AutoCircuit: Agentic Pareto Front Exploration for Analog Circuit Designs

Doyun Kim [1] **Rajath Salegame** [1] **Lars Holdijk** [2] **Jan Ole Ernst** [1]

Abstract

Analog circuit design poses a high-dimensional search problem in which circuit parameters must be jointly optimized with respect to competing performance targets such as gain and power. A range of traditional AI methods have shown promising results on this task, including Bayesian optimization (BO), evolutionary algorithms, and reinforcement learning. More recently, large language models (LLMs) have been proposed as autonomous design agents. However, prior work typically studies these techniques in isolation rather than composing them and often focuses on scalarized performance metrics, which can lead to limited design diversity. To address this, we present **AutoCircuit**, a system with an LLM orchestrator that leverages a set of ML tools to explore the Pareto front in a multi-objective space within a fixed wall-clock budget. We evaluate our system on a representative analog benchmark, a two-stage op-amp circuit implemented in the SKY130 open PDK, and show that LLM orchestration with effective tool use improves log-hypervolume by up to $2.5\times$ over the LLM-only baseline. We further show that the best result comes from a carefully chosen subset of tools (NSGA-II + g_m/I_d) rather than the full set of tools.

1. Introduction

Analog circuit optimization is one of the most challenging tasks in the field of circuit design. Analog circuits consist of tens to thousands of transistors and passive devices. The task requires searching physical parameters of the devices to make the circuit functional and optimized with respect to performance metrics such as gain, bandwidth, and power. This relies heavily on engineering experience and insight. Many optimization algorithms have been adopted to this task. Classical black-box methods such as multi-objective Bayesian optimization with Gaussian-process surrogates, as implemented in modern frameworks such as BoTorch (Balandat et al., 2020), and evolutionary algorithms led by NSGA-II (Deb et al., 2002) have shown strong performance in this domain. Machine learning-based approaches also have gained traction: reinforcement learning has been trained to size analog circuits along with different techniques, starting from AutoCkt (Settaluri et al., 2020), utilizing g_m/I_d information (Chiari et al., 2026), and extending to Pareto- and PVT-aware goal-conditioned policies (Kim et al., 2025). More recent works have introduced large language models (LLM) in the design loop: fine-tuning domain-specific LLMs for op-amp sizing (Chen et al., 2024), coupling in-context LLM proposers with Bayesian optimization (Yin et al., 2024), and orchestrating agentic LLM workflows (Ahmadzadeh et al., 2025; Yu et al., 2026). However, these systems target single design points specified by the user, and report sample efficiency or success rate against those targets.

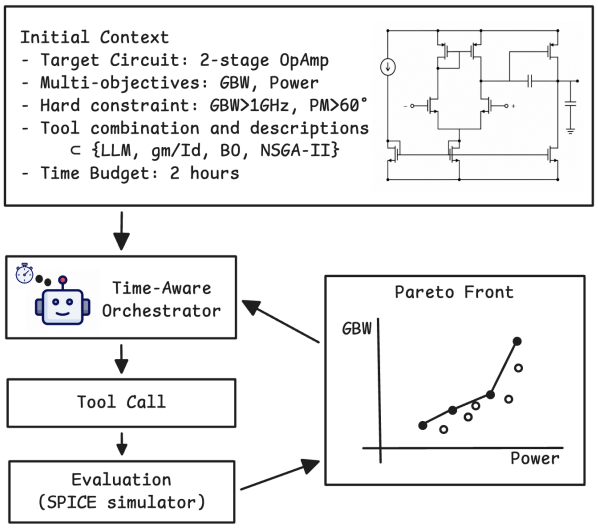

Figure 1. Overall workflow of AutoCircuit

Recent work on Auto Research has shown that large language models (LLMs) can drive the experimental loop of neural network training, in which an agent observes the outcome of each training run and proposes the next hyperparameters (Karpathy, 2026). This loop structure is not unique to neural networks. Many engineering design tasks

[1]Normal Computing [2]University of Oxford. Correspondence to: Doyun Kim <doyun@normalcomputing.com>.

Proceedings of the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

have the same shape, where parameters are iteratively refined against a metric. One such example is analog circuit design where schematic optimization is precisely such a loop in which device sizes are chosen to meet performance specifications. However, the analog setting differs from neural network training in ways that make the loop harder to close.

Two properties of analog circuit optimization distinguish it from neural network hyperparameter sweeps. First, it is genuinely multi-objective: a useful design must satisfy multiple performance metrics such as gain, bandwidth, power, and area, which trade off against one another. The Pareto front is useful for engineers rather than having a single best point since an analog circuit block will be a part of a larger system whose power and performance budget must be shared. Second, large regions of the parameter space produce circuits that do not operate at all. The optimizer therefore receives no useful signal from much of the design space, and cold-start runs cannot rely on prior knowledge of what level of performance is reachable.

To combine the high-level decision-making of large language models with proven conventional optimization algorithms, we present AutoCircuit, an LLM-orchestrated optimization system in which an LLM observes the outcome of each tool call and chooses next tool call among Bayesian optimization (BO), NSGA-II and g_m/I_d information under a given time budget. We evaluate 8 different sets of tools and compare with 2 black-box optimization algorithm baselines on a two-stage operational amplifier (2-stage OpAmp) in the skywater 130nm PDK. This paper makes the following contributions:

- Our system explores Pareto fronts with the use of multiple tools. Instead of asking the LLM to meet a fixed specification, we task it with expanding the GBW–Power Pareto front, which gives downstream designers a continuum of trade-offs rather than a single point.
- Choosing the correct tool sub-set improves performance over a baseline with all tools. The right tool set improves log-HV by $\sim 2.5\times$ over the LLM-only baseline for the same time budget, while providing every available tool does not yield the best outcome.
- Time-aware prompting is necessary but not sufficient. Although we expose the wall-clock budget to the LLM, the orchestrator still exhibits sub-optimal tool-calling behavior.

2. AutoCircuit

2.1. General Workflow

Figure 1 shows workflow of AutoCircuit. It starts each experiment with initial context which includes a target circuit, a time budget, and an allowed tool set. Each tool carries a detailed description to support reasonable LLM decisions. During iterations, an LLM orchestrator manages tool calls, runtime and the current Pareto front. Based on the current Pareto front and remaining time budget, the LLM decides the next tool call. When a tool returns new circuit parameters, the orchestrator runs a SPICE simulation to evaluate and record the circuit's performance. The system iterates until the time budget is exhausted.

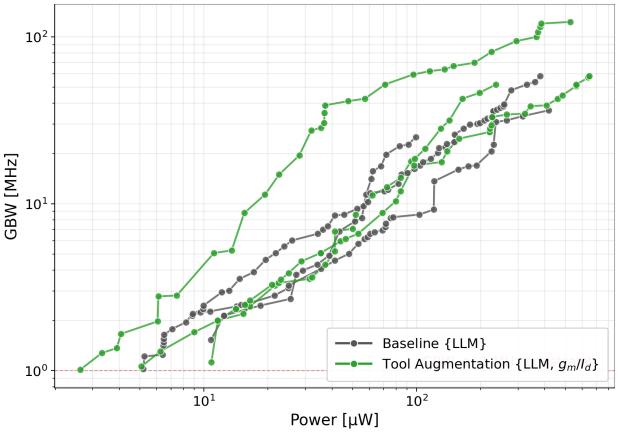

Figure 2. Pareto front Comparison between baseline and g_m/I_d (3 rounds)

2.2. Tools

The LLM is given a configurable subset of three optimization tools, and an always-available manual evaluation using a SPICE simulator. Each tool is exposed as a self-contained black box with a stable command-line interface and provides a detailed description so the orchestrator can invoke any tool in a given state.

Bayesian optimization (BO). BO learns a probabilistic model of how the simulator responds to different parameter choices, and uses that model to suggest the next point most likely to improve the Pareto front. We use the qLogNEHVI multi-objective acquisition function (Daulton et al., 2021) implemented in BoTorch (Balandat et al., 2020), which incorporates the hard constraints (GBW ≥ 1 MHz, phase margin $\geq 45°$) directly into its scoring. BO is inherently sequential because each suggestion uses every prior evaluation, so its quality grows with the number of samples already collected — a longer time budget therefore favors BO over methods that operate in parallel.

NSGA-II. NSGA-II (Deb et al., 2002) maintains a population of candidate parameter sets and improves it across generations through simulated binary crossover and polynomial mutation, ranking individuals by non-dominated sorting and crowding distance. A single call runs an entire $pop \times gen$ evolution and returns the resulting Pareto-improved front.

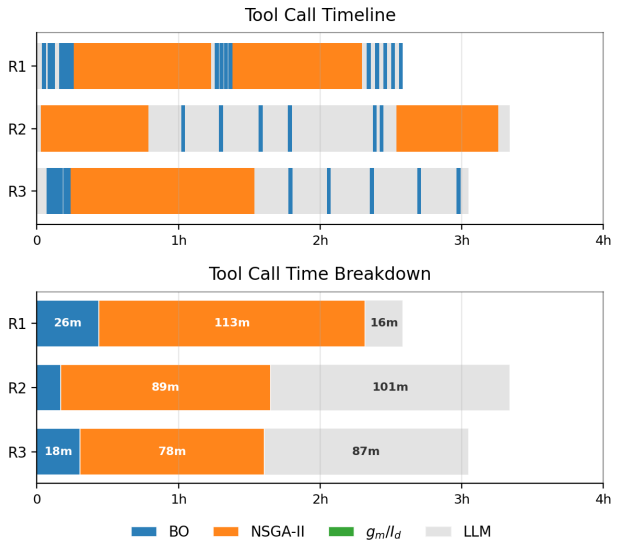

Figure 3. Tool Call Timeline and Time Breakdown for runs with all available tools. Note that LLM includes API response time

We use a custom implementation to keep each workspace dependency-free and to expose a stable CLI to the LLM. The optimizer reads the existing experiment history once at initialization and then evolves internally; information arriving between calls is not absorbed until the agent explicitly relaunches the search. The algorithm is inherently parallel since it evaluates an entire generation simultaneously. This makes NSGA-II suitable for quickly forming an initial Pareto front with short or moderate budgets.

g_m/I_d recommender. Unlike BO and NSGA-II, the g_m/I_d tool is a recommender rather than an optimizer: it does not produce new evaluations. For the given technology, we first characterize each device type with a one-time (V_{gs}, V_{ds}, W, L) sweep to build a g_m/I_d lookup table. At each invocation, the tool analyzes each transistor's operating region (weak, moderate, or strong inversion) by g_m/I_d value for each. Then, it offers suggestions for new parameters to move the operating region based on standard analog-design heuristics, mimicking the manual design process. The tool is therefore *insight-based* rather than *search-based*, and is naturally complementary to BO and NSGA-II. Importantly, the g_m/I_d characterization can capture device-level physical behavior that the LLM may not have seen during pretraining, so the tool injects physics-informed information that the agent could not otherwise infer.

2.3. Time Budget

A defining characteristic of our setup is that the LLM agent manages its own time. Each run is governed by a two-tier timer: a *soft* budget provided to the agent in its prompt drives the LLM's pacing while a *hard* budget at $1.25\times$ the soft budget is enforced by an external wall-clock watchdog as a safety net. Based on this time budget, the LLM orchestrator needs to invoke tool calls strategically to obtain the best Pareto front.

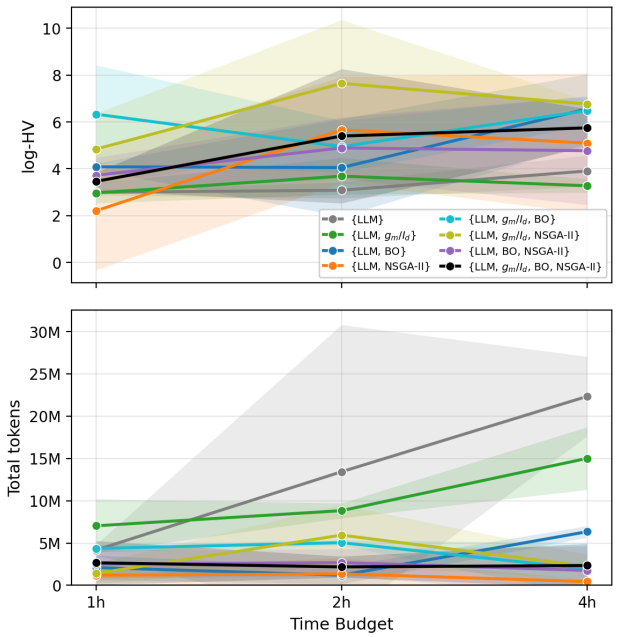

Figure 4. Performance and Token Usage for different time budget

3. Evaluation

3.1. Experimental setup

We evaluate AutoCircuit on two-stage operational amplifier in the SkyWater 130nm open-source PDK (SkyWater Technology Foundry et al., 2020) as shown in Figure 1. The design space consists of 12 device parameters: the width and length of the transistors and the values of the bias current and a coupling capacitor. The circuit is evaluated by `NgSpice` simulator (Vogt et al., 2024). Unlike many analog circuit optimization works, our framework aims to optimize multi objectives: Gain-Bandwidth product (GBW) and power. Also, we impose hard constraints such as $GBW \geq 1\,\text{MHz}$ and $\text{Phase Margin} \geq 60°$ to ensure usefulness of Pareto front samples. AutoCircuit reports the Pareto front in the (Power, GBW) plane with Power as the objective to minimize and log hypervolume (log-HV) of the Pareto front. We use Claude Opus 4.7 as our LLM orchestrator.

3.2. Results

Pareto Front Exploration We investigate the efficacy of tool use in our framework. As a baseline, we use our framework without any tools, meaning the LLM orchestrator itself also operates as an optimizer (LLM). Based on evaluation history, it chooses next parameters based on its reasoning. Fig. 2 shows how an additional tool, the g_m/I_d tool, improves Pareto front exploration. The LLM model is

Table 1. Per-tool-set performance at the 2-hour budget. The rightmost block reports the number of Pareto-contributing evaluations over the total number of times each tool was invoked.

					Pareto contribution / tool calls			
Tool set	log-HV $\mu \pm \sigma$	Pareto #	Evals	Tokens	LLM	g_m/I_d	BO	NSGA-II
{LLM}	3.08 ± 0.34	38.7	233	26.5 M	$55/233$	—	—	—
{LLM, g_m/I_d}	3.69 ± 0.78	32.3	196	17.0 M	—	$42/196$	—	—
{LLM, BO}	4.05 ± 2.08	4.0	127	5.9 M	$0/3$	—	$4/124$	—
{LLM, NSGA-II}	5.66 ± 2.26	22.0	763	5.4 M	—	—	—	$22/763$
{LLM, g_m/I_d, BO}	4.95 ± 1.10	11.3	137	9.4 M	—	$2/24$	$10/113$	—
{LLM, g_m/I_d, NSGA-II}	$\mathbf{7.64 \pm 2.71}$	6.3	413	**11.7 M**	—	$\mathbf{9/69}$	—	$2/344$
{LLM, BO, NSGA-II}	4.89 ± 1.31	8.7	269	7.6 M	$0/1$	—	$2/69$	$<1/199$
{LLM, g_m/I_d, BO, NSGA-II}	5.40 ± 2.87	19.3	380	9.6 M	—	$0/8$	$2/16$	$18/356$
{BO} (No LLM)	8.88 ± 2.14	6.7	200	—	—	—	$7/200$	—
{NSGA-II} (No LLM)	7.60 ± 1.48	8.7	1230	—	—	—	—	$9/1230$

trained with general knowledge about circuit design. Still, it lacks the detailed information about the target circuit in current condition. The g_m/I_d tool instantly provides physics-informed insight on the current circuit parameters and suggests parameter changes to shift each transistor's operating points for better performance. The g_m/I_d Pareto front moves toward the upper left from the baseline Pareto front, meaning the circuit provides better GBW performance with less power consumption.

Performance with different tool sets We evaluate our framework with different sets of tools, which are subset of $\{\text{LLM}, g_m/I_d, \text{BO}, \text{NSGA-II}, \}$. All runs use a two-hour wall-clock time budget. Unlike g_m/I_d, BO and NSGA-II tools take longer run time to obtain new Pareto samples. Therefore, the orchestrator needs to carefully choose tools each turn, taking time budget into account. Table 1 shows a comparison among different tool sets. The results show that {LLM, g_m/I_d, NSGA-II} achieves the highest log-HV. NSGA-II can benefit more from existing Pareto samples than BO since BO needs to take sequential steps to build its surrogate model whereas NSGA-II directly uses the samples for the next population. Therefore, this tool set results in better log-HV than using all tools. Also, g_m/I_d tool contributes to the Pareto front efficiently as mentioned above. Figure 3 shows the timeline and time breakdown for the case with all tools. Different rounds show different tool call patterns due to the LLM's stochasticity, resulting in large performance variation. Note that g_m/I_d is not used at all although it is provided. This illustrates a key failure mode of LLM tool orchestration: providing the agent with a larger tool set can paradoxically cause it to under-use or entirely neglect the tools it most needs.

Additionally, we run BO and NSGA-II without the LLM orchestration for reference. They achieve better log-HV than the orchestrated setting. This is attributed to the small number of parameters in our target circuit. This advantage may diminish when we evaluate larger circuits.

Performance with different time budgets Different time budgets require different strategies for tool calls. In particular, runs with a low time budget should prevent the orchestrator from using tool calls with long run. Figure 4 shows the performance and token usage at different time budgets. Counter-intuitively, performance does not monotonically improve with time budget. A 1-hour time budget is not sufficient to utilize time-taking tools like BO and NSGA-II. Additionally, a larger time budget only leads to more token usage without performance improvement. It indicates that the LLM orchestrator in the current settings cannot fully utilize tools' runtime-dependent characteristics.

4. Conclusion

We present **AutoCircuit**, an LLM-orchestrated multi-objective optimizer that utilizes Bayesian optimization, NSGA-II, and a g_m/I_d recommender as callable tools and applies them to a 12-parameter SKY130 op-amp under realistic op-amp constraints. At the 2-hour budget, the right toolset ({LLM, g_m/I_d, NSGA-II}) achieves a $\sim 2.5\times$ improvement in log-HV over the LLM-only baseline. Counter-intuitively, however, this gain is not maximized by giving the LLM more resources. Adding more tools may cause performance degradation and extending the time budget does not improve performance.

Future work The present results are limited to single circuit design, technology, and model. Extending these will help us to understand the orchestration strategy. Also, improved prompts including more detailed description of each tool's characteristics and example strategies may result in better usage of tool calls.

References

Ahmadzadeh, M., Chen, K., and Gielen, G. AnaFlow: Agentic LLM-based workflow for reasoning-driven explainable and sample-efficient analog circuit sizing. In

Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2025. doi: 10.1109/ICCAD66269.2025.11240818. arXiv:2511.03697.

Balandat, M., Karrer, B., Jiang, D. R., Daulton, S., Letham, B., Wilson, A. G., and Bakshy, E. BoTorch: A framework for efficient Monte-Carlo Bayesian optimization. In *Advances in Neural Information Processing Systems*, 2020.

Chen, Z., Huang, J., Liu, Y., Yang, F., Shang, L., Zhou, D., and Zeng, X. Artisan: Automated operational amplifier design via domain-specific large language model. In *Proceedings of the 61st ACM/IEEE Design Automation Conference (DAC)*, 2024. doi: 10.1145/3649329.3655903.

Chiari, G., Piccoli, M., and Zoni, D. OSIRIS: Bridging analog circuit design and machine learning with scalable dataset generation. *arXiv preprint arXiv:2601.19439*, 2026.

Daulton, S., Balandat, M., and Bakshy, E. Parallel Bayesian optimization of multiple noisy objectives with expected hypervolume improvement. In *Advances in Neural Information Processing Systems*, 2021.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2): 182–197, 2002.

Karpathy, A. autoresearch: AI agents running research on single-GPU nanochat training automatically. `https://github.com/karpathy/autoresearch`, 2026. Accessed: 2026-05-10.

Kim, S. et al. PPAAS: PVT and pareto aware analog sizing via goal-conditioned reinforcement learning. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2025.

Settaluri, K., Haj-Ali, A., Huang, Q., Hakhamaneshi, K., and Nikolić, B. AutoCkt: Deep reinforcement learning of analog circuit designs. In *Proceedings of the Design, Automation & Test in Europe Conference (DATE)*, 2020.

SkyWater Technology Foundry, Google LLC, and Efabless. SkyWater Open Source PDK: SKY130 open source 130 nm CMOS process design kit. `https://github.com/google/skywater-pdk`, 2020. Open-source 130 nm hybrid signal CMOS PDK; documentation at `https://skywater-pdk.readthedocs.io`.

Vogt, H., Hendrix, M., Nenzi, P., and Warning, D. ngspice: Open source mixed-mode, mixed-level circuit simulator. `http://ngspice.sourceforge.net`, 2024. Berkeley SPICE3f5-based open-source circuit simulator; manual at `http://ngspice.sourceforge.net/docs.html`.

Yin, Y., Wang, Y., Xu, B., and Li, P. ADO-LLM: Analog design Bayesian optimization with in-context learning of large language models. In *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–9, 2024. doi: 10.1145/3676536.3676816. arXiv:2406.18770.

Yu, X., Torbunov, D., Mandal, S., and Ren, Y. AutoSizer: Automatic sizing of analog and mixed-signal circuits via large language model (LLM) agents. *arXiv preprint arXiv:2602.02849*, 2026.