LLM-Augmented Multi-Modal Fusion for SoC Design Space Exploration

Donger Luo[1*] Tianyi Li[1*] Xinheng Li[1] Qi Sun[2] Cheng Zhuo[2] Bei Yu[3] Jingyi Yu[1] Hao Geng[1†]
[1]ShanghaiTech University [2]Zhejiang University [3]The Chinese University of Hong Kong

***Abstract*—The increasing complexity of modern SoC designs creates challenges in efficiently exploring vast design spaces. Current approaches often reduce microarchitectures to simple parameter vectors, overlooking their rich information embedded in both functional behaviors and implementation details. This paper proposes an LLM-augmented multi-modal fusion method that captures this dual nature of microarchitecture design. By recombining design parameters with their natural language descriptors, we leverage a domain-knowledge-enhanced LLM to extract semantic features that represent functional behavior. Simultaneously, we process Chisel-compiled RTL through a graph neural network to capture structural implementation details. This multi-modal approach enables more effective feature extraction from limited evaluation data. We integrate these rich features into an MLP enhanced with Monte Carlo dropout. This approach provides uncertainty quantification while enabling end-to-end training, allowing the pre-trained feature extractors to be fine-tuned during exploration through Bayesian optimization. Experimental results and ablation studies on a Gemmini-based RISC-V SoC demonstrate that our approach significantly improves exploration efficiency and prediction quality under data limitations.**

I. Introduction

The exponential growth in artificial intelligence applications has driven unprecedented demands for specialized and agile System-on-Chip (SoC) designs. As AI workloads continue to diversify and scale, modern SoCs must balance multiple competing objectives, including performance, power efficiency, and area constraints, while meeting strict timing requirements. This has led to increasingly complex design spaces with numerous configurable parameters spanning multiple microarchitectural components. In this context, Design Space Exploration (DSE) has emerged as a critical challenge in the front-end digital design workflow.

Most existing DSE methodologies have historically approached the problem from a purely quantitative perspective, treating the microarchitecture configurations as numerical and flattened vectors. As illustrated in the upper panel of Fig. 1, these conventional techniques, like Gaussian processes [1], random forest [2], and support vector machine [3], directly correlate these abstract numerical representations with design metrics like performance, power, and area. However, this reductionist approach fails to capture the critical semantic and structural context that defines how parameters actually function within specific architectural contexts.

The feature representation of microarchitecture must simultaneously consider both the functional behavior and implementation details, not merely numerical values after parameterization. This principle becomes clear when examining specific examples: doubling the number of SRAM banks in a systolic-array accelerator widens on-chip bandwidth and can unlock higher utilization, whereas the same modification in a CPU cache primarily shifts the hit-latency–versus–energy trade-off. Similarly, increasing a buffer_depth parameter in a streaming data-flow engine enables deeper pipelining and hides producer–consumer latency, while in

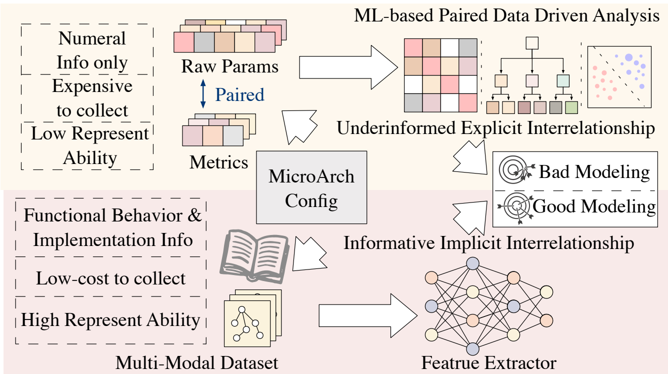

Fig. 1 Current approaches represent microarchitectures as flat numeric vectors, demanding extensive datasets to explicitly learn parameter interrelationships using traditional machine learning methods. Our method extracts multi-modal descriptors from the same inputs, enabling low-cost, high-fidelity embeddings that implicitly capture these interrelationships.

a branch-prediction unit, the identical parameter change mostly increases area with minimal throughput benefit. Such nuances are invisible when parameters are processed as nameless scalars, yet they are fundamental to understanding how design choices propagate through a heterogeneous SoC.

Crucially, conventional approaches lose track of which aspect of the design a parameter affects and how that choice materializes in hardware. The functional behavior describes what the architecture does and how its components interact at a high level, often expressed through natural language descriptions of parameters and their relationships. Implementation details capture how these functions are physically realized in hardware, manifested in the structural patterns of the generated RTL code. Together, these two aspects provide a holistic characterization of any microarchitectural configuration far richer than mere parameter vectors.

When design parameters are paired with their semantic names and descriptions, they encode rich domain knowledge about functional behaviors and design intent. For example, parameters like “cache_size” and “prefetch_depth” may appear mathematically independent as numeric values, but their linguistic representations reveal functional relationships in memory hierarchy design. Similarly, when designs are translated to RTL structures through hardware description languages like Chisel [4] and SpinalHDL [5], the representations contain valuable structural information about implementation details that influence design metrics. Leveraging these complementary aspects—functional behavior expressed through natural language and implementation details manifested in RTL structures—provides deeper insights into how parameter configurations affect design outcomes.

In this paper, we propose a novel multi-modal fusion approach that synergistically integrates these diverse information sources to en-

[*]Equal contributors
[†]Corresponding author, also with Shanghai Engineering Research Center of Energy Efficient and Custom AI IC.

hance microarchitecture DSE. Our method combines two specialized encoders: a domain-knowledge-enhanced LLM-based encoder that captures semantic features from parameter descriptions, and a graph neural network encoder that extracts structural information from RTL representations, as shown in the lower panel of Fig. 1. These encoders transform the parameter design space into a latent representation enriched with domain knowledge about both functional behavior and implementation details. We integrate these rich representations with a multilayer perceptron (MLP) augmented with Monte Carlo (MC) dropout, forming a surrogate model capable of quantifying prediction uncertainty while enabling end-to-end training of the entire pipeline.

The main contributions of this work are:

- A multi-modal fusion framework that leverages both semantic and structural information to enhance microarchitecture DSE, providing a more comprehensive representation of design configurations.
- An LLM-based encoder pre-trained with domain-specific knowledge that extracts rich semantic features from parameter descriptions, enabling the model to capture functional behavior implications.
- A graph neural network approach that processes RTL-level structural information derived from Chisel hardware descriptions, providing insights into the implementation details of microarchitecture parameter choices.
- A Bayesian optimization strategy incorporating MLP with MC dropout that enables simultaneous prediction and uncertainty quantification, while allowing for encoder fine-tuning during exploration.

II. Preliminary

A. Chisel-based Agile Design and Representation of Microarchitecture

Chisel (Constructing Hardware in a Scala Embedded Language) [4] enables hardware design at a higher level of abstraction while maintaining fine-grained control over the generated Register Transfer Level (RTL) implementations. Unlike traditional HDLs, Chisel leverages Scala's programming features to provide powerful parameterization capabilities that are particularly valuable for microarchitectural description. This allows designers to define parameterized hardware generators rather than specific instances, making it possible to instantiate numerous design variants from a single codebase by simply adjusting parameter values.

These design parameters form a multi-dimensional design space that can be formally defined as:

$$\mathcal{D} = \{p_1, p_2, \ldots, p_n\} \quad p_i \in [l_i, u_i], \tag{1}$$

where $\mathcal{D}$ represents the complete design space, p_i denotes each individual microarchitectural parameter (such as cache sizes, pipeline depths, or branch predictor configurations), l_i and u_i are the lower and upper bounds for each parameter. Some parameters may also be categorical rather than numerical, further complicating the design space exploration process.

Notable open-source processor designs like BOOM (Berkeley Out-of-Order Machine) [6] and Rocket Chip [7] demonstrate Chisel's capabilities for creating complex, parameterized microarchitectures. A single parameter change in Chisel can propagate through the design hierarchy, resulting in substantially different RTL implementations with varying performance, power, and area characteristics.

Conventional DSE frameworks often compress a Chisel generator into a bare parameter vector $\mathbf{p} = [p_1, \ldots, p_n]$ and fit a surrogate $f : \mathbf{p} \to (\text{perf}, \text{power}, \text{area})$ using Gaussian processes or boosting trees. Although numerically convenient, this reductionist view discards two information channels that Chisel offers essentially "for free." Linguistic metadata, including parameter names, comments, and hierarchical configuration objects, encapsulate domain knowledge that can bias the search toward semantically meaningful regions. Structural RTL graphs produced during Chisel compilation expose RTL modules hierarchical relationship that foreshadow power, performance, and area trade-offs. Elevating these two modalities to first-class citizens—rather than treating them as noise—is therefore crucial for scaling DSE to modern SoCs.

B. Natural Language Encoders and Graph Encoders

Specialized encoders can extract meaningful features from different information sources and project them into latent spaces suitable for downstream tasks.

Large Language Models (LLMs) [8] process semantic descriptions of design parameters, their functions, and their potential impacts on system performance. These models transform semantic descriptions into dense vector representations in a continuous latent space. Parameters with similar functions or that commonly interact with each other tend to cluster in this latent space, providing valuable inductive bias. This semantic encoding can help models generalize from limited training data by leveraging encoded domain knowledge. Complementarily, graph neural networks (GNNs) extract structural information from RTL representations generated for different parameter configurations. RTL modules' hierarchical information can be represented as graphs, where nodes correspond to hardware modules and edges represent hierarchical relationships between these modules. Through message-passing mechanisms, graph encoders extract features that capture structural properties.

Both encoder types can be pre-trained on domain-specific data before being applied to specific DSE tasks. This pre-training creates a foundation of domain knowledge that does not necessarily require expensive circuit evaluations. The projection of both semantic and structural information into latent spaces enables the integration of these complementary representations, potentially improving prediction accuracy and sample efficiency in the DSE process.

C. Bayesian Optimization and Uncertainty Quantification

Bayesian Optimization (BO) [9] has emerged as an effective framework for DSE due to its ability to balance exploration and exploitation through principled uncertainty quantification. BO consists of two key components: a probabilistic surrogate model that approximates the objective function and an acquisition function that guides the selection of new design points for evaluation.

Probabilistic surrogate models produce both a prediction mean and variance for each point in the design space. For any design point d in the design space $\mathcal{D}$, the surrogate model f_θ with parameters θ outputs:

$$\mu(d), \sigma^2(d) = f_\theta(d) \quad \forall d \in \mathcal{D}, \tag{2}$$

where $\mu(d)$ represents the predicted mean performance and $\sigma^2(d)$ quantifies the prediction uncertainty. This predictive uncertainty helps identify regions that are either promising (high predicted performance) or highly uncertain (potentially containing undiscovered optimal solutions).

The acquisition function leverages both predicted mean and variance to select the next evaluation point, typically balancing exploitation of known good regions with exploration of uncertain regions. For multi-objective optimization problems, Expected Hypervolume Improvement (EHVI) [10] is a commonly used acquisition function:

$$\text{EHVI}(d) = \int max[\text{HV}(R \cup \{y\}) - \text{HV}(R), 0] p(y|d) dy, \tag{3}$$

where R is the current set of evaluated PPAs, HV calculates the hypervolume dominated by R, y represents potential performance metrics for design point d, and $p(y|d)$ is the probability distribution of observing performance y given design parameters d. The design point with the highest EHVI score is selected for the next evaluation.

The quantification of predictive uncertainty across the design space provides valuable guidance for the exploration process, particularly in the early stages when evaluation data is extremely limited. Well-calibrated uncertainty estimates can focus computational resources on the most informative design points, potentially achieving superior design points with fewer evaluations compared to approaches that rely solely on point estimates.

D. Problem Formulation

Problem 1 (Design Space Exploration for Microarchitecture)**.** *The goal of microarchitecture design space exploration is to automatically identify Pareto-optimal configurations that optimize multiple Quality of Results (QoR) metrics, including performance, power consumption, and area.*

III. Overall Framework

Our proposed framework addresses the challenge of microarchitectural DSE through a two-stage approach: design space preprocessing with feature extraction and multi-modal design space exploration.

A. Multi-Modal Data Preparing and Feature Extractor Pretraining

In the preprocessing stage, we convert design configurations into natural language descriptions by combining parameter choices with their explanations, capturing domain expertise and parameter relationships. Concurrently, we compile corresponding Chisel files to generate module-level hierarchical graphs representing the RTL circuit structure. An encoder is trained to encode each of the two modalities. For the language modality, we employ an adapter-based approach to continue pretraining a pre-trained large language model using masked language modeling (MLM) on our domain-specific corpus, enabling it to project parameter descriptions into a latent space that preserves their semantic relationships. For the structural modality, we utilize a graph isomorphism network (GIN) [11] pre-trained using Weisfeiler-Lehman (WL) [12] test scores as guidance, mapping RTL hierarchical graphs to a complementary latent space where structurally similar RTL circuits have closer representations. This pre-training phase effectively embeds our model with domain knowledge without requiring expensive circuit evaluations.

B. Multi-Modal based Design Space Exploration Flow

As is shown in Fig. 2, the exploration flow integrates these multi-modal representations within a Bayesian optimization framework for efficient design space navigation. During each exploration iteration, we concatenate the outputs from both encoders and feed them into an MLP augmented with Monte Carlo (MC) dropout, forming an end-to-end prediction model that takes natural language and graph inputs to produce performance, power, and area (PPA) estimates. In the training phase, while certain encoder layers remain frozen to preserve domain knowledge, both encoders undergo partial fine-tuning alongside MLP training using evaluated design configurations. For prediction, each unevaluated candidate design is transformed into its dual-modal representation and processed through the model, with MC dropout enabling uncertainty quantification alongside PPA mean estimates. The Expected Hypervolume Improvement (EHVI) acquisition function then leverages both prediction means and uncertainty measures to select the most promising design configuration for evaluation. In the evaluation phase, we implement the selected

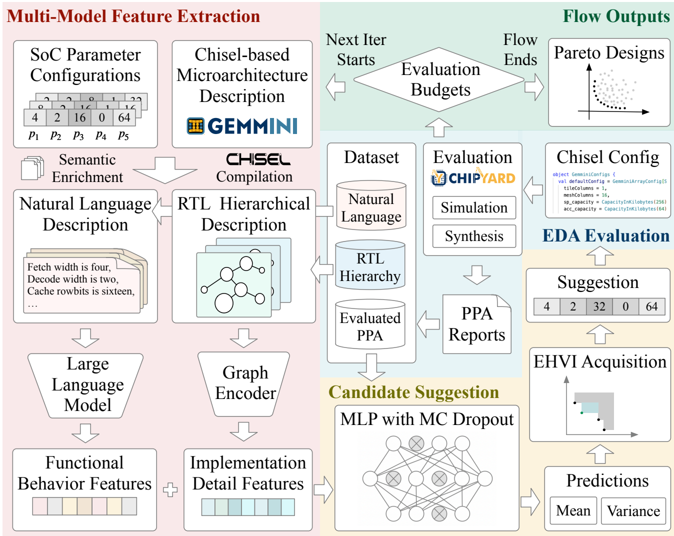

Fig. 2 Overall Flow. The proposed framework synergistically integrates both semantic features (extracted by an LLM-based encoder from parameter descriptions) and structural information (extracted by a graph neural network from RTL representations) to create a more comprehensive latent representation of SoC designs, enhancing exploration efficiency through a Bayesian optimization strategy.

configuration on the Chipyard platform [13], generating RTL-level Verilog code that undergoes simulation through large-model test cases and power/area evaluation through logic synthesis tools. The evaluation results subsequently augment the training dataset for the next iteration, progressively improving model accuracy while efficiently seeking Pareto-optimal designs.

IV. Multi-Modal Feature Extraction of Micro-Architecture Design Space

Current approaches to microarchitectural DSE rely primarily on numerical parameter vectors, which often fail to capture the rich semantic and structural information inherent in design specifications. To address this limitation, we propose a multi-modal feature extraction framework that leverages both natural language descriptions of architecture and RTL hierarchy information. This section introduces the theoretical foundation of our approach and details our two specialized encoders: a BERT-based encoder for processing semantic microarchitecture information and a GIN-based encoder for extracting RTL hierarchy information. Their finetuning during exploration will be introduced in Section V-B.

A. Theoretical Foundation of Multi-Modal Embedding-Based Design Space Exploration

Before diving into the specific encoder architectures, we formalize the theoretical foundation for our multi-modal embedding approach through information theory and functional analysis. Let $\mathcal{X} \subset \mathbb{R}^n$ denote the space of microarchitectural parameters, $\mathcal{T}$ the space of semantic descriptions, and $\mathcal{G}$ the space of RTL hierarchy graphs.

For any parameter configuration $x \in \mathcal{X}$, we define a design metric function $\Phi : \mathcal{X} \to \mathbb{R}^m$ mapping parameters to a vector of m metrics (e.g., power, performance, area). Given a semantic description $T \in \mathcal{T}$ and graph structure $G \in \mathcal{G}$ of a design, we can define two encoder functions:

$$f_T : \mathcal{T} \to \mathbb{R}^{d_1}, \quad f_G : \mathcal{G} \to \mathbb{R}^{d_2}. \tag{4}$$

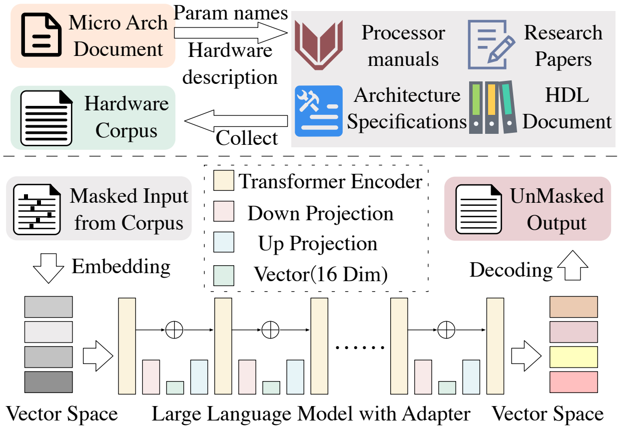

Fig. 3 Pipeline for Domain-Specific Corpus Construction and Continued MLM Pretraining of LLM.

These encoders project high-dimensional, heterogeneous information into compact latent representations $z_T = f_T(T)$ and $z_G = f_G(G)$, respectively.

From an information-theoretic perspective, both modalities reduce uncertainty about design metrics. For a random variable X representing microarchitecture configurations, the conditional entropies satisfy:

$$H(X|T) < H(X), \quad H(X|G) < H(X), \tag{5}$$
$$H(X|T,G) < \min(H(X|T), H(X|G)). \tag{6}$$

The positive mutual information quantities $I(X;T) = H(X) - H(X|T)$ and $I(X;G) = H(X) - H(X|G)$ quantify the information gain from each modality, while the joint information $I(X;T,G)$ exceeds the individual contributions, demonstrating the advantage of our multi-modal approach.

These embeddings induce a prior distribution over parameter configurations:

$$p(x|T,G) \propto \exp(-D(z_x, [z_T; z_G])), \tag{7}$$

where z_x is the latent representation of configuration x, $[z_T; z_G]$ represents the concatenated multi-modal embedding, and D is a distance function in the latent space.

The resulting optimization problem for each design metric can be formulated as:

$$x^* = \arg\max_{x\in\mathcal{X}} \mathbb{E}[\Phi(x)] \cdot p(x|T,G). \tag{8}$$

This formulation effectively constrains the search to a lower-dimensional manifold $\mathcal{M} \subset \mathcal{X}$ where $\mathcal{M} = \{x \in \mathcal{X} | p(x|T,G) > \epsilon\}$ for some threshold ϵ, thereby **reducing the search complexity from $O(|\mathcal{X}|)$ to $O(|\mathcal{M}|)$ where $|\mathcal{M}| \ll |\mathcal{X}|$.**

This theoretical analysis serves as the foundation and starting point of our work, enabling us to **project multi-modal design information into a unified latent space that significantly reduces search complexity while preserving critical semantic and structural information from multiple modalities.**

B. Bert-based Encoder for Design Specification Feature Extraction

BERT (Bidirectional Encoder Representations from Transformers) is a Transformer-based language model pretrained on large-scale corpora using objectives such as Masked Language Modeling (MLM). Its encoder-only architecture enables the extraction of rich semantic features through deep bidirectional self-attention, capturing long-range dependencies and potential relationships among tokenized input. This makes BERT particularly effective for encoding architecture-related natural language descriptions of microarchitecture. By projecting natural language design descriptions into high-dimensional latent spaces, BERT generates dense vector representations that preserve the underlying semantic information, providing a domain knowledge embedded foundation for downstream design space exploration.

While BERT offers powerful semantic encoding capabilities, its application to microarchitectural DSE requires specific adaptations to bridge the gap between general language understanding and domain-specific hardware terminology. The common-used BERT-base model, pretrained on general text corpora, lacks exposure to specialized hardware design vocabulary and the semantic relationships unique to SoC architecture descriptions. To address this limitation, we employ continued pretraining on hardware design corpora and architectural parameter representation learning.

*1) **Microarchitecture-aware Corpora Preparation**:* The upper panel of Fig. 3 illustrates the construction pipeline of the pre-training corpus, the process begins with the extraction of parameter names and hardware-related descriptions from the documentation of the targeted microarchitecture. The extracted terms serve as search keywords to retrieve relevant sentences from various sources, including processor manuals, HDL documentation, SoC architecture specifications, and published research papers. To enhance the diversity and contextual richness of the corpus, a semantic similarity search based on pre-trained sentence embeddings is employed. This approach enables the inclusion of not only exact keyword matches but also sentences containing semantically related expressions or paraphrased variants. This ensures that the resulting corpus captures the required domain-specific accuracy and linguistic variability, enabling the language model to handle previously unseen architectural descriptions with comparable semantics.

*2) **Task-Driven Adapter Design for DSE**:* To match BERT's representations to the dimensional requirements of our downstream PPA prediction task and to inject inductive bias through the model's bottleneck layers, we introduce a lightweight adapter module [14] into each Transformer layer of the pretrained BERT base model. As shown in the lower panel of Fig. 3, the adapter employs a bottleneck architecture:

$$\text{Adapter}(h) = h + W_{\text{up}} \cdot \sigma\left(W_{\text{down}} \cdot h\right), \tag{9}$$

where $h \in \mathbb{R}^{768}$ denotes the hidden state from the original BERT layer, $W_{\text{down}} \in \mathbb{R}^{16\times768}$ and $W_{\text{up}} \in \mathbb{R}^{768\times16}$ are the down-projection and up-projection matrices of the adapter layer, respectively, and σ is a non-linear GELU activation function. This configuration forces the model to compress contextual information into a 16-dimensional latent space before projecting it back, encouraging the emergence of task-relevant compact features. The 16-dimensional output from the last bottleneck layer is also directly extracted as the semantic embedding for each input sentence in our multi-modal fusion framework.

*3) **Aligning LLMs with Design Semantics through Task-Adaptive MLM**:* Domain-adaptive MLM is performed as the continued pre-training objective. In this phase, the pretrained layers remain frozen, and 15% of the tokens in each input sentence are randomly selected for masking and predicted from the surrounding context. Formally, for an input sequence $x = \{x_1, \ldots, x_n\}$, we define a masked index set $\mathcal{M} \subset \{1, \ldots, n\}$ and minimise the negative log-likelihood:

$$\mathcal{L}_{\text{MLM}} = -\sum_{i\in\mathcal{M}} \log P\big(x_i \mid x_{\setminus\mathcal{M}}\big), \tag{10}$$

where $x_{\setminus\mathcal{M}}$ denotes the sequence with tokens at $\mathcal{M}$ removed or replaced. By repeatedly exposing the encoder to our hardware-centric corpus, the model learns to recognize specialized hardware terminology and arrange related tokens in close proximity within latent semantic space, thereby uncovering hidden semantic correlations among micro-architectural parameters and their contextual roles in design exploration.

Through adapter injection and domain-aware MLM pre-training, the BERT encoder is ultimately adapted to transform natural-language descriptions of microarchitectural parameters into compact, hardware-aware embeddings that capture both meaning and context-dependent relationships.

C. GIN-based Encoder for RTL Hierarchy Information Extraction

High-level hardware designs described in Chisel frequently rely on parameterization to enable rapid exploration of diverse microarchitectural configurations. When compiled to RTL, such parameterized designs manifest a graph structure in which hierarchical information between modules embodies critical architectural decisions, such as pipeline depth, memory organization, and specialized functional units. This naturally arising RTL hierarchy preserves both structural and functional relationships that can be leveraged to gain insights into PPA characteristics.

In order to exploit this inherent structural information, the proposed approach converts each RTL hierarchy into a graph, where nodes represent modules, and edges capture the parent-child relationship of RTL modules. Fig. 4 is the training diagram of the proposed graph encoder, a Graph Isomorphism Network (GIN) is employed to learn expressive embeddings of the RTL hierarchy graph.

*1) **Capturing RTL Structure via GIN-based Encoder**:* For our encoder implementation, we adopt GIN to effectively extract the hierarchical relationships in RTL designs. Let $\mathbf{n}_i^{(l)}$ denote the representation of node i at the l-th GIN layer, and let $\mathcal{N}(i)$ be the set of its neighboring nodes. Each layer updates node embeddings by aggregating neighborhood features and applying a learnable residual weight $\delta^{(l)}$, defined as:

$$\mathbf{n}_i^{(l)} = \mathrm{MLP}^{(l)}\left(\left(1+\delta^{(l)}\right)\mathbf{n}_i^{(l-1)} + \sum_{j\in\mathcal{N}(i)} \mathbf{n}_j^{(l-1)}\right), \tag{11}$$

where $\mathrm{MLP}^{(l)}$ is a layer-specific feedforward network. This formulation preserves node-specific information via the weighted self-loop term and enriches the representation with localized structural information. Through progressive stacking of GIN layers, the model captures multi-hop relational dependencies within the RTL hierarchy.

After several GIN layers, a global pooling step aggregates the node embeddings into a single, graph-level representation. A fully connected projection then maps this representation into a 16-dimensional vector, thereby yielding a compact embedding of the entire hierarchical design.

*2) **Leveraging Weisfeiler-Lehman Similarity for Structurally-Aware Embedding**:* To guide the embedding process toward structurally relevant features, we employ the Weisfeiler-Lehman (WL) test score to define a ground-truth similarity metric for pairs of RTL graphs. The WL test iteratively refines node labels based on the labels of neighboring nodes, ultimately producing a histogram of labels that characterizes each graph. Given two graphs G_1 and G_2, the similarity measure is defined as:

$$\begin{aligned} WL(G_1,G_2) &= \sum_{i=0}^{h}\langle\phi_i(G_1),\phi_i(G_2)\rangle \\ &= \sum_{i=0}^{h}\sum_{\ell\in\Sigma_i}\phi_{i,\ell}(G_1)\,\phi_{i,\ell}(G_2), \end{aligned} \tag{12}$$

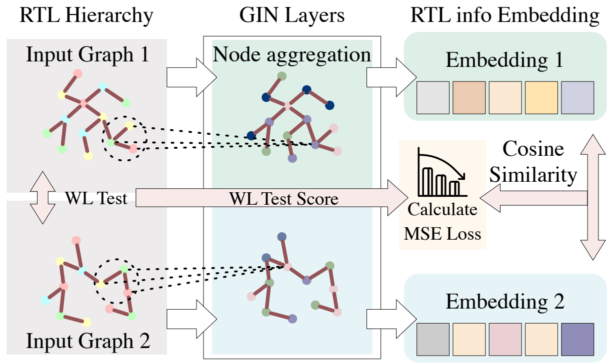

Fig. 4 The Graph Isomorphism Network training process is guided by the Weisfeiler–Lehman test and cosine similarity. In the input graph, each node denotes a module in the RTL Hierarchy.

where $\langle\cdot,\cdot\rangle$ denotes the standard dot product between two vectors, $\phi_{i,\ell}(G)$ is the count of nodes in graph G that have label ℓ at iteration i, Σ_i is the set of all labels present at the i-th iteration, and h is the total number of WL iterations.

During the training process, the GIN-based encoder is employed to generate graph embeddings. Subsequently, the mean squared error of their similarity is calculated based on the WL test score. This supervision compels the encoder to internalize hierarchical information between modules. By aligning the learned representations with actual design similarity, the network produces a final embedding that captures the detailed implementation features of each design configuration.

V. Monte Carlo Dropout-based Neural Network as Surrogate Model in Bayesian Optimization

A. Monte Carlo Dropout for Uncertainty Quantification

To enable uncertainty-aware surrogate modeling within the Bayesian optimization framework, this work adopts Monte Carlo (MC) Dropout as a practical approximation to Bayesian inference. This approach allows quantification of epistemic uncertainty in neural predictions, which is critical for guiding the exploration of embedded multi-modal microarchitecture design space. Compared with Gaussian process as surrogate model, MLP with MC Dropout can get a better understanding of the embedded information and enables a gradient based encoder optimization. The theoretical foundation for using dropout as a Bayesian approximation within our optimization framework is formalized below. This theorem establishes how Monte Carlo Dropout provides the uncertainty estimates needed for effective Bayesian optimization:

Theorem 1 (Dropout as Bayesian Approximation for Uncertainty Estimation [15])**.** *Let $f(\mathbf{z};\mathbf{W})$ be an L-layer neural network with dropout probability p. Define the variational distribution:*

$$q_\theta(\mathbf{W}) = \prod_{l=1}^{L}\Big(p,\delta(\mathbf{W}l) + (1-p),\delta(\mathbf{0})\Big), \tag{13}$$

where $\delta(\cdot)$ denotes a point mass. Then performing dropout at both training and test time and averaging over T stochastic forward passes gives the Monte-Carlo estimators:

$$\hat{\mu}_T(\mathbf{z}) \;=\; \frac{1}{T}\sum_{t=1}^{T} f(\mathbf{z};\mathbf{W}^{(t)}), \tag{14}$$

$$\hat{\sigma}_T^2(\mathbf{z}) \;=\; \frac{1}{T}\sum_{t=1}^{T}\big(f(\mathbf{z};\mathbf{W}^{(t)}) - \hat{\mu}_T\big)^2, \tag{15}$$

which are unbiased estimators of the predictive posterior mean and variance:

$$\mu(\mathbf{z}) = \mathbb{E}q_\theta(\mathbf{W})[f(\mathbf{z};\mathbf{W})], \tag{16}$$

$$\sigma^2(\mathbf{z}) = \mathbb{V}q\theta(\mathbf{W})[f(\mathbf{z};\mathbf{W})]. \tag{17}$$

Proof. The dropout objective can be rewritten as the evidence lower bound (ELBO) $\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{q_\theta(\mathbf{W})}\big[\log p(\mathcal{D} \mid \mathbf{W})\big] - \mathrm{KL}\big(q_\theta(\mathbf{W}) \parallel p(\mathbf{W})\big)$, where $p(\mathbf{W})$ is a weight prior whose scale equals the L_2-regularizer. Stochastic optimisation of $\mathcal{L}_{\text{ELBO}}$ with respect to θ is exactly equivalent to ordinary dropout training. Because $q\theta(\mathbf{W})$ is used during both training and inference, drawing T samples $\mathbf{W}^{(t)}t = 1^T$ yields an i.i.d. set from $q\theta$, so the sample moments are unbiased estimates of the true posterior moments by the law of large numbers. □

Building on this theoretical foundation, we implement MC Dropout in our surrogate model architecture. Following the multi-modal encoding pipeline, each training instance consists of a natural language description $\mathcal{T}$ and its corresponding RTL hierarchy graph $\mathcal{G}$. The BERT-based encoder processes $\mathbf{x}_\text{T}$ to produce a semantic latent vector $\mathbf{z}_\text{T} \in \mathbb{R}^{16}$, while the GIN-based encoder extracts a structural embedding $\mathbf{z}_\text{G} \in \mathbb{R}^{16}$ from $\mathcal{G}$. These two modality-specific representations are concatenated to form a unified 32-dimensional embedding:

$$\mathbf{z} = [\mathbf{z}_\text{T}; \mathbf{z}_\text{G}] \in \mathbb{R}^{32}. \tag{18}$$

This fused representation $\mathbf{z}$ is then fed into a surrogate MLP predictor with Monte Carlo Dropout for PPA estimation:

$$\hat{\mathbf{y}} = f(\mathbf{z}). \tag{19}$$

The surrogate predictor is implemented as a MLP with interleaved dropout layers. Let $f(\cdot)$ denote the MLP with MC Dropout applied during training and inference. Its structure is formally defined as:

$$f(\mathbf{z}) = \mathbf{W}_3 \cdot \text{Dropout}(\sigma(\mathbf{W}_2 \cdot \text{Dropout}(\sigma(\mathbf{W}_1 \cdot \mathbf{z})))), \tag{20}$$

where $\mathbf{W}_1$, $\mathbf{W}_2$, and $\mathbf{W}3$ are the trainable weight matrices of each layer, $\sigma(\cdot)$ denotes the ReLU activation function, and Dropout$(\cdot)$ randomly zeroes out neuron activations with a fixed probability p during each forward pass, in accordance with the variational distribution $q\theta(\mathbf{W})$ defined in the theorem. During inference, the surrogate predictor is applied multiple times with dropout enabled, yielding a set of stochastic predictions for a given input embedding $\mathbf{z}$. Each prediction corresponds to the estimated PPA metrics. Applying the Monte Carlo estimators from the theorem, we directly compute the predictive mean $\hat{\mu}_T(\mathbf{z})$ and variance $\hat{\sigma}^2_T(\mathbf{z})$ of the performance metrics. The predictive mean μ serves as the surrogate model's estimate of expected performance, while the variance σ^2 quantifies the epistemic uncertainty arising from limited training data. These statistics are used by the Bayesian optimization framework to guide the acquisition function, effectively balancing the exploitation of high-performing regions with an exploration of uncertain areas in the design space.

Through this integration of MC Dropout, the surrogate model not only offers reliable performance predictions but also exposes uncertainty estimates that are essential for effective exploration in multi-objective design space optimization.

B. *Finetuning the Multi-Modal Encoders with Evaluated Data*

While pretrained encoders capture rich architecture semantic, and RTL circuit structural information, their original training objectives are not tailored to the demands of PPA prediction. To reconcile this misalignment, the proposed framework introduces a finetuning mechanism that jointly updates the BERT and GIN-based encoders. Departing from conventional surrogate-only training schemes, this design allows gradients to propagate through the full model stack, enabling coordinated optimization of upstream embeddings with respect to the downstream objective.

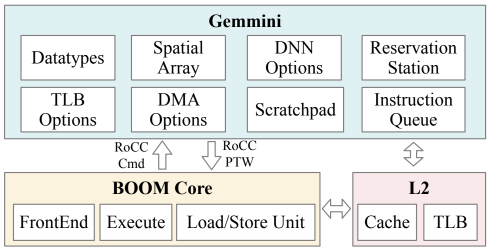

Fig. 5 The blocks refer to the example of aspects in a Gemmini RISC-V SoC microarchitecture described by parameters in Chisel code.

To implement this finetuning strategy in practice, the training process selectively updates parts of the model while keeping others fixed, balancing adaptability and prior preservation. During finetuning, only the top layers of the encoders and the projection heads are updated, while the lower layers remain frozen to preserve general linguistic and structural priors learned during pretraining. The model is optimized using the mean squared error (MSE) between the predicted and normalized ground-truth PPA vectors:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N} \|\hat{\mathbf{y}}_i - \mathbf{y}_i\|^2 . \tag{21}$$

To mitigate overfitting, especially in the early stages of Bayesian optimization when the amount of labeled data is limited, the framework employs a progressive training strategy. At the beginning of the optimization process, the parameters of both encoders are kept frozen, and only the surrogate MLP is updated to adapt quickly to the available supervision. As more labeled samples are gathered, the higher layers of the BERT and GIN encoders are gradually unfrozen, allowing the model to refine its internal representations in a controlled manner.

This progressive finetuning scheme ensures that both encoders evolve from general-purpose pretrained representations toward task-specific embeddings optimized for PPA prediction. As a result, the latent space becomes increasingly aligned with the objectives of design space exploration, leading to improved sample efficiency and optimization performance in subsequent rounds of Bayesian search.

VI. Experiment

A. *Experimental Setup*

For experimental evaluation, we utilized a RISC-V SoC featuring the BOOM core and the Gemmini accelerator [22]. Fig. 5 illustrates the various aspects of the SoC microarchitecture description, which are controlled by corresponding parameters. The performance simulation employed a diverse collection of language models across various scales, including the compact Transformer-small [23], the medium-sized Bert-base [8], and several larger architectures such as Transformer-large, Bert-large, and GPT1-small [24].

The implementation methodology began with hardware description conversion using the Chipyard framework (v1.9.1), translating Chisel descriptions to synthesizable RTL. For performance metrics, we employed the Spike simulator to extract cycle-count measurements during inference operations. The physical implementation incorporated Cadence Genus 201 for logic synthesis on the ASAP7 7nm

TABLE I Comparisons of DSE Performance against SOTA works.

Metrics	DAC'16 [16]	MLCAD'19 [17]	ASPDAC'20 [18]	DAC'23 [19]	AAAI'24 [20]	DAC'24 [21]	**Ours** w/ GP	**Ours**
$\mathbf{HV}_{0,1}$	0.519	0.661	0.549	0.606	0.835	0.710	0.850	**0.887**
$\mathbf{HV}_{0,2}$	0.309	0.473	0.353	0.440	0.487	0.491	0.587	**0.617**
$\mathbf{ADRS}_{\mu}$	0.176	0.102	0.085	0.110	0.091	0.098	0.083	**0.060**
$\mathbf{ADRS}_{\sigma}$	0.201	0.162	0.135	0.140	0.127	0.121	0.135	**0.118**
$\mathbf{HV}_{\mu}$	0.549	0.629	0.593	0.700	0.730	0.725	0.828	**0.854**
$\mathbf{HV}_{\sigma}$	0.132	0.107	0.121	0.105	0.083	0.063	0.097	**0.058**
$\mathbf{Iter}_{\mu}$	16.670	11.491	8.480	8.690	6.300	5.989	3.861	**2.169**
$\mathbf{Iter}_{\sigma}$	18.010	14.921	12.804	8.908	6.573	**2.912**	6.930	3.240

technology node, while energy consumption analytics were performed via Cadence Joules 201.

Our exploration framework examined multiple optimization objectives, including cycles, power, and area. The architectural parameter space encompasses 26 distinct configurable elements, yielding a theoretical design space of approximately 1.3×10^{11} potential Chisel implementations. Through feasibility analysis sampling, we constructed a representative offline dataset comprising 5194 valid configurations for comprehensive evaluation.

B. Comparison Against SOTA Methods

These baseline methods and ours are tested for 100 times, with each method assigned a total quota of 40 times evaluations (from chisel configuration to post-synthesis netlists, including performance simulation) and k-means based 10 initial evaluations from 5 clusters:

- DAC'16 [16]: A method combining statistical sampling and Adaboost-based active learning.
- MLCAD'19 [17]: A method using Gaussian process-based multi-objective Bayesian optimization
- ASPDAC'20 [18]: XGBoost-based method.
- DAC'23 [19]: A method that uses GNN to encode the parameter design space into a latent space for Bayesian optimization.
- AAAI'24 [20]: Method based on reinforcement learning.
- DAC'24 [21]: Method using transformed Bayesian optimization to seek 3 given Pareto-optimal designs.
- Ours w/ GP: Our method without forming the end-to-end model using MC MLP and encoder finetuning, directly exploring the embedded multi-modal space using Gaussian process-based Bayesian optimization.
- Ours: Our full method.

To comprehensively evaluate the performance of our method against the baselines, we use several different metrics: average distance from reference set (**ADRS**), hypervolume (**HV**), and iteration efficiency (**Iter**). We calculate these metrics using normalized Performance, Power, and Area (PPA) metrics with a reference point of $(1.1, 1.1, 1.1)$. $\mathbf{HV}_{0,1}$ is the hypervolume of performance and power. $\mathbf{HV}_{0,2}$ is the hypervolume of performance and area. **HV** is the hypervolume of performance, power, and area. The ADRS and hypervolume values presented in TABLE I are further normalized. We define iteration efficiency as the number of iterations required for each method to reach a hypervolume of 0.5 (exploring approximately half of the potential design space). This metric provides valuable insight into how quickly each approach can identify promising design points, which is critical for practical deployment.

Although each algorithm is constrained to a budget of 50 evaluations, the real-world time cost is substantial. When accounting for the EDA tool processing time per evaluation, the total runtime exceeds one week for a complete design space exploration run. Design engineers typically cannot afford to wait for such extended periods only to receive unexpected and suboptimal results. Therefore, we present both the mean μ and standard deviation σ of metrics across all experiments to provide a more reliable assessment of each method's performance and consistency.

While baseline methods normalize raw parameters via `MinMaxScaler`, our approach encodes design configurations using a multimodal backbone. 50K tokens of domain-specific corpus, augmented through paraphrasing, are used to continue pretraining the BERT encoder via masked language modeling. Simultaneously, 200 randomly picked parameter configurations are compiled into RTL code, and their hierarchy graphs are used to train the GIN-based encoder. This multi-modal feature extraction stage takes 2 hours (about $1/4$ runtime for one EDA tool evaluation) from data collecting (corpus generation and RTL compilation) to encoders' training, on a platform with an A100 40G GPU.

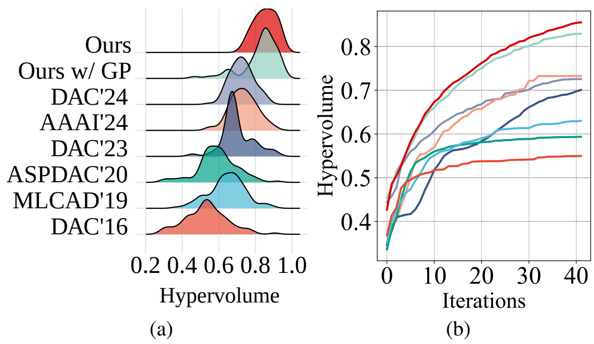

Fig. 6 (a) Hypervolume scores' distribution of different methods' 100 randomly repeated tests. (b) The visualization of different methods' hypervolume changes with iterations.

The results in TABLE I present the comparative performance of our method against state-of-the-art baselines across both hypervolume and iteration efficiency metrics, while Fig. 6 gives the visualization. The results demonstrate several key insights into the effectiveness of our approach.

Our method achieves the minimum mean **ADRS** score of 0.060 and the maximum mean **HV** score of 0.854, representing a significant improvement over existing approaches. The **Iter** results further emphasize the advantages of our approach. Our method requires only 2.169 iterations on average to reach a hypervolume of 0.5 after initialization. Moreover, our method achieves small standard deviations, indicating that our method not only achieves the highest exploration efficiency, but also demonstrates superior stability, ensuring consistent and reliable results in microarchitecture design space exploration scenarios.

C. Ablation Studies

1) Model's Accuracy in DSE Scenario: To isolate the contributions of different components in our proposed approach, we conducted an ablation study comparing the prediction accuracy of various surrogate model configurations under data-constrained scenarios. This

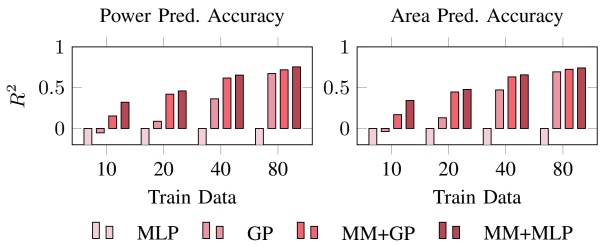

Fig. 7 Power/Area Prediction R^2 under Different Training Size.

study helps elucidate the relative importance of our multi-modal encoders and the choice of prediction model. We evaluated four model configurations:

- MLP: A multilayer perceptron with Monte Carlo dropout operating on scaled microarchitecture parameter space.
- GP: A standard Gaussian process (GP) regression model operating on scaled microarchitecture parameter space.
- MM+GP: Our multi-modal fusion encoder combined with a GP regression model.
- MM+MLP: Our complete approach combining the multi-modal fusion encoder with the Monte Carlo MLP surrogate model.

Fig. 7 shows the prediction R^2 values for the power and area metric across different model configurations and training set sizes. This result reveals that both components of our approach, the multi-modal fusion encoder and the MC MLP surrogate model, contribute significantly to prediction accuracy, especially when the training data size is highly limited (at the early stage of exploration).

2) Multi-Modal's Representation Capability of Design Metric Space: We conducted an analysis using Centered Kernel Alignment (CKA) to measure the alignment between different representation spaces and the target metrics (PPA) space. CKA provides a normalized similarity index between 0 and 1, where higher values indicate stronger alignment between the kernels of two spaces, suggesting that one space better represents the structure relevant to the other.

For this analysis, we constructed kernel matrices using the RBF kernel and linear kernel, respectively. Given two kernel matrices K and L corresponding to different representation spaces:

$$CKA(K, L) = \frac{HSIC(K, L)}{\sqrt{HSIC(K, K) \cdot HSIC(L, L)}}, \tag{22}$$

where $HSIC$ represents the Hilbert-Schmidt Independence Criterion:

$$HSIC(K, L) = \frac{1}{(n-1)^2} trace(K_c L_c), \tag{23}$$

and c represents centering.

TABLE II presents the CKA similarity scores between different representation spaces and the performance metrics space. **Parameter Space** refers to the scaled microarchitecture parameter space, and **Multi-Modal Space** refers to the embedded multi-modal space, where parameterized microarchitectures are represented as natural language descriptions and RTL hierarchy graphs before being embedded. As demonstrated by the results, our embedded multi-modal space exhibits significantly higher CKA similarity scores with the metrics space compared to the parameter space.

TABLE II Representation Capability of Design Metric Space.

Metrics	Parameter Space	Multi-Modal Space
CKA_{rbf}	0.0480	**0.7499**
CKA_{linear}	0.0506	**0.4873**

VII. Conclusion

In this work, we introduced a novel LLM-augmented multi-modal framework for microarchitectural design space exploration (DSE), which integrates natural language and structural RTL representations to enrich the representation of complex SoC configurations. By pairing a domain-adapted BERT encoder with a GIN-based structural encoder, our approach captures both the semantic intent and the physical implementation characteristics of parameterized hardware designs. We further integrated these representations into a Bayesian optimization framework with Monte Carlo dropout, enabling both accurate performance prediction and principled uncertainty quantification. Experimental results on a Gemmini-based RISC-V SoC demonstrate our method's significant performance.

References

[1] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer school on machine learning*. Springer, 2003, pp. 63–71.
[2] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.
[3] M. Hearst, S. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 18–28, 1998.
[4] J. Bachrach *et al.*, "Chisel: Constructing hardware in a scala embedded language," in *Proc. DAC*, 2012.
[5] C. Papon and Y. Xiao, "SpinalHDL." [Online]. Available: https://github.com/SpinalHDL/SpinalHDL
[6] K. Asanovic *et al.*, "The berkeley out-of-order machine (boom): An industry-competitive, synthesizable, parameterized risc-v processor," *University of California at Berkeley Berkeley United States, Tech. Rep*, 2015.
[7] K. Asanovic *et al.*, "The rocket chip generator," *EECS Department, University of California, Berkeley, Tech. Rep.*, 2016.
[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
[9] J. Mockus, "The application of bayesian methods for seeking the extremum," *Towards global optimization*, 1998.
[10] M. T. Emmerich *et al.*, "Hypervolume-based expected improvement: Monotonicity properties and exact computation," in *Proc. CEC*, 2011.
[11] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.
[12] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels." *Journal of Machine Learning Research*, 2011.
[13] A. Amid *et al.*, "Chipyard: Integrated design, simulation, and implementation framework for custom socs," *IEEE Micro*, 2020.
[14] N. Houlsby *et al.*, "Parameter-efficient transfer learning for nlp," in *Proc. ICML*. PMLR, 2019, pp. 2790–2799.
[15] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: representing model uncertainty in deep learning," in *Proc. ICML*, 2016.
[16] D. Li *et al.*, "Efficient design space exploration via statistical sampling and adaboost learning," in *Proc. DAC*, 2016.
[17] Y. Ma, Z. Yu, and B. Yu, "CAD tool design space exploration via bayesian optimization," in *Proc. MLCAD*, 2019.
[18] Z. Xie *et al.*, "Fist: A feature-importance sampling and tree-based method for automatic design flow parameter tuning," in *Proc. ASPDAC*. IEEE, 2020, pp. 19–25.
[19] X. Yi *et al.*, "Graph representation learning for microarchitecture design space exploration," in *Proc. DAC*, 2023.
[20] C. Bai *et al.*, "Towards automated risc-v microarchitecture design with reinforcement learning," *Proc. AAAI*, 2024.
[21] D. Luo *et al.*, "Knowing the spec to explore the design via transformed bayesian optimization," in *Proc. DAC*, 2024.
[22] H. Genc *et al.*, "Gemmini: Enabling systematic deep-learning architecture evaluation via full-stack integration," in *Proc. DAC*, 2021.
[23] A. Vaswani *et al.*, "Attention is all you need," 2017.
[24] A. Radford *et al.*, "Improving language understanding by generative pre-training," https://openai.com/research/language-unsupervised/, 2018.