From Flatland to Forest: Exploring Pareto-optimal Design through RTL Hierarchy Trees

Donger Luo[1] Qi Sun[2] Xingheng Li[1] Cheng Zhuo[2] Bei Yu[3] Hao Geng[1†]
[1]ShanghaiTech University [2]Zhejiang University [3]Dept. CSE, CUHK

***Abstract*—The growing complexity of modern hardware has created vast design spaces that are difficult to explore efficiently. Current design space exploration (DSE) methods treat designs as flat parameter vectors, failing to leverage the rich structural information inherent in hardware architectures. This paper presents a novel RTL hierarchy aware approach to microarchitecture DSE that exploits the natural structure of hardware designs. We propose an RTL hierarchy aware kernel that enables direct comparison of RTL hierarchies, preserving their structural characteristics. Our method incorporates module importance derived from hierarchical synthesis reports through a weighted kernel extension. Additionally, we introduce a clustering method that leverages the proposed kernel to identify distinct architectural patterns, enabling efficient parallel evaluation. Experimental results and ablation studies on a Gemmini-based RISC-V SoC demonstrate the superiority of our approach.**

I. Introduction

Scaling Law, predicting that AI models are doubling in size every six months, more rapidly spawn customer-centric or application-specific silicon solutions like System-on-Chip (SoC). Modern SoCs must balance multiple competing objectives, from maximizing performance and minimizing power consumption to optimizing area utilization and reducing design costs. Each architectural decision, including processor configurations, memory hierarchies, interconnect topologies, and peripheral interfaces, can profoundly influence the overarching system attributes, thereby opening up an extensive design space. The rise of parameterized hardware languages like Chisel [1] and SpinalHDL [2] further expanded the design space by providing flexible microarchitecture design options. Design space exploration (DSE) has thus become crucial, with machine learning-aided approaches, particularly Bayesian optimization (BO), gaining prominence [3]–[8]. These methods use surrogate models to guide the exploration of the designs' microarchitecture parameters, reducing the number of expensive design evaluations needed. However, current approaches suffer from a fundamental limitation: they reduce the rich, multidimensional nature of hardware architecture into flat parameter vectors.

In this paper, we argue that hardware designs should not be represented as flat parameters in DSE frameworks. Consider a modern AI accelerator, it's an intricate system where compute arrays, memory hierarchies and interconnect networks work in concert to achieve optimal performance. Flattening such intricate designs into parameter vectors strips away critical structural information, much like reducing a three-dimensional object to its shadow. The hierarchical relationship between memory levels, the spatial patterns of data flow, and the topological characteristics of the interconnect, all become indistinguishable from auxiliary parameters in a one-dimensional list. Two designs with nearly identical memory hierarchies but different buffer sizes would appear as distant points in the parameter space, preventing search algorithms from leveraging their architectural similarity. This "flattening" of structural information not only makes it harder for machine learning methods to understand design implications but also creates an artificially complex search landscape where natural architectural patterns are obscured.

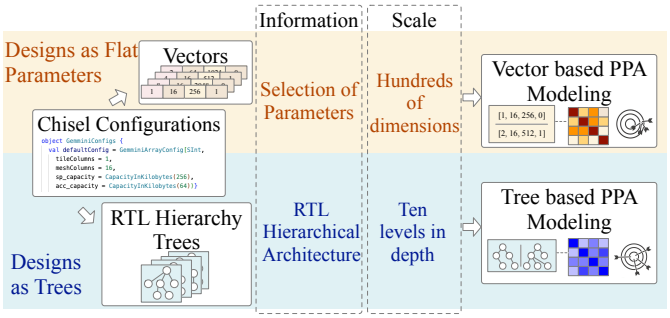


Fig. 1 While Chisel configurations have traditionally been represented as vectors of flat parameters, this approach provides limited information and creates an overly complex modeling space. In contrast, RTL hierarchy trees offer richer information content while providing a more constrained and manageable modeling space.

The flat parameter representation creates two additional critical challenges. First, the sheer number of design parameters, hundreds in Chisel generators like Gemmini [9], leads to the curse of dimensionality [10]. Furthermore, this "flatland" is full of "dead zones" since most parameter combinations fail to pass basic sanity checks. Second, the heterogeneous nature of these parameters poses a fundamental modeling challenge. It is difficult for a machine learning model to effectively handle the diverse parameter types simultaneously, including integer, boolean, and categorical.

To break free from this limited perspective, we propose an RTL hierarchy aware framework that preserves and exploits the natural structure of hardware designs by representing the designs with its hierarchy trees of RTL modules, as shown in Fig. 1. To make comparisons between these trees, we

†Corresponding author, also with Shanghai Engineering Research Center of Energy Efficient and Custom AI IC.

adopt the Weisfeiler-Lehman (WL) subtree kernel method. By constructing a Gaussian process regression model based on a weighted version of the WL subtree kernel, we enable predicting means and variance of PPAs with trees as inputs. We develop a clustering strategy that identifies and groups structurally similar designs, enabling efficient parallel evaluations of designs with distinct architectural patterns.

The main contributions of this work are:

- The first framework to break away from the flat parameter-centric perspective in microarchitecture design space exploration by proposing a novel RTL hierarchy aware kernel, which naturally captures structural similarities between different design variants.
- Theoretical analysis establishes that our approach achieves better sample complexity than the traditional parameter-based RBF kernel methods.
- A Gaussian Progress Regression model based on the weighted extension of the proposed kernel that incorporates module importance, enabling more accurate prediction by accounting for the differential impact of various hardware components.
- A clustering method that leverages the proposed RTL hierarchy aware kernel to identify distinct architectural patterns, enabling efficient parallel evaluation and improved exploration strategies.

II. Preliminaries

A. Chisel-based Agile Design and RTL Hierarchy Tree

Modern hardware design methodologies increasingly adopt Chisel [1] to generate RTL code for its parameterized hardware generation capabilities. Unlike traditional HDLs, Chisel allows designers to create flexible hardware generators that produce related designs through parameter configuration.

In conventional microarchitecture design space exploration (DSE), designs are represented as parameter vectors in Chisel:

$$\mathbf{x} = [p_1, p_2, ..., p_n] \in \mathcal{P}_1 \times \mathcal{P}_2 \times ... \times \mathcal{P}_n, \quad (1)$$

where p_i represents the i-th parameter and $\mathcal{P}_i$ is its corresponding domain. During Chisel code compilation, Chisel creates a hierarchical structure reflecting module relationships. This modularized approach facilitates direct comparison between modules through their instance names, while the hierarchical structure provides valuable architectural insights to guide the exploration process.

B. Bayesian Optimization and Gaussian Process

Bayesian optimization (BO) [11] optimizes expensive black-box functions by constructing a surrogate model and using an acquisition function to identify promising design points. Given design configurations $\mathbf{x}$ and performance metrics $\mathbf{y}$, Gaussian process regression (GPR) serves as the surrogate model, providing predictions and uncertainty estimates.

Multi-objective Bayesian optimization (MOBO) extends this framework to handle multiple objective functions $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), ..., f_M(\mathbf{x})]$, each modeled by an independent GP.

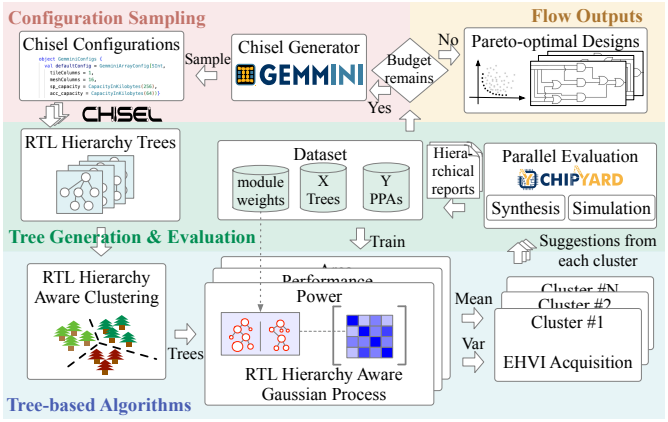


Fig. 2 Overall flow. The proposed approach leverages RTL hierarchy trees for design representation and conducts exploration in a tree-structured design space.

C. Problem Formulation

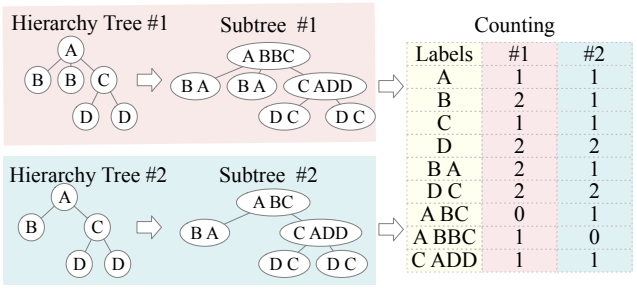
Problem 1 (Design Space Exploration for Microarchitecture)**.** *The goal of microarchitecture design space exploration is to automatically identify Pareto-optimal configurations that optimize multiple Quality of Results (QoR) metrics, including performance, power consumption, and area. This optimization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

III. Overall Flow of Tree-based RTL Hierarchy Aware Exploration Framework

In this section, we present our proposed framework for microarchitecture design space exploration based on RTL hierarchy. This iterative approach takes Chisel configurations as input and generates Pareto-optimal configurations as output. Fig. 2 illustrates the workflow of our method.

Each iteration begins with sampling from the microarchitecture's Chisel parameter design space. Distinguished from conventional approaches, these samples are transformed in parallel into RTL hierarchy trees through the Chisel compiler, turning the design space from a "flatland" into a "forest". This procedure takes a few minutes but is rather quick compared with the evaluation of these designs which takes hours.

The Weisfeiler-Lehman subtree kernel (detailed in Section IV) is employed to process the RTL hierarchical information embedded in these trees. By integrating the kernel with our clustering method (detailed in Section V-B), we partition the "forest" into distinct clusters based on the hierarchical similarities among trees. These clustered trees are then fed into a specialized Gaussian process (GP) regression model (detailed in Section V-A) for PPA prediction. The core of our GP model is an enhanced subtree kernel, where nodes are weighted according to hierarchical information extracted from modern EDA tools' PPA reports. The model's predicted means and variances are grouped by their corresponding tree clusters, and the EHVI algorithm [12] is applied to select one suggestion from each group (detailed in Section V-B).



Labels	#1	#2
A	1	1
B	2	1
C	1	1
D	2	2
B A	2	1
D C	2	2
A BC	0	1
A BBC	1	0
C ADD	1	1

Fig. 3 The Weisfeiler-Lehman subtree kernel computation process with one iteration consists of three steps: (1) Initial RTL hierarchy trees representing two different microarchitectural implementations; (2) Subtree pattern extraction through WL relabeling, where each node is relabeled based on its neighbors; and (3) Feature counting of node and subtree labels to generate feature vectors. The similarity score is computed as the dot product of the two feature count vectors.

The multiple suggestions enable parallel execution of the most time-consuming evaluation phase. These suggestions are evaluated by generating complete RTL code using Chipyard [13]. The generated RTL code is then synthesized and simulated using commercial EDA tools to obtain PPA values and hierarchical reports, which are stored in the dataset for training the GP model in subsequent iterations.

The proposed flow takes advantage of the RTL-level architecture information and hierarchy reports generated from modern EDA tools, enabling more efficient microarchitecture design space exploration.

IV. RTL Hierarchy Aware Kernel

In this section, we first review the fundamental role of kernels in Gaussian Process regression and discuss the limitations of traditional kernels when applied to hardware design. We then present our RTL hierarchy aware kernel based on the Weisfeiler-Lehman subtree algorithm [14], which effectively captures the structural characteristics of hardware designs. Finally, we demonstrate the superiority of the WL subtree kernel over the conventional RBF kernel in microarchitecture DSE through theoretical analysis.

A. Kernels for Gaussian Process

In Gaussian process regression (GPR), kernels enable the quantification of similarities between different inputs, allowing for probabilistic predictions at unobserved points. For microarchitecture design space exploration using Bayesian Optimization (BO), kernels model the relationship between microarchitecture configurations.

The Radial Basis Function (RBF) kernel has been the de facto choice for traditional microarchitecture DSE:

$$k_{\text{RBF}}(x, x') = \exp\left(-\frac{|x - x'|^2}{2l^2}\right), \tag{2}$$

where l is the bandwidth parameter.

Algorithm 1 Weisfeiler-Lehman Subtree Kernel for RTL Hierarchies

Require: RTL hierarchy trees $\mathcal{G}$, $\mathcal{G}'$, number of iterations H
Ensure: Kernel value $k_{\text{WL}}(\mathcal{G}, \mathcal{G}')$

$k_{\text{WL}} \leftarrow 0$ ▷ *Initialize kernel value*
$\ell^{(0)}(v) \leftarrow \text{moduleType}(v)$ for all nodes v ▷ *Set labels*
for $h \leftarrow 0$ to H **do**
  $\Sigma_h \leftarrow \emptyset$ ▷ *Initialize set for current iteration labels*
  if $h > 0$ **then** ▷ *Skip label update in first iteration*
    for each node v in $\mathcal{G}$ and $\mathcal{G}'$ **do**
      $M_v \leftarrow$ collect and sort labels of neighbors of v
      $\ell^{(h)}(v) \leftarrow \text{HASH}(\ell^{(h-1)}(v), M_v)$ ▷ *Generate new label*
    end for
  end if
  for each node v in $\mathcal{G}$ and $\mathcal{G}'$ **do**
    $\Sigma_h \leftarrow \Sigma_h \cup \{\ell^{(h)}(v)\}$ ▷ *Collect all unique labels*
  end for
  Initialize count vectors $c_h(\mathcal{G})$ and $c_h(\mathcal{G}')$ with zeros
  for each label $\ell \in \Sigma_h$ **do**
    $c_h(\mathcal{G})_\ell \leftarrow$ count occurrences of label ℓ in $\mathcal{G}$
    $c_h(\mathcal{G}')_\ell \leftarrow$ count occurrences of label ℓ in $\mathcal{G}'$
  end for
  $k^{(h)} \leftarrow \sum_{\ell \in \Sigma_h} c_h(\mathcal{G})_\ell \cdot c_h(\mathcal{G}')_\ell$ ▷ *Compute similarity*
  $k_{\text{WL}} \leftarrow k_{\text{WL}} + k^{(h)}$ ▷ *Add to total similarity*
end for
return k_{WL}

B. Weisfeiler-Lehman Subtree Kernel for RTL Hierarchy Comparison

To address the limitations discussed in Section I, we propose using the Weisfeiler-Lehman (WL) subtree kernel, which naturally operates on the RTL hierarchy trees generated during the compilation of Chisel code.

Given two RTL hierarchy trees $\mathcal{G}$ and $\mathcal{G}'$, represented as labeled graphs where nodes correspond to hardware modules and edges represent hierarchical relationships, the WL subtree kernel computes their similarity through an iterative relabeling and counting process as shown in Algorithm 1. Fig. 3 gives a visualized example of how the algorithm works.

The WL subtree kernel is particularly well-suited for microarchitecture design space exploration, as it both preserves the inherent structural information of hardware designs and operates directly on hierarchy trees. Through its iterative process, the kernel naturally captures similarities at various hierarchical levels, from individual components to complex module arrangements.

C. Theoretical Analysis of WL Subtree Kernel for Microarchitecture

To establish the advantages of our approach, we first analyze the sample complexity of both RBF and WL kernels through Theorems 1 and 2. Then, in Theorem 3, we demon-

strate that the WL subtree kernel exhibits lower sample complexity compared to the RBF kernel in practical applications.

Theorem 1 (RBF Kernel Sample Complexity)**.** *To learn with RBF kernel on d-dimensional Chisel parameter vectors with accuracy ε and confidence $1-\delta$, the sample complexity is:*

$$N_{RBF} = O(\frac{d}{\varepsilon^2}\log(\frac{1}{\delta})). \tag{3}$$

Proof. This follows from the classic result of [15] on sample complexity bounds for RBF kernels in terms of their Rademacher complexity. □

Theorem 2 (WL Kernel Sample Complexity)**.** *For a WL kernel with H iterations on RTL trees of maximum height h and branching factor b, the sample complexity is:*

$$N_{WL} = O(\frac{H\log(b^h)+\log(\frac{1}{\delta})}{\varepsilon^2}). \tag{4}$$

Proof. By the fundamental theorem of statistical learning, for a hypothesis class with Vapnik-Chervonenkis (VC) dimension [16] d, the sample complexity is $O(\frac{d+\log(\frac{1}{\delta})}{\varepsilon^2})$. The VC dimension of the WL feature space after H iterations is bounded by $O(H\log(b^h))$. □

Theorem 3 (Relative Efficiency)**.** *For typical RTL designs where $d \gg H\log(b^h)$, the WL kernel achieves better sample complexity than the RBF kernel.*

Proof. Modern processors often have hundreds of parameters ($d \approx 200$) while their RTL hierarchies typically have height $h \approx 10$ (GemminiSoC [9]) and branching factor $b \approx 4$. With $H = 3$ iterations:

$$H\log(b^h) \approx 3\log(4^{10}) \approx 42 < d. \tag{5}$$

Given that $\log(\frac{1}{\delta}) \gg 1$ typically holds, we can conclude that $N_{\text{WL}} < N_{\text{RBF}}$ for practical RTL designs. □

This theoretical analysis establishes the superior sample efficiency of our WL kernel approach, particularly for complex RTL designs with deep hierarchies.

V. RTL Hierarchy Aware Gaussian Process and Clustering

In this section, we present two applications of the kernel introduced in the previous section. We first describe how we enhance the basic WL kernel with module-specific weights derived from evaluation reports in Gaussian process regression. We then detail our clustering approach that leverages the kernel's similarity measure to enable efficient parallel design space exploration.

A. Gaussian Process Regression with Weighted WL Kernel for PPA Prediction

Building upon the WL subtree kernel introduced in the previous section, we propose a weighted extension that better captures the varying importance of different modules in power and area prediction. While the basic WL kernel treats all modules equally when computing structural similarities,

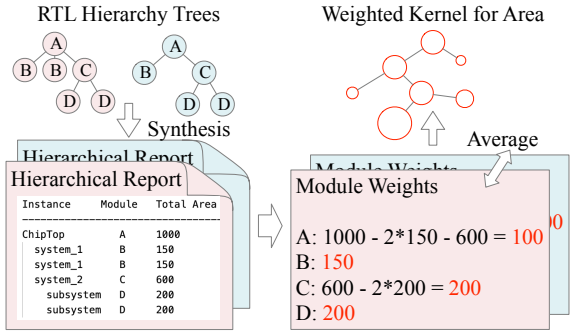


Fig. 4 Example of module weights' extraction in area prediction.

synthesis reports provide valuable information about each module's contribution to the overall PPA metrics.

Given a training dataset $\mathcal{D} = \{(\mathcal{G}_i, y_i)\}_{i=1}^n$, where $\mathcal{G}_i$ represents the RTL hierarchy tree and y_i denotes its corresponding PPA metric, we formulate the GPR model with the weighted WL subtree kernel as $f(\mathcal{G}) \sim \mathcal{GP}(0, k^w_{\text{WL}}(\mathcal{G}, \mathcal{G}'))$.

The weighted WL subtree kernel $k^w_{\text{WL}}(\mathcal{G}, \mathcal{G}')$ is enhanced by incorporating module-level importance weights:

$$k^w_{\text{WL}}(\mathcal{G}, \mathcal{G}') = \sum_{h=0}^{H} \lambda^h k^w_h(\mathcal{G}, \mathcal{G}'), \tag{6}$$

where $\lambda \in (0, 1]$ is a decay factor controlling the influence of higher iterations, and k^w_h is the weighted base kernel at iteration h:

$$k^w_h(\mathcal{G}, \mathcal{G}') = \sum_{\ell \in \Sigma_h} w(\ell) \cdot c^{(\ell)}_h(\mathcal{G}) \cdot c^{(\ell)}_h(\mathcal{G}'), \tag{7}$$

where module weights $w(\ell)$ are derived from synthesis reports, as shown in Fig. 4.

In Theorem 4, we demonstrate that the weighted WL kernel is semi-definite. Subsequently, in Theorem 5, we establish that semi-definite kernels are valid for use in Gaussian process regression models.

Theorem 4 (Positive Semi-definiteness of Weighted WL Subtree Kernel)**.** *The weighted WL subtree kernel $k^w_{WL}(\mathcal{G}, \mathcal{G}')$ is positive semi-definite for any non-negative weight $w(\ell)$.*

Proof. By expanding the weighted base kernel through Equation (7), we can prove that for any iteration h, the weighted base kernel $k^w_h(\mathcal{G}, \mathcal{G}')$ is positive semi-definite if $w(\ell) \geq 0$.

The weighted WL subtree kernel is defined as $k^w_{\text{WL}}(\mathcal{G}, \mathcal{G}') = \sum_{h=0}^{H} \lambda^h k^w_h(\mathcal{G}, \mathcal{G}')$, each k^w_h is positive semi-definite. Since $\lambda \in (0, 1]$, all coefficients λ^h are positive. Therefore, k^w_{WL} is positive semi-definite. □

Theorem 5. *The weighted WL subtree kernel defines a valid Reproducing Kernel Hilbert Space (RKHS) [17], ensuring the existence and uniqueness of the solution in the resulting Gaussian Process regression model.*

Proof. This follows directly from Moore-Aronszajn theorem [18], which establishes a one-to-one correspondence between positive semi-definite kernels and RKHSs. □

B. RTL Hierarchy Aware Design Space Clustering

The weighted WL subtree kernel provides a natural similarity measure that enables kernel k-means clustering [19] to identify distinct architectural patterns within the design space. This clustering capability serves two purposes: improving exploration efficiency and enabling parallel evaluation.

Given a target number of clusters k, the kernel k-means objective is formulated as:

$$\min_{C_1,\ldots,C_k} \sum_{j=1}^{k} \sum_{\mathcal{G}_i \in C_j} \|\phi_w(\mathcal{G}_i) - \mu_j\|^2, \tag{8}$$

where $\phi_w(\mathcal{G})$ is the implicit feature mapping induced by the weighted WL kernel.

The clustering is applied in two key steps:

1) Initial Design Space Sampling:

For initial design points selection, we select one representative design from each of the k clusters, where k is a hyperparameter. The selection strategy aims to identify designs that are most representative of their clusters' centers:

$$\mathcal{G}\text{init}^j = \text{argmin}\, \mathcal{G} \in C_j |\phi_w(\mathcal{G}) - \mu_j|^2. \quad j = 1, \ldots, k \tag{9}$$

2) Parallel Expected Hypervolume Improvement:

During optimization, we enhance the Expected Hypervolume Improvement (EHVI) acquisition function [12] through cluster-based multiple suggestions generation:

$$\mathcal{G}^j_{\text{next}} = \underset{\mathcal{G} \in C_j}{\text{argmax}}\, \text{EHVI}(\mathcal{G}|\mathcal{D}), \tag{10}$$

then the parallel suggestions are evaluated concurrently.

This cluster-based parallelization strategy maintains exploration diversity while enabling efficient concurrent evaluation of distinct designs, particularly beneficial for time-consuming PPA evaluations.

VI. Experiment

A. Experimental Setup

We conducted our experiments on a Gemmini-based RISC-V SoC [9]. The design space of this SoC is illustrated in Fig. 5. We evaluate the performance using a suite of language models spanning different scales, including Transformer-small [20], Bert-base [21], as well as larger variants like Transformer-large, Bert-large, and GPT1-small [22].

The hardware implementation flow starts with Chipyard (version 1.9.1) for converting Chisel-based hardware descriptions into synthesizable RTL. For performance evaluation, we use Spike simulator to obtain cycle-accurate measurements of model inference latency. The physical implementation leverages Cadence Genus 201 with ASAP7 7nm process technology for logic synthesis, while power analysis is performed using Cadence Joules 201.

Our design space exploration considers multiple optimization targets encompassing cycles, power and area. The architectural search space is defined by 26 configurable parameters, theoretically enabling 1.3×10^{11} possible Chisel implementations. Through feasibility analysis sampling, we constructed a representative offline dataset comprising 5198 valid configurations for comprehensive evaluation.

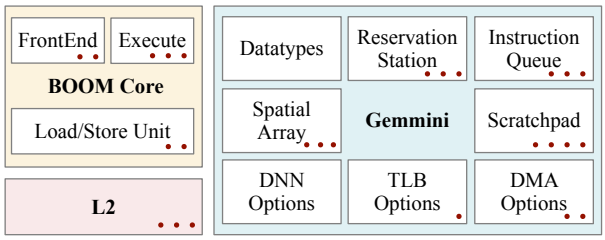


Fig. 5 Parameter selection of Gemmini-based RISC-V SoC. The blocks refer to sets of parameters controlling specific parts in the microarchitecture, while the dots refer to the number of parameters selected in our offline design space.

B. Comparison Against SOTA Methods

We compare our method with the following baselines for 100 times with each method assigned a total quota of 40 times evaluations (from chisel configuration to post-synthesis netlists, including performance simulation):

- DAC'16 [5]: A method combining statistical sampling and Adaboost-based active learning.
- ASPDAC'20 [6]: XGBoost-based method.
- DAC'23 [8]: A method that uses GNN to encode the design space into a latent space for Bayesian optimization.
- AAAI'24 [23]: Method based on reinforcement learning.

For fair comparisons between all baselines, hypervolume, which is one of the most widely used quality indicators in multi-objective optimization problems [24], is used as the indicator. A larger hypervolume indicates better exploration results. The following metrics are used to compare all methods: **HV** is the hypervolume of performance, power, and area. $\mathbf{HV}_{0,1}$ is the hypervolume of performance and power. $\mathbf{HV}_{0,2}$ is the hypervolume of performance and area. **MPP** is the maximum value of the product of the minimum clock period and the minimum power. **MPA** is the maximum value of the product of the minimum clock period and the minimum area. The results are normalized within the range of the shared initializing 5 data's scores to the highest scores. Evaluation quota is set as 40 for all methods.

We allocate the same evaluation quota rather than runtime to all methods because evaluation takes most of the runtime (more than 95%) in all these DSE methods. Giving the same evaluation quota eases the difficulty of understanding the experiment result.

TABLE I shows the comparison of our method with existing methods on Gemmini SoC while Fig. 6 gives the visualization. Given the same evaluation quota, our method achieves higher hypervolume, in other words, explores more design space than these methods. Our method is 29.3% higher than [5], 25.6% higher than [6], 12.9% higher than [8], 8.2% higher than [23] in hypervolume. The clustering algorithm facilitates parallel evaluation, allowing our method (shown by the orange curve) to achieve a k-fold reduction in total evaluation time, where k is the number of clusters, set as 3 in the experiment.

TABLE I Comparisons of Normalized DSE Performance against SOTA works.

Metrics	DAC'16 [5]	ASPDAC'20 [6]	DAC'23 [8]	AAAI'24 [23]	Ours w/o clustering	Ours w/ clustering
HV	0.775	0.796	0.884	0.928	0.967	**1**
$HV_{0,1}$	0.545	0.558	0.551	0.813	0.850	**1**
$HV_{0,2}$	0.553	0.557	0.689	0.833	0.870	**1**
MPP	0.814	0.763	0.907	0.904	0.942	**1**
MPA	0.899	0.791	0.976	0.917	0.974	**1**

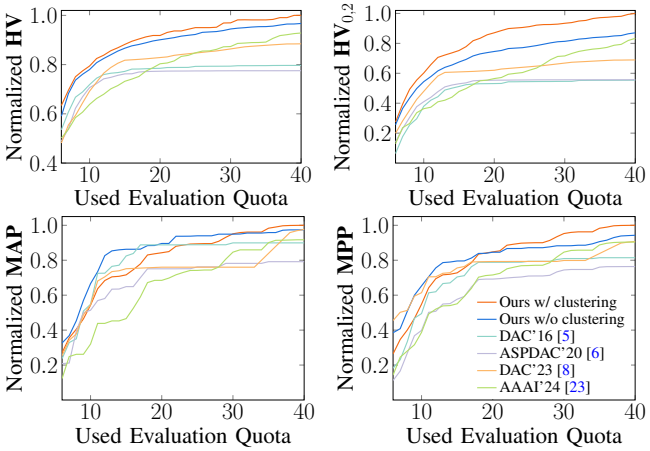


Fig. 6 The visualizations of normalized scores change with evaluations.

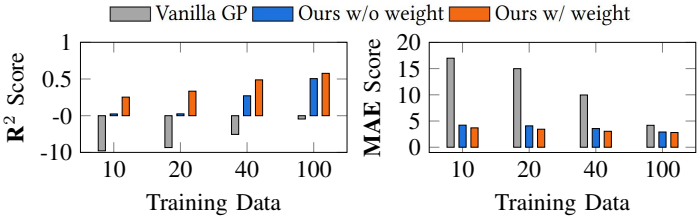


Fig. 7 Accuracies between the Vanilla GP model and our proposed method with small training set sizes. In the first chart, for better visualization, the bottom half of the y-axis is compressed by a factor of 10.

TABLE II Design Space's Clustering Results on PPA Space.

Metrics	K-means	RBF Kernel K-means	Ours
Silhouette	0.377	0.281	**0.597**
Davies-Bouldin	0.786	3.336	**0.473**

C. Ablation Studies

To rigorously evaluate our proposed approach, we conduct comprehensive ablation studies focusing on two key aspects: prediction accuracy under limited data conditions and clustering performance.

1) RTL Hierarchy Aware Gaussian Process:

We first investigate how our RTL hierarchy aware Gaussian Process performs under limited training data conditions, which is crucial for practical microarchitecture DSE. To show the proposed GP's performance in different stages of the exploration, we randomly select $(10, 20, 40, 100)$ training samples and evaluate the model's prediction accuracy using 500 samples from the remaining data as the test set. The test is repeated 200 times.

As the metrics of prediction accuracy, we use $\mathbf{R}^2$ (coefficient of determination) which measures the proportion of variance explained by the model, ranging from 0 to 1 where higher values indicate better fit, and **MAE** (Mean Absolute Error) which directly reflects the average prediction deviation.

Fig. 7 shows the average results of our proposed RTL hierarchy aware GP and Vanilla GP across different training set sizes. The results demonstrate that RTL hierarchy aware GP consistently outperforms the Vanilla GP model, particularly when training data is scarce.

2) RTL Hierarchy Aware Design Space Clustering:

We further evaluate the clustering capability of our approach compared to other clustering methods including standard k-means with RBF kernel. We assess the clustering quality using two widely adopted metrics: the **Silhouette** Score [25] which measures how similar an object is to its own cluster compared to other clusters (ranging from -1 to 1, higher is better), and the **Davies-Bouldin** Index [26] which evaluates intra-cluster similarity and inter-cluster differences (lower values indicate better clustering).

Considering that the inputs of our proposed method are graphs, while others are parameters, we compare the clustering results by testing the clustered labels with their corresponding PPA values, which is more practical in design space exploration. The number of clusters k is set as 3 and the test is repeated 50 times with different random states.

The results in TABLE II show that our RTL hierarchy aware Design Space Clustering consistently achieves better clustering quality. It proves that the proposed clustering method helps explore the design space, while naturally enabling parallel evaluation during the design space exploration.

VII. Conclusion

This paper presents a novel approach to microarchitecture design space exploration that fundamentally breaks away from traditional parameter-centric methods by leveraging the rich structural information contained in RTL hierarchy trees. We adopt WL subtree kernel to measure similarity of trees and theoretically prove that it is more efficient than RBF kernel in microarchitecture DSE. Gaussian process based on the weighted kernel extension incorporates the importance of a synthesis-derived module, improving exploration efficiency. An RTL hierarchy aware clustering method is developed to identify distinct architectural patterns and enable efficient parallel evaluation strategies. Experimental results and ablation studies on a Gemmini-based RISC-V SoC demonstrate the effectiveness of our approach, showing significant improvements over state-of-the-art methods.

REFERENCES

[1] J. Bachrach *et al.*, "Chisel: Constructing hardware in a scala embedded language," in *Proc. DAC*, 2012.

[2] C. Papon and Y. Xiao, "SpinalHDL." [Online]. Available: https://github.com/SpinalHDL/SpinalHDL

[3] M. Barros *et al.*, "Ga-svm feasibility model and optimization kernel applied to analog ic design automation," in *Proc. GLSVLSI*, 2007.

[4] B. C. Lee and D. M. Brooks, "Illustrative design space studies with microarchitectural regression models," in *Proc. HPCA*, 2007.

[5] D. Li *et al.*, "Efficient design space exploration via statistical sampling and adaboost learning," in *Proc. DAC*, 2016.

[6] Z. Xie *et al.*, "Fist: A feature-importance sampling and tree-based method for automatic design flow parameter tuning," in *Proc. ASPDAC*, 2020.

[7] C. Bai *et al.*, "Boom-explorer: Risc-v boom microarchitecture design space exploration framework," in *Proc. ICCAD*, 2021.

[8] X. Yi, J. Lu, X. Xiong, D. Xu, L. Shang, and F. Yang, "Graph representation learning for microarchitecture design space exploration," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*, 2023.

[9] H. Genc *et al.*, "Gemmini: Enabling systematic deep-learning architecture evaluation via full-stack integration," in *Proc. DAC*, 2021.

[10] M. Verleysen and D. François, "The curse of dimensionality in data mining and time series prediction," in *Computational Intelligence and Bioinspired Systems*, J. Cabestany, A. Prieto, and F. Sandoval, Eds., 2005.

[11] J. Mockus, "The application of bayesian methods for seeking the extremum," *Towards global optimization*, 1998.

[12] M. T. Emmerich *et al.*, "Hypervolume-based expected improvement: Monotonicity properties and exact computation," in *Proc. CEC*, 2011.

[13] A. Amid *et al.*, "Chipyard: Integrated design, simulation, and implementation framework for custom socs," *IEEE Micro*, 2020.

[14] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels." *Journal of Machine Learning Research*, 2011.

[15] P. L. Bartlett and S. Mendelson, "Rademacher and gaussian complexities: Risk bounds and structural results," *Journal of Machine Learning Research*, 2002.

[16] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth, "Learnability and the vapnik-chervonenkis dimension," *J. ACM*, vol. 36, no. 4, 1989.

[17] B. Schölkopf, "Learning with kernels: support vector machines, regularization, optimization, and beyond," 2002.

[18] N. Aronszajn, "Theory of reproducing kernels," *Transactions of the American mathematical society*, 1950.

[19] I. S. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means: spectral clustering and normalized cuts," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004.

[20] A. Vaswani *et al.*, "Attention is all you need," *Proc. NeurIPS*, 2017.

[21] 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.

[22] A. Radford *et al.*, "Improving language understanding by generative pre-training," https://openai.com/research/language-unsupervised/, 2018.

[23] C. Bai, J. Zhai, Y. Ma, B. Yu, and M. D. F. Wong, "Towards automated risc-v microarchitecture design with reinforcement learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

[24] M. Li and X. Yao, "Quality evaluation of solution sets in multiobjective optimisation: A survey," *ACM Comput. Surv.*, 2019.

[25] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, 1987.

[26] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1979.