HARDSEA: Hybrid Analog-ReRAM Clustering and Digital-SRAM In-Memory Computing Accelerator for Dynamic Sparse Self-Attention in Transformer

Shiwei Liu, *Graduate Student Member, IEEE*, Chen Mu, *Student Member, IEEE*, Hao Jiang, Yunzhengmao Wang, Jinshan Zhang, Feng Lin, Keji Zhou, *Member, IEEE*, Qi Liu, *Member, IEEE*, and Chixiao Chen, *Member, IEEE*

***Abstract*— Self-attention-based transformers have outperformed recurrent and convolutional neural networks (RNN/CNNs) in many applications. Despite the effectiveness, calculating self-attention is prohibitively costly due to quadratic computation and memory requirements. To solve this challenge, this article proposes a hybrid analog-ReRAM and digital-SRAM in-memory computing accelerator (HARDSEA), a computing-in-memory (CIM) accelerator supporting self-attention in transformer applications. To trade off between energy efficiency and algorithm accuracy, HARDSEA features an algorithm-architecture-circuit codesign. A product-quantization-based scheme dynamically facilitates self-attention sparsity by predicting lightweight token relevance. A hybrid in-memory computing architecture employs both high-efficiency analog ReRAM-CIM and high-precision digital SRAM-CIM to implement the proposed new scheme. The ReRAM-CIM, whose precision is sensitive to circuit nonidealities, takes charge of token relevance prediction where only computing monotonicity is demanded. The SRAM-CIM, utilized for exact sparse attention computing, is reorganized as an on-memory-boundary computing scheme, thus adapting to irregular sparsity patterns. In addition, we propose a time-domain winner-take-all (WTA) circuit to replace the expensive ADCs in ReRAM-CIM macros. Experimental results show that HARDSEA prunes BERT and GPT-2 models to 12%–33% sparsity without accuracy loss, achieving 13.5×–28.5× speedup and 291.6×–1894.3× energy efficiency over GPU. Compared to state-of-the-art transformer accelerators, HARDSEA has 1.2×–14.9× better energy efficiency at the same level of throughput.**

***Index Terms*— Algorithm-architecture-circuit co-design, in-memory computing, self-attention, transformer.**

Manuscript received 18 May 2023; revised 14 September 2023; accepted 16 November 2023. Date of publication 20 December 2023; date of current version 23 January 2024. This work was supported in part by the Major Project of the Science and Technology Innovation 2030 under Grant 2021ZD0114400, in part by the Strategic Priority Research Program of Chinese Academy of Sciences (CAS) under Grant XDB44000000, in part by the National Natural Science Foundation of China (NSFC) under Grant 62322404 and Grant 61732020, in part by the Shanghai Rising-Star Program under Grant 10QA1407300, and in part by the Zhangjiang Fudan International Innovation Center. *(Corresponding author: Chixiao Chen.)*

Shiwei Liu, Chen Mu, Hao Jiang, Yunzhengmao Wang, Jinshan Zhang, and Feng Lin are with the State Key Laboratory of Integrated Chips and Systems, Fudan University, Shanghai 200433, China.

Keji Zhou, Qi Liu, and Chixiao Chen are with the State Key Laboratory of Integrated Chips and Systems, Fudan University, Shanghai 200433, China, and also with Zhangjiang Laboratory, Shanghai 201210, China (e-mail: cxchen@fudan.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TVLSI.2023.3337777.

Digital Object Identifier 10.1109/TVLSI.2023.3337777

I. Introduction

VANILLA transformer [1] and its variants [2], [3], [4] are widely used, from natural language processing [7], [8] to computer vision tasks [9], [10]. The critical significant advance of the transformer is *self-attention*, drawing information across input tokens regardless of their distance. In contrast with traditional recurrent or convolutional neural networks (RNN/CNNs), the emerging algorithm achieved more promising results. However, the demand for computation and memory access due to self-attention grows quadratically with the increase of the input token number, making it difficult to adapt to conventional computing hardware.

Computing-in-memory (CIM)-based architectures are regarded as an optimal solution for computing/memory-intensive algorithms. They feature array-level computing parallelism and ultratight coupling between computing and storage elements, thus reducing the power consumption of data movement. State-of-the-art CIM accelerators adopt nonvolatile (e.g., ReRAM) or SRAM implementation. ReRAM-CIM, generally organized as a crossbar [11], [12], performs analog matrix–vector multiplication and quantizes the final sum through ADCs. Programmable ReRAM devices can store multibit weights as multilevel conductances, thus achieving high density and low power consumption. However, analog computing using tiny-size ReRAM devices is sensitive to process variation and parasitic effects, only achieving up-to-5-bit precision [20], [21] in one cell. In contrast, digital SRAM-CIM designs [17], [18], [19] were proposed to tradeoff between computing precision and energy efficiency. The SRAM cells, paired with XNOR/AND gates, perform multiplication in a bit-serial or bit-parallel scheme. Partial sums are accumulated through a digital adder tree without any loss. Though less efficient than analog ReRAM-CIM, digital SRAM-CIM still consumes less power than traditional digital circuits through diverse custom designs.

Most CIM-based designs exploit data reusing [22], quantization [23], and pruning techniques [24], [25] for computing-intensive CNN or RNN acceleration, but they cannot be directly applied to memory-intensive self-attention. A few trailblazers [42], [43], [45] aim at self-attention acceleration with operation schedule optimization. These works customize pipeline designs for computations across different layers

simultaneously, reducing memory storage and data access for intermediate results. To further perform the self-attention efficiently, sparsity is exploited to reduce power consumption and enhance the computing throughput. Unlike the weight sparsity in CNNs or RNNs, the self-attention sparsity stems from token relevance. In other words, self-attention mainly allocates information from strongly related tokens. Static sparsity patterns are first investigated, limiting the self-attention field to predefined and structured patterns, such as local windows [27], [30], [55] or blocks [31]. In these cases, tokens can only draw information from their neighbors, considering that strong relevance often appears with nearby contexts. This static sparsity distribution is compatible with the CIM-based parallel processing element.

In practice, dynamic sparsity patterns, varying according to different input tokens, outperforms their static counterparts. According to token relevance, dynamic sparsity assigns input tokens to different and irregular buckets in a data-driven fashion. The self-attention is only performed on the tokens belonging to the same bucket. Various search strategies, such as clustering [28], hashing [29], [35], sorting [36], [37], low rank [38], [39], [46], and computing decomposition [40], [41], [44], are used to predict the token relevance before exactly computing self-attention. Compared to static sparsity, dynamic sparsity prunes the weakly related tokens in a more optimal fine-grained pattern, resulting in a higher sparsity at the same level of accuracy.

There is a noteworthy observation that the highly parallel computation for token relevance prediction is insensitive to computing precision as long as distinguishes the strongly related tokens. Moreover, the precise self-attention outputs only require a small portion of accurate calculation if zero-skipping schemes can be developed. Inspired by the above insights, this article proposes a hybrid analog-ReRAM and digital-SRAM in-memory computing accelerator, denoted as HARDSEA, for sparse self-attention implementation. Different from previous attention-oriented CIM designs, HARDSEA supports unstructured and dynamic sparsity in regular CIM arrays. In short, the key contributions are listed as follows.

1) We propose a product-quantization-based sparse self-attention algorithm, where the overall accuracy is insensitive to the approximate token relevance prediction.
2) We use heterogeneous CIM implementation for different computing tasks, where efficient but approximate analog ReRAM-CIM is adopted for the front-end attention sparsification, and loss-less digital SRAM-CIM is applied for the back-end sparse attention computing.
3) We utilize an analog-to-digital-converter (ADC)-free ReRAM-CIM macro, where a custom time-domain winner-take-all (WTA) circuit is proposed to perform multibit clustering in the product-quantized sparse self-attention efficiently.
4) We extensively evaluate HARDSEA on BERT and GPT-2 models with 11 benchmarks from GLUE [5], Enwik-8, and Text-8 datasets [6]. Experiment results show that HARDSEA can prune BERT and GPT-2 models to 12%–33% sparsity without accuracy loss while achieving 13.5×–28.5× speedup and 291.6×–1894.3× energy reduction over GPU. Compared to state-of-the-art transformer accelerators, HARDSEA yields 1.2×–14.9× better energy efficiency at the same level of throughput.

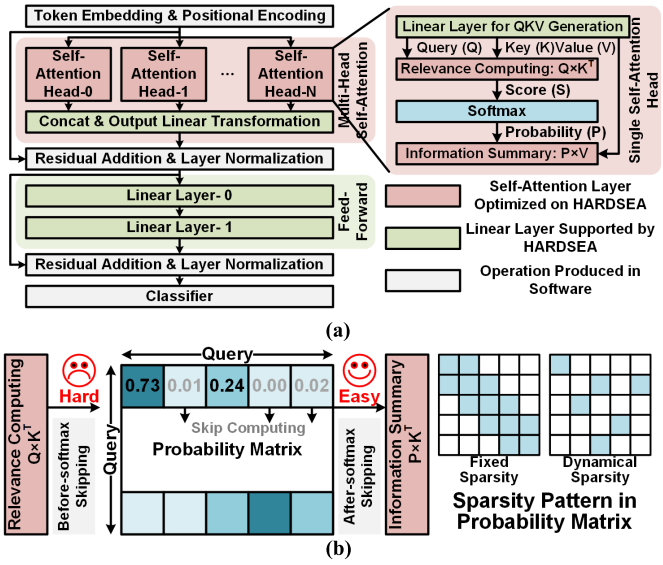

Fig. 1. (a) Transformer structure and operation supported on HARDSEA and (b) self-attention sparsity.

The rest of the article is organized as follows. Section II reviews the self-attention mechanism and the existing attention-oriented accelerators. Section III describes the proposed attention sparsification algorithm. Section IV presents the overall architecture and circuit implementation of the HARDSEA accelerator. Experimental results are demonstrated in Section V, and Section VI concludes this article.

II. Background and Motivation

A. Self-Attention in Transformers

Fig. 1(a) illustrates the multihead transformer structure, where a single self-attention head is detailed in the dashed box. The self-attention receives three input features, namely query (**Q**), key (**K**), and value (**V**), and generates the attention output (**A**). These three features are generated from separate linear transformations on the same token sequence. Each feature is a dense matrix with size $\mathbb{R}^N \times \mathbb{R}^D$, where $\boldsymbol{N}$ is the token number, and $\boldsymbol{D}$ is the feature size. The first step of self-attention is token relevance computing, which multiplies Q and K to obtain the score matrix (**S**). The inner product between the ith query (Q_i) and the jth key (K_j) represents the relevance between the ith token and the jth token. Therefore, the scores in each row indicate the relevance of one specific token to the others. Second, the row-wise softmax normalizes the attention scores to the probabilities (**P**). Finally, P is multiplied by V for the information summary. Note that both relevance computing and information summary require $N^2 \times D$ multiply-and-accumulate (MAC) operations. Therefore, the computation and memory cost is quadratic to the input token number.

The quadratic complexity precludes the transformer application to long input sequences. Fortunately, the softmax normalization converts most of the attention scores to zero or near-zero probabilities, which causes negligible contribution to the final result. Disabling the computation of the corresponding weakly related tokens can efficiently reduce self-attention complexity. The zero-involved computing for the after-softmax information summary can be easily skipped after obtaining the attention scores. Nevertheless, zero skipping for the before-softmax relevance computing is difficult and requires a token relevance prediction scheme. Therefore, the proposed HARDSEA accelerator mainly exploits the sparsity in relevance computing, softmax, and information summary. In addition, this work can also support linear layers for QKV generation and feed-forward layers in the digital SRAM-CIM. The preprocesses like input embedding and positional encoding and postprocesses such as classifier, residual addition, and layer normalization are excluded from our work.

B. Existing Attention-Oriented Accelerators

The rapid development of transformers has led to extensive research on dedicated accelerators [27], [32], [38], [39], [42], [43], [44], [45], [46], [47], [48], [49], [50], [55]. According to the mechanism of token relevance prediction, these accelerators can be divided into two categories, fixed sparsity and dynamic sparsity [26], as shown in Fig. 1(b). The fixed sparsity limits the attention field to predefined and structured patterns, where the token can only draw information from the neighbors. ReBERT [30] and TranCIM [31] prune the attention score matrix by fixed window-wise or block-wise sparsity patterns. Thanks to the structured attention field, sparse attention computing can be easily reorganized into regular ReRAM-CIM or SRAM-CIM macros. However, the fixed sparsity ignores the inherent token relations and generally results in a lower sparsity to maintain accuracy. In contrast, the dynamic sparsity customizes an optimal fine-grained attention sparsity pattern based on token relevance. For example, before computing attention exactly, ELSA [35] predicts token relevance by local-sensitive-hashing, while A3 [36] uses greedy searching. Sanger [40] omits weakly related tokens by low-precision attention detection and high-precision attention recomputing. However, these works exacerbate the hardware overhead for relevance prediction.

In summary, the existing transformer accelerators are challenged by two issues. First, conventional digital accelerators can support dynamic sparsity in self-attention. However, the dynamic token relevance prediction and unstructured data access alleviate the performance gained from attention sparsity. Second, the CIM-based accelerators achieve good energy performance but cannot support irregularly distributed and varying sparsity patterns.

C. Key Ideas of HARDSEA

To adopt dynamical self-attention sparsity in CIM-based architecture, this work proposes HARDSEA via an algorithm-architecture-circuit codesign.

1) Lightweight Token Relevance Prediction: HARDSEA proposes a dynamical product-quantization-based token relevance prediction algorithm. The idea is to decompose the query and key into a series of low-dimensional subspaces and then to quantize each subspace separately. The token relevance can be efficiently estimated from their quantization indices. In addition, our experimental results prove that the product-quantized self-attention sparsification is insensitive to computing precision while distinguishing strongly related tokens. This inherent feature facilitates the hardware implementation of the proposed algorithm.

2) Heterogeneous CIM Architecture: HARDSEA uses a heterogeneous CIM architecture for different computing workloads. The efficient but relatively approximate analog ReRAM-CIM is adopted for front-end token relevance prediction. It materializes the k-means clustering to quantize decomposed queries and keys to their nearest centroids. Although the output from the ReRAM crossbar might be nonlinearly distorted, the monotonicity can be guaranteed for reliable clustering operation. To mitigate the adverse repercussions to model accuracy, the loss-less SRAM-CIM recomputes self-attention for the filtered tokens. It is further reorganized as an on-memory-boundary computing scheme to flexibly adapt the irregular sparsity pattern.

3) ADC-Free ReRAM-CIM Circuit Design: To further reduce the overhead for token relevance prediction, HARDSEA replaces power-hungry ADCs in conventional ReRAM-CIM macros with a time-domain WTA circuit. The proposed WTA amplifies the time-delay difference generated from the output of the ReRAM crossbar, and then searches a winner path via a binary-tree topology.

III. Product-Quantization-Based Sparse Self-Attention Algorithm

Exploring attention sparsity is the critical knob to reduce the total computation and memory cost. Self-attention produces the inner product between the query and key to identify the token relevance. The proposed algorithm aims to predict a lightweight relation notion as solving the maximum inner product search (MIPS) problem. Inspired by applying product quantization [51], [52] to the MIPS problem, we propose a product-quantized self-attention sparsification method.

A. Product-Quantized Self-Attention Sparsification Flow

Fig. 2 depicts the processing flow of the proposed algorithm. For easy comprehension, the key and query are represented by waveform symbols.

1) Clustering and Representation: First, the query or key is split into different subvectors for $\boldsymbol{M}$ subspaces. Next, we produce unique k-means clustering in each subspace and assign the subvector to the closest centroid based on the Euclidean distance. The number of centroids is determined by a hyperparameter $\boldsymbol{C}$. In addition, like [28], the centroid is updated by the exponential moving average of all subvectors assigned to it during training as

$$\text{Centroid} = \lambda \cdot \text{Centroid} + (1-\lambda)\sum \text{Sub\_vectors} \quad (1)$$

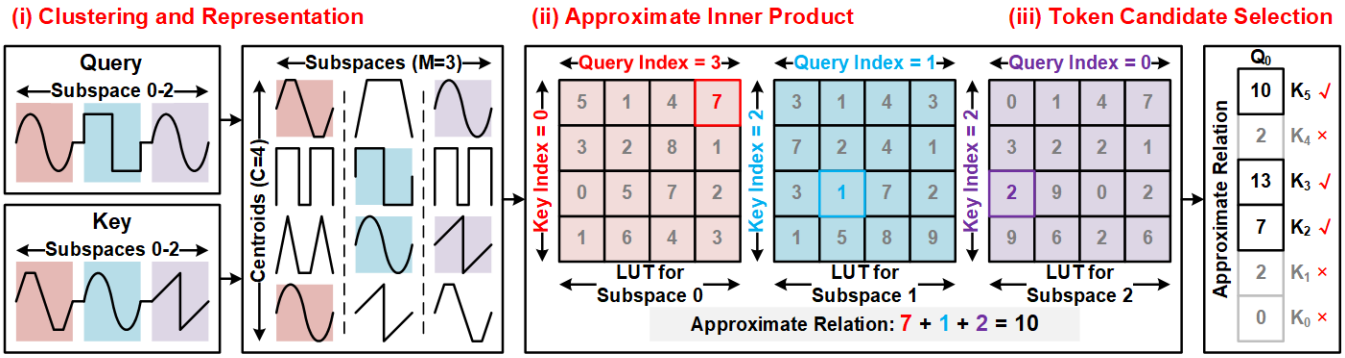

Fig. 2. Product-quantization-based self-attention sparsification flow.

TABLE I
COMPLEXITY COMPARISON FOR TOKEN RELEVANCE MEASUREMENT

	Parameter Storage	Computation	Complexity
KNN [28]	$K \cdot D$	$N \cdot K \cdot D$	$O(K)$
Hashing [29]	$H \cdot D$	$N \cdot H \cdot D$	$O(K/2)$
Our Work	$C \cdot M \cdot D_s = C \cdot D$	$N \cdot C \cdot D$	$O(K^{1/M})$

[1] N is the token number. D is the feature size. D_s is the length of sub-vectors. $D_s = D/M$.

where λ is the decay parameter. Finally, the assigned centroids reconstruct the query or key for the approximate inner product. For brevity, three subspaces with four centroids are adopted in Fig. 2.

2) Approximate Inner Product: Causing the centroids are already known, the partial sums between different centroids can be obtained in advance and stored in separate look-up tables (LUTs). The clustering index is used to access the corresponding partial sum. Then, the product quantization aggregates the partial sums from different subspaces as the approximate inner product between the reconstructed query and key. The approximate result predicts the token relevance.

3) Token Candidate Selection: Finally, only the tokens with the *Top-K* approximate relevance are selected as the candidates for exact self-attention. This method can facilitate computing reduction compared to dense attention.

Table I summarizes the parameter storage and computation complexity of the proposed self-attention sparsification method and compares it with KNN in Routing Transformer [28] and hashing in Reformer [29]. To precisely reconstruct the query or key with K representations, the total number of the representations should be sufficiently large. KNN clusters query or key to K centroids while hashing converts them into H hash tables satisfying $H = K/2$. In our work, the product quantization represents the query or key with C centroids in M subspaces satisfying $K = C^M$. Therefore, our approach achieves the least memory and computation cost, where the complexity is reduced from $O(K)$ in KNN to $O(K^{1/M})$.

B. Product Quantization Hyperparameter Searching

There are three hyperparameters, subspaces number (M), centroid number (C), and token candidate number (*Top-K*), in the mentioned sparsification flow. We adopt a greedy-search-based algorithm to determine these three parameters in Algorithm 1. It takes the base model, target accuracy, and memory usage for centroid storage as inputs. It then outputs the subspace and centroid number for the entire model, and the token candidate number for each attention head.

Algorithm 1 PQ Parameter Search

Input: Memory usage, Base model, Target accuracy
Output: Subspace number (M), Centroid number (C), Token section number for each attention head (*Top-*K_i)

```
// Step 1: Global Top-K search for the entire model.
Top-K ← Token Number N
while Top-K <1 do
    ACC ← Inference(Base model, Top-K, M=1, C=1)
    if ACC >Target Accuracy then
        Top-K ← Top-K - Step
    else
        break
    end
end
// Step 2: Subspace and Centroid Number Search.
C ← Memory Usage / Feature Size (D), M ← 1
while do
    ACC ← Inference(Base Model, Top-K, M, C)
    if ACC <Target Accuracy then
        M ← M + 1
    else
        break
    end
end
// Step 3: Fine-grained Top-K_i Search and Fine-tuning.
foreach i in Attention_Head do
    Top-K_i ← Top-K
    while do
        ACC ← Inference(Base Model, Top-K_i, M, C)
        if ACC >Target Accuracy then
            Top-K_i ← Top-K_i - Step
        else
            break
        end
    end
end
Model ← Finetune(Base Model, Top-K_i, M, C)
```

First, a global *Top-K* (Line 1–10) search aims to find a coarse-grained token candidate number for the entire model. It initializes the *Top-K* to the token number and then performs the inference with the product-quantized sparse attention. If the accuracy exceeds the target, the model still has sparsity potential. Otherwise, the minimum token candidate number is

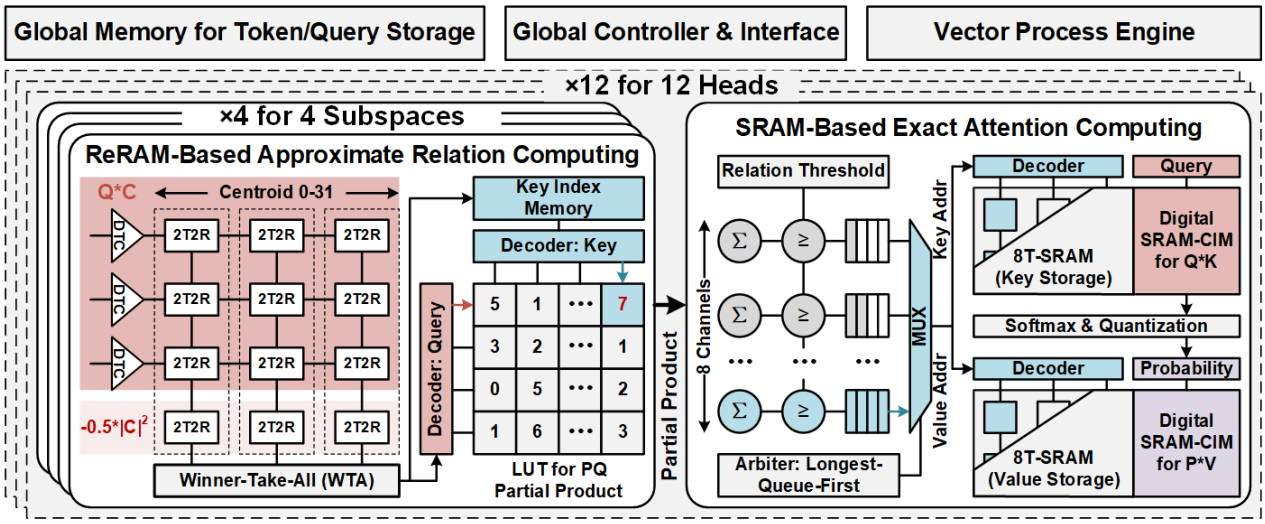

Fig. 3. HARDSEA accelerator architecture.

obtained and assigned to all self-attention heads in the base model.

Next, the number of subspaces and centroids is searched (Lines 11–20). The centroid number is determined by the memory usage for centroid storage and feature size, as Table I shows. After that, the subspace number increases until the accuracy meets the target. It indicates that the query and key require more representation permutations as $K = C^M$ scales up.

Finally, the fine-grained greedy search (Lines 21–33) further reduces attention complexity. The search procedure for each attention head is identical to the one in the first step. After obtaining all parameters, the sparse model is fine-tuned to recover the performance.

IV. HARDSEA Implementation

Although requiring matrix multiplication, the clustering in product quantization can inherently tolerate approximate computing as long as the closest centroid can be distinguished. As mentioned in Section III, its output is the assigned centroid instead of the exact inner product. Based on the above observation, the proposed HARDSEA accelerator exploits different CIM characteristics to improve the overall system efficiency.

A. Overall Architecture and Pipeline Design

As illustrated in Fig. 3, HARDSEA contains 12 cores to process up to 12 self-attention heads in parallel. Each attention core consists of four front-end ReRAM-CIMs for token relevance prediction and a back-end SRAM-CIM for precise attention computing. The Softmax function is produced in the vector process engine. The global memory stores input tokens and intermediate queries. The interface links ON-chip memory and OFF-chip DRAM.

We use a single self-attention head in an encoder-only BERT model [2] to explain the HARDSEA processing flow. Running a decoder-only GPT model [3], [4] can work as a special case with strict data dependency. HARDSEA receives the query, key, and value matrices as inputs, generating attention features for subsequent linear transformation across four steps. First, the ReRAM-CIM crossbar clusters all keys to different subspaces and stores the clustering indices in the key index memory. Second, HARDSEA executes the sparse attention for each query sequentially. The ReRAM-CIM is reused for query clustering. The clustering index serves as an address to access partial sums from LUTs. Third, the SRAM-CIM accumulates all partial products as the approximate inner product between the query and key. The strongly related tokens are selected according to this approximate result. For easy implementation, the *Top-K* operation is replaced by threshold comparison. Finally, precise self-attention starts after selecting token candidates. The exact relation computing of query–key and information summary of value–probability are finished in two cascaded 8T-SRAM CIM macros.

Note that the query only produces an inner product with keys from strongly related tokens. The SRAM-CIM remains stalled unless the predicted token relevance exceeds the threshold. To improve the throughput of the overall system, the SRAM-CIM measures the token relevance between one query and eight keys at a time. The selected keys are fetched from the queues with a simple longest-queue-first scheduling arbiter.

For the linear layer, the cascaded macros in SRAM-CIM are set to complete feature-weight multiplication in parallel and generate query, key, and value matrices sequentially. Given that the global memory can store all of the input tokens ON chip, the SRAM-CIM adopts a weight-stationary workflow to reduce OFF-chip weight accesses. In the meantime, the ReRAM-CIM is gated for power saving.

B. ADC-Less Analog ReRAM-CIM

As described in Section III, the product quantization uses Euclidean distance to cluster the query and key. Rewriting Euclidean distance as

$$\|Q - C_i\|^2 = Q^2 - 2\left(QC_i - \frac{1}{2}C_i^2\right). \tag{2}$$

The query (or key) is constant to all centroids so that the nearest distance has the largest $QC_i - (1/2)C_i^2$. Note that the

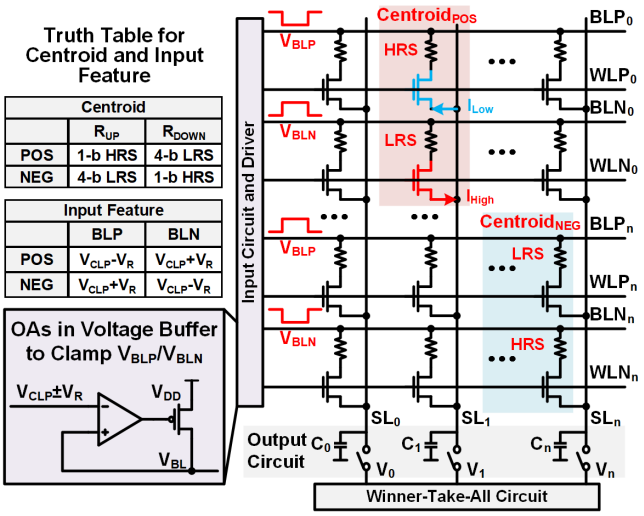

Fig. 4. 2T2R ReRAM-CIM structure and truth table for centroid and input feature representation.

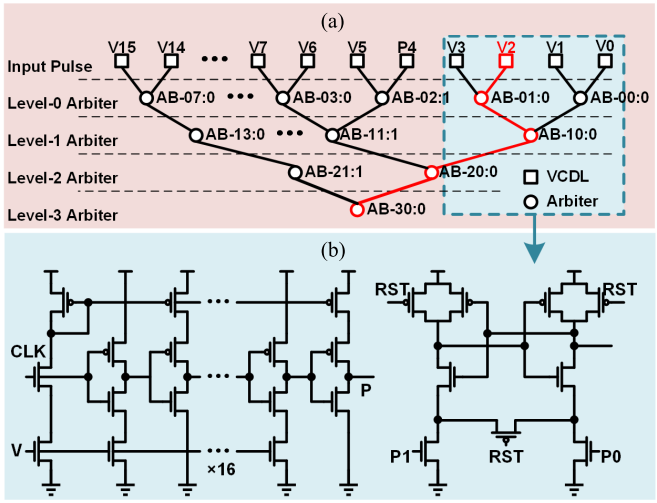

Fig. 5. (a) WTA structure and (b) VCDL and arbiter circuit.

Algorithm 2 Clustering Index Generation

Input: 15-bit Arbiter Result (AR)
Output: 4-bit Clustering Index (CLI)
CLI = 0 ▷ Initialize Clustering Index
AR_Tmp = 0 ▷ The Arbiter Output in Level-i
// Encoding Arbiter Results from Level-3 to Level-0
foreach i in 3, 2, 1, 0 **do**
  AR_Tmp = AR[15-$2^{(3-i)}$-:($2^{(3-i)}$-1)]
  if i == 3 **then**
    CLI[i]=AR_Tmp ▷ For Level-3
  else
    CLI[i]=AR_Tmp[C[3:(i+1)]] ▷ For Level-2 to Level-0
  end
end

centroids are learned offline. The module of each centroid C_i^2 is already known. Therefore, the Euclidean distance can be implemented in a crossbar for QC_i, as shown in Fig. 4.

Recent works [20], [21], [44], [56], [59] have demonstrated that a single 16/32-level ReRAM cell delivers up to 4/5-bit computing precision after including all error sources. Moreover, our experiments show that a 5-bit precision is sufficient for product-quantized clustering, yielding equivalent model accuracy. Therefore, to further increase the error tolerance, we use a more conservative 2T2R ReRAM structure to express the 5-bit centroid. For positive centroids, the top cell sets to a constant high-resistance state (HRS), and the bottom sets to a 16-stage low-resistance state (LRS), reflecting the value's magnitude. The negative centroids have a reversed pattern. On the other hand, the key (or query) is first converted to an analog pulse by a digital-to-time converter (DTC). The pulsewidth is proportional to the input magnitude. This pulse connects to word lines (WLP/WLB) to control discharge duration from bit lines (BLP/BLN) to source lines (SL). Before computing, the SL is first reset to clamp voltage (V_{CLP}). The BLP precharges to $V_{\text{CLP}} - V_{\text{Read}}$, while BLN precharges to $V_{\text{Read}} + V_{\text{CLP}}$ for the positive key. The negative key has a reserved setting. Take the positive centroid and key as an example. The top cell is in HRS, leading to a low current from SL to BLP. The bottom cell is in LRS, leading to a high current from BLN to SL. The rest case of different polarity can be obtained by a similar method. When multiple rows are activated, the LRS cells dominate the accumulated current in SL.

Note that the BLPs/BLNs of ReRAM-CIMs in 12 attention cores share the same voltage source. Therefore, to increase the ReRAM monotonicity, the operational amplifier (OA) in the voltage buffer is used to clamp the BLP/BLN voltage to a stable state. On the other hand, the 16 × 16 ReRAM array only leads to a limited current in SL during parallel computing. Therefore, only one output capacitor is used to convert the SL current to output voltage. The power-hungry OAs are avoided. Although the output capacitor may not ensure the computing linearity, it can maintain the monotonicity. Because the clustering can inherently tolerate approximate computing as long as the closest centroid can be distinguished. The experimental results in Section V verify the design's validity.

C. Custom WTA Circuit in ReRAM-CIM

Although the voltage from the ReRAM crossbar might be nonlinearly distorted, representing inaccurate inner product results, the voltage monotonicity can be guaranteed for correct clustering operation. The WTA circuit only needs to detect the index of the largest voltage as the clustering result. Fig. 5 illustrates the WTA structure based on the binary tree. The WTA detects the winner from 16 inputs. The inner product results are passed to voltage-controlled delay lines (VCDLs) to convert voltage to time-domain delay. Higher voltage takes the lead in triggering a valid time delay. The delays, P_0 and P_1 are coupled in pairs and sent to a two-input arbiter. If P_1 first arrives, the arbiter outputs digital-1. Otherwise, the arbiter outputs digital-0. The OR gate passes the advanced delay phase to the arbiters in the next layer. The local winner completes until the global winner is determined.

Algorithm 2 shows how to convert the arbiter outputs in Fig. 5 to the clustering index. We commence by recording the arbiter outputs from bottom-up, left to right as 0, 10, 0011, 0011_0100. Then, we encode the arbiter outputs level by level. For Level 3, the arbiter output, logic-0, is equal to the most significant bit (MSB) of the clustering index. In addition, it also indicates that the winner path is generated from the rightmost subtree. Therefore, for Level 2, we select

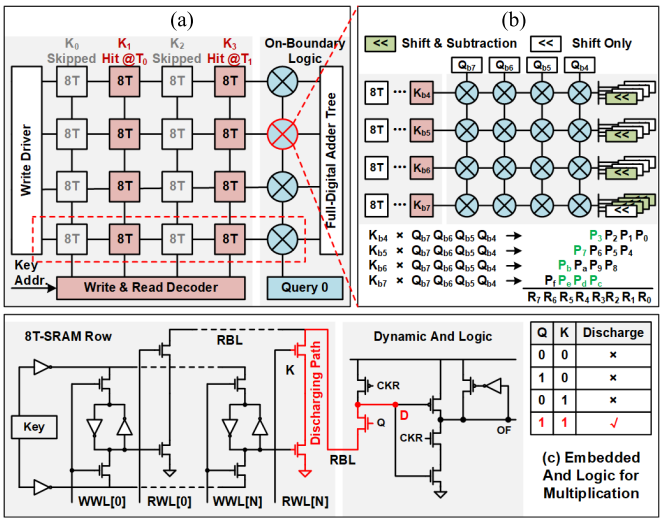

Fig. 6. (a) Computing-on-boundary digital SRAM-CIM, (b) 2's complementary signed computing, and (c) embedded dynamic AND gates for 1-bit multiplication.

arbiter AB-20's output, logic-0, as the second MSB of the clustering index. A similar process is used for Level 1. For the final Level 0, the formerly encoded clustering index, 001, indicates the winner path passes through the second arbiter AB-01. Hence, the paired output, logic-0, is encoded as the least significant bit of the clustering index. Consequently, the final clustering index is generated as 0010, which is the address for the winner signal V2.

D. Digital On-Boundary SRAM-CIM

HARDSEA contains two fully digital SRAM-CIM macros for precise relation computing and information summary. Unlike the approximate computing in ReRAM-CIM, the query, key, and value matrices in SRAM-CIMs are all represented as 8 bits for high-precision computation.

As shown in Fig. 6(a), the SRAM-CIM macro contains an 8T-SRAM array, an on-boundary computing logic, and necessary peripheral circuits, such as the driver and the decoder. Compared to the conventional crossbar, the computing-on-boundary SRAM-CIM is much more flexible in skipping the zero-involved computations, where the multiply-accumulation (MAC) is produced in a memory-access manner. Specifically, in every MAC cycle, the decoder converts the input key address to one-hot enabling signals. Only the keys of the strongly related tokens are enabled to be multiplied with the current query. For example, K_1 and K_3 are selected sequentially at cycles T_0 and T_1 to be multiplied with Q_0, while K_0 and K_2 are skipped. In contrast, the crossbar-based CIM activates the entire SRAM array for multiplication. Hence, the zero-involved computing of K_0 and K_2 cannot be easily skipped, resulting in redundant computation.

The on-boundary computing logic contains an 8×8 AND gate array and a custom full-digital adder tree for one 8-bit MAC operation. For brevity, Fig. 6(b) just illustrates the computing flow for the MSB. The 8-bit query is broadcast vertically, while the key is broadcast horizontally. The 64-bitwise multiplication results are generated and allocated to the adder tree. To accommodate signed (2's complementary) computing, the adder tree is configured to support both shifted addition and shifted subtraction. The bitwise multiplication results from the numerical bits are shifted first and then accumulated. At the same time, the ones from the sign bits are subtracted during the aggregation [marked as green in Fig. 6(b)].

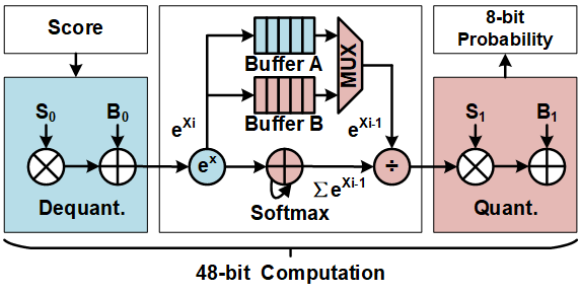

Fig. 7. Dequantized softmax engine.

We adopt the dynamic AND gate from our previous works [19], [55] for 1-bit multiplication. Fig. 6(c) shows that the dynamic gate fuses the readout circuit, the multiplier, and the dynamic latch into one gate. The read bitline (RBL) of the 8T SRAM is connected to a query-controlled NMOS. In every MAC operation, the CKR is set to low voltage first to precharge the latch D. Then, the gate outputs logic-0. Next, the CKR is set to high voltage to start the operation. The CKR is from the system clock. The path from latch D to RBL discharges only when the query and key are both logic-1. Then, the dynamic gate output logic-1 consequently. Otherwise, the dynamical power on the RBL is saved if either the query or key is zero. This feature exploits the bitwise sparsity without any explicit zero-detection logic.

E. Dequantized Softmax Engine

The softmax normalization converts the attention scores to probabilities for information summary. However, the Softmax is hardware-unfriendly because of nonlinear operations. To balance the accuracy and hardware consumption, we propose a dequantized Softmax engine, as illustrated in Fig. 7.

The scores from the query–key inner product are first dequantized to 48 bits using a pair of scaling and bias factors. The following operations also have a 48-bit precision to reduce the numerical error. After that, a pipeline of exponential computation, accumulation, and reciprocal operations is applied to calculate the Softmax results. 16-piecewise functions approximate the exponential and reciprocal units. Note that the normalized results $e^{x_i}/\sum e^{x_j}$ can be only produced after $\sum e^{x_j}$ is obtained. Double buffering is used to promote the pipeline. In other words, the Softmax engine collects e^{x_i} in Buffer A for the ith token while calculating the division from Buffer B for the last token. Finally, the 48-bit Softmax result is quantized again to generate the 8-bit probability. It reduces the hardware cost of the SRAM-CIM macro for probability–value information summary.

V. Evaluation

A. Evaluation Methodology

1) Benchmarks: We evaluate HARDSEA on two discriminate transformer models, BERT-Base (110 M parameters) [2]

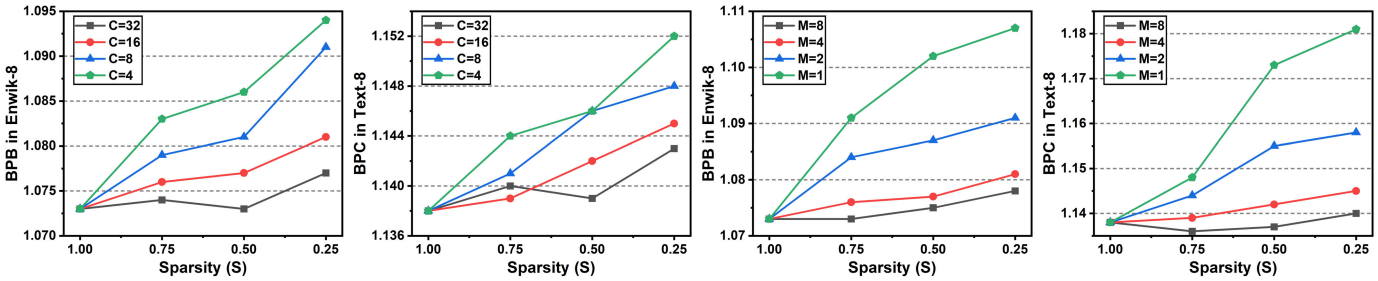

Fig. 8. Product-quantization-based sparse attention performance: GPT-2 performance under different centroid numbers on the (a) Enwik-8 dataset and (b) Text-8 dataset, GPT-2 performance under different subspaces on the (c) Enwik-8 dataset and (d) Text-8 dataset.

and GPT-2-Small (117 M parameters) [3]. Both models contain 12 transformer layers with 12 self-attention heads, where the size of the embedding features is set to 768. The evaluation datasets for BERT-Base are nine tasks from GLUE [5] datasets. The default sequence length is 128. We report Matthews correlation for COLA, F1 Score for MPRC, Person correlation for STS-B, and accuracy for the other tasks. All of these metrics are positively correlated with transformer model performance. Note that the problematic WNLI task is excluded from GLUE following [2], [40]. For the GPT-2-Small model, we test the language modeling task on two typical datasets: Enwik-8 and Text-8 [6]. The sequence length is set to 1024. We report bit-per-byte (BPB) for Enwik-8 and bit-per-character (BPC) for Text-8. Both metrics measure the ability of the transformer language model on text compression. A lower BPB or BPC value indicates better model performance.

2) Software Fine-Tuning Implementation: We fine-tune product-quantized sparse models on the pretrained counterparts from the Huggingface transformer library [54]. All tasks are executed using PyTorch v2.0 and CuDNN v11.7 on four Nvidia RTX 3090 GPUs. For BERT-Base, we use the Adam optimizer with a learning rate of 2e-5 and batch size of 32. The fine-tuning epochs vary from 3 to 15 depending on the target tasks. For GPT-2-Small, we use the Adafactor optimizer with a learning rate of 5e-3 and batch size of 64. We then fine-tune the model for 80 K steps with 10 K linear warm-up steps.

Transformer models should be quantized to a fixed-point representation for HARDSEA execution. Linear transformation, relation computing, and information summary are all quantized to 8-bit precision. Softmax normalization remains a floating point because HARDSEA reserves sufficient precision for nonloss computation (described in Section IV-E). Finally, we recover model accuracy by PyTorch's quantization-aware training.

3) Hardware Implementation: A HARDSEA prototype is designed under 28-nm CMOS technology with Verilog RTL. The custom 8T-SRAM and dynamical AND gate in SRAM-CIM are inherited from our previous works [19], [55] and packaged as standard cells for circuit implementation. We synthesize all digital blocks in HARDSEA with a Synopsys design compiler and use the Synopsys IC compiler for placement and routing. Then, the postlayout estimates the area and power consumption. To evaluate HARDSEA performance in real applications, we develop a cycle-level simulator with the assumption of a 128 GB/s HBM2. The HBM parameters are obtained from [57] to calculate the overall energy and latency.

TABLE II
ReRAM Operation Configuration

Operation	RST	SET	READ
WLP/WLN	2.5V	2V	2.5V
BLP/BLN	0V	1.2V	0.4V±0.2V
SL	1.2V	0V	0.4V

The ReRAM-CIM mainly consists of three parts: ReRAM array, analog peripheral circuit (including DTC, OAs, and WTA circuit), and digital LUT with paired adder for partial sum accumulation. To evaluate the ReRAM array, we adopt a 4-bit ReRAM device from [13], [44], which has been proven for equivalent arithmetical computation after including all error sources. Table II lists the ReRAM operation configurations. The power and area in [44] is scaled to 28 nm for evaluation. The analog peripheral circuits are designed with Cadence Virtuoso for area and power analysis, while the digital LUT and adders are synthesized, placed, and routed in Synopsys DC and IC compilers.

B. Algorithm Performance

1) Impact of Centroid and Subspace Number: We first analyze the impact of different centroid and subspace numbers on GPT-2 performance. As described in Table I, the ReRAM-CIM capacity is proportional to the centroid number. Therefore, we search for the fewest centroids to reduce the ReRAM-CIM cost. We first set a constant *Top-K* configuration and four subspaces for all attention heads. As shown in Fig. 8(a) and (b), the model performance increases as the centroid number rises from four to 32. With more centroids, the approximate inner product is much closer to the exact counterpart since more centroid permutations can reconstruct the query or key precisely. However, the memory and computation requirements increase linearly. Compared with allocating 32 centroids, GPT-2 with 16 centroids only results in a 0.004-BPB/0.003-BPC loss on Enwik and Text-8 tasks under different sparsity levels. Therefore, 16 centroids are sufficient for sparse attention.

Next, we find the optimal subspace number with the specific centroid set. Similarly, a constant *Top-K* configuration is applied to all attention heads. Fig. 8(c) and (d) shows that GPT-2 performance increases as the subspace number raises

TABLE III
PRODUCT-QUANTIZED ATTENTION PERFORMANCE AND COMPARISON

Model		COLA	MMLI	MRPC	QNLI	QQP	RTE	SST2	STSB	Enwik-8	Text-8
BERT	Baseline	56.5	83.9	88.9	90.7	90.7	65.7	**92.3**	88.6	/	/
	Sanger	53.2	83.2	**89.6**	**90.8**	90.5	**68.2**	91.1	88.8	/	/
	ReBERT	58.1	84.4	88.4	/	89.1	62.5	91.7	88.7	/	/
GPT-2	Baseline	/	/	/	/	/	/	/	/	1.16	1.17
	T12	/	/	/	/	/	/	/	/	1.11	1.18
	Transformer-XL (12 Layers)	/	/	/	/	/	/	/	/	**1.06**	1.15
Our Work	w/o ReRAM variation	**58.3**	**84.6**	89.3	90.7	**90.9**	68.1	92.2	**89.7**	1.082	**1.146**
	w/i ReRAM variation	56.9	83.1	86.5	90.2	89.7	64.4	91.4	88.9	1.171	1.224
	Sparsity	**0.31**	**0.12**	**0.16**	**0.17**	**0.25**	**0.15**	**0.33**	**0.28**	**0.15**	**0.24**

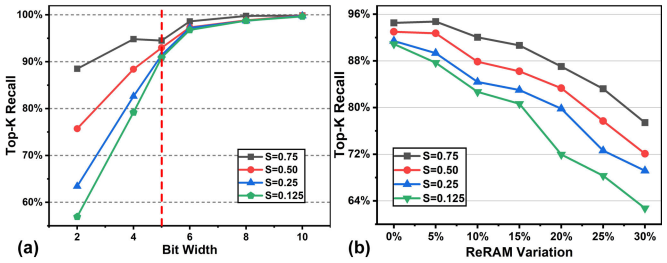

Fig. 9. Sensitivity of *Top-K* product-quantization clustering recall to (a) computing precision (without ReRAM variation) and (b) ReRAM variation (with 5-bit computing precision).

from one to eight. However, the performance improves slightly as increasing the subspace number from four to eight. For example, eight subspaces just lead to 0.003-BPB/0.005-BPC improvement when the model sparsity is 25%. Given that each subspace is paired with one LUT for partial sum aggregation, we use four subspaces to reduce the LUT cost.

In summary, fewer subspaces with more centroids are preferred to balance the model performance and hardware overhead. We use four subspaces with 16 centroids in the following experiments. Section V-B3 shows that this configuration can also ensure BERT-Base performance on the downstream GLUE tasks.

Another interesting phenomenon is that a sparser model could result in better performance, which could be partly attributed to the better concentration of self-attention on important information [60]. With more aggressive pruning, the model fails to extract enough features and leads to degraded performance.

2) Impact of ReRAM Bit Precision: Fig. 9(a) shows the impact of the quantized centroid on the token relevance prediction. For this experiment, the ReRAM array is considered ideal without device variation. Instead of using the numerical error, we choose the recall metric to measure the effectiveness since product quantization is just required to distinguish the strongly related tokens from the total. The baseline is the selected strongly related tokens under full-precision relevance computing. The recall denotes the portion of the approximate-computed *Top-K* token candidates to the baseline. As shown in Fig. 9(a), given the bit precision, the average recall over different tasks decreases as the *Top-K* resolution increases (75%–12.5% sparsity). It indicates that distinguishing fewer strongly related tokens from the total requires more accurate relevance prediction. In addition, the 5-bit product quantization can recall 90.9%–94.5% *Top-K* tokens at 12.5%–75% sparsity. Since the trivial quantization error can be theoretically compensated after fine-tuning, the 5-bit centroid virtually has no impact on the model performance.

Furthermore, we test the impact of ReRAM variation on product-quantization clustering. Previous works [14], [15] report that the ReRAM variation (including device-to-device variation and cycle-to-cycle variation) complies with Gaussian distribution. Following NeuroSim [16], such Gaussian variations are added to the assigned resistance in our simulation. As shown in Fig. 9(b), the recall metric remains stable with 5% device variation (3σ) under 5-bit computing precision. Higher variation leads to a lower clustering performance, as the resistance margin starts to decrease.

3) Accuracy and Comparison: Table III summarizes the product-quantized BERT-Base and GPT-2-Small performance. The baseline corresponds to the original models with dense attention. Without ReRAM variation, our method outperforms the dense BERT-Base slightly. Two factors contribute to the performance improvement. First, according to [44], [60], dynamic sparsity patterns can make self-attention more concentrated on the most relevant tokens as well as keep the long-term information dependency, which can lead to better performance. Second, we produce fine-grained fine-tuning on BERT-Base task by task, while the benchmark only sets the same retraining configuration. We also compare HARDSEA with state-of-the-art sparse transformers: ReBERT [30] and Sanger [40]. Compared to ReBERT with a static sparsity pattern, HARDSEA exploits self-attention sparsity in a data-driven fashion, thus resulting in better accuracy on all GLUE tasks. The Sanger also adopts dynamical sparsity and obtains similar accuracy to HARDSEA. But HARDSEA still wins five tasks out of eight. We also apply our pruning method to the GPT-2-Small model on Enwik-8 and Text-8 datasets. Compared to Transformer-XL [58], HARDSEA incurs a slight

performance degradation on the Enwik-8 dataset. That is because Transformer-XL deals with a much longer sequence to capture longer-term information dependency.

For the BERT-Base model with 128 input tokens, relevance computing and information summary lead to 50.4MOPS computations in one dense self-attention layer. For the GPT-2-Small model with 1024 input tokens, the computation amount is 3.2GOPS. In contrast, HARDSEA reduces the corresponding computation by 3.0×–8.3× with a 0.12-to-0.33 sparsity ratio. Meanwhile, the product-quantization clustering only leads to 3.1MOPS and 25.2MOPS computation overhead for BERT-Base and GPT-2-Small, respectively, which is far less than computations in dense models.

However, as shown in Fig. 9(b), the ReRAM variation exacerbates clustering performance and may severely degrade model performance in turn. Fortunately, variation-aware retraining can reduce the impact of device variation. Inspired by [15], we retrain the transformer parameters to compensate for device variation. Such method can be generally divided into three steps: 1) Run inference with ReRAM variation, and profile the inner-product result difference in clustering with the ideal output; 2) Approximate the output difference distribution as Gaussian distribution and add such noise to clustering during training; and 3) Adjust the transformer parameters for compensation and update the centroid with the exponential moving average in (1). The variation-aware retraining has the same hyperparameter configurations and sparsity ratio as the ideal one. The device variation is set to a medium value of 15% (3σ). As shown in Table III, compared to the ideal results, the BERT model just leads to 0.5%–2.4% accuracy degradation in most tasks with ReRAM-variation-aware training. For MPRC and RTE tasks, BERT suffers from higher accuracy loss at 3.1% and 5.4%, respectively. Two reasons contribute to this reduction. First, MPRC and RTE tasks undergo the severest sparsification. Distinguishing fewer strong-related tokens from the total requires higher computing precision. Second, MPRC and RTE have the smallest training texts. The limited text cannot fully reflect the error distribution caused by ReRAM variation. Consequently, the network performance cannot be fully compensated during retraining. On the other hand, compared to the ideal results, the GPT-2 model leads to 8.2% and 6.8% BPC loss in the Enwik-8 and Text-8 datasets, respectively. That is because the error caused by ReRAM variation would be propagated and accumulated along with the text generation for such a decoder-only model. We also notice that the system linearity can be further improved with redundant data storage, like thermometric coding [33]. Therefore, a 4-bit ReRAM device can ensure algorithm robustness.

C. Hardware Performance

1) Power Consumption and Area: Table IV reports the power and area breakdown of HARDSEA. HARDSEA contains 12 attention cores to process 12 self-attention heads in parallel. For a single self-attention head, each attention core consists of four 16 × 16 × 5-bit ReRAM-CIMs for token relevance prediction and two 128 × 64 × 8-bit SRAM-CIMs for self-attention computing. The 4-bit ReRAM characters are extracted from [13], [44], which have been proven for accurate arithmetical computation after including all noise sources. The ReRAM-CIM totally consumes 32.87 mW power and utilizes 0.35 mm^2 area, which only takes up about 16% power and 7% area of the entire ON-chip logic. In each ReRAM-CIM, the LUT stores the inner product results between different centroids in a 16-bit integer. Consequently, 16 centroids would generate 136 results and require 272B storage spaces. For 12 self-attention heads, 48 ReRAM-CIM macros are required with a total 12.75 KB LUT overhead, which is far less than model parameter storage. On the other hand, the SRAM-CIM has 112.63 mW power consumption and utilizes 2.31 mm^2 area, taking the largest portion in HARDSEA. The rest is contributed to the ON-chip SRAM and scratchpad, such as global memory and key clustering index memory. The entire HARDSEA accelerator occupies 4.95 mm^2 area and consumes 917.53 mW including the power consumption from the external HBM2 modules.

TABLE IV
HARDSEA POWER AND AREA BREAKDOWN

Module	Spec ×1 Head	Power (mW) ×12 Heads	Area (mm^2) ×12 Heads
1. ReRAM-CIM for Approximate Relevance Prediction			
ReRAM Array	16×16×5b/Array ×4	1.38	0.02
DTC	×64	2.84	0.06
WTA	×4	5.50	0.03
LUT	×4	26.15	0.25
2. SRAM-CIM for Accurate Attention Computing			
SRAM Array (QK)	128×64×8b/Array ×1	48.82	1.21
SRAM Array (PV)	128×64×8b/Array ×1	41.10	0.88
Digital Logic	×1	22.71	0.22
3. On-Chip SRAM			
Global Memory	12KB	49.12	2.19
Key Index	0.25KB	6.17	0.09
4. Enternal HBM2			
HBM2	/	713.75	/
5. Total			
wo DRAM	/	203.78	4.95
wi DRAM	/	917.53	/

2) WTA Efficiency: We compare the proposed WTA circuit with a series of ADCs with different resolutions. These ADCs can achieve different output precision for tasks from approximate clustering computing (3-/4-/5-bit ADCs) to accurate neural computing. The WTA circuit is designed with Cadence Virtuoso in 28-nm technology to estimate area and power. For ADCs, the power and area are extracted from NeuroSim [16], a typical simulator for CIM accelerators. To achieve the same throughput, one 16-input WTA circuit is compared with 16 ADCs in parallel. Fig. 10 shows the results. The WTA circuit, consisting of 16 VCDLs and 15 arbiters, consumes an energy of 36.6 pJ per operation,

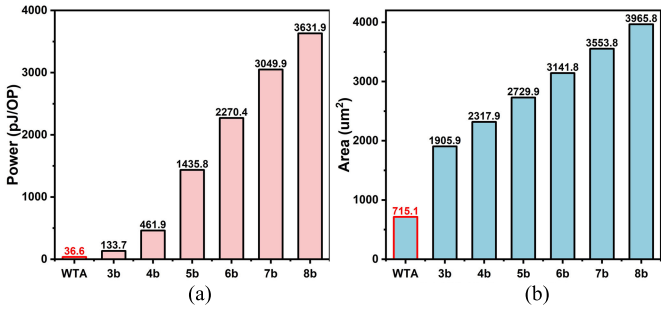

Fig. 10. (a) Power and (b) area comparison between a 16-input WTA circuit and 16 ADCs with different resolutions.

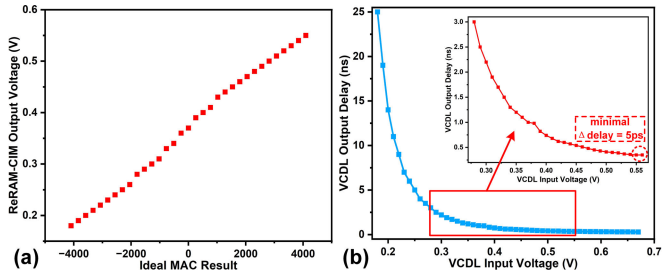

Fig. 11. (a) Simulated ReRAM-CIM output voltage versus ideal MAC results and (b) VCDL input voltage versus VCDL output delay.

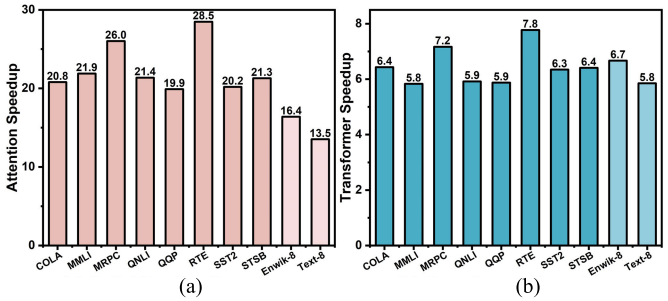

Fig. 12. HARDSEA speedup over RTX 3090 GPU (a) attention (only mapping relevance computing, Softmax, and information summary) and (b) transformer (linear layer included).

3.7×–99.2× better than ADCs. The overall area is 715.1 um^2, 2.7×–5.5× better than ADCs.

3) ReRAM-CIM Monotonicity: We also test the monotonicity of ReRAM-CIM hardware. As shown in Fig. 11(a), the ReRAM-CIM leads to a high computing monotonicity compared with the ideal MAC results. The output is directly connected to VCDL to be converted into the time delay. Then, the time delay passes through the time-to-digital arbiter in the WTA circuit. As shown in Fig. 11(b), the ReRAM-CIM output voltage falls within the range between 0.18 and 0.55 V, leading to a minimal delay difference as 5 ps, which meets the arbiter's 2-ps resolution [53].

4) Comparison With GPU: Figs. 12(a) and 13(a) show the speedup and energy efficiency over RTX 3090 GPU on self-attention processing, respectively. Note that only relevance computing $Q \times K^{\mathrm{T}}$, Softmax, and information summary $P \times V$ are mapped on HARDSEA. The energy efficiency is assessed by the saved energy, which is power × latency. For BERT-Base on the GLUE dataset, HARDSEA achieves 19.9×–28.5× speedup and 1377.3×–1894.3× energy reduction. The performance benefits from skipping the

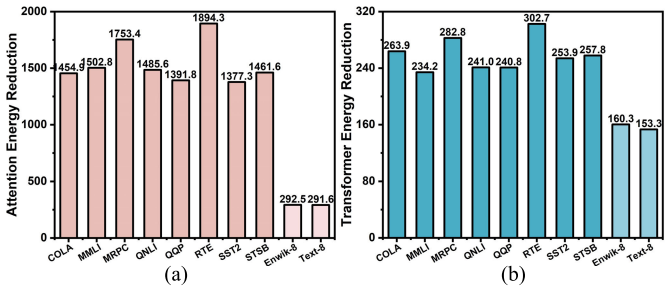

Fig. 13. HARDSEA energy reduction over RTX 3090 GPU (a) attention (only mapping relevance computing, Softmax, and information summary) and (b) transformer (linear layer included).

majority of self-attention due to the hybrid in-memory runtime pruning. For decoder-only GPT-2-Small, HARDSEA is 16.4×/13.5× faster and 292.5×/291.6× more energy-efficient on Enwik-8 and Text-8 datasets. The performance diminishing mainly comes from two aspects. First, HARDSEA needs to cascade attention cores to store all 1024 tokens and corresponding intermediate features. Thus, HARDSEA cannot fully activate 12 self-attention cores in parallel. Second, the decoder-only GPT-2 only generates one token in each iteration, which is less computing-intensive than the encoder-based BERT model.

To further compare the end-to-end performance of HARDSEA with the GPU baseline, we extend HARDSEA to support the entire transformer model including all linear layers. Note that the residual addition and layer normalization are excluded from the experiment. The feature and weight in the linear layers are also quantized to 8 bits and mapped to the SRAM-CIM macros. The ReRAM-CIM is shut off for power saving. As shown in Figs. 12(b) and 13(b), HARDSEA achieves 5.8×–7.8× speedup and 153.3×–302.7× energy efficiency on BERT-Base and GPT-2-Small models. Since HARDSEA is not specially optimized for linear layers, performance degradation (compared with attention-only processing) can be encountered. The ON-chip SRAM-CIM cannot fit the entire linear layers. Therefore, necessary external memory access increases the latency and power consumption. Another observation is that HARDSEA efficiency suffers more from linear layer processing on BERT-Base than GPT-2-Small. That is because linear layers become the bottleneck of execution latency and energy when the input sequence is small. With the product-quantized sparse attention, linear layers take up 73.3% latency and 68.2% power consumption for 128-token BERT-Base on average. In comparison, for 1024-token GPT-2-Small, linear layers take up 55.6% latency and 51.1% power consumption.

5) Comparison With ASIC Designs: Finally, Table V compares HARDSEA with state-of-the-art attention accelerators, including one FPGA design (STA [34]), two digital accelerators (ELSA [35] and DOTA [38]), and two CIM-based accelerators (ReTrans [42] and TransPIM [43]). For a fair comparison, only self-attention with relevance computing $Q \times K^{\mathrm{T}}$, Softmax, and information summary $P \times V$ are mapped on all works. In addition, we only report the performance on dense models since ReTrans and TransPIM do not support sparse self-attention. HARDSEA runs at a 300-MHz frequency and a 0.9-V power supply with a peak

TABLE V
HARDSEA SUMMARY AND COMPARISON WITH STATE-OF-THE-ARTS

	STA [34]	ELSA [35]	DOTA [38]	ReTrans [42]	TransPIM [43]	Our Work
Technology	16nm	40nm	22nm	32nm	22nm	28nm
Implementation	FPGA	Digital CMOS	Digital CMOS	ReRAM CIM	DRAM CIM	ReRAM/SRAM CIM Hybrid
Workload	BERT Only	BERT Only	BERT and GPT	Encoder Only	BERT and GPT	BERT and GPT
Precision	INT16	INT9	INT16	INT8	INT8	INT8
Area (mm^2)	/	1.26	2.75	/	2.15	4.95
Frequency (MHz)	200	1000	1000	/	500	300
Power (W)	26.590	0.956	3.018	0.175	1.046	0.918[2]
Throughput[1] (GOPS)	1466.7	1088.0	2000.0	81.9	734.0	Peak:921.6, 4189.1[3] Avg:802.1, 3641.5[3]
Energy Efficiency[1] (GOPS/W)	55.2	1138.1	662.7	467.7	701.7	Peak:943.7, 4284.4[3] Avg:821.3, 3728.7[3]
External Memory Cost Included	Yes	No	No	Yes	Yes	Yes

1: If not explicitly stated, the performance is obtained by mapping relevance computing, softmax and information summary. No model sparsity included.
2: Off-chip memory cost is included. The HBM parameters are from [57]
3: HARDSEA performance is obtained with 0.22 average sparsity for BERT-Base model on GLUE dataset.

throughput of 921.6GOPS. Considering QKV uploading from OFF-chip memory, HARDSEA yields an average throughput of 802.1GOPS and an average energy efficiency of 821.3GOPS/W. The hardware utilization is 87%. Compared to the-state-of-arts, HARDSEA has 1.2×–14.9× better energy efficiency. Note that ELSA achieves a higher performance than our work. That is because the OFF-chip memory access is excluded from the ELSA result. If considering OFF-chip memory cost, the energy efficiency of ELSA then degrades to 580.9GOPS/W, 1.4× lower than our work. Finally, for different tasks, HARDSEA can prune self-attention in BERT-Base and GPT-2-Small models with 0.22 and 0.20 average sparsity ratios, respectively. In turn, HARDSEA achieves 3.6TOPS throughput and 3.7TOPS/W energy efficiency. The improvement comes from two aspects. First, HARDSEA skips the majority of self-attention with a co-design methodology of lightweight product quantization and negligible ReRAM cost. In contrast, ELSA and DOTA induce expensive preprocessing for hashing and low-rank transformations. The relevance predictor in these works consumes about 6.3× more energy than our work. Second, the on-boundary-computing SRAM-CIM in HARDSEA exploits dynamical sparse attention without extra scheduling logic.

VI. CONCLUSION

This article presents HARDSEA, an algorithm-architecture-circuit codesigned accelerator for sparse self-attention in Transformer. It trades off hardware efficiency and algorithm precision by exploiting different characters of analog ReRAM-CIM and digital SRAM-CIM. The experiments show that HARDSEA achieves 13.5×–28.5× speedup and 291.6×–1894.3× efficiency over RTX 3090 GPU without accuracy loss. In addition, HARDSEA achieves an average energy efficiency of 821.3 GOPS/W at 300-MHz frequency and 0.9-V power supply, 1.2×–14.9× better than state-of-the-arts.

REFERENCES

[1] A. Vaswani et. al., "Attention is all you need," in *Proc. Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 6000–6010.
[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.
[3] A. Radford et. al., "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.
[4] T. B. Brown et al., "Language models are few-shot learners," 2020, *arXiv:2005.14165*.
[5] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–20.
[6] M. Mahoney, "Large text compression benchmark," Matt Mahoney Page, Nov. 2009. [Online]. Available: http://mattmahoney.net/dc/text.html
[7] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," 2020, *arXiv:2004.05150*.
[8] J. Qiu, H. Ma, O. Levy, W.-T. Yih, S. Wang, and J. Tang, "Blockwise self-attention for long document understanding," in *Proc. Findings Assoc. Comput. Linguistics*, 2020, pp. 2555–2565.
[9] N. Parmar et. al., "Image transformer," in *Proc. Mach. Learn. Res. (PMLR)*, 2018, pp. 4055–4064.
[10] Y. Rao, W. Zhao, B. Liu, J. Lu, J. Zhou, and C. J. Hsieh, "DynamicViT: Efficient vision transformers with dynamic token sparsification," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2021, pp. 13937–13949.
[11] H. Zhao et al., "Implementation of discrete Fourier transform using RRAM arrays with quasi-analog mapping for high-fidelity medical image reconstruction," in *IEDM Tech. Dig.*, Dec. 2021, p. 12.
[12] B. Lin et al., "A highly reliable RRAM physically unclonable function utilizing post-process randomness source," *IEEE J. Solid-State Circuits*, vol. 56, no. 5, pp. 1641–1650, May 2021.
[13] Z. Zhang et al., "In-sensor reservoir computing system for latent fingerprint recognition with deep ultraviolet photo-synapses and memristor array," *Nature Commun.*, vol. 13, no. 1, Nov. 2022.

[14] J. Reuben, M. Biglari, and D. Fey, "Incorporating variability of resistive RAM in circuit simulations using the Stanford–PKU model," *IEEE Trans. Nanotechnol.*, vol. 19, pp. 508–518, 2020.

[15] Z. He, J. Lin, R. Ewetz, J.-S. Yuan, and D. Fan, "Noise injection adaption: End-to-end ReRAM crossbar non-ideal effect adaption for neural network mapping," in *Proc. 56th ACM/IEEE Design Autom. Conf. (DAC)*, Jun. 2019, pp. 1–6.

[16] X. Peng, S. Huang, Y. Luo, X. Sun, and S. Yu, "DNN+NeuroSim: An end-to-end benchmarking framework for compute-in-memory accelerators with versatile device technologies," in *IEDM Tech. Dig.*, Dec. 2019, p. 32.

[17] H. Zhu et al., "A communication-aware DNN accelerator on ImageNet using in-memory entry-counting based algorithm-circuit-architecture co-design in 65-nm CMOS," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 10, no. 3, pp. 283–294, Sep. 2020.

[18] Y.-D. Chih et. al., "An 89TOPS/W and 16.3TOPS/mm^2 all-digital SRAM-based full-precision compute-in memory macro in 22 nm for machine-learning edge applications," *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, 2021, pp. 252–254.

[19] H. Zhu et al., "COMB-MCM: Computing-on-memory-boundary NN processor with bipolar bitwise sparsity optimization for scalable multi-chiplet-module edge machine learning," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, vol. 65, Feb. 2022, pp. 1–3.

[20] P. Yao et al., "Fully hardware-implemented memristor convolutional neural network," *Nature*, vol. 577, no. 7792, pp. 641–646, Jan. 2020.

[21] C. Li et al., "Analogue signal and image processing with large memristor crossbars," *Nature Electron.*, vol. 1, no. 1, pp. 52–59, Dec. 2017.

[22] R. V. W. Putra, M. A. Hanif, and M. Shafique, "ROMANet: Fine-grained reuse-driven off-chip memory access management and data organization for deep neural network accelerators," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 29, no. 4, pp. 702–715, Apr. 2021.

[23] Y. Zhang et al., "An 8-bit in resistive memory computing core with regulated passive neuron and bitline weight mapping," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 30, no. 4, pp. 379–391, Apr. 2022.

[24] J.-H. Kim, J. Lee, J. Lee, J. Heo, and J.-Y. Kim, "Z-PIM: A sparsity-aware processing-in-memory architecture with fully variable weight bit-precision for energy-efficient deep neural networks," *IEEE J. Solid-State Circuits*, vol. 56, no. 4, pp. 1093–1104, Apr. 2021.

[25] J. Yue et al., "STICKER-IM: A 65 nm computing-in-memory NN processor using block-wise sparsity optimization and inter/intra-macro data reuse," *IEEE J. Solid-State Circuits*, vol. 57, no. 8, pp. 2560–2573, Aug. 2022.

[26] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient transformers: A survey," 2020, *arXiv:2009.06732*.

[27] T. Tambe et. al., "EdgeBERT: Sentence-level energy optimizations for latency-aware multi-task NLP inference," in *Proc. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, 2021, pp. 830–844.

[28] A. Roy, M. Saffar, A. Vaswani, and D. Grangier, "Efficient content-based sparse attention with routing transformers," 2020, *arXiv:2003.05997*.

[29] N. Kitaev, L. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," 2020, *arXiv:2001.04451*.

[30] M. Kang, H. Shin, and L.-S. Kim, "A framework for accelerating transformer-based language model on ReRAM-based architecture," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 9, pp. 3026–3039, Sep. 2022.

[31] F. Tu et al., "A 28 nm 15.59 μJ/token full-digital bitline-transpose CIM-based sparse transformer accelerator with pipeline/parallel reconfigurable modes," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, vol. 65, Feb. 2022, pp. 466–468.

[32] S. Sridharan, J. R. Stevens, K. Roy, and A. Raghunathan, "X-former: In-memory acceleration of transformers," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 31, no. 8, pp. 1223–1233, Aug. 2023.

[33] P. K. Saragada and B. P. Das, "In-memory computation with improved linearity using adaptive sparsity-based compact thermometric code," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 30, no. 10, pp. 1473–1483, Oct. 2022.

[34] C. Fang, A. Zhou, and Z. Wang, "An algorithm-hardware co-optimized framework for accelerating N: M sparse transformers," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 30, no. 11, pp. 1573–1586, Nov. 2022.

[35] T. J. Ham et al., "ELSA: Hardware–software co-design for efficient, lightweight self-attention mechanism in neural networks," in *Proc. ACM/IEEE 48th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2021, pp. 692–705.

[36] T. J. Ham et. al., "A^3: Accelerating attention mechanisms in neural networks with approximation," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2020, pp. 328–341.

[37] H. Wang, Z. Zhang, and S. Han, "SpAtten: Efficient sparse attention architecture with cascade token and head pruning," in *Proc. IEEE Int. Symp. High-Performance Comput. Archit. (HPCA)*, Feb. 2021, pp. 97–110.

[38] Z. Qu, L. Liu, F. Tu, Z. Chen, Y. Ding, and Y. Xie, "DOTA: Detect and omit weak attentions for scalable transformer acceleration," in *Proc. 27th ACM Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Feb. 2022, pp. 14–26.

[39] J. Dass et al., "ViTALiTy: Unifying low-rank and sparse approximation for vision transformer acceleration with a linear Taylor attention," in *Proc. IEEE Int. Symp. High-Performance Comput. Archit. (HPCA)*, Feb. 2023, pp. 415–428.

[40] L. Lu et al., "Sanger: A co-design framework for enabling sparse attention using reconfigurable architecture," in *Proc. 54th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Oct. 2021, pp. 977–991.

[41] Y. Wang et al., "A 28 nm 27.5 TOPS/W approximate-computing-based transformer processor with asymptotic sparsity speculating and out-of-order computing," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, vol. 65, Feb. 2022, pp. 1–3.

[42] X. Yang, B. Yan, H. Li, and Y. Chen, "ReTransformer: ReRAM-based processing-in-memory architecture for transformer acceleration," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, Nov. 2020, pp. 1–9.

[43] M. Zhou, W. Xu, J. Kang, and T. Rosing, "TransPIM: A memory-based acceleration via software-hardware co-design for transformer," in *Proc. IEEE Int. Symp. High-Performance Comput. Archit. (HPCA)*, Apr. 2022, pp. 1071–1085.

[44] A. Yazdanbakhsh, A. Moradifirouzabadi, Z. Li, and M. Kang, "Sparse attention acceleration with synergistic in-memory pruning and on-chip recomputation," in *Proc. 55th IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2022, pp. 744–762.

[45] M. Zhou, Y. Guo, W. Xu, B. Li, K. W. Eliceiri, and T. Rosing, "MAT: Processing in-memory acceleration for long-sequence attention," in *Proc. 58th ACM/IEEE Design Autom. Conf. (DAC)*, Dec. 2021, pp. 25–30.

[46] H. Fan et al., "Adaptable butterfly accelerator for attention-based NNs via hardware and algorithm co-design," in *Proc. 55th IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2022, pp. 599–615.

[47] S. Hong et al., "DFX: A low-latency multi-FPGA appliance for accelerating transformer-based text generation," in *Proc. 55th IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2022, pp. 616–630.

[48] H. Wang, H. Xu, Y. Wang, and Y. Han, "CTA: Hardware–software co-design for compressed token attention mechanism," in *Proc. IEEE Int. Symp. High-Performance Comput. Archit. (HPCA)*, Feb. 2023, pp. 429–441.

[49] P. Dong et al., "HeatViT: Hardware-efficient adaptive token pruning for vision transformers," in *Proc. IEEE Int. Symp. High-Performance Comput. Archit. (HPCA)*, Feb. 2023, pp. 442–455.

[50] H. You et al., "ViTCoD: Vision transformer acceleration via dedicated algorithm and accelerator co-design," in *Proc. IEEE Int. Symp. High-Performance Comput. Archit. (HPCA)*, Feb. 2023, pp. 273–286.

[51] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 117–128, Jan. 2011.

[52] R. Guo, S. Kumar, K. Choromanski, and D. Simcha, "Quantization based fast inner product search," in *Proc. PMLR*, vol. 51, 2016, pp. 482–490.

[53] J. Yu, F. Dai, and R. Jaeger, "A 12-bit Vernier ring time-to-digital converter in 0.13μ CMOS technology," *IEEE J. Solid-State Circuits*, vol. 45, no. 4, pp. 830–842, Apr. 2010.

[54] T. Wolf et al., "Transformers: State-of-the-art natural language processing," in *Proc. Conf. Empirical Methods Natural Lang. Process., Syst. Demonstrations*, 2020, pp. 38–45.

[55] S. Liu et al., "A 28 nm 53.8 TOPS/W 8b sparse transformer accelerator with in-memory butterfly zero skipper for unstructured-pruned NN and CIM-based local-attention-reusable engine," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2023, pp. 250–252.

[56] M. Hu et al., "Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication," in *Proc. 53rd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2016, pp. 1–6.

[57] M. O'Connor et al., "Fine-grained DRAM: Energy-efficient DRAM for extreme bandwidth systems," in *Proc. 50th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2017, pp. 41–54.
[58] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 2978–2988.
[59] M. Liu and K. Chakrabarty, "Online fault detection in ReRAM-based computing systems for inferencing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 30, no. 4, pp. 392–405, Apr. 2022.
[60] G. Zhao, J. Lin, Z. Zhang, X. Ren, Q. Su, and X. Sun, "Explicit sparse transformer: Concentrated attention through explicit selection," 2019, *arXiv:1912.11637*.

Shiwei Liu (Graduate Student Member, IEEE) received the B.S. and M.S. degrees in microelectronics from Fudan University, Shanghai, China, in 2018 and 2021, respectively, where he is currently working toward the Ph.D. degree at the Frontier Institute of Chip and System.

His current research interests include reconfigurable computing architecture, in-memory computing, and neural network-specific accelerator design.

Chen Mu (Student Member, IEEE) received the B.S. degree from Southeast University, Nanjing, China, in 2020. He is currently working toward the Ph.D. degree at the Frontier Institute of Chip and System, Fudan University, Shanghai, China.

His current research interests include computing-in-memory, high-efficient AI accelerator, and ON-chip training.

Hao Jiang received the B.E. degree from Shandong University, Jinan, China, in 2021. He is currently working toward the Ph.D. degree at Fudan University, Shanghai, China.

His current research interests include high-performance computing-in-memory architecture and the signal processing unit for the brain–machine interface.

Yunzhengmao Wang received the B.S. degree in integrated circuit engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2019. He is currently working toward the M.S. degree in integrated circuit engineering at Fudan University, Shanghai, China.

His current research interests include low-power die-to-die interface design and chiplet technology.

Jinshan Zhang received the B.S. degree in integrated circuit engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2020. He is currently working toward the M.S. degree in computer engineering at the Academy for Engineering and Technology, Fudan University, Shanghai, China.

His current research interests include energy-efficient neural network processors and processing-in-memory system design.

Feng Lin received the B.S. degree in materials science from Fuzhou University, Fuzhou, China, in 2012.

He joined Fudan University, in 2018, as a Research Assistant. His work involves microprocessor hardware/firmware development and Linux driver designs. The project, "Medical Examination Robot of Traditional Chinese Medicine," in which he participated, won the silver award at the 47th Geneva International Invention Exhibition.

Keji Zhou (Member, IEEE) received the B.S. degree from the South China University of Technology, Guangzhou, China, in 2013, and the M.E. degree from Ningbo University, Ningbo, China, in 2016, and the Ph.D. degree from Fudan University, Shanghai, China, in 2021.

He is now an Assistant Research Fellow at Zhangjiang Laboratory, Shanghai. His current research interests include computing-in-memory, neuromorphic computing, and nonvolatile memory for neural networks.

Qi Liu (Member, IEEE) received the Ph.D. degree from Anhui University, Hefei, China, in 2010.

Then, he joined the IMECAS, Beijing, China, and became a Professor, in 2016. He is now a Professor at the Frontier Institute of Chip and System, Fudan University, and a Guest Professor at the IMECAS. His research interest focuses on the fabrication, characterization, and mechanism of the emerging memristor devices for nonvolatile memory, logic circuits, and neuromorphic computing applications.

Chixiao Chen (Member, IEEE) received the B.S. and Ph.D. degrees in microelectronics from Fudan University, Shanghai, China, in 2010 and 2015, respectively.

In 2015, he worked at Calterah Inc., Shanghai, as an Analog/Mixed Signal Circuit Design Engineer. From 2016 to 2018, he was a Postdoctoral Research Associate with the University of Washington, Seattle, WA, USA. In 2019, he joined Fudan University, where he is currently an Associate Professor with the Frontier Institute of Circuits and Systems. He is also the Director of the Integrated Chips Innovation Center of the State Key Laboratory of Integrated Chips and Systems, Fudan University. His current research interests include mixed-signal integrated circuit design and custom intelligent software–hardware codesigns.

Dr. Chen has served as a TPC Member for A-SSCC 2023.