IT-DSE: Invariant Risk Minimized Transfer Microarchitecture Design Space Exploration

Ziyang Yu[1], Chen Bai[1], Shoubo Hu[2], Ran Chen[2], Taohai He[3], Mingxuan Yuan[2], Bei Yu[1], Martin Wong[1]
[1]The Chinese University of Hong Kong [2]Huawei Noah's Ark Lab [3]HiSilicon

***Abstract*—The microarchitecture design of processors faces growing complexity due to expanding design space and time-intensive verification processes. Utilizing historical design task data can improve the search process, but managing distribution discrepancies between different source tasks is essential for enhancing the search method's generalization ability. In light of this, we introduce IT-DSE, a microarchitecture searching framework with the surrogate model pre-trained to absorb knowledge from previous design tasks. The Feature Tokenizer-Transformer (FT-Transformer) serves as a backbone, facilitating feature extraction from source tasks even with varied design spaces. Concurrently, the invariant risk minimization (IRM) paradigm bolsters generalization ability under data distribution discrepancies. Further, IT-DSE exploits a combination of multi-objective Bayesian optimization and a model ensemble to discover Pareto-optimal designs Experimental results indicate that IT-DSE effectively harnesses the knowledge of existing microarchitecture designs and uncovers designs that outperform previous methods in terms of power, performance, and area (PPA).**

I. INTRODUCTION

The microprocessor design cycle requires high workforce input. It involves performance, power, and area (PPA) co-optimization with microarchitecture design and physical implementation. Microarchitecture determines the detailed structures of a microprocessor. Finding out a microarchitecture to achieve a sweet trade-off for PPA as early as possible in the design cycle can reduce non-recurring engineering costs and meet the stringent product delivery deadline.

Nevertheless, escalating computational demands and decreasing feature sizes have largely amplified the complexity of the microarchitecture design task. In the industrial context, the prevalent approach relies on computer architects to manually configure design parameters, which requires both sufficient domain knowledge and a considerable amount of human labor. Besides, rapidly growing design spaces further pose challenges due to their enormity (e.g., with potential CPU designs reaching $\mathcal{O}(10^{40})$), complexity from inter-feature interactions, and the presence of mixed-type parameters leading to abrupt PPA changes. To alleviate these issues, researchers have employed machine learning-based process simulation models to hasten exploration and reduce search overhead [1]–[5], though their effectiveness is contingent upon the quality of training data and model granularity.

Practical optimization often targets specific microarchitectures [6], [7]. However, getting desired evaluation metric reports for one microarchitecture configuration through the very-large-scale integration (VLSI) flow is time-consuming, making diverse configurations challenging to acquire. An approach is proposed to address this by leveraging historical configurations for new tasks using cross-domain mixup and Artificial Neural Network (ANN) feature extraction [8]. This method, however, ignores intricate feature interactions, lacks interpretability, and presumes identical design spaces for source and target tasks, significantly limiting its applicability. We assert the necessity of integrating historical microarchitecture domain knowledge into modeling for facilitating efficient Design Space Exploration (DSE). Our principal insight arises from a commonality among mainstream microarchitectures: although the configuration distributions in different tasks vary, *the characteristics of certain design features have similar or even identical effects on PPA values in different generations of a microarchitecture design.* In other words, some design features maintain the same influence mechanism on PPA values for different microarchitectures.

The reasons behind the phenomenon lie in two folds. First, different generations of microprocessors are developed based on a common baseline microarchitecture. Minor architecture design improvements are applied across consecutive microarchitecture generations. As a result, the design space has not usually undergone significant changes between adjacent generations. For example, a single Skylake microarchitecture is scaled to support the server and client applications by Intel without modifying the underlying microarchitecture structures considerably [9]. AMD EPYC and Ryzen also follow a similar philosophy based on Zen microarchitecture [10]. Second, some general relations exist for design features due to the similar microarchitecture design paradigm. The microarchitecture design paradigm refers to the overall architecture design framework to drive the lifetime of an instruction from fetch to commit (*i.e.*, fetch, decode, rename, dispatch, issue, execution, write back, memory access, and commit). Although some speculative optimization techniques are different, *e.g.*, branch prediction [11], memory dependence prediction [12], prefetch mechanism [13], *etc.*, the general relations are applicable. Consider an out-of-order microarchitecture. The relations between the issue width and performance values follow a well-known *Instruction Window (IW) characteristic* [14], [15]. The IW characteristic unfolds the existence of a power law relationship among the performance and the issue width and benchmarks. Therefore, the general relations provide opportunities for transferring knowledge learned from

This work is supported by The Research Grants Council of Hong Kong SAR (No. CUHK14210723).

previous microarchitecture generations to a new generation.

These insights motivate us to extract that relationship that is approximately invariant among different historical source design tasks, even equipped with different design spaces. In this paper, we propose IT-DSE to boost the microarchitecture transferring design space exploration. Given source design tasks in either the same or different design space, to efficiently extract expressive features, we adopt Feature Tokenizer-Transformer (FT-Transformer) [16] as feature extraction backbone and design a scheme to transfer knowledge from the pre-trained model to a target task with a different design space following [17]. With out-of-distribution (OOD) explored configuration datasets, we pre-train the surrogate model using data from multiple source tasks. By combining the invariant risk minimization (IRM) paradigm with Bayesian inference [18], the surrogate model could learn task-invariant features that are stable even in the case of distributional shifts, improving the generalization ability to unseen target design space exploration task. The design space is explored via multi-objective Bayesian optimization flow [19]. Model ensemble strategy [20] is utilized to improve the uncertainty estimation, leading to finding optimal designs. This method can be applied to massive microarchitecture design spaces, enabling it to find—in the same amount of time—solutions that are superior to those identified by other search techniques. To the best of our knowledge, this is the first design space exploration method that could transfer domain knowledge from different source tasks, even if the design space of source tasks can be different from the target task.

Our major contributions are summarized as follows.

- We utilize the FT-Transformer as the feature extraction backbone. This model is capable of processing variable-length input parameters, allowing for its pretraining on multiple source tasks with diverse design spaces.
- We introduce a prediction model that follows the Bayesian invariant risk minimization paradigm to refine the learning objective, enhancing the surrogate model's generalizability.
- We employ a multi-objective Bayesian optimization with a pre-trained model ensemble as a surrogate model. This approach characterizes the microarchitecture design space and selects the Pareto-optimal set.
- Our framework is verified within complex industry microarchitecture design spaces. Experimental results exhibit excellent transfer performance across diverse design tasks.

II. Preliminaries

A. Typical Microprocessor Components

For the design of a microprocessor, there are serval key components. Fig. 1 exhibits a typical microarchitecture pipeline. In this paper, we mainly focus on searching for the optimal values relates to some of those typical components.

In the front end of the pipeline, the Instruction Fetch Unit (IFU) employs the L1 Instruction Cache (L1 I-Cache) to expedite instruction fetches. Decoded by the instruction decode unit, these instructions are poised for Out-of-Order (OoO) execution to alleviate data dependency constraints. Following this, in the back end of the pipeline, arithmetic computations are executed: the Integer Execution Unit (IEU) for integer and bitwise operations and the Floating-Point Unit (FPU) for floating-point functions, including trigonometric and logarithmic tasks. The Load/Store Unit (LSU) orchestrates data movements between the function unit and L1 D-Cache, leveraging the L1 Data Cache (L1 D-Cache) for swift access to regularly used data. To further streamline data retrieval, the L2 Cache (L2C), a secondary and larger cache, is employed, though at the expense of increased power usage and chip space. Collectively, these units engender efficient instruction execution and microprocessor performance enhancement.

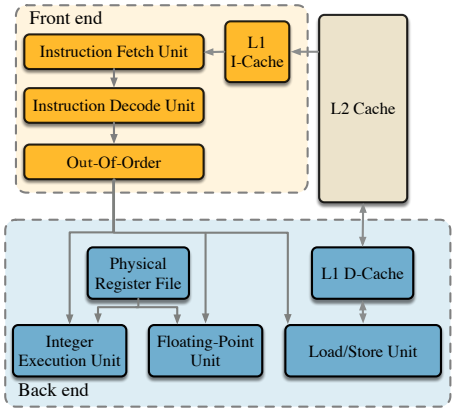


Fig. 1 Visualization of typical microarchitecture pipeline.

B. Invariant Risk Minimization

Invariant risk minimization (IRM) [21] paradigm allows for the automatic extraction of *invariant* features among different but related tasks. Let $\mathcal{X}$ and $\mathcal{Y}$ be the space of X and Y, respectively. Within IRM framework, the prediction function $f' : \mathcal{X} \to \mathcal{Y}$ consists of a feature extractor $\phi_{\boldsymbol{u}}(\cdot)$ and a regressor $h_{\boldsymbol{w}}(\cdot)$ parameterized by $\boldsymbol{u}$ and $\boldsymbol{w}$, respectively. The central idea is to learn a feature representation by $\phi_{\boldsymbol{u}}(\cdot)$ that facilitates the development of a regressor $h_{\boldsymbol{w}}(\cdot)$ optimal for all training tasks concurrently. To achieve this, $\phi_{\boldsymbol{u}}(\cdot)$ is optimized to effectively filter out spurious features, retaining only those that enhance the regressor's generalization capability across various tasks. This can be expressed as follows:

$$\begin{aligned} \min_{\boldsymbol{u},\boldsymbol{w}} \quad & \sum_{s=1}^{S} \mathcal{R}^s(\boldsymbol{u},\boldsymbol{w}), \\ \text{s.t.} \quad & \boldsymbol{w} \in \operatorname*{argmin}_{\boldsymbol{w}^s} \mathcal{R}^s(\boldsymbol{u},\boldsymbol{w}^s), \end{aligned} \tag{1}$$

where $\mathcal{R}^s(\boldsymbol{u},\boldsymbol{w}) = \mathbb{E}_{X^s,Y^s}[\mathcal{L}(f'(X^s),Y^s)]$ is the risk under s-th source task. $\mathcal{L}(\cdot)$ is an appropriate loss function. Equation (1) presents a bi-level optimization challenge, with each constraint necessitating an inner optimization, thereby

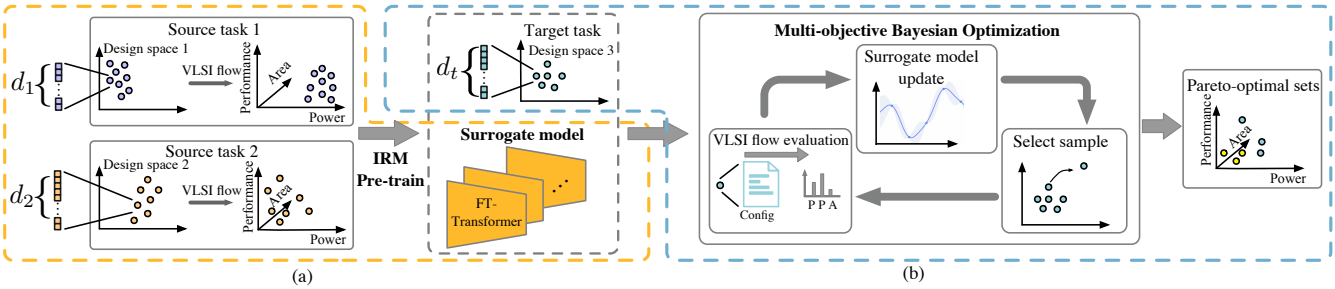


Fig. 2 Workflow of our proposed IT-DSE. (a) In the yellow block, source tasks 1 and 2, featuring search dimensions d_1 and d_2 respectively, are employed to pre-train the surrogate model via IRM. (b) In blue block, the pre-trained surrogate model is then utilized for Pareto-optimal set selection on the target task (with search dimension d_t), using multi-objective Bayesian optimization.

increasing problem-solving complexity. A simplified approximation, known as IRMv1, was proposed in [21]:

$$\min_{\boldsymbol{u},\boldsymbol{w}} \sum_{s=1}^{S} \mathcal{R}^s(\boldsymbol{u},\boldsymbol{w}) + \lambda ||\nabla_w \mathcal{R}^s(\boldsymbol{u},\boldsymbol{w})||^2, \tag{2}$$

where the risk $\mathcal{R}^s(\boldsymbol{u},\boldsymbol{w})$ from s-th source task is the negative log likelihood of data: $\mathcal{R}^s(\boldsymbol{u},\boldsymbol{w}) = -\ln p(\mathcal{T}^s|\boldsymbol{w},\boldsymbol{u}) = -\sum_{i=1}^{n_s} \ln p(\boldsymbol{y}_i^s|\boldsymbol{w},\phi(\boldsymbol{x}_i^s))$. So the objective is to learn the optimal parameters $\boldsymbol{w}$ and $\boldsymbol{u}$ to maximize the likelihood of source task samples.

C. Problem Formulation

In this work, we focus on improving the performance of microarchitecture design space exploration in target tasks with the help of transferring information from source tasks.

Definition 1 (Source Task)**.** *The source tasks are previously explored microarchitecture designs tasks. The s-th source task is composed of the explored sample dataset $\mathcal{D}^s$ containing n_s parameter vectors $X^s = [\boldsymbol{x}_1^s, \ldots, \boldsymbol{x}_{n_s}^s]^\top \in \mathbb{R}^{n_s \times d_s}$ and the evaluated PPA vectors $Y^s = [\boldsymbol{y}_1^s, \ldots, \boldsymbol{y}_{n_s}^s] \in \mathbb{R}^{n_s \times 3}$. The design space contains d_s searching dimensions.*

Definition 2 (Target Task)**.** *The target task is the new microarchitecture design space exploration task with only the set of legal parameter configuration vectors $X^t = [\boldsymbol{x}_1^t, \ldots, \boldsymbol{x}_{n_t}^t]^\top \in \mathbb{R}^{n_t \times d_t}$ available. For the t-th target design space, the searching dimension is d_t.*

Definition 3 (Pareto Optimality)**.** *For a multi-objective minimization problem with M objectives, a solution $\boldsymbol{x}_1$ is deemed to dominate solution $\boldsymbol{x}_2$ if, for all m belonging to the set $\{1, ..., M\}$, the inequality of objective vectors $f_m(\boldsymbol{x}_1) \leqslant f_m(\boldsymbol{x}_2)$ holds true, and there exists at least one k within the same set such that $f_k(\boldsymbol{x}_1) < f_k(\boldsymbol{x}_2)$; this relationship is symbolized by $\boldsymbol{x}_1 \succeq \boldsymbol{x}_2$. The collection of solutions that remain non-dominated by others constitutes the Pareto-optimal set, thereby establishing Pareto optimality within the design space. This assemblage of Pareto-optimal solutions, which represents the optimal balance of competing objectives, is referred to as the Pareto front.*

Given the defined two kinds of design tasks and the concept of Pareto optimality, we can formulate our transferring microprocessor design space exploration problem.

Problem 1 (Microarchitecture Transferring Design Space Exploration)**.** *Given S source tasks with explored datasets $\{\mathcal{D}^1, \mathcal{D}^2, \ldots, \mathcal{D}^S\}$ and the target sample $\mathcal{D}^t$, the objective is to utilize the information from historical source tasks and improve the efficiency of finding a series of microprocessor design configurations X in the target task that forms the Pareto optimality among the associated subset of $Y \subset \mathcal{Y}$, so that $X = \{\boldsymbol{x} | \boldsymbol{x} \succeq \boldsymbol{x}', \forall \boldsymbol{x}' \in X^t\}, Y = \{f(\boldsymbol{x}) | X \in X^t\}$.*

III. Invariant Risk Minimization-based Design Space Exploration Framework

A. Overview

The workflow of our design space exploration framework IT-DSE, which could incorporate knowledge from multiple heterogeneous source tasks, is illustrated in Fig. 2. Each source task consists of configurations of its own design space and their corresponding PPA values. To accommodate varying design spaces, the FT-Transformer ensemble is adopted as surrogate model for robust feature extraction and uncertainty estimation, thereby enhancing prediction reliability. To warm-start a new task by knowledge fusion, the invariant risk minimization paradigm is employed for surrogate model pre-training to improve its generalization ability on the target task with different data distribution and design space. This pre-trained surrogate model is then transferred to the target task for design space exploration, using a multi-objective Bayesian optimization approach. The output of IT-DSE comprises the explored microarchitectures from the iterative optimization process, with Pareto optimality determined from this set.

B. Customized Surrogate Model with Transformer

It is straightforward to improve the DSE efficiency by surrogate model pre-training. However, in real-world industry settings, the number of search dimensions in the target task often varies in comparison to the source task design space, which makes surrogate models trained on source tasks not directly applicable. To address this issue and facilitate the

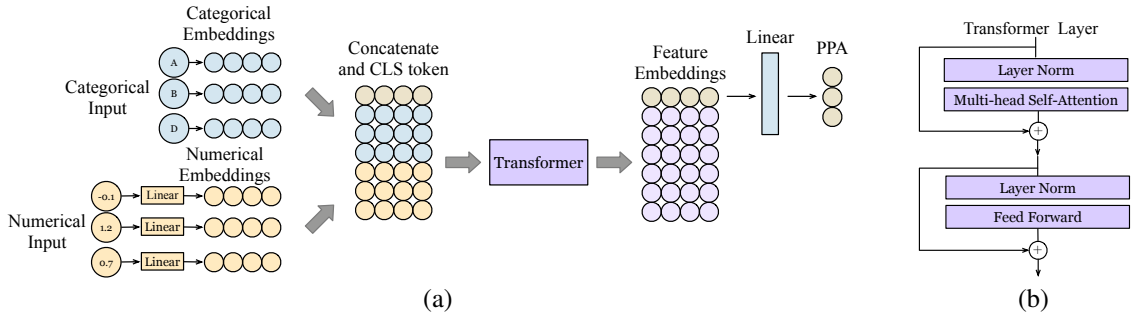


Fig. 3 (a) FT-Transformer architecture. Firstly, Feature Tokenizer transforms input parameters into embeddings. The embedding is then processed by the Transformer module. (b) One Transformer layer.

extraction of features from the source task while guiding the search for optimal samples in the target task, we propose a customized surrogate model. Inspired by recent FT-Transformer advancements [16], [17], our model uses it as a feature extractor, jointly pre-training on multiple source tasks to enhance exploration process efficiency.

FT-Transformer Architecture: Feature Tokenizer (FT)-Transformer is the attention-based model with adaption for the tabular domain. It consists of a Feature-Tokenizer layer and multiple Transformer layers, followed by a prediction layer, as can be seen from Fig. 3.

The Feature Tokenizer layer accepts input parameter vector $\boldsymbol{x}$ containing both numerical parameters and categorical parameters. For i-th numerical parameter x_i^{num}, we compute the feature embedding $\boldsymbol{t}_i^{num} \in \mathbb{R}^d$ via element-wise multiplication with corresponding m-th embedding weight vector $\boldsymbol{w}_m \in \mathbb{R}^d$, followed by adding bias term, b_i^{num}:

$$\boldsymbol{t}_i^{num} = x_i^{num} \cdot \boldsymbol{w}_i^{num} + b_i^{num}. \tag{3}$$

In the above equation, d represents the parameter embedding dimension. For the j-th categorical parameter x_j^{cat}, we query it from lookup table $\boldsymbol{w}_j^{cat}$, and added with a bias term b_j^{cat} as follows:

$$\boldsymbol{t}_j^{cat} = \boldsymbol{e}_j^\top \boldsymbol{w}_j^{cat} + b_j^{cat}, \tag{4}$$

where $\boldsymbol{e}_j^\top$ is the one-hot vector with only j-th element is 1 while all others are 0. The transformed embedding $\boldsymbol{T} \in \mathbb{R}^{(1+d_n+d_c)\times d}$ for the vector $\boldsymbol{x}$ is the stack of all the transformed element prepended with the output `[CLS]` token [22], which serves as the aggregate representation of the input vector for the purpose of feature extraction:

$$\boldsymbol{T} = \text{stack}(\texttt{[CLS]}, \boldsymbol{t}_1^{num}, \ldots \boldsymbol{t}_{d_n}^{num}, \boldsymbol{t}_1^{cat}, \ldots, \boldsymbol{t}_{d_c}^{cat}). \tag{5}$$

Given a batch of transformed embeddings from the same or different source tasks, our goal is to identify the relationships and dependencies between embeddings, regardless of their distance within the sequence. Considering that, we apply the stacked Transformer layers on the transformed embedding $\boldsymbol{T}$ to extract the features, utilizing its self-attention mechanism to simultaneously analyze all input features, thereby capturing local and global dependencies. This process fosters versatile feature extraction, making it suitable for tasks from various source tasks. Additionally, layer normalization and skip-connection techniques within the Transformer mitigate the vanishing gradient problem common in deep extractors, facilitating efficient training.

FT-Transformer Pre-training on multiple source tasks: To preprocess data from multiple source tasks for pre-training the FT-Transformer model, we adopt adaptive power transformation over linear transformation to better handle heteroscedasticity within each objective. Box-Cox transformation [23] is applied when the i-th objective values are consistently negative or positive, while Yeo-Johnson's transformation [24] is used for mixed positive and negative i-th objective values, accommodating diverse value ranges.

Upon normalizing the source task datasets, a multi-source FT-Transformer is initialized. Within the Feature-tokenizer layer, the embedding table represents the comprehensive union of features extracted from all source tasks under consideration. As a simple example, for source task 1 with only parameter vector $\boldsymbol{x}^1 = [x_1, x_2]$ and source task 2 with $\boldsymbol{x}^2 = [x_2, x_3, x_4]$, the embedding weight is $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3, \boldsymbol{w}_4]^\top \in \mathbb{R}^{4\times d}$, together with embedding bias $\boldsymbol{B} = [b_1, b_2, b_3, b_4]^\top \in \mathbb{R}^{4\times d}$. The transformed embeddings $\boldsymbol{T}^1$ and $\boldsymbol{T}^2$ for $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$ are $\boldsymbol{T}^1 = \text{stack}(\texttt{[CLS]}, \boldsymbol{t}_1, \boldsymbol{t}_2)$ and $\boldsymbol{T}^2 = \text{stack}(\texttt{[CLS]}, \boldsymbol{t}_2, \boldsymbol{t}_3, \boldsymbol{t}_4)$, respectively. The calculation for each element depends on the feature datatype, as shown in Equations (3) and (4).

For each PPA metric, we pre-train the FT-Transformer with linear output layer to optimize a pre-designed objective function, which will be discussed in the next section. Once the pre-training is finished, the model can be transferred to a downstream target optimization task. To accomplish this, the Transformer layers, the linear output layer, and the `[CLS]` embedding within the Feature Tokenizer layer are utilized to initialize the target FT-Transformer model. For target task parameters that are already present in the source tasks, the corresponding source embedding vectors are copied to the target Feature Tokenizer layer, ensuring a seamless integration of prior knowledge. For previously unseen target task parameters, a mix-up initialization strategy is implemented. The embeddings are chosen as the convex

combination of two randomly selected source embedding vectors, utilizing uniformly distributed weights.

The model is subsequently fine-tuned with the data from the source task by employing the negative log-likelihood loss at the outset of each Bayesian Optimization iteration. The fine-tuned FT-Transformer model serves as a surrogate model, aiding in the recommendation of samples.

C. Bayesian Invariant Risk Minimized Feature Extraction

Due to the limits of computational cost and strict time budget, the number of explored configurations in each source task will not be large in real industry. Therefore, a common practice is to merge historical data from multiple source tasks to incorporate valuable knowledge into the surrogate model. However, using this simple merging strategy and learning a predictor that minimizes the training error across the pooled data, which is called empirical risk minimization (ERM) principle [25], will lead to unstable task-specific features extraction. To this end, this section describes how to learn from data of multiple previous tasks the invariant relationship between microarchitecture parameters and PPA objectives by surrogate model pre-training, to further improve the efficiency of DSE tasks.

Bayesian Inference Extension: The IRM paradigm is proposed to learn from multiple tasks the invariant feature to improve models' generalization ability. However, directly applying IRM on our FT-Transformer model would encounter an overfitting issue [18]. As discussed in [26], [27], Bayesian inference is a widely recognized technique for mitigating overfitting, it can achieve optimal sample complexity rates in cases of model misspecification, highlighting its efficacy in addressing uncertainty and enhancing generalization. Considering that, we adopt the extended variant of IRMv1 proposed in [18] by incorporating the Bayesian principle.

Given the explored dataset from s-th source task $\mathcal{D}^s = \{X^s, Y^s\} = \{\boldsymbol{x}_i^s, \boldsymbol{y}_i^s\}_{i=1}^{n_s}$, we use $\mathcal{T}^s$ to represent the data from s-th source task transformed by the feature extractor: $\mathcal{T}^s = \{\phi_{\boldsymbol{u}}(\boldsymbol{x}_i^s), \boldsymbol{y}_i^s\}_{i=1}^{n_s}$. The collection of transformed datasets of all S source tasks is represented as $\mathcal{T}^c = \cup_{s=1}^{S} \mathcal{T}^s$.

If the feature extractor focuses more on task-specific features, the distribution of transformed data $\mathcal{T}^s$ and the posterior of predictor given the feature representation $p(\boldsymbol{w}^s|\mathcal{T}^s)$ will vary across tasks s. This alteration causes the $\boldsymbol{w}^s$ parameter-based regressor to rely on the particular s-th source task, compromising unseen target task generalization. Enhanced generalization requires efficient learning of task-invariant features, leading to close posterior $p(\boldsymbol{w}^s|\mathcal{T}^s)$ among different source tasks. This should finally lead to the approximate equivalence between the posterior from each source task to the desired shared posterior: $p(\boldsymbol{w}^s|\mathcal{T}^s) \approx p(\boldsymbol{w}|\mathcal{T}^c)$. The regressor characterized by parameter $\boldsymbol{w}$ will not depend on a certain source task. We use $g_{\boldsymbol{u}}(\boldsymbol{w}) \approx p(\boldsymbol{w}|\mathcal{T}^c)$ and $g_{\boldsymbol{u}}^s(\boldsymbol{w}^s) \approx p(\boldsymbol{w}^s|\mathcal{T}^s)$ to represent the approximate posterior distribution for regressor given the transformed dataset $\mathcal{T}^c$ and $\mathcal{T}^s$. The objective of our task-invariant feature extractor can be expressed as:

$$\max_{\boldsymbol{u}} \sum_s \mathbb{E}_{g_{\boldsymbol{u}}(\boldsymbol{w})}[\ln p(\mathcal{T}^s|\boldsymbol{w},\boldsymbol{u})] + \lambda \left(\mathbb{E}_{g_{\boldsymbol{u}}(\boldsymbol{w})}[\ln p(\mathcal{T}^s|\boldsymbol{w},\boldsymbol{u})] - \mathbb{E}_{g_{\boldsymbol{u}}^s(\boldsymbol{w}^s)}[\ln p(\mathcal{T}^s|\boldsymbol{w}^s,\boldsymbol{u})]\right). \tag{6}$$

The expectation terms in Equation (6) are the expected log likelihood of $g_{\boldsymbol{u}}^s(\boldsymbol{w}^s)$ and $g_{\boldsymbol{u}}(\boldsymbol{w})$:

$$\mathbb{E}_{g_{\boldsymbol{u}}^s(\boldsymbol{w}^s)}[\ln p(\mathcal{T}^s|\boldsymbol{w}^s,\boldsymbol{u})] = \int \ln p(\mathcal{T}^s|\boldsymbol{w}^s,\boldsymbol{u}) g_{\boldsymbol{u}}^s(\boldsymbol{w}^s) d\boldsymbol{w}^s,$$

$$\mathbb{E}_{g_{\boldsymbol{u}}(\boldsymbol{w})}[\ln p(\mathcal{T}_I^s|\boldsymbol{w},\boldsymbol{u})] = \int \ln p(\mathcal{T}_I^s|\boldsymbol{w},\boldsymbol{u}) g_{\boldsymbol{u}}(\boldsymbol{w}) d\boldsymbol{w}.$$

We can observe that the first term in Equation (6) maximizes the expected log-likelihood of the shared posterior $g_{\boldsymbol{u}}(\boldsymbol{w})$ by optimizing over feature extractor parameter $\boldsymbol{u}$, encouraging $\boldsymbol{u}$ to retain information for data distribution fitting. The second term in Equation (6) is a penalty regularized with weight coefficient λ. It requires transformed data distribution to be stable among different tasks with feature extractor $\phi_{\boldsymbol{u}}(\cdot)$, thus necessitates learning task-invariant features. The standard IRM formulation in Equation (2) utilizes a single point estimation of regressor parameter $\boldsymbol{w}$, which may lead to instability in situations with insufficient explored data from source tasks. In contrast, the feature extractor using Bayesian inference of IRM is derived by directly incorporating posterior distributions, offering a more robust approach that is less susceptible to overfitting while still capturing the underlying relationships within the data.

Adaptive Solution: The actual posterior $p(\boldsymbol{w}|\mathcal{T}^c)$ and $p(\boldsymbol{w}^s|\mathcal{T}^s)$ are hard to estimate in large models. To mitigate the issue, we use variational inference to approximate the posterior distributions by maximizing the evidence lower bound (ELBO). By doing so, we can express the objective function to optimal estimation of $g_{\boldsymbol{u}}^s(\boldsymbol{w}^s)$ and $g_{\boldsymbol{u}}(\boldsymbol{w})$ from a set of distribution $\mathcal{G}$:

$$g_{\boldsymbol{u}}^s(\boldsymbol{w}^s) = \operatorname*{argmax}_{g' \in \mathcal{G}} \mathbb{E}_{g'}[\ln p(\mathcal{T}^s|\boldsymbol{w},\boldsymbol{u}) - \mathrm{D}_{\mathrm{KL}}(g'||p_0(\boldsymbol{w}))], \tag{7}$$

$$g_{\boldsymbol{u}}(\boldsymbol{w}) = \operatorname*{argmax}_{g'} \sum_{s=1}^{S} \mathbb{E}_{g' \in \mathcal{G}}[\ln p(\mathcal{T}^s|\boldsymbol{w},\boldsymbol{u}) - \mathrm{D}_{\mathrm{KL}}(g'||p_0(\boldsymbol{w}))], \tag{8}$$

where we assume the prior of regressor parameters $p_0(\boldsymbol{w})$ are the same for each individual source task and the collection of source tasks. The first terms in the equation serve to maximize the expected log-likelihood of the posterior distributions, while the second term use Kullback-Leibler (KL) divergence $\mathrm{D}_{\mathrm{KL}}(g'||p_0(\boldsymbol{w}))$ to minimize the statistical distance between estimated posterior g' and prior $p_0(\boldsymbol{w})$. Adopting mean-field approximation which is a standard practice in variational inference [28], we select the factorized Gaussian distributions as a set of candidate distributions: $\mathcal{G} = \{\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) : \boldsymbol{\mu} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{n_w}]^\top, \boldsymbol{\Sigma} = \mathrm{diag}(\boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_{n_w})\}$, where n_w is the

dimensions of regressor parameter $\boldsymbol{w}$. The prior is set to a zero-mean Gaussian distribution: $p_0(\boldsymbol{w}) = \mathcal{N}(\mathbf{0}, \sigma \boldsymbol{I})$.

However, it is still computationally demanding to maximize ELBO in Equation (8) on all source tasks. Considering that during the training process, $g_{\boldsymbol{u}}^s(\boldsymbol{w}^s)$ approaches $g_{\boldsymbol{u}}(\boldsymbol{w})$ when more task-invariant features are extracted by $\phi_{\boldsymbol{u}}(\cdot)$. This makes it plausible to adopt the adaption solution with model-agnostic meta-learning in [29] to efficiently estimate $g_{\boldsymbol{u}}^s(\boldsymbol{w}^s)$, as can be expressed:

$$g_{\boldsymbol{u}}^s(\boldsymbol{w}^e) = \mathcal{N}(\boldsymbol{\mu} - \nabla_{\boldsymbol{\mu}} \mathbb{E}_{g_{\boldsymbol{u}}(\boldsymbol{w})} \ln p(\mathcal{T}^s | \boldsymbol{w}, \boldsymbol{u}), \boldsymbol{\Sigma}), \quad (9)$$

where we assume $g_{\boldsymbol{u}}(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ in the above equation we use one step of gradient descent of $\boldsymbol{\mu}$ on the transformed data to represent the mean of the Gaussian distribution for s-th source task.

To estimate the penalty term in Equation (6), we use Monte Carlo samples from posterior $g_{\boldsymbol{u}}(\boldsymbol{w})$ and $g_{\boldsymbol{u}}^s(\boldsymbol{w}^s)$. Considering that the expectation term between $g_{\boldsymbol{u}}(\boldsymbol{w})$ and $g_{\boldsymbol{u}}^s(\boldsymbol{w}^s)$ is close, we can use the reparameterization trick [30] with same noise variable ϵ:

$$\boldsymbol{w} = \boldsymbol{\mu} + \epsilon \boldsymbol{\Sigma}, \quad \boldsymbol{w}^s = \boldsymbol{\mu}^s + \epsilon \boldsymbol{\Sigma}^s. \quad (10)$$

Given the above discussion, we can express the full algorithm in Algorithm 1.

Algorithm 1 Feature extraction with Bayesian IRM

Require: Feature extractor $\phi_{\boldsymbol{u}}$, regressor $h_{\boldsymbol{w}}$, collection of data from S source task $\{\mathcal{D}^s\}_{s=1}^S$
Ensure: the learned task-invariant feature extractor $\phi_{\boldsymbol{u}}$, regressor $h_{\boldsymbol{w}}$.
Initialization: prior $\phi_0 \leftarrow \mathcal{N}(\mathbf{0}, \sigma \boldsymbol{I})$.
repeat
  for s in $1, \ldots, S$ **do**
    Sample a batch of data $(X_{batch}^s, Y_{batch}^s)$ from $\mathcal{D}^s$;
    Transform data $\mathcal{T}_{batch}^s \leftarrow (\phi_{\boldsymbol{u}}(X_{batch}^s), Y_{batch}^s)$;
  end for
  Update $g_{\boldsymbol{u}}(\boldsymbol{w})$; ▷ Equation (8)
  for s in $1, \ldots, S$ **do**
    Update $g_{\boldsymbol{u}}^s(\boldsymbol{w})$; ▷ Equation (9)
  end for
  Sample $g_{\boldsymbol{u}}(\boldsymbol{w})$ and $g_{\boldsymbol{u}}^s(\boldsymbol{w})$; ▷ Equation (10)
  Update $\boldsymbol{u}$ to maximize objective; ▷ Equation (6)
until Training finished.
Return $\phi_{\boldsymbol{u}}$, $h_{\boldsymbol{w}}$;

D. Target Task Design Space Exploration in Feature Space

The features extracted from FT-Transformer convert raw parameters into a more expressive and manageable latent feature space. To efficiently searching optimal points from latent space, we adopt the Bayesian Optimization (BO) algorithm to conduct efficient sampling, aiming to suggest potential candidates for evaluation. BO algorithms hinge on a surrogate model, which statistically approximates the optimization target using past data, and an acquisition function. This function adjusts the balance between exploring untested design spaces and exploiting known beneficial regions.

Deep Ensemble as Surrogate Model: Substantial variability issues inherent in single-model PPA value prediction can influence decision-making and predictive ability, particularly with task-specificity. To counter this, we employ a customized deep model ensemble [20] as our surrogate, aggregating predictions from multiple models for enhanced robustness and uncertainty estimation.

In this paper, we ensemble multiple pre-trained FT-Transformers directly as our surrogate model. Given n evaluated samples $\mathcal{D}^t = \{X^t, f'(X^t)\}$ from target task, where $X^t = \{(\boldsymbol{x}_1^t)^\top, (\boldsymbol{x}_2^t)^\top, \ldots, (\boldsymbol{x}_n^t)^\top\}^\top \in \mathbb{R}^{n \times d_t}$ and $f'(X^t) = \{(f'(\boldsymbol{x}_1^t))^\top, f'((\boldsymbol{x}_2^t))^\top, \ldots, f((\boldsymbol{x}_n^t))^\top\}^\top \in \mathbb{R}^{n \times 3}$, when a new point $\boldsymbol{x}_*^t$ is given, the posterior distribution $p(\boldsymbol{y}_*^t | \boldsymbol{x}_*^t)$ is approximated as a Gaussian distributions:

$$p(\boldsymbol{y}_*^t | \boldsymbol{x}_*^t) = \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{y}_*^t | f'(X^t)}, \boldsymbol{\sigma}^2_{\boldsymbol{y}_*^t | f'(X^t)}\right), \quad (11)$$

which is a uniformly-weighted mixture of M ensemble models. To improve the uncertainty estimation, each FT-Transformer model output both the predicted mean $\boldsymbol{\mu}_{\theta_m}(\boldsymbol{x}_*^t)$ and the inherent noise term $\boldsymbol{\sigma}^2_{\theta_m}(\boldsymbol{x}_*^t)$. With the ensemble models, we can express the posterior mean and variance:

$$\boldsymbol{\mu}_{\boldsymbol{y}_*^t | f'(X^t)} = \frac{1}{M} \sum_{m=1}^{M} \boldsymbol{\mu}_{\theta_m}(\boldsymbol{x}_*^t), \quad (12)$$

$$\begin{aligned} \boldsymbol{\sigma}^2_{\boldsymbol{y}_*^t | f'(X^t)} = & \frac{1}{M} \sum_{m=1}^{M} (\boldsymbol{\mu}_{\boldsymbol{y}_*^t | f'(X^t)} - \boldsymbol{\mu}_{\theta_m}(\boldsymbol{x}_*^t))^2 \\ & + \frac{1}{M} \sum_{m=1}^{M} \boldsymbol{\sigma}^2_{\theta_m}(\boldsymbol{x}_*^t). \end{aligned} \quad (13)$$

In Equation (13) the first term called epistemic uncertainty estimates the uncertainty that arises from the limit number of observed samples, while the second term, aleatoric uncertainty, is associated with the inherent noise in the observed sample. By combing these two terms improve the uncertainty estimation and allows for a more informed and robust decision-making process.

Pareto EHVI Acquisition Function: Expected Hypervolume Improvement (EHVI) [31] acquisition function is widely used to sample Pareto-optimal set for negatively correlated objectives. The hypervolume corresponds to the M-dimensional *Lebesgue measure* λ_M of the space dominated by an approximate Pareto frontier $\mathcal{P}(\boldsymbol{y})$ and constrained from below by a reference point $\boldsymbol{y}_{ref} \in \mathbb{R}^M$. This can be expressed as:

$$HV_{\boldsymbol{y}_{ref}}(\mathcal{P}(\boldsymbol{y})) = \lambda_M(\cup_{\boldsymbol{y} \in \mathcal{P}(\boldsymbol{y})}[\boldsymbol{y}, \boldsymbol{y}_{ref}]), \quad (14)$$

where $[\boldsymbol{y}, \boldsymbol{y}_{ref}]$ is the hyperrectangle bounded by $\boldsymbol{y}$ and reference point $\boldsymbol{y}_{ref}$. The hypervolume improvement (HVI) for a new given point $\boldsymbol{y}_*$ given previous Pareto set $\mathcal{P}(\boldsymbol{y})$ can be expressed:

$$HVI(\boldsymbol{y}_* | \mathcal{P}(\boldsymbol{y}), \boldsymbol{y}_{ref}) = V_{\boldsymbol{y}_{ref}}(\mathcal{P}(\boldsymbol{y} \cup \boldsymbol{y}_*)) - V_{\boldsymbol{y}_{ref}}(\mathcal{P}(\boldsymbol{y})). \quad (15)$$

The EHVI computes the expectation of HVI over the posterior distribution $P(f'(\boldsymbol{x}_*)|\mathcal{D})$. In practice, EHVI is approximated using Monte Carlo integration with K sampling times:

$$\begin{aligned} EHVI(\hat{\boldsymbol{x}}) &= \mathbb{E}_{P(f'(\hat{\boldsymbol{x}}_*)|\mathcal{D})}\left(HVI(f'(\hat{\boldsymbol{x}})|\mathcal{P}(\boldsymbol{y}), \boldsymbol{y}_{ref})\right) \\ &\approx \frac{1}{n_k}\sum_{k=1}^{K} HVI(\tilde{f}'_k(\hat{\boldsymbol{x}})|\mathcal{P}(\boldsymbol{y}), \boldsymbol{y}_{ref}). \end{aligned} \tag{16}$$

In above equation, $\tilde{f}'_k$ is the k-th function evaluation from distribution $p(f'(\boldsymbol{x}|\mathcal{D}))$. To further reduce the computational overhead for estimation of EHVI in the whole design dataset, in each BO iteration, we first select non-dominated potential candidates $\hat{X}$ with the mean of predicted function, then we traverse each potential candidate and select the candidate $\boldsymbol{x}_*$ with largest EHVI. In this way, we efficiently suggest new points with potentially desired PPA values.

IV. Experimental Results

We conduct experiments to evaluate the transfer performance of proposed design space exploration method.

The experiments are performed on an in-house 64-bit high-performance commercial microprocessor[1]. We generate different microarchitecture configurations from a design space specification sheet provided by experienced architects. We leverage the VLSI flow to evaluate the PPA values of each microarchitecture configuration. A single VLSI flow costs from days to weeks to evaluate each design. For each microarchitecture configuration, we generate the register-transfer-level (RTL) design from Chisel [32]. We use commercial electronic design automation (EDA) tools to perform logic synthesis, place, and routing with an industrial technology node. The performance and power values are obtained from the benchmark simulation after the physical implementation. And the area value is also reported from the physical implementation tool. Due to the commercial confidentiality of microarchitecture, we normalize the original PPA values.

A. Transfer Design Space Exploration Experiment Setup

Design Tasks: Considering the substantial computational resources required for online validation of each sampled microarchitectural design, we generate offline datasets to facilitate the design process for both source tasks and target tasks. To prove the efficiency of our transferring design space exploration method, we collect four design task datasets. Task A, task B, and Task C share the same design space which is extremely complex. TABLE I lists the number of parameters for each datatype in each interested microarchitecture component, together with the number of combinations. As can be seen from the table, there exist 55 parameters with more than 3×10^{40} combinations, including ordinal, exponential (with base 2) and categorical parameters. For task D, we can only tune 53 parameters out of 55 parameters listed in TABLE I. The tasks' data distribution varies significantly, as can be seen from the tSNE visualization Fig. 4(a). Task A, task B, task C, and Task D contain 1237, 377, 1835 and 3453 evaluated samples, respectively.

[1]The details are omitted due to confidentiality.

TABLE I Statistics of our microarchitecture design space

Module	# Linear	#Pow	# Categorical	# Combinations
IFU	8	4	0	$\sim 4 \times 10^8$
OoO	11	0	0	$\sim 5 \times 10^9$
IEX	12	0	3	$\sim 1 \times 10^9$
FSU	5	0	0	$\sim 2 \times 10^3$
LSU	6	3	0	$\sim 1 \times 10^7$
L2C	1	2	0	$\sim 8 \times 10^2$
Overall	43	9	3	$\sim 3 \times 10^{40}$

Evaluation Metrics: We adopt two commonly used evaluation metrics: Hypervolume (HV) as introduced in Equation (14) and average distance to reference set (ADRS) to offer effective evaluations of the quality of the approximated Pareto front generated. ADRS evaluates the closeness of a learned Pareto-optimal set $\mathcal{P}$ to the ground truth Pareto-optimal set $\mathcal{P}^*$:

$$ADRS(\mathcal{P}^*, \mathcal{P}) = \frac{1}{|\mathcal{P}^*|}\sum_{\boldsymbol{\alpha}\in\mathcal{P}^*}\min_{\boldsymbol{\beta}\in\mathcal{P}} l_2(\boldsymbol{\alpha}, \boldsymbol{\beta}), \tag{17}$$

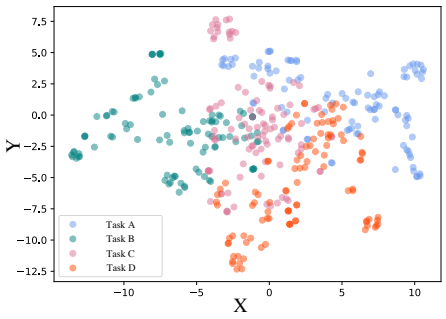
where l_2 is the l_2-norm.

Model Configurations: In the settings of IT-DSE, the feature tokenizer generates 128-dimension embedding, followed by 3 transformer layers. The number of attention heads is 8. The feed-forward multilayer perception (MLP) in each transformer layer has 512 dimensions. The learning rate for the Transformer layer and embedding layer is $1e-5$. The ensemble number is 3.

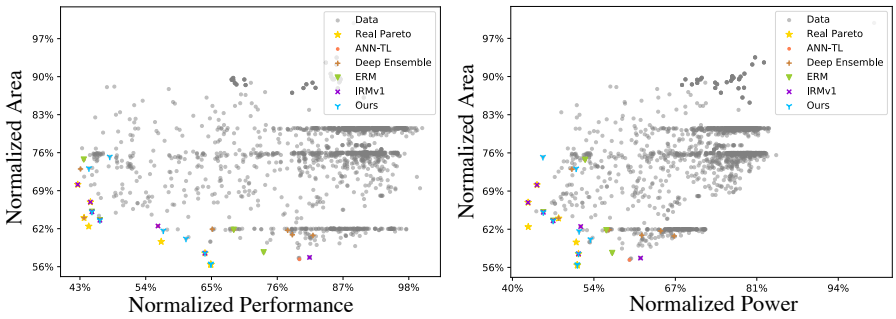
Considering the large design space, every method is initialized with 56 microarchitectures sampled randomly. Then, Bayesian optimization progressively suggests one sample per round, yielding 150 samples overall. All experiments are repeated 10 times along with the baseline, and the corresponding average results are reported.

B. Transferring performance within same design space

In the first experiment we evaluate the performance of design space exploration using transfer power from the same design space. For task A, B and C, we iteratively use two tasks as source task and left one as target task. For each source task we use 200 samples to conduct model pre-train and warm start the multi-objective Bayesian optimization to select Pareto-optimal sets. There exists few previous work focusing on utilizing transferring to conduct microarchitecture design space exploration. ANN-TL [8] builds ANN-based microarchitecture power model with data mixup strategy. Empirical risk minimization (ERM) [25] uses single multilayer perceptron (MLP) pre-trained with the merge of source tasks with NLL loss, For those methods, we implement them as our surrogate models and output not only predict PPA values but also the noise as aleatoric uncertainty. Deep ensemble (Deep Ens) [20] utilizes ensemble of multilayer perceptrons (MLPs) to give better uncertainty estimation, while IRMv1



(a) Data distribution for 4 tasks



(b) Pareto-optimal sets using source task data from same design space. Left: performance versus area; right: power versus area.

TABLE II Comparison of transfer performance in same design space

Methodologies	A, B → C		A, C → B		B, C → A	
	ADRS	HV	ADRS	HV	ADRS	HV
Ground Truth	0.0	0.0984	0.0	0.0684	0.0	0.0809
ANN-TL [8]	0.069	0.0891	0.045	0.0643	0.031	0.0749
Deep-Ens [20]	0.072	0.0840	0.066	0.0599	0.055	0.0727
ERM [25]	0.080	0.0877	0.064	0.0629	0.043	0.0742
IRMv1 [21]	0.025	0.0938	0.023	0.0667	0.027	0.0766
Ours	**0.021**	**0.0944**	**0.017**	**0.0679**	**0.020**	**0.0796**

TABLE III Comparison of transfer performance in different design spaces

Methodologies	A, D → C		A, C → D	
	ADRS	HV	ADRS	HV
Ground Truth	0.0	0.0984	0.0	0.7792
w/o. Pre-train	0.0533	0.0857	0.0853	0.7465
w/o. Ensemble	0.0275	0.0892	0.0722	0.7531
w/o. IRM	0.0293	0.0910	0.0701	0.7569
w/. IRMv1	0.0232	0.0914	0.0687	0.7602
Ours	**0.0217**	**0.0924**	**0.0641**	**0.7624**

[21] use MLP ensemble pre-trained with IRMv1 objective Equation (2) with source tasks.

As we can see from TABLE II, our method achieves the best transfer performance. For three cases, our method outperforms ANN-TL with 228.6%, 164.7% and 55.0% drop in ADRS, together with 5.61%, 5.30% and 5.90% improvement in Hypervolume, respectively. Compared with Deep ensemble, our method could reduce ADRS by 242.9%, 288.2% and 175.0% with 11.02%, 11.78% and 8.67% more Hypervolume in three cases, respectively. Compared with ERM, IT-DSE achieves 280.9%, 276.5% and 115.0% better ADRS with 7.10%, 7.36% and 6.78% better Hypervolume. Besides, our method is better than IRMv1 with 19.1%, 35.3% and 35.0% less ADRS and 0.64%, 1.77% and 3.77% more Hypervolume for three transfer tasks. The Pareto-optimal sets for the case $(A, B \rightarrow C)$ selected by baselines and our method can be seen in Fig. 4(b). These results demonstrate the superiority of our methods in finding better Pareto-optimal sets within limited time budget.

C. Transferring performance within different design space

In the second experiment, to investigate the transfer performance between different design spaces, we use task D with only 53-dimensional design space to be one of the source task $(A, D \rightarrow C)$ and the target task $(A, C \rightarrow D)$, respectively. To the best of our best knowledge, all previous methods cannot spread knowledge across different design spaces. Considering this, we conduct an ablation study on each part of our methods, all compared methods use FT-Transformer as the backbone. The detailed results are listed in TABLE III. For the 'w/. IRMv1' configuration, we substitute the Bayesian IRM principle with the original IRMv1 paradigm, using an FT-Transformer ensemble. Compared to this setup, the full IT-DSE shows a 6.9% and 7.2% reduction in ADRS and a 1.1% and 0.3% increment in Hypervolume for the two cases. In the 'w/o. IRM' configuration, which involves only pre-training the FT-Transformer ensemble on source tasks without IRM regularization, our method demonstrates 35.0% and 9.4% improvements in ADRS with 1.5% and 0.7% increase in Hypervolume for the listed cases. 'w/o. Ensemble', which uses a single FT-Transformer pre-trained with the Bayesian IRM objective, is outperformed by our complete method, which exhibits 26.7% and 12.6% reductions in ADRS with 3.6% and 1.2% increments in Hypervolume. Basically, 'w/o. Pre-train' just initializes the FT-Transformer ensemble without pre-training on source tasks. Our IT-DSE pre-trained on source tasks thus could gain significantly 145.6% and 33.1% less ADRS with 7.8% and 2.1% more Hypervolume.

V. Conclusion

We focus on the transfer multi-objective design space exploration problem for modern industry microarchitecture. In this paper, we propose IT-DSE, an automatic framework to improve transfer design space exploration performance. To the best of our knowledge, this is the first work that facilitates learning from prior design tasks across varied design spaces. Experimental results, drawn from complex, real-world microarchitecture test cases, demonstrate the practical superiority of our methods compared with other methods. We anticipate leveraging this transfer capacity to propel modern microarchitecture development.

Acknowledgment

We thank Wenlong Lv and Zhitang Chen for extremely helpful discussions and providing the implementation of [17].

REFERENCES

[1] Y.-I. Kim and C.-M. Kyung, "Automatic translation of behavioral testbench for fully accelerated simulation," in *IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004.* IEEE, 2004, pp. 218–221.

[2] B. C. Lee and D. M. Brooks, "Illustrative design space studies with microarchitectural regression models," in *2007 IEEE 13th International Symposium on High Performance Computer Architecture*. IEEE, 2007, pp. 340–351.

[3] J. Feldmann, K. Kraft, L. Steiner, N. Wehn, and M. Jung, "Fast and accurate dram simulation: Can we further accelerate it?" in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020, pp. 364–369.

[4] S. Beamer and D. Donofrio, "Efficiently exploiting low activity factors to accelerate rtl simulation," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.

[5] C. Bai, Q. Sun, J. Zhai, Y. Ma, B. Yu, and M. D. Wong, "Boomexplorer: Risc-v boom microarchitecture design space exploration framework," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2021, pp. 1–9.

[6] S. Salamin, M. Rapp, A. Pathania, A. Maity, J. Henkel, T. Mitra, and H. Amrouch, "Power-efficient heterogeneous many-core design with ncfet technology," *IEEE Transactions on Computers*, vol. 70, no. 9, pp. 1484–1497, 2020.

[7] B. Grayson, J. Rupley, G. Z. Zuraski, E. Quinnell, D. A. Jiménez, T. Nakra, P. Kitchin, R. Hensley, E. Brekelbaum, V. Sinha *et al.*, "Evolution of the samsung exynos cpu microarchitecture," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020, pp. 40–51.

[8] J. Zhai, Y. Cai, and B. Yu, "Microarchitecture power modeling via artificial neural network and transfer learning," in *2023 28th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2023, pp. 1–6.

[9] "Intel Skylake," https://en.wikichip.org/wiki/intel/microarchitectures/skylake_(server)#Overview_2.

[10] S. Naffziger, N. Beck, T. Burd, K. Lepak, G. H. Loh, M. Subramony, and S. White, "Pioneering Chiplet Technology and Design for the AMD EPYC™ and Ryzen™ Processor Families: Industrial Product," in *IEEE/ACM International Symposium on Computer Architecture (ISCA)*. IEEE, 2021, pp. 57–70.

[11] A. Seznec and P. Michaud, "A Case for (Partially) TAgged GEometric History Length Branch Prediction," *The Journal of Instruction-Level Parallelism*, vol. 8, p. 23, 2006.

[12] G. Chrysos and J. Emer, "Memory Dependence Prediction using Store Sets," in *IEEE/ACM International Symposium on Computer Architecture (ISCA)*. IEEE Computer Society, 1998, pp. 0142–0142.

[13] R. Bera, K. Kanellopoulos, S. Balachandran, D. Novo, A. Olgun, M. Sadrosadat, and O. Mutlu, "Hermes: Accelerating Long-Latency Load Requests via Perceptron-Based Off-Chip Load Prediction," in *IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2022, pp. 1–18.

[14] T. S. Karkhanis and J. E. Smith, "A First-order Superscalar Processor Model," in *IEEE/ACM International Symposium on Computer Architecture (ISCA)*. IEEE, 2004, pp. 338–349.

[15] S. Eyerman, L. Eeckhout, T. Karkhanis, and J. E. Smith, "A mechanistic performance model for superscalar out-of-order processors," *ACM Transactions on Computer Systems (TOCS)*, vol. 27, no. 2, pp. 1–37, 2009.

[16] Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko, "Revisiting deep learning models for tabular data," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18 932–18 943, 2021.

[17] W. Lyu, S. Hu, J. Chuai, and Z. Chen, "Efficient bayesian optimization with deep kernel learning and transformer pre-trained on multiple heterogeneous datasets," *arXiv preprint arXiv:2308.04660*, 2023.

[18] Y. Lin, H. Dong, H. Wang, and T. Zhang, "Bayesian invariant risk minimization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 16 021–16 030.

[19] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.

[20] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *Advances in neural information processing systems*, vol. 30, 2017.

[21] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz, "Invariant risk minimization," *arXiv preprint arXiv:1907.02893*, 2019.

[22] 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.

[23] G. E. Box and D. R. Cox, "An analysis of transformations," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 26, no. 2, pp. 211–243, 1964.

[24] I.-K. Yeo and R. A. Johnson, "A new family of power transformations to improve normality or symmetry," *Biometrika*, vol. 87, no. 4, pp. 954–959, 2000.

[25] V. Vapnik, "Principles of risk minimization for learning theory, advances in neural information processing nips 4 (pp. 831±838)," 1992.

[26] J.-Y. Audibert, "Progressive mixture rules are deviation suboptimal," *Advances in Neural Information Processing Systems*, vol. 20, 2007.

[27] G. Lecué, "Suboptimality of penalized empirical risk minimization in classification," in *Learning Theory: 20th Annual Conference on Learning Theory, COLT 2007, San Diego, CA, USA; June 13-15, 2007. Proceedings 20*. Springer, 2007.

[28] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[29] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 1126–1135.

[30] P. K. Diederik, M. Welling *et al.*, "Auto-encoding variational bayes," in *Proceedings of the International Conference on Learning Representations (ICLR)*, vol. 1, 2014.

[31] S. Daulton, M. Balandat, and E. Bakshy, "Differentiable expected hypervolume improvement for parallel multi-objective bayesian optimization," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9851–9864, 2020.

[32] J. Bachrach, H. Vo, B. Richards, Y. Lee, A. Waterman, R. Avižienis, J. Wawrzynek, and K. Asanović, "Chisel: Constructing Hardware in a Scala Embedded Language," in *ACM/IEEE Design Automation Conference (DAC)*, 2012, pp. 1216–1225.