FAGE: An FPGA Accelerator for High-Throughput Graphormer Encode Processing

Min-Gwon Song*, Min-Young Lee*, Soomin Rho*, Bohyeon Kang†, and Ki-Seok Chung*
*Department of Electronic Engineering, Hanyang University, Seoul, South Korea
†Department of Nanoscale Semiconductor Engineering, Hanyang University, Seoul, South Korea
Email: {mgsong1005, lmo970901, smrho, kbh2247, kchung}@hanyang.ac.kr

***Abstract*—Graph Neural Networks (GNNs) are a type of neural network capable of processing graph data structures. Graphormer, a Transformer-based GNN, has achieved state-of-the-art performance on molecular property prediction tasks. However, virtual screening over large-scale chemical spaces requires extremely high throughput. While the multi-head attention and the feed-forward layers of Graphormer can be accelerated by existing Transformer accelerators, attempts to accelerate the graph-specific encoding layer have not yet been made actively. Hence, we propose FAGE, an FPGA accelerator specialized for the encoding layer of Graphormer. Implemented on a ZCU111, FAGE achieves an average 2× speedup over an RTX 4090 GPU.**

***Index Terms*—GNN, FPGA, Graphormer, Encoding layer**

I. Introduction

Virtual screening is a key stage in drug discovery that searches large-scale chemical spaces to identify promising hit compounds. As chemical spaces expand toward the trillion-molecule scale, high-throughput molecular inference becomes increasingly important. Meanwhile, Graph Neural Networks (GNNs) are a type of neural network capable of processing graph data structures. Graphormer [1] is a Transformer-based GNN that has achieved high accuracy on molecular graph prediction tasks by adopting the Transformer architecture [2] and incorporating graph-specific structural encodings.

Although Graphormer builds on the Transformer architecture, it differs from conventional Transformer-based models by incorporating three graph-specific structural encodings: (i) centrality encoding, which augments node features with degree information, (ii) spatial encoding, and (iii) edge encodings, which inject graph topology and bond information as attention biases. While multi-head attention and feed-forward layers can benefit from existing Transformer accelerators, these encoding operations are unique to Graphormer, and attempts to accelerate the graph-specific encoding layer have not yet been made actively. Moreover, these encoding operations are non-trivial, accounting for 38%, 18%, and 23% of the inference latency on MolHIV, PCQM4Mv2, and ZINC, respectively [3]. These observations motivate a dedicated accelerator for the Graphormer's encoding layer. Profiling results show that massive lookup operations dominate the time spent processing the Graphormer's encoding layer. Hence, we propose an FPGA-based accelerator, FAGE, that exploits on-chip memory efficiently for lookup tables, and the proposed design achieves an average 2× speedup over a GPU baseline.

II. Background

Categorical node and edge attributes (e.g., atomic number or chirality for nodes, and bond type for edges) are first encoded by summing attribute-wise embedding lookups [3]:

$$x_n = \sum_{i=0}^{D_N-1} N_i[a_n[i]], \quad x_e = \sum_{i=0}^{D_E-1} E_i[a_e[i]], \tag{1}$$

where a_n and a_e are node and edge attribute vectors, respectively, and N_i and E_i are node, edge embedding tables for the i^{th} attribute, respectively.

Centrality encoding augments the node representation as:

$$h_i^{(0)} = x_n + z^-[\deg^-(n_i)] + z^+[\deg^+(n_i)], \tag{2}$$

where z^-, z^+ are centrality embedding tables indexed by the in-degree $\deg^-$ and out-degree $\deg^+$ of node n_i.

Graphormer further adds spatial encoding results and edge encoding results to the self-attention score as bias:

$$A_{ij} = \frac{(h_i W_Q)(h_j W_K)^T}{\sqrt{d}} + s_{ij} + c_{ij}, \tag{3}$$

where s_{ij} and c_{ij} denote the spatial and edge encoding biases, respectively. The spatial encoding bias is computed as

$$s_{ij} = b[\phi(n_i, n_j)], \tag{4}$$

where $\phi(n_i, n_j)$ denotes the shortest-path distance between nodes n_i and n_j, and b is a spatial pos embedding table.

The edge encoding term c_{ij} incorporates edge features along the shortest path:

$$c_{ij} = \frac{1}{N} \sum_{n=1}^{N} x_{e_n} (w_n^E)^T, \tag{5}$$

where $SP_{ij} = (e_1, \ldots, e_N)$ is the shortest path from node n_i to node n_j, x_{e_n} is the feature of the n^{th} edge on this path, and w_n^E is the corresponding distance weight.

III. Proposed Method

A. Access-aware On-Chip Lookup Table Mapping

Embedding table lookups are the core operations of the encoding operations. We therefore use on-chip memories aggressively as lookup tables, including distributed RAMs and BRAMs. However, storing the entire embedding tables in on-chip memory is infeasible due to their massive size. To address

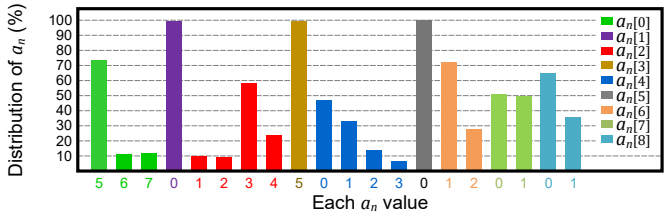


Fig. 1. Distribution of each a_n value.

this issue, we profile access frequencies of each entry during inference. The profiling results show that entry accesses are concentrated on a small subset of the entire entries, while many entries are never used. For example, as shown in Fig. 1, node attribute lookup in Eq. (1) accesses only 75 entries, although 174 entries are available. Similar access concentration is also observed in other encoding operations. Therefore, we map only the actually accessed entries onto on-chip memory instead of storing the entire embedding tables.

B. Dot-Product Unit Elimination

As shown in Eq. (5), the edge encoding involves many dot-product operations between the edge features and the weight vectors. By precomputing the dot-products between entries of E_i and w_n^E, the edge encoding can be replaced with lookup and accumulation operations. This eliminates the need for both a dedicated dot-product unit and buffers for storing w_n^E, thereby reducing compute resource overhead.

C. Hardware Architecture

Fig. 2 shows the FAGE architecture, consisting of three pipeline stages. Since the encoding layer is dominated by embedding lookups, accessing lookup tables from off-chip DRAM would incur high memory access latency. Therefore, the FAGE maps lookup tables onto the on-chip memory, enabling low latency and parallel lookups.

Stage 1 reads node attributes, edge attributes, and spatial positions (i.e, the shortest distance between each node pair) from DRAM through an RDMA and a Fetch unit writes them into dedicated Node, Edge, and Spatial buffers. Stage 2 performs the encoding operations by reading data from the corresponding buffers. The Node Encoder performs the node-attribute embedding lookup and the centrality encoding, as described in Eq. (1) and Eq. (2). The Edge Encoder performs the edge-attribute embedding lookup and the edge encoding, corresponding to Eq. (1) and Eq. (5), respectively. The Spatial Encoder generates spatial biases from the spatial positions, as shown in Eq. (4). To reduce the output data size, the spatial and the edge encoding results are accumulated before being stored. The encoded results are then written to the Encoded Output buffer. Stage 3 reads the encoded outputs through the Commit unit and writes them back to DRAM through a WDMA.

IV. Evaluation

We use Graphormer-base and compare our design with an NVIDIA RTX 4090 GPU on ZINC, PCQM4Mv2, and MolHIV datasets. We implemented FAGE on an Xilinx ZCU111 FPGA operating at 100 MHz.

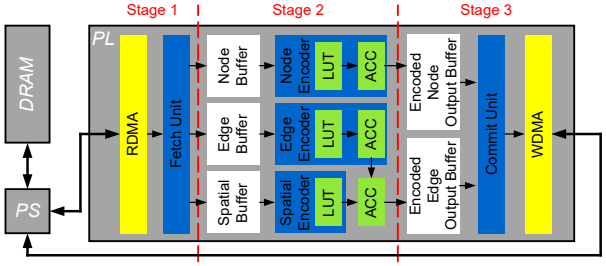


Fig. 2. FAGE architecture block diagram.

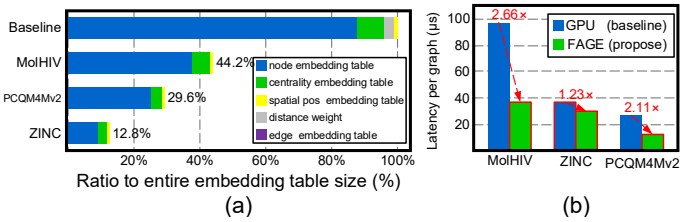


Fig. 3. On-chip embedding-table mapping and per-graph latency. (a) Runtime-accessed embedding entries mapped on chip. (b) Latency comparison between the GPU baseline and FAGE.

TABLE I
Resource usage on Xilinx ZCU111 FPGA.

	LUT	FF	BRAM	DSP
Used Resource	150154	143116	825	3
HW. Utilization	35.31%	16.83%	76.30%	0.07%

Table I reports the resource usage, and the estimated power consumption is 5.31 W. As shown in Fig. 3(a), FAGE maps only 44.2%, 29.6%, and 12.8% of the embedding table to on-chip memory for MolHIV, PCQM4Mv2, and ZINC, respectively. By placing the lookup tables in BRAM and distributed RAM, the FAGE, which has low lookup latency and high memory bandwidth, achieves an average 2× speedup over an RTX 4090 GPU as shown in Fig. 3(b).

V. Conclusion

We presented FAGE, an FPGA accelerator for the lookup-intensive Graphormer encoding layer, by mapping only runtime-accessed lookup entries to on-chip memory and replacing edge-encoding dot products with precomputed lookup tables. Our ZCU111 implementation achieves 2× higher speedup than an NVIDIA RTX 4090 GPU baseline.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean Government (MSIT) (RS-2024-00409492)

References

[1] Ying *et al.*, "Do transformers really perform bad for graph representation?" in *Proceedings of the 35th International Conference on Neural Information Processing Systems*, ser. NIPS '21, 2021.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17, 2017.

[3] W. Hu *et al.*, "Open graph benchmark: Datasets for machine learning on graphs," in *Proc. NeurIPS*, 2020.