T.R.

GEBZE TECHNICAL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

SYSTEMATIC WAY OF SIZING THE CURRENT CONVEYOR TO MEET THE DESIRED SPECIFICATIONS

MUSTAFA ALPEREN SARI

A THESIS SUBMITTED FOR THE DEGREE OF

MASTER OF SCIENCE

DEPARTMENT OF ELECTRONIC ENGINEERING

GEBZE

2019

T.R.

GEBZE TECHNICAL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

SYSTEMATIC WAY OF SIZING THE CURRENT CONVEYOR TO MEET THE DESIRED SPECIFICATIONS

MUSTAFA ALPEREN SARI

A THESIS SUBMITTED FOR THE DEGREE OF

MASTER OF SCIENCE

DEPARTMENT OF ELECTRONIC ENGINEERING

THESIS SUPERVISOR

ASSIST. PROF. DR. ATİLLA UYGUR

GEBZE

2019

T.C.
GEBZE TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

İSTENEN ÖZELLİKLERE SAHİP AKIM TAŞIYICININ SİSTEMATİK OLARAK BOYUTLANDIRILMASI

MUSTAFA ALPEREN SARI
YÜKSEK LİSANS TEZİ
ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

DANIŞMANI
DR. ÖĞR. ÜYESİ ATİLLA UYGUR

GEBZE
2019

GEBZE TEKNİK ÜNİVERSİTESİ	YÜKSEK LİSANS JÜRİ ONAY FORMU

GTÜ Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 19/06/2019 tarih ve 2019/27 sayılı kararıyla oluşturulan jüri tarafından 01/07/2019 tarihinde tez savunma sınavı yapılan Mustafa Alperen SARI'nın tez çalışması Elektronik Mühendisliği Anabilim Dalında YÜKSEK LİSANS tezi olarak kabul edilmiştir.

JÜRİ

ÜYE (TEZ DANIŞMANI)	:Dr. Öğr. Üyesi Atilla UYGUR
ÜYE	:Dr. Öğr. Üyesi Osman Selçuk ŞENTÜRK
ÜYE	:Prof. Dr. Yaşar GÜRBÜZ

ONAY

Gebze Teknik Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun

...../...../..... tarih ve ...../..... sayılı kararı.

SUMMARY

In this thesis, the Reinforcement Learning rapid prototyping framework RELIC is introduced. After that, the modelling of the Markov Decision Process for analog integrated circuit sizing is presented. Lastly, three different Current Conveyor topologies are given to the policy gradient Reinforcement Learning agent to find the transistor widths in order to reach the proper functionality and bandwidth specifications. The Reinforcement Learning agent was able to learn how to size the Current Conveyors and it found the channel widths of transistors at a tolerable accuracy.

Key Words: Current Conveyor, Reinforcement Learning, Electronic Design Automation, Policy Gradient Agent, Analog Integrated Circuit Sizing.

ÖZET

Bu tezde Pekiştirmeli Öğrenme hızlı prototip geliştirme çatısı RELIC tanıtılmıştır. Sonrasında analog tümleşik devrelerin boyutlandırılması için Markov Karar Süreci modellemesi sunulmuştur. Son olarak, politika gradyanlı Pekiştirmeli Öğrenme ajanına üç farklı Akım Taşıyıcı topolojisi verilip gerekli fonksiyonel ve bant genişliği özelliklerini sağlayan transistör kanal genişliklerini bulma görevi verilmiştir. Pekiştirmeli Öğrenme ajanı Akım Taşıyıcıları boyutlandırmayı öğrenip kabul edilebilir hassasiyette istenen özellikleri veren transistör kanal genişliklerini bulabilmiştir.

Anahtar Kelimeler: Akım Taşıyıcı, Pekiştirmeli Öğrenme, Elektronik Dizayn Otomasyonu, Politika Gradyanı Ajanı, Analog Tümleşik Devre Boyutlandırma.

ACKNOWLEDGEMENTS

I want to thank my thesis advisor Assist. Prof. Dr. Atilla UYGUR about the guidance, insights and invaluable ideas during the thesis. Without them, this thesis would not be possible.

Lastly, I want to thank to my family for their love and support. This thesis is dedicated to my family.

TABLE of CONTENTS

LIST of ABBREVIATIONS and ACRONYMS

Abbreviations and Acronyms		**Explanations**
A	:	Action space of the MDP
a_t	:	Action a taken in time step t
α	:	Voltage gain of Current Conveyor
β	:	Current gain of Current Conveyor
C_{ox}	:	Oxide Capacitance
dB	:	Decibel
δ_t	:	Temporal Difference Error
$\mathbb{E}$	:	Expectation
γ	:	Discount Rate
G_t	:	Return following time t
I_D	:	DC drain current
L	:	MOSFET channel length
λ	:	Amount of Perturbation in nanometers
μ	:	Mobility of electrons (or holes)
P	:	State transition distribution of the MDP
π	:	Policy
π(s)	:	Deterministic policy (taken action in state s)
π(a|s)	:	Stochastic policy (probability of taking action a in state s)
π(a|s,θ)	:	Stochastic policy parameterized by θ
$Q^{\pi}(s,a)$	:	Action-Value Function under policy π
$Q^{*}(s,a)$	:	Optimal Action-Value Function
R	:	Reward function of the MDP
r_t	:	Immediate reward in time step t
η	:	Learning rate
s_0	:	Initial state of the MDP
S	:	State space of the MDP
t	:	Time step
θ	:	Policy parameters

$V^{\pi}(s)$	:	State-Value Function under policy π
$V^{*}(s)$	:	Optimal State-Value Function
V_{DS}	:	DC drain to Source Voltage
V_{GS}	:	DC gate to Source Voltage
W	:	MOSFET channel width
X	:	X terminal of the current conveyor
Y	:	Y terminal of the current conveyor
Z	:	Z terminal of the current conveyor
A3C	:	Asynchronous Advantage Actor Critic
AC	:	Alternating Current
CC	:	Current Conveyor
CCI	:	First Generation Current Conveyor
CCII	:	Second Generation Current Conveyor
CCII-	:	Negative Second Generation Current Conveyor
CCII+	:	Positive Second Generation Current Conveyor
CCIII	:	Third Generation Current Conveyor
DC	:	Direct Current
DQN	:	Deep Q Network
EDA	:	Electronic Design Automation
IC	:	Integrated Circuit
MDP	:	Markov Decision Process
ML	:	Machine Learning
MNA	:	Modified Nodal Analysis
PG	:	Policy Gradient
POMDP	:	Partially Observable Markov Decision Process
RELIC	:	REinforcement Learning for Integrated Circuit
ReLU	:	Rectified Linear Unit
RL	:	Reinforcement Learning
TRPO	:	Trust Region Policy Optimization
TD	:	Temporal Difference
VLSI	:	Very Large Scale Integration

LIST of FIGURES

LIST of TABLES

1. INTRODUCTION

1.1. Scope and Motivation

In this thesis, the sizing of the Current Conveyor (CC) task will be assigned to the Reinforcement Learning (RL) agent for meeting the desired specifications. Although the analog Integrated Circuit (IC) design flow has many layers, the circuit level will be the main objective of the design.

1.2. Thesis Organization

The thesis is organized into the six chapters. At the first chapter, the scope and motivation of the thesis and the contribution of it to the literature will be presented. Then, the corresponding literature will be mentioned in a concise manner. Later on, the created rapid prototyping framework RELIC, which is an acronym for REinforcement Learning for Integrated Circuit research, and the features of it with corresponding design flow will be presented. At the fourth and fifth chapters, the modelling of the Markov Decision Process (MDP) and the details of the design with the design choices and the training results will be discussed. Lastly, the conclusion and the possible future directions will be mentioned.

1.3. Research Objective and Contribution

The motivation of this thesis is to apply Markovian RL techniques to the circuit sizing problem which is probably the trickiest part of the design flow to meet the predefined specifications at the higher levels of abstractions. The circuits will be sized here is the current mode ones specifically the second generation current conveyors (CCII). They are versatile elements with both current and voltage terminals. It is also an interesting choice for investigating the concept of automatic sizing of an active building block and testing the performance of the proposed framework. Starting from the simple CCs, more complex structures will be sized

using a Markovian RL agent. Starting from scratch, RL agent will try to learn the environmental constraints that is defined by the prescribed requirements of the design and it will find the sizes of each transistor by interacting with the environment and observing the rewards. By doing so, the circuit sizing procedure will be fully automated. The efficacy of the automated method will be verified using a SPICE simulator. As far as we know, there is no other attempt to include the Markovian RL agent into the IC design flow in the literature. Therefore, it should be considered as an initial approach to create a fully automated RL agent that can accomplish the analog IC sizing tasks.

2. LITERATURE REVIEW

2.1. Current Conveyors

The digital technology is taking the place of the traditional analog applications. But the analog design still preserves the importance since the world is analog in nature. Moreover, the circuits are getting more complex due to Very Large Scale Integration (VLSI) of the transistors.

The CC was designed by K. C. Smith for the sake of creating a voltage controlled waveform generator that is resulting in a circuit which is known as First Generation Current Conveyor (CCI) whose control variable is a current instead of a voltage [1]. In addition to some recognized advantages of designing them almost entirely with transistors without requiring high precision passive components, it also offers one or more of the following advantages when compared to voltage mode counterparts, namely realization of the desired functionality with least number of external components without compromising to the device matching issues, simpler structure of the circuits, better accuracy, higher frequency range, larger dynamic range and better linearity [1]-[4].

The current conveyor is a three terminal device, those are namely, X, Y, and Z. Depending on the generation of the current conveyor, the functionality of the terminals are changing [5]-[7]. But the thing that does not change is the phenomena of conveying the current from one terminal to the another.

2.1.1. Second Generation Current Conveyor

Apart from the first and the third generation current conveyors (CCI and CCIII), the CCII is the most widely used one and it is shown in the Figure 2.1. The reason lies in the fact that it behaves like an ideal MOSFET [2]. The Y port is a high impedance port that draws no current and it can be used as a voltage input, like the gate terminal of a MOSFET. The X port is the current input or output port depending on the purpose and it has a low impedance level. The voltage levels of both X and Y terminals are equal as an ideal case. The third port Z is a high impedance output port

having the same current level as the X port. The direction of currents with respect to the device, determines the sign of the CCII. If both or the currents are entering towards the device, then it is considered as the positive type of CC (CCII+), whereas the inverse direction implies that the device is a negative one (CCII-). For an ideal CCII, there are two important specifications that define the proper operation of it. One is the α parameter that defines the voltage gain of the X with respect to Y terminal of the CCII. The other one is the current gain of the terminal Z with respect to the terminal X and it is called as β parameter. The equations relating the α and β parameters are as follows:

$$\alpha = \frac{v_x}{v_y} \tag{2.1}$$

$$\beta = \frac{i_z}{i_x} \tag{2.2}$$

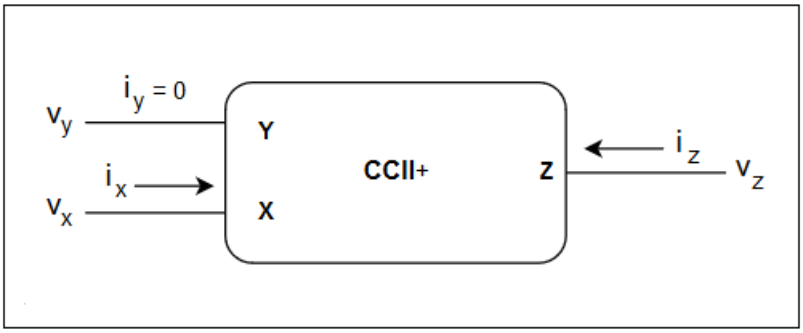

Figure 2.1: Positive Type Second Generation Current Conveyor (CCII+).

2.2. Reinforcement Learning

Apart from the electronics, the machine learning has already taken place in our life [8]-[11]. The computers are getting more intelligent. But there is a specific type of machine learning called Reinforcement Leaning (RL) that is remarkable in terms of the capabilities and ability to surpass the human level. The TD-Gammon [12] is the first known program to reach a superhuman level specifically using RL. After around the twenty years of silence in terms of RL, many of the researchers had believed that it was a unique and a special case.

Recently, a computer program AlphaGo [13] was able to defeat the 18 times world champion Lee Sedol using RL. This was in important leap-forward since the game of Go which was considered an intractable problem [14] since the dawn of artificial intelligence. The reason of this argument lies in the fact that the total number of board positions are more than the atoms in the universe for a 19x19 classic board game. In addition to the effectiveness, it has proved in board games, RL has started to enter in our lives [15]-[17].

RL is a computational approach for learning what to do [18]. In an abstract but intuitive manner, it can be summarized as: the learning entity, called an *agent*, is immersed into the training place *environment* and it is expected to change the behavior, *policy*, by making interactions. The change of behavior is guided through a feedback signal called *reward*. Thus, the agent changes its policy in order to collect more rewards: *learning*. The interaction mechanism works as follows: at certain time instants, *time step*, the agent observes the situation of the environment, *state*, and takes an *action* to change the situation of the environment. Then, the environment changes its state and at the next time step, it sends the state information, *next state*, and reward. The Figure 2.2 shows the interaction mechanism. As the time passes, the agent learns more about the environment due to interactions and starts accumulating rewards in favor of it. Apart from those, there are two optional components: *value function* which assigns a strategic value to the state or an action, and finally a *model* which defines the physical behavior of the environment. After this short intuition, the mathematical framework behind RL, which is called a Markov Decision Process (MDP), will be discussed next.

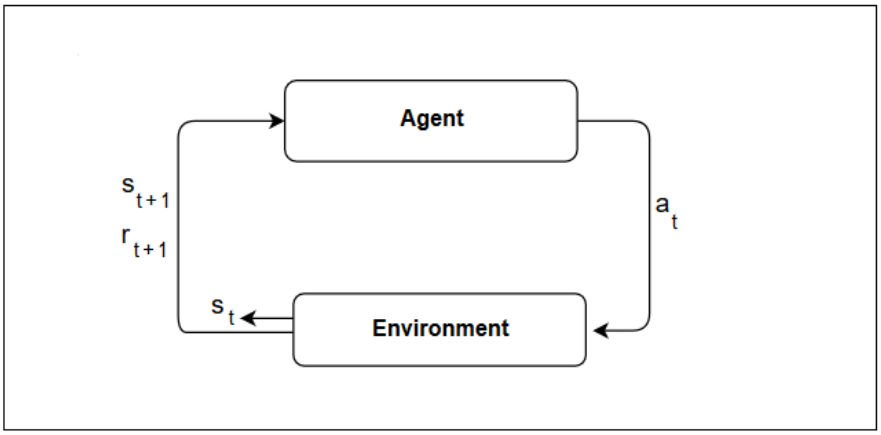

Figure 2.2: Agent and Environment Interface.

2.2.1. Markov Decision Process

A finite MDP is defined as a tuple (S, s_0, A, R, P, γ), where

- S is a finite set of states,
- s_0 is an initial state drawn from S,
- A is a finite set of actions,
- R is a reward function,
- P is a state transition distribution such that $P: S \times A \times S \rightarrow [0,1]$,
- γ is a discount rate in [0, 1].

By using these, an agent can adopt a policy that is a mapping from a state to action and represented as $\pi(a|s)$ if the environment is stochastic. Otherwise, the policy is deterministic and can be represented as $\pi(s)$. Furthermore, if the transition model of the environment is not given, or it is expensive to derive, there are *model-free* methods in the literature where an agent samples empirical data and creates heuristics based on that.

As described previously, the RL agent selects an action starting from state s_0 and transitions to next state, getting reward and repeats this process till reaching to the terminal state in S. A trajectory is the collection of state, action, reward triplets that is defined as:

$$s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_t, a_t r_{t+1} \tag{2.3}$$

During the trajectory the *return* of collected cumulative rewards can be written as:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{r+k+1} \qquad 0 \leq \gamma \leq 1 \tag{2.4}$$

Then, equation 2.4 implies that when the time horizon is infinite, the discount rate γ should be less than 1 for convergence which makes the MDP is a *continuing*

task. Otherwise it reduces to an *episodic* one. Thus, if the reward is constant and nonzero [18], it can be rewritten as a geometric series:

$$G_t = R_{t+1} + \gamma R_{t+1} + \gamma^2 R_{t+1} + \cdots = R_{t+1} + \gamma G_{t+1} = \frac{1}{1-\gamma} R \tag{2.5}$$

Thus, the expected long-term cumulative reward starting from some state s_t can be described in two different ways:

$$V^{\pi}(s) = \mathbb{E}\,[\, G_t \mid s_t \,] \tag{2.6}$$

$$Q^{\pi}(s) = \mathbb{E}\,[\, G_t \mid s_t, a_t] \tag{2.7}$$

Equation 2.6 is called *state-value function* and equation 2.7 is called an *action-value function*. The notation indicates that the value function is calculated under some policy π. Whenever the optimal policy π is found, the value functions related to the optimal behavior is defined as:

$$V^{*}(s) = \max_{\pi} V^{\pi}(s) \tag{2.8}$$

$$Q^{*}(s,a) = \max_{\pi} Q^{\pi}(s,a) \tag{2.9}$$

$$V^{*}(s) = \max_{\pi} Q^{*}(s,a) \tag{2.10}$$

The $V^{*}(s)$ is called an *optimal state-value function* and $Q^{*}(s,a)$ is an *optimal action-value function* and the relation between them is given in equation 2.10.

2.2.2. Model-Based Reinforcement Learning

In RL literature, there are various techniques proposed to find the optimal behavior under a variety of situations. Therefore, a taxonomic classification of the methods is necessary to differentiate the methods among each other. One classification can be made according to the model of the environment. These are

model-based methods if the transition model P of the environment is given or known. Referring to the equation 2.10, it can be rewritten as:

$$V^*(s) = \max_{a \in A} \mathbb{E}\,[G_t \mid s_t, a_t\,] = \mathbb{E}\,[\,R_{t+1} + G_{t+1} \mid s_t, a_t\,] \tag{2.11}$$

Referring to the definition of expectation where X is some random variable having finite possible outcomes $x_1, x_2, \dots, x_n$ with probabilities, $p_1, p_2, \dots, p_n$

$$\mathbb{E}\,[\,X\,] = \sum_{i=1}^{n} x_i p_i \tag{2.12}$$

Combining equations 2.11 and 2.12 yields:

$$V^*(s) = \max_{a \in A} \sum_{s',r} p(s', r \mid s, a)[r + \gamma V^*(s')] \tag{2.13}$$

Equation 2.13 is known as Bellman equation for optimal state-value function. By using the same techniques, it can be written for action-value function. Bellman equation gives a recursive definition of the value functions which exploits recursiveness to divide the problem into subproblems and storing the subsolution to solve bigger problems iteratively. This technique is called *dynamic programming*. Whenever a transition model P is known, and the total number of states are small, dynamic programming can be applied and this creates a foundation of the model-based methods. After finding the optimal value function, an agent can reach the goal by applying the greedy policy that selects the most reward giving actions. An example update rule that forms the basis of an algorithm called *value-iteration* is given in equation 2.14.

$$V(s) \leftarrow \max_{a \in A} \sum_{s',r} p(s', r \mid s, a)[r + \gamma V(s')] \tag{2.14}$$

2.2.3. Model-Free Reinforcement Learning

If the model of the environment is not reachable, *model-free* methods can be used which uses Monte Carlo methods or Temporal Difference (TD) approaches that they both uses sampling from the environment for collecting experience. The experience is used to approximate the value functions, whichever is being used. One of the famous one that is called Q-learning invented by Watkins [19] based on work of Sutton [20] that utilizes a concept of *TD error* δ_t:

$$\delta_t \leftarrow R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \tag{2.15}$$

Then, the update rule based on TD error is defined as:

$$Q(s,a) \leftarrow (1-\eta)Q(S,A) + \eta\,[R + \gamma \overset{max}{a}\, Q(S',a) - Q(S,A)] \tag{2.16}$$

where η is called a *learning rate* such that $0 < \eta \leq 1$.

This kind of value function driven approaches are generally referred as the *critic-only* methods, and they have generally low variance but sometimes demanding in terms of computation due to calculation of value functions especially the action space is continuous [21],[22].

Another model-free approach is directly optimizing policy rather than indirect value-function based methods. If value function cannot be estimated easily or computationally expensive, then a policy gradient-based methods can be used so that the gradient is determined by the empirically collected samples, and the parameterized policy $\pi(a|s,\theta)$ that is updated through the gradient ascent [21], [23]-[25] where θ represents the parameters of the policy (e.g.: The parameters of the neural network). The derivation is beyond the scope of this text and it can be found in related papers [21],[23]-[25]. These types of direct methods are called in the literature as *actor-only* methods since the agent is responsible only for the optimization of the policy.

There is also an *actor-critic* type of methods which combines the best of both worlds. The agent-environment interface of the actor-critic architecture is given in Figure 2.3. At each time step, the actor determines which action should be selected based on the observation and the evaluation done by the critic.

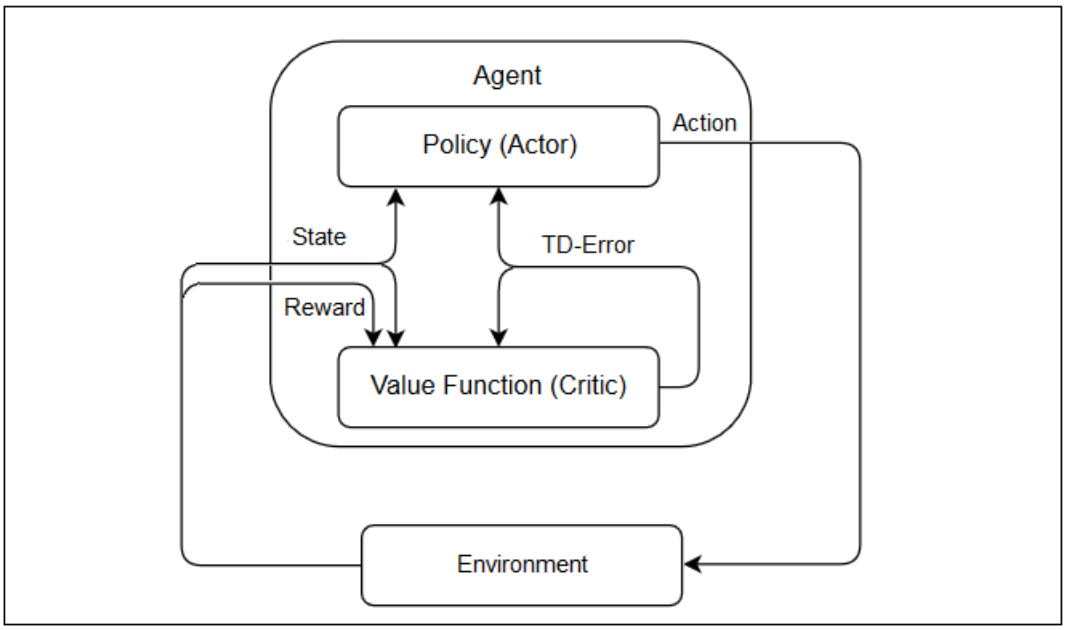

Figure 2.3: Agent - Environment Interface in Actor-Critic Architecture.

2.3. Related Work

As mentioned in the introduction part, as far as we know, there is no known Markovian RL approach to the sizing of the analog IC. But the literature contains a collection of different techniques that worth mentioning here and each of them has several advantages and disadvantages. There are evolutionary and genetic algorithms [26]-[28], bacterial foraging optimization [29], particle swarm optimization [30], [31] and other neural network based [32]-[34] approaches exist in the literature either being voltage mode or current mode. Although, the evolution based methods are seen in the category of RL, those methods are not Markovian and apply multiple static policies over a separate copy of the environment and the most reward giving policies are transferred to the next generation as the simulation of biological evolution. Unfortunately, they generally require significant amount of time if the policies are not small [18].

3. RELIC TRAINING FRAMEWORK

The RL research requires important software engineering skills in addition to machine learning skills. Additionally, significant amount of detailed work is required to combine related pieces together to make it work. Due to this entrance barrier, there is no known RL research software specifically for analog IC design.

Moreover, analog IC design is not fully automated yet. For example, sizing of the transistors to meet the desired requirements in analog domain still is an important question [35]-[37]. Apart from those, a glueing middleware software should be used for interfacing of both worlds.

These difficulties lead to creation of a fast prototyping framework called RELIC. Excluding the design and programming of it, the entire research is conducted by using that. While the research is ongoing, it is also developed step by step. The details of the functionality of the framework will be discussed next.

3.1. Architectural Overview

The RELIC framework is developed in Python programming language and the design is fully object oriented. The framework is separated into manageable packages that makes customizations easy. In addition to its simulator interface and RL problem modelling tool, it has various logging and visualization utilities which makes the training more interactive. The architectural overview of the framework is given in Figure 3.1.

The key features for the framework are: it can simulate any circuit how complex it is (SPICE simulator limitations apply, (e.g.: some simulators have limitations on maximum number of nodes), reads and parses the circuit simulator output to the memory and modify the contents in addition to modifying the circuit topology. This feature allows the training of the RL agent possible by repetitively applying the perturbations to the circuit, then performing measurements during training. Apart from that, it can simulate the circuit in a custom configuration. As an example, if the program constraints require the usage of only AC-sweep simulations many times sequentially, it can be configured to do so. Currently it is capable of

performing operating point, AC-sweep, DC-sweep and transient simulation related operations. In the near future, more modes such as transfer function and noise simulations can be added.

Last but not least, it is capable of logging the results in relational database and querying the database for create, read and update operations. Deletion is not added intentionally in order not to destroy the original content as a side effect during training. The remaining part will discuss the packages of RELIC.

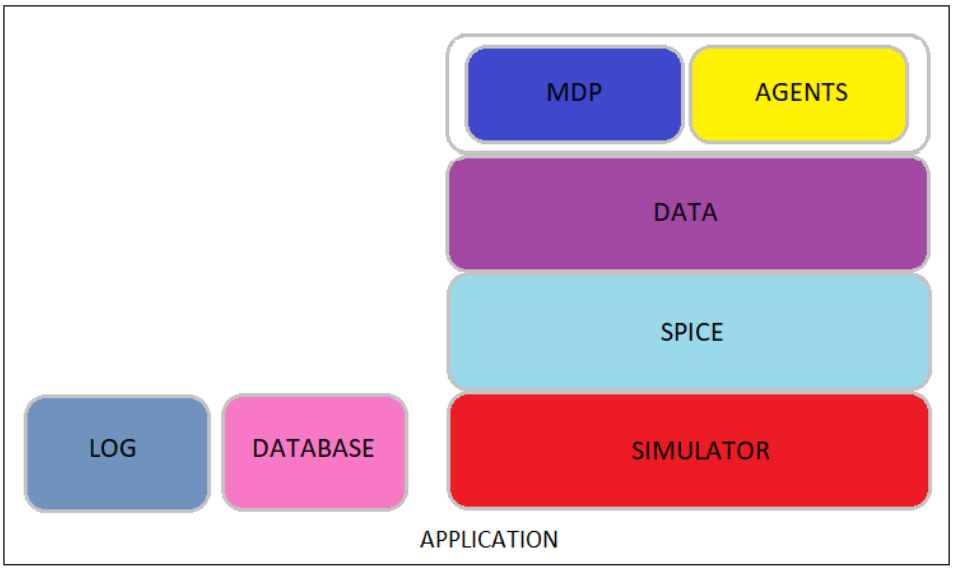

Figure 3.1: The Architecture of RELIC.

3.2. Packages

3.2.1. Agents Package

This package is the host of RL agents. Although the agent is part of the RL problem and the code in this package can be put in MDP package, it is a design choice and the reason is that there are many types of agents in RL literature. Thus, separation of the agent from MDP code as a different package, reduces coupling between the two, makes tinkering easier, and the interfacing a custom or legacy code much easier. Currently this package has policy gradient agent code. Other agents such as DQN [38], TRPO [39], A3C [40] etc. can be implemented later.

3.2.2. Data Package

This package is for SPICE simulation related files. The simulation requires a netlist or schematic file. If the circuit contains non-standard or custom model of MOSFET, then it should be located here also. In addition to them, modified netlist content (e.g.: applying the perturbations) and simulator outputs resides there. During each step of the training, the selected MOSFET's width is changed and the changed content is written to the corresponding file to be simulated again. The simulator outputs .raw extension files, that stores the node voltages and branch currents of the circuit, are here for further processing by other packages.

3.2.3. Database Package

Database related code and database configuration resides here. At the start of the training, the code automatically detects the size of the tables and forms the relational database SQL query to create them. During training, the key metrics for training is written to the database by this package periodically. It can also extract the information from the database by creating the required relational database query automatically. Another helpful feature is that it can select and compile random samples from the database by creating related SQL queries for Monte Carlo methods and preprocessing for feature scaling.

3.2.4. Logging Package

The training framework logs the training results here in addition to the database. The difference from the database package is that, in addition to training metrics it stores the training agent's state and training related information here. It also contains visualization related code of the training log.

3.2.5. MDP Package

This package stores the MDP related code. Initialization of the MDP, environment design, reward structure, and important configuration code reside here. This package is the upmost layer of abstraction from the simulator interface, that abstracts away the details of the simulator-specific low-level operations from the MDP modelling. The details of the design of the MDP is in the next chapter.

3.2.6. SPICE Package

This package is the host of simulator related code. The simulator is triggered by this package. It has certain drivers such that it can drive the simulator, read and parse all the simulation data. While performing those, it communicates with the data package and it reads the related content from there, loads them to the simulator, runs it and parses the resultant output for further processing. It acts as the middleware between the MDP environment and SPICE simulator and glues them both as a one entity with the help of data package.

3.3. Dependencies

The prerequisites of RELIC are not exhaustive, and all is open source software that anyone can reach except for the simulator. The basic requirements can be itemized as:

- Python 3.6 or above,
- Numpy 1.14.5 or above for array related operations,
- Pandas 0.24.2 or above for data manipulation,
- Python database adapter for desired database (optional),
- Matplotlib 2.2.0 or above and Seaborn 0.9.0 or above for visualization,
- Any machine learning library for realization of the agent.
- A circuit simulator.

3.4. Training with RELIC

Training with RELIC first requires a netlist file that contains the network structure of the circuit. Since writing the netlist file from scratch is difficult for complex circuits, there are many schematic capture programs in the market. Using one of them, a netlist can be created from the schematic of the circuit easily.

Using the netlist file, RELIC automatically detects the number of MOSFET transistors in the circuit that can be sized. After that, a correct operation metrics of the circuit should be defined in the MDP package. For example, the correct operation of the CCII is dependent upon the so called α and β parameters and both should be unity. After those metrics defined, additional metrics that relate performance of the circuit can be defined for experimentation. For instance, in this thesis, the bandwidth metric is defined as the additional metric. Apart from those, there are examples of how to create metrics in the package such as power consumption, gain.

After this initial effort, everything is virtually ready for the training except for the training agent selection. Since there is only one agent right now for the training, which must be selected, and training starts. The training workflow of RELIC is shown in Figure 3.2.

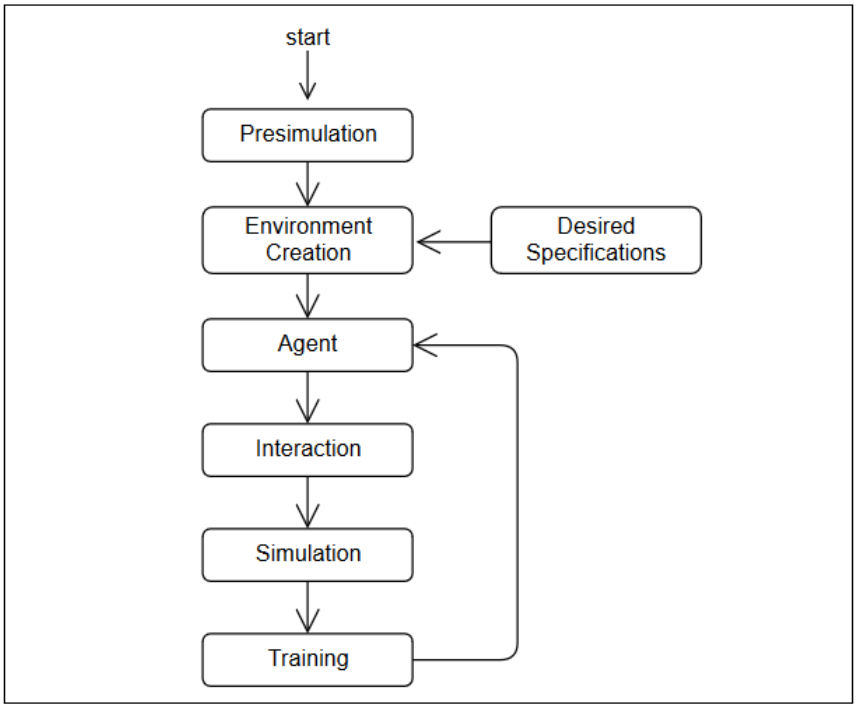

Figure 3.2: RELIC Training Workflow.

4. MODELLING THE MARKOV DECISION PROCESS

As discussed previously, RL problems should be modeled as an MDP. This section will discuss how the CCII sizing problem was modeled as an MDP. At the beginning, the overview of the design and the relevant design choices will be presented with their reasoning behind it. After that, the design itself and the important details will be mentioned.

4.1. Design Choices

Engineering problem solutions start with observation and explaining the governing phenomenon that creates the problem. At the first stage some assumptions and abstractions are made frequently. The quality of the abstraction is an important factor since it encompasses what to include in it. Some design choices related to the observations should be made at the first glance. The important observations about the analog IC design can be itemized as:

- The modern analog ICs mostly use CMOS technology which comprises MOSFETs.
- As a designer, apart from the layout customization, one can only control the width and length parameters of the MOSFET once the technology is selected beforehand.
- Analog ICs generally involve differential pairs that should be matched to suppress noise [41].
- Any modern analog IC includes many stages and the conveying of the signal is done through the usage of current mirrors.
- The topology of the circuit determines the behavior of the circuit and the same behavior can be modeled by many topologies. Therefore, there is a one-to-many relationship between the behavior and the topology. For example, an amplification behavior can be made in many ways.

In addition to the above-mentioned observations, the RL framework requires certain considerations that must be mentioned here.

- The RL problems are modeled as an MDP and it is a discrete stochastic process.
- The RL problems involve two entities: the agent and environment which should be designed separately.
- Most of the literature contains agents that require the usage of neural networks or tree like data structures. Although there are instance-based methods exist in the literature [42], it is left as an option for a later research.
- The tree like structures are nonlinear in nature but they are more intuitive.
- Neural networks can be linear, and the linearity comes with a cost of incapability of doing complex tasks (e.g. XOR problem). For accomplishing complex tasks, deep and nonlinear architectures are needed [43].
- Either being linear or nonlinear neural networks are less intuitive when they are compared to trees. One cannot observe what is going on in the network easily.
- Tree like structures are generally expensive in terms of memory and pruning [44] can be required for elimination of the overfitting.
- Neural networks are constant in size and does not grow like the tree-like data structure but those can also overfit.
- The environment should be modeled with the nonvisible electronic world whose behavior cannot easily interpreted but can be simulated.
- The environment should include simulator interface and the simulator should be selected apart. Furthermore, custom code should be written.

One should consider the capabilities of the simulator since it will be the main driving force of both the environment design of the MDP and it relates the physical accuracy to the electronic world. SPICE simulators use modified nodal analysis (MNA) [45] which outputs the branch currents and node voltages. Therefore, there is no information about the circuit topology which creates the real behavior. In addition, the MDP becomes partially observable MDP (POMDP) since the agent cannot know anything about the topology by only looking at the simulator output.

After above mentioned observations some choices should be made to abstract away the details. Since the problem becomes POMDP and the agent should observe the node voltages and branch currents (and some other information that will be discussed later), the more tractable decision is not to change the topology of the circuit during training since it creates additional concern of changing the behavior of the circuit that the agent cannot observe (i.e. an amplifier can become a filter after a slight topology change). The viable choice is to change the transistor sizes during learning. Apart from that, a neural network approach is adopted since it does not grow indefinitely. There are other design choices made which will be mentioned in the relevant places.

4.2. Mathematical Preliminaries

Before starting a design, some mathematical intuition is required. A MOSFET has three regions of operation, namely: subthreshold conduction, linear (triode), and saturation. The equations for linear and saturation region are defined as:

For linear (triode) region of operation:

$$I_D = \mu C_{ox} \frac{W}{L} (V_{GS} - V_t - \frac{V_{DS}}{2}) V_{DS} \tag{4.1}$$

For saturation region of operation (neglecting channel length modulation):

$$I_D = \mu C_{ox} \frac{W}{2L} (V_{GS} - V_t)^2 \tag{4.2}$$

For practical reasons, the subthreshold conduction region of operation is neglected since the drain current I_D is negligibly small.

4.3. Policy Gradient Agent

Policy gradient (PG) methods are first studied by Williams in his REINFORCE paper [23]. Due to its relative slowness when compared to the value function

approximation-based methods, the researchers give little importance to that for some time.

Another major research was conducted to increase its performance, generalization [21],[42] and availability even for the case of POMDPs [46]. In PG methods, the policy π is parameterized by parameters θ. The parameters of the policy are updated by the empirically calculated gradient based on the collection of the observations during training. Since it's availability to the POMDP's are studied, the research is conducted by using a PG agent with a learning rate γ of 0.915.

4.4. Environment Design

For any reinforcement learning problem, the agent and the environment should be designed separately. As far as we know, this is the first attempt to involve Markovian RL into the IC sizing problem. Since there is no proxy for goodness or the badness of the design for comparison in the literature, this leaves the problem as the trickiest part of the research. Because of this reason, this attempt should be considered as the first and a humble approach which will be a proxy for the next generation reinforcement learning agents to solve complex integrated circuit design tasks.

4.4.1. State and Action Representation

The SPICE simulators give the result for the node voltages and branch currents by using MNA. Therefore, the incoming information is carrying node voltages and branch currents and they should be included in the observation space.

Another interesting thing about the problem is analog ICs have at least one or many current mirror and differential pair like structures that should be matched for proper operation. Especially, current mirrors convey the signal between stages and those are generally have to have the same size or some scaled replica of the signal. The crucial thing that must be mentioned is if the neural network's observation space has some correlated information (i.e. linear dependence), it faces with significant performance drop [47]. Although tree like data structures, and creation of random forest [48] can handle this problem, it requires additional memory cost that we

cannot afford. Because of the limited memory capacity, this option is left for another research. Thus, the final solution of this conflicting case is left to the initialization of the environment part and the trick is in the action space design.

One other thing to mention about the case of current mirrors is that one can also consider dropping of the conveyed current components from consecutive stages to eliminate redundancy but since the problem is POMDP and this may create another problems. For example, each transistor's operation is determined by the quiescent points (operating point) that involves the drain current, gate to source voltage and drain to source voltage. Therefore, they are kept in the observation space while knowing this can create performance drop as mentioned above.

Another important consideration is the content of the data incoming from the simulator. The operation of analog ICs requires DC biasing of MOSFETs to set up the operating points for proper operation and AC information. The importance is that SPICE uses complex domain analysis (i.e. phasors) for solving AC signal related operations and outputs them in that form but uses time domain information for large signal operations (i.e. transient simulation) and the simulation output is in the time domain form. A designer should keep track of both information to design a good quality circuit. Therefore, the two entirely different scaled information and different natured content creates another problem. Feature scaling is an important phenomenon in the literature of neural networks since it speeds up learning in general. But in rare cases, it can be ignored where some of the information content is less significant than the others [47]. To demonstrate the case, looking to the RL research is important.

In RL literature, state representation is an important aspect. There are different perspectives how to represent the state. One important approach that is related to this problem is in the computer games. For instance, when designing a Pong playing agent where there are two paddles and one ball (the agent controls one of the paddles and the other paddle is controlled by a decent AI), looking at the single frame does not give any information about the ball's movement whether it is being right or left. The situation is shown in the Figure 4.1.

Therefore, to encode motion researchers [38] has designed the observation space as the difference of the consecutive frames. The exact approach is borrowed here. The reason is when designing a state for sizing problem here, one of the incoming information stores the AC response of the circuit. For instance, when

looking at a Bode plot of the circuit, if the taken action leads to better result which makes the response closer to the 0 dB for CCII case, the agent should detect this movement.

Figure 4.1: The game of Pong.

Unlike the case of Pong, the problem here is the information content is not homogeneous. The term homogeneity lies in the fact that, the voltage like content of the circuit is generally in volts or millivolts range, but the currents are generally in microamperes or even smaller range. Researchers took the difference of frames where both the frames are in similar range, that effectively cancels out the nonmoving similar content (e.g.: background pixels) but taking the difference of important information content to take the residue as the proxy of movement as an observation. It is also known that; neural networks can learn most when a surprising information comes which makes this situation is an example [47]. For example, there are more specialized neural networks such as convolutional neural networks that can detect the edges and corners exactly in the same manner as how visual cortex does [49]. Therefore, another design choice is done here by incorporating the consecutive frames approach with a slight difference. Adding some state related *reinforcer* [18], [19], information to the observation space in addition to taking the difference of two simulation outputs. In addition to the mathematical background, the reasoning here is something philosophical: letting the agent choose what to include and amplify by looking at some reinforcers.

The other interesting part is the design of the action space before exploring the reinforcer signals. To perturb the circuit, a MOSFET or possibly bunch of MOSFET's (can be all of them) should be selected as the candidate for perturbation. The choice being made here is selecting the MOSFET one by one since selection of several MOSFET's and perturbing all at the same time may disturb the data representation indefinitely (e.g.: Nonstationarity). Furthermore, the RL agents act sequentially and this behavior fits to this choice. Thus, the action space is designed as three sets of actions for each MOSFET.

Every time step t, only one MOSFET can be selected and the selected action can be one of the three:

- Decrease the width by λ where λ is the amount of perturbation in nanometers,
- Increase the width by λ,
- Copy the MOSFET width at this time step without perturbing.

The last action is applied in two steps. At time step t, a candidate MOSFET is selected as the copy source and at the next time step t+1, the candidate MOSFET's width is copied to the another MOSFET that will be selected. Whenever the copy action selected, the other actions are overridden at time step t+1. To preserve state information, the reinforcer signals should be itemized here as:

- Time step t,
- Selected action at time step t,
- Copy action stack status (useful for tracking copy action),
- Agent hit the boundary or not,
- The last hit action to the boundary,
- The instantaneous widths of the MOSFETs.

The first reinforcer action is added for both tracking the time step of the episode by the agent and reducing the problem to an episodic task whose length is selected to be 50 steps long which is a reasonable length for moderate level of conventional analog ICs. The second and third actions are useful together and they are for the observation of which action selected by the agent and it is loaded to the action stack for copying. Whenever a MOSFET is selected for copy operation, the

action signal stores which MOSFET will be copied at next time step, and stack status is for tracking that it is a copy action or a normal action. The stack stores only the last action. The details of the last two actions is related to the reward structure and will be explained in that section. All these reinforcer signals are appended at each time step to the difference of the other observations.

4.4.2. Reward Structure

One other important piece that worth mentioning about any RL problem is the structure of the reward and it is one other trickiest part of the problem. The reward of the environment should not be correlated with the observations for the learning in a real manner [18]. But the simulation information is encoded in the state and all the information that governing node voltages and branch currents are in there. Neural networks with more than two layers can perform almost any nonlinear calculations [50]. At the same time, the reward should also represent the desired behavior without any correlation. Although an approach called reward shaping [51] that gives guidance to the agent to speed up learning is an option, the presented solution is rather different but flexible since the policy can be broken by the change of the reward function (e.g.: policy invariance).

The solution is creation of region that represents the desired behavior. If the taken action results in a situation lying in that region, then it is favored. Otherwise, it should be punished by giving negative reward. Furthermore, due to the flexibility of this approach, any arbitrary region can be defined as the reward giving. The CCII behavior will be ideal if α and β parameters are close to unity and the bandwidth is enough for the desired application. Therefore, an example region that represents the desired behavior is shown in the Figure 4.2. The lower bound determines the bandwidth and precision of the desired behavior while upper bound is useful for bounding the result from the upper part. Then the pseudocode for the reward is given in Table 4.1.

Table 4.1: Pseudocode for the Reward.

Pseudocode for the reward
region ← desiredregion *aValue ← percentageOfAlphaInDesiredRegion* *bValue ← percentageOfBetaInDesiredRegion* *if aValue + bValue ≥ 0.9 then* *reward ←1* *else* *reward ← - 1*

Apart from that, if an agent pushes the same boundary (i.e. maximum or minimum of the allowed width) two times, an episode ends with a highly negative reward (punishment) of -100. The reason of adding this constraint is when an agent pushes the same upper or lower boundary many times, and if somehow that action gives positive reward, the agent cannot learn anything and stucks at local optima since its impetus is to collect more positive rewards. To eliminate that behavior, this type of reward added intentionally.

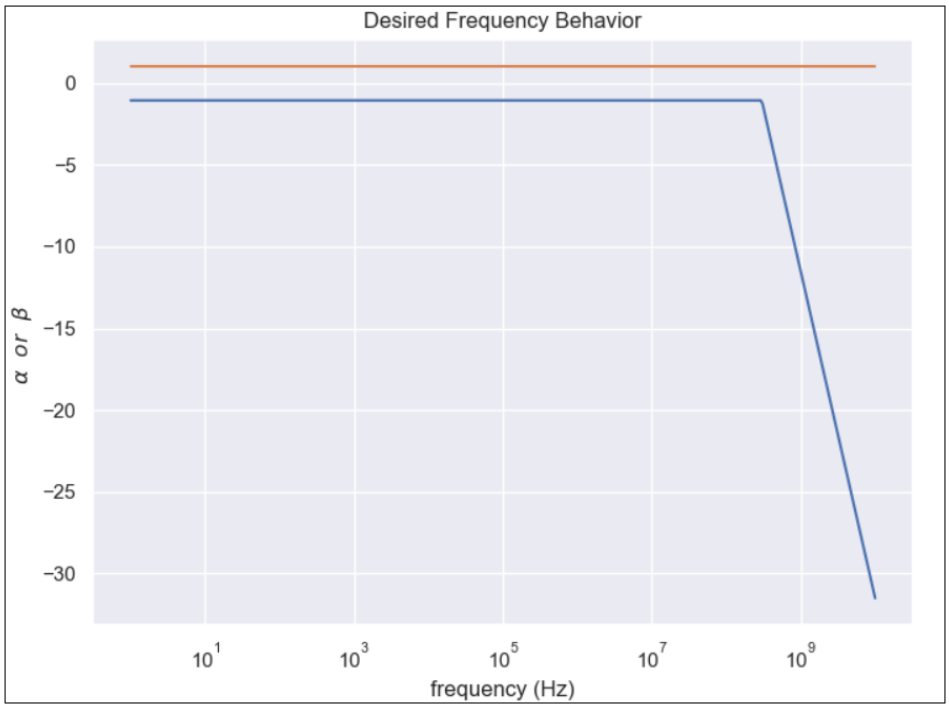

Figure 4.2: An Example Region That Represents the Desired Behavior.

4.4.3. Neural Network and Initialization

Another piece of information about the design is the architecture of the neural network. It has three layers such that the first hidden layer consists of 1024 neurons and the second one contains 128 neurons both having an activation of ReLU [52],[53]. At the final layer, 3n number of neurons are used where n represents the number of MOSFETs in the circuit with softmax activation.

The final piece of information is the initialization of the widths of the MOSFETs. Since the widths cannot be smaller than the minimum allowed one determined by the process parameters, but can be bigger than that without any bound, the upper bound should be determined first and it is selected as 4 micrometers to limit probability space and computation time which is a reasonable choice for modern processes. Therefore, the initial widths are sampled from a uniform distribution of 10000 samples generated randomly within the bounds.

4.5. Experiment Design and Circuits

The experimentation is done with three different circuits. One of them is the 8 transistor CCII+ that is shown in the Figure 4.3 and it is intentionally selected as a wide bandwidth topology that is derived from current mirrors [54]. The reason of the selection is to inspect the behavior of the RL agent when it is faced with an idealized behavior with an easier goal. To make the problem a little more complicated, two different regions are defined as the reward giving. One region is selected in between +0.25 dB and -0.25 dB. The second region is defined in such a way that the cutoff frequency is 300MHz starting linearly from -0.25 dB and decaying at -20dB / decade after passing cutoff frequency. It is also bounded up by +0.25 dB linearly as shown in Figure 4.2.

The second and third circuits [55] have a narrower bandwidth but high precision. Those are also CCII+ topologies but they are more complex such that they have 13 and 15 transistors respectively. The intention is to emulate the behavior of the agent under conflicting design choices and it will be given an assignment of designing a higher bandwidth circuit with the narrow bandwidth topology. The topologies are shown in Figure 4.4 and Figure 4.5. The desired region is also stricter

because of the precision of the topology. The region for α is designed to be in -0.05 dB with a cutoff frequency of 300 MHz again and upper bounded by +0.05 dB. The region for β is kept between -0.08 dB and +0.08 dB which is a reasonable amount of precision.

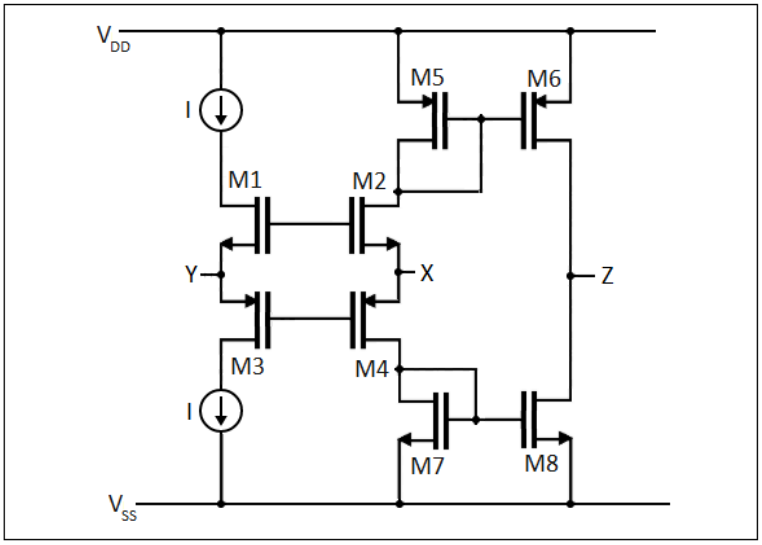

Figure 4.3: 8 Transistor CCII+.

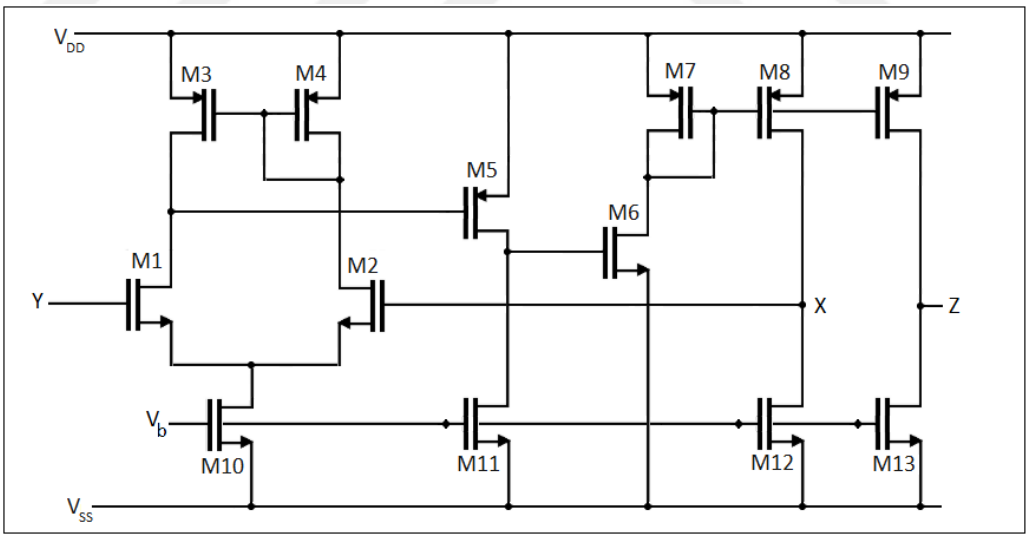

Figure 4.4: 13 Transistor CCII+.

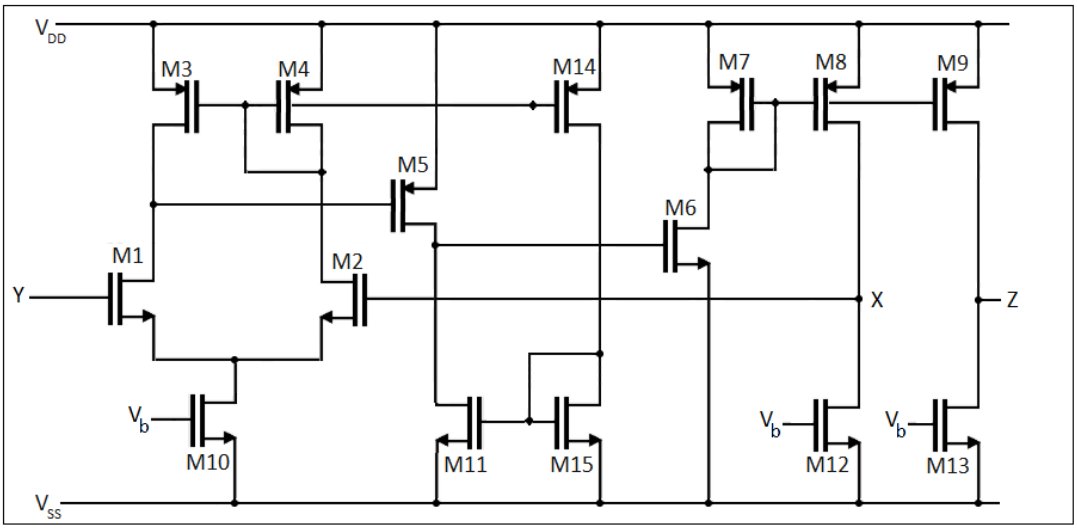

Figure 4.5: 15 Transistor CCII+.

4.6. Methodology and Metrics

Before starting the training, some important considerations should be mentioned first. The reason is that this is the first known attempt and a humble approach yet to make the sizing procedure automated in an end-to-end manner by utilizing the RL framework. Definition of some performance metrics, important considerations and methodology will be a good start.

4.6.1. Exploration and Exploitation

The agent should observe the environment enough to make wise enough decisions about the widths of the MOSFETs. However, it should have sufficient knowledge about the environment to make wise enough choices. The implication of sufficient knowledge is the experience that was tried and found to be reward giving at the past by experiences. As the name implies, the experience has a cost to bear the consequences, and in return the more exploration gives higher chance to explore more reward giving situations. In the RL literature, this phenomenon is known as the *exploration and exploitation dilemma* and it defines the optimality of the behavior of the agent which is responsible for creating highly efficient strategies to be followed.

Watkins [19] mentioned that the optimality of the learning occurs if the allocated capacity for maximally efficient performance is valuable and it cheaply explores the environment and the time taken to learn the behavior is shorter than the time period during which the behavior will be used. Therefore, it should be noted that the last part of the optimality argument signifies that the learning of the efficient strategy is more important than the time devoted to learning it. Thus, to measure the efficacy of the agent, the change of the normalized widths of all MOSFETs over the time devoted to the learning it will be measured as one metric. The normalization is useful since the sizes of transistors can be in different scales. This metric has also a dual effect of measuring the exploration of the agent about the surroundings. Furthermore, it can guide the research through how effective each one of the MOSFETs in terms of the desired performance. In addition to that, the change of reward over time will be the other indicator for the quality of the learning as expected. One last comment that deserves mentioning about is the stationarity concept which will be discussed next.

4.6.2. Stationarity

Stationarity can be defined as the invariancy of the statistical properties such as mean, variance, and autocorrelation over time. For any machine learning task, this property of the data plays an important role for the solution of the agent. In addition to the stationarity of the environment, the stationarity of the solution is more important for this case. Since the sizing problem here offers a stationary simulation environment that does not change over time (e.g. independent and identical distribution of data), the point that will be discussed here is the stationarity of the solution. After all, the main reason for teaching how to size the CCII to the agent, is to find widths that gives the desired behavior of the circuit.

The key metrics for tracking stationarity of the solution is two-fold. One is the tracking of the cumulative statistical properties of the widths of each transistor during entire training. This will give the measure of how overall stationarity of the solution changes over time. The other one is the measurement of same statistics for the last few episodes that will give more exact result about the final widths that should be applied to reach the desired behavior for CCII. In order to achieve these

goals, the statistical properties will be measured for each 250 episodes of training. The final widths are also collected from the last 250 episodes of data.

4.6.3. Preprocessing and Training Hardware

In order to speed up training, some preprocessing of the simulation data is crucial. The AC sweep simulation has real and imaginary components and transient simulation data is also high dimensional. For correctly detecting the cutoff frequency of the circuit in AC sweep simulation, a reasonably good precision can be achieved by increasing the number of points sampled in a decade. Furthermore, a CCII can have a wide bandwidth which makes the situation harder. For example, the AC sweep simulation creates 391 points for each of the node voltage or branch current if the simulation is setup for 39 points per decade. For a reasonably complex circuit, there are at least 40 unknowns. For instance, A MOSFET is a four terminal device which creates four unknowns just for real part of the currents. When combined with imaginary parts and transient simulation data, the dimensionality drastically increases. Therefore, to reduce the dimensionality, some preprocessing is required inevitably.

As an initial step, the invariant components of the circuit are eliminated. For example, DC sources either being current or voltage are not changing with respect to time and they are dropped. Furthermore, the AC sweep and transient simulations have components that they are not changing if the circuit is perturbed. For instance the ‘frequency’ parameter from the AC sweep simulation and ‘time’ parameter from the transient simulation is dropped. Although, the elimination of invariants with respect to perturbations is good for reducing the dimensionality, some other techniques are required.

As a more effective way of reducing the dimensionality, the down-sampling is applied where applicable without disturbing the consistency of simulation data. As a result, AC sweep simulations are down-sampled by 3, and transient simulation data is down-sampled by 20 in such a way that a neural network can easily detect. To demonstrate the situation, the original graph of the transient simulation data and the down-sampled data is shown in the Figure 4.6.

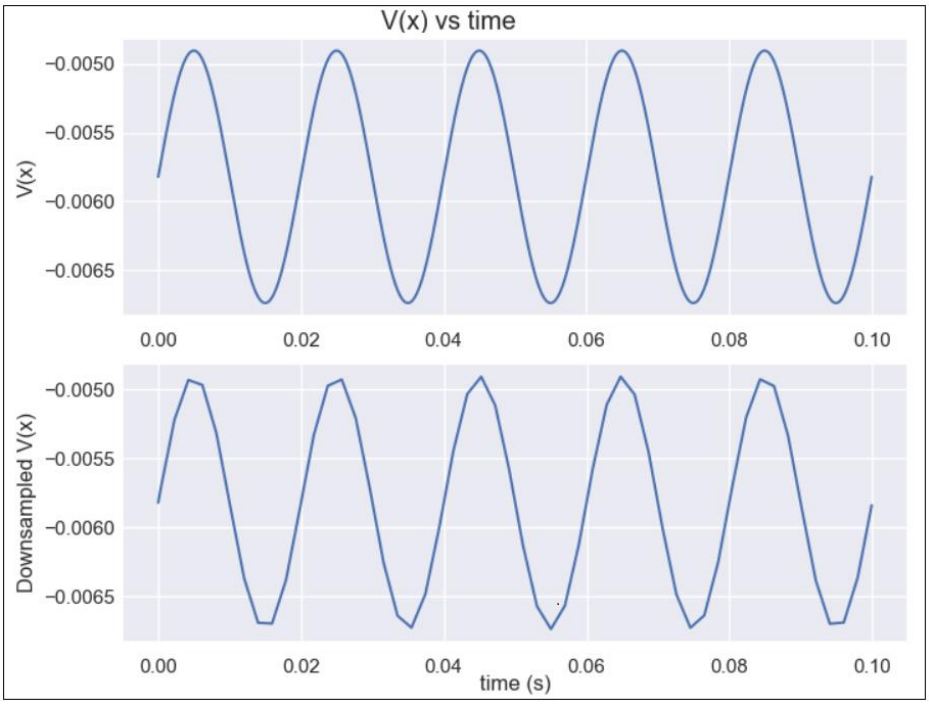

Figure 4.6: Initial Transient Simulation Data and Down-Sampled Data.

Finally, the research is conducted through a moderate level notebook having a processor of Intel ® Core ™ i5 7200U CPU having 2.5 GHz base frequency with 8 GB of RAM operating at 2133 MHz. The Table 4.2 shows the approximate training time in hours for each of the circuit.

Table 4.2: Training Times of CCII Circuits.

Circuit	**Approximate Training Time (Hours)**
8 Transistor CCII+	~40
13 Transistor CCII+	~63
15 Transistor CCII+	~50

5. RESULTS

In this section, all of the training results for each of the circuits, important statistics, and metrics in addition to the implications of them will be presented. Transistor sizes are determined using RL agent. Currently, transistor sizes are not aimed for actual chip production so that the widths are given in the figures without considering layout effects, process variations, temperature effects, supply variations, etc. due to the study is being of the preliminary results of an ongoing research. At the last part, conclusion and possible directions as the future work will be presented.

5.1. Training Results

5.1.1. 8 Transistor CCII

The training result of the 8 transistor CCII topology is provided in Figure 5.1 that represents how the reward is distributed over time. As it can be seen from the figure, the agent can learn how to size the transistor since the average of the collected reward crosses over the boundary of 0. Since the data is noisy, it is smoothed by averaging the data over a sliding window size of 1000. The change of widths over time is also provided in Figure 5.2.

Based on the training data, some observations should be mentioned. Due to random initialization of the widths, the agent had decided to reduce the width of transistor M1. This behavior is interesting since the DC drain currents of the transistors M1 and M3 are not changing due to ideal current sources, whereas the changing thing with perturbation is the terminal voltages. It can be verified that from the transistor's current equations 4.1 and 4.2.

The agent had decided to decrease the width of M1. It slightly increased the width of M3 and kept them in a tighter margin than the other transistors after crossing over the boundary around 40000 steps.

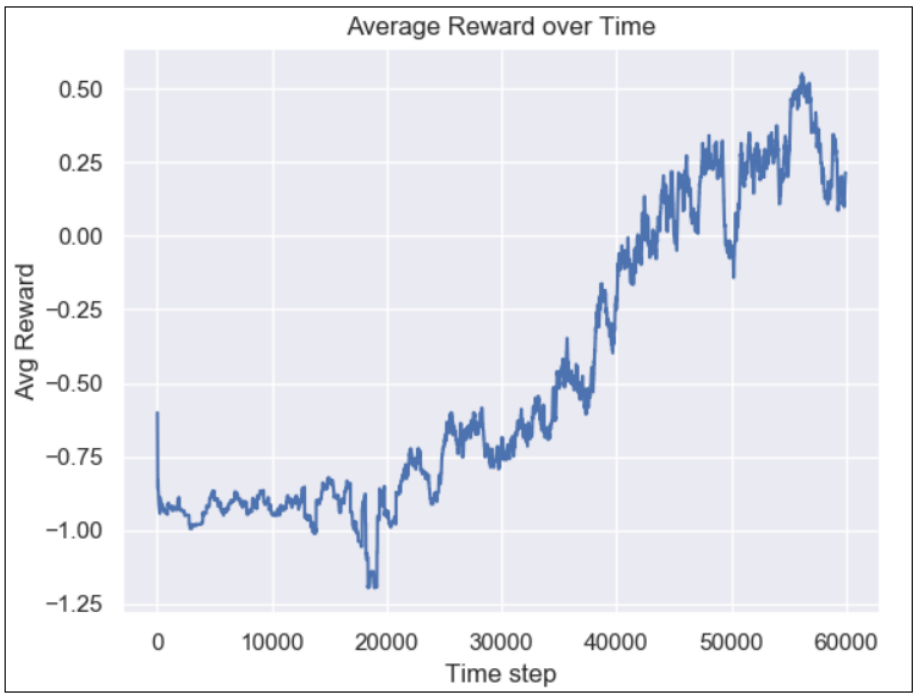

Figure 5.1: 8 Transistor Average Reward over Time.

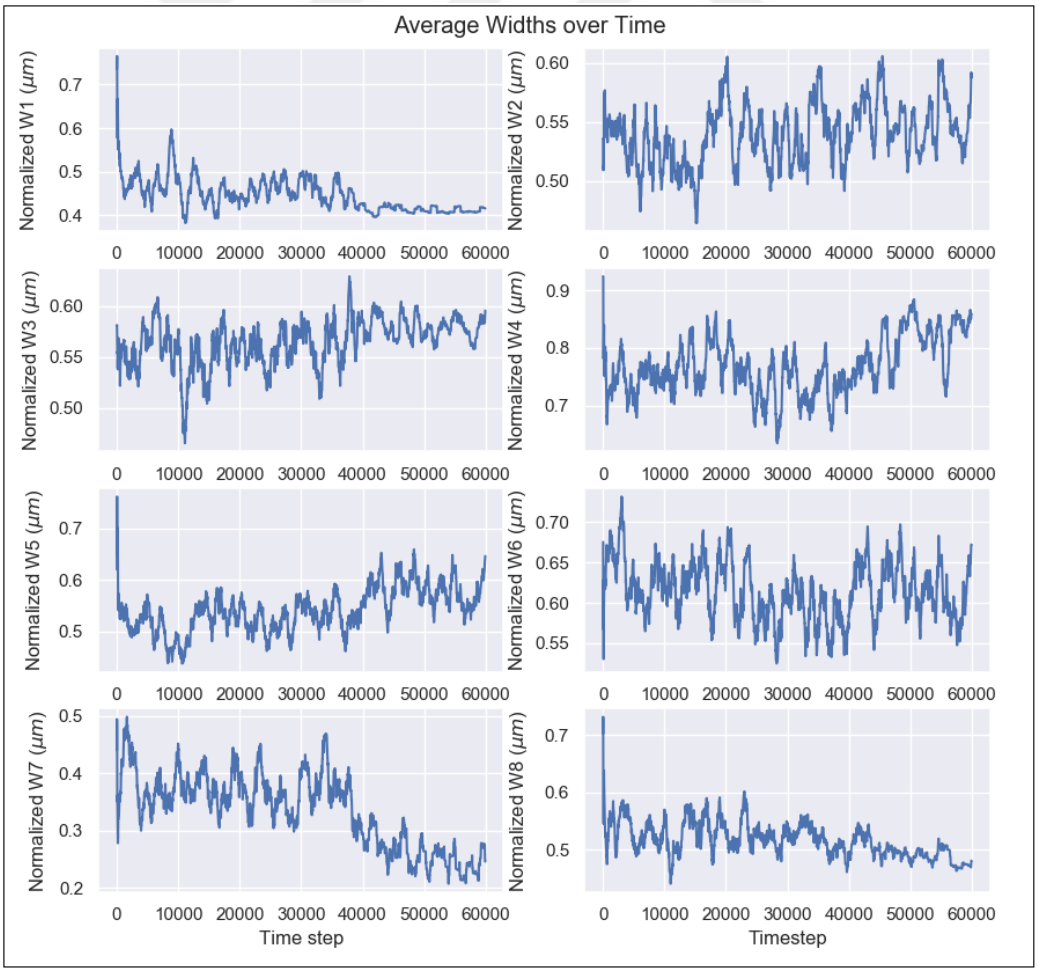

Figure 5.2: Normalized Widths of 8 Transistor CCII+.

One approach to explain this behavior is related to the exploration-exploitation dilemma that was mentioned previously. The agent probably detected that the efficacy of the perturbations applied is not worth exploring more by keeping them in a tighter margin. The perturbations do not change current but terminal voltages and hence after correctly selecting the width, it had decided not to change much. This behavior can be investigated more as the future work.

Apart from those, the agent had systematically shifted the width of M5 upwards while shifting the transistor M7 downwards after crossing the boundary. One important thing to mention here is that the agent was able to learn the importance of matching of the transistors, without even knowing what a transistor is in a partially observable environment, since it had kept the widths of both M5 and M6 transistors within the same normalized range of 0.55 and 0.65 micrometers only after passing the boundary.

Finally, it is also important to mention that the region of operation concept of the transistor is not provided to the agent. The same argument cannot be said for the matching of the transistors M7 and M8 since the agent had decreased the width of transistor M7 around the normalized widths of 0.25 micrometers but trying the normalized widths of 0.5 micrometers for M8. However, since the matching of PMOS transistors are done, it can be tolerated for now and further investigated in the future. Finally, the normalized and actual widths are tabulated in Table 5.1.

Table 5.1: 8 Transistor CCII+ Sizes.

	W1	W2	W3	W4	W5	W6	W7	W8
Normal Widths	0.41	0.55	0.58	0.83	0.58	0.62	0.25	0.49
Actual Widths	1.75	2.28	2.39	3.35	2.41	2.54	1.12	2.04

5.1.2. 13 Transistor CCII

As before, the training result of the 13 transistor CCII topology is provided in Figure 5.3 that shows how the reward is distributed over time. The trend shows that the agent can learn how to size the circuit but this time, not only the learning takes more time as expected but also it is noisier than 8 transistor circuit. It is also

smoothed by averaging the data over a sliding window of 1000. The normalized and actual widths found by the agent are also tabulated in Table 5.2.

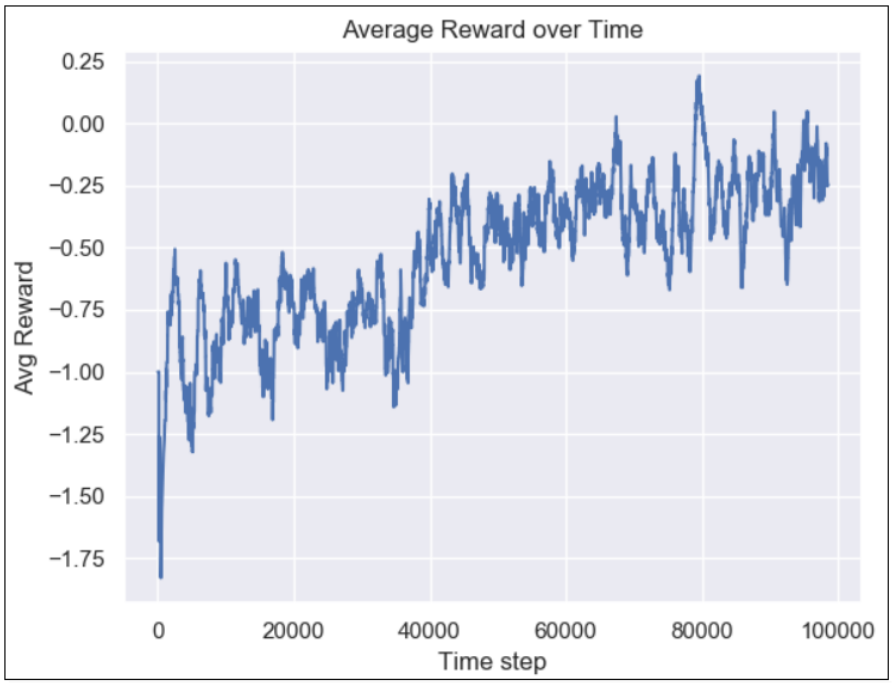

Figure 5.3: 13 Transistor Average Reward over Time.

Table 5.2: 13 Transistor CCII+ Sizes.

	W1	W2	W3	W4	W5	W6	W7	W8
Normal Widths	0.46	0.57	0.61	0.88	0.50	0.74	0.27	0.55
Actual Widths	1.94	2.36	2.51	3.54	2.09	3.01	1.21	2.28
	W9	**W10**	**W11**	**W12**	**W13**			
Normal Widths	0.25	0.66	0.35	0.60	0.73			
Actual Widths	1.14	2.70	1.52	2.47	2.97			

The change of widths is provided in Figure 5.4. The training data shows that the agent can learn a policy to size the CCII, but it takes longer time. Furthermore, the crossing over the boundary is not smooth this time that there is a slight oscillation about the policy after 70000 steps of training until 90000 steps. Then, the policy improves slightly. To give more intuition about the training, the scrutiny of the graphs is required.

This time, the input stage contains a differential pair and both M1 and M2 should be matched as known from analog circuit design. The widths of M1 lies in the

range of 0.4 to 0.55 micrometers and M2 is in the range of 0.5 to 0.65. Although, there is a slight difference, it can be concluded that the agent can at least adapt to the situation.

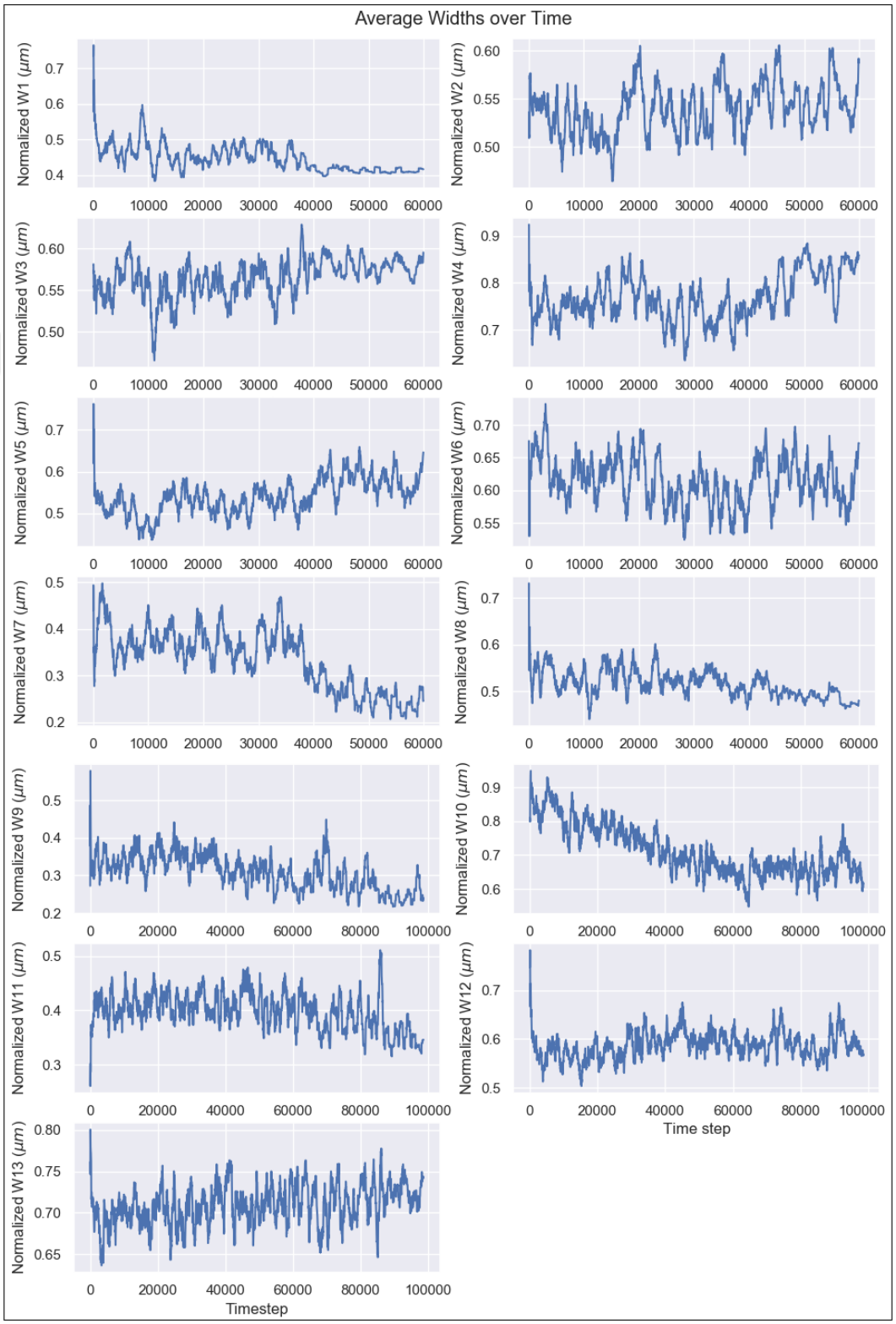

Figure 5.4: Normalized Widths of 13 Transistor CCII+.

Apart from differential pairs, the agent tends to match the transistors M7 and M9 as it can be seen after 90000 steps. The more interesting case is the obvious decrease and stabilization of width after 60000 steps of M10 which is responsible for controlling the operation of the differential pairs at the input.

5.1.3. 15 Transistor CCII

The distribution of reward over time from the training of the 15 transistor CCII+ topology is provided in Figure 5.5. Unlike from the 13 transistor case, it is more evident that the agent can learn how to size the CCII after around 45000 steps. The change of widths is also provided in Figure 5.6. It is also smoothed again by averaging the data over a sliding window of 1000.

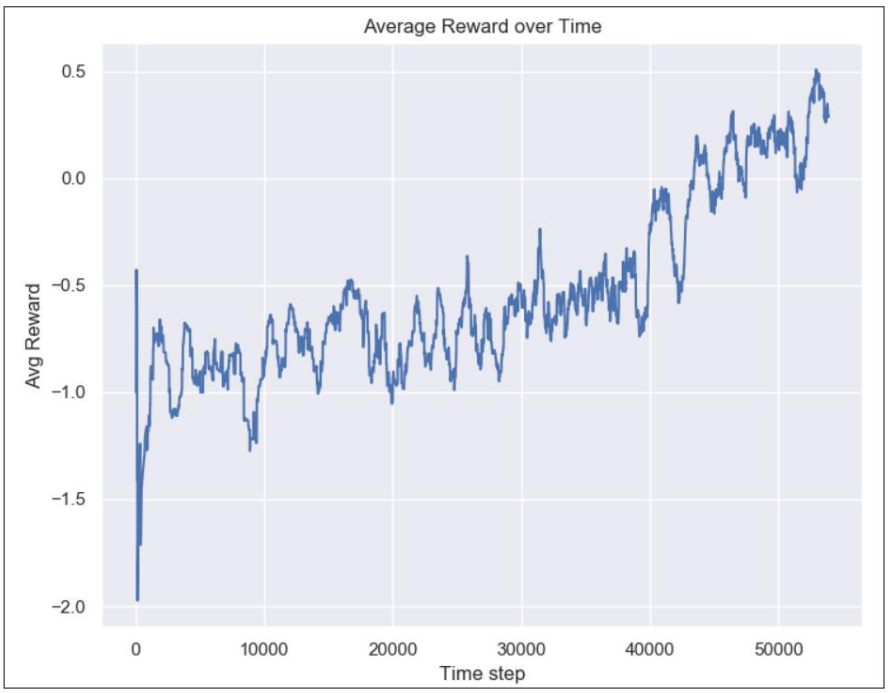

Figure 5.5: 15 Transistor Average Reward over Time.

The input stage also contains a differential pair as in the case of 13 transistor topology. Although, the expected behavior was the matching of the transistors, the agent found a solution in the range of 0.4 to 0.42 micrometers for M1, and in the range of 0.54 to 0.56 micrometers for M2. Interestingly, the difference of the both are almost the same and the reason can be investigated further in the future.

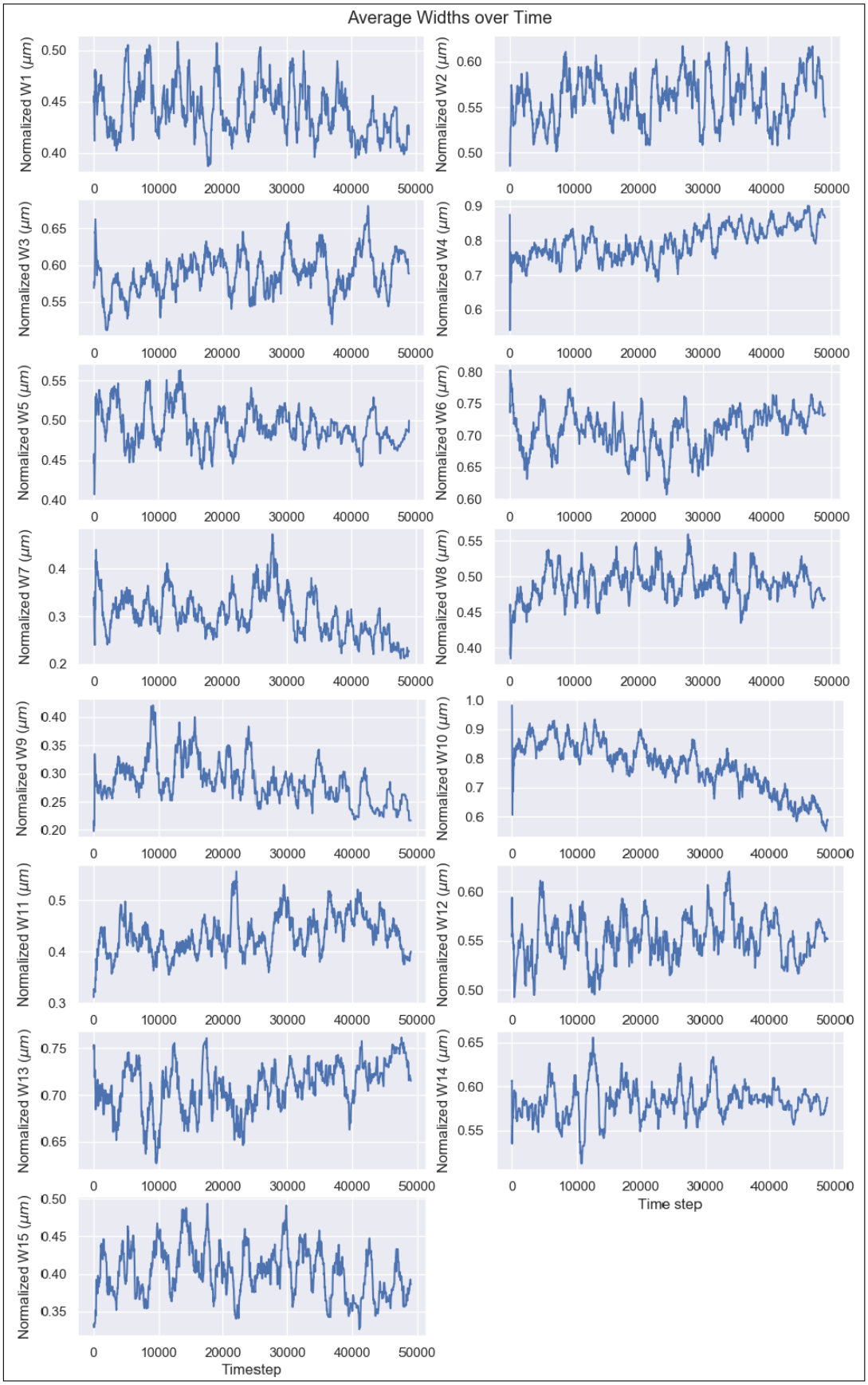

Figure 5.6: Normalized Widths of 15 Transistor CCII+.

.

Another interesting thing is, the agent is also reducing the transistor M10 which controls the current flowing through the differential pair in addition to reducing M9 and M11. Finally, the widths found by the agent are given in Table 5.3.

Table 5.3: 15 Transistor CCII+ Sizes.

	W1	W2	W3	W4	W5	W6	W7	W8
Normal Widths	0.41	0.57	0.60	0.87	0.48	0.74	0.25	0.48
Actual Widths	1.75	2.36	2.47	3.50	2.01	3.01	1.14	2.01
	W9	**W10**	**W11**	**W12**	**W13**	**W14**	**W15**	
Normal Widths	0.24	0.60	0.41	0.55	0.73	0.58	0.37	
Actual Widths	1.10	2.47	1.75	2.28	2.97	2.40	1.59	

5.2. Stationarity

As described in the methodology section, tracking the cumulative stationarity of the MOSFETs is useful for the stationarity of the solution. Hence, the stationarity information of each topology is provided as a heatmap in the Figures 5.7, 5.8, 5.9. As it can be seen from the figures, the agent changes the transistor sizes smoothly.

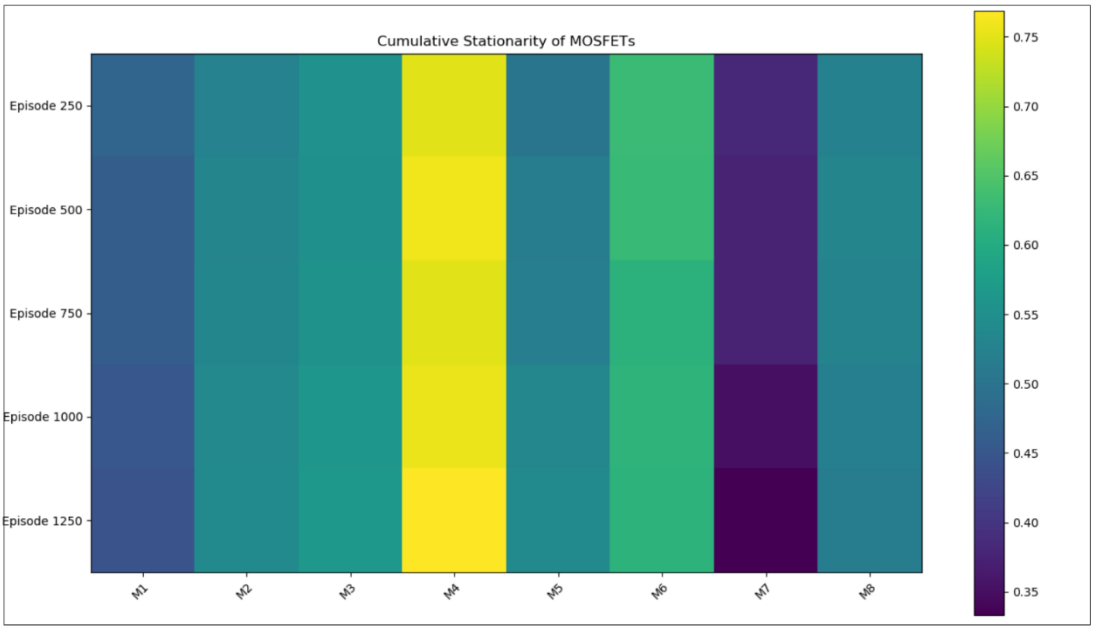

Figure 5.7: Cumulative Stationarity of 8 Transistor CCII+.

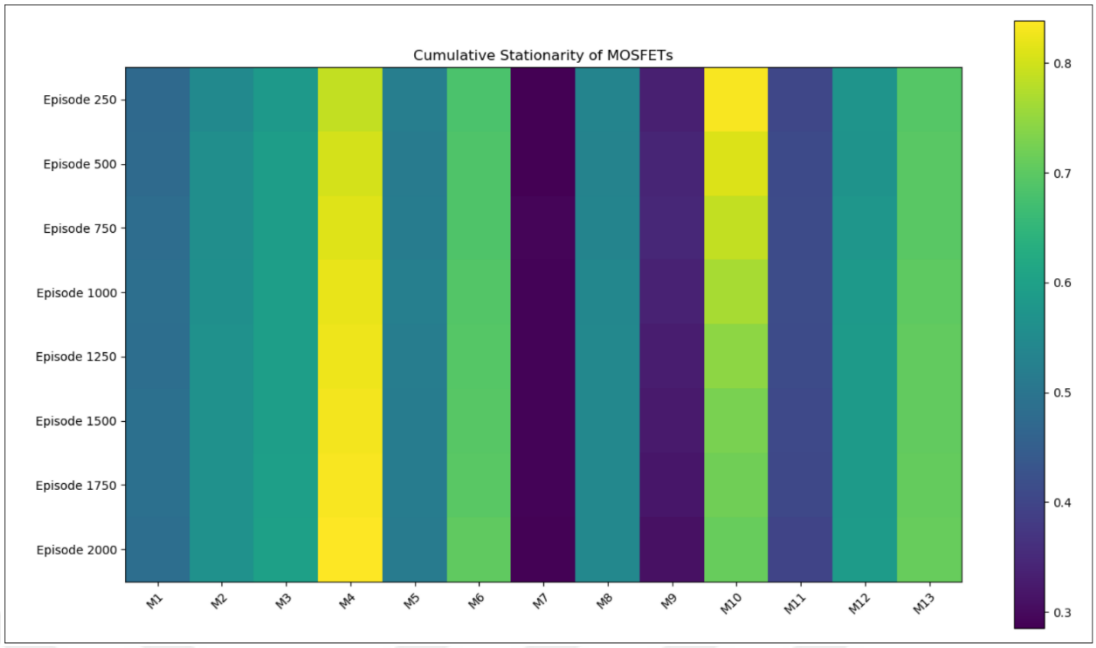

Figure 5.8: Cumulative Stationarity of 13 Transistor CCII+.

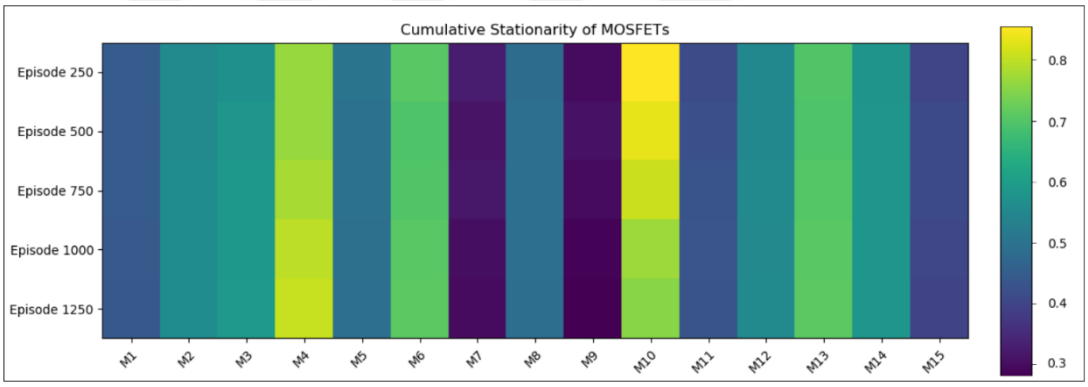

Figure 5.9: Cumulative Stationarity of 15 Transistor CCII+.

In order to cross validate the solutions found by the RL agent, the given widths from the agent are manually simulated by using SPICE and the results are provided in Figures 5.10, 5.11, and 5.12.

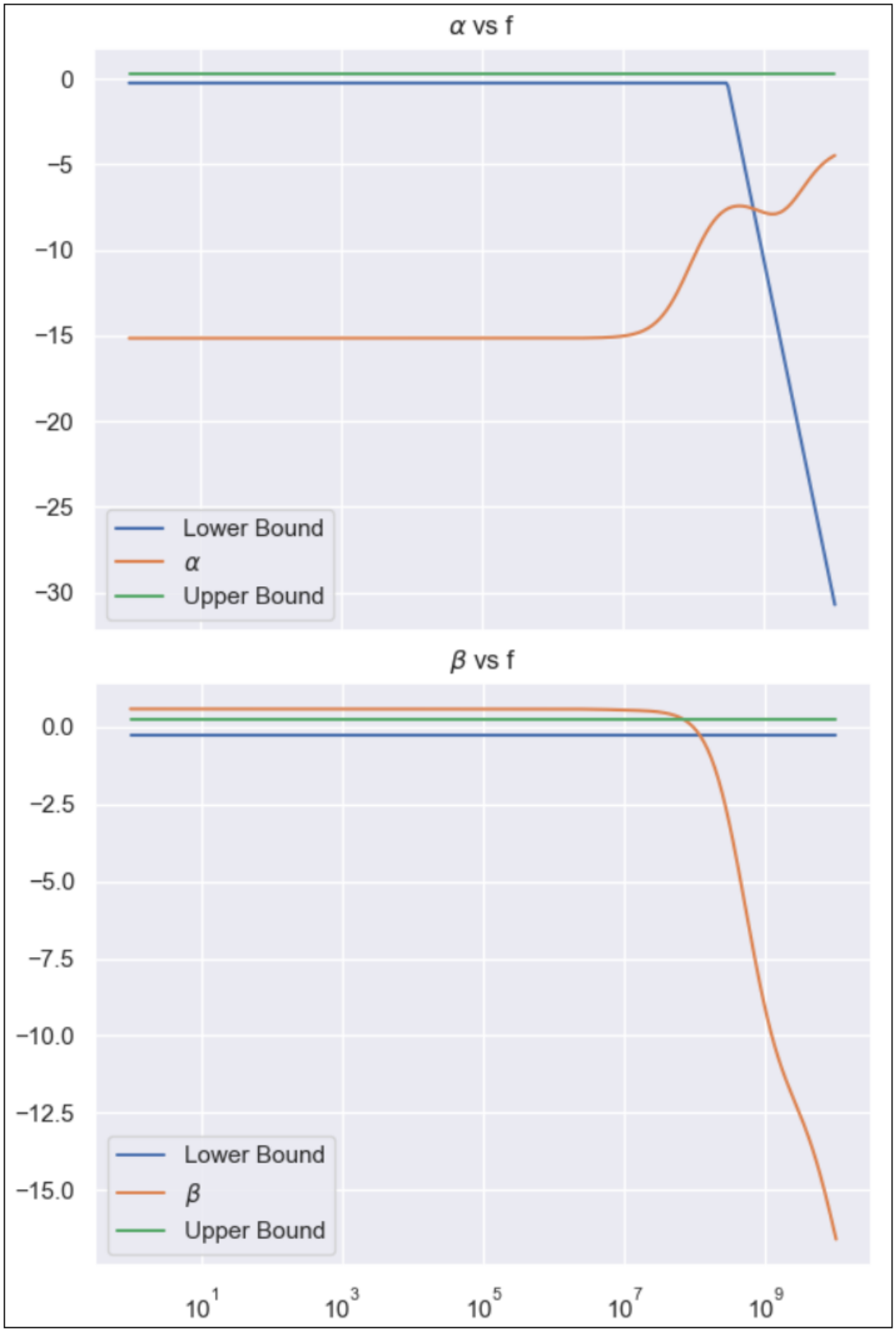

Figure 5.10: Cross Validation of 8 Transistor CCII+.

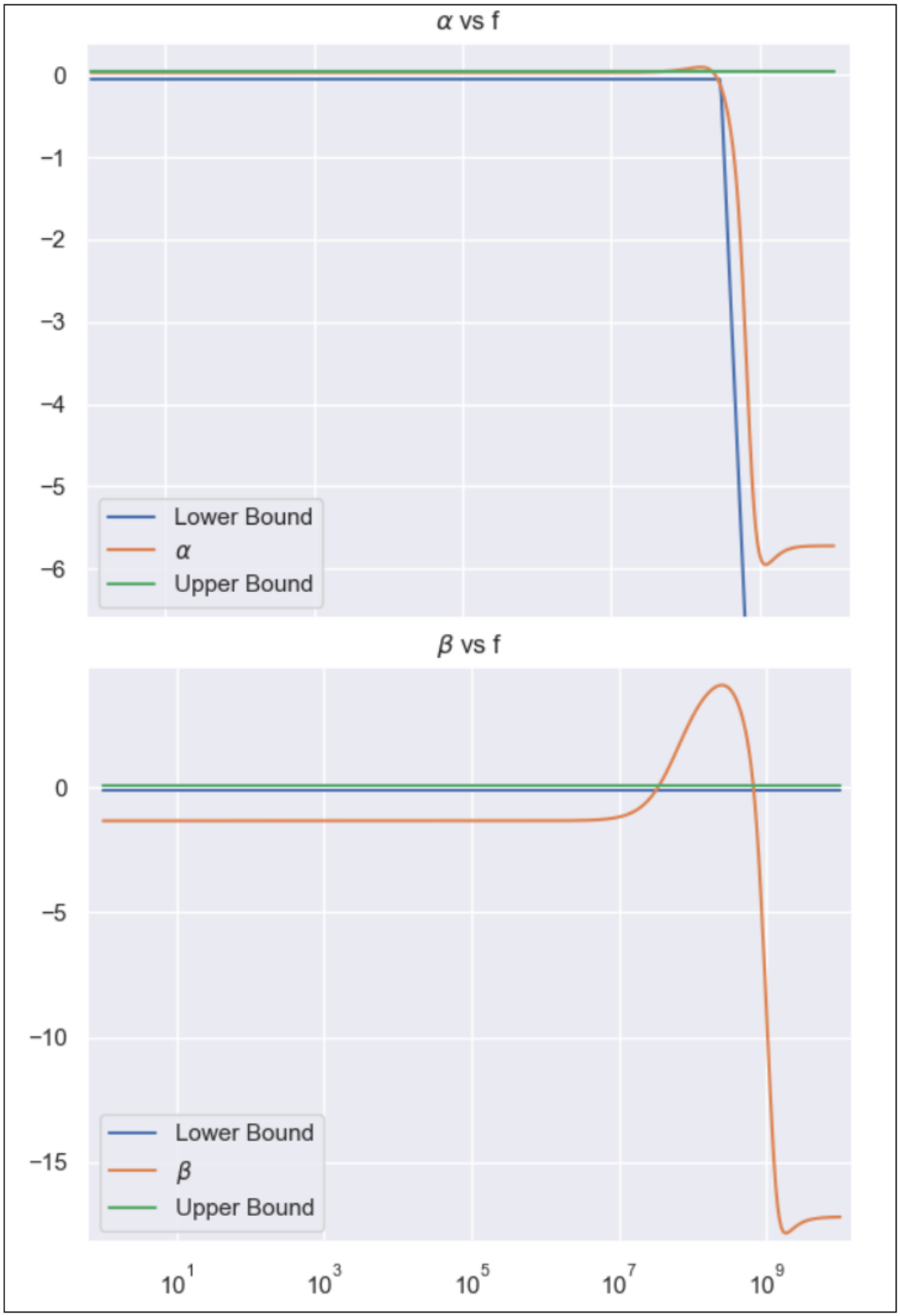

Figure 5.11: Cross Validation of 13 Transistor CCII+.

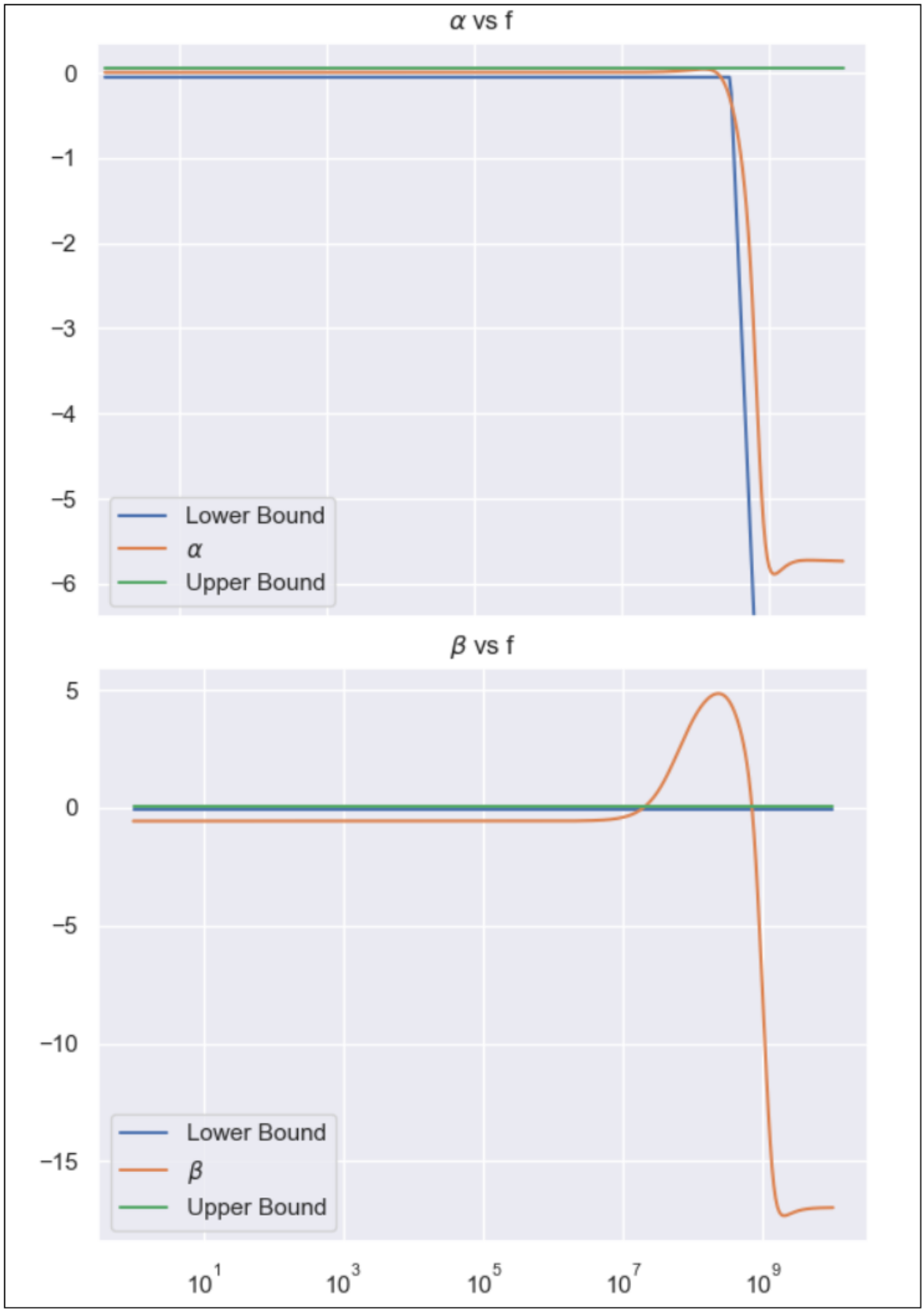

Figure 5.12: Cross Validation of 15 Transistor CCII+.

5.3. Topology Agnosticism

During the training of all circuits, except for the outputs of the neural network, none of the parameters about the architecture of the agent is changed. The only changed thing is the output of the neural network for properly perturbing the circuit based on the number of transistors in the circuit. In addition to that, the same agent was able to learn how to size the CCII circuits even it cannot observe the topology of the circuit. Therefore, it can be concluded that the same agent can learn how to size different type of CCII circuits, and it is *topology agnostic*.

5.4. Limitations

Although the agent was able to learn how to size the CCII circuits, there are some limitations. First of all, the only changed thing about the circuits was the width parameters of the transistors. In general, the lengths of the transistors should also be perturbed. Generally, the lengths of the transistors in analog ICs does not need to be equal but the assumption here was to fix them all at 1 micrometer. Although it is tolerable for now, there are errors in the circuit such that the current mirrors of the topologies should be matched.

Another limitation is related to the handling of the MDP. PG methods generally slower than the competing methods [42]. There are variants of actor-critic algorithms [21],[22], parallelized architectures [40] and value function based methods [38],[56] and other methods [57], [39] and some of which can be applied as the future work.

One last limitation is the SPICE level accuracy of the sizing. Since SPICE is only a simulator, the real circuits can diverge from the SPICE results and the accuracy is heavily dependent upon the accuracy of the model.

6. CONCLUSION

In this thesis, the sizing of the three CCII topologies are assigned to the RL agent and it is shown that it can learn how to size them in a desired behavior. Since this is the first known attempt to use an MDP-based RL approach for the sizing problem of analog ICs, this should be considered as a preliminary study for full application of RL based methods to them and an initial step to fully automatizing the design of analog circuits which is still mainly subject to human optimization dependent on the experience of the designer.

Additionally, to improve the design of circuits, one promising approach may be the extraction of the reward by using inverse RL techniques [58]. By utilizing those, the related reward structure can be extracted from the knowledge of an expert and it can be applied again to the RL agent. By doing so, the more accurate model for the reward can be found.

Another direction for the improvement can be the trial of other algorithms such as actor-critic methods since they frequently learn faster than PG methods. One another algorithm that can be the value function based methods.

Finally, a tree like data structures can take the role of a neural network and that may improve the design.

REFERENCES

[1] Sedra A.S., Roberts G.W., Gohh, F., (1990), "The current conveyor: history, progress and new results", IEEE Proceedings G-Circuits, Devices and Systems, 137-2, 78-87.

[2] Ferri G., Guerrini N.C., (2003), "Low-Voltage Low-Power CMOS Current Conveyors", 1st Edition, Kluwer Academic Publishers.

[3] Senani R., Bhaskar D.R., Singh A.K., (2015), "Current Conveyors - Variants, Applications and Hardware Implementations", 1st Edition, Springer.

[4] Uygur A., Kuntman H., (2013), "An Ultra Low-Voltage, Ultra Low-Power DTMOS-Based CCII Design for Speech Processing Filters", In 2013 8th International Conference on Electrical and Electronics Engineering (ELECO), 31–35, IEEE.

[5] Smith K.C., Sedra A., (1968), "The current conveyor - a new circuit building block", Proceedings of the IEEE, 56, 1368–1369.

[6] Sedra A., Smith K.C., (1970), "A second-generation current conveyor and its applications", IEEE Transactions on Circuit Theory, February, 132–134.

[7] Fabre A., (1995), "Third-generation current conveyor: a new helpful active element", Electronics Letters, 31, 338–339.

[8] LeCun Y., Boser B., Denker J.S., Henderson D., Howard R.E., Hubbard W., Jackel L.D., (1990), "Handwritten digit recognition with a back-propagation network", Advances in Neural Information Processing Systems (NIPS 1989), Denver, CO, 2, 396–404, Morgan-Kaufmann.

[9] Guzella T.S., Caminhas W.M., (2009), "A review of machine learning approaches to spam filtering", Expert Systems with Applications, 36, 10206–10222.

[10] Kontrimas V., Verikas A., (2011), "The mass appraisal of the real estate by computational intelligence", Applied Soft Computing, 11 (1), 443–448.

[11] LeCun Y., Bengio Y., Hinton G., (2015), "Deep learning", Nature, 521, 436–444.

[12] Tesauro G., (1995), "Temporal difference learning and td-gammon", Communications of the ACM, 38 (3), 58-68.

[13] Silver D., Schrittwieser J., Simonyan K., Antonoglou I., Huang A., Guez A., Hubert T., Baker L., Lai M., Bolton A., Chen Y., Lillicrap T., Hui F., Sifre L., van den Driessche G., Graepel T., Hassabis D., (2016), "Mastering the game of Go without human knowledge", Nature, 550, 354–359.

[14] Müller M., (2002), "Computer Go", Artificial Intelligence, 134, 145–179.

[15] Mao H., Alizadeh M., Menache I., Kandula S., (2016), "Resource management with deep reinforcement learning", ACM, Proceedings of the 15th ACM Workshop on Hot Topics in Networks, 50–56.

[16] Michels J., Saxena A., Ng A.Y., (2005), "High speed obstacle avoidance using monocular vision and reinforcement learning". Proceedings of the 22nd International Conference on Machine Learning, 593-600.

[17] Sallab A.E., Abdou M., Perot E., Yogamani S., (2017), "Deep reinforcement learning framework for autonomous driving", Autonomous Vehicles and Machines, 70–76. Society for Imaging Science and Technology.

[18] Sutton R.S., Barto A.G., (2018), "Reinforcement Learning - An Introduction", 2nd edition, MIT Press.

[19] Watkins C.J.C.H., (1989), "Learning from Delayed Rewards", PhD Thesis, King's College.

[20] Sutton R.S., (1988), "Learning to Predict by the Methods of Temporal Differences", Machine Learning, 3, 9–44.

[21] Konda V.R., Tsitsiklis J.N., (1999), "Actor-Critic Algorithms", Proceedings of the 12th International Conference on Neural Information Processing Systems, 1008–1014, MIT Press.

[22] Grondman I., Busoniu L., Lopes G., Babuska R., (2012), "A survey of actor-critic reinforcement learning: standard and natural policy gradients", IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE, 42 (6), 1291–1307.

[23] Williams R. J., (1992), "Simple statistical gradient-following algorithms for connectionist reinforcement learning". Machine Learning, 8, 229–256.

[24] Marbach P., Tsitsiklis J.N., (1998), "Simulation-Based Optimization of Markov Reward Processes", Technical report, Massachusetts Institute of Technology.

[25] Marbach P., Tsitsiklis J.N., (1999), "Simulation-based optimization of Markov reward processes: implementation issues", Proceedings of the 38th IEEE Conference on Decision and Control, 1769–1774, IEEE

[26] Liu B., Fernández F.V., Gielen, G.G.E., (2011) "Efficient and Accurate Statistical Analog Yield Optimization and Variation-Aware Circuit Sizing based on Computational Intelligence Techniques", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 30 (6), 793–805.

[27] Bennett III F. H., Koza J.R., Andre D., Keane M.A., (1996), "Evolution of a 60 Decibel Op Amp Using Genetic Programming", Proceedings of the 1st International Conference on Evolvable Systems (ICES-96), 453-469.

[28] Koza J.R., Bennett III, F.H., Lohn J., Dunlap F., Keane M.A., Andre D., (1997), "Automated Synthesis of Computational Circuits Using Genetic Programming", Proceedings of the 1997 IEEE Conference on Evolutionary Computation, 447–452.

[29] Chatterjee A., Fakhfakh M., Siarry P., (2010), "Design of second-generation current conveyors employing bacterial foraging optimization", Microelectronics Journal, 41 (10), 616-626.

[30] Fakhfakh M., Cooren Y., Sallem A., Loulou M., Siarry P., (2010) "Analog circuit design optimization through the particle swarm optimization technique", Analog Integrated Circ. Sig Process, 63 (1), 71–82.

[31] Vural R.A., Yıldırım T., (2012), "Analog circuit sizing via swarm intelligence". Int. J. Electron. Commun. (AEU), 66 (9), 732–740.

[32] Fish A., Milrud V., Yadid-Pecht O., (2005), "High-speed and high-precision current winner-take-all circuit", IEEE Transactions on Circuits and Systems II: Express Briefs, 52 (3), 131–135.

[33] Zargaran-Yazd A., Sudhakaran S., (2017), "Using deep neural networks to model nonlinear circuit blocks in wireline links", 2017 IEEE 26th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS).

[34] Kahraman N., Kıyan T., (2014), "A neural network-based design automation of a second generation current conveyor", International Conference on Computational Science and Computational Intelligence, 313-314, IEEE.

[35] Iskander R., Louërat M.M., Kaiser A., (2013), "Hierarchical sizing and biasing of analog firm intellectual properties", Integration, The VLSI Journal, 46 (2), 172–188.

[36] Jespers P.G.A., Murmann B., (2017), "Systematic Design of Analog CMOS Circuits", Cambridge University Press, 1st edition.

[37] Jespers P.G.A., (2010), "The g_m/I_D Methodology - a sizing tool for low-voltage analog CMOS Circuits", Springer, 1st edition.

[38] Mnih V., Kavukcuoglu K., Silver D., Graves A., Antonoglou I., Wierstra D., Riedmiller M., (2013) "Playing Atari with Deep Reinforcement Learning", arXiv Preprint, NIPS Deep Learning Workshop 2013, arXiv:1312.5602.

[39] Schulman J., Levine S., Moritz P., Jordan M., Abbeel P., (2015), “Trust Region Policy Optimization”, Proceedings of the 32nd International Conference on Machine Learning, PMLR, 37, 1889-1897.

[40] Mnih V., Badia A.P., Mirza M., Graves A., Harley T., Lillicrap T.P., Silver D., Kavukcuoglu K., (2016), “Asynchronous methods for deep reinforcement learning”. Proceedings of the 33rd International Conference on International Conference on Machine Learning, 48,1928–1937, JMLR.

[41] Sedra A., Smith K.C., (2011), “Microelectronic Circuits”, 6th edition, Oxford University Press.

[42] Sutton R.S., McAllester D., Singh S., Mansour Y., (2000), “Policy gradient methods for reinforcement learning with function approximation”. Advances in Neural Information Processing Systems 12, 1057–1063. MIT Press.

[43] Bengio Y., (2009), “Learning Deep Architectures for AI”, Technical report, University of Montreal.

[44] Breiman L., Friedman J.H., Olshen R.A., Stone C.J., (1984), “Classification and Regression Trees”. CRC Press, 1st edition.

[45] Ogrodzki J., (1994), “Circuit Simulation Methods and Algorithms”, CRC Press, 1st edition.

[46] Singh S.P., Joakkola T., Jordan M.I., (1994), “Learning without state-estimation in partially observable markovian decision processes”. Machine Learning Proceedings 1994, Proceedings of the Eleventh International Conference, 284–292. New Brunswick/NJ/US, July.

[47] LeCun Y., Bottou L., Orr G.B., Müller K.R., (1998), “Efficient backprop”, Neural Networks: Tricks of the Trade, Lecture Notes in Computer Science, 7700, 9–48.

[48] Breiman L., (2001), “Random forests”, Machine Learning, 45 (1), 5–32.

[49] Hubel D.H., Wiesel T.N., (1959), “Receptive fields of single neurones in the cat’s striate cortex”, The Journal of Physiology, 148, 574–591.

[50] Lippmann R.P., (1987), “An introduction to computing with neural networks”, IEEE ASSP Magazine, April, 4–22.

[51] Ng A., (2003), “Shaping and policy search in Reinforcement learning”, PhD Thesis, University of California, Berkeley.

[52] Hahnloser R.H.R., Sarpeshkar R., Mahowald M.A., Douglas R.J., Seung, H.S., (2000), “Digital selection and analogue amplification coexist in a cortex inspired silicon circuit”, Nature, 405, 947–951.

[53] Glorot X., Bordes A., Bengio Y., (2011), “Deep sparse rectifier neural networks”, Proceedings of the 14th International Conference on Artificial Intelligence and Statistics, 15, 315–323, PMLR.

[54] Siripruchyanun M., Silapan P., Jaikla W., (2009), “Realization of CMOS current controlled current conveyor transconductance amplifier (CCCCTA) and its applications”, Journal of Active and Passive Electronic Devices, 4, 35–53.

[55] Hassanein W.S., Awad I.A., Soliman A.M., (2005), “New high accuracy CMOS current conveyors”, AEU – International Journal of Electronics and Communications, 59 (7), 384–391.

[56] van Hasselt H., Guez A., Silver D., (2016), “Deep Reinforcement Learning with Double Q-Learning”, Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, 2094–2100, AAAI Press.

[57] Schulman J., Wolski F., Dhariwal P., Radford A., Klimov O., (2017), “Proximal Policy Optimization Algorithms”, arXiv Preprint, arXiv:1707.06347.

[58] Ng A.Y., Russell S., (2000), “Algorithms for inverse reinforcement learning”, Proceedings of the Seventeenth International Conference on Machine Learning, ICML 2000, 663-670, Morgan Kaufmann.

BIOGRAPHY

Mustafa Alperen SARI was born in Istanbul, 1987. He received his B. Sc. degree in Electrical and Electronics Engineering from Middle East Technical University Northern Cyprus Campus in 2015. Currently, he is pursuing towards his M. Sc. degree in Electronics Engineering from Gebze Technical University.

APPENDIX

Other Appendix (CD)