PHILOSOPHICAL TRANSACTIONS A

royalsocietypublishing.org/journal/rsta

Research

Article submitted to journal

**Subject Areas:**
Hardware Acceleration, Edge Computing, Natural Language Processing

**Keywords:**
Language Model Accelerator, BERT, Edge Devices, Near-Data Processing, AFPOS

**Author for correspondence:**
Joycee Mekie
e-mail: joycee@iitgn.ac.in

# AxLaM: Energy-Efficient **Acc**elerator Design for **La**nguage **M**odels for Edge Computing

Tom Glint[1], Bhumika Mittal[2], Santripta Sharma[2], Abdul Qadir Ronak[3], Abhinav Goud[3], Neerja Kasture[3], Zaqi Momin[3], Aravind Krishna[3] and Joycee Mekie[3]

[1]Forschungszentrum Jülich
[2]Ashoka University
[3]Indian Institute of Technology Gandhinagar

Modern language models like BERT have revolutionized natural language processing tasks but are computationally intensive, limiting their deployment on edge devices. This paper presents an energy-efficient accelerator design tailored for encoder-based language models, enabling their integration into mobile and edge computing environments. A data-flow aware hardware accelerator design for language models inspired by SIMBA, use of Approximate Fixed-Point POSIT (AFPOS) based multipliers, and use of high bandwidth memory achieves significant improvements in computational efficiency, power consumption, area, and latency compared to the hardware-realized scalable accelerator Simba. Compared to SIMBA, AxLaM achieves 9$\times$ energy reduction, 58% area reduction and 1.2$\times$ improved latency, making it suitable for deployment in edge devices. The energy efficiency of AxLaN is 1.8 TOPS/W, 65% higher than FACT which needs pre-processing of language model before implementing it on the hardware.

## 1. Introduction

The rapid advancements in natural language processing (NLP) have led to the development of powerful language models like GPT and BERT [1,2], which excel in tasks such as text classification, question answering, and language translation. However, the computational intensity and memory requirements of these models pose significant challenges for deployment on edge

THE ROYAL SOCIETY PUBLISHING

Edge deployment of language models is crucial for applications requiring real-time processing and data privacy, such as mobile assistants, IoT devices, and autonomous systems. Offloading computations to the cloud introduces latency and potential security risks due to data transmission [4]. Therefore, there is a pressing need for specialized hardware accelerators that can efficiently run these models within the stringent power and area budgets of edge devices, compared to general-purpose computing environments.

In this paper, we present an energy-efficient accelerator design, for battery-powered edge devices, specifically optimized for encoder-based models like BERT. Our main contributions are as follows:
**1)** We, for the first time, show that a highly quantized and approximated variant of POSIT [5] can incur a negligible loss of accuracy compared to Float-Point32 (FP32) encoding scheme for BERT. The resultant Multiply-Accumulate (MAC) Unit require $\sim$0.3 pJ for a MAC operation in 65 nm ($\sim$69$\times$ better than FP32). **2)** We present a fine-tuning method that allows the direct use of the pre-trained BERT to use the number system without structural changes to the neural network. To the best of our knowledge, this is the first time quantization without accuracy loss has been achieved without structural changes and retraining. **3)** Based on the hardware software-co-design approach, we introduce an optimized Processing Element (PE) architecture that leverages Near-Data Processing on HBM [6] and the Approximate Fixed POSIT (AFPOS) number system to significantly reduce power consumption and area, by leveraging, for the first time, the three dimensional spatial data reuse opportunity in BERT. **4)** We provide a comprehensive evaluation of our design against the hardware-realized accelerator Simba, demonstrating improvements in computational efficiency, power consumption, area, and latency, while maintaining negligible accuracy loss on the CoLA dataset using BERT large. Furthermore, we compare our work with eight recent accelerators for LLMs (that require structural modification to BERT) and demonstrate superior area and energy efficiency.

The rest of the paper is organized as follows: Section 2 reviews related work. Section 3 details our accelerator design approach. Section 4 describes the methodology and evaluation infrastructure. Section 5 presents the experimental results and analysis. Section 6 discusses the implications and limitations, and Section 7 concludes the paper.

## 2. Related Work

Hardware-software co-design has long been central to optimizing system performance by considering both hardware and software components. Early research, such as [3] and [7], emphasized automatic parallelization and self-programming systems to reduce memory operations and enhance data communication. In machine learning accelerators, especially for deep learning, several architectures have emerged. Simba [8] introduced a scalable multi-chip framework optimized for convolutional neural networks (CNNs), but it faces challenges with transformer models due to differing computational patterns.

Recent accelerators specifically target transformers. For instance, OPTIMUS [9] focuses on enhancing matrix multiplication efficiency, while $A^3$ [10] accelerates the attention mechanism through approximation techniques. NVIDIA's Transformer Engine [11] employs specialized tensor cores for faster inference, and SwiftTron [12] applies 8-bit quantization for resource-constrained environments. Further advancements include co-optimization and processing-in-memory (PIM) strategies. The H3D-Transformer [13] combines PIM with edge-specific optimizations, while ReTransformer [14] utilizes a ReRAM-based PIM architecture. TransPIM [15] focuses on memory-based acceleration through hardware-software co-design. More details on these comparisons are provided in Section 5, specifically in Subsection (c). *Current designs often require modifications to neural networks or lack comprehensive system-level implementations, particularly in managing off-chip memory access and supporting various integer formats. Moreover, these accelerators typically necessitate significant changes to the model structure to fully support transformers.*

Our work introduces an accelerator leveraging High Bandwidth Memory 3 (HBM3) and the Approximate Fixed-Point POSIT (AFPOS) number system, optimized for encoder-based models like BERT. This design offers a system-level solution for edge deployment, improving performance and efficiency without altering the model structure. Additionally, number system-aware fine-tuning of the BERT model ensures accuracy comparable to FP32. For hardware comparison, Simba is used due to its flexibility and broad acceptance, as recent works lack silicon implementation. Nonetheless, results are evaluated against recent accelerators, as discussed in Section 5, particularly in Subsection (c).

## 3. Accelerator Design Approach

### (a) Background - BERT: Use Cases, and Internal Structure

BERT [1] (Bidirectional Encoder Representations from Transformers) is a pre-trained language model introduced by Google in 2018. Unlike traditional CNN models, BERT uses bidirectional training, leveraging both preceding and succeeding contexts to predict words, which results in a deeper understanding of language. This approach has led to state-of-the-art performance across various NLP tasks. BERT excels in tasks such as text classification, named entity recognition, question answering, sentiment analysis, and language translation [16]. Its pre-trained nature allows for fine-tuning, making it versatile and well-suited for edge applications.

BERT is based on the Transformer architecture, consisting of an input embedding layer, multiple Transformer encoder blocks, and an output layer. Each encoder block includes a multi-head self-attention mechanism and a feed-forward neural network. Tokens are embedded with positional encodings and processed through Transformer blocks. The BERT-Large model has specific parameters: a model dimensionality ($d_{model}$) of 1024, 24 Transformer blocks ($N$), each with 16 attention heads ($H$). The key/query ($d_k$) and value ($d_v$) vectors have dimensions $\frac{d_{model}}{H} = 64$. Each attention head computes a 1024x1024 matrix, with concatenated outputs passed through a linear transformation to produce the final layer output:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots)W_O$$

where $W_O$ is the weight matrix that maps the concatenated output to the embedding size, $d_{model} = 1024$. After multi-head attention, the output undergoes a residual connection, layer normalization, and is processed through a feed-forward neural network, which first increases and then reduces dimensionality. The matrix operations are summarized in Table 1. These operations enhance computational efficiency and consistency. BERT's impact on NLP tasks has made it a leading model, with scalable configurations like BERT-Base and BERT-Large suitable for various environments. BERT's performance is recognized in *MLPerf* [16], making it the baseline workload for our design.

**Table 1.** Summary of unique matrix operations in BERT's encoder

| Operation | Matrix L Dimensions | Matrix R Dimensions | Result Dimensions |
|---|---|---|---|
| Q/K/V | $1024 \times 1024$ | $1024 \times 64$ | $1024 \times 64$ |
| Attention | $1024 \times 64$ | $64 \times 1024$ | $1024 \times 1024$ |
| Multi-Head | $1024 \times 1024$ | $1024 \times 1024$ | $1024 \times 1024$ |
| Bottleneck Expand | $1024 \times 1024$ | $1024 \times 4096$ | $1024 \times 4096$ |
| Bottleneck Contract | $1024 \times 4096$ | $4096 \times 1024$ | $1024 \times 1024$ |

### (b) Background - Spatial and temporal reuse in Hardware Accelerator

While efforts exist to accelerate BERT [17,18], no accelerators have been specifically optimized for edge scenarios, as noted in [2,3]. In edge computing, the *Simba* architecture is a leading solution for deep neural network inference [8,19], thanks to its advanced vector multipliers and accumulators that share input activations while outputting to multiple channels. Notably, *Simba* features a data-agnostic design, making it robust against data distribution variations.

At the core of *Simba*, illustrated in Fig. 1, is the Processing Element (PE), responsible for executing convolutional layers, fully-connected layers, and post-processing operations like bias

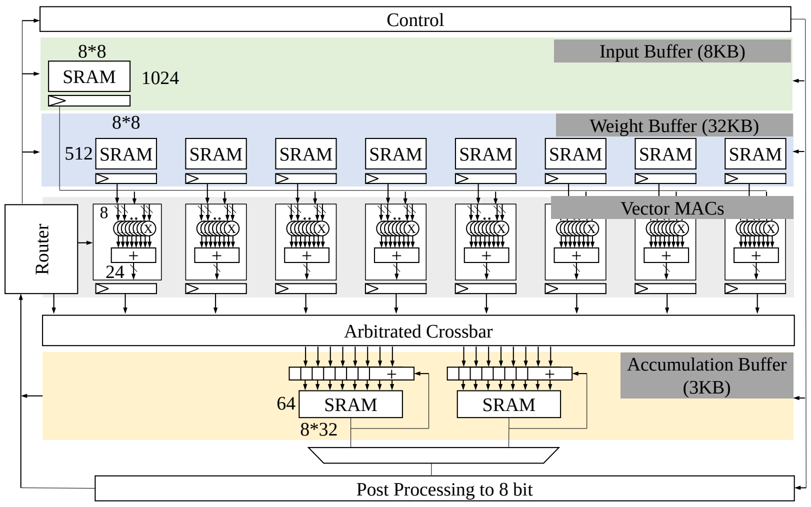

**Figure 1.** Simba's PE architecture [8]

addition, ReLU, and pooling. Each PE contains eight lanes, each using a distinct weight tensor to generate elements for a single output channel (K). An 8-bit precision vector MAC unit within each lane multiplies eight input elements from different channels (C) with eight weight elements, summing them to produce a single output. This approach is highly efficient, given typical channel counts (64 to 1024). The architecture is supported by local input activation, output activation, and accumulation SRAMs, which buffer the datapath. Energy efficiency is optimized by minimizing SRAM access: input activation SRAM is read every cycle, and its elements are forwarded across lanes. The weight SRAM, wider than the input activation SRAM, supplies distinct vectors to each lane and reuses weights across multiple inputs, reusing them P $\times$ Q times (where P and Q are output dimensions). Accumulation SRAM stores intermediate sums, conserving energy by accumulating the eight-wide vector from C channels. The output size, often exceeding the accumulation buffer's capacity, requires temporal tiling to generate portions of output activations sequentially. This buffer also supports cross-PE reductions when the weight kernel spans multiple PEs. Post-processing tasks like ReLU, bias addition, and pooling are performed after accumulation. The buffer is dual-banked, allowing simultaneous access by MACs, routers, and post-processing units, with an arbitration crossbar to resolve bank conflicts.

## (c) Motivation for a New Architecture Design

The growing complexity and computational demands of Natural Language Processing, particularly models like BERT, have exposed the limitations of existing hardware accelerators for ML. Accelerators for BERT, which are primarily based on FPGA platforms [18] or the more recent ASIC designs [2], often require significant modifications to the neural network architecture to achieve operational efficiency. Additionally, such modifications, including extreme quantization or mixed-signal processing techniques, frequently result in a notable loss of accuracy [2,13,20]. While the Simba architecture, due to silicon realization and data-agnostic operation, has been recognized as a leading solution for deep neural network inference, its efficacy diminishes when confronted with the distinct computational demands of BERT's matrix operations. The various matrix multiplications, particularly those involving the QKV (Query, Key, Value) operations, impose a substantial power burden, primarily due to the significant DRAM accesses required, as illustrated in Fig. 2. These multiplications, as summarized in Table 1, highlight the computational challenges that Simba faces when deployed for models like BERT.

The QKV operations, though not the largest in magnitude, play a critical role in BERT's performance due to relatively frequent computation (16 times each per encoder block). The power consumption associated with these operations is exacerbated by the limited data reuse

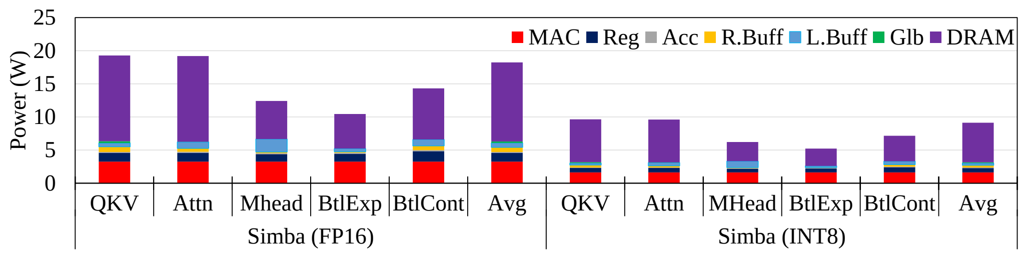

**Figure 2.** Simba's (with FP16 MAC for accuracy preservation) system power for processing matrices in BERT's encoder

inherent in their matrix multiplications. In energy-efficient computing, small buffer sizes are often preferred to optimize the power consumption of such operations. However, as the buffer sizes increase, the energy required for data access also rises, leading to further inefficiencies. Conversely, operations such as Bottleneck Contraction (BtlCont), Bottleneck Expansion (BtlExp), and Multi-Head Attention (MHead) exhibit a more pronounced data reuse pattern. Each output word in these operations depends on approximately 1024 input words, allowing for a significant reduction in access counts as buffer sizes grow. This characteristic makes large buffer sizes essential to fully exploit data reuse, thereby improving power efficiency in these layers.

Despite these insights, the Simba architecture's current design is not well-suited to meet the diverse computational requirements of BERT's matrix operations. Simba's average power consumption, 9 and 19 watts, respectively for INT8 and FP16 pipelines, is far beyond the acceptable range for edge and portable applications, which typically impose stringent power constraints, often below 5 watts [21]. Therefore, while Simba excels in certain scenarios, its inability to efficiently handle the unique demands of BERT's matrix multiplications underscores the need for a new architecture that can better address these challenges.

## (d) Design Consideration: Challenges to be Addressed

Efficient hardware accelerators for neural networks like BERT must address several key challenges: high DRAM access count, on-chip buffer energy consumption, and on-chip compute energy efficiency.

**(i)** Reducing High DRAM Access Count and Energy Consumption: High DRAM access significantly impacts overall energy consumption. To mitigate this, increasing the on-chip buffer size can reduce the frequency of DRAM accesses by enhancing data reuse and minimizing energy-intensive memory transactions. This approach leverages larger buffers to store more data locally, thereby reducing the need for frequent access to external memory.

**(ii)** Reducing Energy at the On-Chip Buffer Level: While larger on-chip buffers can reduce DRAM access, they can also lead to increased energy consumption within the chip. This challenge can be managed by optimizing memory hierarchies with multi-level buffering strategies and exploiting opportunities for temporal and data sharing within the application. Effective buffer management ensures that the energy benefits of reduced DRAM access are not offset by increased internal energy costs.

**(iii)** The energy required for on-chip computations, particularly multiply-accumulate (MAC) operations, is a significant consideration for BERT's intensive computations. Reducing this energy footprint can be achieved by employing low-power arithmetic units with reduced precision and implementing operand sharing techniques to maximize the reuse of input data. These strategies help lower the energy consumption associated with each computation, making the overall processing more efficient.

## (e) Design Consideration: On-Chip Buffer Size and Memory Accesses

The unique matrix multiplications within BERT's encoder exhibit distinct DRAM access patterns that significantly impact the power consumption associated with different buffer sizes. The primary objective is to select a buffer size that minimizes the overall system energy consumption (considering on-chip data reuse) during data accesses and computational processes. As shown in

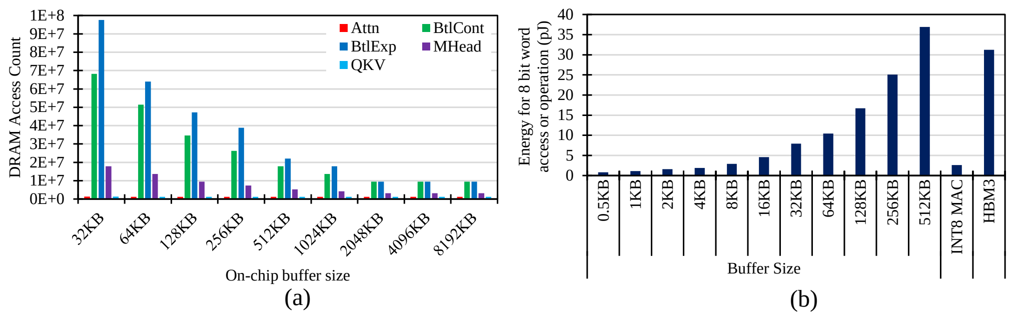

**Figure 3.** (a) DRAM access count vs on-chip buffer size while processing BERT. (b) Access energy of buffers against size, compute, and HBM.

Fig. 3.a, the DRAM access count varies with increasing total buffer size, offering valuable insights into data reuse and how frequently each matrix multiplication operation requires access to main memory. Simultaneously, Fig. 3.b illustrates the energy per byte access data for different buffer sizes, highlighting the energy implications of choosing specific buffer configurations. Balancing the frequency of DRAM accesses against the energy overhead of larger buffers is critical in determining the most energy-efficient buffer size. To achieve optimal energy efficiency, both the frequency of DRAM accesses for each matrix multiplication and the energy cost associated with each access—given a specific buffer size—must be considered. Additionally, the repetitive nature of BERT's matrix operations within the encoder should be taken into account, as certain operations recur more frequently, potentially skewing the overall energy profile. For instance, while QKV operations may require fewer DRAM accesses in some buffer configurations, their central role in the encoder and frequent occurrence contribute significantly to the total energy consumption.

Comparing the DRAM access count data with buffer access energy, it becomes evident that smaller buffer sizes generally exhibit lower energy costs per access. However, as buffer sizes increase, while DRAM access counts typically decrease, the energy cost per access rises. Considering all available metrics, an intermediate buffer size appears to strike the best balance between reducing DRAM accesses and maintaining manageable energy costs per access. Analysis suggests that buffer sizes in the range of 64KB to 128KB may be appropriate, as indicated by the trends in the provided data. However, further refinement of this range is necessary, particularly by considering the specific frequency of each matrix operation in BERT's encoder.

## (f) Opportunity: HBM logic layer for bandwidth and low access energy

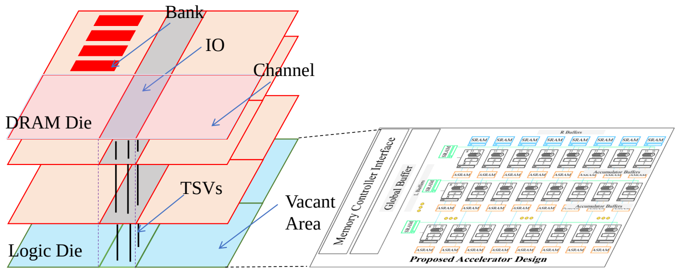

**Figure 4.** HBM3 structure showing vacant space for accelerator placement

Traditional DDR DRAM (such as DDR4 DRAM) is insufficient for handling the high-frequency memory accesses required by BERT's encoders, where DRAM access energy is a major concern [22]. High Bandwidth Memory 3 (HBM3) [6] offers a tenfold reduction in access energy per bit (4.2 pJ vs. 46 pJ), making it a superior choice for BERT's frequent matrix multiplications, significantly enhancing accelerator efficiency. HBM3's design is well-suited for BERT-like models, allowing direct integration of accelerator designs within its logic layer (up to 30 mm$^2$ and 8.5W Power [23]), as supported by recent manufacturer initiatives [24]. This integration minimizes data movement, reducing latencies and further cutting energy consumption, crucial for the high data reuse in BERT's encoders. HBM3's structure stacks DRAM dies atop a logic layer, connected via Through-Silicon Vias (TSVs), achieving a compact memory footprint. It vastly outperforms DDR4 in bandwidth, offering around 512 GB/s per stack compared to DDR4's 25.6 GB/s, making HBM3 ideal for high-throughput, rapid-access applications [6]. However, this presents three main challenges: ensuring balanced access to all HBM memory channels, managing design area constraints, and addressing thermal and power limitations.

## (g) Addressing the High MAC Energy

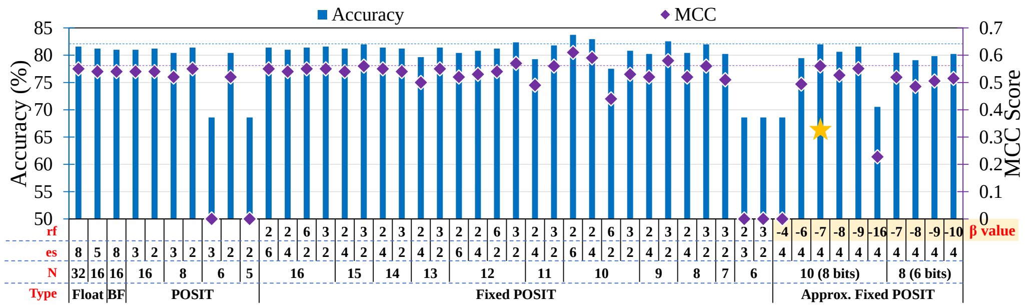

**Figure 5.** Model Accuracy and MCC Score of BERT for various encoding schemes, Fine-tuned on CoLA Dataset. Approx. Fixed POSIT value = $(-1)^{\text{sign}} \times 2^{\beta} \times 2^{\text{exp}} \times (1.\text{mantissa})$

**Table 2.** Minimum Latency of design, and Power and Area of Multipliers against different data formats.

| Type | Bits | Config. | Latency (ps) | Area ($\mu m^2$) | Energy (pJ) |
|---|---|---|---|---|---|
| FP32 | 32 | | 1299 | 13830.84 | 22.50 |
| FP16 | 16 | | 979 | 3095.28 | 4.55 |
| BF16 | 16 | | 832 | 2148.12 | 2.75 |
| POSIT | 16 | 16,3 | 1906 | 8634.60 | 13.45 |
| POSIT | 16 | 16,2 | 1925 | 9327.60 | 14.84 |
| POSIT | 8 | 8,3 | 1808 | 1062.72 | 1.58 |
| POSIT | 8 | 8,2 | 1224 | 2508.48 | 5.08 |
| POSIT | 6 | 6,3 | 775 | 965.88 | 2.90 |
| FPOS | 16 | 16,6,2 | 1225 | 597.96 | 0.61 |
| FPOS | 16 | 16,4,2 | 1114 | 510.84 | 0.60 |
| FPOS | 16 | 16,2,6 | 826 | 1723.68 | 1.76 |
| FPOS | 16 | 16,2,3 | 814 | 851.40 | 1.40 |
| FPOS | 12 | 12,6,2 | 797 | 582.48 | 0.89 |
| FPOS | 12 | 12,4,2 | 797 | 502.92 | 0.92 |
| FPOS | 12 | 12,2,6 | 799 | 1459.80 | 1.26 |
| FPOS | 12 | 12,2,3 | 796 | 681.84 | 0.86 |
| FPOS | 15 | 15,4,2 | 797 | 576.00 | 1.01 |
| FPOS | 15 | 15,2,3 | 675 | 1112.04 | 1.63 |

| Type | Bits | Config. | Latency (ps) | Area ($\mu m^2$) | Energy (pJ) |
|---|---|---|---|---|---|
| FPOS | 14 | 14,4,2 | 558 | 1003.68 | 1.68 |
| FPOS | 14 | 14,2,3 | 656 | 991.08 | 1.12 |
| FPOS | 13 | 13,4,2 | 528 | 1047.24 | 1.82 |
| FPOS | 13 | 13,2,3 | 656 | 982.08 | 1.62 |
| FPOS | 11 | 11,4,2 | 533 | 914.04 | 2.01 |
| FPOS | 11 | 11,2,3 | 680 | 904.32 | 1.90 |
| FPOS | 10 | 10,6,2 | 1004 | 480.24 | 0.51 |
| FPOS | 10 | 10,4,2 | 999 | 409.32 | 0.48 |
| FPOS | 10 | 10,2,6 | 827 | 1578.24 | 1.11 |
| FPOS | 10 | 10,2,3 | 636 | 919.08 | 1.50 |
| FPOS | 9 | 9,4,2 | 532 | 898.92 | 1.73 |
| FPOS | 9 | 9,2,3 | 640 | 774.36 | 1.55 |
| FPOS | 8 | 8,4,2 | 550 | 524.88 | 1.18 |
| FPOS | 8 | 8,2,3 | 648 | 790.20 | 0.74 |
| FPOS | 7 | 7,2,3 | 629 | 709.92 | 1.63 |
| APOS | 8 | 10,4 | 465 | 766.08 | 0.51 |
| APOS | 6 | 10,4 | 435 | 628.56 | 0.26 |

**Table 3.** Synthesis Result: Area, Energy, and Latency Values of AFPOS Vector MAC (16 Multipliers and Adder Tree)

| Unit | Constraint | Area ($\mu m^2$) | Timing (ps) | Power (with stimulus) (W) | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | Switching | Internal | Dynamic | Leakage | Total |
| AFPOS (6,2) | max Freq | 2.38E+03 | 5.75E+02 | 2.11E-05 | 1.20E-04 | 1.41E-04 | 2.45E-07 | 1.41E-04 |
| AFPOS (8,4) | max Freq | 8.53E+03 | 9.56E+02 | 1.09E-03 | 1.45E-03 | 2.54E-03 | 8.81E-07 | 2.54E-03 |
| AFPOS (10,4) | max Freq | 1.55E+04 | 1.96E+03 | 1.22E-03 | 1.43E-03 | 2.65E-03 | 1.60E-06 | 2.65E-03 |

In accelerators for complex language models like BERT, MAC operations are the second highest contributors to power consumption after DRAM access, as shown in Fig. 2. Traditional 8-bit integer operations, while efficient, do not meet the precision demands of BERT, necessitating a solution that combines floating-point precision with reduced power overheads. Recent advancements in multiplier design, such as the POSIT number system introduced by Gustafson and Yonemoto [5], offer a promising alternative. POSITs, defined by parameters $(N, es)$, provide dynamic flexibility, allowing efficient space usage by encoding exponent and fraction parts only when necessary. However, the increased area and power requirements of POSITs present challenges. To mitigate these, the fixed-posit representation was developed [25], which standardizes the regime length $(rf)$ alongside $N$ and $es$. The Approximate Fixed POSIT Multiplier

(AFPOS) further refines this approach, optimizing for area, energy, and latency while requiring 25% less bit storage than fixed-POSITs [22]. Our experimentation has shown that fixing the $2^{\beta}$ term for AFPOS (with 1-bit sign, 4-bit exponent, and 3-bit mantissa) to $2^{-7}$ results in negligible loss of accuracy compared to FP32, as shown in Fig. 5. *For the rest of the paper, AFPOS refers to this ($N = 10, es = 4, \beta = -7$) configuration*. Tables 2 shows the Latency, Area and Energy of these multipliers. We have highlighted the designs with the highest quantization yet having similar accuracy and MCC score as FP32. Compared to a quantized 16-bit floating-point multiplier, AFPOS reduces area by 4.04×, energy by 8.9×, and latency by 2.1×. We use a 16-wide Vector MAC, with area, latency, and power as shown in Table 3, for efficient spatial reduction.

## (h) Processing Element Design Details and Innovations

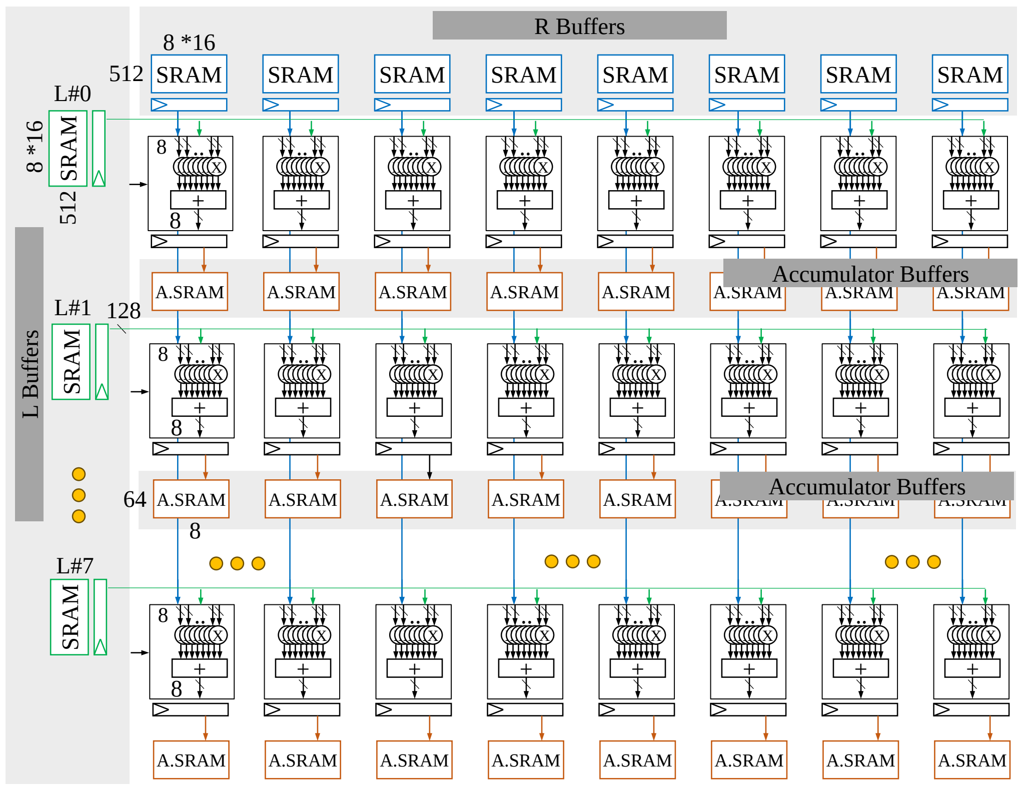

**Figure 6.** Unified PE Design of the Proposed Accelerator

BERT's computational architecture, particularly its vector multipliers, poses unique challenges distinct from frameworks like Simba. In Simba, shared input activations among a set of vector multipliers, along with exclusive weights, allow weights to be read once and reused multiple times. This resource optimization, however, is not inherent in BERT. When dealing with matrices $L$ and $R$ (as seen in Table 1), if the elements of $R$ are temporarily stationary, those of $L$ require constant cycling, and vice versa. This dynamic underscores the need to efficiently amortize read energy costs, leading to a redesigned Vector MAC and PE, as shown in Fig. 6. Instead of employing 16 separate PEs, we propose a unified PE that optimizes shared resource utilization while maintaining an acceptable fan-out. For each read of an element in $L$, the element contributes to the partial product of 8 distinct columns, while efficient reads from $R$ enable participation in 8 different rows of the output matrix. A simplified representation of this concept is illustrated in Fig. 7, where words read together are annotated with alphanumeric labels and sources are color-coded equally compared to Fig. 6. Further, within a column or row, sixteen elements are multiplied together and accumulated using the Vector MAC and written to the Accumulator Buffer (A.SRAM). We choose this configuration, as the baseline Simba instance, in efficiency mode, offers 1 TOPS throughput (which offers sub-one second inference of BERT-Large). Our goal is to achieve similar throughput with our Vector MAC configuration, which the parallel adder tree. The operational strategy dictates that either the $L$ or $R$ matrix elements must be refreshed upon completing each operand set. To balance energy efficiency and performance, our design incorporates a 8 KB buffer for all $L$ buffers and an equivalent capacity for the $R$ buffers. The Accumulation buffer (A.SRAM) is designed to cache partial products, enabling efficient summation.

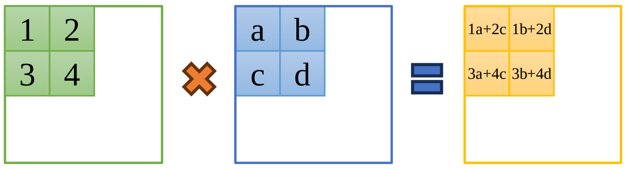

**Figure 7.** Simplified example showing operations that can be performed in a single cycle in parallel

### (i) Design Parameter Justification

The Unified Processing Element (PE) is directly interfaced with all 16 channels of HBM through memory controllers, occupying only 2.5 mm$^2$ of the available 30 mm$^2$ space, leaving ample room for additional designs, as shown in Fig. 4. The vector MAC operates at 500 MHz (as detailed in Table 3) to optimize energy efficiency. The choice of a 16-wide vector MAC is based on achieving the desired throughput of 1 TOPS, equivalent to Simba's efficiency mode. Further widening of the vector MAC would introduce significant combinational delays, while increasing the operating frequency would substantially raise area and energy consumption. This throughput also meets the sub-1 second response time expected for edge devices. Simulations revealed that increasing the total number of multipliers led to diminishing returns due to memory bandwidth limitations. The 8KB buffer size was selected to strike a balance between on-logic die storage and energy consumption. While larger buffers reduce DRAM accesses, they also increase access energy per byte, as shown in Figure 3.b. Our analysis determined that 8KB of local buffer per row and column offers an optimal balance for handling BERT's workload efficiently.

## 4. Methodology

### (a) Evaluation Infrastructure

At the software level: We model BERT using a PyTorch-based GPU accelerated framework, modified to support different number representations, including AFPOS. In the framework, the flow of data is altered such that the data is encoded into the specified encoding scheme (like BFloat, POSIT, or AFPOS) before multiplication. This allows the framework to use IEEE 754 Float based pre-trained BERT model without alteration. After fine-tuning the model with CoLA dataset, we measure the accuracy and MCC scores against the full-precision model (Result in Fig. 5). We use this software framework for reproducibility and for exact computation and score calculation.

For hardware modeling: We use Cadence Genus to synthesize the MAC units (of all encoding schemes) at a **65nm node**, ensuring timing constraints are met at 500MHz frequency. SRAM buffers are modeled using CACTI [26]. System-level modeling and optimal data search (for least latency followed by least energy) is performed with Timeloop, Accelergy, and Alladin framework [7,27], defining both Simba and our proposed architecture based on synthesized values. For Simba, the external DRAM memory used is DDR4 DRAM and the hardware model is identical to the hardware realized version of Simba [8]. Our proposed work is modeled as a design instantiated on the logic layer of the HBM3 memory. We use this hardware model framework as it has been validated against real hardware and allows for flexible extensions.

### (b) Workload Description

We focus on the unique matrix operations within BERT's encoder, as summarized in Table 1. These operations include the QKV computations, attention mechanisms, multi-head concatenations, and feed-forward networks. The network level performance of any variation of BERT large can be extrapolated from these matrices.

## 5. Results

### (a) Latency and Energy

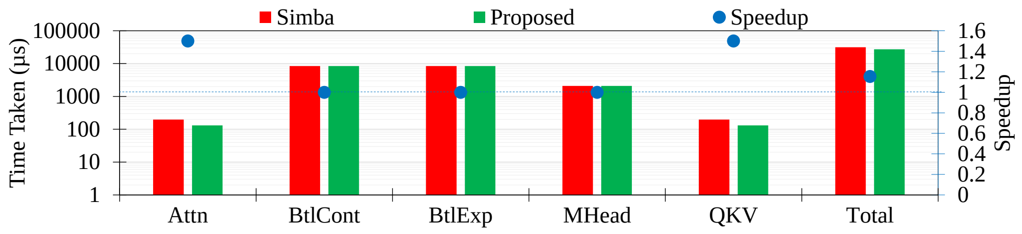

**Figure 8.** Performance Comparison

Fig. 8 compares the latency during the processing of each unique matrix by Simba and the proposed design. Although both designs feature an identical RAW throughput of 1 TOPS, Simba's reduced data reuse, particularly in the QKV and attention matrices, limits its ability to fully utilize all MAC units due to bandwidth constraints. This inefficiency results in a $1.2\times$ speedup for the proposed design. Each encoder is processed within 27 ms, enabling the complete BERT large model to be processed in 0.6 seconds, well within the acceptable latency threshold for handheld/edge devices performing natural language tasks, even when processing up to 1000 tokens.

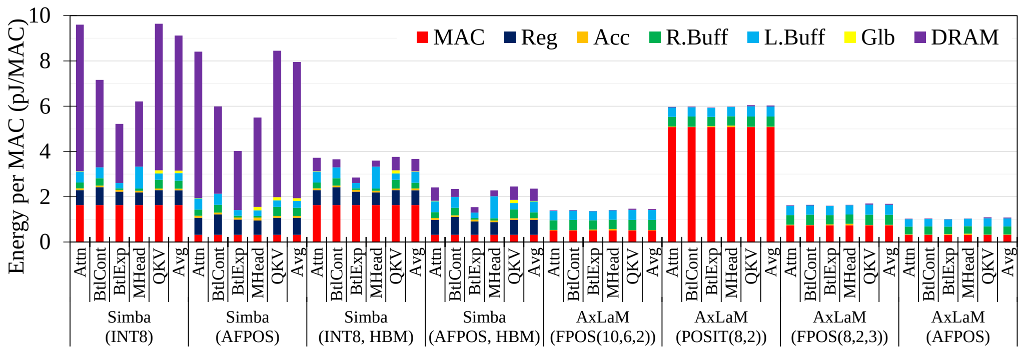

**Figure 9.** Energy Comparison

Fig. 9 shows the amortized energy cost of a MAC operation at the system level, comparing Simba, a variant of Simba (Simba with AFPOS multipliers, Simba implemented in HBM3's logic layer, Simba implemented with AFPOS multiplier in HBM3's logic layer) and the proposed designs implemented with HBM3 and the candidate multipliers. The Avg of each implementation in Fig. 9 is the weighted average operations corresponding to an encoder in BERT-Large. The AFPOS-adapted version provides insight into the impact of changing the number system alone. The proposed design (AxLaM (AFPOS)) demonstrates the lowest energy per MAC operation with a $9\times$ reduction in energy consumption, primarily due to decreased DRAM access and multiplication energy compared to Simba. By only altering the multiplier in Simba, the energy difference narrows to $7\times$ for the proposed design. Additionally, the internal buffer energy is halved in the proposed design, which is attributed to the improved PE organization.

## (b) Area and Power Constraints Comparison: Simba vs. Proposed Design

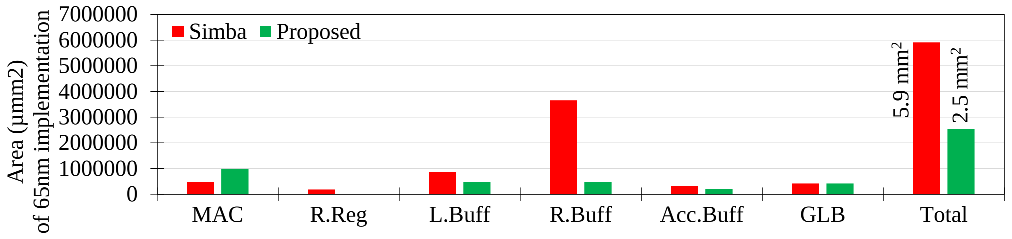

**Figure 10.** Simba vs. Proposed Design Area Comparison

Integrating the proposed design within the available area of the HBM3 memory's logic die offers substantial benefits in bandwidth and energy efficiency. Figure 10 illustrates the area distribution for the various components of the design. Compared to the Simba design, the proposed design not only adheres to the 36 mm$^2$ area constraint of HBM3 but also significantly reduces it to just 2.5 mm$^2$. This efficiency is primarily due to the streamlined Single PE design of the proposed system, as opposed to Simba's more expansive 16 PE architecture, resulting in an impressive 58% reduction in area.

In terms of power consumption, the proposed design shows marked improvement, consuming only 1.103 watts, which is a ninefold reduction compared to Simba's 9.34 watts (INT8) . This significant decrease ensures that the proposed design comfortably fits within HBM3's 8.5 W power limit [6]. Power consumption is calculated based on the energy and time required to process one full encoder of BERT.

## (c) Comparison with Other Recent Accelerators

**Table 4.** Comparison with recent accelerators

| Accelerator | Original Implementation Detail | Target Models | Area (mm2) | Throughput (TOPS) | Normalized E. Efficiency (TOPS/W) | Remarks |
|---|---|---|---|---|---|---|
| OPTIMUS [9] | ASIC Simulation, 28 nm synthesized circuits | Transformers | 5.2 | 0.5 | 0.127 | INT 16 |
| A3 [10] | ASIC Simulation, 40 nm synthesized circuits | BERT base | 2.08 | N/A | N/A | INT 8 |
| Transformer Engine [11] | Hardware Implementation inside 4 nm H100 GPU | ANY | N/A | 1978 | 0.04 | FP 8 |
| Swiftron [12] | 65 nm ASIC synthesized and simulated | RoBERTa | 273 | 8 | 0.236 | INT 8 |
| FACT [20] | 28 nm ASIC synthesized and simulated | BERT | 6.05 | 0.928 | 1.1* | INT 4 + 8 (HW-SW Co-Des) |
| ReTransformer [14] | 28 nm RRAM based CIM simulated | Custom | N/A | 0.081 | 0.2 | MIXED SIGNAL |
| TransPIM [15] | 22 nm DRAM based CIM simulated | RoBERTa | 53.15 | 0.734 | n/a | BIT-SERIAL |
| H3D [13] | 7 nm Digital and 22 nm Analog CIM simulated | BERT/GPT | 47.3 | 1.6 | 1.07 | MIXED SIGNAL |
| SIMBA [8] | 16 nm hardware implementation | Vector X Matrix | 6 | 0.3 to 4 | 0.039 to 0.31 | **INT8 (Silicon Realized)** |
| **AxLaM (OURS)** | **ASIC Simulation, 65 nm synthesized circuits** | **BERT** | **2.5** | **1** | **1.85** | **AFPOS** |

Table 4 presents a comparison between our proposed design, AxLaM, and recent accelerators such as OPTIMUS, A$^3$, Transformer Engine, SwiftTron, and others. Energy efficiency has been normalized to a 65 nm node based on voltage and capacitance scaling models. AxLaM distinguishes itself through the use of the Approximate Fixed POSIT System (AFPOS), achieving a throughput of 1 TOPS with a normalized energy efficiency of 1.3 TOPS/W. This positions AxLaM as one of the most energy-efficient designs, particularly for BERT, all within a compact 2.5 mm$^2$ area on a 65 nm process node.

While other accelerators, like the Transformer Engine, offer higher raw throughput, they fall short in terms of energy efficiency. AxLaM also demonstrates superior area efficiency, especially when compared to designs like Swiftron and H3D, which require significantly more space. AxLaM effectively balances throughput, energy efficiency, and area, making it a highly competitive solution for edge devices focused on natural language processing tasks.

In terms of energy efficiency, H3D [13] and FACT [20] are the closest competitors. However, FACT requires modifications to the language model. The actual benefit comes from Algorithm changes applied to BERT before porting it on the Hardware. On the other hand, H3D, being based on analog compute, suffers from high variability, and is not suitable for mass manufacturability. This further emphasizes AxLaM's robustness and suitability for practical deployment.

## 6. Discussion

Our results demonstrate that the proposed accelerator design significantly improves energy efficiency, area utilization, and latency for running language models like BERT on edge devices. By integrating with HBM3 and utilizing the AFPOS number system, we address the key challenges of power consumption and memory access bottlenecks.

The accuracy of our AFPOS BERT-Large model remains comparable to the full-precision model, ensuring that performance is not sacrificed for efficiency. The design scales well with input token sizes, making it suitable for a range of NLP applications.

However, our study is limited by the availability of detailed data for some recent accelerators, and actual hardware implementation is required to validate the simulation results fully. Future work includes implementing the design on an FPGA or ASIC platform and exploring further optimizations. The exploration and analysis of encoding schemes, including all variations of POSITs, though performed, have been left out for brevity.

## 7. Conclusion

We have presented an energy-efficient accelerator design tailored for encoder-based language models like BERT, suitable for deployment on mobile and edge devices. By leveraging Near-Data Processing, an optimized PE architecture, and the AFPOS number system, our design achieves significant improvements over the hardware realized state-of-the-art accelerator Simba. We also show the energy and area improvements over other LLM accelerators that require extensive DNN structural changes.

Our work contributes to enabling real-time, privacy-preserving NLP applications on edge devices, opening avenues for further research in efficient hardware designs for complex language models.

Acknowledgements. This work was supported by the Science and Engineering Research Board (SERB), Government of India, under the SERB-SUPRA grant SPR/2020/000450, the Ministry of Electronics and Information Technology (MEITY) under the Chips to Startup (C2S) program, and the Semiconductor Research Corporation (SRC) through contract 2020-IR-2980. Additional support was provided by the Federal Ministry of Education and Research (BMBF, Germany) under the NEUROTEC II project (Project No. 16ME0398K) and the NeuroSys project (Grant No. 03ZU1106AB).

## References

1. Devlin J, Chang MW, Lee K, Toutanova K. 2018 Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
2. Koilia N, Kachris C. 2024 Hardware Acceleration of LLMs: A comprehensive survey and comparison. *arXiv preprint arXiv:2409.03384*.
3. Akkad G, Mansour A, Inaty E. 2023 Embedded Deep Learning Accelerators: A Survey on Recent Advances. *IEEE Transactions on Artificial Intelligence*.
4. Alwarafy A, Al-Thelaya KA, Abdallah M, Schneider J, Hamdi M. 2020 A survey on security and privacy issues in edge-computing-assisted internet of things. *IEEE Internet of Things Journal* **8**, 4004–4022.
5. Gustafson J. 2017 Posit arithmetic. *Mathematica Notebook describing the posit number system*.
6. Park MJ, Lee J, Cho K, Park J, Moon J, Lee SH, Kim TK, Oh S, Choi S, Choi Y et al.. 2022 A 192-Gb 12-high 896-GB/s HBM3 DRAM with a TSV auto-calibration scheme and machine-learning-based layout optimization. *IEEE Journal of Solid-State Circuits* **58**, 256–269.

7. Parashar A, Raina P, Shao YS, Chen YH, Ying VA, Mukkara A, Venkatesan R, Khailany B, Keckler SW, Emer J. 2019 Timeloop: A systematic approach to dnn accelerator evaluation. In *ISPASS* pp. 304–315. IEEE.
8. Zimmer B, Venkatesan R, Shao YS, Clemons J, Fojtik M, Jiang N, Keller B, Klinefelter A, Pinckney N, Raina P et al.. 2020 A 0.32–128 TOPS, scalable multi-chip-module-based deep neural network inference accelerator with ground-referenced signaling in 16 nm. *IEEE Journal of Solid-State Circuits* **55**, 920–932.
9. Park J, Yoon H, Ahn D, Choi J, Kim JJ. 2020 OPTIMUS: OPTImized matrix MUltiplication Structure for Transformer neural network accelerator. *Proceedings of Machine Learning and Systems* **2**, 363–378.
10. Ham TJ, Jung SJ, Kim S, Oh YH, Park Y, Song Y, Park JH, Lee S, Park K, Lee JW et al.. 2020 Aˆ 3: Accelerating attention mechanisms in neural networks with approximation. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)* pp. 328–341. IEEE.
11. Corporation N. 2023 NVIDIA H100 Tensor Core GPU Architecture Overview. Accessed: 2024-06-04.
12. Marchisio A, Dura D, Capra M, Martina M, Masera G, Shafique M. 2023 SwiftTron: An efficient hardware accelerator for quantized transformers. In *2023 International Joint Conference on Neural Networks (IJCNN)* pp. 1–9. IEEE.
13. Luo Y, Yu S. 2024 H3D-Transformer: A Heterogeneous 3D (H3D) Computing Platform for Transformer Model Acceleration on Edge Devices. *ACM Transactions on Design Automation of Electronic Systems* **29**, 1–19.
14. Yang X, Yan B, Li H, Chen Y. 2020 ReTransformer: ReRAM-based processing-in-memory architecture for transformer acceleration. In *Proceedings of the 39th International Conference on Computer-Aided Design* pp. 1–9.
15. Zhou M, Xu W, Kang J, Rosing T. 2022 Transpim: A memory-based acceleration via software-hardware co-design for transformer. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)* pp. 1071–1085. IEEE.
16. Mattson P, Cheng C, Coleman C, Diamos G, Micikevicius P, Patterson D, Tang H, Wei GY, Bailis P, Bittorf V et al.. 2019 MLPerf training benchmark. *arXiv preprint arXiv:1910.01500*.
17. Liu Z, Li G, Cheng J. 2021 Hardware acceleration of fully quantized bert for efficient natural language processing. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)* pp. 513–516. IEEE.
18. Amirshahi A, Klein JAH, Ansaloni G, Atienza D. 2023 Tic-sat: Tightly-coupled systolic accelerator for transformers. In *Proceedings of the 28th Asia and South Pacific Design Automation Conference* pp. 657–663.
19. Glint T, Jha CK, Awasthi M, Mekie J. 2023 Analysis of Conventional, Near-Memory, and In-Memory DNN Accelerators. In *2023 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)* pp. 349–351. IEEE.
20. Qin Y, Wang Y, Deng D, Zhao Z, Yang X, Liu L, Wei S, Hu Y, Yin S. 2023 Fact: Ffn-attention co-optimized transformer architecture with eager correlation prediction. In *Proceedings of the 50th Annual International Symposium on Computer Architecture* pp. 1–14.
21. Kim KM, Jeong YS, Bang IC. 2019 Thermal analysis of lithium ion battery-equipped smartphone explosions. *Engineering Science and Technology, an International Journal* **22**, 610–617.
22. Glint T, Prasad K, Dagli J, Gandhi K, Gupta A, Patel V, Shah N, Mekie J. 2023 Hardware-software codesign of dnn accelerators using approximate posit multipliers. In *ASPDAC* pp. 469–474.
23. Glint T, Awasthi M, Mekie J. 2024 Hardware-Software Co-Design of a Collaborative DNN Accelerator for 3D Stacked Memories with Multi-Channel Data. In *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)* pp. 454–459. IEEE.
24. SK hynix. 2024 SK hynix Partners with TSMC to Strengthen HBM Technological Leadership. https://news.skhynix.com/sk-hynix-partners-with-tsmc-to-strengthen-hbm-technological-leadership/. Accessed: 2024-09-16.
25. Gohil V, Walia S, Mekie J, Awasthi M. 2021 Fixed-Posit: A Floating-Point Representation for Error-Resilient Applications. *IEEE Transactions on Circuits and Systems II: Express Briefs* **68**, 3341–3345. (10.1109/TCSII.2021.3072217)
26. Muralimanohar N, Balasubramonian R, Jouppi NP. 2009 CACTI 6.0: A tool to model large caches. *HP laboratories* **27**, 28.
27. Wu YN, Emer JS, Sze V. 2019 Accelergy: An architecture-level energy estimation methodology for accelerator designs. In *ICCAD* pp. 1–8. IEEE.