



[ 졸업작품전 - 작품 ]

# SMART - DQN 알고리즘을 이용한 아날로그 회로 최적화

- 김도희 : 융합전자공학부 2021000482
- 이수훈 : 융합전자공학부 2019037965



# 목차

---

- I. Introduction**
- II. Circuit Design**
- III. SMART – DQN Algorithm**
- IV. Result & Novelty**
- V. Conclusion**
- VI. Appendix**

# 연구 개요

## 연구 배경

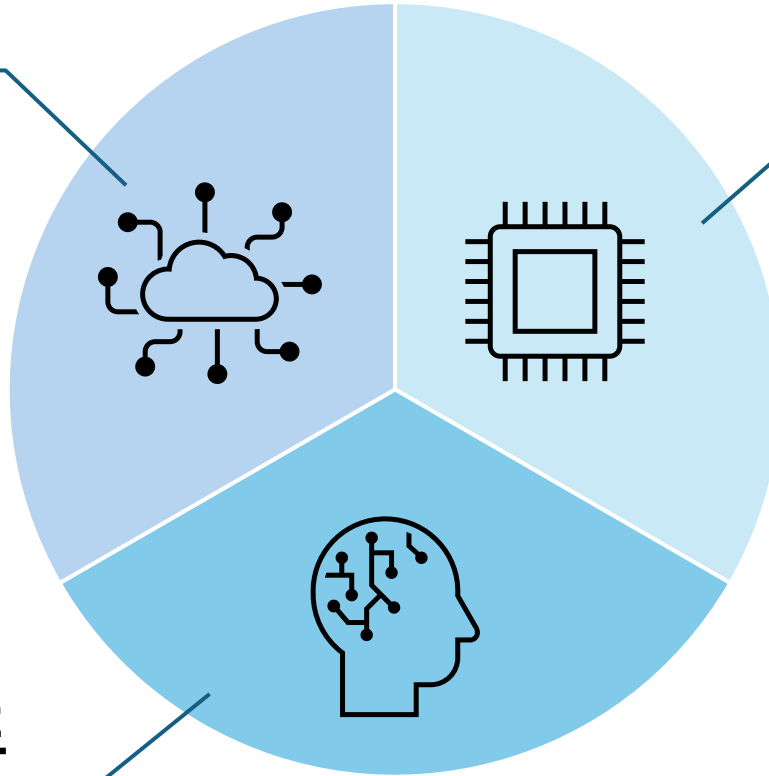
- 아날로그 회로는 설계자의 직관과 노동 집약적인 작업이 필요함
- 인공지능 기술이 발전하며 딥러닝과 강화학습을 활용한 학습 적용이 중요해짐.

## 기존 연구 문제점

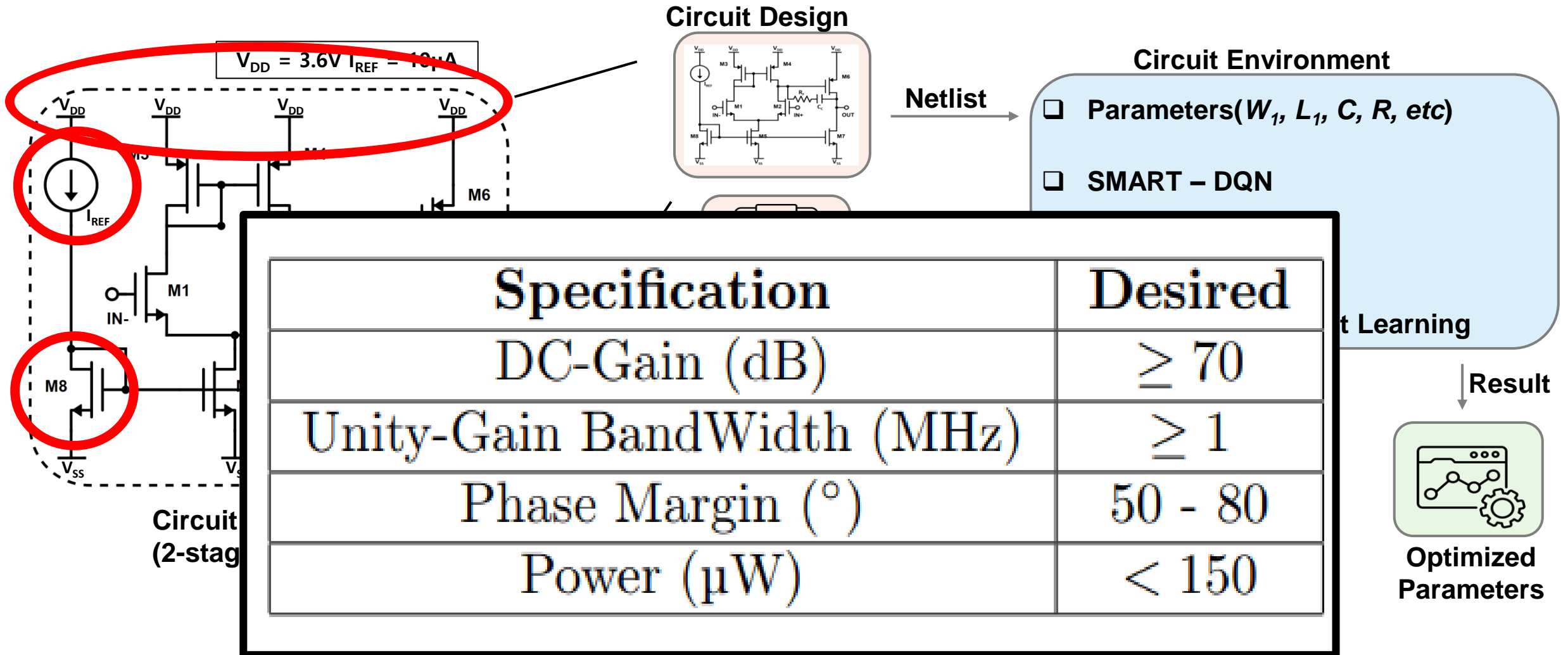
- 아날로그 회로 설계의 복잡성
- 기존 강화학습 알고리즘의 한계
- 학습 데이터 부족
- 병렬 처리의 제약

## 연구 목표

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

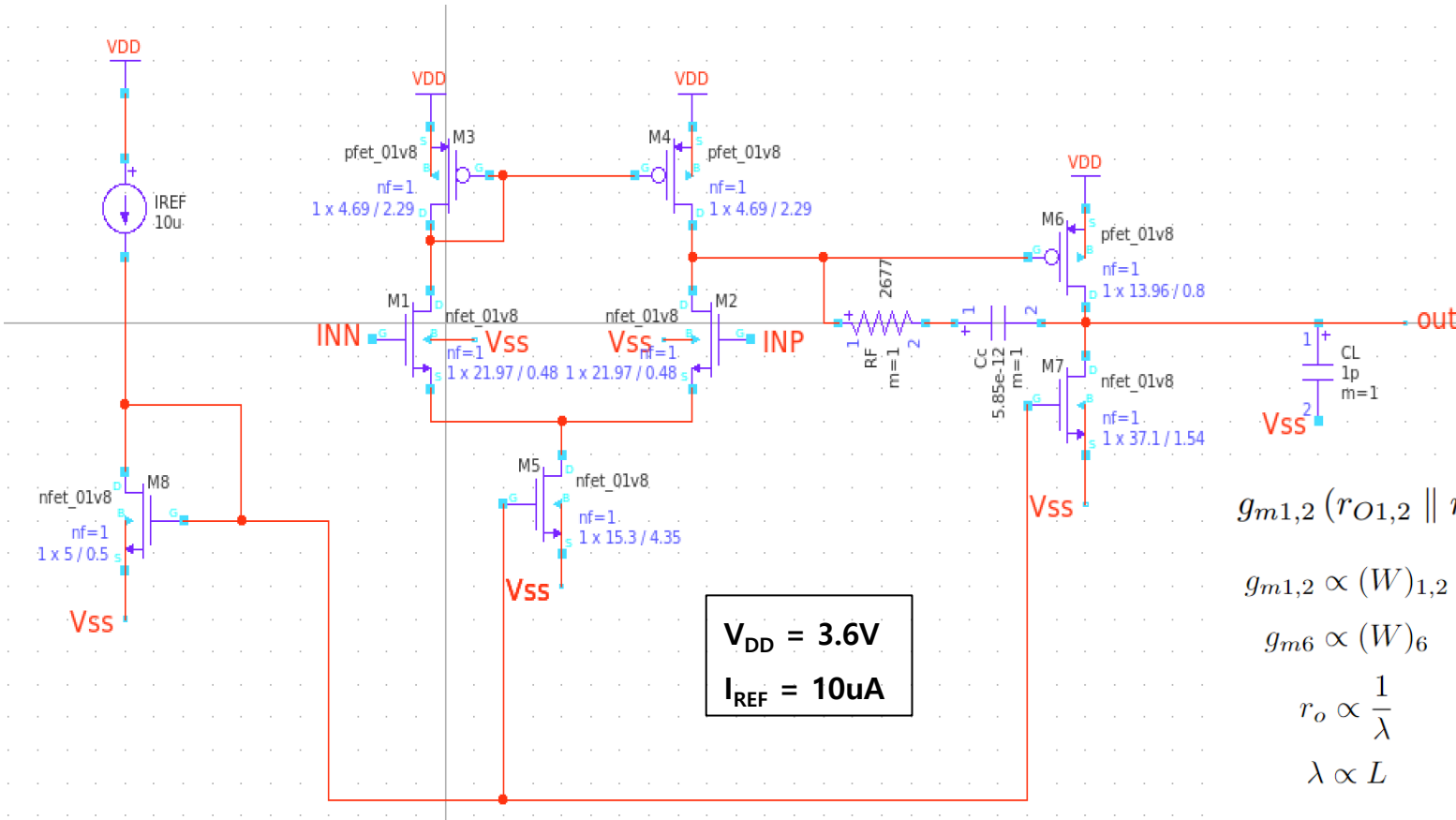


# 회로 최적화 개요



# 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  $\mu W$ .
2. Adjust the  $W$  and  $L$  ratios to increase gain, thereby increasing  $g_{m1,2}$ ,  $g_{m6}$ ,  $r_{on}$ , 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.

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

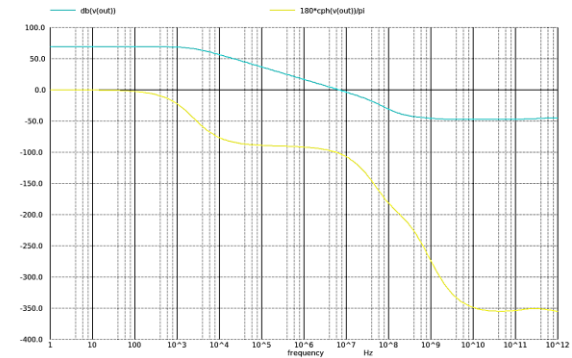
$$g_{m1,2} \propto (W)_{1,2}$$

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

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

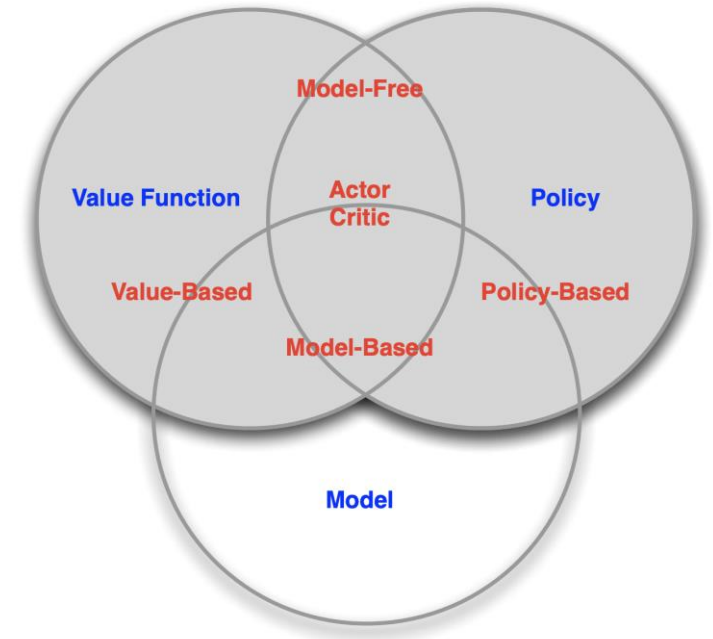
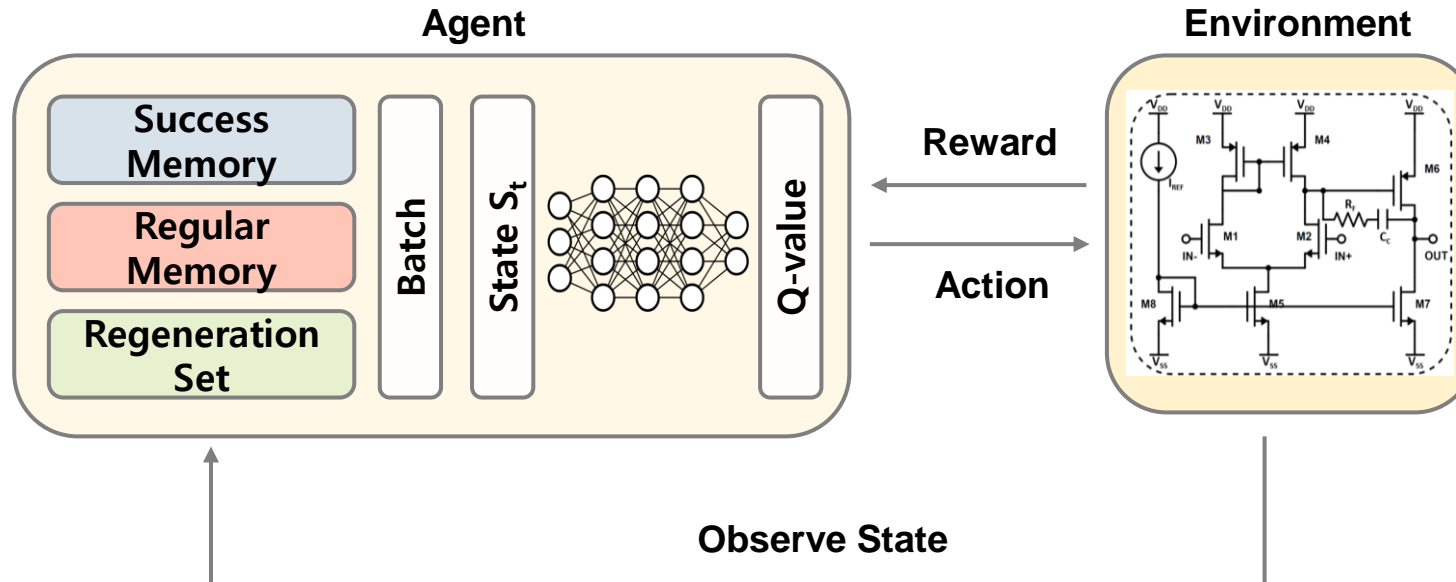
$$\lambda \propto L$$

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

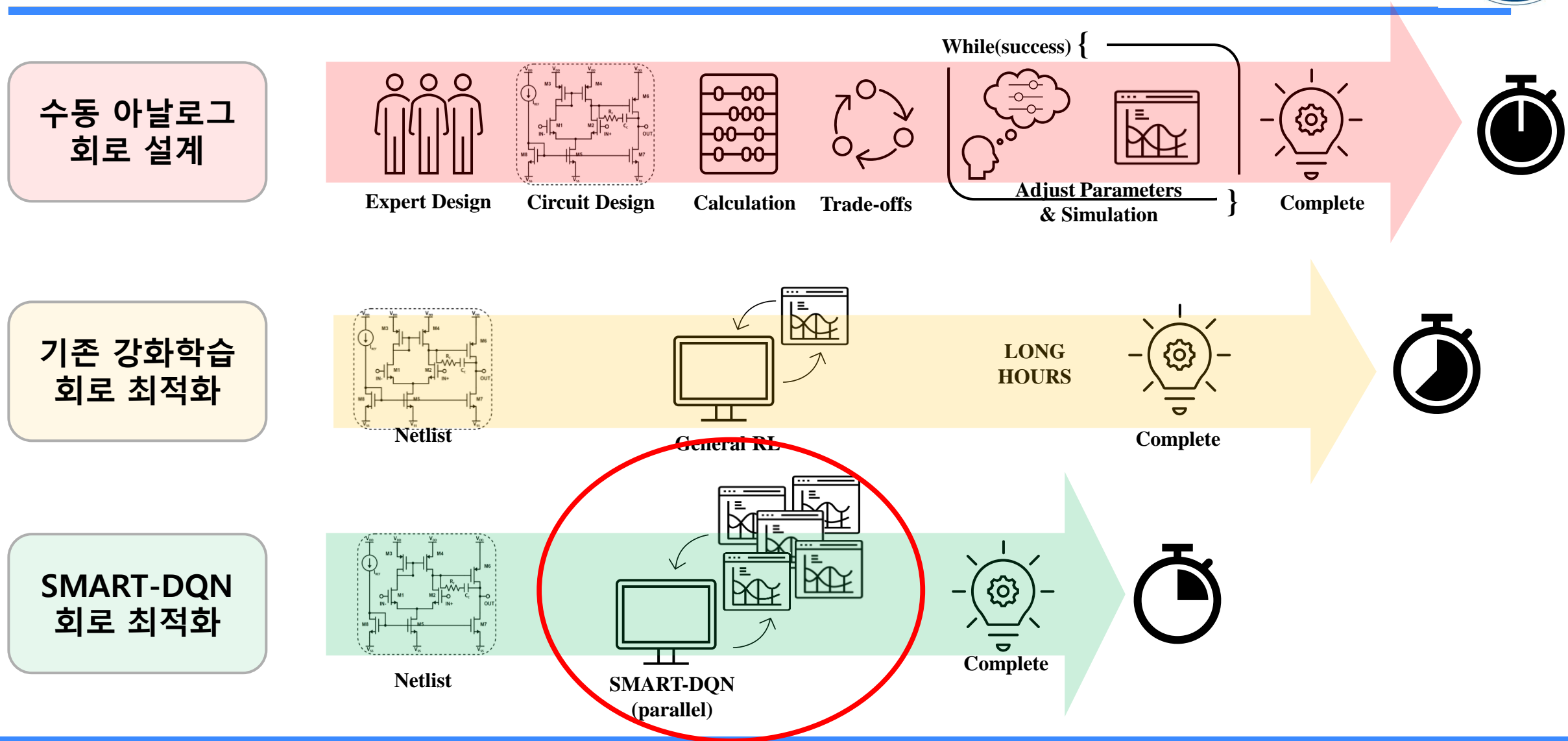


# 알고리즘

- 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

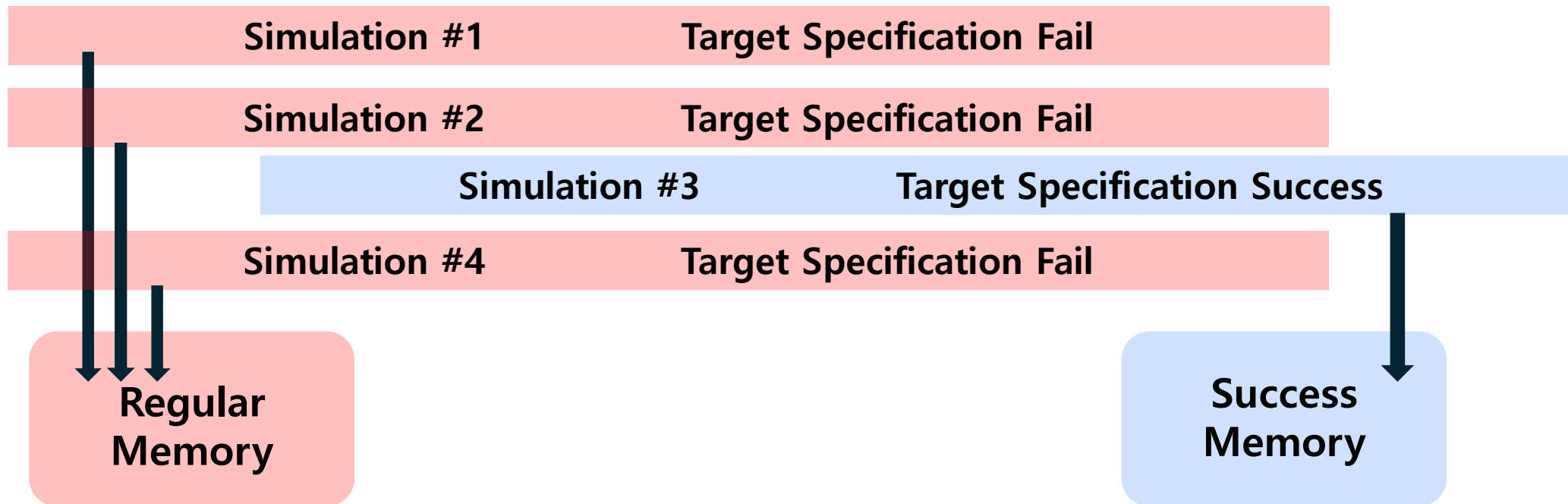


## Process

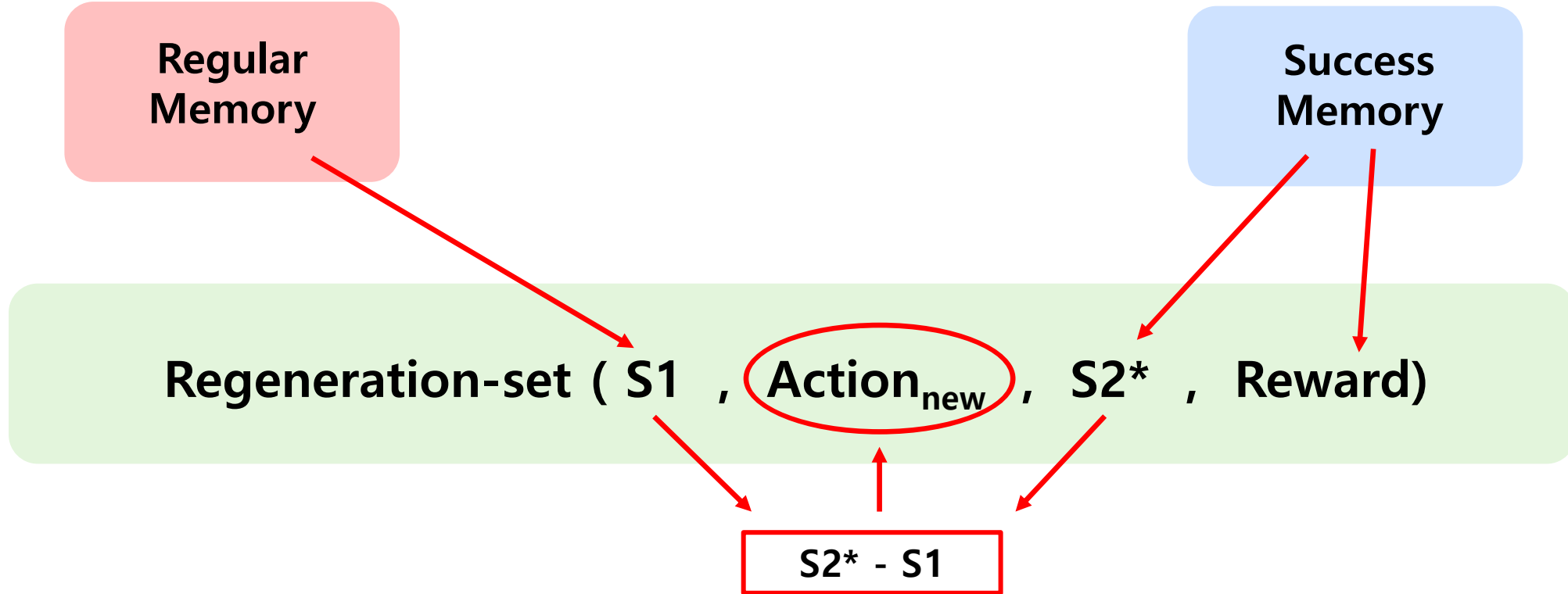


# Success Memory & Regeneration-set Training

- 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 만들

# 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*)

```

1 if  $r_t \geq r\_threshold$  then
2   | Store the experience  $(s_t, a_t, r_t, s'_t, done_t)$  in success_memory;
3 end
4 return success_memory
```

### Algorithm 2: Regeneration-Set Training

**Input:** Current state  $s_t$ , Success Memory, Regular Memory

**Output:** Regeneration Set (new experience)

```

1 if there are sufficient success memory and regular memory samples
  then
2   | Select an experience from regular memory:
     |  $(s_1, a_1, r_1, s'_1, done_1) \leftarrow \text{RandomPick}(\text{regular\_memory})$ ;
3   | Select an experience from success memory:
     |  $(s_2, a_2, r_2, s'_2, done_2) \leftarrow \text{RandomPick}(\text{success\_memory})$ ;
4   | Generate a new action  $a_{new}$  based on the state difference:
     |  $a_{new} \leftarrow s'_2 - s_1$ ;
5   | Assign a new reward  $r_{new}$  from the success experience:  $r_{new} \leftarrow r_2$ ;
6   | Form a new experience  $(s_1, a_{new}, r_{new}, s'_2, \text{False})$ ;
7   | Store the new experience for future training;
8 end
9 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 $\Omega$	4320 $\Omega$

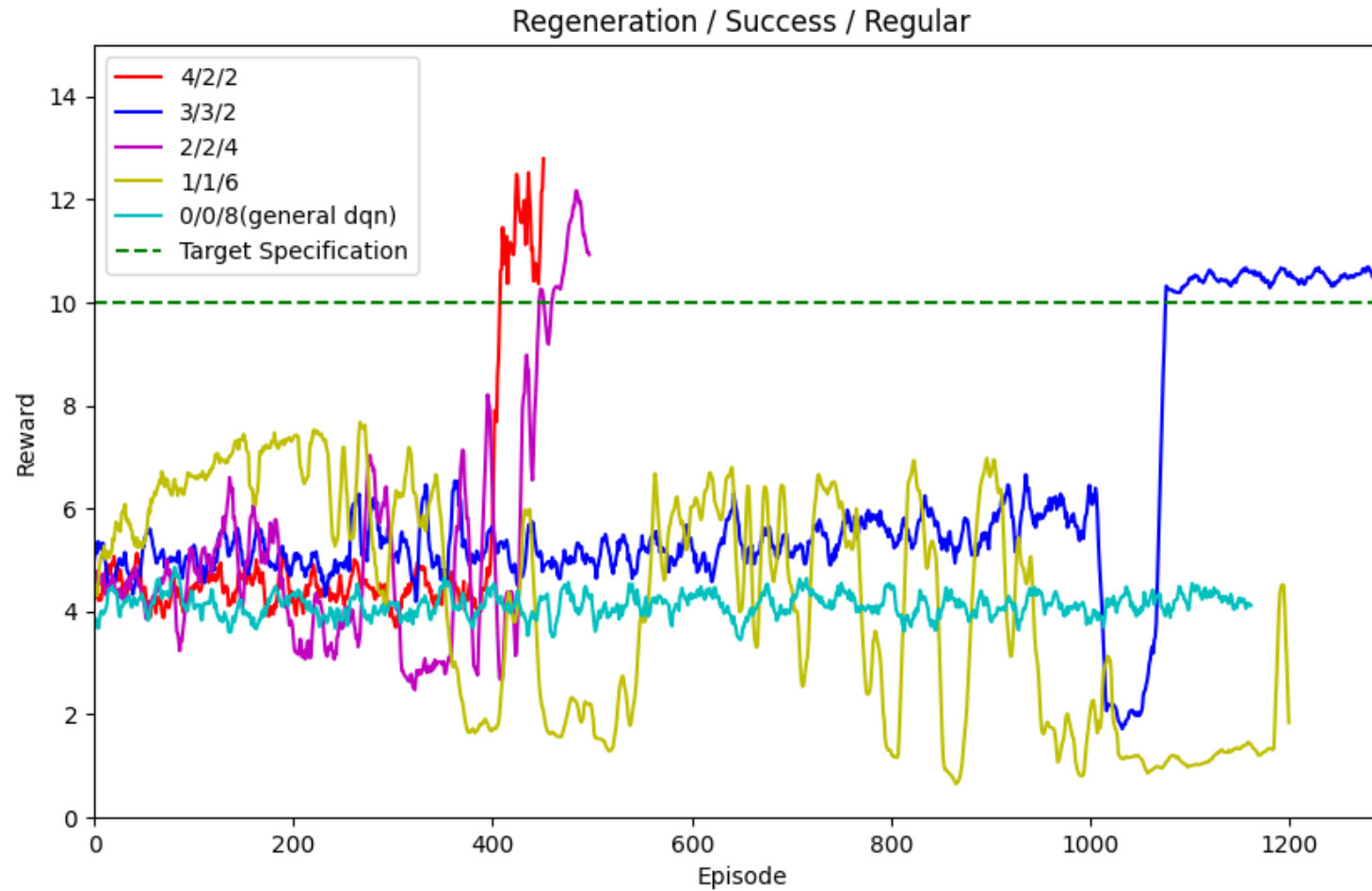
Optimized Parameter

Specification	Human Designed	Optimized
DC-Gain (dB)	82	92.5
Unity Gain Band Width (MHz)	4.6	12.8
Phase Margin ( $^{\circ}$ )	72.62	68.4
Power ( $\mu$ W)	99.46	61.25
<b>FoM</b>	<b>3.79</b>	<b>19.33</b>

Target Specification

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

# 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

## □ 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

- 기존 설계 프로세스는 설계자가 요구하는 성능에 따라 회로 설계와 회로 시뮬레이션을 반복적으로 수행하는 과정이다. 이 과정에서 설계자는 회로 설계와 회로 시뮬레이션을 반복적으로 수행하는 과정이다. 이 과정에서 설계자는 회로 설계와 회로 시뮬레이션을 반복적으로 수행하는 과정이다.

#### SMART-DQN Algorithm

SMART-DQN (Success Memory and Regeneration-set Training Deep Q-Network) is 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.

#### Conclusion

- SMART-DQN 알고리즘을 적용하여 기존 강화학습 알고리즘보다 빠르고 안정적인 학습을 통해 회로 설계와 회로 시뮬레이션을 반복적으로 수행하는 과정이다. 이 과정에서 설계자는 회로 설계와 회로 시뮬레이션을 반복적으로 수행하는 과정이다. 이 과정에서 설계자는 회로 설계와 회로 시뮬레이션을 반복적으로 수행하는 과정이다.

### 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)

요약

이 논문에서는 회로 설계를 효율적으로 최적화하기 위해 success memory와 regeneration-set training을 사용하는 강화 학습 알고리즘인 SMART-DQN(Success Memory and Regeneration-set Training Deep Q-Network)을 제안한다. 회로는 설계자는 두 단계의 전신 동작에 구동되며, 방대한 설계는 SMART-DQN이 success memory와 regeneration-set training을 통해 회로 설계와 회로 시뮬레이션을 반복적으로 수행하는 과정이다. 이 과정에서 설계자는 회로 설계와 회로 시뮬레이션을 반복적으로 수행하는 과정이다. 이 과정에서 설계자는 회로 설계와 회로 시뮬레이션을 반복적으로 수행하는 과정이다.

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. 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, SMART-DQN

### 1. 서론

다지된 회로 설계는 그 논리적 복잡성에 따라 설계 자동화(EDA) 도구가 발전하여 높은 수준의 자동화가 이루어진 현재, 하드웨어 회로 설계는 그 복잡성과 비선형성으로 인해 여전히 수작업이 많이 요구된다.

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

Sooheon Lee  
Department of Electronic Engineering  
Hanyang University  
Seoul, Republic of Korea  
sooheon22@hanyang.ac.kr

Dohyeon Kim  
Department of Electronic Engineering  
Hanyang University  
Seoul, Republic of Korea  
mdohyeon@hanyang.ac.kr

#### Abstract

In model-free reinforcement learning algorithms are divided into on-policy and off-policy types. On-policy algorithms require collecting and learning data using the same policy, which often demands large amounts of data and can lead to inefficient learning. In highly complex circuit design environments, this can also hinder convergence. In contrast, off-policy algorithms can collect data with a different policy from the one used for learning, making them more data-efficient [6]. Representative off-policy algorithms include deep Q-network (DQN) and deep deterministic policy gradient (DDPG). While DDPG has the advantage of handling continuous state spaces, it can be inefficient for exploring the entire circuit design space. On the other hand, DQN allows for efficient exploration in discrete state spaces, enabling the rapid exploration of various design points.

Still, DQN also has a few limitations such as instability during the learning process due to the random sampling of experiences and high-order dimensions [7], which requires many simulations and can slow down the learning process [8]. To address these limitations, this paper suggests success-memory and regeneration-set training, which aim to overcome the shortcomings of existing algorithms and achieve the desired design specifications through fast and stable learning. These methods improve data efficiency and learning stability, allowing the algorithm to quickly and reliably converge to the desired design specifications.

#### 1. INTRODUCTION

Due to the advancement of electronic design automation (EDA) tools, achieving a high level of automation. In contrast, analog circuit design remains a challenging task due to its complexity and nonlinearity [1]. Digital circuits utilize predefined gates and transistor-level standard cells, facilitating large-scale integration [2]. However, analog circuits must consider various performance requirements and physical characteristics, resulting in a much larger and more complex design space [3]. To overcome these challenges in analog circuit design, optimization techniques such as reinforcement learning have gained attention. Through automated exploration and optimization, reinforcement learning can quickly and precisely derive optimal solutions from complex design spaces, making it particularly effective for analog circuit design [4].

Reinforcement learning can be broadly categorized into model-based and model-free approaches. Model-based methods either learn a model of the environment's dynamics or are provided with one, which allows for prediction and planning. On the other hand, Model-free methods learn through interaction with the environment without prior knowledge and are mainly applied to complex systems. While model-based methods can be efficient when a clear model of the environment exists or can be learned, building an accurate model in complex environments is extremely challenging, limiting its applicability [5]. Therefore, it is highly nonlinear and difficult-to-model environments like

#### 2. THE OVERVIEW OF CIRCUIT OPTIMIZATION SYSTEM

Fig. 1 presents the schematic of the two-stage operational amplifier optimization using the proposed SMART-DQN. The circuit employs Miller-capacitor compensation and lead compensation techniques. The circuit is configured with  $V_{DD} = 3.3V$ ,  $I_{bias} = 10\mu A$ , and  $(W/L)_M = 10$ . A current mirror is used to provide stable current supply. A total of 16 parameters, including the widths and lengths of transistors M1 to M7, as well as  $R_2$  and  $C_{eq}$ , were adjusted to meet the target design specifications. Within the specified parameter ranges, the design objectives were set to a gain of 70 dB or higher, a unity-gain bandwidth (UGBW) of at least 1 MHz, a phase margin (PM) between 50-80 degrees, and power consumption below 150  $\mu W$ . The overall system process and

#### 3. EXPERIMENTAL RESULTS

Fig. 2 shows the comparison of the optimization results between SMART-DQN and the general DQN. SMART-DQN demonstrates superior performance in terms of convergence speed and final performance metrics. The optimization results for SMART-DQN are summarized in Table 1.

total rewards. The overall architecture and process of SMART-DQN, including the agent's interaction with the environment and the memory management, are depicted in Fig. 3.

Algorithm 2 describes the success memory algorithm. Success memory stores experiences where the circuit satisfies the target specifications, i.e., when the reward is high enough. By training on data sampled from batches that include success memory, faster and more stable learning can be achieved. This approach enables more efficient exploration and convergence, especially in circuit design environments where achieving the target specifications is challenging.

While success memory maximizes efficiency by focusing on successful experiences, it may reduce the model's ability to discover new patterns or generalize effectively due to the lack of diversity in successful experiences. If success cases are limited to specific conditions, it may reduce the diversity during training. Moreover, AI learning can be conducted in parallel, but circuit simulations must proceed sequentially. To address this, the regeneration-set training technique, proposed in Algorithm 3, was introduced. This method increases the diversity of successful experiences and generates new experiences by reusing existing ones for training. After selecting experiences from both the regular memory and success memory, new actions are generated based on the state differences between them [9], and the rewards from the success experiences are assigned to create new experiences. In this case, additional simulation data is

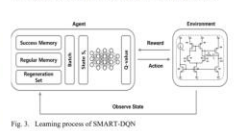


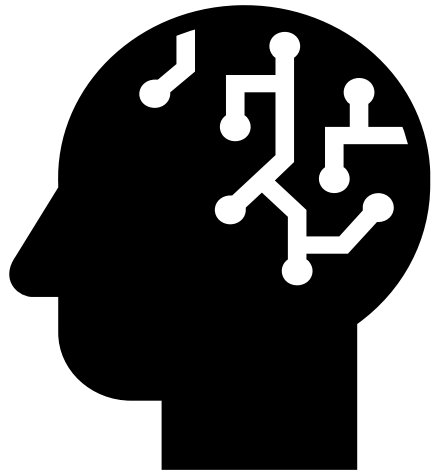
Fig. 3. Learning process of SMART-DQN

Algorithm 1: Deep Q-learning with Experience Replay

```
1 Initialize replay memory  $D$  to capacity  $M$ 
2 Initialize action-value function  $Q$  with random weights
3 for  $t = 1$  to  $T$  do
4   Initialize episode  $s_1 = \{x\}$  and preprocess sequence  $s_t = s_t(x)$ 
5   for  $t = 1$  to  $T$  do
6     With probability  $\epsilon$  select a random action  $a_t$ 
7     otherwise select  $a_t = \arg \max_a Q(s_t, a)$ 
8     Execute action  $a_t$  in the environment and observe reward  $r_t$  and image  $s_{t+1}$ 
9     Store transition  $(s_t, a_t, r_t, s_{t+1})$  in  $D$ 
10    Sample random minibatch of transitions  $(s, a, r, s')$  from  $D$ 
11    for  $i = 1$  to  $B$  do
12       $y = r + \gamma \max_{a'} Q(s', a')$  for non-terminal  $s'$ 
13       $y = r$  for terminal  $s'$ 
14       $Q(s, a) \leftarrow (1 - \alpha) Q(s, a) + \alpha y$  for non-terminal  $s'$ 
15      Perform a gradient descent step on  $\|Q(s, a) - y\|^2$  according to eqn. (3)
16    end
```



# 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



---

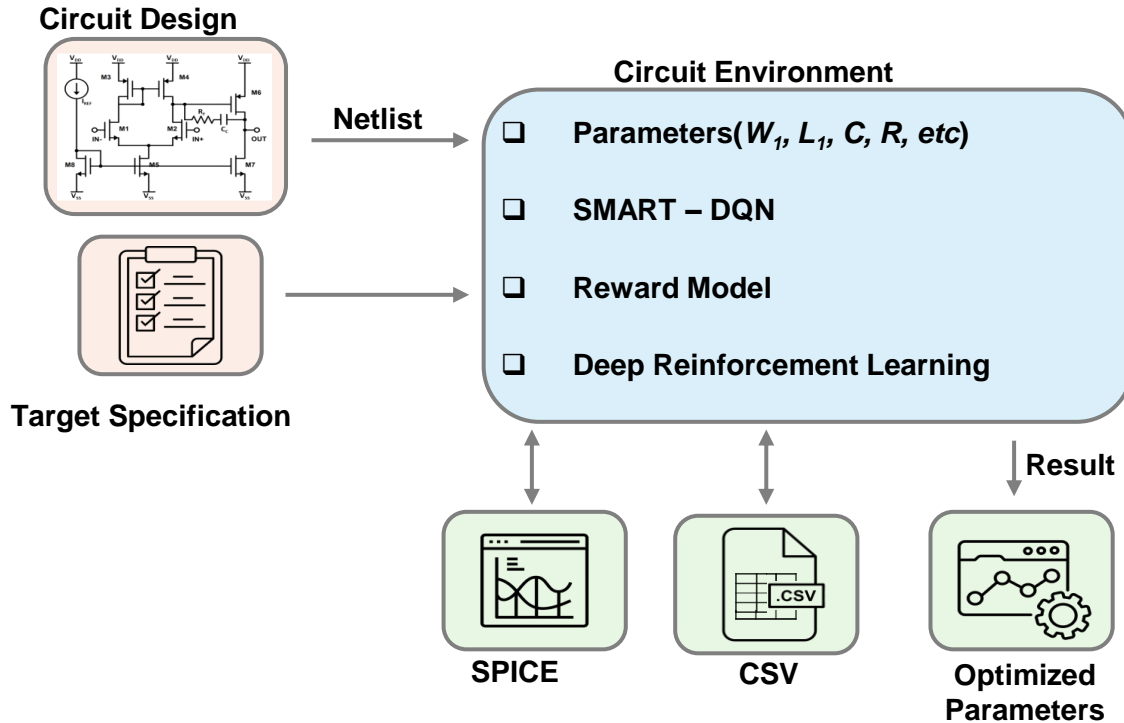
# 감사합니다

---

# Appendix

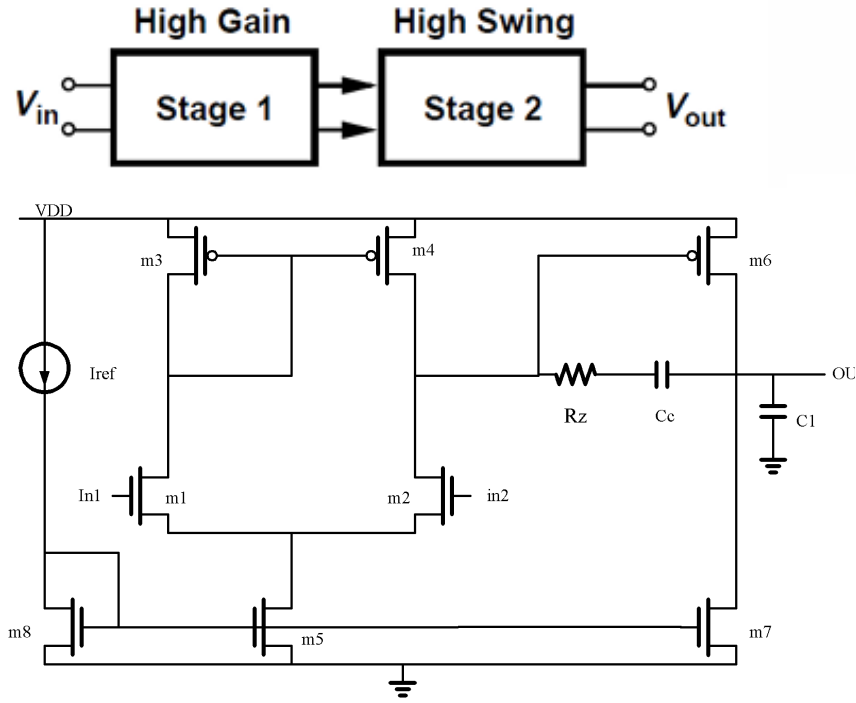
A solid blue horizontal bar that spans the entire width of the slide, positioned below the "Appendix" title.

# 회로 최적화 개요

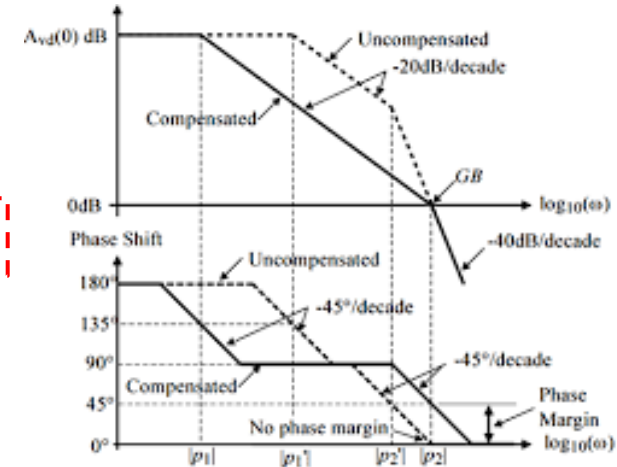
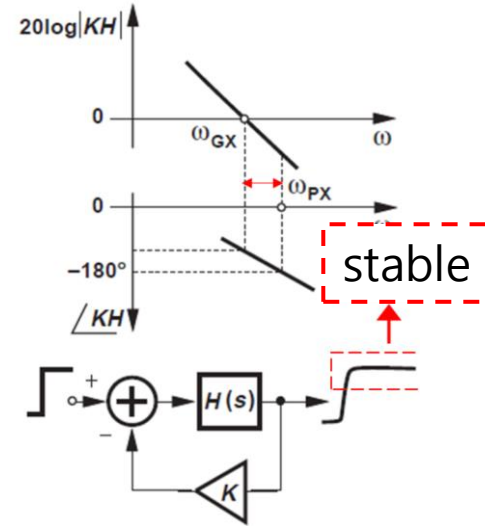
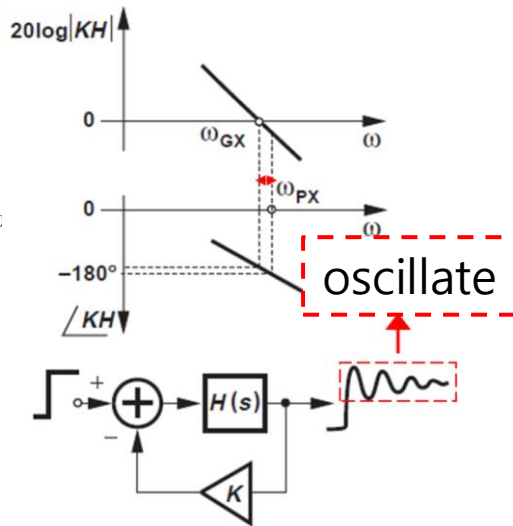


- ① Xschem을 활용하여 회로 설계
- ② 설계된 회로의 Spice파일을 python으로 연결
- ③ Target specification을 고려하여 구성된 Reward 함수를 바탕으로 회로 최적화
- ④ SMART-DQN 알고리즘으로 파라미터를 조정하며 강화학습
- ⑤ 1회의 에피소드 동안(시뮬레이션 20번) 모든 시뮬레이션이 목표에 도달하는 방향으로 학습

# 2단 연산 증폭기



## Phase Margin & Compensation method



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

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

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

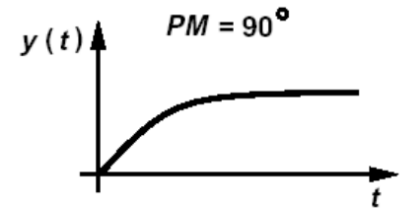
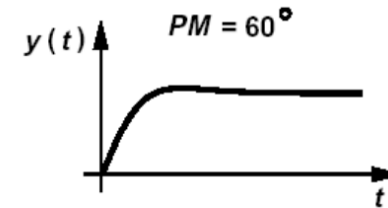
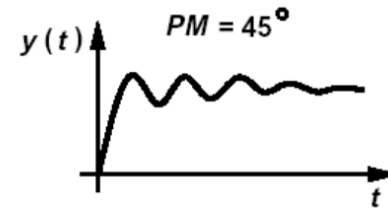
$$I_7 \propto \left(\frac{W}{L}\right)_7$$

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

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

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

$$\lambda \propto L$$



# SMART – DQN performance 비교 설명

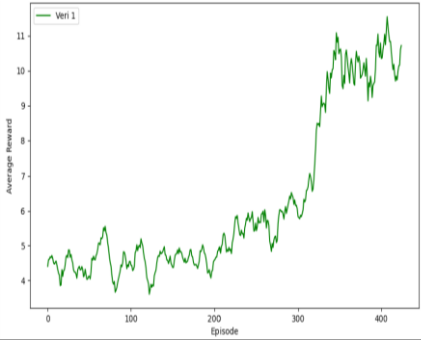
- Success memory와 Regeneration-set Training(SMART)이 절반 이상 포함되어 있을 경우 학습 **성공률이 약 10.6배 향상**하였다.
- 일반 DQN 알고리즘은 학습이 되지 않거나 수렴하지 못하며 불안정성을 보인 반면, SMART-DQN 으로 학습한 것은 **10배 이상의 속도**를 보이며 최적화된 지점으로 수렴한다.
- 한 에피소드는 20번의 시뮬레이션으로 구성되고, Target specification은 20번의 시뮬레이션에서 모든 목표 스펙을 만족한 경우를 의미한다.
- 일반 DQN의 경우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

# Hyperparameter

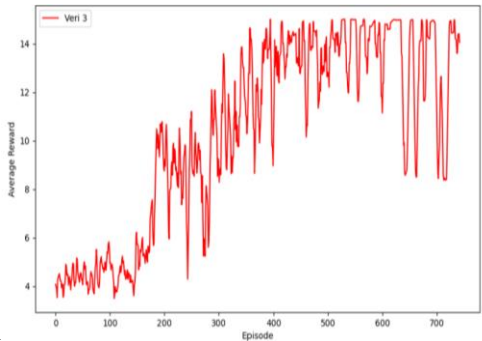
Hyperparameter	Value
Layers	3
Input Layer Neurons	16
Hidden Layer Neurons	64
Output Layer Neurons	16
Learning Rate (lr)	0.001
Learning Rate Scheduler	0.99 per 64 steps
Gamma	0.95
Tau	$1 \times 10^{-3}$
Update Period	4 steps

# 검증

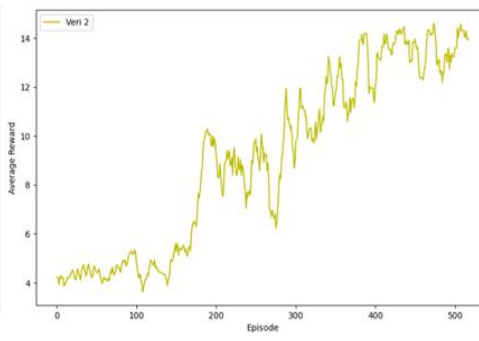
2/2/4



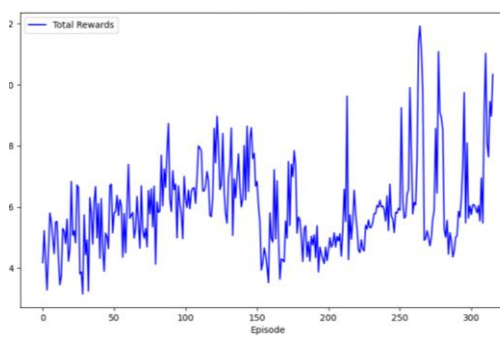
3/3/2



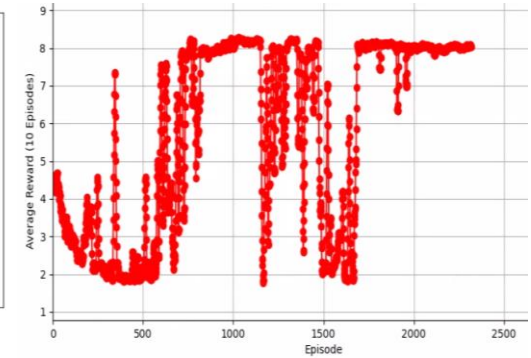
4/4/2



2/2/4



1/1/6

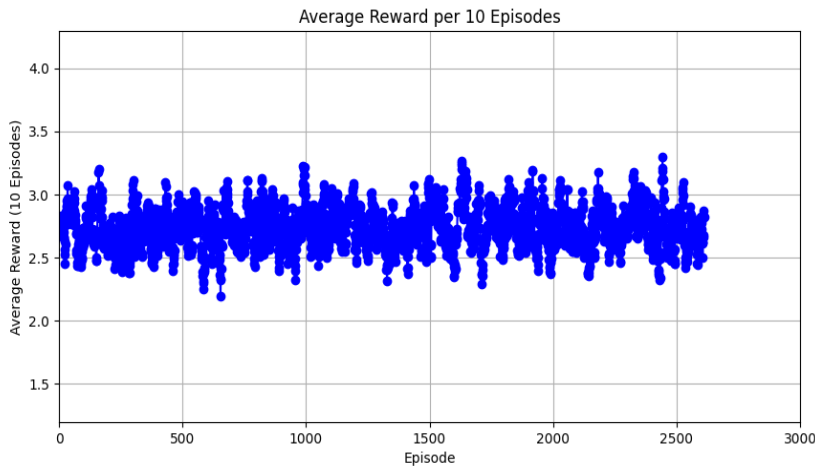


Regeneration/Success/Regular	Success/Total
4/2/2	5/7
3/3/2	5/7
2/2/4	6/7
1/1/6	1/7
0/0/8	0/7

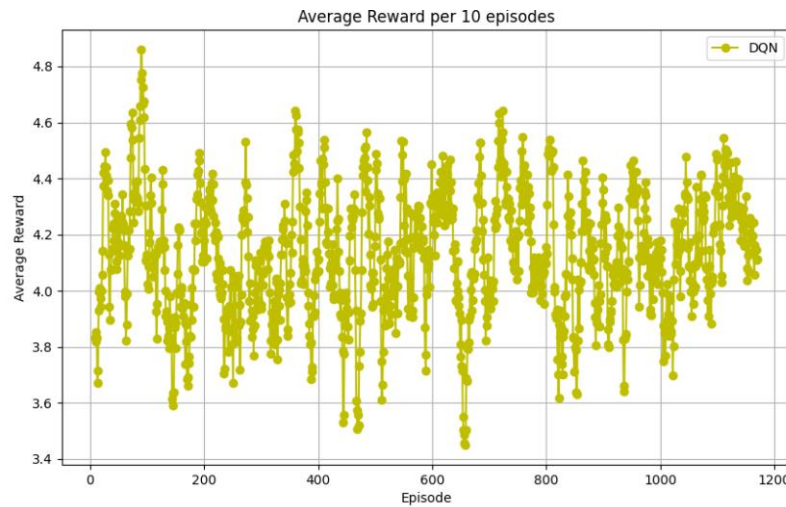
Number of successful simulation

# Reward Graph Comparison

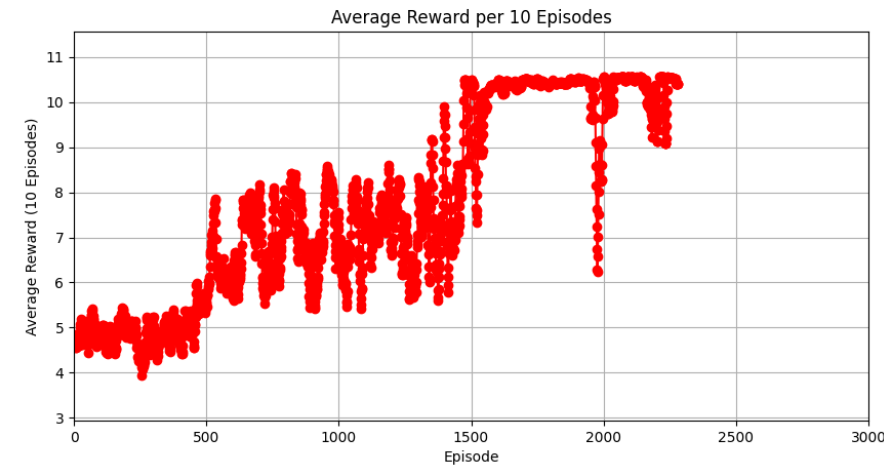
## General RL algorithm (PPO)



## 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

```
main.py X CircuitEnv.py Designer.py Reader.py Spicekernel.py Circuit.py
code > main.py
1 from utils.Circuit import Circuit
2 from utils.Spicekernel import Spicekernel
3 from utils.Reader import Reader
4 from utils.Designer import Designer
5 from utils.CircuitEnv import CircuitEnv
6 from utils.DQNAgent import DQNAgent
7 import pandas as pd
8 import numpy as np
9 import os
10 import matplotlib.pyplot as plt
11 import torch
12 from collections import deque
13 import matplotlib.pyplot as plt
14
15 # GPU 사용 여부
16 device = torch.device('cpu')
17
18 def calculate_gmw_and_pm(gmw_result):
19     frequency = gmw_result['frequency']
20     gain_db = gmw_result['dv(vout)']
21     phase = gmw_result['phase_deg']
22     gain_linear = 10 ** (gain_db / 20)
23     index_8db = np.argmax(np.abs(gain_db))
24     gmw_frequency = frequency[index_8db]
25     phase_at_gmw = phase[index_8db]
26     phase_margin = phase_at_gmw + 180
27     return gmw_frequency, phase_margin
28
29 netlist_path = "/home/paigeekins/.xscem/simulations/two_stage_opamp.spice"
30 DC_OP_result_path = "/home/paigeekins/projects/DC_OP.csv"
31 AC_OP_result_path = "/home/paigeekins/projects/AC_OP.csv"
32 GMW_OP_result_path = "/home/paigeekins/projects/GMW_OP.csv"
33
34 circuit = Circuit(netlist_path)
35 sk = Spicekernel()
36 sk.run(circuit)
```

# 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.