ASADI: Accelerating Sparse Attention using Diagonal-based In-situ Computing

Huize Li, Zhaoying Li, Zhenyu Bai, and Tulika Mitra

School of Computing,
National University of Singapore

HPCA 2024 Session A (Chair: B)
4th December 2023, Edinburgh, Scotland

1. Background
2. Motivations
3. DIA-based PUM
4. Architecture and Dataflow
5. Evaluation

Transformer and Attention Mechanism

- Transformer-based [1] large models achieve SOTA performance on various NLP and CV tasks

T	Model	Average	ARC	HellaSwa	MMLU	Truthful	Winogran	Type	#Params
	TigerResearch/tigerbot-70b-chat-v2	69.76	87.03	82.83	66	75.4	79.16	fine-tun	70
	bhenrym14/platypus-yi-34b	68.96	68.43	85.21	78.13	54.48	84.06	instruct	34.39
	01-ai/Yi-34B	68.68	64.59	85.69	76.35	56.23	83.03	pretrain	34
	chargoddard/Yi-34B-Llama	68.4	64.59	85.63	76.31	55.6	82.79	pretrain	34.39
	MayaPH/GodziLLa2-70B	67.01	71.42	87.53	69.88	61.54	83.19	instruct	68.98
	sequelbox/StellarBright	66.98	72.95	87.82	71.17	64.46	83.27	fine-tun	69.24
	garage-bAInd/Platypus2-70B-instruct	66.89	71.84	87.94	70.48	62.26	82.72	fine-tun	68.72
	upstage/SOLAR-0-70b-16bit	66.88	71.08	87.89	70.58	62.25	83.58	instruct	68.72
	Sao10K/Euryale-1.3-L2-70B	66.58	70.82	87.92	70.39	59.85	82.79	fine-tun	68.72
	psmathur/model_101	66.55	68.69	86.42	69.92	58.85	82.08	fine-tun	68.72
	OpenBuddy/openbuddy-llama2-70b-v10.1-	66.47	61.86	83.13	67.41	56.18	80.11	fine-tun	68.76
	budecosystem/genz-70b	66.34	71.42	87.99	70.78	62.66	83.5	instruct	68.72

Figure 1: From HuggingFace's Open LLM Leaderboard [2]

Transformer and Attention Mechanism

- Attention mechanism with quadratic complexity to sequence length is the bottleneck of Transformer-based models

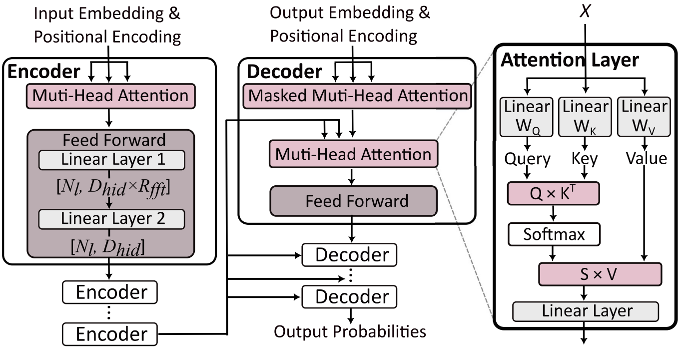
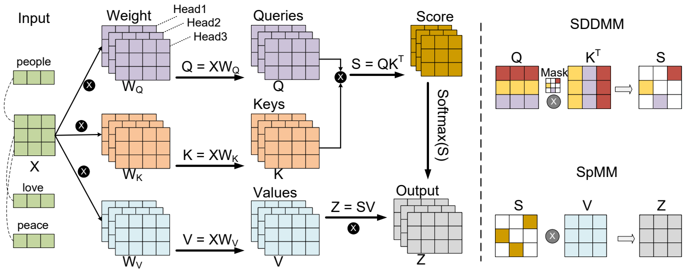


Figure 2: End-to-end Transformer

Basics of Sparse Attention

- Sparse attention is proposed to reduce computational complexity by pruning weak connected tokens, e.g., Longformer [3], DOTA [4], and Sanger [5]



(a) Computation stages of dense attention mechanism (b) Sparse attention calculation

Figure 3: (a) Multi-head attention mechanism, and (b) Converting GEMM to SDDMM and SpMM

Basics of Sparse Attention

- **Two types of sparse attention:**
- Static sparse: Pre-determining the sparse mask matrix before receiving the input sequences, e.g., Longformer [3]
- Dynamic sparse: Employing a quantize-and-pruning phase to determine the sparse mask matrix, e.g., Sanger [5]

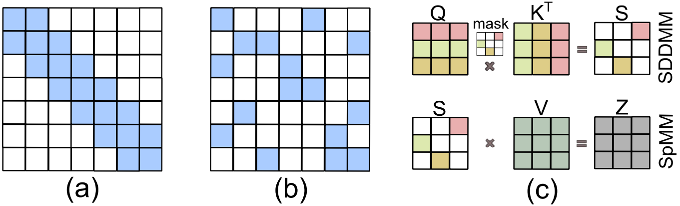


Figure 4: (a) Mask matrix of Longformer, (b) Mask matrix of Sanger, (c) Converting GEMM to SDDMM and SpMM

Basics of Sparse Compression Formats

- The most popular compression formats are CSR, CSC, and COO, and we use CSR as an example
- This paper focuses on diagonal locality, so we also introduce DIA format

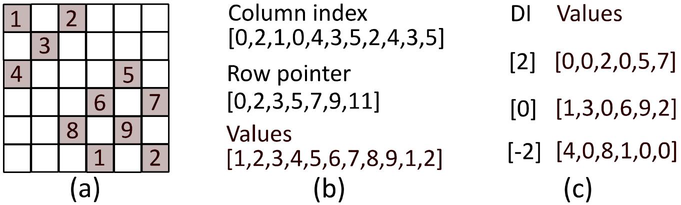


Figure 5: (a) An example of sparse mask matrix, (b) CSR format, and (c) DIA format

Basics of Memroy-centric Computing

- Sparse attention produces many irregular intermediate matrices, making it a memory-intensive kernel
- Memory-centric platforms are promising to accelerate memory-intensive kernels, such as PIM and PUM
- This paper focuses on PUM (processing-using-memory), i.e., in-situ computing

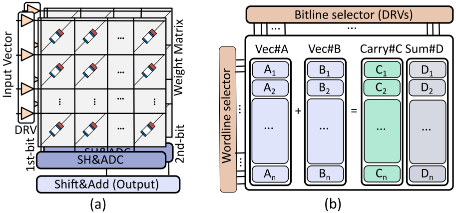


Figure 6: (a) Analog in-situ computing of ReRAM array, (b) Digital in-situ computing of ReRAM array

Observations from Current Sparse Attention Accelerators

- **Observation#1:** Diagonal locality is prevalent in both static and dynamic sparse attention
- **Observation#2:** Current PIM-based sparse attention accelerators have high on-chip communication overhead

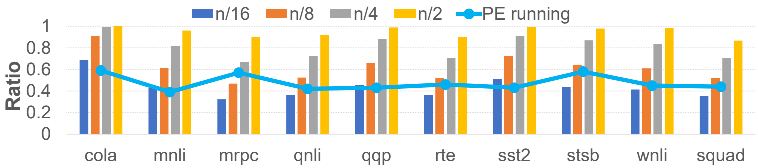


Figure 7: (a) The distribution of non-zeros in Sanger [5] with various ω (four bars), (b) The ratio of on-chip PE runtime to overall PIM chip runtime (broken lines)

Our Goals from the Observations

- **Opportunity from Observation#1:** Current accelerators convert diagonal locality into row/column locality because the lack of DIA-based computation paradigm, e.g., Sanger [5] and SPRINT [6]
- **Goal#1:** Designing a new matrix multiplication computation paradigm to efficiently support the DIA format

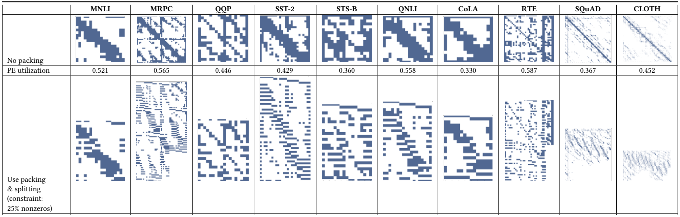


Figure 8: (a) The sparse plots obtained from Sanger [5]

Our Goals from the Observations

- **Opportunity from Observation#2:** Utilizing row/column wise SpMM/SDDMM computation paradigm to PUM platforms will introduce many zeros, greatly decreasing PE utilization (Quantitative analysis in the paper)
- **Goal#2:** Designing DIA-based SpMM/SDDMM computation paradigm for PUM platforms to increase PE utilization

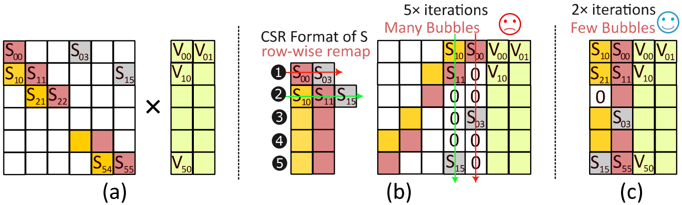


Figure 9: (a) SpMM between S and V, (b) In-situ computing with CSR format of S, (c) In-situ computing with DIA format of S

Our Goals from the Observations

- **Opportunity from Observation#2:** Utilizing row/column wise SpMM/SDDMM computation paradigm to PUM platforms will introduce many zeros, greatly decreasing PE utilization (Quantitative analysis in the paper)
- **Goal#2:** Designing DIA-based SpMM/SDDMM computation paradigm for PUM platforms to increase PE utilization

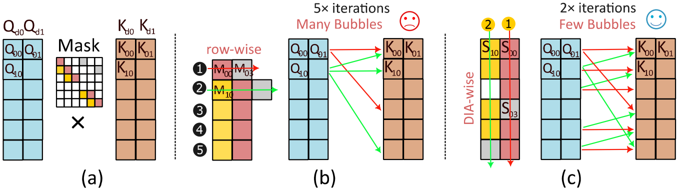


Figure 10: (a) SDDMM between Q and K^T, (b) In-situ computing with CSR format of M, (c) In-situ computing with DIA format of M

Novel Compression Format

- **Classic bubble-free DIA:** Diagonal with all non-zeros, like Longformer [3], which has perfect diagonal locality
- **Advantages of DIA:** Each diagonal at least has $n - \frac{\omega}{2}$ non-zeros while each row has up to ω non-zeros. Assuming $\omega = \frac{n}{8}$, one diagonal has $7.5\times$ non-zeros than one row

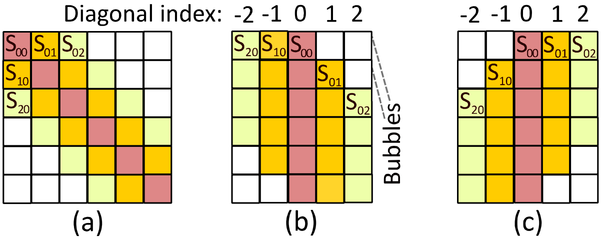


Figure 11: (a) Sparse S matrix ($\omega = 5$ and $n = 6$) without bubbles, (b) Bubble-free DIA compression, (c) Decompressed DIA format

Novel Compression Format

- **Bubble-containing DIA:** Diagonal with zeros and non-zeros, like Sanger [5], which has good diagonal locality in ω area
- **Enhance diagonal locality:** Move non-zeros in other region to the bubbles of ω region, maintaining column coordinates

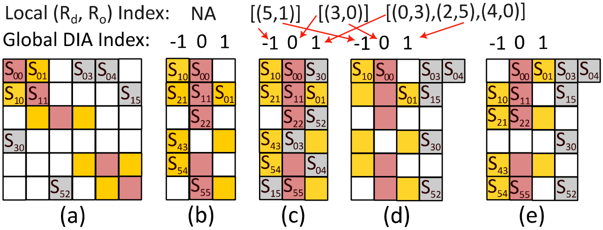
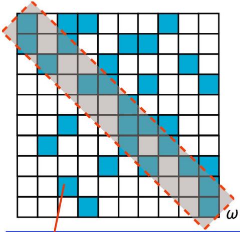


Figure 12: (a) Sparse S matrix with bubbles, (b) Bubble-free DIA compression, (c) Bubble-containing DIA compression, (d) Decompress non-central diagonals, (e) Decompress central diagonals

Novel Compression Format

- **How to select ω:** We refer to the central ω diagonal region as ω region while referring the rest as other region. After calculation, we choose $\frac{n}{8}$ as the default configuration of ω



$\omega = n/4$
$NNZ_\omega = 0.7 * NNZ$
$NNZ_\omega = 0.07n^2$
$NAE_\omega = 0.25n^2$
$NZ_\omega = 0.18n^2$
$NNZ_o = 0.03n^2$
$NZ_\omega \gg NNZ_o$ ☹
Too many bubbles

$\omega = n/8$
$NNZ_\omega = 0.5 * NNZ$
$NNZ_\omega = 0.05n^2$
$NAE_\omega = 0.125n^2$
$NZ_\omega = 0.075n^2$
$NNZ_o = 0.05n^2$
$NZ_\omega > NNZ_o$ ☺
Matched

$\omega = n/16$
$NNZ_\omega = 0.4 * NNZ$
$NNZ_\omega = 0.04n^2$
$NAE_\omega = 0.0625n^2$
$NZ_\omega = 0.0225n^2$
$NNZ_o = 0.06n^2$
$NNZ_o \gg NZ_\omega$ ☹
No enough bubbles

We will move the non-zero elements in other region to the bubbles of ω region. Therefore, it is necessary to ensure that NZ_ω is slightly bigger than NNZ_o

NAE: number of all elements (n^2)
NNZ: number of non-zeros ($0.1*n^2$)
NZ: number of zeros ($0.9*n^2$)
NAE_ω: NAE of ω region ($\omega*n$)
NNZ_ω: NNZ of ω region (from Figure 5)
NZ_ω: NZ of ω region ($NAE_\omega - NNZ_\omega$)
NNZ_o: NNZ of other region ($NNZ - NNZ_\omega$)

Figure 13: Details to calculate NZ_ω and NNZ_o

Before We Start

- **See a simple example first:** Assuming we only have one diagonal, then what should we do

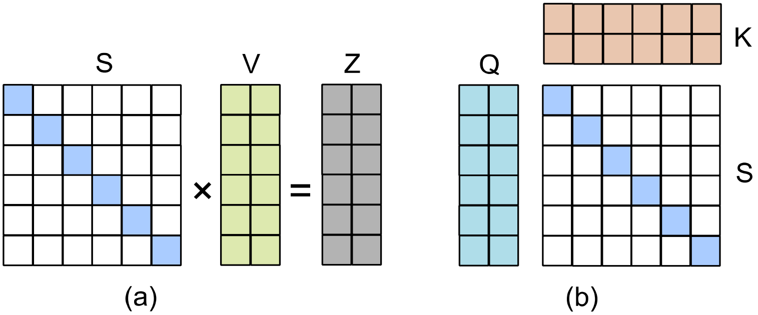


Figure 14: (a) SpMM between S and V, (b) SDDMM between Q and K^T

In-situ $S \times V$

- **Details of computation paradigm:** The following Figure presents a visualization version of our method, the pseudo-code and detailed description are in the paper

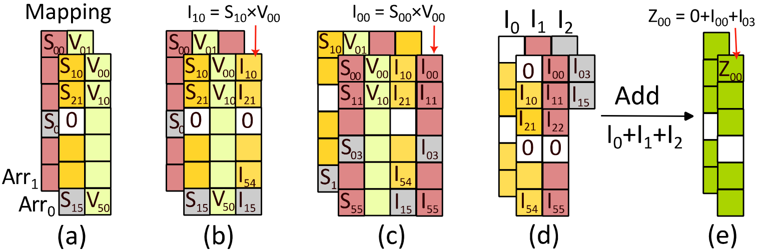


Figure 15: (a) Mapping matrices S and V to two ReRAM arrays, (b) Intermediate results of the first iteration of vector-vector multiplication, (c) Intermediate results of the second iteration of vector-vector multiplication, (d) Decompressed intermediate results, (e) Output Z matrix

In-situ $Q \times K^T$

- **Details of computation paradigm:** The following Figure presents a visualization version of our method, the pseudo-code and detailed description are in the paper

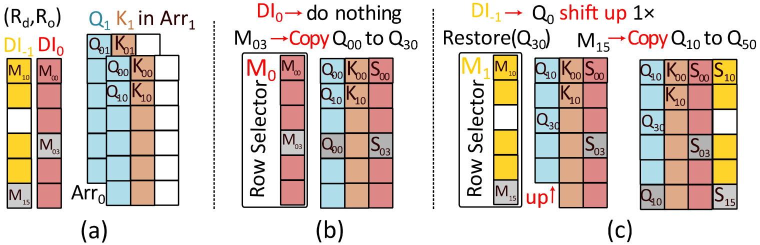


Figure 16: (a) Matrices Q and K in two ReRAM arrays, (b) Vector-vector multiplication of DI_0, (c) Vector-vector multiplication of DI_{-1}

In-situ $Q \times K^T$

- **Details of computation paradigm:** The following Figure presents a visualization version of our method, the pseudo-code and detailed description are in the paper

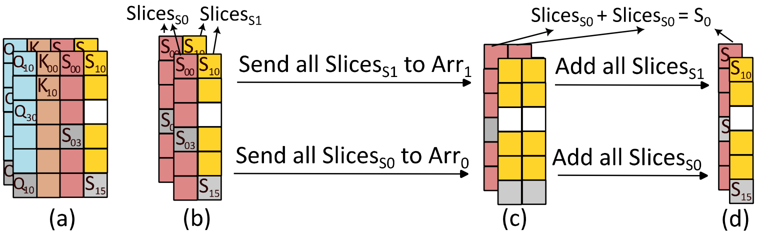


Figure 17: (a) Two slices of $Q_0 \times K_0$ and $Q_1 \times K_1$, (b) We refer the slices of DI_0 as $Slices_{S0}$ and DI_{-1} to $Slices_{S1}$, (c) All $Slices_{S0}$ and $Slices_{S1}$ are transferred to the same ReRAM array, (d) Results of DIA-based S matrix

In-situ Linear Layer

- **Details of computation paradigm:** The following Figure presents a visualization version of our method. $Q = I \times W_Q$

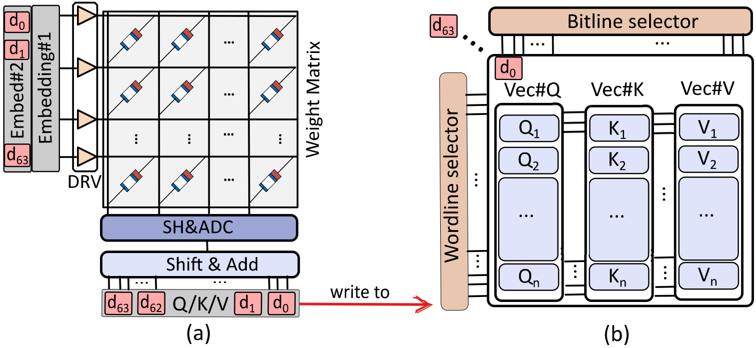


Figure 18: (a) Linear layer computation, (b) The data mapping of matrices Q, K, and V

In-situ Softmax

- **Details of computation paradigm:** The following Figure presents a visualization version of 2^x. Since $e^x = 2^{x \log_2 e}$, we can perform in-situ vector-vector multiplication $y = x \log_2 e$ to get vector y, followed by 2^y to obtain e^x.
  $\textbf{\textit{softmax}}(s_i) = \frac{e^{s_i - s_{max}}}{\sum_{c=1}^{n} e^{s_c - s_{max}}}$

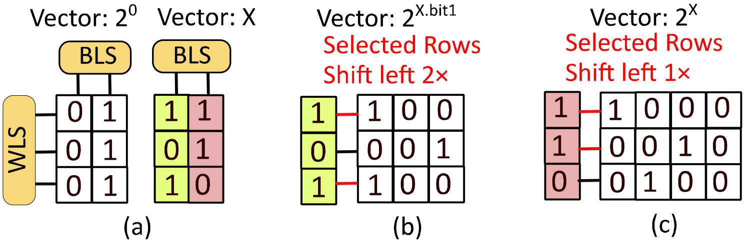


Figure 19: (a) Vector 2^0 and vector x, (b) Shift operation of bit1, (c) Shift operation of bit0

1. Background
2. Motivations
3. DIA-based PUM
4. **Architecture and Dataflow**
5. Evaluation

Overall Architecture

- **Details of architecture:** ASADI contains multiple En-PEs and De-PEs; Each En/De-PE contains two analog modules and one digital module.

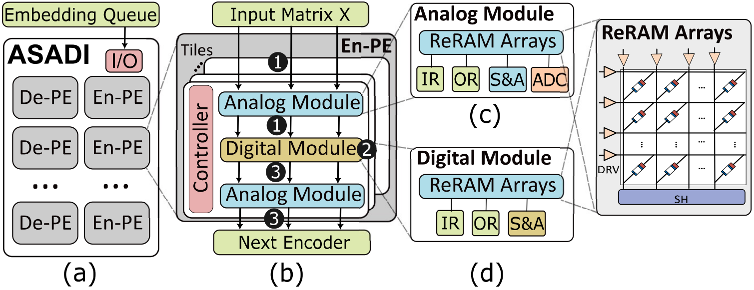
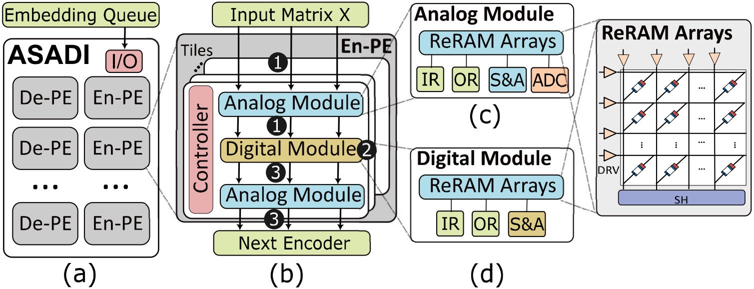


Figure 20: (a) Overall ASADI architecture, (b) Details of one En-PE, (c) Details of the analog module, (d) Details of the digital module

Dataflow

- **Inter-PE dataflow:** One En/De-PE for one Encoder/Decoder layer; we use pipeline to achieve parallelism between PEs
- **Intra-PE dataflow:** The generation of matrices Q, K, and V (❶). The digital module performs the in-situ $Q \times K^T$, $S \times V$, and softmax operations (❷). The matrix Z is sequentially read and sent to the second analog module (❸).

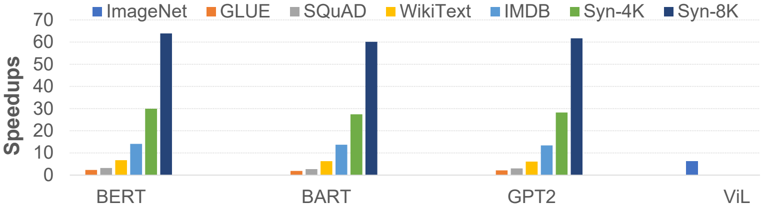


Evaluation Setup

- **Models:** BERT, BART, GPT-2, and ViL
- **Datasets:** GLUE, SQuAD v1.1, WikeText-2, IMDB, ImageNet-1K, Syn-4K, and Syn-8K
- **Baseline:** The PIM baseline employs Samsung's novel function-in-memory DRAM (FIMDRAM). We use Ramulator-PIM to obtain latency and energy consumption.
- **Other platforms:** NVIDIA RTX A6000, SPRINT, and CPSAA
- **Configuration of ASADI:** Table 1 of the paper
- **Pre-processing:** Our code is modified from the GitHub project of Sanger [5]. All models and datasets are obtained from Hugging Face's models library [2] and datasets library.

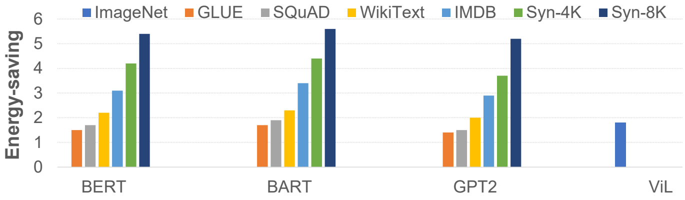
Overall Performance and Analysis

- **ASADI vs. PIM baseline**
- **Results:** ViL: 6.4× speedup; BERT: 2.3× to 63.7× on GLUE, SQuAD, WikeText, IMDB, Syn-4K, and Syn-8k datasets. BART: 1.9× to 60.1× speedups; GPT2: 2.1× to 61.7×.
- **Reasons:** Reducing on-chip random access; The PIM baseline uses near-memory computation, where the on-chip logic units need to random access many cross-bank data.



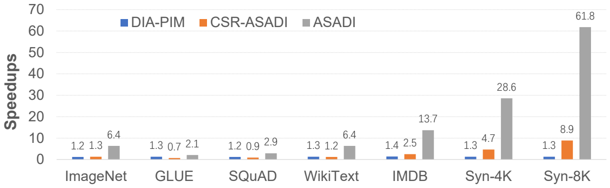
Overall Energy Saving and Analysis

- **ASADI vs. PIM baseline**
- **Results:** ViL: 1.8×; BERT, BART, GPT-2: 2.3× to 63.7× across all datasets.
- **Reasons:** The reduced data transfers between on-chip memory and PEs. With increasing sequence length, ASADI is capable of reducing more on-chip transfers, which results in more energy savings.



Software and hardware efficiency

- **DIA-PIM:** DIA format involves both compression and decompression phases. While DIA-PIM benefits from the compression phase, it does not gain any advantage from the decompression phase, as it requires the same number of cross-bank transfers as the CSR format.
- **CSR-ASADI:** The CSR-based SDDMM and SpMM computation paradigms involve many bubbles in PUM platform and severely impact the overall parallelism.



Latency and Energy Breakdown

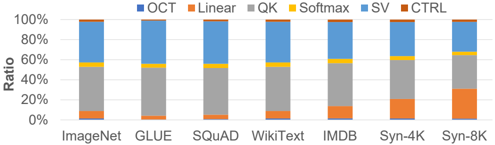


Figure 21: Latency breakdown. OCT and CTRL: less than 4%; Softmax: 5%; QK and SV: more than 80%.

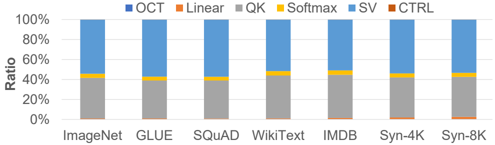


Figure 22: Energy breakdown. Digital module: more than 98%; Linear layer: 1%; CTRL: less than 1%

Sensitivity Analysis

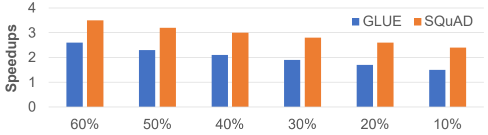


Figure 23: Impact of diagonal locality. Performance degradation as diagonal locality decreases. Lower diagonal locality leads to more bubbles in the DIA format and reduces parallelism in the ReRAM arrays.

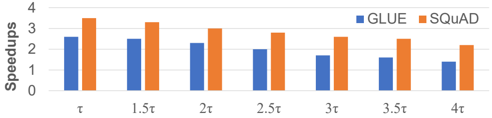
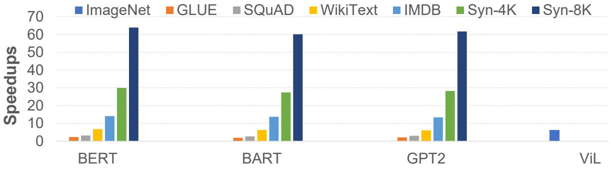


Figure 24: Impact of sparsity. Performance degradation as sparsity (τ) increases. More bubbles will increase the ratio of invalid computations, which in turn decreases ASADI's performance.

Scalability

- **Results:** ASADI achieves linearly increased speedups compared to the baseline when processing longer sequences.
- **Analysis:** ASADI's latency grows linearly while baseline's latency grows quadratically with sequence length. The *overall latency* (OL) is related to the *latency of one iteration* (LOI) and the *number of iterations* (NI), i.e., $OL = LOI \times NI$. For the PIM baseline, both the NI and LOI increase as sequence length increase, indicating quadratic increasing.



Summary

- **Observations:** 1) We observe the prevalence of diagonal locality in various sparse attention mechanisms; 2) We observe the high on-chip communication overhead in PIM solutions
- **Opportunities:** Current solutions convert diagonal locality to row/column locality. How to directly support diagonal locality
- **Contributions:** i) a new compression method to further enhance the diagonal locality; ii) we propose DIA-based sparse matrix computation paradigm and conduct quantitative comparisons with the CSR computing paradigm; iii) we present architecture and dataflow utilizing in-situ computing, which we refer to as ASADI
- **Evaluations:** The results indicate that ASADI exhibits superior performance and energy efficiency.

Thanks!

[1] Ashish Vaswani et al.
Attention is all you need.
In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[2] Thomas Wolf et al.
Transformers: State-of-the-art natural language processing.
In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[3] Iz Beltagy et al.
Longformer: The long-document transformer.
CoRR, abs/2004.05150, 2020.

[4] Zheng Qu et al.
DOTA: Detect and omit weak attentions for scalable transformer acceleration.
In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS 2022, page 1426, New York, NY, USA, 2022. Association for Computing Machinery.

[5] Liqiang Lu et al.
Sanger: A co-design framework for enabling sparse attention using reconfigurable architecture.
In *Proceedings of 54th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '21, page 977991, New York, NY, USA, 2021. Association for Computing Machinery.

[6] Amir Yazdanbakhsh et al.
Sparse attention acceleration with synergistic in-memory pruning and on-chip recomputation.
In *Proceedings of 2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 744–762, 2022.