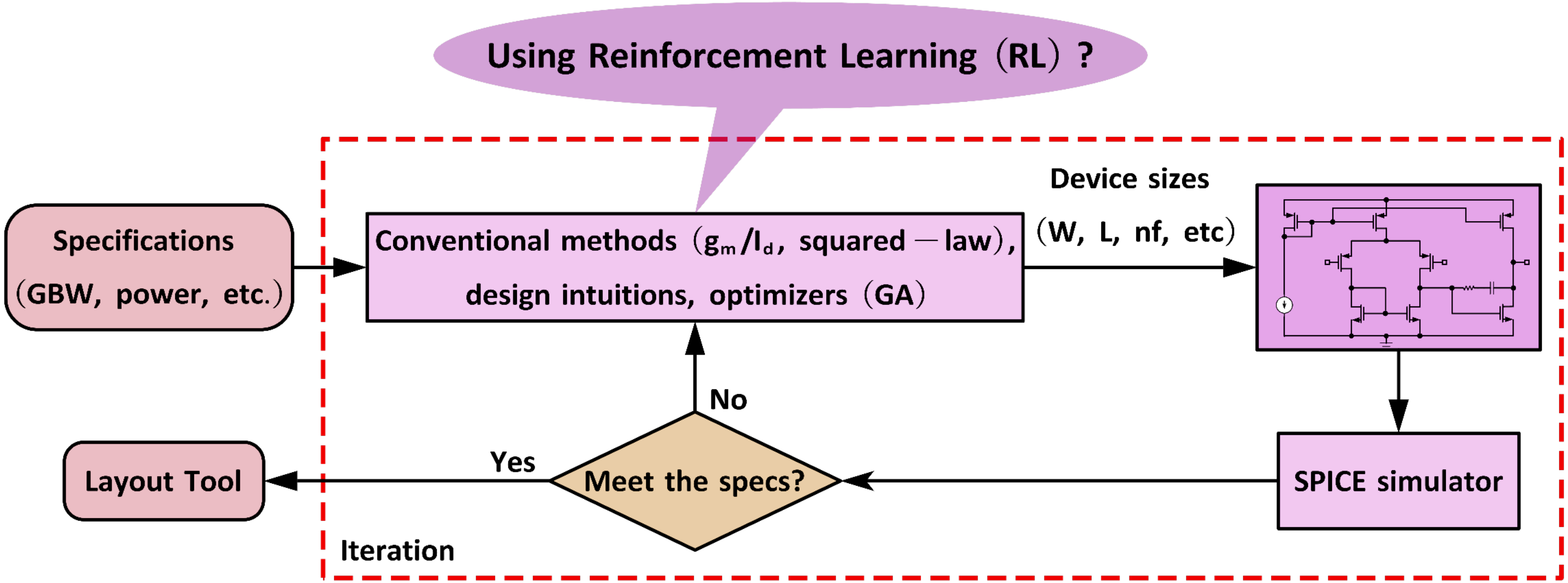
# Design and Optimization of Low-Dropout Voltage Regulator Using Relational Graph Neural Network and Reinforcement Learning in Open-Source SKY130 Process

Zonghao Li, Anthony Chan Carusone
Integrated Systems Laboratory
University of Toronto

# Motivation

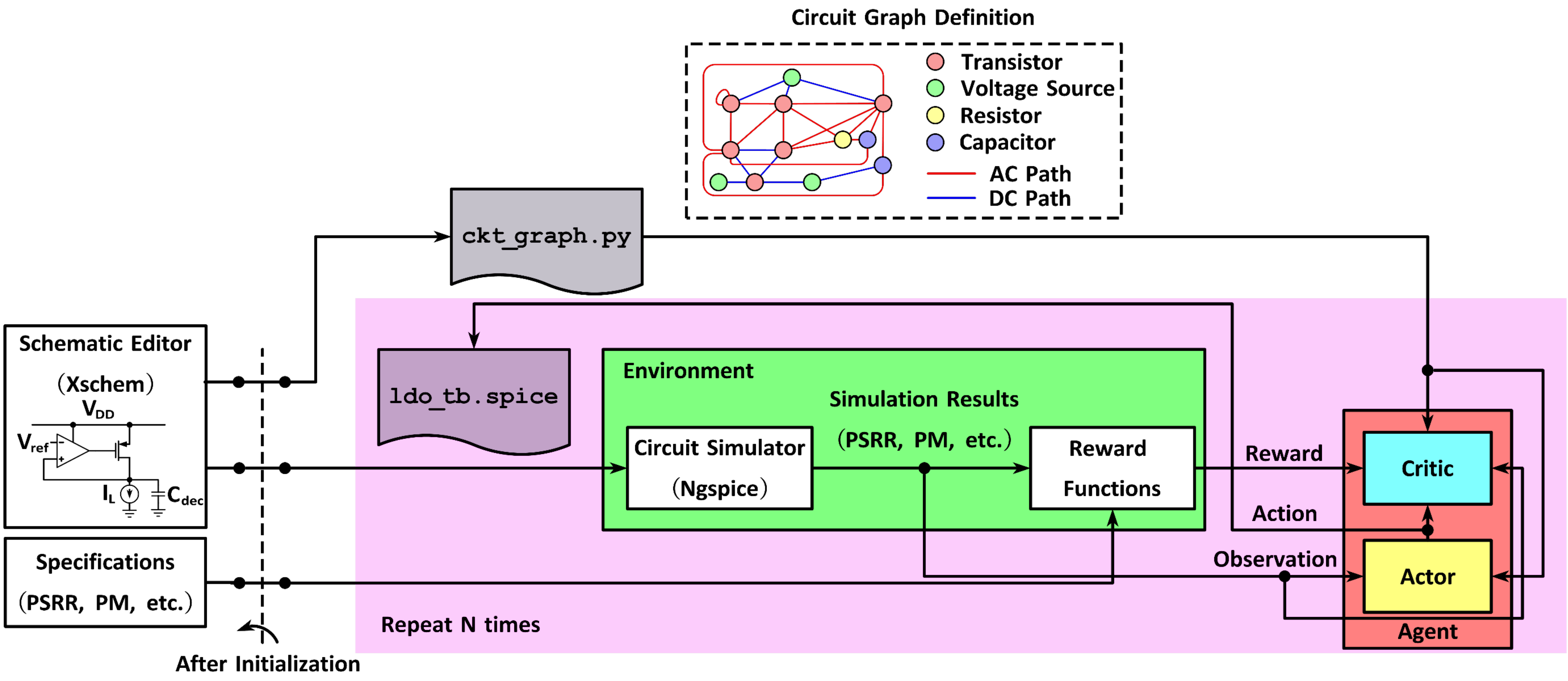



- In analog IC design, sizing circuit components for a given circuit to meet specifications can be challenging.
- Reinforcement learning (RL) has been introduced as an alternative approach to design and optimize analog circuits.

# Motivation

- Using RL to optimize analog ICs:
  - Can be more sample efficient than evolutionary algorithms.
  - Transfer learning (TL) makes RL-trained agent reusable.
- In the domain of ML, reproducibility is crucial, however:
  - Commercial PDKs come with NDA, no public access to the dataset.
  - Trained RL agents cannot be publicized, exacerbate their reusability.
- Open-source PDKs are the good solutions since:
  - They are completely open-sourced without any NDA restrictions.
  - Unrestricted access to the dataset, better reusability.
  - They are manufacturable, so the designs can be taped out.

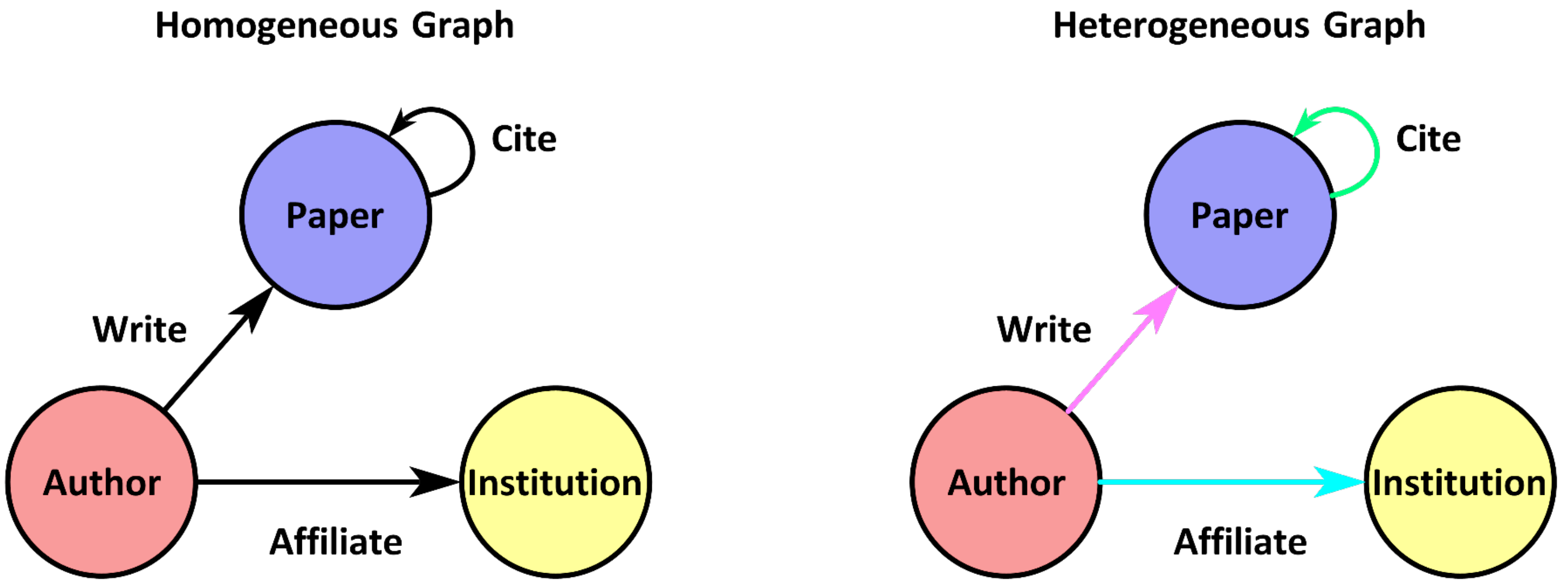
# Overview



Circuit Graph Definition
Transistor
Voltage Source
Resistor
Capacitor
AC Path
DC Path
ckt_graph.py
Schematic Editor
(Xschem)
$V_{DD}$
$V_{ref}$
$I_L$
$C_{dec}$
Specifications
(PSRR, PM, etc.)
After Initialization
ldo_tb.spice
Environment
Simulation Results
(PSRR, PM, etc.)
Circuit Simulator
(Ngspice)
Reward
Functions
Reward
Action
Observation
Critic
Actor
Agent
Repeat N times

# Using GNN as Function Approximator for RL Agent

- Recently, using graph neural network (GNN) as the function approximator is getting popular, such as graph convolutional neural network (GCN).
- We explore applying a kind of heterogeneous GNN called relational graph convolutional neural network (RGCN).
- **The biggest difference between homogeneous and heterogeneous GNN is the latter allows different edge types.**

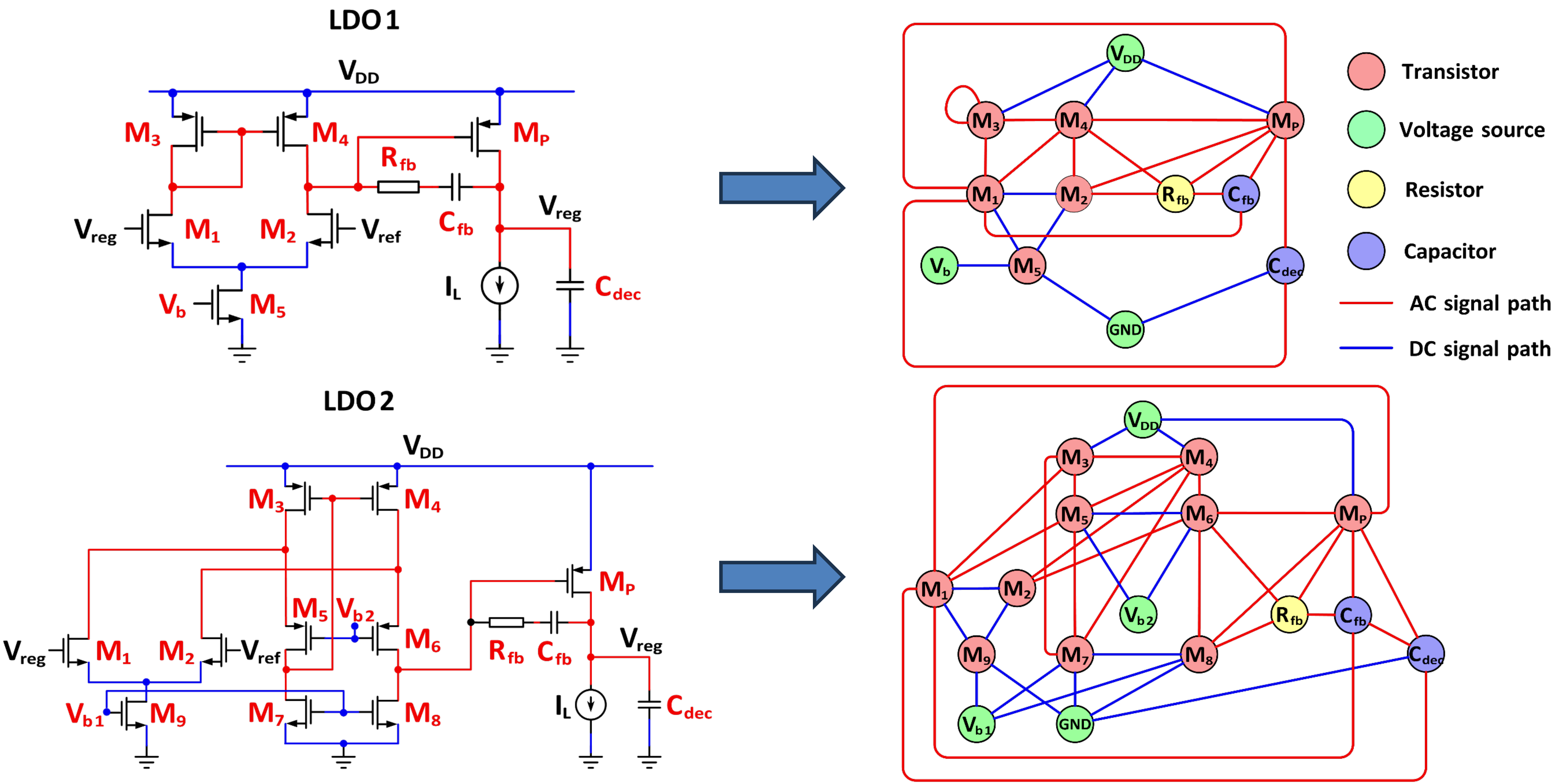


# Using GNN as Function Approximator for RL Agent

- GCN: $H^{(l+1)} = \sigma(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)})$
  - $\tilde{A}$ is the adjacent matric, $\widetilde{D}$ is the degree matrix, $\sigma$ is the non-linear activation function, $H^{(l)}$ is the hidden feature of layer $l$ and $W^{(l)}$ is the layer $l$ weight shared by all edges (since there is only one edge type)
- RGCN: $h_i^{(l+1)} = \sigma(\sum_{r\in R}\sum_{j\in N_i^r}\frac{1}{c_{i,r}}W_r^{(l)}h_j^{(l)} + W_0^l h_i^l)$
  - $r$ is the edge type, $N_i^r$ indicate node $i$ is connected with edge type $r$.
  - Each type $r$ has a learnable weight matrix $W_r^{(l)}$ for layer $l$
- **As an example, we could try to categorize a circuit net based on the signals they are carrying: AC and DC signals.**

[Schlichtkrull, 2017]

# Test Vehicle: Low-Dropout Voltage Regulator (LDO)



# Reward Function

- The reward function $R$ is formulated as:

$$r_i = \omega_i \beta_i \left\{ \frac{O_i - O_i^*}{O_i + O_i^*}, 0 \right\}_{min}$$

$$R = \begin{cases} \sum_{i=1}^{N} r_i & if\ \exists r_i < 0 \\ \sum_{j=1}^{N} \omega_j \beta_j \frac{O_j - O_j^*}{O_j + O_j^*} + C & else \end{cases}$$

TABLE I: LDO Specifications

| | |
|---|---|
| $V_{DD}\,(V)$ | 2 |
| $V_{reg}\,(V)$ | 1.8 |
| $V_{drop}\,(mV)$ | $\leq 200$ |
| $I_L$ | $[10\,\mu A, 10\,mA]$ |
| $PSRR_{\leq 10kHz}\,(dB)$ | $\leq -30$ (at $I_{L,min}$ and $I_{L,max}$) |
| $PSRR_{\leq 1MHz}\,(dB)$ | $\leq -20$ (at $I_{L,min}$ and $I_{L,max}$) |
| $PSRR_{\geq 1MHz}\,(dB)$ | $\leq -5$ (at $I_{L,min}$ and $I_{L,max}$) |
| $PM\,(deg)$ | $\geq 60°$ (at $I_{L,min}$ and $I_{L,max}$) |
| $I_q\,(\mu A)$ | $\leq 200$ ($\leq 400$ for LDO2) |
| $\Delta V_{reg}\,(mV)$ | $\leq 36$ |
| $C_{dec}$ | As small as possible |

- Here, $O$ is the simulated result and $O^*$ is the target specification. $\omega$ is the weight assigned to a specification. $i$ represents the hard specifications and $j$ are the soft specifications. $\beta$ would be either -1 (minimization task) or 1 (maximization task). $C$ is a constant.
- Taking the $min$ action is to avoid over-optimizing.
- If $R = 0$, all hard specifications are met.

# Reward Function

- A few modifications are made to the reward function $R$:
  - Change $\omega_{PSRR \leq 10kHz} = 0.5$
  - Modify $r_{PSRR}$:

$$r_{PSRR} = \begin{cases} -1 & if\ PSRR \geq 0 \\ -\omega_i \left\{ \dfrac{O_i - O_i^*}{O_i + O_i^*}, 0 \right\}_{min} & else \end{cases}$$

  - Modify $r_{PM}$:

$$r_{PM} = \begin{cases} -1 + \omega_i \left\{ \dfrac{O_i - O_i^*}{O_i + O_i^*}, 0 \right\}_{min} & if\ PM \leq 45° \\ \omega_i \left\{ \dfrac{O_i - O_i^*}{O_i + O_i^*}, 0 \right\}_{min} & else \end{cases}$$

# Action

- Design in SKY130 CMOS Process.
- We use high voltage transistors (5V/10.5V) for both NMOS and PMOS.
- For $R_{fb}$ we use high sheet resistance (1112.4 Ω/□) poly resistor.
- For $C_{fb}$ and $C_{dec}$ we use MiM capacitor with a capacitance density of $2fF/\mu m^2$
- Using device multiplier $M$ to reduce the action dimension of resistor and capacitor by fixing their $W$ and $L$.

TABLE II: LDO Action Space

| LDO1 | $W\ (\mu m)$ | $L\ (\mu m)$ | $M$ | $V$ |
|---|---|---|---|---|
| $M_1$ - $M_5$ | $[1, 100]$ | $[0.5, 2]$ | 1 | |
| $M_P$ | $[10, 100]$ | $[0.5, 1]$ | $[100, 2000]$ | |
| $R_{fb}$ | 0.35 | 1 | [1] $[1, 20]$ | |
| $C_{fb}$ | 10 | 10 | [2] $[1, 50]$ | |
| $C_{dec}$ | 30 | 30 | [3] $[10, 300]$ | |
| $V_b$ | | | | $[0.9, 1.4]$ |
| **LDO2** | $W\ (\mu m)$ | $L\ (\mu m)$ | $M$ | $V$ |
| $M_1$ - $M_9$ | $[1, 100]$ | $[0.5, 2]$ | 1 | |
| $M_P$ | $[10, 100]$ | $[0.5, 1]$ | $[100, 2000]$ | |
| $R_{fb}$ | 0.35 | 1 | [1] $[1, 20]$ | |
| $C_{fb}$ | 10 | 10 | [2] $[1, 50]$ | |
| $C_{dec}$ | 30 | 30 | [3] $[10, 300]$ | |
| $V_{b1}$ | | | | $[0.9, 1.4]$ |
| $V_{b2}$ | | | | $[0, 1]$ |

[1] Corresponds to $[476.7, 9335]\ \Omega$.
[2] Corresponds to $[0.2076, 10.38]\ pF$.
[3] Corresponds to $[18.23, 546.8]\ pF$.

# Action

- **Using min-max normalization to do the feature scaling (bounded by $[-\mathbf{1}, \mathbf{1}]$):**

$$A_{scaled} = \frac{\left(A - \frac{A_{max} + A_{min}}{2}\right)}{\left(\frac{A_{max} - A_{min}}{2}\right)}$$

- To help RL agent explore the action space, **we added the truncated uniform noise (bounded by $[-\mathbf{1}, \mathbf{1}]$) to the action taken by the actor**, with an initial noise volume $\sigma = 2$ and an exponential decay factor $\epsilon = 0.999$ $(0.9995$ for LDO2 for more explorations) .

[Wang, 2020]

# Observation

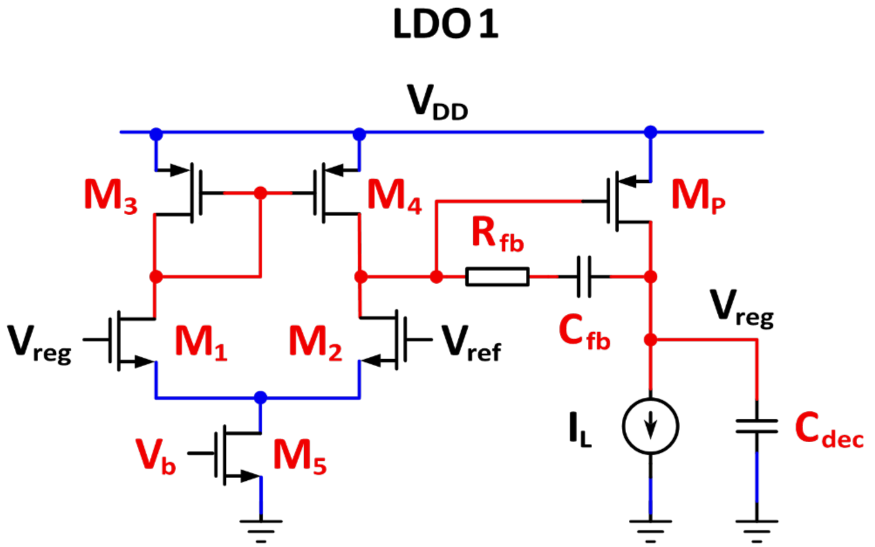
- The observations are the attributes of each circuit components.
- **Attributes for resistor, capacitor and voltage sources are simply their resistance, capacitance, and voltage values, respectively.**
- **Transistor attributes can be obtained by running DCOP analysis of the circuit.**
- For transistors, the node attributes will be a vector containing some essential attributes of a transistor $M$ is:

$$S_M = [i_d, g_m, g_{ds}, V_{th}, V_{dsat}, V_{ds}, V_{gs}]$$

# Observation

- As an example, the observation matrix for LDO1 is:

$$S = \begin{bmatrix} S_{M_N} & \mathbf{0}_{6\times 6} \\ \mathbf{0}_{6\times 7} & diag(V_b, V_{DD}, V_{GND}, R_{fb}, C_{fb}, C_L) \end{bmatrix}$$




where:

$$S_{M_N} = \begin{bmatrix} S_{M_1} \\ S_{M_2} \\ \dots \\ S_{M_P} \end{bmatrix}$$

# Observation

- It is also important to normalize the observations.
- For passive components, their ranges are explicitly defined, can therefore be normalized using min-max normalization.
- **For transistors, it is hard to define a clear range for each attribute.**
- Therefore, z-score normalization is used:

$$\hat{S}_{M_N} = \frac{S_{M_N} - \mu_{S_{M_N}}}{\sigma_{S_{M_N}}}$$

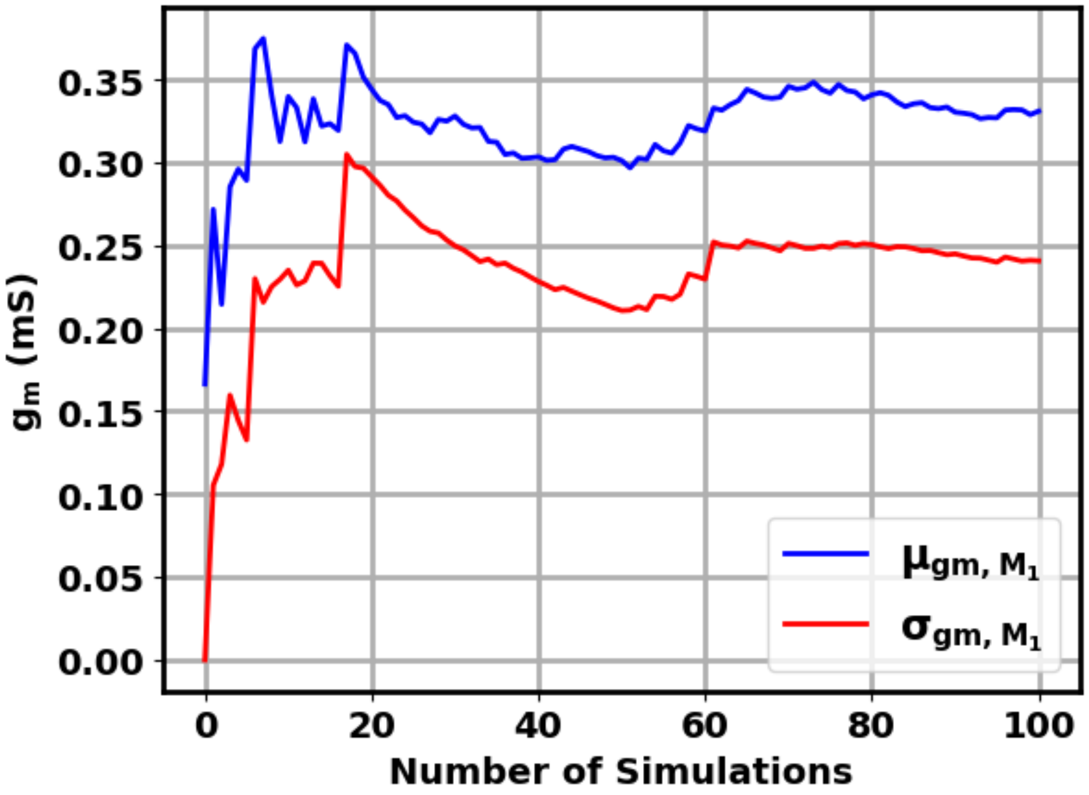
- **We also do not want to having the $\mu$ and $\sigma$ change across simulation run.**

# Observation

**Algorithm 1** Find $\mu_{S_{M_N}}$ and $\sigma_{S_{M_N}}$

**Require:** $n \in \mathbb{N}^+$ ▷ Such as $n = 100$
**Require:** $S^* = [\,]$ ▷ Empty list to store observations

$i \leftarrow 0$
**while** $i < n$ **do**
  Sample $A$ randomly from Table III
  Run `OP` analysis
  Store $S_{M_N}$ in $S^*$
  $i \leftarrow i + 1$
**end while**
$\mu_{S_{M_N}} = mean(S^*)$
$\sigma_{S_{M_N}} = std(S^*)$

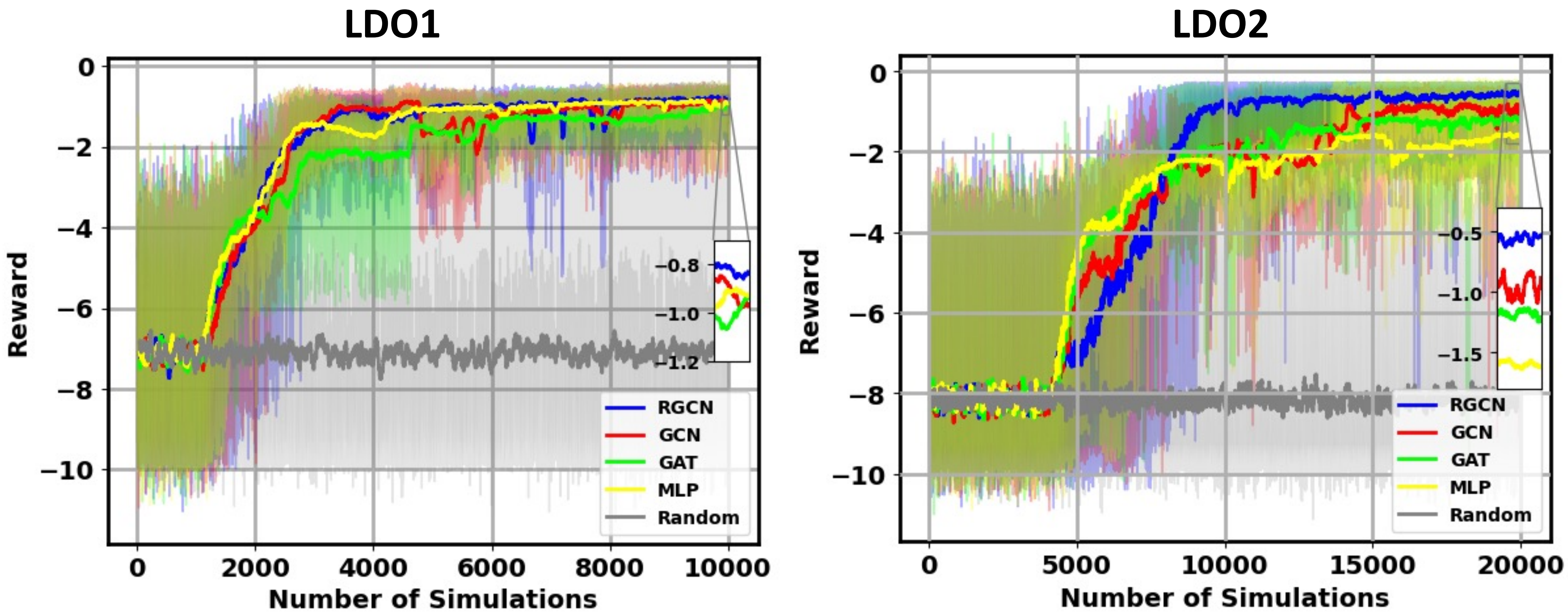



- Only need to be done once for a circuit.

# Optimization Setup

- Using Deep Deterministic Policy Gradient (DDPG) as the RL algorithm.
  - Off-policy, can be more sample efficient than on-policy methods.
  - For continuous action space.
  - Stability might be a concern.
- Four different function approximators are experimented for each LDO circuit: RGCN, GCN, Graph Attention Network (GAT), and Multi-layer Perceptron (MLP).
  - All NNs have four layers.
  - Each experiment has been run three times independently.

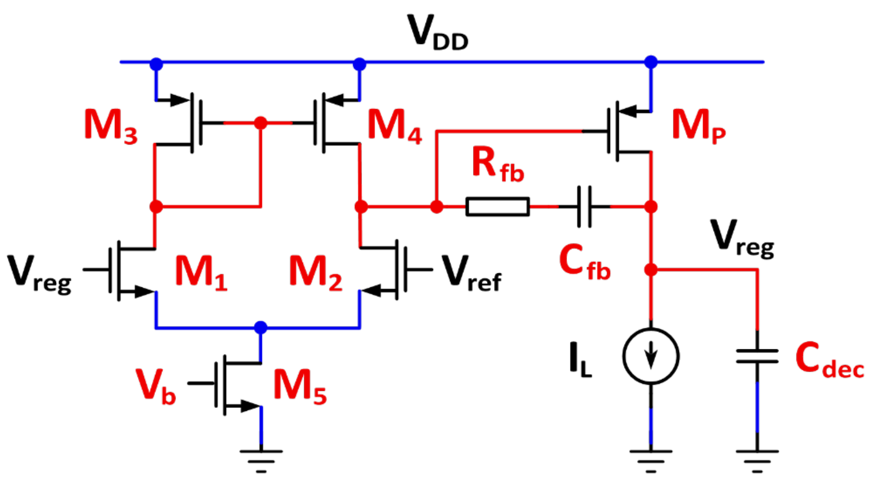
# Optimization Results

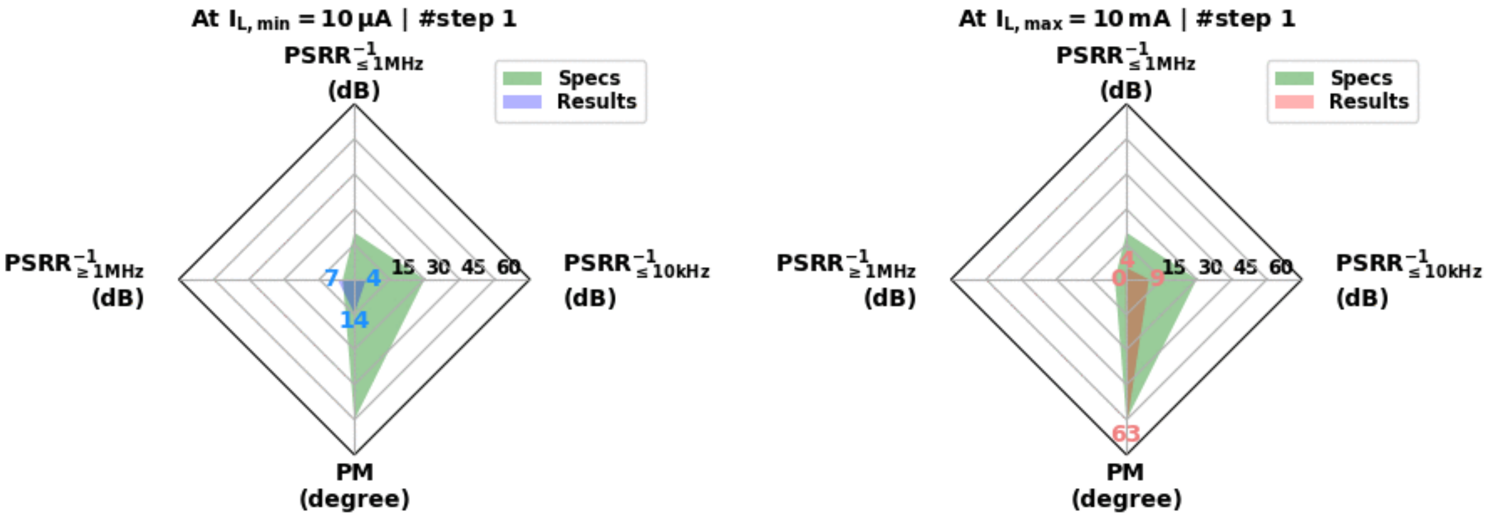


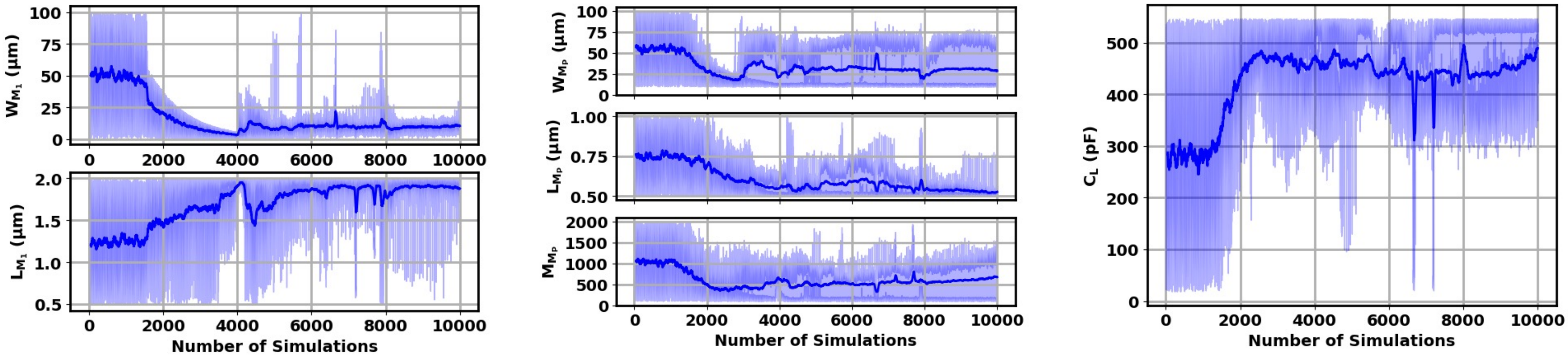

- Small reward value difference sometimes can still mean a big performance difference, since they are normalized.

# Optimization Results

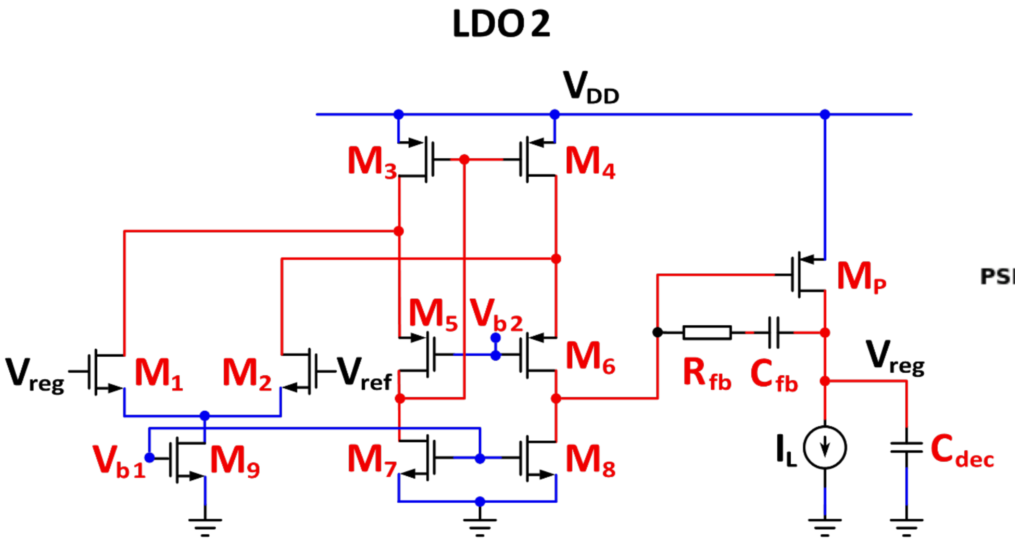
LDO 1

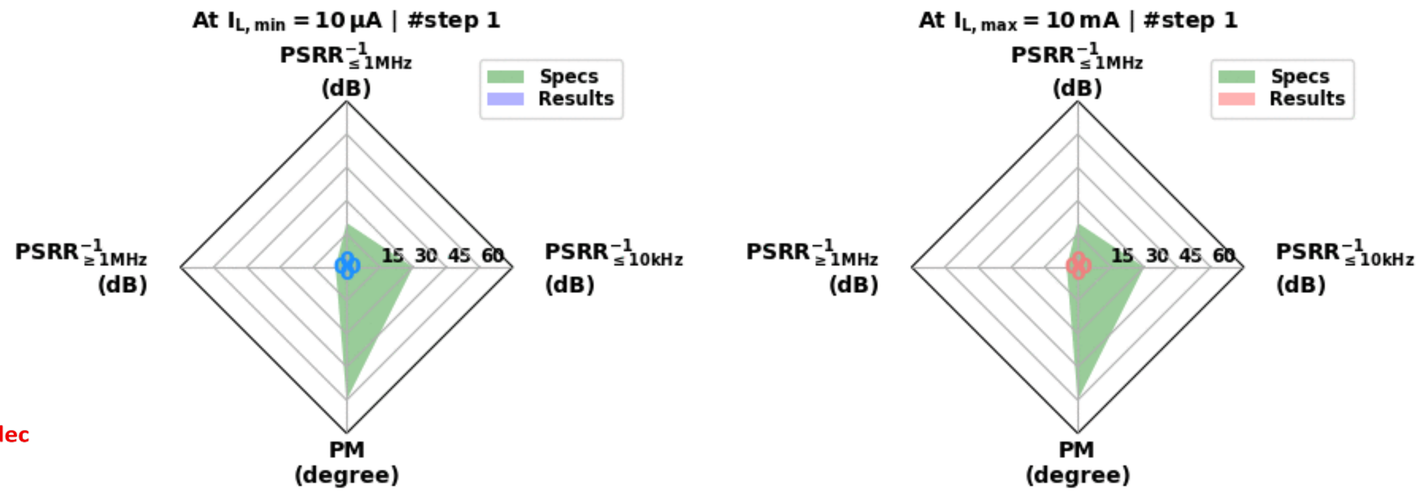


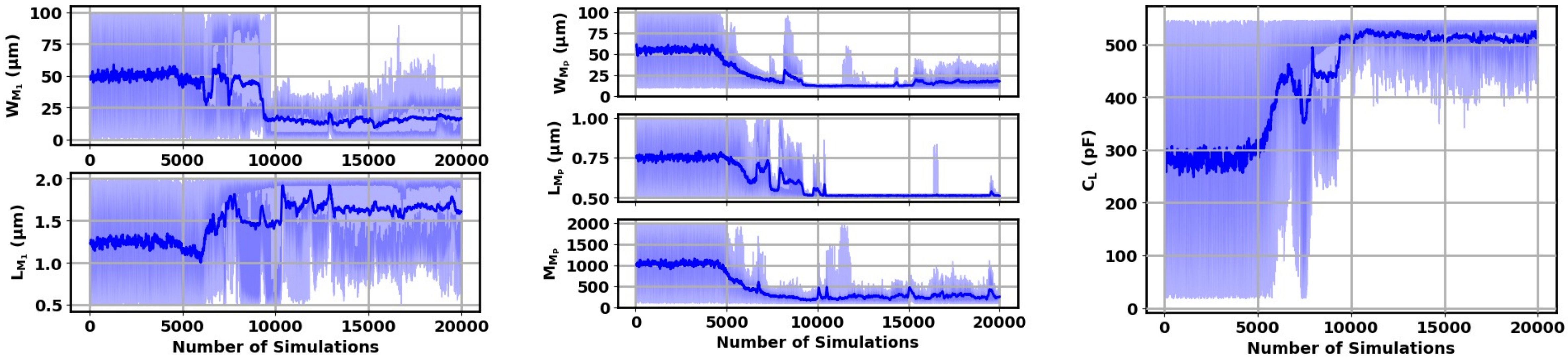



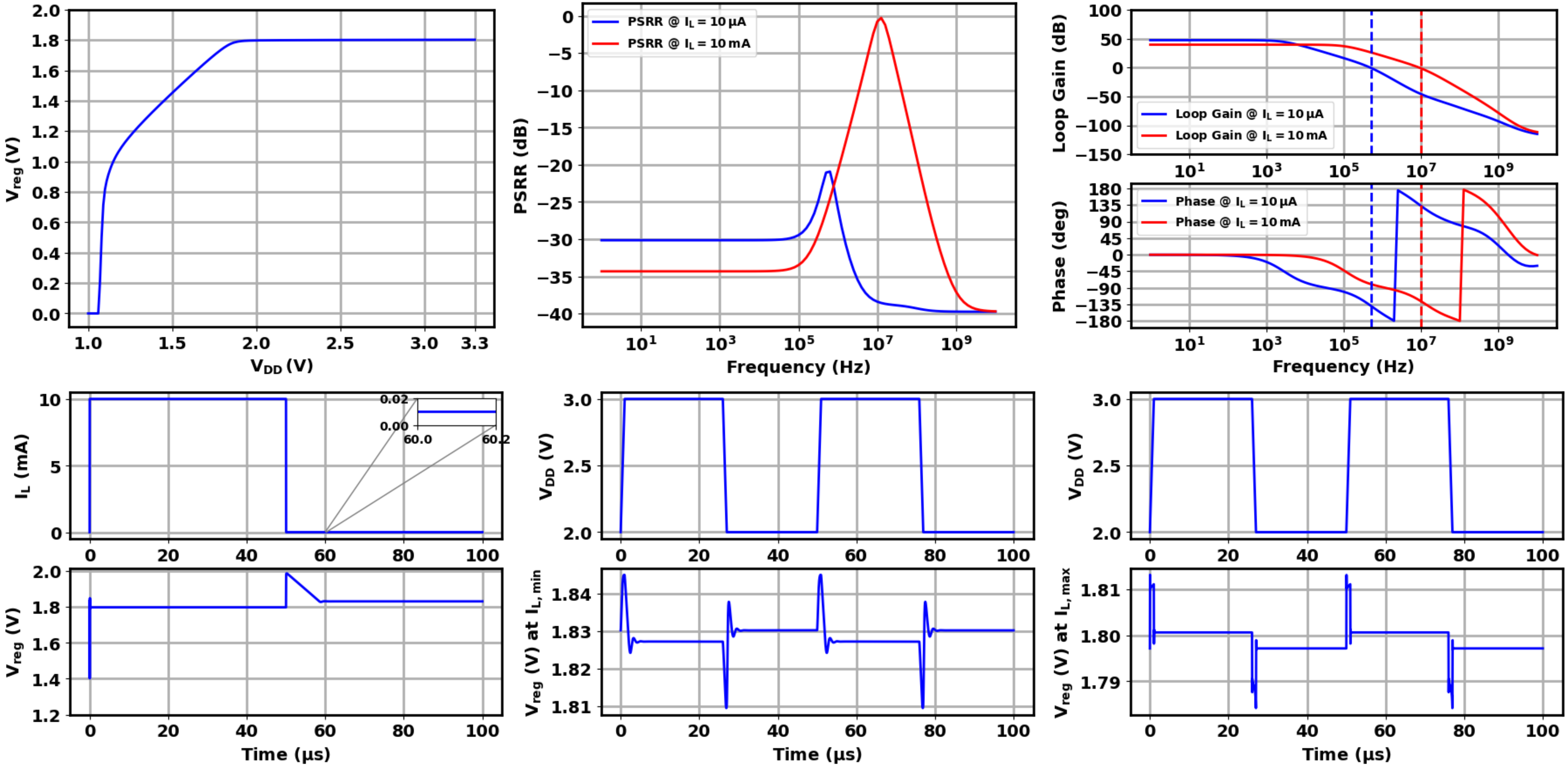
# Optimization Results



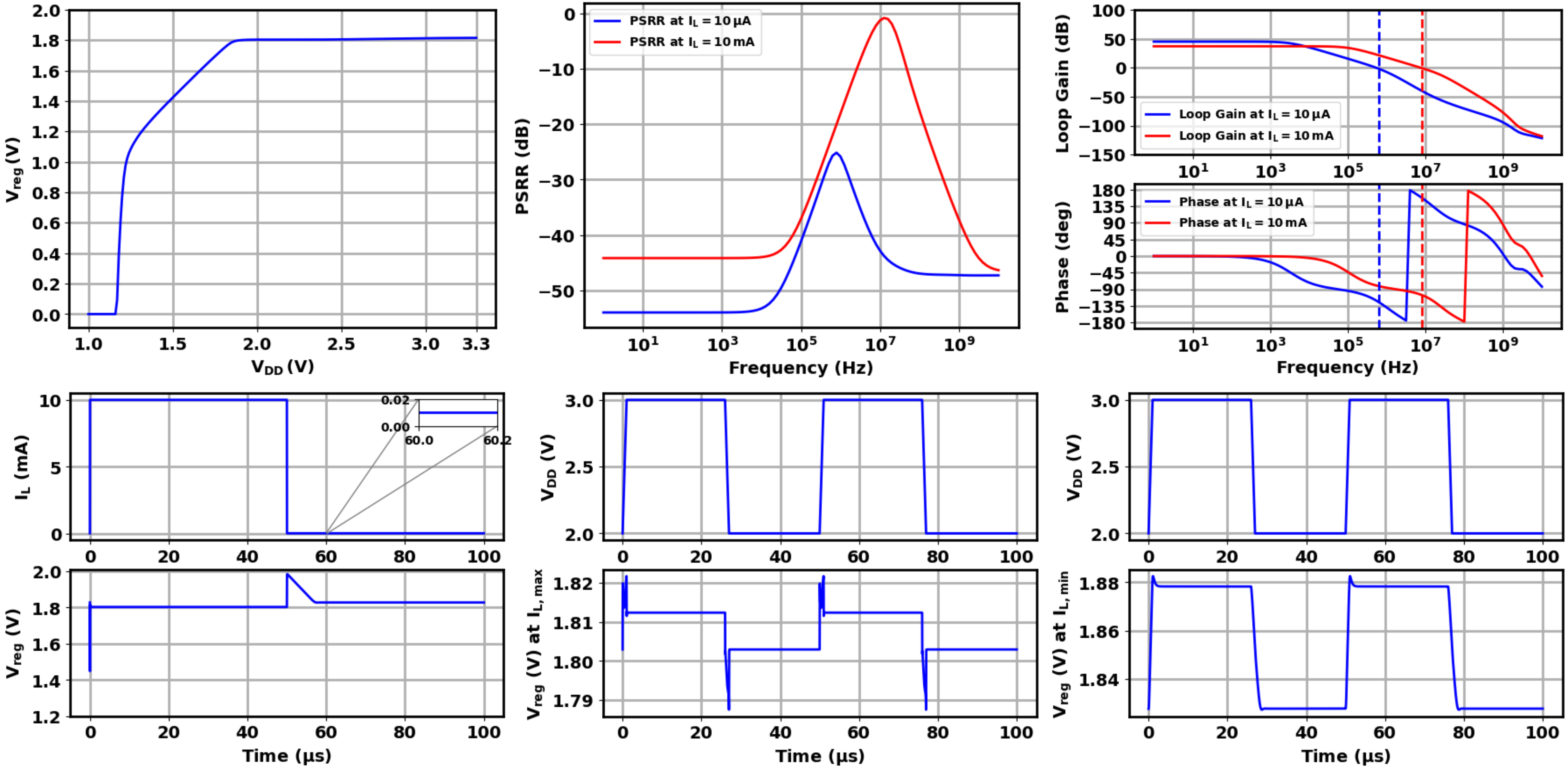



# Optimization Results – LDO1



# Optimization Results – LDO2



# Optimization Results

TABLE IV: Optimization Results of LDOs

| | Specifications | LDO1 | LDO2 |
|---|---|---|---|
| $V_{drop}\,(mV)$ at $I_{L,max}$ | $\leq 220$ | 203 | 197 |
| $I_L$ | $[10\,\mu A, 10\,mA]$ | $[10\,\mu A, 10\,mA]$ | $[10\,\mu A, 10\,mA]$ |
| $PSRR_{\leq 10kHz}\,(dB)$ | $\leq -30$ (at $I_{L,min}$ and $I_{L,max}$) | **-29.3** at $I_{L,min}$ and -34 at $I_{L,max}$ | -53.53 at $I_{L,min}$ and -44.1 at $I_{L,max}$ |
| $PSRR_{\leq 1MHz}\,(dB)$ | $\leq -20$ (at $I_{L,min}$ and $I_{L,max}$) | -22 at $I_{L,min}$ and -23.2 at $I_{L,max}$ | -25.2 at $I_{L,min}$ and -20 at $I_{L,max}$ |
| $PSRR_{\geq 1MHz}\,(dB)$ | $\leq -5$ (at $I_{L,min}$ and $I_{L,max}$) | -25.75 at $I_{L,min}$ and **-0.32** at $I_{L,max}$ | -25.86 at $I_{L,min}$ and **-0.9** at $I_{L,max}$ |
| $PM\,(deg)$ | $\geq 60°$ (at $I_{L,min}$ and $I_{L,max}$) | **54.4**° at $I_{L,min}$ and 61° at $I_{L,max}$ | 61.2° at $I_{L,min}$ and 78.2° at $I_{L,max}$ |
| $I_q\,(\mu A)$ | $\leq 200$ ($\leq 400$ for LDO2) | 175.32 | 398 |
| $\Delta V_{reg}\,(mV)$ | $\leq 36$ | **40** | 24.74 |
| $C_{dec}\,(pF)$ | As small as possible | 476.78 | 435 |

# Comparison

| | Wang NIPS'18 | Wang DAC'20 | Settaluri TCAD'22 | Cao DAC'22 | This work |
|---|---|---|---|---|---|
| RL algorithm | DDPG | DDPG | PPO | PPO | DDPG |
| RL function approximator | RNN | GCN | MLP | GCN, GAT | RGCN, GCN, GAT, MLP |
| Circuits considered | TIA | OTA, TIA, LDO | TIA, OTA | OTA, PA | LDO |
| Technology | 180nm CMOS | 180nm CMOS | 16nm CMOS | 45nm CMOS, 150nm GaN | SKY130 CMOS |
| Action dimensions | 19 | 25 | 29 | 15 | 18 |
| Number of specifications | 4 | 5 | 7 | 4 | 13 |
| Fully open-sourced? | No | No | No | No | Yes |

# Conclusion

- We have demonstrated an open-sourced RL framework for designing and optimizing vanilla LDO circuits in the open-source SKY130 CMOS process, conditioned on a series of specifications.
- We believe that our proposed framework, with certain reward engineering and hyperparameter modifications, can be generalized to other analog circuits.
- In the future, we plan to expand our model to include post-layout simulation inside the optimization loop to deliver an end-to-end solution. Furthermore, by leveraging the ability of transfer learning, we aim to explore how the trained RL agent could be applied to other technology nodes.
- Linked to the GitHub: https://github.com/ChrisZonghaoLi/sky130_ldo_rl

# IEEE SSCS "Code-a-Chip" Travel Grant Awards at ISSCC'24

## IEEE SSCS Open-Source Ecosystem "Code-a-Chip" Travel Grant Awards at ISSCC'24

**The IEEE SSCS Code-a-Chip Travel Grant Award** was created to:

1. Promote *reproducible* chip design using *open-source* tools and *notebook-driven* design flows and
2. Enable up-and-coming *talents* as well as seasoned *open-source enthusiasts* to travel to IEEE SSCS conferences and interact with the leading-edge chip design community.

- For more information: https://github.com/sscs-ose/sscs-ose-code-a-chip.github.io/

# References


- M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, “Modeling relational data with graph convolutional networks,” 2017.
- H. Wang, J. Yang, H.-S. Lee, and S. Han, “Learning to design circuits,” 2020
- K. Settaluri, Z. Liu, R. Khurana, A. Mirhaj, R. Jain, and B. Nikolic, “Automated design of analog circuits using reinforcement learning,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 41, no. 9, pp. 2794–2807, 2022.
- H. Wang, K. Wang, J. Yang, L. Shen, N. Sun, H.-S. Lee, and S. Han, “Gcn-rl circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning,” in 2020 57th ACM/IEEE Design Automation Conference (DAC), pp. 1–6, 2020.
- W. Cao, M. Benosman, X. Zhang, and R. Ma, “Domain knowledge-infused deep learning for automated analog/radio-frequency circuit parameter optimization,” in Proceedings of the 59th ACM/IEEE Design Automation Conference, ACM, jul 2022.

# Thank You!