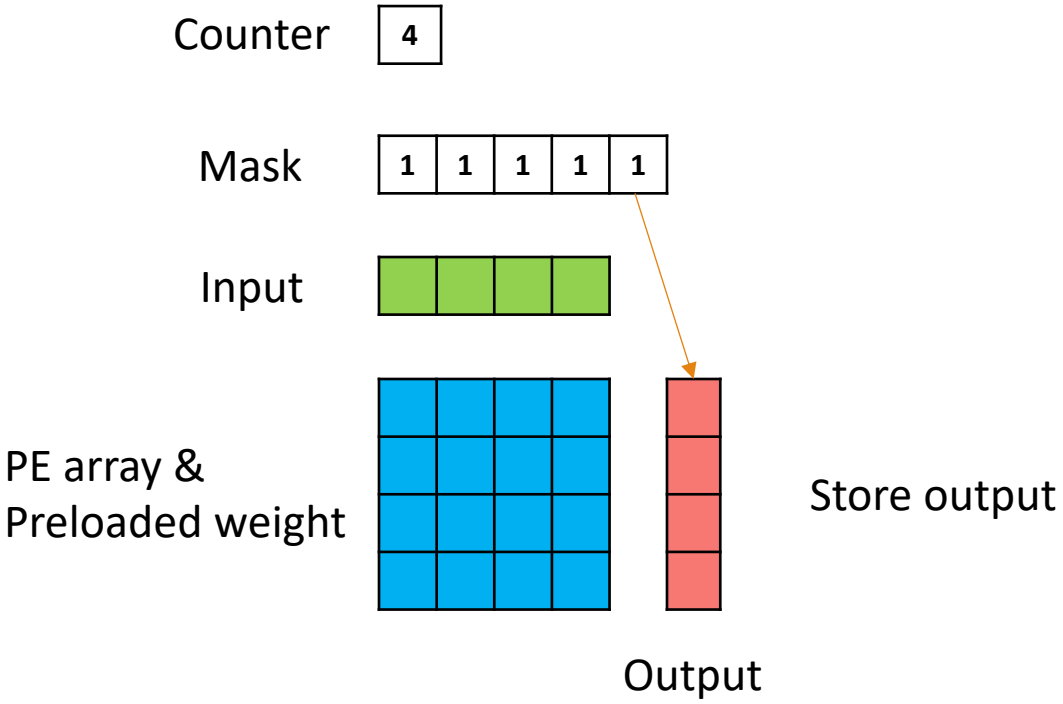
Cycle : 4



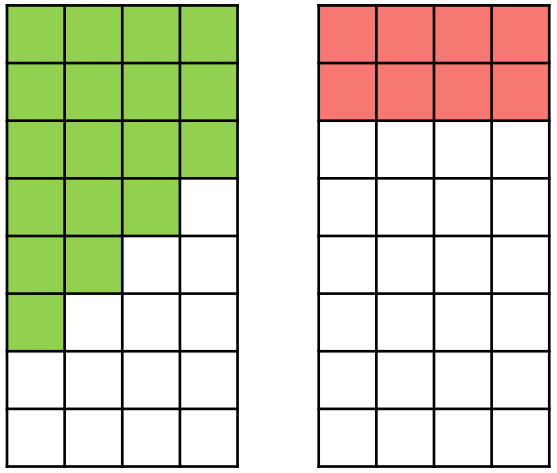

Counter
5
Mask
1 1 1 1 0
Input
PE array &
Preloaded weight
Output
Cycle : 4

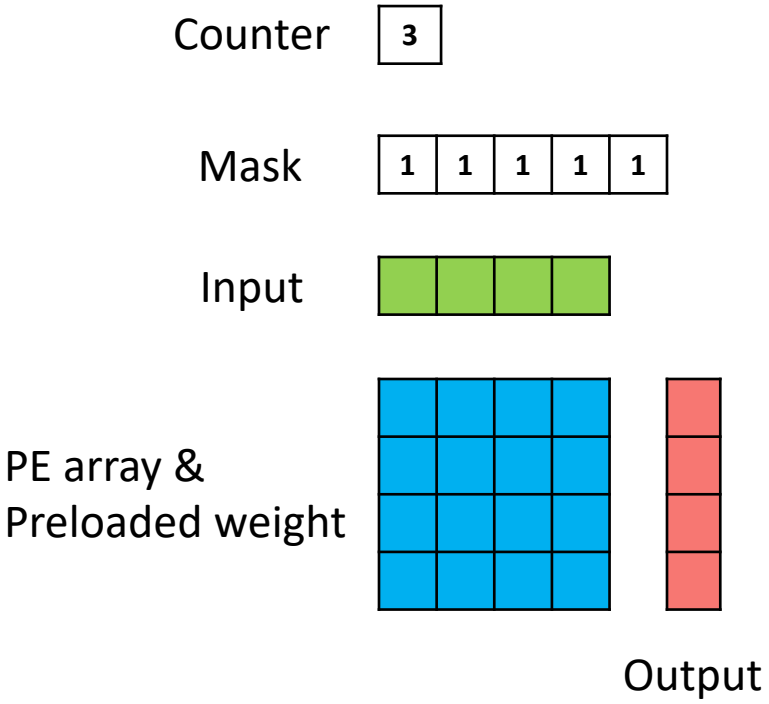
Cycle : 5


Continuous

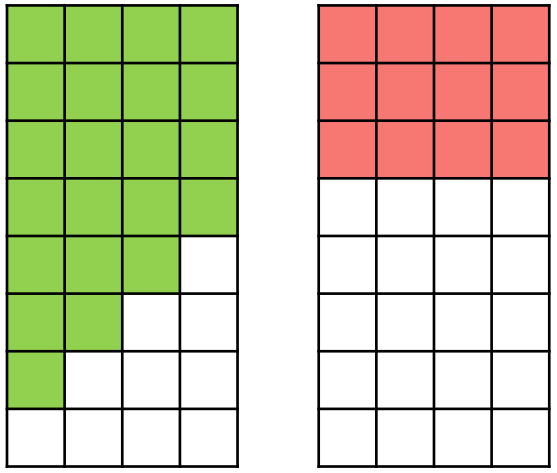

Counter
4
Mask
1 1 1 1 1
Input
PE array &
Preloaded weight
Store output
Output
Cycle : 5

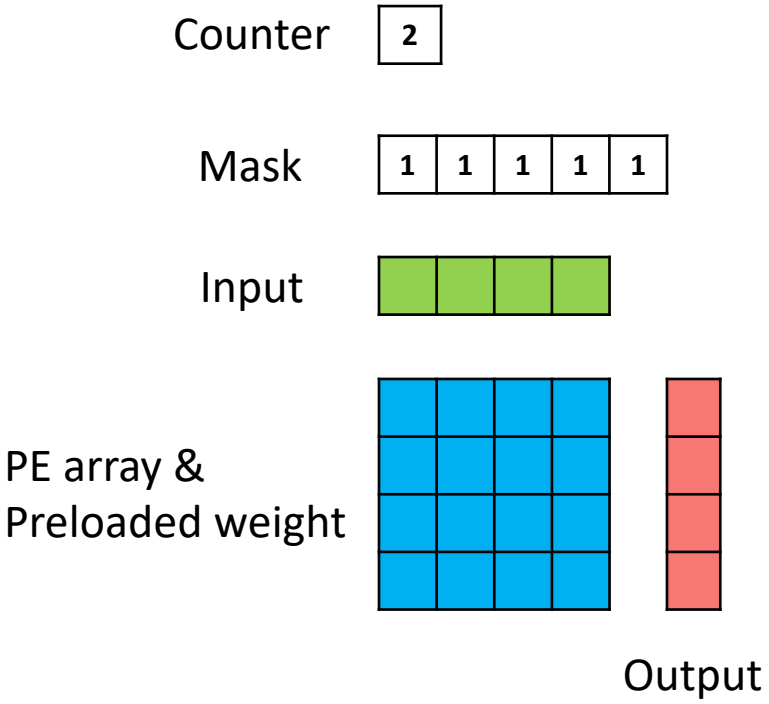
Cycle : 6



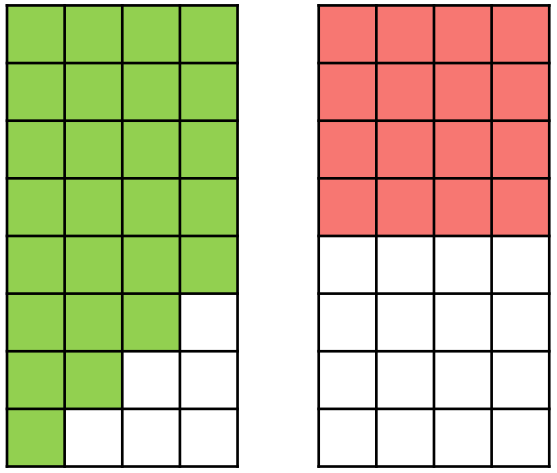

Counter
3
Mask
1 1 1 1 1
Input
PE array &
Preloaded weight
Output
Cycle : 6

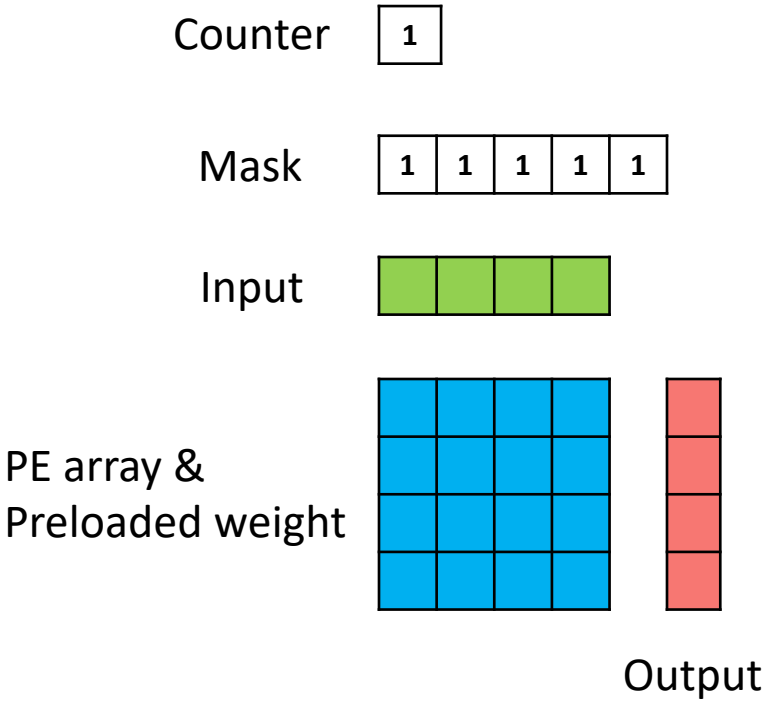
Cycle : 7



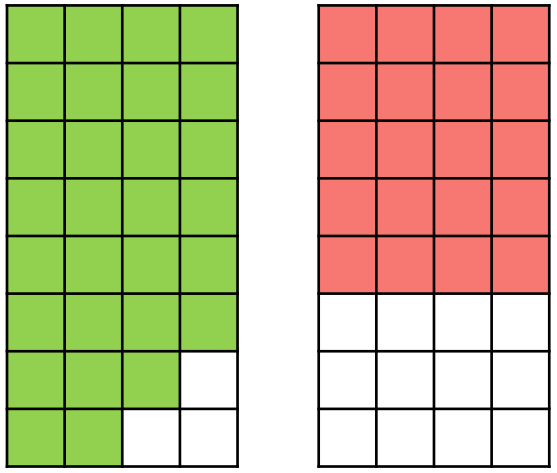

Counter
2
Mask
1 1 1 1 1
Input
PE array &
Preloaded weight
Output
Cycle : 7

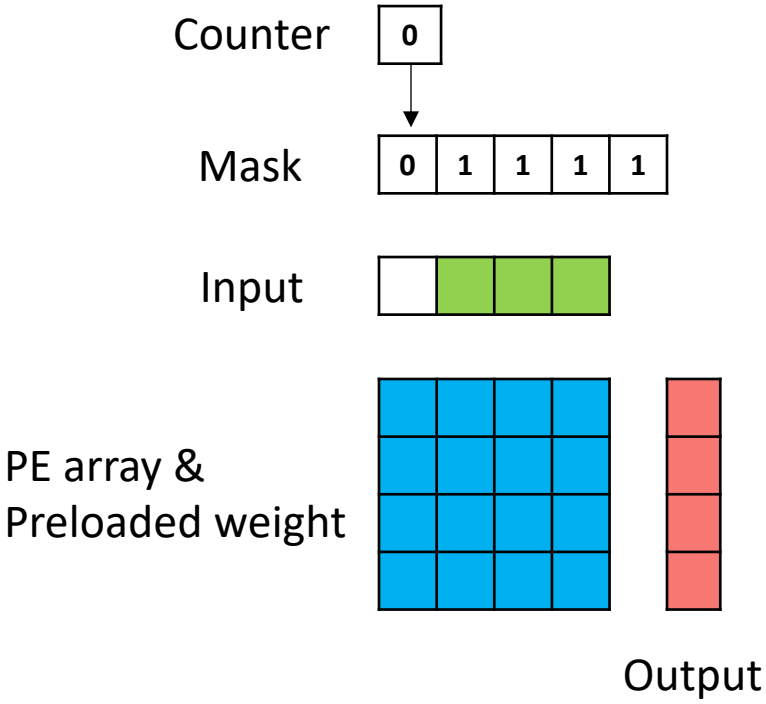
Cycle : 8



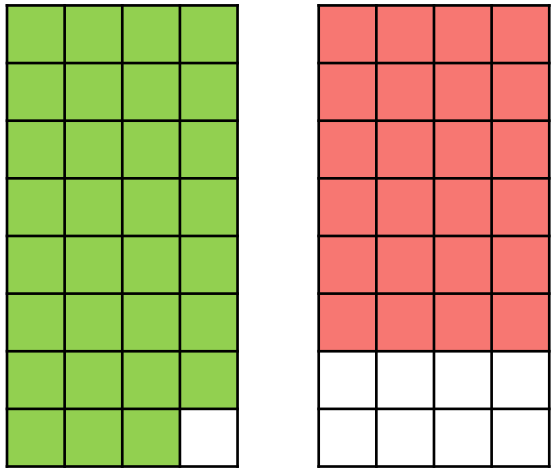

Counter
1
Mask
1 1 1 1 1
Input
PE array &
Preloaded weight
Output
Cycle : 8

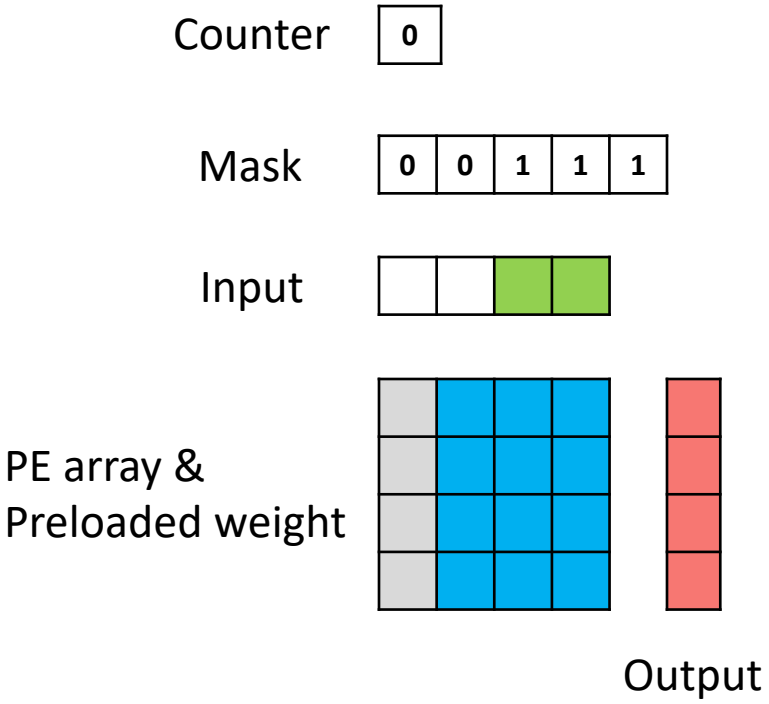
Cycle : 9



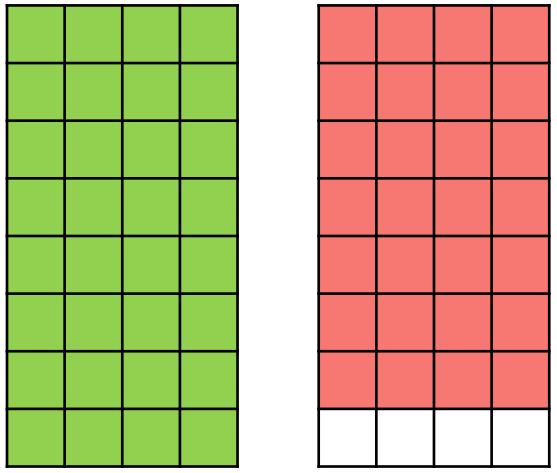

Counter
0
Mask
0
1
1
1
1
Input
PE array &
Preloaded weight
Output
Cycle : 9

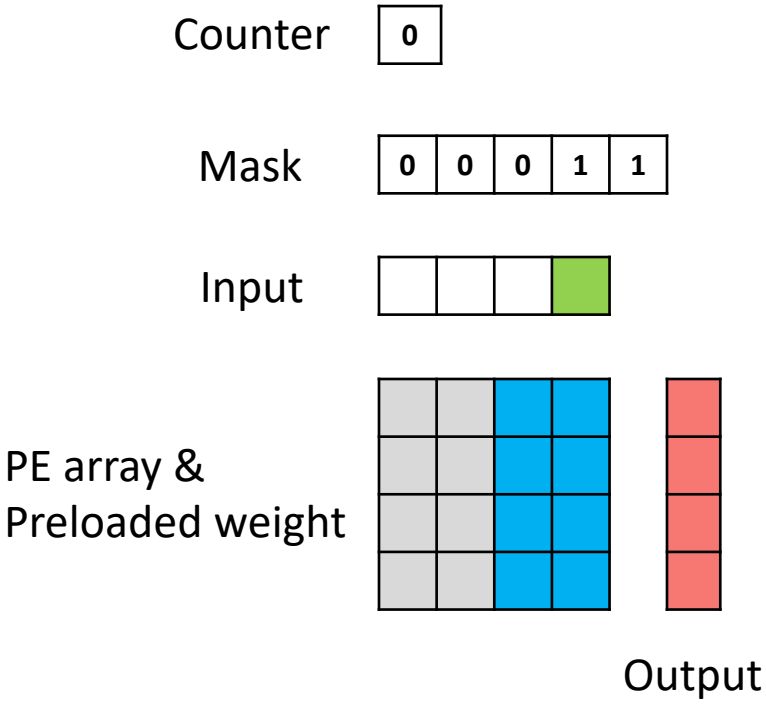
Cycle : 10



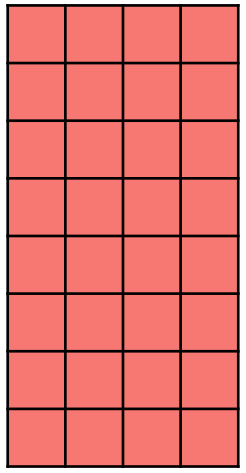

Counter
0
Mask
0 0 1 1 1
Input
PE array &
Preloaded weight
Output
Cycle : 10

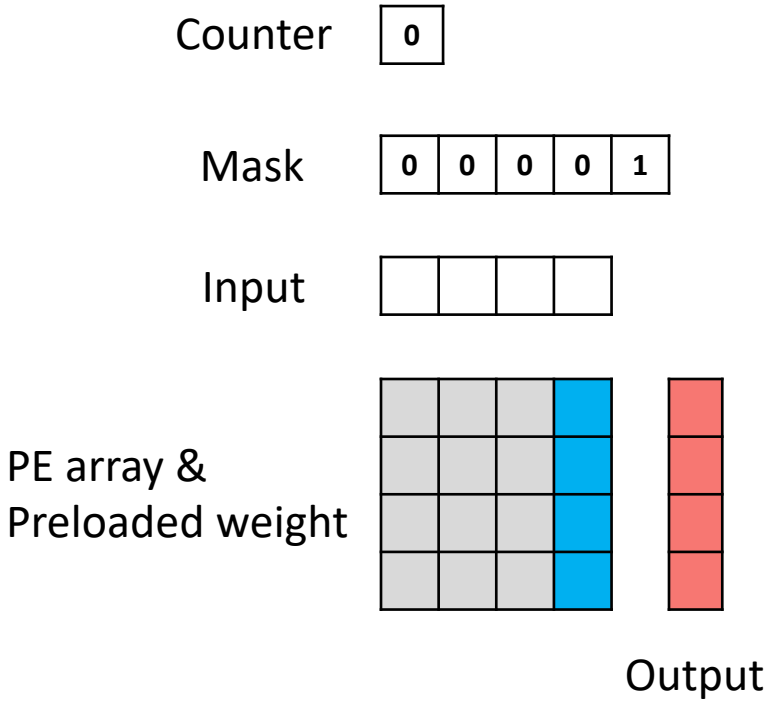
Cycle : 11




Counter
0
Mask
0 0 0 1 1
Input
PE array &
Preloaded weight
Output
Cycle : 11

Cycle : 12




Counter
0
Mask
0 0 0 0 1
Input
PE array &
Preloaded weight
Output
Cycle : 12

Software Design: Tiling GEMM