StateTune: Transforming LLM-Assisted EDA Flow Tuning into a Stateful, Closed-Loop Process

KunLong Li*
Fudan University
Shanghai, China
klli24@m.fudan.edu.cn

Shangshang Yao*
Independent Researcher
Shanghai, China
yaoshangshang96@outlook.com

Su Zheng
Chinese University of Hong Kong
Hong Kong, China
szheng22@cse.cuhk.edu.hk

Lingli Wang†
Fudan University
Shanghai, China
llwang@fudan.edu.cn

Abstract

EDA flow parameter tuning is critical for quality-of-results (QoR), yet the parameter space is large, tightly coupled, and full evaluations are prohibitively expensive. Prior LLM-assisted tuners mainly use the LLM as an external proposer with transient working context; we instead present **StateTune**, which reformulates LLM-assisted EDA tuning as a closed-loop, state-carrying process. Its optimizer state is a typed, evidence-gated *persistent optimization memory* that is updated by every evaluation and shared between candidate generation and budget allocation. On top of this optimizer state, an expected hypervolume improvement (EHVI)-guided, runtime-aware promotion policy ranks quick-stage candidates by expected Pareto frontier gain per unit of runtime cost. Evaluated on a Cadence industrial flow across six benchmark blocks (two technology nodes × three designs), against five baselines including LLM+retrieval-augmented generation (RAG) and preference-based Bayesian optimization (BO) tuners, StateTune achieves the strongest final hypervolume on all six benchmark blocks, showing a stable improvement in frontier quality across the full matrix; it also matches or surpasses the strongest baselines on worst negative slack (WNS), area, and power across the same set. Ablation shows persistent memory is the largest contributor: removing it costs 58.5% of the hypervolume. Dedicated analyses of evidence-gating sensitivity, memory poisoning, cross-design transfer, and three-seed reproducibility (CV < 7% on five of six blocks) further validate the memory design.

Keywords

EDA flow tuning, large language models, persistent memory, multi-fidelity optimization, EHVI, physical design

1 Introduction

Modern physical-implementation flows expose many interacting knobs across floorplanning, placement, clock-tree synthesis (CTS), and routing. These knobs are tightly coupled: even modest changes to utilization, timing effort, or density can materially alter timing, area, and power—or cause a run to fail. Because full register-transfer-level (RTL)-to-route evaluations take hours, a practitioner's tuning budget is strictly limited. This makes EDA flow tuning a *budgeted* decision problem that must simultaneously address three challenges: proposing good configurations, learning from each evaluation, and deciding which candidates deserve the scarce full-flow budget.

Automated methods have made significant progress. METRICS2.1 [10] standardized no-human-in-the-loop evaluation, and algorithmic advances such as PTPT [7], REMOTune [25], RankTuner [21], and FlowTuner [13] have demonstrated multi-objective Bayesian optimization, trust-region search, preference-based tuning, and cross-stage knowledge transfer. However, these methods treat the EDA tool as a black box: the design knowledge uncovered during tuning—which parameter ranges trigger failures, which interactions help, and why configurations succeed or fail—remains implicit in the optimizer state and is discarded after each run, so subsequent iterations and new runs cannot benefit from earlier insights.

Large language models (LLMs) have been adopted in EDA for tool interaction, documentation retrieval, and design-space exploration [9, 20, 26]. For flow tuning specifically, CROP combined circuit-level retrieval-augmented generation (RAG) with LLM-guided search [14] and ORFS-agent demonstrated iterative LLM-based parameter optimization for OpenROAD [8], showing that LLMs can leverage domain knowledge to generate stronger proposals than blind numerical search. However, these systems primarily rely on transient search context assembled at inference time rather than a structured, evolving memory that distills experience into reusable artifacts. Moreover, they focus exclusively on candidate *generation* and leave the complementary *promotion* decision—which of several proposals deserves the scarce full-flow budget—to ad-hoc filtering or unselective evaluation.

StateTune takes a different route. Rather than treating the LLM as a smarter proposer bolted onto a standard tuning loop, we *reformulate* LLM-assisted EDA tuning as a closed-loop, state-carrying process whose optimizer state is an explicit, typed, evidence-gated **persistent optimization memory** updated by every evaluation and shared between candidate generation and budget allocation (Section 3.2). On top of this state, an **expected hypervolume improvement (EHVI)-guided, runtime-aware promotion** policy ranks candidates by expected Pareto frontier gain per unit of runtime cost, directing the expensive budget toward the most promising proposals. We evaluate StateTune on a Cadence Genus/Innovus industrial flow across ASAP7 and NanGate45 with benchmark blocks covering JPEG, AES, and IBEX, and compare against five baselines including two strong LLM/retrieval-based tuners. Our code is open-sourced in our repository.

The main contributions are:

- **A closed-loop reformulation of LLM-assisted EDA tuning** in which the optimizer state is a typed, evidence-gated persistent memory $\mathcal{M}_t$ that jointly drives candidate generation and budget allocation. Evidence gating prevents knowledge poisoning [3]; ablation shows this is the largest single contributor (HV −58.5% when removed) (Section 3.2).

*Both authors contributed equally to this research.
†Corresponding author.

- **EHVI-guided, runtime-aware promotion** that ranks quick-stage candidates by expected Pareto frontier gain per unit of runtime risk, directly reading from $\mathcal{M}_t$ so that improvements to memory propagate to budget allocation through a shared state object.
- **Empirical validation on a 2×3 benchmark matrix** (two technology nodes × three designs), with a six-way comparison against BO (qEHVI), Optuna-TPE, Random, RankTuner [21], and CROP [14]. StateTune attains the best WNS, best power, and best hypervolume on all six benchmark blocks, together with the best or tied-best area on all six. Component-level evidence from five ablation variants, dedicated analyses of evidence-gating sensitivity and memory poisoning, and a three-seed reproducibility study support these results.

2 Related Work

We review three lines of work that StateTune builds upon: multi-objective optimization under budget constraints, structured memory for LLM agents, and automated EDA flow tuning.

2.1 Multi-Objective BO and Cost-Aware Search

Bayesian optimization (BO) models objectives via Gaussian process (GP) surrogates and uses acquisition functions to guide search [16]. In multi-objective settings, qEHVI [5] quantifies the expected volume gained by adding a new point to the Pareto frontier, providing a principled criterion for improving the entire frontier rather than a single metric. When evaluations are expensive, multi-fidelity strategies can improve sample efficiency by exploiting cheaper proxy evaluations before committing to full runs; FABOLAS and BOHB demonstrated this for hyperparameter optimization [6, 12]. Optuna applies tree-structured Parzen estimators (TPE) to multi-objective problems [1, 2]. StateTune adopts this budget-aware perspective but embeds it inside an EDA-specific loop where quick-stage memory, accumulated design knowledge, and runtime-aware promotion interact tightly.

2.2 Structured Memory for LLM Agents

A growing body of work shows that LLM agents benefit from structured memory beyond raw conversation history. Reflexion [17] introduced verbal reinforcement learning via trajectory-level reflection; A-MEM [22] organizes knowledge notes via Zettelkasten-style indexing; MemoryOS [11] designs a hierarchical short/mid/long-term memory; and CFGM [23] grounds memories at multiple granularities for planning. RSR [24] decomposes agent memory into retrieval, scheduling, and reflection stages, demonstrating that episodic memory with a reflection loop improves task performance in sequential decision-making.

A key concern with persistent memory is *quality*: incorrect or outdated information can degrade later decisions. AgentPoison [3] formalizes this risk in the security context, and in optimization settings the same risk arises self-inflicted when early-stage LLM diagnoses based on insufficient data produce incorrect rules.

StateTune differs from these general-purpose memory frameworks in two structural respects. First, its memory is *typed and evidence-gated*: each artifact (hard rule, soft heuristic, sensitivity pattern, failure summary) has an explicit type and activation condition, with hard rules requiring k=12 corroborating full-valid observations before activation; the memory-poisoning experiment in Section 5.4 confirms that removing this gating causes incorrect rules to accumulate and degrades HV by 43.4%. Second, the memory is *shared* between candidate generation and a downstream budget-allocation policy, so that an improvement to a rule or sensitivity pattern propagates to both proposal quality and promotion ranking through a single state object—a coupling that RSR's episodic reflection and A-MEM's note indexing do not provide (Table 1).

Table 1: Capability comparison of EDA flow tuning methods. ● = supported; ○ = not supported; ⊙ = partial.

Method	LLM search	RAG	Memory type	Gate	Multi-fid.	Multi-obj.	Cross-design
PTPT [7]	○	○	–	○	○	●	○
FlowTuner [13]	○	○	–	○	●[a]	○	○
RankTuner [21]	○	○	–	○	○	●	○
CROP [14]	●	●[b]	Prompt hist.	○	○	○	○
ORFS-agent [8]	●	○	Prompt hist.	○	○	●[c]	○
AstroTune [19]	⊙[d]	●	–	○	●	●	○
RSR [24]	●	⊙[f]	Episodic	○	○	○	○
StateTune (Ours)	●	●[e]	**Persistent**	●	●	●	●

[a] Stage transfer via jump-start/early-stop. [b] Design-level retrieval from similar circuits. [c] Weighted-sum / constrained multi-objective prompts. [d] Retrieval augmented by LLM; BO handles search. [e] Design- and run-level retrieval via RAG-EDA [15]. [f] Retrieval of past reflection traces; no EDA-specific RAG.

2.3 EDA Flow Tuning and LLM-Assisted Search

Automated EDA flow tuning has evolved from reusable infrastructure to specialized algorithmic search. METRICS2.1 [10] established a no-human-in-the-loop evaluation stack; PTPT [7] formulated tuning as multi-task GP-based BO; REMOTune [25] scaled to high dimensions via random embedding; RankTuner [21] introduced preference-based BO for noisy QoR; FlowTuner [13] transferred knowledge across stages. These methods advance numerical search but do not externalize the non-numerical knowledge—failure modes, parameter interactions, design heuristics—that emerges during tuning.

On the LLM side, ChatEDA [20] demonstrated autonomous EDA tool interaction; CROP [14] combined circuit-level RAG with LLM-guided tuning; ORFS-agent [8] showed iterative LLM-based parameter optimization for OpenROAD. AstroTune [19] integrated AST-based retrieval with stage-wise tournament BO, combining retrieval-augmented LLM proposals with a BO backbone; however, its retrieval feeds only the proposal stage and does not parameterize a downstream budget-allocation policy. Recent surveys provide broader coverage [9, 26]. These systems keep search context largely transient, and none externalizes a *typed, evidence-gated* optimizer state that is shared between candidate generation and budget allocation—the three structural gaps (typed state, evidence gating, shared promotion) that StateTune fills through a single persistent memory object $\mathcal{M}_t$.

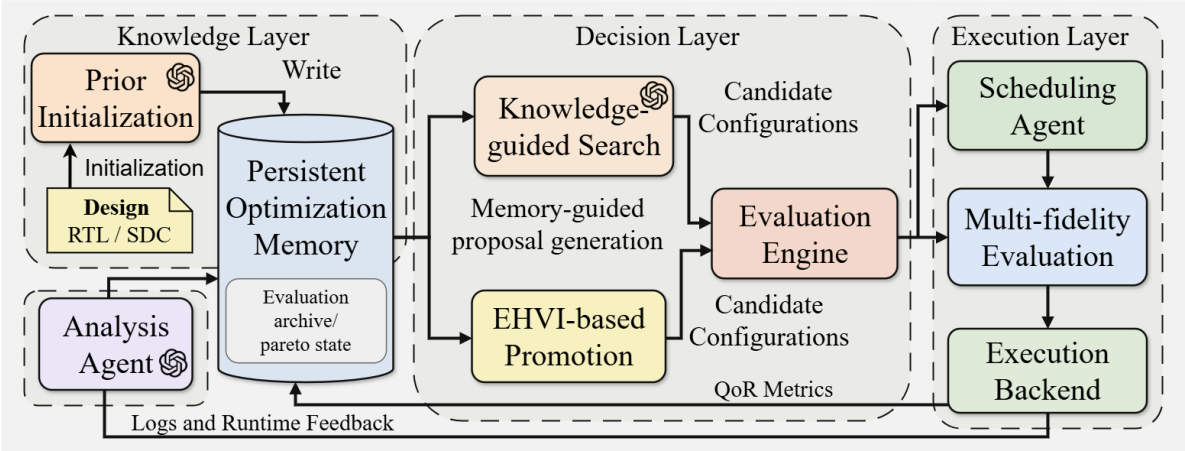


Figure 1: Overall workflow of StateTune. The knowledge layer (left) builds and updates persistent optimization memory from design inputs and evaluation feedback via Prior Initialization and an Analysis Agent. The decision layer (center) generates candidate configurations through Knowledge-guided Search and ranks them via EHVI-based Promotion; an Evaluation Engine arbitrates the final selection. The execution layer (right) schedules multi-fidelity evaluations through a Scheduling Agent and returns QoR metrics. The persistent optimization memory connects all three layers.

3 Method

This section describes the three layers of StateTune—knowledge, decision, and execution—and details the persistent optimization memory and EHVI-guided promotion that connect them.

3.1 Problem Setting and Overview

We consider a budgeted multi-objective tuning problem over a parameter vector $x \in \mathcal{X}$. A full evaluation $e(x)$ returns a QoR vector $\mathbf{f}(x) = \big(f_{\text{WNS}}(x), f_{\text{area}}(x), f_{\text{power}}(x)\big)$, where WNS denotes worst negative slack, runtime $c(x)$, and terminal status $\delta(x) \in \{0, 1\}$. The tuning goal is to approximate the Pareto frontier

$$\mathcal{P}^* = \big\{x \in \mathcal{X} : \nexists\, x' \text{ s.t. } \mathbf{f}(x') \succ \mathbf{f}(x)\big\} \tag{1}$$

so as to maximize the dominated hypervolume $\text{HV}(\mathcal{P}, \mathbf{r})$ with respect to a fixed reference point $\mathbf{r}$, subject to a total runtime budget $\sum_i c(x_i) \le B$. Because full evaluations are expensive, the system first executes a cheaper quick stage (cost $c_q \ll c_f$) and promotes only selected candidates to the full stage. The quick-stage cutoff point is configurable; in our experiments we use CTS (clock-tree synthesis) as the quick-stage endpoint, which provides richer timing signal than placement alone (Section 4). StateTune tunes 19 parameters spanning floorplanning, placement, pre-CTS/CTS, and routing (Table 5). The optimizer state is a typed, evidence-gated persistent optimization memory $\mathcal{M}_t$ carried across iterations, rather than a growing prompt transcript (detailed in Section 3.2).

StateTune is organized into three interacting layers (Figure 1).

The **knowledge layer** manages the persistent optimization memory. A Prior Initialization module initializes a design prior from Synopsys Design Constraints (SDC), RTL characteristics, and RAG-retrieved tool documentation. As evaluations complete, an Analysis Agent periodically runs failure pattern analysis, parameter sensitivity analysis (including threshold-effect detection), and prior refinement, feeding curated results back into memory.

The **decision layer** uses the memory to propose and filter candidate configurations. A Knowledge-guided Search module queries the memory and selects among five search modes—*explore*, *exploit*, *diversify*, *repair*, and *promote*—based on recent progress, frontier diversity, and failure patterns (detailed in Section 3.3). Chain-of-thought (CoT) reasoning distills the rich context into structured diagnoses, and a non-reasoning model formats final parameter vectors. An EHVI-based Promotion module then ranks the generated candidates by expected Pareto frontier gain per unit of runtime cost (Section 3.4), and an Evaluation Engine arbitrates the final selection for full evaluation.

The **execution layer** manages multi-fidelity scheduling through a Scheduling Agent. Quick-stage evaluations—CTS in our experiments, though the cutoff is configurable—are cheap and run continuously. Full-stage evaluations through routing are expensive and allocated via EHVI-guided promotion. Results and runtime measurements flow back to the knowledge layer, closing the loop.

The central design choice is that $\mathcal{M}_t$ is persistent and reusable: knowledge from early iterations directly shapes later proposals and promotion decisions. Three properties distinguish $\mathcal{M}_t$ from prior optimizer states: it stores typed artifacts (rules, sensitivities, failure patterns) that a GP posterior or transient context cannot represent robustly; hard rules require k=12 corroborating full-valid observations before activation; and the same shared object parameterizes both candidate generation and EHVI-guided promotion. Ablation confirms that this explicit optimizer state is the largest single contributor to final hypervolume (Section 5.2).

3.2 Persistent Optimization Memory

The persistent optimization memory at iteration t is

$$\mathcal{M}_t = \{P_t, R_t, F_t, S_t, H_t, \Pi_t, T_t\}, \tag{2}$$

where P_t is the design prior, R_t contains reusable rules, F_t is a failure summary, S_t is a sensitivity summary, H_t is the evaluation history,

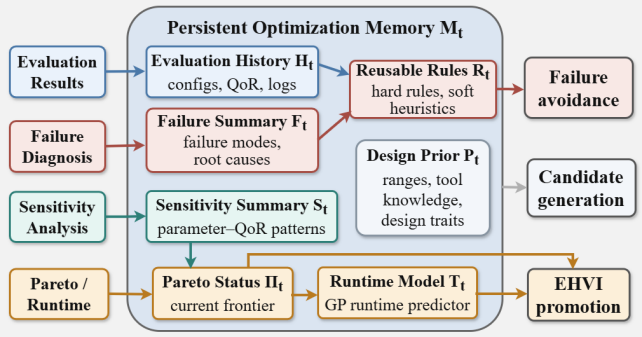


Figure 2: Structure of the persistent optimization memory $\mathcal{M}_t$. Evaluation results, failure diagnoses, and sensitivity analyses feed into six components. The design prior P_t conditions generation; reusable rules R_t guide failure avoidance; Pareto status Π_t and the runtime model T_t drive EHVI-based promotion.

Table 2: Persistent memory components.

Component	Content	Role
Prior P_t	Parameter ranges, tool knowledge, design characteristics	Conditions proposals
Rules R_t	Evidence-gated hard rules & soft heuristics	Filters proposals
Failure F_t	Categorized failures with root-cause analysis	Guides repair
Sensitivity S_t	Parameter–QoR correlations & threshold effects	Prioritizes knobs
History H_t	All configs, QoR, terminal status	LLM context & GP fit
Pareto Π_t	Current non-dominated frontier	EHVI targets
Runtime T_t	kNN runtime predictions	Promotion cost

Π_t is Pareto status, and T_t is the runtime model. Table 2 describes each component.

The prior P_t is initialized from SDC specifications, RTL characteristics, and RAG-retrieved tool documentation, then periodically refined as evaluations accumulate so that it adapts to each design's characteristics. The failure summary F_t periodically categorizes failed runs by type and associated parameter combinations, injecting dominant patterns as compact text into the search prompt in *repair* mode.

Sensitivity analysis. The sensitivity summary S_t combines Spearman rank correlations (with a weighted impact score across WNS, area, power) and a threshold-effect detector that flags nonlinear cliff effects invisible to correlation analysis alone. Both outputs are formatted as compact text and injected into the proposal prompt; concrete examples appear in Table 3.

Evidence gating and memory quality control. LLM-emitted rules based on insufficient data can be incorrect; if stored directly, such errors accumulate and progressively degrade search quality—a self-inflicted variant of the memory-poisoning risk documented in LLM agent systems [3]. Evidence gating prevents this: hard rules require at least k=12 corroborating full-valid evaluations before activation (Section 5.3 and Figure 5 validate this threshold), the number of active hard rules is capped at 8, and a periodic trim removes stale or contradicted entries. Soft heuristics act as overridable directional preferences. The memory-poisoning experiment in Section 5.4 confirms that removing gating degrades HV by 43.4%.

Table 3: Knowledge artifacts automatically generated by StateTune.

Type	Content	Evidence
Hard rule	Ban `CORE_UTIL=0.78` with `CORE_MARGIN=0`: catastrophic WNS (−32k) from placement congestion.	Multiple failures
Soft heur.	Increase `CORE_UTIL` & `PLACE_MAX_DENSITY`; keep `CORE_MARGIN` low. WNS corr. +0.24.	Sensitivity
Sensitivity	`CTS_POST_OPT` corr. −0.19 with WNS; its heuristics degrade setup timing.	365 runs
Prior upd.	`ASPECT_RATIO`: 1.2⟶1.0 (square). Rectangular shapes worse for bus-heavy datapath.	12 full-valid obs.

3.3 Knowledge-Guided Search

The knowledge-guided search module uses the persistent memory to generate context-aware candidate configurations. For each iteration, it selects among five search modes based on recent progress, frontier diversity, and failure patterns: *explore* broadens coverage of the parameter space; *exploit* refines configurations near the current best; *diversify* increases Pareto frontier spread; *repair* targets known failure modes using the failure summary F_t with corrective adjustments; and *promote* nominates strong quick-stage candidates for full evaluation. Mode selection follows a heuristic priority: *repair* is triggered when the recent failure rate exceeds a threshold; *promote* is selected when the promotion pool contains sufficiently strong candidates; among the remaining modes, the system alternates based on frontier stagnation (favoring *diversify*) and convergence signals (favoring *exploit*). The proposal prompt is conditioned on the parameter space, recent history, Pareto points, the updated design prior, failure and sensitivity summaries, and stage-specific constraints.

Dual-model inference. StateTune uses CoT to organize RAG-retrieved documentation, evaluation histories, and parameter interactions into structured failure and sensitivity diagnoses, while a non-reasoning model formats parsable candidate vectors. We use a strong open-weight reasoning model and a fast generation model (Table 4) to balance analytical depth with output reliability. Ablation shows that removing CoT reduces HV by 31.5% (Section 5.2). LLM calls are made only when the candidate buffer is depleted, not at every iteration, keeping token consumption bounded.

Candidate buffer and frontier-driven refresh. Each LLM call produces multiple candidate configurations that are stored in a buffer and consumed one-by-one in subsequent iterations. To ensure that proposals reflect the latest optimization state, the buffer is *flushed* whenever a new Pareto-optimal point is discovered or the hypervolume improves: all remaining buffered candidates are discarded, forcing the next iteration to issue a fresh LLM call conditioned on the updated history and frontier. This mechanism prevents the system from wasting iterations on candidates generated from stale context. **Stage-aware conditioning.** Quick-stage proposals are biased toward placement-effective parameters, since routing knobs have little visible effect at the quick stage.

Rule emission and statistical validation. The proposal module emits hard rules and soft heuristics stored in memory. Representative artifacts include banning parameter combinations repeatedly associated with catastrophic congestion, promoting sensitivity-backed density adjustments, and updating design priors when repeated full-valid observations support a tighter setting. Evidence gating (Section 3.2) prevents premature knowledge from entering

memory. Specifically, each candidate rule is validated *post hoc* against accumulated full-valid observations: the system partitions the observation set into rule-conforming and rule-violating groups and applies a one-sided Mann–Whitney U test ($\alpha = 0.05$) to determine whether the conforming group exhibits statistically significantly better QoR. Only rules that pass this test and have been corroborated by at least k observations are promoted to active status. Across the ASAP7 JPEG run, only 1 of 13 hard rules and 30 of 101 soft heuristics were validated; 6 soft heuristics were contradicted. This noise is precisely why evidence gating is necessary.

3.4 EHVI-Guided, Runtime-Aware Promotion

LLM-based search generates candidates of varying quality, so evaluating them indiscriminately wastes scarce full-evaluation budget. StateTune therefore ranks candidates by expected Pareto frontier gain per unit of runtime cost and promotes only the most promising ones.

To leverage the abundant quick-stage observations alongside scarce full-stage data, StateTune fits a multi-fidelity GP surrogate for each QoR objective. Quick-stage and full-stage histories are jointly modeled using a binary fidelity indicator (z=0 for quick, z=1 for full); when insufficient full-stage data makes the multi-fidelity fit unreliable, the system falls back to a standard single-fidelity GP trained on full-stage data only. At scoring time, candidates are projected to the full-fidelity surface for each QoR objective $f_k(x)$, $k = 1, \ldots, K$:

$$f_k(x) \sim \mathcal{GP}\big(\mu_k(x),\, \sigma_k^2(x)\big), \tag{3}$$

where $\mu_k(x)$ and $\sigma_k^2(x)$ are the posterior mean and variance given H_t, and $K = 3$ (WNS, area, power).

Given the current Pareto frontier Π_t and reference point $\mathbf{r}$, EHVI quantifies expected gain:

$$\mathrm{EHVI}(x) = \mathbb{E}_{\mathbf{f}(x)}\big[\mathrm{HV}(\Pi_t \cup \{\mathbf{f}(x)\},\, \mathbf{r}) - \mathrm{HV}(\Pi_t,\, \mathbf{r})\big], \tag{4}$$

following the qEHVI formulation [5].

Runtime cost model. Predicting full-stage runtime is challenging because few full evaluations are available early on. GP-based runtime models require careful kernel selection and sufficient data for reliable posteriors. StateTune uses a k-nearest-neighbor (kNN) estimator over H_t for runtime prediction, which is stable with small sample sizes, requires no hyperparameter tuning, and provides a natural uncertainty estimate via neighbor variance. Since promotion only needs to *rank* candidates correctly rather than predict absolute runtimes, kNN's simplicity is well matched to the task. The kNN model estimates runtime as:

$$\hat{t}(x) = \frac{1}{k} \sum_{i=1}^{k} t(x_i), \tag{5}$$

where x_i are the k nearest evaluated configurations. An uncertainty-aware upper confidence bound is:

$$\hat{t}_{\mathrm{ucb}}(x) = \hat{t}(x) + \beta \cdot s_t(x), \tag{6}$$

where $s_t(x)$ is the sample standard deviation and $\beta > 0$ controls cost pessimism. Promotion is ranked by:

$$\mathrm{score}(x) = \frac{\mathrm{EHVI}(x)}{\hat{t}_{\mathrm{ucb}}(x) + \epsilon}, \tag{7}$$

Table 4: Experimental setup and LLM overhead.

Benchmarks & Flow	
Flow	Cadence Genus 21.17 / Innovus 21.18
Benchmarks	JPEG / AES / IBEX on both ASAP7 [4] (7 nm) and NanGate45 [18] (45 nm)
Objectives	Maximize WNS; minimize area & power
Fidelities	Quick = CTS (configurable); Full = route
Parameters	19 across 4 stages (Table 5)
Budget & Hardware	
Main budget	192 thread·h† (4 threads × 48 h)
Ablation budget	48 thread·h (1 thread × 48 h)
CPU	Intel Xeon Plat. 8354H @ 3.10 GHz
Models & Baselines	
CoT model	Qwen3-235B-A22B-Thinking-2507
Non-reasoning	DeepSeek-V3.2
RAG framework	RAG-EDA [15]
Baselines	BO (qEHVI), Optuna-TPE, Random, RankTuner (ICCAD'24) [21], CROP (ICCAD'25) [14]
Ablations	w/o KA, w/o Persist., w/o EHVI, w/o RAG, w/o CoT
LLM Overhead	
CoT analysis	∼15 s/call, 1 call/iter, <2% of wall time
RAG + gen.	∼5 s/call, 1–2 calls/iter, <1%
Quick EDA	∼8 min, ∼85% of wall time
Full EDA	∼45 min, when promoted
Total tokens	∼6.0 M per 6-benchmark suite (74% prompt, 26% completion; 81% DeepSeek-V3.2, 19% Qwen3-235B)

† One thread·hour denotes one EDA worker thread running for one hour; this measures physical-thread occupancy rather than full-CPU-core time.

where $\epsilon > 0$ is a stabilizing constant. This score optimizes expected frontier gain per unit of runtime risk, directing the budget toward cost-effective frontier improvements.

Empirically, the EHVI-guided promotion concentrates full-evaluation budget on strong quick-stage candidates: across the reported benchmark set, the fraction of promoted candidates that rank in the historical top-20 of quick-stage evaluations averages 0.96 ± 0.08 ($p < 0.001$ vs. the 0.5 null, paired t-test), confirming that promotion functions as a quality filter. Ablation confirms this mechanism's importance: removing it reduces the top-region hit rate from 34.29% to 8.57% (Section 5.2).

4 Experimental Setup

This section describes the benchmarks, baselines, and evaluation protocol.

All methods share the same execution stack, quick-to-full protocol, and runtime accounting; only candidate proposal strategies differ. LLM calls are made only when the candidate buffer is depleted (typically every 4–8 iterations), not at every iteration. Sensitivity and failure analyses are pre-computed into compact text before prompt injection. Consequently, token growth is tied to buffer-refill points rather than every evaluated trajectory. Relative to CROP-style retrieval-guided prompting, the main difference is where experience is kept: StateTune accumulates validated summaries in persistent memory and refreshes prompt context only at buffer-refill points, instead of depending as heavily on reassembled online context throughout the search.

Since neither RankTuner [21] nor CROP [14] releases source code targeting our flow, we re-implemented both within our evaluation framework—RankTuner retaining its pairwise GP and Duel-Thompson sampling, CROP using the same RAG-EDA [15]

Table 5: Tuned parameters. C=continuous; Cat=categorical; B=binary.

Stg	Parameter	Type	Range
FP	CORE_UTIL	C	[0.55, 0.78]
	ASPECT_RATIO	C	[0.80, 1.20]
	CORE_MARGIN	Int	[0, 16]
PL	PLACE_CONG_EFFORT	Cat	low/med/high
	PLACE_TIMING_EFFORT	Cat	med/high
	PLACE_UNIFORM_DENSITY	Cat	true/false
	PLACE_IO_PINS_AWARE	Cat	true/false
	TD_PLACE	B	{0, 1}
	PLACE_MAX_DENSITY	C	[0.75, 0.95]
CTS	PRECTS_SETUP_OPT	B	{0, 1}
	PRECTS_HOLD_OPT	B	{0, 1}
	DRV_EFFORT	Cat	low/med/high
	RUN_CTS	B	{0, 1}
	CTS_TARGET_SKEW	C	[0.03, 0.15]
	CTS_POST_OPT	B	{0, 1}
RT	ROUTE_TIMING_DRIVEN	Cat	true/false
	ROUTE_SI_DRIVEN	Cat	true/false
	ROUTE_EFFORT	Cat	low/med/high
	ROUTE_LAYER_EFFORT	Cat	low/med/high

retrieval as StateTune—so that all methods share identical execution, promotion, and budget-accounting pipelines.

Quick-stage endpoint. The quick-stage cutoff is a configurable parameter. We use CTS as the quick-stage endpoint rather than placement alone, because CTS provides richer timing signals that improve the fidelity of quick-stage QoR estimates and enable more effective promotion decisions.

Evaluation metrics. Best WNS, area, and power are the per-objective extremes observed during the run. *Final hypervolume* (HV) is the dominated hypervolume of the Pareto frontier at run termination, computed with respect to a fixed reference point $\mathbf{r}$ that is dominated by all feasible evaluations; it serves as a comprehensive frontier metric [7]. The reference point is determined independently for each comparison group (main comparison, ablation, and sensitivity studies), so absolute HV values are not directly comparable across tables. *Global Pareto hits* (GP Hits) counts the number of a method's full-valid evaluations that are non-dominated with respect to the union of all methods' evaluations on the same benchmark. In the main comparison, we focus the narrative on WNS, area, power, and HV; GP hits are used as auxiliary evidence, while time to first global Pareto hit (First Hit) is retained in the ablation analysis.

5 Results

This section reports the unified comparison across the benchmark matrix, the ablation study, and dedicated analyses of evidence-gating sensitivity and memory poisoning.

5.1 Comparison Across the Working Benchmark Matrix

Table 6 reports the six benchmark blocks in the full matrix. We focus on aggregate trends across the full set rather than on any single case.

Best per-objective results and frontier quality. StateTune attains the best WNS, best power, and best final HV on all six benchmark blocks. For area, it is the outright best method on ASAP-JPEG, ASAP-IBEX, and NAN-AES, and ties the best value on ASAP-AES, NAN-JPEG, and NAN-IBEX. Most importantly, the final-HV advantage is consistent across the full benchmark matrix: +33.7% over Optuna-TPE on ASAP-JPEG, +3.3% over Optuna-TPE on ASAP-AES, +1.2% over Optuna-TPE on ASAP-IBEX, +0.7% over Optuna-TPE on NAN-JPEG, +1.3% over BO (qEHVI) on NAN-AES, and +1.7% over BO (qEHVI) on NAN-IBEX. Thus, StateTune consistently produces a stronger final Pareto frontier across the full matrix.

Auxiliary evidence from GP hits. StateTune also leads GP hits on ASAP-JPEG, ASAP-AES, NAN-JPEG, and NAN-AES, and ties the best GP-hit count on ASAP-IBEX. The GP-hit advantage on these five blocks can be traced to the persistent memory mechanism: accumulated failure patterns and sensitivity signals steer the search away from known-poor regions, concentrating evaluation budget near the Pareto frontier rather than re-exploring dominated configurations. This separation is useful: on NAN-IBEX, BO attains more GP hits, yet StateTune still delivers the best final HV and the best power, showing that frontier quality is not determined by hit count alone.

LLM baselines do not automatically win. CROP—despite using an LLM with RAG—remains behind StateTune on all six blocks, and even trails random search on ASAP-JPEG and NAN-IBEX in final HV. Adding an LLM and retrieval to a tuning loop is not by itself sufficient. Without evidence gating, noisy LLM suggestions enter the search context unfiltered, progressively biasing the proposal distribution; without a surrogate-guided promotion policy, the system cannot distinguish high-frontier-gain candidates from plausible-looking but dominated ones, wasting the scarce full-evaluation budget. RankTuner (ICCAD'24) [21] also trails on every block: its preference-based ranking struggles in this mixed continuous-categorical space with 19 parameters, the pairwise GP scales poorly with dimensionality, and without cross-stage knowledge transfer or persistent memory, each run starts from scratch.

Reproducibility. To assess run-to-run stability, we repeat StateTune three times on every benchmark block under identical budgets. Table 7 reports the HV mean and standard deviation; Figure 4 visualizes the anytime trajectory for ASAP7-AES. The coefficient of variation (CV) remains below 7% on five of six blocks (e.g., 0.89% on ASAP7-AES and 0.51% on NAN-JPEG), confirming reproducibility across seeds. ASAP7-JPEG exhibits the highest CV (15.4%), but even its worst single-seed HV ($\approx 4.5 \times 10^5$) still exceeds all baselines in Table 6. In all six cases, the repeated-run mean matches or exceeds the single-run HV in Table 6, indicating that the main comparison is not an outlier.

5.2 Ablation Study

Each ablation variant removes one component while keeping the rest intact (Table 8).

w/o Persistence. Persistent memory propagates accumulated failure patterns, constraints, and sensitivity signals across iterations. Without it, HV drops to 4.19 M (−58.5%) with zero Pareto hits—despite retaining LLM, RAG, CoT, and EHVI. This confirms that persistent memory is the largest single contributor and a distinct design axis from LLM-based search.

w/o KA. The knowledge agent translates memory into context-aware proposals conditioned on failures, sensitivities, and the design prior. Without it, HV falls to 4.57 M (−54.8%) and only two

Table 6: Results across the 2×3 benchmark matrix under the shared budget caps. Six methods are compared: BO (qEHVI), Optuna-TPE, Random, RankTuner (ICCAD'24) [21], CROP (ICCAD'25) [14], and StateTune. Bold = best per benchmark (per column).

ASAP Benchmarks							NAN Benchmarks						
Design	Method	WNS	Area	Power	GP Hits	HV	Design	Method	WNS	Area	Power	GP Hits	HV
ASAP-JPEG	Random	-26.1420	6042	29.9512	0	1.806e5	NAN-JPEG	Random	-0.0080	106638	111.9460	0	1.911e6
	Optuna-TPE	-13.3730	5973	29.9231	2	3.412e5		Optuna-TPE	0.0130	106204	111.3026	1	2.082e6
	BO (qEHVI)	-26.1130	6025	29.7187	1	2.348e5		BO (qEHVI)	-0.0050	**106163**	111.6088	0	2.064e6
	RankTuner	-29.5560	6256	33.7636	0	9.938e4		RankTuner	0.0090	108508	126.8801	0	7.849e5
	CROP	-10.8240	6924	33.5890	1	1.286e5		CROP	0.0120	106224	111.6372	0	2.030e6
	StateTune	**-6.7520**	**5946**	**29.4496**	**12**	**4.564e5**		**StateTune**	**0.0150**	**106163**	**111.2779**	**10**	**2.097e6**
ASAP-AES	Random	-170.7360	1678	7.9457	0	3.178e6	NAN-AES	Random	-0.0230	22371	21.6310	1	3.887e3
	Optuna-TPE	-174.7420	1670	7.7038	5	3.436e6		Optuna-TPE	-0.0190	22379	21.5630	1	4.072e3
	BO (qEHVI)	-201.8410	1670	7.7462	0	3.284e6		BO (qEHVI)	-0.0180	22355	21.5925	4	4.120e3
	RankTuner	-175.3810	1672	7.9520	0	3.233e6		RankTuner	-0.0650	22648	21.8425	0	2.357e3
	CROP	-164.6620	1702	8.6772	0	2.633e6		CROP	-0.0250	22433	21.6294	0	3.813e3
	StateTune	**-162.8580**	**1669**	**7.5967**	**10**	**3.548e6**		**StateTune**	**-0.0120**	**22353**	**21.4920**	**9**	**4.316e3**
ASAP-IBEX	Random	-353.5490	2063	6.5417	0	2.747e5	NAN-IBEX	Random	0.0240	29030	11.6956	0	1.288e3
	Optuna-TPE	-258.6870	2052	6.2220	**6**	4.486e5		Optuna-TPE	0.0370	28943	11.6197	0	1.515e3
	BO (qEHVI)	-335.1390	2181	7.1288	0	1.061e5		BO (qEHVI)	0.0590	**28928**	11.5741	**9**	1.780e3
	RankTuner	-274.0760	2061	6.2730	0	3.947e5		RankTuner	0.0130	29068	11.7663	0	1.147e3
	CROP	-270.8550	2063	6.5617	0	3.239e5		CROP	0.0120	29377	11.5660	0	1.113e3
	StateTune	**-246.9210**	**2050**	**6.2107**	6	**4.539e5**		**StateTune**	**0.0600**	**28928**	**11.5597**	5	**1.810e3**

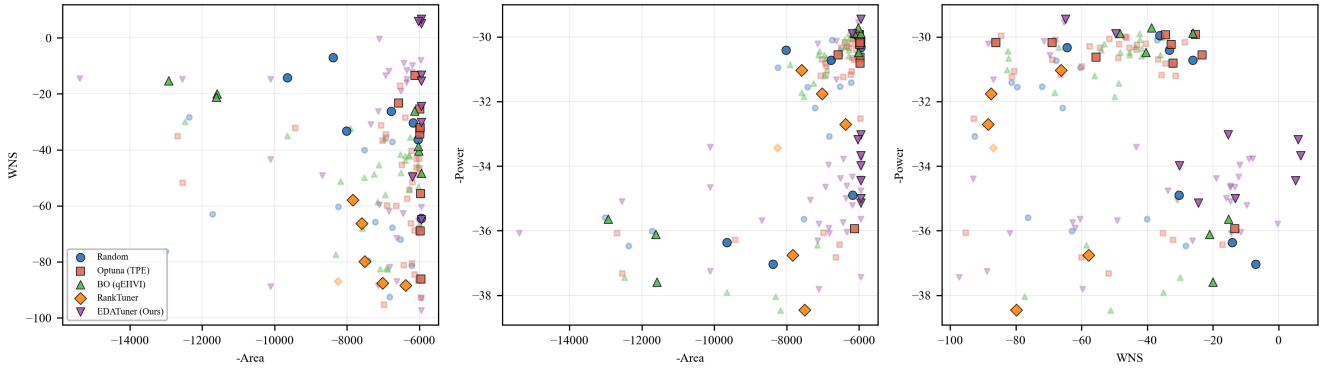


Figure 3: Pareto frontier projections for the ASAP-JPEG case (−Area vs. WNS, −Area vs. −Power, WNS vs. −Power). StateTune (purple triangles) occupies the strongest region among the compared methods.

Table 7: StateTune reproducibility over three independent runs.

ASAP7			NAN		
Benchmark	HV ($\mu \pm \sigma$)	CV (%)	Benchmark	HV ($\mu \pm \sigma$)	CV (%)
JPEG	$5.33e5 \pm 8.21e4$	15.4	JPEG	$2.53e6 \pm 1.29e4$	0.51
AES	$3.73e6 \pm 3.32e4$	0.89	AES	$4.42e3 \pm 6.34e1$	1.43
IBEX	$4.57e5 \pm 2.86e4$	6.27	IBEX	$1.66e3 \pm 9.00e1$	5.42

Table 8: Ablation on ASAP7 JPEG under 48 thread·h budget. Variants ordered by degradation severity.

Variant	Final HV (M)	HV Drop (%)	Pareto Hits	First Hit (h)
Full system	10.10	–	6	10.29
w/o Persistence	4.19	58.5	0	–
w/o KA	4.57	54.8	2	2.10
w/o RAG	5.44	46.1	0	–
w/o CoT	6.92	31.5	1	29.62
w/o EHVI	8.32	17.6	4	12.93

Pareto hits remain. Together with the persistence ablation, this confirms that both memory and LLM guidance are necessary—neither alone suffices.

w/o RAG. RAG grounds proposals in EDA tool documentation and design-specific best practices; without it, CoT, memory, and EHVI remain intact but the LLM relies solely on parametric knowledge. HV drops to 5.44 M (−46.1%) with zero Pareto hits, confirming that general LLM knowledge is insufficient without retrieval-augmented domain grounding.

w/o CoT. CoT reasoning structures failure diagnosis and sensitivity assessment into analytical traces that inform proposals. Without it, HV drops to 6.92 M (−31.5%), confirming that RAG supplies domain context but CoT supplies the reasoning machinery to act on it.

w/o EHVI. EHVI-guided promotion ranks candidates by expected frontier gain per unit of runtime cost. Without it, the first global Pareto hit is delayed from 10.29 h to 12.93 h and HV falls to

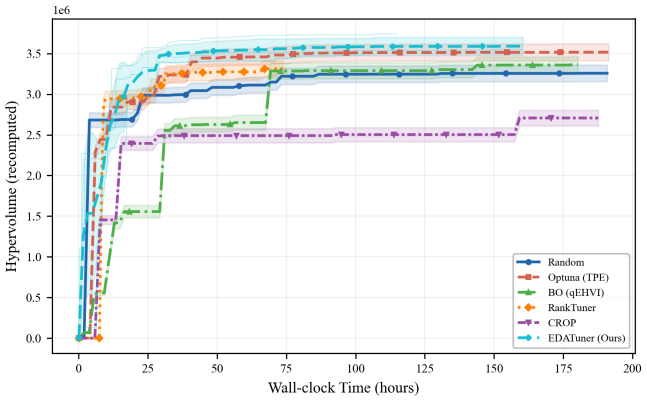
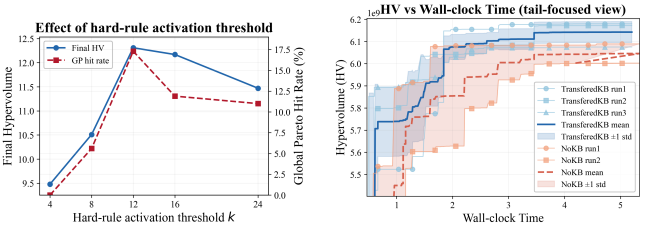


Figure 4: Anytime hypervolume trajectory for ASAP7-AES, averaged over three independent runs per method with shaded ±1 standard-deviation bands. StateTune (cyan) separates from all baselines early and maintains a clear advantage throughout the run.



(a) Evidence-gating sensitivity. **(b) Cross-design transfer.**

Figure 5: Additional validation on ASAP7 JPEG under 48 thread·h. (a) Final HV and global Pareto hit rate peak at k=12. (b) Transferred memory converges faster and reaches higher final HV than no transfer. Thin curves denote individual runs, thick curves denote means, and shaded bands indicate ±1 standard deviation.

8.32 M (−17.6%), confirming that unguided promotion wastes the scarce full-evaluation budget on dominated proposals.

Overall, persistent memory is the primary contributor, while EHVI-guided promotion provides a complementary gain.

5.3 Sensitivity to Evidence-Gating Threshold k

The evidence-gating threshold k controls how many corroborating full-valid observations are required before a hard rule becomes active. To experimentally validate the choice of k=12, we sweep $k \in \{4, 8, 12, 16, 24\}$ on ASAP7 JPEG under the 48 thread·h ablation budget (Figure 5a).

HV rises steeply from 9.48 M at k=4 to a peak of 12.31 M at k=12 (Pareto hit rate 17.3%), then declines to 11.47 M at k=24. This non-monotonic pattern confirms that k=12 balances premature activation of incorrect rules against delayed activation of useful rules.

5.4 Memory Poisoning

To test whether ungated memory accumulates errors, we compare *Pure* (default evidence gating) and *Poisoned* (all LLM-emitted rules admitted without validation) on ASAP7 JPEG under a 48 thread·h budget.

The Poisoned variant suffers substantial degradation: full-valid rate drops from 5.42% to 2.31%, and the validated-rule rate falls from 30.4% to 17.4% while the contradicted-rule rate rises from 0% to 17.4%. These results confirm that persistent memory without evidence gating accumulates incorrect rules and materially harms search quality, validating the gating mechanism as a necessary component rather than an optional filter.

5.5 Cross-Design Knowledge Transfer

Because the persistent memory stores typed, design-agnostic artifacts (e.g., parameter constraints, sensitivity patterns, and failure modes), it can be transferred across designs. We test this by aggregating validated memory from the four non-JPEG benchmark blocks, ranking artifacts by a transfer-confidence score, and using the retained top-k artifacts to initialize a new ASAP7-JPEG run under the same 48 thread·h budget.

The effect is consistent across both seeds (Figure 5b). Transferred runs enter the $6.10–6.12 \times 10^9$ HV band and remain there, whereas no-transfer runs plateau lower at about 5.95×10^9 and 6.03×10^9. The transferred mean also reaches 6.0×10^9 earlier, at around 1.8–1.9 hours, while the no-transfer mean stays below that level until near termination. By the end of the run, transfer improves mean HV from about 5.99×10^9 to 6.12×10^9, an absolute gain of roughly 1.3×10^8. Overall, transferred memory improves both convergence speed and final-HV stability across repeated runs.

5.6 Limitations

The six-block matrix spans two technology nodes and JPEG/AES/IBEX, showing consistent cross-case gains; broader benchmarks and industrial SoC studies would further strengthen external validity. The flat memory structure could be extended to hierarchical or graph-based organizations [11, 22], which may help when memory grows to thousands of rules across many designs.

6 Conclusion

StateTune formulates LLM-assisted EDA flow tuning as a closed-loop, state-carrying process in which a typed, evidence-gated persistent memory jointly supports candidate generation and EHVI-guided promotion. Across six benchmark blocks and five baselines, it achieves the best final hypervolume on every block. Ablation shows that persistent memory is the primary contributor (−58.5% HV when removed), while EHVI promotion provides a secondary gain (−17.6%). Evidence gating, cross-design transfer, and reproducibility experiments further validate the proposed memory design. Future work includes evaluation on larger industrial designs and richer memory organizations for scaling across more benchmark families.

Acknowledgments

This work is supported by National Science and Technology Major Project (2021ZD0114701).

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2623–2631. doi:10.1145/3292500.3330701

[2] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for Hyper-Parameter Optimization. In *Advances in Neural Information Processing Systems 24 (NeurIPS 2011)*. 2546–2554.

[3] Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song, and Bo Li. 2024. AgentPoison: Red-teaming LLM Agents via Poisoning Memory or Knowledge Bases. In *Advances in Neural Information Processing Systems*. https://proceedings.neurips.cc/paper_files/paper/2024/hash/eb113910e9c3f6242541c1652e30dfd6-Abstract-Conference.html

[4] Lawrence T. Clark, Vinay Vashishtha, Lucian Shifren, Aditya Gujja, Saurabh Sinha, Brian Cline, Chandarasekaran Ramamurthy, and Greg Yeric. 2016. ASAP7: A 7-nm finFET Predictive Process Design Kit. *Microelectronics Journal* 53 (2016), 105–115.

[5] Samuel Daulton, Maximilian Balandat, and Eytan Bakshy. 2020. Differentiable Expected Hypervolume Improvement for Parallel Multi-Objective Bayesian Optimization. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*. 9851–9864.

[6] Stefan Falkner, Aaron Klein, and Frank Hutter. 2018. BOHB: Robust and Efficient Hyperparameter Optimization at Scale. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*. 1436–1445.

[7] Hao Geng, Tinghuan Chen, Yuzhe Ma, Binwu Zhu, and Bei Yu. 2023. PTPT: Physical Design Tool Parameter Tuning via Multi-Objective Bayesian Optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 42, 1 (2023), 178–189. doi:10.1109/TCAD.2022.3167858

[8] Amur Ghose, Andrew B. Kahng, Sayak Kundu, and Zhiang Wang. 2025. ORFS-agent: Tool-Using Agents for Chip Design Optimization. In *Proceedings of the 7th ACM/IEEE Symposium on Machine Learning for CAD (MLCAD)*. 1–13.

[9] Zhuolun He, Yuan Pu, Haoyuan Wu, Tairu Qiu, and Bei Yu. 2025. Large Language Models for EDA: Future or Mirage? *ACM Transactions on Design Automation of Electronic Systems* 30, 6 (2025), 90:1–90:53. doi:10.1145/3736167

[10] Jinwook Jung, Andrew B. Kahng, Seungwon Kim, and Ravi Varadarajan. 2021. METRICS2.1 and Flow Tuning in the IEEE CEDA Robust Design Flow and OpenROAD ICCAD Special Session Paper. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 1–9. doi:10.1109/ICCAD51958.2021.9643541

[11] Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. 2025. Memory OS of AI Agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 25961–25970. doi:10.18653/v1/2025.emnlp-main.1318

[12] Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. 2017. Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*. 528–536.

[13] Rongjian Liang, Jinwook Jung, Hua Xiang, Lakshmi N. Reddy, Alexey Lvov, Jiang Hu, and Gi-Joon Nam. 2021. FlowTuner: A Multi-Stage EDA Flow Tuner Exploiting Parameter Knowledge Transfer. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 1–9. doi:10.1109/ICCAD51958.2021.9643564

[14] Jingyu Pan, Isaac Jacobson, Zheng Zhao, Tung-Chieh Chen, Guanglei Zhou, Chen-Chia Chang, Vineet Rashingkar, and Yiran Chen. 2025. CROP: Circuit Retrieval and Optimization with Parameter Guidance using LLMs. *CoRR* abs/2507.02128 (2025). arXiv:2507.02128 https://arxiv.org/abs/2507.02128

[15] Yuan Pu, Zhuolun He, Yuqi Jiang, Tairu Qiu, Haoyuan Wu, Qi Sun, Cheng Zhuo, and Bei Yu. 2025. Customized Retrieval Augmented Generation and Benchmarking for EDA Tool Documentation QA. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 44, 12 (2025), 4615–4628. doi:10.1109/TCAD.2025.3568776

[16] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. 2016. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proc. IEEE* 104, 1 (2016), 148–175. doi:10.1109/JPROC.2015.2494218

[17] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language Agents with Verbal Reinforcement Learning. In *Advances in Neural Information Processing Systems*, Vol. 36. Curran Associates, Inc., 8634–8652. https://proceedings.neurips.cc/paper_files/paper/2023/file/1b44b878bb782e6954cd888628510e90-Paper-Conference.pdf

[18] James E. Stine, Ivan Castellanos, Michael Wood, Jeff Henson, Fred Love, W. Rhett Davis, Paul D. Franzon, Michael Bucher, Sudarshan Basavarajaiah, Julie Oh, and Raphael Jennings. 2007. FreePDK: An Open-Source Variation-Aware Design Kit. In *Proc. IEEE Int. Conf. Microelectronic Systems Education (MSE)*. 173–174.

[19] Runzhi Wang, Jingyu Pan, Yiran Chen, and Jiang Hu. 2026. AstroTune: AST-Assisted LLM Retrieval for Cross-Stage Design Flow Parameter Tuner. In *Proceedings of the 2026 International Symposium on Physical Design (ISPD)*. ACM, 144–152. doi:10.1145/3764386.3779579

[20] Haoyuan Wu, Zhuolun He, Xinyun Zhang, Xufeng Yao, Su Zheng, Haisheng Zheng, and Bei Yu. 2024. ChatEDA: A Large Language Model Powered Autonomous Agent for EDA. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 43, 10 (2024), 3184–3197. doi:10.1109/TCAD.2024.3383347

[21] Peng Xu, Su Zheng, Yuyang Ye, Chen Bai, Siyuan Xu, Hao Geng, Tsung-Yi Ho, and Bei Yu. 2024. RankTuner: When Design Tool Parameter Tuning Meets Preference Bayesian Optimization. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 122:1–122:7. doi:10.1145/3676536.3676782

[22] Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025. A-MEM: Agentic Memory for LLM Agents. *CoRR* abs/2502.12110 (2025). arXiv:2502.12110 https://arxiv.org/abs/2502.12110

[23] Wei Yang, Jinwei Xiao, Hongming Zhang, Qingyang Zhang, Yanna Wang, and Bo Xu. 2025. Coarse-to-Fine Grounded Memory for LLM Agent Planning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 13029–13056. doi:10.18653/v1/2025.emnlp-main.659

[24] Ruiqi Zhao, Yiqun Liu, Liang Pang, Xiao Zhang, and Min Zhang. 2025. Retrieve, Schedule, Reflect: LLM Agents with Adaptive Memory Management. *CoRR* abs/2503.09476 (2025). arXiv:2503.09476 https://arxiv.org/abs/2503.09476

[25] Su Zheng, Hao Geng, Chen Bai, Bei Yu, and Martin D. F. Wong. 2023. Boosting VLSI Design Flow Parameter Tuning with Random Embedding and Multi-objective Trust-region Bayesian Optimization. *ACM Transactions on Design Automation of Electronic Systems* 28, 5 (2023), 74:1–74:23. doi:10.1145/3597931

[26] Ruizhe Zhong, Xingbo Du, Shixiong Kai, Zhentao Tang, Siyuan Xu, Hui-Ling Zhen, Jianye Hao, Qiang Xu, Mingxuan Yuan, and Junchi Yan. 2024. LLM4EDA: Emerging Progress in Large Language Models for Electronic Design Automation. *CoRR* abs/2401.12224 (2024). arXiv:2401.12224 doi:10.48550/ARXIV.2401.12224