ISSCC 2022

SESSION 29

ML Chips for Emerging Applications

184QPS/W 64Mb/mm^2 3D Logic-to-DRAM Hybrid Bonding with Process-Near- Memory Engine for Recommendation System

Dimin Niu[1], Shuangchen Li[1], Yuhao Wang[1], Wei Han[1], Zhe Zhang[2], Yijin Guan[2], Tianchan Guan[3], Fei Sun[1], Fei Xue[1], Lide Duan[1], Yuanwei Fang[1], Hongzhong Zheng[1], Xiping Jiang[4], Song Wang[4], Fengguo Zuo[4], Yubing Wang[4], Bing Yu[4], Qiwei Ren[4], Yuan Xie[1]

[1]Alibaba DAMO Academy, Sunnyvale, CA, [2]Alibaba DAMO Academy, Beijing, China, [3]Alibaba DAMO Academy, Shanghai, China, [4]UnilC, Xi'an, China

Self Introduction

Education Background

- B.S and M.S degree in electronic engineering from Tsinghua University
- Ph.D. degree in computer engineering from Pennsylvania State University

Work Experience

- Computing Technology Lab, DAMO Academy, Alibaba since 2019
- Memory Solutions Lab, Samsung Semiconductor 2014 - 2019

Research Interests

- Computer Architecture, Computing in/with Memory, Non-volatile memory

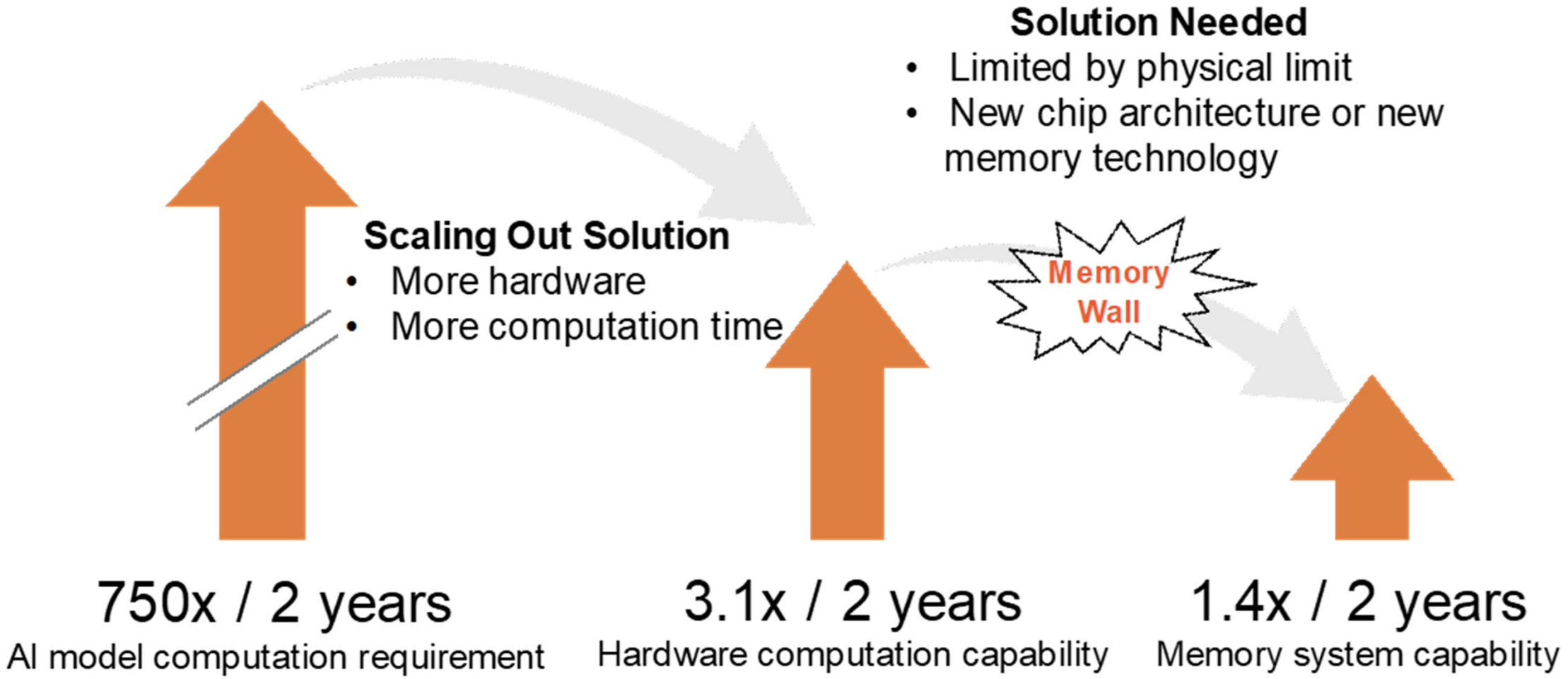
Outline

- **Motivation**
- **System and Chip Architecture**
  - 3D Logic-to-DRAM Hybrid Bonding
  - PNM Engine for Recommendation System
- **Measurement Results**
- **Conclusion**

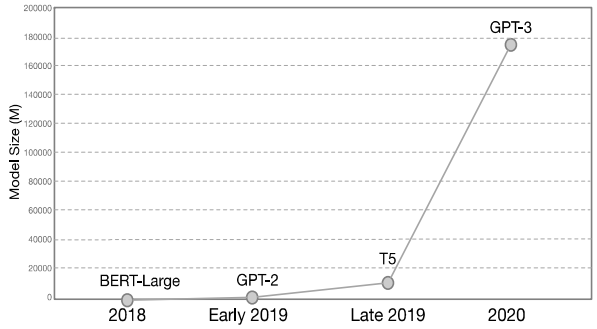
Outline

- **Motivation**
- System and Chip Architecture
  - 3D Logic-to-DRAM Hybrid Bonding
  - PNM Engine for Recommendation System
- Measurement Results
- Conclusion

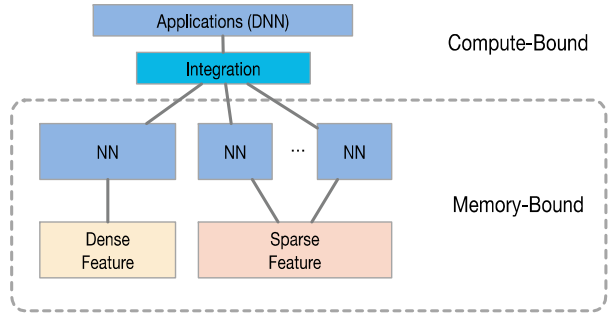
Memory Wall in AI Era



Memory-Bound Applications

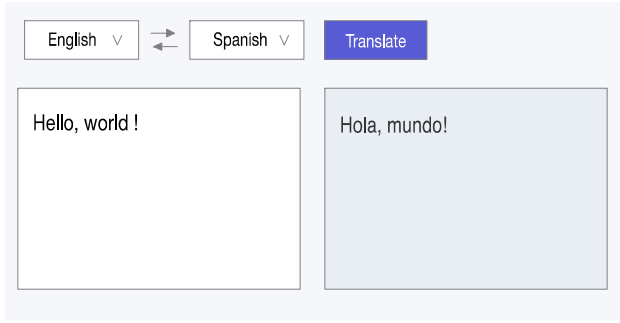


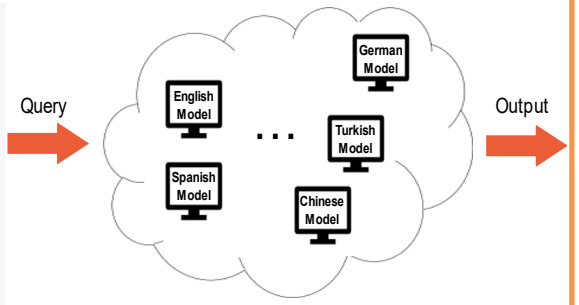
Natural Language Processing



Recommendation Systems

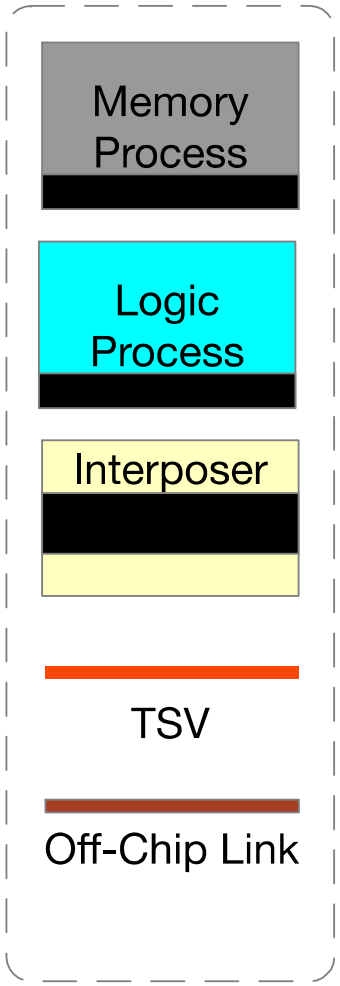
Graph Neural Network

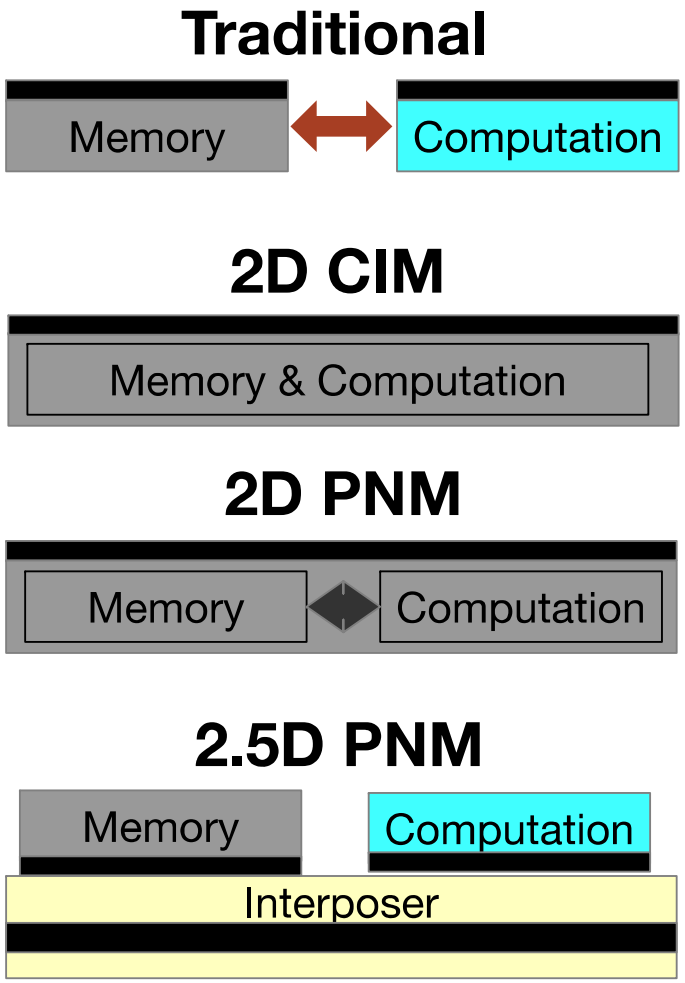


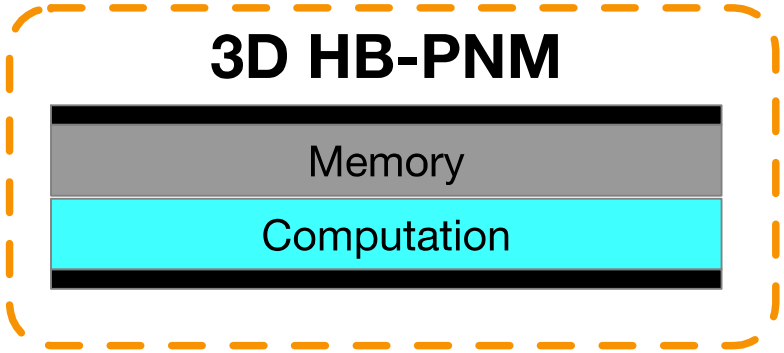


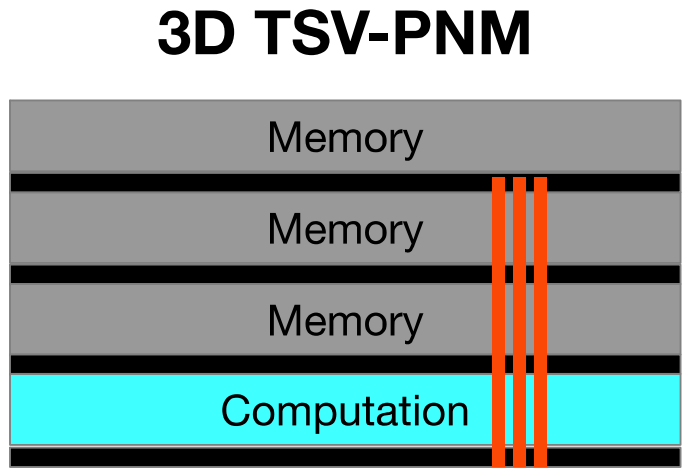
Multi-Task Online Inference

State-of-the-art PNM/CIM Solutions







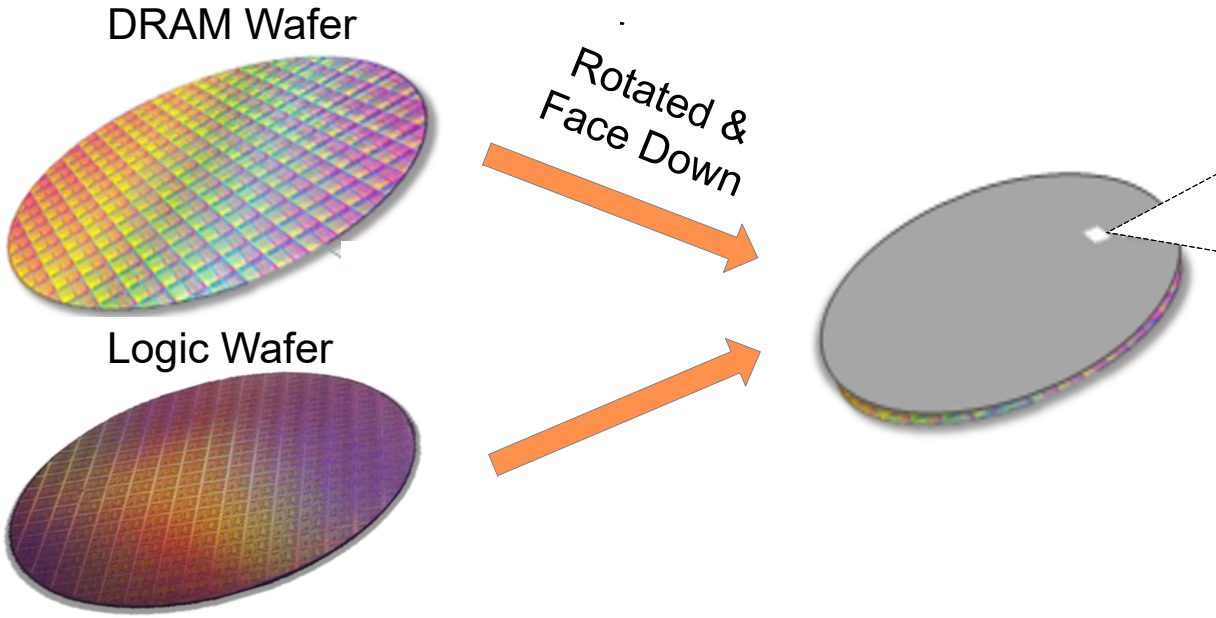


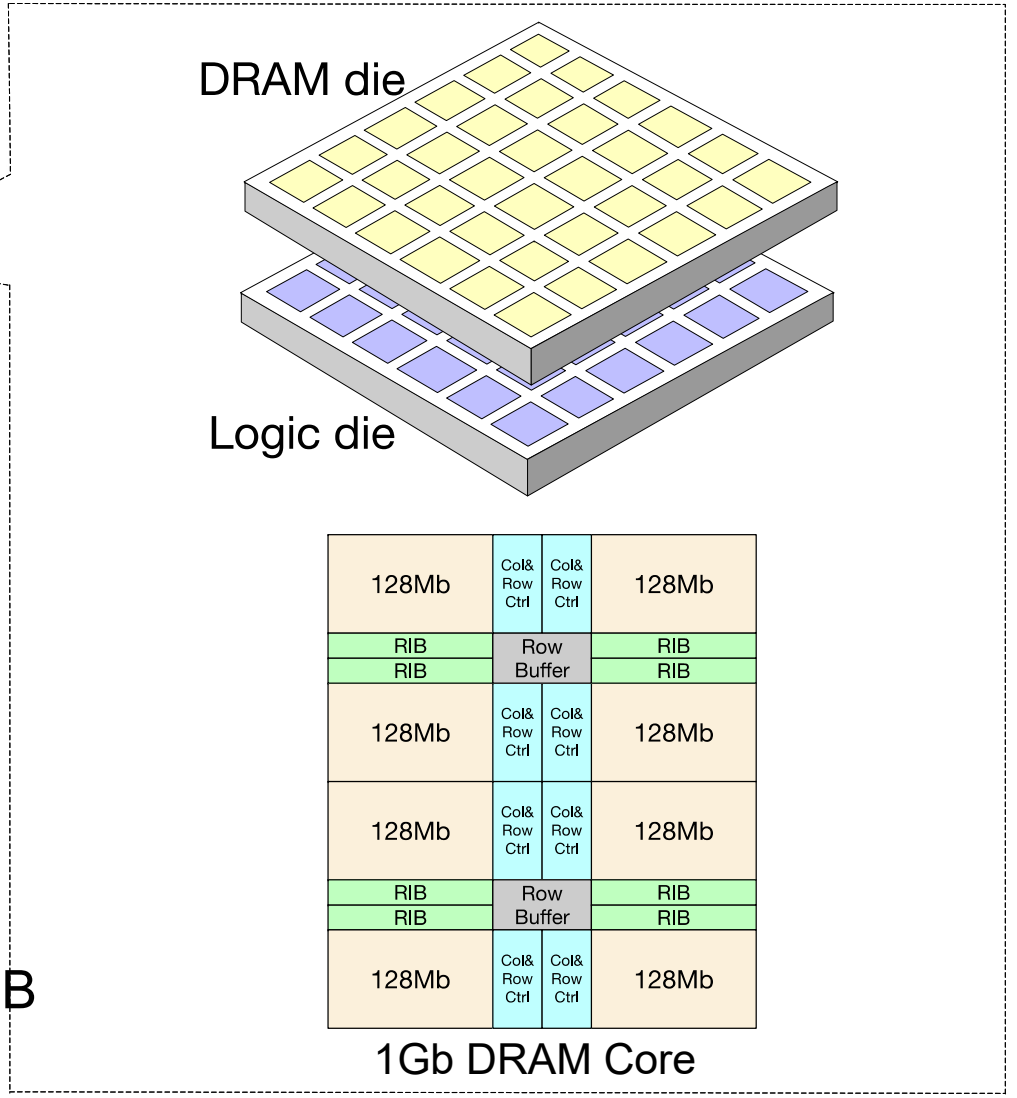
PNM : Process Near Memory
CIM: Compute In Memory
HB : Hybrid Bonding
TSV: Through-silicon Via

Outline

- Motivation
- **System and Chip Architecture**
  - 3D Logic-to-DRAM Hybrid Bonding
  - PNM Engine for Recommendation System
- Measurement Results
- Conclusion

3D Logic-to-DRAM Hybrid Bonding

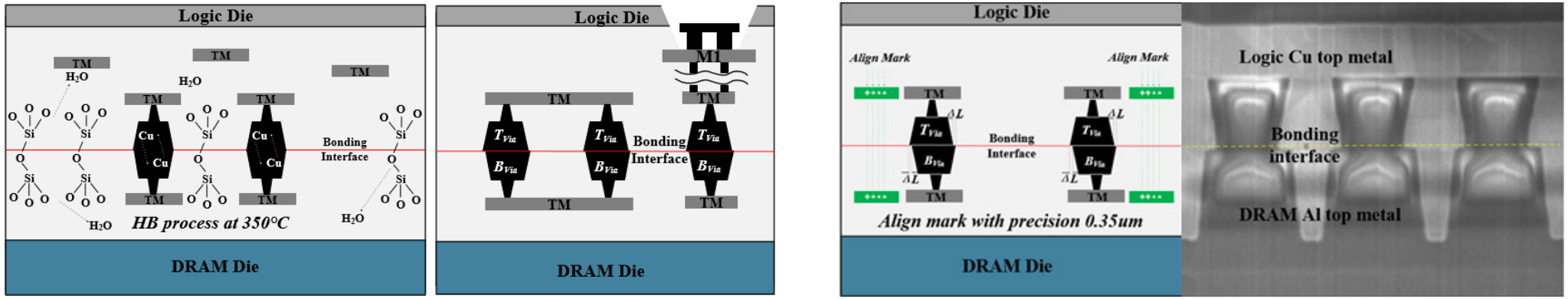




- Logic-to-DRAM face-to-face **H**ybrid wafer **B**onding
- 25nm DRAM technology with 36 x 1Gbits array
- 1Gbits DRAM core with 8 banks and on-chip ECC
- Each bank with 128 bits I/O, and implemented with HB

Hybrid-bonding Interconnection

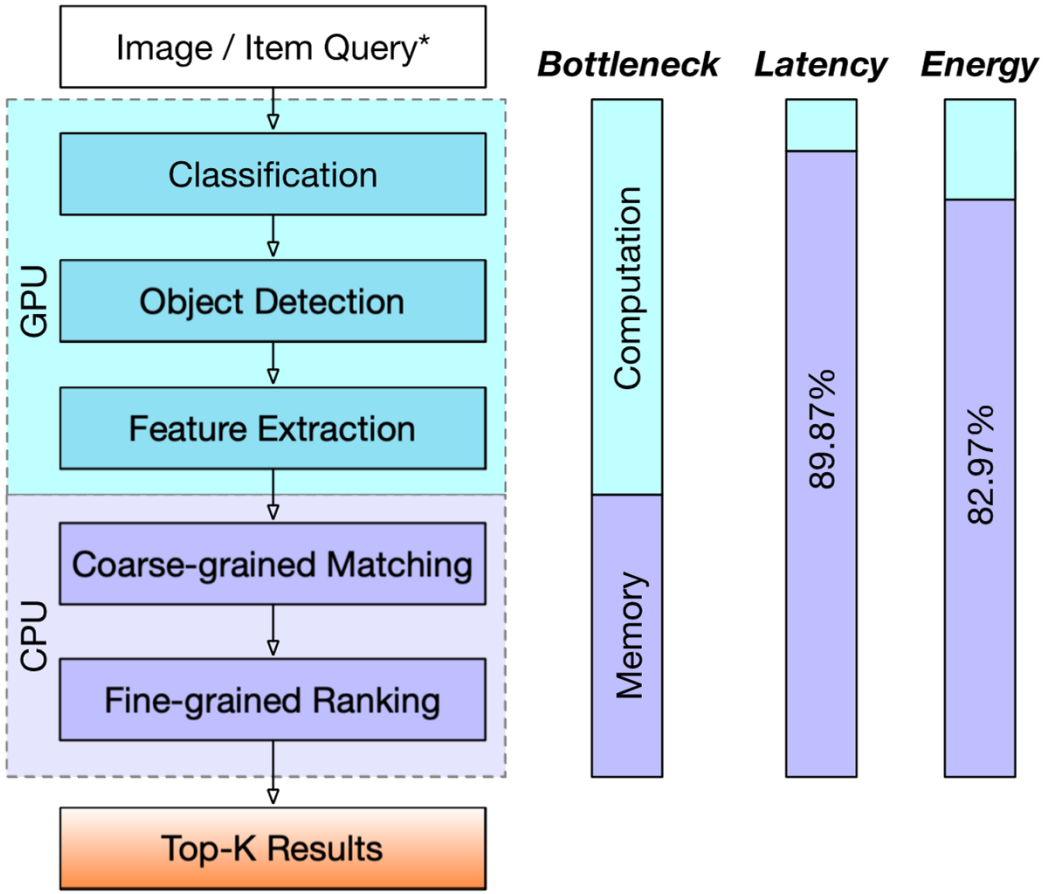
- Cu-Cu direct fusion with low bonding temperature (< 350°C)
- Up to 110,000/mm² integration density
- Small pitch size of 3μm
- Align marker with high precision of 0.35μm



Outline

- Motivation
- **System and Chip Architecture**
  - 3D Logic-to-DRAM Hybrid Bonding
  - PNM Engine for Recommendation System
- Measurement Results
- Conclusion

Typical Recommendation System

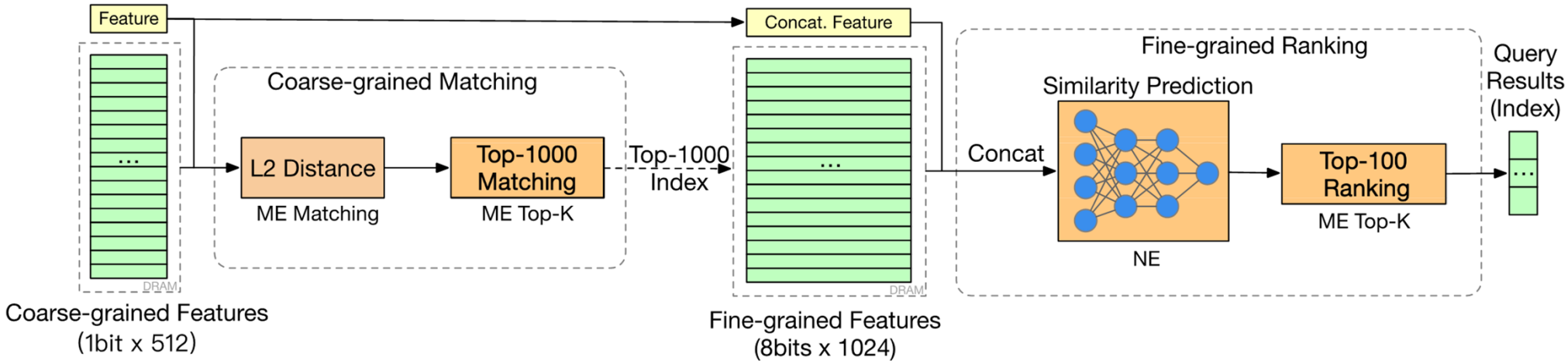


*Item feature can be extracted from different methods. Here is a typical case for image queries.

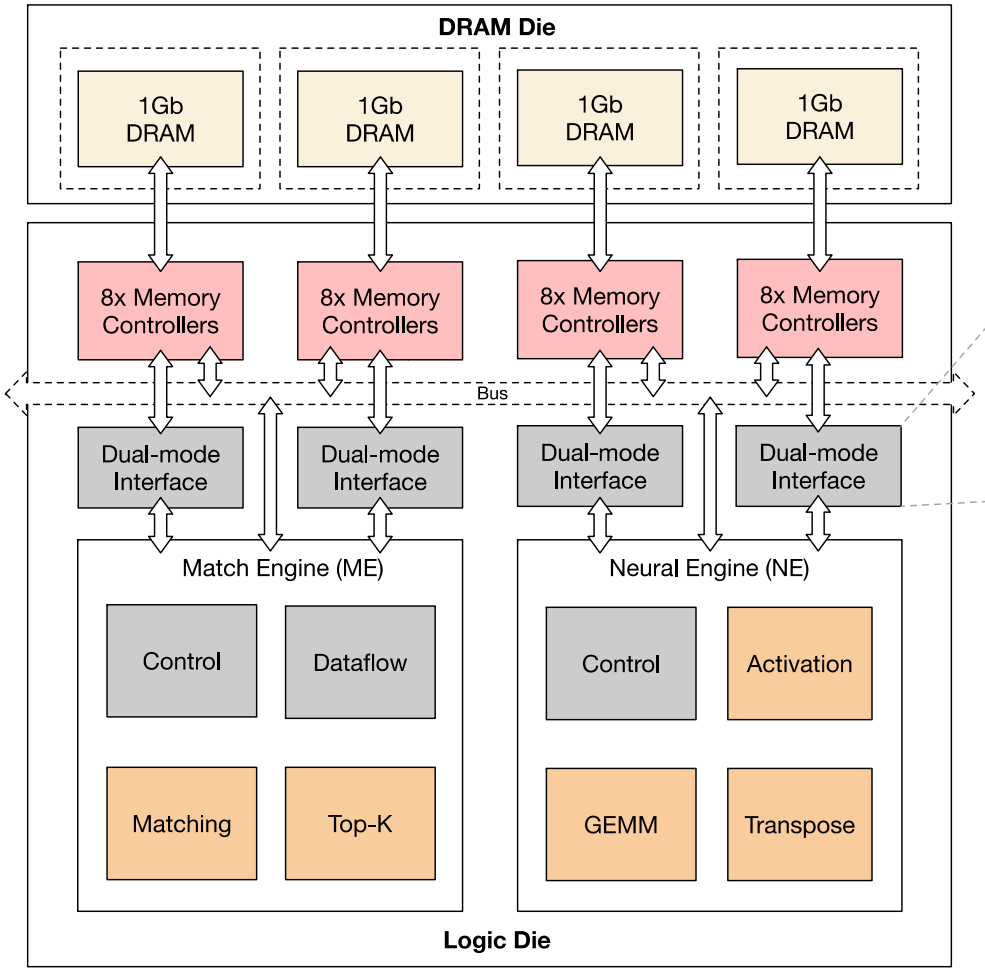
- A two-step Recommendation System
  - Feature Generation
    - Classification, object detection and feature extraction
    - Computation-bound
    - Typically executed on GPU
  - Matching & Ranking
    - Coarse-grained matching and fine-grained ranking
    - Memory-bound
    - Typically executed on CPU and commercial DRAM as external memory
    - Consumes most latency (**89.87%**) and energy (**82.97%**)
    - Requires **high-bandwidth, large-capacity and energy-efficient** memory

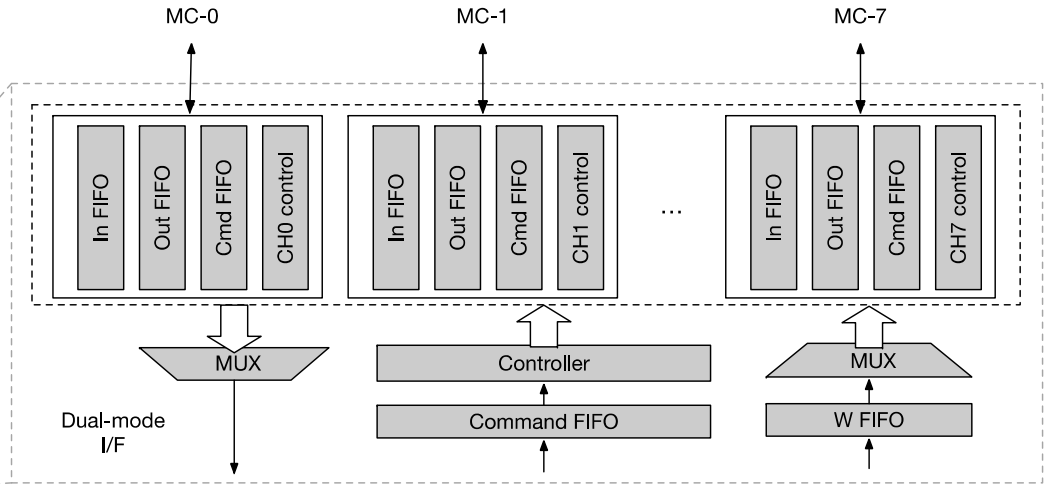
Ranking & Matching

- Coarse-grained Matching
  - Coarse-grained features with 1bit x 512 dimensions
  - Matching: L2 distance calculation
  - Top-1000 items selected from 40K items
- Fine-grained Ranking
  - Fine-grained features with 8bits x 1024 dimensions
  - Similarity prediction: three-layer MLP (2048-256-64-1)
  - Top-100 ranking results selected from 1K items



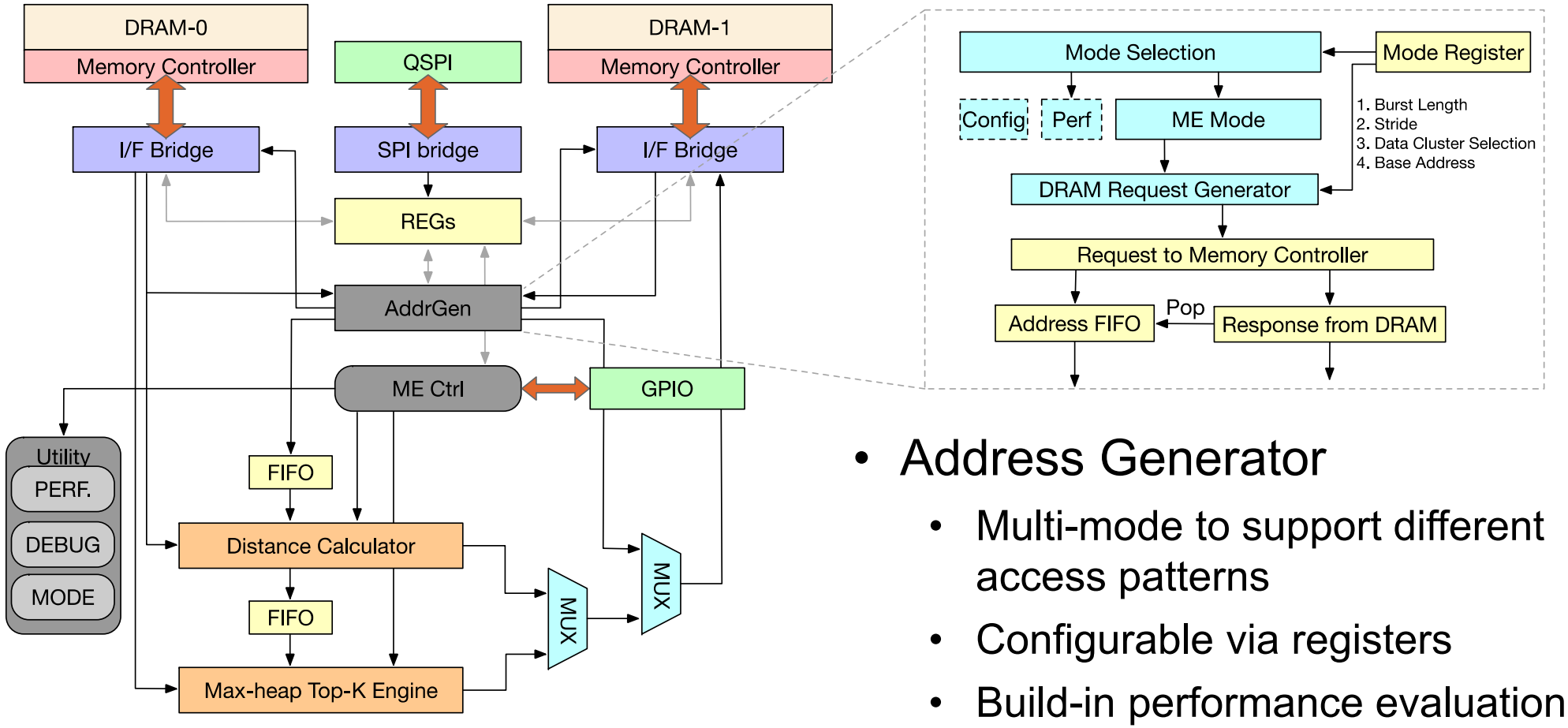
Overall Architecture





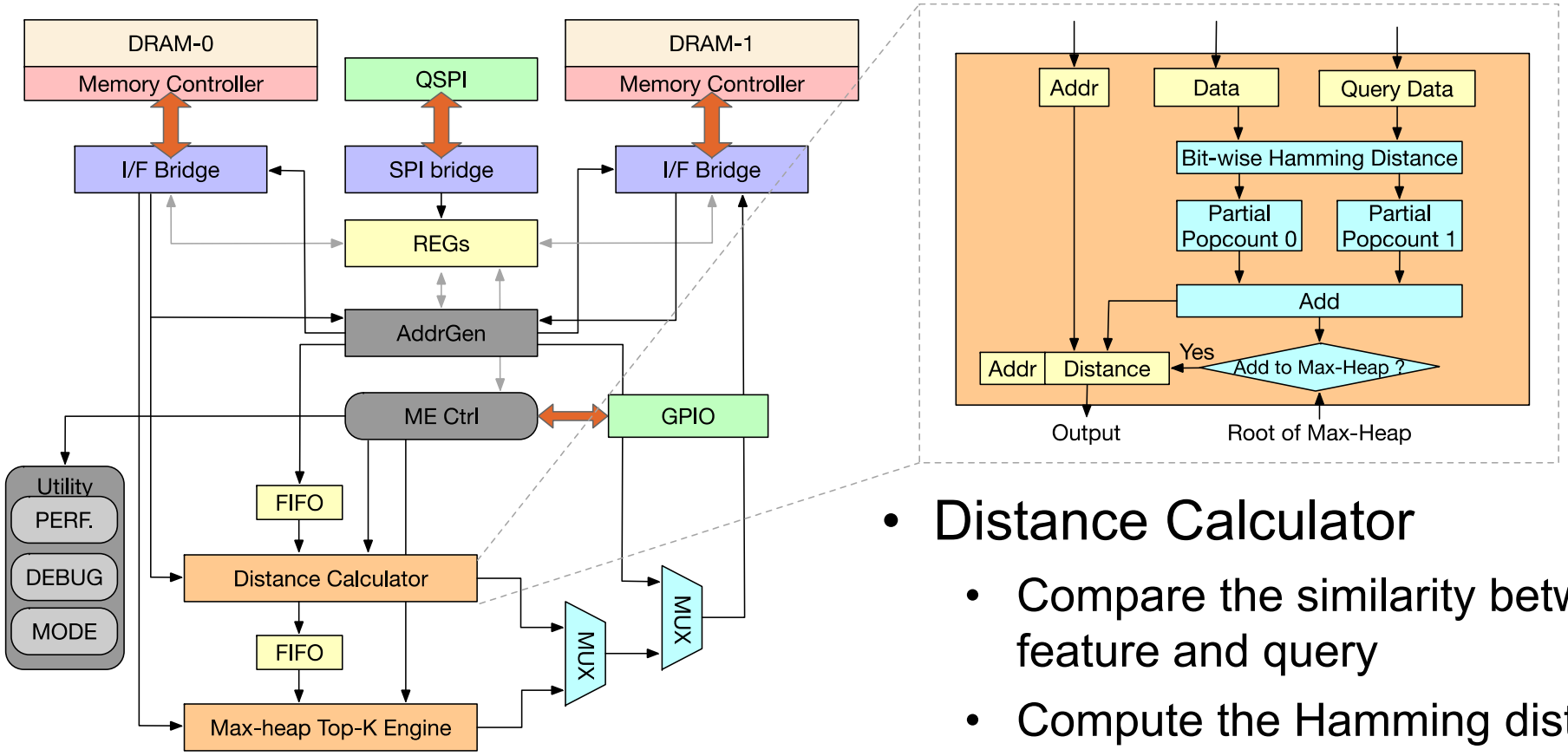
- Memory
  - 4 x 1Gb blocks with 4096 bits I/O
  - 38.4GB/s on-chip bandwidth per block
- Compute
  - Match Engine: Coarse-grained Matching
  - Neural Engine: Fine-grained Ranking
- Dual-mode Interface

Match Engine Architecture (1)



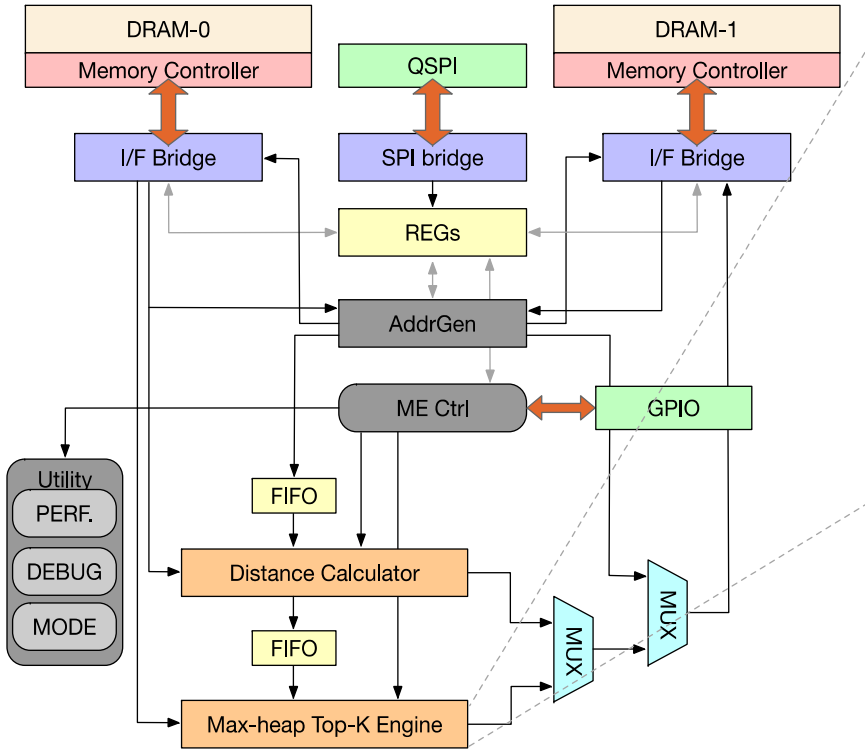
- Address Generator
  - Multi-mode to support different access patterns
  - Configurable via registers
  - Build-in performance evaluation mode & performance counter

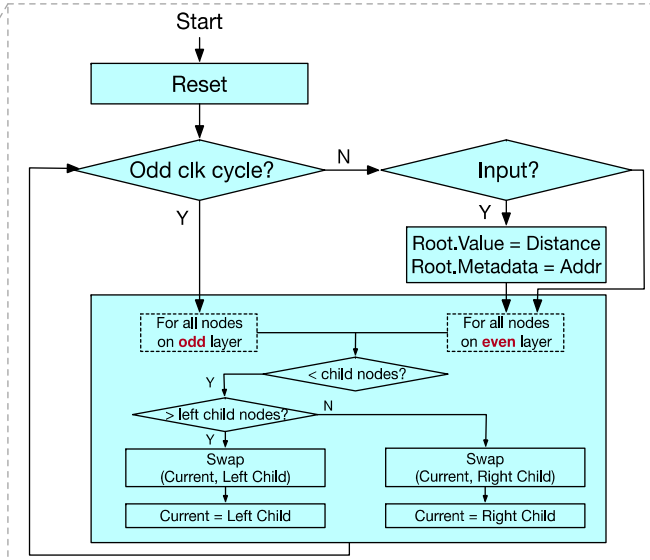
Match Engine Architecture (2)

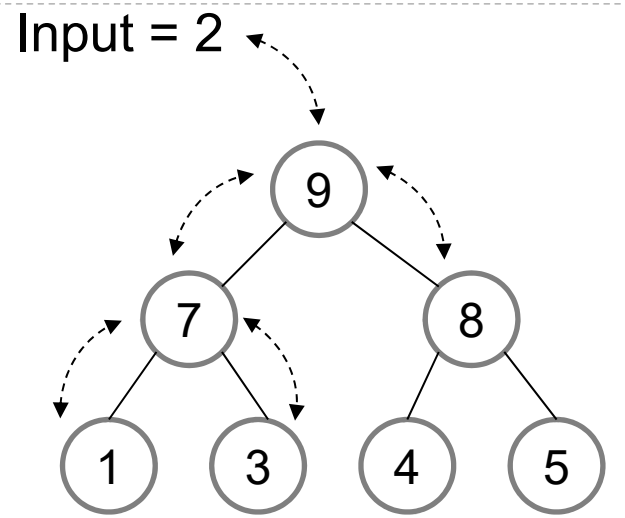


- Distance Calculator
  - Compare the similarity between input feature and query
  - Compute the Hamming distance of two 512-bit features
  - Filtered by rot of max-heap

Match Engine Architecture (3)

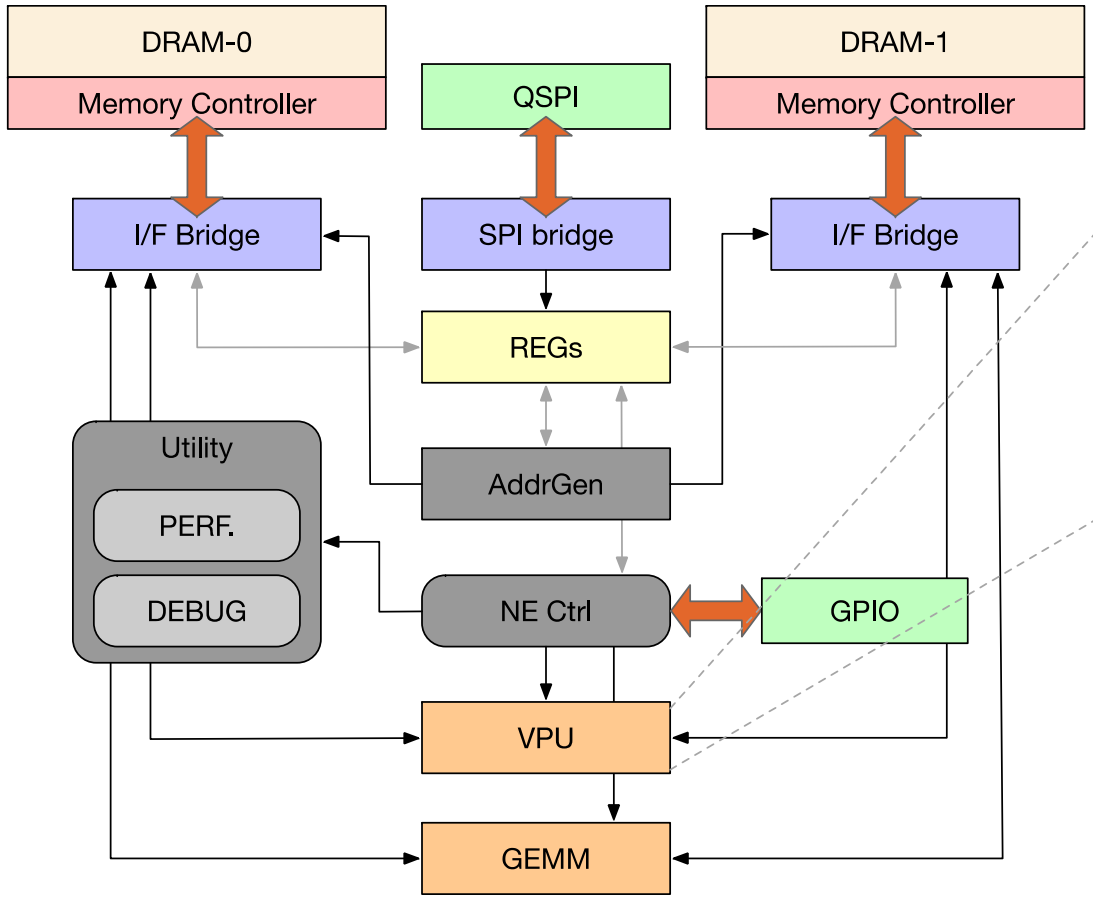


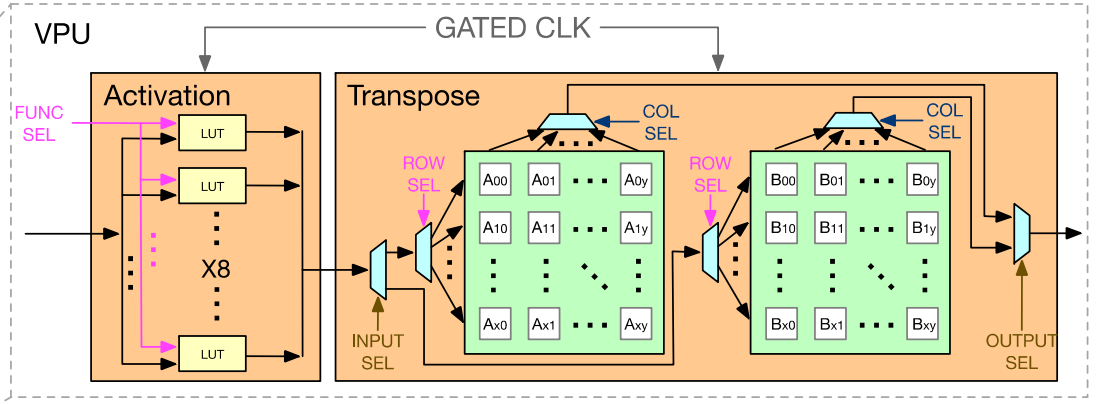




- Top-k Engine
  - Maintain a max-heap hardware block
  - Receives input every two cycles
  - Alternately heapifies nodes in odd layers and even layers
  - Stores the top-1000 matching results

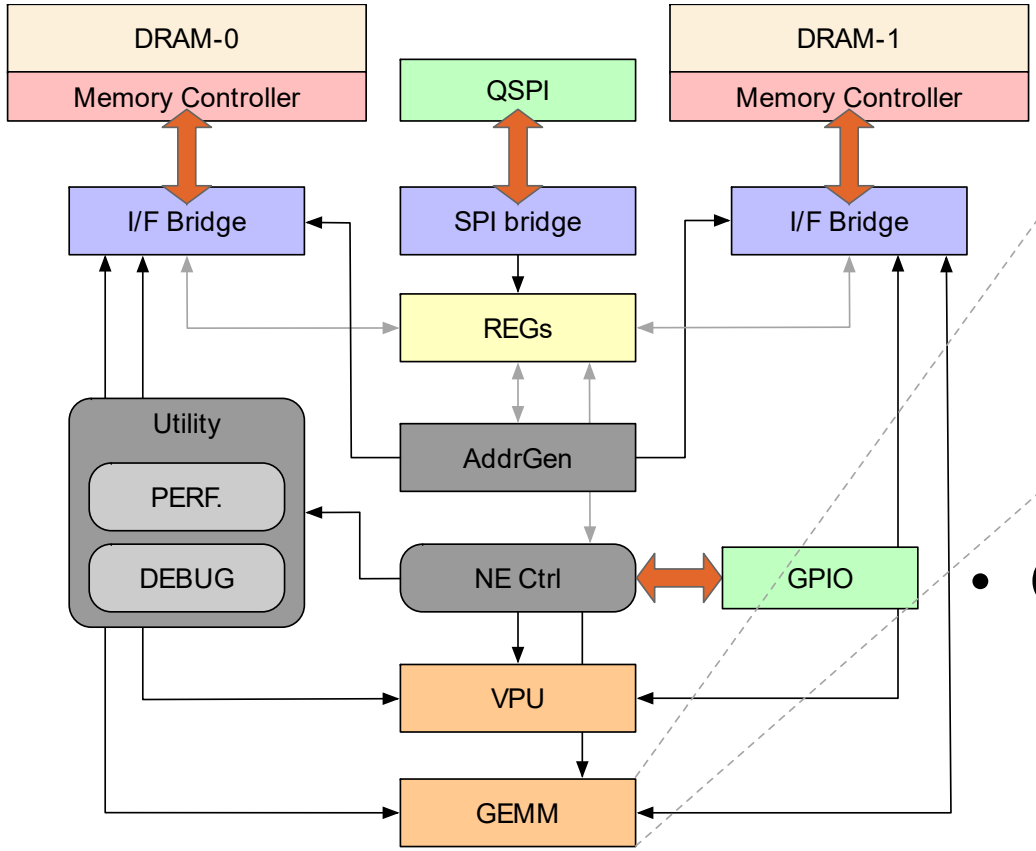
Neural Engine Architecture (1)

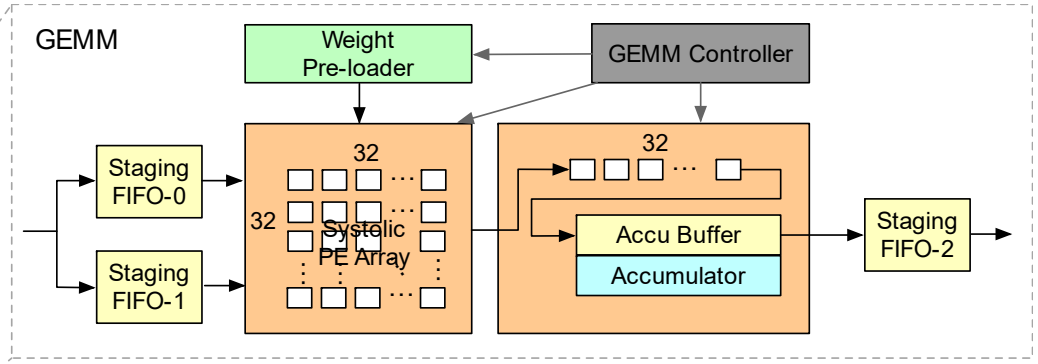




- Vector Process Unit
  - Activations
    - LUT based design
    - Supports GeLU & Exp
  - Transpose
    - Transpose 16x16 matrix with ping-pong array
    - Implemented with 2D register file array
    - Supports row-based writes and column-based reads

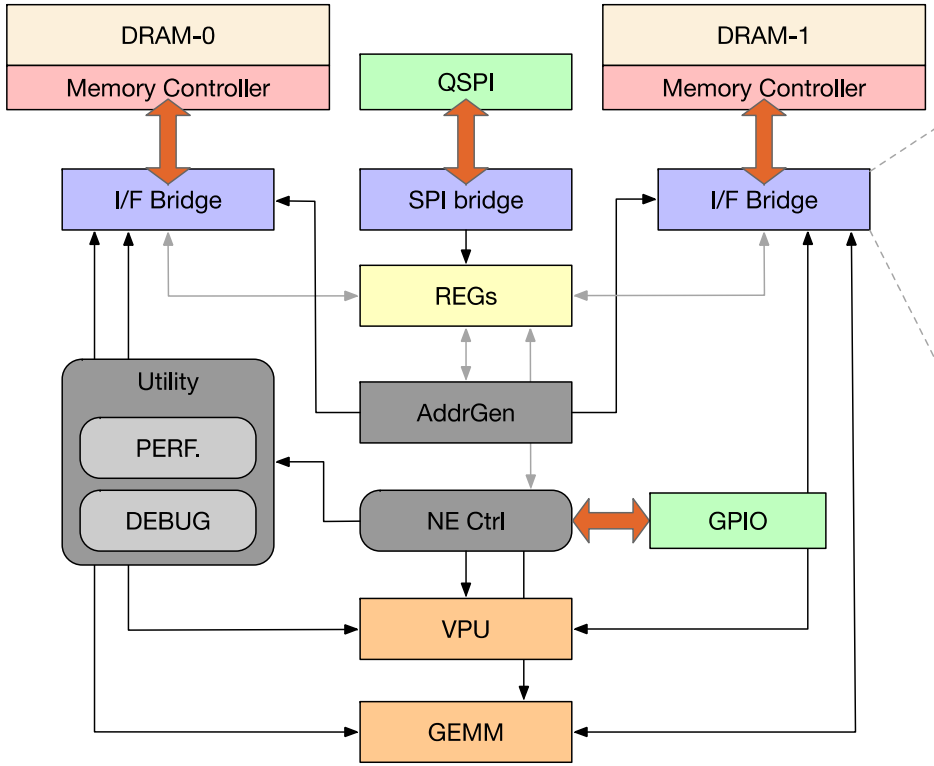
Neural Engine Architecture (2)

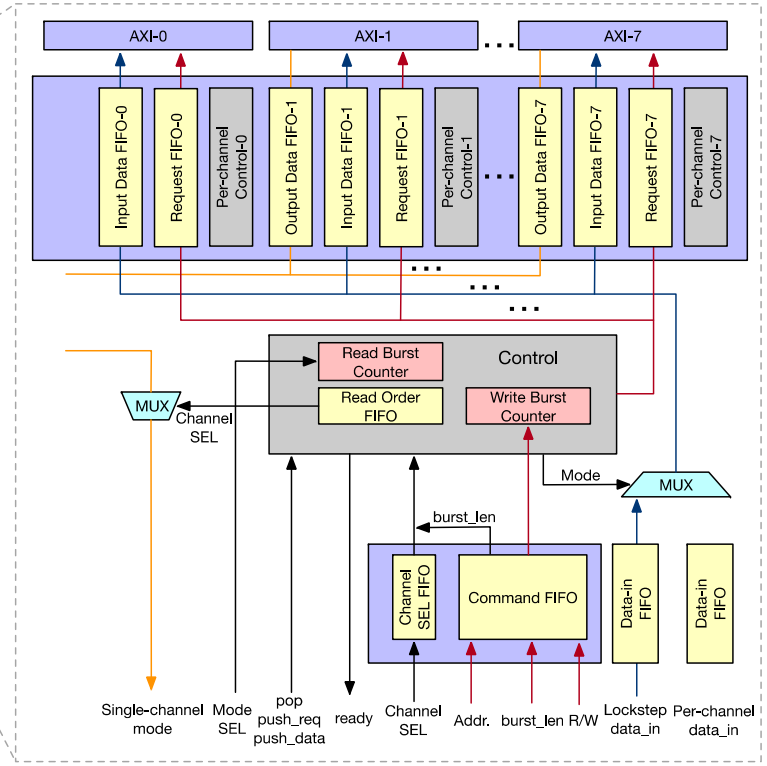




- GEMM
  - 32 by 32 systolic PE array (INT8)
  - Partial sum accumulated by the accumulator (INT32)
  - 600GOPS (@300MHz)

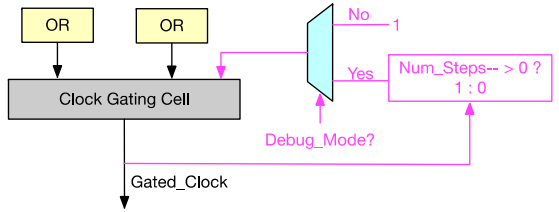
Interface Bridge & Debug





- Support both single-channel mode and lockstep-mode
- Read/write counter to support burst requests
- Support cycle-wise debug with clock gating

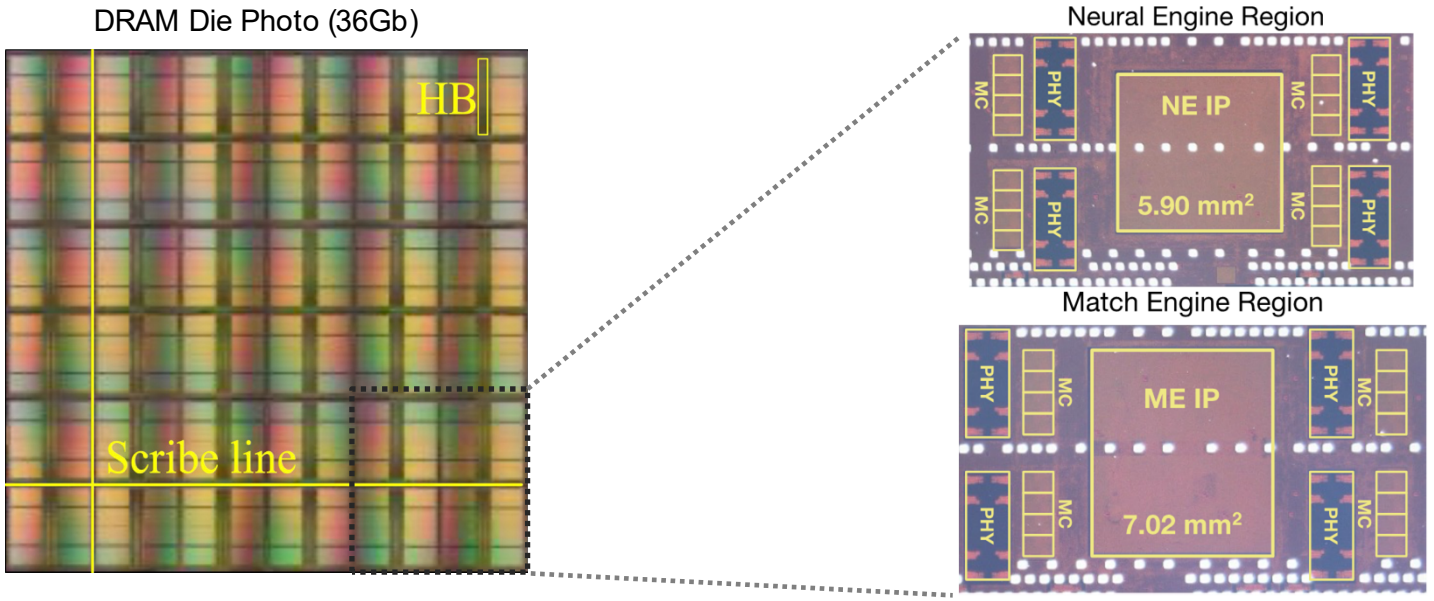
Lock-step DEBUG



Outline

- Motivation
- System and Chip Architecture
  - 3D Logic-to-DRAM Hybrid Bonding
  - PNM Engine for Recommendation System
- **Measurement Results**
- Conclusion

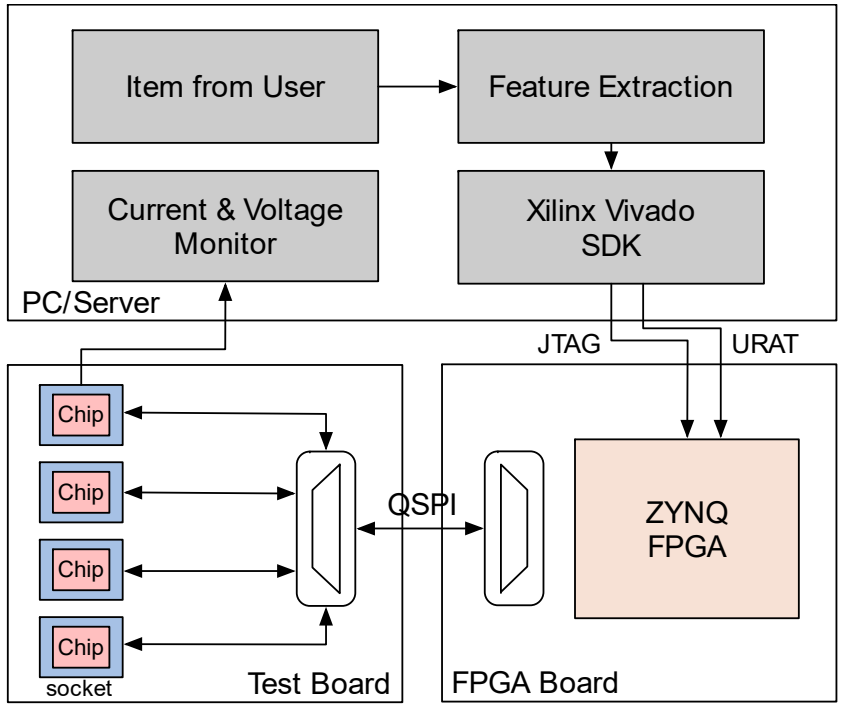
Die Photo and Summary

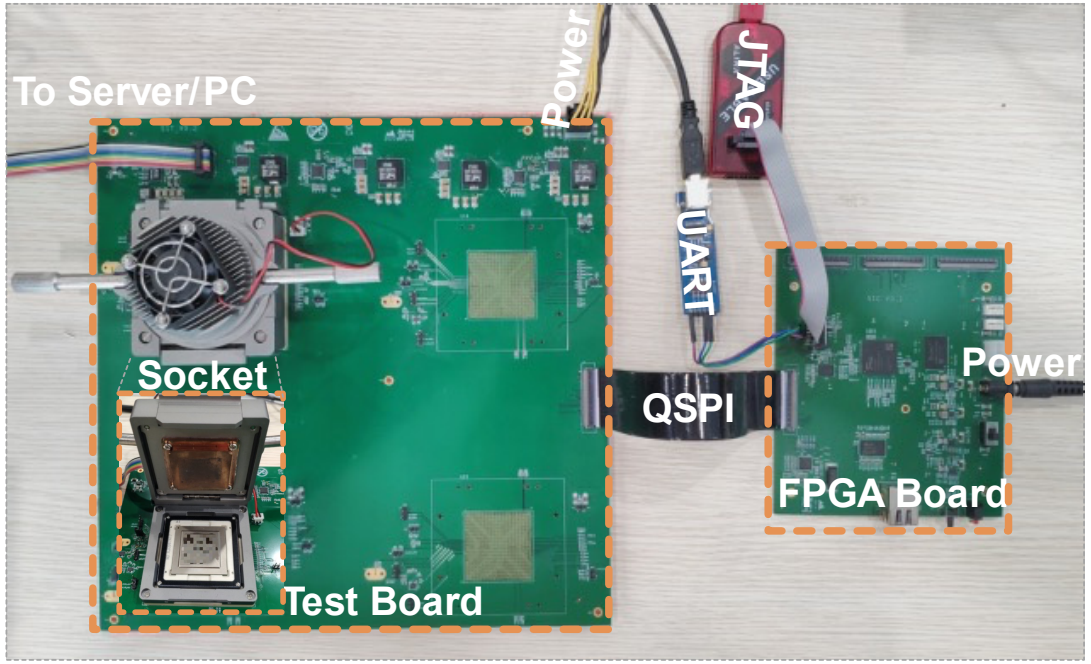


DRAM Die		
Technology	25nm	
Area	Total*	602.22 mm²
	Neural Engine	32 mm²
	Match Engine	32 mm²
Voltage	1.1 V	
Frequency (max)	150 MHz	
Power	300 mW per 1Gb	
Bandwidth**	153.60 GB/s / 1.38 TB/s	

Logic Die		
Technology	55nm	
Area	Total*	602.22 mm²
	Neural Engine	5.90 mm²
	Match Engine	7.02 mm²
# of MC	16 per IP	
Voltage	1.2 V	
Frequency	300 MHz	
Power	977.70 mW	
Precision	INT8	

Evaluation Platform





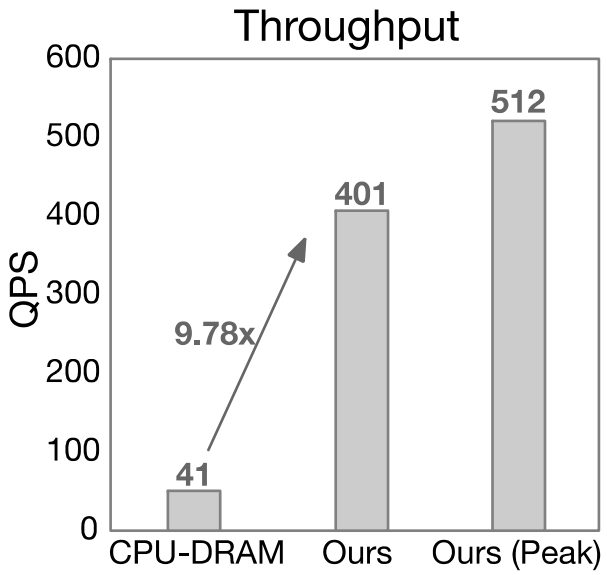
- Test board capable to mount up to 4 HB
- FPGA board responsible to write/read data and generate configuration to the chip register

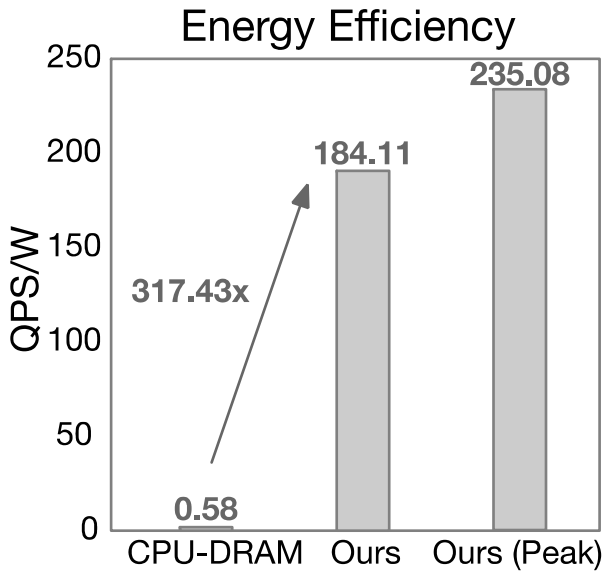
Performance

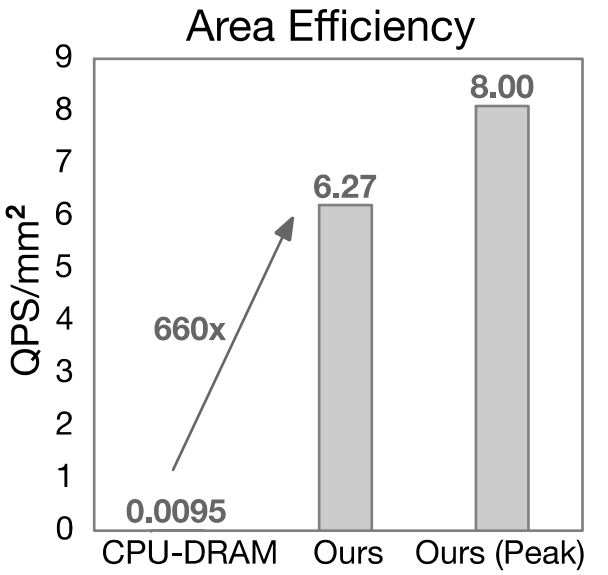
	CPU - DRAM*	This work
Logic Technology	14 nm	55 nm
Frequency	2.20 GHz	300 MHz
Area	4294 mm^2	64 mm^2 (4Gb)
Precision	INT8	INT8
Power**	70.17 W (TDP: 125 W)	2.178 W

** CPU: Intel Xeon Gold 5220@2.20GHz, tested on Pytorch*
*** CPU power measured by PyRAPL*

- Measurement vs. Peak: **~20%** initialization and memory subsystem overhead







QPS: Queries per second

Comparison

	2D CIM *	2D PNM **	2.5D PNM ***	3D TSV (Hybrid) ****	This work
Type of Memory	SRAM	DDR4	HBM2	HBM2	LPDDR4
Technology (Memory/Logic)	16nm	2xnm / 2xnm	1y# / 7nm	20nm / 20nm	25nm / 55nm
Capacity	4.5 Mb	8GB / DIMM	80GB	6GB / cube	4.5GB
Bandwidth	-	128GB/s / DIMM	1935GB/s	1200GB/s / cube#	38.4GB/s / 1Gb
Frequency (Logic)	200MHz	500MHz	1410MHz	300MHz	300MHz
Bandwidth/Capacity (a.u.)	-	16	24.2	200	**307**
Energy	-	~25pJ/bit	4.47pJ/bit	2.75pJ/bit	**0.88pJ/bit**

#Estimated

- High off-chip bandwidth
- High bandwidth per capacity
- Low energy per bit

* H. Jia et al, ISSCC 2021.
** F. Devaux et al, Hotchip 2019
*** J. Choquette et al, Hotchip 2020
**** Y. C. Kwon et al, ISSCC 2021

Outline

- Motivation
- System and Chip Architecture
  - 3D Logic-to-DRAM Hybrid Bonding
  - PNM Engine for Recommendation System
- Measurement Results
- Comparison
- **Conclusion**

Conclusion

- Memory-bound application can significantly benefit from process-near-memory and computing-in-memory
- A 3D Logic-to-DRAM Hybrid Bonding Chip with Process-Near-Memory Engine for Recommendation System is demonstrated featuring with:
  - High-bandwidth and energy-efficient memory with hybrid bonding
  - High-throughput streaming processing units for matching and ranking
  - **2.4GB/s/mm^2** bandwidth density and **0.88pJ/bit** energy consumption
  - **~10x** performance improvement and over **300x** energy-efficiency improvement over conventional CPU+DRAM system

A 28nm 27.5TOPS/W Approximate-Computing-Based Transformer Processor with Asymptotic Sparsity Speculating and Out-of-Order Computing

Yang Wang[1], Yubin Qin[1], Dazheng Deng[1], Jingchuan Wei[1], Yang Zhou[1], Yuanqi Fan[1], Tianbao Chen[2], Hao Sun[1], Leibo Liu[1], Shaojun Wei[1], Shouyi Yin[1*]

[1] Tsinghua University, Beijing

[2] TsingMicro Tech

Self Introduction

- **Yang Wang** received the B.S. degree from Xidian University, China, 2014.
- Received the Ph.D. degree from Institute of Microelectronics, Chinese Academy of Sciences, China, 2019.
- He is currently a postdoctoral researcher in Tsinghua University, China.

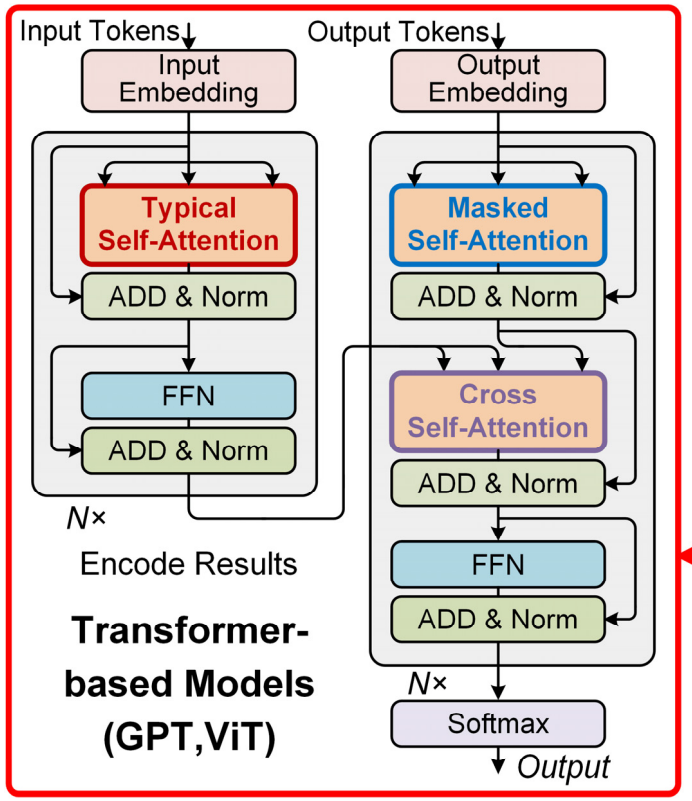
- **Research Interests:** Computer architecture and VLSI DSP design, efficient DNN inference and training processor, hardware-software system co-design, and neuromorphic algorithms acceleration.

Outline

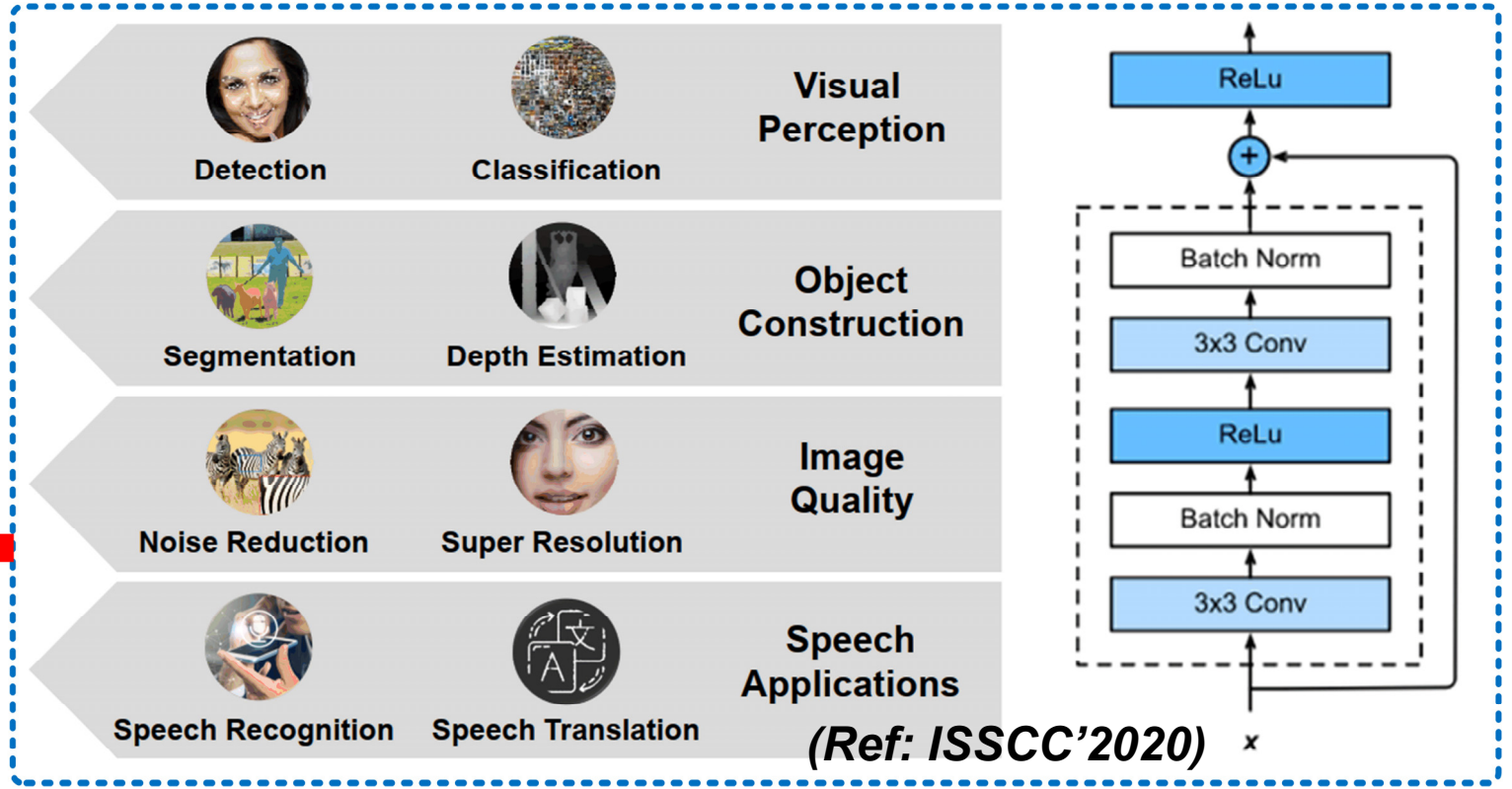
- **Background and Motivation**
- Challenges of Global-attention-based Transformer Processor
- Proposed Transformer Processor
  - Big-exact-small-approximate Processing Element
  - Bidirectional Asymptotic Sparsity Speculation
  - Out-of-order PE-line Computing
- Measurement and Comparison
- Conclusion

Transformer-based Models

■ Versatile for Multiple AI Tasks from NLP to CV



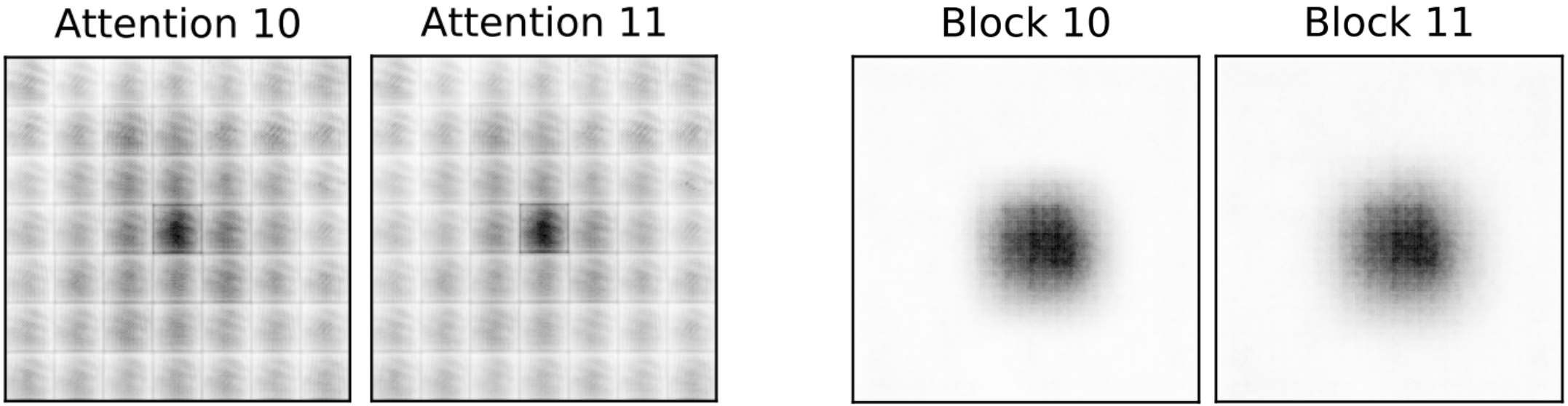
Moving to



Transformer-based models surpass conventional DNN comprehensively.

Global Attention Mechanism

- **Global-level Receptive Field with Attention Mechanism**



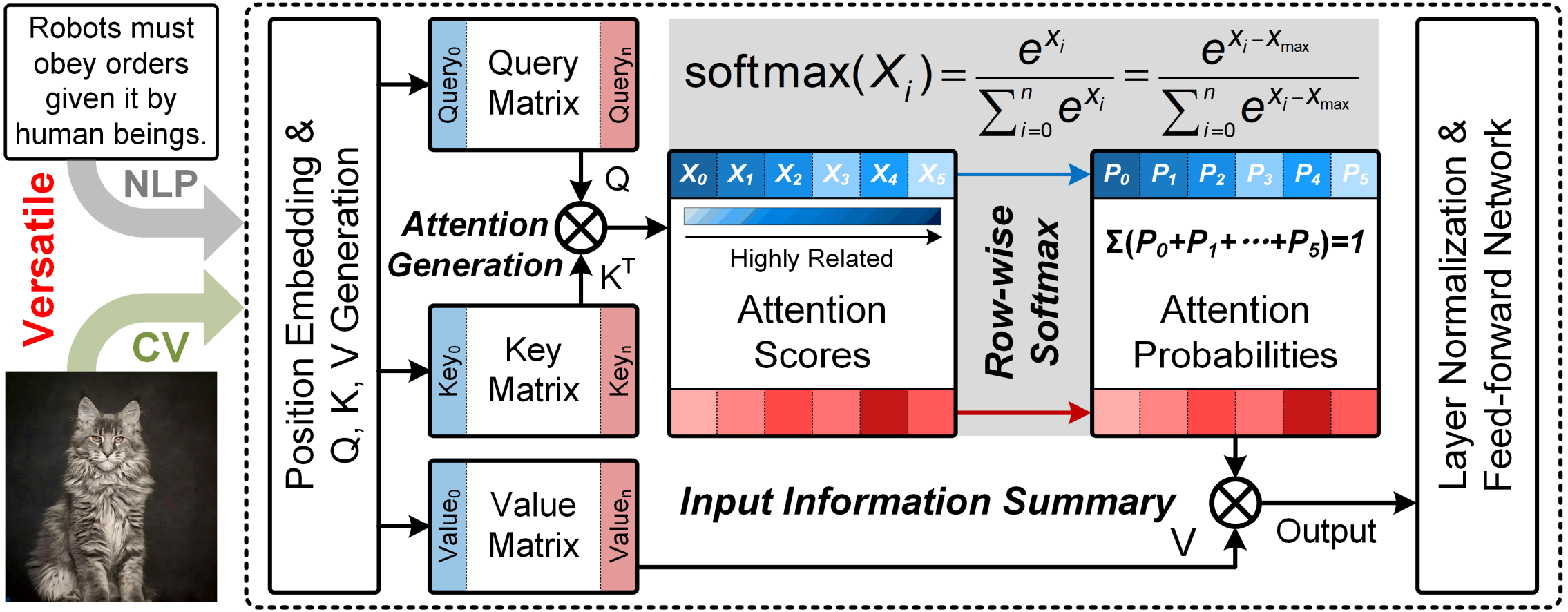
ViT-B/32 Attention Blocks **ResNet50 Convolution Blocks**

(Google Research, arXiv'2021)

Global receptive field achieved with attention mechanism is the key for the superior performance of Transformer.

Global Attention Mechanism

Computation Details of Attention Block



High Performance vs. Huge Computation

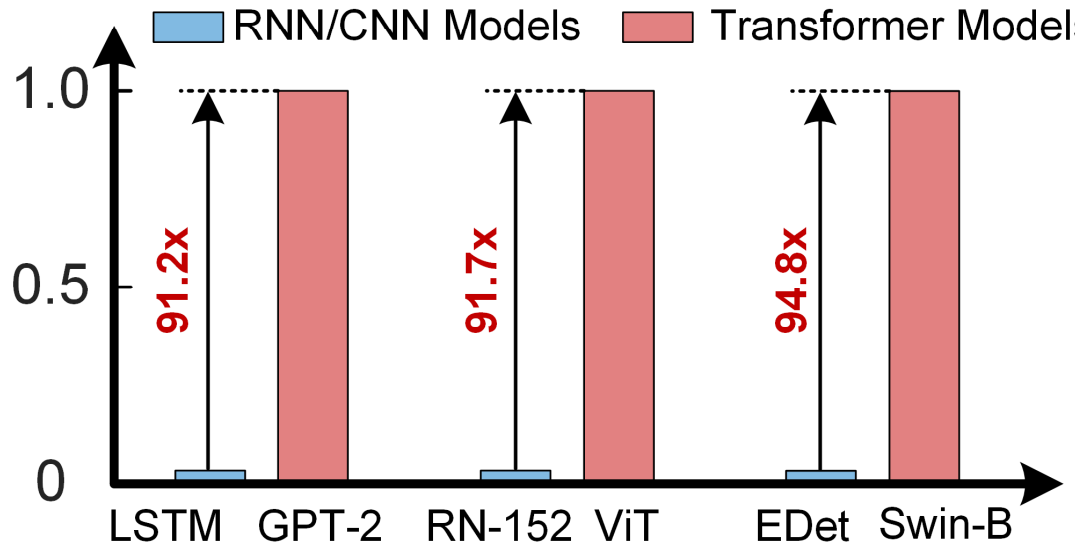
■ High Performance Comes at Increased Computation

Performance Comparison

Tasks	RNN/CNN Models	Transformer Models
Language Modelling (Penn Treebank)	44.9 (LSTM)	**35.76 (GPT-2)[1]**
Classification (ImageNet)	78.57% (ResNet-152)	**90.45% (ViT-G/14)[2]**
Object Detection (COCO 2017)	43.9 (EfficientDet)	**56.4 (Swin-B)[3]**

☺ **Up to 20.1% accuracy improvement**

Operations Comparison



☹ **Nearly 100x more computation**

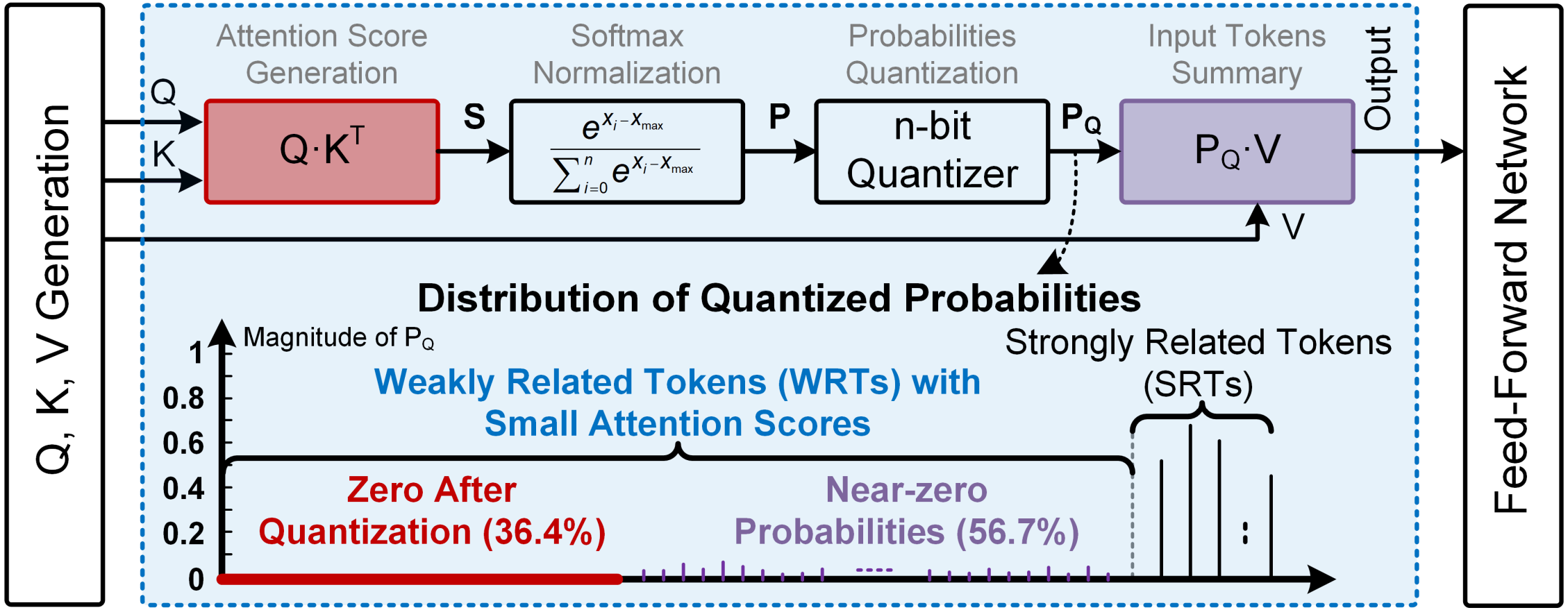
References: [1] Language Models are Unsupervised Multitask Learners.
[2] An image is worth 16x16 words: Transformers for image recognition at scale.
[3] Swin Transformer: Hierarchical Vision Transformer using Shifted Windows.

Outline

- Background and Motivation
- **Challenges of Global-attention-based Transformer Processor**
- Proposed Transformer Processor
  - Big-exact-small-approximate Processing Element
  - Bidirectional Asymptotic Sparsity Speculation
  - Out-of-order PE-line Computing
- Measurement and Comparison
- Conclusion

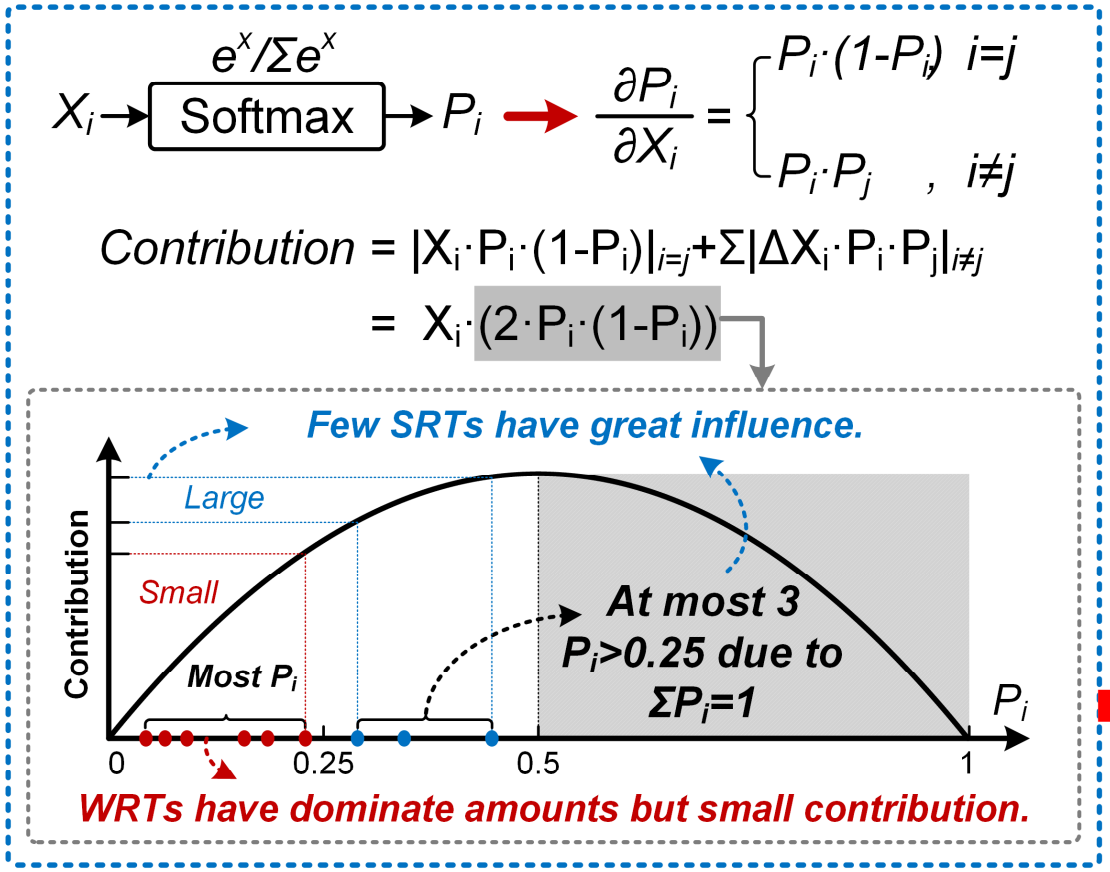
Challenges of Global Attention

■ Weak-Related Tokens Take Dominated Amounts

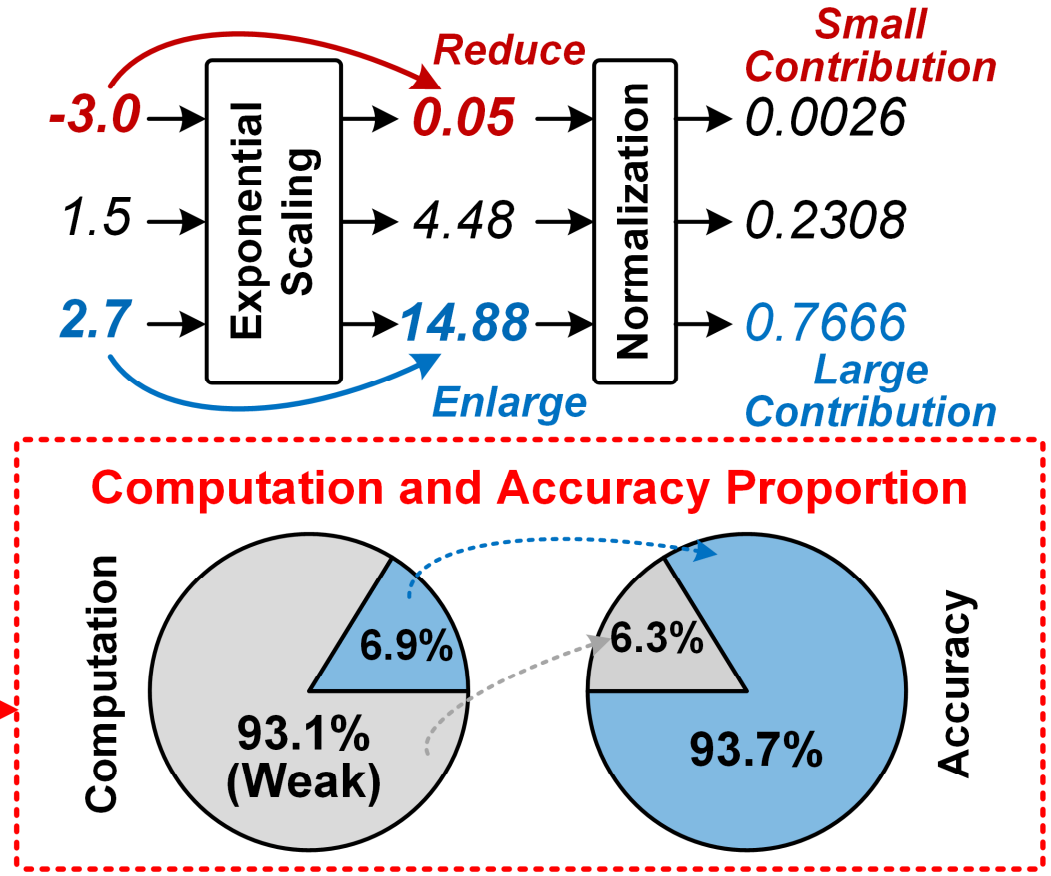


Challenge 1: Imbalanced Contribution

Large Energy Cost but Limited Contribution of WR-Tokens

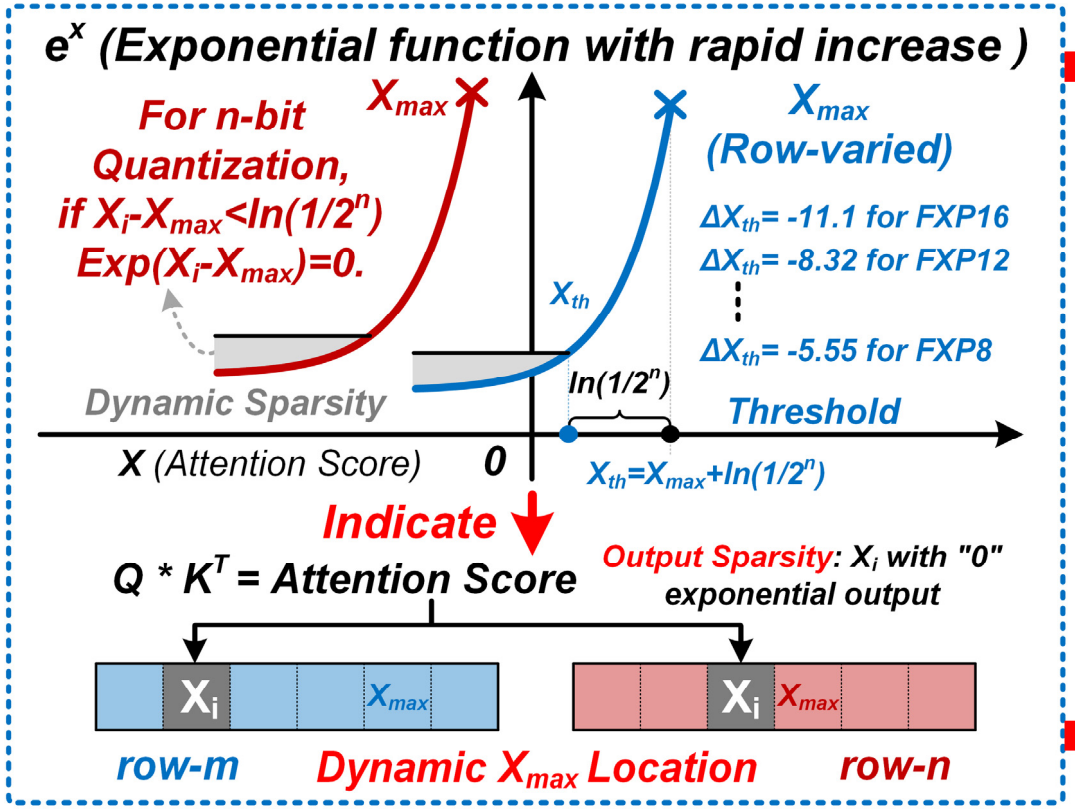


Imbalance Contribution



Challenge 2: Dynamic Redundancy

Redundant Computation Due to Dynamic Output Sparsity



Dynamic Redundancy

Sparsity with TWO degrees of freedom

Element-wise variable X_i Row-wise variable X_{max}

Sparsity determined by X_i-X_{max}

Increased Difficulty of Sparsity Speculation

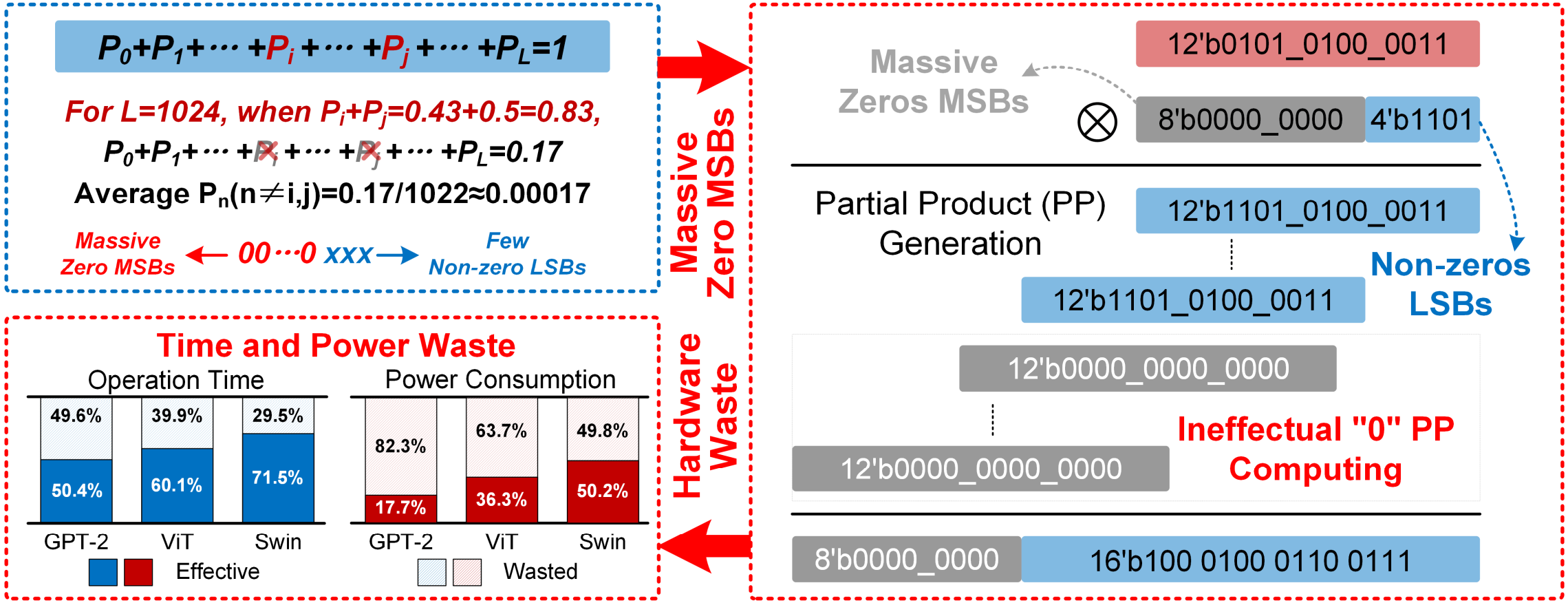
Energy of Redundant Computation

Models	Attention Energy		Redun. Ratio
	Total	Redundant	
GPT-2	80.9mJ	29.4mJ	***36.4%***
ViT	44.2mJ	16.1mJ	***36.5%***
Swin	13.9mJ	4.18mJ	***30.1%***

Dynamic sparsity has two degrees of freedom that is hard to speculate.

Challenge 3: Hardware Underutilization

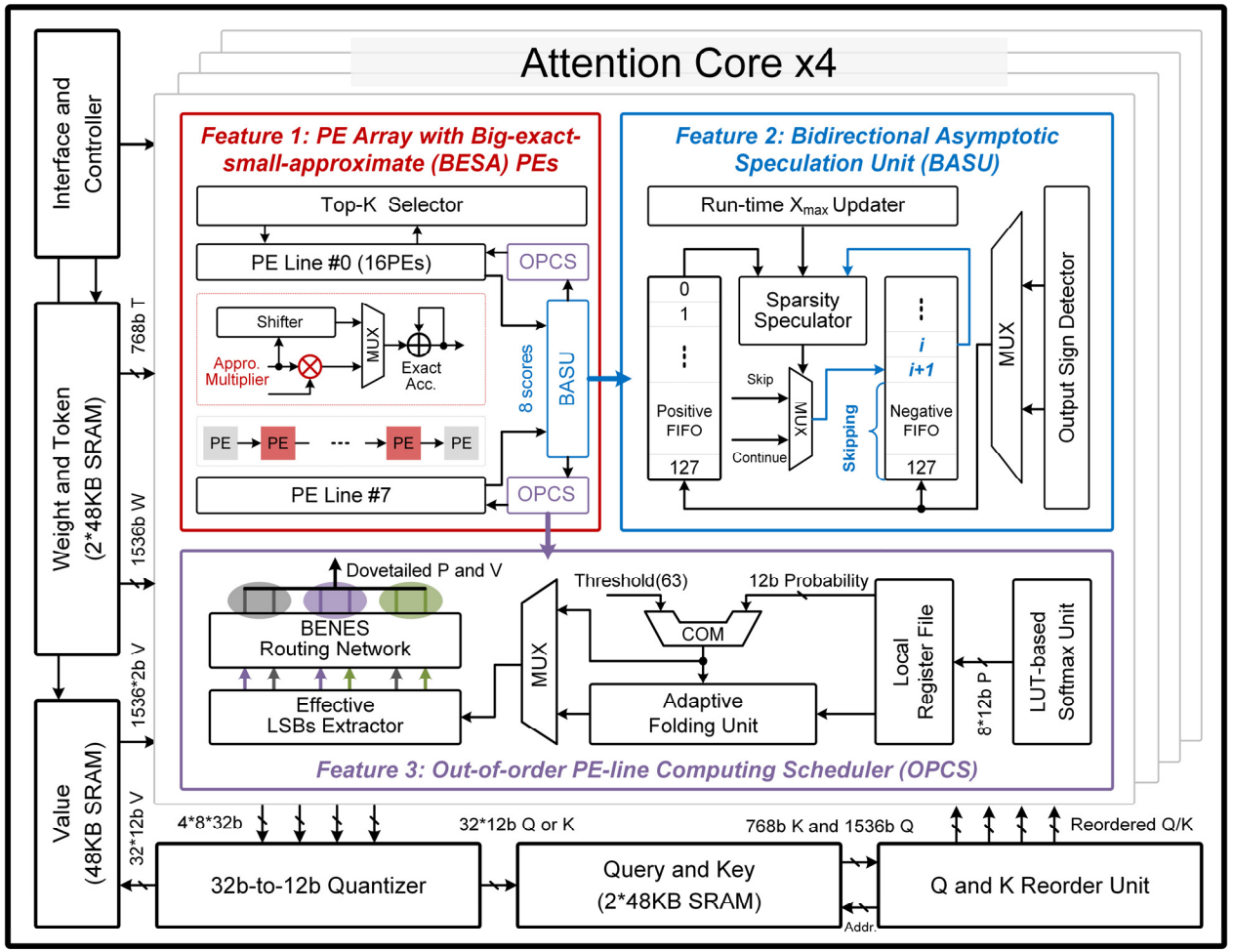
Hardware Underutilization Due to Near-zero Probabilities



Outline

- Background and Motivation
- Challenges of Global-attention-based Transformer Processor
- **Proposed Transformer Processor**
  - **Big-exact-small-approximate Processing Element**
  - **Bidirectional Asymptotic Sparsity Speculation**
  - **Out-of-order PE-line Computing**
- Measurement and Comparison
- Conclusion

Overall Architecture



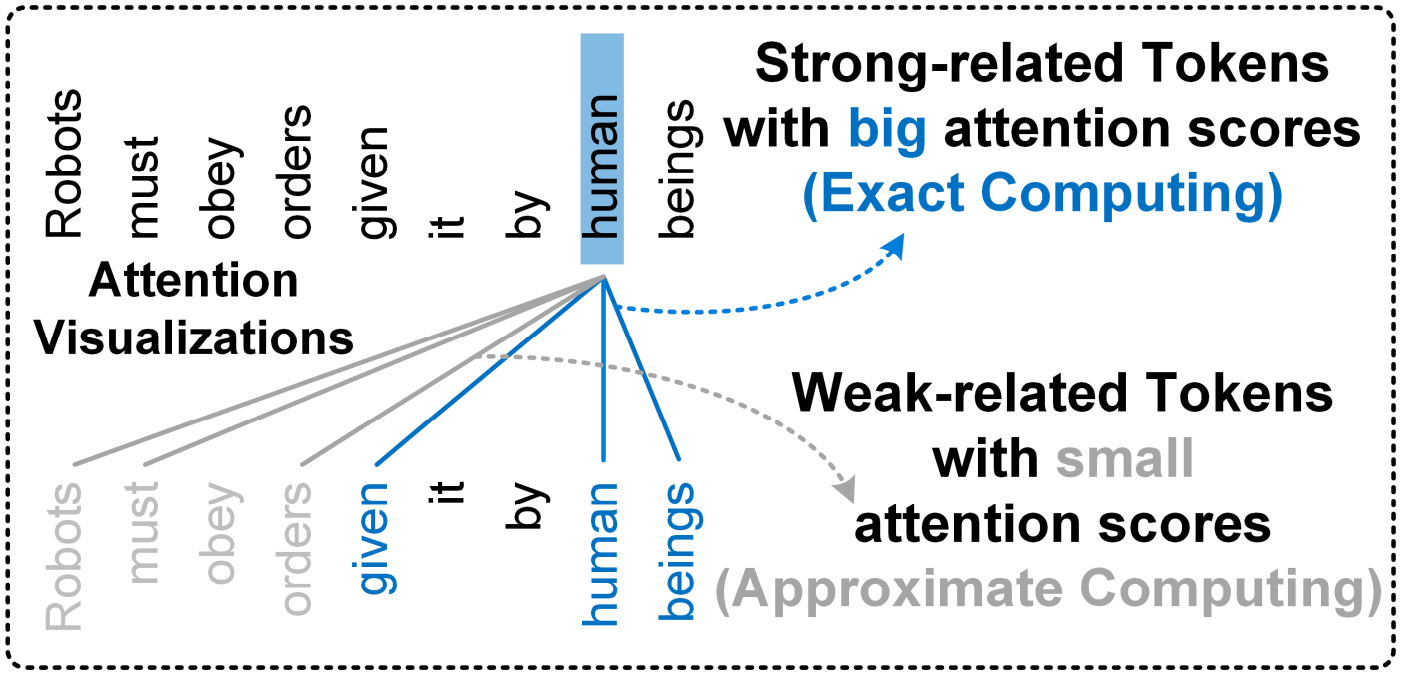
- **BESA PE** computes WR-Tokens approximately to reduce energy cost of entire attention block
- **BASU** speculates output sparsity to remove redundant attention computation in $Q \times K^T$
- **OPCS** dovetails multiplications to skip zero partial products computing in $P \times V$

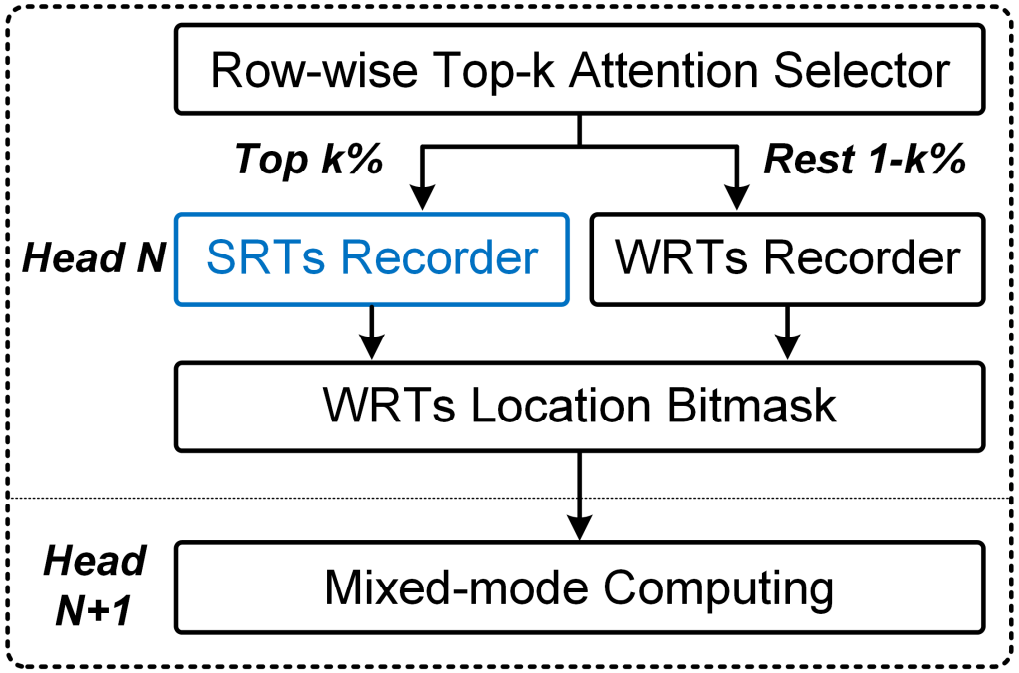
Outline

- Background and Motivation
- Challenges of Global-attention-based Transformer Processor
- **Proposed Transformer Processor**
  - **Big-exact-small-approximate Processing Element**
  - Bidirectional Asymptotic Sparsity Speculation
  - Out-of-order PE-line Computing
- Measurement and Comparison
- Conclusion

Feature 1: Big-exact-small-approximate PE

■ Approximate Computing for Energy Reduction of WR-Tokens



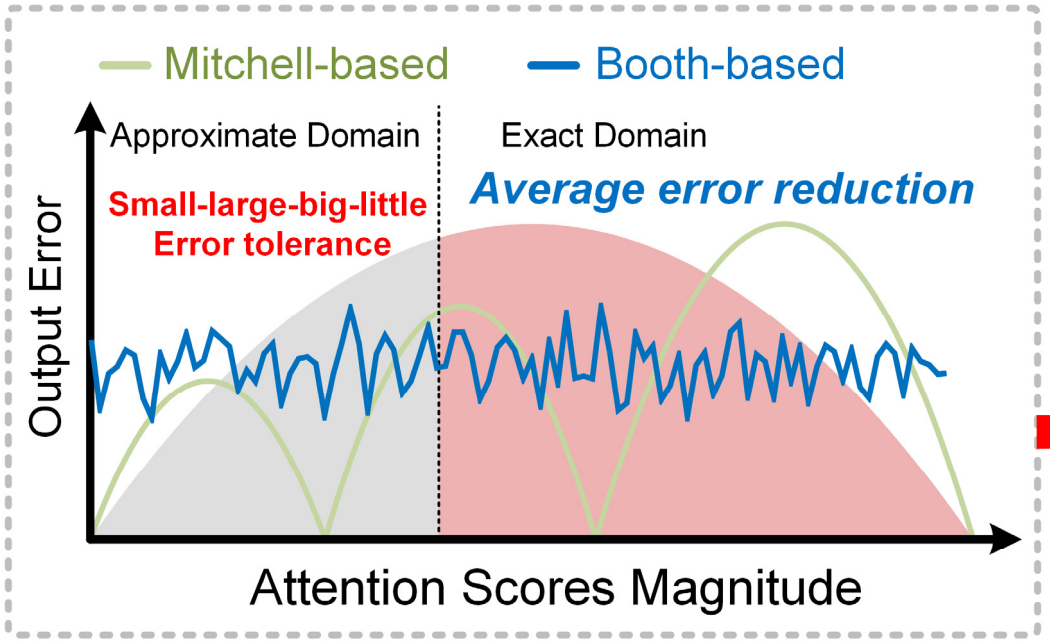


■ Adjustable Top-K-based approximate threshold determination.

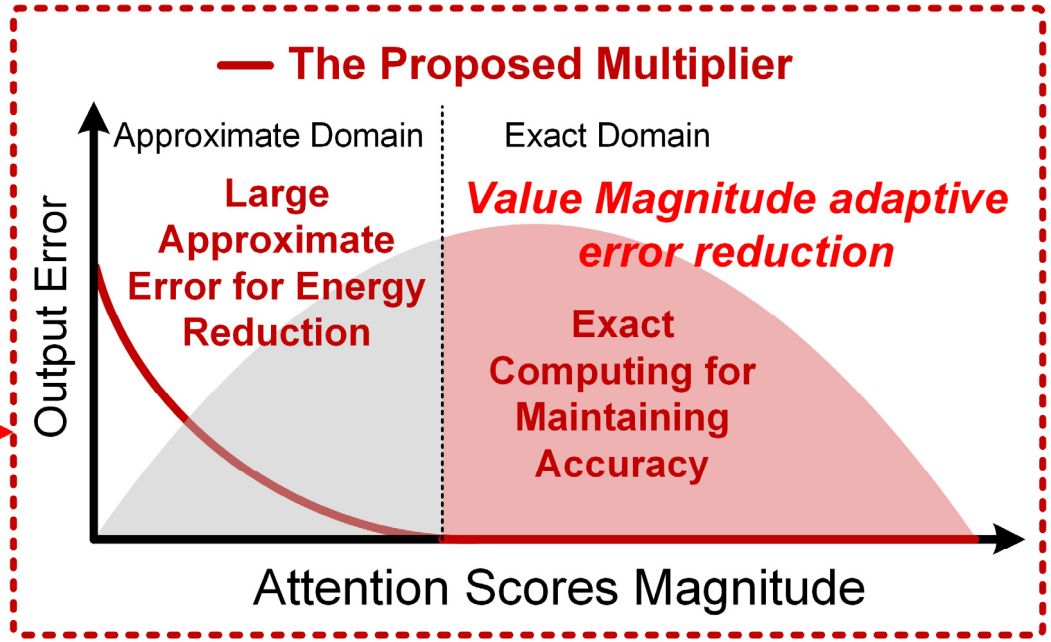
■ Intra-head approximate attention scores location updating.

Feature 1: Big-exact-small-approximate PE

■ Error Tolerance Mismatch of Previous Approximate Methods



Solution



☹ Limited Energy Reduction

☹ Accuracy Degradation

☺ More Energy Reduction

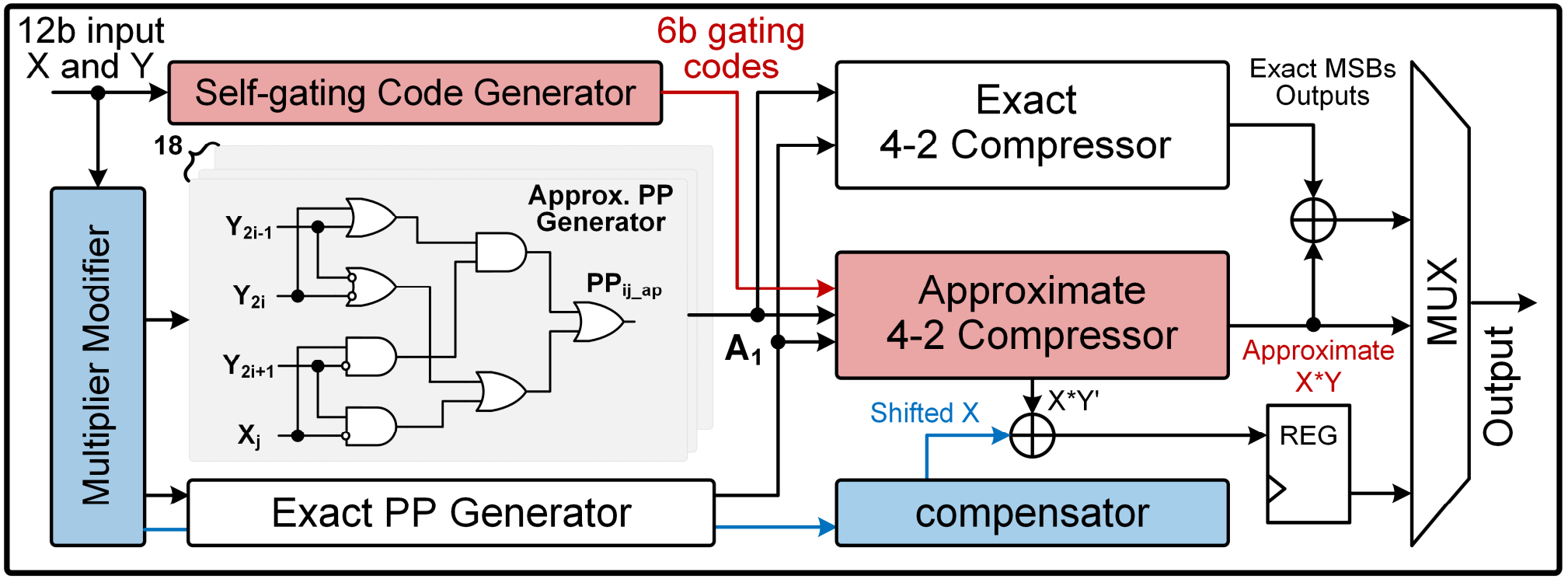
☺ Maintain High Accuracy

Previous methods focus on average error instead of value-adaptive error.

Feature 1: Big-exact-small-approximate PE

- **Proposed BESA PE Achieved with Approximate Multiplier**

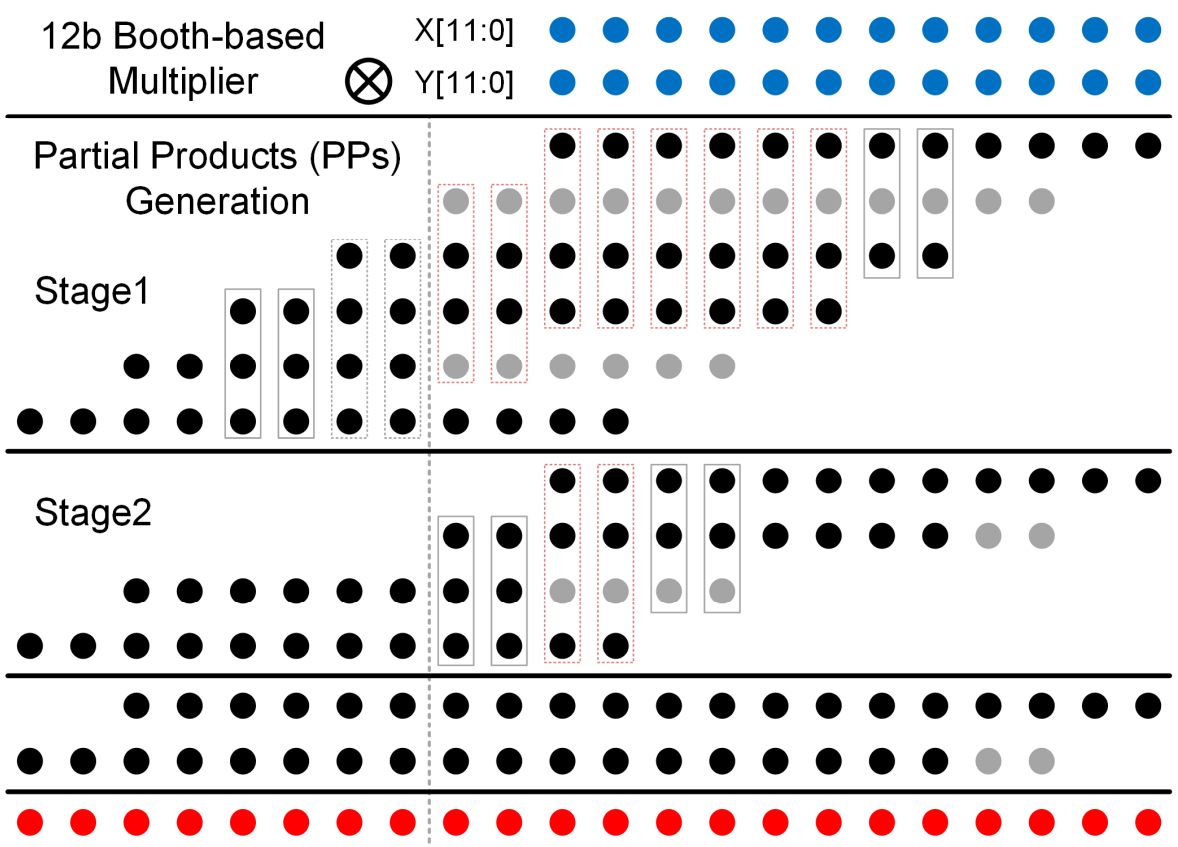
***Value magnitude adaptive approximation* —> Self-gating circuits.**

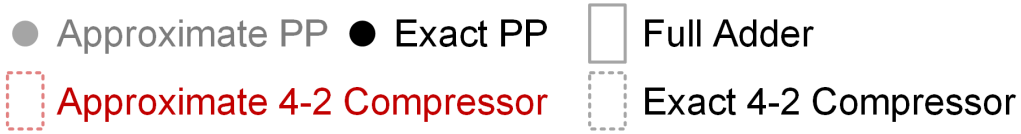


***Exact computing with approximate logic* —> Adaptive modification.**

Feature 1: Big-exact-small-approximate PE

Detailed Diagram of The Proposed Approximate Multiplier





- There are total 6 partial products rows with 24 columns.
- Approximate partial products generators are used in row-2 and row-5.
- Approximate 4-2 compressors are used in columns 9-to-16.

Feature 1: Big-exact-small-approximate PE

Approximate Computing Components

Truth Table of PP_{ij}

Coding Bits			Booth Value(BV)	PP_{ij} results for X_{ij}			
Y_{2i-1}	Y_{2i}	Y_{2i-1}		00	01	10	11
	⋮		⋮				
0	1	0	1	0	0	1	1
0	1	1	+2	0	1→0	0→1	1
1	0	0	-2	1	0→1	1→0	0
1	0	1	-1	1	1	0	0
	⋮		⋮				

NO Approximate ERROR for BV=0 or ±1.

$$PP_{ij\_ap}=(Y_{2i}+Y_{2i-1})\cdot Y'_{2i+1}\cdot X_j+(Y'_{2i}+Y'_{2i-1})\cdot Y_{2i+1}\cdot X'_j$$

Truth Table of 4-2 Compressor

Compressor inputs				Exact Output			Approximate Output		
A_1	A_2	A_3	A_4	C_{out}	Sum	Carry	C_{out}	Sum	Carry
0	0	0	0	0	0	0	0	0	0
	⋮	⋮							
0	1	1	1	1	1	0	1	1	0
1	0	0	0	0	1	0	1	0	0
	⋮	⋮							
1	1	1	1	1	1	1	1	1	1

NO Approximate ERROR for A_1=0.

$$C_{out} = A_1+A_2\cdot A_3 \quad Sum = A_3+A_4 \quad Carry = (A_2 \odot A_3)'\cdot A_4$$

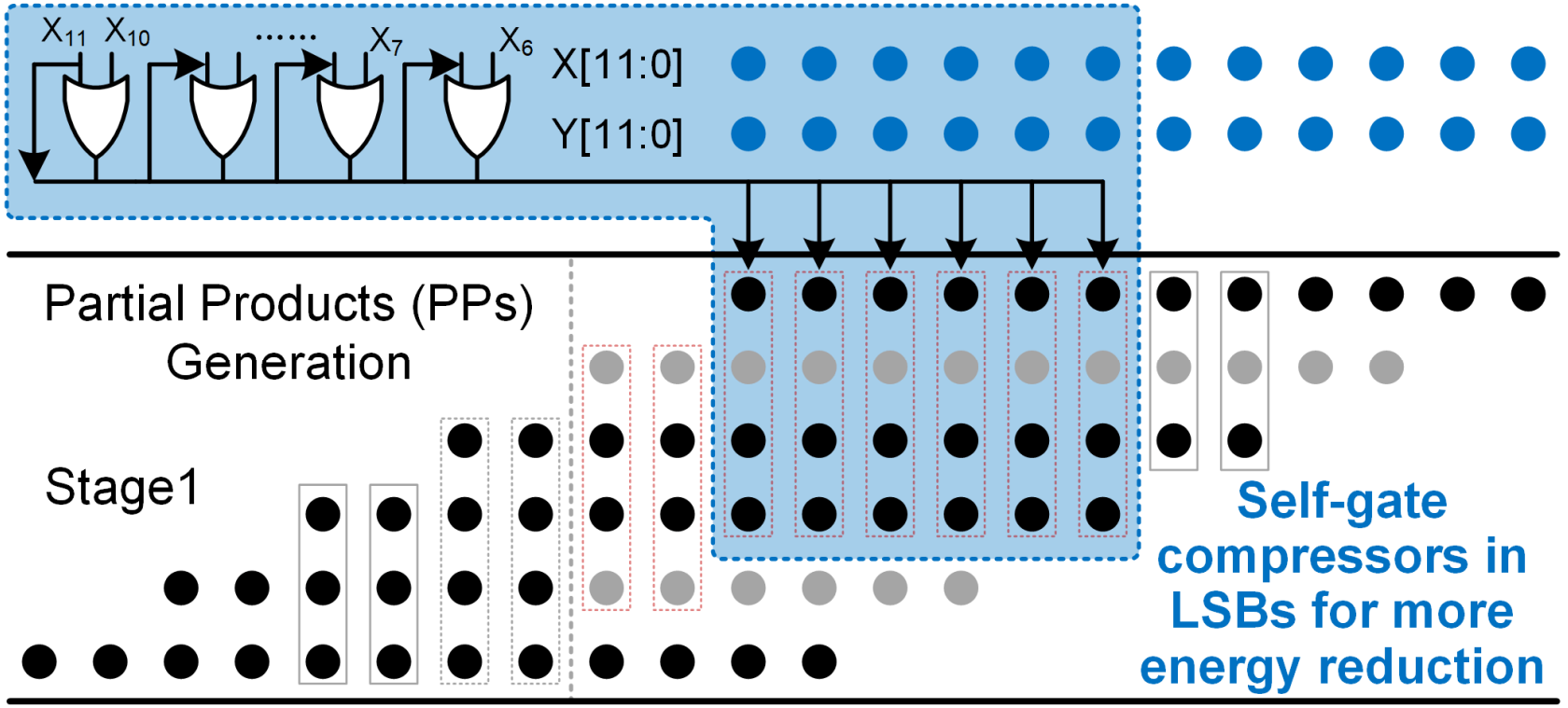
- Approximate components have **regular exact space**.
- Approximate components reduce **35.6%** area and **47.3%** power.

Feature 1: Big-exact-small-approximate PE

■ Approximate Computing with Self-gating Mechanism

Cascaded OR (AND) operation with 6-bit MSBs of the positive (negative) operands.

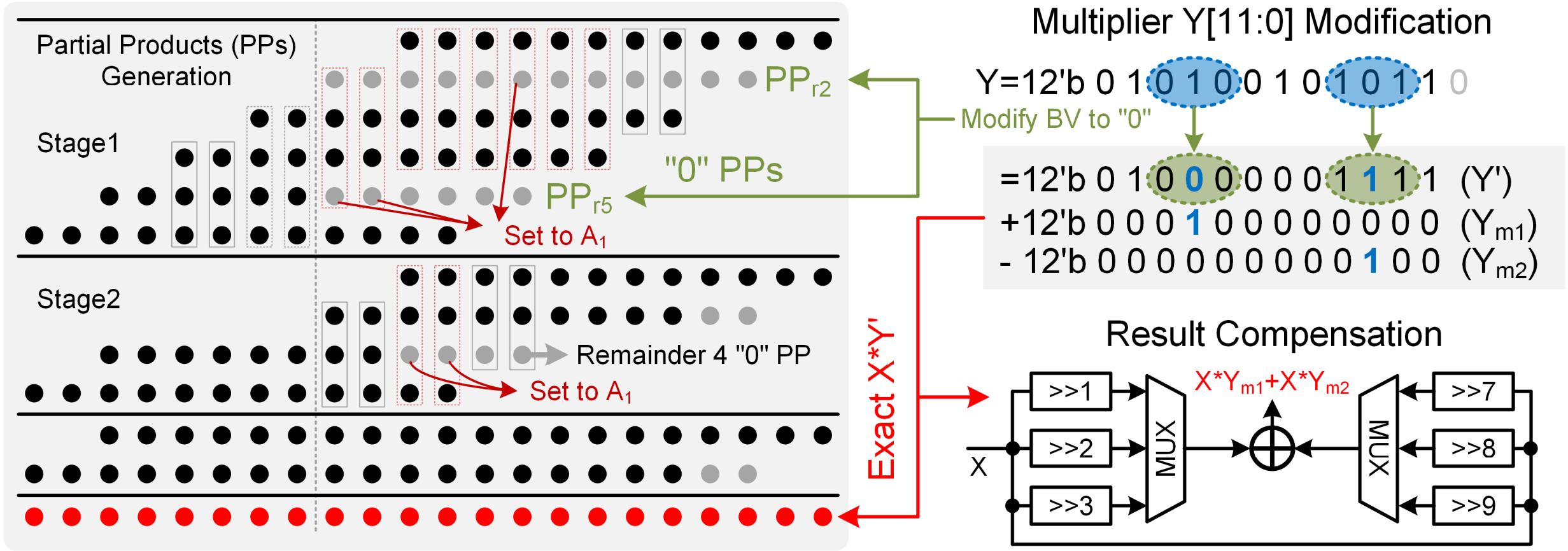
Small values result in more "0" bits used to gate compressors in LSBs parts.



Increase error but reduce more energy for small values.

Feature 1: Big-exact-small-approximate PE

■ Exact Computing with Modification and Compensation



Feature 1: Big-exact-small-approximate PE

Performance Improvement with BESA PE

Methods	Accuracy Loss	Energy Reduction
Mtichell	6.51%	21.6%
Booth	4.85%	15.3%
Ours	**0.43%**	**40.8%**

Models	Computation Energy		Energy Efficiency Improvement
	W/O Approx.	W/ Approx.	
GPT-2	1.41uJ	**0.83uJ**	**1.69x**
ViT	1.94uJ	**1.16uJ**	**1.67x**
Swin	5.76uJ	**3.83uJ**	**1.51x**

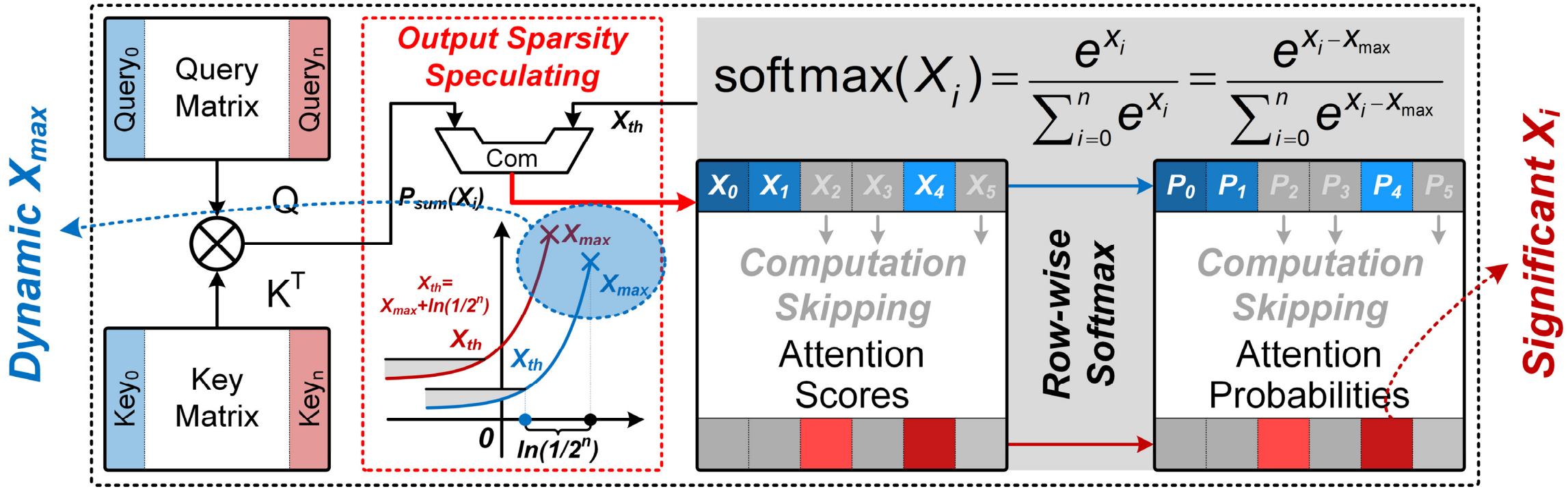
- BESA PE achieves **4.4% higher accuracy** and reduces **25.5% more energy** than previous methods.
- BESA PE improves **1.62x** energy efficiency for attention block.

Outline

- Background and Motivation
- Challenges of Global-attention-based Transformer Processor
- **Proposed Transformer Processor**
  - Big-exact-small-approximate Processing Element
  - **Bidirectional Asymptotic Sparsity Speculation**
  - Out-of-order PE-line Computing
- Measurement and Comparison
- Conclusion

Feature 2: Asymptotical Sparsity Speculation

Output Sparsity Speculation for Redundancy Skipping



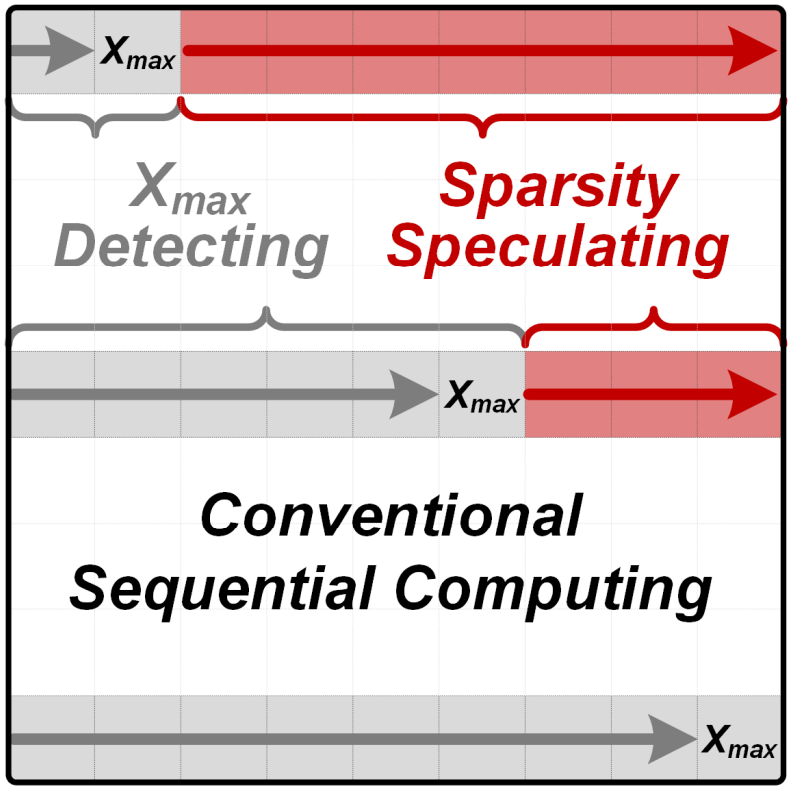
- **Dynamic** row-wise X_{max} detecting —> Diagonal-prior computing.
- **Lossless** X_i redundancy skipping —> Positive-prior speculating.

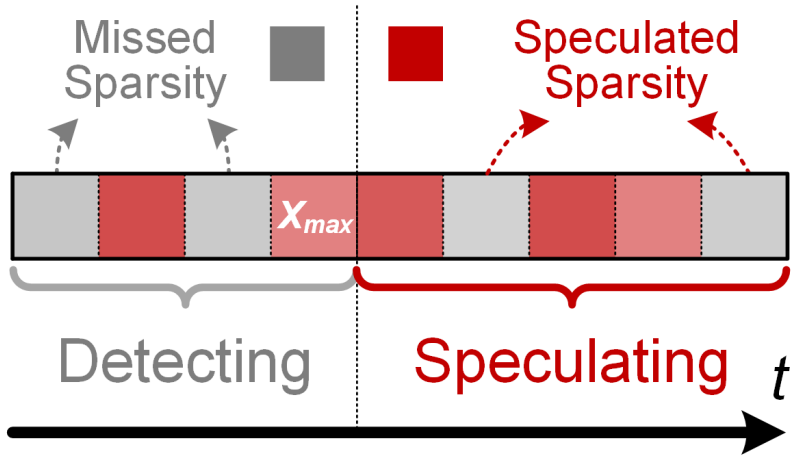
Feature 2: Asymptotical Sparsity Speculation

- **Low Speculation Efficiency with Previous Sequential Computing**

Step1: Find the row-wise varied maximum attention score X_{max}

Step2: **Speculate the output sparsity with X_i and X_{max} based on X_i-X_{max}<$(1/\ln 2^n)$**



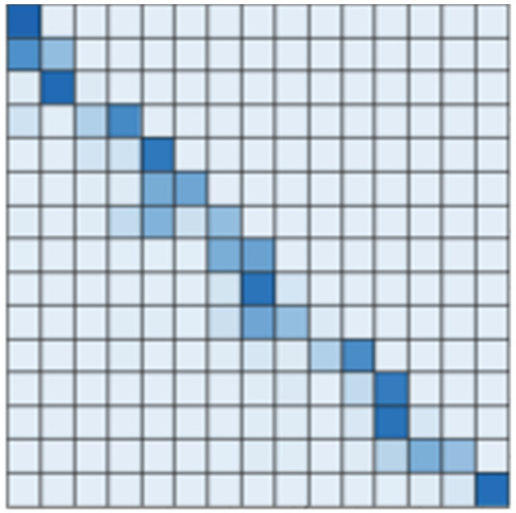


Missing Average 49.6% Dynamic Sparsity

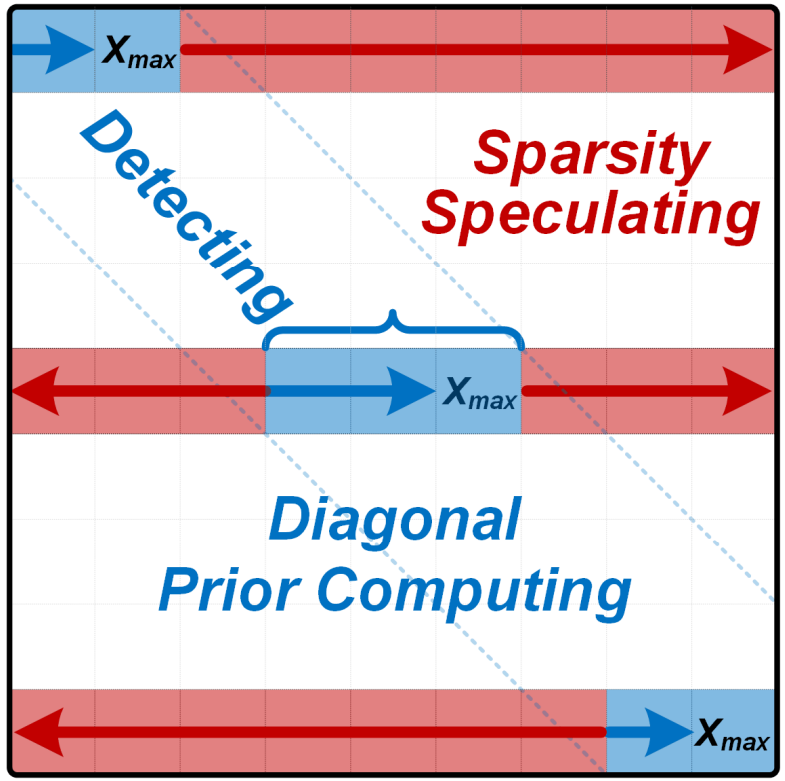
(ImageNet on ViT-B Model)

Feature 2: Asymptotical Sparsity Speculation

■ Step1: Proposed Efficient Diagonal-prior Computing



A specific token always has strong relevance with its near tokens.



Update X^*_{max} if $X_{max\_n} > X^*_{max}$

X^*_{max} Speculating with X^*_{max} X_{max_n}

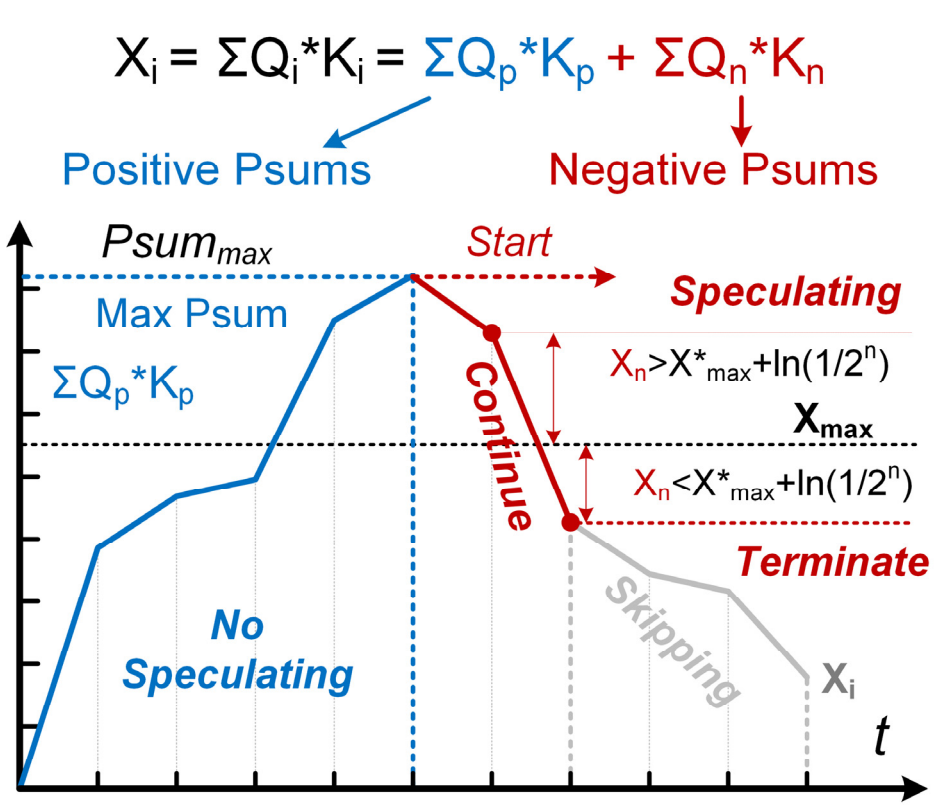
New X_{max}

$X_i - X^*_{max} \leq X_i - X_{max}$

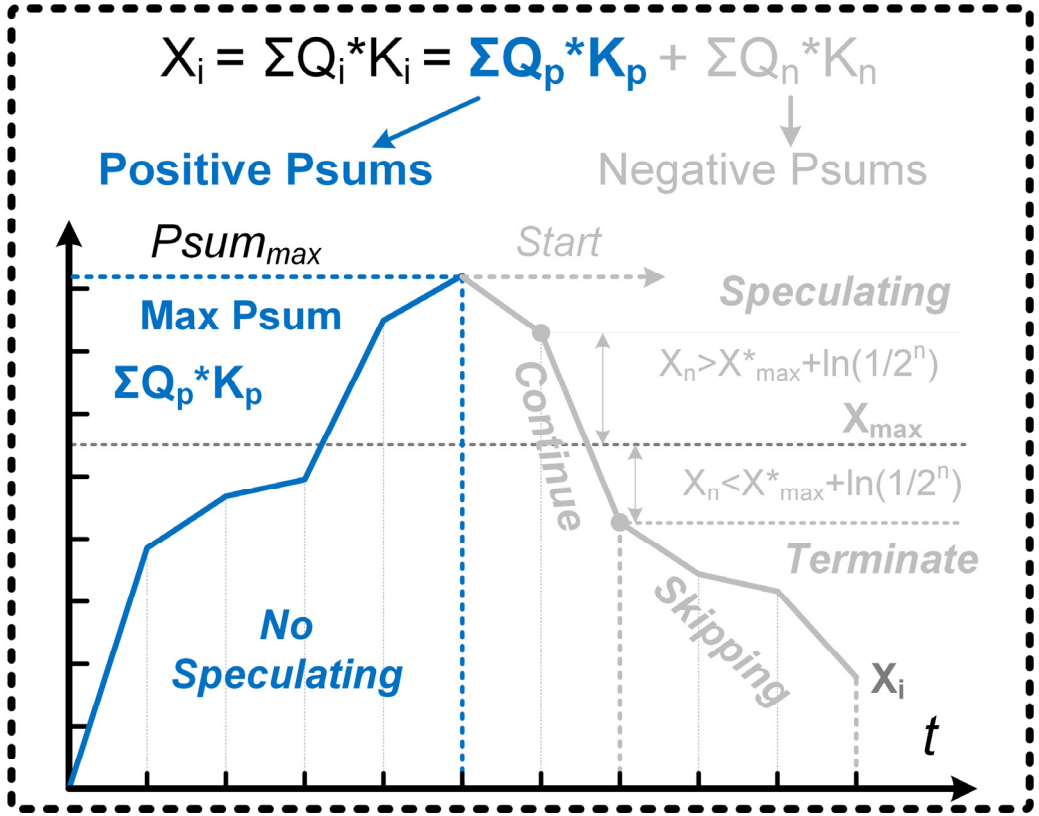
Numerical Fidelity for Speculation

Feature 2: Asymptotical Sparsity Speculation

■ Step2: Proposed Lossless Positive-prior Speculating



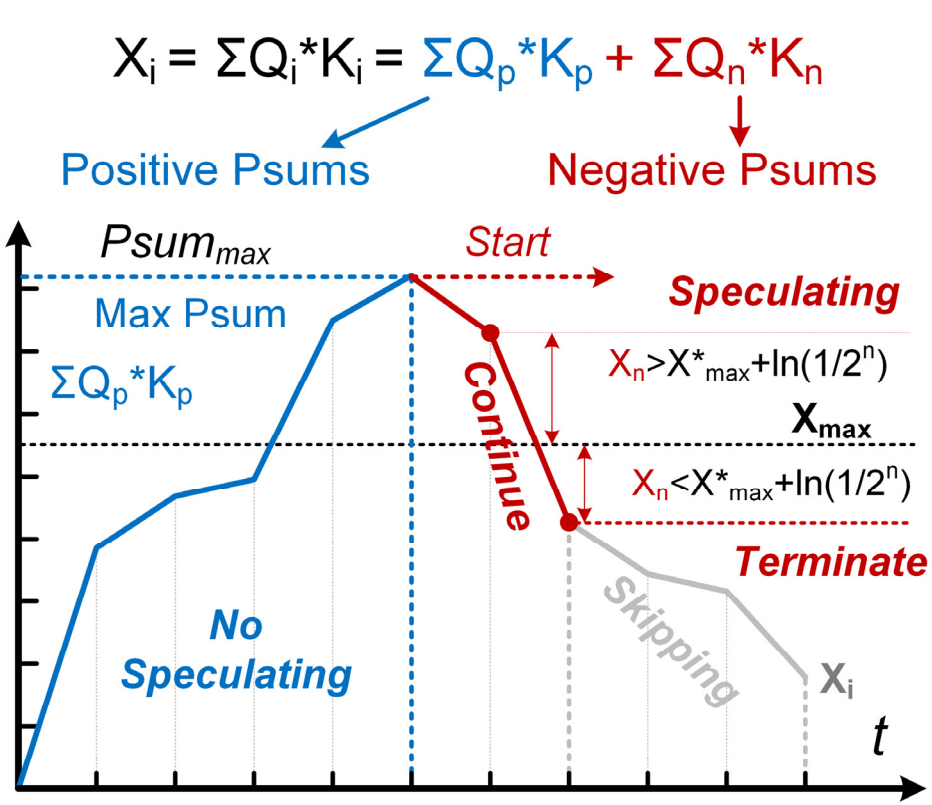
X_i Redundancy Skipping



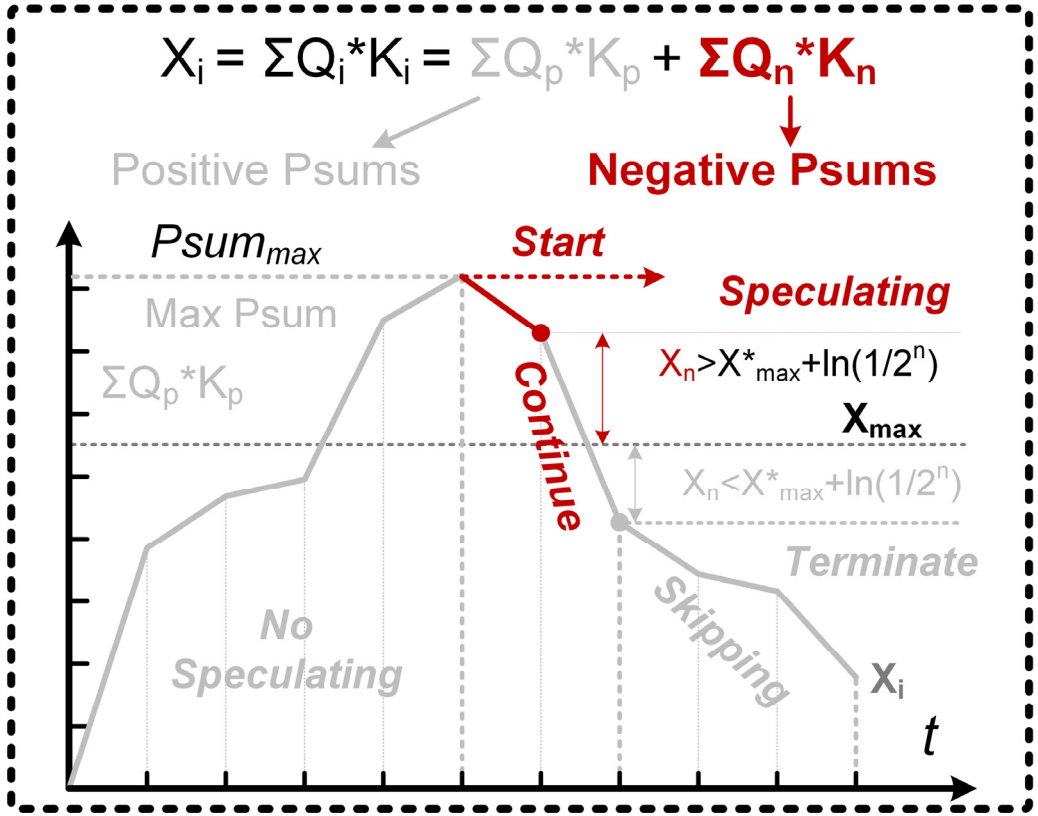
Positive Output Computing

Feature 2: Asymptotical Sparsity Speculation

■ Step2: Proposed Lossless Positive-prior Speculating



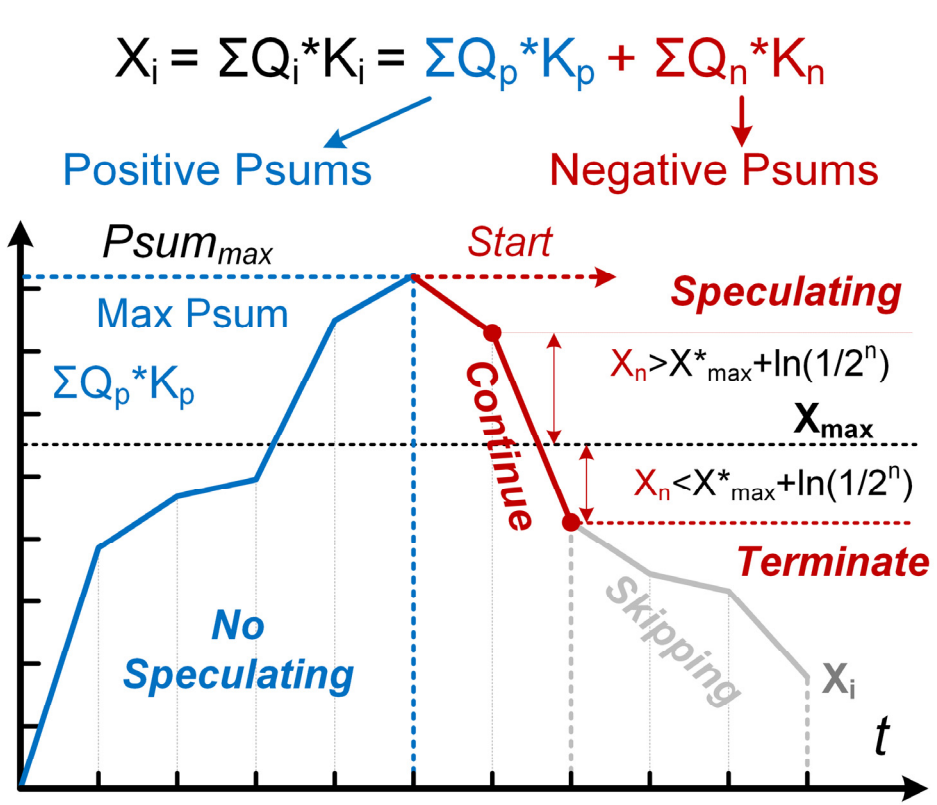
X_i Redundancy Skipping



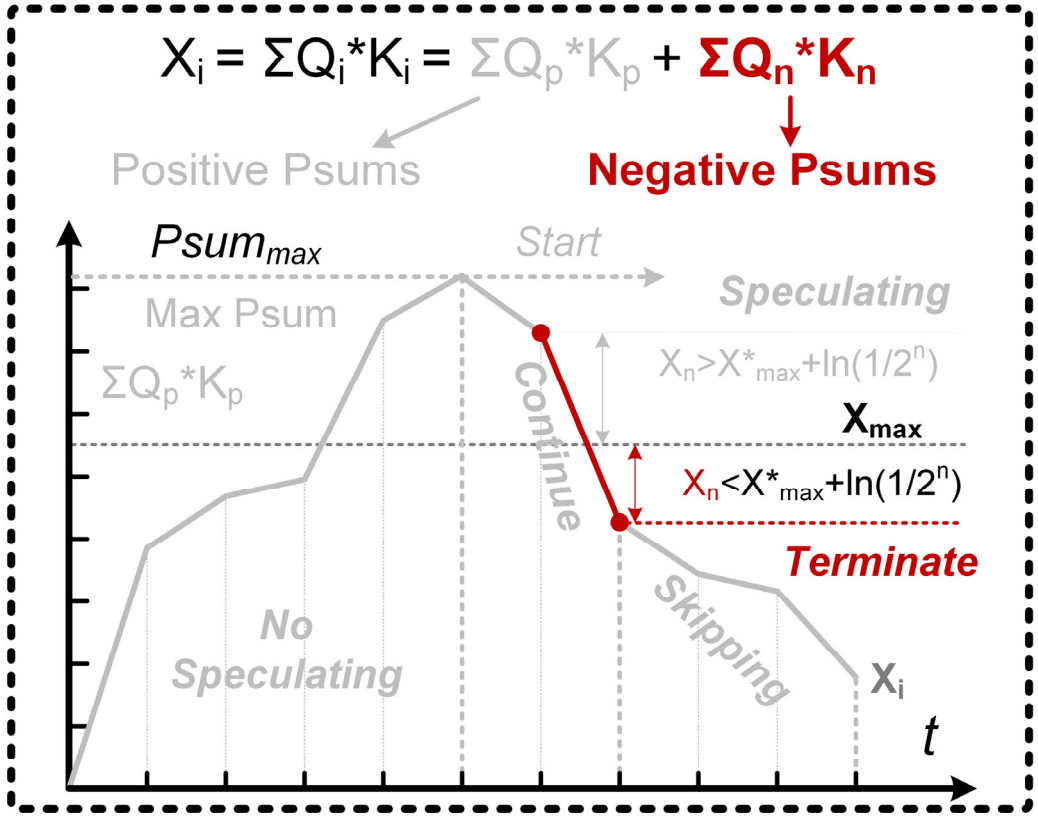
Start Speculating

Feature 2: Asymptotical Sparsity Speculation

■ Step2: Proposed Lossless Positive-prior Speculating



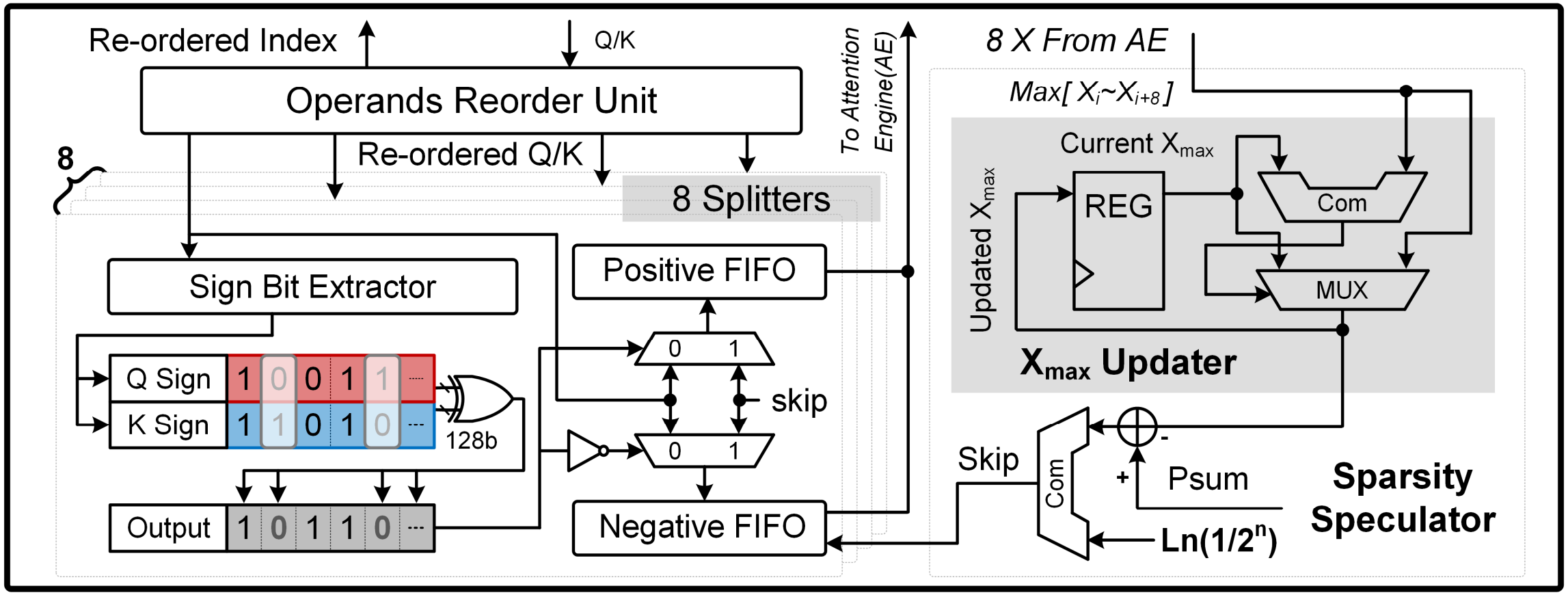
X_i Redundancy Skipping



Terminate Computing

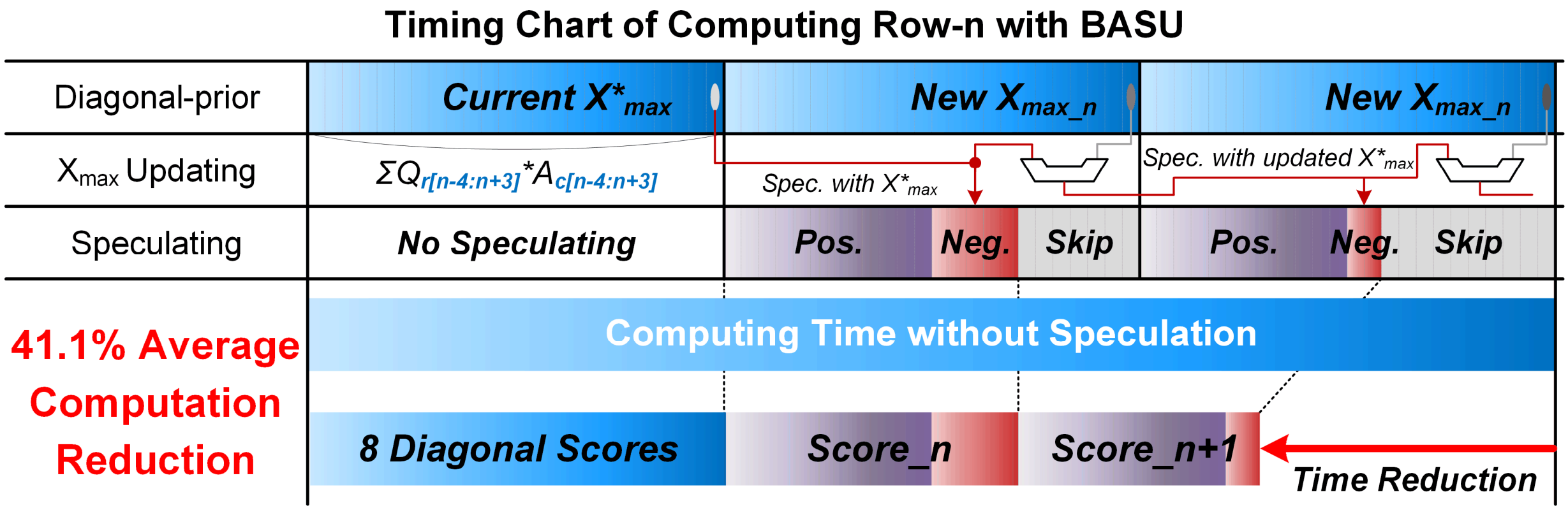
Feature 2: Asymptotical Sparsity Speculation

■ Architecture of Bidirectional Asymptotical Speculation Unit



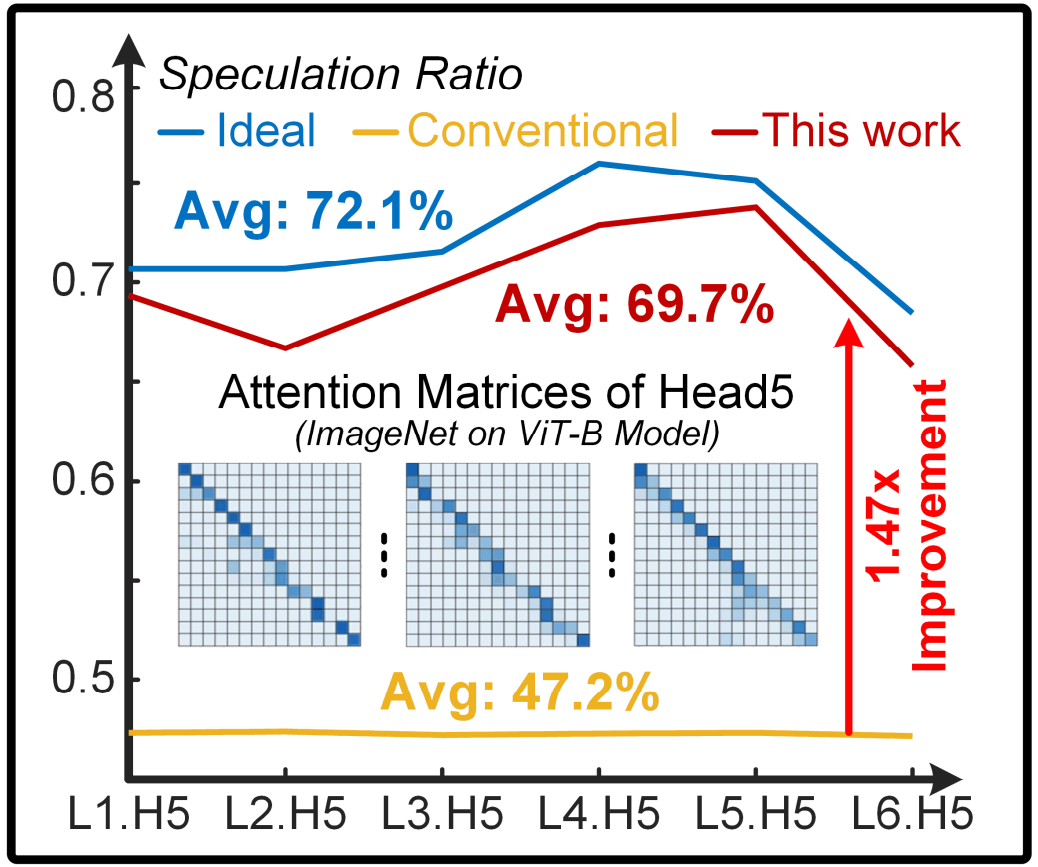
Feature 2: Asymptotical Sparsity Speculation

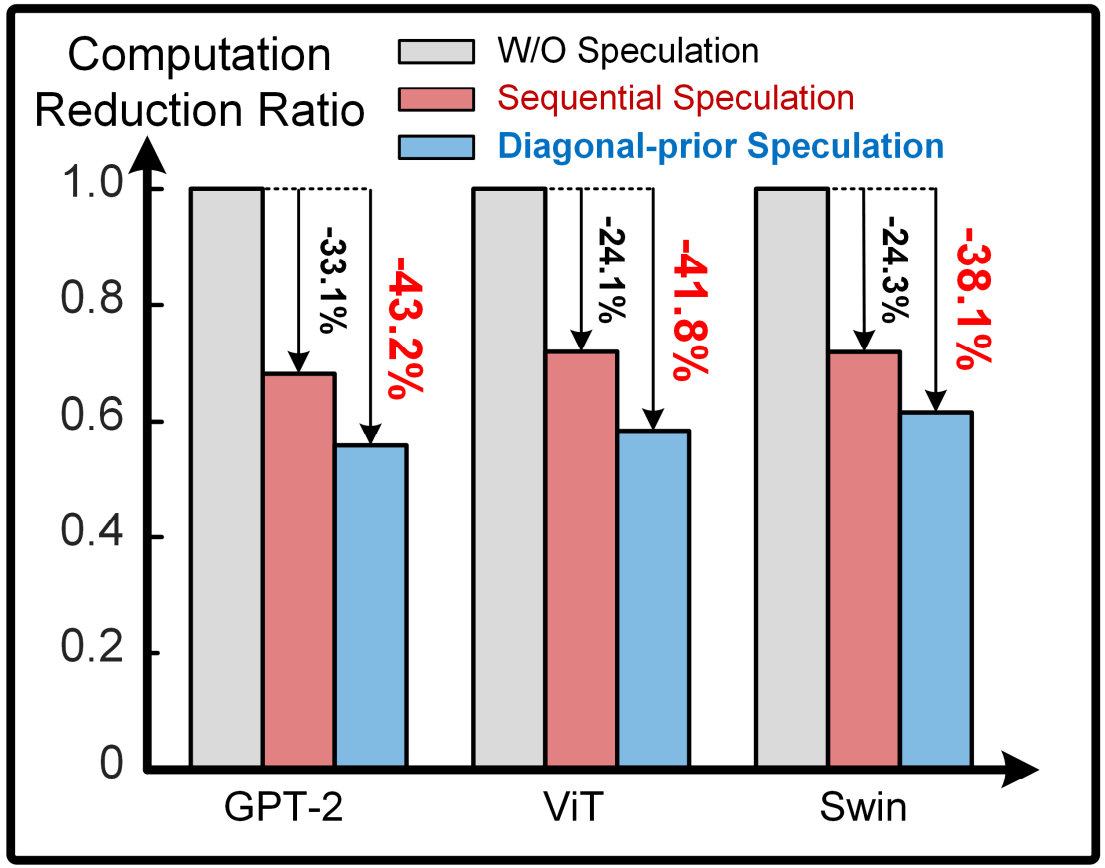
■ Time Chart of Bidirectional Asymptotical Speculation Unit



Feature 2: Asymptotical Sparsity Speculation

■ Performance Improvement with Asymptotical Speculation



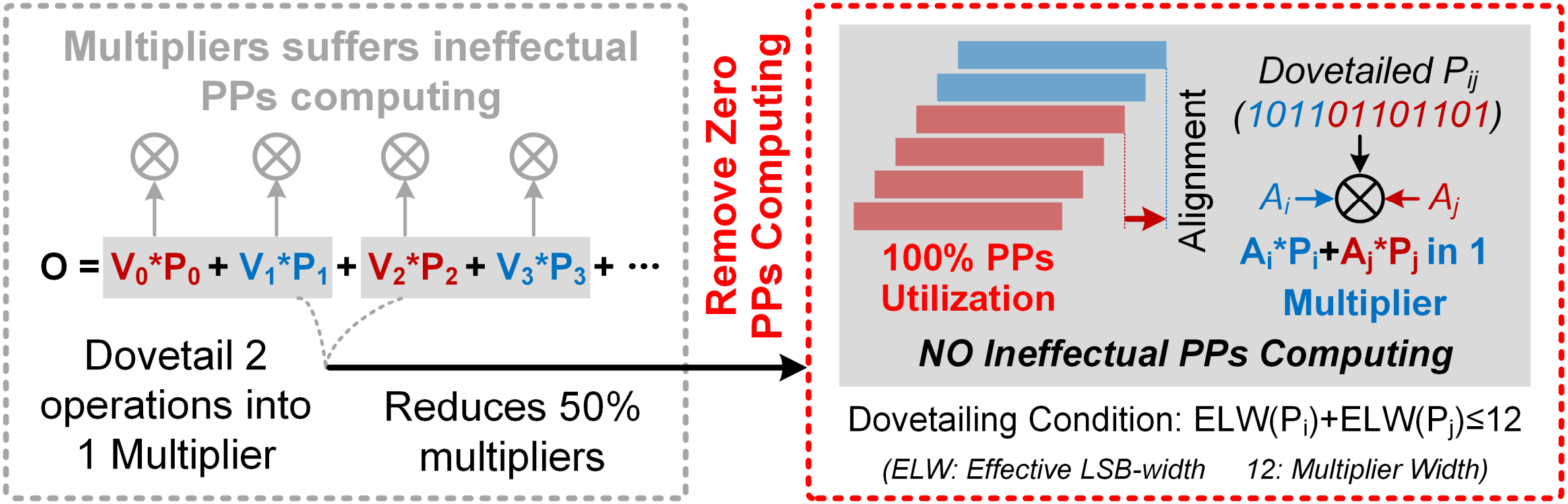


Outline

- Background and Motivation
- Challenges of Global-attention-based Transformer Processor
- **Proposed Transformer Processor**
  - Big-exact-small-approximate Processing Element
  - Bidirectional Asymptotic Sparsity Speculation
  - **Out-of-order PE-line Computing**
- Measurement and Comparison
- Conclusion

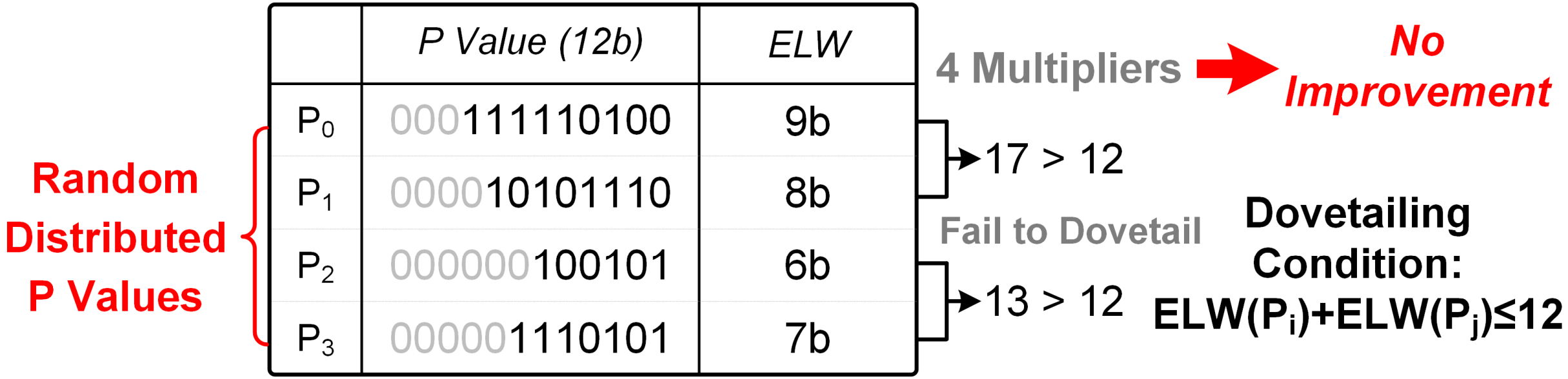
Feature 3: Out-of-order PE-line Computing

■ Operation Dovetailing for Hardware Utilization Improvement



Feature 3: Out-of-order PE-line Computing

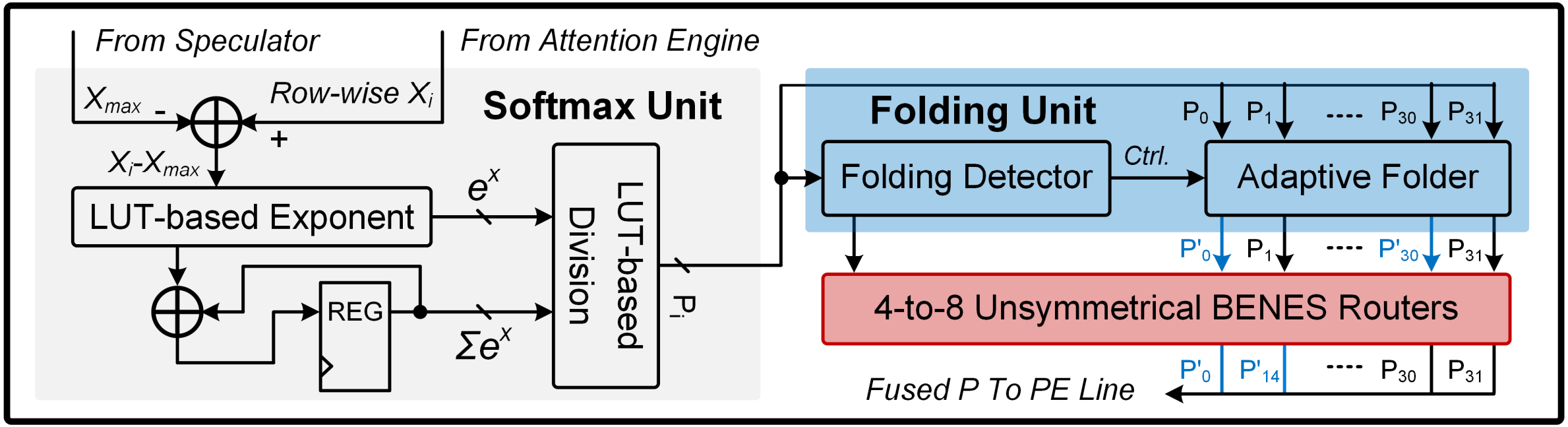
- **Limited Utilization Increase with Previous In-order Dovetailing**



- **Two factors affect the dovetailing ratio:**
  - **The effective bit-width of each probabilities value.**
  - **The effective bit-width sum of the adjacent two probabilities values.**

Feature 3: Out-of-order PE-line Computing

- **Proposed Out-of-order PE-line Computing**



- **Adaptive folding unit reduces effective bit-width.**
- **Unsymmetrical BENES router reduces reordering overhead.**

Feature 3: Out-of-order PE-line Computing

Adaptive Folding for Effective Bit-width Reduction

	P Value (12b)	ELW
P_0	000111110100	9b
P_1	00010101110	8b
P_2	000000100101	6b
P_3	00000111010 1	7b

4 Multipliers

→ 17 > 12

Fail to Dovetail

→ 13 > 12

After Folding

	P Value (12b)	ELW
P'_0	000000001011	4b
P_1	00010101110	8b
P_2	000000100101	6b
P'_3	00000001010	4b

2 Multipliers

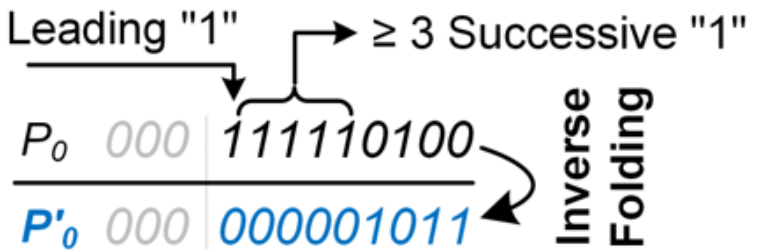
→ 12

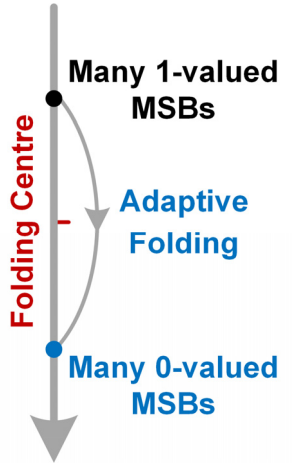
Dovetailed

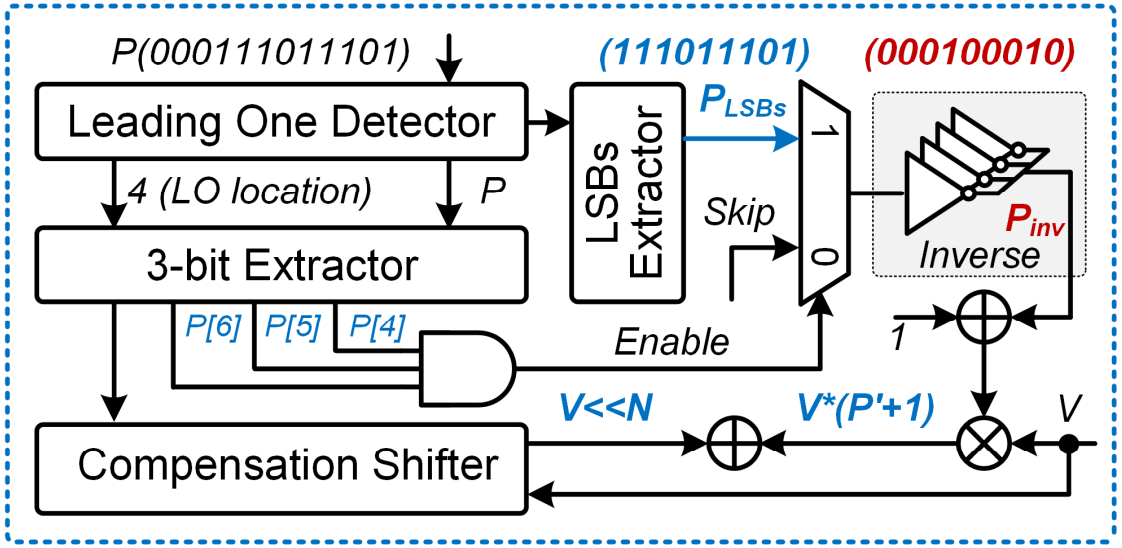
→ 12

Location of leading "1"

$$V_i*P_i = V_i*(P'_i+1) + V_i<<N$$

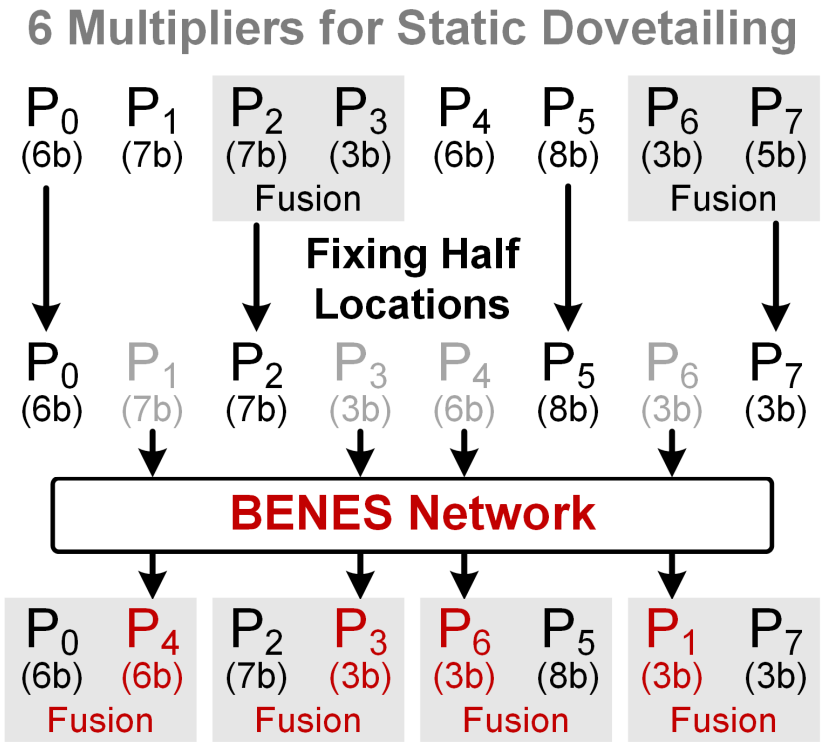


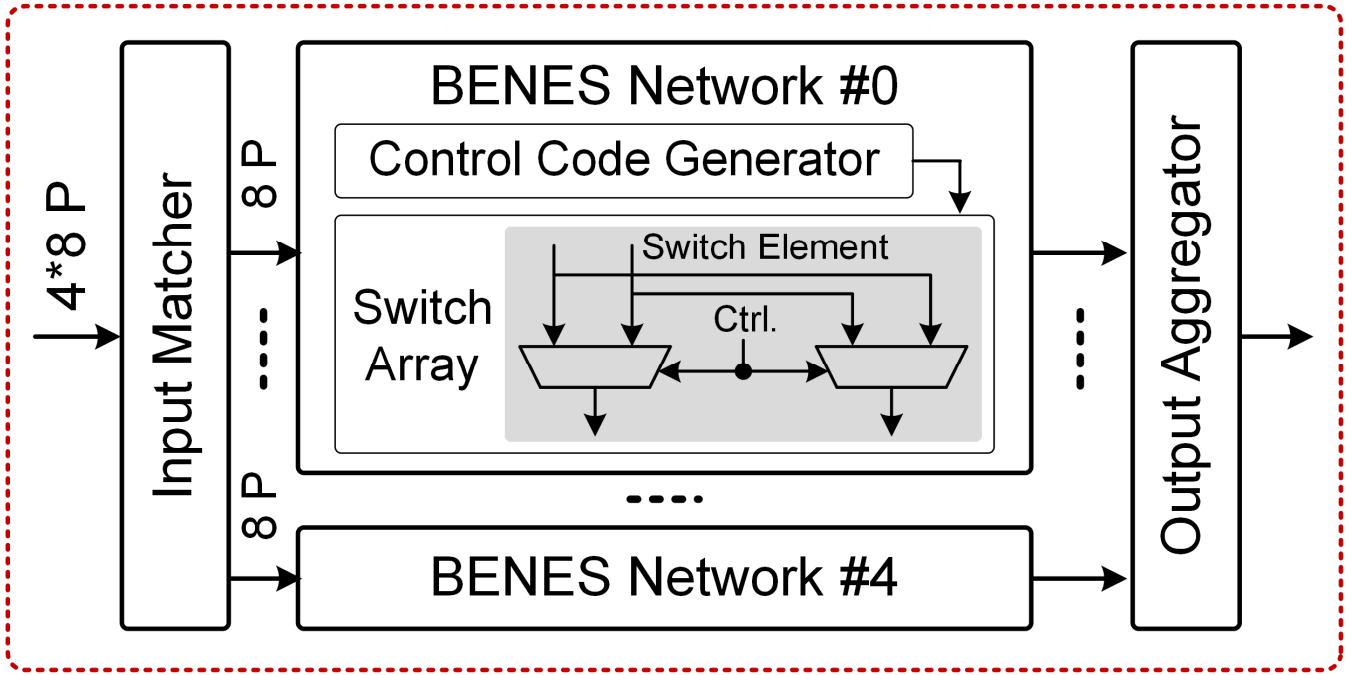




Feature 3: Out-of-order PE-line Computing

Operands Reordering with BENES Network

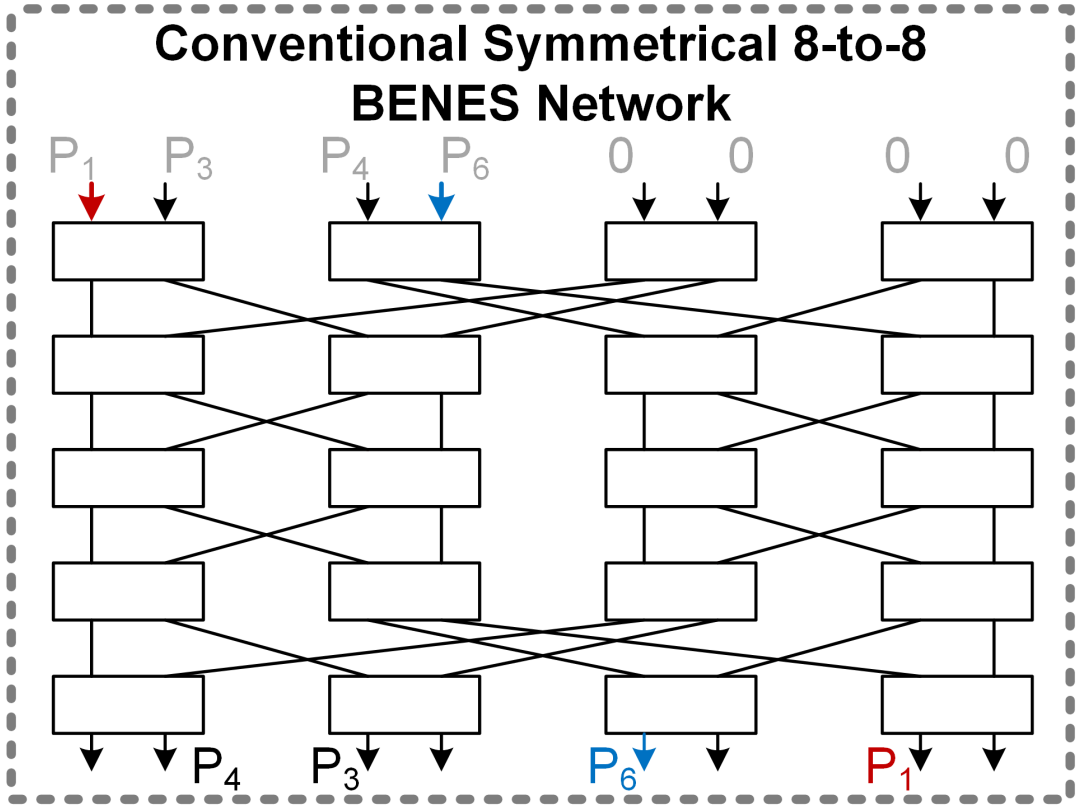


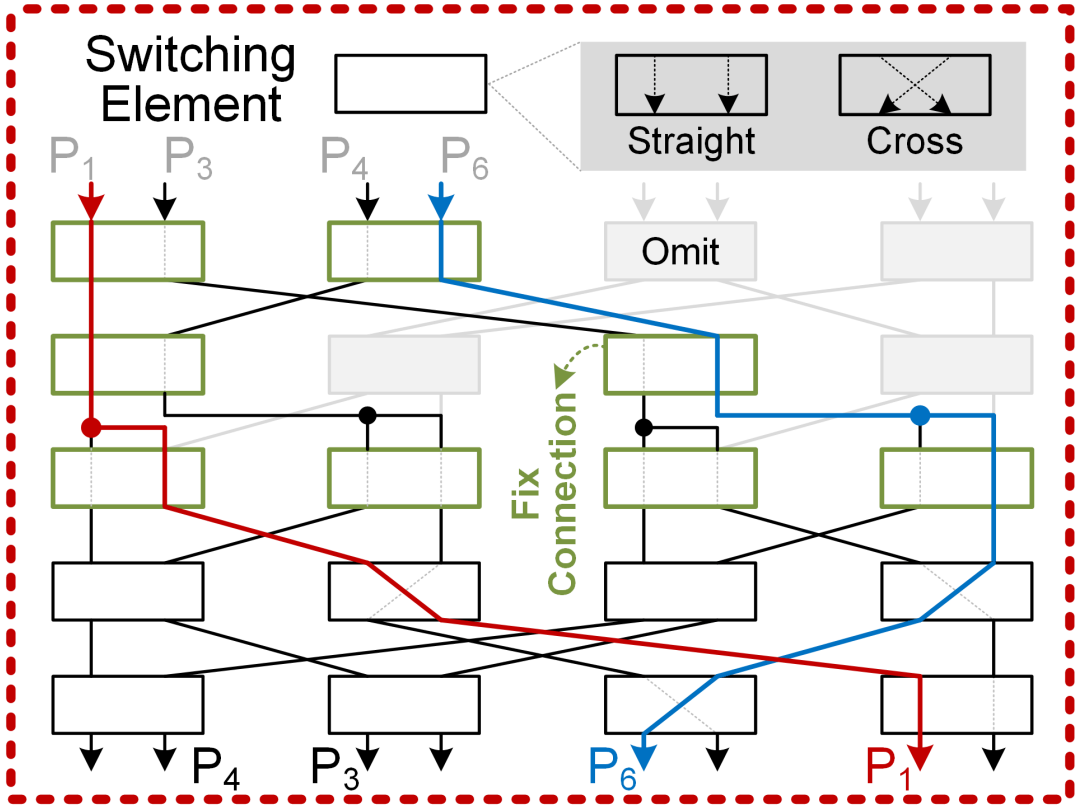


Computation reordering increase 32.3% dovetailing efficiency.

Feature 3: Out-of-order PE-line Computing

- **Asymmetrical BENES for Low-power Reordering**



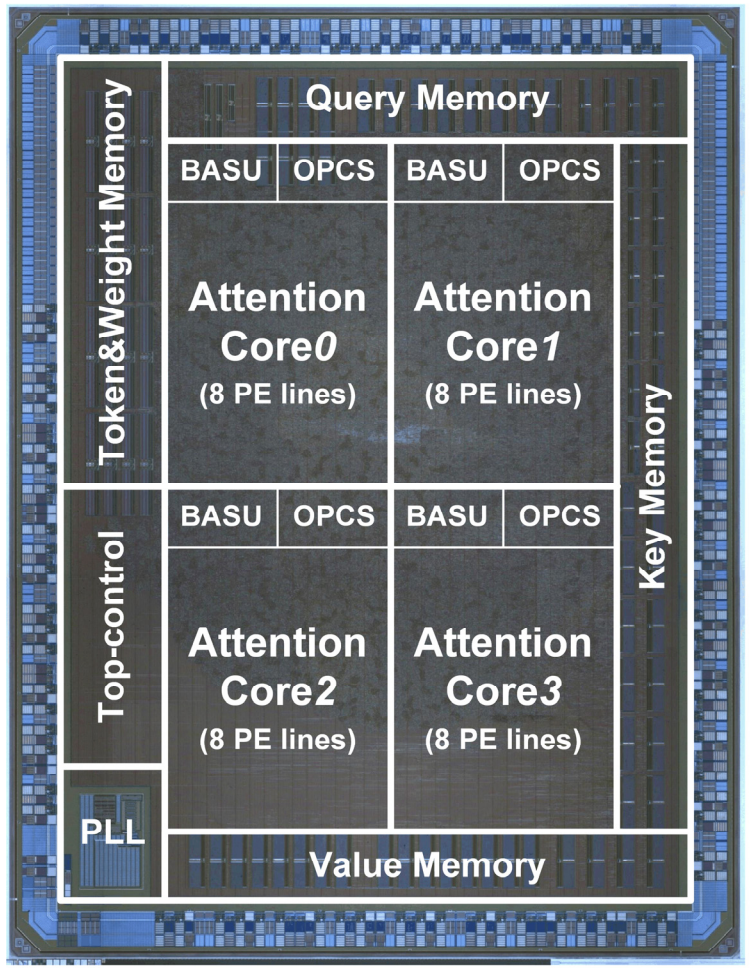


Asymmetrical BENES reduces 56.7% power and 46.2% area overhead.

Outline

- Background and Motivation
- Challenges of Global-attention-based Transformer Processor
- Proposed Transformer Processor
  - Big-exact-small-approximate Processing Element
  - Bidirectional Asymptotic Sparsity Speculation
  - Out-of-order PE-line Computing
- **Measurement and Comparison**
- Conclusion

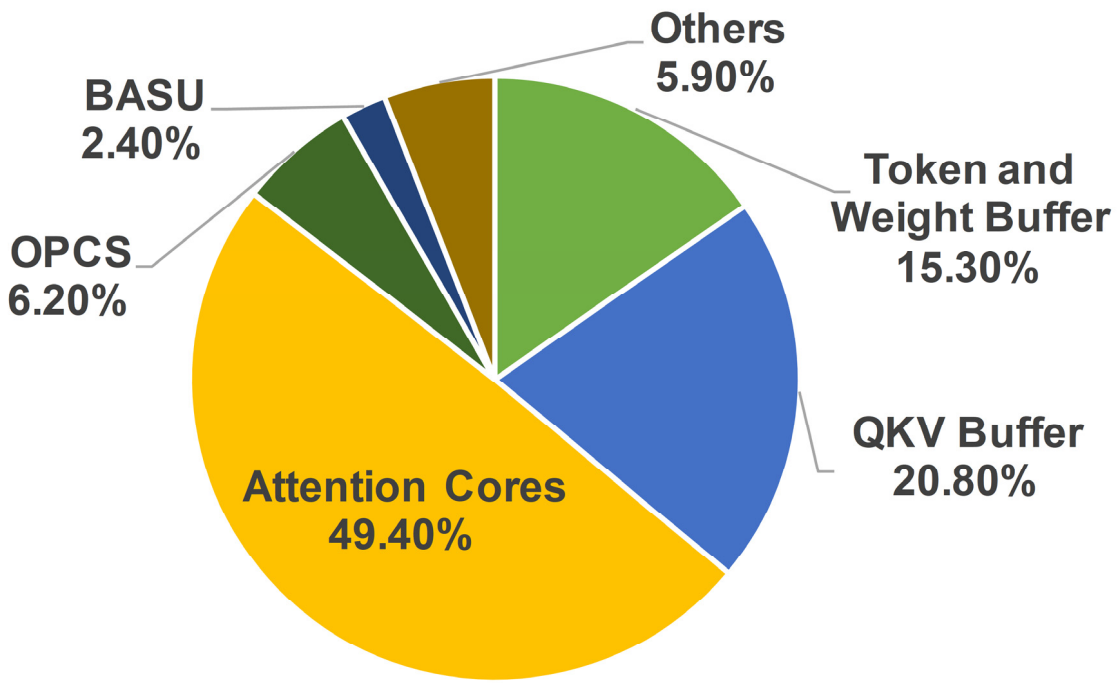
Chip Photograph and Summary



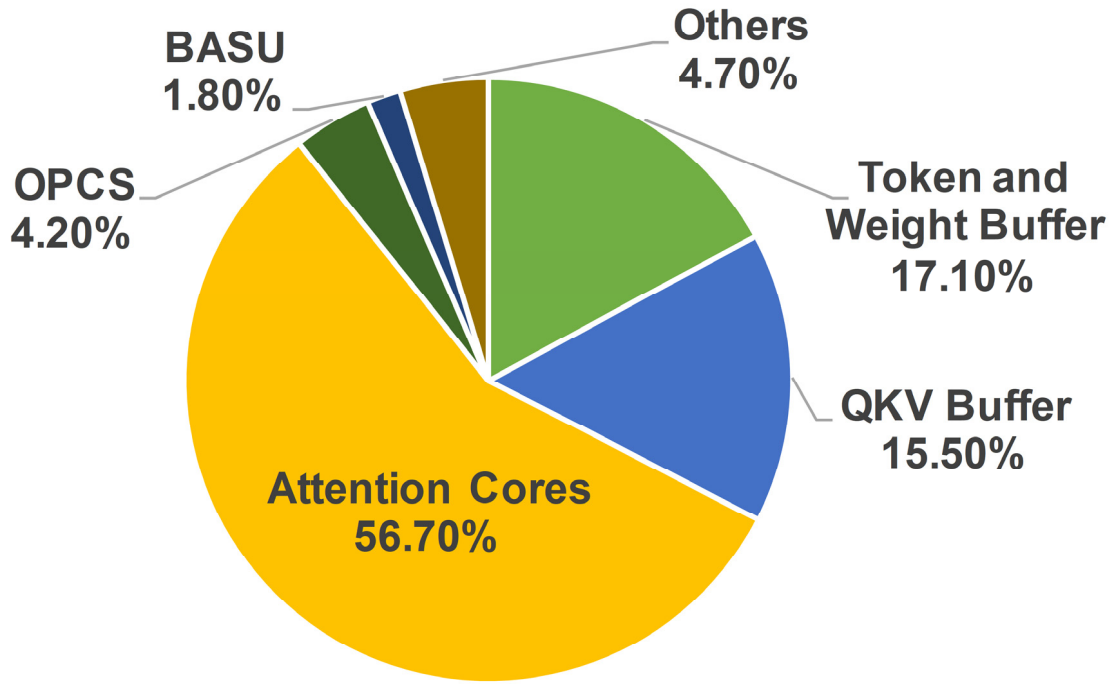
	Specifications	
Technology	28nm CMOS	
Die Area (mm^2)	6.82	
SRAM (KB)	336	
Voltage (V)	0.56 - 1.1	
Frequency (MHz)	50 - 510	
Data Precision	INT12	
Power (mW)	12.06 – 272.8	
Peak Performance (TOPS)	MM [1]	0.522 @ 1.1V, 510MHz
	QK^T	0.522[2] – 1.26[3] @ 1.1V, 510MHz
	PV	0.522[2] – 4.07[4] @1.1V, 510MHz
Energy Efficiency (TOPS/W)	MM [1]	1.91 @ 1.1V, 510MHz
		4.25 @ 0.56V, 50MHz
	QK^T	2.60[2] – 6.27[2][3] @ 1.1V, 510MHz
		5.45[2] – 12.47[2][3] @ 0.56V, 50MHz
	PV	2.60[2] – 14.28[2][4] @ 1.1V, 510MHz
		5.45[2] – 27.56[2][4] @ 0.56V, 50MHz

One operation (OP) represents one multiplication or addition.
1) Including Q, K, V generation and FFN computation.
2) 90% weak-related tokens for approximate computing.
3) 80% output sparsity ratio in Q and K^T multiplications.
4) 80% near-zero and zero probabilities in P and V multiplications.

Power and Area Breakdown



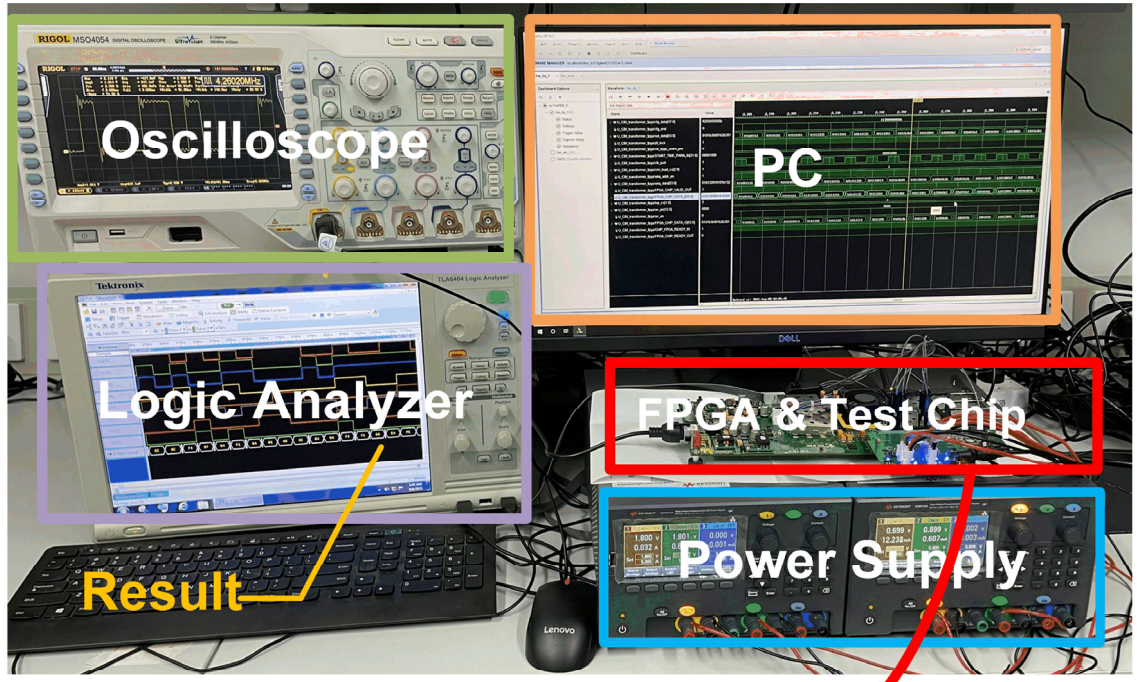
Area Breakdown

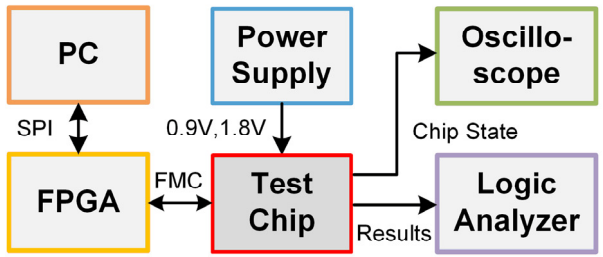


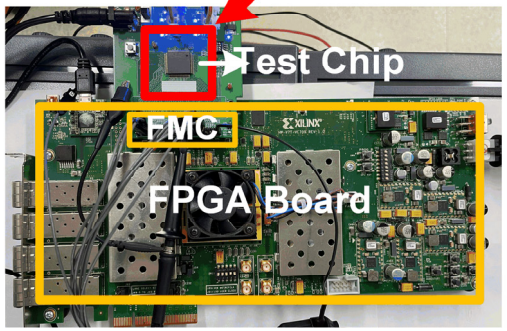
Power Breakdown

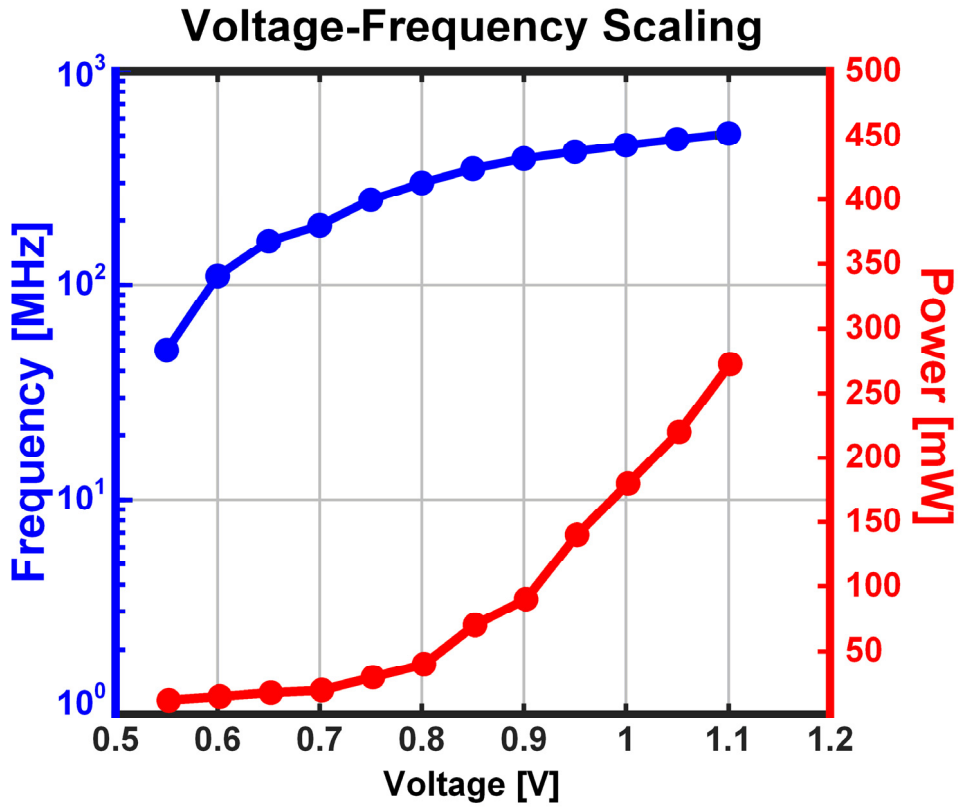
- BASU and OPCS take limited area (8.8%) and power (6.0%)
- Attention cores take most area (49.4%) and power (56.7%)

Test Platform and Power Scaling







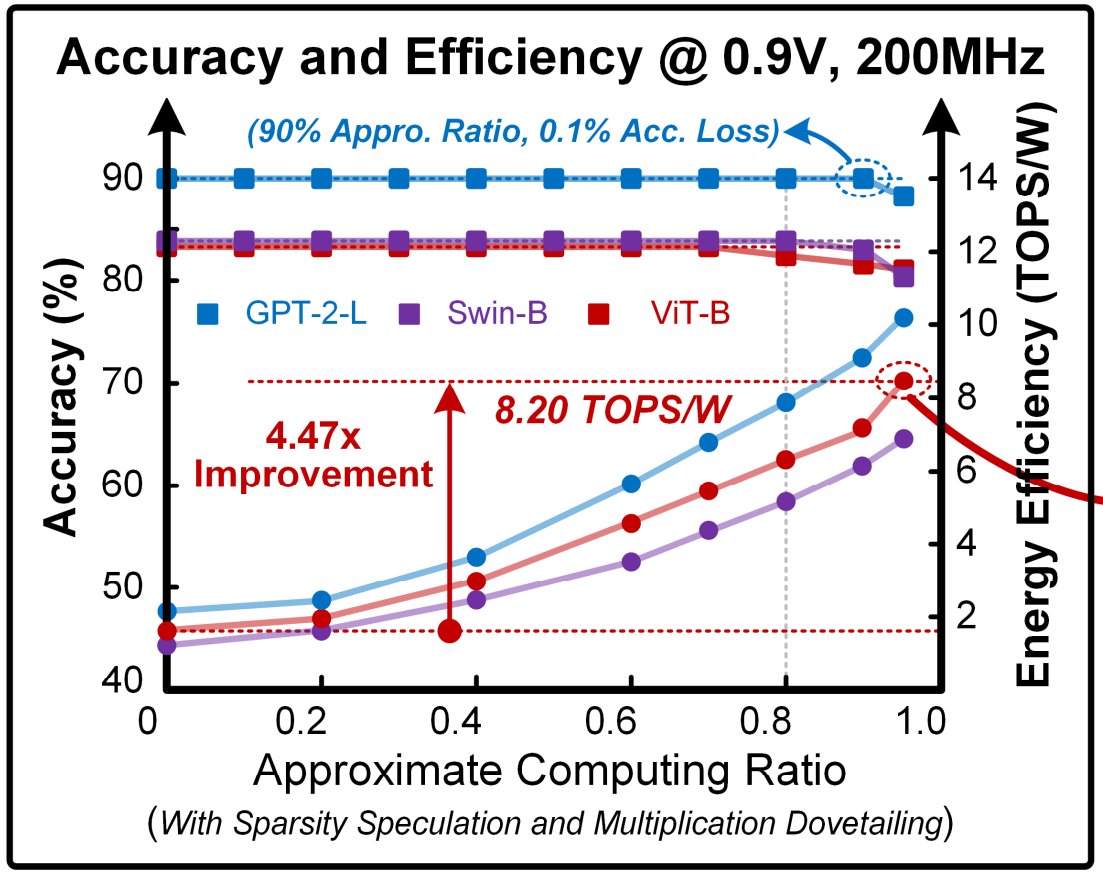


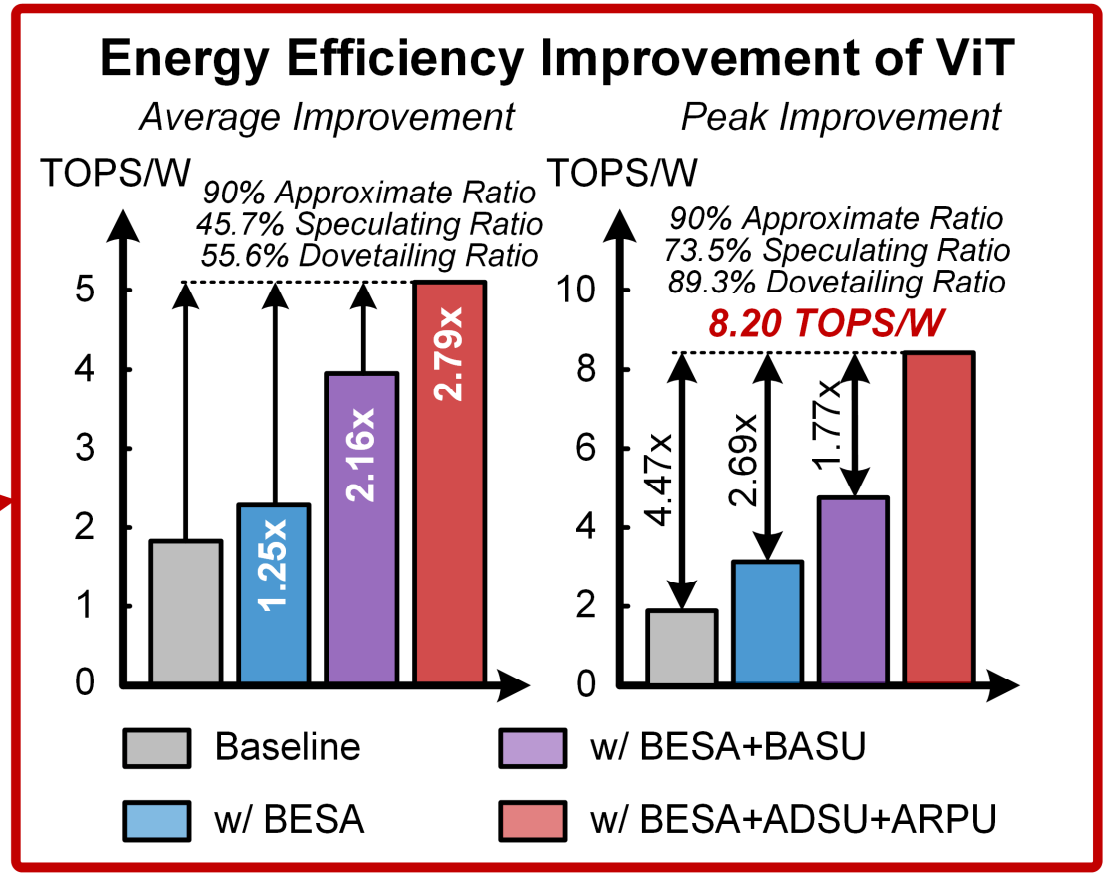
(27.56TOPS/W@0.56V, 50MHz)

(14.28TOPS/W@1.1V,510MHz)

Performance Evaluation

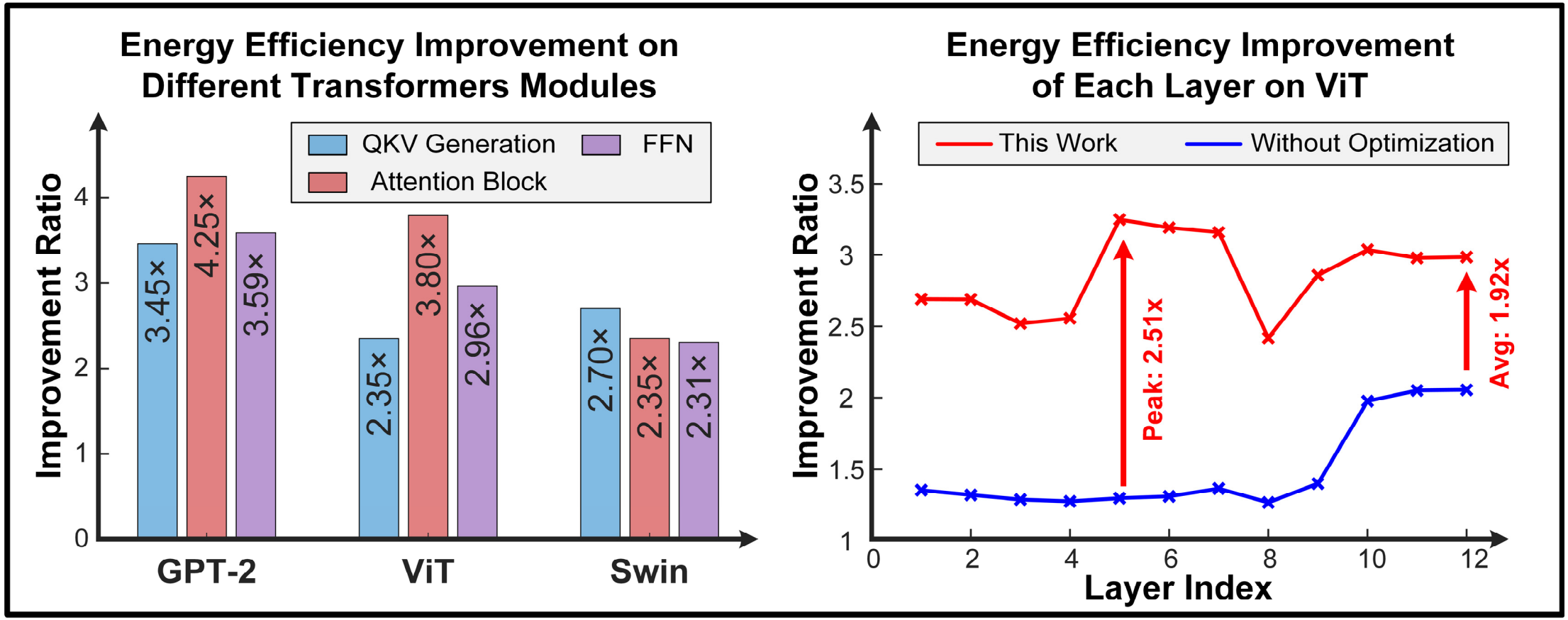
Energy Efficiency Improvement with Proposed Techniques





Performance Evaluation

- **Effective for Q/K/V Generation, Attention Block, and FFN**



Comparison with SOTA Processors

	GPU A100 [6]	JSSC'20 [4]	ISSCC'21 [5]	HPCA'20 [2]	ISCA'21 [3]	This Work
Function[1]	CNN, RNN, SpMM	LSTM, RNN	RNN, CNN, LSTM	Transformer[2]	Transformer	**Transformer**[3]
Sparsity Support	Input	Input	Input	No	Output	**Input/Output**
Technique (nm)	7	65	7	40	40	**28**
Die Area (mm^2)	826	7.74	19.6	2.08	1.26	**6.82**
Supply Voltage (V)	NA	0.6 - 1.1	0.55 – 0.75	1.1	1.1	**0.56 – 1.1**
Frequency (MHz)	1410	8 - 80	1000 - 1600	1000	1000	**50-510**
Precision	FP64/32/16, INT8/4	6 (Weight), 13 (Activation)	HFP8, FP32/16, INT4/2	INT9	INT8, FP16	**INT12**
Power (mW)	400000	1.85 – 67.3	NA	110.42	969.36	**12.06 – 272.8**
Performance (TOPS or TFLOPS)	9.7@FP64 2486@INT4[4]	0.025 – 0.16	8@FP16 102.4@INT4	0.22	1.09	**0.52 – 4.07**[5]
Energy Efficiency (TOPS/W or TFLOPS/W)	0.024@FP64 6.24@INT4[4]	2.45-8.93	0.98@FP16 16.5@INT4	1.99	1.12	**1.91 – 27.56**[5]

1) The main computations of RNN, LSTM and Transformer are matrix multiplication. It is fair to compare them together.
2) Only support single token NLP tasks, such as Q/A application. 3) Support multiple attention tasks, including translation, classification, detection and so on.
4) Only support 50% structured weight sparsity. 5) P and V multiplication with 90% weakly related token ratio.

Outline

- Background and Motivation
- Challenges of Global-attention-based Transformer Processor
- Proposed Transformer Processor
  - Big-exact-small-approximate Processing Element
  - Bidirectional Asymptotic Sparsity Speculation
  - Out-of-order PE-line Computing
- Measurement and Comparison
- **Conclusion**

Conclusion

- An Energy Efficient Transformer Processor
  - Big-exact-small-approximate Processing Element
    - ✓ Reduce Energy Consumption for Weak-Related Tokens
  - Bidirectional Asymptotic Sparsity Speculation
    - ✓ Remove Redundant Attention Computation in $Q \times K^T$
  - Out-of-order PE-line Computing
    - ✓ Improve Hardware Utilization in $P \times V$

An Approximate-computing-based Transformer Processor with Asymptotic Sparsity Speculation and Out-of-order Computing Achieving 27.5TOPS/W Energy Efficiency

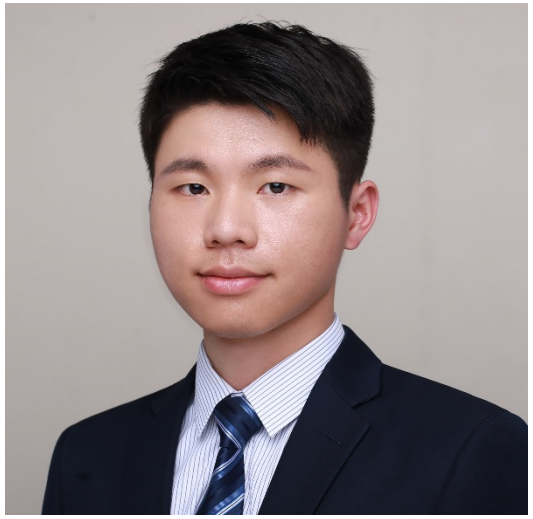
A 28nm 15.59μJ/Token Full-Digital Bitline-Transpose CIM-based Sparse Transformer Accelerator with Pipeline/Parallel Reconfigurable Modes

Fengbin Tu[1,2], Zihan Wu[1], Yiqi Wang[1], Ling Liang[2], Liu Liu[2], Yufei Ding[2], Leibo Liu[1], Shaojun Wei[1], Yuan Xie[2], Shouyi Yin[1]

[1]Tsinghua University, Beijing, China

[2]University of California, Santa Barbara, CA

Self Introduction



Fengbin Tu

- Ph.D. degree from Tsinghua University, Beijing, China, in 2019.

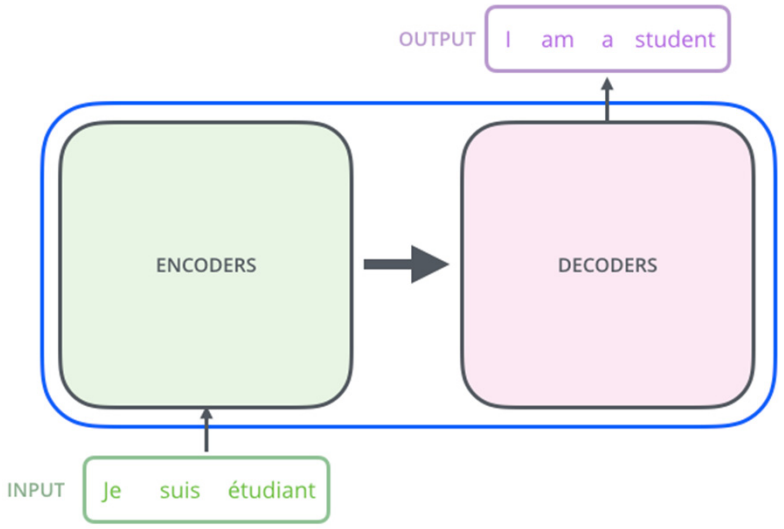
- Postdoc Researcher at University of California, Santa Barbara, CA, since 2019.

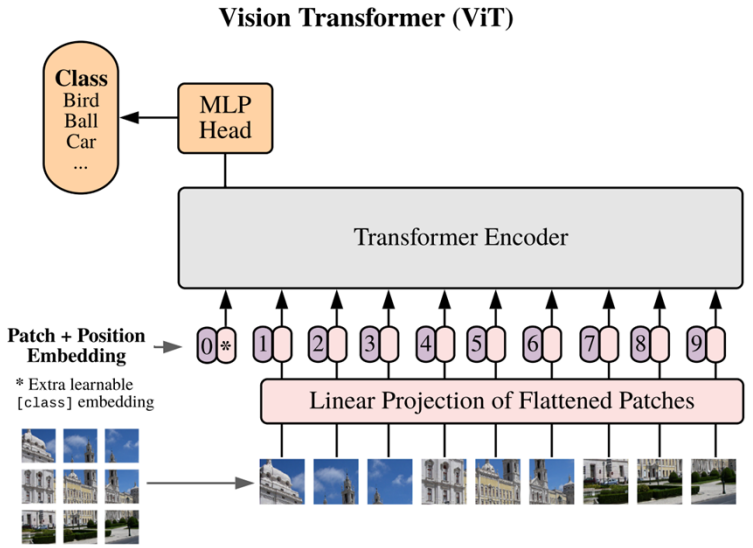
- My research interests include deep learning, in-memory computing, computer architecture, and reconfigurable computing.

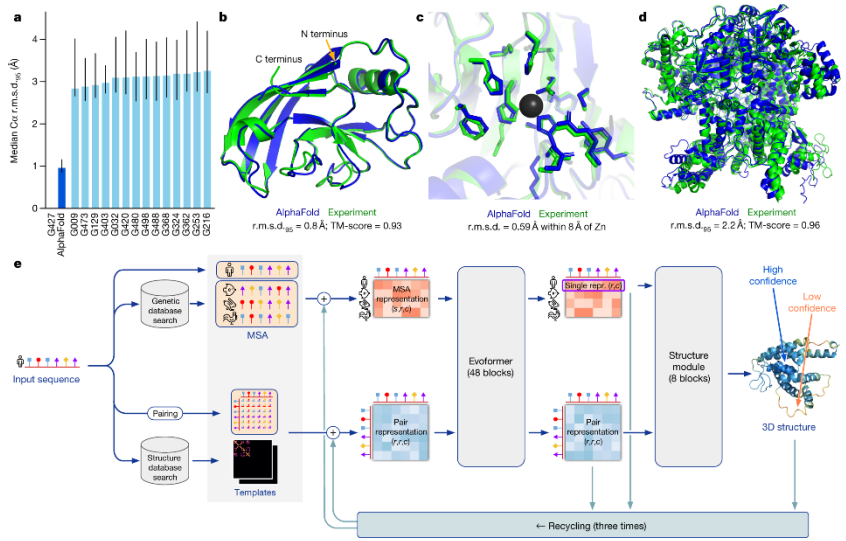
Outline

- **Motivation**
- **TranCIM Accelerator Features**
  - Pipeline/Parallel Reconfigurable Modes
  - Bitline-Transpose-CIM (BLT-CIM)
  - Sparse Attention Scheduler (SAS)
- **Measurement Results**
- **Conclusion**

Transformer Models are Widely Used

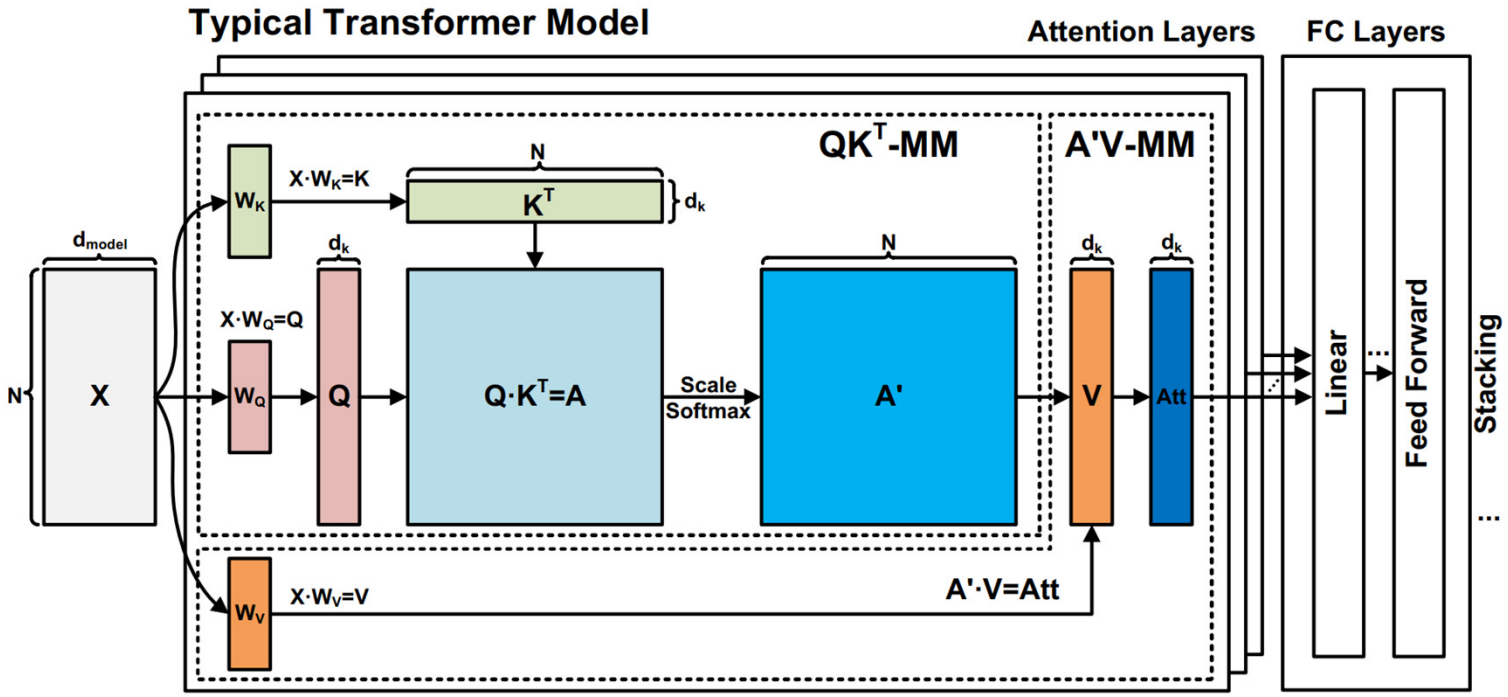






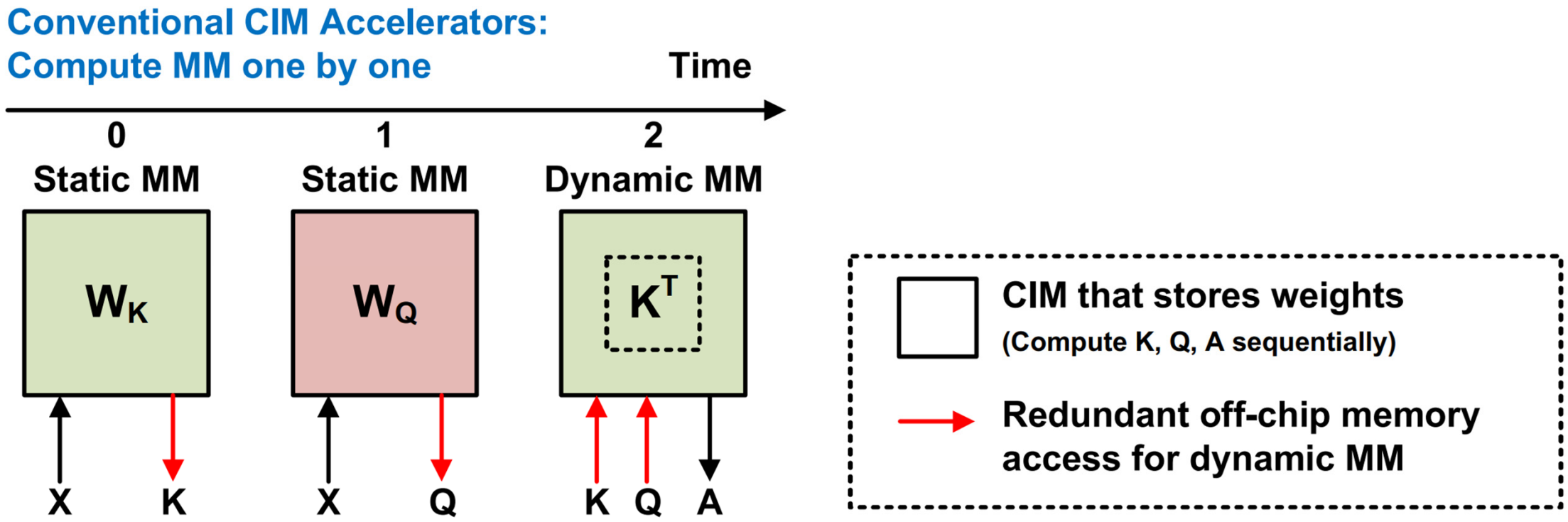
Natural Language Processing | **Computer Vision** | **Bioinformatics**

Can CIM Accelerate Transformer?



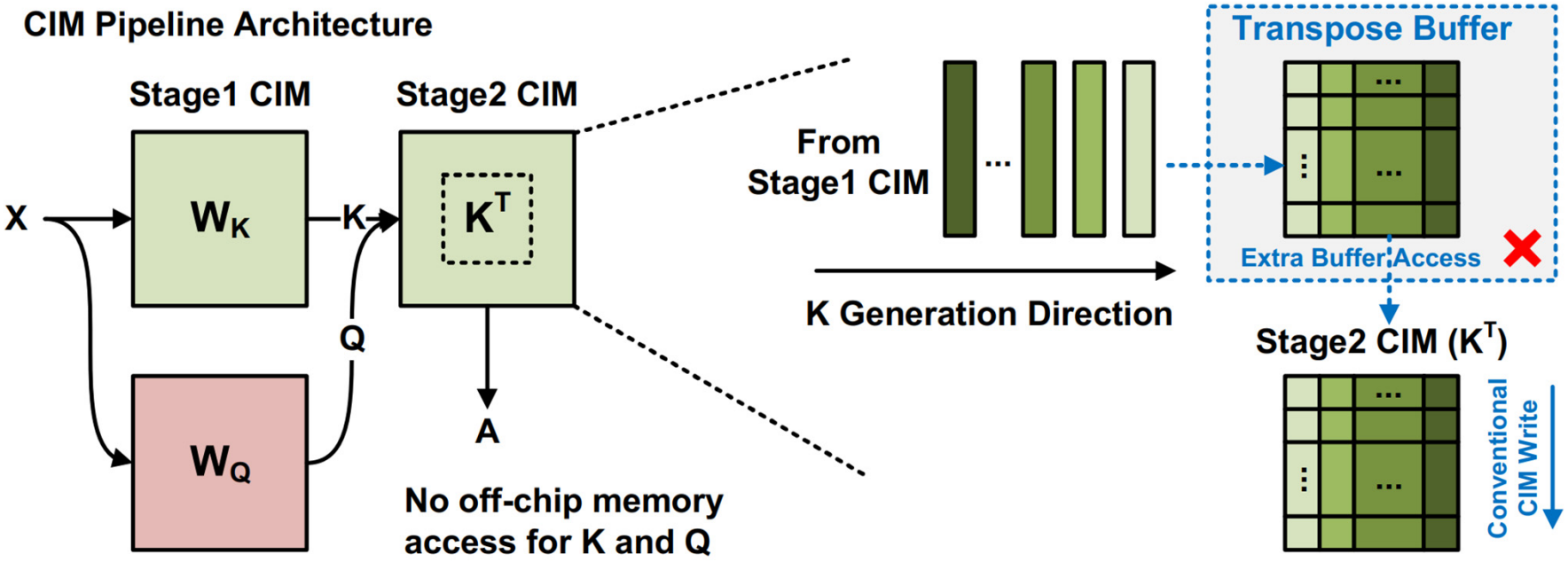
- **Transformer has many matrix multiplications (MM)**
- **Computing-in-Memory (CIM) is an efficient MM architecture**
- **Attention mechanism raises new challenges**
  - Memory access and computation

Challenge1a: Redundant Off-Chip Memory Access



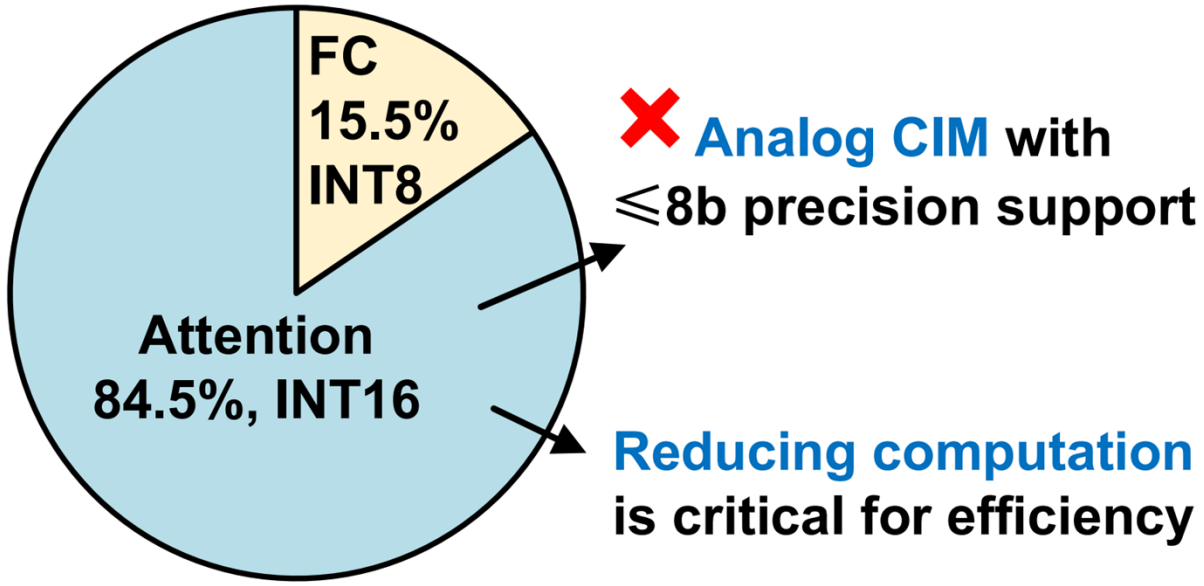
- **Dynamic MM: QK^T and A'V**
  - Weights and inputs are both generated during runtime
- **Redundant off-chip memory access for intermediate data**

Challenge1b: Extra Transpose Buffer and Access



- **CIM pipeline can mitigate Challenge1a**
- **QK^T pipeline needs a large transpose buffer**
  - K generation direction doesn't match the conventional CIM write direction

Challenge2: Attention Layer Computation



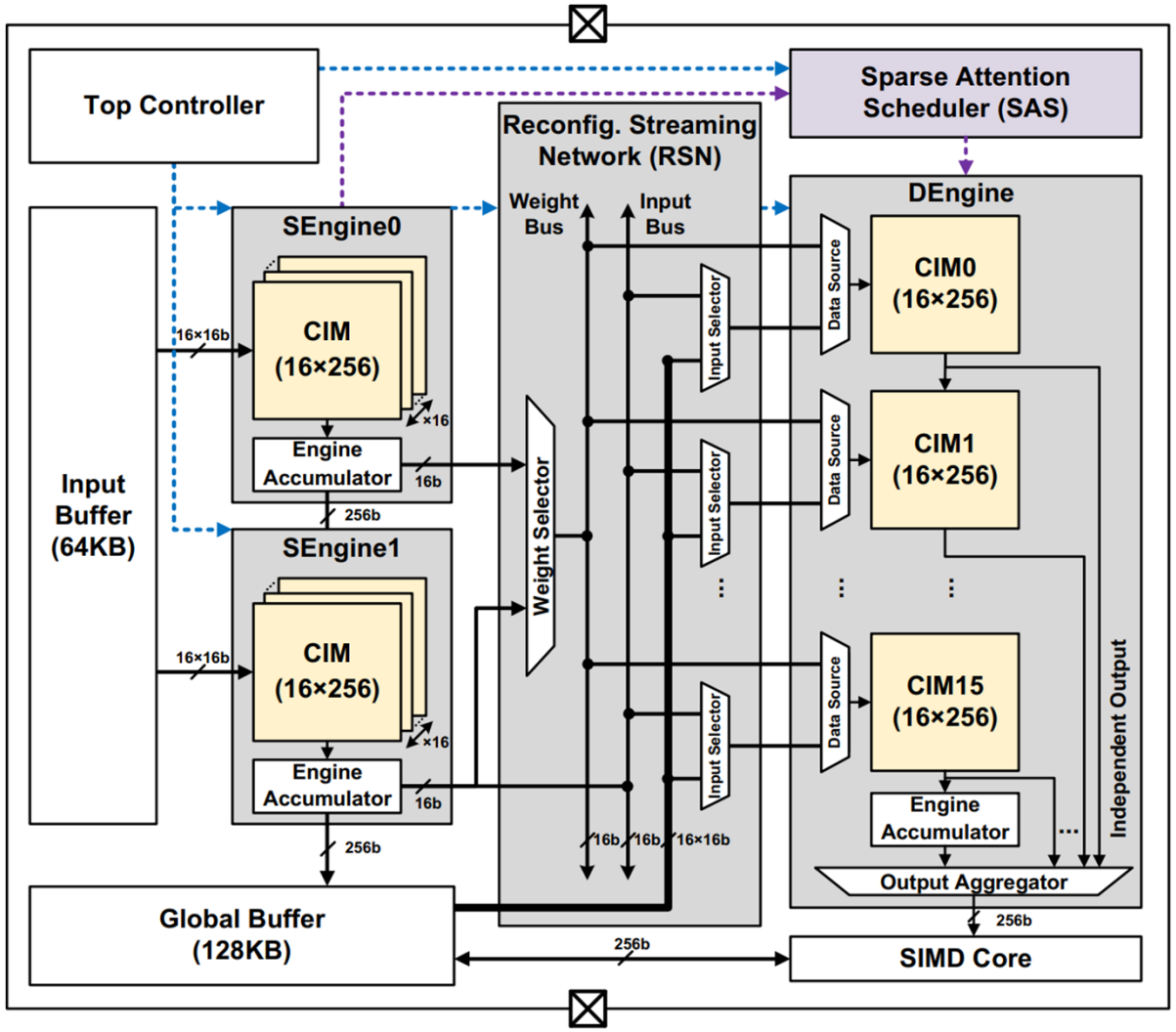
*Compute energy breakdown of BERT-base. Token Count N = 4096. Full attention.

- **Attention requires >8b precision and dominates computation**
  - Analog CIMs with ≤8b precision can't be directly used
  - Reducing attention computation is critical

Outline

- Motivation
- **TranCIM Accelerator Features**
  - Pipeline/Parallel Reconfigurable Modes
  - Bitline-Transpose-CIM (BLT-CIM)
  - Sparse Attention Scheduler (SAS)
- Measurement Results
- Conclusion

TranCIM (Transformer CIM) Accelerator



Feature1 Reconfigurable Architecture: Pipeline/Parallel Mode

Feature2 Bitline-Transpose-CIM Macro

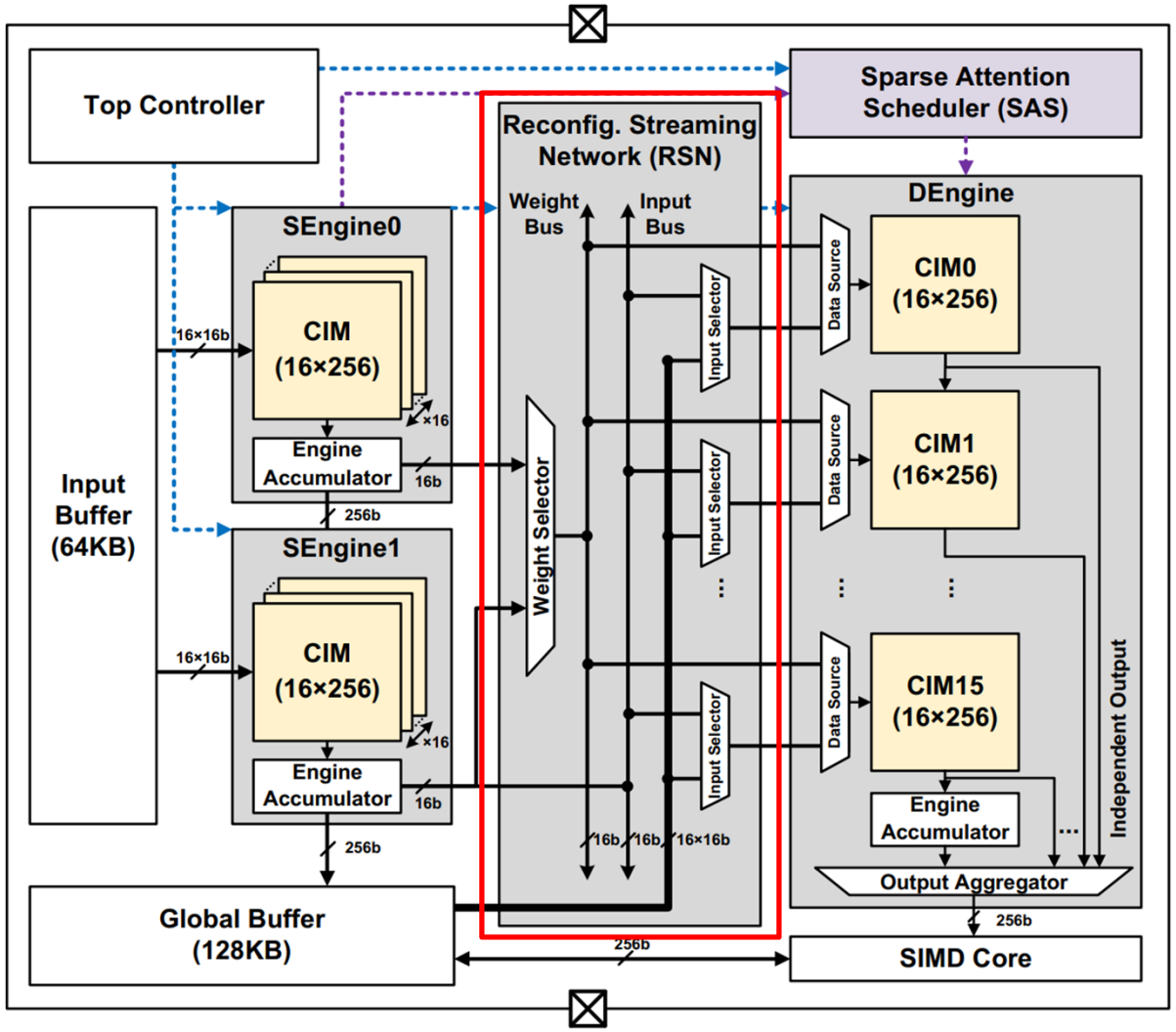
Feature3 Sparse Attention Scheduler

Mode Configuration
(Pipeline/Parallel Mode, Layer Parameters, Sparse Attention Pattern)

Workload Configuration
(Sparse Attention Workload Scheduling in the Pipeline Mode)

For Challenge1a

Reconfigurable Streaming Network (RSN)



Feature1 Reconfigurable Architecture: Pipeline/Parallel Mode

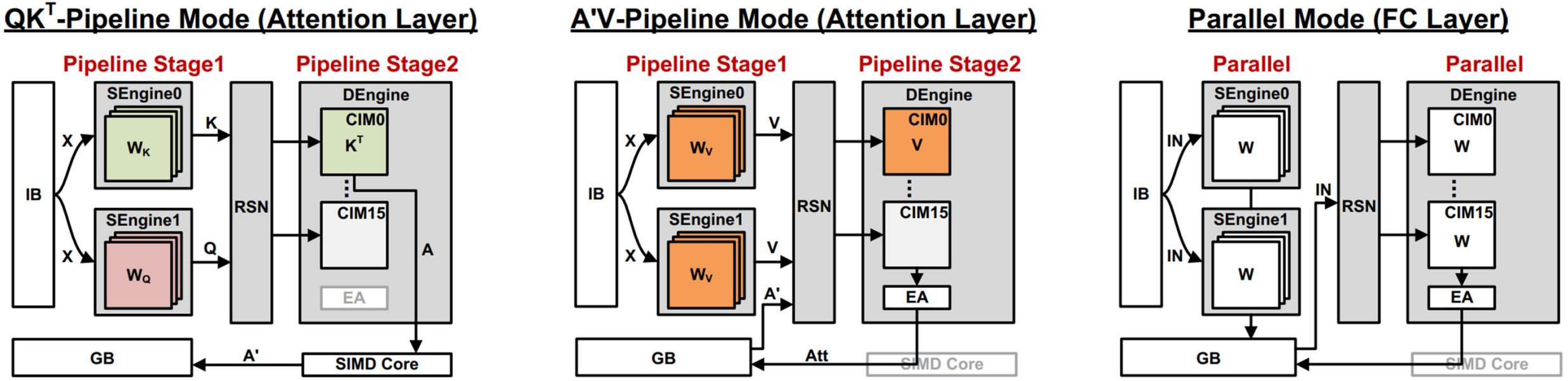
Feature2 Bitline-Transpose-CIM Macro

Feature3 Sparse Attention Scheduler

Mode Configuration
(Pipeline/Parallel Mode, Layer Parameters, Sparse Attention Pattern)

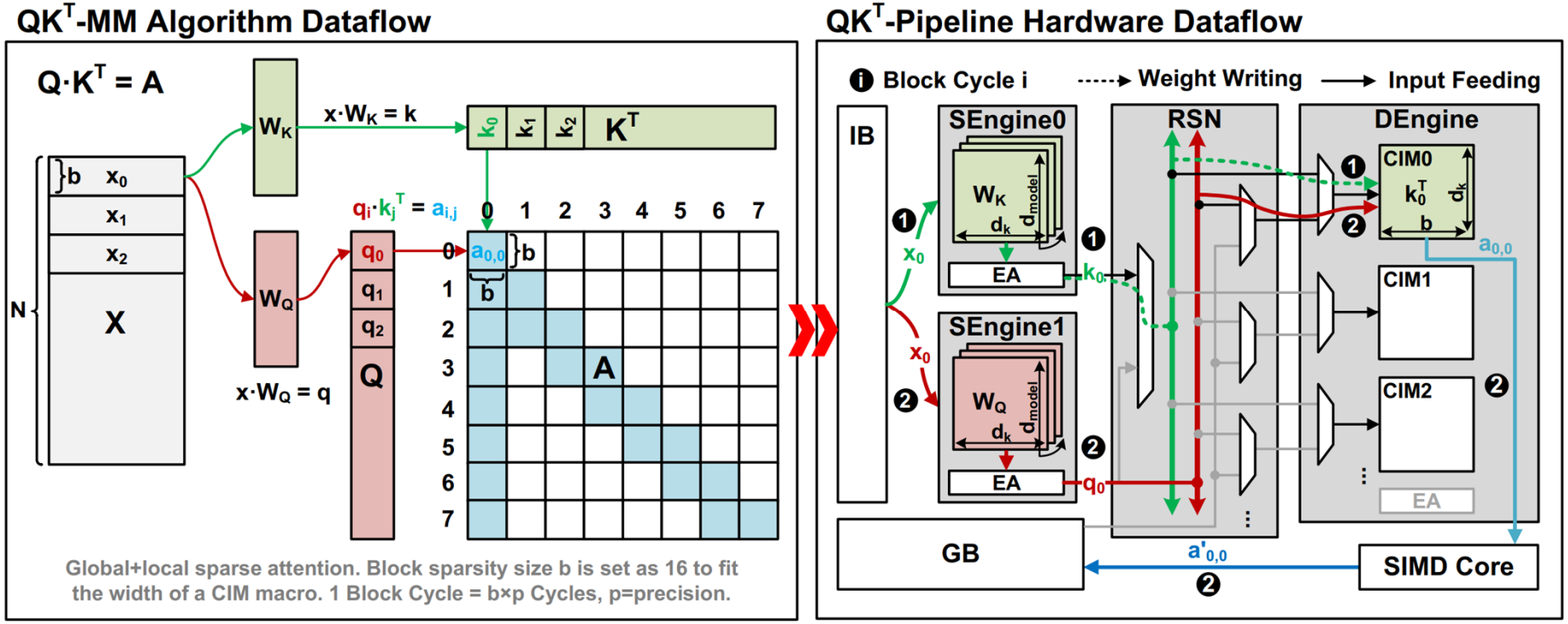
Workload Configuration
(Sparse Attention Workload Scheduling in the Pipeline Mode)

Pipeline/Parallel Reconfigurable Modes



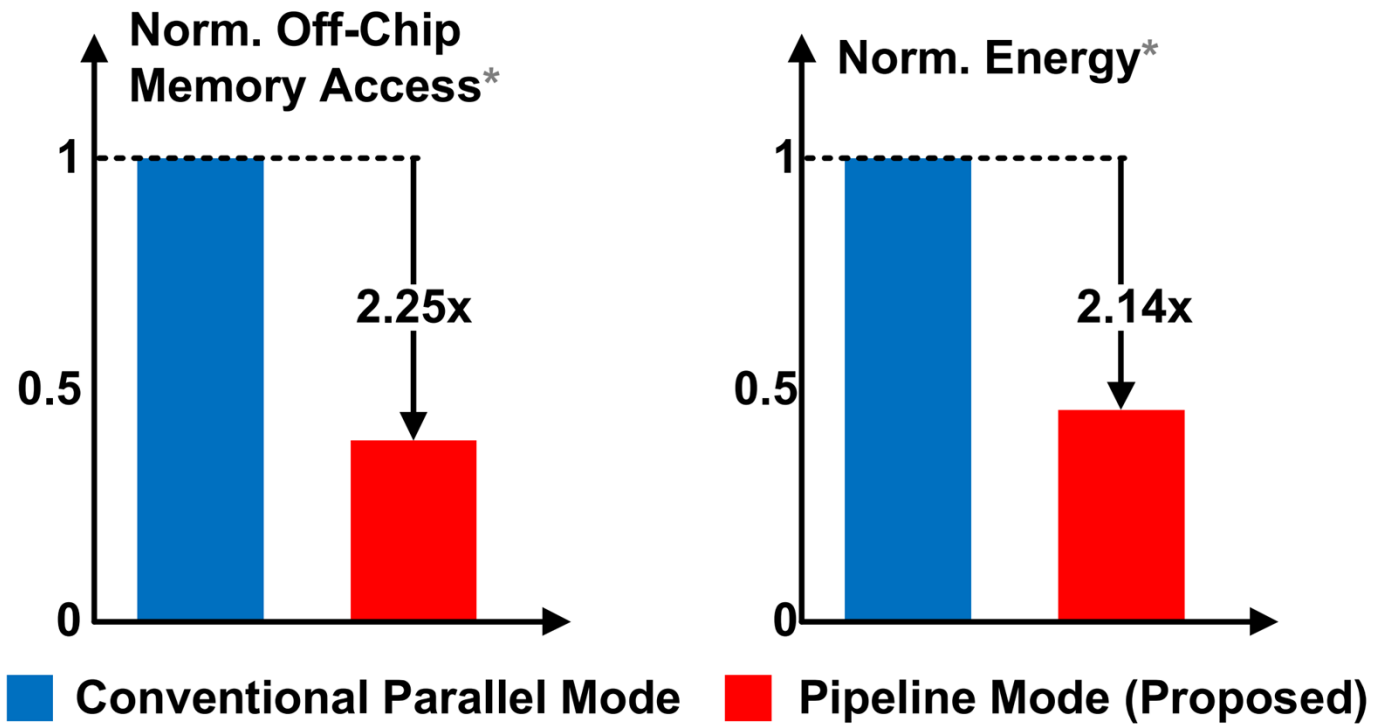
- **TranCIM's reconfigurable architecture**
- **Static Engine (SEngine), Dynamic Engine (DEngine)**
  - Attention layer: Pipeline mode
  - FC layer: Parallel mode (similar to previous CIM accelerators)

QK^T-Pipeline Example



- **1st stage: SEngine0, SEngine1**
  - $Q=XW_Q$, $K=XW_K$
- **2nd stage: DEngine**
  - $A=QK^T$

Parallel Mode vs. Pipeline Mode (Ours)

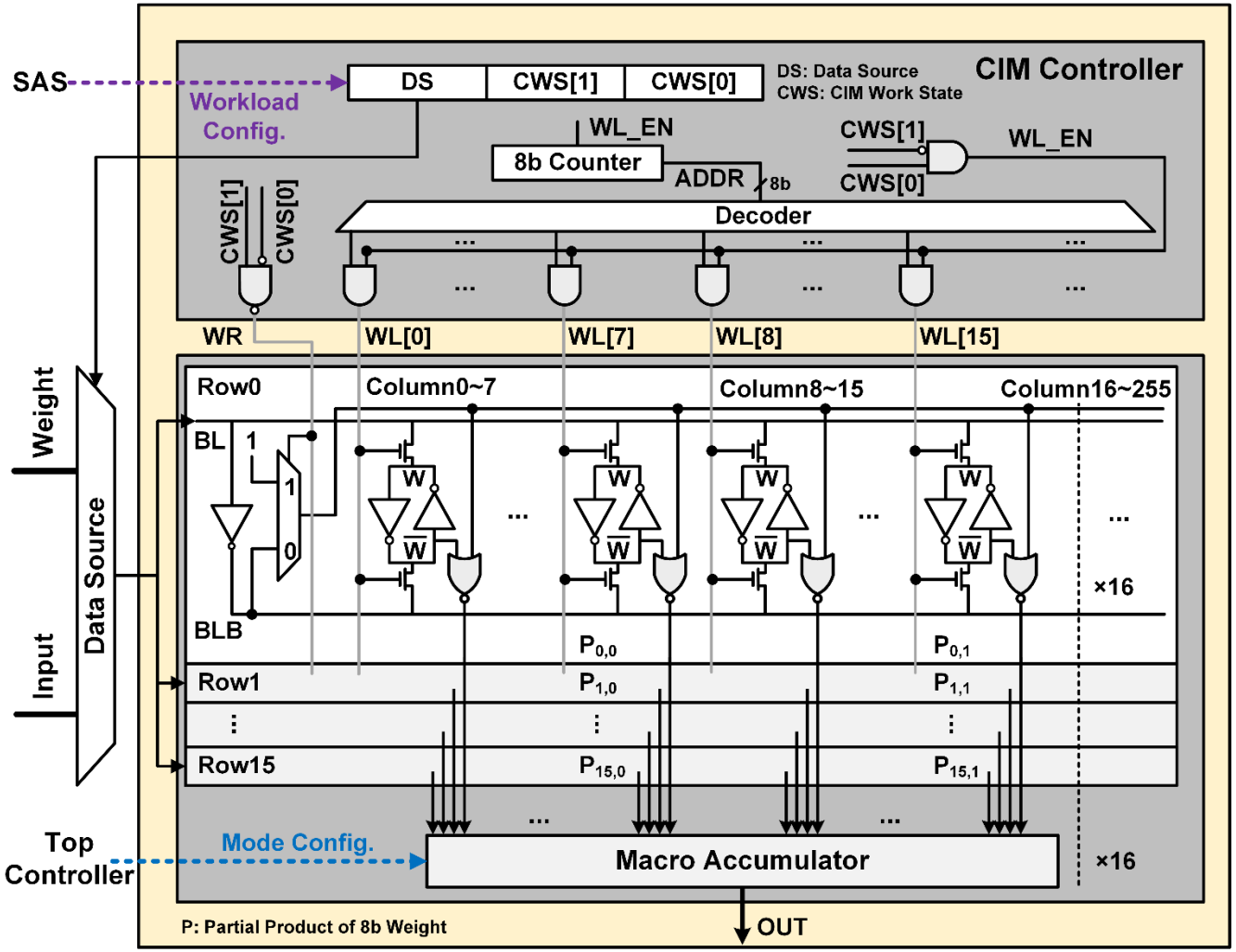


*QK^T-MM Parameters: INT16, N=1024, d_{model}=256, d_k=16, b=16. Evaluated at 1.0V, 240MHz. Off-chip DDR3 memory is included.

- **Load X only once, and directly reuse Q, K on chip**
  - 2.25x less off-chip memory access
  - 2.14x energy saving

For Challenge1b

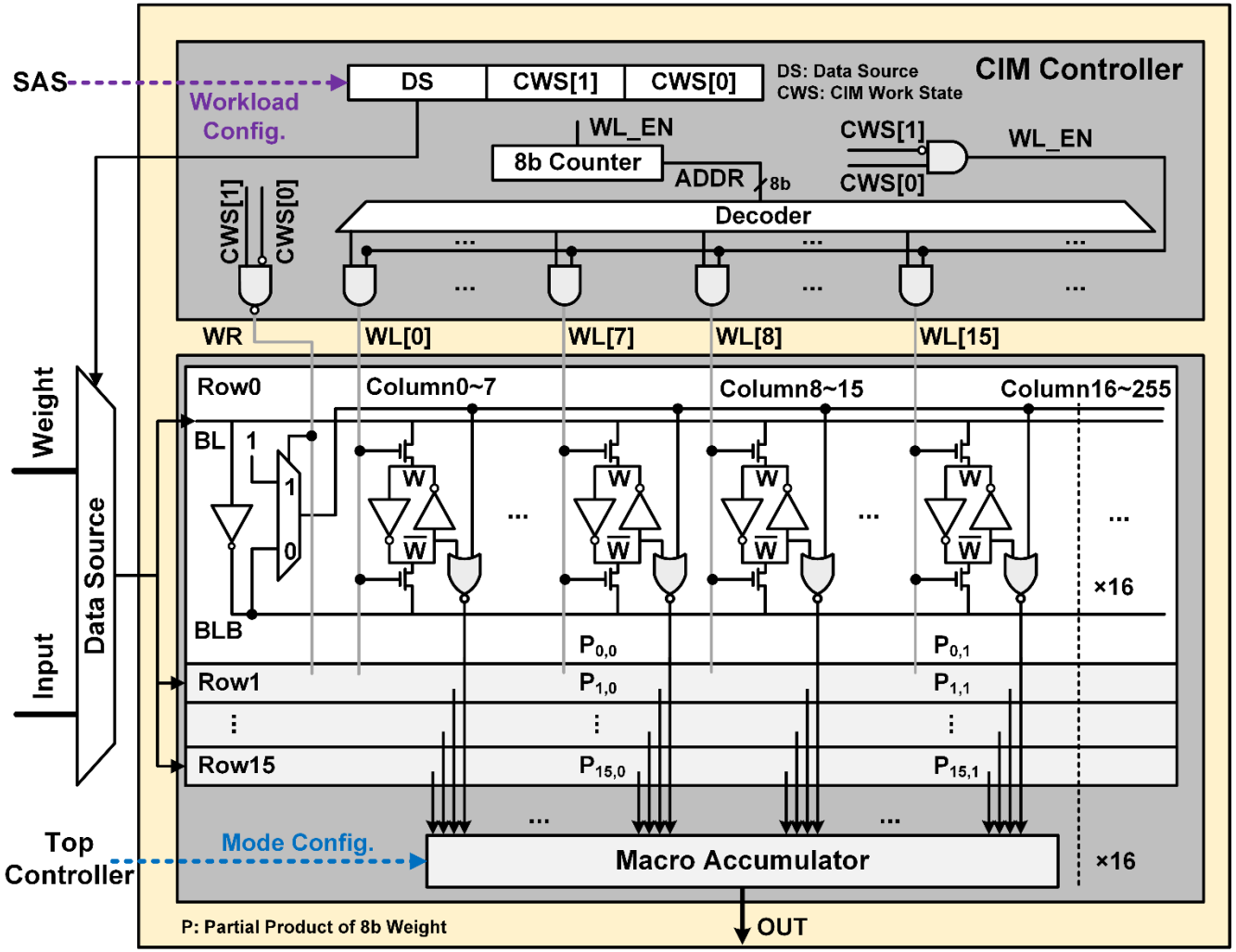
Bitline-Transpose-CIM (BLT-CIM)



- **BLT-CIM macro**
  - CIM controller
  - SRAM-CIM

For Challenge1b

Bitline-Transpose-CIM (BLT-CIM)



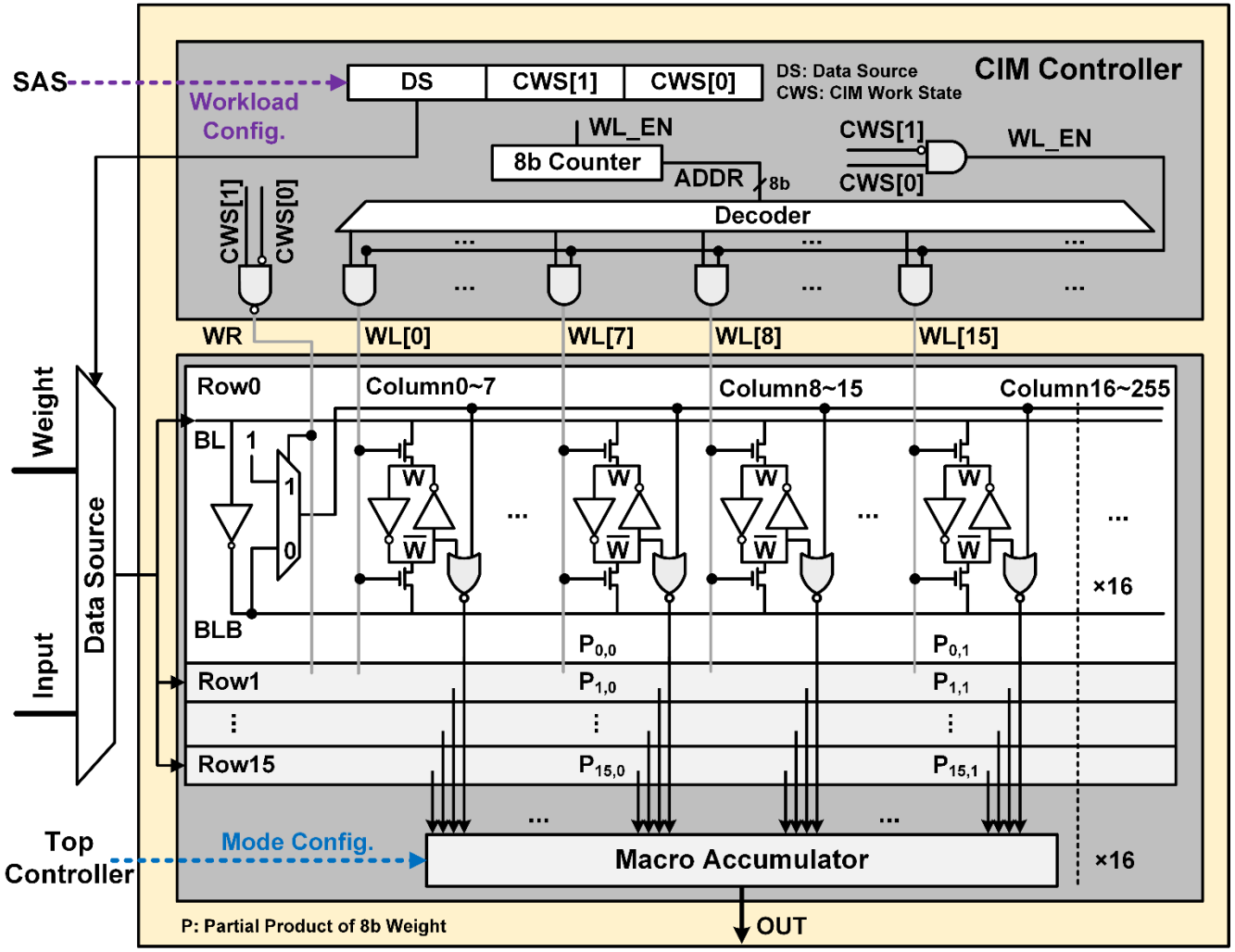
CIM controller

- Data source
- CIM work state

Indicate whether to use the current data for input feeding or weight writing

For Challenge1b

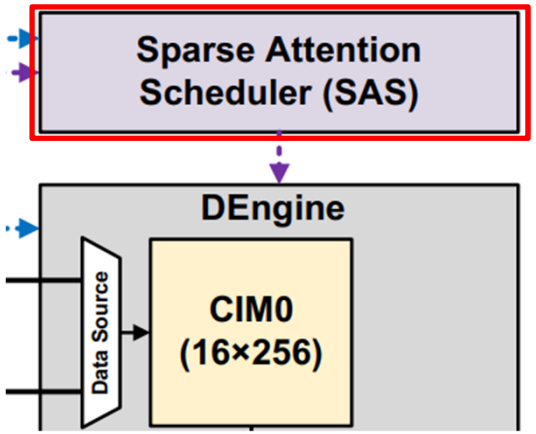
Bitline-Transpose-CIM (BLT-CIM)



CIM controller

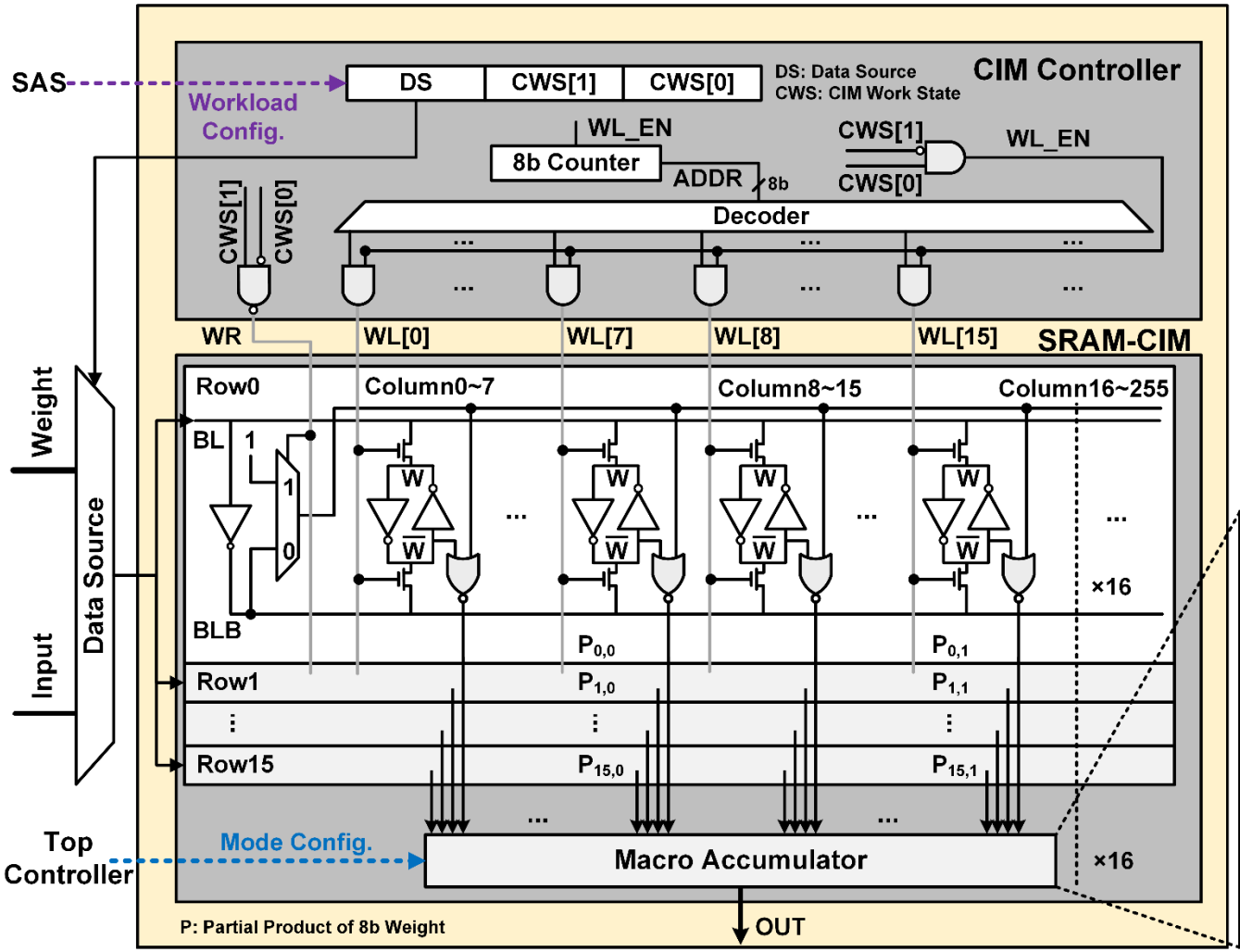
- Data source
- CIM work state

DEngine's CIM controller receives dynamic workload configurations from SAS in the pipeline mode.



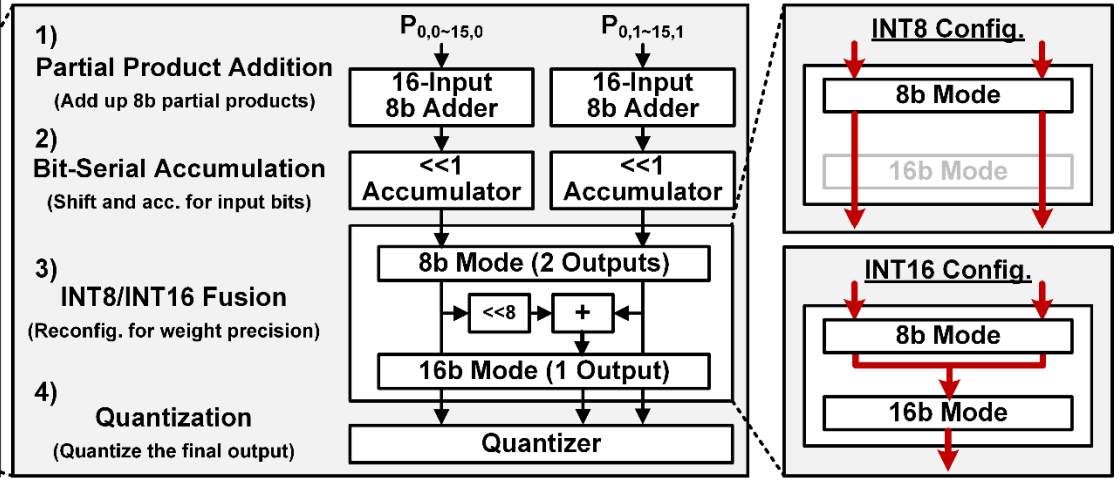
For Challenge1b

Bitline-Transpose-CIM (BLT-CIM)

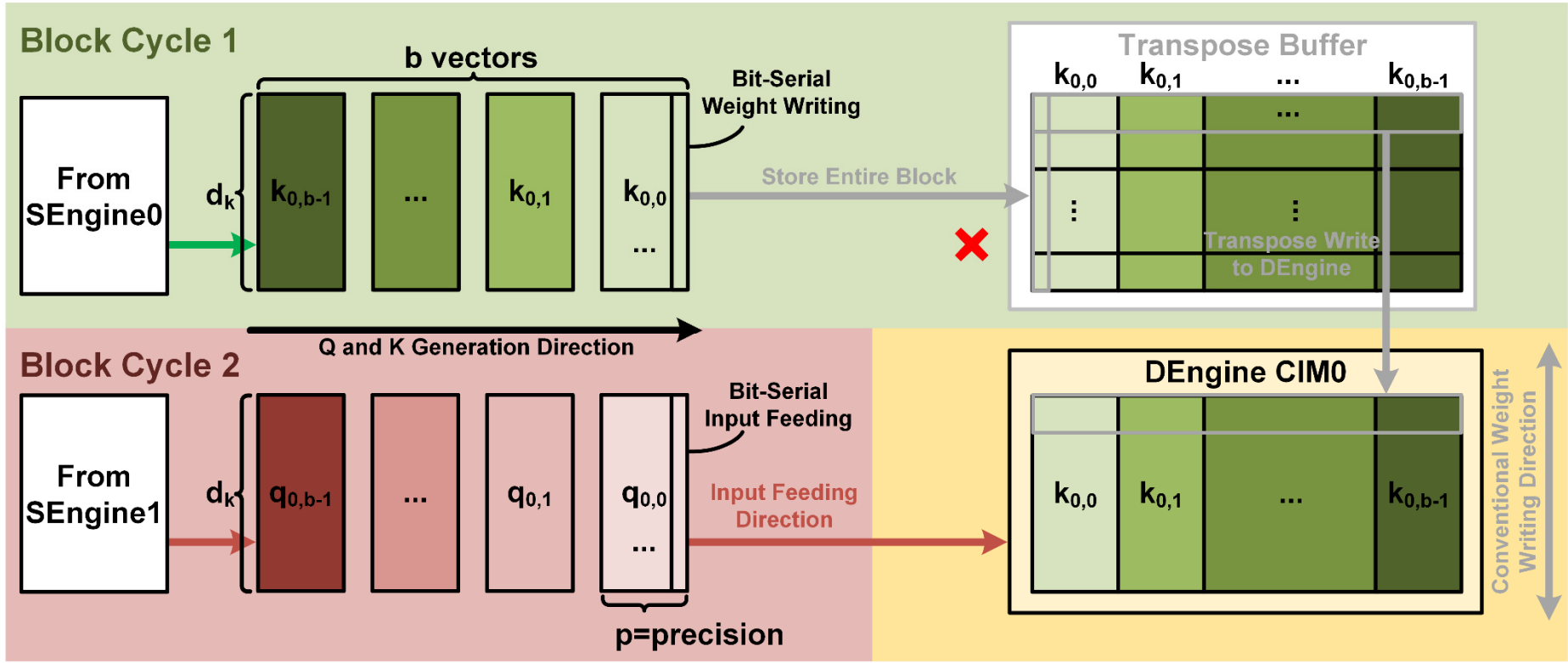


■ SRAM-CIM

- Full-digital CIM design
- 4-stage macro accumulator
- INT8/INT16, no accuracy loss

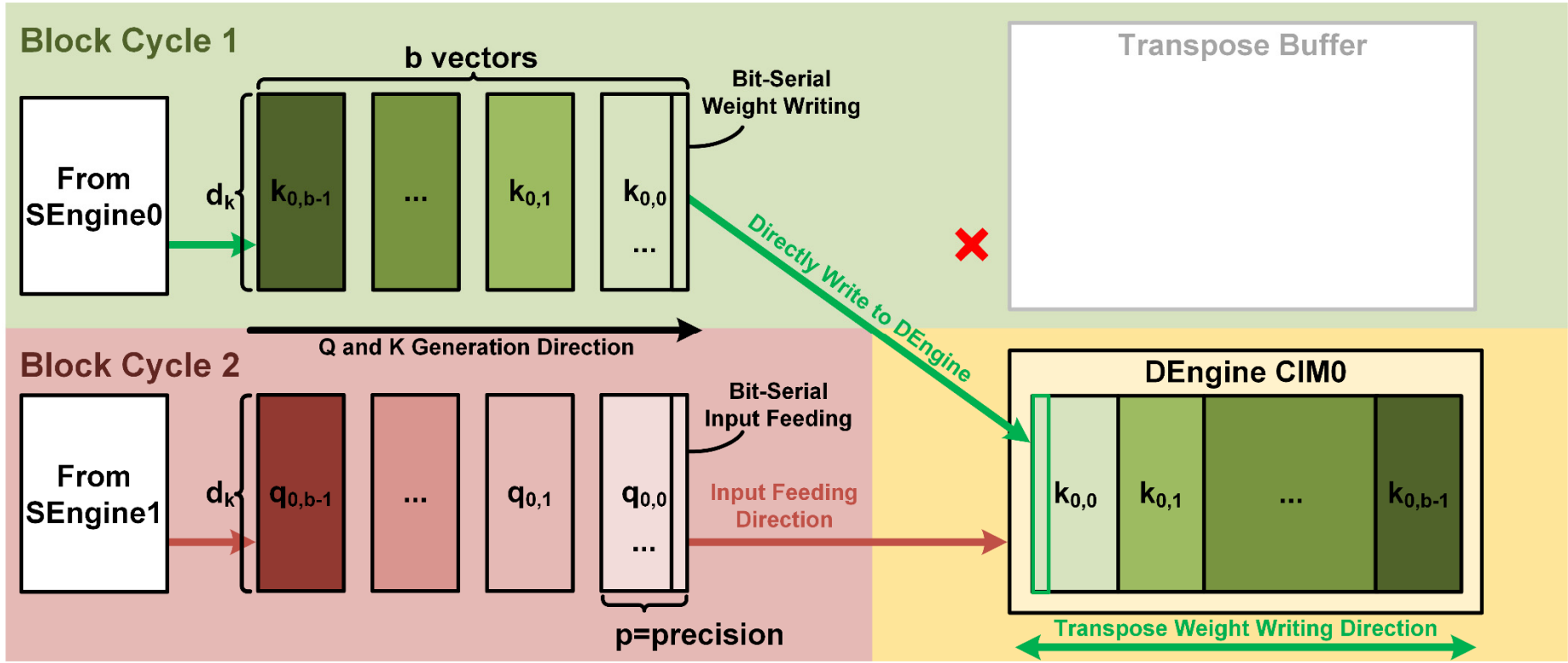


Conventional CIM + Transpose Buffer



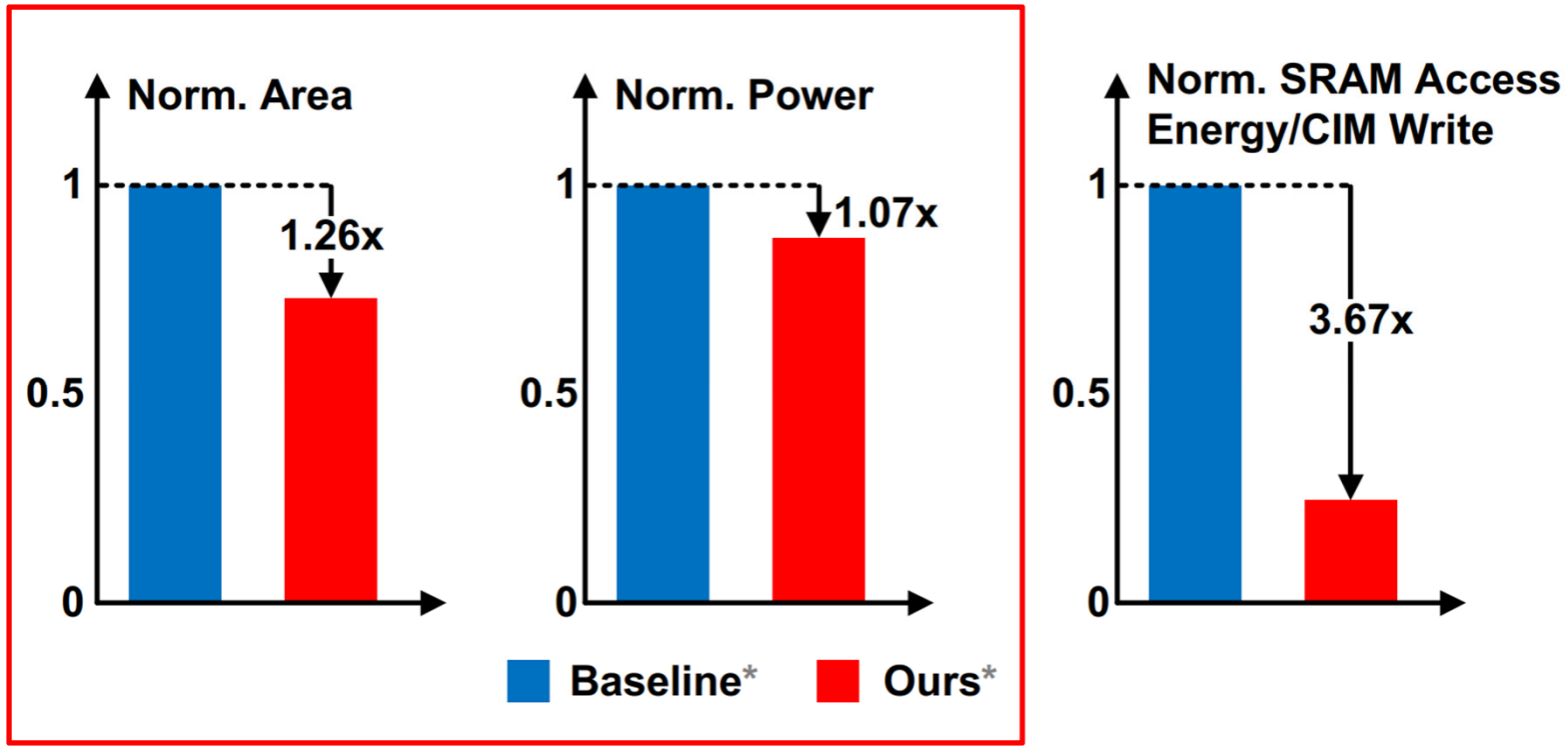
- **QK^T demands DEngine store K in a transpose way**
  - An extra transpose buffer is needed to change the K writing direction

Bitline-Transpose-CIM (BLT-CIM)



- **BLT-CIM uses bitlines to receive inputs for computation**
  - Directly write K to DEngine
  - Multiply QK^T without pipeline stall

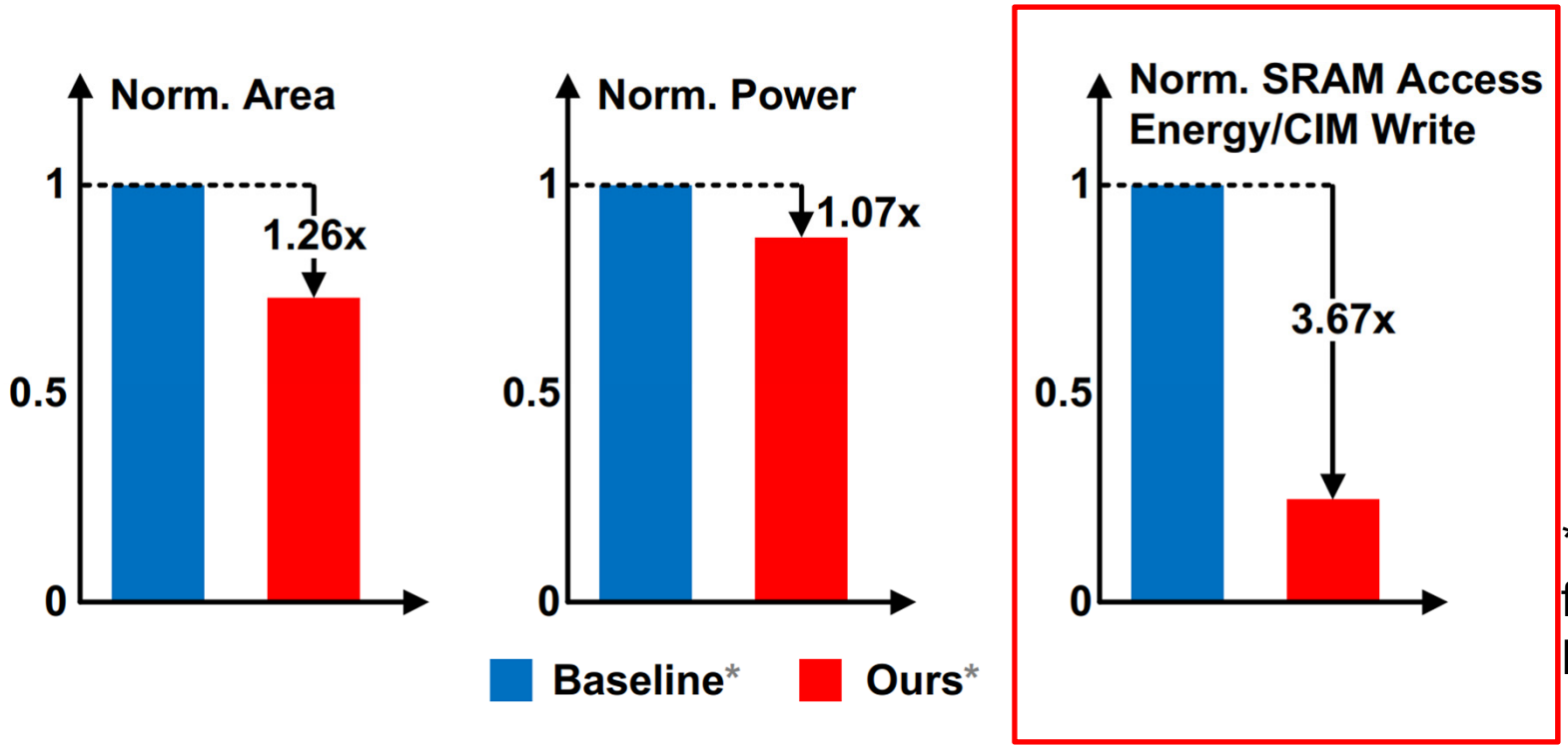
Conventional CIM+Transpose Buffer vs. BLT-CIM



*Baseline: 8KB wordline-direction-feeding CIM+8KB transpose buffer. Evaluated at 28nm, 1.0V, 240MHz.

- No need for a transpose buffer
- Save area by 1.26x and power by 1.07x

Conventional CIM+Transpose Buffer vs. BLT-CIM

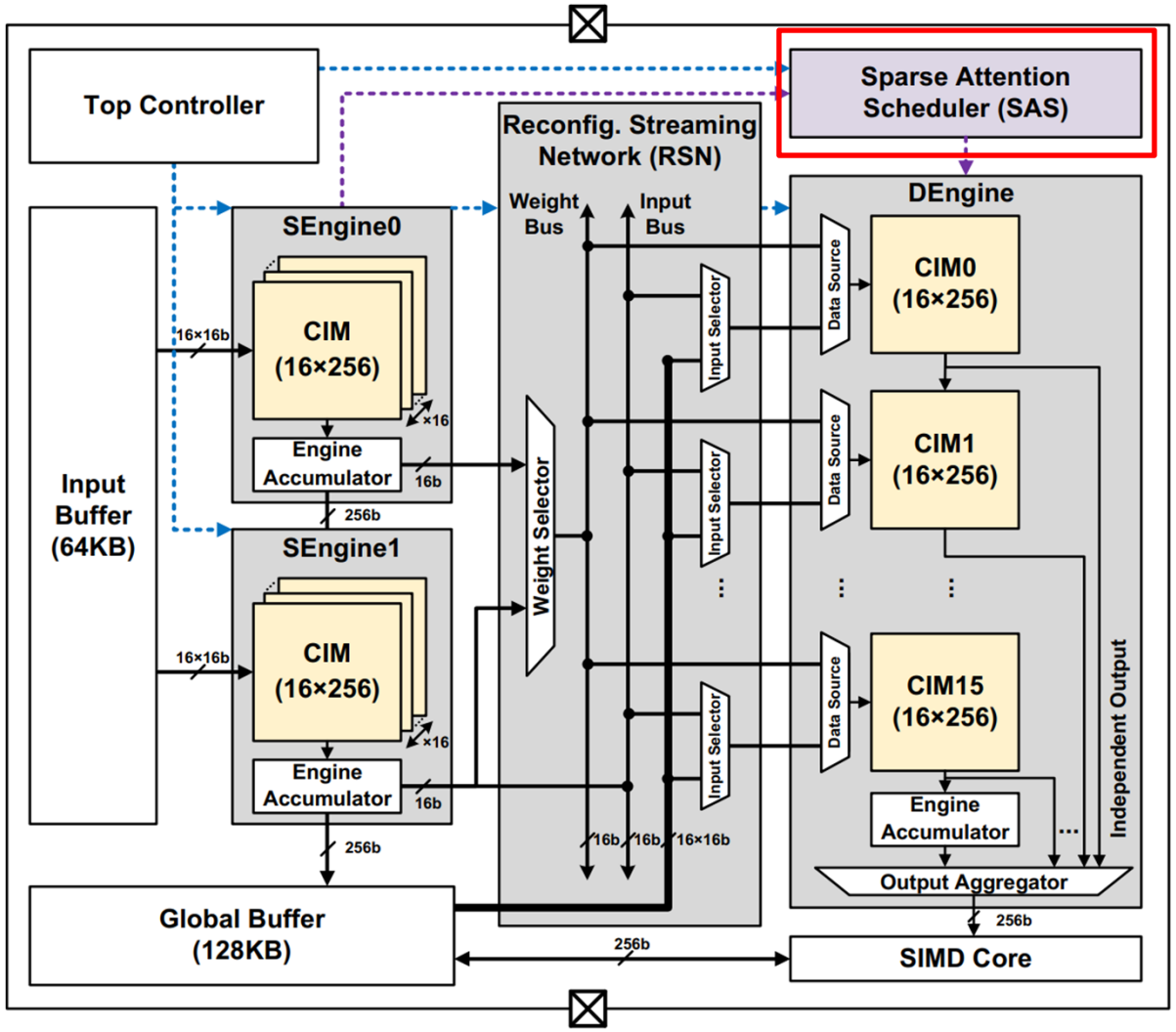


*Baseline: 8KB wordline-direction-feeding CIM+8KB transpose buffer. Evaluated at 28nm, 1.0V, 240MHz.

- No need to access a transpose buffer
- Reduce 3.67x SRAM access energy for each CIM write

For Challenge2

Sparse Attention Scheduler (SAS)



Feature1 Reconfigurable Architecture: Pipeline/Parallel Mode

Feature2 Bitline-Transpose-CIM Macro

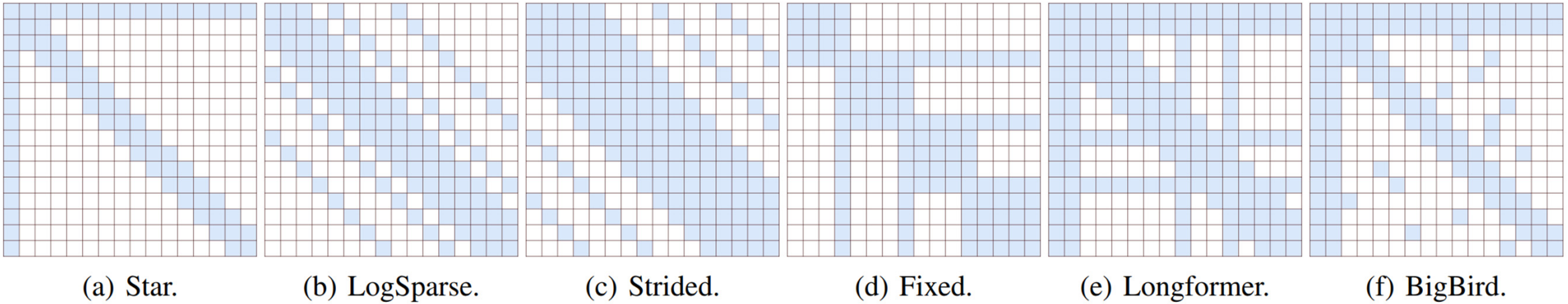
Feature3 Sparse Attention Scheduler

Mode Configuration
(Pipeline/Parallel Mode, Layer Parameters, Sparse Attention Pattern)

Workload Configuration
(Sparse Attention Workload Scheduling in the Pipeline Mode)

For Challenge2

Sparse Transformer Algorithm

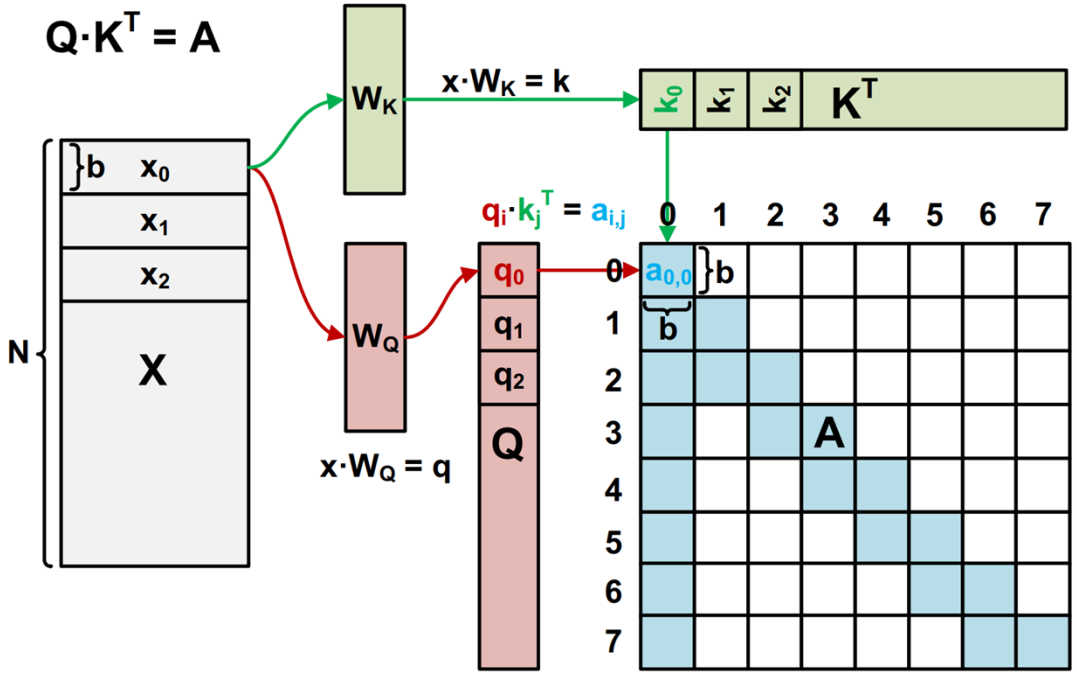


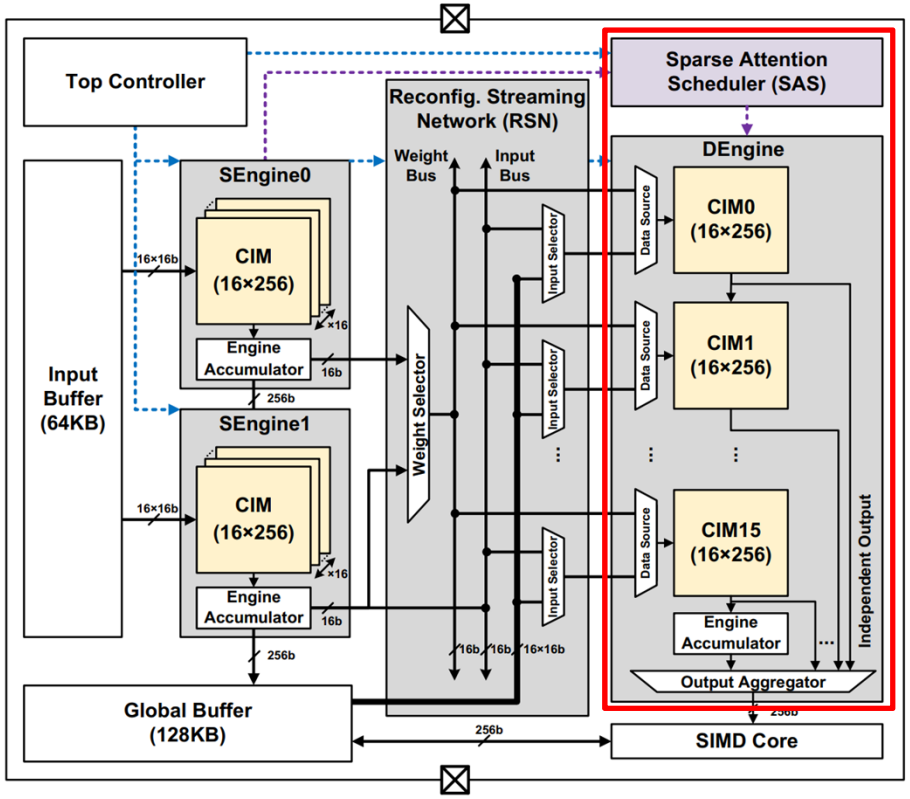
*[ICML'21] SparseBERT: Rethinking the Importance Analysis in Self-attention

- **Attention dominates Transformer computation**
- **Full attention complexity $O(N^2)$, N = Token Count**
- **Sparse attention complexity O(N)**

For Challenge2

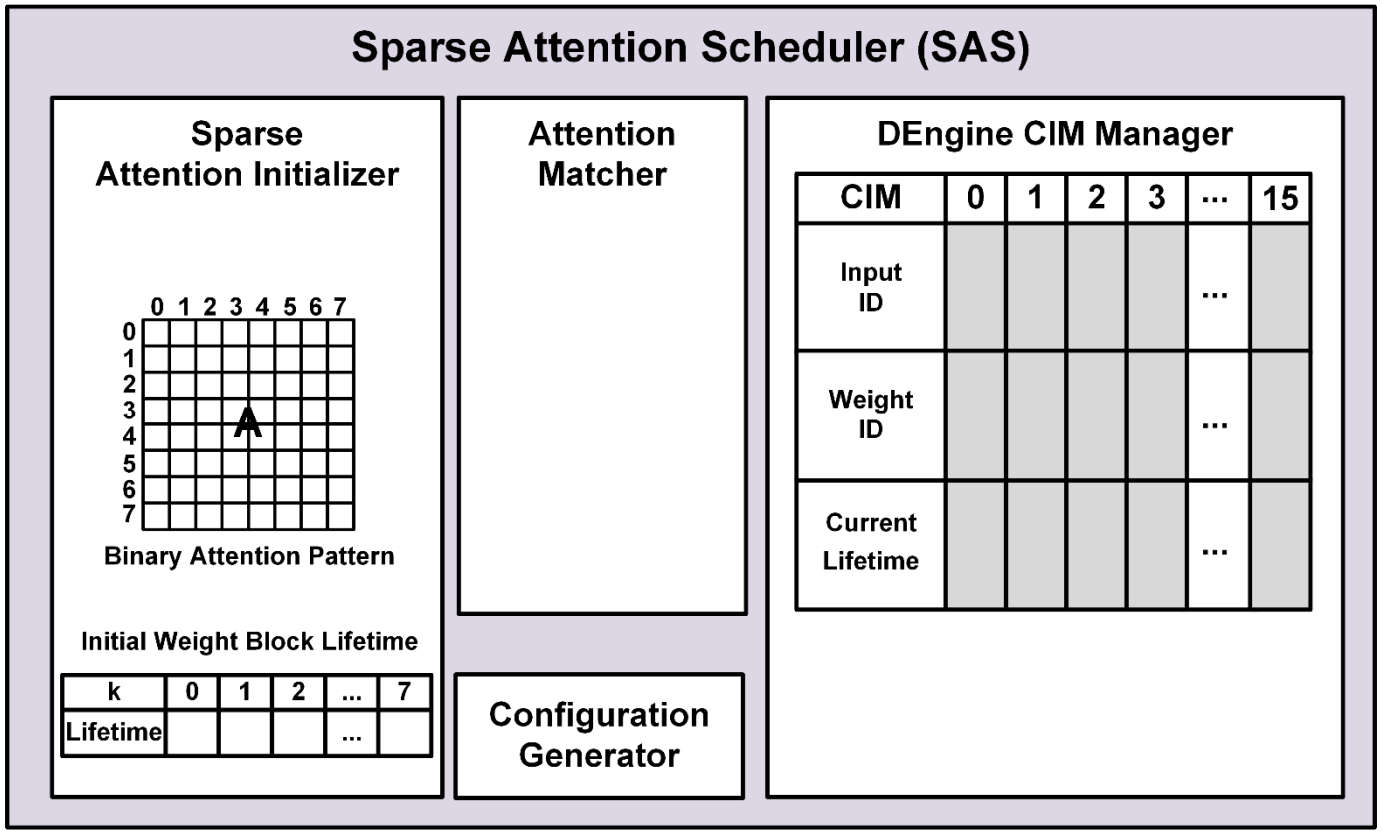
Sparse Attention Scheduler (SAS)





- Sparse attention makes multiplication of various qk-pairs
- DEngine's CIM workload always changes during runtime
- SAS enables dynamic reconfiguration for DEngine

Sparse Attention Scheduler (SAS)

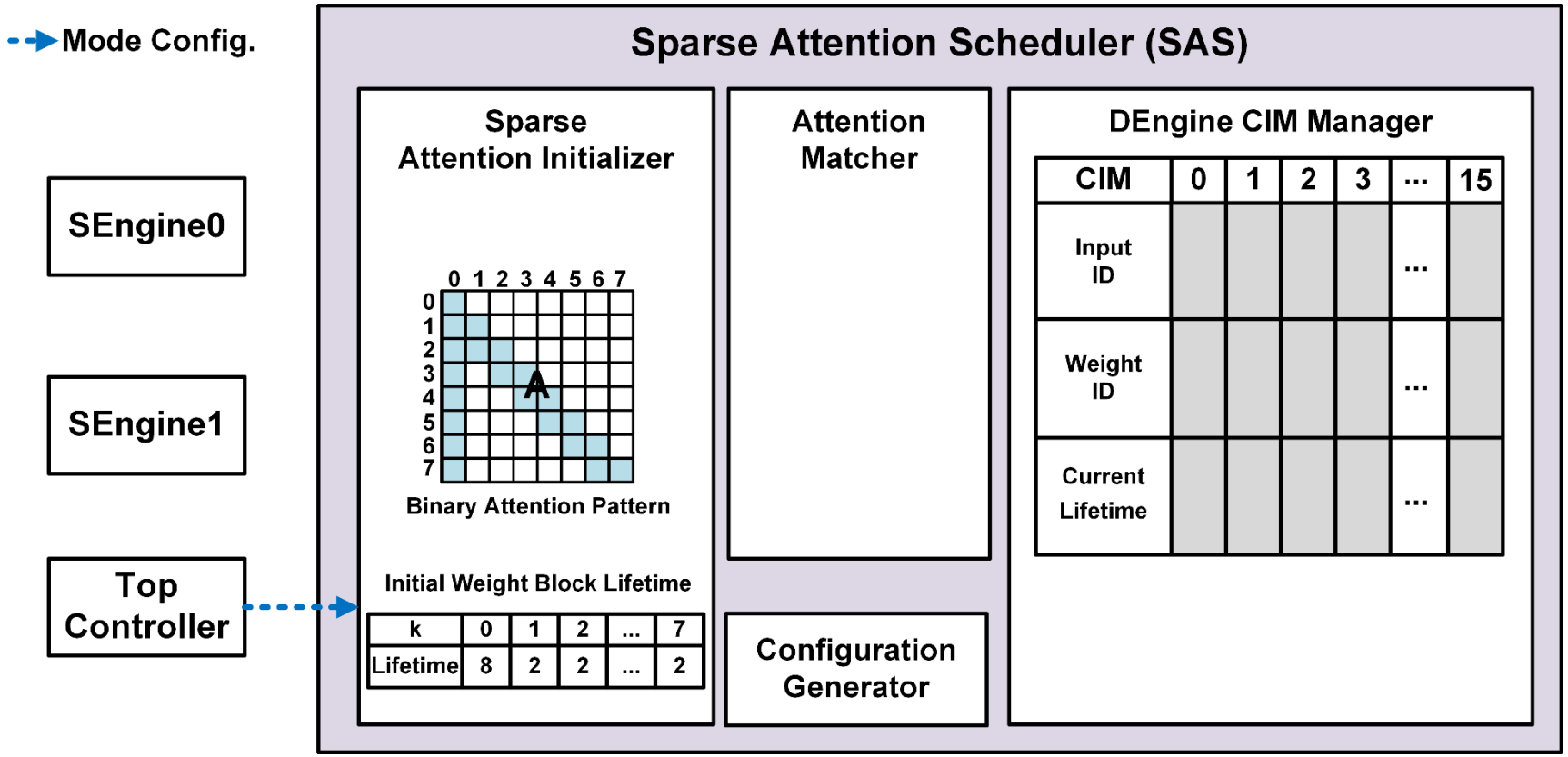


■ SAS architecture

- Sparse attention initializer, attention matcher, DEngine CIM manager, and configuration generator

For Challenge2

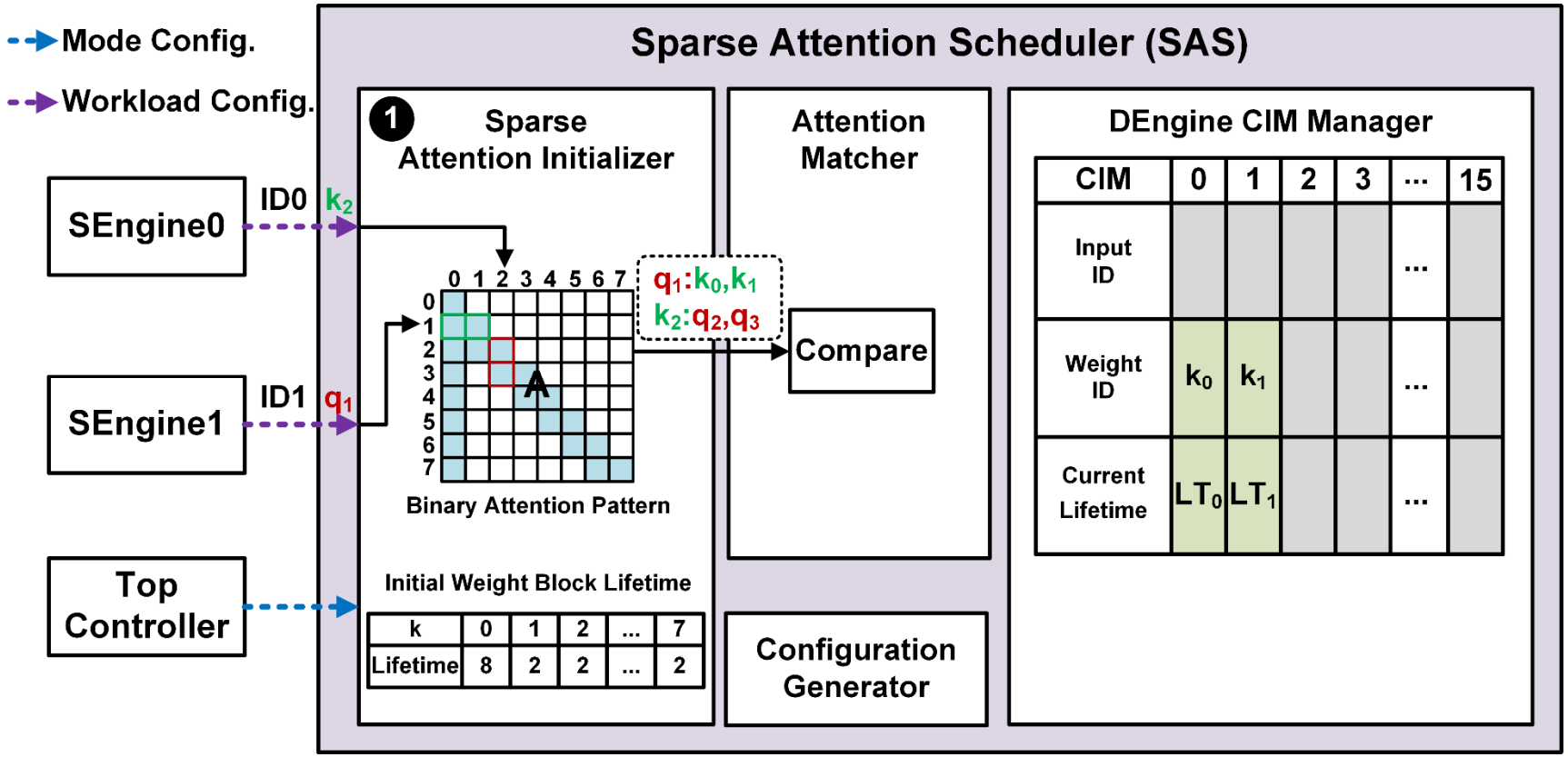
SAS: Sparse Attention Initializer

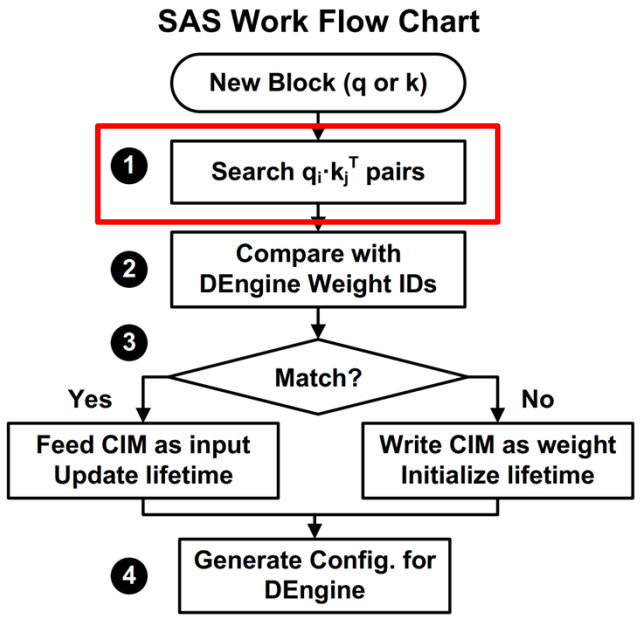


- **Mode configuration phase**
  - Binary attention pattern
  - Weight block lifetime

For Challenge2

SAS: Sparse Attention Initializer

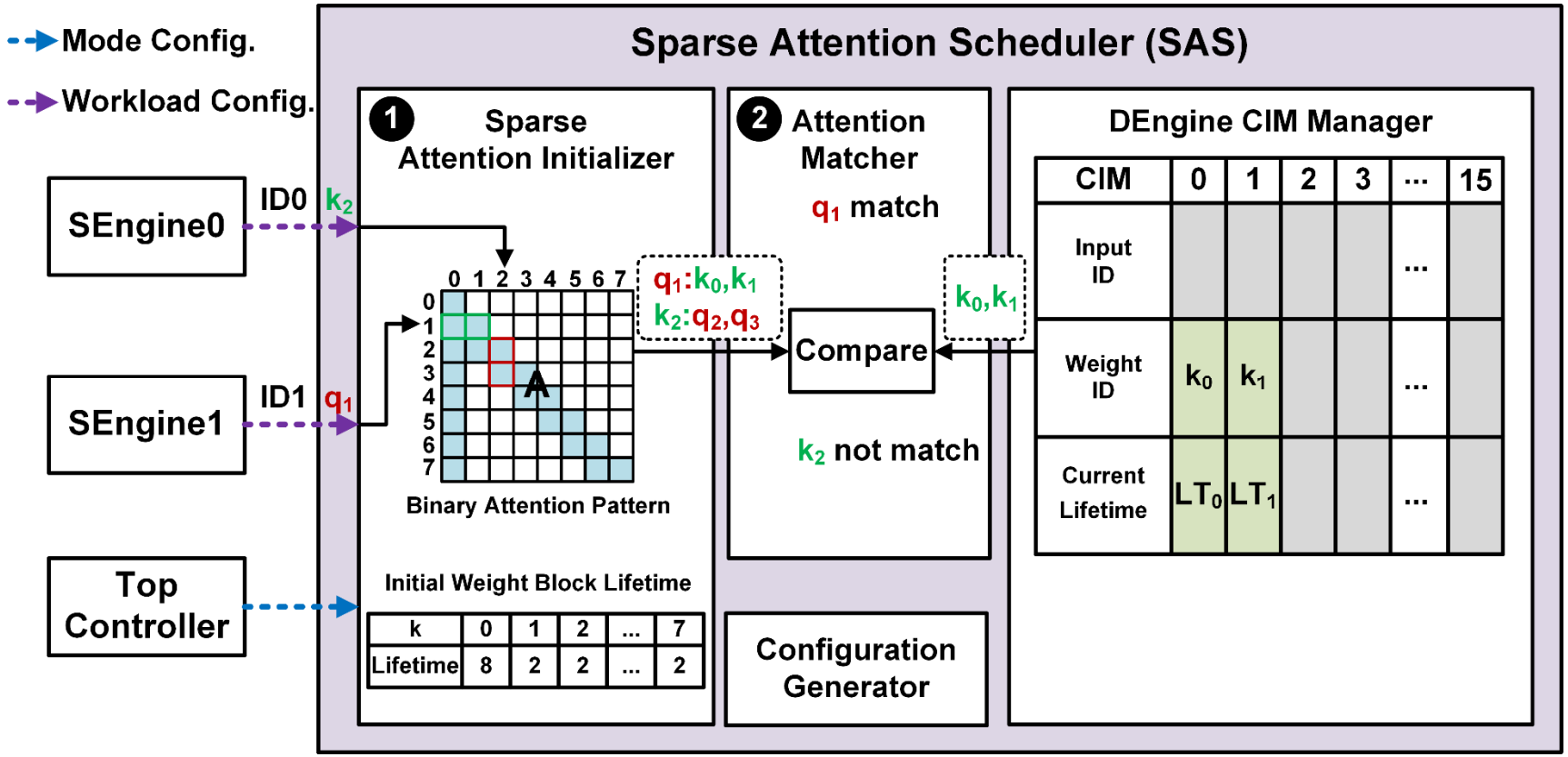


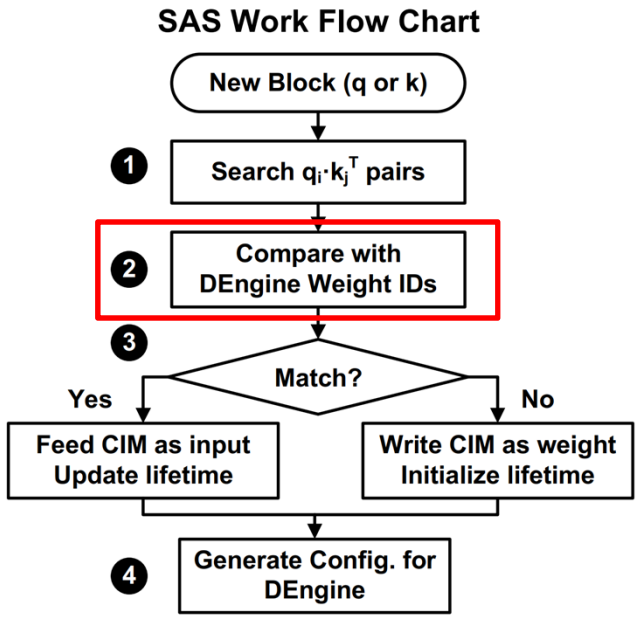


- Once SEngine generates a new block, the sparse attention initializer searches the corresponding qk-pairs

For Challenge2

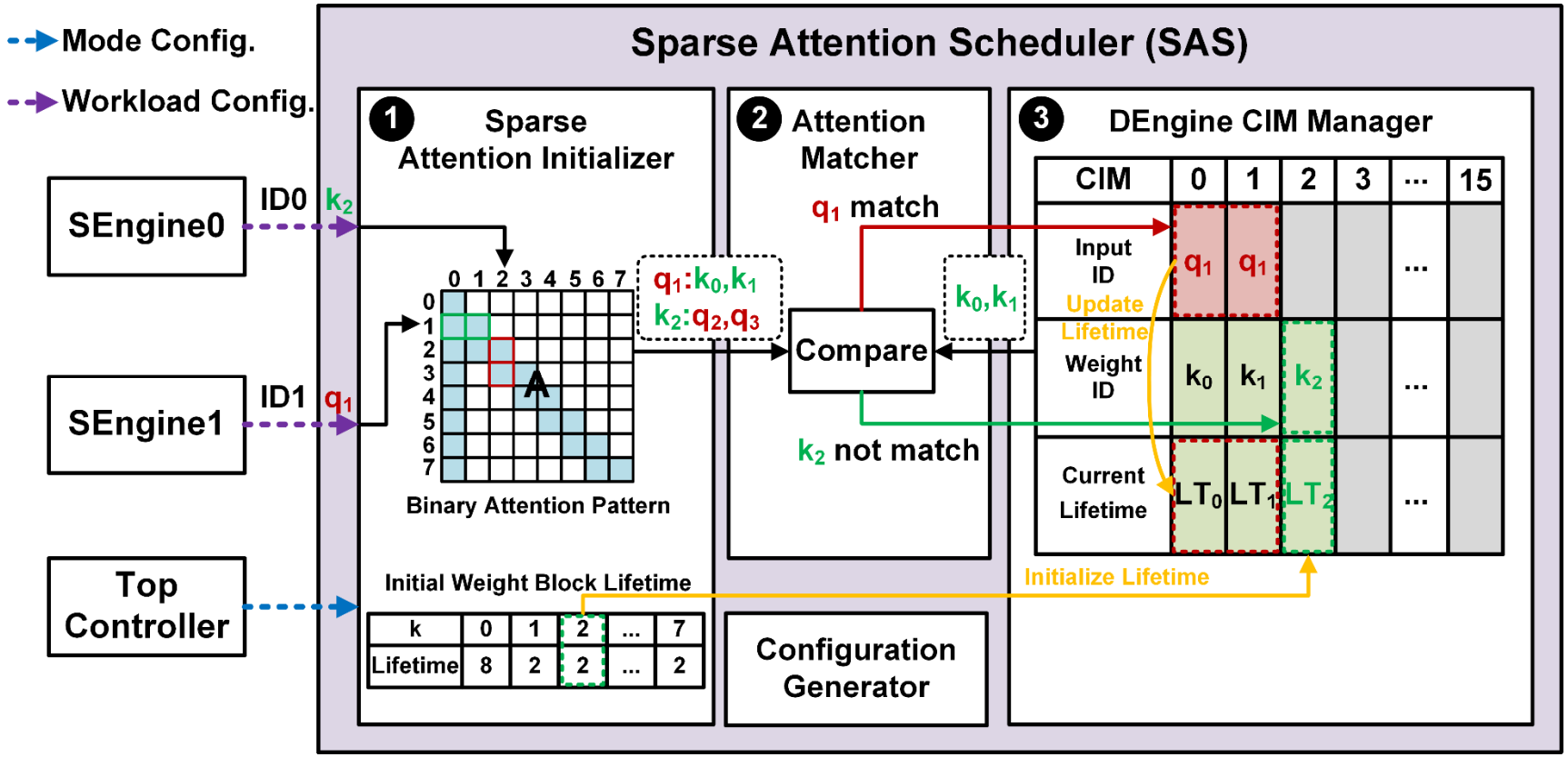
SAS: Attention Matcher

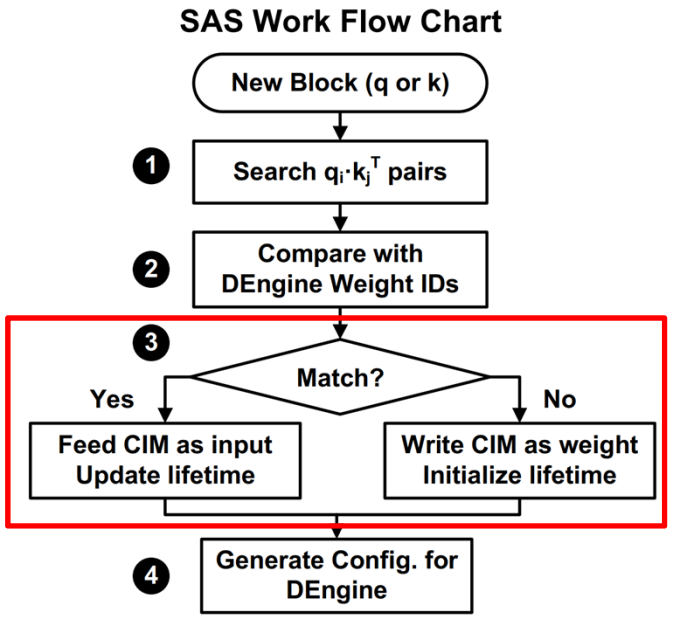




- Compare the attended blocks with the weight blocks already stored in DEngine

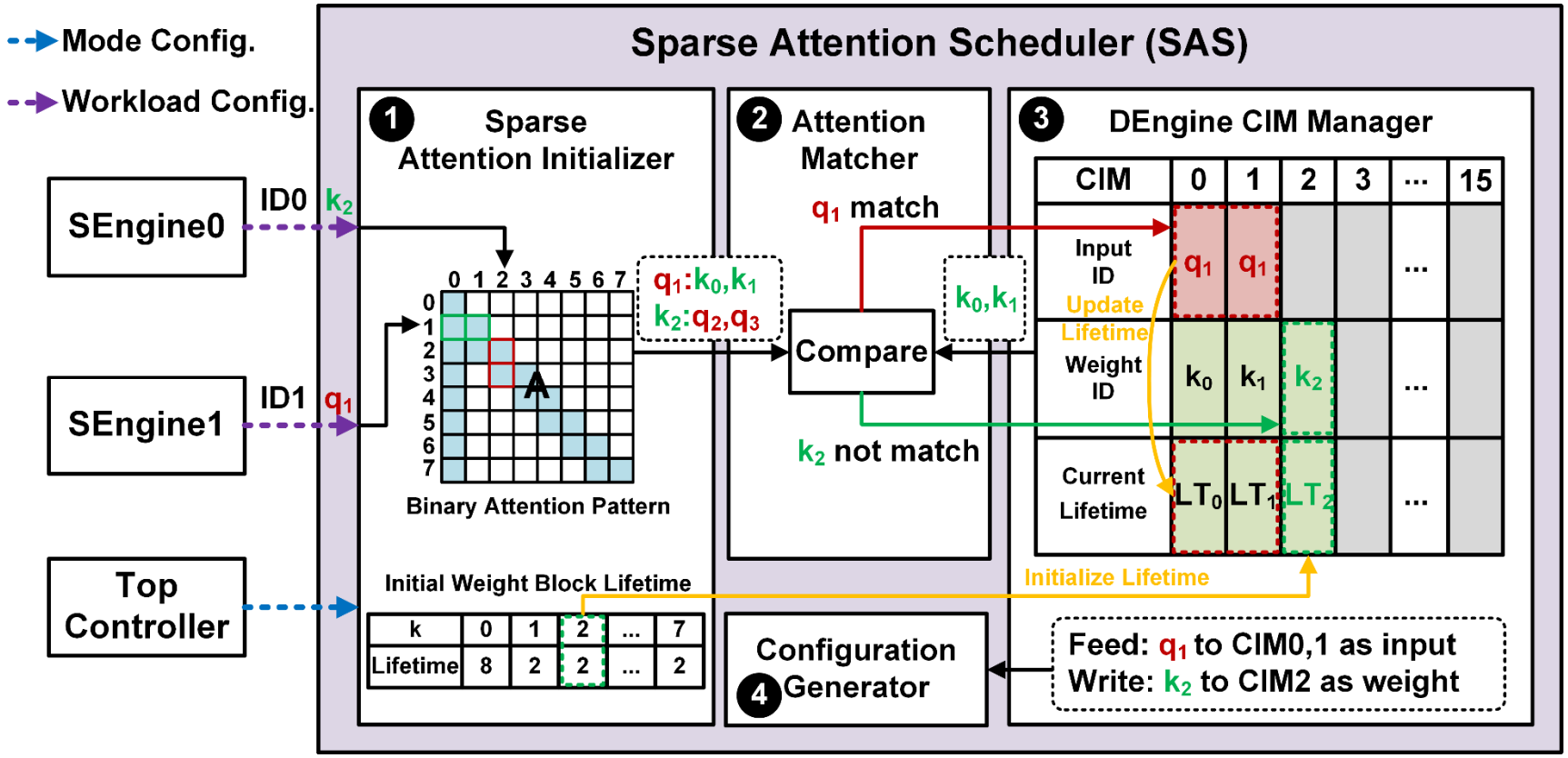
SAS: DEngine CIM Manager

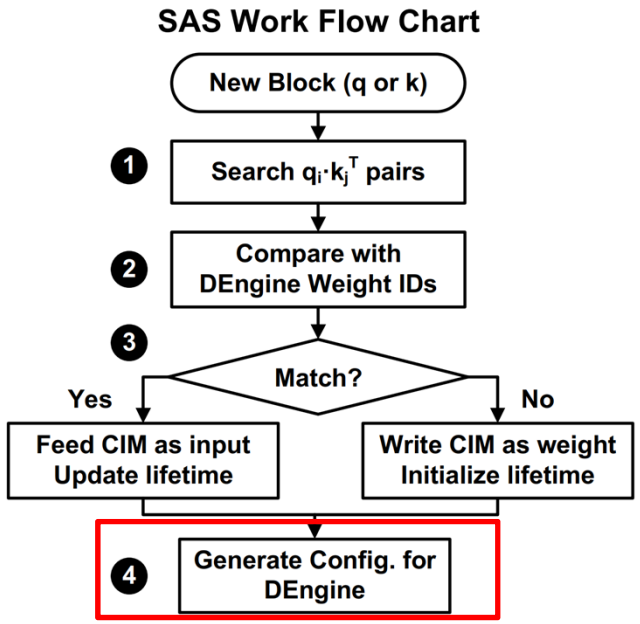




- **Maintain workload information**
  - Input ID, Weight ID, and Current Lifetime

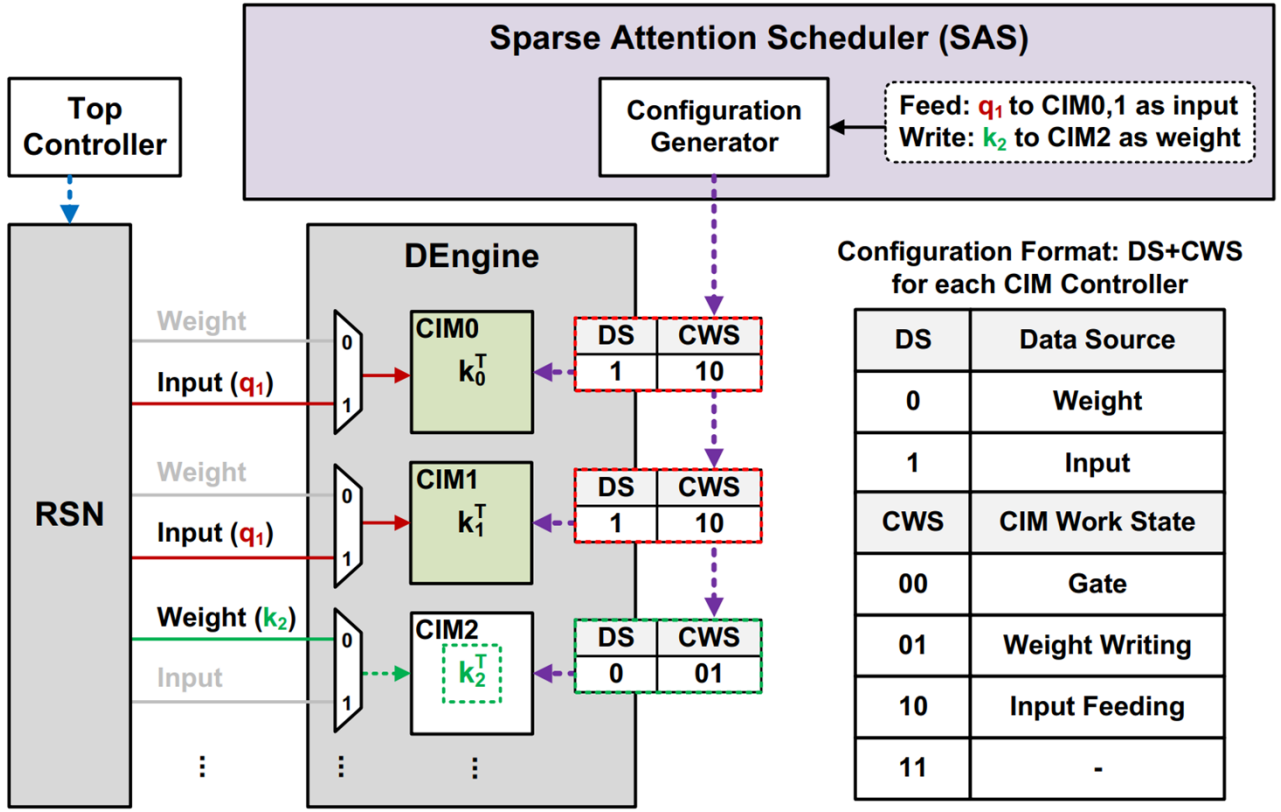
SAS: Configuration Generator





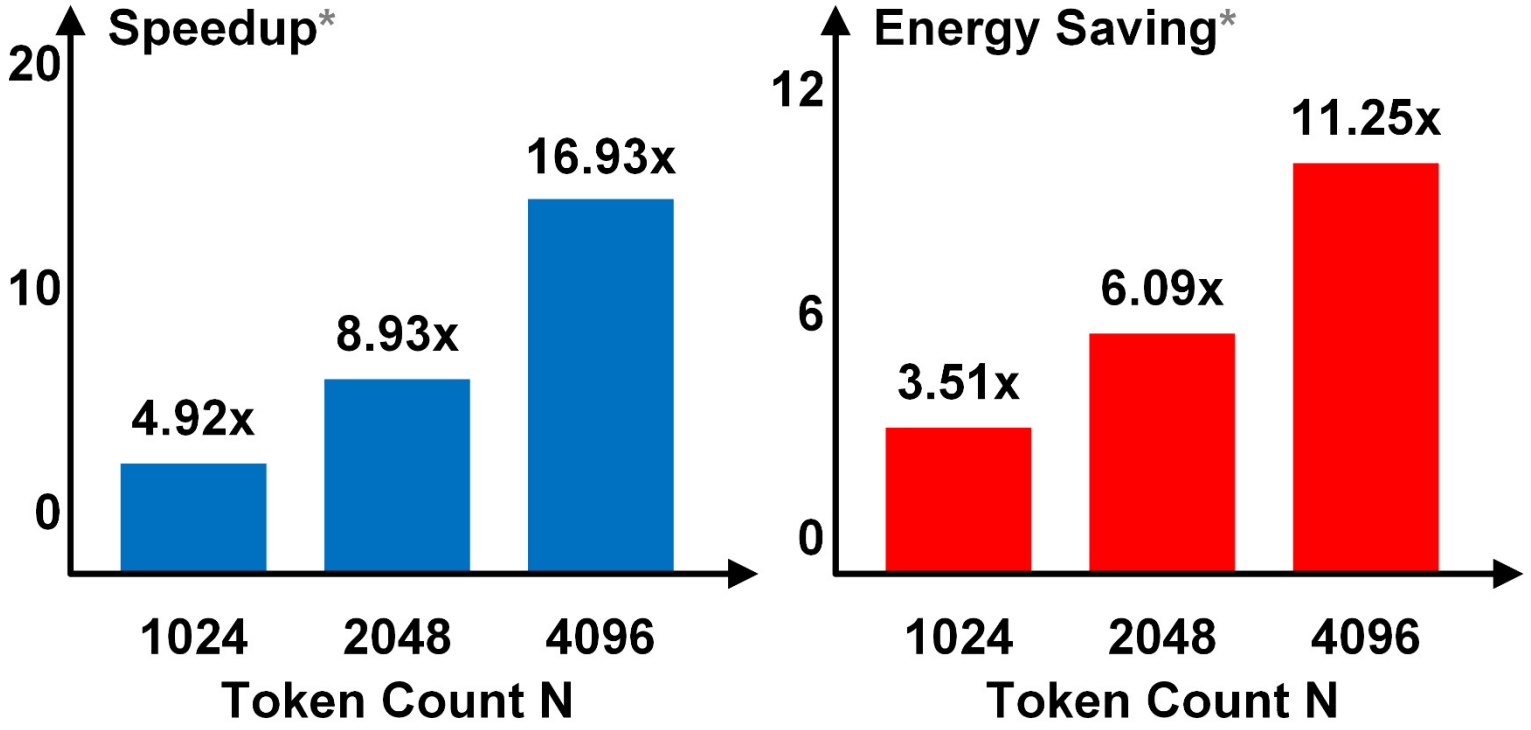
- **Generate configurations for DEngine's CIMs**
  - Control each CIM macro's data source and work state

SAS: Configuration Generator



- **Generate configurations for DEngine's CIMs**
  - Control each CIM macro's data source and work state

Full Attention vs. Sparse Attention (Ours)



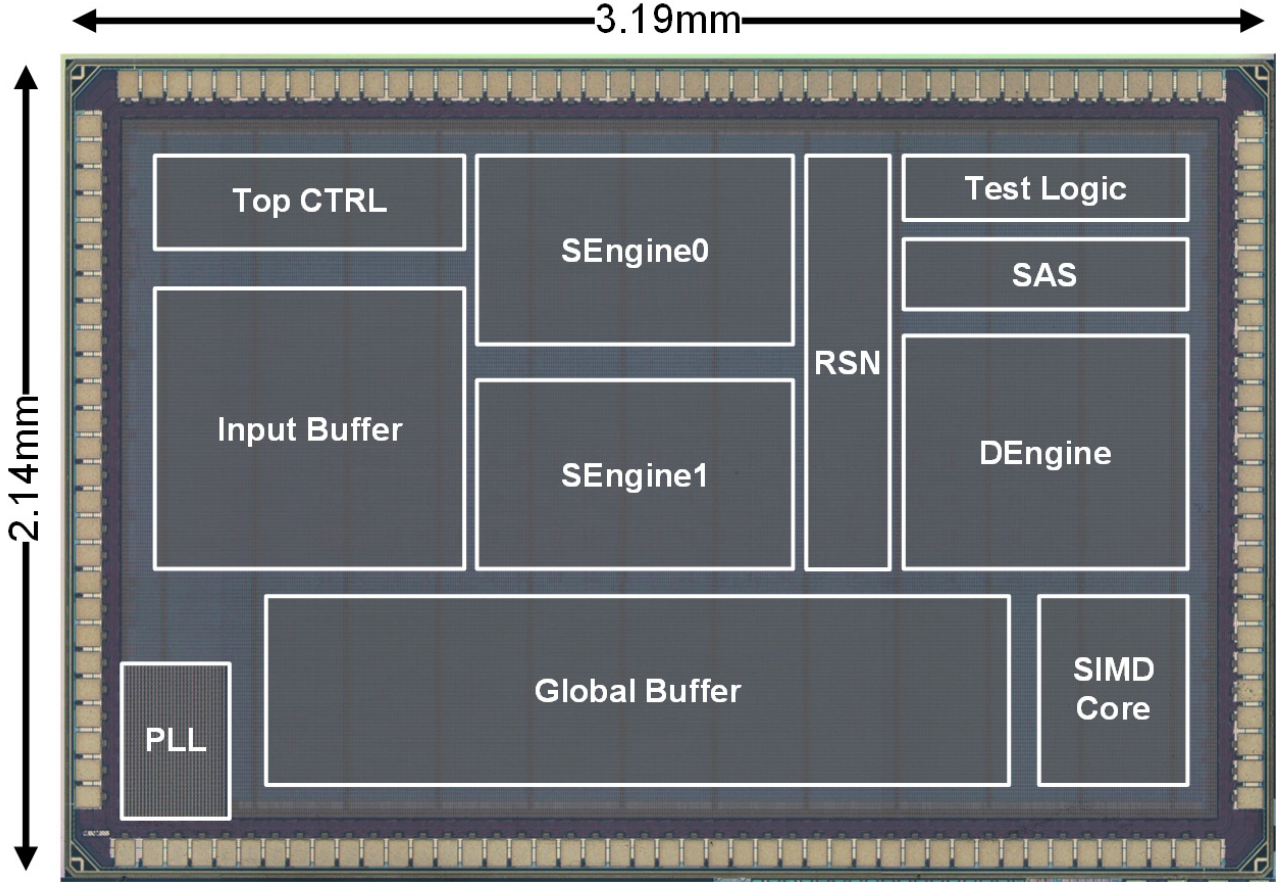
*QK^T-MM Parameters:
INT16, d_{model}=256, d_k=16, b=16.
Token Count N=1024~4096.
Global+local sparse attention.
The baseline is TranCIM working in the pipeline mode with full attention.
Evaluated at 1.0V, 240MHz.

- **SAS assigns only dense and necessary workload to DEngine**
  - 4.92x~16.93x speedup
  - 3.51x~11.25x energy saving

Outline

- Motivation
- TranCIM Accelerator Features
  - Pipeline/Parallel Reconfigurable Modes
  - Bitline-Transpose-CIM (BLT-CIM)
  - Sparse Attention Scheduler (SAS)
- **Measurement Results**
- Conclusion

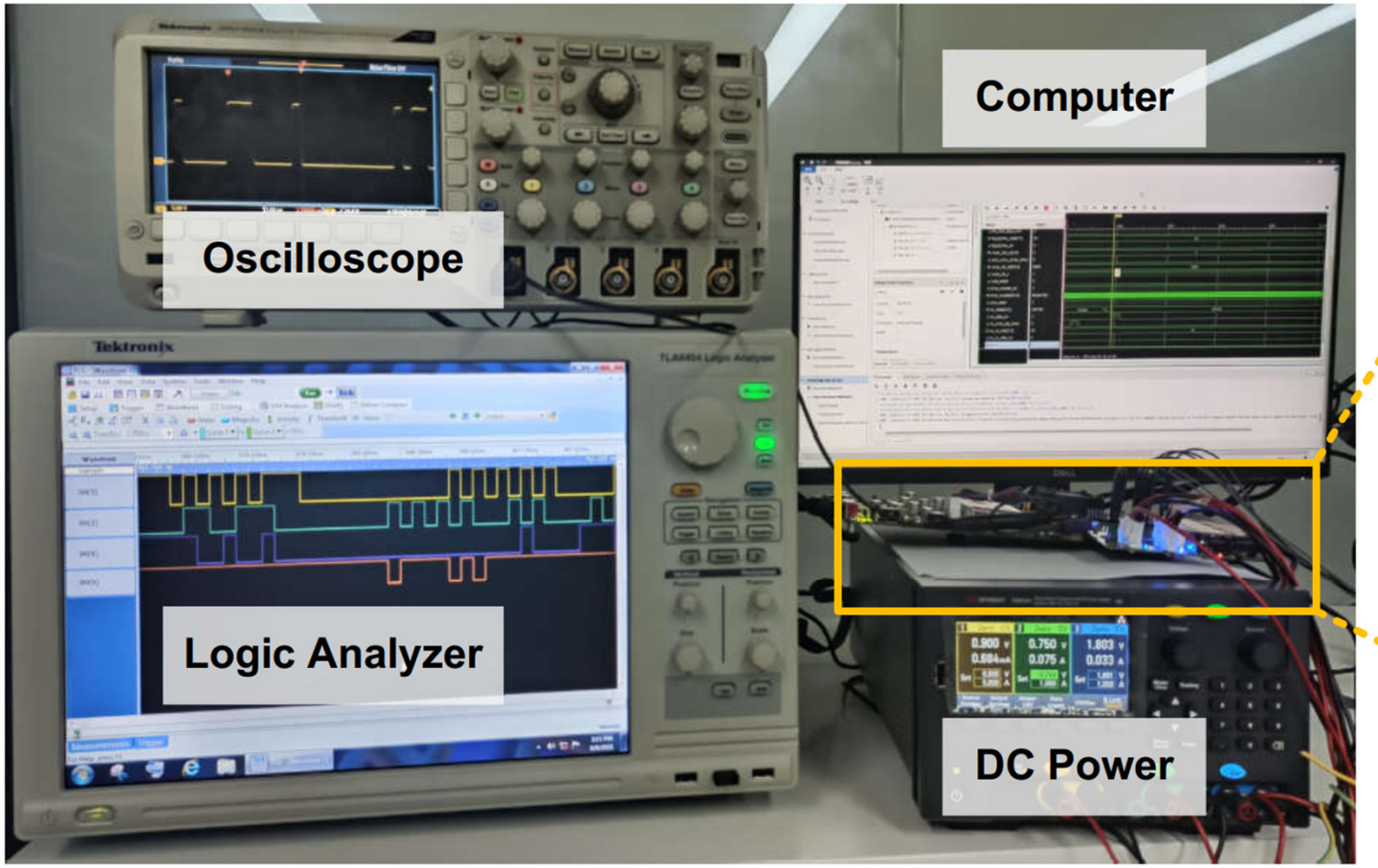
Die Photo and Summary Table

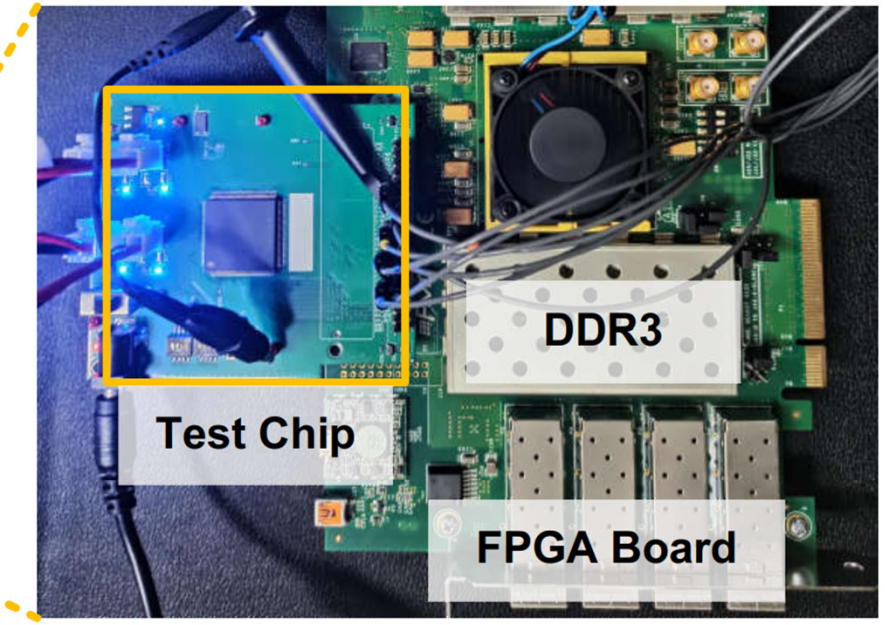


Technology	TSMC 28nm CMOS	
Die Area	2.14mm×3.19mm	
Supply Voltage	0.6–1.0V	
Frequency	80–240MHz	
Buffer Size	192KB	
CIM Size	24KB	
Data Precision	INT8, INT16	
Work Mode	Pipeline/Parallel	
Power	27.04mW @0.6V, 80MHz	
	118.21mW @1.0V, 240MHz	
Peak Performance*1	INT8	1.48TOPS
	INT16	0.37TOPS
Energy Efficiency*1*2	INT8	12.5-20.5TOPS/W
	INT16	3.1-5.1TOPS/W
Energy/Token*2	BERT-base	15.59μJ/Token
	BERT-large	55.10μJ/Token

*1: Highest performance (lowest efficiency) point, 1.0V, 240MHz.
*2: Highest efficiency point, 0.65V, 125MHz. Chip-level energy efficiency includes all on-chip components. Token Count N=4096. Global+local sparse attention.

Test Platform for TranCIM





Evaluation on Different Transformer Models

Model	BERT-base	BERT-large
Dataset	Natural Questions	
Data Precision	INT16 for attention layers, INT8 for FC layers	
Accuracy (LA/SA)	0.736/0.544	0.765/0.570
Accuracy Loss*1	-0.012/-0.008	-0.014/-0.011
Execution Time*2	1684ms (DDR3 included)	6646ms (DDR3 included)
Speedup*2*3	5.22x (attention layers) 2.63x (entire model)	4.27x (attention layers) 2.12x (entire model)
Energy Saving*2*3	7.99x (attention layers) 2.39x (entire model)	6.58x (attention layers) 1.87x (entire model)
Energy/Token*4	15.59μJ/Token	55.10μJ/Token

*1: Compared with accuracy at FP32. Use F1 score as the metric. Token Count N=4096. Global+local sparse attention.
*2: Off-chip DDR3 memory is included into latency and energy. Measured at 1.0V, 240MHz for high-performance evaluation.
*3: The baseline is TranCIM working in the parallel mode with full attention.
*4: Off-chip DDR3 memory is excluded to evaluate chip-level energy consumption. Measured at 0.65V, 125MHz for high-efficiency evaluation.

Comparison with State-of-the-art

		ISSCC'21-15.2 [1]	ISSCC'21-16.3 [2]	ISSCC'21-16.4 [3]	This Work
MAC Implementation		Analog CIM	Analog CIM	Digital CIM	**Digital CIM**
Data Precision		INT2/4/6/8	INT4/8	INT4/8/12/16	**√ INT8/16**
Work Mode		Parallel	Parallel	Parallel	**√ Pipeline/Parallel**
Technology		65nm	28nm	22nm	**28nm**
Supply Voltage (V)		0.62-1.0	0.7-0.9	0.72	**0.6-1.0**
Frequency (MHz)		25-100	138.9-250	500	**80-240**
Die Area (mm²)		12	0.772 (array)	0.202 (macro)	**6.83**
Power (mW)		18.6-84.1	-	-	**27.04-118.21**
Peak Performance*1 (TOPS)	INT8	0.013	0.213	0.917	**1.48***4
	INT16	N/A	N/A	0.243*3	**0.37***4
Energy Efficiency (TOPS/W)	INT8	2.75 (INT8/4)	7.29*2	11.85*2	**20.5***5
	INT16	N/A	N/A	3.14*2*3	**5.1***5
Energy Consumption*1 on BERT-base (µJ/Token)		N/A (w/o INT16 support)	N/A (w/o INT16 support)	188.33 (12.08x)*2*3	**15.59 (1x)***5

*1: Off-chip DDR3 memory is excluded for chip-level evaluation. One operation (OP) represents one multiplication or one addition.
*2: We add 192KB on-chip buffers same as TranCIM for fair comparison. The estimation is also applied to BERT-base evaluation.
*3: [3] can only work in the parallel mode with full attention. The INT16 results are projected from the INT4 results reported in [3].
*4: Measured at the highest performance point, 1.0V, 240MHz.
*5: Measured at the highest efficiency point, 0.65V, 125MHz. TranCIM has pipeline/parallel modes and sparse attention support.

Conclusion

TranCIM Accelerator Features

- **Pipeline/Parallel Reconfigurable Modes**
  - Reduce memory access for intermediate data
- **Bitline-Transpose-CIM (BLT-CIM)**
  - Avoid transpose buffer overhead and support high precision
- **Sparse Attention Scheduler (SAS)**
  - Reduce attention computation

A **Sparse Transformer Accelerator** based on **Full-Digital Bitline-Transpose SRAM-CIM** with **Pipeline/Parallel Reconfigurable Modes**

RecKOn: A 28nm Sub-mm² Task-Agnostic Spiking Recurrent Neural Network Processor Enabling On-Chip Learning over Second-Long Timescales

Charlotte Frenkel and Giacomo Indiveri

Institute of Neuroinformatics, UZH and ETH Zurich, Switzerland

University of Zurich[UZH]

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Self introduction

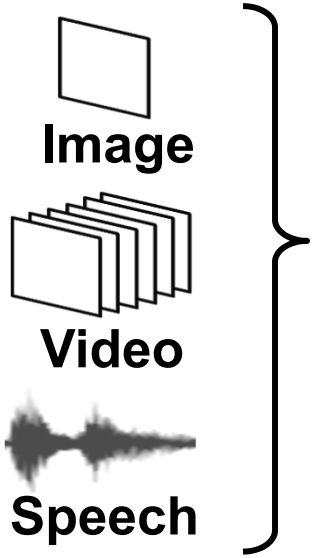
Background

- B.Sc., M.Sc. and Ph.D. from Université catholique de Louvain (UCLouvain), Belgium.
- Since Feb. 2020, postdoctoral researcher at the Institute of Neuroinformatics, UZH/ETH Zurich, Switzerland.
- Incoming assistant professor at TU Delft, Netherlands.

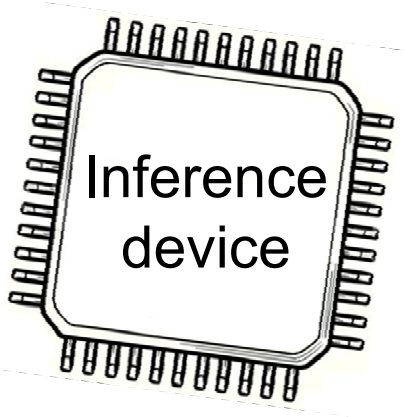
Interests

Neuromorphic engineering, hardware-aware training algorithms, digital design and emerging devices for adaptive edge computing.

On-chip learning – Why?

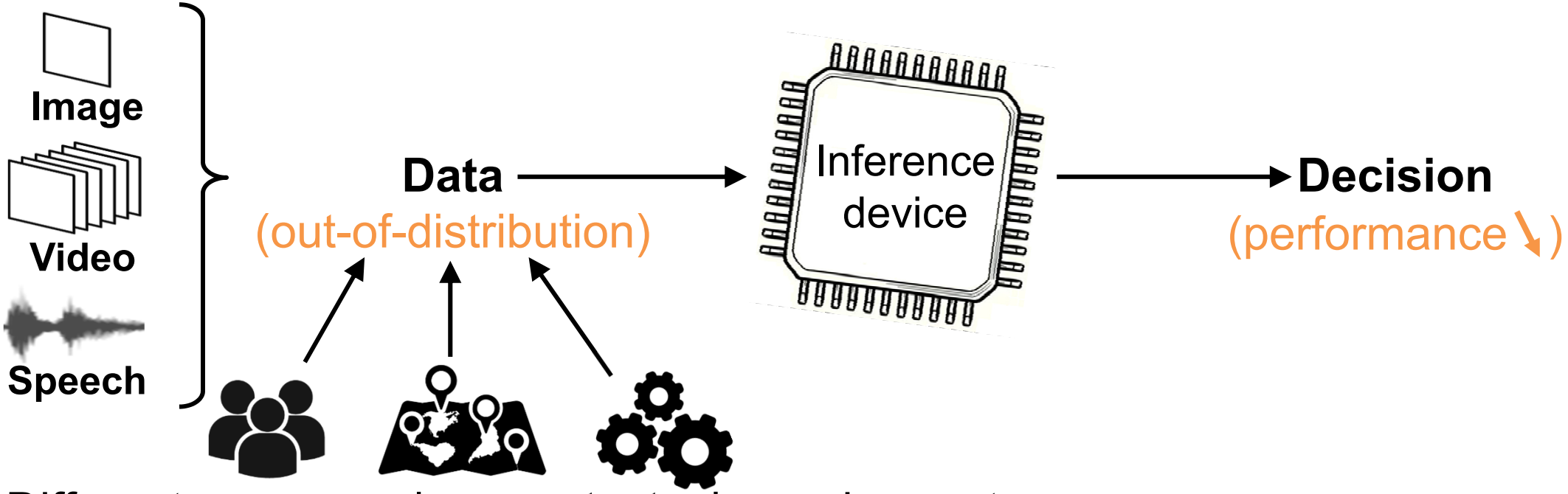


Data
(in-distribution)



Decision
(performance within specs)

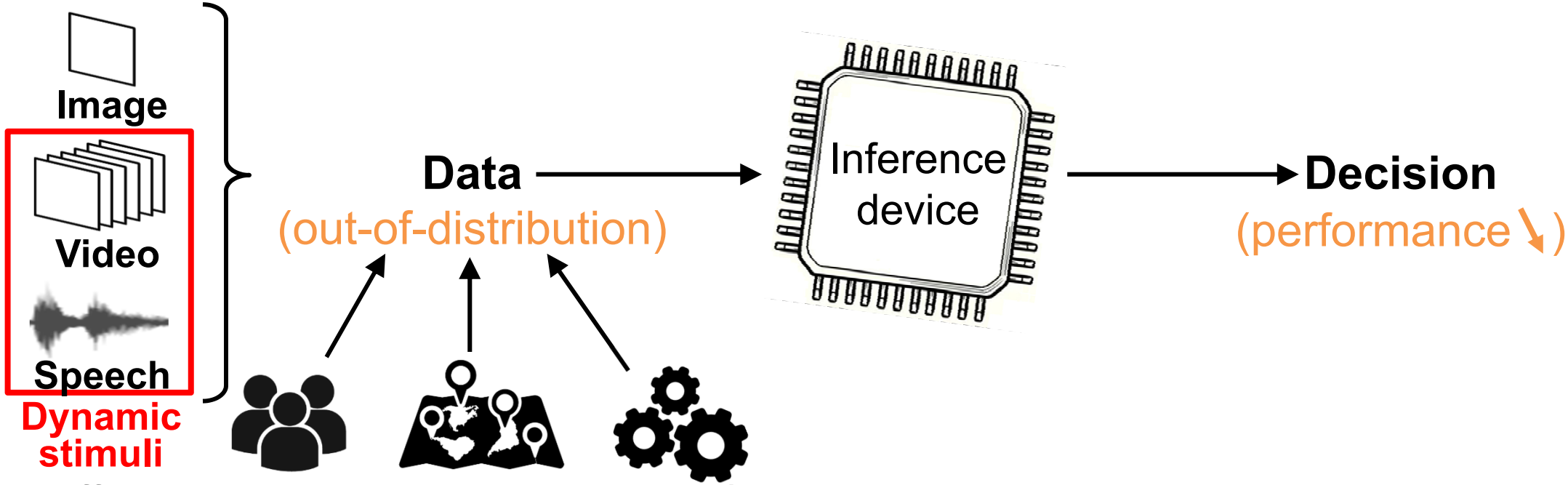
On-chip learning – Why?



Different users, environments, task requirements

- **More training data before deployment?**
  - Issues: cost, robustness, flexibility
- **Data exchange with the cloud?**
  - Issues: power budget, privacy

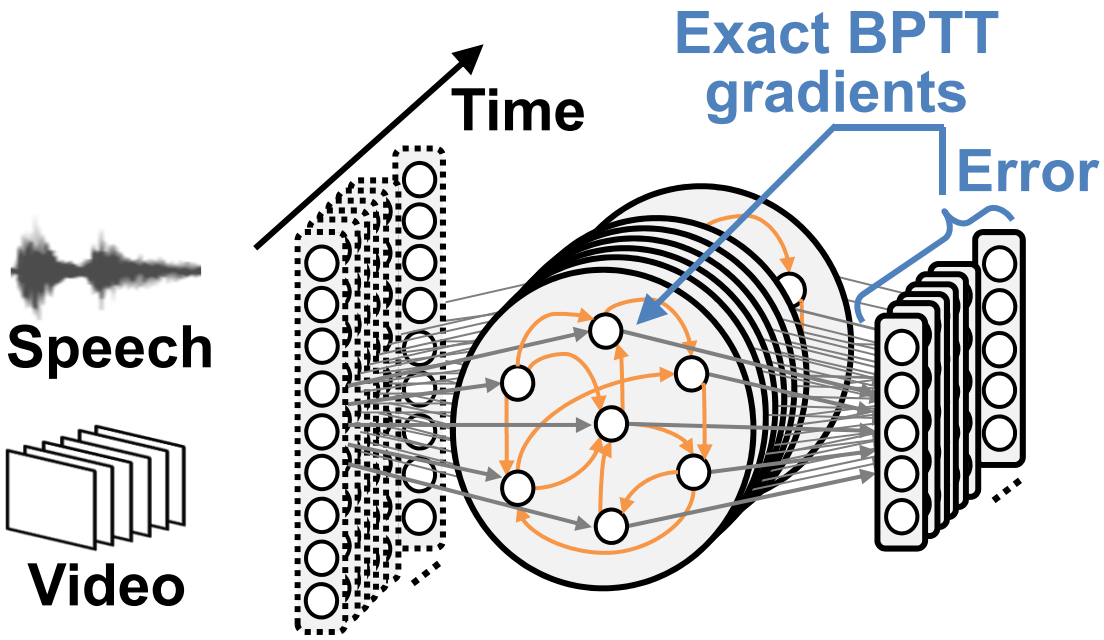
On-chip learning – Why?



- **More training data before deployment?**
  - Issues: cost, robustness, flexibility
- **Data exchange with the cloud?**
  - Issues: power budget, privacy

On-chip training (end-to-end)

On-chip learning with dynamic stimuli – How?



- Unrolling in time: very deep network
  (current learning ICs for static stimuli: ≤3 layers)
- Intractable memory/latency requirements
- No end-to-end on-chip solution to date

Key challenge: On-chip learning over long timescales while keeping a fine-grained temporal resolution

Outline

I. Context and motivation

II. **Training algorithm**
- Bio-inspired solution
- Optimizations for space and time locality

III. **Hardware architecture**
- Spike-based system overview
- Forward state update logic
- Weight update module and skipping mechanism

IV. **Measurement results and comparison**

V. **Summary**

Outline

I. Context and motivation

II. **Training algorithm**

- Bio-inspired solution
- Optimizations for space and time locality

III. Hardware architecture

- Spike-based system overview
- Forward state update logic
- Weight update module and skipping mechanism

IV. Measurement results and comparison

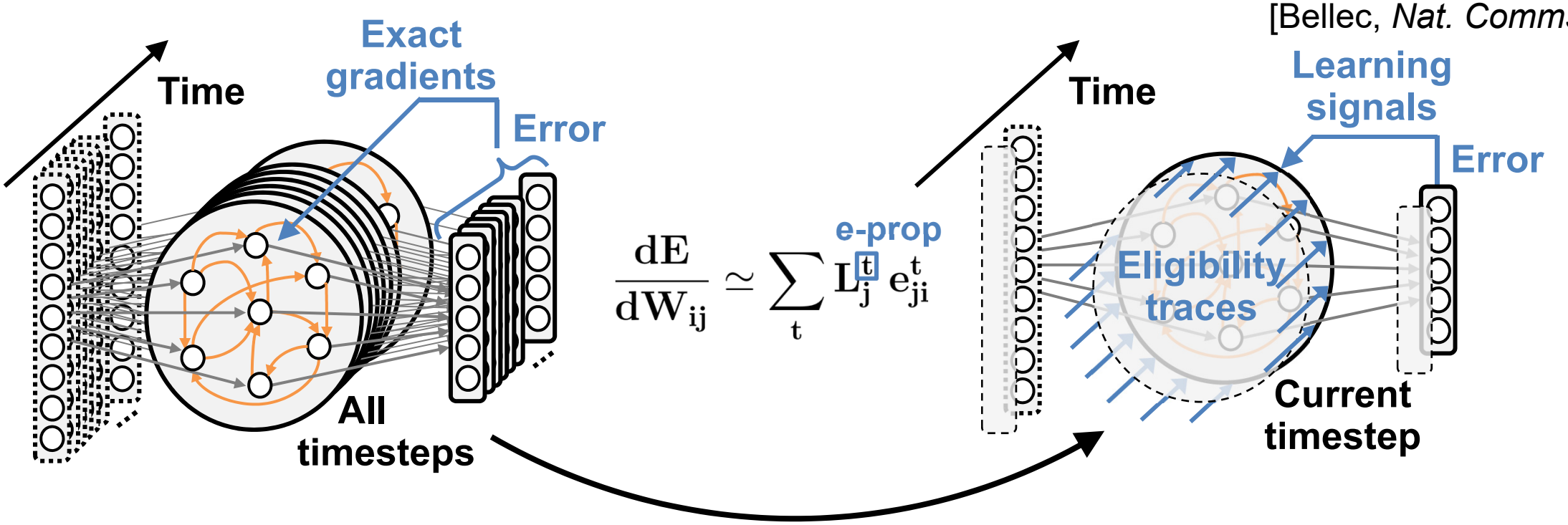
V. Summary

Training – Backward and forward modes

Backprop through time (BPTT, backward)

Eligibility propagation (e-prop, forward)

[Bellec, *Nat. Comms*'20]



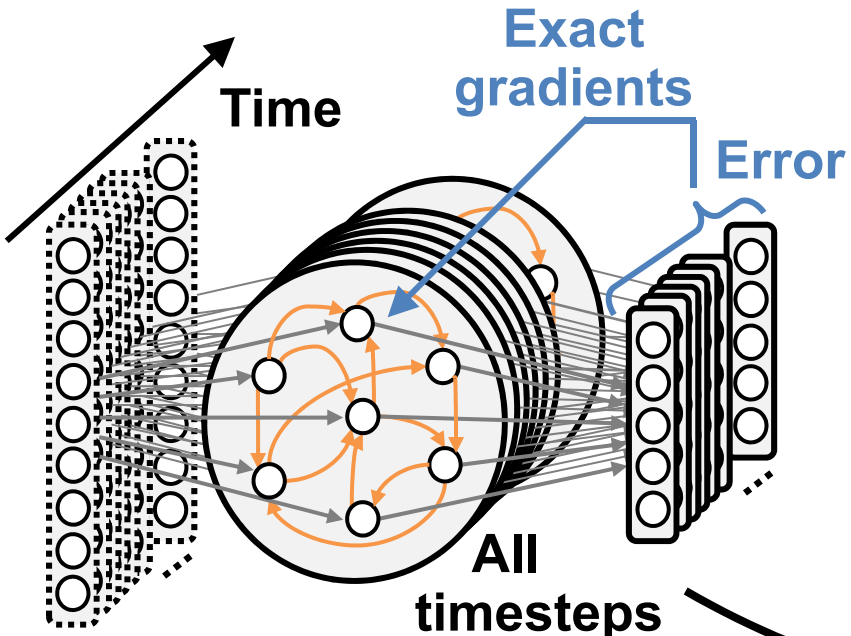
Biological plausibility ↗

Space and time locality ↗

On-chip memory requirements ↘

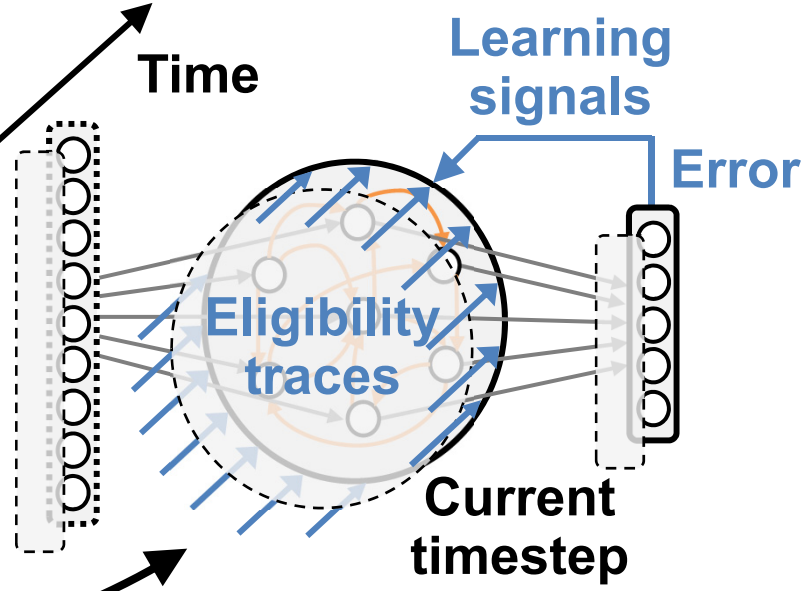
Training – Backward and forward modes

Backprop through time (BPTT, backward)



Eligibility propagation (e-prop, forward)

[Bellec, *Nat. Comms*'20]



$$\frac{dE}{dW_{ij}} \simeq \sum_t L_j^t \, e_{ji}^t$$

e-prop

Remaining challenges solved in 3 steps

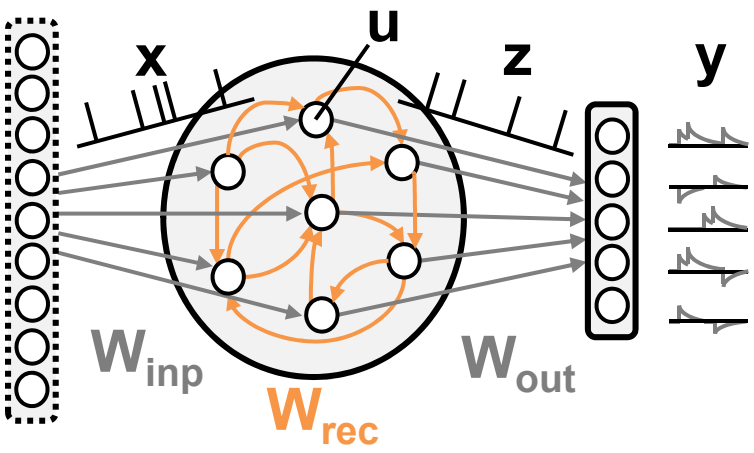
Biological plausibility ↗

Space and time locality ↗

On-chip memory requirements ↘

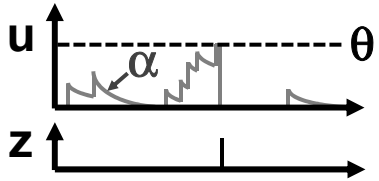
Definitions – Network model, evaluation task

Network model

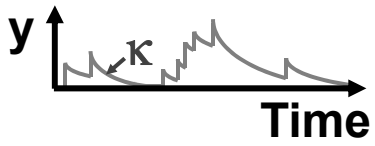


Sample input

Leaky integrate-and-fire (LIF)

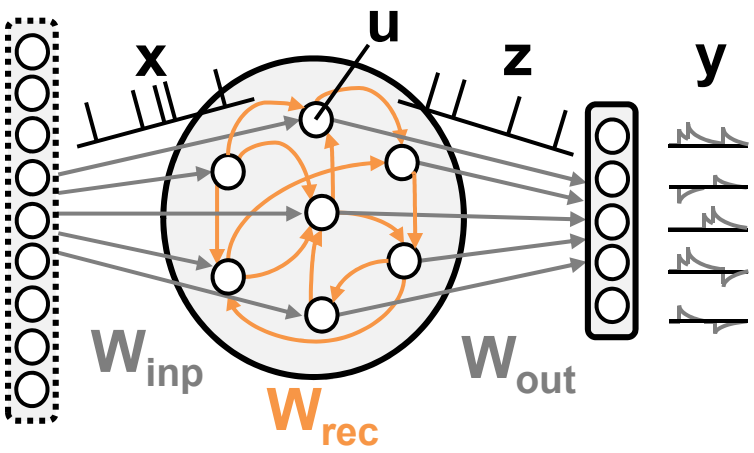


Leaky integrator (LI)



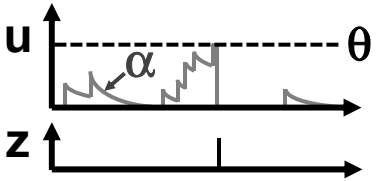
Definitions – Network model, evaluation task

Network model

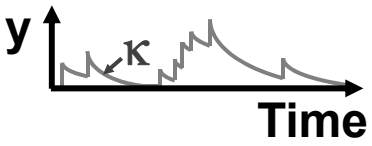


Sample input

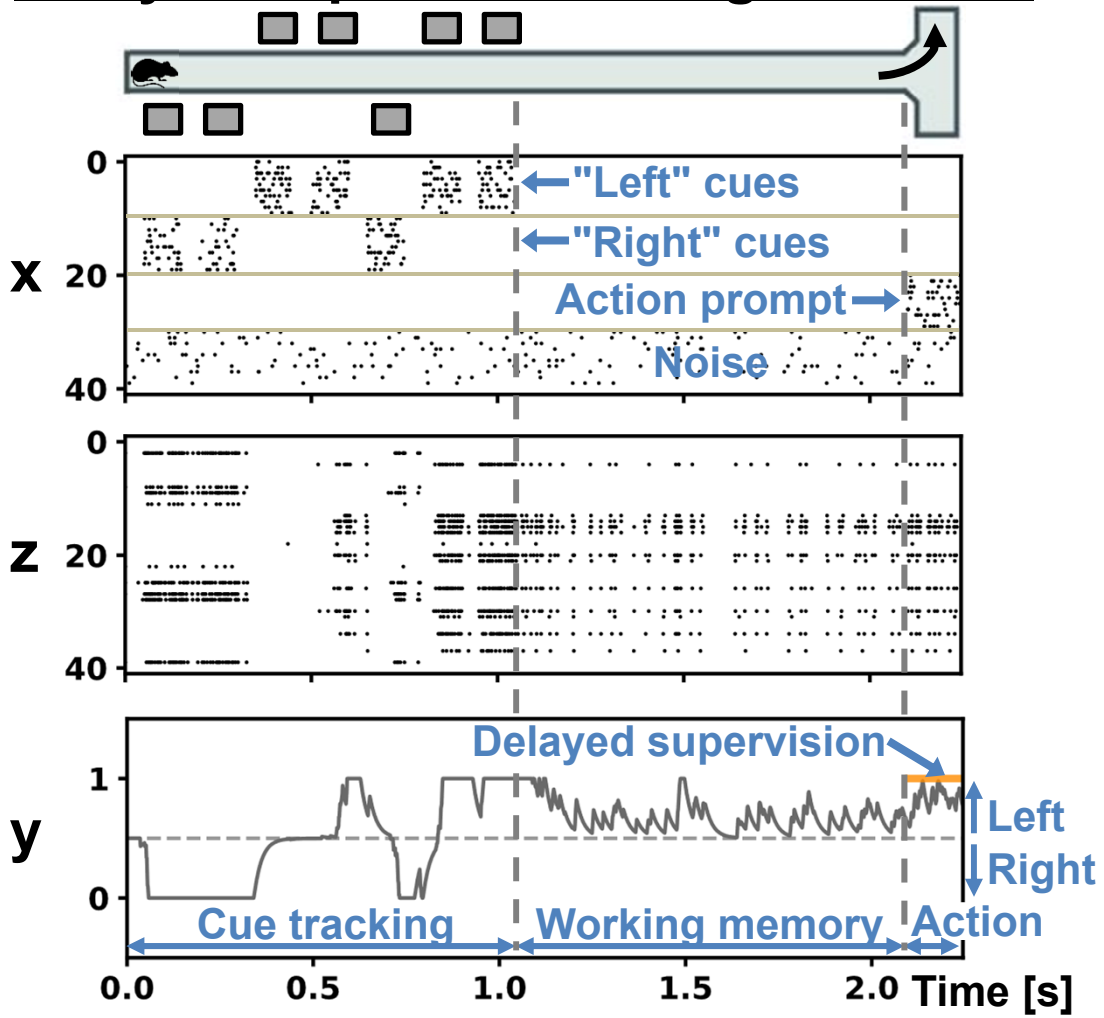
Leaky integrate-and-fire (LIF)



Leaky integrator (LI)

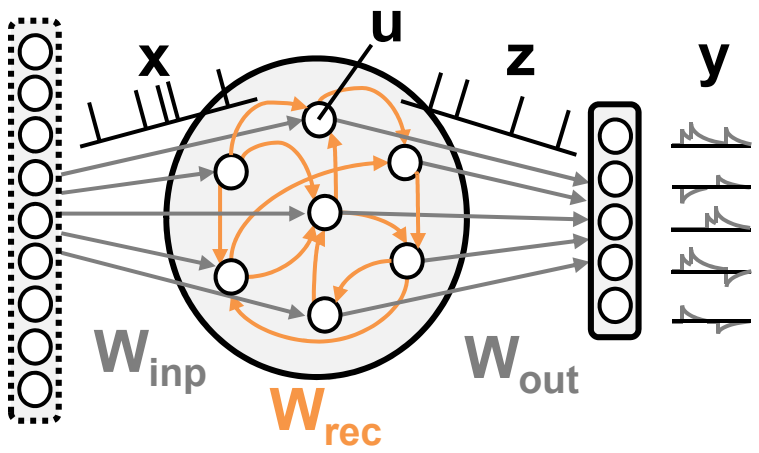


Delayed-supervision navigation task



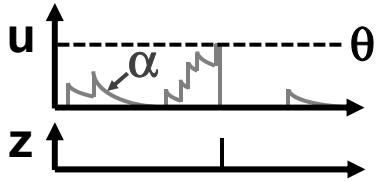
Step 1 – Neuron model selection

Network model

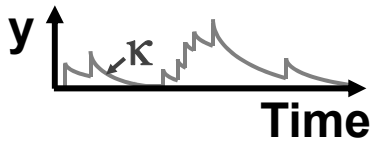


Sample input

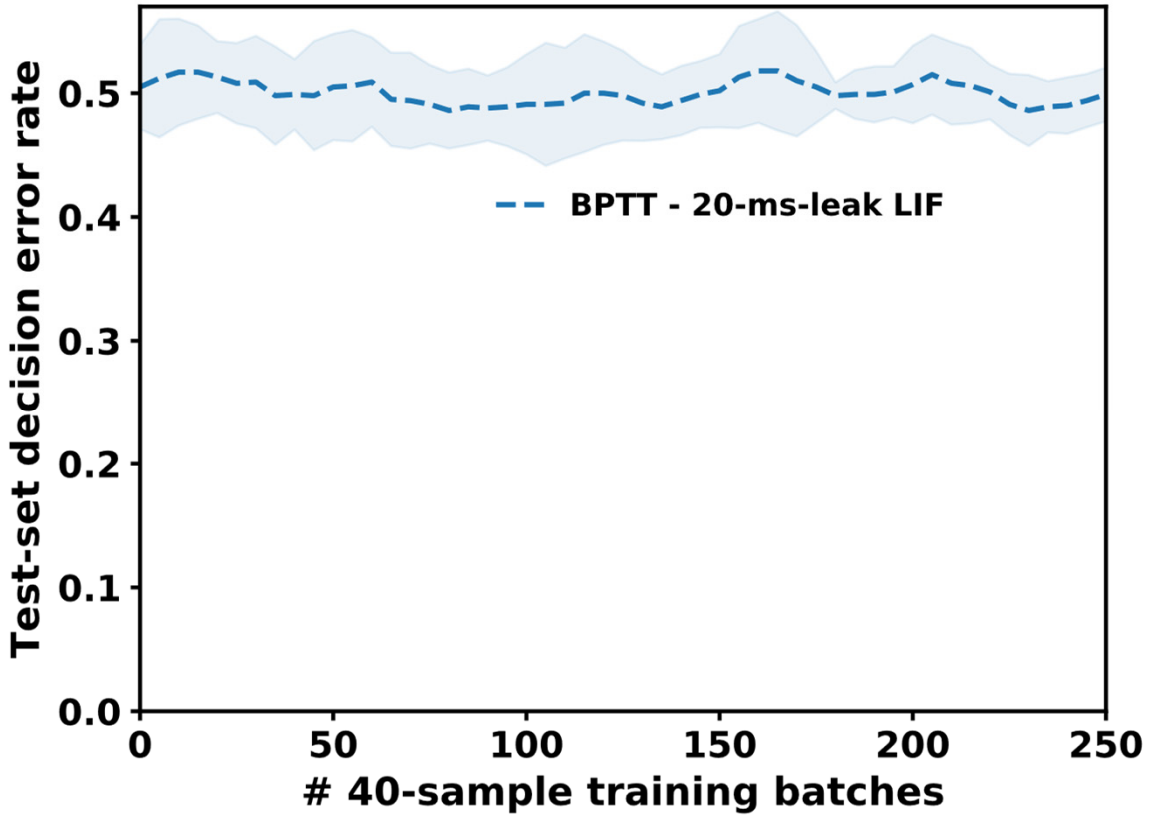
Leaky integrate-and-fire (LIF)



Leaky integrator (LI)

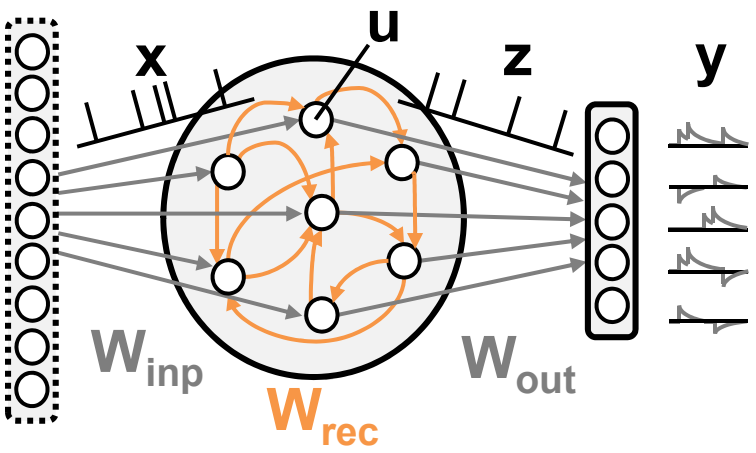


Task performance



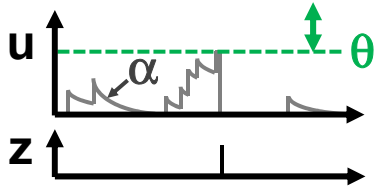
Step 1 – Neuron model selection

Network model

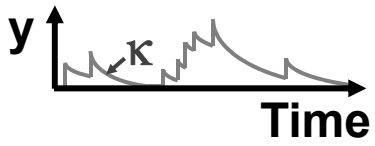


Sample input

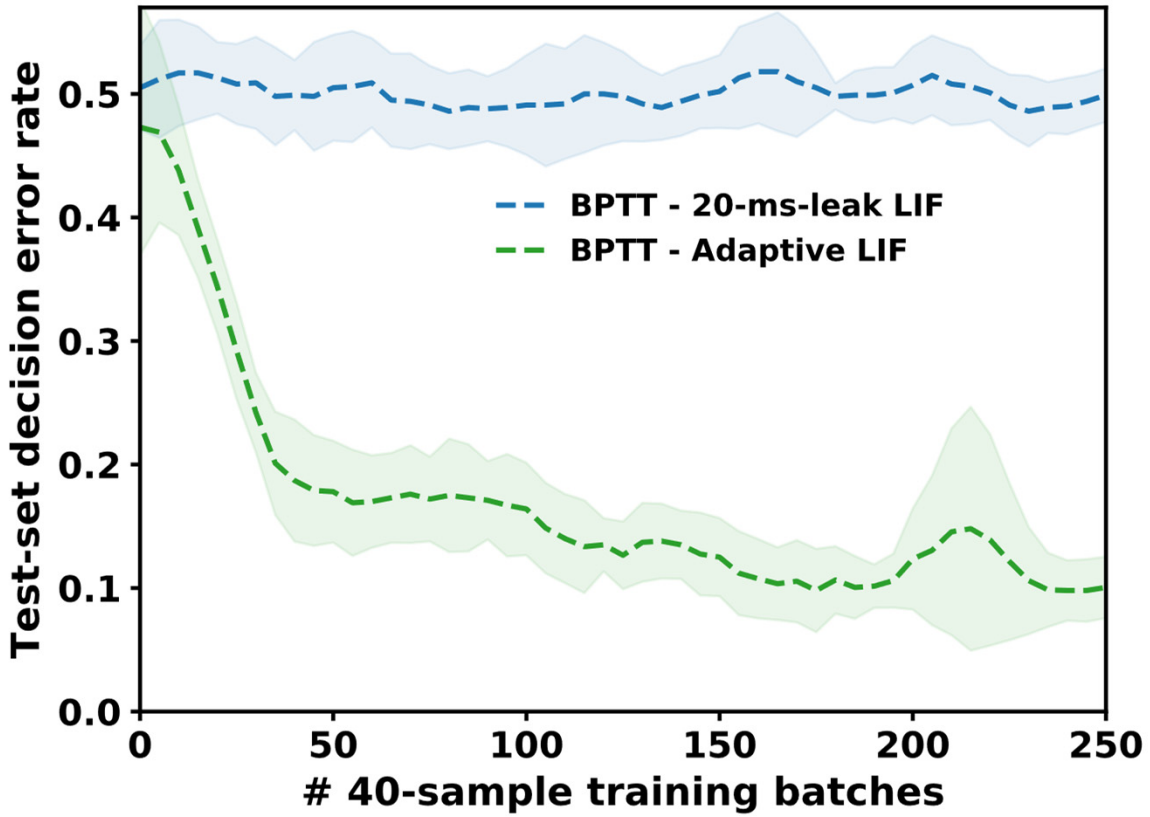
Leaky integrate-and-fire (LIF)
or adaptive LIF (ALIF)



Leaky integrator (LI)

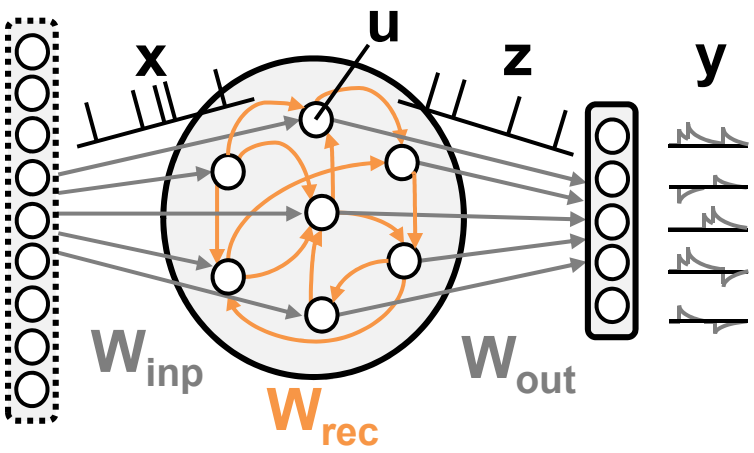


Task performance

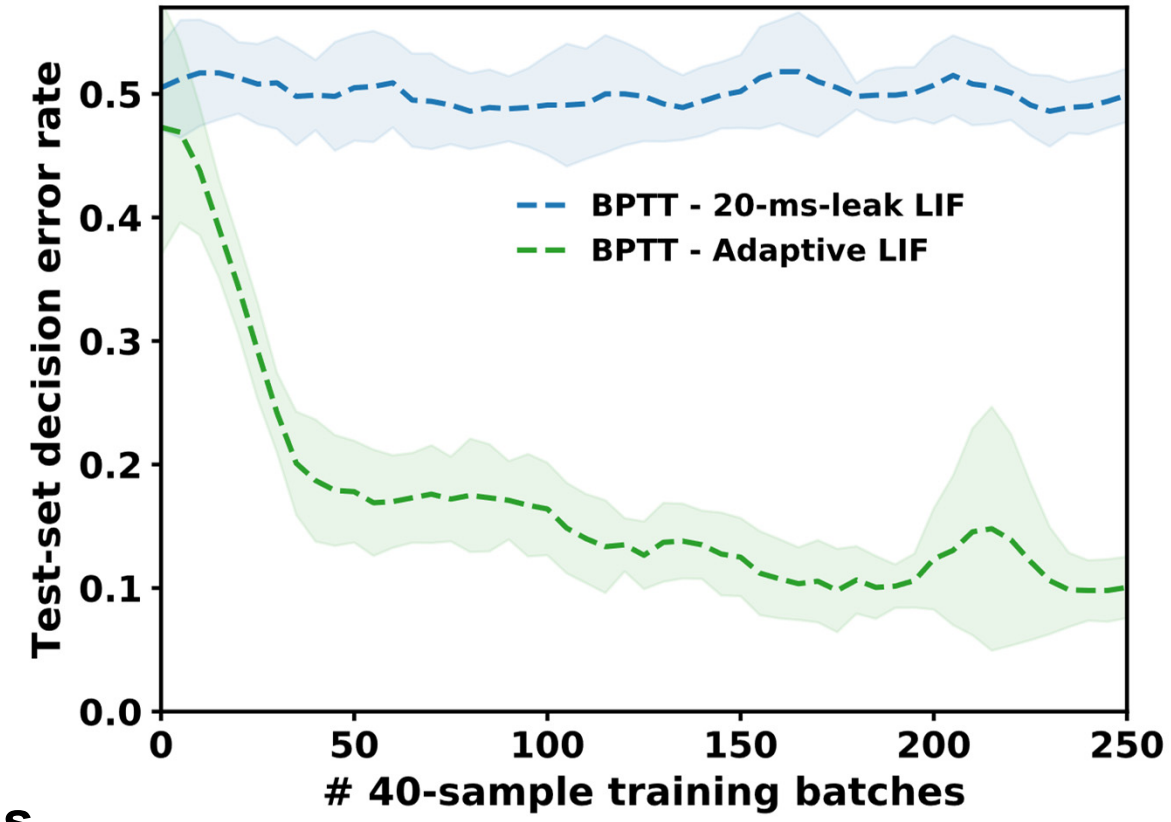


Step 1 – Neuron model selection

Network model



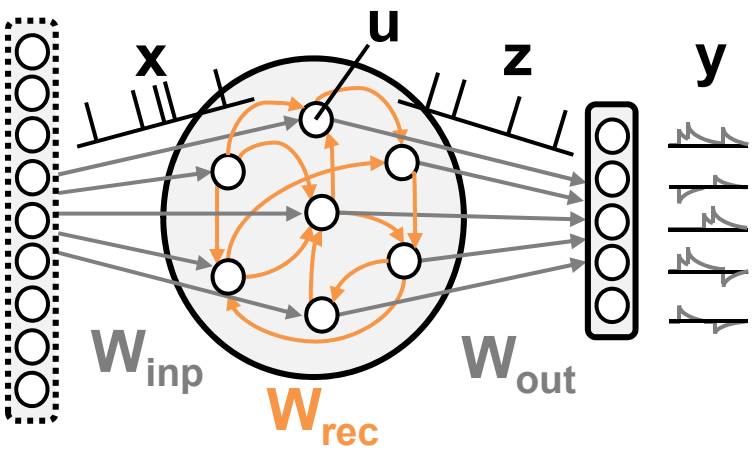
Task performance



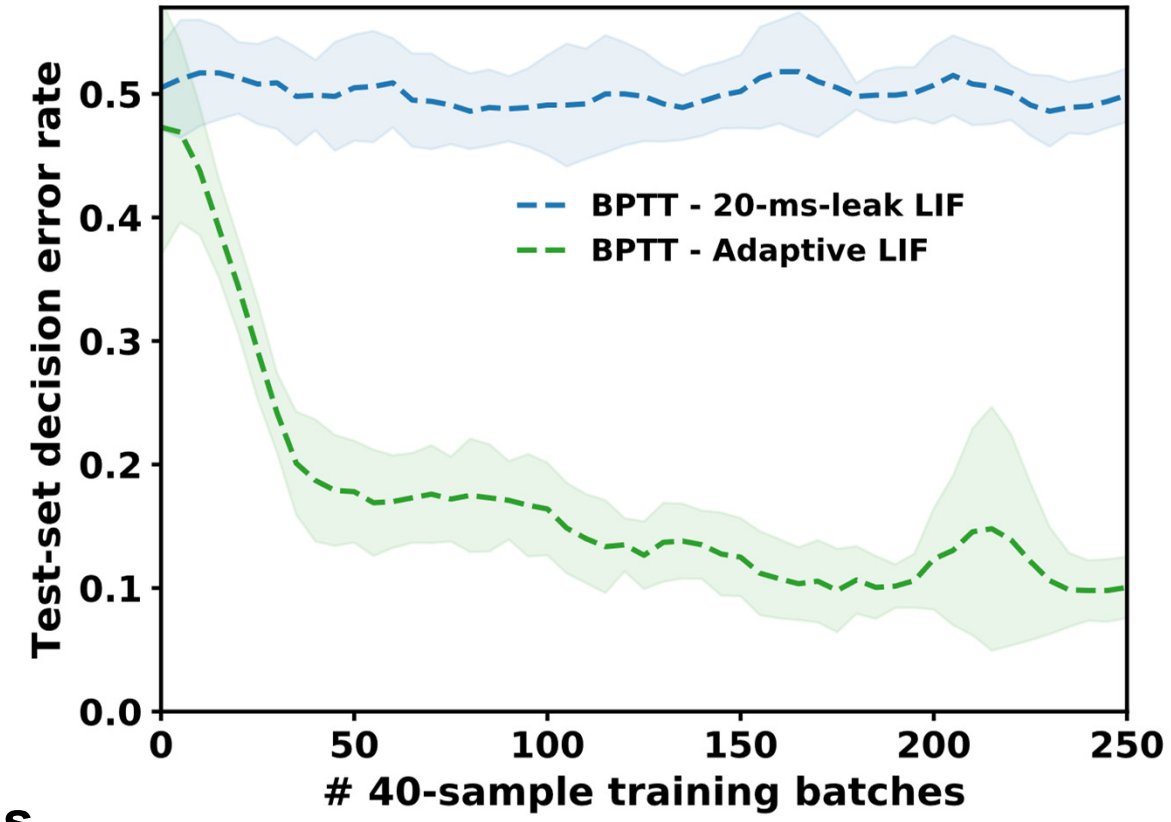
- **Leaky integrate-and-fire (LIF):**
  - ✗ Only a short time constant (**~20-ms leak**)
- **Adaptive LIF (ALIF):**
  - ✓ Embeds threshold adaptation over **100s of ms**

Step 1 – Neuron model selection

Network model



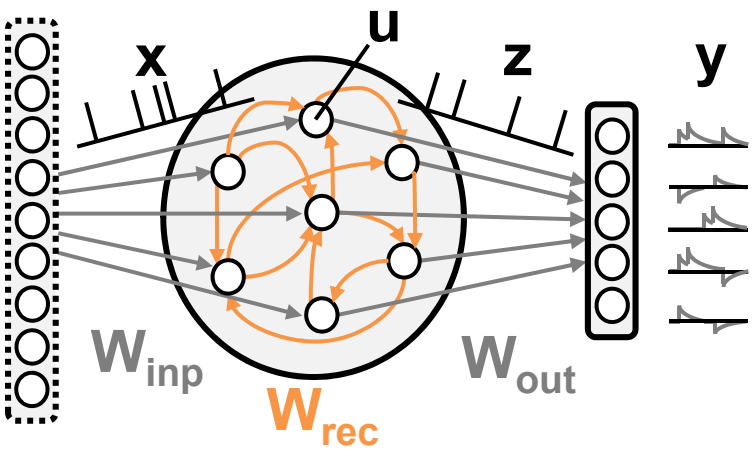
Task performance



- **Leaky integrate-and-fire (LIF):**
  - ✗ Only a short time constant (~**20-ms leak**)
  - ✓ Eligibility traces: simple activity LPF
- **Adaptive LIF (ALIF):**
  - ✓ Embeds threshold adaptation over **100s of ms**
  - ✗ Eligibility traces: Complex per-synapse multi-scale filtering

Step 1 – Neuron model selection

Network model



- **Leaky integrate-and-fire (LIF):**
  - ✗ Only a short time constant (~**20-ms leak**)
  - ✓ Eligibility traces: simple activity LPF
- **Adaptive LIF (ALIF):**
  - ✓ Embeds threshold adaptation over **100s of ms**
  - ✗ ET: Complex per-synapse multi-scale filtering

- **LIF with configurable leak:**
  - ✓ Flexible time constant (**ms to sec**)
  - ✓ Eligibility traces: simple activity LPF
  - ≈ Less biologically plausible

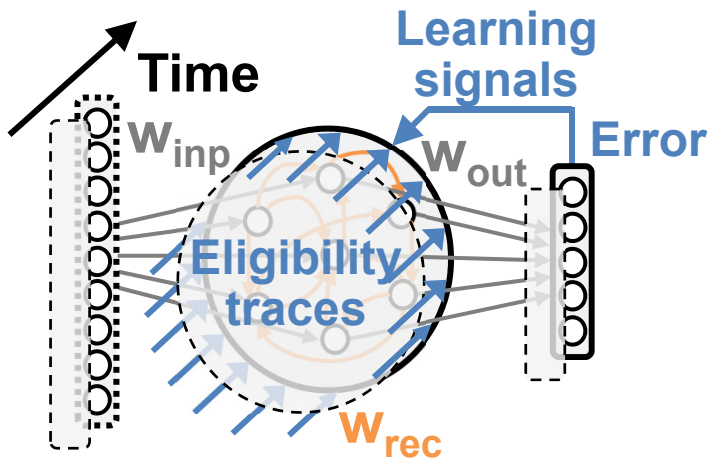
Step 2 – From LIF e-prop to space/time locality

Learning signals (LS) | Eligibility traces (ET)

$$\Delta w_{out,kj} \propto \sum_t \left(y_k^{*,t} - y_k^t\right) \sum_{t' \le t} \left(\kappa^{t-t'} z_j^{t'}\right)$$

$$\Delta w_{rec,ji} \propto \sum_t \left(\sum_k w_{out,kj}\left(y_k^{*,t} - y_k^t\right)\right) \sum_{t' \le t} \left(\kappa^{t-t'} h_j^{t'} \sum_{t'' \le t'} \left(\alpha^{t'-t''} z_i^{t''}\right)\right)$$

$$\Delta w_{inp,ji} \propto \sum_t \left(\sum_k w_{out,kj}\left(y_k^{*,t} - y_k^t\right)\right) \sum_{t' \le t} \left(\kappa^{t-t'} h_j^{t'} \sum_{t'' \le t'} \left(\alpha^{t'-t''} x_i^{t''}\right)\right)$$



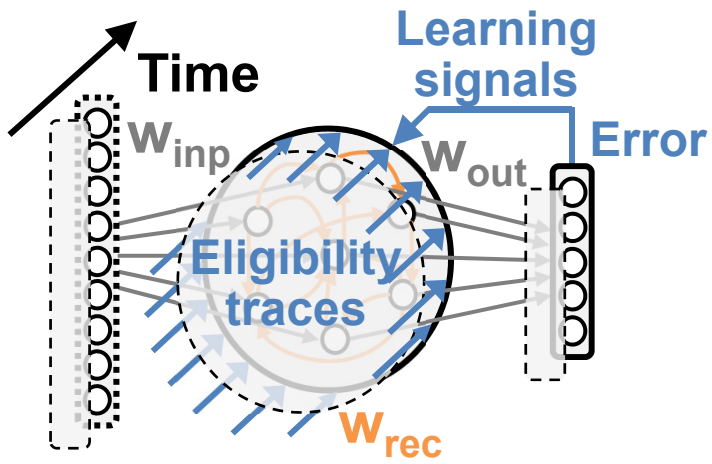
Step 2 – From LIF e-prop to space/time locality

Learning signals (LS) | Eligibility traces (ET)

$$\Delta w_{out,kj} \propto \textcircled{1} \sum_t \left(y_k^{*,t} - y_k^t\right) \sum_{t' \le t} \left(\kappa^{t-t'} z_j^{t'}\right)$$

$$\Delta w_{rec,ji} \propto \textcircled{1} \sum_t \left(\sum_k w_{out,kj} \left(y_k^{*,t} - y_k^t\right)\right) \sum_{t' \le t} \left(\kappa^{t-t'} h_j^{t'} \sum_{t'' \le t'} \left(\alpha^{t'-t''} z_i^{t''}\right)\right)$$

$$\Delta w_{inp,ji} \propto \textcircled{1} \sum_t \left(\sum_k w_{out,kj} \left(y_k^{*,t} - y_k^t\right)\right) \sum_{t' \le t} \left(\kappa^{t-t'} h_j^{t'} \sum_{t'' \le t'} \left(\alpha^{t'-t''} x_i^{t''}\right)\right)$$



① Requires a dedicated gradient memory ⟶ Per-timestep updates

Step 2 – From LIF e-prop to space/time locality

Learning signals (LS) | Eligibility traces (ET)

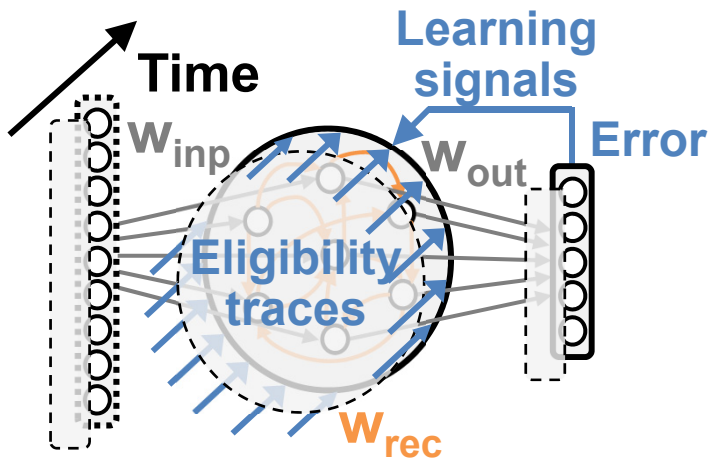
$$\Delta w^{t}_{out,kj} \propto \left(y^{*,t}_{k} - y^{t}_{k}\right) \sum_{t' \le t} \left(\kappa^{t-t'} z^{t'}_{j}\right)$$

$$\Delta w^{t}_{rec,ji} \propto \left(\sum_{k} w^{t}_{out,kj}\left(y^{*,t}_{k} - y^{t}_{k}\right)\right) \sum_{t' \le t} \left(\kappa^{t-t'} h^{t'}_{j}\right) \sum_{t'' \le t'} \left(\alpha^{t'-t''} z^{t''}_{i}\right)$$

$$\Delta w^{t}_{inp,ji} \propto \left(\sum_{k} w^{t}_{out,kj}\left(y^{*,t}_{k} - y^{t}_{k}\right)\right) \sum_{t' \le t} \left(\kappa^{t-t'} h^{t'}_{j}\right) \sum_{t'' \le t'} \left(\alpha^{t'-t''} x^{t''}_{i}\right)$$

② Post-synaptic straight-through estimator (STE)

Pre-synaptic activity LPF



② Temporal coupling of pre- and post-synaptic terms ⟶ Can be neglected

Step 2 – From LIF e-prop to space/time locality

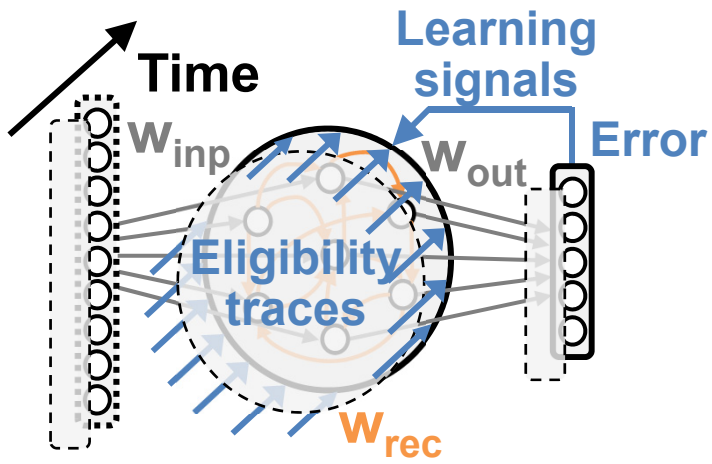
Post-synaptic | Pre-synaptic

$$\Delta w^t_{out,kj} \propto Err^t_k \times ET^t_{out,j}$$

$$\Delta w^t_{rec,ji} \propto LS^t_j \times STE^t_j \times ET^t_{rec,i}$$

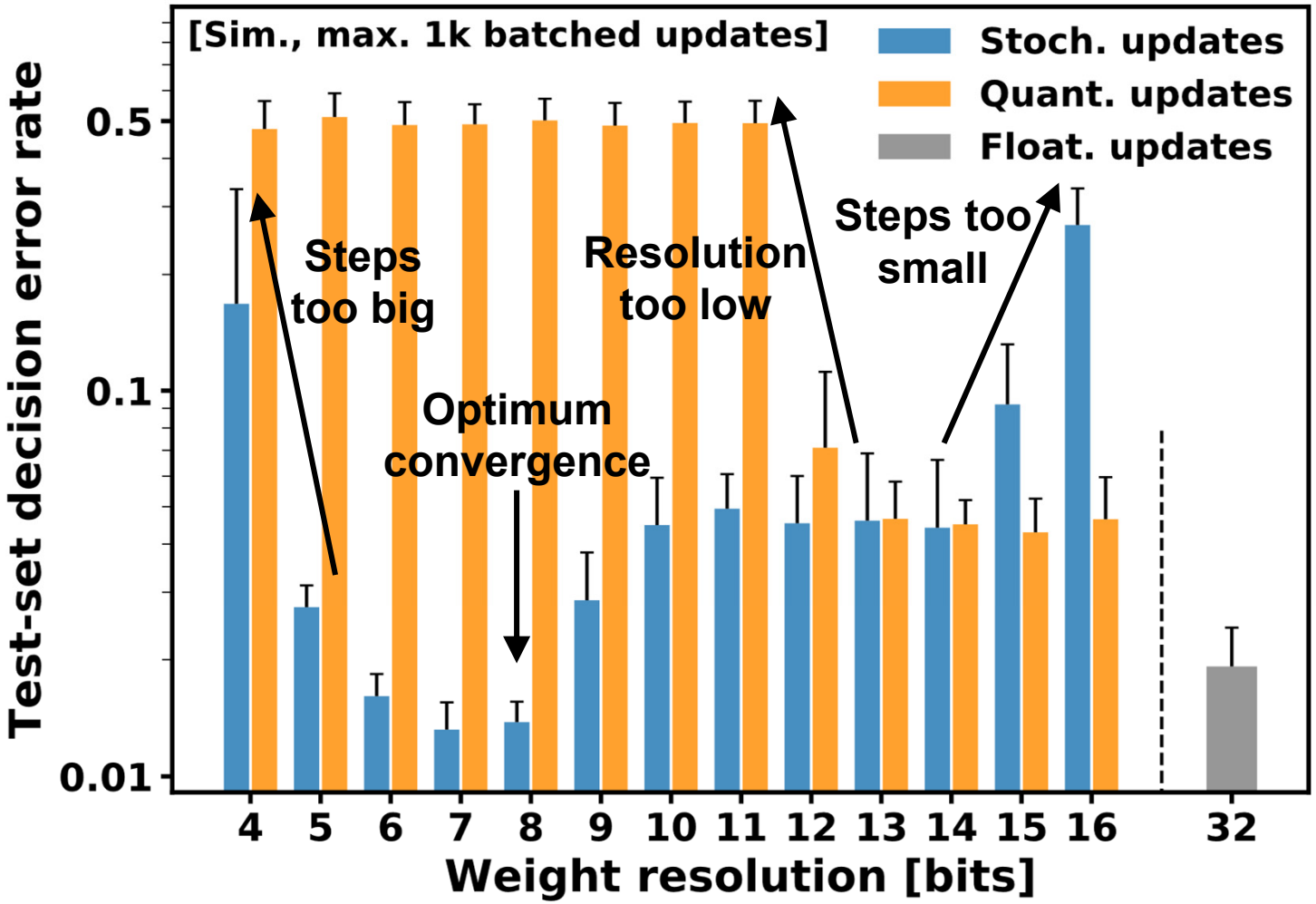
$$\Delta w^t_{inp,ji} \propto LS^t_j \times STE^t_j \times ET^t_{inp,i}$$

Learning signals (LS) | Straight-through estimator (STE) | Eligibility traces (ET)



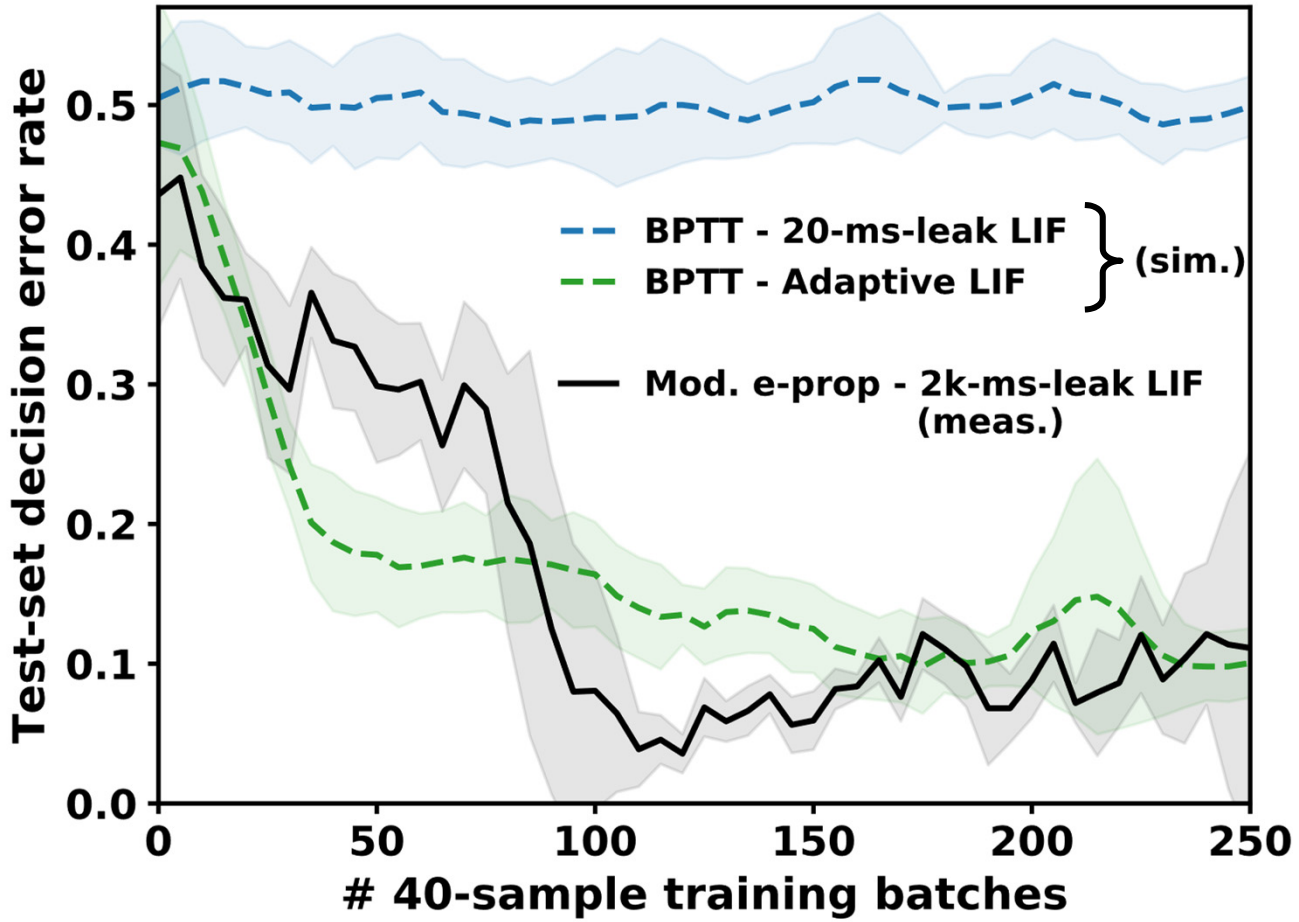
- Full space/time locality
- Pre/post decoupling
- **Per-neuron** ET storage (only memory overhead)

Step 3 – Weight and update quantization



- Quantized updates do not offer sufficient granularity
- Stochasticity helps, updates converge optimally around 8b
- 8b weights are used during both forward & update phases

Modified stochastic e-prop – Final performance



- Competitive with BPTT-trained network of ALIF neurons
- Memory overhead reduced to 0.8% of the inference-only network

Outline

I. Context and motivation

II. Training algorithm

- Bio-inspired solution
- Optimizations for space and time locality

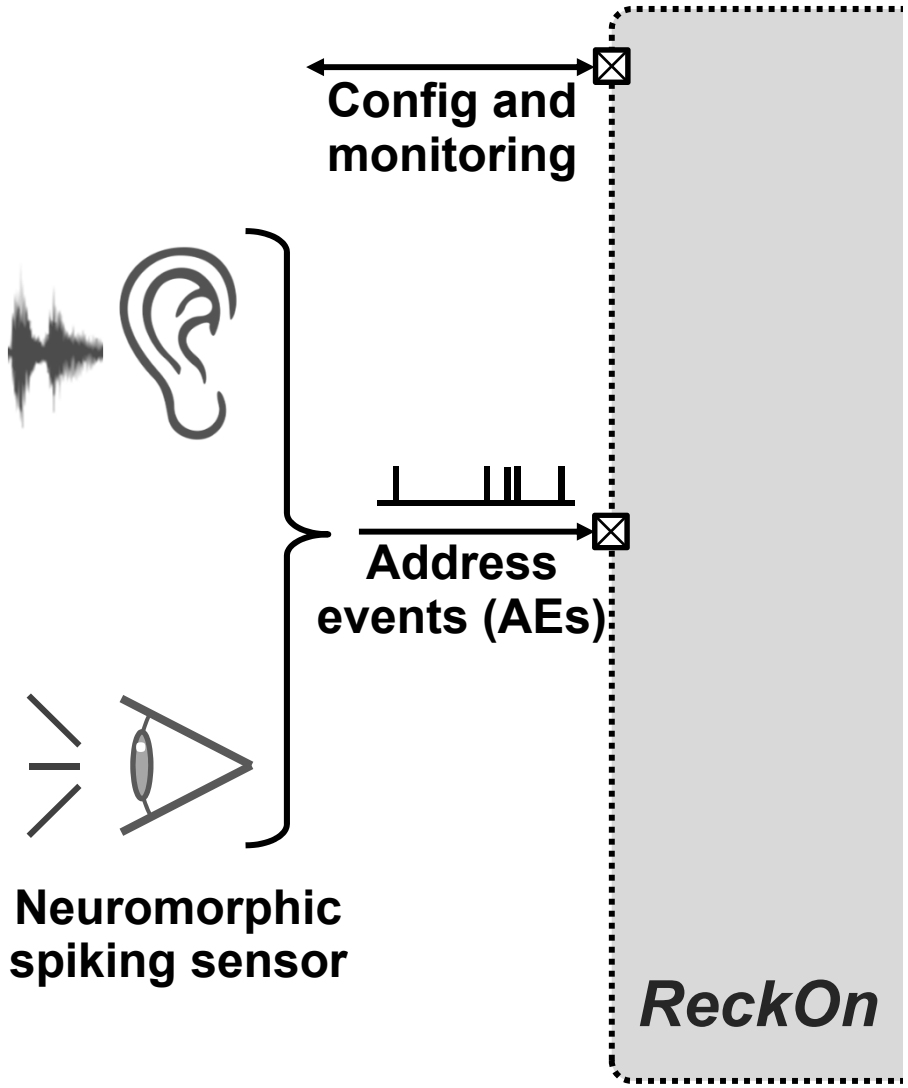
III. Hardware architecture

- Spike-based system overview
- Forward state update logic
- Weight update module and skipping mechanism

IV. Measurement results and comparison

V. Summary

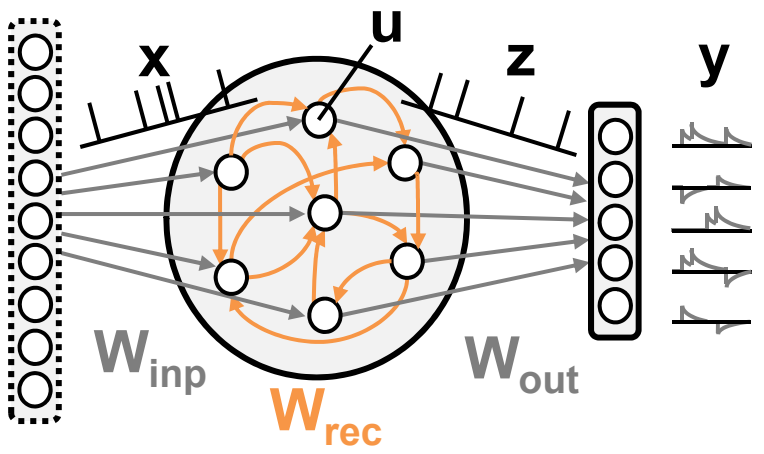
ReckOn – System diagram



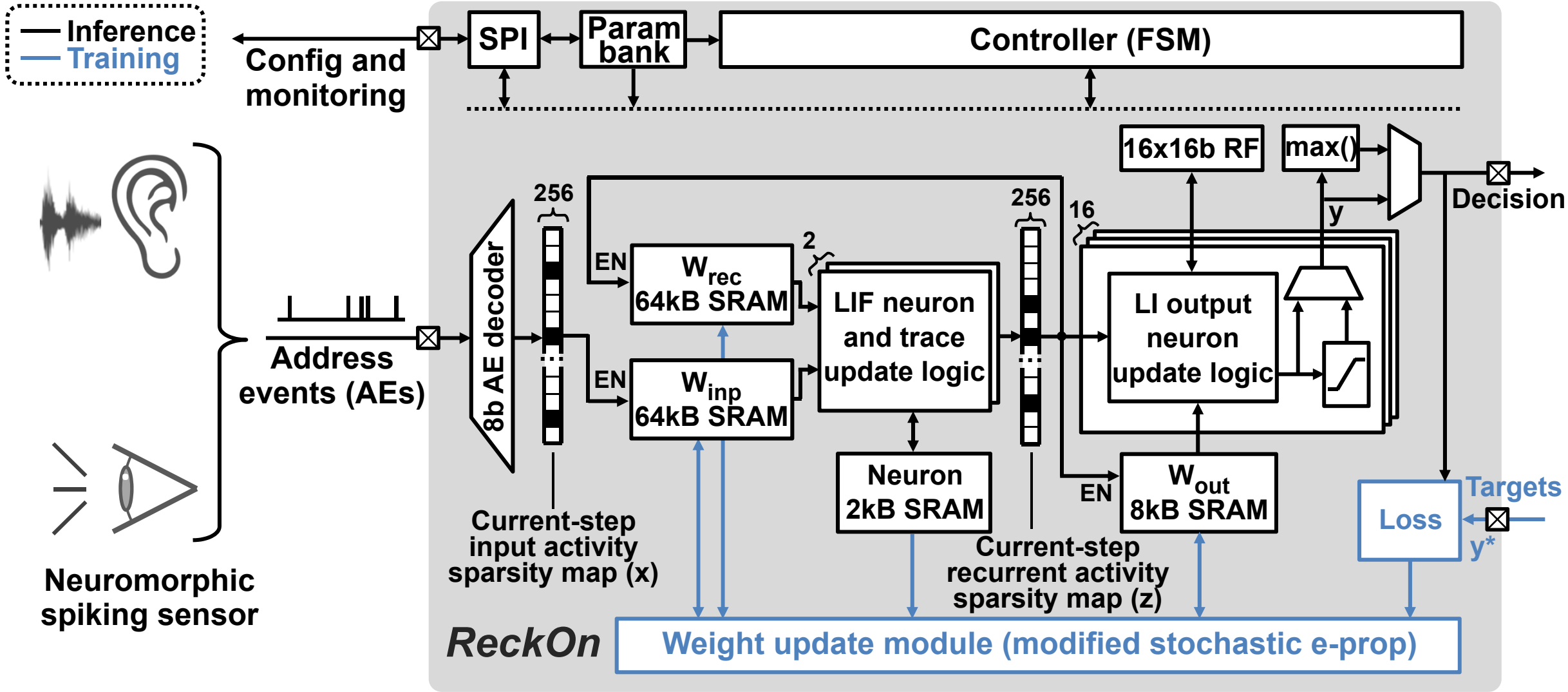
Bio-plausibility is not the only reason for spikes:

- event-driven / sparsity-aware computation
- sensor-agnostic raw-data processing
- task-agnostic processing and learning

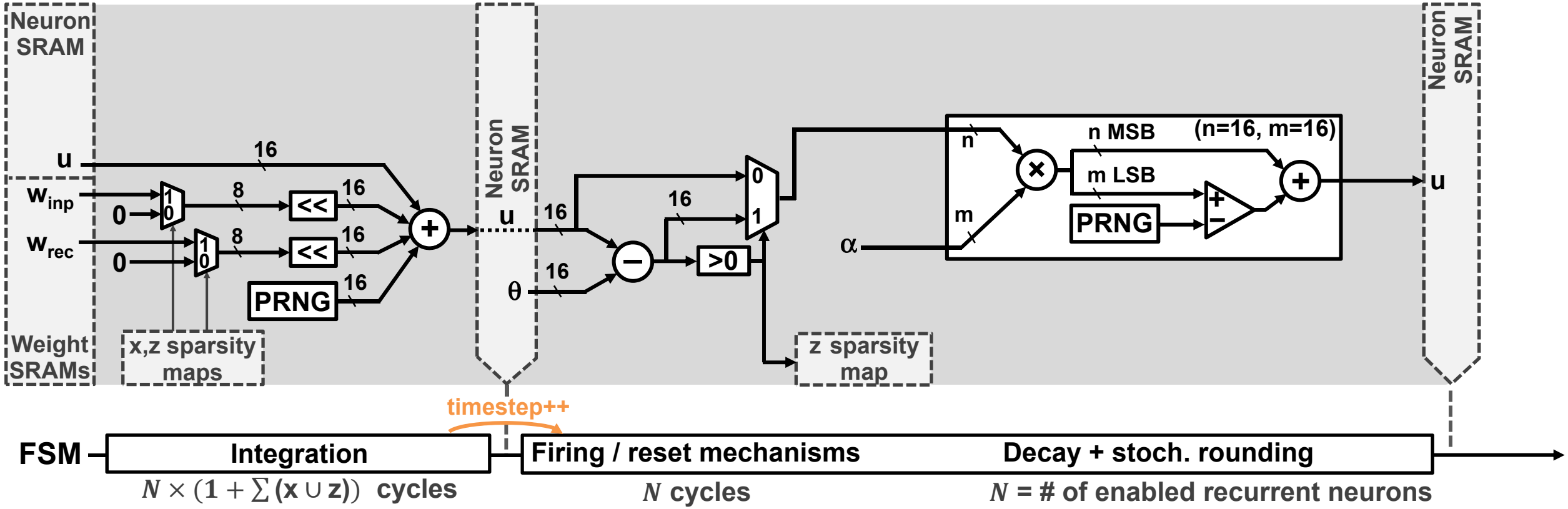
Implemented topology: (256)-256-16:



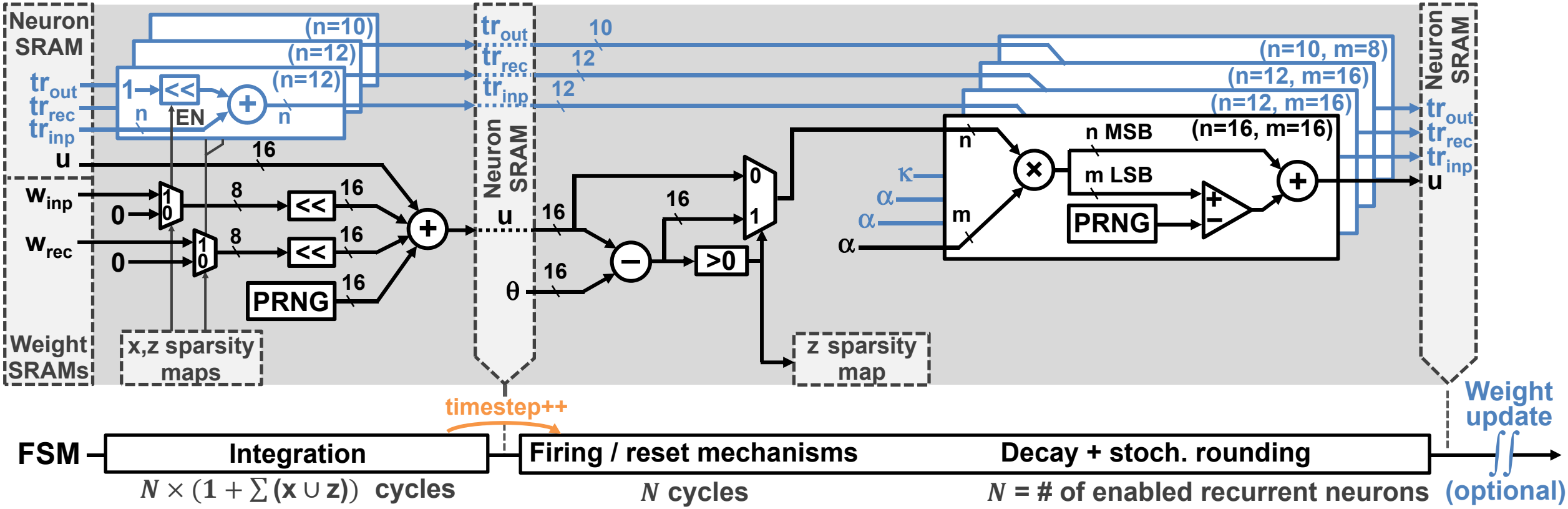
ReckOn – System diagram



Forward path – Neuron update logic

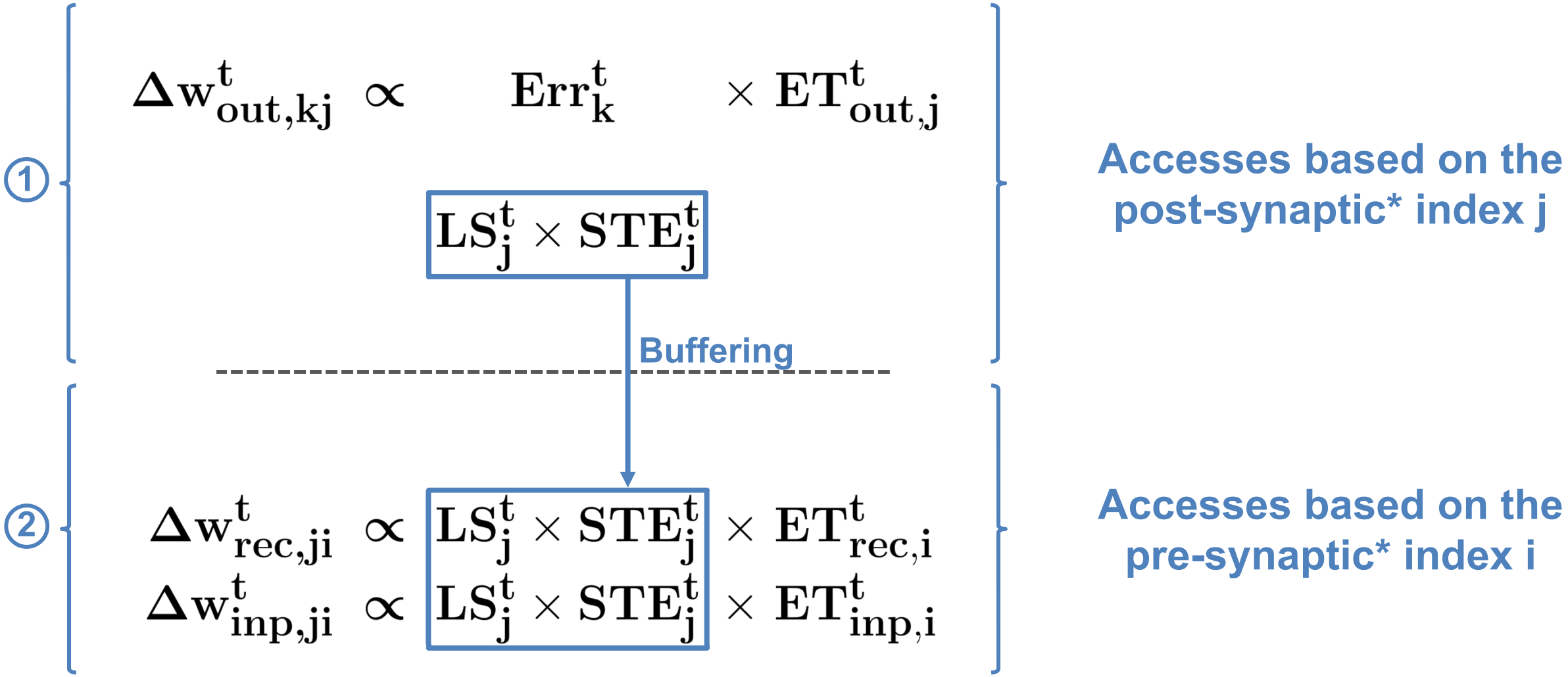


Forward path – Neuron and trace update logic



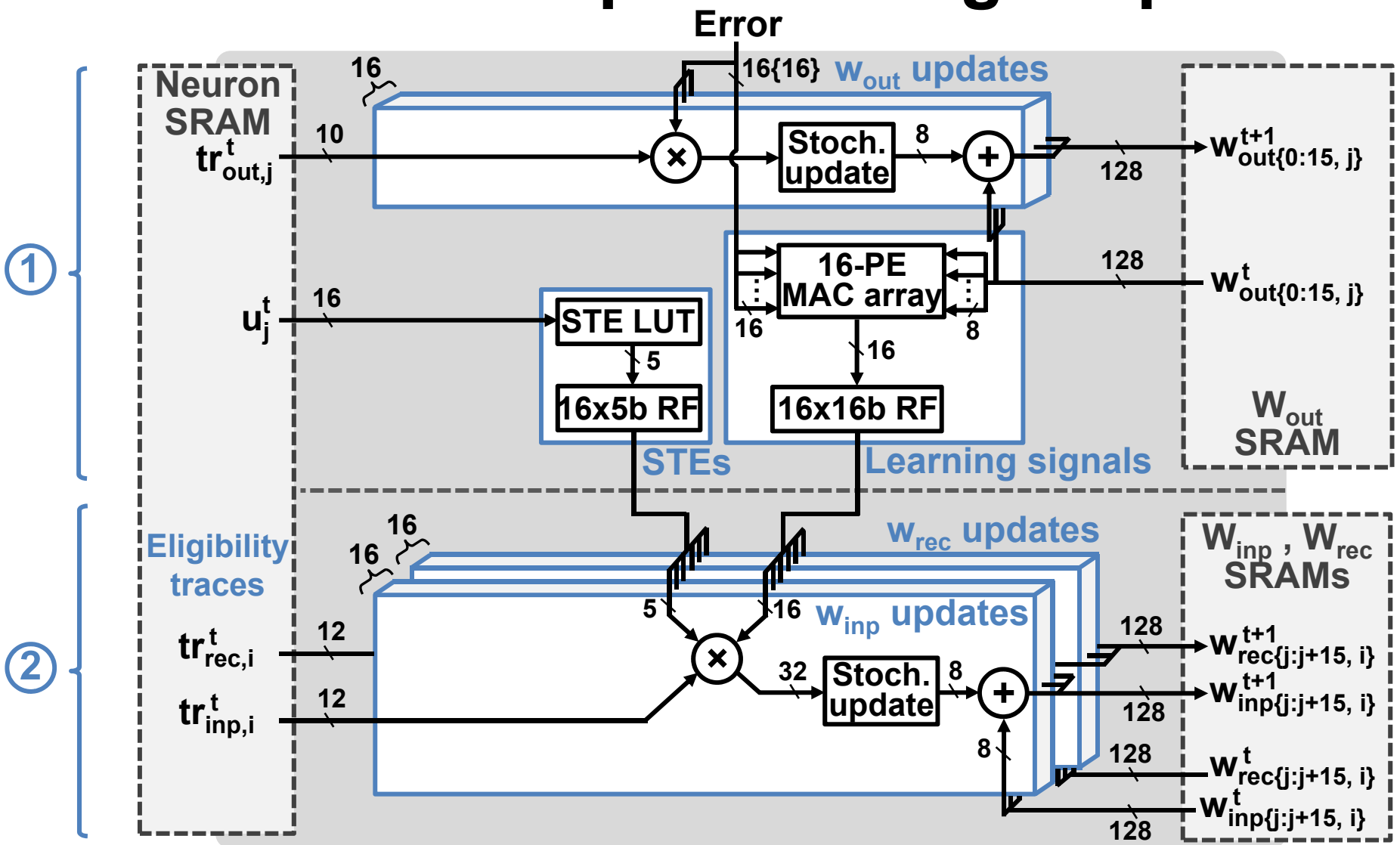
- ETs are processed in parallel with the neurons in the forward path

Local backward path – Weight update logic



* From the point of view of recurrent neurons

Local backward path – Weight update logic



$$\Delta w^t_{out,kj} \propto Err^t_k \times ET^t_{out,j}$$

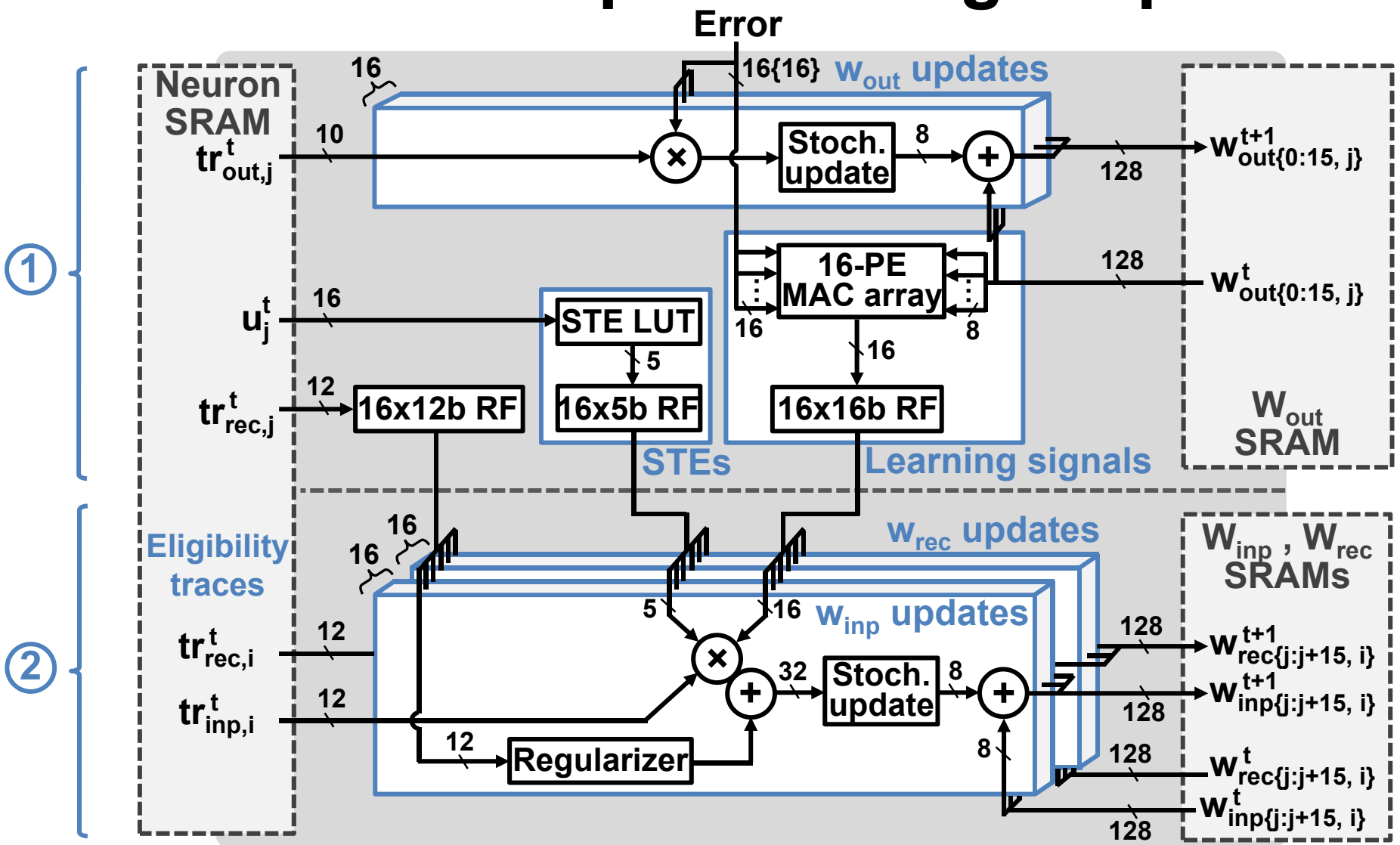
$LS^t_j \times STE^t_j$

Buffering

$$\Delta w^t_{rec,ji} \propto LS^t_j \times STE^t_j \times ET^t_{rec,i}$$

$$\Delta w^t_{inp,ji} \propto LS^t_j \times STE^t_j \times ET^t_{inp,i}$$

Local backward path – Weight update logic



$$\Delta w^t_{out,kj} \propto Err^t_k \times ET^t_{out,j}$$

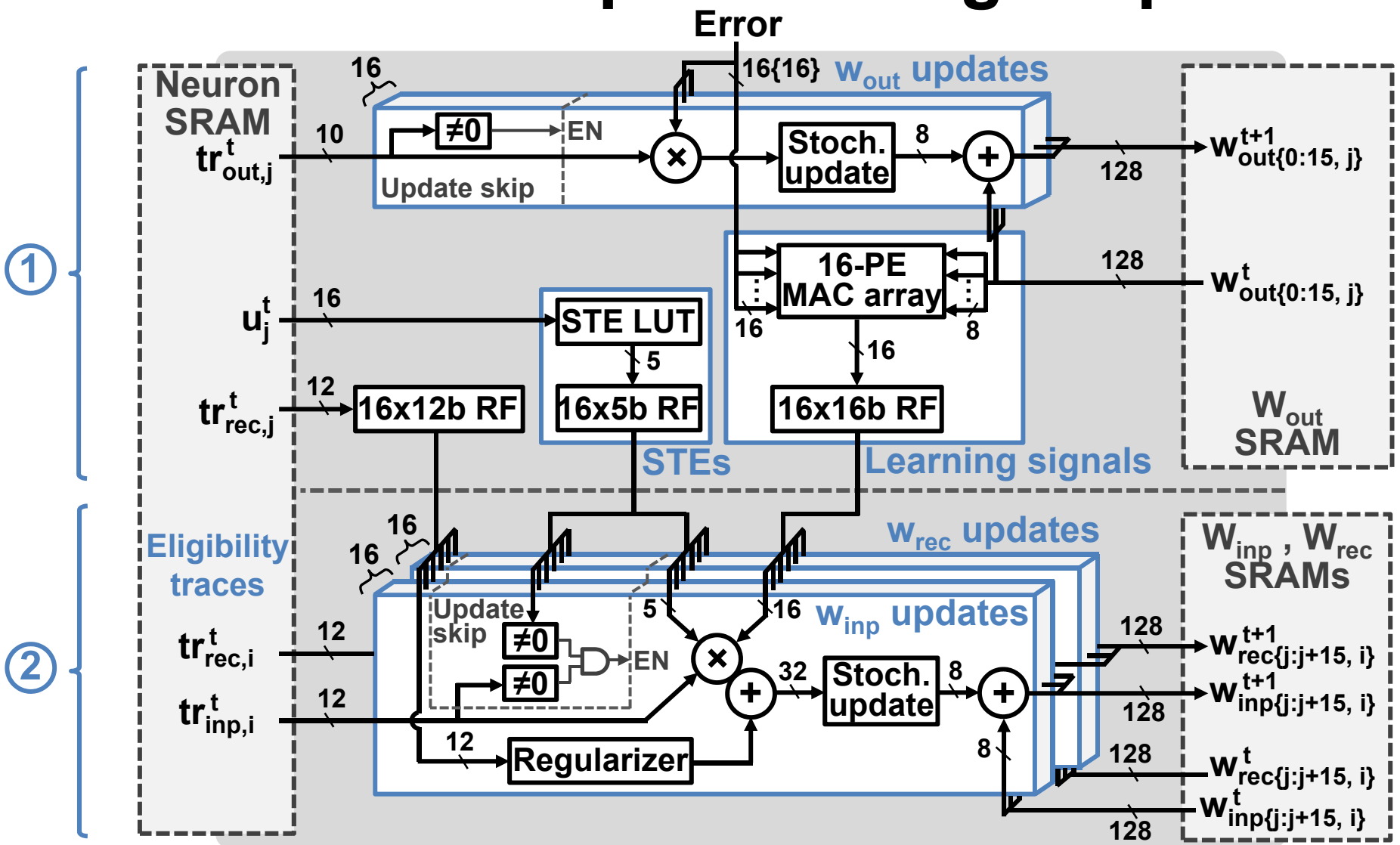
$$LS^t_j \times STE^t_j$$

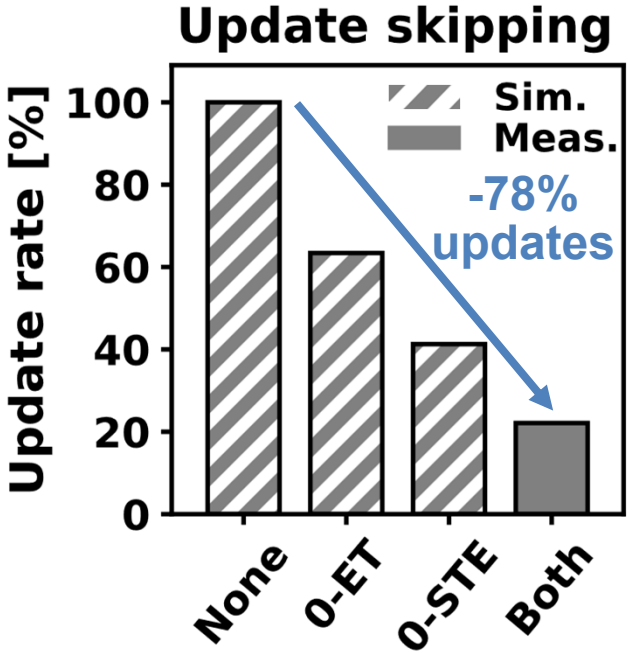
Buffering

$$\Delta w^t_{rec,ji} \propto LS^t_j \times STE^t_j \times ET^t_{rec,i}$$

$$\Delta w^t_{inp,ji} \propto LS^t_j \times STE^t_j \times ET^t_{inp,i}$$

Local backward path – Weight update logic





Outline

I. Context and motivation

II. Training algorithm

- Bio-inspired solution
- Optimizations for space and time locality

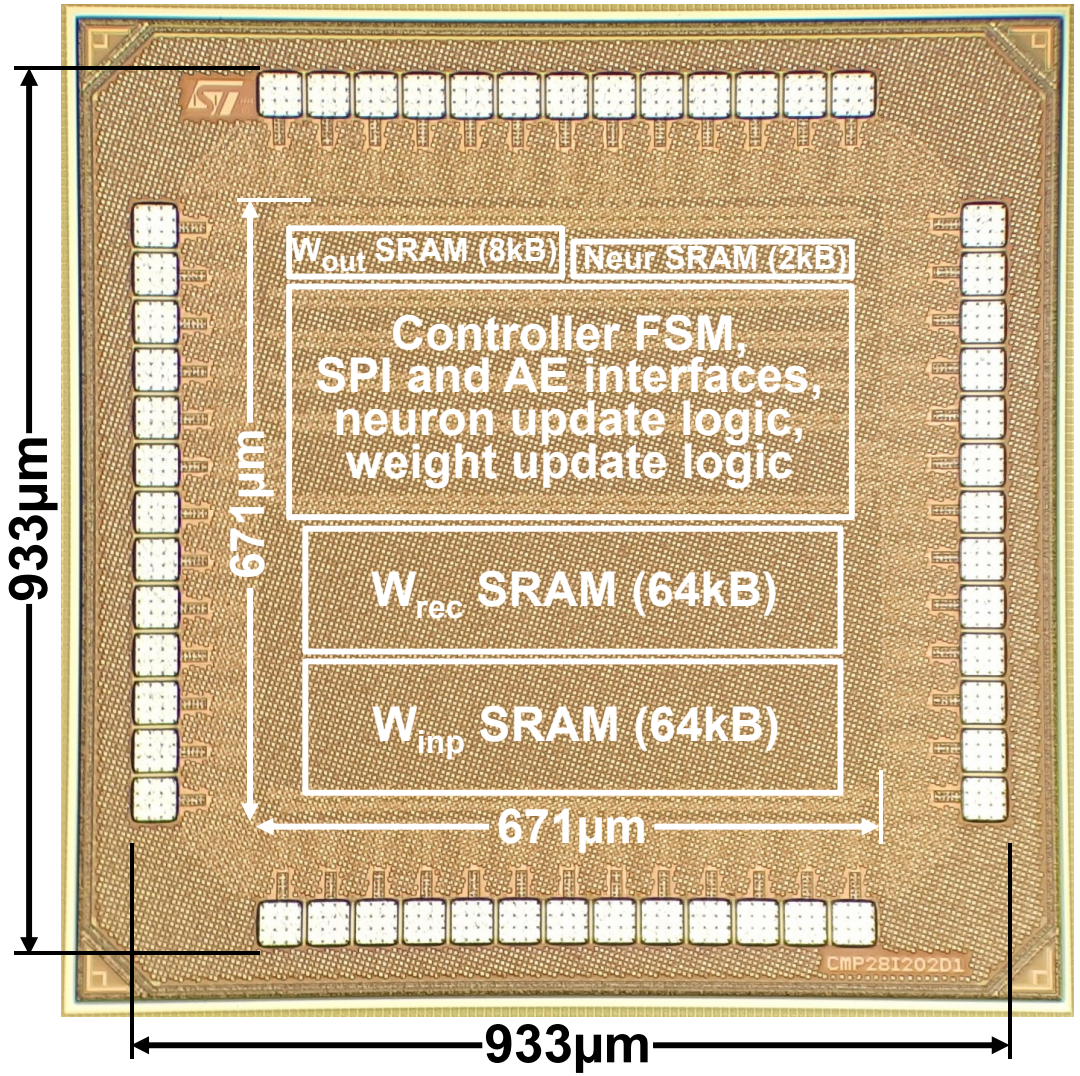
III. Hardware architecture

- Spike-based system overview
- Forward state update logic
- Weight update module and skipping mechanism

IV. Measurement results and comparison

V. Summary

Chip microphotograph and summary



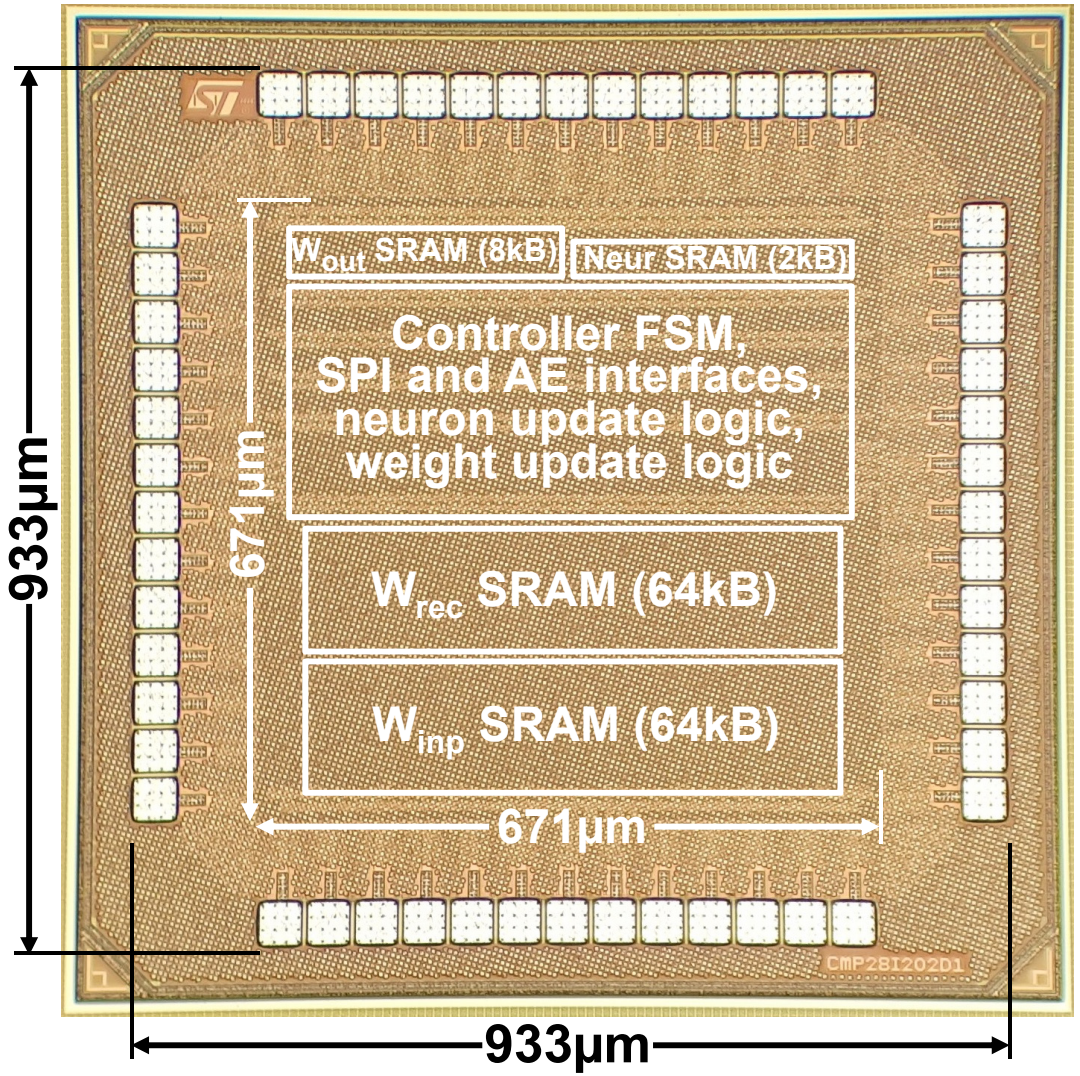
Technology	28nm FDSOI CMOS
Core size	0.67 x 0.67 mm² **0.45mm²**
Die size	0.93 x 0.93 mm²
SRAM	138kB
Network	Spiking RNN
Training timespan	Max. 32k steps
Features	Task-agnostic, raw-data, real-time-to-accelerated on-chip learning over second-long timescales

Supply voltage	0.5 V	0.8 V
Leakage	13µW	55µW
Max. frequency	13 MHz	115 MHz
Worst-case Δt for real time*	5.7ms	0.6ms
Peak energy efficiency	5.3pJ/SOP	12.8pJ/SOP

For high acceleration vs. real-time processing (see paper)

* Full network utilization, 0% sparsity at each timestep

Chip microphotograph and summary



Technology	28nm FDSOI CMOS
Core size	0.67 x 0.67 mm² **0.45mm²**
Die size	0.93 x 0.93 mm²
SRAM	138kB
Network	Spiking RNN
Training timespan	Max. 32k steps
Features	Task-agnostic, raw-data, real-time-to-accelerated on-chip learning over second-long timescales

Supply voltage	0.5 V	0.8 V
Leakage	13µW	55µW
Max. frequency	13 MHz	115 MHz
Worst-case Δt for real time*	5.7ms	0.6ms
Peak energy efficiency	5.3pJ/SOP	12.8pJ/SOP

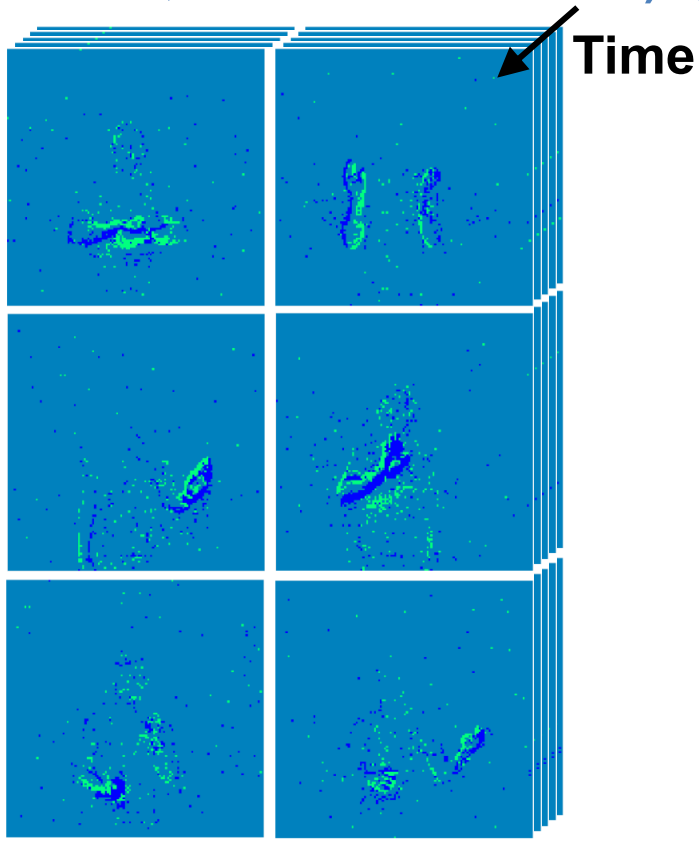
For low-power real-time to low-acceleration processing and learning (this talk)

* Full network utilization, 0% sparsity at each timestep

Task-agnostic learning – Three benchmarks

Vision

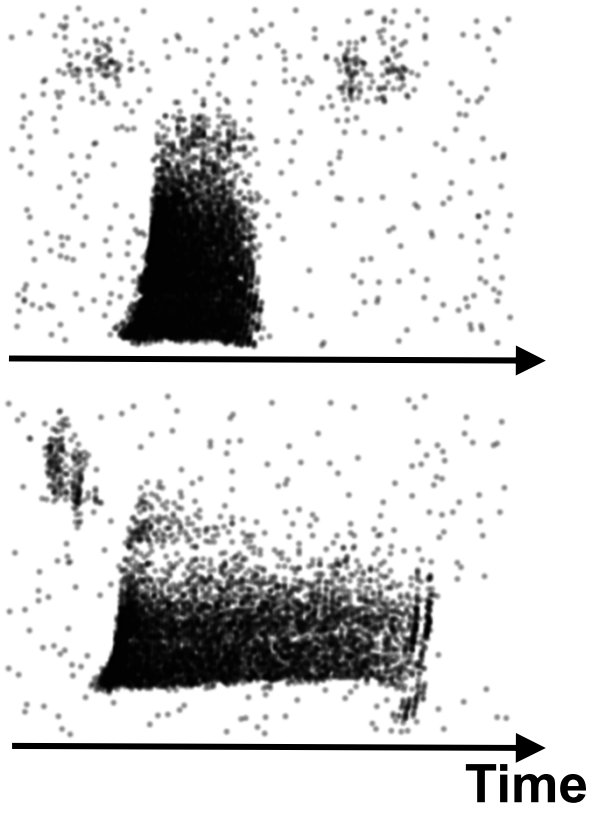
IBM DVS Gestures dataset
(10 classes, shrinked to 16x16)



Accuracy: 87.3%

Audition

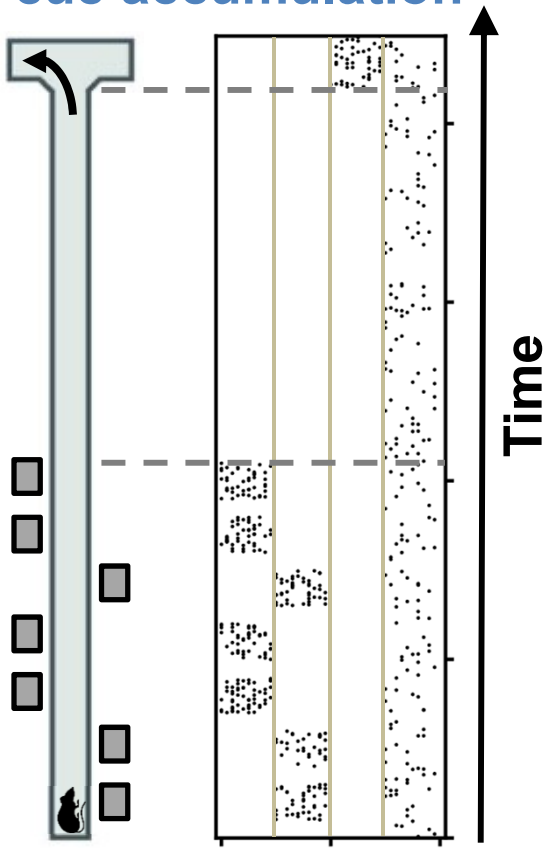
Spiking Heidelberg Digits (EN) dataset
(1-word KWS, 1:1 target vs. filler, 1:3 sub)



Accuracy: 90.7%

Navigation

Delayed-supervision
cue accumulation



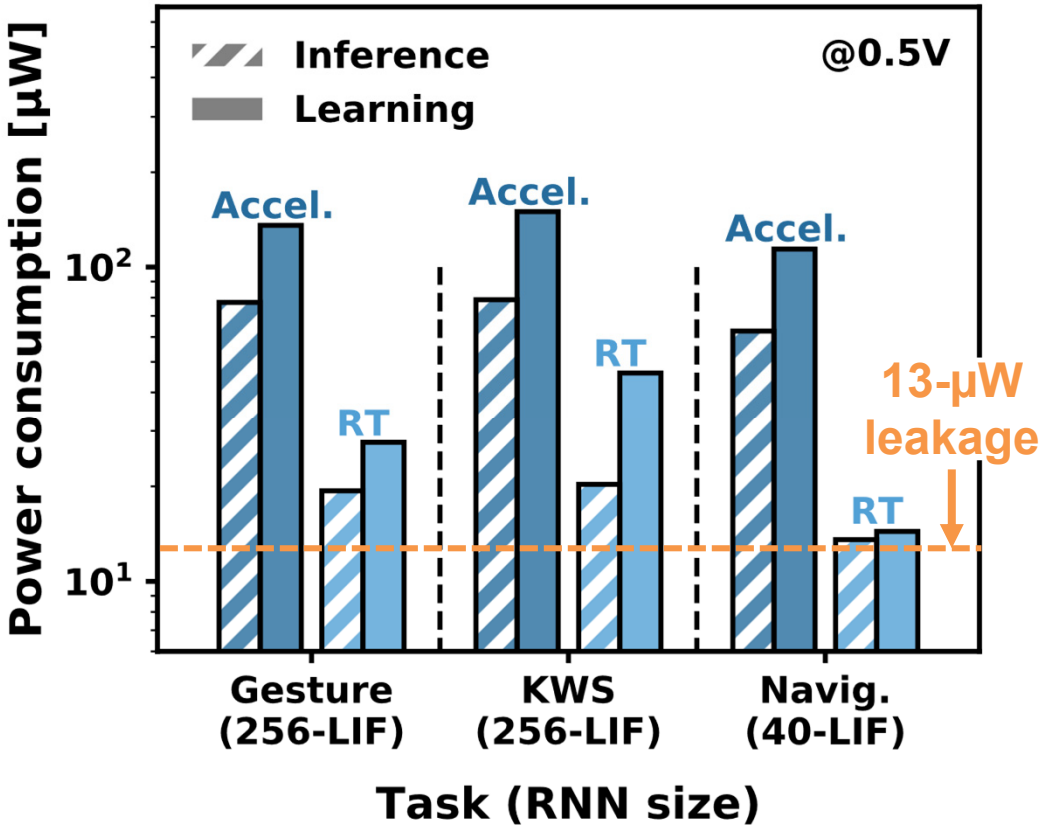
Accuracy: 96.4%

Task-agnostic learning – Power consumption

Vision
IBM DVS Gestures dataset
(10 classes, shrinked to 16x16)

Audition
Spiking Heidelberg Digits (EN) dataset
(1-word KWS, 1:1 target vs. filler, 1:3 sub)

Navigation
Delayed-supervision
cue accumulation



- **At 0.5V, max. acceleration:**
  - 62µW to 79µW (inference)
  - 114µW to 150µW (learning)
- **At 0.5V, real-time processing:**
  - 13.5µW to 20µW (inference)
  - 14.5µW to 46µW (learning)

→ Power budget for always-on processing / adaptation

Comparison with prior works – Specifications

	This work	VLSI'15	VLSI'17	ISSCC'18	VLSI'18	ISSCC'19	ISSCC'18
Technology Implementation Core area Memory Energy metric	28nm Digital 0.45mm² 138kB 5.3pJ/SOP	65nm Digital 1.8mm² 37.6kB 5.7pJ/pix	40nm Mixed-signal 1.3mm² N/A 48.9pJ/pix	65nm Mixed-signal (IMC) 0.8mm² 16kB 0.32pJ/OP	10nm Digital 1.72mm² 896kB 3.8pJ/SOP	65nm Digital 10.1mm² 353kB 0.29pJ/OP	55nm Mixed-signal 3.1mm² 0.4kB 0.32pJ/OP
Network type # Neurons # Synapses (width)	Spiking RNN (256)-256-16 132k (8-bit)	Spiking LCA 4x64 83k (4,5,14-bit)	Spiking LCA 8x64 N/A	SVM 128 8k (16-bit)	Multicore SNN 64x64 1M (7-bit)	Binary NN (784)-200-200-10 194k (14-bit)	ANN (3) - 84 - 3 0.5k (6-bit)
On-chip learning - algorithm - multilayer - dynamics	✓ Mod. stoch. e-prop ✓ Few ms to seconds	✓ SGD ✗ ✗	✓ N/A ✗ ✗	✓ SGD ✗ ✗	✓ STDP ✗ Few ms	✓ Mod. SD ✓ ✗	✓ SGD ✓ ✗

- All designs: from-scratch end-to-end on-chip learning (no off-chip mem)
- Comparison includes both spiking and non-spiking designs

Comparison with prior works – Specifications

	This work	VLSI'15	VLSI'17	ISSCC'18	VLSI'18	ISSCC'19	ISSCC'18
Technology Implementation	28nm Digital	65nm Digital	40nm Mixed-signal	65nm Mixed-signal (IMC)	10nm Digital	65nm Digital	55nm Mixed-signal
Core area	0.45mm²	1.8mm²	1.3mm²	0.8mm²	1.72mm²	10.1mm²	3.1mm²
Memory Energy metric	138kB 5.3pJ/SOP	37.6kB 5.7pJ/pix	N/A 48.9pJ/pix	16kB 0.32pJ/OP	896kB 3.8pJ/SOP	353kB 0.29pJ/OP	0.4kB 0.32pJ/OP
Network type # Neurons	Spiking RNN (256)-256-16	Spiking LCA 4x64	Spiking LCA 8x64	SVM 128	Multicore SNN 64x64	Binary NN (784)-200-200-10	ANN (3) - 84 - 3
# Synapses (width)	132k (8-bit)	83k (4,5,14-bit)	N/A	8k (16-bit)	1M (7-bit)	194k (14-bit)	0.5k (6-bit)
On-chip learning - algorithm - multilayer - dynamics	✓ Mod. stoch. e-prop ✓ Few ms to seconds	✓ SGD ✗ ✗	✓ N/A ✗ ✗	✓ SGD ✗ ✗	✓ STDP ✗ Few ms	✓ Mod. SD ✓ ✗	✓ SGD ✓ ✗

- All designs: from-scratch end-to-end on-chip learning (no off-chip mem)
- Comparison includes both spiking and non-spiking designs
- Unique properties of ReckOn:
  - lowest area, highest synaptic density (with & without tech. normalization)
  - multi-layer learning over second-long timescales

Comparison with prior works – Benchmarks

	This work	VLSI'15	VLSI'17	ISSCC'18	VLSI'18	ISSCC'19	ISSCC'18
Task	Hand gesture classif. Keyword spotting Navigation	Image classif.	Image classif.	Image classif.	Image classif.	Image classif.	Obstacle avoid.
Dataset	IBM DVS Gestures Spiking Heidelberg Digits Delayed cue integration	MNIST	MNIST	MIT CBCL	MNIST	MNIST	Custom autonomous robot
Average input data depth	Gest: 1318 steps @Δt=5ms KWS: 104 steps @Δt=5ms Nav: 2250 steps @Δt=1ms	1 frame	1 frame	1 frame	1 frame	1 frame	N/A (1-step decisions)
Accuracy with on-chip training	Gest: 87.3% @10classes KWS: 90.7% @1word Nav: 96.4% @2decisions	84%-90%	88%	91.6%	89%	97.8%	N/A
Power (infer / learn)	Gest: 77µW / 135µW KWS: 79µW / 150µW Nav: 62µW / 114µW	268mW / 526mW	87mW / N/A	1.3mW / 3.1mW	6.2mW / N/A	23.6mW / 23.1mW	690µW / N/A
Energy per step (infer / learn)	Gest: 35nJ / 85nJ KWS: 42nJ / 178nJ Nav: 0.6nJ / 1.5nJ	27-162nJ / 94.7µJ	50.1nJ / N/A	42pJ / 150pJ	1.0µJ / N/A	236nJ / 254nJ	0.69nJ / 1.5nJ
Throughput in steps/sec (infer/learn)	Gest: 2.2k / 1.6k KWS: 1.9k / 0.8k Nav: 98k / 76k	9.9M-1.6M / 5.5k	1.7M / N/A	32M / ~21M	6.25k / N/A	100k / 94.3k	254k / 118k

- **Task-agnostic learning**

Comparison with prior works – Benchmarks

	This work	VLSI'15	VLSI'17	ISSCC'18	VLSI'18	ISSCC'19	ISSCC'18
Task	Hand gesture classif. Keyword spotting Navigation	Image classif.	Image classif.	Image classif.	Image classif.	Image classif.	Obstacle avoid.
Dataset	IBM DVS Gestures Spiking Heidelberg Digits Delayed cue integration	MNIST	MNIST	MIT CBCL	MNIST	MNIST	Custom autonomous robot
Average input data depth	Gest: 1318 steps @Δt=5ms KWS: 104 steps @Δt=5ms Nav: 2250 steps @Δt=1ms	1 frame	1 frame	1 frame	1 frame	1 frame	N/A (1-step decisions)
Accuracy with on-chip training	Gest: 87.3% @10classes KWS: 90.7% @1word Nav: 96.4% @2decisions	84%-90%	88%	91.6%	89%	97.8%	N/A
Power (infer / learn)	Gest: 77µW / 135µW KWS: 79µW / 150µW Nav: 62µW / 114µW	268mW / 526mW	87mW / N/A	1.3mW / 3.1mW	6.2mW / N/A	23.6mW / 23.1mW	690µW / N/A
Energy per step (infer / learn)	Gest: 35nJ / 85nJ KWS: 42nJ / 178nJ Nav: 0.6nJ / 1.5nJ	27-162nJ / 94.7µJ	50.1nJ / N/A	42pJ / 150pJ	1.0µJ / N/A	236nJ / 254nJ	0.69nJ / 1.5nJ
Throughput in steps/sec (infer/learn)	Gest: 2.2k / 1.6k KWS: 1.9k / 0.8k Nav: 98k / 76k	9.9M-1.6M / 5.5k	1.7M / N/A	32M / ~21M	6.25k / N/A	100k / 94.3k	254k / 118k

- **Second-long-timescale learning at 1-to-5-ms temporal resolution**

Comparison with prior works – Benchmarks

	This work	VLSI'15	VLSI'17	ISSCC'18	VLSI'18	ISSCC'19	ISSCC'18
Task	Hand gesture classif. Keyword spotting Navigation	Image classif.	Image classif.	Image classif.	Image classif.	Image classif.	Obstacle avoid.
Dataset	IBM DVS Gestures Spiking Heidelberg Digits Delayed cue integration	MNIST	MNIST	MIT CBCL	MNIST	MNIST	Custom autonomous robot
Average input data depth	Gest: 1318 steps @Δt=5ms KWS: 104 steps @Δt=5ms Nav: 2250 steps @Δt=1ms	1 frame	1 frame	1 frame	1 frame	1 frame	N/A (1-step decisions)
Accuracy with on-chip training	Gest: 87.3% @10classes KWS: 90.7% @1word Nav: 96.4% @2decisions *	84%-90%	88%	91.6%	89%	97.8%	N/A
Power (infer / learn)	Gest: 77µW / 135µW KWS: 79µW / 150µW Nav: 62µW / 114µW	268mW / 526mW	87mW / N/A	1.3mW / 3.1mW	6.2mW / N/A	23.6mW / 23.1mW	690µW / N/A
Energy per step (infer / learn)	Gest: 35nJ / 85nJ KWS: 42nJ / 178nJ Nav: 0.6nJ / 1.5nJ	27-162nJ / 94.7µJ	50.1nJ / N/A	42pJ / 150pJ	1.0µJ / N/A	236nJ / 254nJ	0.69nJ / 1.5nJ
Throughput in steps/sec (infer/learn)	Gest: 2.2k / 1.6k KWS: 1.9k / 0.8k Nav: 98k / 76k	9.9M-1.6M / 5.5k	1.7M / N/A	32M / ~21M	6.25k / N/A	100k / 94.3k	254k / 118k

*At 0.5V, 13MHz

- Competitive E/step, low-power real-time to low-acceleration processing

Outline

I. Context and motivation

II. Training algorithm

- Bio-inspired solution
- Optimizations for space and time locality

III. Hardware architecture

- Spike-based system overview
- Forward state update logic
- Weight update module and skipping mechanism

IV. Measurement results and comparison

V. Summary

Summary

- ReckOn achieves end-to-end on-chip learning over second-long timescales while keeping a milli-second temporal resolution
- it is a low-cost solution: 0.45-mm² core area, 5.3pJ/SOP at 0.5V, and 0.8-% memory overhead vs. inference-only network
- it exploits a spike-based representation for task-agnostic learning toward user customization and chip repurposing at the edge

What next?

- Data at the edge is not cheap ⟶ Few-shot learning
- Scalability to more complex tasks ➔ Sparse/hierarchical connectivity

Acknowledgments: Funding by the EU and the SNSF (No. 826655, 20CH21186999/1). Circuit fabrication by ST Microelectronics.

ReckOn is an open-source project

A first release of the Verilog code and documentation of ReckOn is publicly available:

https://github.com/ChFrenkel/ReckOn

Note: unpublished parts have been removed from this first release of the code + doc.

Thank you!

✉ charlotte@ini.uzh.ch