CHIP-MAP: A Collaborative Optimization Framework for Macro Placement Using Large Language Models

Yiming Du[1,*] yimingdu@stu.ahu.edu.cn
RenYe Yan[1,*] victory@stu.pku.edu.cn
yunfan yang[1] yangfanfan@pku.edu.cn
Frank Qu[2] zhengqu.ece@gmail.com
Jiajun Tan[1] javin@stu.ahu.edu.cn
ZhiYu Zheng[4] zyzheng24@m.fudan.edu.cn
Yiming Gan[3] ganyiming@ict.ac.cn
LING LIANG[1,‡] lingliang@pku.edu.cn
Zongwei Wang[1] wangzongwei@pku.edu.cn
YiMao Cai[1,‡] caiyimao@pku.edu.cn

[1]School of Integrated Circuits, Beijing Advanced Innovation Center for Integrated Circuits, Peking University, Beijing, China
[2]Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA, USA
[3]Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[4]State Key Laboratory of Integrated Chip and System, Fudan University, Shanghai, China

***Abstract*—As integrated circuits continue to grow in both scale and complexity, macro placement plays a critical role in physical design, directly affecting chip-level performance, power, and area (PPA). Traditional macro placement methods, such as simulated annealing, analytical optimization, and reinforcement learning, face limitations including slow convergence, heavy dependence on large datasets, and over-reliance on intermediate PPA indicators rather than final PPA. Large language models (LLMs) offer strong generative power and semantic reasoning that can potentially automate macro layout tasks while addressing the aforementioned problems in traditional methods, but their limited understanding of layout rules and lack of iterative, feedback-driven refinement make direct application challenging. To address this, we propose CHIP-MAP, a macro placement framework based on multi-agent collaboration and feedback-driven optimization. Furthermore, we introduce two innovative tools: the Module Link Weight Analyzer (MWA) and the Standard Cell Usability Score (SCUS), which are designed to guide fine-grained layout refinement. We evaluate CHIP-MAP on five benchmarks ranging from low-power cores to large multi-core processors implemented at 130nm and 45nm technology nodes. Results show that it achieves up to 1.5% area reduction and an average repair of 61.6% of total negative slack (TNS), while also reducing wirelength and improving timing.**

***Index Terms*—Electronic design automation; Macro Placement; Large language models; physical design.**

I. Introduction

Macro placement plays a critical role in modern digital chip design, directly affecting the chip's performance, power, and area (PPA). As technology nodes continue to advance, the complexity of design and constraints increases, making macro placement optimization an increasingly challenging task [1]–[6]. In traditional industrial practice, macro placement often relies on manual efforts that are heavily experience-driven, which makes the process time-consuming and difficult to generalize. While existing automated placement methods can generate reasonable layouts, there are still bottlenecks in further optimizing PPA performance. Therefore, proposing a novel optimization method that can effectively improve macro placement PPA performance has become a key issue that needs to be addressed.

Existing macro placement methods primarily include black-box optimization, analytical methods, and reinforcement learning techniques. Black-box optimization methods, such as Simulated Annealing (SA) [7], treat the placement task as a combinatorial search guided by a heuristic cost function. Although simple and widely used, they often converge slowly and are not well suited for larger or more complex designs. Analytical methods, such as DREAMPlace [8], express placement objectives (e.g., wirelength, density) as differentiable functions, enabling the optimization of layout quality. However, these methods are prone to getting stuck in local optimal and may overlook non-differentiable design constraints. Reinforcement learning methods, such as ChiPFormer [9], provide a data-driven perspective by learning placement strategies from historical chip layouts. While they have potential in sequential decision-making, these methods heavily rely on large labeled datasets and proxy objectives (e.g., macro HPWL) that correlate poorly with final PPA metrics. Furthermore, they typically lack effective feedback mechanisms for iterative improvement.

The integration of LLMs into EDA provides new possibilities for addressing these challenges. Recent studies [10] highlight how LLMs enhance design efficiency by extracting insights from complex datasets and automating iterative design processes. However, directly applying LLMs to macro placement remains difficult: they fail to accurately interpret physical design constraints, resulting in illegal layouts or poor area utilization; they lack domain-specific knowledge and struggle to iteratively refine solutions based on layout feedback.

To address these limitations, we propose CHIP-MAP, a hierarchical multi-agent framework that integrates LLMs for macro

*Equal Contribution ‡Corresponding Author

Beijing Natural Science Foundation (L223004), National Natural Science Foundation of China (62341407, 62322401, U24B20166, 62495102), in part by the '111' Project (B18001)

978-3-9826741-1-7/DATE26/© 2026 EDAA

placement optimization. This framework translates physical constraints into LLM-interpretable representations and iteratively refines macro placement through multi-agent evaluation and feedback. The contributions of this paper are as follows:

- We proposed the CHIP-MAP framework, a closed-loop macro placement optimization system based on multi-agent collaboration that leverages large language models to enhance overall coordination and iterative refinement in the physical design process.
- We propose a novel constraint-structuring and adaptive grouping mechanism. It leverages LLM reasoning to make macro classification and complex constraint interpretation more effective in physical design.
- We developed the Module Link Weight Analyzer (MWA) and Standard Cell Usability Score (SCUS) to guide feedback-driven macro placement refinement.
- We conducted experiments on multiple industrial-scale design cases, and the results show that our method achieved up to a 1.5% reduction in layout area and significantly improved timing performance, with an average repair of 61.6% of the total negative timing slack (TNS).

II. Related Work

LLMs have recently shown strong potential in physical design optimization, particularly in macro placement tasks, where they use natural language instructions and advanced generative capabilities for design automation. ChatPattern [11] enables flexible layout pattern creation through natural language interaction; LayoutCopilot [12] translates high-level design intent into executable commands; and LayoutLLM [13] employs layout-aware pretraining to better understand hierarchical layout structures. For macro placement optimization, LLMs combine high-level constraints with reinforcement learning strategies. In standard cell implementation, Ho et al. [15] achieved 19.4% cell area reduction and 23.5% improvement in LVS/DRC-clean layouts.

III. Motivation

Although large language models (LLMs) have shown promise in electronic design automation (EDA), their direct application to macro placement remains challenging. Our preliminary evaluation shows that LLMs can generate layouts that satisfy basic alignment and spacing rules, indicating their potential as high-level design assistants. However, these layouts still exhibit critical deficiencies that limit their practical usability in real-world macro placement tasks.

A. Limited Understanding of Physical Design Constraints

LLMs struggle to correctly interpret physical design constraints specified in EDA files such as LEF, LIB, and SDC. As illustrated in Fig. 1(a), key parameters (e.g., CORE_AREA and MACRO_PLACE_HALO) are often misinterpreted, resulting in macros placed outside the core or overlapping with each other. Unlike human designers, who can quickly reason about constraints and adjust placements accordingly, LLMs lack such domain-specific understanding, making their layouts infeasible for practical design.

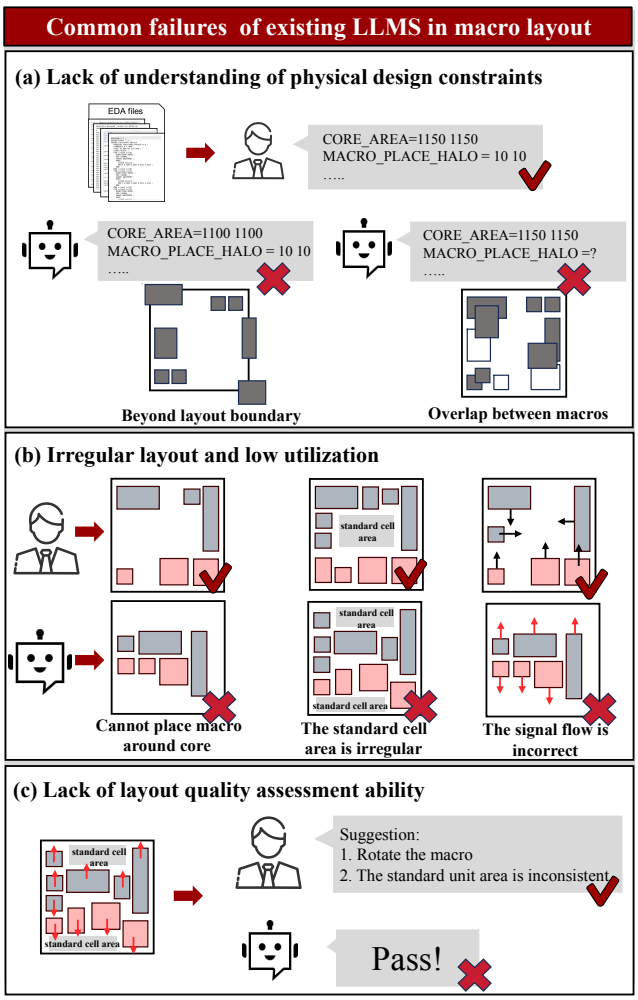

Fig. 1: Comparison between LLM-based and human engineer approaches in macro layout design. The figure illustrates common limitations when LLMs attempt macro placement compared to experienced human engineers.

B. Irregular Layout and Low Area Utilization

LLMs often fail to place macros efficiently around the core region, leading to irregular layouts and low area utilization. As shown in Fig. 1(b), generated layouts typically contain large unused spaces and fragmented standard cell regions. In contrast, human designers leverage design experience and signal flow considerations to achieve compact and balanced layouts. The absence of such structured reasoning results in suboptimal macro arrangements that degrade PPA performance.

C. Lack of Layout Quality Evaluation and Refinement

LLMs lack an integrated layout quality assessment mechanism to identify and correct critical issues such as component misalignment, inconsistencies in standard cell regions, or routing congestion. As illustrated in Fig. 1(c), their layouts are often labeled as `Pass` without detailed checks. In contrast, human designers iteratively refine layouts by identifying and fixing defects to meet PPA requirements. Without such feedback and optimization, LLMs cannot ensure design quality.

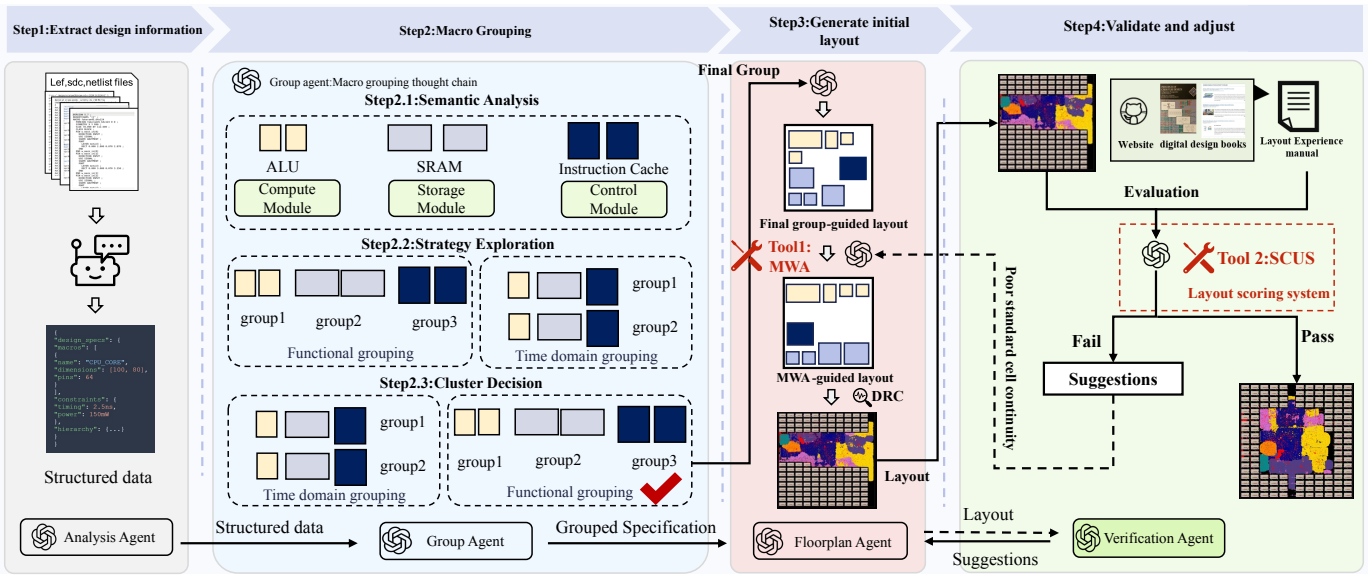

Fig. 2: Overview of the CHIP-MAP framework. The process consists of four stages: extracting structured design data, performing semantic macro grouping, generating an initial layout guided by MWA, and validating the result with SCUS scoring.

Overall, these observations indicate that while LLMs can generate preliminary macro layouts, they lack the domain knowledge and iterative optimization capability required for practical macro placement. This motivates the need for a framework that combines LLM-based generation with targeted optimization and feedback to produce high-quality, manufacturable layouts.

IV. Framework Of CHIP-MAP

A. Overview of The CHIP-MAP

We propose *CHIP-MAP*, a multi-agent collaborative framework that systematically integrates design understanding, logical grouping, layout optimization, and iterative quality evaluation. As shown in Fig. 2, CHIP-MAP consists of four specialized agents: Analysis Agent, Group Agent, Floorplan Agent, and Verification Agent. These agents work together in a closed-loop process to refine macro placement quality.

Analysis Agent: To address the challenge of large language models in understanding physical design rules and constraints, we introduce the Analysis Agent as the first step of the framework. As shown in Fig. 3, this agent receives the input EDA files (LEF, LIB, and SDC). Since the original Verilog netlist is too large and directly inputting it into the LLM would exceed the context length, we employ script-based automatic extraction: the Verilog source is parsed to obtain module definitions, instantiation relationships, and port connections. Based on this information, a module hierarchy tree and a critical connection summary are constructed, enabling the LLM to better capture the design structure and timing relationships.

Subsequently, the Analysis Agent parses the input data and transforms it into structured representation according to a predefined Spec List: macro information captures size and orientation constraints, hierarchical data describe module dependencies, physical rules encode placement restrictions, and design objectives map to timing and area requirements. This structured representation not only preserves design semantics but also provides reliable input for subsequent optimization.

Group Agent: To address the issues of low area utilization and disrupted signal flow in LLM-generated layouts, the Group Agent performs semantic clustering of macros based on Chain-of-Thought reasoning. It first receives structured design data from the Analysis Agent and classifies macros according to their functional roles, such as grouping ALU as compute modules and instruction cache as control modules. The Group Agent then generates and compares different grouping strategies: a function-driven strategy clusters macros with similar functions and high constraint correlation to improve area utilization, while a timing-driven strategy places macros on critical timing paths closer together to reduce wirelength and delay. Finally, through step-by-step reasoning, the Group Agent selects the optimal grouping scheme and outputs the final group assignments together with the structured design data in JSON format, which are passed to the next agent for further processing.

Floorplan Agent: After receiving the structured grouping information, the Floorplan Agent generates the initial macro layout. At first, the macros are placed in a random initialization. To refine placement, the Floorplan Agent leverages the connectivity weights generated by MWA to determine the positions of modules both within and between groups.

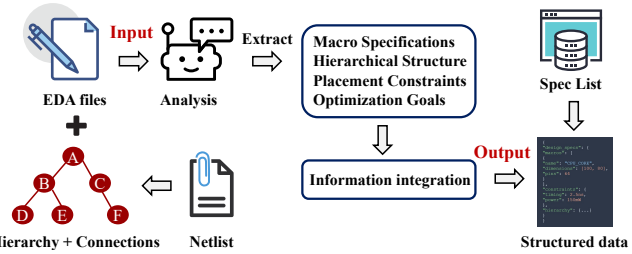

Fig. 3: Analysis Agent workflow: Transform complex design information into structured and organized data.

Prompt:
"Summarize the [Task Objective] from the [Source Artifacts] with the [Agent Notes]. Respond in JSON following the [Output]."

Analysis Agent
[Task Objective]: Extract structured constraints such as macro sizes, pins, orientation restrictions, macro halo, clock domains, and layout dimensions…
[Source Artifacts]: The information extracted from the netlist, LEF, LIB,SDC.
[Agent Notes]:The macro orientation format is {R0,R90,R180,R270,MX,MY}…
[Output]: {**"core_area"**: [1150.0, 1150.0], **"die_area"**: [1200.0, 1200.0], **"macro_constraints"**: [{"name": "Macro1", "size": [128.5, 64.2], "orientation_restrictions": ["R0", "R180", "MY"]…}

Group Agent
[Task Objective]: Classify each macro by functional characteristics, generate multiple grouping strategies, and select the optimal strategy for downstream placement.
[Source Artifacts]: Analysis Agent Output.
[Agent Notes]: Follow three-step process: (1) classify each macro by type and characteristics; (2) generate functional and timing-based grouping strategies with reasoning; (3) select optimal strategy based on area utilization benefits.
[Output]: {**"classification"**:{"Macro1":"Storage","Macro2":"Compute","..."},**"strategies"**:{"functional":{"groups":[{"macros":["Macro2","Macro3","..."]},"..."],"reasoning":"[]"},"timing":{"groups":[{"macros":["Macro2","Macro4","..."]},"..."],"reasoning":"[]"},"..."},**"selection"**:{"strategy":"functional","final_groups":[{"id":"group1","macros":["Macro2","Macro3","..."]},"..."]}}

Floorplan Agent
[Task Objective]: Generate macro coordinates using MWA connectivity weights, ensure DRC compliance, and iteratively optimize layout based on verification feedback.
[Source Artifacts]: Group Agent Output, Verification Agent feedback (suggestions), MWA.
[Agent Notes]: Use MWA to prioritize highly connected macro pairs for proximity placement. Ensure no overlaps, respect halo boundaries. Apply verification suggestions for edge alignment and gap optimization.
[Output]: {macro1 MY 100 200;macro2 R0 52 64…}

Verification Agent
[Task Objective]: Evaluate layout quality using SCUS metrics, detect continuity and alignment issues, generate optimization suggestions, and determine convergence status for iterative refinement.
[Source Artifacts]: Floorplan Agent coordinates, Layout Experience Manual,SCUS.
[Agent Notes]: Calculate SCUS score considering continuity and compactness. Detect issues like poor standard cell continuity, excessive gaps, misalignment. Generate structured suggestions with target areas for improvement.
[Output]: {**"scus_score"**:0.8877,**"layout_quality"**:"PASS",**"issues_detected"**:[],**"suggestions"**:[{"action":"increase_macro_edge_alignment","target_area":"upper_right_vacant"}],"convergence_achieved":true…}

Fig. 4: CHIP-MAP Multi-Agent Prompt Engineering Framework. Each agent card shows the standardized prompt structure. The four agents form a sequential workflow from constraint extraction to layout optimization.

Modules with higher intragroup connectivity are placed in compact and regular patterns (e.g., an $m \times n$ matrix), while the relative positions between groups are determined by their overall connectivity strength. Meanwhile, the Floorplan Agent also considers physical constraints from the structured data, including minimum spacing, halo boundaries, and I/O interface locations. Finally, when the preliminary layout is completed, the layout coordinates after performing the Design Rule Check (DRC) are provided as input to the next agent.

Verification Agent: The Verification Agent evaluates the layout using the SCUS, which measures the continuity and compactness of the remaining standard cell regions. If the SCUS score decreases compared to the previous iteration or design rule violations are detected, the Verification Agent refers to the Layout Experience manual to generate targeted optimization suggestions and feeds them back to the Floorplan Agent for updates. The feedback cycle is executed up to three rounds until the SCUS score is maximized and all design rules are satisfied, ensuring the final layout reaches its optimal state.

Integration of Human Expertise: In the Verification Agent, we integrate human expertise on layout strategies to generate targeted optimization suggestions. The Layout Experience Manual consolidates years of practical experience and heuristic rules from the chip design community, academic resources, and textbooks, encoding them as: (i) design rules and checklists (spacing, channel width, macro orientation, power/clock alignment, etc.); (ii) strategies for congestion and timing issues (buffering, routing, blockage handling); (iii) templates that convert expert knowledge into verifiable constraints. The agent uses this knowledge to evaluate floorplans and make deterministic decisions based on process specifications and SCUS scores.

B. Tools for Guidance and Placement Assessment

We designed a Dual-Tool Framework consisting of the MWA and the SCUS to support and enhance the chip layout process.

Module Link Weight Analyzer (MWA): MWA quantifies the proximity priority between any two macro modules A and B during placement. It constructs a sequential graph with flip-flops, macro units, and I/O pins as nodes, and traverses directed combinational logic paths using a breadth-first search algorithm. For each path, MWA computes the register hop count and signal bitwidth, and applies an exponential decay factor to penalize longer and indirect connections. The resulting connection weights guide the Floorplan Agent to place strongly connected macro pairs closer together, reducing critical path delay and improving layout efficiency. In this work, we adopt a method similar to [19] and define the MWA formula as:

$$W(A,B) = \sum_{p \in P(A,B)} \sum_{i=1}^{|p|} \frac{\text{bitwidth}(s_i)}{2^i} \tag{1}$$

where $P(A,B)$ is the set of signal paths from module A to B, including direct and indirect connections; $|p|$ is the number of register hops in path p; i denotes the position of the i-th signal segment in the path; $\text{bitwidth}(s_i)$ is the width of signal segment s_i; and dividing by 2^i penalizes longer paths so that closer modules receive higher weights. Paths with hop count exceeding 4 are discarded.

Standard Cell Usability Score (SCUS): SCUS is a metric for evaluating layout quality by balancing connectivity and compactness. Based on the macro coordinates generated by the Floorplan Agent, SCUS computes the size of the largest contiguous standard cell region and the effective usable standard cell area. These factors are combined into a numerical score that reflects area utilization quality. The score is used as feedback to guide further layout optimization. The SCUS formula is defined as follows:

$$C_{\text{final}} = \alpha \cdot \left(\max_{R \in \mathcal{R}} |R| \right) + \beta \cdot \left(1 - \frac{A_{\text{notch}}}{A_{\text{total}}} \right) \tag{2}$$

Where $\max_{R \in \mathcal{R}} |R|$ represents the size of the largest connected and uninterrupted standard cell region in the layout. A_{notch} denotes the area of irregular gaps (commonly referred to as notches) left between macros. These regions are unusable for standard cell placement, fragment the layout into smaller disconnected areas, and may increase routing congestion. A_{total} is the total chip area including both macros and standard cells. The weighting factors α and β balance the importance of connectivity and compactness in the final score.

TABLE I: EXPERIMENTAL RESULTS COMPARING LLM-BASED PLACEMENT AND BASELINE METHODS

Name	Tech	#Macros	#Cells	Dens. (%)	Freq. (ns)	Method	Timing (ps)		cong.	Power (mW)	WL (μm)	Area (μm^2)	DRC	NVP	Time (min)
							TNS	WNS							
Black_Parrot	45nm	220	1024k	68	3	**CHIP-MAP(GPT-4o)**	-60.25	-1.56	0.356	241	8145856	810234	545	521	5.6
						CHIP-MAP(Deepseek-R1)	-61.28	-1.58	0.355	241	8193282	810548	566	520	7.2
						OpenROAD	-71.88	-1.96	0.360	246	8209505	810360	825	541	28
						SA	-165.72	-1.90	0.385	248	8775993	813755	1253	552	—
						DREAMPlace	-198.21	-1.88	0.385	244	8914932	813703	1352	605	—
						ChiPFormer	-463.36	-1.75	0.392	250	8993077	816983	2035	1687	—
Ariane136	45nm	136	151k	68	2	**CHIP-MAP(GPT-4o)**	-3053.54	-1.28	0.225	541	6782462	783253	0	3123	3.6
						CHIP-MAP(Deepseek-R1)	-2996.20	-1.21	0.206	541	6772743	783426	0	3232	4.2
						OpenROAD	-3204.87	-1.32	0.249	550	7176803	792446	0	3468	5.1
						SA	-4116.46	-1.71	0.252	574	7296870	792731	0	3662	—
						DREAMPlace	-3293.09	-1.38	0.232	574	6885717	792608	0	3581	—
						ChiPFormer	-3266.36	-1.34	0.232	563	6705656	789664	0	5521	—
Swerv_wrapper	45nm	28	98k	65	2	**CHIP-MAP(GPT-4o)**	-515.07	-0.63	0.321	620	4008348	673169	103	1172	3.4
						CHIP-MAP(Deepseek-R1)	-569.54	-0.72	0.334	622	4014585	672647	112	1168	3.8
						OpenROAD	-751.62	-0.98	0.342	627	4015115	675207	147	1531	4.2
						SA	-880.45	-1.20	0.455	682	5313519	679744	152	1789	—
						DREAMPlace	-640.11	-0.89	0.375	666	4631643	675207	137	1165	—
						ChiPFormer	-843.14	-1.08	0.440	684	5144218	679744	149	1756	—
Chameleon	130nm	6	72k	50	7	**CHIP-MAP(GPT-4o)**	-1.72	-0.02	0	880	1515861	236835	0	29	3.2
						CHIP-MAP(Deepseek-R1)	-1.93	-0.02	0	891	1535475	237412	0	32	3.4
						OpenROAD	-2.22	-0.03	0	881	1559775	237956	0	38	0.04
						SA	-3.12	-0.04	0	882	1569845	238401	0	32	—
						DREAMPlace	-2.56	-0.03	0	892	1565845	238401	0	29	—
						ChiPFormer	-3.62	-0.04	0	882	1584586	238401	0	37	—
Microwatt	130nm	6	67k	50	15	**CHIP-MAP(GPT-4o)**	0.00	0.00	0	101	9413645	4901578	0	32	3.2
						CHIP-MAP(Deepseek-R1)	0.00	0.00	0	103	9511663	4943464	0	36	3.6
						OpenROAD	-0.37	-0.07	0	103	9612127	4962674	0	38	0.1
						SA	-0.48	-0.05	0	104	9645845	4951578	0	40	—
						DREAMPlace	-0.56	-0.07	0	103	9624545	4943464	0	39	—
						ChiPFormer	-0.52	-0.06	0	104	9686447	4975851	0	39	—

C. Prompt Structured Multi Agent Orchestration

As illustrated in Fig. 4, CHIP-MAP adopts a prompt-structured collaborative framework that coordinates multiple specialized agents in a closed loop. As illustrated in Fig. 3, the coordination layer defines standardized inputs and outputs, enabling each agent to operate under a unified scheme and exchange information in a machine-executable format. The entire workflow, from EDA file ingestion to layout generation and verification, is represented in JSON, supporting iterative feedback integration, incremental optimization, and convergence checking. Through this modular and auditable mechanism, CHIP-MAP consistently produces legal and high-quality macro layouts under strict design rules and achieves efficient convergence within a limited number of iterations.

V. Experimental Analysis

A. Experiment Setup

The framework is built on the principles of prompt engineering, implemented in Python with a multi-agent architecture where all agents are configured with temperature set to 0 for deterministic outputs. The experiments were carried out on an Intel® Core™ i9-13900H CPU. The backbone language model is GPT-4o [17], accessed via the OpenAI API, while DeepSeek-R1 [18] was additionally used for comparison. The framework is fully compatible with mainstream EDA toolchains: it can replace existing macro placement engines such as Hier-RTLMP in OpenROAD, or serve as an enhancement module within commercial flows including Synopsys and Cadence. Its output is generated as macro coordinate files, which can be directly mappable to standard layout input formats accepted by Synopsys, Cadence, and OpenROAD, ensuring seamless integration.

To evaluate the framework, we employed public benchmarks from the OpenROAD GitHub repository [16], including 64-bit RISC-V cores (BlackParrot and Ariane136), SweRV_wrapper, Microwatt, and Chameleon. The benchmarks range from small designs such as Microwatt (6 macros, 67k cells) to complex processors like BlackParrot (220 macros, over 1M cells), providing comprehensive evaluation coverage.

B. Benchmark Comparison of Baseline Macro Placement Algorithms

Following the evaluation framework of Wang et al. [20], we performed an end-to-end evaluation of five macropositioning methods, including simulated annealing (SA), DREAMPlace, ChiPFormer, OpenROAD's Hier-RTLMP, and our proposed CHIP-MAP, on five representative benchmarks. For each design, the LEF/DEF output from the floorplanning stage was converted to Bookshelf format and processed by each placement engine. The resulting placements were then re-imported into the OpenROAD flow to perform standard cell placement (RePlace), clock tree synthesis (TritonCTS), and detailed routing (TritonRoute). We collect PPA metrics (TNS, WNS, power, WL, area) along with congestion, DRC, NVP (the number of network violations), and runtime (the execution time of the floor planning stage).

As shown in Table I, under legality constraints, CHIP-MAP achieves substantial improvements on multiple benchmarks. For BlackParrot, the MWA-guided Floorplan Agent places highly coupled and frequently communicating macros closer together, effectively shortening critical signal paths and reducing buffer overhead. As a result, TNS improves from -71.88 ps to -60.25 ps, WL decreases from 8.21M μm to 8.15M μm, and violations are also reduced. Furthermore, with function and timing aware grouping and a three round floorplan verification feedback loop, the system iteratively corrects problematic placements and excessive spacings, reducing violations while stabilizing timing. In terms of runtime, since the baselines are implemented in OpenROAD, we only compare macro-placement search time: CHIP-MAP is faster on large designs, while slightly slower on

small designs but still within an acceptable range. All results are averaged over 10 independent runs, with up to three feedback iterations per run.

C. Evaluation Under Extreme Area Constraints

We conducted a core area compression experiment to evaluate the robustness of the proposed CHIP-MAP framework against the traditional OpenROAD flow under extreme area constraints. For each benchmark and each placement method, we gradually reduced the available core region until reaching the minimum core area (MinCA) and recorded the key PPA metrics at their respective MinCA points, including timing (TNS, WNS), power, wirelength, area, and MinCA. In this evaluation, CHIP-MAP consistently achieved legal layouts at smaller MinCA values and delivered superior PPA results compared to OpenROAD. In contrast, OpenROAD's PPA metrics degraded more severely under its compression limitations. As shown in Table II, the optimization percentages relative to OpenROAD highlight the ability of CHIP-MAP to maintain design quality under tighter area constraints. These results demonstrate that our framework can generate ultracompact layouts with more favorable PPA trade-offs under extreme area constraints. The superior performance of CHIP-MAP under aggressive core area compression establishes it as a more robust solution than conventional methods for workflows that require maximal area efficiency.

TABLE II: PPA COMPARISON UNDER MINIMUM CORE AREA COMPRESSION LIMITS.

Name	Method	Timing (ps)		Power (mW)	WL (μm)	Area (μm^2)	MinCA
		TNS	WNS				
Black Parrot	CHIP-MAP	-142.17	-1.52	261	8054852	785426	6597190
	OpenROAD	-163.24	-2.10	264	8456325	806325	7182400
Ariane136	CHIP-MAP	-3022.32	-1.34	568	6642368	778456	1257760
	OpenROAD	-3366.36	-1.71	572	6693242	789339	1320200
Swerv_wrapper	CHIP-MAP	-584.04	-0.60	681	4781118	677565	935610
	OpenROAD	-683.55	-0.83	684	5131627	679442	935614
Chameleon	CHIP-MAP	-2.14	-0.03	882	1482132	234672	9832500
	OpenROAD	-3.62	-0.06	886	1502348	235423	9520000
Microwatt	CHIP-MAP	-0.23	-0.07	101	9488196	4953600	9334900
	OpenROAD	-0.84	-0.11	104	11354329	4982174	9396400

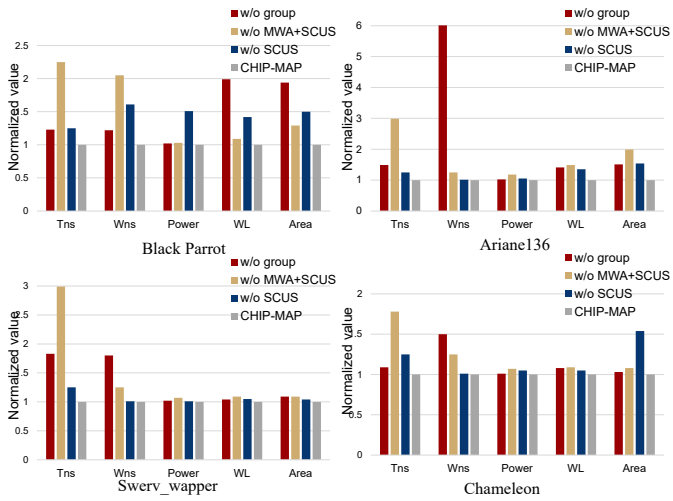

Fig. 5: Performance comparison of different CHIP-MAP component combinations on key metrics, averaged over 10 independent runs, each with 3 feedback iterations.

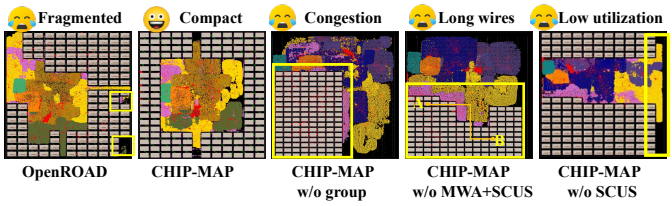

Fig. 6: Layout comparison on Ariane136 with ablation study. Yellow boxes indicate problematic regions in each layout.

D. Ablation Experiment

In this ablation study, we initially attempted a single-agent setting. However, a single LLM was unable to handle multiple tasks simultaneously, often resulting in invalid layouts or non-convergence on large-scale designs. Therefore, we only report on the remaining three variants. All experimental results are normalized, with our method set at 1 as the baseline to ensure comparability between metrics on different scales. Our method integrates group agent chain-of-thought reasoning together with the MWA and SCUS optimization tools. The w/o group setting removes the COT reasoning of the group agent; the w/o MWA+SCUS setting discards all optimization tools; and the w/o SCUS setting removes only SCUS while retaining MWA and group-agent reasoning.

As shown in Fig. 5, the results indicate that our method achieves the best performance in all benchmarks. In Ariane136, the removal of the group agent degrades WNS by up to 6×, while the removal of MWA + SCUS produces better results than the removal of the group agent, but still suffers significant degradation. Similar trends are observed in other benchmarks, where the w/o SCUS method shows some improvement over w/o MWA+SCUS.

As shown in Fig. 6, the placement results clearly explain the performance degradation. The yellow regions indicate that OpenROAD produces large unusable notch areas A_{notch}, which fragment the standard cell regions and reduce overall utilization. In the w/o group method, the absence of grouping constraints causes macros to pile up, leading to severe congestion and increased timing delay. In the w/o MWA+SCUS method, strongly connected modules (A and B) fail to remain adjacent, forcing interconnects to become unnecessarily long and significantly increasing critical path delay. Although w/o SCUS produces reasonable placement, the lack of layout quality evaluation results in insufficient compactness, reducing utilization and limiting PPA improvements.

VI. Conclusion

In this paper, we propose CHIP-MAP, a fully automated multi-agent framework for macro placement that leverages large language models to coordinate four specialized agents with MWA and SCUS evaluators, achieving iterative layout optimization and consistent PPA improvements on OpenROAD benchmarks. Through standardized agent I/O and coordinate output support, the framework enables plug-and-play integration with industrial EDA flows. This study is the first to validate the effectiveness of large language models in macro placement with good generalization across different designs.

REFERENCES

[1] S. Garg and N. K. Shukla, "A study of floorplanning challenges and analysis of macro placement approaches in physically aware synthesis," *Int. J. Hybrid Inf. Technol.*, vol. 9, no. 1, pp. 279–290, 2016.

[2] H. Pan, C. Lan, Y. Liu, Z. Wang, L. Shang, X. Zeng, F. Yang, and K. Zhu, "Physically aware synthesis revisited: guiding technology mapping with primitive logic gate placement," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des. (ICCAD)*, 2024, pp. 1–9.

[3] Y. Zhao, L. Zou, and B. Yu, "Invited: Physical design for advanced 3D ICs: Challenges and solutions," in *Proc. Int. Symp. Physical Design (ISPD)*, 2025, pp. 209–216.

[4] I. L. Markov, J. Hu, and M.-C. Kim, "Progress and challenges in VLSI placement research," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des. (ICCAD)*, 2012, pp. 275–282.

[5] A. B. Kahng, "Solvers, engines, tools and flows: The next wave for AI/ML in physical design," in *Proc. Int. Symp. Physical Design (ISPD)*, 2024, pp. 117–124.

[6] I.-L. Tseng, "Challenges in floorplanning and macro placement for modern SoCs," in *Proc. Int. Symp. Physical Design (ISPD)*, 2024, pp. 71–72.

[7] C. K. Cheng, A. B. Kahng, S. Kundu, Y. Wang, and Z. Wang, "Assessment of reinforcement learning for macro placement," in *Proc. Int. Symp. Physical Design (ISPD)*, 2023, pp. 158–164.

[8] P. Liao, S. Liu, Z. Chen, W. Lv, Y. Lin, and B. Yu, "DREAMPlace: Timing-driven global placement with momentum-based net weighting," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, 2022, pp. 939–944.

[9] Y. Lai, J. Liu, Z. Tang, B. Wang, J. Hao, and P. Luo, "ChipFormer: Transferable chip placement via offline decision transformer," *arXiv preprint arXiv:2306.14744*, 2023.

[10] J. Pan, G. Zhou, C. C. Chang, et al., "A survey of research in large language models for electronic design automation," *ACM Trans. Design Autom. Electron. Syst.*, in press.

[11] Z. Wang, Y. Shen, X. Yao, et al., "ChatPattern: Layout pattern customization via natural language," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, 2024, pp. 1–6.

[12] B. Liu, H. Zhang, X. Gao, et al., "LayoutCopilot: An LLM-powered multi-agent collaborative framework for interactive analog layout design," *arXiv preprint arXiv:2406.18873*, 2024.

[13] C. Luo, Y. Shen, Z. Zhu, et al., "LayoutLLM: Layout instruction tuning with large language models for document understanding," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2024, pp. 15630–15640.

[14] Y. Lai, Y. Mu, and P. Luo, "MaskPlace: Fast chip placement via reinforced visual representation learning," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 35, pp. 24019–24030, 2022.

[15] C. T. Ho and H. Ren, "Large language model (LLM) for standard cell layout design optimization," *arXiv preprint arXiv:2406.06549*, 2024.

[16] OpenROAD, "OpenROAD," [Online]. Available: https://github.com/The-OpenROAD-Project/

[17] OpenAI, "GPT-4o," [Online]. Available: https://openai.com/index/hello-gpt-4o

[18] DeepSeek, "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via reinforcement learning," [Online]. Available: https://www.deepseek.com/

[19] A. Vidal-Obiols, et al., "RTL-aware dataflow-driven macro placement," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, 2019, pp. 924–929.

[20] Z. Wang, Z. Geng, Z. Tu, et al., "Benchmarking end-to-end performance of AI-based chip placement algorithms," *arXiv preprint arXiv:2407.15026*, 2024.