UNIVERSIDAD DE SEVILLA

Departamento de Electrónica y Electromagnetismo

TESIS DOCTORAL

Diseño de circuitos analógicos y de señal mixta con consideraciones de diseño fisico y variabilidad

Memoria presentada por

Antonio Toro Frías

para optar al grado de Doctor

Directores

Rafael Castro López

Francisco V. Fernández Fernández

IMSE-cnm Instituto de Microelectrónica de Sevilla

CSIC Consejo Superior de Investigaciones Científicas

UNIVERSIDAD DE SEVILLA

Departamento de Electrónica y Electromagnetismo

TESIS DOCTORAL

Diseño de circuitos analógicos y de señal mixta con consideraciones de diseño fisico y variabilidad

Memoria presentada por

Antonio Toro Frías

para optar al grado de Doctor

Directores

Rafael Castro López

Francisco V. Fernández Fernández

Acknowledgements

Me gustaría mostrar mi agradecimiento a todos los que me han ayudado durante mi camino hasta acabar este trabajo. Sin su ayuda y sobre todo sin su apoyo no habría conseguido llegar hasta aquí.

A mis directores Rafa y Paco, junto a Elisenda, por su apoyo durante estos años, por todo lo que he aprendido a su lado y por su paciencia. Junto a ellos, a los demás "platform", Reinier y Manu, hacer la tesis a vuestro lado siempre fue más fácil, fue una suerte trabajar juntos. A mis compañeros de despacho, en especial en los tiempos del PA34. Al resto de personal del IMSE que de alguna manera me ha ayudado durante este tiempo, con sus palabras de ánimo y sus sabios consejos.

Agradecer infinitamente el apoyo de mi familia, en especial por el apoyo y el ánimo, especialmente en los malos momentos, que me han ayudado a llegar hasta aquí, gracias papa, gracias mama. A mi hermana, futura ingeniera.

Por último, gracias a Elisabet, por estar a mi lado durante estos años, por, al menos, intentar entender algo de lo que hay en este trabajo, gracias por tu apoyo, contigo este tiempo ha sido más fácil.

Gracias.

A mis padre y mi hermana …
A Eli …

To my parents and my sister...
To my future wife ...

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Motivation

Advances in microelectronic technology has been based on an increasing capacity to integrate transistors, moving this industry to the nanoelectronics realm in recent years. Moore's Law [1] has predicted (and somehow governed) the growth of the capacity to integrate transistors in a single IC. Nevertheless, while this capacity has grown steadily, the increasing number of design tasks that are involved in the creation of the integrated circuit and their complexity has led to a phenomenon known as the ``design gap´´. This is the difference between what can theoretically be integrated and what can practically be designed.

Since the early 2000s, the International Technology Roadmap of Semiconductors (ITRS) reports, published by the Semiconductor Industry Association (SIA), alert about the necessity to limit the growth of the design cost by increasing the productivity of the designer to continue the semiconductor industry's growth. Design automation arises as a key element to close this "design gap".

In this sense, electronic design automation (EDA) tools have reached a level of maturity for digital circuits that is far behind the EDA tools that are made for analog circuit design automation. While digital circuits rely, in general, on two stable operation states (which brings inherent robustness against numerous imperfections and interferences, leading to few design constraints like area, speed or power consumption), analog signal processing, on the other hand, demands compliance with lots of constraints (e.g., matching, noise, robustness, ...). The triumph of digital CMOS circuits, thanks to their mentioned robustness, has, ultimately, facilitated the way that circuits can be processed by algorithms, abstraction levels and description languages,

as well as how the design information traverse the hierarchical levels of a digital system. The field of analog design automation faces many more difficulties due to the many sources of perturbation, such as the well-know process variability, and the difficulty in treating these systematically, like digital tools can do. In this Thesis, different design flows are proposed, focusing on new design methodologies for analog circuits, thus, trying to close the ”gap” between digital and analog EDA tools.

In this chapter, the most important sources for perturbations and their impact on the analog design process are discussed in Section 1.2. The traditional analog design flow is discussed in 1.3. Emerging design methodologies that try to reduce the ”design gap” are presented in Section 1.4 where the key concept of Pareto-Optimal Front (POF) is explained. This concept, brought from the field of economics, models the analog circuit performances into a set of solutions that show the optimal trade-offs among conflicting circuit performances (e.g. DC-gain and unity-gain frequency). Finally, the goals of this thesis are presented in Section 1.5.

1.2 Analog design: sources of perturbation

As said before, there is a plethora of perturbations that plague the nominal performance of analog circuits. The difficulty in treating these perturbation systematically is one of the reasons for the lag of analog EDA with respect to digital EDA. In this section, a description of three key perturbations (the first two common in all modern integrating technologies and the other gaining importance with today’s scale of integration) are described.

1.2.1 Layout-induced parasitic devices

Since the first silicon technology, the presence of parasitic devices in ICs was reported [2]. A parasitic device is an unwanted element (e.g., resistance, capacitance) which appears in the manufacturing process. Parasitic devices are unavoidable because all materials used in the manufacturing process possess resistive and capacitive characteristics.

These parasitics may critically impact the initial intended performance of the circuit (what is known as nominal performance). To avoid this, the parasitics should be taken into account when designing any circuit. For instance, the parasitic capacitances that appear in the transistor

structure of the inverter in Figure 1.1, can be easily estimated and accounted for during the sizing[a] of this circuit block.

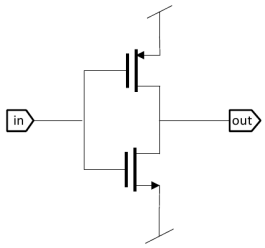

Fig. 1.1 *Schematic of an inverter circuit.*

Strictly speaking, the accurate estimation of device parasitics also requires that the layout phase has been completed. Traditionally, this process takes place after the electrical sizing is completed. Each component and its interconnections are converted into geometric representation of shapes (i.e., the circuit layout) that, when manufactured with the corresponding set of layered materials, will ensure the required functionality of the circuit. As an illustration, the layout of the inverter circuit (Figure 1.1) is shown in Figure 1.2 where a new schematic of this circuit is illustrated with the inclusion of routing parasitic (only resistive elements).

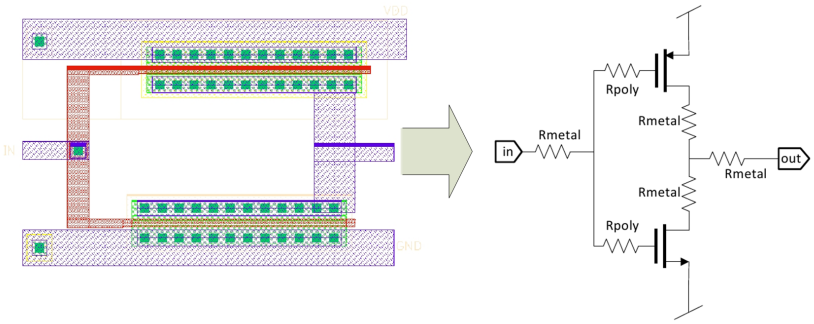

Fig. 1.2 *Layout of an inverter and its schematic with inclusion of resistive routing parasitics.*

A very simple design, as that illustrated here, can contain many parasitic elements (in this case, six parasitic resistors due to routing, but also there are parasitic capacitances due to routing and diffusion capacitors in the transistors). In an analog design, the number of parasitic devices can be very high (in the order of hundreds even for small analog blocks such as simple operational amplifiers). Nowadays, different methods can be used to calculate or estimate these

[a]Broadly speaking, sizing refers to the process of assigning values to the fundamental characteristics of the circuit devices, such as the transistor width and length, resistor values and capacitor values.

parasitic devices. Commercial design packages include parasitic extraction tools to know the impact of this effects.

1.2.2 Process variability

The problem of parasitic devices is not the only one created by imperfections in the manufactured process. All manufacturing processes exhibit small variations in the many steps that need to be followed to progressively build the integrated circuit; in this case, random and systematic errors in the manufacturing of an integrated circuit cause a degree of variability in several transistors parameters [3, 4]. There exist two source of process variability: intradie and interdie. Interdie is a global variability; it refers to variations between devices that are separated by a long distance (i.e. die to die or wafer to wafer), it is usually called process variability. The local variations between transistors in the same circuit, hence, close to each other, are called intradie or mismatch. In this thesis, this source of variability (which include both intradie and interdie) will be called spatial variability (SV) or time-zero variability (TZV) because this perturbation source appears when the circuit is manufactured and to mark a difference with the perturbation source introduced in the following section.

Mismatch can have a critical impact in the circuit performances of both, analog circuits, downgrading or even destroying nominal performances such as offset; and digital circuits, reducing the operation speed due to a variable delay in their devices. For this reason, design techniques are used to reduce the effects of the device variation; for instance, usually making devices large enough so that the local variations become negligible. Another example of the mismatch-aware design technique is the use of common-centroid techniques, carried out at the layout level, to improve the matching of transistors in differential structures.

Nowadays, this is a well-know problem and several methods are available to assess their impact on the circuit performance [5–7]. Commercial design tools evaluate the impact of process variability using a set of model files provided by the foundries. The use of Monte-Carlo simulation is a well-accepted method to evaluate the impact of these non-ideal effects on the circuit behavior.

1.2.3 Aging-induced variability

The aging phenomena have been observed in ICs since the early eighties but their impact on ICs has become more and more an issue due to the scaling of devices and the increasing

electric fields. The most important aging phenomena observed in CMOS technologies are: Bias Temperature Instability (BTI), Hot Carrier Injection (HCI) and Dielectric Breakdown (DB).

Bias Temperature Instability (BTI)

Nowadays, Bias Temperature Instability is considered one of the main transistor wear-out effects. An increase in the threshold voltage (V_{th}) and consequent decrease in drain current is caused by BTI effects. This effect has been commonly observed for pMOS transistors when stressed [b] with negative gate voltages at elevated temperatures (known as NBTI) since 1966 [8]. Since then, several models have been proposed to explain this effect.

Models based on reaction-diffusion mechanism [9] have been one the of the most popular NBTI models [10, 11]. Although this model has been updated to include the saturation effect for long stress times [12], reaction-dispersive-diffusion model has been proposed [13, 14] to explain the immediate recovery of the transistor degradation after the stress is removed. However this model cannot satisfactorily explain the time-dependent recovery effect.

The hole-trapping model, based on a combination of interface state generation and hole trapping has been proposed to describe the permanent and recovery component under one unified model [15, 16]. In addition, with the introduction of high-K metal gates, the effects of PBTI have dramatically increased their importance in nMOS transistors.

Finally, measurements on very small devices have revealed a recovery component in discrete steps, which are different for each devices. This reveals a stochastic behavior of the transistor degradation due to BTI. New models based on trapping-detrapping mechanism have been proposed in the last years [17], which allow to describe the stochastic behavior of BTI on nanometer technologies.

Hot Carrier Injection (HCI)

In the eighties, the transistors were continuously scaled while maintaining the supply voltage, which gave raise to this aging-related issue in ICs. The high electric field in the transistor channel causes that some particles, electrons or holes, attain a very high kinetic energy. These hot carriers gain enough kinetic energy to overcome the potential barrier necessary to break an interface state. When this occur, these carriers are injected into forbidden regions[c] of the

[b] Stress or stress conditions are the voltages in the terminals of the devices during the operation time of a circuit.

[c] Forbidden regions of a device, i.e., as the gate dielectric, where the hot carriers can get trapped or cause interface states to be generated.

transistor and it causes a shift (degradation) in the electrical characteristic (e.g., in the threshold voltage or in the output conductance).

Although HCI was the main effect for transistor degradation during the eighties, in the nineties this problem almost disappeared due to the reduction in the operating voltages of the circuits. Nowadays, HCI is again a problem because the non-scalability of the threshold voltage prevents the reduction of the supply voltage at the same pace of the transistor scaling.

A large number of compact models has been developed to evaluate the HCI effects. Most of them [18–20] are based on the ”lucky electron” concept [21]. Alternative models have been developed when scaling to deep submicron technologies to include other sources of HCI [22, 23]. In nanometric technologies, the impact of HCI can then no longer be described as a deterministic value but as an statistical distribution [24].

Dielectric Breakdown (DB)

This phenomenon is caused by dielectric breakdown due to gate oxide damage. Usually, DB effects result in a circuit failure because the gate current is noticeable increased and the control of the device by its gate voltage is lost. The introduction of thin gate dielectrics has created the so-called *soft breakdown*. This new effect does not necessarily correspond with a circuit failure. Although there are models [25] to capture the soft breakdown and to calculate the probability of transistor failure due to DB [26], this phenomenon has a little impact in analog circuit due to: (1) the small number of transistors, compared to digital circuits, which decreases the chances to suffer a dielectric breakdown and; (2) the voltages at which this happens are not usually used for analog circuits (typically the maximum voltages for the technology).

The aging-induced variability is called as time dependent variability (TDV) because it depends on the operation time of a circuit.

1.3 Traditional design flow

The difficulty of the analog circuit design problem is well known. The design of an IC starts with the definition of the system-level required performances (i.e., system.level specifications). The divide-and-conquer technique reduces the complexity of the problem by recursively breaking it into “manageable pieces”, as shown in figure 1.3. The most basic design conceptual methodologies for analog circuits use either *Top-Down* or *Bottom-up* flows, although the most common design methodology for analog ICs uses the so-called *Top-Down / Bottom-Up* flow.

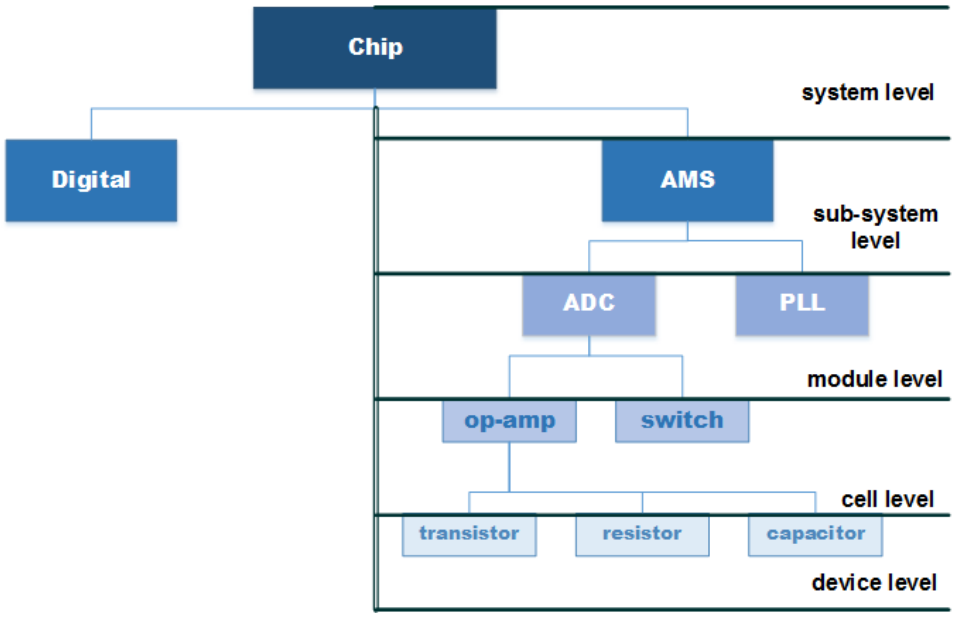

Fig. 1.3 *Hierarchical structure used in circuit design.*

On the one hand, in a *Top-Down* design flow, a circuit is designed with successively more detail. That is, the system is decomposed in subsystems and its specifications are mapped into a sub-set of specifications for each subsystem. This process is repeated until, at the device level, the active and passive devices (transistors, resistors, etc.) are sized. The main advantage in the *Top-Down* design approach is that the final system performance is known (but only estimated) since the beginning of the design process. If any of the different subsystems does not attain the required specifications, the designer must go back to correct this error. This can affect several components of the same or higher levels, and, in the worst scenario, a change in the system architecture may be necessary, which could be very costly in terms of design time.

On the other hand, the *Bottom-up* flow begins by designing the individual cell-level blocks and ends up with the assembly of the system-level circuit. The main disadvantage of bottom-up flows is that the system performance cannot be verified until all the blocks have been completed. That could lead to important design changes late in the design process (and, possibly, changes that are probably very difficult to undertake, since the change may involve different levels).

For these reasons, in practice, the most used methodology is a hierarchical design flow combining the *Top-Down* and *Bottom-up* flows, together with redesign loops when sources of perturbations are included. This hierarchical design flow consists in a top-down electrical synthesis and verification and a bottom-up physical synthesis and verification. The design flow

is depicted in Figure 1.4 , where the steps between any two hierarchical levels are shown. These steps are:

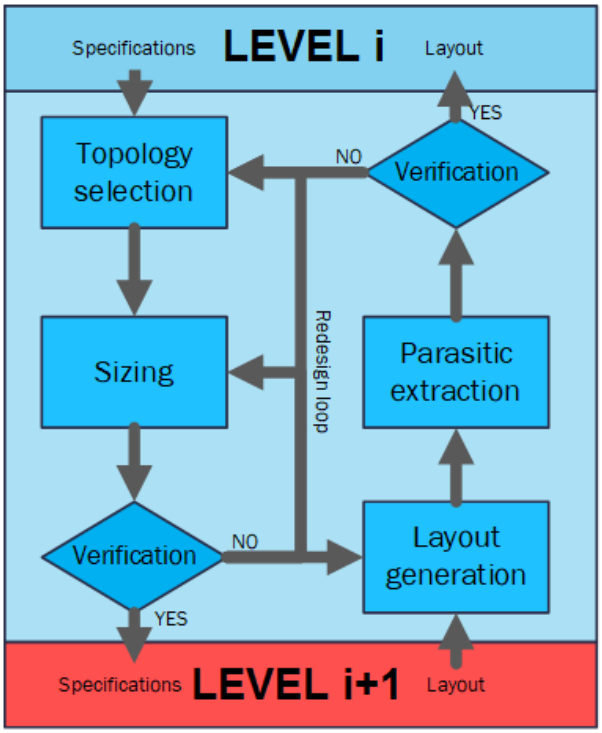

Fig. 1.4 *Top-Down / Bottom-Up design flow.*

1. **Electrical synthesis**

   - **Topology selection:** given the performance specifications from the higher level, the architecture/topology for this level that can address these specifications is selected.
   - **Sizing:** at higher levels, sizing is the process of mapping the specifications for each block of the immediately lower level. At the device level, sizing is the process of dimensioning of the passive and active components.
   - **Verification:** the design is simulated and verified against initial specifications. If specifications are fulfilled, the flow continues to next level, and in the device level, it is followed by the bottom-up physical synthesis. If the specifications are not fulfilled, any of the previous steps will have to be repeated. The verification should ideally include as many sources of perturbation as possible, such as process variability through a Monte-Carlo simulation.

2. **Physical synthesis**

   - **Layout generation:** using the set of masks (or materials) available in the technology, the layout of the devices or blocks and, the placement and routing of these, are generated.
   - **Parasitic extraction:** the layout is first checked for violations of the process-specific design rules (DRC) and then a layout-versus-schematic (LVS) check is performed. The parasitic extraction process is used to obtain the layout-induced effects, which means that a set of parasitic devices (e.g., capacitances, resistances) are added to the circuit as explained in Section 1.2.
   - **Verification:** the circuit, with the inclusion of the parasitic devices, is simulated. As in the electrical verification, if specifications are not fulfilled, any of the previous steps would have to be repeated. It is important to note that this process of finding the parasitics that causes the circuit to fail to meet its specifications is very time-consuming and not very well systematized.

In this design flows, it is possible to automate separately one or more than one steps. For example, the use of an optimization engine with a circuit performance evaluator (e.g., an electrical simulator) to obtain the nominal design circuit sizing is shown in Figure 1.5.

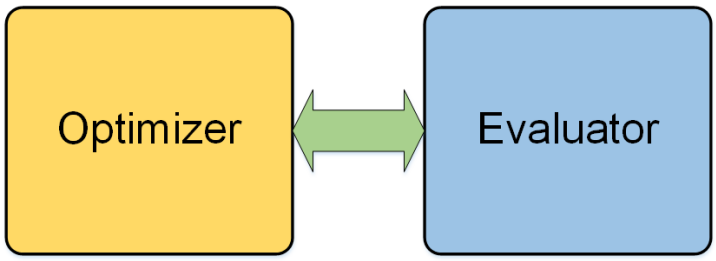

Fig. 1.5 *The combination between an optimizer and an evaluator as a cornerstone of design automation methodologies.*

1.4 Multi-Objective Bottom-Up Methodology

In traditional design flow, the electrical and physical synthesis are completely separated using a top-down and a bottom-up design flows respectively. This presents some drawbacks: (1) at the beginning of the design process, it is difficult to estimate the power consumption and area

occupied by the design, as these figures, which are key to the circuit design, rely on low level design details ; (2) it is possible that specifications at lower levels cannot be met and; (3) it is possible that when the top-level system is verified (during the bottom-up portion of the flow), it does not meet the desired specifications.

These drawbacks can imply repeating the whole traditional design process, causing a considerable increase of the design time and possibly hampering design closure. Alternative design flows, known as *platform-based* methodologies [27, 28], have been proposed in the literature to overcome these drawbacks. In these methodologies, the *feasibility space* of lower levels is generated, that is, the complete set of values for the circuit performances that the specifications can take for a specific topology.

One of the main advantages of the platform-based methodologies is the reuse because the feasibility space generated for a certain topology can be reused in many cases. Nevertheless, the transmission of the feasibility space through the hierarchy and how this information is generated are two key problems to use this design flow. Moreover, *platform-based* methodologies still inherit the drawbacks of pure Top-Down design flows (e.g., the flow must go back if any subsystems does not attain the required specifications). Due to these drawbacks, *Multi-Objective Bottom-up* (MOBU) flows, shown in Figure 1.6, have been proposed [33, 30]. Even so, one key elements of MOBU flows that helps reducing the complexity of dealing with entire feasibility spaces is the concept of Pareto-optimal Fronts (POFs). In the following section, this concept will be described in detail and the optimization method used to generate them will be explained.

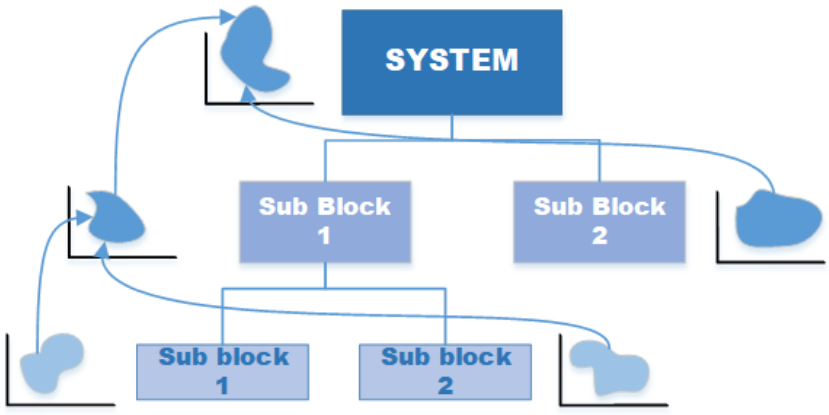

Fig. 1.6 *Bottom-Up methodology.*

1.4.1 Pareto Optimal Fronts

POFs have been successfully applied to the electrical sizing of analog circuits. For this purpose, multi-objective optimization techniques are used to generate POFs, another improvement towards bridging the design gap in the analog arena.

Most problems in nature have several (possibly conflicting) objectives to be optimized. Many of these problems are frequently treated as single-objective optimization problems by transforming all but one objective into constraints or by a cost function. The second option refers to using a single cost function, a simplification made possible by using a weighted sum of the objectives.

In analog design, the design problem will usually be multiobjective, and the goal here is to find optimal *trade-offs* rather than a single solution. The multi-objective optimization problems can be posed as:

$$\begin{aligned} \text{Minimize} \quad & \vec{F}(\vec{x}), && \vec{F}(\vec{x}) = \{f_1(\vec{x}), f_2(\vec{x}), ..., f_n(\vec{x})\} \in \mathbb{R}^n \\ \text{subject to} \quad & \vec{G}(\vec{x}) > 0, && \vec{G}(\vec{x}) = \{g_1(\vec{x}), g_2(\vec{x}), ..., g_m(\vec{x})\} \in \mathbb{R}^m \\ \text{where} \quad & x_{Li} \leq x_i \leq x_{Ui}, && i \in [1, p] \end{aligned}$$

The vector $\vec{x}$ is the set of variables and x_{Li} and x_{Ui} are their lower and upper bounds. The vectors $\vec{F}(\vec{x})$ and $\vec{G}(\vec{x})$ correspond to the objectives to be optimized and the constraints, respectively. This formulation is for an all-minimization problem; if any of the objectives should be maximized, it can be easily transformed, e.g., by simply multiplying it by -1.

In this context, Pareto efficiency is a concept of economics with applications in engineering and social sciences. The term is named after Vilfredo Pareto, an economist who used the concept in his studies of economic efficiency and income distribution. In a Pareto efficient economic system no allocation of given goods can be made without making at least one individual worse off. Given an initial allocation of goods among a set of individuals, a change to a different allocation that makes at least one individual better off without making any other individual worse off is called a Pareto improvement. The concept of POF allows reducing the complete feasibility space and thus, to reduce the complexity of dealing with this information, using only the information of the POF.

A definition of the concept of Pareto-dominance criterion in a multi-objective optimization problem is that a solution $\vec{x}_1$ is said to *dominate* another solution $\vec{x}_2$ if $\vec{F}(\vec{x}_1) < \vec{F}(\vec{x}_2)$ and $f_i(\vec{x}_1) < f_i(\vec{x}_2)$ for at least one function f_i. This definition is valid when both individuals fulfill

all design constraints. Otherwise, solution $\vec{x}_1$ is said to *dominate* solution $\vec{x}_2$ if $\vec{x}_1$ violates design constraints to a lesser extent than $\vec{x}_2$. The non-dominated set of the entire feasible search space is known as the Pareto-optimal front (POF).

These concepts are shown in Figure 1.7, where both objective functions are minimized. Vector $\vec{x}_D$ is dominated by vector $\vec{x}_C$, but $\vec{x}_C$ is not Pareto optimal because it is dominated by vectors $\vec{x}_A$ and $\vec{x}_B$. These vectors are non-dominated because no vector dominates them, so vectors $\vec{x}_A$ and $\vec{x}_B$ are Pareto-optimal. The Pareto front is made up by those individuals that are not dominated. Though in figure 1.7 the Pareto front seems to be a continuous frontier, using a multi-objective algorithm, it is actually a discrete set of circuit designs.

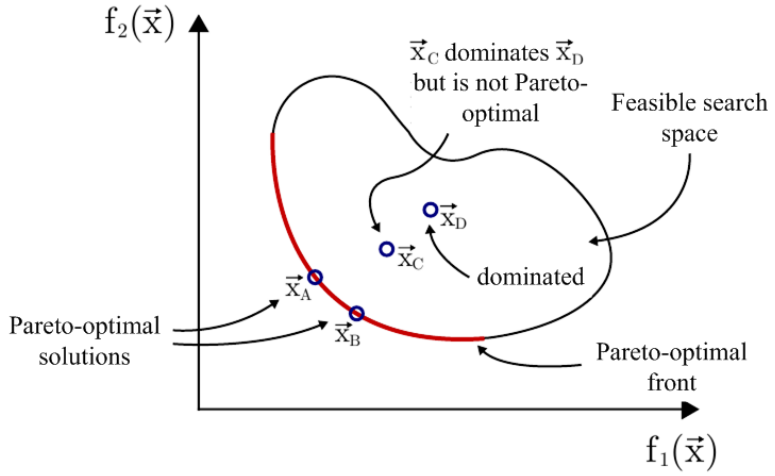

Fig. 1.7 *Basic concepts of a Pareto-Optimal Front.*

Figure 1.8 shows a feasible search space in a two objectives problem. In this space, four different POFs can be defined depending if the objectives are minimized or maximized. An example of Pareto-Optimal Front of an analog circuit is shown in figure 1.9, in which the objectives to optimize are the DC-gain and the unity-gain frequency.

Pareto-optimal fronts are the solution of the so-called Multi-Objective Optimization problems. The algorithms used to solve this type of problems start with a random set of design solutions (known as individuals) and, very much like the evolutionary process in nature, mutate, cross-over and selects new individuals (across a number of generations) to optimize the circuit performance. The goal is to attain the Pareto-optimal front of the circuit: the set of designs with best trade-offs between the conflicting performances.

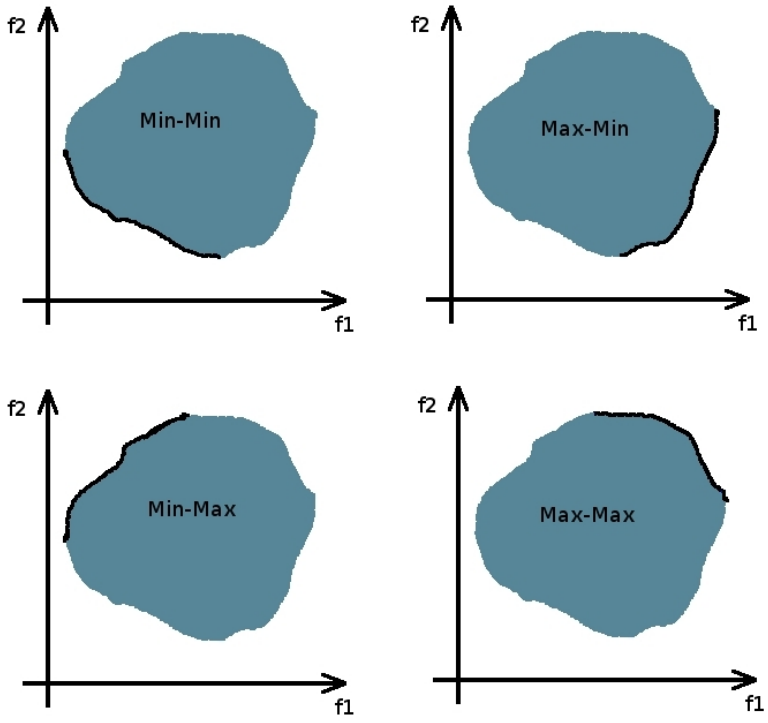

Fig. 1.8 *Different POFs in the same feasible search space.*

The use of Pareto-optimal Fronts in the design of analog circuits is presented in [30], [31] and [32]. But these solutions do not include information about the impact of the sources of perturbation.

Using the concept of Pareto-Optimal Front and a hierarchical composition of performance models [33], it is possible to generate the best feasible designs until the system level, as shown in Figure 1.10. It is important to note that if the impact of the sources of perturbations is not included in the POFs used, when the solution is chosen at the system level, the impact of these sources needs to be included in the design of each sub-block. In this process, re-design loops to include these effects can be needed and even the performance of the chosen solution can be degraded.

1.4.2 Optimizer

The problem of finding the POF of an analog circuit is a multi-objective optimization problem (MOOP). During the past 15 years, multi-objective evolutionary algorithms (MOEAs) have

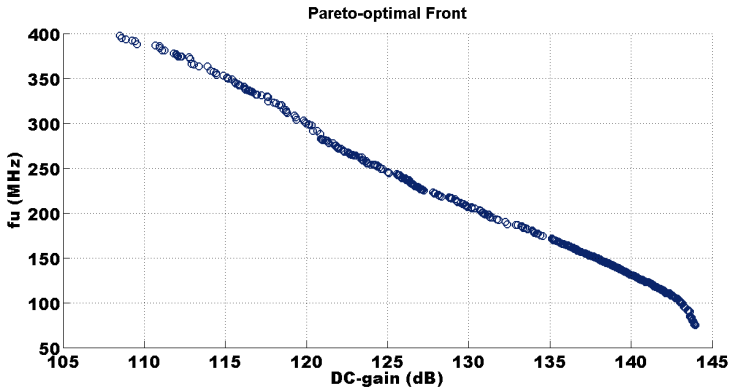

Fig. 1.9 *Pareto-Optimal Front of an analog circuit.*

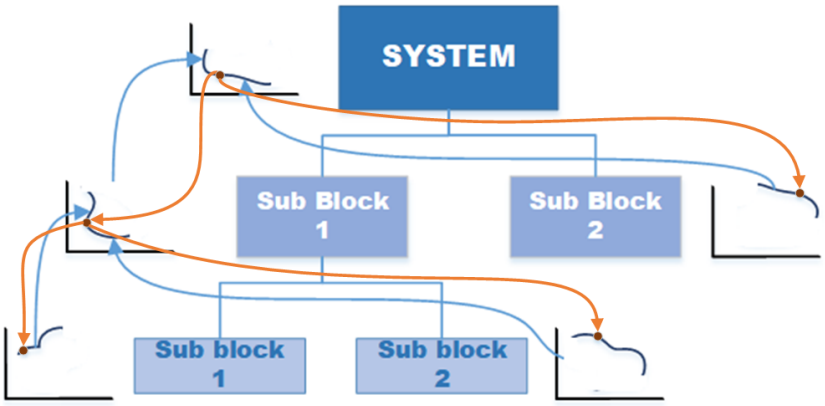

Fig. 1.10 *Bottom Up design flow with the use of POFs and Top-Down solution.*

shown their usefulness for solving multi-objective optimization problems. In the field of analog integrated circuits, they provide a set of feasible solutions for the optimal synthesis and sizing of different kinds of circuits. As in natural evolutionary processes, emergence of new individuals in each generation is a requirement to introduce improvements in the population. For this purpose, these algorithms make use of the mutation and crossover operators that work on already existing individuals.

In a multi-objective optimization problem posed for analog circuits, the vector $\vec{x}$ is the set of design variables (e.g., transistor W and L). The vectors $\vec{F}(\vec{x})$ and $\vec{G}(\vec{x})$ correspond to the design objectives to be optimized (e.g., area, DC-gain) and the design constraints (e.g., margin phase$> 50^o$), respectively. This formulation is for an all-minimization problem; if any of the objectives should be maximized, it can be easily transformed, e.g., by simply multiplying it by -1.

The solution to this problem is a non-dominated set of individuals (i.e., sized circuits), that is, the Pareto front. To solve the optimization problem, the vectors $\vec{x}_i$ are evaluated in $\vec{F}(\vec{x})$. In this case, the circuits are simulated to know the value of their design objectives and design constraints. The individuals that attain the design constraints forms the *feasible search space*. Using the concept of *Pareto dominance*, the algorithm will try to reach the *Pareto-optimal front*. The evolutionary algorithm called NSGA-II [d] will be used in this Thesis.

NSGA II

NSGA-II is based on the evolution of a population of individuals, formed by a *popsize* individuals, through a number of M_{gen} generations. To generate the first population, a set of steps are used by NSGA-II:

- **Initialize population:** the initial population is generated. The variables of each individual in this initial population are chosen randomly within pre-defined ranges. These ranges must be defined by the designer, so they rely on previous design knowledge.
- **Evaluate population:** using an evaluator, the values of design objectives and constraints is calculated for each individual of the population. If any design constraint is not fulfilled, the constraint violation (CV) parameter is calculated. This parameter is used by NSGA-II to select the best individuals when none of them fulfills the design constraint. For each individual we have to evaluate these functions and, if any of them is less than 0 (note that the definition of these functions forces them to take positive values), its negative value is accumulated in this parameter.
- **Assign ranking and crowding distance (CD):** all individuals are classified into fronts as follows: non-dominated individuals in front 1 (F1), individuals dominated by at least one individual of front 1, are located in front 2 (F2), individuals dominated by at least one individual of front 2 are located in front 3 (F3), and so on; this step is also called rank assignation, rank 1 for F1, rank 2 for F2, etc. The crowding distance (a measure of how well distributed are the solutions in the performance space) is assigned to each individual and it is a measure of the distance between this individual and those which are closest in its front (in each dimension). If individuals do not meet the design constraints, the constraint violation parameter is used to assign the ranking.

[d] NSGA-II is the most widespread multiobjective evolutionary algorithm in the scientific community with very positive results and the source code is available for free.

To generate a new population, the first population is used as *parent population*. This population is used to generate a new *child population* and the rest of the process is repeated across M_{gen} generations as shown in Figure 1.11:

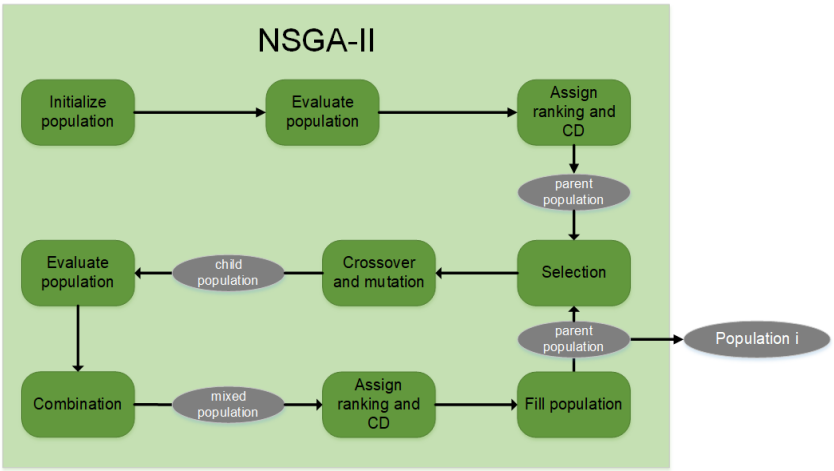

Fig. 1.11 *NSGA-II internal operation flow.*

- **Selection:** The individuals are taken four by four from this *parent population*. From these four individuals, a tournament is carried out two by two. This tournament consists in, first, a dominance check. If as a result of this check the individuals are non-dominated, then the one with larger crowding distance is selected.
- **Crossover and mutation:** The selected individuals are used to generate new individuals using the crossover operator. Finally, the mutation operator performs random variations on them. This populations is called *child population*.
- **Evaluate population:** The individuals forming *child population* are evaluated.
- **Combination:** the *child population* and *parent population* are mixed.
- **Assign ranking and crowding distance (CD):** A new rank assignment process and crowding distance are performed to the new population, *mixed population*.
- **Fill population:** the best individuals of the population are selected to generate the individuals of the first population. First, the individuals are sorted from lowest to highest rank and the individuals with the same rank are sorted by higher crowding distance. The first *popsize* individuals form the *population 1*.

After using this procedure during M_{gen} generations, the Pareto-Optimal Front is reported in the *final population*. For that, an evaluator it is needed to obtain the values of design objectives and constraints.

1.5 Goals of this Thesis

In this work, several methodologies are proposed to improve the analog designer productivity. These methodologies provide the solutions in form of Pareto-Optimal Fronts, which promote reuse-based design practices and their use in MOBU design flows.

To reduce or, at least, minimize the iteration between different design steps, the methodologies proposed in this work combine the use of POFs with the automated inclusion of sources of perturbations, which means each solution in the POF is completely sized and with accurate information about the impact of these sources. For this purpose, a set of design flows based on the use of the optimization algorithm presented in this chapter and different evaluators, similar to that shown in the Figure 1.12, are proposed in this Thesis.

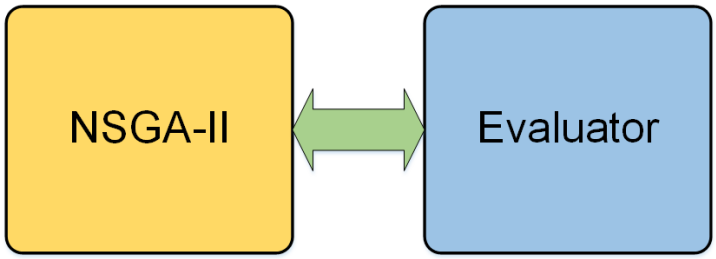

Fig. 1.12 *Optimization technique using NSGA-II and an evaluator.*

In Chapter 2, a new design flow is explained in detail, in which the evaluator includes the physical implementation, which allows including the impact of the parasitic elements in the electrical performance. This process avoids iterations between electrical and physical synthesis.

In order to improve this methodology, a method to assess the quality of the automated layout generation technique is discussed in Chapter 3. A new design flow including this improvement is presented in Chapter 4.

In addition to the impact of parasitic devices, the impact of spatial variability and aging is growing in new technologies, which causes reliability circuit problems in these technologies. To assess this impact, a reliability simulator is proposed in Chapter 5, which can be used as evaluator to include accurate reliability information during the POFs generation. A new design

methodology to generate POF with reliability information provided by this new evaluator, which is a reliability simulator, is explained in Chapter 6. Finally, conclusions are drawn in Chapter 7.

Chapter 2

POF-based Layout-Aware Design Flow

As mentioned in Chapter 1, the parasitic devices usually cause a degradation of the circuit performance. Traditionally, electrical sizing and layout design have been addressed by trial and error iterations to compensate for the undesirable effects that parasitic devices cause in the electrical behavior. The first logical step towards efficient automation of the design flow is the undertaking, separately, of the automation of electrical and physical phases. To automate the first one, the most usual technique is an optimization engine linked to a circuit performance evaluator (e.g., an electrical simulator). To automate the second, knowledge-based and optimization-based techniques have been reported.

The goal in this chapter is to develop a new design methodology that includes accurate information about the impact of parasitic devices in the POF generation. For this, a new evaluation process is proposed to include this source of perturbation automatically. The followings goals are defined:

- **Avoid iterations.** Iterations between layout generation and electrical sizing to compensate for parasitics prevents fast design closure, not only for the time that these iterations add but also to the complexity of tackling the impact of one parasitic among a list of thousands.
- **Improve synthesis.** Analog circuit design requires very specific expert knowledge to be incorporated into the automated synthesis processes to improve the efficiency of the design process.
- **Reuse.** It is important to extend the notion of library-based approaches, and thereby reduce overall design time. In this sense, the use of POFs helps increasing the degree of reusability.

This chapter is devoted to present a technique that fulfills all three objectives above.

2.1 The layout-aware design flow

To avoid iterations between the electrical and physical design phases, a layout-aware design methodology has been proposed in [34]. This methodology brings the physical implementation information into the electrical synthesis process. Figure 2.1 shows the basic flow diagram of this solution.

The idea behind the layout-aware solution is to size the analog circuits with all the information about the physical implementation (that is, with an accurate estimation of the parasitic elements that cause deviations from the required performance of the circuit). An optimizer is used to carry out the sizing process together with a circuit performance evaluator. In addition, to knowing the impact of parasitic devices on the circuit performances, a circuit layout generator and a parasitic extraction method are needed to know the impact of parasitic devices on the circuit performances.

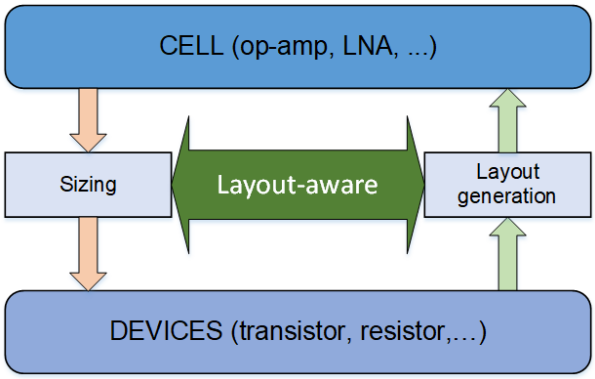

Fig. 2.1 *Basic idea behind the Layout-aware technique.*

Using an optimizer typically implies hundreds or thousands evaluations of the circuit performance. For this reason, one of the critical elements in this layout-aware flow is to have a fast and high-quality automated layout generator [35]. To automate the layout generation, there are two different approaches:

- **Optimization-based approaches:** The design rules are introduced into an optimization algorithm (this would mean an optimization loop for layout generation within another optimization loop, the main one, for the layout-aware design). The time to layout generation using this technique can unfortunately be very high (from minutes to some hours).

During the circuit optimization, many layout generations (>1000) may be necessary, so this technique may not be adequate in terms of CPU time.

- **Knowledge-based approaches:**

  There is a predefined design plan to find and combine the elements such that layout requirements (symmetries, compaction, abutment, ...) are met. The design plans, in the form of design equations, design heuristic strategies, or both, are implemented to mimic the steps an expert designer may take to reach an optimum solution of the layout design problem. One of the most adequate techniques is the *template-based* approach, in which the expert knowledge of a designer is captured in a layout template. A predetermined placement and routing scheme is defined with all necessary component-to-component and routing relationships. Using this technique, the time to layout generation is usually less than one second.

There are several works in the literature that report an optimization including parasitics devices. For example, an evolutionary algorithm and a procedural method for layout generation are used in [36], but the extracted information about parasitics is very limited. A knowledge-based sizing method is used in [37] but the information about parasitics is incorporated only in the final phases of the process.

The solution in [38] is implemented in two phases: first, only electrical sizing is done, and then, information about parasitic is added, but only the number of fingers in the transistors is used to minimize the area.

The use of linear programming for layout generation at the same time than sizing is reported in [37], but the occupied area is not optimized because it uses heuristic estimates.

Layout-templates and commercial tools for parasitics extraction are used in [39], but the use of symbolic analysis restricts the knowledge about circuit performance to the small-signal behavior.

2.2 Proposed solution: a POF-based Layout-Aware Design Flow

The proposed idea is a new methodology, in which the layout-aware design advantages are combined with the use of Pareto-optimal fronts (POFs). In the previous section, the layout-aware design flow has been presented. This design flow is now extended to the MOBU methodology.

As noted in Chapter 1, having Pareto fronts is essential because, in the design of analog circuits, there is rarely a single goal to optimize, but many, which turns the design into a multi-objective optimization problem.

In recent years, a method to generate Layout-Aware Pareto-Optimal Front has been proposed in [40], where AIDA [41] is used to generate the layout and estimate the parasitic devices. Although the impact of parasitic devices is well estimated using this methodology, the layout generated by AIDA suffers of some sections of unused silicon area because no technique to optimize the occupied area is used. In addition, the CPU-time needed by the automated routing task can be very high.

2.2.1 General Flow

The Pareto-Based Layout-Aware flow presented here include two components: (1) geometry-aware sizing and (2) parasitic-aware sizing. The first optimizes the design in terms of occupied area, trying to attain adequate geometry-related parameter values (such as the number of fingers of a MOS transistor) so that several geometry aspects (e.g., area occupation, aspect ratio) are correctly optimized. With the parasitic-aware component, all layout parasitics are included in the electrical evaluation of the circuit during the optimization so that the final solutions are robust against the impact of resistive and capacitive parasitics. It is important to note that both components are essentially linked: for instance, changing the number of fingers changes diffusion parasitics, and reducing diffusion parasitics means finding out the optimum number of fingers.

The proposed POF-based layout aware flow provides a set of solutions that: (1) include the physical implementation, so the iterations between electrical and physical synthesis are avoided; (2) generate a performance trade-off in form of Pareto-Optimal Front; (3) optimize the design from the geometry perspective (i.e., minimizing the occupied are); (4) are robust against the impact of resistive and capacitive parasitics. Besides, the power consumption is optimized to prevent that the energy is used to compensate the performance degradation due to parasitics and not to the main circuit operation. This effect usually happens when the parasitic elements are included just in the final phases of the design process.

A basic block diagram of the proposed methodology is shown in figure 2.2. The methodology consists in an iterative process between the multi-objective optimization algorithm NSGA-II and an evaluation process.

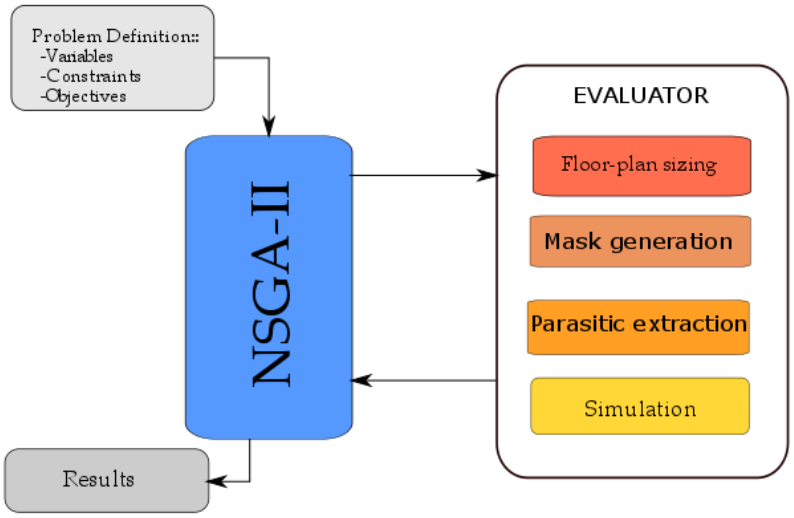

Fig. 2.2 *Basic diagram of the proposed methodology.*

As can be seen in figure 2.2, the first step to generate the layout-aware POF is the problem definition, that is composed of the following elements:

- **Variables.** In this case, the design variables can be the transistor sizes, passive device values, and bias currents and voltages. For each variable, a range of values has to be defined.
- **Objectives.** The parameters to be optimized are the design objectives, such as the maximization of the gain-bandwidth product, or the minimization of the power consumption.
- **Constraints.** They are used to ensure the correct operation of the circuit, for example, its stability. Additional constraints can be imposed to ensure specific design considerations, e.g., minimum gain or maximum power.

The evolutionary algorithm called NSGA-II has been selected in the proposed methodology. The reader is referred to Section 1.4.2 for further details. The evaluator, in this case, has several elements that come into play to provide geometry and parasitics-related information:

- **Floor-plan sizing.** This element of the evaluator provides the geometry-related parameter values, such as the number of fingers of a MOS transistor, to satisfy a set of geometry constraints that depend on the specific placement of all devices in the circuit and that will be detailed later. The geometry-related parameter values are not provided by the

optimizer because the optimization problem would become very complex, which would compromise its convergence.

- **Mask generation.** This element carries out the generation of the masks representing the physical view of the circuit.
- **Parasitic extraction.** A parasitic extraction tool is used to include all parasitic devices in the electrical evaluation of the circuit during the optimization. The parasitic elements are extracted and added to the circuit elements. This *netlist* is known as *extracted view* or *extracted circuit*.
- **Simulation.** The use of the electrical simulator (Hspice, Spectre, etc.) is necessary to evaluate the performances of the extracted circuit. Different analyses can be carried out (AC, DC, transient, ...) to properly measure all design objectives and constraints.

2.3 Implementation of the proposed methodology

The complete diagram flow of the implementation is shown in figure 2.3. This flow is completely automated. To implement this automation capability, PERL [42], shell [43], and OceanScripting [44] have been used; all scripts used to help the communication between different tools are shown in green. In addition, the main communication between Cadence, a very well-known and commonly used IC design framework, and the optimizer have been developed using *pipes* [45], which provides an efficient communication channel.

2.3.1 Evaluator

The four tasks of the evaluator are carried out by two separate tools. On the one hand, the floor-plan sizing task is carried out by a purposely develop module since there were none available and, on the other hand, mask generation, parasitic extraction and simulation, are carried out using Cadence, a commercial design suite.

Floor-plan sizing

The floor-plan sizing task, that is, the task of assigning a value to a set of geometrical parameters to comply with specific geometry constraints (such as minimizing the occupied area) is carried out by the Geometric Constraints Module (*GCModule*). Among the different methods to carry

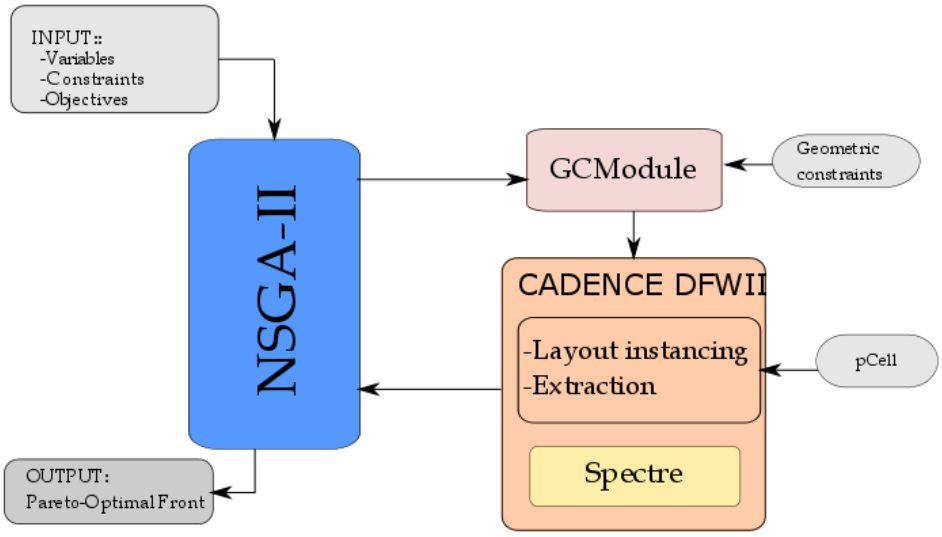

Fig. 2.3 *Tools used in the POF-based Layout-Aware implementation.*

out the automated layout generation that has to take place in the loop of Layout-Aware design, the technique based on layout templates, one of the knowledge-based approaches mentioned in Section 2.1, has been selected due to the short time needed to generate each layout using this technique. Layout templates are often accused of lack of flexibility. For this reason, the use of a modified version of the Stockmeyer's algorithm [46] is applied here, which helps alleviating the flexibility problem. The use of Stockmeyer's algorithm in analog design with geometric constraints is reported in [47] and [48]. However, none of these works include the electrical sizing at the same time than the layout generation.

To carry out the floor-plan sizing, the first important step is defining the style of the placement that is going to be used. In this Thesis, the *slicing style* has been selected, that is, considering the layout as a rectangle:

- there are no overlapping rectangles;
- each basic rectangle is a building block[a] or, a line segment (horizontal or vertical) divides it in two pieces such that each piece fulfills these two conditions.

[a] A building block can be a transistor, a capacitor, a resistor or a group of them, for example a differential pair or other structures.

An example of such floorplan style is shown in figure 2.4 (in this example, the channels or spaces between blocks for inter-block routing are not shown). A useful way to describe the slicing floorplan is by representing the hierarchy structure of the slices with an oriented rooted binary tree called a *slicing tree*, as shown in this figure. Each node specifies whether the slice is horizontal (**H**) or vertical (**V**) or a basic building block.

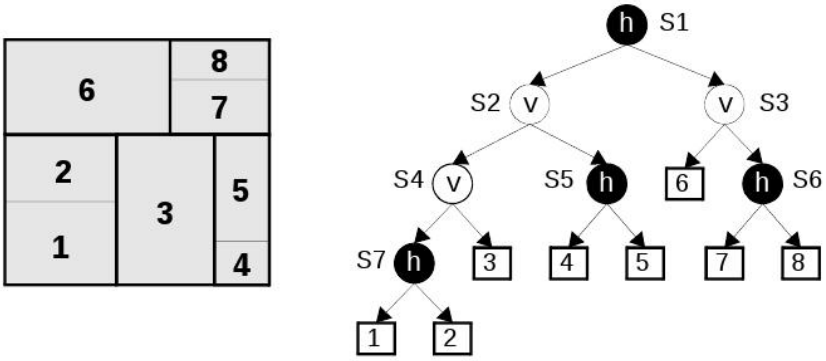

Fig. 2.4 *Example of a binary slicing tree representation of a slicing-style layout template.*

Each basic building block has several possible shapes, that is, different values of the pair (width, height). Given an electrical sizing of the devices of the basic block, all possible combinations of geometric parameters (e.g., number of fingers of the transistors) are examined. This process builds up the so-called *shape function* for each block.

Figure 2.5 shows an example of basic block: a differential pair. In this example, the electrical sizing of the transistors (W,H) is the same, but different geometrical parameters have been used, so that the pair have different forms (width and height pair) for each one. An example of shape function is shown in figure 2.6, for a single transistor block. The geometric parameter here is the number of fingers.

GCModule selects the combination of geometric parameters to optimize a certain objective function $\Phi(W,H)$ (in this case, the area), in two phases. First, shape functions of basic blocks are combined with a modified Stockmeyer's algorithm that generates a list of pairs (width and height) of all possible solutions. Then, the point in the shape function with minimum area is selected, and if there are more than one solution with equal minimum area, the solution with minimum area loss (i.e., the amount of area that is not used by the devices or the routing lines, as shown in figure 2.7) is selected.

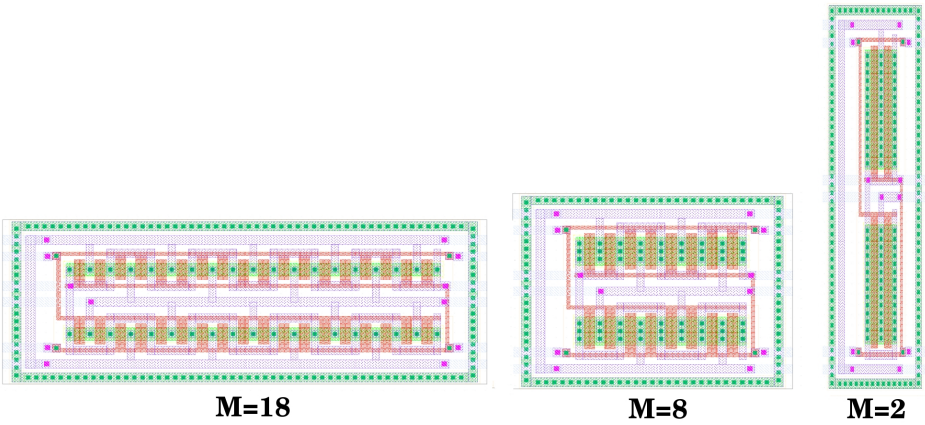

Fig. 2.5 *Differential pair layout with different number of fingers.*

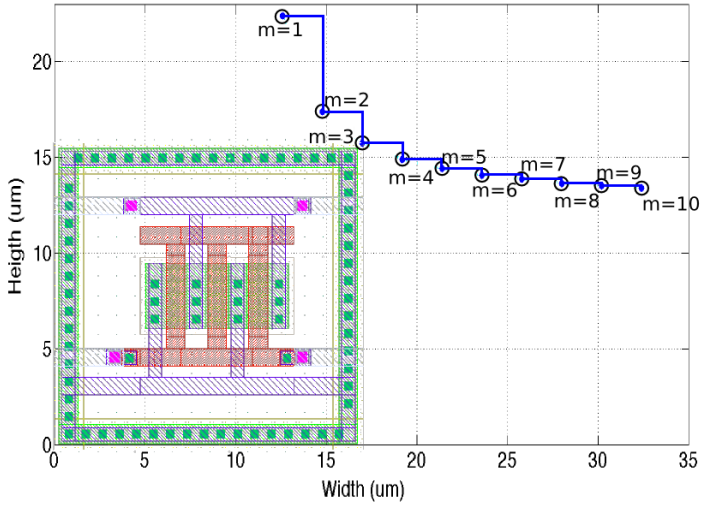

Fig. 2.6 *Shape function example.*

GCModule has been implemented in C++. In addition, to area and area loss, *GCModule* can calculate the solution according to certain geometric constraints set by the designer. The geometric constraints that can be defined are:

- **Aspect ratio**. The ratio between circuit layout width and height ($AR = W_{template}/H_{template}$) can be specified. Due to the fact that it is almost impossible to achieve an exact aspect ratio, an acceptable deviation (Ear) can also be defined. The solution in the shape function will be selected according to the inequality: $AR - EarleqW_{template}/H_{template} \leq AR + Ear$

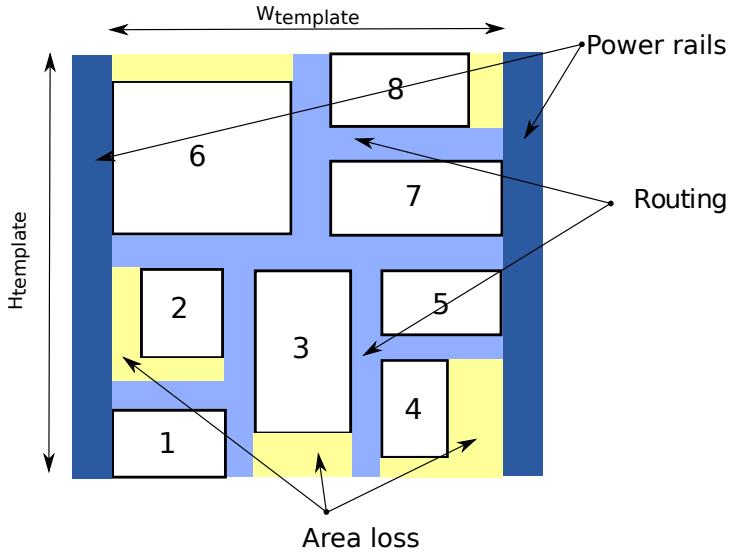

Fig. 2.7 *Area loss example.*

- **Width**. The designer can impose maximum and minimum allowable values of the circuit layout width ($W_{template}$). These values are defined by a predetermined value (W) and a margin (W_m). The solution have to satisfy the inequation: $W - W_m \leq W_{template} \leq W + W_m$
- **Height**. In the same way as the minimum and maximum Width, the designer can impose a predetermined height of the layout solution. $H - H_m \leq H_{template} \leq H + H_m$
- Simultaneous width and height limit values.

Mask generation, parasitic extraction, and electrical simulation

To implement the floor-plan sizing and thus generate all masks in order to extract parasitics and then evaluate the circuit performances through electrical simulation, the commercial platform *Cadence Design FrameWork II (DFWII)* [49] is used. Among many different functionalities, this design enviroment includes SKILL [50], a Lisp dialect used as scripting language and parameterized cells (pCell) description language. In addition, Cadence DFWII provides numerous SKILL-function libraries for different tasks, like “open a view type”, “do a Design Rules check (DRC)”, “open a file”, etc. To implement all actions within Cadence DFWII in an automated flow, a SKILL script has been developed, that also communicates Cadence DFWII with the optimization tool and *GCModule* (this will be explained in detail in Section 2.3.2). The different tasks carried out in Cadence DFWII are:

- **Mask generation using Parametrized Cells (pCells)**

  To automate the layout generation based on templates, a complete library of parametrized cells (differential pair, cascode structure, resistor, capacitor, etc) is needed. A pCell is a flexible layout generator that can be easily programmed and that accepts certain device values (e.g., W, L) and geometric parameters (e.g., number of fingers). A pCell can be as simple as a MOS transistor and as complex as an entire analog system. These pCells have been developed using the SKILL language. The key to SKILL's power is a large set of library functions that allows to manipulate data structures such as cells, nets, mask information, etc. pCells can be used hierarchically, meaning that a large pCell can contain other smaller pCells

- **Parasitic extraction**

  Parasitic extraction is the calculation of the parasitic elements in both the designed devices and the required wiring interconnects of an electronic circuit: detailed device parameters, parasitic capacitances, parasitic resistances and parasitic inductances. The major purpose of parasitic extraction is to create an accurate model of the real circuit, so that detailed simulations can emulate actual circuit responses.

  For each individual, the corresponding SKILL script proceeds as follows in batch-mode[b]: (1) it opens a database of the layout editor (*Virtuoso*); (2) it instances the layout template of the circuit under optimization, using an appropriate pCell, and its variables (coming from the optimizer and *GCModule*); (3) it flattens the layout (it is important to access to the pin names); (4) it extracts all parasitics and saves the extracted view in a new cell; (5) it closes the Virtuoso database.

  To implement step (4), there are parasitic extraction tools like *Diva* [51] and *Assura* [52] (although the use of another tool can be easily implemented using SKILL language). In this case, *Diva* has been used for two different reasons:

  1. Automating the parasitic extraction is easier with *Diva*, thanks to a single SKILL function for this purpose.
  2. In contrast to other tools it allows the parasitic extraction without DRC (*Design Rule Checking*) and LVS (*Layout Versus Schematic*) test, which represents a time saving (as instances of layout templates should be DRC and LVS-correct by construction).

[b]Batch-mode is the execution of a set of tasks in a program on a computer without manual intervention.

- **Electrical simulation**

  To evaluate the circuit compliance with respect to the optimization problem, the value of design objectives and constraints has to be calculated, that is, a simulation of the extracted view is required. For this analog simulation, we use OceanScripting within Cadence DFWII. OceanScripting allows choosing the simulator, the type of analysis required, and allows including SKILL code that permits automating different measures. Besides, it is possible to use it in batch mode, which is very important to automate the design flow.

  OceanScripting files can contain three different types of commands:

  - **Simulation Set-up**. First, the designer specifies the simulator. In this case, we have used Spectre because it works better with the design kit used in the demonstration part of this Thesis (any other available simulator can be use nonetheless).
  - **Simulator Run.** The declared electrical analyses are carried out.
  - **Data Access.** It is possible to access to the results as well as to perform any calculation on the results using SKILL. Finally, the output information (design objectives, constraints) can be saved to a file.

2.3.2 Communication between the optimizer module and the evaluator module

As it has been discussed in a previous section, there are different tools in this design flow. The integration and communication between these tools is key to automate the design flow. Figure 2.8 shows all tools and their communication using scripts and *pipes*.

To communicate Cadence DFWII with external tools (like the optimizer), an efficient communication protocol have been programmed using *InterProcess Communications (IPC)* SKILL functions. This communication protocol allows running the Layout-Aware flow in full batch mode to enhance automation. To start the flow, Cadence is initialized by using the non-graphic mode, which allows speeding up the layout-aware flow. After this, a single SKILL script is launched.

A diagram with the execution thread is shown in Figure 2.9. Cadence DFWII is the parent process, and it launches the optimizer (NSGA-II) as a child process, while the optimizer launches *GCModule* as a child process to parallelize the execution. While a solution is processed by Cadence DFWII (mask generation, extraction and simulation), this flow allows a parallel

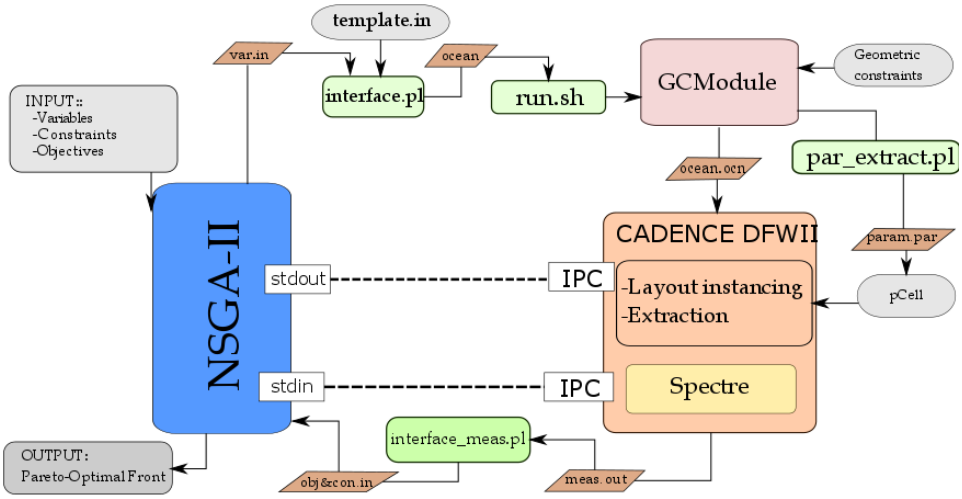

Fig. 2.8 *Full diagram flow of the POF-based Layout-Aware implementation.*

execution of the *GCModule* and these internal Cadence processes (that is, while a solution is being extracted and simulated, the *GCModule* is sizing the floor-plan of the next circuit).

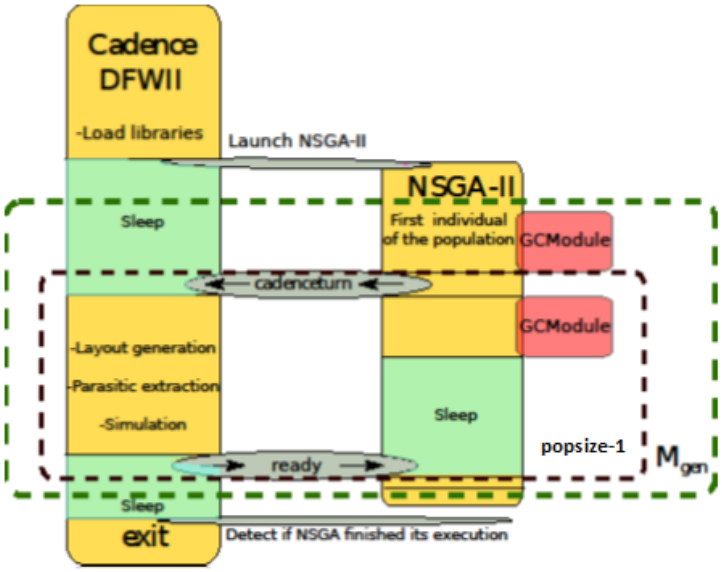

Fig. 2.9 *Diagram of execution and communication between Cadence and NSGA-II.*

This communication protocol is implemented in a SKILL script. The IPC functions allow the parent process (Cadence) to communicate with a child process by writing to the child process's *stdin* channel and reading from its *stdout* channel. This communication is synchronous, and it uses *busy-waiting* or *spinning*[c]. Cadence DFWII checks if a child process is still alive, that is, if the optimizer is running. If the optimizer has finished its execution (that all generations have been completed, or the Pareto-front has been found), Cadence is ended.

The scripts that implement the communication between tools were shown in figure 2.8. These scripts allow evaluating each individual from the values of their variables and the contents of the file *template.in* in which the information generated during the thread of execution is collected.

The file *template.in* is the main file in the process. All information about the circuit, the OceanScripting file and geometric constraints is in this file.

In summary, the problem is defined by a set of: variables, design objectives, design constraints, and geometric constraints. With this information, NSGA-II creates a random population formed by N_{ind} (*popsize*) individuals, and each individual is evaluated. For this, *GCModule* generates the values of its geometric parameters to minimize the area (attaining the geometric constraints). Within Cadence, the layout is generated and its parasitics are extracted before the electrical simulation of the circuit. NSGA-II uses selection, crossover and mutation operators to generate a new population across M_{gen} generations. At the end of the evolutionary process, the Pareto-optimal Front is returned. Therefore, the obtained designs are optimized geometrically and they are robust against the impact of parasitics, meaning that the optimized performances already consider the impact of parasitics and there is no need for entering a complex, non-systematic and time-consuming cycle of iterations to compensate for the damage of these layout parasitics.

Following the design flow, the scripts are:

1. The optimizer writes the value of variables in the file **var.in**.
2. The PERL script *interface.pl* adds the values of these variables into *template.in*. The resulting file is called **ocean**.
3. The shell script *run.sh* prepares the execution of the *GCModule*.
4. *GCModule* collects information on layout slicing structure (its binary tree) and the geometric constraints.

[c]The *busy-waiting* or *spinning* is a technique in which a process repeatedly checks to see if a condition is true, such as whether a keyboard input is available

First, *GCModule* reads the number of slices (horizontal divisions), and the number of basic blocks in each slice. Then, *GCModule* runs the modified version of the Stockmeyer algorithm, in which no horizontal divisions are allowed after a vertical division. Finally, it incorporates the values of the geometrical variables, creating a file called **ocean.ocn**.

5. The PERL script *par_extract.pl* gets the values of all variables and creates a file called **param.par** in order to instance the layout template with Virtuoso.

6. OceanScripting and Spectre perform the evaluation of the individual using the Ocean-Scripting file **ocean.ocn**.

7. The PERL script *interface_meas.pl* extracts the values of the design objectives and the design constraints from a file called **meas.out** written by the OceanScripting. This information is written in a file "obj&con.out" to be read by the optimizer.

2.4 Case studies

The circuit block selected to illustrate the POF-based Layout Aware Design Flow is an operational amplifier (op-amp). Op-amps are among the most widely used electronic circuits, being used in a vast array of consumer, industrial, and scientific devices. This is because using op-amps, it is easy to make amplifiers, filters, oscillators, data converters, and more. Mathematical functions like signal addition, subtraction, multiplication, and integration can be easily accomplished. The technological process used to demonstrate the POF-based Layout-Aware design methodology is a 0.35-μm, 4-metal, CMOS technology. However, this methodology can be applied to any modern technology. Different case studies are presented to show the advantages of the proposed methodology.

2.4.1 Circuit description

The schematic of the two-stage fully differential, RC Miller-compensated CMOS operational amplifier is shown in Figure 2.10. In order to stabilize the common-mode voltages at the output nodes a Common-mode Feedback Circuit *(CMFB)* is required. However, to simplify the optimization process and for the sake of illustration of the methodology, this CMFB circuit will be considered as ideal, as shown in Figure 2.11.

The circuit includes a Miller compensation with nulling resistor. The compensation is coupled between the output voltage and the cascode stage. Stability is improved by exploiting

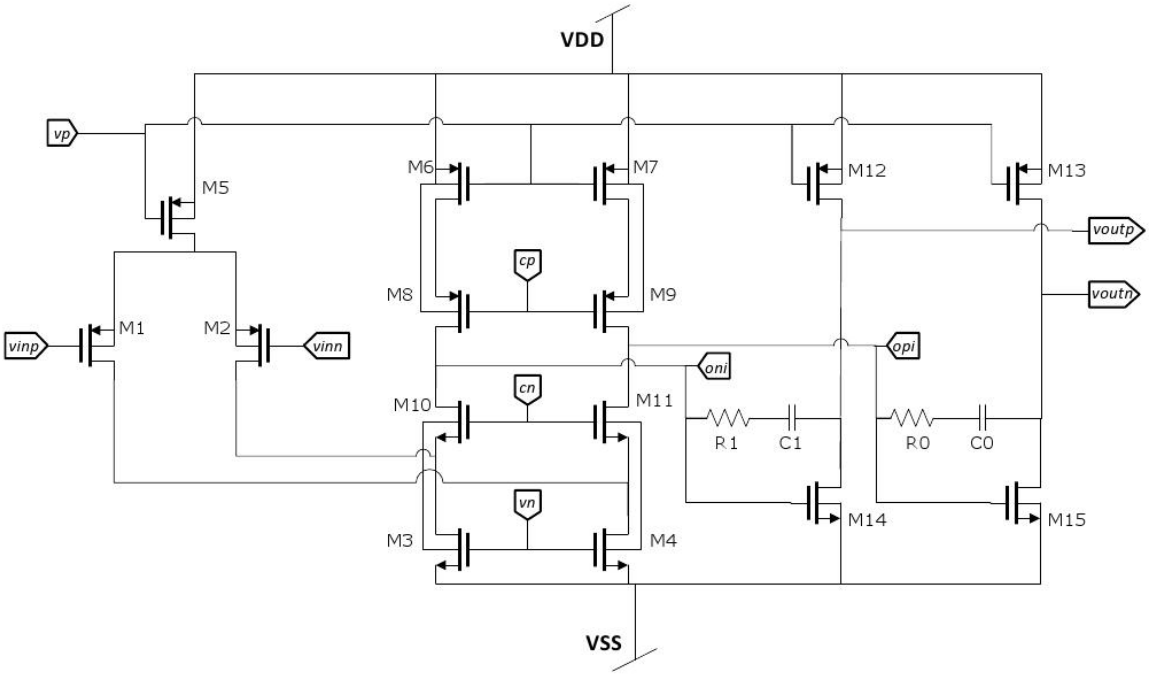

Fig. 2.10 *Schematic of the op-amp used in the optimizations.*

the pole splitting technique, that is, separating the dominant poles. However, the zero introduced by the Miller compensation can interfere, limiting the advantages of the higher frequency pole. To eliminate or move this zero, a nulling resistor is used.

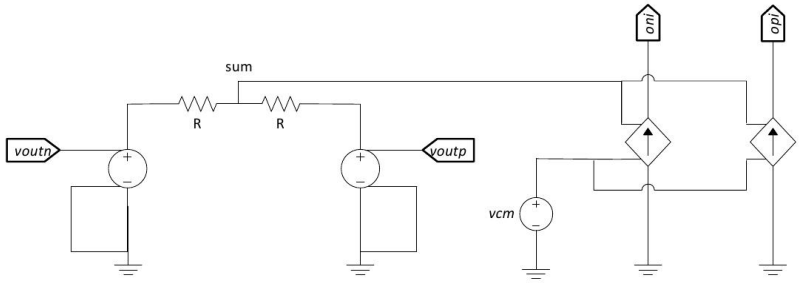

Fig. 2.11 *Ideal CMFB circuit.*

Circuit Layout template

As discussed previously, a *layout template* is used to speed up the layout generation required in the flow. The transistors are grouped in different basic blocks, as shown in Table 2.1. The placement is defined in a binary tree shown in figure 2.12.

Block	Device	Variables	Geometric parameter	Description
A	R_1	R_{sep}, R_{width}, Z_{valor}	zm	Folded Resistor
B	M_3, M_4, M_{10}, M_{11}	W_3, L_3, W_{10}, L_{10}	m3, m10	Cascode Structure
C	R_0	R_{sep}, R_{width}, Z_{valor}	zm	Folded Resistor
D	C_1	C_{valor}	cx	Unit Capacitor
E	M_{14}	W_{14}, L_{14}	m14	Folded Transistor
F	M_1, M_2	W_1, L_1	m1	Differential Pair
G	M_{15}	W_{14}, L_{14}	m14	Folded Transistor
H	C_0	C_{value}	cx	Unit Capacitor
I	M_6, M_7, M_8, M_9	W_6, L_6	m6	Cascode Structure
J	M_5, M_{12}, M_{13}	W_5, L_5, W_{12}, L_{12}	m5, m12	Current Mirror

Table 2.1 *Slicing tree and devices of the op-amp in figure 2.12.*

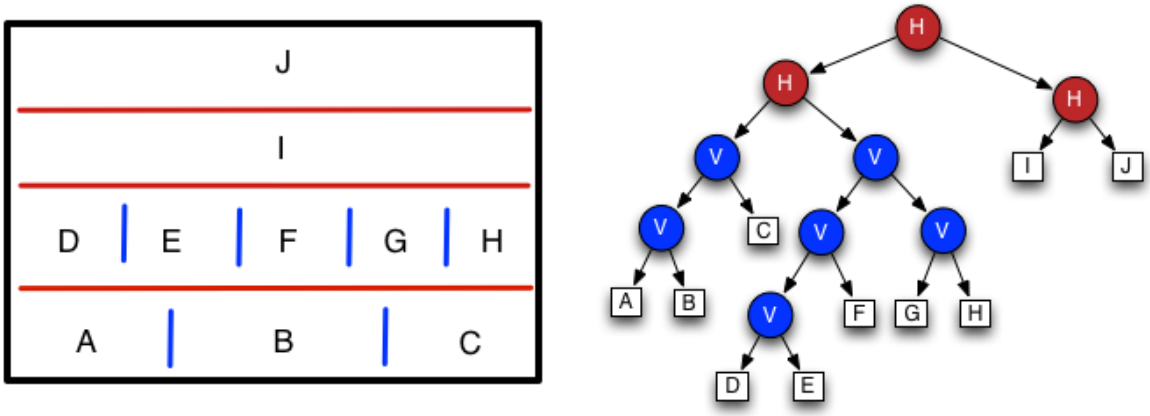

Fig. 2.12 *Binary tree of the layout template of the op-amp circuit.*

Each basic block is a pCell programmed in SKILL, and it has one or more geometric parameters (used to adjust its geometry), as shown in Table 2.1. The values of the variables of each basic block are proposed by NSGA-II and with this information, *GCModule* calculates the value of each geometric parameter. These values define the points of each shape function

(for each basic block) that form the solution of minimum area for the template. Finally, the placement of these basic blocks and routing are programmed in a pCell of the whole circuit. A block-by-block view of this template is shown in Figure 2.13

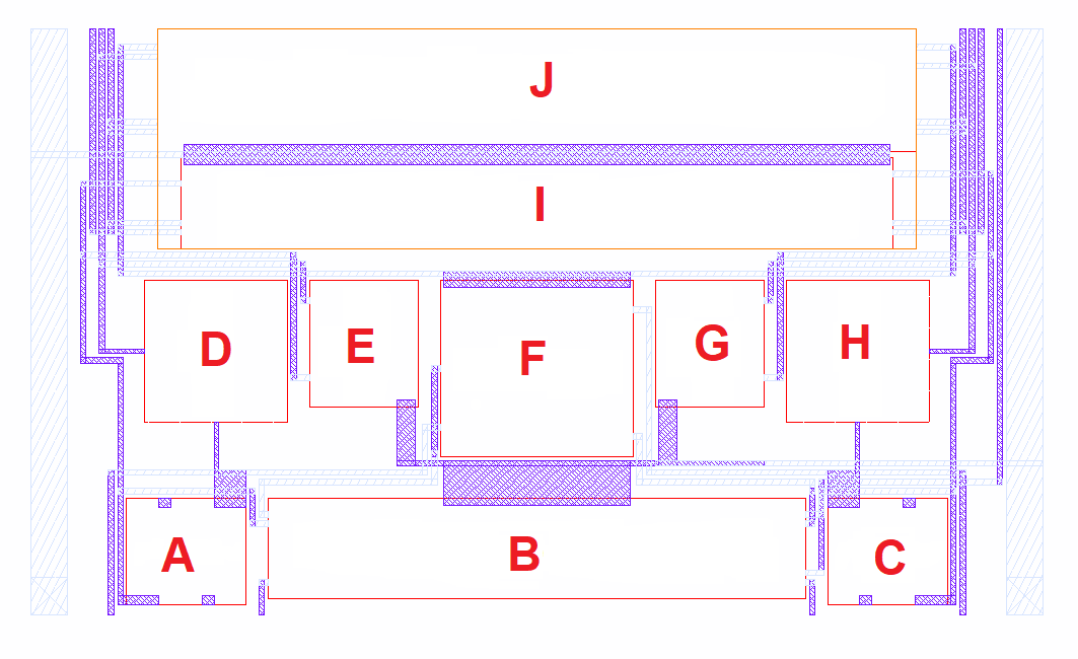

Fig. 2.13 *Layout template programmed in a pCell.*

In the optimization problem, the design variables are the transistors sizes, the bias current and the values of the compensation capacitor and nulling resistor. For the resistor, two more variables are defined: the width of the strips, and the separation between each strip. Figure 2.14 shows a resistor to illustrate these variables. The geometric parameter used by *GCModule* to minimize the area is the number of strips that form the resistor.

The size of the transistors Mbn and Mbp, used to bias the opa-amp have been added as design variables. All variables and their lower and upper bounds are detailed in Table 2.2. Due to the symmetry of the circuit topology, some of these variables are related. In addition, the value for each transistor's length is fixed, so that only 11 of the 40 variables are independent.

To ensure that transistor M10 and M11 still operate in the saturation region for unbalanced operation of the input differential pair, the condition

$$I_6 > I_{bias}/2 \Rightarrow I_6 > (0.5+k)\cdot I_{bias} \tag{2.1}$$

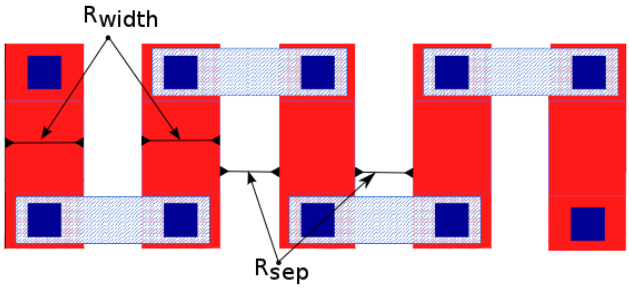

Fig. 2.14 *Variables defined for resistors R1 and R0 (blocks A and C).*

where $k \cdot I_{bias}$ is the minimum current through these transistor to ensure saturation operation. Typically $k = 0.1$, so this relationship is translated into a relationship between the width of the transistor.

The design variable α is related to the overdrive voltage of transistors M14 and M15.

$$I_{M14} = I_{M15} = \frac{\mu C_{ox}}{2 \cdot \alpha} \frac{W_{15}}{L_{15}} \quad \text{,where} \quad \alpha = \frac{1}{(V_{gs} - V_{th})^2} \tag{2.2}$$

The variable α is used to maintain a constant current density flowing through these output transistors. In this way, their overdrive voltage is constrained to vary over a range such that points of the design space featuring better output swing are favored.

Performance characteristics

A test-bench circuit is needed to simulate the op-amp. The circuit used is shown in figure 2.15. The supply voltages of the op-amp are VDD $(1.65V)$ and $VSS(-1.65V)$, and it is biased with current source *ib*. Cascode transistors are biased with *cn* and *cp* voltages to nMOS (M10 and M11) and pMOS (M8 and M9) respectively.

To carry out the different measurements of the op-amp performances, two analyses need to be carried out: an operating point analysis (op) and an AC analysis. The load is formed by the parallel combination of a $10k\Omega$ resistor and a $1pF$ capacitor.

The value of the following parameters are used as objectives or constraints in the optimization problem:

Name	Min. Value	Max. Value	Relationship	Name	Min. value	Max. value	Relationship
W_1	$2\cdot10^{-6}$	$800\cdot10^{-6}$		L_1	$1\cdot10^{-6}$	$2\cdot10^{-6}$	$=1.2\cdot10^{-6}$
W_2	$2\cdot10^{-6}$	$800\cdot10^{-6}$	$=W_1$	L_2	$1\cdot10^{-6}$	$2\cdot10^{-6}$	$=L_1$
W_3	$1\cdot10^{-6}$	$800\cdot10^{-6}$		L_3	$1\cdot10^{-6}$	$2\cdot10^{-6}$	$=L_{bn}$
W_4	$1\cdot10^{-6}$	$800\cdot10^{-6}$	$=W_3$	L_4	$1\cdot10^{-6}$	$2\cdot10^{-6}$	$=L_{bn}$
W_5	$2\cdot10^{-6}$	$800\cdot10^{-6}$		L_5	$1\cdot10^{-6}$	$2\cdot10^{-6}$	$=L_{bn}$
W_6	$1.2\cdot10^{-6}$	$480\cdot10^{-6}$	$=0.6\cdot W_5$	L_6	$1\cdot10^{-6}$	$2\cdot10^{-6}$	$=L_{bn}$
W_7	$1.2\cdot10^{-6}$	$480\cdot10^{-6}$	$=W_6$	L_7	$1\cdot10^{-6}$	$2\cdot10^{-6}$	$=L_{bn}$
W_8	$1.2\cdot10^{-6}$	$480\cdot10^{-6}$	$=W_6$	L_8	$1\cdot10^{-6}$	$2\cdot10^{-6}$	$=L_{bn}$
W_9	$1.2\cdot10^{-6}$	$480\cdot10^{-6}$	$=W_6$	L_9	$1\cdot10^{-6}$	$2\cdot10^{-6}$	$=L_{bn}$
W_{10}	$1\cdot10^{-6}$	$500\cdot10^{-6}$		L_{10}	$0.35\cdot10^{-6}$	$2\cdot10^{-6}$	$=0.55\cdot10^{-6}$
W_{11}	$1\cdot10^{-6}$	$500\cdot10^{-6}$	$=W_{10}$	L_{11}	$0.35\cdot10^{-6}$	$2\cdot10^{-6}$	$=L_{10}$
W_{12}	$2\cdot10^{-6}$	$800\cdot10^{-6}$		L_{12}	$1\cdot10^{-6}$	$2\cdot10^{-6}$	$=L_{bn}$
W_{13}	$2\cdot10^{-6}$	$800\cdot10^{-6}$	$=W_{12}$	L_{13}	$1\cdot10^{-6}$	$2\cdot10^{-6}$	$=L_{bn}$
W_{14}	$2\cdot10^{-6}$	$800\cdot10^{-6}$	$=\alpha\cdot ib\cdot L_{14}\frac{W_{12}}{W_5}/50\cdot10^{-6}$	L_{14}	$0.35\cdot10^{-6}$	$2\cdot10^{-6}$	$=0.35\cdot10^{-6}$
W_{15}	$2\cdot10^{-6}$	$800\cdot10^{-6}$	$=W_{14}$	L_{15}	$0.35\cdot10^{-6}$	$2\cdot10^{-6}$	$=L_{14}$
W_{bp}	$2\cdot10^{-6}$	$800\cdot10^{-6}$	$=W_5$	L_{bn}	$1\cdot10^{-6}$	$2\cdot10^{-6}$	$=1.2\cdot10^{-6}$
W_{bn}	$1.8182\cdot10^{-6}$	$727.28\cdot10^{-6}$	$=0.9091\cdot W_3$	L_{bp}	$1\cdot10^{-6}$	$2\cdot10^{-6}$	$=L_{bn}$
R_{value}	500	5000		C_{value}	$0.07\cdot10^{-9}$	$2\cdot10^{-9}$	
R_{sep}	$0.6\cdot10^{-6}$	$3\cdot10^{-6}$		ib	$2\cdot10^{-6}$	$1\cdot10^{-3}$	
R_{width}	$1.8\cdot10^{-6}$	$5\cdot10^{-6}$		α	1	200	

Table 2.2 *Design variables.*

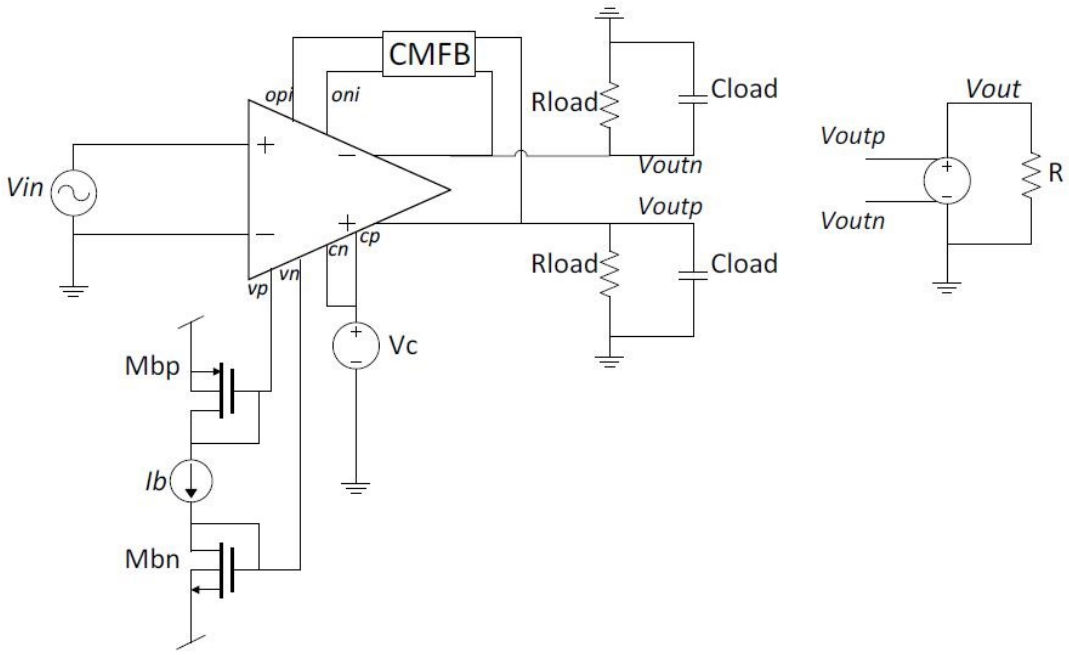

Fig. 2.15 *Test bench circuit.*

- *os*. Output Swing. Being a differential circuit, the output swing ican be approximately calculated as:

$$\mathbf{os} = 2 * (vdd - vss - |vp - vdd - vthp| - |opi - vss - vthn|) \tag{2.3}$$

- *dmX*. It is used to ensure that transistor X is saturated. Its formulation is as follows:

$$\mathbf{dmX} = \frac{V_{DS}}{V_{DS_{sat}}} = \frac{V_D - V_S}{V_G - V_S - V_{th}} \tag{2.4}$$

- *power consumption*.
- *DC-gain*.
- *fu*. Unity-gain frequency.
- *pm*. Phase margin.

- *diff*. It is defined as the difference between *gain-bandwidth product* and *fu*. It is used to ensure that the zero introduced by the Miller compensation is not inserted between the dominant poles.

Finally, other parameters, which, strictly speaking, do not require or do not use in this case an electrical simulation, can be provided to the optimizer. These are:

- *Atemplate*. It is the area occupied by the layout, which is calculated by *GCModule*.
- *Alost*. It is the percentage of the area that is not used by the devices or the routing lines (*Aloss*), which is also calculated by *GCModule*.

$$\textbf{Alost} = \frac{Aloss}{Atemplate} \cdot 100 \tag{2.5}$$

- *SR*. Slew Rate. To calculate the slew rate, there are two options: a transient analysis or the use of equations. In this case, the second option is used because a transient analysis would have to be repeated a many times in the optimization (number of individuals times number of generations) and it is an expensive analysis. In this case the minimum value between the internal slew rate (dependent of the miller capacitor) and external slew rate (dependent of the load capacitor) is calculated as:

$$\begin{aligned} SR_{internal} &= \frac{ib}{C_{miller}}, \\ SR_{external} &= (\frac{W_{12} \cdot L_{15}}{L_{12} \cdot W_5} - 1) \cdot \frac{ib}{C_{load}} \\ \textbf{SR} &= minimum(SR_{internal}, SR_{external}) \end{aligned}$$

Some of these measurements can be used, in the multi-objective optimization problem, both as a design objective and as a constraint. However, some other measurements, which are needed to ensure the correct operation of the op-amp, can only be posed as constraints and not as objectives since their minimization or maximization does not make sense. These constraints, as well as their thresholds, are detailed in Table 2.3.

Table 2.4 shows the design objectives together with the optimization direction and which is their specification when they are used as an additional constraint.

Using the circuit described in this section, three cases studies are presented in this Chapter in order to compare the impact of the parasitics. In case study 1, a non-Layout-Aware (non-LA) and a Geometric-Layout-Aware (G-LA) design flows are presented. In a non-LA design flow,

Name	Constraint	Explanation
pm	$> 60°$	To ensure circuit stability (can be used as an objective)
dm1	> 1.1	To ensure that the transistors M1 and M2 are in saturation region
dm3	> 1.1	To ensure that the transistors M3 and M4 are in saturation region
dm5	> 1.1	To ensure that the transistor M5 is in saturation region
dm6	> 1.1	To ensure that the transistors M6 and M7 are in saturation region
dm8	> 1.1	To ensure that the transistors M8 and M9 are in saturation region
dm10	> 1.1	To ensure that the transistors M10 and M11 are in saturation region
dm12	> 1.1	To ensure that the transistors M12 and M13 are in saturation region
dm14	> 1.1	To ensure that the transistors M14 and M15 are in saturation region
diff	> 0	To avoid that there are any zeros between the dominantes poles

Table 2.3 *Constraints.*

Name	Optimization	Constraint
area_template	*to minimize*	$<$ *area_maxima*
power	*minimize*	$<2.5mW$
a0	*maximize*	$>100dB$
fu	*maximize*	$>80MHz$
os	*maximize*	$>4.0V$
sr	*maximize*	$>55V/\mu s$

Table 2.4 *Design objectives or additional constraints.*

no information about physical implementation is used while in a G-LA design flow, the physical implementation is considered. That is, G-LA allows to include accurate information about the diffusion capacitors in each transistor, due to the implementation of each transistor is known. A second case study is presented, in which the complete Layout-Aware (LA) design flow is used, that is, in addition to the parasitic devices considered in G-LA, the parasitic devices due to the routing are also included during the optimization process. Finally, a third case study is presented using different constraints on the total occupied area.

The Pareto-optimal fronts in these case studies have been generated using 100 individuals evolving across 220 generations; these values were set heuristically, by simply checking that no significant evolution took place after such number of generations.

2.4.2 Case study 1: Non Layout-aware versus Geometric-Layout-Aware

The first case study intends to compare the results using the proposed Geometric-Layout-Aware flow with a traditional, non-Layout-Aware flow depicted in figure 2.16. As in the design flow explained in this chapter, the operation of this flow is based on an iterative loop involving an optimizer and an evaluator. In this case, it is not necessary any information about layout so the design flow is simpler than the one implemented in this Thesis.

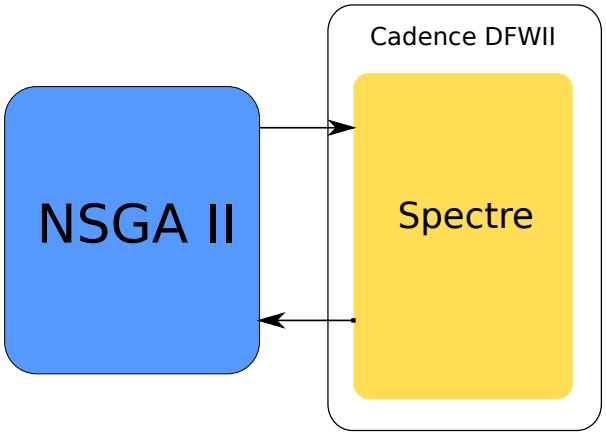

Fig. 2.16 *Traditional optimization flow*

The electrical synthesis (or traditional optimization) has been implemented using NSGA-II and Spectre (within Cadence DFWII) as optimizer and evaluator respectively. In this process, diffusion capacitors are the only information about parasitics that is added, but the transistors are implemented without multiplicity, that is, using only one finger.

As this optimization flow does not take into account accurate information on the layout implementation (as the Layout Aware Design Flow does), it is not possible to know the area in the physical implementation of the circuit. For this reason, it is necessary to use a rough estimation of the area. For this, it is assumed that the occupied area can be, as a first approximation, proportional to the area of the devices. To calculate this area, first, the area of the gate of all transistors is calculated using equation 2.6 and the values for the area occupied by the strips of the resistors and square capacitors are calculated using

$$\begin{aligned} \text{Technological parameters} \quad & R_{sq} = 50\Omega \;\; W_{eff} = 0.35\mu m \\ & C_a = 0.86 \cdot 10^{-3} pF \;\; C_{pp} = 0.092 \cdot 10^{-3} pF \end{aligned}$$

$$\begin{aligned} \text{area}_{tr} &= 2 \cdot W_1 L_1 + 2 \cdot W_3 L_3 + W_5 L_5 + 4 \cdot W_6 L_6 \\ &\quad + 2 \cdot W_{10} L_{10} + 2 \cdot W_{12} L_{12} + 2 \cdot W_{14} L_{14} \qquad (2.6) \\ \text{area}_{res} &= R_{width} \cdot R_{value} \cdot (R_{width} - W_{eff}) R_{sq} \qquad (2.7) \\ \text{area}_{cap} &= [(-4 \cdot C_{pp} + \sqrt{16 \cdot C_{pp}^2 + 4 \cdot C_a C_{value}}\,)/(2 \cdot C_a)]^2 \qquad (2.8) \end{aligned}$$

Finally, the total are is calculated as

$$\textbf{Area} = \text{area}_{tr} + \text{area}_{res} + \text{area}_{con} \qquad (2.9)$$

Using this non Layout Aware (non-LA) multi-objective optimization, a case study is presented, in which, the optimizer has been configured to maximize the DC-gain and the unity-gain frequency (fu) and minimize the area. The additional constraints that have been used in this optimization are the output swing ($os > 4.0V$) and the slew rate ($sr > 55V/\mu s$).

After the optimization, the final population forms a Pareto-optimal front because all individuals are non-dominated and they all fulfill constraints. Figure 2.17 shows this population.

If a designer wants to use a circuit design from this POF, the circuit layout needs to be created. Once this is done, the impact of parasitics could be added. However, as it will be shown later, this could mean carrying out several time-consuming iterations to correct the non-desired perturbations. Before illustrating the complete POF-based Layout Aware design flow, an intermediate case study is presented (which will also serve as baseline to contrast the results of the non-LA solution). This case study only considers the geometry aspects of the final layout but not the extracted parasitics due to routing.

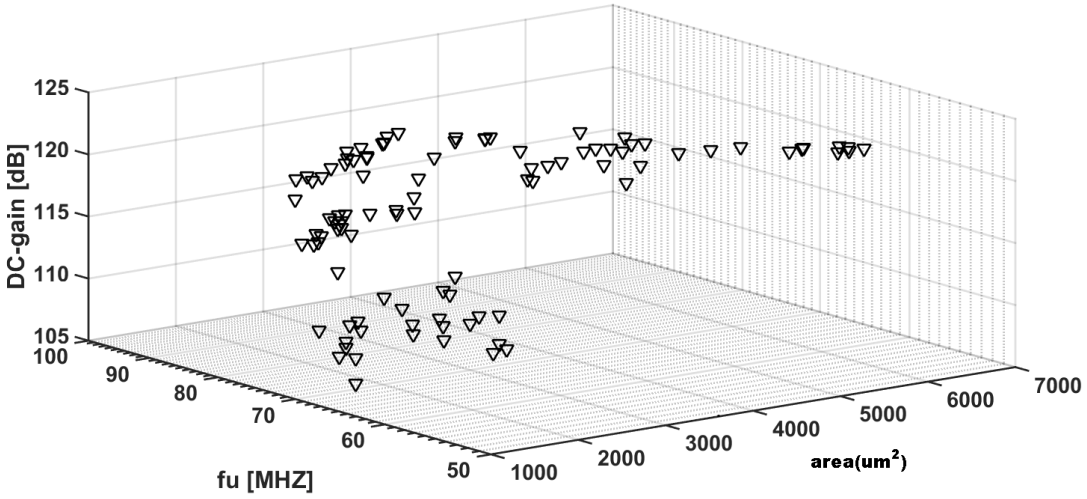

Fig. 2.17 *POF generated with a non-LA optimization.*

This flow is called Geometric-Layout-Aware (G-LA) design flow, in which, only the diffusion parasitics are included, albeit with a difference to the non-LA flow: the diffusion parasitics are better calculated since there is precise information on the actual geometry of, for instance, folding of transistors. The use of layout templates and the modified Stockmeyer's algorithm allows to define geometric constraints in the geometry-aware sizing. With this case study, the goal is to show that electrical sizing together with geometry optimization (i.e., floorplan sizing) is a first step towards full layout-aware (LA) (i.e., geometry + full parasitics) sizing.

As a side note, it is worth mentioning an aspect related to the lack of flexibility of templates. The Geometric-Layout-Aware (G-LA) sizing complements the use of the template-based approach with a way to improve that flexibility by finding the most adequate values of the geometric parameters to improve area and area usage. These effects are clearly shown in Figure 2.18. The figure depicts two layouts (generated using *GCModule*) with the same design variables but different aspect ratios and a layout solution also from the same template with obvious quality differences (the layout on the left occupies much less area and have less area loss). This means that for a layout-aware sizing solution (always requiring a fast and high-quality layout generation method), a floorplan-sizing technique is mandatory if the "lack of flexibility" is to be alleviated. Next chapters will show an additional method to further improve the way layout templates are used in the LA design flow.

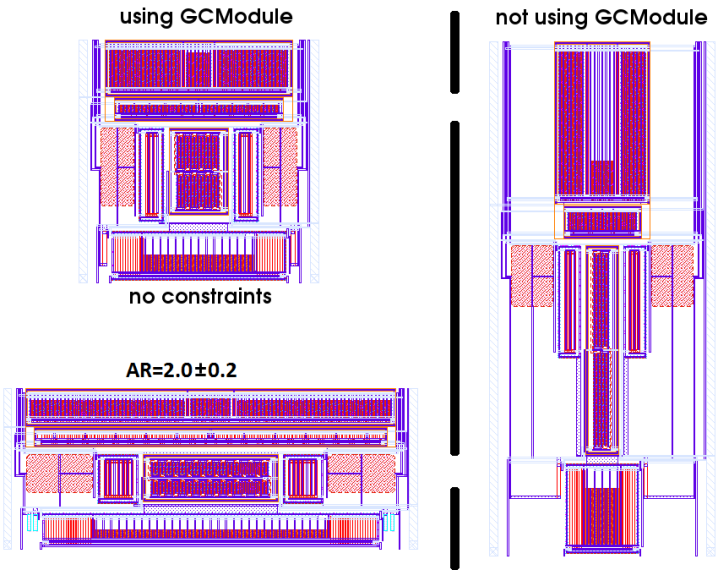

Fig. 2.18 *G-LA sizing examples.*

Using the G-LA sizing, a new POF has been generated with the same design objectives and constraints than in the previous non-LA optimization. The difference from the previous optimization is that the area occupied can be minimized accurately because the physical synthesis is included in the optimization process. In addition, the layout geometry is limited using a geometric constraint for the aspect ratio. In this case, the chosen value is 1.0 with a margin of error of 20% (approximately square geometries). The final population is shown in Figure 2.19

Different examples of individuals which form the POF are shown in Figure 2.20. It is worth mentioning that the area loss (*Alost*) in this case is very small.

To compare this result with the POF obtained previously using a non-LA optimization, the physical implementation of each circuit design obtained in non-LA POF would have to be generated. For this reason, to see what happens if floorplan sizing is carried out for the design whose geometry was not optimized during the electrical sizing, we have used *GCModule* on these individuals and then, they have been simulated. In addition, *GCModule* has been configured to generate the layout with the same geometric constraint than in G-LA POF, that is, AR$\sim$ 1.

The results obtained are shown in Figure 2.21 and projections of the results onto the 2D objectives planes are shown in Figure 2.22 to ease its visualization. The area increases noticeably, but this happens because the area used in non-LA optimization is only the area of

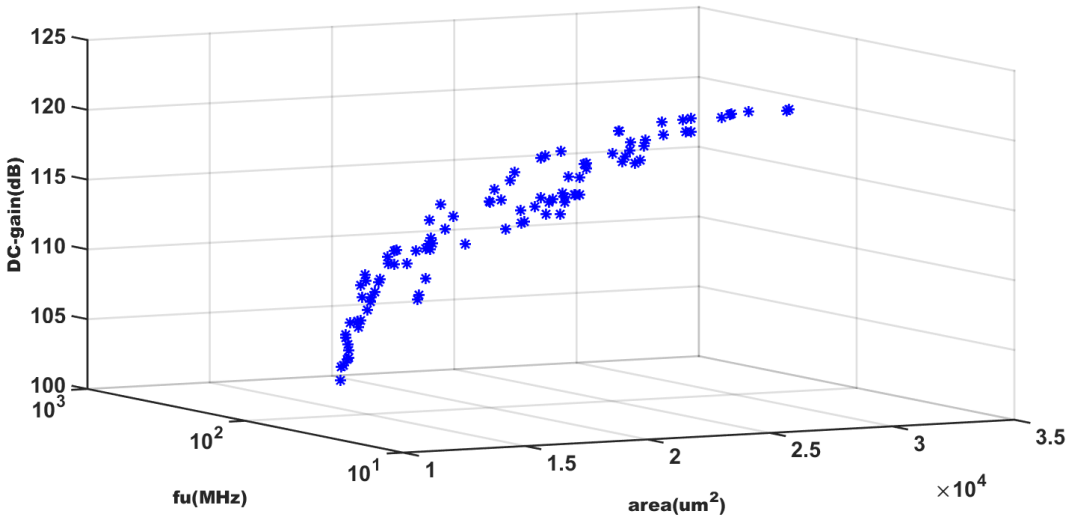

Fig. 2.19 *G-LA POF with AR*∼ 1.0.

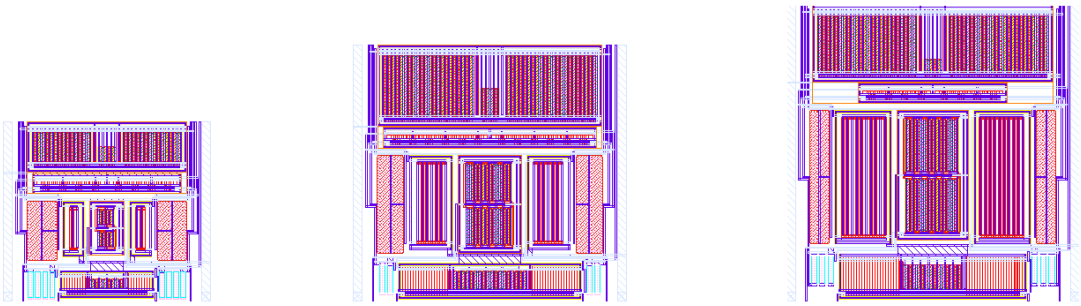

Fig. 2.20 *Examples with AR*∼ 1.0

the devices, and the area used in individuals plotted in figures 2.21 and 2.22 includes the actual layout implementation.

In addition, we can see that the performance in DC-gain and unity gain frequency are slightly better. This improvement comes from the fact the transistors have been folded, which may reduce the diffusion capacitances as well as modify the diffusion resistances (in the traditional optimization, the multiplicities that were used were all equal to 1, which yield very large

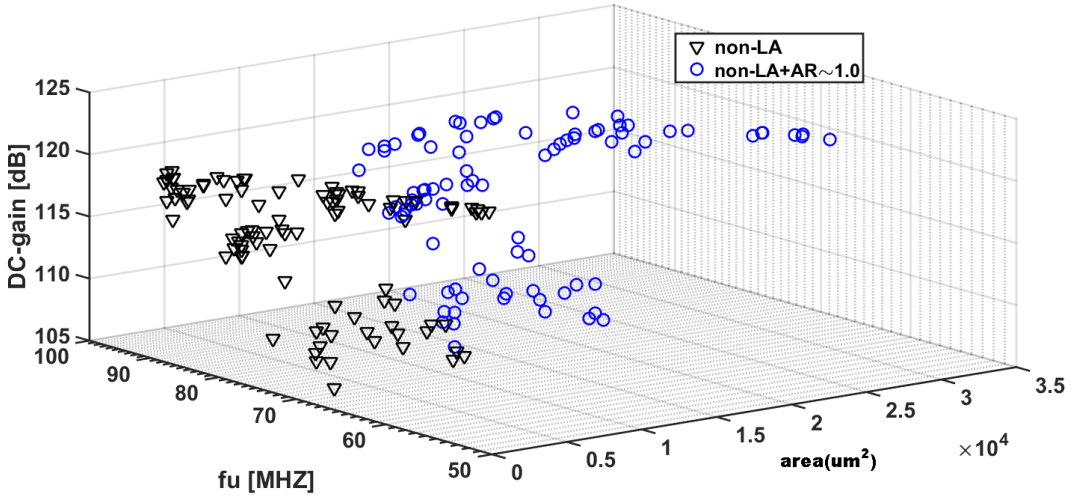

Fig. 2.21 *non-LA-POF and simulation using GCModule with AR∼* 1.0

diffusion regions for large transistors). For the sake of illustration, the following formula provides the value of the diffusion capacitor when a transistor is folded in m fingers:

$$C_{folded} = C^{A}_{before}\left(\frac{m+1}{2m}\right) + C^{P}_{before}\left(\frac{mK+\frac{W}{m}+1}{W+2K}\right) \tag{2.10}$$

;where:

C_{folded}: capacitance after doing folding

C_{before}: capacitance before doing folding

C^{A}: capacitance per unit area

C^{P}: capacitance per unit perimeter

m: multiplicity

W: full area of the transistor

K: default width of the diffussion region, specified by the technology

Note from equation 2.10 that, if we ignore the perimeter, and use a large value of m, the capacitance can be halved. But although the circuit performances seem to improve when the physical implementation is generated, one important aspect deserve consideration here: all individuals fail to complete all or some constraints, which makes this non-LA POF inadequate to cope with the goal of avoiding iterations between electrical and physical design phases. This means that all individuals in this front should be redesigned in order to meet all constraints.

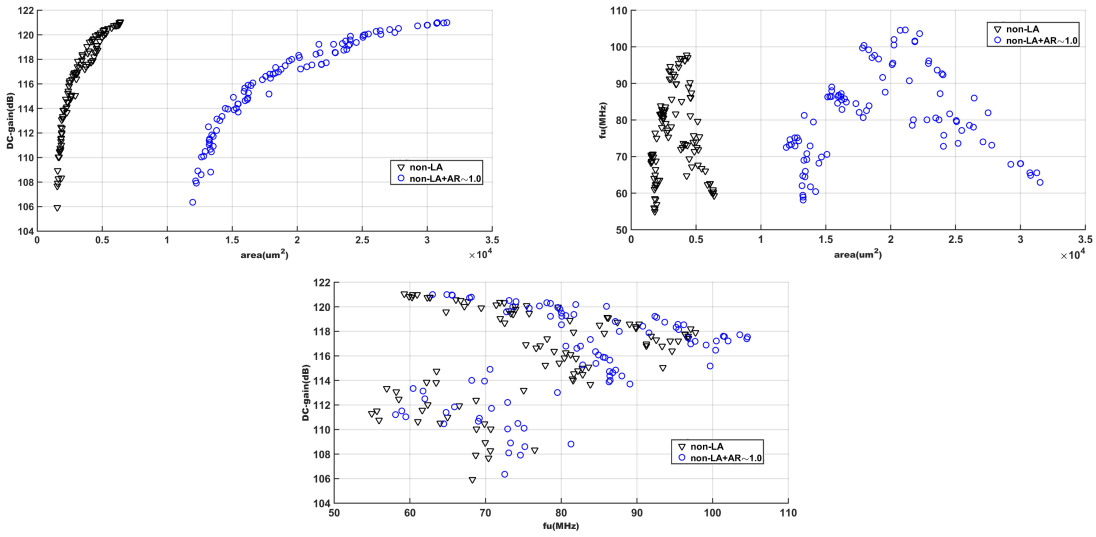

Fig. 2.22 *Non-LA POF+(G-LA). Projections of the objectives on 2D planes.*

2.4.3 Case study 2: Geometric-Layout-Aware and Layout-Aware

In the previous case study, a Geometric-Layout-Aware (G-LA) POF has been used to show the impact of the physical implementation into the design flow. For this, a geometric constraint over the aspect ratio was used. In this section, a new group of case studies are shown in which different values for the aspect ratio are considered as geometric constraints to demonstrate that:

- geometric optimization is possible;
- different geometric constraints may have an impact on the electrical performance
- the template used (with its placement and routing) can have an impact on the quality of the solutions.

The design objectives and constraints are the same as in the previous POF generated using the geometric constraint AR$\sim$ 1. In this case, two additional G-LA POF have been generated using two different values for the aspect ratio: 0.25 and 4.0, with a margin of error of 20%.

The three final populations are shown in Figure 2.23. As it can be seen, using the same design objectives and constraints but different geometric constraints, the Pareto fronts obtained are very different.

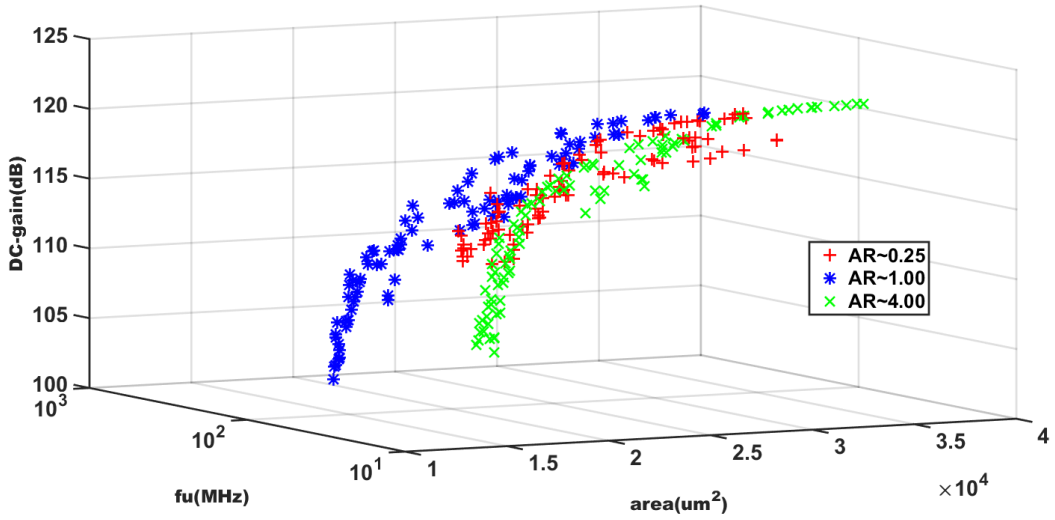

Fig. 2.23 *G-LA POFs generated with different values of AR*

To better visualize the impact of the geometric constraint *AR* on the circuit performances, all possible projections of the fronts onto 2D objective planes are shown in Figure 2.24. In the DC-gain/fu projection, we can see that populations have a similar performance but the population with higher aspect ratio (~ 4) does not get high values for unity-gain frequency. This is due to the fact that transistors implemented in the individuals of this population have a very high value of the multiplicity, which makes the diffusion parasitic capacitors of these transistors larger. On the other hand, individuals in population with a square layout ($AR \sim 1$) have a much smaller area than individuals in other populations. In addition, we can note that *Alost* is much larger in layouts with aspect ratio ~ 0.25 and ~ 4 than in layouts with aspect ratio ~ 1, which are illustrated in Figure 2.19. For this, different examples of individuals in each new population are shown in Figures 2.25 and 2.26, using an aspect ratio 0.25 and 4.00 respectively.

In view of the results obtained, it is demonstrated that the use of the Stockmeyer's algorithm gives provides a certain degree of flexibility to the layout template. Moreover, using this G-LA sizing technique, it is quite straightforward to note that an optimal geometry (in this case an

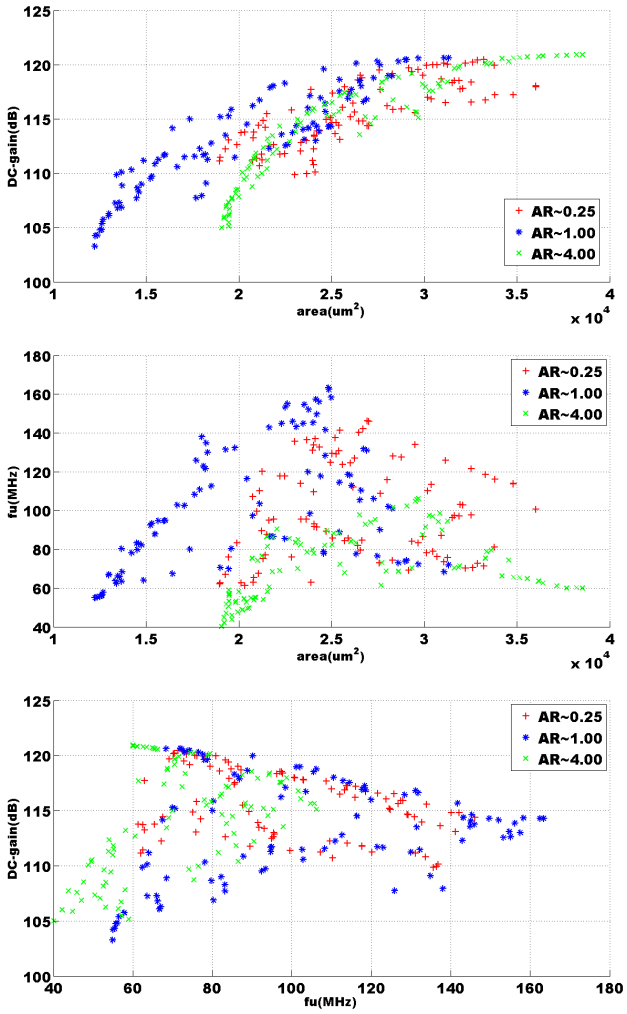

Fig. 2.24 *Geometry-Aware POF generated with different AR. Projections of the objectives onto 2D planes.*

aspect ratio close to 1) can make an impact on the circuit performance, due to the strong link between geometry (e.g., number of fingers) and parasitics. This is why any Layout-Aware sizing solution has to deal with both geometry and sizing at the same time. In addition, to evaluate the impact of all parasitic devices, a new group of three POFs have been generated using the full design methodology implemented. These new POFs are called Layout-Aware (LA) POF since the individuals that form them have been evaluated with full Layout-Aware design flow, that is, with full set of parasitics devices (both from diffusion and routing).

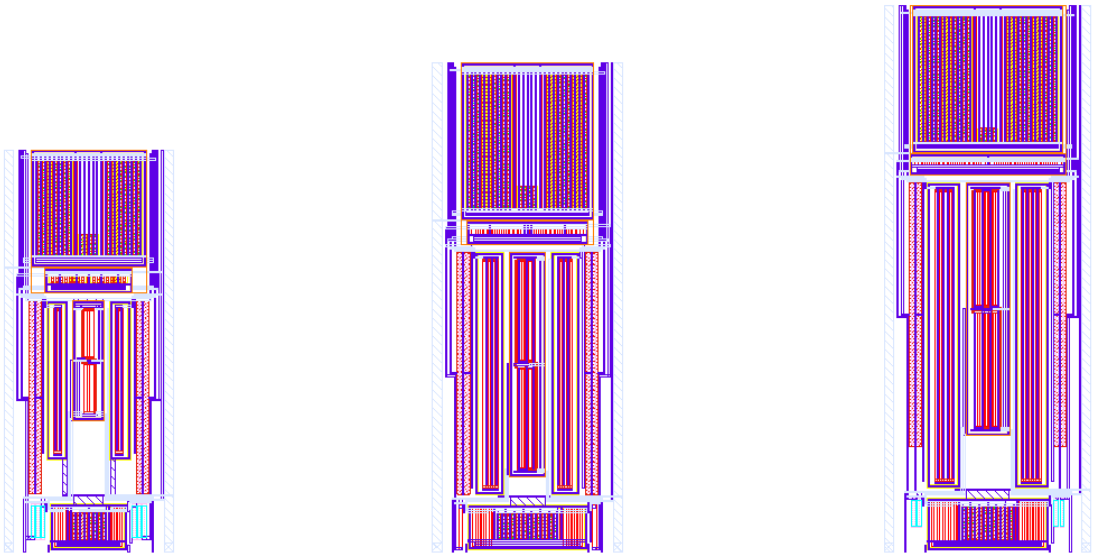

Fig. 2.25 *Examples with AR*∼ 0.25

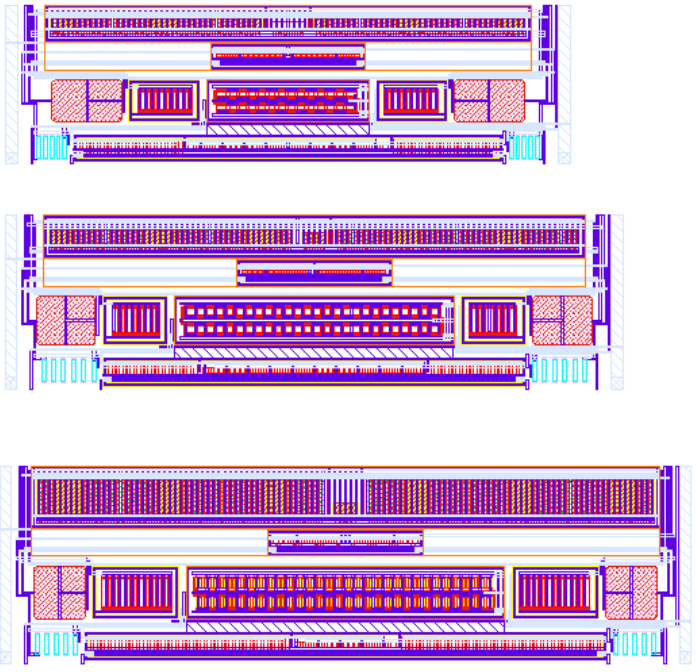

Fig. 2.26 *Examples with AR*∼ 4.00

The three final populations obtained are shown in figure 2.27, corresponding to aspect ratios 0.25, 1.0 and 4.0, with a margin of 20%.

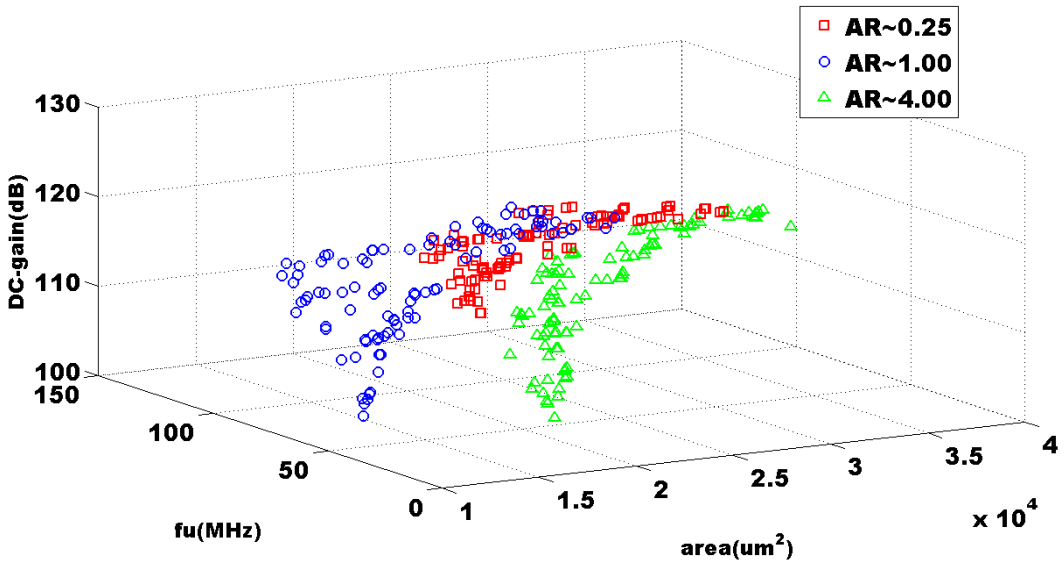

Fig. 2.27 *LA POFs using different values of AR.*

These individuals have been optimized with all the information about parasitics, which means that the performance of each circuit will not be degraded after simulating the extracted layout because of parasitics. As in the G-LA POFs, we can see that different geometries have an impact on electrical performance. In the DC-gain/fu projection, shown in Figure 2.24, the populations have a similar performance but the POF with higher aspect ratio (~ 4) has again much less unity gain frequency, and the POF with aspect ratio ~ 1 has again a much smaller area than the others.

To compare these results with the obtained using geometric-aware sizing and to evaluate the impact of routing parasitic, we will check each different geometric constraint case separately. For this, the figures have 3 different populations: (1) LA, population obtained adding all parasitics in its evaluation, that is, in a LA POF; (2) G-LA, population obtained in a Geometric-Layout-Aware POF; (3) G-LA+parasitic, in which the populations from the G-LA POF are re-simulated to include the routing parasitics. To improve the visualization of the differences between these three cases, all possible projections on 2D objectives planes are shown. Each POF generated with different AR is discussed:

- **AR=1.0$\pm$0.2** (Figure 2.28).

Individuals again occupy approximately the same area, the DC-gain is a little better for LA individuals but the unity gain frequency is lower (about $10MHz$). When adding routing parasitics, the individuals obtained in a G-LA POF do not attain the design constraints.

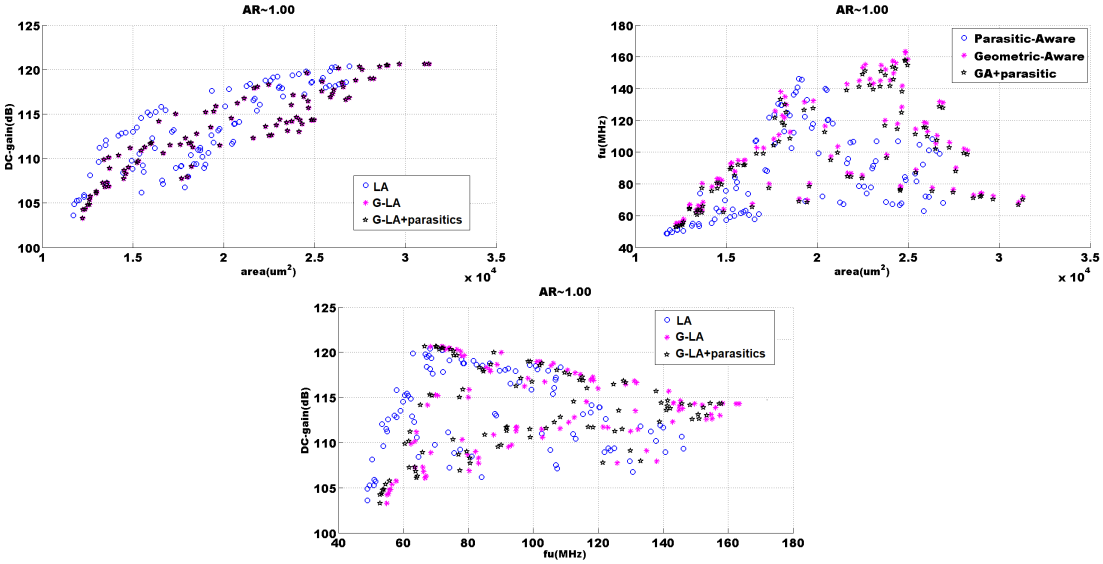

Fig. 2.28 *Comparison G-LA vs LA. AR=1.0*

- **AR=0.25±0.05** (Figure 2.29).

  In this case, individuals that occupy approximately the same area (in G-LA and LA POFs) have different values in the other objectives. The inclusion of routing parasitics in the optimization loop causes that the LA individuals obtained have a bit more DC-gain ($1dB$ or $2dB$) although less unity gain frequency (until $30MHz$) than individuals obtained without this information (G-LA). Besides, for G-LA individuals, their performance is slightly degraded, in DC-gain (about $1dB$), but significantly in fu (on average $2.5MHz$) when adding parasitics. But more important than this, all individuals in the re-simulated population (G-LA+parasitics) become inadequate because they do not attain the design constraints.

- **AR=4.0±0.8** (Figure 2.30).

  In this case, LA individuals occupy more area (about $1000um^2$) for the same values of DC-gain while the unity gain frequency is approximately the same. Therefore, performance on this front is almost the same in DC-gain and fu, but with more area. Using this aspect ratio, the number of fingers in transistors is very high, which means high values for

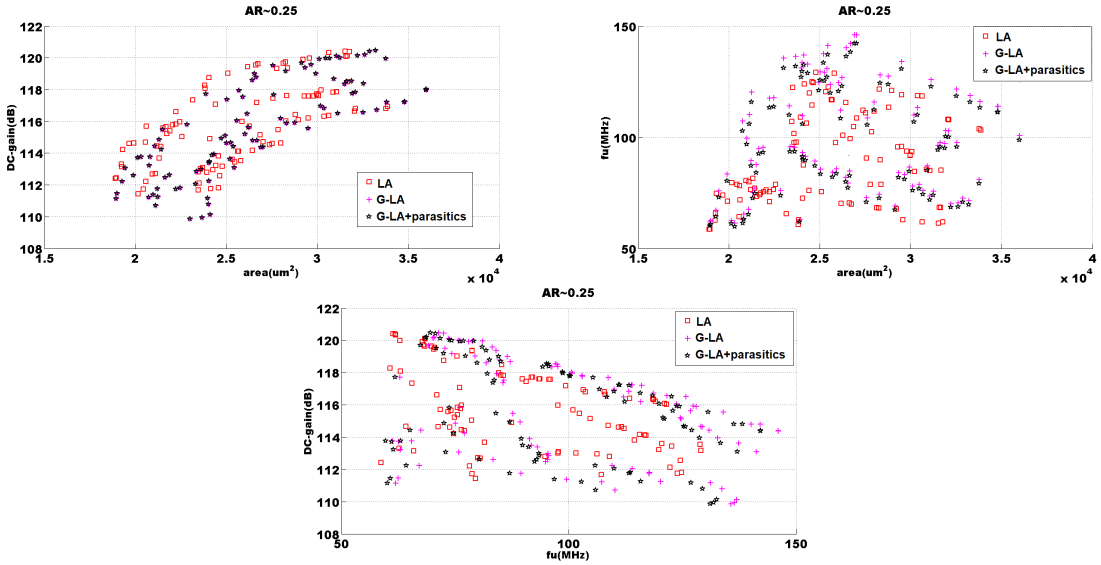

Fig. 2.29 *Comparison G-LA vs LA. AR=0.25*

diffusion capacitors; these parasitics (already included in population G-LA) have a more noticeable impact than routing parasitics in this case. Again, all individuals obtained in G-LA POF do not attain the design constraint when adding routing parasitics.

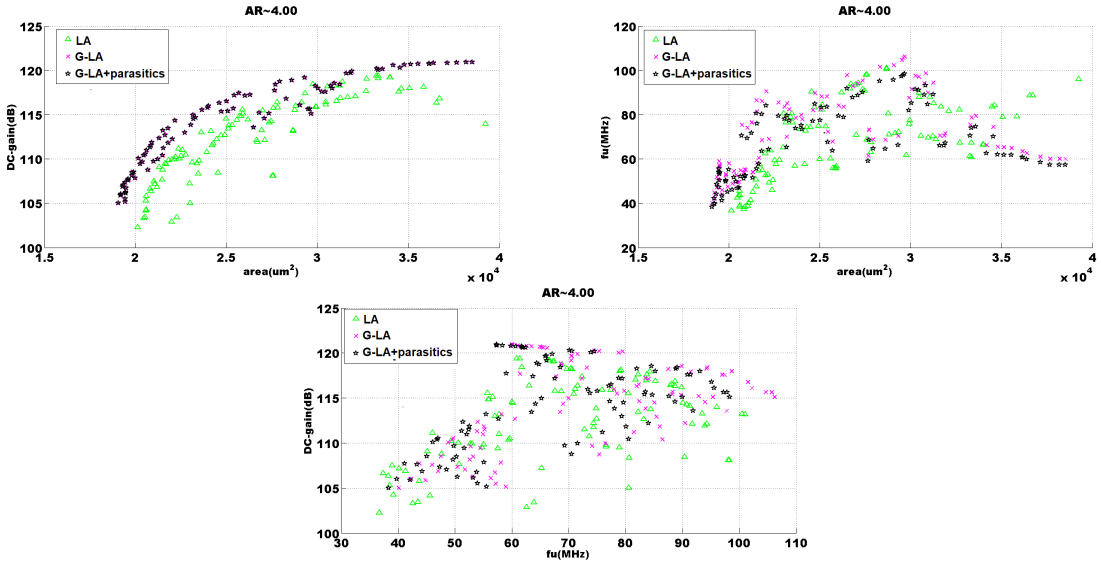

Fig. 2.30 *Comparison G-LA vs LA. AR=4.0*

In view of these results, the obtained performances and the impact of parasitics are very different depending of the geometry of the layout for the template used. In addition, the inclusion

of all parasitics becomes key to avoid iterations between sizing process and layout. Individuals with no parasitic-aware sizing needs a redesign process because the design constraints are not fulfilled when the impact of all parasitics is included.

2.4.4 Case study 3: LA with constraints on the total occupied area

In the previous section, different values of the aspect ratio geometric constraint have been used. In the case study in this section, a maximum available area together with maximum height or width of the layout are used. That is a common situation to complex analog layout industry requirements.

Two different POFs have been generated using the parasitic-aware optimization, both with a constraint on the maximum available area. In the first case, a constraint on the width of the instance ($<= 100\mu m$) has been imposed; in the second POF, the same constraint value has been imposed on the layout height. The maximum area available is $20000\mu m^2$, which becomes an additional constraint for the optimizer; additional constraints are the output swing ($os > 4.0V$) and slew rate ($sr > 55V/\mu s$). The design objectives are the maximization of the DC-gain and the unity-gain frequency (fu), to be maximized and the minimization of the power consumption. The two different POFs obtained are shown in figure 2.31.

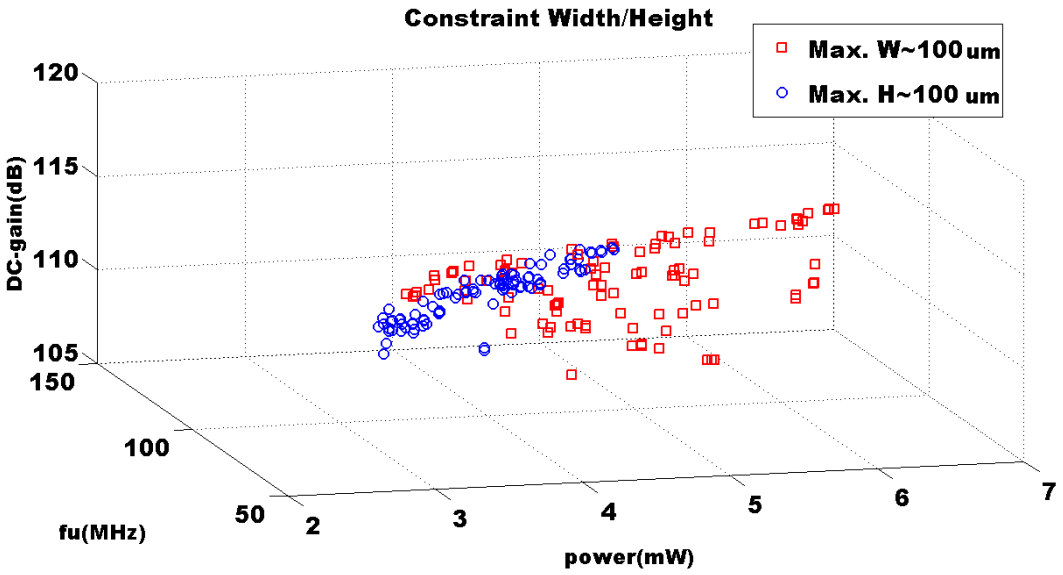

Fig. 2.31 *Layout-Aware POFs with a maximum available area.*

As before, all possible projections of the front onto 2D objectives planes are shown in Figure 2.32. In this figure, it can be noted that using a predetermined height, there are only

individuals in the low range of unity gain frequency. This happens here because transistors tend to have a higher number of fingers (as the width of the layout has the blocks stacked vertically and, thus, each device can have more fingers since the transistors are oriented so that the gates are stacked horizontally), increasing considerable the length of the horizontal routing lines (remember that the area is limited but not minimized). In addition, the optimizer tries that all circuit solutions use almost all available area to improve the design objectives, as shown in Figure **??**.

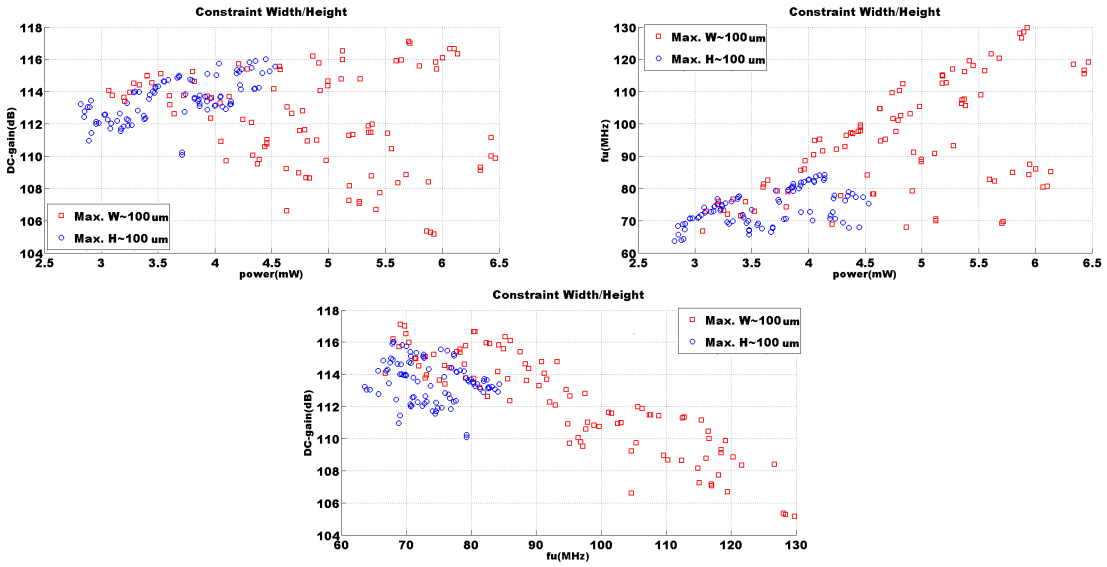

Fig. 2.32 *Layout-Aware POFs with a maximum available area. Projections of the objectives on 2D planes.*

If we check the dominance between the individuals which form the two POFs, 42 individuals in the population with maximum height are non-dominated and almost all individuals (99) in population with maximum width are non-dominated. In view of these results, it is quite clear that the obtained performances are very dependent not only on the area available but also on the shape of such area and how it fits the particular layout template that has been used. This aspect will be improved in the following chapters.

CPU time of the Case Studies

The CPU time (on a 2.2-GHz dual core) of the different actions per individual is shown in Table 2.5 for the POF generated with G-LA sizing and parasitic-aware sizing. The difference comes from the time required for parasitic extraction. To evaluate each generation, due to the

parallel execution of the *GCModule* and the internal Cadence processes, the total CPU time is:

$$Time_{cpu} = time_{scripts+EQ} + N_{ind}(time_{generation+extraction} + time_{simulation} + time_{NSGA}) \quad (2.11)$$

In this case, the number of individuals in each generation is 100, and the number of generations is 220, hence the total CPU time for the three types of optimizations are 6.35 hours for non-LA POF, 20.77 hours for G-LA POF and 22.30 hours for LA POF.

Action	**Opt. non-LA**	**Opt. G-LA**	**Opt. LA**
Scripts + EQ	-	0.75s	0.75s
Layout generation and parasitic extraction	-	1.0s	**1.25s**
Simulation	1.00s	1.25s	1.25s
NSGA	0.4s	0.4s	0.4s

Table 2.5 *CPU time per individual.*

Using the implemented methodology, the CPU time in optimizations with information of all parasitics barely exceeds 7.5% of the time in the G-LA optimizations. Although the CPU time is high, about three times that needed in non-LA optimizations, this is compensated by: (1) the robustness of the solution against parasitics and (2) the reuse possibility of the fronts obtained.

2.5 Summary

In this chapter, a new methodology called POF-based Layout-Aware design flow has been proposed and a set of case studies has been presented. The results show the importance of parasitics during the optimization process. Only including all parasitic information in the simulation, the iterations between electrical and physical synthesis can be avoided. Otherwise, the circuit performance is degraded, and, more important, the designs obtained become inadequate when all parasitics are included because they do not attain the design constraints.

Besides, the solution implemented for layout generation (template-based) allows to optimize the area used in the physical implementation and to restrict the geometry using the geometric constraints. This provides better flexibility to the layout template. This is demonstrated in the cases studies with different geometric constraints, in which using the same template, the geometry of the layout is very different due to the different values used for aspect ratio or for maximum height or width. But two important aspects deserve attention: that different

geometries have an impact on electrical performance and in the quality of the solutions. The first occurs because the number of fingers of the transistors and the length of the routing lines is very different, and hence, the parasitics are very different. The second is due to the use of only one template, which means that the relative placement and routing are fixed, and maybe the same template is not the most adequate for any geometry.

In the next Chapter, a methodology to improve this problem is proposed, in order to achieve a POF-based Layout-Aware design flow where the geometry does not limit the performance obtained in the POF generation. With the method proposed in Chapter 3, a new POF based Layout-Aware design flow is proposed in Chapter 4, where the quality of the instance of the layout templates is improved.

Chapter 5

A stochastic reliability simulator

5.1 Motivation

In addition to the impact of the parasitic devices, as mentioned in Chapter 1, the variability due to the fabrication process and aging has an important impact on the circuit reliability. The impact on the circuit performance caused by process variability has been well studied [56] [7] [5]. Nowadays, commercial and electrical simulators allow evaluating the impact of process variability on the circuit performance during the design process. The mechanisms and effects of transistor aging have been reported in the last decades. To evaluate the circuit reliability in the presence of these effects, different simulators have been proposed in the literature. In this chapter, a new reliability simulator is proposed, which can be used as an evaluator to include accurate reliability information during the generation of POFs.

The first reliability simulators were developed during the late 1980s and early 1990s. The most complete of them is the Berkeley Reliability Tools (BERT) [57] developed at the University of California Berkeley. These tools are a set of methods to simulate the impact of HCI degradation (it should be noted that HCI was the major effect that caused transistor aging in 80's technological nodes) and to predict the circuit failure due to oxide breakdown and/or electromigration.

Later, BTI effects became more important among the aging phenomena and commercial simulation tools were been developed to assess the impact on the circuit performance. In this sense, RelXpert [**?**], a reliability simulator developed by Celestry (acquired later by Cadence), allows the inclusion of BTI and HCI effects in the circuit performance simulation after a period of full-time operation (called T_{final} or T_{age}). Eldo [**?**], from Mentor Graphics, and MOSRA

[**?**], from Synopsys, have similar capabilities. All commercial reliability simulators use deterministic models to include the aging phenomena on the circuit reliability simulation.

It is very important to note at this point that all these commercial reliability simulators use deterministic models to include the aging phenomena on the circuit reliability simulation. However, as discussed in Chapter 1, the continuous scaling of CMOS technologies requires traditional deterministic models to be replaced by new ones that account for stochastically distributed failure mechanisms. In this sense, the first stochastic reliability simulator was presented in [58], in which spatial stochastic reliability effects (i.e., process variability), temporal stochastic reliability effects (Soft Break-down) and temporal deterministic reliability effects (in this case, HCI and BTI) are included. Its simulation flow is based on an iterative application of a deterministic reliability simulator, that is, the stochastically distributed failure mechanisms of aging are not accurately included because a deterministic model for BTI is used.

The new tool proposed in this chapter is the first in using a true stochastic model to calculate the transistor degradation together with another stochastic model to take into account the process variability. The stochastic models used to include both sources of variability are presented in Section 5.2. The use of a stochastic model presents, however, fundamental new problems for circuit reliability simulation, for example, how to include two sources of variability, spatial and temporal, or how to implement an efficient simulation flow that spends an acceptable CPU time for reliability simulation of analog circuits. Each of these problems is discussed in Section 5.3, and finally, the simulator is presented in Section 5.4.

5.2 SV and TDV models

In this section, the model used to include the spatial variability (SV) and time-dependent variability (TDV) in a reliability simulator are presented. These models will be illustrated with values for its parameter corresponding to a UMC 65-*nm* CMOS process. To evaluate the impact on the circuit performances of both stochastic sources of perturbations, a number of samples of the statistical distribution of the variability will be used in the new stochastic reliability simulator.

5.2.1 Spatial variability

Process variability and mismatch cannot be removed from the circuit manufacturing process as discussed in Section 1.2.2. For this reason, semiconductor foundries analyze and characterize

the device variability for each new technology node. This information is typically provided to designers as *model files* that can be used by EDA tools for simulation of their circuit designs. These *model files* allow to carry out a worst-case, corner, or Monte-Carlo analysis, which is the most accepted and well-known technique to accurately know the impact of variability on the circuit performance.

In this Thesis, a stochastic SV model is used by the reliability simulator to generate a number of samples of SV to evaluate the impact of SV on the circuit performances. For this purpose, a set of equations has been extracted from the Monte-Carlo UMC 65-*nm* file. Using these equations, a number of samples can be generated for the most important transistor parameters threshold voltage (V_{th}) and mobility (μ) in order to include this information in a Monte-Carlo analysis using the new stochastic reliability simulator. The remaining transistor parameters that are affected by SV (e.g., tox) can be included in the simulation quickly and easily.

5.2.2 Time-dependent variability

As mentioned above, deterministic models to calculate transistor degradation have to be replaced by stochastic models due to the stochastic behavior of the aging phenomena in nanometer technologies. In this context, the Probabilistic Defect Occupancy (PDO) model [17] has been developed in recent years. This approach allows including the stochastic BTI and HCI effects at the same time. The PDO model estimates the transistor degradation as the sum of a permanent component and a recovery component. The recovery component is attributed to the BTI effects, whereas both BTI and HCI effects also cause a permanent degradation.

In nanoelectronic circuits, one of the most important degradation mechanisms is BTI, which appears when a gate voltage is applied. BTI produces discrete threshold voltage shifts (ΔV_{th}). This degradation starts relaxing very quickly after the removal of the gate voltage. In addition to this, when a drain voltage is applied together with a gate voltage, the transistor degradation is caused by the combined effect of BTI and HCI. Most reported approaches to evaluate the degradation on the device characteristics assume that the degradation mainly manifests on a threshold voltage variation. In this work, this assumption is kept.

Recovery component of the transistor degradation

To estimate the recovery component of the transistor degradation, the PDO model emulates the charging/discharging of the different defects that coexist in each transistor channel. Typically, various defects coexist at the same time in the same device and the transistor degradation (ΔV_{th})

caused in one device can be calculated as the additive contribution of all the charged defects. Each defect is defined by τ_c, the time required to capture the charge, τ_e, the time required to emit the charge and η, the shift produced in the threshold voltage ΔV_{th} when the defect is occupied.

The capture time τ_c and the emission time τ_e of each defect are technology-dependent and also depend of the so-called stress conditions, which are given by the voltages applied at the transistor terminals and the temperature. Figure 5.1 illustrated the distribution of defects in the τ_c-τ_e space ($Ddefect(\tau_c, \tau_e)$) for a pMOS transistor, which has been characterized by using the technique presented in [59]. For a defect, this figure represents the probability of a defect is defined for each pair τ_c-τ_e at $V_{gb} = 1.2V$, $V_{db} = 0V$ and T=25°C. $Ddefect$ is a bivariate log-normal distribution defined by the mean values of capture and emission times $<\tau_c>$ and $<\tau_e>$, their standard deviations σ_{τ_c} and σ_{τ_e} and a correlation coefficient ρ. Table 5.1 shows the parameter used in this work to include the distribution of defects for pMOS and nMOS transistors in UMC 65-*nm* technology.

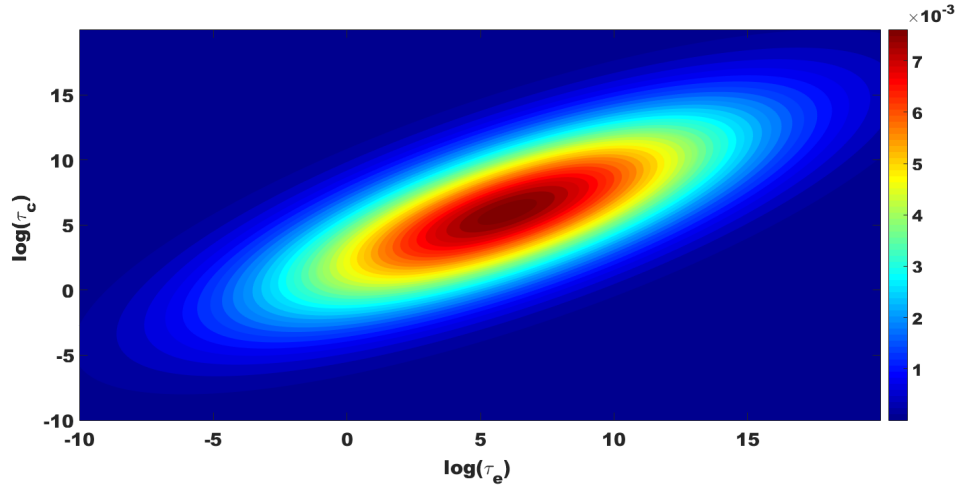

Fig. 5.1 *Distribution of defects (Ddefect) at* $V_{gb} = 1.2V$, $V_{db} = 0V$ *and temperature* T=25°C.

The process of trapping/de-trapping these defects are energy-activated processes, so it is expected that device degradation gets worse at higher temperatures. The relation between τ_e and τ_c, and the temperature follows the equation:

$$\begin{aligned} \tau'_e &= \tau_{e0} \cdot e^{Ea_{\tau_e}/(K_B \cdot T)} \\ \tau'_c &= \tau_{c0} \cdot e^{Ea_{\tau_c}/(K_B \cdot T)} \end{aligned} \tag{5.1}$$

	Log_{10}				
	$<\tau_e>$	$<\tau_c>$	σ_{τ_e}	σ_{τ_c}	ρ
pMOS	6.03562	6.56476	5.76383	6.00236	0.8923
nMOS	4.65463	6.00120	6.10030	5.99200	0.8954

Table 5.1 *Parameters of the defect distribution ($Ddefect$) at $V_{gb} = 1.2V$, $V_{db} = 0V$ and temperature T=25°C.*

where τ_{e0} and τ_{c0} are the capture and emission times at T=25°C, Ea_{τ_e} and Ea_{τ_e} are parameters related to the energy defect level and the energy band diagram [60] and K_B is the Boltzmann constant.

Figure 5.2 shows the dependence of both times with the temperature used in this Thesis. As can be seen in this figure, high temperatures cause the emission and capture times to decrease; this means the trapping/de-trapping processes are much faster and, hence, the transistor degradation is accelerated.

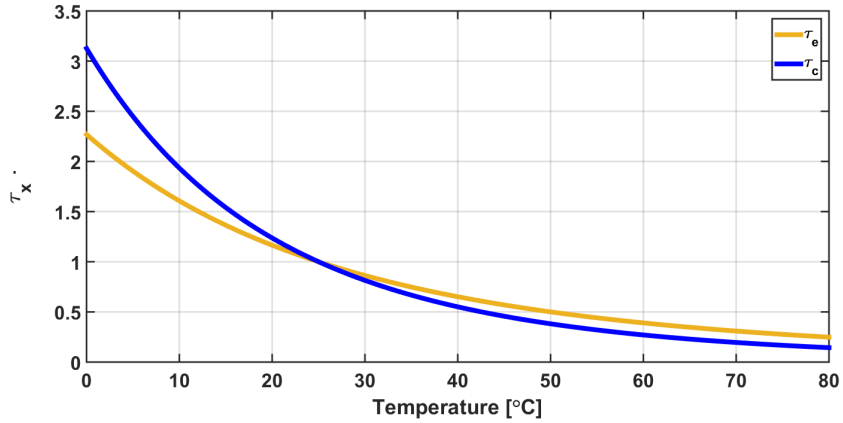

Fig. 5.2 *Temperature dependence of the time constants for a pMOS*

In addition to the temperature, the stress conditions are defined by the voltages applied at the transistor terminal. The relationships between the emission and capture times and the gate and drain voltages applied to the transistor are shown in Figure 5.3. When the gate voltage is high, the vertical electric field in the transistor is intense. This causes an increase in transistor degradation because the time needed to capture a charge decreases at the same time that the emission times increase. When a drain voltage is applied, the vertical electric field near the drain is smaller than when $V_{db} = 0$ [a], which leads to an increase of the capture times. In this

[a]PDO model assumes that $V_s = V_b$.

situation, the HCI effects appear, increasing at the same time the permanent degradation and the emission times.

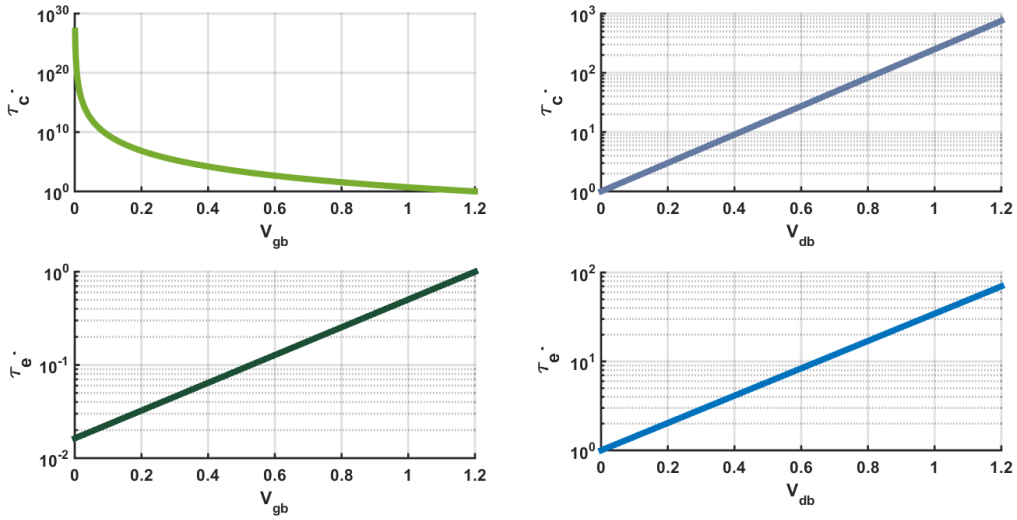

Fig. 5.3 *Dependences of time constants with the gate and drain voltage for a pMOS transistor*

To know the emission and capture times when different gate and drain voltages are applied to the transistor terminals, the following equations are used:

$$\begin{aligned} \tau'_e &= \tau_{e0} \cdot e^{\beta_e \cdot V_{gb}} \cdot e^{\gamma_e \cdot V_{db}} \\ \tau'_c &= \tau_{c0} \cdot \left(\frac{V_{gb}}{V_{ref}} \right)^{\beta_c} \cdot e^{\gamma_c \cdot V_{db}} \end{aligned} \tag{5.2}$$

where V_{ref} is the gate-bulk voltage for which the distribution of defects ($Ddefect$) has been characterized experimentally and, in this case $V_{ref} = 1.2V$, and τ_{e0} and τ_{c0} are the capture and emission times at this voltage. The dependence of the emission time with drain voltages and the dependence of the emission time with gate voltage are exponential, however the capture times follows a potential law with gate voltage. This is because when no voltage is applied to the transistor gate, no defects can be charged.

Table 5.2 shows the parameters used in this work for the dependencies with stress conditions. Using this information, in the τ_c-τ_e space where $Ddefect$ is characterized, the PDO model calculates the probability for each defect to be occupied, i.e., the probability of occupancy $pocc(\tau_c, \tau_e)$.

	Ea_{τ_e}	Ea_{τ_c}	β_e	β_c	γ_e	γ_c
pMOS	0.23	0.32	3.43	-8.84	3.54	5.52

Table 5.2 *Parameters for the dependences with stress conditions for pMOS transistor.*

The probability of occupancy, *pocc*, is calculated with the PDO model using the stress conditions in the each transistor. Using equations 5.1 and 5.2, parameters τ'_e and τ'_c are calculated in the space τ_c-τ_e where $Ddefect$ is defined. When these stress conditions are applied during a time interval Δt, the probability that a defect defined by τ'_c and τ'_e is emitted (p_e) or captured (p_c) is calculated as:

$$\begin{aligned} p_e &= \frac{\Delta t}{\tau'_e} \\ p_c &= \frac{\Delta t}{\tau'_c} \end{aligned} \tag{5.3}$$

Algorithm 1 is used to calculate the probability of occupancy of each defect defined by τ_c-τ_e. If p_e and p_c are greater than 1, during Δt the defect could be emitted and captured more than once ($\tau_c, \tau_e < \Delta t$). In this case, the probability of occupancy for this defect is calculated as the ratio between the number of times the defect could be captured and the sum of the number of times that could be emitted and captured. This equation is valid if only p_c and/or p_e is greater than 1. Nevertheless, if p_c and p_e are between 0 and 1, the probability of occupancy for this defect is the sum of: (1) the probability that this defect is already charged ($pocc_n$) and not being emitted during Δt, and (2) the probability that this defect is not charged and is captured during Δt,

Algorithm 1 Calculate *pocc* for a defect defined by (τ_c, τ_e) in the time $n+1$ after Δt

Require: $pocc_n(\tau_c, \tau_e)$, p_e, p_c
1: **if** ($p_e > 1$ or $p_c > 1$) **then**
2: $\quad pocc_{n+1} = p_c/(p_e + p_c)$
3: **else**
4: $\quad pocc_{n+1} = pocc_n \cdot (1 - p_e) + (1 - pocc_n) \cdot p_c$
5: **end if**
6: **return** $pocc_{n+1}(\tau_c, \tau_e)$

Using Algorithm 1, an example of *pocc* has been calculated when a transistor is stressed during $10^5 s$. with $V_{gb} = V_{db} = 0.7V$ and temperature $T = 25°C$. The probability of occupancy

(*pocc*) obtained is shown in Figure 5.4. This figure represents the probability that a defect characterized by the pair (τ_c, τ_e) is charged when these stress conditions are applied to a transistor. The defects with small capture times and high emission times are charged, shown in red in this figure. On the other hand, the defects with high capture times are discharged, shown in blue.

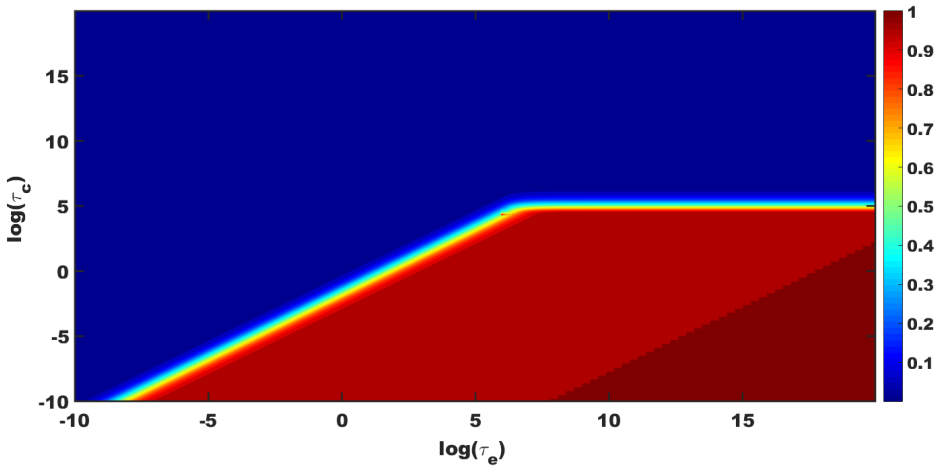

Fig. 5.4 *Probabilistic defect occupancy (pocc) with Vgb=0.7V during 1000s.*

Using the distribution of defects ($Ddefect$) and the probability of occupancy, *pocc*, shown in Figures 5.1 and 5.4, and the specific transistor size, a large number of samples of the transistor degradation ($\Delta\vec{V}_{th}^{\,rec}$) for each device can be obtained. Algorithm 2 shows how the information generated (*pocc* and $Ddefect$) is used to generate a number of samples ($N_{samples}$) of the variation of the threshold voltage ($\Delta\vec{V}_{th}^{\,rec}$) induced by the recovery component of the transistor degradation.

For each sample, the number of defects D that coexist in the transistor is calculated, which strongly depends on the area of the device. First, using the area of the device and the defects density for the technology D_{dens}, the mean number of defects for this device area is calculated. If the number of defects is very high (larger than D_{max}), that is, if the channel area of the transistor is relatively large, the standard deviation of the recovery component is very small. This is the reason why in older technologies, where the area of the devices is larger than in sub 90*nm* technologies, the transistor degradation could be modeled deterministically.

If the mean value of the number of defects is less than D_{max}, a Poisson-distributed function is used to generate the number of defects for each sample (line 6 in Algorithm 2). This means that two devices with the same area may not have the same number of defects. Then, the capture

Algorithm 2 Calculation of the stochastic recovery component

Require: $pocc$, $Ddefect$, transistor sizes (W,L), $N_{samples}$, D_{dens}

1: i=0
2: $D_{mean} = W \cdot L \cdot D_{dens}$
3: **while** i<$N_{samples}$ **do**
4: $\Delta V_{th}(i) = 0$
5: **if** $D_{mean} < D_{max}$ **then**
6: $D = genNumberDefect(D_{mean})$
7: **while** j<D **do**
8: j++
9: $[\tau_c, \tau_e] = genDdefect(Ddefect)$
10: $P = random$
11: **if** $P < pocc(\tau_c, \tau_e)$ **then**
12: $\Delta V_{th}(i)^{rec} += genDV(W \cdot L)$
13: **end if**
14: **end while**
15: **else**
16: $\Delta V_{th}(i)^{rec} = detV_{th}$
17: **end if**
18: i++
19: **end while**
20: **return** $\vec{\Delta V_{th}}^{rec}$

and emission times are generated for each defect using the information of the distribution of defects ($Ddefect$) (line 9 in Algorithm 2). With the defect defined (knowing its τ_e and τ_c) the probability of occupancy ($pocc$) is used to know if this defect is charged or discharged. If the defect is charged, its contribution η to ΔV_{th}^{rec} of this sample is generated and added to ΔV_{th}^{rec} of this sample. This contribution η is exponentially distributed in a set of defects and it is inversely proportional to the area of the device [61]. Finally, Algorithm 2 generates a vector ΔV_{th}^{rec} for each transistor where $N_{samples}$ of threshold voltage degradation have been calculated. If the transistor degradation can be approximated by a deterministic value, its value $detV_{th}$ is calculated from:

$$detV_{th} = D_{dens} \cdot < \eta > \cdot \iint Ddefect(\tau_c, \tau_e) \cdot pocc(\tau_c, \tau_e)\, d\tau_e\, d\tau_c \tag{5.4}$$

Permanent component of the transistor degradation

In addition to the recovery component, the transistor has a permanent degradation due to BTI and HCI. To include this damage, a semi-empirical model based on a predictive technology model (PTM[b] [62]) is used. In this case, the permanent component degradation due to BTI effects is calculated:

$$\Delta V_{th}^{P_{BTI}} = (A \cdot e^{(aV_{gs}+bV_{ds})})t^n \tag{5.5}$$

Like the recovery component, the permanent degradation is higher when V_{gs} is increased, and is lower when V_{ds} is high.

The permanent component of HCI is calculated with:

$$\Delta V_{th}^{P_{HCI}} = (C \cdot \sqrt{V_{i0} + V_{gs} - V_{th}} \cdot e^{d(V_{gs}-V_{th})} \cdot e^{\frac{-c}{V_{ds}-(V_{gs}-V_{th})+V_{m0}}} \cdot e^{\frac{-f}{L}})t^n \tag{5.6}$$

The permanent damage strongly depends on the drain-source voltage V_{ds} and the overdrive voltage ($V_{gs} - V_{th}$), and it is inversely proportional to the transistor length (L).

Finally, the total transistor degradation (in terms of ΔV_{th}) is calculated by adding the permanent component to the $N_{samples}$ samples of the recovery component generated.

$$\Delta\vec{V}_{th} = \Delta\vec{V}_{th}^{\,rec} + \Delta V_{th}^{P_{BTI}} + \Delta V_{th}^{P_{HCI}} \tag{5.7}$$

5.3 The reliability simulation flow

In this section, a new stochastic reliability simulator is presented, which embeds a commercial circuit simulator and can be easily extended to other circuit simulators. This commercial electrical simulator is needed to know the stress conditions in each transistor. With this information, the TDV model is used to calculate the transistor degradation. Finally, this degradation is added to the circuit to evaluate the circuit performances by using again a commercial simulator. The reliability simulator presented is based on the implementation of the spatial and time-dependent variability models presented in section 5.2 and an efficient simulation flow detailed in this section. The main goals of the implemented reliability simulator are:

- The automated control of a commercial electrical simulator.

[b] At the moment of the writing of this thesis, this permanent component is being characterized experimentally for the UMC 65*nm* technology. Unfortunately, results are not yet in so the semi-empirical model has been developed.

- The inclusion of the stochastic nature of aging.
- The inclusion of the bi-directional link between stress conditions and aging.
- The inclusion of the combined effect of spatial variability and time-dependent variability.
- Being efficient in terms of computational effort

First, a basic simulation flow is presented to include a stochastic aging model in a reliability simulation and to show the automated use of a commercial electrical simulator. The inclusion of both stochastic SV and TDV models creates a new problem when the simulator need to be very accurate, as shown and solved in Section 5.3.2. A method to improve the accuracy of the results is finally proposed.

5.3.1 Basic reliability simulation flow

In this section, a basic simulation flow to include and evaluate the impact of reliability (through the simultaneous evaluation of the process variability and aging-related effects) in the circuit performance evaluation is described. This flow is depicted in Figure 5.5. First, a transient analysis is launched using a commercial simulator (Hspice in this example) to know the voltages applied (i.e., stress conditions) at the transistor terminals and their temporal evolution. With this information, and using the aging model, a probability of occupancy (*pocc*) for each transistor is calculated.

Each calculated *pocc* contains information about the transistor degradation after the transient analysis, but the goal is to calculate the transistor degradation in time T=T_{final}. For this reason, each *pocc* needs to be extrapolated to this time. As this *pocc* depends on the voltages applied to the transistor, if these conditions do not change during the circuit operation, *pocc* at T_{final} can be easily calculated. Figure 5.6 shows three different *pocc* obtained when the same stress conditions are applied during $1s$ (green), $10^3 s$ (red) and $10^7 s$ (blue). As shown in this figure, when the stress conditions are constant, *pocc* at T_{final} can be calculated by "shifting" the probability of occupancy *pocc* obtained with the transient analysis because *pocc* grows according to:

$$\begin{aligned} \Delta_{\tau_c} &= \log_{10}\left(\frac{T_{final}}{T_{transient}}\right) \\ \Delta_{\tau_e} &= \Delta_{\tau_c} \end{aligned} \tag{5.8}$$

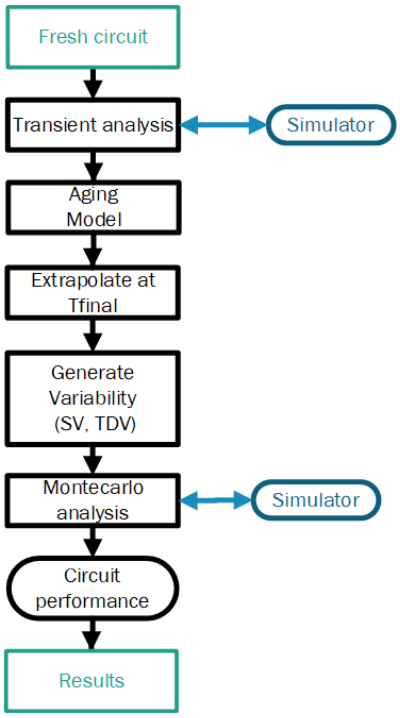

Fig. 5.5 *Block diagram of the basic reliability simulation flow.*

where $T_{transient}$ is the time used to calculate the stress conditions and T_{final} is the time where the transistor degradation is calculated to evaluate their impact on the circuit performances.

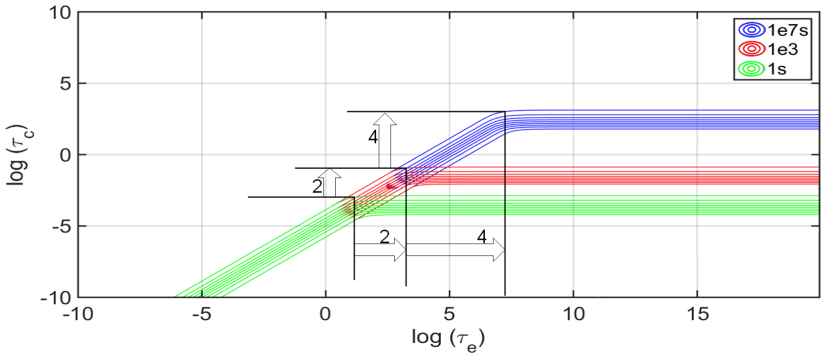

Fig. 5.6 *Example of pocc after* 1*s,* $10^3 s$ *and* $10^7 s$.

To extrapolate the transistor degradation at T=T_{final}, the growth of the probability of occupancy *pocc* is emulated by "shifting" the distribution of defects $Ddefect$ using equation 5.8 as shown in Figure 5.7. This is faster than calculating a new *pocc* for each transistor because $Ddefect$ is the same for all transistors. In addition, the results are similar to the growth of each

pocc as shown in Figure 5.6. The permanent component of the aging-induced degradation is calculated using the following equation

$$\Delta V_{th}^{PP} = \frac{\Delta V_{th}^{PP,T_i}}{(T_i)^n} \cdot (T_{final})^n \tag{5.9}$$

where T_i is the time after the transient analysis and $\Delta V_{th}^{PP,T_i}$ is the threshold voltage degradation after time T_i . At this point, the total variability information can be generated. On the other hand, the process variability is generated by using the foundry models of the technology, and the aging-time dependent variability is obtained from *pocc* and $Ddefect$.

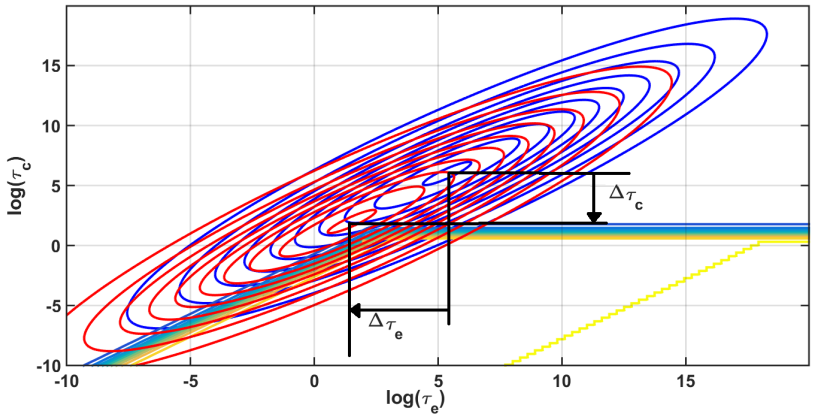

Fig. 5.7 *Shifting the distribution of defects to extrapolate the aging.*

Using a commercial simulator, a Monte-Carlo analysis is carried out with the variability information generated. In addition, the *time dependent yield* (TDY), that is, the percentage of samples that meet the design specifications or design constraints after the period of operation (T_{final} in this case) can be calculated.

5.3.2 Link between stress condition and aging

Updating the stress conditions

In the previous section, a basic reliability simulation flow has been presented where a stochastic model for aging has been used. An advantage of this simulation flow is that the stochastically distributed failure mechanisms of aging have been included in the evaluation of the circuit performances. On the other hand, the spatial (SV) and temporal (TDV) variabilities have

been included as non-correlated sources of variability. In this sense, [63] discusses different simulation flows to include these two sources of variability and their correlation. Nonetheless, a very important problem was not addressed in [63]: the stress conditions do not remain constant during the circuit operation time but, on the contrary, these conditions do change due to the device degradation over the operation time. In addition, this change does, in turn, alter the device degradation. This means that there exists a bi-directional link between stress conditions and transistor degradation.

For this reason, a number of M intermediate steps is needed to update the stress conditions to know the changes caused by the transistor degradation. This is an option available in commercial simulator like RelXpert or MOSRA. These tools allow to use linear or logarithmic scales to define the distribution of M steps. Using a deterministic model for transistor aging and M steps, a single value of transistor degradation is enough to calculate the new stress condition (using a new transient analysis).

It is important to note that, if stress conditions are not constant during the circuit operation time, the method presented in the previous section to extrapolate *pocc* is not valid any more. This is because the process presented in the previous section, where the solution is "to shift" the distribution of defect $Ddefect$ is valid only if the stress conditions are constant. If the stress conditions change, the new information of the stress conditions needs to be incorporated in the probability of occupancy (*pocc*). In this new process, shown in Figure 5.8, the stress waveforms obtained with a transient analysis (shown in red) are stretched out in time until the next step (shown in blue). In the example shown in the figure, $T_{transient}$ is 1 second and T_{final} is 10 seconds, that is, the waveform obtained with the transient analysis (from 0 to 1 second) is stretched out after this analysis until T_{final}, in this case from 1 to 10 seconds. This assumes that the transistor aging is approximately frequency independent, as shown in Figure 5.9, where different frequencies are used to generate different probabilities of occupancy (*pocc*) using Algorithm 1. As shown in this figure, the probabilities of occupancy obtained using different frequencies are very similar, therefore the differences are very small, less than 0.01% (in the mean values) when the transistor degradation is calculated. That is, the new stress conditions are stretched out in time to incorporate this information at *pocc* of each transistor.

The use of intermediate steps and inclusion of SV and TDV

A stochastic reliability simulator based on a time-consuming iterative use of a deterministic aging simulator is presented in [58]. This individually approach ages each SV sample, that is, for each SV sample, the stress conditions and its changes over time are calculated by using

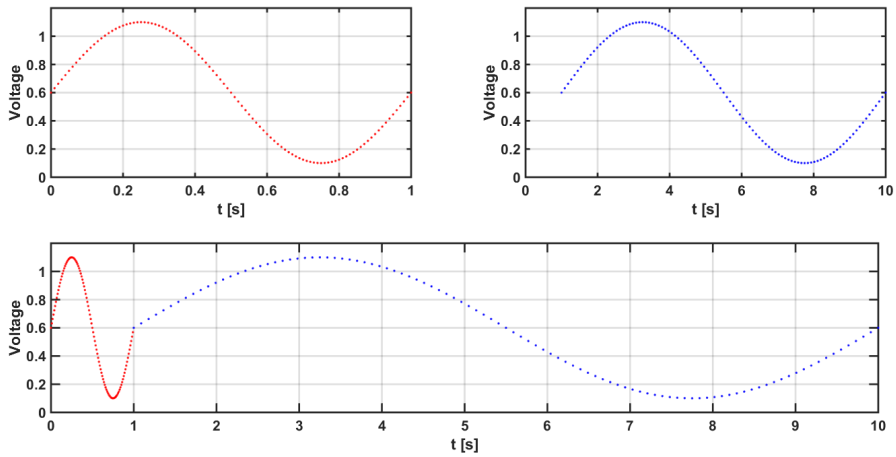

Fig. 5.8 *Stretching the stress waveform until the next intermediate step*

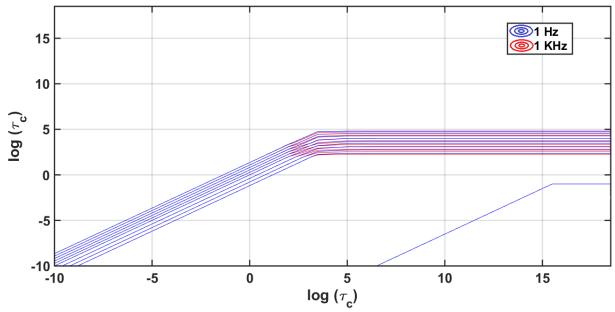

Fig. 5.9 *A set of pocc generated using different frequencies*

a deterministic TDV model, which does not accurately account for the underlying physics phenomena of aging in sub 90-*nm* scales.

The use of stochastic TDV models creates a new problem to include intermediate steps to update the stress conditions. At each step, instead of a single value, the model provides a statistical distribution characterizing the degradation. Therefore, computing the stress conditions eventually turns into a very time-consuming, even incomputable in practice, process.

The most accurate approach to simultaneously include both sources of unreliability, SV and TDV, and account for the bi-directional feedback between stress and degradation is depicted in Figure 5.10. In this simulation flow, labeled as *Flow A*, each SV sample is aged separately, and at each intermediate step, each sample of the statistical distribution of TDV is used to update the stress conditions. Taking into account that each stress update requires a transient analysis of the circuit, the total number of transient analyses with *Flow A* is:

$$N_{tran} = \sum_{i=1}^{M} N_{SV} \cdot (N_{TDV})^i \tag{5.10}$$

where N_{SV} is the number of samples for SV, N_{TDV} is the number of samples for TDV and M is the number of steps. Like in the previous section, a final Monte-Carlo simulation is carried out to obtain the degraded performances of the circuit. The number of samples to carry out the Monte-Carlo analysis is, in this case, $N_{SV} \cdot (N_{TDV})^M$. In summary, using *Flow A* each initial SV sample is considered in the final Monte-Carlo analysis with $(N_{TDV})^M$ samples accounting for TDV. Considering that the minimum value for statistical significance is 30 (both for SV and TDV), that the number of steps M can be between 30 and 100, and that the time for a single transient analysis and to calculate *pocc* can take at least one second, the minimum time to perform only the transient analyses rises up to $2 \cdot 10^{38}$ years, which means that *Flow A* , despite being the most accurate one, is incomputable in practice.

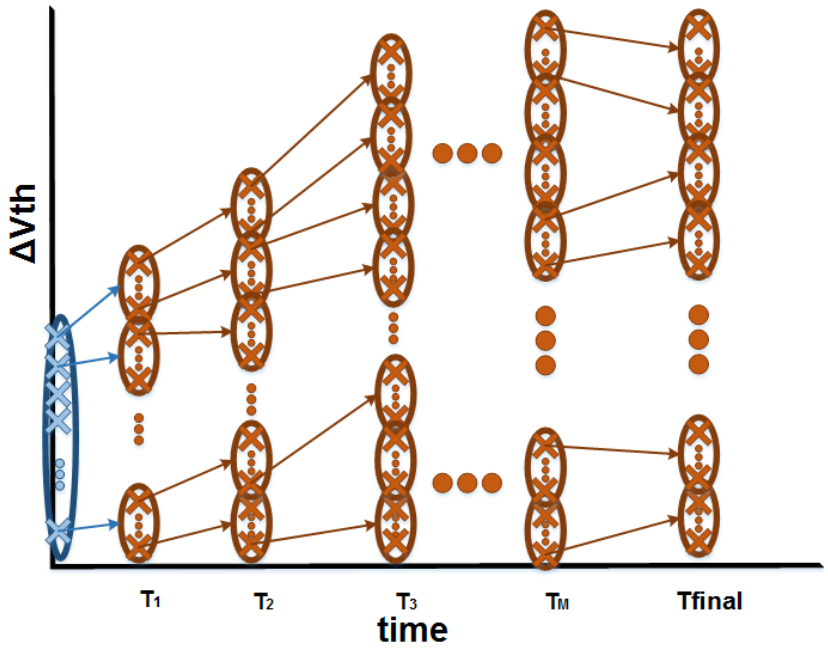

Fig. 5.10 *Flow A: the SV and TDV samples are aged separately.*

This means that the use of a stochastic model for aging and the intermediate steps required to take into account the bi-directional link between aging and stress conditions to the simulation flow proposed in Figure 5.10 makes this problem unaffordable. A possible solution to improve how the stochastic aging models are dealt with the simulation flow and, thus, make it affordable

is presented in Figure 5.11. This simulation flow reduces the number of transient analyses by generating just one TDV sample for each SV sample. This flow tries to imitate what would happen in reality, where each SV sample is aged separately; and at each intermediate step, a random sample of the statistical TDV distribution is used to update the stress conditions.

Although *Flow B* avoids the incomputability problem of *Flow A*, the use of only one sample of the TDV probability distribution at each intermediate step means that a lot of information about the stochastic nature of aging is lost. Using this simulation flow, the total number of transient analyses is $N_{SV} \cdot M$ and the number of samples generated for the Monte-Carlo analysis is just N_{SV}.

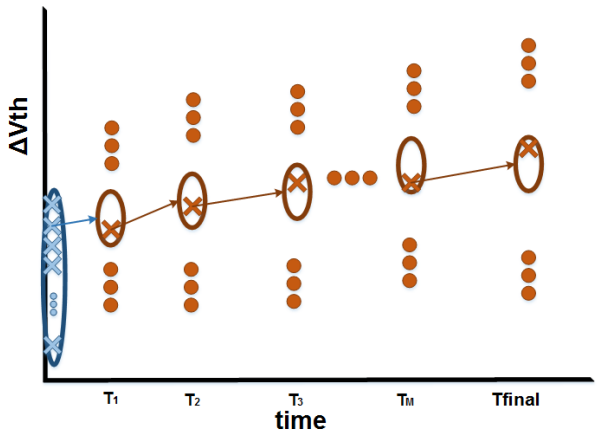

Fig. 5.11 *Flow B: each SV sample is aged separately using a random TDV sample in each intermediate step (only one SV sample is shown).*

A new simulation flow is proposed in Figure 5.12.a, which tries to palliate the loss of information caused by using *Flow B*. In *Flow C.1*, each SV sample is also aged separately. This means that after an initial transient analysis, one probability of occupancy (*pocc*) is generated for each SV sample. This process provides N_{TDV} samples for the degradation of each SV sample at time T_1. These samples are used to obtain the new stress conditions. This means that N_{TDV} probabilities of occupancy (*pocc*) are now calculated for each SV sample. In this point, unlike in *Flow A*, these N_{TDV} distributions are collapsed into a single *pocc* to generate N_{TDV} samples at time T_2 (using a single *pocc*). Therefore, only N_{TDV} transient analyses are carried out (for each SV sample) at T_i (whereas *Flow A* needs $(N_{TDV})^i$ at T_i). It is important to point out that the collapsing process can only be used for TDV but not for spatial variability because the correlation between TDV and SV would be lost.

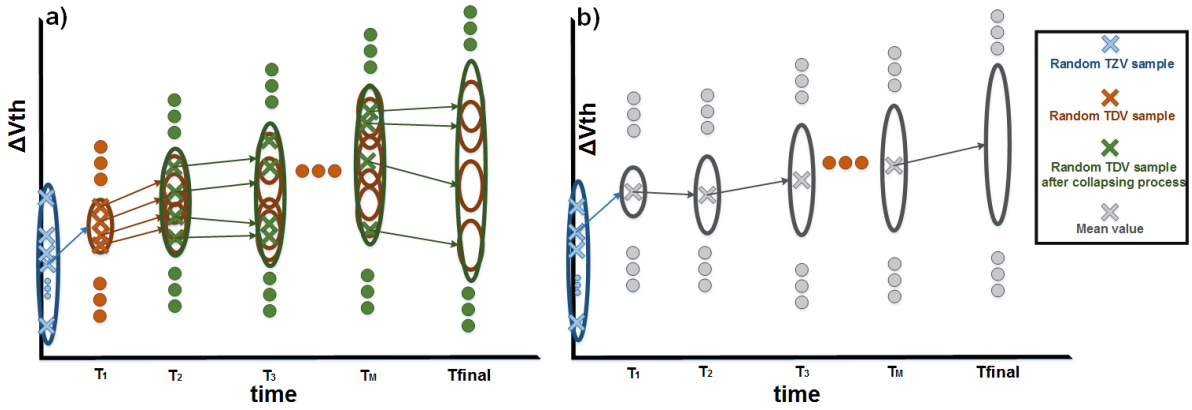

Fig. 5.12 *(a) Flow C.1 and (b) Flow C.2: each SV sample is aged separately trying to not lose information about TDV (only one SV sample is shown).*

First, to demonstrate that no information is lost when collapsing *pocc* between T_i and T_{i+1}, Figure 5.13 compares the transistor degradation histograms obtained by using 100 different transient analysis from 100 TDV samples at T_i and then:

- generating, at T_{i+1}, 100 TDV samples using each *pocc* obtained with each transient analysis (i.e., 10,000 total samples), or
- generating, at T_{i+1}, 10,000 TDV samples after collapsing the 100 *pocc*s obtained from each transient analysis.

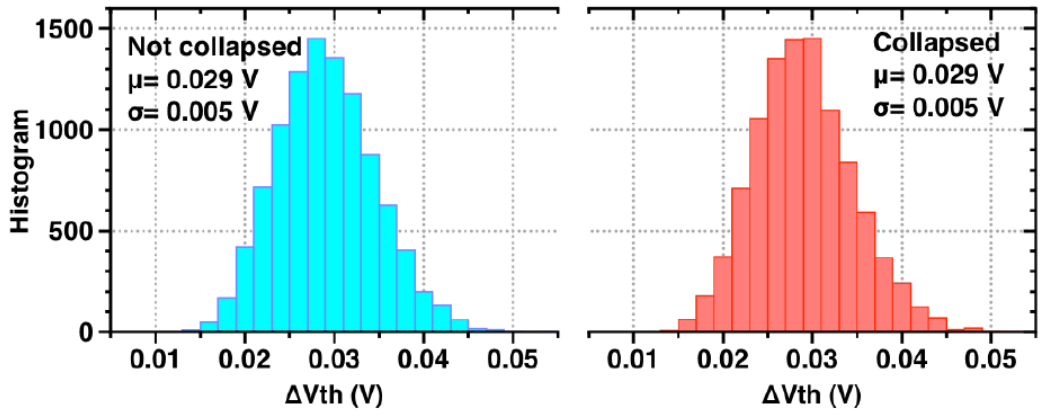

Fig. 5.13 ΔV_{th} *generated using 100 samples of TDV with and whitout collapsing pocc).*

The mean value and standard deviation differences between the two TDV distributions shown in Figure 5.13 are smaller than 0.1%. Therefore, *Flow C.1* does not lose any TDV information at each intermediate step when the obtained probabilities of occupancy (*pocc*) are collapsed into a single *pocc*. Thus, the number of transient analyses required to carry out this simulation flow is $N_{SV} \cdot N_{TDV} \cdot M$ which is quite a big reduction compared with the number needed using *Flow A* but it still is very high.

To further reduce this number while keeping the accuracy of the reliability simulation, a new simulation flow *Flow C.2* is presented in Figure 5.12.b. As in the previous simulation flows presented, each SV sample is aged separately, so one *pocc* is generated for each sample (through a transient analysis). But, unlike in *Flow C.1*, instead of generating N_{TDV} samples in each intermediate step, the mean value of these samples is calculated as explained in section 5.2.2 (equation 5.4). The mean value is then used as the characteristic that is representing the distribution to obtain the new stress conditions. In this way, only one *pocc* is generated and the resulting number of transient analysis is $N_{SV} \cdot M$. At time instant T_{final}, N_{TDV} samples can be generated using such *pocc*. This means that $N_{SV} \cdot N_{TDV}$ samples would be used in the Monte-Carlo analysis.

To study the impact of these three simulation flows, Table 5.3 shows the degradation of the transistor in Figure 5.14, obtained from 3 different samples of SV at $T_{final} = 1$ year. In this table, *Flow B* generates a single TDV sample and *Flow C.1* and *Flow C.2* generate two almost identical distributions of TDV samples of transistor degradation. The different values of mean and standard deviation reveal the existing correlation between SV and TDV (i.e., different samples of SV yield different shift in the threshold voltage) that comes from the bi-directional link between stress and aging.

Simulation flow	B		C.1		C.2		
	μ	σ	μ	σ	μ	σ	Units
Sample #1	18.1	-	21.20	4.82	21.20	4.83	*mV*
Sample #2	22.6	-	26.24	5.21	26.24	5.21	*mV*
Sample #3	21.0	-	19.86	5.12	19.87	5.12	*mV*

Table 5.3 *TDV characteristics values obtained from different SV samples.*

A reliability simulation of the single transistor configuration shown in Figure 5.14 has been carried out with all three simulation flows presented. The comparison between the different methods is made such that, at T_{final}, the number of samples in the final Monte-Carlo simulation is the same (10,000) and 50 intermediate steps are used. For this, *Flow B* uses 10,000 samples

of SV and generates 1 TDV for each, while *Flow C.1* and *Flow C.2* use 100 SV samples and generate 100 TDV samples for each.

The QQ-plots in Figure 5.14 show the ΔV_{th} degradation at $T_{final} = 1$ year. In this case, as shown in Figure 5.14(a), *Flows C.x* obtain similar statistical characteristics, but significantly differ from the results obtained with *Flow B*. This difference stems from the fact that *Flow B* uses many more samples of SV. Figure 5.14(b) shows the contribution of TDV to the total ΔV_{th} variability This QQ-plot shows that TDV distributions derived by the aging model are very similar because all three methods use a relatively high number of SV samples. This means that, for the same number of Monte-Carlo samples at the final circuit evaluation, *Flow B* would yield better accuracy. Nevertheless, *Flow B* and *Flow C.1* require a high number of transient analyses (in this case, $5 \cdot 10^5$), while *Flow C.2* can generate the same number of samples to include in a Monte-Carlo analysis taking much less time (in this case, $5 \cdot 10^3$).

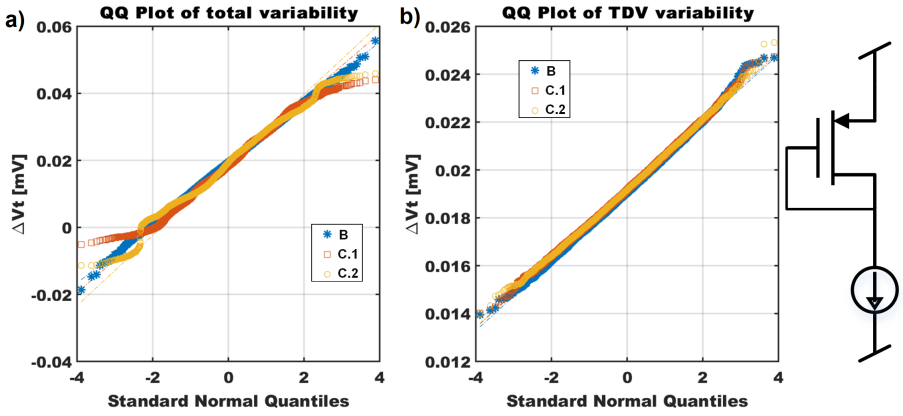

Fig. 5.14 ΔV_{th} *obtained using simulation flows B, C.*1 *and C.*2*.*

Although the three simulation flows presented allow addressing the problem of the inclusion of a stochastic aging model in a reliability simulator, the CPU time needed is still very high for a single reliability analysis. For this reason, a new simulation flow is proposed that tries to reach the same accuracy level of *Flow B* while drastically reducing the CPU time.

Similar to *Flow C.2*, where the mean value of the TDV-induced degradation is used, in this new flow, labeled *Flow D*, the mean value of the SV distribution is used as a characteristic value of the SV distribution. This simulation flow is illustrated in Figure 5.15. Using this approach, the final Monte-Carlo analysis at T=T_{final} is carried out by using different samples of TDV obtained with M transient analysis together with samples generated by SV. This means that, unlike the simulation flows previous presented, the number of transient analysis needed is independent of the number of samples used in Monte-Carlo analysis. Using this simulation

flow, a single probability of occupancy (*pocc*) is obtained at T_{final} and random TDV samples are generated and included in the Monte-Carlo analyses together with random SV samples.

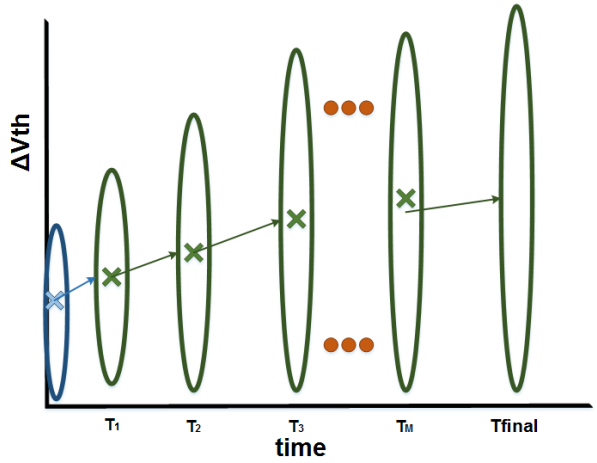

Fig. 5.15 *Flow D: the mean values of SV and TDV samples are used in each intermediate step.*

To compare all four simulation flows described above, a 3-stage current mirror, shown in Figure 5.16, has been used. This circuit is especially sensitive to transistor mismatch caused by both SV and TDV, and the stress conditions are very dependent of the degradation of transistors $M1$, $M3$ and $M5$.

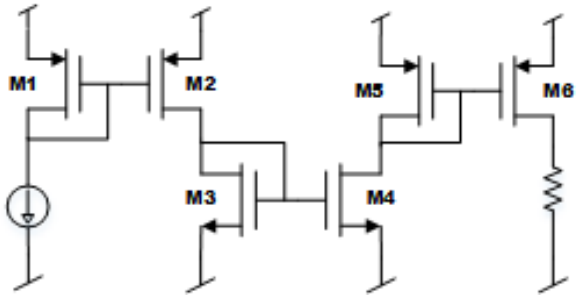

Fig. 5.16 *3-stage current mirror used to compare the proposed simulation flows.*

Table 5.4 shows a comparison of the number of transient analysis needed using each simulation flow when 50 intermediate steps are used and $T_{final} = 5$ years. The cumulative distribution function of the copy factor is shown in Figure 5.17. As it can be seen, the results are very similar between *Flow B* and *Flow D*. *Flow C.1* and *Flow C.2* obtain different performance because the impact of SV samples is quite important. Note that in these simulation flows, each SV sample is used 100 times in the Monte-Carlo analysis while in the other two, each SV

sample is used just once in the Monte-Carlo analysis (thus increasing the number of SV samples as shown in Table 5.4).

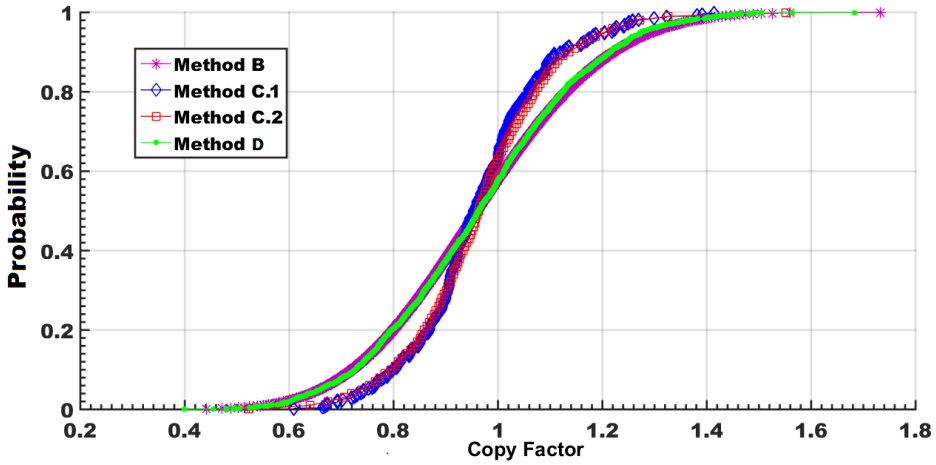

Fig. 5.17 *CDF of the copy factor of the 3-stage current mirror.*

As it can be seen in the CDFs, *Flow D* is quite similar to *Flow B*, but with far lower CPU time requirements (the number of transient analyses is 10,000 times smaller). Any difference between these two simulation flows is caused by the impact of SV in the bi-directional link between stress and aging, which is not considered in *Flow D*. Nevertheless, although the bi-directional link between stress conditions and aging is not fully considered in this simulation flow (as far as the influence of SV is concerned), the results can still be accurate simply because *Flow D* can account for many more SV samples with far less CPU time, and this is because the mean is a sufficiently representative of the SV distribution (to calculate the transistor degradation).

Flow	N_{SV}	N_{TDV} generated for each SV	M	of transient analysis	MC samples
B	10,000	1	50	500,000	10,000
C.1	100	100	50	500,000	10,000
C.2	100	100	50	5,000	10,000
D	10,000	1	50	50	10,000

Table 5.4 *Comparison of proposed simulation flows.*

A set of simulation flows has been proposed to deal with an efficient reliability simulator for nano-scale analog ICs, which involves using accurate SV and TDV models. The incomputability of this problem has been addressed by proposing a set of simulation flows. The proposed simulation flows provide a good trade-off between CPU time and accuracy where:

- *Flow B* can include the correlation between SV and TDV very accurately, but at the expense of larger computation (from days to years).
- *Flow C.1* can include the correlation between SV and TDV with a high computation time. In addition, each SV sample is used N_{TDV} times, which can be a problem.
- *Flow C.2* can include the correlation between SV and TDV reducing the computation time needed, but each SV sample is used N_{TDV} times, which can be a problem.
- *Flow D* can handle a very large number of samples with a reduced number of transient analyses as long as the correlation between SV and TDV is sufficiently weak.

For these reasons, even if the correlation between SV and TDV is quite important, the large number of samples involves in the Monte-Carlo analysis in *Flow D* makes this simulation flow the most convenient in a reliability simulator.

5.3.3 Time-step adaptive algorithm

In the previous section, a simulation flow to take into account stochastic models for SV and TDV together with the use of intermediate steps to update the stress conditions has been presented. These stress conditions do change during the period of full-time operation due to the device degradation and, therefore, a number of intermediate step is required. This option is already included in some commercial tools like RelXpert or MOSRA, where these steps may be uniformly distributed along a linear or logarithmic scale, as can be seen in Figure 5.18. Note that a higher number of steps implies a higher computational cost due to the need for new transient analyses and the re-calculation of transistor degradation at each step.

However, using intermediate steps by uniformly distributing a user-defined number of them along a linear or logarithmic scale presents some drawbacks. On the one hand, if the number of steps selected is too small, the stress conditions are not correctly updated and accuracy might be lost. On the other hand, if the number of intermediate steps is too high, no new information is provided for these intermediate steps and computation time is, therefore, wasted. Another important drawback of current commercial tools is that these simulators use deterministic, instead of stochastic, models for aging, which is another source for accuracy loss.

Another problem when intermediate steps are used is to guess the appropriate size of these steps, beyond linear or logarithmic scales, to improve the accuracy with less steps than using conventional approaches. To solve this problem, a method to maximize the step size while

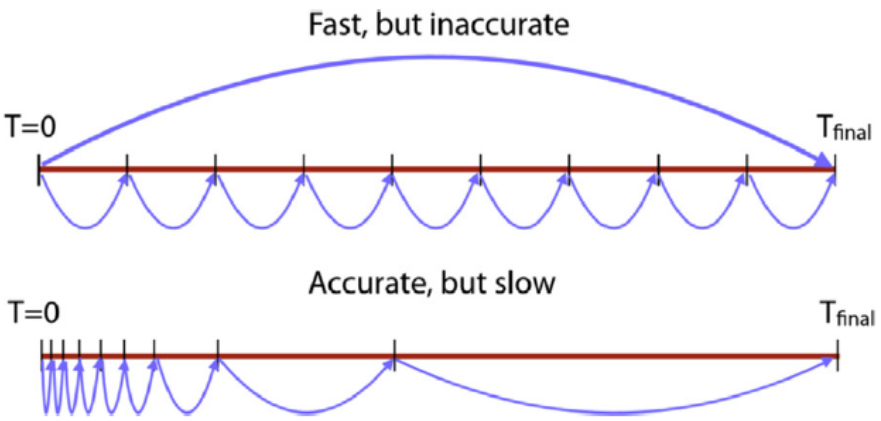

Fig. 5.18 *Different options to update the stress conditions through multiple steps: linear scale (above) or logarithmic scale (below).*

guaranteeing the simulation accuracy is proposed in [58]. For this, a maximum relative shift of each circuit performance is used that cannot be exceeded at each intermediate step. The next step size is calculated with an empirical formula if the limits defined by the designer are not exceeded. If that is not the case, it means that the step size was too large and the step needs to be repeated with a smaller size. This approach then adapts the step size using the circuit performance to maximize its size. However, this approach presents two main drawbacks:

- The required computational effort could be very high due to the required performance analysis. When a stochastic TDV is used, this implies an expensive Monte-Carlo analysis. In addition, some step could have to be repeated (which means that a performance analysis would have to be repeated as well).
- The decision on the step size is based on the relative shift of the circuit performances rather than on the stress conditions themselves. It is obvious that a change in the circuit performance always involve a change in the stress conditions (and, hence, a new transient analysis is needed), but a change in the stress conditions (only caused by a transistor wear-out) does not always involve a change in the circuit performances. Paying attention only to circuit performances could mean that changes in the stress conditions are not really and always detected and, therefore, accuracy losses could occur.

A different solution to this problem has been presented in [64], based on the aging-induced variation of the threshold voltage ΔV_{th}. In this solution, the designer defines an interval where

an accurate aging analysis with a very high number of steps is carried out, and an accurate value of ΔV_{th} is calculated in this interval. Then, the analysis is repeated but with a larger step size in this same interval. The maximum size is selected as the size where a maximum allowed error, defined by the designer, is still fulfilled. This all means that the adaptation of the step size is really only done at the beginning of the simulation (that is, the size of the step is calculated for every different simulation), but the step size is kept constant during the simulation. This is, in fact, a problem since the initially calculated step size could not be valid for the whole reliability simulation. Moreover, critical decisions are left to the designer, such as the size of the initial interval or how many steps are needed for the accurate reference value. For example, if the first interval is too small, the selected step size will be small and a lot of unnecessary steps could be carried out.

In this section, a new algorithm to calculate an adaptive step size is proposed. The target of this algorithm is that the stress conditions can be efficiently updated taking into account that:

- The distribution of the steps must be adaptive: the algorithm must self-adapt to the different evolution of aging and stress conditions (in contrast to the fixed scales used by the current commercial tools).
- The information required by the algorithm must not depend on user-defined parameters (such as an initial step size) that may drastically impact the efficiency of the reliability simulation. With this aim, the drawbacks of the approach in [64] are avoided.
- The algorithm must track changes in the stress conditions, which are always produced by changes in the wear-out of the transistors. However, tracking the stress conditions implies performing a transient analysis with the subsequent CPU load increase. Therefore, using an appropriate aging model, this algorithm tracks the transistor wear-out (i.e., shift in V_{th}) which does not require any additional analysis. With this aim, the drawback of the approach in [58] (which tracked circuit performance variations) is avoided.

To avoid the drawbacks in [58] and [64], the method presented in this Thesis adapts the step size based on the transistor wear-out. The cornerstone of this method is the maximum variation in the transistor degradation[c] (ΔV_{th}) before updating the stress conditions.

An example using a single transistor is shown in Figure 5.19. At each step, the algorithm defines, as an objective, a specific value for the transistor degradation (ΔV_{th}^{objM}). The step size

[c]Without loss of generality, other parameters like variation in the mobility of the transistor could be included in this algorithm.

is defined when the algorithm estimates that the transistor degradation is equal to this value (ΔV_{th}^{objM}). In this Thesis, where a stochastic aging model is used, the mean value of transistor degradation is used to define the time step size. Notice that this mean value is a representative value of the TDV distribution as explained in section 5.3.2. Therefore, the algorithm needs to address two problems: (1) how to estimate the transistor degradation quickly and (2) how to define the value for the transistor degradation (ΔV_{th}^{objM}) which decides the next step size.

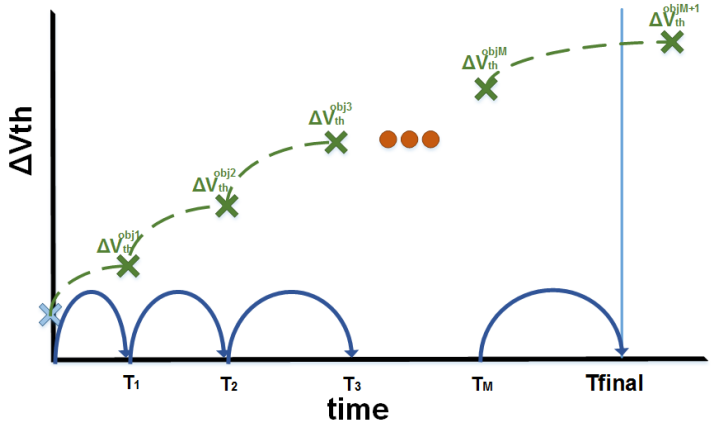

Fig. 5.19 *Illustration of intermediate steps for a single transistor.*

First, to quickly estimate the value of transistor degradation, the stress conditions are assumed constant. The method introduced in Section 5.3.1, which is based on "shifting" the distribution of defects ($Ddefect$) to emulate the change of the probability of occupancy ($pocc$), is used to quickly calculate the time T_i. The step size is calculated by using Algorithm 3. This algorithm requires the calculation of $pocc$ at time T_i (if it is the first transient, $T_i = T_{transient}$), the value of the transistor degradation desired (ΔV_{th}^{objM}, which is previously calculated as discussed later in this section) and the size of the previous step (Δ_τ). The algorithm calculates the next step (T_{i+1}) using a new Δ_τ (line 6 in Algorithm 3). This is an iterative process where at each iteration, the next step size is calculated by "shifting" the distribution of defects, $Ddefect$, defined by Δ_τ to estimate the transistor degradation. After this, two options are possible:

- If the size of the next step is increased and the variation of the threshold voltage ΔV_{th} is smaller than the objective, the size of the next step is increased "shifting" the distribution of defects defined by δ_τ. However, if ΔV_{th} is larger than the objective, the step size needs to be decreased (dir=backward, in line 10 in Algorithm 3) and a new value of the step is used (δ_τ/k).

- If the size decreases, and ΔV_{th} is larger than the objective, the size is decreased by "shifting" the distribution of defects by a value δ_τ. if ΔV_{th} is smaller than the objective, the step size needs to be increased (dir=forward, in line 15) and a new value of the step is used (δ_τ/k).

Algorithm 3 Algorithm to calculate T_{i+1}

Require: T_i, $pocc$, Δ_τ, ΔV_{th}^{objM}
1: $\delta_\tau = \Delta_\tau$
2: Δ_τ=0
3: dir = forward
4: **while** abs(δ_τ) > δ_τ^{min} **do**
5: $\Delta_\tau = \Delta_\tau + \delta_\tau$
6: T_{i+1}=$T_i \cdot 10^{\Delta_\tau}$
7: ΔV_{th}= calcVth($pocc$,Δ_τ)
8: **if** dir = forward **then**
9: **if** ΔV_{th}> ΔV_{th}^{objM} **then**
10: dir= backward
11: δ_τ=-δ_τ/k
12: **end if**
13: **else**
14: **if** ΔV_{th}< ΔV_{th}^{objM} **then**
15: dir= forward
16: δ_τ=-δ_τ/k
17: **end if**
18: **end if**
19: **end while**
20: **return** T_{i+1}

The algorithm stops this iterative process when δ_τ is lower than a defined δ_τ^{min} (typically 0.0001, that is, $Ddefect$ remains practically unaltered when this value is used and, therefore, no further changes in the degradations are detected). One main advantage of this process is that if there is more than one transistor in the circuit, the strategy is similar to the single transistor case, that is, the resolution in the step size is defined by δ_τ^{min}, independently of the number of transistors.

The second issue, how to define the value for the transistor degradation (ΔV_{th}^{objM}), is addressed with two approaches, which define two models of use of the adaptive step size algorithm presented in this Thesis: (1) the designer defines the accuracy or, (2) the designer defines a fixed number of steps.

The first option allows designers to specify a relative threshold voltage shift (*deltaV*) to calculate ΔV_{th}^{objM}. RelXpert allows a similar option to calculate the circuit lifetime. In RelXpert, when an arbitrary transistor surpasses the relative threshold voltage shift defined, the circuit is considered to have failed. This is, therefore, a very intuitive option for designers. In the case of this Thesis, when an arbitrary transistor surpasses the relative threshold voltage shift defined, the stress conditions are updated. The downside is that the number of steps are unknown and hence, the CPU time needed for the reliability analysis is not bounded.

The second option is defined using a fixed number of steps (M), which allows allocating an approximated CPU time. Using this option, the algorithm optimizes the distribution of the steps, trying to maximize the accuracy of the reliability analysis.

The approach used to fix the number of steps is illustrated in Figure 5.20 for a single transistor. At each intermediate step, first, the algorithm estimates the transistor degradation at T_{final} (assuming constant stress conditions) $\Delta V_{th}{}^{final,i}$ and calculates ΔV_{th}^{objM+1} as:

$$\Delta V_{th}^{objM+1} = \Delta V_{th}{}^{i} + \frac{\Delta V_{th}^{final,i} - \Delta V_{th}{}^{i}}{M-i} \tag{5.11}$$

where the difference between the actual transistor degradation and the transistor degradation at T_{final} is divided by the remaining steps.

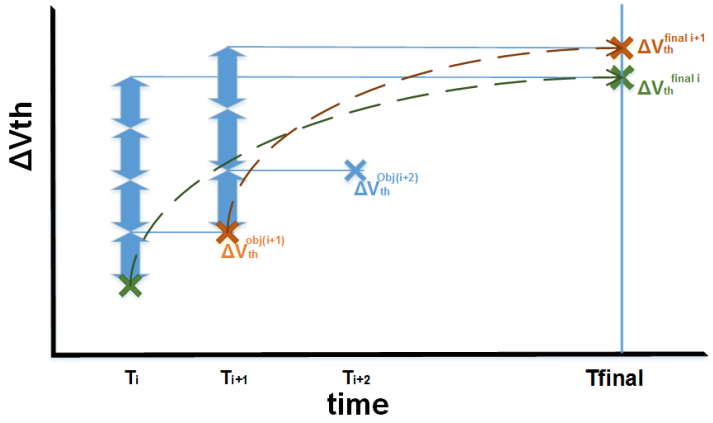

Fig. 5.20 *Selection of intermediate steps for a fixed number of steps.*

As discussed in the previous section, updating the stress conditions is a key problem to improve the accuracy of the reliability simulation and it can be a very time consuming process. The adaptive algorithm developed in this work allows fixing the number of steps and maximizes the accuracy with these steps or minimizes the number of steps to get a desired accuracy. It is

important to note that the CPU-time consumed by the algorithm is insignificant compared with the time required for updating the stress conditions (less than 1% of this time).

Figure 5.21 shows the evolution of the mean value of the threshold voltage shift and the stress conditions in the input transistor in a pMOS current mirror (like the one shown in Figure 5.14). This figure has been generated by using a reliability simulation with a very high number of steps ($M = 1{,}000$). The results using 1,000 intermediate steps obtained with the adaptive step size algorithm, linear scale and logarithmic scale are the same. Therefore, this result is used to set the reference in terms of accuracy.

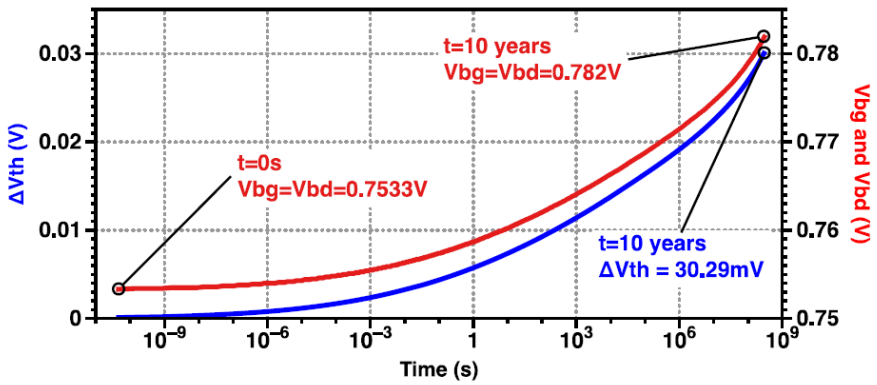

Fig. 5.21 *Evolution of the threshold voltage shift and stress conditions in the input transistor in a pMOS current mirror.*

The relative error in the threshold voltage shift obtained with a fixed number of steps is shown in Figure 5.22. The use of the presented adaptive step size algorithm improves the accuracy in relation to the linear or logarithmic scales. In addition, in this simple case, the adaptive step size algorithm achieves very accurate results using a number of steps around 10 while the other scales needs many more steps to reach the same accuracy.

On the one hand, the adaptive algorithm improves the accuracy to know how transistors are aged, but a more complex circuit is used to evaluate its impact on the circuit performances. A two-stage Miller op-amp shown in Figure 5.23 is selected in this case.

To compare the results obtained with the different scales, Figure 5.24 shows the mean values (calculated with 10,000 samples for the Monte-Carlo analysis) obtained for two op-amp performances, the DC-gain and the unity-gain frequency (*fu*). A reliability simulation using 1,000 steps is used again as the most accurate standard to compare to. Both adaptive scales

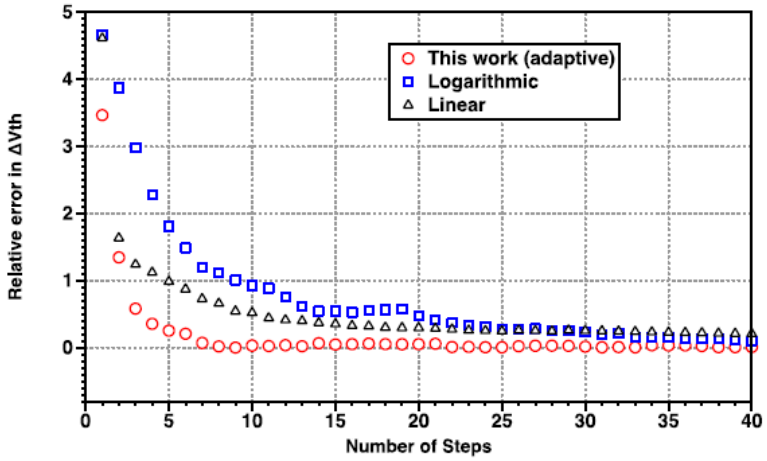

Fig. 5.22 *Relative error in the threshold voltage shift versus the number of steps.*

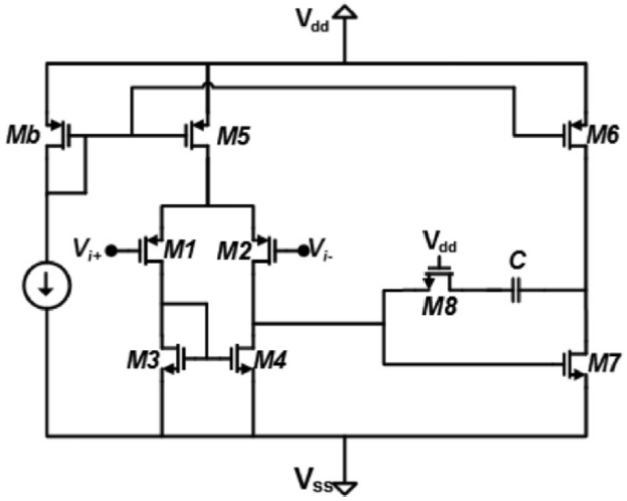

Fig. 5.23 *Schematic of a two-stage Miller op-amp*

(number fixed and accuracy fixed) allows improving the accuracy when the error is very low, especially in DC-gain performance.

Figure 5.24 shows the results obtained using the adaptive step size with fixed accuracy (defined by the parameter *deltaV*). The same number of intermediate steps can be used for different values of *deltaV* when the fixed accuracy is used. Moreover, for a high number of steps, this option does not use all possible values of intermediate steps. This occurs because

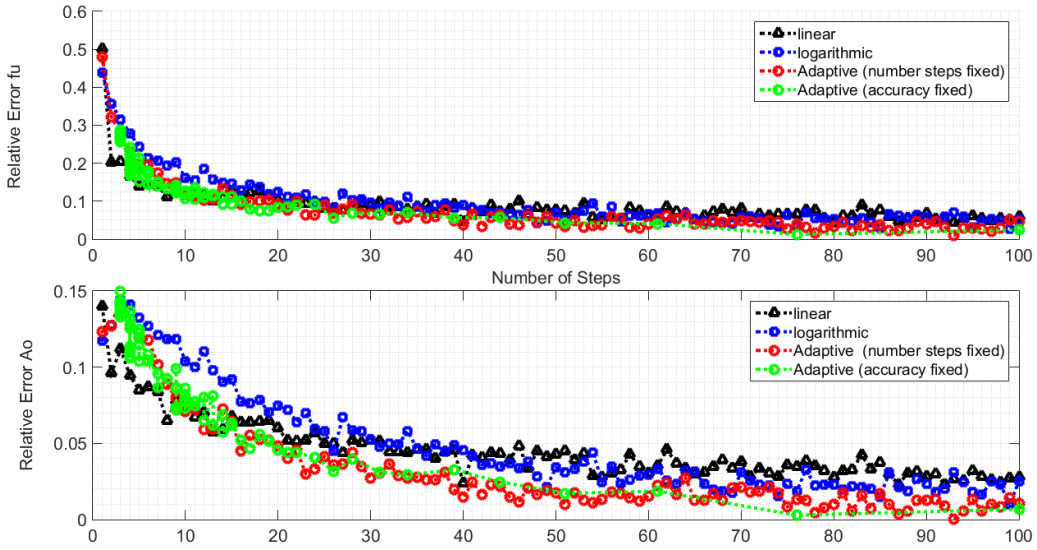

Fig. 5.24 *Relative error against number of steps in the calculation of the performances of the circuit for different scales.*

when the accuracy is fixed, the number of steps is selected by the algorithmic as shown in Figure 5.25. When the fixed accuracy is very high (small threshold voltage shift ()$deltaV$) is selected), the number of steps increases exponentially. However, if the defined accuracy is small, the same number of steps are used for more than one accuracy requirement. Although the number of steps would be the same, each accuracy requirement gets different results for the circuit performances since the times where the stress conditions have been updated can be totally different. Figure 5.25 shows the number of steps used versus the maximum variation allowed of the threshold voltage when the accuracy requirement option is used. In addition to the case discussed previously in Figure 5.25 (Miller op-amp 1, labeled as *miller op-amp 1*), another example is illustrated also in this figure, labeled as *miller op-amp 2*, whose sizing is different than the use in the first example. The number of steps used for the same accuracy requirements in both examples is different which shows how the algorithm adapts the number of steps depending on the specific case. Finally, from the tests and simulations run in this Thesis, a value of $deltaV$ between 0.1 and 1.0 is a convenient option to achieve a good trade-off between the accuracy and the number of steps.

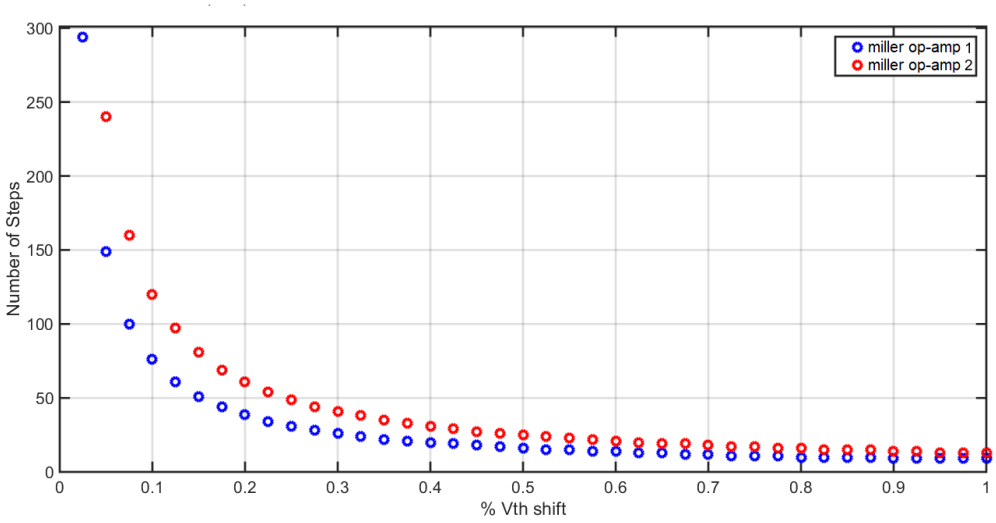

Fig. 5.25 *Number of steps used versus maximum deviation in threshold voltage* V_{th}.

5.3.4 Lifetime calculation

In this scenario of many variability sources affecting the circuit performance over time, a critical issue is the accurate calculation of the circuit lifetime. To do so, the concept of *time-dependent yield* ($TDY(t)$) is introduced,, which is defined as the time-varying percentage of designs that fulfill a set of performance specifications or performance constraints at time t. These design constraints are defined by the designer and usually refer to circuit performances (such as keeping the DC-gain of an op-amp greater than a given value) and to operating conditions (for example that a MOS transistor operates in the saturation region). The lifetime of the circuit is then defined as the time that the circuit is operating with a TDY larger than a certain threshold value (for one or more performance specifications or constraints).

To calculate the yield of a circuit during the design phase, Monte-Carlo simulation is the most commonly used technique. The yield at time zero is then computed by taking the ratio of the number of Monte-Carlo samples that fulfill constraints to the number of samples that do not fulfill those same constraints. It is reasonable to think that accuracy should be a main factor in calculating TDY and lifetime (so as not ending up with over-design, for instance). However, as with many other design tools and methodologies, accuracy comes at the price of computation time. Therefore, whichever the reliability simulation solution used, efficiency, in terms of low CPU time together with the highest level of accuracy, should be a main focus. To emphasize the importance of being efficient in terms of CPU time, it is important to remark

that, in most reported solutions for the computation and optimization of lifetime [64, 66], a reliability simulation is used iteratively and, hence, called in multiple times, so reducing the CPU load is essential.

Though reported solutions for yield and lifetime calculation are based on failure models due the different reliability effects, no published work, to the best of the authors' knowledge, takes into account the stochastic nature of aging phenomena using a stochastic simulator. Nevertheless, some contributions can be mentioned here. The works presented in [66, 67] take into account SV and TDV jointly but, instead of a complete stochastic analysis of the aged distribution of the design, they are only focused on the estimation of worst-case designs (so neither a stochastic distribution of the performance of the circuit at a target time nor the yield is provided). In [68], a single-objective optimization tool is presented. The lifetime is defined as a new constraint and the reliability simulator is used iteratively to evaluate the design candidates of the solution. However, although yield is defined as a percentage of the designs that fulfill the design constraints in a predefined target time in a similar way that this work does, a deterministic simulator is used, so the calculation of the lifetime does not take into account the stochastic nature of aging.

In contrast to the works in the literature, this Thesis focuses on the calculation of the lifetime without introducing any approximation (such as considering only the worst case of a design) to calculate the yield of the circuit during its lifetime. For this purpose, a Monte-Carlo based method to evaluate the performances is required. A trivial solution would be to compute yield and lifetime at each of the M steps of a reliability simulation. However, this can become computationally prohibitive and, therefore, an efficient methodology is required to reach the same accuracy level while improving CPU usage.

The simulation methodology to calculate the lifetime is based on the stochastic reliability simulation flow using the adaptive step size algorithm proposed in section 5.3.3. In the proposed solution, to avoid unnecessary calculations, the designer can define a temporal window, setting a minimum time (T_{min}) and a maximum time (T_{max}), where the analysis of the lifetime is carried out. To set the adaptive step size algorithm, the designer must define a parameter $deltaV$ to specify the accuracy for the adaptive step size algorithm. Besides, the designer defines, as with a regular simulation, the design specifications or design constraints and the number of samples to carry out the Monte-Carlo analysis (MC). To determine the lifetime of the circuit, a minimum TDY (TDY_{min}) is defined, which will be used, at each step of the lifetime calculation, as the value the calculated yield is compared to with.

The complete flow is depicted in Figure 5.26. As an accurate Monte-Carlo analysis ($\approx 1,000$ samples) is very time consuming, the designer has then to define the number of samples for an accurate Monte-Carlo, *MC*, and the reduction factor, *K*, for a reduced Monte-Carlo analysis. Typically, a value of *K* between 10 and 50 is used because this is a typical number of steps between T_{min} and T_{max}. To distinguish both computations, hereinafter two concepts will be used: *yield calculation* (involving a full Monte-Carlo analysis of *MC* samples) and *yield estimation* (with a reduced Monte- Carlo analysis of *MC/K* samples). Figure 5.26 has then two different parts: on the left, the yield is *estimated* and, on the right, the yield and the lifetime are calculated. These two different parts consist of: a first one where a reduced number of samples (MC/K) is used to *estimate* the yield at each step (TDY_i'), and a second one where the yield is calculated using the defined number of samples by the designer (*MC*) and the lifetime is calculated. During the lifetime calculation, if the *estimated* yield (TDY_i') is lower than the minimum yield defined (Y_{min}), a new Monte-Carlo is carry out with the full number of samples (*MC*).

It is important to notice that the lifetime calculation is always done using the number of samples defined by the designer (*MC*) and if the lifetime happens to lie between T_{min} and T_{max}, at least two Monte-Carlo analysis have been carried out. In that case, it is considered that there is a moment in time T_i where the *estimated* yield is lower than the minimum yield. Then a complete reliability analysis using *MC* samples is carried out. If the obtained TDY is lower than the minimum yield defined Y_{min}(that is, both the estimated value and the calculated value lie below the minimum yield), the yield must be evaluated again in the previous time step T_{i-1} using *MC* samples too. If the obtained yield is larger than the minimum one, it means that the lifetime has been found (T_i). On the other hand, if the lifetime is larger than T_{max} or lower than T_{min}, only one Monte- Carlo is carried out at either point in time.

A lifetime reliability analysis has been carried out using the two-stage Miller amplifier defining 0.7 as minimum yield. The moment in time when the yield over time falls below this value, the circuit is considered "dead" and the lifetime equals that time value. The design constraints to compute the yield are: phase margin $> 60^o$, DC-gain $> 50dB$, unity gain frequency $> 45MHz$ and all transistors set on their correct operating regime.

For the first example, the temporal window is established between 1 month and 20 years. The lifetime time calculated by the simulation flow is $2.395297 \cdot 10^7$ seconds ($\sim$9 months and 1 week). Figure 5.27 shows the Monte-Carlo analysis used to calculate the circuit lifetime (red-X if the yield is estimated and blue- X if it is calculated). On the x-axis, the moments in time when the stress conditions have been updated are shown. In this case, T_{min} was set to 30 days ($2.592 \cdot 10^6$ seconds) and for this reason, in the first steps (which have been used to update the

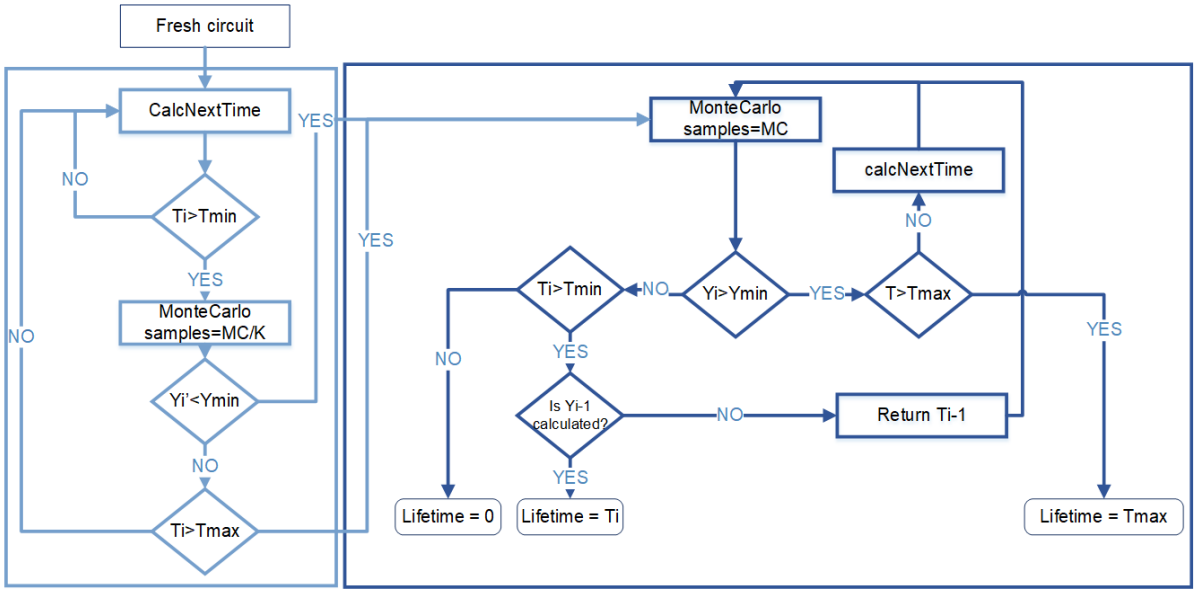

Fig. 5.26 *Diagram used to calculate the circuit lifetime.*

stress conditions before T_{min}) no Monte-Carlo analysis has been carried out since this moment is not considered in the defined temporal window.

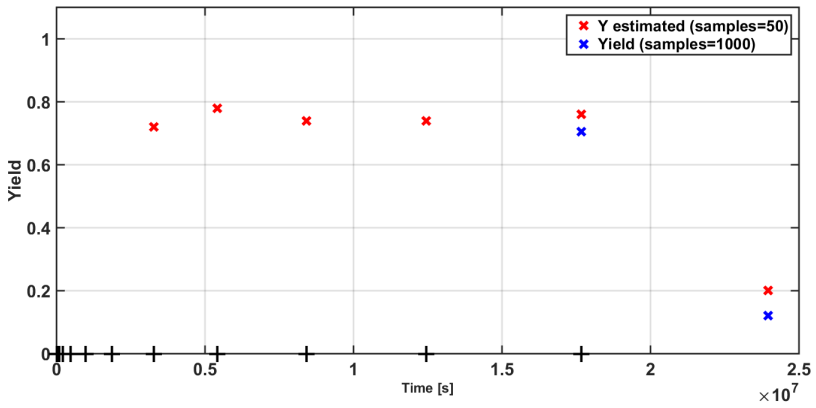

Fig. 5.27 *Example 1 of lifetime calculation.*

Another case study is illustrated in Figure 5.28, where the same circuit topology but a different sizing is used. In this case, the lifetime is $3.591648 \cdot 10^8$ seconds (around 11 years and 4 months). In this case a total of 112 analyses have been needed to *estimate* the yield and 3 full Monte-Carlo analyses have been carried out to calculate the circuit lifetime. This example is also shown to illustrate the importance of the temporal window. In this example, the CPU-time

needed is very high, because T_{min} is equal to the first case (30 days). If this time were longer (e.g., 1 year), the CPU time would have been drastically reduced.

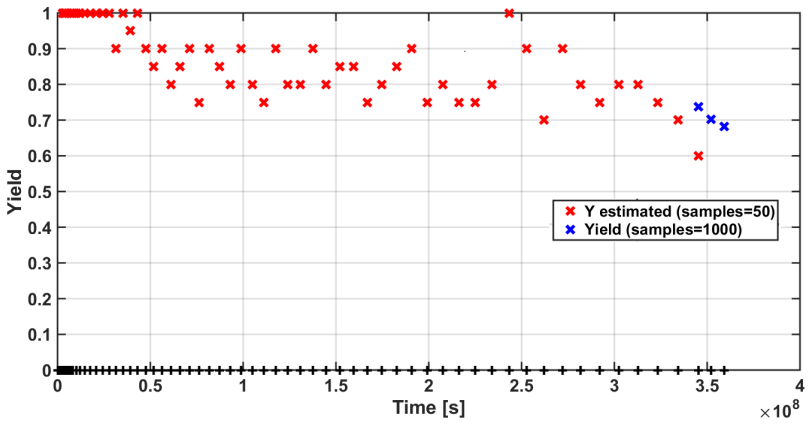

Fig. 5.28 *Example 2 of lifetime calculation.*

These two examples show how the circuit lifetime simulation flow works in different situations, where the lifetime *calculated* is smaller (first case) or larger (second case) than the *estimated*. In the first example, when the *estimated TDY* is lower than the Y_{min}, the TDY is *calculated* and it results in a lower value than Y_{min}.

In this situation, the search of the calculated lifetime is carried out *backwards* in time until the moment that $TDY(t) > Y_{min}$ is found (which occurs, for the first example, in the previous step T_{i-1}). In the second example, when the *estimated TDY* is lower than Y_{min}, the TDY is *calculated* but, now, the result obtained is larger than Y_{min}. In this situation, the circuit is not considered "dead" yet, so the search of the calculated lifetime continues *forward* until $TDY(t) < Y_{min}$, which requires, for this second example, two more steps.

5.4 The reliability simulator CASE

This section presents CASE, a reliability simulator based on the approaches presented in the previous section. In addition, the implementation of an efficient and automated simulation flow and the use of a commercial electrical simulator allow to achieve very accurate results with an affordable CPU time (note that once of our goals is to use this simulator as evaluator in an iterative optimization process). The simulation flow is depicted in Figure 5.29, where the use of an off-the-shelf electrical simulator is needed to evaluate the circuit performance. The inputs of the reliability simulator are contained in two separate files:

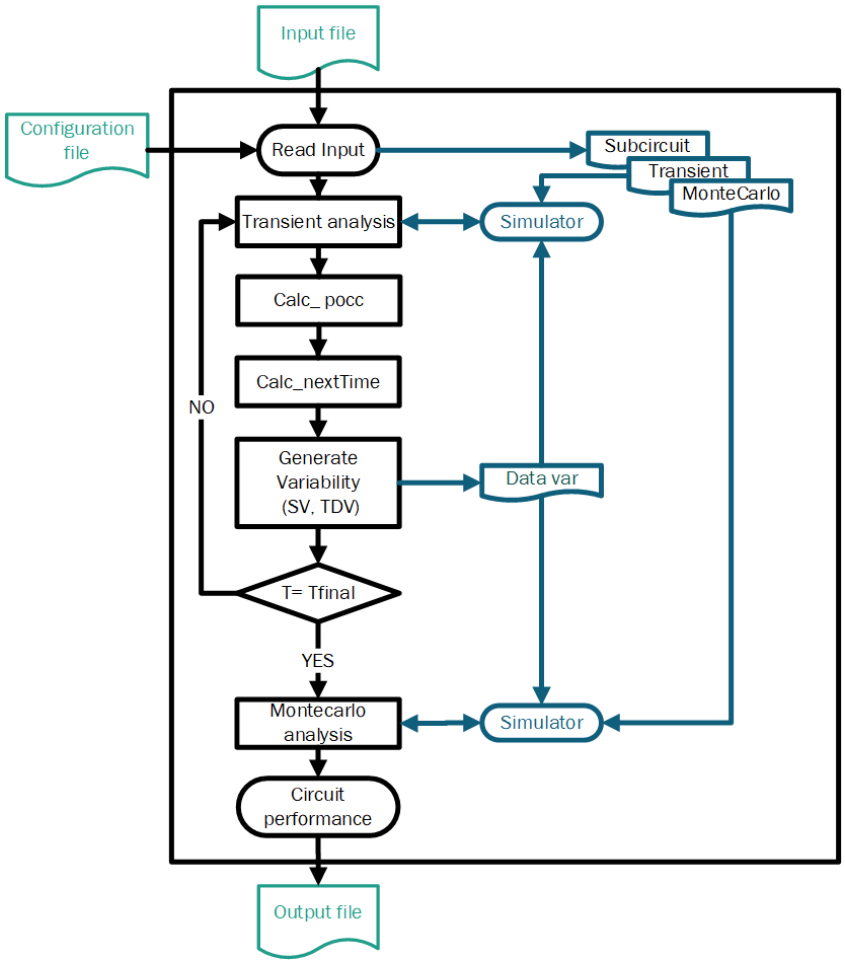

Fig. 5.29 *Simulation flow implemented in CASE.*

- **Configuration file.** It contains a set of parameters to configure the simulation method, such as the number of samples used in Monte Carlo simulation ($N_{samples}$), the time where the variability analysis is done (T_{final}) or the mesh where τ_c-τ_e are defined, that is the maximum, minimum and grid value where the probabilities of defects occupancy *pocc* and the distribution of defects $Ddefect$ are calculated.

- **Input file.** This file contains the user-entered information about the circuit, the simulation test-benches (for the transient and Monte Carlo analyses) and the circuit performance information to be rendered by the simulator.

The flow of this reliability simulator begins by reading the input file, and the three files (Sub-circuit, Transient and Monte-Carlo files, as shown in Figure 5.29) are automatically

generated. The circuit under study is defined in the Sub-circuit file; here, additional devices (such as a voltage source to emulate a threshold voltage degradation (ΔV_{th}) and the parameters to include SV for those devices selected for reliability analysis) are automatically included as shown in Figures 5.30 and 5.31. Transistors whose name begins by "mr" are used in the reliability analysis. In the example in Figure 5.30, a current mirror circuit is defined where transistors $m3$ and $m4$ are not included in the reliability analysis.

```
Circuit{mirror in out

mr1 in   in vdd vdd pmodel      W=W1 L=L1
mr2 n1   in vdd vdd pmodel      W=W2 L=L2
m3 n1    n1 vss vss nmodel    W=W3 L=L3
m4 out   n1 vss vss nmodel    W=W4 L=L4
}
```

Fig. 5.30 *Circuit section in the input file where a current mirror is defined.*

```
.SUBCKT mirror in out
+ vr1 = 0  dvt_r1 = 0 du_r1 = 0
+ vr2 = 0  dvt_r2 = 0 du_r2 = 0

mr1 nmr1  nmr1 vdd vdd pmodel      W=W1 L=L1
+ delvt0=dvt_r1 mulu0=du_r1
Vr1 nmr1  in  vr1
mr2 n1  nmr2 vdd vdd pmodel      W=W2 L=L2
+ delvt0=dvt_r2 mulu0=du_r2
Vr2 nmr2  in  vr2
m3 n1    n1 vss vss nmodel    W=W3 L=L3
m4 out   n1 vss vss nmodel    W=W4 L=L4

.ends
```

Fig. 5.31 *Subcircuit automatically generated by CASE using the circuit defined in Figure 5.30.*

The file for the transient analysis to obtain the stress suffered by each selected device, and the file to perform a Monte-Carlo simulation of the aged circuit is also generated and the sub-circuit is automatically included. In addition, these files are prepared to perform a Monte Carlo analysis as shown in Figure 5.33 using the test-bench defined in the MonteCarlo section of the input file defined in Figure 5.32.

```
MonteCarlo{nameMC

.....
xin in out mirror
.....

}
```

Fig. 5.32 *MonteCarlo section in the input file where a current mirror is defined.*

```
MonteCarlo{nameMC

.....
xin in out mirror
.....

}
```

Fig. 5.33 *File automatically generated using CASE with information defined in Figure 5.32.*

When the files are generated, at each intermediate step (the first is time zero ($t = 0$)), a transient analysis is carried out including the transistor degradation defined in the file *Data_var* (as shown in Figure 5.29).

Finally, at T_{final}, a Monte-Carlo analysis is carried out. For this, the complete variability information is written in the file *"Data_var"*. The circuit performances defined in the input file are read, each performance can be defined as a design constraint as shown Figure 5.34, where P1 has to be greater than X, P2 smaller than Y and P3 has to be greater than 80% of the nominal design value for this performance.

```
Performance{
P1 > X
P2 < Y
P3 >% 80
P4 max
}
```

Fig. 5.34 *Performance section in input file.*

With the simulation method described, it is then possible to carry out a thorough statistical analysis of the aged circuit, with which the designer can accurately assess the reliability of the circuit. Statistical values for each specified performance are calculated as shown in Figure 5.35. In addition, the yield over time or time dependent yield ($TDY(T_{final})$) is calculated, that is, the ratio of the number of samples that attain the design specification or constraints defined in section performances after a period of full-time operation T_{final}.

```
Measure Nominal    Mean     StdDeviation  Minimum    Maximum    Yield(YNom.)
fcopy  1.065e+00  1.058e+00  2.822e 02     9.640e 01  1.182e+00  1.000


YIELD= 1.000
Y Nominal= 0.410
```

Fig. 5.35 *Output file example.*

In addition, CASE includes the simulation flow to calculate lifetime automatically. CASE is coded in C language and it is easily used in a automated design methodology due to the possibility of working in *batch-mode*.

5.5 Summary

This Chapter presents a a reliability simulation flow that takes into account the stochastic nature of aging, the joint inclusion of SV and TDV and the much needed update of stress conditions over time. A tool named CASE has been implemented to fully automate that reliability simulation flow. CASE wraps around a commercial simulator to carry out the essential electrical analyses that are needed in the reliability simulation flow. CASE is versatile in the sense that it can carry out these analyses with a variety of options, like, for instance, the use of an adaptive algorithm to set the intermediate steps in time (selecting either a desired accuracy level or a CPU time budget), or the capability to evaluate the impact of either SV, TDV, or both, on the circuit performances at a specified time T_{final}. With the help of CASE, designers can efficiently account for the impact of process variability and aging in the design process.

CASE can calculate the circuit performance after a period of full-time operation, calculating the time dependent yield (TDY) at this time. In addition, CASE can calculate the circuit lifetime,

that is, the time when a proportion, larger than a certain threshold (Y_{min}), of the designs stops fulfilling constraints ($TDY(t) < Y_{min}$).

In addition, the reliability simulation flow and CASE have been developed such a proper trade-off between accuracy and CPU time is achieved. This trade-off is at the base of the adaptive step size algorithm as well as the technique to simultaneously include both statistical distributions of SV and TDV.

Chapter 7

Conclusions

Electronic design automation (EDA) tools for analog circuits are far, in terms of impact, widespread use and evolution, from EDA tools for digital circuits. One of the main reasons behind this lag is that analog design automation is a quite challenging problem due to the many sources of perturbation and the difficulty in systematically dealing with them. It is in this field of design methodologies for analog and mixed signal circuits, where this Thesis has presented a contribution.

A set of design flows are proposed in this thesis to systematically include two very relevant sources of perturbations in the generation of the Pareto-Optimal fronts (POFs) of analog circuits. These fronts capture the best trade-offs between a set of conflicting circuit performances and are also a the key element in Multi-Objective Bottom-Up (MOBU) design methodologies for analog circuits. One of the most successful ways to generate a POF is through the combination of an optimization engine and a circuit evaluator. This Thesis proposes the accurate inclusion of layout parasitics and unreliability effects within the iterative loop of an optimization process to account for these perturbations in the generation of POFs.

First, a POF-based layout-aware design flow is proposed, which combines the layout-aware design advantages with the use of POFs. The main benefit that stems from this proposal is that the time-consuming iterations between circuit sizing and layout generation are removed due to the inclusion of the physical implementation within the electrical design. On the one hand, a template-based technique has been used to automate the layout generation. A geometric constraint module, which uses a modified Stockmeyer's algorithm to minimize the layout area, has been developed to optimize the geometry of the physical implementation. On the other hand, commercial tools are used to extract the parasitics and to evaluate their impact on the circuit performances.

A set of case studies has demonstrated the proposed design flow. For this purpose, a two-stage fully differential, Miller-compensated CMOS operational amplifier in a $0.35\mu m$, 4-metal, CMOS technology has been considered. Three different types of POFs were considered in order to compare the impact of the parasitic devices. It is concluded that only by including all parasitic information in the simulation that the iterations between electrical and physical design can be avoided.

In order to improve the proposed layout-aware design flow, a method to assess the quality of the generated layouts (based on templates) has been proposed. Using this method and the cases studies using a single layout template, a library of templates has been designed. This library of layout templates has been included in the layout-aware design flow. For this purpose, a new tool (Minimum Area Selector Stocmeyer's Algorithm (MASSA)) that can decide which layout template is most adequate from the geometric and electrical point of view, has been developed.

Device variability due to the imperfections of the fabrication process and aging has an important impact on the circuit reliability. To evaluate this impact, a new reliability simulator (CASE) has been developed. CASE implements a fully automated simulation flow by embedding a commercial circuit simulator (Hspice). This new reliability simulator is the first that takes into account the stochastic nature of aging and the joint inclusion of the stochastic spatial variability (SV) and stochastic time-dependent variability (TDV). The bi-directional link between stress conditions and aging calls for the use of intermediate steps in order to update the stress conditions. In this sense, this Thesis provides a new adaptive step size algorithm that also reduce the CPU time efficiently while updating the stress conditions following user-defined directives. The simulator CASE can be used: (1) to calculate the circuit performance after a period of full-time operation; and (2) to calculate the circuit lifetime.

Using CASE, the circuit lifetime has been added as an optimization objective. For this purpose, a new evaluation process is proposed to efficiently generate Lifetime-Aware POFs. This allows generating Reliability-Aware POFs, where the impact of SV and TDV on the circuit performances is included in the POF generation.

In summary, a set of design flows has been proposed in this Thesis to automatically include important and relevant sources of perturbation, layout parasitics and spatial and time-dependent variability, during the generation of analog POFs.

References

[1] G. E. Moore, "Cramming More Components onto Integrated Circuits," *Electronics*, pp. 114–117, 1965.

[2] D. Baker and E. Herr, "Parasitic effects in microelectronic circuits," *IEEE Transactions on Electron Devices*, vol. 12, pp. 161–167, apr 1965.

[3] M. Pelgrom, A. Duinmaijer, and A. Welbers, "Matching properties of MOS transistors," *IEEE Journal of Solid-State Circuits*, vol. 24, pp. 1433–1439, oct 1989.

[4] W. Zhao, F. Liu, K. Agarwal, D. Acharyya, S. R. Nassif, K. J. Nowka, and Y. Cao, "Rigorous Extraction of Process Variations for 65-nm CMOS Design," *IEEE Transactions on Semiconductor Manufacturing*, vol. 22, pp. 196–203, feb 2009.

[5] R. Rao, A. Devgan, D. Blaauw, and D. Sylvester, "Analytical yield prediction considering leakage/performance correlation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, pp. 1685–1695, sep 2006.

[6] M. B. Yelten, P. D. Franzon, and M. B. Steer, "Comparison of modeling techniques in circuit variability analysis," *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, vol. 25, pp. 288–302, may 2012.

[7] S. R. Nassif, "Technology modeling and characterization beyond the 45nm node," in *2008 Asia and South Pacific Design Automation Conference*, pp. 219–219, IEEE, jan 2008.

[8] Y. Miura and Y. Matukura, "Investigation of Silicon-Silicon Dioxide Interface Using MOS Structure," *Japanese Journal of Applied Physics*, vol. 5, pp. 180–180, feb 1966.

[9] K. O. Jeppson and C. M. Svensson, "Negative bias stress of MOS devices at high electric fields and degradation of MNOS devices," *Journal of Applied Physics*, vol. 48, pp. 2004–2014, may 1977.

[10] M. Alam, "A critical examination of the mechanics of dynamic NBTI for PMOSFETs," in *IEEE International Electron Devices Meeting 2003*, pp. 14.4.1–14.4.4, IEEE.

[11] S. Chakravarthi, A. Krishnan, V. Reddy, C. Machala, and S. Krishnan, "A comprehensive framework for predictive modeling of negative bias temperature instability," in *2004 IEEE International Reliability Physics Symposium. Proceedings*, pp. 273–282, IEEE.

[12] M. Alam, H. Kufluoglu, D. Varghese, and S. Mahapatra, "A comprehensive model for PMOS NBTI degradation: Recent progress," *Microelectronics Reliability*, vol. 47, no. 6, pp. 853–862, 2007.

[13] B. Kaczer, V. Arkbipov, R. Degraeve, N. Collaert, G. Groeseneken, and M. Goodwin, "Disorder-controlled-kinetics model for negative bias temperature instability and its experimental verification," in *2005 IEEE International Reliability Physics Symposium, 2005. Proceedings. 43rd Annual.*, pp. 381–387, IEEE.

[14] T. Grasser, W. Gos, and B. Kaczer, "Dispersive Transport and Negative Bias Temperature Instability: Boundary Conditions, Initial Conditions, and Transport Models," *IEEE Transactions on Device and Materials Reliability*, vol. 8, pp. 79–97, mar 2008.

[15] T. Grasser and B. Kaczer, "Evidence That Two Tightly Coupled Mechanisms Are Responsible for Negative Bias Temperature Instability in Oxynitride MOSFETs," *IEEE Transactions on Electron Devices*, vol. 56, pp. 1056–1062, may 2009.

[16] D. Ielmini, M. Manigrasso, F. Gattel, and G. Valentini, "A unified model for permanent and recoverable NBTI based on hole trapping and structure relaxation," in *2009 IEEE International Reliability Physics Symposium*, pp. 26–32, IEEE, 2009.

[17] J. Martin-Martinez, B. Kaczer, M. Toledano-Luque, R. Rodriguez, M. Nafria, X. Aymerich, and G. Groeseneken, "Probabilistic defect occupancy model for NBTI," in *2011 International Reliability Physics Symposium*, pp. XT.4.1–XT.4.6, IEEE, apr 2011.

[18] E. Takeda, N. Suzuki, and T. Hagiwara, "Device performance degradation to hot-carrier injection at energies below the Si-SiO2energy barrier," in *1983 International Electron Devices Meeting*, pp. 396–399, IRE, 1983.

[19] Chenming Hu, Simon C. Tam, Fu-Chieh Hsu, Ping-Keung Ko, Tung-Yi Chan, and K. Terrill, "Hot-Electron-Induced MOSFET Degradation - Model, Monitor, and Improvement," *IEEE Journal of Solid-State Circuits*, vol. 20, pp. 295–305, feb 1985.

[20] Wenping Wang, V. Reddy, A. Krishnan, R. Vattikonda, S. Krishnan, and Yu Cao, "Compact Modeling and Simulation of Circuit Reliability for 65-nm CMOS Technology," *IEEE Transactions on Device and Materials Reliability*, vol. 7, pp. 509–517, dec 2007.

[21] W. Shockley, "Problems related to p-n junctions in silicon," *Solid-State Electronics*, vol. 2, pp. 35–67, jan 1961.

[22] C. Guerin, V. Huard, and A. Bravaix, "The Energy-Driven Hot-Carrier Degradation Modes of nMOSFETs," *IEEE Transactions on Device and Materials Reliability*, vol. 7, pp. 225–235, jun 2007.

[23] A. Bravaix, C. Guerin, V. Huard, D. Roy, J. Roux, and E. Vincent, "Hot-Carrier acceleration factors for low power management in DC-AC stressed 40nm NMOS node at high temperature," in *2009 IEEE International Reliability Physics Symposium*, pp. 531–548, IEEE, 2009.

[24] P. Magnone, F. Crupi, N. Wils, R. Jain, H. Tuinhout, P. Andricciola, G. Giusi, and C. Fiegna, "Impact of Hot Carriers on nMOSFET Variability in 45- and 65-nm CMOS Technologies," *IEEE Transactions on Electron Devices*, vol. 58, pp. 2347–2353, aug 2011.

[25] S. Sahhaf, R. Degraeve, P. J. Roussel, B. Kaczer, T. Kauerauf, and G. Groeseneken, "A New TDDB Reliability Prediction Methodology Accounting for Multiple SBD and Wear Out," *IEEE Transactions on Electron Devices*, vol. 56, pp. 1424–1432, jul 2009.

[26] J. Martin-Martinez, B. Kaczer, R. Degraeve, P. J. Roussel, R. Rodriguez, M. Nafria, X. Aymerich, B. Dierickx, and G. Groeseneken, "Circuit Design-Oriented Stochastic Piecewise Modeling of the Postbreakdown Gate Current in MOSFETs: Application to Ring Oscillators," *IEEE Transactions on Device and Materials Reliability*, vol. 12, pp. 78–85, mar 2012.

[27] A. Sangiovanni-Vincentelli, L. Carloni, F. De Bernardinis, and M. Sgroi, "Benefits and challenges for platform-based design," in *Proceedings of the 41st annual conference on Design automation - DAC '04*, (New York, New York, USA), p. 409, ACM Press, 2004.

[28] H. Chang, *Surviving the soc revolution : a guide to platform -based design.* Springer-Verlag New York, 2013.

[29] M. Velasco-Jiménez, R. Castro-López, E. Roca, and F. V. Fernández, "Design Space Exploration Using Hierarchical Composition of Performance Models,"

[30] T. Eeckelaert, T. McConaghy, and G. Gielen, "Efficient Multiobjective Synthesis of Analog Circuits using Hierarchical Pareto-Optimal Performance Hypersurfaces," in *Design, Automation and Test in Europe*, pp. 1070–1075, IEEE.

[31] T. Eeckelaert, R. Schoofs, G. Gielen, M. Steyaert, and W. Sansen, "Hierarchical bottom–up analog optimization methodology validated by a delta–sigma A/D converter design for the 802.11a/b/g standard," in *Proceedings of the 43rd annual conference on Design automation - DAC '06*, (New York, New York, USA), p. 25, ACM Press, 2006.

[32] G. Stehr, H. E. Graeb, and K. J. Antreich, "Analog Performance Space Exploration by Normal-Boundary Intersection and by Fourier–Motzkin Elimination," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, pp. 1733–1748, oct 2007.

[33] M. Velasco-Jimenez, R. Castro-Lopez, E. Roca, and F. V. Fernandez, "Design space exploration using hierarchical composition of performance models," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1941–1944, IEEE, may 2015.

[34] R. Castro-Lopez, O. Guerra, E. Roca, and F. V. Fernandez, "An Integrated Layout-Synthesis Approach for Analog ICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, pp. 1179–1189, jul 2008.

[35] R. Castro-Lopez, *Reuse-based methodologies and tools in the design of analog and mixed-signal integrated circuits*. Springer, 2006.

[36] P. Vancorenland, G. Van der Plas, M. Steyaert, G. Gielen, and W. Sansen, "A layout-aware synthesis methodology for RF circuits," in *IEEE/ACM International Conference on Computer Aided Design. ICCAD 2001. IEEE/ACM Digest of Technical Papers (Cat. No.01CH37281)*, pp. 358–362, IEEE.

[37] M. Dessouky and M.-M. Louerat, "A layout approach for electrical and physical design integration of high-performance analog circuits," in *Proceedings IEEE 2000 First International Symposium on Quality Electronic Design (Cat. No. PR00525)*, pp. 291–298, IEEE Comput. Soc.

[38] H. Onodera, H. Kanbara, and K. Tamaru, "Operational-amplifier compilation with performance optimization," *IEEE Journal of Solid-State Circuits*, vol. 25, pp. 466–473, apr 1990.

[39] M. Ranjan, W. Verhaegen, A. Agarwal, H. Sampath, R. Vemuri, and G. Gielen, "Fast, layout-inclusive analog circuit synthesis using pre-compiled parasitic-aware symbolic performance models," in *Proceedings Design, Automation and Test in Europe Conference and Exhibition*, pp. 604–609, IEEE Comput. Soc.

[40] R. Martins, N. Lourenco, A. Canelas, R. Povoa, and N. Horta, "AIDA: Robust layout-aware synthesis of analog ICs including sizing and layout generation," in *2015 International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, pp. 1–4, IEEE, sep 2015.

[41] R. Martins, N. Lourenco, S. Rodrigues, J. Guilherme, and N. Horta, "AIDA: Automated analog IC design flow from circuit level to layout," in *2012 International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, pp. 29–32, IEEE, sep 2012.

[42] T. Christiansen, L. Wall, J. Orwant, and B. D. Foy, *Programming Perl*. O'Reilly, 2012.

[43] B. W. Kernighan and R. Pike, "Using the shell," in *The UNIX Programming Environment*, ch. 3, pp. 71–101, Prentice Hall, 1984.

[44] Cadence Design Systems, Inc., *OCEAN Reference*, 2001.

[45] Cadence Design Systems, Inc., *Cadence Interprocess Communication SKILL Reference*.

[46] L. Stockmeyer, "Optimal orientations of cells in slicing floorplan designs," *Information and Control*, vol. 57, pp. 91–101, may 1983.

[47] H. Koh, C. Sequin, and P. Gray, "OPASYN: a compiler for CMOS operational amplifiers," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 9, no. 2, pp. 113–125, 1990.

[48] J. Conway and G. Schrooten, "An automatic layout generator for analog circuits," in *[1992] Proceedings The European Conference on Design Automation*, pp. 513–519, IEEE Comput. Soc. Press.

[49] Cadence Design Systems, Inc., *Cadence las referencias las buscaré el finde*.

[50] T. Barnes, "SKILL: a CAD system extension language," in *27th ACM/IEEE Design Automation Conference*, pp. 266–271, IEEE.

[51] Cadence Design Systems, Inc., *Diva® Reference*, 2005.

[52] Cadence Design Systems, Inc., *Assura® Parasitic Extraction*, 2005.

[53] H. Habal and H. Graeb, "Constraint-Based Layout-Driven Sizing of Analog Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, pp. 1089–1102, aug 2011.

[54] N. Lourenço, R. Martins, and N. Horta, "Layout-aware sizing of analog ics using floorplan & routing estimates for parasitic extraction," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, pp. 1156–1161, EDA Consortium, 2015.

[55] R. Martins, A. Canelas, N. Lourenco, and N. Horta, "On-the-fly exploration of placement templates for analog IC layout-aware sizing methodologies," in *2016 13th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, pp. 1–4, IEEE, jun 2016.

[56] S. K. Saha, "Compact MOSFET Modeling for Process Variability-Aware VLSI Circuit Design," *IEEE Access*, vol. 2, pp. 104–115, 2014.

[57] R. Tu, E. Rosenbaum, W. Chan, C. Li, E. Minami, K. Quader, P. Ko, and C. Hu, "Berkeley reliability tools-BERT," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 10, pp. 1524–1534, 1993.

[58] E. Maricau and G. Gielen, "Background on IC Reliability Simulation," in *Analog IC Reliability in Nanometer CMOS*, pp. 79–91, New York, NY: Springer New York, 2013.

[59] M. Moras, J. Martin-Martinez, V. Velayudhan, R. Rodriguez, M. Nafria, X. Aymerich, and E. Simoen, "Negative bias temperature instabilities in pMOSFETS: Ultrafast characterization and modelling," in *2015 10th Spanish Conference on Electron Devices (CDE)*, pp. 1–4, IEEE, feb 2015.

[60] A. Avellán, D. Schroeder, and W. Krautschneider, "Modeling random telegraph signals in the gate current of metal–oxide–semiconductor field effect transistors after oxide breakdown," *Journal of Applied Physics*, vol. 94, pp. 703–708, jul 2003.

[61] B. Kaczer, P. J. Roussel, T. Grasser, and G. Groeseneken, "Statistics of Multiple Trapped Charges in the Gate Oxide of Deeply Scaled MOSFET Devices—Application to NBTI," *IEEE Electron Device Letters*, vol. 31, pp. 411–413, may 2010.

[62] http://ptm.asu.edu/, "Arizona state university predictive technology model."

[63] A. Zhang, C. Huang, T. Guo, A. Chen, S. Guo, R. Wang, R. Huang, and J. Xie, "Reliability variability simulation methodology for IC design: An EDA perspective," in *2015 IEEE International Electron Devices Meeting (IEDM)*, pp. 11.5.1–11.5.4, IEEE, dec 2015.

[64] E. Afacan, G. Berkol, G. Dundar, A. E. Pusane, and F. Baskaya, "A deterministic aging simulator and an analog circuit sizing tool robust to aging phenomena," in *2015 International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, pp. 1–4, IEEE, sep 2015.

[65] E. Afacan, G. Berkol, A. E. Pusane, G. Dundar, and F. Baskaya, "Adaptive sized Quasi-Monte Carlo based yield aware analog circuit optimization tool," in *2014 5th European Workshop on CMOS Variability (VARI)*, pp. 1–6, IEEE, sep 2014.

[66] X. Pan and H. Graeb, "Reliability optimization of analog integrated circuits considering the trade-off between lifetime and area," *Microelectronics Reliability*, vol. 52, no. 8, pp. 1559–1564, 2012.

[67] X. Pan and H. Graeb, "Reliability analysis of analog circuits using quadratic lifetime worst-case distance prediction," in *IEEE Custom Integrated Circuits Conference 2010*, pp. 1–4, IEEE, sep 2010.

[68] E. Afacan, G. Berkol, G. Dundar, A. E. Pusane, and F. Baskaya, "A lifetime-aware analog circuit sizing tool," *Integration, the VLSI Journal*, vol. 55, pp. 349–356, sep 2016.

[69] Seobin Jung, Jiho Lee, and Jaeha Kim, "Yield-Aware Pareto Front Extraction for Discrete Hierarchical Optimization of Analog Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, pp. 1437–1449, oct 2014.

[70] G. Berkol, E. Afacan, G. Dundar, A. E. Pusane, and F. Baskaya, "A novel yield aware multi-objective analog circuit optimization tool," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2652–2655, IEEE, may 2015.

[71] B. Liu, F. V. Fernandez, and G. G. E. Gielen, "Efficient and Accurate Statistical Analog Yield Optimization and Variation-Aware Circuit Sizing Based on Computational Intelligence Techniques," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, pp. 793–805, jun 2011.

[72] F. V. Fernandez, J. Esteban-Muller, E. Roca, and R. Castro-Lopez, "Stopping criteria in evolutionary algorithms for multi-objective performance optimization of integrated inductors," in *IEEE Congress on Evolutionary Computation*, pp. 1–8, IEEE, jul 2010.