GaN Distributed RF Power Amplifier Automation Design with Deep Reinforcement Learning

Sun, Yuxiang; Benosman, Mouhacine; Ma, Rui

TR2022-074 July 23, 2022

Abstract

Radio frequency (RF) circuit design demands rich experience of practical know-how and extensive simulation. Complicated interactions among different building components must be considered. This becomes more challenging at higher frequency and for sophisticated circuits. In this study, we proposed a novel design automation methodology based on deep reinforcement learning (RL). For the first time, we applied RL to design a wideband non-uniform distributed RF power amplifier known for its high dimensional design challenges. Our results show that the design principles can be learned effectively and the agent can generate the optimal circuit parameters to meet the design specifications including operating frequency range (2-18GHz), output power (>37dBm), gain flatness (<4dB) and average return loss (>5.8 dB) with GaN technology. Notably, our well-trained RL agent outperforms human expert given the same design task, with 78% accuracy and offers generalizability, which is lacked in the conventional optimization approach to shorten the time-to-market.

International Conference on Artificial Intelligence Circuits and Systems (AICAS) 2022

GaN Distributed RF Power Amplifier Automation Design with Deep Reinforcement Learning

Yuxiang Sun*, Mouhacine Benosman, and Rui Ma
Mtsubishi Electric Research Laboratories (MERL), Cambridge, MA, USA (rma@merl.com)

***Abstract*— Radio frequency (RF) circuit design demands rich experience of practical know-how and extensive simulation. Complicated interactions among different building components need to be considered. This becomes even more challenging at higher operating frequency and for sophisticated circuits. In this study, we proposed a novel design automation methodology based on deep reinforcement learning (RL). For the first time, we applied RL to design a wideband non-uniform distributed RF power amplifier, which is known for its high dimensional design challenges. Our results show that the design principles can be learned effectively, and the agent can generate the optimal circuit parameters to meet the design specifications including operating frequency range (2-18GHz), output power (>37dBm), gain flatness (<4dB) and average return loss (>5.8 dB) with GaN technology. Notably, our well-trained RL agent outperforms human expert given the same design task, with 78% accuracy and offers generalizability, which is lacked in the conventional optimization approach to shorten the time-to-market.**

Keywords*— *RF circuits, deep reinforcement learning, automation design.

I. Introduction

High power, wide bandwidth, and high efficiency radio frequency power amplifier (RFPA) has been in the central demand of RF hardware for modern wireless transmitters. However, RFPA design is challenging due to various technical difficulties including complicated transistors operation and device parasitic. Recently, wideband non-uniform distributed power amplifier (NDPA) is empowered by advanced semiconductor devices, e.g., GaN high electron mobility transistors (HEMTs) to offer improved performance [1]. Nevertheless, NDPA design in practice requires sophisticated manual synthesis and tedious tuning of high dimensional circuit parameters such as impedance matching network [2, 3, 4] and transistor sizing. Optimal design aims to get a trade-off between various demanding specifications, which often needs exhaustive manual search, mainly due to the nonlinearities and complex correlations among circuit characteristics.

Very recently, artificial intelligent (AI) technique is being explored to address analog circuit design challenges via hardware design automation [5], which have been reported to be efficient and effective [5, 6, 7,13]. Nevertheless, to the authors' best knowledge, this exploration for radio frequency integrated circuits (RFIC) design is still very limited in terms of nonlinearity and circuit complexity. Hence, this current study focuses on how to leverage the powerful approximation ability of deep reinforcement leaning to address the highly nonlinear properties in RFIC such as power amplifier design. For the first time, we proposed a reinforcement learning-based algorithm to design NDPA automatically. In specific, given design specifications, fixed number of active transistor cells and topology, a trained deep RL model by applying our algorithm can generate the optimal device parameters that fulfill the design goals rapidly.

II. Manual Design of NDPA

A. Descriptions of NDPA Schematic

A typical schematic of NDPA with three cells is shown in Fig. 1. With the distributed structure, the effect of parasitic shunt capacitance of transistor could be reduced so that a higher cut-off frequency is achievable. The resistor in parallel RC network plays the role of stabilization, and the capacitor helps to cancel out the shunt parasitic capacitance of the transistors [8], to increase cut-off frequency. This effect could be further mitigated by the transmission lines (TL) deployed on both gate and drain sides. The non-uniform structure allows for different choices of transistor size, so that the left drain termination resistor in distributed power amplifier could be removed and more power could be delivered to the load.

B. Challenges in Manual Tuning

Based on the theory of NDPA design, with optimal combination of these parasitic-sensitive components, most of the power can be delivered without reflection between cells within the desired wide frequency range. However, the design (searching space) for the best combination is tremendous, and the conventional optimization method [3, 4] used in manual design is less effective, suffering from the curse of dimensionality. As a result, even for an experienced RF designer, the optimization could quickly become prohibitive when the number of cells exceeds certain number (i.e., >5), due to the intractable complexity. Moreover, optimization must be re-run every time when there is update on the design specification or circuit topology, stemming from a lack of design knowledge transfer, and generalizability.

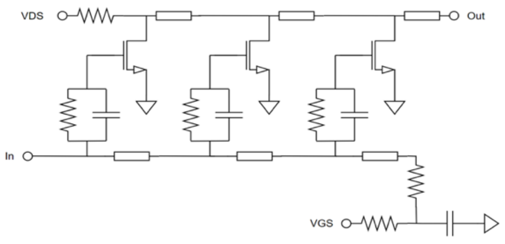

Fig. 1. A typical schematic of 3-cell non-uniform distributed PA.

III. RL Approach

A. Overview

The proposed RL algorithm is expected to perform design automation for NDPA, with suggested devices parameters within pre-defined ranges to guarantee manufacturability for reasonable design goals. Specifically, given a fixed topology of NDPA, once the RL model is well trained, the RL agent is

*This work was done during Yuxiang Sun's internship at MERL.

supposed to output the optimal set of all device parameters fulfilling the given design goals during the deployment phase. This can be done normally with a few hundred exploration trials within a couple of minutes using desktop PC.

B. The RL Algorithm Formulation

A standard environment for a reinforcement learning agent is formulated as a Markov decision process (MDP), defined by a tuple (S, A, R, γ) [9], where S is the state space. A represents the action space. R(s, a) is a reward function, scoring how far away the agent is approaching to design goal, and more details are given in the following section. Given an observable state $s \in S$, of the environment (which is an EDA simulator), the RL agent is supposed to learn a policy $\pi(a|s)$ to maximize the expectation of cumulative discounted rewards, where s_0 is drawn from a random initial distribution, and $\gamma<1$ is a discount factor for penalizing over optimism with regard to future rewards.

State: Denoted by a vector obtained by concatenating three types of features:

1) Node type encoding h_{type}, representing device types, e.g. the binary code [0, 0, 0] and [0, 0, 1] represent transistor and transmission line (TL) respectively.

2) Device parameters h_{para}, normalized by the upper bound of each specific parameter. For transmission line, the algorithm tunes the width and length simultaneously.

3) Circuits characteristics h_{spec}, S-parameter (SP) simulation results [S21, S11, S22], with 17 data points for each term over the working frequency band with step of 1 GHz. Note that HB (harmonic balance) simulation reveals more detailed information for power amplifier but time consuming. Hence, HB is replaced by SP simulations during training phase and only used for final validation after training.

Action: Discrete action space is adopted in the algorithm, in the form of vector e.g., [-2, -1, 0, 1, 2], with the options of decrementing or incrementing by one or two units or keeping the device parameters unchanged. The number of action options is a hyper parameter, determining how aggressively we expect the agent to act.

Reward function: The design of the reward function is empirical and greatly affects the RL learning efficiency. We choose the reward function as formulated in equation (1).

$$R=\sum_{i=1}^{6} w^i * r^i,\ \ r^i = min(exp((g^i\text{-}g^*)/g^*),\ 1.2), \quad (1)$$

where g^i is the i-th specification obtained under current policy, and g* is the ideal specification. This reward mapping encourages the agent to approach the goal with more reward due to the exponential function, and the value 1.2 (a free parameter) is the saturation value for discouraging too much overdesign once the specification is satisfied and encouraging robustness pursuit. Empirically, assigning different weights helps to guide the agent to pursuit the critical performance. We use the six SP simulation performance terms g*: [$S21_{min}$, $S21_{mean}$, $S21_{variance}$, $S11_{mean}$, $S22_{mean}$, flatness], with the empirical weights [5, 0.5, 1, 1, 1, 1.5]. If all of them are fulfilled, the learning episode terminates, and the obtainable reward should be above 10.

C. Model Embedding

Electronic circuit is naturally well-suited for graphical representation, which captures the interplay between nodes. Recent research work [6,13] has shown its effectiveness in analog circuits design, but at low frequency. We embed NDPA with GCN (graphical convolutional network) [10], with each device denoted by one node, plus three common DC biasing nodes (GND, VDS, VGS) connecting all cells with three layers of GCN embedding. The node feature is the concatenated vector [h_{type}, h_{para}].

Another embedding with MLP (multiple layer perceptron) is created for circuits characteristics h_{spec} to extract their highly coupled relations, with each layer being 128 neurons. Its last hidden feature representation is concatenated with the last hidden representation from GCN and fed as the input of the last fully connected layer for outputting the approximation of the RL value function.

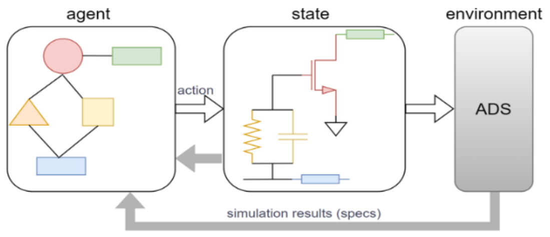

Fig. 2. RL agent interaction with high fidelity simulator (ADS).

D. Domain Knowledge Fusion

Domain knowledge of design principles fusion is proved to be able to accelerate RL convergences, by reducing inefficient state space exploration. In the case of NDPA, the left-most transistor should be the biggest size, contributing the most power. Therefore, the first constraint is setting the lower bound and upper bound of transistor width properly (note that transistor size decreases from left to right). Second, considering the current density of the drain line, as the power is flowing from left to right, the width of TL should be increased. Therefore, the width (W_DTL1) of the left most TL is set as the reference value, and its neighbor's width is set by an increment, i.e., W_DTL2 = W_DTL1 + ΔW. Additionally, we force all cells to have the same resistance and width of gain TL. All the constraint detailed information is listed in Table I.

TABLE I. DEVICE PARAMETERS RANGES FOR 3-CELL NDPA

No	*Transistor (μm)*	*Resistor (ohm)*	*Capacitor (fF)*	*TL-drain (μm)*	*TL-gate (μm)*
1	[120, 200]	[5,500]	[50, 4000]	W=[5,50] L=[50, 2000]	W=[5,50] L=[50, 1000]
2	[80, 120]	[5,500]	[50, 4000]	ΔW=[5, 50] L=[50, 2000]	W=[5,50] L=[50, 2000]
3	[40, 80]	[5,500]	[50, 4000]	ΔW=[5, 50] L=[50, 2000]	W=[5,50] L=[50, 2000]

E. RL Algorithm Descriptions

The training procedure of the proposed RL agent is depicted in Fig. 2. The agent is equipped with the latest learned policy based on the embeddings illustrated in section C and designed to output action to update the circuit parameters. Thus, a new schematic is generated and sent to

ADS for simulation, and the simulation performance results along with the new parameter state form the next state for another policy iteration.

We select the proximal policy optimization (PPO) [11], one on-policy gradient based RL algorithm, due to its advantage of higher data sampling efficiency than off-policy learning, like DDPG [12]. In PPO, there are two approximators: the "actor" is to make decision under learned policy. The "critic" is the state value function, estimating the expectation of return, after visiting the current state, indicating where to go for finding better states. Correspondingly, two objective functions are used to learn the actor-critic pair. The first objective function is formulated as follows,

$$L^{cllip}(\theta) = \mathbb{E}_t[min(b_t(\theta), clip(b_t(\theta), 1- \epsilon, 1+\epsilon)A_t] \qquad (2)$$

Where $b(\theta) = \pi(a|s;\theta)/\pi_{old}(a|s)$ is the ratio of new policy and old policy in terms of action probability distribution, and A_t is the advantage function, calculated as $A_t = V_\theta(s_t)$-G_t, $G_t = \sum\gamma^t * R(s,a)$, is discounted return in one episode. The clipping technique is constraint for avoiding aggressive policy updating, with a small $\epsilon = 0.2$. The estimation of $V_\theta(s)$ is learned by minimizing mean squared error of the two quantities, as formulated in the second loss function:

$$L^{VF}(\theta) = \mathbb{E}_t[V_\theta(s_t) - G_t]^2 \qquad (3)$$

The two loss functions are added up for joint minimization, $Loss = -L^{clip}(\theta) + L^{VF}(\theta)$.

IV. Training Details

We verify our RL agent performance in the case 3-cell NDPA design with GaN HEMT technology.

A. Experiment Setup

The interface of data stream is constructed between ADS and Python for the agent's interaction with environment. The targeted S-parameter specifications are listed in the second row of Table II, with a range for each specification to generalize for broad design goals. Thus, with a well-trained RL agent, one does not have to retrain from scratch for a different design goal. All the other hyper parameters related to RL algorithm are set the same as in the original PPO version, and the learning rate is chosen as a decaying function of time.

One critical setting is the start state initialization, which determines the policy exploitation and exploration, and a proper initialization manner can accelerate learning without loss of policy generalization. A "good" initial state defined as near to optimal solution should make the solution search faster, but this is found less effective for broad specifications coverage. The main reason is the lack of explosion of diverse states value. Considering the characteristics of our problem, we propose a dual-level weighted initialization method. At the high level, the probability that whether the initial state is initialized randomly or not, is drawn from a Bernoulli distribution, with probability p of random initialization and 1-p non-random initialization, where p decays in the exponential form: p=exp(-4*t/T), and t and T are the current episode index and the total number of episodes, respectively.

Next, a "non-random" initialization mode is used at the low level, where the initial state is sampled from one "successful" episode, in which the problem has been solved, and the chance of each state being sampled is associated with the weight with regard to reward: $\alpha*(R_t- R_{t-1}) + (1-\alpha)* R_t$, the probability distribution is formed with softmax function. The parameter α decays from 1: max(t/T, 0.2), where the t and T are used to denote the current step and the final step in one training episode. The intuition is that the state visitation demonstrating positive progress (increased R_t-R_{t-1}) should be more likely to be used for initialization at the early stage of training. While as the policy improves, the state with high reward is more likely to be used for fast policy convergence.

B. Training Procedure

The agent implements the latest policy, outputs the actions for all parameters, then the corresponding experience is stored into a temporary buffer for policy updating. The buffer size is the multiplication of the episode maximum length 30 and the maximum number of policy updates 30, namely, 900. The training takes roughly 12 hours for 2000 episodes with 8 CPU cores.

In Fig 3, we compare the effectiveness of two state initialization approaches: random initialization and the proposed advanced initialization. The average episodic reward about 9 is achieved at peak point with the proposed initialization, near 90% of full score 10. The average length is a measurement of how many steps are needed within one episode on average, this number drops after episode of 500, but it basically does not change in the random initialization method.

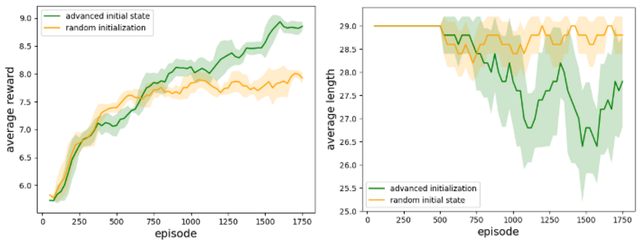

Fig. 3. RL agent learning curve: average reward and average length. Green: with the proposed state initialization; Orange: with random initial states.

C. Deployment Method

In deployment, given any specification, the agent with optimal policy is expected to iterate up to 200 steps to output the desirable optimal parameters. The initial state is sampled from the data set where good initializations states have been stored during the training phase. We deploy the agent up to 50 times given random design goals. The obtained deployment accuracy is reported in Fig. 4-(a). The agent can accomplish this design task with the rate of 78% at best (at episode=1500). For the 22% failures, we found that the specifications in these cases are very close to targets and can meet the goals with minor tuning. For instance, if the "failed" parameters are set as the initial sate for another round of deployment, they will be fine-tuned to satisfy all the desired performances. Fig. 5 demonstrates how broad specifications the RL agent can cover. Compared to the training goal, it shows that the mean values are basically satisfied, and the variance indicates that the RL agent can cover more desired specifications.

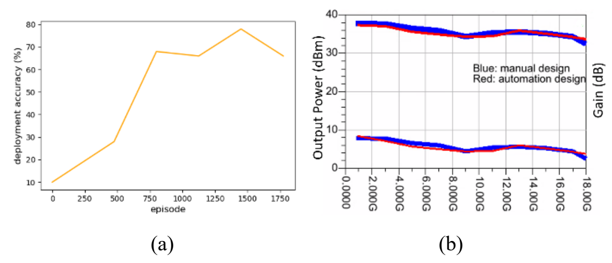

Fig. 4. (a): deployment evaluation and (b): one design case comparison.

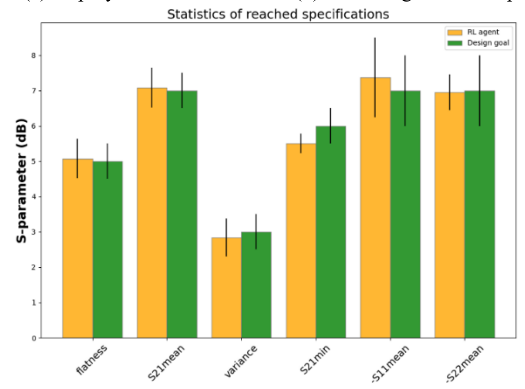

Fig. 5. Comaprison: statistics of specifications reached by RL agent at episodoe = 1500 vs. predefined design goals.

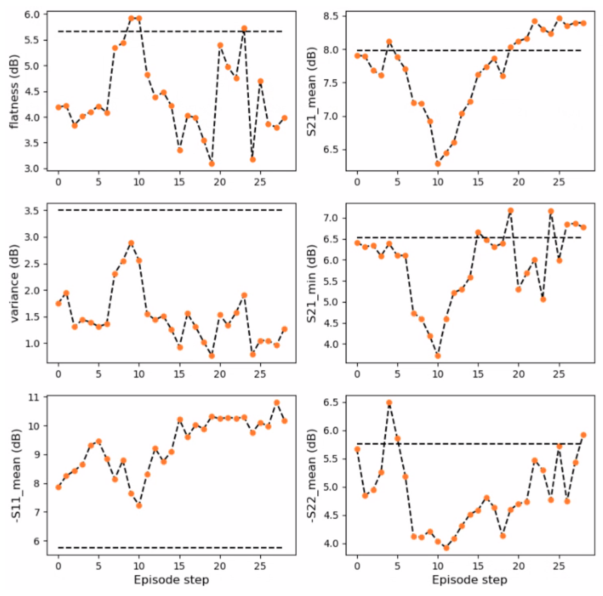

Fig. 6. The succesful deployment given desired sepcification from a human expert: horizontal dashline marks the given specificaiton.

D. Results and Analysis

For fair evaluation, we compare our automated design results with human expert design in one specific successful design case. Given the specification a human expert can reach (after weeks of tedious tunings) in Table II, a well-trained agent takes only 26 steps of trials to mostly achieve the same goal shown in Fig. 6. The HB simulation comparison is depicted in Fig. 4-(b). It demonstrates that the RL agent is competitive to a human expert but with much shortened design cycle.

TABLE II. ONE EXAMPLE SP SIMULATION RESULTS COMPARISON

	$S21_{mean}$ (dB)	$S21_{min}$ (dB)	$S21_{var}$ (dB)	$S11_{mean}$ (dB)	$S22_{mean}$ (dB)	*flatness* (dB)
Training	[6,8]	[5,7]	[2,4]	[-9, -5]	[-9, -5]	[4, 6]
RL agent	**8.4**	**6.7**	**1.3**	**-10.4**	-5.8	**4.0**
Manual	8.2	5.9	2.0	-10	**-6.5**	4.2

V. Conclusions

We demonstrated a novel RL-based method for NDPA automation design coving 2-18GHz with GaN, which shows the effectiveness of RL agent learning ability in optimal device parameters' tuning, with only minor domain knowledge. Compared to a human expert, a well-trained RL agent displays fast problem-solving capability with high accuracy in achieving the design goals. This method can be applied to radio frequency amplifier development to reduce design cost and shorten development time.

References

[1] Ayasli, Yalcin, et al. "A monolithic GaAs 1-13-GHz traveling-wave amplifier." IEEE Transactions on Microwave Theory and Techniques 30.7 (1982): 976-981.

[2] Duperrier, Cedric, et al. "New design method of uniform and nonuniform distributed power amplifiers." IEEE Transactions on Microwave Theory and Techniques 49.12 (2001): 2494-2500.

[3] Beyer, James B., et al. "MESFET distributed amplifier design guidelines." IEEE Transactions on Microwave Theory and Techniques 32.3 (1984): 268-275.

[4] Campbell, Charles, et al. "A wideband power amplifier MMIC utilizing GaN on SiC HEMT technology." IEEE Journal of Solid-state circuits 44.10 (2009): 2640-2647.

[5] Cao, Weidong, et al. "NeuADC: Neural network-inspired synthesizable analog-to-digital conversion." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 39.9 (2019): 1841-1854.

[6] Wang, Hanrui, et al. "GCN-RL circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning." 2020 57th ACM/IEEE Design Automation Conference (DAC). IEEE, 2020.

[7] Somayaji, NS Karthik, Hanbin Hu, and Peng Li. "Prioritized reinforcement learning for analog circuit optimization with design knowledge." 2021 58th ACM/IEEE Design Automation Conference (DAC). IEEE, 2021.

[8] Wu, Haifeng, et al. "A 2 to 18 GHz compact high-gain and high-power GaN amplifier." 2019 IEEE MTT-S International Microwave Symposium (IMS). IEEE, 2019.

[9] Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

[10] Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." arXiv preprint arXiv:1609.02907 (2016).

[11] Schulman, John, et al. "Proximal policy optimization algorithms." arXiv preprint arXiv:1707.06347 (2017).

[12] Lillicrap, Timothy P., et al. "Continuous control with deep reinforcement learning." arXiv preprint arXiv:1509.02971 (2015).

[13] Cao, W., Benosman, M., Zhang, X., Ma, R., "Domain knowledge-based automated analog circuit design with deep reinforcement Learning", AAAI-22 Workshop (AI2ASE) Conference on Artificial Intelligence, February 2022 (to be published).