applied sciences

MDPI

Article

A Survey on Hardware Accelerators for Large Language Models

Christoforos Kachris

Department of Electrical and Electronics Engineering, University of West Attica, 12243 Egaleo, Greece; kachris@uniwa.gr

Abstract: Large language models (LLMs) have emerged as powerful tools for natural language processing tasks, revolutionizing the field with their ability to understand and generate human-like text. As the demand for more sophisticated LLMs continues to grow, there is a pressing need to address the computational challenges associated with their scale and complexity. This paper presents a comprehensive survey of hardware accelerators designed to enhance the performance and energy efficiency of large language models. By examining a diverse range of accelerators, including GPUs, FPGAs, and custom-designed architectures, we explore the landscape of hardware solutions tailored to meet the unique computational demands of LLMs. The survey encompasses an in-depth analysis of architecture, performance metrics, and energy efficiency considerations, providing valuable insights for researchers, engineers, and decision-makers aiming to optimize the deployment of LLMs in real-world applications.

Keywords: large language models; hardware accelerators; FPGAs; GPU; survey; Transformer; energy efficiency

1. Introduction

Large language models (LLMs), such as GPT-3, are a class of artificial intelligence systems designed to understand, generate, and process human language with a level of complexity and nuance previously unattainable. These models are trained on extensive datasets, encompassing a broad spectrum of human discourse, which enables them to perform a wide array of language-related tasks, from translation and summarization to conversation and content creation. Over the past few years, the field of LLMs has witnessed remarkable evolution, characterized by significant advancements in model architecture, training methodologies, and data processing capabilities.

This evolution has been driven by breakthroughs in deep learning and the increasing availability of large-scale computational resources. The impact of LLMs on society and business is profound and multifaceted. They have revolutionized the way we interact with technology, introducing new efficiencies and capabilities in areas such as customer service, content generation, and decision support systems. Moreover, LLMs have become instrumental in driving innovation in various sectors, including healthcare, finance, and education, by providing enhanced capabilities for data analysis, pattern recognition, and predictive modeling. This transformative influence underscores the importance of exploring and understanding the underpinnings of these models, their operational mechanisms, and the implications of their widespread application in diverse domains.

Hardware acceleration plays a pivotal role in advancing large language modeling (LLM) by optimizing computational efficiency and accelerating training and inference. The field is currently focused on developing specialized hardware, improving software–hardware integration, and exploring novel architectures to enhance performance. Several

Academic Editor: Alexander Barkalov

Received: 29 November 2024
Revised: 30 December 2024
Accepted: 6 January 2025
Published: 9 January 2025

Citation: Kachris, C. A Survey on Hardware Accelerators for Large Language Models. *Appl. Sci.* **2025**, *15*, 586. https://doi.org/10.3390/app15020586

architectures have been proposed that are focused on utilizing specialized hardware, memory optimizations, and increasing energy efficiency using the accelerators.

Specialized hardware: Various hardware accelerators are employed to optimize the performance of large language models (LLMs), each with unique strengths and applications:

- Graphics processing units (GPUs): GPUs, such as NVIDIA's A100 and H100, are the most widely used accelerators for training and inference in LLMs. Their high parallelism and specialized tensor cores make them ideal for matrix multiplication, which is fundamental to deep learning. GPUs are versatile and widely supported by software frameworks such as PyTorch and TensorFlow, excelling in both training and inference scenarios.
- Field-programmable gate arrays (FPGAs): FPGAs are customizable chips that can be reprogrammed for specific tasks, offering flexibility and efficiency for AI workloads. They are particularly useful in scenarios requiring tailored optimization, such as low-latency inference or energy-efficient deployments. However, their adoption for LLMs is limited due to higher programming complexity compared to GPUs.
- In-memory computing: In-memory computing is an emerging paradigm that integrates computation directly within memory cells, reducing the data movement bottleneck. This approach shows promise for tasks involving large-scale matrix operations and is being explored for energy-efficient LLM inference. While still in its early stages, IMC could overcome the limitations of traditional memory bandwidth.
- Application-specific integrated circuits (ASICs): ASICs are custom-designed chips tailored for specific AI workloads, offering unmatched efficiency and performance. Examples include Google's tensor processing units (TPUs) and Cerebras' wafer-scale engines. These accelerators are optimized for large-scale training and inference, with specialized architectures that maximize throughput and energy efficiency. However, their lack of versatility and high development costs limit their widespread use.

Until now there has been no comprehensive survey on the hardware accelerators to speed up the most computationally intensive tasks of Transformers. In [1], a survey was presented on the hardware acceleration of Transformer networks for autonomous driving. The paper presented several efforts on the acceleration of tasks such as object detection, 3D segmentation, and lane detection.

In 2022, Huang et al. presented a survey on hardware acceleration for Transformers [2]. The paper mostly focused on the Transformer model compression algorithm based on the hardware accelerator and was limited mostly to FPGA-based implementation.

In 2023, Emani et al. [3] presented a comprehensive performance study of LLMs on several computing platforms and evaluated the performance characteristics of these models. The authors evaluated these systems using micro-benchmarks, a GPT-2 model, and an LLM-driven science use case, called GenSLM. For the evaluation, they used GPUs and other AI accelerators such as Sambanova, Cerebras, Graphcore, and Habana Gaudi 2. Specifically, they used LLM applications to evaluate the performances of the systems and categorized the proposed solution for the acceleration of the LLM applications.

In this paper, we present a comprehensive survey of various research efforts that have aimed to accelerate Transformer networks for large language models and NLP using hardware accelerators. This survey outlines the proposed frameworks and then provides a qualitative and quantitative comparison of each, focusing on the technology, processing platform (GPU, FPGA, in-memory computing, ASIC), speedup, and energy efficiency of each framework.

2. Computational and Energy Requirements

2.1. Computational Complexity

Large language models (LLMs) are among the most computationally intensive applications in contemporary artificial intelligence research. The computational intensity of LLMs is primarily attributed to their architecture and the scale of the data they process. These models typically consist of hundreds of millions to billions of parameters, which are the learned weights that determine how the model processes and generates language. Training these parameters requires iterating over vast datasets multiple times, a process that demands substantial computational resources.

Moreover, the complexity of the tasks LLMs are designed to perform exacerbates their computational demands. Tasks such as contextual understanding, nuanced language generation, and handling ambiguities in human language require deep neural networks with multiple layers, each contributing to the overall computational load. The iterative nature of model training and fine-tuning, involving continuous forward and backward propagation through these deep networks, further amplifies the computational requirements.

One of the most computationally intensive parts of the LLMs is the Transformer networks that were introduced in 2017 [4]. Transformer networks use a technique called attention. Attention, adopted by the field of neuroscience, is the ability to selectively concentrate on specific data while ignoring other data of the environment. In deep learning, we imitate this technique through attention mechanisms. One approach is to encode a sequence not into a single fixed vector, but rather to create a model that produces a unique vector for each output step, by adding a set of weights that will later be optimized.

Consequently, it does not simply learn what to produce in the output, but also how to selectively weigh specific input data, maximizing the probability of the correct output. For example, BERT uses multiple attention blocks. Each attention block converts the input using GEMM (general matrix multiplication) operations and then uses both GEMM and non-linear functions such as softmax, layer normalization, and Gaussian error linear unit (GELU) to produce the output.

Softmax: Softmax is a mathematical function that normalizes input scores into a probability distribution, making the outputs sum to 1. In attention mechanisms, it is used to compute attention weights, highlighting the relative importance of different elements in a sequence. Mathematically, it is defined as follows:

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

where x_i represents the input scores.

ReLU (rectified linear unit): ReLU is an activation function that introduces non-linearity by allowing only positive values to pass through and setting negative values to zero. It is computationally efficient and helps prevent the vanishing gradient problem. The ReLU function is expressed as follows:

$$\text{ReLU}(x) = \max(0, x)$$

GELU (Gaussian Error Linear Unit): GELU is a smooth, non-linear activation function that combines the benefits of ReLU and sigmoid functions. It enables models to capture nuanced patterns by scaling inputs based on their value. The GELU function is approximated as follows:

$$\text{GELU}(x) = x \cdot \Phi(x)$$

where $\Phi(x)$ is the cumulative distribution function of a standard normal distribution. GELU is commonly used in Transformers as it provides better gradient flow and performance compared to ReLU.

An approximation of $\Phi(x)$ is often used in practice, as follows:

$$\text{GELU}(x) \approx 0.5 \cdot x \cdot \left(1 + \tanh\left(\sqrt{\frac{2}{\pi}}(x + 0.044715 \cdot x^3)\right)\right)$$

These building blocks work in tandem to enable attention networks to model complex relationships within data, contributing to their effectiveness in natural language processing and other tasks. Together with GEMM, these building blocks are often the most computationally intensive parts of the LLM that are often mapped to specialized hardware resources.

2.2. Computational Complexity in Training

The training phase of large language models (LLMs) is inherently complex due to the extensive computational resources required to process and learn from massive datasets. This complexity stems from several key factors:

- **Volume of parameters:** LLMs, such as GPT-3, comprise an extraordinarily high number of parameters (often in the billions), each requiring computation during the training process. Adjusting these parameters to minimize errors involves extensive computational efforts.
- **Training data sizes:** LLMs are trained on vast datasets, necessitating significant computational power to process and learn from the data. This process includes tasks like tokenization, encoding, and the application of complex algorithms to understand linguistic patterns and structures.
- **Depth of neural networks:** Deep neural network architectures employed in LLMs, consisting of multiple layers, add to the computational load. Each layer contributes to the model's ability to understand and generate language, requiring substantial computation for both forward and backward propagation during training.

2.3. Computational Complexity in Inference

While it is often presumed that the computational complexity of LLMs is predominantly a concern during the training phase, the inference phase also demands significant computational resources:

- **Model size and memory footprint:** The large number of parameters in LLMs translates to a substantial memory footprint, even during inference. Managing and accessing these parameters efficiently requires considerable computational resources, especially when processing complex queries or large volumes of data.
- **Real-time processing requirements:** For applications requiring real-time responses, such as chatbots or interactive tools, the LLM must process input and generate output rapidly. This necessitates the allocation of significant computational resources to ensure low latency and high throughput.
- **Contextual and nuanced language processing:** The ability of LLMs to understand context and nuance in language, although less computationally intensive than training, still requires a considerable amount of processing power. This is particularly true for tasks involving long context windows or complex linguistic structures.

In conclusion, both the training and inference stages of LLMs are characterized by substantial computational complexity. While the nature and scale of this complexity may differ between the two phases, both necessitate significant computational resources to

achieve the high levels of performance and accuracy expected from state-of-the-art language models. This underlines the importance of ongoing research into more efficient model architectures and computational strategies to optimize both the training and application of LLMs.

2.4. Energy Consumption

The immense computational requirements of LLMs translate into significant energy consumption. Training a state-of-the-art LLM can consume as much energy as the lifetime consumption of multiple cars, highlighting the environmental impact of these models. This energy consumption primarily stems from the use of high-performance GPUs or TPUs that are necessary for the parallel processing capabilities required in training these models.

The energy-intensive nature of LLMs raises concerns regarding their carbon footprint, especially when trained and operated in data centers powered by non-renewable energy sources. The continuous operation of these models for inference tasks also contributes to ongoing energy usage, although it is typically less than that required for training. The trade-off between the benefits offered by LLMs and their environmental impact is an ongoing area of research and debate, emphasizing the need for more energy-efficient model architectures and the adoption of green computing practices in the AI field.

Efforts to mitigate the energy consumption of LLMs include optimizing model architecture for efficiency, implementing more energy-efficient hardware, and utilizing renewable energy sources for data centers. These measures aim to reduce the environmental impact and maintain the advanced capabilities of LLMs.

2.5. Benchmarks for LLMs

Large language models (LLMs) such as BERT [5], GPT-2, and others are evaluated using well-established benchmarks to assess their performance across various natural language processing tasks. Common benchmarks include GLUE [6] (general language understanding evaluation) for tasks like sentiment analysis and natural language inference, SQuAD [7] (Stanford Question Answering Dataset) for extractive question answering, and SuperGLUE [8] for more complex understanding tasks. Additionally, tasks like text classification, summarization, machine translation, and conversational AI are widely used to test these models' capabilities. These benchmarks and applications provide a standardized way to measure progress and compare different models in diverse language understanding and generation scenarios.

3. FPGA-Based Accelerators

This section presents the most promising schemes for accelerating LLMs using field-programmable gate arrays (FPGAs). The main advantage of FPGAs is that they can be programmed with specialized architectures customized to the specific applications they need to accelerate.

3.1. MNNFast

In 2019, Jang et al. proposed MNNFast to accelerate large language models [9]. MNNFast accelerates large language models using three optimizations. First, to reduce memory bandwidth consumption, MNNFast introduces a new column-based algorithm with streaming, which minimizes the size of data spills and conceals most of the off-chip memory access overhead. Second, to decrease high computational overhead, MNNFast implements a zero-skipping optimization that bypasses a significant amount of output computation. Lastly, MNNFast introduces a dedicated embedding cache designed to efficiently cache the embedding matrix.

MNNFast has been implemented across various platforms (CPU, GPU, and FPGA). MNNFast improves the overall throughput by up to 5.38×, 4.34×, and 2.01× on the CPU, GPU, and FPGA, respectively. The implementation that has been ported to FPGA achieves 6.54× higher energy efficiency.

3.2. FTRANS

In 2020, Li et al. [10] presented a hardware acceleration framework named FTRANS, which targets the acceleration of Transformer-based large-scale language representations. The proposed framework incorporates an enhanced block-circulant matrix (BCM)-based weight representation to enable model compression at the algorithmic level for large-scale language representations, significantly reducing the model size (by up to 16 times) with minimal accuracy degradation.

FTRANS partitions the model into the embedding layer and encoder/decoder stacks, with a focus on off-loading the embedding layer to off-chip memory to optimize computation. Additionally, it includes the development of a design automation and optimization technique for maximum throughput. FTRANS addresses the large size and complex dataflow of Transformer-based models by proposing a two-stage optimization approach. This method effectively schedules computation resources to optimize latency and throughput on FPGA platforms.

Based on the performance evaluation, FTRANS achieves a 27× speedup and an 81× improvement in energy efficiency compared to CPUs, and up to an 8.8× improvement in energy efficiency compared to GPUs.

3.3. Multi-Head Attention

In 2020, Lu et al. presented an FPGA-based architecture for the acceleration of the most computationally intensive parts of Transformer networks [11]. In their work, they propose a novel hardware accelerator for two key components, i.e., the multi-head attention (MHA) ResBlock and the position-wise feedforward network (FFN) ResBlock, which are the two most complex layers in the Transformer.

Firstly, an efficient method is introduced to partition the huge matrices in the Transformer, allowing the two ResBlocks to share most of the hardware resources. Secondly, the computational flow is well-designed to ensure high hardware utilization of the systolic array, which is the largest module in the design. Thirdly, complicated nonlinear functions are highly optimized to further reduce the hardware complexity and the latency of the entire system.

The proposed framework is implemented on a Xilinx FPGA. Based on the performance evaluation, the proposed design achieves a speedup of 14.6× compared to a V100 GPU.

3.4. FPGA NPE

In 2021, Khan et al. presented an FPGA acceleration for language models called NPE [12]. The NPE architecture consists of an instruction control unit (ICU), a memory read unit (MRU), a memory write unit (MWU), a matrix multiply unit (MMU), and a nonlinear vector unit (NVU).

The MMU computation consists of five stages: data selection, inner product, adder tree reduction, accumulation, and quantization. Data selection loads the necessary matrix operands from the input buffers and rearranges them as needed. The matrix multiplication is implemented using an array of PEs, where each PE performs an inner product using an array of multipliers followed by an adder tree. The MMU implementation features 128 PEs, each equipped with 16 multiply-accumulate units (for a total of 2048 multipliers). In our implementation, these multiply-accumulate units are mapped to the FPGA's DSP slices.

The key novel component of NPE's design is the nonlinear vector unit (NVU) that handles high-throughput nonlinear function computation with minimal resource overhead. The NVU is a data-parallel vector load/store architecture that performs arithmetic and logical operations on multiple elements in each clock cycle.

NPE was implemented on a Xilinx Zynq Z-7100 FPGA board clocked at 200 MHz and is compared with other frameworks like FTRANS, as well as implementations on CPUs and GPUs. Although it does not achieve significant speedup compared to other computing platforms, its main advantage lies in energy efficiency. NPE achieves approximately 4× better energy efficiency compared to a CPU (i7-8700k) and 6× compared to a GPU (RTX 5000).

3.5. Column-Balanced Block-Wise Pruning

In 2021, Peng et al. presented a novel scheme for accelerating Transformer networks using column-balanced block-wise pruning [13]. Column-balanced block-wise pruning combines the key features of both bank-balanced pruning and block-wise pruning. Column-balanced block-wise pruning ranks the blocks' L2 norm by each column to obtain the pruning thresholds and prune blocks for each column.

By combining the compressed sparse bank (CSB) format and the block-compressed sparse row (BCSR) format, a compressed sparse column block (CSCB) is formed. A specialized process element (PE) is then introduced for the sparse matrix multiplication accelerator, and multiple PEs can be used to increase the accelerator throughput.

The proposed framework was implemented on different hardware platforms (Intel i5-5257U (2.7 GHZ) CPU, NVIDIA Jetson TX2 GPU, and Xilinx Alveo U200 FPGA) for a further comparison of latency and throughput. The experimental results showed that the FPGA platform achieves an 11× speedup compared to the CPU platform and a 2× speedup compared to the GPU platform.

3.6. FPGA DFX

In 2022, Hong et al. presented DFX [14] for the acceleration of the Transformer networks used in LLMs. Similar to NPE, the DFX architecture proposed a modular architecture consisting of several computer cores for the acceleration of the Transformer networks.

To address the sequential characteristic of text generation, the DFX compute core is optimized for single token processing. It also uses an efficient tiling scheme and dataflow architecture based on utilizing the high bandwidth of HBM memory. DFX uses model parallelism on the multi-FPGA system to increase the physical number of compute cores that work in parallel while evenly assigning a full workload to each device. DFX proposes custom instruction set architecture (ISA) at the assembly language level to support the end-to-end processing for inference. Similar to NPE, the DFX core has two processing units, namely, the matrix processing unit (MPU) and the vector processing unit (VPU).

For the evaluation, DFX was implemented on an Intel Xeon Gold 6226R CPU with four Xilinx Alveo U280 data center acceleration cards. DFX achieved an average of 3.8× throughput and 4× higher energy efficiency compared to the NVIDIA V100 GPU.

3.7. FPGA OPU

In 2023, Bai et al. proposed another scheme for the acceleration of Transformer networks, called Overaly OPU [15]. To support the inference of diverse networks, they proposed a configurable computation unit. Specifically, they proposed 48 processing elements (PEs) that were configured for the acceleration of the Transformer networks. The output stage of the adder tree could be switched during the inference process. Thus, data from the forward modules could flow through the computation unit in the pre-defined connection state. The proposed scheme achieves 5×–15× speedup compared to a CPU, a

1.1–2.9× speedup compared to a GPU (RTX 3090), and a 1.10–2.5× speedup compared to other FPGA accelerators, such as NPE [12].

3.8. FPGA Acceleration of Transformer Networks

In 2022, Tzanos et al. presented a high-performance hardware accelerator for the Transformer networks [16].

The paper presented a novel architecture targeting the FPGA platform used to speed up the following specific functions; general matrix multiplication, softmax, layer normalization, and GELU.

Since GEMM accounts for about 70% percent of the total execution time, the architecture primarily focuses on optimally accelerating the matrix multiplication operations of the network. Furthermore, the goal of this scheme was to integrate matrix multiplication—the most computationally intensive part of the Transformer network—with functions that have minimal impact on runtime, such as GELU, softmax, and LayerNorm. This integration aims to prevent unnecessary data transfers between the host and the FPGA kernel and vice versa, which would otherwise add significant cost to the implementation. The kernels were developed using Vivado HLS, targeting the Alveo U200 FPGA card, and were clocked at 411 MHz.

Performance evaluation showed that the proposed framework can achieve a 2.3× system speedup for the BERT model compared to a 40-thread processor and 80.5× speedup over a single-core CPU.

3.9. FlexRun

In 2023, Hur et al. presented an FPGA-based accelerator to speedup the diverse and complex NLP models, called FlexRun [17]. The paper focused on accelerating both recurrent neural network (RNN) models, such as SRNN or long short-term memory (LSTM) models, and attention-based NLP models, such as Transformer and GPT2.

FlexRun consists of three main schemes, namely, FlexRun:Architecture, FlexRun:Algorithm, and FlexRun:Automation.

The main advantage of FlexRun is that it exploits the high reconfigurability of FPGAs to dynamically adapt the architecture to the target model and its configuration. The base architecture of FlexRun alleviates the overhead of vector operations by adopting a deeply pipelined architecture. The architecture consists of parameterized pre-defined basic modules that can be configured to fit the input model and its configuration.

Next, researchers conducted a design space exploration of the algorithms to identify the optimal compute unit design by selecting the best modules for the input models using the FlexRun:Algorithm framework. Finally, using the results from FlexRun:Algorithm, they determined the best architecture for the specific models and implemented this architecture on the FPGAs using a tool called FlexRun:Automation.

Based on the profiling of NLP and LLM applications, they found that the most computationally intensive tasks were general matrix multiplication and vector processing. Hence, they proposed a modular architecture that consisted of general matrix multiplication units (GEMV units) and vector processing units. GEMV units are composed of multiple SIMD arithmetic units and the vector unit executes vector operations with some additional operators (i.e., reduction, exp, GELU) to support attention-based NLP models.

For evaluation, they compared FlexRun with Intel's Brainwave-like architecture on a Stratix-10 GX FPGA and a Tesla V100 GPU with tensor cores enabled. Compared to the FPGA baseline, FlexRun achieved an average speedup of 1.59× on various configurations of BERT. For GPT-2, FlexRun achieved an average speedup of 1.31×. Furthermore, compared to

the GPU implementation, FlexRun improved performance by 2.79× for BERT and 2.59× for GPT-2, respectively.

3.10. ODE-Based Acceleration

In 2024, Okubo et al. proposed a hybrid approach for the acceleration of the Transformer networks [18]. The proposed scheme uses ResNet as a backbone architecture and replaces a part of its convolution layers with an MHSA (multi-head self-attention) mechanism. Using this approach they managed to significantly reduce the parameter size of such models by using neural ODE (ordinary differential equation) as the backbone architecture instead of ResNet. The proposed hybrid model reduced the parameter size by 94.6% (compared to the CNN-based models) without degrading the accuracy.

The proposed model was deployed on a modestly sized FPGA device for edge computing applications. To further reduce FPGA resource utilization, the researchers employed a quantization-aware training (QAT) scheme, rather than post-training quantization (PTQ), to minimize accuracy loss. As a result, an extremely lightweight Transformer-based model can be implemented on resource-limited FPGAs, making it ideal for edge applications.

A performance evaluation on the Xilinx Zynq UltraScale+ MPSoC platform showed that the proposed FPGA implementation achieved 12.8× speedup and 9.2× energy efficiency compared to an ARM Cortex-A53 CPU implementation. This scheme shows how the parameter size reduction can significantly affect the performance of the hardware accelerators.

4. CPU- and GPU-Based Accelerators

4.1. SoftMax

In 2022, Choi et al. presented a novel framework for the acceleration of Transformer networks through recomposing softmax layers [19]. The softmax layer normalizes the elements of the attention matrix to values between 0 and 1. This operation was conducted along the row vector of the attention matrix. Based on the profiling, the softmax layer in the scaled dot-product attention (SDA) block used 36%, 18%, 40%, and 42% of the total execution times of BERT, GPT-Neo, BigBird, and Longformer, respectively.

In their work, the authors proposed decomposing the softmax layer into sub-layers to match their data access patterns with adjacent layers. Then, by fusing the decomposed softmax sub-layers with the subsequent and preceding operations, they reduced the off-chip memory traffic and, thus, reduced the total execution time for this layer.

Softmax recomposition achieved speedups of up to 1.25× for BERT, 1.12× for GPT-Neo, 1.57× for BigBird, and 1.65× for Longformer on an A100 GPU by significantly reducing off-chip memory traffic without any degradation in model accuracy.

The reduction in off-chip memory traffic can also reduce energy consumption, although this was not measured in the specific paper.

4.2. LightSeq2

In 2022, Wang et al. proposed a series of GPU optimizations to accelerate training for a general family of Transformer models on GPUs, called LightSeq2 [20].

LightSeq2 proposes three techniques for the acceleration of the training of Transformer networks. Firstly, for all types of Transformers, LightSeq2 uses fused kernel operators for both the encoder and decoder layers. Adjacent fine-grained element-wise kernels are fused into one coarse-grained kernel, resulting in fewer kernel launches and intermediate results. For example, the last kernel of the self-attention layer implements bias adding, dropout, and residual kernels with only one kernel launch.

LightSeq2 also applies mixed-precision updates for training. During the initialization of the trainer, LightSeq2 copies all pieces of parameters/gradients into one tensor. Then

it resets and links them as fragments of workspace. During each training step, LightSeq2 only executes the trainer kernel once to update the workspace, which prevents launching a huge number of fragmented GPU kernels on every parameter/gradient piece.

Finally, LightSeq2 reduces the memory footprint by minimizing memory allocations and releases, at no extra cost. LightSeq2 divides GPU memory into permanent sections with fixed sizes for storing parameters and gradients, and temporary sections with variable sizes for storing intermediate tensors.

The performance evaluation shows that LightSeq2 is consistently faster (1.4–3.5×) than previous systems on different GPUs and can achieve up to a 3× speedup on large public datasets.

4.3. Simplified Transformer Networks

He and Hofmann [21] also proposed a novel framework to accelerate Transformer networks in GPUs by simplified Transformers without compromising convergence properties and downstream task performance.

Based on the signal propagation theory and empirical evidence, they found that many parts can be removed to simplify GPT-like decoder architectures as well as encoder-style BERT models. Specifically, they showed that it is possible to remove skip connections, value parameters, projection parameters, and sequential sub-blocks, all while matching the standard Transformer in terms of training speed and downstream task performance.

Based on the performance evaluation (both on the autoregressive decoder-only and BERT encoder-only models), simplified Transformers emulate the per-update training speed and performance of standard Transformers, while enjoying 15% faster training throughput in GPUs, and using 15% fewer parameters.

4.4. LLMA

In 2023, Microsoft presented LLMA [22], an LLM accelerator that is used to speed up large language model (LLM) inference with references. LLMA was mainly motivated by the observation that there are many identical text spans between the decoding result by an LLM and the reference that is available in many real-world scenarios (e.g., retrieved documents). LLMA first selects a text span from the reference and then copies its tokens to the decoder. It then evaluates the tokens' appropriateness as the decoding results in parallel within a single decoding step. This improved computational parallelism enables LLMA to achieve over a 2× speedup for large language models (LLMs) with generation results identical to those of greedy decoding. However, this speedup is achieved only in use cases where there is significant overlap between in-context references and outputs, such as in search engines and multi-turn conversations.

The main advantage of LLMA, compared to other efficient decoding algorithms such as speculative decoding [23] and speculative sampling [24] is significant. , While these methods require an additional efficient drafter model to generate a draft for checking, LLMA does not require an additional model and is much easier to implement and deploy.

For the performance evaluation, the authors used the LLaMA model with 7 B, 13 B, and 30 B parameters, and all the inferences were conducted in half-floating numbers. For the 7 B and 13 B models, the inferences were performed on one NVIDIA 32 GB V100 GPU, and for the 30 B model, the inference was performed on four NVIDIA 32 GB V100 GPUs on a single machine.

The improved computational parallelism enabled LLMA to achieve a 2× to 3× speedup over the baseline for LLMs, with generation results identical to those of greedy decoding in many practical scenarios, where significant overlap between in-context reference and outputs existed (e.g., search engines and multi-turn conversations).

4.5. UltraFastBERT

In 2023, Belcak and Wattenhofer presented a novel scheme for the acceleration of language models purely on software called UltraFastBERT [25]. UltraFastBERT selectively engages just 12 out of 4095 neurons for each layer inference. This is achieved by replacing feedforward networks with fast feedforward networks. The intermediate layers of UltraFastBERT are exponentially faster by design: given a feedforward (FF) and a fast feedforward (FFF) network, each with n neurons, the time complexity of a forward pass through the FFF is O(log2 n) instead of O(n) for FF.

UltraFastBERT achieved a 78× speedup over the optimized baseline feedforward implementation, and the PyTorch implementation delivered a 40x speedup over the equivalent batched feedforward inference.

5. ASIC Accelerators

5.1. A3

One of the early research studies on the acceleration of Transformer networks was proposed in 2020 by Hma et al., called A3 [26]. The authors proposed a hardware accelerator for attention mechanisms in NNs, which not only focused on the efficient implementation of the attention mechanism in hardware but also on reducing the computation amount in the attention mechanism through algorithmic optimization and approximation. The authors presented an approximate candidate selection mechanism to reduce the number of search targets (and, thus, the computation amount).

Additionally, they proposed a specialized hardware pipeline that exploits parallelism to accelerate approximated attention mechanisms. This enhancement not only boosts the system's performance but also increases its energy efficiency.

The proposed scheme has not been implemented on FPGA but has been implemented on a cycle-accurate Verilog design targeting a TSMC 40nm ASIC clocked at 1GHz. Based on the performance evaluation, the proposed scheme can achieve up to a 7× speedup compared to an Intel Gold 6128 CPU implementation and up to 11x better energy efficiency versus a CPU implementation.

5.2. ELSA

In 2021, Ham et al. presented a hardware–software co-design approach for the acceleration of Transformer networks, called Elsa [27].

Based on the fact that irrelevant relations can be effectively filtered out by computing approximate similarity, ELSA substantially reduces computational waste in a self-attention operation. Unlike conventional hardware such as CPUs or GPUs, which cannot benefit from approximation, ELSA proposes specialized hardware that directly translates this reduction to further improve performance and energy efficiency.

The approximate self-attention scheme consists of three sub-operations. The first one estimates the angle between two vectors (e.g., a key and a query) with minimal computation by utilizing the concise representations of the key and the query. Then, an estimated angle is utilized to compute the approximate similarity between a query and a key, based on the fact that the dot product is directly proportional to the cosine of the angle between two vectors. Finally, the approximate similarity is compared with a certain threshold to identify whether a specific key is relevant to the query or not.

The authors evaluated several representative self-attention-oriented NN models to demonstrate the effectiveness of ELSA. For performance evaluation, they implemented a custom simulator for ELSA, targeting a 40 nm ASIC clocked at 1 GHz. ELSA-moderate achieved up to a 157× speedup compared to GPUs, along with improvements in energy efficiency by two orders of magnitude over the GPU for the self-attention computation.

5.3. SpAtten

In 2021, Wang et al. presented a framework for the acceleration of large language models, called SpAtten [28].

SpAtten proposed a novel scheme for the acceleration of NLP using three algorithmic optimizations: cascade token pruning, cascade head pruning, and progressive quantization to reduce computation and memory access.

Pruning is applied to the tokens and heads, and not to weights (as is usually the case). Cascade means that once a token/head is pruned, it is removed in all following layers, so one layer only needs to process the remaining tokens/heads from previous layers. The deeper the layer, the more tokens/heads are pruned.

The three techniques are input-dependent since the pruned computation and bit-width are adaptive to input instances. Cascade pruning requires sorting token/head importance scores on the fly. The proposed hardware architecture leverages high parallelism in top-k engines for token and head selections, a specialized memory hierarchy, and a fully pipelined data path to improve performance and reduce energy consumption

The proposed scheme was implemented on a cycle-accurate design using SpinalHDL and mapped to ASIC using a 40 nm TSMC library. SpAtten achieved speedups of 162× and 347× over a TITAN Xp GPU and Xeon CPU, respectively. In terms of energy efficiency, SpAtten achieved energy savings of 1193× and 4059× compared to the GPU and CPU.

5.4. Sanger

In 2021, Lu et al. presented another novel approach for the acceleration of Transformer networks, called Sanger [29].

Sanger accelerates the sparse attention models by combining dynamic sparsity patterns and reconfigurable architecture. The software part provides sparsity patterns, which can achieve high performance and a balanced workload. The architecture is designed with reconfigurability to support the dynamic characteristics of sparsity, which helps to improve the compression ratio.

At the software level, Sanger proposes an algorithm to predict dynamic sparsity by computing a low-bit version of the attention matrix and zeroing out small attention weights through binary thresholding. To ensure load balance for efficient hardware implementation, Sanger encodes the resulting attention mask, which has unstructured sparsity, into multiple fine-grained structured blocks. The prediction and encoding processes are performed on the fly as Sanger dynamically determines the sparsity patterns on a per-sample basis.

At the hardware level, Sanger proposes a score-stationary dataflow that unifies the computation of SDDMM and SpMM operations. Using this dataflow, Sanger keeps the sparse scores stationary in the processing elements until the computation is finished, which effectively avoids the decoding overhead.

To enable greater flexibility in the sparsity patterns, Sanger proposes a reconfigurable systolic array based on this dataflow. The implementation was carried out in Chisel, which was then translated to Verilog RTL. The design targeted an ASIC using UMC 55nm technology and was clocked at 500 MHz.

Experiments on BERT show that Sanger can prune the model to a sparsity of 0.08–0.27 without losing accuracy, achieving speedups of 4.64× compared to the V100 GPU, 22.7× compared to the AMD Ryzen Threadripper 3970X CPU, and 2.39× and 1.47× compared to the state-of-the-art attention accelerators A3 and SpAtten, respectively

5.5. Energon

In 2023, Zhou et al. presented an algorithm–architecture co-design approach that accelerates various Transformers using dynamic sparse attention, called Energon [30]. En-

ergon proposes a mix-precision multi-round filtering (MP-MRF) algorithm to dynamically identify query–key pairs at runtime.

Energon adopts a low bit-width in each filtering round and only the finally selected pairs are used for high-precision tensors in the attention stage to reduce overall complexity. Thus, they manage to reduce computation costs by 4× to 8× with negligible accuracy loss.

Energon is implemented as a co-processor using a 45nm ASIC library. Based on the performance evaluation, Energon achieved speedups of 168× and 8.7×, and energy reductions of up to 10,000× and 1,000× over the Intel Xeon 5220 CPU and NVIDIA V100 GPU, respectively.

5.6. H3D Transformer

In 2024, a new 3D heterogeneous accelerator design was proposed for Transformer models [31]. It combines compute-in-memory (CIM) and digital tensor processing units (TPUs) to address chip area and energy consumption issues. The design uses 22 nm FeFET digital CIM chips for the high-density on-chip memory and processes MatMul tasks efficiently. It achieves 10 TOPS/W (10,000 GOPs/W) for BERT and GPT-2 models, which is 2.6× to 3.1× better than 7 nm TPU and FeFET memory baselines.

6. In-Memory Hardware Accelerators

6.1. ATT

In 2020, Guo et al. presented another approach for the acceleration of attention-based accelerators, called ATT [32], based on resistive RAM. ATT is based on crossbar-based resistive RAM that can eliminate weight movement between memory and processing units; it has a dedicated pipeline design for attention-based neural networks. The proposed scheme consists of several modules.

The Q-K-V engine employs a crossbar-based ReRAM to perform the matrix–vector multiplications. The mask issuer computes masks according to the matrix fetched by the Q-K-V engine. Two different masks are output from this module. One is forwarded to the attention engine, and the other is stored in the mask cache to filter the inputs and outputs of the fully connected engine.

Finally, the attention engine performs matrix-to-matrix multiplication in three stages. The first stage computes the inner products between q vectors and the transposed k vectors. The second stage applies softmax to the results of these inner products. The third stage involves multiplying the softmax results by the v vectors

The proposed scheme was simulated using CACTI 7.0 at 32 nm to model the power and area of the SRAM buffer and the mask cache. Based on the performance evaluation, ATT can achieve a speedup of 202× compared to the NVIDIA GTX 1080 Ti GPU.

6.2. ReTransformer

In 2020, Yang et al. proposed an in-memory framework for the acceleration of Transformers, called ReTransformer [33]. ReTransformer is a ReRAM-based in-memory architecture for Transformer acceleration that not only accelerates the scaled dot-product attention of the Transformer using ReRAM-based in-memory architecture but also eliminates some data dependency by avoiding writing the intermediate results using the proposed matrix decomposition technique. Furthermore, ReTransformer proposes a new sub-matrix pipeline design for multi-head self-attention.

ReRAM is used to perform matrix multiplication and scaling operations in Transformer networks. ReTransformer also optimizes softmax by incorporating in-memory logic to maximally exploit on-chip ReRAM crossbar arrays in the processing sub-arrays. Specifically,

ReTransformer uses ReRAM-based compare-and-select logic to identify the maximum values and uses look-up tables to perform the exponential and logarithm functions.

The performance evaluation shows that—compared to GPU—ReTransformer can achieve a speedup of up to 23.21× while the corresponding overall power is reduced by 1086×.

6.3. iMCAT

In 2021, Laguna et al. presented a novel in-memory architecture for the acceleration of Transformer networks for long sentences, called iMCAT [34]. The proposed framework uses a combination of Xbars and CAMs to accelerate Transformer networks. The acceleration of Transformer networks is achieved through a combination of several techniques. These include computing in-memory to minimize memory transfer overhead, caching reusable parameters to reduce the number of operations, exploiting available parallelism in the attention mechanism, and using locality-sensitive hashing to selectively filter sequence elements based on their importance.

The performance evaluation shows that this approach achieves a 200× speedup and 41× energy improvement for a sequence length of 4098.

6.4. X-Former

In 2023, Sridharan et al. presented a novel in-memory hardware acceleration to speed up Transformer networks, called X-Former [35]. X-Former is a hybrid spatial in-memory hardware accelerator that consists of both NVM and CMOS processing elements to execute Transformer workloads efficiently.

X-Former primarily consists of a projection engine equipped with NVM processing tiles for executing MV MStatic operations, and an attention engine with CMOS processing tiles for executing MV MDynamic operations. The main difference compared to other in-memory architectures is that the weights of all the layers are stored in the projection engine to prevent reprogramming the NVM tiles, while the attention engine is optimized to only process the largest self-attention layer due to area constraints.

X-Former proposes a sequence blocking dataflow, which overlaps the computations of the two processing elements and reduces the total execution time. X-Former has also a dedicated bus-based interconnect network for data transfers between the Projection engine and the attention engine that is used to reduce energy consumption and latency. Compared to the previous in-memory schemes, X-Former avoids frequent reprogramming by mapping static operations to NVM crossbars while processing the dynamic operations using in-memory CMOS processing elements.

Based on the performance evaluation, it is shown that X-Former achieves improvements of up to 85× and 7.5× in latency and energy over a NVIDIA GeForce GTX 1060 GPU and improvements of up to 10.7× and 4.6× in latency and energy over a state-of-the-art in-memory NVM accelerator.

6.5. Flash

In 2024, Apple presented a novel scheme for the efficient deployment of Transformer networks utilizing flash memory [36]. The paper tackled the challenge of efficiently running LLMs that exceed available DRAM capacity by storing model parameters in flash memory and retrieving them on demand to DRAM. The proposed method develops an inference cost model that considers the characteristics of flash memory to optimize in two critical areas: reducing the volume of data transferred from flash and reading data in larger, more contiguous chunks. Within this hardware-aware framework, Apple introduces two principal techniques.

The first technique, called "windowing", reduces data transfer by reusing previously activated neurons, and the second, "row-column bundling", tailored to the sequential data access strengths of flash memory, increases the data chunk sizes read from flash memory. These methods collectively enable running models of up to twice the size of the available DRAM, with increases of 4–5× and 20–25× in inference speed compared to naive loading approaches in CPU and GPU, respectively.

6.6. FlashAttention-3

FlashAttention elaborates on an approach to speed up attention on GPUs by minimizing memory reads/writes. A paper presented in 2024 proposed three main techniques to speed up attention on Hopper GPUs. These include exploiting the asynchrony of Tensor Cores and TMA to (1) overlap overall computation with data movement through warp-specialization, (2) interleave block-wise matrix multiplication and softmax operations, and (3) implement block quantization and incoherent processing, which takes advantage of hardware support for FP8 low-precision. Based on these novel techniques, the performance evaluation demonstrated that FlashAttention-3, achieves speedups on H100 GPUs by 1.5–2.0× with FP16 reaching up to 740 TFLOPs/s, and with FP8 reaching close to 1.2 PFLOPs/s. At the same time, it was validated that FP8 FlashAttention-3 achieved a 2.6× lower numerical error than baseline FP8 attention.

7. Quantitative Comparison

Table 1 shows all of the hardware-based accelerators that have been proposed, along with the main features of each accelerator. Each row lists the name of the accelerator, its type (FPGA/GPU/ASIC/In-memory), the speedup, the energy efficiency, and the reference platform for comparison. In some cases, the proposed scheme is compared against both a CPU and a GPU. In these cases, both speedup and energy efficiencies are shown. In this section, we present a quantitative and qualitative comparison of the proposed schemes.

Table 1. LLM Transformer accelerators.

Year	Framework	Technology	Speedup	Energy Efficiency	Baseline
2019	MNNFast [9]	FPGA 7020	5.4×	6.5×	CPU (Xeon 24-core)
2020	FTRANS [10]	FPGA VCU118	27×–81×	8.8×	CPU/GPU RTX5000
2020	Multi-Head [11]	FPGA XCVUP13	14×	—	GPU V100
2021	NPE [12]	FPGA Zynq7100	35×	4×–6×	CPU/GPU RTX5000
2021	Pruning[13]	FPGA U200	11×/2×	—	CPU (i5)/GPU TX2
2022	DFX [14]	FPGA U280	3.8×	4×	GPU V100
2022	Transformer [16]	FPGA U200	2.3×	—	CPU (80-threads)
2023	OPU [15]	FPGA	15×/2.9×	—	CPU (Gold)/GPU RTX3090
2023	FlexRun [17]	FPGA S10	2.7×	—	GPU V100
2024	ODE [18]	FPGA	12.8×	9.2×	ARM A53
2022	SoftMax [19]	GPU A100	2.5×	—	GPU A100
2022	LightSeq2 [20]	GPU A100	3×	—	GPU A100
2023	LLMA [22]	GPU V100	2×	—	GPU V100
2023	UltraFastBERT [25]	CPU	78×	—	CPU
2020	A3 [26]	ASIC 40 nm	7×	11×	CPU (Gold)/GPU Volta
2021	ELSA [27]	ASIC 40 nm	157×	1265×	GPU V100/TPU
2021	SpAtten [28]	ASIC 40 nm	347×/162×	4059×/1093×	CPU (Xeon)/GPU Titan
2021	Sanger [29]	ASIC 55 nm	22.7×/4.64×	—	CPU (3970)/GPU V100
2023	Energon [30]	ASIC 45 nm	168×/8.7×	10,000×/1000×	CPU/GPU V100
2024	H3D Transformer	ASIC 22 nm	–	3.1×	TPU (7 nm)
2020	ATT [32]	In-memory	202×	11×	CPU (Gold)/GPU Volta
2020	ReTransformer [33]	In-memory	23×	1086×	GPU

Table 1. *Cont.*

Year	Framework	Technology	Speedup	Energy Efficiency	Baseline
2022	iMCAT [34]	In-memory	200×	41×	GPU Titan RTX
2023	X-Former [35]	In-memory	85×	7.5×	GPU GTX1060
2023	Flash [36]	Flash	25×/5×	—	CPU/GPU
2024	FlashAttention-3 [37]	Flash	2×	–	GPU H100

7.1. Technology

There are four main computing platforms that researchers use to accelerate LLMs and Transformer networks: GPUs, FPGAs, ASICs, and in-memory computing. GPUs are the most obvious platform, as many LLMs utilize them for both training and inference. FPGAs can be customized with tailor-made architectures that speed up specific functions. The main idea is to offload the most computationally intensive tasks from the CPU to the FPGAs. This allows developers to maintain the flexibility of CPUs while simultaneously speeding up applications. The main advantage of FPGAs is that the proposed scheme can be implemented and evaluated on them to measure the real overall speedup.

There are also several schemes that target an ASIC implementation. These schemes propose a custom architecture that can be used as a co-processor to speed up the most computationally intensive tasks, like matrix multiplication, etc. Although none of the proposed schemes have been implemented in a real ASIC, all have been modeled in a cycle-accurate design and evaluated using cycle-accurate simulators and power estimators such as CACTI [38]. Most of the schemes have been evaluated using 40 nm process technology, while some have been implemented with 45 nm or 55 nm processes.

Finally, some accelerators target in-memory computing. In-memory computing is a promising technology that can utilize NVM and memristors to process data such as matrix multiplication. Although in-memory computing has many advantages, it is not yet widely used in the industry and it has several limitations in commercialization.

An interesting exception on this list is MNNFast [9], where the proposed optimizations have been applied on several platforms like CPUs, GPUs, and FPGAs. For CPUs, MNNFast was implemented in C++ using the open-source BLAS library, OpenBLAS. On GPUs, the optimizations utilized cuBLAS [39] from CUDA Toolkit 10.0 for matrix-to-matrix multiplication. Finally, for FPGAs, a dedicated FPGA-based accelerator for MNNFast was designed and implemented using Vivado high-level synthesis (HLS).

7.2. Speedup

The speedup is the execution time of the proposed scheme divided by the reference design. However, each accelerator is compared against a different reference computing platform, making it hard to evaluate the higher speedup against the same reference platform. Furthermore, some of the accelerators use a reference platform based on CPUs while others use a reference design based on GPUs. Even in cases where a scheme is compared against a CPU, it is not always clear whether the reference design is a single-core CPU or a multi-core CPU implementation. However, there are some general conclusions that can be drawn based on the comparison.

As shown in the table, ASIC and in-memory-based accelerators tend to provide much higher speedups compared to FPGA- and GPU-based accelerators. In-memory Transformer accelerators can achieve speedups of up to 200× while SpAtten achieves a speedup of up to 347 compared to CPUs. However, ASICs and in-memory computing need huge investments in terms of time and money for fabrication. FPGAs may offer lower speedups (up to 81× in the case of FTrans), but these platforms can be utilized immediately without the additional cost and time associated with ASICs.

7.3. Energy Efficiency

Energy efficiency refers to the total energy consumption to perform an operation compared to the reference platform. Again, each scheme is compared against different computing platforms, making it hard to evaluate the most energy-efficient solution. However, similar to speedup, ASICs and in-memory computing accelerators provide much better energy efficiency compared to FPGAs and GPUs. Energon claims up to four orders of magnitude better energy efficiency compared with CPUs and three orders of magnitude better compared to GPUs (V100). SpAtten and ELSA, both targeting a 40nm process technology, also achieve three orders of magnitude better energy efficiency compared to GPUs, thanks to algorithmic optimizations. However, as with speedup, FPGAs may offer lower energy efficiency but are readily available and can be integrated with off-the-shelf components in current data centers. Figure 1 shows the speedup versus the energy efficiency for the presented frameworks.

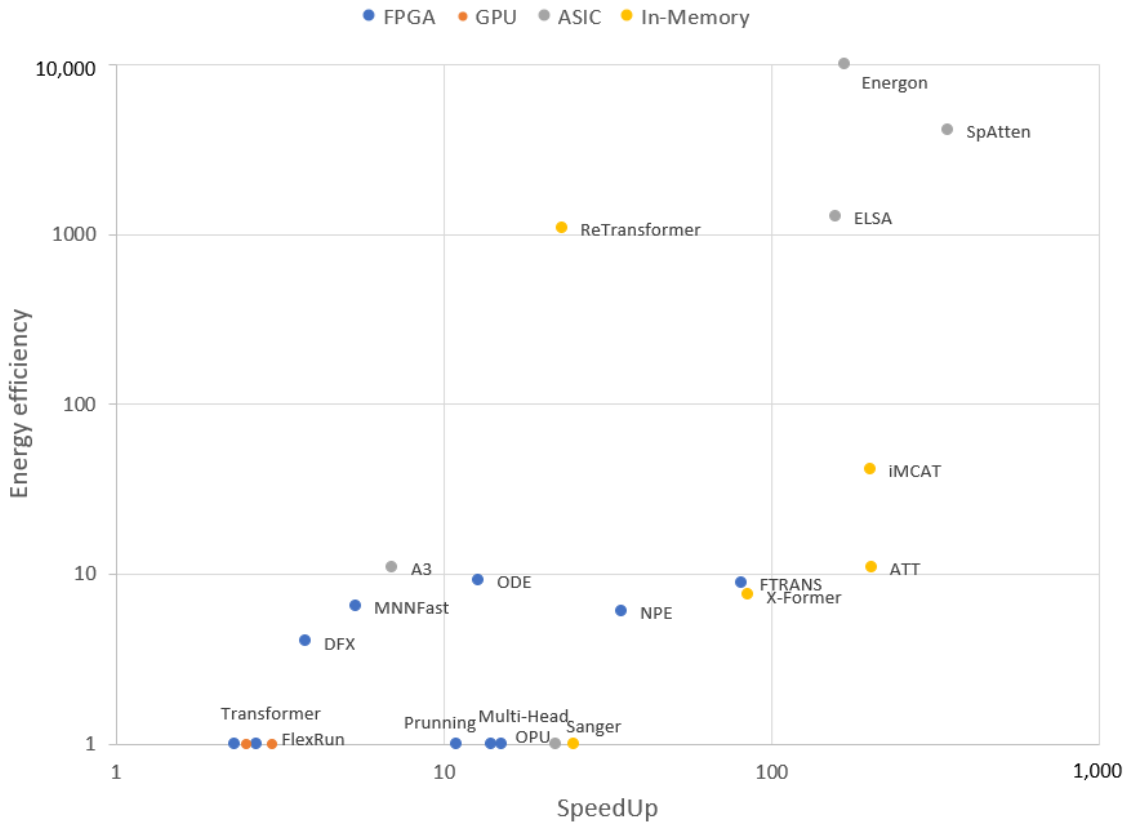

Figure 1. Speedup versus energy efficiency.

7.4. Scalability and Flexibility

One of the most important issues in the domain of LLM platforms involves scalability and flexibility, especially in the face of a growing model size and complexity. Each technology faces unique scalability and flexibility limitations when adapting to the demands of modern AI workloads. GPUs are versatile and widely used for LLMs; advanced interconnect technologies such as NVLink and NVSwitch allow multiple GPUs to be interconnected, facilitating easy scalability. However, their scalability is constrained by memory bandwidth and interconnect limitations. As model sizes grow into trillions of parameters, fitting models across multiple GPUs requires advanced techniques like tensor parallelism, which adds complexity to the system design.

FPGAs offer flexibility through re-programmability, making them suitable for customizing operations and optimizing energy efficiency. However, their scalability is limited

by lower computational density compared to GPUs and ASICs. Additionally, the time and expertise required to program FPGAs for large-scale LLMs reduce their practicality for rapidly evolving AI research.

ASICs, such as TPUs, provide unmatched efficiency and performance for specific tasks, making them ideal for large-scale training and inference. However, their fixed-function design lacks flexibility, making it challenging to adapt to new model architectures or unforeseen workloads. The high development costs and lead times for ASIC production further hinder their ability to keep pace with the rapidly advancing field of LLMs.

In summary, while GPUs provide the most flexibility, they struggle with energy efficiency. FPGAs excel in adaptability for new models but face limitations in performance and ease of use. ASICs offer the best scalability for predefined tasks but lack the flexibility needed to address the ever-evolving landscape of LLM architectures.

8. Conclusions

Large language models have emerged as promising and powerful technologies for science and society in general. However, their immense computational complexity poses new challenges in data centers, as these applications consume vast amounts of energy. Hardware accelerators can be used to speed up these applications and significantly reduce energy requirements. The architectures proposed so far demonstrate that hardware accelerators can be customized to speed up the most demanding functions of Transformer networks and can reduce energy requirements in data centers by more than four orders of magnitude. Other techniques, like FlashAttention [37], which are based on software optimizations, can also achieve significant speedups over the baseline and can significantly reduce energy consumption. These techniques could also be applied to hardware accelerators to boost the performance of LLM models. The reduction in energy consumption also leads to lower carbon emissions and reduced water use for cooling.

As LLMs continue to increase in complexity and processing requirements, the utilization of hardware accelerators will become essential in future data centers. Currently GPUs are the main hardware platforms used for the training and inference of LLMs. Future accelerators—whether based on GPUs, FPGAs, or ASICs—will likely feature highly specialized designs tailored to LLM workloads, such as sparse computations, low-precision arithmetic, and optimized memory hierarchies. Currently, there are some specialized start-ups, like d-Matrix and Etched, developing ASICs specifically for Transformer networks in LLMs. Innovations like in-memory computing and neuromorphic processors are very promising, and if they can be made cost-efficient for commercial products, they could provide a viable alternative to the dominance of GPUs.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data available in a publicly accessible repository https://github.com/kachris/survey_HA_LLM (accessed on 30 December 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Zhong, J.; Liu, Z.; Chen, X. Transformer-based models and hardware acceleration analysis in autonomous driving: A survey. *arXiv* **2023**. [CrossRef]
2. Huang, S.; Tang, E.; Li, S.; Ping, X.; Chen, R. Hardware-friendly compression and hardware acceleration for transformer: A survey. *Electron. Res. Arch.* **2022**, *30*, 3755–3785. [CrossRef]

3. Emani, M.; Foreman, S.; Sastry, V.; Xie, Z.; Raskar, S.; Arnold, W.; Thakur, R.; Vishwanath, V.; Papka, M.E. A Comprehensive Performance Study of Large Language Models on Novel AI Accelerators. *arXiv* **2023**. [CrossRef]
4. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need. *arXiv* **2023**. [CrossRef]
5. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv* **2019**. [CrossRef]
6. Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; Bowman, S.R. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. *arXiv* **2019**. [CrossRef]
7. Rajpurkar, P.; Zhang, J.; Lopyrev, K.; Liang, P. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *arXiv* **2016**. [CrossRef]
8. Wang, A.; Pruksachatkun, Y.; Nangia, N.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; Bowman, S.R. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. *arXiv* **2020**. [CrossRef]
9. Jang, H.; Kim, J.; Jo, J.E.; Lee, J.; Kim, J. MnnFast: A Fast and Scalable System Architecture for Memory-Augmented Neural Networks. In Proceedings of the 2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA), Phoenix, AZ, USA, 22–26 June 2019 ; pp. 250–263.
10. Li, B.; Pandey, S.; Fang, H.; Lyv, Y.; Li, J.; Chen, J.; Xie, M.; Wan, L.; Liu, H.; Ding, C. FTRANS: Energy-Efficient Acceleration of Transformers Using FPGA. In Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design, New York, NY, USA, 10–11 August 2020; ISLPED'20, pp. 175–180. [CrossRef]
11. Lu, S.; Wang, M.; Liang, S.; Lin, J.; Wang, Z. Hardware Accelerator for Multi-Head Attention and Position-Wise Feed-Forward in the Transformer. In Proceedings of the 2020 IEEE 33rd International System-on-Chip Conference (SOCC), Virtual, 8–11 September 2020; pp. 84–89. [CrossRef]
12. Khan, H.; Khan, A.; Khan, Z.; Huang, L.B.; Wang, K.; He, L. *NPE: An FPGA-Based Overlay Processor for Natural Language Processing*; Association for Computing Machinery: New York, NY, USA, 2021; FPGA '21, p. 227. [CrossRef]
13. Peng, H.; Huang, S.; Geng, T.; Li, A.; Jiang, W.; Liu, H.; Wang, S.; Ding, C. Accelerating Transformer-based Deep Learning Models on FPGAs using Column Balanced Block Pruning. In Proceedings of the 2021 22nd International Symposium on Quality Electronic Design (ISQED), Santa Clara, CA, USA, 7–9 April 2021; pp. 142–148. [CrossRef]
14. Hong, S.; Moon, S.; Kim, J.; Lee, S.; Kim, M.; Lee, D.; Kim, J.Y. DFX: A Low-latency Multi-FPGA Appliance for Accelerating Transformer-based Text Generation. *arXiv* **2022**. [CrossRef]
15. Bai, Y.; Zhou, H.; Zhao, K.; Chen, J.; Yu, J.; Wang, K. Transformer-OPU: An FPGA-based Overlay Processor for Transformer Networks. In Proceedings of the 2023 IEEE 31st Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), Marina Del Rey, CA, USA, 8–11 May 2023; pp. 221–221. [CrossRef]
16. Tzanos, G.; Kachris, C.; Soudris, D. Hardware Acceleration of Transformer Networks using FPGAs. In Proceedings of the 2022 Panhellenic Conference on Electronics and Telecommunications (PACET), Tripolis, Greece, 2–3 December 2022; pp. 1–5. [CrossRef]
17. Hur, S.; Na, S.; Kwon, D.; Kim, J.; Boutros, A.; Nurvitadhi, E.; Kim, J. A Fast and Flexible FPGA-Based Accelerator for Natural Language Processing Neural Networks. *ACM Trans. Archit. Code Optim.* **2023**, *20*, 1–24. [CrossRef]
18. Okubo, I.; Sugiura, K.; Matsutani, H. A Cost-Efficient FPGA Implementation of Tiny Transformer Model using Neural ODE. *arXiv* **2024**. [CrossRef]
19. Choi, J.; Li, H.; Kim, B.; Hwang, S.; Ahn, J.H. Accelerating Transformer Networks through Recomposing Softmax Layers. In Proceedings of the 2022 IEEE International Symposium on Workload Characterization (IISWC), Austin, TX, USA, 6–8 November 2022; pp. 92–103. [CrossRef]
20. Wang, X.; Wei, Y.; Xiong, Y.; Huang, G.; Qian, X.; Ding, Y.; Wang, M.; Li, L. LightSeq2: Accelerated Training for Transformer-Based Models on GPUs. In Proceedings of the SC22: International Conference for High Performance Computing, Networking, Storage and Analysis, Dallas, TX, USA, 13–18 November 2022; pp. 1–14. [CrossRef]
21. He, B.; Hofmann, T. Simplifying Transformer Blocks. *arXiv* **2023**. [CrossRef]
22. Yang, N.; Ge, T.; Wang, L.; Jiao, B.; Jiang, D.; Yang, L.; Majumder, R.; Wei, F. Inference with Reference: Lossless Acceleration of Large Language Models. *arXiv* **2023**. [CrossRef]
23. Xia, H.; Ge, T.; Wang, P.; Chen, S.Q.; Wei, F.; Sui, Z. Speculative Decoding: Exploiting Speculative Execution for Accelerating Seq2seq Generation. *arXiv* **2023**. [CrossRef]
24. Chen, C.; Borgeaud, S.; Irving, G.; Lespiau, J.B.; Sifre, L.; Jumper, J. Accelerating Large Language Model Decoding with Speculative Sampling. *arXiv* **2023**. [CrossRef]
25. Belcak, P.; Wattenhofer, R. Exponentially Faster Language Modelling. *arXiv* **2023**. [CrossRef]
26. Ham, T.J.; Jung, S.J.; Kim, S.; Oh, Y.H.; Park, Y.; Song, Y.; Park, J.H.; Lee, S.; Park, K.; Lee, J.W.; et al. A^3: Accelerating Attention Mechanisms in Neural Networks with Approximation. *arXiv* **2020**. [CrossRef]

27. Ham, T.J.; Lee, Y.; Seo, S.H.; Kim, S.; Choi, H.; Jung, S.J.; Lee, J.W. ELSA: Hardware-Software Co-design for Efficient, Lightweight Self-Attention Mechanism in Neural Networks. In Proceedings of the 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA), Valencia, Spain, 14–18 June 2021; pp. 692–705. [CrossRef]
28. Wang, H.; Zhang, Z.; Han, S. SpAtten: Efficient Sparse Attention Architecture with Cascade Token and Head Pruning. *arXiv* **2021**. [CrossRef]
29. Lu, L.; Jin, Y.; Bi, H.; Luo, Z.; Li, P.; Wang, T.; Liang, Y. Sanger: A Co-Design Framework for Enabling Sparse Attention Using Reconfigurable Architecture. In Proceedings of the MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture, New York, NY, USA, 18–22 October 2021; MICRO '21, pp. 977–991. [CrossRef]
30. Zhou, Z.; Liu, J.; Gu, Z.; Sun, G. Energon: Toward Efficient Acceleration of Transformers Using Dynamic Sparse Attention. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2023**, *42*, 136–149. [CrossRef]
31. Luo, Y.; Yu, S. H3D-Transformer: A Heterogeneous 3D (H3D) Computing Platform for Transformer Model Acceleration on Edge Devices. *ACM Trans. Des. Autom. Electron. Syst.* **2024**, *29*, 1–19. [CrossRef]
32. Guo, H.; Peng, L.; Zhang, J.; Chen, Q.; LeCompte, T.D. ATT: A Fault-Tolerant ReRAM Accelerator for Attention-based Neural Networks. In Proceedings of the 2020 IEEE 38th International Conference on Computer Design (ICCD), Los Alamitos, CA, USA, 18–21 October 2020; pp. 213–221. [CrossRef]
33. Yang, X.; Yan, B.; Li, H.; Chen, Y. ReTransformer: ReRAM-based Processing-in-Memory Architecture for Transformer Acceleration. In Proceedings of the 2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD), Virtual, 2–5 November 2020; pp. 1–9.
34. Laguna, A.F.; Kazemi, A.; Niemier, M.; Hu, X.S. In-Memory Computing based Accelerator for Transformer Networks for Long Sequences. In Proceedings of the 2021 Design, Automation and Test in Europe Conference and Exhibition (DATE), Virtual, 1–5 February 2021; pp. 1839–1844. [CrossRef]
35. Sridharan, S.; Stevens, J.R.; Roy, K.; Raghunathan, A. X-Former: In-Memory Acceleration of Transformers. *arXiv* **2023**. [CrossRef]
36. Alizadeh, K.; Mirzadeh, I.; Belenko, D.; Khatamifard, K.; Cho, M.; Mundo, C.C.D.; Rastegari, M.; Farajtabar, M. LLM in a flash: Efficient Large Language Model Inference with Limited Memory. *arXiv* **2024**. [CrossRef]
37. Shah, J.; Bikshandi, G.; Zhang, Y.; Thakkar, V.; Ramani, P.; Dao, T. FlashAttention-3: Fast and Accurate Attention with Asynchrony and Low-precision. *arXiv* **2024**. [CrossRef]
38. Shivakumar, P.; Jouppi, N. *CACTI 3.0: An Integrated Cache Timing, Power, and Area Model*; Western Research Laboratory: Palo Alto, CA, USA, 2001.
39. Zhang, B.; Yang, X.; Yang, F.; Yang, X.; Qin, C.; Han, D.; Ma, X.; Liu, K.; Tian, J. The CUBLAS and CULA based GPU acceleration of adaptive finite element framework for bioluminescence tomography. *Opt. Express* **2010**, *18*, 20201–20214. [CrossRef]