HEIRS: Hybrid Three-Dimension RRAM- and SRAM-CIM Architecture for Multi-task Transformer Acceleration

Liukai Xu[1], Shuai Yuan[1], Dengfeng Wang[1], Yiming Chen[2], Xueqing Li[2], Yanan Sun[1]
[1]The Department of Micro-Nano Electronics, Shanghai Jiao Tong University, Shanghai, China
[2]BNRist/ICFC, EE, Tsinghua University
*Corresponding Authors: {sunyanan@sjtu.edu.cn, xueqingli@tsinghua.edu.cn}

ABSTRACT

Large-scale transformer with millions of weights achieves great success in multiple natural language processing (NLP) tasks. To release the memory bottleneck of multi-task model deployment, transfer learning tunes part of weights with shared parameters among tasks. Moreover, computing-in-memory (CIM) emerges as an efficient solution for neural network acceleration. With higher storage density, RRAM-CIM can store the large-scale model without costly weight loading, compared with another mainstream SRAM-CIM. However, the RRAM rewrite for tuned and dynamic weight matrix-vector-multiplication (MVM) in transformers requires high-cost RRAM writing in RRAM-CIM. Current hybrid CIM can compensate for the weakness of RRAM-CIM by adding SRAM-CIM with independent MVM. However, the tuned weights in transfer learning cannot be implemented due to the demand for the cooperative addition of MVM results from both shared and tuned weights. In this paper, a hybrid three-dimension RRAM-CIM and SRAM-CIM architecture (HEIRS) is proposed for multi-task transformer acceleration, with monolithically 3D integration of high-density RRAM-CIM and high-performance SRAM-CIM. The 3D RRAM-CIM with ultra-high density stores the whole model with mitigated off-chip weight loading. The SRAM-CIM is employed for efficiently performing dynamic weight MVM without RRAM rewrite. Moreover, a novel hybrid-CIM paradigm is proposed with an input selective adder tree, to support cooperative addition in transfer learning. Experiments show that, compared with RRAM-CIM and SRAM-CIM, the proposed HEIRS improves the energy efficiency by up to 7.83x and 2.29x on BERT, respectively. Meanwhile, the latency is also reduced by up to 85.5% and the storage density is enhanced by 7.2x, compared to RRAM-CIM.

KEYWORDS

RRAM-CIM, SRAM-CIM, Hybrid, Three-dimension, Transformer, Task Adaptation, Transfer Learning, NN Acceleration

DAC '24, June 23–27, 2024, San Francisco, CA, USA

ACM 979-8-4007-0601-1/24/06...
https://doi.org/10.1145/3649329.3657327

1 INTRODUCTION

Recently, neural networks (NNs) have achieved remarkable success in multiple tasks in NLP, which also leads to a significant increase in the NN parameter size in modern NN architectures like transformers. Meanwhile, to adapt the NN model to different tasks, weight needs to be tuned according to the specific task, which even increases the memory cost of the model deployment. By tuning part of weights with shared parameters, transfer learning [1-5] release the memory bottleneck in multi-task NN computation. However, although the number of task-specific weights is reduced, the data-intensive and computing-intensive transformer still suffers from the memory wall problem in the traditional von Neumann architecture. To mitigate the massive data movement with excessive energy and latency costs, the CIM technique [6-15] has been recognized as a promising candidate with parallel MVM operations performed in the memory array.

Current CIM works can be categorized based on memory technologies, such as resistance RAM (RRAM) [6-11] and static RAM (SRAM) [12-15]. Among these technologies, RRAM-CIM emerges as a promising candidate to store large-scale models and reduce off-chip DRAM access, thanks to the higher potential density of RRAM than SRAM in CMOS. To further enhance the on-chip capacity, 3D architecture [10, 11] is employed in RRAM-CIM for ultra-high storage density with stacking RRAM layers. Notably, existing CIM designs mainly focus on conventional NNs, with only MVM with static weight (SMVM). However, the MVM of attention layers in transformers involves two real-time generated operands, one of which is denoted as dynamic weight, as depicted in Fig. 1. Different from SMVM, frequent memory write is inevitable to perform the dynamic weight MVM (DMVM) and the task-specific model adaptation, leading to high energy and latency cost, and even more worrisome endurance concerns in traditional RRAM-CIM. Compared with RRAM-CIM, SRAM-CIM [12-15] achieves higher performance with less costly SRAM write to perform DMVM and adapt the model to new tasks. However, existing SRAM-CIM suffers from the large cell area and limited on-chip capacity, which induces off-chip memory access for weight loading and degrades energy efficiency.

Hybrid RRAM- and SRAM-CIM architectures [4, 16, 17] have also been explored previously. However, the conventional hybrid CIM in prior works suffers from accuracy loss or cannot support the cooperative addition of MVM results of shared weights and tuned weights, which incurs extra memory write for weight loading. In [4], the HyperX adapts the NN model for new tasks by tuning the last layers in SRAM-CIM, with initial layers in the

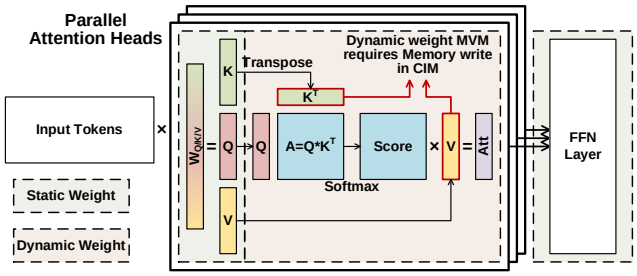

Figure 1: Typical attention layer structure in BERT with large parameter size and DMVM, which requires inevitable RRAM write and degrades the energy efficiency.

high-density RRAM-CIM. However, finetuning of last layers requires a substantial number of parameter updates and may not achieve competitive accuracy on the new task.

In this paper, a hybrid three-dimension RRAM-CIM and SRAM-CIM Architecture (HEIRS) is proposed for multi-task transformer acceleration. Employing the high-density 3D RRAM-CIM, HEIRS achieves both high on-chip capacity and high energy efficiency with significantly reduced off-chip memory access. The high performance of the transformer acceleration is attained with DMVM in SRAM-CIM. Moreover, with the tuned weights loaded in the SRAM-CIM array, a novel RRAM-and-SRAM hybrid-CIM is proposed with input-selective adder tree, to support the cooperative addition from RRAM-CIM and SRAM-CIM. Both RRAM-CIM and SRAM-CIM are employed in digital domain to ensure robustness. Key contributions of this paper are summarized as follows:

(1) A novel HEIRS architecture is proposed with hybrid 3D RRAM-CIM and SRAM-CIM. By integrating the high-dense 3D RRAM-CIM and high-performance SRAM-CIM, the proposed HEIRS achieves both high energy efficiency and low latency for transformer acceleration.

(2) A novel hybrid-CIM paradigm with the input selection adder tree is proposed for multi-task NN acceleration. With the task-specific weights loaded to SRAM-CIM arrays, the product of input and tuned weights can be accumulated with products of shared weights from RRAM-CIM. The high-cost RRAM write is avoided and the NN model can be quickly adapted to new tasks.

(3) The experiments are performed on the BERT-Large model with the CoLA, MRPC, and RTE datasets. The result shows that the proposed HEIRS improves the energy efficiency by up to 7.83x and 2.29x, compared with the traditional RRAM-CIM and SRAM-CIM, respectively. The latency of model adaptation is also reduced by up to 85.5% compared with RRAM-CIM.

The remainder of this paper is organized as follows. Section 2 introduces the background of transformer and transfer learning and challenges of previous RRAM-CIM and SRAM-CIM works. Section 3 presents the SMVM and DMVM in HEIRS performed by RRAM-CIM and SRAM-CIM. The hybrid-CIM for multi-task model adaptation is proposed in Section 4 with a hardware-aware weight complementary method to mitigate the number of loading weights. The evaluation results and analysis are shown in Section 5. Finally, the conclusion is provided in Section 6.

2 BACKGROUND AND CHALLENGES

2.1 Transformer and Transfer Learning

The transformers are built on encoder and decoder blocks with similar structures, including attention layers and feed-forward networks (FFN). As shown in Fig. 1, the MVM in transformer can be divided into SMVM and DMVM. The SMVM multiplies the input vector with static weight, including the Q/K/V generation and FFN layers. The DMVM multiplies two real-time generated matrices, including the $Q \times K^T$ and SV in transformers. One of the generated input matrices needs to be programmed to the memory array for CIM, inducing memory write operations.

Moreover, to adapt the NN model to different tasks, many transfer learning algorithms [1-4] have been proposed, as shown in Fig. 2. The traditional finetuning approach in Fig. 2(a) achieves good performance but is less feasible with millions of weights. The HyperX [4] in Fig. 2(b) enables multiple applications by tuning the last NN layers with frozen initial layers, which tune a large number of parameters for each task. The adapter [1] and LoRA [2] introduce extra adapter and branch layers with fewer parameters. However, as shown in Fig. 2(c) and (d), additional layers are also introduced to the system with extra energy and latency overhead. Diff pruning [3] in Fig. 2(e) plus the delta weight with the pre-trained weight to perform the model adaptation. By pruning the delta weight, diff pruning requires less tuned weights for high accuracy on multiple tasks. Moreover, since the overall NN structure maintains in diff pruning, the additional computation overhead is also avoided.

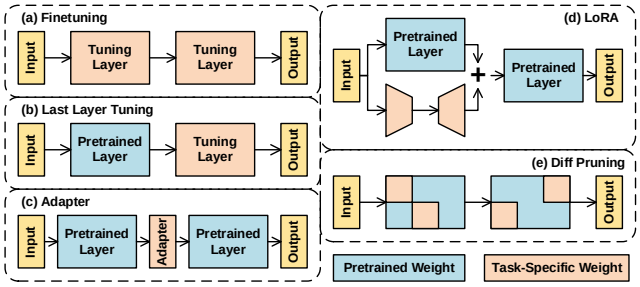

Figure 2: Transfer learning to adapt the NN model to new tasks, which requires the memory write to update weights.

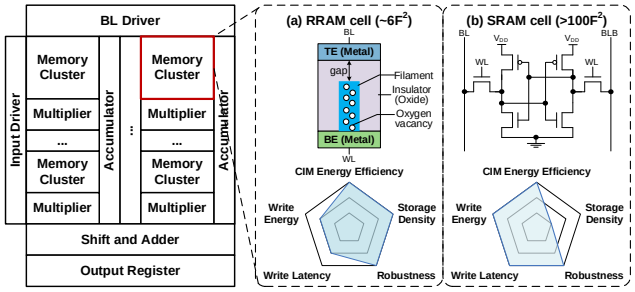

Figure 3: Existing CIM architectures based on: (a) RRAM with large on-chip capacity to store large-scale models and (b) SRAM with quick and energy-efficient write operation.

2.2 Previous RRAM-CIM and SRAM-CIM

To accelerate the MVM in NN, many CIM solutions have been proposed with RRAM and SRAM. As shown in Fig. 3, with inputs

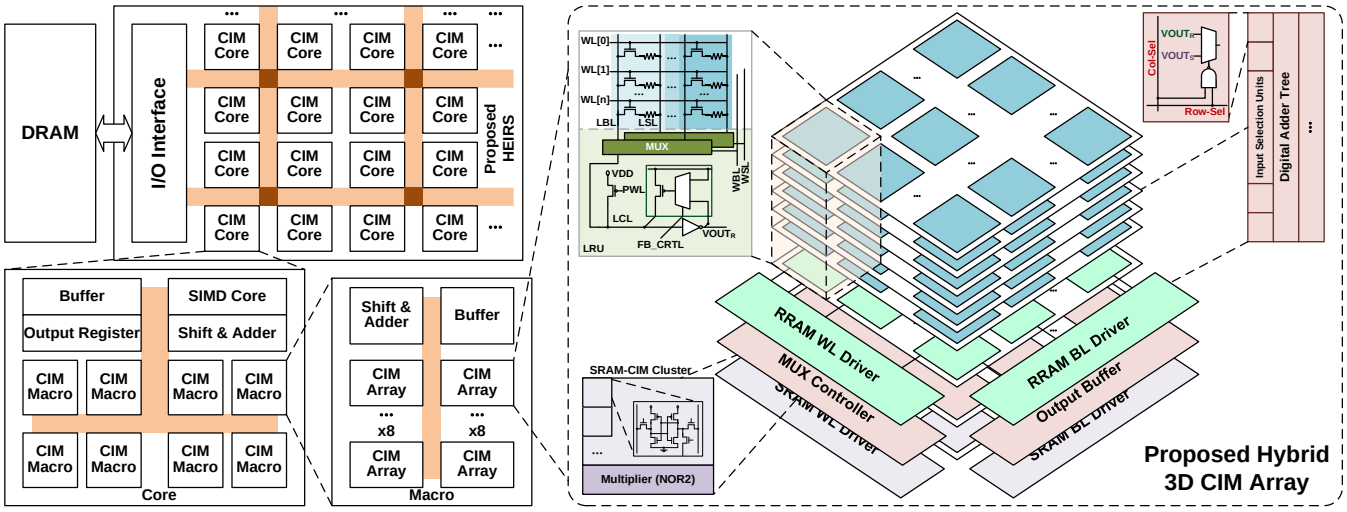

Figure 4: The overall architecture of the proposed HEIRS architecture.

quantized and sent to multipliers for multiplication with weight stored in memory clusters, partial products from the same column are generated and sent to accumulators for summation.

In RRAM-CIM, the NN weights are represented by the RRAM conductance. Thanks to the small footprint of RRAM cells ($6F^2$ in principle), RRAM-CIM offers high storage density and large on-chip capacity. However, the traditional analog-domain RRAM-CIM suffers from the accuracy loss, caused by the resistance variation. To address this issue, the digital-domain RRAM-CIM [8, 9] macros are proposed to recover the variation and achieve comparable computing accuracy with SRAM-CIM. Moreover, by employing adder trees, the digital domain RRAM-CIM exhibits a similar computing energy efficiency to SRAM-CIM. However, the large RRAM write energy and latency limit the RRAM-CIM from weight reloading scenarios, such as multi-task transformer acceleration. Compared with RRAM-CIM, the SRAM-CIM is more suitable for performing the DMVM and the parameter tuning with the low-cost SRAM write operation.

3 PROPOSED HEIRS ARCHITECTURE

3.1 An Overview of HEIRS Architecture

Fig. 4 shows the overview of the hierarchy of the proposed HEIRS. The HEIRS contains multiple CIM cores, constituted by data buffer, output registers, shift&adders, SIMD core, and CIM macros. Each macro is composed of local buffer, shift&adders, and hybrid 3D CIM arrays. As shown in Fig. 4, each CIM array consists of three layers: RRAM-CIM layer, accumulation layer, and SRAM-CIM layer.

The top RRAM-CIM layer comprises the WL driver, BL driver, and the vertically stacked 3D RRAM array with local recovery units (LRUs) [6]. The size of the stacked 3D RRAM is assumed to be 8 layers with 128 rows and 512 columns in each layer. Every 8 vertically stacked RRAMs form an RRAM group and are connected by the same *BL* and *BLB* attached to MUX. Through the MUX, each LRU is connected to an 8x32 RRAM cluster, with 32 RRAM groups and 8 RRAMs in each group. Composed of an inverter and a feedback loop, the LRU generates the product of 1-bit input and 1-bit weight in the digital domain, which is sent to the accumulation layer straight down for summation.

The bottom SRAM-CIM layer consists of SRAM WL driver, SRAM BL driver, and a 128x128 SRAM array with embedded NOR2 gates serving as multipliers [12]. Each SRAM is attached with a selection transistor. When the selection signal is high, the weight bit in the SRAM cell is sent to the NOR2 gate for multiplication with the 1-bit input. The 1-bit product is then sent straight up to the accumulation layer for further calculation.

Positioned in the middle of the RRAM-CIM layer and SRAM-CIM layer, the accumulation layer sums the partial products from CIM layers with adder trees. With a certain number of inputs of adder trees, the input selection unit (ISU) is placed before the adder tree, to select the input from a particular CIM layer. The summations generated by adder trees are sent to the addition unit in the CIM macro for further accumulation.

In HEIRS, both SMVM and DMVM are executed efficiently, without the costly RRAM write. Meanwhile, the weight tuning and the cooperative addition in transfer learning are supported by the hybrid-CIM, detailed in Section 4. Moreover, leveraging the ultra-high density of 3D RRAM-CIM, the off-chip DRAM access for weight loading is also mitigated to avoid energy efficiency degradation. To fully utilize the high-density RRAM-CIM, the entire hybrid 3D array is designed with cross-layer area matching to maximize the area efficiency. As depicted in Fig. 5, in RRAM-CIM, the layout of the MUX and LRU with 8 RRAM layers stacked above is designed with matching area of one SRAM column in a cluster in the SRAM-CIM layer. Furthermore, the adder tree in the accumulation layers is designed within the area constraint according to the corresponding memory size to avoid the extra area overhead. In this context, the 28T/14T interleaving adder tree is employed for accumulation [12].

3.2 SMVM and DMVM in HEIRS

In the proposed HEIRS, the pretrained weights of transformer models are initially stored in the high-density 3D RRAM-CIM layer. With the proposed HEIRS architecture, both SMVM and

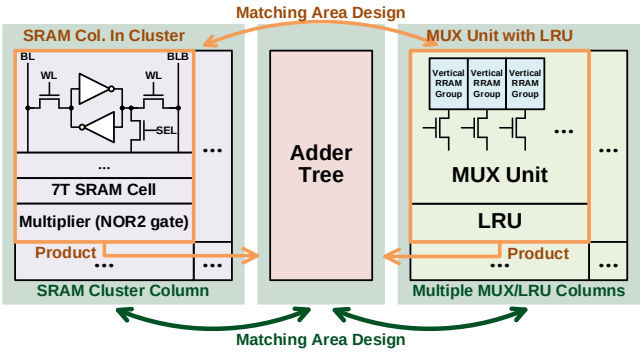

Figure 5: The cross-layer area matching design in HEIRS to maximize the area efficiency.

DMVM operations can be performed efficiently, without the high-cost RRAM write operation.

The SMVM is performed by RRAM-CIM in HEIRS with LRU and RRAM cluster. The static weight bits are represented by the RRAM resistance. For single-level cells, the high resistance state is '0', and the low resistance state is '1'. Only one RRAM cell is selected in each cluster each time for computing. During the calculation, the *PWL* turns low first to precharge the inner computing node *LCL*. After the precharging, the input bit is applied to the *WL* corresponding to the selected RRAM, and the relevant *LSL* is selected to connect to *LCL*. The MUX unit in the feedback loop also switches to the output of the inverter. When the input is '1' (*WL* is high), according to the resistance state, the *LCL* can be discharged to '0' through the RRAM or maintained at '1' by pulling up through the feedback loop. When the input is '0' (*WL* is low), the *LCL* keeps high with no discharge happening. Therefore, the output signal $VOUT_R$ changes to high only when both the input and weight bits are '1'. The 1-bit digital product is then sent to the adder tree in the accumulation layer. As depicted in Fig. 6(a), since the adder trees are already occupied by the RRAM-CIM for SMVM, the computing of the SRAM-CIM layer is stopped. However, the dynamic weights generated can still be loaded to the SRAM-CIM array to prepare for the DMVM.

The DMVM operation is performed by the SRAM-CIM in HEIRS with the multiplier and the SRAM cluster. The generated K^T or V serving as dynamic weights are programmed into SRAMs at first. During the calculation, by controlling the selection transistor, the target weights are loaded to the NOR2 gate with the input bit. The digital 1-bit output is then generated and sent vertically to the accumulation layer for summation. As shown in Fig. 6(b), during the DMVM operations in SRAM-CIM, the computing of SMVM is stopped with the idle RRAM-CIM, since the adder trees are already occupied.

4 Hybrid-CIM for Multi-task Model Adaptation

With the rapidly developing NN applications, task switching with the same model becomes more common, where the weights need to be updated for model adaptation. In HEIRS, with the pretrained model stored in RRAMs, the task-specific tuned-parameters can be loaded to SRAM array quickly and perform the hybrid-CIM together with the pretrained weight in RRAM.

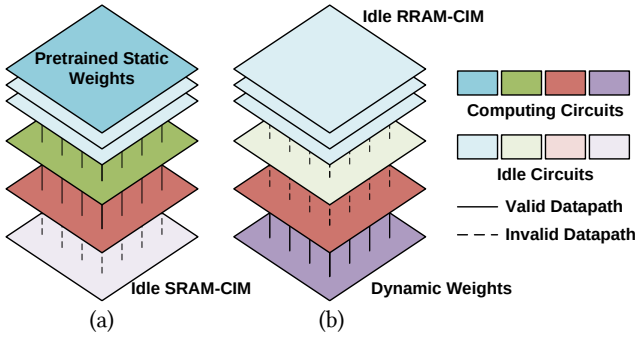

Figure 6: (a) The RRAM-CIM for SMVM and (b) the SRAM-CIM for DMVM in the proposed HEIRS architecture.

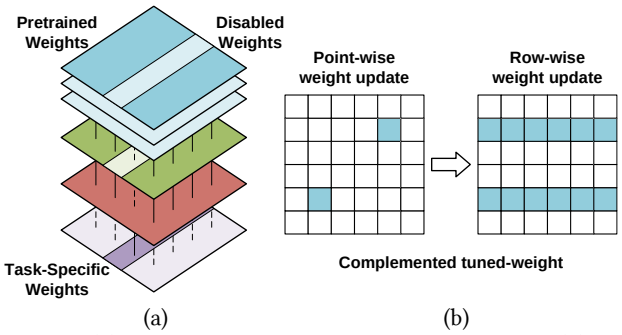

Figure 7: (a) The hybrid-CIM for model adaptation. (b) An example of the row-wise tuned weight complementary to simplify the array control.

The model adaptation and hybrid-CIM of HEIRS are shown in Fig. 7(a). The weights in the pretrained model which need to be updated are disabled in the RRAM-CIM (represented by the light blue part in Fig. 7(a)). Meanwhile, the task-specific tuned weights are loaded to the corresponding position in SRAM array (represented by the deep purple part). During the hybrid-CIM, inputs used to multiply with the disabled weights are now sent to the SRAM-CIM layer and multiplied with the tuned weights. Therefore, the MVM result of the tuned weights is generated and sent to the accumulation layer, together with the MVM result of the shared weights from the RRAM-CIM layer. By controlling the ISU, the cooperative addition of the partial products generated by the SRAM-CIM and RRAM-CIM can be performed by adder trees in the accumulation layer. Leveraging the digital-domain RRAM-CIM and SRAM-CIM with full-swing output, the HEIRS can perform the accumulation within the same adder tree.

Moreover, as shown in Fig. 7(b), with diff pruning illustrated in Section 2.1, the pruned task-specific weights usually distribute in a scatter pattern in weight matrix, which is not suitable for the CIM array controlling. Therefore, a hardware-aware weight complementary method is proposed for weight update. The proposed method updates the whole weight row or column with the tuned weight inside, which makes it possible for row-wise or column-wise control. An example of row-wise complementary is shown in Fig. 7(b). The scattered two tuned-parameters are complemented to whole weight rows. The decision to complement the updated weight to a row or column depends on the complemented weight number, which is determined by the array size and the tuned weight distribution in the weight matrix.

5 EVALUATION AND ANALYSIS

The energy efficiency and the performance of the proposed HEIRS architecture are evaluated and discussed in this section. Since the previous hybrid CIM works as separated RRAM-CIM and SRAM-CIM which cannot perform cooperative addition in transfer learning [4, 16], or suffers from the accuracy loss caused by the RRAM variation [17], the modified digital-domain RRAM-CIM [6] and SRAM-CIM [12] with promising computation accuracy are chosen as baselines in this context.

5.1 Experimental Setup

A typical transformer model, BERT-Large, is used to evaluate the proposed HEIRS architecture, on the CoLA, MRPC, and RTE datasets. Both weight and activations are quantized to 16-bit. Two transfer learning algorithms, finetuning and diff pruning [3], are evaluated to perform the model adaptation. According to the array size and weight complementary method, the overall number of loaded weights is less than 4%. For hardware, the architecture configurations of the proposed HEIRS and two baselines are listed in Table 1 in detail. With the physics-based Verilog-A RRAM model [18], the power of the RRAM-CIM layer and SRAM-CIM layer are obtained from the SPICE simulation in 28nm CMOS technology. The power supply voltage is 0.9V.

5.2 Result and Analysis

Energy efficiency. Fig. 8 shows the evaluation results of the energy efficiency. Compared with RRAM-CIM with finetuning, HEIRS achieves up to 7.83x improvement in energy efficiency with less tuned weight and no RRAM write operation. Even with the diff pruning employed in RRAM-CIM for minimizing the number of task-specific parameters, HEIRS still achieves 1.31x improvement with hybrid-CIM, avoiding RRAM programming. Meanwhile, with high-density 3D RRAM-CIM holding the whole pretrained model on-chip, HEIRS greatly mitigates the off-chip DRAM access and enhances the energy efficiency by 2.29x, compared with the SRAM-CIM with limited on-chip capacity. Moreover, in single-task scenarios with less demand for RRAM write, the HEIRS still enhances the energy efficiency by 12.0% by eliminating the RRAM write, compared with the RRAM-CIM.

Latency. The evaluated normalized latency is also shown in Fig. 8. All of the latency is normalized to the NN inference without model adaptation in HEIRS. As the target task switches, the proposed HEIRS architecture can adapt the NN model to a new task with only 0.8% latency overhead. Compared with the RRAM-CIM with diff pruning, the latency is reduced by 24.8% by eliminating the time-costly RRAM write operation. Moreover, HEIRS achieves an 85.5% reduction in latency, in comparison to the RRAM-CIM with finetuning, where millions of weights need to be updated. Compared with the SRAM-CIM, the latency is reduced by 18.2% with the greatly reduced off-chip memory access for weight loading. For single-task inference, by eliminating the RRAM write operations for DMVM, the latency of the NN inference is reduced by 10.5%.

Storage Density. With the 3D stacked RRAM-CIM, the storage density of the proposed HEIRS is improved by 7.2x and 3.14x respectively, compared with the RRAM-CIM with the 2D array and nvSRAM-CIM [17]. Meanwhile, to support the 3D signal controlling, the area overhead of the proposed HEIRS is 11.1% higher than the SRAM-CIM.

Table 1: Configuration parameters of the proposed HEIRS architecture and baselines [6, 12, 19]

Proposed HEIRS Accelerator			
Array Num.	16	**Macro Num.**	8
Core Num.	72	**Technology**	28nm
Bandwidth	128-bit	**Storage density**	43.0bit/um^2
RRAM-CIM Layer in HEIRS			
Stacked Layers	8	**Array Size**	128 x 512
Cluster Size	8 x 32	**Cell Precision**	1-b
LRU Number	32 x 128	**CIM Power**	5.51mW
RRAM parameters [19]			
RRAM Set Energy	10.9pJ/bit	**Write Latency**	50ns
RRAM Reset Energy	10.1pJ/bit		
SRAM-CIM Layer in HEIRS			
Array Size	128 x 128	**Cluster size**	8 x 16
CIM Power	4.43mW	**Write Latency**	2ns
SRAM Write Energy	1.2fJ/bit		
Accumulation Layer in HEIRS			
Adder Tree Num.	8	**Power**	2.17mW
Adder Tree Input	16 x 16-b		
Baseline RRAM-CIM Accelerator [6]			
Array num.	108	**Macro num.**	8
Core num.	72	**Technology**	28nm
Array Size	128 x 512	**Cluster Size**	16 x 8
LRU Number	32 x 128	**Storage Density**	6.05 bit/um^2
Baseline SRAM-CIM Accelerator [12]			
Array num.	16	**Macro num.**	8
Core num.	72	**Technology**	28nm
Array Size	128 x 128	**Cluster Size**	8 x 16
NvSRAM-CIM Accelerator [17]			
Array Size	256 x 256	**Storage Density**	13.7 bit/um^2

Accuracy. Both weights and activations in transformers are quantized to 16-bit. With the robust full-digital CIM mechanism in RRAM-CIM, SRAM-CIM, and hybrid-CIM, the proposed HEIRS achieves similar accuracy with the software.

5.3 Discussion

Task-specific Weight Percentage. To further confirm the improvement achieved by HEIRS, an exploration is performed with the percentage of the task-specific weights. As shown in Fig. 9(a), compared to the RRAM-CIM with finetuning weights, where all of the RRAMs need to be programmed for model adaptation, the energy efficiency improvement descends with the number of task-specific weights increasing, results from the increasing off-chip weight loading. Moreover, the latency overhead remains stable as the number of update weights increases, thanks to the hybrid-CIM paradigm in the HEIRS with quick SRAM write. Fig. 9(b) shows the comparison result of HEIRS and RRAM-CIM both with diff pruning. With more task-specific weights, the HEIRS achieves higher improvement in energy efficiency with lower latency. The evaluation results show that HEIRS improves the energy efficiency by 1.59x with the latency reduced by 39.8%, when 10% of weights in the whole model need to be updated.

Endurance. With the SRAM-CIM for DMVM and hybrid-CIM for transfer learning, the RRAM write is totally eliminated. Therefore, the proposed HEIRS exhibits a prolonged lifespan.

Scalability. With multiple chips cascading, the HEIRS can accommodate larger transformer models, since the NN layers are executed in sequence. The energy overhead for data transfer between chips may degrade the energy efficiency slightly.

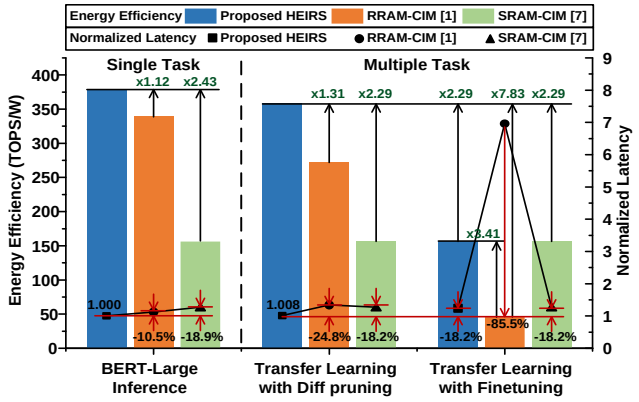

Figure 8: The energy efficiency and the normalized latency of the proposed HEIRS architecture.

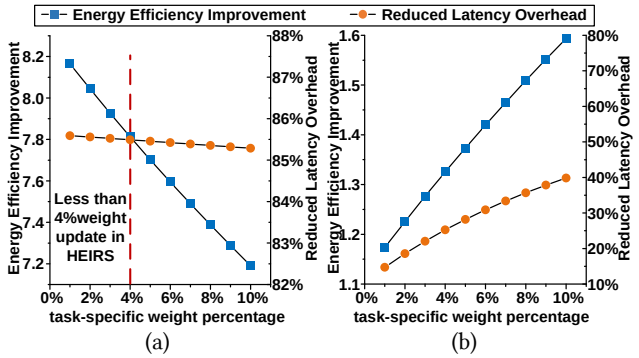

Figure 9: With increasing task-specific weight number, (a) the energy efficiency improvement of the proposed HEIRS decreases but the latency reduction remains stable, (b) HEIRS achieves higher improvement in the energy efficiency and latency, compared with the pruned task-specific weight RRAM-CIM baseline.

6 CONCLUSION

In this paper, a hybrid three-dimension RRAM- and SRAM-CIM architecture, HEIRS, is proposed for the energy-efficient multi-task transformer acceleration. With the high-density RRAM-CIM and high-performance SRAM-CIM, the proposed HEIRS performs both SMVM and DMVM in transformers efficiently, with mitigated costly RRAM write operation and off-chip DRAM access. Meanwhile, with the novel hybrid-CIM paradigm in HEIRS, the cooperative addition of the MVM results from the shared weights and tuned weights can be performed directly with the low-cost SRAM write operation. Moreover, by employing the ultra-high dense 3D RRAM-CIM, the storage density of HEIRS is improved by 7.2x compared with the traditional RRAM-CIM. The experimental results show that the proposed HEIRS improves the energy efficiency of transformer inference to up to 7.83x and 2.29x respectively, compared with RRAM-CIM and SRAM-CIM. The latency can also be reduced by up to 85.5%, compared with the traditional RRAM-CIM with finetuning.

ACKNOWLEDGMENTS

This work was supported in part by National Key R&D Program of China (#2023YFB4502200 , #2021YFA0717400), National Natural Science Foundation of China (#62174110, #U21B2030), and Natural Science Foundation of Shanghai (#23ZR1433200).

REFERENCES

[1] Y.-L. Sung, J. Cho and M. Bansal, "VL-ADAPTER: Parameter-Efficient Transfer Learning for Vision-and-Language Tasks," *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5227-5237, June 2021.

[2] J. E. Hu, Y. Shen, P. Wallis, *et al.*, "LoRA: Low-Rank Adaptation of Large Language Models," *ArXiv*, Vol. abs/2106.09685, October 2021.

[3] D. Guo, A. M. Rush and Y. Kim, "Parameter-Efficient Transfer Learning with Diff Pruning," *Annual Meeting of the Association for Computational Linguistics*, pp. 4884–4896, August 2021.

[4] A. Kosta, E. Soufleri, I. Chakraborty, *et al.*, "HyperX: A Hybrid RRAM-SRAM partitioned system for error recovery in memristive Xbars," *Design, Automation & Test in Europe Conference & Exhibition*, pp. 88-91, March 2022.

[5] Z. Fan, Q. Zhang, P. Abillama, *et al.*, "TaskFusion: An Efficient Transfer Learning Architecture with Dual Delta Sparsity for Multi-Task Natural Language Processing," *Proceedings of the 50th Annual International Symposium on Computer Architecture* pp. 1-14, June 2023.

[6] M. Lee, W. Tang, Y. Chen, *et al.*, "Victor: A Variation-resilient Approach Using Cell-Clustered Charge-domain computing for High-density High-throughput MLC CiM," *ACM/IEEE Design Automation Conference* pp. 1-6, July 2023.

[7] B. Kim, S. Li and H. Li, "INCA: Input-stationary Dataflow at Outside-the-box Thinking about Deep Learning Accelerators," *IEEE International Symposium on High-Performance Computer Architecture* pp. 29-41, February 2023.

[8] V. Sharma, H. Kim and T. T. H. Kim, "A 64 Kb Reconfigurable Full-Precision Digital ReRAM-Based Compute-In-Memory for Artificial Intelligence Applications," *IEEE Transactions on Circuits and Systems I*, Vol. 69, No. 8, pp. 3284-3296, June 2022.

[9] Y. He, J. Yue, X. Feng, *et al.*, "An RRAM-Based Digital Computing-in-Memory Macro With Dynamic Voltage Sense Amplifier and Sparse-Aware Approximate Adder Tree," *IEEE Transactions on Circuits and Systems II*, Vol. 70, No. 2, pp. 416-420, September 2023.

[10] Q. Huo, Y. Yang, Y. Wang, *et al.*, "A computing-in-memory macro based on three-dimensional resistive random-access memory," *Nature Electronics*, Vol. 5, No. 7, pp. 469-477, July 2022.

[11] P. Lin, C. Li, Z. Wang, *et al.*, "Three-dimensional memristor circuits as complex neural networks," *Nature Electronics*, Vol. 3, No. 4, pp. 225-232, April 2020.

[12] J. Chen, F. Tu, K. Shao, *et al.*, "AutoDCIM: An Automated Digital CIM Compiler," *ACM/IEEE Design Automation Conference*, pp. 1-6, July 2023.

[13] F. Tu, Z. Wu, Y. Wang, *et al.*, "TranCIM: Full-Digital Bitline-Transpose CIM-based Sparse Transformer Accelerator With Pipeline/Parallel Reconfigurable Modes," *IEEE Journal of Solid-State Circuits*, pp. 1798 - 1809, October 2022.

[14] F. Tu, Z. Wu, Y. Wang, *et al.*, "MulTCIM: Digital Computing-in-Memory-Based Multimodal Transformer Accelerator With Attention-Token-Bit Hybrid Sparsity," *IEEE Journal of Solid-State Circuits*, pp. 1-12, 2023.

[15] S. Liu, P. Li, J. Zhang, *et al.*, "A 28nm 53.8TOPS/W 8b Sparse Transformer Accelerator with In-Memory Butterfly Zero Skipper for Unstructured-Pruned NN and CIM-Based Local-Attention-Reusable Engine," *IEEE International Solid- State Circuits Conference (ISSCC)*, pp. 250-252, February 2023.

[16] G. Krishnan, Z. Wang, I. Yeo, *et al.*, "Hybrid RRAM/SRAM In-Memory Computing for Robust DNN Acceleration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 4241-4252, August 2022.

[17] Y. Sun, D. Wang, L. Xu, *et al.*, "CREAM: Computing in ReRAM-Assisted Energy- and Area-Efficient SRAM for Reliable Neural Network Acceleration," *IEEE Transactions on Circuits and Systems I: Regular Papers*, Vol. 70, No. 8, pp. 3198-3211, August 2023 2023.

[18] P. Chen and S. Yu, "Compact Modeling of RRAM Devices and Its Applications in 1T1R and 1S1R Array Design," *IEEE Transactions on Electron Devices*, Vol. 62, No. 12, pp. 4022-4028, October 2015.

[19] T. F. Wu, B. Q. Le, R. Radway, *et al.*, "A 43pJ/Cycle Non-Volatile Microcontroller with 4.7µs Shutdown/Wake-up Integrating 2.3-bit/Cell Resistive RAM and Resilience Techniques," *IEEE International Solid-State Circuits Conference*, pp. 226-228, February 2019.