Towards Multi-Agent AI-Driven Design Space Exploration for Systolic Array Architectures

Alexander Anokhin Younghyun Cho
Santa Clara University
{aanokhin, younghyun.cho}@scu.edu

***Abstract*—This paper presents an approach to AI-driven hardware design space exploration, specifically to optimize systolic array-based architectures. We present multiple designs for AI-driven hardware optimization. First, we utilize a Large-Language Model (LLM) agent in a feedback loop, where we use the obtained evaluation data as a prior context to guide the generation of the next configuration to evaluate. We extend this into multi-agentic frameworks, where multiple LLM agents can challenge and influence each other, leading to a more robust and sample-efficient search. Our experiments show that AI-agentic approaches can converge to competitive configurations faster than conventional optimization methods such as Bayesian optimization or genetic algorithm. However, our results still show that conventional Bayesian optimization can become more effective than purely AI-driven approaches once the obtained data becomes sufficiently large. Our analysis provides useful preliminary insights into AI-agentic approaches for hardware optimization.**

I. Introduction

Large-Language Models (LLMs) [13] followed by the agentic Artificial Intelligence (AI) paradigms [11] offer several new, potential advantages for hardware Design Space Exploration (DSE) and architecture optimizations. Large AI models trained on extensive literature offer rich prior knowledge to guide DSE processes. AI agents can also interpret and analyze the tabular data collected during a DSE process, while enhancing the portability of tools. This paradigm raises new questions for researchers: can LLM-based tools address hardware DSE problems effectively, discover new, superior hardware configurations, and enable adaptable tools for current and next-generation hardware architectures?

Toward this new AI-driven DSE paradigm, this paper explores the applicability of LLM-agentic approaches for systolic array architecture optimizations, and we compare with widely-used search-based techniques such as Gaussian Process-based Bayesian optimization (GPBO) and genetic algorithm (GA). We present multiple DSE pipelines to leverage LLM agents to systematically automate and enhance hardware optimization. Our first pipeline, *LLM-in-the-loop*, utilizes an LLM agent in a feedback loop, using prior context to guide the generation of the next candidate. However, this approach risks converging to local optima from the LLM's prior knowledge. Our second pipeline, *multi-LLM-agents*, uses two or more agents, where they challenge and influence each other. This can encourage broader exploration to potentially overcome the limitations of a single agent.

Our DSE pipelines are evaluated with several scenarios for optimizing systolic arrays in terms of their Energy-Delay Product (EDP) or Energy-Delay-Area Product (EDAP), from individual DNN models to multiple DNNs, to per-layer dataflow optimization, using a systolic array simulator SCALE-Sim [8], [10] and an energy/power analysis tool Accelergy [14]. Our results show that LLM-based DSE can converge to competitive configurations significantly faster than GPBO or GA, while demonstrating workload-aware reasoning such as aligning array dimensions to workload structure. On the more challenging per-layer dataflow optimization targeting reconfigurable architectures, prompt engineering techniques such as temperature annealing and multi-agent explore-exploit architectures substantially improve search quality. On the other hand, when the number of collected sample data from simulations becomes sufficiently large, GPBO starts to perform more effectively than purely AI-based approaches.

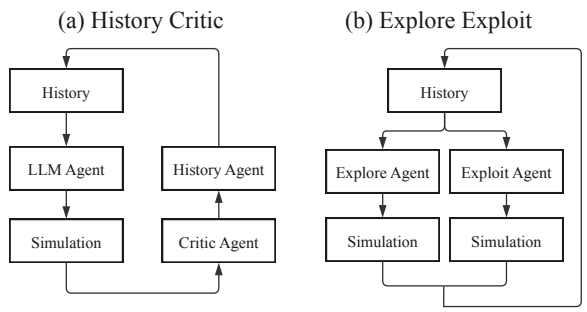

Fig. 1. Design of the AI-driven multi-agent optimization pipelines.

II. DSE Pipelines

A. LLM-in-the-loop

The first pipeline places an LLM agent in an iterative feedback loop, where the agent proposes candidate configurations and refines its proposals based on accumulated evaluation results. At each iteration, the agent receives the design space constraints along with the target DNN topology describing the layer dimensions of the workload. A sliding window of the N most recent evaluations and the best configuration found so far are provided as context. Based on this information, the agent generates a new candidate via structured output, which is parsed and passed to the simulator for evaluation. The resulting metric is then appended to the history for the next iteration.

Providing the workload topology (i.e., the DNN layers' information) directly to the agent is a key design choice, as

TABLE I
DESIGN SPACE PARAMETERS.

Parameter	Range
Array Height	{32, 34, 36, ..., 512}
Array Width	{32, 34, 36, ..., 512}
Dataflow	{OS, WS, IS}
SRAM Buffer Size†	{64, 66, ..., 8192} KB
Per-layer Dataflow††	{OS, WS, IS} $\times L$ layers

†EDAP experiments only. ††Scenarios 3–5 only.

it enables the LLM to reason about the relationship between layer structure and the array configuration, rather than treating the design space as a black box.

B. Multi-LLM-agents

The second set of pipelines (Figure 1) extends the single-agent approach by introducing multiple LLM agents that serve distinct roles within the optimization loop [15], to encourage broader exploration and avoid local optima.

The first variant, the *history-critic* architecture, augments the proposer agent with a separate critic agent. Every 5 iterations, the critic produces a structured analysis identifying best-performing patterns, promising regions, underexplored areas, and bottleneck layers. This analysis is fed to a history agent, which creates a rolling summary that is injected into the proposer's context in subsequent iterations. The rolling summary helps the proposer maintain focus on the most relevant patterns and avoid losing track of important findings as the history grows, effectively serving as a focused memory across the optimization run.

The second variant, the explore-exploit architecture, establishes two competing agents that share a common evaluation history. At each iteration, both agents receive the same history but operate under different directives: the explore agent is instructed to propose configurations meaningfully different from what has been tried, while the exploit agent is instructed to refine the best-known configuration through targeted local adjustments. Both proposals are evaluated concurrently by the simulator, and recorded in the shared history.

III. EXPERIMENTS

A. Experiment Setup

For systolic array simulations, we use SCALE-Sim v3 [8], [10], a cycle-accurate simulator for systolic array architectures, integrated with Accelergy [14] for energy estimation using Aladdin and CACTI 7.0 component models at a 45nm technology node. Area is estimated using the same component models, combining the PE area (scaled by array dimensions) with SRAM area for three double-buffered banks. The primary optimization metrics are EDP (Energy-Delay Product) and EDAP (Energy-Delay-Area Product), defined as the product of compute cycles and energy (in pJ), with EDAP additionally incorporating chip area (in mm^2). Lower values indicate more efficient designs.

We mainly compare with the following optimization options as baselines:

TABLE II
SINGLE-WORKLOAD EDP OPTIMIZATION RESULTS (50 ITERATIONS).

Workload	Method	Mean EDP	Std	Best
ViT-S	LLM	3.42e+13	0	3.42e+13
	GPBO	3.88e+13	1.69e+12	3.67e+13
ViT-L	LLM	6.87e+14	0	6.87e+14
	GPBO	7.36e+14	4.55e+13	7.02e+14
ResNet18	LLM	3.62e+15	0	3.62e+15
	GPBO	3.68e+15	5.54e+13	3.61e+15
MobileNet	LLM	3.73e+15	9.77e+14	2.75e+15
	GPBO	2.53e+15	2.97e+13	2.48e+15

- Random search: A uniformly random search.
- GPBO: A GP-based Bayesian optimization (GPBO) implemented using scikit-optimize [5] with 10 initial random exploration points.
- GA: A genetic algorithm baseline using PyGAD [3].

Experiments are repeated with 3 random seeds to measure variance, as detailed in each scenario.

The design space (Table I) consists of the systolic array height and width (even integers from 32 to 512), dataflow selection (output stationary, weight stationary, or input stationary), and SRAM buffer size (even integers from 64 to 8192 KB). For scenarios with per-layer dataflow optimization, each layer has an independent dataflow parameter, expanding the search dimensionality with the number of layers in the workload.

For the LLM-based pipelines, we use Claude Sonnet with 0.7 temperature as the primary LLM with structured output via a dynamically generated Pydantic schema. The agent receives a sliding window of the 20 most recent evaluations and the best configuration found so far. For the initial single-workload experiments, we additionally evaluate Claude Opus, Gemini 2.5 Pro, and OpenAI o1 to assess sensitivity to the choice of LLM.

B. Scenario 1. Optimizing SAs for individual DNNs

In the first scenario, we optimize a systolic array configuration for a single DNN workload. We evaluate the LLM-in-the-loop pipeline against GPBO baseline across multiple workloads spanning different model families: ViT-S and ViT-L (5 GEMM layers from a single transformer block each), ResNet18, and MobileNet. Each method runs for 50 iterations, and we report the EDP.

Table II presents the EDP results for 3 seeds. On most workloads, the LLM agent converges to competitive configurations within the first 5–10 iterations, while GPBO requires its full initial exploration phase before improving. The LLM achieves lower mean EDP than GPBO on ViT-S (by 11.9%), ViT-L (by 6.6%), and ResNet18 (by 1.6%), with zero variance across seeds on all three workloads, indicating it consistently converges to the same optimal configuration. On ViT-S, all three LLM models (Claude Opus, Gemini 2.5 Pro, and OpenAI o1) discover 196-row arrays matching the workload's patch dimension (M=196 for 224×224 input with 16×16 patches), while GPBO converges to generic power-of-two sizes. This illustrates the LLM's ability to match array

dimensions to workload structure, a capability absent in black-box optimizers. However, on MobileNet, GPBO outperforms the LLM by 32.2%. MobileNet features unusual depthwise layers (num_filters=1) that the LLM fails to recognize. The LLM proposes $\sim$256$\times$256 arrays while GPBO converges to $\sim$200$\times$70, suggesting that workload-aware reasoning is less effective when layer structures fall outside patterns commonly seen in training data.

In Scenarios 3–5, we use the EDAP metric with per-layer dataflow, which presents a substantially more challenging optimization landscape.

C. Scenario 2. Optimizing SA for multiple DNNs

In the second scenario, we optimize a single systolic array configuration to serve multiple DNN workloads under a temporal multi-tenancy model. We evaluate using three concurrent workloads (AlexNet, ViT-S, and MobileNet) with geometric mean of per-workload EDP as the objective. The LLM achieves a best geomean EDP of 4.09e+14 compared to GPBO's 4.37e+14. The LLM again demonstrates workload-aware sizing, with one seed discovering a 196$\times$256 configuration that matches the ViT-S patch dimension, achieving a 6.4% improvement over the best GPBO result. Both methods converge within the first 20 iterations on this workload mix, suggesting that temporal multi-tenancy over workloads with similar computational profiles does not present a substantially harder challenge for the LLM than single-workload DSE.

D. Scenario 3. Optimizing SA and its dataflow selection

Recent reconfigurable systolic array designs such as Flex-TPU [2] and ReDas [4] enable runtime switching between output-stationary, weight-stationary, and input-stationary dataflows on a per-layer basis with minimal hardware overhead. Optimizing for such architectures requires jointly selecting the hardware parameters and a dataflow assignment for each layer, creating a mixed discrete-continuous search space that grows combinatorially with the number of layers. We evaluate this scenario on AlexNet (5 layers, $3^5 = 243$ dataflow combinations) using the EDAP metric. Compared to the fixed-dataflow scenario in Section III-B, the per-layer search space presents a substantially harder optimization challenge. On AlexNet EDAP, the LLM baseline achieves a mean of 2.36e+16, while GPBO achieves 1.79e+16 over 30 iterations (Table III). Both methods discover that early convolutional layers with larger filters benefit from weight-stationary execution, while later layers favor output-stationary, confirming that per-layer dataflow selection captures meaningful architectural tradeoffs. The baseline LLM underperforms GPBO on this harder problem, motivating the ablation study and multi-agent experiments in the following sections, where we investigate techniques to close this gap.

E. Scenario 4. Optimizing SA using more advanced LLM modifications

We conduct an ablation study to investigate whether augmenting the LLM agent with additional prompt engineering techniques can improve search quality on the per-layer dataflow optimization task. We evaluate six modifications on top of a minimal baseline (B0), each adding a single feature, on the AlexNet EDAP per-layer task using Claude Sonnet with 30 iterations, 3 random seeds each. The six additions are: an explanation of systolic array folding techniques (F1), an explicit exploration directive (E1), increased sampling temperature of 0.9 (T1), temperature annealing from 1.0 to 0.3 (T2), periodic self-critique every 10 iterations (C1), and per-layer cycle and energy breakdowns in the history (L1).

TABLE III
ABLATION STUDY AND AGENT ARCHITECTURE RESULTS ON ALEXNET EDAP PER-LAYER (30 ITERATIONS).

Method	Mean EDAP	Std	Best
Feature-flag ablation			
T2 annealing	1.49e+16	1.89e+15	1.37e+16
E1 +explore	1.66e+16	5.18e+15	1.34e+16
T1 temp=0.9	2.03e+16	8.43e+15	1.42e+16
B0 baseline	2.36e+16	6.28e+15	1.99e+16
L1 +layer-stats	2.43e+16	7.62e+15	1.84e+16
C1 +critique	3.46e+16	1.91e+16	1.54e+16
F1 +folding	3.57e+16	3.77e+16	1.03e+16
Agent architectures			
A1 baseline LLM	2.36e+16	6.28e+15	1.99e+16
A2 history+critic	2.30e+16	1.14e+16	1.45e+16
A3 explore+exploit	1.80e+16	6.89e+15	1.02e+16
A2+features*	**1.09e+16**	**3.57e+15**	**6.56e+15**
Baselines			
GPBO	1.79e+16	4.80e+15	1.31e+16
Random	8.12e+16	4.15e+16	3.29e+16

*A2 + T2 annealing, E1 explore, F1 folding, L1 layer-stats.

As shown in Table III, temperature annealing (T2) proves most effective, achieving the best mean EDAP with the lowest variance across seeds. The exploration directive (E1) is the second-best variant, confirming that explicitly encouraging diversity improves search coverage.

The folding explanation (F1) presents an interesting case: domain-specific knowledge helps when the agent applies it well, but can mislead when the agent over-anchors on heuristics, causing high variance.

F. Scenario 5. Optimizing SA using Multi-agentic systems

We evaluate whether multi-agent pipelines improve over the single-agent baseline on the per-layer dataflow optimization task described in Section III-D, comparing three agent architectures on AlexNet EDAP, each run for 30 iterations with 3 random seeds using Claude Sonnet. The single-agent baseline (A1) uses the LLM-in-the-loop pipeline described in Section II-A. The history-critic architecture (A2) augments this with a periodic critic agent and history summarizer as described in Section II-B.

The explore-exploit architecture (A3) separates exploration and exploitation into two competing agents that propose configurations concurrently. A2 demonstrates improved best-case performance over A1, with its strongest seed reaching 1.45e+16 compared to A1's best of 1.99e+16, though it exhibits higher variance across seeds (Table III). A3 achieves the strongest overall results with a mean EDAP of 1.80e+16, a 24% improvement over A1, and its best seed reaches

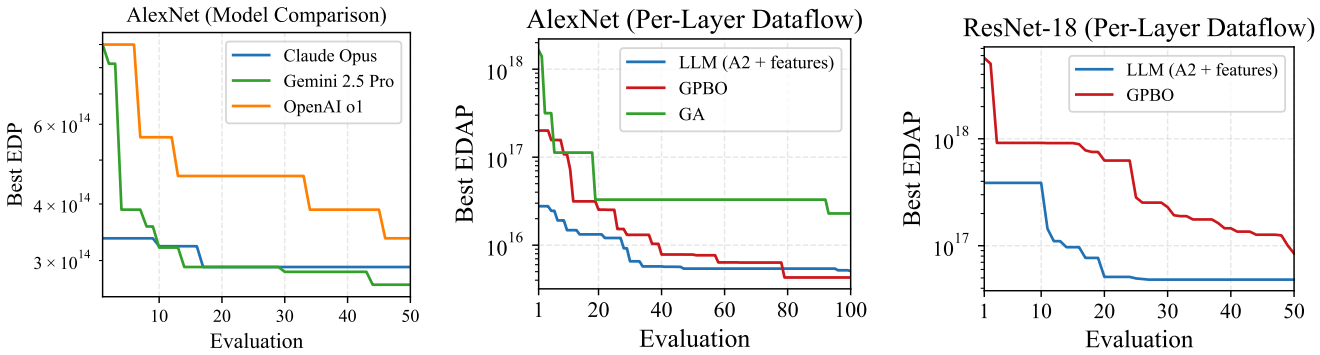

Fig. 2. (a) LLM model comparison on AlexNet EDP. (b) LLM vs GPBO on AlexNet EDAP per-layer. (c) LLM vs GPBO on ResNet18 EDAP per-layer.

1.02e+16, the lowest EDAP observed across all our standalone LLM configurations. The exploit agent wins the majority of head-to-head selections, but the explore agent is often responsible for significant breakthroughs that redirect the search toward better regions of the design space. However, A3 requires two concurrent simulator evaluations per iteration, doubling simulation cost relative to single-agent methods. Finally, combining selected ablation features (T2, E1, F1, L1) with the A2 history-critic architecture yields rapid convergence, with its best seed achieving substantially lower EDAP than GPBO's best at 30 iterations (Figure 2). However, given sufficient iterations (80+), GPBO narrows this gap and eventually surpasses the LLM result, suggesting the LLM's advantage is sample efficiency, not final solution quality.

IV. Discussions and Future Directions

Our experimental evaluation is subject to several simplifying assumptions. The energy and area models used (Aladdin and CACTI 7.0 at 45nm) provide estimates rather than precise silicon measurements, and results may differ under more detailed physical design tools or at smaller technology nodes. SCALE-Sim, while cycle-accurate for compute, does not model all aspects of a real hardware deployment. The results in Section III were obtained from SCALE-Sim's CALC mode for more rapid DSE. We ran extensive experiments in USER mode for more detailed memory stalls, and found that the main trends are still preserved, especially between GPBO and agentic methods. For large array configurations, propagation delay accumulates as data moves, which could impact the practical viability of the larger configurations discovered by the optimizer. SCALE-Sim supports three dataflow modes (output stationary, weight stationary, and input stationary) but does not consider their dynamic reconfiguration aspects. Our experiments for Scenarios 3–5 in Section III did not take into account such dataflow reconfiguration costs. Each iteration also requires an LLM inference call costing approximately \$0.02, totaling almost \$20 for all LLM experiments in Table III, substantially higher than statistical methods like BO or GA which require only local compute.

Several future directions follow naturally from this work. Expanding the design space to include memory hierarchy parameters, tiling strategies, and sparsity-aware configurations would better reflect the complexity of modern accelerator design. A particularly promising direction is the integration of LLM agents with conventional but mathematically rigorous models such as GPBO, which may open a new opportunity to leverage the advantages from both approaches. Finally, as multi-core and chiplet-based accelerator architectures become increasingly prevalent, extending the DSE framework to handle spatial multi-tenancy across discrete compute tiles presents a compelling direction for future investigation.

V. Related Work

There have been numerous DSE techniques for systolic array-based accelerators. For example, Timeloop [7] and CoSA [6] enable accelerator optimizations with analytical modeling. Dinh et al. leverage GP-based BO for sample-efficient search for DNN mapping [1]. Polaris [9] utilizes both analytical modeling and automated search-based optimizations for multi-fidelity optimization. Our work, on the other hand, explores an AI-agentic approach for accelerators.

AI and LLM-driven optimizations also enable many interesting hardware optimizations. Zhang et al. [17] present an LLM-based approach for GPU architecture exploration. Shih et al. [12] present a multi-LLM agentic approach for optimizing an autonomous driving system. Our work focuses on AI-driven optimizations for systolic array-based accelerators, building on the LLM-as-optimizer paradigm [16].

VI. Conclusion

This paper explores the feasibility of LLM-agentic approaches for design space exploration of systolic array architectures. We present two DSE pipelines, an iterative LLM-in-the-loop and a multi-agent framework with history-critic and explore-exploit variants, and evaluate them against GPBO and genetic algorithm baselines across multiple scenarios using SCALE-Sim and Accelergy. Our results suggest that LLM-based DSE is a promising direction, particularly for mixed discrete-continuous design spaces and when the search budget is tight, where the LLM's semantic understanding of workload structure complements traditional statistical optimization.

Acknowledgments

We used generative AI tools solely for general writing and scripting assistance. All technical content, results, and conclusions remain the authors' original work.

References

[1] G. Dinh, I. K. J. Valsala, H. Luo, C. Hong, Y. Cho, J. Demmel, S. Li, and Y. Liu, "Sample-Efficient Mapspace Optimization for DNN Accelerators with Bayesian Learning," in *Architecture and System Support for Transformer Models (ASSYST @ISCA 2023)*, 2023. [Online]. Available: https://openreview.net/forum?id=el1iPvMqqTY

[2] M. Elbtity, P. Chandarana, and R. Zand, "Flex-TPU: A Flexible TPU with Runtime Reconfigurable Dataflow Architecture," 2024. [Online]. Available: https://arxiv.org/abs/2407.08700

[3] A. F. Gad, "PyGAD: An Intuitive Genetic Algorithm Python Library," *Multimedia Tools and Applications*, pp. 1–14, 2023.

[4] M. Han, L. Wang, L. Xiao, T. Cai, Z. Wang, X. Xu, and C. Zhang, "ReDas: A Lightweight Architecture for Supporting Fine-Grained Reshaping and Multiple Dataflows on Systolic Array," *IEEE Transactions on Computers*, vol. 73, no. 8, pp. 1997–2011, 2024.

[5] T. Head, M. Kumar, H. Nahrstaedt, G. Louppe, and I. Shcherbatyi, "Scikit-Optimize: Sequential Model-Based Optimization," 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5565057

[6] Q. Huang, M. Kang, G. Dinh, T. Norell, A. Kalaiah, J. Demmel, J. Wawrzynek, and Y. S. Shao, "CoSA: Scheduling by Constrained Optimization for Spatial Accelerators," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 2021, pp. 554–566.

[7] A. Parashar, P. Raina, Y. S. Shao, Y.-H. Chen, V. A. Ying, A. Mukkara, R. Venkatesan, B. Khailany, S. W. Keckler, and J. Emer, "Timeloop: A Systematic Approach to DNN Accelerator Evaluation," in *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2019, pp. 304–315.

[8] R. Raj, S. Banerjee, N. Chandra, Z. Wan, J. Tong, A. Samajdar, and T. Krishna, "SCALE-Sim v3: A modular cycle-accurate systolic accelerator simulator for end-to-end system analysis," in *2025 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 2025, pp. 186–200.

[9] C. Sakhuja, C. Hong, and C. Lin, "Polaris: Multi-Fidelity Design Space Exploration of Deep Learning Accelerators," 2024. [Online]. Available: https://arxiv.org/abs/2412.15548

[10] A. Samajdar, Y. Zhu, P. Whatmough, M. Mattina, and T. Krishna, "SCALE-Sim: Systolic CNN Accelerator Simulator," *arXiv preprint arXiv:1811.02883*, 2018.

[11] T. Schick, J. Dwivedi-Yu, R. Dessi, R. Raileanu, M. Lomeli, E. Hambro, L. Zettlemoyer, N. Cancedda, and T. Scialom, "Toolformer: Language Models Can Teach Themselves to Use Tools," in *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36. Curran Associates, Inc., 2023, pp. 68 539–68 551. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/file/d842425e4bf79ba039352da0f658a906-Paper-Conference.pdf

[12] P.-A. Shih, S.-H. Wang, Y.-C. Li, C.-H. Tu, and C.-H. Chang, "A Multi-Agent LLM Framework for Design Space Exploration in Autonomous Driving Systems," 2025. [Online]. Available: https://arxiv.org/abs/2512.08476

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[14] Y. N. Wu, J. S. Emer, and V. Sze, "Accelergy: An Architecture-Level Energy Estimation Methodology for Accelerator Designs," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2019, pp. 1–8.

[15] J. Xu, F.-F. Wei, and W.-N. Chen, "RoCo: Role-Based LLMs Collaboration for Automatic Heuristic Design," 2025. [Online]. Available: https://arxiv.org/abs/2512.03762

[16] C. Yang, X. Wang, Y. Lu, H. Liu, Q. V. Le, D. Zhou, and X. Chen, "Large Language Models as Optimizers," in *International Conference on Learning Representations*, B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun, Eds., vol. 2024, 2024, pp. 12 028–12 068. [Online]. Available: https://proceedings.iclr.cc/paper_files/paper/2024/file/3339f19c5fcee3ad74502947a32be9e6-Paper-Conference.pdf

[17] T. Zhang, R. Ma, S. Xu, Y. Xiong, and P. Cheng, "LUMINA: LLM-Guided GPU Architecture Exploration via Bottleneck Analysis," 2026. [Online]. Available: https://arxiv.org/abs/2603.05904