A Comparative Study of Reinforcement Learning Techniques for Analog Electronic Design Automation

Kannan Sankaragomathi [*1] **Tony Cai** [*1] **Qingwei Zeng** [*1]

Abstract

In this project we compare the performance of two reinforcement learning (RL) techniques when applied to the transistor sizing problem in the analog design automation field. The two RL techniques we investigated are GCN-RL (Graph Convolutional Neural Network - RL) and Non-graph RL (NRL). We compare the techniques based on number of simulations needed for convergence and training stability. Our findings indicate that GCN outperforms NRL in terms of training stability and almost always finds a valid circuit that meets the specifications quicker than NRL (less number of circuit simulations) for the three circuits we experimented with (OTA-5T, AMP2,LDO). This indicates that using the graph structure inherent in the circuit topology is critical in building an AI agent that designs analog circuits. These findings are inline with recent literature(Wang et al., 2020).

1. Motivation

Analog-Mixed-Signal (AMS) circuits are ubiquitous in electronic devices that we use everyday such as mobile phones, medical sensors etc. Currently, AMS circuit design is a very manual process where human expertise is needed in several steps of the process (such as transistor sizing, custom layout design etc.) increasing cost / time-to-market (see Figure 1). Increasingly stringent design rules in state-of-the-art semiconductor processes and frequent requirements to manually port a design from one semiconductor process to another exacerbate the problem of manual AMS design.

2. Transistor sizing problem

Recently, researchers have started applying machine learning (ML) algorithms to automate various steps of the analog design process to alleviate the problems mentioned above. In this project, we are targeting the specific problem of 'transistor-sizing'. The input to the problem is a circuit topology made up of unit circuit elements (transistors, capacitors, etc.) and a set of performance specifications that the circuit needs to achieve. The circuit elements are parameterized by a set of sizing parameters (eg. width/length of transistors, area of capacitors) that can impact the design specifications. The goal of the designer (in this case the RL agent) is to identify the circuit parameters that achieve / exceed the performance specifications. The search space for the sizing problem is enormous eliminating the possibility of a brute force search.

Reinforcement Learning (RL) based approaches are becoming attractive solutions for solving the transistor sizing problem. In (Wang et al., 2018) the authors have built a DDPG agent with a Recurrent Neural Network (RNN) based actor/critic networks, that outputs circuit parameters as continuous actions and demonstrate their performance on a class of linear circuits called operational amplifiers. They further extended this work by incorporating the circuit graph information into the actor/critic networks using a Graph-Convolutional-Neural-Network (GCN), calling it the GCN-RL technique (Wang et al., 2020) .

3. Methods

3.1. RL environment

We use NGPSICE as the circuit simulator that takes the transistor netlist and outputs the circuit performance metrics. We combine the performance metrics using a weighted mean to to create single figure of merit (FoM) defined as below

$$\text{FOM} = \sum_{i=1}^{n} \left(w_i \frac{\min(M_i, M_{i,spec}) - M_{i,spec_{min}}}{M_{i,spec_{max}} - M_{i,spec_{min}}} \right) \quad (1)$$

Where M_is are the simulated performance metrics, $M_{i,spec}$is the required performance metric. We also normalize the metric with the minimum and maximum possible values for the specifications. The methodology allows for specifications to have a maximum bound beyond which increasing the metric yields no benefit to the circuit. w_i are the user defined weights that specify the importance of

[*]Equal contribution [1]Department of Computer Science, Stanford University. Correspondence to: Kannan Sankaragomathi <kannansa@stanford.edu>.

CS229 Spring 2021 Project report

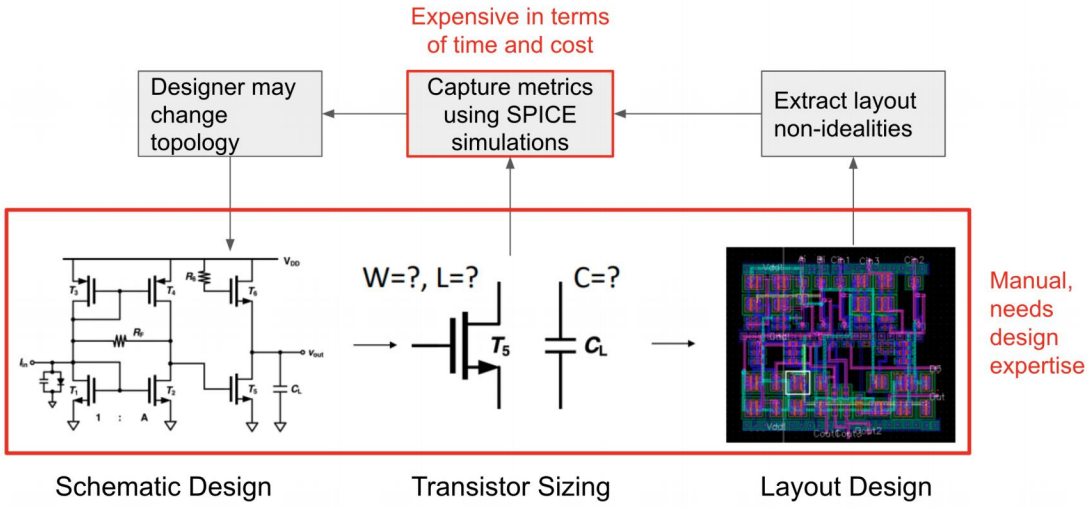

Figure 1. AMS circuit design process showing labor intensive manual steps

each circuit metric. The RL environment wraps around the NGSPICE simulator. It takes sizing parameters (actions) as inputs, triggers the circuit simulator, processes the simulator's output and returns an observation and reward. The reward is the FoM as described in Equation (1). The observation includes the DC operating point information (g_m, v_{ds}, v_{gs}, g_{ds}) of all the transistors in the circuit in a matrix form. The environment also includes the definition of the adjacency matrix of the circuit graph which is fed to the GCN-RL agent (described in next section) if needed.

3.2. RL agent setup

We use a DDPG agent similar to (Wang et al., 2018). DDPG is an off-policy reinforcement learning algorithm which consists of an actor network and a critic network. DDPG works as follows:

1. In each iteration, DDPG selects an action according to the actor's policy. The tuple (S, A, R, S') (corresponding to current state, action, reward, and next state are stored) in a buffer.

2. If the buffer is big enough, we minimize the loss function for critic network w.r.t. θ^Q, defined as:

$$L(\theta^Q) = \mathbb{E}\left[(Q(s_t, a_t|\theta^Q) - y_t)^2\right] \quad (2)$$

   where $y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, \mu(s_{t+1}|]\theta^Q)$.

3. The actor network for the actor network, whose gradient is approximated by:

$$\nabla_{\theta^Q} J \approx \mathbb{E}\left[\nabla_{\theta^\mu} Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t|\theta^Q)}\right] \quad (3)$$

We started with the DDPG agent code from (Tabor) which implements the DDPG description from (Lillicrap et al., 2016) as a baseline and modify it to suit our application. The implementation uses a replay buffer, an actor network, a critic network, a target-actor network and a target-critic network (all FCNN in our implementation). The implementation uses discount rate of 0.99, batch size of 64, and learning rates of 0.001 and 0.002 for the actor / critic respectively. Because circuit simulation is a one-step process, we created a multi-step environment whereby the agent simulates $n = 150$ times per episode.

3.3. NRL agent

We experimented with two different agent architecture for the actor and critic network. The first is a very simple architecture consisting of two hidden FC dense layers with ReLU activation and a final dense layer with tanh activation. This architecture only uses local observation in prediction, but does not take connectivity into account. In order to embed connectivity information into learning, we modelled the circuit as a graph where the nodes are the circuit components and the edges are the component connections, and learn on the memory buffer using GCN. This is described in more details in the next subsection.

3.4. GCN-RL Agent Architecture

To embed circuit topology information in the agents, we leverage Graph Convolution Network (GCN). We used the symmetric optimization as proposed in [3],

$$f(H^{(l)}, A) = \sigma\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \quad (4)$$

Here, $\hat{A}$ is the modified adjacency matrix for A with added self-loops.

This normalization ensures that nodes with few neighbours have more weights.

The actor and critic networks have very similar architecture. In particular, they consist of three layers of GCN combined with a mean-pooling layer stacked sequentially. The mean-pooling layer serves to ensure that critic-network predictions is independent of the ordering of nodes. The input layer for the critic network is the concatenation of state and action vectors (so the feature for each node in the critic-network graph is the concatenation of states and action for each circuit component). The output layer layer is a FC dense layer with 1 output unit for the critic network and a FC dense layer with $2n$ output units for the actor network (since two actions are supported per component).

4. Experiments and results

4.1. Circuit setup

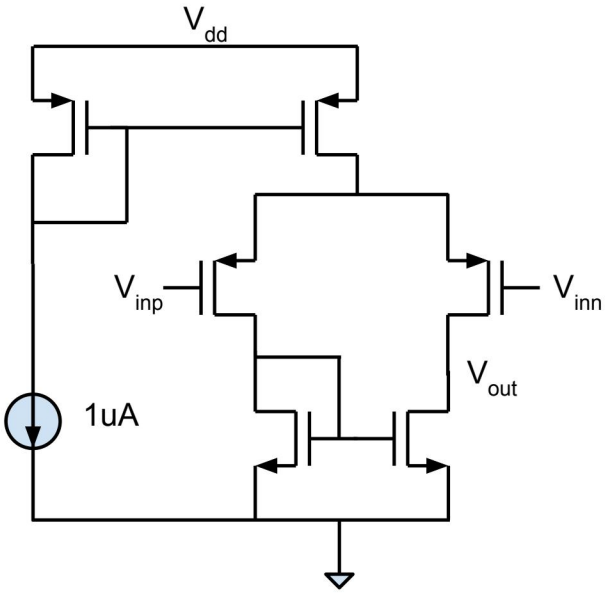

Figure 2. OTA-5T

For our experiments, we tested the two RL agents on three different circuits.

- A 5 transistor Operational Transconductance Amplifier (OTA-5T)
- Two stage Miller Compensated Amplifier (AMP2)
- Linear DropOut regulator (LDO)

The transistor level diagram of the three circuits can be seen in figure. 2, figure.3, figure.4 respectively. The three

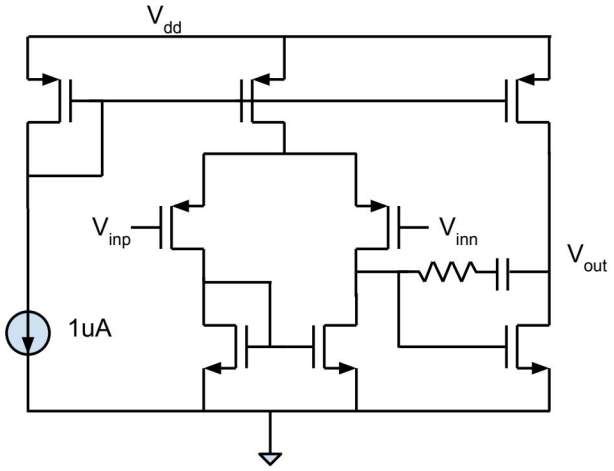

Figure 3. AMP2 circuit

circuits use a combination of transistors (PMOS & NMOS), resistors (R) and capacitors (C). The number of elements of each type in the above three circuits can be seen in table 1. The sizing parameters needed for PMOS & NMOS are

Circuit	PMOS	NMOS	R	C
OTA-5T	4	2	0	0
AMP2	5	3	1	1
LDO	5	4	3	1

Table 1. Number of circuit components in the three circuits used

the width (W) and Length(L). The resistors and capacitors need just one parameter (R & C) respectively to completely define their behavior.

The circuit performance is captured by performance metrics given in table 2. We construct our performance metrics such that a higher number indicates better performance.

Figure. 5-Figure. 10 show samples from running the GCN-RL and NRL agents for the three circuits. The green line shows the maximum possible FoM for the given specs. Hitting the green line indicates that the algorithm has found a valid solution. We can see that GCN reaches a valid solution and bounces around closer to the valid solution in further episodes.

Figure.11 Figure 12 show the run-to-run variation of the NRL agent and GCN-RL agent respectively on the OTA-5T circuit. We also see that the NRL agent does not learn very well indicated by the lack of consistent increase in Fom over simulations.
To show significance, we ran the DDPG algorithm for both

circuit	Performance metric
OTA-5T	DC-Gain, Unity-Gain-Bandwidth, Phase margin, $\frac{1}{\text{power consumption}}$
AMP2	DC-Gain, Unity-Gain-Bandwidth, Phase margin, $\frac{1}{\text{power consumption}}$
LDO	$\frac{1}{\text{Settling time}}$, $\frac{1}{\text{Settling error}}$, $\frac{1}{\text{power consumption}}$

Table 2. Performance metrics used for circuits

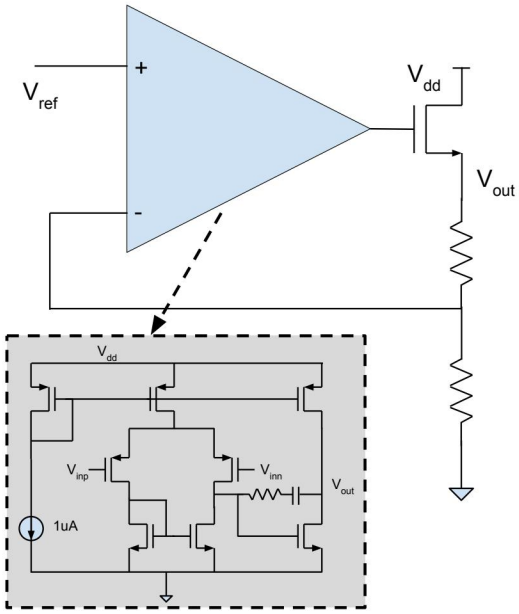

Figure 4. LDO circuit

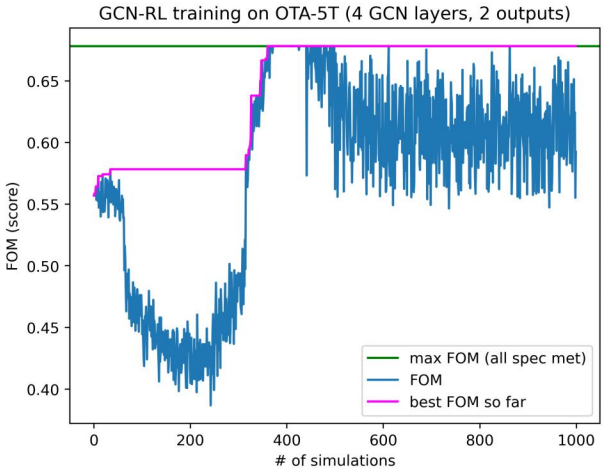

Figure 5. Learning Curve for OTA-5T with GCN-RL

GCN-RL and NRL 5 times each. We see that the average number of simulations for convergence is 243 for GCN-RL for OTA-5T circuit, while NRL agent never converged even after $>$ 1200 simulations. This indicates that using the circuit-graph information is critical to the learning process.

5. Future Work

The current GCN-RL implementation in literature (Wang et al., 2020) uses linear transistor parameters (V_{sat}, V_{th}) which do not capture behavior of non-linear circuits (comparators, oscillators etc). Additional non-linear features need to be explored further to support non-linear-circuits.

References

Lillicrap, T. P. et al. Continuous control with deep reinforcement learning. In *ICLR*, 2016.

Tabor, P. Github repo: https://github.com/philtabor/youtube-code-repository/tree/master/reinforcementlearning/ policygradient/ddpg/tensorflow2/pendulum.

Wang, H. et al. Learning to design circuits. In *NeurIPS Machine Learning for Systems Workshop,*, 2018.

Wang, H. et al. Gcn-rl circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning. In *Design Automation Conference*, 2020.

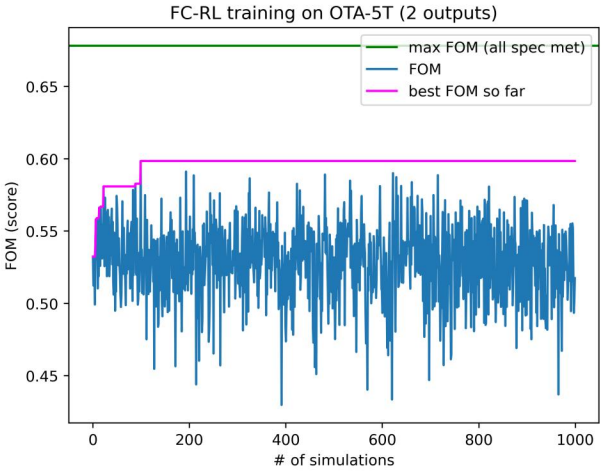

Figure 6. Learning Curve for OTA-5T with NRL

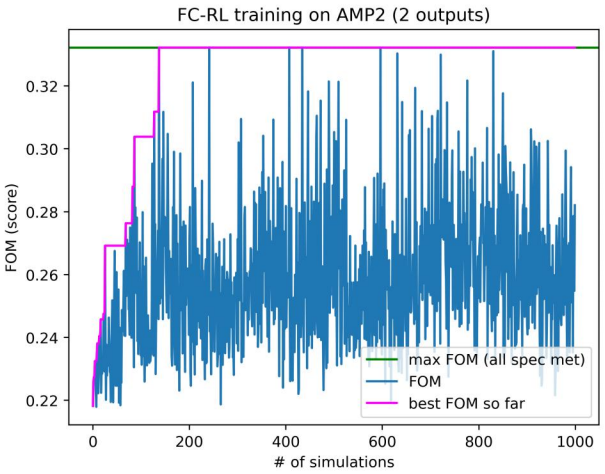

Figure 8. Learning Curve for AMP2 with NRL

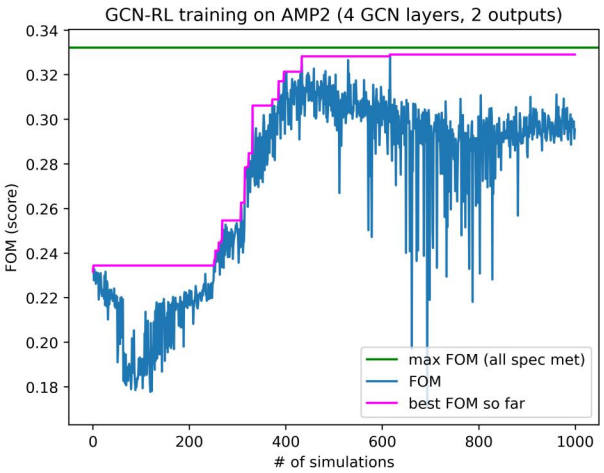

Figure 7. Learning Curve for AMP2 with GCN-RL

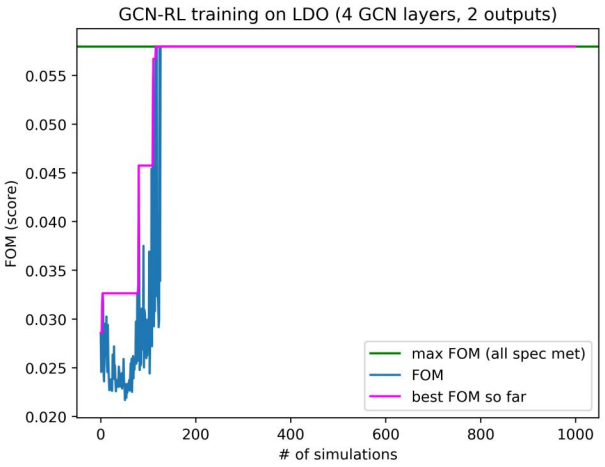

Figure 9. Learning Curve for LDO with GCN-RL

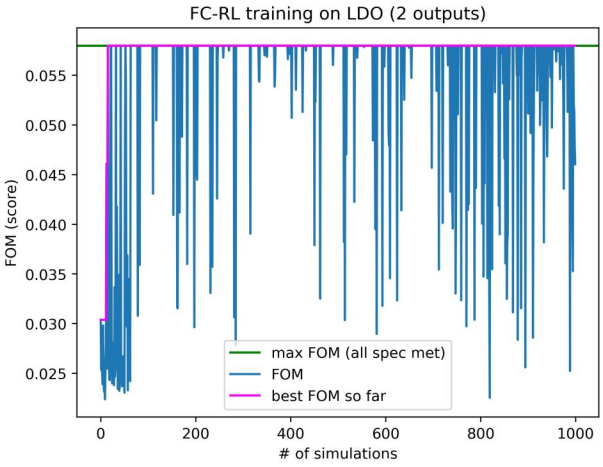

Figure 10. Learning Curve for LDO with NRL

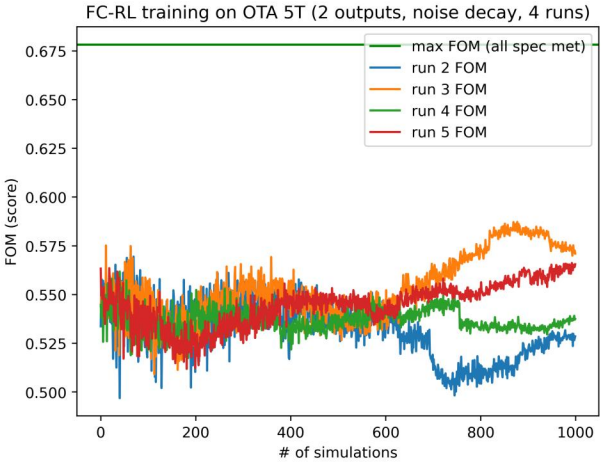

Figure 11. Learning Curve for OTA-5T with NRL for 5 runs

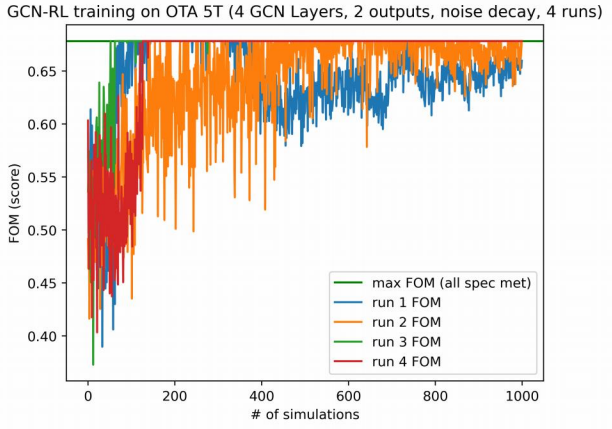

Figure 12. Learning Curve for OTA-5T with GCN-RL for 5 runs