[졸업작품전 – 작품]

SMART – DQN 알고리즘을 이용한 아날로그 회로 최적화

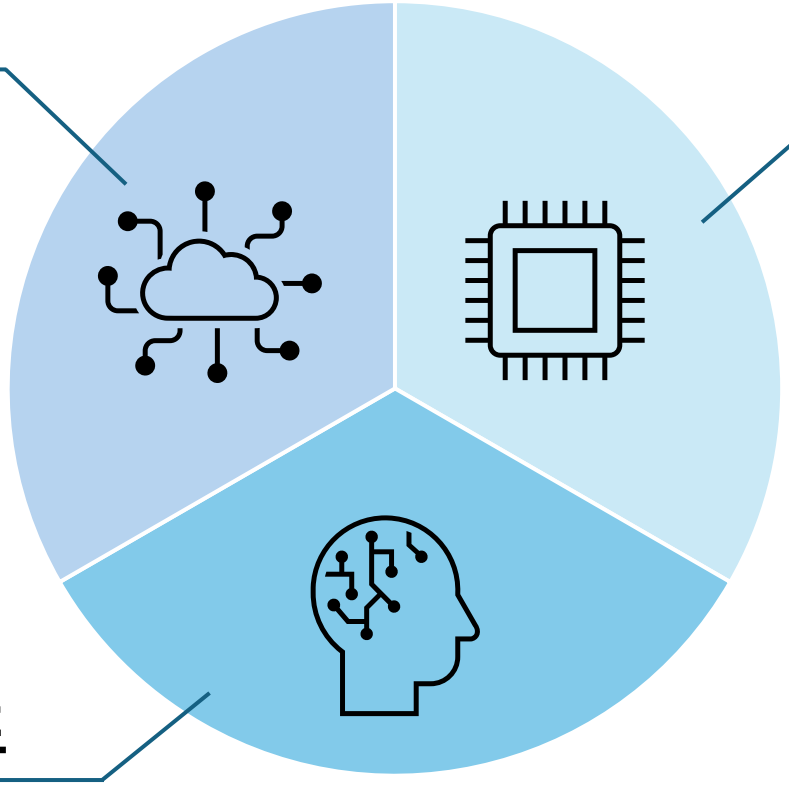
- 김도희 : 융합전자공학부 2021000482
- 이수훈 : 융합전자공학부 2019037965

목차

연구 개요

연구 배경

- 아날로그 회로는 설계자의 직관과 노동 집약적인 작업이 필요함
- 인공지능 기술이 발전하며 딥러닝과 강화학습을 활용한 학습 적용이 중요해짐.



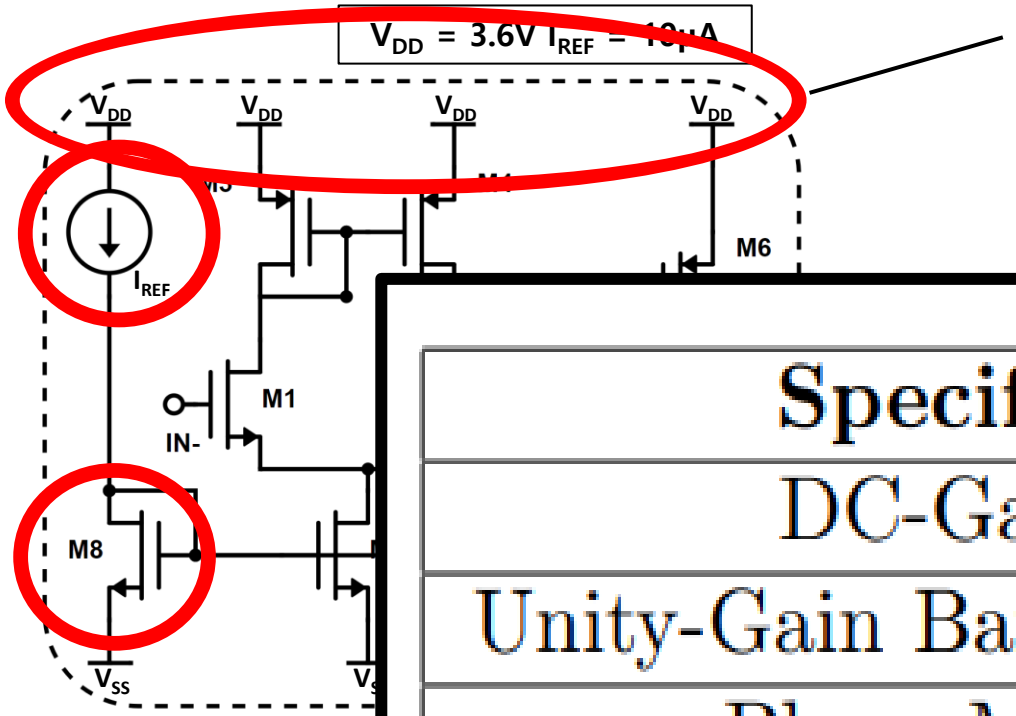
기존 연구 문제점

- 아날로그 회로 설계의 복잡성
- 기존 강화학습 알고리즘의 한계
- 학습 데이터 부족
- 병렬 처리의 제약

연구 목표

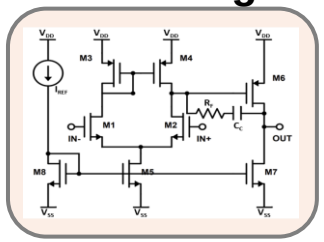
- 딥러닝과 강화학습을 이용하여 회로 설계를 자동화하고 나아가 더 빠르고 안정적으로 목표 스펙을 달성하는 알고리즘을 개발

회로 최적화 개요



Circuit
(2-stag

Circuit Design



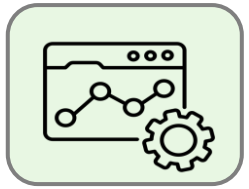
Netlist

Circuit Environment

- ❑ Parameters(W_1, L_1, C, R, *etc*)
- ❑ SMART – DQN

t Learning

Result

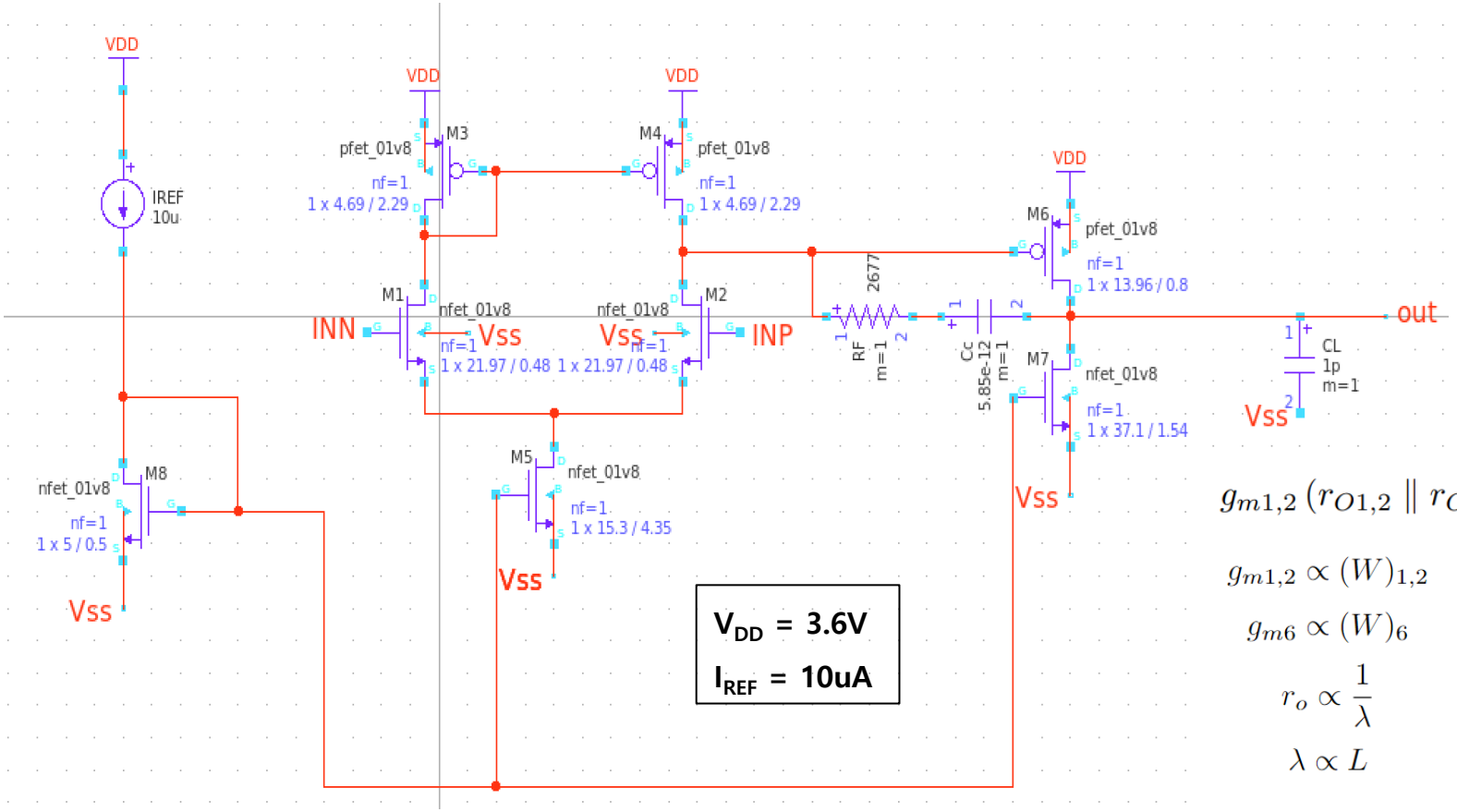


Optimized
Parameters

Specification	Desired
DC-Gain (dB)	≥ 70
Unity-Gain BandWidth (MHz)	≥ 1
Phase Margin (°)	50 - 80
Power (μW)	< 150

Human Designed 회로

❑ AI 알고리즘 기반의 자동 파라미터 조정에 앞서, 설계 지식을 바탕으로 목표 사양에 맞춰 파라미터 크기를 수동으로 조정하여 완성한 회로 설계



Design Process

1. Set the power budget to 100 µW.
2. Adjust the W and L ratios to increase gain, thereby increasing g_{mn1}, g_{mp6}, r_{on}, and r_{op} by individually increasing W and L.
3. Tune R_z to ensure $R_z > \frac{1}{g_{m6}}$, then increase C_c to achieve the desired phase margin.
4. Increase W across all stages to improve UGBW, acknowledging a trade-off in reduced gain.

$$g_{m1,2}\left(r_{O1,2} \parallel r_{O3,4}\right) g_{m6}\left(r_{O6} \parallel r_{O7}\right)$$

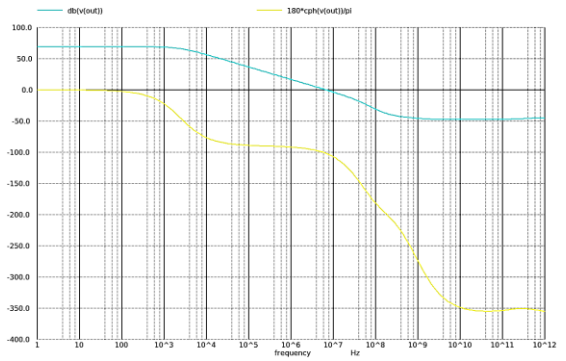
$$g_{m1,2} \propto (W)_{1,2}$$

$$g_{m6} \propto (W)_6$$

$$r_o \propto \frac{1}{\lambda}$$

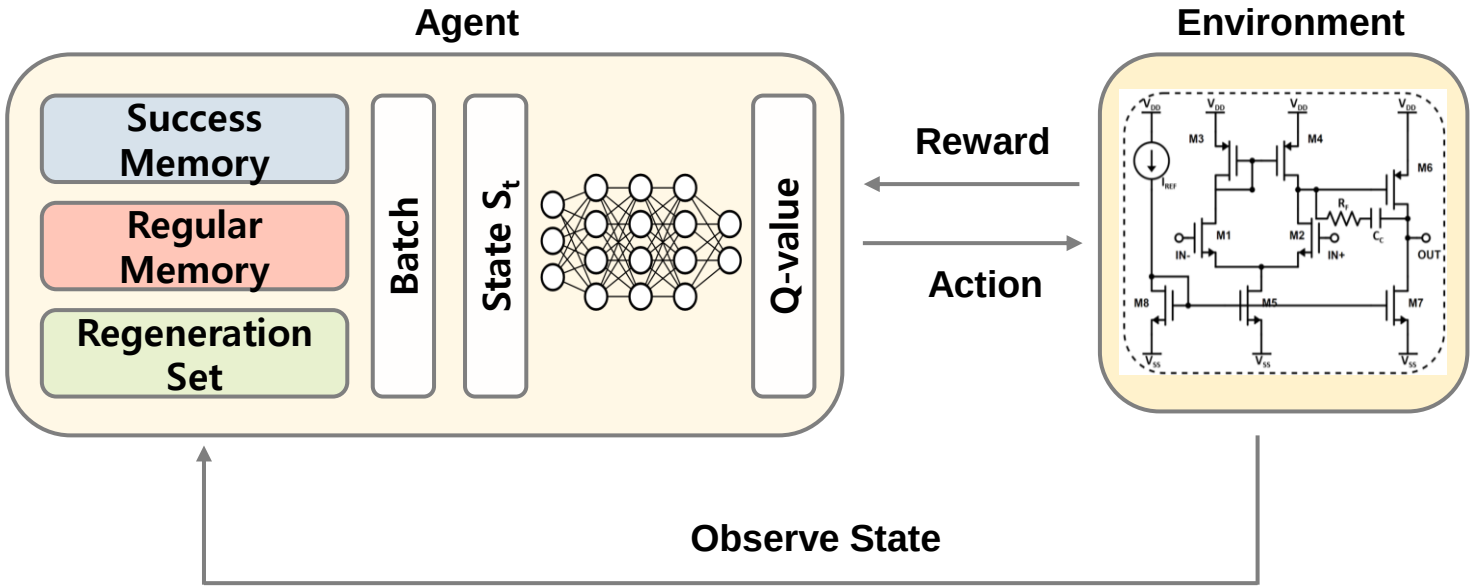
$$\lambda \propto L$$

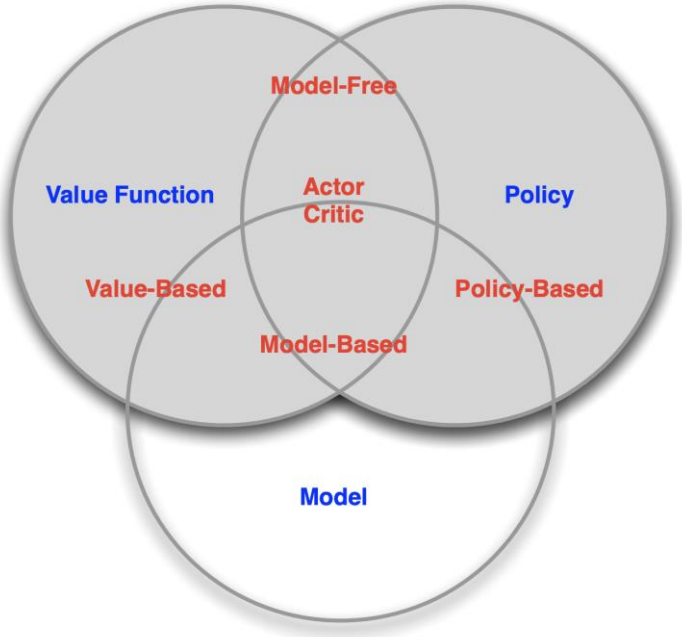
$$P = V_{DD} \times (I_5 + I_7)$$



알고리즘

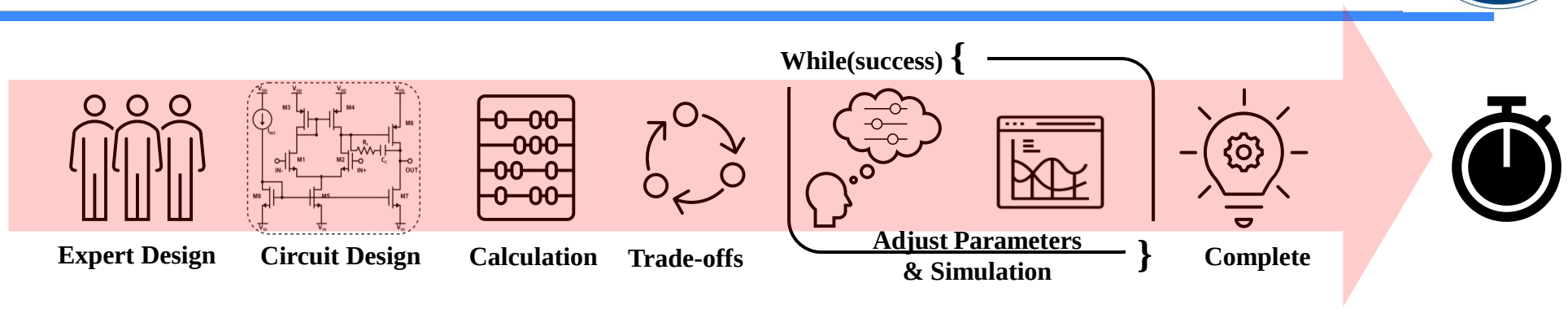
- Off-policy, Model-free인 DQN과 DDPG 알고리즘이 효율적
- 이산 공간을 탐색하는 DQN 알고리즘이 연속 공간을 탐색하는 DDPG 알고리즘보다 효과적
- DQN (Deep Q Network): Off policy로 동작하는 딥러닝과 강화학습을 결합한 알고리즘
- SMART-DQN (Success Memory And Regeneration-set Training DQN): 기존 알고리즘에 success memory 와 regeneration-set 을 도입한 알고리즘



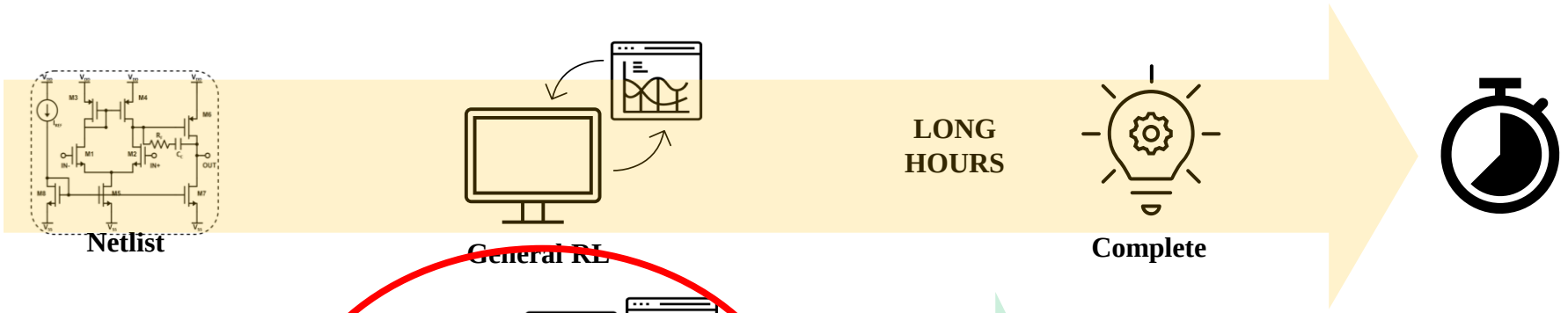


Process

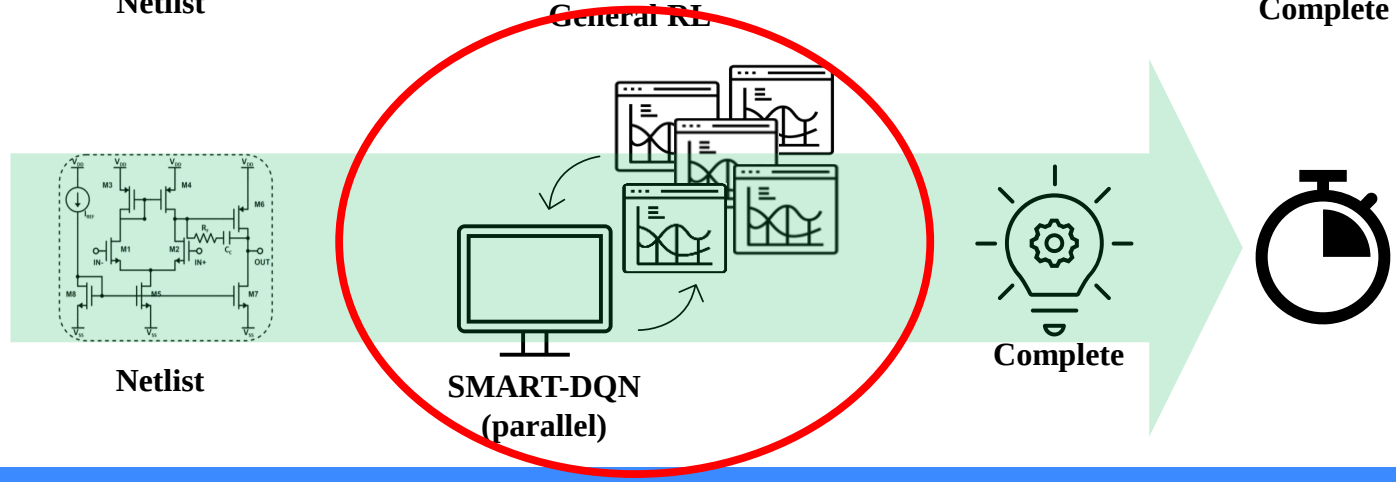
수동 아날로그 회로 설계



기존 강화학습 회로 최적화

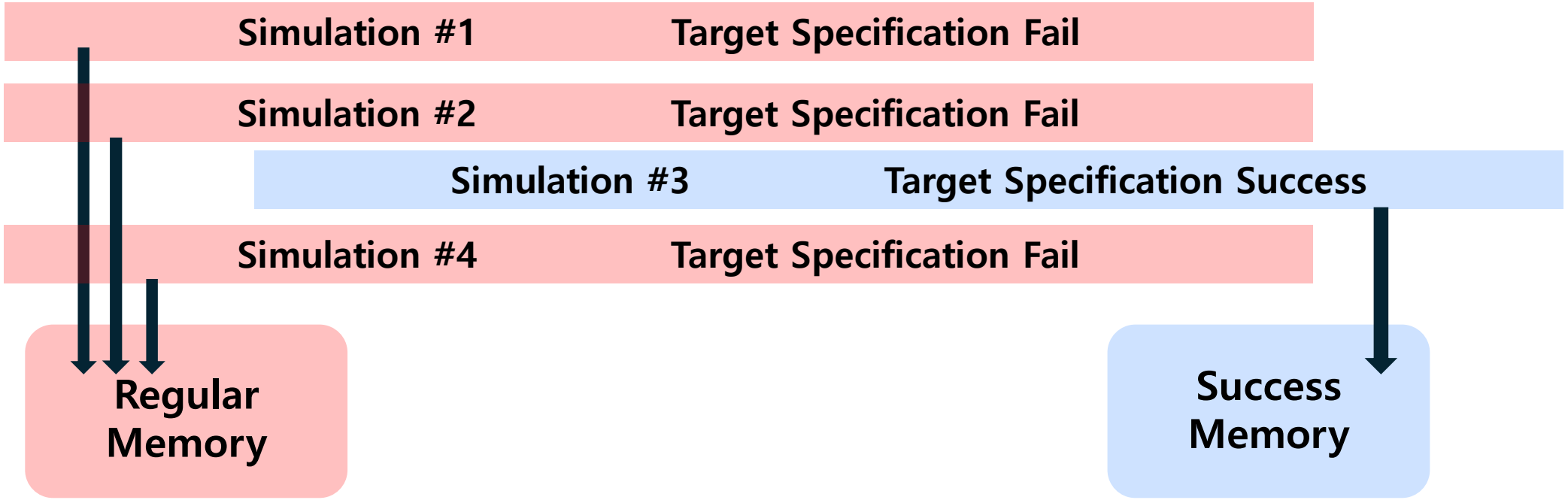


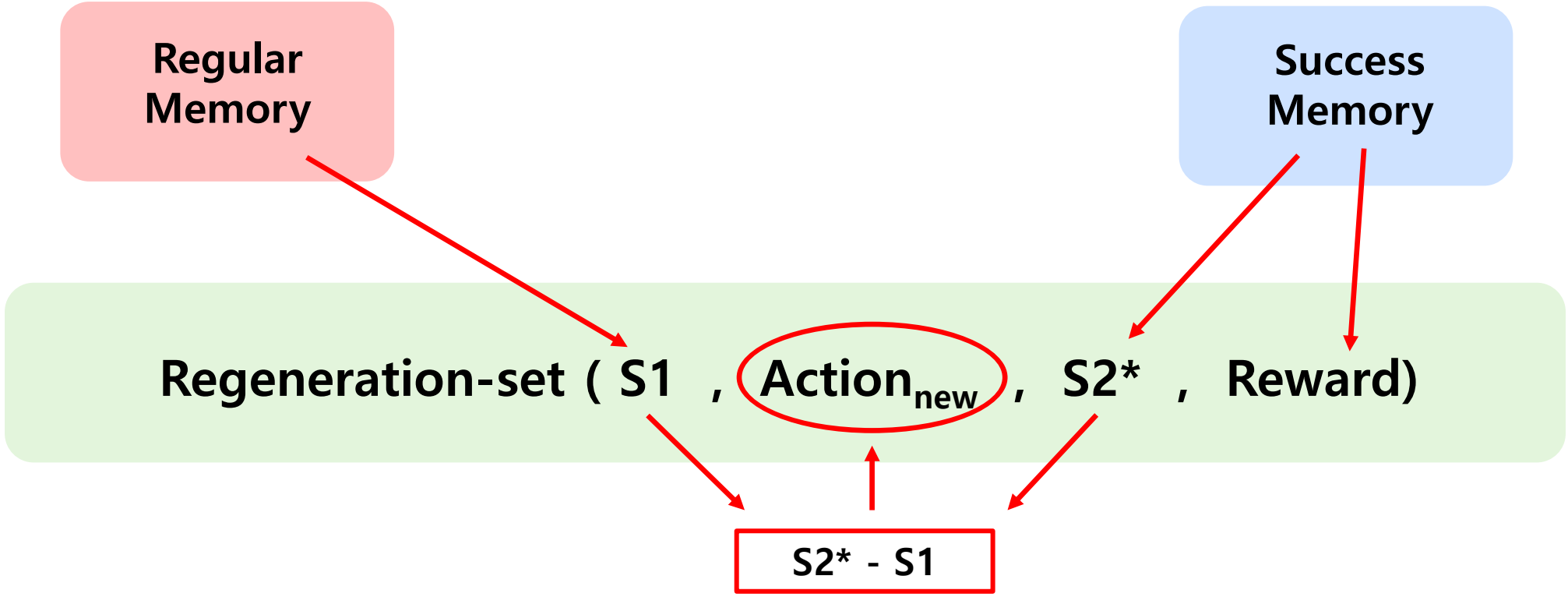
SMART-DQN 회로 최적화



Success Memory & Regeneration-set Training

- 문제: 아날로그 회로설계는 성공 경험이 아주 드물게 발생하는데, 이를 임의 추출하여 사용하는 경우 학습이 지연됨
- 알고리즘: 일반 메모리와 성공 메모리를 분리함. 초기 탐험 시에 목표 스펙을 만족하는 설계 경험을 성공 메모리에 저장하고 학습에 이용





- 문제: 시뮬레이터는 순차적인 특성을 가지므로 이것은 회로 설계와 AI의 co-design 시에 엄청난 bottle-neck으로 작용
- 알고리즘: 성공 데이터와 일반 데이터를 불러와 설계 상태의 차이와 성공 상태의 보상으로 시뮬레이션 없이 가상 데이터를 만듦

SMART - DQN

- 연구 성과:

① 원래 시뮬레이션은 순차적으로 진행되지만, 가상 데이터를 계속 생성하므로 병렬 시뮬레이터가 작동하는 효과 발생 -> 학습 속도 대폭 향상

② 성공 방향에 대한 가이드라인을 제시하여 학습 안정성 증가

③ 이전 시뮬레이션에만 의존하여 발생하는 데이터 부족 문제를 해결

Algorithm 1: Success Memory Update

Input: Current state s_t, action a_t, reward r_t, next state s'_t, done flag $done_t$
Output: Updated success memory ($success\_memory$)

```
if r_t ≥ r_threshold then
    Store the experience (s_t, a_t, r_t, s'_t, done_t) in success_memory;
end
return success_memory
```

Algorithm 2: Regeneration-Set Training

Input: Current state s_t, Success Memory, Regular Memory
Output: Regeneration Set (new experience)

```
if there are sufficient success memory and regular memory samples then
    Select an experience from regular memory:
      (s_1, a_1, r_1, s'_1, done_1) ← RandomPick(regular_memory);
    Select an experience from success memory:
      (s_2, a_2, r_2, s'_2, done_2) ← RandomPick(success_memory);
    Generate a new action a_new based on the state difference:
      a_new ← s'_2 − s_1;
    Assign a new reward r_new from the success experience: r_new ← r_2;
    Form a new experience (s_1, a_new, r_new, s'_2, False);
    Store the new experience for future training;
end
return new experience;
```

시뮬레이션 결과

	Variables	Min	Max
Parameter	W1	0.5	50
	L1	0.15	5
	W2	0.5	50
	L2	0.15	5
	W3	0.5	50
	L3	0.15	5
	W4	0.5	50
	L4	0.15	5
	W5	0.5	50
	L5	0.15	5
	W6	0.5	100
	L6	0.15	5
	W7	0.5	60
	L7	0.15	5
	Cc	1e-13	10e-12
	Rf	800	5000

Circuit Environment

Parameter	Human Designed Value	Optimized Value
$(W/L)_{1,2}$	21.97/0.48	36.3/2.3
$(W/L)_{3,4}$	4.69/2.29	33.7/1.7
$(W/L)_5$	15.3/4.35	32.8/3.7
$(W/L)_6$	13.96/0.8	56.4/1.8
$(W/L)_7$	37.1/1.54	16.3/2.0
C_C	5.85 pF	7 pF
R_F	2677 Ω	4320 Ω

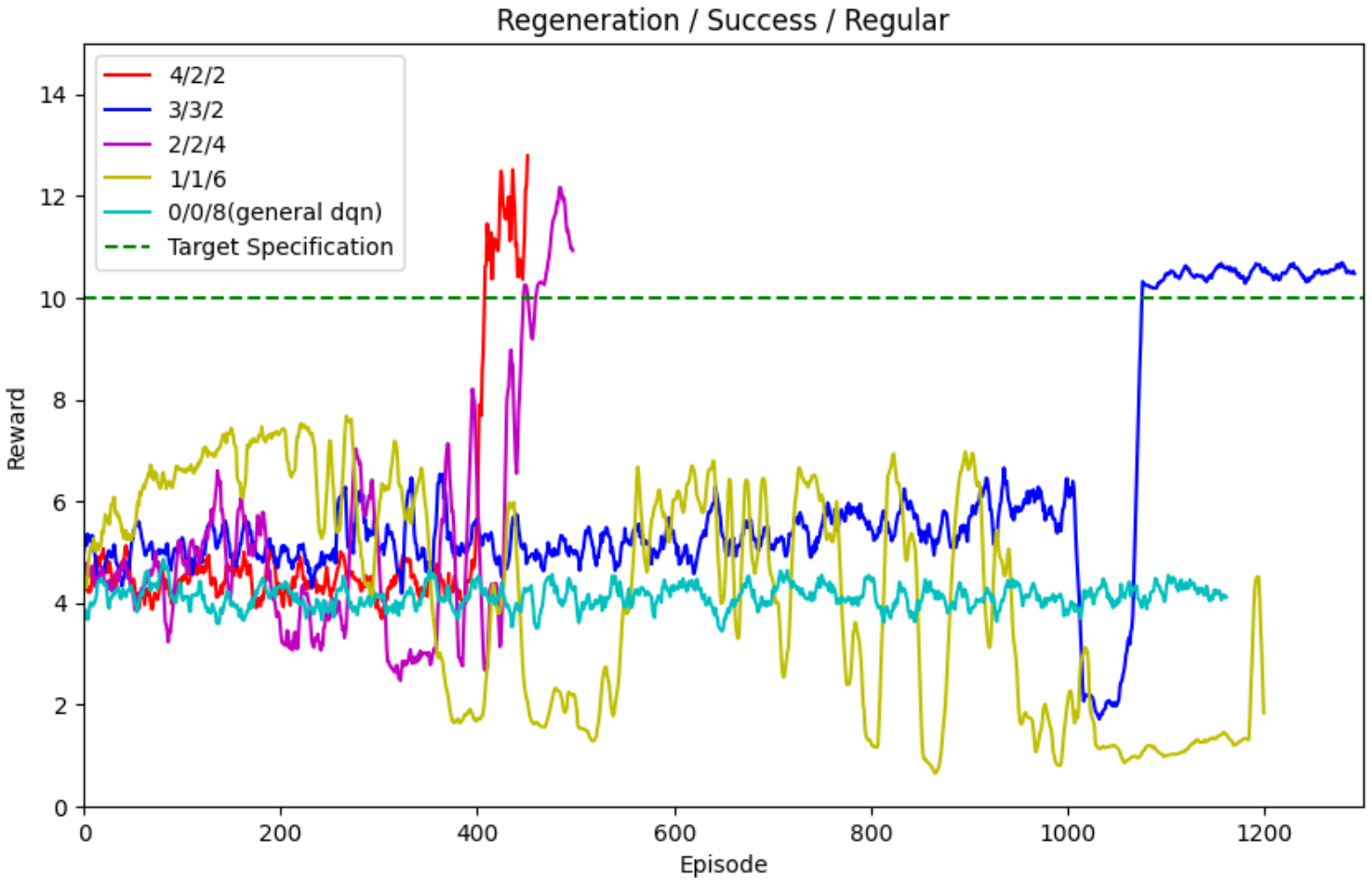
Optimized Parameter

Specification	Human Designed	Optimized
DC-Gain (dB)	82	92.5
Unity Gain Band Width (MHz)	4.6	12.8
Phase Margin (°)	72.62	68.4
Power (μW)	99.46	61.25
FoM	**3.79**	**19.33**

Target Specification

$$FoM = \frac{G \times UGBW}{P} \quad (50^\circ \leq PM \leq 80^\circ)$$

SMART – DQN performance 비교



논문 비교

① 설계 변수 2배 증가 및 환경 복잡도 증가

② SMART-DQN 을 이용하여 시뮬레이션 횟수 대폭 감소

③ Running Time 대폭 감소

④ 빠른 학습과 안정적인 수렴을 통해 회로 최적화 성능 대폭 증가

	Previous Work (2022)	**This Work**
# of Parameters	8	**16**
# of Simulations	160,000	**8,000**
Running Time	48h	**8h**

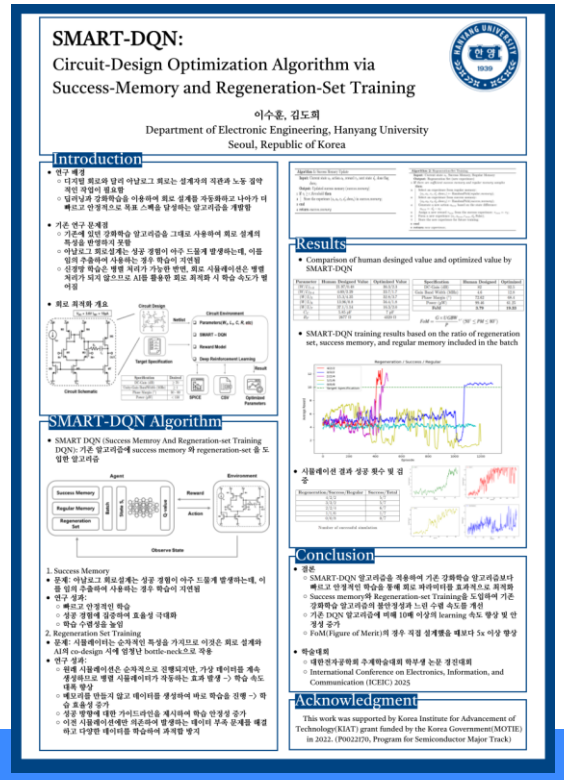
Comparison between Previous Work (2022) and This Work

Reference : 머신 러닝 기법을 이용한 아날로그 회로 설계 최적화 (2022년 융합전자공학부 졸업작품 최우수상작)

학술대회

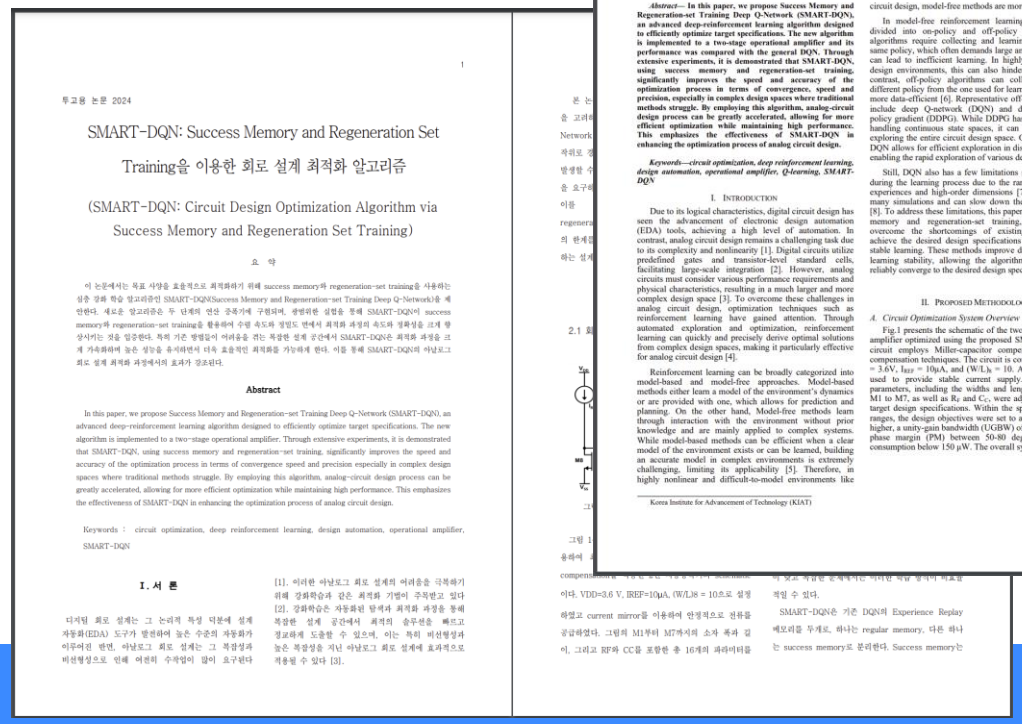
❑ 대한전자공학회 추계학술대회 학부생 논문 경진대회

❑ International Conference on Electronics, Information, and Communication (ICEIC) 2025


SMART-DQN:
Circuit-Design Optimization Algorithm via
Success-Memory and Regeneration-Set Training

이수훈, 김도희
Department of Electronic Engineering, Hanyang University
Seoul, Republic of Korea

Introduction
Results
SMART-DQN Algorithm
Conclusion
Acknowledgment


투고용 논문 2024

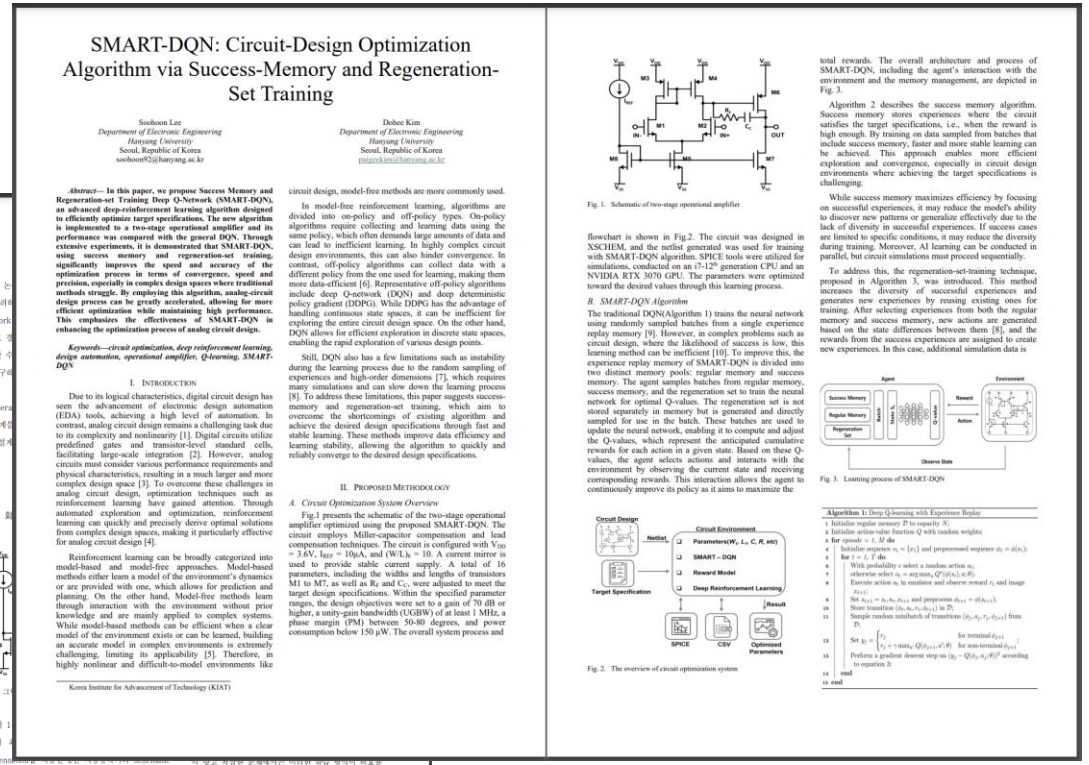
SMART-DQN: Success Memory and Regeneration Set Training을 이용한 회로 설계 최적화 알고리즘

(SMART-DQN: Circuit Design Optimization Algorithm via Success Memory and Regeneration Set Training)

요 약

Abstract

I. 서 론


SMART-DQN: Circuit-Design Optimization Algorithm via Success-Memory and Regeneration-Set Training

Soohoon Lee
Department of Electronic Engineering
Hanyang University
Seoul, Republic of Korea
soohoon92@hanyang.ac.kr

Dohee Kim
Department of Electronic Engineering
Hanyang University
Seoul, Republic of Korea

Abstract— **In this paper, we propose Success Memory and Regeneration-set Training Deep Q-Network (SMART-DQN), an advanced deep-reinforcement learning algorithm designed to efficiently optimize target specifications. The new algorithm is implemented to a two-stage operational amplifier and its performance was compared with the general DQN. Through extensive experiments, it is demonstrated that SMART-DQN, using success memory and regeneration-set training, significantly improves the speed and accuracy of the optimization process in terms of convergence, speed and precision, especially in complex design spaces where traditional methods struggle. By employing this algorithm, analog-circuit design process can be greatly accelerated, allowing for more efficient optimization while maintaining high performance. This emphasizes the effectiveness of SMART-DQN in enhancing the optimization process of analog circuit design.**

Keywords—circuit optimization, deep reinforcement learning, design automation, operational amplifier, Q-learning, SMART-DQN

I. Introduction

II. Proposed Methodology

A. Circuit Optimization System Overview

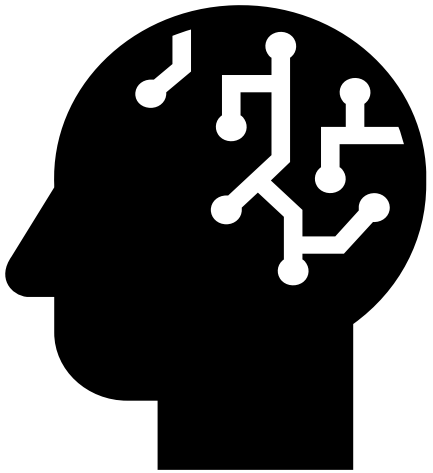
Korea Institute for Advancement of Technology (KIAT)

Fig. 1. Schematic of two-stage operational amplifier

Fig. 2. The overview of circuit optimization system

Fig. 3. Learning process of SMART-DQN

결론



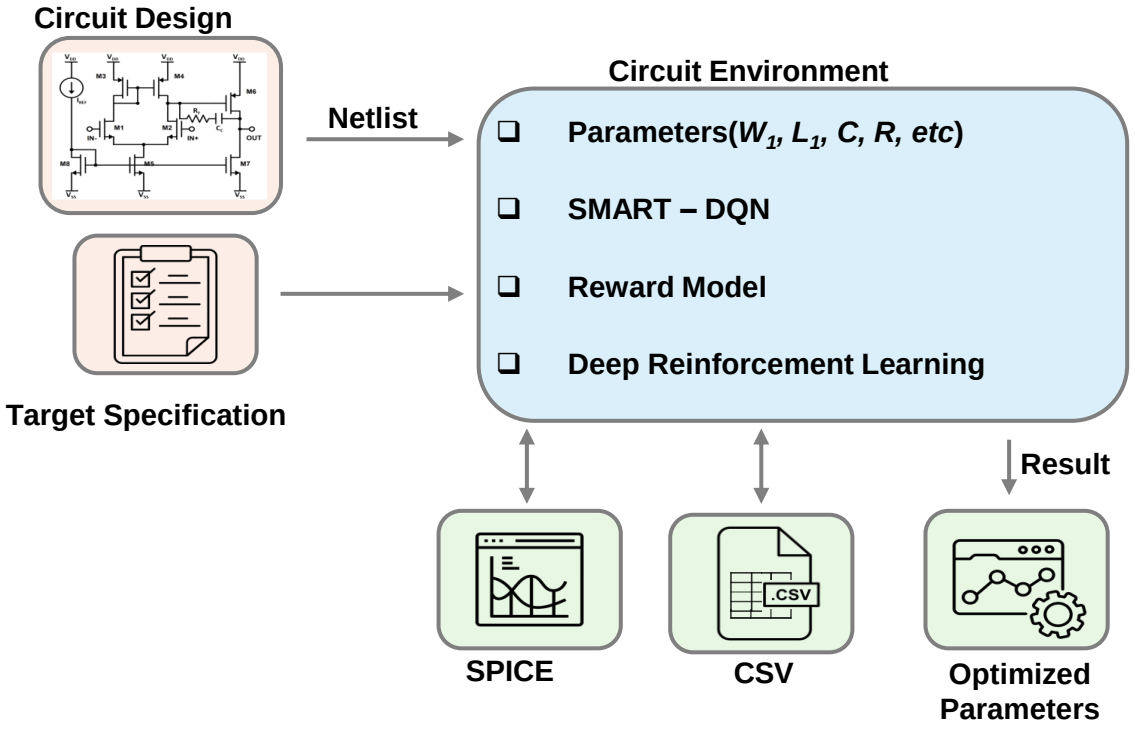
- SMART-DQN 알고리즘을 적용하여 기존 강화학습 알고리즘보다 빠르고 안정적인 학습을 통해 회로 파라미터를 효과적으로 최적화
- 시뮬레이터 병렬 동작 구현으로 회로 설계 강화학습의 병목 완화
- 기존 DQN 알고리즘에 비해 10배 이상의 learning 속도 향상 및 성공률 증가
- FoM(Figure of Merit) 수동 설계보다 5배 이상 향상

Q & A

감사합니다

Appendix

회로 최적화 개요



① Xschem을 활용하여 회로 설계

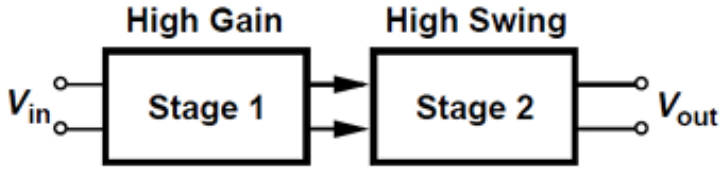
② 설계된 회로의 Spice파일을 python으로 연결

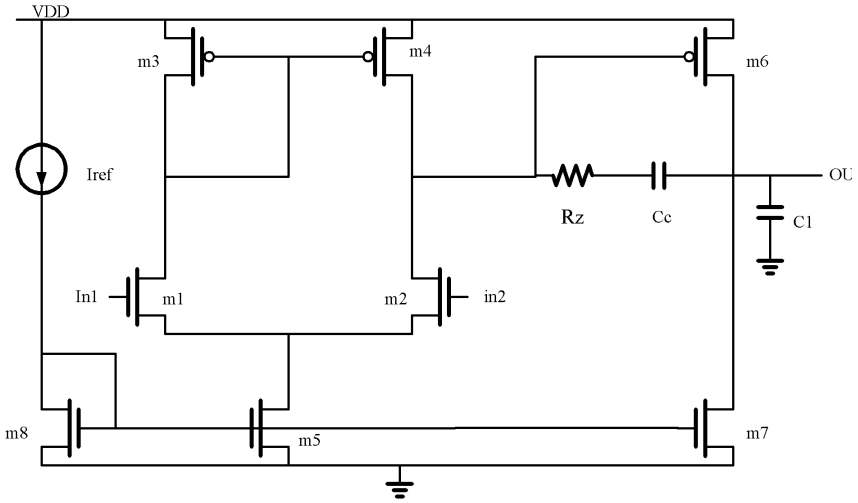
③ Target specification을 고려하여 구성된 Reward 함수를 바탕으로 회로 최적화

④ SMART-DQN 알고리즘으로 파라미터를 조정하며 강화학습

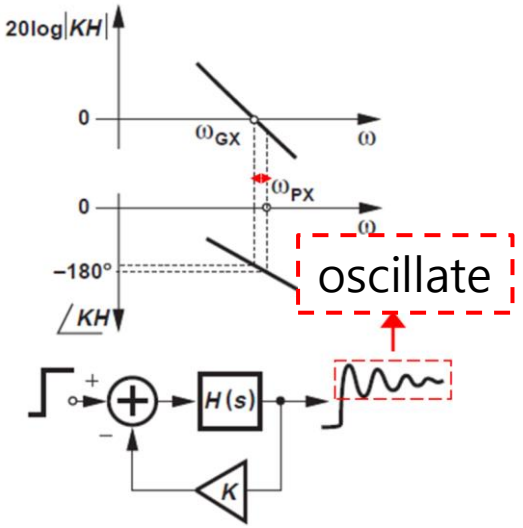
⑤ 1회의 에피소드 동안(시뮬레이션 20번) 모든 시뮬레이션이 목표에 도달하는 방향으로 학습

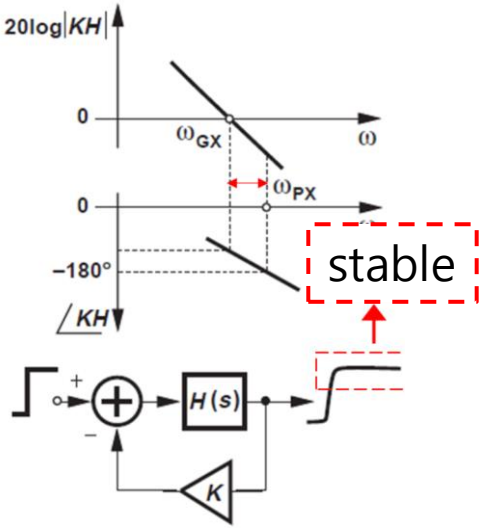
2단 연산 증폭기

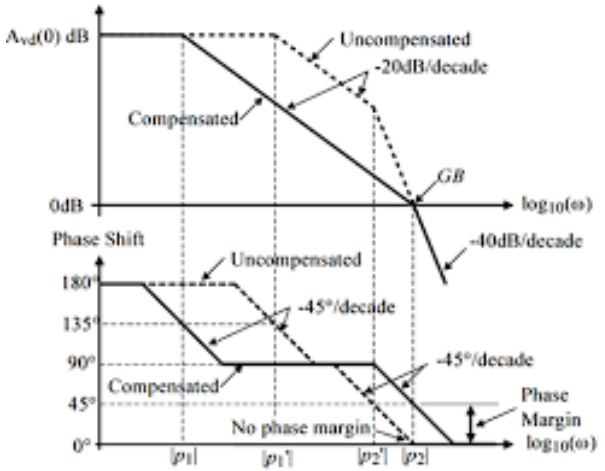




Phase Margin & Compensation method







Gain $g_{m1,2}\left(r_{O1,2} \parallel r_{O3,4}\right) g_{m6}\left(r_{O6} \parallel r_{O7}\right)$

Power $P = V_{DD} \times (I_5 + I_7)$

$$I_5 \propto \left(\frac{W}{L}\right)_5$$

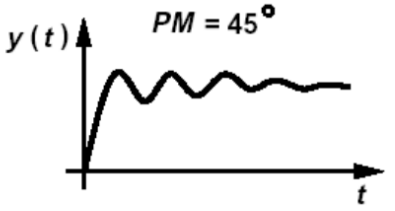
$$I_7 \propto \left(\frac{W}{L}\right)_7$$

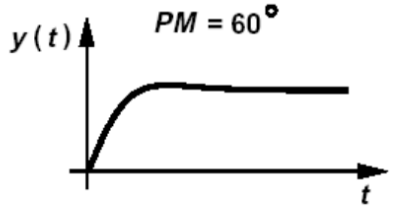
$$g_{m1,2} \propto (W)_{1,2}$$

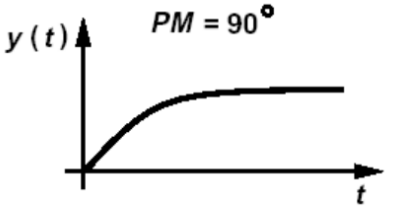
$$g_{m6} \propto (W)_6$$

$$r_o \propto \frac{1}{\lambda}$$

$$\lambda \propto L$$







SMART – DQN performance 비교 설명

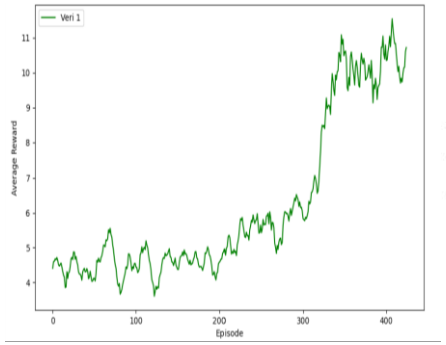
- Success memory와 Regeneration-set Training(SMART)이 절반 이상 포함되어 있을 경우 학습 성공률이 약 10.6배 향상하였다.
- 일반 DQN 알고리즘은 학습이 되지 않거나 수렴하지 못하며 불안정성을 보인 반면, SMART-DQN 으로 학습한 것은 10배 이상의 속도를 보이며 최적화된 지점으로 수렴한다.
- 한 에피소드는 20번의 시뮬레이션으로 구성되고, Target specification은 20번의 시뮬레이션에서 모든 목표 스펙을 만족한 경우를 의미한다.
- 일반 DQN의 경우 변수가 많고 복잡한 환경에서 Target specification 에 도달하지 못했다.

Hyperparameter

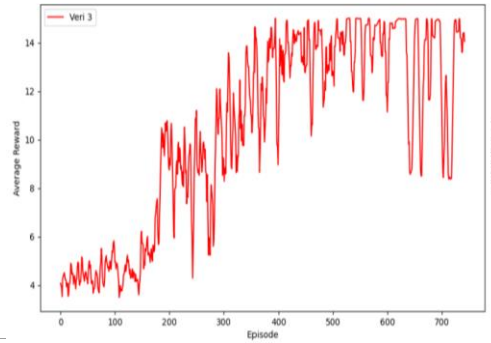
Hyperparameter	Value
Layers	3
Input Layer Neurons	16
Hidden Layer Neurons	64
Output Layer Neurons	16
Learning Rate (lr)	0.001
Learning Rate Scheduler	0.99 per 64 steps
Gamma	0.95
Tau	1×10^{-3}
Update Period	4 steps

검증

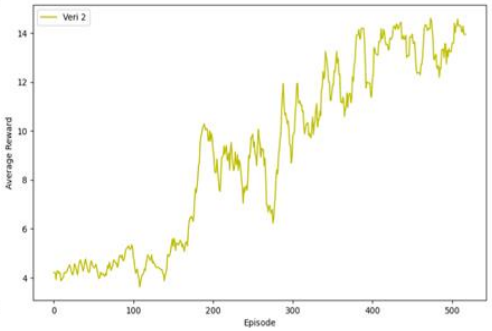
2/2/4



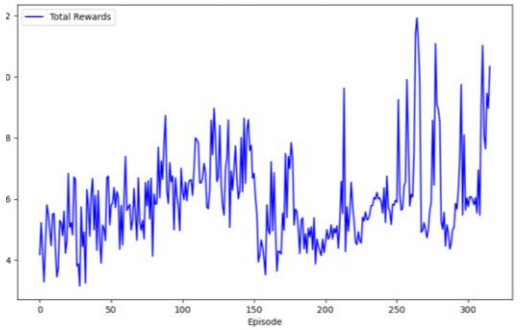
3/3/2



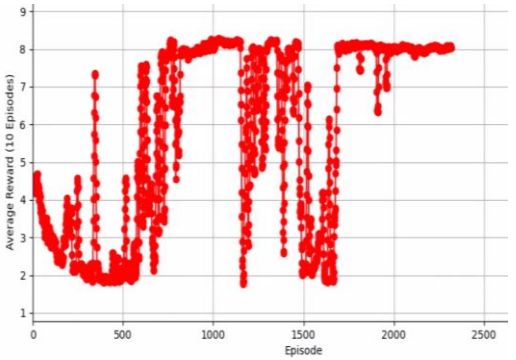
4/4/2



2/2/4



1/1/6

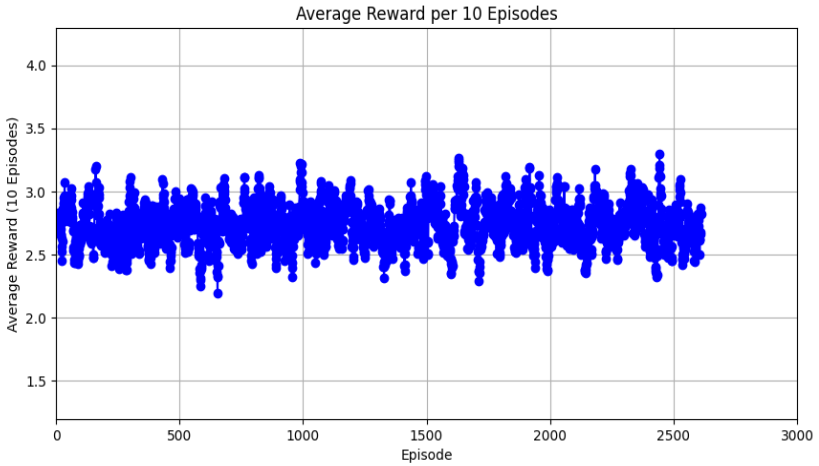


Regeneration/Success/Regular	Success/Total
4/2/2	5/7
3/3/2	5/7
2/2/4	6/7
1/1/6	1/7
0/0/8	0/7

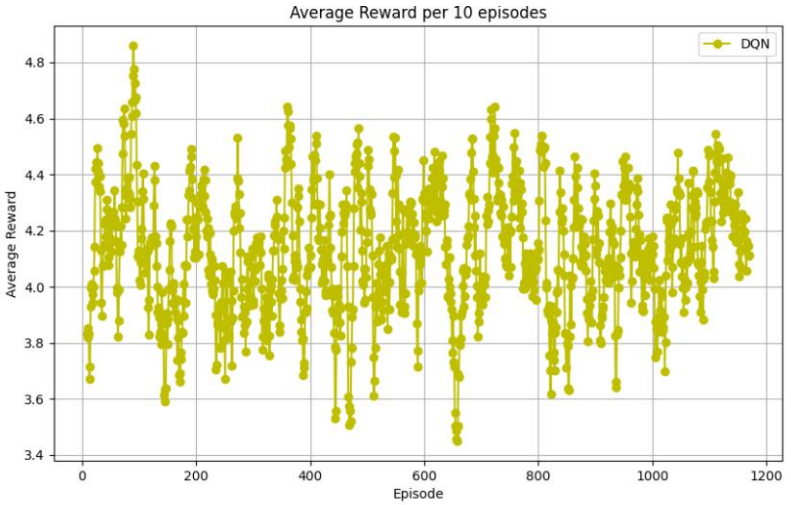
Number of successful simulation

Reward Graph Comparison

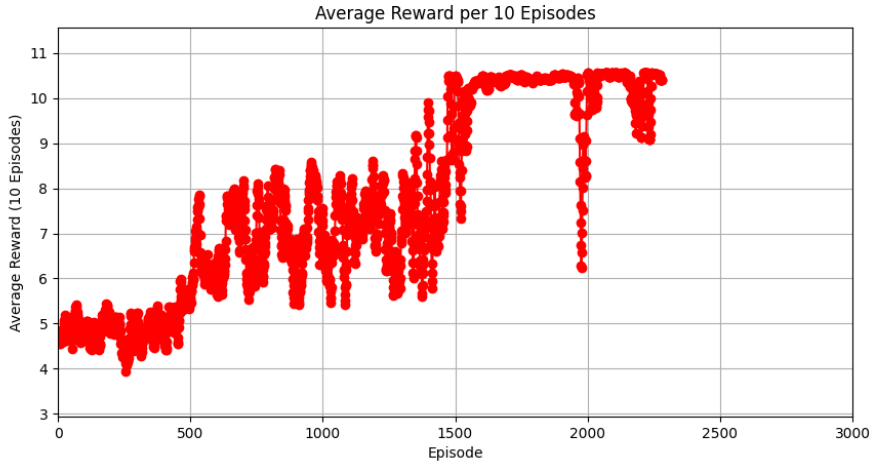
General RL algorithm (PPO)



General DQN



SMART - DQN



- 아날로그 회로 설계에서 정책을 학습하는 것은 무의미함. 따라서 최적의 Q-value를 찾아가는 DQN알고리즘이 탐색 측면에서 우수함
- 일반 DQN 알고리즘은 학습이 제대로 되지 않거나 불안정함. 따라서 SMART-DQN을 사용하면 안정적으로 수렴하며 최적화되는 경향을 보임.

코드 구현

Top Level

✓ Main.py: control unit

Low Level

✓ Reader.py
- Read and write data from csv

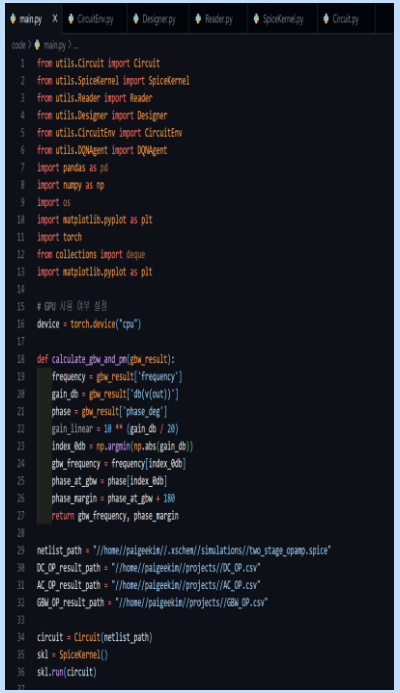
✓ Spicekernel.py
- Run simulation with Ngspice

✓ Circuit.py
- Read spice and connect with python

✓ Designer.py
- Designing parameter

✓ CircuitEnv.py
- Custom environment

✓ DQNAgent.py
- SMART-DQN algorithm



Simulation Environment

❑ Simulation Environment Setup

- Ubuntu-based setup using WSL with essential circuit design tools like xschem, open pdks, and ngspice.

❑ Python-Based Optimization Framework

- Deep reinforcement learning implemented in Python with PyTorch to optimize circuit parameters.

❑ System Integration for Efficient Simulation

- Seamless integration of Python with xschem and ngspice for real-time simulation and optimization.

❑ Hardware Setup

- Intel i7-12th Gen CPU and an NVIDIA RTX 3070 GPU for accelerated computation.

Reference

[1] W. Cao, M. Benosman, X. Zhang, R. Ma, "Domain knowledge-based automated analog circuit design with deep reinforcement learning," arXiv, vol. 2022.13185, Feb. 2022.

[2] H. Kaeslin, Digital Integrated Circuit Design: From VLSI Architectures to CMOS Fabrication, Cambridge University Press, 2008, pp. 386-458.

[3] B. Razavi, Design of Analog CMOS Integrated Circuits, 2nd ed., McGraw-Hill Education, 2016, ISBN 978-1259255090.

[4] K. Somayaji N.S., H. Hu, and P. Li, "Prioritized reinforcement learning for analog circuit optimization with design knowledge," 58th ACM/IEEE Design Automation Conference (DAC), pp. 1-6, Dec. 2021.

[5] Wang, T., Bao, X., Clavera, I., Hoang, J., Wen, Y., Langlois, E., Zhang, S., Zhang, G., Abbeel, P., & Ba, J., “Benchmarking model-based reinforcement learning,” ICLR Conference, 2020.

[6] Hausknecht, M., & Stone, “On-policy vs. off-policy updates for deep reinforcement learning,” IJCAI 2016 Workshop: Deep Reinforcement Learning, 2016.

[7] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in Proceedings of the International Conference on Learning Representations (ICLR), 2016.

[8] J. Hong, S. Kim, J. Kim, and D. Jeon, "Fast automatic circuit optimization using deep learning," Proceedings of the International SoC Design Conference (ISOCC), pp. 207-210, 2021.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," arXiv preprint arXiv:1312.5602, 2013.

[10] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," International Conference on Learning Representations (ICLR), 2016.