# Summer Fellowship Programme 2025, IIT Madras

Department of Electrical Engineering

**Submitted To-**

**Prof. Sankaran Aniruddhan**

# Design and Automation of Two Stage Operational Amplifiers

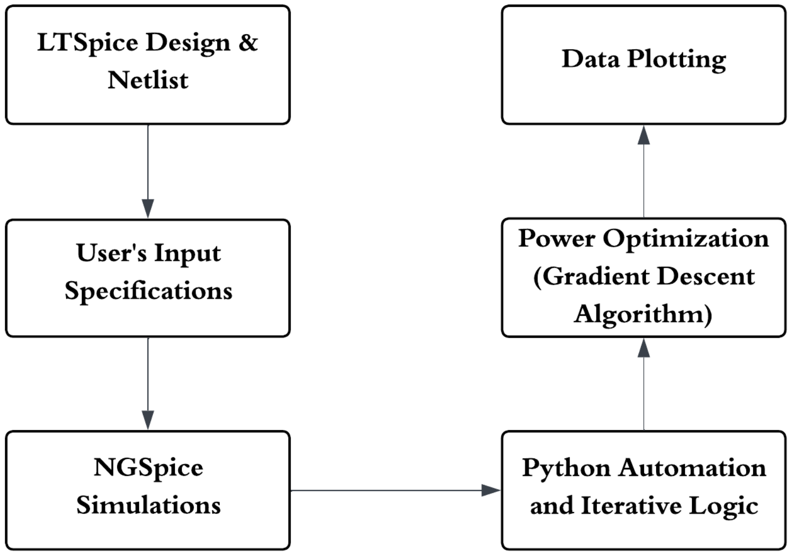
**By-**

Prajwal Singh

SF0005160-EEE

## 1 Introduction

- Manual transistor sizing in analog IC design is time-consuming and heavily dependent on designer intuition; this project develops an automated, specification-driven design flow for a two-stage CMOS operational amplifier to address that gap.
- Beginning with hand-designed and simulated topologies in LTspice—alongside study of telescopic and folded-cascode variants for trade-off analysis—the work implements a Python-based framework that translates high-level user requirements (DC gain, unity-gain bandwidth, phase margin, power budget, ICMR/OCMR) into initial device W/L ratios and passive component values.
- The framework integrates with NGSpice to run simulations from extracted netlists, analyzes bode responses, and iteratively refines device dimensions (principally transistor lengths) to converge on the target specifications.
- To further enhance the automation framework, we are working on incorporating a gradient-descent-based optimization module. This technique, guided by a configurable cost function, will enable users to assign relative importance to competing specifications, thereby facilitating power minimization while ensuring that prioritized design metrics are preserved.
- The project produces automated netlist modifications, simulation-validated performance plots, and a power-optimized sizing solution, demonstrating an end-to-end Python–NGSpice automation flow that bridges circuit-level understanding with algorithmic optimization and offers a scalable foundation for extending the approach to more complex analog blocks.


LTSpice Design & Netlist
User's Input Specifications
NGSpice Simulations
Python Automation and Iterative Logic
Power Optimization (Gradient Descent Algorithm)
Data Plotting

## 2 Implementation

### 2.1 Design and Simulation in LTSpice

We started with learning about two stage operational amplifier and the associated design process and trade-offs. The calculations of various aspect ratios and passive component values was based on the following.

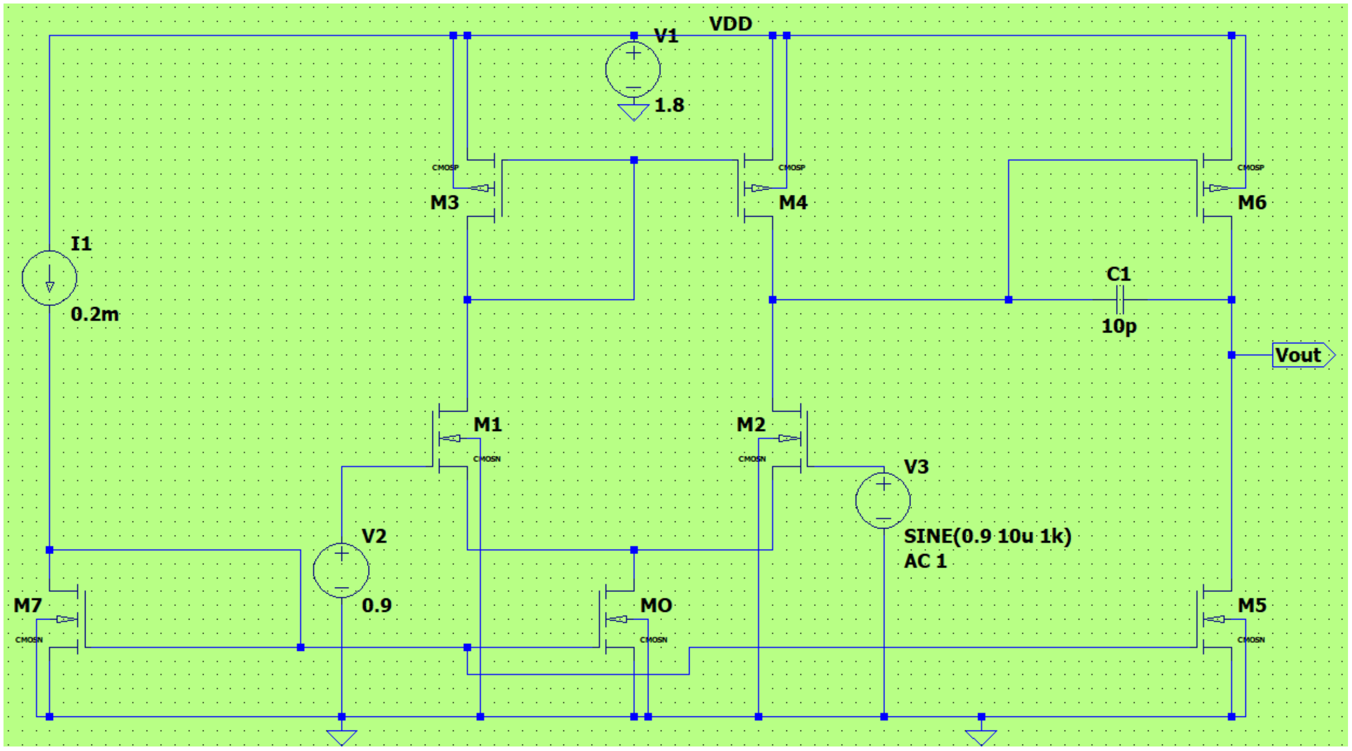


Figure 1: Two Stage OP-AMP Default Design

User Inputs the following values-

- Gain
- Unity Gain Bandwidth
- Phase Margin
- Power Budget
- Input Common Mode Range
- Output Common Mode Range

Now using the above values we do the following calculations, which here in this project is automatically done during runtime of the python program.

- $$I_{\text{available}} = \frac{P_{\text{budget}}}{V_{DD}} \tag{1}$$

- $$I_{\text{available}} = I_{\text{1st stage}} + I_{\text{2nd stage}} \tag{2}$$

- Now the ratio of the current in first stage and second depends on the requirements. We took it to be 1:10 which helps in a decent phase margin as well as meeting current requirements.

$$\frac{I_{\text{second stage}}}{I_{\text{first stage}}} = x = 10(here) \tag{3}$$

- $$OCMR_{\text{min}} = V_{\text{DSAT},M5} \text{ or } V_{\text{ov},M5} \tag{4}$$

$$\left(\frac{W}{L}\right)_{M5} = \frac{2I_2}{\mu_n C_{ox} \left(V_{DSAT,M5}\right)^2} \tag{5}$$

- As we can see, $V_{GS,M0} = V_{GS,M5}$.

$$\frac{I_1}{\left(\frac{W}{L}\right)_{M0}} = \frac{I_2}{\left(\frac{W}{L}\right)_{M5}} \tag{6}$$

$$\left(\frac{W}{L}\right)_{M0} = \frac{I_1}{I_2} \cdot \left(\frac{W}{L}\right)_{M5} \tag{7}$$

- $$OCMR_{\text{max}} = V_{DD} - V_{DSAT,M6} \tag{8}$$

$$\left(\frac{W}{L}\right)_{M6} = \frac{2I_2}{\mu_n C_{ox} \left(V_{DSAT,M6}\right)^2} \tag{9}$$

- As we can see, $V_{SG,M4} = V_{SG,M6}$.

$$\frac{I_1/2}{\left(\frac{W}{L}\right)_{M4}} = \frac{I_2}{\left(\frac{W}{L}\right)_{M6}} \tag{10}$$

$$\left(\frac{W}{L}\right)_{M3} = \left(\frac{W}{L}\right)_{M4} \tag{11}$$

- $$ICMR_{\text{min}} = V_{GS,M1} + V_{DSAT,M0} \tag{12}$$

$$V_{DSAT,M0} = V_{DSAT,M5} \tag{13}$$

$$\left(\frac{W}{L}\right)_{M1} = \left(\frac{W}{L}\right)_{M2} = \frac{I_1}{\mu_n C_{ox} \left(V_{GS,M1} - V_{Tn}\right)^2} \tag{14}$$

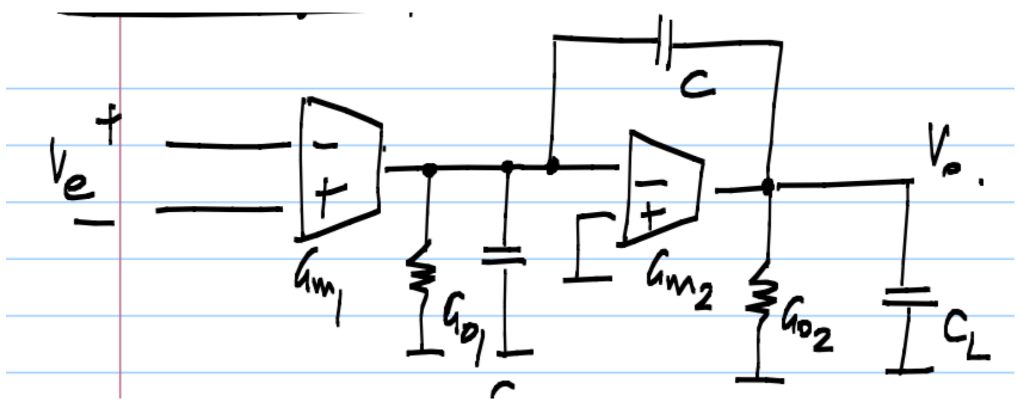


Figure 2: Source- Lecture 14, Analog IC Design, IIT Madras

- With Miller Compensation Capacitor C, pole1 (P1) shifts closer to the origin whereas pole2 (P2) shift further away from the unity gain bandwidth in order to replicate a first order response near the unity gain bandwidth.

$$P_1 = \frac{g_{o1}}{C_1 + C\,(1 + A_v)} \tag{15}$$

$$P_2 = \frac{g_{m2}\left(\frac{C}{C+C_1}\right) + g_{o2}}{\frac{C\,C_1}{C+C_1} + C_2} \tag{16}$$

$$Z_1 = \frac{g_{m2}}{C_c} \tag{17}$$

$$\omega_u = \frac{g_{m1}}{C_c} \tag{18}$$

$$\text{Phase Margin} = 90^\circ - \tan^{-1}\left(\frac{\omega_u}{P_2}\right) - \tan^{-1}\left(\frac{g_{m1}}{g_{m2}}\right) \tag{19}$$

To have a decent value of Phase margin i.e. greater than 60 degrees we need to have the location of P2 to be further away from wu, also the ratio of gm1 and gm2 should also be smaller. This is the reason we have I2/I1 to be 10 here.

## 3 Python Program and Iterative Logic

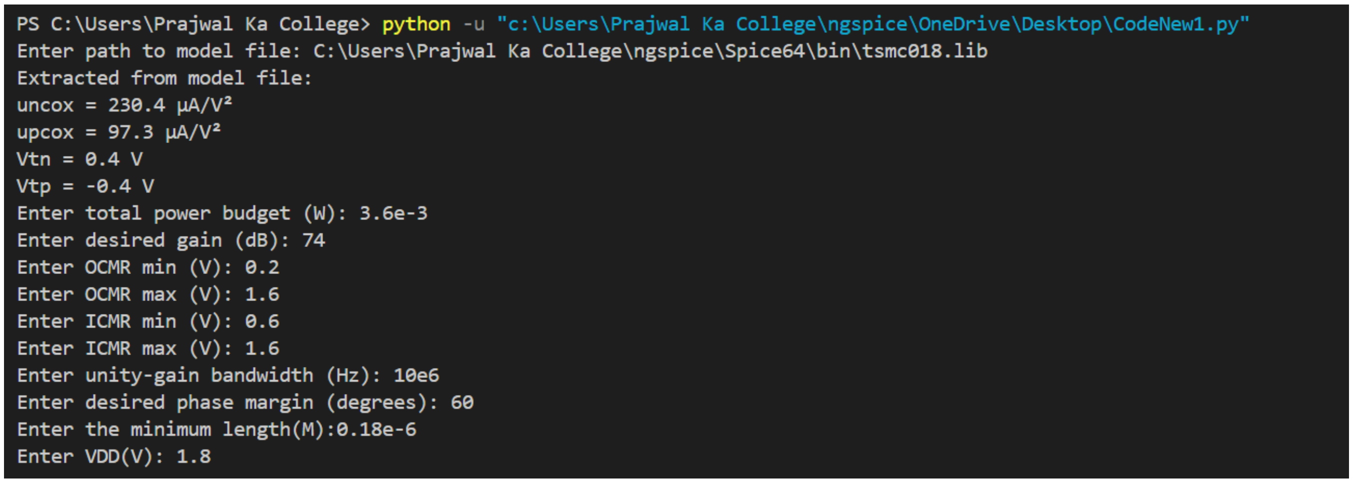
1. **Import Libraries**: Import required Python libraries such as `os`, `subprocess`, `pandas`, `numpy`, `matplotlib`, `re`, and `datetime`.
2. **Parameter Extraction from Model File**:
   - Define function to read the SPICE model file.
   - Extract NMOS/PMOS parameters (`u0`, `tox`, `vth0`) using regex.
   - Compute derived parameters: `uncox`, `upcox`, `cox`, `tox`, threshold voltages.
3. **Model File Input**: Prompt user for model file path. If valid, extract parameters and print.
4. **Specification Input**: Request user input for specs (power budget, gain, OCMR/ICMR, UGB, phase margin, minimum length, VDD).
5. **Initialization**: Initialize VDD, iteration count, minimum transistor lengths, and compensation capacitance.
6. **Log Parsing Functions**: Define functions for extracting `gm`, `gds`, node voltages, and threshold voltages from simulation log files.
7. **Main Iterative Optimization Loop**:
   - For up to 100 iterations, compute bias currents and W/L ratios.
   - Adjust W/L if below minimum and calculate transistor widths.
8. **Netlist Modification**:
   - Read original netlist.
   - Remove control block and update transistor parameters.
   - Update voltage/current sources and model references.
9. **Add Control Block for Simulation**:
   - Append AC analysis commands and parameter prints.
10. **Write Updated Netlist**: Save modified netlist to file.
11. **Run Simulation**:
    - Change directory and run NGSPICE.
    - Handle simulation errors.
12. **Extract Simulation Results**:

- Parse and print gm values, node voltages, and threshold voltages.

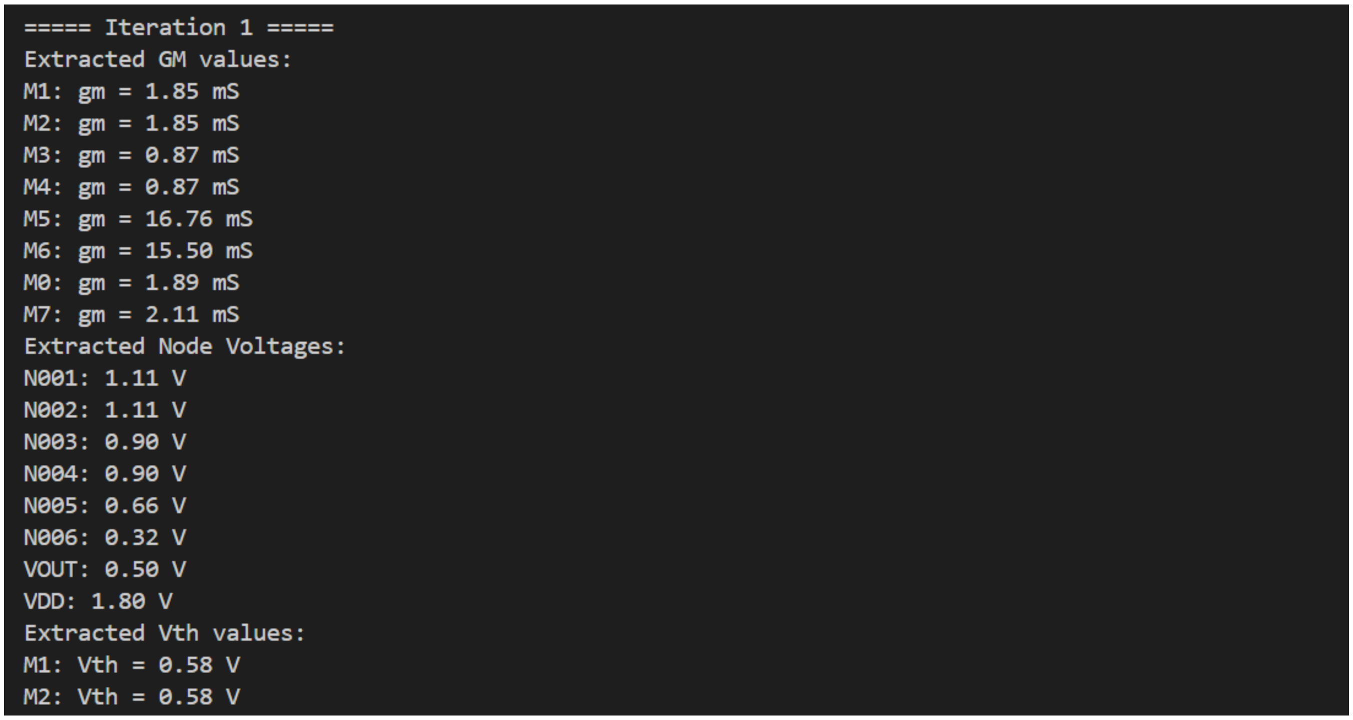
13. **Determine MOSFET Operating Regions**:
    - For each transistor, compute voltages.
    - Classify region (cutoff, saturation, triode).
14. **Extract gds and Update L Values for Balance**:
    - Get gds, balance differential and cascode pairs, update L.
15. **Process AC Data Output**:
    - Read AC data, calculate gain and phase margin.
    - Interpolate UGB and adjust compensation capacitance.
16. **Check Specifications**:
    - If gain, UGB, and phase margin are satisfactory, print results and plot graphs.
17. **Error Handling**:
    - Handle any file reading or processing errors.
18. **End Iterations**: Increment iteration or print failure message after maximum iterations.

## 4 Demonstration of Automated Flow

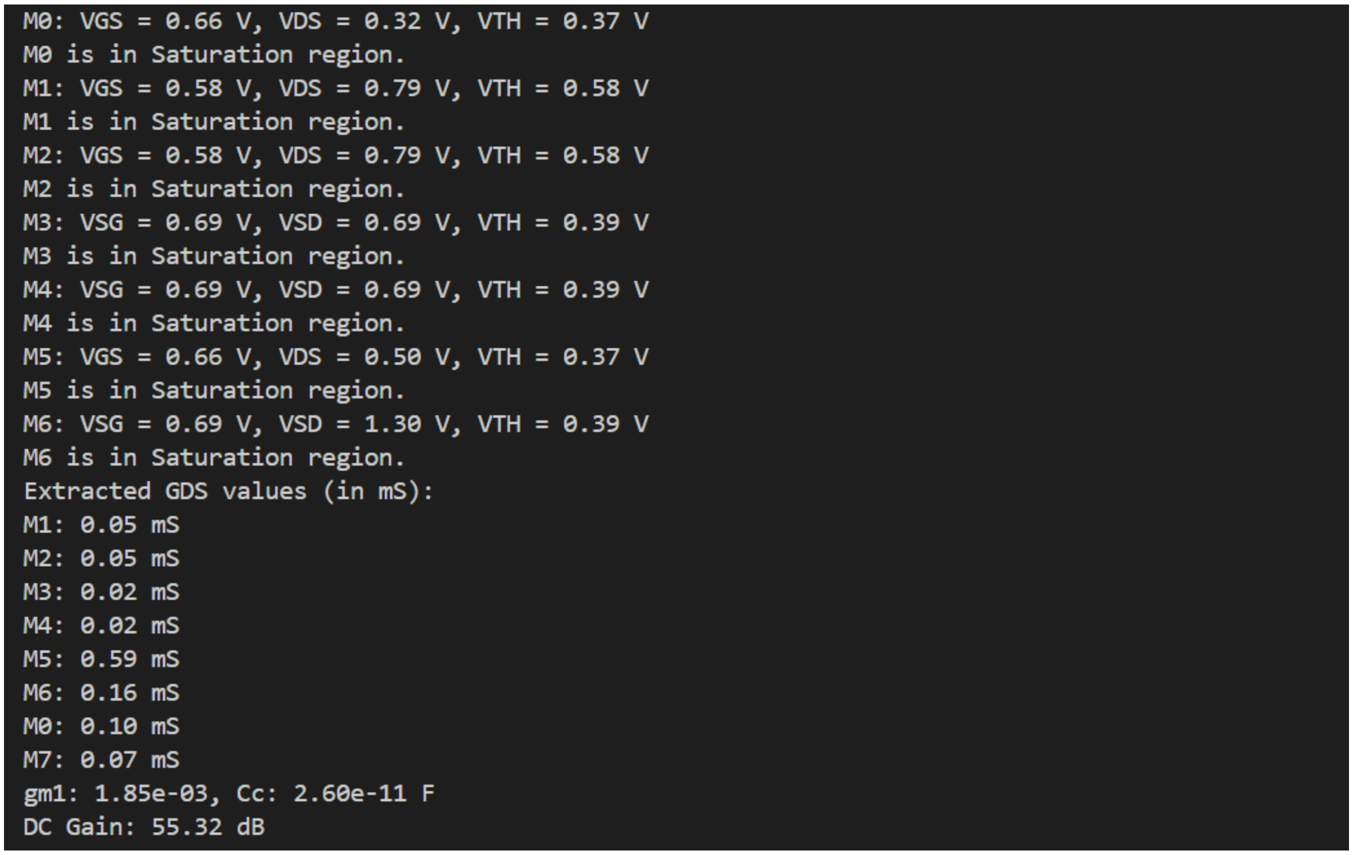
The following section presents a sample run for a given set of specifications. This example illustrates the execution of the automation framework and provides a clearer understanding of how the program processes user inputs, performs iterative sizing, and verifies the design through simulation.



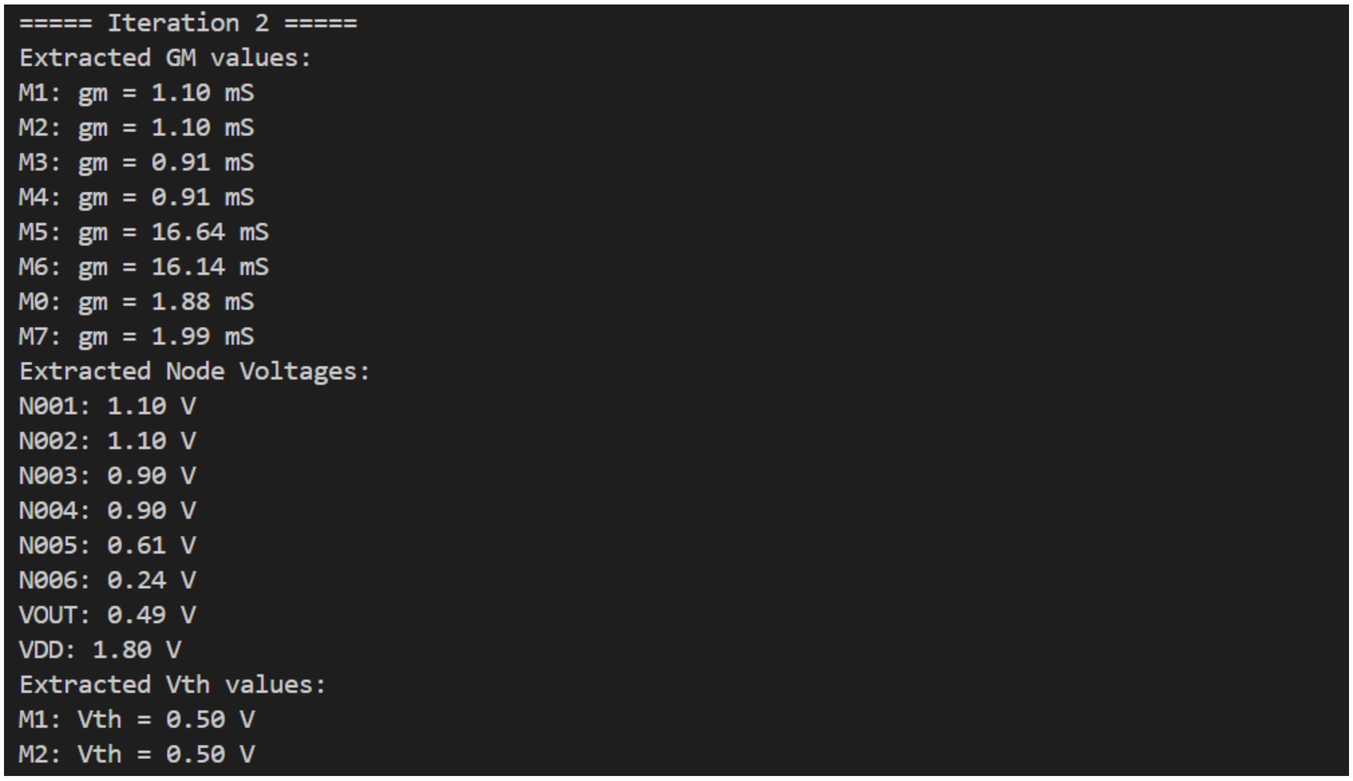
```
PS C:\Users\Prajwal Ka College> python -u "c:\Users\Prajwal Ka College\ngspice\OneDrive\Desktop\CodeNew1.py"
Enter path to model file: C:\Users\Prajwal Ka College\ngspice\Spice64\bin\tsmc018.lib
Extracted from model file:
uncox = 230.4 µA/V²
upcox = 97.3 µA/V²
Vtn = 0.4 V
Vtp = -0.4 V
Enter total power budget (W): 3.6e-3
Enter desired gain (dB): 74
Enter OCMR min (V): 0.2
Enter OCMR max (V): 1.6
Enter ICMR min (V): 0.6
Enter ICMR max (V): 1.6
Enter unity-gain bandwidth (Hz): 10e6
Enter desired phase margin (degrees): 60
Enter the minimum length(M):0.18e-6
Enter VDD(V): 1.8
```



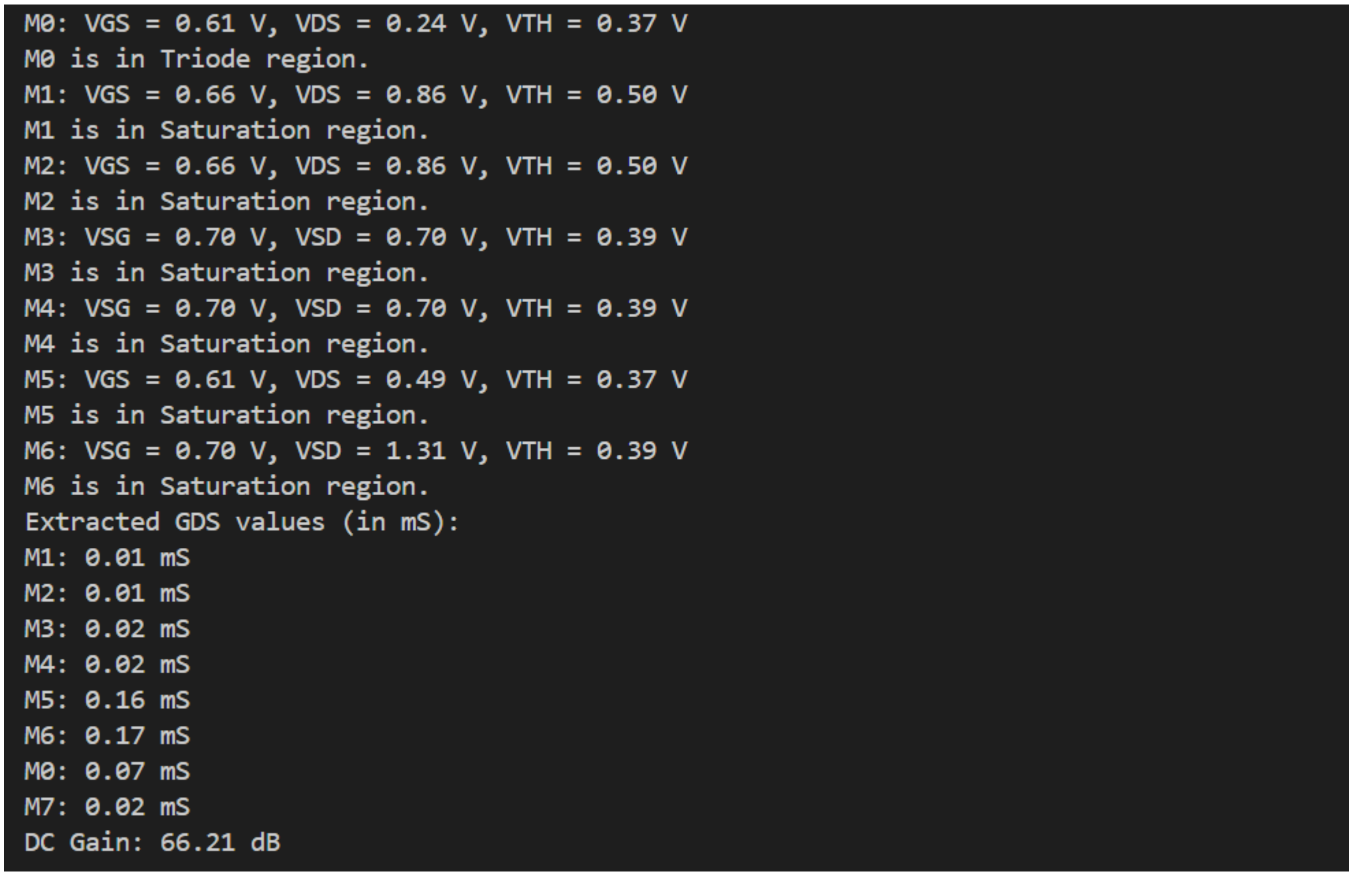
```
===== Iteration 1 =====
Extracted GM values:
M1: gm = 1.85 mS
M2: gm = 1.85 mS
M3: gm = 0.87 mS
M4: gm = 0.87 mS
M5: gm = 16.76 mS
M6: gm = 15.50 mS
M0: gm = 1.89 mS
M7: gm = 2.11 mS
Extracted Node Voltages:
N001: 1.11 V
N002: 1.11 V
N003: 0.90 V
N004: 0.90 V
N005: 0.66 V
N006: 0.32 V
VOUT: 0.50 V
VDD: 1.80 V
Extracted Vth values:
M1: Vth = 0.58 V
M2: Vth = 0.58 V
```



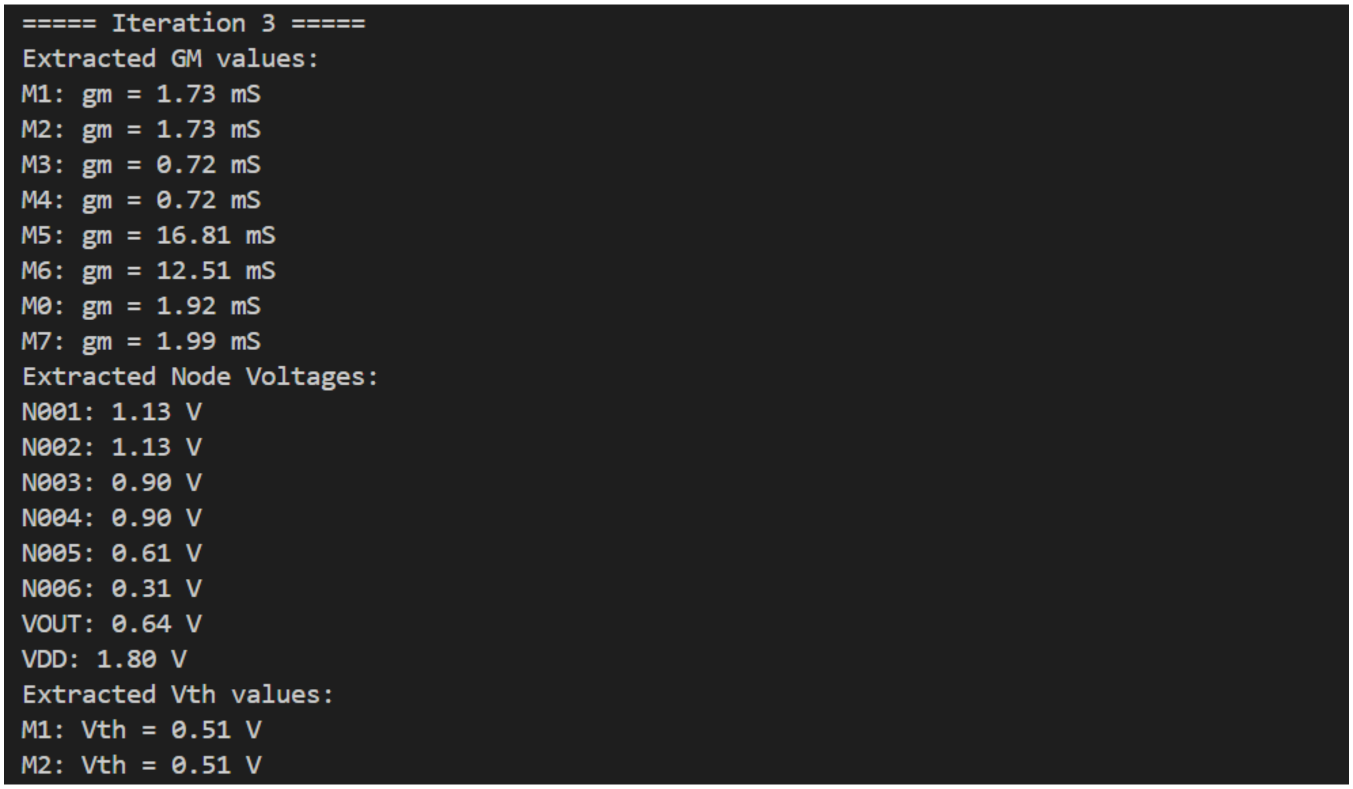
```
M0: VGS = 0.66 V, VDS = 0.32 V, VTH = 0.37 V
M0 is in Saturation region.
M1: VGS = 0.58 V, VDS = 0.79 V, VTH = 0.58 V
M1 is in Saturation region.
M2: VGS = 0.58 V, VDS = 0.79 V, VTH = 0.58 V
M2 is in Saturation region.
M3: VSG = 0.69 V, VSD = 0.69 V, VTH = 0.39 V
M3 is in Saturation region.
M4: VSG = 0.69 V, VSD = 0.69 V, VTH = 0.39 V
M4 is in Saturation region.
M5: VGS = 0.66 V, VDS = 0.50 V, VTH = 0.37 V
M5 is in Saturation region.
M6: VSG = 0.69 V, VSD = 1.30 V, VTH = 0.39 V
M6 is in Saturation region.
Extracted GDS values (in mS):
M1: 0.05 mS
M2: 0.05 mS
M3: 0.02 mS
M4: 0.02 mS
M5: 0.59 mS
M6: 0.16 mS
M0: 0.10 mS
M7: 0.07 mS
gm1: 1.85e-03, Cc: 2.60e-11 F
DC Gain: 55.32 dB
```



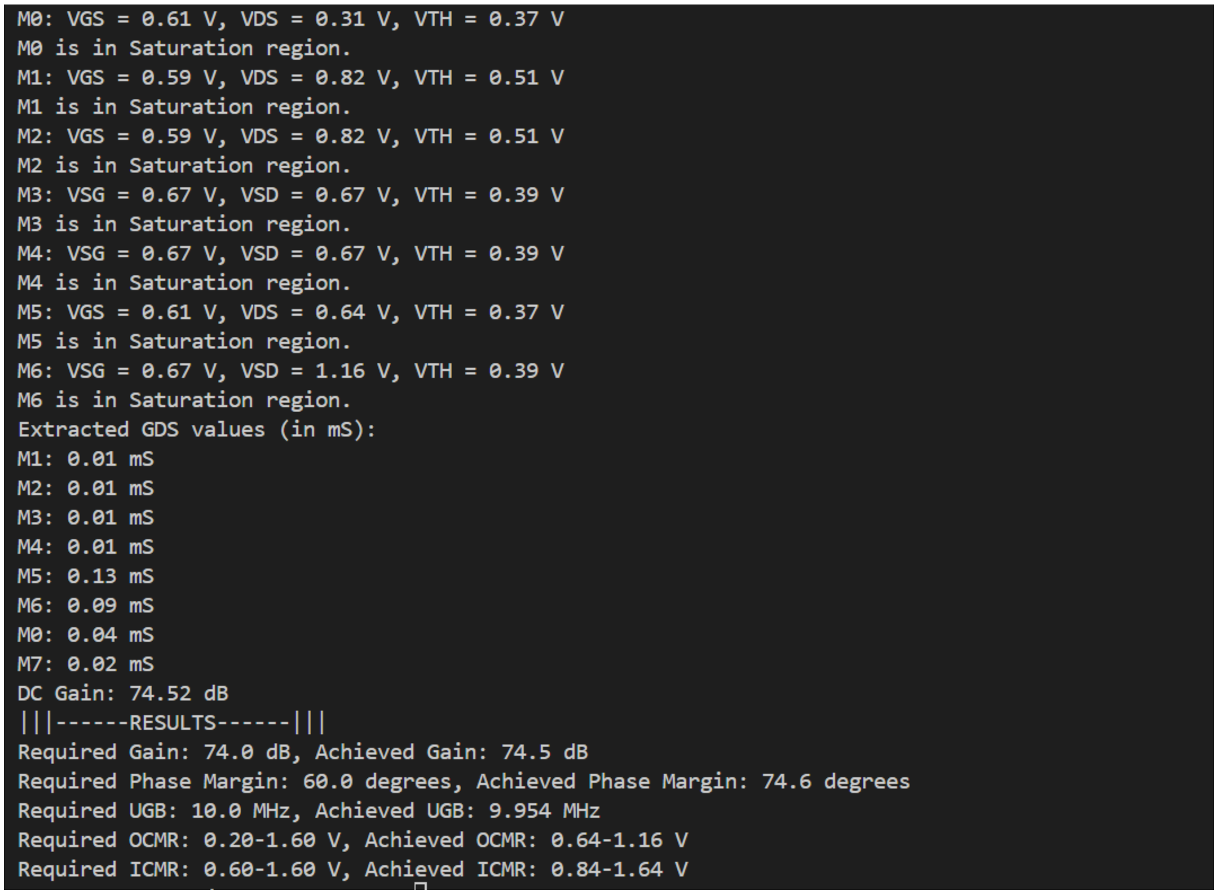
```
===== Iteration 2 =====
Extracted GM values:
M1: gm = 1.10 mS
M2: gm = 1.10 mS
M3: gm = 0.91 mS
M4: gm = 0.91 mS
M5: gm = 16.64 mS
M6: gm = 16.14 mS
M0: gm = 1.88 mS
M7: gm = 1.99 mS
Extracted Node Voltages:
N001: 1.10 V
N002: 1.10 V
N003: 0.90 V
N004: 0.90 V
N005: 0.61 V
N006: 0.24 V
VOUT: 0.49 V
VDD: 1.80 V
Extracted Vth values:
M1: Vth = 0.50 V
M2: Vth = 0.50 V
```



```
M0: VGS = 0.61 V, VDS = 0.24 V, VTH = 0.37 V
M0 is in Triode region.
M1: VGS = 0.66 V, VDS = 0.86 V, VTH = 0.50 V
M1 is in Saturation region.
M2: VGS = 0.66 V, VDS = 0.86 V, VTH = 0.50 V
M2 is in Saturation region.
M3: VSG = 0.70 V, VSD = 0.70 V, VTH = 0.39 V
M3 is in Saturation region.
M4: VSG = 0.70 V, VSD = 0.70 V, VTH = 0.39 V
M4 is in Saturation region.
M5: VGS = 0.61 V, VDS = 0.49 V, VTH = 0.37 V
M5 is in Saturation region.
M6: VSG = 0.70 V, VSD = 1.31 V, VTH = 0.39 V
M6 is in Saturation region.
Extracted GDS values (in mS):
M1: 0.01 mS
M2: 0.01 mS
M3: 0.02 mS
M4: 0.02 mS
M5: 0.16 mS
M6: 0.17 mS
M0: 0.07 mS
M7: 0.02 mS
DC Gain: 66.21 dB
```



```
===== Iteration 3 =====
Extracted GM values:
M1: gm = 1.73 mS
M2: gm = 1.73 mS
M3: gm = 0.72 mS
M4: gm = 0.72 mS
M5: gm = 16.81 mS
M6: gm = 12.51 mS
M0: gm = 1.92 mS
M7: gm = 1.99 mS
Extracted Node Voltages:
N001: 1.13 V
N002: 1.13 V
N003: 0.90 V
N004: 0.90 V
N005: 0.61 V
N006: 0.31 V
VOUT: 0.64 V
VDD: 1.80 V
Extracted Vth values:
M1: Vth = 0.51 V
M2: Vth = 0.51 V
```



```
M0: VGS = 0.61 V, VDS = 0.31 V, VTH = 0.37 V
M0 is in Saturation region.
M1: VGS = 0.59 V, VDS = 0.82 V, VTH = 0.51 V
M1 is in Saturation region.
M2: VGS = 0.59 V, VDS = 0.82 V, VTH = 0.51 V
M2 is in Saturation region.
M3: VSG = 0.67 V, VSD = 0.67 V, VTH = 0.39 V
M3 is in Saturation region.
M4: VSG = 0.67 V, VSD = 0.67 V, VTH = 0.39 V
M4 is in Saturation region.
M5: VGS = 0.61 V, VDS = 0.64 V, VTH = 0.37 V
M5 is in Saturation region.
M6: VSG = 0.67 V, VSD = 1.16 V, VTH = 0.39 V
M6 is in Saturation region.
Extracted GDS values (in mS):
M1: 0.01 mS
M2: 0.01 mS
M3: 0.01 mS
M4: 0.01 mS
M5: 0.13 mS
M6: 0.09 mS
M0: 0.04 mS
M7: 0.02 mS
DC Gain: 74.52 dB
|||------RESULTS------|||
Required Gain: 74.0 dB, Achieved Gain: 74.5 dB
Required Phase Margin: 60.0 degrees, Achieved Phase Margin: 74.6 degrees
Required UGB: 10.0 MHz, Achieved UGB: 9.954 MHz
Required OCMR: 0.20-1.60 V, Achieved OCMR: 0.64-1.16 V
Required ICMR: 0.60-1.60 V, Achieved ICMR: 0.84-1.64 V
```

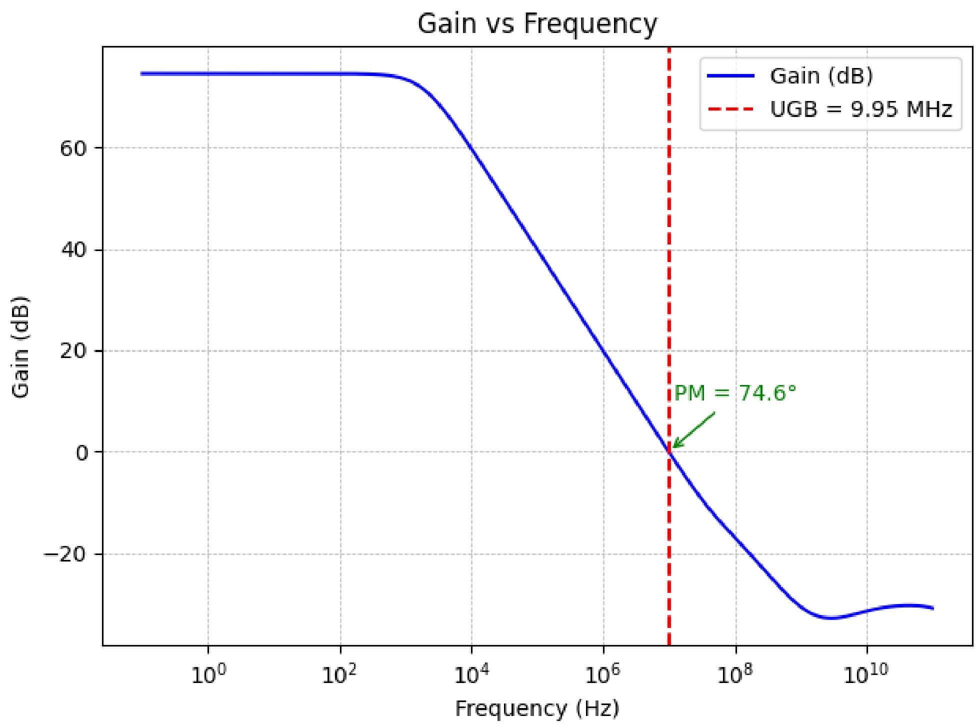


Figure 3: Gain vs Frequency

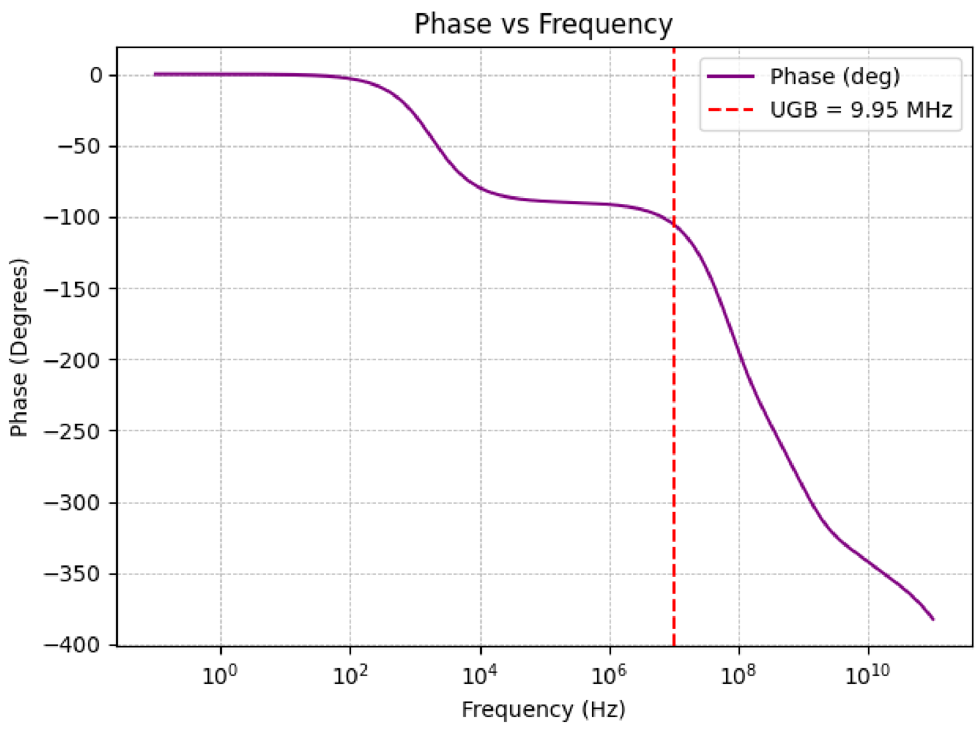


Figure 4: Phase vs Frequency

The automated framework successfully translated the user-defined specifications into a functional two-stage CMOS op-amp design. As shown in the results, the achieved performance metrics closely match the required targets, with a gain of 74.5 dB (target 74 dB), phase margin of 74.6° (target 60°), and unity-gain bandwidth of 9.95 MHz (target 10 MHz). The common-mode and input common-mode ranges are also within acceptable limits, thereby validating the effectiveness of the proposed automation flow.

- Eliminated the need for manual trial-and-error iterations, replacing them with systematic parameter extraction, sizing, and verification.
- Incorporated feedback from simulation outputs (gm, gds, Vth, node voltages) into the sizing loop for improved accuracy.
- Integrated performance verification (Gain, Phase Margin, UGB, ICMR, OCMR) into the design flow automatically.
- Achieved close alignment between required and simulated results, validating the correctness of the automated methodology.
- Provided plots (Gain/Phase response) for intuitive visualization of results after convergence.

## 5 Advantages Over Manual Iteration

- **Time Efficiency** – avoids repetitive re-sizing, netlist editing, and re-running simulations manually.
- **Reduced Human Error** – automation minimizes mistakes in parameter updates, netlist formatting, or reading simulation outputs.
- **Repeatability** – the same specifications always yield the same optimized design, unlike manual tuning where results may vary.
- **Scalability** – once built, the framework can be extended to other op-amp architectures or performance specs.
- **Design Space Exploration** – makes it easier to test different design trade-offs quickly (e.g., gain vs. power vs. bandwidth).
- **Productivity Boost** – allows the designer to focus on architecture-level insights rather than routine low-level sizing.

**In summary, this framework shows that open-source tools, when combined with Python automation, can achieve results comparable to traditional design flows while providing greater adaptability for research-driven optimization techniques. Unlike standard commercial design flows, our approach highlights flexibility for extensions such as gradient-descent–based optimization and yield-aware sizing.**

## 6 Ongoing/Future Work

- **Gradient-descent power minimization (in progress)**- We are working on incorporating a gradient-descent-based optimization module. This technique, guided by a configurable cost function, will enable users to assign relative importance to competing specifications, thereby facilitating power minimization while ensuring that prioritized design metrics are preserved.

- **Extension to Alternative Op-Amp Architectures**- Expand the automation flow from the two-stage Miller op-amp to folded-cascode and telescopic op-amps, which are widely used for low-voltage, high-speed, and high-gain applications. This will allow the tool to automatically suggest the most suitable topology for given supply and spec constraints.

- **Inclusion of Secondary Analog Specifications**- Incorporate slew rate, settling time, output swing, noise performance (thermal/flicker), and offset voltage into the automated framework. These are critical in precision and high-speed analog systems but currently require manual checks.

- **Robust Design Across PVT Variations**- Automate the verification of op-amp performance across process (SS/FF/TT), voltage, and temperature corners, and extend the optimization loop to ensure that ICMR, OCMR, gain, and bandwidth targets hold under worst-case conditions.

- **Mismatch and Monte Carlo Analysis**- Integrate Monte Carlo simulations into the automation to account for random device mismatch. The tool could be extended to size transistors for a target yield (e.g., 95% of chips meet specs) instead of just nominal performance.

- **Compensation Network Optimization**- Extend beyond fixed Miller compensation by automating the sizing of nulling resistors, lead compensation, or even multi-pole compensation networks. This would provide more robust stability across loading conditions.

- **Noise-Constrained Design**- Add noise optimization as part of the cost function — e.g., minimizing input-referred noise while meeting gain-bandwidth and power targets. This would make the tool relevant for low-noise amplifiers (LNAs) and sensor interfaces.

- **Analog Building Block Generalization**- Expand the methodology beyond op-amps to include comparators, current mirrors, and biasing circuits, creating a foundation for an automated analog cell library design tool.

## 7 References

- N. Krishnapura and S. Aniruddhan, ”Analog Integrated Circuit Design,” NPTEL Online Course, Indian Institute of Technology Madras, 2018. [Online]
- B. Razavi, Microelectronics, 2nd ed. Hoboken, NJ, USA: Wiley, 2014.
- B. Razavi, Design of Analog CMOS Integrated Circuits, 2nd ed. New York, NY, USA: McGraw-Hill, 2016.