Automated Op-Amp Netlist Generation Using Large Language Models

LLM_Sizer@Prime
Zichen Kong
Advised by Xiyuan Tang
Peking University

LLM_Sizer@Prime
Zichen Kong
Advised by Xiyuan Tang
Peking University

I. Introduction

This report presents a method for designing operational amplifiers (Op-Amps) using Large Language Models (LLMs). The dimensions of the components are determined by the LLM, which ensures that the design meets key performance specifications: a gain of 60 dB, bandwidth of 20 MHz, slew rate of 20 V/μs, phase margin of 60°, and a DC current of 500 µA. Meanwhile, the prompts should be as succinct as possible.

During the final round, an additional function of the model is that the agent should be able to refine the netlist according to high-level instructions to the improvement on a specific metric. This event also requires deploying the Qwen2.5 series of LLMs on a server equipped with the Yitian 710 CPU and invoking them via Ollama. The final runtime performance will also be factored into the scoring criteria.

The process begins with the generation of an initial SPICE netlist, based on estimated values for multipliers and capacitors. To meet the second requirement of improving specific metric, this rough design is then simulated to evaluate performance. The simulation results provide feedback, which is then used to adjust the dimensions in the next iteration. The design will be iteratively refined using prompt engineering to adjust the design toward the required specifications.

Based on the above, this work presents several technical challenges, namely

a) **Topology understanding:** LLMs cannot directly understand the topology and functions of components from a SPICE netlist. Such information should be provided in an effective method.

b) **Excessive Runtime:** Running large models locally for inference incurs significant time costs, and invoking the simulator further adds substantial overhead. However, both steps are indispensable for task completion.

c) **Complexity of Metric-Specific Optimization:** The relationship between a circuit's performance metrics and its component parameters is often non-trivial, making the task of optimizing for specific targets inherently complex. This necessitates a meticulously designed and computationally efficient execution framework.

In light of these challenges, we propose:

a) **Topology with Highly Reusable Parameters:** We employ a structurally simple, highly symmetric circuit topology to maximize parameter reuse from the LLM, thereby reducing design complexity. Our implementation adopts a basic two-stage op-amp configuration consisting of a simple bias current source with Miller compensation.

b) **Efficient Simulation-based Iteration:** The whole process includes iteration based on simulation results. During the iteration, the LLM receives feedback from previous simulations, and only adjust the value of determinant parameters as instructed, thus ensuring the efficiency of iteration.

c) **Runtime Efficiency Optimization:** For initial netlist generation, we enable the large model to directly produce viable designs without simulator intervention. By having the model generate parameter assignment statements rather than complete netlist code, we significantly reduce runtime overhead. Furthermore, we employ a lightweight model variant for inference and optimize local execution through parameter compression settings. Regarding the metric-specific design, we make the agent call low-cost simulations and only adjust limited parameters to keep the circuit functional while improving toward the given limitations.

II. Methods

This section outlines the methodology used to design operational amplifiers (Op-Amps) using Large Language Models (LLMs) to generate SPICE netlists. The process consists of several stages: selection of the circuit topology, initial netlist generation, iterative design refinement, and efforts to compress runtime. The steps will be introduced in the following section.

- **A. Topology Selection**

The core Op-Amp circuit topology we use is a two-stage operational amplifier, with a self-biased current mirror load and input pairs forming 5-transistor OTA in the first stage and a common source transistor in the second stage. A capacitor is added in the second stage for Miller compensation. This two-stage topology is well-suited for reusing design parameters with its highly symmetric first stage topology and great tolerance to operating points shifting. Once the transistors have their relative sizes properly assigned and work in saturation region, the whole amplifier would achieve a acceptable performance. The relative size of transistors can be discriminated by letting the LLM design the multipliers of the transistors, which is a more effective and straightforward way since it spares the effort to calculate design gate widths directly and can be easily understanded by the language model agent.

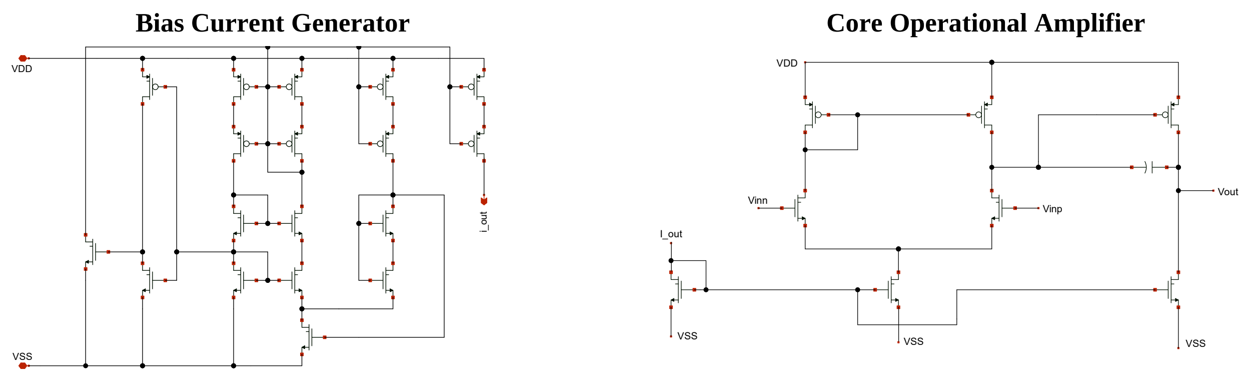

Fig1. The topology of our circuit, consisting of a bias current generator and a core amplifier

The circuit also includes a bias current generator as required. We also choose a design-friendly topology for this part. By setting initial gate lengths fixed, the parameters needed to be designed are reduced to fewer than 15.

Parameters need to be designed	
Parameter name	Number
Gate length (initial value)	2
Gate width (initial value)	2
Multiplier & W/L ratio	7
Capacitor	1

As shown in the table, only 12 parameters need to be generated for the initial netlist thanks to the highly symmetric circuit topology. The schematic of our circuit is shown in **Fig1**.

- **B. Prompt engineering and settings**

The whole design process starts with generating a roughly designed netlist. This initial netlist should be not only easily achievable but also functional with acceptable simulation results. Consequently, we propose some techniques in prompt structure and language usage in this section.

a) Parameter Estimation Using Differentiated Adjectives: To obtain well-differentiated initial parameters, adjectives such as "small", "medium", and "large" are used to describe the size of components like gate lengths (L) and multipliers. This step-by-step approach helps generate stable and meaningful answers from the LLM, ensuring clear distinctions in the parameters.

b) Accurate Data Entry and Calculation with Parameter Control: Tasks that require high accuracy, such as generating reference dimensions, simple calculations, and entering data into a fixed format, are broken down into smaller steps. These smaller tasks are handled separately by the LLM with a temperature setting in the range of 0 to ensure stable and reliable results.

C. Iterative Design Refinement

After the generation of an initial design, the model will refine the sizing through an iterative process. In this section, we will outline the agent assignments and the iteration framework used for optimization. The framework of the iteration-agent is shown in **Fig 2**.

The agents aiming at metric-specific design have slightly different structure regarding the metric specified. But their backbone structures are similar and we will only introduce one of them here to illustrate how the iterative process is carried out.

a) Simulation and Feedback: During each iteration, the design is simulated, and performance metrics (e.g., gain, bandwidth, slew rate, phase margin) are evaluated. The LLM agents use feedback from these simulations to adjust component dimensions. Note that only low-cost simulations (simulation concerning the specified metric and under TT conner) is called to enhance time efficiency.

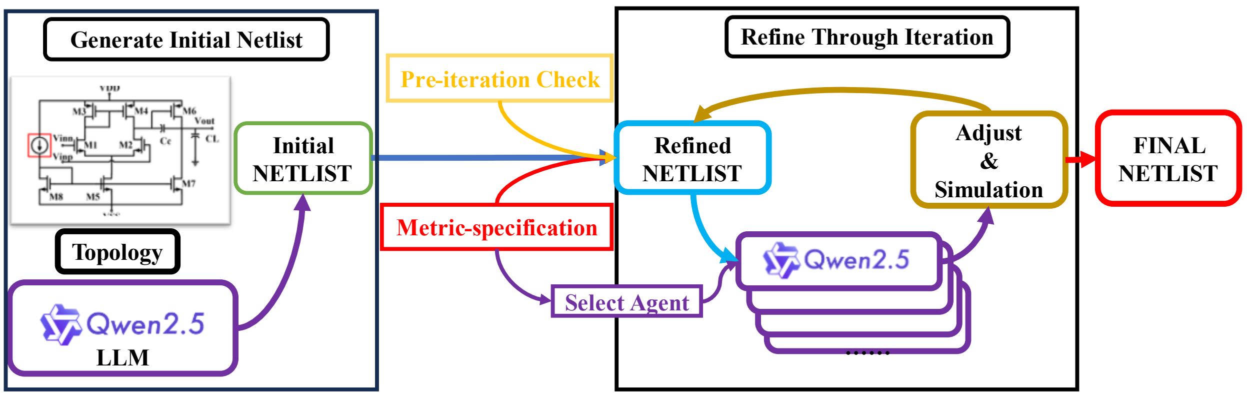

Fig 2. The overall framework of the iteration agent

b) Iteration Process: The design is iterated by calling the relevant agents based on simulation feedback. For example, if the Gain-bandwidth production is insufficient, the agents responsible for adjusting the miller capacitor will be activated. The process continues until either the desired performance metrics are reached or the maximum iteration limit is reached.

c) Stopping Criteria and Pre-iteration Checks: While our adopted circuit topology offers considerable flexibility, its optimization range for individual performance metrics remains inherently constrained. When receiving optimization instructions for a specific metric, the agent first conducts preliminary tests to determine whether the target can be achieved through parameter adjustments within its current topological framework. For instance, when optimizing slew rate (SR), the agent initially estimates the theoretical maximum SR under given current conditions by temporarily removing the Miller capacitor, before proceeding with iterative refinement. Although this approach cannot achieve unbounded optimization for all metrics simultaneously, it delivers substantial optimization capability for individual targets.

This iterative adjustment process can be formalized in the following pseudocode:

```
GBW Optimization Algorithm Workflow

run Topological Feasibility Check
   Calculate the theoretical maximum GBW
If target GBW exceeds limit:
      enlarge the first-stage bias current to modify the topology
run Parameter Search Initialization
while (metric specification not met):
     Run simulation with current parameters
     Extract measured GBW value
     If measured GBW is within tolerance of target:
         exit and return success
     If measured GBW > target:
         Set current parameter as new upper bound, search lower half
     If measured GBW < target:
         Set current parameter as new lower bound, search upper half
End of Iteration
```

• D. Runtime Efficiency Optimization

To further optimize runtime efficiency, we implemented optimizations at both the large model interaction level and deployment architecture. These enhancements ensure we can obtain required responses with minimal input/output tokens while maximizing utilization of server computational resources to accelerate model inference speed.

a) Optimization in LLM Interaction: To further enhance runtime efficiency while maintaining output quality, we strategically transitioned from the Qwen-7B model to more compact variants in the Qwen2.5 series, specifically adopting the 1.5B and 0.5B parameter models. This architectural shift was carefully implemented through a combination of model optimization techniques and intelligent task decomposition. By breaking down complex problems into manageable sub-tasks and implementing token-level prompt optimization, we achieved significant performance improvements without compromising output stability. Our evaluations demonstrated that these smaller models deliver 3.2× faster inference speeds compared to the 7B baseline while maintaining remarkable output consistency, showing less than 5% variance in controlled benchmarks. The solution proved particularly effective in compute-intensive scenarios like circuit parameter optimization, where it simultaneously reduced memory footprint by 78% and maintained crucial numerical precision.

b) Optimization in LLM Deployment:

The competition requirements mandated using Yitian 710 CPU servers with Ollama for model inference. While the prescribed interaction framework limited optimization opportunities, we implemented process monitoring to evaluate CPU utilization efficiency. Our benchmarks revealed the server's CPU specifications as follows:

CPU resource	
Architecture	Arm V9 64bit
Cores	16 physical cores, no hyper-threading
Frequency	3.0 GHz
L1 cache	1 MiB L1d & 1MiB L1i
L2 cache	16 MiB
L3 cache	64 MiB

Based on the above and the test results presented by Zhaode Wang at AICAS Grand Challenge 2025 Hangzhou Workshop[i], we predict that the 16-core Yitian 710 chip, without quantization, will achieve a theoretical decoding-stage performance limit of ~300 tokens/s for the Qwen2.5 0.5B model and ~100 tokens/s for the 1.5B model. By pre-launching the Ollama service and loading the model in advance, we can partially mitigate performance bottlenecks caused by model loading and platform initialization processes.

III. Results

Through our design approach, we have successfully developed a model capable of generating a functional two-stage operational amplifier (Op-Amp) netlist. Based on the competition scoring criteria, the model produced a design that is capable of meeting the basic limits.

a) Initial Netlist Generation: First, we test the initial netlist generated by the agent under five different corners *(TT, SS, FF, SF, FS)*. According to the criteria of the Grand Challenge, the score will be calculated based on the worst performance of each metric under the aforementioned five test corners.

The key parameters of the initially generated netlist are as follows (the worst value among five simulation corners):

Metric	**Value**	**Additional Metrics**	
Gain	*71.6 dB*	**Metric**	**Value**
GBW	*29.3 MHz*	***Noise***	*0.43 uV*
PM	*55.6 deg*	***CMRR***	*77.2 dB*
SR	*50.9 V/us*	***PSRR***	*74.0 dB*
Idc	*131.7 uA*		

According to simulation results, the initially designed circuit works with a first stage current of around *14 uA* and the second stage accounting for *114 uA*. The bias circuit costs only about *1 uA* under TT corner. With the second stage having a *gm* of *160 us,* about ten times larger than the first stage load, the miller capacitor can achieve a relatively small value of *0.6 pF*, only 0.3 times the value of load capacitor. This will leave a sufficient optimization room for gain-bandwidth production and slew rate.

During the test, this design shows reasonable tolerance to different PVT conditions and is not complicated to size just as mentioned before. Besides, the topology can be generated by the language model almost without error. The accuracy can reach *90%* in a test of ten rounds.

b) Metric-specified Optimization: Regarding the metric-specified optimization stage, we propose several adjustments to our circuit format in order to cater the need for moving toward different design specifications.

Through careful circuit topology refinement, we enhanced architectural flexibility to enable a much wider optimization ranges for individual performance metrics. However, the circuit's performance regarding other metrics cannot be ensured when a too drastic optimization is needed.

The table below demonstrates the achievable optimization bounds across key metrics within 20 simulation iterations:

Metric	Optimization Bounds*****
Gain	*45 dB~92 dB*
GBW	*12 MHz~108 MHz*
SR	*10 V/us~504 V/us*
PM	*< 90deg*
Idc	*>20 uA*

(*: These values are the limits of individual metric regardless of whether the circuit is functioning well or not)

c) LLM Deployment: The complete system was executed on a server equipped with the Yitian 710 CPU, achieving a total runtime of 2.7 seconds. The runtime breakdown revealed that the Python orchestration layer accounted for 0.5 seconds (18.5% of total runtime), while model inference constituted the majority of computational overhead. Specifically, the Qwen2.5-0.5B model completed inference in 0.4 seconds with a throughput of 190 tokens/s, and the Qwen2.5-1.5B model required 1.7 seconds at 70 tokens/s. A residual 0.1 seconds was due to Ollama's communication protocol and I/O processing latency.

Notably, the observed throughput metrics deviated slightly from pre-deployment predictions, a discrepancy likely caused by Ollama's invocation protocol overhead and suboptimal token batching in the current I/O pipeline. Despite these minor variances, the deployment demonstrated robust performance characteristics: memory utilization remained stable for both model variants, total runtime is kept low, indicating efficient hardware resource allocation.

These results validate the feasibility of deploying lightweight LLMs for time-sensitive electronic design automation (EDA) tasks. The system's performance aligns with requirements for rapid design space exploration and multi-agent optimization workflows, particularly in resource-constrained environments.

[i] Zhaode Wang, *Practice of Optimizing LLM Inference on the Device based on MNN*, Hangzhou Workshop@2025 IEEE AICAS Grand Challenge