Divergent Thoughts toward One Goal: LLM-based Multi-Agent Collaboration System for Electronic Design Automation

Haoyuan Wu[♠] **Haisheng Zheng**[♡] **Zhuolun He**[♠,♣] **Bei Yu**[♠]

[♠] The Chinese University of Hong Kong, Hong Kong SAR [♡] Shanghai Artificial Intelligent Laboratory, China [♣] ChatEDA Tech, China

Introduction

Background:

Electronic design automation (EDA) is indispensable for the design of integrated circuits (ICs). EDA tools are integrated into a complex design flow and utilize programming interfaces to control the design process. EDA platforms such as OpenROAD and iEDA, consist of complex procedures with various configurations.

Challenge:

- Although LLMs excel at understanding natural language, they lack specialized knowledge of EDA tool usage.
- Errors may occur in intermediate steps during a long-chain tool-calling process, introducing instability into the EDA flow automation.

Contribution:

- We develop the ChipLlama-powered agent, collaborating with few-shot CoT prompts, to develop the performance and portability of the single-agent system;
- Propose EDAid, a multi-agent system that collaborates multiple agents including divergent-thoughts agents and a decision-making agent for EDA flow automation;
- Perform extensive evaluations, which demonstrate the SOTA performance of ChipLlama models, the effectiveness of the few-shot CoT prompting, and the superior performance of our EDAid for EDA flow automation.

ChipLlama-powered Agent

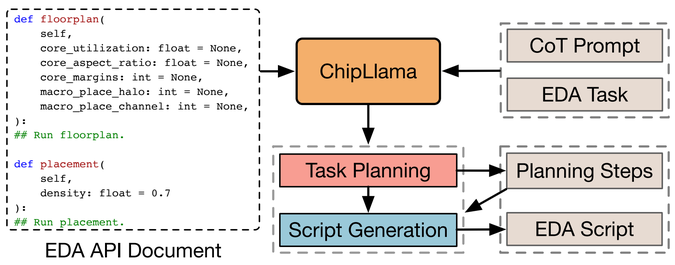

Figure 1. Overview of the ChipLlama-powered agent for task planning and EDA script generation.

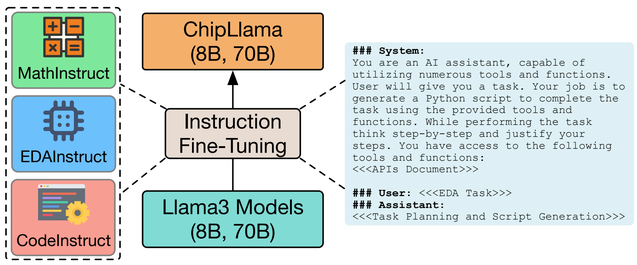

Figure 2. Overview of hybrid instruction tuning.

```
### System: You are an AI assistant, capable of utilizing numerous tools
and functions. User will give you a task. Your job is to generate a Python
script to complete the task using the provided tools and functions. While
performing the task think step-by-step and justify your steps. You have
access to the following tools and functions:
<<<APIs Document>>>

### User: <<<EDA Task 1>>>
### Assistant: <<<Solution 1>>>

### User: <<<EDA Task 2>>>
### Assistant: <<<Solution 2>>>

### User:
For the "aes" circuit, I want to run the steps from setup to detailed
routing on the platform "asap7"?
Let's first describe and explain what the task is asking. Then, analyze how
to complete the task step by step using the provided tools and functions.
Finally, generate the Python script according to your analysis.
### Assistant:
```

Figure 3. Few-shot CoT prompt template utilized in ChipLlama-powered agent.

Multi-Agent Collaboration System

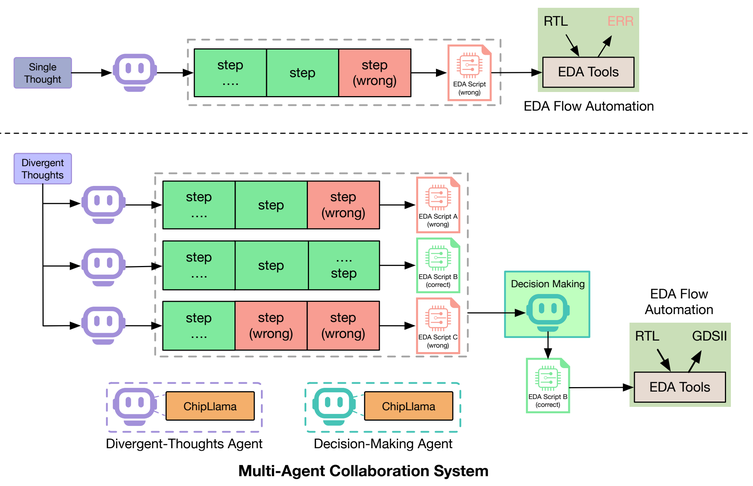

Figure 4. Overview of EDAid, the multi-agent collaboration system. Given an EDA task, multiple agents (including divergent-thoughts agents (role R_0) and a decision-making agent (role R_1)) collaborate to generate the EDA script. Finally, the generated EDA script will automate the EDA flow interfacing the EDA tools via APIs.

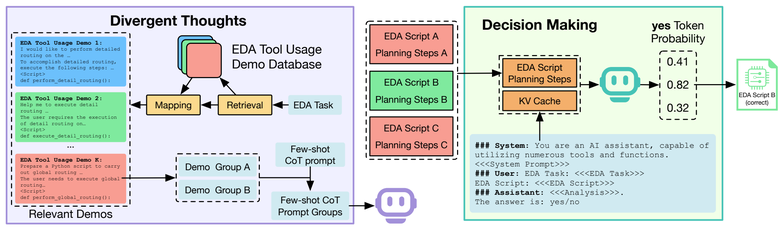

Figure 5. Divergent-thoughts agents (role R_0) and decision-making agent (role R_1)).

Evaluation Benchmarks

We utilize a comprehensive evaluation benchmark ChatEDA-bench and iEDA-bench, both of which comprise 50 distinct tasks, to evaluate the performance of our EDAid and ChipLlama models. ChatEDA-bench and iEDA-bench use the accuracy of the generated EDA script as the evaluation metric.

Both benchmarks are comprehensive evaluation benchmarks comprising 50 distinct tasks including three distinct categories: simple flow calls (30%), complex flow calls (30%), and parameter flow calls (40%).

- Simple flow call requires the successful execution of the whole process, including evaluation.
- Complex flow call requires a higher proficiency in EDA tool usage, including traversing parameters.
- Parameter flow call requires agent systems to provide a parameter-tuning solution.

Main Experiment Results

System	Powered LLM	ChatEDA-bench	iEDA-bench
ChatEDA	GPT-3.5[♢]	28%	30%
ChatEDA	GPT-4[♢]	62%	70%
ChatEDA	AutoMage-70B[♢]	74%	-
ChatEDA	AutoMage2-70B[♢]	82%	-
EDAid	ChipLlama-8B	88%	84%
EDAid	ChipLlama-70B	100%	100%

[♢] The accuracy values of GPT-3.5, GPT-4 and AutoMage models on the ChatEDA-bench are directly cited from the ChatEDA. Moreover, we can only evaluate AutoMage models on the ChatEDA-bench due to the unavailability of closed-source models.

Table 1. The main results of EDA script generation on ChatEDA-bench and iEDA-bench.

Ablation Studies

System	Base LLM	Hybrid Instruction Tuning	ChatEDA-bench	iEDA-bench
Single-Agent	Llama3-8B	✗	78%	50%
Single-Agent		✓	78%	76%
Single-Agent	Llama3-70B	✗	88%	74%
Single-Agent		✓	94%	96%

Table 2. Ablation study on hybrid instruction tuning.

System	Powered LLM	ChatEDA-bench		iEDA-bench	
		zero-shot	few-shot	zero-shot	few-shot
Single-Agent	GPT-3.5	28%	56%	30%	50%
Single-Agent	GPT-4	62%	82%	70%	84%
Single-Agent	ChipLlama-8B	74%	78%	64%	76%
Single-Agent	ChipLlama-70B	90%	94%	90%	96%

Table 3. Ablation study on few-shot prompting and powered LLMs of the single-agent system.

System	Powered LLM	ChatEDA-bench	iEDA-bench
Single-Agent	ChipLlama-8B	78%	76%
Multi-Agent		88%	84%
Single-Agent	ChipLlama-70B	94%	96%
Multi-Agent		100%	100%

Table 4. Ablation study on single/multi-agent systems powered by different LLMs.

Case Study

EDA Task

```
Can you help me check whether the provided script can complete the task and
provide the reason.
Here is the task:
I want to perform cts for my design asjdk on nangate45.
Here is the script:
def perform_cts(tns_end_percent):
    eda = chateda()
    # Setup EDA tool
    eda.setup(design_name="asjdk", platform="nangate45")
    # Run logic synthesis
    eda.run_synthesis()
    eda.floorplan()
    # Run placement
    eda.placement(tns_end_percent=tns_end_percent)
    # Run CTS
    eda.cts()
perform_cts(10)
If the script can't complete the task, please provide a script that can complete
the task.
```

Task Planning Pathway

```
The task is to perform Clock Tree Synthesis (CTS) for the design named "asjdk" on
the "nangate45" platform. The provided script attempts to do this, but it has an
error in the placement step. The "placement" method does not have a parameter
called "tns_end_percent". The "tns_end_percent" is actually a parameter for the
"cts" method.
Here's the correct way to complete the task:
1. Setup the EDA tool with the design name and platform.
2. Run logic synthesis.
3. Perform floorplanning.
4. Run placement.
5. Run CTS with the desired "tns_end_percent".
```

EDA Script Generation

```
def perform_cts(tns_end_percent):
    eda = chateda()
    # Setup EDA tool
    eda.setup(design_name="asjdk", platform="nangate45")
    # Run logic synthesis
    eda.run_synthesis()
    # Run floorplan
    eda.floorplan()
    # Run placement
    eda.placement()
    # Run CTS
    eda.cts(tns_end_percent=tns_end_percent)
perform_cts(10)
```

Nations of the Americas Chapter of the Association for Computational Linguistics, Albuquerque, New Mexico, Apr. 29–May 4, 2025.