Bio-Inspired Circuit Sizing and Trimming Methods for Dynamically Reconfigurable Sensor Electronics in Industrial Embedded Systems

Biologisch inspirierte Methoden zur Schaltungsdimensionierung und Justierung für dynamisch rekonfigurierbare Sensorelekronik in industriellen eingebetteten Systemen

vom
Fachbereich Elektro- und Informationstechnik
der Universität Kaiserslautern
zum verleihung des akademischen Grades
Doktor der Ingenierwissenschaften (Dr.-Ing.)
genehmigte Dissertation

von
M. Sc. Peter Messiha Mehanny Tawdross
geb. in Kairo (Ägypten)

D386

Eingereicht am :	24. Oktober 2007
Tag der mündlichen Prüfung:	14. Dezember 2007
Dekan des Fachbereichs:	Prof. Dr.-Ing. Steven Liu

Promotionskommission

Vorsitzender:	Prof. Dr. Remigius Zengerle
Berichterstattende:	Prof. Dr. Andreas König
	Prof. Dr. Nikola Kasabov

Abstract

Analog sensor electronics requires special care during design in order to increase the quality and precision of the signal, and the life time of the product. Nevertheless, it can experience static deviations due to the manufacturing tolerances, and dynamic deviations due to operating in non-ideal environment. Therefore, the advanced applications such as MEMS technology employs calibration loop to deal with the deviations, but unfortunately, it is considered only in the digital domain, which cannot cope with all the analog deviations such as saturation of the analog signal, etc. On the other hand, rapid-prototyping is essential to decrease the development time, and the cost of the products for small quantities. Recently, evolvable hardware has been developed with the motivation to cope with the mentioned sensor electronic problems. However the industrial specifications and requirements are not considered in the hardware learning loop. Indeed, it minimizes the error between the required output and the real output generated due to given test signal. The aim of this thesis is to synthesize the generic organic-computing sensor electronics and return hardware with predictable behavior for embedded system applications that gains the industrial acceptance; therefore, the hardware topology is constrained to the standard hardware topologies, the hardware standard specifications are included in the optimization, and hierarchical optimization are abstracted from the synthesis tools to evolve first the building blocks, then evolve the abstract level that employs these optimized blocks. On the other hand, measuring some of the industrial specifications needs expensive equipments and some others are time consuming which is not fortunate for embedded system applications. Therefore, the novel approach "mixtrinsic multi-objective optimization" is proposed that simulates/estimates the set of the specifications that is hard to be measured due to the cost or time requirements, while it measures intrinsically the set of the specifications that has high sensitivity to deviations. These approaches succeed to optimize the hardware to meet the industrial specifications with low cost measurement setup which is essential for embedded system applications.

Acknowledgements

I would like to acknowledge my supervisor Professor Andreas König for suggestion of the topic, his support, help, and encouragement. His comments, suggestions, and contributions are very valuable to this research.

I would like to thank my colleague Senthil Kumar Lakshmanan for his communication and supporting me with the information about the hardware he is designing, which is employed in this thesis.

I would also like to acknowledge PHYTEC Messtechnik GmbH for partially supporting sub-projects related to this thesis such as patent study, and feasibility investigation of the generic organic-computing sensor electronics.

To my parents

"If I take the wings of the dawn,
and settle in the uttermost parts of the sea;
even there your hand will lead me,
and your right hand will hold me." Psalms(139:9-10)

Contents

Abstract iii

Acknowledgements iv

1 Introduction **1**

1.1 Sensor Electronics Role, Requirements and Design Challenge 1

1.2 State-of-the-Art Evolutionary Electronics 2

1.3 The Goals of the Thesis 3

1.4 Thesis Structure 5

2 Improvement Requirements and Open Issues Sensor Electronics **6**

2.1 Sensor Electronics System 7

2.2 Static and Dynamic Deviations 8

2.3 Self-Calibration Analog to Digital Converters 9

2.4 Current Mode Circuits Building Blocks 16

2.5 Commercial Analog Reconfigurable ICs 21

2.5.1 Field Programmable Analog Arrays 21

2.5.2 Trimmable Hardware 22

2.6 Discussion 22

3 Evolutionary Computation **25**

3.1 Historical Background . . . 25
3.2 Genetic Algorithm . . . 27
3.2.1 Initialization . . . 28
3.2.2 Selection . . . 28
3.2.3 Crossover . . . 28
3.2.4 Mutation . . . 30
3.2.5 Survivor Selection . . . 30
3.2.6 Genetic Algorithm Techniques . . . 30
3.3 Genetic Programming . . . 31
3.4 Particle Swarm Optimization PSO . . . 32
3.4.1 Modifications of Particle Swarm Optimization . . . 34
3.5 Optimization in Dynamic Environment . . . 42
3.6 Proposed Approach: Local Parameters PSO . . . 44
3.6.1 Experimental Setup . . . 45
3.6.2 Benchmark Test Functions . . . 47
3.6.3 Results . . . 50
3.7 Multi-Objective Extension . . . 60
3.7.1 Weighted Aggregating Function Based Optimization . . . 60
3.7.2 Pareto Based Methods . . . 61
3.8 Discussion . . . 61

4 Evolutionary Electronics 63

4.1 Reconfigurable Evolvable Hardware . . . 63
4.1.1 Coarse Granularity . . . 64
4.1.2 Fine Granularity . . . 66
4.2 Evolutionary Computation Adaptation for Evolvable Hardware . . . 72
4.2.1 Evolving the Hardware Topology by Genetic Programming . . . 73
4.2.2 Turtle GA . . . 74

4.3 Extrinsic Evolution . . . 75
4.3.1 Extrinsic Evolution as a Synthesis Tool . . . 75
4.3.2 Extrinsic Evolution for a Target Programmable Hardware . . . 76
4.4 Intrinsic Evolution . . . 79
4.5 Mixtrinsic Evolution . . . 81
4.6 Fault Tolerance and Dynamic Environment . . . 83
4.7 Summary . . . 83

5 Proposed Design Methodology **85**
5.1 Aspired Generic Organic-Computing Sensor System . . . 85
5.2 Extrinsic Multi-Objective Evolution . . . 88
5.3 Intrinsic Multi-Objective Evolution . . . 89
5.4 Mixtrinsic Multi-Objective Evolution . . . 91
5.4.1 Simulated Models . . . 92
5.4.2 Formal Models . . . 93
5.4.3 Estimation Models . . . 94
5.5 Dynamic Environment and Fault Tolerance . . . 96
5.6 Design Flow . . . 98
5.7 Summary . . . 102

6 Experimental Work and Case Studies **103**
6.1 The Reconfiguration Environment . . . 103
6.2 Baseline Hardware . . . 106
6.3 Building Block Level . . . 109
6.3.1 Operational Amplifier . . . 109
6.3.2 CCII . . . 134
6.4 Functional Level . . . 136
6.4.1 Flash ADC . . . 137
6.4.2 Low-Pass Filter . . . 142

7 Conclusion and Summary **146**

7.1 Conclusion 146

7.2 Novel Contributions of the Thesis 147

7.3 Future Work 148

7.4 Kurzfassung in Deutscher Sprache 149

7.4.1 Motivation 149

7.4.2 Beiträge dieser Arbeit 150

Bibliography **155**

List of Figures **172**

List of Tables **180**

List of Procedures **182**

List of Symbols and Abbreviations **183**

CHAPTER 1

Introduction

Although digital electronics has been widely used in many applications as it is easy to design, immune to noises and deviations, etc., analog electronics is essential for interfacing the real world application –even if digital electronics are employed– to communicate with the sensors and convert their signal to the digital domain if it is not processed directly in analog domain. Therefore, the necessity of the analog circuits in industrial application is unavoidable.

Mimicking the biological systems, evolvable hardware has attempted to include the self-x properties (self-calibration, self-maintaining, self-repairing, etc.), and to accomplish rapid prototyping analog hardware. However, it does not concern the industrial requirements, therefore, the returned configuration may not have predictable behavior. Thus, this thesis is aimed to purpose a methodology to evolve the analog hardware for sensor electronics applications in an efficient way to recover from hardware deviations, and to return a hardware with predictable behavior.

This chapter highlights the motivation of the proposed work in this thesis. Therefore, an overview on the requirements of the sensor electronics is provided. Afterwards, the state-of-the-art evolutionary electronics is summarized. Thereafter, the structure of the thesis is provided.

1.1 Sensor Electronics Role, Requirements and Design Challenge

Sensor networks are ubiquitous in the field of modern systems such as automotive, automation, medical engineering, ambient intelligence, etc. Designing the sensor electronics requires special care such as yield optimization, and worst case design in order to reduce the influences of the deviations. However, the nominal operating point is assumed in the design phase, and therefore, the influences of the dynamic deviations are not completely considered. In many applications, trimming in the deployment time is necessary as it is susceptible to manufacturing drift and other static deviations that affect its target specifications. Self-calibration techniques can cope with some of the dynamic deviations. Nevertheless, not all the deviations are recoverable as the trimming is achieved by employing few programmable components, and consider only few of the hardware specifications in the loop.

For example, the rejustors have introduced by Microbridge Technology Corp., which is an analog controlled resistor to cope with the dynamic deviation by a feedback signal, this feedback signal is controlled digitally in many applications. The PGA309 [Ins] is a programmable gain amplifier from Texas Instruments, which employs digital temperature calibration, and autozeroing to cope with dynamic deviations. Thus, the industry attempts to build robust systems, not fault tolerant systems. Robust systems can cope with the expected deviations, while the fault tolerant systems are designed to treat the unexpected variations.

For multi-sensor systems, affording special care in the design phase, trimming in the manufacturing phase, and employing calibration and trimming techniques for each of the sensors increases the sensor electronics cost. In addition, pluging in new sensors requires many modifications in the hardware. Rapid prototyping is an open topic that decreases the cost as the same hardware can be programmed for many sensors and many applications. Therefore, reconfigurable sensor electronics with self-x properties is desirable.

1.2 State-of-the-Art Evolutionary Electronics

Recently, genetic programming and algorithms are adapted for synthesis of the analog hardware. The genetic programming has been employed to design hardware with arbitrary topology [KFHBAK97]. This can yield unpredictable behavior and explosion in the design size as many devices can be included in an ineffective fashion. On the contrary, genetic algorithm has been applied to synthesize standard hardware topologies [ZPV02]. Nevertheless, recovering from static and dynamic deviations can not be considered in these synthesis tools.

On the other hand, analog reconfigurable hardware, –also called evolvable hardware– is aspired to recover from static and dynamic deviations [SKZ+00]. The state-of-the-art evolvable analog hardware utilizes the genetic algorithm to invent arbitrary topology [HHE02, SKZ+00, Tre06]. However, the state-of-the-art evolutionary electronics does not employ dynamic environment suitable approach to cope efficiently with environmental changes. Indeed, it starts from scratch after any environmental change [Tho97a, KZJS00, SKZ01, ZGK+04]. No special technique is mentioned in the state-of-the-art to detect if a hardware deviation has occurred. Instead, the deviations are applied and controlled manually. In the state-of-the-art evolutionary electronics, the number of the required blocks to find a suitable design equivalent to the given specifications is not known [Tho96, KFHBAK97, SKZ+00, Tre06]. Indeed, the maximum number of allowed blocks is determined by trial and error. Thus, the area of the chip to fulfill a certain functionality is not predictable.

In order to recover from the deviations, the evolution of the hardware has to be achieved intrinsically[1]. The open topics that the state-of-the-art evolvable hardware have not considered in order to gain the industrial acceptance and to recover from deviations are summarized as the following:

- The behavior of the employed topology has to be predictable.
- The industrial specifications of the hardware have to be optimized according to the application requirements.

[1] Intrinsic evolution means that the optimization algorithm runs directly on a real hardware.

- Measuring the complete specifications of the hardware requires expensive measuring equipment, which limit the applicability of the approach. Therefore, the state-of-the-art has considered only the relation between the ideal output and real output of the hardware. Measuring the transient and the AC characteristics of the hardware is limited to the conversion speed of the employed ADCs and DACs.
- The calibration loop has to compensate the deviations. As the evolvable hardware mainly measures the current specifications of the system to evaluate the downloaded configuration, any deviation in the measurement equipment propagates to the entire evolvable system. Although this problem is not treated by the state-of-the-art evolvable hardware, the state-of-the-art ADCs offers self-calibration techniques in both digital [LS92, KLB93, FDLH98, WHL04] and analog domain [CGN04, RRS+04]. However, complicate measurement setup requires more care about self-calibration for each of its components

In the highlight of the mentioned problems above, this thesis is proposed to consider the industrial requirements of the generic organic-computing sensor electronics for embedded system applications as described in the next section.

1.3 The Goals of the Thesis

The aim of our group[2] is to build reliable rapid prototyping generic sensor electronics front-end with self-x properties and predictable behavior for embedded systems that gains the industrial acceptance. This thesis is focused on software and algorithmic aspects, and on manipulating the static deviations of system in the deployment phase and the dynamic deviation in the operation phase. On the other hand, a parallel work by MSc. S. Lakshmanan is focused on the hardware aspects of our generic sensor electronics front-end.

The industrial applications require reliable hardware with predictable behavior according to a set of industrial specifications. Arbitrary hardware topologies, and optimizing the system only by minimizing the error between the output and the reference output may return a hardware with unpredictable behavior, thus, it is objectionable from the industrial point of view. In this thesis, in order to return hardware with predictable behavior, the hardware flexibility is constrained to the standard topologies and the hardware standard specifications are optimized, which is a multi-objective optimization criterion as shown in figure 1.1. The values of these specifications are defined according to the application requirements.

The target hardware for this thesis is a reconfigurable analog hardware with programmable structure and dimensions as shown in figure 1.2. An example of the target building blocks is the programmable operational amplifier proposed by Lakshmanan et al. [LK05], where standard topologies are employed with programmable dimensions and continuous passive components are utilized. Current-mode designs –which means that the individual circuit elements interacts by means of currents, not voltages [TLH90]– has proved its ability to work with low voltage supply, high-speed and required small area in the cost of non-linearity [BG04, Kol00]. In addition, the output of many sensors is current output. Thus, the building blocks of the current mode designs are considered in this thesis in order to achieve a general purpose programmable self-calibration environment.

[2] Institute of integrated sensors, TU Kaiserslautern.

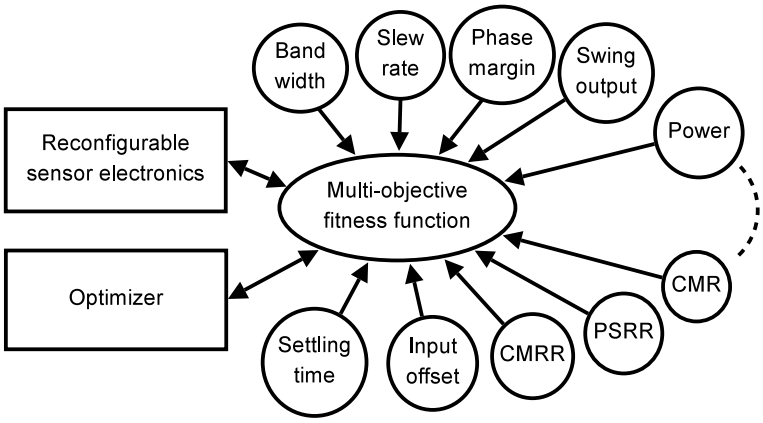


Figure 1.1: The hardware standard specifications are optimized as a multi-objective criterion

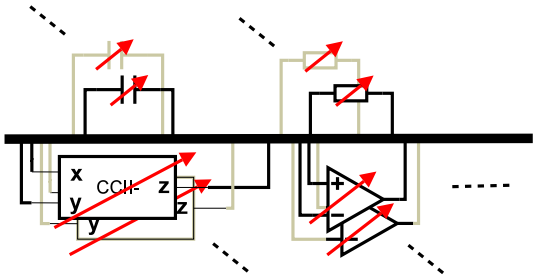


Figure 1.2: Conceptual block diagram of the target hardware.

From the algorithm aspects, the complete system is considered in the-state-of-the-art as a single black-box with a given input-output relation [SKZ+00, Lan05], and the optimization target is to minimize the error between the actual output and the ideal output. The hardware standard specifications and the industrial requirements are not considered, specially in the intrinsic approach[3]. Measuring the complete hardware specifications using the standard methods and equipment is an expensive and time consuming approach. Depending on the target specification, the measurement can be time consuming, requires special expensive wide-band setup, etc. For example, measuring the AC-characteristics of the hardware requires tuning the hardware frequency, which is time consuming process for low frequencies and requires expensive wide-band setup for high frequencies, measuring the transient characteristics requires high speed ADCs, etc. Therefore, techniques for assessing the measurement loop are investigated.

As described above, the state-of-the-art evolutionary electronics starts from scratch after any environmental change, which can result in going from operation mode to trimming mode frequently and for long time-intervals –if no background calibration is utilized– as all the previous knowledge of the evolution is lost. In order to increase the reliability in dynamic environment, dynamic environment optimization approaches are investigated in this thesis, which are capable to detect environmental changes during the evolution and cope with them without starting from scratch after any environmental change for potential improvement in convergence speed and optimality of the solution. In addition, potential improvement to the state-of-the-art evolutionary computation is investigated by benchmark

[3]Parallel to the work pursued in this thesis [TLK05, TK05a, TK05b], Trefzer et al. have included few of the hardware specifications to evolve the hardware extrinsically [TLMS05], but they have allowed the evolution to invent the hardware topology, which returns a hardware with unpredictable behavior.

functions[4].

1.4 Thesis Structure

The structure of the thesis is as the following:

- Chapter 2: An overview of sensor electronics and its requirements is provided. Afterwards, the significant needs to construct generic organic-computing sensor electronics front-end that fulfill those requirements is assessed.
- Chapter 3: The background of evolutionary computation is described, the state-of-the-art particle swarm optimization is illustrated, the dynamic environments suitable approaches are outlined, an improved version of particle swarm optimization is proposed and compared with the state-of-the-art particle swarm optimization using the standard test bench problems in static and dynamic environment, and a brief overview on the multi-objective optimization is provided.
- Chapter 4: Study of the state-of-the-art evolvable hardware, and the missing link to the industrial acceptance in order to build hardware with predictable behavior.
- Chapter 5: A methodology is proposed to evolve the sensor electronics organic-computing front-end in order to return hardware with predictable behavior that recovers from deviations, for embedded system applications, where low cost assessment is required.
- Chapter 6: The experimental work on the selected case-studies is described.
- Chapter 7: The thesis is concluded and summarized.

[4] A benchmark that is commonly used in the comparison of the approaches in order to prove the generality of the improvement.

CHAPTER 2

Improvement Requirements and Open Issues Sensor Electronics

Sensor electronics plays an essential role in the interfacing between digital electronics and real world applications such as analog signal processing and conditioning, industrial process control, motion control, ambient intelligent and biomedical measurement. The micro-electro-mechanical system *(MEMS)* technology allows the design of high performance sensors and actuators, but they require high performance sensor electronics in order to keep their precision. However, sensor electronics is susceptible to static and dynamic deviations. Static deviations are due to tolerance during the manufacturing process and dynamic deviations are due to the environmental changes caused by various operational conditions such as measuring the pressure and flow rate inside a turbine engine for further control. State-of-the-art sensor electronics attempts to prevent the system from static deviations in the design phase by yield optimization and worst case design [dMHBL01], or to employ self-calibration techniques; however, self-calibration techniques considers only few of the system specifications, such as the auto-zeroing technique that reduces the offset. Future semiconductor technologies will allow a wide temperature range, which enhance the MEMS technology to design integrated sensors for high temperature applications. However, it requires more advanced techniques to deal with the deviations that can occur over the whole operation range. For example, diamond semiconductor technology can operate at a temperature range up to $1055^{\circ}C$. Evolvable hardware attempts to eliminate the deviations by extending the hardware flexibility, but the evolvable hardware assessment tools can also experience deviations, especially, the analog-to-digital converter *(ADC)*. Nevertheless, innovative ADCs offer several self-calibration approaches that can cope with the ADC deviations.

In this chapter, an overview of the role of the sensor electronics and the deviations that it can experience is provided. In order to calibrate the sensor electronics, assessment circuits are required to measure the hardware specifications as described in chapter 5, however, they can experience the same deviations. The basic element that is employed in the assessment circuit is the ADCs. Fortunately, the state-of-the-art ADCs considers the self-calibration as described later on in this chapter. The sensor electronics can be either voltage- or current-mode circuits. , an overview on the current mode basic building blocks and their application –which can add flexibility to the target generic reconfigurable sensor electronics– are provided. Afterwards, the commercial flexible hardware is outlined. Finally,

the chapter is summarized and concluded.

2.1 Sensor Electronics System

In figure 2.1, an overview of the sensor system is provided. The physical signals are converted to electrical signals through sensors. Thereafter, appropriate pre-processing is applied to be suitable for sampling and quantization. Afterwards, the signals are digitalized by employing analog to digital converters *(ADCs)* for further digital processing. An example of the pre-processing is filtering the signal with anti-aliasing filter, amplifying it, convert it from current to voltage signal, demodulate it, etc. In some applications, the digital system controls the sensor parameters, such as brightness, speed and resolution of image sensors. The sensor control signal can be processed in analog or digital domain. In control applications, the digital or analog electronics control the system actuators. Advanced MEMS sensors may employ an actuator to control the sensor mechanical settings [OC92, Fra00].

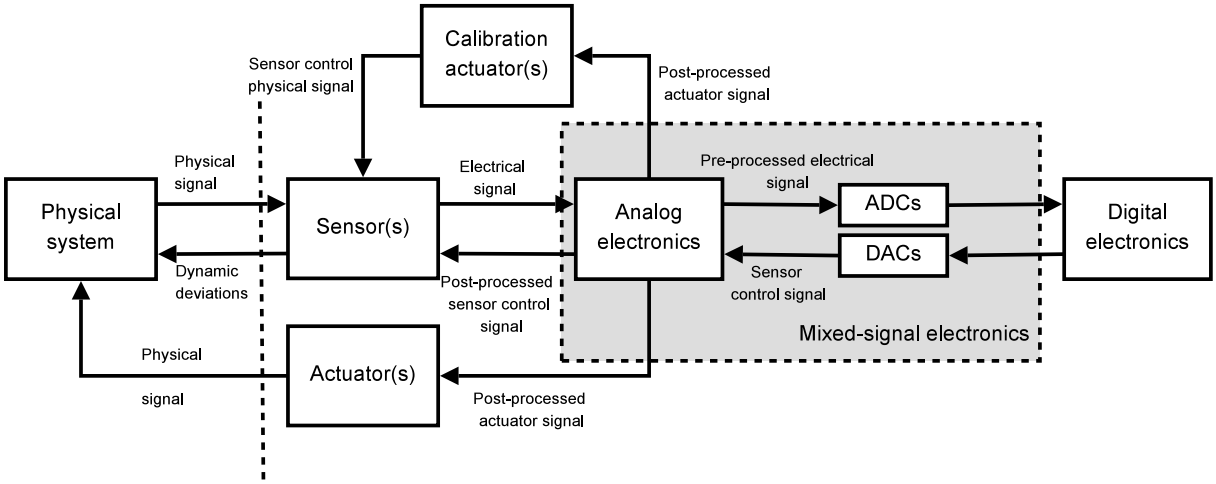


Figure 2.1: Block-diagram of a simplified modern sensor system.

Special care is obliged in designing the analog portion of the sensor electronics system for each sensor and each actuator separately, which increases the design time and cost. The system can deviate statically due to the industrial tolerance, or dynamically due to the operating point conditions as described in section 2.2. In some applications, the sensor signal processing is pre-designed, and integrated with the sensor or actuator if it is allowed by the technology such as MEMS technology. Yield optimization and worst case design in the design phase can minimized the system deviations [AGW93, AGK94, LGA99], but they can not deal with all the deviation problems.

The MEMS micro-stereo-lithography technology allows the design of complicate sensor and actuator shapes which increases the manufacturing cost in order to increase the sensor electronics accuracy. Therefore, its stability and reliability are required even if they increases the die area of the sensor electronics section.

The aim of advancing integration technology is increasing the accuracy, stability, reliability, and consequently, the integration scale, in the cost of the die area. On the other hand, scaling the MEMS

sensor is constrained due to the physical signal requirements. Therefore, the area employed by the MEMS sensor is partially technology independent, while the area of the calibration electronics is technology dependent and thus advancing the integration technology reduce the relative cost of the calibration electronics.

The fourth generation of MEMS sensors integrates the sensors with their analog signal conditioning, analog to digital converter, memory cells for calibration- and temperature compensation data, digital intelligent system to compensate the system deviation. However, the calibration and temperature compensation is deliberated only in the digital domain. Limiting the calibration of the system to the digital domain is not sufficient for many applications. For example, the digital domain calibration is not able to calibrate the system correctly if the analog output is saturated due to deviations such as , e.g., the gain is higher than the required gain, or the common mode range or the swing output voltages are insufficient.

As the MEMS technology integrates the sensor with its electronics in the same chip, its electronics can experience dynamic deviations more than the conventional electronics. For example, measuring the flow rate [KS06] of a hot liquid heats up the whole chip, and result in dynamic deviations. The deviation of the analog domain electronics can be very critical in some application such as medical equipment. For example, it can change the medicament injection rate for a patient due to high offset, or gain error.

2.2 Static and Dynamic Deviations

In figure 2.2 varies phases of deviations on the sensor electronics is shown. According to the application, the specifications of the required sensor electronics are designated. Thereon, yield optimization and worst case design are employed to design hardware with low sensitivity to industrial tolerance. Afterwards, the hardware is manufactured. During the hardware manufacturing, static deviations are introduced due to industrial process tolerance. For example, deviation of the transistor dimensions, thickness of the layers, etc. To eliminate the effect of the static deviations, trimming techniques are employed such as laser trimming, to calibrate the hardware by adjusting the dimensions of some of the components in the deployment phase to minimize the error. The trimming techniques required a loop measurement in order to measure the specifications of the design under calibration, and thus the products are trimmed one by one on a calibration loop. Therefore, trimming is a time consuming and costly process. However, later on, other static deviations due to aging are introduced to the hardware. Ultimately, after trimming the system against static deviations, dynamic deviations occurs in the hardware due to environmental dynamics, e.g., operating at different operation point. Although the conventional electronics operates till $75°C$ for industrial applications, and $125°C$ for military applications, few research labs start to focus on high temperature electronics as sensor electronics can operate at high temperature, e.g., the operational amplifier HT1104 from Honeywell can operates up to $225°C$, which allows the hardware to operate at high environmental dynamic range. The temperature operating range of various semiconductors technologies are summarized in table 2.1. The future technology aims to allow a dynamic operation temperature range up to $1055°C$. Thus, flexibility is required to reconfigure the hardware components to operate at different operating points with similar performance.

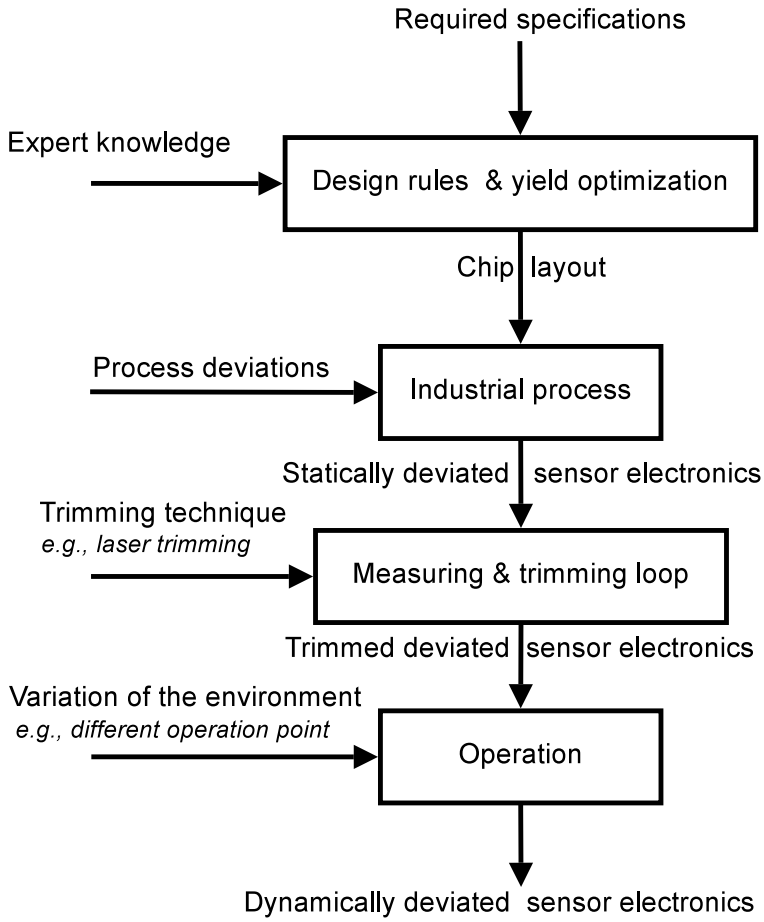


Figure 2.2: Phases of sensor electronics deviations.

Table 2.1: High temperature semiconductor technologies [Goe98].

Technology	Maturity	Temp. range	Source
Si CMOS	Production	-55 to $150°C$	Multiple
SOI (silicon on insulator)	Production	-55 to $300°C$	Honeywell and Allied Signal
E/D GaAs	Production	-55 to $150°C$	Vitesse
Complimentary GaAs	Development	-55 to $350°C$	Honeywell
SiC (silicon carbide)	Early development	-55 to $600°C$	CREE
Diamond	Early development	-55 to $1000°C$	Research labs

2.3 Self-Calibration Analog to Digital Converters

The key of the evolvable hardware is evaluating the specifications of each individual and collate them. Assessing tools such as measurement circuits are employed for the evaluation. As the evolutionary algorithms run on digital hardware platform, the analog signal of the device under test is converted to digital signal by employing ADCs. However, as the ADCs are utilized to measure the system performance, any conversion inaccuracy due to static or dynamic deviations propagates over the entire system, and thus it may evolve to a faulty system instead of the recovering from deviations. For

example, if an ADC has high offset or non-linearity, it will propagate to the whole system. Fortunately, the state-of-the-art ADCs cogitate about the calibration of the ADC.

The employed manufacture technology and the ADC architecture determine the speed, resolution and cost of the ADC. In figure 2.3, the relation between the architecture, resolution and speed is shown.

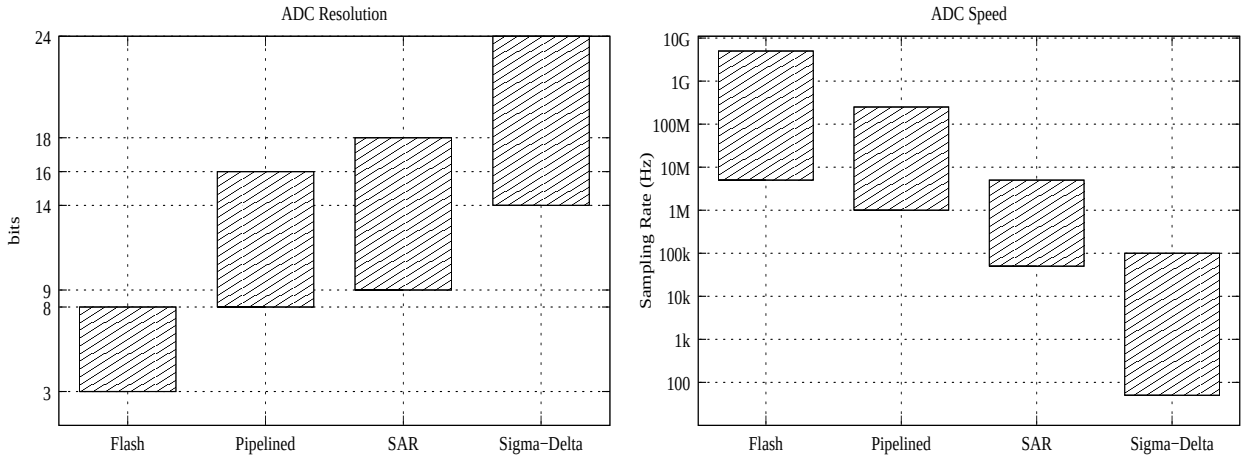


Figure 2.3: Performance of various ADC architectures [Yoo03]. a)ADCs resolution. b) ADCs speed.

The flash ADC converts the signal by comparing the input signal with each of the potential quantization levels, and then encodes the outputs of the comparators to binary code. For a b-bit flash ADC, $2^b - 1$ comparators and $2^b + 1$ passive components are required in addition to the digital encoder. The advantage of the flash ADC is that the conversion time depends only on the propagation delay of the comparators and the digital encoder. On the other hand, the flash ADCs require large die area as the die area increases exponentially with the flash ADC resolution. In addition, the time delay of the encoding and the complexity of the digital encoder are increased by increasing the number of bits. Hence, it is useful for the application that requires fast conversion and low resolution.

Decreasing the cost of the flash ADC can be accomplished by employing multi-step flash ADCs in which, each step holds the signal, converts a certain number of bits, and then amplifies the *residual signal* to the next step. Producing 8-bits in one step requires $2^8 - 1 = 255$ comparators, and $2^8 + 1 = 257$ passive components, while producing them in two steps –each step produces 4 bits– requires $2 \cdot (2^4 - 1) = 30$ comparators, $2 \cdot (2^4 + 1) = 34$ passive components, a 4-bit digital to analog converter (DAC) to convert the digital signal of the first stage to analog signal in order to subtract it from the input signal to obtain the residual signal, and a sample and hold amplifier to amplify the residual signal by $2^4 = 16$ and hold it to the next step. In many arrangements, multiplying digital to analog converter *(MDAC)* is employed as a digital to analog converter, a sample and hold, and an amplifier to obtain the residual signal. The multi-step ADC can employ only one ADC in which, the residual voltage is obtained and hold as a input to the same stage, which decreases the cost.

Pipelined ADC is multi-stage analog to digital converter, in which, each stage produces b_{stage} bits and obtains the residual signal to the next stage. In many implementations, each stage produces one effective bit, and a fraction of bit as redundant information, e.g., produces 1.5-bits; one effective bit, and a half-bit as redundancy. The main advantage of the pipelined ADC is that all the stages convert

different samples simultaneously, which means that it produces one sample each clock, but the output is delayed by $\frac{b}{b_{stage}}$ clocks as the analog input is converted to digital signal after being handled by $\frac{b}{b_{stage}}$ stages. The pipelined ADCs are suitable for moderate and high speed applications as its speed depends only on the conversion speed of one stage. The delay of each stage depends only on the delay of both the *residue amplifier* and the ADC. The disadvantage of the pipelined ADC is that it is very sensitive to deviations as any error in the early stages is magnified and accumulated each next stage.

The successive approximation ADC *(SAR)* employs a counter, a DAC to convert the output of the counter to an analog signal, and a comparator to decide whether to count up if the analog input value is greater than the DAC output, or count down otherwise. If the resolution of the converter is 10 bits, the input changes from zero to full-scale, the counter output must be changed from 0 to 1023 in one sampling period. Thus the SAR ADCs are slow converters. On the other hand, the required die area of the SAR ADCs is relatively small, and the SAR topology allows high resolution as the conversion error is dependent only on the comparator offset and DAC error. Thus it can be employed for low speed but high resolution applications.

The sigma-delta converters employ only one bit ADC (comparator), and integrator to integrate the error due to the one bit conversion. In order to produce the output, many bits are generated, and accumulated to the output. Thus sigma-delta ADC is slow, but its conversion resolution is high and it requires a small die area.

The algorithmic ADC is similar to the pipelined ADC, but it has only a single 1-bit stage. The residual signal is held after each conversion as an input to the same stage.

The static deviations such as components mismatching, and dynamic deviations due to environmental operating point changing cause; offset, gain error, and non-linearity error. The offset error is to have a shift in the entire output signal. The gain error is the difference between the nominal and actual gain after the offset error has been eliminated. Non-linearity error is to have non-linear relation between the analog input signal, and the converted signal. The pipelined, algorithmic, and multi-step ADCs suffer from high non-linearity error due to the gain deviations in each stage.

The state-of-the-art ADCs is concentrated on the pipelined ADC architecture as it balances between the speed, the chip size, the resolution and the consumed power which is fortunate for evolvable hardware application, and its embedded aspects. For example, the flash converters need many comparators and passive components, consumes much power and area, and suffers from input capacitance. The multi-step ADC architecture uses less comparators than the flash ADC, but still it employs a lot of comparator and consumes more power and area with respect to the pipelined architecture. The algorithmic and the SAR ADCs hold the analog signal, and operate in a loop till they find an equivalent digital signal, which is a slow approach, but they can be employed for the applications that require high resolution. On the other hand, the speed and resolution of the state-of-the-art pipelined ADCs are sufficient to cope with moderate and high speed sensor electronics applications. For example, AD9230 from analog devices is a 12-bit pipelined ADC with sampling rate of 250 MSPS, and it consumes 434 mW. Increasing the sampling rate of the ADC can be achieved by adopting interleaved ADCs [BH80]. For example, AD12401 utilizes two interleaved ADCS, consumes 2.2 W with sampling rate of 400 MSPS. However, the offset and the gain difference between the interleaved ADCs cause strong noise as the output error of each of the converters is different from each other, and thus the ADCs must be calibrated to have the same offset, gain, and non-linearity error. The calibration can be done on the background by employing an additional instance of the pipelined ADC [DFLH98]

in which, each of the interleaved ADC instances is disconnected and swapped with a calibrated ADC, thereafter, the uncalibrated ADC is calibrated on the background. A reference ADC and a calibration signal generator are utilized in which, after applying the calibration signal to the ADC under calibration and the reference ADC, the error between both of them is minimized. All the interleaved ADCs are calibrated one by one in order to comprise the same the offset and the gain error as in the reference ADC. However, the reference ADC itself is not calibrated.

In order to minimize the gain error between, the gain of both ADCs is estimated, and the signal of one of them is divided by its ADC gain, and multiplied by the gain of the other ADC in order to obtain the same resultant gain as shown in figure 2.4. For estimating the gain of an ADC, a pseudo-random-number generator is employed to generate uncorrelated signal, then this signal is converted to analog signal by a 1-bit DAC, added to the original input signal, and sampled again by both ADCs, afterwards, the random signal is subtract in the digital domain. The pseudo random number generator produces an uncorrelated signal, and therefore, it can be separated from the input signal and the gain of the ADCs is thus estimated by the ratio between the random signal before and after and before the conversion.

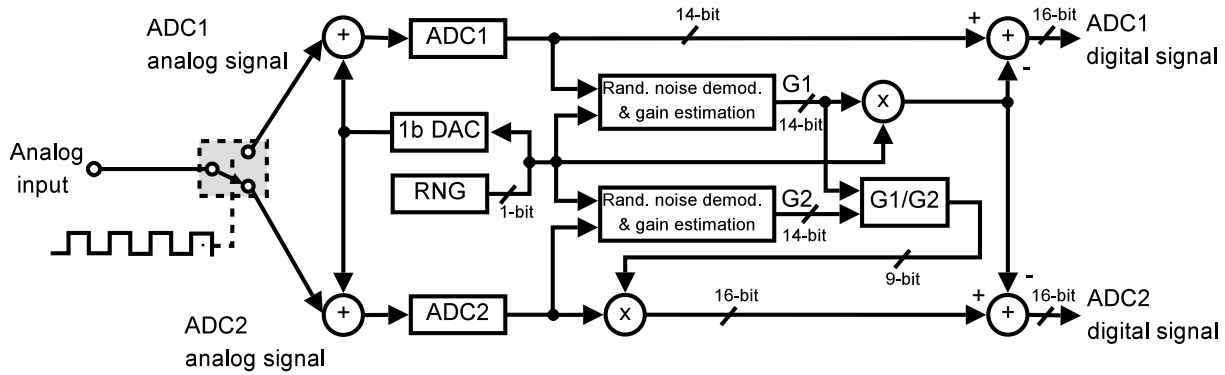


Figure 2.4: Simplified diagram of digital background gain calibration of interleaved ADCs [FDLH98].

After eliminating the gain error, the offset difference between two interleaved ADCs can be minimized in the digital domain by adding it to one of the ADCs [FDLH98] as shown in figure 2.5. The value of offset difference that is added to one of the ADCs is estimated by employing a proportional-controller to minimize the error between the two ADCs.

However, this technique adjusts the offset and the gain of the ADCs to be the same, but it does not calibrate the ADCs them self to reduce the offset the adjusted the gain towards ideal ones. Thus, a precise ADC can deviate to an imprecise one in order to reduce the noise which is generated due to the gain and offset difference. In addition, it assumes that all errors that occurs in the ADC is a single offset and gain error and ignore the conversion non-linearity.

The calibration of the pipelined ADC can be done either in analog (trimming) or in digital domain. The background calibration of the pipelined ADCs can be accomplished by employing an additional stage instance in the pipeline ADC instead of additional complete ADC instance [JW98]. In [ST88], error averaging technique is employed by resorting the amplifier capacitors to minimize the capacitors mismatching. The disadvantage of the error averaging technique is that it needs extra clocks for rearranging the capacitor positions of the stages, and it does not consider the whole system deviations,

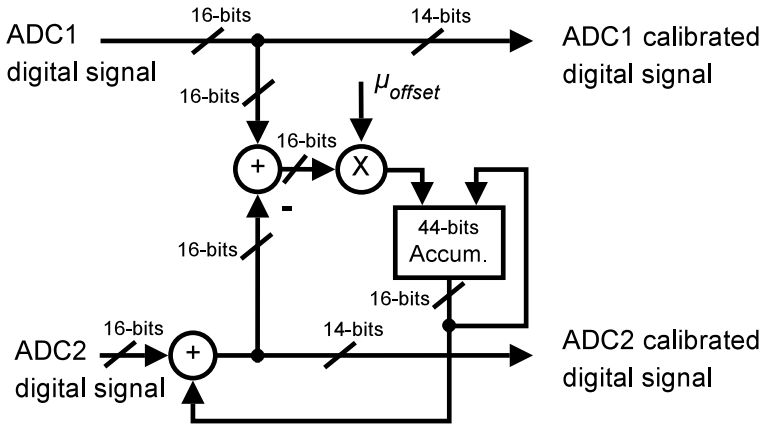


Figure 2.5: Digital background offset calibration of interleaved ADCs [FDLH98].

such as non-ideal open-loop gain. Self-calibration is achieved in analog domain by employing programmable capacitors [LKG91] in the stage amplifier, and trim them in order to get precise gain, while zeroing technique is utilized in each stage to cancel the offset, which has been claimed in [LKG91] to be faster than the error averaging in [ST88].

The disadvantage of analog calibration (trimming) is that it is technology dependent, and requires high power while the newer technologies support lower power. In order to achieve the calibration in the digital domain, redundant bits are required to cope with missing codes. Lee et al. [LS92] introduced digital domain calibration of a two steps multi-step flash ADC by employing an extra-bit each step for the calibration with the architecture shown in figure 2.6.

The ADC in [LS92] employs only one flash ADC that is utilized in both steps. The conversion requires three clocks; in the first clock, the input is sampled in the MDAC capacitor array, and a unity gain configuration is employed in the MDAC in order to have an amplification of one. During the second clock, the sampled data is converted into $b_{step}+1$, which are the *coarse bits*. During the third clock, the residual voltage is converted into $b_{step}+1$ *fine bits*. The residual signal is amplified by $2^{b_{step}}$ in the third clock by employing a capacitor with the value of 2C in the feed back of the MDAC amplifier. The coarse $b_{step}+1$ bits and the fine $b_{step}+1$ bits are utilized to generate and correct $2b_{step}$ bits. Lee Assumed that all the non-linearity due too the deviations appear only on the MDAC, and that the specifications of the flash ADC itself do not deviate. Following this assumption, the error is measured for each quantization level by applying an equivalent digital value to the MDAC, and subtracting it from the converted signal.

The redundancy bits for digital domain calibration of pipeline ADC can be achieved by either employing extra stages and reduce the radix to be less than two [KLB93], or by employing stages that generates redundant bits per stage locally, e.g., each stage generate 1.5 bits, half bit is redundant for error correction, and a bit is the converted data. The architecture of the 1.5 bit/stage pipelined ADC is shown in figure 2.7.

The idea voltage output for one stage is

$$\begin{aligned} V_o &= 2V_{in} - D_s \cdot V_{ref} \\ &= 2\left(V_{in} - D_s \cdot \frac{V_{ref}}{2}\right) \end{aligned} \tag{2.1}$$

where D_s is ± 1 or 0 depending on the input voltage level.

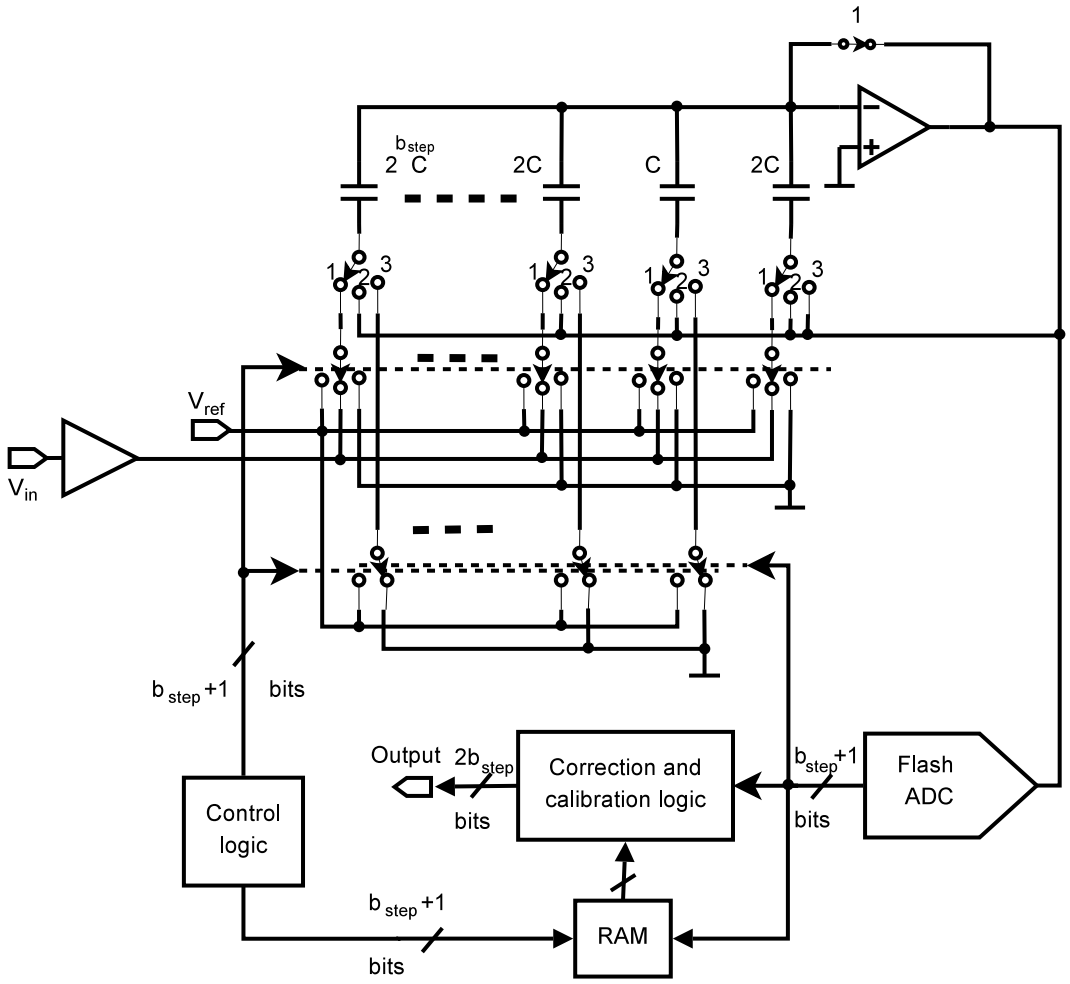


Figure 2.6: Digital domain calibration multi-step flash ADC in [LS92].

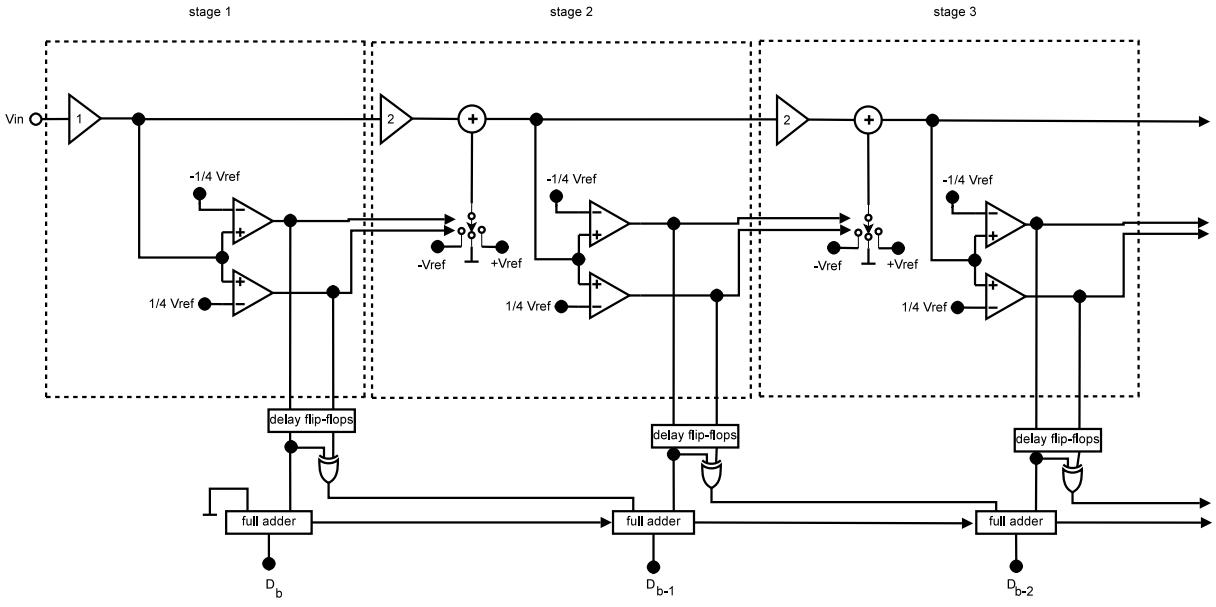


Figure 2.7: Architecture of 1.5 bits/stage pipelined ADC.

Due to static and dynamic deviations, the radix deviates in each stage [LM03], and their offsets are accumulated [WHL04]. The voltage output

$$V_{o_{dev}} = ra \cdot \left(V_{in} - D_s \cdot \frac{V_{ref}}{2} \right) + offset_{stage} \tag{2.2}$$

The offset of all the stages can be subtracted in the digital domain at once instead of the subtracting it in each stage separately [WHL04], and thus

$$V_{o_{dec2}} = ra \cdot \left(V_{in} - D_s \cdot \frac{V_{ref}}{2} \right) \tag{2.3}$$

The digital output due to having different radices in each stage is [LM03]

$$\begin{aligned} D_{out} \quad = \quad & D_b + D_{b-1} \cdot ra_{b-1} + D_{b-2} \cdot ra_{b-1} ra_{b-2} + D_{b-2} \cdot ra_{b-1} \cdot ra_{b-2} \\ & + ... + D1 \cdot ra_{b-1} \cdot ra_{b-2} ra_{b-3} \cdot ra_{b-4} ..... ra_2 \cdot ra_1 \end{aligned} \tag{2.4}$$

This expression can be simplified to [WHL04]

$$D_{out} = \sum_{a=1}^{a=b} 0.5^a \cdot D_a + \sum_{a=1}^{a=b_e} 0.5^a \cdot \varepsilon_a \cdot D_a + offset \tag{2.5}$$

where b_e is the number of the stages that are highly affecting the output because of its deviation; as shown in equation 2.4, the gain of the first stage ra_{b-1} is propagated to all the other stages, and thus a small deviation in it result in a large output error, while ra_1 is not affecting any other stage, therefore, a large deviation in it may not strongly affect the output. Experimentally, Wang et al. have found that $b_e = 5$ is adequate to calibrate 12-bit pipelined ADC, and thus estimating five coefficients and an offset are enough to calibrate 12-bit pipelined ADC.

Most of the calibration schemes are achieved in the foreground during the system power-up or standby mode [LM03]. The disadvantage of the foreground calibration is that it can not recover from dynamic deviations such as changing the operating temperature.

In order to achieve background calibration in the pipelined ADC, several schemes are proposed. The skip and fill scheme [KSB97] occasionally skip a sample from the input and sample a test sample instead in order to estimate the error. The output of the skipped sample is filled digitally by applying non-linear interpolation of the sampled data. The bandwidth has to be limited in which the non-linear interpolation can generate an approximation of the input signal. Correlation based schemes [ML01] modulate the calibration signal with a pseudo-random-number that is uncorrelated to the input signal, add it to the input signal, then demodulate it in the digital domain to estimate the error coefficients. The reported correlation-based schemes are complicate and slow to converge, while it requires a slow but accurate ADC in each stage to extract the pseudo-random signal. Instead of employing a slow but accurate ADC in each stage and pseudo-random-number generator, only one redundant slow but accurate ADC, which has resolution equal to or greater than the target ADC, is employed, e.g., sigma delta [SSB97] or algorithmic ADC [WHL04]. The error is extracted by sampling the input signal by both the redundant and the target ADC and subtract them as shown in figure 2.8. The advantage of employing redundant ADC scheme is that estimating the error does not interrupt the target ADC, its implementation is simple, and no limitation is introduced to the input signal due to employing this approach. Furthermore, the redundant ADC can be calibrated in the foreground.

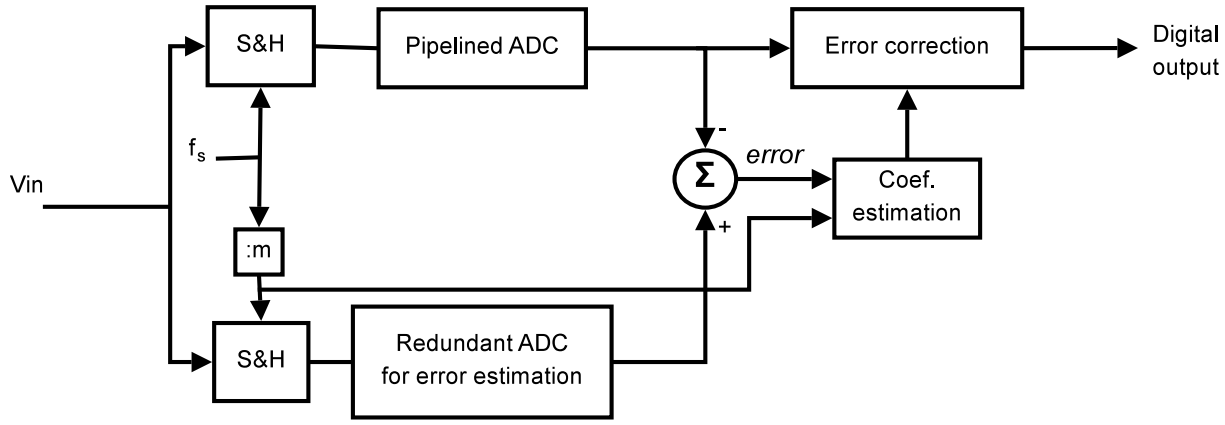


Figure 2.8: Digital background self-calibration of pipelined ADC based on employing redundant ADC.

Table 2.2: The achieved bit resolution after calibrations.

Calibration scheme	ADC topology	Achieved Nr. of bit	Max. INL[1] [LSB]
Digital-domain [FDLH98]	Time-interleaved ADC	10	0.34
Digital-domain [LS92]	Multi-step	11	0.9
Nested digital background [WHL04]	Pipelined	12	0.47
Error-averaging [CGN04]	Pipelined	14	0.54
Capacitor trimming [RRS$^+$04]	Pipelined	14	1
Digital-domain [KLB93]	Pipelined	15	1.25

Calibration of the algorithmic ADC is very simple; the offset is adjusted to precise zero, and the radix to precise two which can be achieved in analog domain [LKG91], or in digital domain. Several approaches to calibrate the algorithmic ADC in digital domain is described in [WHL04]. Calibrating the sigma delta ADC in analog domain [LSF98] can be achieved by employing programmable devices to trim the comparator offset, and the programmable integrator gain. A reference input signal in the feedback circuit is employed for the calibration.

The achieved number of bits by the mentioned calibration scheme regarding the high- and the medium-speed applications is summarized in table 2.2.

2.4 Current Mode Circuits Building Blocks

Many sensors produce current output, such as photo-diode sensors and many pressure sensors. The current output of the sensors can be transformed to voltage, or processed directly in current domain, and converted to digital signal through a current mode ADC. The current mode circuits have the advantage of low power supply, and high speed [BG04, Kol00] in many designs.

In this section, the building blocks of the second generation current conveyor *(CCII)*, current differ-

[1]The integral nonlinearity *(INL)* describes the difference between the actual reference voltages and the ideal reference voltages at all transitions points

encing transconductance amplifier *(CDTA)*, and their applications are described briefly as examples of current mode circuits.

The second generation current conveyor has been introduced by Smith and Sedra in 1970 [Wil90] as a building block that can be used in voltage or current mode applications. In figure 2.9, the second generation current conveyor building block symbol is shown. The operation of the CCII is described

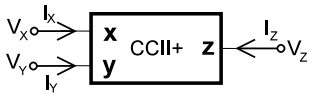


Figure 2.9: Second generation current conveyor symbol.

by equation 2.6.

$$\begin{bmatrix} I_y \\ V_x \\ I_z \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & \mathfrak{s} & 0 \end{bmatrix} \begin{bmatrix} V_y \\ I_x \\ V_z \end{bmatrix} \tag{2.6}$$

where $\mathfrak{s} = 1$ for CCII+, and $\mathfrak{s} = -1$ for CCII-. The voltage at the node x is equal to the voltage the node y; $V_x = V_y$, while the output current at the node z is equal to the input current at the node x for the CCII+; $I_z = I_x$, and is opposite in sign for CCII-; $I_z = -I_x$.

Many applications have been proposed using current conveyors, for example, floating frequency dependent negative resistance *(FDNR)* [Sen84] in figure 2.10, and feedback voltage amplifier [Wil90] in figure 2.11. More description on CCII and its applications are found in [Wil90, HK05]. As the CCII block is sufficient as a basic building block for designing many circuits, it is employed for developing a field programmable analog array [Gau97].

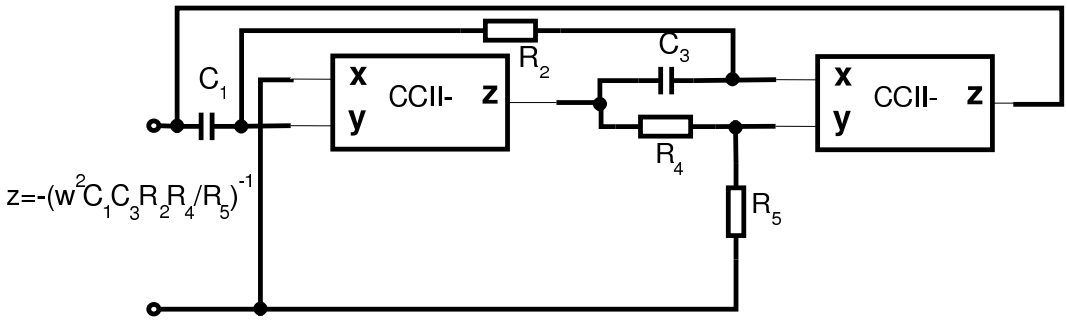


Figure 2.10: CCII based floating FDNR [Sen84].

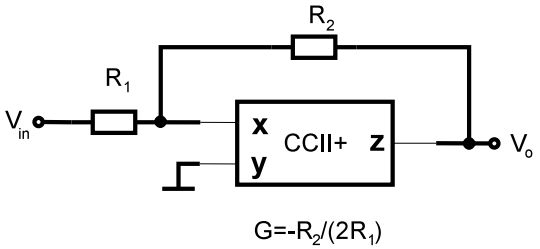


Figure 2.11: CCII amplifier [Wil90].

In figure 2.12, an example of the CCII implementation [ITF02] is shown. This implementation is

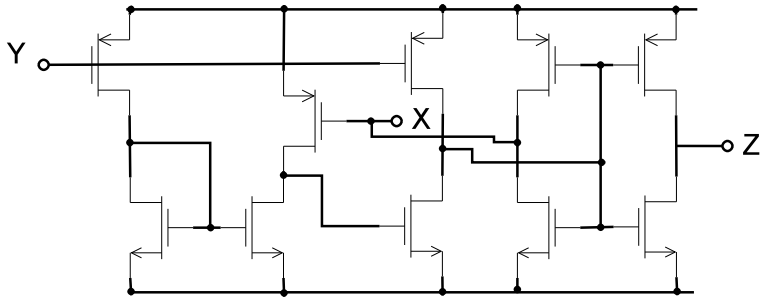
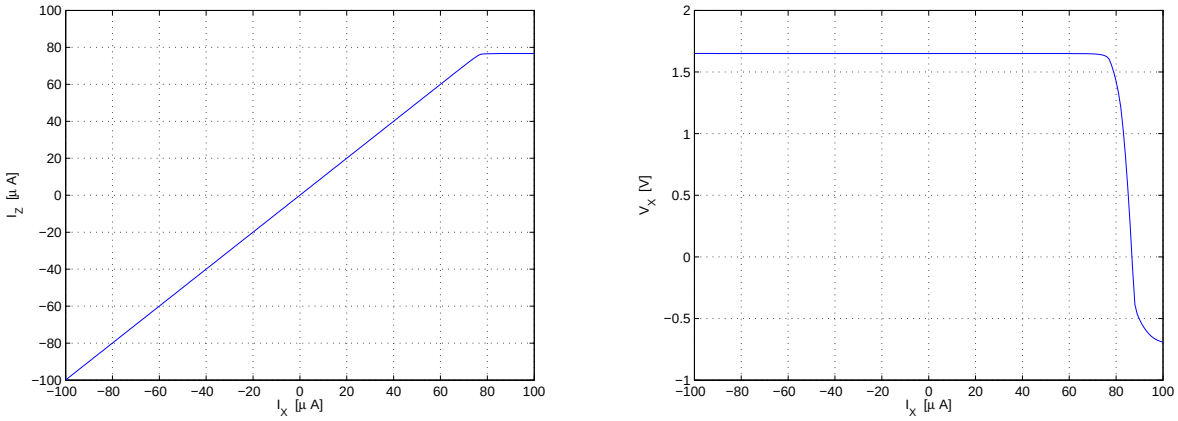


Figure 2.12: A low-voltage, low-power CCII implementation [ITF02].

simulated by Austriamicrosystems $0.350\mu m$ 3.3V CMOS technology spice model. The lengths and the widths of the transistors are set to $1\mu m$ in the simulation. The current transfer characteristics of I_Z and I_X is shown in figure 2.13(a). In figure 2.13(b), the voltage V_X is plots various I_X when the



(a) (b)

Figure 2.13: DC characteristics of the CCII proposed in [ITF02]. a) Current transfer characteristics. b) The V_X vs. I_X when $V_Y = 1.65V$

voltage $V_Y = 1.65V$ is shown. The voltage V_X is equal to V_Y for a wide range of the input current. In figure 2.14, the gain I_Z/I_X is shown. The 3dB bandwidth of the CCII is about 33MHz. However, the presented CCII is not optimized. Later on, in chapter 6, it is described that CCII in [ITF02] was found to be experiemently not stable in many configurations, and therefore, a compensation capacitor is added to the original design. More about the characteristics of the CCII is found in [ITF02, Wil90].

The CDTA [Bio03] block symbol and an example of its implementation are shown in figure 2.15, the outputs of the CDTA block are produced by multi-output operational transconductance amplifier *(MOTA)*. The MOTA input is dependent on the voltage at the point z. The voltage at the point z is produced due to the current difference between the nodes n and p multiplied by the load impedance at the node z. If no load is connected to the node z, the current difference between the nodes n and p is magnified theoretically by $A_o \rightarrow \infty$. Thus, $I_{x^+} = (I_p - I_n) \cdot A_o$, and $I_{x^-} = -(I_p - I_n) \cdot A_o$, which is similar to the operational amplifier formula. The number of outputs of the CDTA is user defined, depending on the application requirements.

In figure 2.16, and example of employing the CDTA as an current mode amplifier is shown.

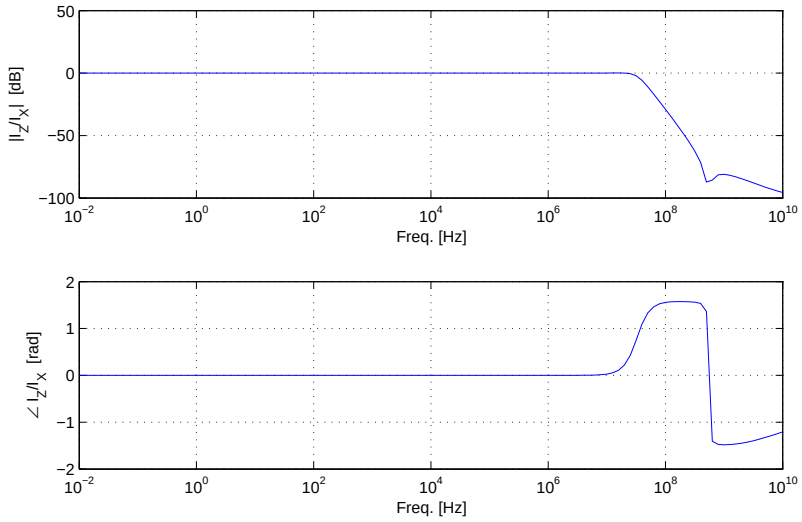


Figure 2.14: AC characteristics of the CCII in [ITF02]

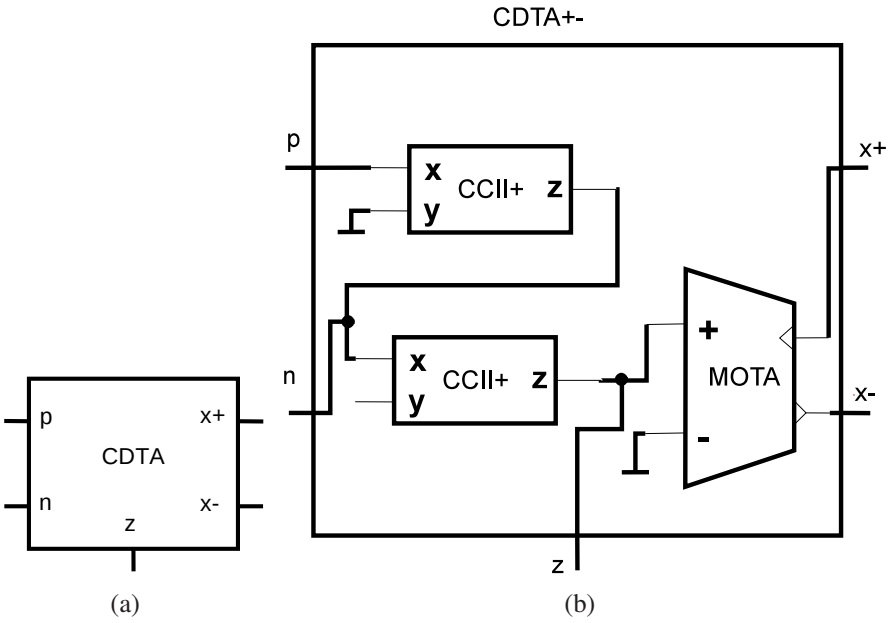


Figure 2.15: CDTA block a) Its symbol. b) Its implementation using CCII+ and MOTA.

The flow graph of the CDTA element is shown in figure 2.17(a). The transformation between the CDTA circuits and flow graph is straight forward, which simplifies the design of current mode circuits. For example, the flow graph in figure 2.17(b) is implemented using a single CDTA element, while its implementation in voltage mode requires two operational amplifiers [Bio03].

Employing the g_m of CDTA allows the design of g_m-C like filters, which is called CDTA-C filters [BVB05]. The second-order of the well known Kerwin-Heulsman-Newcomb filter *(KHN filter)* is implemented in [BVB05] with three CDTA elements and two capacitors as shown in figure 2.18. The

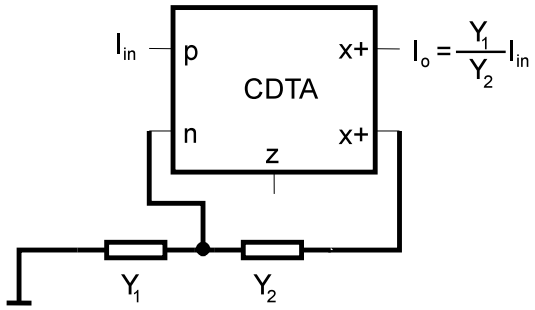
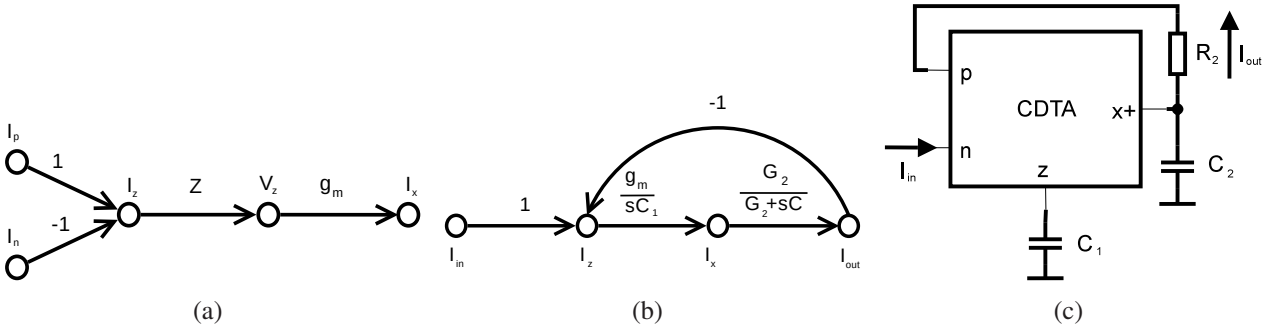


Figure 2.16: CDTA operating as an amplifier.



(a) (b) (c)

Figure 2.17: a) The flow graph of the CDTA. b) The flow graph of second order LP filter using CDTA [Bio03]. c) The implementation of the filter in figure 2.17(b) using a single CDTA element.

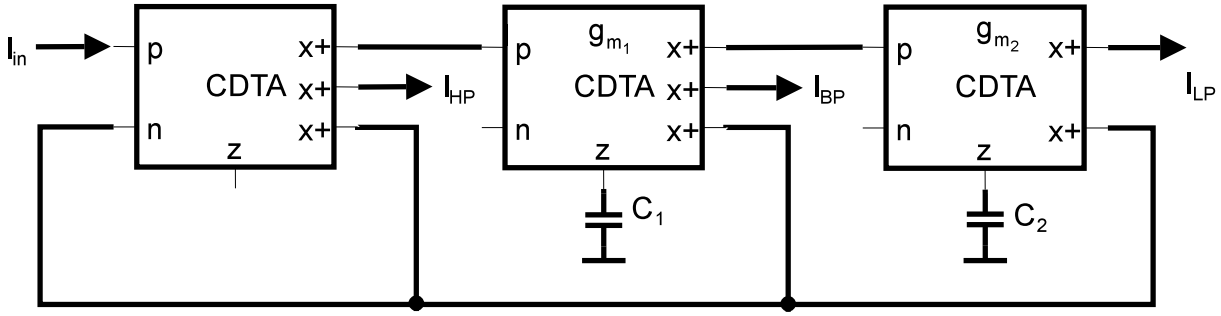


Figure 2.18: Second-order KHN filter using CDTA [BVB05].

output currents of the filter are

$$\frac{I_{HP}}{I_{in}} = \frac{s^2}{s^2 + \frac{\omega_o}{\mathbb{Q}}s + \omega_o^2} \tag{2.7}$$

$$\frac{I_{BP}}{I_{in}} = \frac{\frac{\omega_o}{\mathbb{Q}}s}{s^2 + \frac{\omega_o}{\mathbb{Q}}s + \omega_o^2} \tag{2.8}$$

$$\frac{I_{LP}}{I_{in}} = \frac{\omega_o^2}{s^2 + \frac{\omega_o}{\mathbb{Q}}s + \omega_o^2} \tag{2.9}$$

where $\mathbb{Q}$ is the quality of the filter, and $\omega_o = \sqrt{\frac{g_{m1}g_{m2}}{C_1C_2}}$ is the filter corner frequency. However, the design of this filter is not investigated in this thesis.

More about current mode building blocks are found in [BG04].

2.5 Commercial Analog Reconfigurable ICs

The commercial reconfigurable chips are oriented toward two target approaches:

- Hardware with reconfigurable structure to program the circuit functionality for rapid prototyping.
- Hardware with fixed structure, but few soft-trimmable components in order to control few of its specifications to recover from deviations.

An overview on the commercial flexible hardware with the programmable functionality and on the trimmable approaches is provided the following subsections.

2.5.1 Field Programmable Analog Arrays

Rapid prototyping is necessarily in many analog applications for cost reduction if small quantities are required to be produced. Various flexible hardware have been developed with the motivation of rapid prototyping rather than self-calibration for non-precise applications.

Field-programmable analog array *(FPAA)* is an analog integrated circuit which has programmable connections between the utilized building blocks to implement user defined functionality. The active components such as operational amplifiers are not programmable, but the passive components can be programmable to tune the functionality *(e.g. to choose the gain of an amplifier)*.

The basic programmable cell in FPAAs is called CAB *(configurable analog block)*. Each CAB can be reconfigurable to construct a certain functionality. The connections between the CABs are programmable in order to build the system functionality.

The IMP, Inc.'s version of the FPAA which is called EPAC *(electrically programmable analog circuit)*, the Motorolla MPAA020 series, and the Anadigm® FPAA are discrete-time based on switched-capacitor technology which limits the operation frequency and generates noise and non-linearity due to the switching activity. Although the bandwidth of the MPAA020 is about 200kHz because of the discrete components, the typical bandwidth of the Anadigm® AN121E04 is about 2MHz, but the effective bandwidth is less, depending on the design.

Zetex's FPAA which is called TRAC *(totally reconfigurable analog circuit)* is continuous time FPAA in which, CAB can be configured to a prescribed functionality such as add, negate, non-inverting pass, log, anti-log, rectifier, or to employ external passive components *(e.g., to construct an amplifier, a filter, etc)*. The bandwidth of TRAC020 is 4 MHz - 12 MHz depending on the reconfiguration, and it includes 20 CABs in the chip.

2.5.2 Trimmable Hardware

Programmable passive components –such as AD8403 [Dev] is a programmable resistor from analog devices, X90100 [Int] is programmable capacitor from intersil– can be employed for trimming the system as described in section 2.3. It is more preferable in many applications to employ programmable capacitors rather than programmable resistors in order to avoid integrating the resistors. However, the employed switches react as a low-pass filter due to its internal resistance and its parasitic capacitance, which limit the operation bandwidth. As an example of employing programmable components with fixed structure is the Texas Instruments smart sensor conditioning chip PGA309, which has offset adjustment, programmable gain, a look up table is employed to calibrate the error according to environment temperature, etc.

Rejustors [Tec] are a analog controlled resistors that are introduced by Microbridge Technology Corp. for sensor electronics calibration. It is based on employing the MEMS technology to place a heater in the substrate of an integrated resistor. The main disadvantage of the rejustors is that integrating them with other devices in the same chip may increases the dynamic deviations in the other devices due to the heat distribution inside the chip. The rejustor development board employs an ADC and a DAC to adjust the value of the rejustor iteratively. The settling time of the heating system has to be considered during the iterative adjustment. Due to the self heating of the rejustor, it is useful only for low power application. The main advantage of the rejustor over the digitally programmable resistor is that no digital switches are required, and thus, it can be employed in high frequency applications as it has less parasitic impedance.

Employing programmable or adjustable passive components can adjust few of the hardware specifications, e.g., offset, closed loop gain, but it can not reconfigure the hardware against strong system deviations or to suit another application requirements.

2.6 Discussion

Organic systems are realized as a process of enormous complexity [vdM04, MSvdMW04]. Organic-computing mimicks the abstract behavior of the organic systems; aiming on building a robust, safe, flexible, high complexity, trustworthy system by including the self-x properties such as self-organizing, self-configuring, self-optimizing, self-healing, self-protecting, and self-explaining. The state-of-the-art organic-computing is concentrated in the digital domain. This research contributes to abstract the organic-computing to the analog domain by including the self-x properties in sensor electronics [KLT06, KÖ6].

Adding the self-x properties *(self-organization, self-configuration, self-optimization, self-healing, etc.)* to sensor electronics is essential to keep the precision of the sensor system and cope with static and dynamic deviations. On the other hand, rapid prototyping reduces the cost of the development and design phases especially for low quantity products. The MEMS technology accommodates manufacturing of high quality complex sensors with the cost of die area and the number of required masks. However, the state-of-the-art MEMS technology offers only digital domain self-calibration. The topology of the MEMS sensor electronics can be fixed according to the sensor specifications. Thus, self-x properties can be included by employing programmable components to the pre-designed

sensor electronics hardware in order to extend the hardware specification range without varying the system functionality [TK05b]. Therefore, adding flexibility allows the system to cope with harsh environment. Nevertheless, the output signal of the sensor system is application dependent, and thus expanding the flexibility in the functional level widen the system application range. The high temperature electronics research field aims to improve the semiconductor technology to operate at high temperature. In the state-of-the-art high temperature electronics, a dynamic operation temperature range up to $1055°C$ has been attained in laboratories by employing diamonds semiconductors. Thus, it is worth adding flexibility to many of the high temperature sensor electronics applications to keep the system performance at different operating points. As the die area and the number of masks required for MEMS technologies is large, increasing the flexibility to the functionality level in order to widen the range of applications for a single MEMS, and including self-x properties is essential. Although adding flexibility increases the design die area, extending the flexibility to build general purpose generic sensor electronics system decreases the cost and the design time as the same hardware can be used for many applications. FPAAs has a programmable hardware functionality for rapid prototyping in the design phase, but it does not consider the self-x properties, while conventional self-calibration hardware includes some of the self-x properties in the run time but only against few of the possible system deviation. For example, auto-zeroing can minimize only the offset, and gain calibration consider only the closed loop gain.

In order to obtain generic organic-computing sensor electronics, current mode building blocks are included to extend the hardware flexibility, to simplify the implementation of current mode circuits instead of converting the signals to voltage mode before applying the required processing, to enhances testing new circuits on the rapid prototyping generic sensor system at low cost, and to increase the hardware speed as current-mode circuits are faster than voltage-mode circuits in the cost of the linearity [BG04, Kol00].

A blueprint for the target generic organic-computing sensor electronics front-end is shown in figure 2.19. Analog to digital converters and digital to analog converters are required to assess the sensor

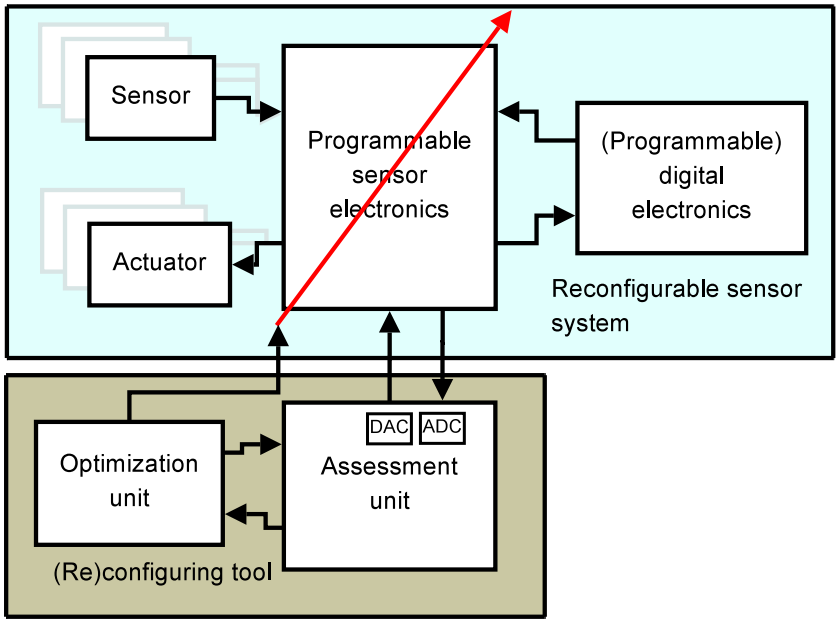


Figure 2.19: Generic organic-computing sensor electronic system.

electronics calibration. Any error in the signals of the assessing ADCs propagates to the whole system. If the ADCs are calibrated, the DACs can be calibrated easily in analog or digital domain. Fortunately, the state-of-the-art ADCs employs self calibration techniques to compensate the deviations. Therefore, simple assessment circuits that contain ADCs, DACs and very few passive components can be self-calibrated.

CHAPTER 3

Evolutionary Computation

The evolutionary computation methods are applied in a wide range of engineering applications [SD01, Tho95, Tho97b, PLF+99, Yao99]. They are stochastic search methods inspired from biological systems. The advantage of the evolutionary computation over exact optimization methods is that the exact optimization methods consume much time that makes them hard to be applied in many of the practical engineering applications.

In this chapter, the background of evolutionary computation is described, which is used later on in the thesis. Adapting the evolutionary computation for dynamic environment is discussed, in which the algorithm does not lose the previous information after the occurrence of an environmental change. In addition, a new variant of the particle swarm optimization is described, and its performance compared and tested using the benchmark functions which are widely used for evaluating the evolutionary computation optimization approaches.

3.1 Historical Background

The evolutionary computation can be considered under the family of optimization techniques as in figure 3.1, or under the family of computational intelligence as shown in figure 3.2.

The evolutionary methods can be used to program the problem dimensions *(e.g., genetic algorithms, differential evolution, and particle swarm optimization)*, to design the problem structure *(e.g., Genetic programming)*, to find the optimal path *(e.g., ant colony optimization*[1] *[Dor92])*, or in pattern recognition, *(e.g., artificial immune system*[2] *[FPP86])*. Both particle swarm optimization and ant colony optimization are members belong to the family of swarm intelligence. The swarm intelligence inspired from the behavior of the swarm and the interaction between its individuals, while genetic

[1] Ant colony optimization is inspired from the ant colony and the techniques they use to find the shortest path for the food, which means, it is a path minimization technique.

[2] Artificial immune system [FPP86] is inspired from the immune system of the vertebrate and how it recognizes the foreign objects in the body.

algorithms, genetic programming, evolutionary strategies, and evolutionary programming are based on the Darwinian evolution [ES03].

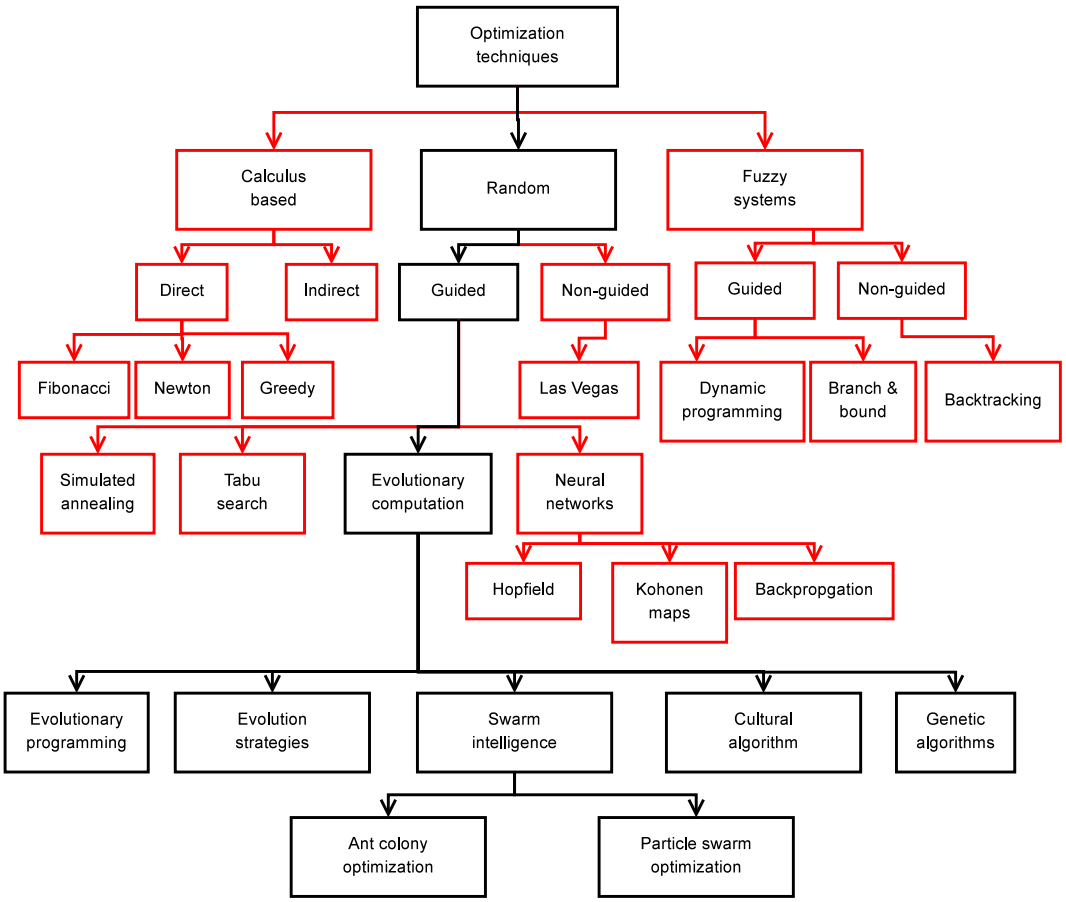


Figure 3.1: Optimization techniques' taxonomy [Aff05].

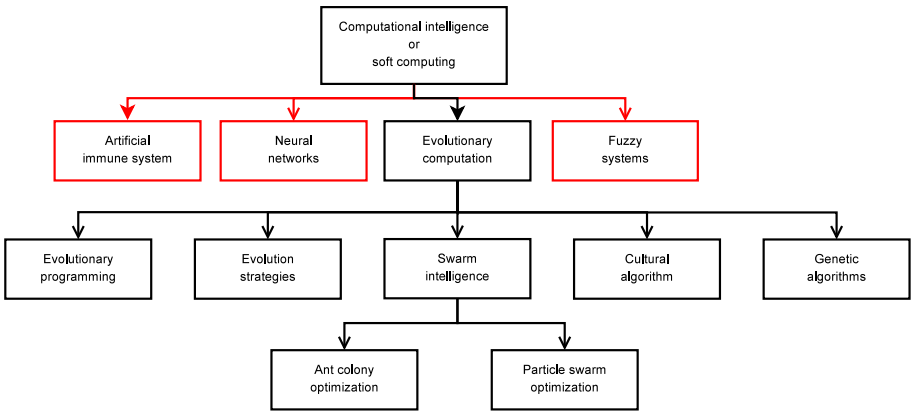


Figure 3.2: Computation intelligence taxonomy [Chr03].

Alan Turing identified [KIAK99] in his paper "Intelligent Machinery" in 1948 [TS00]: *"There is the genetical or evolutionary search by which a combination of genes is looked for, the criterion being the survival value."*. In his paper in 1950 "Computing Machinery and Intelligence", Turing further added: *"We cannot expect to find a good child-machine at the first attempt. One must experiment with teaching some such machine and see how well it learns.*

Structure of the child-machine = Hereditary material
Changes in the child-machine = Mutations
Natural Selection = Judgment of the Experimenter".
Afterwards, Fraser introduced genetic system which is the early work relied on mutation rather than mating to generate new gene combinations in [Fra57]. In 1962, Bremermann executed computer experiment [Bre62] of the genetic system and added a kind of mating where the characteristic of offspring were determined by summing up corresponding genes in two parents. By the mid-1960's Holland had developed the genetic algorithm, that is well suited to evolution by both mating and mutation as presented in [Hol73], while Fogel called it evolutionary programming [FOW65]. Meanwhile, Rechenberg and Schwefel invented evolution strategies [Rec73b].

The first experiments with Genetic programming were reported by Smith [S.F80] and Crame [Cra85], then by John Koza [Koz90].

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Eberhart and Kennedy [EK95], inspired by social behavior of bird flocking and fish schooling.

3.2 Genetic Algorithm

In the genetic algorithm (GA), the problem is represented in a chromosome, also called genome. The genome consists of a gene string. Each of the genes represents a dimension in the problem space. The genetic algorithm is based on Darwinian evolution theory which assumes that the fitness of the population improves over the generations. The genetic algorithm is a population based algorithm, which means, it has a set of solutions each generation *(iteration)*. Each of the solutions is called an individual. Evaluating the individuals is carried out by a fitness function. The fitness is represented by a real number, called fitness value, in order to differentiate between the fitness of the individuals by its value. Evaluating the genome is performed by calling a fitness function which returns a fitness value corresponding to the gene values. Sometimes a scaling is used after evaluating the individuals.

Procedure 1 Genetic algorithm procedures.

```
Population.Genome ⇐ Random(seed)
Population.Fitness⇐ Evaluate(Population.Genome,Fitness Function)
Population.Fitness⇐ Scale(Offspring.Fitness)
IterationCounter⇐ 0
repeat
  Parents ⇐ Select Parents (Population,Selection Operator)
  Offspring.Genome⇐ Produce Offspring (Parents,CrossOver Operator)
  Offspring.Genome⇐ Mutute(Offspring,Mutation Operator)
  Offspring.Fitness⇐ Evaluate(Offspring.Genome,Fitness Function)
  Offspring.Fitness⇐ Scale(Offspring.Fitness)
  Population⇐ Insert(Population,Offspring,Population Size,Mixing Operator)
  IterationCounter⇐ IterationCounter+1
until IterationCounter>MaxIteration OR TerminationCondition==TRUE
```

As in procedures 1, the basic operators used by genetic algorithm are the initialization, the selection, crossover, mutation, and survivor selection.

In the following subsections, a brief description of the genetic algorithm operators [ES03] is expressed.

3.2.1 Initialization

The initializer operator initializes the initial generation by random numbers. The initialization plays a role in finding a good solution; if the initializer luckily to initialize the genomes with a value near by the global best solution, it is easier to the algorithm to find the best solution.

An integer called seed is employed by the initializer, in which the initializer produces the same random number sequence each time the same seed is used.

The main difference between the initializers is the distribution of the random numbers that it generates. The uniform initializer generates uniform distributed random numbers. It is preferable in many of the applications to use the uniform initializer as it spreads the individuals over the search space uniformly. Some applications need special initializers like the adjacency-type problems, which have an atomic number in each gene that can not be repeated. The ordered initializer, for instance, initializes the genome with a given set in which each of the value can appear only in one gene in the genome.

3.2.2 Selection

By applying a selection approach, some of the individuals is selected and recombined in the next generation. Choosing the parents for mating is one of the key operators in genetic algorithm. A weak selection method can lead to convergence in a local minima or maxima.

The simplest selection method is the uniform method which selects the parents randomly with uniform distribution random generator. This method gives all the individuals in the populations the same chance for mating.

In roulette-wheel selection, each individual is weighted for selection according to its fitness value. The disadvantage of this method is that it can choose a parent frequently for mating if its fitness is much higher than the rest of the population. This can trap the population in a local extrema.

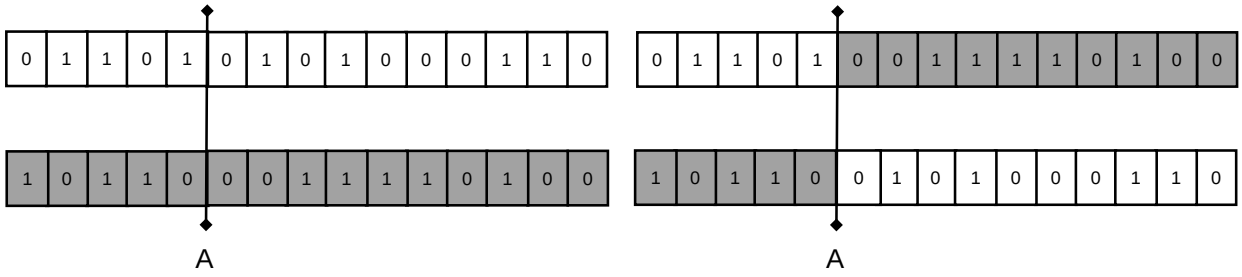
The ranking-based selection considers the drawback of the roulette-wheel selection by ranking individuals instead of weighting them according to their fitness. It sorts the population according to their fitness, and then selects the individuals by probability according to their rank in sorted list.

Tournament selection method selects randomly two or more individuals, then the fittest of them is chosen.

3.2.3 Crossover

The crossover is employed to create new offspring from the parents. It can be achieved by choosing one or more cutting point in the parents, in which the crossover is done at them. For example in figure

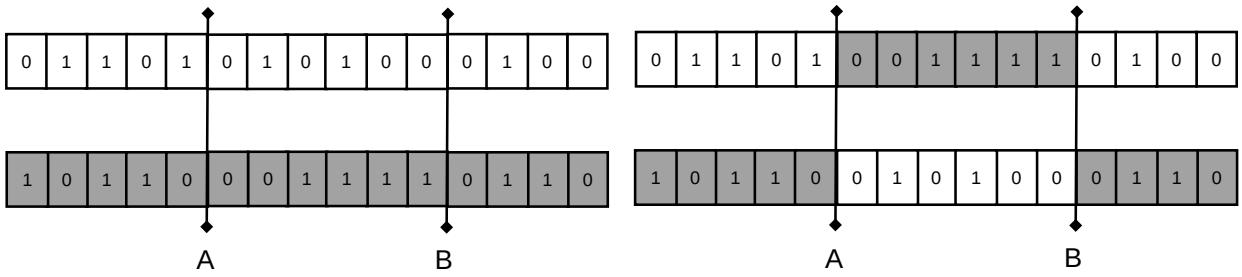
3.3, the parents in figure 3.3(a) produce the offspring in figure 3.3(b) after a one point crossover at the point ”A” in figure 3.3(a).



(a) (b)

Figure 3.3: Example of one point crossover. a) Parents. b) Offspring.

In figure 3.4, two point crossover is shown, where the parents in figure 3.4(a) produced the offspring in figure 3.4(b).



(a) (b)

Figure 3.4: Example of two point crossover. a) Parents. b) Offspring.

The odd-even crossover takes the even bits from one of the parents, and the odd from the other.

Uniform crossover chooses genes from any of the parents randomly; it takes some genes from the first parent and rest of the other. The uniform crossover produces only one offspring from two parents.

Arithmetic crossover is applied only to real number problems, it generates a new value from the parents using the following equation:

$$\begin{aligned} \mathbf{x}_{Offspring_1} &= \alpha \cdot \mathbf{x}_{Parent_1} + (1-\alpha) \cdot \mathbf{x}_{Parent_2} \\ \mathbf{x}_{Offspring_2} &= (1-\alpha) \cdot \mathbf{x}_{Parent_1} + \alpha \cdot \mathbf{x}_{Parent_2} \end{aligned} \tag{3.1}$$

where α is a parameter needed for arithmetic crossover, $\mathbf{x}_{Offspring_1}$ and $\mathbf{x}_{Offspring_2}$ are two arrays that contain the values of the genes of the new offspring, while, $\mathbf{x}_{Parent_1}$ and $\mathbf{x}_{Parent_2}$ are two arrays that contain the values of the genes of the parents. If $\alpha = 0.5$, the arithmetic crossover will return the average of the parent values, which returns two identical offspring. The offspring need modifications afterwards in order to have different values, e.g., a big mutation probability.

An example of the arithmetic crossover with $\alpha = 0.5$ is shown in figure 3.5, where the offspring in figure 3.5(b) are the average of the parents in figure 3.5(a).

Blend crossover is also applied to real number problems, it generates a new random value based on the interval between the parents.

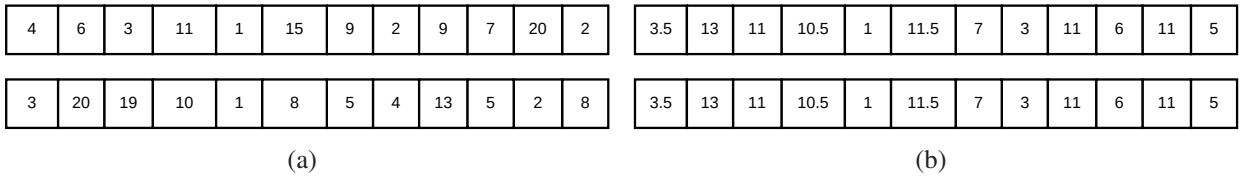


Figure 3.5: Example of arithmetic crossover with $\alpha = 0.5$. a) Parents. b) Offspring.

3.2.4 Mutation

A percentage of the genes is required to mutate. A parameter called mutation probability, which has a value between 0 and 1, is employed to select randomly the genes that will mutate. Before mutating any gene, first a random number with uniform distribution is generated and compared with the mutation probability. If the mutation probability is bigger than the generated random number, the mutation is applied.

The flip mutation or the swap mutation can be used for binary genomes. The flip mutation flips the gene value. It converts the 1 to 0 and the 0 to 1. The swap mutation swaps the value of two genes in the offspring.

For real value genomes, the Gaussian mutation can be use. The Gaussian mutation generates a new random value based on a Gaussian distribution around the current value. This returns a small deviation in the mutated genes.

3.2.5 Survivor Selection

After generating the offspring, a selection between the offspring and the parents is required in order to form the new generation. There are several methods for the survivor selection. Age-based replacement removes all the individuals from the generation after a given age. This means that after some generations, the best individuals will be removed if no improvement in the algorithm in achieved.

Another approach is fitness based replacement which keeps the fitter individuals and replaces the worse. This method does not control the mixing between the old generation and the new generation.

If only a percentage of the old generation should be kept, elitism can be used to keep only a given percentage from the old generation. It copies the required most fit percentage of individuals to the new generation.

3.2.6 Genetic Algorithm Techniques

The simple genetic algorithm technique consists of one population. For each new generation, the elitism can be used to keep a percentage of the old generation, where choosing the old individuals is done concerning its fitness.

In the steady state genetic algorithm, only a certain number the individuals are replaced by new offspring, and the rest of the population is kept. The replaced individuals can be selected according

to their fitness, or selected randomly.

3.3 Genetic Programming

Genetic programming (GP) has similar operations like the genetic algorithm; the main difference is that genetic programming search for mathematical formula, object structure, or a computer program to solve the problem. The problem is represented as a syntax tree. The leaves of the tree are either variables or constants, while any other node in the tree is an operator. For example the tree in figure 3.6 represents the mathematical formula $(((X+Y)/2)-B)$.

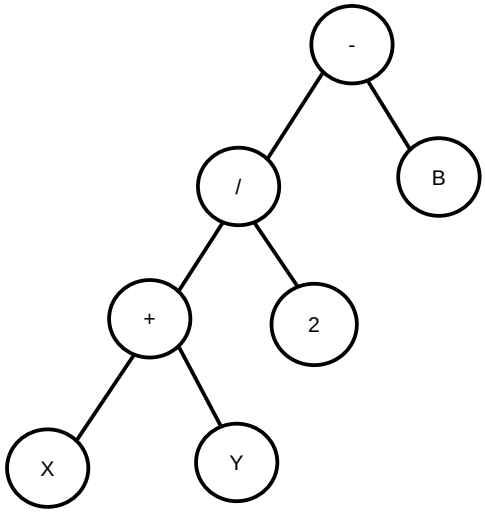


Figure 3.6: Example of Genetic programming syntax tree.

The fitness of the individual is represented by assigning a value to it like in genetic algorithm, which has to be minimized or maximized. For generating the offspring in genetic programming, selection, crossover, mutation and survivor selection operators are utilized. The selection and survivor selections are similar to the genetic algorithm.

The crossover can be applied on either identical or different parents. It is done by choosing one or more cutting points in the parents and recombining them at these points.

For example, the parents in figure 3.7 produce the offspring in figure 3.8 after crossover at the cutting points marked in gray.

Crossover for identical parents is achieved by choosing different cutting point in the parents and recombining them at these cutting points. As example, the parents in figure 3.9 produce the offspring in figure 3.10 after recombining at the gray marked points.

After the crossover, some of the nodes (operators, variables or constants) mutate. Mutation probability is employed for selecting the nodes to mutate.

As example for mutation in shown is figure 3.11. The nodes which are selected for mutation of the offspring in figure 3.11(a) are marked gray. After mutation, the mutated offspring is in figure 3.11(b).

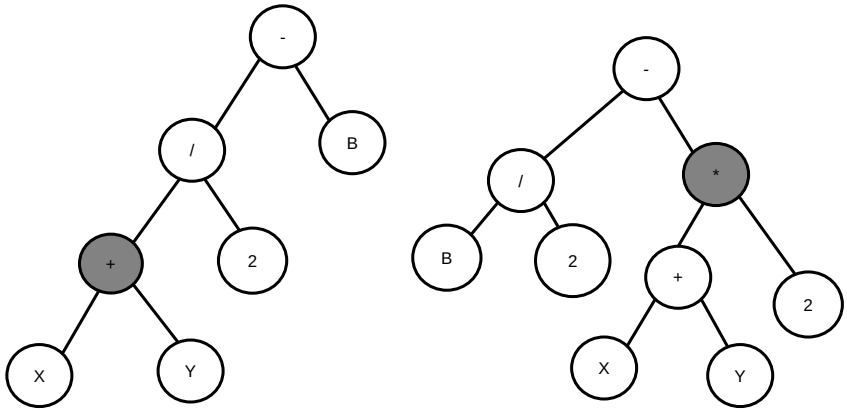


Figure 3.7: Two selected parents for crossover.

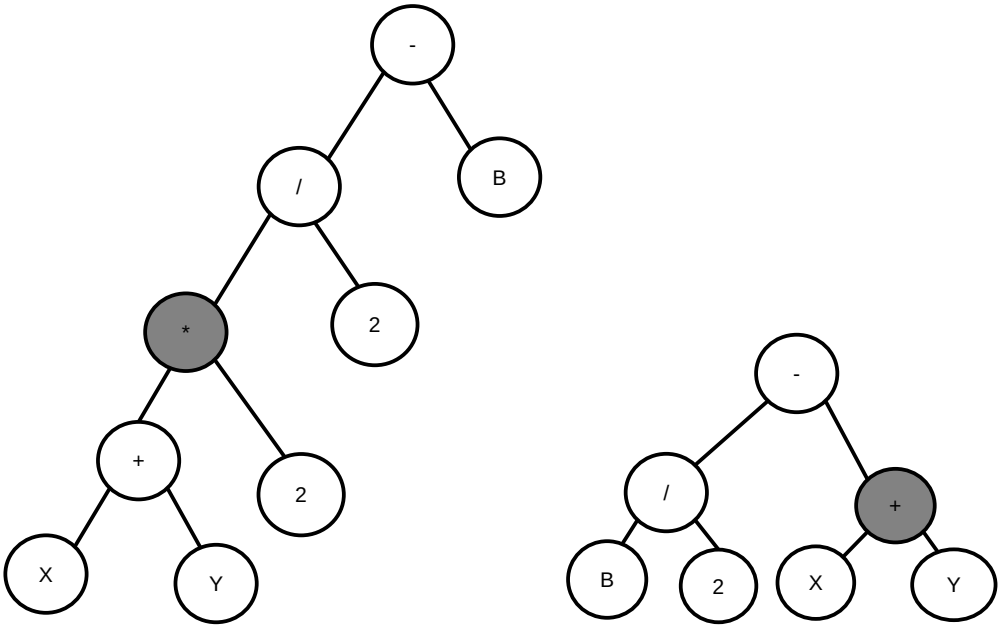


Figure 3.8: The offspring after crossing over of the parents in figure 3.7.

A limitation on the tree size has to be imposed in the evolution to avoid the tree explosion.

3.4 Particle Swarm Optimization PSO

In particle swarm optimization *(PSO)*, the potential solutions are the position of the individuals. The individuals in PSO are called particles. Each particle flies in the problem space. The speed of each particle is determined by equation 3.2, where $\mathbf{x}$ represents the position of the particle, $\mathbf{x}_{t,i}$ is the position of the particle i at the time *(iteration)* t, and $\mathbf{p}_{t,i}$ is the position of the particle i at the time t where it has achieved its best fitness value. The particle g is the global best particle, which is the particle that has achieved the best fitness over all the population. $\mathbf{p}_{t,g}$ is the position of the particle g where it has achieved its best fitness. $\mathbf{rand}()$ is a random number generator. w is called the inertia

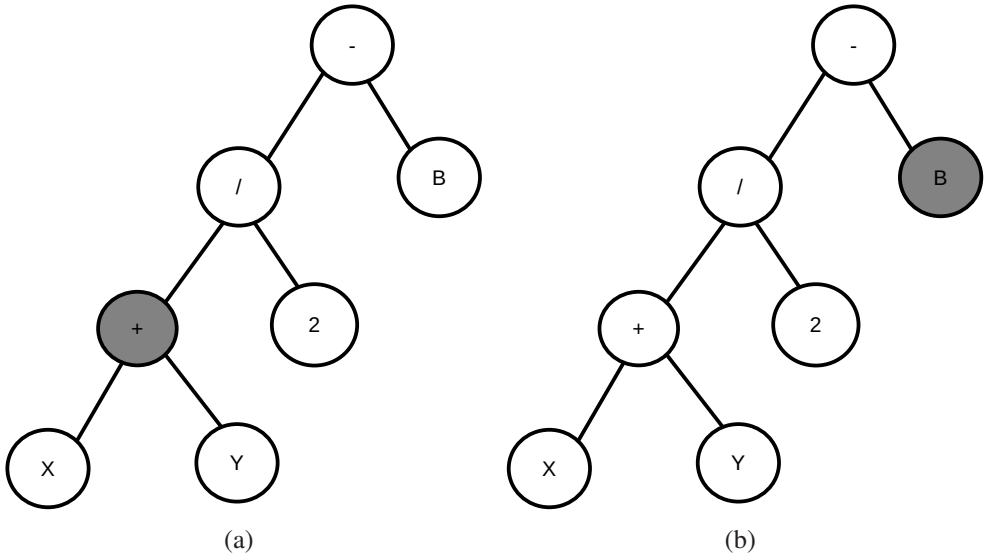


Figure 3.9: Selected parent for identical crossover.

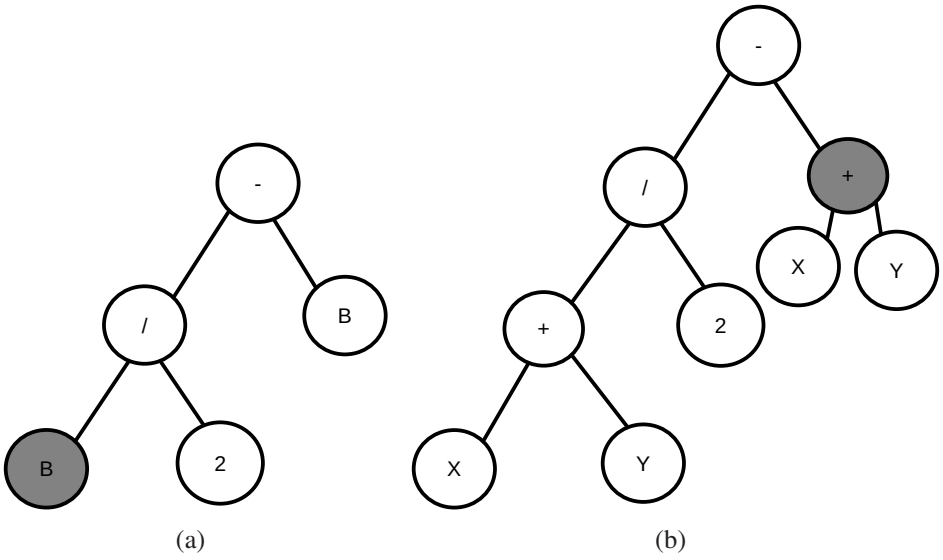


Figure 3.10: The offspring after crossover of the identical parents in figure 3.9.

weight, it is introduced by Shi and Eberhart in [SE98b, SE98a]. In the initial PSO, w was equal to one. C_1 and C_2 are called the acceleration coefficients. The first part of the equation is its inertia, the second part is called the "cognitive" part, which represents learning from its own old experience and the third part is the "social" part, which represents the group flying experience.

$$\mathbf{v}_{t+1,i} = w\mathbf{v}_{t,i} + C_1 \cdot \mathbf{rand}() \cdot (\mathbf{p}_{t,i} - \mathbf{x}_{t,i}) + C_2 \cdot \mathbf{rand}() \cdot (\mathbf{p}_{t,g} - \mathbf{x}_{t,i}) \tag{3.2}$$

The position of the particle is updated each iteration by equation 3.3

$$\mathbf{x}_{t+1,i} = \mathbf{x}_{t,i} + \mathbf{v}_{t+1,i} \tag{3.3}$$

The procedures for traditional PSO are as shown in procedures 2, where x_{max} is the maximum value that $\mathbf{x}$ can take in any of its dimensions, and v_{max} is the maximum allowed velocity which is used

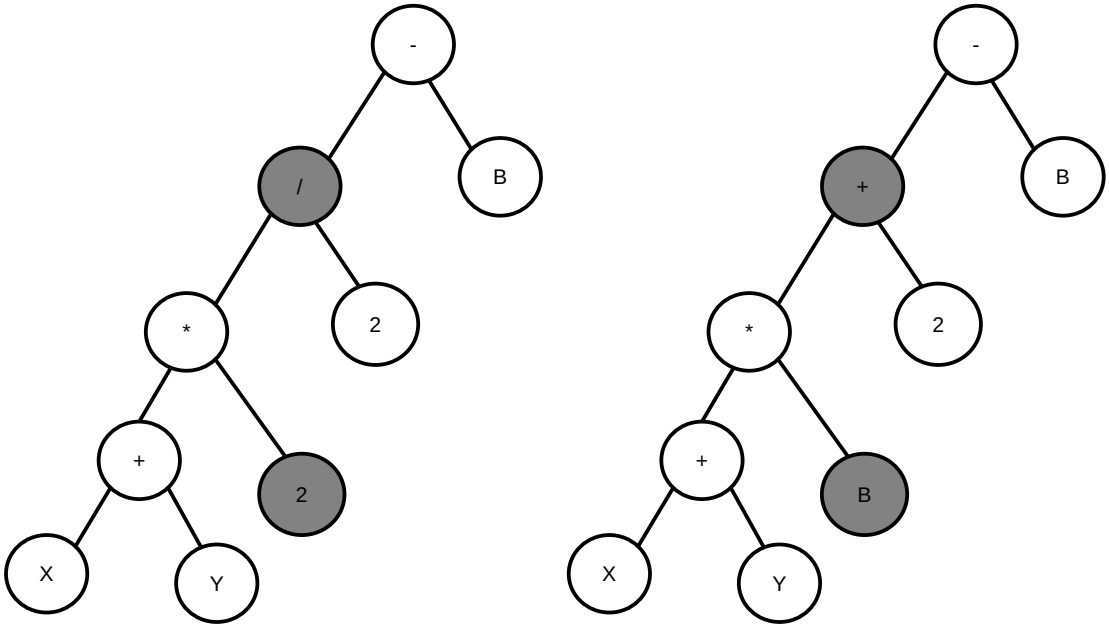


Figure 3.11: a) An offspring before applying mutation. b) The offspring in figure 3.11(a) after mutation.

during the initialization, and in some PSO deviations to limit the velocity during the run. Each particle has knowledge about its current position, its fitness value at the current position, its velocity, the best fitness value it has achieved and its location, and the global best particle. The fitness value of all the particles is compared in order to find the global best particle g. [3].

3.4.1 Modifications of Particle Swarm Optimization

Many modifications of the original particle swarm optimization have been developed to improve its performance. In the following subsections, the PSO variants that are mostly used the literature are summarized.

Neighborhood Topologies

When all the particles have only one global best, it is called *gbest* model. It is possible to have local best *lbest* model in which the neighbors of each particle is a subset of the population instead of all the population. As a result, it has a local best particle which is the best particle in its neighbors [KM02, Men04]. In figure 3.12, an example of a swarm with a population size of 20 particles is shown with global neighboring. Each particle is aware of the fitness of the others, as a result, all the particles aware of the best particle and its position. In figure 3.4.1 an example of an *lbest* model PSO is shown, in figure 3.13(a), each particle has only 2 neighbors, while in figure 3.13(b), each particle has 6 neighbors.

[3]The detection can be done iteratively inside the updating loop after calculating the fitness of each particle

[4]The comparison operator is $<$ in case of minimization, $>$ in case of maximization.

Procedure 2 Particle Swarm Optimization procedures.

Require: N , FitnessFunction f
Ensure: Find Optimal Solution
{Initialization}
Initialize the random generator with the seed
for $i = 1$ to N **do**
$\mathbf{x}_i = (2 \cdot rand() - 1) \cdot x_{max}$
$\mathbf{v}_i = (2 \cdot rand() - 1) \cdot v_{max}$
$f_i = f(\mathbf{x}_i)$
$\mathbf{p}_i = \mathbf{x}_i$
$pf_i = f_i$
end for
g $\Leftarrow$ Identify the particle with the global best
{Learning}
for $t = 0$ to t_{max} **do**
for $i = 1$ to N **do**
$\mathbf{v}_i = w \cdot \mathbf{v}_i + C_1 \cdot \mathbf{rand}() \cdot (\mathbf{p}_i - \mathbf{x}_i) + C_2 \cdot \mathbf{rand}() \cdot (\mathbf{p}_g - \mathbf{x}_i)$
$\mathbf{x}_i = \mathbf{x}_i + \mathbf{v}_i$
$f_i = f(\mathbf{x}_i)$
if $f_i <^4 pf_i$ **then**
$\mathbf{p}_i = \mathbf{x}_i$
$pf_i = f_i$ {*pf_i is value of the best achieved fitness by the particle i*}
end if
end for
g $\Leftarrow$ Identify the particle with the global best
if TerminationCondition==TRUE **then**
break
end if
end for

Constriction Factor

Clerc [Cle99] and Clerc and Kennedy [CjK02] introduced the constriction factor into the PSO to ensure convergence. The constriction factor is utilized to choose the value of C_1, C_2 and w. The speed of the particle is updated by equation 3.4 instead of equation 3.2, while the value of χ is calculated by equation 3.6

$$\mathbf{v}_{t+1,i} = \chi \cdot (\mathbf{v}_{t,i} + \Phi_1 \cdot \mathbf{rand}() \cdot (\mathbf{p}_{t,i} - \mathbf{x}_{t,i}) + \Phi_2 \cdot \mathbf{rand}() \cdot (\mathbf{p}_{t,g} - \mathbf{x}_{t,i})) \tag{3.4}$$

$$\Phi = \Phi_1 + \Phi_2 \tag{3.5}$$

$$\chi = \frac{2\kappa}{|\, 2 - \Phi - \sqrt{\Phi^2 - 4\Phi}\,|} \tag{3.6}$$

where $\kappa \in [0,1]$ and $\Phi > 4$. Usually κ is set to 1 and both Φ_1 and Φ_2 are set to 2.05 [ES00, Tre03]. This variant is called canonical PSO.

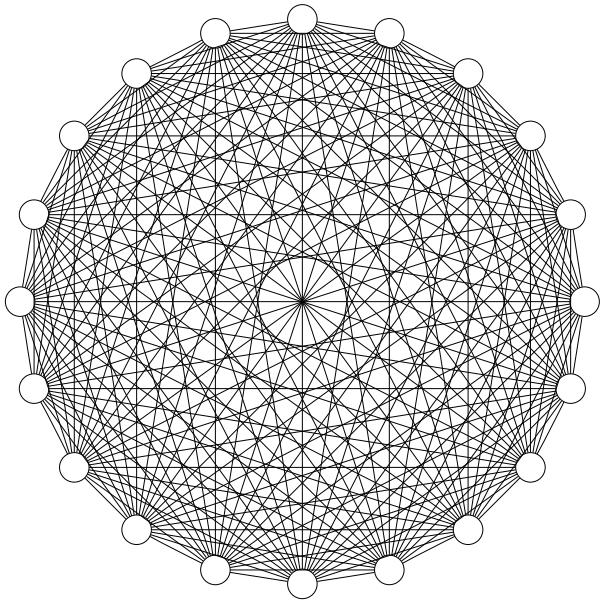
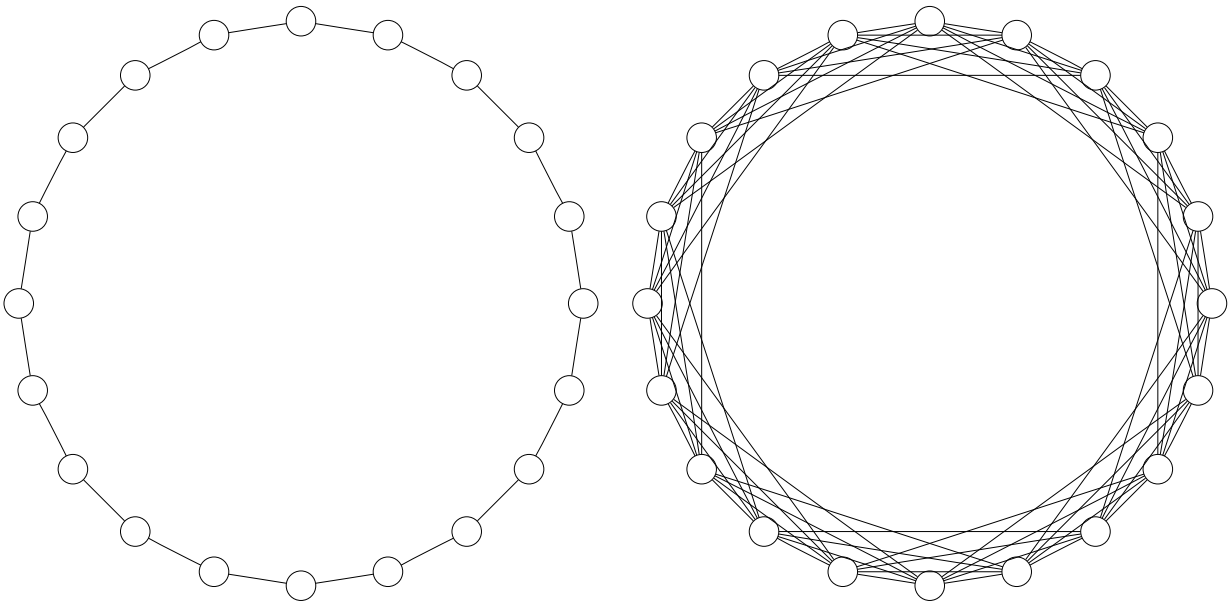


Figure 3.12: gbest model; each particle is aware of all the population.



(a) (b)

Figure 3.13: lbest model; a) each particle has two neighbor. b) each particle has six neighbors.

Charged PSO

In the charged swarms *CPSO* in [BB02a], some particles in the swarm are charged. The charged particles dispel each other. The aim of using charged particles is to maintain the population diversity [Bla05]. The dispelling is done by adding a new component a_i to the velocity equation of the particle swarm optimization as shown in equation 3.7, which is called extra particle acceleration of the particle i.

$$\mathbf{a}_i = \sum_{j \neq i} \frac{Q_i \cdot Q_j}{r_{ij}^2} \frac{\mathbf{r}_{ij}}{r_{ij}} \qquad P_{core} < r_{ij} < P \tag{3.7}$$

where $\mathbf{r}_{ij} = \mathbf{x}_i - \mathbf{x}_j$, $r_{ij} = \mid \mathbf{x}_i - \mathbf{x}_j \mid$, each particle i has a charge of magnitude Q_i and current position $\mathbf{x}$. Neutral particles have no charge, which means $Q_i = 0$ and therefore $\mathbf{a}_i = \mathbf{0}$. Repulsion can occur within the shell $P_{core} < r_{ij} < P$. P_{core} is to protect against division by zero, while P specifies the shell outer radius.

The extra particle acceleration $\mathbf{a}_i$ is added to the velocity update equation as in equation 3.8

$$\mathbf{v}_{i,t+1} = w \cdot \mathbf{v}_{i,t} + C_1 \cdot \mathbf{rand}() \cdot (\mathbf{p}_{i,t} - \mathbf{x}_{i,t}) + C_2 \cdot \mathbf{rand}() \cdot (\mathbf{p}_{g,t} - \mathbf{x}_{i,t}) + mathbfa_i \tag{3.8}$$

In [BB02a], two types of swarms with charged particles were introduced; *charged swarm* and *atomic swarm*. All the particles in the charged swarm are charged. Half the particles of the atomic swarm are charged, and the rest are neutral. The results Blackwell achieved with the atomic swarm was better than the results he achieved with the charged swarm.

It is clear in equation 3.7 that the complexity of the CPSO is $O(N^2)$ instead of $O(N)$ in the traditional PSO, where N is the population size.

Quantum Swarm Optimization

The quantum swarm optimizer (QSO) is built on the atomic metaphor of the CPSO. If any particle gets close to the best particle, a random number is added to it to dispel it into the orbit cloud [Sil07].

Multi-Swarm Particle Swarm Optimizer

In multi-swarm particle swarm optimizer *(MSPSO)* [BB04], the population of the particle swarm optimization is divided into n_{swarm} sub-swarms. Each of the sub-swarms has its own *attractor*, which is equivalent to the global best particle in the PSO. The attractor is the optimal particle in the sub-swarm. The optimal value of the swarm is the best over all the sub-swarms. If two sub-swarms are moving toward the same optimal point, the swarm with the worse attractor will be reinitialized. Detecting if two swarms are flying toward the same optimal point is based on measuring the distance between the sub-swarm attractors, then compare it with a minimum allowed distance r_{excl}. Some of the particles in the swarm are charged. The charged particles dispel each other as in the CPSO and QSO.

Hierarchical Particle Swarm Optimizer

The Hierarchical Particle Swarm Optimizer (HPSO) is introduced by Janson and Middendorf [JM03]. All the particles are arranged in a hierarchy tree that defines the neighborhood structure. Each node of the tree is a particle, each particle flies toward its own best position and toward the best position of the particle that is directly above it in the hierarchical tree. The neighborhood of the particles in HPSO does not need extra computation as the tree structure is fixed. The hierarchical tree has *branching*

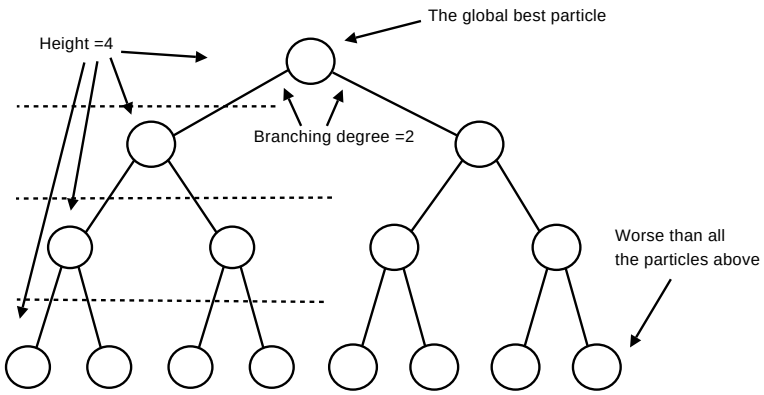


Figure 3.14: Schema of HPSO with $h = 4$ and $d = 2$.

degree d, and height *h*. It is formed in the initialization phase and updated each iteration. The tree is formed in which each particle is better than the particle below it in the tree. Thus, the best particle is the root of the tree. Each new iteration, the tree is updated in which, if the particle in the lower level is improved to be better than the particle above it, they swap their position in the tree. In the implemented version; in order to form the tree, the particles are sorted first according to their fitness value. Then they are assigned to the tree from top to bottom according to their position in the sorted list.

Adaptive Hierarchical Particle Swarm Optimizer

The adaptive hierarchical particle swarm optimizer *(AHPSO)* [JM05] decrease the branching degree during the optimization. In other words, the hierarchy reduce the branching degree from d to $d-1$ every f_{adapt} iterations.

Time-Varying Inertia Weight PSO

In [SE98a, SE99], Shi and Eberhart suggest changing the inertia coefficient with the number of iterations. Experimentally, they found out that the best results are achieved when w starts with 0.9 and decreases linearly to 0.4. The inertia weight can be increasing [SE98a, SE99, SE98b] or decreasing [ZMZQ03b, ZMZQ03a].

Self-Organized Hierarchical Particle Swarm Optimizer with Time Varying Acceleration Coefficients

In [RHW04], Ratnaweera et al. used time-varying acceleration coefficients *(TVAC)*, in which, C_1 and C_2 values are time dependent. The best results were achieved when C_2 start by 0.5 increases linearly to reach 2.5 in the last iteration, and C_1 starts with 2.5 and decreases to 0.5 in the last iteration. In case of no improvement, the velocity can mutate *(MPSO)* by a given mutation probability. The velocity mutates by the maximum allowed velocity after multipling it by the mutation step size. In [RHW04], the mutation step size starts with 1 and decreases linearly to 0.1 in the last iteration. Ratnaweera et

al. has used the time varying acceleration coefficients with PSO *(PSO-TVAC)* in which the inertia coefficient is changing as in [SE98b]. The inertia coefficient is set to zero when it is used with HPSO *(HPSO-TVAC)*. In HPSO-TVAC, if the velocity of any of the dimensions is attenuated to zero, it is reinitialize. More details about TVAC approaches are found in [RHW04].

Fully Informed PSO

The fully informed particle swarm optimizer *(FIPSO)* is proposed by Mends et al. [MKN04], instead of using only best achieved value of only the best particle and the particle i to calculate the velocity of the particle i, all the population is used as shown in equation 3.9, where N is the set of neighbors of the particle i, Φ is the acceleration coefficient, and $W(j)$ is the weighing function of the neighbor j.

$$\mathbf{v}_{t+1,i} = \chi \cdot \left[\mathbf{v}_{t,i} + \sum_{j \in N} \frac{\Phi}{N} \cdot W(j) \cdot \mathbf{rand}() \cdot (\mathbf{p}_{t,j} - \mathbf{x}_{t,i}) \right] \tag{3.9}$$

Gregarious Particle Swarm Optimizer

In the gregarious particle swarm optimizer (G-PSO) [PB06], all particles fly only toward the global best achieved position as in equation 3.10. Any particle very close to the global best is considered as a trapped particle and its velocity is re-initialized, otherwise,its velocity calculated by the different between its speed and the position of the global optimal solution multiplied by a γ.

$$\begin{aligned} &if(\|\mathbf{x}_{t,i} - \mathbf{g}_t\| \leq \varepsilon) \\ &\qquad \mathbf{v}_{t+1,i} = \mathbf{rand}(-V_{max}, V_{max}) \\ &else \\ &\qquad \mathbf{v}_{t+1,i} = \gamma \cdot \mathbf{rand}(0,1) \cdot (\mathbf{x}_{t,i} - \mathbf{g}_t) \end{aligned} \tag{3.10}$$

The value of γ is reduced by σ each iteration the swarm improves to have smaller steps in order to search better, and σ is added to γ if no improvement. There is a range that γ is not allowed go out of it.

Guaranteed Convergence PSO

The guaranteed convergence particle swarm optimizer *(GCPSO)* [dB02,dBE02] was mainly designed in order to avoid premature convergence which can happen when the global best position is equal to the current position. Thus, only the global particle is treated in special way in GCPSO in which its velocity is updated by equation 3.11.

$$\mathbf{v}_{t+1,g} = -\mathbf{x}_{t,g} + w \cdot \mathbf{v}_{t,g} + \rho_t(1 - 2 \cdot \mathbf{rand}()) \tag{3.11}$$

The details of updating ρ_t is in [dB02, dBE02]. However, GCPSO is likely to trap into local optima in multimodal functions [dB02].

Multi-Start PSO

Multi-start PSO is a modified version of the GCPSO to solve its trapping problem. It saves the optimal result when the PSO is trapped and re-initialize the population.

Attractive and Repulsive PSO (ARPSO)

Based on the diversity measurment, the ARPSO [RV02] switches between attraction and repulsion phases. In the attraction phase, ARPSO uses the PSO to allow fast convergence which results in fast diversity reduction. In the repulsion phase, the population flies far away from the best particle, which increases the diversity.

Hybrid Approaches

Some hybrid approaches use genetic algorithm, or any of its operators with particle swarm optimization. Angeline [Ang98] used the tournament selection method with PSO. Among a group of selected particles, the winner particle awarded one point in the tournament. Afterwards, the population particles are sorted according to the number of points each particle awarded. Then the top half of the population, which has high score is copied over the bottom half which has lower score. However, this hybrid approach performed worse than the PSO for multimodal functions as its diversity decreases very fast.

Breeding is introduced by Løvjberg [LRK01]uses arithmetic crossover operator as a breeding operator. A breeding probability is used as the crossover probability in genetic algorithm. The parents are replaced with the offspring. This approach performed better than the PSO with multimodal functions [LRK01].

The life cycle model hybridding PSO with GA and hill-climbing by running them after each other for the same population [Lø02]. It is observed that the original PSO performed well compared to this hybrid approaches.

Random mutation is employed in [XZY02] in order to prevent premature convergence. Gaussian mutation is used instead in [HI03].

Craziness

In each iteration, the position of a few particles is randomized, while the velocity of the rest is reinitialized to the cognitive velocity component [VSS03].

Self-Organized Criticality PSO

In the self-organized criticality particle swarm optimizer *(SOC PSO)*, the particles close to each others are reallocated by a measure of the diversity called criticality [Lø02]. Each particle has its critical

value, which is initialized by zero. If the distance between any two particles is less than a given threshold distance, the critical value of each of them increases by one. The SOC PSO has a global set criticality limit *CL*. If the critical value of any particle exceeds this limit, it disperse its criticality within the surrounding neighbors by increasing the criticality of the first *CL* neighbor particles by one, then decrease its critcality by *CL*. Afterwards, it reallocates itself. The reallocation is achieved by employing a reallocation schemes, e.g., reinitialize the particle [Lø02].

The SOC PSO requires more operators than the original PSO, and its complexity is $O(N^2)$ as the distance between all particles is measured. On the other hand, its basic principle is similar to charged PSO, where charged particles that are closed to each other dispel each other by a regular mathematical formula to keep the diversity.

Fitness-Distance Ratio based PSO

In the fitness-distance ratio based particle swarm optimization (FDR-PSO), a new term is added which gives the PSO particles the ability to fly to their fittest neighbor. The velocity of each particle is updated by equation 3.12.

$$\mathbf{v}_{t+1,i} = w \cdot \mathbf{v}_{t,i} + \psi_1 \cdot \mathbf{rand}() \cdot (\mathbf{p}_{t,i} - \mathbf{x}_{t,i}) + \psi_2 \cdot \mathbf{rand}() \cdot (\mathbf{p}_{t,g} - \mathbf{x}_{t,i}) + \psi_3 \cdot \mathbf{rand}() \cdot (\mathbf{p}_{t,\eta} - \mathbf{x}_{t,i}) \quad (3.12)$$

The particle η is determined by minizing equation 3.13

$$\eta = min \frac{f(\mathbf{x}_{t,i}) - \mathbf{p}_{t,\eta}}{|\mathbf{p}_{t,\eta} - \mathbf{x}_{t,\eta}|} \quad (3.13)$$

Unified PSO

The Unified Particle Swarm Optimization *(UPSO)* [PV04], the PSO with global neighboring is merged with the PSO local neighboring as the following, calculate the global speed update $\mathbf{G}_{t+1,i}$ by equation 3.14, calculate the local speed update $\mathbf{L}_{t+1,i}$ by equation 3.15, and Update the particle speed by equation 3.16, where u is a coefficient for mixing the global and local velocities.

$$\mathbf{G}_{t+1,i} = w \cdot \mathbf{v}_{t,i} + C_1 \cdot \mathbf{rand}() \cdot (\mathbf{p}_{t,i} - \mathbf{x}_{t,i}) + C_2 \cdot \mathbf{rand}() \cdot (\mathbf{p}_{t,g} - \mathbf{x}_{t,i}) \quad (3.14)$$

$$\mathbf{L}_{t+1,i} = w \cdot \mathbf{v}_{t,i} + C_1 \cdot \mathbf{rand}() \cdot (\mathbf{p}_{t,i} - \mathbf{x}_{t,i}) + C_2 \cdot \mathbf{rand}() \cdot (\mathbf{p}_{t,g_i} - \mathbf{x}_{t,i}) \quad (3.15)$$

$$\mathbf{v}_{t+1,i} = (1-u) \cdot \mathbf{L}_{t+1,i} + u \cdot \mathbf{G}_{t+1,i}, u \in [0,1] \quad (3.16)$$

Division of Labor in PSO

Division of labor in particle swarm optimization (DoL PSO) [VRK02] employs the division of labor model in [BTD96]; a stimuli signal s is associated with local optima search, if the stimuli signal s is higher than the threshold θ, the particle can start a local search task by the probability given in equation 3.17. The probability to stop the local search task is a constant.The stimuli s will increase when the fitness does not improve.

$$P_{l \to g,i} = \frac{s^n}{\theta_i^n + s^n} \quad (3.17)$$

where n the steepness factor, the higher n, the probability to switch to local search if s need is slightly higher than the threshold is higher. The threshold θ is updated by equation 3.18, in which, with time, local search is needed. This helps the swarm to search for global best first, then to local best.

$$\theta_i(t) = b \cdot e^{-a \cdot t} \quad a, b \in \mathbb{R}^+ \tag{3.18}$$

Niching Particle Swarm Optimizer

The niching PSO [BEB02] employs the cognition only PSO model [Ken97] such that each particle search for a local solution around its best achieved fitness value.

$$\mathbf{v}_{t+1,i} = w \cdot \mathbf{v}_{t,i} + C_1 \cdot \mathbf{rand}() \cdot (\mathbf{p}_{t,i} - \mathbf{x}_{t,i}) \tag{3.19}$$

Any particle that shows a little improvement over a small number of iteration creates a new subswarm. The closest nieghbor to this particle is added to the sub-swarm. The subswarm o has the radius $r_o = max\|\mathbf{x}_{o,g} - \mathbf{x}_{o,i}\|$, where g is the best particle in the subswarm, and i represents the other subswarm particles. Any particle flies inside the subswarm is attracted to it, which means this particle is added to the sub-swarm. If two subswarm intersect, they merge. The GCPSO is employed in the subswarms, thus, the velocity of the best particle of any subswarm is updated by equation 3.11, while the velocity of the rest of the subswarm particles is updated by equation 3.2.

3.5 Optimization in Dynamic Environment

In many engineering systems, the optimization is done under the assumption that the environment is static, which means that the system at the time t performs with the performance at the time $t + t_0$, which is not true in real systems. As an example, designing an operational amplifier with some specifications, and assume that this amplifier behavior does not change during optimization because of the environmental dynamics. In real systems, the environment is non-stationary, consequently, the system performance can be changed during or after the optimization.

The main three requirements for dynamic environment optimization approach are continuous adaptation, flexibility and robustness. In static environment suitable approachs, most of the individuals converge to around the best potential solution. In case of dynamic environment, the standard approaches can not be efficiently used as most of the individuals can get trapped in nearest local optima after any environmental change. The population diversity must be kept even after finding an optimal solution.

Branke [Bra04] divided the possible remedies for dynamic environment into five types:

1. To reset the population after every environmental change; for example, the evolvable hardware in [HHE02, SKZ+00, SZK01] start the optimization from scratch each time the environment is changed.
2. Generate diversity after detecting environmental changes; for example [Cob90] uses hypermutation for few iterations after detection environmental change with high enough mutation rate in order that

the individuals mutate to far solutions from the local optima. Vavak [VJF97] employed variable local search which increases the mutation gradually after detecting environmental change. The disadvantage of this approach is that it destroys the information because of the randomization.

3. Maintain the population diversity during the run; for example, the charged particles [BB02a] dispel each other to keep the population diversity. According to [Bra04], maintaining the diversity disturbs the optimization.

4. Memory-enhanced EAs; allow storage of good solution and reuse them. it is useful only if the environment is changing periodically. It may slow down convergence and favor diversity [Bra99]. This approach is not useful for the evolvable hardware as the environmental dynamics is not periodic, and allowing the store of all the good solutions during the search may consume the embedded system resources. However, the particle swarm optimization implicitly keeps an optimal solution for each particle.

5. Multi-Population approaches [Bra04]; for example, in multi-swarm particle swarm optimization [BB04], the population is divided into several sub-swarms in which each subswarm converges at a different solution. The diversity in this approach is useful as it is kept at optimal solutions. In addition, one of this solutions can be the global optimal solution after the environmental change [Bra01a].

In general, the best achieved fitness value for each particle is not guaranteed to be valid for the new environment. Carlisle et al. [CD00], proposed a method for adapting particle swarm optimization for dynamic environment. Their method assumes that the $\mathbf{p}$ vector is not useful anymore after any environmental change as it is not guaranteed to be valid. Thus, the $\mathbf{p}$ vectors of all the particles are resetted to the current positions after any environmental change. In Carlisle et al. experimental work, the resetting is done either periodically or after an environmental change. Later on, they used one or more randomly chosen particles to detect the environmental change by reevaluating their fitness function of their $\mathbf{p}$ vectors [CD01]. They called these particles *sentries* . More than a sentry can be employed if increasing the probability of detecting localized changes is required.

In [CD02], they modified their method in which the algorithm reevaluate the fitness at the position best achieved fitness and replace it with the current position only if the fitness of the current position is better. They called this modification Adaptive Particle Swarm Optimizer *(APSO)*

Based on Carlisle et al. approach, the unified particle swarm optimization [PV05], multi-swarm optimization [BB04], and hierarchical particle swarm optimizer [JM04] are modified to dynamic environment.

Eberhart and Shi [ES01] proposed PSO in which, the weight coefficient is randomized each iteration within the range $[0.5, 1.0]$. This PSO change the weight coefficient to improve the search, but the change is not time dependent as in [ES00]. Note that Zheng et al. [ZMZQ03b, ZMZQ03a] showed that increasing the weight coefficient and decreasing it return the same results.

In [CHR$^+$05], Cui et al. assumed that the environment getting changed each iteration, which means that the $\mathbf{p}_i$ vector of the particle i does not have the same fitness value anymore after each iteration. A new term is introduced, which is the evaporating rate T. This term was introduced first in ant colony optimization *(ACO)*. Assigning the achieved fitness value fp is updated by equation 3.20 instead of equation 3.21 for minimization problem. The vector $\mathbf{p}$ is updated according to the current value if the

fitness of the current value is taken as the best achieved fitness value for the particle.

$$f_{i,t+1} = \begin{cases} fp_{i,t} \cdot T & f(\mathbf{x}_{i,t+1}) \leq fp_{i,t} \cdot T \\ f(\mathbf{x}_{i,t+1}) & f(\mathbf{x}_{i,t+1}) > fp_{i,t} \cdot T \end{cases} \tag{3.20}$$

$$f_{i,t+1} = \begin{cases} fp_{i,t} & f(\mathbf{x}_{i,t+1}) \leq fp_{i,t} \\ f(\mathbf{x}_{i,t+1}) & f(\mathbf{x}_{i,t+1}) > fp_{i,t} \end{cases} \tag{3.21}$$

3.6 Proposed Approach: Local Parameters PSO

The state of the art assumes that all the particles are similar to each other in their behavior, but in real world, each particle in the swarm has different behavior, and thus different parameters. For example, a big heavy bird has higher inertia than little light one. This helps the particles to distribute over the space as each of them flies from its current position by a different path. For example, if we assume that two particles have the same **p** and **x** vectors, they will converge to the same point in the standard approach *(under the fact that the mean value of the random number generator should be* 0.5*)*.

In the proposed solution [TK06d], local parameters are employed for each particle, which results in improvement in the diversity during the search, and thus can improve the quality of the solution. Similarly, using local learning rate has been successfully applied to neural networks [MPV02] to improve its effiency.

The velocity of the particle i is computed by equations 3.22 instead of equation 3.2

$$\mathbf{v}_{i,t+1} = w_i \cdot \mathbf{v}_{i,t} + C_{1,i} \cdot \mathbf{rand}() \cdot (\mathbf{p}_{i,t} - \mathbf{x}_{i,t}) + C_{2,i} \cdot \mathbf{rand}() \cdot (\mathbf{p}_{g,t} - \mathbf{x}_{i,t}) \tag{3.22}$$

where w_i, $C_{1,i}$ and $C_{2,i}$ are the local parameters of the particle i. These local parameters are updated after each iteration by a controller in order to improve itself for each particle individually. The maximum velocity can be constrained to be within $\pm\mathbf{v}_{max}$, in the proposed approach. The procedures for the proposed PSO is shown in procedures 3. The original PSO is printed in gray, the modification is printed in black.

The inertia coefficient w_i is initialized by a random number of the range between 0.5 and 1, while $C_{1,i}$ and $C_{2,i}$ are initialized by a random number of the range between 1.75 and 2.5. And these local parameters are updated by a naive algorithm, which is shown in procedures 4, as an example of the parameter controllers. The basic idea of the update algorithm is that; if the fitness value of a particle is improving during flying toward the global best particle, then it flies faster toward the global best particle by increase $C_{2,i}$. If a particle is not improving, it starts to fly faster toward the position of its own best achieved fitness and its velocity mutates with a mutation probability of 0.007 *(after it finishes searching for the local optimum around the global best, it searches for the global best distributed in the search space)*. In addition, as in [RHW04], if the velocity of any dimension of the particle is attenuated to zero, it is reinitialized.
More advanced controller can help the global best particle to search for the local optimum, and can take special care of the parameters after environmental change *(e.g., reinitialize the parameters)*. If more engineering aspects are known about the application like the dynamic frequency, the parameters can be updated according to this knowledge. However, it is not expected that the same controller can work in all the engineering applications efficiently.

Procedure 3 LPSO procedures.

Require: PopulationSize , FitnessFunction
Ensure: Find Optimal Solution

```
{Initialization}
Initialize the random generator with the seed
for i = 1 to N do
    x_i = (2 · rand() − 1) · x_max
    v_i = (2 · rand() − 1) · v_max
    f_i = f(x_i)
    p_i = x_i
    pf_i = f_i
    {C_1,i, C_2,i), w_i} ⇐ InitializeParameters(i)
end for
g ⇐ Identify the particle with the global best
{Learning}
for t = 0 to t_max do
    for i = 1 to N do
        v_i = w_i v_i + C_1,i · rand() · (p_i − x_i) + C_2,i · rand() · (p_g − x_i)
        x_i = x_i + v_i
        {C_1,i, C_2,i, w_i} ⇐ updateParameters(i)
        f_i = f(x_i)
        if f_i < 5 pf_i then
            p_i = x_i
            pf_i = f_i
        end if
    end for
    g ⇐ Identify the particle with the global best
    if TerminationCondition==TRUE then
        break
    end if
end for
```

3.6.1 Experimental Setup

As the particle swarm optimization has low sensitivity to the population size [vdBE01], a small population of 20 particles is employed. The benchmark problems in subsection 3.6.2 are employed in the comparison with state of the art. As the parameters of the proposed approach is changing dynamically, the approach are compared with the other approaches which tunes the acceleration coefficients of the particle swarm optimization [RHW04], which are; HPSO-TVAC, MPSO-TVAC, PSO-TVAC. Two variations are tested with the proposed local parameters approach; local parameters particle swarm optimizer (LPSO), and local parameters hierarchical particle swarm optimizer (LHPSO). In [RHW04], the time variant acceleration coefficient approaches best performance when the algorithm runs for 5000 iterations, thus in the experimental work, running the algorithm for 5000 is chosen. The various PSO approaches are implemented in C++. Each of the approaches is tested for 30 runs with different random number generator seeds, then the mean value and the standard deviation of the 30 runs are calculated and plotted. The performance of the approaches is investigated in static and dynamic

Procedure 4 Procedures to update the local parameters of particle i in LPSO.

Require: PopulationSize , FitnessFunction
Ensure: Find Optimal Solution {Parameters update}
  if $BestFitnessIteration_i == t$ **then**
    $C_{2,i} \Leftarrow C_{2,i} \cdot \alpha_1$
    if $C_{2,i} > 3$ **then**
      $C_{2,i} \Leftarrow 2.5$
      $w_i \Leftarrow w_i \cdot \beta_1$
    end if
  else
    $C_{2,i} \Leftarrow C_{2,i} \cdot \alpha_2$
    if $C_{2,i} < 0.5$ **then**
      $C_{2,i} \Leftarrow 0.7$
      $w_i \Leftarrow w_i \cdot \beta_2$
      if $rand_3 < MutationProbability$ **then**
        $\mathbf{v}_i \Leftarrow Mutate(\mathbf{v}_i, MutationStep)$
      end if
    end if
  end if
  return $\{C_{1,i}, C_{2,i}, w_i\}$

environments.

The dynamic environment is modelled by using change severity is 0.2 *(20%)* with dynamic frequency [BS02, Bra01b] of 20 iteration, in which the position of any particle deviates by $\mathbf{x} = \mathbf{x} + \mathbf{x}_{dev}$, where $\mathbf{x}_{dev}$ deviates every 20 iterations by $\mathbf{x}_{dev} = \mathbf{x}_{dev} + 0.2 \cdot \mathbf{rand}() \cdot \mathbf{x}_{max}$.

It was found that the following parameters are applicable to most of the functions; $\alpha_1 = 1.05$, $\alpha_2 = 0.9750$ and $\beta_2 = 1/1.2$ for both LPSO and LHPSO, while $\beta_1 = 1.2$ for LHPSO and $\beta_1 = 1.07$ for LPSO. However, the controller parameters are chosen by some testing in static environment, more adjustments of these parameters according to the application can improve the search.

The sensitivity of the parameters is investigated by varying them randomly by a maximum of $\pm 10\%$ of their original value each run. This approach is applied to the start and the end values of reinitialization step size , w, C_1 and C_2 , the mutation probability and the mutation step size for the time variant acceleration coefficient PSO approaches. On the other hand, α_1, α_2, β_1 and β_2 are multiplicative coefficients that represent the rate of changes each iteration, $\alpha_1 = 1.05$ means rate of change of 5% each iteration, thus the variation of this coefficients is applied to the change rates, e.g. $\alpha_{1_d} = 1 + (\alpha_1 - 1) \cdot (1 + 0.1 \cdot (2 \cdot rand() - 1))$, where α_{1_d} is the coefficient α_1 after variation; first one is subtracted from the coefficient to prepare the change rate, the result of the subtraction is varied by 10%, then add a one to the coefficient. The mean value and the standard deviation of 900 runs with different parameter values are observed to find out the sensitivity of the parameters to each approach. The parameters are changed every 30 runs.

3.6.2 Benchmark Test Functions

The benchmark test functions used in [SE98a, Ang98, Sug99, SE99, Ken00, ES00, LRK01, SE01, vdBE01, BB02b, RHW04, KFFT04] are employed to compare the proposed approach with the state of the art. These test functions are classified into two classes; unimodal functions, which have only one peak, and multimodal functions which have more than a peak. In [SE98a, Ang98, Sug99, SE99, Ken00, ES00, LRK01, SE01, vdBE01, BB02b, RHW04, KFFT04], the sphere function is employed as unimodal function, while Schaffer, Rastrigin, Rosenbrock, and Griewangk is utilized as multimodal functions. Axis parallel hyper ellipsoid function is combined with the benchmark as an additional unimodal function.

Sphere Model

The sphere function [Rec73a, Jon75] in equation 3.23 for D-dimensions has only global minimum solution at $\mathbf{x} = \mathbf{0}$, it does not have any local minimum.

$$F(\mathbf{x}) = \sum_{d=1}^{D} x_d^2 \tag{3.23}$$

The search domain is $x_d \in [5.12, 5.12]$. In figure 3.15, the function fitness value is shown in two dimensions.

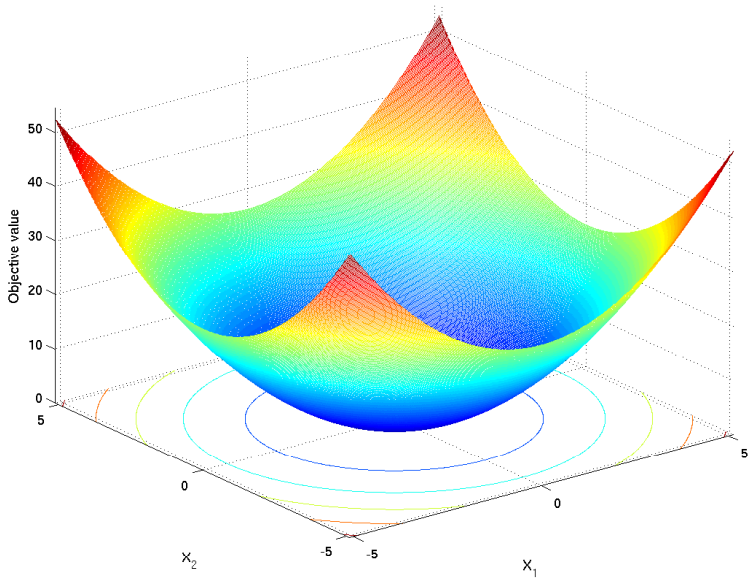


Figure 3.15: Sphere function in 2D.

Axis Parallel Hyper-Ellipsoid Function

The equation of the axis parallel hyper-ellipsoid function is given in equation 3.24. It has only one minimum solution at $\mathbf{x} = \mathbf{0}$.

$$F(\mathbf{x}) = \sum_{d=1}^{D} i \cdot x_d^2 \tag{3.24}$$

The plot of two-dimensions ellipsoid function is shown in figure 3.16 the search space range is $x_d \in [-5.12, 5.12]$. The global minimum solution of this problem is located at $\mathbf{x} = \mathbf{0}$

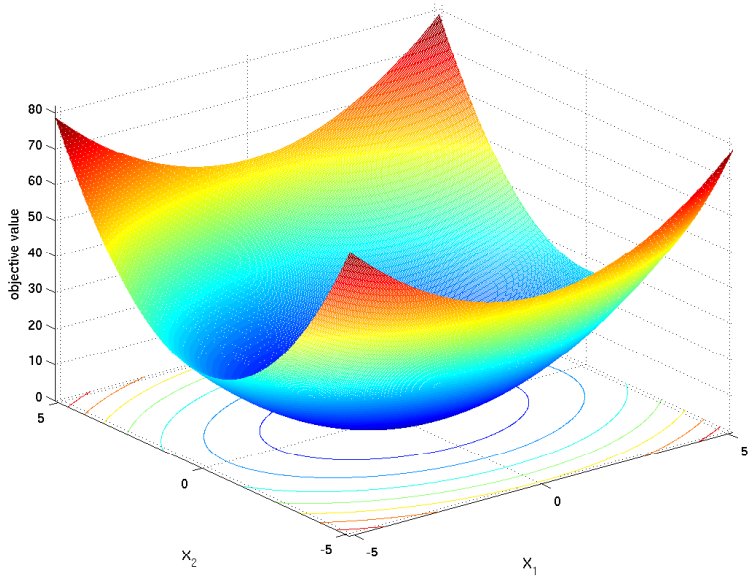


Figure 3.16: Two-dimensions ellipsoid function.

Schaffer's f_6 Function

The Schaffer's f_6 is mainly a two dimension function which is defined as maximization problem in equation 3.25 with maximum point at $x = 0, y = 0$. However, it is used as a minimization problem [Mis06] as in equation 3.26 with minimum point at $x = 0, y = 0$.

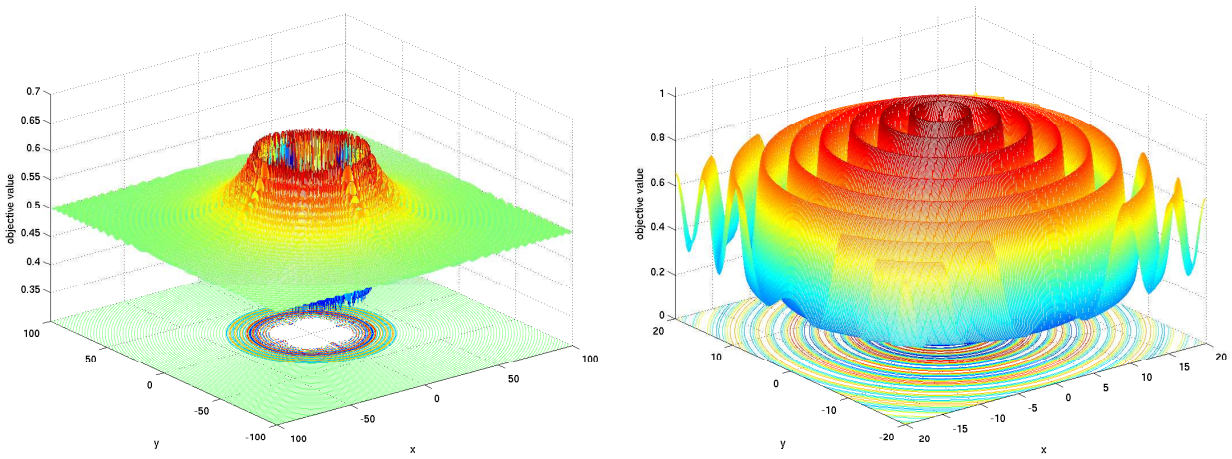
$$F(x,y) = 0.5 - \frac{sin^2\left[x^2+y^2\right] - 0.5}{[1+0.001\cdot(x^2+y^2)]} \tag{3.25}$$

$$F(x,y) = 0.5 + \frac{sin^2\left[x^2+y^2\right] - 0.5}{[1+0.001\cdot(x^2+y^2)]} \tag{3.26}$$

Its search range is $(x,y) \in [-100, 100]$. In figure 3.17, schaffer function is shown.

Rastrigin Function

The Rastrigin function is introduced in [TZ89] as a two dimensions function, and is generalized in [MSB91] to D-dimensions. The search range of $x_d \in [-5.12, 5.12]$. The equation of Rastrigin



(a) (b)

Figure 3.17: Schaffer function. a) The whole search range. b) Zoomed to the range $x_d \in [-20, 20]$.

is shown in equation 3.27, while, the plot of the Rastrigin function fitness value in two dimensions search space is shown in figure 3.18.

$$F(\mathbf{x}) = 10 \cdot D + \sum_{d=1}^{D} x_d^2 - 10 \cdot \cos(2 \cdot \pi \cdot x_d) \tag{3.27}$$

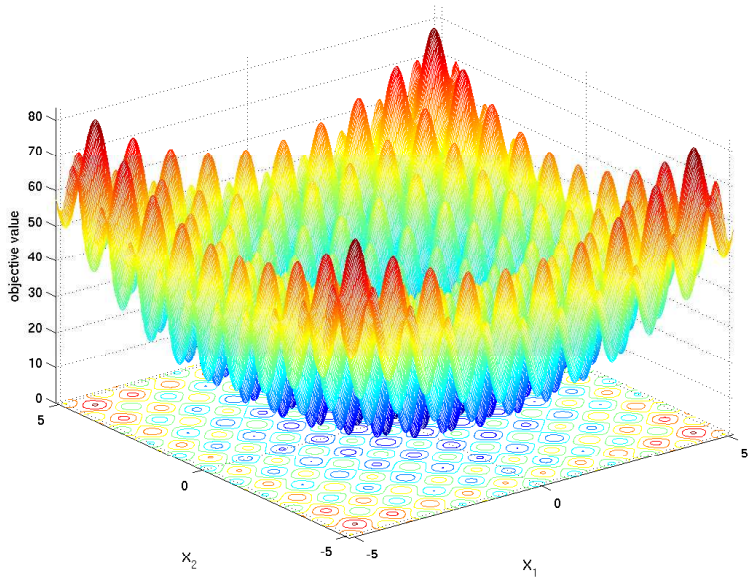


Figure 3.18: Rastrigin function in 2D.

Rosenbrock Function

The equation of Rosenbrock function [Ros60] is shown in equation 3.28. Its search space range is $x_d \in [-2.048, 2.048]$. The Rosenbrock function plot for two dimensions search space is shown in

figure 3.19.

$$F(\mathbf{x}) = \sum_{d=1}^{D-1} (100(x_{d+1} - x_d^2)^2 + (x_d - 1)^2)) \tag{3.28}$$

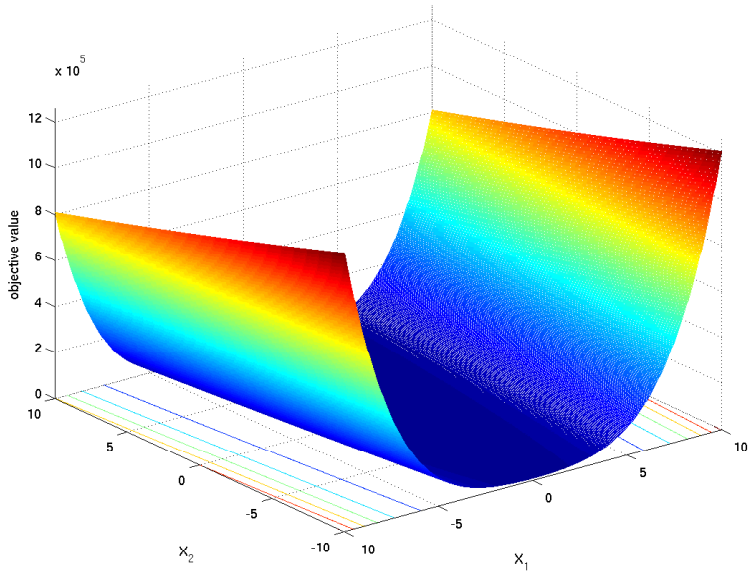


Figure 3.19: Rosenbrock function in 2D.

Griewangk Function

The equation of the Griewangk function [TP89] is shown in equation figure 3.29. The search space range of the function is $x_d \in [-600, 600]$. In figure 3.20, plot of the function in two dimensions search space is shown.

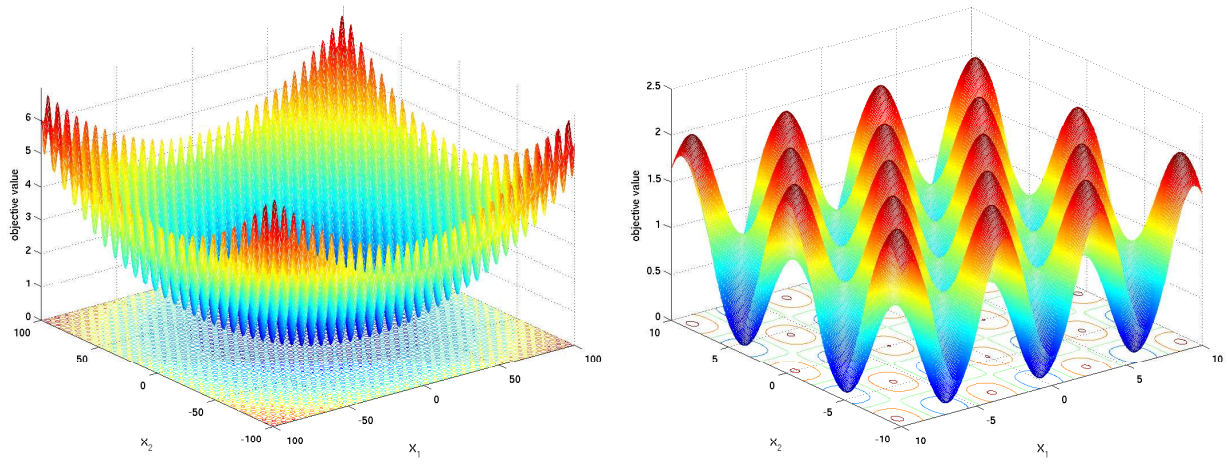
$$F(\mathbf{x}) = 1 + \sum_{d=1}^{D} (\frac{x_d^2}{4000}) - \prod_{d=1}^{D} \cos(\frac{x_d}{\sqrt{d}}) \tag{3.29}$$

3.6.3 Results

The convergence curve of the mean value of the runs and the standard deviation curve in static environment of the sphere, axis parallel hyper ellipsoid, Schaffer, Rastrigin, Rosenbrock, and Griewangk are shown in figure 3.21, 3.22, 3.23, 3.24, 3.25, and 3.26 respectively.

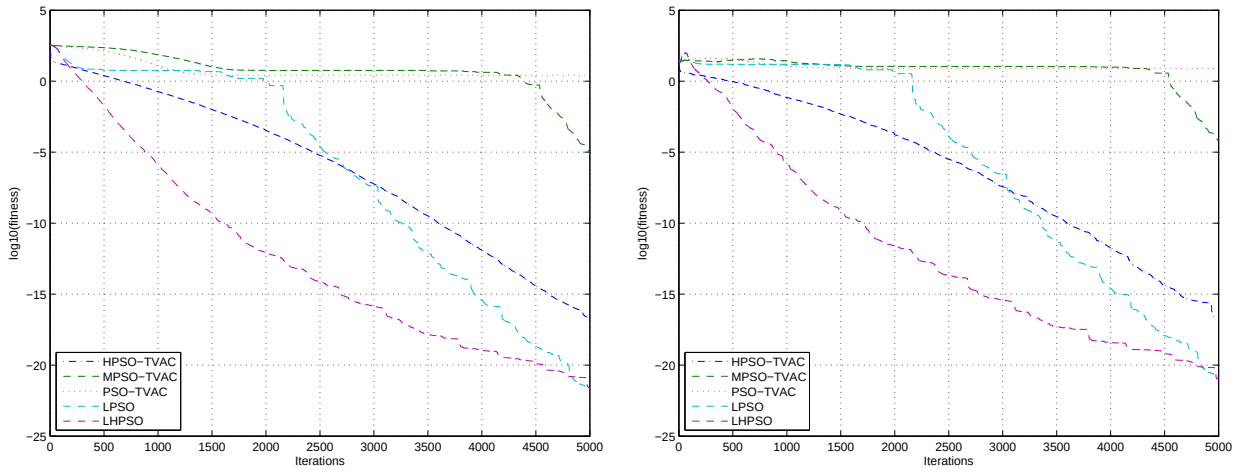
In table 3.1, the mean value of the last iteration of the optimization in static environment is shown, while, its standard deviation is shown in table 3.2. The best value for each function is printed in boldface font.

The mean value and the standard deviation of the local parameter approaches in static environment overcome the state of the art in all the functions except the schaffer function, where, only the MPSO converges to zero.



(a) (b)

Figure 3.20: Griewangk function in 2D. a) The search range $\mathbf{x} \in [-100, 100]$. b) Zoomed to the search range $x_d \in [-10, 10]$.



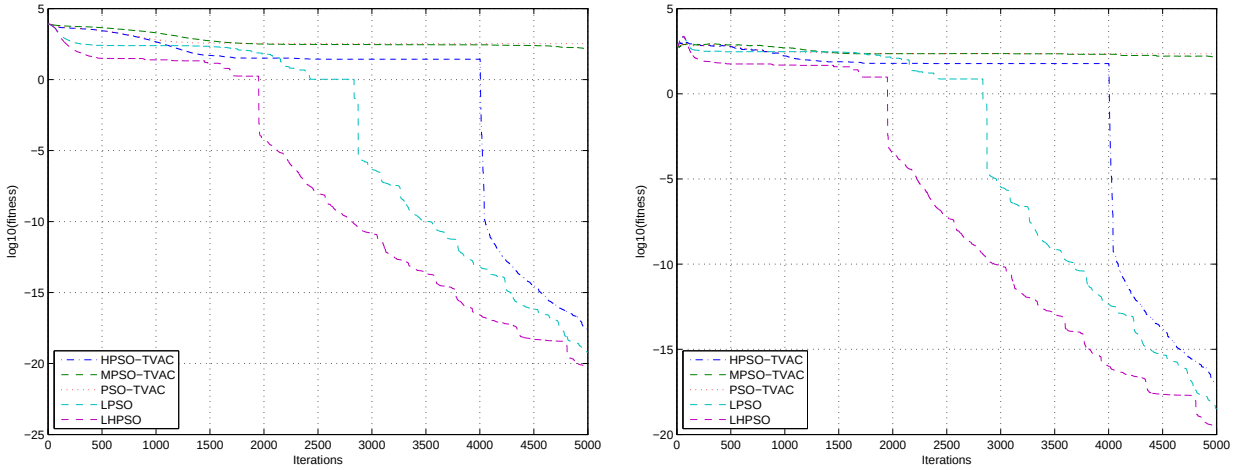
(a) (b)

Figure 3.21: Sphere function. a) The convergence curve of the mean value. b) Standard deviation curve.

Table 3.1: The mean value of the last iteration in static environment.

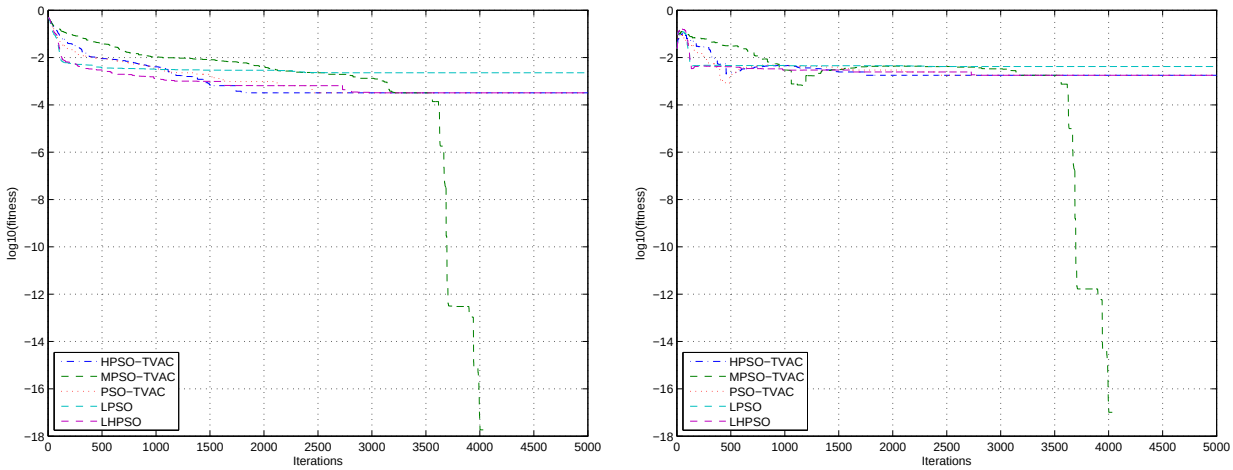
	State of the art [RHW04]			Proposed PSO [TK06d]	
	HPSO-TVAC	MPSO-TVAC	PSO-TVAC	LPSO	LHPSO
Sphere	$1.933 \cdot 10^{-17}$	$9.66 \cdot 10^{-6}$	2.62262	$\mathbf{2.179 \cdot 10^{-22}}$	$2.544 \cdot 10^{-22}$
Hyper ellipsoid	$2.425 \cdot 10^{-18}$	149.722	341.85	$5.884 \cdot 10^{-20}$	$\mathbf{6.581 \cdot 10^{-21}}$
Schaffer	$3.239 \cdot 10^{-4}$	**0**	$3.239 \cdot 10^{-4}$	$2.267 \cdot 10^{-3}$	$3.239 \cdot 10^{-4}$
Rastrigin	143.43	263.636	281.142	166.896	**94.6978**
Rosenbrock	547.825	$6.0733 \cdot 10^{4}$	$3.107 \cdot 10^{4}$	**92.6825**	273.831
Griewangk	0.0191485	0.0383698	10.9467	0.0249028	**0.015994**

The convergence curve of the mean value of the runs in dynamic environment –due to the dynamic environment model described in page 46– of the sphere, axis parallel hyper ellipsoid, Schaffer, Rastrigin,



(a) (b)

Figure 3.22: Axis Parallel Hyper ellipsoid function. a) The convergence curve of the mean value. b) Standard deviation curve.



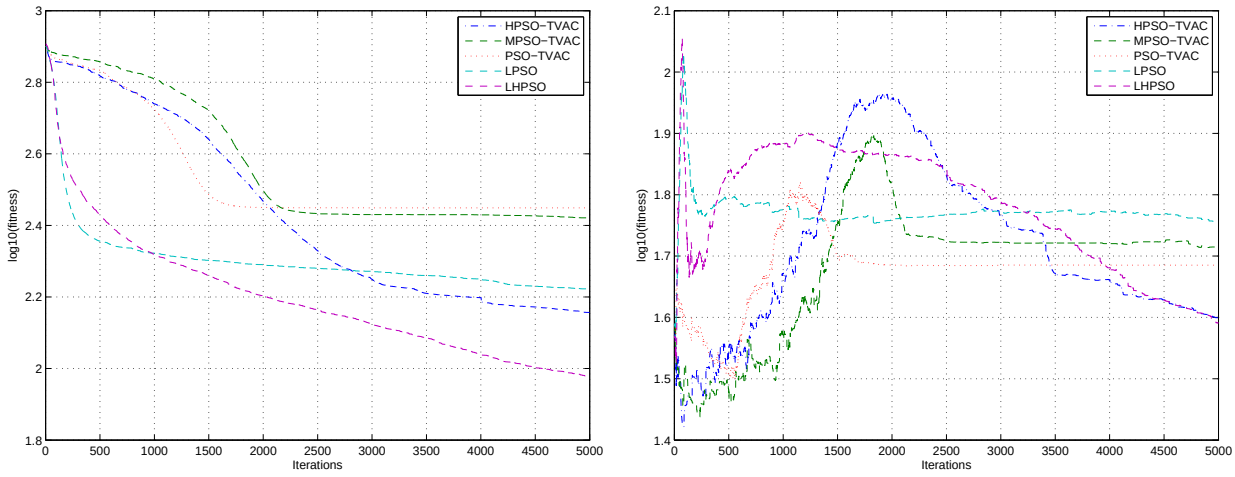
(a) (b)

Figure 3.23: Generalized Schaffer function. a) The convergence curve of the mean value. b) Standard deviation curve.

Table 3.2: The standard deviation of the last iteration in static environment.

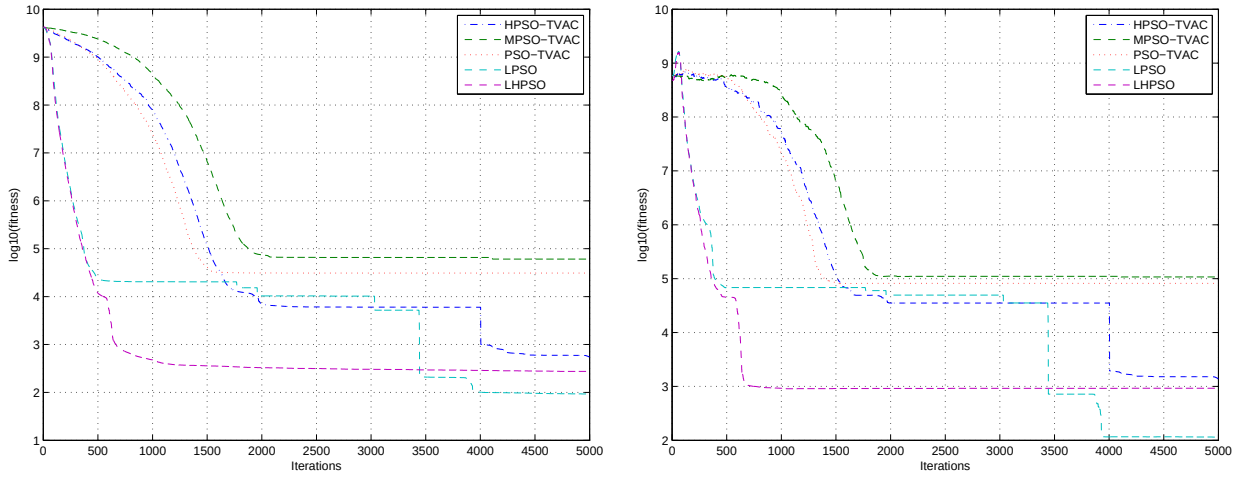
	State of the art [RHW04]			Proposed PSO [TK06d]	
	HPSO-TVAC	MPSO-TVAC	PSO-TVAC	LPSO	LHPSO
Sphere	$1.951 \cdot 10^{-17}$	$6.1155 \cdot 10^{-5}$	7.9439	$1.46 \cdot 10^{-21}$	$\mathbf{1.02 \cdot 10^{-21}}$
Hyper ellipsoid	$9.793 \cdot 10^{-18}$	148.09	226.18	$3.11 \cdot 10^{-19}$	$\mathbf{3.260 \cdot 10^{-20}}$
Schaffer	0.0017739	**0**	0.0017739	0.0041796	0.0017739
Rastrigin	39.766	51.766	48.43	56.993	**38.996**
Rosenbrock	1387.6	$1.0751 \cdot 10^{5}$	81757	**114.77**	924.79
griewangk	0.03027	0.05377	29.666	0.030546	**0.025164**

Rosenbrock, and Griewangk are shown in figures 3.27, 3.28, 3.29, 3.30, 3.31, and 3.32 respectively. The whole convergence curves are shown in figures 3.27(a), 3.28(a), 3.29(a), 3.30(a), 3.31(a), and



(a) (b)

Figure 3.24: Rastrigin function. a) The convergence curve of the mean value. b) Standard deviation curve.

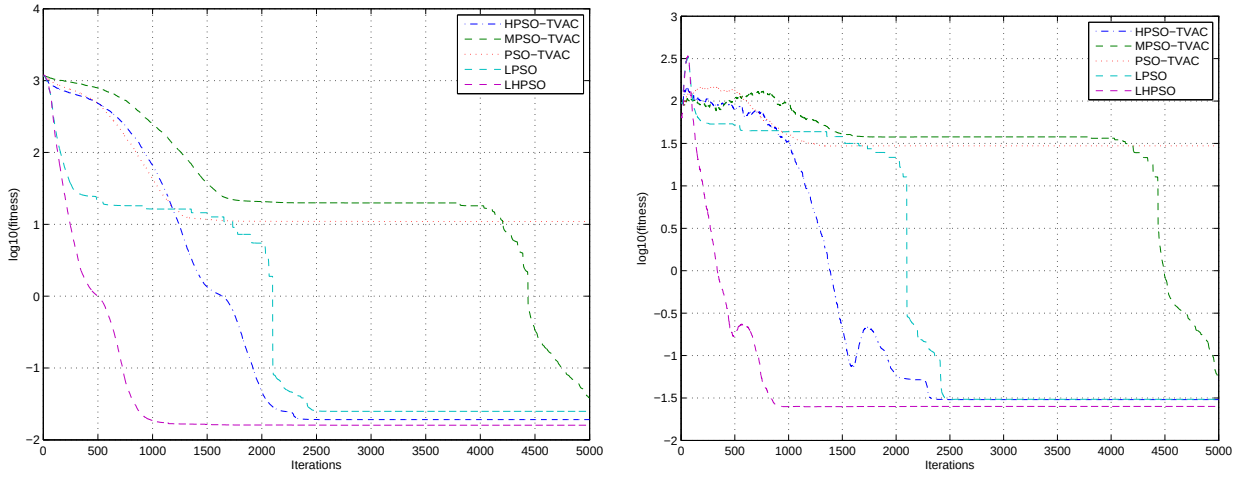


(a) (b)

Figure 3.25: Rosenbrock function. a) The convergence curve of the mean value. b) Standard deviation curve.

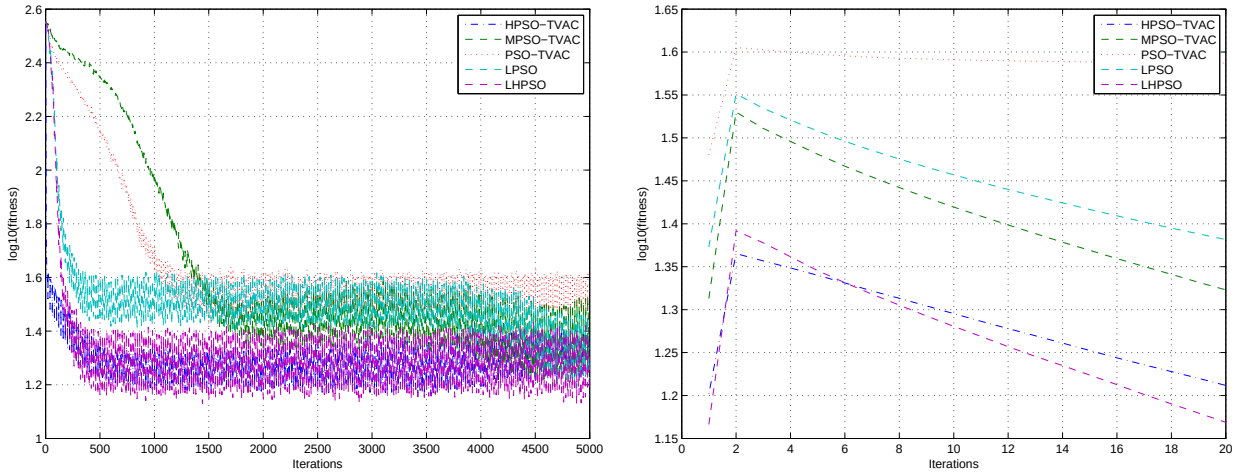
3.32(a) respectively. Starting from the middle of the optimization *(iteration number 2500)*, the mean value of each 20 iteration of all the runs are plotted in the figures 3.27(b), 3.28(b), 3.29(b), 3.30(b), 3.31(b), and 3.32(b) respectively. The first 2500 iterations are excluded from taking the average every 20 iterations as they represent convergence speed rather than the stability of the algorithm in dynamic environment.

The standard deviation curve of the sphere, axis parallel hyper ellipsoid, Schaffer, Rastrigin, Rosenbrock, and Griewangk are shown in figure 3.33, 3.34, 3.35, 3.36, 3.37, and 3.38 respectively. The convergence curves of their whole curves are shown in figures 3.33(a), 3.34(a), 3.35(a), 3.36(a), 3.37(a), and 3.38(a) respectively. Starting from the middle of the optimization, the mean value of each 20 iterations of their standard deviation value are plotted in the figures 3.33(b), 3.34(b), 3.35(b), 3.36(b), 3.37(b), and 3.38(b).



(a) (b)

Figure 3.26: Griewangk function. a) The convergence curve of the mean value. b) Standard deviation curve.

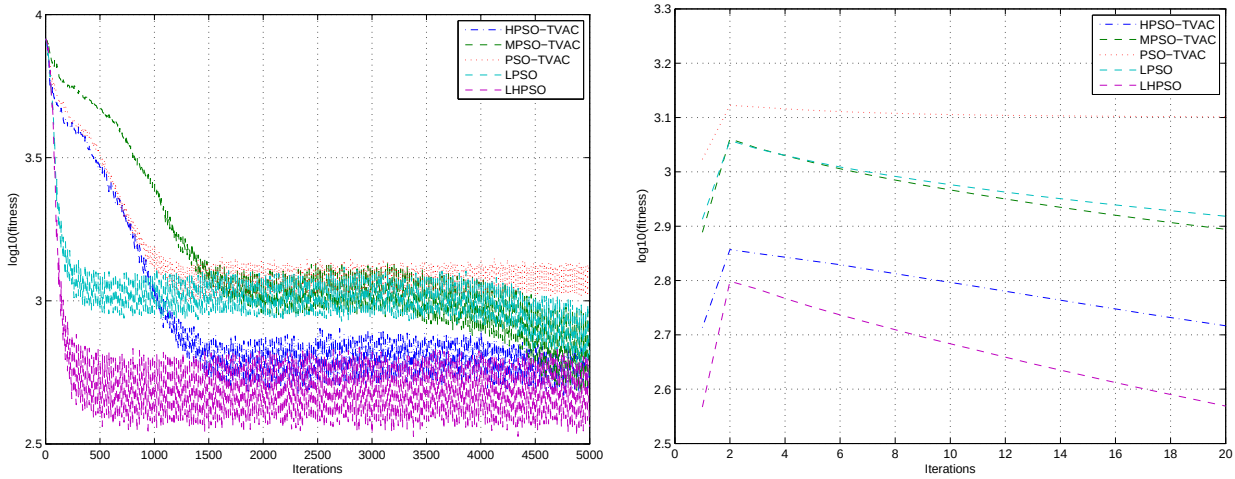


(a) (b)

Figure 3.27: The convergence curve of the sphere function in dynamic environment. a) The whole curve. b) The mean value of every 20 iterations.

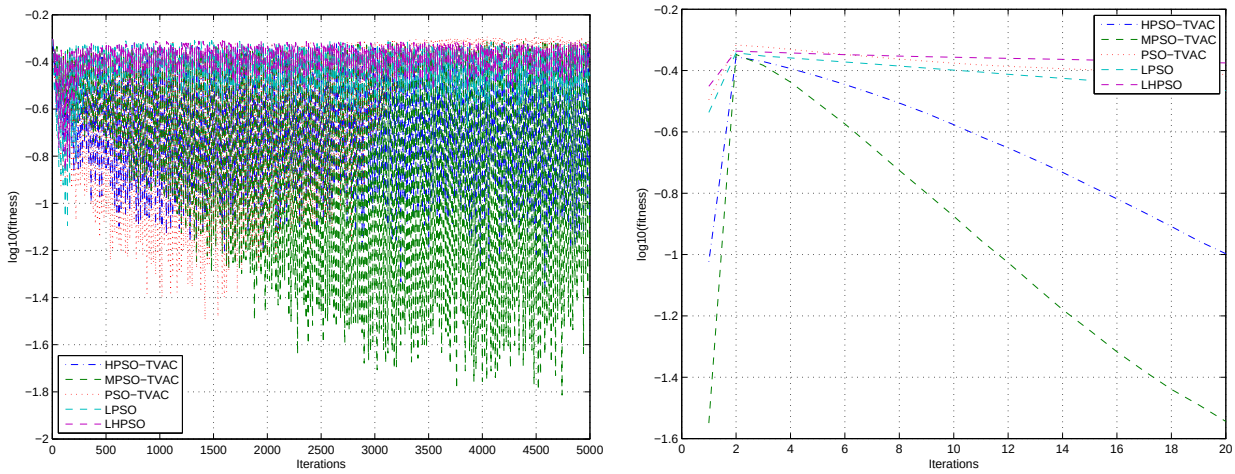
The mean value and the standard deviation of the last iteration of the optimization in dynamic environment are shown in tables 3.3 and 3.4 respectively.

As shown in the curves, the local parameter approaches delivers better results than the other approaches in all the cases except the Schaffer function. The standard deviations of the LPSO and LHPSO in table 3.4 are better than that of the other approaches as well. As shown in the figures 3.33(a), 3.36(a), and 3.38(a); the standard deviation of the local parameter approaches is high at the beginning, then it converged to lower value, hence the average of the standard deviation is higher in the local parameters approaches in figure 3.33(b), 3.36(b), and 3.38(b), and thus in the table as well. The thickness of the convergence and the standard deviation curves represents the influence of the deviation on them. The lines appears thicker when the curve converge more as the results are displayed in log scale. As the Schaffer function has many oscillations in its surface as shown in figure 3.17, the



(a) (b)

Figure 3.28: The convergence curve of the Axis parallel hyper ellipsoid function in dynamic environment. a) The whole curve. b) The mean value of every 20 iterations.

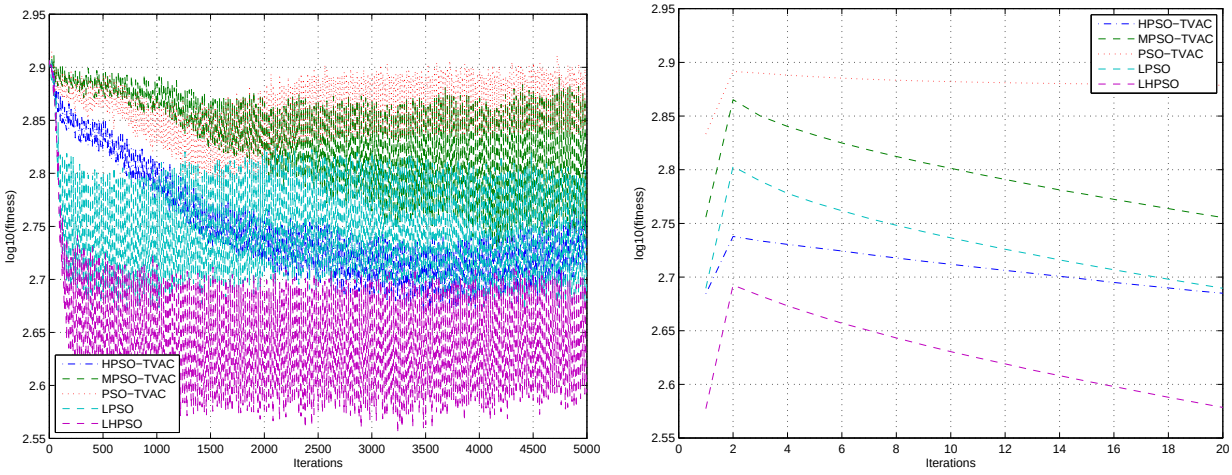


(a) (b)

Figure 3.29: The convergence curve of the Schaffer function in dynamic environment. a) The whole curve. b) The mean value of every 20 iterations.

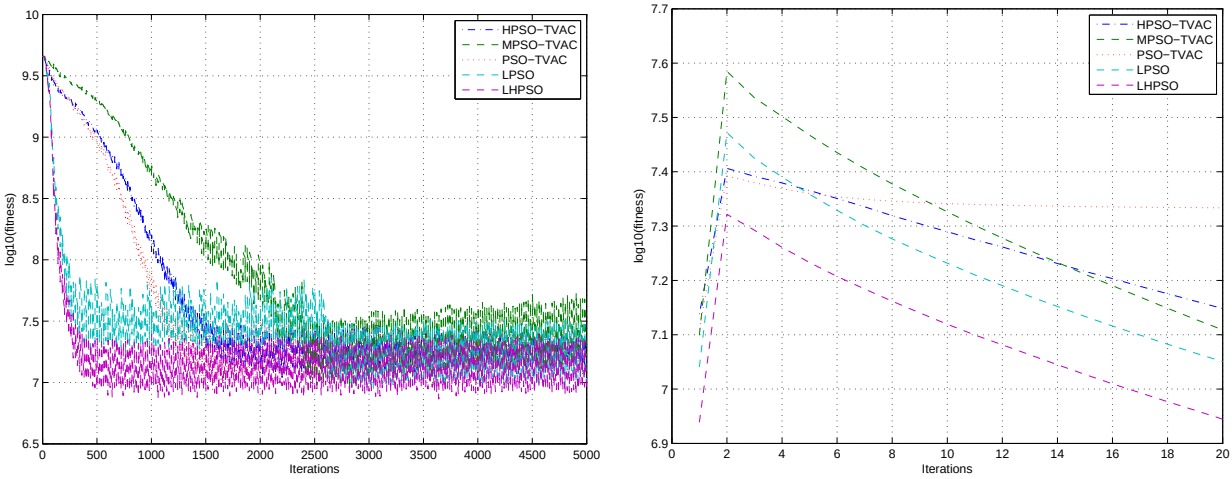
Table 3.3: The mean value of the last iteration before the environmental change in dynamic environment.

	State of the art [RHW04]			Proposed PSO [TK06d]	
	HPSO-TVAC	MPSO-TVAC	PSO-TVAC	LPSO	LHPSO
Sphere	16.28	21.04	38.65	24.08	**14.75**
Hyper ellipsoid	520.81	783.92	1262.4	828.63	**370.52**
Schaffer	0.10067	**0.02856**	0.3865	0.34245	0.42165
Rastrigin	484.14	569.41	756.33	489.32	**378.95**
Rosenbrock	$1.41 \cdot 10^7$	$1.291 \cdot 10^7$	$2.16 \cdot 10^7$	$1.12 \cdot 10^7$	$\mathbf{8.8 \cdot 10^6}$
Griewangk	62.26	73.23	133.7	83.42	**51.53**



(a) (b)

Figure 3.30: The convergence curve of the Rastrigin function in dynamic environment. a) The whole curve. b) The mean value of every 20 iterations.

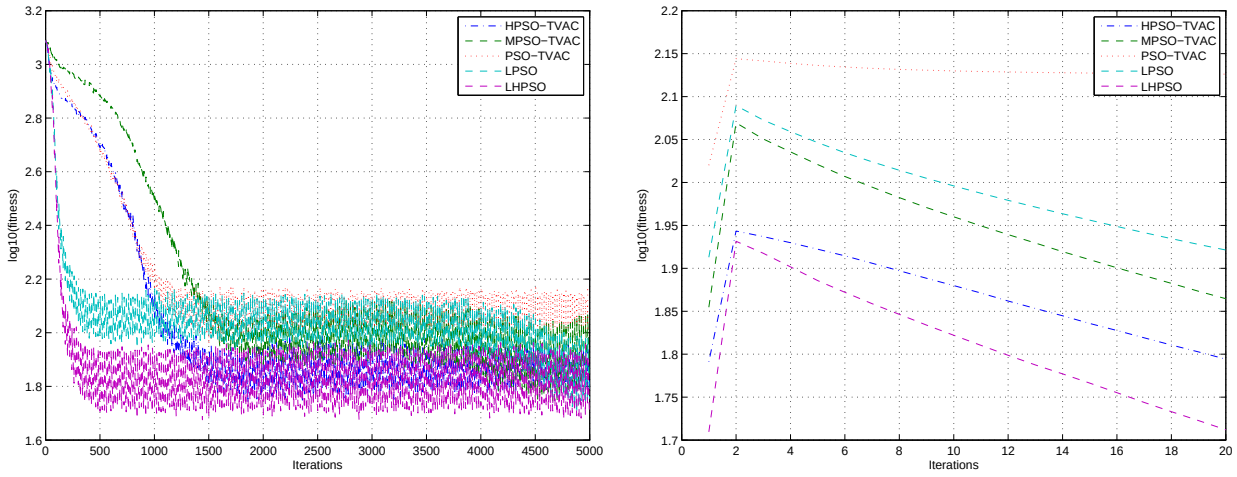


(a) (b)

Figure 3.31: The convergence curve of the Rosenbrock function in dynamic environment. a) The whole curve. b) The mean value of every 20 iterations.

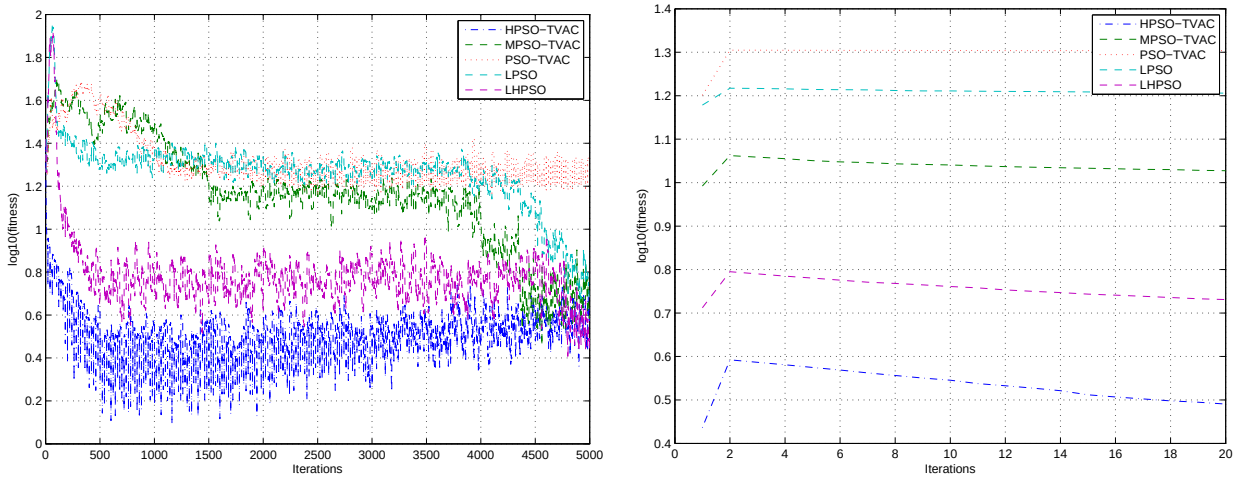
Table 3.4: The standard deviation of the last iteration in dynamic environment.

	State of the art [RHW04]			Proposed PSO [TK06d]	
	HPSO-TVAC	MPSO-TVAC	PSO-TVAC	LPSO	LHPSO
Sphere	**3.0943**	10.654	20.124	16.061	5.3809
Hyper ellipsoid	135.31	332.26	432.86	443.32	**116.82**
Schaffer	0.1227	**0.0381**	0.1064	0.1782	0.127
Rastrigin	**39.332**	63.536	76.768	63.663	42.269
Rosenbrock	$5.2 \cdot 10^6$	$5.97 \cdot 10^6$	$7.36 \cdot 10^6$	$7.9 \cdot 10^6$	$\mathbf{3.36 \cdot 10^6}$
Griewangk	**11.754**	36.577	69.09	54.16	18.37



(a) (b)

Figure 3.32: The convergence curve of the Griewangk function in dynamic environment. a) The whole curve. b) The mean value of every 20 iterations.



(a) (b)

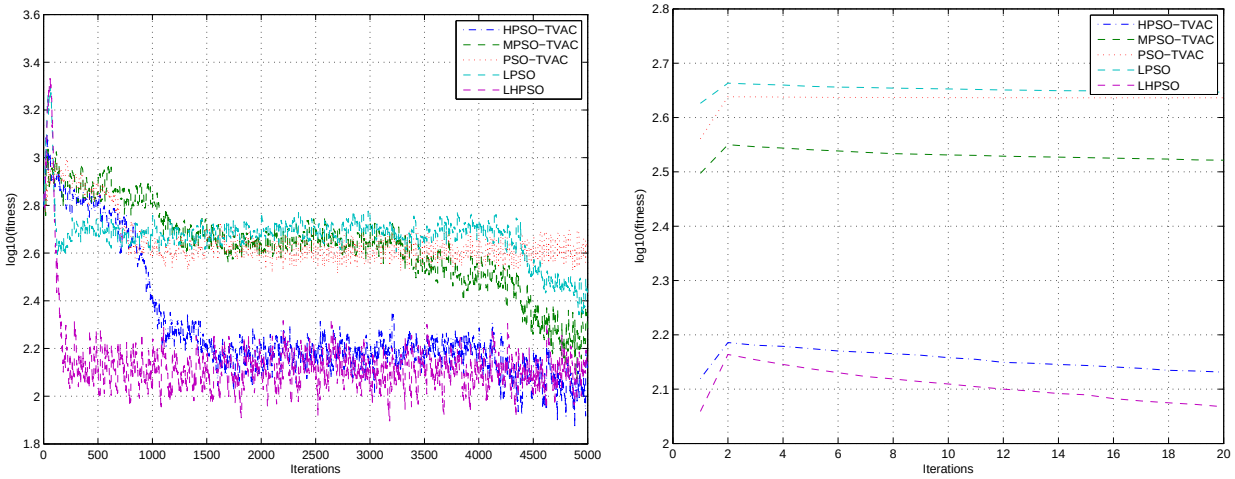
Figure 3.33: The standard deviation curve of the sphere function in dynamic environment. a) The whole curve. b) The mean value of every 20 iterations.

variations in the Schaffer function results in strong change of the returned value as shown in figures 3.29 and 3.35.

Although a naive parameters controller is employed to investigate using local parameters in particle swarm optimization, and the dynamic environment is not considered (e.g., by resetting the parameters as any environmental change), it succeeds to deal with the environmental dynamics with high frequency (an environmental change every 20 iterations).

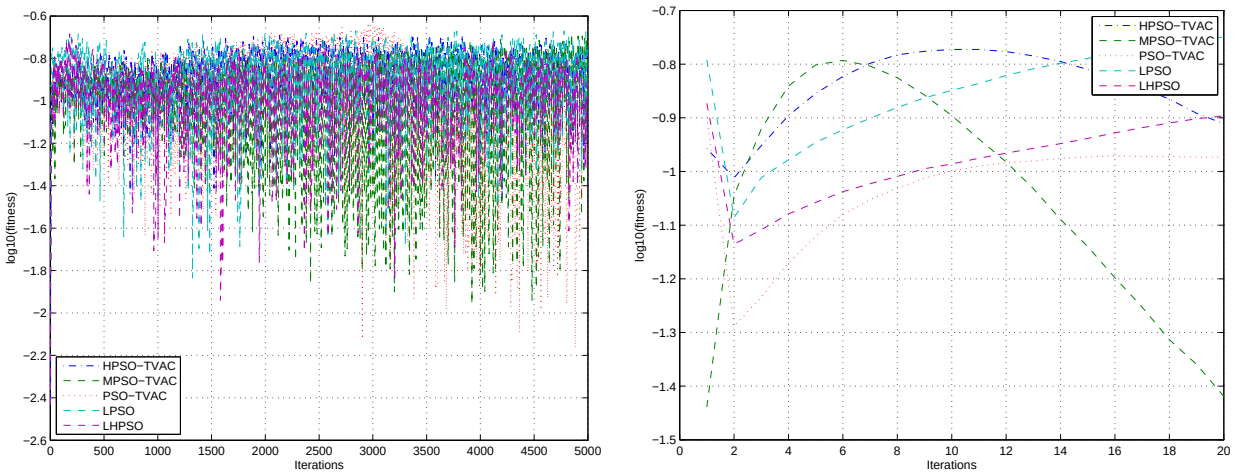
The sensitivity analysis mean values and standard deviations of the 100 runs last iterations are shown in table 3.5 and 3.6.

The starting values of C_1, C_2, and w are initialized randomly in the local parameter particle swarm optimization, thus it is not sensitive to the starting value of the inertia and acceleration coefficients.



(a) (b)

Figure 3.34: The standard deviation curve of the Axis parallel hyper ellipsoid function in dynamic environment. a) The whole curve. b) The mean value of every 20 iterations.

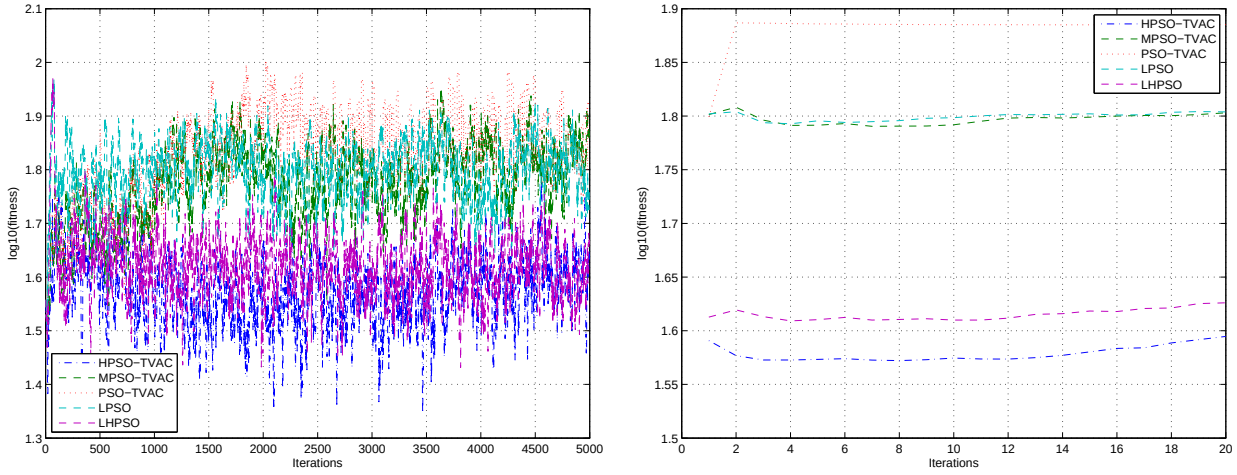


(a) (b)

Figure 3.35: The standard deviation curve of the Schaffer function in dynamic environment. a) The whole curve. b) The mean value of every 20 iterations.

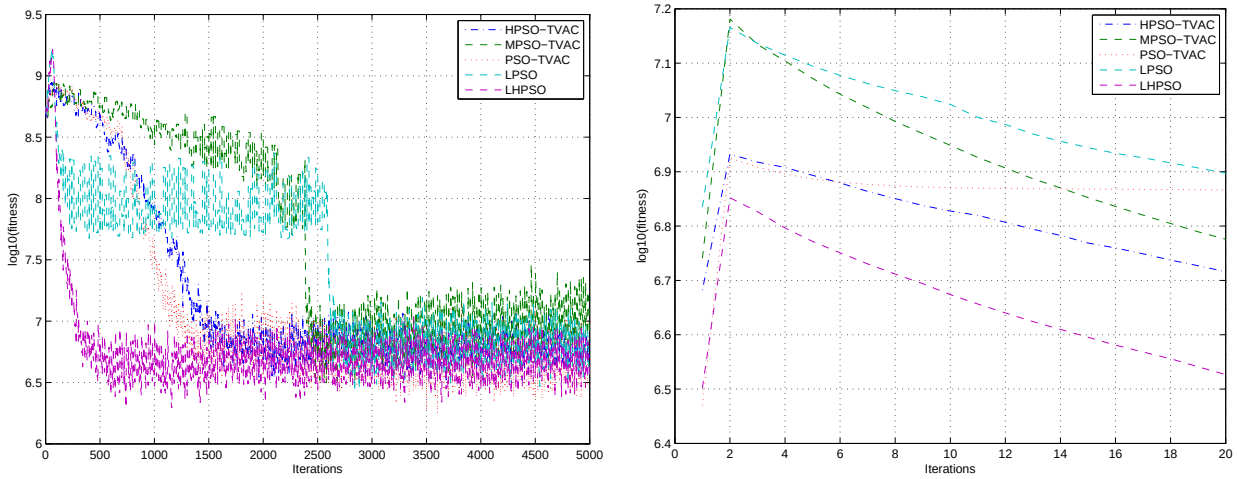
Table 3.5: The mean value of the last iteration sensitivity analysis.

	State of the art [RHW04]			Proposed PSO [TK06d]	
	HPSO-TVAC	MPSO-TVAC	PSO-TVAC	LPSO	LHPSO
Sphere	$2.825 \cdot 10^{-17}$	$5.889 \cdot 10^{-3}$	5.483	$3.54 \cdot 10^{-21}$	$\mathbf{5.867 \cdot 10^{-22}}$
Hyper ellipsoid	$8.727 \cdot 10^{-14}$	113.627	263.122	$1.420 \cdot 10^{-13}$	$\mathbf{2.347 \cdot 10^{-16}}$
Schaffer	$3.239 \cdot 10^{-4}$	**0.0**	$3.239 \cdot 10^{-4}$	$2.267 \cdot 10^{-3}$	$3.239 \cdot 10^{-4}$
Rastrigin	142.385	249.319	250.283	169.369	**94.534**
Rosenbrock	**196.265**	$3.64461 \cdot 10^{4}$	$2.8064 \cdot 10^{6}$	594.9	206.962
Griewangk	$2.387 \cdot 10^{-2}$	0.10001	19.442	$3.575 \cdot 10^{-2}$	$\mathbf{1.814 \cdot 10^{-2}}$



(a) (b)

Figure 3.36: The standard deviation curve of the Rastrigin function in dynamic environment. a) The whole curve. b) The mean value of every 20 iterations.

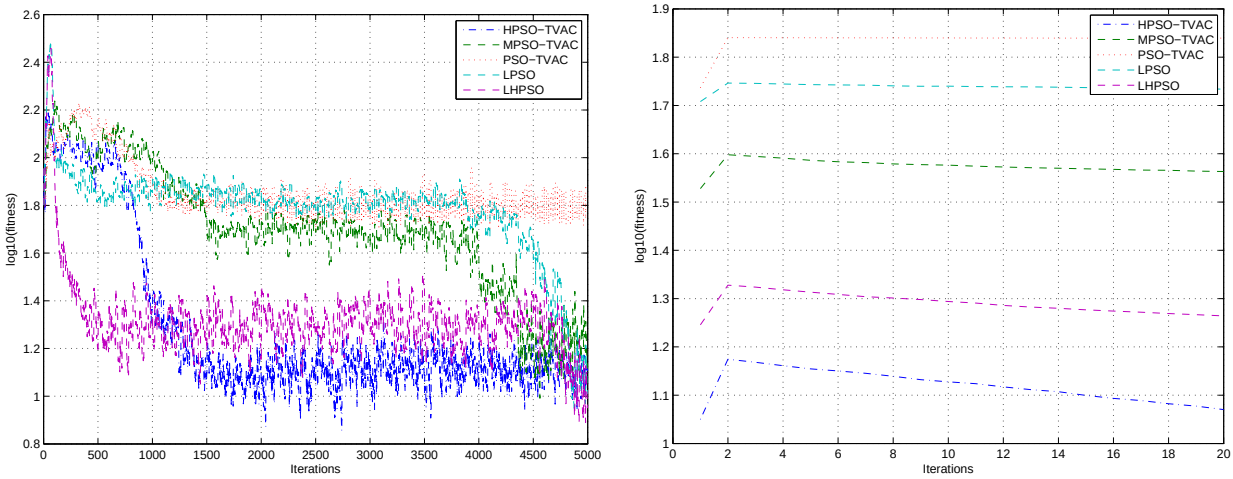


(a) (b)

Figure 3.37: The standard deviation curve of the Rosenbrock function in dynamic environment. a) The whole curve. b) The mean value of every 20 iterations.

Table 3.6: The standard deviation of the last iteration sensitivity analysis.

	State of the art [RHW04]			Proposed PSO [TK06d]	
	HPSO-TVAC	MPSO-TVAC	PSO-TVAC	LPSO	LHPSO
Sphere	$5.462 \cdot 10^{-17}$	$7.752 \cdot 10^{-2}$	12.54	$8.216 \cdot 10^{-20}$	$\mathbf{7.64 \cdot 10^{-21}}$
Hyper ellipsoid	$1.6226 \cdot 10^{-12}$	129.3112	301.2758	$3.011 \cdot 10^{-12}$	$\mathbf{6.282 \cdot 10^{-15}}$
Schaffer	$1.774 \cdot 10^{-3}$	**0.0**	$1.774 \cdot 10^{-3}$	$4.18 \cdot 10^{-3}$	$1.774 \cdot 10^{-3}$
Rastrigin	48.002	53.711	57.406	59.344	**32.634**
Rosenbrock	**598.121**	$8.745 \cdot 10^{4}$	$4.133 \cdot 10^{7}$	$8.3841 \cdot 10^{3}$	627.04
Griewangk	$4.042 \cdot 10^{-2}$	$4.02 \cdot 10^{-1}$	43.4643	$5.055 \cdot 10^{-2}$	$\mathbf{3.148 \cdot 10^{-2}}$



(a) (b)

Figure 3.38: The standard deviation curve of the Griewangk function in dynamic environment. a) The whole curve. b) The mean value of every 20 iterations.

The achieved results in the sensitivity analysis shows that the proposed controller parameter sensitivity is acceptable in the proposed PSO as the LHPSO achieved the best results in most of the functions. The standard deviations of the LHPSO are better than the others in most of the cases, and thus, the solution achieved each run is closed to the each other.

3.7 Multi-Objective Extension

In multi-objective optimization –also called multi-criteria or multi-vector optimization–, more than an objective are requireded to be optimized simultaneously. as in equation 3.30.

$$min \ \mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ f_3(\mathbf{x}) \end{bmatrix} \tag{3.30}$$

For example, for a given hardware, the complete set of specifications has to be consided in the optimization. In subsection 3.7.1, the aggregating function is described, which merge the set of object functions to one function. In subsection 3.7.2, Pareto based optimization is briefly mentioned.

3.7.1 Weighted Aggregating Function Based Optimization

In this approach, only one function is optimizated [PV02], which the sum of objective functions after weighting them as in equation 3.31.

$$F_a = \sum k_o \cdot f_o \tag{3.31}$$

This approach succeeds to get solutions from all parts of the Pareto set only for convex problems [DD96], dynamic weights can be used with non-convex problems to improve its performance [KdW06].

3.7.2 Pareto Based Methods

Definition: *A point* $\mathbf{x}^* \in C$ *is said to be Pareto optimal solution point for the multi-objective problem iff there does not exist* $\mathbf{x} \in C$ *such that* $\mathbf{f}(\mathbf{x}) \preceq f(\mathbf{x}*)$ *with at least one strict inequality.*

$\mathbf{x}$ *is said to dominate* $\mathbf{y}$ *if* $f(\mathbf{x}) \preceq f(\mathbf{y})$. *Let* $Y \subset C$ *and* $\mathbf{y} \in Y$. *If there is no* $\mathbf{x} \in Y$, *that dominates* $\mathbf{y}$, $\mathbf{y}$ *is said to be nondominated, where* C *is the search space of the vector* $\mathbf{x}$.

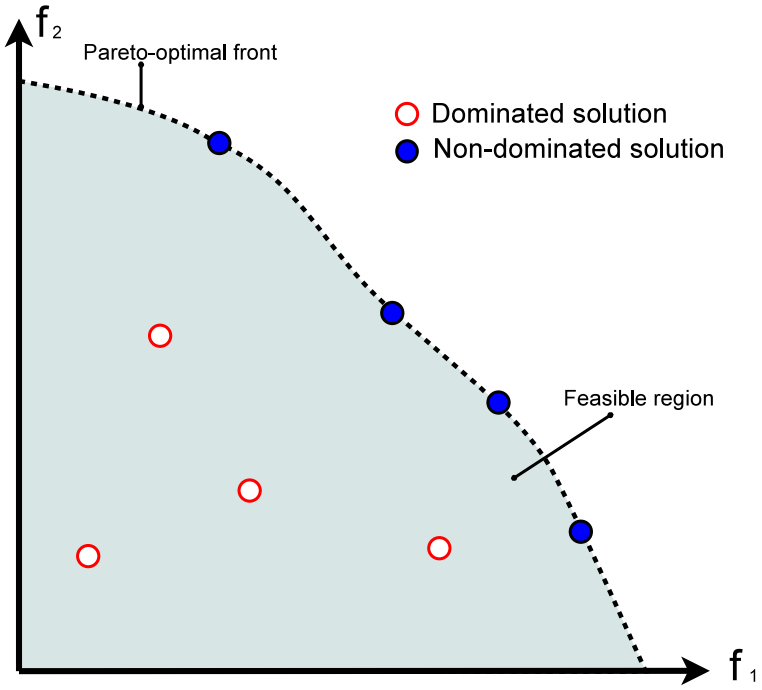


Figure 3.39: Illustrative example of Pareto for a maximization problem of f_1 and f_2.

All the Pareto based methods employ the nondominated solutions [FF93, HNG94, SD94] during the optimization, or copy them to external archive [Zit99, DAPM00, Fie04]. For a large number of objectives, the Pareto based methods are not efficient as many individuals are nondominated each iteration. However, even for big population, it is not expected that Pareto based methods can find good solution due to the large dimensionality of the solution space [FF95].

3.8 Discussion

A PSO modification that employs local parameters to each particle in the PSO is proposed and investigated. Including the local parameters for each particle improves the search path of the PSO as each particle can take the decision which searching path it will take depending on its local parameters, which keeps the diversity, improves the convergence speed and the solution quality in static and dynamic environments. A naive parameter controller is proposed for investigating the approach, and was sufficient to overcome the state of the art in most of the benchmark functions [TK06d]. After the parameters of the particle swarm optimization approaches vary in order to investigate the sensitivity

of employing local parameters, the achieved results are better than the other approaches in most of the benchmark problems. However, the acceleration coefficients of the proposed approach are initialized randomly in each particle in the initialization phase even without the sensitivity analysis.

As particle swarm optimization is already applied to engineering problems in the design phase *(e.g., telescope array, Golinski's speed reducer, satellite design [HCDV05], evolvable hardware [CLA02, GV04, LCA04, TK05b],etc.)*, such improvement in the convergence speed and the quality of the solution in dynamic environment is imperative.

Pareto based multi-objective optimization methods are concentrated on problems pursuing two or three objectives [FPL05], while many engineering problems requires many objectives optimizations. On the other hand, weighted aggregating function can be employed to optimize many objectives. Although quality of the returned solutions depend on the employed weights, it is simple, easy to implement, consequently, it is favorable for embedded systems.

CHAPTER 4

Evolutionary Electronics

The evolutionary electronics has gained more interest in the last two decades for its rapid prototyping capability that does not require complete knowledge about the design procedures and the employed technology, while it embodies self-x properties. For example, if the system is configured at the certain environment, soon after, its performance deviates due to the environmental dynamics, reconfiguring the system can improve its performance again towards the specified requirements if a feasible solution exists. The initial configuration copes with the static and the dynamic deviation of the system at the deployment phase, then the system is reconfigured to recover from any additional dynamic deviations at the run time.

The evolution of any system can be carried out extrinsically, intrinsically, or mixtrinsically. The extrinsic evolution evaluates the configurations in simulated hardware, therefore, it is used in the design phase only. The intrinsic evolution downloads the configurations to a real hardware, and evaluates it by real measurements. Thus, it can be employed at the deployment phase and the run-time, and it can cope with static and dynamic deviations. The mixtrinsic evolution contain some intrinsic and some extrinsic individuals in the same population, the reason on introducing the mixtrinsic evolution is described later on in this chapter.

The aim of this chapter is to introduce the state-of-the-art evolutionary electronics, compare it with the industrial requirements, and to show the missing link between them. Thus, a selection of the evolvable hardware that represents the perspective of the state-of-the-art is presented. Afterwards, the extrinsic, intrinsic and mixtrinsic evolution approaches, and their evolved circuits from the industrial point of view are discussed. Later on, the concept of coping with dynamic environment and faults are outlined. Finally, the chapter is summarized and concluded.

4.1 Reconfigurable Evolvable Hardware

The taxonomy of the evolvable hardware is shown in figure 4.1. It can be either analog or digital evolvable hardware. Reconfigurable analog evolvable hardware is analog hardware that embodies flexibil-

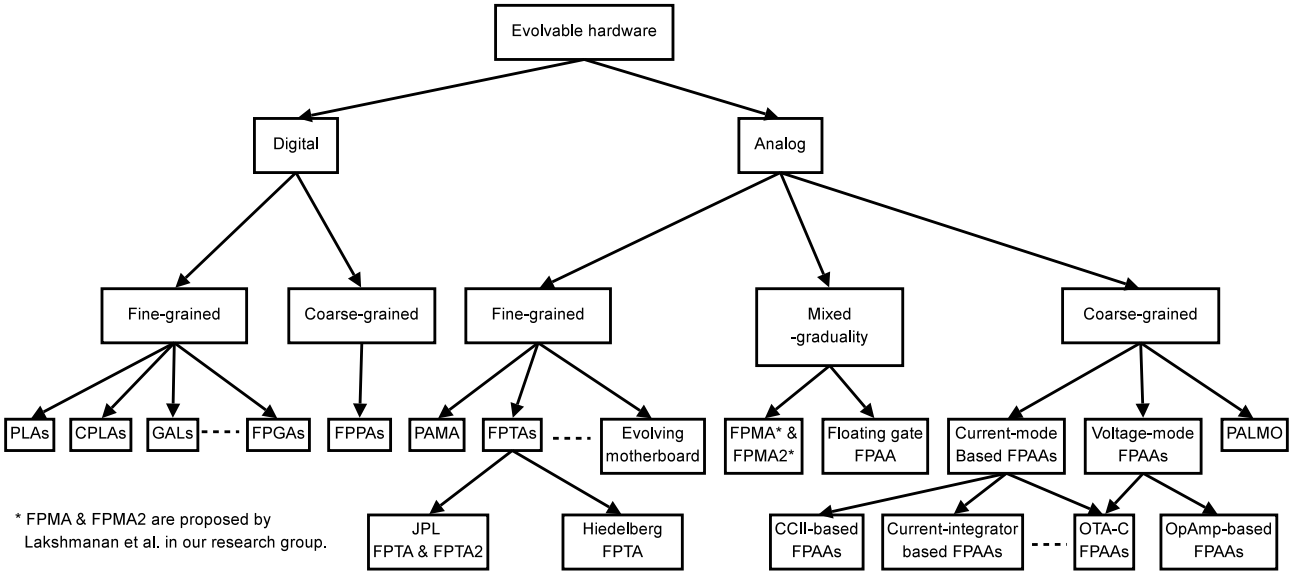


Figure 4.1: Evolvable hardware taxonomy

ity in its structure to program its specifications, its functionality, or both. It is called coarse-grained analog hardware if it incorporates flexibility in the functionality level in which, building blocks such as operational amplifier are combined with programmable passive components to construct a required function, where the building blocks are not programmable. If the hardware is programmable in the transistor level, it is called fine-grained analog hardware. On digital systems, the granularity of the hardware is referred to the data width of the processing elements. If processing element data width is one or more bit-wide, but not the whole processing word size such as in the configurable logic blocks (CLBs) in the FPGAs, it is called fine-grained architecture. If the processing element operates in the word-width, such as field programmable processor array *(FPPA)*, it is called coarse-grained hardware.

In this section, the basics of the fine- and coarse-grained evolvable hardware are described. The floating gate approaches [HHA02] are programmable only for a limited number of times, therefore, it is irrelevant to this thesis as it can be destroyed during the evolution of the hardware.

4.1.1 Coarse Granularity

Programmable digital systems are focused on the fine-grained approaches, but few work are concentrated on coarse granularity such as field programmable processor arrays *(FPPAs)* that target parallel processing applications and their fault tolerance enhancement.

Coarse granularity evolvable analog hardware such as FPAAs employs building blocks in addition to passive components if required to accomplish certain functionality.

The commercial FPAAs is described in subsection 2.5.1 employs operational amplifier as a building block in their architecture.

Non-commercial FPAAs employs other building components such as CCII-FPAA [Gau97] utilizes CCII as the building block, and two CMOS transistors to synthesis resistors. Other non-commercial FPAA employs current mirror and current-mode integrator as the building blocks for current mode

signal processing [EQO+98]. Others do not require any additional passive components such as Palmo described below, or target special applications such as OTA-C FPAAs that targets filter applications is described in this subsection as well.

Palmo

In many applications ramp analog to digital converter topology is employed to convert the signal to the digital domain in which the analog input signal is compared with a ramp function. The output of the comparison is a pulse with width proportional to the input analog signal level. The width of this pulse in encoded to the digital domain by further digital electronics. The Palmo architecture processes the pulse width directly without encoding it to digital signal.

The Palmo system is named after the Greek word "Παλμος" that means series of pulses. It is a coarse-grained FPAA developed by university of Edinburgh [HPTB98]. It employs the pulse based technique in which, the input and output signals of the Palmo cells are pulse width modulated signals; the width of the pulse is equivalent to the signal amplitude, and the level of the clock when the pulse is occurred represents the signal signs *(if the input pulse occurs while the clock level is high the signal sign is positive, and vice-versa)* as shown in figure 4.2.

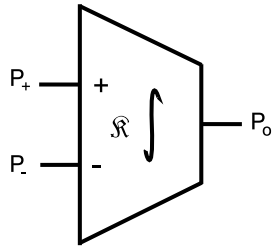


Figure 4.2: The symbol of Palmo cell

The functionality of each of the Palmo cells is an integrator that has an 8-bit integration constant $\Re$ as shown in equation 4.1. P_o is the width of the output pulse, and P_+ and P_- are the widths of the two inputs of the Palmo cell.

$$P_o = K\frac{(P_+ - P_-)}{s} \tag{4.1}$$

The main advantage of the Palmo cell is that no extra passive components are required; indeed, it employs $\Re$ as a programmable component to build filters, etc. Thus, functional level circuits are built only by choosing the value of $\Re$ and the connection between the cells. On the other hand, as the parasitic capacitance of bus switches and their on-resistance react as an RC filter, consequently, it can reshape the pulses. Thus, the signal has to be coded with wide width pulses which decrease the operating frequency. In addition, inputs of the Palmo cell control switches, which limit the bandwidth and add noise and non-linearities due to the switching activities. The operation frequency of the Palmo cells can not exceed one or two MHz [Pap98]. However, a comparator is employed internally to convert the level of the analog signal to a pulse. Therefore, the deviations of the comparator such as the offset effect the output pulse width. Furthermore, the Palmo cells require pre-processing analog hardware to convert the analog sensor signal to a proper equivalent pulses.

OTA-C Based FPAA

As the resistors are hard to be integrated regarding the consumed die area, and tolerance, many techniques attempt to employ capacitors instead of the resistors such as switched capacitors techniques, and OTA-C filters. The OTA-C filters employ the transconductance amplification *(G_m)* of the operational transconductance amplifier (OTA) and capacitors to build filters without the necessity of any discrete time components.

Pankiewicz et al. proposed an OTA-C based FPAA for continuous time filter application [PWSS02]. The transconductance amplification *(G_m)* of each OTA is programmable through programmable current mirror. The current mirror is programmable by employing 5-bit programmable transistor. The employed capacitors are 5-bits programmable capacitors as well. Thus, the G_m-C can be tuned to a wide range. A single configurable analog block *(CAB)* is shown in figure 4.3.

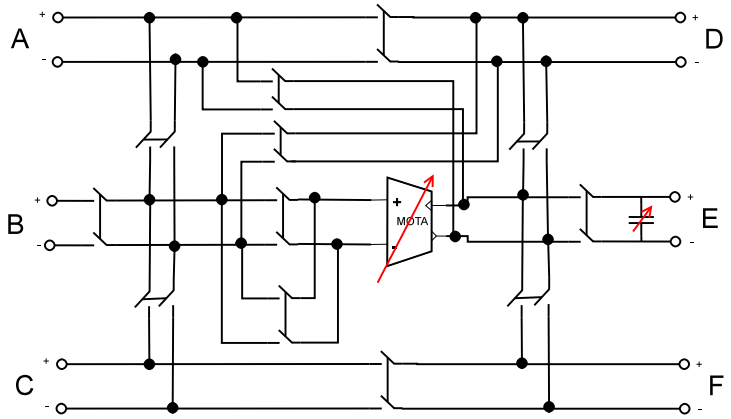


Figure 4.3: The CAB of the OTA-C based FPAA [PWSS02].

Pankiewicz OTA-C FPAA consists of 5×8 CABs and employs $2\mu m$ CMOS technology. Its structure incorporates flexibility that allows it to construct complex OTA-C filter structures. The bandwidth of the OTA is larger than 20 MHz. Utilizing programmable capacitors and OTAs embodies a possibility of building filters of frequencies from several kHz to several MHz [PWSS02].

Later on, Becker et al. [BM05] proposed a similar approach, but different architecture; it employs 7 OTAs and their parasitic capacitance. As shown in figure 4.4, the CAB cell has a hexagonal shape. Thus, it is connected to other 6 CAB cells.

The OTA bandwidth of Becker FPAA is about 293 MHz [BTHM07] which allows the operation at very high frequency *(VHF)*, on the other hand, Pankiewicz FPAA embodies more flexible structure and a wider programmable range to each G_m-C element. It has been demonstrated in [BM05] by a fourth order biquad Butterworth band-pass filter. The disadvantages of the OTA-C FPAA are that it is limited to filter applications, and it cannot recover from deviations such as offset.

4.1.2 Fine Granularity

Since the 70's, programmable fine-grained digital chips was available in the market such as PLAs *(programmable logic arrays)*, GALs *(generic array logic)*, then later on, CPLAs *(complex pro-*

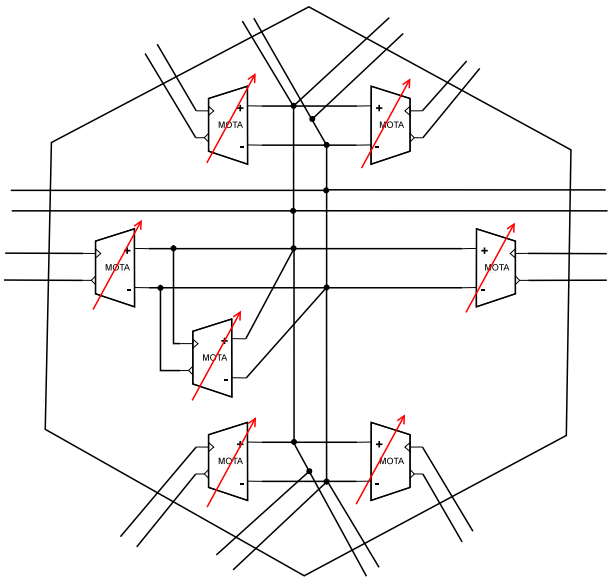


Figure 4.4: The CAB of the OTA-C based FPAA [BM05].

grammable logic arrays) and FPGAs which can construct more complex circuits than PLAs and GALs. Any reprogrammable digital hardware can be used as an evolvable hardware. The current technology allows designing complicate digital circuits using FPGAs, thus, FPGAs can be efficient employed as an evolvable digital hardware [Tho96].

The analog fine-grained hardware started to meet potential interest end of the 90's. This subsection is focused on the analog fine-grained evolvable hardware and their architecture.

Programmable Analog Multiplexer Array

Programmable analog multiplexer array *(PAMA)* [ZVP01] is developed by Catholic University of Rio de Janeiro. Its structure is shown in figure 4.5.

The PAMA employs arbitrary building components such as resistors, capacitors, transistors, or higher level components like operational amplifiers. A common $\mathfrak{N}$ lines analog bus is utilized to allow connecting all the components to each other. The terminals of each component are connected to an $\mathfrak{N} \times 1$ multiplexer in order to have access to any of the analog bus lines. It includes flexibility in which, any component terminal can be connected to any other one.

The operation frequency of the PAMA depends on the employed components, multiplexer characteristics, etc. For large number of components, huge number of multiplexers and a very wide analog bus are required which results in strong parasitic effects and large die area as each component requires multiplexers of $\mathfrak{N} \times 1$ attached to each of its terminals.

The AC behavior of the PAMA is not published, but it depends on the number of employed components and multiplexers, and the behavior of each of them.

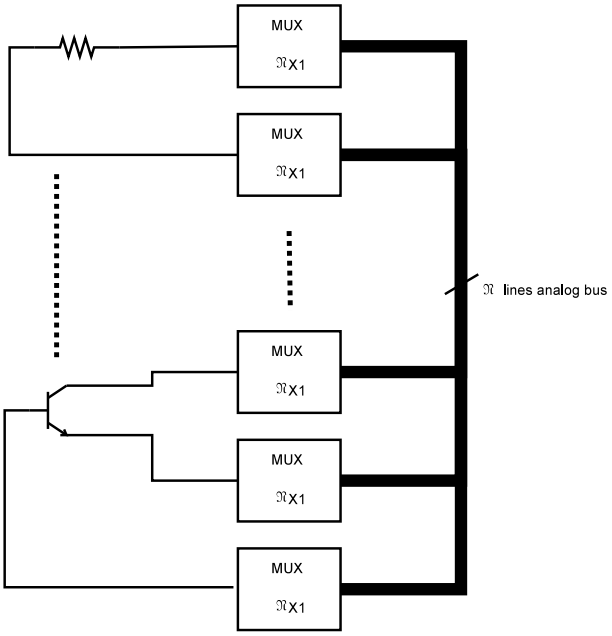


Figure 4.5: The PAMA architecture.

Evolvable Motherboard

The evolvable motherboard [Lay98] is a fine-grained evolvable hardware developed by university of Sussex. It consists of active components such as bipolar or CMOS transistors and many switches to connect them with one another as shown in figure 4.6. The terminals of any two components can be

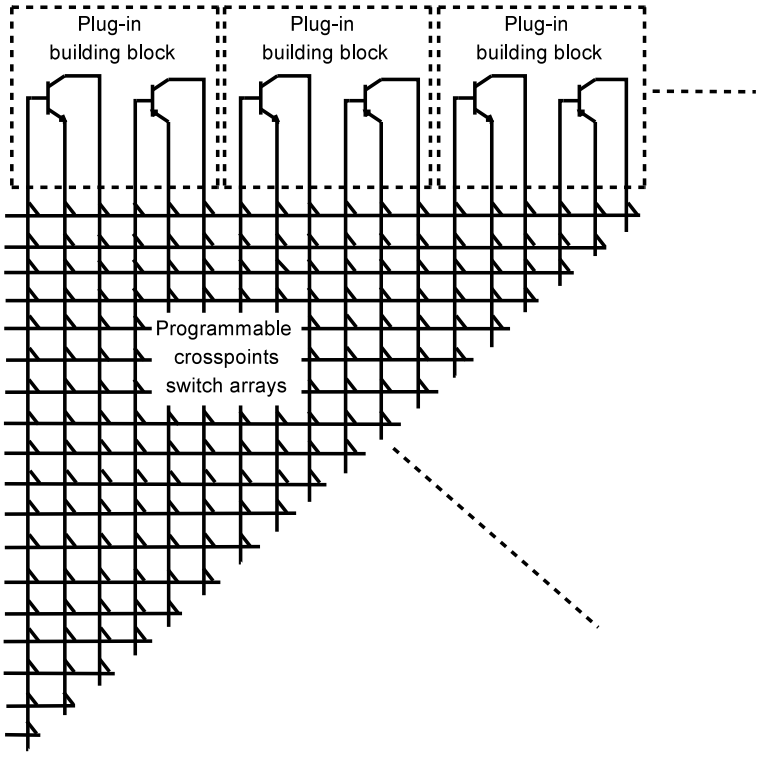


Figure 4.6: The architecture of the evolvable motherboard [Lay98].

connected by several paths, and by arbitrary number of switches. The evolving motherboard employs the switch on-resistances to design the system by utilizing more or less switches in the routing path, such as employing parallel switches to reduce their resultant on-resistance, etc. As each component is connect to an array of switches, the parasitic capacitance increases dramatically. Thus, the evolvable motherboard is limited to low frequency application. However, the resistance of the on-switches is not a constant as the voltage-current relation is non-linear. Therefore, non-linearities are added to the system in large signal applications.

The number of switches for $\mathfrak{T}$ terminals is $\frac{\mathfrak{T}(\mathfrak{T}-1)}{2}$. Thus, for ten transistors, two inputs, one output, power supply and ground, more than a hundred switches are required.

The AC behavior of the evolvable motherboard is not reported. As many switches are employed that increases the parasitic effects, its bandwidth is expected to be narrow.

JPL Field Programmable Transistor Arrays

The NASA's jet propulsion laboratory *(JPL)* has proposed field programmable transistor array *FPTA* [SKT$^+$99] and its recent version *FPTA2* [SZF$^+$02]. The structure of the JPL FPTA is shown in figure 4.7. Each transistor can be disconnected from the circuit, or short-circuited. Many transistors

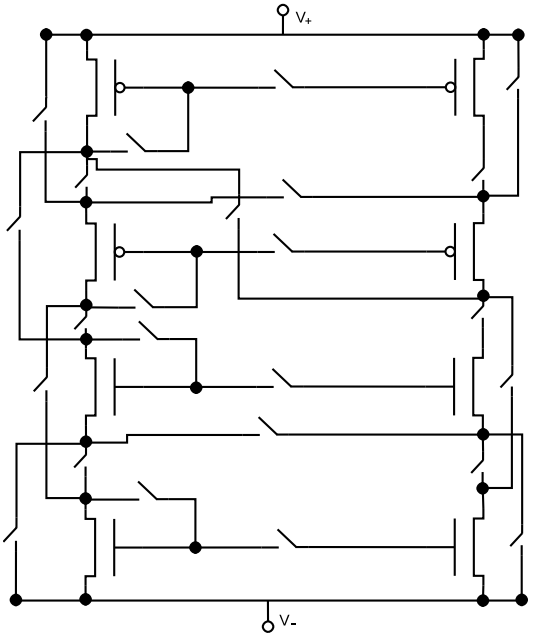


Figure 4.7: The JPL FPTA [SKT$^+$99].

between the sources and the ground; 4 transistor and 3 switches, although the transistors and its related switches can be short-circuited according to the configuration, the voltage drop due to employing many switches may decrease the dynamic operation range. The structure of the JPL FPTA employs many switches to connect 8 transistors to each other. It is programmable to many arbitrary structures, but transistor dimensions are fixed. Thus, the hardware standard specifications are not programmable.

The JPL FPTA does not have an output stage, while, the output impedance of the FPTA cell is high due to employing many switches and transistors between the power supply terminals.

Later on, the JPL group proposed FPTA2, which contains 64 cells, each cell contains three configurable stages as shown in figure 4.8, and other programmable resistors and static capacitors [SZF+02] that are not shown in the figure. An output stage with two transistors and one switch is employed to

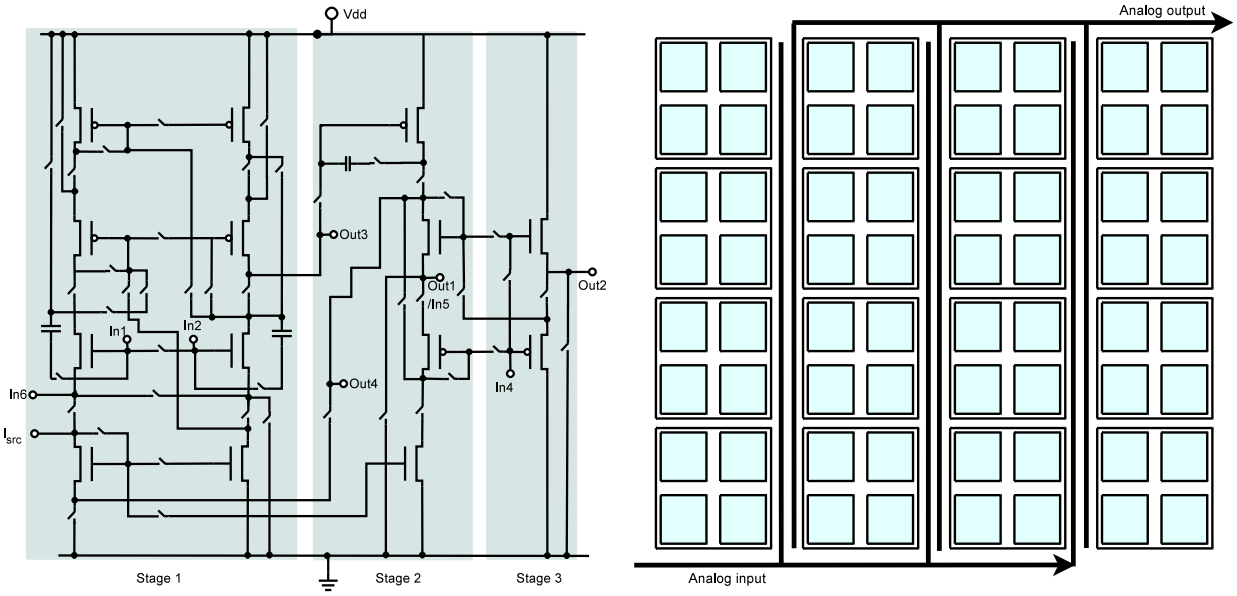


Figure 4.8: JPL FPTA2 [SZF+02] a) Structure of one cell. b) The FPTA2 chip consists of 64 cells.

inherit low output impedance. The three stages of the FPTA2 is suitable to be configured as a complete operational amplifier; the first stage can be configured as differential input stage, second stage as gain stage, and the third stage as the output stage of the operational amplifier. The configuration is downloaded to random access memory by 9-bit address bus, 16-bit data bus for fast programming. The up to 7 transistors can be employed between the source and the ground in first stage which can results in voltage drop on the switch and therefore, small common mode range. The transistor of the FPTA2 is not programmable as well. Thus, the hardware topology is programmable, but the specifications of the hardware are not programmable. On the other hand some of the functional level specifications can be tuned by employing the mentioned programmable resistors.

The AC characteristics of the FPTA and FPTA2 are not investigated in the published work so far. However, in [ZSK00], an AM filter with center frequency of about 200kHz was evolved on a single FPTA simulated model where the parasitic capacitance of the switches was not included in the simulations.

University of Heidelberg FPTA

The University of Heidelberg proposed an FPTA that consists of 16 $\times$ 16 programmable transistors [Lan05]. Figure 4.9 illustrates the outlines of Heidelberg FPTA. There are two types of transistor modules, PMOS modules, and NMOS modules. Each transistor module is connected to its north, south, east and west neighbors transistors –except the transistors at the edges–. It can be programmed as a wire between any two poles, or as a transistor with a give width and length. The neighbors of the NMOS transistor module is PMOS modules, and vice versa. The structure of a NMOS transistor

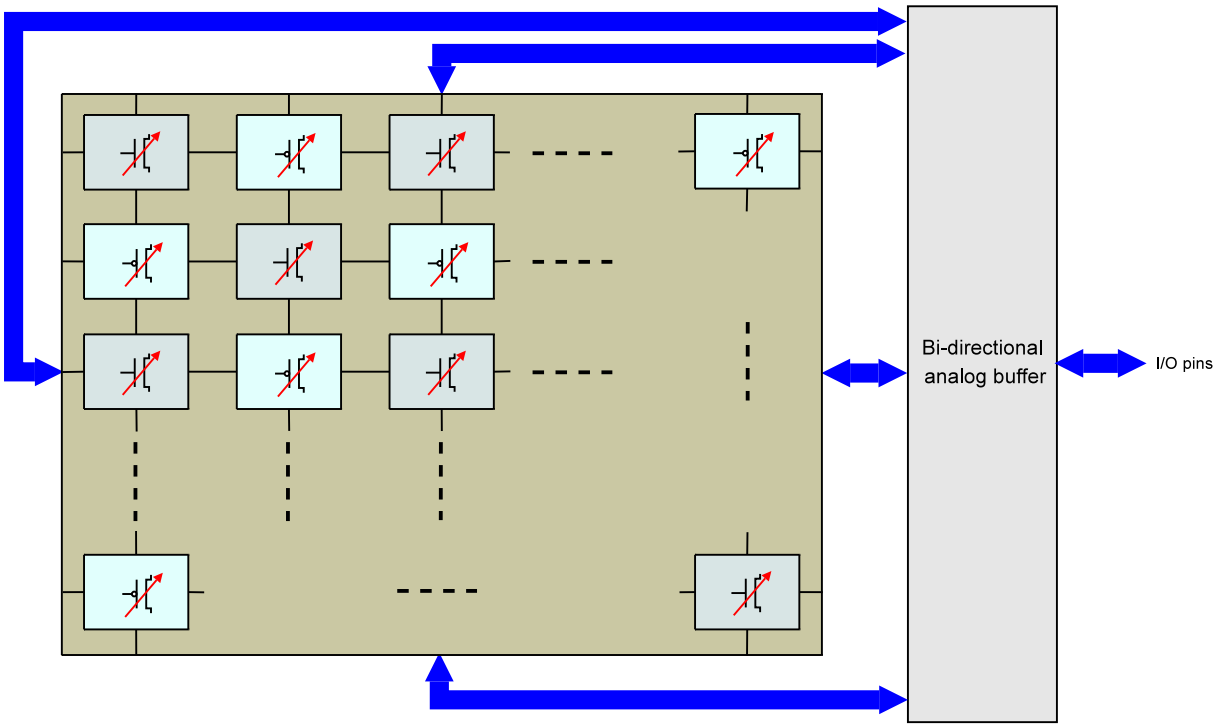


Figure 4.9: A simplified structure of the Heidelberg FPTA [Lan05].

Table 4.1: Widths of the employed switches [Lan05].

Switch Type	PMOS [μm]	NMOS [μm]
Current	30	10
Voltage	1.4	1.4

module is shown in figure 4.10. Each module can be programmed to 6 different lengths, while the width is programmable by 4 bits. Internally inside each module, all the transistor drain pins are connected together, and similarly the source pins of the transistors. Programming the transistor modules is achieved by controlling the gate of its internal transistors. Each of the gates of the internal transistors can be connected to the global gate pin, or to ground inorder to turn the transistor off. Each drain, source, and gate of a transistor module can be routed to north, west, south, east, power supply, or ground.

The lengths of all the switches is fixed to $0.6\mu m$, table 4.1 summarize the widths of the employed switches. As shown, two types of switches are employed, the voltage switches are small in area to control the gate of the module internal transistors. Thus, the required current that it should conduct is negligible. The current switches are bigger in area in order to conduct higher current without leaving the Triode region. The current switches are employed for routing purpose.

In order to accomplish low output impedance and high input impedance, inputs and outputs of the chip are buffered

Big area is consumed to program the length and the width of the transistor modules. It would be sufficient to fix the length of the transistor to a reasonal length and size the transistor width to program its G_m. The Heidelberg FPTA has no passive components in its internal structure, which limit its

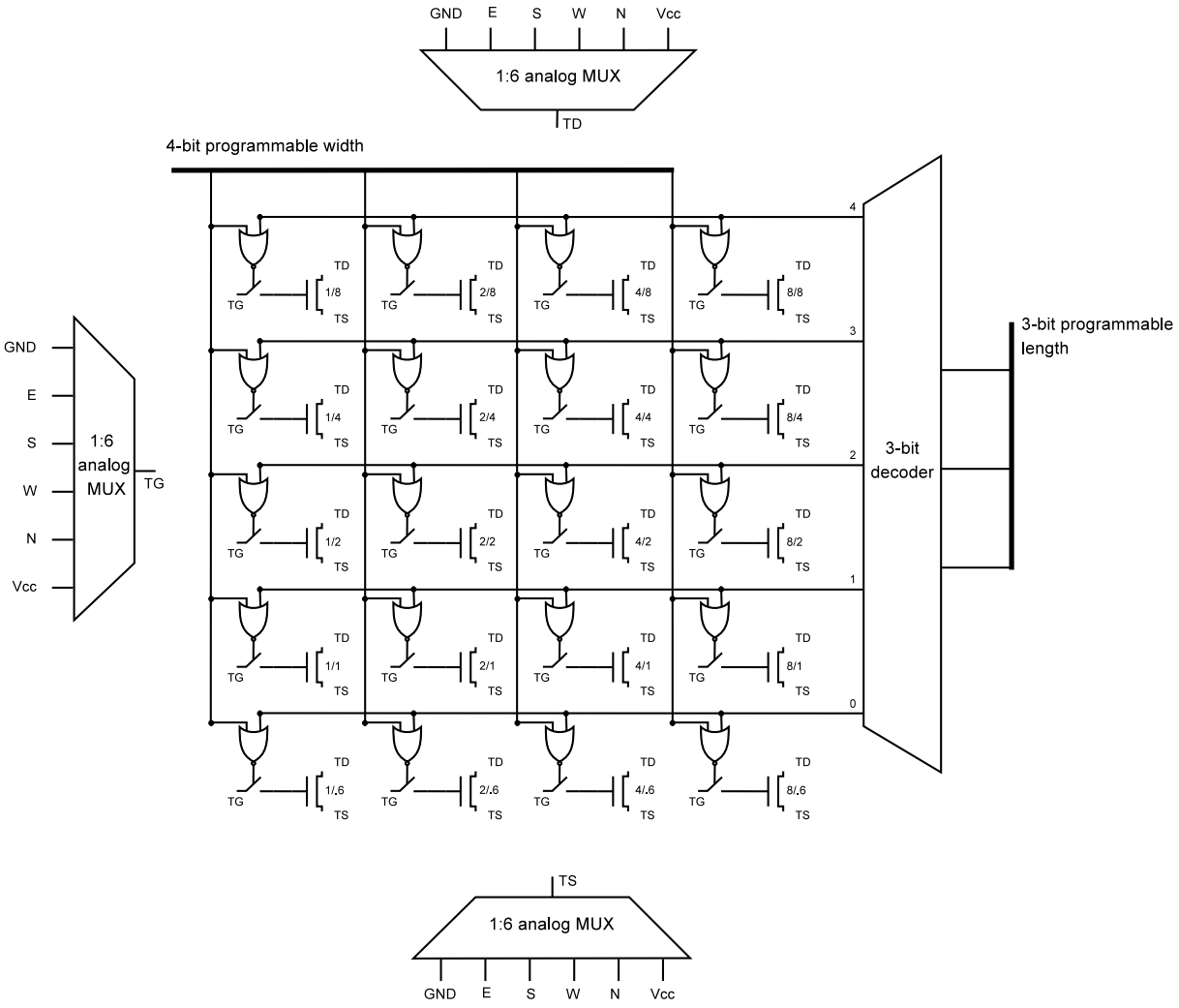


Figure 4.10: The structure of a single programmable transistor module of Heidelberg FPTA [Lan05].

application. Moreover, some transistors modules are employed to operate as a wire, which limit the use of the chip to very small design. For example, 5×5 cells are employed to design a logic gate [LMS02].

Trefzer mentioned an evolved operational amplifier on Heidelberg with 0db frequency of 8MHz [Tre06].

4.2 Evolutionary Computation Adaptation for Evolvable Hardware

Focusing on evolution of the hardware topology, genetic programming is modified to invent new hardware topologies. Targeting a given evolvable hardware, the genetic algorithm can be employed to control the hardware switches. Other modifications of the genetic algorithm for topology optimization are accomplished. The relevant modifications of evolutionary computation are described in this

section.

4.2.1 Evolving the Hardware Topology by Genetic Programming

The genetic programming can evolve the hardware structure [KBA+97] starting from scratch. An input circuit and output circuit are employed to assess the evolving circuit. As shown in figure 4.11, the evolving circuit is initialized by having only a wire connecting the output to the input as a primal soup. This wire is a component that connects the input to the output. In each iteration, any component can be flipped to another one *(e.g. wire to resistor)*, or divide into several components *(e.g. resistor to resistor-wire-resistor, or resistor to two parallel resistors)*, the terminals of any component can be reconnected to another node, etc. The algorithm iterates till it finds a good configuration. For example, the wire in figure 4.11, can be flipped to any other component, such as resistor as shown in figure 4.12(a), then the resistor can be divided into two parallel or series resistors as shown in figure 4.12(b).

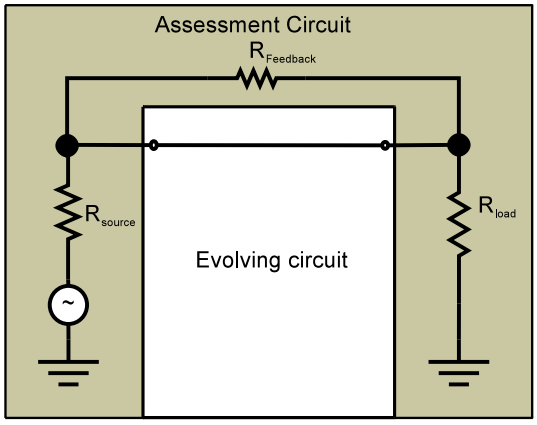
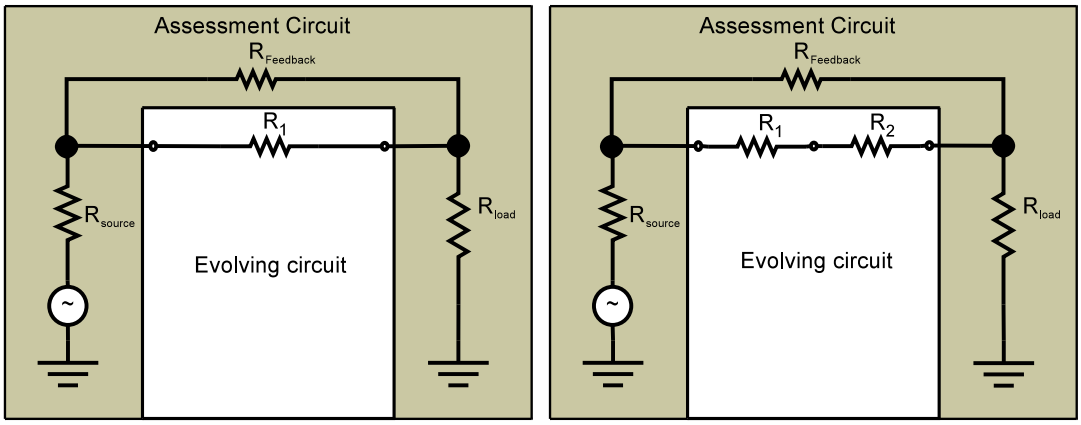


Figure 4.11: The initial circuit in designing an operational amplifier with GP.



(a) (b)

Figure 4.12: Example of GA operations in circuits. a) Component flipping. b) Component dividing.

Koza et al. have employed the genetic programming to evolve 96 decibel operational amplifier [KFH-BAK97]. A feedback circuit has been employed as an assessment circuit with closed loop gain

$G_{closedloop} = 10^6$. Based on the SPICE DC analysis at five points between $-10mV$ and $10mV$; the gain error, offset and linearity error are weighted, then accumulated in a fitness function.

The selected specifications for the optimization do not have a have strong influence to the industrial specification, although it has some relations to few of them. For example, the nonlinearity which was measured depending one the input voltage range between $-10mV$ to $10mV$ is correlated to the output swing voltage. If the input voltage is higher than the input level during the evolution, it is not guaranteed that the same amplifier can operate as the common mode range is not optimized. Koza claimed that genetic programming designed 96db operational amplifier without taking care of the industrial specifications that the human consider [KFHBAK97]. Evolving the hardware topologies increases the possibility of returning hardware with unpredictable behavior. Employing few of the industrial specifications constructs a poorly specified problem that can return a non-working hardware that satisfies the problem requirement. For example, a non-stable operational amplifier can be returned if only DC simulation is considered.

Zebulum et al. [ZPV99] evolved filter circuit starting from scratch. The frequency response of the filter is optimized by equation 4.2, where f_F is the objective function that stands for frequency response of the filter, $\mathfrak{f}_{max}$ is the number of simulated points in frequency domain, $V_{out}(\mathfrak{f})$ is the amplitude of the output at the frequency $\mathfrak{f}$, $V_{in}(\mathfrak{f})$ is the filter input at the frequency $\mathfrak{f}$, and $\mathfrak{w}_\mathfrak{f}$ weight of $V_{out}(\mathfrak{f}) - V_{in}(\mathfrak{f})$ at the frequency $\mathfrak{f}$, its value is positive for the passband to maximize gain, and is negative for the stopband to minimize the gain.

$$f_F = \sum_{\mathfrak{f}=\mathfrak{f}_{min}}^{\mathfrak{f}_{max}} \mathfrak{w}_\mathfrak{f} \cdot (V_{out}(\mathfrak{f}) - V_{in}(\mathfrak{f})) \tag{4.2}$$

Experimentally, Zebulum has chosen the weights 33 for the passband, -4 for the high frequency stopband, and -1 for low frequency stopband. In addition to the frequency response optimization, Zebulum minimized the power dissipation, etc. However, the industrial specifications are not considered such as offset, required stopband attenuation, and passband gain, input signal maximum amplitude *(in other words, the common mode range of the active amplifier)*, etc. are not included in the optimizations. The topology of the filter is arbitrary as it is designed by the evolution, therefore, its behavior is unpredictable. The results presented in [ZPV99] have many distortions in the transient analysis.

Applying the genetic programming to evolve a topology of a building block (e.g. operational amplifier, logic gates, etc) intrinsically is not possible as it requires a fully flexible analog hardware that is hard to be developed with the current technology, therefore, it can be applied only in a simulated environment.

4.2.2 Turtle GA

The Turtle GA has been invented to evolve Heidelberg FPTA in figure 4.9. It employs the genetic algorithm *(GA)* to evolve the hardware aiming to return a hardware without any floating terminals, and to find a useful dimensions of the employed transistor modules during the search. The following new operators are added to the basic operators of the GA in chapter 3 section 3.2:

- Random wires (mutation) operator: Randomly select a starting node (a pole at any cell). The selected node can be connected to any of other three poles in the same module, to any of the

internal transistor three terminals, or any of the three poles in the adjacent neighbor cell. Nodes are recursively connected or deleted till all the nodes are connected.

- Implanting a foreign block of cells (crossover): this operator is done on two stages:
  - Choose a random rectangle from the partner, and copy it to the new offspring.
  - Connect the floating nodes as in the random wires till no more nodes are floating.
- Logic OR of Individuals (crossover): Merge the partners; all the transistors and the connections of both are copied to the offspring. If a transistor is used in both parents, the value of aspect ratio of the first transistor is employed.

In addition to these operators, the aspect ratio of any transistor module mutates by a given mutation probability and rate.

The Turtle GA search for new topologies as the genetic programming, but it targets hardware that has less flexibility and does not employ any passive components. It evolves arbitrary structure as the genetic programming. Therefore, it is not useful for the industrial applications as the behavior of the arbitrary topologies is not predictable.

4.3 Extrinsic Evolution

The design automation is important to reduce the design time and effort of the hardware. The extrinsic evolution is either employed as a synthesis tool to design hardware, or utilized to a target hardware to find a configuration to it. In this section an overview on the state-of-the-art extrinsic evolution is provided. In the state of the art, it may have the freedom to invent the topology.

4.3.1 Extrinsic Evolution as a Synthesis Tool

Aguirre et al. [ACB99] have employed the genetic programming to construct digital multiplexer tree structure. The problem has been described by a given truth table. The fitness function consists of two stages; the first stage is to find working solutions. Therefore, maximizing the number of hits[1]. Once a working design is found, the second stage function gives better fitness value for the fully functional circuits with less multiplexers. Thus, the optimization objective is to find a low cost working design. The variables are allowed only to control the multiplexers, the input of any multiplexer is either 0 or 1. Although genetic programming is used to construct the multiplexer tree, using multiplexers in combinational circuits design is well known technique in synthesis of digital circuits. Therefore, the behavior of the returned hardware is predictable.

Zebulum et al. have synthesized an operational amplifier [ZPV98] by selecting standard amplifier topology, and dimension its components. Few of the industrial specifications have been included in the optimization criterion; the open-loop gain, the band-width, phase margin, and power consumption.

[1] A hit is to get a correct output at a specific input according the truth table. Maximizing the hits minimizes the output error.

Simplified analytical expressions have been employed to describe the operational amplifier behavior. Weighted aggregating function based optimization have been utilized to achieve multi-objective optimization of the included industrial specifications. The length and the width of the transistors are represented as real numbers. The genetic algorithm is utilized as the optimization tool. However, for this problem representation, geometric programming is more powerful [dMHBL01] as it converts the problem to convex problem with only one optimal solution and employs the convex problem optimization methods to find the optimal solution. The behavior of the operational amplifier that has been synthesized in [ZPV98] is partially predictable behavior as only few of the standard specifications are employed in the optimization. The SPICE can be employed instead of the analytical expressions in order to achieve more accurate results [LMU04]

4.3.2 Extrinsic Evolution for a Target Programmable Hardware

Coello et al. [CLA02] used the hardware in figure 4.13 in which, the evolutionary algorithm chooses the inputs and the gate type in the initial level gates, and only the type of the gates in the rest of the levels. The gate type can be AND, OR, NOT, XOR gates, or a wire. As Coello approach employed

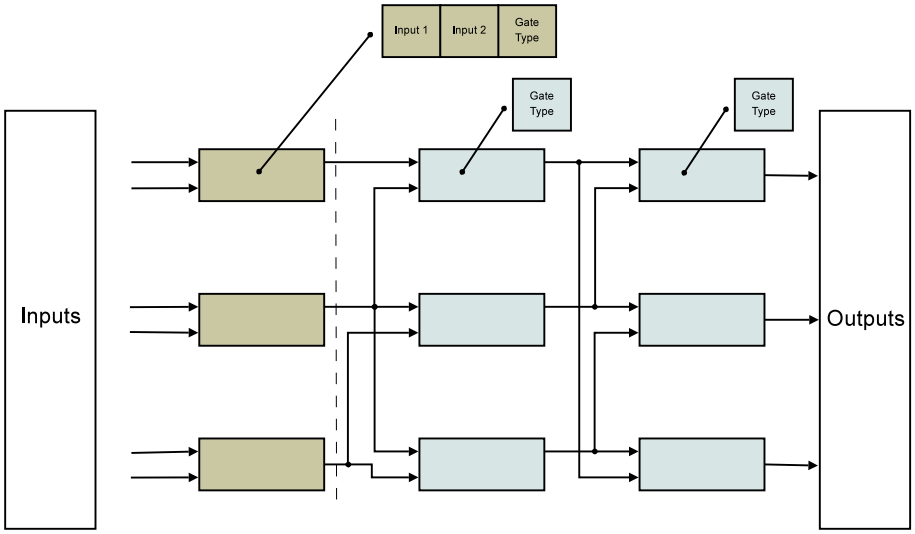


Figure 4.13: The simulated hardware used by Coello to synthesis digital combinational circuits, extract from [CLA02].

standard digital gates well known topology, the behavior of the evolved digital circuit is completely predictable. Similar to Aguirre [ACB99], Coello employed objective function with two phases; the first phase is to maximize the number of hits, once a working solution is found, the second stage of the objective function is applied which, maximizes the number of gates that are employed as a wire. Any component that is employed as a wire can be removed in the final representation, therefore, maximizing the components that are used as wires results in minimizing the employed components in the final design. Although this optimization method targeting a programmable hardware and optimized it, it is employed as a synthesis tool as the hardware does not exist in a real chip, and the optimization aim is to minimize the number of components that are employed as a wire, otherwise, the optimization aim should be maximizing the unused components

The JPL has employed binary genetic algorithm to evolve the FPTA in figure 4.7 by controlling its

switches [SKT+99]. In both the extrinsic and intrinsic evolution, the objective is to minimize the error between an output reference signal and the actual output signal of the FPTA due to a given input signal, which is a poorly specified problem as the complete industrial specifications are not included in the optimization. No passive programmable components are integrated in the first version of the JPL FPTA. The evolution search for a topology, but it does not dimension the transistors to achieve a given specifications as the transistors dimensions are fixed. The main advantage of employing the extrinsic evolution to evolve the JPL FPTA is to avoid short circuits as arbitrary topologies is allowed. However, short circuits can be investigated without a complete simulation. An example of the achieved results by this approaches is the AND gate evolved extrinsically on the JPL FPTA that is shown in figure 4.14. A current source is connected to the first input, thus, the first input should

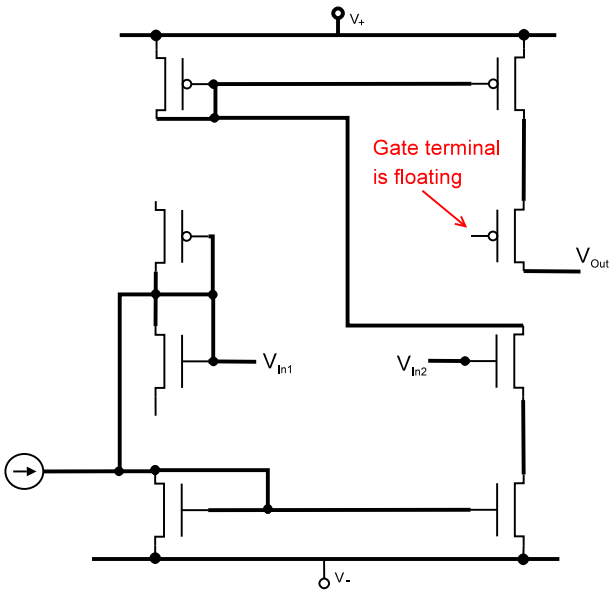


Figure 4.14: An extrinsically evolved AND gate [SZK00].

be able to sink the current. The output is supplied through a transistor that its gate is connected to a floating terminal. Therefore, it can operate in simulation only when the switches off-resistance is simulated, and no noise affects the gate. The input impedance is low as one of the inputs is connected to a current source, while the output impedance is unpredictable as the output is supplied through a transistor that its gate is connected to a floating terminal. Thus, the output is dependent on the charge on the parasitic capacitance at the gate terminal. The configuration of the evolved AND gate in figure 4.14 has not worked when downloaded to real chip. The industrial specifications of the designed AND gate is not considered in the optimization, therefore, the behavior is unpredictable.

The JPL FPTA2 in figure 4.7 is employed to evolve more complicated structures with the same principle the JPL FPTA used; evolving the topology to optimize the error between the actual and a reference output [SZF+02]. The number of the required cells to evolve specific design is unknown, and can be determined by trial and error. Two cells are employed to implement a half wave rectifier, which can be achieved by single diode. Ten cells are employed to evolve a filter.

Aguirre et al. have employed the same hardware, to optimized digital design DC specifications which is a multi-objective approach [AZC04], but they do not include the industrial specifications. However, they considered the outputs of a 2-bit ADC as different objectives, and the levels of the output voltage as constrains. As the number of objective was small –only the two outputs of the 2-bit ADC–, they

have employed Pareto-based optimization approach. As in the previous work by JPL FPTA, the topology of the evolved hardware is arbitrary, therefore, the behavior is not predictable.

Parallel to the proposed extrinsically evolution in the next chapter [TLK05], Trefzer et al. evolved Heidelberg FPTA that is shown in figure 4.9 targeting single objective or few of the industrial specifications [TLMS05]. The Turtle GA is employed to evolve hardware with arbitrary topology. In order to include a limited selection of the commercial set of specifications of the hardware industrial specifications, the hardware is evolved extrinsically, then the returned configuration is downloaded to the chip assuming that it can work intrinsically. According to Stoica et al. [SZK00] evolved systems extrinsically may not work intrinsically, as the operating condition of the evolved hardware is different from the real operation conditions.

In figure 4.15, the topology of an evolved operation is shown. The sources of the transistors that are drawn in gray have no path to the power supply, thus, only their parasitic impedance are affecting the design. The gate of some of the transistor modules is connected to ground or the power supply,

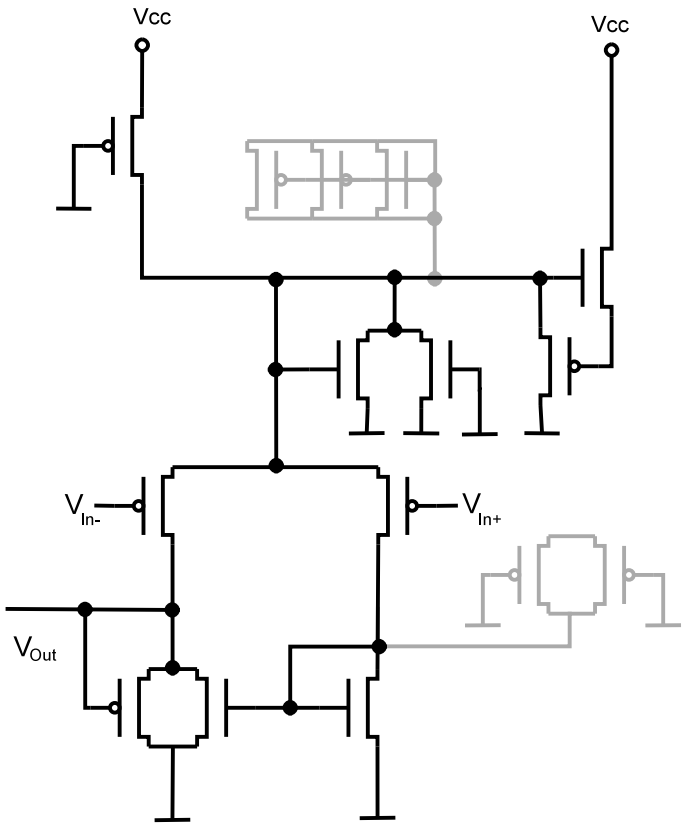


Figure 4.15: An operational amplifier implemented on Heidelberg FPTA [TLMS05].

therefore, it reacts as an on- or off-switch. The evolution can connect any current path to ground, which can results in power consumption and self heating. The number of required cells to evolve hardware is unknown, and can be determined by trial and error.

4.4 Intrinsic Evolution

Thompson introduced the first intrinsic evolution in FPGA [Tho96]. He has used the genetic algorithm to program 10x10 cells in the FPGA through a genome of 1800 bits. The task which the evolution had to fulfill was to discriminate between 1 kHz and 10 kHz square waves, which is a simple task. Later on, Thompson et al. analyzed the result of the evolution [TL99]. Evolving a digital system is more straight forward approach than an analog system regarding the problem representation. Unlike the analog systems, the specification of the digital systems can be simply defined and measured by observing the DC behavior. For example, intrinsic evolution of combinational circuits can maximize the number of succeed hits as the objective function, which is a straight forward approach, especially that the complete truth table for the digital circuit is optimized. However, if the complete specifications of the digital system are included, the time delay between the input and the output signals has to be optimized according to the application requirements as well. The output current, and some other specifications are already optimized in the technology as the evolution is done at the gate level. Designing the gates is considered as an analog design as the exact analog level, transient characteristics, etc. are required to be optimized. Optimizing them considering only the input-output relation without considering the time delay, the output current, input resistance etc. [TW00, Tho02] is a poorly specified problem, and the result of the evolution is not guaranteed to work.

Vinger et al. evolved an 8-tab finite impulse response *(FIR)*-filter on an FPGA [VT03]. The FIR-filter structure is shown in figure 4.16. The hardware is evolved by tuning the filter parameters, thus, the

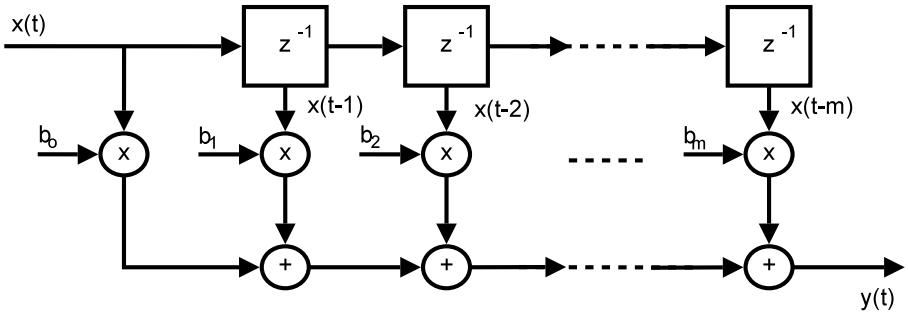


Figure 4.16: FIR filter.

behavior of the filter is predictable. However, the fitness value in [VT03] is the difference between the output of the FIR-filter and the output of a reference output. Thus, a reference FIR-filter has to be implemented and to process the input signal in order to evaluate the evolving filter. Thus, any improvement in the filter specifications such as larger stop band attenuation in the evolving FIR-filter impairs the fitness value of the filter.

Stefatos et al. proposed a low power version of the evolvable FIR-filter [SAKF06] by designing a digital hardware that includes programmable FIR-filters. Stefatos hardware can recover from permanent errors such as fault latch state or memory cell by finding new coefficient pattern for the FIR filter [SAKF06]. The advantage of the FPGAs over [SAKF06] is that the fault cells in the FPGA can detected [ESSA00], then the same FIR-filter configuration can be remapped to the working cells without the need of a complete reconfiguration loop in order to avoid a training period and learning loop. However, the fault can occur in the reference FIR-filter, consequently, the functional FIR-filter can deviate.

Evolving coarse-grained analog hardware is achieved on commercial FPAA chips [FS98] to evolve arbitrary topology. The objective was to minimize the error between an output and a reference output. Potential connections may destroy the chips by connecting the power supply terminals with each other during the optimization. As the FPAA does not utilize programmable building block devices, it does not embody the self-x properties.

The JPL FPTA and FPTA2 are intrinsically evolved by specifying the relation between the output and reference output due to a stimuli input signal [SKZ+02]. An example of the evolved hardware is the AND gate shown in figure 4.17. The evolved AND gate contains transistor gate terminals that are floating, and the input impedance of the first input is very low. In addition, the structure that is evolved is not guaranteed to work. The evolved hardware in figure 4.17 does not work in simulation [SZK00].

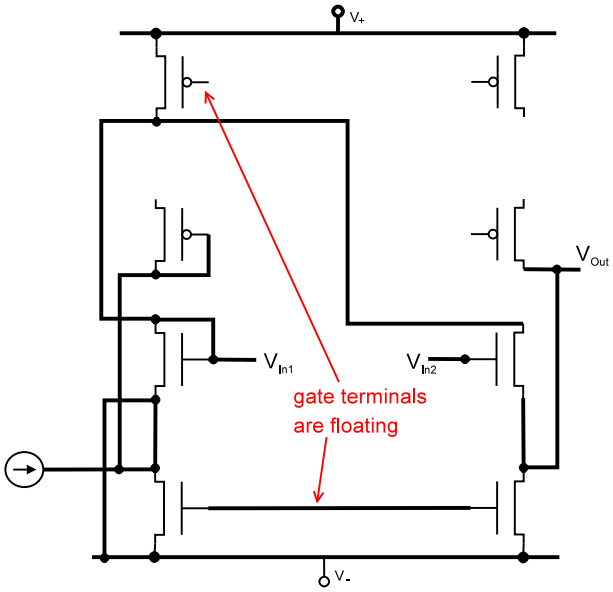


Figure 4.17: An intrinsically evolved circuit [SZK00].

Amaral et al. have employed genetic algorithm to evolve PAMA [dAdAS+04] with 32 analog multiplexer, each has 16-to-1 lines. The error between the DC behavior of the output and a reference output signals stimulated by a reference input is employed as the fitness function. Dangerous connections such as connecting the power supply with the ground has to be detected and removed before downloading it to the hardware. In figure 4.18, an evolved logarithmic amplifier is shown. As shown, the evolution employs some non-operating components, connects output of some components to one of the power supply terminals, leaves some of the terminals floating, connects some resistors between the supply terminals, etc. Additionally, only single objective function for the DC characteristics is considered. Thus, the behavior is unpredictable, and some configuration can results in dynamic deviations such as self heating, or even damage the hardware. Therefore, the configuration of the evolved hardware is not guaranteed to operate again after evolution. The same problems exist for the evolvable motherboard [Lay98]. The proposed result of evolving a NOT gate in [Lay98] contains many floating terminals, and useless components.

Langeheine et al. employed the Heidelberg FPTA [LMS03] considering its DC behavior to evolve logic gates, analog Gaussian function circuit, etc. The evolution invents the topology and searches for the transistor dimensions. The Turtle GA [TLSM04] guarantees that no floating terminal is found in

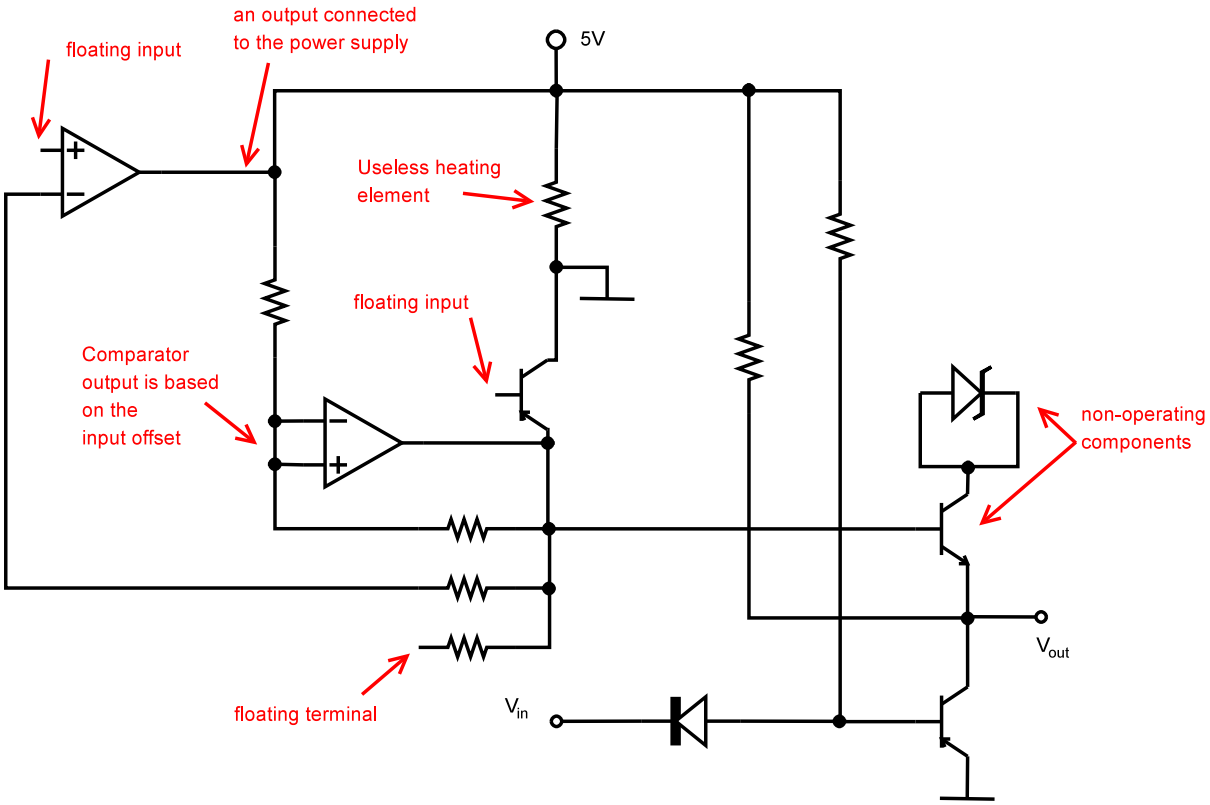


Figure 4.18: An evolved logarithmic amplifier [dAdAS+04].

the design. However, similar to the approaches mentioned above, optimizing only DC characteristics and employing arbitrary topologies results in unpredictable behavior and can not be accepted by industry.

4.5 Mixtrinsic Evolution

Mixtrinsic evolution has been introduced by Stoica et al. [SZK00], trying to solve the portability problem, which is; the hardware designed by intrinsic evolution may not work in simulation and vice-versa. As shown in figure 4.19, the mixtrinsic approach evolve some individuals extrinsically and the rest intrinsically in order to improve the portability. However it does not consider complete industrial specifications of the hardware in the optimization criteria. Indeed, it minimizes the error between a reference output and the actual output generated by a given test signal. The intrinsic individuals in figure 4.19 are called *"HWi"* as their fitness is evaluated by hardware measurement. Similarly, extrinsic individuals are called *"SWi"* as their fitness is evaluated by software simulation. The reason of the portability problem mentioned at [SZK00] is that the hardware structure is arbitrary and the complete industrial specifications are not included in the optimization. Thus, the evolved hardware is not guaranteed to work due to any small environmental variation. Two types of mixtrinsic evolution are described in [SZK00]:

- Combined mixtrinsic evolution: *The same objective function* of each particle is evaluated both intrinsically and extrinsically, the final fitness value of the average.

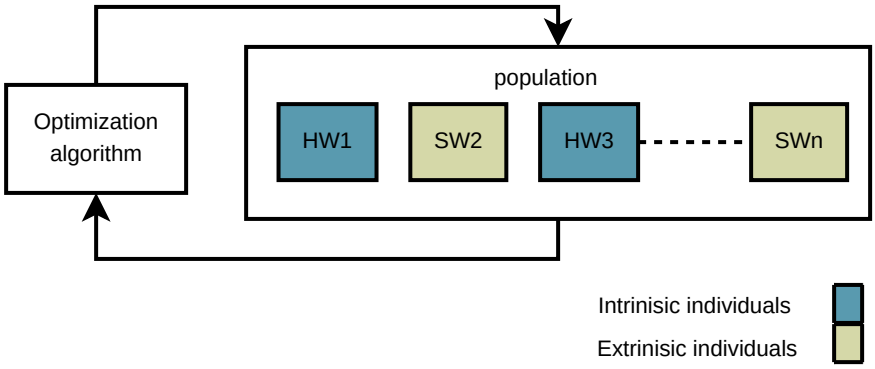


Figure 4.19: The population in mixtrinsic evolution [SZK00].

- Complementary mixtrinsic evolution: The objective function of each particle is evaluated either intrinsically and extrinsically, but not both.

For example, an AND gate is evolved extrinsically to the circuit shown in figure 4.14, intrinsically to the circuit given in figure 4.17, while mixtrinsically to the circuit given in figure 4.20.

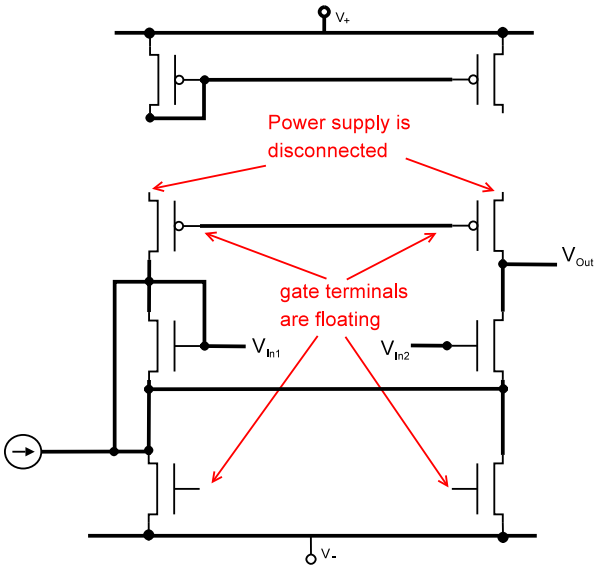


Figure 4.20: An evolved circuit mixtrinsically [SZK00].

The circuits that were intrinsically and extrinsically evolved have gate terminals that are floating. Thus, the output depends on the noise inside the chip during the operation, the charge of the parasitic capacitance in the previous configuration, the switch off-resistance, output load, etc. The portability problem is due to employing arbitrary topology, and that the problem is poorly specified without including the industrial specifications during the evolution of the AND gate. The mixtrinsic evolution does not solve the problem, but it develops a circuits that are not connected to the power supply, contains four gate terminals that are floating, and one of the input is connected to a current source. Therefore, the behavior of the evolved circuit with mixtrinsic evolution is not predictable, and is not feasible with respect to the industrial requirements.

4.6 Fault Tolerance and Dynamic Environment

Robust systems can deal with problems that are expected to occur, fault tolerant systems can deal with the unexpected problems. Human tend to design robust systems rather than fault tolerant systems. For example, auto zeroing can cope with the input offset. On the other hand, the evolution tend to cope with faults as employing the faulty parts results in declination in the fitness value, consequently, the evolution abstains from it. Thus, it can adapt the system to operate in harsh environment as well. Treating the fault is achieved by two means:

- Start the evolution from scratch [Tho97a, KZJS00] in order to find a new configuration that compensates the fault is a feasible solution exists. If the problem is poorly described by the relation between the output and a reference output, this method is not guaranteed to work well as the fault cells may affect the other characteristics such as the input and output impedance.
- Detecting the fault cells then map the target functionality to the working cells [ESSA00]. This method is more efficient in digital systems as the digital hardware is less sensitive to deviations, and the behavior of all the functional cells is similar. Thus, remapping an old configuration to the functional cells does not change the hardware behavior.

Coping with dynamic environment is achieved by evolving the hardware in the new environment. The state-of-the-art evolvable hardware starts optimization from scratch after any environmental change to find a new configuration that suite the current environment [SKZ01, ZGK$^+$04].

4.7 Summary

Evolvable hardware is aspired to design customized hardware for rapid prototyping to cope with static and dynamic deviations that are described in chapter 2, which is fortunate for the targeted generic organic-computing sensor electronics. FPGAs are sufficient platform to evolve digital systems [Tho96] regarding its flexibility, speed, and applicable design size. Yet, no powerful analog evolvable hardware is available that contains enough flexibility and components for generic sensor electronic systems, and returns hardware with predictable behavior which is essential for industrial acceptance. The state-of-the-art analog evolvable hardware either employs non-programmable building blocks such as FPAAs, hardware with flexibility in its structure level but fixed devices dimensions such as JPL FPTA, hardware that employs a huge common analog bus for very few devices such as PAMA, or hardware that contains no passive components, but employs full flexibility in the width and length –consequently, increases the cost while programming the length is not useful–, which results in consuming a big die area for very few functionalities.

The evolution of the hardware can be done extrinsically, intrinsically, or mixtrinsically [SZK00]. Extrinsic evolution evaluates the hardware by a simulation environment. It can be employed for hardware synthesis [ZPV98], or to find a configuration for an evolvable hardware avoiding danger configurations such as short-circuits [SKT$^+$99]. The synthesis of the hardware by mean of genetic programming targeting invention of topology includes only single or few of the industrial specifications in the state-of-the-art. Thus, it employs arbitrary topology without known complete industrial specifications [KFHBAK97], which results in unpredictable behavior. Extrinsic evolution

targeting a programmable hardware minimizing the error between the output and a reference output to find an arbitrary topology, thus, this configuration may not work when downloaded to the hardware [SZK00]. However, other work in digital systems employs known topologies such as logic gates, or multiplexers and program the connection between them to return a hardware with predictable behavior [CLA02, ACB99].

The state-of-the-art intrinsic evolution suffers mainly from two problems; the first problem is that it designs the hardware topology, which results in hardware with unpredictable behavior, and the second problem is that the evolution problem defined in a poorly specified fashion. It employs a reference output to minimize the error between the actual and the reference output stimulated by a given input signal [SKT$^+$99]. It can consider phase delay as an error, thus, attempts to decrease the phase margin which results in evolving an unstable system, etc. However, the evolved system may not work in simulation [SZK00], as it does not concern the industrial requirement. For example, evolving an AND gate with low input impedance as shown in figure 4.17, while the terminals of the gates of some of the employed transistor are floating, and therefore, the output depends on the charge of the parasitic capacitance due to the previous configuration.

As the simulation of the hardware that is evolved intrinsically may not work in the poorly specified problems with arbitrary topologies, and downloading a configuration of a hardware that is evolved extrinsically to real chip may not work as well , Stoica et al. invented the mixtrinsic evolution to tackle this problem by evolving some individuals intrinsically, and some other extrinsically in the same population. As shown in figure 4.20, the behavior of the AND gate evolved mixtrinsically is not predictable as it has many gate terminals of the employed transistors are floating and the low input impedance. Evolution of the functionality level is done by considering it as a single black-box block and minimizing the error between the output single and the reference output signal [ZSK00, Lan05], which can implicitly include few of the hardware specifications, but does not consider the standard specifications completely. Thus, the state-of-the-art analog evolvable hardware so far is behind the industrial requirements. In order to returning hardware with predictable behavior, the industrial hardware specifications should be included during the evolution of the hardware, and the hardware topology should be constrained standard topologies. Parallel to the proposed extrinsic evolution in chapter 5, Trefzer et al. included few of the hardware specifications in evolving hardware with arbitrary topology [TLSM04], therefore, its behavior is not completely predictable as it invents topologies with partially known behavior. The hardware specifications should be included in the intrinsic evolution as well, which is not concerned in the state of the art. As measuring some of the hardware specifications require special costly measurements setup, new approaches are required to measure the specifications with low cost setup. In order to concern the standard specifications of the functional block completely, hierarchical optimization should be included to evolve the building blocks first, then the functional level as described in chapter 5 in order to return hardware with predictable behavior.

The evolvable hardware copes with dynamic environment by evolving itself in the new environment starting from scratch in order to find a new suitable valid configuration [SKZ01], which returns fault tolerance as well [KZJS00]. Another approach for fault tolerance is to detect the faulty cells and avoid mapping the hardware to them [ESSA00] –which is useful for digital systems as the deviations does not have strong effect on the design–. Starting from scratch after any environmental change is time consuming, while the state of the art evolutionary computation can cope with dynamic environment without losing all the previous information as described in the chapter 3.

CHAPTER 5

Proposed Design Methodology

The aim of the thesis is to evolve organic-computing analog sensor electronics front-end with predictable behavior for rapid prototyping in such a way that gains the industrial acceptance, copes with the dynamic environment, and considers its embedding and integrability. Thus, the industrial specifications are included in the optimization, the standard topologies are employed, and dynamic environment suitable approaches are applied to cope with the non-stationary environment without starting from scratch after any environmental change. The proposed methodology to accomplish those aims is described in this chapter.

The flow of this chapter is as the following; first the sensor electronics reconfigurable hardware that is aspired in this thesis is outlined. Afterwards, evolving it extrinsically and intrinsically is described. As measuring some of the hardware specifications intrinsically requires an expensive setup, while extrinsic evolution cannot include the self-x properties, a novel proposed mixtrinsic multi-objective evolution is described, in which; the hardware specifications are partitioned into extrinsic and intrinsic specification sets according to their features such as sensitivity to deviations, etc., then they are evaluated either extrinsically or intrinsically. Consequently, low cost assessment of the complete industrial specifications is achievable. After that, the aspired design flow of the analog evolvable hardware is described. Finally, the chapter is summarized.

5.1 Aspired Generic Organic-Computing Sensor System

The aim of our group[1] is to build reliable rapid prototyping generic sensor electronics front-end with self-x properties and predictable behavior for industrial acceptance. This thesis is focused on software and algorithmic aspects, and on manipulating the static deviations of system in the deployment phase and the dynamic deviation in the operation phase. On the other hand, a parallel work by MSc. S. Lakshmanan is focused on the hardware aspects.

In order to achieve the self-x properties, the behavior of the hardware has to be calibrated in the loop.

[1] Institute of integrated sensors, TU Kaiserslautern.

This requires assessment unit to extract hardware current specifications, computational unit to search for a useful configuration based on bio-inspired techniques, and a pre- and post-processing unit to communicate between them. Therefore, the aspired organic-computing generic sensor electronics front-end consists of four main blocks as shown in figure 5.1; the optimization unit, pre- and post-

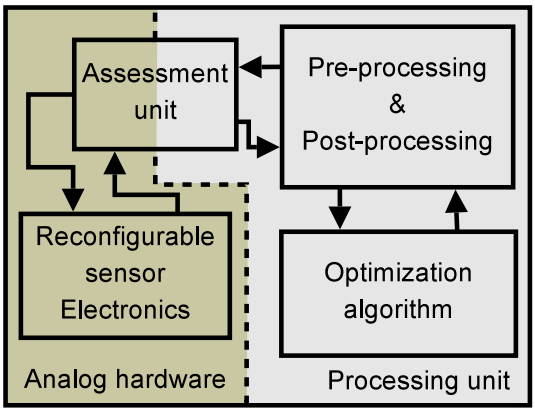


Figure 5.1: Block diagram of the reconfigurable generic sensor system.

processing unit, and the assessment unit, and the reconfigurable hardware.

The optimization unit adapts bio-inspired optimization algorithm in order to evolve the hardware. It contains several evolutionary computation approaches such as several PSO implementations and genetic algorithm in order to find out the optimal algorithm that is suitable for the application. The state-of-the-art evolutionary computation starts the optimization from scratch after any environmental change [Tho97a, KZJS00], which is time consuming approach. Therefore, dynamic environment suitable approaches are included in the optimization unit in order to cope with the non-stationary environment without starting from scratch after any environmental change.

The pre- and post-processing converts the individuals of the optimization unit to bit-stream configurations for intrinsic evolution, to netlist files for extrinsic evolution, or to both for proposed novel approach multi-objective mixtrinsic evolution which is described later on in this chapter. Afterwards, it generates the test signal to measure each of the hardware specifications, and extract the hardware specifications from the measured signals.

The assessment unit contains the necessarily measurement setup for measuring the hardware specifications due to the downloaded configuration. Therefore, the assessment can be achieved intrinsically, extrinsically, or multi-objective mixtrinsically. The extrinsic evolution is assessed by simulated measurement setup. Contrarily, the intrinsic evolution is assessed by real measurement setup. The proposed novel approach multi-objective mixtrinsic evolution is assessed by both real measurement setup, and simulated measurement setup as described later on in this chapter.

In order to return hardware with predictable behavior, standard building blocks are employed, and its standard hardware specifications are optmized. Targeting to involve rapid prototyping of current- and voltage-mode designs, the aspired generic programmable sensor electronic hardware contains the necessarily building block for both the current- and the voltage-mode circuits as shown in figure 5.2. For example, the operational amplifier is voltage-mode building block, while the second generation current conveyor *(CCII)*, the current-differencing transconductance amplifier *(CDTA)*, and the current splitter are current-mode building blocks. In addition, common blocks are used such as multi-output

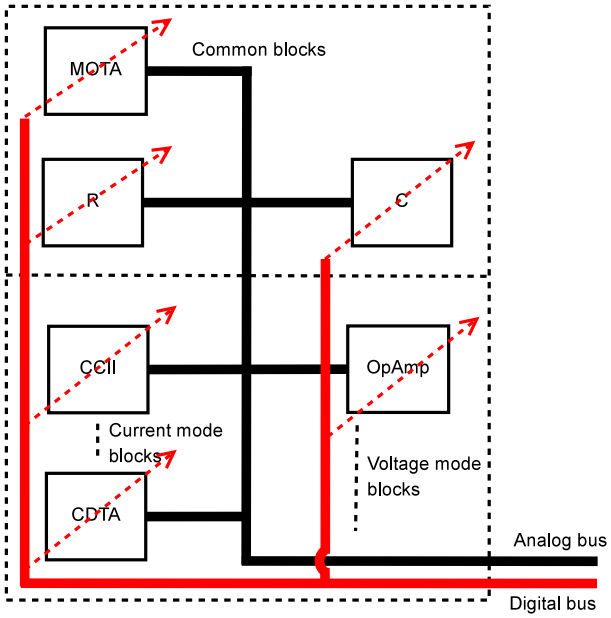


Figure 5.2: The complete aspired target system.

OTA and passive components that can be employed for current- and voltage-mode circuits. A single analog configurable block can be programmed to various building blocks. For example, the reconfigurable block in figure 5.3 that contains an OTA and an output stage can be programmed either as an OTA or as an operational amplifier. The disadvantage of employing many different types of building

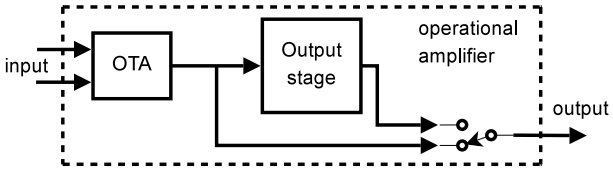


Figure 5.3: A programmable block that can be programmed as OTA or as operational amplifier.

blocks is that it decreases the efficiency of the hardware as only few of them are employed depending on the required building blocks for the design. The state-of-the-art FPAAs employs either voltage mode building blocks such as many commercial FPAA that are based on operational amplifiers as described in section 2.5, etc., or current mode building blocks such CCII [Gau97], etc.

All building blocks are connected to a programmable analog bus that routes them with each other in order to build the abstract level. The topology of the building blocks is constrained to standard topologies, while the dimensions of its devices are programmable. However, only the operational amplifier is implemented and tested by Lakshmanan et al. in a real chip [LK05, LTK06] as a building block, and the instrumentation amplifier is implemented and currently in manufacturing phase as a functional level block, the other blocks are optimized and tested extrinsically. This thesis focus on the algorithmic aspects, therefore, the optimal interconnections between the blocks, and the selection of the entire hardware building blocks is not considered in this thesis.

5.2 Extrinsic Multi-Objective Evolution

State-of-the-art evolutionary electronics adopts the evolutionary computation to invent an arbitrary hardware topology, which means; the behavior of the returned design is not completely predictable, while the industrial specifications and requirements of the target design is not considered during the optimization except in [TLMS05], which is parallel to the proposed work [TLK05, TK05a, TK05b]. Indeed, the error between the required output and the output due to a given test signal is minimized, which cannot guarantee that the returned design is suitable for industrial applications as only a set of the industrial specifications is implicitly optimized without considering its values. For example, if the target hardware is an operational amplifier, the error due to the settling time is usually larger than the error caused by the offset if the input signal is a rectangular wave. Consequently, minimizing the signal error can increase the offset and decrease the phase margin to decrease the error that is produced due to the settling time and the slew rate, which can result in unstable hardware with high offset.

The extrinsic multi-objective evolution synthesize the hardware in the design time without considering the hardware deviations. Many synthesis tools employ optimization methods such as geometric programming [dMHBL01] and genetic algorithm [ZPV98] to dimension the devices of a given hardware topology. Utilizing standard topologies and including the complete hardware specifications are necessary for industrial acceptance. Aiming to evolve the target generic sensor hardware to return hardware with predictable behavior that gains the industrial acceptance, the hardware topology is constrained to standard topologies, while the evolution optimizes the dimensions of its devices. Parallel to this work [TLK05, TK05a, TK05b], Trefzer et al. has included few of the industrial specifications to evolve hardware such as operational amplifier [TLMS05]. However, the evolution has invented the hardware topology. Thus, its behavior may not be completely predictable.

As shown in figure 5.4, simulation tools such as SPICE are employed to evaluate the hardware performance in the extrinsic evolution. The hardware specifications are measured by employing as-

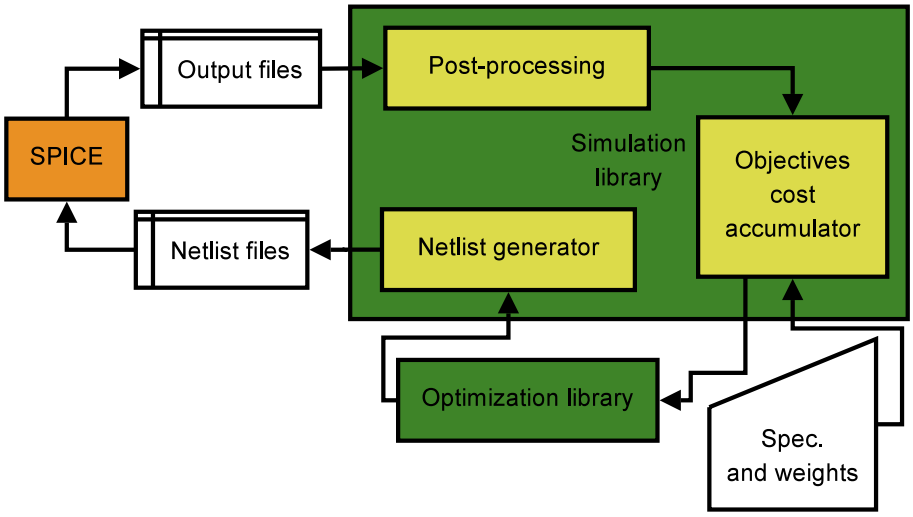


Figure 5.4: Extrinsic reconfiguration environment.

sessment circuits that are written to netlist files and simulated. The outputs of the simulations are written to output raw files, then the output signals of the raw files are post-processed to extract the

hardware specifications. The main advantages of employing a simulation tool over estimation models [Mei96, DGS03, MG05] and formal models [dMHBL01] for the extrinsic approach is that accurate results can be obtained by employing accurate transistor models in the cost of the computational effort, while, modeling the complete hardware behavior is not required.

The hardware specifications are application dependent. For example, the operational amplifier for video applications should be able to operate at high speed, while operational amplifier that is employed for a sensor application such as measuring pressure should have low offset, and high open-loop gain, etc. Thus, the hardware specification values are determined by the user according to the application requirements. As many objectives are optimized, Pareto based multi-objective optimization methods cannot be efficiently employed for large number of objectives as described in chapter 3. Thus, weighted aggregating function based multi-objective optimization is employed to evolve the hardware complete specifications. The required value of the given specification or objective $\mathfrak{o}$ is denoted as $\mathfrak{s}_\mathfrak{o}$. The error of the objective $\mathfrak{o}$ is calculated by equation 5.1, where $m_\mathfrak{o}$ is the value of the objective $\mathfrak{o}$ that the simulation returns, or the measured value for the intrinsic evolution.

$$f_\mathfrak{o} = \begin{cases} 0 & \forall \quad \mathfrak{s}_\mathfrak{o} \quad satisfied \\ \dfrac{|m_\mathfrak{o} - \mathfrak{s}_\mathfrak{o}|}{\mathfrak{s}_\mathfrak{o}} & otherwise \end{cases} \tag{5.1}$$

The specification value $\mathfrak{s}_\mathfrak{o}$ is satisfied if the value of $f_\mathfrak{o}$ is better[2] than the value of $\mathfrak{s}_\mathfrak{o}$. The optimization attempts to fulfill a given specification value for each objective, if the current hardware have some of the specifications better than the requirement, it does not include them in the aggregating function in order to improve the other objectives.

The error functions of all the specifications are accumulated to construct aggregating function the by equation 5.2, where $k_\mathfrak{o}$ is the weight of the objective $\mathfrak{o}$.

$$F_a = \sum k_\mathfrak{o} \cdot f_\mathfrak{o} \tag{5.2}$$

The dynamic deviations can be partially treated extrinsically by employing the available environmental information in the simulation, such as measuring the temperature of the hardware and employing it in the simulation [TK05b].

However, extrinsic evolution cannot treat the static and the dynamic deviations completely. For example it cannot detect the dynamic deviations due to different temperature distribution inside the chip, or self heating during the operation. Thus, it can be employed for rapid-prototyping of sensor electronics hardware without including the self-x properties. In order to include the self-x properties, the optimization should be achieved intrinsically to cope with hardware deviations.

5.3 Intrinsic Multi-Objective Evolution

The intrinsic evolution trim the hardware in the deployment and run time to recover from deviations. The state-of-the-art intrinsic evolution of analog hardware minimizes the error between the output

[2] "Greater" in case of maximizing an objective, and "less" when minimizing an objective

signal and reference output signal to invent blackbox hardware without considering the industrial requirements. Aiming to include the self-x properties to the reconfigurable generic sensor electronics hardware, standard topologies are employed, and the industrial specifications of the hardware are optimized intrinsically [TLK06].

The architecture of the implementation of the intrinsic multi-objective evolution environment is shown in figure 5.5. The blocks marked in gray are implemented in hardware. The configuration is down-

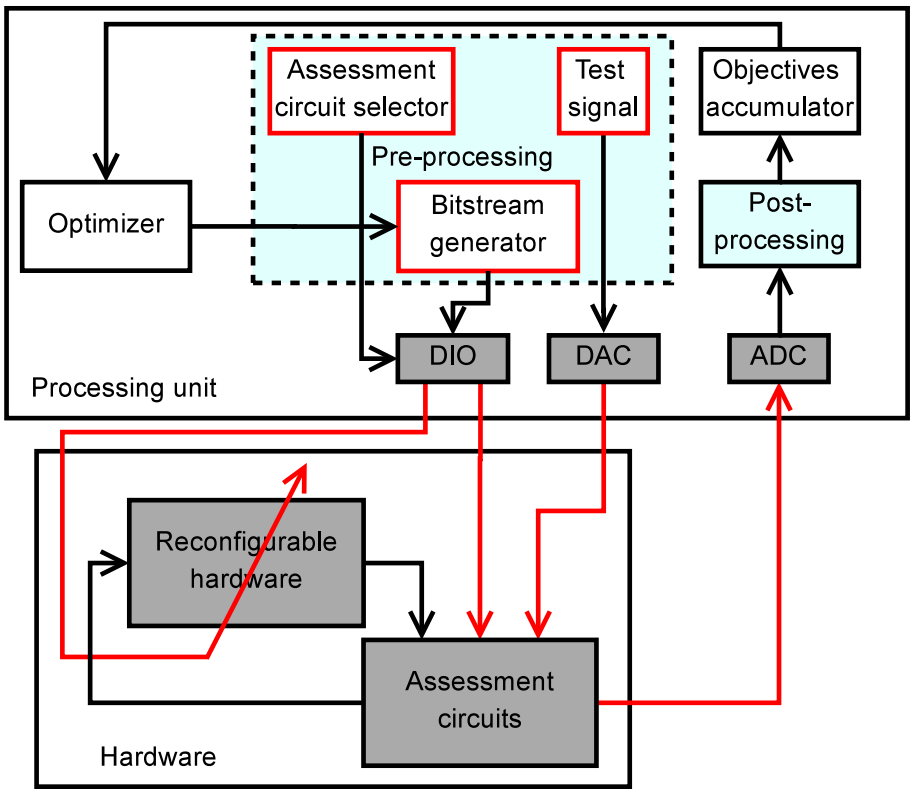


Figure 5.5: The intrinsic evolution environment.

loaded to the hardware in the form of bit stream [LK05] that represents the hardware topology, and the dimensions of the hardware devices. After downloading the configuration to the hardware, the suitable assessment circuit is selected to measure each of the specifications, and the appropriate test signal is generated and fed into the hardware through the DAC *(digital to analog converter)*. Afterwards, the output of the hardware due to the test signal is converted into the digital domain through the ADC *(analog to digital converter)*, and the necessary post-processing is employed to extract the hardware specifications from the hardware output.

The limitation of the intrinsic evolution is that it may utilize many assessing circuit which increases the cost and the hardware area which is not fortunate for embedded systems, while measuring the AC and transient specifications of the target hardware is limited to the ADC conversion rate, and the operation bandwidth of the assessment circuits. On the other hand, some other specifications require complicate assessment circuits. For example, measuring the open loop gain of an operational amplifier requires assessment circuits that can compensate the operational amplifier offset, scale down the DAC signal to precise few micro-volts, etc. Increasing the assessment circuit complexity increases the probability that it deviates, and consequently the whole system.

5.4 Mixtrinsic Multi-Objective Evolution

As described above, the extrinsic evolution cannot cope with all the system deviations, while evolving the system intrinsically requires measuring the hardware specifications with measurement assessment circuits. On the other hand, measuring some of the hardware specifications requires expensive equipment, while measuring some others is time-consuming approach. In addition, employing many measurement devices to optimize the hardware in the loop limits its applicability to the laboratory level as not all the measurement instruments are integratable or embeddable with respect to the cost and the implementation area.

Thus, a novel approach is proposed that divides the hardware specifications into two sets [TK07]. The first set consists of those specifications, which are hard to be measured due to the cost/time requirements, and are less sensitive to the deviations such as open-loop gain, phase margin, output resistance, etc. This set of specifications is evaluated extrinsically. If the target specifications overfulfill the application requirement, the influence of deviations on this set of specifications does not lead to any dramatic specifications variation.

The second set of specifications contains the specifications that are sensitive to deviations and can cause direct distortion in the signal, while they are easy to measure at low cost such as offset, swing output voltage, common mode range (CMR), etc. For example, the change of the CMR due to deviations can cause signal distortion, and the offset is very sensitive to deviations and cannot be handled with simulation. This set of specifications is measured intrinsically. The optimization criteria are multi-objective optimization where each individual has intrinsic and extrinsic objectives as shown in figure 5.6.

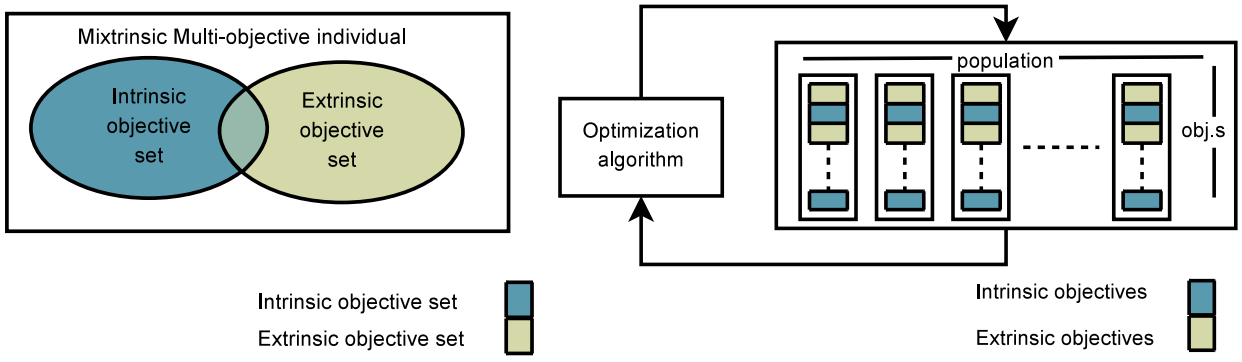


Figure 5.6: The mixtrinsic multi-objective evolution a) A single individual. b) The complete population.

The two sets of specifications may intersect with each other, which means that some of the specifications can be treated intrinsically and extrinsically. In this case, the required value of the intrinsic and extrinsic measurement can be different. For example, the settling time of operational amplifier can be optimized intrinsically to ensure the stability and the basic operation of the amplifier, while it is optimized extrinsically as the intrinsic optimization of the settling time is limited to the ADC conversion speed, therefore, the operational amplifier may become stable within a single sample of the analog to digital converter.

The measurable environmental data such as the hardware temperature can be measured and employed in the evaluation of the extrinsic specification set. Therefore, the complete hardware specifications are optimized at low cost, with integratable, and embeddable calibration loop.

The term mixtrinsic evolution [SZK00] means the population contains intrinsic and extrinsic individual as described in chapter 4. The novel proposed mixtrinsic multi-objective evolution employs both intrinsic and extrinsic evolutions for each individual, but for measuring different objectives.

The extrinsic objectives can be evaluated by employing simulation environment [LMU04], formal models for the hardware [dMHBL01] or estimation models [Mei96, DGS03, MG05]. The outlines of their models for mixtrinsic evolution are described in the following subsections.

5.4.1 Simulated Models

The mixtrinsic multi-objective arrangement based on simulated models is shown in figure 5.7. For a

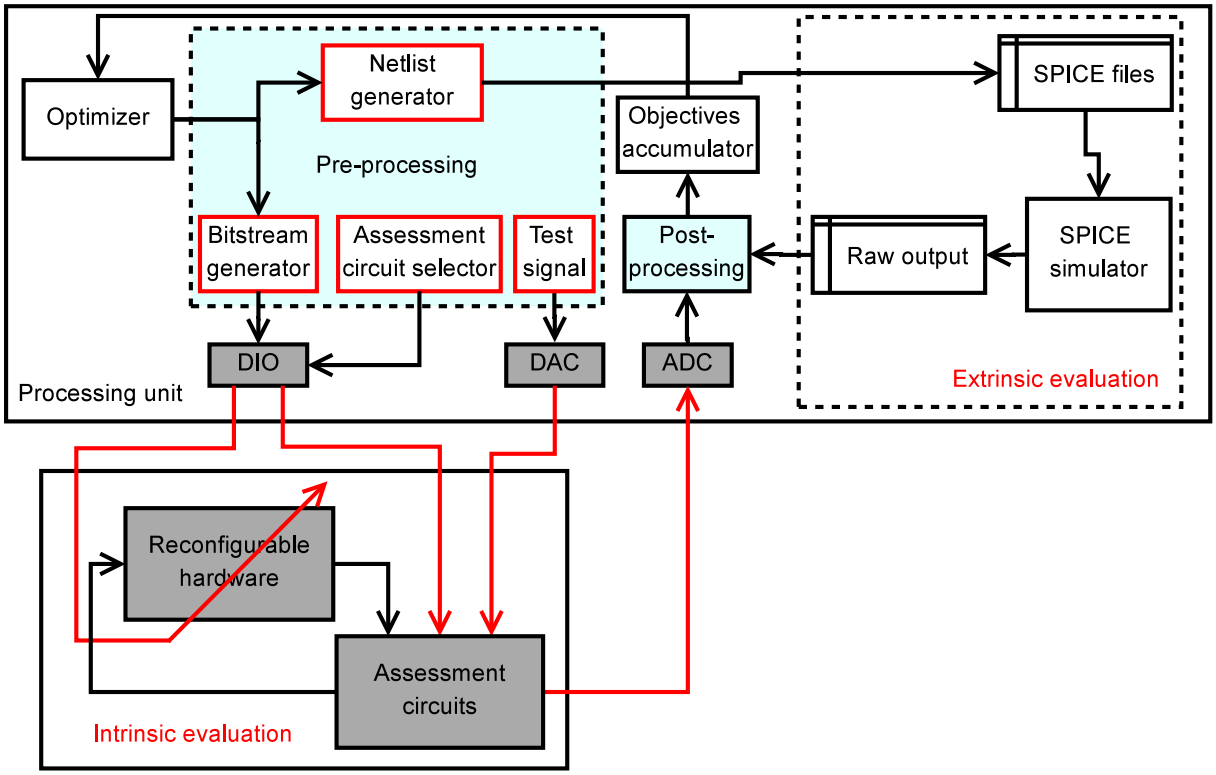


Figure 5.7: The mixtrinsic evolution.

given configuration, the hardware is simulated to extract specifications of the extrinsic set of objectives, and the other set is measured intrinsically. In order to evaluate the extrinsic set of objectives, the suitable netlist files are generated, and then simulated by a simulation environment such as SPICE. The simulation environment writes the results of the simulations to RAW files. Afterwards, the results of the simulations are post-processed to extract the specifications of the target configuration. The hardware temperature can be measured and included in the simulation, and accurate transistor models can be employed such as BSIM 3.3, etc., which is more accurate than then formal models and the

estimation models but requires more computational effort. Depending on the precision of the assessment circuits and the effect of their susceptibility to noise, the simulated models can be more accurate than intrinsic measurements for some specifications. For example, measuring the open-loop gain can be more accurate extrinsically regarding the resolution of the ADCs, the noise effect on the hardware during the measurement, the bandwidth of the assessment circuits, etc. The main disadvantage of the simulated models is their computational effort that consumes time and power.

5.4.2 Formal Models

Embedded systems have less resources than the PCs. Running tasks that requires high computational effort such as simulation tools is not desirable in embedded systems with medium resources, and not feasible for embedded systems with low resources. In order to determine the hardware specification values with low computational effort and less hardware resources, lean models can be employed. Designing the formal models requires complete knowledge about the utilized technology and the target topology. It obligates modeling the hardware topology for large and small signal analysis. Although designing the formal models requires high design effort, it is worth making them as they reduce strongly the computational effort for each instances (individual in the population). In many sub-circuits, numerical analysis is needed in order to solve the large signal analysis, while analytical expressions are sufficient for small signal analysis as shown in figure 5.8. The numerical analysis may

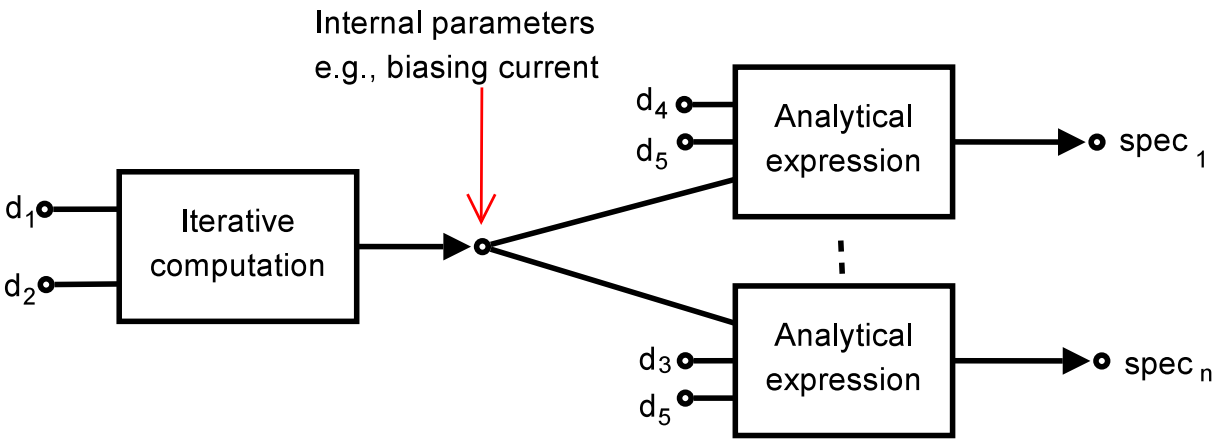


Figure 5.8: Determining the hardware specifications by formal models.

converge to non-optimal solution, or may diverge for some configuration as the formal models employ simple transistor models. Including the temperature in the formal models increases their design effort, thus, it is not preferable.

Hybrid models can improve the precision and guarantee the convergence of the internal sub-circuits, while they require less computational effort than the simulated models. For example, the large signal analysis such as biasing current and voltage should be accurate and may require iterative loop, thus, it can be simulated. On the other hand, the AC specifications, etc. can be calculated by a single analytical expression, therefore, a formal model can be employed as shown in figure 5.9, where ϑ is the environmental temperature. The hardware temperature can be included in the simulated part in the hybrid model.

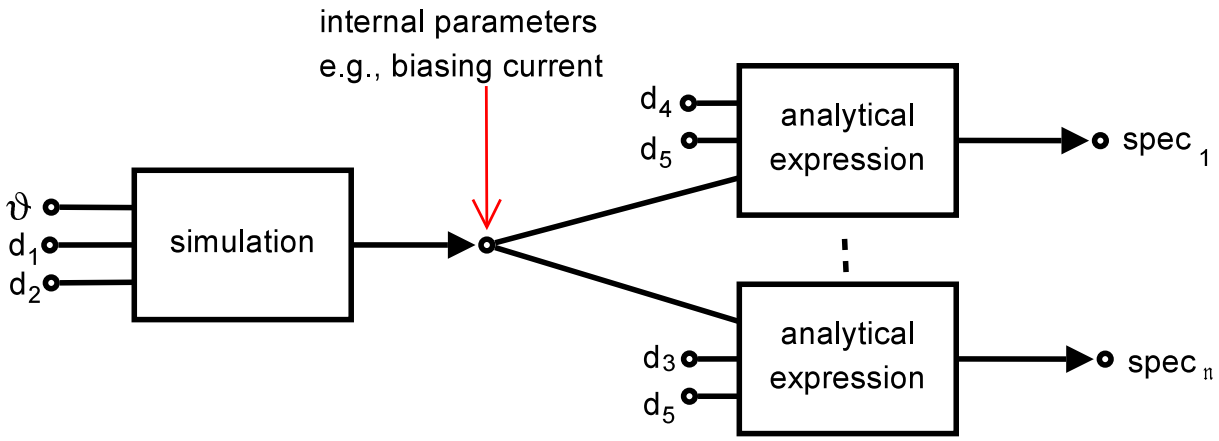


Figure 5.9: Determining the hardware specifications by a hybrid simulated/formal model.

An example of an operational amplifier that employs 10 components is shown in figure 5.10. The dimensions of the first two components are utilized in the biasing circuit, therefore, only the biasing circuit is simulated. Afterwards, the transconductance of all the transistors are computed, then the poles of the amplifier and its specifications.

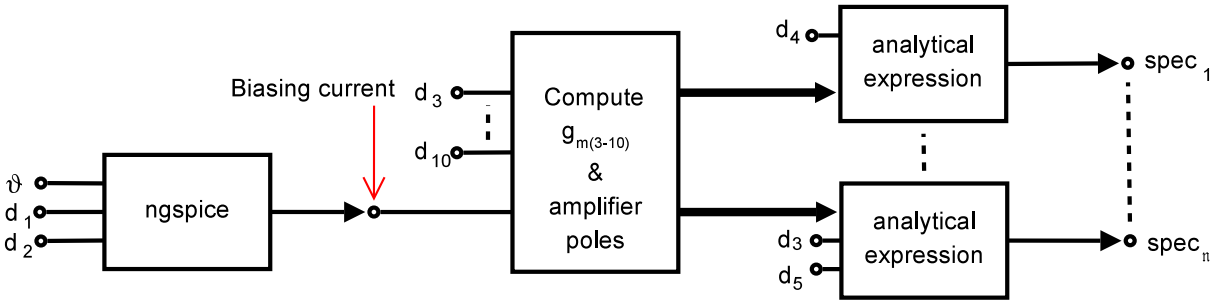


Figure 5.10: Determining an operational amplifier specifications by first simulating the biasing current at a given temperature, compute the transconductance of all the transistors, and the poles of the amplifier, then employing lean models to calculate its specifications.

The main drawback of the formal models is that they require complete modeling of the target hardware, where they employ simple non-accurate transistor modules in order to simplify the modeling. Besides, it requires numerical optimization to solve the large signal analysis in many circuits. The estimation modules in the next subsection cope with these drawbacks as described below. Nevertheless, commercial tools are available such as Analog Insydes [Ana] to extract the symbolic model that describes the hardware, which simplifies the design of the formal models.

5.4.3 Estimation Models

Aiming to avoid the necessity of complete knowledge about the hardware technology and topology, estimation models can be employed, which do not require knowledge about the hardware technology and topology, but indeed they are trained before being employed in evolution. As shown in figure 5.11

a data set generator is employed to produce several configurations. These configurations are simulated

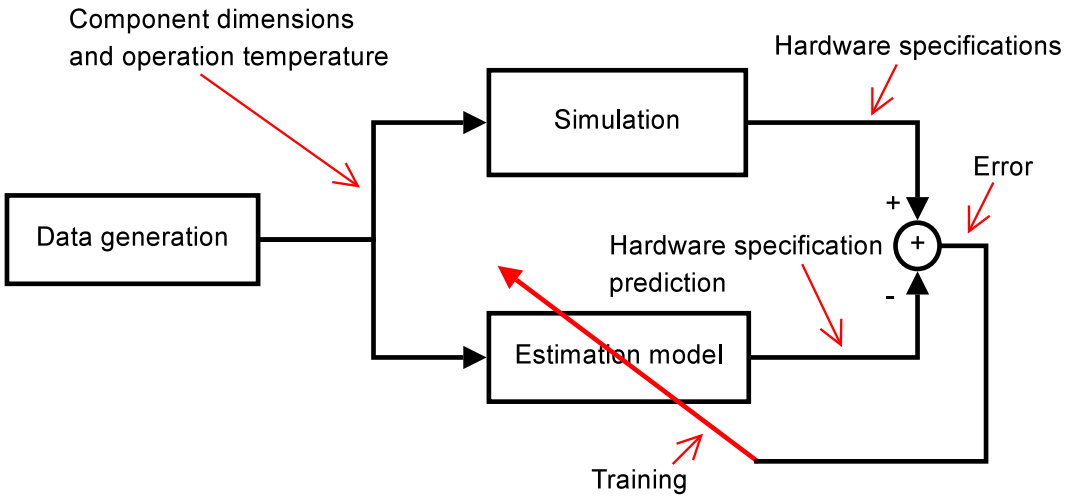


Figure 5.11: Estimation models training.

and its specifications are extracted in the training phase. Accurate transistor modules such as BSIM 3.3 can be employed to obtain accurate training data. In order to estimate the hardware specifications at different environmental settings such as different temperatures, the various environmental settings are included in the training set. Afterwards, estimation models are trained off-line by minimizing the error between the estimated and the simulated specifications when the same configurations are fed to both. Ultimately, the estimation models are employed to estimate the extrinsic specifications of the hardware. The estimation models are employed in the state-of-the-art analog synthesis tools, such as; neural networks [Mei96], automatically created posynomials [DGS03], support vector machines [KG04], splines [MG05], genetic programming [MG05], etc. An example of the estimation models using the neural networks is shown in figure 5.12, where $\mathfrak{n}$ is the number of hardware extrinsic specifications.

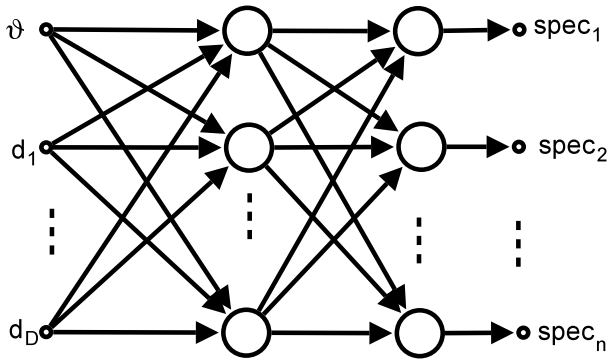


Figure 5.12: Neural network as an estimation model.

In order to obtain estimation models that are accurate for most of the configurations, a sufficient training data set are demanded. The smaller the number of configurations used in the training of the models, the more it tends to be specialized to the given data set rather than the hardware itself. The larger the number of configurations that are engaged in the training, the more it is hard to train the estimation models. Thus, the relevant inputs for each of the specifications can be clustered and the

estimation model can be partitioned into smaller models by a clustering algorithm, or by employing a priori knowledge about the topology to find the relative effective components for each of the specifications. For example, an estimation model that employs neural network is shown in figure 5.13. It is partitioned in which, first the large signal parameters such as biasing currents and voltages are

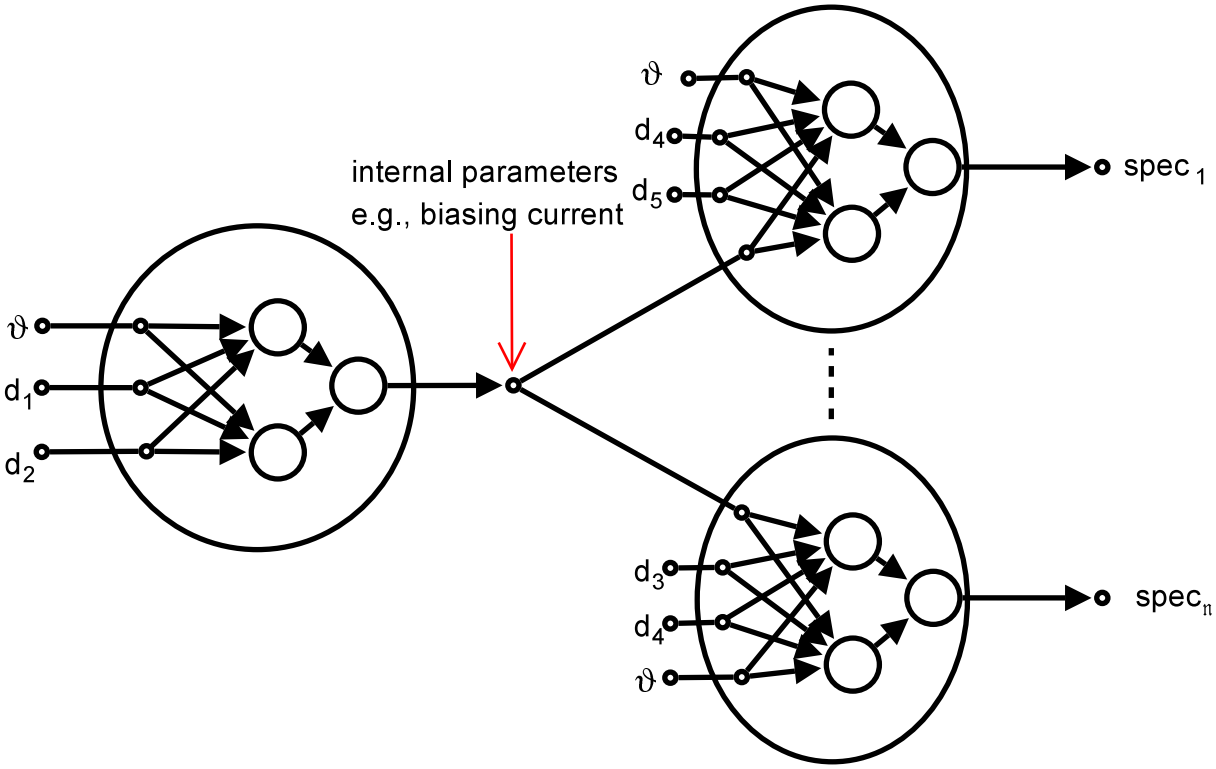


Figure 5.13: A neural network specification estimation model that is partitioned smaller models by employing a priori knowledge.

estimated, then these biasing signals and the relevant device dimensions are used as input signals to small estimation models that determine the value of each of the specifications.

Employing the estimation models can be more accurate than the formal models, especially if iterative computations are required as the formal models utilize simple non-accurate transistor models, while it requires more computational effort to solve the numerical optimized problem that works iteratively. If the topology model is well known, a hybrid approach can be engaged in which, internal signals that require numerical optimizations are evaluated by estimation models to obtain more accurate results at less computational effort, then analytical expressions are employed to extract the hardware specifications at less computational effort. The example in figure 5.13 is advanced to the model in figure 5.14 that estimate the hardware internal signals by a neural network estimation model, then apply the formal models to extract the hardware specifications.

5.5 Dynamic Environment and Fault Tolerance

The deviations due to the dynamic environment are treated in the state-of-the-art evolvable hardware by starting from scratch after any environmental change, which is time consuming process as it loses

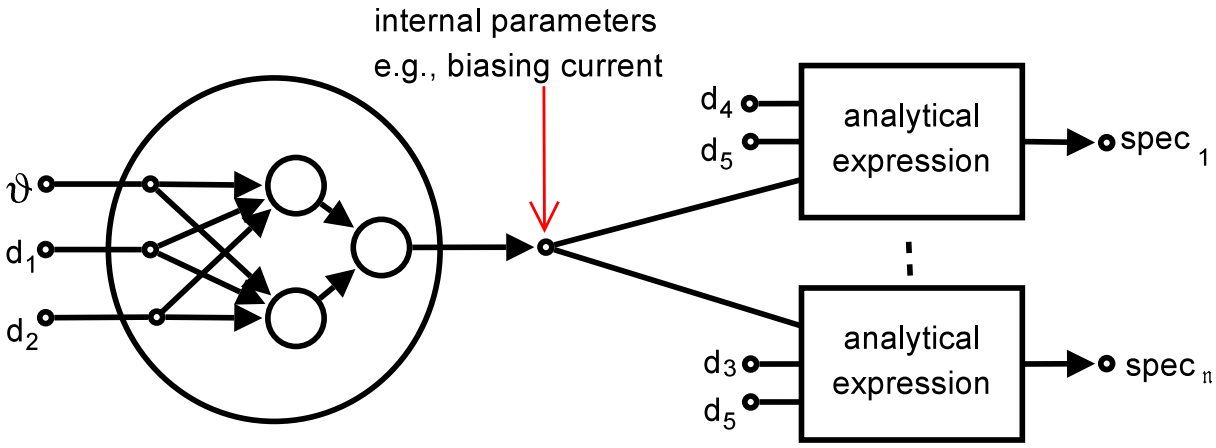


Figure 5.14: Extracting the hardware specifications by a hybrid estimation/formal model.

all the previous information. In order to overcome this problem, dynamic environment suitable approaches are employed [TLK06] as described in chapter 3.

The fault tolerance capability depends on the programmable components design, which is a trade-off between mitigating the faults, and adding extra-flexibility that increases the die area and reduces the operation frequency of the hardware. For example, in [LK05], Lakshmanan et al. has proposed a programmable transistor module that is shown in figure 5.15, any faulty transistor can be completely disconnected from the design.

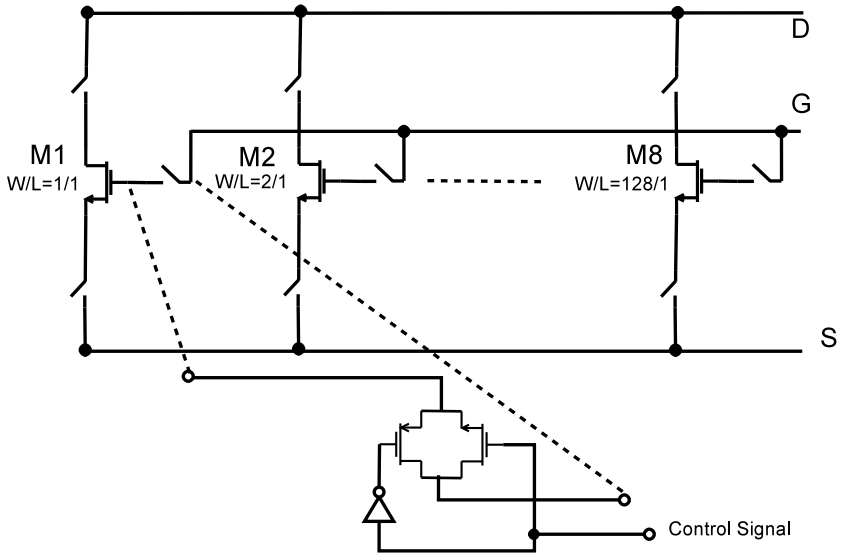


Figure 5.15: A programmable NMOS transistor [LK05] of the FPMA.

Regarding the voltage drop that can occur on the switches and the frequency behavior, Lakshmanan et al. has employed another solution in [LK07] that controls only the gates of the transistors as shown in figure 5.16. On the other hand, its flexibility does not allow disconnecting the faulty transistors completely as in [LK05]. Thus, some fault cases cannot be treated such as stuck-at-on faults, which turns the faulty transistor on without considering the voltage level applied at the transistor gate.

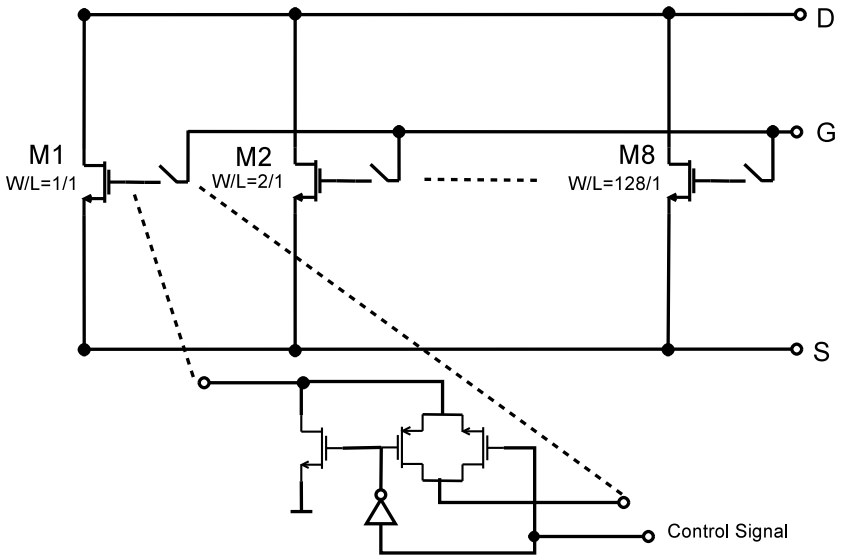


Figure 5.16: A programmable NMOS transistor [LK07] of the FPMA2.

From the algorithmic point of view, employing faulty transistors results in bad fitness values. Thus, the optimization tends to find a working solution that fulfills the given specifications by suppressing the configurations with bad fitness due to using faulty transistors.

5.6 Design Flow

A rough proposed design flow of the aspired generic organic-computing sensor electronics is shown in figure 5.17. The assignment of each numerated steps in the graph is as the following:

Step 1 (entering the target design): The user enter his target design in different abstract levels using schematic entry environment, or hardware description language (HDL) such as VHDL-AMS or verilog-AMS.

Step 2 (convert the design to intermediate HDL): The various entry formats are converted to intermediate HDL for further processing.

Step 3 (extract the functional blocks): The hardware should be partitioned to functional blocks, in which, the required specifications of each of them can be achieved by the available functional blocks in the hardware, such as, function decomposition of a given function in order to constrain the returned functional block requirements to be achievable [WV02]. For example, decomposing a fourth order filter into two second order filters.

Step 4 (topology selection): Specifying the hardware topology depends on the specifications that each topology can achieve and the required specifications. Therefore, knowledge about the specification values that each topology can accomplish is needed for the selection. If the user chooses the specifications from the database, the topology that is suitable for this setting can be mapped directly. In order to avoid the necessity of expert knowledge, classification models such as neural networks [PF98] can be utilized to choose the hardware topology. However, the topology of the unused blocks should be considered during the topology selection.

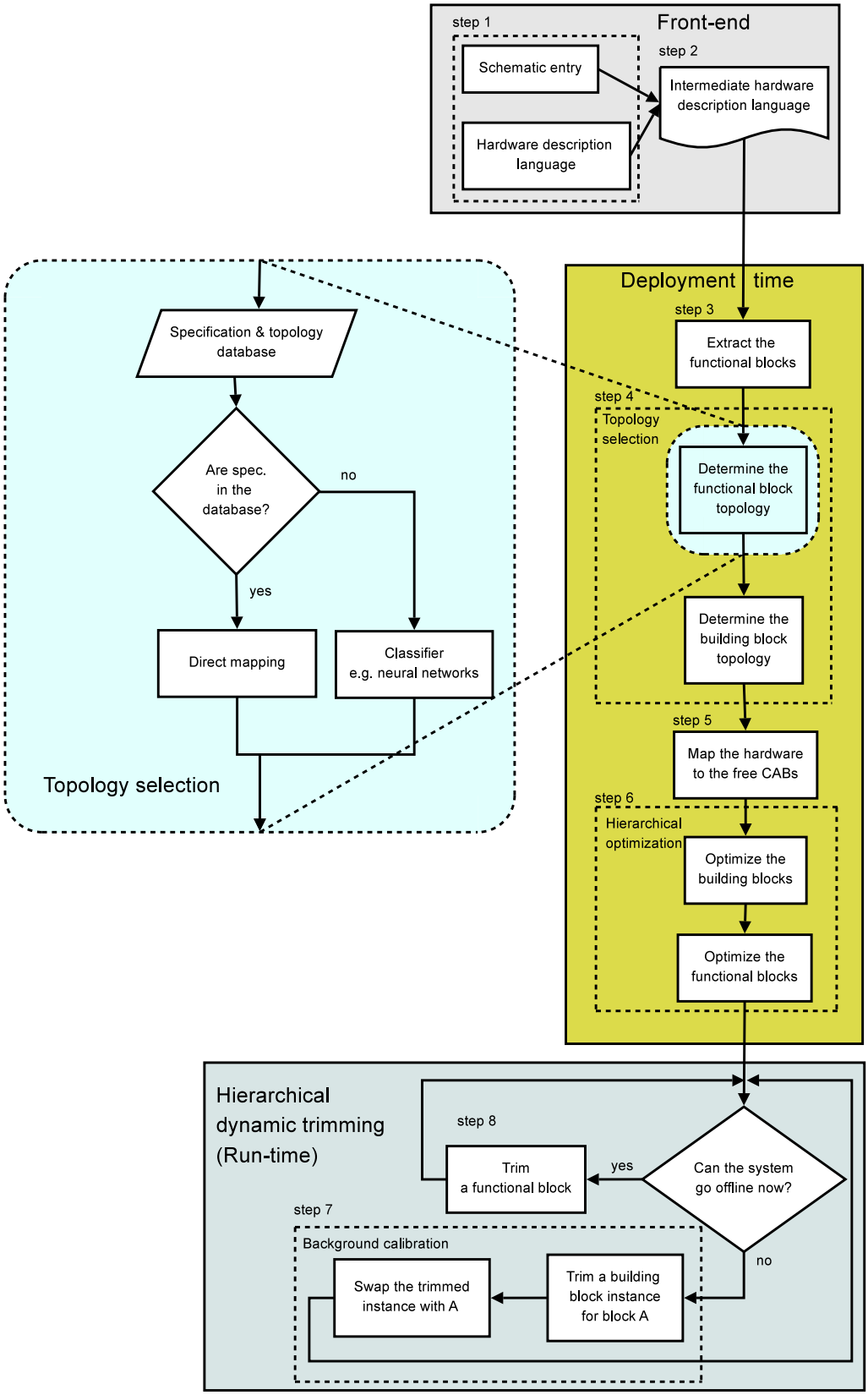


Figure 5.17: The proposed design flow for generic organic-computing sensor electronics.

The topology selection is done in both the functional and the building block level. First the topology of the functional blocks is selected, then the topology of its internal building blocks are choosen.

If the topology of a hardware building block is programmable, all the relevant block transistors and switches can be employed in the optimization in which; the evolution dimensions the transistors, and select the topology by controlling the routing switches. However, such representation engages a very large searching space, and thus, it is hard to find a good solution.

Step 5 (mapping and routing): After the topologies are selected, the hardware blocks should be mapped to the unused blocks. The placement and routing should consider the parasitic capacitance during the mapping. Related work is proposed in [WV02, BRL$^+$05] for FPAA applications. If no enough free resources, error massage should be produced.

Step 6 (hierarchical optimization): Hierarchical optimization, –also called multi-objective bottom-up methodology [GME05]– is already employed in synthesis tools [GME05], in which, it partitions the system to components from the lowest level such as building blocks, to the higher level such as the functional level circuits. Then it dimensions them starting from the lowest level building blocks, then the upper level such as an advanced building block, or a functional level.

Inspired from the hierarchical optimization that has been developed in the synthesis tools, the hierarchical optimization is abstracted to size the generic organic-computing sensor electronics, in which, first the building blocks are dimensioned, then the next level of abstract such as the functional circuits [TK06b], or more advanced building blocks is optimized.

For example, the operational amplifier can be combined with passive components to build voltage-mode a filter or an instrumentation amplifier as shown in figure 5.18. First the operational amplifier is optimized, and then the filter, or the instrumentation amplifier is evolved. The reconfigurable

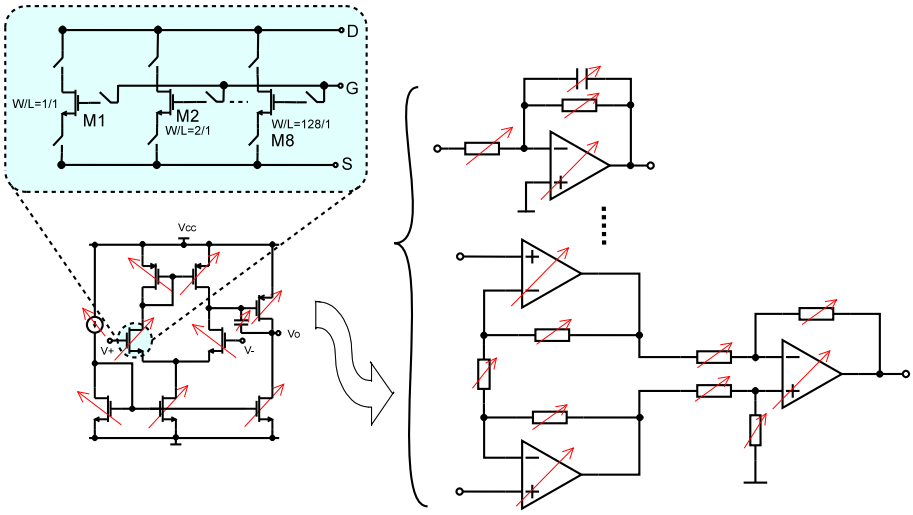


Figure 5.18: The hierarchical optimization of evolvable hardware, building functional block level hardware such as filter and instrumental amplifier from programmable operational amplifiers.

instrumentation amplifier is already designed by Lakshmanan et al., and it is currently in the manufacturing phase. The CCII block can be combined with multi-output OTA block to build a CDTA block [BVB05] as shown in figure 5.19, etc. Therefore, the multi-output OTA and the CCII are optimized separately.

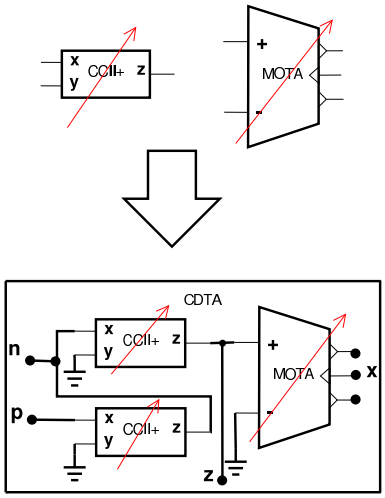


Figure 5.19: Building CDTA from CCII and MOTA.

The main advantage of the hierarchical optimization over the flat optimization which optimizes the whole system at once is that the behavior of each component is optimized to a given required specification set that guarantees that it has predictable behavior, and therefore behavior of the complete system is predictable. Additionally, the search space is reduced by partitioning the problem which simplifies the optimization problem [GME05].

Step 7 and 8 (hierarchical dynamic trimming): In this step, the and further deviations that are introduced into the hardware during the operation is compensated. However, the reoptimizing all the building blocks is time consuming, which may result in going offline for long periods. Therefore, background calibration can be achieved by using additional building blocks instances in which a building block is optimized in the background and replaced with a non-calibrated block [JW98, TK07]. For example, in figure 5.6, the block **A** is replaced with the trimmed block **B** then the optimization of the block **A** is started. The same principle can be employed in the functional

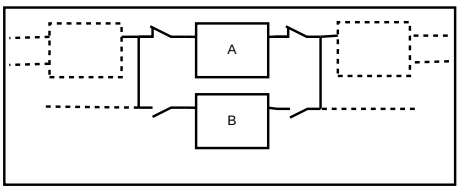


Figure 5.20: Outlines of background calibration of sensor electronics hardware.

block level as long as enough resources are available. However, if the functional block utilizes many resources, and not enough resources are available for trimming it in the background, only the passive components need optimization in the foreground as the other building blocks can be optimized in the background one by one. The passive components of the functional blocks can be initialized using the design procedures [TK06b] in order to recover from the deviations after few iterations. Nevertheless, the initial values of the passive components are often efficient enough at the operation point if the design procedures are employed. Swapping the blocks adds noise to the output signal due to the switching activity and the phase shift between the two building blocks. Therefore, the techniques that can reduce the noise should be used, such as, swapping the blocks in the period between the sample and hold of two consequent samples if the signal is converted to the

digital domain as shown in figure 5.6, or employ a zero detect circuit and swap them when the output signal level is zero. All the building blocks are optimized on the background one by one, while

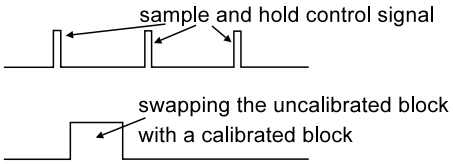


Figure 5.21: Synchronizing the blocks swapping with the sample and hold of the ADC.

the large functional blocks are optimized in the foreground when the system is in standby mode. The population of each of the building blocks is save in order to keep the previous information of the evolution, which is necessary to prevent the evolution from starting from scratch.

5.7 Summary

Analog evolvable hardware has emerged in the last decade to support rapid prototyping analog hardware that includes the self-x properties such as self-healing, self-configuring, etc. However, the state-of-the-art analog evolvable hardware have not considered the industrial requirement to build the hardware as described in chapter 4. On the other hand, the synthesis tools cannot cope with all the deviations as hardware specifications are evaluated extrinsically, therefore, employing synthesis tools to reconfigure the generic organic-computing sensor electronics such as the tools employed in FPAAs [WV02, BRL$^+$05] returns hardware that does not include the self-x properties.

In this thesis, the design flow for the aspired generic organic-computing sensor electronics is presented. It employs known hardware topologies, and optimizes its standard specifications in order to return hardware with predictable behavior. However, including the complete hardware specifications intrinsically requires expensive equipment that limits the approach to laboratory level only. Therefore, a novel proposed scheme partitions the hardware specifications according to their sensitivity and measurement requirements, in which, the specifications that are highly sensitive to deviations are evolved intrinsically, and the rest of the specifications are measured extrinsically, which allows the embedding of the aspired generic organic-computing sensor electronics. In order to cope with the dynamic environment without restarting the optimization after any environmental change, dynamic environment suitable approaches are employed [TLK06]. Instead of selecting the hardware topology by expert knowledge [ZPV98, dMHBL01], classification models can be employed to detect the useful topology for the required specifications as done in synthesis level [PF98]. Hierarchical optimization is abstracted from the synthesis tools [GME05] as well to optimize first the building blocks, then employ them in the functionality level. In order to ensure that the system does not go offline for long time after environmental changes, background calibration can be employed.

Therefore, the proposed methodology affords generic organic-computing sensor electronics that requires low cost integratable assessment for the trimming loop, which is essential for embedded system applications.

CHAPTER 6

Experimental Work and Case Studies

The contributions of the experimental work are for the marked in gray blocks in the presented design flow of the aspired generic organic-computing sensor electronics in figure 6.1, which are the hierarchical evolution of the generic sensor electronics considering the industrial requirements, its dynamic trimming, and its assessing. Therefore, the calibration of the building blocks in static environment for the deployment time, and the dynamic environment for the run-time are demonstrated. These building blocks are employed afterwards in the hierarchical optimization of the functional blocks. The operational amplifier is selected as a case study for voltage-mode sensor electronics as it is commonly used in many applications. As a current-mode case study, the second generation current conveyor is selected as it is a well known current mode building block, and employed in many applications such as building FPAAs [Gau97]. As the programmable operational amplifier is already implemented in [LK05], it is demonstrated for the dynamic reconfiguration by the novel proposed approach mixtrinsic multi-objective evolution, which assesses the optimization of the hardware against dynamic deviation at low cost setup, while it considers the industrial requirement in the optimization. The selected case studies for hierarchical optimization are the filters as it is necessary in many sensor electronic applications, and the 3-bit flash ADC, which is published in advance [TK06b]. The functional level blocks are optimized extrinsically as the target functional level blocks are not available currently in a real hardware.

6.1 The Reconfiguration Environment

A block diagram of the implemented reconfiguration environment is shown in figure 6.2. It is a modular toolbox for evolving the sensor electronics, that can be extended by adding any hardware block to it. It can evolve the hardware in the deployment phase by the mean of extrinsic evolution, or in the run-time by the mean of intrinsic and mixtrinsic multi-objective evolution. Models of dynamic and static variations that can occur in each block can be included to evaluate the hardware behavior in various operation conditions. The optimization library in figure 6.2 contains several PSO approaches, and it links GAlib [GAl] that contains several GA operators and approaches to investigate the various optimization methods regarding their speed, and consequently, the power consumption in the embed-

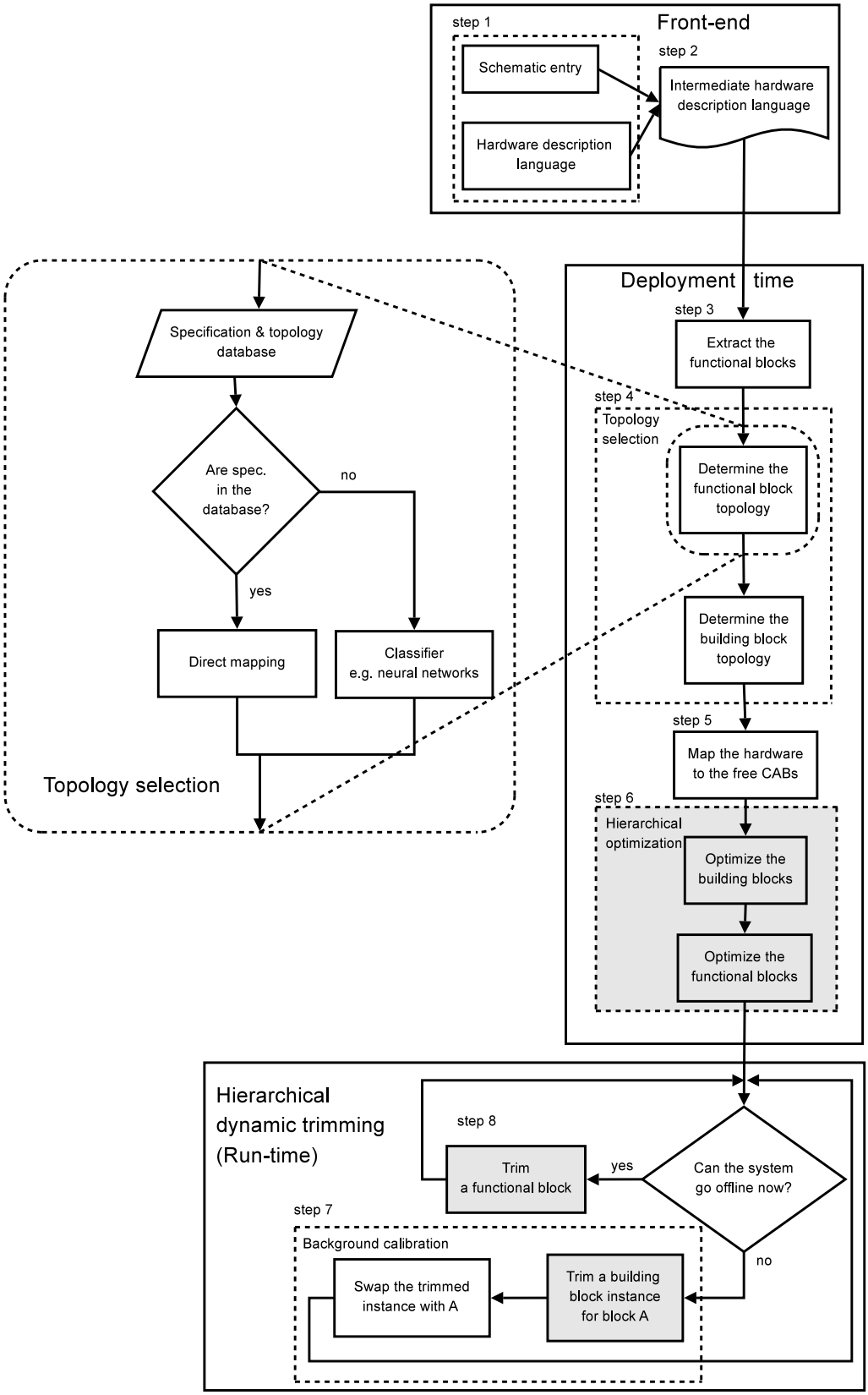


Figure 6.1: The proposed design flow from chapter 5 for the aspired generic organic-computing sensor electronics, the contributed blocks are marked in gray.

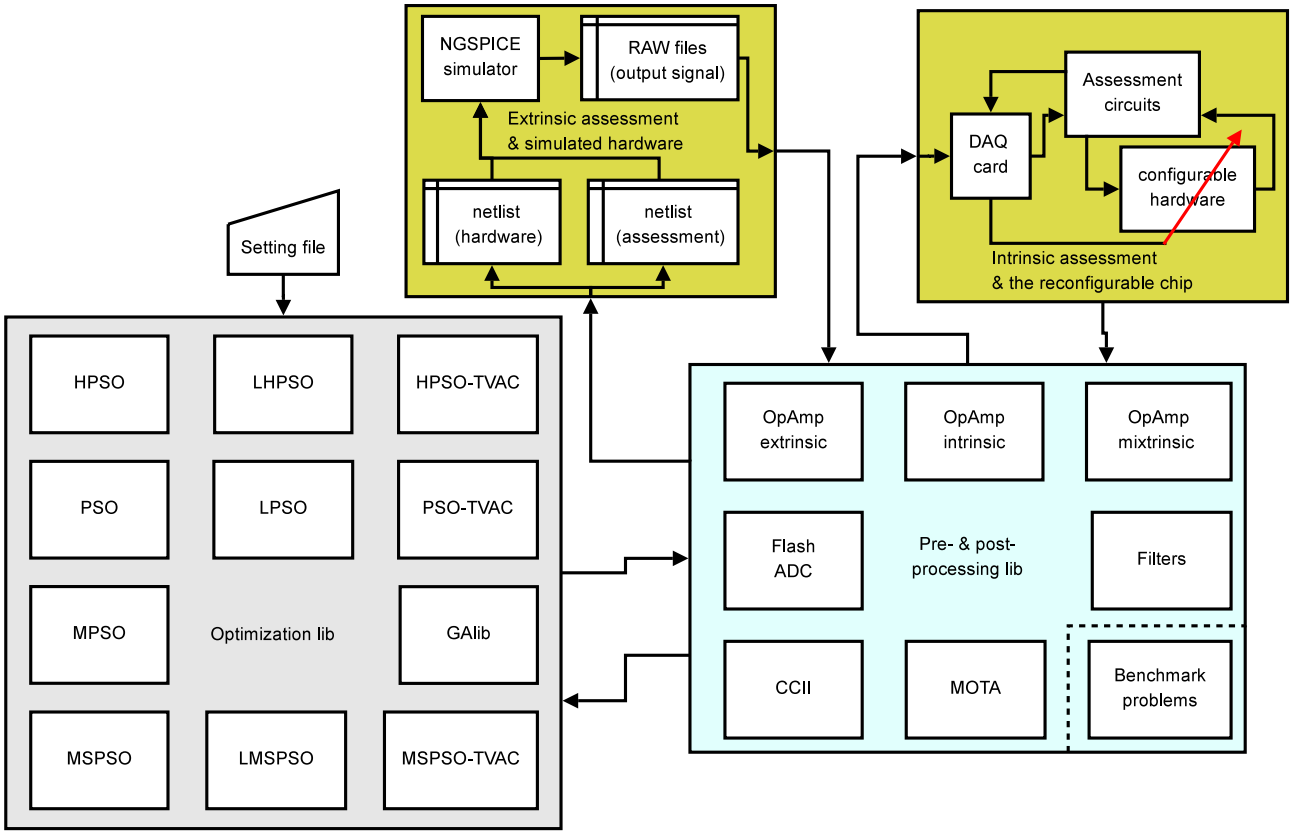


Figure 6.2: Block diagram of the implemented reconfiguration environment.

ded system. The pre- and post-processing library contains the necessary pre- and post-processing of several building and functional blocks. The currently implemented building blocks are operational amplifier, second generation current conveyor, multi-output operational transconductance amplifier. In the functional block level, 3-bits flash converters, and filters are implemented. Only the operational amplifier is available in a real chip, all the other blocks are evolved extrinsic in the current state of the research. The pre- and post-processing library is connected to a simulation environment, and to the reconfigurable chip. It generates netlist files that contain the assessment circuits, the configured hardware, and the deviation models to extract the specifications of the target configuration by running a simulation tool such as NGSPICE. A data acquisition (DAQ) card is employed to download the configuration to the chip, to select the assessment circuit, to generate the test signal, and to convert the output signal to the digital domain. The reconfigurable environment allows installing the DAQ card in another PC, and communicates with it through a network socket that is developed for the hardware level communications. The design flow is not fully automated in the current tool implementation.

The experimental setup for the intrinsic multi-objective evolution is simplified in figure 6.3. It consists

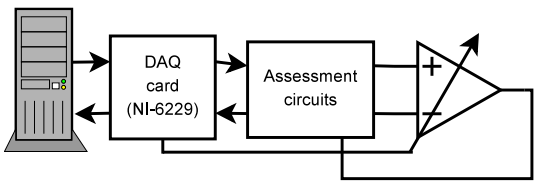


Figure 6.3: A simplified block diagram of the intrinsic evolution prototype.

of the processing unit that runs the evolutionary algorithm and the necessarily post-processing, a DAQ card as an interface between the computation unit and the chip, calibration circuits to measure the hardware specifications, and the evolvable chip. The utilized DAQ card is NI-6229 from National Instruments. The reconfigurable chip is currently the programmable operational amplifier described below.

The intrinsic evolution is mapped to the embedded system in figure 6.4. It employs XScale 400MHz processor, which has the core of the fifth generation of the ARM architecture. Its data bus width is 32 bits. The embedded system contains 64MByte RAMs, LAN, USB, serial port, etc. The intrinsic environment is compiled to the ARM architecture by ELINOS, which compiles a linux kernel as well for the embedded system. As the DAQ card is installed in another PC, the embedded system communicates with the other PC using a network socket as described above. The embedded system supports mounting an NFS (Network File System) as its root directory during the development. Therefore, the evolution environment is compiled and copied to a host machine, and the embedded system uses the NFS to communicate with it. The embedded system gets the kernel image through TFTP (Trivial File Transfer Protocol). The user interface is through a serial console, which employs the serial port as a console on a host machine.

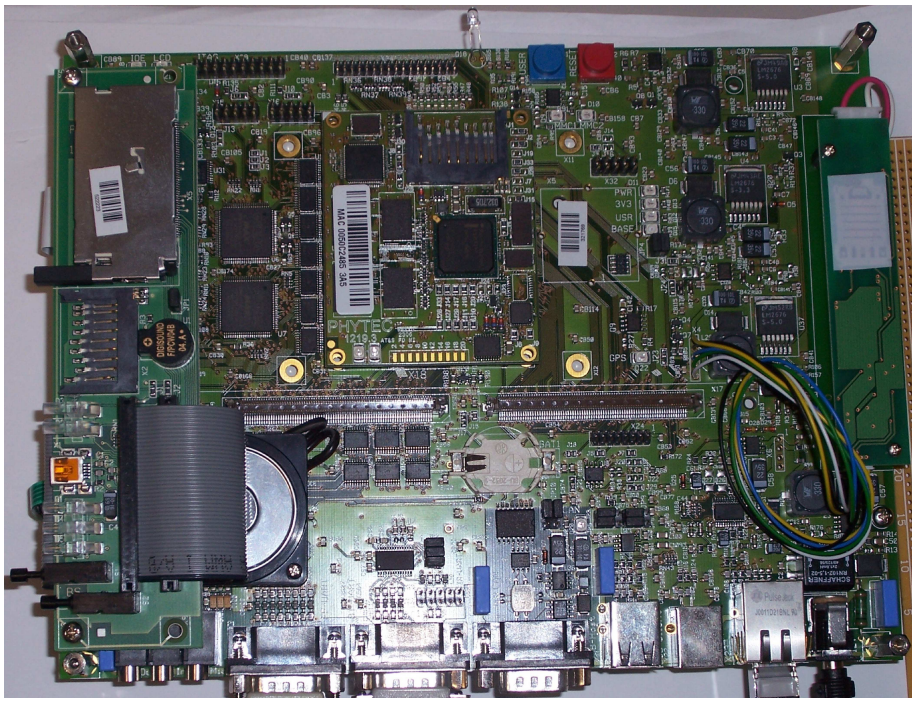
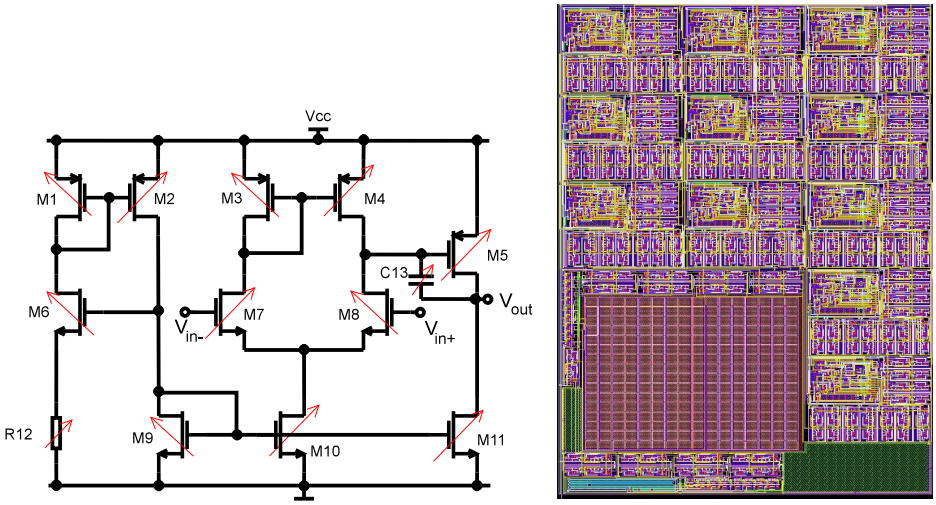


Figure 6.4: Photograph of the target embedded system.

6.2 Baseline Hardware

The baseline of the intrinsic hardware evolution is an operational amplifier with programmable dimensions that utilizes standard topologies, which is proposed by Lakshmanan et al. [LK05]. The operational amplifier is the basic block of many sensor electronic applications. They have implemented Miller and folded-cascode topologies. The Miller operational amplifier consists of 13 devices, while

the folded-cascode operational amplifier consists of 25 devices. The experimental work focuses on Miller topology operational amplifier as it is well known and investigated, and as the folded-cascode search space is large, and therefore, its optimization consumes more time. In figure 6.5, the schematic and the layout of a programmable Miller topology operational amplifier [LK05] is shown. The enumeration of the devices in figure 6.5(a) is according to the programming sequence in the bitstream.The schematic of folded-cascode amplifier is shown in figure 6.6. The same transistor and passive component modules that are used in Miller topology are employed in the cascode amplifier.



(a) (b)

Figure 6.5: The evolvable chip in [LK05]. a) Reconfigurable Miller module schematic. b) Reconfigurable Miller module layout.

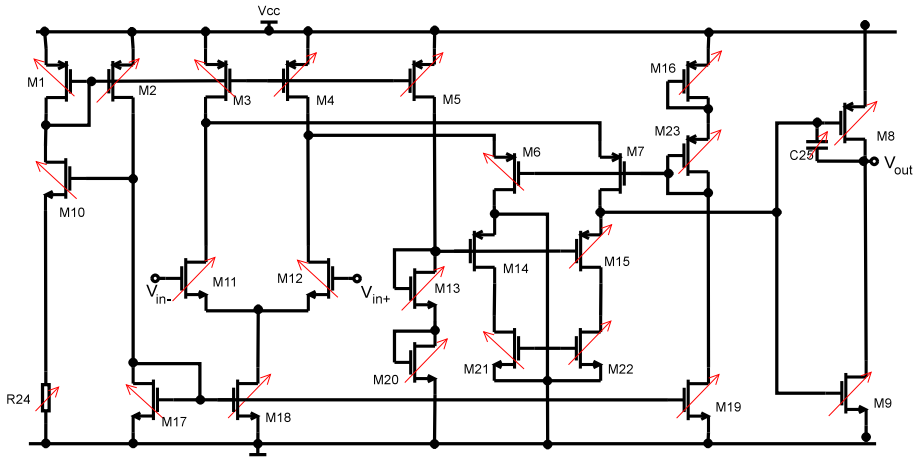


Figure 6.6: The schematic of the programmable folded-cascode operational amplifier module in [LK05].

Each transistor of the amplifier is replaced by an array of parallel CMOS transistors each of them has electronic switches connected in series with its terminals as shown in figure 5.15. The width of the programmable transistor module is programmed by connecting the transistors in parallel to each other through the switches. The implemented chip contains an array of eleven transistors with widths of 1, 1, 1, 1, 2, 4, 8, 16, 32, 64, and 128 μm [LK05]. Multiple instances of unit width are included

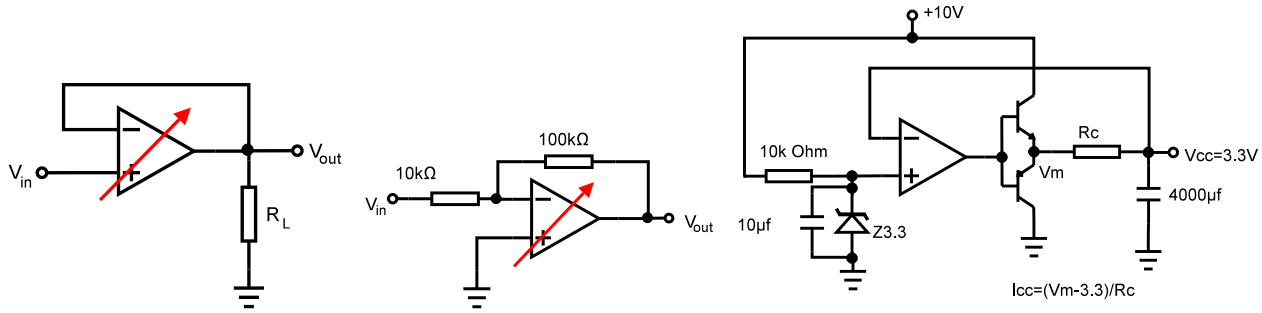
in order to improve the tolerance as they can particularly be affected by processing deviations. The lengths of the transistors are fixed to $1\mu m$ [LK05]. Each of the NMOS transistors is replaced with the programmable module in figure 5.15, each of the PMOS transistors is replaced with an equivalent module with PMOS transistors. Similarly, the capacitor module consists of a set of capacitors in parallel and a switch is connected to each of them in series. The capacitor programmable range is from 125 fF to 31.875 pF with a resolution of 125 fF *(8-bits)*. The programmable resistor module consists of a set of resistors in series, and switches in parallel to each of them. The resistor programmable range is from 125Ω to 31.875$k\Omega$ with a resolution of 125Ω. The hardware is implemented using Austriamicrosystems $0.35\mu m$ 3.3V CMOS technology. Obviously, the same technology is employed for the extrinsic evolution of the hardware. Therefore, the Cadance technology transistor models are exported to NGSPICE

The switches of the currently available chip has a narrow width, therefore, they can leave the resistive region and operate in the saturation region due to low current, which increases the voltage drop over the switch and limit the flowing current. This problem is treated in the FPMA2 [LK07] by modifying the transistor module to be programmed by controlling only the gate voltage of the transistors as shown in figure 5.16.

As the on-resistance of the current switch implementation is in the range of several kilo-ohms, the programmable resistance range, and the characteristics of the programmable capacitor are affected by this on-resistance, but are omitted in the experimental works as this problem is considered in FPMA2 [LK07] by employing wider switches.

Therefore, the behavior of the real chip is declined from the simulated behavior with ideal switches. However, the mentioned problems above are treated in FPMA2 which is currently in the manufacturing phase.

For the intrinsic evolution of the operational amplifier, only two circuits are needed; the first circuit with unity gain configuration with a load resistor of $10k\Omega$ to measure the input voltage offset, the slew rate, the settling time and the common mode range as shown in figure 6.7(a). The second circuit has

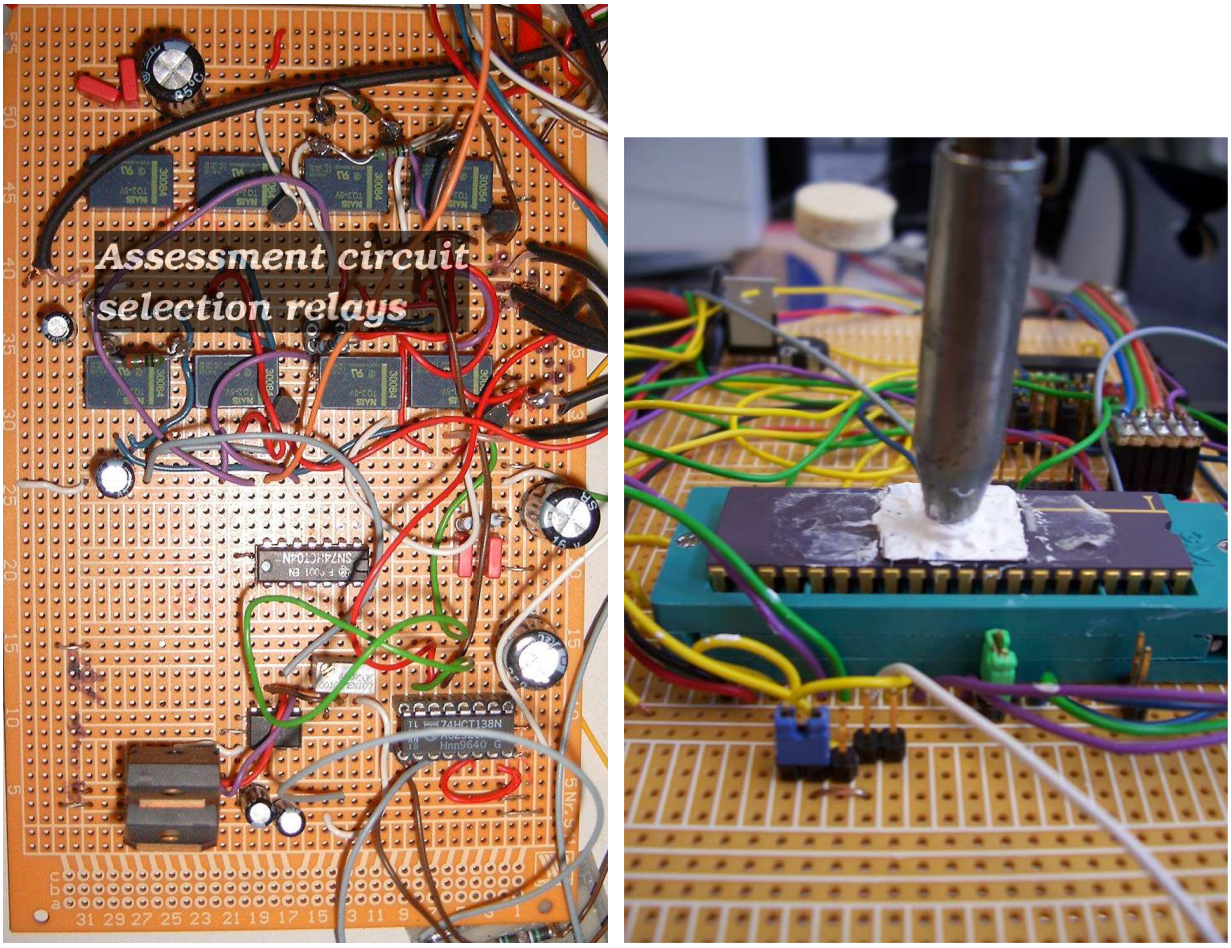


(a) (b) (c)

Figure 6.7: The intrinsic measurement circuits. a) Measuring the input voltage offset, slew rate, settling time, and CMR. b) Measuring the output swing voltage. c) Measuring the current by the voltage regulator circuit.

loop gain of -10 and is designed for measuring the output swing voltage as shown in figure 6.7(b). If the quiescent power consumption is required to be measured, an additional circuit is employed as the power supply regulator to measure the current as shown in figure 6.7(c), note that the operational amplifier in figure 6.7(c) is not the reconfiguration chip, while it is the reconfigurable hardware

(the device under test) in figures 6.7(a) and 6.7(b). A batch mode is designed that evaluates all the



(a) (b)

Figure 6.8: Photos of the experimental hardware setup. a) The assessment circuits. b) Heating the chip by soldering machine for the intrinsic dynamic environment experiment.

population for each of the specifications after selecting the required assessment circuit. Therefore, each assessment circuit is selected once for the complete generation, which increases the lifetime of the relays, and reduced the required delay time due to switching activities. In the non-stationary environment intrinsic experiment, the chip is heated by a soldering machine as shown in figure 6.8(b).

6.3 Building Block Level

6.3.1 Operational Amplifier

The operational amplifier is used in most of the sensor electronics applications. It is a voltage-mode building block for many circuits such as amplifiers, adders, oscillators, etc. Therefore, the operational amplifier is selected as a building block case study.

Each component of the operational amplifier is represented by a dimension in the search space. The

complete Miller topology programmable operational amplifier has 13 components [LK05], which means that the search space is R13 space. No symmetry constraints are made as they may obstruct recovering from static and dynamic deviations. Although omitting these constraints is not necessary in the extrinsic evolution, while it is essential in intrinsic and multi-objective mixtrinsic evolution, it is omitted also in the extrinsic evolution as the aim of the extrinsic evolution is to model the behavior of the intrinsically evolved hardware. Therefore, modeled deviations is applied as well in the extrinsic evolution. The search space range of the each of the dimensions is constrained to the equivalent component programmable range. The specifications that are included in the optimization are the common mode range (CMR), rising and falling slew rate ($SR\uparrow$,$SR\downarrow$), rising and falling settling time ($T_s\uparrow$,$T_s\downarrow$), input offset voltage, swing output voltage, open loop gain (A_0), common mode rejection ratio (CMRR), power supply rejection ratio (PSRR), output resistance (R_o), bandwidth (BW), phase margin (ϕ), and quiescent power consumption (P_c). The thermal noise of the amplifier is not optimized in the experimental work as the noise that the current chip generates is higher than the simulated thermal noise due to the on-resistance of the switches that are connected to the drain and the source of each of the programmable transistor module. This noise is minimized in FPMA2 [LK07] which is currently in manufacturing phase. However, the developed tool support optimizing the noise as well.

The aim of the experimental work of the operational amplifier is to demonstrate the optimization by various optimizers, to investigate performance of the dynamic environment suitable approaches, to evolve the standard hardware specifications extrinsically, intrinsically and multi-objective mixtrinsically, and to obtain the building blocks to the hierarchical optimization of the functional level.

Extrinsic Multi-Objective Evolution

In this experiment, the industrial specifications of the operational amplifier are optimized by several evolutionary computation optimizers. The extrinsic multi-objective evolution is investigated by three different required specifications and weights. The required specifications and their weights for the first, second, and third experiments are shown in tables 6.1, 6.3, and 6.5 respectively. The notation $spec_o$ refers to the required specification value of the objective o, f_o is an example of the objective values of the hardware that the optimizer returns.

The optimization of the operational amplifier is investigated extrinsically by HPSO *(hierarchical particle swarm optimization)*, PSO-TVAC *(particle swarm optimization with time-varying acceleration coefficients)*, MSPSO-TVAC *(multi-swarm particle swarm optimization with time-varying acceleration coefficients*), HPSO-TVAC *(hierarchical particle swarm optimization with time-varying acceleration coefficients)* and GA *(genetic algorithm)*. The original PSO with the commonly used parameter values was found not to perform well in comparison to GA with the selected parameter values mentioned below, therefore, it is not included in the experimental work. The results of this experiment have been partially published in advance in [TK05a, TK05b, TK06c, TK06a]. Evolving the operational amplifier with few of the standard specifications using GA is published in [TK05a]. the complete hardware specifications presented in this experiment, and comparing it with PSO is published in [TK05b], which was good enough to be invited for a book chapter in [TK06c]. Optimizing an operational amplifier as a comparator using PSO, MSPSO, and HPSO is published in [TK06a]. However, more PSO methods are presented here, and few improvements and debugging is done in the

[1]This value is selected under the consideration that this amplifier is followed by an ADC with a conversion speed of 10kSPS.

Table 6.1: The first variant of the target and the achieved specifications for extrinsic evolution of operational amplifier in static environment.

spec. name		k_o	$spec_o$	f_o
CMR	[V]	10	≥ 2.5	2.53
$SR\uparrow$	[V/µsec]	100	≥ 2	26.34
$SR\downarrow$	[V/µsec]	100	≥ 2	22.16
$T_s\uparrow$	[µsec]	100	≤ 100[1]	0.0964
$T_s\downarrow$	[µsec]	100	≤ 100[1]	0.3
$Offset$	[mV]	1	≤ 1	0.455
$Swing$	[V]	1	≥ 2.5	2.86
A_0	[dB]	15	≥ 74	83.026
$CMRR$	[dB]	0.1	≥ 60	80.208
$PSRR$	[dB]	0.03	≥ 60	67.503
R_0	[kΩ]	1	≤ 20	19.819
BW	[MHz]	1	≥ 10	26.233
ϕ	[°]	10	≥ 60	63.78
P_c	[mW]	Not optimized		22.1

Table 6.2: Transistor widths and passive component values of a returned configuration to the specification variant at table 6.1.

Component		Returned value
M1	[μm]	1
M2	[μm]	255
M3	[μm]	255
M4	[μm]	255
M5	[μm]	255
M6	[μm]	255
M7	[μm]	237
M8	[μm]	255
M9	[μm]	255
M10	[μm]	1
M11	[μm]	27
R12	[Ω]	125
C13	[pf]	5.75

post-processing to achieve more accurate assessment. In addition, improvement in the GA setting is demonstrated as well.

The commonly used parameter values are employed in the PSO variants; the inertia weight w starts by 0.9 and decreases linearly to 0.4, $C_1 = 2.05$ and $C_2 = 2.05$ for constant acceleration coefficients PSOs, while C_1 starts with 2.5 and ends with 0.5, and C_2 starts with 0.5 and ends with 2.5 for time-varying acceleration coefficients approaches. As described in chapter 5, the inertia weight is set to zero for the HPSO-TVAC, if the velocity becomes zero in any of the dimensions, it is reinitialized. The population in the PSO variants contains 20 particles. The multi-swarm PSO consists of 4 sub-swarms, each of them contains 2 charged particles, and 3 neutral particles. The experiment consists of

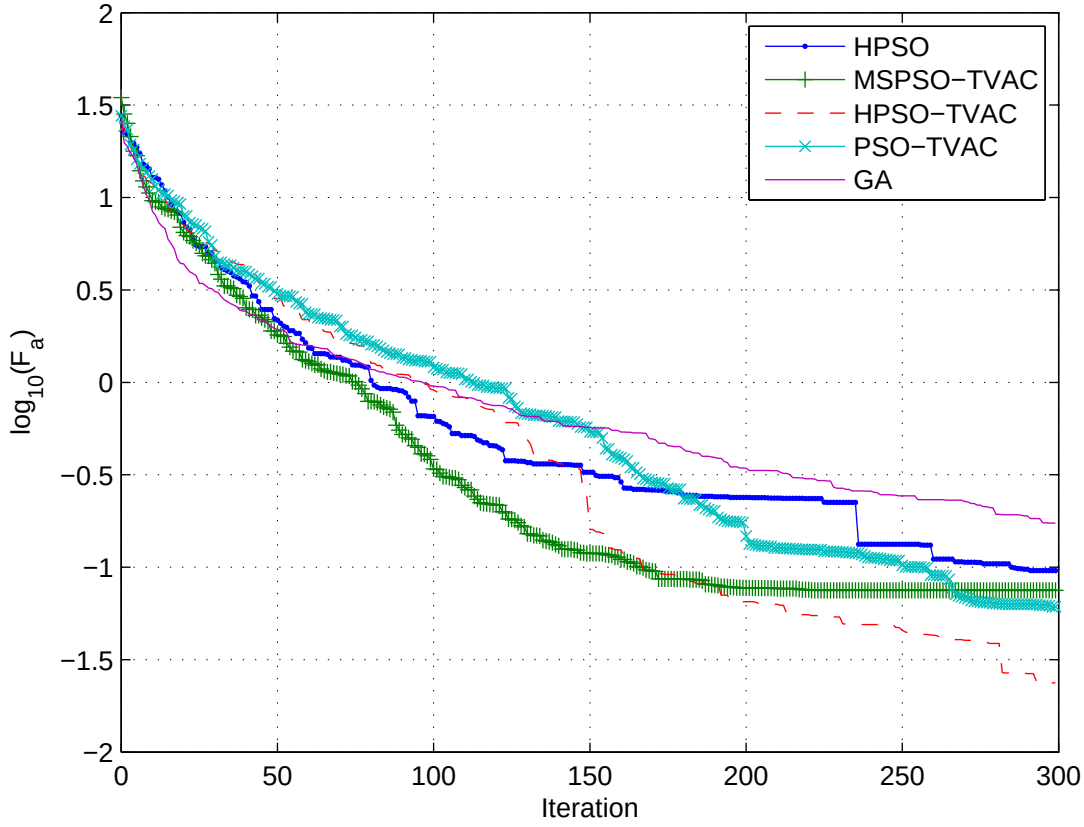


Figure 6.9: The convergence curve of the extrinsic evolution of the operational amplifier in [LK05] for the requirements in table 6.1.

Table 6.3: The second variant of the target and the achieved specifications for extrinsic evolution of operational amplifier in static environment.

spec. name		k_o	$spec_o$	f_o
CMR	[V]	10	≥ 2	2.215
$SR\uparrow$	[V/µsec]	100	≥ 2	174.02
$SR\downarrow$	[V/µsec]	100	≥ 2	33.85
$T_s\uparrow$	[µsec]	100	≤ 1	0.2822
$T_s\downarrow$	[µsec]	100	≤ 1	0.3882
$Offset$	[mV]	1	≤ 1	0.86
$Swing$	[V]	1	≥ 2	3.08826
A_0	[dB]	15	≥ 74	87.726
$CMRR$	[dB]	0.1	≥ 60	79.818
$PSRR$	[dB]	0.03	≥ 60	80.896
R_0	[kΩ]	1	≤ 40	38.71
BW	[MHz]	1	≥ 10	19.22
ϕ	[°]	10	≥ 60	60.616
P_c	[mW]	0.01	≤ 5	2.07

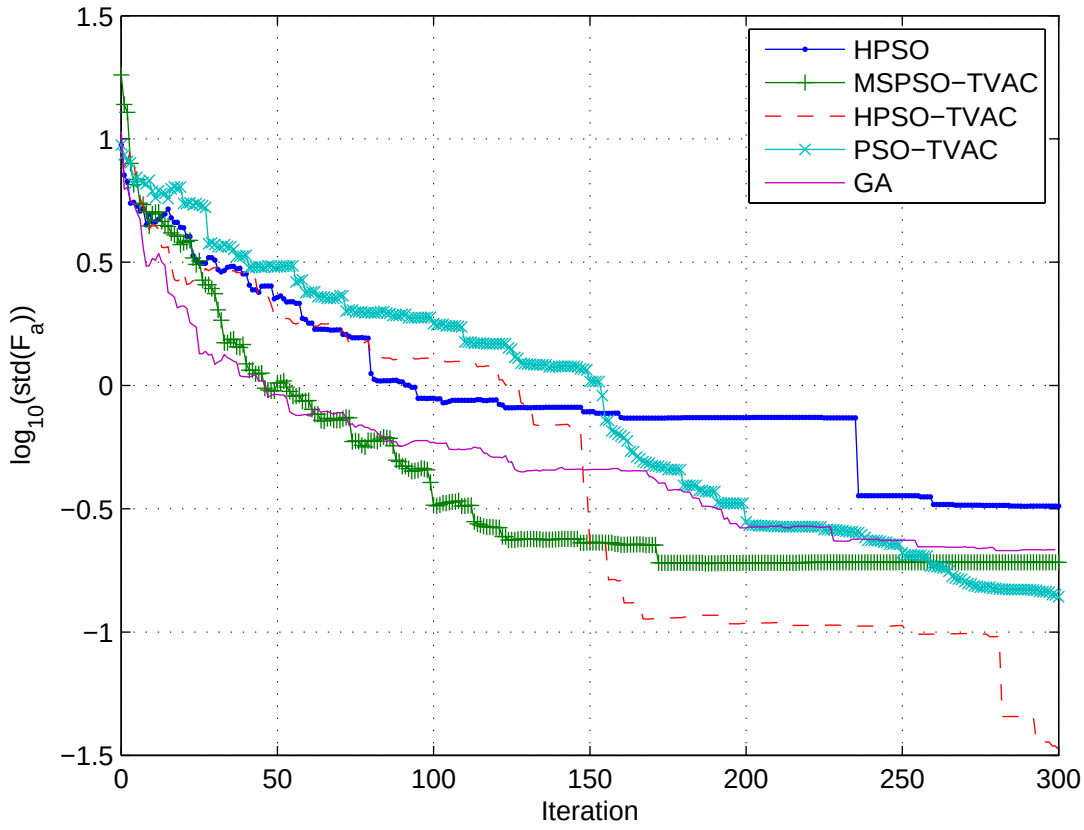


Figure 6.10: The standard deviation curve of the extrinsic evolution of the operational amplifier in [LK05] for the requirements in table 6.1.

Table 6.4: Transistor widths and passive component values of a returned configuration to the specification variant at table 6.3.

Component	Returned value
M1 [μm]	255
M2 [μm]	149
M3 [μm]	255
M4 [μm]	255
M5 [μm]	255
M6 [μm]	61
M7 [μm]	247
M8 [μm]	255
M9 [μm]	1
M10 [μm]	1
M11 [μm]	234
R12 [$k\Omega$]	31.875
C13 [pf]	0.25

10 runs by each optimizer for each of the three required specification variants. The maximum allowed

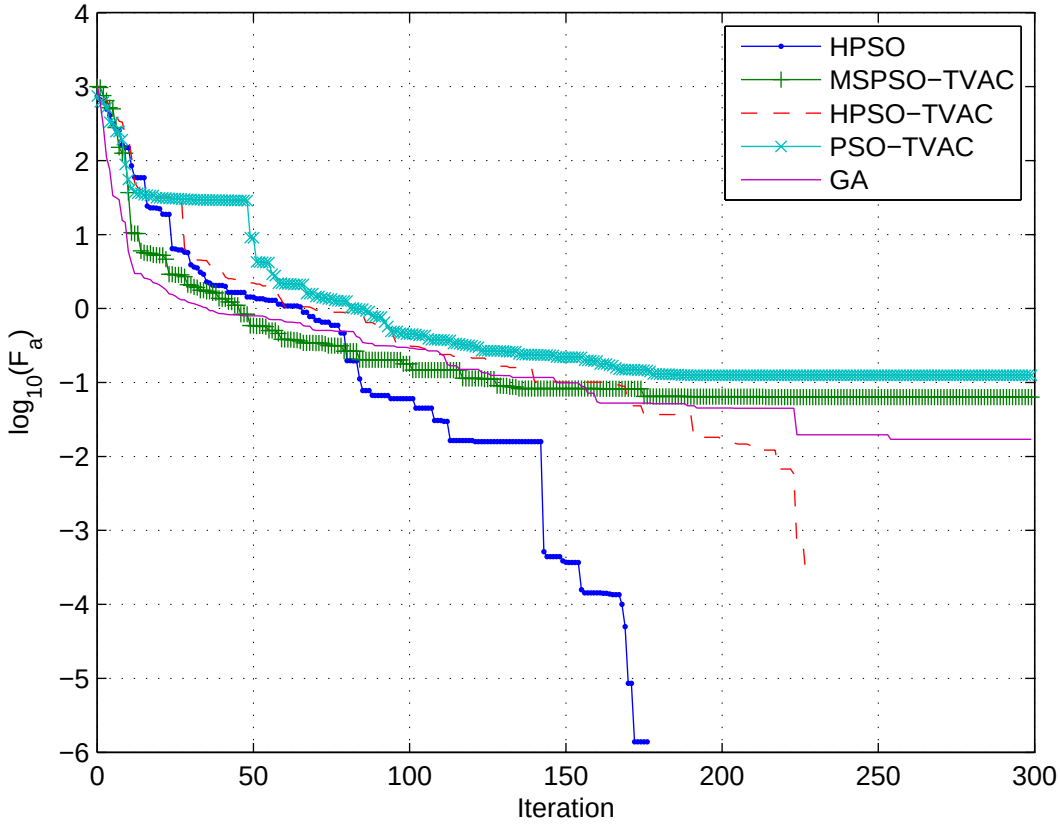


Figure 6.11: The convergence curve of the extrinsic evolution of the operational amplifier in [LK05] for the requirements in table 6.3.

Table 6.5: The third variant of the target and the achieved specifications for extrinsic evolution of operational amplifier in static environment.

spec. name		k_o	$spec_o$	f_o
CMR	[V]	10	≥ 2.5	2.516
$SR\uparrow$	[V/µsec]	1	≥ 10	17.84
$SR\downarrow$	[V/µsec]	1	≥ 10	14.815
$T_s\uparrow$	[µsec]	1	≤ 1	0.1584
$T_s\downarrow$	[µsec]	1	≤ 1	0.3667
$Offset$	[mV]	1	≤ 1	0.26
$Swing$	[V]	1	≥ 2.5	2.82
A_0	[dB]	5	≥ 74	83.3
$CMRR$	[dB]	0.1	≥ 60	81.1
$PSRR$	[dB]	1	≥ 60	71.14
R_0	[kΩ]	1	≤ 20	20.21
BW	[MHz]	1	≥ 10	15.4
ϕ	[°]	10	≥ 60	63.354
P_c	[mW]	1	≤ 5	4.901

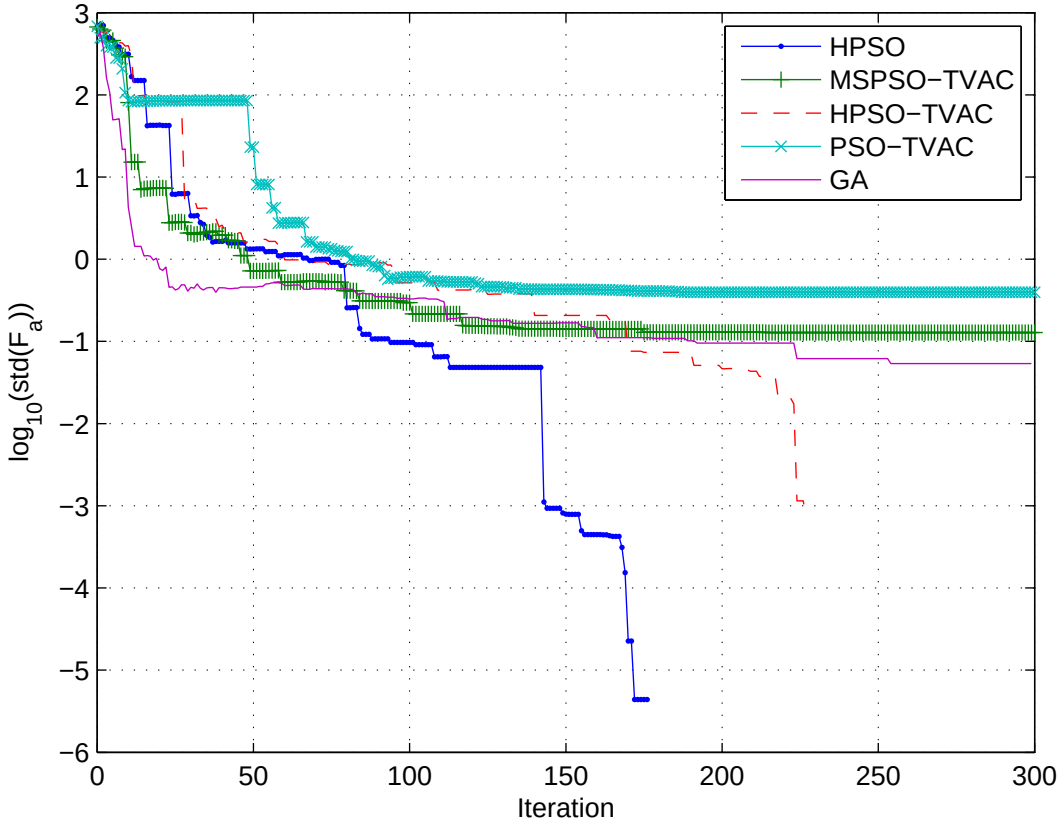


Figure 6.12: The standard deviation curve of the extrinsic evolution of the operational amplifier in [LK05] for the requirements in table 6.3.

Table 6.6: Transistor widths and passive component values of a returned configuration to the specification variant at table 6.5.

Component	Returned value
M1 [μm]	184
M2 [μm]	240
M3 [μm]	253
M4 [μm]	255
M5 [μm]	255
M6 [μm]	255
M7 [μm]	237
M8 [μm]	255
M9 [μm]	2
M10 [μm]	3
M11 [μm]	253
R12 [$k\Omega$]	23.875
C13 [pf]	4.875

velocity of each particle is the 254, and 127 for the first specification variant.

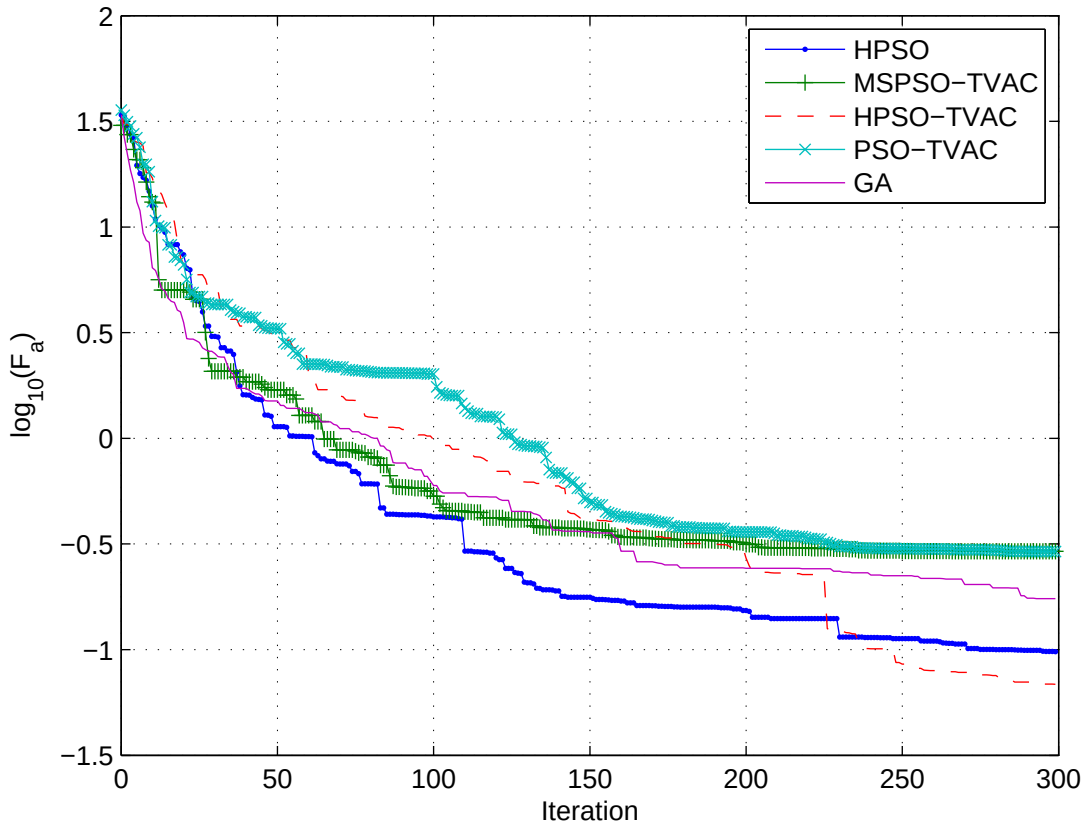


Figure 6.13: The convergence curve of the extrinsic evolution of the operational amplifier in [LK05] for the requirements in table 6.5.

As the PSO employs a memory element for each particle, more individuals are employed in GA. The GA population consists of 30 individuals, the best 10 individuals are kept to the next generation, and the rest is replaced each generation. The mutation probability in the GA is set to 0.1, and the crossover probability is set to 0.3. The parents are selected by the tournament selection, and the arithmetic crossover is employed for combining the parents. These GA parameter values are determined by extensive effort. The GA with this parameter values delivered better results than the GA in the previous publications [TK05a, TK05b, TK06c]. Its detailed sensitivity is not investigate here.

The convergence curves of the PSO variants and the GA of the settings in tables 6.1, 6.3 and 6.5 are shown in figures 6.9, 6.11 and 6.13 respectively. Their standard deviation curve is shown in figures 6.10, 6.12 and 6.14 correspondingly. The required time for a complete run (300 iterations) is about 2 hours on an AMD Athlon64 2.4GHz processor.

An example of the returned configuration for the first, second, and third specification variants are shown tables 6.2, 6.4, and 6.6.

The AC characteristics curve and the step response of the configuration in table 6.6 is shown in figures 6.15 and 6.16 respectively as an example of the returned results.

The last iteration mean values of all the runs using the GA, HPSO, HPSO-TVAC, MSPSO-TVAC,

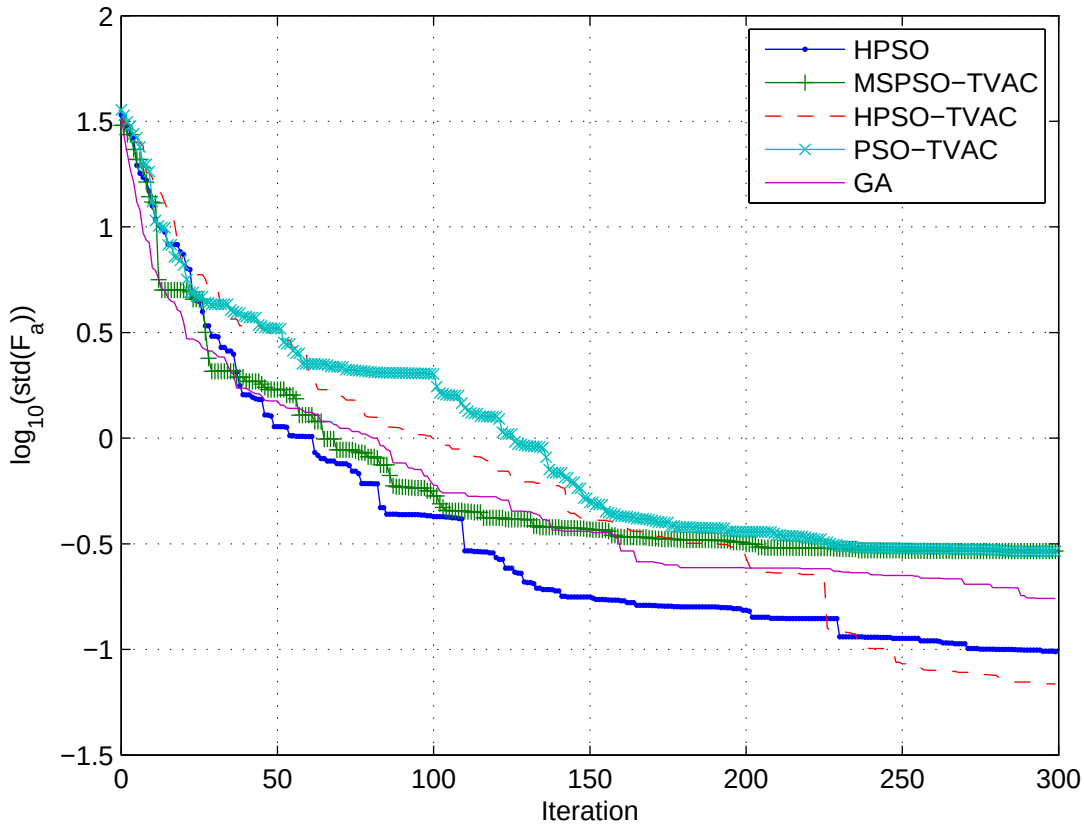


Figure 6.14: The standard deviation of the extrinsic evolution of the operational amplifier in [LK05] for the requirements in table 6.5.

Table 6.7: The last iteration mean value of all the runs using the GA, HPSO, HPSO-TVAC, MSPSO-TVAC, and PSO-TVAC.

	Target setting in table 6.1	Target setting in table 6.3	Target setting in table 6.5
GA	0.1727	0.017	0.174
HPSO	0.0959	0	0.1707
HPSO-TVAC	2.363×10^{-2}	0	0.1802
MSPSO-TVAC	7.485×10^{-2}	4.857×10^{-2}	0.1886
PSO-TVAC	6.11^{-2}	0.2502	0.2316

and PSO-TVAC are shown in table 6.7, there standard deviations are given in table 6.8 The HPSO performed better than the GA in the three experiments. However, the standard deviation of the GA is better than the HPSO in two of the three experiments.

The behavior of the algorithm is evaluated in dynamic environment by modeling the effect of varying the temperature of the transistor M8 to $100°C$ gradually over 300 iteration. The temperature fluctuation is applied every 20 iterations. This means that each 20 iteration, the temperature of M8 is increases by $4.8667°C$. The aim of this experiment is to investigate if starting from scratch is necessary after any environmental change, which is done in the state-of-the-art evolvable hardware. The

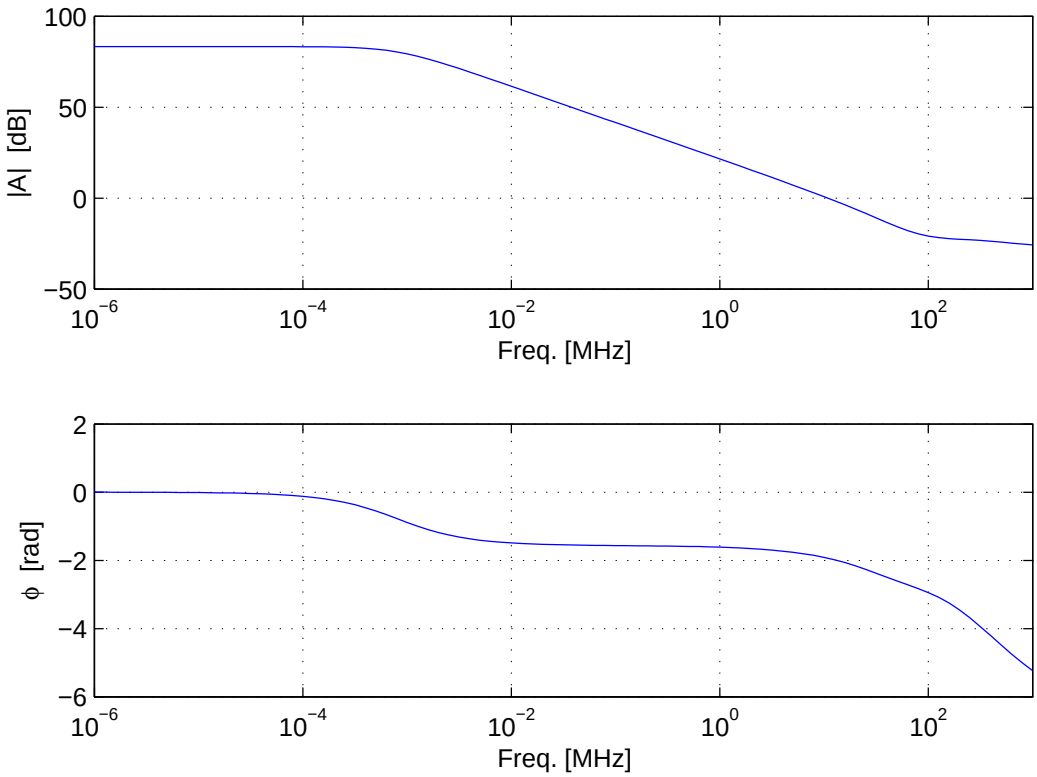


Figure 6.15: AC characteristics curve of the amplifier with the configuration in table 6.6.

Table 6.8: The last iteration standard deviation using the GA, HPSO, HPSO-TVAC, MSPSO-TVAC, and PSO-TVAC.

	Target setting in table 6.1	Target setting in table 6.3	Target setting in table 6.5
GA	0.2155	0.05376	0.1223
HPSO	0.3242	0	0.1456
HPSO-TVAC	0.0345	0	0.0733
MSPSO-TVAC	0.1915	0.1279	0.2754
PSO-TVAC	0.1417	0.3955	0.2634

required weights and the values of the operational amplifier specifications are given in table 6.9. In results published in advance mentioned above [TK05a, TK05b, TK06c, TK06a], the evolution starts from scratch after the environmental change in order to investigate if the evolvable hardware can cope with deviations. On the contrary, the results presented here demonstrates an improvement to the state-of-the-art evolution electronics, where the evolution does not start from scratch after every environmental change.

The optimization is accomplished by the MSPSO (multi-swarm PSO), and HPSO for dynamic environment. The time variant acceleration coefficient approaches are not employed for the non-stationary environment experiment as the evolution is expected to be employed in the run time of the hardware, therefore, the number of iterations is not limited as the evolution calibrates the system every time it

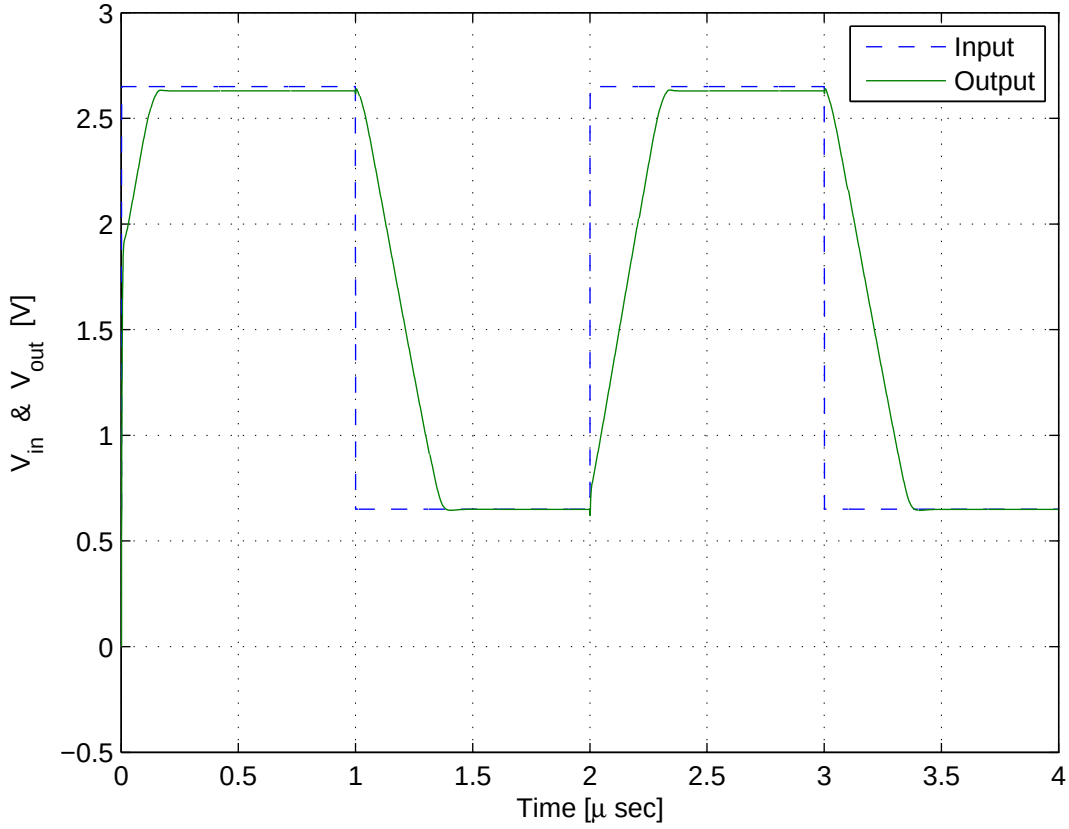


Figure 6.16: Step response of the amplifier with the configuration in table 6.6.

Table 6.9: The target specifications for operational amplifier in dynamic environment.

spec. name		k_o	$spec_o$
CMR	[V]	10	≥ 2
$SR\uparrow$	[V/µsec]	100	≥ 2
$SR\downarrow$	[V/µsec]	100	≥ 2
$T_s\uparrow$	[µsec]	100	≤ 1
$T_s\downarrow$	[µsec]	100	≤ 1
$Offset$	[mV]	1	≤ 1
$Swing$	[V]	1	≥ 2
A_0	[dB]	15	≥ 74
$CMRR$	[dB]	0.1	≥ 60
$PSRR$	[dB]	0.03	≥ 60
R_0	[kΩ]	1	≤ 40
BW	[MHz]	1	≥ 10
ϕ	[°]	10	≥ 60
P_c	[mW]	0.01	≤ 5

changes. The mean value of 10 runs of the convergence curve during the evolution is shown in figure 6.17. The corresponding standard deviation curve is shown in figure 6.18

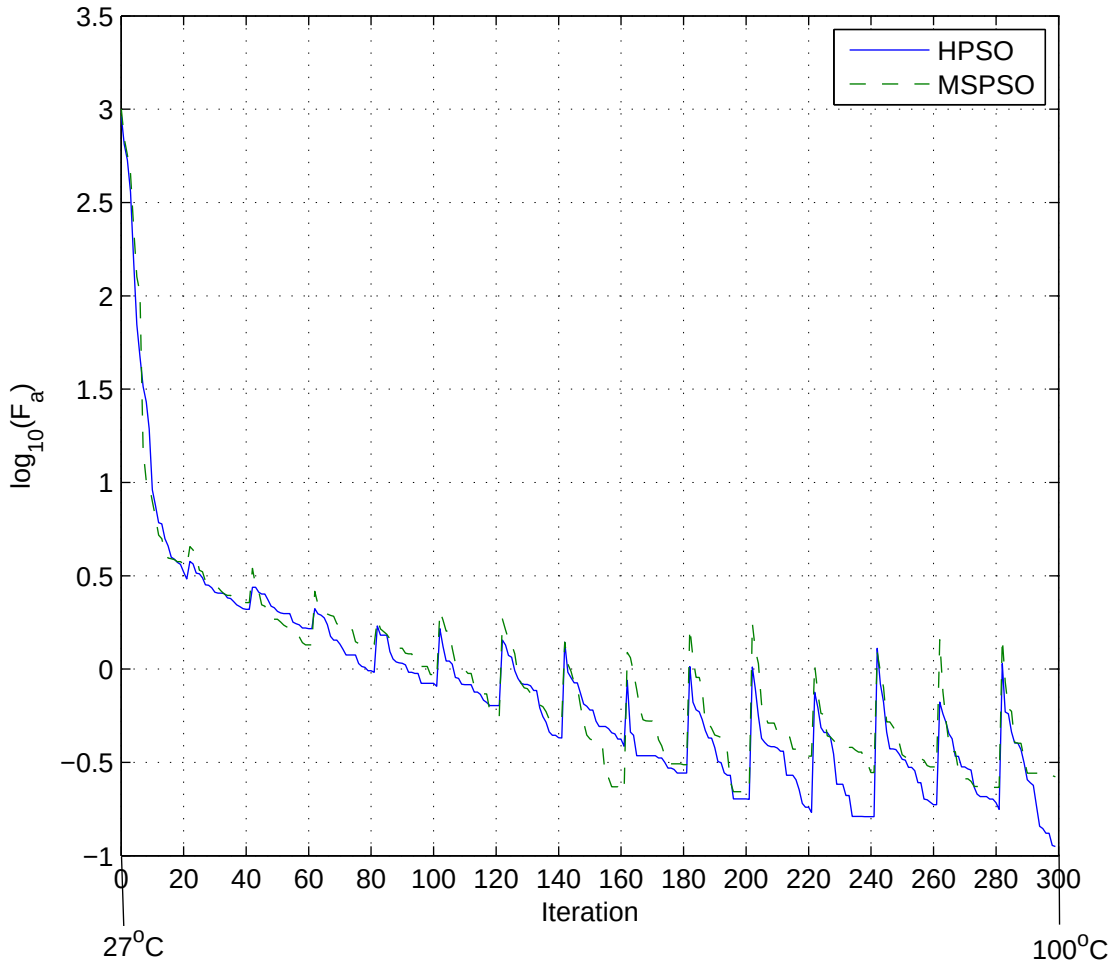


Figure 6.17: The convergence curve of the extrinsic evolution of the operational amplifier in dynamic environment.

In the figures 6.19 to 6.25, the convergence curve of each of the specifications for one of the runs is shown.

The environmental change is detected by reevaluating the global best achieved fitness each iteration. The best achieved fitness values of all the particles are reevaluated if any environmental change is detected [CD00]. As shown, employing the hardware previous experience reduced the required number of iterations to find a new suitable solution. In this experiment, the HPSO behaves better then MSPSO in dynamic environment. Depending on the objective functions, and the dynamic environment changing rate, etc., the MSPSO may converge faster in some cases as shown below in intrinsic multi-objective optimization in dynamic environment.

Intrinsic Multi-Objective Evolution

The extrinsic evolution of the hardware cannot cope with the system deviations as it does not has a complete model about the instant deviations of the hardware. Therefore, the intrinsic evolution

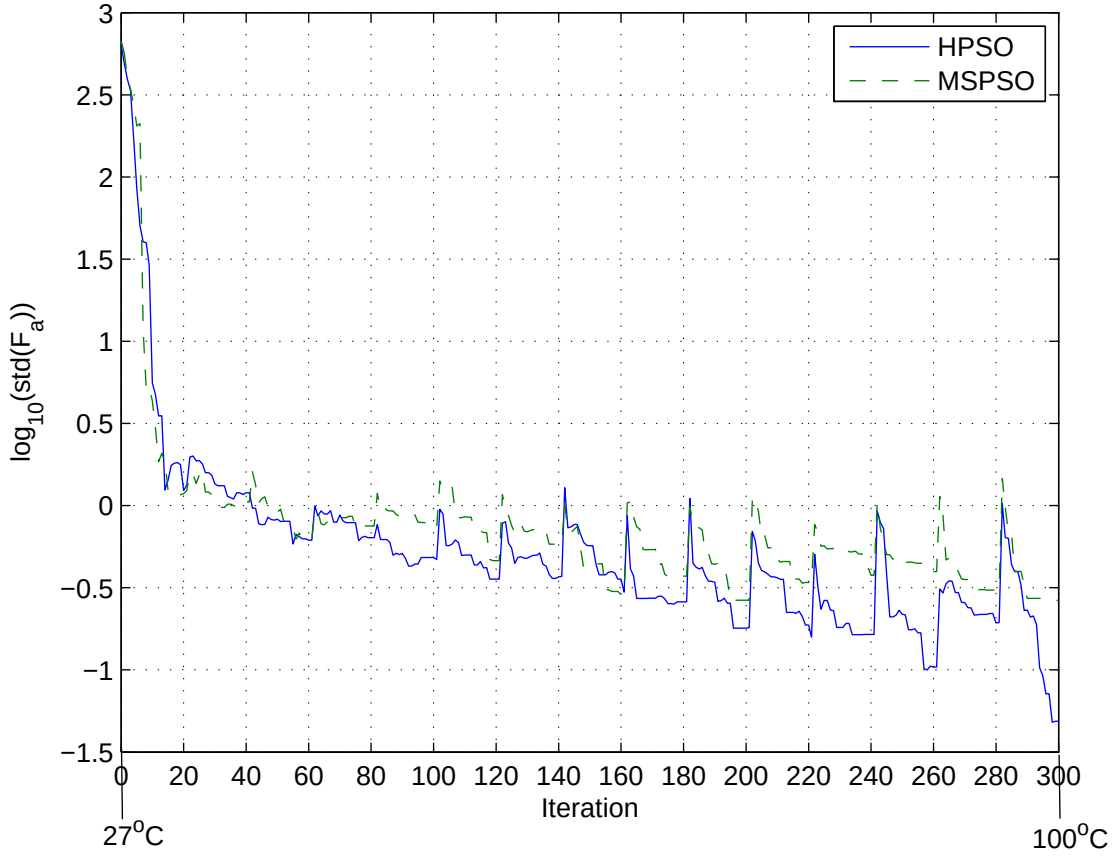
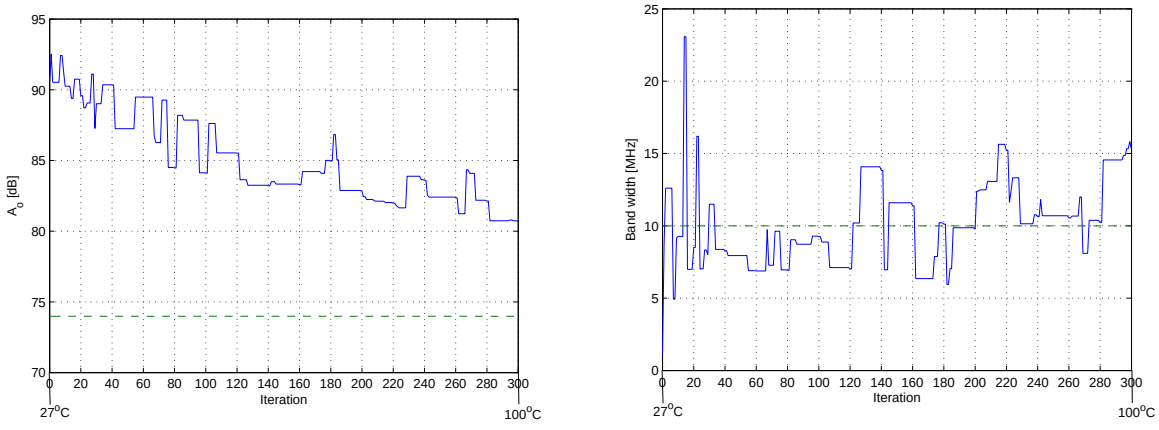


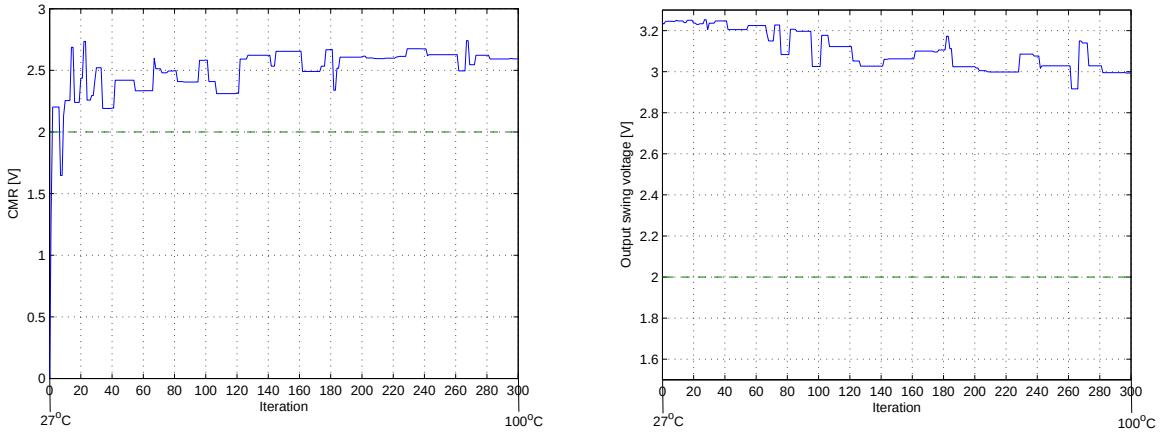
Figure 6.18: The standard deviation curve of the extrinsic evolution of the operational amplifier in dynamic environment.



(a) (b)

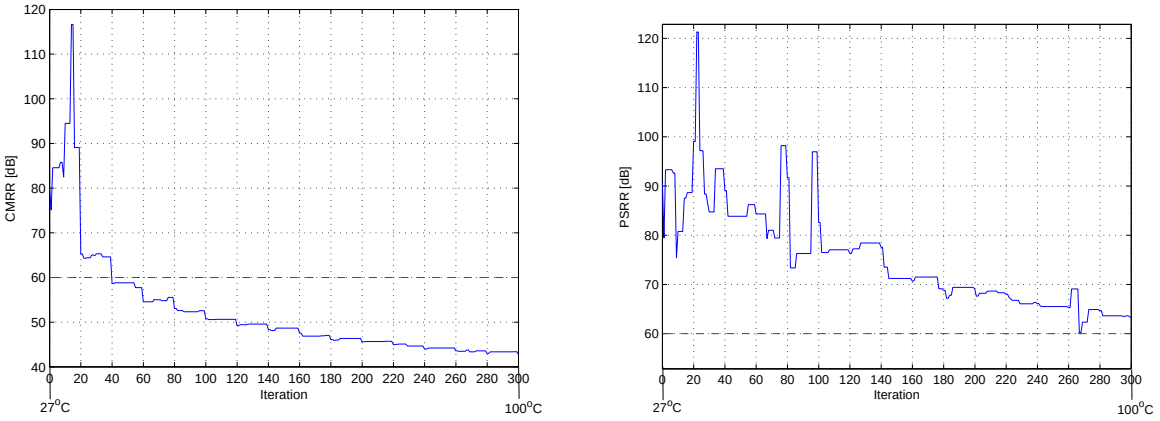
Figure 6.19: The achieved specifications of the evolution of operational amplifier in dynamic environment. a) Open-loop gain. b) Bandwidth.

optimizes the hardware by real measurements to recover from deviations. The aim of this experiment is to evolve the hardware by real measurements to recover from the instant system deviations.



(a) (b)

Figure 6.20: The achieved specifications of the evolution of operational amplifier in dynamic environment. a) CMR. b) Output swing voltage.



(a) (b)

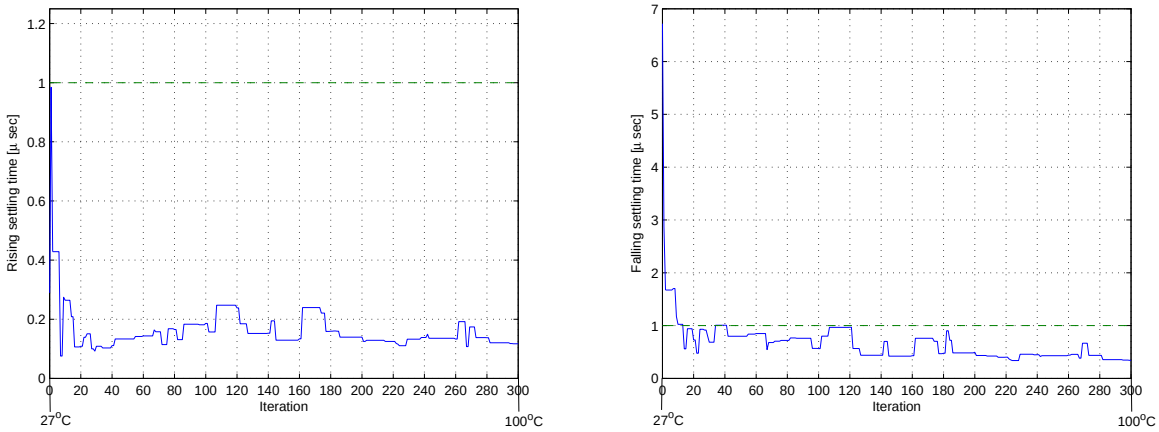
Figure 6.21: The achieved specifications of the evolution of operational amplifier in dynamic environment. a) CMRR. b) PSRR.

For the evaluation of each individual, the processing unit first downloads the configuration to the evolvable chip, chooses the required measurement circuits by switching the circuit selection relays, generates the required test signals, measures the output signals, then runs the necessarily post-processing to extract the hardware specifications from the output signals.

The specifications that are optimized intrinsically in the experimental work regarding the measurement setup cost are common mode range *(CMR)*, swing output voltage, input voltage offset, rising settling time, falling settling time, rising slew rate, and falling slew rate[2]. The open loop frequency response is implicitly optimized by optimizing the settling time and the slew rate[3]. The required specification for the settling time is 100 μsec, and its weight is 10, while it is 0.01 $V/\mu sec$ for the slew rate and its weight is 10^3. The required offset is $0.1mV$, and its weight is 0.1. For the CMR, the

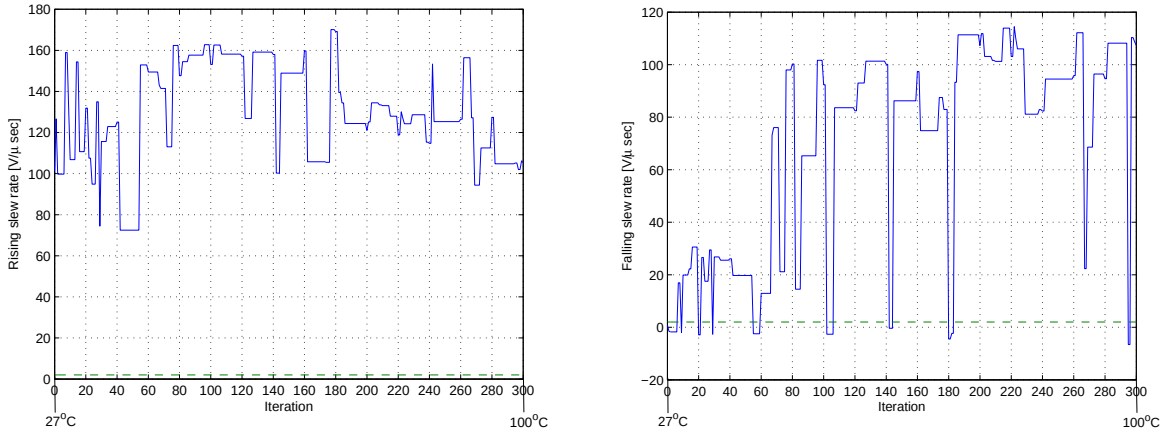
[2]The value of slew rate and settling time is limited to the employed ADC conversion speed.

[3]Therefore, their required values are limited to the conversion speed of the ADC as well



(a) (b)

Figure 6.22: The achieved specifications of the evolution of operational amplifier in dynamic environment. a) Rising settling time. b) Falling settling time.



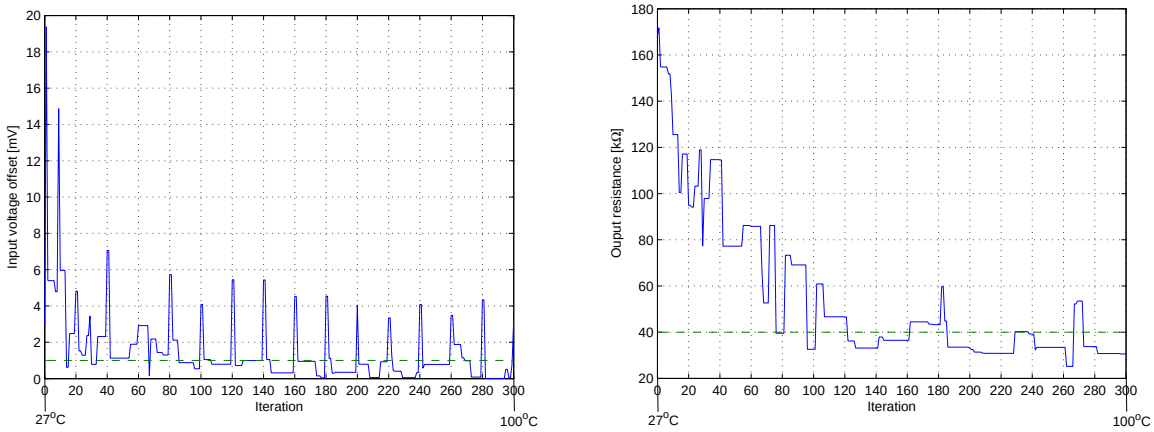
(a) (b)

Figure 6.23: The achieved specifications of the evolution of operational amplifier in dynamic environment. a) Rising slew rate. b) Falling slew rate.

required specification is $1.5V$, and its weight is 10, while it is 2.5 for the output swing voltage, and its weight is 1. The quiescent power consumption is optimized to $10\mu W$ for low power applications, its weight is 1.

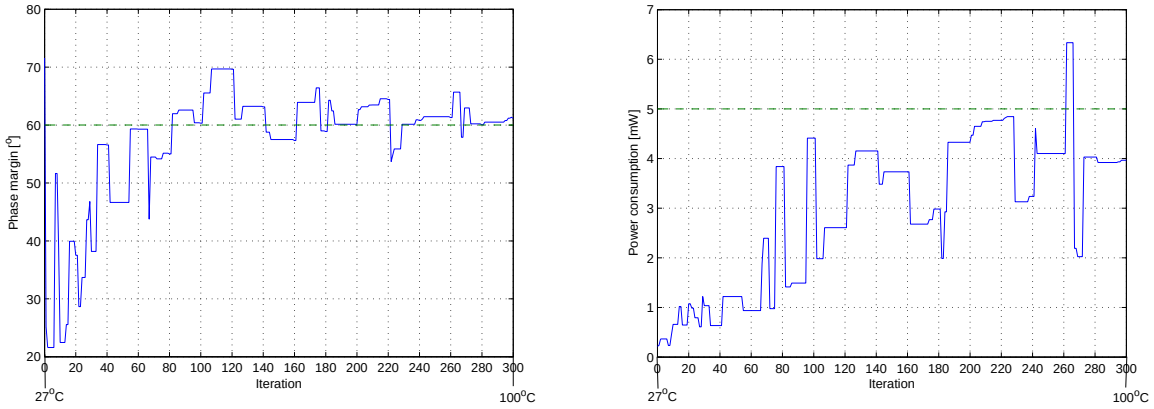
The convergence curve of the mean value of 10 runs of the optimization using HPSO, MSPSO-TVAC, HPSO-TVAC, and PSO-TVAC is shown in figure 6.26, and the standard deviation curve is shown in figure 6.27.

The population size is 20 particles, and it runs for 100 iterations. The maximum velocity for each particle is half the search space. The value of the inertia weight w starts with 0.9, and decreases linearly to 0.4 over the 100 iteration. For the HPSO, $C_1 = 2$ and $C_2 = 2$. For the time variant acceleration coefficient approaches, C_1 starts with 2.5 and decreases linearly to 0.5, while C_2 starts with 0.5 and increases linearly to 2.5. The MSPSO consists of 4 sub-swarms, each sub-swarm has 5 particles, in which, two of them are charged with a charge of 10.



(a) (b)

Figure 6.24: The achieved specifications of the evolution of operational amplifier in dynamic environment. a) Offset. b) Output resistance.



(a) (b)

Figure 6.25: The achieved specifications of the evolution of operational amplifier in dynamic environment. a) Phase margin. b) Quiescent power consumption.

The batch assessment mode that is described above is employed in which; an assessment circuit is selected and the equivalent objectives are evaluated in all the population, then the next assessment circuit is selected, and so on. This minimizes the need of waiting till the relays are stable, and increases the life-time of the relays. A photo of the assessment circuits of the chip is shown in figure 6.8(a). The required time for a complete run is about 8 minutes.

A snapshot for the input and the output signals of the circuit in figure 6.7(a) is shown in figure 6.28. After the configuration had been downloaded, the offset is measured by feeding a signal with virtual ground to the input, which is 1.65 V, as the chip is implemented by 3.3V CMOS technology as mentioned above. In order to reduce the noise, the mean value of 4000 samples is taken. Thereafter, the slew rate and the settling time are measured with the same signal which rises from 1.65V by 0.5V to 2.15V for 1000 samples, then drop again to 1.65V for 1000 samples in order to measure the rising slew rate and the rising settling time. The input voltage level for large signal such as measuring the slew rate should be more than 0.5V with respecting to virtual ground, but the same signal is employed

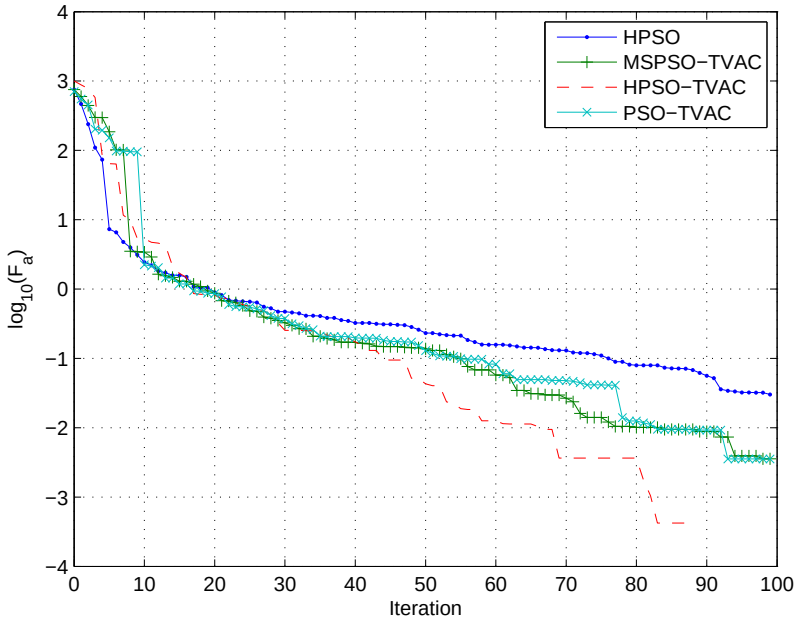


Figure 6.26: The convergence curve of the intrinsic evolution in the static environment.

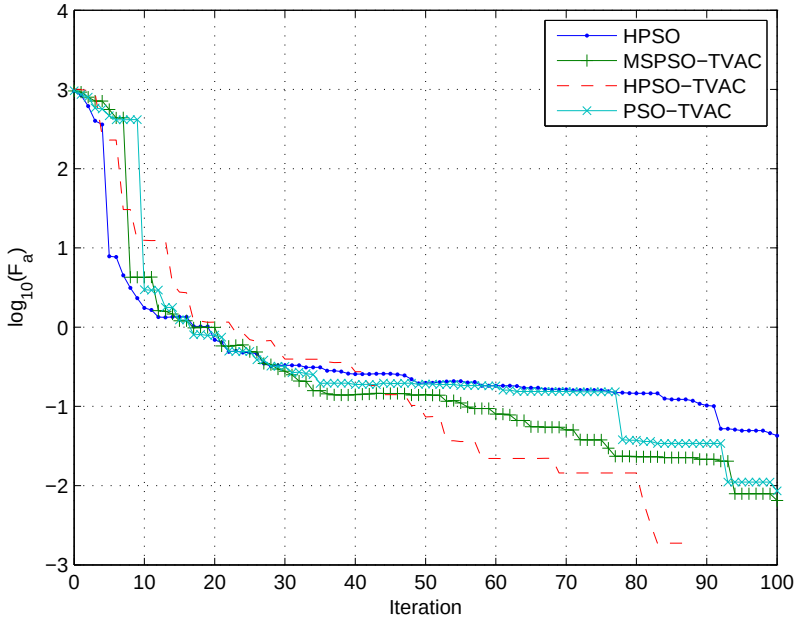


Figure 6.27: The standard deviation curve of the intrinsic evolution in the static environment.

for measuring the settling time and slew rate as an approximation to reduce the calibration time, this input level is selected to compromise on both the settling time and the slew rate. Afterwards, the input signal drops to 1.15V (which is -0.5 with respect to virtual ground) in order to measure the falling slew rate and the falling settling time. As the sampling time is 10µ*sec*, the fastest measurable settling time is 10µ*sec*, which means that the signal is stable in less than a sample. Similarly, the maximum

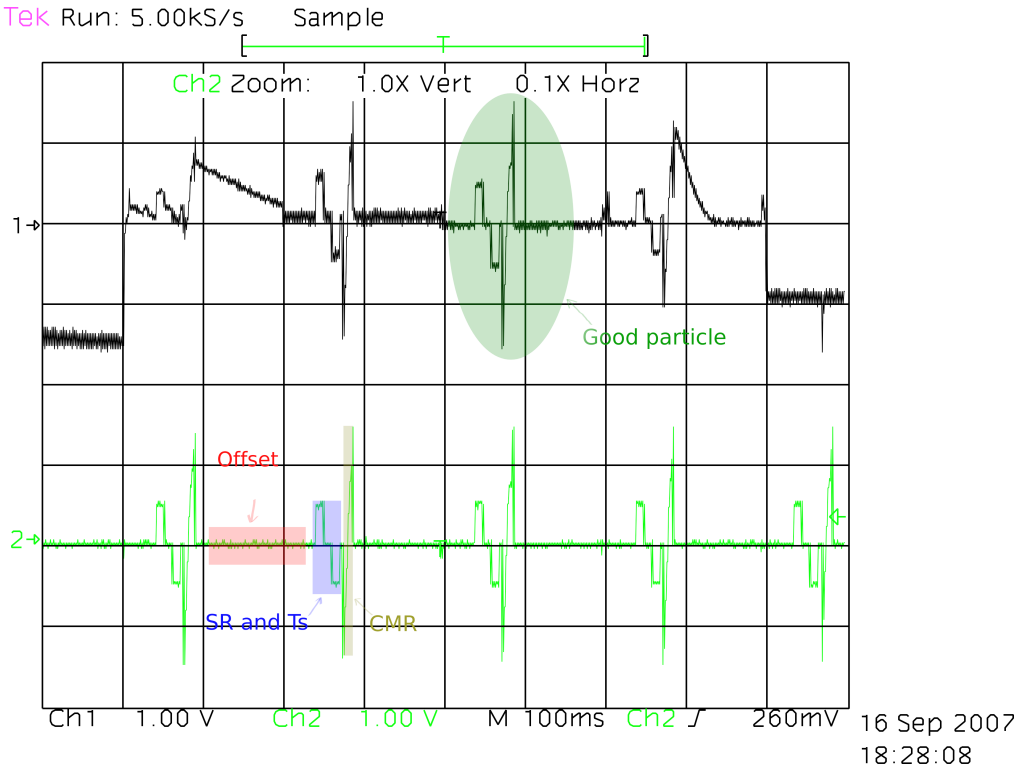


Figure 6.28: Measuring offset, CMR, T_s and SR intrinsically.

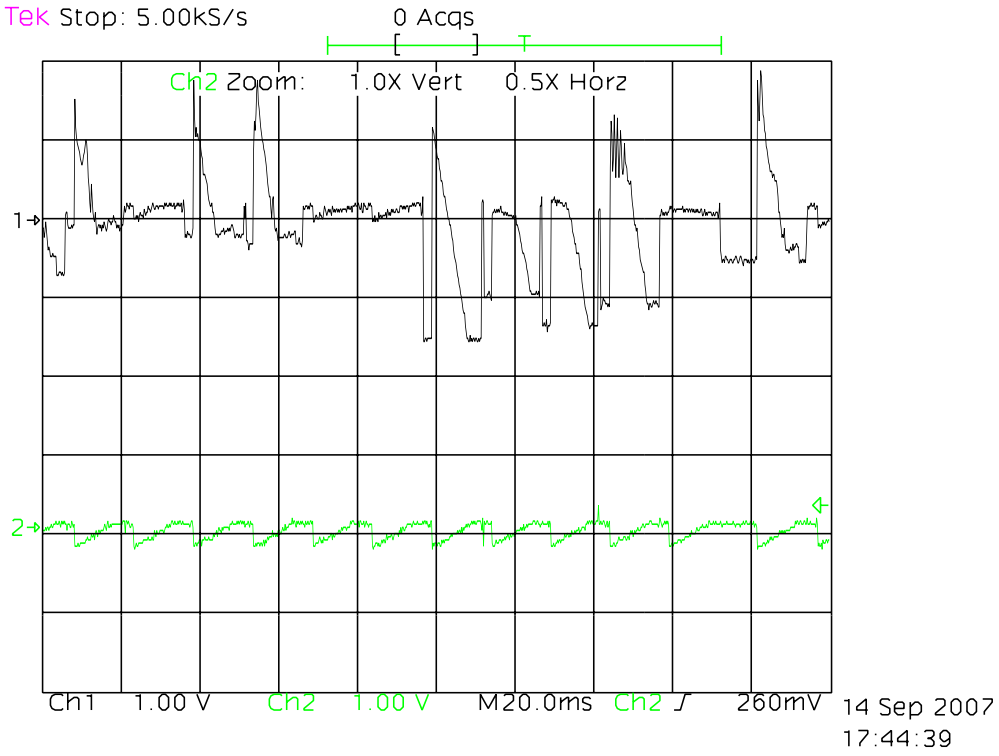


Figure 6.29: Measuring the output swing voltage intrinsically.

slew rate is $\frac{0.5V}{10\mu sec} = 0.05V/\mu sec$. The values shown in the table for both the settling time and the slew rate are because the signal reached the output in less than one sample. In order to reduce the

measurement noise, a moving average filter with width of 3 samples is used, and the specification values are obtained by the mean value of 5 measurements. For measuring the common mode range, an input signal is generated that starts by 0.5V, and ends by 3V in 100 steps. Each step consists of 20 samples. Then the linear range around the ground is returned. The output swing voltage is measured by the test signal shown in figure 6.29. The test signal is a linearly increasing signal around the virtual ground with the range between $1.5250 - offset$ and $1.7750 - offset$.

In figure 6.30, an example of the response signal of an individual is shown that is good in most of the objectives but with small CMR. In figure 6.31, an example of an individual with too a small

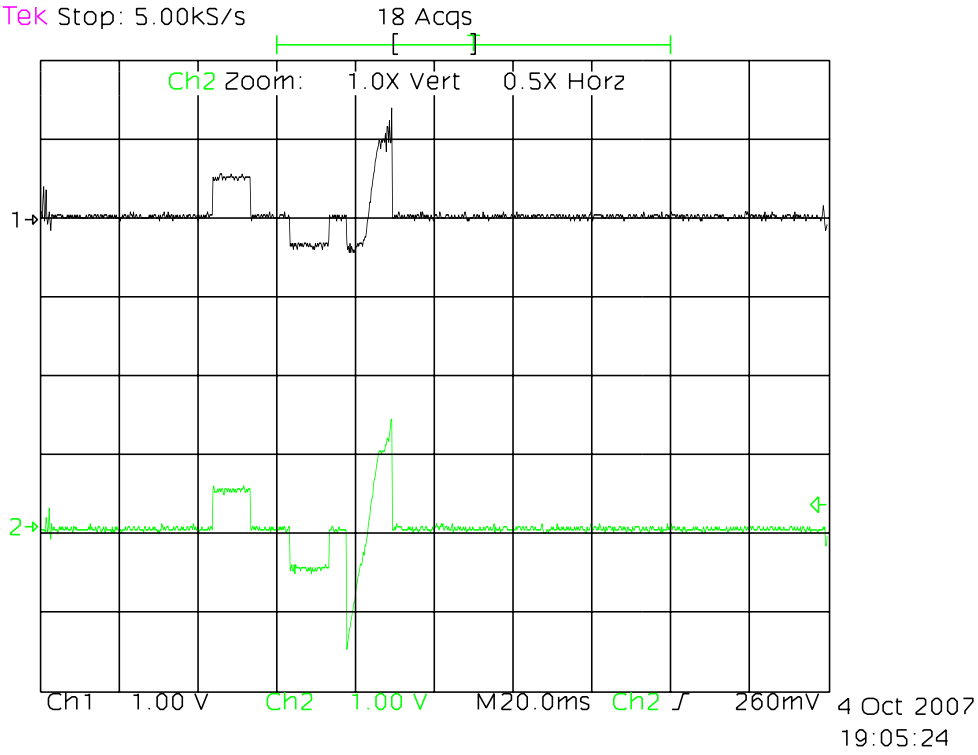


Figure 6.30: The effect of small CMR.

phase margin is shown. The output appears to be a thick line because of the oscillations due to the operational amplifier instability.

The intrinsic evolution is mapped to the embedded system in figure 6.4, the compiled programm size is 165.388 kBytes. The CPU load is about 1.7%, and the employed memory is 1.2 MBytes. The time required for one run is about 8 minutes. The HPSO-TVAC with the parameters given above is tested for checking the required time for one run.

The behavior of the hardware evolution in dynamic environment is investigated by heating the chip by a soldering machine as shown in figure 6.8(b). First the optimization algorithm is run to evolve the amplifier at the nominal temperature for 20 iterations, then the soldering machine is turned on. The temperature of the soldering machine increases gradually to 150°C. The setup applies irregular deviation to the chip as the air movement inside the laboratory can change the environmental settings as it cools down the chip, therefore, the same experimental results are not repeatable. However, aim of the experiment is to demonstrate coping with non-regulator dynamic deviations. This results are published earlier in [TLK06].

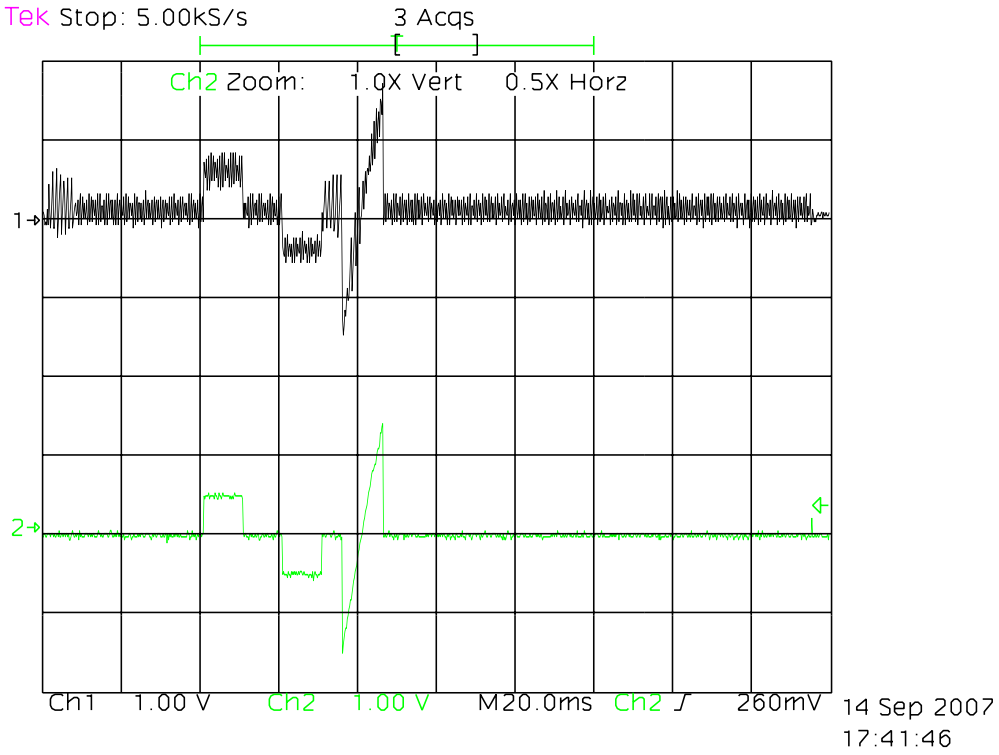


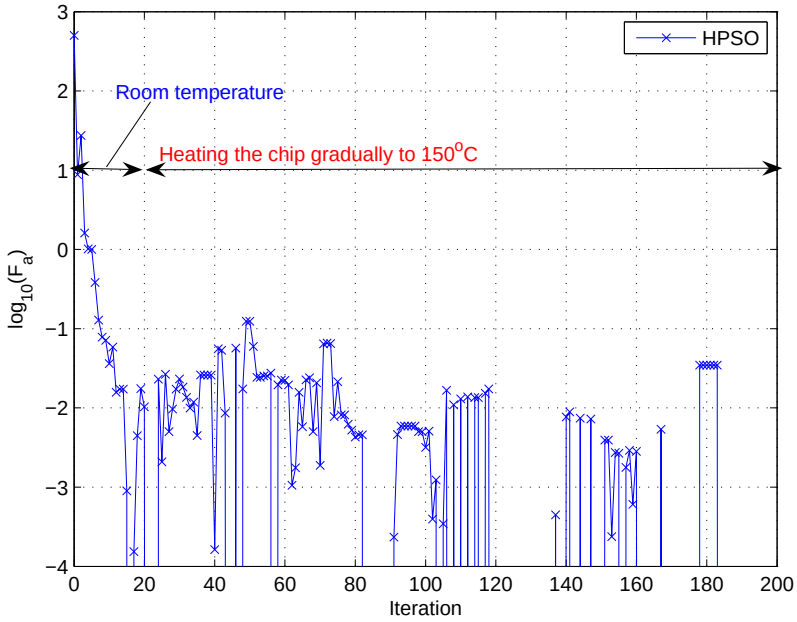
Figure 6.31: The effect of small phase margin.

The evolvable intrinsic evolution is investigated by MSPSO, and HPSO. If any environmental change is detected, the algorithm reevaluates the fitness of the position equivalent to the best achieved fitness value (the **p** vector) of all the particles. The target specifications are the same set of specifications that are optimized intrinsically above, except that the power consumption and the swing output voltage are not optimized in this experiment in order to give the freedom to the evolution to find working configurations in harsh environments even if it consumes more power than the above experiment, and can drive output signal with smaller peak value. For both HPSO and MSPSO, the population size is 20 particles, they run for 200 iteration. The maximum velocity for each particle is 32. The value of w starts by 1 and decreases linearly to reach 0.7 at the last iteration, and C_1 and C_2 are equal 2.05. Similar to the static environment experiment, the MSPSO consists of 4 swarms, each swarm has 5 particles, and two of them are charged with a charge of 10.

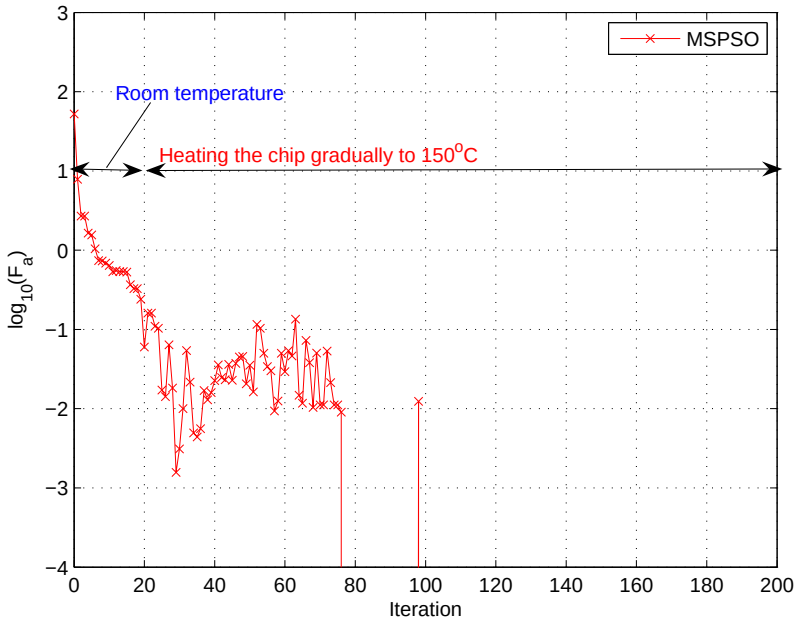
The global best is reevaluated each iteration to detect the environmental changes. In order to decrease false environmental change detection that can be done because of the noise, 10% of environmental change is accepted without taking any action. This means, any small deviation when the fitness value is near zero is treated as an environmental change.

In figure 6.32, the convergence curves of the mean value of 5 runs for both MSPSO and HPSO are shown, the correspond standard deviation curves are shown in figure 6.33. The discontinuity in the curve is because $log(0)$ is undefined. In some cases, due to noise, the measured specifications of the best particle are changed in the first 20 iterations before applying the heating. The static environment intrinsic evolution above does not converge as fast as here because the power consumption is optimized in the static environment experiment.

Employing dynamic environment suitable approaches enables the evolution to cope with dynamic



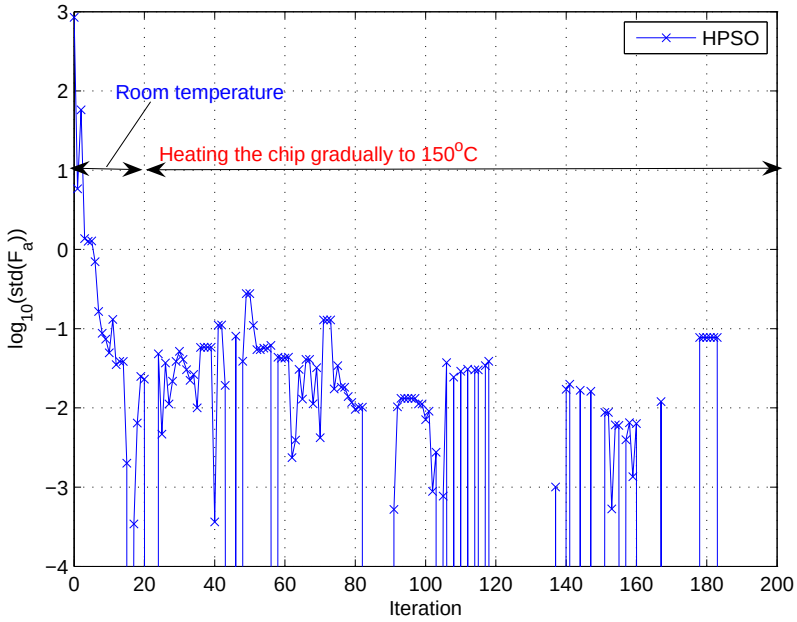
(a)



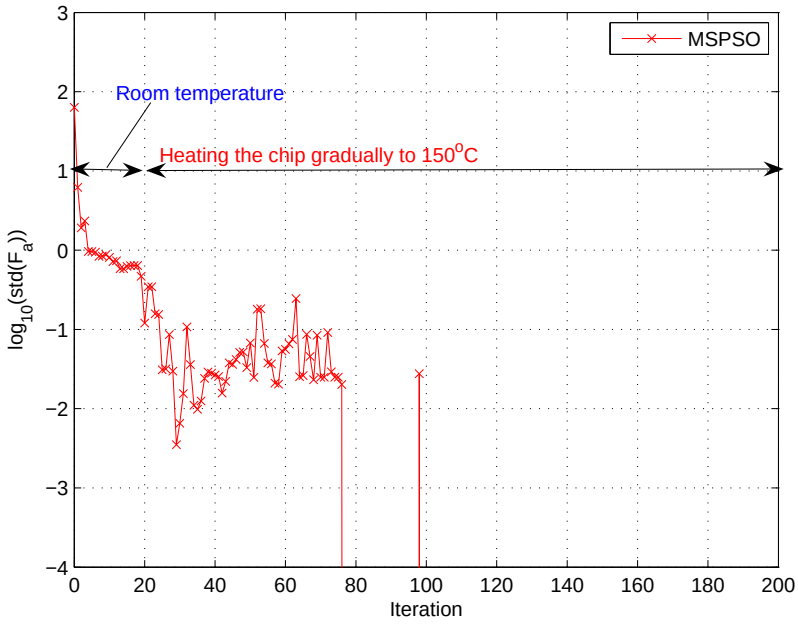
(b)

Figure 6.32: The convergence curves of intrinsic evolution in the stationary environment. a) Using HPSO, b) Using MSPSO.

environment without starting from scratch after every environmental change. The MSPSO coped with the environmental changes in this experiment better than the HPSO. However, it depends on the required specifications, weights, the environmental changes, and the noise as well.



(a)



(b)

Figure 6.33: The standard deviation curves of intrinsic evolution in the stationary environment. a) Using HPSO, b) Using MSPSO.

Mixtrinsic Multi-Objective

Building a complete assessment hardware setup that measures all the hardware specifications is not desirable regarding its cost and integratability. Therefore, the hardware specifications are partitioned

Table 6.10: The target and the achieved specifications for operational amplifier.

spec. name		K_o	$spec_o$	f_o	type
CMR	[V]	10	≥ 2	2.176	intrinsic
$SR\uparrow$[4]	[V/µsec]	100	≥ 0.01	0.0482	intrinsic
$SR\downarrow$[4]	[V/µsec]	100	≥ 0.01	0.05	intrinsic
$T_s\uparrow$[4]	[µsec]	100	≤ 100	10	intrinsic
$T_s\downarrow$[4]	[µsec]	100	≤ 100	10	intrinsic
$Offset$	[mV]	1	≤ 1	0.977	intrinsic
$Swing$	[V]	1	≥ 2	2.16198	intrinsic
A_0	[dB]	15	≥ 74	79.8	extrinsic
$CMRR$	[dB]	0.1	≥ 60	**42.4**	extrinsic
$PSRR$	[dB]	0.03	≥ 60	61.5	extrinsic
R_0	[kΩ]	1	≤ 40	13.9085	extrinsic
BW	[MHz]	1	≥ 10	25	extrinsic
ϕ	[°]	10	≥ 60	62.016	extrinsic
P_c	[mW]	1	≤ 5	1.54	intrinsic

into intrinsic specification set and extrinsic specification set as described in chapter 5. The operation point of the hardware is the most important for its basic operation, while it is sensitive to the deviations. On the other hand, the AC characteristics of the hardware are not strongly affected by the deviations. Therefore, the DC specifications are evaluated intrinsically by low or moderate speed, but accurate assessment setup, while, the AC specifications are evaluated extrinsic.

The experimental work is done to fulfill the given specifications in table 6.10. The hierarchical particle swarm optimization is employed in optimizing the mixtrinsic multi-objective evolution as it is shown above that it converges well for most of the hardware specifications variants in the previous experiments. The same setting of the HPSO for extrinsic evolution is employed. The mixtrinsic evolution is investigated by 10 runs; each run consists of 300 iterations. The optimization algorithm and simulation environment are tested on an AMD Athlon 64 2.4GHz processor, each run takes about 75 minutes. The mixtrinsic multi-objective evolution is published in advance in [TK07]. However the results in [TK07] are prepared using MSPSO, while this results are prepared by HPSO as it is experimentally more efficient.

As described above, in the current setup, the noise has relatively high influence on the intrinsic measurement. Thus, the measurement is repeated five times in the intrinsic evaluation, and the mean value of measurements is returned.

The convergence curve of the mean value of the 10 runs is shown in figure 6.34. Some oscillations appear in the curve due to the disturbance caused by the noise. The correspond standard deviation curve is shown in figure 6.35.

In table 6.10 column 3 *(f_o)*, the specifications of a best individual during one of the optimization runs is shown.

[4]The required value of the settling time and slew rate is limited to the analog to digital converter speed. Improving the settling time and the slew rate can be done by including them again extrinsically, but it is not necessary as they are implicitly included in the phase margin and the bandwidth optimization.

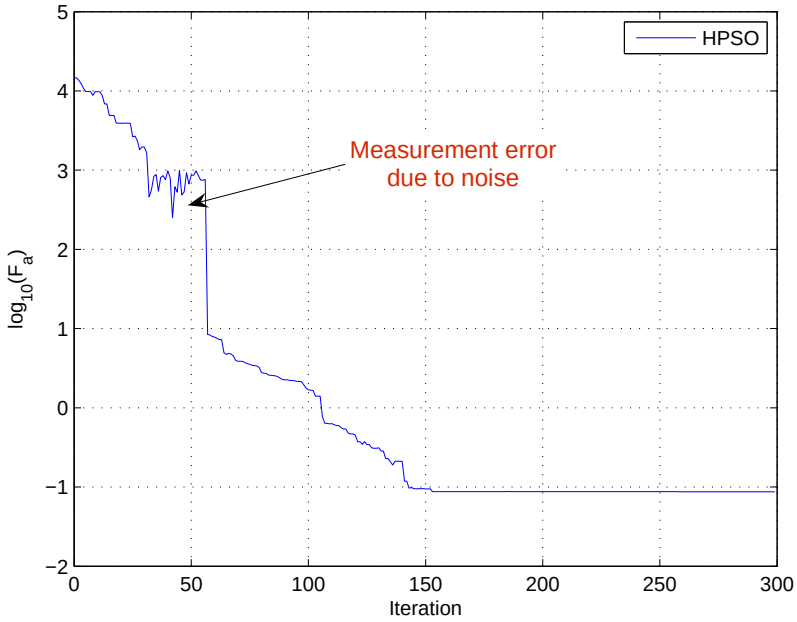


Figure 6.34: Mixtrinsic multi-objective evolution convergence curve.

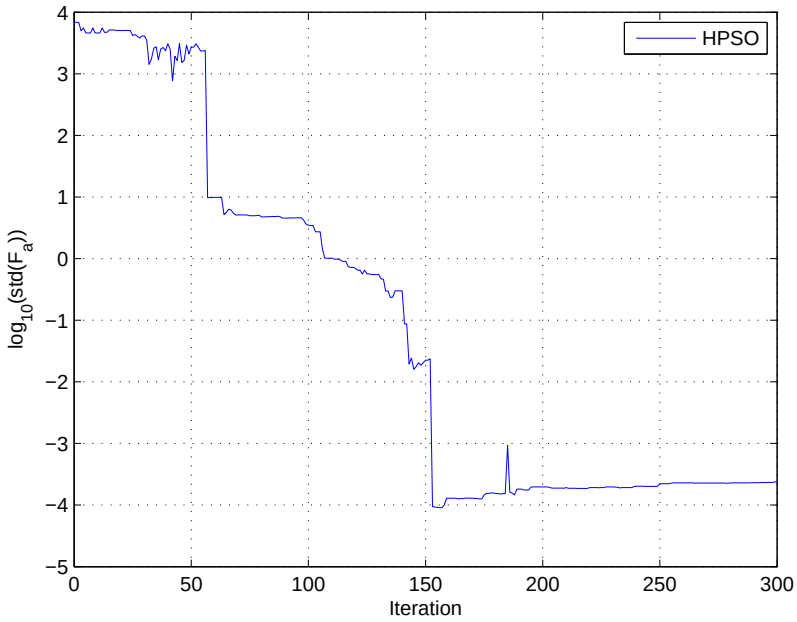


Figure 6.35: Mixtrinsic multi-objective evolution standard deviation curve.

An example of the returned configurations is shown in table 6.11. The specifications equivalent to this configuration is shown in table 6.10 column 4.

As shown in all the results, the output resistance of miller topology is high as it does not employ any output stage. If low output impedance is required, folded-cascode topology can be used. However, as it employs large search space (R25), it requires more iterations to find good solution. An example of

Table 6.11: Transistor widths and passive component values of a returned configurations in the mixtrinsic multi-objective evolution.

Component	Returned value
M1 [μm]	1
M2 [μm]	258
M3 [μm]	258
M4 [μm]	258
M5 [μm]	258
M6 [μm]	258
M7 [μm]	258
M8 [μm]	258
M9 [μm]	258
M10 [μm]	3
M11 [μm]	73
R12 [$k\Omega$]	18.375
C13 [pf]	14.25

Table 6.12: Transistor widths and passive component values of a returned folded-cascode operational amplifier configurations in the mixtrinsic multi-objective evolution.

Component	Returned value	Component	Returned value
M1 [μm]	177	M14 [μm]	255
M2 [μm]	255	M15 [μm]	255
M3 [μm]	255	M16 [μm]	254
M4 [μm]	255	M17 [μm]	204
M5 [μm]	252	M18 [μm]	255
M6 [μm]	255	M19 [μm]	255
M7 [μm]	255	M20 [μm]	115
M8 [μm]	55	M21 [μm]	254
M9 [μm]	255	M22 [μm]	255
M10 [μm]	255	M23 [μm]	69
M11 [μm]	233	R24 [Ω]	375
M12 [μm]	255	C25 [pf]	31.875
M13 [μm]	1		

the returned configurations of the mixtrinsic multi-objective evolution for folded-cascode operational amplifier is table 6.12. The performance of this amplifier and the required performance is shown in table 6.13. The evolution of this amplifier requires about 500 iteration or more to find a good solution.

As strong dynamic environment is already investigated intrinsically and extrinsically above, the mixtrinsic multi-objective evolution is investigated only in static environment. However, as shown in figure 6.34, it copes with environmental dynamics such as noise without starting from scratch. In order to achieve successful investigation for strong dynamic environment using mixtrinsic evolution, it is required either to measure the hardware temperature and utilize it in the simulation, or measure the operating point setting, and employ it in a modeled hardware.

Table 6.13: The target and the achieved specifications for folded-cascode operational amplifier.

spec. name		K_o	$spec_o$	f_o	type
CMR	[V]	10	≥ 2	*1.95*	intrinsic
$SR\uparrow^5$	[V/µsec]	100	≥ 0.01	0.0436	intrinsic
$SR\downarrow^5$	[V/µsec]	100	≥ 0.01	0.0512	intrinsic
$T_s\uparrow^5$	[µsec]	100	≤ 100	10	intrinsic
$T_s\downarrow^5$	[µsec]	100	≤ 100	10	intrinsic
$Offset$	[mV]	1	≤ 1	0.36	intrinsic
$Swing$	[V]	1	≥ 2	2.89	intrinsic
A_0	[dB]	15	≥ 86	87.4	extrinsic
$CMRR$	[dB]	0.1	≥ 60	144	extrinsic
$PSRR$	[dB]	0.03	≥ 60	70.5	extrinsic
R_0	[kΩ]	1	≤ 6	**6.134**	extrinsic
BW	[MHz]	1	≥ 10	25	extrinsic
ϕ	[°]	10	≥ 60	69.3	extrinsic
P_c	[mW]	1	≤ 5	0.442	intrinsic

6.3.2 CCII

The second generation current conveyor (CCII) is investigated in many current mode application [Wil90]. FPAA can be developed based on the CCII element [Gau97]. However, it is not commercially available, consequently, no industrial specifications is found describing its behavior so far. Therefore, the target specifications are chosen as the following:

- Settling time (T_s): The interval of time between step input current at the node X, and the output current at the node Z to be stable at the same value. The output is assumed to be stable if its ripple is less than $\pm 5\%$ of the final value. Similar to the operational amplifier, the settling time is optimized by two objectives; the rising settling time ($T_s\uparrow$), and the falling settling time ($T_s\downarrow$).
- Slew rate (SR): The maximum rate of change of the output current. It is consists of rising slew rate ($SR\uparrow$) and falling slew rate ($SR\downarrow$) as in the operational amplifier.
- Current offset (I_{offset}): The input current when the output current is zero.
- Voltage offset (V_{offset}): The voltage at the node Y (with respect to the virtual ground) when the node X is equal to virtual ground voltage.
- Band width (BW_{3dB}): The frequency range in which the current gain is more than -3dB.
- Linear current range (CR): The range of the input current in which the input-output current relation (at the nodes X and Z) is linear.
- Linear voltage range (VR): The range of the input voltage in which the input-output voltage relation (at the nodes Y and X) is linear.
- Current input range that keeps $V_X = V_Y$ when V_Y is connected to ground ($CR_{V_X=V_Y}$).

Table 6.14: The target and the achieved specifications for extrinsic evolution of the CCII.

spec. name		k_o	$spec_o$	f_o
CMR_I	$[\mu A]$	100	≥ 200	250
CMR_V	$[V]$	10	≥ 1.5	1.534
$SR\uparrow$	$[\mu A/\mu sec]$	1	≥ 50	189[6]
$SR\downarrow$	$[\mu A/\mu sec]$	1	≥ 50	192[6]
$T_s\uparrow$	[µsec]	10	≤ 1	0.4[6]
$T_s\downarrow$	[µsec]	10	≤ 1	0.6[6]
V_{offset}	[mV]	1	≤ 10	7.12
I_{offset}	$[\mu A]$	1	≤ 0.1	0.068
BW_{3dB}	[MHz]	1	≥ 10	316.2
P_c	[mW]	Not optimized		21
$CR_{V_X=V_Y}$	$[\mu A]$	100	≥ 200	250

The stability of the CCII is implicitly optimized by optimizing the settling time. The CCII in [ITF02] is chosen as a case study as it employs only 10 components, only two transistors are connected between the power supply terminals which reduces the voltage drop over the transistors, and consequently enhances the low voltage operation. The schematic of the CCII is shown in figure 6.36. The compensation capacitor $C_c = 50nf$ is added manually after the reconfiguration as it was found that although the circuit is stable in transient simulation, it is not stable if the simulation time step decreased without this capacitor.

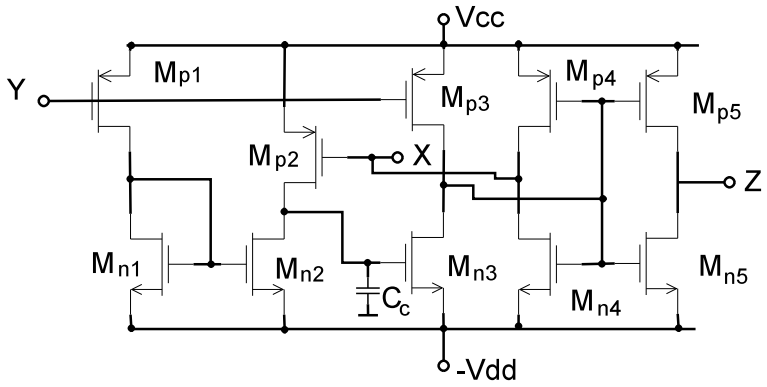


Figure 6.36: A low-voltage, low-power CCII implementation [ITF02].

The HPSO is employs to evolve the CCII building block with the same setting that is used to evolve the operational amplifier. The required specifications $spec_o$ and their weights k_o are given in table 6.14. The lengths of the transistors are fixed to 1µm, and the width is programmable with the range of $1-257\mu$m. The HPSO returned the CCII with the dimensions in table 6.15 after 33 iterations. The achieved specification values f_o by the configuration in table 6.15 is given in last column in table 6.14. The frequency response of the evolved CCII is shown in figure 6.37. The frequency response after adding the compensation capacitor to the original design in [ITF02] is shown in figure 6.38. The relation between the input current I_X and the output current I_Z when $V_Y = 0$ is shown in figure 6.39. The relation between the input current I_X and the voltage V_X when $V_Y = 0$ is shown in figure 6.40.

[6]This values are obtained after adding the compensation capacitor C_c to the circuit

Table 6.15: The CCII configuration that the evolution returned.

Transistor name	Transistor width
M_{P1}	257
M_{N1}	255
M_{P1}	257
M_{N2}	257
M_{P3}	257
M_{N3}	1
M_{P4}	257
M_{N4}	1
M_{P5}	257
M_{N5}	93

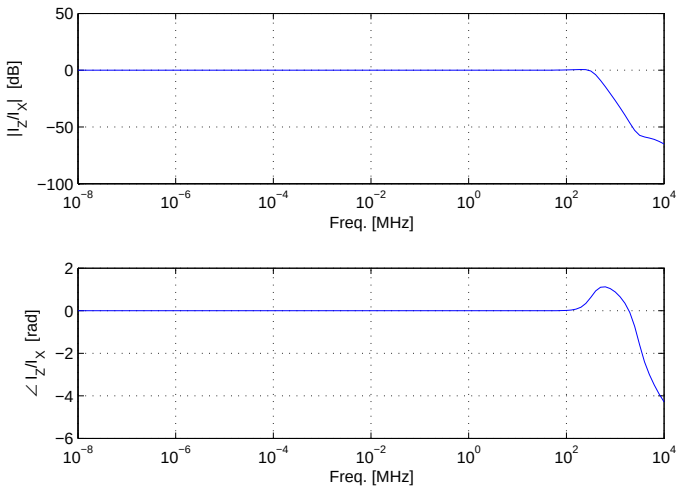


Figure 6.37: The frequency response of the evolved CCII.

The step response of the CCII after adding the compensation capacitor is shown in figure 6.41. The relation between input voltage V_Y and output voltage V_X when $I_X = 0$ is shown in figure 6.42.

6.4 Functional Level

In this section, the hardware is evolved using hierarchical optimization, therefore, the building block configurations obtained above are employed to evolve the functional level hierarchically. As the target functional level blocks are not available currently in a real hardware, they are evolved extrinsically. The instrumentation amplifier is not investigated as it was faulty in the first chip. It will be investigate intrinsically in the availability of the new chip FPMA2, which is currently in the manufacturing phase.

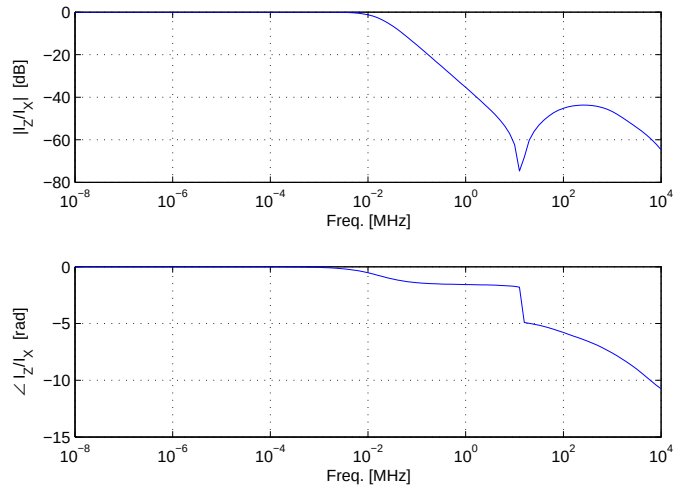


Figure 6.38: The frequency response of the evolved CCII after adding the compensation capacitor.

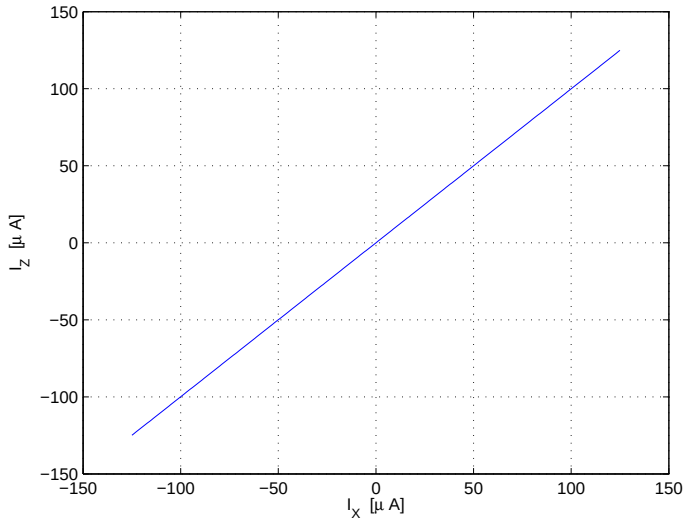


Figure 6.39: The relation between the input current I_X and the output current I_Z of the evolved CCII.

6.4.1 Flash ADC

Analog to digital converters are the interfacing element between the analog and the digital sensor electronics. In chapter 2, it is described that the state-of-the-art ADCs have self-calibration capability that enhance coping with static and dynamic deviations. Some sensor electronic applications such as high speed image sensors require fast ADCs with low resolution. Therefore, slow but accurate ADC with self-calibration capability can be employed to evolve fast ADC such as flash ADC.

The schematic of the 3 bit flash ADC is shown in figure 6.43. Its evolution is done by hierarchical optimization [TK06b] in which, first the comparators are optimized to reduce the offset to be less than $1mV$, the propagation delay be less than $20nsec$, and the CMR to $2V$. Then, they are utilized in designing the ADC by adding programmable resistors to them, thereafter, ADC is optimized by

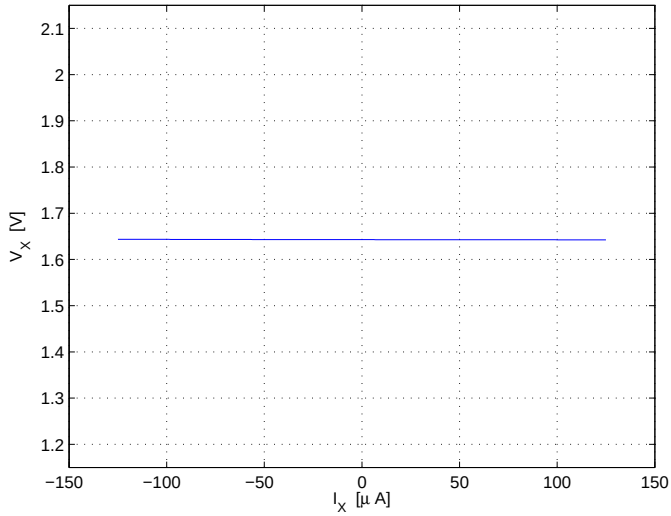


Figure 6.40: The relation between the input current I_X and the output voltage V_X of the evolved CCII.

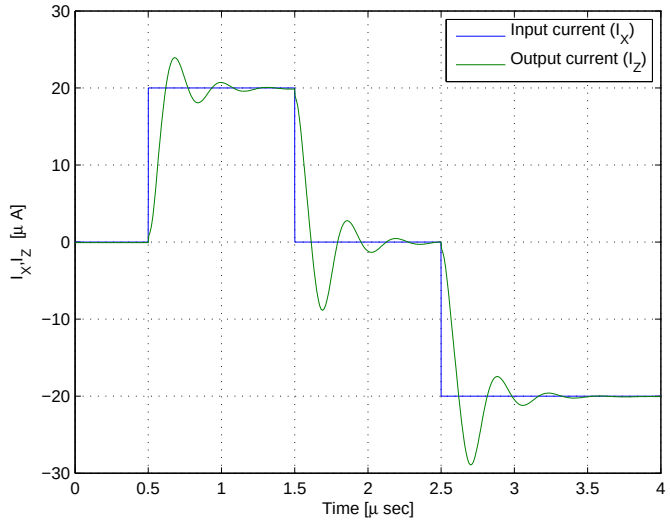


Figure 6.41: The step response of the evolved CCII after adding the compensation capacitor.

programming the passive components. Each of the components is replaced by a programmable one.

The reference voltage of the ADC in the experimental work is $1V$ with respect to the virtual ground. The virtual ground is $1.65V$, which is half of the supply voltage. Each of the output transition levels is considered as an objective function. The optimization aim to adjust the transition levels of the ADC to a reference transition levels, which minimizes the conversion error. These transition levels are the outputs of the comparators in figure 6.43. The target transition levels are given in table 6.16. The RMS error between the target and the achieved transition levels is minimized. The speed of the flash ADC is optimized by optimizing the comparators propagation delay. For simplicity, the optimizing of the quiescent power consumption is not included in the optimization. The 3-bit flash ADC published in advance in [TK06b], where the comparators propagation delay was $200nsec$, the common mode range was not optimized, and the objective function was minimizing the total error between the input

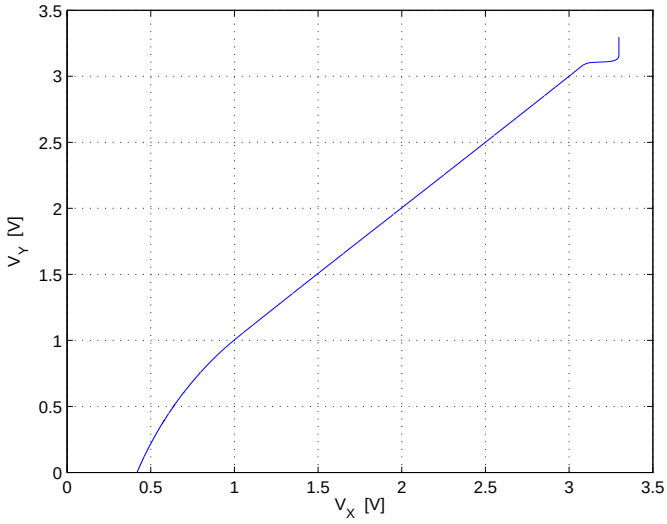


Figure 6.42: The relation between V_X and V_Y.

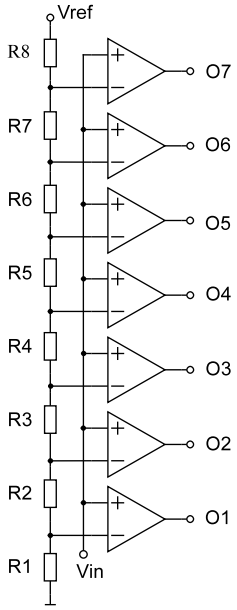


Figure 6.43: The target 3 bit ADC flash converter.

single and the quantized signal, which implicitly optimizes the ADC signal, but does not describe the behavior, in addition, as the RMS error was not optimized, the error in some levels may get high in order to reduce the error of the others.

A returned comparator configuration is given in table 6.17.

Two deviation models are investigated; the first deviation model is increasing the resistor R5 by 50%, decreasing R6 by 25%, and increasing R8 by 20%, which can occurs due to manufacturing tolerance, or thermal distribution. The second deviation is a simulated increase of the temperature of the comparator transistor M8 to 100°C.

Table 6.16: The required and the achieved transition levels for the flash ADC, and the effect of the deviations on them.

Transition levels figure 6.43	Transition voltage	Initial configuration	Applying model 1 deviations	Recovering model 1 deviations	Applying model 2 deviations	Recovering model 2 deviations
O1 [V]	0.0625	0.062	0.058	0.063	0.001	0.063
O2 [V]	0.1875	0.188	0.175	0.187	0.125	0.187
O3 [V]	0.3125	0.312	0.292	0.312	0.25	0.312
O4 [V]	0.4375	0.437	0.408	0.437	0.374	0.437
O5 [V]	0.5625	0.565	0.585	0.568	0.499	0.562
O6 [V]	0.6875	0.684	0.673	0.685	0.625	0.686
O7 [V]	0.8125	0.811	0.789	0.814	0.748	0.811

Table 6.17: Transistor widths and passive component values of a returned comparator.

Component	Returned value
M1 [μm]	1
M2 [μm]	257
M3 [μm]	257
M4 [μm]	257
M5 [μm]	257
M6 [μm]	35
M7 [μm]	180
M8 [μm]	181
M9 [μm]	1
M10 [μm]	2
M11 [μm]	1
R12 [$k\Omega$]	31.875
C13 [pf]	No used

The HPSO is used in this investigation, its population consists of 20 particles. The algorithm run for 300 iterations. The values of inertia weight and acceleration coefficients are similar to the previous experiments; w starts with 0.9, and decreases linearly to 0.4, $C_1 = 2.05$ and $C_2 = 2.05$.

An appropriate initializer is implemented that initializes the ADC resistors by generating a single random number, and employing the design procedures to calculate the value of all the resistors using this random number. In table 6.18 column 2, the initialization rules for the resistors are mentioned, where R_{rand} is an integer random value in the range of 2 to $\frac{2\times255}{3}$ multiplied by the step size of the resistance which is 125Ω. This initialization results in finding the initial configuration during the initialization of the population, and recovering from deviations within few iterations. However, for strong deviations, uniform initializer performs better.

The output of the comparators are simulated by a thousand sample, therefore, the minimum detectable transition voltage error is 0.001 V. The achieved transition levels of the initial configuration is shown in table 6.16 column 3. The initial configuration itself is shown in the third column in table 6.18. After applying the first deviation model to this configuration, the transition levels deviates to the transition levels in the fourth column in table 6.16, while after applying the second model of deviation,

Table 6.18: The resistor values before and after recovering.

Resistor name	Initialization rule	Initial configuration	Recovering from first deviation model	Recovering from second deviation model
R1 [$k\Omega$]	$\frac{R_{rand}}{2}$	5.357	12.5	31.875
R2 [$k\Omega$]	R_{rand}	10.875	25.375	31.625
R3 [$k\Omega$]	R_{rand}	10.875	25.125	31.875
R4 [$k\Omega$]	R_{rand}	10.875	25.250	31.875
R5 [$k\Omega$]	R_{rand}	10.875	17.375+50%	31.875
R6 [$k\Omega$]	R_{rand}	10.875	31.875-25%	31.875
R7 [$k\Omega$]	R_{rand}	10.875	25.75	31.875
R8 [$k\Omega$]	$\frac{3\times R_{rand}}{2}$	16.250	31.875+20%	31.875

they deviate to the sixth column in the same table. The reconfiguration of the flash ADC finds the configuration in table 6.16 column 4 and 5 to recover from the first and the second deviation models respectively. The value of the transition voltage after recovering is given in table 6.16 column 5 and 7 respectively. In figure 6.44, the quantized signal of the initial configuration, the effect of the first model of deviation on it, and the quantized signal after recovering are shown. Similarly, in figure 6.44, the quantized signal by the initial configuration, the effect of the second model of deviation on it and the quantized signal after recovering are shown.

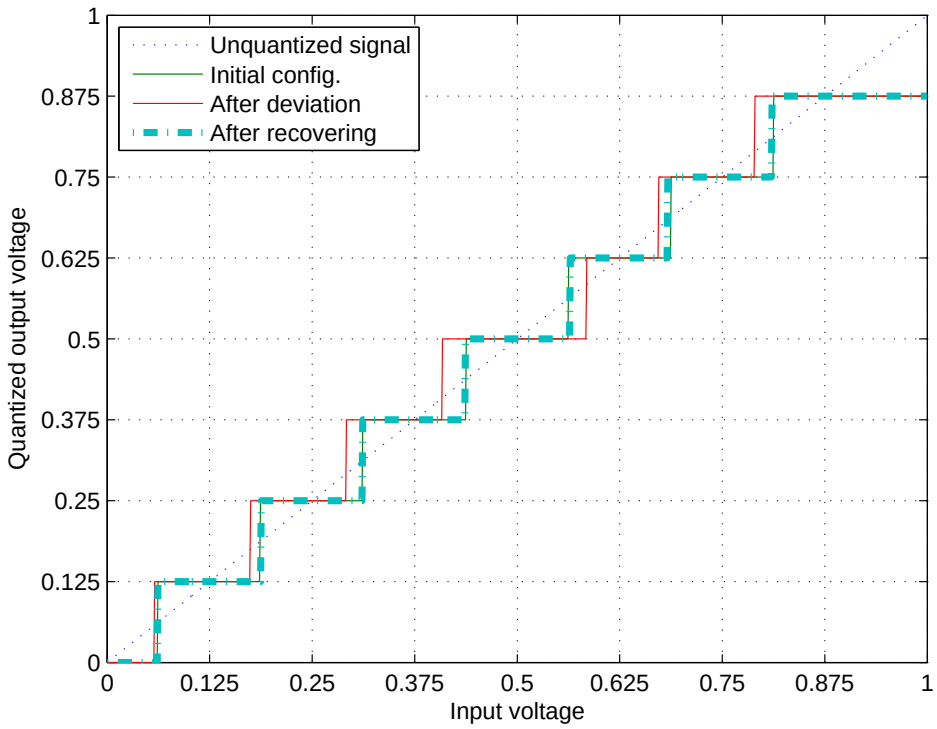


Figure 6.44: The quantized signal by the initial configuration, the effect of the first deviation model on the quantized signal, and the signal after recovering.

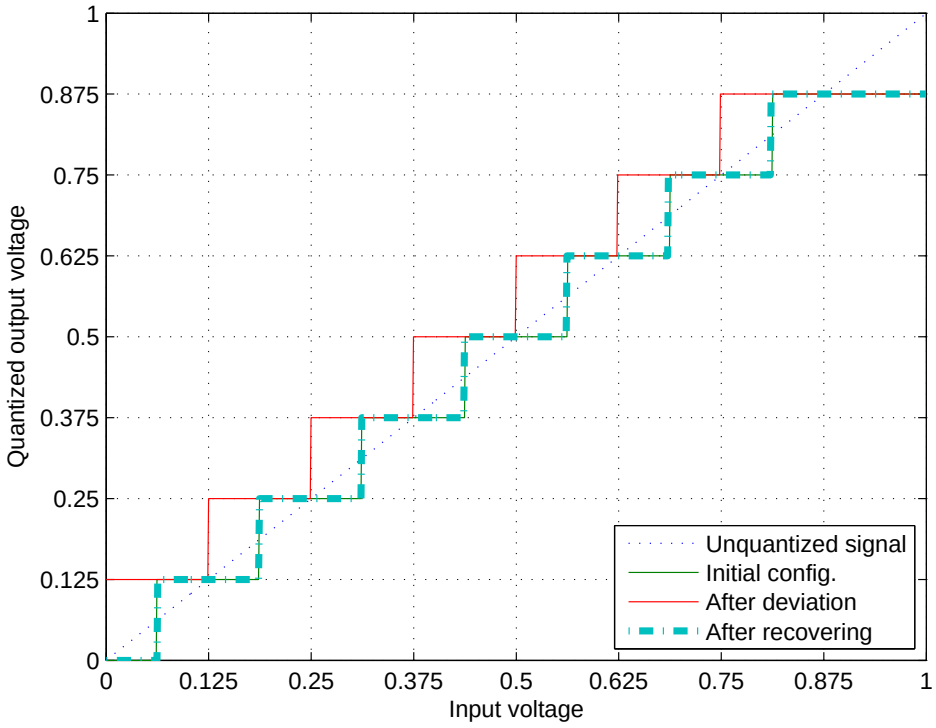


Figure 6.45: The quantized signal by the initial configuration, the effect of the second deviation model on the quantized signal, and the signal after recovering.

6.4.2 Low-Pass Filter

Low pass filters are widely used in sensor electronics to reduce the noise, as an anti-aliasing filter before the ADCs, etc. Therefore, it is selected as a case study for the hierarchical optimization of the evolvable hardware. The low pass filter is described by the specification in figure 6.46. It consists of three bands; the pass band at the frequency zero to f_p, the transition region with frequency range f_p to f_s, and the stop band which its frequency is greater than f_s. The signal gain in the pass band is greater than $2-\delta_p$ and less than δ_p. The stop band gain is less than δ_s. The amplitude in the transition region is not defined in the design of the filter. Therefore, δ_p and δ_s are optimized in the filter for a given pass and stop frequency. The tool is design in which any filter can be extrinsically optimized as it reads the filter netlist from external file that contains information about the filter structure, and the programmable components. The ranges of the programmable components are given in another setting file. The schematic of the target filter for this experiment is shown in figure 6.47.

The programmable resistors and capacitor for the filters are 11 bits in order to cover large frequency range. The resistor programmable range is from 250Ω to $511.75k\Omega$ with resolution of 250Ω. The capacitor programmable range is from $0.125pf$ to $255.88pf$ with resolution of $0.125pf$. The filters are optimized in hierarchy in which, first the building blocks such as operational amplifiers are optimized, then the filter passive components. The amplifier in table 6.6 is employed in this investigation. Two filter different setting are optimized, the first filter settings is given in the second column in table 6.19, while the settings of the second filter is given in the third column in the same table. The evolution returns a solution within few seconds. The frequency response of the returned filters is shown in

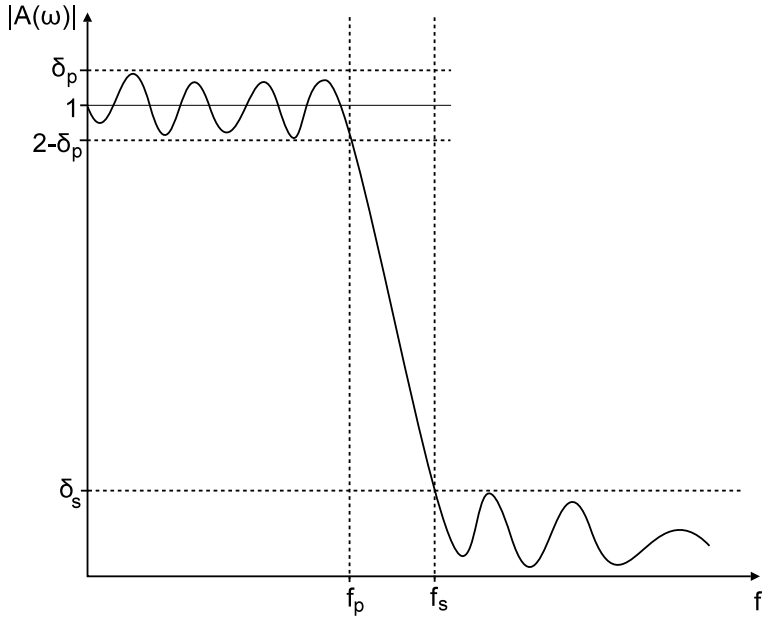


Figure 6.46: Depiction of filter specification.

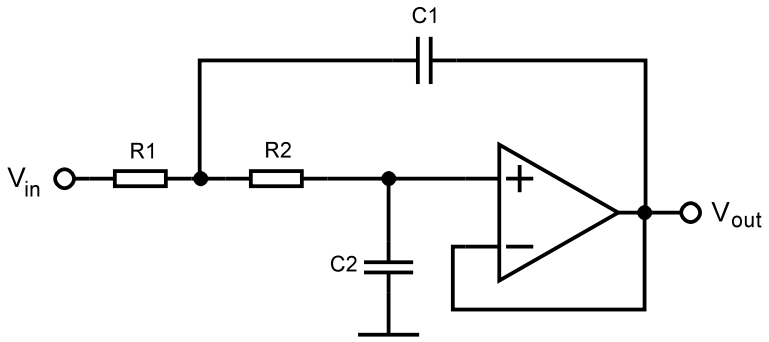


Figure 6.47: Schematic of the target voltage mode filter.

Table 6.19: The requirement of the filter.

	First filter	Second filter
f_p [Hz]	10k	100
f_s [Hz]	100k	1k
δ_s [dB]	-40	-40
δ_p [dB]	0.82785	0.82785

figure 6.48 and 6.49 respectively. The dimensions of the passive components that the evolution return is shown in table 6.20.

The filter in figure 6.50 is an example of the current mode functional level blocks. The optimized CCII+ dimensions are given in table 6.15. This filter is optimized to the same specifications in table 6.19. The filter response of the first and second filter settings are shown in figures 6.51 and 6.52 respectively. The dimensions of the returned filters are given in table 6.21.

The achieved setting in voltage and current mode circuits are shown in table 6.22. The evolution could

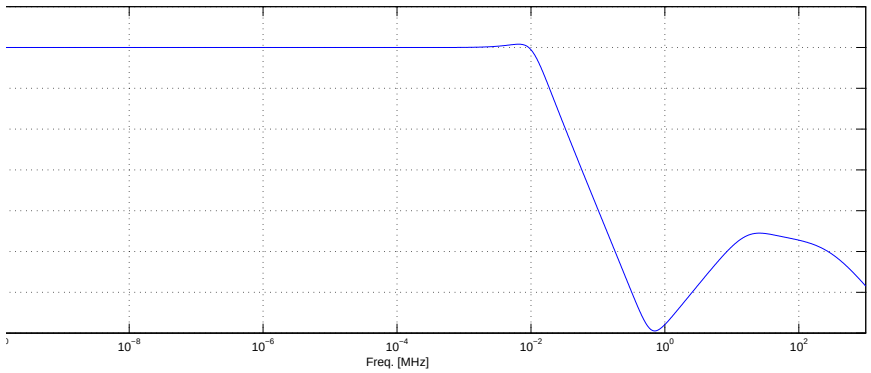


Figure 6.48: The returned voltage mode filter frequency response of the first setting filter.

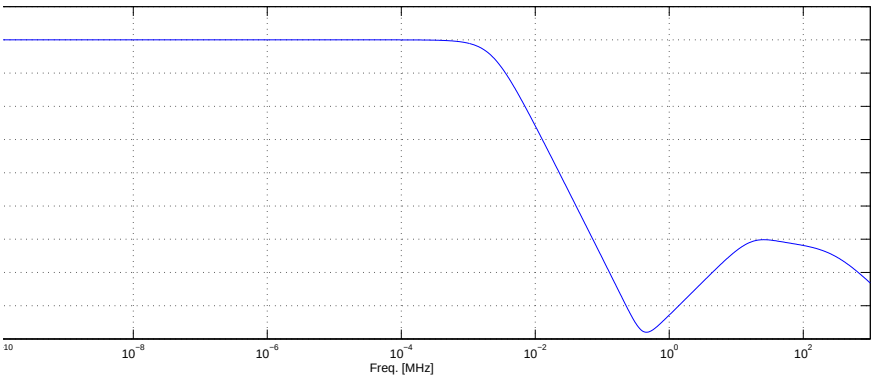


Figure 6.49: The returned voltage mode filter frequency response of the second setting filter.

Table 6.20: Returned voltage mode filter configurations.

Component	First filter	Second filter
R1 $[k\Omega]$	89.750	487.750
R2 $[k\Omega]$	152.250	260.250
C1 $[pf]$	255.875	229.25
C2 $[pf]$	69.625	165.75

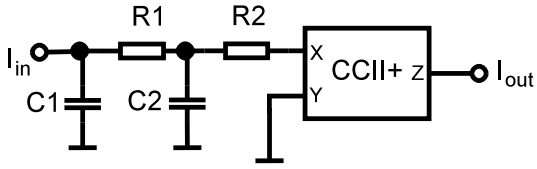


Figure 6.50: Schematic of the target current-mode filter.

not find a suitable configuration for the second required filter settings. The weight of the pass band gain is adjusted higher than the weight of the stop band by the engineering knowledge. Therefore, the returned filter has acceptable setting due to employing the engineering knowledge even if the filter structure does not support the requirements.

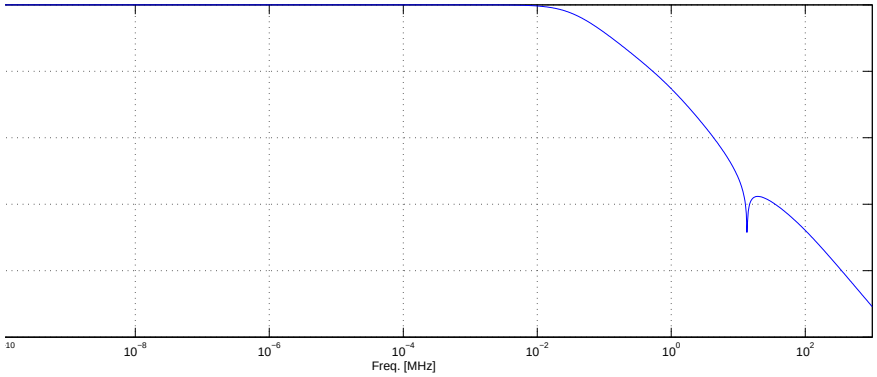


Figure 6.51: The returned current mode filter frequency response of the first setting filter.

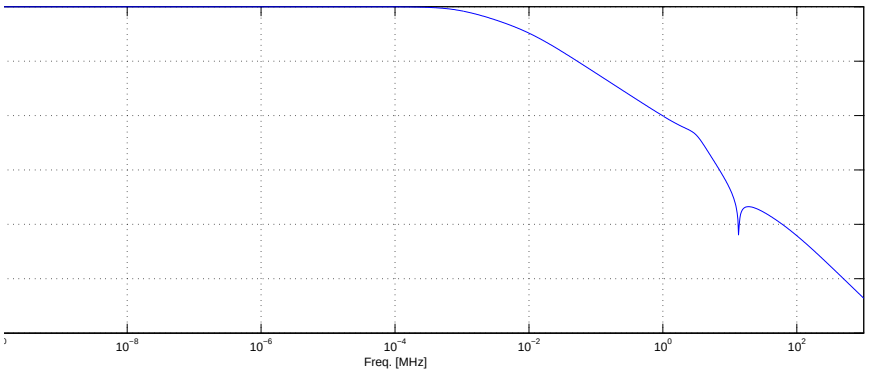


Figure 6.52: The returned current mode filter frequency response of the second setting filter.

Table 6.21: Returned current mode filter configurations.

Component	First filter	Second filter
R1 $[k\Omega]$	510.25	165.750
R2 $[k\Omega]$	20	419.250
C1 $[pf]$	9.875	199.5
C2 $[pf]$	255.875	206.5

Table 6.22: The achieved filter settings

	Voltage mode		Current mode	
	First filter	Second filter	First filter	Second filter
δ_p [dB]	0.82343	0.010148	0.82785	0.064332
δ_s [dB]	-40.082	-44.844	**-21.199**	-40.758

CHAPTER 7

Conclusion and Summary

7.1 Conclusion

The analog sensor electronics requires special care in the design time, trimming or calibration in the deployment time and the run time, which increase the design effort, production time, and consequently the cost. Nevertheless, dynamic deviations cannot be completely considered in the state-of-the-art sensor electronics. On the other hand, manufacturing a specially designed hardware for each sensor increases the development time and cost. The state-of-the-art FPAAs offers programmable analog hardware with flexibility only in the functional level, while its building blocks are not programmable. Thus, it cannot cope with all the deviations, and its applicability is limited to few of the sensor electronic applications. Therefore, generic sensor electronics platform that inherent the self-x properties is desirable. The state-of-the-art evolvable hardware attempts to accomplish the rapid prototyping and to include the self-x properties. On the other hand, it invents the hardware topology without considering the industrial requirements, and it does not concern its standard specifications. Consequently, it returns hardware with unpredictable behavior that cannot be accept in industry.

In this thesis, a design flow for generic organic-computing sensor electronics is presented to synthesize the user target design and include the self-x properties, while it considers the industrial requirements. The following points are investigated in the presented design flow:

- Evolving the hardware by hierarchical optimization to simplify the hardware search space and return hardware, in which, the behavior of all its builing blocks are predictable, and consequently, its entire behavior.
- Optimizing the hardware by employing real measurement for its specifications in order to recover from deviations.
- Employing low cost integratable assessment setup that is suitable for embedded system applications.
- Investigate techniques that allow the optimization to use the old information it has during the dynamic trimming in order to reduce the reconfiguration time.

In order to return hardware with predicable behavior, the hardware topologies are constrained to the known topologies, and the standard hardware specifications are optimized. In order to recover from deviations, the evolutionary computation employs intrinsic evolution that measures the hardware specifications by a hardware measurement setup. Apart from the state-of-the-art evolutionary electronics, the standard specifications of the hardware are evolved intrinsically, and the optimization does not start from scratch after each environment change in order to reduce the dynamic trimming required time. However, some of the hardware specifications require special high cost measurement setup, while some others are time consuming, which limit the applicability of the intrinsic multi-objective evolution. Thus, the novel proposed approach "mixtrinsic multi-objective evolution" evolves the hardware specifications that are sensitive to deviations intrinsically, and the rest of the specifications extrinsically. The measurable environmental parameters such as the temperature of the chip can be included in the extrinsic evaluation to increase the accuracy. The complete sensor electronic system is evolved in hierarchy in which, the basic blocks are evolved first, then the functional blocks, etc. Therefore, low cost integratable assessment setup is sufficient for the calibration loop of the generic organic-computing sensor electronic hardware, which is essential for embedded system applications. The behavior of the complete system is predictable as it employs standard topologies and optimizes its specifications. As the complete system is optimized in hierarchy, the optimization is simplified as the search space is partitioned into sub-problems, and the behavior of the returned hardware is predictable as standard specifications of the building blocks are optimized.

7.2 Novel Contributions of the Thesis

The design flow for reconfiguring the generic organic-computing sensor electronics presented in figure 6.1 aim to convert a user given design to a valid configuration, and trim the hardware afterwards to recover from any further deviations in the run time.

This thesis focuses on blocks marked in gray in figure 6.1; the hierarchical optimization of the generic sensor electronics, and dynamic trimming of the building and functional blocks. The novelties introduced by this thesis are summarized in the follows:

- Considering the industrial requirements to evolve hardware with predictable behavior.
  - Optimizing the industrial specifications of the sensor electronics hardware extrinsically [TK05a, TLK05, TK06c] and intrinsically [TLK06]
  - Adapting the evolutionary computation to constrain the flexibility of the hardware to standard topologies in current- and voltage- mode circuits during the extrinsic [TK06c], the intrinsic [TLK06] evolution.
  - Evolving the complete system by hierarchical approach in which the building block are evolved first, then the functional level blocks [TK06b].
  - Proposing a rough design flow for the aspired generic organic-computing sensor electronics.
- Investigation embedding the optimization loop with the evolutionary computation algorithm and the generic reconfigurable sensor electronics hardware without the need of high cost measurement equipment.

  - Proposing and investigating a novel scheme, *mixtrinsic multi-objective* evolution in which; the hardware specifications are partitioned into intrinsic and extrinsic sets according to their features, then each of them is evaluated intrinsically, extrinsically, or *both*. The mixtrinsic multi-objective optimization allows low-cost assessment for reconfigurable sensor electronics [TK07], which is necessary for embedded system applications.
  - Proposing a methodology that can employ estimation or behavioral models to evaluate the extrinsic set of specifications in the mixtrinsic multi-objective optimization in order to reduce the computational effort. Both models have been applied in synthesis tools, but not in evolvable hardware.
  - Proposing hybrid models that can combine simulation, estimation, and behavioral models to balance between the computational effort and the accuracy of the evaluation.

- Investigating PSO in optimizing the sensor electronics generic hardware, and search for potential improvement in the state-of-the-art.
  - Employing PSO in extrinsic [TK05b, TK06c], intrinsic [TLK06] and mixtrinsic multi-objective [TK07] evolution of analog reconfigurable hardware to improve the conversion speed, and simplify the implementation.
  - Applying the various state-of-the-art PSO approaches to the target aspired generic reconfigurable organic-computing sensor electronics hardware, and compare them with each other [TLK06] and with the GA [TK06c].
  - Applying dynamic environment approaches to evolve the sensor electronics reconfigurable hardware without starting from scratch after each environmental change *(trimming in dynamic environment)*, which results in improvement in the convergence speed and optimality of the solution [TLK06].

- Improving the particle swarm optimization technique by employing local parameters to each individual, and a controller to control them [TK06d].

7.3 Future Work

On the availability of the hardware, further research in decomposition and mapping techniques such as the techniques employed for FPAAs [WV02, BRL+05] will be abstracted to the generic organic computing sensor electronics, and topology selection techniques such as classification based selection techniques [PF98] will be investigated.
After investigating all the design flow components, the user interface will be designed to convert the user entry into an intermediate hardware description language.

7.4 Kurzfassung in Deutscher Sprache

7.4.1 Motivation

Sensornetzwerke sind allgegenwärtig auf dem Gebiet der modernen Systeme wie der Automobiltechnik, der Automatisierungstechnik, der Medizintechnik, ”Ambient Intelligence”, usw. Der Entwurf von Sensorelektronik erfordert besondere Sorgfalt wie Ausbeuteoptimierung und den Entwurf unter Berücksichtigung des ungünstigsten Falles um die Einflüsse der Abweichungen zu verringern. Jedoch wird im Entwurf von typischen Arbeitspunktswerten ausgegangen und daher die Einflüsse der statischer und dynamischer Abweichungen nicht vollständig berücksichtigt. In vielen Anwendungen, ist die Justierung in der Herstellungsphase von groer Notwendigkeit, da die Sensorelektronik für Herstellungsabweichungen und andere Einflussgröen anfällig ist, die auf die Zielspezifikationen Einfluss nehmen könnten. Die Abweichungen aufgrund von Alterung können durch die Kalibrierung in der Inbetriebnahme behandelt werden, zudem können die Selbst-Kalibrierungstechniken einige der dynamischen Abweichungen bewältigen. Allerdings sind nicht alle Abweichungen kompensierbar, da die Kalibrierung durch den Einsatz wenigen programmierbaren Komponenten erreicht wird und nur wenige der Hardware-Spezifikationen in die Schleife berücksichtigt werden. Zum Beispiel, die sogenannten Rejustors von Microbridge Technology Corp., die aus einem analog kontrollierten Widerstand, um die dynamischen Abweichungen durch ein Rückkopplungssignal zu bewältigen; das Rückmeldungssignal wird digital in vielen Anwendungen kontrolliert. Der PGA309 ist ein programmierbarer Verstärker von Texas Instruments, der digitale Temperaturkalibrierung und sogenanntes Autozeroing einsetzt, um die dynamischen Abweichungen zu bewältigen. So versucht die Industrie robuste jedoch nicht fehlertolerante Systeme zu bauen. Robuste Systeme können die zu erwartenden Abweichungen bewältigen, während die fehlertoleranten Systeme für die Behandlung der unerwarteten Variationen geeignet sind.

Für Multi-Sensorsysteme, die eine besondere Sorgfalt in der Planungsphase erfordern, eine Justierung bei der Herstellungsphase, und die Verwendung von Kalibrierungsverfahren für jeden der Sensoren erhöht die Kosten der Sensorelektronik. Darüber hinaus verursacht das Hinzufügen von neuen Sensoren viele Änderungen in der Hardware. Rapid Prototyping ist ein offenes Thema, da die Kosten verringert werden können für die gleiche Hardware, die für viele Sensoren und viele Anwendungen programmiert werden kann. Deshalb ist die rekonfigurierbare Sensor Elektronik mit self x Eigenschaften wünschenswert.

Kürzlich, wurden genetische Programmierung und genetische Algorithmen für die Synthese der analogen Hardware angepasst zum Einsatz gebracht. Die genetische Programmierung wurde angewendet, um Hardware mit beliebiger Topologie [KFHBAK97] zu gestalten. Dies kann zu unvorhersehbarem Verhalten führen und u.a. zu exzessiven Bauelementdimensionen führen Dem gegenüber wurden genetische Algorithmen zur Synthese von Standardschaltungstopologien [ZPV02] angewandt. Jedoch können statische und dynamische Abweichungen in diesen Syntheseansätzen nicht oder nur eingeschränkt berücksichtigt werden.

Auf der anderen Seite bietet analoge rekonfigurierbare Hardware, - auch evolutionären Hardware genannt, das Potenzial statische und dynamische Abweichungen [SKZ$^+$00] zu kompensieren. Zum Stand der Technik werden genetische Algorithmen eingesetzt, um beliebige Topologie [HHE02, SKZ$^+$00, Tre06] analoger Schaltungen zu erfinden. Allerdings der Stand der Technik evolutionärer Elektronik für ein dynamisches Umfeld kein geeigneter Ansatz, um effizient auf Veränderungen der

Randbedingungen zu reagieren. In der Tat wird nach jeder Veränderung der Umwelt mit dem Optimierungsprozess wieder bei Null begonnen [Tho97a, KZJS00, SKZ01, ZGK+04]. Keine spezielle Technik oder Methode ist auf dem neuesten Stand der Technik aufzufinden, die auftretende Abweichung in einer Hardware geeignet detektiert. Stattdessen, sind die Abweichungen manuell erstellt und kontrolliert. Zum neuesten Stand der Technik evolutionärer Elektronik, ist die Zahl der erforderlichen Bausteine, um eine geeignete Konstruktion zu finden, die den angegebenen Spezifikationsforderungen genügt, nicht bekannt, [Tho96, KFHBAK97, SKZ+00, Tre06]. Allerdings wird die maximale Anzahl der erlaubten Blöcke durch Versuch und Irrtum bestimmt. Dadurch, ist die verwendete Fläche des Bausteins zur Erfüllung der gewünschten Funktionalität nicht genau vorhersehbar.

Um auftretende Abweichungen kompensieren zu können, muss die sogenannte intrinsische Evolution der Hardware angewendet werden. Die offenen Fragen, die zum neuesten Stand der Technik entsprechender Elektronik zur Kompensation relevanter Abweichungen bei Erlangung industrieller Akzeptanz noch nicht berücksichtigt sind, sind folgend aufgeführt:

- Das Verhalten der gewählten Topologie sollte vorhersehbar sein und Standards entsprechen.
- Die industrielle Spezifikationen der Hardware muss optimiert sein entsprechend den Anforderungen einer Anwendung.
- Messung der kompletten Spezifikationen der Hardware erfordert teure Messgeräte, die die Anwendbarkeit des Ansatzes einschränkt. Daher wird zum Stand der Technik hat sich nur die Abweichung zwischen den ideale und den realen Ausgangswerten der vorliegenden Elektronik bestimmt. Die Messung der transienten und der Kleinsignaleigenschaften der Hardware ist durch die Umwandlungsgeschwindigkeit der verwendeten ADCs und DACs beschränkt.
- Die Kalibrierungschleife muss die Abweichung kompensieren. Da die evolutionäre Hardware anhand der aktuell eingeschriebenen Konfiguration das Verhalten der realisierten Schaltung auf Konformität mit der gewünschten Spezifikation prüft, gehen alle Abweichungen der verwendeten Messungseinrichtung als Fehlerquellen in das Resultat ein. Obwohl dieses Kernproblem zum Stand der Technik keine adäquate Behandlung erfährt, bietet evolutionäre Hardware prinzipielle Selbst- oder Eigen-Kalibrierungseigenschaften in digitalen [LS92, KLB93, FDLH98, WHL04] und analogen Schaltungen [CGN04, RRS+04]. Allerdings erfordern kompliziertere Messungen mehr Sorgfalt für die Eigen-Kalibrierung in allen Komponenten.

Die vorliegende Arbeit bietet im Hinblick auf den angesprochenen Stand der Technik und verbleibende Probleme Lösungsansätze für den industriellen Einsatz von generischer, durch Organic-Computing Ansätze inspirierte Sensorelektronik für eingebettete Anwendungssysteme.

7.4.2 Beiträge dieser Arbeit

Die industrielle Applikation erfordert zuverlässige Elektronik mit vorhersagbaren Verhalten in Übereinstimmung mit üblichen industriellen Spezifikationen. Die Zulassung beliebiger Schaltungstopologien und die Optimierung des Systems nur durch Minimierung des Fehlers zwischen Ist- und Soll-Ausgangsignal wird möglicherweise Elektronik mit unerwarteten, unvorhersagbarem Verhalten ergeben. Dies wird nur eine sehr eingeschränkte Akzeptanz unter industriellen Gesichtspunkten erwarten lassen.

Damit die Elektronik vorhersagbares Verhalten aufweist, werden Einschränkungen der Hardware-Flexibilität in Form der Vorgabe von Standardtopologien gemacht. Weiterhin wird die Elektronik gemä Standardspezifikation optimiert, d.h. ein multikriterielles Optimierungsproblem in Angriff genommen und eine Lösung gefunden. Die vorgebenenen Spezifikationswerte resultieren aus dem jeweiligen Bedarf der vorliegenden Anwendung.

Um die sogenannten Self-x Eigenschaften von Organic Computing Systemen zu erzielen muss das Verhalten der Hardware in einer Kalibrierungsschleife bestimmt bzw. gemessen werden. Dies erfordert eine Bewertungseinheit um die Hardwarespezifikation zu bestimmen sowie eine Optimierungseinheit um unter Einsatz von biologoisch-inspirierten Verfahren nach einer nutzbaren Konfiguration. Weiter hin wird eine Einheit zur Vor- und Nachbearbeitung und der Kommunikation zwischen den Einheiten benötigt. Daher besteht das angestrebte, durch Organic-Computing-Konzepte inspirierte, generische Sensorelektronik-Frontend besteht aus vier Blöcken, wie in Abbildung 7.4.2 wiedergegeben, zur Optimierung, Vor- und Nacherarbeitung, der Bewertung, und der rekonfigurierbaren Elektronik selbst.

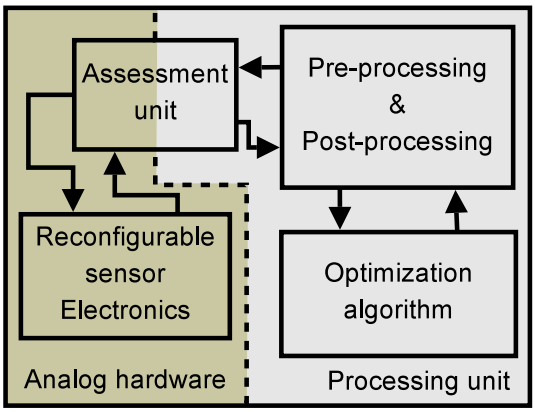


Figure 7.1: Block diagram of the reconfigurable generic sensor system.

Die Zielhardware für diese Arbeit ist eine analoge rekonfigurierbare Hardware mit programmierbarer Struktur funktionaler Ebene und Dimensionen in der Bausteinebene, wie in Abbildung 7.2 zu sehen ist. Ein Beispiel einen Baustein der Zielelektronik ist der programmierbare, zeitkontinuier-

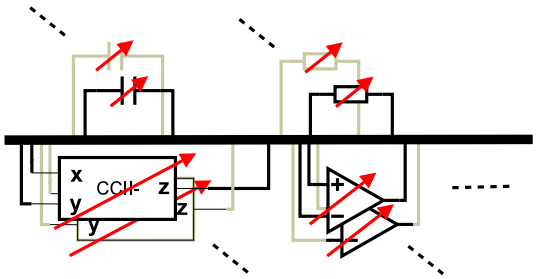


Figure 7.2: Blockschaltbild der rekonfigurierbaren generische Sensor System.

lich arbeitende Operationsverstärker, vorgeschlagen von Lakshmanan u.a. [LK05], in dem Standardschaltungstopologien mit programmierbaren Dimensionen von aktiven und passiven Bauelementen eingesetzt werden. Der sogenannte "'Current Mode"' oder die strombezogene Schaltungstechnik, - das bedeutet, dass die einzelnen Schaltelemente durch Ströme, nicht durch Spannungen interagieren [TLH90] - arbeitet mit niedrigen Betriebsspannungen, hohen Geschwindigkeiten und der

notwendigen kleinen Fläche auf Kosten der Nichtlinearität [BG04, Kol00]. Darüber hinaus ist der Ausgang von vielen Sensoren ein Stromausgang. Daher werden die Bausteine des Strombetriebs in dieser Arbeit in Betracht gezogen, um ein programmierbares, selbstkalibrierendes vielseitige verwendbares System zu schaffen.

Eine prinzipielle Entwurfsweise und ein erster Entwurfsablauf wird vorgeschlagen, um funktionale Blöcke des benutzergezielten Entwurfs zu extrahieren, geeignete Topologien für die Blöcke zu wählen, den Entwurf auf die generische OC Sensorelektronik abzubilden, diese hierarchisch zu optimieren und dann letztlich dynamisch fortlaufend zu justieren, um weitere, auftretende Abweichungen zu kompensieren. Der Schwerpunkt der Arbeit liegt auf der hierarchischen Optimierung, deren Ergebnisbewertungen unter den Randbedingungen eingebetteter Applikationen, und der dynamischen, hierarchischen Justierung der betrachteten Schaltungen zur Laufzeit.

Die hierarchische Optimierung wird zur Reduktion des Suchraums und der Erreichung eines vorhersagbaren Verhaltens zunächst die Optimierung der Basisbausteine, z.B. Operationsverstärker, vorgenommen. Diese werden dann in die funktionalen Blöcke eingesetzt und die Optimierung des jeweiligen funktionalen mit den passiven Komponenten, z.B. eines Gegenkopplungsnetzwerks vorgenommen. Die Optimierung erfolgt durch evolutionäre Optimierungstechniken, wie der sogenannten Partikel-Schwarm-Optimierung. Zum Beispiel, der Operationsverstärker kann mit passiven Bauelementen in üblicher Form zu einem Filter oder einem Instrumentierungsverstärker kombiniert werden, wie in Abbildung 7.3 dargestellt. Zuerst werden für den letzteren Fall die Operationsverstärker optimiert, und dann der Instrumentierungsverstärker.

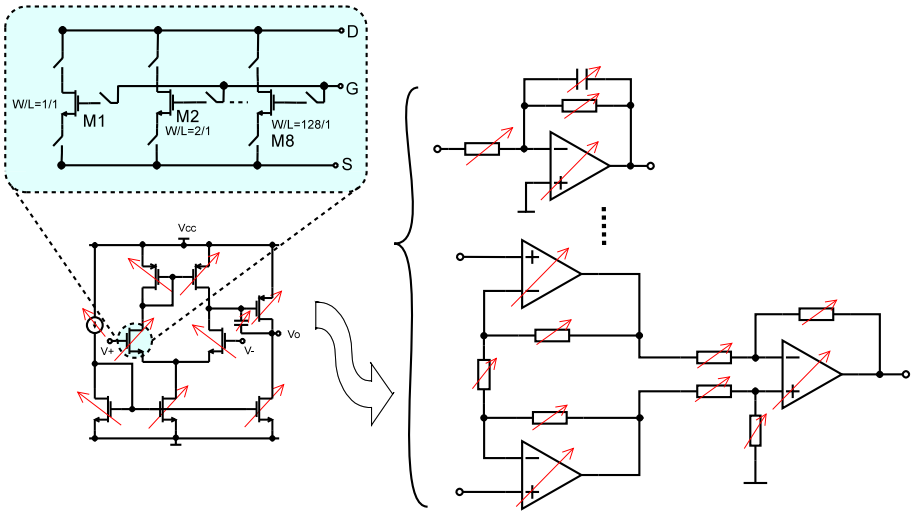


Figure 7.3: Die hierarchische Optimierung generischer Sensorelektronik, Zusammensetzung zu funktionalen Blöcken, wie Instrumentierungsverstärker aus rekonfigurierbaren Operationsverstärkern und passiven Komponenten.

Ein rekonfigurierbarer Instrumentierungserstärker wurde bereits von Lakshmanan u.a. entworfen und befindet sich derzeit in der Herstellung.

Die dynamische Justierung erfolgt durch Justierung der Hardware anhand eines geeigneten Messaufbaus zur Kompensation von Abweichungen. Dies kann entweder im Vordergrund oder im Hintergrund durchgeführt werden. Die Justierung im Hintergrund kann erreicht werden, indem zusätzliche Blockinstanzen benutzt werden, von denen ein Block im Hintergrund optimiert wird und solange

durch einen anderen nicht bzw. im vorigen Schritt kalibrierten Block ersetzt wird [JW98, TK07]. Die Blöcke, die aufgrund der Nichtverfügbarkeit von Ressourcen nicht im Hintergrund justiert werden können, werden im Vordergrund justiert, wenn das System sich im Standby-Modus befindet.

Messung der Hardware Spezifikationen mittels Standardmethoden und Geräten ist ein teurer und zeitaufwendiger Ansatz. Abhängig von der Zielspezifikation kann die Messung u.a. sehr zeitaufwendig sein oder spezielle teure Bandbreiteneinstellung erfordern. Die Messung der Kleinsignaleigenschaften der Hardware erfordert beispielsweise das Einstellen der Stimulusfrequenz, was für niedrige Frequenzen ein zeitaufwendiger Prozess und für hohe Frequenzen eine teure Bandbreiteneinstellung bildet; die Messung der transienten Eigenschaften erfordert ADCs mit hoher Geschwindigkeit, usw.

Daher wird ein neuartiger Ansatz vorgeschlagen, der die Hardwarespezifikationen in zwei Gruppen teilt [TK07]. Die erste Gruppe besteht aus den Spezifikationen, die nur schwer wegen der Kosten / Zeit Anforderungen zu messen sind, und sind weniger anfällig für solche Abweichungen wie Leerlaufverstärkung, Phasenrand , Ausgangswiderstand, usw. sind. Diese Gruppe von Spezifikationen werden extrinsisch ausgewertet. Wenn die Zielspezifikationen die Anwendungsanforderungen übererfüllen, führt der Einfluss von Abweichungen auf dieser Gruppe von Spezifikationen nicht zu einer dramatischen Veränderung der Spezifikationen. Die zweite Gruppe von Spezifikationen enthält die Spezifikationen, die empfindlich auf Abweichungen reagieren und dazu führen können, dass unmittelbare Verzerrung des Signals auftritt, während sie mit einfachen Mitteln und zu geringen Kosten zu messen sind wie Spannnungsversatz (Offset), Aussteuerbereich, Eingangsgleichtaktbereich, usw. Diese Gruppe von Spezifikationen wird intrisisch gemessen. Die Optimierungskriterien der vorliegenden Mehrzieloptimierung, in der jeder einzelne hat intrinsische und extrinsische Ziele wie in Abbildung 7.4.

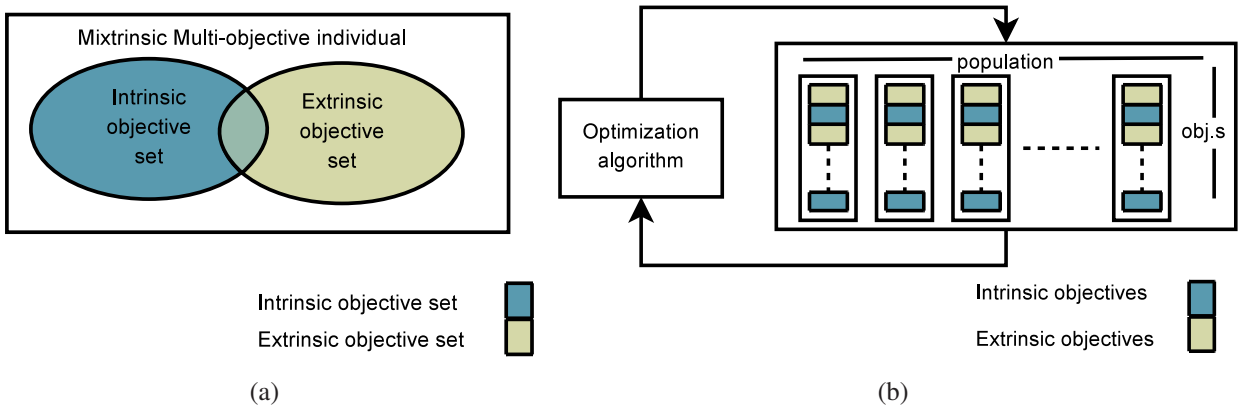


Figure 7.4: Die mixtrinsische multikriterielle Evolution a) Ein einzelnes Individuum. B) Die vollständige Population.

Die beiden Gruppen von Spezifikationen können sich überlappen, was bedeutet, dass einige der Spezifikationen intrinsisch als auch extrinsisch behandelt werden können. In diesem Fall kann sich die geforderte intrinsische und extrinsische Messung unterscheiden. Daher kann der Operationsverstärker innerhalb einer einzigen Abtastung des ADC stabil werden.

Die messbaren Umgebungsdaten, wie die Elektroniktemperatur, können gemessen und in der Bewertung der extrinsischen Spezifikation verwendet werden. Daher sind die kompletten Hardwarespezifikationen mit integrierbarer und einbettbarer Kalibrierungsschleife auch bei niedrigen Kosten optimierbar.

Die mixtrinsische multikriterielle Vorgehensweise, beruhend auf simulierten Modellen, wird in Abbildung 7.5 dargestellt. Für eine bestimmte Konfiguration simuliert die Hardware das Extrahieren der

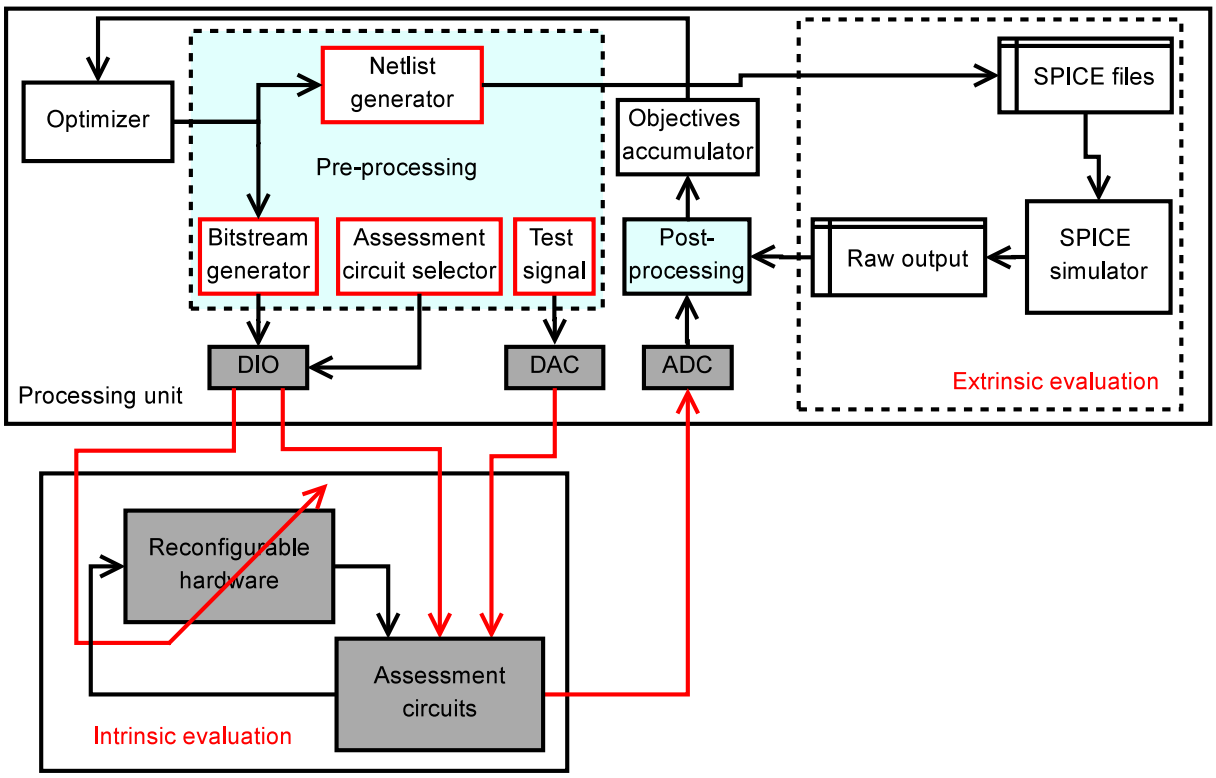


Figure 7.5: Die mixtrinsische multikriterielle Evolution Umgebung.

technischen Daten der extrinsischen Ziele, während das andere Gerät intrinsisch misst.

Um die extrinsischen Ziele zu beurteilen, werden die passenden Netzlistendateien erzeugt, und dann durch eine Simulationsumgebung wie SPICE simuliert. Anschlieend werden die Ergebnisse der Simulationen aufbereitet, um die Spezifikationen der Zielgruppenkonfiguration zu extrahieren. Abhängig von der Genauigkeit der Bewertungsschaltkreise und der Wirkung ihrer Empfindlichkeit gegenüber Rauschen, können die simulierten Modelle genauer sein als die intrinsische Messung der Spezifikationswerte.

Zum Stand der Technik beginnt die Rekonfiguration evolutionärer Elektronik nach jeder Veränderung der Umwelt praktisch bei Null, was dazu führen kann, dass die Häufigkeit und Dauer von Kalibrierungs- bzw. Justierungsvorgängen erheblich zunimmt, da ständig Vorwissen der Evolution verloren geht. Zur Erhöhung der Zuverlässigkeit und Effizienz im dynamischen Umfeld, sind in dieser Arbeit Ansätze zur schritthaltenden Optimierung betrachtet worden, die in der Lage sind, umgebungsbedingten Veränderungen während der Evolution zu erkennen und mit der Situation zurecht zu kommen, ohne jeweils bei Null nach jeder Veränderung der Umwelt anfangen zu müssen. Dieser Ansatz bietet eine erhebliches Potenzial für die Verbesserung von Konvergenzgeschwindigkeit und Lösungsgüte.

Bibliography

[ACB99] A. Hernádez Aguirre, C. Coello, and B. Buckles. A genetic programming approach to logic function synthesis by means of multiplexers. In Adrian Stoica, Jason Lohn, and Didier Keymeulen, editors, *The First NASA/DoD Workshop on Evolvable Hardware*, pages 46–53, Pasadena, California, 19-21 July 1999. Jet Propulsion Laboratory, California Institute of Technology, IEEE Computer Society.

[Aff05] Michael Affenzeller, editor. *Population Genetics and Evolutionary Computation: Theoretical and Practical Aspects*. Reihe C. Technik und Naturwissenschaften. Schriften der Johannes-Kepler Universität Linz, 2005.

[AGK94] Kurt J. Antreich, Helmut E. Graeb, and Rudolf K. Koblitz. *Advanced Yield Optimization Techniques*, volume 8 (Statistical Appro of *advcad*. Elsevier Science Publishers, Amsterdam, 1994.

[AGW93] Kurt J. Antreich, Helmut E. Graeb, and Claudia U. Wieser. Practical methods for worst-case and yield analysis of analog integrated circuits. *International Journal of High Speed Electronics and Systems (IJHSES)*, 4(3):261–282, August 1993.

[Ana] http://www.analog-insydes.de/.

[Ang98] P. J. Angeline. Using selection to improve particle swarm optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1998)*, Anchorage, Alaska, USA, 1998.

[AZC04] Arturo Hernández Aguirre, Ricardo Salem Zebulum, and Carlos A. Coello Coello. Evolutionary multiobjective design targeting a field programmable transistor array. In *Proceedings of 6th NASA / DoD Workshop on Evolvable Hardware (EH 2004)*, pages 199–, Seattle, WA, USA, June 2004. IEEE Computer Society.

[BB02a] T. M. Blackwell and Peter J. Bentley. Dynamic search with charged swarms. In *In Proceedings of the Genetic and Evolutionary Computation Conference 2002 (GECCO)*, pages 19–26, 2002.

[BB02b] Tim Blackwell and Peter J. Bentley. Improvised music with swarms. In David B. Fogel, Mohamed A. El-Sharkawi, Xin Yao, Garry Greenwood, Hitoshi Iba, Paul Marrow, and Mark Shackleton, editors, *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, pages 1462–1467. IEEE Press, 2002.

[BB04] Tim Blackwell and Jürgen Branke. Multi-swarm optimization in dynamic environments. In *Proceedings of Applications of Evolutionary Computing, EvoWorkshops 2004: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoMUSART, and EvoSTOC*, pages 489–500, Coimbra, Portugal, April 2004.

[BEB02] R. Brits, A. P. Engelbrecht, and B. Bergh. A niching particle swarm optimizer. In *In Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'02)*, volume 2, pages 692–696, Orchid Country Club, Singapore, November 2002. Nanyang Technical University, 2002.

[BG04] D. Biolek and T. Gubek. New circuit elements for current-mode signal processing. In *International Journal Elektrorevue, www.elektrorevue.cz*, May 2004.

[BH80] W.C. Black and D.A. Hodges. Time-interleaved converter arrays. In *IEEE Journal of Solid State Circuits*, pages 1024–1029, December 1980.

[Bio03] Dalibor Biolek. CDTA- building block for current-mode analog signal processing. In *Proceedings of the European Conference on Circuit Theory and Design ECCTD03 Krakow, Poland*, volume 3, pages 397–400, September 2003.

[Bla05] T. Blackwell. Particle swarms and population diversity. *Soft Comput.*, 9(11):793–802, 2005.

[BM05] J. Becker and Y. Manoli. Eine FPAA-Architektur zur rekonfigurierbaren Instantierung von zeitkontinuierlichen Analogfiltern. In *Advances in Radio Science (ARS), Kleinheubacher Berichte*, volume 3, pages 371–375, 2005.

[Bra99] Jürgen Branke. Evolutionary approaches to dynamic optimization problems - a survey. In *GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, pages 134–137, 1999.

[Bra01a] Jürgen Branke. Evolutionary approaches to dynamic optimization problems - updated survey. In *GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, pages 27–30, 2001.

[Bra01b] Jürgen Branke. *Evolutionary optimization in dynamic environments*. PhD thesis, der Fakultät für Wirtschaftswissenschaften, Universität Fridericiana zu Karlsruhe, Kluwer, 2001.

[Bra04] Jürgen Branke. Optimization in dynamic environments. GECCO Tutorial 2004, June 2004.

[Bre62] H. J. Bremermann. *Self-Organizing Systems*, chapter Optimization Through Evolution and Recombination, pages 93–106. Spartan Books, Washington DC, 1962.

[BRL+05] Faik Baskaya, Sasank Reddy, Sung Kyu Lim, Tyson Hall, and David V. Anderson. Mapping algorithm for large-scale field programmable analog array. In *ISPD '05: Proceedings of the 2005 international symposium on Physical design*, pages 152–158, New York, NY, USA, 2005. ACM Press.

[BS02] J. Branke and H. Schmeck. *Theory and Application of Evolutionary Computation: Recent Trends*, chapter Designing evolutionary algorithms for dynamic optimization problems, pages 239–262. Springer, 2002.

[BTD96] E. Bonabeau, G. Theraulaz, and J.-L Deneubourg. Quantitative study of the fixed threshold model for the regulation of division of labour in insect societies. In *Proceedings - Royal Society of London. Biological sciences*, pages 1565–1569, 1996.

[BTHM07] Joachim Becker, Stanis Trendelenburg, Fabian Henrici, and Yiannos Manoli. Synthesis of analog filters on an evolvable hardware platform using a genetic algorithm. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation (GECCO '07)*, pages 190–197, New York, NY, USA, 2007. ACM Press.

[BVB05] D. Biolek and V. V. Biolková. Cdta-c current-mode universal 2nd-order filter. In *Proceedings of the 5th WSEAS Int. Conf. on Applied Informatics and Communications, Malta*, pages 411–414, September 2005.

[CD00] A. Carlisle and G. Dozier. Adapting particle swarm optimization to dynamic environments. In *Proceedings of International Conference on Artificial Intelligence (ICAI 2000), Las Vegas, Nevada, USA*, pages 429–434, 2000.

[CD01] A. Carlisle and G. Dozier. Tracking changing extrema with particle swarm optimizer. Technical report, Auburn University Technical Report CSSE01-08, 2001.

[CD02] A. Carlisle and G. Dozier. Tracking changing extrema with adaptive particle swarm optimizer. In *Proceedings of the 5th Biannual World Automation Congress, 2002*, volume 13, pages 265–270, 2002.

[CGN04] Yun Chiu, Paul R. Gray, and Borivoje Nikolić. A 14-bit, 12-MS/s CMOS pipeline ADC with over 100-dB SFDR. *IEEE Journal of Solid-State Circuits*, 39(12):2139–2151, December 2004.

[Chr03] George Chronis. Lecture notes: Evolutionary computation; an introductory presentation, 2003.

[CHR+05] X. Cui, C. T. Hardin, R. K. Ragade, T. E. Potok, and A. S. Elmaghraby. Tracking non-stationary optimal solution by particle swarm optimizer. In *Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Networks SNPD/SAWN'05*, pages 133–138, 2005.

[CjK02] M. Clerc and j. Kennedy. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6:58–73, 2002.

[CLA02] Carlos A. Coello Coello, Erika Hernández Luna, and Arturo Hernández Aguirre. Use of particle swarm optimization to design combinational logic circuits. Technical report, EVOCINV-04-2002, Evolutionary Computation Group at CINVESTAV, Sección de Computación, Departamento de Ingenierí a Eléctrica, CINVESTAV-IPN, México 2002., October 2002.

[Cle99] M. Clerc. The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. In *Proceedings of the Congress on Evolutionary Computation*, pages 1951–1957, Washington, DC, 1999. IEEE press.

[Cob90] Helen G. Cobb. An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments. Technical Report 6760 (NLR Memorandum), Navy Center for Applied Research in Artificial Intelligence, Naval Research Laboratory, Code 5514, Washington, D.C. 20375-5320, December 1990.

[Cra85] N. L. Cramer. A representation for the adapative generation of simple sequential programs. In *International Conference on Genetic Algorithms and their Applications [ICGA85]*, 1985.

[dAdAS+04] José Franco Machado do Amaral, Jorge Luís Machado do Amaral, Cristina Costa Santini, Marco Aurélio Cavalcanti Pacheco, Ricardo Tanscheit, and Moisés H. Szwarcman. Intrinsic evolution of analog circuits on a programmable analog multiplexer array. In *Proceedings of the 4th International Conference of Computational Science - ICCS 2004*, pages 1273–1280, Kraków, Poland, June 2004.

[DAPM00] Kalyanmoy Deb, Samir Agrawal, Amrit Pratab, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-Objective Optimization: NSGA-II. In Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, J. J. Merelo, and Hans-Paul Schwefel, editors, *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 849–858, Paris, France, 2000. Springer. Lecture Notes in Computer Science No. 1917.

[dB02] Frans Van den Bergh. *An analysis of particle swarm optimizers*. PhD thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2002. Supervisor-A. P. Engelbrecht.

[dBE02] F. Van den Bergh and A. P. Engelbrecht. A new locally convergent particle swarm optimiser. In *Proceedings of 2002 IEEE International Conference on Systems, Man and Cybernetics*, October 2002.

[DD96] I. Das and J. Dennis. A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. Technical Report 96–36, Dept. Of Computational and Applied Mathematics Tech Report 96–36. Rice University, Houston, TX, 1996.

[Dev] Analog Devices. AD8403 4-channel digital potentiometer. http://www.analog.com/en/prod/0,,761_797_AD8403,00.html.

[DFLH98] K. Dyer, D. Fu, S. Lewis, and P. Hurst. Analog background calibration of a 10b 40MSample/s parallel pipelined ADC. In *Proceedings of the IEEE International Solid State Circuits Conference (ISSC98)*, pages 140,141, 1998.

[DGS03] W. Daems, G. Gielen, and W. Sansen. Simulation-based generation of posynomial performance models for the sizing of analog integrated circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22:517– 534, May 2003.

[dMHBL01] Maria del Mar Hershenson, Stephen P. Boyd, and Thomas H. Lee. Optimal design of a CMOS op-amp via geometric programming. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 20(1):1–21, 2001.

[Dor92] M. Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, 1992.

[EK95] R. C. Eberhard and J. Kennedy. New optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 1995.

[EQO+98] S. H. K. Embabi, X. Quan, N. Oki, A. Manjrekar, and E. Sánchez-Sinencio. A current-mode based field-programmable analog array for signal processing applications. *Analog Integr. Circuits Signal Process.*, 17(1-2):125–142, 1998.

[ES00] R. C. Eberhart and Y. Shi. Comparing inertia weights and constriction factors in particle swarm optimization. In *Proceedings of the 2000 Congress on Evolutionary Computation CEC00*, pages 84–88, La Jolla Marriott Hotel La Jolla, California, USA, 6-9 July 2000. IEEE Press.

[ES01] R. C. Eberhart and Y. Shi. Tracking and optimizing dynamic systems with particle swarms. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2001), Seoul, Korea*, pages 94–97, 2001.

[ES03] A. E. Eiben and J. E. Smith, editors. *Introduction to Evolutionary Computing*. Springer, 1 edition, November 2003.

[ESSA00] John Emmert, Charles Stroud, Brandon Skaggs, and Miron Abramovici. Dynamic fault tolerance in FPGAs via partial reconfiguration. In *FCCM '00: Proceedings of the 2000 IEEE Symposium on Field-Programmable Custom Computing Machines*, page 165, Washington, DC, USA, 2000. IEEE Computer Society.

[FDLH98] D. Fu, K. Dyer, S. Lewis, and P. Hurst. Digital background calibration technique for time-interleaved analog-to-digital converters. *IEEE Journal of Solid-State Circuits*, 33(12):1904–1011, December 1998.

[FF93] Carlos M. Fonseca and Peter J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Genetic Algorithms: Proceedings of the Fifth International Conference*, pages 416–423. Morgan Kaufmann, 1993.

[FF95] Carlos M. Fonseca and Peter J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, 1995.

[Fie04] Jonathan E. Fieldsend. Multi-objective particle swarm optimisation methods. Technical Report 419, Department of Computer Science, University of Exeter, March 2004.

[FOW65] L. J. Fogel, A. J. Owens, and M. J. Walsh. Artificial intelligence through a simulation of evolution, in biophysics and cybernetics systems. In *Proceedings of the Second Cybernetics Sciences Symposium, ed. by M. Maxfield, A. Callahan, and L. J. Fogel, Spartan Books, Washington*, 1965.

[FPL05] Peter J. Fleming, Robin C. Purshouse, and Robert J. Lygoe. Many-objective optimization: An engineering design perspective. In *Proceedings of the 3rd International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)*, pages 14–32, Guanajuato, Mexico, March 2005. Springer.

[FPP86] J.D. Farmer, N. Packard, and A. Perelson. The immune system, adaptation and machine learning. In *Physica D, vol. 22*, pages 187–204, 1986.

[Fra57] A. S. Fraser. Simulation of genetic systems by automatic digital computers. ii: Effects of linkage on rates under selection. *Austral. J. Biol. Sci. 10*, 1957.

[Fra00] Randy Frank. *Understanding Smart Sensors*. Artech House, Inc., Norwood, MA, USA, 2nd ed. edition, 2000.

[FS98] Stuart J. Flockton and Kevin Sheehan. Intrinsic circuit evolution using programmable analogue arrays. *Lecture Notes in Computer Science*, 1478:144–153, 1998.

[GAl] Galib: Matthew's c++ genetic algorithms library, http://lancet.mit.edu/galib-2.4/galib.html.

[Gau97] Vincent Charles Gaudet. Design of a CMOS current conveyor-based field programmable analog array. Master's thesis, Department of Electrical and computer Engineering, University of Toronto, 1997.

[GME05] Georges Gielen, Trent McConaghy, and Tom Eeckelaert. Performance space modeling for hierarchical synthesis of analog integrated circuits. In *Proceedings of the 42nd annual conference on Design automation (DAC '05)*, pages 881–886, New York, NY, USA, 2005. ACM Press.

[Goe98] Jay Goetz. High temperature electronics for sensor interface and data acquisition. In *Sensors Expo & Conference*, 1998.

[GV04] V. G. Gudise and G. K. Venayagamoorthy. FPGA placement and routing using particle swarm optimization. In *IEEE Computer society Annual Symposium on VLSI, 2004*, pages 307–308, February 2004.

[HCDV05] R. Hassan, B. Cohanim, O. L. DeWeck, and G. Venter. A comparison of particle swarm optimization and the genetic algorithm. In *AIAA-2005-1897, 1st AIAA Multidisciplinary Design Optimization Specialist Conference*, Austin, Tex, USA, April 2005.

[HHA02] Tyson S. Hall, Paul Hasler, and David V. Anderson. Field-programmable analog arrays: A floating-gate approach. In *FPL '02: Proceedings of the Reconfigurable Computing Is Going Mainstream, 12th International Conference on Field-Programmable Logic and Applications*, pages 424–433, London, UK, 2002. Springer-Verlag.

[HHE02] M. Hartmann, P. Haddow, and F. Eskelund. Evolvable hardware solutions for extreme temperature electronics. In *The 2002 NASA/DoD Conference on Evolvable Hardware*, pages 36–45, Alexandria, Virginia, 15-18 July 2002. Jet Propulsion Laboratory, California Institute of Technology, IEEE Computer Society.

[HI03] N. Higashi and H. Iba. Particle swarm optimization with gaussian mutation. In *Proceedings of the 2003 Swarm Intelligence Symposium SIS'02*, pages 72–79, 2003.

[HK05] Mohsen Hayati and Umesh Kumar. Novel variable current gain active filters using current conveyors. *Journal of Active and Passive Electronic Devices*, 1:91–96, 2005.

[HNG94] Jeffrey Horn, Nicholas Nafpliotis, and David E. Goldberg. A niched pareto genetic algorithm for multiobjective optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, volume 1, pages 82–87, Piscataway, New Jersey, 1994. IEEE Service Center.

[Hol73] J. H. Holland. Genetic algorithms and the optimal allocation of trials. *SIAM j. of Computing*, pages 88–105, 1973.

[HPTB98] Alister Hamilton, Kostis Papathanasiou, Morgan Tamplin, and Thomas Brandtner. Palmo: Field programmable analogue and mixed-signal vlsi for evolvable hardware. In *ICES '98: Proceedings of the Second International Conference on Evolvable Systems*, pages 335–344, London, UK, 1998. Springer-Verlag.

[Ins] Texas Instruments. PGA309 voltage output programmable sensor conditioner. http://focus.ti.com/docs/prod/folders/print/pga309.html.

[Int] Intersil. X90100 non-volatile electronically programmable capacitor. http://www.intersil.com/cda/deviceinfo/0,0,X90100.html.

[ITF02] Isao Imazeki, Shigetaka Takagi, and Nobuo Fujii. Low-voltage, low-power, second-generation current conveyors. *Electrical Engineering in Japan*, 138:41–48, November 2002.

[JM03] S. Janson and M. Middendorf. A hierarchical particle awarm optimizer. In *The Congress on Evolutionary Computation, 2003. CEC'03*, volume 2, pages 770 – 776, December 2003.

[JM04] Stefan Janson and Martin Middendorf. A hierarchical particle swarm optimizer for dynamic optimization problems. In Guenther R. Raidl, Stefano Cagnoni, Jurgen Branke, David W. Corne, Rolf Drechsler, Yaochu Jin, Colin Johnson, Penousal Machado, Elena Marchiori, Franz Rothlauf, George D. Smith, and Giovanni Squillero, editors, *Applications of Evolutionary Computing, EvoWorkshops2004: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoMUSART, EvoSTOC*, volume 3005 of *LNCS*, pages 511–522, Coimbra, Portugal, 5-7 April 2004. Springer Verlag.

[JM05] Stefan Janson and Martin Middendorf. A hierarchical particle swarm optimizer and its adaptive variant. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 35(6):1272–1282, 2005.

[Jon75] K. D. De Jong. *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, Departament of Computer and Communication Sciences, University of Michigan, Ann Arbor, 1975.

[JW98] J. Ingino Jr. and B. Wooley. A continuously-calibrated 10 M Sample/s 12 b 3.3 V ADC. In *Proceedings of the 1998 IEEE International Solid-State Circuits Conference. Digest of Technical Papers. 45th ISSCC 1998*, pages 144–145, February 1998.

[KÖ6] Andreas König. Status and perspective of intelligent system design automation. In *HIS '06: Proceedings of the Sixth International Conference on Hybrid Intelligent Systems*, page 2, Washington, DC, USA, 2006. IEEE Computer Society.

[KBA+97] John R. Koza, Forrest H Bennett III, David Andre, Martin A. Keane, and Frank Dunlap. Automated synthesis of analog electrical circuits by means of genetic programming. *IEEE Transactions on Evolutionary Computation*, 1(2):109–128, July 1997.

[KdW06] Il Yong Kim and Olivier de Weck. Adaptive weighted sum method for multiobjective optimization: a new method for pareto front generation. *Structural and Multidisciplinary Optimization*, 31:105–116, 2006.

[Ken97] J. Kennedy. The particle swarm: Social adaptation of knowledge. In *In Proceedings of the International Conference on Evolutionary Computation*, pages 303–308, 1997.

[Ken00] J. Kennedy. Evolutionary computation, 2000. proceedings of the 2000 congress on. In *Proceedings of the 2000 Congress on Evolutionary Computation*, pages 1507 – 1512. IEEE, 2000.

[KFFT04] Thiemo Krink, Bogdan Filipic, Gary Fogel, and Rene Thomsen. Noisy optimization problems - a particular challenge for differential evolution? In *Congress onEvolutionary Computation, 2004. CEC2004.*, volume 1, pages 332–339, June 2004.

[KFHBAK97] John R. Koza, III Forrest H. Bennett, David Andre, and Martin A. Keane. Evolution using genetic programming of a low-distortion, 96 decibel operational amplifier. In *SAC '97: Proceedings of the 1997 ACM symposium on Applied computing*, pages 207–216, New York, NY, USA, 1997. ACM Press.

[KG04] Tholom Kiely and Georges Gielen. Performance modeling of analog integrated circuits using least-squares support vector machines. In *DATE '04: Proceedings of the conference on Design, automation and test in Europe*, page 10448, Washington, DC, USA, 2004. IEEE Computer Society.

[KIAK99] J. R. KOZA, F. H BENNETT III, D. ANDRE, and M. A. KEANE. Genetic programming: Turings third way to achieve machine intelligence. In *1999 for EUROGEN workshop*, Jyvdskyld, Finland, May-June 1999.

[KLB93] A. N. Karanicolas, Hae-Seung Lee, and K. L. Barcrania. A 15-b 1-Msample/s digitally self-calibrated pipeline ADC. *IEEE Journal of Solid-State Circuits*, 28:1207–1215, December 1993.

[KLT06] A. König, S. K. Lakshmanan, and P. M. Tawdross. *International Congress Series Brain-Inspired IT II: Decision and Behavioral Choice Organized by Natural and Artificial Brains*, volume 1291, chapter Towards Organic Sensing Systems-Dynamic

reconfigurable Mixed Signal Electronics for Adaptive Sensing in Organic Computing Systems, pages 38–45. ELSEVIER, 2006.

[KM02] J. Kennedy and R. Mendes. Topological structure and particle swarm performance. In David B. Fogel, Xin Yao, Garry Greenwood, Hitoshi Iba, Paul Marrow, and Mark Shackleton, editors, *Proceedings of the Fourth Congress on Evolutionary Computation (CEC-2002)*, Honolulu, Hawaii, May 2002. IEEE Computer Society.

[Kol00] Kimmo Koli. *CMOS Current Amplifiers: Speed versus Nonlinearity*. PhD thesis, Helsinki University of Technology, Espoo, Finland, November 2000.

[Koz90] J. R. Koza. Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems. Technical Report STAN-CS-90-1314, Stanford University Computer Science Department, June 1990.

[KS06] U. Kanne and C. Sauvain. Digital flow sensors: Reaching new levels. *Medical device technology (Med. device technol.)*, 17:12–14, June 2006.

[KSB97] Sung-Ung Kwak, Bang-Sup Song, and Kantilal Bacrania. A 15 b 5 MSample/s low-spurious CMOS ADC. In *Proceedings of the 1997 IEEE International Solid-State Circuits Conference. Digest of Technical Papers. 44th ISSCC 1997*, pages 146 – 147, 445, February 1997.

[KZJS00] D. Keymeulen, R. S. Zebulum, Y. Jin, and A. Stoica. Fault-tolerant evolvable hardware using field-programmable transistor arrays. In *IEEE Transactions on Reliability*, volume 49, pages 305–316, September 2000.

[Lan05] Jo̎rg Langeheine. *Intrinsic Hardware Evolution on the Transistor Level*. PhD thesis, Electronic Vision(s) Group, Kirchhoff-Institut für Physik, Heidelberg Universität, 2005.

[Lay98] P. Layzell. Evolvable motherboard': A test platform for the research of intrinsic hardware evolution, 1998.

[LCA04] E. H. Luna, C. A. Coello Coello, and A. H. Aguirre. On the use of a population-based particle swarm optimizer to design combinational logic circuits. In *Proceedings of the 2004 NASA/DoD Conference on Evolvable Hardware.*, pages 183–190, June 2004.

[LGA99] Walter M. Lindermeir, Helmut E. Graeb, and Kurt Antreich. Analog testing by characteristic observation inference. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 18(9):1353–1368, 1999.

[LK05] S.K. Lakshmanan and A. König. A contribution to reconfigurable analog electronics by a digitally programmable sensor signal amplifier. In *Advances in Radio Science*, 2005.

[LK07] Senthil Kumar Lakshmanan and Andreas König. Towards organic sensing system-first measurement result of self-x sensor signal amplifier. In *Proceedings of 7th International Conference on Hybrid Intelligent Systems (HIS 2007)*, Kaiserslautern, Germany, September 2007. IEEE Computer Society.

[LKG91] Y.-M. Lin, B. Kim, and P .R. Gray. A 13-b 2.5-MHz self-calibrated pipelined A/D converter in 3-μm CMOS. *IEEE Journal of Solid-State Circuits*, 26:628–636, April 1991.

[LM03] Jipeng Li and Un-Ku Moon. Background calibration techniques for multistage pipelined adcs with digital redundancy. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 50(9):531–538, September 2003.

[LMS02] J. Langeheine, K. Meier, and J. Schemmel. Intrinsic evolution of quasi DC solutions for transistor level analog electronic circuits using a CMOS FPTA chip. In *The 2002 NASA/DoD Conference on Evolvable Hardware*, pages 76–85, Alexandria, Virginia, 15-18 July 2002. Jet Propulsion Laboratory, California Institute of Technology, IEEE Computer Society.

[LMS03] J. Langeheine, K. Meier, and J. Schemmel. Intrinsic evolution of analog electronic circuits using a CMOS FPTA chip. In *Proceedings of the 5th Conference on Evolutionary Methods for Designing, Optimization and Control (EUROGEN)*, September 2003.

[LMU04] P. Lakshmikanthan, S. Mulchandani, and A. Núñ (USA). Sizing analog circuits using an improved optimization-based tool. *Journal of Circuits, Signals, and Systems*, 449, 2004.

[Lø02] M. Løvbjerg. Improving particle swarm optimization by hybridization of stochastic search heuristics and self-organized criticality. Master's thesis, Department of Computer Science, University of Aarhus, Denmark, 2002.

[LRK01] Morten Løvbjerg, Thomas Kiel Rasmussen, and Thiemo Krink. Hybrid particle swarm optimizer with breeding and subpopulations. In *In Proceedings of the Genetic and Evolutionary Computation Conference 2001 (GECCO)*, volume 1, pages 469–476, 2001.

[LS92] S.-H. Lee and B.-S. Song. Digital-domain calibration of multistep analog-to-digital converters. *IEEE Journal of Solid-State Circuits*, 27:1679–1688, December 1992.

[LSF98] C. Azeredo Leme, José B. Silva, and J. E. Franca. A full 12-bit switched-current delta-sigma modulator with self-calibration. In *Proceedings of the 22nd European Solid-State Circuits Conference ESSCIRC'96*, 1998.

[LTK06] Senthil Kumar Lakshmanan, Peter Tawdross, and Andreas König. Towards organic sensing system-first measurement result of self-x sensor signal amplifier. In *Proceedings of 6th International Conference on Hybrid Intelligent Systems (HIS 2006)*, pages 62–65, Auckland, New Zealand, December 2006. IEEE Computer Society.

[Mei96] Peter Bartus Leonard Meijer. *Neural Network Applications in Device and Subcircuit Modelling for Circuit Simulation*. PhD thesis, Technische Universiteit Eindhoven,, 1996.

[Men04] Rui Mendes. *Population Topologies and Their Influence in Particle Swarm Performance*. PhD thesis, Escola de Engenharia, Universidade do Minho, May 2004.

[MG05] Trent McConaghy and Georges G. E. Gielen. Analysis of simulation-driven numerical performance modeling techniques for application to analog circuit optimization. In *In the proceedings of the IEEE International Symposium on Circuits and Systems 2005 (ISCAS 2005)*, pages 1298–1301, May 2005.

[Mis06] Sudhanshu Mishra. Some new test functions for global optimization and performance of repulsive particle swarm method. MPRA Paper 2718, University Library of Munich, Germany, August 2006. available at http://ideas.repec.org/p/pra/mprapa/2718.html.

[MKN04] Rui Mendes, James Kennedy, and José Neves. The fully informed particle swarm: Simpler, maybe better. *IEEE Trans. Evolutionary Computation*, 8(3):204–210, 2004.

[ML01] Jun Ming and Stephen H. Lewis. An 8-bit 80-Msample/s pipelined analog-to-digital. converter with background calibration. *IEEE Journal of Solid-State Circuits*, 36:1489–1497, October 2001.

[MPV02] George D. Magoulas, Vassilis P. Plagianakos, and Michael N. Vrahatis. Globally convergent algorithms with local learning rates. *IEEE Transactions in Neural Networks*, pages 774–779, 2002.

[MSB91] H. Mühlenbein, D. Schomisch, and J. Born. The Parallel Genetic Algorithm as Function Optimizer. *Parallel Computing*, 17(6-7):619–632, 1991.

[MSvdMW04] Christian Müller-Schloer, Christoph von der Malsburg, and Rolf P. Würtz. Organic computing. *Informatik Spektrum*, 27(4):332–336, 2004.

[OC92] Y. Ohba and Y. Chba, editors. *Intelligent Sensor Technology*. John Wiley & Sons, Inc., New York, NY, USA, 1992.

[Pap98] Konstandinos Papathanasiou. *Palmo: a novel pulsed based signal processing technique for programmable mixed-signal VLSI*. PhD thesis, Dept. of Elecrical Engineering, University of Edinburgh, Scotland, 1998.

[PB06] Srinivas Pasupuleti and Roberto Battiti. The gregarious particle swarm optimizer (g-pso). In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 67–74, New York, NY, USA, 2006. ACM Press.

[PF98] K. Prakobwaitayakit and N. Fujii. Circuit topology selection based on neural networks. In *Proceedings of the 1998 IEEE Asia-Pacific Conference on Circuits and Systems (APCCAS 1998)*, pages 387 – 390, November 1998.

[PLF+99] J. Pollack, H. Lipson, P. Funes, S. Ficici, and G. Hornby. Coevolutionary robotics. In Adrian Stoica, Jason Lohn, and Didier Keymeulen, editors, *The First NASA/DoD Workshop on Evolvable Hardware*, pages 208–216, Pasadena, California, 19-21 July 1999. Jet Propulsion Laboratory, California Institute of Technology, IEEE Computer Society.

[PV02] K. E. Parsopoulos and M. N. Vrahatis. Particle swarm optimization method in multiobjective problems. In *SAC '02: Proceedings of the 2002 ACM symposium on Applied computing*, pages 603–607, New York, NY, USA, 2002. ACM Press.

[PV04] K.E. Parsopoulos and M.N. Vrahatis. A unified particle swarm optimization scheme. In *Lecture Series on Computer and Computational Sciences, Vol. 1, Proceedings of the International Conference on Computational Methods in Sciences and Engineering (ICCMSE 2004), VSP International Science Publishers, Zeist, The Netherlands*, pages 868–873, 2004.

[PV05] Konstantinos E. Parsopoulos and Michael N. Vrahatis. Unified particle swarm optimization in dynamic environments. In *Proceedings of Applications of Evolutionary Computing, EvoWorkshops 2005: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoMUSART, and EvoSTOC*, pages 590–599, Lausanne, Switzerland, March-April 2005. Springer.

[PWSS02] B. Pankiewicz, M. Wojcikowski, S. Szczepanski, and Y. Sun. A field programmable analog array for cmos continuous-time ota-c filter applications. *IEEE journal of solid state circuits*, 27(2):125–136, February 2002.

[Rec73a] I. Rechenberg. *Evolutionsstrategie-Optimierum technischer Systeme nach Prinzipien der biologischen Evolution*. PhD thesis, Stuttgart-Bad Cannstatt: Frommann-Holzboog, 1973.

[Rec73b] I. Rechenberg. *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*. Frommen-Hozlboog Verlag, Stuttgard, 1973.

[RHW04] Asanga Ratnaweera, Saman K. Halgamuge, and Harry C. Watson. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Trans. Evolutionary Computation*, 8(3):240–255, 2004.

[Ros60] H. H. Rosenbrock. An automatic method for finding the greatest or least value of a function. *Computer J. 3*, pages 175–184, 1960.

[RRS+04] Seung-Tak Ryu, Sourja Ray, Bang-Sup Song, Gyu-Hyeong Cho, and Kanti Bacrania. A 14-b linear capacitor self-trimming pipelined ADC. *IEEE Journal of Solid-State Circuits*, 39(11):2046–2051, November 2004.

[RV02] J. Riget and J. Vesterstroem. A diversity-guided particle swarm optimizer - the arpso. Technical report, Department of Computer Science, University of Aarhus, 2002.

[SAKF06] Evangelos F. Stefatos, Tughrul Arslan, Didier Keymeulen, and Ian Ferguson. Towards the integration of drive control loop electronics of the jpl/boeing gyroscope within an autonomous robust custom-reconfigurable platform. In *Proceedings of First NASA/ESA Conference on Adaptive Hardware and Systems (AHS 2006)*, pages 207–214, Istanbul, Turkey, June 2006. IEEE Computer Society.

[SD94] N. Srinivas and Kalyanmoy Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

[SD01] Eduardo D. V. Simoes and Keith R. Dimond. Embedding a distributed evolutionary system into a population of autonomous mobile robots. In *Proceedings of the 2001 IEEE International Conference on Systems, Man, and Cybernetics*, volume 2, pages 1069–1074, 2001.

[SE98a] Y. Shi and R. C. Eberhard. A modified particle swarm optimizer. In *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation, 69-73. Poscataway, NJ: IEEE Press*, May 1998.

[SE98b] Y. Shi and R. C. Eberhard. Parameter selection in particle swarm optimization. In *Proceedings of the 1998 Annual Conference on Evolutionary Computation*, March 1998.

[SE99] Y. Shi and R. C. Eberhart. Empirical study of particle swarm optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1999), Piscataway, NJ.*, pages 1945–1950, 1999.

[SE01] Y. Shi and R. C. Eberhart. Fuzzy adaptive particle swarm optimization. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, pages 101–106, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea, 27-30 May 2001. IEEE Press.

[Sen84] R. Senani. Floating ideal FDNR using only two current conveyors. *Electronics Letters, IEE (UK)*, 20(5):205–206, March 1984.

[S.F80] S.F.Smith. *A Learning System Based on Genetic Adaptive Algorithms.* PhD thesis, Univ. of Pittsburgh, December 1980.

[Sil07] Z. K. Silagadze. Finding two-dimensional peaks. *Physics of Particles and Nuclei Letters*, 4(1), February 2007.

[SKT+99] A. Stoica, D. Keymeulen, R. Tawel, C. Salazar-Lazaro, and W. Li. Evolutionary experiments with a fine-grained reconfigurable architecture for analog and digital CMOS circuits. In Adrian Stoica, Jason Lohn, and Didier Keymeulen, editors, *The First NASA/DoD Workshop on Evolvable Hardware*, pages 76–84, Pasadena, California, 19-21 July 1999. Jet Propulsion Laboratory, California Institute of Technology, IEEE Computer Society.

[SKZ+00] A. Stoica, D. Keymeulen, R. Zebulum, A. Thakoor, T. Daud, G. Klimeck, Y. Jin, R. Tawel, and V. Duong. Evolution of analog circuits on field programmable transistor arrays. In *The Second NASA/DoD workshop on Evolvable Hardware*, pages 99–108, Palo Alto, California, 13-15 July 2000. IEEE Computer Society.

[SKZ01] A. Stoica, D. Keymeulen, and R. Zebulum. Evolvable hardware solutions for extreme temperature electronics. In *EH '01: Proceedings of the The 3rd NASA/DoD Workshop on Evolvable Hardware*, pages 93–97, Washington, DC, USA, 2001. IEEE Computer Society.

[SKZ+02] A. Stoica, D. Keymeulen, R.S. Zebulum, M.I. Ferguson, and X. Guo. On two new trends in evolvable hardware: Employment of HDL-based structuring. and design of multi-functional circuits. In Adrian Stoica, Jason Lohn, Rich Katz, Didier Keymeulen, and Ricardo Salem Zebulum, editors, *The 2002 NASA/DoD Conference on Evolvable Hardware*, pages 56–59, Alexadria, Virginia, 15-18 July 2002. Jet Propulsion Laboratory, California Institute of Technology, IEEE Computer Society.

[SSB97] T. Shu, B. Song, and K. Bacrania. A 13-b, 10-Msample/s ADC digitally calibrated with oversampling delta-sigma converter. *IEEE Journal of Solid-State Circuits*, 32:1866–1875, December 1997.

[ST88] Bang-Sup Song and M.F. Tompsett. A 12b 1MHz capacitor error averaging pipelined A/D converter. In *Proceedings of the 1988 IEEE International Solid-State Circuits Conference. Digest of Technical Papers.ISSCC 1988*, pages 226–227, February 1988.

[Sug99] P. N. Suganthan. Particle swarm optimizer with neighborhood operator. In *Proc. IEEE Int. Congr. Evolutionary Computation*, volume 3, pages 1958–1962, 1999.

[SZF+02] A. Stoica, R. S. Zebulum, M. I. Ferguson, D. Keymeulen, V. Duong, and X. Guo. Evolving circuits in seconds: Experiments with a stand-alone board-level evolvable system. In *The 2002 NASA/DoD Conference on Evolvable Hardware*, pages 67–75, Alexandria, Virginia, 12-14 July 2002. Jet Propulsion Laboratory, California Institute of Technology, IEEE Computer Society.

[SZK00] Adrian Stoica, Ricardo S. Zebulum, and Didier Keymeulen. Mixtrinsic evolution. In *ICES '00: Proceedings of the Third International Conference on Evolvable Systems*, pages 208–217, London, UK, 2000. Springer-Verlag.

[SZK01] A. Stoica, R. Zebulum, and D. Keymeulen. Progress and challenges in building evolvable devices. In *The Third NASA/DoD workshop on Evolvable Hardware*, pages 33–35, Long Beach, California, 12-14 July 2001. IEEE Computer Society.

[Tec] Microbridge Technologies. Rejustor primer. http://www.mbridgetech.com/pdfs/MB-APP01-Rejustor-Primer-AN.pdf.

[Tho95] A. Thompson. Evolving electronic robot controllers that exploit hardware resources. In F. Morán, A. Moreno, J. J. Merelo, and P. Chacon, editors, *Advances in Artificial Life: Proc. 3rd Eur. Conf. on Artificial Life (ECAL95)*, volume 929 of *LNAI*, pages 640–656. Springer-Verlag, 1995.

[Tho96] A. Thompson. An evolved circuit, intrinsic in silicon, entwined with physics. In Tetsuya Higuchi, Masaya Iwata, and L. Weixin, editors, *Proc. 1st Int. Conf. on Evolvable Systems (ICES'96)*, volume 1259 of *LNCS*, pages 390–405. Springer-Verlag, 1996.

[Tho97a] A. Thompson. Evolving inherently fault-tolerant systems. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 211:365–371, 1997.

[Tho97b] Adrian Thompson. Artificial evolution in the physical world. In T. Gomi, editor, *Evolutionary Robotics: From intelligent robots to artificial life (ER'97)*, pages 101–125. AAI Books, 1997.

[Tho02] A. Thompson. Notes on design through artificial evolution: Opportunities and algorithms. In I. C. Parmee, editor, *Adaptive computing in design and manufacture V*, pages 17–26. Springer-Verlag, 2002.

[TK05a] P. Tawdross and A. König. Dynamic reconfiguration algorithm for field programmable analog scalable device array (FPADA) with fixed topology. In *Proceedings of the 10th Online World Conference on Soft Computing in Industrial Applications (WSC10)*. Springer, November 2005.

[TK05b] Peter Tawdross and Andreas König. Investigation of particle swarm optimization for dynamic reconfiguration of field-programmable analog circuits. In *Proceedings of 5th International Conference on Hybrid Intelligent Systems (HIS 2005)*, pages 259–264, Rio de Janeiro, Brazil, November 2005. IEEE Computer Society.

[TK06a] P. Tawdross and A. König. An efficient algorithm for reconfiguring a reconfigurable analog hardware to recover the effect of industrial tolerance and extrema in dynamic environment. In *In Workshop Analog Integrated Circuits, TU Kaiserslautern, March 13-14, 2006*, TU Kaiserslautern, March 2006. Springer-Verlag.

[TK06b] P. Tawdross and A. König. Particle swarm optimization for reconfigurable sensor electronics - case study: 3 bit flash ADC. In *Workshop on Intelligent Solutions in Embedded Systems (WISES'06)*, Vienna University of Technology, Austria, June 2006.

[TK06c] P. Tawdross and A. König. *Swarm Intelligent Systems*, chapter Comparative Investigation of PSO vs. GA for Dynamically Reconfigurable Sensor Electronics, pages 67–81. NOVA, 2006.

[TK06d] Peter Tawdross and Andreas König. Local parameters particle swarm optimization. In *Proceedings of the 6th International Conference on Hybrid Intelligent Systems (HIS'06)*, page 52, Auckland, New Zealand, December 2006. IEEE Computer Society.

[TK07] Peter Tawdross and Andreas König. Mixtrinsic multi-objective reconfiguration of evolvable sensor electronics. In *Proceedings of Second NASA/ESA Conference on Adaptive Hardware and Systems (AHS 2007)*, pages 51–57, Edinburgh, Scotland, United Kingdom, August 2007. IEEE Computer Society.

[TL99] A. Thompson and P. Layzell. Analysis of unconventional evolved electronics. *Communications of the ACM*, 42(4):71–79, April 1999.

[TLH90] C. Tomazou, F.J. Lidgey, and D.G. Haigh. *Analogue IC design: the current-mode approach*. Peter Peregrinus Ltd, 1990.

[TLK05] P. Tawdross, S. Lakshmanan, and A. König. Konzept und erste Ergebnisse zu dynamisch rekonfigurierbarer Sensorelektronik für die Mess- und Automatisierungstechnik. In *In proceedings of der Arbeitkreises der Hochschullehrer für Messtechnik e.V. (AHMT 2005)*, September 2005.

[TLK06] Peter Tawdross, Senthil K. Lakshmanan, and Andreas Konig. Intrinsic evolution of predictable behavior evolvable hardware in dynamic environment. In *Sixth International Conference on Hybrid Intelligent Systems (HIS'06)*, pages 60–63, December 2006.

[TLMS05] Martin Trefzer, Jörg Langeheine, Karlheinz Meier, and Johannes Schemmel. Operational amplifiers: An example for multi-objective optimization on an analog evolvable hardware platform. In *Proceedings of Evolvable Systems: From Biology to Hardware, 6th International Conference (ICES 2005)*, pages 86–97, Sitges, Spain, September 2005. Springer.

[TLSM04] Martin Trefzer, Jörg Langeheine, Johannes Schemmel, and Karlheinz Meier. New genetic operators to facilitate understanding of evolved transistor circuits. In *6th NASA / DoD Workshop on Evolvable Hardware (EH 2004)*, pages 217–224, Seattle, WA, USA, June 2004.

[TP89] A. Törn and A. Pilinskas. Global optimization. In *Lecture Notes in Computer Science 350*. Springer-Verlag, 1989. Berlin Heidelberg.

[Tre03] Ioan Cristian Trelea. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Inf. Process. Lett.*, 85(6):317–325, March 2003.

[Tre06] Martin Albrecht Trefzer. *Intrinsic Hardware Evolution on the Transistor Level*. PhD thesis, Electronic Vision(s) Group, Kirchhoff-Institut für Physik, Heidelberg Universität, 2006.

[TS00] C. Teuscher and E. Sanchez. A revival of Turing's forgotten connectionist ideas: Exploring unorganized machines. In R. M. French and J. P. Sougné, editors, *Connectionist Models of Learning, Development and Evolution. Proceedings of the 6th Neural Computation and Psychology Workshop, NCPW6*, Perspectives in Neural Computing, pages 153–162, Liège, Belgium, September 2000. Springer-Verlag, London.

[TW00] A. Thompson and C. Wasshuber. Evolutionary design of single electron systems. In Jason Lohn, Adrian Stoica, and Didier Keymeulen, editors, *The Second NASA/DoD workshop on Evolvable Hardware*, pages 109–116, Palo Alto, California, 13-15 July 2000. Jet Propulsion Laboratory, California Institute of Technology, IEEE Computer Society.

[TZ89] A. Törn and A. Zilinskas. Global Optimization. *Lecture Notes in Computer Science*, 350, 1989.

[vdBE01] F. van den Bergh and A. P. Engelbrecht. Effects of swarm size on cooperative particle swarm optimisers. In Lee Spector, Erik D. Goodman, Annie Wu, W. B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 892–899, San Francisco, California, USA, 7-11 2001. Morgan Kaufmann.

[vdM04] Christoph von der Malsburg. Vision as an exercise in organic computing. In *INFORMATIK 2004 - Informatik verbindet, Band 2, Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), Ulm*, volume 51 of *LNI*, pages 631–635. GI, September 2004.

[VJF97] Frank Vavak, Ken Jukes, and Terence C. Fogarty. Adaptive combustion balancing in multiple burner boiler using a genetic algorithm with variable range of local search. In *Proceedings of the 7th International Conference on Genetic Algorithms*, pages 719–726, East Lansing, MI, USA, July 1997. Morgan Kaufmann.

[VRK02] Jakob S. Vesterstrøm, Jacques Riget, and Thiemo Krink. Division of labor in particle swarm opimisation. In *Proceedings of the 2002 Congress on Evolutionary Computation CEC '02.*, 2002.

[VSS03] G. Venter and J. Sobieszczanski-Sobieski. Particle swarm optimization. In *AIAA Journal, 41(8), pp.*, pages 1583–1589, 2003.

[VT03] Knut Arne Vinger and Jim Torresen. Implementing evolution of FIR-filters efficiently in an FPGA. In *Proceedings of 5th NASA / DoD Workshop on Evolvable Hardware (EH 2003)*, pages 26–32, Chicago, IL, USA, July 2003. IEEE Computer Society.

[WHL04] X. Wang, P.J. Hurst, and S.H. Lewis. A 12-bit 20-Msample/s pipelined analog-to-digital converter with nested digital background calibration. In *IEEE Journal of Solid-State Circuits*, volume 39, pages 1799–1808, November 2004.

[Wil90] B. Wilson. Recent developments in current conveyors and current-mode circuits. In *Circuits, Devices and Systems, IEE Proceedings G*, volume 137, pages 63–77, April 1990.

[WV02] Haibo Wang and Sarma B. K. Vrudhula. Behavioral synthesis of field programmable analog array circuits. *ACM Trans. Des. Autom. Electron. Syst.*, 7(4):563–604, 2002.

[XZY02] Xiao-Feng Xie, Wen-Jun Zhang, and Zhi-Lian Yang. A dissipative particle swarm optimization. In *Congress on Evolutionary Computation (CEC)*, volume 2, pages 1456–1461, 2002.

[Yao99] Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87:1423–1447, September 1999.

[Yoo03] Jincheol Yoo. *A TIQ Based CMOS Flash A/D Converter for System-on-Chip Applications*. PhD thesis, The Pennsylvania State University, Department of Computer Science and Engineering, May 2003.

[ZGK+04] R. S. Zebulumand, Xin Guo, D. Keymeulen, M. I. Ferguson, Vu Duong, and A. Stoica. High temperature experiments using programmable transistor array. In *Aerospace Conference, 2004. Proceedings. 2004 IEEE*, volume 4, pages 2437–2448, March 2004.

[Zit99] Eckart Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, ETH Zurich, December 1999.

[ZMZQ03a] Y. Zheng, L. Ma, L. Zhang, and J. Qian. Empirical study of particle swarm optimizer with an increasing inertia weight. In *Proceedings of IEEE Congress on Evolutionary Computation 2003 (CEC 2003), Canbella, Australia*, pages 221–226, 2003.

[ZMZQ03b] Y. Zheng, L. Ma, L. Zhang, and J. Qian. On the convergence analysis and parameter selection in particle swarm optimization. In *Proceedings of International Conference on Machine Learning and Cybernetics 2003*, pages 1802–1807, 2003.

[ZPV98] R. S. Zebulum, M. A. Pacheco, and M. Vellasco. A multi-objective optimisation methodology applied to the synthesis of low-power operational amplifiers. In Ivan Jorge Cheuri and Carlos Alberto dos Reis Filho, editors, *Proceedings of the XIII International Conference in Microelectronics and Packaging*, volume 1, pages 264–271, Curitiba, Brazil, 1998.

[ZPV99] R. Zebulum, M. Pacheco, and M. Vellasco. Artificial evolution of active filters: A case study. In Adrian Stoica, Jason Lohn, and Didier Keymeulen, editors, *The First NASA/DoD Workshop on Evolvable Hardware*, pages 66–75, Pasadena, California, 19-21 July 1999. Jet Propulsion Laboratory, California Institute of Technology, IEEE Computer Society.

[ZPV02] R. S. Zebulum, M. A. C. Pacheco, and M. M. B. Vellasco, editors. *Evolutionary Electronics. Automatic Design of Electronic Circuits and Systems by Genetic Algorithm*. CRC Press, 2002.

[ZSK00] Ricardo S. Zebulum, Adrian Stoica, and Didier Keymeulen. A flexible model of a cmos field programmable transistor array targeted for hardware evolution. In *ICES '00: Proceedings of the Third International Conference on Evolvable Systems*, pages 274–283, London, UK, 2000. Springer-Verlag.

[ZVP01] Ricardo Salem Zebulum, Marley Maria Bernard Vellasco, and Marco Aurelio Pacheco. *Evolutionary Electronics: Automatic Design of Electronic Circuits and Systems by Genetic Algorithms*. CRC Press, Inc., Boca Raton, FL, USA, 2001.

List of Figures

1.1 The hardware standard specifications are optimized as a multi-objective criterion . . 4

1.2 Conceptual block diagram of the target hardware. 4

2.1 Block-diagram of a simplified modern sensor system. 7

2.2 Phases of sensor electronics deviations. 9

2.3 Performance of various ADC architectures [Yoo03]. a)ADCs resolution. b) ADCs speed. 10

2.4 Simplified diagram of digital background gain calibration of interleaved ADCs [FDLH98]. 12

2.5 Digital background offset calibration of interleaved ADCs [FDLH98]. 13

2.6 Digital domain calibration multi-step flash ADC in [LS92]. 14

2.7 Architecture of 1.5 bits/stage pipelined ADC. 14

2.8 Digital background self-calibration of pipelined ADC based on employing redundant ADC. 16

2.9 Second generation current conveyor symbol. 17

2.10 CCII based floating FDNR [Sen84]. 17

2.11 CCII amplifier [Wil90]. 17

2.12 A low-voltage, low-power CCII implementation [ITF02]. 18

2.13 DC characteristics of the CCII proposed in [ITF02]. a) Current transfer characteristics. b) The V_X vs. I_X when $V_Y = 1.65V$ 18

2.14 AC characteristics of the CCII in [ITF02] 19

2.15 CDTA block a) Its symbol. b) Its implementation using CCII+ and MOTA. 19

2.16 CDTA operating as an amplifier. 20

2.17 a) The flow graph of the CDTA. b) The flow graph of second order LP filter using CDTA [Bio03]. c) The implementation of the filter in figure 2.17(b) using a single CDTA element. 20

2.18 Second-order KHN filter using CDTA [BVB05]. 20

2.19 Generic organic-computing sensor electronic system. 23

3.1 Optimization techniques' taxonomy [Aff05]. 26

3.2 Computation intelligence taxonomy [Chr03]. 26

3.3 Example of one point crossover. a) Parents. b) Offspring. 29

3.4 Example of two point crossover. a) Parents. b) Offspring. 29

3.5 Example of arithmetic crossover with $\alpha = 0.5$. a) Parents. b) Offspring. 30

3.6 Example of Genetic programming syntax tree. 31

3.7 Two selected parents for crossover. 32

3.8 The offspring after crossing over of the parents in figure 3.7. 32

3.9 Selected parent for identical crossover. 33

3.10 The offspring after crossover of the identical parents in figure 3.9. 33

3.11 a) An offspring before applying mutation. b) The offspring in figure 3.11(a) after mutation. 34

3.12 gbest model; each particle is aware of all the population. 36

3.13 lbest model; a) each particle has two neighbor. b) each particle has six neighbors. . . 36

3.14 Schema of HPSO with $h = 4$ and $d = 2$. 38

3.15 Sphere function in 2D. 47

3.16 Two-dimensions ellipsoid function. 48

3.17 Schaffer function. a) The whole search range. b) Zoomed to the range $x_d \in [-20, 20]$. 49

3.18 Rastrigin function in 2D. 49

3.19 Rosenbrock function in 2D. 50

3.20 Griewangk function in 2D. a) The search range $\mathbf{x} \in [-100, 100]$. b) Zoomed to the search range $x_d \in [-10, 10]$. 51

3.21 Sphere function. a) The convergence curve of the mean value. b) Standard deviation curve. 51

3.22 Axis Parallel Hyper ellipsoid function. a) The convergence curve of the mean value. b) Standard deviation curve. 52

3.23 Generalized Schaffer function. a) The convergence curve of the mean value. b) Standard deviation curve. 52

3.24 Rastrigin function. a) The convergence curve of the mean value. b) Standard deviation curve. 53

3.25 Rosenbrock function. a) The convergence curve of the mean value. b) Standard deviation curve. 53

3.26 Griewangk function. a) The convergence curve of the mean value. b) Standard deviation curve. 54

3.27 The convergence curve of the sphere function in dynamic environment. a) The whole curve. b) The mean value of every 20 iterations. 54

3.28 The convergence curve of the Axis parallel hyper ellipsoid function in dynamic environment. a) The whole curve. b) The mean value of every 20 iterations. 55

3.29 The convergence curve of the Schaffer function in dynamic environment. a) The whole curve. b) The mean value of every 20 iterations. 55

3.30 The convergence curve of the Rastrigin function in dynamic environment. a) The whole curve. b) The mean value of every 20 iterations. 56

3.31 The convergence curve of the Rosenbrock function in dynamic environment. a) The whole curve. b) The mean value of every 20 iterations. 56

3.32 The convergence curve of the Griewangk function in dynamic environment. a) The whole curve. b) The mean value of every 20 iterations. 57

3.33 The standard deviation curve of the sphere function in dynamic environment. a) The whole curve. b) The mean value of every 20 iterations. 57

3.34 The standard deviation curve of the Axis parallel hyper ellipsoid function in dynamic environment. a) The whole curve. b) The mean value of every 20 iterations. 58

3.35 The standard deviation curve of the Schaffer function in dynamic environment. a) The whole curve. b) The mean value of every 20 iterations. 58

3.36 The standard deviation curve of the Rastrigin function in dynamic environment. a) The whole curve. b) The mean value of every 20 iterations. 59

3.37 The standard deviation curve of the Rosenbrock function in dynamic environment. a) The whole curve. b) The mean value of every 20 iterations. 59

3.38 The standard deviation curve of the Griewangk function in dynamic environment. a) The whole curve. b) The mean value of every 20 iterations. 60

3.39 Illustrative example of Pareto for a maximization problem of f_1 and f_2. 61

4.1 Evolvable hardware taxonomy 64

4.2 The symbol of Palmo cell 65

4.3 The CAB of the OTA-C based FPAA [PWSS02]. 66

4.4 The CAB of the OTA-C based FPAA [BM05]. 67

4.5 The PAMA architecture. 68

4.6 The architecture of the evolvable motherboard [Lay98]. 68

4.7 The JPL FPTA [SKT+99]. 69

4.8 JPL FPTA2 [SZF+02] a) Structure of one cell. b) The FPTA2 chip consists of 64 cells. 70

4.9 A simplified structure of the Heidelberg FPTA [Lan05]. 71

4.10 The structure of a single programmable transistor module of Heidelberg FPTA [Lan05]. 72

4.11 The initial circuit in designing an operational amplifier with GP. 73

4.12 Example of GA operations in circuits. a) Component flipping. b) Component dividing. 73

4.13 The simulated hardware used by Coello to synthesis digital combinational circuits, extract from [CLA02]. 76

4.14 An extrinsically evolved AND gate [SZK00]. 77

4.15 An operational amplifier implemented on Heidelberg FPTA [TLMS05]. 78

4.16 FIR filter. 79

4.17 An intrinsically evolved circuit [SZK00]. 80

4.18 An evolved logarithmic amplifier [dAdAS+04]. 81

4.19 The population in mixtrinsic evolution [SZK00]. 82

4.20 An evolved circuit mixtrinsically [SZK00]. 82

5.1 Block diagram of the reconfigurable generic sensor system. 86

5.2 The complete aspired target system. 87

5.3 A programmable block that can be programmed as OTA or as operational amplifier. . 87

5.4 Extrinsic reconfiguration environment. 88

5.5 The intrinsic evolution environment. 90

5.6 The mixtrinsic multi-objective evolution a) A single individual. b) The complete population. 91

5.7 The mixtrinsic evolution. 92

5.8 Determining the hardware specifications by formal models. 93

5.9 Determining the hardware specifications by a hybrid simulated/formal model. 94

5.10 Determining an operational amplifier specifications by first simulating the biasing current at a given temperature, compute the transconductance of all the transistors, and the poles of the amplifier, then employing lean models to calculate its specifications. 94

5.11 Estimation models training. 95

5.12 Neural network as an estimation model. 95

5.13 A neural network specification estimation model that is partitioned smaller models by employing a priori knowledge. 96

5.14 Extracting the hardware specifications by a hybrid estimation/formal model. 97

5.15 A programmable NMOS transistor [LK05] of the FPMA. 97

5.16 A programmable NMOS transistor [LK07] of the FPMA2. 98

5.17 The proposed design flow for generic organic-computing sensor electronics. 99

5.18 The hierarchical optimization of evolvable hardware, building functional block level hardware such as filter and instrumental amplifier from programmable operational amplifiers. 100

5.19 Building CDTA from CCII and MOTA. 101

5.20 Outlines of background calibration of sensor electronics hardware. 101

5.21 Synchronizing the blocks swapping with the sample and hold of the ADC. 102

6.1 The proposed design flow from chapter 5 for the aspired generic organic-computing sensor electronics, the contributed blocks are marked in gray. 104

6.2 Block diagram of the implemented reconfiguration environment. 105

6.3 A simplified block diagram of the intrinsic evolution prototype. 105

6.4 Photograph of the target embedded system. 106

6.5 The evolvable chip in [LK05]. a) Reconfigurable Miller module schematic. b) Reconfigurable Miller module layout. 107

6.6 The schematic of the programmable folded-cascode operational amplifier module in [LK05]. 107

6.7 The intrinsic measurement circuits. a) Measuring the input voltage offset, slew rate, settling time, and CMR. b) Measuring the output swing voltage. c) Measuring the current by the voltage regulator circuit. 108

6.8 Photos of the experimental hardware setup. a) The assessment circuits. b) Heating the chip by soldering machine for the intrinsic dynamic environment experiment. . . 109

6.9 The convergence curve of the extrinsic evolution of the operational amplifier in [LK05] for the requirements in table 6.1. 112

6.10 The standard deviation curve of the extrinsic evolution of the operational amplifier in [LK05] for the requirements in table 6.1. 113

6.11 The convergence curve of the extrinsic evolution of the operational amplifier in [LK05] for the requirements in table 6.3. 114

6.12 The standard deviation curve of the extrinsic evolution of the operational amplifier in [LK05] for the requirements in table 6.3. 115

6.13 The convergence curve of the extrinsic evolution of the operational amplifier in [LK05] for the requirements in table 6.5. 116

6.14 The standard deviation of the extrinsic evolution of the operational amplifier in [LK05] for the requirements in table 6.5. 117

6.15 AC characteristics curve of the amplifier with the configuration in table 6.6. 118

6.16 Step response of the amplifier with the configuration in table 6.6. 119

6.17 The convergence curve of the extrinsic evolution of the operational amplifier in dynamic environment. 120

6.18 The standard deviation curve of the extrinsic evolution of the operational amplifier in dynamic environment. 121

6.19 The achieved specifications of the evolution of operational amplifier in dynamic environment. a) Open-loop gain. b) Bandwidth. 121

6.20 The achieved specifications of the evolution of operational amplifier in dynamic environment. a) CMR. b) Output swing voltage. 122

6.21 The achieved specifications of the evolution of operational amplifier in dynamic environment. a) CMRR. b) PSRR. 122

6.22 The achieved specifications of the evolution of operational amplifier in dynamic environment. a) Rising settling time. b) Falling settling time. 123

6.23 The achieved specifications of the evolution of operational amplifier in dynamic environment. a) Rising slew rate. b) Falling slew rate. 123

6.24 The achieved specifications of the evolution of operational amplifier in dynamic environment. a) Offset. b) Output resistance. 124

6.25 The achieved specifications of the evolution of operational amplifier in dynamic environment. a) Phase margin. b) Quiescent power consumption. 124

6.26 The convergence curve of the intrinsic evolution in the static environment. 125

6.27 The standard deviation curve of the intrinsic evolution in the static environment. . . . 125

6.28 Measuring offset, CMR, T_s and SR intrinsically. 126

6.29 Measuring the output swing voltage intrinsically. 126

6.30 The effect of small CMR. 127

6.31 The effect of small phase margin. 128

6.32 The convergence curves of intrinsic evolution in the stationary environment. a) Using HPSO, b) Using MSPSO. 129

6.33 The standard deviation curves of intrinsic evolution in the stationary environment. a) Using HPSO, b) Using MSPSO. 130

6.34 Mixtrinsic multi-objective evolution convergence curve. 132

6.35 Mixtrinsic multi-objective evolution standard deviation curve. 132

6.36 A low-voltage, low-power CCII implementation [ITF02]. 135

6.37 The frequency response of the evolved CCII. 136

6.38 The frequency response of the evolved CCII after adding the compensation capacitor. 137

6.39 The relation between the input current I_X and the output current I_Z of the evolved CCII. 137

6.40 The relation between the input current I_X and the output voltage V_X of the evolved CCII. 138

6.41 The step response of the evolved CCII after adding the compensation capacitor. . . . 138

6.42 The relation between V_X and V_Y. 139

6.43 The target 3 bit ADC flash converter. 139

6.44 The quantized signal by the initial configuration, the effect of the first deviation model on the quantized signal, and the signal after recovering. 141

6.45 The quantized signal by the initial configuration, the effect of the second deviation model on the quantized signal, and the signal after recovering. 142

6.46 Depiction of filter specification. 143

6.47 Schematic of the target voltage mode filter. 143

6.48 The returned voltage mode filter frequency response of the first setting filter. 144

6.49 The returned voltage mode filter frequency response of the second setting filter. . . . 144

6.50 Schematic of the target current-mode filter. 144

6.51 The returned current mode filter frequency response of the first setting filter. 145

6.52 The returned current mode filter frequency response of the second setting filter. . . . 145

7.1 Block diagram of the reconfigurable generic sensor system. 151

7.2 Blockschaltbild der rekonfigurierbaren generische Sensor System. 151

7.3 Die hierarchische Optimierung generischer Sensorelektronik, Zusammensetzung zu funktionalen Blöcken, wie Instrumentierungsverstärker aus rekonfigurierbaren Operationsverstärkern und passiven Komponenten. 152

7.4 Die mixtrinsische multikriterielle Evolution a) Ein einzelnes Individuum. B) Die vollständige Population. 153

7.5 Die mixtrinsische multikriterielle Evolution Umgebung. 154

List of Tables

2.1 High temperature semiconductor technologies [Goe98]. 9

2.2 The achieved bit resolution after calibrations. 16

3.1 The mean value of the last iteration in static environment. 51

3.2 The standard deviation of the last iteration in static environment. 52

3.3 The mean value of the last iteration before the environmental change in dynamic environment. 55

3.4 The standard deviation of the last iteration in dynamic environment. 56

3.5 The mean value of the last iteration sensitivity analysis. 58

3.6 The standard deviation of the last iteration sensitivity analysis. 59

4.1 Widths of the employed switches [Lan05]. 71

6.1 The first variant of the target and the achieved specifications for extrinsic evolution of operational amplifier in static environment. 111

6.2 Transistor widths and passive component values of a returned configuration to the specification variant at table 6.1. 111

6.3 The second variant of the target and the achieved specifications for extrinsic evolution of operational amplifier in static environment. 112

6.4 Transistor widths and passive component values of a returned configuration to the specification variant at table 6.3. 113

6.5 The third variant of the target and the achieved specifications for extrinsic evolution of operational amplifier in static environment. 114

6.6 Transistor widths and passive component values of a returned configuration to the specification variant at table 6.5. 115

6.7 The last iteration mean value of all the runs using the GA, HPSO, HPSO-TVAC, MSPSO-TVAC, and PSO-TVAC. 117

6.8 The last iteration standard deviation using the GA, HPSO, HPSO-TVAC, MSPSO-TVAC, and PSO-TVAC. 118

6.9 The target specifications for operational amplifier in dynamic environment. 119

6.10 The target and the achieved specifications for operational amplifier. 131

6.11 Transistor widths and passive component values of a returned configurations in the mixtrinsic multi-objective evolution. 133

6.12 Transistor widths and passive component values of a returned folded-cascode operational amplifier configurations in the mixtrinsic multi-objective evolution. 133

6.13 The target and the achieved specifications for folded-cascode operational amplifier. . 134

6.14 The target and the achieved specifications for extrinsic evolution of the CCII. 135

6.15 The CCII configuration that the evolution returned. 136

6.16 The required and the achieved transition levels for the flash ADC, and the effect of the deviations on them. 140

6.17 Transistor widths and passive component values of a returned comparator. 140

6.18 The resistor values before and after recovering. 141

6.19 The requirement of the filter. 143

6.20 Returned voltage mode filter configurations. 144

6.21 Returned current mode filter configurations. 145

6.22 The achieved filter settings 145

List of Procedures

1 Genetic algorithm procedures. 27
2 Particle Swarm Optimization procedures. 35
3 LPSO procedures. 45
4 Procedures to update the local parameters of particle i in LPSO. 46

List of Symbols and Abbreviation

Symbols

Symbol	Description
α	Arithmetic crossover factor for mixing the parents
α_1	A coefficient used in LPSO and LHPSO controller
α_2	A coefficient used in LPSO and LHPSO controller
α_{1_d}	The coefficient α_1 after variation for the sensitivity analysis
$beta_1$	A coefficient used in LPSO and LHPSO controller
$beta_2$	A coefficient used in LPSO and LHPSO controller
γ	An acceleration coefficient used in gregarious particle swarm optimizer
σ	The rate of change of γ per unit iteration
δ_p	The gain of the pass band of a filter is less thanδ_p and greater than $2-\delta_p$
δ_s	The gain of the stop band of a filter
θ	Threshold used in division of labor PSO
ϑ	The environmental temperature
ϕ	Phase margin
Φ	The acceleration coefficient in fully informed PSO
η	Index of the fittest neighbor for the particle i in FDR-PSO
χ	The constriction factor in particle swarm optimization
ε_a	The error in the bit a of the ADC output due to the gain errors in the previous stages
ρ	The mutation step size in guaranteed convergence PSO
ψ_1, ψ_s, and ψ_3	The acceleration coefficients in FDR-PSO
a	Index to a bit number binary ADC
$\mathbf{a}_i$	An extra particle acceleration added in charged PSO
A_o	Open-loop gain
BW_{3db}	The 3db bandwidth
b	The bit resolution of an ADC or a DAC

continued on next page

continued from previous page

Symbol	Description
b_{stage}	the number of effective bits the ADC generates each stage
b_e	The number of the stages that are highly affecting the output because of its deviation
$^{\circ}C$	Degree Celsius
C	The problem space
C_1	The cognitive acceleration coefficient in PSO
C_2	The social acceleration coefficient in PSO
$C_{1,i}$	The cognitive acceleration coefficient of the particle i in LPSO and LHPSO
$C_{2,i}$	The social acceleration coefficient of the particle i in LPSO and LHPSO
CL	The SOC PSO has a global set criticality limit
$CR_{V_X=V_Y}$	Current input range that keeps $V_X = V_Y$ when V_Y is connected to ground
D_{out}	The digital output of an ADC
D_a	The bit number a of the ADC output D_{out}
D_s	The value of the output of single stage in pipelined ADC
$D(s)$	Denominator of a filter transfer function in the s-domain
d	The branching degree of the hierarchical tree in HPSO
D	The number of dimension in the search space
$\mathbf{f}(\mathbf{x})$	An array of objective functions
F_a	Aggregation function
f_{adapt}	The frequency in which, the AHPSO brunching degree change according to it
f_F	The objective function to optimized the filter frequency response defined in [ZPV99]
$\mathfrak{f}$	A variable representing frequency sweeping of AC
$\mathfrak{f}_{max}$	The end frequency of AC simulation
$\mathfrak{f}_{min}$	The start frequency of AC simulation
f_i	The fitness value of the particle i
$f_{\mathfrak{o}}$	The objective function of the objective $\mathfrak{o}$
$f(\mathbf{x})$	The fitness function that returns the fitness value equivalent to $\mathbf{x}$
$\mathbf{G}$	The global velocity in unified PSO
g	The index of the particle with the global achieved fitness in PSO
$G_{closedloop}$	Closed loop gain (of an amplifier)
h	The number of levels between the root and the leaves in HPSO
I_{BP}	Current output of band-pass current-mode filter
I_{HP}	Current output of high-pass current-mode filter
I_{LP}	Current output of low-pass current-mode filter
I_{offset}	The input current when the output current is equal to zero ampere
$\mathfrak{k}$	A coefficient used in enhancing the PSO with constriction factor
$\mathfrak{K}$	The gain of the Palmo integrator
$k_{\mathfrak{o}}$	Weight associated to the objective function $\mathfrak{o}$
$\mathbf{L}$	The local velocity in unified PSO
$m_{\mathfrak{o}}$	The value of the objective $\mathfrak{o}$ that the simulation or measurement return
n_{swarm}	The number of swarms in multi-swarm PSO
N	The number of particles in the population, or the neighbors of a particle if local best model is used

continued on next page

continued from previous page

Symbol	Description
n	The steepness factor in division of labor PSO
$\mathfrak{n}$	The number of hardware extrinsic specifications
$O(\ )$	An operator to show the complexity of an algorithm, for example: $O(N)$ means that the algorithm is proportional to N $O(N^2)$means that the algorithm is proportional to N^2
o	Index to a sub-swarm
$\mathfrak{o}$	index to an object in multi-objective optimization
$offspring1$ & $offspring2$	Indices of offspring
$parent1$ & $parent2$	Indices of parents; particular selected individuals to generate offspring
P	The radius of the shell in which if any particle enter it, it will dispel the particle in the shell center in charged PSO
$P_{g\rightarrow l,i}$	The probability that the particle i will start local search in division of the labor PSO
P_{core}	The radius of the shell in which if any particle enter it, it will not dispel the particle in the shell center to protect division by zero
P_o	The pulse width of the palmo block output
P_+	The pulse width of the palmo block positive input
P_-	The pulse width of the palmo block negative input
$\mathbf{p}_i$	The position where the particle i in the PSO achieved its best fitness value
P_c	power consumption
pf_i	the best achieved fitness value by the particle i
Q	Quantity of charge in a particle
$\mathbb{Q}$	Quality factor of a filter
r_{excl}	The minimum allowed distance between sub-swarms in multi-swarm PSO
r_o	The redius of the subswarm o
R_o	Output resistance
$rand()$	Random number generator that generates a Random value between 0 and 1
R_{rand}	A random number
$\mathbf{r}_{ij}$	The displacement vector between the particle j and the particle i
r_{ij}	The euclidean distance between the particle i and the particle j
ra	Radix of a given stage in the pipelines ADC
ra_a	Radix of the stage a in the pipelines ADC
s	Stimuli signal used in division of the labor in PSO
$\mathfrak{s}$	The sign associated to the output of the CCII element its value is equal to 1 for the CCII+, and -1 for the CCII-
$\mathfrak{s}_\mathfrak{o}$	the required specification for the objective $\mathfrak{o}$
t	Iteration number at a certain time instance
$t+1$	The next iteration to the iteration t
$t+t_0$	The iteration that is occurred t_0 iterations after the Iteration t
t_{max}	The maximum allowed number of iterations (in GA or PSO)

continued on next page

continued from previous page

Symbol	Description
T	Evaporation rate of the pheromone
T_s	Setting time
u	A coefficient for mixing the global and local velocities in unified PSO
v_{max}	The maximum allowed speed in PSO
$V_{o_{dev}}$	The output voltage after the deviation
V_{offset}	The input voltage when the output voltage is equal to zero volt
$V_{out}(\mathfrak{f}$	RMS of the output signal for the frequency $\mathfrak{f}$
$V_{in}(\mathfrak{f})$	RMS of the input signal
w_i	The inertia coefficient of the particle i in LPSO and LHPSO
$W(j)$	A weighting function to acceleration part due to the particle j in fully informed PSO
$\mathfrak{w}_\mathfrak{f}$	The weighting value at the frequency $\mathfrak{f}$
$\mathbf{x}$	A solution in the search space, e.g. position of particle in particle swarm
x_{max}	The maximum value in the search space
$\mathbf{x}_{dev}$	The deviation vector which is a random vector added to the solutions, e.g. position is PSO to model system deviation optimization, or gnome in genetic algorithm
x_d	The dimension d in the search space
x	Index to a termimal in an analog design
$\mathbf{y}$	A solution in the search space
y	Index to a termimal in an analog design
z	Index to a termimal in an analog design

Abbreviations

Abbreviation	Description
ADC	Analog to Digital Converter
APSO	Adaptive Particle Swarm Optimizer
ACO	Ant Colony Optimization
AHPSO	Adaptive Hierarchical Particle Swarm Optimizer
AMS	Analog Mixed Signal
ARPSO	Attractive and Repulsive Particle Swarm Optimizer
BW	Bandwidth
CR	linear Current Range
CAB	Configurable Analog Block
CDBA	Current Differencing Buffered Amplifier
CDTA	Current Differencing Transconductance Amplifier
CLB	Configurable Logic Block
CMR	Common Mode Range
CMRR	Common Mode Rejection Ratio
CPLA	Complex Programmable Logic Array
CPSO	Charged Particle Swarm Optimizer
DAC	Digital to Analog Converter
DAQ	Data Acquisition
DoL PSO	Division of Labor in Particle Swarm Optimization
DSP	Digital Signal Processor
EHW	Evolvable Hardware
EPAC	Electrically Programmable Analog Circuit
FDNR	Frequency Dependent Negative Resistance
FDR-PSO	Fitness-Distance Ratio based Particle Swarm Optimization
FIPSO	Fully Informed Particle Swarm Optimizer
FIPSOC	Field Programmable System-On-a-Chip
FIR	Finite Impulse Response
FPAA	Field-Programmable Analog Array
FPGA	Field Programmable Gate Array
FPMA	Field-Programmable Mixed-Analog-Digital Array
FPPA	Field Programmable Processor Array
FPTA	Field Programmable Transistor Array
GCPSO	Guaranteed Convergence Particle Swarm Optimizer
G-PSO	Gregarious Particle Swarm Optimizer
GA	Genetic Algorithm
GAL	Generic Array Logic
GP	Genetic Programming
HDL	Hardware Description Language
HPSO	Hierarchical Particle Swarm Optimizer

continued on next page

continued from previous page

Abbreviation	Description
INL	Integral Non-Linearity error
JPL	Jet Propulsion Laboratory
KHN filter	Kerwin-Huelsman-Newcomb filter
LPSO	Local Parameters Particle Swarm Optimizer
LHPSO	Local Parameters Hierarchical Particle Swarm Optimizer
MDAC	Multiplying Digital to Analog Converter
MEMS	Micro-Electro-Mechanical Systems
MOTA	Multi-output Operational Transconductance Amplifier
MSPSO	Multi-swarm particle swarm optimizer
NFS	Network File System
OpAmp	Operational Amplifier
OTA	Operational Transconductance Amplifier
PD	Propagation Delay
PLA	Programmable Logic Array
PROM	Programmable Read Only Memory
PSO	Particle Swarm Optimization
PSRR	Power Supply Rejection Ratio
QSO	Quantum Swarm Optimizer
RMS	Root Mean Square
S&H	Sample and Hold
SOC PSO	Self-Organized Criticality Particle Swarm Optimizer
SR	Slew Rate
TCR	Temperature Coefficient of Resistance
TFTP	Trivial File Transfer Protocol
TVAC	Time-Varying Acceleration Coefficients
TRAC	Totally Reconfigurable Analog Circuit
UPSO	Unified Particle Swarm Optimization
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuits
VHF	Very High Frequency

Lebenslauf

Persönliche Daten

Name:	Peter Messiha Mehanny Tawdross
Geboren:	24.März 1978
Staatangehörigkeit:	Ägypter
Familienstand:	ledig

Bildungsgang

Schule	1983-1989 St. Joseph Maronite School / Kairo , Ägypten 1989-1996 St. Fatima Language School / Kairo , Ägypten
B.Sc.	9/1996-10/2001 Higher Technological Institute, 10^{th} of Ramadan, Ägypten B.Sc. der Elektroingenieurwesen und Informatik Abschluss projekt "Three degree of freedom robot arm position control using induction motor"
M.Sc.	3/2002-4/2004 TU Kaiserslautern M.Sc. der Elektrotechnik und Informationstechnik Master thesis "Structural decomposition of signal transition graphs and transformation into XBM machines"
Beschäftigung mit dem Ziel der Promotion:	10/2004-2/2005 Tätigkeit als studentische Hilfskraft mit Abschluss bis zur Verfügbarkeit einer Landesstelle an der TU Kaiserslautern, Fachbereich Elektrotechnik und Informationstechnik Lehrstuhl Integrierte Sensorsysteme. Von 2/2005-12/2007 Wissenschaftlicher Mitarbeiter am Lehrstuhl Integrierte Sensorsysteme, Abschluss der Promotion Dezember 2007.