MICRO 2024

Cambricon-C: Efficient 4-bit Matrix Unit via Primitivization

Yi Chen

USTC & ICT, CAS

MICRO 2024

Cambricon-C: Efficient 4-bit Matrix Unit via Primitivization

Yi Chen
USTC SKLP, ICT, CAS
chen_yi@mail.ustc.edu.cn

Yongwei Zhao
SKLP, ICT, CAS
zhaoyongwei@ict.ac.cn

Yifan Hao
SKLP, ICT, CAS
haoyifan@ict.ac.cn

Yuanbo Wen
SKLP, ICT, CAS
wenyuanbo@ict.ac.cn

Yuntao Dai
USTC
daiyuntao@mail.ustc.edu.cn

Xiaqing Li
SKLP, ICT, CAS
lixiaqing@ict.ac.cn

Yang Liu
SKLP, ICT, CAS UCAS
liuyang22z1@ict.ac.cn

Rui Zhang
SKLP, ICT, CAS
zhangrui@ict.ac.cn

Mo Zou
SKLP, ICT, CAS
zoumo@ict.ac.cn

Xinkai Song
SKLP, ICT, CAS
songxinkai@ict.ac.cn

Xing Hu
SKLP, ICT, CAS SHIC
huxing@ict.ac.cn

Zidong Du
SKLP, ICT, CAS SHIC
duzidong@ict.ac.cn

Huaping Chen
USTC
hpchen@ustc.edu.cn

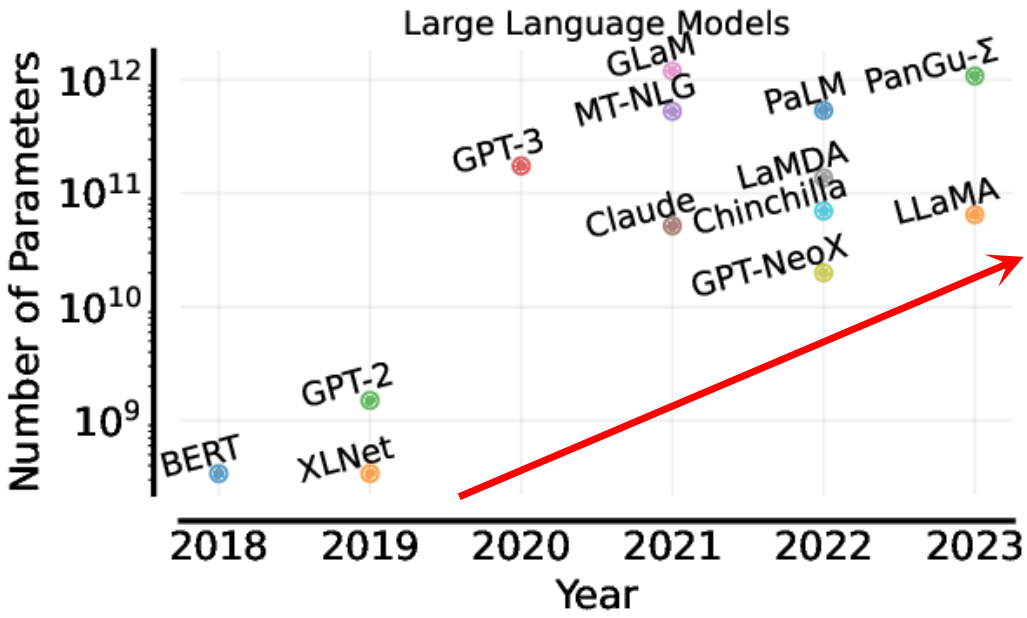
Qi Guo*
SKLP, ICT, CAS
guoqi@ict.ac.cn

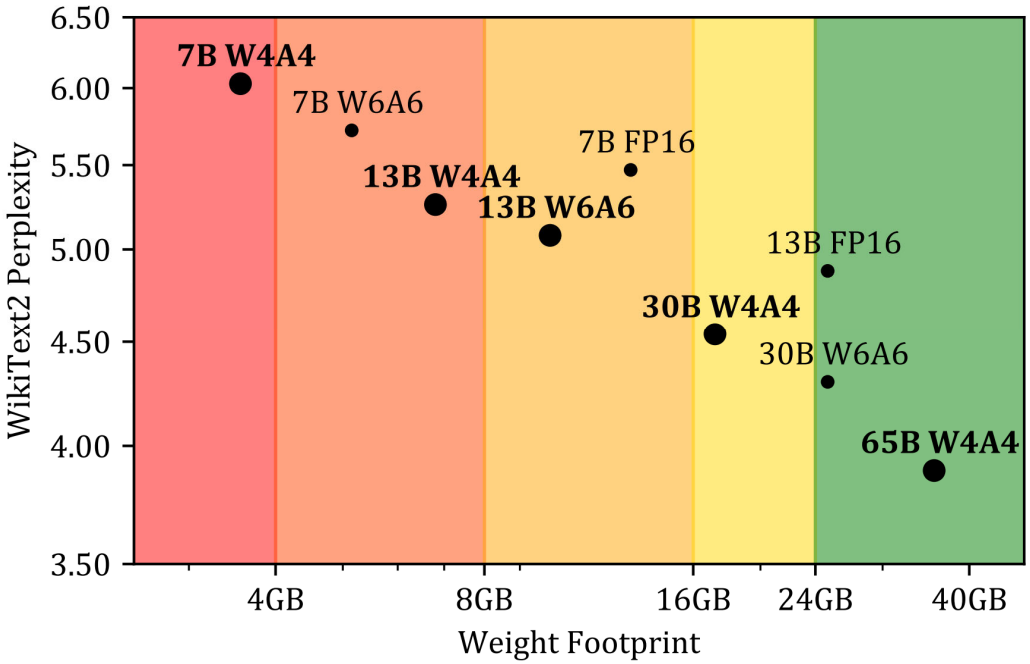
Tianshi Chen
Cambricon Technologies
tchen@cambricon.com

Outlines

- Background & Motivation
- Algorithm
  - Primitivized Matrix Multiplication
  - Quarter Square Multiplication
- Architecture
  - Overview
  - PE design
  - Converter
- Evaluation and Results
- Conclusion

Background: Emerging 4-bit Large Model





Model Size Increases

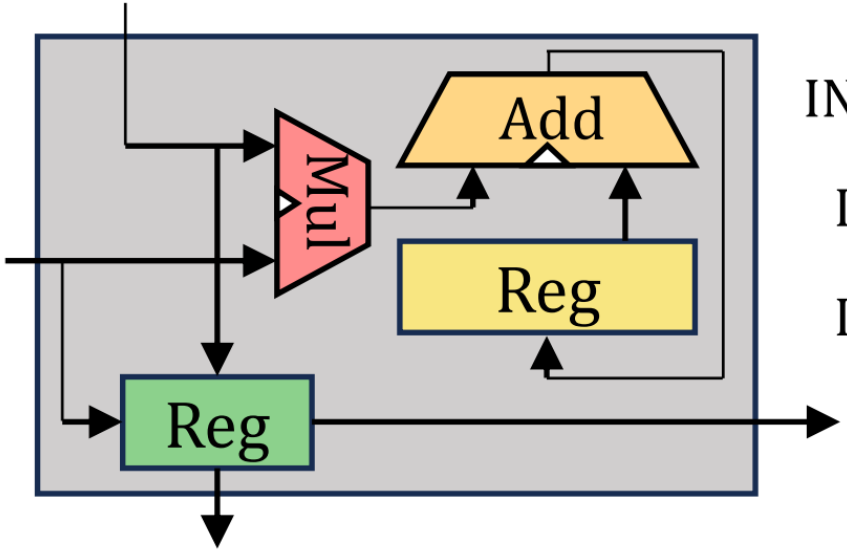
- Model performance is related to model scale
- Model size imposes significant demands on the computational resources

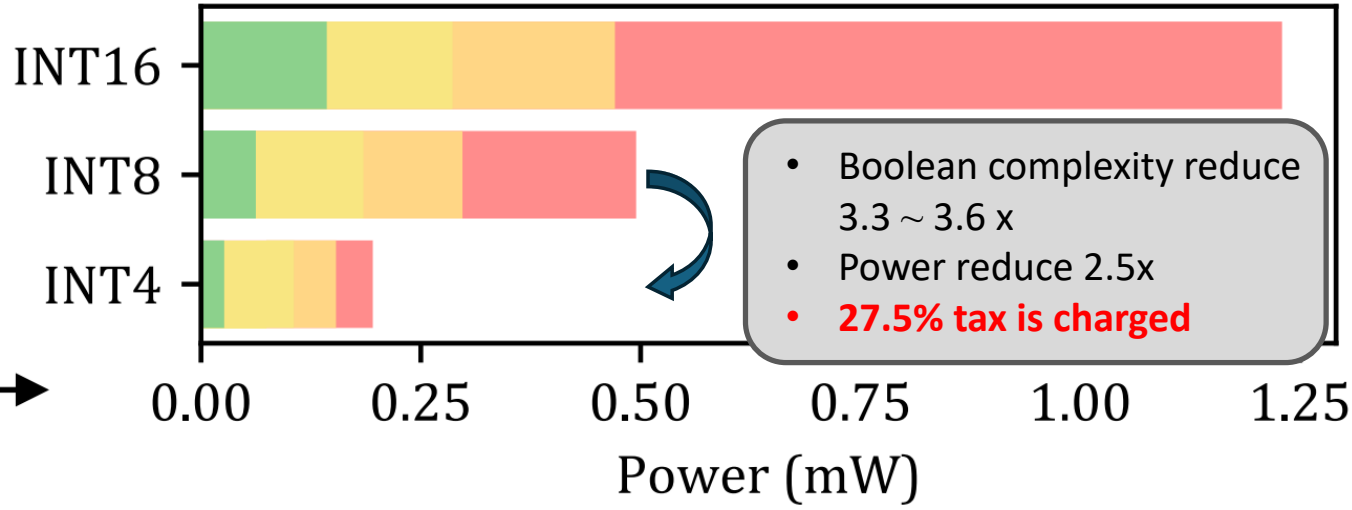
Quantization

Model Deployment

- 4-bit variants get best performance under the same memory budget.
- GMML, vLLM, TensorRT-LLM, Open-VINO

Background: Hardware Matrix Unit





1. Repeating Computation

- INT4 multiplication presents < 256 unique value.
- Matrix dimensions are often far > 256

2. Accumulator dominating

- n dimension INT4 matrix needs $8 + log_2 n$ bits Accumulator

27.5% Tax !

Algorithm: Primitivized Matrix Multiplication

Key1: Avoid duplicated multiplications by pre-computing a Look-up Table

Suppose an INT4 Multiplication:

$$y = \sum_{i=0}^{N-1} w_i x_i , \quad N \gg 256$$

Define pre-computed $\boldsymbol{m}$ includes all $w_i x_i$ possible values

$$\boldsymbol{m}=[\begin{matrix} 0\times 0, & 0\times 1, & 0\times 2, & ..., & 0\times -1, \\ 1\times 0, & 1\times 1, & 1\times 2, & ..., & 1\times -1, \\ 2\times 0, & 2\times 1, & 2\times 2, & ..., & 2\times -1, \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -1\times 0, & -1\times 1, & -1\times 2, & ..., & -1\times -1 \end{matrix}]$$

Formula can also be presented as:

$$y = \sum_{i=0}^{N-1} \boldsymbol{p}_i \boldsymbol{m}^T ,$$

Where $\boldsymbol{p}_i$ is the one-hot vector

$$\boldsymbol{p}_i = [p_0, p_1, ..., p_{255}] , p_j = \begin{cases} 1, & \text{if } j = x_i \circ w_i \\ 0, & \text{otherwise} \end{cases}$$

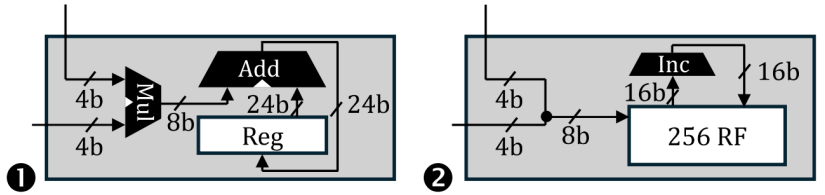
Key2: Reduce the strength of additions

With the distributive law of multiplication:

$$y = \sum_{i=0}^{N-1} \boldsymbol{p}_i \boldsymbol{m}^T = (\sum_{i=0}^{N-1} \boldsymbol{p}_i)\boldsymbol{m}^T$$

Major computation part: summation of **one-hot** vector

⇒ 256 number increment + final once multiplication

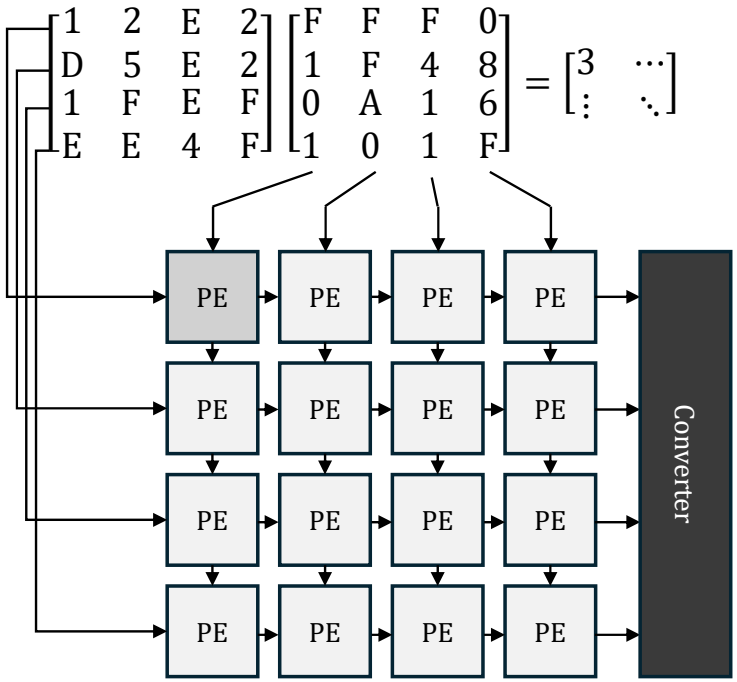


	Tr. Count	Energy	Conventional MM	Primitivized MM
4b Mul	241 T	0.227 pJ	× 11,008	× 256*
24b Add	679 T	0.584 pJ	× 11,007	× 255
❸ 16b Inc	122 T	0.028 pJ	/	× 11,008

(∗ required in the last step multiplying $\boldsymbol{m}^{\mathrm{T}}$, 8b×16b to be precise.)

MICRO 2024

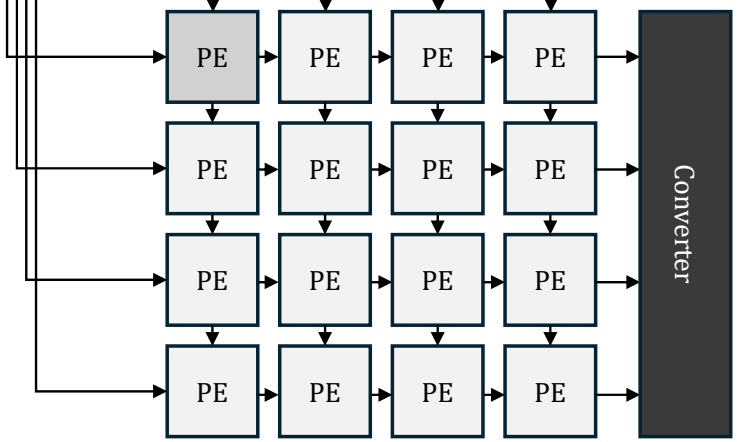
Algorithm: Primitivized Matrix Multiplication

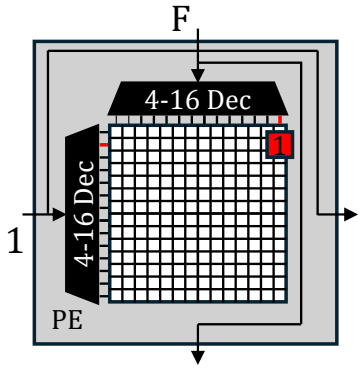


Algorithm: Primitivized Matrix Multiplication

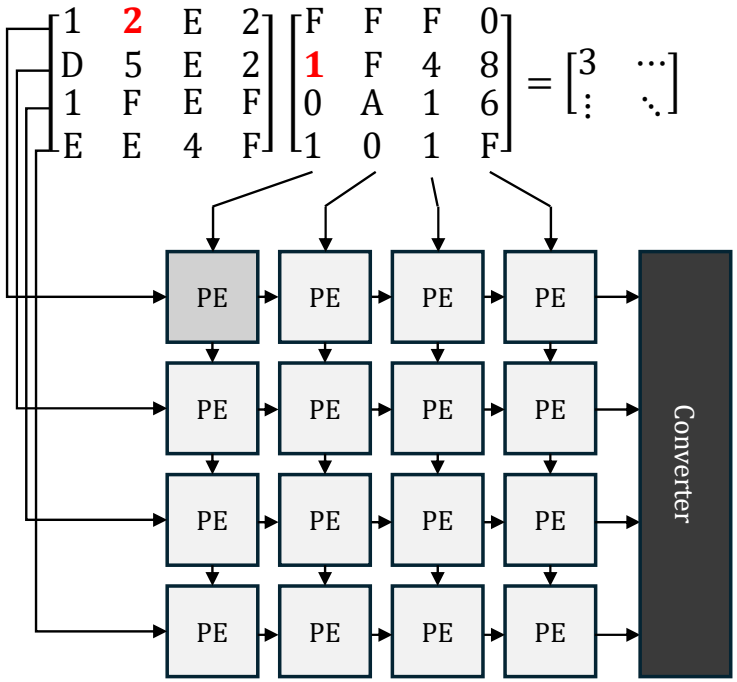
Compute $\sum_{i=0}^{N-1} \boldsymbol{p}_i$

$$\begin{bmatrix} \mathbf{1} & 2 & E & 2 \\ D & 5 & E & 2 \\ 1 & F & E & F \\ E & E & 4 & F \end{bmatrix} \begin{bmatrix} \mathbf{F} & F & F & 0 \\ 1 & F & 4 & 8 \\ 0 & A & 1 & 6 \\ 1 & 0 & 1 & F \end{bmatrix} = \begin{bmatrix} 3 & \cdots \\ \vdots & \ddots \end{bmatrix}$$

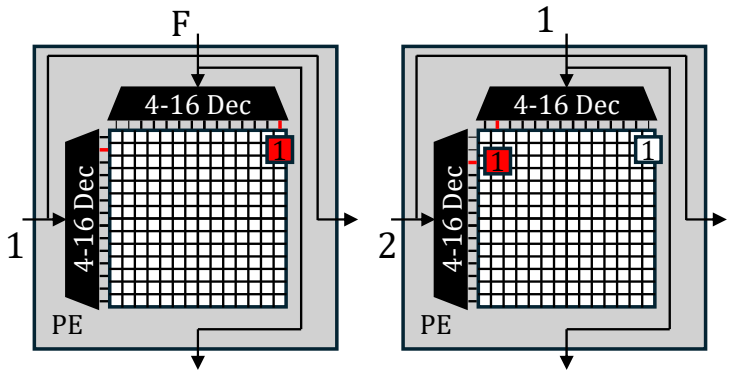




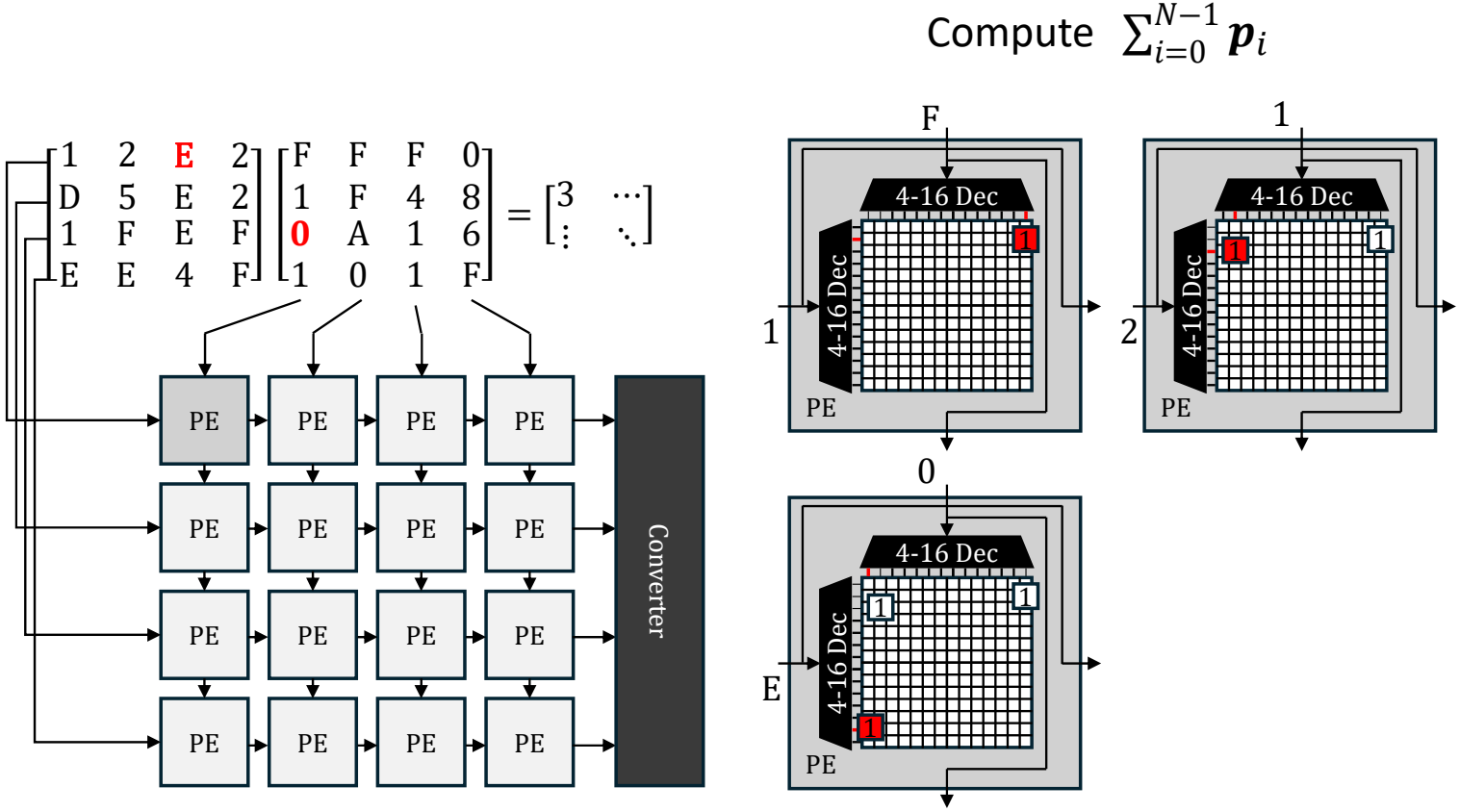
Algorithm: Primitivized Matrix Multiplication



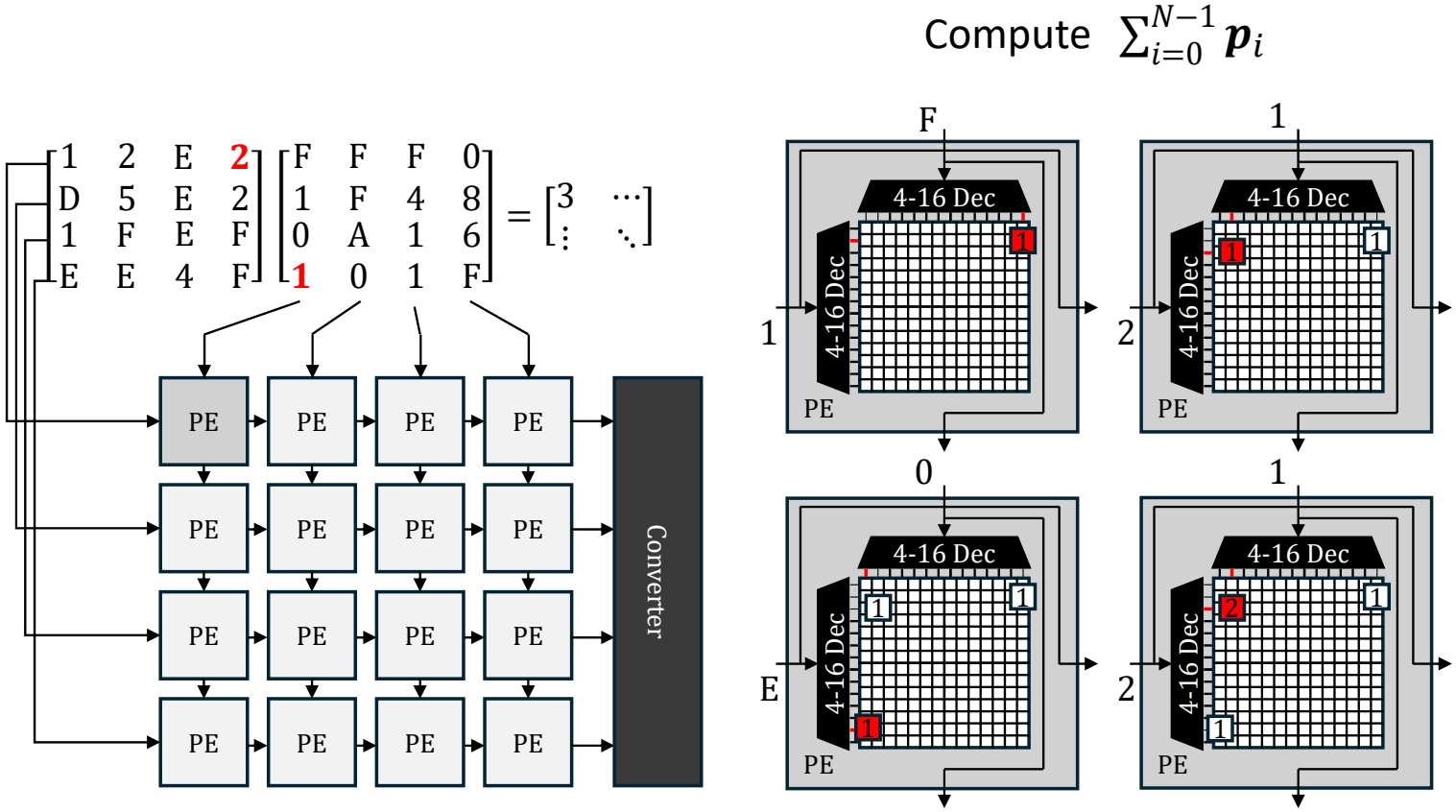
Compute $\sum_{i=0}^{N-1} \boldsymbol{p}_i$



Algorithm: Primitivized Matrix Multiplication

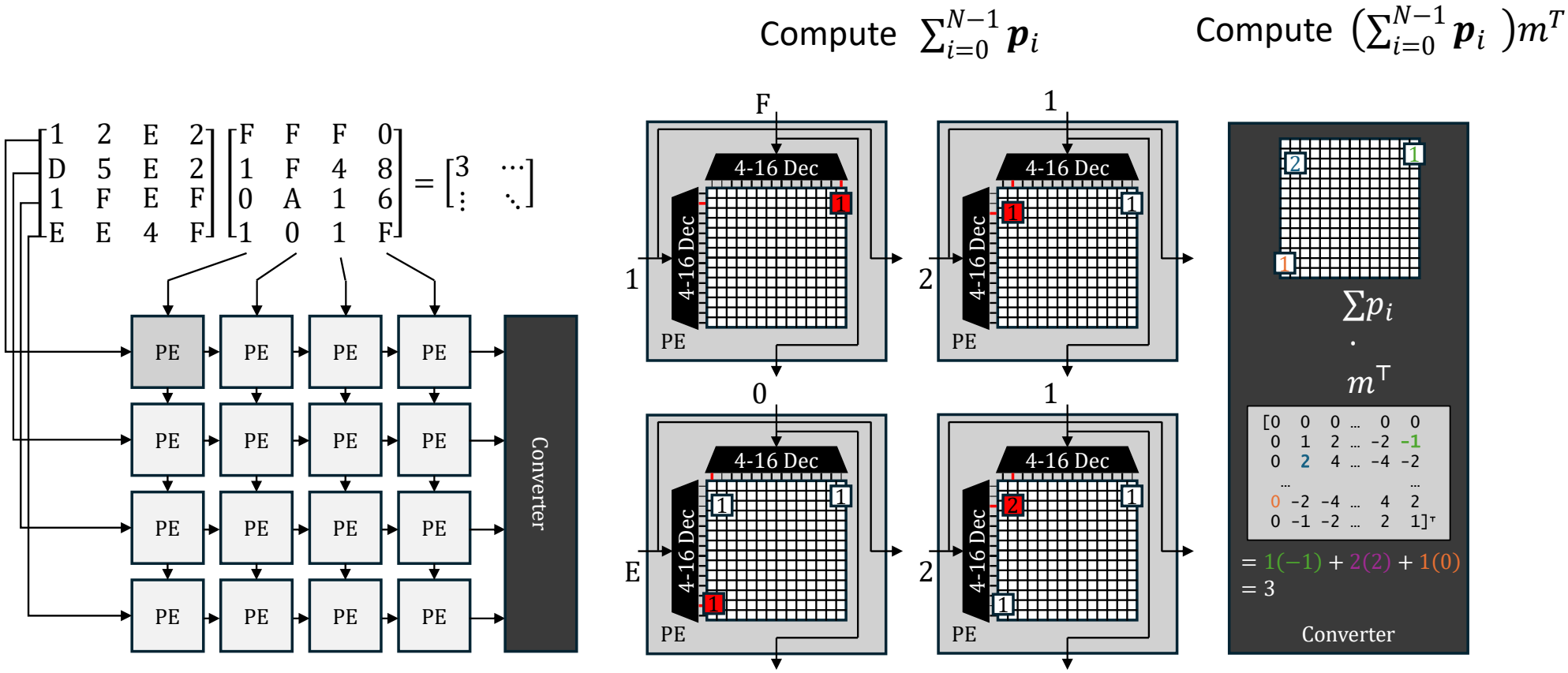


Algorithm: Primitivized Matrix Multiplication



MICRO 2024

Algorithm: Primitivized Matrix Multiplication



Algorithm: Quarter Square Multiplication

Key: Reduce the number of counters

It is intuitively obvious that

$$\frac{(x+y)^2}{4} - \frac{(x-y)^2}{4} = xy$$

As x and y are integers.

$\Rightarrow x+y$ and $x-y$ are both odd or even

$\Rightarrow$ The remainders will be cancelled.

i.e. $(x+y)^2 \equiv (x-y)^2 \ mod\ 4$

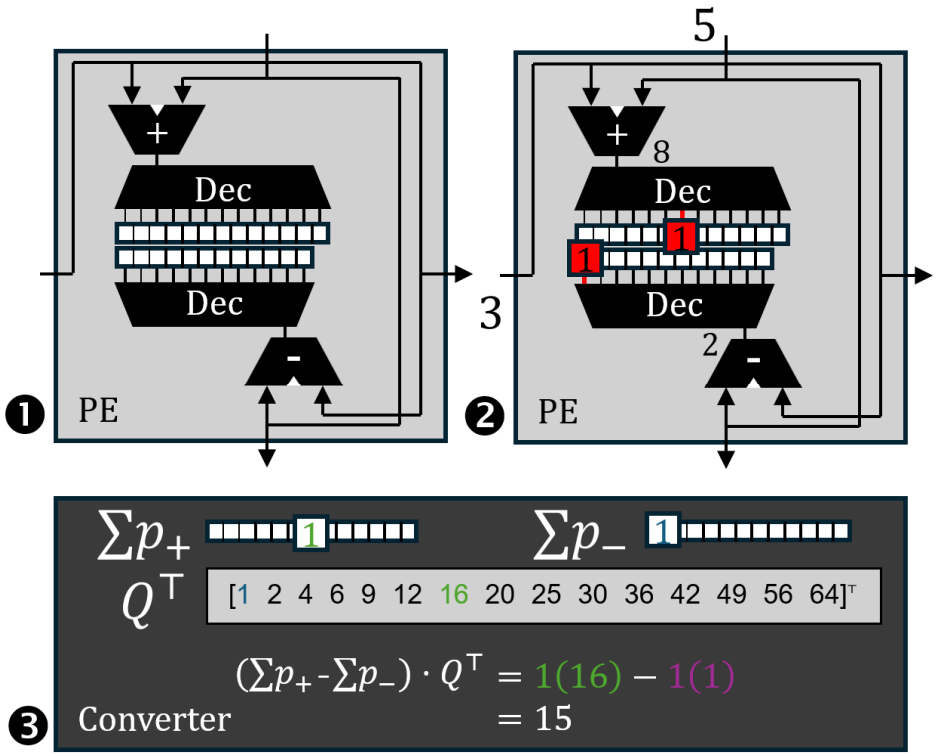
$$\Rightarrow \left\lfloor \frac{(x+y)^2}{4} \right\rfloor - \left\lfloor \frac{(x-y)^2}{4} \right\rfloor = xy, \quad x,\ y \in \mathrm{Z}$$

The pre-compute look-up table becomes $\boldsymbol{Q}(n) = \lfloor n^2/4 \rfloor$

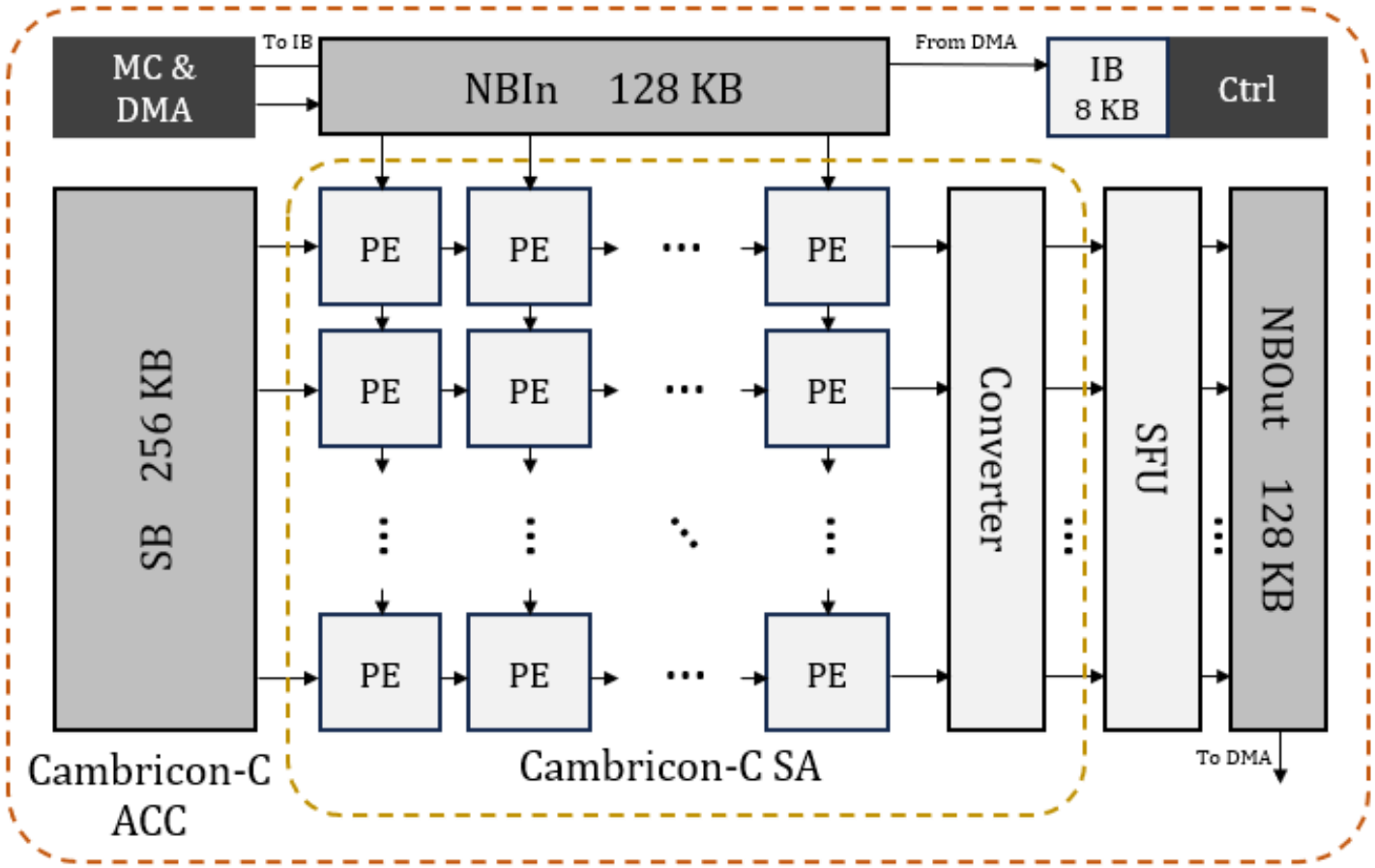
As $\boldsymbol{Q}(0) = \boldsymbol{Q}(1) = 0$ and $\boldsymbol{Q}(n) \equiv Q(-n)$

$x+y \in [-16,14]$, $x-y \in [-15,15]$,

$\Rightarrow \boldsymbol{Q} = [1,2,4,6,9,12,16,20,25,30,36,42,49,56,64]$

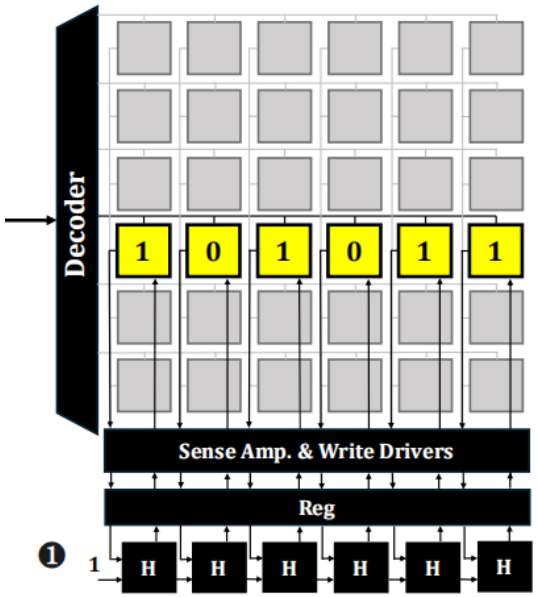


Architecture: Overview



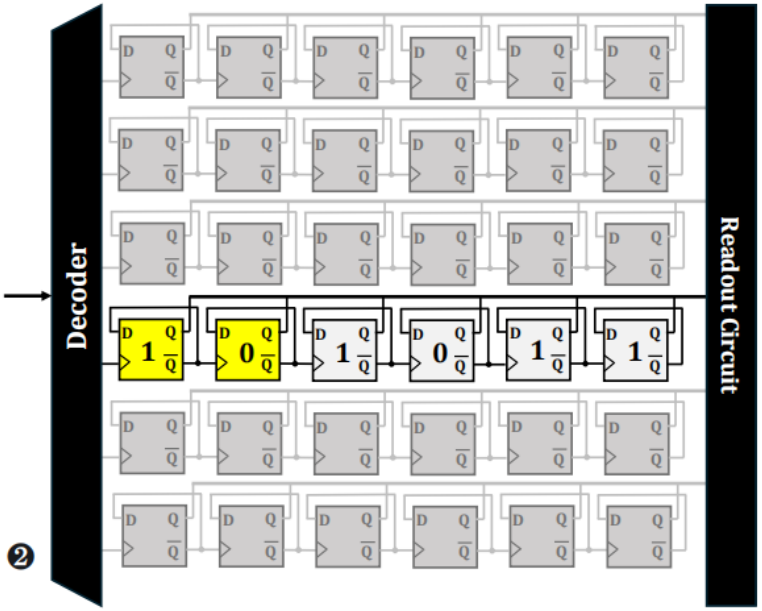
- Cambricon-C ACC
  - Cambricon-C SA
    - 32x32 PE array
    - Converter
  - 128KB NBIn
  - 128KB NBOut
  - 256KB SB
  - Controller (Ctrl)
  - SFU
  - 8KB Instruction buffer (IB)
  - Direct memory access module (DMA)

Architecture: Counters



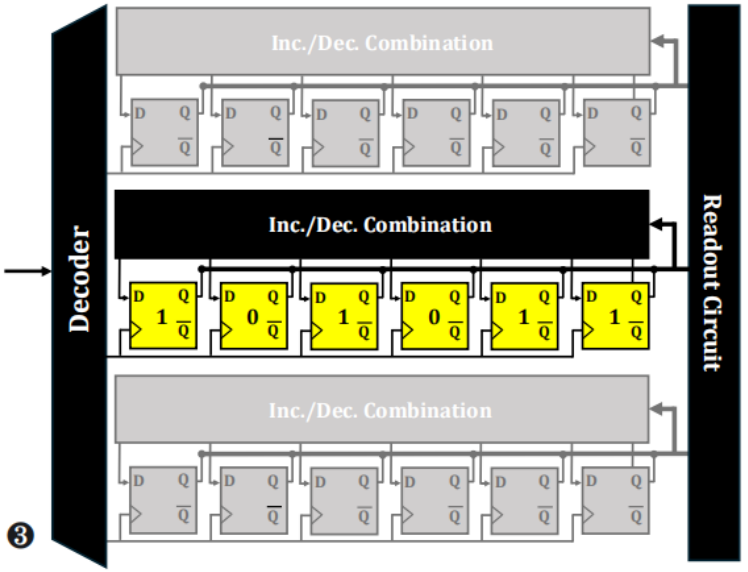
SRAM-based Counters

+ Lower leakage power
+ Lower area
- Inefficient incrementing
- Hard to customize



Ripple Counters

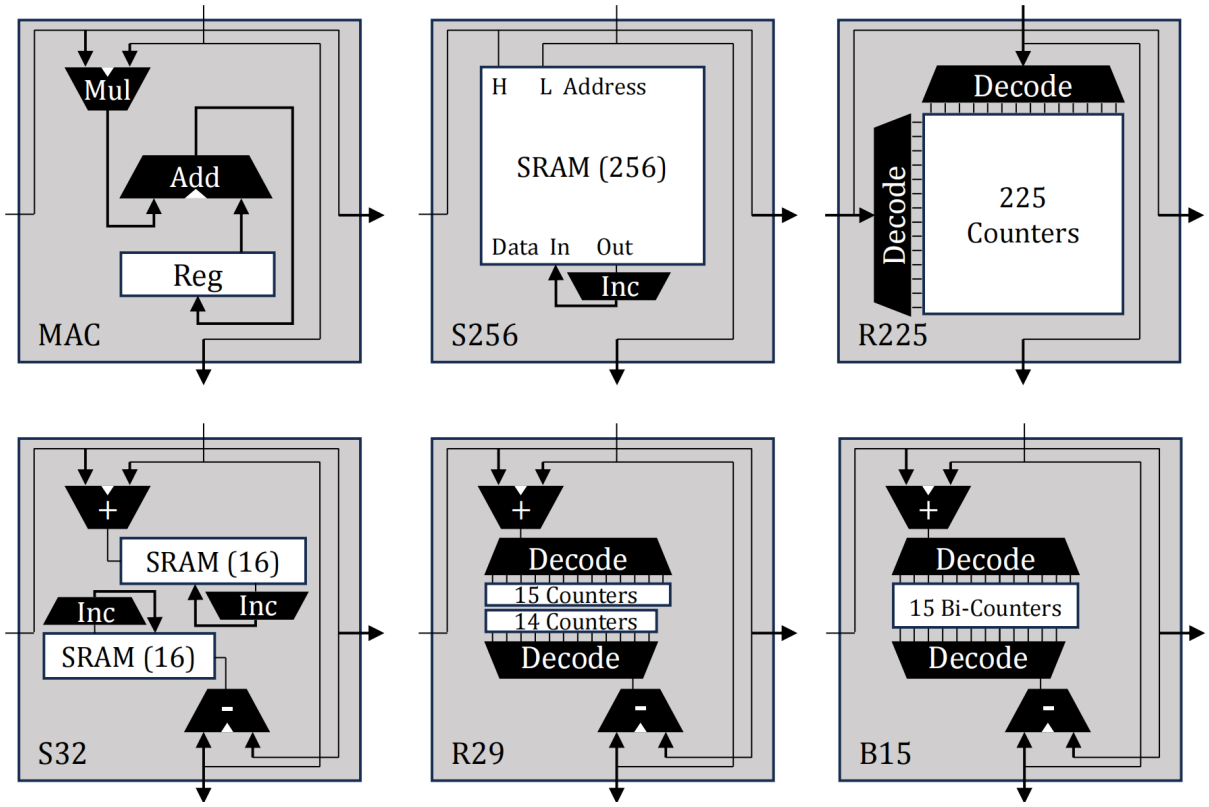
+ Efficient incrementing
+ High speed
+ Easy to customize
- Large area and leakage power
- Unstable while incrementing



Bi-directional Counters

+ Combine up/down counters
+ Easy to customize
- High leakage power
- Less efficient.

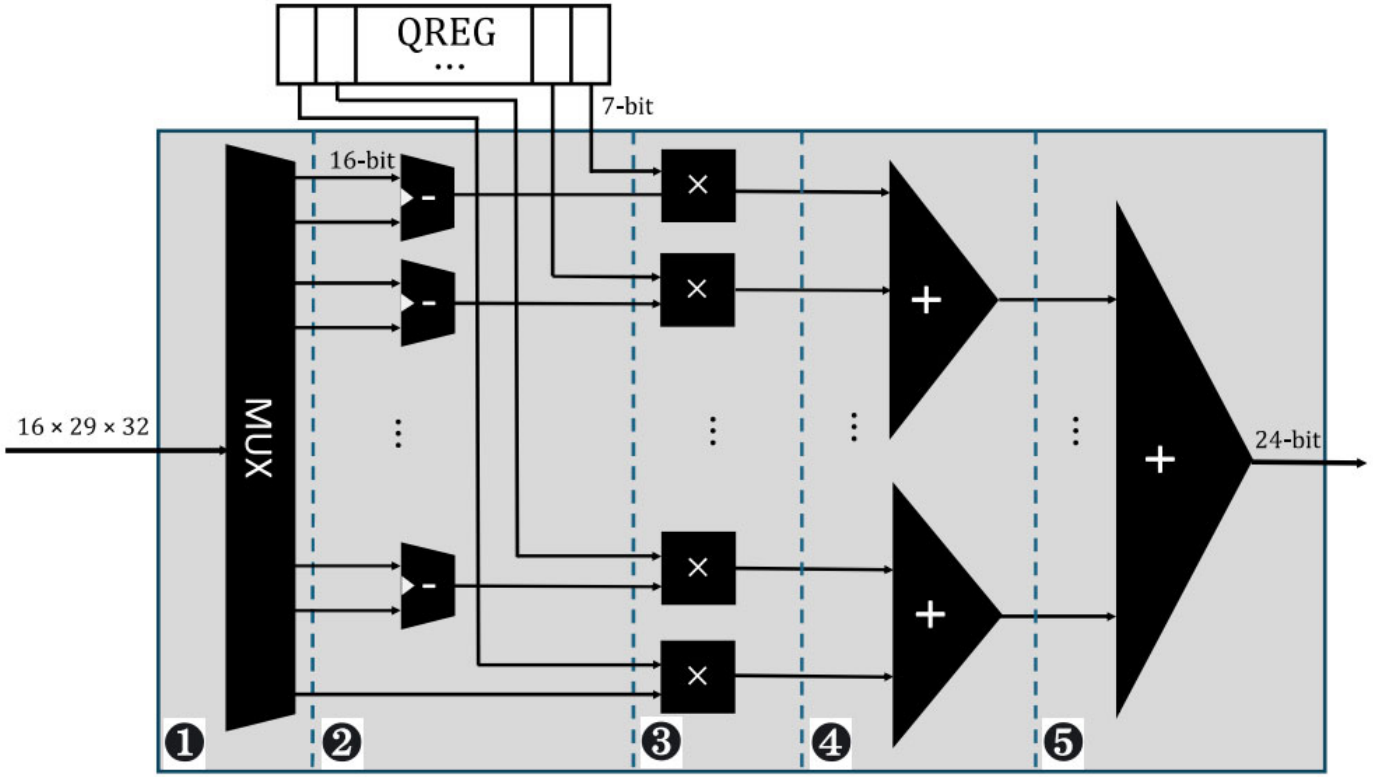
Architecture: PE Designs



PE Design	Area (um^2)	Energy (pJ)	Frequency (MHz)
MAC	**1422**	2.508	357
S256	21480	3.367	**400**
R225	57113	2.525	**400**
S32	13045	2.439	**400**
R29	8914	**1.274**	**400**
B15	9821	2.734	300

- Sacrifice area to gain **lower energy**
- **QSM** significantly increases the PE performance.
- **R29** outperform other schemes
- **S32** and **B15** has larger area and power. As it introduces more combination circuit.

Architecture: Converter



- Converter for R29-PE array
  - Counter Readout
  - Substraction of counter values
  - Multiplication by Pre-loading Value
  - Final summation

Evaluation: Implementation

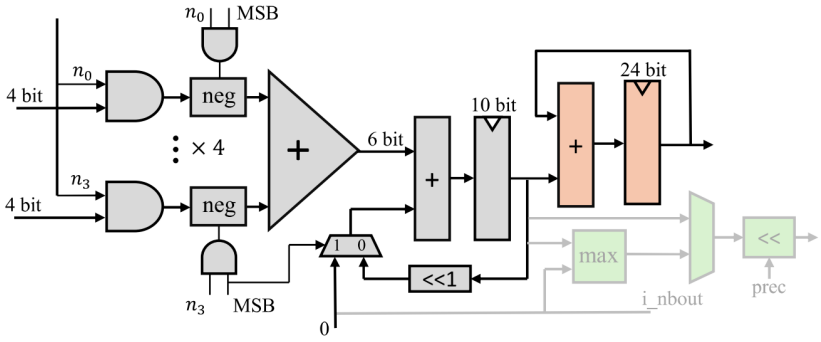
Software

Model	# Param.	Quantization	Avg. Rank
VGG-16	138 M	LSQ* [11]	1141
ResNet-18	11.2 M		730
BERT-Base	110 M	KDLSQ* [22]	789
BERT-Large	340 M		1045
OPT-1.3B	1.3 B	QLLM [27]	2235
OPT-2.7B	2.7 B		2664
LLaMA2-7B	6.6 B		4013
LLaMA2-13B	12.7 B		5002

- 8 DNN models
- Compute same output value with the same quantization scale factor. (Increment will not be interrupted)
- **Rank**: number of MAC operations to compute one output value in matrix multiplication.

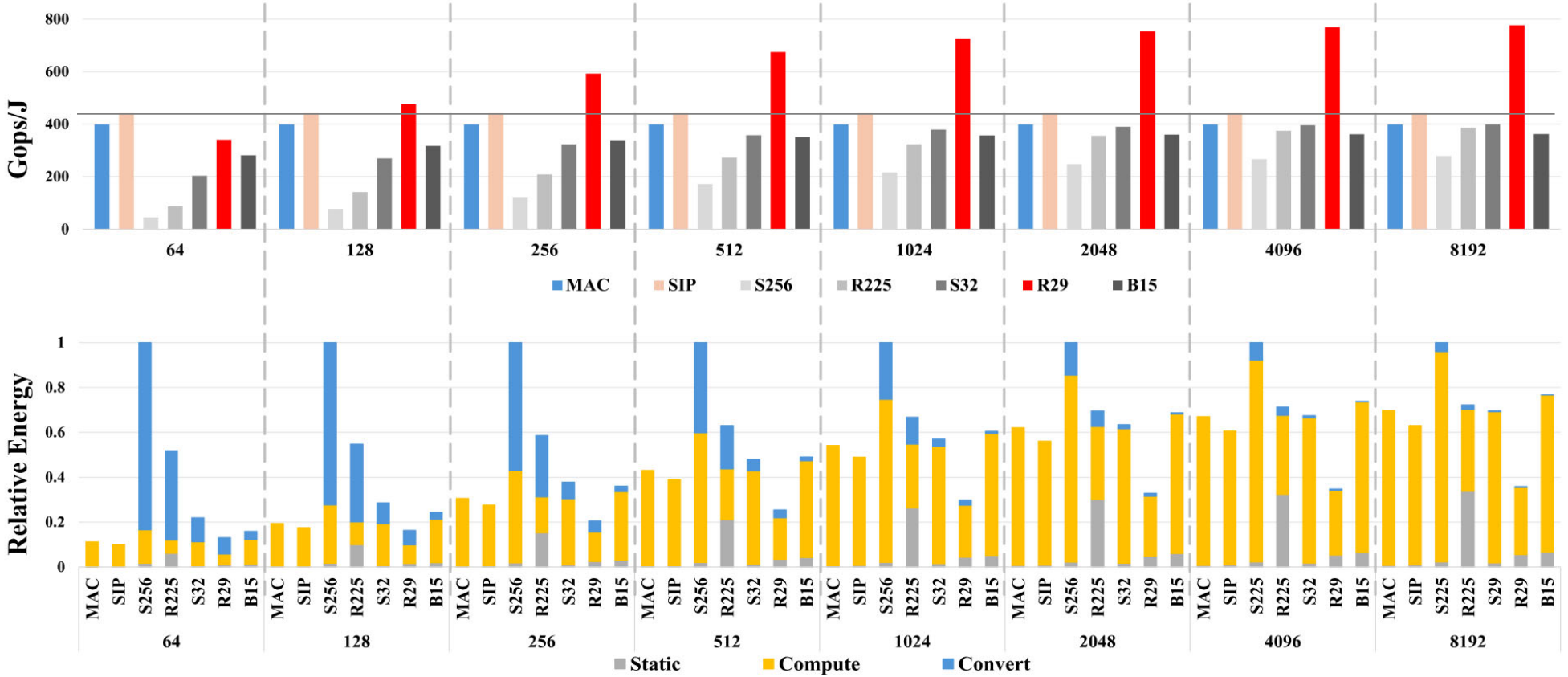
Hardware

- PE: MAC, SIP, S256, R225, S32, R29, and B15
- ACC: TPU, Stripes, Camb-C ACC (R29 Based)
- Implementation:
  - SRAM: Open-RAM and CACTI-7
  - Other designs: Verilog
  - Synthesis: Design Compiler
  - PDK: FreePDK 45nm
- Simulation:
  - Power: gate-level VCD and Prime-Time PX
  - Clock Frequency: SDF file and VCS
  - Simulator: DNNWeaver and ANT



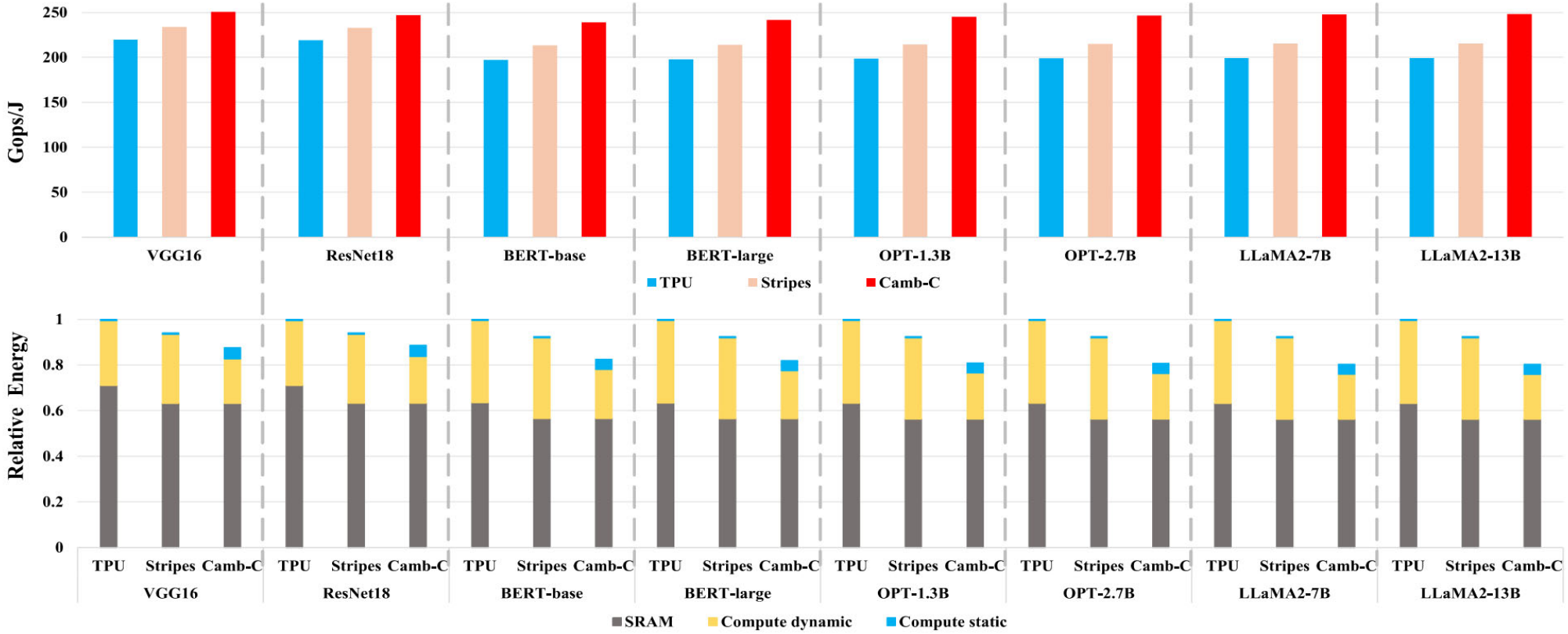
MICRO 2024

Evaluation: Matrix dimension



- R29 outperform other PEs when matrix dimensions reach 128.
- The proportion of conversion energy decreases with the dimension increasing.
- The maximum benefit of R29 reaches 1.95x than MAC and 1.76x than SIP

Evaluation: Performance on DNN



- Cambricon-C consistently outperforms TPU and Stripes in various DNN models.
- Cambricon-C gains 1.15x and 1.25x improvement compared with Stripes and TPU.

Evaluation: Comparison with Other Methods

Design	Energy (pJ)	Frequency (MHz)
MAC [33]	2.508	357
SIP [23]	2.265	400
PIP [1]	2.549	400
FuseKNA [40]	6.450	400
CARAT [34]	6.638	400
R29	**1.273**	400

- PIP: Skip zero-bit in bit-serial compute.
  - Extra decoder and shifter
- FuseKNA: compute with Look-up table
  - LUT-based AGU involves multiple registers
- CARAT: Avoid repeating value computation
  - Suitable for float point data, but not INT.
  - longer array size to shed through put loss.

The circuit design for INT4 Computation is quite simple and neat, making it not easy to achieve net gains.

Conclusion

- Algorithm:
  - Primitivized matrix multiplication
  - Quarter square multiplication
- Architecture
  - 5 different PE design
  - Converter
  - Cambricon-C accelerator
- Evaluation and Results
  - R29 gets 1.95x energy saving than MAC
  - Cambricon-C ACC get 1.25x energy saving than TPU

Thanks!

Q & A

Feel free to contact me on
Whova app or e-mail: chen_yi@mail.ustc.edu.cn