

# 國立交通大學

電子工程學系 電子研究所

## 碩士論文

使用敏捷式基因探勘與隨機最佳化來改善類比電路

合成的效率

On Improving Analog Synthesis Efficiency via Agile  
Genetic Exploration and Stochastic Optimization

研究生：林建志

指導教授：陳宏明教授

江蕙如教授

中華民國一〇一年十月

使用敏捷式基因探勘與隨機最佳化來改善類比電路合成的  
效率

On Improving Analog Synthesis Efficiency via Agile Genetic  
Exploration and Stochastic Optimization

研究生: 林建志

Student: Chien-Chih, Lin

指導教授: 陳宏明教授

Advisor: Prof. Hung-Ming Chen

江蕙如教授

Prof. Hui-Ru Jiang



A Thesis

Submitted to Department of Electronics Engineering and  
Institute of Electronics

College of Electrical and Computer Engineering

National Chiao Tung University

In partial Fulfillment of the Requirements

for the Degree of

Master of Science

In

Electronics Engineering

October 2012

Hsinchu, Taiwan, Republic of China

中華民國一〇一年十月

# 使用敏捷式基因探勘與隨機最佳化來改善類比電路合 成的效率

學生: 林建志      指導教授: 陳宏明教授

江蕙如教授

國立交通大學 電子工程學系 電子研究所 碩士班

## 摘 要

此篇論文提出了一個在類比電路的階層合成框架上的性能探勘技術和一個動態非均勻的模擬技巧。不同於規格針對性的設計，這篇研究主要是透過平行化的基因演算法探勘類比電路效能的極限，以達到尋找出比人為所不易找到的類比電路設計結果。不同於其他基於演化的拓撲探勘，這個方法能夠把性能視為基因組合來用在演化上且利用多人口群的特點來解決多目標的問題。在人口群裡所選擇的性能能夠利用重新針對的技巧轉換為器件參數。基於把器件參數正規化，一個概率動態模擬顯著地減少找到電路性能全域最佳解的收斂時間。這個演算法發展於分散式的 OpenMp。實驗結果顯示出我們提出的類比電路合成方法在不同製程上的 RFDA 和 Op-Amp 電路能夠得到更好的運行時間且有更高的品質。

# On Improving Analog Synthesis Efficiency via Agile Genetic Exploration and Stochastic Optimization

Student: Chien-Chih Lin

Advisor: Prof. Hung-Ming Chen  
and Prof. Hui-Ru Jiang

Department of Electronics Engineering  
Institute of Electronics  
National Chiao Tung University

## ABSTRACT

This thesis presents a performance exploration technique and a stochastic non-uniform simulation in hierarchical synthesis framework for analog circuit. Different from spec targeted designs, this proposed approach can help to search the solutions better than designers' expectation. A parallel genetic algorithm method is employed for performance exploration. Unlike other evolution-based topology explorations, this is a method that regards performance constraints as input genome for evolution and resolves the multiple-objective problem with the multiple-population feature. Populations of selected performance are transferred to device variables by re-targeting technique. Based on a normalization of device variable distribution, a probabilistic stochastic simulation significantly reduces the convergence time to find the global optima of circuit performance. This algorithm is developed and run on distributed OpenMP. Experimental results show that our approach on radio-frequency distributed amplifier (RFDA) and folded cascode operational amplifier (Op-Amp) in different technologies can obtain better runtime and higher quality in analog synthesis.

## 致 謝

在這兩年的碩士生涯中，最要感謝的是我的指導教授陳宏明老師，老師在我人生最低潮的時候給了我一些鼓勵和人生方向，也在這一年多來給了我很多做研究上的技巧和方法，對於學習與研究上的關心更是給了我很大的動力，真的很感謝老師的諄諄教誨，以後在社會與工作上也能夠時時提醒自己。

在這段VDA的日子裡，各個學長姐的照顧真的是很感謝，你們做研究的態度也是我學習的好榜樣。更要再次的謝謝Benbean，給了我很多研究上的建議和方向，也幫助我在做研究與寫論文上的瓶頸，真的非常感謝，讓我進步了很多很多。仁國，謝謝你幫我了這麼多忙，每次問你問題你總是心平氣和的幫我解決。均鴻，總是在我氣餒的時候給我一些鼓勵，一起努力的時光真的很令人懷念。冠廷，一起打球的時光總是很快樂，不服輸的精神也讓我見識到了呢！新鈞和川嘉，你們的聰明與對課業的努力執著我仍然遠遠不及呀！也謝謝你們倆平時的加油打氣！以恩，一起修課時的那些時光謝謝你總是幫我解決許多疑惑。孟伶，感謝妳平常meeting前為我們準備的，也辛苦妳的跑腿報帳了！

謝謝我的女友，卉凌，謝謝妳這一年多來的容忍及鼓勵，在我最低潮時總是在我身邊，在研究所生涯裡遇見妳是我一生中最浪漫的事。

謝謝我的父母以及家人，讓我沒有後顧之憂的攻讀碩士，謝謝你們平常的關心以及鼓勵，這份碩士畢業證書對我來說意義重大。我想跟你們說：爸媽，我做到了。

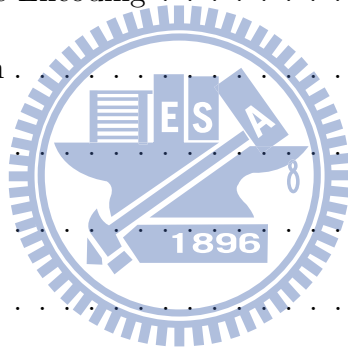
最後我把這份喜悅要獻給我在上天最敬愛的爺爺及奶奶。

# Contents

<b>Abstract (Chinese)</b>	<b>i</b>
<b>Abstract</b>	<b>ii</b>
<b>Acknowledgements</b>	<b>iii</b>
<b>Contents</b>	<b>iv</b>
<b>List of Tables</b>	<b>vi</b>
<b>List of Figures</b>	<b>vii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Previous Works . . . . .	2
1.2 Our Contributions . . . . .	4
<b>2 Preliminaries</b>	<b>7</b>
2.1 Problem Description . . . . .	7
2.2 Hierarchical Performance Pareto-front Mapping Methodology . . . . .	8
2.2.1 Overview . . . . .	8
2.2.2 Device Fitting . . . . .	9



2.2.3	Performance Space Exploration . . . . .	9
2.2.4	Design Re-targeting . . . . .	10
2.2.5	Stochastic Fine Tuning . . . . .	11
<b>3</b>	<b>Utmost Performance Space Exploration via Multi-objective Parallel Genetic Algorithm and Stochastic Optimization</b>	<b>12</b>
3.1	Introduction to Parallel Genetic Algorithm . . . . .	12
3.2	Parallel Genetic Algorithm on Performance Space Exploration . . . . .	15
3.2.1	Overview . . . . .	15
3.2.2	Chromosome Encoding . . . . .	16
3.2.3	Initialization . . . . .	18
3.2.4	Evolution . . . . .	18
3.2.5	Migration . . . . .	20
3.2.6	Merge . . . . .	20
3.3	Probabilistic Stochastic Circuit Simulation . . . . .	21
<b>4</b>	<b>Experimental Results</b>	<b>27</b>
4.1	Performance Exploration Stage . . . . .	28
4.2	Stochastic Fine-tuning Stage . . . . .	29
<b>5</b>	<b>Conclusion</b>	<b>38</b>



# List of Tables

4.1	Device statistics of RFDA and OpAmp . . . . .	27
4.2	The population results and genetic exploration runtimefor RFDA and Op-Amp circuit with umc 65nm, umc 90nm and tsmc 90nm technologies on GA <sub>256</sub> , PGA <sub>120</sub> and PGA <sub>256</sub> based performance exploration. . . . .	28
4.3	Best performance of population results for RFDA and Op-Amp circuit with umc 65nm, umc 90nm and tsmc 90nm technologies on GA <sub>256</sub> , PGA <sub>120</sub> and PGA <sub>256</sub> based performance exploration. . . . .	29
4.4	The performance exploration results for RFDA and Op-Amp circuit with umc 65nm, umc 90nm and tsmc 90nm technologies on [16], GAP <sub>256</sub> , PGAP <sub>120</sub> , PGANP <sub>120</sub> ,PGAP <sub>256</sub> and PGANP <sub>256</sub> framework. . . . .	37

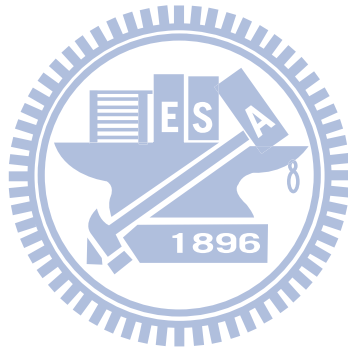


# List of Figures

1.1	While traditional sizing strategy iteratively search the solution beyond the fixed spec, a performance constraints exploration approach unfolds the utmost of performance space. Moreover, an evolutionary methodology application elaborates exploration faster and more precise.	4
2.1	The Complete Performance Exploration Flow.	8
2.2	Mapping device-level variables of one NMOS(number of fingers, device channel width/length and current) to circuit-level variables ( $G_m$ , $R_o$ , $C_D$ , $C_G$ and $\Sigma$ )	10
2.3	Given merged populations of performance specs, a reversed process is to retrieve the corresponding design variables of each device (PMOS) in the circuit for optimal sizing values.	11
3.1	Three different model of parallel genetic-algorithm: (A) master-slave, (B) coarse-grained, and (C) fine-grained.	13
3.2	Two different environment of migration topology: (A) network topology, and (B) ring topology.	14
3.3	Illustration of performance characteristic (e.g., $A_v$ , $P_{dc}$ , $P_{out}$ , $BW$ , $F_{cent}$ , etc.) for each circuit.	16

3.4	The Coarse-grained parallel genetic approach from major population P to partitioned population $P_i$ for parallel evolution. The migration step benefits each sub-population on diversity every iteration. . . . .	19
3.5	When sub-population $P_2$ and sub-population $P_4$ perform migration operation with migration rate $M_P$ , $P_2$ sends its best individual with respect to spec type $P_4$ , and the receiving sub-population $P_4$ replaces its worst individual with respect to its own spec type $P_4$ . . . . .	21
3.6	Compared with different distribution for stochastic simulation. (a) Population samplings for each device level variable from population and the global optimum location. (b) Normal distributions for each device level variable after normalize and the global optimum location.	22
3.7	Illustration an example of swap function in MOSA step by step. . . . .	26
4.1	Population distribution of the RFDA design instance using UMC65nm after genetic exploration. (a), (c) and (e) are plotted with respect to coordinates of PDC, Pout, and Av. (b), (d) and (f) are plotted with respect to coordinates of PDC, fcent and BW. . . . .	30
4.2	Population distribution of the Op-Amp design instance using UMC65nm after genetic exploration. (a), (c) and (e) are plotted with respect to coordinates of PDC, Pout, and Av. (b), (d) and (f) are plotted with respect to coordinates of PDC, fcent and BW. . . . .	31
4.3	Population distribution of the Op-Amp design instance using UMC90nm after genetic exploration. (a), (c) and (e) are plotted with respect to coordinates of PDC, Pout, and Av. (b), (d) and (f) are plotted with respect to coordinates of PDC, fcent and BW . . . . .	32

4.4 Population distribution of the Op-Amp design instance using TSMC90nm after genetic exploration. (a), (c) and (e) are plotted with respect to coordinates of PDC, Pout, and Av. (b), (d) and (f) are plotted with respect to coordinates of PDC, fcent and BW . . . . . 33



# Chapter 1

## Introduction

Nowadays, analog circuit block in a SoC design is often critical. Analog components are heavily influenced by nonlinear physical effects, which create a great barrier for analog automation. Overall, the analog synthesis process consists of topology generation [15, 13, 14], circuit sizing [7] and layout synthesis. As topology generation are fully illustrated in [15, 13, 14], the circuit sizing methodologies also take the indispensable role. Comparing to digital system, the development for analog design automation is still in development. Either topology selection or circuit sizing methodologies are considered in time complexity and the accuracy caused by process variation, parasitics effect and operation conditions.

There are plenty of works which are already well-developed on how to find optimal design parameters for a prior selected topology. However, as many performance specifications need to be considered, finding the optima solution for multi-objective performance at sizing stage as a priori problem is still uncertain. It seems that deterministic optimization for circuit sizing still has space to improve. Since deterministic optimization keeps the efficiency and full-circuit SPICE-based simulation maintains the accuracy, selecting methodology for analog circuit sizing is beyond trade-off.

It is therefore essential to have an agile multi-objective synthesizer which explores

the limitation towards required technology. Moreover, it is capable of searching the performance space and re-targeting to design parameters with accuracy and efficiency for analog circuit design perspectives. One such searching method is genetic programming. Genetic algorithm is a single objective optimization algorithm that mimics the process of natural evolution[8]. In genetic algorithm, several genes encode to a chromosome and a group of chromosomes forms population. In each iteration of genetic algorithm, chromosomes of the population are evolved by way of one or more operation such as reproduction, crossover and mutation. These evolutionary operation terminated when termination condition is met. If the algorithm has terminated, a satisfactory solution may have been reached. In spite that genetic algorithm is able to find good solutions in optimization problem, but as time passes, the scale of optimization problem becomes bigger and bigger. We massive population to obtain good solution, so the time consumption is getting critical. Usually traditional genetic algorithm just maintain one population. Actually, the environment of natural evolution maintain many groups and with higher diversity. It is possible to achieve this environment by parallel genetic algorithm. Parallel genetic algorithm is not just a parallel version of traditional genetic algorithm. Parallel genetic algorithm actually reaches the ideal goal of having a good parallelism and maintains the overall diversity via parameters setting. That makes parallel genetic algorithm a better search and optimization algorithm than traditional genetic algorithm.

## 1.1 Previous Works

In algorithm of [20], one of the earliest GA-based analog synthesizer has been proposed. In the proposed algorithm, chromosomes encoded as a list of circuit topology and transistor size. By the way, [12] also elaborates the genetic programming on topology selection generation and parameters optimization. Similar to [20], where each chromosome is encoded as circuit topology, McConaghy et al. [14] propose a

method to generate templet-free symbolic performance models of analog circuits. In this thesis, McConaghy et al. have shown the usage of multi-objective genetic algorithm to solve the problem. Another analog synthesizer has been proposed in [4]. Conca et al. introduce an algorithm, that is able to synthesize an analog circuit using industrial components and design better circuits in terms of frequency response and number of components.

State-of-the-art analog synthesis methodologies were formulated as a numerical problem which relies on macro-modeling. Usually a complete circuit schematic and the circuit's performance spec are given, then the sizes and biasing value of all devices have to be determined. As a result, the optimal values meet the specs of the required circuit. This kind of optimization engine determines these optimal values and there exists an evaluation engine to assess the performance. It is very likely that the initial sizing result in a near-optimal design, therefore, a further fine-tuning follows for improving yield and design robustness [18].

In all, due to the requirement for more probability on performance metrics, we believe that it is important to have mechanism at the early design stage in exploring the performance limitation before optimization. Meng et al.[16] provide a hierarchical performance Pareto-front mapping methodology to acquire performance metrics before circuit optimization stage. Unlike traditional optimization approaches which put emphasis on design variables as input, [16] first traverses the performance specs as constraints in a set of convex problem. Therefore, a combination of performance space is generated. Moreover, the corresponding design variables can be obtained according to the design equation. Labrak et al. [10] perform a hybrid optimizer with a multi-objective optimization problem(MOOP) for a CMOS Op-Amp. Our method integrates the hierarchical synthesis strategy with a performance mapping methodology and a non-uniform circuit simulation approach to meet the multi-objective performance requirement wisely.

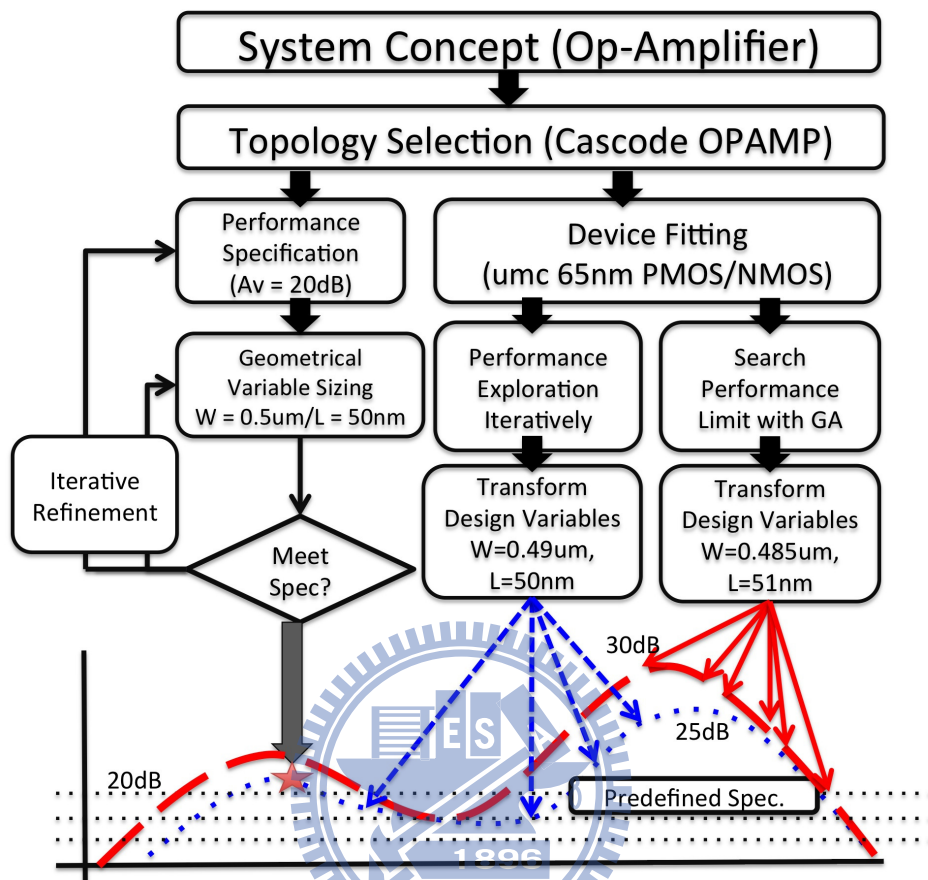
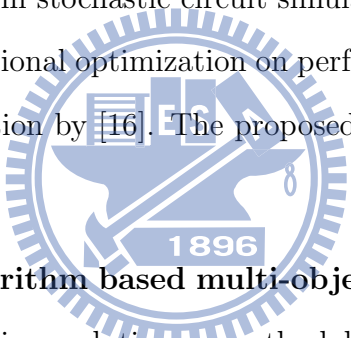


Figure 1.1: While traditional sizing strategy iteratively search the solution beyond the fixed spec, a performance constraints exploration approach unfolds the utmost of performance space. Moreover, an evolutionary methodology application elaborates exploration faster and more precise.

## 1.2 Our Contributions

According to the trade-off among performance specs on analog designs, it is rarely possible to find an optimal solution for all metrics (ex: voltage gain, bandwidth, output power or power dissipation). Therefore, it is practical to search the performance limitation with agile and accurate synthesis procedure. Referring to [16], Meng et al. attempt to search the performance space by re-targeting back to the corresponding design variables, in iterative manner. Nevertheless, the strategy which costs time complexity up to  $O(N^K)$  ( $N$  stands for the performance range in discrete number and

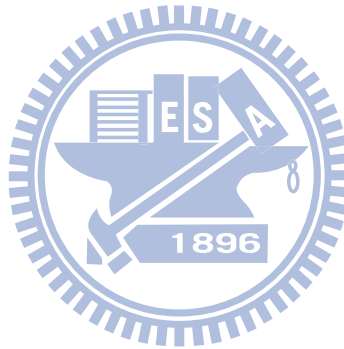
K represents the number of performance specification), which is time-consuming and will lose the effective prediction of performance metrics. The proposed work employs a coarse-grained parallel genetic algorithm for traversing performance space as optimization constraints. A parallel genetic algorithm like [3, 1, 11] can produce a more realistic model of nature. The classic parallel genetic algorithm is to divide the target population into several sub-populations. With variations in model, parameter and topology settings, the evolution environment would be more flexible. According to these features, a parallel genetic algorithm can divide the target population into several sub-populations with particular performance specs and reunion after evolution. The obtained populations resulted from genetic exploration can be re-targeted to the non-uniform stochastic circuit simulation. Fig. 1.1 illustrates the comparison among the traditional optimization on performance specification, the iterative performance exploration by [16]. The proposed work achieves two principle contributions as follows:

- 
**• Parallel genetic algorithm based multi-objective performance exploration.** A multi-objective evolutionary methodology like parallel genetic algorithm not only performs evolution in parallel for diverse performance metrics, but also migrates chromosome by interleaving among populations. Parallel genetic algorithm brings out a population of potential solution for selected performance among metrics. Such population is also projected to the feasible design parameters so that we affirm the global optimal solution is located nearby this optimization result by exploration.
- Non-uniform stochastic circuit simulation.** The performance space not only stands for the potential optimum, but also represents the possibility of suitable design parameters which cause better performance. This work integrates each design variables to analyze the possibility distribution. Other than uniformly swapping the values of design variables in stochastic searching, a in-



terval with higher possibility earns more searching resources. A non-uniform step searching for circuit SPICE simulation is proposed.

The rest of this thesis is organized as follows, Chapter 2 first defines problem formulation and introduction a hierarchical performance Pareto-front mapping methodology [16]. In Chapter 3, we apply our parallel genetic algorithm to improve the performance exploration and introduction a probabilistic multi-objective stochastic optimization. Finally, we draw a conclusion in Section 5.



# Chapter 2

## Preliminaries

### 2.1 Problem Description

State-of-the-art acknowledges a collection of Pareto-fronts which sketch the performance space, later an optima point is selected for local search problem which didn't mention that how to define optima point among the space. Instead of collecting the performance space information, this thesis aggressively define performance limit as exploration main objective:

**Definition 1 Circuit Performance:** *A circuit performance is consisted of multiple values, such as DC voltage gain, 3dB gain bandwidth and power consumption. Different circuit has different circuit performance target.*

**Definition 2 Performance limit exploration for global search problem:** *Given a circuit design equation in posynomial forms with a set of feasible circuit-level design variables and a set of circuit performance constraints, perform convex optimization with different performance value to traverse the utmost performance space of the given circuit.<sup>1</sup>*

---

<sup>1</sup>Here, the maximum and minimum performance values are investigated whether feasible or not. Therefore, it can tell that the global search process generates a space of feasible performances and each represents a set of optimal design variables. Although the global search obtains a space of performance, it is not the exact optimal solution. The global search is a preparation for later local search. This work proposes a flexible non-uniform stochastic simulation as local search for optimal sizing solution.

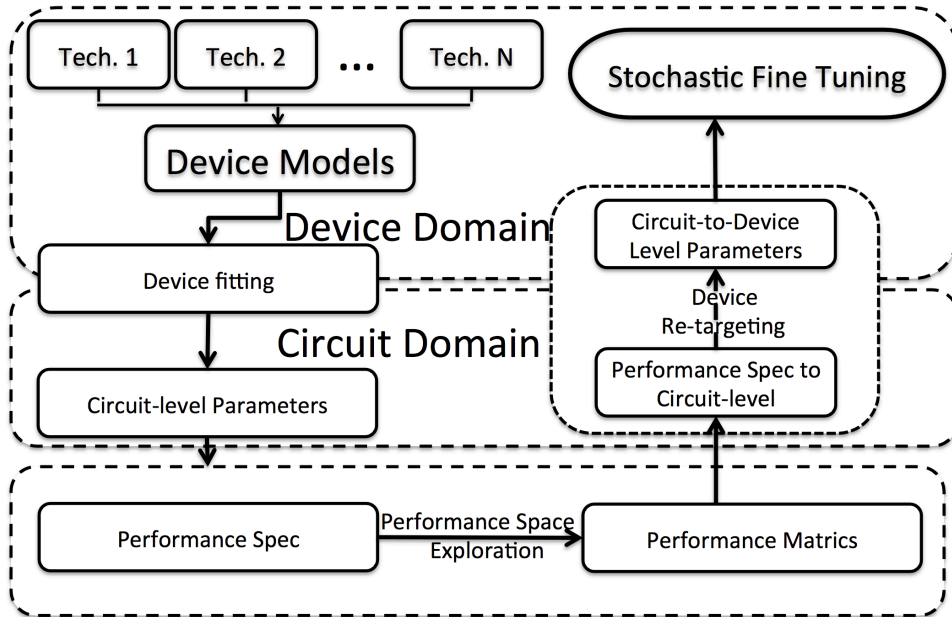


Figure 2.1: The Complete Performance Exploration Flow.

**Definition 3 Stochastic simulation for local search:** *A set of feasible performance is re-targeted to corresponding design variables. The local search practices a stochastic SPICE simulation w.r.t. these selected design variables.*

## 2.2 Hierarchical Performance Pareto-front Mapping Methodology

### 2.2.1 Overview

The framework of overall methodology is summarized in Fig. 2.1. Here, the process to find solutions for multiple performance targets can be divided into bi-direction search stages. The bottom-up global search stage begins from device model level to feasible performance limitation, and the top-down local search starts from the performance optima to re-target back to design variables, and then performing a guided stochastic simulation for optimality. A series of technology candidate are prepared in *Device Fitting*. As the feasible design parameters are collected, *Performance Space Exploration* attempts to search the performance space. In

*Geometric Design Re – targeting* step, geometry-biasing-level design variables of devices are determined. Then the *Stochastic FineTuning* process takes the optimization results of previous steps as an initial guess. Moreover, we can tell that the global optimal is close to such optimization result.

### 2.2.2 Device Fitting

At the begin of synthesis framework, we have an abstraction from device model to circuit-level variables is performed. Given the required device of the target circuit design, the foundry device models provide such device characteristics with SPICE modeling. Inspired by [16], a set of analytical design equations are capable to map device-level variables into circuit-level design variables by modeling techniques such as symbolic analysis or curve fitting like [9, 6, 5]. Two steps of technique accomplish the abstraction for design variables. First of all, it is necessary to discover feasible variable values for each attendant device. In other words, all design variable values which cause device failed should be exclusive at this stage. By SPICE simulator, a matrix of accessible device level variable value are generated. Secondly, such matrix of device-level variables are further mapped into circuit-level variables by curve fitting. An vivid example for design variable mapping is shown in Fig.2.2.

### 2.2.3 Performance Space Exploration

The second step is a circuit-level performance space exploration. From previous step, the design equations of the devices, along with the parasitic effects and fitting parameters, are integrated into circuit-level design equations for performance space exploration of the circuit. Note that the parasitic effects of the devices are included so as to explore the trade-off between each aspect of performance metrics and circuit-level design variables. A set of specification are swept as the constraints for an optimization performance to get performance metrics iteratively.

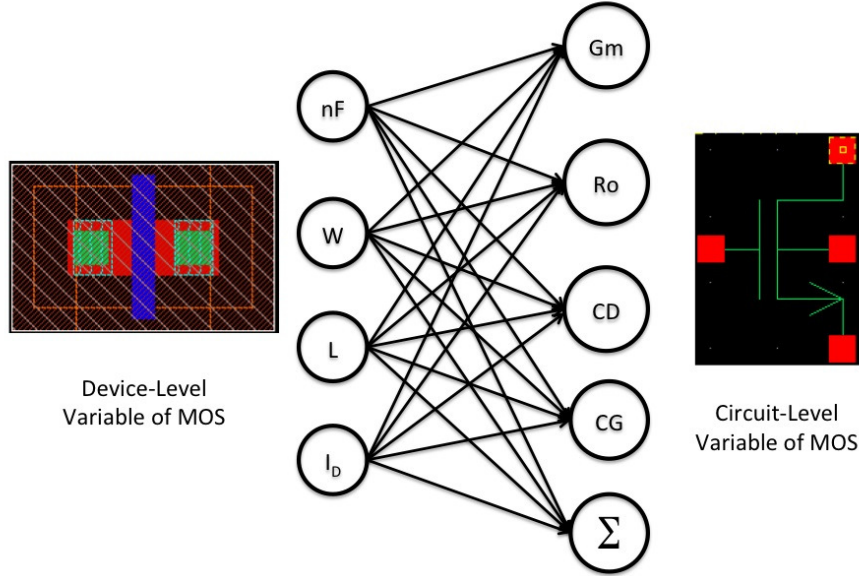


Figure 2.2: Mapping device-level variables of one NMOS(number of fingers, device channel width/length and current) to circuit-level variables ( $G_m$ ,  $R_o$ ,  $C_D$ ,  $C_G$  and  $\Sigma$ )

#### 2.2.4 Design Re-targeting

After generating specialized populations by parallel genetic algorithm evolution, this step is a reverse interpolation from a series of performance specifications through circuit-level design variables to device-level design variables. Hence,  $K$  groups of optimal-potential performance spaces of the circuit under chosen technology are traversed. Since a set of performance metrics directly represents the limitation of specifications, such group of optimal performance specification is locked from optimal performance. Ideally, the optimization engine should be capable of directly finding the geometry-biasing-level design variables.

Next, we want to find the optimal candidates of device-level design variables. From previous stage, the circuit-level design variables are obtained. Here the distribution of the device-level design variables are also obtained through this design re-targeting stage.

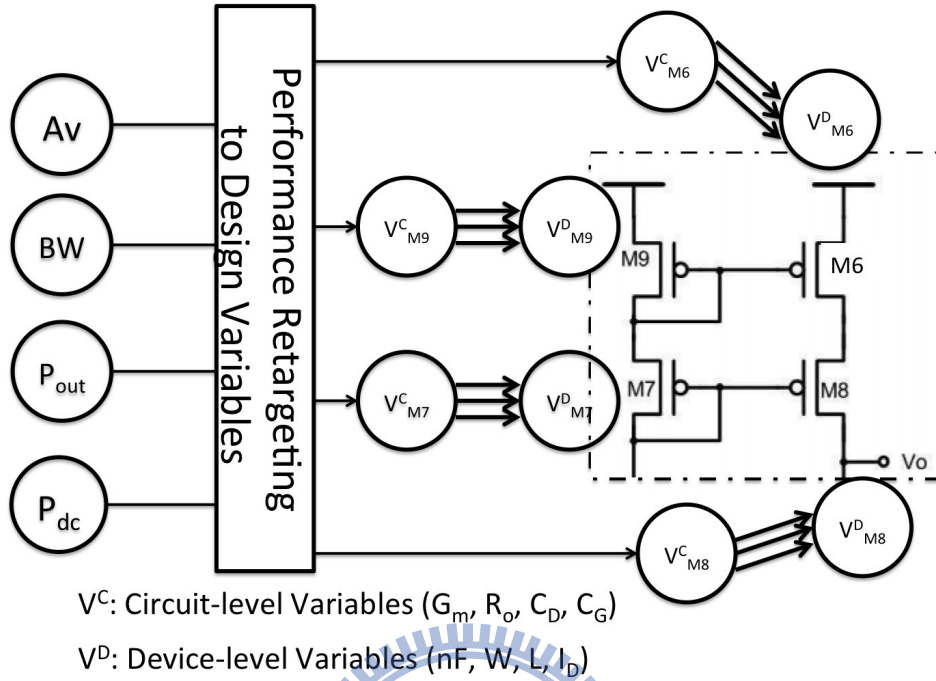



Figure 2.3: Given merged populations of performance specs, a reversed process is to retrieve the corresponding design variables of each device (PMOS) in the circuit for optimal sizing values.

## 2.2.5 Stochastic Fine Tuning

The final step performs a refinement on these potential device-level design parameters. The design variables w.r.t. the Pareto Front of performance metrics are collected by previous step. By employing global optimization algorithm, such as simulated annealing, the entire circuit is fed into SPICE simulator using real models and parameters from the foundry. Therefore the stochastic simulation searches the global optimized value.

## Chapter 3

# Utmost Performance Space Exploration via Multi-objective Parallel Genetic Algorithm and Stochastic Optimization



In analog circuit synthesis, resolving optimization problem is a complex problem which may result in time-consuming and inextricable. Unlike performance exploration iteratively and uniform distribution stochastic fine-tuning in Meng et al.[16], this chapter presents two efficient algorithm for analog circuit synthesis via resolving multi-objective optimization problem.

### 3.1 Introduction to Parallel Genetic Algorithm

Different from traditional genetic algorithm, the basic idea of parallel genetic algorithm is divide-and-conquer based, and can be applied to genetic algorithm in many different variations. Alba et al. [1] and Cant-Paz [3] classify the parallel genetic algorithm into three main types: (1) master-slave genetic algorithm, (2) fine-grained genetic algorithm, and (3) coarse-grained genetic algorithm. Master-slave genetic algorithm still maintain a big population, but each slave processes evaluate and calculate the fitness value. Fine-grained genetic algorithm suited for

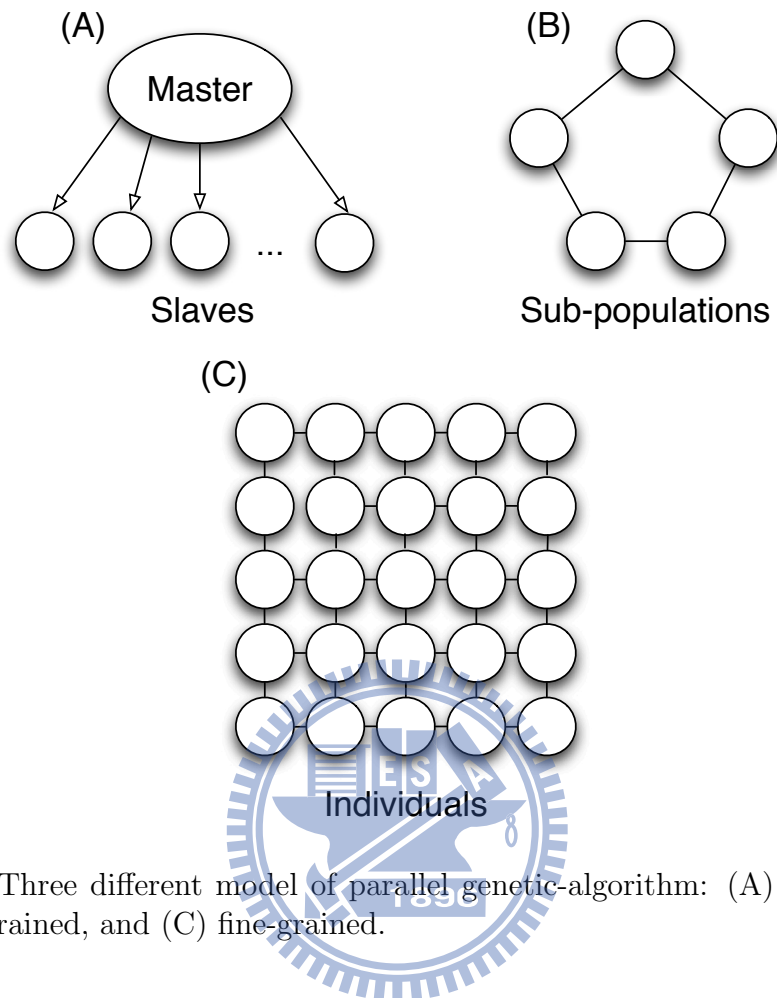


Figure 3.1: Three different model of parallel genetic-algorithm: (A) master-slave, (B) coarse-grained, and (C) fine-grained.

enormous amount parallel computers since each sub-population contains of one individual and assigned to one processor. Coarse-grained genetic algorithm divide a major population to several sub-population and evolve independently. The local reproduction, crossover and mutation operation allow the sub-population to evolve locally, and diversity is enhanced by migration. Fig. 3.1 shows three types parallel genetic-algorithm as mentioned above.

In coarse-grained genetic algorithm, migration is a important communication mechanism and has three main parameter to set: (A) migration topology, (B) migration rate, and (C) migration gap. As with the animal migration, migration topology determines the environment where animal migrates. Fig. 3.2 shows two examples of migration topology. More complex topology lead to more communica-



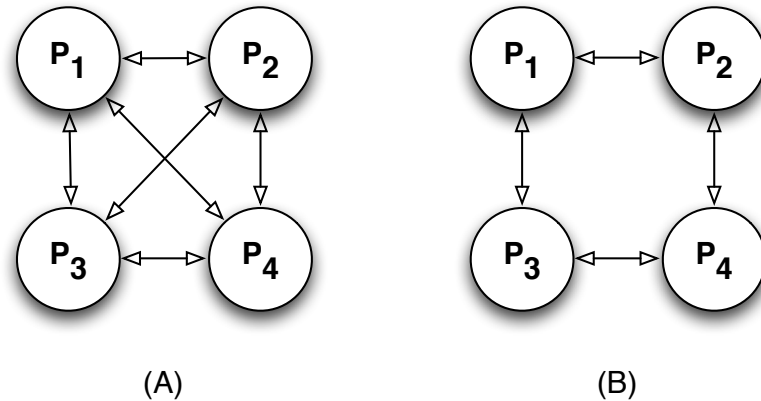


Figure 3.2: Two different environment of migration topology: (A) network topology, and (B) ring topology.

tion overhead and vice versa. Each sub-population exchange their individuals with migration rate and every migration interval called migration gap.

Coarse-grained genetic algorithm homogeneity is classed into two types. In homogeneous coarse-grained genetic algorithm, every sub-population has same parameters (population size, crossover rate, mutation rate, migration interval, etc.), genetic operators, objective functions, and encoding methods, but the heterogeneous coarse-grained genetic algorithm allow the sub-populations have different parameters [11]. Multi-objective optimization problem can achieved via different parameter and exchange their individuals between each sub-population.

Parallel genetic algorithm is just not parallel versions of traditional genetic algorithm, parallel genetic algorithm can actually reach the ideal goal via various parameters setting. For each of the parallel genetic algorithm type, there are multiple variations proposed by researchers to improve its performance or to suite a particular problem.

## 3.2 Parallel Genetic Algorithm on Performance Space Exploration

From *DeviceFitting* step, as mention previous, the design equations of the devices, along with the parasitic effects and mapping parameters are integrated into circuit-level design equations for the circuit. An optimization problem in Eq.(3.1) describes a unit performance optimization step.

$$\begin{aligned}
 \text{Variables : } & v_i^C | 1 \leq i \leq |V^C| & V^C : & \text{circuit variables} \\
 & p_k | 1 \leq k \leq |P| & P : & \text{parasitics} \\
 & f_k | 1 \leq k \leq |F| & F : & \text{fitting parameters} \\
 & r_k | 1 \leq k \leq |R| & R : & \text{performance result} \\
 \text{minimize } & f_{OBJ}(v_i^C, p_k, f_k) & & \\
 \text{subject to } & r_1 = \text{Perf}_1(V^C, P, F) \geq z_1 & & \\
 & r_2 = \text{Perf}_2(V^C, P, F) \geq z_2 & & \\
 & \vdots & & \\
 & r_k = \text{Perf}_k(V^C, P, F) \geq z_k & & 
 \end{aligned} \tag{3.1}$$

Each optimization result in a set of performance value  $(r_1, \dots, r_k)$  corresponding to the given specification of performance  $(z_1, \dots, z_k)$ . Therefore, according to the same design equation for optimization, it is a one-by-one mapping relationship from spec to result of performance and the corresponding circuit-level design variables. As a result, it is similar to a set of chromosome. An evolutionary computing with genetic algorithm for traversing solution space is employed.

### 3.2.1 Overview

Our parallel approach is summarized as shown in Algorithm 1. According to [1], our approach selects the coarse-grained genetic algorithm in order to low connectivity than fine-grained genetic algorithm and simply to achieve. In coarse-grained genetic algorithm, Each sub-population contains a large number of individuals so the migration time between sub-populations is typically smaller than fine-grained

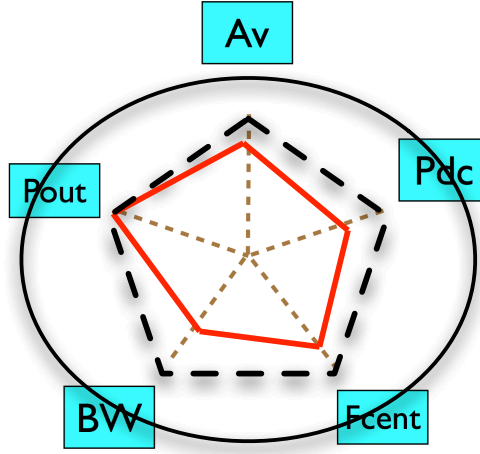


Figure 3.3: Illustration of performance characteristic (e.g., Av, Pdc, Pout, BW, Fcent, etc.) for each circuit.

genetic algorithm. Because this analog sizing is a multi-objective problem as illustrated in Fig. 3.3, so we use heterogeneous coarse-grained genetic algorithm to solve this multi-objective optimization problem. Each sub-population has its own fitness function and evolve independently. By special migration strategy, the diversity can be enhanced. Through these evolution operation, the optimal population can be obtained with respect to technology candidates.

### 3.2.2 Chromosome Encoding

A set of performance boundaries,  $\{[Z_{min}, Z_{MAX}] = \{z_{kmin}, z_{kmax} | 1 \leq k \leq K\}$ , on the performance are swept as the constraints for an optimization problem in [16]. Here we expand the performance space as an  $S \times N$  matrix in Eq.(3.2). Moreover, each performance spec from constraints is encoded as chromosome G from maximal to minimal in a set of  $G = \{g_k | 1 \leq k \leq |G|\}$ . For example,  $g_i \in G$  randomly obtains value of the  $k^{th}$  spec from  $z_{k,1}$  to  $z_{k,S}$ .

---

**Algorithm 1** PGA( $Z_{K \times S}, N_P, S, k, T, C, M_P$ )

---

$Z_{K \times S}$ : Performance Space Matrix.  
 $N_P$ : Number of Individuals in major population.  
 $S$ : Sampling number for each performance spec  $z_k$

**Input:**  $k$ : Number of Performance spec type  
 $T$ : Technology type.  
 $C$ : Circuit type.  
 $M_P$ : Migration rate.

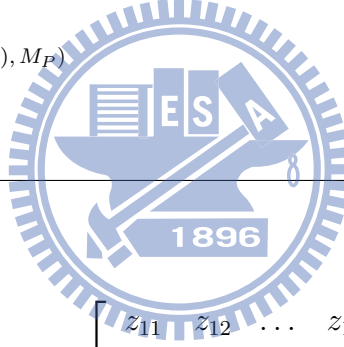
**Output:**  $R_{K \times S}$ : The result performance space  
 $P$ : The population of performance specs after PGA

```

1: for  $i = 1 \rightarrow N_P$  do
2:   for  $j = 1 \rightarrow k$  do
3:      $G_i \leftarrow \text{RandGetSpec}(Z_{j \times S})$  {Randomly generate spec value from  $Z$ }
4:   end for
5:    $P \leftarrow G_i$ 
6: end for
7: for  $i = 1 \rightarrow k$  do
8:    $P_i \leftarrow \text{Partition}(P, i)$ 
9: end for
10: while Convergence criterion satisfied do
11:   for all  $i = 1 \rightarrow k$  do in parallel
12:      $R_i \leftarrow \text{Evaluate}(P_i, T, C)$ 
13:      $\text{Pool}_i \leftarrow \text{Reproduction}(P_i, \text{Fitness}(T, C, i, R_i))$ 
14:      $P_i \leftarrow \text{Crossover}(P_i, \text{Pool}_i)$ 
15:      $P_i \leftarrow \text{Mutation}(P_i)$ 
16:   end for
17:   for  $i = 1 \rightarrow k$  do
18:      $\text{Migration}(P_i, \text{exclusive}(P, P_i), M_P)$ 
19:   end for
20: end while
21:  $P \leftarrow \text{Merge}(P_1, P_2, \dots, P_k)$ 
22: return  $P, R_{K \times S}$ 

```

---



$$Z_{K \times S} = \begin{bmatrix} z_{11} & z_{12} & \dots & z_{1S} \\ z_{21} & z_{22} & \dots & z_{2S} \\ \vdots & \vdots & \vdots & \vdots \\ z_{K1} & \dots & \dots & z_{KS} \end{bmatrix} \quad (3.2)$$

where

- $K$ : the number of performance specification types. (eg: Av, BW, ..., etc.)
- $[Z_{min}, Z_{MAX}]_k$ ,  $k = 1, \dots, K$  is the  $k_{th}$  type specification range of the performance space.
- $S$  is the sampling number for each  $Z_S$  between  $Z_{kmin}$  and  $Z_{kMAX}$ .
- $\forall z_{ki} \in Z_{K \times S}$ , if  $i = 1$ ,  $z_{ki} = Z_{Smin}$  and if  $i = S$ ,  $z_{ki} = Z_{SMAX}$

---

**Algorithm 2** MIGRATION( $P_i, exclusive(P, P_i), M_P$ )

---

$P_i$ : The population which want to migration, where  $i = 1 \dots k$   
**Input:**  $exclusive(P, P_i)$ : Return a population set except  $P_i$ .  
 $M_P$ : Migration rate.

- 1: **for**  $j = 1 \rightarrow k$  **do**
- 2:   **if**  $j \neq i$  **then**
- 3:      $R_P \leftarrow rand(0, 1)$
- 4:     **if**  $R_P \leq M_P$  **then**
- 5:        $I_S = bestIndividual(P_i, fitness(T, C, j, R_i))$
- 6:        $I_R = worstIndividual(P_j, fitness(T, C, j, R_j))$
- 7:        $replaceIndividual(P_i, P_j, I_S, I_R)$
- 8:     **end if**
- 9:   **end if**
- 10: **end for**
- 11:  $P \leftarrow Merge(P_1, P_2, \dots, P_k)$

---

### 3.2.3 Initialization

As algorithm 1 described in *Input*, a set of performance space matrix  $Z_{K \times S}$  is given. At the beginning of parallel genetic algorithm, a major population is constructed according to  $Z_{K \times S}$ . Algorithm 1 Line 1 to Line 6 illustrate the process to assign performance specs as chromosome for each individual of the major population iteratively. Because we want to achieve multi-objective optimization,  $k$  sub-populations are generated and evolved independently. Therefore, a major population  $P$  is generated. According to the size of sub-population  $k$ , master processor allocates individuals to each slave processor uniformly as sub-population  $P_1 \dots P_k$ .

### 3.2.4 Evolution

An evolution is executed between algorithm 1 Line 10 and Line 20. Because the *Evaluate*, *Reproduction*, *Crossover* and *Mutation* parts are independent, the parallel parts can be performed from algorithm 1 Line 11 to Line 16. Hence, in each sub-population, each location of gene in one individual is fed into  $Evaluate(P, T, C)$  as target constraints for performance to the design equation (shown in Eq.(3.1)) and a corresponding result  $R_i$  is obtained. However, if one combination of performance metrics produces the solution space which is not convex, such individual would be failed by *Evaluate*. Then, a random generated individual replaces and redoes

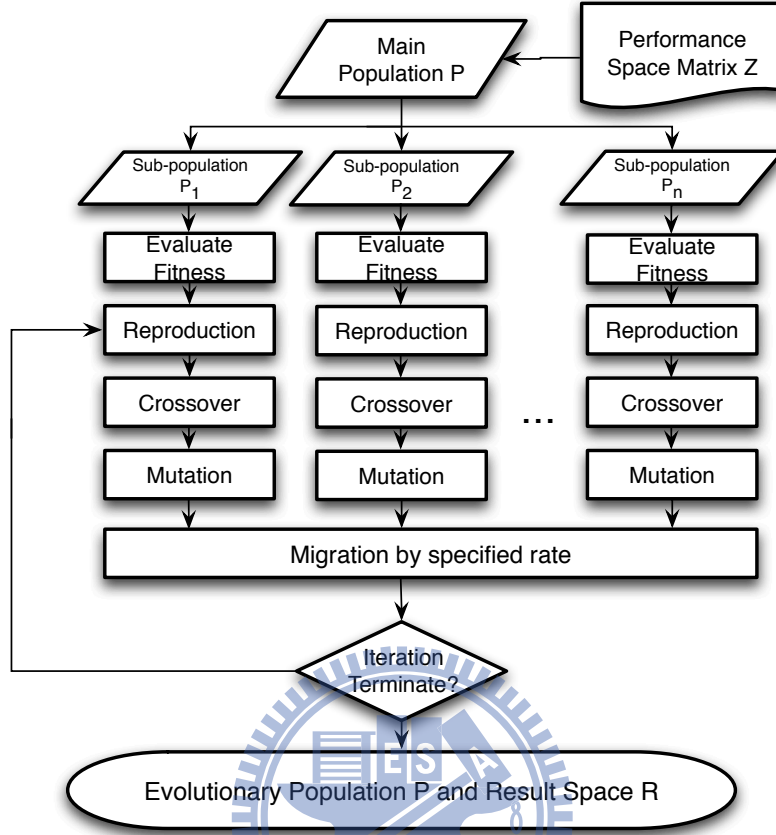


Figure 3.4: The Coarse-grained parallel genetic approach from major population  $P$  to partitioned population  $P_i$  for parallel evolution. The migration step benefits each sub-population on diversity every iteration.

*Evaluate* again until each chromosome  $G$  has obtained its corresponding result  $R = \{r_k | 1 \leq k \leq |R|\}$ .

Fig. 3.4 shows the flow of the parallel genetic evolution from random performance space matrix ( $Z_{K \times S}$ ) to convergence. Each sub-population experiences a evolution with *Evaluate*, *Reproduction*, *Crossover* and *Mutation*. Between algorithm 1 Line 12 to Line 15, parallel genetic algorithm utilizes a fitness function to determine the suitability for each  $G_i$ . In our algorithm, we use the *Heterogeneous Island* architecture [11]. According to our requirement, we tend to specialize the particular spec, such as voltage gain ( $A_v$ ). A set for fitness function is given,  $Fitness = \{fitness_i | 1 \leq i \leq k\}$  as kind of objective function to determine how important each individual

is with the *Evaluate* result. Each sub-population  $P_i$  applies one particular fitness function which is related to the required performance result  $R_k$  of *Evaluate* value. Therefore, the fitness function determines the qualified individuals to be preserved to the crossover pool in a weighted ratio, and the others should be extinct. In *Crossover* of algorithm 1 Line 14, the *Crossover* step selects each two genes  $G_i$  and  $G_j$ , where  $\{G_i, G_j \in Pool; 1 < i < j < N\}$  for population. Since each individual has  $K$  types of spec, these two individuals exchange  $c$  specs and reserve  $K - c$  specs with each other. In the end of parallel sub-level evolution, the *Mutation* step picks up one individual with mutation rate and then replaces one gene value by one slot of  $Z_K$ .

### 3.2.5 Migration

For each sub-population  $P_i$ , we perform *Mutation* and obtain an updated  $P_i$ . *Migration* is summarized as shown in Algorithm 2 aims to exchange individuals in the population network shown in the bottom of Fig. 2.1. We use the complete network migrated topology illustrated in Fig.3.2 (a). Therefore, each  $P_i$  should operate *Migration* with the others. From algorithm 2 Line 5 to Line 7, each *Migration* between  $P_i$  and  $P_j$ ,  $P_i$  exchanges its best individual with respect to  $fitness_j$ , and  $P_j$  replaces its worst individual with respect to own  $fitness_j$  with migration rate  $M_P$  in algorithm 2 Line 4. Different sub-population owns its fitness function and uses the particular migration strategy can close to natural evolution. An example of a migration operation between two sub-populations is shown in Fig.3.5. After the *Migration* is executed, the composition of each  $P_i$  is updated with higher diversity.

### 3.2.6 Merge

After all, while the termination condition meets, all the sub-populations are generated. These sub-populations not only have good solution with respect to all

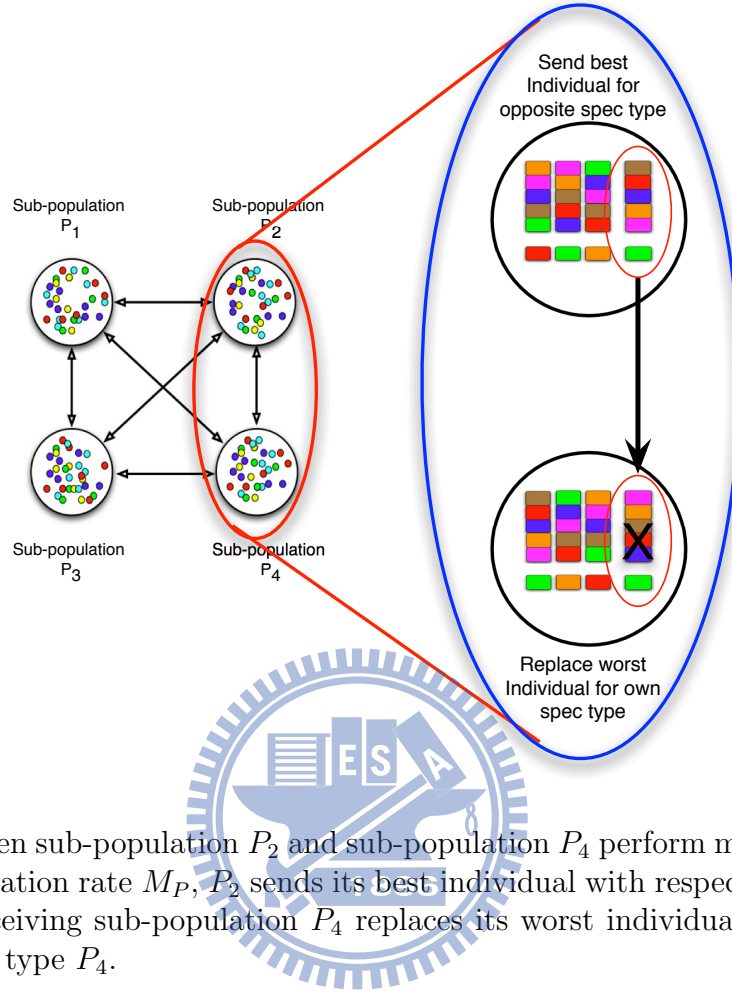


Figure 3.5: When sub-population  $P_2$  and sub-population  $P_4$  perform migration operation with migration rate  $M_P$ ,  $P_2$  sends its best individual with respect to spec type  $P_4$ , and the receiving sub-population  $P_4$  replaces its worst individual with respect to its own spec type  $P_4$ .

spec, but also enhance diversity. All individuals in each sub-population merge to a main population one after another. After merge, main population has all the individuals generated from evolution.

### 3.3 Probabilistic Stochastic Circuit Simulation

This step performs a full-circuit stochastic simulation. After design re-targeting step, a set of overall device level variable values are collected,  $\Psi = \{V_j^D | 1 \leq j \leq N_{VD}\}$  from populations.  $N_{VD}$  collects the number of all device variables type. Since these design variables are transformed from performance populations by parallel genetic algorithm, each population of performance metrics directly projects to a set



of variable distribution. A basic multi-objective SA based stochastic simulation is implemented in Meng et al.[16]. After performance exploration, a Pareto front and a Pareto point can be founded and regard the Pareto point as a good initial solution fed into multi-objective SA based simulation. In [16], each swap is according to a uniform manufacturable range with respect to each device variable. Since the range of each device variable is wide, the solution set becomes bigger and bigger. Even if multi-objective SA has a good initial solution, the full range stochastic simulation leads enormous computing time to converge.

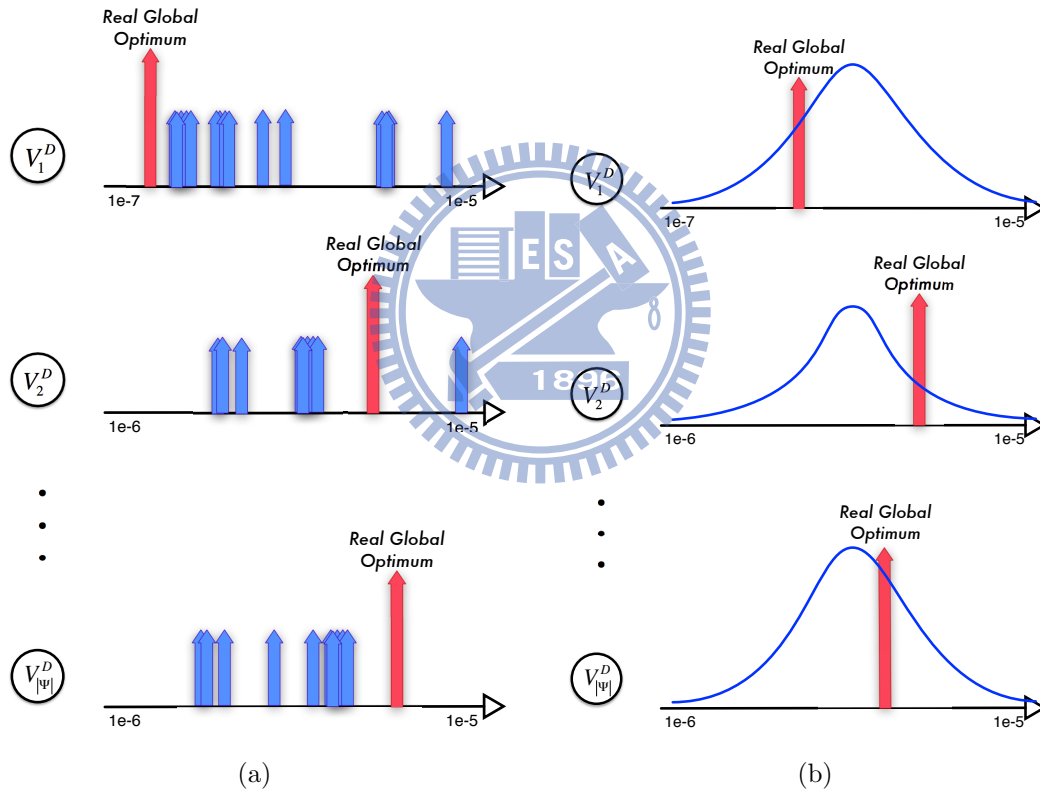


Figure 3.6: Compared with different distribution for stochastic simulation. (a) Population samplings for each device level variable from population and the global optimum location. (b) Normal distributions for each device level variable after normalize and the global optimum location.

After performing multi-objective genetic exploration, an evolved population will be generated. These sampling sets from population form distributions with respect to each device variable. Different from the uniform manufacturable range, these

distributions have several valleys. Theoretically, the global optimum should be located among these valleys with respect to each device variable. However, every performance result should be quite close to real circuit simulation results such as SPICE. In other words, every prediction result from genetic based performance exploration projects to a device variable set, but every variable set feeds into circuit simulator to get real simulation result just close to the prediction result from genetic based exploration. As illustrated in Fig. 3.6 (a), when swapping according to population sampling in SA, these valleys may not determine the real global optimum result accurately. In our methodology, a normal distribution for each device variable generated is fed into SA. Fig. 3.6 (b) shows that based on a probabilistic mathematical model, the nearest point to the mean is not only quite close to global optimum but the normal distribution PDF also ensures the probability of every point in feasible manufacturable range. The PDF also describes the population sampling distribution. In the ends of normal distribution, the probability becomes smaller and smaller. Thus, the nearby point of the ends of normal distribution means bad solution with respect to the population after performing genetic exploration. Since all samplings have equal probability in uniform distribution, SA based simulation spends more time on bad solution set.

After performance exploration stage, given the variable sets, normalized distributions for each design variable are generated by calculating the mean values and standard deviations. In other words, it represents the probability distribution function (PDF) for that device-level variable. The Box-Muller method [2] uses two independent random numbers, then generates two standard normal distribution variables. As Eq.(3.3) illustrated, two independent random numbers  $U$  and  $V$  distributed uniformly on  $(0,1)$ , then the two standard normal distribution variables  $X$  and  $Y$  are generated. Furthermore, two normal distribution variables  $X'$  and  $Y'$  are transformed,  $\mu$  is the mean and  $\sigma^2$  is the variance. A set of PDFs is generated for each

device-level variable is generated after divide all of samplings into several intervals and calculate the number of interval sampling to get the pdf from  $V_{max}^D$  to  $V_{min}^D$ .

$$PDF = \{pdf_k | 1 \leq k \leq |\Psi|\}.$$

$$\begin{aligned} & \exists U \sim \mathcal{U}(0, 1) \text{ and } V \sim \mathcal{U}(0, 1) \\ & \Rightarrow \begin{cases} X = \sqrt{-2\ln U} \cos(2\pi V) \\ Y = \sqrt{-2\ln U} \sin(2\pi V) \end{cases} \quad X, Y \sim \mathcal{N}(0, 1) \\ & \text{Extend :} \\ & \begin{cases} X' = \sqrt{-2\ln U} \cos(2\pi V) \times \sigma + \mu \\ Y' = \sqrt{-2\ln U} \sin(2\pi V) \times \sigma + \mu \end{cases} \quad X, Y \sim \mathcal{N}(\mu, \sigma^2) \end{aligned} \quad (3.3)$$

---

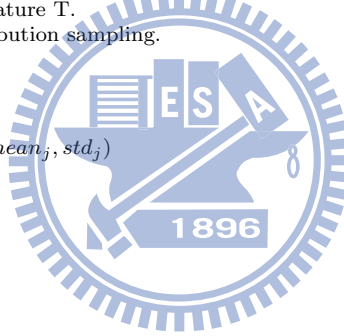
**Algorithm 3** MOSA( $T, \Psi, r, N_s$ )

---

$T$ : A initial temperature,  $T > 0$ .  
 $\Psi$ : A set of overall device variable values from main population.  
**Input:**  $r$ : A factor to reduce temperature  $T$ .  
 $N_s$ : Number of normal distribution sampling.

- 1: **for**  $j = 1 \rightarrow |\Psi|$  **do**
- 2:    $mean_j = CalculateMean(V_j^D)$
- 3:    $std_j = CalculateVariance(V_j^D)$
- 4:    $pdf_j = normalPDFGenerator(mean_j, std_j)$
- 5: **end for**
- 6:  $R = Evaluate(mean)$
- 7: **while**  $T$  not yet frozen **do**
- 8:   **for**  $j = 1 \rightarrow |\Psi|$  **do**
- 9:      $S'_j = Swap(pdf_j)$
- 10:   **end for**
- 11:    $R' = Evaluate(S')$
- 12:   **if**  $R' < R$  **then**
- 13:      $R' \leftarrow R$
- 14:   **else**
- 15:      $R' \leftarrow R$  with probability  $\min(1, \exp\{-\frac{\delta E(R', R)}{T}\})$
- 16:   **end if**
- 17:    $T \leftarrow rT$
- 18: **end while**
- 19: **return**  $R$

---



We assign maximum and minimum values for each  $V^D$  from technology design rules, and apply stochastic circuit simulation among these variable boundaries with an multi-objective SA based search. Our multi-objective SA is summarized as shown in Algorithm 3. Algorithm 3 Line 1 to Line 4 illustrate the PDF generates process for each device variable and the sub-function *normalPDFGenerator* using Box-Muller method is shown in Algorithm 4. After evaluation via circuit simulator to get initial solution  $R$ , Line 7 to Line 18 in Algorithm 3 is the main function of MOSA. Each normal distribution consist of sampling with respect to each device

---

**Algorithm 4** NORMALPDFGENERATOR( $M, Std, N_s$ )

---

$M$ : Mean of population sampling.  
 $Std$ : Standard deviation of population sampling.  
**Input:**  $N_{total}$ : Number of Box-Muller method sampling.  
 $N_p$ : Number of division of the manufacturable range

```
1: for  $j = 1 \rightarrow N_{total}$  do
2:    $U = rand(0,1)$ 
3:    $V = rand(0,1)$ 
4:    $X = \sqrt{-2\ln U} \cos(2\pi V) \times Std + M$ 
5:    $Y = \sqrt{-2\ln U} \sin(2\pi V) \times Std + M$ 
6: end for
7:  $Z \leftarrow X, Y$ 
8: for  $i = 1 \rightarrow N_p$  do
9:    $pdf_i \leftarrow \frac{NumberOfSampling(Z,i,i-1)}{N_{total}}$ 
10: end for
11: return pdf
```

---

variable via Box-Muller method. The *Swap* function selects a point between  $V_{max}^D$  to  $V_{min}^D$  randomly, then determined according to the *pdf* of such  $V_D$ . Fig. 3.7 shows an example of *Swap* operation step by step. After *Swap* operation in Algorithm 4 Line 9, the set of device variables  $S'$  is fed into circuit simulator to get result  $R'$ . If  $R'$  dominates  $R$ , the new solution  $R'$  accepted and the dominate relation defined in Definition 1. Engrand et al. [17] provide a composite function method to get the combined cost of multi-objective values in Definition 2. The accept method when  $R'$  non-dominate  $R$  is shown in algorithm 3 Line 15 and then reduce the temperature  $T$  by a factor. The *MOSA* stopped when the temperature reach frozen then return a Pareto point result  $R$ .

**Definition 1**  $x \prec y$  called  $x$  dominates  $y$  iff  $E_i(x) \leq E_i(y)$ , for  $i=1, \dots, K$  and  $E_j(x) \leq E_j(y)$  for least one objective function  $E_j$ .

**Definition 2** The composite objective function is defined,  $E = \sum_{n=1}^K \ln(w_n f_n(x))$ , where  $f_1, \dots, f_K$  are  $K$  objectives to be optimized and  $w_i$  is normalized weighting value corresponding to each objective result, for  $i=1, \dots, k$ .

A hierarchical synthesis for circuit sizing strategy is accomplished after the optimal solution is converged when stochastic simulator or the termination requirement

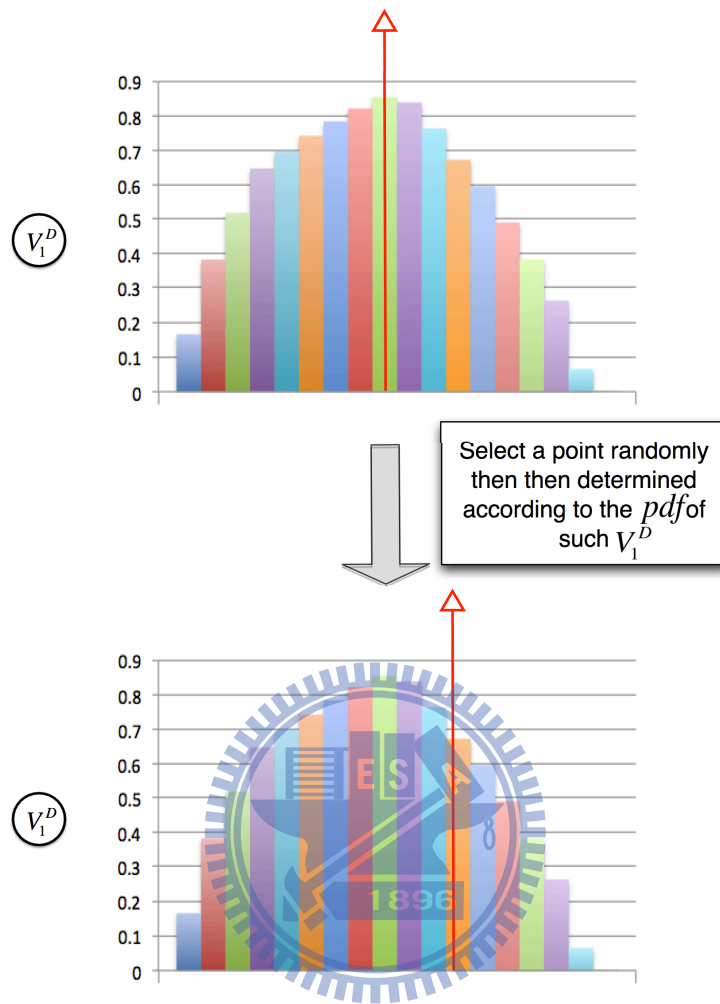


Figure 3.7: Illustration an example of swap function in MOSA step by step.

is met. Based on a probabilistic stochastic fine-tuning, the convergence time less than non-probabilistic approach because the times of sweep are become fewer.

# Chapter 4

## Experimental Results

Table 4.1: Device statistics of RFDA and OpAmp

circuits	Device Number				
	MOS	Capacitor	Resistor	Inductor	Total
RFDA	12	30	0	30	72
Spec	$A_v(\frac{v}{v})$	$P_{dc}(\mu W)$	$P_{out}(mW)$	$F_{cent}(GHz)$	$BW(GHz)$
	$\geq 5$	$\leq 0.5$	$\geq 2$	$\geq 5$	$\geq 10$
Op-Amp	18	1	0	0	19
Spec	$A_v(\frac{v}{v})$	$P_{dc}(\mu W)$	$P_{out}(\mu W)$	$BW(MHz)$	Phase Margin
	$\geq 40$	$\leq 100$	$\geq 0.1$	$\geq 60$	$\geq 50$

As demonstration of our methodology, we obtain a radio-frequency distributed amplifier(RFDA) in [16] and another folded cascode operational amplifier to be synthesized automatically by our framework through three technology processes: umc 65nm, umc 90nm and tsmc 90nm. Table 4.1 shows the statistics of the two circuits, including the components of each and the original performance specifications. Our methodology is implemented by GCC version 4.3.4, Matlab R2011a, and the cvx optimization package. This algorithm is run on openMP and Matlab distributed computing toolbox. Since we rewrite the Matlab cvx package with our design equations of RFDA and OP-Amp into our C++ based parallel genetic algorithm, the interpretation is accomplished by Matlab Runtime Compiler 4.15 on Intel Xeon E5620 2.4GHz with 72GB memory.

Table 4.2: The population results and genetic exploration runtime for RFDA and Op-Amp circuit with umc 65nm, umc 90nm and tsmc 90nm technologies on GA<sub>256</sub>, PGA<sub>120</sub> and PGA<sub>256</sub> based performance exploration.

RFDA	Algorithm	Runtime(s)	$A_v(\frac{v}{v})$	$P_{dc}(\mu W)$	$P_{out}(mW)$	$F_{cent}(GHz)$	$BW(GHz)$
umc 65nm	GA <sub>256</sub>	4122	4 - 5.2143	0.368 - 0.997	20.9 - 21.2	11.0 - 13.56	18.92 - 24.10
	PGA <sub>120</sub>	3416	4 - 5.2143	0.354 - 0.990	20.9 - 21.2	11.04 - 13.56	19.07 - 24.10
	PGA <sub>256</sub>	8053	4 - 5.2298	0.002 - 1.019	20.9 - 27.8	11.12 - 13.52	19.23 - 23.94
Op-Amp	Algorithm	Runtime(s)	$A_v(\frac{v}{v})$	$P_{dc}(\mu W)$	$P_{out}(\mu W)$	$F_{cent}$	$BW$
umc 65nm	GA <sub>256</sub>	5958	10 - 65.71	2.24 - 33	0.90 - 1.53	4183 - 5461	5019 - 6553
	PGA <sub>120</sub>	1855	10 - 65.71	2.24 - 33	0.91 - 1.53	4183 - 5461	5019 - 6553
	PGA <sub>256</sub>	7992	10 - 65.71	2.24 - 33	0.90 - 1.53	4183 - 5461	5019 - 6553
umc 90nm	GA <sub>256</sub>	5836	10 - 400	1.28 - 30	0.31 - 0.88	2442 - 4159	2930 - 4990
	PGA <sub>120</sub>	1813	10 - 400	1.28 - 30	0.35 - 0.89	2605 - 4159	3126 - 4990
	PGA <sub>256</sub>	4037	10 - 400	1.28 - 30	0.25 - 0.89	2112 - 4158	2654 - 4990
tsmc 90nm	GA <sub>256</sub>	5918	10 - 74.29	0.05 - 29.7	0.12 - 1.03	1323 - 4535	1215 - 5370
	PGA <sub>120</sub>	2805	10 - 74.29	0.14 - 28.68	0.18 - 1.03	1158 - 4431	1390 - 5317
	PGA <sub>256</sub>	7108	10 - 74.29	0.13 - 29.69	0.17 - 0.96	1144 - 4311	1373 - 5174

## 4.1 Performance Exploration Stage

Here we implement genetic algorithm at performance exploration stage as follows:

1. **GA<sub>256</sub>**, a master-slave genetic algorithm based performance exploration with total population size of 256 and 8 number of threads.
2. **PGA<sub>120</sub>**, a multi-objective parallel genetic algorithm based performance exploration with total population size of 120 and 5 number of threads.
3. **PGA<sub>256</sub>**, a multi-objective parallel genetic algorithm based performance exploration with total population size of 256 and 5 number of threads.

Fig.4.1, Fig.4.2, Fig.4.3 and Fig.4.4 are illustrations of population results distribution after genetic exploration. When implement with PGA or increase PGA population size, the PDC point distribution will gradually decrease. Table 4.2 and Table 4.3 show each of population result ranges and best performance result in population set respectively. The population result ranges in umc 65nm RFDA and umc 90nm Op-Amp, all ranges on GAP<sub>120</sub> and GAP<sub>256</sub> have better quality than the GA<sub>256</sub>. In tsmc 90nm Op-Amp, GAP<sub>120</sub> has better performance in  $BW$  and  $P_{out}$ . For the best result in population, in RFDA and Op-Amp tsmc 90nm, GAP<sub>120</sub> has

Table 4.3: Best performance of population results for RFDA and Op-Amp circuit with umc 65nm, umc 90nm and tsmc 90nm technologies on GA<sub>256</sub>, PGA<sub>120</sub> and PGA<sub>256</sub> based performance exploration.

RFDA	Algorithm	$A_v(\frac{v}{v})$	$P_{dc}(\mu W)$	$P_{out}(mW)$	$F_{cent}(GHz)$	$BW(GHz)$
umc 65nm	GA <sub>256</sub>	5.2143	0.7218	21.2	11.2	19.2
	PGA <sub>120</sub>	5.2143	0.7465	21.2	11.3	20.0
	PGA <sub>256</sub>	5.2285	0.002	26.9	13.5	23.0
Op-Amp	Algorithm	$A_v(\frac{v}{v})$	$P_{dc}(\mu W)$	$P_{out}(\mu W)$	$F_{cent}$	$BW$
umc 65nm	GA <sub>256</sub>	65.71	33	0.90	4183	5019
	PGA <sub>120</sub>	65.71	33	0.90	4180	5020
	PGA <sub>256</sub>	65.71	33	0.90	4183	5109
umc 90nm	GA <sub>256</sub>	65.71	7.09	0.57	3338	4006
	PGA <sub>120</sub>	65.71	7.03	0.61	3455	4146
	PGA <sub>256</sub>	65.71	7.09	0.57	3338	4006
tsmc 90nm	GA <sub>256</sub>	35.71	7.6	0.587	3387	4064
	PGA <sub>120</sub>	48.57	12.18	0.54	3244	3892
	PGA <sub>256</sub>	35.71	8.81	0.68	3644	4373

better performance result than GA<sub>256</sub>. Big population size and multi-objective GA implementation lead to good results. In general, PGA based performance exploration and population size increase lead to better quality in performance result ranges and performance result.

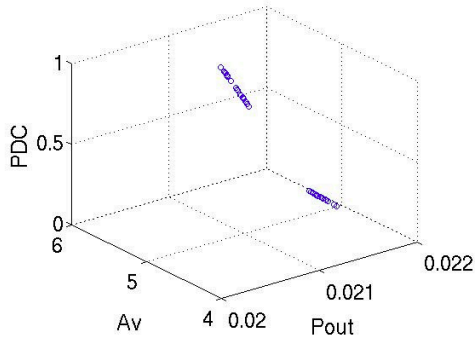
Table 4.4 also shows the genetic exploration runtime. Since genetic exploration spend more time on evaluate and migration operation in parallel GA, master-slave architecture GA has better efficiency than parallel GA exploration in same population size.

## 4.2 Stochastic Fine-tuning Stage

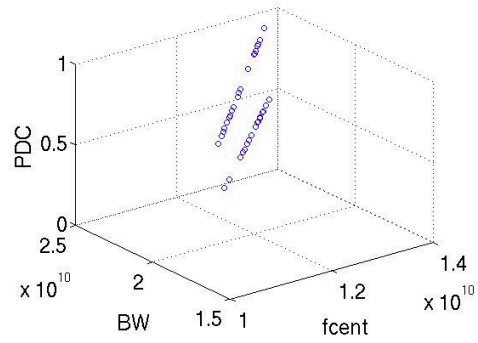
Here we implement overall hierarchical mapping frameworks as follows:

1. **Suman et al. in [19]**, a multi-objective simulated annealing method.
2. **Meng et al. in [16]**, an exhaustive performance exploration method with a basic stochastic circuit simulation.
3. **GAP<sub>256</sub>**, a master-slave genetic algorithm based performance exploration with total population size of 256 and 8 number of threads plus probabilistic stochas-

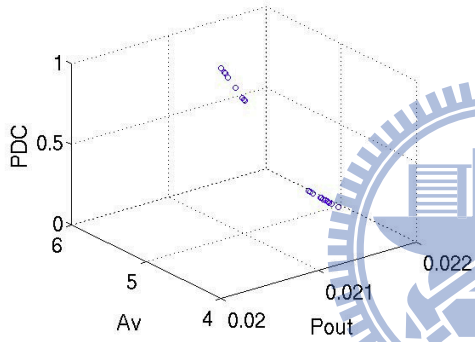




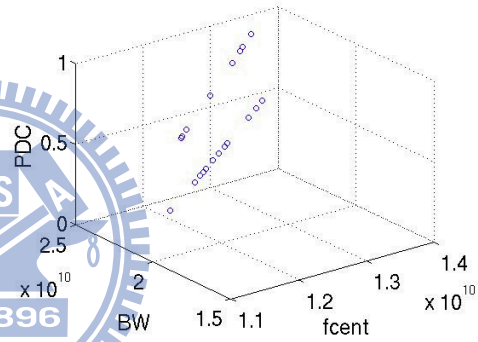
(a) Population distribution via GA<sub>256</sub>



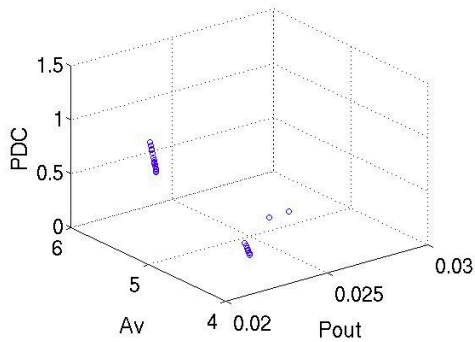
(b) Population distribution via GA<sub>256</sub>



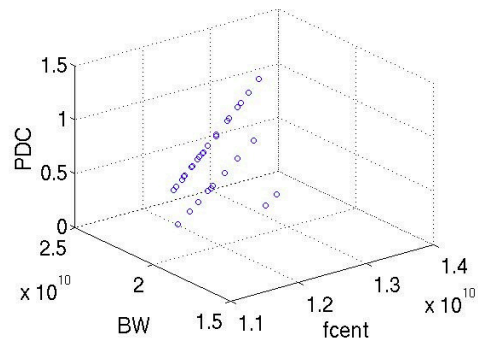
(c) Population distribution via PGA<sub>120</sub>



(d) Population distribution via PGA<sub>120</sub>

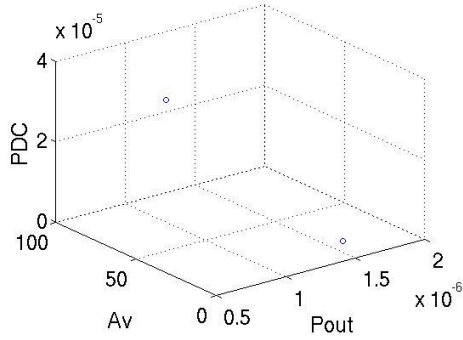


(e) Population distribution via PGA<sub>256</sub>

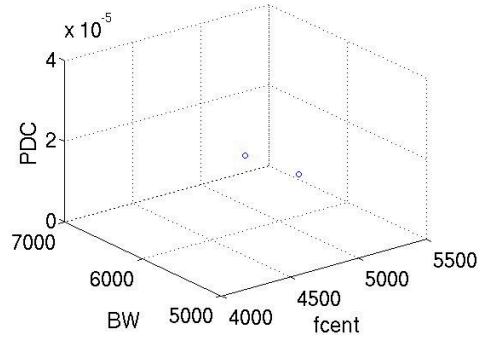


(f) Population distribution via PGA<sub>256</sub>

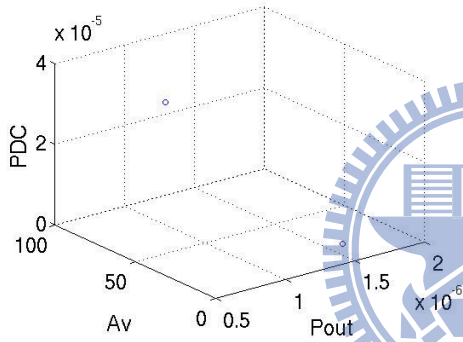
Figure 4.1: Population distribution of the RFDA design instance using UMC65nm after genetic exploration. (a), (c) and (e) are plotted with respect to coordinates of PDC, Pout, and Av. (b), (d) and (f) are plotted with respect to coordinates of PDC, fcent and BW.



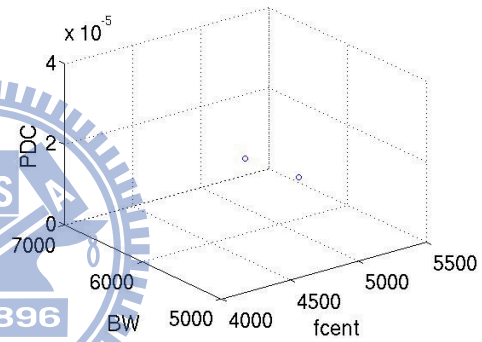
(a) Population distribution via GA<sub>256</sub>



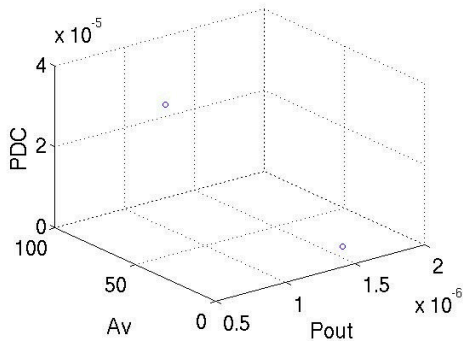
(b) Population distribution via GA<sub>256</sub>



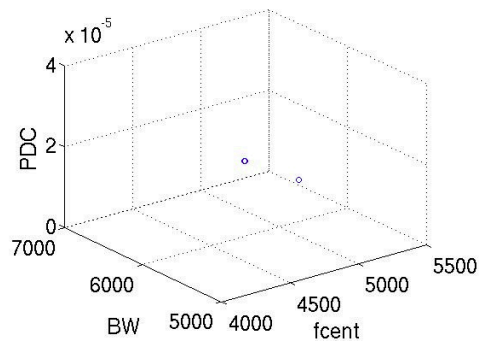
(c) Population distribution via PGA<sub>120</sub>



(d) Population distribution via PGA<sub>120</sub>

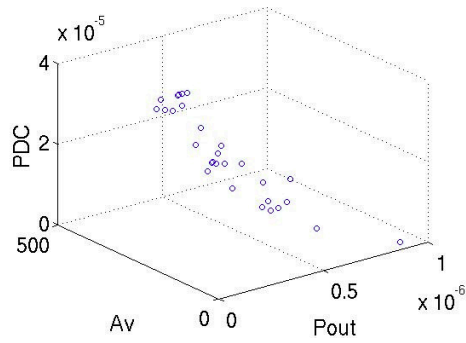


(e) Population distribution via PGA<sub>256</sub>

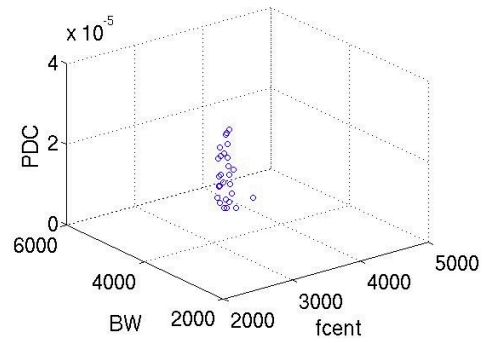


(f) Population distribution via PGA<sub>256</sub>

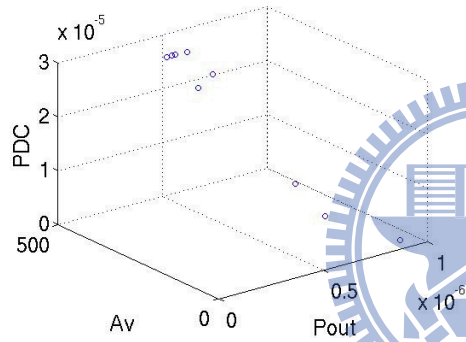
Figure 4.2: Population distribution of the Op-Amp design instance using UMC65nm after genetic exploration. (a), (c) and (e) are plotted with respect to coordinates of PDC, Pout, and Av. (b), (d) and (f) are plotted with respect to coordinates of PDC, fcent and BW.



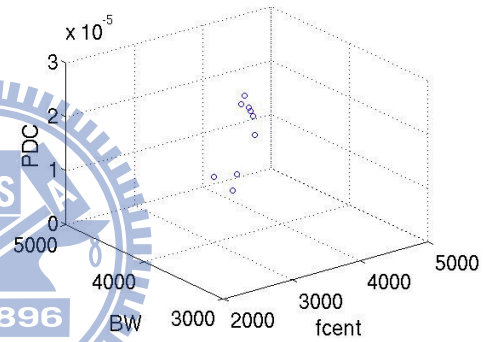
(a) Population distribution via GA<sub>256</sub>



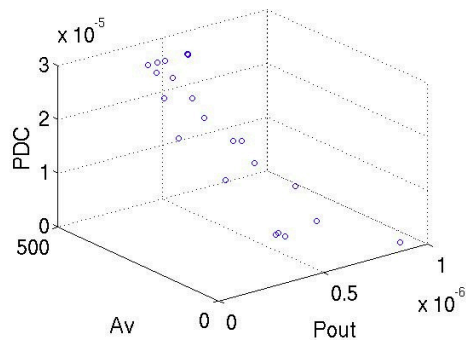
(b) Population distribution via GA<sub>256</sub>



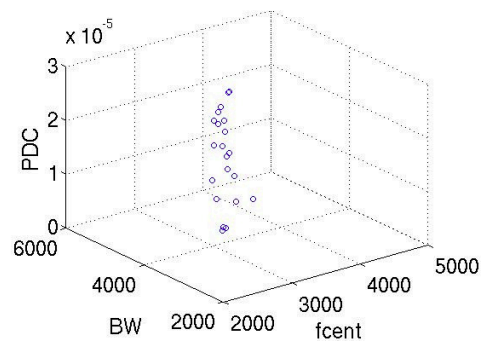
(c) Population distribution via PGA<sub>120</sub>



(d) Population distribution via PGA<sub>120</sub>

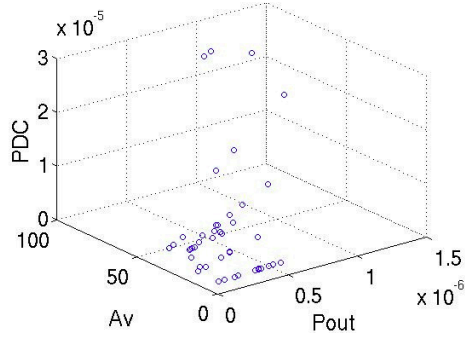


(e) Population distribution via PGA<sub>120</sub>

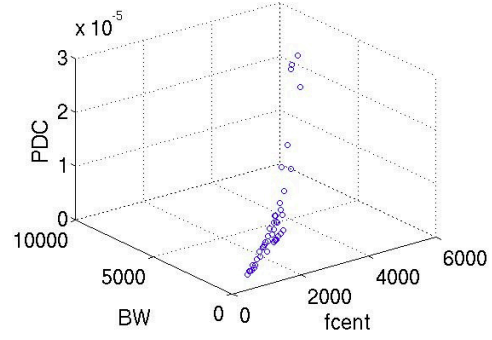


(f) Population distribution via PGA<sub>256</sub>

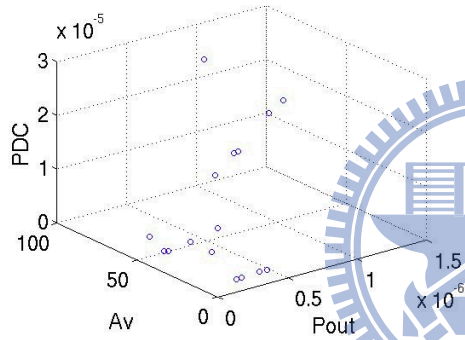
Figure 4.3: Population distribution of the Op-Amp design instance using UMC90nm after genetic exploration. (a), (c) and (e) are plotted with respect to coordinates of PDC, Pout, and Av. (b), (d) and (f) are plotted with respect to coordinates of PDC, fcent and BW



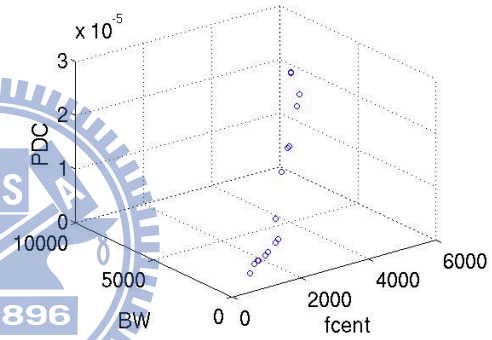
(a) Population distribution via GA<sub>256</sub>



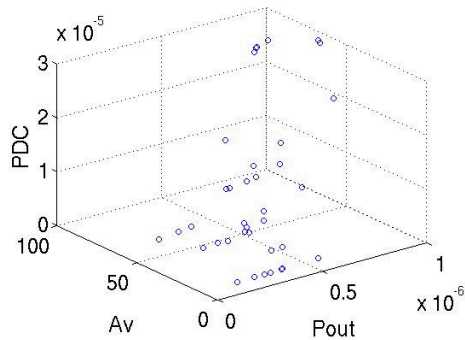
(b) Population distribution via GA<sub>256</sub>



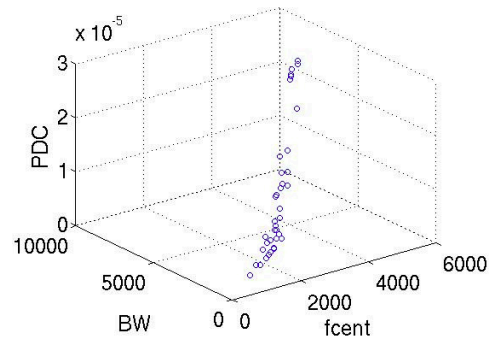
(c) Population distribution via PGA<sub>120</sub>



(d) Population distribution via PGA<sub>120</sub>



(e) Population distribution via PGA<sub>256</sub>



(f) Population distribution via PGA<sub>256</sub>

Figure 4.4: Population distribution of the Op-Amp design instance using TSMC90nm after genetic exploration. (a), (c) and (e) are plotted with respect to coordinates of PDC, Pout, and Av. (b), (d) and (f) are plotted with respect to coordinates of PDC, fcent and BW

tic circuit simulation framework.

4. **PGAP<sub>120</sub>**, a multi-objective parallel genetic algorithm based performance exploration with total population size of 120 and 5 number of threads plus probabilistic stochastic circuit simulation framework.
5. **PGANP<sub>120</sub>**, a multi-objective parallel genetic algorithm based performance exploration with total population size of 120 and 5 number of threads plus basic stochastic circuit simulation framework.
6. **PGAP<sub>256</sub>**, a multi-objective parallel genetic algorithm based performance exploration with total population size of 256 and 5 number of threads plus probabilistic stochastic circuit simulation framework.
7. **PGANP<sub>256</sub>**, a multi-objective parallel genetic algorithm based performance exploration with total population size of 256 and 5 number of threads plus basic stochastic circuit simulation framework.

Table 4.4 shows the comparison among above frameworks for analog circuit synthesis. By definition, parallel GA can search the limitation of performance metrics and also find the performance Pareto-fronts with particular populations. Since genetic algorithm has ability to traverse different combination with crossover and mutation wisely, our approach can collect a bunch of potential spec combinations as a population, and transfer them to re-target back to the desired design variables. Not only obtaining a good initial performance metric population is important, also a design equation which can precisely sketch out the circuit characteristic is necessary. However, we tend to keep the stochastic simulation for a final search, but also using an evolutionary methodology to reduce the convergent resource. As we can see, **GAP<sub>256</sub>** already earns runtime improvement at umc65-RFDA, umc65-OPamp, umc90-OpAmp and tsmc90-Opamp with 398% , 227%, 174% and 224%

than the exhaustive performance exploration method way respectively. Moreover, the PGAP<sub>120</sub> earns even better quality by simultaneously explore the performance space with 617%, 568%, 427% and 419% than [16] respectively. When increase population size and with migration each sub-population in parallel GA based performance exploration, PGAP<sub>120</sub> still earns runtime at umc65-RFDA, umc65-OPamp, umc90-OpAmp and tsmc90-Opamp with 330% , 161%, 190% and 186% than [16]. The runtime report from each GA-based experiment shows the good quality in efficiency.

For the target performance result, in umc 65nm RFDA, all performance requirements have better quality than the exhaustive way and [19]. GAP<sub>120</sub> explores the unprecedented results on  $BW$  and  $P_{out}$ , while PGAP<sub>120</sub> has obvious improvement on  $A_v$  and  $F_{cent}$ . On first umc65 folded cascode Op-Amp, although [16] has good quality on  $A_v$  and  $BW$ , the  $PM$  and  $P_{out}$  are sacrificed. GAP<sub>256</sub> and PGAP<sub>120</sub> have good performance than [19] except  $PM$ . On the other hands, GAP<sub>256</sub> and PGAP<sub>120</sub> come to the quality on each performance target. We can tell that genetic-algorithm based approach can balance the multi-objective optimization via evolution. For umc90-Opamp case, [16] generated the results far from optimal region with  $A_v$  and  $BW$  to earn the better output power  $P_{out}$ . Although [19] has better quality on  $P_{dc}$  and  $P_{out}$ , but  $A_v$ ,  $BW$  and  $PM$  greater than [19]. As well as the tsmc90-Opamp case,  $A_v$ ,  $P_{dc}$  and  $PM$  are poor while such approach searched into optimal region of  $P_{out}$  and  $BW$ . The quality for exhaustive methodology is uncertain and time-consuming. On the contrary, as the lower part of Table 4.4, GAP<sub>256</sub> successfully searches the optimal region of all target performances than [16] and has better performance on  $BW$  and  $PM$  than [19]. Moreover, PGAP<sub>120</sub> reaches the better quality on  $A_v$ ,  $P_{dc}$  and  $P_{out}$  for umc90-Opamp case respectively than [16]. Likewise, PGAP<sub>120</sub> also explores new limitation for  $A_v$ ,  $P_{dc}$  and  $PM$  at tsmc90-OpAmp as different technology implementation. As a result, the exhaustive approach for performance exploration and

MOSA method needs more timing resource to explore the Pareto-fronts but the uncertainty is indispensable. In contrast, the parallel genetic-algorithm based approach for performance exploration with the probabilistic stochastic simulation resolves the problem efficiently and effectively, although the population size of PGAP<sub>120</sub> is less than the population size of GAP<sub>256</sub>.

For the comparison among non-uniform stochastic simulation and uniform distribution based stochastic simulation, in umc 65nm RFDA, no matter whether PGANP<sub>120</sub> and PGANP<sub>256</sub>, all performance have worse quality than PGAP<sub>120</sub> or PGAP<sub>256</sub>. In Op-Amp, all of the non-uniform stochastic simulation have better *BW*, but the *Av*, *Pdc*, *PM* and *P<sub>out</sub>* are sacrificed. In general, the normal distribution based stochastic simulation has better quality than uniform distribution based SA simulation.

Table 4.4 also shows the performance results among different population size with parallel GA implementation. In umc 65nm RFDA, all the performance have obvious improvement except *Av*. On umc 65nm cascade Op-Amp and umc 90nm cacode Op-Amp, although PGAP<sub>120</sub> has good quality on *Av*, *Pdc* and *PM*, the *Pdc* and *P<sub>out</sub>* are sacrificed. In tsmc 90nm cascode Op-Amp, PGAP<sub>256</sub> leads better quality on each performance target except *Av*. As a while, large size of population shows trade-off between effective on performance exploration and computing efficiency.

Table 4.4: The performance exploration results for RFDA and Op-Amp circuit with umc 65nm, umc 90nm and tsmc 90nm technologies on [16], GAP<sub>256</sub>, PGAP<sub>120</sub>, PGANP<sub>120</sub>,PGAP<sub>256</sub> and PGANP<sub>256</sub> framework.

RFDA	Algorithm	Runtime(s)	Improv.(%)	$A_v(\frac{v}{v})$	$P_{dc}(\mu W)$	$P_{out}(mW)$	$F_{cent}(GHz)$	$BW(GHz)$
umc 65nm	[16]	38880	-	6.4322	0.23	2.65	9.85	17.48
	GAP <sub>256</sub>	9756.02	398%	7.38	0.175	23.3	20.9	40.2
	PGAP <sub>120</sub>	6300.15	617%	8.505	0.183	18.8	22.5	40.4
	PGANP <sub>120</sub>	-	-	1.1843	3.88	1.74	8.95	5.0
	PGAP <sub>256</sub>	11772	330%	6.1852	0.141	28.9	24	42.8
	PGANP <sub>256</sub>	-	-	2.2818	11.83	3.43	5.53	10.54
Op-Amp	Algorithm	Runtime(s)	Improv.(%)	$A_v(\frac{v}{v})$	$P_{dc}(\mu W)$	$P_{out}(\mu W)$	$BW(MHz)$	Phase Margin
umc 65nm	[19]	4699	-	42.96	93.75	0.27	97	76.8
	[16]	19432	-	45.73	102	0.21	144	45.7
	GAP <sub>256</sub>	8568	227%	44.88	93.76	0.428	102	67.84
	PGAP <sub>120</sub>	3424	568%	44.17	93.94	0.527	102	67.7
	PGANP <sub>120</sub>	-	-	43.641	88.652	0.263	57	87.5
	PGAP <sub>256</sub>	12051	161%	43.107	105.5	1.159	114.3	58.7
umc 90nm	[19]	6023	-	40.68	90.96	2.64	56	65.56
	[16]	15285	-	33.16	95.6	1.72	78.58	65.872
	GAP <sub>256</sub>	8797	174%	44.247	96.36	0.38	111	74.45
	PGAP <sub>120</sub>	3583.6	427%	44.981	95.11	0.763	110	74.4
	PGANP <sub>120</sub>	-	-	44.075	88.779	0.37	57.7	77.9
	PGAP <sub>256</sub>	8054	190%	41.785	104.16	2.40	145.3	87.7
tsmc 90nm	[19]	17860	-	41.88	89.584	1.39	58	121.5
	[16]	19488	-	38.42	111	1.2	284	50
	GAP <sub>256</sub>	8703	224%	40.46	100.1	0.27	100	82
	PGAP <sub>120</sub>	4651	419%	45.734	97.75	0.22	129	87.4
	PGANP <sub>120</sub>	-	-	39.282	91.1	0.43	64	86.07
	PGAP <sub>256</sub>	10496	186%	40.365	95.5	0.66	114.4	143.56
	PGANP <sub>256</sub>	-	-	28.858	87.21	2.90	28.02	91.17



# Chapter 5

## Conclusion

In this thesis, we have proposed a performance utmost exploration framework for analog synthesis framework with a parallel genetic algorithm based approach to efficiently explore a potential performance space for optimal solution. Unlike exhaustive search the performance space which is time-consuming, this work first transforms the problem set as chromosome and then implements a parallel evolutionary algorithm to resolve multi-objective performance optimization. After a re-targeting transformation between performance and design variables, we also implement a probabilistic stochastic simulation with respect to the design variable distribution. Our methodology also minimizes time to converge the global optima with accuracy. As demonstration for our methodology, a RFDA circuit and an Op-Amp are practiced via 3 different technologies to show that our proposed performance exploration approach and probabilistic stochastic simulation are effective and efficient for analog circuit synthesis.

# Bibliography

- [1] E. Alba and J. M. Troya. A Survey of Parallel Distributed Genetic Algorithms. *Transaction on Complexity*, 4(4):31–52, Mar. 1999.
- [2] G. E. P. Box and M. E. Muller. A Note on the Generation of Random Normal Deviates. *The Annals of Mathematical Statistics*, 29(2):610–611, 1958.
- [3] E. Cant-Paz. A Survey of Parallel Genetic Algorithms. *Transaction on ILLI-GAL report 97003*, 10, May 1997.
- [4] P. Conca, G. Nicosia, G. Stracquadanio, and J. Timmis. Nominal-Yield-Area Tradeoff in Automatic Synthesis of Analog Circuits: A Genetic Programming Approach Using Immune-Inspired Operators. In *Adaptive Hardware and Systems, 2009. AHS 2009. NASA/ESA Conference on*, pages 399–406, 2009.
- [5] W. Daems, G. Gielen, and W. Sansen. An Efficient Optimization-Based Technique to Generate Posynomial Performance Models for Analog Integrated Circuits. In *Design Automatic Conference*, pages 431–436, 2002.
- [6] T. Eeckelaert, W. Daems, G. Gielen, and W. Sansen. Generalized Posynomial Performance Modeling. In *Design, Automation and Test in Europe Conference and Exhibition*, pages 250–255, 2003.
- [7] M. Eick and H. E. Graeb. Mars: Matching-Driven Analog Sizing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 31(8):1145–158, Aug. 2012.

- [8] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.
- [9] J. Kim, J. Lee, L. Vandenberghe, and C.-K. K. Yang. Techniques for Improving the Accuracy of Geometric-Programming Based Analog Circuit Design Optimization. In *International Conference on Computer-Aided Design*, pages 863–870, 2004.
- [10] L. Labrak, T. Tixier, Y. Fellah, and N. Abouchi. A Hybrid Approach for Analog Design Optimisation. In *IEEE Transaction on Midwest Symposium on Circuits and Systems*, pages 718–721, 2007.
- [11] S.-C. Lin, I. Punch, W.F., and E. Goodman. Coarse-Grain Parallel Genetic Algorithms: Categorization and New Approach. In *Sixth IEEE Symposium on Parallel and Distributed Processing, 1994. Proceedings*, pages 28–37, 1994.
- [12] E. Martens and G. Gielen. Top-Down Heterogeneous Synthesis of Analog and Mixed-Signal Systems. In *Proceedings of Design, Automation and Test in Europe*, pages 1–6, 2006.
- [13] P. Maulik, L. Carley, and R. Rutenbar. Integer Programming Based Topology Selection of Cell-Level Analog Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14(4):401–412, Apr. 1995.
- [14] T. McConaghy, T. Eeckelaert, and G. Gielen. Caffeine: Template-Free Symbolic Model Generation of Analog Circuits via Canonical form Functions and Genetic Programming. In *Design, Automation and Test in Europe*, volume 2, pages 1082 – 1087, 2005.

- [15] T. McConaghy, P. Palmers, G. Gielen, and M. Steyaert. Automated Extraction of Expert Knowledge in Analog Topology Selection and Sizing. In *International Conference on Computer-Aided Design*, pages 392–395, 2008.
- [16] K.-H. Meng, P.-C. Pan, and H.-M. Chen. Integrated Hierarchical Synthesis of Analog/RF Circuits with Accurate Performance Mapping. In *International Symposium on Quality Electronic Design*, pages 1–8, 2011.
- [17] E. P. A Multi-objective Approach Based on Simulated Annealing and Its Application to Nuclear Fuel Management. In *International Conference on Nuclear Engineering*, pages 416–423, 1997.
- [18] R. Rutenbar, G. Gielen, and J. Roychowdhury. Hierarchical Modeling, Optimization, and Synthesis for System-Level Analog and Rf Designs. *IEEE Transaction on Proceedings*, 95(3):640–669, March 2007.
- [19] B. Suman. Study of Simulated Annealing Based Algorithms for Multiobjective Optimization of a Constrained Problem. *Transaction on Computers and Chemical Engineering*, 28(9):1849–1871, 2004.
- [20] D. L. Wim Kruiskamp. Darwin: Cmos opamp synthesis by means of a genetic algorithm. In *Design Automation Conference*, volume 2, pages 433–438, 1995.