

Anchor-and-Adapt: HLS QoR Prediction using Ground-Truth Seeding and Few-Shot Fine-Tuning

Gabriel C. Tavares*, Heitor C. de Andrade*, Fábio P. Itturriet†, and Gabriel L. Nazar*

**Institute of Informatics, Federal University of Rio Grande do Sul, Porto Alegre, Brazil*

{gctavares, hcandrade, glnazar}@inf.ufrgs.br

†*Dept. of Automation and Electrical Eng., Federal University of Technology – Paraná, Curitiba, Brazil*

fabioitturriet@utfpr.edu.br

Abstract—Graph Neural Networks (GNNs) have emerged as powerful tools for guiding Design Space Exploration (DSE) in High-Level Synthesis (HLS), but they face a critical trade-off: models that base inference on pre-HLS inputs are often inaccurate, while accurate models based on post-HLS inputs are too slow for iterative exploration. This paper introduces a novel framework that targets this dilemma. Our approach centers on an *anchor-based graph representation*, where a single, ground-truth hardware implementation is used to seed a design graph with rich, post-implementation data. A heterogeneous GNN is then trained to predict the *QoR delta* caused by applying new optimization directives to this anchor, enabling rapid and high-fidelity estimation without re-running the HLS toolchain. Furthermore, we propose a dual-strategy framework that offers both a quick-setup *Ensemble Model* for general use and a specialized, high-accuracy *Few-Shot Fine-Tuned Model* for maximum per-kernel precision. Our results demonstrate that this methodology achieves significantly higher accuracy than pre-HLS methods, reducing the prediction error by 35-50% while maintaining a comparable exploration speed. More remarkably, our few-shot specialized model surpasses the prediction accuracy of post-HLS methods by more than 10% without incurring their restrictive per-point runtime cost.¹

Index Terms—High-Level Synthesis (HLS), Design Space Exploration (DSE), Machine Learning, Graph Neural Networks (GNNs), Heterogeneous Graphs, Few-Shot Learning, Ensemble Methods

I. INTRODUCTION

High-Level Synthesis (HLS) accelerates hardware development by automatically translating behavioral specifications written in high-level languages, such as C/C++, into Register-Transfer Level (RTL) designs. This process is guided by synthesis directives—pragmas that allow designers to explore a vast space of functionally equivalent microarchitectures without modifying the original source code. The choice of directives is critical, as different combinations can produce designs with drastic variations in area, latency, and power consumption. While the HLS design flow is significantly more productive and less error-prone than traditional RTL development, the quality of the resulting hardware is intrinsically dependent on selecting an adequate set of these directives.

Navigating this vast design space to find optimal directive combinations is a major challenge due to the exponential

growth of the search space. Moreover, the prohibitive runtime of the hardware synthesis and implementation processes, required for accessing ground-truth Quality of Results (QoR) metrics, combined with the often poor predictive accuracy of HLS tools, introduces a significant obstacle on the development of exploration strategies. Driven by the need for efficient exploration on this design space, numerous Design Space Exploration (DSE) frameworks and predictive QoR models have been proposed in the literature [1], [2]. In recent years, the adoption of advanced models like Graph Neural Networks (GNNs) has shown great promise, as they can effectively learn from the rich, graph-structured representation of hardware designs. This has led to a notable improvement over prior DSE heuristics [3], [4].

Nevertheless, even these state-of-the-art models have not fully resolved a critical dilemma in this domain: methods that rely on pre-HLS information are fast but often inaccurate, while those using post-HLS data are accurate but prohibitively slow for large design spaces. Post-HLS methods enable faster training and higher accuracy by providing the model with data across the entire HLS workflow [4]–[6]. However, this dependency on post-HLS results creates a critical bottleneck for DSE, where hundreds or thousands of design points might be evaluated. On the other hand, pre-HLS methods enable extremely efficient inference by relying only on the pre-HLS LLVM Intermediate Representation (IR) and user directives [7]–[10]. While this approach is fast, its reliance on purely high-level information makes the GNN task significantly harder, resulting in considerably lower predictive accuracy when compared to the aforementioned post-HLS methodologies that provide the model with richer, low-level information about the kernel.

To address this trade-off, we introduce a novel DSE framework that provides quick inferences without sacrificing accuracy. Our approach achieves this balance by learning the QoR delta from a single, pre-implemented reference design, combining the speed of pre-HLS analysis with the fidelity of post-implementation data. Additionally, our work leverages two powerful ML techniques: *Ensemble Learning* and *Few-Shot Fine-Tuning*. We utilize an ensemble of models to provide highly accurate predictions on unseen kernels, and we introduce a novel fine-tuning methodology for efficient kernel-specific specialization. Thus, the proposed framework offers

This work was supported by the National Council for Scientific and Technological Development (CNPq).

¹Source code available at: <https://github.com/gct02/Anchor-and-Adapt>

two modes of operation, providing the flexibility to navigate the accuracy-efficiency trade-off based on project needs.

Furthermore, this work extends the scope of model-based DSE by effectively modeling two high-impact but traditionally challenging directives: `loop_merge` and `loop_flatten`. These directives are often omitted from automated exploration due to the complex, topological transformations they impose on the design’s graph structure, making their effects difficult to predict. However, their proper application can enable high-performance designs, as they can significantly improve data locality and unlock superior pipelining opportunities. By demonstrating that our anchor-based graph representation can accurately capture the non-local effects of these directives, we expand the searchable design space to include more complex and powerful optimization pathways.

In summary, the main contributions of this paper are:

- We introduce an anchor-based DSE framework that combines the efficiency of pre-HLS analysis with the fidelity of post-implementation data by learning the QoR delta from a single ground-truth implementation.
- We present a dual-strategy framework that offers both a quick-setup *Ensemble Model* for rapid deployment on DSE tasks and a specialized, *Few-Shot Fine-Tuned Model* for maximum per-kernel accuracy, accommodating different user needs.
- Experimental results show that our methodology achieves a 35-50% reduction in prediction error over state-of-the-art pre-HLS QoR estimators. Moreover, with fine-tuning, it surpasses the accuracy of post-HLS approaches by more than 10% with a substantially lower runtime.

II. BACKGROUND & RELATED WORKS

Early analytical approaches to automating DSE for HLS, while appealing for their formal structure, struggle to capture the full complexity of the modern HLS-to-bitstream toolchain, which employs a cascade of internal heuristics and proprietary optimizations. Consequently, data-driven Machine Learning (ML) methods have become the state-of-the-art due to their superior adaptability and generalization power in modeling this complex behavior [11]–[13].

Among these data-driven techniques, Graph Neural Networks (GNNs) have recently emerged as the dominant approach. HLS designs, represented as Control-Data Flow Graphs (CDFGs), are inherently graph-structured data. Unlike other models that require flattening this rich topological information into a simple feature vector, GNNs are purpose-built to learn directly from the connectivity and structure of the graph. Through a process of message passing, they can effectively model the complex dataflow, control flow, and hierarchical relationships between design components, leading to more accurate and generalizable predictions [3], [4].

While many GNN-based estimators have been proposed, they can be broadly categorized based on when in the HLS workflow they extract their input data: pre-HLS or post-HLS. Each approach represents a different point in the fundamental trade-off between inference speed and predictive accuracy.

A. Pre-HLS QoR Estimation

This category includes models that prioritize speed by operating on the design before the often time-consuming HLS process begins. The primary advantage of pre-HLS estimation is its speed, which enables the rapid evaluation of thousands of design points [7]–[10]. A state-of-the-art example is HARP [7], which uses a GNN to make predictions based directly on the pre-HLS LLVM IR and user directives. By learning from this data in a high abstraction level, HARP avoids the HLS runtime bottleneck entirely. However, this reliance on abstract, pre-HLS information means the model has no visibility into the complex, downstream effects of HLS scheduling and resource binding. As a result, the accuracy of such models can be limited, particularly for complex and unfamiliar kernel architectures where the behavior of HLS tools is less predictable.

B. Post-HLS QoR Estimation

To achieve higher accuracy, other methods incorporate information generated during or after the HLS process. Post-HLS models achieve state-of-the-art accuracy by constructing their input graphs from the detailed data available after HLS has run [4]–[6]. For instance, HGBO-DSE [5] builds its graph using the scheduled Control-Data Flow Graph (CDFG) and other low-level reports from the HLS tool. This provides the GNN with a rich, hardware-aware representation that is much closer to the final RTL, allowing it to make highly accurate predictions. The fundamental drawback of this approach is its prohibitive cost for DSE; since the HLS tool must be executed for each design point to generate the input graph, it severely impacts the time required to explore large design spaces, which may require hundreds to thousands of inferences.

III. ANCHOR-AND-ADAPT METHODOLOGY

Our work is positioned as a novel hybrid approach, aiming to combine the benefits of both pre- and post-HLS methods. To address the challenge of providing fast yet accurate QoR predictions, we developed a novel methodology centered on a single ground-truth *anchor instance*. The process begins by generating this anchor—a “pure” implementation of the target kernel—from which we construct a richly annotated graph. This graph serves as the input to a powerful hierarchical GNN, which is pre-trained on a diverse set of kernels and then rapidly fine-tuned for the specific target using a few-shot approach. Thereby, the resulting specialized model can then guide a DSE heuristic with near-instantaneous, high-fidelity QoR estimates. This allows our framework to bridge the gap, providing the DSE heuristic with both the speed necessary for broad exploration and the accuracy required for reliable decision-making. Each component of this framework is detailed in the following sections.

A. Ground-Truth Anchoring

Our framework distinctively uses a single, fully implemented design—the *anchor instance*—to provide the model with low-level, ground-truth information for a specific kernel.

This anchor is generated by synthesizing and implementing the kernel with all user and automatic optimization directives disabled, creating a baseline implementation.

While hardware implementation is typically time-consuming, the cost of implementing this constrained anchor point is minimal. Moreover, disabling automatic optimizations ensures the results are easily traceable, allowing low-level hardware components to be reliably mapped back to their corresponding high-level constructs. This traceability is what enables the creation of our richly annotated, multi-level graph representation of the kernel.

B. Anchor Instance Construction

Initially, a single graph is built for the kernel using its anchor instance. The graph’s core topology and features are derived from the IR operator information (*.adb files) generated by the HLS tool. These files provide a detailed Control-Data Flow Graph (CDFG) based on a lowered Intermediate Representation (IR), which has been transformed by the HLS front-end to be more hardware-aware. The resulting CDFG is further enriched with HLS-specific information, including a mapping between source-level constructs and their final Register-Transfer Level (RTL) components. A simplified, high-level view of a graph created under this procedure, for a synthetic example, is shown in Fig. 1.

This initial representation, however, lacks certain high-level characteristics that are crucial for accurately modeling the impact of HLS directives. For example, the original multi-dimensional structure of arrays—information lost in the lowered IR—is fundamental for predicting the effects of `array_partition` directives. To recover this essential information, our framework executes a custom LLVM pass on an earlier, higher-level version of the IR. This pass extracts critical attributes for all arrays, such as the length of each dimension, which are then used to annotate the features of the corresponding nodes in the graph. Additionally, another custom LLVM pass analyzes the lowered IR to extract detailed loop metadata not directly available in the IR operator information, such as nesting depth and perfect nest information, which are important for modeling the effects of `loop_flatten`.

Next, the graph is annotated with ground-truth data from the post-implementation reports at two levels of granularity. The final, design-wide QoR metrics are added as global graph attributes, while per-module results are used to enrich the features of their corresponding nodes. An overview of the anchor instance construction pipeline is presented in Fig.2.

To evaluate a new design point, its directives are applied not by re-running the HLS tools, but by simply updating the features of the targeted nodes on this anchor graph. This transformation is nearly instantaneous and results in a unique graph representation for each design point.

Before being used by the model, all numerical features—at both the node and graph levels—undergo a standard pre-processing step. To handle heavily-tailed distributions, features such as resource utilization metrics are first transformed using

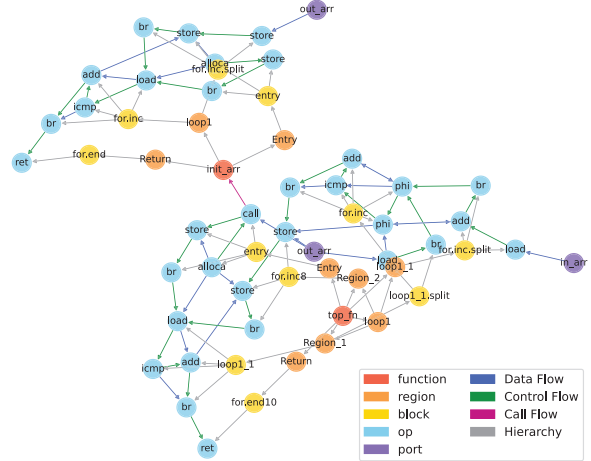


Fig. 1. A simplified visualization of our multi-level graph representation for an HLS kernel. This version of the graph is optimized for clarity, focusing on the essential structural and functional relationships, and includes only the major node and edge types.

a $\log(1 + x)$ function. Subsequently, all features are standardized by subtracting the mean and dividing by the standard deviation, with these statistics being computed exclusively from the training set to prevent data leakage.

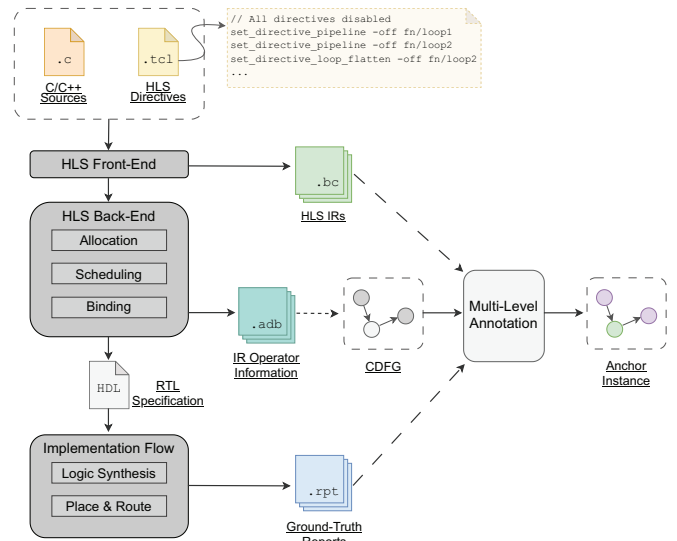


Fig. 2. Overview of our anchor instance creation process. A “pure” implementation is first generated by running the HLS and implementation flows with all directives disabled. The resulting CDFG and ground-truth reports are then used to construct the final, richly annotated anchor instance graph.

C. GNN Model Architecture

To address the challenge of HLS QoR estimation, we propose a model built upon heterogeneous GNNs [14]. While prior state-of-the-art methods have used homogeneous GNNs, these architectures apply uniform message-passing rules to all

nodes and edges. This approach can lead to semantic dilution in HLS CDFGs, where the distinct meanings of different node and edge types (e.g., control flow vs. data dependency) are muddled together, resulting in less powerful representations. In contrast, heterogeneous GNNs are purpose-built for such richly-typed graphs. They employ type-specific transformations and aggregation schemes, using different learnable weights for each unique type of interaction between nodes [15], [16]. This allows our model to capture the distinct semantics of the HLS design, leading to a more nuanced and accurate understanding of the underlying hardware structure.

Our estimator first processes the input graph through a stack of Heterogeneous Graph Transformer (HGT) [17] layers, with their outputs aggregated by a Jumping Knowledge Network (JKN) [18] to produce a set of rich node embeddings. To generate a graph-level summary, we employ a two-stage hierarchical pooling mechanism. First, a soft-attention aggregation layer [19] is applied independently to each node type, producing a single summary vector for each type (e.g., a summary of all `op` nodes, a summary of all `port` nodes, etc.). Second, these per-type summary vectors are fed into a multi-head self-attention layer [20], which learns to weigh their relative importance before averaging them into a final, fixed-size graph embedding.

In parallel, the global graph-level attributes are processed by a dedicated Multi-Layer Perceptron (MLP). Its output is then concatenated with the final graph embedding from the GNN, forming an information-dense vector that is fed into a set of independent MLP heads to predict each QoR metric. A high-level view of this architecture is provided in Fig. 3.

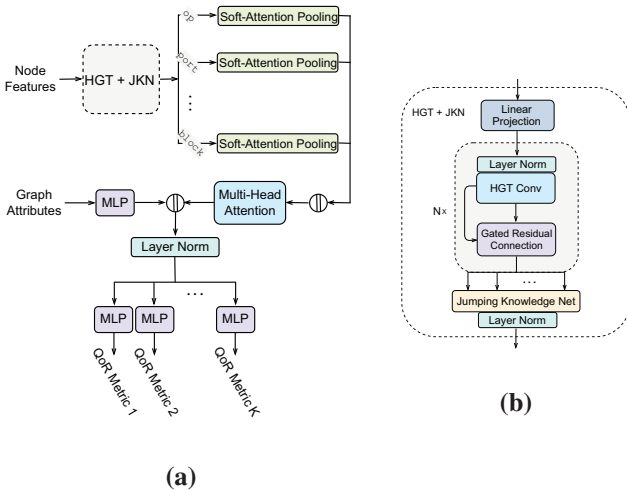


Fig. 3. Overview of the proposed hierarchical GNN architecture. (a) The top-level estimator, showing the two-stage pooling mechanism. (b) The internal structure of the *HGT-JKN* message-passing block.

D. Dual-Strategy Framework

To further explore the efficiency-accuracy tradeoff—the central point of this work—our framework offers two distinct modes of operation, each representing a different point

on the efficiency-accuracy spectrum. Our approach allows a designer to start with a quick-setup “Generalist” model, which requires only the anchor instance, and then progress to a high-fidelity fine-tuned “Specialist” model by paying an additional implementation cost. Both models are derived from a single, robust pre-trained model generated via Snapshot Ensemble and weighted averaging. Figure 4 shows an overview of this dual-strategy framework.

1) *Pre-trained Model Generation*: We first generate a robust pre-trained model by creating an ensemble from a single training run, a technique inspired by Snapshot Ensembles [21]. This is achieved using a cyclical learning rate schedule, which guides the optimizer to converge to multiple different local minima. We save a model “snapshot” at each of these minima. This training is performed with a dataset that completely excludes the kernel of interest and takes place only once.

These snapshots are then aggregated into a single model using a weighted average that prioritizes both performance and earliness, a concept related to Stochastic Weight Averaging (SWA) [22]. Each snapshot is assigned a score based on its validation error and its training stage, as described in eq. (1). After normalizing these scores, we compute the final averaged model parameters, as shown in eq. (2).

$$S_i = \frac{1}{\text{MAPE}_i + \epsilon} \cdot \alpha^{m_i} \quad (1)$$

$$\theta_{SWA} = \frac{\sum_{i=1}^k S_i \cdot \theta_i}{\sum_{j=1}^k S_j} \quad (2)$$

2) *The Generalist Model: Anchor Calibration*: The quick-setup “Generalist” model is designed for rapid estimation on new kernels. It is created by taking the pre-trained SWA model and applying a rapid *Anchor Calibration* procedure. This involves fine-tuning for a very small number of epochs on the anchor instance alone. This procedure allows the model to correct for any systematic prediction bias (e.g., a scaling error) for the new kernel while preserving its core generalization power.

3) *The Specialist Model: Few-Shot Fine-Tuning*: The high-fidelity “Specialist” model is created for high per-kernel accuracy. It also starts from the pre-trained SWA model but undergoes a more intensive few-shot fine-tuning process. To achieve meaningful specialization with minimal overhead, we employ a selective 10-point sampling strategy. The sampling set comprises: one anchor instance; five constrained, single-directive points to provide a clean, disentangled learning signal; and four complex, multi-directive points to capture realistic interactions between directives. The inclusion of the constrained points, which are significantly faster to implement, is key to creating a rich fine-tuning dataset without excessive runtime cost.

To maximize the information gained from our constrained 10-point fine-tuning set, we employed a data-driven sampling strategy. First, we identified which directives were the most challenging for the pre-trained model to predict. For this, we

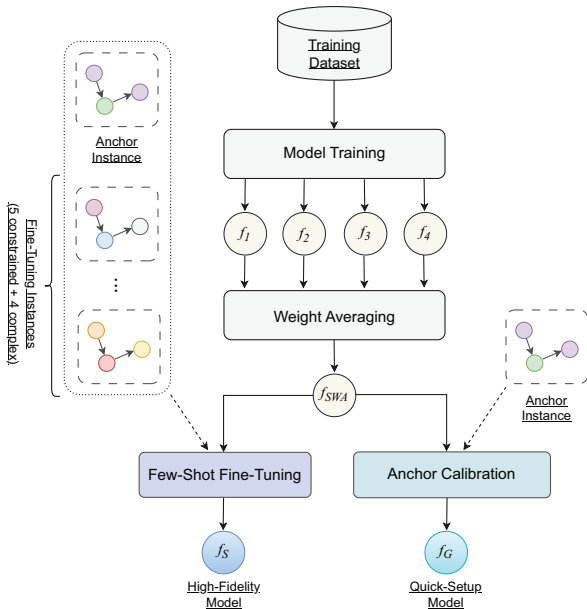


Fig. 4. High-level overview of our proposed dual-strategy framework, showing the pre-training, weight averaging, and the two final model specialization paths: *Anchor Calibration* and *Few-Shot Fine-Tuning*.

collected results from prior experiments and used SHAP [23] to analyze a surrogate model trained to predict our QoR estimator’s error based on a given set of directives.

This analysis revealed that high prediction errors were most strongly associated with `unroll` and `pipeline` directives on high-latency loops, and `array_partition` directives on interface or constant arrays. These critical directives were therefore used to generate our five constrained, single-directive points, providing the model with a clean signal to learn its most difficult cases. Conversely, when generating the four complex, multi-directive points, we deliberately avoided these directives to prevent the model from overfitting to a few extreme outliers.

IV. EXPERIMENTAL RESULTS

This section presents the experimental validation of our proposed framework. We first detail the experimental setup, including the dataset, evaluation methodology, and training parameters for all models. We then compare the performance of our approach against state-of-the-art baselines along two primary axes: predictive accuracy and exploration efficiency.

The experiments in this work focus on area estimation, training the models to predict the usage of four key resources: LUTs, FFs, DSPs, and BRAMs. To create a single, unified metric, we aggregate these predictions by summing the utilization of each resource, normalized by the total amount available on the target device. While our current study focuses on resource usage, the proposed framework can be readily extended to other metrics like latency and power.

A. Experimental Setup

Dataset and Tools. Our experiments leverage a comprehensive dataset of over 3000 fully-implemented designs. We selected 10 real-world applications from the CHStone [24], MachSuite [25], and PolyBench [26] benchmark suites. For each application, we generated at least 200 unique design points by applying a diverse set of synthesis directives, including `unroll`, `pipeline`, `array_partition`, `loop_merge` and `loop_flatten`. All design points were generated with Vitis 2023.2 targeting device XCU50-FSVH2104-2-E.

Baselines. To evaluate our approach’s effectiveness, we compare it against two representative baselines: one pre-HLS and one post-HLS.

- For the pre-HLS baseline, we selected HARP [7], a state-of-the-art, open-sourced GNN-based methodology. Since the original HARP methodology did not model array partitioning directives, it lacks array-specific features necessary for a fair comparison on our dataset. To address this, we re-implemented its framework, extending it to include these features while keeping the core graph properties faithful to the original methodology.
- For the post-HLS baseline, we chose HGBO-DSE [5], as its model targets the same resource metrics as our own and reports remarkably high estimation accuracy.

Training and Evaluation. All models considered in this study were evaluated using a 10-fold leave-one-benchmark-out cross-validation. In each fold, one entire benchmark was held out as the test set, while the instances from the remaining nine benchmarks were split into 85% for training and 15% for validation. This ensures that our final results reflect the model’s ability to generalize to entirely unseen kernels. The models were trained for a maximum of 300 epochs using the AdamW optimizer [27] with a learning rate of 10^{-4} and a weight decay of 10^{-4} . To prevent exploding gradients, we employed gradient clipping with a maximum norm threshold of 5.0.

B. Results

The prediction accuracy of our proposed models and the baselines was evaluated using a 10-fold leave-one-benchmark-out cross-validation. Fig. 5 provides a detailed, per-benchmark performance comparison between our two proposed approaches, while Fig. 6 shows the average Mean Absolute Percentage Error (MAPE) across all held-out test benchmarks, comparing our models against the selected baselines.

As illustrated in Fig. 6, our proposed models demonstrate a clear advantage in predictive accuracy. Our quick-setup model significantly outperforms the fast HARP baseline, reducing the average error by approximately 35%. This confirms that our anchor-based approach provides a much richer feature set than a purely pre-HLS approach. Most notably, our specialized fine-tuned model achieves the lowest error of all methods, even surpassing the time-consuming, post-HLS HGBO-DSE baseline by approximately 11%. This highlights the power of our

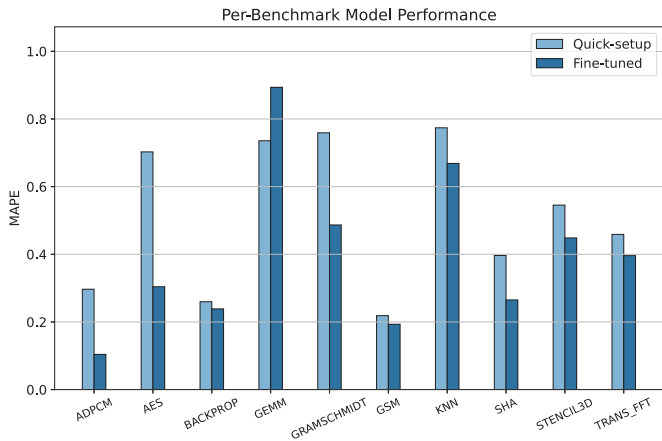


Fig. 5. Per-benchmark Mean Absolute Percentage Error (MAPE) on each held-out test set, comparing the performance of our quick-setup and fine-tuned models.

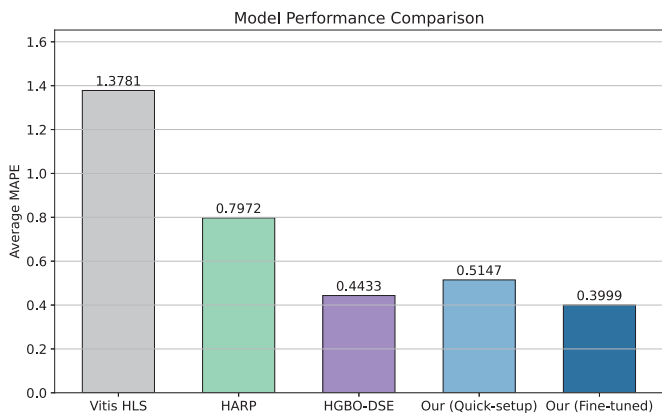


Fig. 6. Average Mean Absolute Percentage Error (MAPE) across all benchmarks. The figure compares our two proposed models against the HARP and HGBO-DSE baselines.

few-shot fine-tuning methodology to adapt to a new kernel’s specific architecture and deliver state-of-the-art accuracy.

As shown in Fig. 5, our fine-tuning methodology significantly improves prediction accuracy across the majority of benchmarks. A notable exception is GEMM, where the 10-point fine-tuning strategy worsened the performance. This suggests a case of negative transfer, where the pre-trained model’s knowledge was misaligned with this kernel’s unique complexity [28], [29]. Crucially, subsequent experiments showed that increasing the fine-tuning set to 15 instances for this benchmark resolved the issue, confirming that a stronger learning signal can overcome this initial bias.

To evaluate the exploration efficiency, we measured the end-to-end runtime for each methodology for an increasing number of design points, as shown in Fig. 7, which reports an average over all kernels. The upfront cost for our anchor-based approaches includes the one-time implementation of the base design and, for the fine-tuned model, the 9 additional fine-tuning points.

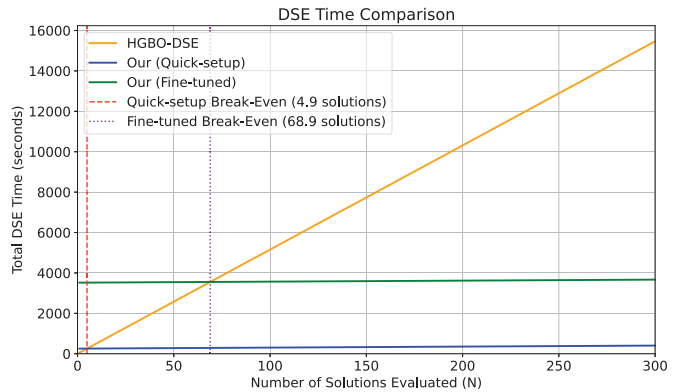


Fig. 7. Total DSE time versus the number of evaluated solutions. The plot shows the break-even points where our quick-setup approach (blue) and our fine-tuned approach (green) become more time-efficient than the HLS-per-point baseline (orange), demonstrating their scalability for large design space explorations.

The results of our efficiency comparison in Fig. 7 confirms the scalability of our framework for DSE. Our quick-setup model becomes more time-efficient than the post-HLS baseline after only approximately 5 evaluations. Even with the upfront cost of implementing 10 designs, our fine-tuned model reaches its break-even point after less than 70 evaluations. For a realistic DSE run involving hundreds or thousands of candidates, our framework offers a dramatic reduction in total exploration time. Furthermore, our framework offers significant flexibility, allowing designers to customize the fine-tuning process by using a variable number of ground-truth-labeled instances to meet their specific accuracy and runtime budget.

V. CONCLUSION & FUTURE WORKS

This paper introduced *Anchor-and-Adapt*, a novel framework for HLS QoR estimation that resolves the critical trade-off between inference speed and predictive accuracy. By leveraging a single fully-implemented *anchor instance* to inform a predictor model with ground-truth information of the kernel, our method achieves significantly higher accuracy than fast, pre-HLS baselines without the prohibitive runtime of post-HLS approaches. Furthermore, we presented a dual-strategy approach, offering both a quick-setup *Ensemble Model* for general use and a high-fidelity *Few-Shot Fine-Tuned Model* for per-kernel specialization. Our experimental results confirm that this methodology provides a practical and powerful solution for guiding high-quality design space exploration.

To further enhance the model’s robustness, a promising direction is to explore more advanced ensembling techniques. One such approach is to create a cross-validated ensemble, where multiple models are trained for each fold using different validation set splits, with the validation set being composed of an entire unseen benchmark instance set. While this methodology is computationally intensive, combining the predictions from this diverse set of models could lead to a state-of-the-art predictor with superior generalization performance.

REFERENCES

- [1] B. C. Schafer and Z. Wang, "High-level synthesis design space exploration: Past, present, and future," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2628–2639, 2020.
- [2] R. S. Molina, V. Gil-Costa, M. L. Crespo, and G. Ramponi, "High-level synthesis hardware design for FPGA-based accelerators: Models, methodologies, and frameworks," *IEEE Access*, vol. 10, pp. 90429–90455, 2022.
- [3] D. S. Lopera, L. Servadei, G. N. Kiprit, S. Hazra, R. Wille, and W. Ecker, "A survey of graph neural networks for electronic design automation," in *2021 ACM/IEEE 3rd Workshop on Machine Learning for CAD (MLCAD)*, 2021, pp. 1–6.
- [4] N. Wu, H. Yang, Y. Xie, P. Li, and C. Hao, "High-level synthesis performance prediction using GNNs: Benchmarking, modeling, and advancing," in *Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC '22)*, 2022, pp. 49–54.
- [5] H. Kuang, X. Cao, J. Li, and L. Wang, "HGBO-DSE: Hierarchical GNN and bayesian optimization based HLS design space exploration," in *2023 International Conference on Field Programmable Technology (ICFPT)*, 2023, pp. 106–114.
- [6] Z. Lin, Z. Yuan, J. Zhao, W. Zhang, H. Wang, and Y. Tian, "PowerGear: Early-stage power estimation in FPGA HLS via heterogeneous edge-centric GNNs," 2022.
- [7] A. Sohrabizadeh, Y. Bai, Y. Sun, and J. Cong, "Robust GNN-based representation learning for HLS," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 2023, pp. 1–9.
- [8] —, "Automated accelerator optimization aided by graph neural networks," in *Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC '22)*, 2022, pp. 55–60.
- [9] M. Gao, J. Zhao, Z. Lin, and M. Guo, "Hierarchical source-to-post-route qor prediction in high-level synthesis with gnn," 2024. [Online]. Available: <https://arxiv.org/abs/2401.08696>
- [10] M. U. Jamal, Z. Li, M. T. Lazarescu, and L. Lavagno, "A graph neural network model for fast and accurate quality of result estimation for high-level synthesis," *IEEE Access*, vol. 11, pp. 85785–85798, 2023.
- [11] G. Huang, J. Hu, Y. He, J. Liu, M. Ma, Z. Shen, J. Wu, Y. Xu, H. Zhang, K. Zhong, X. Ning, Y. Ma, H. Yang, B. Yu, H. Yang, and Y. Wang, "Machine learning for electronic design automation: A survey," 2021.
- [12] B. Yu, "Machine learning in EDA: When and how," in *2023 ACM/IEEE 5th Workshop on Machine Learning for CAD (MLCAD)*, 2023, pp. 1–6.
- [13] E. Celik and D. Dal, "A systematic review of machine learning-driven design space exploration in high-level synthesis," *Integration*, vol. 105, p. 102513, 2025. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167926025001701>
- [14] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous graph neural network," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 793–803. [Online]. Available: <https://doi.org/10.1145/3292500.3330961>
- [15] R. Bing, G. Yuan, M. Zhu, F. Meng, H. Ma, and S. Qiao, "Heterogeneous graph neural networks analysis: a survey of techniques, evaluations and applications," *Artif. Intell. Rev.*, vol. 56, no. 8, pp. 8003–8042, 2023. [Online]. Available: <https://doi.org/10.1007/s10462-022-10375-2>
- [16] X. Wang, D. Bo, C. Shi, S. Fan, Y. Ye, and P. S. Yu, "A survey on heterogeneous graph embedding: Methods, techniques, applications and sources," 2020. [Online]. Available: <https://arxiv.org/abs/2011.14867>
- [17] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous graph transformer," in *Proceedings of The Web Conference 2020 (WWW '20)*, 2020, pp. 2704–2710.
- [18] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2018, pp. 5453–5462.
- [19] Y. Li, C. Gu, T. Dullien, O. Vinyals, and P. Kohli, "Graph matching networks for learning the similarity of graph structured objects," 2019.
- [20] 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 (NIPS'17)*, 2017, pp. 6000–6010.
- [21] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, "Snapshot ensembles: Train 1, get M for free," 2017.
- [22] P. Izmailov, D. Podoprikin, T. Garipov, D. Vetrov, and A. G. Wilson, "Averaging weights leads to wider optima and better generalization," 2019.
- [23] S. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," 2017. [Online]. Available: <https://arxiv.org/abs/1705.07874>
- [24] Y. Hara, H. Tomiyama, S. Honda, H. Takada, and K. Ishii, "CHStone: A benchmark program suite for practical c-based high-level synthesis," in *2008 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2008, pp. 1192–1195.
- [25] B. Reagen, R. Adolf, Y. S. Shao, G.-Y. Wei, and D. Brooks, "MachSuite: Benchmarks for accelerator design and customized architectures," in *2014 IEEE International Symposium on Workload Characterization (IISWC)*, 2014, pp. 110–119.
- [26] L.-N. Pouchet and T. Yuki, "PolyBench/C: The polyhedral benchmark suite," 2016. [Online]. Available: <http://sourceforge.net/projects/polybench/>
- [27] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2019.
- [28] W. Zhang, L. Deng, L. Zhang, and D. Wu, "A survey on negative transfer," *IEEE/CAA Journal of Automatica Sinica*, vol. 10, no. 2, p. 305–329, Feb. 2023. [Online]. Available: <http://dx.doi.org/10.1109/JAS.2022.106004>
- [29] Z. Wang, Z. Dai, B. Póczos, and J. Carbonell, "Characterizing and avoiding negative transfer," 2019. [Online]. Available: <https://arxiv.org/abs/1811.09751>