FLAME: Fully Leveraging MoE Sparsity for Transformer on FPGA

Xuanda Lin, Huinan Tian, Wenxiao Xue, Lanqi Ma, Jialin Cao, Manting Zhang, Jun Yu, Kun Wang*
School of Microelectronics, Fudan University, Shanghai, China
Email: *kun.wang@ieee.org

ABSTRACT

MoE (Mixture-of-Experts) mechanism has been widely adopted in transformer-based models to facilitate further expansion of model parameter size and enhance generalization capabilities. However, the practical deployment of MoE mechanism for transformer on resource-constrained platforms, such as FPGA, remains challenging due to heavy memory footprints and impractical runtime costs introduced by the MoE mechanism. Diving into the MoE mechanism, we raise two key observations: (1) Expert weights are heavy but cold, making it ideal to leverage expert weight sparsity. (2) There exists highly skewed expert activation paths for MoE layers in transformer-based models, making it feasible to conduct expert prediction and prefetching. Motivated by these two observations, we propose **FLAME**, the first algorithm-hardware co-optimized MoE accelerating framework designed to fully leverage MoE sparsity for efficient transformer deployment on FPGA. First, to leverage expert weight sparsity, we integrate an N:M pruning algorithm, allowing for the pruning of expert weights without significantly compromising model accuracy. Second, to settle expert activation sparsity, we propose a circular expert prediction (*CEPR*) strategy. *CEPR* prefetches expert weights from external storage to on-chip cache before the activated expert index is determined. Last, we co-optimize both MoE sparsity through the introduction of an efficient pruning-aware expert buffering (*PA-BUF*) mechanism. Experimental results demonstrate that FLAME achieves 84.4% accuracy of expert prediction with merely two expert caches on-chip. In comparison with CPU and GPU, FLAME achieves 4.12× and 1.49× speedup, respectively.

ACM Reference Format:
Xuanda Lin, Huinan Tian, Wenxiao Xue, Lanqi Ma, Jialin Cao, Manting Zhang, Jun Yu, Kun Wang*, School of Microelectronics, Fudan University, Shanghai, China, and Email: *kun.wang@ieee.org. 2024. FLAME: Fully Leveraging MoE Sparsity for Transformer on FPGA. In *61st ACM/IEEE Design Automation Conference (DAC '24), June 23–27, 2024, San Francisco, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3649329.3656507

*Corresponding author
This work was financially supported in part by National Key Research and Development Program of China under Grant 2021YFA1003602, in part by Shanghai Pujiang Program under Grant 22PJD003.

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

ACM ISBN 979-8-4007-0601-1/24/06…$15.00
https://doi.org/10.1145/3649329.3656507

1 INTRODUCTION

Transformer-based models are currently revolutionizing the field of machine intelligence with their remarkable performance. However, when the model parameter size approaches the scale of hundreds of billions, the further expansion of model size becomes difficult. The incorporation of the Mixture-of-Experts (MoE) mechanism [3, 6, 13] effectively tackles this issue. By replacing specific layers (*e.g.*, linear layer) with multi-expert MoE layers, MoE-based transformers sparsely activate only a few selected experts during inference, while all experts are employed during training. This approach facilitates further expansion of model parameter size while conserving computational resources, thus enhancing the inference and generalization capabilities of transformer models.

However, the introduction of the MoE mechanism also poses significant **challenges** to transformer models when it comes to deploying these models on edge devices like FPGA. On one hand, the MoE-based transformer necessitates an on-chip storage capacity far exceeds that of a conventional transformer model. This heightened storage requirement is often unaffordable for most edge devices. On the other hand, given that the MoE layer's experts are sparsely activated and dynamically determined on-the-fly, there are two choices to be made: one must either allocate substantial on-chip memory to store all expert weights, incurring a substantial memory overhead, or frequently load the activated experts with heavy off-chip data movement.

We have raised two key **observations** to tackle these challenges in MoE deployment for transformer on FPGA, which will be further illustrated in Section 4-6. First, expert weights are heavy but cold, leaving vast space for expert weight pruning without severely hurting model accuracy. With aggressively reduced expert weight parameters and sparsely activated experts, MoE deployment will be more feasible. Second, we pave the way to settle the challenges incurred by expert activation sparsity (see Figure 1(c)). We find that the expert activation path, *i.e.*, a vector of activated expert indexes of all transformer layers for one token, is highly skewed. It shows great potential on expert prediction and prefetching, giving rise to the idea of utilizing the expert cache for efficient edge deployment of MoE mechanism for transformer.

Based on these two key observations, we propose **FLAME**, an algorithm-hardware co-optimized MoE accelerating framework which fully leverages MoE sparsity for transformer on FPGA. We summarize the contributions as follows:

- To the best of our knowledge, FLAME is the first algorithm-hardware co-optimized MoE accelerating framework which fully exploits MoE sparsity for transformer on FPGA. The MoE sparsity exploration is orthogonal to optimizations tailored for traditional transformer models, and therefore can be applied together to achieve better performance.
- To leverage expert weight sparsity, we integrate N:M pruning to prune expert weights without severely hurting inference

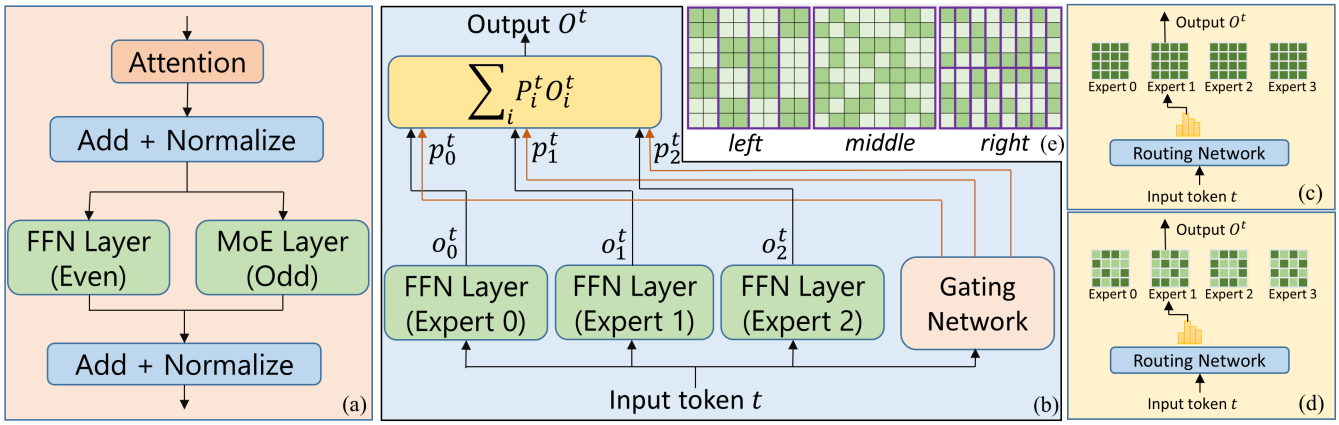

Figure 1: Illustration of Mixture-of-Experts (MoE) mechanism in transformer-based models. (a) MoE layer in switch transformer block. (b) Vanilla MoE layer. The gating network assigns weights to the outputs from each expert, facilitating the computation of a weighted sum. (c) Sparse MoE layer (top-1 routing). (d) Sparse MoE layer with sparse expert weights (top-1 routing). (e) Illustration of three pruning strategy (with 50% sparsity). *left*: Column-balanced block pruning. *middle*: Unstructured pruning. *right*: N:M pruning.

accuracy. To settle expert activation sparsity, we propose a circular expert prediction (*CEPR*) strategy after conducting comprehensive expert activation pattern modeling.

- To co-optimize MoE sparsity, we propose pruning-aware expert buffering (*PA-BUF*) based on *CEPR*, to further enhance MoE deployment for transformer on FPGA.
- Experimental results demonstrate that FLAME achieves 84.4% accuracy of expert prediction with merely two expert caches on-chip. In comparison with CPU and GPU, FLAME achieves 4.12× and 1.49× speedup, respectively.

2 BACKGROUND AND MOTIVATION

2.1 Vanilla MoE (Mixture-of-Experts)

In actual data distributions, there exist numerous natural subsets, such as distinct domains, various languages and different modalities. When utilizing a single model for learning, particularly when the model's capacity is limited, these diverse subsets may introduce noise and disrupt the model's fitting process, leading to slow training and challenging generalization. MoE mechanism [7] addresses this problem by breaking the target task into subtasks and training an expert model for each subtask. As depicted in Figure 1(b), a gating network is employed to allocate output weights to each individual expert. For an input token t, if the output of the i^{th} expert is denoted as o^i, then the output of the MoE layer for this single token can be defined as $O^t = \sum_i p_i^t o_i^t$, where p_i denotes the weight allocated to the i^{th} expert by the gating network.

2.2 Sparse MoE

Transformer-based models quickly integrate MoE mechanism to enhance generalization capabilities. Typically, the standard Feed Forward Network (FFN) layer is substituted with a MoE layer for some transformer blocks (odd for switch transformers [13], see Figure 1(a)). However, the vanilla MoE mechanism weights all experts during computation, placing an obstacle for the scaling of model parameter size. Sparse MoE [11] offers an effective solution towards this problem. Sparse MoE utilizes a sparsely activated gating network, choosing top-k experts instead of weighting all experts during inference, which allows for the potential expansion of the model parameter size (see Figure 1(c,d)). Sparse MoE has been widely adopted in mainstream transformer-based models including GLaM [6], Switch Transformers [13] and GShard [3].

2.3 MoE Deployment on Edge Devices

The application of transformers [14] equipped with a sparse MoE mechanism proves to be considerably challenging for deployment on edge devices. For one thing, the incorporation of the multi-expert feature introduces a substantial number of expert weights, presenting a significant challenge due to limited on-chip storage capacity on edge devices. For another, owing to the sparsely activated nature of the mechanism, the activated expert index for the decoder cannot be determined until the routing network completes its computations. This introduces a noteworthy memory access time overhead.

As a pioneering effort, Sarkar et al. [12] investigate the deployment of MoE-based transformer models on Field-Programmable Gate Arrays (FPGAs) and introduce an expert-by-expert computation strategy to maximize the reuse of loaded experts. However, this approach is only applicable to non-autoregressive models like Vision Transformers (ViT) and does not address the latency-intensive decoder part. Furthermore, EdgeMoE [9] proposes expert prediction based on expert activation path analysis, yet its predictive capability is deemed insufficient for edge devices. Huang [4] and Hwang [8] employ expert buffering, but their primary focus is on GPUs with relatively ample on-chip storage, making their research findings less suitable for resource-constrained devices such as FPGAs.

To address both limited on-chip storage and insufficient expert prediction, inspired by [9], we propose **FLAME** to comprehensively explore the efficient deployment of MoE mechanism for transformer models on FPGA by leveraging MoE sparsity.

3 FLAME OVERVIEW

Standing as a framework tailored for accelerating MoE mechanism for transformer on FPGA, FLAME fully exploits MoE sparsity. MoE sparsity manifests in two key dimensions: (1) Expert weight sparsity; (2) Expert activation sparsity. FLAME effectively leverages

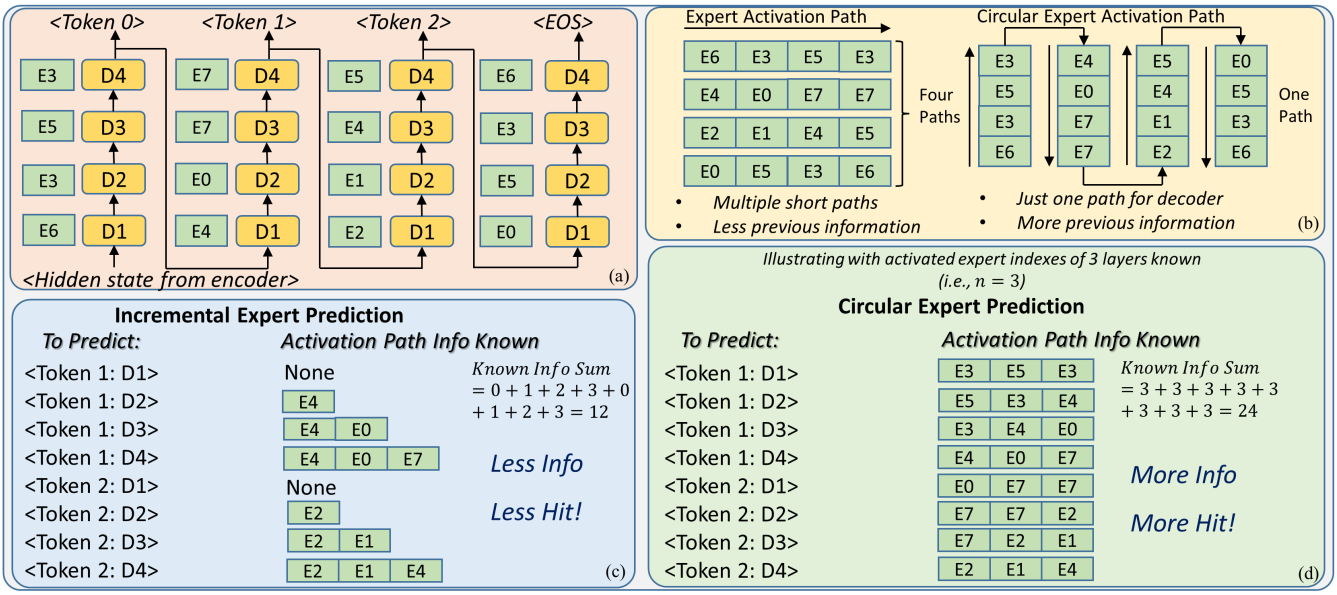

Figure 2: (a) Illustration of the dataflow of a typical MoE-based transformer decoder. (Assuming each MoE layer routes top-1 expert from a total of 8 experts.) E represents the activated expert index, and D represents the MoE layer index. For simplicity, this figure does not depict transformer layers other than MoE layers. (b) Traditional expert activation path *vs.* circular expert activation path. (c) Illustration of incremental expert prediction. (d) Illustration of circular expert prediction.

both of the sparsity throughout the offline and online stages of its operation (as depicted in Figure 3).

Offline stage: FLAME preprocesses the model to get both pruned experts and expert activation profile. Tackling expert weight sparsity, FLAME incorporates an N:M pruning approach to facilitate expert weight pruning, as detailed in Section 4. Addressing the challenges posed by expert activation sparsity, FLAME undertakes a comprehensive expert activation pattern modeling, leading to the proposal of a practical expert prediction strategy (expounded in Section 5). In a concerted effort to co-optimize both sparsity, FLAME introduces pruning-aware expert buffering (discussed in Section 6).

Online stage: FLAME instantiates a preload/computing pipeline. Armed with the expert activation profile derived during the offline stage, FLAME pre-determines the experts to be fetched with remarkable accuracy. Subsequently, FLAME preloads the pruned experts into the on-chip expert cache prior to the routing network.

4 LEVERAGING EXPERT WEIGHT SPARSITY

Expert weight is inefficient in the inference process, but accounts for a substantial share of the total parameters in MoE-based transformer models. Due to the sparsely activated nature of MoE-based transformer, each token typically activates just top-1 or top-2 experts in each MoE layer during the inference process, which is negligible compared to large expert numbers per MoE layer (128-2048 for Gshard [3], 32-256 for GLaM [6], and 2-2048 for switch transformers [13]). Multiple unselected experts remain "cold" until chosen by a new token. However, expert weight takes up a substantial share of the total parameters in MoE-based transformer models, imposing a significant burden on on-chip parameter storage. As Figure 4(a) illustrated, when there are 8 experts per MoE layer, expert weight accounts for 76.15% of the total weight, and the percentage goes up to 98.98% when there are 256 experts per MoE layer. Due to the heavy but cold nature of expert weight, it is an ideal choice for model sparsity. Pruning expert weights not only enhances model inference efficiency but also allows for the allocation of spare resources to enhance caching capabilities.

N:M pruning is optimal for leveraging expert weight sparsity. N:M pruning [2] is an optimal approach for leveraging expert weight sparsity. N:M pruning typically reserves N elements out of every M elements within the entire weight map. N:M pruning can serve as the tradeoff between column-balanced structured pruning [5] and unstructured pruning [10] (refer to Figure 1(e)): It attains superior inference accuracy compared to coarse-grained column-balanced structured pruning, while simultaneously achieving substantial savings in index storage when contrasted with unstructured pruning. Experimental results (see Figure 4(b)) demonstrate that, among the three commonly adopted pruning strategies for weight sparsity exploration, N:M pruning exhibits relatively superior performance. Under the same 50% sparsity, N:M pruning effectively reduces twice the index storage for expert weights in the simplified scenario, incurring only a negligible 0.85% accuracy loss compared to unstructured pruning. In comparison with column-balanced structured pruning, the fine-grained N:M pruning yields a remarkable 37.6% improvement in accuracy.

5 SETTLING EXPERT ACTIVATION SPARSITY

For MoE-based transformers, as discussed in Section 1, a routing network is commonly employed to dynamically determine which expert to activate for each token in each layer on-the-fly (*i.e.*, sparsely activated expert). However, this approach poses challenges for edge deployment. On one hand, edge devices typically lack sufficient on-chip memory to accommodate all expert weights, which indicates the necessity of device storage hierarchy (on-device cache vs. off-device storage). On the other hand, the activated expert index can be only determined after the routing network completes

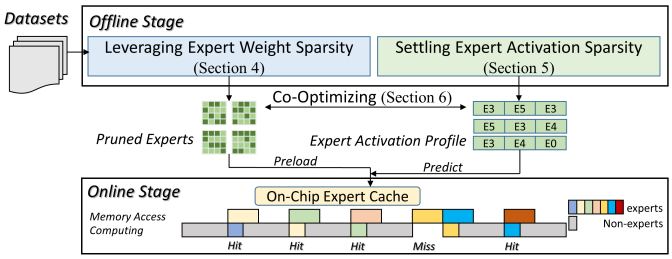

Figure 3: System architecture of FLAME.

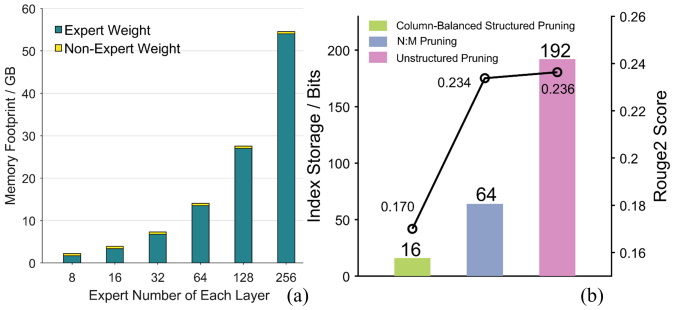

Figure 4: (a) Memory footprint comparison of expert weight and non-expert weight. (b) Accuracy and index storage comparison of three pruning strategies. For vivid illustration, index storage reflects the scenario of Figure 1(e) (Model: switch-base-8).

its computations. As a consequence, the FFN must just wait until the expert weights are loaded from off-device storage, leading to redundant time wastage.

Expert prediction offers an effective solution to these challenges. By predicting the activated expert index prior to the routing network's completion of computations, we can proactively prefetch the predicted expert from off-device storage into the on-device cache. With a substantial cache hit ratio, we can effectively save time on the weight-loading, thereby reducing the inference latency of MoE-based transformers.

5.1 Expert Activation Path Analysis

The concept of expert activation path refers to the sequential activation of experts within different MoE layers for a single token (as depicted in Figure 2(a,b)). For instance, in a scenario with two MoE layers, if token 1 activates expert x in the first MoE layer and expert y in the second MoE layer, the expert activation path can be represented as $[x, y]$.

Expert activation paths are highly skewed in MoE-based transformers. Our study finds that among 262144 possible paths for switch transformers [13] with 8 experts per MoE layer, only 11693 different paths are activated in the text summarization task with samsum dataset [1]. Furthermore, the activation frequency of each valid path is also highly skewed among these 11693 paths, with a substantial number of tokens concentrating on a few dominant "hot" paths (as depicted in Figure 5(a)). Figure 5(b) further illustrates this skewed distribution. Out of a total of two hundred thousand valid tokens passing through six MoE layers sequentially, 15, 110 tokens follow the $No.1$ path, and 10, 239 follow the $No.2$ path (see Figure 6(b)). Remarkably, the top 10 paths, accounting for merely 0.2% of all valid paths, are chosen by 28.7% of the valid tokens. In

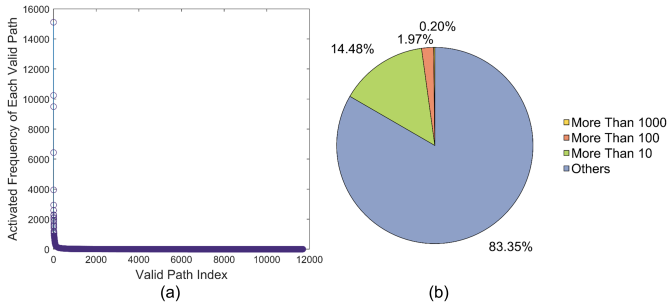

Figure 5: (a) The activation frequency of each valid path. The majority of valid paths experience infrequent activation, whereas a small fraction undergo notably frequent activation. (b) Path activation frequency distribution. A merely 0.20% **of valid paths are each activated by exceeding** 1000 **tokens, while** 83.35% **of valid paths are each activated by no more than** 10 **tokens. (Model: switch-base-8; Dataset: samsum)**

contrast, 83.35% of the paths are activated by fewer than 10 tokens. This exciting feature indicates the feasibility of expert prediction via expert activation path modeling.

Expert prediction can be achieved by expert activation path modeling. Given the knowledge of activated expert indexes in the $n-m, n-m+1, ..., n-1$ MoE layers, we can estimate the activation probability of expert x in the n^{th} MoE layer with great confidence, formulated as $P(E_n = i|E_{n-m}, E_{n-m+1}, ..., E_{n-1})$, where i denotes the index of the expert, n refers to the index of the MoE layer to be predicted, and m represents the number of prior MoE layers with known activation. We first investigate the activation probability of each expert in the n^{th} MoE layer when we only have knowledge of the activated expert index in the $(n-1)^{th}$ MoE layer (i.e., $m = 1$). As illustrated in Figure 6(a), the activation probability of different experts in the n^{th} MoE layer is highly skewed. For example, expert 3 is most likely to be activated in the n^{th} MoE layer if expert 5 is activated in the $(n-1)^{th}$ MoE layer, with $P(E_n = 3|E_{n-1} = 5) = 53.6\%$. Subsequently, we conduct a thorough analysis of the accuracy of expert prediction with various numbers of prior known MoE layers. The results consistently demonstrate that accuracy improves steadily as the number of prior known MoE layers (*i.e.*, m) increases.

5.2 Proposed Expert Prediction Strategy

Inefficiency of Incremental Prediction in the Early MoE Layers Due to Information Imbalance. From the above expert activation path analysis, we can naturally derive incremental prediction [9], which refers to predicting the activated expert index of the n^{th} MoE layer based on the information of all n previous MoE layers for the same token (refer to Figure 6(a)). Specifically, to predict the activated expert index for the 3^{th} MoE layer of a given token, information of the 0^{th}, 1^{th}, and 2^{th} MoE layers is utilized to formulate expert activation paths, and if we would like to predict the activated expert index for the 5^{th}, which is the last MoE layer for a single token in decoder, we can utilize all previous 5 layers to conduct expert activation path analysis, and that is the exact concept of *"incremental"*. However, when predicting the early MoE layers, inefficiency arises due to the absence of information from previous layers. For instance, the 0^{th} MoE layer lacks any preceding

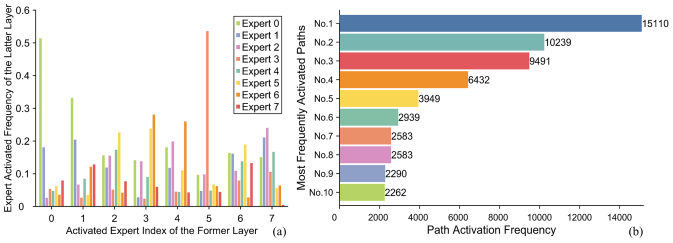

Figure 6: (a) The activation probability of different experts in the n^{th} MoE layer, given the knowledge of the activated expert index in the $(n-1)^{th}$ MoE layer. (b) Top 10 most frequently activated paths among all valid paths. The top 1 path is activated by 7.5% of the total number of tokens tested.

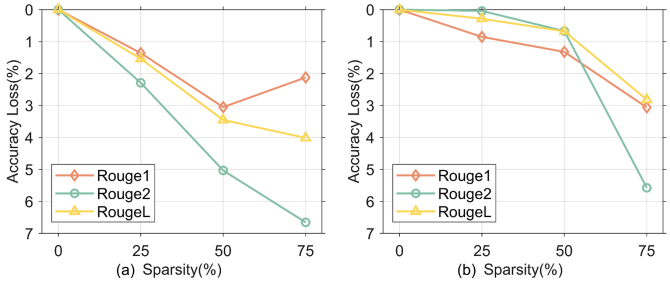

Figure 7: Accuracy loss of N:M expert weight pruning under different sparsity ratios. (a) On switch-base-8 model. (b) On switch-base-16 model. (Dataset: samsum)

layers for a given token, leading to failing to formulate any expert activation path modeling to serve for its prediction. This results in a considerable accuracy loss of expert prediction and thus impedes further reduction of memory access time overhead.

Circular Expert Prediction to Address Information Imbalance. To address the information imbalance among different MoE layers, we propose a circular expert prediction(*CEPR*) strategy (see Figure 2 (b,d)). Unlike incremental prediction (see Figure 2 (b,c)), which just treats activated expert indexes from the 0^{th} to the 5^{th} MoE layer for the given token as an expert activation path, we instead treat the activated expert indexes during the whole decoding process of an example as a continuous expert activation path and take information of the last few layers for the former token into account. This approach is justified by the typical combination of partial output sentences from the decoder and features from the encoder output in the cross-attention layer of transformers decoders. Consequently, the last few layers for the former token are highly relevant to the first few layers for the current token in terms of activated expert indexes. Leveraging information from the former token, we enhance the accuracy of activation-path-based expert prediction.

6 CO-OPTIMIZING MOE SPARSITY

Expert activation pattern modeling, as detailed in Section 5, introduces the concept of *expert buffering*. This approach preloads expert weights from external storage to on-chip cache before the completion of computation by the routing network, thereby mitigating memory access time overhead. Meanwhile, the application of N:M expert weight pruning (as discussed in Section 4) effectively diminishes on-chip storage while maintaining model inference accuracy. In pursuit of more expeditious memory access and enhanced efficiency, we integrate expert buffering with N:M expert weight pruning to introduce *pruning-aware expert buffering*. This innovative strategy enables the incorporation of more caches on-chip by optimizing the pruning ratio. While the on-chip storage capacity remains constant, a higher sparsity of the expert weight will significantly economize on-chip storage, leading to a notable increase in the number of caches. The expansion of on-chip expert weights is anticipated to not only reduce memory access time but also enhance the accuracy of expert predictions (refer to Section 5).

7 EVALUATION

7.1 Evaluation Setup

Devices. Software evaluation experiments are conducted on the Intel(R) Xeon(R) Gold 5218R CPU and NVIDIA 3090 GPUs with a batch size of $1-4$. On the hardware side, our design is synthesized via Vivado 2022.2 with the Xilinx Alveo U200 FPGA serving as the evaluation platform.

Models. We select *switch-base-8* model (with 8 experts each MoE layer) and *switch-base-16* (with 16 experts each MoE layer) from switch transformers, which are prevalent types of MoE-based transformers, as our evaluation models without losing generality.

Datasets and Metrics. We use *samsum* dataset to validate the effectiveness of both N:M pruning and our proposed circular expert prediction(*CEPR*) strategy. We choose ROUGE (including ROUGE-1, ROUGE-2, and ROUGE-L) as our evaluation metrics.

7.2 Software Evaluation

Pruning Accuracy. We conduct N:M expert weight pruning on different MoE-based transformer models under various pruning ratios (see Figure 7). Experimental results indicate that N:M expert weight pruning effectively preserve inference accuracy when sparsity increases.

Prediction Accuracy. We conduct assessments of expert prediction accuracy with *CEPR* and incremental prediction. A baseline comparison is established with random prediction, wherein the prediction result is generated by randomly picking an expert. Theoretically, in a model featuring 8 experts per MoE layer, the accuracy of random prediction stands at 12.5%. Experimental results (see Figure 8(a)) demonstrate that our proposed circular expert prediction (*CEPR*) yields a more balanced accuracy of expert prediction across different MoE layers in the transformer decoder compared to incremental prediction. Overall accuracy outperforms that of incremental prediction by up to 37.8% (assuming the utilization of the former six layers for expert activation path modeling). With approximately the same index storage overhead, *CEPR* still maintains a superior overall accuracy over incremental prediction by 9.86%. In comparison with the baseline, *CEPR* achieves 3.42× ∼ 6.02× improvements of prediction accuracy.

7.3 Hardware Evaluation

Cache Hit Ratio. We evaluate our proposed *CEPR* strategy under varying cache capacities and known layer numbers (see Figure 8(b)). For fair comparisons, we disable hardware pipelining to mitigate its potential impact on cache hit ratios. The results demonstrate that *CEPR* exhibits significant performance in expert prediction. Specifically, *CEPR* achieves cache hit ratios of 42.8% ∼ 75.8% and 71.4% ∼ 86.5% for the switch transformer decoder with cache capacities set at 1 and 5, respectively. When compared to the SOTA MoE accelerator EdgeMoE [9], *CEPR* outperforms it with 1.52× ∼ 2.71×

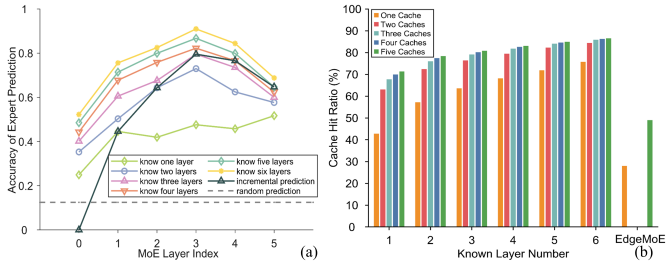

Figure 8: (a) Accuracy of expert prediction for the k^{th} MoE layer ($k = 0, 1, 2, 3, 4, 5$) with circular expert prediction (with activated expert index of n layers known, $n = 1, 2, ..., 6$) and incremental prediction. (b) Expert weight cache hit ratios with various cache capacities and different numbers of known layers. EdgeMoE [9] does not report cache capacities of 2, 3 and 4.

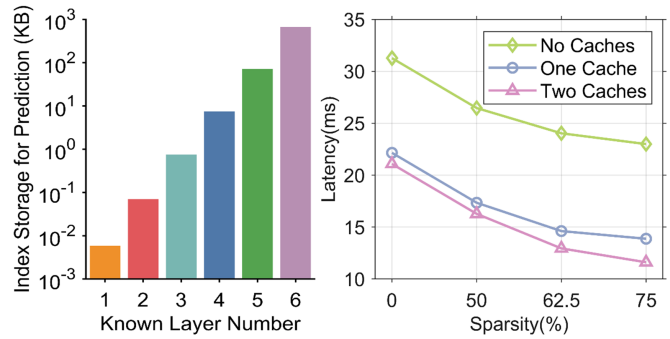

Figure 9: (a) Index storage for expert prediction with the knowledge of n preceding layers. (b) Per-token inference latency with various cache capacities and different sparsity ratios.

and 1.46× ∼ 1.76× enhancement (approx.) in cache hit ratios. *CEPR* proves particularly advantageous for edge devices with highly-constrained on-chip storage. With merely 1-2 expert weight caches on-chip –a configuration commonly feasible for edge devices such as FPGA –*CEPR* achieves up to 75.8% and 84.4% accuracy in expert prediction. This implies the feasibility to correctly preload most of the experts before the determination of the activated expert indexes. By implementing proper pipelining, we can hide a substantial portion of the memory access time, thereby contributing to a significant enhancement in inference speed, which will be further explained afterwards.

Index Storage for Prediction. We conduct an assessment of index storage for expert prediction, incorporating knowledge of the preceding n layers, where n varies from 1 to 6. The corresponding results are illustrated in Figure 9(a). In practical implementation, rather than employing information from all six preceding layers to predict the next activated expert index, a judicious approach involves selecting n as 2 or 3. This strategic choice yields a sufficiently high cache hit ratio, surpassing 70%, while concurrently limiting the index storage requirement to less than 2KB.

Latency Comparison. We conduct per-token inference latency measurements for the switch transformer decoder, as illustrated in Figure 9(b). To better validate FLAME's efficacy, we selectively excludes all other layers, retaining only the MoE layers in our latency assessments. The experimental results demonstrate the effectiveness of FLAME. By leveraging expert weight sparsity (as evidenced by increasing the sparsity ratio in Figure 9(b)), we achieve a reduction in per-token inference latency by 1.36× ∼ 1.82×. By settling expert activation sparsity through the utilization of the *CEPR* strategy for caching experts on-chip, we reduce per-token inference latency by 1.48× ∼ 1.98×. In comparison with CPU and GPU (see Table 1), FLAME exhibits a considerable speedup, achieving 4.12× and 1.49× improvements, respectively.

Table 1: Performance Comparison of Different Platforms

Platform	Configuration	Latency(ms)	Speedup
CPU		47.88	1.00×
GPU		17.28	2.77×
Proposed FLAME	baseline	31.27	1.53×
	with N:M	22.98	2.08×
	with *CEPR*	21.08	2.27×
	with both	11.61	4.12×

8 CONCLUSION

We propose **FLAME**, the first MoE accelerating framework which fully exploits MoE sparsity for transformer on FPGA. To leverage expert weight sparsity, we effectively incorporate N:M pruning approach to enhance computational efficiency. To settle the challenge posed by expert activation sparsity, we propose the circular expert prediction (*CEPR*) strategy to alleviate the memory access time overhead associated with expert weights. Experimental results demonstrate that FLAME achieves 84.4% accuracy of expert prediction with only two expert caches on-chip. In comparison with CPU and GPU, FLAME achieves 4.12× and 1.49× speedup, respectively.

REFERENCES

[1] Bogdan Gliwa et.al. 2019. SAMSum Corpus: A Human-annotated Dialogue Dataset for Abstractive Summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*. Association for Computational Linguistics, Hong Kong, China, 70–79. https://doi.org/10.18653/v1/D19-5409
[2] Chao Fang et.al. 2022. An algorithm–hardware co-optimized framework for accelerating N: M sparse transformers. *VLSI* 30, 11 (2022), 1573–1586.
[3] Dmitry Lepikhin et.al. 2020. GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding. arXiv:2006.16668 [cs.CL]
[4] Haiyang Huang et.al. 2023. Towards MoE Deployment: Mitigating Inefficiencies in Mixture-of-Expert (MoE) Inference. *arXiv preprint arXiv:2303.06182* (2023).
[5] Hongwu Peng et.al. 2021. Accelerating Transformer-based Deep Learning Models on FPGAs using Column Balanced Block Pruning. In *ISQED*. 142–148. https://doi.org/10.1109/ISQED51717.2021.9424344
[6] Nan Du et.al. 2022. GLaM: Efficient Scaling of Language Models with Mixture-of-Experts. arXiv:2112.06905 [cs.CL]
[7] Robert A. Jacobs et.al. 1991. Adaptive Mixtures of Local Experts. *Neural Computation* 3, 1 (1991), 79–87. https://doi.org/10.1162/neco.1991.3.1.79
[8] Ranggi Hwang et.al. 2023. Pre-gated MoE: An Algorithm-System Co-Design for Fast and Scalable Mixture-of-Expert Inference. *arXiv preprint arXiv:2308.12066* (2023).
[9] Rongjie Yi et.al. 2023. Edgemoe: Fast on-device inference of moe-based large language models. *arXiv preprint arXiv:2308.14352* (2023).
[10] Song Han et.al. 2015. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems* 28 (2015).
[11] Shazeer Noam et.al. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538* (2017).
[12] Sarkar Rishov et.al. 2023. Edge-MoE: Memory-Efficient Multi-Task Vision Transformer Architecture with Task-level Sparsity via Mixture-of-Experts. *arXiv preprint arXiv:2305.18691* (2023).
[13] William Fedus et.al. 2022. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. arXiv:2101.03961 [cs.LG]
[14] Yueyin Bai et.al. 2023. LTrans-OPU: A Low-Latency FPGA-Based Overlay Processor for Transformer Networks. In *FPL*. 283–287. https://doi.org/10.1109/FPL60245.2023.00048