

EDA Flow Matters: Stage-Aware Parameter Optimization of Tool Chain

Xinheng Li^{1,6*}, Donger Luo^{1*}, Peng Xu², Ziyang Yu², Qi Sun³, Tinghuan Chen⁴, Bei Yu², Hao Geng^{1,5,6†}

¹ShanghaiTech University ²Chinese University of Hong Kong

³Zhejiang University ⁴Chinese University of Hong Kong, Shenzhen

⁵Shanghai Engineering Research Center of Energy Efficient and Custom AI IC ⁶ZeroShot Co., Ltd.

Abstract—Optimizing Electronic Design Automation (EDA) tool parameters with only dozens of affordable evaluations represents one of the most challenging problems in today’s EDA flow management, where each experiment costs hours to days yet directly impacts final PPA outcomes. While Bayesian Optimization (BO) naturally fits such sample-constrained scenarios, it models the entire EDA flow as a monolithic formulation, blindly ignoring the sequential structure that each stage in the EDA flow affects the next. In this work, we propose a stage-aware optimization framework that fundamentally rethinks EDA parameter tuning. The proposed stage-aware Gaussian process explicitly models cascading relationships between EDA stages through interconnected GP layers, extracting abundant information from each expensive evaluation. To better meet realistic needs, we further introduce Expected Hypervolume Improvement (EHVI)-Efficiency, a time-aware acquisition function that exploits evaluation runtime estimation and EDA tools’ checkpoint reuse to balance design metrics’ expected improvement against EDA flow’s computational cost. Experiments and ablation studies on 6 designs across 3 process nodes demonstrate the effectiveness of our proposed method.

I. INTRODUCTION

Imagine spending weeks tuning hundreds of parameters in the EDA flow, only to realize that a minor adjustment made in an early stage could have produced a better outcome. This scenario represents one of the most critical challenges in today’s EDA flow management: engineers face millions of possible parameter combinations but are constrained to only a few dozen trials because of the extremely long runtime of the EDA flow, making it nearly impossible to reach optimal solutions by relying on manual expertise or inherited ‘golden’ configurations. To overcome these limitations and move beyond inefficient manual approaches, Artificial Intelligence (AI) algorithms are increasingly emerging as promising tools for EDA parameter tuning.

Most existing modeling approaches treat the EDA flow as a single monolithic formulation, relying heavily on AI models to extract relationships between scarce data relating tool parameters and output metrics. With models like XGBoost [1] and Gaussian Process (GP) regression [2] used as surrogate models, they build iterative optimization to explore the design space of tool parameters [3]–[6]. Although the vision of such modeling is captivating, it overlooks one critical characteristic: the inherently multi-stage structure of the EDA flow in the real world.

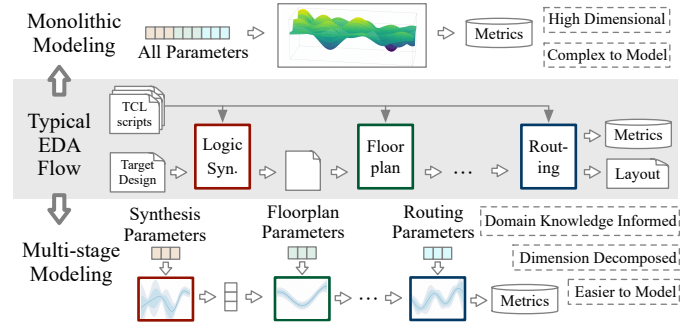


Fig. 1 Monolithic modeling of the EDA flow suffers from the ‘Curse of Dimensionality’ and complex parameter-metric mapping. In contrast, stage-aware modeling leverages domain knowledge to make full use of the precious training data and decomposes the mapping.

Our key insight is that the **multi-stage structure of EDA flows provides untapped domain knowledge that can dramatically improve optimization efficiency**. As shown in Fig. 1, the fundamental issue with monolithic approaches is that they ignore how EDA flows actually operate: as a cascade of interdependent stages where each stage’s output becomes the next stage’s input. Forcing AI to learn the entire parameter-to-metric mapping at once creates an unnecessarily complex learning problem—asking it to simultaneously capture all EDA tools’ effects while they compound across stages. Monolithic modeling of the mapping also suffers from the ‘Curse of Dimensionality’ [7], which is a critical defect in data-constrained scenarios. By exploiting the stage-wise structure, we can decompose this complex mapping into simpler sub-problems, where each stage learns how local parameters transform the inherited design state. This decomposition reduces dimensionality, captures hierarchical dependencies, and aligns with how engineers actually approach EDA optimization.

Building on this insight, we propose a multi-stage GP framework that mirrors the natural hierarchy of the EDA flow, with stage-wise GP layers that take both inherited features and local parameters as inputs. This decomposition mitigates the curse of dimensionality by breaking the parameter space into manageable chunks while explicitly modeling cross-stage dependencies. We also introduce Expected Hypervolume Improvement (EHVI)-Efficiency, an acquisition function that finally acknowledges what every engineer knows: time is money. By balancing expected design improvement against computa-

*Equal contributors

†Corresponding author.

tional cost and intelligently exploiting checkpoint reuse opportunities, EHVI-Efficiency ensures that every evaluation hour is spent wisely, focusing on parameter configurations that offer the best return on computational investment.

The contributions of this work represent a fundamental transition in how we approach EDA parameter optimization:

- We introduce a multi-stage perspective for EDA parameter optimization and a corresponding framework that optimizes based on the cascading dependencies between design stages.
- We develop a multi-stage Gaussian Process that decomposes the high-dimensional parameter space according to the natural stage boundaries, enabling efficient learning of complex parameter-metric relationships.
- We propose EHVI-Efficiency, a time-aware acquisition function that considers both expected design improvement and computational cost through runtime estimation and checkpoint reuse, aligning optimization strategies with practical time budgets.

II. PRELIMINARIES

A. VLSI Design Flow and EDA Tools' Parameters

Modern VLSI design follows a well-established flow that transforms RTL descriptions into manufacturable layouts through specialized EDA stages. As shown in Fig. 2, the industrial design flow consists of several key stages. The **Logic Synthesis stage** converts RTL into gate-level netlists, mapping PDK standard cells and optimizing for timing, power, and area constraints. The **Floorplan stage** determines die area, power domains, and macro placement strategies. During **Placement stage and CTS stage**, tools perform hierarchical placement optimization and construct balanced clock distribution networks. Finally, the **Routing stage** establishes physical interconnections through global and detailed routing.

B. Bayesian Optimization

Bayesian optimization (BO) [8] has emerged as the preferred method for expensive black-box optimization with severely limited evaluation budgets [9]. Instead of treating each evaluation independently, BO builds a probabilistic model of the objective function to intelligently guide the search. This iterative framework suits EDA parameter tuning, where each PPA evaluation consumes hours of compute time.

At the core of BO, the Gaussian process (GP) serves as a surrogate model. Given observed configurations \mathbf{X} with corresponding metrics \mathbf{y} , the GP provides predictions for any new configuration \mathbf{x}^* as:

$$f(\mathbf{x}^*) \sim \mathcal{N}(\mu(\mathbf{x}^*), \sigma^2(\mathbf{x}^*)), \quad (1)$$

where $\mu(\mathbf{x}^*)$ represents the expected PPA value and $\sigma^2(\mathbf{x}^*)$ quantifies uncertainty.

The BO framework operates through an iterative cycle: given the current GP model, an acquisition function $\alpha(\mathbf{x})$ leverages both prediction and uncertainty to select the next parameter configuration for evaluation. After evaluating the selected configuration, the new observation is incorporated into

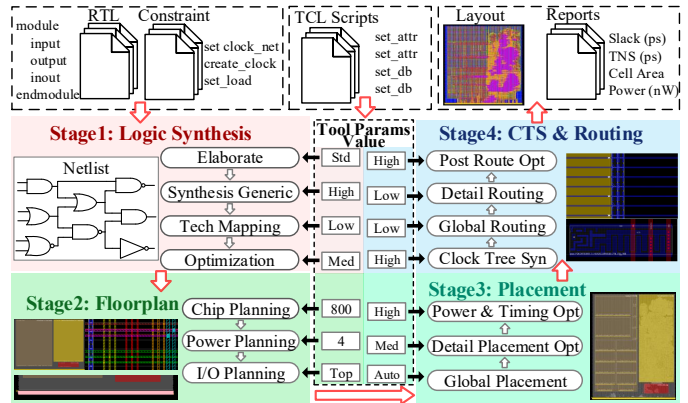


Fig. 2 A typical VLSI design's EDA Flow

the observation history to update the GP, and the iteration repeats until the evaluation budget is exhausted.

However, standard BO formulations poorly match the characteristics of EDA flows. The monolithic formulation of parameter-metric mapping forces a single GP to model complex cross-stage interactions without leveraging stage-wise structure or intermediate results. Classical acquisition functions also ignore varying computational costs and checkpoint opportunities, leading to inefficient use of limited time budgets.

C. Problem Formulation

We consider the EDA parameter optimization problem with a multi-stage structure. Let the EDA flow consist of N sequential stages (e.g., synthesis, placement, routing), where each stage $i \in \{1, \dots, N\}$ has its own parameter vector $\mathbf{x}_i \in \mathcal{X}_i$. The complete parameter configuration is $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N]$ with total dimension $d = \sum_{i=1}^N |\mathbf{x}_i|$.

Definition 1 (Pareto optimality). *For the multi-objective minimization problem with N objectives, a parameter combination X_1 is deemed to dominate parameter combination X_2 , which can be symbolized as $X_1 \geq X_2$, if for all n belonging to the objective set $1, \dots, N$, the inequality of objective vectors $f_n(X_1) \leq f_n(X_2)$ always true, and there exists at least one objective k for which $f_k(x_1) < f_k(x_2)$.*

A set of parameters that is not dominated by other parameters forms the Pareto optimal set [10], establishing Pareto optimality within the parameter space. The Pareto optimal parameter set represents the best trade-off among competing objectives and is referred to as the Pareto front.

Problem 1 (Parameter Tuning for EDA synthesis Tool). *Given the multi-stage structure of EDA tools' parameters and the QoR metrics to be optimized, the objective of EDA tool parameter tuning is to automatically search the Pareto-optimal parameter configurations in a given time budget, which bring about the high design quality concerning multiple QoR metrics like delay versus power versus area.*

III. MULTI-STAGE OPTIMIZATION FLOW

Our multi-stage Bayesian optimization framework orchestrates the complex interplay between intelligent parameter exploration and practical computational constraints in EDA

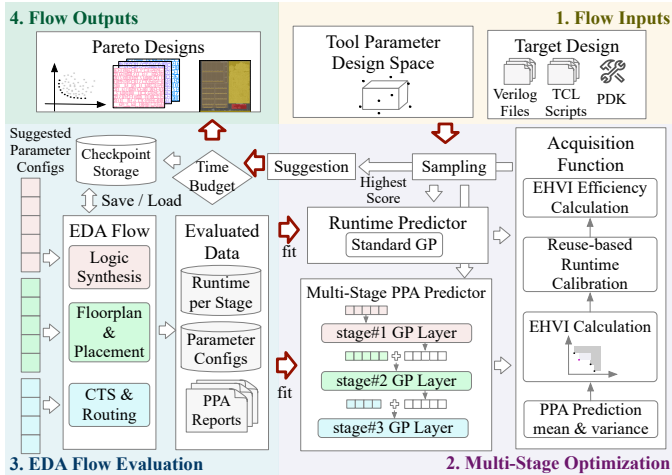


Fig. 3 Overview of the proposed multi-stage Bayesian optimization framework for EDA parameter tuning, featuring stage-aware metric modeling and efficiency-based acquisition. The flow starts with stage 1, iterates between stage 2 & 3, and ends with stage 4.

tool optimization. Fig. 3 shows the overall flow of the proposed framework. It operates as a closed-loop system beginning with stage 1 and ending with stage 4, where each iteration between stage 2 and stage 3 refines both our understanding of the parameter-PPA relationship and our ability to exploit computational shortcuts through checkpoint reuse.

In the first stage **Flow Inputs**, we establish the foundation for exploration by preparing the target design artifacts—including Verilog files, TCL scripts, and PDK information—alongside defining the complete EDA tool parameter design space across all stages. This phase sets the boundaries for our search while ensuring all necessary design collateral is ready for repeated evaluation cycles.

Once the parameter design space is settled, the framework enters its main optimization loop, where stage 2, **Multi-Stage Optimization**, drives intelligent parameter selection. At each iteration, our system samples candidate configurations from the parameter design space and feeds them through two parallel predictive pathways: a Runtime Predictor that estimates computational cost and a Multi-stage PPA Predictor that forecasts design quality metrics with associated uncertainty bounds. These predictions flow into our EHVI-Efficiency acquisition function, which evaluates each candidate by balancing expected PPA improvement against computational investment while accounting for potential time savings from checkpoint reuse when early-stage parameters remain consistent. The candidate achieving the highest efficiency score becomes our suggestion for actual evaluation.

The selected configuration then enters stage 3, **EDA Flow Evaluation**, where our framework’s awareness of stage-wise structure delivers practical benefits. Rather than naively executing the entire flow from scratch, the system examines which early-stage parameters match previous runs and retrieves corresponding checkpoints from storage, resuming computation only from the point of divergence. As the EDA flow

progresses through different stages, we capture stage-level checkpoints, runtime measurements, and ultimately the complete PPA report. These observations enrich our dataset, fitting both predictors in each iteration to improve their accuracy.

This iterative refinement continues until the allocated time budget expires, at which point the framework transitions to **Flow Outputs**. Each point on the frontier of evaluated data represents a different trade-off among power, performance, and area metrics, providing engineers with a curated set of high-quality solutions. The framework thus transforms the daunting challenge of navigating vast parameter spaces into a systematic, budget-aware exploration that leverages both the structural properties of EDA flows and the practical realities of computational resources.

IV. STAGE-AWARE GAUSSIAN PROCESS REGRESSION FOR METRIC PREDICTION

A. From Monolithic Formulation to Multi-Stage Formulation

Traditional Bayesian optimization approaches treat EDA flow as a monolithic function $f : \mathbb{R}^d \rightarrow \mathbb{R}^m$, where d denotes the total number of tunable parameters across all EDA stages and m represents the number of design metrics (e.g., power, performance, area). This approach aggregates parameters from all EDA stages into one high-dimensional input vector. This monolithic perspective encounters the curse of dimensionality and discards valuable domain knowledge about the cascading structure of EDA flows.

The key insight is recognizing that EDA flows are inherently compositional. Instead of optimizing a single high-dimensional function, we decompose it as:

$$f(\mathbf{x}) = f_L(f_{L-1}(\dots f_2(f_1(\mathbf{x}_1), \mathbf{x}_2) \dots, \mathbf{x}_{L-1}), \mathbf{x}_L), \quad (2)$$

where L is the total number of EDA stages, f_i represents the transformation performed by the i -th EDA stage ($i = 1, 2, \dots, L$), and $\mathbf{x}_i \in \mathbb{R}^{s_i}$ denotes the parameter vector specific to stage i . This nested representation clearly illustrates the sequential data flow: stage 1 processes parameters \mathbf{x}_1 to produce intermediate results, which are then combined with \mathbf{x}_2 in stage 2, and so forth until the final stage L produces the output. This formulation mirrors how EDA tools actually operate: With fixed target design, the synthesis stage takes in synthesis parameters and transforms RTL into netlists, the floorplan and placement stage takes in corresponding parameters and inherits netlists to produce physical coordinates, and so forth until the final stage outputs a layout with metric reports.

B. Theoretical Foundations and Complexity Analysis

Under standard smoothness assumptions, estimating a function $f : \mathbb{R}^d \rightarrow \mathbb{R}^m$ to ϵ -accuracy requires sample complexity:

$$c_{\text{black-box}} = O\left(\left(\frac{1}{\epsilon}\right)^d\right) \quad (3)$$

where $\epsilon > 0$ is the desired approximation error. This exponential dependence on dimension d is prohibitive for high-dimensional EDA parameter spaces.

For our multi-stage approach, each stage i learns a mapping from dimension $(d_i + q_{i-1})$ to dimension q_i (with $q_0 = 0$ and $q_L = m$). The total sample complexity becomes:

$$\mathcal{C}_{\text{multi-stage}} = \sum_{i=1}^L O\left(\left(\frac{1}{\epsilon}\right)^{d_i + q_{i-1}}\right) \quad (4)$$

For EDA flows with balanced stages parameters ($d_i \approx d/L$) and compact intermediate representations ($q_i = O(\log d)$), we have: $\mathcal{C}_{\text{multi-stage}} \ll \mathcal{C}_{\text{black-box}}$.

For example, with $d = 30$, $L = 3$, and $q_i = 5$, the black-box approach requires $O(2^{30}) \approx 10^9$ samples, while our approach needs only $3 \cdot O(2^{15}) \approx 10^5$ samples—a four orders of magnitude reduction.

This exponential improvement in sample efficiency is crucial for expensive EDA tool evaluations, where each design point may require hours of computation. The stage-wise decomposition effectively converts an intractable high-dimensional optimization into a sequence of tractable lower-dimensional problems.

C. Multi-Stage GP Implementation

We partition the d -dimensional parameter space $\mathbf{x} \in \mathbb{R}^d$ according to stage allocation $\{s_1, s_2, \dots, s_L\}$, where s_i represents the number of parameters belonging to stage i , and $\sum_{i=1}^L s_i = d$:

$$\mathbf{x} = [\mathbf{x}_1; \mathbf{x}_2; \dots; \mathbf{x}_L]. \quad (5)$$

Our Multi-Stage Gaussian Process consists of L interconnected GP layers, each modeling the transformation performed by its corresponding EDA stage. The architecture enforces the cascading information flow inherent in EDA processes through carefully designed input-output relationships between adjacent layers.

Initial Stage: The first GP layer operates solely on the initial stage parameters:

$$\mathbf{h}_1 = f_1(\mathbf{x}_1) \sim \mathcal{GP}_1(\mathbf{x}_1; \boldsymbol{\theta}_1), \quad (6)$$

where $\mathbf{h}_1 \in \mathbb{R}^{q_1}$ represents the learned intermediate features capturing the essential characteristics of the design after stage 1, and $\boldsymbol{\theta}_1$ denotes the hyperparameter set of the first GP (including kernel parameters, noise variance, etc.).

Intermediate Stages ($i = 2, \dots, L-1$): Each intermediate stage receives both the learned features from the previous stage and its own parameters:

$$\mathbf{h}_i = f_i([\mathbf{h}_{i-1}; \mathbf{x}_i]) \sim \mathcal{GP}_i([\mathbf{h}_{i-1}; \mathbf{x}_i]; \boldsymbol{\theta}_i), \quad (7)$$

where $[\mathbf{h}_{i-1}; \mathbf{x}_i]$ denotes the concatenation of the previous stage's output features with the current stage's parameters. This formulation ensures that stage i has access to the cumulative design information from all previous stages through \mathbf{h}_{i-1} while incorporating its own parameter-specific transformations through \mathbf{x}_i .

Final Stage: The last stage produces the final design metrics:

$$\mathbf{y} = f_L([\mathbf{h}_{L-1}; \mathbf{x}_L]) \sim \mathcal{GP}_L([\mathbf{h}_{L-1}; \mathbf{x}_L]; \boldsymbol{\theta}_L), \quad (8)$$

where $\mathbf{y} \in \mathbb{R}^m$ represents the final design metrics (power, performance, area).

For prediction at a new configuration $\mathbf{x}^* = [\mathbf{x}_1^*; \mathbf{x}_2^*; \dots; \mathbf{x}_L^*]$, we perform sequential forward propagation through the cascaded GP layers:

$$p(\mathbf{h}_1^* | \mathbf{x}_1^*, \mathcal{D}) \sim \mathcal{N}(\boldsymbol{\mu}_1^*, \boldsymbol{\Sigma}_1^*) \quad (9)$$

$$p(\mathbf{h}_2^* | \mathbf{h}_1^*, \mathbf{x}_2^*, \mathcal{D}) \sim \mathcal{N}(\boldsymbol{\mu}_2^*, \boldsymbol{\Sigma}_2^*) \quad (10)$$

⋮

$$p(\mathbf{y}^* | \mathbf{h}_{L-1}^*, \mathbf{x}_L^*, \mathcal{D}) \sim \mathcal{N}(\boldsymbol{\mu}_y^*, \boldsymbol{\Sigma}_y^*), \quad (11)$$

where $\mathcal{D} = \{(\mathbf{x}^{(j)}, \mathbf{y}^{(j)})\}_{j=1}^n$ represents the dataset with n observed design configurations and their corresponding objective evaluations, $\boldsymbol{\mu}_i^*$ and $\boldsymbol{\Sigma}_i^*$ denote the predictive mean and covariance matrix for the i -th stage output at the test point \mathbf{x}^* , respectively. The predictive uncertainty naturally propagates through the cascade, capturing both parameter uncertainty within each stage and accumulated uncertainty from previous stages.

V. EXPECTED HYPERVOLUME IMPROVEMENT WITH RUNTIME EFFICIENCY

A. Runtime-Aware Acquisition Function

In Bayesian optimization, acquisition functions guide the selection of the next evaluation point by balancing exploration of uncertain regions with exploitation of promising areas. For multi-objective optimization problems [11] like EDA parameter tuning—where we simultaneously optimize power, performance, and area—Expected Hypervolume Improvement (EHVI) [12] has emerged as the acquisition function of choice. EHVI quantifies the expected increase in the hypervolume of the Pareto front, providing a scalar measure of multi-objective improvement. Mathematically, for a candidate point \mathbf{x} , EHVI computes:

$$\text{EHVI}(\mathbf{x} | \mathcal{D}) = \mathbb{E}[\text{HV}(\mathcal{P} \cup \{\mathbf{y}(\mathbf{x})\}) - \text{HV}(\mathcal{P}) | \mathcal{D}], \quad (12)$$

where \mathcal{P} is the current Pareto front, $\mathbf{y}(\mathbf{x})$ represents the uncertain objective values at \mathbf{x} , and HV denotes the hypervolume indicator.

However, traditional EHVI overlooks a critical reality: different parameter configurations trigger vastly different computational costs in EDA tools. Under strict runtime budgets and tape-out deadlines, we must transition from optimizing design quality alone to maximizing quality improvement per compute hour.

We propose EHVI-Efficiency, which explicitly balances expected design improvement against computational cost:

$$\alpha_{\text{EHVI-E}}(\mathbf{x}) = \frac{\text{EHVI}(\mathbf{x} | \mathcal{D})}{T_{\text{expected}}(\mathbf{x})}, \quad (13)$$

where $T_{\text{expected}}(\mathbf{x})$ is the expected runtime for configuration \mathbf{x} .

To estimate runtime, we construct an auxiliary Gaussian process: trained on historical parameter-runtime pairs $\mathcal{D}_T = \{(\mathbf{x}^{(j)}, t^{(j)})\}_{j=1}^n$. The expected runtime is: $T_{\text{expected}}(\mathbf{x}) = \mathbb{E}[T(\mathbf{x}) | \mathcal{D}_T]$.

TABLE I Benchmark Designs

Name	#Cells	Process Node	Clock Period (ps)
fft16	6025	asap7-7nm	2000
aes128	17321	asap7-7nm	1000
picorv32	5376	freepdk-45nm	1000
dark_riscv	165453	freepdk-45nm	2000
gcd_unit	318	skywater-130nm	100
nanoRV32	5798	skywater-130nm	500

This formulation naturally handles exploration-exploitation-runtime trade-offs: configurations with high expected runtime are automatically down-weighted unless they offer proportionally larger quality improvements.

B. Checkpoint-Aware Runtime Modeling

Modern EDA flows save intermediate results (checkpoints) after each stage. When exploring parameters, we can reuse these cached results for substantial computational savings.

For a new configuration $\mathbf{x}^{\text{new}} = [\mathbf{x}_1^{\text{new}}; \mathbf{x}_2^{\text{new}}; \mathbf{x}_3^{\text{new}}]$ and a cached configuration $\mathbf{x}^{\text{cached}}$, we identify the reusable prefix:

$$k^* = \max\{k : \mathbf{x}_i^{\text{new}} = \mathbf{x}_i^{\text{cached}}, \forall i \in \{1, \dots, k\}\}. \quad (14)$$

For example, if $k^* = 2$, we might skip synthesis and placement, directly loading the placed design and proceeding with CTS and routing. The time saved is:

$$T_{\text{saved}} = \sum_{i=1}^{k^*} T_i(\mathbf{x}_i^{\text{cached}}), \quad (15)$$

where $T_i(\mathbf{x}_i)$ is the runtime of stage i .

After adjusting the runtime prediction incorporating checkpoint reuse, our final acquisition function is:

$$\alpha_{\text{EHVI-E}}(\mathbf{x}) = \frac{\text{EHVI}(\mathbf{x}|\mathcal{D})}{T_{\text{total}}(\mathbf{x}) - \max_{\mathbf{x}^c \in \mathcal{C}} T_{\text{saved}}(\mathbf{x}, \mathbf{x}^c)}, \quad (16)$$

where \mathcal{C} is the set of cached configurations.

This formulation creates intelligent exploration patterns: the optimizer develops affinity for configurations sharing early-stage parameters with promising cached results, effectively creating exploration branches. When discovering strong synthesis parameters, it thoroughly explores downstream placement and routing variations before attempting new synthesis configurations—mirroring expert engineering strategies.

VI. EXPERIMENTAL RESULTS

A. Experimental Setup and Benchmarks

We test our Explorer with a VLSI design flow consisting of Cadence Genus (version 21.1) and Innovus (version 20.1), on a platform with 2 Xeon Platinum 8383C CPU processors.

To evaluate the effectiveness of our approach, our experiments are conducted on 6 different benchmark designs using 3 different process nodes, including a 7-nm [13], 45-nm [14], and 130-nm [15] technology nodes.

TABLE I shows the benchmark designs in our experiment; Gcd_Unit [16] is a fundamental arithmetic unit used for calculating the greatest common divisor. fft16 [17] is a 16-bit wide computational unit that implements the Fast Fourier

TABLE II Examples of Parameters

Parameters	Stage	Range
<i>syn_generic_effort</i>		high/medium/low
<i>syn_map_effort</i>	synthesis(8)	high/medium/low
<i>syn_opt_effort</i>		high/medium/low/none
<i>aspect_ratio</i>		0.5-2.0
<i>core_margin</i>	floorplan(7)	io/die
<i>stdCellDensity</i>		0.4-0.7
<i>place_global_clock_gate_aware</i>		true/false
<i>place_global_clock_power_driven_effort</i>	placement(8)	high/standard/low
<i>place_global_soft_guide_strength</i>		high/medium/high
<i>routeWithLithoDriven</i>		true/false
<i>routeWithTimingDriven</i>	routing(5)	true/false
<i>routePostRouteViaPillarEffort</i>		high/medium/low

Transform (FFT) based on butterfly operations. aes128 [18] is a hardware accelerator for the AES encryption algorithm, supporting encryption and decryption with a 128-bit key length. NanoRV32 [19], picorv32 [20], and Dark_riscv [21] are processor designs based on the RV32I instruction set, incorporating basic arithmetic and logical functions as well as communication interfaces.

With base EDA flow script generated using mflowgen [22], 25 design performance-relative parameters are selected for experiments according to the design experience; parameters related to optimization options and adjustment effort are primarily selected. These parameters are divided into three stages, 8 for logic synthesis, 12 for floorplan and placement, 5 for CTS and routing. Total negative slack(TNS), power, and area are chosen as the metrics to be optimized simultaneously. Examples of the chosen tool parameters in each EDA stage are shown in TABLE II.

B. Comparisons Against SOTA Works

We compare our method with the following baselines:

- MLCAD'19 [4]: A Bayesian optimization method.
- ASPDAC'20 [3]: XGBoost-based method enhanced with feature learning.
- TCAD'22 [5]: A multi-objective Bayesian optimization method with multi-task GP.
- DATE'24 [6]: A Bayesian optimization method using the hybrid space Gaussian process.
- Ours w/o EHVI Efficiency: Our method without using EHVI-Efficiency. Standard EHVI is used as the acquisition function instead.

For each design in our benchmark, a total runtime budget of 50 EDA flow runs under default EDA tool parameters is assigned to each method.

To comprehensively evaluate the performance of our method against the baselines, we use several different metrics: TNS-power hypervolume $HV_{0,1}$, TNS-area hypervolume $HV_{0,2}$, and hypervolume of all targeted metrics HV .

The results in TABLE III present the comparative performance of our method against state-of-the-art (SOTA) baselines in the 6 benchmark designs, while Fig. 4 gives the visualization. For each design, we highlight the best result among the three metrics, and it can be observed that our method consistently outperforms the SOTA by a clear margin.

TABLE III Comparisons between our approach and SOTA works on the 6 benchmark designs.

Design	Metric	MLCAD'19 [4]	ASPDAC'20 [3]	TCAD'22 [5]	DATE'24 [6]	Ours w/o EHVI Efficiency	Ours
fft16	$HV_{0,1}$	32.68	31.63	34.95	36.03	37.56	39.23
	$HV_{0,2}$	40.86	39.89	42.03	43.77	45.15	46.88
	HV	44.78	43.61	46.51	47.26	48.50	49.49
aes128	$HV_{0,1}$	37.25	36.22	49.96	46.02	47.33	49.19
	$HV_{0,2}$	49.21	44.31	55.12	54.67	55.29	57.86
	HV	56.46	53.34	60.62	62.15	64.87	66.94
picorv32	$HV_{0,1}$	79.47	80.02	80.78	80.17	82.94	84.21
	$HV_{0,2}$	84.19	84.72	85.10	85.32	89.46	90.22
	HV	93.63	94.11	94.95	94.53	96.33	98.96
dark_riscv	$HV_{0,1}$	30.26	36.24	36.88	35.42	37.52	39.21
	$HV_{0,2}$	36.82	41.55	42.15	42.31	43.24	44.34
	HV	39.33	43.27	45.06	45.14	46.65	47.11
gcd_unit	$HV_{0,1}$	67.30	63.39	72.63	70.79	74.51	74.19
	$HV_{0,2}$	70.16	66.54	73.18	72.82	78.52	79.28
	HV	78.83	72.50	82.00	80.39	85.03	85.96
nanoRV32	$HV_{0,1}$	46.92	51.92	61.93	62.04	63.11	64.84
	$HV_{0,2}$	49.93	68.62	66.38	65.33	65.82	66.58
	HV	53.33	74.04	76.38	76.97	78.71	80.27
Average $HV_{0,1}$		48.98 (0.83)	49.90 (0.85)	56.18 (0.96)	55.08 (0.94)	57.11 (0.98)	58.53 (1.00)
Average $HV_{0,2}$		55.20 (0.86)	57.61 (0.90)	60.66 (0.94)	60.70 (0.95)	62.91 (0.98)	64.19 (1.00)
Average HV		61.06 (0.85)	63.48 (0.88)	67.59 (0.94)	67.74 (0.95)	70.02 (0.98)	71.46 (1.00)

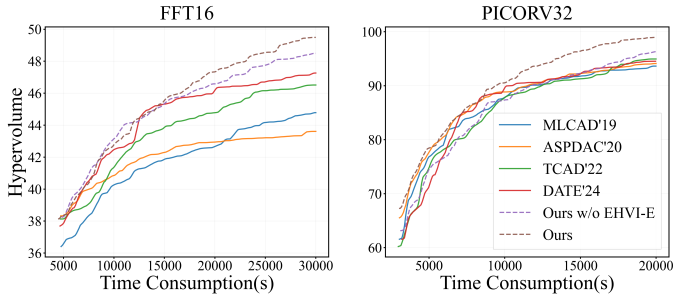


Fig. 4 The visualization of different methods' hypervolume changes with time budget consumption on 2 benchmarks.

C. Ablation Study

To validate the effectiveness of our multi-stage Gaussian process model in learning parameter-PPA relationships with limited data, we conduct a systematic comparison under varying training set sizes. We evaluate three GP variants:

- Traditional GP treating all parameters as a single high-dimensional input.
- Multi-layer GP with stage-wise structure, treating all parameters as a single high-dimensional input (a 3-layer deep GP [23]).
- Our multi-stage GP modeling the cascading EDA flow.

For each training size n in $[10, 20, 40, 80]$, we randomly sample n configurations from our collected dataset and train each model. 2000 remaining samples are served as a test set for fair comparison. We repeat this process 20 times with each model sharing the same random seed each time.

Fig. 5 shows the prediction R^2 values for the power and area metric in *gcd_unit* benchmark. The results demonstrate that our multi-stage GP consistently outperforms other GP approaches in the data-scarce regime common in EDA flow management and parameter optimization. This superior data efficiency stems from our method's ability to leverage domain

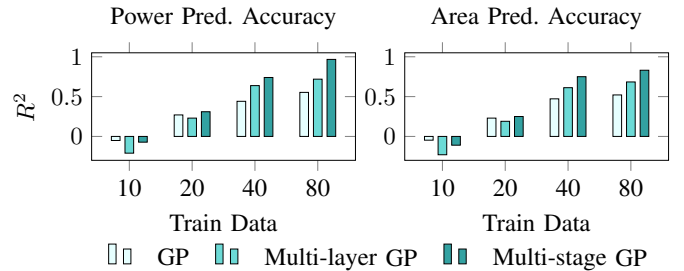


Fig. 5 Power/Area Prediction under Different Training Size.

knowledge about EDA stage dependencies, effectively reducing the complexity of the learning problem through structured decomposition rather than brute-force modeling.

VII. CONCLUSION

To enhance parameter optimization for management of today's EDA flow, this paper proposes a stage-aware parameter tuning framework that leverages the inherent stage-wise information in the EDA flow, fundamentally breaking away from conventional monolithic modeling approaches. We present a multi-stage Gaussian Process framework that exploits the natural hierarchy of EDA flows to efficiently learn complex parameter-metric relationships with limited evaluations. We also introduce EHVI-Efficiency, a runtime-aware acquisition function that balances design improvement with EDA tools, runtime, ensuring optimization strategies remain both effective and practical. Experimental results and ablation studies on 6 different designs across 3 process nodes demonstrate the effectiveness of our approach, showing improvements over state-of-the-art methods.

ACKNOWLEDGMENT

This work is sponsored by Natural Science Foundation of Shanghai (Project No.25JD1403000)

REFERENCES

- [1] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2016, pp. 785–794.
- [2] Rasmussen and C. Edward, "Gaussian processes in machine learning," in *Summer school on machine learning*. Springer, 2003, pp. 63–71.
- [3] Z. Xie, G.-Q. Fang, Y.-H. Huang, H. Ren, Y. Zhang, B. Khailany, S.-Y. Fang, J. Hu, Y. Chen, and E. C. Barboza, "FIST: A Feature-Importance Sampling and Tree-Based Method for Automatic Design Flow Parameter Tuning," in *Proc. ASPDAC*, 2020, pp. 19–25.
- [4] Y. Ma, Z. Yu, and B. Yu, "CAD tool Design Space Exploration via Bayesian Optimization," in *ACM/IEEE Workshop on Machine Learning CAD (MLCAD)*, 2019, pp. 1–6.
- [5] H. Geng, T. Chen, Y. Ma, B. Zhu, and B. Yu, "PTPT: Physical Design Tool Parameter Tuning via Multi-objective Bayesian Optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 42, no. 1, pp. 178–189, 2022.
- [6] D. Luo, Q. Sun, Q. Xu, T. Chen, and H. Geng, "Attention-based eda tool parameter explorer: From hybrid parameters to multi-qor metrics," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2024.
- [7] Z. Wang, F. Hutter, M. Zoghi, D. Matheson, and N. De Freitas, "Bayesian optimization in a billion dimensions via random embeddings," *Journal of Artificial Intelligence Research*, vol. 55, pp. 361–387, 2016.
- [8] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Proc. NIPS*, vol. 25, 2012.
- [9] 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.
- [10] Y. Censor, "Pareto optimality in multiobjective problems," *Applied Mathematics and Optimization*, vol. 4, no. 1, pp. 41–59, 1977.
- [11] K. Yang, M. Emmerich, A. Deutz, and T. Bäck, "Multi-objective bayesian global optimization using expected hypervolume improvement gradient," *Swarm and evolutionary computation*, vol. 44, pp. 945–956, 2019.
- [12] M. T. Emmerich *et al.*, "Hypervolume-based expected improvement: Monotonicity properties and exact computation," in *Proc. CEC*, 2011.
- [13] L. T. Clark, V. Vashishtha, L. Shifren, A. Gujja, S. Sinha, B. Cline, C. Ramamurthy, and G. Yeric, "ASAP7: A 7-nm finFET predictive process design kit," *Microelectronics Journal*, vol. 53, pp. 105–115, 2016.
- [14] J. E. Stine, I. Castellanos, M. Wood, J. Henson, F. Love, W. R. Davis, P. D. Franzon, M. Bucher, S. Basavarajaiah, J. Oh, and R. Jenkal, "Freepdk: An open-source variation-aware design kit," in *2007 IEEE International Conference on Microelectronic Systems Education (MSE'07)*, 2007.
- [15] R. T. Edwards, "Google/skywater and the promise of the open pdk," in *Workshop on Open-Source EDA Technology*, 2020.
- [16] K. A.-H. *et al.* [Online], "Integrated gcd unit," Available: https://github.com/priyanka-raina/gcdunit_caravel_user_project, 2022.
- [17] A. [Online], "16-point fft is developed in order to accurately model that of the matlab function," Available: <https://github.com/ameyk1/Fast-Fourier-Transform>, 2016.
- [18] E. B. Joachim Strömbergson, Olof Kindgren and N. [Online], "Verilog implementation of the symmetric block cipher aes (advanced encryption standard) as specified in nist fips 197," Available: <https://github.com/secworks/aes>, 2023.
- [19] E. F. [Online], "Nanorv32 – a small rv32i[mc] core capable of running rtos," Available: <https://github.com/elvisfox/nanorv32>, 2022.
- [20] P. M. *et al.* [Online], "Picorv32 - a size-optimized risc-v cpu," Available: <https://github.com/darklife/darkriscv>, 2025.
- [21] J. H. *et al.* [Online], "opensouce risc-v cpu core implemented in verilog," Available: <https://github.com/YosysHQ/picorv32>, 2019.
- [22] A. Carsello, J. Thomas, A. Nayak, P.-H. Chen, M. Horowitz, P. Raina, and C. Torng, "mflowgen: A modular flow generator and ecosystem for community-driven physical design," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, 2022, pp. 1339–1342.
- [23] A. Damianou and N. D. Lawrence, "Deep gaussian processes," in *Artificial intelligence and statistics*. PMLR, 2013, pp. 207–215.