On Advanced Methodologies for Microarchitecture Design Space Exploration

Tianji Liu*
CSE Department, CUHK
tjliu@cse.cuhk.edu.hk

Qijing Wang*
CSE Department, CUHK
qjwang21@cse.cuhk.edu.hk

Lixin Liu
CSE Department, CUHK
lxliu@cse.cuhk.edu.hk

Fangzhou Wang
CSE Department, CUHK
fzwang@cse.cuhk.edu.hk

Evangeline F.Y. Young
CSE Department, CUHK
fyyoung@cse.cuhk.edu.hk

ABSTRACT

With the ever-increasing complexity of microprocessors, microarchitectural design becomes over-challenging. Design space exploration (DSE) of microarchitecture configurations to obtain high-quality designs with different PPA trade-offs is time-consuming, due to the huge configuration space and inefficient VLSI verification flow. Many DSE frameworks proposed in previous works failed to systematically analyze the contribution of each algorithmic component to the full flow. This paper provides a novel methodology for designing DSE frameworks by separating DSE flow into stages, and discussing algorithmic instantiations in each stage with theoretical and experimental analyses. Newly formulated DSE frameworks guided by this methodology achieve state-of-the-art results in ICCAD'22 DSE contest evaluation environments.

ACM Reference Format:
Tianji Liu*, Qijing Wang*, Lixin Liu, Fangzhou Wang, and Evangeline F.Y. Young. 2024. On Advanced Methodologies for Microarchitecture Design Space Exploration. In *Great Lakes Symposium on VLSI 2024 (GLSVLSI '24), June 12–14, 2024, Clearwater, FL, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3649476.3658764

1 INTRODUCTION

As the demand for computing power continues to rise, modern microprocessors such as central processing units (CPUs) have become increasingly complex. In order to design and produce suitable and high-quality microprocessors for various requirements, exploring the trade-off of various targets (e.g., performance, power, area (PPA)) for different configuration combinations in microarchitectures, i.e., design space exploration (DSE), has become a key step in the whole process. Nowadays, due to the increasing number of configurations involved in microarchitectures, the relationships between components and performance have become very complex, making it extremely difficult to manually select an appropriate combination based on prior knowledge. A commonly adopted approach is to use a very large-scale integration (VLSI) verification flow implemented by software simulators to evaluate the PPAs of configuration combinations. However, as the flow involves many steps and is thus extremely time-consuming, it is unrealistic to obtain the knowledge of the target distribution by evaluating every configuration combination in a huge design space.

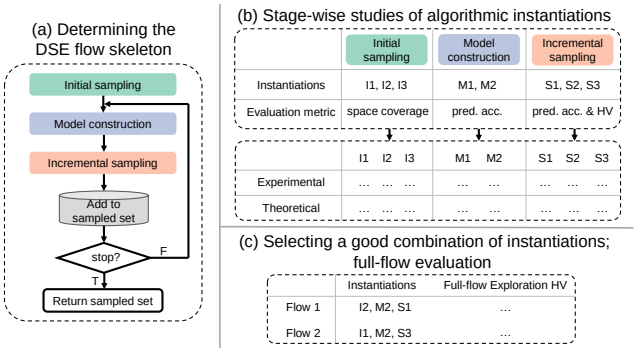


Figure 1: The methodology to construct a DSE framework.

A promising approach to resolve this problem is by modeling the relationship between the configurations and the targets using a small number of samples in the design space, and predicting the targets of the unsampled points using the model, so that the number of verifications needed can be greatly reduced. There has emerged a bunch of microarchitecture DSE works that are based on machine learning models thanks to their high accuracy, including the adoption of Random Forest [9] and active learning based AdaBoost [7], to name a few. More recently, the work [2] used a microarchitecture-aware active learning algorithm and a Gaussian Process model with deep kernel learning, obtaining promising results.

A DSE framework is usually composed of a few algorithmic components (e.g., modeling method, new point sampling strategy). Although a variety of DSE frameworks have been proposed in previous works, the contributions of each algorithmic component to the performance of the full DSE flow are rarely analyzed in a systematic way. While such works provide useful out-of-the-box DSE frameworks for microprocessor designs, they do not give much insight for future research on new DSE algorithms. To fill in this research gap, we develop a novel methodology for guiding how to obtain a high-performance DSE flow in this work, as illustrated in Figure 1. First, a skeleton of the DSE flow is determined in which the basic units are subroutines that achieve certain functionalities (Figure 1a). Each basic unit is called a *stage* of the DSE flow.

*Both authors contributed equally to this research.

GLSVLSI '24, June 12–14, 2024, Clearwater, FL, USA

ACM ISBN 979-8-4007-0605-9/24/06
https://doi.org/10.1145/3649476.3658764

Second, a list of possible algorithmic instantiations of each stage is comprehensively examined, by designing metrics, performing experimental and theoretical analyses dedicated to each stage (Figure 1b). Third, theoretically sound and practically effective DSE flows are obtained as combinations of some algorithmic instantiations based on the stage-wise studies, and finally evaluation of the overall DSE flow is performed (Figure 1c).

As a case study, we utilize this methodology to design new DSE frameworks on the Berkeley Out-of-Order Machine (BOOM) [4] microarchitecture. The outcomes of our case study are novel DSE flows that outperform previous works [2, 7, 9] and the contest top-3 teams in terms of Pareto optimality when tested in the ICCAD'22 DSE contest [8] evaluation environments.

2 BACKGROUNDS

Before conducting the BOOM case study, we first introduce some background concepts as well as the formulation and objective of the microarchitecture design space exploration problem.

Terminologies. The *design space* $\mathcal{X}$ is the set of all feasible microarchitecture configuration vectors[1]. The *target/output space* $\mathcal{Y}$ is the set of target values (i.e., PPAs) of all the configurations in $\mathcal{X}$. A *model* refers to a quantitative description of the relation between the design space and the target space. *Sampling* refers to selecting one configuration vector to obtain the corresponding PPA values through the VLSI verification flow.

Module space. Some algorithms require that the m-dimensional input space is a hyper-rectangle, or in the discrete settings, a *grid-like* space in which there is a one-to-one correspondence between any m-tuple of integers $(i_1, \ldots, i_m)$ and a feasible point, where $1 \le i_j \le l_j$ and l_j is the number of feasible values for the j-th input variable. In our case, the BOOM microarchitecture design space $\mathcal{X}$ is not grid-like. However, if we consider the sub-vector of a particular architectural module (e.g., LSU) containing all the variables of this module, and represent each feasible value of this sub-vector using a unique indexing integer, every configuration vector in $\mathcal{X}$ is transformed into a new vector consisting of c indexing integers where c is the number of modules. We call the space containing all such vectors as the *module space* $\mathcal{C}$. As the modules are formulated without inter-dependency (Section 4.1), this module space is grid-like.

Problem Formulation. Since the area, power and performance metrics for different microarchitecture configurations are correlated and need to be traded-off, we aim to find the configurations that correspond to the Pareto optimal PPAs $\mathcal{P}(\mathcal{Y}) = \{\boldsymbol{y} \in \mathcal{Y} : \forall \boldsymbol{y}' \in \mathcal{Y} \setminus \{\boldsymbol{y}\}, \boldsymbol{y}' \not\succeq \boldsymbol{y}\}$, where "$\boldsymbol{y}' \succeq \boldsymbol{y}$" stands for "$\boldsymbol{y}'$ dominates $\boldsymbol{y}$", i.e., $y'_i \ge y_i$ for all target variables $i = 1, \ldots, t$ ($t = 3$ in our case). However, in practice, the Pareto optimal points may not be discovered due to limited resources, and we thus leverage the Pareto hypervolume [13] as a metric indicating the "degree of Pareto optimality" of the selected points quantitatively. For a set of obtained PPAs $\mathcal{D}_y$, the Pareto hypervolume of $\mathcal{P}(\mathcal{D}_y)$ with respect to a reference point $\boldsymbol{v}_{\text{ref}} \in \mathbb{R}^t$ is defined as

$$\text{HV}_{\boldsymbol{v}_{\text{ref}}}(\mathcal{P}(\mathcal{D}_y)) = \int_{\mathbb{R}^t} \Big[\boldsymbol{y} \succeq \boldsymbol{v}_{\text{ref}}\Big] \Big[\bigvee_{\boldsymbol{y}' \in \mathcal{P}(\mathcal{D}_y)} \boldsymbol{y}' \succeq \boldsymbol{y}\Big] \mathrm{d}\boldsymbol{y}, \quad (1)$$

where $[\cdot]$ is the Iverson bracket which takes value 1 if the inner statement is true and 0 otherwise. The reference point $\boldsymbol{v}_{\text{ref}}$ can be chosen as an arbitrary point that is dominated by every point in $\mathcal{P}(\mathcal{D}_y)$[2]. Then, our DSE problem can be formalized as follows.

Problem 1 (Microarchitecture Design Space Exploration, DSE)**.** For a design space $\mathcal{X}$, select a set of microarchitecture configurations $\mathcal{D}_x = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subseteq \mathcal{X}$ and obtain the corresponding PPA values $\mathcal{D}_y = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n\} \subseteq \mathcal{Y}$ via the VLSI verification flow under the constraint of limited resources (e.g., the number of configurations n or the total time of VLSI flows is no greater than a given value), such that the Pareto hypervolume $\text{HV}(\mathcal{P}(\mathcal{D}_y))$ is maximized.

3 METHODOLOGY

This section performs the first and second step of our methodology.

3.1 DSE Flow Skeleton

In this work, we examine the algorithm of interests using a skeleton shown in Figure 1a where the algorithms being studied serve as instantiations of different stages. This framework begins with an initial sampling stage to obtain some sample points to construct the first model. It then performs iterative exploration by repeating two stages. First, the model is updated using the newly sampled points (model construction). Second, one new point is selected with its PPA obtained for improving the model in the next iteration (incremental sampling). In this way, the model will be maintained up-to-date and accurate after sampling each new point.

3.2 Initial Sampling

The target of the initial sampling stage is to obtain a comprehensive understanding about the distribution of PPAs by sampling a small batch of configurations $\mathcal{D}_x$ from the design space $\mathcal{X}$. Under the assumption that the Pareto optimal configurations are uniformly distributed in $\mathcal{X}$ *a priori*, the objective of this stage can be considered as: select n configurations, such that they are diverse and can cover the whole design space as much as possible. This problem has been extensively studied in the statistics literature as *experimental design* [12]. In this work, we examine three representative algorithms, and to our knowledge two of them have never been studied in the microarchitecture DSE literature.

Maximin and minimax design. The objectives of maximin and minimax design [11] are formulated respectively as

$$\max_{\mathcal{D}_x} \min_{\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{D}_x, i \ne j} ||\boldsymbol{x}_i - \boldsymbol{x}_j||, \quad (2)$$

$$\min_{\mathcal{D}_x} \max_{\boldsymbol{x} \in \mathcal{X}} \min_{\boldsymbol{x}_i \in \mathcal{D}_x} ||\boldsymbol{x} - \boldsymbol{x}_i||. \quad (3)$$

The maximin criterion maximizes the nearest neighbour distance of $\mathcal{D}_x$. The minimax criterion minimizes the Hausdorff distance, a metric indicating the dissimilarity of two sets, $\mathcal{D}_x$ and $\mathcal{X}$. We use an exchange-based algorithm for finding $\mathcal{D}_x$ that optimizes the maximin or minimax criteria, elaborated in Algorithm 1.

[1] We simply call a configuration vector a *configuration* when there is no ambiguity.

[2] We omit $\boldsymbol{v}_{\text{ref}}$ in the notation of Pareto hypervolume for brevity.

Algorithm 1 Maximin and minimax design

1: Initialize $\mathcal{D}_x$ by iteratively selecting a new point furthest to the currently selected point set
2: **while** the criterion keeps improving **do**
3: **for each** $\boldsymbol{x}_i \in \mathcal{D}_x$ in random order **do**
4: Find $\boldsymbol{x}^* \in \mathcal{X}$, s.t. replacing $\boldsymbol{x}_i$ by $\boldsymbol{x}^*$ in $\mathcal{D}_x$ optimizes the criterion
5: Perform such replacement

Latin Hypercube design (LHD). Under the setting of a grid-like space where each input variable has k feasible values, the method of LHD is to sample k points, such that for each input variable i and feasible value j, there is exactly one sample $\boldsymbol{x}$ such that $x^{(i)} = j$. For example, if the number of dimensions $m = 2$, there should be exactly one sample in every row and column of the 2D grid. In our problem, LHD is applied to the module space (Section 2). Since the number of feasible values for each variable $k^{(i)}$ may not be the same, we perform LHD with $k = \max_i k^{(i)}$, and select the nearest feasible neighbour of each sample. LHD has the following problems: (a) there is a huge number of feasible LHD results for a given space; (b) the number of LHD samples k cannot be adjusted as it is an attribute of the input space. To address these issues, we combine LHD with the minimax criterion that enables generating a good subset of LHD samples with an arbitrary size k' ($k' \leq k$). Specifically, a batch of different LHD sample sets (say, $100 \times k$ samples) are first generated, and all possible subsets of size k' for each sample set are considered. The best subset in terms of Equation (3) is selected among all such subsets. We denote this method as *minimax-resampled LHD*.

Transductive experimental design (TED). TED [14] assumes that the relation between the input and output space can be modeled by a kernel regression on $\mathcal{D}_x$, and it finds a $\mathcal{D}_x$ that minimizes the variance of the prediction error on $\mathcal{X}$. Hence, unlike the methods introduced above whose optimization processes are only relevant to the design space, the optimization objective of TED is in fact in the target space[3], which correlates closer with the objective of our DSE problem. Thus, the performance of TED can be potentially better than the methods introduced above, *if the assumed model describes the true distribution well* (i.e., we have good prior knowledge); otherwise, the performance of TED can be much worse, as shown by the experiments in Section 3.3. To our knowledge, this issue has been neglected in many previous design space exploration works on EDA that utilize TED. For the implementation of TED, we use the sequential design algorithm in [14].

Comparison. In order to quantitatively measure the performance of these algorithms, we designed a metric indicating the coverage of the sampled points in the design space. Specifically, the covered volume of the sample set $\mathcal{D}_x$ is estimated by placing m-dimensional balls B_m centered at each sampled configuration:

$$V(\mathcal{D}_x) = \mathrm{Vol}\left[\left(\bigcup_{\boldsymbol{x}_i \in \mathcal{D}_x} B_m(\boldsymbol{x}_i, r)\right) \bigcap \mathrm{box}(\mathcal{X})\right], \tag{4}$$

where $\mathrm{box}(\mathcal{X})$ denotes the tight bounding hyper-rectangle of $\mathcal{X}$, and the radius r is set as the radius of an m-dimensional ball with volume $\mathrm{Vol}(\mathrm{box}(\mathcal{X}))/|\mathcal{D}_x|$. Monte Carlo methods are adopted for computing $V(\mathcal{D}_x)$. We conduct an experiment using the design space of the example dataset provided in the ICCAD'22 DSE contest codebase (denoted as the "contest example dataset" in the following), and the details about the design space are introduced in Section 4.1. Results are shown in Figure 2. In particular, TED performs the worst among all methods (even worse than random sampling for some cases) in terms of this coverage metric. This is understandable since the coverage is with regard to the design space, whereas the objective of TED is formulated with regard to the target space. For the other initialization methods, minimax-resampled LHD and minimax obtain slightly higher coverage than maximin.

[3]Despite of this, the PPA data are not required for TED; see [14].

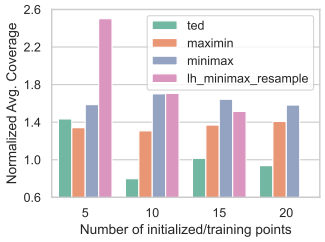


Figure 2: Normalized coverage of the initialization samples generated by different methods, averaged over 50 trials. Random sampling corresponds to 1.0 coverage.

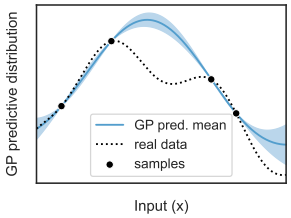


Figure 3: Gaussian Process regression illustration. The shaded area represents 95% confidence interval of the GP predictive distribution.

3.3 Model Construction

The objective of this stage is to construct a model $\mathcal{M}$ describing the relationship between the design space $\mathcal{X}$ and the target space $\mathcal{Y}$ using the currently obtained configurations and PPAs $\mathcal{D} = (\mathcal{D}_x, \mathcal{D}_y)$, in order to guide the selection of the next configuration. Apparently, an accurate modeling can provide valuable information and potential for the DSE algorithm to obtain high Pareto hypervolume. The models of interests include:

Random forest (RF). RF [3] is a bagging ensemble method that combines multiple regression (decision) trees as base models. Each regression tree is fit with randomness (1) in selection of training samples and (2) by ignoring a subset of input variables. The final prediction value is the averaged prediction of all the base models. RF has low prediction variance because of its bagging nature.

XGBoost (XGB). XGB [5] is a gradient boosting method that also utilizes multiple regression trees. Different from RF, the regression trees in XGB are constructed and combined one by one, according to the prediction error (formulated as the gradient of loss w.r.t. prediction) of the current model on the target value ($\boldsymbol{y}$). In other words, the new regression tree refines the current model. XGB has high generalization ability and is widely applied in practice.

Gaussian Process regression (GPR). In contrast to the other models introduced above, GPR is intrinsically able to quantify the uncertainty of its predictions. Informally, GPR prediction can be regarded as interpolation and extrapolation using the training data, and it is more "confident" about the predictions at the input locations where the training data are more densely sampled, as shown in Figure 3. In GPR, the model function values $\boldsymbol{f}$ at a set of input points $\mathbf{X} = \{\boldsymbol{x}_i\}_{i=1}^n$ are treated as random variables, where $\boldsymbol{f} = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_n)]^T$. According to the definition of Gaussian

Process (GP), the joint distribution of these random variables (GP prior) follows a multivariate Gaussian distribution, specified as

$$f|\mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathbf{K_{XX}}), \tag{5}$$

where $(\mathbf{K_{XX}})_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $k(\cdot,\cdot)$ is the covariance/kernel function of the GP. Suppose we have a dataset $(\mathbf{X}, \boldsymbol{y})$ where the target observations $y_i = f(\boldsymbol{x}_i) + \varepsilon$, and ε is a zero-mean Gaussian noise with variance σ_n^2. The predictive distribution of $f_* = f(\boldsymbol{x}_*)$ at a new point $\boldsymbol{x}_*$ can then be derived from the joint prior of $[\boldsymbol{y}, f_*]^T$:

$$\begin{bmatrix} \boldsymbol{y} \\ f_* \end{bmatrix} \Bigg| \mathbf{X}, \boldsymbol{x}_* \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K_{XX}} + \sigma_n^2\mathbf{I} & \mathbf{K_{X}\boldsymbol{x}_*} \\ \mathbf{K_{\boldsymbol{x}_*X}} & k(\boldsymbol{x}_*, \boldsymbol{x}_*) \end{bmatrix}\right), \tag{6}$$

$$f_*|\mathbf{X}, \boldsymbol{x}_*, \boldsymbol{y} \sim \mathcal{N}(\mathbf{K_{\boldsymbol{x}_*X}}[\mathbf{K_{XX}} + \sigma_n^2\mathbf{I}]^{-1}\boldsymbol{y}, \tag{7}$$
$$k(\boldsymbol{x}_*, \boldsymbol{x}_*) - \mathbf{K_{\boldsymbol{x}_*X}}[\mathbf{K_{XX}} + \sigma_n^2\mathbf{I}]^{-1}\mathbf{K_{X\boldsymbol{x}_*}}).$$

The mean and variance of the Gaussian in Equation (7) are the GPR prediction and the quantified prediction uncertainty respectively.

There are many possible instantiations for the kernel function k. In this work, we focus on the radial basis function (RBF) $k(\boldsymbol{x}_a, \boldsymbol{x}_b) = \exp(-||\boldsymbol{x}_a - \boldsymbol{x}_b||_2^2/2\sigma^2)$, where σ is a hyper-parameter controlling the length scale. The advantage of the RBF kernel is that it is theoretically a universal approximator [10] and thus has very strong expressive power. Besides, RBF implies that the target values of input points that are close to each other are highly correlated, which is reasonable in architectural design since we expect the PPAs to be smoothly distributed over the design space without abrupt changes.

Other machine learning models. We also looked into models including linear regression, Support Vector Regression, AdaBoost and Multi-layer Perceptron, but none of them performs well in terms of modeling accuracy and maximizing the Pareto hypervolume, so we do not include them in this work for further investigation.

Comparison. We perform a comprehensive experiment regarding the prediction errors on the full design space of different models when fit on different numbers of points generated by different initialization methods, using the contest example dataset. The results are shown in Figure 4, where the reported error is the averaged value over 50 trials with different random seeds. We adopt the RBF kernel in TED following the previous work [9]. When comparing different models, the performance of RF and GPR are close, and in general GPR is better when the number of training points is small ($\leq$ 10), thanks to its Bayesian nature.

When comparing different initialization methods in Figure 4, the results of TED with $\sigma = 10$ are the best, which indicates the effectiveness of its prediction error formulation in the y-space. To ensure TED truly works well, the underlying model assumption of TED should be consistent with the real data distribution, which is not always the case. To demonstrate this, the value of σ in the RBF kernel of TED is adjusted and we can see the model's prediction error can increase dramatically, as shown in Figure 4f.

3.4 Incremental Sampling

The target of this stage is to sample the next configuration $\boldsymbol{x}_*$ and obtain its PPA following the guidance of the constructed model $\mathcal{M}$. It is intuitive to greedily select a configuration that maximizes the objective of the DSE problem (i.e., Pareto hypervolume) under the current modeling of the PPA distribution. Alternatively, we

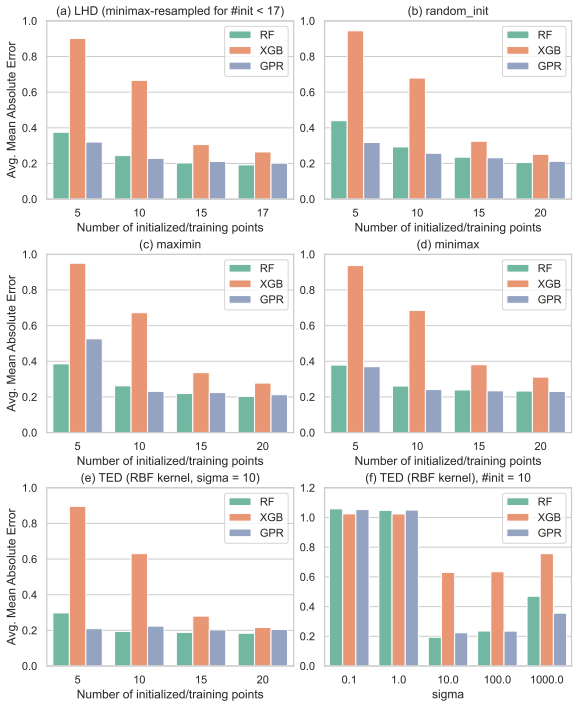


Figure 4: Prediction errors of models with different configs.

may select a configuration such that our knowledge about the PPA distribution can be enriched by choosing this configuration, which enables better modeling and creates chances that lead to better samples in the future. These two high-level ideas are referred to as *exploitation* and *exploration* respectively, and the trade-off between them has been studied for a long time, especially in the reinforcement learning literature. Intuitively, an effective concrete sampling strategy needs to balance exploration and exploitation, and we analyze a few strategies in the aspect of these two concepts.

Pure exploitation. The configuration $\boldsymbol{x}_*$ is sampled such that its predicted PPA increases the current Pareto hypervolume most, i.e., maximizing $\text{HV}(\mathcal{P}(\mathcal{D}_y \cup \{\hat{\boldsymbol{y}}\}))$, where $\hat{\boldsymbol{y}} = \mathcal{M}(\boldsymbol{x})$ is the predicted PPA at an unsampled configuration $\boldsymbol{x}$. (`xplt`)

Pure exploration. A configuration $\boldsymbol{x}_*$ with the largest uncertainty is sampled. The uncertainty can be measured in various forms, and here we focus on two representative instantiations: design space uncertainty (`xplr_cos`) and target space uncertainty (`xplr_var`). The design space uncertainty of a point $\boldsymbol{x}$ is the sparseness of the sampled configurations $\mathcal{D}_x$ in the neighbourhood of $\boldsymbol{x}$. We quantify this value as the sum of cosine dissimilarities between $\boldsymbol{x}$ and all sampled configurations:

$$U(\boldsymbol{x}, \mathcal{D}_x) = \sum_{\boldsymbol{x}' \in \mathcal{D}_x} \left(1 - \frac{\boldsymbol{x} \cdot \boldsymbol{x}'}{||\boldsymbol{x}||_2 ||\boldsymbol{x}'||_2}\right) \Big/ 2. \tag{8}$$

The target space uncertainty is related to the modeling. Some models have intrinsic prediction uncertainties, such as the variance of the GPR predictive distribution (Equation 7). For models that are unable to provide explicit prediction uncertainty (e.g., RF, XGB), we adopt the disagreement-based method [7] that for multiple models

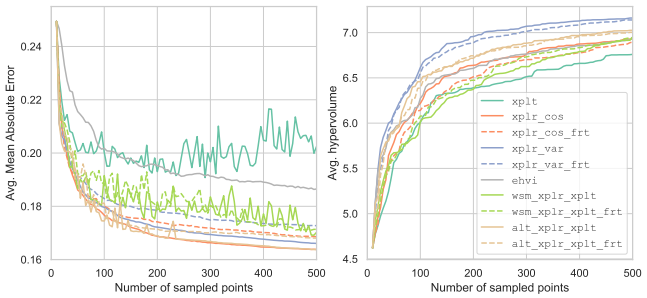


Figure 5: Model prediction errors (lower is preferred) and Pareto hypervolumes (higher is preferred) along DSE flows with different sampling strategies. Minimax initialization of 10 points and GPR model is used. For `wsm_xplr_xplt` and `alt_xplr_xplt`, the exploration metric is prediction variance.

of the same type but with different random seeds being fit, the variance in their predicted values at $\boldsymbol{x}$ is treated as the uncertainty.

Hybrid strategies. Theoretically, exploration and exploitation should be balanced in order to achieve a better final result with higher controllability, since pure exploration may fail to converge and pure exploitation may fall into local optima. Here we consider three approaches that combine our aforementioned pure strategies.

The first approach (`wsm_xplr_xplt`) considers both the exploitation metric (Pareto hypervolume) and the exploration metric (cosine dissimilarity or prediction variance), and then the configuration with the maximum weighted sum of the two is selected. The second approach (`alt_xplr_xplt`) simply alternates exploration and exploitation at a frequency of every k iterations where we put $k = 5$ in our experiments. In the third approach (suffixed by `_frt`), we predict PPAs for all unsampled configurations and select a point among those configurations whose PPAs are in-frontier (exploitation), i.e., belong to $\mathcal{P}(\{\hat{\boldsymbol{y}} : \hat{\boldsymbol{y}} = \mathcal{M}(\boldsymbol{x}), \boldsymbol{x} \in \mathcal{X} \setminus \mathcal{D}_x\})$. The selection strategy of an in-frontier configuration includes pure exploration, and the above two hybrid strategies.

Expected Hypervolume Improvement (EHVI). EHVI is an approach for jointly considering exploration and exploitation in which a sample with the maximum expected Pareto hypervolume increment is selected. The EHVI of sampling $\boldsymbol{x}$ is specified as

$$\begin{aligned} \mathrm{EHVI}(\boldsymbol{y}|\mathcal{D}) &= \mathbb{E}_{\boldsymbol{y}|\mathcal{D}}[\mathrm{HV}(\mathcal{P}(\mathcal{D}_y \cup \{\boldsymbol{y}\})) - \mathrm{HV}(\mathcal{P}(\mathcal{D}_y))] \\ &= \int_{\mathbb{R}^t} \mathrm{HV}(\mathcal{P}(\mathcal{D}_y \cup \{\boldsymbol{y}\}))p(\boldsymbol{y}|\mathcal{D})\mathrm{d}\boldsymbol{y} - \mathrm{HV}(\mathcal{P}(\mathcal{D}_y)), \end{aligned} \tag{9}$$

where $p(\boldsymbol{y}|\mathcal{D})$ is the pdf of the GP predictive distribution at $\boldsymbol{x}$ (Equation 7). Note that $\boldsymbol{y}$ in (9) is a random variable modeled by GP prediction, but the predicted value $\hat{\boldsymbol{y}}$ is not, which makes EHVI different from the pure exploitation strategy though they appear to be similar. We use Monte Carlo methods for computing EHVI [6].

Comparison. Figure 5 shows the experimental results comparing different sampling strategies, where ten DSE flows are executed with different random seeds for each strategy and their averaged results are shown in one curve. From the modeling error results, we have the following observations. The first one is intuitive: the broader extent of exploration, the better modeling accuracy, which can be seen from the fact that the pure exploration strategies perform the best, and the in-frontier strategies (with `_frt` suffixes) have larger errors than their non-in-frontier counterparts. Second, there are fluctuations in modeling accuracy for strategies with exploitation. This is because the selected point with maximum predicted hypervolume may be located very close to a previous sample in the design space, which leads to dense clusters of samples and the overfitting of the GPR model that in turn causes high prediction errors in the sparsely sampled regions.

For the Pareto hypervolume results, it is interesting that the pure exploration using the prediction uncertainty (`xplr_var`) performs the best, rather than a hybrid strategy with well-balanced exploration and exploitation. The reason could be that the number of sampled points is too few compared with the number of feasible configurations, such that the design space is not covered enough and many closer-to-optimal points are unrevealed. In this situation, it would be better to keep on discovering. Notably, in Figure 5 the hypervolume of the weighted-sum hybrid strategy (`wsm_xplr_xplt`) continues to increase when there are 400-500 samples, but meanwhile the other strategies are nearly saturated. Hence, exploitation is likely to become important when we have more points sampled, but this is not investigated in our experiments due to high computational complexity.

In particular, although EHVI is supported by a solid theoretical background, it fails to outperform other heuristic-based strategies in terms of both modeling accuracy and obtaining large hypervolume. Besides, the heuristic-based strategies are more flexible than EHVI in terms of exploration and exploitation trade-off.

4 FULL-FLOW EVALUATIONS

In this section, we perform the third step of our methodology.

4.1 Design Space of the Contest Environments

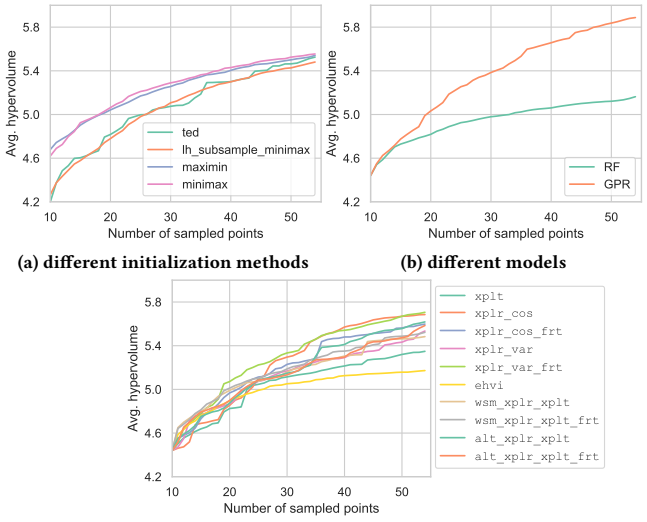
The design space for the BOOM [4] microarchitecture in the ICCAD'22 DSE contest [8] evaluation environments (including the example dataset and an online platform [1]) is shown in Table 1. There are 9 modules, each of which consists of one or more components, and each component has a list of candidate values. There are constraints over these values among different components inside a module, i.e., some of the combinations are feasible and some are not, which are not shown here. The example dataset and the online evaluation platform data consist of a subset of the feasible values for each component in Table 1 respectively. The number of points in the example dataset design space is approximately 16k. Note that the design space and PPAs of the online data are hidden from us.

4.2 Comparison of Stage-wise Algorithms

We test different combinations of stage-wise algorithmic components on the contest example dataset. Since the VLSI verification flows for processors can be very time-consuming, we set a relatively small budget for the total number of samples to be explored as 54, in which 10 samples are retrieved during initialization and the rest 44 points are obtained along the repetitive model construction and incremental sampling steps. All possible choices for the initialization methods, models and incremental sampling strategies

Table 1: Microarchitecture Design Space

Module	Component	Candidate Values
Fetch	FetchWidth	4, 8
Decoder	DecodeWidth	1, 2, 3, 4, 5
ISU	MEM_INST.DispatchWidth	1, 2, 3, 4, 5
	MEM_INST.IssueWidth	1, 2
	MEM_INST.NumEntries	6, 8, 10, 12, 14, 16, 20, 22, 24, 26, 28
	INT_INST.DispatchWidth	1, 2, 3, 4, 5
	INT_INST.IssueWidth	1, 2, 3, 4, 5
	INT_INST.NumEntries	6, 8, 12, 20, 24, 32, 36, 40, 44, 48
	FP_INST.DispatchWidth	1, 2, 3, 4, 5
	FP_INST.IssueWidth	1, 2, 3, 4, 5
	FP_INST.NumEntries	6, 8, 12, 16, 20, 24, 28, 32, 36, 40
IFU	BranchTag	6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26
	FetchBufferEntries	6, 8, 12, 14, 16, 20, 21, 24, 30, 32, 35, 36, 40, 45
	FetchTargetQueue	14, 16, 20, 30, 32, 36, 40, 44, 48, 50
ROB	ROBEntries	30, 32, 34, 36, 60, 64, 72, 90, 96, 108, 120, 128, 130, 132, 136, 140
PRF	INTPhysRegisters	42, 52, 62, 70, 80, 90, 100, 110, 118, 128, 138, 146
	FPPhysRegisters	38, 48, 54, 58, 64, 74, 86, 96, 106, 118, 128, 138, 146
LSU	LDQEntries	6, 8, 12, 16, 20, 24, 28, 32, 36
	STQEntries	6, 8, 12, 16, 20, 24, 28, 32, 36
I-Cache/MMU	Sets	32, 64
	Ways	1, 4, 8
	I-TLBSets	1, 2
	I-TLBWays	16, 32
D-Cache/MMU	Sets	64
	Ways	2, 4, 8
	ReplacementPolicy	0, 1
	MSHR	2, 4, 8
	D-TLBSets	1, 2
	D-TLBWays	8, 32



(a) different initialization methods **(b) different models**

(c) different incremental sampling strategy

Figure 6: Averaged Pareto hypervolumes along DSE flows.

investigated in the stage-wise experiments are included here, except the XGB model due to its high prediction error shown in the Section 3.3 experiments. The results are shown in Figure 6, where each curve shows the averaged Pareto hypervolumes of all the experiments with the corresponding algorithm instantiation. For example, each curve in Figure 6a is the averaged result over 2 models × 10 sampling strategies × 10 random seeds = 200 experiments. We can observe that on average, minimax, GPR and `xplr_var_frt` achieve advanced performance in their corresponding tasks, which are consistent with our stage-wise studies.

Table 2: Results of our DSE flows compared with other works.

DSE flow	Ex. Dataset HV	Online Platform HV	Online Platform Norm. time†
DAC'13 [9]	5.411	-	-
DAC'16 [7]	6.098	-	-
ICCAD'21 [2]	6.136	-	-
Contest 1st Place	-	7.900	0.93
Contest 2nd Place	-	7.782	1.01
Contest 3rd Place	-	7.820	1.30
minimax + GPR + `xplr_var_frt`	6.110	**7.916**	1.00
minimax + GPR + `xplr_var`	6.158	7.748	1.13
TED + GPR + `xplr_var_frt`	**6.513**	7.898	1.02

† Including the time of VLSI verification flows and DSE algorithm.

4.3 Final Flow Formulation

We now select high-performace DSE flows based on our previous experiments and stage-wise theoretical analyses, which serve as the product of our comprehensive study. From the results in Section 4.2, we obtain one candidate flow of minimax + GPR + `xplr_var_frt`. Besides, pure exploration (`xplr_var`) shows superiority in the incremental sampling experiments (with minimax + GPR), while TED is very effective when having a proper σ as analyzed in Section 3.3, so we also consider the following two DSE flows: minimax + GPR + `xplr_var`, and TED + GPR + `xplr_var_frt`.

4.4 Comparison with Other Works

We evaluate our solutions on the ICCAD'22 DSE contest example dataset and the contest online evaluation platform [1], compared with three previous published works, as well as the results of the contest top-3 teams retrieved from the online platform ranking. DAC'16 [7] utilized orthogonal array for initialization, but it is computationally intractable on the BOOM design space, so we alternatively use LHD for DAC'16 since orthogonal array is known to be an extension of LHD [12]. For all the experiments in both evaluation environments, we use the same number of total and initialization samples as in Section 4.2. The results are shown in Table 2. All of our three flows outperform DAC'13 and DAC'16, among which two flows surpass the state-of-the-art DSE framework ICCAD'21 [2] on the contest example dataset. For the online platform evaluation, two of our flows obtain higher or similar Pareto hypervolume compared with the contest first place. The number of samples used by the contestants on the platform is unknown to us, so their runtimes are provided in Table 2 as rough references. Since the algorithm runtime is negligible compared to the VLSI flow time, the number of samples used in the first and second place flows should be similar to ours. To conclude, the promising results indicate the effectiveness of our obtained DSE flows and the DSE design methodology.

5 CONCLUSION

In this paper, we proposed a novel methodology for guiding the design of DSE frameworks. This design methodology starts with obtaining a DSE flow skeleton that consists of different stages, followed by theoretical and experimental analyses of possible algorithmic instantiations for each stage systematically. We conducted a case study on the BOOM microarchitecture following the methodology and obtained novel DSE flows based on the comprehensive stage-wise studies, which achieved state-of-the-art performances in

terms of maximizing the Pareto hypervolume in the ICCAD'22 DSE contest evaluation environments. Our methodology can be applied to general DSE problems including microarchitecture design.

REFERENCES

[1] 2022. DSE Contest Evaluation Platform (as of Sep 05, 2022). http://47.93.191.38/

[2] Chen Bai et al. 2021. BOOM-Explorer: RISC-V BOOM Microarchitecture Design Space Exploration Framework. In *Proc. ICCAD*.

[3] Leo Breiman. 2001. Random forests. *Machine learning* (2001).

[4] Christopher Celio et al. 2015. The berkeley out-of-order machine (boom): An industry-competitive, synthesizable, parameterized risc-v processor. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2015-167* (2015).

[5] Tianqi Chen et al. 2016. XGBoost: a scalable tree boosting system. In *Proc. KDD*.

[6] Samuel Daulton et al. 2020. Differentiable Expected Hypervolume Improvement for Parallel Multi-Objective Bayesian Optimization. In *Proc. NeurIPS*.

[7] Dandan Li et al. 2016. Efficient design space exploration via statistical sampling and AdaBoost learning. In *Proc. DAC*.

[8] Sicheng Li et al. 2022. 2022 ICCAD CAD Contest Problem C: Microarchitecture Design Space Exploration. In *Proc. ICCAD*.

[9] Hung-Yi Liu et al. 2013. On learning-based methods for design-space exploration with High-Level Synthesis. In *Proc. DAC*.

[10] Charles A Micchelli, Yuesheng Xu, and Haizhang Zhang. 2006. Universal Kernels. *Journal of Machine Learning Research* (2006).

[11] Luc Pronzato. 2017. Minimax and maximin space-filling designs: some properties and methods for construction. *Journal de la Société Française de Statistique* 158, 1 (2017), 7–36.

[12] Thomas J Santner, Brian J Williams, and William I Notz. 2003. *The design and analysis of computer experiments*. Springer.

[13] Amar Shah et al. 2016. Pareto frontier learning with expensive correlated objectives. In *Proc. ICML*.

[14] Kai Yu et al. 2006. Active learning via transductive experimental design. In *Proc. ICML*.