A DRAM-based Near-Memory Architecture for Accelerated and Energy-Efficient Execution of Transformers

Gian Singh
gsingh58@asu.edu
Arizona State University
Tempe, AZ, USA

Sarma Vrudhula
svrudhul@asu.edu
Arizona State University
Tempe, AZ, USA

ABSTRACT

Transformers-based language models have achieved remarkable accuracy in various NLP tasks, employing self-attention mechanisms primarily based on matrix multiplication. However, their significant size leads to data movement issues, causing latency and energy efficiency challenges in conventional Von-Neumann systems. To mitigate these issues, several in-memory and near-memory architectures have been proposed. This paper introduces PACT-3D, a near-memory architecture featuring novel computing units integrated with DRAM banks. PACT-3D significantly reduces latency by 1.7× and improves energy efficiency by 18.7× compared to state-of-the-art near-memory architectures.

CCS CONCEPTS

• **Computer systems organization** → **Single instruction, multiple data**; • **Computing methodologies** → *Natural language processing*; • **Hardware** → **Memory and dense storage**.

KEYWORDS

In/Near-memory Processing, LLMs, Transformers, DRAM, Memory Wall, Energy Efficiency

ACM Reference Format:
Gian Singh and Sarma Vrudhula. 2024. A DRAM-based Near-Memory Architecture for Accelerated and Energy-Efficient Execution of Transformers. In *Great Lakes Symposium on VLSI 2024 (GLSVLSI '24), June 12–14, 2024, Clearwater, FL, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3649476.3658732

1 INTRODUCTION

Several state-of-the-art natural language processing (NLP) models such as GPT-4 [13], BERT [4], BART [10], etc., are based on the *Transformer* architecture. Transformers employ a self-attention mechanism that creates a context of the input sequence of data and improves the prediction capabilities of the Transformer model. Self-attention involves finding a correlation between every pair of words in the input sequence to determine their relationship and dependencies, making it a highly compute-intensive task. As the NLP tasks increase in complexity, it also increases the size of the Transformer model requiring high memory bandwidth and capacity. For example, GPT-3 has 175 billion parameters. Hence, the Transformer models are both data and compute-intensive.

Conventional CPU/GPU-based architectures are highly energy inefficient to execute Transformers due to the massive data movement on the channel connecting the DRAM to CPU/GPU. This also leads to an increase in the latency of computation. To overcome the memory bottleneck, the in/near-memory architectures have been particularly successful. In/near-memory architecture brings computation closer to the memory by either placing digital logic in memory or by using the memory arrays as parallel compute units.

SRAM and DRAM have been used extensively to design in/near-memory architectures. In SRAM, either analog multiply and accumulate (MAC) with small ADCs (quantization $<$ 4 bits) are used [17] or bit-serial digital computation is employed [8]. SRAM-based compute-in-memory (SRAM-CIM) is usually implemented in the cache of CPU/GPU of small size ($<$ 100 MB), requiring at least one-time data transfer from an external larger capacity memory such as a DRAM. Thus, for large models, data transfers on the memory channel dominate the energy consumption of the system.

A DRAM-based near-memory architecture eliminates all the data movement on the memory channel. A DRAM has a much larger capacity (>10 GB) and provides parallelism at various levels of memory hierarchy such as Ranks [6], Banks [9], Memory arrays [7] etc, which can be exploited by integrating the compute elements inside DRAM. The data parallelism (#bits accessed in parallel) and hence, the number of associated compute elements decreases as we move from the memory array to rank in the memory hierarchy. However, due to the stringent area and power constraints of the DRAM, only primitive compute elements can be placed near memory arrays. Therefore, the prior DRAM-based near-memory architectures have either low compute parallelism [6], reduced DRAM capacity [9], or have high latency for arithmetic operations [7, 14].

The key operation of Transformers includes large-scale matrix-matrix multiplication (MM). MM operation generates a lot of internal data movement and requires a large number of computation resources. To meet this requirement, this paper presents a DRAM-based near-memory architecture PACT-3D, that uses the large parallelism of in/near array architectures and substantially reduces the latency of computing arithmetic operations as compared to prior in-array architectures. PACT-3D uses an array of Neuron Processing Elements (NPEs) (as described in [16]) interfaced with the outputs of the row buffer of each DRAM bank. They have extremely small area and low-power compute elements that provide a high degree of compute parallelism through SIMD operation. The NPEs are composed of digital configurable neurons (CNs), and local registers, whose area and power are substantially lower than a functionally

GLSVLSI '24, June 12–14, 2024, Clearwater, FL, USA

ACM ISBN 979-8-4007-0605-9/24/06
https://doi.org/10.1145/3649476.3658732

equivalent CMOS implementation [16]. The main contributions of this paper are summarized below:

- PACT-3D is a new near-memory architecture for Transformers, which integrates NPEs within the strict area, power, and timing constraints of the DRAM.
- PACT-3D supports multiple bit-precisions (4, 8, 16 bits) as the NPEs can configured during runtime to operate on varying bit-width of data. It does this with high energy efficiency.
- PACT-3D is evaluated on encoder only and encoder-decoder-based Transformer architectures against the state-of-the-art near-memory architectures also designed for Transformers.
- For different workloads, PACT-3D achieves on an average 28.6× reduction in the latency against a GPU, 1.7× reduction in latency, and 18.7× reduction in energy consumption as compared to the state-of-the-art near-memory architecture.

2 BACKGROUND AND PRIOR WORK

2.1 Transformers

Transformer models have become central in Natural Language Processing (NLP) tasks. They consist of three main layers: an embedding layer, multiple encoder layers, and a classification layer as shown in Fig. 1. The embedding layer transforms input tokens or words into vectors, forming an embedding matrix that serves as the input to the fully connected (FC) layers within the encoder. The FC layers produce Query (Q), Key (K), and Value (V) matrices through matrix multiplication operations with weight matrices.

The encoder layers include self-attention (SA) layers, each containing multiple self-attention heads. The SA layers aim to identify semantic dependencies among input tokens, a distinctive feature that sets Transformers apart from traditional deep-learning models. In the SA layers, the Q and K matrices undergo multiplication, normalization via softmax operation, and multiplication by the V matrix. The resulting attention output is fed into a feed-forward (FFN) layer within the encoder and to subsequent encoder layers.

Decoder layers, while similar to encoder layers, include an additional cross-attention layer that attends to the encoder's output. This modification allows the decoder to consider information from the input sequence while generating the output sequence.

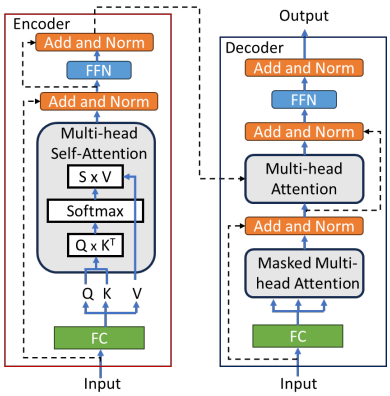

Figure 1: Structure of the encoder and decoder in Transformers.

2.2 In/Near-memory Transformers Acceleration

The execution of Transformers includes a series of large-sized matrix-matrix multiplication (MM) as described in section 2.1. Such MM operations require large computation resources and bandwidth for energy-efficient execution. Recently, several near-memory architecture proposals such as TransPIM [18], X-Former [15], HAIMA [5], utilize high internal memory parallelism and increase computation parallelism. X-Former uses a hybrid of SRAM-based and emerging non-volatile memory (NVM) based compute-in-memory structures. It uses analog crossbar arrays to perform MAC operations. However, due to analog operations, the use of ADC, and the necessity for a larger external DRAM to store Transformer parameters, X-Former's computation encounters reliability issues and does not effectively alleviate the memory bottleneck in computation.

On the other hand, HAIMA [5] and TransPIM [18] are based on highly scalable and high-capacity DRAM namely, high bandwidth memory (HBM). They perform digital computation and are integrated with a host CPU. Hence, HAIMA and TransPIM are used as baseline architectures for comparison with the proposed design. The TransPIM [18] architecture adds digital logic to the DRAM banks to perform computation and solve the communication challenge of implementing Transformers. TransPIM adopts existing schemes, such as Ambit [14], to perform the point-wise multiplications on data in a bit-serial manner. While this offers maximal parallelism, the multiplication has high latency. For processing *n-bit* data, *n* rows must be activated (ACT) and precharged (PRE) serially. Moreover, numerous majority operations are required, each involving multiple Activate-Activate-Precharge (AAP) operations, necessitating **hundreds of cycles** to execute a **single 8-bit multiplication** [7, 11, 14]. Furthermore, **the AAP command also disrupts DRAM timing** and the execution of the majority operation **requires modifications to row decoder** to activate multiple rows in parallel. To add all the partial products, and store the intermediate data, TransPIM adds an auxiliary compute unit (ACU) to each bank consisting of an adder tree and data buffer. TransPIM also uses a ring broadcast unit to transfer data between banks without using the shared global buffer to reduce the data-transfer latency.

The HAIMA [5] architecture uses a hybrid of SRAM and DRAM-based compute-in-memory (CIM) architectures. It distributes the Transformer workload among the compute elements in DRAM, SRAM, and the host CPU. The DRAM-CIM in HAIMA is based on a modification of TransPIM architecture. Dedicated 8-bit multipliers are used in each bank replacing point-wise multiplication of TransPIM to reduce the latency. **This however increases the area of HAIMA over TransPIM by 3×**. The SRAM-CIM unit of HAIMA is based on the architecture Colonnade [8]. It computes parallel 8-bit MAC in SRAM with a **area overhead of 48%**. Since HAIMA uses SRAM-CIM and DRAM-CIM, there are high latency and energy-consuming data transactions on the CPU-DRAM channel. Additionally, SRAM's lower parallelism and higher energy consumption than a DRAM leads to energy-efficiency degradation.

Summary of The Two SoA Designs:

(1) TransPIM uses bit-serial multiplication based on prior architectures [7, 14], which results in high computation latency (hundreds of cycles) for the multiplication operation.

(2) Like several prior architectures, TransPIM requires modifications to DRAM memory cells and the row decoder thereby increasing the cost of DRAM.
(3) TransPIM requires changes to the DRAM timing protocol.
(4) HAIMA distributes the workload among SRAM-CIM and DRAM-CIM, therefore the high energy and high latency data transactions on the CPU-DRAM bus cannot be avoided.
(5) Due to the small size of SRAM, SRAM-CIM is energy efficient only for relatively smaller workloads.

Key Features of PACT-3D: The Proposed Design

(1) The compute elements in PACT-3D consume substantially **(125×) lower power** and have about **16× lower area** as compared to a bit precision equivalent CMOS standard cell implementation of a MAC unit [16]. Thousands of elements are connected to the memory arrays without reducing the DRAM capacity. This SIMD style operation improves parallelism and therefore, throughput.
(2) PACT-3D does not require modifications to the DRAM memory array or row decoder.
(3) It uses conventional DRAM commands for its operation and adheres to the timing protocol.
(4) Though the architecture performs bit-serial computation, the latency of the multi-bit operations is substantially reduced by eliminating the memory writes for intermediate data. The low-latency bit-serial operation makes the architecture ideal for a low-cost implementation.

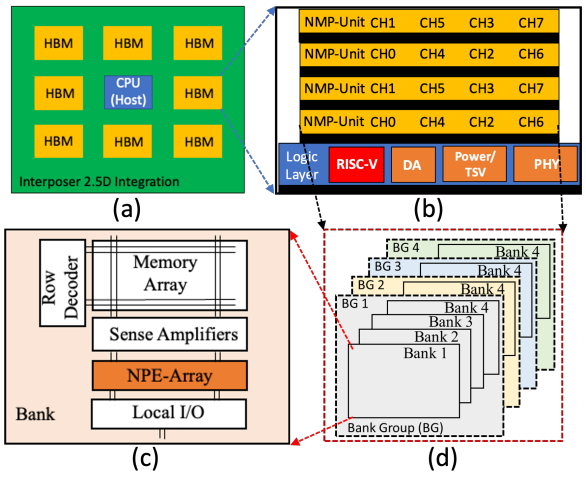

Figure 2: Top-level architecture of PACT-3D using High Bandwidth Memories (HBM) integrated with a host CPU.

3 PACT-3D ARCHITECTURE

Fig. 2a shows the top-level architecture of PACT-3D. It includes High Bandwidth Memory (HBM) cubes integrated with a host CPU on an interposer. An HBM consists of multiple DRAM layers and a base logic layer connected using the through silicon vias (TSVs) to form a 3D integrated memory with high density. Each DRAM layer in the HBM consists of *NMP-Units* formed by an array of Neuron Processing Elements (NPEs) called *NPE-Array* connected to a DRAM bank (see Fig. 2c). An NPE-Array is integrated with a memory array without interfering with the timing constraints or access protocols of the memory. An NPE is the basic computing element that can be instantly configured to perform different arithmetic, logic, and other operations common to DNNs and Transformers [16]. NPE-Array enables the parallel execution of the matrix-matrix (MM) and matrix-vector (MV) multiplications of the Transformer layers. Also, as shown in Fig. 2b, the logic layer of HBM includes the RISC-V processors to compute the normalization and the softmax function of the Transformer. The NMP-Unit of PACT-3D can be adapted to different DRAM organizations such as 2D-DIMM (DDR, GDDR, LPDDR, etc.) and 3D DRAM (HMC and HBM). PACT-3D is fully scalable with DRAM capacity, organization, and the DRAM interface with the host CPU.

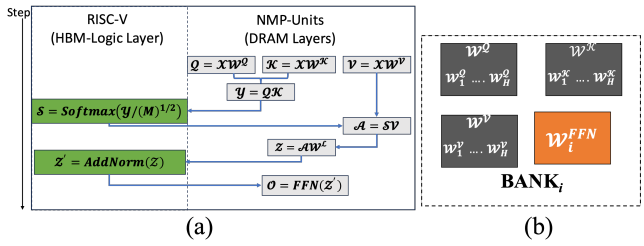

Figure 3: (a) Datalflow of Transformer execution in PACT-3D, (b) Initial data mapping in each memory bank of PACT-3D.

3.1 PACT-3D Data-flow and Transformer Execution

Transformers Layer Mapping and Execution Data-flow in PACT-3D: Consider a transformer architecture as shown in Fig. 1 for I input tokens of M length each. Let there be H heads in the multi-head attention layer (MHAL) of the transformer. The main computation steps in the transformer are:

(1) **Fully Connected Layer (FC)**: It involves matrix multiplication operation of input tokens positional embedding matrix $\mathcal{X} \in \mathcal{R}^{IxM}$ and weight matrices $\mathcal{W}^{\mathcal{Q}} \in \mathcal{R}^{MxM}$, $\mathcal{W}^{K} \in \mathcal{R}^{MxM}$, and $\mathcal{W}^{V} \in \mathcal{R}^{MxM}$ to generate Query ($\mathcal{Q} \in \mathcal{R}^{IxM}$), Key ($\mathcal{K} \in \mathcal{R}^{IxM}$), and Value ($\mathcal{V} \in \mathcal{R}^{IxM}$) matrices respectively. Given H heads in the transformer, $\mathcal{Q}, \mathcal{K}, \mathcal{V}$ are divided into H parts by column and multiplied by the corresponding weight matrices to generate all $\mathcal{Q}_h, \mathcal{K}_h, \mathcal{V}_h, h \in H$ in parallel.
(2) **Multi-Head Attention Layer (MHAL)**: It consists of 3 matrix multiplications, $\mathcal{Y}_h = \mathcal{Q}_h\mathcal{K}_h^T$, $\mathcal{S}_h = \text{Softmax}(\mathcal{Y}_h/(M)^{1/2})$, and $\mathcal{A}_h = \mathcal{S}_h\mathcal{V}_h$, where $\mathcal{A}_h$ is the attention weight matrix of a head. The final attention matrix $\mathcal{A}$ is created by concatenating all $\mathcal{A}_h, h \in H$ and is then multiplied with a weight matrix $\mathcal{W}^L \in \mathcal{R}^{MxM}$ to generate the output of MHAL, $\mathcal{Z} = \mathcal{A}\mathcal{W}^L$.
(3) **Feed Forward Network (FFN)**: It involves FC layers that take the attention output matrix as an input and generate the block output which can be used as an input for the next block such as decoder, encoder, or an output layer for classification.

Fig. 3a shows the data flow of layers of the Transformer. The host CPU (Fig. 2a) offloads the execution to HBM cubes which consist of NMP-Units and RISC-V processors. The NMP-Units perform all

the matrix multiplication operations in parallel as will be explained later in this section, while the RISC-V processors in the logic layer perform the softmax and normalization operation.

To extract maximum memory parallelism and reduce the data movement across the memory banks, input token-based data sharding [18] is used in PACT-3D. The input tokens are divided among all the available banks and all the data needed to process all layers of the transformers is written in all the banks as shown in Fig. 3b. In this way, all the banks can operate in parallel and can handle end-to-end Transformer inference for a set of input tokens. Further, to enable fast interbank communication, PACT-3D uses a ring broadcast unit as used in the TransPIM [18] and HAIMA [5]. In PACT-3D matrix multiplication (MM) is obtained by scheduling multiple independent vector-vector multiplication (dot product) computations in a NMP-Unit.

Matrix Multiplication Execution in a NMP-Unit: As shown in Fig. 4, PACT-3D adds NPEs to each bank in the DRAM to form a single NMP-Unit. The NPEs inside the DRAM bank are placed between the bit-line sense amplifier (BLSA) output and the local I/O of the bank. BLSA latches the entire row data (C-bits) of the memory array and delivers the maximum amount of data in parallel inside DRAM to the compute elements. Each NPE is connected to K (here $K = 8$) BLSA output bits, and therefore, there are C/K NPEs in each bank. The NPEs work in SIMD fashion by sharing the control signals generated by an external controller. They perform operations on the operands local to the bank they are directly interfaced with. Multiple operands are placed in different rows and share the same columns connected to an NPE shown in Fig. 4.

Each NPE performs a dot product of two vectors $OP1$ and $OP2$ using a multiply and accumulate operation (MAC). An NPE receives elements of the input vectors sequentially by using the DRAM row activation (ACT) and precharge (PRE) commands. A single DRAM activation activates all the bits in a row and supplies to the BLSA and hence to all connected NPEs. All the NPEs perform the same operation on different vectors providing massive parallel execution of the matrix multiplication operation. For example, in DDR4 memory, a row consists of 8192 bits ($C = 8192$), which enables $n = \frac{C}{K} = \frac{8192}{8} = 1024$ parallel dot product operations in a single bank. The NPEs store the intermediate results in local registers and finally write back the results (C-bits) to the rows reserved in the memory array for the outputs using the write drivers in the local I/O block. There are no writebacks to the memory banks during the computation of the dot-product of vectors as is the case with many prior DRAM-based near-memory architectures [3, 7].

3.2 Neuron Processing Element (NPE)

Element-wise multiplication and accumulation are two basic operations in a vector dot product computation. PACT-3D uses an NPE [16] to perform these operations. The structure of the NPE is shown in Fig. 5. It consists of four computing clusters, a set of four local registers and external inputs, control signals, and an internal connectivity routing hub.

NPE Operations: Each computing cluster consists of non-CMOS compute primitives described in section 3.3 to execute 5-bit primitive operations in **one or two clock cycles**. These primitive operations are **Addition, Comparison, and Logic** as shown in Fig. 5.

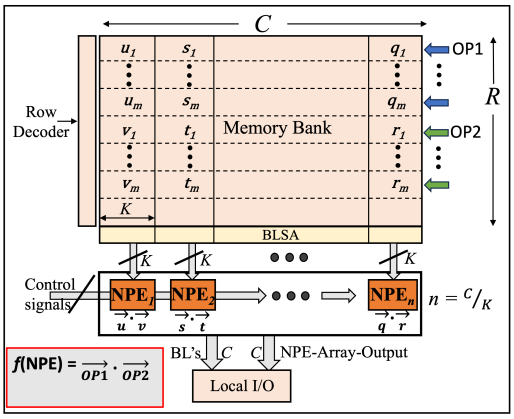

Figure 4: Data Mapping on to a bank of PACT-3D to compute inner products of vectors in parallel.

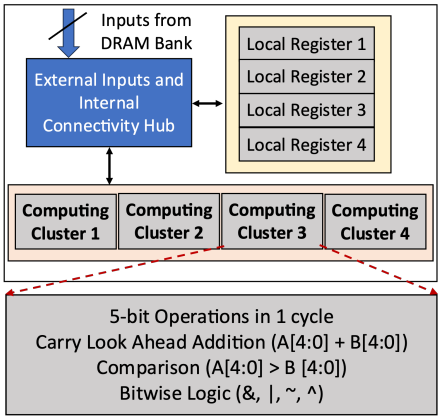

Figure 5: Neuron Processing Element (NPE) [16].

The addition operation takes **two clock cycles** to compute in which carryout (C_{out}) of ≤ 5 bits is computed first in a single cycle and then Sum bits ($S_i, 1 \leq i \leq 5$) are computed in the next cycle. The comparison and logic operations on operands with bit-width ≤ 5 are computed in a **single cycle**. The NPE can also perform ($N > 5$), addition, comparison, and logic, by decomposing the N-bit operations into 5-bit primitive operations and executed sequentially on the NPE as described in detail in [16].

Multiplication operation on NPE: is computed by decomposing multiplication into logic operations to compute the partial products and addition operation of the partial products. An NPE takes multiple clock cycles to compute a multiplication operation.

Note: Compared to conventional MAC unit which computes in a single cycle, the multi-cycle operations on NPE still consume less energy as each NPE has substantially **(125×) lower power** and has about **16× lower area**. Due to the much smaller area, many NPEs can be replicated in the same area as a single MAC and operated in a SIMD fashion resulting in higher throughput.

An NPE achieves lower power and area by using **unique, non-CMOS** logic primitives called **Configurable Neurons (CNs)** (see section 3.3 for details) in each of its computing clusters which allows an NPE to (1) implement multiple functions on the same

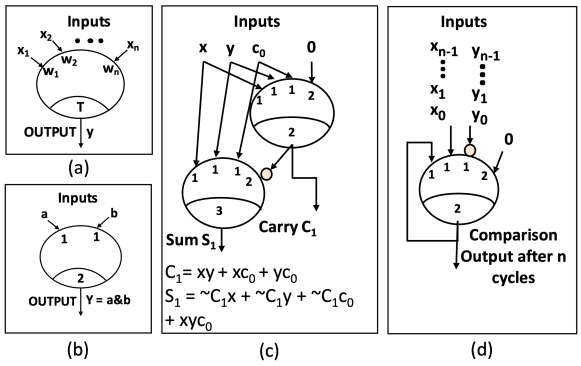

Figure 6: (a) Graphical representation of threshold logic function (TF). (b) TF for an AND gate, (c) TF for full adder. (d) TF for sequential comparison of two n bit numbers. Note: All these functions have the same physical implementation.

hardware structure, (2) instantaneously switch between the various functions by configuring the CN, and (3) utilize the exact hardware to compute an operand of particular bit-width.

3.3 Logic Primitive Configurable Neuron (CN)

A Boolean function $f(x_1, x_2, \cdots, x_n)$ is called a threshold function if there exists a set of weights $W = (w_1, w_2, \cdots, w_n)$, and a threshold T such that[1]

$$f(x_1, x_2, \cdots x_n) = 1 \;\Leftrightarrow\; \sum_{i=1}^{n} w_i x_i \geq T, \tag{1}$$

where $\sum$ denotes the arithmetic sum.

A graphical representation of a threshold logic function is shown in Fig. 6a. Fig. 6 shows that by selecting appropriate parameters [**W**;T], in equation 1, different Boolean functions and predicate operations can be implemented. Many analog and digital implementations of threshold functions exist in literature. A mixed signal implementation with a digital output is called a configurable neuron (CN) [16]. Each cluster of the NPEs consists of 5 CNs to perform 5-bit logic, addition, and comparison operations.

The advantages of a single CN over the CMOS equivalent are demonstrated in [16]. For instance, *a 5-input CN in 40nm, which is about the size of a high drive strength D-FF* can replace a complex function such as a 3-out-of-5 majority function $f(x_1, x_2, x_3, x_4, x_5) = x_1x_2x_3 + x_1x_2x_4 + x_1x_3x_4 + x_2x_3x_4 + x_1x_2x_5 + x_1x_3x_5 + x_2x_3x_5 + x_1x_4x_5 + x_2x_4x_5 + x_3x_4x_5$ *and* the D-FF that f drives. Many other functions that would normally require several levels of logic can be replaced by a single CN. Overall, at the individual cell level, [16] shows that a 5-input CN results in improvements in area, power, and delay of [80%, 60%, 40%] respectively, over the performance optimized, functionally equivalent CMOS circuit.

4 EXPERIMENTS AND RESULTS

4.1 Design and evaluation methodology

The proposed PACT-3D architecture consists of two major components: the NPEs and the High Bandwidth Memory (HBM). The energy and performance models of NPEs are obtained from [16]. The area and power numbers of the NPEs at 500 MHz frequency are shown in Table 1 and used in the custom-designed behavioral level simulator written in concert with DRAMPower [2]. This simulator takes workload and HBM specifications as the input to characterize the latency and energy consumption of the (NPEs + HBM).

We present a comparison against the state-of-the-art near-memory architecture HAIMA [5]. HAIMA is a hybrid architecture that adds digital logic to the SRAM and DRAM to perform computation inside the memory. An SRAM processing unit (SPU) performs 256 parallel 8-bit MAC operations in one SRAM-CIM block. Inside the DRAM, HAIMA uses a bank processing unit (BPU) with each DRAM bank to compute a dot product between two vectors of 32 8-bit elements each. A BPU consists of digital multipliers, adders, and MUXs to perform the computation. Table 1 shows the power and area overhead of HAIMA's compute elements. Table 2 shows the hardware configuration of PACT-3D and the baseline HAIMA [5].

Note: We also use our behavior level simulator to simulate HAIMA as well based on the hardware description, power, and latency of different units provided in the paper [5].

Table 1: Power and area of logic added in PACT-3D **and HAIMA [5].**

Component	Power (mW)	Area (mm^2)
BPU of HAIMA	148.0	0.014
32-to-1 MUX Of HAIMA	2,990	0.033
SPU of HAIMA	1,554.3	0.125
NPE [16]	0.207	0.004

We also provide a comparison against another DRAM-based near-memory architecture, TransPIM [18] and NVIDIA Tesla V100 GPU based on data provided in [5]. TransPIM adds digital logic to HBM DRAM to accelerate the Transformer execution. For a meaningful comparison, the same HBM configuration as described in Table 2 is used for PACT-3D, HAIMA, and TransPIM. Additionally, for the simulation framework, HBM to host CPU bandwidth is set to 256GB/s and the rate of energy consumption is 35GB/J [1].

4.2 Workloads

This paper evaluates on two Transformer-based architectures, BERT [4] and BART [10] used for various NLP tasks. BERT is based on encoder architecture with two configurations, BERT base (12 Encoder layers) and BERT large (24 encoder layers). On the other hand, BART consists of both encoder and decoder layers. Similar to BERT, it also has a BART small (6 encoders and 6 decoders) and BART large (12 encoders and 12 decoders) configuration. An input sequence of 512 to 4096 tokens is common in various NLP tasks and therefore, used for evaluation.

4.3 Results and discussion

The evaluation of PACT-3D is carried out for 8-bit precision of the workload, same as the HAIMA and TransPIM. The results are presented for matrix multiplication operations in the attention and feed-forward layers of the Transformers.

Latency and Energy Efficiency: Fig. 7 shows the speedup of TransPIM, HAIMA, and PACT-3D architectures over GPU for different workloads. On average PACT-3D achieves 1.7×, 2.4×, and 28.6× speedup over the HAIMA, TransPIM, and GPU respectively.

[1] W.L.O.G. the weights w_i and threshold T can be integers [12].

Table 2: Configuration of the platforms used.

Architecture	HAIMA	PACT-3D
Processing Elements	1 BPU/Bank in DRAM, 1 SPU/SRAM-CIM, 8-core Host CPU	1024 NPEs/Bank in DRAM, 1 RISC-V/Channel in HBM logic layer
SRAM-CIM	32 units, 256 rows, 256 x 8 bits/row	Not Used
DRAM (HBM)	1 Rank, 4 Bank groups (BG), 4 Banks/BG, 32768 rows/Bank, 1024 x 8 bits/row t_{RAS} = 29, t_{RP} = 14, t_{RCD} = 16, t_{CL}, t_{CCD} = 2, t_{WR} = 16, t_{RC} = 45, t_{RRD} = 2	

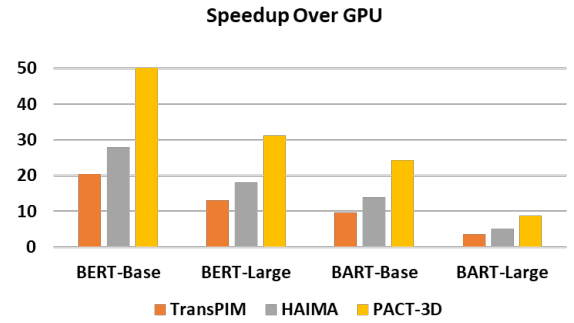

Figure 7: Speedup of PACT-3D as compared to the baseline architectures. PACT-3D has 1.7×, 2.4×, and 28.6×, speed-up on an average over HAIMA, TransPIM and GPU respectively.

Due to the small size of the NPEs, PACT-3D utilizes the maximum available parallelism inside the DRAM by interfacing the NPEs with BLSA (row buffer) outputs. This configuration not only increases the compute parallelism but also reduces the high latency and energy-consuming DRAM operations of activating the row (ACT), reading (RD), and write-back (WR) to the DRAM banks. As the entire computation takes place inside the DRAM as opposed to HAIMA, the data transaction on the CPU-HBM memory channel is avoided. This leads to a substantial reduction in latency and energy consumption in PACT-3D. Furthermore, the use of extremely low-power NPEs, computation of the entire workload is performed within the HBM, and substantial reduction in the latency leads to a highly energy-efficient execution of the Transformers in PACT-3D. Furthermore, on average over all workloads, PACT-3D has **18.7×** higher energy efficiency than HAIMA.

Effect of Scaling the workload: Table 3 shows the speedup and energy efficiency improvement of PACT-3D over HAIMA for increasing the length of the input sequence. As the hardware configuration and the capacity of the HBM are kept constant for both HAIMA and PACT-3D, a nearly constant improvement in the speed-up and energy efficiency is observed. This shows a complete utilization of the compute resources and bandwidth in PACT-3D. If the workload increases, the HAIMA architecture's energy efficiency degrades due to two main reasons: (1) limited SRAM size restricts computation resources, increasing latency and energy consumption, and (2) more data transactions on the host-HBM memory channel drastically increase energy consumption and latency.

Table 3: Speedup and Energy Efficiency of PACT-3D normalized to HAIMA for different workloads.

	BERT-Large (HAIMA = 1)		BART-Large (HAIMA = 1)	
Seq. Length	**Speedup**	**Energy Eff.**	**Speedup**	**Energy Eff.**
512	1.75	19.20	1.72	18.65
1024	1.74	18.43	1.71	17.98
2048	1.74	17.89	1.71	17.42
4096	1.74	17.58	1.71	17.10

5 CONCLUSION

This paper presents PACT-3D a near-memory architecture that integrates novel neuron processing elements (NPEs) to utilize the maximum available parallelism inside a DRAM and mitigate the memory bottleneck of executing large-scale deep neural network models. Low-power and low-area NPEs enable a large number of parallel compute units inside DRAM to perform the execution of the Transformer model without transferring data over the host-DRAM memory channel. This paper evaluates encoder-only (BERT) and encoder-decoder (BART) Transformer architectures with varying sequence lengths. PACT-3D achieves about 1.7× lower latency and 18.7× higher energy efficiency than the state-of-the-art near-memory architecture for Transformers and demonstrates the ability of PACT-3D to scale larger workloads with high energy efficiency.

ACKNOWLEDGMENTS

This work was supported in part by the NSF I/UCRC for IDEAS and from NSF grants #2231620 and #2008244. The authors also gratefully acknowledge Qualcomm Inc. for the support.

REFERENCES

[1] AMD. 2015. High-Bandwidth Memory (HBM). https://www.amd.com/system/files/documents/high-bandwidth-memory-hbm.pdf.
[2] K. Chandrasekar et al. 2024. DRAMPower: Open-source DRAM Power and Energy Estimation Tool,. http://www.drampower.info/.
[3] Q. Deng et al. 2018. DrAcc: a DRAM based accelerator for accurate CNN inference. In *DAC'18*.
[4] J. Devlin et al. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805
[5] Y. Ding et al. 2023. HAIMA: A Hybrid SRAM and DRAM Accelerator-in-Memory Architecture for Transformer. In *DAC*.
[6] J. Gomez-Luna et al. 2022. Benchmarking a New Paradigm: Experimental Analysis and Characterization of a Real Processing-in-Memory System. *IEEE Access* (2022).
[7] N. Hajinazar et al. 2021. SIMDRAM: A Framework for Bit-Serial SIMD Processing using DRAM. In *ASPLOS'21*.
[8] H. Kim et al. 2021. Colonnade: A Reconfigurable SRAM-Based Digital Bit-Serial Compute-In-Memory Macro for Processing Neural Networks. *JSSC* 56, 7 (2021).
[9] J.H. Kim et al. 2021. Aquabolt-XL: Samsung HBM2-PIM with in-memory Processing for ML Accelerators and Beyond. In *2021 HCS*.
[10] M. Lewis et al. 2019. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *CoRR* abs/1910.13461 (2019). arXiv:1910.13461
[11] S. Li et al. 2017. DRISA: a DRAM-based Reconfigurable In-Situ Accelerator. In *IEEE/ACM MICRO'17*.
[12] S. Muroga. 1971. *Threshold logic and its applications.* Wiley-Interscience, NY.
[13] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]
[14] V. Seshadri et al. 2017. Ambit: in-memory accelerator for bulk bitwise operations using commodity DRAM technology. In *MICRO'17*.
[15] S. Sridharan et al. 2023. X-Former: In-Memory Acceleration of Transformers. *TVLSI* (2023).
[16] A. Wagle et al. 2024. An ASIC Accelerator for QNN With Variable Precision and Tunable Energy-Efficiency. *IEEE TCAD* (2024).
[17] S. Yin et al. 2020. Vesti: Energy-Efficient In-Memory Computing Accelerator for Deep Neural Networks. *TVLSI'20* (2020).
[18] M. Zhou et al. 2022. TransPIM: A Memory-based Acceleration via Software-Hardware Co-Design for Transformer. In *HPCA*.