DESIGN, AUTOMATION & TEST IN EUROPE
14 – 15 March 2022 · on-site event
16 – 23 March 2022 · online event
The European Event for Electronic System Design & Test

DTQAtten: Leveraging Dynamic Token-based Quantization for Efficient Attention Architecture

Tao Yang, Dongyue Li, Li Jiang et al.

Shanghai Jiao Tong University

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Outline

- Background
- Motivation
- Dynamic Token-based Quantization Algorithm
- Hardware Architecture for Our Quantization Algorithm
- Experimental Results
- Conclusion

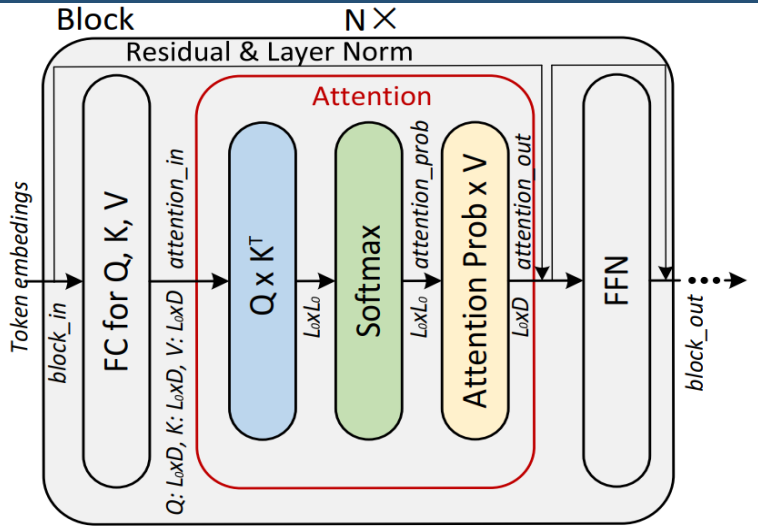
Background

Attention-based block

- BERT and GPT family.
- Attention_prob and Attention_out.

Pruning and Quantization

- For attention-based NLP models, on Q, K and V.
- Pruning: MnnFast [1], A3 [2].
- Quantization: I-BERT [3], Q-BERT [4].
- Pruning & Quantization: SpAtten [5] (token-wise pruning and layer-wise quantization) .



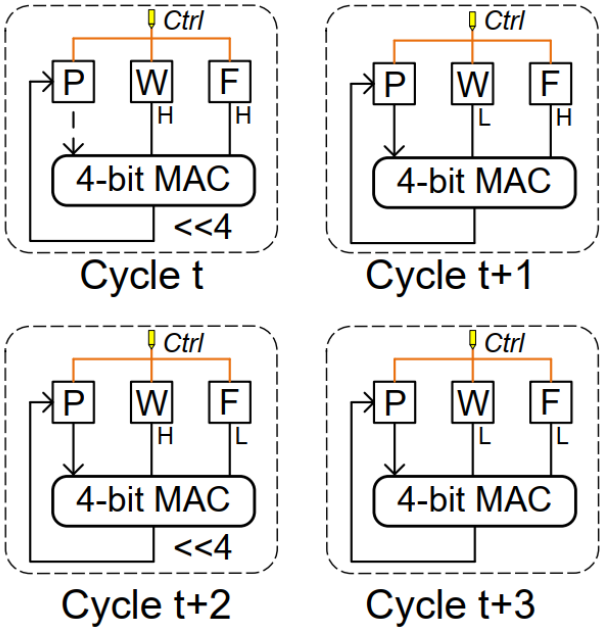
Problem

- Quantization for attention-based NLP models are still coarse-grained.
- Quantization is conducted separately from pruning for attention-based NLP models.

[1] H. Jang *et al.*, “Mnnfast: A fast and scalable system architecture for memory-augmented neural networks,” in *ISCA*, 2019.
[2] T. J. Ham et al., “A3: Accelerating attention mechanisms in neural networks with approximation,” in *HPCA*, 2020.
[3] S. Kim *et al.*, “I-bert: Integer-only bert quantization,” *ICML*, 2021.
[4] O. Zafrir *et al.*, “Q8BERT: quantized 8bit BERT,” *CoRR*, 2019.
[5] Wang *et al.*, “Spatten: Efficient sparse attention architecture with cascade token and head pruning,” in *HPCA*, 2021.

Background

Variable-speed Systolic Array (VSSA)



- **Normal Systolic array:**
  - **Advantage**: friendliness for very large scale integration (VLSI) with high speed and low cost.
  - **Disadvantage**: can only support “one” precision.
- **Variable-speed Systolic Array (VSSA)**[1]**:**
  - **Feature**: using 4-bit MACs, each can be united into an 8-bit MAC taking four cycles in a timing-multiplexing manner.
  - **Advantage**: both support 4-bit and 8-bit computing.
  - **Disadvantage**: The pipeline stall occurs when the adjacent PEs execute MACs with different precisions.

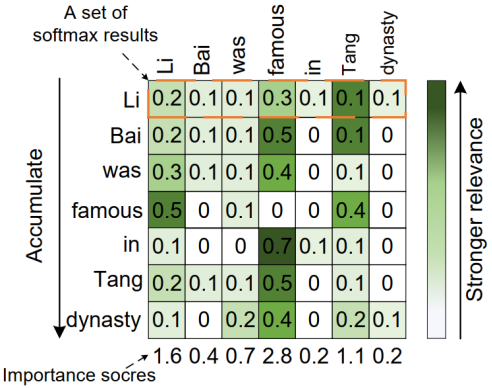
Problem: The high pipeline stall ratio incurs the low computational efficiency of VSSA.

[1] Z. Song et al., “Drq: Dynamic region-based quantization for deep neural network acceleration,” in ISCA, 2020

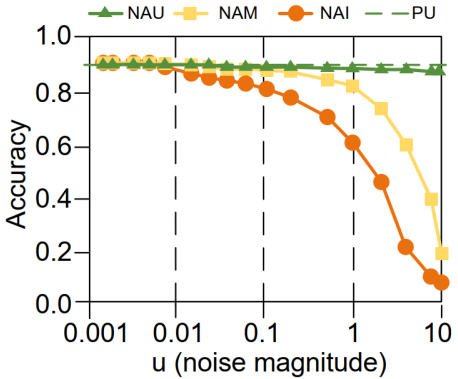
Motivation

Different tokens show Different Tolerance to Noise

- **Split into three segments （according to the importance scores calculated in Fig(a).）：**
  - Segment 0 : **15% important** tokens
  - Segment 1: **70% middle important** tokens
  - Segment 2 : **15% most unimportant** tokens
- **Experiments:**
  - **NAI**: only add noise to segments 0.
  - **NAM**: only add noise to segments 1.
  - **NAU:** only add noise to segments 2.
  - **PU:** directly prune the tokens in segment 2.
- **Finding (The results are shown in Figure(b).):**
  - tokens in different segments show varied tolerance to noise.
  - token's tolerance to noise is inversely proportional to its importance degree
  - pruning unimportant tokens has a negligible effect on the model accuracy.
- **Conclusion**: achieving a high compression ratio while maintaining accuracy requires:
  - restricting the quantization impact (equivalent to noise) for the important tokens.
  - relaxing the quantization for the unimportant tokens.



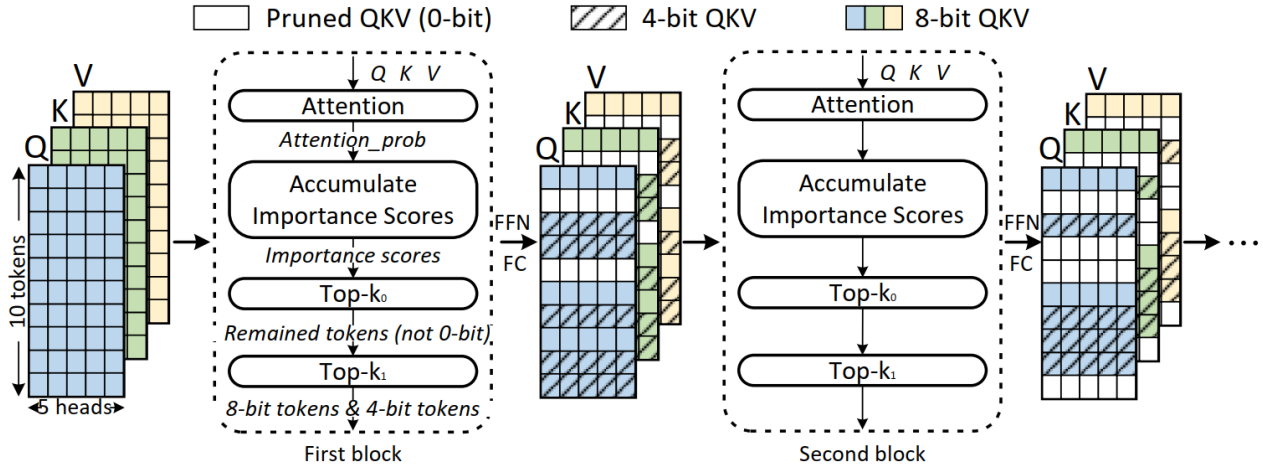
(a) Attention probabilities are summed over each column of the attention_prob to get importance scores.



(b) BERT-Base accuracy on task qnli[1] with noise (u means noise magnitude) added on tokens with different importance scores.

[1] D. W. Otter et al., "A survey of the usages of deep learning for natural language processing," TNNLS, 2021.

Dynamic Token-based Quantization Algorithm



- Use attention probabilities to calculate the importance score of each input token dynamically in each attention block.
- Two-level top-K engine to split the tokens into three segmentation according to their importance scores.
- important tokens (segment 0) → **high-precision (8-bit);**
  middle important tokens (segment 1) → **low-precision (4-bit);**
  unimportant tokens (segment 2) → **prune (0-bit);**

- **Design Space (percentage of each segmentation in each block) Explore**
  - **Method:**
    Bayesian optimization
  - **Target problem:**

$$\textbf{minimize} \quad \mathcal{L}(R) = \mathcal{L}_{en} + \lambda * \mathcal{L}_{ops}$$

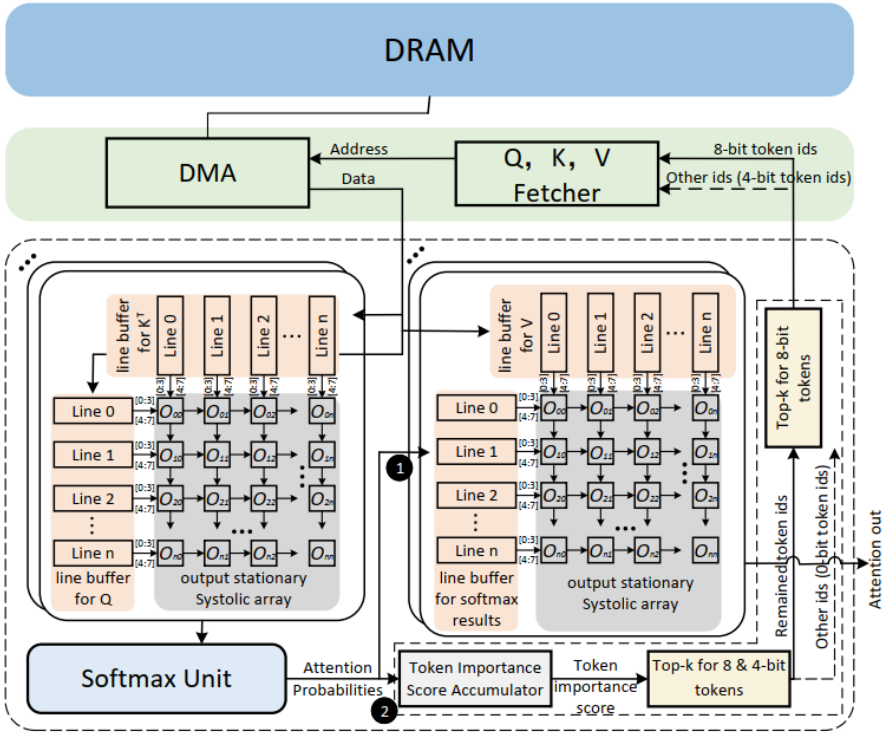
$\mathcal{L}_{en}$: model cross-entropy loss

$\mathcal{L}_{ops}$: bit-level operation number (BOPs) of the model.

Our method is a joint quantization algorithm!

Hardware Architecture

Overview



- **VSSA** → support mix-precision (4-bit and 8bit) matrix multiplication.
- Attention_prob is broadcasted to two modules:
  - Module ❶: contains **VSSAs** → produce the final results of the attention.
  - Module ❷: consists of a token importance score accumulator and two top-K engines:
    - **token importance score accumulator** → accumulates the attention probabilities to achieve the importance score of each token.
    - **Two top-k engine** → split the input tokens into 8-bit, 4-bit and 0-bit segments according to the importance scores
  - The executions of Module ❶ and module ❷ are in parallel, thus, the execution time of module ❷ can be hidden because of the large calculation quantity of the matrix multiplication in module ❶.
- **Q, K, V fetcher** → calculate the physical start address and the length of the Q, K, V vectors using the token ids produced by the two top-k engines.
- **DMA** → prefetch the Q, K, V vectors for the next layer.

Hardware Architecture

PE Efficiency Optimization Strategy

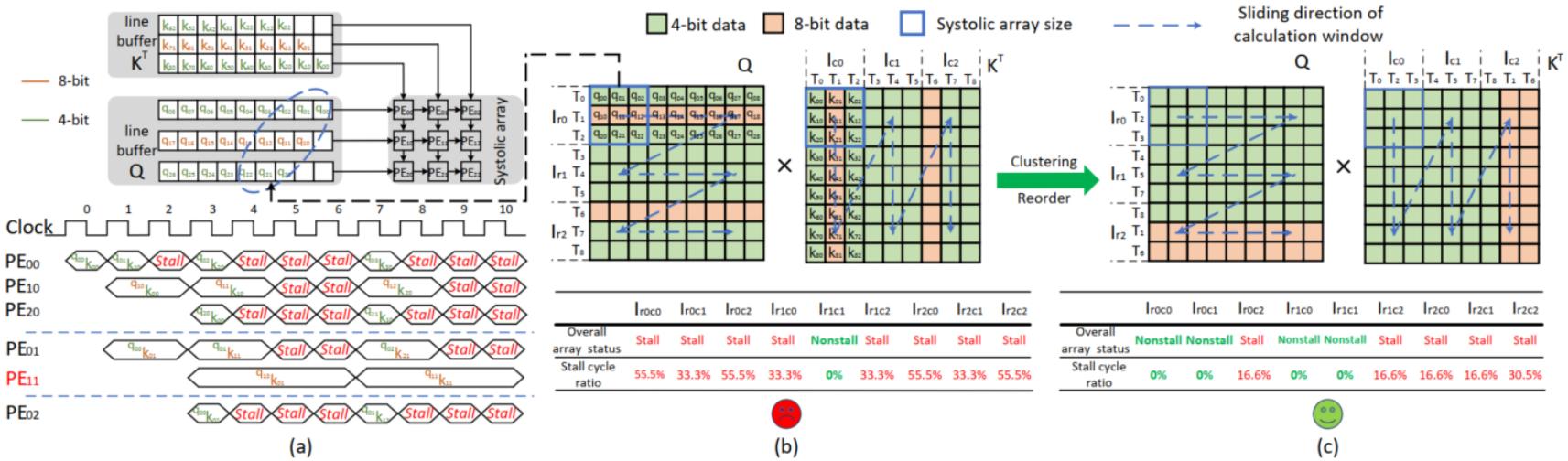


Fig 1. (a) The dataflow and the executing timing of VSSA in iteration $Ir0c0$. (b) The executing window sliding in $Q \times K^T$ and the stalling cases when the input tokens are in normal order. (c) The executing window sliding in $Q \times K^T$ and the stalling cases after clustering and reordering the tokens.

- The whole process is split into 9 iterations as shown in Fig1.(b).
- Stall problem: The calculation of each PE should be synchronized with others for the strict dataflow requirement in systolic array
  - If the accuracy of the two matrices involved in the calculation is not of single-precision (e.g. the first iteration $Ir0c0$ of the calculation in Fig1.(b), of which the dataflow is shown in Fig1.(a)), the PE responsible for low-precision calculation~(PE00,PE10,PE20,PE01,PE02 in Fig1.(a)) needs to stall to wait for the PE of high-precision calculation~(PE11 in Fig1.(a), which is the "critical path", determine the bottle neck of the calculation throughput).

Hardware Architecture

PE Efficiency Optimization Strategy

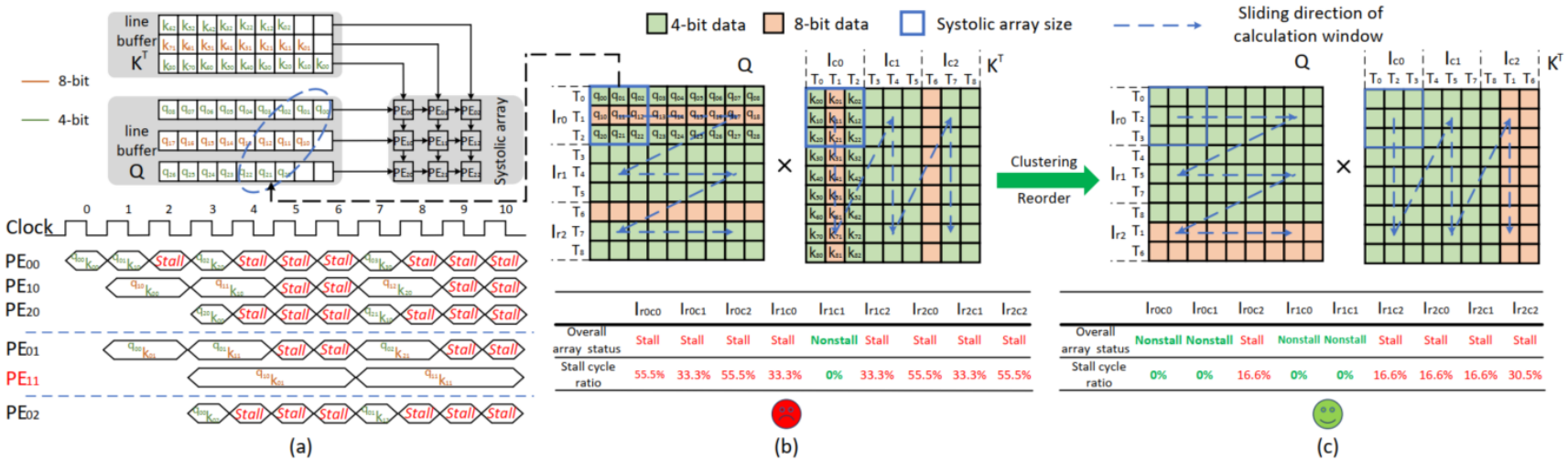


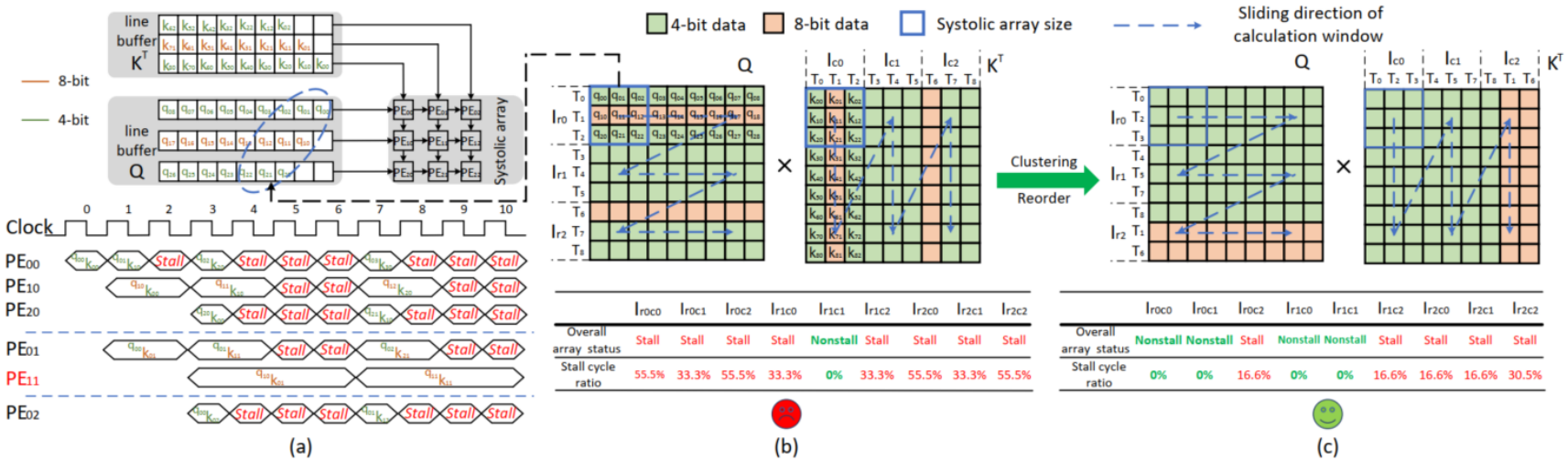
Fig 1. (a) The dataflow and the executing timing of VSSA in iteration I_{r0c0}. (b) The executing window sliding in $Q \times K^T$ and the stalling cases when the input tokens are in normal order. (c) The executing window sliding in $Q \times K^T$ and the stalling cases after clustering and reordering the tokens.

- Severity of the stall problem:
  - 8 of the 9 iterations in Fig1.(b) meet the stall problem
  - the average 39.5% of the execution cycles meet the stall problem.

Very low efficiency of the computing resources !!!

Hardware Architecture

PE Efficiency Optimization Strategy



	I_{r0c0}	I_{r0c1}	I_{r0c2}	I_{r1c0}	I_{r1c1}	I_{r1c2}	I_{r2c0}	I_{r2c1}	I_{r2c2}
Overall array status	Stall	Stall	Stall	Stall	Nonstall	Stall	Stall	Stall	Stall
Stall cycle ratio	55.5%	33.3%	55.5%	33.3%	0%	33.3%	55.5%	33.3%	55.5%

	I_{r0c0}	I_{r0c1}	I_{r0c2}	I_{r1c0}	I_{r1c1}	I_{r1c2}	I_{r2c0}	I_{r2c1}	I_{r2c2}
Overall array status	Nonstall	Nonstall	Stall	Nonstall	Nonstall	Stall	Stall	Stall	Stall
Stall cycle ratio	0%	0%	16.6%	0%	0%	16.6%	16.6%	16.6%	30.5%

Fig 1. (a) The dataflow and the executing timing of VSSA in iteration I_{r0c0}. (b) The executing window sliding in $Q \times K^T$ and the stalling cases when the input tokens are in normal order. (c) The executing window sliding in $Q \times K^T$ and the stalling cases after clustering and reordering the tokens.

- PE Efficiency Optimization Strategy:
  - Method: reorder the rows/columns of Q/K matrix to cluster the tokens of the same precision together as shown in Fig 1.(c).
  - Effect:
    - reduce the stall iteration number from 8 to 5.
    - Reduce the ratio of the average stall cycles in the stall iteration from 39.5% to 19.38%.

The efficiency of the computing resources is improved !!!

Hardware Architecture

PE Efficiency Optimization Strategy

- The effect of the reorder the input tokens to the Attention output:

$$\begin{aligned} Attention\_out^{*} &= Softmax[(A \times Q) \times (K^{T} \times A^{T})] \times (A \times V) \\ &= A \times Softmax[Q \times K^{T} \times (A^{T} \times A)] \times V) \\ &= A \times [Softmax(Q \times K^{T}) \times V] \\ &= A \times Attention\_out \end{aligned}$$

- Reordering the input tokens results in a reordered *Attention output* without impacting the output values.

- The effect of the reorder the input tokens to the Importance scores and token ids:

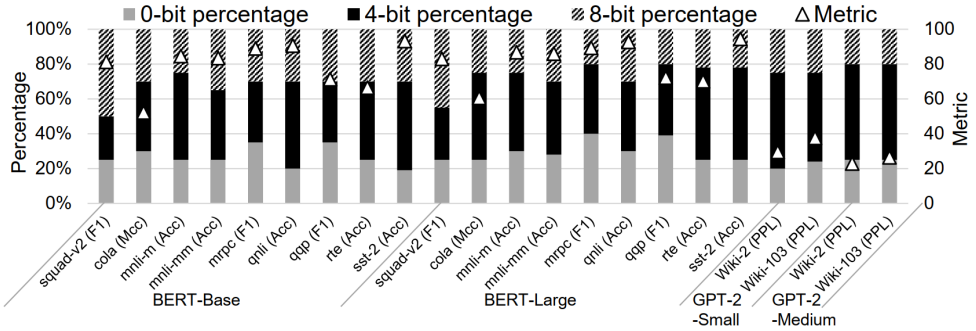
$$\begin{aligned} Attention\_prob^{*} &= Softmax[(A \times Q) \times (K^{T} \times A^{T})] \\ &= A \times Softmax(Q \times K^{T}) \times A^{T} \\ &= A \times Attention\_prob \times A^{T} \end{aligned}$$

- Token importance scores are achieved by accumulating the *Attention prob* in each column, the left *A* has no impact on token importance scores.
- The right A^{T} makes the order of importance scores matches that of the input tokens to the next attention-based block.
- Correspondingly, the token ids obtained by sorting the importance scores are also based on the order caused by *A*.

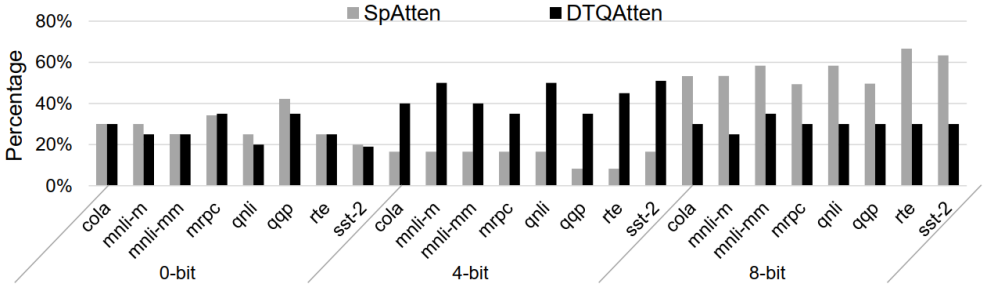
Thus, we can sequentially store the block results of ordered input tokens on DRAM and then use the token ids directly in the next block to fetch tokens with no extra hardware overhead.

Experimental Results

Quantization Algorithm Performance



Comparisons of the model performance and percentages of 8/4/0-bit tokens on different datasets.



Comparison of the percentages of 8/4/0-bit tokens with the state-of-the-art 2-step compression method (SpAtten[1])

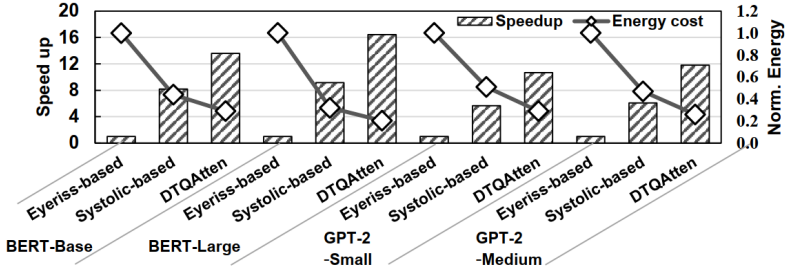
- Compressed BERT-Base and compressed BERT-Large show negligible accuracy degradation (within 1%) on the nine datasets compared with that of the original models, except 1.7% for squad-v2. The average percentages of the 0-bit, 4-bit and 8-bit parts in BERT are 28%, 41.4% and 30.6%, respectively.
- GPT-2 exhibits no performance loss on each dataset except a negligible 0.12% PPL enhancement for GPT-2-small on Wiki-103 (37.62% vs 37.5%). The average percentages of the 0-bit, 4-bit and 8-bit parts in these two GPT networks account for 21.8%, 52.0% and 26.2%, respectively.

- the average 0-bit ratio in SpAtten (29.8%) is a little bit higher than that in DTQAtten (26.3%).
- the average ratio of the 4-bit part in SpAtten (18.9%) is much lower than that in DTQAtten (40.5%).
- Thus, DTQAtten has a totally higher compression ratio.

[1] Wang *et al.*, "Spatten: Efficient sparse attention architecture with cascade token and head pruning," in *HPCA*, 2021.

Experimental Results

Computational Performance



Performance comparisons with each NN-based accelerator.

	MNNFast	A^3	SpAtten	**DTQAtten**
Technology	FPGA (28nm)	ASIC(40nm)	ASIC (40nm)	**ASIC (40nm)**
Frequency	1GHz (projected)	1GHz	1GHz	**1GHz**
Area (mm^2)	-	$2.08mm^2$	$1.55mm^2$	**$1.41mm^2$**
Throughput (GOP/s)	120 (1×)	221 (1.8×)	360 (3.0×)	**952.8 (7.94×)**
Energy Effi. (GOP/j)	120 (1×)	269 (2.2×)	382 (3.2×)	**1298.4 (10.82×)**
Area Effi. (GOP/s/mm2)		106 (1×)	238 (2.2×)	**678.4 (6.4×)**

Comparisons among four Attention accelerators.

- **Compared with Eyeriss-based design, DTQAtten achieves 13.57×,16.42×, 10.67× and 11.8× speedup and reduce 71%, 79%, 71%, 74% energy cost for each model.**

- **Among the four attention accelerators, DTQAtten achieves the highest throughput (7.94× compared with MNNFast) and the smallest area cost (67% of the area cost in A3) during the four designs.**

Conclusion

- We find that different tokens in attention-based NLP models show different tolerance to noise.
- We propose to dynamically quantize tokens with different precision (0-bit, 4-bit and 8-bit) according to their importance level to reduce the computation complexity without losing accuracy.
- We propose a hardware architecture with an optimization strategy to exploit our quantization algorithm efficiently.

Thanks!