AUTO-PRUNE: Automated DNN Pruning and Mapping for ReRAM-Based Accelerator

Siling Yang[*] , Weijian Chen[*], Xuechen Zhang[#], Shuibing He[*],
Yanlong Yin[$], Xian-He Sun[+]

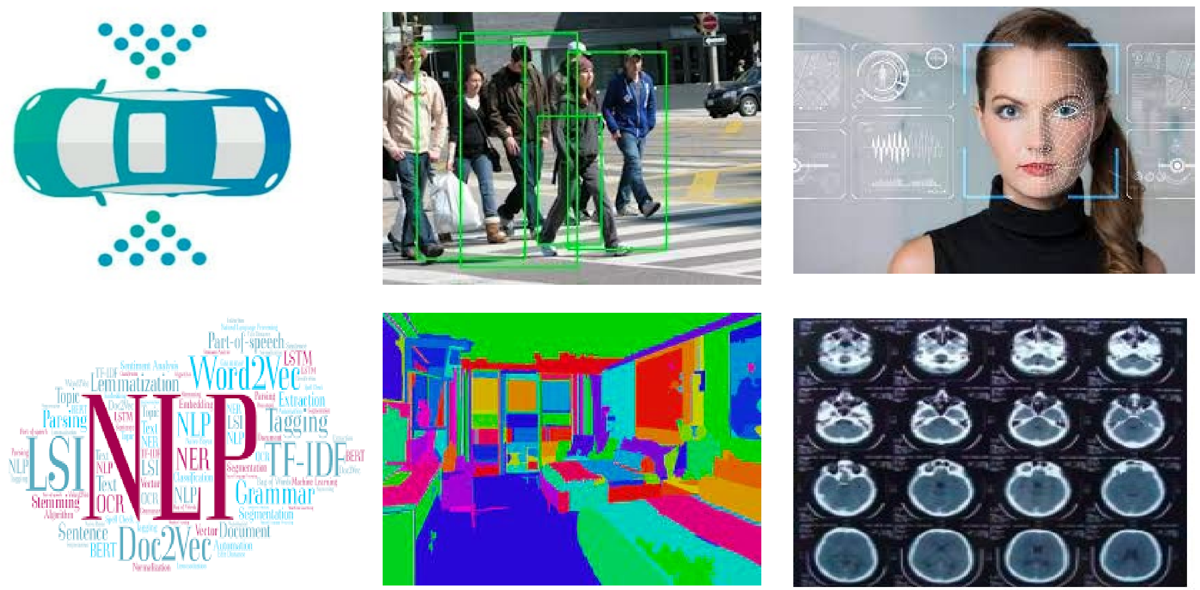
* 浙江大学 Zhejiang University

Washington State University

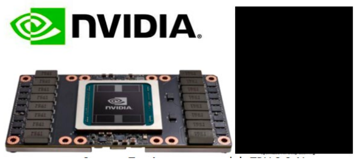
$ ZHEJIANG LAB 之江实验室

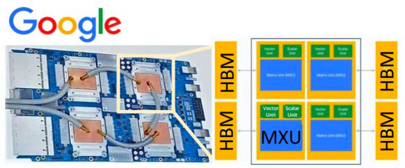
+ ILLINOIS TECH

Accelerating the DNN



Deep neural network (DNN) is popular in various fields.





...

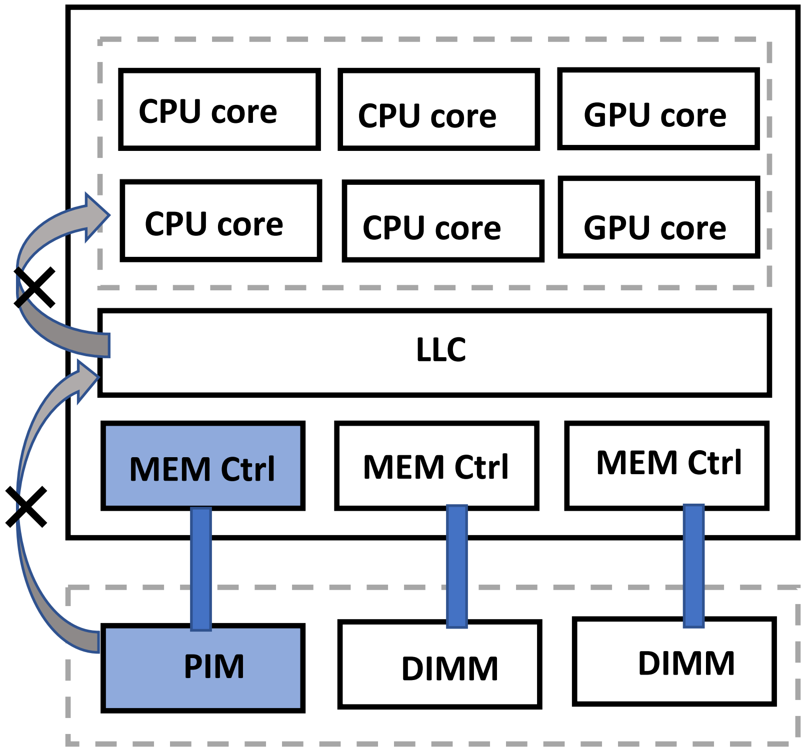
- DNN Hardware accelerator
  - GPU
  - TPU
  - ASIC
  - Novel architectures and emerging devices

Von Neumann Architecture vs. Processing-in-Memory

Energy Wall

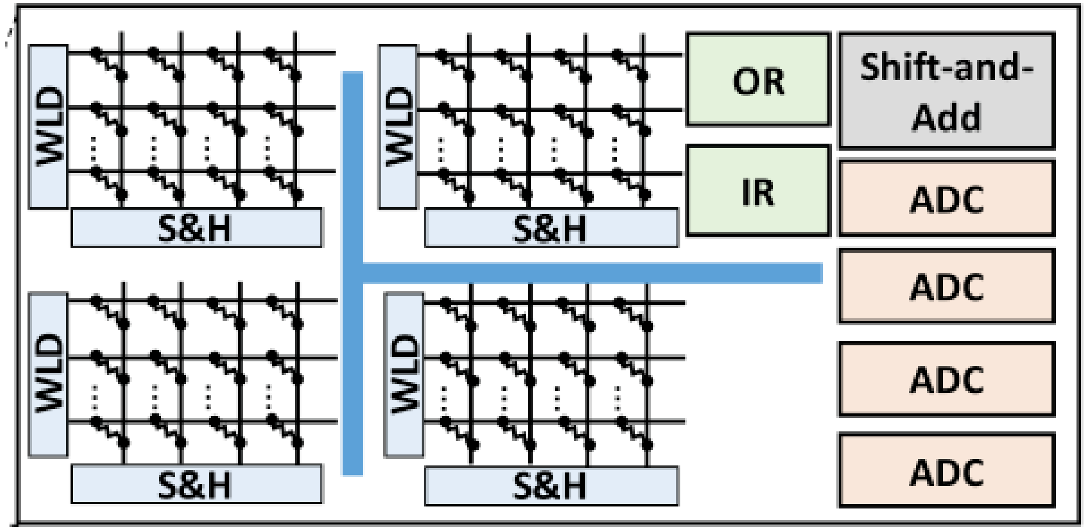
Operation	Energy(pJ)
16b Add	0.05
32b Add	0.1
16b FP Add	0.4
32b FP Add	0.9
32b Mult	3.1
16b FP Mult	1.1
32b FP Mult	3.7
32b SRAM Read(8KB)	5
32b DRAM Read	640

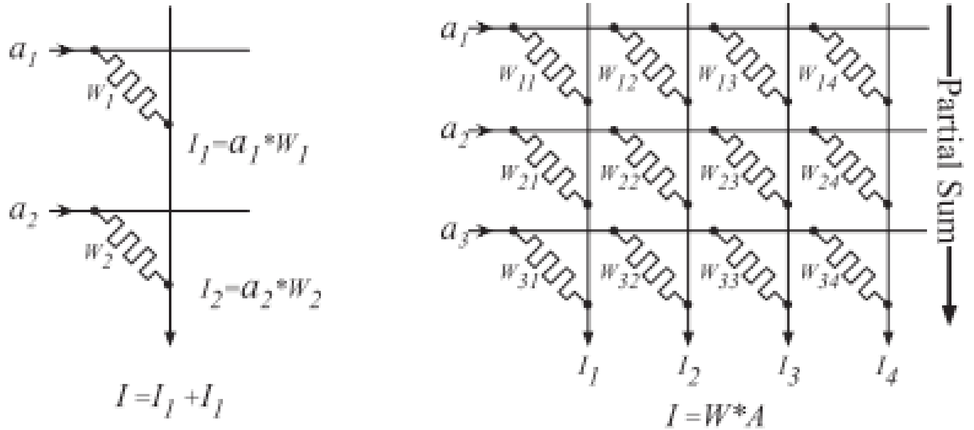
Processing in Memory



PIM and emerging devices can alleviate the energy wall.

ReRAM-based Accelerator

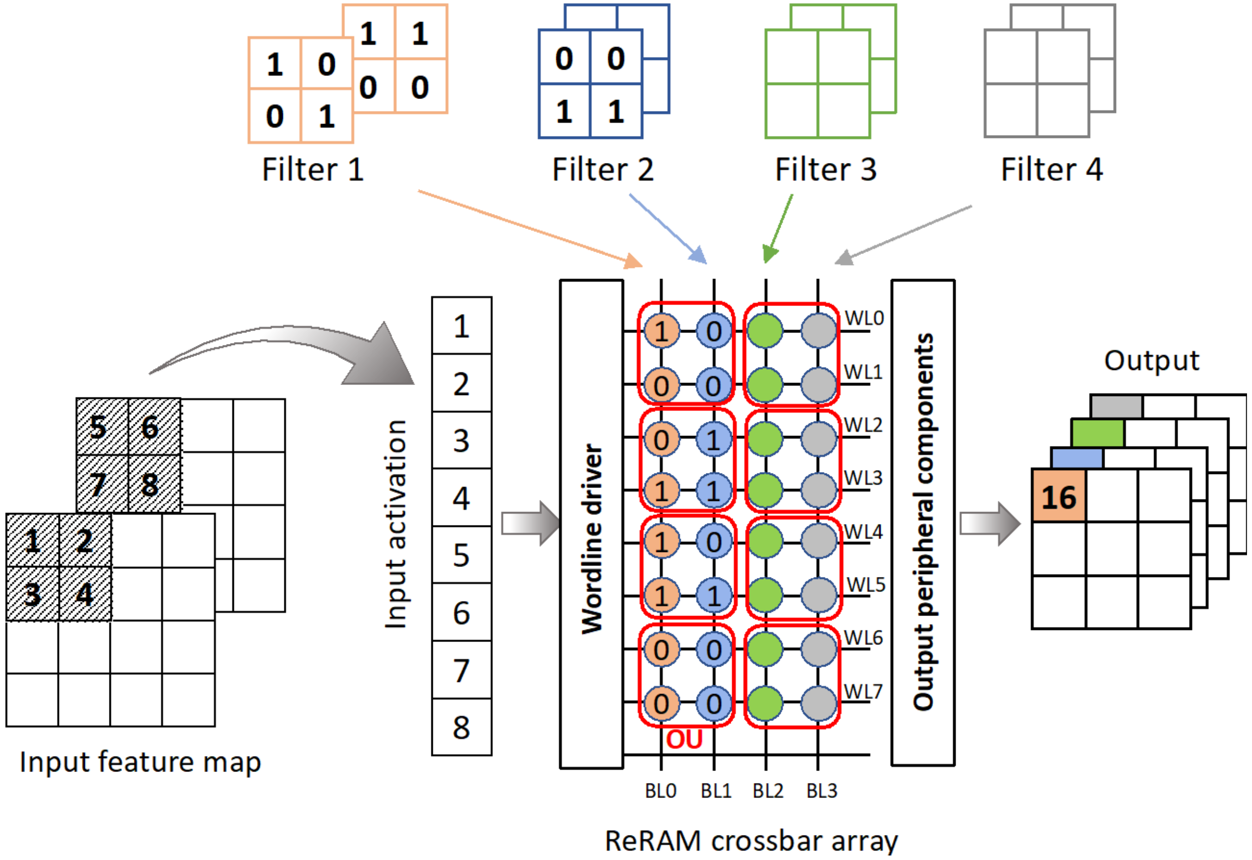




(a) Multiply and Accumulation
(b) Matrix-Vector Multiplication

ReRAM-based DNN accelerator architecture.[SRE-19]

Mapping Filter Weights of DNNs in ReRAM-based Accelerators



➢ **Operation Unit (OU)**

Illustration of mapping filter weights to a crossbar array used in the architecture of ReRAM-based accelerators.

Filter weight matrices of DNN models are sparse.

Related Work & Motivation

Pruning techqiue	Method	Hardware customization	Pattern for pruning	Use OU in data-path
LSR[ASPDAC19]	heuristics	✕	Unimportant weight groups	✕
SRE[ISCA20]	heuristics	✕	All-zero row/column vectors	✓
PIM-Prune[DAC20]	heuristics	✕	Unimportant rows and columns	✕
Pattern pruning[arxiv20]	heuristics	✕	Patterns	✓

1. They use heuristics to prune the weights, leading to **suboptimal pruning policies**.
2. They mostly focus on improving compression ratio, thus **may not meet accuracy constraints**.
3. They **ignore direct feedback of hardware**, e.g., the number of occupied crossbars or energy consumption.

Objectives of Our Work

Make a global optimal pruning policy

+

Make a pruning and mapping policy tailored for different hardware

+

Avoid the dislocation problem

⬇

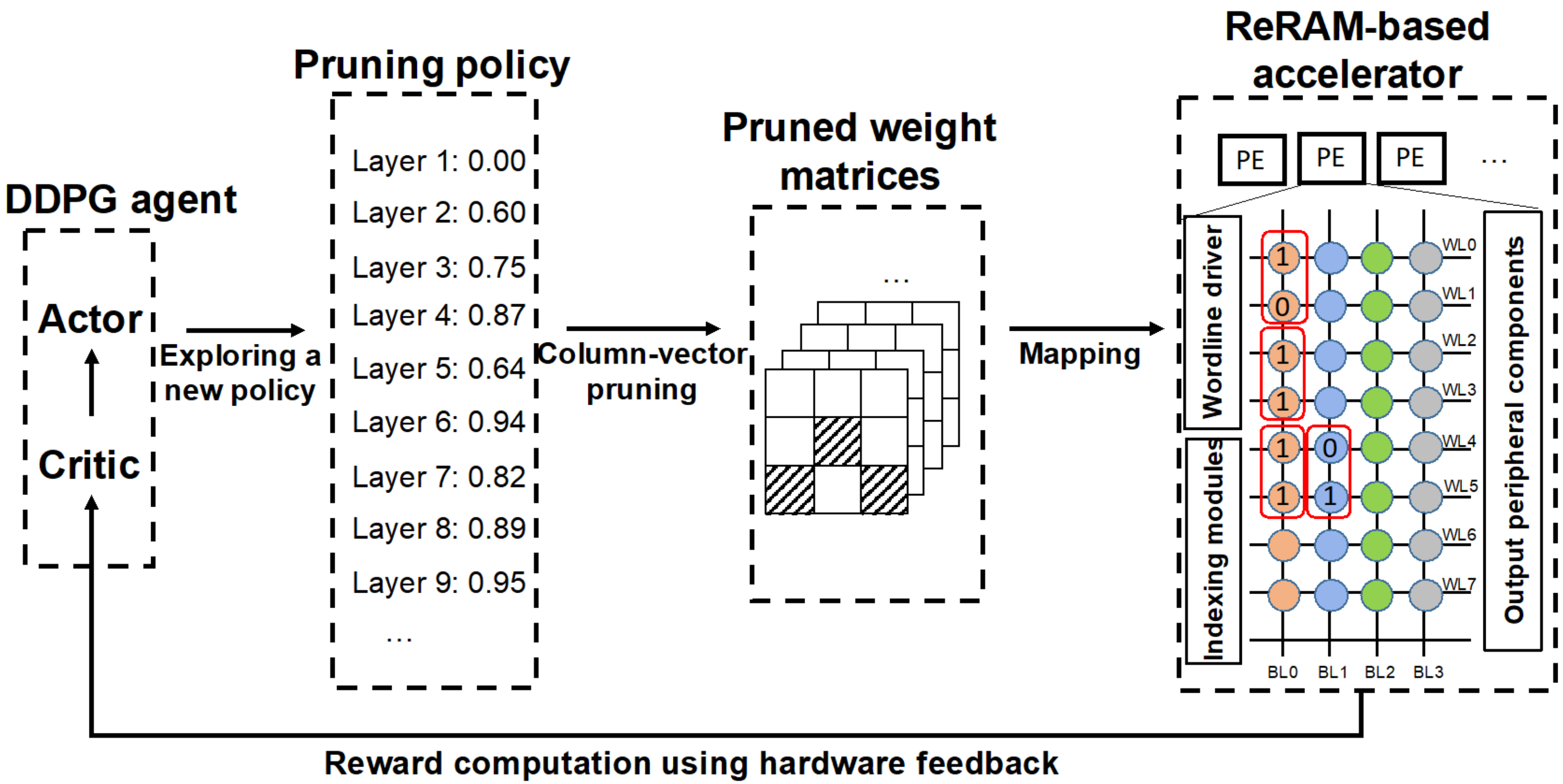
AUTO-PRUNE

Design of AUTO-PRUNE

Main Design

- ➢DDPG Algorithm for ReRAM-based Accelerator
- ➢Column-Vector Based Pruning and OU Formation
- ➢Data-Path Design

Overview of AUTO-PRUNE



1. DDPG Algorithm for ReRAM-based Accelerator

- State Space: identify a layer with its characteristics

$$(k, t, inc, outc, ks, h, w, s, xb[k], xb_{saved}[k], xb_{rest}[k], a_{k-1})$$

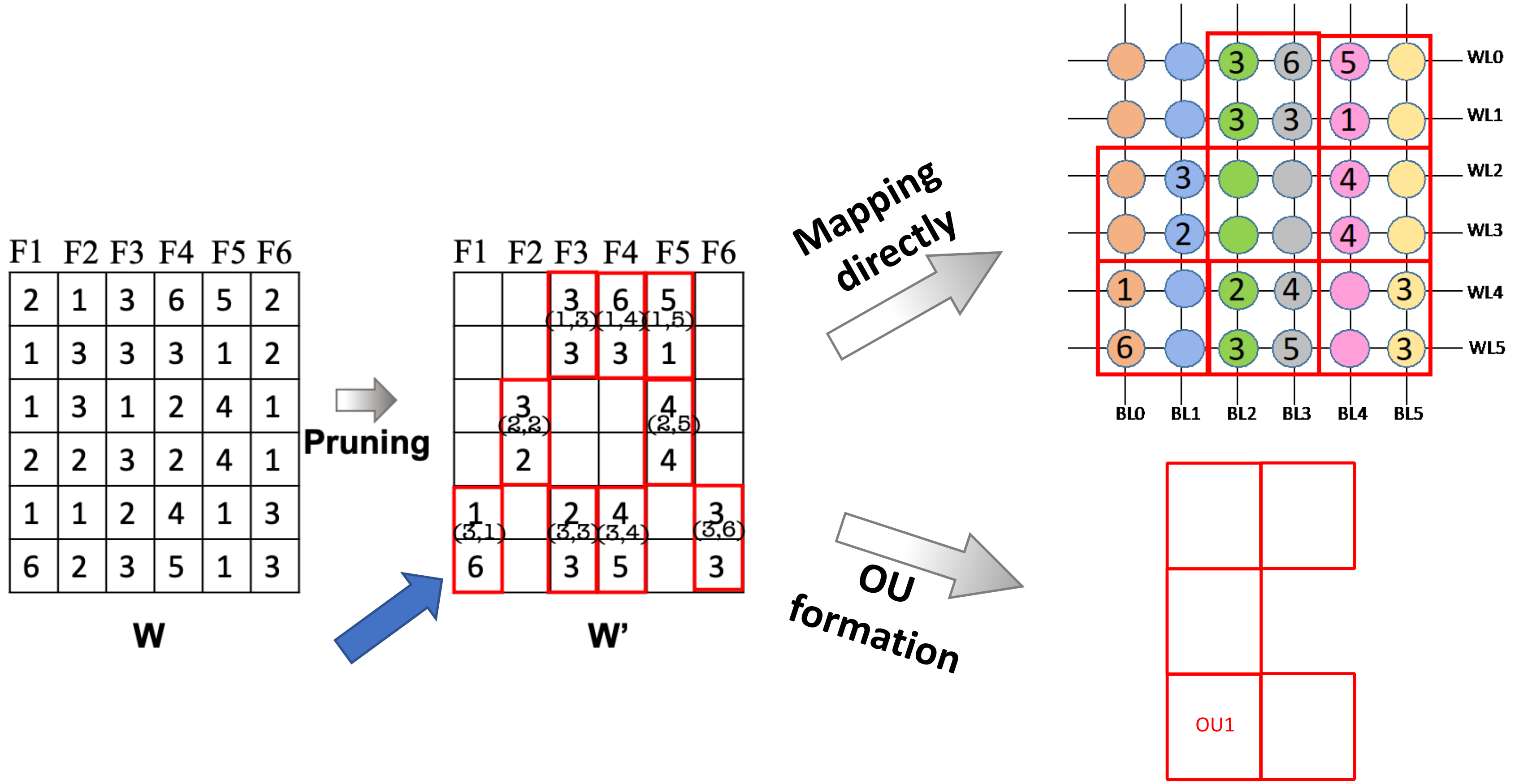
- Action Space: pruning rate for a specified layer
  - $a_k \epsilon (0, 1]$

- Reward Function: related with compression rate and accuracy

$$Reward = (1 - \frac{1}{rate_{compression}^{xb}})^{\alpha} \times \mathrm{acc}_{reram}$$

Symbol	Meaning
k	layer index
t	layer type: CONV:1 ; FC: 0
inc	number of channels in the input feature map
$outc$	number of channels produced by the convolution
ks	number of elements of a convolving kernel
h	height of the input feature maps
w	width of the input feature maps
s	stride of the convolution
$xb[k]$	number of crossbars required for mapping layer k
$xb_{saved}[k]$	accumulated number of the crossbar saved from the first layer to layer $k-1$
$xb_{rest}[k]$	number of crossbars required from layer $k+1$ to the last layer
$size_{xb}$	length of the crossbar size
acc_{reram}	accuracy reported by the ReRAM-based accelerator simulator
a_{k-1}	action from the last time step

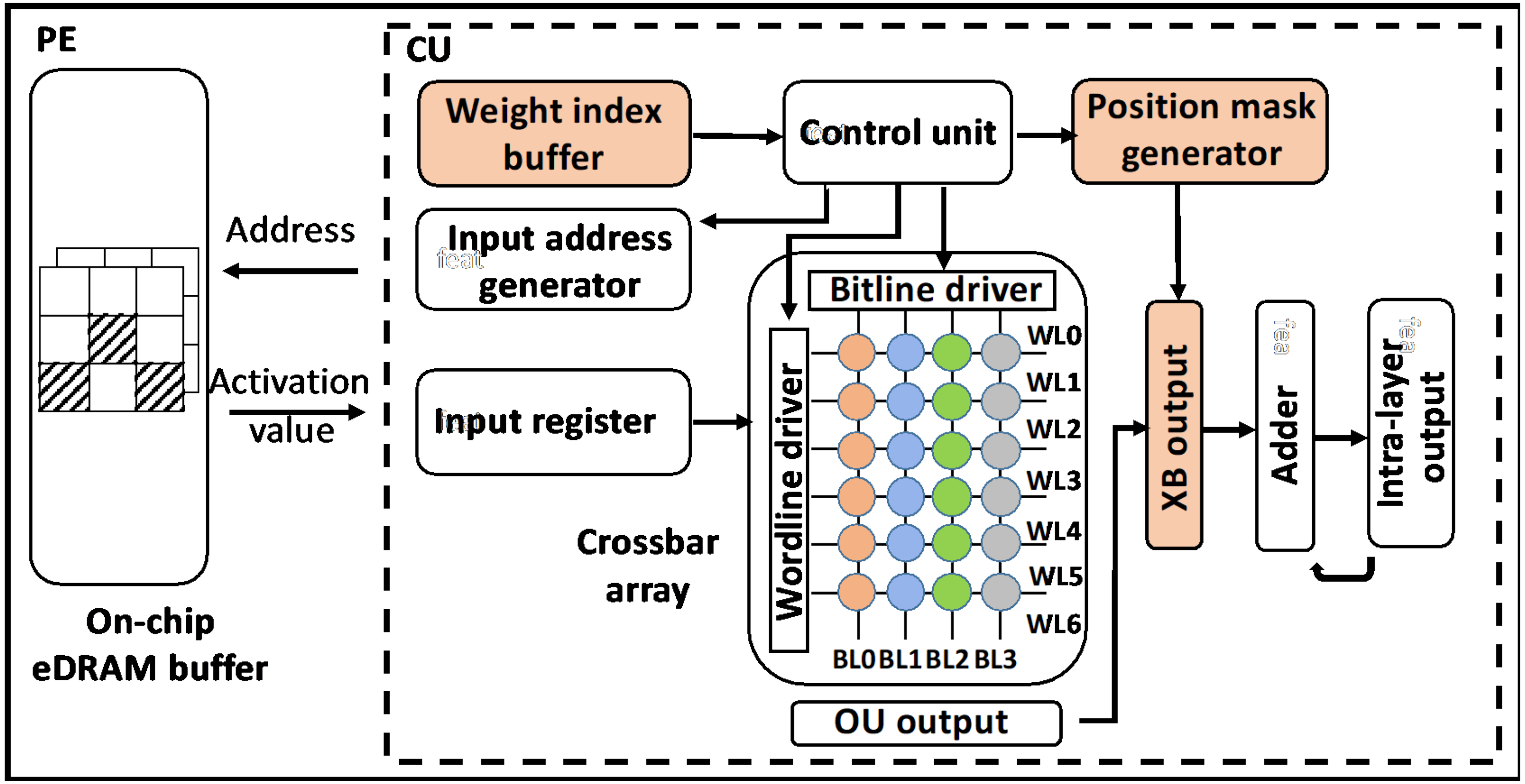
2. Column-Vector Based Pruning and OU Formation



- OU List

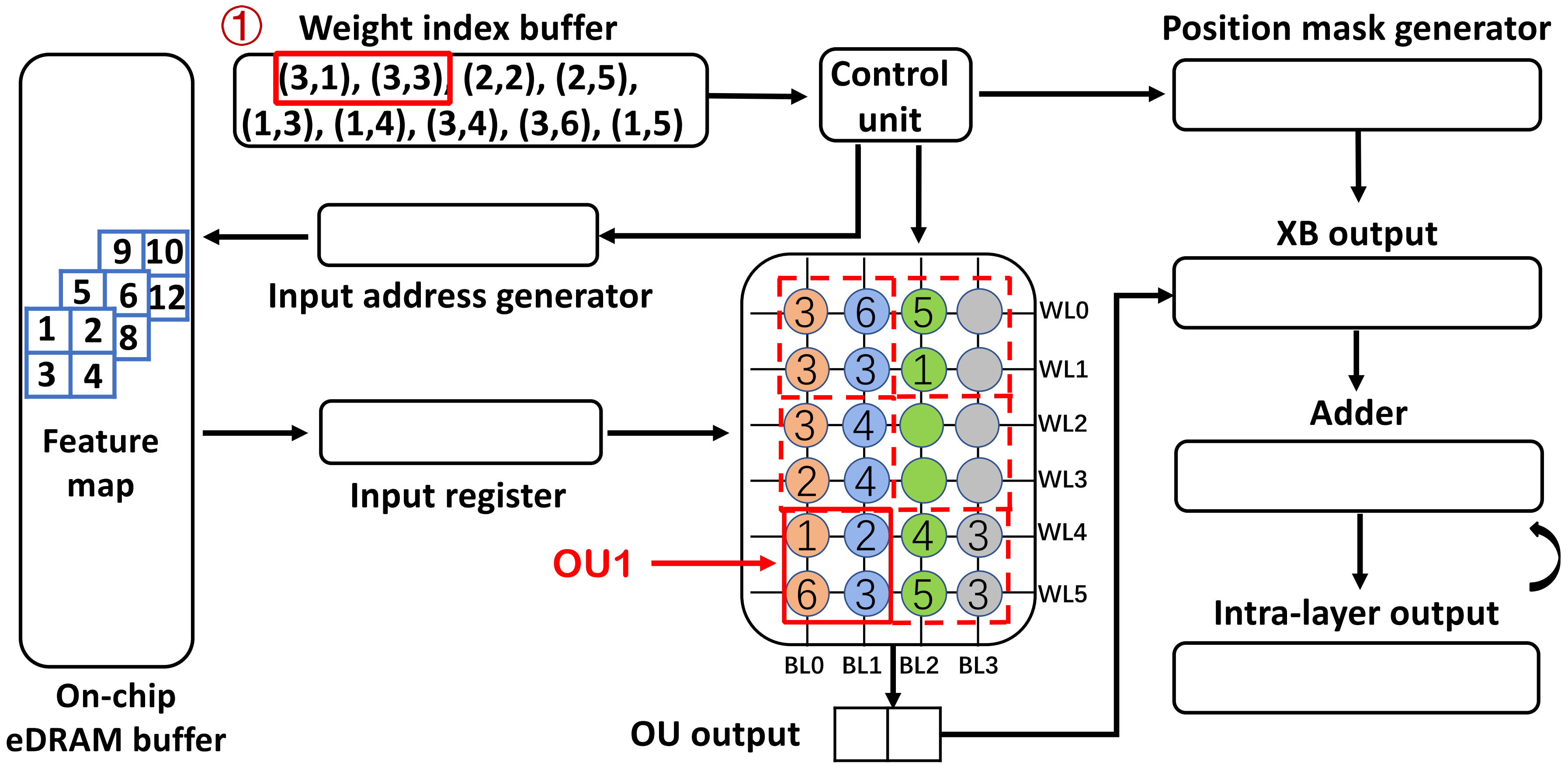
{(3,1), (3,3), (2,2), (2,5), (1,3), (1,4), (3,4), (3,6), (1,5)}

3. Data-Path Design

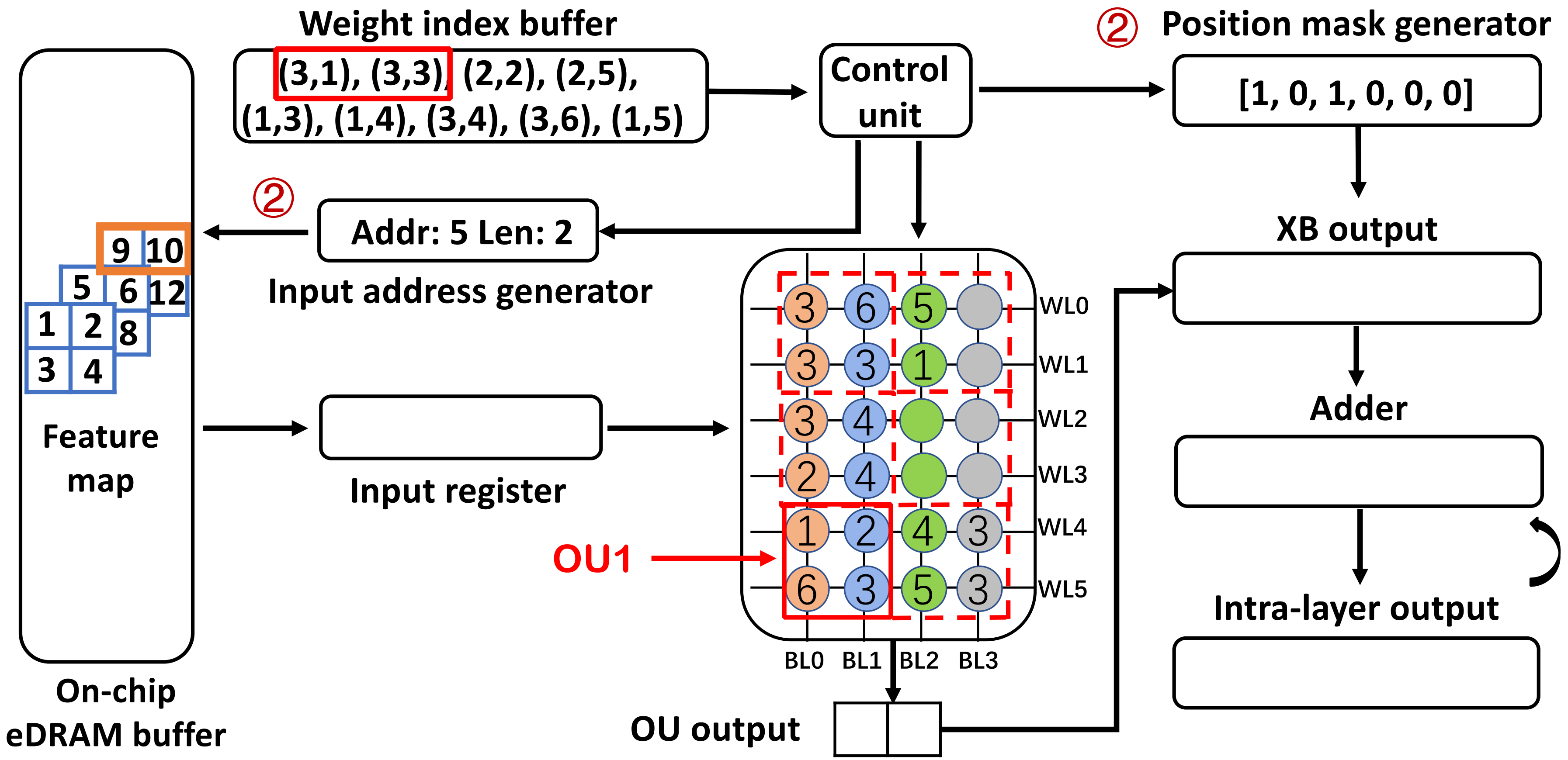


An overview of the data-path for Auto-prune.

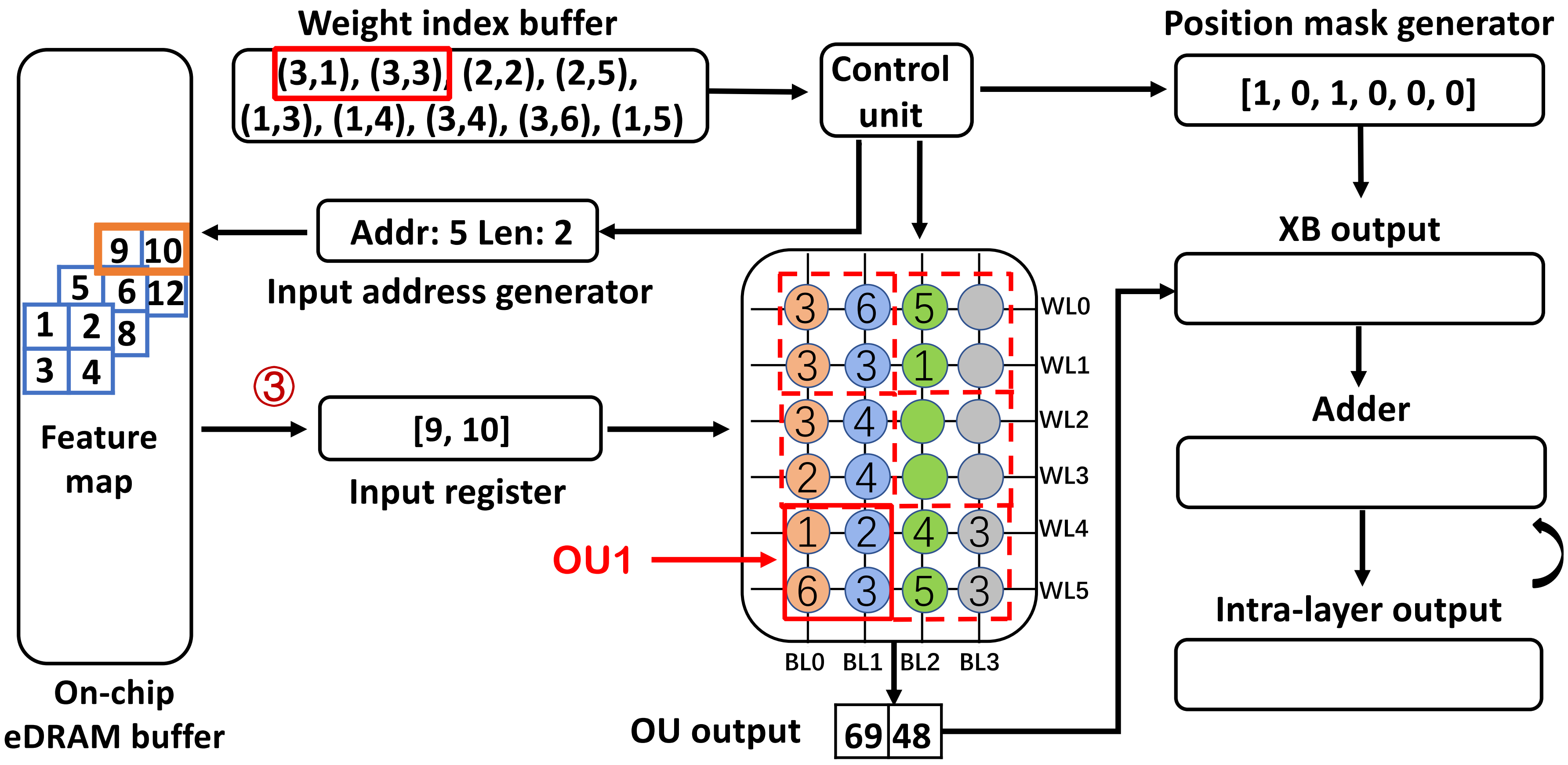
An example: conv operation in OU1



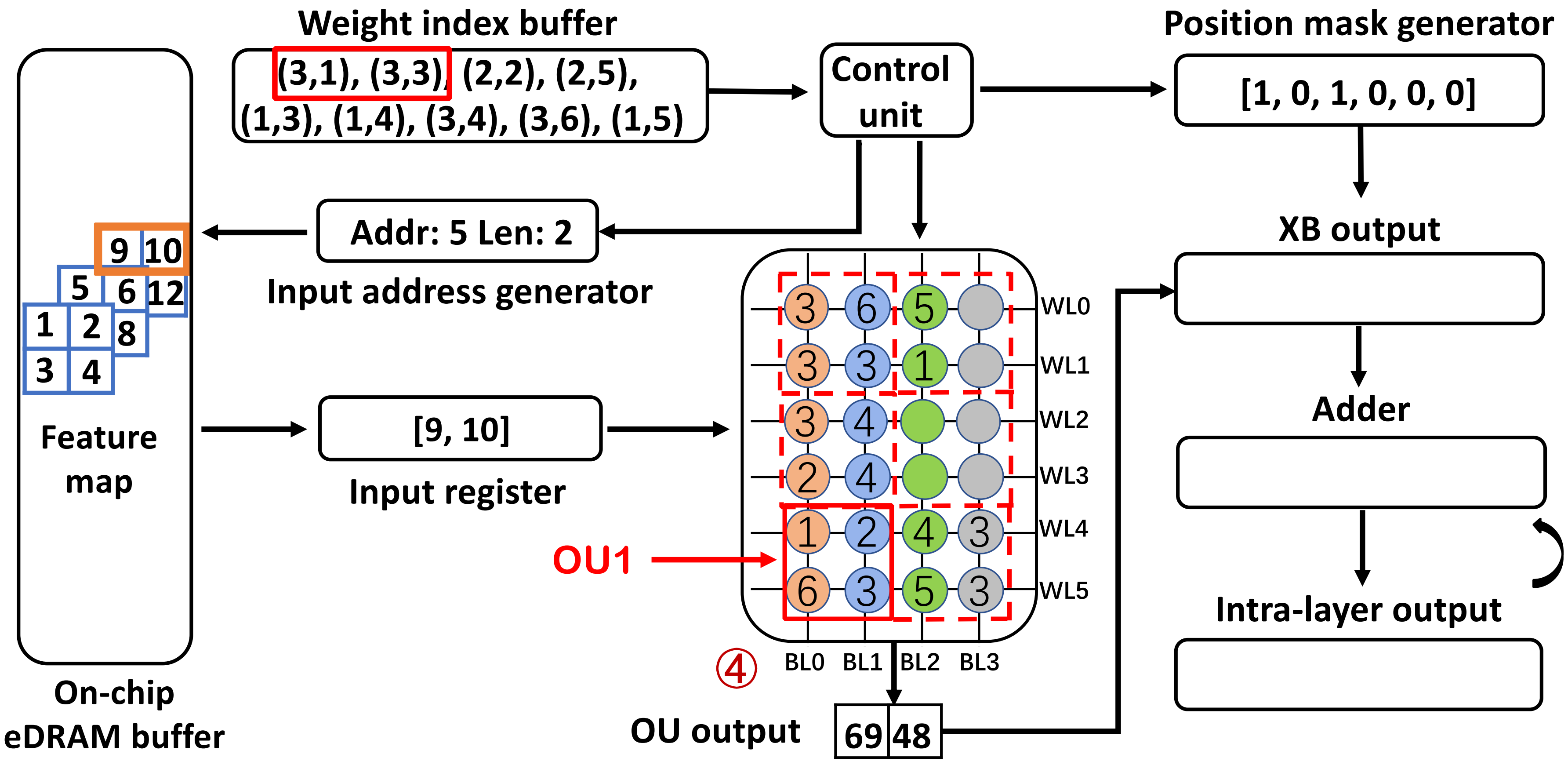
An example: conv operation in OU1



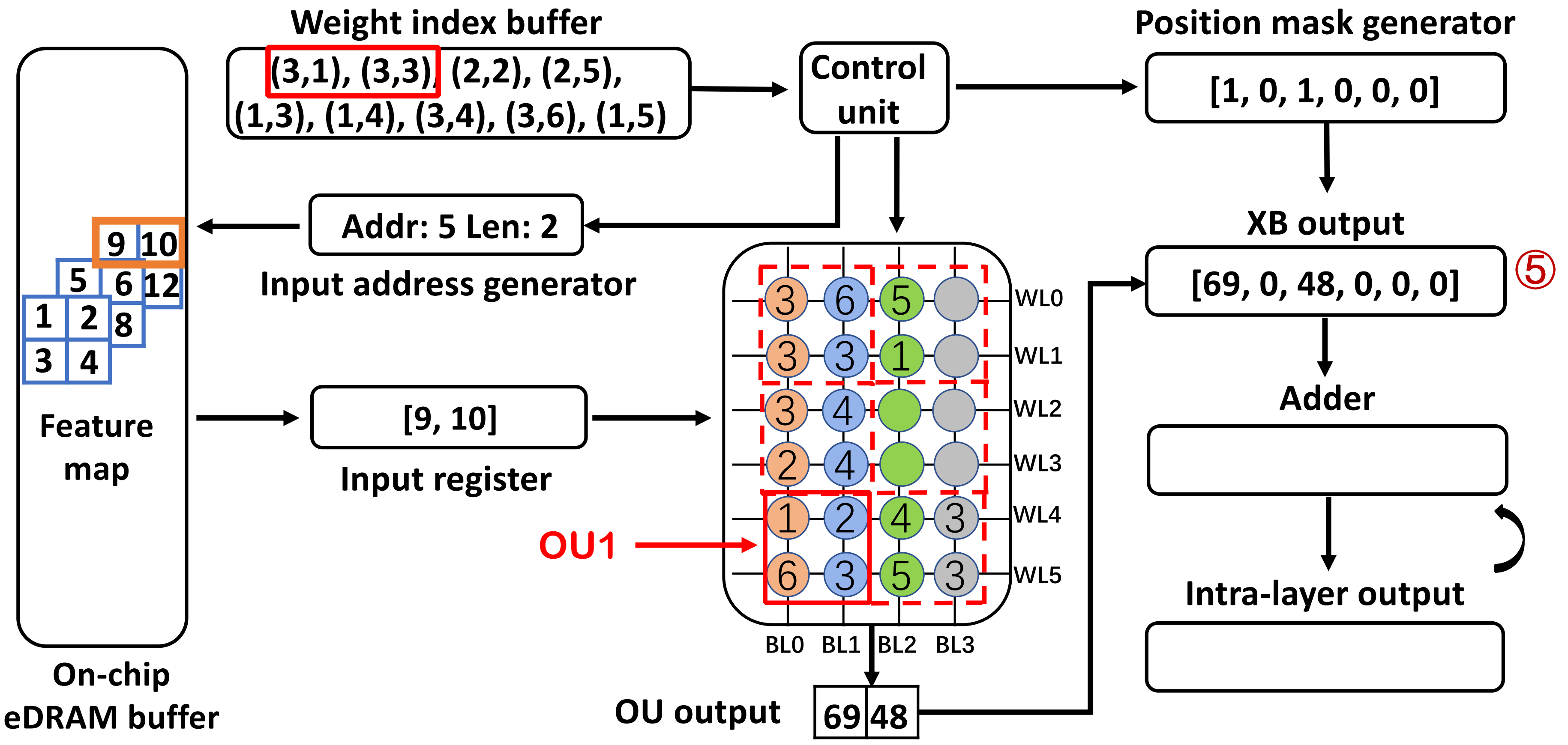
An example: conv operation in OU1



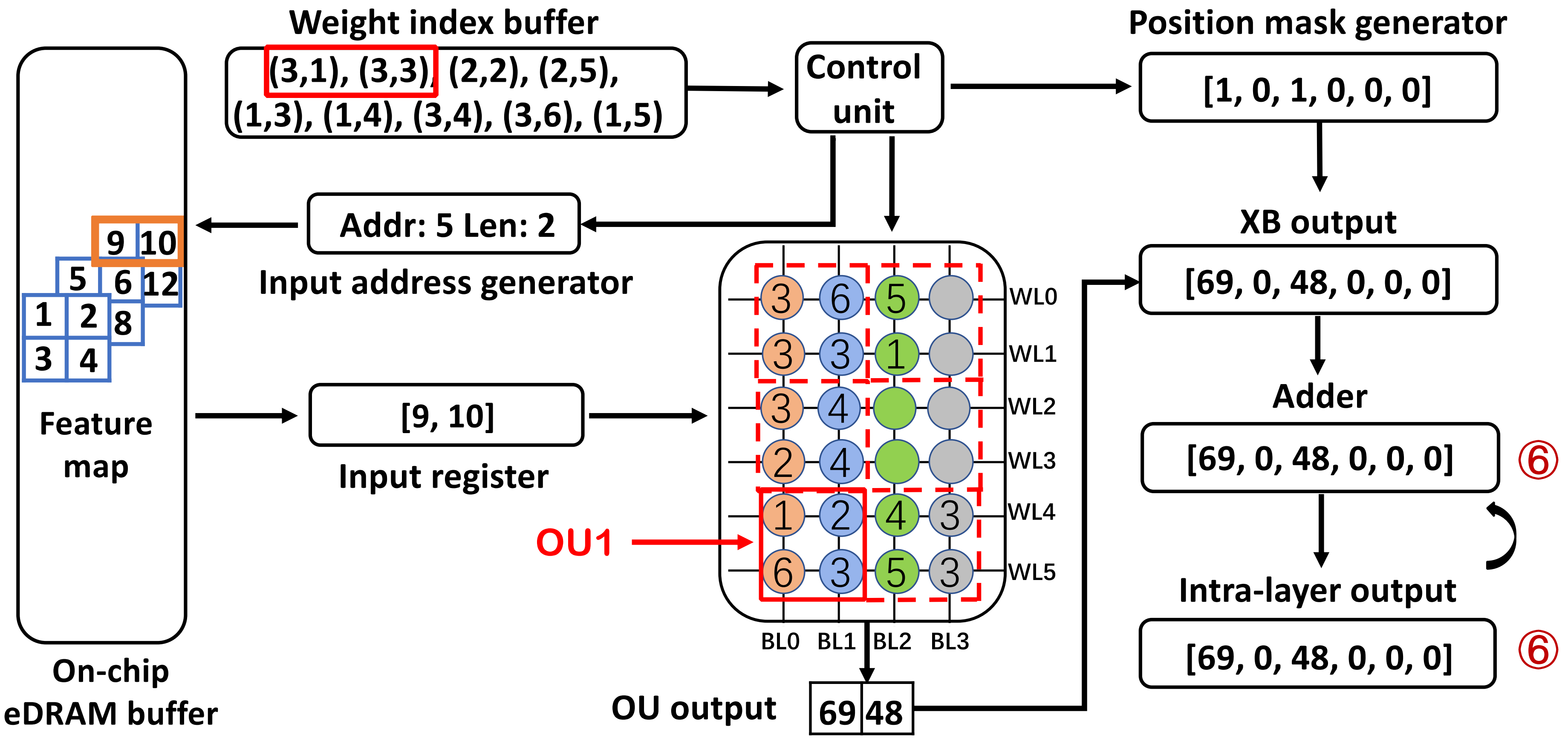
An example: conv operation in OU1



An example: conv operation in OU1



An example: conv operation in OU1



Experimental Setting

- simulator: MNSIM-2.0
  - Crossbar size： 128 x 128
  - bit-per-cell: 1 bits
  - OU size: 32 x 32

- Counterparts
  - Naïve
  - PIM-Prune[DAC-20]
  - Pattern-Prune[Arxiv-20]

- Workloads and datasets
  - NN: AlexNet, VGG16, Plain20
  - Dataset: CIFAR10, MNIST

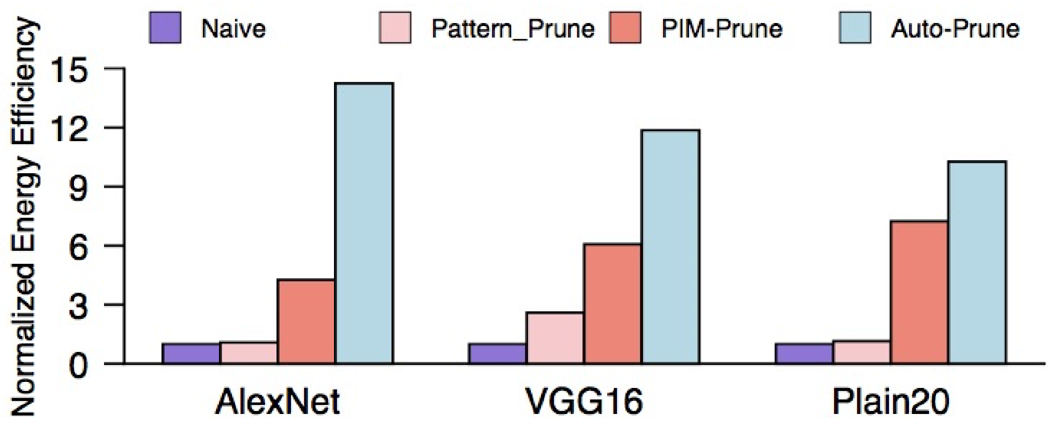
- Metrics
  - Compression rate
  - Energy & area efficiency
- Discussion
  - Sensitivity & Overhead

Compression rate

Network	Method	CR on XBs	Acc5	Acc Drop
AlexNet	Naïve	1	99.36%	-
	PIM-Prune	4.3	98.81%	0.55%
	Pattern-Prune	1.1	96.48%	2.88%
	Auto-Prune	14.3	99.10%	0.26%
VGG16	Naïve	1	99.29%	-
	PIM-Prune	6.1	98.62%	0.67%
	Pattern-Prune	2.6	98.43%	0.86%
	Auto-Prune	11.9	98.62%	0.67%
Plain20	Naïve	1	98.14%	-
	PIM-Prune	7.3	98.19%	-0.05%
	Pattern-Prune	1.2	98.24%	-0.10%
	Auto-Prune	10.3	98.29%	-0.15%

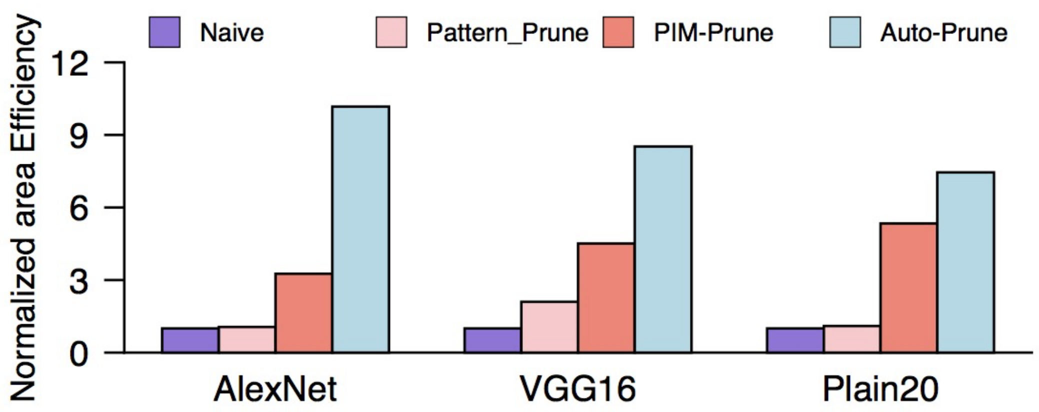
the same or higher accuracy, compression rate up to:
3.3X PIM-Prune
13X Pattern-Prune

Energy efficiency & area efficiency



(a) CIFAR10.

the result of energy efficiency on CIFAR10



(a) CIFAR10.

the result of area efficiency on CIFAR10

12.2 Pattern-Prune
2.3 PIM-Prune

3.1X Pattern-Prune
9.6X PIM-Prune

Sensitivity study

- granularity of column-vector
- ……

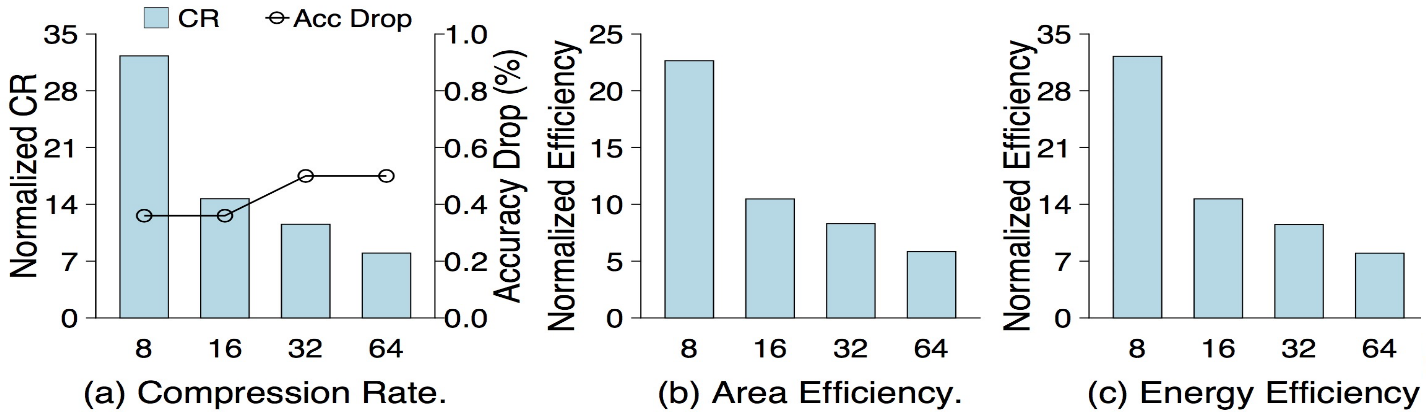


Figure 10: Compression rate, area efficiency and energy efficiency for AlexNet with various granularities of column-vectors.

The smaller granularity of column-vector, the higher compression rate.

Index overhead

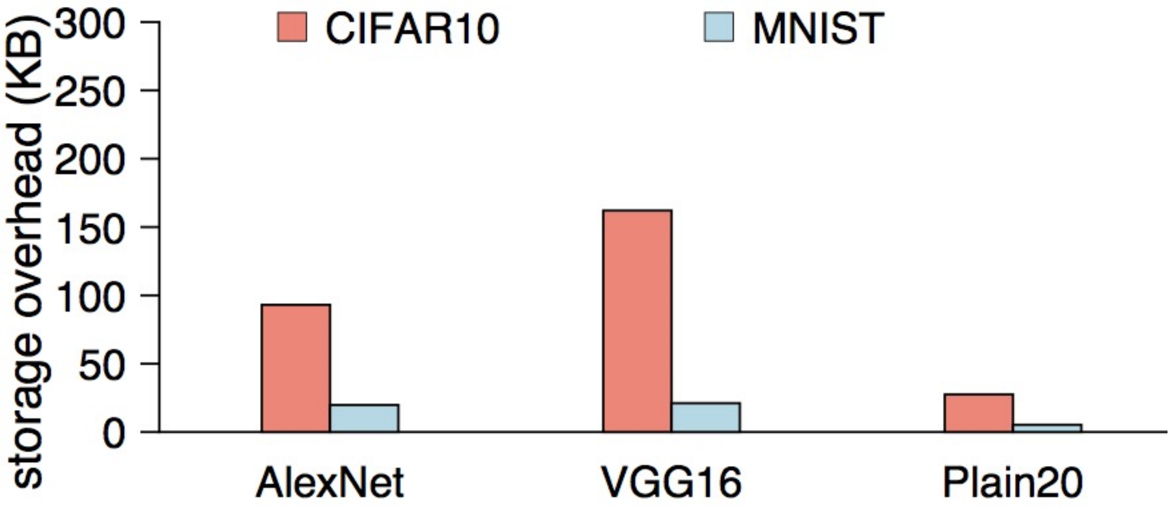


Figure 12: The storage overhead of Weight Index Buffer for different networks on CIFAR10 and MNIST respectively.

The index overhead is ignorable.

Conclusions

- AUTO-PRUNE is a hardware-aware automated DNN pruning and mapping framework for ReRAM-based accelerators. It leverages RL to automatically determine a global optimum pruning policy, considering the direct hardware feedback.

- We propose a new data-path to correctly index and feed input to matrix-vector computation.

- AUTO-PRUNE achieves up to 3.3X compression rate, 3.1X area efficiency, and 3.3X energy efficiency compared to PIM-Prune while maintaining a similar or even higher accuracy.

Thanks for your attention!

Siling Yang@ZJU
slingzjunet@zju.edu.cn

浙江大学
Zhejiang University